aws

User Guide

AWS Elastic Disaster Recovery



Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

AWS Elastic Disaster Recovery: User Guide

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

Table of Contents

What is Elastic Disaster Recovery? 1
Getting started 2
Sign up for an AWS account 2
Disaster recovery overview 2
Resources 3
Initialization and permissions 3
Initializing AWS Elastic Disaster Recovery 3
Additional policies 4
Initializing DRS through the API 4
Programmatically initializing DRS 8
Accessing the console 10
Supported AWS Regions 10
Using the console 13
Source servers page 13
Best practices 15
Planning 15
Drilling 15
Monitoring 16
Limits 16
Protecting Point-In-Time snapshots 16
Controlling agent installation permissions 17
Recovery best practices 17
Failback best practices 19
Security best practices 19
Disaster recovery at scale 20
Account and Region planning 20
Network planning and benchmarking 20
Storage benchmarking 21
Agent deployment at scale 21
DR readiness and compliance monitoring 22
Service quotas and API limits 22
Recovery planning at scale 22
Quick start guide 23
First time setup 23

Adding source servers ... 27
Configuring launch settings ... 27
Launching a drill instance ... 28
Launching a recovery instance ... 29
Performing a failback ... 30
Replication network requirements ... 31
Network diagrams ... 31
General Architecture - On-Premises to AWS ... 32
On-Prem to AWS ... 32
AWS Cloud to AWS Cloud via VPC Peering ... 33
On-Prem to Outposts ... 34
AWS to Outposts ... 35
On-Premises to AWS Local Zone ... 35
AWS Local Zone to Region ... 36
AWS Local Zone to AWS Local Zone ... 37
AWS Failback to On-Prem ... 38
Network setting preparations ... 39
Staging area subnet ... 40
Network requirements ... 40
Operational subnets ... 40
Network requirements ... 41
Communication over TCP port 443 ... 41
Communication between the source servers and Elastic Disaster Recovery over TCP port 443 ... 43
Communication between the staging area subnet and AWS Elastic Disaster Recovery over TCP port 443 ... 47
Communication between the source servers and the Staging Area Subnet over TCP port 1500 ... 48
AWS DRS settings ... 49
Replication settings ... 49
Default replication ... 50
Individual replication settings ... 51
Launch settings ... 65
Default AWS DRS launch settings ... 65
Default Amazon Elastic Compute Cloud (Amazon EC2) launch template ... 71
Post-launch actions ... 74

Install the required IAM roles if needed ... 77
Activating post-launch actions default settings ... 77
Adding custom actions ... 77
Activating, deactivating and editing predefined or custom actions ... 79
Deleting custom actions ... 81
Predefined post-launch actions ... 82
Validate disk space ... 85
Amazon EC2 connectivity checks ... 85
Verify HTTP/HTTPS response ... 85
Verify tags ... 85
Source servers ... 86
Adding source servers to AWS DRS ... 86
Installation requirements ... 87
Supported Windows operating systems ... 92
Supported Linux operating systems ... 96
Installing the AWS Replication Agent ... 103
Adding instances from the Amazon EC2 Console ... 173
Source servers page ... 176
Interacting with the Source Servers page ... 177
Command menus ... 179
Filtering ... 181
Server details ... 181
Recovery dashboard ... 184
Server info ... 189
Tags ... 190
Disk settings ... 191
Replication settings ... 192
AWS DRS launch settings ... 192
Post-launch settings ... 193
Source networks ... 199
Source network page ... 199
Adding source networks ... 200
Installing the AWS Replication Agent ... 201
Creating the required role ... 201
Replicating your network configurations in Elastic Disaster Recovery ... 203
Trusted accounts ... 205

Trusted account page ... 205
Adding a trusted account ... 206
Creating the Staging role ... 206
Creating the Network role ... 207
Creating the Failback and in-AWS right-sizing roles ... 208
Configuring launch settings ... 211
Preparing for drill and recovery instance launch ... 211
Launch settings ... 212
DRS launch settings ... 213
DRS launch settings parameters ... 213
EC2 launch template ... 218
EC2 launch template parameters ... 220
Key considerations for EC2 launch templates ... 223
Full launch template setting review ... 224
Flexible Instance Types ... 228
How it works ... 228
How to use ... 229
Attribute selection ... 229
Using Elastic Disaster Recovery for recovery and failback ... 231
Key terminology ... 231
Recovery and Failback overview ... 232
Understanding recovery ... 232
Understanding failback ... 232
Preparing for recovery ... 233
Validate launch settings ... 233
Recovery drill overview ... 234
Recovery drill objectives ... 234
Performing recovery drills ... 234
Post recovery drill actions ... 236
Recovery plans ... 237
Recovery plan concepts ... 238
How a recovery plan runs ... 239
Validation ... 240
Controlling how server failures affect a plan ... 241
Creating a recovery plan ... 242
Adding and ordering steps ... 243

Running a recovery plan ... 245
Monitoring an execution ... 249
Retrying, skipping, and canceling ... 251
Editing and deleting recovery plans ... 253
Recovery plan quotas ... 254
Performing a failover ... 254
Launching recovery instances ... 255
Performing a failback ... 256
Failback to on-premises environment ... 257
Performing a cross-Region failback ... 280
Performing a cross-account failback ... 290
Cross Availability Zone recovery ... 300
Cross Availability Zone (AZ) setup ... 300
Recovery Instances page ... 304
Recovery instances overview ... 304
Monitoring recovery instances ... 304
Recovery instance categories ... 304
Recovery instances actions ... 307
Recovery instance details view ... 309
Launch dashboard ... 310
Instance information ... 311
Tags ... 312
Failback replication settings ... 312
Post-launch actions status ... 314
Recovery job history ... 315
Recovery job history ... 315
Overview ... 316
Job Details ... 319
Using multiple staging accounts with AWS DRS ... 324
Overview ... 324
Extending source servers from a staging account into a target AWS account ... 325
Onboarding a new staging account ... 326
Using an existing account as a staging account ... 327
Share the EBS encryption key with the target account ... 328
Managing extended source servers within the target AWS account ... 329
Initializing the target account ... 329

Create extended source servers ... 329
Manage source servers ... 330
Removing an extended source server ... 331
Troubleshooting ... 332
Working with AWS DRS and AWS Outposts ... 333
Default Replication Settings ... 333
Source Server Replication Settings ... 192
Default Launch Template ... 338
Source Server Launch Templates ... 341
Source Server Page ... 343
Important Outpost Notes ... 344
Outpost Storage ... 344
Replication and Launch Subnets ... 344
Instance Types and Operating Systems (OSs) ... 344
Monitoring ... 345
Security ... 346
Overview ... 346
Identity and access management ... 347
Federated identity ... 347
Authenticating with identities ... 347
Grant permission to tag resources during creation ... 348
AWS managed policies ... 350
Managing access using policies ... 374
Using service-linked roles ... 408
Policy structure ... 410
Resilience ... 411
Infrastructure security ... 411
AWS GovCloud ... 413
Compliance validation ... 413
Cross-service confused deputy prevention ... 414
Monitoring ... 416
Logging AWS Elastic Disaster Recovery API calls using AWS CloudTrail ... 416
AWS Elastic Disaster Recovery information in CloudTrail ... 416
Understanding AWS Elastic Disaster Recovery log file entries ... 417
CloudWatch Metrics for DRS ... 418
Alarm events and EventBridge ... 419

Sample events for Elastic Disaster Recovery ... 419
Registering event rules ... 423
Troubleshooting ... 426
Troubleshooting agent installation ... 426
Agent logs and diagnostics ... 427
Agent installation prerequisites ... 430
Linux installation errors ... 433
Windows installation errors ... 445
Common agent installation errors ... 449
Troubleshooting replication errors ... 458
Verifying network connectivity ... 459
Agent communication errors ... 464
Replication infrastructure errors ... 470
Replication performance errors ... 473
Agent runtime errors ... 477
Bandwidth requirements ... 485
Troubleshooting launch errors ... 508
Conversion errors ... 508
Launch configuration errors ... 511
Windows post-launch issues ... 518
Troubleshooting failback errors ... 519
Failback Client errors ... 520
Failback replication ... 523
FAQ ... 527
What source infrastructure does AWS Elastic Disaster Recovery support? ... 527
Can I upgrade from CloudEndure Disaster Recovery to AWS Elastic Disaster Recovery? ... 527
Can AWS Elastic Disaster Recovery protect physical servers? ... 527
What data is stored on and transmitted through AWS Elastic Disaster Recovery servers? ... 527
What is the Recovery Time Objective (RTO) of AWS Elastic Disaster Recovery? ... 528
What is the Recovery Point Objective (RPO) of AWS Elastic Disaster Recovery? ... 528
What to consider when replicating Active Directory ... 528
Does AWS Elastic Disaster Recovery work with LVM and RAID configurations? ... 528
Do Replication Servers have the SSM agent installed on them? ... 529
What is there to note regarding SAN/NAS Support? ... 529
Does AWS Elastic Disaster Recovery support Windows License Migration? ... 529
Can you perform an OS (Operating System) upgrade with AWS Elastic Disaster Recovery? ... 529

What are the private APIs used by AWS DRS to define actions in the IAM Policy? 530
What post-launch scripts does AWS Elastic Disaster Recovery support? 531
Is BitLocker encryption supported? 531
Can I set instance metadata on my launched instance to support IMDSv2 only? 531
Elastic Disaster Recovery Concepts 532
What is the Recovery Time Objective (RTO) of Elastic Disaster Recovery? 532
What is the Recovery Point Objective (RPO) of Elastic Disaster Recovery? 533
What are Point in Time Snapshots? 534
Agent related 537
What does the AWS Replication Agent do? 537
What kind of data is transferred between the Agent and the AWS Elastic Disaster Recovery Service Manager? 537
Can a proxy server be used between the source server and the Elastic Disaster Recovery Console? 538
What are the pre-requisites needed to install the AWS Replication Agent? 538
What ports does the AWS Replication Agent utilize? 538
What kind of resources does the AWS Replication Agent utilize? 538
Can Elastic Disaster Recovery migrate containers? 538
Does the AWS Replication Agent cache any data to disk? 539
How is communication between the AWS Replication Agent and the Elastic Disaster Recovery Service Manager secured? 539
Is it possible to change the port the AWS Replication Agent utilizes from TCP Port 1500 to a different port? 539
How do I manually uninstall the Elastic Disaster Recovery Agent from a server? 539
When do I need to reinstall the Agent? 539
How much bandwidth does the AWS Replication Agent consume? 540
How many disks can the AWS Replication Agent replicate? 540
Is it possible to add a disk to replication without a complete resync of any disks that have already been replicated? 540
Which Windows and Linux OSs support no-rescan upon reboot? 540
No-Rescan Upon Reboot Limitations 541
How do temporary credentials work? 541
Where can I find the AWS DRS Replication Agent logs 541
Replication related 542
Can we set specific IP addresses for the replication server or conversion server in the AWS DRS staging area? 542

What do Lag and Backlog mean during replication? 542
Is the replicated data encrypted? 542
How is the replication server provisioned and managed in the Staging Area? 542
What type of replication server is utilized in the Elastic Disaster Recovery Staging Area? .. 543
What happens if the configured replication server instance type is unavailable? 543
Does AWS Elastic Disaster Recovery compress data during replication? 543
Are events that are generated by the AWS Elastic Disaster Recovery servers logged in Cloudtrail in AWS? 543
How many snapshots does Elastic Disaster Recovery create? 543
Does Elastic Disaster Recovery delete snapshots? 544
How much capacity is allocated to the staging area? 544
Why is 0.0.0.0:1500 added to inbound rules in the Staging Area? 544
What should I know about rescans? 544
How does AWS Elastic Disaster Recovery ensure that replication servers are patched and follow security best practices? 545
Is the Elastic Disaster Recovery replication crash consistent? 545
How can I run a network bandwidth test to an Amazon EC2 endpoint? 545
AWS related 545
What does the Elastic Disaster Recovery machine conversion server do? 545
How do I change the server AMI on AWS after recovery? 545
Which AWS services are automatically installed when launching a drill or recovery instance? 546
How long does it take to copy a disk from the AWS Elastic Disaster Recovery staging area to production? 546
What are the differences between conversion servers and replication servers? 546
Can I prevent Elastic Disaster Recovery from cleaning up drill instance resources in AWS? 547
Why are my Windows Server disks read-only after launching the drill or recovery instance? 547
What impacts the conversion and boot time of drill and recovery instances? 547
How is the AWS Licensing Model Tenancy chosen for Elastic Disaster Recovery? 548
How does Elastic Disaster Recovery interact with interface VPC endpoints? 548
Will AWS Elastic Disaster Recovery reserve EC2 capacity for recovery? 549
Advanced FAQ 549
Does AWS DRS support Nutanix? 549
Does AWS DRS support VMware vSphere? 550

Does AWS DRS support Microsoft Hyper-V? ... 550
Release Notes ... 551
AWS Elastic Disaster Recovery Service Release Notes ... 551
September 2026 ... 551
August 2026 ... 551
July 2026 ... 552
June 2026 ... 552
April 2025 ... 553
February 2025 ... 553
October 2024 ... 553
September 2024 ... 553
July 2024 ... 553
May 2024 ... 553
April 2024 ... 554
January 2024 ... 554
November 2023 ... 554
October 2023 ... 555
September 2023 ... 555
August 2023 ... 555
July 2023 ... 556
June 2023 ... 556
May 2023 ... 556
April 2023 ... 556
March 2023 ... 557
February 2023 ... 557
December 2022 ... 557
November 2022 ... 557
October 2022 ... 557
September 2022 ... 557
June 2022 ... 557
May 2022 ... 558
April 2022 ... 558
March 2022 ... 558
February 2022 ... 558
January 2022 ... 558
November 2021 ... 558

AWS Elastic Disaster Recovery Client Release Notes 559
What's in a Release? 559
Agent Version History 559
Failback Client Version History 570
DRSFA Version History 572
Document history 574

What is Elastic Disaster Recovery?

AWS Elastic Disaster Recovery minimizes downtime and data loss with fast, reliable recovery of on-premises and cloud-based applications using affordable storage, minimal compute, and point-in-time recovery.

You can increase IT resilience when you use AWS Elastic Disaster Recovery to replicate on-premises or cloud-based applications running on supported operating systems. Use the AWS Management Console to configure replication and launch settings, monitor data replication, and launch instances for drills or recovery.

Set up AWS Elastic Disaster Recovery on your source servers to initiate secure data replication. Your data is replicated to a staging area subnet in your AWS account, in the AWS Region you select. The staging area design reduces costs by using affordable storage and minimal compute resources to maintain ongoing replication.

You can perform non-disruptive tests to confirm that implementation is complete. During normal operation, maintain readiness by monitoring replication and periodically performing non-disruptive recovery and failback drills. AWS Elastic Disaster Recovery automatically converts your servers to boot and run natively on AWS when you launch instances for drills or recovery. If you need to recover applications, you can launch recovery instances on AWS within minutes, using the most up-to-date server state or a previous point in time. After your applications are running on AWS, you can choose to keep them there, or you can initiate data replication back to your primary site when the issue is resolved. You can fail back to your primary site whenever you're ready.

Getting started with AWS Elastic Disaster Recovery

Topics

- Sign up for an AWS account
- Disaster recovery overview
- Elastic Disaster Recovery initialization and permissions
- Accessing the AWS Elastic Disaster Recovery Console
- AWS Elastic Disaster Recovery supported AWS Regions
- Using the AWS Elastic Disaster Recovery Console
- Best practices for Elastic Disaster Recovery
- Disaster recovery at scale
- Elastic Disaster Recovery quick start guide

Sign up for an AWS account

To get started with AWS, you need an AWS account. There are two ways to sign up for AWS. If you want complete control over account configuration and where your content is stored, or if you have regulated workloads such as HIPAA or FedRAMP, use Sign up for AWS (advanced). If you are a new user and want to use a login you already own to sign in to AWS and create an AWS environment with preconfigured defaults in select AWS Regions, use Sign up for AWS.

Disaster recovery overview

The general process is:

1. Initialize AWS Elastic Disaster Recovery in the target AWS Region. You can initialize through the console or API. See the list of supported AWS Regions.
2. Install the AWS Replication Agent on the source server.
3. Wait until initial sync is finished. After installing the agent, the initial synchronization process performs block-level replication from the source server to the replication server in the staging area.
4. Launch drill instances. Perform acceptance drills on the servers. After the drill is tested successfully, finalize the drill and delete the instance.

5. Configure post-launch actions if needed.
6. Confirm that there is no replication lag.
7. Initiate a failover by redirecting traffic.
8. Confirm that the Recovery instance was launched successfully.
9. To recover your data, initiate a failback.
10. Complete the failback.
11. Return to normal operations.

For service quotas and limits, see AWS Elastic Disaster Recovery endpoints and quotas.

Resources

The following free technical trainings are available for DRS:

- AWS Elastic Disaster Recovery - A Technical Introduction

Elastic Disaster Recovery initialization and permissions

In order to use AWS Elastic Disaster Recovery, the service must first be initialized for any AWS Region in which you plan to use Elastic Disaster Recovery.

Initializing AWS Elastic Disaster Recovery

AWS Elastic Disaster Recovery must be initialized upon first use from within the AWS Elastic Disaster Recovery Console. The initialization process occurs automatically once a user accesses the AWS Elastic Disaster Recovery Console. The user is directed to create the default replication settings, and upon saving the template, the service is initialized by creating the IAM roles which are required for the service to work. Learn more about creating the default replication settings as part of the quick start guide.

AWS Elastic Disaster Recovery can only be initialized by the Admin user of your AWS Account. During initialization, the following IAM roles are created:

- **AWSServiceRoleForElasticDisasterRecovery**
- **AWSElasticDisasterRecoveryReplicationServerRole**
- **AWSElasticDisasterRecoveryConversionServerRole**

- **AWSElasticDisasterRecoveryRecoveryInstanceRole**
- **AWSElasticDisasterRecoveryAgentRole**
- **AWSElasticDisasterRecoveryFailbackRole**
- **AWSElasticDisasterRecoveryRecoveryInstanceWithLaunchActionsRole**

Additional policies

You can create roles with granular permission for AWS Elastic Disaster Recovery. The service comes with the following predefined managed IAM policies:

- AWSElasticDisasterRecoveryConsoleFullAccess
- AWSElasticDisasterRecoveryReadOnlyAccess
- AWSElasticDisasterRecoveryAgentPolicy
- AWSElasticDisasterRecoveryAgentInstallationPolicy
- AWSElasticDisasterRecoveryFailbackPolicy
- AWSElasticDisasterRecoveryFailbackInstallationPolicy
- AWSElasticDisasterRecoveryInstancePolicy
- AWSElasticDisasterRecoveryServiceRolePolicy
- AWSElasticDisasterRecoveryLaunchActionsPolicy

Learn more about AWS Elastic Disaster Recovery roles and managed policies.

Initializing DRS through the API

You can initialize AWS Elastic Disaster Recovery through the API. This can help you automate service initialization by script when initializing multiple accounts.

> **Note**
>
> You need to create the replication settings template after initializing the service.

To initialize AWS Elastic Disaster Recovery manually, create the following IAM roles through the IAM CreateRole API. Learn more about creating IAM roles in the AWS IAM documentation.

Creation of each role must include the following parameters:

Role name	Path	Trusted Entity
AWSElasticDisasterRecoveryAgentRole	/service-role/	drs.amazonaws.com
AWSElasticDisasterRecoveryFailbackRole	/service-role/	drs.amazonaws.com
AWSElasticDisasterRecoveryConversionServerRole	/service-role/	ec2.amazonaws.com
AWSElasticDisasterRecoveryRecoveryInstanceRole	/service-role/	ec2.amazonaws.com
AWSElasticDisasterRecoveryReplicationServerRole	/service-role/	ec2.amazonaws.com
AWSElasticDisasterRecoveryRecoveryInstanceWithLaunchActionsRole	/service-role/	ec2.amazonaws.com

Example using the AWS CLI: `aws iam create-role --path "/service-role/" --role-name AWSElasticDisasterRecoveryReplicationServerRole --assume-role-policy-document '{"Version": "2012-10-17", "Statement": [{"Effect":"Allow","Principal": {"Service":"ec2.amazonaws.com"},"Action":"sts:AssumeRole"}]}'`

After the roles have been created, attach the following AWS managed policies to the roles through the IAM AttachRolePolicy API. Learn more about adding and removing IAM identity permissions in the AWS IAM documentation.

1. Attach Managed Policy **AWSElasticDisasterRecoveryAgentPolicy** to Role **AWSElasticDisasterRecoveryAgentRole**
2. Attach Managed Policy **AWSElasticDisasterRecoveryFailbackPolicy** to Role **AWSElasticDisasterRecoveryFailbackRole**

3. Attach Managed Policy **AWSElasticDisasterRecoveryConversionServerPolicy** to Role **AWSElasticDisasterRecoveryConversionServerRole**
4. Attach Managed Policy **AWSElasticDisasterRecoveryRecoveryInstancePolicy** to Role **AWSElasticDisasterRecoveryRecoveryInstanceRole**
5. Attach Managed Policy **AWSElasticDisasterRecoveryReplicationServerPolicy** to Role **AWSElasticDisasterRecoveryReplicationServerRole**
6. Attach Managed Policy **AWSElasticDisasterRecoveryRecoveryInstancePolicy** and **AmazonSSMManagedInstanceCore** to Role **AWSElasticDisasterRecoveryRecoveryInstanceWithLaunchActionsRole**

> **Note**
>
> Roles must also have a trust policy defined. The trust policy needs to define source identity and source account for security reasons, and allow the service to call SetSourceIdentity and AssumeRole. See the following policy examples.

Example 1: creating a role for the **AWSElasticDisasterRecoveryAgentRole** with trusted entity relationships via the CreateRole API:

Role: AWSElasticDisasterRecoveryAgentRole

```
$ aws iam create-role --path "/service-role/" --role-name
   AWSElasticDisasterRecoveryAgentRole --assume-role-policy-document file://agent-
source-drs-trust-policy.json
```

agent-source-drs-trust-policy.json

Example 2: creating a role for the **AWSElasticDisasterRecoveryFailbackRole** with trusted entity relationships via the CreateRole API:

Role: AWSElasticDisasterRecoveryFailbackRole

```
$ aws iam create-role --path "/service-role/" --role-name
   AWSElasticDisasterRecoveryFailbackRole --assume-role-policy-document file://
failback-source-drs-trust-policy.json
```

failback-source-drs-trust-policy.json

Example 3: creating roles for the **AWSElasticDisasterRecoveryConversionServerRole**, **AWSElasticDisasterRecoveryRecoveryInstanceRole**, and **AWSElasticDisasterRecoveryReplicationServerRole** with trusted entity relationships via the CreateRole API:

Role: AWSElasticDisasterRecoveryConversionServerRole

```
$ aws iam create-role --path "/service-role/" --role-name
   AWSElasticDisasterRecoveryConversionServerRole --assume-role-policy-document file://
source-drs-trust-policy.json
```

Role: AWSElasticDisasterRecoveryRecoveryInstanceRole

```
$ aws iam create-role --path "/service-role/" --role-name
   AWSElasticDisasterRecoveryRecoveryInstanceRole --assume-role-policy-document file://
source-drs-trust-policy.json
```

Role: AWSElasticDisasterRecoveryReplicationServerRole

```
$ aws iam create-role --path "/service-role/" --role-name
   AWSElasticDisasterRecoveryReplicationServerRole --assume-role-policy-document
 file://source-drs-trust-policy.json
```

source-drs-trust-policy.json

JSON

```
{
    "Version":"2012-10-17",
     "Statement": [
        {
             "Effect":  "Allow",
             "Principal": {
                 "Service":  "ec2.amazonaws.com"
            },
```

```
            "Action":  "sts:AssumeRole"
        }
    ]
}
```

Once the policies are attached to the roles, run the `aws drs initialize-service` command. This automatically creates the service-linked role (**AWSServiceRoleForElasticDisasterRecovery**), creates instance profiles, adds roles to instance profiles, and finishes service initialization.

Learn more about AWS Elastic Disaster Recovery roles and managed policies.

Programmatically initializing DRS

To programmatically initialize the service, create an IAM role with the following IAM policy:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:AttachRolePolicy",
            "Resource": "*",
            "Condition": {
                "ForAnyValue:ArnEquals": {
                    "iam:PolicyARN": [
                        "arn:aws:iam::aws:policy/service-role/
AWSElasticDisasterRecoveryAgentPolicy",
                        "arn:aws:iam::aws:policy/service-role/
AWSElasticDisasterRecoveryFailbackPolicy",
                        "arn:aws:iam::aws:policy/service-role/
AWSElasticDisasterRecoveryConversionServerPolicy",
                        "arn:aws:iam::aws:policy/service-role/
AWSElasticDisasterRecoveryRecoveryInstancePolicy",
                        "arn:aws:iam::aws:policy/service-role/
AWSElasticDisasterRecoveryReplicationServerPolicy"
                    ]
                }
            }
```

```
        },
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::*:role/*",
            "Condition": {
                "ForAnyValue:StringLike": {
                    "iam:PassedToService": [
                        "ec2.amazonaws.com",
                        "drs.amazonaws.com"
                    ]
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "drs:InitializeService",
                "drs:ListTagsForResource",
                "drs:GetReplicationConfiguration",
                "drs:CreateLaunchConfigurationTemplate",
                "drs:GetLaunchConfiguration",
                "drs:CreateReplicationConfigurationTemplate",
                "drs:*ReplicationConfigurationTemplate*",
                "iam:TagRole",
                "iam:CreateRole",
                "iam:GetServiceLinkedRoleDeletionStatus",
                "iam:ListAttachedRolePolicies",
                "iam:ListRolePolicies",
                "iam:GetRole",
                "iam:DeleteRole",
                "iam:DeleteServiceLinkedRole",
                "ec2:CreateSecurityGroup",
                "ec2:CreateTags",
                "sts:DecodeAuthorizationMessage",
                "ec2:DescribeSecurityGroups",
                "ec2:Get*"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
```

```
            "Resource": "arn:aws:iam::*:role/aws-service-role/drs.amazonaws.com/
AWSServiceRoleForElasticDisasterRecovery"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:CreateInstanceProfile",
                "iam:ListInstanceProfilesForRole",
                "iam:GetInstanceProfile",
                "iam:ListInstanceProfiles",
                "iam:AddRoleToInstanceProfile"
            ],
            "Resource": [
                "arn:aws:iam::*:instance-profile/*",
                "arn:aws:iam::*:role/*"
            ]
        }
    ]
}
```

Once the policies are attached to the roles, run the `aws drs initialize-service` command. This automatically creates the service-linked role (**AWSServiceRoleForElasticDisasterRecovery**), creates instance profiles, adds roles to instance profiles, and finishes service initialization.

Learn more about AWS Elastic Disaster Recovery roles and managed policies.

Accessing the AWS Elastic Disaster Recovery Console

You can access AWS Elastic Disaster Recovery directly through the AWS Console or through the following links:

- Commercial AWS Regions: https://console.aws.amazon.com/drs/home
- AWS GovCloud Regions: https://console.amazonaws-us-gov.com/drs/home

AWS Elastic Disaster Recovery supported AWS Regions

The following AWS Regions are supported by AWS Elastic Disaster Recovery:

Region name	Region identity	Support in AWS Elastic Disaster Recovery
Africa (Cape Town)	af-south-1	Yes
Asia Pacific (Hong Kong)	ap-east-1	Yes
Asia Pacific (Taipei)	ap-east-2	Yes
Asia Pacific (Tokyo)	ap-northeast-1	Yes
Asia Pacific (Seoul)	ap-northeast-2	Yes
Asia Pacific (Osaka)	ap-northeast-3	Yes
Asia Pacific (Mumbai)	ap-south-1	Yes
Asia Pacific (Hyderabad)	ap-south-2	Yes
Asia Pacific (Singapore)	ap-southeast-1	Yes
Asia Pacific (Sydney)	ap-southeast-2	Yes
Asia Pacific (Jakarta)	ap-southeast-3	Yes
Asia Pacific (Melbourne)	ap-southeast-4	Yes
Asia Pacific (Malaysia)	ap-southeast-5	Yes
Asia Pacific (New Zealand)	ap-southeast-6	Yes
Asia Pacific (Thailand)	ap-southeast-7	Yes
Canada (Central)	ca-central-1	Yes
Canada West (Calgary)	ca-west-1	Yes
Europe (Frankfurt)	eu-central-1	Yes
Europe (Zurich)	eu-central-2	Yes
Europe (Stockholm)	eu-north-1	Yes

Region name	Region identity	Support in AWS Elastic Disaster Recovery
Europe (Milan)	eu-south-1	Yes
Europe (Spain)	eu-south-2	Yes
Europe (Ireland)	eu-west-1	Yes
Europe (London)	eu-west-2	Yes
Europe (Paris)	eu-west-3	Yes
Israel (Tel Aviv)	il-central-1	Yes
Middle East (UAE)	me-central-1	Yes
Middle East (Bahrain)	me-south-1	Yes
Mexico (Central)	mx-central-1	Yes
South America (São Paulo)	sa-east-1	Yes
US East (N. Virginia)	us-east-1	Yes
US East (Ohio)	us-east-2	Yes
AWS GovCloud (US-East)	us-gov-east-1	Yes
AWS GovCloud (US-West)	us-gov-west-1	Yes
US West (N. California)	us-west-1	Yes
US West (Oregon)	us-west-2	Yes

Learn more about AWS Services by Region.

AWS Elastic Disaster Recovery regional support includes AWS Local Zones associated with the above supported regions.

Using the AWS Elastic Disaster Recovery Console

AWS Elastic Disaster Recovery is AWS Region-specific. Make sure that you select the correct Region from the **Select a Region** menu when using AWS Elastic Disaster Recovery, just like you would with other AWS Region-specific services such as Amazon EC2.

AWS Elastic Disaster Recovery is divided into several primary pages. Each page contains additional tabs and actions. The default view for the AWS Elastic Disaster Recovery Console is the **Source servers** page. This page automatically opens every time you open AWS Elastic Disaster Recovery. You can navigate to other AWS Elastic Disaster Recovery pages through the left pane **AWS Elastic Disaster Recovery** navigation menu.

Each Elastic Disaster Recovery page opens in the right pane.

Source servers page

The Source Servers page lists all of the source servers you added to AWS Elastic Disaster Recovery and allows you to interact with your servers and perform actions. Learn more about the Source servers page.

Control your source servers in the AWS Elastic Disaster Recovery console through the **Actions**, **Replication**, and **Initiate recovery job** menus.

Review the progress of commands through the **Recovery job history** tab. Learn more about recovery job history.

The commands in the **Actions** and **Initiate recovery job** menus influence the specific source servers you selected. You can select a single source server or multiple source servers for any command.

Use the **Filter source servers by property or value** field to filter servers.

AWS Elastic Disaster Recovery color codes the state of each source server. Use the **Alerts** column to easily determine the state of your server.

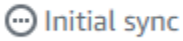
- A server that is ready to launch Drill or Recovery instances displays the green checkmark and states **Ready**.



A server that is ready to launch Drill or Recovery instances, but is experiencing a non-critical issue such as lag displays the blue info sign and states **Ready** and displays the lag duration to the right. You may need to take action to fix the lag.



A server that is still undergoing initial sync displays a gray circle with three dots and states **Initial sync**.



A server that is disconnected displays the gray warning sign and states **Disconnected**.



A server that is not ready due to a significant error, such as a stall, displays a red **X** and states **Not ready**. The Not Ready state is only shown for servers that are not replicating and do not have any previously created Points in Time. Action must be taken in order to fix the issue.

When some commands are initiated AWS Elastic Disaster Recovery displays information messages at the top of the **Source servers** page. AWS Elastic Disaster Recovery color codes these messages for clarity. A green message means that a command was completed successfully. A red message means that a command was not completed successfully. Each message provides details and links to supplemental information.

AWS Elastic Disaster Recovery allows you to interact with and manage each server. Choose the server hostname to be redirected to the server details view.

The **Server details** view tab shows specific details for an individual server. From here, you can see an overview of the server's recovery state, as well as various technical details, manage tags, manage disks, edit the server's replication settings, and edit the server's launch settings through the various tabs. Learn more about the Server Details view.

Certain Elastic Disaster Recovery commands, such as **Edit replication settings**, allow you to interact with multiple source servers at once. When multiple source servers are selected and the **Replication > Edit replication settings** option is chosen, AWS Elastic Disaster Recovery indicates which servers are being edited.

In order for setting changes you have made in the AWS Elastic Disaster Recovery Console to take effect, be sure to choose **Save** at the bottom of each Settings page.

Best practices for Elastic Disaster Recovery

For a more complete discussion of best practices for planning, implementing, and maintaining disaster recovery for on-premises applications using AWS, see this white paper.

Planning

1. Being ready for a real recovery event requires pre-planning. Simply having your servers replicating to AWS, and even having launched them once is not enough. You should have a written recovery plan of what to do in the event of a real recovery event. To learn more, read this Checklist for your IT disaster recovery plan.
2. Once your source servers have reached the Healthy state (after initial sync has completed), you should launch Drill instances for each of your applications and ensure that each application as a whole is working as expected when running in your recovery AWS Region. As you go through this process, you will likely create the necessary network resources required (together with security groups and other related resources). While you can keep these recovery networks (and related resources) up and running even when not in use, it is recommended that once you have them set up properly, create a CloudFormation template that can create them on demand, should the need arise. You should discover and record the order in which servers, and applications need to be launched, and record this in the recovery plan.

Drilling

Regular drills are an integral part of any Disaster Recovery solution. With DRS, drilling is simple and nondisruptive (both to the servers at the source, and to the replication process itself). We recommend drilling as often as is practical, and at least several times a year, and updating the recovery plan with any findings and required changes. Testing and understanding failback is also important. Be sure to include it in your initial drill, and in at least some of your regular drills.

Regular testing can help ensure that your resources are properly prepared for both disasters and scheduled drills. Before conducting large-scale scheduled drills, make sure you meet all the prerequisites and run the required tests. To allow our support team to assist you in case of misconfiguration or other issues, conduct the preliminary testing a week or 2 before the scheduled drill.

> **Note**
>
> While your drill instances are up and running, you are paying for them as per your standard Amazon EC2 rates. Make sure to terminate the drill instances when the drill is done, and include this as a step in your recovery plan.

Monitoring

You can monitor the health of the ongoing replication using the DRS console or programmatically. In the AWS DRS console, go to the **Servers list** page, and look at the **Ready for recovery** column. Any server that is not showing as **Ready** with a green checkmark, may require attention. Servers that show **stalled** in the **Data replication status** column require your intervention to resolve. Servers that are showing **Lag**, may resolve themselves (unless they are also stalled). You should monitor and explore to see if the Lag is a persistent problem (for example, due to insufficient network bandwidth). You can use a scripted solution and the DRS API to respond to servers becoming stalled, or going into lag, or you can use Amazon EventBridge and the EventBridge events generated by AWS DRS.

Limits

Due to Amazon EBS limits on the rate at which EBS snapshots can be taken, the maximum number of servers that can be replicated using DRS in a single AWS account is limited to 300. To replicate more than the maximum number of servers, use multiple AWS accounts, or multiple target AWS Regions (you need to set up DRS separately for each account/ Region).

You can also use multiple staging or target accounts, as described in Using multiple staging accounts with AWS DRS.

Protecting Point-In-Time snapshots

DRS uses EBS snapshots to maintain recovery Points-In-Time. If these are deleted, then you can only recover from the latest state, as maintained on the replication server (and if it is terminated, then you can no longer recover at all). In the event of a breach, which includes not just corruption of your data at source, but also access to your AWS account, then the malicious actor could delete your Point-In-Time snapshots, unless you take extra measures to protect them.

Controlling agent installation permissions

You should control who can install the AWS Replication Agent in your account. Once an agent is installed you immediately begin accruing charges for DRS, and for replication resources (such as EBS, etc.) The agent installation permissions should be as limited as is practical. The recommended way for controlling who can install agents is to create an IAM role, and to allow users to assume the role.

1. Create an IAM role (IAM docs link | IAM console link), based on the DRS managed permission for agent installation. If this role is to be used by someone outside of your AWS account make sure to use the external ID functionality. Send the role ARN to the users who need to install agents (ARN is not secret and can be sent via email). Use permission boundaries to further limit what can be done using that role. For example, you can control which AWS Region it can be used for, how long the temporary credentials created with the role are good for, specify tags that must be provided (or may not be provided) during agent installation, and more.
2. Users who install the agents assumes that role (must be a user of an AWS account, either yours, or another; you configure who the role is for in step 1). This creates temporary IAM credentials for that user which are used for agent installation. These credentials are limited to only the permissions required for agent installation (and further limited by the permission boundaries you defined), yet are associated with the user (for example, so their usage can be tracked using CloudTrail).

Recovery best practices

1. **Overview:** DRS makes successful recovery possible, by handling ongoing replication, and the on-demand launching of actual Recovery instances. The re-routing of traffic (failover) is not done via DRS, and should be done using your preferred DNS routing service, such as Amazon Route 53. Your recovery plan should include details of which service to use, who in your organization owns this service, and what conditions must be met to perform the re-routing (for example: launch Recovery instances using DRS, perform successful launch-validation test, wait for system X, Y, and Z to also launch and pass test, then re-route).
2. **Termination protection for recovery instances:** When you launch recovery instances in case of a real event, you should prevent them from being inadvertently terminated. This should be done after you have performed launch-validation test, and before data re-routing. You can turn on termination protection directly from the Amazon EC2 console, by selecting the instances, and from the **Actions** menu choosing **Instance settings, change termination protection**, and

choosing **Yes, Enable.** You should document this step in your recovery plan. Learn more about termination protection.

3. **Understanding failover costs:** Your EC2 recovery instances are created according to the launch settings you have configured for each source server. Recovery instances accrue EC2 and EBS charges as per AWS rates for your account in the target AWS Region. While you use the Recovery instances, you also continue paying for DRS, and the replication resources it created.
4. **Recovery dos and don'ts:** Do not use the **Disconnect from AWS** action in the DRS console for servers for which you launched Recovery instances, even in the case of a real recovery event. Performing a disconnect terminates all replication resources related to these source servers, including your Point-In-Time (PIT) recovery points. You may need these PITs while you are in failover state, for regulatory reasons, or to re-launch Recovery instances for any reason (for instance if you discover that the PIT from which you launched includes corrupt or malicious data, and you want to relaunch from an earlier PIT). While you use your Recovery instances as your primary, and new data is presumably written to them, these recovery instances are not themselves being replicated, and you are not creating any new PITs for these changes. It is possible to configure the recovery instances as new source servers and replicate them cross-Region, to have disaster recovery for your recovery site. This carries with it additional costs, as noted in Performing a cross-Region failback
5. **Using recovery for migration:** Once you launch and use recovery instances on AWS for a real event, you may wish to go on using them permanently, instead of your original servers. The primary additional steps you need to do are:
   a. Set up cross region replication, so that these recovery instances become new source servers;
   b. Wait for these new source servers to have to full number of daily PITs that you need to maintain;
   c. Perform the **Disconnect from AWS** action on the original source servers, so as to avoid confusion, and to stop paying for DRS and related replication resources for these original source servers. You can also then choose **Delete** from the **Actions** menu, and this causes DRS to forget everything it knows about these source servers, and for them to no longer appear in the Elastic Disaster Recovery console.
6. **Recover into existing instance:** Use if you want to recover into an instance that already exists instead of launching a new one for recovery, drill or failback. The instance to recover into must be of the same operating system platform (Linux or Windows) as the source instance, it must be stopped and it must have the tag key *AWSDRS* and tag value *AllowLaunchingIntoThisInstance*. Learn more about recover into existing instance.

Failback best practices

1. **Mass failback:** If you are failing back more than several servers, and your source environment is VMware vCenter, then consider using DRS Mass Failback Automation client.
2. **Return to normal operation:** make sure that the failed-back servers at the source are replicating back to AWS, and appear as source servers in the DRS console. If they do appear in the DRS console and are not replicating, explore the reason (such as firewall settings, etc.) If they do not appear in the DRS console you may need to install / re-install the AWS Replication Agent on them. Make sure that you do not end up with two source server entities in the DRS console, one representing the original server, and one the failed-back server.
3. **Cleanup after return to normal operation:** Once you have completed failback, there may be multiple AWS resources left behind that you no longer need and that are costly to maintain:

   After performing a failback to on-premises environment, perform the following steps:

   - Clean Recovery instances: Terminate these instances from the **Recovery instances** page of the DRS Console.
   - Source servers: These appear in the Source Servers page of the DRS console. Make sure that you only have one source server in the DRS console for each actual server at the source. Source servers are billed by DRS and consume replication resources (billed by other AWS services) until you perform the **Disconnect from AWS** action. If you do have duplicate source servers, do not disconnect/delete the original ones until the new ones have accumulated all the Point-In-Time recovery points (PITs) you need. Performing the **disconnect from AWS** action causes the PITs from the original sources servers to be discarded. If your source is also in AWS, then you have more resources that need to be cleaned up. Learn more about cleaning up these resources.

   > **Note**
   >
   > The cleanup process following a cross-region failback is different. Learn how to perform a cleanup following a cross-region failback.

Security best practices

You can review security best practices in the Security chapter.

Disaster recovery at scale

When protecting a large number of servers (100+) with AWS Elastic Disaster Recovery, additional planning is required to ensure reliable replication, successful recovery, and manageable operations. This section provides guidance for operating Elastic Disaster Recovery at scale.

Topics

- Account and Region planning
- Network planning and benchmarking
- Storage benchmarking
- Agent deployment at scale
- DR readiness and compliance monitoring
- Service quotas and API limits
- Recovery planning at scale

Account and Region planning

A single AWS account supports up to 300 concurrently replicating source servers. For larger environments, distribute source servers across multiple staging accounts or target AWS Regions.

- Use multiple staging accounts to scale beyond the 300-server limit per account.
- Plan your account structure early — moving source servers between accounts requires reinstalling the agent.
- When using multiple accounts, ensure that EBS encryption keys (KMS) are shared across accounts if you use custom encryption.
- Establish a consistent IAM policy management strategy across all accounts. Use AWS Organizations and Service Control Policies (SCPs) to enforce guardrails.

Network planning and benchmarking

Network bandwidth is a critical factor for replication performance at scale. Before deploying agents, benchmark your network to ensure it can sustain the required throughput.

1. **Benchmark network bandwidth:** Test the bandwidth between your source environment and the staging area subnet over TLS on TCP port 1500. For instructions on running this test, see Running a network bandwidth test to an EC2 endpoint.
2. **Plan for aggregate bandwidth:** Calculate the total write throughput across all source servers (see calculating required bandwidth). Ensure your network connection (Direct Connect, VPN, or internet) can sustain this aggregate throughput with headroom for spikes.
3. **IP planning:** Plan your staging area and recovery VPC CIDR ranges to accommodate the number of replication servers, recovery instances, and any network infrastructure (NAT gateways, transit gateways, load balancers). Ensure there is no IP overlap between source and recovery environments if using VPN or Direct Connect.

Storage benchmarking

Understanding the storage write patterns of your source servers helps you provision appropriate replication resources and avoid replication lag.

1. Capture storage performance metrics on your source servers using `iostat` on Linux or Performance Monitor on Windows. Focus on write IOPS and write throughput (MB/s) per disk.
2. Servers with high write rates (such as database servers) may require dedicated replication servers with an instance type that can handle the required EBS IOPS and throughput.
3. Consider excluding high-churn volumes that are not needed for disaster recovery (such as database tempdb or backup disks) using the `--devices` installer parameter to reduce replication load.

Agent deployment at scale

Deploying the AWS Replication Agent across hundreds of servers requires automation. Consider the following approaches:

- Use configuration management tools (such as AWS Systems Manager Run Command, Ansible, or Chef) to deploy the agent across multiple servers simultaneously.
- Use the `--no-prompt` installer parameter for unattended installation. Combine with `--devices` to specify disks explicitly when automatic detection is not suitable.
- Deploy agents in batches rather than all at once to avoid overwhelming the staging area network and to stay within API limits.

- Verify that all source servers meet the installation prerequisites before beginning deployment.

DR readiness and compliance monitoring

At scale, manually monitoring replication health and drill compliance is impractical. Implement automated monitoring to maintain DR readiness.

1. **Replication health:** Use the Elastic Disaster Recovery API (`describe-source-servers`) or Amazon EventBridge to monitor replication state across all source servers. Alert on servers in **Stalled**, **Disconnected**, or **Lag** states.
2. **Drill compliance:** Track when each source server was last tested. Use the `describe-source-servers` API to retrieve the last launch date and type (drill or recovery) from the `lifeCycle.lastLaunch` field. Flag servers that have not been drilled within your organization's required interval.
3. **Dashboard:** For multi-account environments, consider building a centralized dashboard using AWS Organizations and cross-account IAM roles to aggregate replication status across all accounts.

Service quotas and API limits

Large-scale deployments can encounter service quota limits. Review and plan for the following:

- **Elastic Disaster Recovery quotas:** Maximum 300 concurrently replicating source servers per account, 100 source servers per recovery job, 500 source servers across all active jobs, and 20 concurrent jobs. See Elastic Disaster Recovery service quotas.
- **Amazon EC2 quotas:** Plan for the number of replication server instances, recovery instances, and associated EBS volumes that will run concurrently. Request quota increases in advance.
- **EBS snapshot limits:** Elastic Disaster Recovery creates EBS snapshots for point-in-time recovery. At scale, the number of snapshots can grow significantly based on your retention policy.
- **API throttling:** When using automation to manage large numbers of servers, implement exponential backoff and retry logic to handle API throttling gracefully.

Recovery planning at scale

Recovering hundreds of servers simultaneously requires careful orchestration:

- **Group servers by application:** Identify dependencies between servers and group them so that dependent servers are recovered together in the correct order.
- **Stagger recovery jobs:** Launch recovery instances in batches to stay within the concurrent job limits (20 concurrent jobs, 100 servers per job, 500 servers across all active jobs) and to avoid overwhelming the target environment.
- **Automate recovery orchestration:** Use the Elastic Disaster Recovery API and AWS Step Functions or similar orchestration tools to automate the recovery sequence, including post-launch validation.
- **Plan VPC capacity:** Ensure your recovery VPCs have sufficient IP addresses, subnets, and network resources for all recovery instances.

Elastic Disaster Recovery quick start guide

This section guides you through your initial Elastic Disaster Recovery setup, including:

Topics

- First time setup
- Adding source servers
- Configuring launch settings
- Launching a drill instance
- Launching a recovery instance
- Performing a failback

First time setup

In order to use AWS Elastic Disaster Recovery (AWS DRS), you first need to set it up in each AWS Region in which you want to use it (the Region into which you will be replicating, and where you will launch Recovery instances). Setting up the service consists of defining default replication settings and creating the roles and permissions required for the service to operate.

> **Note**
>
> You need to be the admin user of the AWS account, or have a role with the AWSElasticDisasterRecoveryConsoleFullAccess permission in order to set up the service

The first setup step for AWS DRS is setting the default replication settings. Choose **Set default replication settings** on the AWS Elastic Disaster Recovery landing page. You are guided through the steps of setting up your default replication settings, default launch settings, and EC2 template. These default settings are applied to every source server that is added to AWS Elastic Disaster Recovery. You can change both the default settings and individual source server settings for one or more source servers at any time. Learn more about editing your replication settings and launch settings. To learn more about each setting, select the **Info** links next to each section.

> **Important**
>
> Before configuring your default settings, ensure that you meet the Network requirements for running AWS Elastic Disaster Recovery

On the first page of the wizard, you are asked to **Set up replication servers**. Replication servers are lightweight Amazon EC2 instances that are used to replicate data between your source servers and AWS. Replication servers are automatically launched and terminated as needed. You can start using AWS Elastic Disaster Recovery with the default replication server settings or you can configure your own settings. Learn more about replication server settings.

- Configurable replication server settings include:
  - The subnet within which the replication server will be launched
  - Replication server instance type

During this step you can review the service linked role and additional policies created during Elastic Disaster Recovery initialization. Choose **View details** to learn more.

On the second page of the wizard you are asked to **Specify volumes and security groups**. For each disk on an added source server there is an identically-sized EBS volume attached to a replication server, and each replication server can handle replication of disks from multiple source servers. Learn more about volumes.

A security group acts as a virtual firewall, which controls the inbound and outbound traffic of the staging area. The best practice is to have AWS Elastic Disaster Recovery automatically attach to and monitor the default AWS Elastic Disaster Recovery security group. This group opens inbound TCP Port 1500 for receiving the transferred replicated data. Learn more about security groups.

Configurable volumes and security groups settings include:

- EBS volume type
- EBS encryption
- Always use AWS Elastic Disaster Recovery security group

On the third page of the wizard you can **Configure additional replication settings**. These include **Data routing and throttling**, **Point in time (PIT) policy**, and **Tags**.

- **Data routing and throttling** controls how data flows from the external server to the replication servers. If you choose not to use a private IP, your replication servers are automatically assigned a public IP and data flows over the public internet. Learn more about data routing and throttling.
- Point in Time (PIT) is a disaster recovery feature which allows launching an instance from a snapshot captured at a specific point in time. As source servers are replicated, snapshots are taken over time. The **Point in time (PIT) policy** section allows to configure a retention policy that determines which snapshots are not required after a defined duration.
- The **Tags** section allows you to add custom tags to resources created by AWS Elastic Disaster Recovery in your AWS account.

Additional configurable settings include:

- Use private IP for data replication
- Create public IP
- Throttle network bandwidth
- Snapshot retention
- Tags

On the fourth page of the wizard you can **Set default DRS launch settings**.

Default launch settings define how drill or recovery instances are launched in AWS. You can start using AWS Elastic Disaster Recovery with the default launch settings or configure your own. Learn more about default DRS launch settings.

Configurable options include:

- Instance type right sizing
- Start instance upon launch

- Copy private IP
- Transfer server tags
- OS licensing

The fifth page of the wizard: **Set default EC2 launch settings** is where you configure the default Amazon EC2 launch template which defines how instances are launched in AWS. Changes you make to the template only affect new servers, but you can edit the template for multiple servers according to your preferences. Learn more about default EC2 launch template. The EC2 launch template includes basic and advanced settings.

Basic configurable options include:

- Subnet
- Security groups
- Instance type
- EBS volume type

You only need to change advanced configurable options in specific operational scenarios. They include:

- IAM instance profile
- Tenancy

The sixth page is where you **Review and initialize**.

Review the settings you configured. To change a specific setting select **Edit**, which redirects you to the page in the wizard on which the setting appears. Go through the remaining pages to return to the **Review and create** page.

Once you have reviewed all of the settings you chose, select **Configure and initialize**. The default template is created and you return to the AWS Elastic Disaster Recovery console.

> **Note**
>
> You can always edit the default replication or launch settings by choosing the appropriate item from the **Settings** page, which you can open from the left-hand navigation menu.

Remember that changes you make are only applied to newly added servers and not to existing servers.

Adding source servers

Add source servers to AWS Elastic Disaster Recovery by installing the AWS Replication Agent (also referred to as "the Agent") on them. The Agent can be installed on both Linux and Windows servers. Learn more about adding source servers.

Prior to adding your source servers, ensure that you meet all of the Network requirements.

Note

DRS agents can only be installed on instances that are in AWS Regions that are supported by Elastic Disaster Recovery.

Configuring launch settings

After you have added your source servers to the AWS Elastic Disaster Recovery console, you need to configure the launch settings for each server. The launch settings are a set of instructions that determine how a recovery instance is launched for each source server on AWS. You must configure the launch settings prior to launching test or recovery instances. You can use the default settings or configure the settings to fit your requirements.

Note

You can change the launch settings after a drill or recovery instance has been launched. You need to launch a new Drill or Recovery instance for the new settings to take effect.

You can access the launch settings by selecting the hostname of a source server on the **Source servers** page.

Within the individual server view, navigate to the **Launch settings** tab.

Here you can see your **General launch settings** and your **EC2 launch template**. Select **Edit** to edit your launch settings or your EC2 launch template.

Launch settings include:

- **Instance type right-sizing** – The Instance type right-sizing feature allows AWS Elastic Disaster Recovery to launch a drill or recovery instance type that best matches the hardware configuration of the source server. When activated, this feature overrides the instance type selected in the EC2 launch template.

  > **Note**
  >
  > You cannot select **Active (basic)** right-sizing for `arm64` source servers. Select **Active (in-aws)** or **Inactive** instead. See DRS launch settings parameters.

- **Start instance upon launch** – Choose whether you want to start your Initiate recovery job instances automatically upon launch or whether you want to start them manually through the Amazon EC2 Console.
- **Copy private IP** – Choose whether you want AWS Elastic Disaster Recovery to verify that the private IP used by the drill or recovery instance matches the private IP used by the source server.
- **Transfer server tags** – Choose whether you want AWS Elastic Disaster Recovery to transfer any user-configured custom tags from your source servers to your drill or recovery instance.

AWS Elastic Disaster Recovery automatically creates an **EC2 launch template** for each new source server. AWS Elastic Disaster Recovery bases the majority of the instance launch settings on this template. You can edit this template to fit your needs.

Learn more about Launch settings.

Launching a drill instance

After you have added all of your source servers and configured their launch settings, you are ready to launch a drill instance. It is crucial to drill the recovery of your source servers to AWS prior to initiating a recovery in order to verify that your source servers function properly within the AWS environment.

> **Important**
>
> - When launching a drill, recovery, or an in-AWS failback, you can launch up to 100 source servers in a single operation. Additional source servers can be launched in subsequent operations.

- It is a best practice to perform drills regularly. After launching drill instances, use either SSH (Linux) or RDP (Windows) to connect to your instance and ensure that everything is working correctly.

You can drill one source server at a time, or simultaneously drill multiple source servers. For each source server, you are informed of the success or failure of the drill. You can drill your source server as many times as you want. Each new drill first deletes any previously launched drill or recovery instance and dependent resources. Then, a new Drill instance is launched, which reflects the chosen Point-in-time state of the source server. After the drill, data replication continues as before. The new and modified data on the source server is transferred to the Staging Area Subnet and not to the Recovery instances that were launched during the test.

Note

- Windows source servers need to have at least 2 GB of free space to successfully launch a recovery instance.
- Take into consideration that once a drill instance is launched, actual resources are used in your AWS account and you will be billed for these resources. You can terminate the operation of launched Recovery instances once you verify that they are working properly without impact to data replication.

Learn more about launching drill instances as part of the overall recovery and failback framework.

Launching a recovery instance

Once you have finalized the testing of all of your source servers, you are ready for recovery. You should perform the recovery at a set date and time. The recovery migrates your source servers to the recovery instances on AWS.

You can recover one source server at a time, or simultaneously recover multiple source servers. For each source server you are informed of the success or failure of the Recovery. For each new recovery, AWS Elastic Disaster Recovery first deletes any previously launched recovery instance and dependent resources. Then, it launches a new Recovery instance which reflects the most up-to-date state of the source server. After the Recovery, data replication continues as before. The new

and modified data on the source server is transferred to the Staging Area Subnet, and not to the recovery instances that were launched during the recovery.

Learn more about launching Recovery instances as part of the overall recovery and failback framework.

Performing a failback

Once the disaster is over, you can perform a failback to your original source server or to any other AWS Elastic Disaster Recovery Failback Client on the server. In order to use the Failback Client, you need to generate Elastic Disaster Recovery-specific credentials. Once the failback is complete, you can opt to either terminate, delete, or disconnect the Recovery instance.

Learn more about performing a failback.

Elastic Disaster Recovery replication network requirements

Before activating AWS Elastic Disaster Recovery, ensure that your environments are prepared and set accordingly. These preparations include setting the correct network settings, defining network requirements, and opening the correct ports. This documentation guides you through preparing network settings in the AWS Elastic Disaster Recovery Service Manager and user console, opening TCP Port 443 and TCP Port 1500 connections, setting firewall rules, using a proxy, and solving common networking issues.

Topics

- Elastic Disaster Recovery network diagrams
- Elastic Disaster Recovery network setting preparations
- Elastic Disaster Recovery network requirements

Elastic Disaster Recovery network diagrams

AWS Elastic Disaster Recovery supports the following source infrastructure types:

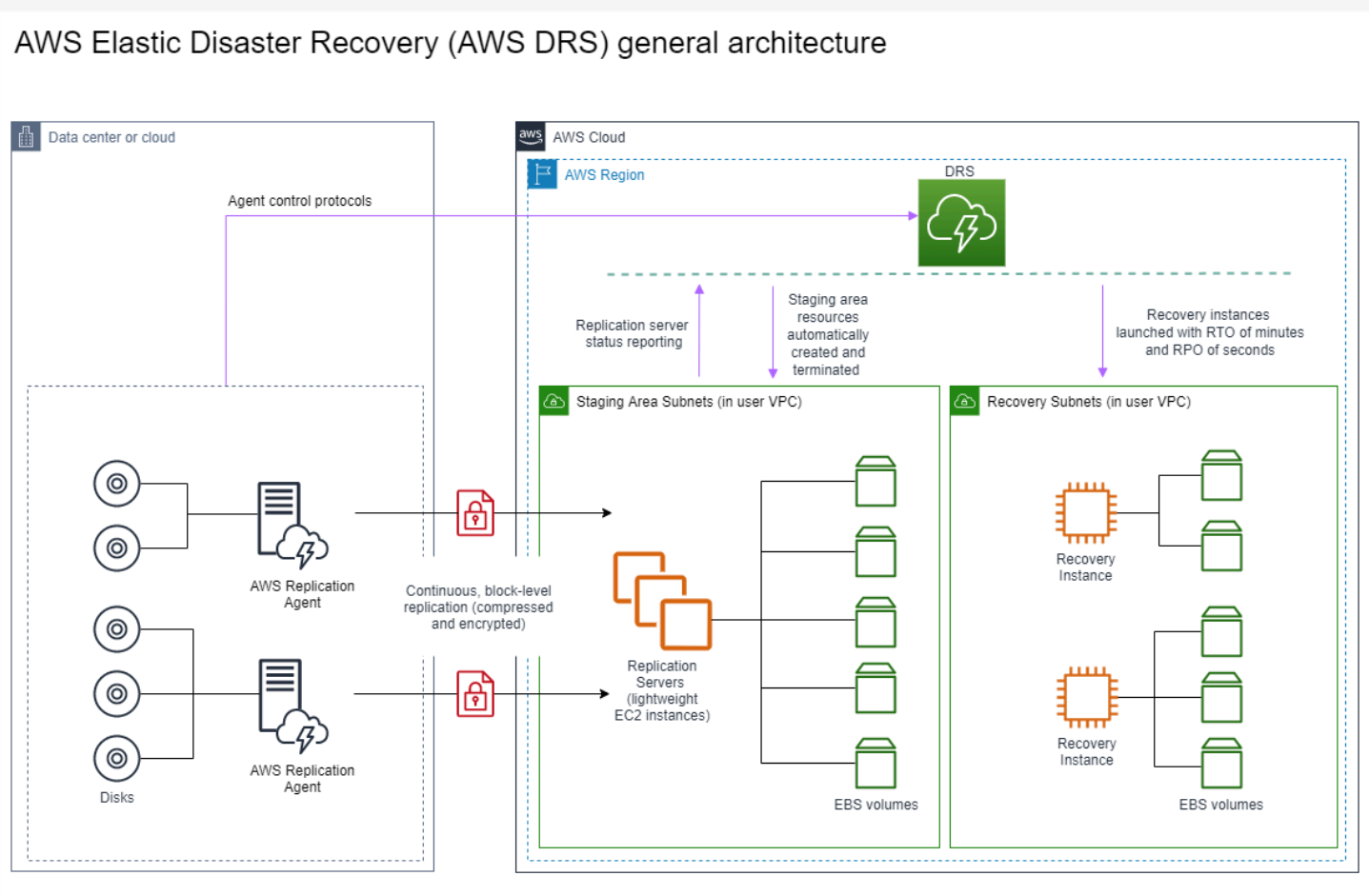
- **On-premises to AWS** – Protect physical or virtual servers in your data center by replicating to an AWS Region.
- **AWS to AWS (cross-Region)** – Protect Amazon EC2 instances by replicating from one AWS Region to another. Cross-Region replication is recommended for disaster recovery to help protect against Region-level events.
- **AWS to AWS (cross-Availability Zone)** – Replicate Amazon EC2 instances to a different Availability Zone within the same AWS Region. For comprehensive disaster recovery protection, we recommend cross-Region replication.
- **VMware to AWS** – Protect VMware vSphere environments, including both on-premises vSphere and VMware Cloud on AWS. See Disaster recovery for VMware Cloud on AWS using AWS Elastic Disaster Recovery.
- **Other clouds to AWS** – Protect workloads running on other cloud providers such as Microsoft Azure or Google Cloud. See Building a disaster recovery site on AWS for workloads on Microsoft Azure.

- **AWS to on-premises (failback)** – After a disaster recovery event, fail back from AWS to your original source environment.

The following are the network diagrams for AWS Elastic Disaster Recovery :

General Architecture - On-Premises to AWS

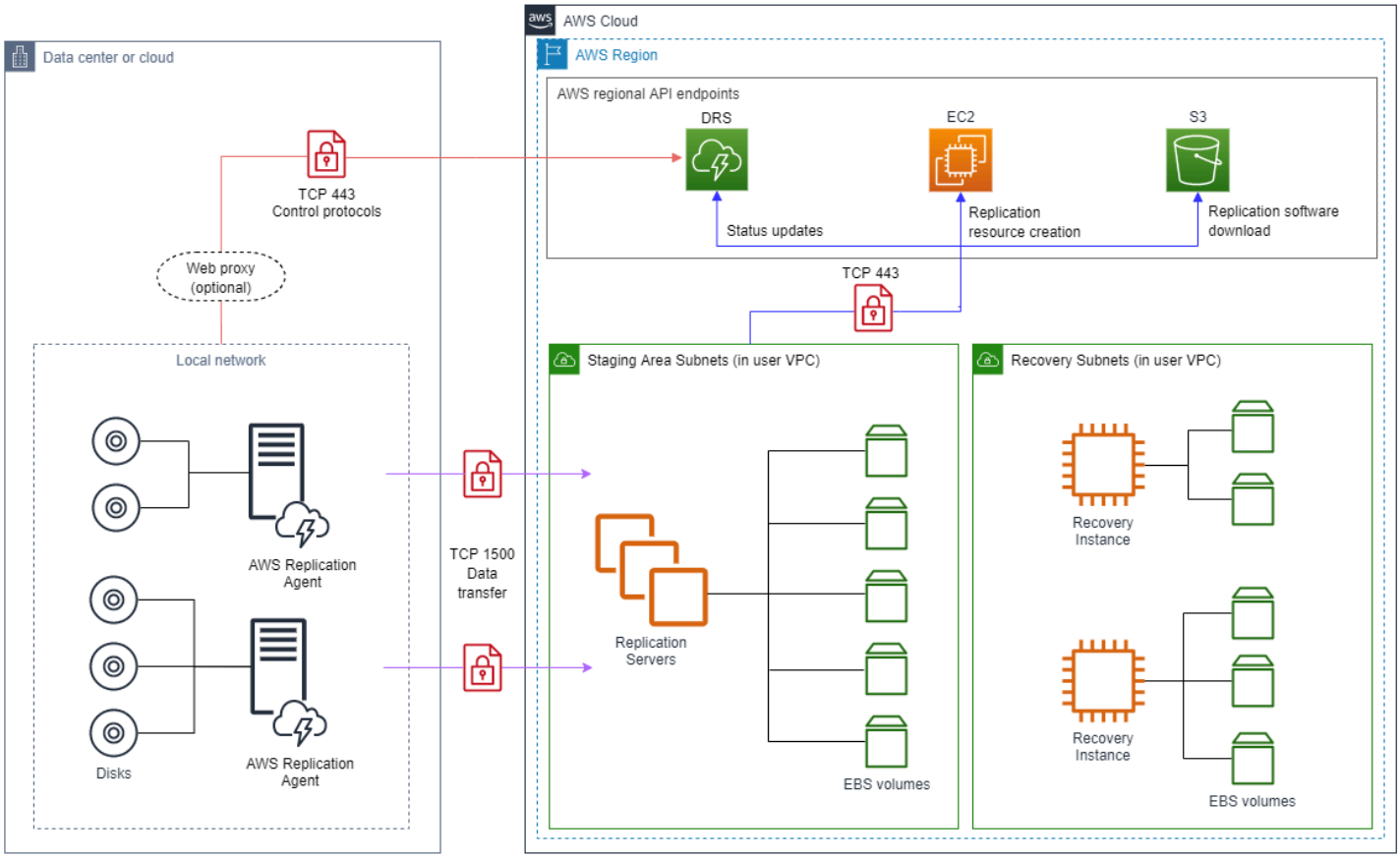
This diagram shows the general architecture of DRS protecting source servers located in an on-premises environment.



On-Prem to AWS

This diagram shows the network architecture of DRS protecting source servers located in an on-premises environment.

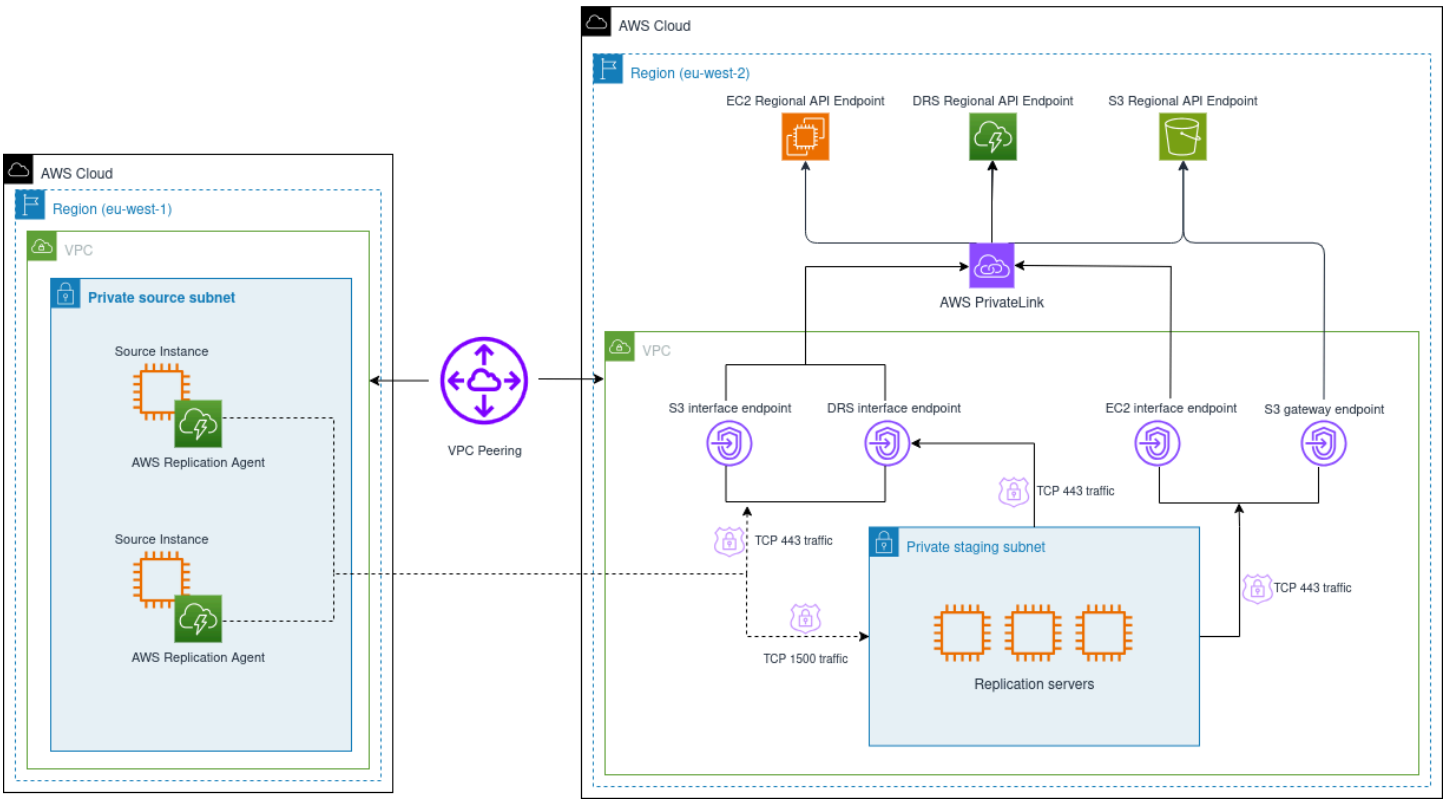
AWS Elastic Disaster Recovery (AWS DRS) network architecture



AWS Cloud to AWS Cloud via VPC Peering

This diagram shows the network architecture of AWS DRS protecting source servers located in an AWS VPC. Data replication between the source VPC and the target staging area, along with communication with the AWS DRS service, flows through a VPC peering connection.

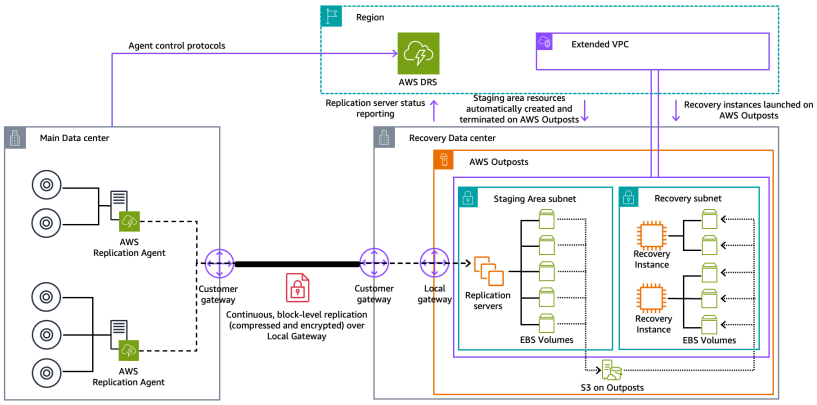
AWS Elastic Disaster Recovery (AWS DRS) communication using VPC Peering



On-Prem to Outposts

This diagram shows the network architecture of DRS protecting source servers located in an on-premises environment. The staging and recovery are both located on AWS Outposts. [Find out more about protecting source servers using Outposts.]

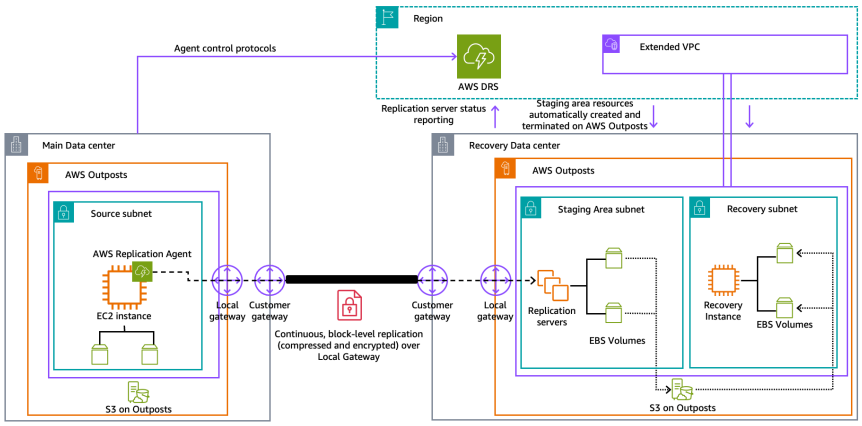
AWS Elastic Disaster Recovery (AWS DRS) with AWS Outposts



AWS to Outposts

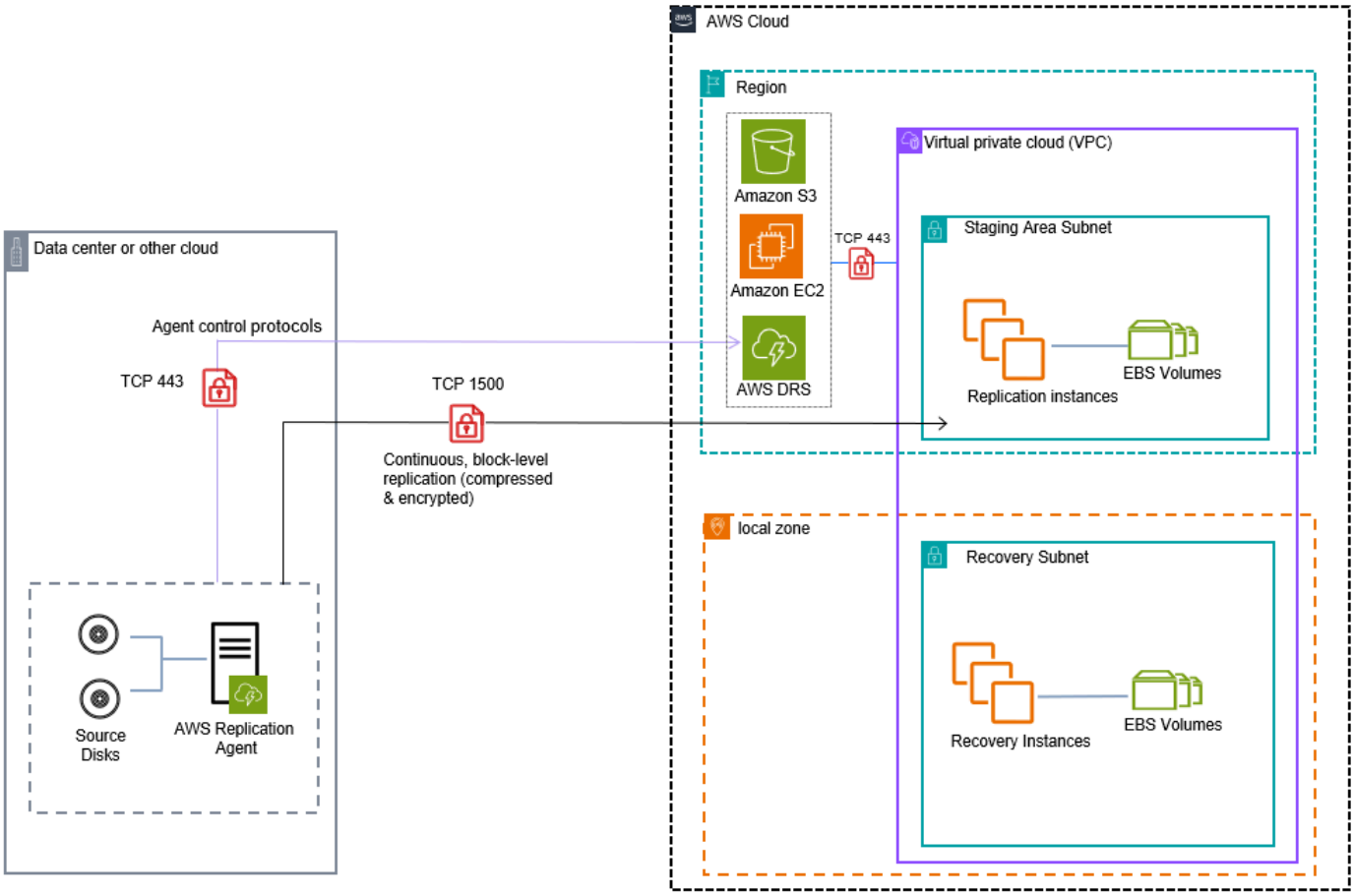
This diagram shows the network architecture of DRS protecting source servers located in AWS. The staging and recovery are both located on AWS Outposts. Find out more about protecting source servers using Outposts.

AWS Elastic Disaster Recovery (AWS DRS) AWS Outposts to AWS Outposts



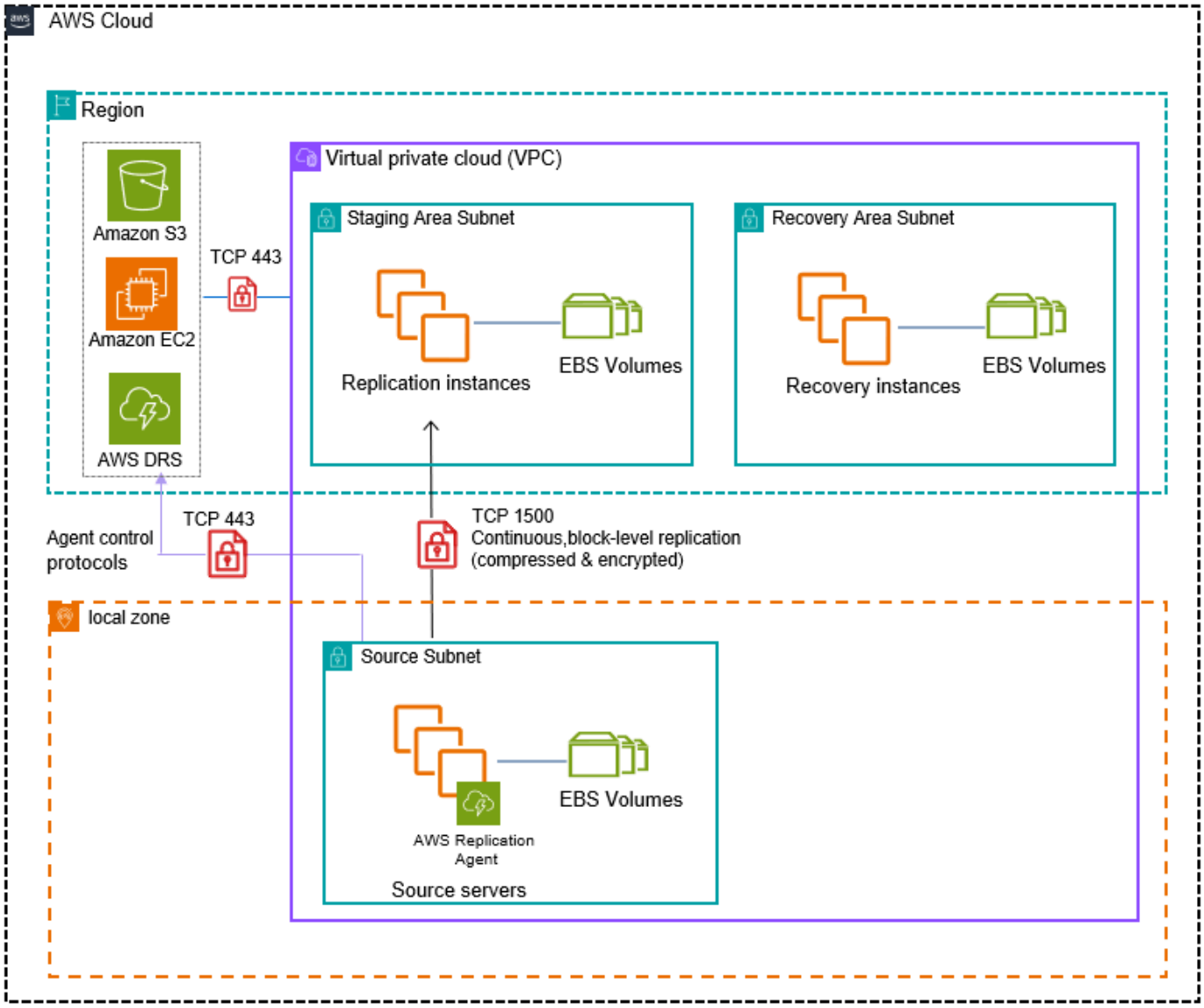
On-Premises to AWS Local Zone

This diagram shows the network architecture of DRS protecting source servers located in an on-premises environment. The staging area is located in an AWS Region and the recovery is in an AWS Local Zone.



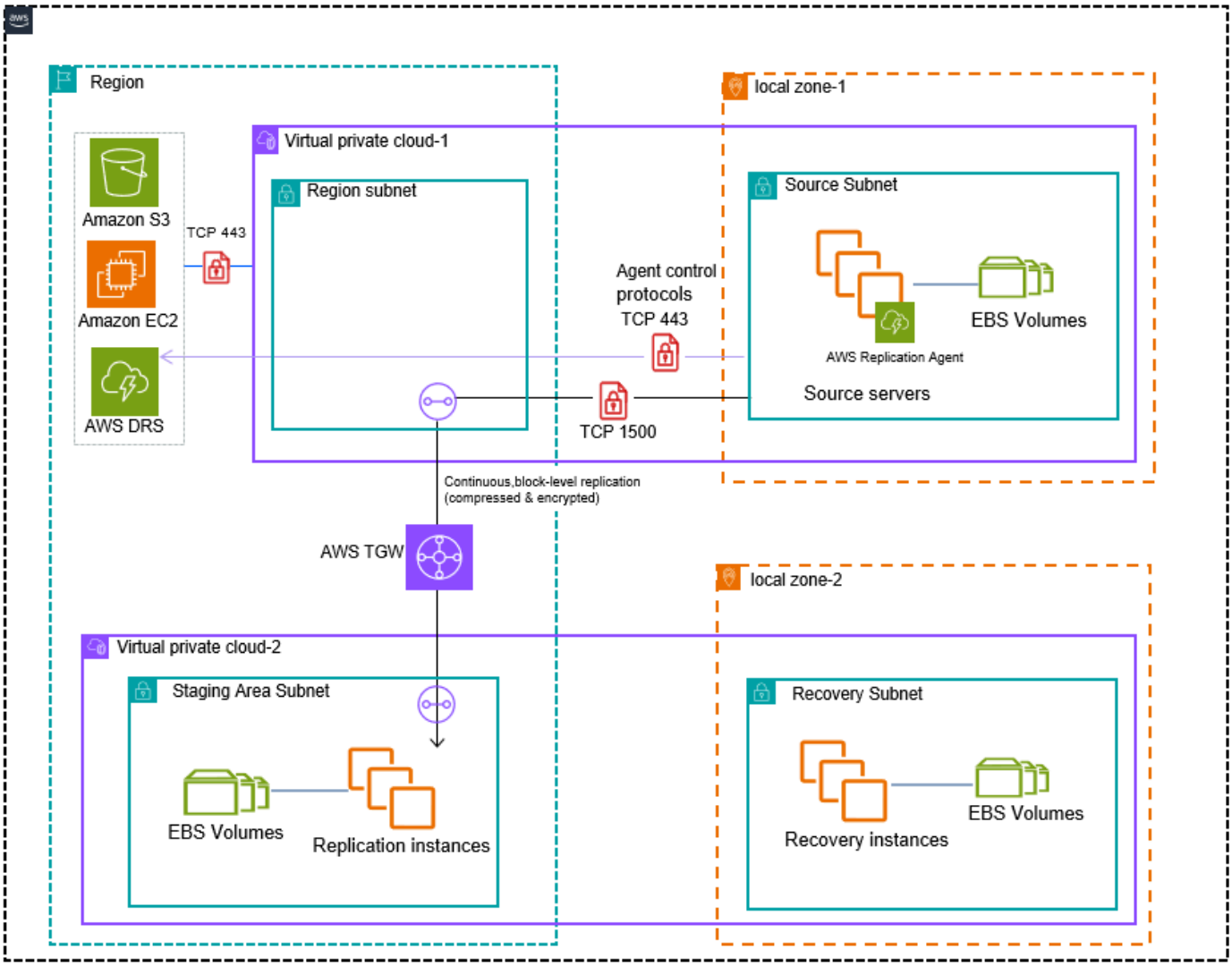
AWS Local Zone to Region

This diagram shows the network architecture of DRS protecting source servers located in an AWS Local Zone. The staging and recovery environment are both located in an AWS Region.



AWS Local Zone to AWS Local Zone

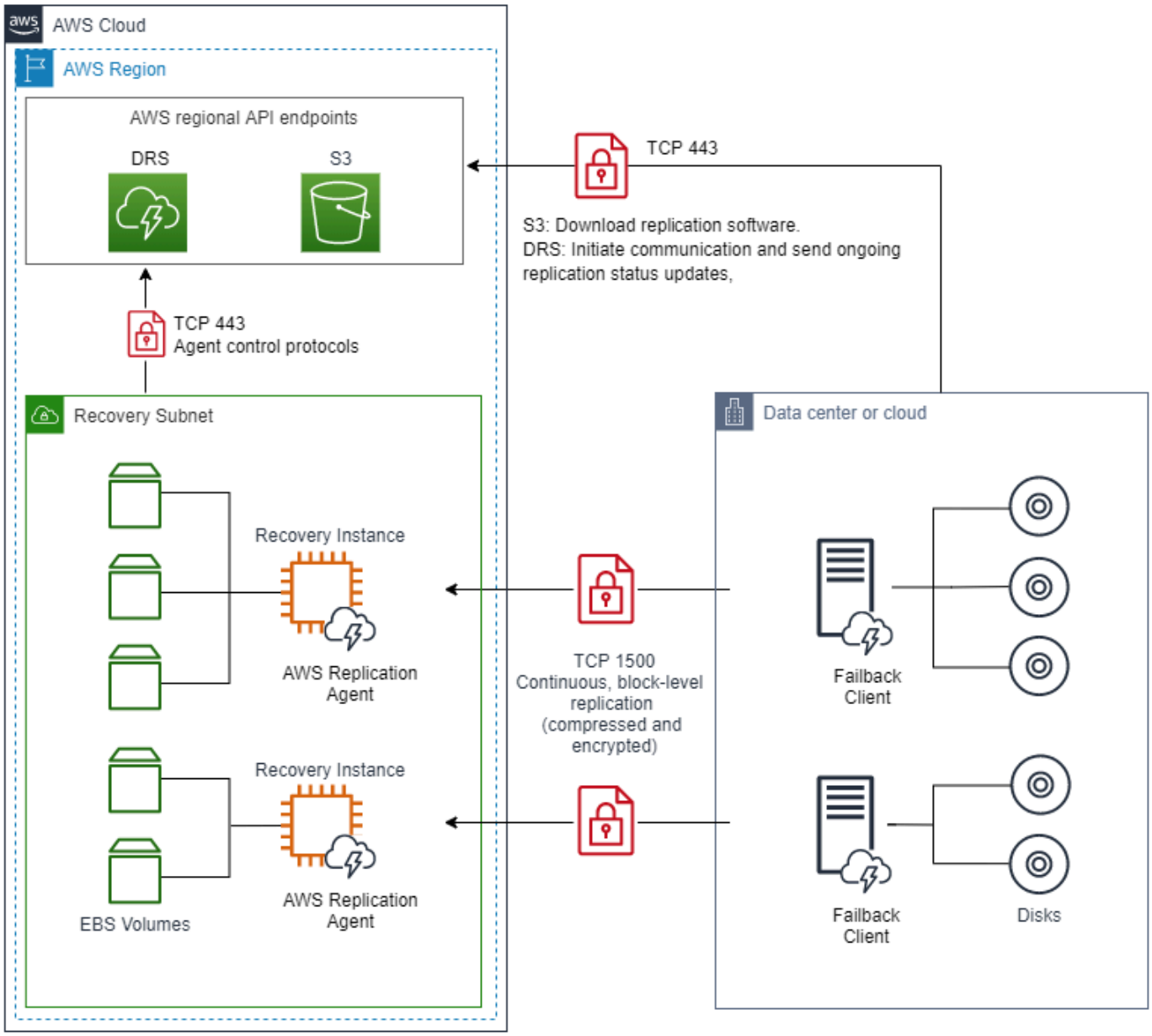
This diagram shows the network architecture of DRS protecting source servers located in an AWS Local Zone. The staging environment is located in an AWS Region and the recovery environment is in another AWS Local Zone.



AWS Failback to On-Prem

This diagram shows the network architecture of DRS performing Failback to an on-premises environment after performing a recovery into AWS.

AWS Elastic Disaster Recovery (AWS DRS) failback replication – architecture and networking



Elastic Disaster Recovery network setting preparations

Topics

- Staging area subnet
- Network requirements
- Operational subnets

Staging area subnet

Before setting up AWS Elastic Disaster Recovery (AWS DRS), you should create a subnet which will be used by AWS DRS as a staging area for data replicated from your source servers to AWS. You must specify this subnet in the replication template. You can override this subnet for specific source servers in the replication settings. While you can use an existing subnet in your AWS account, the best practice is to create a new dedicated subnet for this purpose. Learn more about replication settings.

> **Note**
>
> When planning to recover into an AWS Local Zone, we recommend setting the staging area subnet within the AWS Region, and not the AWS Local Zone. This ensures optimal launch conditions for the replication servers and conversion servers involved in the recovery process. However, if your recovery requirements necessitate the use of the AWS Local Zone for the staging area subnet, we recommend performing recovery drills to validate that you can replicate and recover your production workloads without any issues.

Network requirements

The replication servers launched by AWS Elastic Disaster Recovery in your staging area subnet need to be able to send data over TCP port 443 to the AWS Elastic Disaster Recovery API endpoint at https://drs.{region}.amazonaws.com/. Replace “{region}” with the AWS Region code you are replicating to, for example “us-east-1” .

The source servers on which the AWS Replication Agent is installed need to be able to send data over TCP port 1500 to the Replication Servers in the staging area subnet. They also need to be able to send data to AWS DRS's API endpoint at https://drs.{region}.amazonaws.com/. Replace “{region}” with the AWS Region code you are replicating to, for example “us-east-1” .

Operational subnets

Drill and recovery instances are launched in a subnet you specify in the Amazon EC2 launch template associated with each source server. The Amazon EC2 launch template is created automatically when you add a source server to AWS Elastic Disaster Recovery. Learn more about launching drill and recovery instances. Learn more about Amazon EC2 launch templates.

Elastic Disaster Recovery network requirements

To prepare your network for running Elastic Disaster Recovery, set these connectivity settings:

> **Note**
>
> All communication is encrypted with TLS.

Communication over TCP Port 443:

Topics

- Communication over TCP port 443
- Communication between the source servers and Elastic Disaster Recovery over TCP port 443
- Communication between the staging area subnet and AWS Elastic Disaster Recovery over TCP port 443
- Communication between the source servers and the Staging Area Subnet over TCP port 1500

Communication over TCP Port 1500:

- Between the Source Machines and the staging area Subnet

Communication over TCP port 443

Add these IP addresses and URLs to your firewall:

The Elastic Disaster Recovery AWS Region-specific Console address:

- (drs.<region>.amazonaws.com *example: drs.eu-west-1.amazonaws.com*)

Amazon S3 service URLs (required for downloading AWS Elastic Disaster Recovery software)

- The AWS Replication Agent installer should have access to the S3 bucket URL of the AWS Region you are using with Elastic Disaster Recovery.
- The staging area subnet should have access to S3.
- Allow these S3 buckets:

```
https://aws-drs-clients-<REGION>.s3.<REGION>.amazonaws.com/
https://aws-drs-clients-hashes-<REGION>.s3.<REGION>.amazonaws.com/
https://aws-drs-internal-<REGION>.s3.<REGION>.amazonaws.com/
https://aws-drs-internal-hashes-<REGION>.s3.<REGION>.amazonaws.com/
https://aws-elastic-disaster-recovery-<REGION>.s3.<REGION>.amazonaws.com/
https://aws-elastic-disaster-recovery-hashes-<REGION>.s3.<REGION>.amazonaws.com/
https://al2023-repos-<REGION>-de612dc2.s3.<REGION>.amazonaws.com/
```

Note

- Agent installation and replication server components require an Amazon S3 bucket for service functionality.
- Ensure the relevant VPC endpoint policy includes access to all the required Amazon S3 buckets.

Important

Ensure that your Amazon S3 VPC gateway endpoint policy allows access to the Amazon Linux 2023 package repository bucket (`al2023-repos-<REGION>-de612dc2`). Replication servers run Amazon Linux 2023 and require access to this repository for package installation and updates.

When using an S3 VPC Endpoint, you must provide sufficient permissions for service functionality. See example policy for replicating to us-east-1:

JSON

```
{
 "Version":"2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Principal": {
```

```
    "AWS": "*"
   },
   "Action": "s3:GetObject",
   "Resource": [
    "arn:aws:s3:::aws-drs-clients-us-east-1/*",
    "arn:aws:s3:::aws-drs-clients-hashes-us-east-1/*",
    "arn:aws:s3:::aws-drs-internal-us-east-1/*",
    "arn:aws:s3:::aws-drs-internal-hashes-us-east-1/*",
    "arn:aws:s3:::aws-elastic-disaster-recovery-us-east-1/*",
    "arn:aws:s3:::aws-elastic-disaster-recovery-hashes-us-east-1/*",
    "arn:aws:s3:::al2023-repos-us-east-1-de612dc2/*"
   ]
  }
 ]
}
```

AWS specific

The staging area subnet requires outbound access to the Amazon EC2 endpoint of its AWS Region.

TCP port 443 is used for two communication routes:

1. Between the source servers and Elastic Disaster Recovery.

2. Between the staging area subnet and AWS Elastic Disaster Recovery.

Communication between the source servers and Elastic Disaster Recovery over TCP port 443

Each source server that is added to AWS Elastic Disaster Recovery (AWS DRS) must continuously communicate with AWS DRS (DRS.<region>.amazonaws.com) over TCP port 443.

These are the main operations performed through TCP port 443:

- Downloading the AWS Replication Agent on the source servers.
- Upgrading installed Agents.
- Connecting the source servers to the AWS DRS Console and displaying their replication status.
- Monitoring the source servers for internal troubleshooting and the use of resource consumption metrics (such as CPU, RAM).
- Reporting source server-related events (for example, a removal or resizing of a disk).

- Transmitting source server-related information to the AWS DRS Console (including hardware information, running services, installed applications and packages, and more).
- Preparing the source servers for drill or recovery.

> **Important**
>
> Make sure that your corporate firewall allows connections over TCP port 443.

Solving communication problems over TCP port 443 between the source servers and AWS Elastic Disaster Recovery

If there is no connection between your source servers and AWS Elastic Disaster Recovery, make sure that your corporate firewall facilitates connectivity from the source servers to AWS Elastic Disaster Recovery over TCP Port 443. If the connectivity is blocked, activate it.

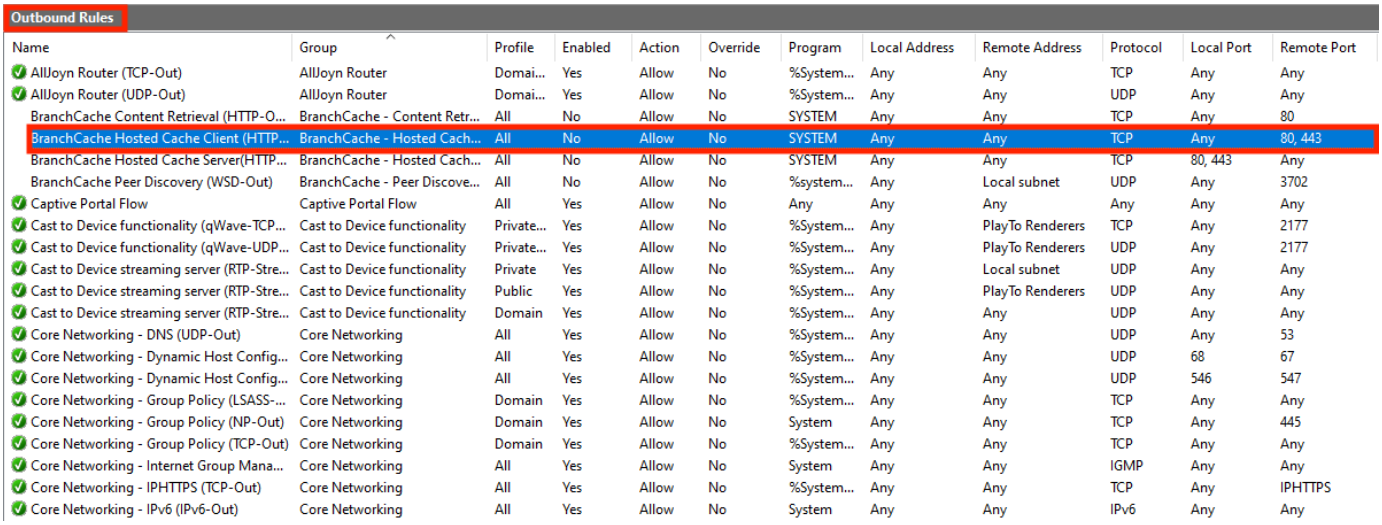
Enabling Windows Firewall for TCP port 443 connectivity

> **Important**
>
> The information provided in this section is for general security and firewall guidance only. The information is provided on "AS IS" basis, with no guarantee of completeness, accuracy or timeliness, and without warranty or representations of any kind, expressed or implied. In no event will AWS and/or its subsidiaries and/or their employees or service providers be liable to you or anyone else for any decision made or action taken in reliance on the information provided here or for any direct, indirect, consequential, special or similar damages (including any kind of loss), even if advised of the possibility of such damages. AWS is not responsible for the update, validation or support of security and firewall information.

> **Note**
>
> Enabling Windows Firewall for TCP port 443 connectivity allows your servers to achieve outbound connectivity. You may still need to adjust other external components, such as firewall blocking or incorrect routes, in order to achieve full connectivity.

> **Note**
>
> These instructions are intended for the default OS firewall. Consult the documentation of any third-party local firewall you use to learn how to enable TCP port 443 connectivity.

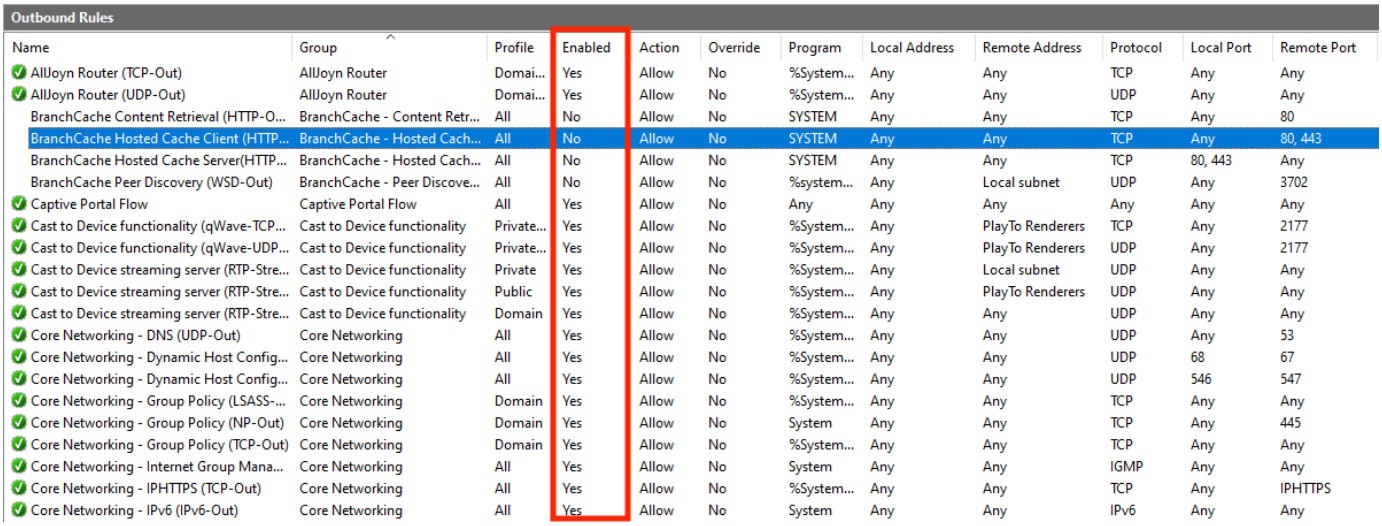
1. On the source server, open the **Windows Firewall** console.
2. On the console, select the **Outbound Rules** option from the tree.



Outbound Rules

Name	Group	Profile	Enabled	Action	Override	Program	Local Address	Remote Address	Protocol	Local Port	Remote Port
AllJoyn Router (TCP-Out)	AllJoyn Router	Domai...	Yes	Allow	No	%System...	Any	Any	TCP	Any	Any
AllJoyn Router (UDP-Out)	AllJoyn Router	Domai...	Yes	Allow	No	%System...	Any	Any	UDP	Any	Any
BranchCache Content Retrieval (HTTP-O...	BranchCache - Content Retr...	All	No	Allow	No	SYSTEM	Any	Any	TCP	Any	80
BranchCache Hosted Cache Client (HTTP...	BranchCache - Hosted Cach...	All	No	Allow	No	SYSTEM	Any	Any	TCP	Any	80, 443
BranchCache Hosted Cache Server(HTTP...	BranchCache - Hosted Cach...	All	No	Allow	No	SYSTEM	Any	Any	TCP	80, 443	Any
BranchCache Peer Discovery (WSD-Out)	BranchCache - Peer Discove...	All	No	Allow	No	%system...	Any	Local subnet	UDP	Any	3702
Captive Portal Flow	Captive Portal Flow	All	Yes	Allow	No	Any	Any	Any	Any	Any	Any
Cast to Device functionality (qWave-TCP...	Cast to Device functionality	Private...	Yes	Allow	No	%System...	Any	PlayTo Renderers	TCP	Any	2177
Cast to Device functionality (qWave-UDP...	Cast to Device functionality	Private...	Yes	Allow	No	%System...	Any	PlayTo Renderers	UDP	Any	2177
Cast to Device streaming server (RTP-Stre...	Cast to Device functionality	Private	Yes	Allow	No	%System...	Any	Local subnet	UDP	Any	Any
Cast to Device streaming server (RTP-Stre...	Cast to Device functionality	Public	Yes	Allow	No	%System...	Any	PlayTo Renderers	UDP	Any	Any
Cast to Device streaming server (RTP-Stre...	Cast to Device functionality	Domain	Yes	Allow	No	%System...	Any	Any	UDP	Any	Any
Core Networking - DNS (UDP-Out)	Core Networking	All	Yes	Allow	No	%System...	Any	Any	UDP	Any	53
Core Networking - Dynamic Host Config...	Core Networking	All	Yes	Allow	No	%System...	Any	Any	UDP	68	67
Core Networking - Dynamic Host Config...	Core Networking	All	Yes	Allow	No	%System...	Any	Any	UDP	546	547
Core Networking - Group Policy (LSASS-...	Core Networking	Domain	Yes	Allow	No	%System...	Any	Any	TCP	Any	Any
Core Networking - Group Policy (NP-Out)	Core Networking	Domain	Yes	Allow	No	System	Any	Any	TCP	Any	445
Core Networking - Group Policy (TCP-Out)	Core Networking	Domain	Yes	Allow	No	%System...	Any	Any	TCP	Any	Any
Core Networking - Internet Group Mana...	Core Networking	All	Yes	Allow	No	System	Any	Any	IGMP	Any	Any
Core Networking - IPHTTPS (TCP-Out)	Core Networking	All	Yes	Allow	No	%System...	Any	Any	TCP	Any	IPHTTPS
Core Networking - IPv6 (IPv6-Out)	Core Networking	All	Yes	Allow	No	System	Any	Any	IPv6	Any	Any

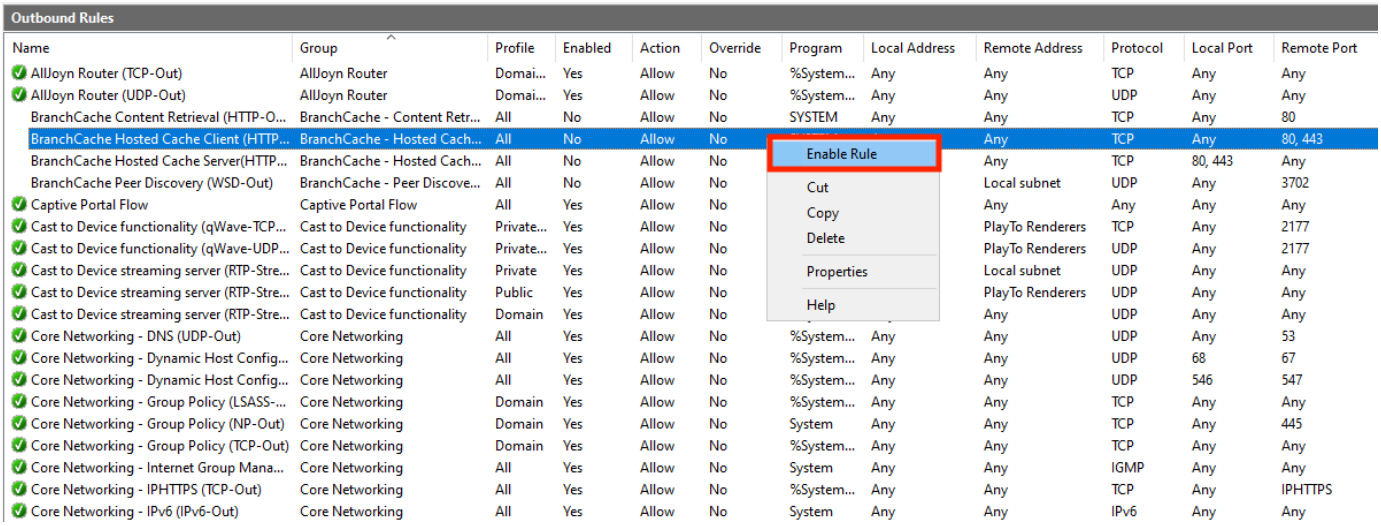
3. On the **Outbound Rules** table, select the rule that relates to the connectivity to Remote Port - 443. Check if the **Enabled** status is **Yes**.



Outbound Rules

Name	Group	Profile	Enabled	Action	Override	Program	Local Address	Remote Address	Protocol	Local Port	Remote Port
AllJoyn Router (TCP-Out)	AllJoyn Router	Domai...	Yes	Allow	No	%System...	Any	Any	TCP	Any	Any
AllJoyn Router (UDP-Out)	AllJoyn Router	Domai...	Yes	Allow	No	%System...	Any	Any	UDP	Any	Any
BranchCache Content Retrieval (HTTP-O...	BranchCache - Content Retr...	All	No	Allow	No	SYSTEM	Any	Any	TCP	Any	80
BranchCache Hosted Cache Client (HTTP...	BranchCache - Hosted Cach...	All	No	Allow	No	SYSTEM	Any	Any	TCP	Any	80, 443
BranchCache Hosted Cache Server(HTTP...	BranchCache - Hosted Cach...	All	No	Allow	No	SYSTEM	Any	Any	TCP	80, 443	Any
BranchCache Peer Discovery (WSD-Out)	BranchCache - Peer Discove...	All	No	Allow	No	%system...	Any	Local subnet	UDP	Any	3702
Captive Portal Flow	Captive Portal Flow	All	Yes	Allow	No	Any	Any	Any	Any	Any	Any
Cast to Device functionality (qWave-TCP...	Cast to Device functionality	Private...	Yes	Allow	No	%System...	Any	PlayTo Renderers	TCP	Any	2177
Cast to Device functionality (qWave-UDP...	Cast to Device functionality	Private...	Yes	Allow	No	%System...	Any	PlayTo Renderers	UDP	Any	2177
Cast to Device streaming server (RTP-Stre...	Cast to Device functionality	Private	Yes	Allow	No	%System...	Any	Local subnet	UDP	Any	Any
Cast to Device streaming server (RTP-Stre...	Cast to Device functionality	Public	Yes	Allow	No	%System...	Any	PlayTo Renderers	UDP	Any	Any
Cast to Device streaming server (RTP-Stre...	Cast to Device functionality	Domain	Yes	Allow	No	%System...	Any	Any	UDP	Any	Any
Core Networking - DNS (UDP-Out)	Core Networking	All	Yes	Allow	No	%System...	Any	Any	UDP	Any	53
Core Networking - Dynamic Host Config...	Core Networking	All	Yes	Allow	No	%System...	Any	Any	UDP	68	67
Core Networking - Dynamic Host Config...	Core Networking	All	Yes	Allow	No	%System...	Any	Any	UDP	546	547
Core Networking - Group Policy (LSASS-...	Core Networking	Domain	Yes	Allow	No	%System...	Any	Any	TCP	Any	Any
Core Networking - Group Policy (NP-Out)	Core Networking	Domain	Yes	Allow	No	System	Any	Any	TCP	Any	445
Core Networking - Group Policy (TCP-Out)	Core Networking	Domain	Yes	Allow	No	%System...	Any	Any	TCP	Any	Any
Core Networking - Internet Group Mana...	Core Networking	All	Yes	Allow	No	System	Any	Any	IGMP	Any	Any
Core Networking - IPHTTPS (TCP-Out)	Core Networking	All	Yes	Allow	No	%System...	Any	Any	TCP	Any	IPHTTPS
Core Networking - IPv6 (IPv6-Out)	Core Networking	All	Yes	Allow	No	System	Any	Any	IPv6	Any	Any

4. If the Enabled status of the rule is **No**, right-click it and select **Enable Rule** from the pop-up menu.



Enabling Linux Firewall for TCP port 443 connectivity

1. Enter this command to add the required Firewall rule:

   sudo iptables -A OUTPUT -p tcp --dport 443 -j ACCEPT

2. To verify the creation of the Firewall rule, enter these commands:

   sudo iptables -L

   Chain INPUT (policy ACCEPT)

   target prot opt source destination

   Chain FORWARD (policy ACCEPT)

   target prot opt source destination

   Chain OUTPUT (policy ACCEPT)

   target prot opt source destination

   ACCEPT tcp -- anywhere anywhere tcp dpt:443

Communication between the staging area subnet and AWS Elastic Disaster Recovery over TCP port 443

The replication servers in the staging area subnet must continuously communicate with Elastic Disaster Recovery over TCP port 443. The main operations that are performed through this route are:

- Downloading the replication software by the replication servers.
- Connecting the replication servers to AWS Elastic Disaster Recovery, and displaying their replication status.
- Monitoring the replication servers for internal troubleshooting use and resource consumption metrics (such as CPU, RAM).
- Reporting replication-related events.

> **Note**
>
> The staging area subnet requires S3 access.

Configuring communication over TCP port 443 between the staging area subnet and AWS Elastic Disaster Recovery

You can establish communication between the staging area subnet and AWS Elastic Disaster Recovery over TCP port 443 directly.

There are two ways to establish direct connectivity to the Internet for the VPC of the staging area, as described in the VPC FAQ.

1. Public IP address + Internet gateway
2. Private IP address + NAT instance

Communication between the source servers and the Staging Area Subnet over TCP port 1500

Each source server with an installed AWS Replication Agent continuously communicates with the AWS Elastic Disaster Recovery replication servers in the staging area subnet over TCP port 1500. TCP port 1500 is needed for the transfer of replicated data from the source servers to the staging area subnet.

The replicated data is encrypted and compressed when transferred over TCP port 1500. Prior to being moved into the Staging Area Subnet, the data is encrypted on the source infrastructure. The data is decrypted after it arrives at the staging area subnet and before it is written to the volumes.

TCP port 1500 is primarily used for the replication server data replication stream.

Elastic Disaster Recovery uses TLS 1.2 end to end from the agent installed on the source server to the replication server. Each replication server gets assigned a specific TLS server certificate, which is distributed to the corresponding agent and validated against on the agent side.

Establishing communication over TCP port 1500

> **Important**
>
> To allow traffic over TCP port 1500, make sure that your corporate firewall enables this connectivity.

Required bandwidth between the source servers and the staging area subnet

Replicated data is transferred from the source servers to the staging area over the network. For replication to succeed, your average network bandwidth must be higher than the write rate on the source servers. If you attempt to conduct a replication of a write intensive source server under low bandwidth conditions, it will likely lag.

AWS Elastic Disaster Recovery (AWS DRS) settings

AWS Elastic Disaster Recovery includes multiple configuration options for resources consumed and produced by the service. There are three main categories that these configuration options fall into:

- **Replication Settings** - Configuration Options for Replication Servers.
- **Launch Settings** - Configuration Options for Source Server Launches.
- **Post Launch Actions** - SSM Documents associated with Source Servers after Recovery Instances launch.

Launch Settings and **Replication Settings** are configurable on an individual Source Server, as well as defaults for a Region. **Default launch** settings and **Default replication** settings are initially configured while Initializing the AWS Elastic Disaster Recovery Service. A newly added Source Server is implicitly configured with the same settings defined in the **Default launch** settings and **Default replication** settings upon installation. You can adjust the individual configuration of a Source Server's Settings anytime after installation.

Topics

- AWS DRS replication settings
- AWS DRS launch settings
- Configuring the default post-launch actions

AWS DRS replication settings

Replication Settings govern how data is replicated from Source Servers and stored within AWS.

These topics address the different types of replication settings:

Topics

- AWS DRS default replication
- AWS DRS individual replication settings

AWS DRS default replication

Default replication settings are created during the DRS Service Initialization within a Region. Learn more about configuring your **Default replication** settings. The options configured within the **Default replication** automatically apply to any newly added Source Server. Any changes made to the **Default replication** only apply to any Source Server added after the changes were made, they do not automatically update the corresponding settings on existing Source Servers.

Most Replication Settings can be configured through the Default replication settings:

Replication setting	Default replication
Staging area subnet	Supported
Replication server instance type	Supported
EBS volume type	Supported
EBS encryption	Supported
Automatically replicate new disks	Supported
Always use AWS Elastic Disaster Recovery security group	Supported
Security Group	Supported
Dedicated instance for replication server	Unsupported
Data Routing (Private IP)	Supported
IP Version	Supported
Create public IP	Supported
Network Bandwidth Throttling	Supported
Point in time (PIT) policy	Supported
MAP program tagging	Supported

Replication setting	Default replication
Tags	Supported

AWS DRS individual replication settings

AWS Elastic Disaster Recovery attempts to reduce costs by consolidating the replication of as many source servers as possible onto the same Replication Server based on the individual Source Server **Replication Settings**. Source Servers must have identical **Replication Settings** to be considered for consolidation, and must not have Use dedicated replication instance enabled. For example, DRS does not consolidate Source Servers that have a different **Staging area subnet** specified in their **Replication Settings**. To reduce EC2 usage, we recommend having as many as possible Source Servers have identical **Replication Settings** to one another.

Modifying the **Replication Settings** of an existing Source Server can impact existing replication, depending on the settings configured. Additionally, most **Replication Settings** options can be modified in bulk for multiple Source Servers through the AWS Elastic Disaster Recovery Console:

Replication Setting	Impact	Bulk Editing
Staging area subnet	Small pause while reconnecting Source Server to new Replicator.	Supported
Replication server instance type	Small pause while reconnecting Source Server to new Replicator.	Supported
Dedicated instance for replication server	Small pause while reconnecting Source Server to new Replicator.	Supported
EBS encryption	Full Sync may be required.	Supported

Replication Setting	Impact	Bulk Editing
Data Routing (Private IP)	No impact.	Supported
IP Version	Small pause while reconnecting Source Server to new Replicator.	Supported
Network Bandwidth Throttling	No impact.	Supported
Point in time (PIT) policy	Replication server is disconnected as a safety measure. This ensures proper handling of retention policy changes that might affect replication state.	Supported
MAP program tagging	No impact.	Supported
Tags	Small pause while reconnecting Source Server to new Replicator.	Supported

Replication server configuration

Replication Servers are AWS EC2 Instances automatically launched by AWS Elastic Disaster Recovery to support Continuous Data Replication from Source Servers.

Staging area subnet

The **Staging area subnet** setting defines which VPC Subnet that the Replication Server for a Source Server uses. A Source Server must be able to successfully initialize connections to the subnet configured within its **Staging area subnet** setting. The best practice is to create a single dedicated, separate subnet for recovery in your AWS Account. Learn more about creating subnets in this AWS VPC article. Unless Use private ip is enabled and valid routing within the VPC exists, Replication Servers must be in a Public subnet. By default, a Replication Server assigns itself a Public IPv4 without any additional configuration needed.

DRS Console

Updating the Staging area subnet

1. Navigate to the AWS Elastic Disaster Recovery Console. In the left navigation pane, select **Source Servers**.
2. Select one or more source servers, then select **Replication**.
3. Select **Edit replication settings**.
4. Navigate to **Replication server configuration**, then select the drop down box under **Staging area subnet**.
5. Select a new VPC Subnet from the drop down box.
6. Save settings by selecting **Save replication settings**.

Command Line

Updating the Staging area subnet

- Updating the Staging area subnet via command line
  - describe-recovery-instances (AWS CLI)

```
aws drs describe-recovery-instances --source-server-id s-123456789abcdefgh
 --staging-area-subnet-id subnet-123456789abcd
```

  - Update-EDRSReplicationConfiguration (DRS Tools for Windows PowerShell)

```
Update-EDRSReplicationConfiguration -SourceServerID s-123456789abcdefgh -
StagingAreaSubnetId subnet-123456789abcd
```

Replication server instance type

The **Replication server instance type** determines the EC2 Instance type and size that is used for the launch of a source server's replication server. AWS Elastic Disaster Recovery replication servers support only x86_64 Amazon EC2 instance types. This applies even when the source server uses `arm64`.

By default, AWS Elastic Disaster Recovery utilizes the t3.small instance type, and should work well for most common workloads. We recommend monitoring the Cloudwatch metrics of a replication

server, if your Source Server is experiencing frequent Lag or Backlog. Metrics to monitor include EBSWriteBytes or EBSWriteOps, which may indicate the **Replication server instance type** is improperly sized to protect your source server.

AWS Elastic Disaster Recovery supports replicating Source Servers with up to 60 volumes, however the **Replication server instance type** must also support an equal or greater number of EBS Volume attachments. We recommend reviewing the Dedicated Amazon EBS volume limit Documentation to ensure an appropriately sized EC2 Instance Type is selected.

> **Note**
>
> If the configured **Replication server instance type** is unavailable in the staging area subnet's Availability Zone, AWS Elastic Disaster Recovery automatically launches the replication server using an alternative instance type. This behavior maintains continuous data replication and protects your RPO. For more information, see What happens if the configured replication server instance type is unavailable?

DRS Console

Updating the Replication server instance type

1. Navigate to the AWS Elastic Disaster Recovery Console. In the left navigation pane, select **Source Servers**.
2. Select one or more source servers, then select **Replication**.
3. Select **Edit replication settings**.
4. Navigate to **Replication server configuration**, then select the drop down box under **Dedicated instance for replication server**.
5. Select **Replication server instance** from the drop down box.
6. Save settings by selecting **Save replication settings**.

Command Line

Modifying Dedicated instance for replication server

- Updating the Replication server instance type via command line
  - update-replication-configuration (AWS CLI)

```
aws drs update-replication-configuration --source-server-
id s-123456789abcdefgh --replication-server-instance-type m5.large
```

- Update-EDRSReplicationConfiguration (DRS Tools for Windows PowerShell)

```
Update-EDRSReplicationConfiguration -SourceServerID s-123456789abcdefgh -
ReplicationServerInstanceType  m5.large
```

Dedicated instance for replication server

The **Dedicated instance for replication server** setting specifies whether or not the Source Server can use a Replication Server shared with other Source Servers. By default, AWS Elastic Disaster Recovery attempts to consolidate as many Source Servers as possible onto a single Replication Server, based on a variety of factors. Setting **Dedicated instance for replication server** to **use dedicated replication instance** ensures that only this Source Server replicates data to the Replication Server.

We recommend leaving **Dedicated instance for replication server** as **do not use dedicated replication instance** unless the Source Server is experiencing frequent Lag or Backlog due to sharing a Replication Server with other Source Servers. Using a dedicated replication server may increase EC2 costs associated with protecting a Source Server.

DRS Console

Enabling Dedicated instance for replication server

1. Navigate to the AWS Elastic Disaster Recovery Console. In the left navigation pane, select **Source Servers**.
2. Select one or more source servers, then select **Replication**.
3. Select **Edit replication settings**.
4. Navigate to **Replication server configuration**, then select the drop down box under **Dedicated instance for replication server**.
5. Select **use dedicated replication instance** from the drop down box.
6. Save settings by selecting **Save replication settings**.

Command Line

Modifying Dedicated instance for replication server

1. Enabling Dedicated instance for replication server via command line
   - update-replication-configuration (AWS CLI)

     ```
     aws drs update-replication-configuration --source-server-
     id s-123456789abcdefgh --use-dedicated-replication-server
     ```

   - Update-EDRSReplicationConfiguration (DRS Tools for Windows PowerShell)

     ```
     Update-EDRSReplicationConfiguration -SourceServerID s-123456789abcdefgh -
     UseDedicatedReplicationServer $true
     ```

2. Disabling Dedicated instance for replication server via commandline
   - update-replication-configuration (AWS CLI)

     ```
     aws drs update-replication-configuration --source-server-
     id s-123456789abcdefgh --no-use-dedicated-replication-server
     ```

   - Update-EDRSReplicationConfiguration (DRS Tools for Windows PowerShell)

     ```
     Update-EDRSReplicationConfiguration -SourceServerID s-123456789abcdefgh -
     UseDedicatedReplicationServer $false
     ```

Amazon EBS volumes

Set the default Amazon EBS volume type used by the replication servers, whether to use Amazon EBS encryption, and whether to automatically replicate newly added disks.

Amazon EBS volume type

Each disk has minimum and maximum sizes and varying performance metrics and pricing. Learn more about Amazon EBS volume types in this Amazon EBS article. The best practice is to not change the default **Auto volume type selection** volume type, unless there is a business need for doing so.

Choose the default **Amazon EBS volume type** to be used by the replication servers for large disks:

- With **Auto volume type selection** the service dynamically switches between performance/cost optimized volume type according to the replicated disk write throughput.

  > **Note**
  >
  > This option only affects disks over 125 GiB (by default, smaller disks always use Magnetic HDD volumes).

- The default **Lower cost, Throughput Optimized HDD (st1)** option utilizes slower, less expensive disks.

  You may want to use this option if:

  - You want to keep costs low
  - Your large disks do not change frequently
  - You are not concerned with how long the Initial Sync process takes

- The **Faster, General Purpose SSD (gp2)** and Faster, **General Purpose SSD (gp3)** options utilize faster, but more expensive disks.

  You may want to use this option if:

  - Your source server has disks with a high write rate or if you want faster performance in general
  - You want to speed up the initial sync process
  - You are willing to pay more for speed

> **Note**
>
> You can customize the Amazon EBS volume type used by each disk within each source server in that source server's settings. Learn more about changing individual source server volume types.

Amazon EBS encryption

Choose your encryption approach:

- When you choose **Default**, the default key is used. This can be an EBS-managed key or a customer-managed key. This option encrypts your replicated data at rest on the staging area subnet disks and the replicated disks.

- Choose **Custom** and then enter the ARN or key ID of a customer-managed key from your account or another AWS account in the **EBS encryption key** field. Enter the key, such as a cross-account KMS key, in standard key ID format. For example, KMS key format is `1234abcd-12ab-34cd-56ef-1234567890ab`. This option encrypts your replicated data at rest on the staging area subnet disks and the replicated disks.
- Choose **Create an AWS KMS key** to be redirected to the Key Management Service (KMS) Console where you can create a new key to use.

Learn more about EBS Volume Encryption in Amazon EBS encryption.

> **Important**
>
> Changing the encryption option after data replication has started causes data replication to start from the beginning.

Automatic replication of new disks

AWS Elastic Disaster Recovery (AWS DRS) allows you to automatically replicate newly added disks. When you add new disks to your source environment, AWS DRS initiates data replication to the staging area subnet in your AWS account.

Automating replication of new disks assists you in maintaining continuous data replication, saves time and resources, and reduces the risk of data loss in the event of a disruption.

This feature is activated automatically for newly added servers.

To deactivate or reactivate this feature for newly added servers:

- Under **Settings** on the left-hand navigation menu, choose **Default replication settings**.
- Select **Edit**.
- Under **Volumes**, uncheck the **Automatically replicate new disks** checkbox.

To activate or deactivate or reactivate this feature for a specific server:

- Go to the replication settings.
- Select **Edit**.

- Under **Volumes**, uncheck the **Automatically replicate new disks** checkbox.

> **Note**
>
> - This feature is only supported for new agent versions (version 4.6 or higher). For older versions, you must reinstall your agent to activate automatic replication of new disks.
> - Auto replication of new disks is not supported with --force-volumes.
> - It might take up to 10 minutes for new disks to start replicating.
> - New disks are only replicated once the feature is activated and are not replicated retroactively.

Elastic Disaster Recovery security groups

A security group acts as a virtual firewall, which controls the inbound and outbound traffic of the staging area. We recommend that you have AWS Elastic Disaster Recovery automatically attach and monitor the default Elastic Disaster Recovery security group. This group opens inbound TCP Port 1500 for receiving the transferred replicated data. When you use the default Elastic Disaster Recovery security group, Elastic Disaster Recovery constantly monitors whether the rules within this security group are enforced, in order to maintain uninterrupted data replication. If these rules are altered, Elastic Disaster Recovery automatically fixes the issue. Choose:

- Recommended - Select **Always use AWS Elastic Disaster Recovery security group** to allow data to flow from your source servers to the replication servers, and so that the replication servers can communicate their state to the AWS Elastic Disaster Recovery servers.
- Not recommended - Deselect **Always use AWS Elastic Disaster Recovery security group** option. Then, select the drop-down menu to choose from the list of available security groups. The list of available security groups changes according to the **Staging area subnet** that you selected.
  - To search for a specific security group, use the search box.
  - If you add security groups via the AWS Console, they appear on the Security group drop-down list in the AWS Elastic Disaster Recovery Console. Learn more about AWS security groups in this VPC article.
  - Any security group that you select is added to the default AWS Elastic Disaster Recovery group, because the default security group is essential for the operation of AWS Elastic Disaster Recovery.

Data routing and throttling

AWS Elastic Disaster Recovery lets you control how data is routed from your source servers to the replication servers on AWS through the **Data routing and throttling** settings. By default, data is sent from the source servers to the replication servers over the public internet, using the public IPv4 address that was automatically assigned to the replication servers. Transferred data is always encrypted in transit. Choose **Use private IP for data replication...** if you want to route the replicated data from your source servers to the staging area subnet through a private network with a VPN, AWS Direct Connect, VPC peering, or another type of existing private connection. Data replication does not work unless you have already set up the VPN, AWS Direct Connect, or VPC peering in the AWS Console. Use this option if you want to:

- Allocate a dedicated bandwidth for replication;
- Use another level of encryption;
- Add another layer of security by transferring the replicated data from one private IP address (source) to another private IP address (on AWS).

Note

- If you selected the Default subnet, it is unlikely that the Private IP is used for that Subnet. Ensure that Private IP (VPN, AWS Direct Connect, or VPC peering) is used for your chosen subnet if you use this option.
- You can safely select and deselect **Use private IP for data replication....** even after data replication has begun. This switch causes a short pause in replication, and does not have long-term effects on the replication.
- Choosing the **Use Private IP for data replication...** option does not create a new private connection.
- When you select the **Use private IP** option, you choose to **Create public IP**. Public IPs are used by default.

IP version

The **IP version** setting controls the Internet Protocol version that AWS Elastic Disaster Recovery uses for data replication and for communication between your source servers and the staging area. You can choose between **IPv4** (default) and **IPv6**.

When you select **IPv6**, the following changes apply:

- Data replication from the AWS Replication Agent to the replication server uses IPv6.
- The replication server receives an IPv6 address and does not receive a public IPv4 address.
- Communication during drills and recoveries uses IPv6.

Important

Before you select **IPv6**, verify the following prerequisites:

- Your staging area subnet must have an IPv6 CIDR block.
- Your replication server instance type must support IPv6.

There is no automatic fallback to IPv4 for data replication. If IPv6 connectivity between your source servers and the replication servers is unavailable, data replication fails.

Note

When you select **IPv6**, other IP-related options (such as **Use private IP** and **Create public IP**) are hidden in the console. Your prior IPv4 configurations are preserved and take effect if you switch back to **IPv4**.

Note

The **IP version** setting does not affect recovery instance networking. Recovery instances use the networking configuration defined in your launch settings.

Note

If you use the Failback Client for failback to on-premises infrastructure, the Failback Client currently supports IPv4 only. In-AWS failback uses the configured IP version.

The **IP version** setting is separate from the `--dualstack` installer parameter. The `--dualstack` parameter controls which API endpoints the agent uses to communicate with AWS services, and does not change the IP version used for data replication. For more information, see AWS Replication Agent Installer parameters.

Throttle network bandwidth

You can control the amount of network bandwidth used for data replication per server. By default, AWS Elastic Disaster Recovery uses all available network bandwidth over five concurrent connections.

Choose **Throttle network bandwidth...** to control the transfer rate of data sent from your source servers to the replication servers over TCP Port 1500. Enter the bandwidth in Mbps in the bandwidth field

Point in time (PIT) policy

AWS Elastic Disaster Recovery allows you to select the number of days for which point in time snapshots are retained through the **Point in time (PIT) policy** field.

You can select to save PIT snapshots for 1 to 365 days. Saving PIT snapshots for more days allows you more recovery options, but also results in increased costs. Learn more about Point in time.

> **Important**
>
> The PIT policy must contain exactly three rules: one for MINUTE, one for HOUR, and one for DAY. The snapshot frequency intervals and retention durations for the MINUTE rule (`interval=10`, `retentionDuration=60`) and HOUR rule (`interval=1`, `retentionDuration=24`) are fixed and cannot be modified. Only the DAY rule's `retentionDuration` is configurable, with a value from 1 to 365 days.

DRS Console

Adjusting PIT Retention Rate

1. Navigate to the AWS Elastic Disaster Recovery Console. In the left navigation pane, select **Source Servers**
2. Select one or more source servers, then select **Replication**.
3. Select **Edit replication settings**.

4. Navigate to **Point in time (PIT) policy**.
5. Enter a new Integer from 1 to 365 in **Snapshot retention (in days)**.
6. Select **Save replication settings**.

Command Line

Adjusting PIT Retention Rate

- update-replication-configuration (AWS CLI)

```
aws drs update-replication-configuration --source-server-id s-123456789abcdefgh
 --pit-policy
 enabled=true,interval=10,retentionDuration=60,ruleID=1,units="MINUTE"
 enabled=true,interval=1,retentionDuration=24,ruleID=2,units="HOUR"
 enabled=true,interval=1,retentionDuration=14,ruleID=3,units="DAY"
```

Elastic Disaster Recovery tags

Add custom **tags** to resources created by AWS Elastic Disaster Recovery in your AWS account. You can add up to 50 tags.

These are resources required to facilitate data replication, drilling and recovery. Each tag consists of a key and an optional value. You can add a custom tag to all of the AWS resources that are created on your AWS account during the normal operation of AWS Elastic Disaster Recovery.

To add new tags:

1. Choose **Add new tag**.
2. Enter a **custom tag key** and an optional tag value.

Note

- AWS Elastic Disaster Recovery already adds tags to every resource it creates, including service tags and user tags.

  These resources include:

  - Amazon EC2 instances

- Amazon EC2 launch templates
- Amazon EBS volumes
- Snapshots

Learn more about AWS tags in Tag your Amazon EC2 resources.

MAP program tagging

The AWS Migration Acceleration Program (MAP) provides tools that are designed to reduce costs, boost productivity, improve operational resilience and increase business agility.

The DRS MAP program tagging is a feature that allows you to apply MAP program tags to your source servers and replication resources in order to offset the ongoing cost of protecting your servers.

Learn more about the AWS Migration Acceleration Program (MAP).

Select **Add MAP tag to Launched Instances option**, if you want Application Migration Service to automatically tag your launched instances with the tag key and value combination required for the MAP program. Then, specify the MAP tag value that is used in your MAP tagging. Application Migration Service automatically tags your migrated resources with the key: “map-migrated”, and the value of the tag that you provided. For more details about the tag value that should be used here, please refer to the MAP tagging guide provided in your MAP term.

You can choose to add tags to:

- One or more existing source servers and replication resources
- All newly added source servers and replication resources

Adding tags to existing source servers and replication sources

To add tags to one or more existing source servers and replication sources:

- Select the relevant source servers.
- Select **Edit replication settings** from the replication drop-down menu
- Check the box to the left of **Add MAP tag to the source servers and replication resources**.
- Specify the MAP tag value that is used in your MAP tagging.

DRS automatically tags your source servers and replication resources with the tag key "map-migrated" and the value of the tag that you provide.

Adding tags to newly added source servers and replication sources

To add tags to all newly added source servers and replication sources:

- Select **Settings** from the left-hand menu.
- Select **Edit** to change the default replication settings.
- Check the box to the left of **Add MAP tag to the source servers and replication resources** option.
- Specify the MAP tag value that is used in your MAP tagging.
- Select **Save changes**.

AWS Elastic Disaster Recovery automatically tags every newly-added source server and replication resources with the tag key "map-migrated" and the value of the tag that you provide.

For more details about the tag value that should be used here, please refer to the MAP tagging guide provided in your MAP term.

AWS DRS launch settings

The launch settings are a set of instructions that determine how drill or recovery instances are launched in AWS. They include two sections: general launch settings and the EC2 launch template. These settings are created automatically every time you add a source server to AWS Elastic Disaster Recovery. The launch settings can be modified at any time, even before a specific source server has completed its initial sync.

Topics

- Default AWS DRS launch settings
- Default Amazon Elastic Compute Cloud (Amazon EC2) launch template

Default AWS DRS launch settings

AWS Elastic Disaster Recovery (AWS DRS) allows you to configure the default launch settings and change them at any time.

The default launch settings apply to any new source server added to AWS DRS. You are prompted to configure your default launch settings upon your first use of AWS DRS.

Launch settings can also be edited manually for individual servers.

Editing the default AWS DRS launch settings

The default launch settings are applied to every newly launched source server in AWS Elastic Disaster Recovery (AWS DRS). You can change these settings for a single or multiple servers whenever you choose.

To edit these settings, follow these steps:

1. Select **Default launch** from the left-hand navigation menu (under **Settings**).
2. Select **Edit** in the **Default DRS launch settings** section.
3. Change the settings according to your preferences.
4. Select **Save**.

Launch settings parameters

AWS Elastic Disaster Recovery (AWS DRS) launch settings include:

- **Instance type right sizing –** Allow the service to automatically update the instance type on the EC2 launch template, based on the CPU and RAM of the source server. If this setting is active (default), any modification you make to the instance type in the EC2 launch template is overwritten by the service. For `arm64` right-sizing requirements, see DRS launch settings parameters.
- **Start instance upon launch –** Configure how the EC2 recovery instance should be launched – running or in a stopped state.
- **Copy private IP –** Define whether the private IP should be copied from the source server's primary network interface to the EC2 launch template. If this setting is on, make sure that the subnet defined in the EC2 launch template includes that IP in its range.
- **Transfer server tags –** Define if the launched EC2 instance should have the same tags as the source server resource.
- **Launch into source instance -** Define whether DRS automatically assigns the ID of the source instances to the **Launch into instance ID** field in the Launch Settings of newly added source servers in this region. A source instance is the EC2 instance in this region that was the source of

the data before replication was reversed to this region or availability zone. The EC2 instance to launch into must have a tag with key *AWSDRS* and value *AllowLaunchingIntoThisInstance*, and it must be stopped before launching into it. If **Launch into instance ID** is automatically set for a source server, the **Transfer server tags**, and **Copy private IP** settings need to be deactivated for that server, as they cannot apply to an already launched instance.

> **Note**
>
> Note that for the instance to appear as a recovery instance in DRS, it needs to have an instance profile that includes the policy **AWSElasticDisasterRecoveryRecoveryInstancePolicy**. The role **AWSElasticDisasterRecoveryRecoveryInstanceRole**, which is added to an account when initializing the service, contains this policy and can be used as an instance profile.

Learn more about Launch into source instance.

- **OS licensing –** Choose the launched instance's license type for Windows Servers – License-included or Bring Your Own License (BYOL). Linux servers and Windows Home are automatically launched as BYOL. If you launch a Windows Server or Windows Home as BYOL, you must select Dedicated host for the Tenancy setting in the advanced settings of the EC2 launch template.
- **Recovery mode –** Choose the recovery mode for launching recovery instances:
  - **Optimal** (default) – Runs the full conversion process before launch. The process includes driver injection, boot configuration, and filesystem modifications. Use this mode when you are uncertain about source server compatibility, or for cross-platform scenarios.
  - **Fast** – Skips the conversion process and launches instances directly from replicated snapshots, which reduces recovery time. Use this mode for AWS-to-AWS disaster recovery scenarios where source servers already have AWS-compatible drivers and configurations.

> **Note**
>
> Fast recovery mode requires DRS agent version 6.42.20 or later.

Launch into source instance

This setting is only valid when the replication and recovery are done in-AWS, between 2 AWS regions or availability zones. This default setting applies to newly added source servers. Such

servers have their **Launch into instance ID** field in the Launch Settings set to the EC2 instance ID of the source instance, that was the source of the data in the same region or availability zone. See the examples below for more details.

Pre-requisites

Start reversed replication or **Protect recovered instance** fails to create a source server if this setting is active and one of these conditions is not met:

1. The instance to launch into must have the required tag with key *AWSDRS* and value *AllowLaunchingIntoThisInstance*.
2. The instance to launch into must have the same operating system platform (Linux or Windows) as that of the recovery instance the **Start reversed replication** or **Protect recovered instance** was called on.
3. The instance to launch into must support the boot mode of the recovery instance the **Start reversed replication** or **Protect recovered instance** was called on. If the recovery instance uses the UEFI boot mode, the instance to launch into must also use the UEFI boot mode. If the recovery instance uses the legacy BIOS boot mode, the instance to launch into can use either boot mode. This applies to both Linux and Windows. Elastic Disaster Recovery checks the boot mode when it launches into the instance, and the instance to launch into must report its boot mode to Amazon EC2. Instances launched before March 2023 might not report a boot mode.
4. The instance to launch into must meet these requirements:
   - The same processor architecture as the source server: `x86_64` or `arm64`.
   - Hardware virtual machine (HVM) virtualization.
   - An Amazon Elastic Block Store (Amazon EBS) root device.
5. **OS licensing** in **Default DRS launch settings** can only be **Bring Your Own License (BYOL)** if the instance's platform is Linux or if the instance's **tenancy** is **dedicated host**.
6. **Transfer server tags** and **Copy private IP** must be deactivated in **Default DRS launch settings**.

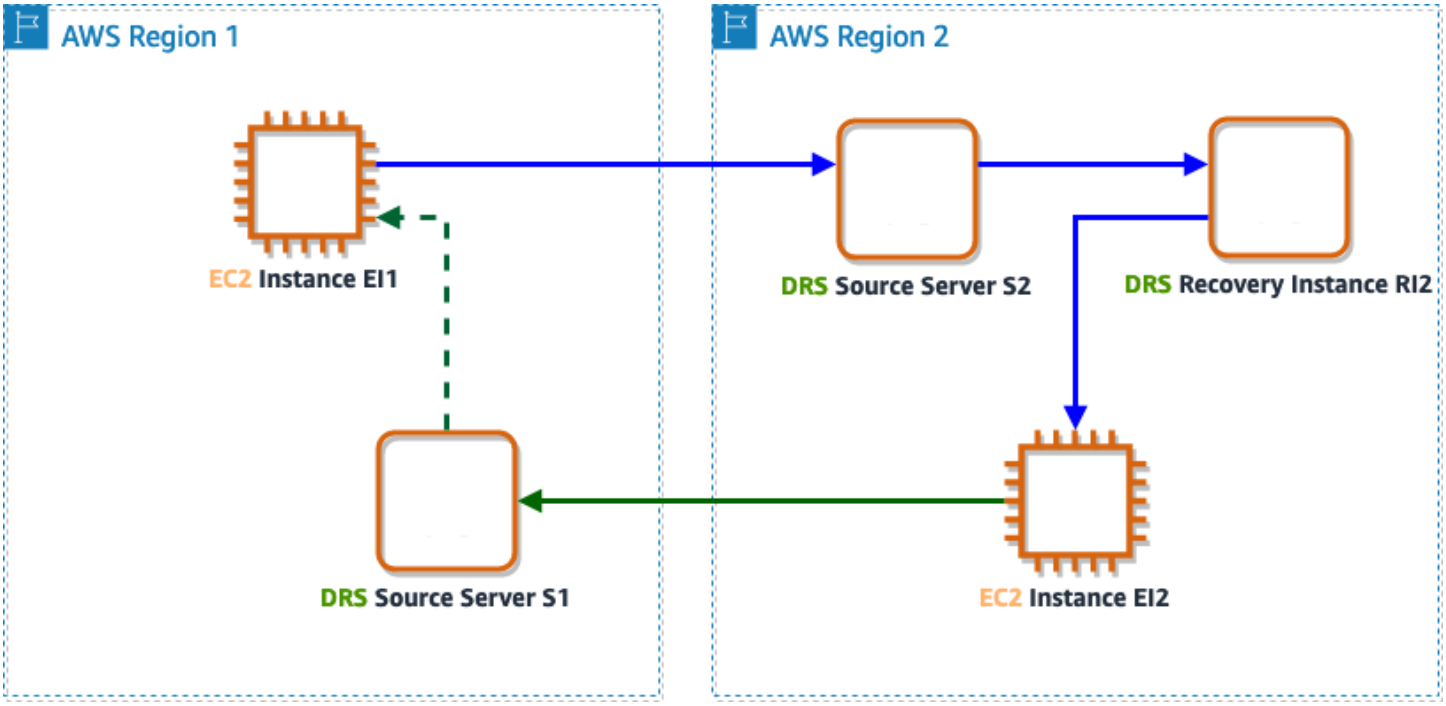
Cross-region

With this setting active, customers who replicate their EC2 instances between two AWS regions, launch in the second region from an instance in the first region, and call **Start reversed replication** to go back to the first region will have their source servers in the first region automatically set **Launch into instance ID** to the instance ID of the instance in the first region they initially launched from.

Using the diagram below as an example, the setting applies to source servers such as Source Server **S1** and automatically sets the **Launch into instance ID** to the instance ID of EC2 instance **EI1** (marked by the dotted green arrow in the diagram).

This only happens if:

1. **Launch into source instance** was set to be active in the **Default DRS launch settings** on region 1.
2. EC2 instance **EI1** (on AWS region 1) replicated into region 2 (replication handled through source server **S2** on AWS region 2), and was launched in AWS region 2 (launch handled through source server **S2** on AWS region 2), creating recovery instance **RI2**.
3. Source server **S1** was created by calling **Start reversed replication** in region 2 on recovery instance **RI2** (marked by the solid green arrow), replicating the data of EC2 instance **EI2**.



Cross Availability Zone

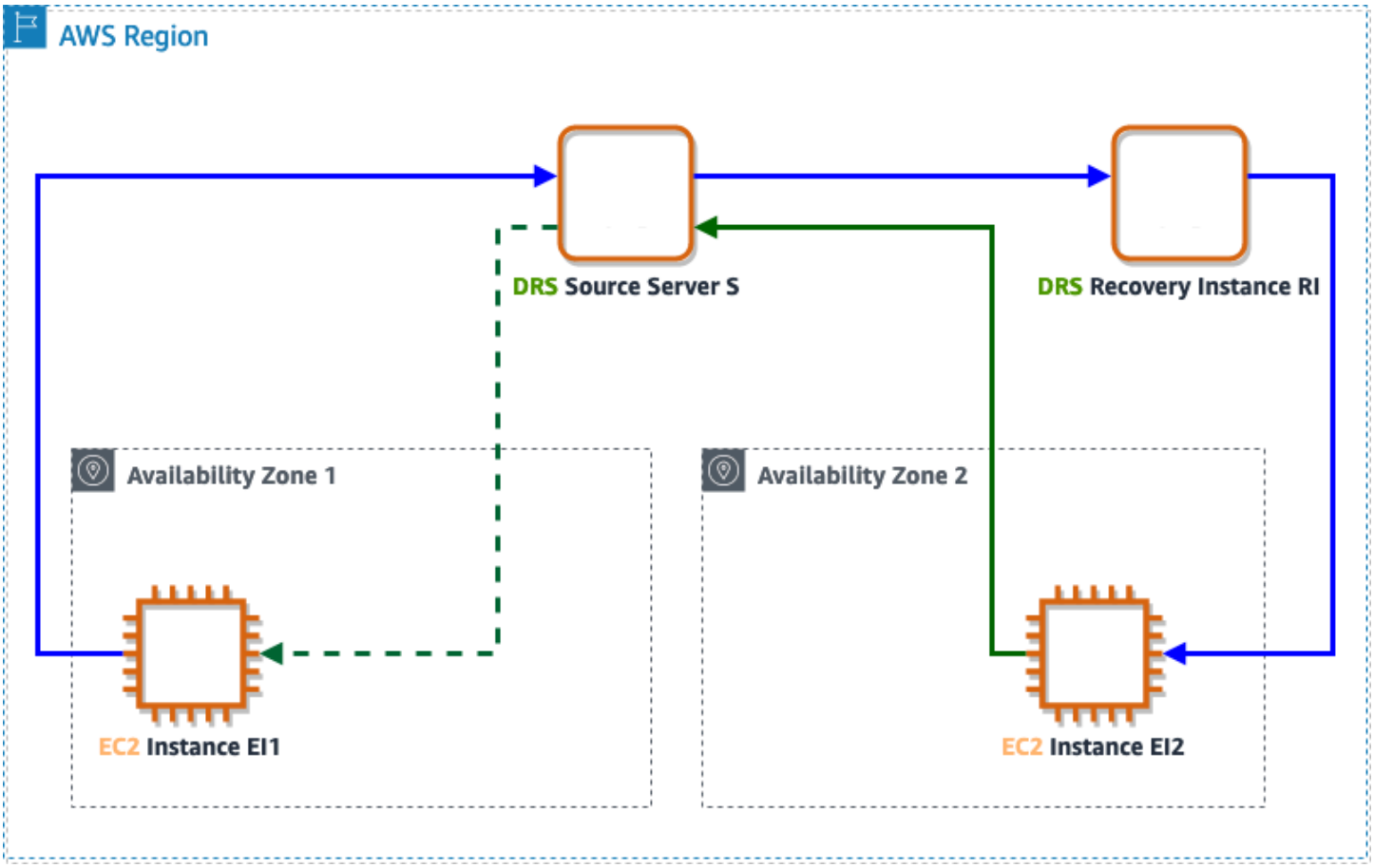
With this setting active, customers who replicate their EC2 instances into the same region (and if following our recommendation, into a different availability zone within that region), launch in the region (recommended to launch into the other availability zone) from an instance in the first availability zone, and perform **Protect recovered instance** on the source server after this launch,

will have the source server automatically set **Launch into instance ID** to the instance ID of the instance in the first availability zone.

Using the diagram below as an example, the setting applies to source servers such as Source Server **S** and automatically sets the **Launch into instance ID** to the instance ID of EC2 instance **EI1** (marked by the dotted green arrow in the diagram).

This only happens if:

1. **Launch into source instance** was set to be active in the **Default DRS launch settings** on this region.
2. EC2 instance **EI1** (on AWS region 1) replicated into availability zone 2 (replication handled through source server **S**), and was launched in availability zone 2 (launch handled through source server **S**), creating recovery instance **RI**.
3. Source server **S** was then updated to protect recovery instance **RI** (marked by the solid green arrow), by calling Protect recovered instance, replicating the data of EC2 instance EI2.



Default Amazon Elastic Compute Cloud (Amazon EC2) launch template

The default Amazon EC2 launch template sets the default values that are copied to EC2 templates created for newly added source servers. This template defines how drill, recovery, or failback instances are launched. If you didn't create any default EC2 template, AWS DRS copies the default values for each setting to EC2 launch templates for newly added servers.

You can usually launch a drill instance without modifying the automatically created EC2 launch template (unless you have removed the default VPC/subnet from your AWS account).

Topics

- Editing the default EC2 launch template

Editing the default EC2 launch template

To edit the default EC2 launch template, follow these steps:

1. Select **Default launch** from the left-hand navigation menu (under **Settings**).
2. Select **Edit** in the **Default EC2 launch template** section.
3. Change the settings according to your preferences.
4. Select **Save**.

Amazon EC2 launch template parameters

AWS Elastic Disaster Recovery (AWS DRS) Amazon EC2 launch settings are divided into basic and advanced settings.

The basic settings include:

- **Subnet –** When you specify a subnet, this field defines where the instance is launched. When selecting a subnet, only the default network interface is updated. If you do not include a subnet, the launched instance uses the Region's default subnet.
- **Security groups –** The selected security groups to assign to the instance, applied to the subnet selected for the default network interface. If no security group is selected, there is no default value and no group is used. Security groups can only be selected if a subnet is included.
- **Instance type –** The default instance type to use when launching. If instance type right-sizing is active, the system disregards this setting. If no instance type is included, a default value is used.

You can either select an instance type, or you can specify instance attributes and let Amazon EC2 identify the instance types with those attributes. For source-server architecture requirements, see Key considerations for EC2 launch templates.

- **EBS volume type –** Applies to all volumes for which this type is relevant. If an unmatching type exists, the default type (GP3) is used instead. Some volume types require setting additional values such as IOPS or throughput.

Instance type attributes:

- **Number of vCPUs:** Enter the minimum and maximum number of vCPUs for your compute requirements. To indicate no limits, mark the no minimum or no maximum checkboxes.
- **Amount of memory (MiB):** Enter the minimum and maximum amount of memory, in MiB, for your compute requirements. To indicate no limits, mark the no minimum or no maximum checkboxes.
- Expand **Optional instance type attributes** and choose **Add attribute** to express your compute requirements in more detail. For information about each attribute, see InstanceRequirementsRequest in the Amazon EC2 API Reference.
- **Preview matching instance types:** You can preview the instance types that match the specified attributes. To exclude instance types, you can select the instance types you want to exclude from the previewed list of instance types, but only if you did not allow list instance types, as you can either exclude or allowlist instance types but not both.

See more about these attributes here: How attribute-based instance type selection works. EC2 uses fleets to launch your instances, and applies price protection to ensure the fleet does not pick expensive instance types for you. We use the default protection settings, so we protect against selecting instance types that are 20% more expensive than the lowest cost instance type. To learn more about price protection using fleets, visit: Price protection.

To learn more about using instance type attributes in DRS, visit Flexible Instance Types

Advanced settings include additional parameters that add specific features to the EC2 template. If you choose not to include these parameters in the template, the specific capabilities are not added.

The advanced settings include:

- **IAM instance profile –** Attach a specific profile to the instance that will be launched. Make sure the instance profile has the AWSElasticDisasterRecoveryRecoveryInstancePolicy IAM policy attached in addition to any other policy.
- **Auto assign public IP –** Automatically assign a public IP to the launched instance.
- **Termination protection –** Protect the launched instance from accidental termination using the EC2 console.
- **Tenancy –** Set tenancy information, such as dedicated host needed in conjunction with setting BYOL for Windows servers and Windows Home.
- **Capacity reservation –** Apply reservation consideration to the launched instances.
- **Key pair –** Associate a key pair with launched instances that are based on EC2 instances.

> **Note**
>
> AWS DRS only supports major EC2 template parameters. If you want to change values that are not supported by this feature, you can still do so by editing the EC2 launch template via the Amazon EC2 console:
>
> - Create a new EC2 template version with the required changes.
> - Mark it as default.

> **Important**
>
> Every time you modify an EC2 launch template on the Amazon EC2 console, a new version is created. AWS DRS uses the version that is marked as the default. If you prefer to use the EC2 launch template you just modified, make sure to mark it as the default. Changes made through the AWS DRS console are automatically set as the default version.

EC2 launch template tags – In addition to the basic and advanced settings, you can also add up to 50 tags. These are transferred to EC2 launch template created by AWS Elastic Disaster Recovery (AWS DRS) service.

Learn more about EC2 launch template settings and configuration options in this EC2 article.

Amazon EC2 template considerations

1. **Revert to previous version –** The right-sizing mechanism can fix issues such as an incorrect instance type, but other issues may still occur. If you encounter any issues with the launch template, you can quickly address them by choosing the original default launch template that was created by AWS DRS when the agent was installed. Alternatively, you can edit the relevant fields from the AWS DRS console.
2. **IOPS –** If needed, set the number of I/O operations per second that the volume can support via the Amazon EC2 console. You can select any number as long as it matches the Amazon EBS guidelines.
   - Provisioned IOPS SSD (io1): 50 IOPS per GiB of storage
   - Provisioned IOPS SSD (io2): 500 IOPS per GiB of storage
   - General Purpose SSD (gp3): 500 IOPS per GiB of storage

Configuring the default post-launch actions

After finishing the AWS DRS initialization process, you can configure your default post-launch actions settings. The default post-launch actions settings apply to newly added source servers and controls which post-launch actions run when launching new instances. These settings are created automatically for each server based on the default settings and can be modified at any time for any individual source server.

You can also use these settings to install the IAM roles required for post-launch actions to work, if the roles were not already installed in your account during the first initialization of AWS DRS. The IAM roles need to be installed once per AWS account, regardless of the region used.

Post-launch actions can be of two different types: command and automation. Command post-launch actions run on the launched instance using the instance profile attached to the instance. Note that if no instance profile is defined on the EC2 launch template, AWS Elastic Disaster Recovery (AWS DRS) places the **AWSElasticDisasterRecoveryRecoveryInstanceWithLaunchActionsRole** instance profile by default if post-launch actions are active for the source server.

Automation actions run with the credentials of the same IAM entity that started the drill, recovery or failback. In addition some automation actions accept a parameter that is sent to the assumeRole key in the SSM document if provided, the action assumes that role for that action execution.

Note

- In order to use post-launch actions, you should make sure you have the required permissions. To get these permissions, you can attach the **AWSElasticDisasterRecoveryLaunchActionsPolicy** or **AWSElasticDisasterRecoveryConsoleFullAccess_v2** policies to your IAM identity. These policies contain the permissions needed to run SSM Command and Automation documents that are owned by Amazon or by the account as post-launch actions.
- Installation of the SSM Agent requires a minimum of 200 MB of free disk space and 200 KB of free disk space in the `/var` directory.
- Installation of the SSM Agent is not supported on these operating systems:
  - CentOS 5.x
  - CentOS 6.x
  - RHEL 6.x
  - Oracle 6.x
  - Amazon Linux 1
  - Windows 2008
  - Windows 2008 R2

Note

In order to run an SSM command or Automation document owned by a different account as a post-launch action you should provide the permission to use `ssm:SendCommand` and `ssm:StartAutomation` on the relevant document.
For example, if you have shared the SSM documents MyCommand (command) and MyAutomation (automation) from account 111111111111, you should attach these permissions to your IAM entities:

Example

```
{
    "Effect": "Allow",
    "Action": [
        "ssm:SendCommand",
```

```
                ],
                "Resource": "arn:aws:ssm:*:111111111111:document/MyCommand",
                "Condition": {
                    "ForAnyValue:StringEquals": {
                        "aws:CalledVia": [
                            "drs.amazonaws.com"
                        ]
                    }
                }
            },
            {
                "Effect": "Allow",
                "Action": [
                    "ssm:StartAutomationExecution"
                ],
                "Resource": "arn:aws:ssm:*:111111111111:automation-definition/
MyAutomation",
                "Condition": {
                    "ForAnyValue:StringEquals": {
                        "aws:CalledVia": [
                            "drs.amazonaws.com"
                        ]
                    }
                }
            }
```

Topics

- Install the required IAM roles if needed
- Activating post-launch actions default settings
- Adding custom actions
- Activating, deactivating and editing predefined or custom actions
- Deleting custom actions
- Predefined post-launch actions
- Validate disk space
- Amazon EC2 connectivity checks
- Verify HTTP/HTTPS response

- Verify tags

Install the required IAM roles if needed

To operate post-launch actions and allow SSM documents to run on launched instances, certain IAM roles must be installed. Usually these roles are installed into an AWS account when AWS DRS is initialized in the account for the first time in any region.

If you have already initialized Elastic Disaster Recovery in your account before September 13, 2023, it's possible that the required IAM roles were not installed in your account.

To verify the IAM roles are installed or install them if not installed (a one-time operation), go to **Settings** → **Default post-launch actions** and check **Post-launch actions settings**. If you see the message **Install the required IAM roles to allow using post-launch actions** select **Install post-launch IAM roles**. If the roles were installed successfully, the message to install the roles is not present in **Post-launch actions settings**.

Activating post-launch actions default settings

Activate post-launch actions in the default settings, to make it active by default for newly added source servers and to enable updating of the default list of actions. Activating and deactivating is only possible if the required IAM roles have been installed.

To activate, make sure the required IAM role is installed by following this guide. After that, go to **Settings** → **Default post-launch actions** and check **Post-launch actions settings** to see if **Post-launch actions** is set to **Active**. In case it is not, select **Edit** and make sure **Post-launch actions activated** is checked. Then select **Save** to store these settings.

To deactivate, go to **Settings** → **Default post-launch actions** and check **Post-launch actions settings** to see if **Post-launch actions** is set to **Not active**. In case it is not, select **Edit** and make sure **Post-launch actions activated** is not checked. Then select **Save** to store these settings.

Adding custom actions

AWS Elastic Disaster Recovery (AWS DRS) allows you to run any SSM document that you like – public SSM documents or ones you created and uploaded to your account. You can configure a custom action to run any SSM document that is available in your account. To be able to create, edit or delete a default custom action, make sure the post-launch actions are activated in the default settings.

Create a custom action

Adding a custom action through the default settings adds it to newly added servers. To add a custom action to an existing source server use the **Post-launch settings** tab in the source server details page. To add a new custom action to the default post-launch action settings, go to **Settings** → **Default post-launch actions**. If the default post-launch actions settings is **Active**, you can create new custom actions by selecting **Add action**.

The **Add action** page includes these parameters:

Action name – The name of the action in AWS DRS, which should be intuitive, meaningful and unique in this AWS account and region.

Activate this action – Use this checkbox to activate or deactivate the action by default. Newly added source servers have the action set to active or not active according to the value this field had when the source server was added.

Mark launch as successful only if this action finishes running successfully – This checkbox dictates whether or not the launch is marked as successful, based on the successful run of this action. Instance launches still progress normally regardless of the success of the action.

Systems Manager document name – Select any Systems Manager document that is available to be used in this account.

View in Systems Manager – select to open **Systems Manager** and view additional information about the document.

Description – Add a description or keep the default.

Document version – Select which SSM document version to run. AWS DRS can run a default version, the latest version, or a specific version, according to your preferences.

Category – Select from various available categories including monitoring, validation, security and more.

Order – Specify the order in which the actions are executed. The lower the number, the earlier the action is executed. Values allowed are between 2 and 10,000. The numbers must be unique but don’t need to be consecutive.

Platform – Taken from the SSM document and reports which Operating System platform (Windows/Linux) is supported by the action.

Creator – Who created the action. For custom actions, the default is always **This account**.

The **Action parameters** change according to the specific SSM document that is selected. Note that for the instance ID parameter, you can choose to use the launch instance ID, in which case, AWS DRS dynamically populates the value.

> **Note**
>
> AWS Elastic Disaster Recovery (AWS DRS) places **AWSElasticDisasterRecoveryRecoveryInstanceWithLaunchActionsRole** instance profile on the launch instance if post-launch actions is active for the source server. If you add an SSM command action that requires additional permissions in the launch instance, you must ensure that the instance profile has the right policies or the right permissions. In order to do so, create a role that has the required permissions as per the policies above or has a policy or policies with those permissions attached to it. Go to **Launch settings** > **EC2 launch template** > **Modify** > **Advanced** > **IAM instance profile**. Use an existing profile or create a new one using the **Create new IAM profile** link.

> **Note**
>
> Only trusted, authorized users should have access to the parameter store. For enhanced security, ensure that users who do not have permissions to execute SSM documents / commands, do not have access to parameter store. Learn more about restricting access to Systems Manager parameters. Action parameters are stored in the SSM parameter store as regular strings. Changing parameters in the SSM Parameter store may impact the post launch action run on target instances. We recommend considering security implications when choosing to use parameters that contain scripts or sensitive information, such as API keys and database passwords.

Activating, deactivating and editing predefined or custom actions

You can activate, deactivate and edit actions available in the default post-launch actions settings. Activating an action in the default settings, adds the action as activated to newly added servers. Likewise, deactivating it, adds it as a non-active action to newly added servers. Source servers already created with this action are not affected by changes in the default settings.

Editing an action in the default settings, adds the edited action to newly added servers. Servers already created with the action before the edit, are not updated with the changes present in any edit to the default settings that was made after their creation. Changes to actions on an existing source server can be made from the **Source server details** page, by going to the **Post-launch settings** tab and performing the change there.

To be able to activate, create, deactivate, edit, or delete a custom action and to activate, deactivate or edit predefined actions, make sure the post-launch actions are activated in the default settings.

Activate, deactivate or edit a post-launch action

To activate, deactivate or edit a post launch action in the default post-launch actions settings, go to **Settings** → **Default post-launch actions**. If **Post-launch actions settings** shows **Post-launch actions** to be **Active**, you can edit any action defined in the default settings.

Locate the action you want to edit in the **Actions** card view, or use the search field to filter the actions by name. Select the action's card to select it, and then select **Edit**.

To activate the action, make sure the **Activate this action** setting is checked and select **Save**. To deactivate, make sure the **Activate this action** setting is un-checked and select **Save**.

The edit page allows to change the value of some of the parameters for both pre-defined actions and custom actions. Some parameters can only be edited if the action is a custom action. See below for specific information.

The parameters that appear on the edit page:

Action name – Editable for custom actions. The name of the action in AWS DRS, which should be intuitive, meaningful and unique in this AWS account and region.

Activate this action – Use this checkbox to activate or deactivate the action by default. Newly added source servers have the action set to active or not active according to the value this field had when the source server was added.

Mark launch as successful only if this action finishes running successfully – This checkbox dictates whether or not the launch is marked as successful, based on the successful run of this action. Instance launches still progress normally regardless of the success of the action.

Systems Manager document name – Editable for custom actions. Select any Systems Manager document that is available to be used in this account.

View in Systems Manager – Select to open **Systems Manager** and view additional information about the document.

Description – Editable for custom actions. Add a description or keep the default.

Document version – Editable for custom actions. Select which SSM document version to run. AWS DRS can run a default version, the latest version, or a specific version, according to your preferences.

Category – Editable for custom actions. Select from various available categories including monitoring, validation, security and more.

Order – Specify the order in which the actions run. The lower the number, the earlier the action runs. Values allowed are between 2 and 10,000. The numbers must be unique but don't need to be consecutive.

Platform – Not editable. Taken from the SSM document and reports which Operating System platform (Windows/Linux) is supported by the action.

Creator – Not editable. Who created the action. For custom actions, the default is always **This account**.

The **Action parameters** change according to the specific SSM document that is selected. Note that for the instance ID parameter, you can choose to use the launch instance ID, in which case, AWS DRS dynamically populates the value. Some predefined actions, where applicable allow to use a dynamically populated value for the volumes. This value is dynamically populated by AWS DRS with the volumes of the instance being launched.

After making the required changes, select **Save**, to save the changes and **Cancel** to abort them.

Deleting custom actions

Custom actions created in default settings can also be deleted. Deleting a custom action in the default settings removes it from the default settings and means the action is no longer added to newly added servers. Deleting the action in the default settings does not remove it from existing source servers that have it. To delete a custom action from existing servers, go to the **Source server details** page, select the **Post-launch settings** tab and delete the action from there. Pre-defined actions cannot be deleted through AWS console. If a pre-defined action is not required, it can be deactivated or deleted via API.

Locate the action you want to delete in the **Actions** card view, or use the search field to filter the actions by name. Select the action, and select **Delete**. To confirm, press **Delete**.

Predefined post-launch actions

AWS Elastic Disaster Recovery allows you to run various predefined post-launch actions on your EC2 launched instance. Use these out-of-the-box actions to validate your launch or improve your launch flexibility.

Choose from a variety of predefined post-launch actions

- Enable SSM
- Install a CloudWatch Agent
- Create AMI from instance
- Configure Time Sync
- Volume integrity validation
- Process status validation
- Validate disk space
- EC2 connectivity check
- HTTP/HTTPS response validation
- Verify tags

Note

These predefined post-launch actions are supported in the Middle East (UAE) Region:

- Enable SSM
- CloudWatch agent installation
- Create AMI from instance
- Volume integrity validation
- Process status validation
- Validate disk space
- EC2 connectivity check
- HTTP/HTTPS response validation

- Verify tags

Enable SSM

The AWS Systems Manager (AWS SSM) allows AWS Elastic Disaster Recovery to run post-launch actions on your recovery instances after they are launched. When you activate the post-launch actions, AWS Elastic Disaster Recovery installs the **AWS SSM agent**.

The AWS SSM agent must be installed for any other post-launch action to run. Therefore, this is the only post-launch action that is activated by default and cannot be deactivated. Learn more about SSM.

Install a CloudWatch Agent

Use the **CloudWatch agent installation** feature to install and configure the CloudWatch Agent and Application Insights. The launched instance requires these policies:

CloudWatchAgentServerPolicy – The permissions required to use AmazonCloudWatchAgent on servers

AmazonSSMManagedInstanceCore – The policy for Amazon EC2 Role to enable AWS Systems Manager service core functionality

To ensure that the launched instance has the right policies, create a role that has the required permissions as per the policies above or have access to a role with those permissions. Go to **Launch settings > EC2 launch template > Modify > Advanced > IAM instance profile**. Use an existing profile or create a new one using the **Create new IAM profile** link.

> **Note**
>
> You must attach both policies to the template for the CloudWatch agent to operate. Without the **CloudWatchAgentServerPolicy**, the action is still marked as successful but the CloudWatch Agent will not be active. Configuring the Application Insights is optional. You can choose to skip the Application Insights agent configuration and only install the CloudWatch agent. To do so, simply provide the required parameterStoreName parameter and leave the other parameters empty.

Learn more about the CloudWatch Agent.

Create AMI from instance

Use the Create AMI from Instance feature to create a new Amazon Machine Image (AMI) from your AWS DRS launched instance.

The action uses these APIs:

- CreateImage
- DescribeImages

To allow the SSM document to run these APIs, you need to have the required permissions or have access to a role with those permissions and then provide the role's ARN as an input parameter to the SSM automation document. Learn more about creating AMI from instance.

Configure Time Sync

Use the **Time Sync** feature to set the time for your Linux instance using ATSS.

Learn more about Amazon Time Sync.

Volume integrity validation

Use the **Volume integrity validation** to ensure that EBS volumes on the launched instance are:

- The same size as the source (rounded up)
- Properly mounted on the Amazon EC2 instance
- Accessible

This feature allows you to conduct the required validations automatically and saves the time of manual validations.

> **Note**
>
> Up to 50 volumes can be checked in a single action.

Process status validation

Use the **Process status validation** feature to ensure that processes are in running state following instance launch. You need to provide a list of processes that you want to verify, and define how long the service should wait before testing begins.

To check a specific process that should run multiple times, include it several times in the list.

Validate disk space

Use the **Disk space validation** feature to obtain visibility into the disk space that you have at your disposal, as well as logs with actionable insights.

Amazon EC2 connectivity checks

Use the **EC2 connectivity checks** feature to conduct network connectivity checks to a predefined list of ports and hosts.

> **Note**
>
> Up to 5 Port:IP couples can be checked in a single action.

Verify HTTP/HTTPS response

Use the **Verify HTTP/HTTPS response** feature to conduct HTTP/HTTPS connectivity checks to a predefined list of URLs. The feature verifies that HTTP/HTTPS requests (for example, https://localhost) receive the correct response.

Verify tags

Use the **Verify Tags** feature to validate that tags which have been defined in the launch template and on the source server are copied to the migrated server.

AWS DRS source servers

You must add your source servers to the AWS Elastic Disaster Recovery console in order to replicate them into AWS. Source servers are added by installing the AWS Replication Agent on each individual server. The following documentation provides installation paths for both Linux and Windows servers. Ensure that your servers are supported by AWS Elastic Disaster Recovery by reviewing the supported Windows operating systems and supported Linux operating systems documentation.

Once your source servers have been added to AWS Elastic Disaster Recovery, you can monitor and interact with them from the **Source Servers** page. The source servers page is the default view in the AWS Elastic Disaster Recovery Console, and is the page that you interact with the most. On the **Source Servers** page you can view all of your source servers, monitor their recovery readiness and data replication state, see the last recovery result, see any pending actions, and sort your servers by column contents. The command menus on the page allow you to perform source server actions such as adding source servers, editing settings, disconnecting, and deleting source servers.

You can choose the hostname of any individual source server on the source servers page in order to access the server details view. This view allows you to see the details for individual servers. Here you are able to see an in-depth overview of the server's recovery state, view the server's technical details, manage tags, manage disks, and most importantly, configure the individual replication settings and launch settings for the server.

Topics

- Adding source servers to AWS DRS
- AWS DRS source servers page
- View server details with AWS DRS

Adding source servers to AWS DRS

Add source servers to AWS Elastic Disaster Recovery by installing the AWS Replication Agent (the Agent) on them. The Agent can be installed on both Linux and Windows servers.

Linux installation instructions

Windows installation instructions

Topics

- Installation requirements for AWS Replication Agent
- Windows operating systems supported by Elastic Disaster Recovery
- AWS DRS supported Linux operating systems
- Installing the AWS Replication Agent
- Adding instances from the Amazon EC2 Console

Installation requirements for AWS Replication Agent

Before installing the AWS Replication Agent on your source servers, ensure that they meet the following requirements:

Topics

- General requirements
- Source server requirements
- Linux installation requirements
- Windows installation requirements

General requirements

- Ensure that the source server operating system is supported by AWS.
  - Supported Windows operating systems.
  - Supported Linux operating systems.
- Ensure that your setup meets all replication networking requirements. Learn more about network requirements.
- Ensure MAC address stability – ensure that the MAC addresses of the source servers do not change upon a reboot or any other common changes in your network environment. AWS Elastic Disaster Recovery calculates the unique ID of the source server from the MAC address. When a MAC address changes, AWS Elastic Disaster Recovery is no longer able to correctly identify the source server. Consequently, replication stops. If this happens, you need to reinstall the AWS Replication Agent and start replication from the beginning.

> **Note**
>
> Elastic Disaster Recovery Agents can only be installed on instances that are in AWS Regions that are supported by Elastic Disaster Recovery. In case of AWS-AWS disaster recovery (in-AWS), Elastic Disaster Recovery should be initialized in both source and target region (done by going through the initialization wizard).

Source server requirements

- Verify that your source server has at least 300 MB of free RAM to run the AWS Replication Agent.
- AWS Elastic Disaster Recovery supports 64-bit operating systems on the `x86_64` architecture. AWS Elastic Disaster Recovery also supports 64-bit Arm (`arm64`) Linux source servers on AWS Graviton-based Amazon EC2 instances that recover to AWS. Elastic Disaster Recovery supports `arm64` only for AWS-to-AWS recovery. For details, see AWS DRS supported Linux operating systems.
- AWS Elastic Disaster Recovery does not support paravirtualized source servers.
- The AWS Replication Agent installer supports multipath.

Linux installation requirements

Ensure that your Linux source server meets the following installation requirements prior to installing the AWS Replication Agent:

- Python 2 (2.4 or above) or Python 3 (3.0 or above) is installed on the server.
- Verify that you meet these disk space requirements:
  - At least 4 GB of free disk space on the root directory (/) of your source server for the installation. To check the available disk space on the root directory, run the `df -h /` command.
  - At least 500 MB of free disk space on the */tmp* directory for the duration of the installation process. To check the available disk space on the /tmp directory run the `df -h /tmp` command.
  - If `/boot` is a separate partition, ensure that it has a minimum of 50 MB free space needed for the installation. To check the available disk space on the /boot directory run the `df -h /boot` command.

- The active bootloader software is GRUB 1 or 2.
- Secure Boot is not supported in Linux.
- Machines that boot off a disk configured with GPT partitioning must have the package 'grub2-pc-modules' installed
- When performing a failback for a Linux server, you must boot the Failback Client with the same boot mode (BIOS or UEFI) as the recovery instance that you are failing back from. The Failback Client supports `x86_64` servers only. For `arm64` source servers, fail back within AWS. See Performing a failback with Elastic Disaster Recovery.
- Ensure that /tmp is mounted as read+write.
- Boot disks that span multiple physical disks are not supported.
- Ensure that /tmp is mounted with the exec option. Verify that the /tmp directory is mounted in a way that allows you to run scripts and applications from it.

  To verify that the /tmp directory is mounted without the noexec option, run the `sudo mount | grep '/tmp'` command.

  If the result is similar to the example, the issue exists in your OS: /dev/xvda1 on /tmp type ext4 (rw,noexec)

  To fix and remove the noexec option from the mounted /tmp directory, run the `sudo mount -o remount,exec /tmp` command.

  Example of the troubleshooting procedure:

```
ubuntu@Linux-1:~$ sudo mount | grep '/tmp'
/dev/xvda1 on /tmp type ext4 (rw,noexec)
ubuntu@Linux-1:~$ sudo mount -o remount,exec /tmp
ubuntu@Linux-1:~$ sudo mount | grep '/tmp'
/dev/xvda1 on /tmp type ext4 (rw)
```

- The AWS Elastic Disaster Recovery user needs to be a user in the sudoers list - a user who can perform sudo.
- Ensure that the dhclient package is installed. The DHCP client is required because AWS Elastic Disaster Recovery configures recovered instances to use DHCP networking. If the package is not installed, use the appropriate command for your distribution:
  - On RHEL/CentOS/Oracle/Amazon Linux: `sudo yum install dhclient` or `sudo yum install dhcp-client`
  - On Debian/Ubuntu: `sudo apt install isc-dhcp-client`
  - On SUSE: `sudo zypper install dhcp-client`

- Verify that you have kernel-devel/linux-headers installed that are exactly the same version as the kernel you are running.

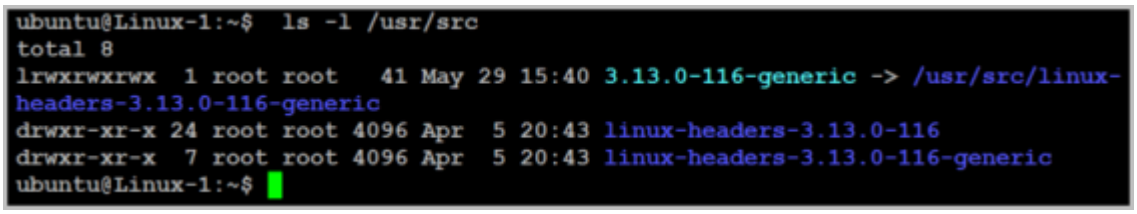
  The version number of the kernel headers should be identical to the version number of the kernel. Follow these steps:

  1. Identify the version of your running kernel by running the `uname -r` command. The 'uname -r' output version should match the version of one of the installed kernel headers packages (kernel-devel-<version number> / linux-headers-<version number>).
  2. Identify the version of your kernel-devel/linux-headers by running:
     - On RHEL/CENTOS/Oracle/SUSE: `rpm -qa | grep kernel`. This command looks for kernel-devel.
     - On Debian/Ubuntu: `apt-cache search linux-headers`.
  3. Verify that the folder that contains the kernel-devel/linux-headers is not a symbolic link. If the content of the kernel-devel/linux-headers, which match the version of the kernel, is actually a symbolic link, you need to remove the link before installing the required package.

     To verify that the folder that contains the kernel-devel/linux-headers is not a symbolic link, run the following command:

     - On RHEL/CENTOS/Oracle: `ls -l /usr/src/kernels`
     - On Debian/Ubuntu/SUSE: `ls -l /usr/src`

     These results show that the linux-headers are not a symbolic link.

  4. If the content of the kernel-devel/linux-headers, which match the version of the kernel, is a symbolic link, you need to delete the link using the `rm /usr/src/<LINK NAME>` command.

     For example: rm /usr/src/linux-headers-4.4.1

  5. Install the correct kernel-devel/linux-headers from the repositories.

     If none of the already installed kernel-devel/linux-headers packages match your running kernel version, you need to install the matching package.

> **Note**
>
> You can have several kernel headers versions simultaneously on your OS, and you can therefore safely install new kernel headers packages in addition to your existing ones, without uninstalling the other versions of the package. A new kernel headers package does not impact the kernel, and does not overwrite older versions of the kernel headers.

You must install a kernel headers package with the exact same version number of the running kernel. To install the correct kernel-devel/linux-headers, run:

- On RHEL/CENTOS/Oracle/SUSE: `sudo yum install kernel-devel-`uname -r``
- On Oracle with Unbreakable Enterprise Kernel: `sudo yum install kernel-uek-devel-`uname -r``
- On Debian/Ubuntu: `sudo apt-get install linux-headers-`uname -r``

6. If no matching package was found on the repositories configured on your server, you can download it manually from the Internet and then install it.

   To download the matching kernel-devel/linux-headers package, navigate to:

   - RHEL, CENTOS, Oracle, and SUSE package directory
   - Debian package directory
   - Ubuntu package directory

Windows installation requirements

- Install all available Windows updates on the server.
- A minimum of 4 GB of free disk space is required to launch a drill or recovery instance.
- When performing a recovery, you must boot the Failback Client with the same boot mode (BIOS or UEFI) as the Windows source server.
- A graceful reboot from the OS menu or Windows CLI of a Windows source server does not trigger a rescan in AWS DRS once the source server is restarted. Hard reboots, disk changes, and crashes trigger a rescan.
- Mount points must be assigned a drive letter to be recognized by Elastic Disaster Recovery. A folder path is not recognized.

Windows operating systems supported by Elastic Disaster Recovery

AWS Elastic Disaster Recovery allows replication of physical, virtual or cloud-based source servers to the AWS Cloud for several versions of Windows.

General Notes

Important

Windows 2003 is no longer supported.

Review the AWS Replication Agent installation requirements.

Note

AWS Elastic Disaster Recovery supports Windows source servers on the `x86_64` architecture only. Elastic Disaster Recovery does not support Windows on `arm64`.

These Windows operating systems are supported:

Operating system	Supported versions	Prerequisites and Limitations
Microsoft Windows Server 2025 64-bit		Requires .NET Framework version 4.5 or above.
Microsoft Windows Server 2022 64-bit		Requires .Net Framework version 4.5 or above to be installed by the end user.
Microsoft Windows Server 2019 64-bit		Requires .Net Framework version 4.5 or above to be installed by the end user.
Microsoft Windows Server 2016 64-bit		Requires .Net Framework version 4.5 or above to be installed by the end user.

Operating system	Supported versions	Prerequisites and Limitations
Microsoft Windows 10 64-bit		Ensure that the auto sleep function in Windows 10 is disabled. Data replication may be interrupted if the feature is activated.
Microsoft Windows Server 2012 **This version has reached end of life. We recommend that you update to a more recent version.**	64-bit and R2 64-bit	• Microsoft Windows Server version 2012 uses a version of the AWS Replication Agent, AwsReplicationWindows2012LegacyInstaller.exe, that is only valid for that version. You can download it from `https://aws-elastic-disaster-recovery-<REGION>.s3.amazonaws.com/latest/windows_legacy/windows_2012_legacy/AwsReplicationWindows2012LegacyInstaller.exe` . Replace `<REGION>` with the AWS Region into which you are replicating. • Requires .Net Framework version 4.5 or above to be installed by the end user.

Operating system	Supported versions	Prerequisites and Limitations
Microsoft Windows Server 2008 **This version has reached end of life. We recommend that you update to a more recent version.**	64-bit and R2 64-bit	• Windows Server 2008 requires .Net Framework version 3.5 to be installed by the end user. Windows Server 2008 **R2** requires .Net Framework version 4.5 or above to be installed by the end user. • Windows 2008 x64 requires SP2 and other Microsoft updates to support the SHA-2 signature of the AWS Replication Agent driver. • The AWS Replication Agent and agent installer requires a separate installer file, `AwsReplicationWindowsLegacyInstaller.exe` for end-of-life versions of Windows because they use older versions of software components that cannot be upgraded. • **Windows Server 2008 and Windows Server 2008 R2 with GPT-partitioned system disks (UEFI boot) are not supported.** These operating systems must use an MBR-partitioned

Operating system	Supported versions	Prerequisites and Limitations
		system disk (BIOS boot) for recovery. • Nitro instances can only be used with Windows Server 2008 R2 and upwards. Earlier versions are not supported. • A shutdown (from the OS menu or Windows CLI) of a Windows source server triggers a rescan in AWS DRS once the source server is restarted.

Operating system	Supported versions	Prerequisites and Limitations
Microsoft Windows 7 64-bit **This version has reached end of life. We recommend that you update to a more recent version.**		• The AWS Replication Agent and agent installer requires a separate installer file, `AwsReplicationWindowsLegacyInstaller.exe` for end-of-life versions of Windows because they use older versions of software components that cannot be upgraded. • Nitro instances can only be used with Windows Server 2008 R2 and upwards. Earlier versions are not supported. • A shutdown (from the OS menu or Windows CLI) of a Windows source server triggers a rescan in AWS DRS once the source server is restarted.

AWS DRS supported Linux operating systems

General Notes

- Review the AWS Replication Agent installation requirements.
- For `arm64` support, see AWS Graviton source servers.
- Linux kernel versions up to 6.14 are supported.
- For source machines configured with LVM, on RHEL/Oracle version less than or equal to 9.4, please make sure to update the lvm package to `lvm2-2.03.23-1.el9` or later.

- AWS Elastic Disaster Recovery does not support 32 bit versions of Linux.
- Hard reboots, disk changes, and crashes trigger a rescan. Graceful reboots do not trigger a rescan in the following versions:
  - RHEL/CentOS/Oracle Linux 6+ (kernel versions 2.6.32–431 and above)
  - SUSE 12+
  - Ubuntu 16+ LTS
  - AL 2 and AL 2023
  - Rocky 8+
  - Debian 9+

AWS Graviton source servers

`arm64` is the 64-bit Arm processor architecture used by AWS Graviton processors. AWS Elastic Disaster Recovery supports `arm64` source servers that run on AWS Graviton-based Amazon EC2 instances and recover to AWS. Elastic Disaster Recovery supports `arm64` only for AWS-to-AWS recovery. Source servers in on-premises environments or other clouds must use `x86_64`.

The AWS Replication Agent installer is architecture-specific. For an `arm64` server, download the `arm64` installer. See Installing the AWS Replication Agent on Linux.

For the launch requirements that apply to `arm64` source servers, see Key considerations for EC2 launch templates.

Support removal notices

The following operating systems are no longer supported, or will no longer be supported after the listed date. AWS does not provide troubleshooting assistance or compatibility fixes for unsupported operating systems.

- **RHEL 5.x and CentOS 5.x** — Support ended December 30, 2025.
- **Debian 6.x–9.x** — Support ended April 30, 2026.
- **Ubuntu 12.04** — Support ends August 20, 2026.
- **CentOS 6.x, SUSE (SLES) 11.x, and Oracle Linux 6.x** — Support ends August 28, 2026.

These Linux operating systems are supported:

Operating system	Supported versions	Prerequisites and Limitations
Amazon Linux	1, 2, 2023	• Amazon Linux 1 is only supported for AWS to AWS recovery. • Amazon Linux 2023 is supported up to kernel version 6.12.
RHEL	6.0 to 10.1	• For RHEL 8.x, a prerequisite is to run `$ sudo yum install elfutils-libelf-devel` • Kernel versions 2.6.32-71 are not supported in RHEL 6.0 • The post-launch actions feature is not supported on RHEL 6.x. • Nitro instance types work with RHEL 7.4+ • AWS requires that servers running Red Hat Enterprise Linux (RHEL) must have Cloud Access (BYOL) licenses in order to be recovered to AWS. Note that servers running RHEL Cloud Access Gold Images allow you to access AWS Red Hat Update Infrastructure (RHUI), Red Hat Satellite, or Red Hat Subscription Manager

Operating system	Supported versions	Prerequisites and Limitations
		(RHSM). If you are using RHEL Cloud Access Gold Images, you will not be able to access RHUI upon failover to AWS unless you link your AWS account to your Red Hat account via the Red Hat portal, and select the Gold image AMI in the launch template. • You must select an AWS provided RHEL AMI in the Launch Template for servers running Red Hat Enterprise Linux (RHEL) Pay as You Go (PAYG) images. This allows access to RHUI after failover.
CentOS	6.0 to 8.0	• Kernel versions 2.6.32-71 are not supported in CentOS 6.0 • The post-launch actions feature is not supported on CentOS 6.x. • Nitro instance types work with CentOS 7.4+

Operating system	Supported versions	Prerequisites and Limitations

Oracle Linux	6.0 to 7.0, 8.5 to 8.10, 9.0 to 9.8, and 10.1 to 10.2	• For Oracle Linux 8.x, a prerequisite is to run `$ sudo yum install elfutils-libelf-devel`
		• Kernel versions 2.6.32-71 are not supported in Oracle Linux 6.0
		• The post-launch actions feature is not supported on Oracle Linux 6.x.
		• Nitro instance types work with Oracle Linux 7.4+
		• Oracle Linux 6.0 to 7.0 source servers must be running either Unbreakable Enterprise Kernel Release 3 or higher or a Red Hat Compatible Kernel.
		• Oracle Linux 8.5 to 8.9 (running either Unbreakable Enterprise Kernel Release 3 or higher or a Red Hat Compatible Kernel) – the following UEK kernels were tested:
		  • 5.15.0-200.131.27.el9uek.x86_64
		  • 5.15.0-101.103.2.1.el9uek.x86_64
		  • 5.15.0-3.60.5.1.el9uek.x86_64

Operating system	Supported versions	Prerequisites and Limitations
		• 5.15.0-0.30.19.el9 uek.x86_64 • 5.15.0-206.153.7.1 .el8uek.x86_64 • 5.15.0-200.131.27. el8uek.x86_64 • 5.15.0-101.103.2.1 .el8uek.x86_64 • 5.15.0-3.60.5.1.el 8uek.x86_64 • 5.4.17-2136.314.6. 3.el8uek.x86_64 • 5.4.17-2136.307.3. 1.el8uek.x86_64 • 5.4.17-2136.300.7. el8uek.x86_64 • 4.18.0-372.32.1.0. 1.el8_6.x86_64 • Oracle Linux 9.0 to 9.8 supports Unbreakable Enterprise Kernel Release 7 or Release 8, or a Red Hat Compatible Kernel. The following kernels were tested on Oracle Linux 9.8: • 6.12.0-205.92.4.2. el9uek.x86_64 (Unbreakable Enterprise Kernel Release 8)

Operating system	Supported versions	Prerequisites and Limitations
		• 5.14.0-687.39.1.el 9_8.x86_64 (Red Hat Compatible Kernel) • Oracle Linux 10.1 to 10.2 source servers must be running either Unbreakable Enterprise Kernel Release 8 or a Red Hat Compatible Kernel.
Rocky Linux	8, 9.7	For Rocky Linux 8.x, a prerequisite is to run `$ sudo yum install elfutils-libelf-devel`
SUSE	11 SP4 to 15 SP5	• The AWS Replication Agent is supported on SUSE Linux Enterprise Server (SLES) 11 SP4 and higher. • For SUSE Linux (SLES) 11 SP4 to work, you must install the Xen drivers and then reboot the servers before installing the AWS Replication Agent. Use this command to install the drivers: `$ sudo zypper install -y xen-kmp-default .`

Operating system	Supported versions	Prerequisites and Limitations
Ubuntu	12.04 to 24.04	• Only Kernel 3.x or above are supported • Azure kernels are not supported as they are not compatible with the Amazon EC2 hardware. Ubuntu servers from Azure are required to switch the kernel to a standard kernel or the AWS tuned Ubuntu kernel 'linux-aws'.
Debian	10 to 13	Only Kernel 3.x or above are supported

Installing the AWS Replication Agent

You must install the AWS Replication Agent on each source server that you want to add to AWS Elastic Disaster Recovery. Agent installation is composed of the following steps:

Topics

- Generating the required AWS credentials
- Using an instance profile for agent installation in AWS
- Installation instructions for the AWS Replication Agent
- Installing the agent on a secured network
- Uninstalling the agent
- Reinstalling the agent
- Supporting marketplace licenses

Generating the required AWS credentials

In order to install the AWS Replication Agent, you must first generate the required AWS credentials. You can create temporary credentials with AWS STS.

> **Important**
>
> Temporary credentials have many advantages. You do not need to rotate them or revoke them when they are no longer needed, and they cannot be reused after they expire. You can specify for how long the credentials are valid, up to a maximum limit. Because they provide enhanced security, using temporary credentials is considered best practice and the recommended option. For more information, see IAM security best practices.

Temporary credentials

Before you install the AWS Replication Agent, you need to generate temporary AWS security credentials. The temporary credentials provided by AWS Elastic Disaster Recovery utilize a similar mechanism to the one used by IAM Roles Anywhere.

To create temporary credentials, take the following steps:

1. Create a new IAM Role with the **AWSElasticDisasterRecoveryAgentInstallationPolicy** policy.
2. Request temporary security credentials via AWS STS using the AssumeRole API.

Learn more about how temporary credentials work.

> **Note**
>
> You can also create the default IAM role with the required permissions as an instance profile, as described in Instance profile role installation.

Using an instance profile for agent installation in AWS

When installing the AWS Replication Agent on an Amazon EC2 instance (when the source and recovery servers are both in AWS Regions), you don't need to generate credentials. Instead, you can use an instance profile with the required IAM policy. For the actual agent installation steps, see Installing on Linux or Installing on Windows.

- Go to the EC2 console and select your EC2 instance.
- From the top right-hand menu, select **Actions > Security > Modify IAM role**.
- Use a role that contains the AWSElasticDisasterRecoveryEc2InstancePolicy policy.

  If none exists, click **Create new IAM role**, attach the policy and return to the EC2 console window.
- Select your new role from the drop-down list and click **Update**.

Installation instructions for the AWS Replication Agent

Once you have generated the required AWS credentials, you can install the AWS Replication Agent on your source servers. There are separate installation instructions for Linux and for Windows. Each operating system has its own installer.

Topics

- Installing the AWS Replication Agent on Linux
- Installing the AWS Replication Agent on Windows
- AWS Replication Agent Installer parameters

Installing the AWS Replication Agent on Linux

To install the agent on a Linux source server, you should ensure that your source meets all the requirements listed in the supported Linux operating systems documentation.

Before installing, please ensure that you are aware of the following:

- You need root privileges to run the Agent installer file on a Linux server. Alternatively, you can run the Agent Installer file with sudo permissions.
- The Linux installer creates the **"aws-replication"** group and **"aws-replication"** user within that group. The Agent runs within the context of the newly created user. Agent installation attempts to add the user to **"sudoers"**. Installation fails if the Agent is unable to add the newly created **"aws-replication"** user to **"sudoers"**.

1. Download the agent installer `aws-replication-installer-init` onto your Linux source server.

The installer is architecture-specific. Download the installer that matches the source server architecture:

- x86_64: `https://aws-elastic-disaster-recovery-<REGION>.s3.<REGION>.amazonaws.com/latest/linux/aws-replication-installer-init`
- arm64: `https://aws-elastic-disaster-recovery-<REGION>.s3.<REGION>.amazonaws.com/latest/linux_arm64/aws-replication-installer-init`

> **Note**
>
> Replace <REGION> with the AWS Region into which you are replicating.
> Run `uname -m` on the source server to find its architecture. `aarch64` means `arm64`.
> The `x86_64` installer does not run on `arm64` servers. Elastic Disaster Recovery supports `arm64` only for AWS Graviton-based Amazon EC2 source instances that recover to AWS.

The following examples download the installer file from the us-east-1 region.

wget

```
# x86_64
wget -O ./aws-replication-installer-init https://aws-elastic-disaster-recovery-us-east-1.s3.us-east-1.amazonaws.com/latest/linux/aws-replication-installer-init
# arm64
wget -O ./aws-replication-installer-init https://aws-elastic-disaster-recovery-us-east-1.s3.us-east-1.amazonaws.com/latest/linux_arm64/aws-replication-installer-init
```

curl

```
# x86_64
curl -o aws-replication-installer-init https://aws-elastic-disaster-recovery-us-east-1.s3.us-east-1.amazonaws.com/latest/linux/aws-replication-installer-init
# arm64
curl -o aws-replication-installer-init https://aws-elastic-disaster-recovery-us-east-1.s3.us-east-1.amazonaws.com/latest/linux_arm64/aws-replication-installer-init
```

Note

If you are using a legacy Linux OS that does not support TLS 1.2, you need to download the installer on a different server with an OS that supports TLS 1.2 and copy it to the legacy servers you intend to install the agent on.

The command line indicates when the installer has been successfully downloaded.

Important

If you need to validate the installer hash, use the hash file that matches the installer architecture:

- x86_64: `https://aws-elastic-disaster-recovery-hashes-<REGION>.s3.<REGION>.amazonaws.com/latest/linux/aws-replication-installer-init.sha512`
- arm64: `https://aws-elastic-disaster-recovery-hashes-<REGION>.s3.<REGION>.amazonaws.com/latest/linux_arm64/aws-replication-installer-init.sha512`

Replace `<REGION>` with the AWS Region into which you are replicating.
For example, when using the **us-east-1** Region:

- x86_64: `https://aws-elastic-disaster-recovery-hashes-us-east-1.s3.us-east-1.amazonaws.com/latest/linux/aws-replication-installer-init.sha512`
- arm64: `https://aws-elastic-disaster-recovery-hashes-us-east-1.s3.us-east-1.amazonaws.com/latest/linux_arm64/aws-replication-installer-init.sha512`

Note

AWS Regions that are not opt-in also support the shorter installer path: `https://aws-elastic-disaster-recovery-<REGION>.s3.amazonaws.com/latest/linux/aws-replication-installer-init`. Replace `<REGION>` with the

AWS Region into which you are replicating. For an `arm64` server, use `linux_arm64` instead of `linux`.

2. Use this command on your source server in order to run the installation script.

```
chmod +x aws-replication-installer-init; sudo ./aws-replication-installer-init
```

> **Note**
>
> To install the agent on a secured network, learn about the additional required configurations.

If you require additional customization, you can add a variety of parameters to the installer script in order to manipulate the way the Agent is installed on your server. See the AWS Replication Agent Installer Parameters for more information.

The installer confirms that the installation of the AWS Replication Agent has started.

```
$ chmod +x aws-replication-installer-init; sudo ./aws-replication-installer-init
The installation of the AWS Replication Agent has started.
```

3. The installer prompts you to enter your **AWS Region Name**, the **AWS Access Key ID** and **AWS Secret Access Key** that you previously generated. Enter the complete AWS Region name (for example, eu-central-1), the full AWS Access Key ID and the full AWS Secret Access Key. If you are using temporary credentials, you also need to specify the session token.

```
$ chmod +x aws-replication-installer-init; sudo ./aws-replication-installer-init
The installation of the AWS Replication Agent has started.
AWS Region name: us-east-1
AWS Access Key ID: AKIAI0SF0DNN71EXAMPLE
AWS Secret Access Key: wJalrXUtnFEMI/K71MDENG/bPxRfiCYEXAMPLEKEY
```

> **Note**
>
> You can also enter these values as part of the installation script command parameters. If you do not enter these parameters as part of the installation script, you are prompted to enter them one by one as described above. (for example: `chmod +x aws-`

```
replication-installer-init; sudo ./aws-replication-installer-init
--region regionname --aws-access-key-id AKIAIOSFODNN71EXAMPLE --
aws-secret-access-key wJalrXUtnFEMI/K71MDENG/bPxRfiCYEXAMPLEKEY)
```

Note

You can also pass credentials through environment variables. We recommend using temporary credentials from AWS STS. Set `AWS_ACCESS_KEY_ID`, `AWS_SECRET_ACCESS_KEY`, and `AWS_SESSION_TOKEN`. Then run the installer with `sudo -E` and `--no-prompt`:

```
export AWS_ACCESS_KEY_ID=AKIAIOSFODNN7EXAMPLE
export AWS_SECRET_ACCESS_KEY=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
export AWS_SESSION_TOKEN=AQoDYXdzEJr/////////wEa8AMDSomethingEXAMPLE
chmod +x aws-replication-installer-init
sudo -E ./aws-replication-installer-init --region us-east-1 --no-prompt
```

4. Once you have entered your credentials, the installer identifies volumes for replication. The installer displays the identified disks and prompts you to choose the disks you want to replicate.

```
$ chmod +x aws-replication-installer-init; sudo ./aws-replication-installer-init
...
AWS Secret Access Key: wJalrXUtnFEMI/K71MDENG/bPxRfiCYEXAMPLEKEY
Identifying volumes for replication.
Choose the disks you want to replicate. Your disks are: /dev/sda,/dev/xvda
To replicate some of the disks, type the path of the disks, separated with a comma
 (for example, /dev/sda,/dev/sdb).
To replicate all disks, press Enter:
```

To replicate some of the disks, type the path of the disks, separated by a comma, as illustrated in the installer (such as: /dev/sda, /dev/sdb, etc). To replicate all of the disks, press Enter. The installer identifies the selected disks and prints their size.

```
$ chmod +x aws-replication-installer-init; sudo ./aws-replication-installer-init
...
To replicate some of the disks, type the path of the disks, separated with a comma
 (for example, /dev/sda,/dev/sdb).
To replicate all disks, press Enter:
```

```
Identified volume for replication: /dev/xvda of size 8 GiB
```

The installer confirms that all disks were successfully identified.

```
$ chmod +x aws-replication-installer-init; sudo ./aws-replication-installer-init
...
Identified volume for replication: /dev/xvda of size 8 GiB
All volumes for replication were successfully identified.
```

Note

When identifying specific disks for replication, do not use apostrophes, brackets, or disk paths that do not exist. Type only existing disk paths. Each disk you selected for replication is displayed with the caption **Disk to replicate identified**. However, the displayed list of identified disks for replication may differ from the data you entered. This difference can be due to several reasons:

- The root disk of the source server is always replicated, whether you select it or not. Therefore, it always appears on the list of identified disks for replication.
- AWS Elastic Disaster Recovery replicates whole disks. Therefore, if you choose to replicate a partition, its entire disk appears on the list and is later replicated. If several partitions on the same disk are selected, then the disk encompassing all of them appears only once on the list.
- Incorrect disks may be chosen by accident. Ensure that the correct disks have been chosen.

Important

If disks are disconnected from a server, AWS Elastic Disaster Recovery can no longer replicate them, so they are removed from the list of replicated disks. When they are reconnected, the AWS Replication Agent cannot know that these were the same disks that were disconnected and therefore does not add them automatically. To add the disks after they are reconnected, rerun the AWS Replication Agent installer on the server. Note that the returned disks need to be replicated from the beginning. Any disk size change is automatically identified, but also causes a resync. Perform a test after installing the Agent to ensure that the correct disks have been added.

5. After all of the disks to be replicated have been successfully identified, the installer downloads and installs the AWS Replication Agent on the source server.

```
$ chmod +x aws-replication-installer-init; sudo ./aws-replication-installer-init
...
Identified volume for replication: /dev/xvda of size 8 GiB
All volumes for replication were successfully identified.
Downloading the AWS Replication Agent onto the source server... Finished
Installing the AWS Replication Agent onto the source server... Finished
```

6. Once the AWS Replication Agent is installed, the server is added to the AWS Elastic Disaster Recovery console and undergoes the initial sync process. The installer provides the source server's ID.

```
$ chmod +x aws-replication-installer-init; sudo ./aws-replication-installer-init
...
Installing the AWS Replication Agent onto the source server... Finished
Syncing the source server with the AWS Elastic Disaster Recovery console... Finished
The following is the source server ID: s-3146f90b19example
The AWS Replication Agent was successfully installed.
$
```

   You can review this process in real time on the **Source servers** page.

Installing the AWS Replication Agent on Windows

To install the AWS Replication Agent on a Windows source server, you should ensure that your source meets all the requirements listed in the supported Windows operating systems documentation.

Prior to installing the AWS Replication Agent, please ensure that you are aware of the following:

- You need to run the agent installer file as an Administrator on each Windows server.
- We recommend using Windows PowerShell, which supports the 'Ctrl+V' shortcut for pasting. Windows Command Prompt (cmd) does not support this functionality.

Downloading the installer

Before installing the AWS Replication Agent, `AwsReplicationWindowsInstaller.exe`, it needs to be downloaded. Copy or distribute the downloaded agent installer to each Windows source server that you want to add to AWS Elastic Disaster Recovery.

The agent installer follows the following format:

`https://aws-elastic-disaster-recovery-<REGION>.s3.<REGION>.amazonaws.com/latest/windows/AwsReplicationWindowsInstaller.exe`

> **Note**
>
> Replace <REGION> with the AWS Region into which you are replicating.

The following is an example command for downloading the installer file from the us-east-1 region:

> **Note**
>
> If you are using Windows Server 2016 or older, you may need to enable TLS 1.2 in PowerShell before downloading: `[System.Net.ServicePointManager]::SecurityProtocol = 'TLS12'`

```
Invoke-WebRequest "https://aws-elastic-disaster-recovery-us-east-1.s3.us-east-1.amazonaws.com/latest/windows/AwsReplicationWindowsInstaller.exe" -OutFile .\AwsReplicationWindowsInstaller.exe
```

The command line indicates when the installer has been successfully downloaded.

> **Note**
>
> - AWS Regions that are not opt-in also support the shorter installer path: `https://aws-elastic-disaster-recovery-<REGION>.s3.amazonaws.com/latest/windows/AwsReplicationWindowsInstaller.exe` . Replace <REGION> with the AWS Region into which you are replicating.
> - Microsoft Windows Server versions 2008 and 2008 R2 use a version of the AWS Replication Agent that is only valid for those versions -

AwsReplicationWindowsLegacyInstaller.exe. DO NOT use this installer file to install the agent on any other OS types. You can download it from https://aws-elastic-disaster-recovery-<REGION>.s3.amazonaws.com/latest/windows_legacy/AwsReplicationWindowsLegacyInstaller.exe . Replace <REGION> with the AWS Region into which you are replicating.

- Microsoft Windows Server 2012 uses a version of the AWS Replication Agent that is only valid for that version AwsReplicationWindows2012LegacyInstaller.exe. DO NOT use this installer file to install the agent on any other OS types. You can download it from https://aws-elastic-disaster-recovery-<REGION>.s3.amazonaws.com/latest/windows_legacy/windows_2012_legacy/AwsReplicationWindows2012LegacyInstaller.exe . Replace <REGION> with the AWS Region into which you are replicating.

  If you need to validate the installer hash, the correct hash is here: https://aws-elastic-disaster-recovery-hashes-<REGION>.s3.<REGION>.amazonaws.com/latest/windows_legacy/windows_2012_legacy/AwsReplicationWindows2012LegacyInstaller.exe.sha512 (replace <REGION> with the AWS Region into which you are replicating.)

AWS Replication Agent download URL for Windows for each supported AWS Region

Region name	Region identity	Download Link
Africa (Cape Town)	af-south-1	https://aws-elastic-disaster-recovery-af-south-1.s3.af-south-1.amazonaws.com/latest/windows/AwsReplicationWindowsInstaller.exe
Asia Pacific (Hong Kong)	ap-east-1	https://aws-elastic-disaster-recovery-ap-east-1.s3.ap-east-1.amazonaws.com/latest/windows/AwsReplicationWindowsInstaller.exe

Region name	Region identity	Download Link
Asia Pacific (Taipei)	ap-east-2	https://aws-elastic-disaster-recovery-ap-east-2-f1dcde1a.s3.ap-east-2.amazonaws.com/latest/windows/AwsReplicationWindowsInstaller.exe
Asia Pacific (Tokyo)	ap-northeast-1	https://aws-elastic-disaster-recovery-ap-northeast-1.s3.ap-northeast-1.amazonaws.com/latest/windows/AwsReplicationWindowsInstaller.exe
Asia Pacific (Seoul)	ap-northeast-2	https://aws-elastic-disaster-recovery-ap-northeast-2.s3.ap-northeast-2.amazonaws.com/latest/windows/AwsReplicationWindowsInstaller.exe
Asia Pacific (Osaka)	ap-northeast-3	https://aws-elastic-disaster-recovery-ap-northeast-3.s3.ap-northeast-3.amazonaws.com/latest/windows/AwsReplicationWindowsInstaller.exe
Asia Pacific (Mumbai)	ap-south-1	https://aws-elastic-disaster-recovery-ap-south-1.s3.ap-south-1.amazonaws.com/latest/windows/AwsReplicationWindowsInstaller.exe

Region name	Region identity	Download Link
Asia Pacific (Hyderabad)	ap-south-2	https://aws-elastic-disaster-recovery-ap-south-2.s3.ap-south-2.amazonaws.com/latest/windows/AwsReplicationWindowsInstaller.exe
Asia Pacific (Singapore)	ap-southeast-1	https://aws-elastic-disaster-recovery-ap-southeast-1.s3.ap-southeast-1.amazonaws.com/latest/windows/AwsReplicationWindowsInstaller.exe
Asia Pacific (Sydney)	ap-southeast-2	https://aws-elastic-disaster-recovery-ap-southeast-2.s3.ap-southeast-2.amazonaws.com/latest/windows/AwsReplicationWindowsInstaller.exe
Asia Pacific (Jakarta)	ap-southeast-3	https://aws-elastic-disaster-recovery-ap-southeast-3.s3.ap-southeast-3.amazonaws.com/latest/windows/AwsReplicationWindowsInstaller.exe
Asia Pacific (Melbourne)	ap-southeast-4	https://aws-elastic-disaster-recovery-ap-southeast-4.s3.ap-southeast-4.amazonaws.com/latest/windows/AwsReplicationWindowsInstaller.exe

Region name	Region identity	Download Link
Asia Pacific (Malaysia)	ap-southeast-5	https://aws-elastic-disaster-recovery-ap-southeast-5-ebaf53cb.s3.ap-southeast-5.amazonaws.com/latest/windows/AwsReplicationWindowsInstaller.exe
Asia Pacific (New Zealand)	ap-southeast-6	https://aws-elastic-disaster-recovery-ap-southeast-6-8a759f92.s3.ap-southeast-6.amazonaws.com/latest/windows/AwsReplicationWindowsInstaller.exe
Asia Pacific (Thailand)	ap-southeast-7	https://aws-elastic-disaster-recovery-ap-southeast-7-69ef66ac.s3.ap-southeast-7.amazonaws.com/latest/windows/AwsReplicationWindowsInstaller.exe
Canada (Central)	ca-central-1	https://aws-elastic-disaster-recovery-ca-central-1.s3.ca-central-1.amazonaws.com/latest/windows/AwsReplicationWindowsInstaller.exe
Canada West (Calgary)	ca-west-1	https://aws-elastic-disaster-recovery-ca-west-1-2590fa22.s3.ca-west-1.amazonaws.com/latest/windows/AwsReplicationWindowsInstaller.exe

Region name	Region identity	Download Link
Europe (Frankfurt)	eu-central-1	https://aws-elastic-disaster-recovery-eu-central-1.s3.eu-central-1.amazonaws.com/latest/windows/AwsReplicationWindowsInstaller.exe
Europe (Zurich)	eu-central-2	https://aws-elastic-disaster-recovery-eu-central-2.s3.eu-central-2.amazonaws.com/latest/windows/AwsReplicationWindowsInstaller.exe
Europe (Stockholm)	eu-north-1	https://aws-elastic-disaster-recovery-eu-north-1.s3.eu-north-1.amazonaws.com/latest/windows/AwsReplicationWindowsInstaller.exe
Europe (Milan)	eu-south-1	https://aws-elastic-disaster-recovery-eu-south-1.s3.eu-south-1.amazonaws.com/latest/windows/AwsReplicationWindowsInstaller.exe
Europe (Spain)	eu-south-2	https://aws-elastic-disaster-recovery-eu-south-2.s3.eu-south-2.amazonaws.com/latest/windows/AwsReplicationWindowsInstaller.exe
Europe (Ireland)	eu-west-1	https://aws-elastic-disaster-recovery-eu-west-1.s3.eu-west-1.amazonaws.com/latest/windows/AwsReplicationWindowsInstaller.exe

Region name	Region identity	Download Link
Europe (London)	eu-west-2	https://aws-elastic-disaster-recovery-eu-west-2.s3.eu-west-2.amazonaws.com/latest/windows/AwsReplicationWindowsInstaller.exe
Europe (Paris)	eu-west-3	https://aws-elastic-disaster-recovery-eu-west-3.s3.eu-west-3.amazonaws.com/latest/windows/AwsReplicationWindowsInstaller.exe
Israel (Tel Aviv)	il-central-1	https://aws-elastic-disaster-recovery-il-central-1.s3.il-central-1.amazonaws.com/latest/windows/AwsReplicationWindowsInstaller.exe
Middle East (UAE)	me-central-1	https://aws-elastic-disaster-recovery-me-central-1.s3.me-central-1.amazonaws.com/latest/windows/AwsReplicationWindowsInstaller.exe
Middle East (Bahrain)	me-south-1	https://aws-elastic-disaster-recovery-me-south-1.s3.me-south-1.amazonaws.com/latest/windows/AwsReplicationWindowsInstaller.exe
Mexico (Central)	mx-central-1	https://aws-elastic-disaster-recovery-mx-central-1-1f310737.s3.mx-central-1.amazonaws.com/latest/windows/AwsReplicationWindowsInstaller.exe

Region name	Region identity	Download Link
South America (São Paulo)	sa-east-1	https://aws-elastic-disaster-recovery-sa-east-1.s3.sa-east-1.amazonaws.com/latest/windows/AwsReplicationWindowsInstaller.exe
US East (N. Virginia)	us-east-1	https://aws-elastic-disaster-recovery-us-east-1.s3.us-east-1.amazonaws.com/latest/windows/AwsReplicationWindowsInstaller.exe
US East (Ohio)	us-east-2	https://aws-elastic-disaster-recovery-us-east-2.s3.us-east-2.amazonaws.com/latest/windows/AwsReplicationWindowsInstaller.exe
US West (N. California)	us-west-1	https://aws-elastic-disaster-recovery-us-west-1.s3.us-west-1.amazonaws.com/latest/windows/AwsReplicationWindowsInstaller.exe
US West (Oregon)	us-west-2	https://aws-elastic-disaster-recovery-us-west-2.s3.us-west-2.amazonaws.com/latest/windows/AwsReplicationWindowsInstaller.exe

Validating the downloaded AWS Replication Agent installer for Windows.

Important

If you need to validate the installer hash, the correct hash is here:

`https://aws-elastic-disaster-recovery-hashes-<REGION>.s3.<REGION>.amazonaws.com/latest/windows/AwsReplicationWindowsInstaller.exe.sha512`

Replace <REGION> with the AWS Region into which you are replicating, for example: us-east-1:

`https://aws-elastic-disaster-recovery-hashes-us-east-1.s3.us-east-1.amazonaws.com/latest/windows/AwsReplicationWindowsInstaller.exe.sha512`

Region name	Region identity	SHA512 Hash Download Link
Africa (Cape Town)	af-south-1	https://aws-elastic-disaster-recovery-hashes-af-south-1.s3.af-south-1.amazonaws.com/latest/windows/AwsReplicationWindowsInstaller.exe.sha512
Asia Pacific (Hong Kong)	ap-east-1	https://aws-elastic-disaster-recovery-hashes-ap-east-1.s3.ap-east-1.amazonaws.com/latest/windows/AwsReplicationWindowsInstaller.exe.sha512
Asia Pacific (Taipei)	ap-east-2	https://aws-elastic-disaster-recovery-hashes-ap-east-2-f1dcde1a.s3.ap-east-2.amazonaws.com/latest/windows/AwsReplicationWindowsInstaller.exe.sha512
Asia Pacific (Tokyo)	ap-northeast-1	https://aws-elastic-disaster-recovery-hashes-ap-northeast-1.s3.ap-northeast-1.amazonaws.com/latest/windows/

Region name	Region identity	SHA512 Hash Download Link
		AwsReplicationWindowsInstaller.exe.sha512
Asia Pacific (Seoul)	ap-northeast-2	https://aws-elastic-disaster-recovery-hashes-ap-northeast-2.s3.ap-northeast-2.amazonaws.com/latest/windows/AwsReplicationWindowsInstaller.exe.sha512
Asia Pacific (Osaka)	ap-northeast-3	https://aws-elastic-disaster-recovery-hashes-ap-northeast-3.s3.ap-northeast-3.amazonaws.com/latest/windows/AwsReplicationWindowsInstaller.exe.sha512
Asia Pacific (Mumbai)	ap-south-1	https://aws-elastic-disaster-recovery-hashes-ap-south-1.s3.ap-south-1.amazonaws.com/latest/windows/AwsReplicationWindowsInstaller.exe.sha512
Asia Pacific (Hyderabad)	ap-south-2	https://aws-elastic-disaster-recovery-hashes-ap-south-2.s3.ap-south-2.amazonaws.com/latest/windows/AwsReplicationWindowsInstaller.exe.sha512

Region name	Region identity	SHA512 Hash Download Link
Asia Pacific (Singapore)	ap-southeast-1	https://aws-elastic-disaster-recovery-hashes-ap-southeast-1.s3.ap-southeast-1.amazonaws.com/latest/windows/AwsReplicationWindowsInstaller.exe.sha512
Asia Pacific (Sydney)	ap-southeast-2	https://aws-elastic-disaster-recovery-hashes-ap-southeast-2.s3.ap-southeast-2.amazonaws.com/latest/windows/AwsReplicationWindowsInstaller.exe.sha512
Asia Pacific (Jakarta)	ap-southeast-3	https://aws-elastic-disaster-recovery-hashes-ap-southeast-3.s3.ap-southeast-3.amazonaws.com/latest/windows/AwsReplicationWindowsInstaller.exe.sha512
Asia Pacific (Melbourne)	ap-southeast-4	https://aws-elastic-disaster-recovery-hashes-ap-southeast-4.s3.ap-southeast-4.amazonaws.com/latest/windows/AwsReplicationWindowsInstaller.exe.sha512
Asia Pacific (Malaysia)	ap-southeast-5	https://aws-elastic-disaster-recovery-hashes-ap-southeast-5-ebaf53cb.s3.ap-southeast-5.amazonaws.com/latest/windows/AwsReplicationWindowsInstaller.exe.sha512

Region name	Region identity	SHA512 Hash Download Link
Asia Pacific (New Zealand)	ap-southeast-6	https://aws-elastic-disaster-recovery-hashes-ap-southeast-6-8a759f92.s3.ap-southeast-6.amazonaws.com/latest/windows/AwsReplicationWindowsInstaller.exe.sha512
Asia Pacific (Thailand)	ap-southeast-7	https://aws-elastic-disaster-recovery-hashes-ap-southeast-7-69ef66ac.s3.ap-southeast-7.amazonaws.com/latest/windows/AwsReplicationWindowsInstaller.exe.sha512
Canada (Central)	ca-central-1	https://aws-elastic-disaster-recovery-hashes-ca-central-1.s3.ca-central-1.amazonaws.com/latest/windows/AwsReplicationWindowsInstaller.exe.sha512
Canada West (Calgary)	ca-west-1	https://aws-elastic-disaster-recovery-hashes-ca-west-1-2590fa22.s3.ca-west-1.amazonaws.com/latest/windows/AwsReplicationWindowsInstaller.exe.sha512

Region name	Region identity	SHA512 Hash Download Link
Europe (Frankfurt)	eu-central-1	https://aws-elastic-disaster-recovery-hashes-eu-central-1.s3.eu-central-1.amazonaws.com/latest/windows/AwsReplicationWindowsInstaller.exe.sha512
Europe (Zurich)	eu-central-2	https://aws-elastic-disaster-recovery-hashes-eu-central-2.s3.eu-central-2.amazonaws.com/latest/windows/AwsReplicationWindowsInstaller.exe.sha512
Europe (Stockholm)	eu-north-1	https://aws-elastic-disaster-recovery-hashes-eu-north-1.s3.eu-north-1.amazonaws.com/latest/windows/AwsReplicationWindowsInstaller.exe.sha512
Europe (Milan)	eu-south-1	https://aws-elastic-disaster-recovery-hashes-eu-south-1.s3.eu-south-1.amazonaws.com/latest/windows/AwsReplicationWindowsInstaller.exe.sha512
Europe (Spain)	eu-south-2	https://aws-elastic-disaster-recovery-hashes-eu-south-2.s3.eu-south-2.amazonaws.com/latest/windows/AwsReplicationWindowsInstaller.exe.sha512

Region name	Region identity	SHA512 Hash Download Link
Europe (Ireland)	eu-west-1	https://aws-elastic-disaster-recovery-hashes-eu-west-1.s3.eu-west-1.amazonaws.com/latest/windows/AwsReplicationWindowsInstaller.exe.sha512
Europe (London)	eu-west-2	https://aws-elastic-disaster-recovery-hashes-eu-west-2.s3.eu-west-2.amazonaws.com/latest/windows/AwsReplicationWindowsInstaller.exe.sha512
Europe (Paris)	eu-west-3	https://aws-elastic-disaster-recovery-hashes-eu-west-3.s3.eu-west-3.amazonaws.com/latest/windows/AwsReplicationWindowsInstaller.exe.sha512
Middle East (UAE)	me-central-1	https://aws-elastic-disaster-recovery-hashes-me-central-1.s3.me-central-1.amazonaws.com/latest/windows/AwsReplicationWindowsInstaller.exe.sha512
Middle East (Bahrain)	me-south-1	https://aws-elastic-disaster-recovery-hashes-me-south-1.s3.me-south-1.amazonaws.com/latest/windows/AwsReplicationWindowsInstaller.exe.sha512

Region name	Region identity	SHA512 Hash Download Link
Mexico (Central)	mx-central-1	https://aws-elastic-disaster-recovery-hashes-mx-central-1-1f310737.s3.mx-central-1.amazonaws.com/latest/windows/AwsReplicationWindowsInstaller.exe.sha512
South America (São Paulo)	sa-east-1	https://aws-elastic-disaster-recovery-hashes-sa-east-1.s3.sa-east-1.amazonaws.com/latest/windows/AwsReplicationWindowsInstaller.exe.sha512
US East (N. Virginia)	us-east-1	https://aws-elastic-disaster-recovery-hashes-us-east-1.s3.us-east-1.amazonaws.com/latest/windows/AwsReplicationWindowsInstaller.exe.sha512
US East (Ohio)	us-east-2	https://aws-elastic-disaster-recovery-hashes-us-east-2.s3.us-east-2.amazonaws.com/latest/windows/AwsReplicationWindowsInstaller.exe.sha512
US West (N. California)	us-west-1	https://aws-elastic-disaster-recovery-hashes-us-west-1.s3.us-west-1.amazonaws.com/latest/windows/AwsReplicationWindowsInstaller.exe.sha512

Region name	Region identity	SHA512 Hash Download Link
US West (Oregon)	us-west-2	https://aws-elastic-disaster-recovery-hashes-us-west-2.s3.us-west-2.amazonaws.com/latest/windows/AwsReplicationWindowsInstaller.exe.sha512

AWS Replication Agent download URL for Windows versions 2008 and 2008 R2 for each supported AWS Region

Region name	Region identity	Download Link
Africa (Cape Town)	af-south-1	https://aws-elastic-disaster-recovery-af-south-1.s3.af-south-1.amazonaws.com/latest/windows_legacy/AwsReplicationWindowsLegacyInstaller.exe
Asia Pacific (Hong Kong)	ap-east-1	https://aws-elastic-disaster-recovery-ap-east-1.s3.ap-east-1.amazonaws.com/latest/windows_legacy/AwsReplicationWindowsLegacyInstaller.exe
Asia Pacific (Taipei)	ap-east-2	https://aws-elastic-disaster-recovery-ap-east-2-f1dcde1a.s3.ap-east-2.amazonaws.com/latest/windows_legacy/AwsReplicationWindowsLegacyInstaller.exe
Asia Pacific (Tokyo)	ap-northeast-1	https://aws-elastic-disaster-recovery-ap-northeast-1.s

Region name	Region identity	Download Link
		3.ap-northeast-1.amazonaws.com/latest/windows_legacy/AwsReplicationWindowsLegacyInstaller.exe
Asia Pacific (Seoul)	ap-northeast-2	https://aws-elastic-disaster-recovery-ap-northeast-2.s3.ap-northeast-2.amazonaws.com/latest/windows_legacy/AwsReplicationWindowsLegacyInstaller.exe
Asia Pacific (Osaka)	ap-northeast-3	https://aws-elastic-disaster-recovery-ap-northeast-3.s3.ap-northeast-3.amazonaws.com/latest/windows_legacy/AwsReplicationWindowsLegacyInstaller.exe
Asia Pacific (Mumbai)	ap-south-1	https://aws-elastic-disaster-recovery-ap-south-1.s3.ap-south-1.amazonaws.com/latest/windows_legacy/AwsReplicationWindowsLegacyInstaller.exe
Asia Pacific (Hyderabad)	ap-south-2	https://aws-elastic-disaster-recovery-ap-south-2.s3.ap-south-2.amazonaws.com/latest/windows_legacy/AwsReplicationWindowsLegacyInstaller.exe

Region name	Region identity	Download Link
Asia Pacific (Singapore)	ap-southeast-1	https://aws-elastic-disaster-recovery-ap-southeast-1.s3.ap-southeast-1.amazonaws.com/latest/windows_legacy/AwsReplicationWindowsLegacyInstaller.exe
Asia Pacific (Sydney)	ap-southeast-2	https://aws-elastic-disaster-recovery-ap-southeast-2.s3.ap-southeast-2.amazonaws.com/latest/windows_legacy/AwsReplicationWindowsLegacyInstaller.exe
Asia Pacific (Jakarta)	ap-southeast-3	https://aws-elastic-disaster-recovery-ap-southeast-3.s3.ap-southeast-3.amazonaws.com/latest/windows_legacy/AwsReplicationWindowsLegacyInstaller.exe
Asia Pacific (Melbourne)	ap-southeast-4	https://aws-elastic-disaster-recovery-ap-southeast-4.s3.ap-southeast-4.amazonaws.com/latest/windows_legacy/AwsReplicationWindowsLegacyInstaller.exe
Asia Pacific (Malaysia)	ap-southeast-5	https://aws-elastic-disaster-recovery-ap-southeast-5-ebaf53cb.s3.ap-southeast-5.amazonaws.com/latest/windows_legacy/AwsReplicationWindowsLegacyInstaller.exe

Region name	Region identity	Download Link
Asia Pacific (New Zealand)	ap-southeast-6	https://aws-elastic-disaster-recovery-ap-southeast-6-8a759f92.s3.ap-southeast-6.amazonaws.com/latest/windows_legacy/AwsReplicationWindowsLegacyInstaller.exe
Asia Pacific (Thailand)	ap-southeast-7	https://aws-elastic-disaster-recovery-ap-southeast-7-69ef66ac.s3.ap-southeast-7.amazonaws.com/latest/windows_legacy/AwsReplicationWindowsLegacyInstaller.exe
Canada (Central)	ca-central-1	https://aws-elastic-disaster-recovery-ca-central-1.s3.ca-central-1.amazonaws.com/latest/windows_legacy/AwsReplicationWindowsLegacyInstaller.exe
Canada West (Calgary)	ca-west-1	https://aws-elastic-disaster-recovery-ca-west-1-2590fa22.s3.ca-west-1.amazonaws.com/latest/windows_legacy/AwsReplicationWindowsLegacyInstaller.exe

Region name	Region identity	Download Link
Europe (Frankfurt)	eu-central-1	https://aws-elastic-disaster-recovery-eu-central-1.s3.eu-central-1.amazonaws.com/latest/windows_legacy/AwsReplicationWindowsLegacyInstaller.exe
Europe (Zurich)	eu-central-2	https://aws-elastic-disaster-recovery-eu-central-2.s3.eu-central-2.amazonaws.com/latest/windows_legacy/AwsReplicationWindowsLegacyInstaller.exe
Europe (Stockholm)	eu-north-1	https://aws-elastic-disaster-recovery-eu-north-1.s3.eu-north-1.amazonaws.com/latest/windows_legacy/AwsReplicationWindowsLegacyInstaller.exe
Europe (Milan)	eu-south-1	https://aws-elastic-disaster-recovery-eu-south-1.s3.eu-south-1.amazonaws.com/latest/windows_legacy/AwsReplicationWindowsLegacyInstaller.exe
Europe (Spain)	eu-south-2	https://aws-elastic-disaster-recovery-eu-south-2.s3.eu-south-2.amazonaws.com/latest/windows_legacy/AwsReplicationWindowsLegacyInstaller.exe

Region name	Region identity	Download Link
Europe (Ireland)	eu-west-1	https://aws-elastic-disaster-recovery-eu-west-1.s3.eu-west-1.amazonaws.com/latest/windows_legacy/AwsReplicationWindowsLegacyInstaller.exe
Europe (London)	eu-west-2	https://aws-elastic-disaster-recovery-eu-west-2.s3.eu-west-2.amazonaws.com/latest/windows_legacy/AwsReplicationWindowsLegacyInstaller.exe
Europe (Paris)	eu-west-3	https://aws-elastic-disaster-recovery-eu-west-3.s3.eu-west-3.amazonaws.com/latest/windows_legacy/AwsReplicationWindowsLegacyInstaller.exe
Middle East (UAE)	me-central-1	https://aws-elastic-disaster-recovery-me-central-1.s3.me-central-1.amazonaws.com/latest/windows_legacy/AwsReplicationWindowsLegacyInstaller.exe
Middle East (Bahrain)	me-south-1	https://aws-elastic-disaster-recovery-me-south-1.s3.me-south-1.amazonaws.com/latest/windows_legacy/AwsReplicationWindowsLegacyInstaller.exe

Region name	Region identity	Download Link
Mexico (Central)	mx-central-1	https://aws-elastic-disaster-recovery-mx-central-1-1f310737.s3.mx-central-1.amazonaws.com/latest/windows_legacy/AwsReplicationWindowsLegacyInstaller.exe
South America (São Paulo)	sa-east-1	https://aws-elastic-disaster-recovery-sa-east-1.s3.sa-east-1.amazonaws.com/latest/windows_legacy/AwsReplicationWindowsLegacyInstaller.exe
US East (N. Virginia)	us-east-1	https://aws-elastic-disaster-recovery-us-east-1.s3.us-east-1.amazonaws.com/latest/windows_legacy/AwsReplicationWindowsLegacyInstaller.exe
US East (Ohio)	us-east-2	https://aws-elastic-disaster-recovery-us-east-2.s3.us-east-2.amazonaws.com/latest/windows_legacy/AwsReplicationWindowsLegacyInstaller.exe
US West (N. California)	us-west-1	https://aws-elastic-disaster-recovery-us-west-1.s3.us-west-1.amazonaws.com/latest/windows_legacy/AwsReplicationWindowsLegacyInstaller.exe

Region name	Region identity	Download Link
US West (Oregon)	us-west-2	https://aws-elastic-disaster-recovery-us-west-2.s3.us-west-2.amazonaws.com/latest/windows_legacy/AwsReplicationWindowsLegacyInstaller.exe

Validating the downloaded AWS Replication Agent installer for Windows versions 2008 and 2008 R2.

Important

If you need to validate the installer hash, the correct hash is here: `https://aws-elastic-disaster-recovery-hashes-<REGION>.s3.<REGION>.amazonaws.com/latest/windows_legacy/AwsReplicationWindowsLegacyInstaller.exe.sha512`
Replace `<REGION>` with the AWS Region into which you are replicating, for example: us-east-1:
`https://aws-elastic-disaster-recovery-hashes-us-east-1.s3.us-east-1.amazonaws.com/latest/windows_legacy/AwsReplicationWindowsLegacyInstaller.exe.sha512`

Region name	Region identity	SHA512 Hash Download Link
Africa (Cape Town)	af-south-1	https://aws-elastic-disaster-recovery-hashes-af-south-1.s3.af-south-1.amazonaws.com/latest/windows_legacy/AwsReplicationWindowsLegacyInstaller.exe.sha512
Asia Pacific (Hong Kong)	ap-east-1	https://aws-elastic-disaster-recovery-hashes-ap-east-1

Region name	Region identity	SHA512 Hash Download Link
		.s3.ap-east-1.amazonaws.com/latest/windows_legacy/AwsReplicationWindowsLegacyInstaller.exe.sha512
Asia Pacific (Taipei)	ap-east-2	https://aws-elastic-disaster-recovery-hashes-ap-east-2-f1dcde1a.s3.ap-east-2.amazonaws.com/latest/windows_legacy/AwsReplicationWindowsLegacyInstaller.exe.sha512
Asia Pacific (Tokyo)	ap-northeast-1	https://aws-elastic-disaster-recovery-hashes-ap-northeast-1.s3.ap-northeast-1.amazonaws.com/latest/windows_legacy/AwsReplicationWindowsLegacyInstaller.exe.sha512
Asia Pacific (Seoul)	ap-northeast-2	https://aws-elastic-disaster-recovery-hashes-ap-northeast-2.s3.ap-northeast-2.amazonaws.com/latest/windows_legacy/AwsReplicationWindowsLegacyInstaller.exe.sha512
Asia Pacific (Osaka)	ap-northeast-3	https://aws-elastic-disaster-recovery-hashes-ap-northeast-3.s3.ap-northeast-3.amazonaws.com/latest/windows_legacy/AwsReplicationWindowsLegacyInstaller.exe.sha512

Region name	Region identity	SHA512 Hash Download Link
Asia Pacific (Mumbai)	ap-south-1	https://aws-elastic-disaster-recovery-hashes-ap-south-1.s3.ap-south-1.amazonaws.com/latest/windows_legacy/AwsReplicationWindowsLegacyInstaller.exe.sha512
Asia Pacific (Hyderabad)	ap-south-2	https://aws-elastic-disaster-recovery-hashes-ap-south-2.s3.ap-south-2.amazonaws.com/latest/windows_legacy/AwsReplicationWindowsLegacyInstaller.exe.sha512
Asia Pacific (Singapore)	ap-southeast-1	https://aws-elastic-disaster-recovery-hashes-ap-southeast-1.s3.ap-southeast-1.amazonaws.com/latest/windows_legacy/AwsReplicationWindowsLegacyInstaller.exe.sha512
Asia Pacific (Sydney)	ap-southeast-2	https://aws-elastic-disaster-recovery-hashes-ap-southeast-2.s3.ap-southeast-2.amazonaws.com/latest/windows_legacy/AwsReplicationWindowsLegacyInstaller.exe.sha512

Region name	Region identity	SHA512 Hash Download Link
Asia Pacific (Jakarta)	ap-southeast-3	https://aws-elastic-disaster-recovery-hashes-ap-southeast-3.s3.ap-southeast-3.amazonaws.com/latest/windows_legacy/AwsReplicationWindowsLegacyInstaller.exe.sha512
Asia Pacific (Melbourne)	ap-southeast-4	https://aws-elastic-disaster-recovery-hashes-ap-southeast-4.s3.ap-southeast-4.amazonaws.com/latest/windows_legacy/AwsReplicationWindowsLegacyInstaller.exe.sha512
Asia Pacific (Malaysia)	ap-southeast-5	https://aws-elastic-disaster-recovery-hashes-ap-southeast-5-ebaf53cb.s3.ap-southeast-5.amazonaws.com/latest/windows_legacy/AwsReplicationWindowsLegacyInstaller.exe.sha512
Asia Pacific (New Zealand)	ap-southeast-6	https://aws-elastic-disaster-recovery-hashes-ap-southeast-6-8a759f92.s3.ap-southeast-6.amazonaws.com/latest/windows_legacy/AwsReplicationWindowsLegacyInstaller.exe.sha512

Region name	Region identity	SHA512 Hash Download Link
Asia Pacific (Thailand)	ap-southeast-7	https://aws-elastic-disaster-recovery-hashes-ap-southeast-7-69ef66ac.s3.ap-southeast-7.amazonaws.com/latest/windows_legacy/AwsReplicationWindowsLegacyInstaller.exe.sha512
Canada (Central)	ca-central-1	https://aws-elastic-disaster-recovery-hashes-ca-central-1.s3.ca-central-1.amazonaws.com/latest/windows_legacy/AwsReplicationWindowsLegacyInstaller.exe.sha512
Canada West (Calgary)	ca-west-1	https://aws-elastic-disaster-recovery-hashes-ca-west-1-2590fa22.s3.ca-west-1.amazonaws.com/latest/windows_legacy/AwsReplicationWindowsLegacyInstaller.exe.sha512
Europe (Frankfurt)	eu-central-1	https://aws-elastic-disaster-recovery-hashes-eu-central-1.s3.eu-central-1.amazonaws.com/latest/windows_legacy/AwsReplicationWindowsLegacyInstaller.exe.sha512

Region name	Region identity	SHA512 Hash Download Link
Europe (Zurich)	eu-central-2	https://aws-elastic-disaster-recovery-hashes-eu-central-2.s3.eu-central-2.amazonaws.com/latest/windows_legacy/AwsReplicationWindowsLegacyInstaller.exe.sha512
Europe (Stockholm)	eu-north-1	https://aws-elastic-disaster-recovery-hashes-eu-north-1.s3.eu-north-1.amazonaws.com/latest/windows_legacy/AwsReplicationWindowsLegacyInstaller.exe.sha512
Europe (Milan)	eu-south-1	https://aws-elastic-disaster-recovery-hashes-eu-south-1.s3.eu-south-1.amazonaws.com/latest/windows_legacy/AwsReplicationWindowsLegacyInstaller.exe.sha512
Europe (Spain)	eu-south-2	https://aws-elastic-disaster-recovery-hashes-eu-south-2.s3.eu-south-2.amazonaws.com/latest/windows_legacy/AwsReplicationWindowsLegacyInstaller.exe.sha512

Region name	Region identity	SHA512 Hash Download Link
Europe (Ireland)	eu-west-1	https://aws-elastic-disaster-recovery-hashes-eu-west-1.s3.eu-west-1.amazonaws.com/latest/windows_legacy/AwsReplicationWindowsLegacyInstaller.exe.sha512
Europe (London)	eu-west-2	https://aws-elastic-disaster-recovery-hashes-eu-west-2.s3.eu-west-2.amazonaws.com/latest/windows_legacy/AwsReplicationWindowsLegacyInstaller.exe.sha512
Europe (Paris)	eu-west-3	https://aws-elastic-disaster-recovery-hashes-eu-west-3.s3.eu-west-3.amazonaws.com/latest/windows_legacy/AwsReplicationWindowsLegacyInstaller.exe.sha512
Middle East (UAE)	me-central-1	https://aws-elastic-disaster-recovery-hashes-me-central-1.s3.me-central-1.amazonaws.com/latest/windows_legacy/AwsReplicationWindowsLegacyInstaller.exe.sha512

Region name	Region identity	SHA512 Hash Download Link
Middle East (Bahrain)	me-south-1	https://aws-elastic-disaster-recovery-hashes-me-south-1.s3.me-south-1.amazonaws.com/latest/windows_legacy/AwsReplicationWindowsLegacyInstaller.exe.sha512
Mexico (Central)	mx-central-1	https://aws-elastic-disaster-recovery-hashes-mx-central-1-1f310737.s3.mx-central-1.amazonaws.com/latest/windows_legacy/AwsReplicationWindowsLegacyInstaller.exe.sha512
South America (São Paulo)	sa-east-1	https://aws-elastic-disaster-recovery-hashes-sa-east-1.s3.sa-east-1.amazonaws.com/latest/windows_legacy/AwsReplicationWindowsLegacyInstaller.exe.sha512
US East (N. Virginia)	us-east-1	https://aws-elastic-disaster-recovery-hashes-us-east-1.s3.us-east-1.amazonaws.com/latest/windows_legacy/AwsReplicationWindowsLegacyInstaller.exe.sha512
US East (Ohio)	us-east-2	https://aws-elastic-disaster-recovery-hashes-us-east-2.s3.us-east-2.amazonaws.com/latest/windows_legacy/AwsReplicationWindowsLegacyInstaller.exe.sha512

Region name	Region identity	SHA512 Hash Download Link
US West (N. California)	us-west-1	https://aws-elastic-disaster-recovery-hashes-us-west-1.s3.us-west-1.amazonaws.com/latest/windows_legacy/AwsReplicationWindowsLegacyInstaller.exe.sha512
US West (Oregon)	us-west-2	https://aws-elastic-disaster-recovery-hashes-us-west-2.s3.us-west-2.amazonaws.com/latest/windows_legacy/AwsReplicationWindowsLegacyInstaller.exe.sha512

AWS Replication Agent download URL for Windows 2012 for each supported AWS Region

Region name	Region identity	Download Link
Africa (Cape Town)	af-south-1	https://aws-elastic-disaster-recovery-af-south-1.s3.af-south-1.amazonaws.com/latest/windows_legacy/windows_2012_legacy/AwsReplicationWindows2012LegacyInstaller.exe
Asia Pacific (Hong Kong)	ap-east-1	https://aws-elastic-disaster-recovery-ap-east-1.s3.ap-east-1.amazonaws.com/latest/windows_legacy/windows_2012_legacy/AwsReplicationWindows2012LegacyInstaller.exe

Region name	Region identity	Download Link
Asia Pacific (Taipei)	ap-east-2	https://aws-elastic-disaster-recovery-ap-east-2-f1dcde1a.s3.ap-east-2.amazonaws.com/latest/windows_legacy/windows_2012_legacy/AwsReplicationWindows2012LegacyInstaller.exe
Asia Pacific (Tokyo)	ap-northeast-1	https://aws-elastic-disaster-recovery-ap-northeast-1.s3.ap-northeast-1.amazonaws.com/latest/windows_legacy/windows_2012_legacy/AwsReplicationWindows2012LegacyInstaller.exe
Asia Pacific (Seoul)	ap-northeast-2	https://aws-elastic-disaster-recovery-ap-northeast-2.s3.ap-northeast-2.amazonaws.com/latest/windows_legacy/windows_2012_legacy/AwsReplicationWindows2012LegacyInstaller.exe
Asia Pacific (Osaka)	ap-northeast-3	https://aws-elastic-disaster-recovery-ap-northeast-3.s3.ap-northeast-3.amazonaws.com/latest/windows_legacy/windows_2012_legacy/AwsReplicationWindows2012LegacyInstaller.exe

Region name	Region identity	Download Link
Asia Pacific (Mumbai)	ap-south-1	https://aws-elastic-disaster-recovery-ap-south-1.s3.ap-south-1.amazonaws.com/latest/windows_legacy/windows_2012_legacy/AwsReplicationWindows2012LegacyInstaller.exe
Asia Pacific (Hyderabad)	ap-south-2	https://aws-elastic-disaster-recovery-ap-south-2.s3.ap-south-2.amazonaws.com/latest/windows_legacy/windows_2012_legacy/AwsReplicationWindows2012LegacyInstaller.exe
Asia Pacific (Singapore)	ap-southeast-1	https://aws-elastic-disaster-recovery-ap-southeast-1.s3.ap-southeast-1.amazonaws.com/latest/windows_legacy/windows_2012_legacy/AwsReplicationWindows2012LegacyInstaller.exe
Asia Pacific (Sydney)	ap-southeast-2	https://aws-elastic-disaster-recovery-ap-southeast-2.s3.ap-southeast-2.amazonaws.com/latest/windows_legacy/windows_2012_legacy/AwsReplicationWindows2012LegacyInstaller.exe

Region name	Region identity	Download Link
Asia Pacific (Jakarta)	ap-southeast-3	https://aws-elastic-disaster-recovery-ap-southeast-3.s3.ap-southeast-3.amazonaws.com/latest/windows_legacy/windows_2012_legacy/AwsReplicationWindows2012LegacyInstaller.exe
Asia Pacific (Melbourne)	ap-southeast-4	https://aws-elastic-disaster-recovery-ap-southeast-4.s3.ap-southeast-4.amazonaws.com/latest/windows_legacy/windows_2012_legacy/AwsReplicationWindows2012LegacyInstaller.exe
Asia Pacific (Malaysia)	ap-southeast-5	https://aws-elastic-disaster-recovery-ap-southeast-5-ebaf53cb.s3.ap-southeast-5.amazonaws.com/latest/windows_legacy/windows_2012_legacy/AwsReplicationWindows2012LegacyInstaller.exe
Asia Pacific (New Zealand)	ap-southeast-6	https://aws-elastic-disaster-recovery-ap-southeast-6-8a759f92.s3.ap-southeast-6.amazonaws.com/latest/windows_legacy/windows_2012_legacy/AwsReplicationWindows2012LegacyInstaller.exe

Region name	Region identity	Download Link
Asia Pacific (Thailand)	ap-southeast-7	https://aws-elastic-disaster-recovery-ap-southeast-7-69ef66ac.s3.ap-southeast-7.amazonaws.com/latest/windows_legacy/windows_2012_legacy/AwsReplicationWindows2012LegacyInstaller.exe
Canada (Central)	ca-central-1	https://aws-elastic-disaster-recovery-ca-central-1.s3.ca-central-1.amazonaws.com/latest/windows_legacy/windows_2012_legacy/AwsReplicationWindows2012LegacyInstaller.exe
Canada West (Calgary)	ca-west-1	https://aws-elastic-disaster-recovery-ca-west-1-2590fa22.s3.ca-west-1.amazonaws.com/latest/windows_legacy/windows_2012_legacy/AwsReplicationWindows2012LegacyInstaller.exe
Europe (Frankfurt)	eu-central-1	https://aws-elastic-disaster-recovery-eu-central-1.s3.eu-central-1.amazonaws.com/latest/windows_legacy/windows_2012_legacy/AwsReplicationWindows2012LegacyInstaller.exe

Region name	Region identity	Download Link
Europe (Zurich)	eu-central-2	https://aws-elastic-disaster-recovery-eu-central-2.s3.eu-central-2.amazonaws.com/latest/windows_legacy/windows_2012_legacy/AwsReplicationWindows2012LegacyInstaller.exe
Europe (Stockholm)	eu-north-1	https://aws-elastic-disaster-recovery-eu-north-1.s3.eu-north-1.amazonaws.com/latest/windows_legacy/windows_2012_legacy/AwsReplicationWindows2012LegacyInstaller.exe
Europe (Milan)	eu-south-1	https://aws-elastic-disaster-recovery-eu-south-1.s3.eu-south-1.amazonaws.com/latest/windows_legacy/windows_2012_legacy/AwsReplicationWindows2012LegacyInstaller.exe
Europe (Spain)	eu-south-2	https://aws-elastic-disaster-recovery-eu-south-2.s3.eu-south-2.amazonaws.com/latest/windows_legacy/windows_2012_legacy/AwsReplicationWindows2012LegacyInstaller.exe

Region name	Region identity	Download Link
Europe (Ireland)	eu-west-1	https://aws-elastic-disaster-recovery-eu-west-1.s3.eu-west-1.amazonaws.com/latest/windows_legacy/windows_2012_legacy/AwsReplicationWindows2012LegacyInstaller.exe
Europe (London)	eu-west-2	https://aws-elastic-disaster-recovery-eu-west-2.s3.eu-west-2.amazonaws.com/latest/windows_legacy/windows_2012_legacy/AwsReplicationWindows2012LegacyInstaller.exe
Europe (Paris)	eu-west-3	https://aws-elastic-disaster-recovery-eu-west-3.s3.eu-west-3.amazonaws.com/latest/windows_legacy/windows_2012_legacy/AwsReplicationWindows2012LegacyInstaller.exe
Middle East (UAE)	me-central-1	https://aws-elastic-disaster-recovery-me-central-1.s3.me-central-1.amazonaws.com/latest/windows_legacy/windows_2012_legacy/AwsReplicationWindows2012LegacyInstaller.exe

Region name	Region identity	Download Link
Middle East (Bahrain)	me-south-1	https://aws-elastic-disaster-recovery-me-south-1.s3.me-south-1.amazonaws.com/latest/windows_legacy/windows_2012_legacy/AwsReplicationWindows2012LegacyInstaller.exe
Mexico (Central)	mx-central-1	https://aws-elastic-disaster-recovery-mx-central-1-1f310737.s3.mx-central-1.amazonaws.com/latest/windows_legacy/windows_2012_legacy/AwsReplicationWindows2012LegacyInstaller.exe
South America (São Paulo)	sa-east-1	https://aws-elastic-disaster-recovery-sa-east-1.s3.sa-east-1.amazonaws.com/latest/windows_legacy/windows_2012_legacy/AwsReplicationWindows2012LegacyInstaller.exe
US East (N. Virginia)	us-east-1	https://aws-elastic-disaster-recovery-us-east-1.s3.us-east-1.amazonaws.com/latest/windows_legacy/windows_2012_legacy/AwsReplicationWindows2012LegacyInstaller.exe

Region name	Region identity	Download Link
US East (Ohio)	us-east-2	https://aws-elastic-disaster-recovery-us-east-2.s3.us-east-2.amazonaws.com/latest/windows_legacy/windows_2012_legacy/AwsReplicationWindows2012LegacyInstaller.exe
US West (N. California)	us-west-1	https://aws-elastic-disaster-recovery-us-west-1.s3.us-west-1.amazonaws.com/latest/windows_legacy/windows_2012_legacy/AwsReplicationWindows2012LegacyInstaller.exe
US West (Oregon)	us-west-2	https://aws-elastic-disaster-recovery-us-west-2.s3.us-west-2.amazonaws.com/latest/windows_legacy/windows_2012_legacy/AwsReplicationWindows2012LegacyInstaller.exe

Validating the downloaded AWS Replication Agent installer for Windows 2012.

Important

If you need to validate the installer hash, the correct hash is here: `https://aws-elastic-disaster-recovery-hashes-<REGION>.s3.<REGION>.amazonaws.com/latest/windows_legacy/windows_2012_legacy/AwsReplicationWindows2012LegacyInstaller.exe.sha512`
Replace <REGION> with the AWS Region into which you are replicating, for example: us-east-1:

```
https://aws-elastic-disaster-recovery-hashes-us-east-1.s3.us-
east-1.amazonaws.com/latest/windows_legacy/windows_2012_legacy/
AwsReplicationWindows2012LegacyInstaller.exe.sha512
```

Region name	Region identity	SHA512 Hash Download Link
Africa (Cape Town)	af-south-1	https://aws-elastic-disaster-recovery-hashes-af-south-1.s3.af-south-1.amazonaws.com/latest/windows_legacy/windows_2012_legacy/AwsReplicationWindows2012LegacyInstaller.exe.sha512
Asia Pacific (Hong Kong)	ap-east-1	https://aws-elastic-disaster-recovery-hashes-ap-east-1.s3.ap-east-1.amazonaws.com/latest/windows_legacy/windows_2012_legacy/AwsReplicationWindows2012LegacyInstaller.exe.sha512
Asia Pacific (Taipei)	ap-east-2	https://aws-elastic-disaster-recovery-hashes-ap-east-2-f1dcde1a.s3.ap-east-2.amazonaws.com/latest/windows_legacy/windows_2012_legacy/AwsReplicationWindows2012LegacyInstaller.exe.sha512
Asia Pacific (Tokyo)	ap-northeast-1	https://aws-elastic-disaster-recovery-hashes-ap-

Region name	Region identity	SHA512 Hash Download Link
		northeast-1.s3.ap-northeast-1.amazonaws.com/latest/windows_legacy/windows_2012_legacy/AwsReplicationWindows2012LegacyInstaller.exe.sha512
Asia Pacific (Seoul)	ap-northeast-2	https://aws-elastic-disaster-recovery-hashes-ap-northeast-2.s3.ap-northeast-2.amazonaws.com/latest/windows_legacy/windows_2012_legacy/AwsReplicationWindows2012LegacyInstaller.exe.sha512
Asia Pacific (Osaka)	ap-northeast-3	https://aws-elastic-disaster-recovery-hashes-ap-northeast-3.s3.ap-northeast-3.amazonaws.com/latest/windows_legacy/windows_2012_legacy/AwsReplicationWindows2012LegacyInstaller.exe.sha512

Region name	Region identity	SHA512 Hash Download Link
Asia Pacific (Mumbai)	ap-south-1	https://aws-elastic-disaster-recovery-hashes-ap-south-1.s3.ap-south-1.amazonaws.com/latest/windows_legacy/windows_2012_legacy/AwsReplicationWindows2012LegacyInstaller.exe.sha512
Asia Pacific (Hyderabad)	ap-south-2	https://aws-elastic-disaster-recovery-hashes-ap-south-2.s3.ap-south-2.amazonaws.com/latest/windows_legacy/windows_2012_legacy/AwsReplicationWindows2012LegacyInstaller.exe.sha512
Asia Pacific (Singapore)	ap-southeast-1	https://aws-elastic-disaster-recovery-hashes-ap-southeast-1.s3.ap-southeast-1.amazonaws.com/latest/windows_legacy/windows_2012_legacy/AwsReplicationWindows2012LegacyInstaller.exe.sha512

Region name	Region identity	SHA512 Hash Download Link
Asia Pacific (Sydney)	ap-southeast-2	https://aws-elastic-disaster-recovery-hashes-ap-southeast-2.s3.ap-southeast-2.amazonaws.com/latest/windows_legacy/windows_2012_legacy/AwsReplicationWindows2012LegacyInstaller.exe.sha512
Asia Pacific (Jakarta)	ap-southeast-3	https://aws-elastic-disaster-recovery-hashes-ap-southeast-3.s3.ap-southeast-3.amazonaws.com/latest/windows_legacy/windows_2012_legacy/AwsReplicationWindows2012LegacyInstaller.exe.sha512
Asia Pacific (Melbourne)	ap-southeast-4	https://aws-elastic-disaster-recovery-hashes-ap-southeast-4.s3.ap-southeast-4.amazonaws.com/latest/windows_legacy/windows_2012_legacy/AwsReplicationWindows2012LegacyInstaller.exe.sha512

Region name	Region identity	SHA512 Hash Download Link
Asia Pacific (Malaysia)	ap-southeast-5	https://aws-elastic-disaster-recovery-hashes-ap-southeast-5-ebaf53cb.s3.ap-southeast-5.amazonaws.com/latest/windows_legacy/windows_2012_legacy/AwsReplicationWindows2012LegacyInstaller.exe.sha512
Asia Pacific (New Zealand)	ap-southeast-6	https://aws-elastic-disaster-recovery-hashes-ap-southeast-6-8a759f92.s3.ap-southeast-6.amazonaws.com/latest/windows_legacy/windows_2012_legacy/AwsReplicationWindows2012LegacyInstaller.exe.sha512
Asia Pacific (Thailand)	ap-southeast-7	https://aws-elastic-disaster-recovery-hashes-ap-southeast-7-69ef66ac.s3.ap-southeast-7.amazonaws.com/latest/windows_legacy/windows_2012_legacy/AwsReplicationWindows2012LegacyInstaller.exe.sha512

Region name	Region identity	SHA512 Hash Download Link
Canada (Central)	ca-central-1	https://aws-elastic-disaster-recovery-hashes-ca-central-1.s3.ca-central-1.amazonaws.com/latest/windows_legacy/windows_2012_legacy/AwsReplicationWindows2012LegacyInstaller.exe.sha512
Canada West (Calgary)	ca-west-1	https://aws-elastic-disaster-recovery-hashes-ca-west-1-2590fa22.s3.ca-west-1.amazonaws.com/latest/windows_legacy/windows_2012_legacy/AwsReplicationWindows2012LegacyInstaller.exe.sha512
Europe (Frankfurt)	eu-central-1	https://aws-elastic-disaster-recovery-hashes-eu-central-1.s3.eu-central-1.amazonaws.com/latest/windows_legacy/windows_2012_legacy/AwsReplicationWindows2012LegacyInstaller.exe.sha512
Europe (Zurich)	eu-central-2	https://aws-elastic-disaster-recovery-hashes-eu-central-2.s3.eu-central-2.amazonaws.com/latest/windows_legacy/windows_2012_legacy/AwsReplicationWindows2012LegacyInstaller.exe.sha512

Region name	Region identity	SHA512 Hash Download Link
Europe (Stockholm)	eu-north-1	https://aws-elastic-disaster-recovery-hashes-eu-north-1.s3.eu-north-1.amazonaws.com/latest/windows_legacy/windows_2012_legacy/AwsReplicationWindows2012LegacyInstaller.exe.sha512
Europe (Milan)	eu-south-1	https://aws-elastic-disaster-recovery-hashes-eu-south-1.s3.eu-south-1.amazonaws.com/latest/windows_legacy/windows_2012_legacy/AwsReplicationWindows2012LegacyInstaller.exe.sha512
Europe (Spain)	eu-south-2	https://aws-elastic-disaster-recovery-hashes-eu-south-2.s3.eu-south-2.amazonaws.com/latest/windows_legacy/windows_2012_legacy/AwsReplicationWindows2012LegacyInstaller.exe.sha512

Region name	Region identity	SHA512 Hash Download Link
Europe (Ireland)	eu-west-1	https://aws-elastic-disaster-recovery-hashes-eu-west-1.s3.eu-west-1.amazonaws.com/latest/windows_legacy/windows_2012_legacy/AwsReplicationWindows2012LegacyInstaller.exe.sha512
Europe (London)	eu-west-2	https://aws-elastic-disaster-recovery-hashes-eu-west-2.s3.eu-west-2.amazonaws.com/latest/windows_legacy/windows_2012_legacy/AwsReplicationWindows2012LegacyInstaller.exe.sha512
Europe (Paris)	eu-west-3	https://aws-elastic-disaster-recovery-hashes-eu-west-3.s3.eu-west-3.amazonaws.com/latest/windows_legacy/windows_2012_legacy/AwsReplicationWindows2012LegacyInstaller.exe.sha512

Region name	Region identity	SHA512 Hash Download Link
Middle East (UAE)	me-central-1	https://aws-elastic-disaster-recovery-hashes-me-central-1.s3.me-central-1.amazonaws.com/latest/windows_legacy/windows_2012_legacy/AwsReplicationWindows2012LegacyInstaller.exe.sha512
Middle East (Bahrain)	me-south-1	https://aws-elastic-disaster-recovery-hashes-me-south-1.s3.me-south-1.amazonaws.com/latest/windows_legacy/windows_2012_legacy/AwsReplicationWindows2012LegacyInstaller.exe.sha512
Mexico (Central)	mx-central-1	https://aws-elastic-disaster-recovery-hashes-mx-central-1-1f310737.s3.mx-central-1.amazonaws.com/latest/windows_legacy/windows_2012_legacy/AwsReplicationWindows2012LegacyInstaller.exe.sha512
South America (São Paulo)	sa-east-1	https://aws-elastic-disaster-recovery-hashes-sa-east-1.s3.sa-east-1.amazonaws.com/latest/windows_legacy/windows_2012_legacy/AwsReplicationWindows2012LegacyInstaller.exe.sha512

Region name	Region identity	SHA512 Hash Download Link
US East (N. Virginia)	us-east-1	https://aws-elastic-disaster-recovery-hashes-us-east-1.s3.us-east-1.amazonaws.com/latest/windows_legacy/windows_2012_legacy/AwsReplicationWindows2012LegacyInstaller.exe.sha512
US East (Ohio)	us-east-2	https://aws-elastic-disaster-recovery-hashes-us-east-2.s3.us-east-2.amazonaws.com/latest/windows_legacy/windows_2012_legacy/AwsReplicationWindows2012LegacyInstaller.exe.sha512
US West (N. California)	us-west-1	https://aws-elastic-disaster-recovery-hashes-us-west-1.s3.us-west-1.amazonaws.com/latest/windows_legacy/windows_2012_legacy/AwsReplicationWindows2012LegacyInstaller.exe.sha512
US West (Oregon)	us-west-2	https://aws-elastic-disaster-recovery-hashes-us-west-2.s3.us-west-2.amazonaws.com/latest/windows_legacy/windows_2012_legacy/AwsReplicationWindows2012LegacyInstaller.exe.sha512

Installing the agent

1. Run the agent installer file `AwsReplicationWindowsInstaller.exe` as an Administrator.

```
.\AwsReplicationWindowsInstaller.exe
```

   The installer confirms that the installation of the AWS Replication Agent has started.

```
The installation of the AWS Replication Agent has started.
```

> **Note**
>
> To install the agent on a secured network, learn about the additional required configurations.

2. The installer prompts you to enter your **AWS Region Name**, the **AWS Access Key ID** and the **AWS Secret Access Key** that you previously generated. Enter the complete AWS Region name (for example: eu-central-1), and the full AWS Access Key ID and AWS Secret Access Key. If you are using temporary credentials, you also need to specify the session token.

```
The installation of the AWS Replication Agent has started.
AWS Region name: us-east-1
AWS Access Key ID: AKIAI0SF0DNN71EXAMPLE
AWS Secret Access Key: wJalrXUtnFEMI/K71MDENG/bPxRfiCYEXAMPLEKEY
```

> **Note**
>
> You can also enter these values as part of the installation script command parameters. If you do not enter these parameters as part of the installation script, you are prompted to enter them one by one as described above. (for example: `.\AwsReplicationWindowsInstaller.exe --region regionname --aws-access-key-id AKIAIOSFODNN7EXAMPLE --aws-secret-access-key wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY`)

> **Note**
>
> You can also pass credentials through environment variables. We recommend using temporary credentials from AWS STS. Set `AWS_ACCESS_KEY_ID`, `AWS_SECRET_ACCESS_KEY`, and `AWS_SESSION_TOKEN` in your PowerShell session. Then run the installer with `--no-prompt`:
>
> ```
> $env:AWS_ACCESS_KEY_ID = "AKIAIOSFODNN7EXAMPLE"
> $env:AWS_SECRET_ACCESS_KEY = "wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY"
> $env:AWS_SESSION_TOKEN = "AQoDYXdzEJr/////////wEa8AMDSomethingEXAMPLE"
> .\AwsReplicationWindowsInstaller.exe --region us-east-1 --no-prompt
> ```

If you require additional customization, you can add a variety of parameters to the installation script in order to manipulate the way the Agent is installed on your server. See the Installer Parameters for more information.

3. Once you have entered your credentials, the installer verifies that the source server has enough free disk space for Agent installation and identify volumes for replication. The installer displays the identified disks and prompts you to choose the disks you want to replicate.

```
...
AWS Secret Access Key: wJalrXUtnFEMI/K71MDENG/bPxRfiCYEXAMPLEKEY
Verifying that the source server has enough free disk space to install the AWS
 Replication Agent.
(a minimum of 2GB of free disk space is required)
Identifying volumes for replication.
Choose the disks you want to replicate. Your disks are: c:
To replicate some of the disks, type the path of the disks, separated with a comma
 (for example, C:,D:).
To replicate all disks, press Enter:
```

To replicate some of the disks, type the path of the disks, separated by a comma, as illustrated in the installer (for example: C:, D:, etc). To replicate all of the disks, press **Enter**. The installer identifies the selected disks and prints their size.

```
...
Identifying volumes for replication.
```

```
Choose the disks you want to replicate. Your disks are: c:
To replicate some of the disks, type the path of the disks, separated with a comma
 (for example, C:,D:).
To replicate all disks, press Enter:
Disk to replicate identified: c:0 of size 30GiB
```

The installer confirms that all of the disks were successfully identified.

```
...
Identifying volumes for replication.
Choose the disks you want to replicate. Your disks are: c:
To replicate some of the disks, type the path of the disks, separated with a comma
 (for example, C:,D:).
To replicate all disks, press Enter:
Disk to replicate identified: c:0 of size 30GiB
All volumes for replication were successfully identified
```

Note

When identifying specific disks for replication, do not use apostrophes, brackets, or disk paths that do not exist. Type only existing disk paths. Each disk that you selected for replication is displayed with the caption **Disk to replicate identified**. However, the displayed list of identified disks for replication may differ from the data you entered. This difference can be due to several reasons:

- The root disk of the source server is always replicated, whether you select it or not. Therefore, it always appears on the list of identified disks for replication.
- AWS Elastic Disaster Recovery replicates whole disks. Therefore, if you choose to replicate a partition, its entire disk appears on the list and is later replicated. If several partitions on the same disk are selected, then the disk encompassing all of them appears only once on the list.
- Incorrect disks may be chosen by accident. Ensure that the correct disks have been chosen.

> **Important**
>
> If disks are disconnected from a server, AWS Elastic Disaster Recovery can no longer replicate them, so they are removed from the list of replicated disks. When they are reconnected, the AWS Replication Agent cannot know that these were the same disks that were disconnected and therefore does not add them automatically. To add the disks after they are reconnected, rerun the AWS Replication Agent installer on the server.
>
> Note that the returned disks need to be replicated from the beginning. Any disk size changes are automatically identified, but also cause a resync. Perform a test after installing the Agent to ensure that the correct disks have been added.

4. After all of the disks to be replicated have been successfully identified, the installer downloads and installs the AWS Replication Agent on the source server.

```
...
All volumes for replication were successfully identified
Downloading the AWS Replication Agent onto the source server... Finished
Installing the AWS Replication Agent onto the source server... Finished
```

5. Once the AWS Replication Agent is installed, the server is added to the AWS Elastic Disaster Recovery console and undergoes the initial sync process. The installer provides the source server's ID.

```
...
All volumes for replication were successfully identified
Downloading the AWS Replication Agent onto the source server... Finished
Installing the AWS Replication Agent onto the source server... Finished
Syncing the source server with the Elastic Disaster Recovery Console... Finished
The following is the source server ID: s-3146f90b19example
The AWS Replication Agent was successfully installed.
Press Enter to close...
```

You can review this process in real time on the **Source servers** page.

AWS Replication Agent Installer parameters

The AWS Replication Agent Installer supports the following command line parameters.

--region

The region into which the installer registers the source server.

--aws-access-key-id

The AWS IAM Access Key used for authenticating the installing user. If this parameter is not provided, the installer prompts for it.

--aws-secret-access-key

The AWS IAM Secret Access Key tied to the AWS IAM Access Key used for authenticating the installing user. If this parameter is not provided, the installer prompts for it.

--aws-session-token

The session token is generated when using temporary credentials generated using AWS STS.

--account-id

Use this parameter to install the DRS agent on an EC2 instance to replicate to another AWS account without any additional access key or temporary credentials. Specify the 12 digit ID of the account into which you want to replicate your source server. This action requires an EC2 instance profile with the AWSElasticDisasterRecoveryEc2InstancePolicy policy, to define the account to replicate into as a Trusted Account and select the roles in **Failback and in-AWS right-sizing roles**.

--no-prompt

Run the installation without prompting the user.

--devices

Specify exactly which disks to replicate.

--force-volumes

This parameter must be used with the *--no-prompt* parameter. This parameter cancels the automatic detection of physical disks to replicate. You need to specify the exact disks to replicate using the *--devices* parameter (including the root disk, failure to specify the root disk causes replication to fail). This parameter should only be used as a troubleshooting tool if the *--devices* parameter fails to identify the disks correctly.

--tags

Use this parameter to add resource tags to the source server. Use a space to separate each tag.

```
--tags {"Key1"="Value1" "Key2"="Value2"}
```

> **Note**
>
> This flag may only be used when adding new source servers to AWS DRS. You cannot use the --tags flag to modify tags of source servers that have already been added to AWS DRS.

--s3-endpoint

Use this parameter to specify a VPC endpoint you created for S3 if you do not wish to open your firewall ports to access the default S3 endpoint. Learn more about installing the Agent on a blocked network.

--endpoint

Use this parameter to specify the Private Link endpoint you created for Elastic Disaster Recovery if you do not wish to open your firewall ports to access the default AWS Elastic Disaster Recovery endpoint. Learn more about installing the agent on a blocked network.

> **Note**
>
> We do not recommend using this flag when installing the AWS Elastic Disaster Recovery Agent on an EC2 Instance, as it can prevent successful failback from occurring. We recommend ensuring DNS automatically resolves the `{region}.drs.amazonaws.com` entry to the Private Link endpoint rather than leveraging this parameter.

--install-as-recovery-instance

Use this parameter to add an existing AWS instance to AWS Elastic Disaster Recovery as a recovery instance. You may opt to add recovery instances if you have added additional EC2 instances to AWS and now want to recover them into source servers. You are asked to pair the newly added recovery instance with a source server during AWS Replication Agent installation.

--proxy-address

Linux Installer only.

Use this parameter to configure the agent to use a specific proxy server: `--proxy-address https://PROXY:PORT/`. Ensure the proxy configuration has the trailing forward slash (/).

--exclude-instance-store-volumes

Use this parameter to exclude instance store volumes from replication.

--dualstack

Use this parameter to configure the agent to use Elastic Disaster Recovery dual-stack API endpoints. When you specify this parameter, the agent communicates with Elastic Disaster Recovery through `drs.{region}.api.aws` instead of `drs.{region}.amazonaws.com`, and with Amazon S3 through `s3.dualstack.{region}.amazonaws.com` instead of `s3.{region}.amazonaws.com`.

> **Note**
>
> This parameter enables IPv6 support for API communication between the agent and AWS services, with IPv4 as a fallback. This parameter does not set the **IP version** in the replication configuration settings, which determines the Internet Protocol version used for data replication. For more information about the **IP version** setting, see IP version.

> **Environment variable credentials**
>
> Instead of passing credentials as command line parameters, you can set the following environment variables:
>
> - `AWS_ACCESS_KEY_ID`
> - `AWS_SECRET_ACCESS_KEY`
> - `AWS_SESSION_TOKEN` (required when using temporary credentials)
>
> If you set these environment variables and do not provide credential command line parameters, the installer uses the environment variable values. If you provide both command line parameters and environment variables, the installer uses the command line parameters.
> On Linux, use `sudo -E` to preserve the environment variables when running the installer as root.

Installing the agent on a secured network

The AWS Replication Agent installer needs network access to AWS Elastic Disaster Recovery and S3 endpoints. If your on-premises network is not open to AWS Elastic Disaster Recovery and S3 endpoints, you can install the agent by using PrivateLink.

VPC endpoints created through PrivateLink handle only **management traffic** between the AWS Replication Agent and the AWS Elastic Disaster Recovery service. Management traffic includes agent installation, authentication, and API communication.

Replication traffic (the actual data replication between your source servers and the replication servers in AWS) does not use VPC endpoints. Replication traffic flows directly between the source and replication servers over your VPN or DirectConnect connection. To enable private replication, activate the private IP option in the replication settings. VPC endpoints are not required for private replication.

You can connect your on-premises network to the subnet in your staging area VPC using AWS VPN or DirectConnect. To use AWS VPN or DirectConnect, you must activate private IP in the replication settings.

The following topics describe the connectivity prerequisites that enable you to install the agent. All of the settings apply to the target account (or the staging account in a multi-account scenario) and Region where you want to handle the recovery.

Create a VPC Endpoint for AWS Elastic Disaster Recovery

To allow the AWS Replication Agent installer to communicate with AWS Elastic Disaster Recovery, create an interface VPC endpoint for AWS Elastic Disaster Recovery in your staging area subnet. This VPC endpoint is used exclusively for management traffic; replication data is transmitted directly between the source and replication servers. For more information, see Creating an Interface Endpoint in the Amazon VPC User Guide.

If the AWS replication agents are installed with a principal using AWSElasticDisasterRecoveryAgentInstallationPolicy and a VPCE policy is used (to scope down access), add the following statement to your policy:

```
{
     "Effect": "Allow",
     "Principal": "*",
     "Action": "execute-api:Invoke",
```

```
    "Resource": "arn:aws:execute-api:<region>:*.*/POST/CreateSessionForDrs"
    }
```

Use the created VPC Endpoint for AWS Elastic Disaster Recovery

After you create the VPC Endpoint, the AWS Replication Agent can connect to AWS Elastic Disaster Recovery through VPN or DirectConnect by using the `--endpoint` installation parameter. For more information, see Private DNS for interface endpoints in the Amazon VPC User Guide.

Run the AWS Replication Agent installer with the `--endpoint` parameter. Enter your endpoint-specific DNS hostname within the parameter. The installer then connects to AWS Elastic Disaster Recovery through the endpoint over your VPN or DirectConnect connection.

Example of an interface endpoint DNS name: `vpce-0123456789-abcdef.drs.<REGION>.vpce.amazonaws.com`

Create an S3 Endpoint for AWS Elastic Disaster Recovery

To allow the AWS Replication Agent installer to communicate with S3, create an interface S3 endpoint for AWS Elastic Disaster Recovery in your staging area subnet. For more information, see Endpoints for Amazon S3 in the Amazon VPC User Guide. The endpoint requires a security group that allows connection from the agent, enabling it to download components it needs for the installation.

Use the created S3 Endpoint for AWS Elastic Disaster Recovery

After you create the interface VPC Endpoint, the AWS Replication Agent can connect to S3 through VPN or DirectConnect by using the `--s3-endpoint` installation parameter. For more information, see Private DNS for interface endpoints in the Amazon VPC User Guide.

Run the AWS Replication Agent installer with the `--s3-endpoint` parameter. Enter your endpoint-specific DNS hostname. The installer then connects to S3 through the endpoint over your VPN or DirectConnect connection.

Example of an interface endpoint DNS name: `vpce-0123456789-abcdef.s3.<REGION>.vpce.amazonaws.com`

Preparing the AWS VPC

To prepare the staging area subnet in a private subnet, create two more endpoints to ensure the successful creation of the replication servers.

- **EC2 Interface Endpoint** – Used to establish connectivity to the EC2 endpoint from the staging area subnet.
- **S3 Gateway Endpoint** – Used by the replication servers to download the replication software from S3.

For more information about setting up AWS Elastic Disaster Recovery with a site-to-site VPN connection, see Cross-Region AWS Elastic Disaster Recovery agent installation in a secured network.

Uninstalling the agent

Uninstalling the AWS Replication Agent from a source server stops the replication of that server. Uninstalling the AWS Replication Agent removes the source server from the Elastic Disaster Recovery Console.

Uninstalling the Agent through the AWS Elastic Disaster Recovery console

To uninstall the AWS Replication Agent through the AWS Elastic Disaster Recovery console.

Navigate to the **Source servers** page.

Check the box to the left of each server that you want to disconnect from Elastic Disaster Recovery (by uninstalling the AWS Replication Agent). Open the **Actions** menu, and choose the **Disconnect from AWS** option to disconnect the selected server from AWS Elastic Disaster Recovery and AWS.

When the **Disconnect X server/s from service** dialog appears, click **Disconnect**.

The AWS Replication Agent is uninstalled from all of the selected source servers.

Uninstalling the Agent manually through the source server

To uninstall the AWS Replication Agent manually through the source server:

Windows

Copy the following folder to a new location:`C:\Program Files (x86)\AWS Replication Agent\dist`

From the new location, run in CMD as an administrator:

```
install_agent_windows.exe --remove
```

Linux

As root, cd to `/var/lib/aws-replication-agent`.

Run the following commands from that folder:

```
./stopAgent.sh
```

```
./uninstall_agent_linux.sh
```

Reinstalling the agent

To reinstall the AWS Replication Agent, download the latest version of the agent and follow the installation instructions. You do not need to remove any previous versions prior to reinstalling the agent.

- Linux
- Windows

> **Note**
>
> You must reinstall the agent to benefit from new features.

Reinstalling the agent on a recovery instance

If you are reinstalling an agent on a recovery instance:

1. Select your recovery instance and choose **Disconnect from AWS** from the **Actions** drop-down menu.
2. When reinstalling the agent, include the **--install-as-recovery-instance** parameter.

Example:

```
chmod +x aws-replication-installer-init; sudo ./aws-replication-installer-init --
install-as-recovery-instance s-abcd01234567890
```

> **Note**
>
> In order to reinstall the agent on a recovery instance, you need to provide the temporary credentials for a role that has the AWSElasticDisasterRecoveryAgentInstallationPolicy policy.

Supporting marketplace licenses

Installing the AWS replication agent on an EC2 instance on AWS that has one or more active subscriptions to a marketplace license requires taking the following points into consideration:

- Some marketplace products do not function with certain instance types or in certain regions. DRS does not verify if the marketplace license applies to the instance type and region defined. To see if the marketplace product applies to the current settings, visit the marketplace product page. It is also highly recommended to do periodic drills as some of these incompatibilities are only identified upon launch.
- If an agent is to be installed on an EC2 instance existing on one account (source account) which is a different AWS account than the AWS account where DRS is operated (the target account), it is mandatory to provide permissions that allow getting the marketplace license information from the source account. Create a Failback and in-AWS right-sizing role for trusted account using the target account AWS account ID. This role must be created in the source account, or the agent installation fails. If this role is removed or modified, launch operations might fail if new marketplace licenses are added.
- If an agent was installed on an EC2 instance existing on one account (source account), and DRS is operated on a different account (target account), and a new volume, that has a marketplace license associated with it, is connected to the instance with the **Automatically replicate new disks** setting active, the volume might fail to be added if permissions to allow getting the marketplace license information were removed or do not exist. Create a Failback and in-AWS right-sizing role for trusted account using the target account AWS account ID, and re-install the agent if a volume fails to be added due to this reason.
- In case of EC2 instances from one account that replicate to a staging account (see multi-account) and launch in one or more target accounts, only the staging account must have a Failback and in-AWS right-sizing role created for.
- The AWS account where recovery instances are launched (the target account or staging account) must have an active subscription to the same AWS Marketplace product associated with the

source volumes. If the account is not subscribed, the launch will fail with an `OptInRequired` error during volume creation. Ensure that all accounts used for recovery are subscribed to the required Marketplace products before initiating a drill or recovery.

Adding instances from the Amazon EC2 Console

You can now add EC2 instances as source servers in DRS, starting from the EC2 console. New or existing instances can be added by selecting the appropriate action on the EC2 console, sending you to the AWS focused page allowing you to install the AWS replication agent used by DRS on the selected instances.

Add instances

You can protect your EC2 instances using AWS Elastic Disaster Recovery (DRS) in the chosen AWS Region, by adding them to AWS DRS as source servers. Utilize AWS Systems Manager (SSM) if present on your instance to install the AWS replication agent, a step needed to start replicating data from your instance to AWS. Only instances managed by AWS Systems Manager would be able to have the AWS replication agent installed on them.

> **Note**
>
> You need an instance profile with the policies listed below in order to have your instances managed by SSM and for installing the AWS replication agent:
>
> 1. AmazonSSMManagedInstanceCore
> 2. AWSElasticDisasterRecoveryEC2InstancePolicy

Successfully installing the AWS replication agent adds the instance to AWS DRS (as a **source server**) in the chosen target region.

Supported EC2 instances

> **Note**
>
> Any additional EBS volumes added during the EC2 Instance creation that are offline, unmounted, or unformatted are not replicated. Any volume that is later placed online or

mounted with a valid file system is automatically replicated if Automatically replicate new disks is enabled.

This section lists all the instances that were selected to be protected by AWS DRS. The list shows which instances are currently managed by SSM and which instances are currently not managed. Only instances managed by SSM can have the AWS replication agent installed on them using this page. You can also install the agent using the installer as defined in Installing the AWS Replication Agent, without requiring the SSM agent to be present and active on the server to be protected.

To have an instance managed by SSM, requires the SSM agent to be installed on a compatible operating system (or preinstalled in the AMI), and the instance to have the correct permissions (as defined in the AmazonSSMManagedInstanceCore and the AWSElasticDisasterRecoveryEC2InstancePolicy policies). To update the instance profiles, the **Instance profile role installation** section allows you to create the default instance profile (with the two policies mentioned above) if needed. The **Instance profiles** section allows you to assign instance profiles to instances, and automatically assigns the default instance profile to all instances that do not have any instance profile attached to them. Use the **Attach profiles to all instances** button to attach the assigned instance profiles to the instances in case the default profile was created and automatically assigned to them or if you changed the assigned instance profile.

Target disaster recovery region

On this section, you can define the target disaster recovery region. This can be the same region where the instances are present in, or it can be a different region, for cross-region protection. AWS DRS must be initialized in the target region in order to protect the instances onto that region. The indicator next to the region's name shows if AWS DRS is already initialized in the target region, or not. If the region is not initialized, a button labelled Initialize and configure AWS Elastic Disaster Recovery is visible and active. Choosing this button opens the AWS DRS initialization wizard for AWS DRS in the target region on another browser tab.

Instance profile role installation - optional

This section provides you with the option to create the default IAM role with the required permissions as an instance profile. The role **AWSElasticDisasterRecoveryAutomatedAgentInstallRole** includes the permissions defined in the policies AmazonSSMManagedInstanceCore and AWSElasticDisasterRecoveryEC2InstancePolicy. These permissions are required to allow the SSM agent to operate and to install the AWS

replication agent, respectively. Clicking the **Install default IAM role** installs this role. This needs to be done only once per account. If the role was already installed in the account, this button is inactive. The default instance profile role is automatically assigned to instances without an instance profile in the **Instance profiles** section. If you click the **Attach profiles to all instances** button, this role is attached to all instances it was assigned to in the **Instance profiles** section. If this default IAM role is not installed, you need to make sure you have an instance profile with the AmazonSSMManagedInstanceCore and AWSElasticDisasterRecoveryEC2InstancePolicy policies (or the combined set of permissions within both of these policies).

Instance profiles

This section lists all the instances that were selected to be protected by adding them as source servers to AWS DRS and their current instance profiles. Instances without any instance profile have the **AWSElasticDisasterRecoveryAutomatedAgentInstallRole** instance profile and IAM role assigned to them if it exists on this account. Using the default profile is not mandatory, as any instance profile in the account can be assigned to any instance, but care must be taken to verify each instance has an instance profile with the permissions defined in the AmazonSSMManagedInstanceCore and AWSElasticDisasterRecoveryEC2InstancePolicy policies.

> **Note**
>
> AWS DRS does not validate the instance profile has the required permissions to support working with the SSM agent or installing the AWS replication agent for DRS.

> **Note**
>
> Attaching an instance profile with the needed permissions is a mandatory step if you want to install AWS DRS on instances that have the SSM agent installed on them (manually, or preinstalled on AMI) but are not managed on SSM due to missing an instance profile with the AmazonSSMManagedInstanceCore policy.

Click the button labelled **Attach profiles to all instances** to attach the assigned instance profiles to their instances.

After attaching such a profile, allow AWS DRS a few minutes to identify the instance as managed by SSM. If SSM is present on the instance, and an instance profile with the needed permissions was

attached to the instance, then within a few minutes, the marker near the instance ID changes to show that the instance is currently managed by SSM.

Attach profiles to all instances

Clicking this button attaches the instance profiles assigned in the **Instance profiles** section to their instances. After attaching appropriate instance profiles to instances, allow a few minutes for DRS to detect if these instances are managed by SSM.

Add instances

Click this button to install the AWS replication agent on all instances that are currently managed by SSM. If there are such instances, AWS DRS lists those instances and the progress of installing the AWS replication agent on them. Successfully installing the AWS replication agent on these instances adds them as source servers to AWS DRS. If there are no instances that are currently managed by SSM, try installing the SSM agent on these instances, then attach an appropriate instance profile to them.

Add instances result page

On this page you can view the result of adding instances to AWS DRS by installing the AWS replication agent on them. The page shows the progress of this process if currently running, or the summary of the last run. In addition, for each instance that is currently managed by SSM, there is a table listing the following:

Instance ID - The ID of the instance. This also links to the instance on the EC2 console page (opens in a different browser tab).

Status - The current status of the installation, possible values include **Success**, **In Progress**, **Pending** and **Error**.

Details - holds a link to the source servers page on the target region for successful installations, or a link to the run log on the SSM console (opens in a new browser tab) for runs that have failed, are pending or are in progress.

AWS DRS source servers page

The **Source servers** page lists all of the source servers that were added to AWS Elastic Disaster Recovery. It allows you to manage your source servers and perform commands for one or more

servers, such as controlling replication and launching Initiate recovery job instances. You are likely to interact with AWS Elastic Disaster Recovery predominantly through this page.

Topics

- Interacting with the Source Servers page
- Source servers page command menus
- Filtering on the source servers page

Interacting with the Source Servers page

The **Source servers** page shows a list of source servers. Each row on the list represents a single server.

The **Source servers** page provides key information for each source server under each of the columns on the page.

The columns include:

- **Selector column** – This blank checkbox selector column allows you to select one or more source servers. When you select a server you can interact with it through the **Actions**, **Replication**, and **Initiate recovery job** menus. Selected servers are highlighted.
- **Hostname** – This column shows the unique server hostname for each source server.
- - **Ready for recovery** – This column shows whether the server is ready for recovery. You can use this column to easily tell whether a server is ready or not and the server's exact status. You can learn more about the server's status by reviewing the **Data replication status** column.
  - A server that is ready shows the green checkmark and **Ready**.
  - A server that is ready, but is experiencing a non-critical issue such as lag shows the blue info sign and **Ready** and displays the lag duration to the right.
  - A server that is still undergoing initial sync shows a gray circle with three dots and **Initial sync**.
  - A server that is disconnected shows the gray warning sign and **Disconnected**.
  - A server that is not ready due to a significant error, such as a stall, shows a red **X** and **Not Ready**. Servers that have one or more marketplace licenses assigned to them may not be able to launch if there was an error reading their license information.
- **Data replication status** – This column shows the current status of data replication for the server:

- **Initiating** – The server has just been added to AWS Elastic Disaster Recovery and replication is being initiated.
- **Initial sync** – The server is undergoing the initial sync process. The console displays the percentage of the server data that has been synced and the step the server is undergoing in the initial sync process. You can learn more about the exact state of the server in the server info view.
- **Rescanning** – The server is undergoing a rescan. The console displays the percentage of the server data that has been rescanned successfully.
- **Healthy** – The server is healthy and is ready to initiate a recovery job.
- **Lag** – The server is experiencing lag. The console displays the amount of lag time. You can learn more about the exact state of the server in the server info view.
- **Stalled** – The server is stalled due to a replication error. You can learn more about the specific cause of the stall in the server info view.
- **Disconnected** – The server has been disconnected from AWS Elastic Disaster Recovery.

- **Last recovery result** – This column shows the result of the last recovery job launch. The column is empty if no recovery job has ever been launched for the server:
  - **Successful** - Recovery launch job was completed successfully. The console indicates how long ago the job was completed.
  - **Failed** – Recovery launch job failed. The console indicates how long ago the job failed. You can learn more about why the job failed in the job history.
  - **Pending** – Recovery launch job is pending. The console indicates how long ago the job was initiated.
- **Pending actions** – This column shows any pending actions that need to be performed on the server. This column appears empty unless there is an actionable pending action. Actions include:
  - **Initiate drill** – The source server is healthy, but no drill instances have been launched for the source server. Initiate a drill by launching a drill instance.
  - **Resolve cause of stall** – The source server is stalled. Resolve the cause of the stall for the server to return to healthy function.
  - **Reinstall AWS Replication Agent** – The AWS Replication Agent was removed from the source server. Reinstall the agent for replication to resume.
  - **Error: Missing permissions to retrieve marketplace licenses from the source account, cannot launch this server** – The marketplace license belongs to a different AWS account, permissions

to get information about this marketplace license are missing. Create a Failback and in-AWS right-sizing role for trusted account using the target account AWS account ID.

- **Warning: server uses marketplace product, drill recommended** – This source server uses one or multiple marketplace licenses. Doing a drill is strongly recommended as some marketplace incompatibilities can only be identified during launch. Learn more here.

Source servers page command menus

You can perform a variety of actions, control data replication, and manage your drill and recovery instances for one or more source servers through the command menu buttons. Select one or more servers on the **Source servers** page and choose the **Actions**, **Replication**, or **Initiate recovery job** menu to control your source servers.

Topics

- Actions menu
- Initiate recovery job menu
- Replication menu

Actions menu

The **Actions** menu allows you to perform the following actions:

- **Add servers** – Choosing this option redirects you to the AWS Replication Agent installation instructions.
- **Create extended source servers** – Choose this to start a wizard to create extended source servers from source servers replicating into staging accounts, in multi-account setups.
- **Edit DRS launch settings** – Choose this option to edit a single or multiple selected source servers for their DRS launch settings.
- **Edit EC2 launch template** – Choose this option to edit a single or multiple selected source servers for their EC2 launch template.
- **Edit post-launch action settings** – Choose this option to activate or deactivate post-launch actions for a single or multiple selected source servers.
- **View server details** – Choose this option to enter the source server's **Server details view.**
- **Disconnect from AWS** – Choose this option to disconnect the selected server from AWS Elastic Disaster Recovery and AWS.

When the **Disconnect X server/s from service** dialog appears, click **Disconnect**.

> **Important**
>
> This uninstalls the AWS Replication Agent from the source server and data replication will stop for the source server. This action does not affect any Drill or Recovery instances that have been launched for this source server, but you are no longer able to identify which source servers your Amazon EC2 instances correspond to.

- **Delete server** - Choose the **Delete server** option to permanently delete a source server from AWS Elastic Disaster Recovery. This removes all information related to the server from the AWS Elastic Disaster Recovery service. You can only delete servers that have been disconnected from AWS. You need to reinstall the AWS Replication Agent on a deleted source server to add it back to AWS Elastic Disaster Recovery.

  When the **Delete X servers** dialog appears, click **Permanently delete**. Then, if the servers have associated recovery instances, you can either:

  - delete them, keeping the EC2 instances intact,
  - terminate them, which deletes the EC2 instances.

Initiate recovery job menu

The **Initiate recovery job** menu allows you to start drills and recoveries by launching drill and recovery instances as part of the overall failback process. You can learn more about the entire failback and failover process with AWS Elastic Disaster Recovery in the Performing a failback and failover with AWS Elastic Disaster Recovery documentation.

- **Initiate drill** – Choose this option to launch a drill instance for this server or group of servers for the purpose of testing your recovery solution. You should perform periodic drills in order to ensure that you are ready for recovery. Learn more about launching Drill instances in AWS Elastic Disaster Recovery.
- **Initiate recovery** – Choose this option to launch a Recovery instance for this server or group of servers for the purpose of recovering the server in the event of a disaster. Learn more about launching Recovery instances in AWS Elastic Disaster Recovery.

Replication menu

The **Replication** menu allows you to perform the following actions:

- **Stop replication** – You can stop replication of a source server at any time. After you stop the replication, you will no longer be charged for the ongoing replication and the staging area infrastructure. Changes will not be reported by the agent to the replication server, and all saved snapshots will be deleted, leaving this instance unprotected. The agent remains installed during this process. If you want to replicate this EC2 instance again, simply click the **Start replication** button. This triggers an initial sync.
- **Start replication** – You can start replication of a previously stopped source server. After you start the replication, the agent replicates the selected instances.

Filtering on the source servers page

You can customize the **Source servers** page through filtering by recovery readiness.

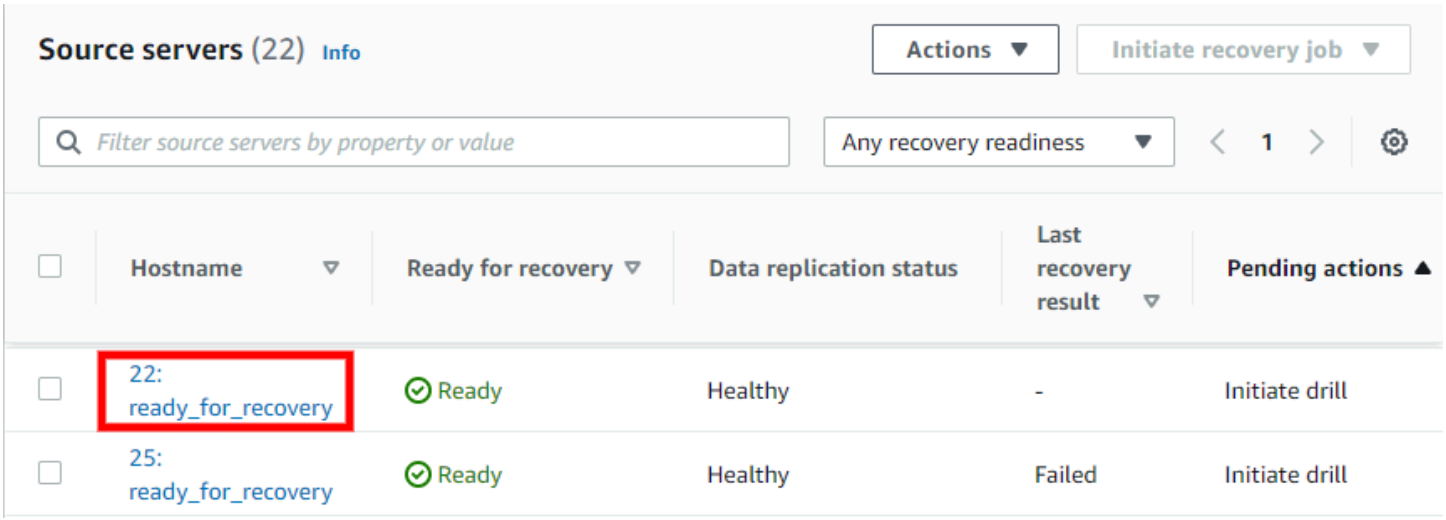
In the **Filter source servers....** field, choose the filtering property from the menu.

You can filter by a variety of properties, including:

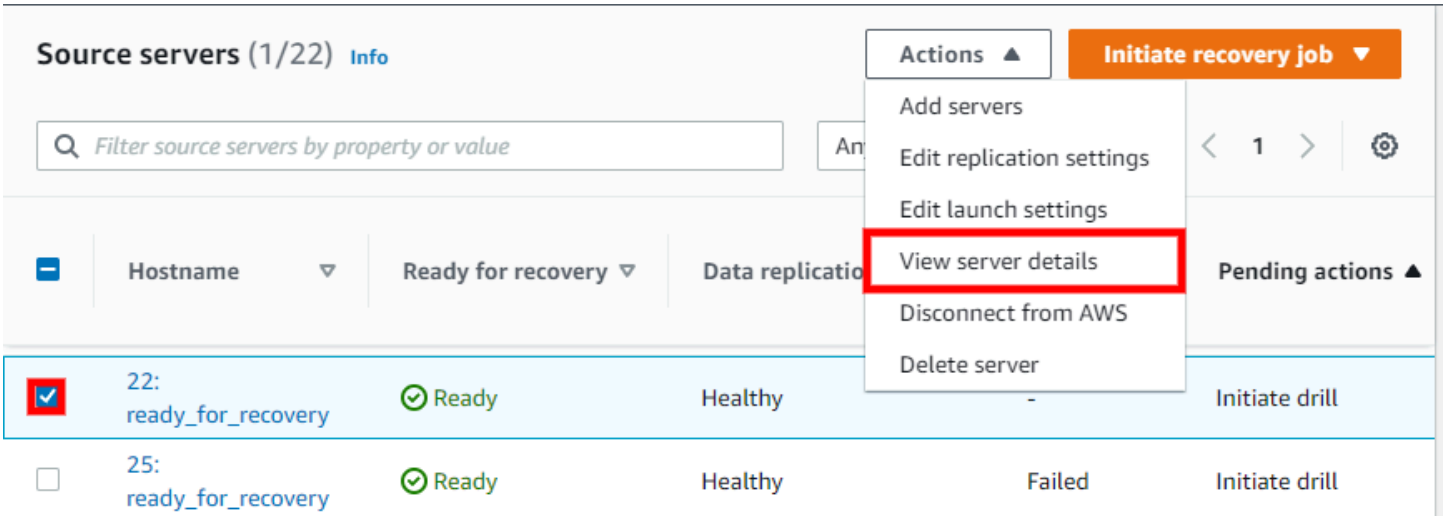
- Any recovery readiness – Filter by specific alert (lagging, stalled, launched)
- Not ready – Filter by a specific hostname or a specific string of characters
- Ready – Filter by the recovery lifecycle state
- Initial sync – Filter by the data replication status
- Ready with lag
- Disconnected

View server details with AWS DRS

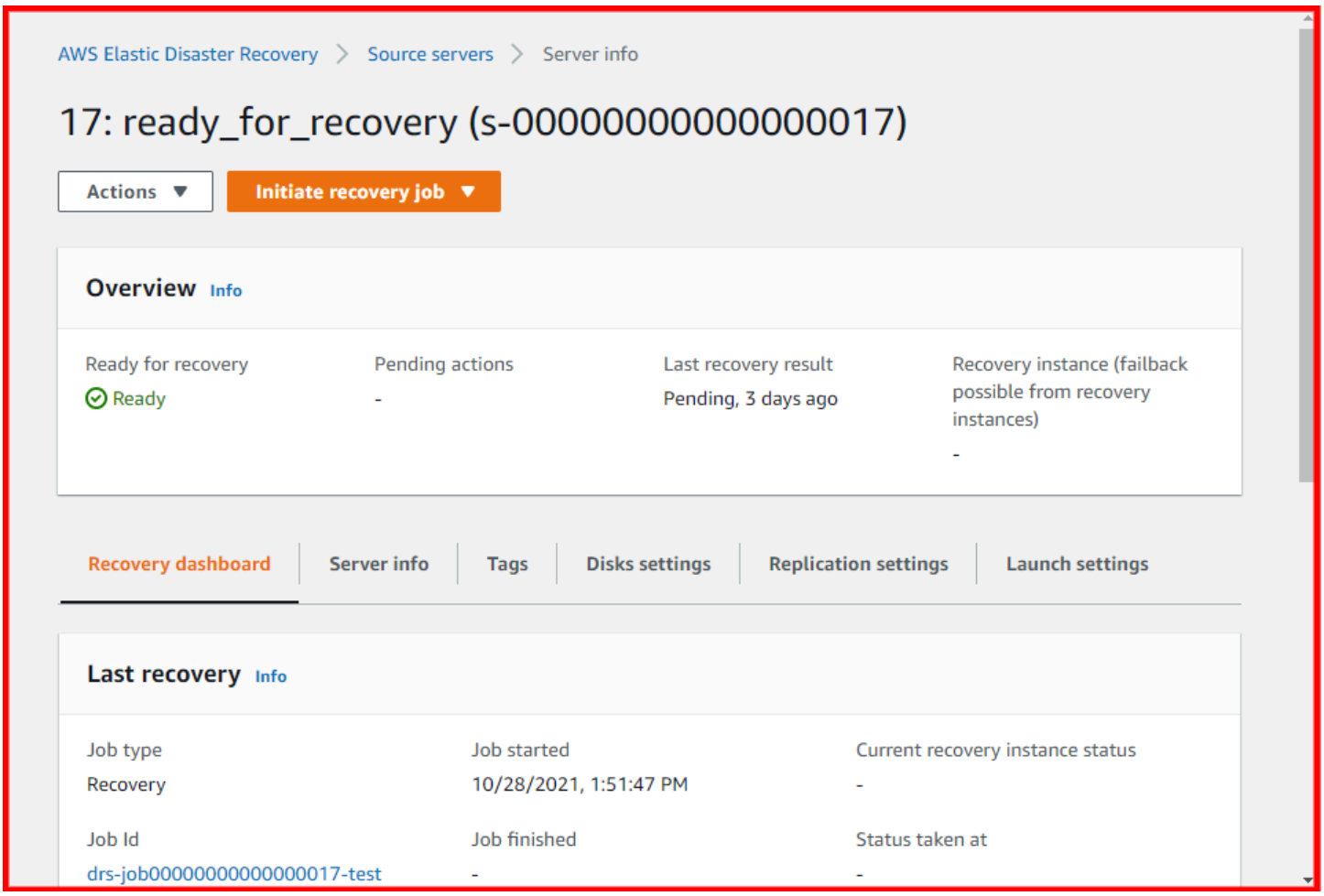
To access the server details view, click the **Hostname** of any server on the **Source servers** page.



You can also access the server details view by checking the box to the left of any single source server on the **Source servers** page and choosing **Actions > View server details**.

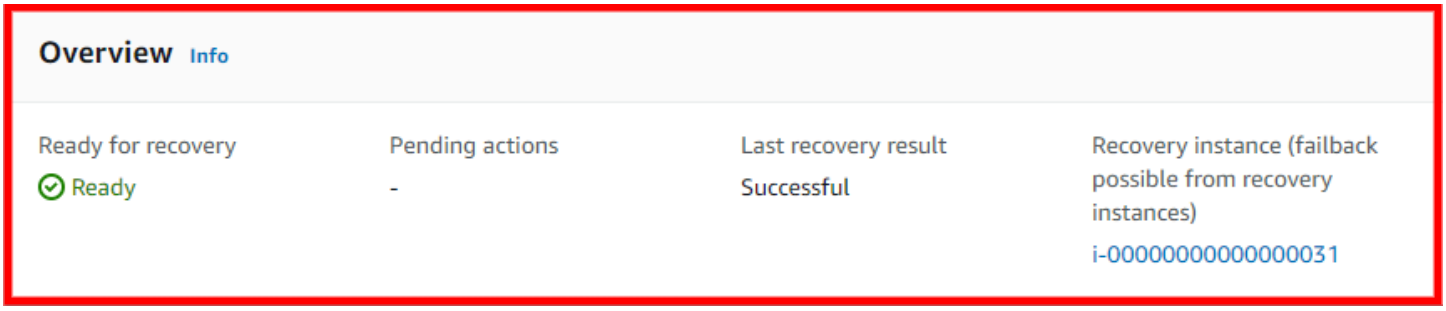


The server details view shows information and options for an individual server. Here, you can fully control and monitor the individual server.



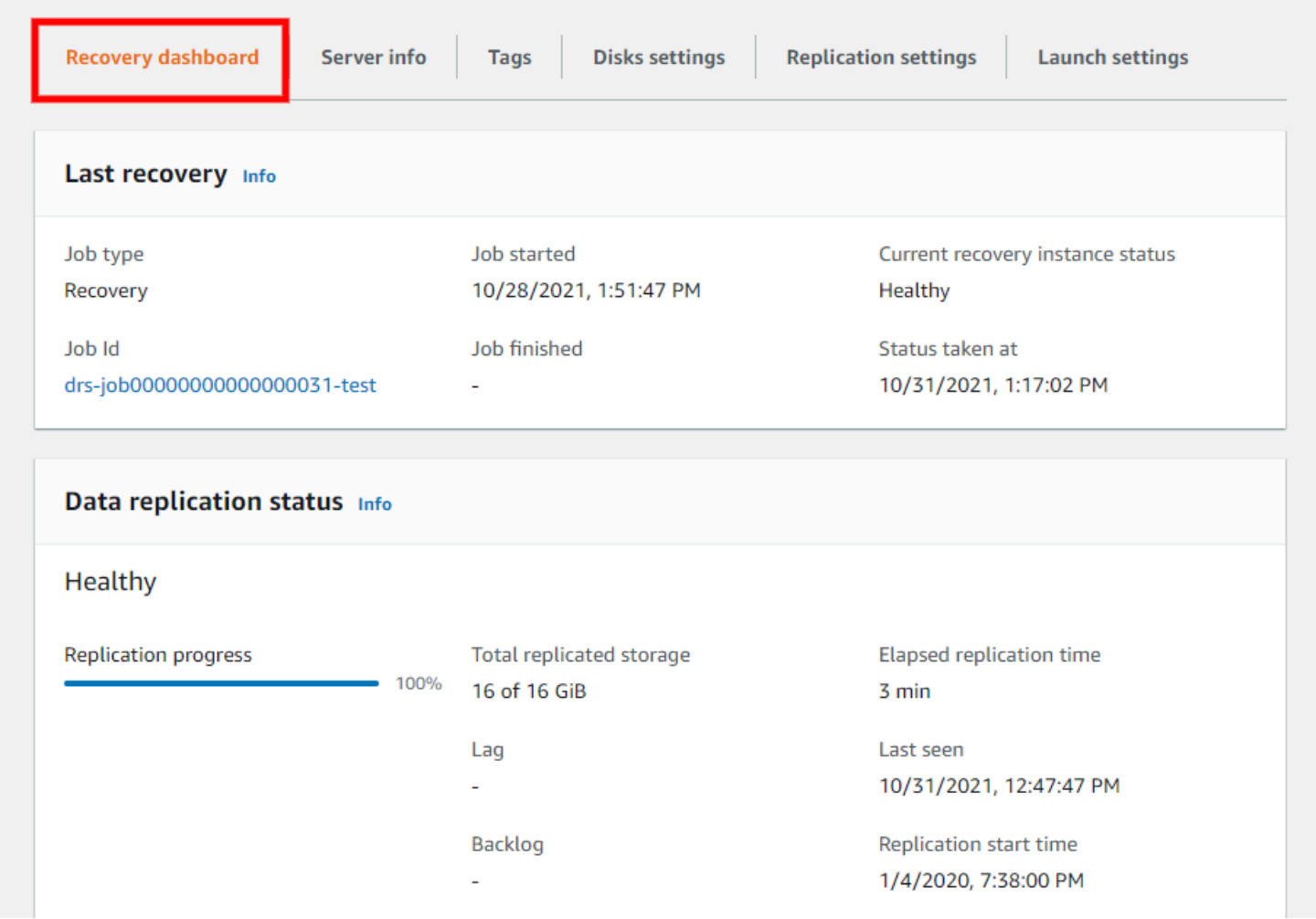
You can also perform a variety of actions, control replication, and launch Recovery instances for the individual server from the server details view.

The **Overview** box provides a basic overview of the server's status, including whether the server is ready for recovery, any pending actions, the last recovery result (if any), and a link to the Recovery instance (if one was launched for the server).



AWS DRS recovery dashboard

The **Recovery dashboard** tab allows you to monitor the server, its data replication status, and view events and metrics in CloudTrail.

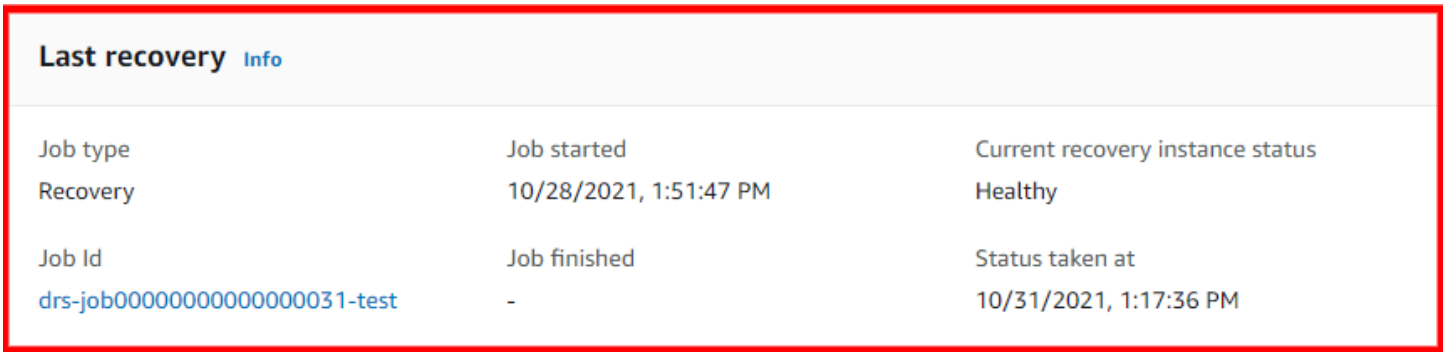


Topics

- Last recovery
- Data replication status
- Events and metrics
- Server actions and replication control

Last recovery

The **Last recovery** box provides an overview of the recovery process for the server.



Here, you can see the following:

- **Job type** – The type of recovery job performed (drill or recovery)
- **Job ID** – The ID of the last recovery job. Choose the **Job Id** to be redirected to the **Job** page for that specific recovery launch within the **Recovery job history**.
- **Job started** – The date and time the last recovery job was started.
- **Job finished** – The date and time the last recovery job was finished. This field is blank if the job is still ongoing.
- **Current recovery instance status** – The current status of the latest Recovery instance (if one has been launched).
- **Status taken at** – The last date and time the **current recovery instance status** was queried.

Data replication status

The **Data replication status** section provides an overview of the overall source server status, including:

- **Replication progress** – The percentage of the server's storage that was successfully replicated.
- **Rescan progress** – In the event of a rescan, the percentage of the server's storage that was rescanned.
- **Total replicated storage** – The total amount of storage replicated (in GiB).
- **Lag** – Whether the server is experiencing any lag. If it is - the lag time is indicated.
- **Backlog** – Whether there is any backlog on the server (in MiB)
- **Elapsed replication time** – Time elapsed since replication first began on the server.
- **Last seen** – The last time the server successfully connected to AWS Elastic Disaster Recovery.
- **Replication start time** – The date and time replication first began on the server.

Data replication can be in one of several states, as indicated in the panel title:

- **Initial sync**: initial copying of data from external servers is not done. Progress bar and **Total replicated storage** fields indicate how far along the process is.
- **Healthy**: all data has been copied and any changes at source are continuously being replicated (data is flowing).
- **Rescan**: an event happened that forced the agent on the external server to rescan all blocks on all replicated disks. This is the same as an initial sync but is faster because only changed blocks need to be copied; a rescan progress bar appears.
- **Stalled**: data is not flowing and user intervention is required (either initial sync never completes, or the state at the source becomes increasingly different from the state at AWS). When the state is stalled, then the replication initiation checklist is also shown, indicating where the error occurred that caused the stalled state.

This panel also shows:

- **Total replicated storage:** size of all disks being replicated for this source server, and how much has been copied to AWS (once initial sync is complete)

  Lag: if you launch a recovery instance now, how far behind it is from the state at the source. Normally this should be none.

  Backlog: how much data has been written at source but has not yet been copied to AWS. Normally this should be none.

  Last seen: when is the last time the AWS Replication Agent communicated with the AWS DRS service or the replication server.

If everything is working as it should and replication has finished initializing, the Data replication progress section displays a **Healthy** status.

If there are initialization, replication, or connectivity errors, the **Data replication status** section displays the cause of the issue, for example, a stall. If the error occurred during the initialization process, then the exact step during which the error occurred is marked with a red "x" under **Replication initiation steps**.

Events and metrics

You can review AWS Elastic Disaster Recovery events and metrics in AWS CloudTrail. Choose **View CloudTrail event history** to open AWS CloudTrail in a new tab. Learn more about AWS CloudTrail events in the AWS CloudTrail user guide.

Server actions and replication control

You can perform a variety of actions, control data replication, and manage your recovery and drill instances for an individual server from the server details view.

Topics

- Actions menu
- Initiate recovery job menu
- Alerts and errors

Actions menu

The **Actions** menu allows you to perform the following actions:

- **Add servers** – Choosing this option redirects you to the AWS Replication Agent installation instructions.
- **Edit replication settings** – Choose this option to edit the replication settings for the selected server or group of servers on the **Edit replication settings** tab.
- **Edit launch settings** – Choose this option to enter the source server's **Server details view > Launch settings** tab.
- **View server details** – Choose this option to enter the source server's **Server details view**.
- **Disconnect from AWS** – Choose this option to disconnect the selected server from AWS Elastic Disaster Recovery and AWS.

  On the **Disconnect X server/s from service** dialog, choose **Disconnect**.

  > **Important**
  >
  > This uninstalls the AWS Replication Agent from the source server, and data replication stops for the source server. This action does not affect any drill or recovery instances that

have been launched for this source server, but you are no longer able to identify which source servers your Amazon EC2 instances correspond to.

- **Delete server** – Choose this option to permanently delete a source server from AWS Elastic Disaster Recovery. This removes all information related to the server from the AWS Elastic Disaster Recovery service. You can only delete servers that have been disconnected from AWS. You need to reinstall the AWS Replication Agent on a deleted source server to add it back to AWS Elastic Disaster Recovery.

  When the **Delete X servers** dialog appears, click **Permanently delete**.

Initiate recovery job menu

The Initiate recovery job menu allows you to start drills and recoveries by launching drill and recovery instances as part of the overall failback process. You can learn more about the entire failback and failover process with AWS Elastic Disaster Recovery in the Performing a failback and failover with AWS Elastic Disaster Recovery documentation.

- **Initiate drill** – Choose the **Initiate drill** option to launch a drill instance for this server or group of servers for the purpose of testing your recovery solution. You should perform periodic drills in order to ensure that you are ready for recovery. Learn more about launching drill instances in AWS Elastic Disaster Recovery.
- **Initiate recovery** – Choose the **Initiate recovery** option to launch a recovery instance for this server or group of servers for the purpose of recovering the server in the event of a disaster. Learn more about launching recovery instances in AWS Elastic Disaster Recovery.

Alerts and errors

You can distinguish between healthy servers and servers that are experiencing issues on the **Recovery dashboard** in several ways. The AWS Elastic Disaster Recovery console is color-coded for ease of use.

- Healthy servers with no errors are characterized by the color blue. The **Data replication status** boxes displays steps and information in blue if the server is healthy.
- Servers that are experiencing temporary issues are characterized by the color yellow. This can include issues such as lag or a rescan. These issues do not break replication, but may delay replication or indicate a bigger problem.

- Servers that are experiencing serious issues are characterized by the color red. These issues can include a loss of connection, a stall, or other issues. You have to fix these issues in order for data replication to resume.

  The **Data replication status** box includes details of the issue.

  If the stall occurred during initiation, scroll down to **Replication initiation steps**. The step where the issue arose is marked with a red "x".

AWS DRS server info

The **Server info** tab shows general server information, hardware, and network information:

- **General information**
  - **Last updated**: when was the data in this tab updated.
  - **Date added**: when was this server added to the service.
  - **AWS ID**: the ID of this source server resource.
  - **arn**: the AWS Resource Name for this source server.
- **Identification hint**s: under most circumstances, the hostname is the best identifier, as it is what is used throughout the console as the name of the source server. If you need to validate which external server this is referring to in your data center, you can use one of the additional fields: Fully qualified domain name, VMware virtual machine identifier (only if source is VMware), AWS instance ID (only if source is running on AWS).
- **Hardware and operating system**: the CPUs, RAM, disks, and network interfaces on the external server, as well as the type and full name of the operating system running on that server. The disks shown are all the disks on the source server, and may include disks not being replicated.
- **Recommended instance type**: this is the EC2 instance type the service is auto-recommending to use for the launched recovery instance. This is based only on the CPUs and RAM at the source (and not on utilization information). This is the instance type that is launched for this server by default. For `arm64` source servers, the recommendation is an AWS Graviton instance type. If your default launch settings use **Active (basic)**, Elastic Disaster Recovery applies **Active (in-aws)** to new `arm64` source servers; you cannot select **Active (basic)** for an `arm64` source server. With **Active (in-aws)**, Elastic Disaster Recovery launches the source EC2 instance type when the recovery Region offers it. Otherwise, Elastic Disaster Recovery uses the recommended instance type.

Information shown includes:

- **Last updated**
- **Date added**
- **AWS ID** (if relevant)
- **Hostname**
- **Fully qualified domain name**
- **VMware virtual machine identifier** (if relevant)
- **AWS instance ID**
- **AWS ID**
- **CPUs**
- **RAM**
- **Disks**
- **Network interfaces**
- **Operating system** information
- **Recommended instance type**

Managing tags with AWS DRS

The Tags section shows any tags that have been assigned to the server. A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs. Learn more about AWS tags in this Amazon EC2 article.

> **Important**
>
> **Do not** alter the **Name** tag of resources created by AWS DRS (replication servers, EBS volumes, EBS snapshots, Conversion servers).

Choose **Manage tags** to open the **Manage tags** page to add or remove tags.

- Choose **Add new tag** to add a new tag. Add a tag **Key** and an optional tag **Value**. Choose **Save** to save your added tags.

- To remove a tag, choose **Remove** to the right of the tag you want to remove, and then choose **Save**.

AWS DRS disk settings

The **Disk settings** tab shows a list of all of the disks on the source server and information for each disk:

- **Disk name**
- **Staging disk type** – The corresponding Amazon EBS volume disk type that is being used for the disk.
- **Replicated storage** – The amount of storage that has been replicated from the disk to the Replication Server.
- **Total storage** – The total storage capacity of the disk.
- **Status** – shows the status of each disk, values can be either **Normal**, **Normal with marketplace license**, **Error** (with error description). Normal with marketplace license means that the server has at least one marketplace license associated with this volume. Volumes with marketplace licenses pose some limitations on launch: the target region and the selected instance type must support this license. If launching into a different account, the marketplace product must be subscribed to in that account as well or the launch fails. The state is set to Error if there is a problem with the volume, such as not having permissions to read the marketplace license details if the server is owned by a different AWS account. The value can also be empty if the status is not known at this time.

Change staging disk type

You can change the EBS volume disk type for each disk or for a group of disks. To change the EBS volume disk type:

1. Select the circle to the left of each disk name and choose **Change staging disk type**.
2. On the **Change staging disk type** dialog, select the type of EBS volume to use for the disk or group of disks.
3. Select the **AUTO** option if your volume's size is greater than 125 GiB and you want AWS Elastic Disaster Recovery to automatically select the most cost-effective EBS volume disk type for each disk based on the disk size and type based on the option you defined in the **Replication**

settings (either the default **Lower cost, Throughput Optimized HDD (st1)** option or the **Faster, General Purpose SSD (gp2) or (gp3)** option).

AWS Elastic Disaster Recovery uses a single Replication Server per 15 source disks. Selecting the **Auto** option ensures that the fewest number of replication servers are used, resulting in increased cost savings.

> **Note**
>
> AWS Elastic Disaster Recovery always uses EBS magnetic volumes for disks that are under 125 GiB in size when you choose the **Auto** option.

If you do not want AWS Elastic Disaster Recovery to automatically select a disk, you can select a disk manually. Select the disk type from the **EBS volume type** menu.

For certain disks, you can configure the amount of IOPS to be allocated per GB of disk space under **IOPS**. You can allocate up to 50 IOPS per GB. 64,000 IOPS are available for Nitro-based instances. Other instances are guaranteed up to 32,000 IOPS. The maximum IOPS per instance is 80,000.

Choose **Change** to confirm the change.

AWS DRS replication settings

The **Replication settings** tab allows you to edit the replication settings for an individual source server. After the source server is added to AWS Elastic Disaster Recovery, the replication settings that are defined in the Replication Settings template are automatically applied to the server. You can later edit them for a single source server or multiple source servers through the **Replication settings** tab.

Edit each setting as required and then choose **Save replication settings**.

Learn more about replication settings.

AWS DRS launch settings

The launch settings are a set of instructions that comprise an EC2 launch template and other settings, which determine how a recovery instance is launched for each source server on AWS.

Launch settings, including the EC2 launch template, are automatically created every time you add a server to AWS Elastic Disaster Recovery.

The launch settings can be modified at any time, including before the source servers have even completed initial sync.

Learn more about individual launch settings.

Important

If the source server's instance type includes instance store, please consider the following:

- It is **not** recommended to change the instance type of an instance to a type that has no ephemeral volumes, or has a different number of ephemeral volumes, as such changes could lead to data inconsistencies and may even cause recovery, drill, or failback to fail.

Post-launch settings

Post-launch settings allow you to control and automate actions performed after a recovery instance has been launched for the source server in AWS. These settings are created automatically based on the **Default post-launch actions**.

Activating the post-launch actions for a specific source server:

- Navigate to the **Source servers** page and select a source server.
- Go to the **Post-launch settings** tab. If **Post launch action settings** has **Post launch actions** set to **Active**, click **Edit** for **Post launch action settings**.
- You will be redirected to the **Edit post-launch settings** screen. Make sure the **Post-launch actions active** option is not checked and click **Save**.

Alternatively, you can activate and deactivate post-launch actions for multiple servers by navigating to the **Source servers** page, selecting the servers you want to update and clicking **Actions > Edit post-launch action settings**. To activate, make sure the **Post-launch actions active** option is checked, and to deactivate, it should be unchecked. If you made a change, click **Save**.

Topics

- Adding custom actions with AWS DRS
- Activating, deactivating, and editing predefined or custom actions
- Deleting custom actions
- Predefined post-launch actions

Adding custom actions with AWS DRS

AWS Elastic Disaster Recovery (AWS DRS) allows you to run any SSM document that you like – public SSM documents, SSM documents that you created and uploaded to your account or SSM documents that are shared with you. You can configure a custom action to run any SSM document that is available in your account. To be able to create, edit or delete a custom action, make sure the post-launch actions are activated for this source server. Custom actions added to the default settings are automatically added to newly added source servers.

Create a custom action

Adding a custom action through source server's **Post-launch settings**, adds it to this source server. To add a custom action to all newly added source servers, use the **Settings** → **Default post-launch actions** page. To add a new custom action to the source server, go to **Source server details** → **Post-launch settings** tab. If the **Post-launch actions** post-launch actions settings is **Active**, you can create new custom actions by clicking on the **Add action** button.

The **Add action** page includes these parameters:

- **Action name** – The name of the action in AWS DRS, which should be intuitive, meaningful and unique in this AWS account and region.
- **Activate this action** – Use this checkbox to activate or deactivate the custom action for this source server. Only active actions run after the launch of a recovery instance.
- **Mark launch as successful only if this action finishes running successfully** – This checkbox dictates whether or not the launch is marked as successful, based on the successful run of this action. Instance launches progress normally regardless of the success of the action.
- **Systems Manager document name** – Select any Systems Manager document that is available to be used in this account.
- **View in Systems Manager** – Click to open **Systems Manager** and view additional information about the document.
- **Description** – Add a description or keep the default.

- **Document version** – Select which SSM document version to run. AWS DRS can run a default version, the latest version, or a specific version, according to your preferences.
- **Category** – Select from various available categories including monitoring, validation, security and more.
- **Order** – Specify the order in which the actions are executed. The lower the number, the earlier the action is executed. Values allowed are between 2 and 10,000. The numbers must be unique but don't need to be consecutive.
- **Platform** – Taken from the SSM document and reports which Operating System platform (Windows/Linux) is supported by the action.
- **Creator** – Who created the action. For custom actions, the default is always **This account**.

The **Action parameters** change according to the specific SSM document that is selected. Note that for the instance ID parameter, you can choose to use the launch instance ID, in which case, AWS DRS dynamically populates the value.

Note

AWS Elastic Disaster Recovery (AWS DRS) places **AWSElasticDisasterRecoveryRecoveryInstanceWithLaunchActionsRole** instance profile on the launch instance if post-launch actions is active for the source server. If you add an SSM command action that requires additional permissions in the launch instance, you must ensure that the instance profile has the right policies or the right permissions. In order to do so, create a role that has the required permissions as per the policies above or has a policy or policies with those permissions attached to it. Go to **Launch settings** > **EC2 launch template** > **Modify** > **Advanced** > **IAM instance profile**. Use an existing profile or create a new one using the **Create new IAM profile** link.

Note

Only trusted, authorized users should have access to the parameter store. For enhanced security, ensure that users who do not have permissions to execute SSM documents / commands, do not have access to parameter store. Learn more about restricting access to Systems Manager parameters. Action parameters are stored in the SSM parameter store as regular strings. Changing parameters in the SSM Parameter store may impact the post launch action run on target instances. We recommend to consider security implications,

when choosing to use parameters that contain scripts or sensitive information, such as API keys and database passwords.

Activating, deactivating, and editing predefined or custom actions

You can activate, deactivate and edit actions available for this source server. Activating an action ensures it runs after launching a recovery instance. Likewise, deactivating it prevents it from being run after launching a recovery instance. The default settings are not affected by activating, deactivating or editing an action for a source server. Editing an action for a source server updates it for that source server. These changes are not reflected on the action, if it exists in the default post-launch actions settings. Changes to actions in the default settings, so as to apply to newly added source servers, can be done from the **Settings** → **Default post-launch actions** page.

To be able to activate, create, deactivate, edit, or delete a custom action and to activate, deactivate or edit predefined actions for a source server, make sure the post-launch actions are activated for that source server.

Activating, deactivating and editing predefined or custom actions

To activate, deactivate or edit a post launch action in the default post-launch actions settings, go to **Source server details** page, and visit the **Post-launch settings** tab. If **Post-launch actions settings** shows **Post-launch actions** to be **Active**, you can edit any action defined for the source server.

Locate the action you want to edit in the **Actions** card view, or use the search field to filter the actions by name.

Choose the action's card to select it, and then choose the **Edit** button.

To activate the action, make sure the **Activate this action setting** is checked and click the **Save** button. To deactivate, make sure the **Activate this action** setting is un-checked and click the **Save** button.

The edit page allows you to change the value of some of the parameters for both pre-defined actions and custom actions. Some parameters can only be edited if the action is a custom action. See below for specific information.

The parameters that appear on the edit page:

- **Action name** – Editable for custom actions. The name of the action in AWS DRS, which should be intuitive, meaningful and unique in this AWS account and region.
- **Activate this action** – Use this checkbox to activate or deactivate the action for this source server. Only active actions run after the launch of a recovery instance.
- **Mark launch as successful only if this action finishes running successfully** – This checkbox dictates whether or not the launch is marked as successful, based on the successful run of this action. Instance launches progress normally regardless of the success of the action.
- **Systems Manager document name** – Editable for custom actions. Select any Systems Manager document that is available to be used in this account.
- **View in Systems Manager** – Click to open **Systems Manager** and view additional information about the document.
- **Description** – Editable for custom actions. Add a description or keep the default.
- **Document version** – Editable for custom actions. Select which SSM document version to run. AWS DRS can run a default version, the latest version, or a specific version, according to your preferences.
- **Category** – Editable for custom actions. Select from various available categories including monitoring, validation, security and more.
- **Order** – Specify the order in which the actions run. The lower the number, the earlier the action runs. Values allowed are between 2 and 10,000. The numbers must be unique but don't need to be consecutive.
- **Platform** – Not editable. Taken from the SSM document and reports which Operating System platform (Windows/Linux) is supported by the action.
- **Creator** – Not editable. Who created the action. For custom actions, the default is always **This account**.

The **Action parameters** change according to the specific SSM document that is selected. Note that for the instance ID parameter, you can choose to use the launch instance ID, in which case, AWS DRS dynamically populates the value. Some predefined actions, where applicable allow to use a dynamically populated value for the volumes. This value is dynamically populated by AWS DRS with the volumes of the instance being launched.

After making the required changes, click **Save**, to save the changes and **Cancel** to abort them.

Deleting custom actions

Custom actions added to a source server from the default settings on creation or created later for that source server can also be deleted. Deleting a custom action for a source server removes it from that source server and means the action is no longer available to that source server. Deleting the action for a source server does not remove it from the default settings if the action was defined there as well. To delete a custom action from the default settings to avoid adding it to newly added source servers, go to the **Settings** → **Default post-launch actions** page, and delete the action from there. Pre-defined actions cannot be deleted. If a pre-defined action is not required, it can be deactivated.

Locate the action you want to delete in the **Actions** card view, or use the search field to filter the actions by name. Select the action, and click the **Delete** button. To confirm, press **Delete**.

Predefined post-launch actions

AWS Elastic Disaster Recovery allows you to run various predefined post-launch actions on your EC2 launched instance. Use these out-of-the-box actions to improve your launch flexibility.

These actions can be activated, edited or deactivated for a specific source server.

List of available predefined actions

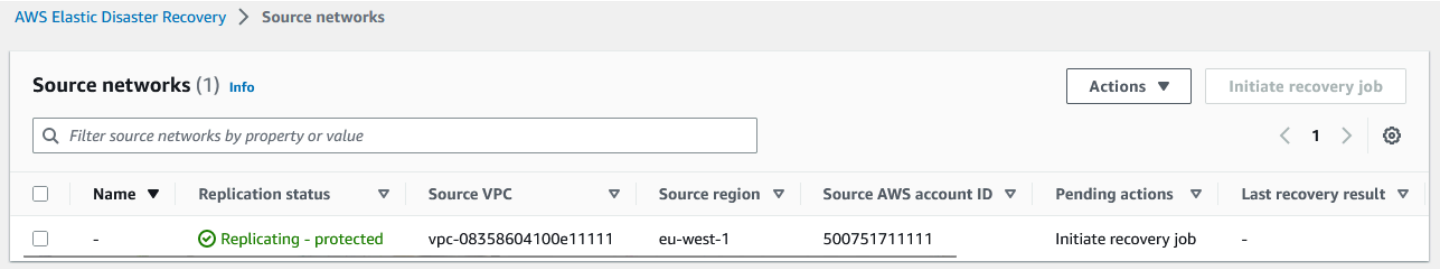
Source networks

The network replication feature allows you to keep track of network changes and perform quick updates. The feature helps prevent configuration mismatch during recovery, saves time and resources and provides enhanced security. For example, when a security group is updated, this change will be automatically replicated, ensuring compliance and preventing potential security risks. In addition, recovery instances will be launched within the recovered source networks automatically, preventing the need to configure each server manually.

> **Important**
>
> Only in-AWS networks can be replicated.

AWS DRS source network page

The **Source networks** page automatically presents all of the available source networks. This page allows you to manage your source networks, view their specifications, and perform updates.



Each row represents a specific network. It includes various network parameters including:

- Name – the selected source network name
- Replication status – options include **Replicating - protected**, **Stopped**, **In progress**, and **Error**
- Source region – the AWS Region of the source network
- Source AWS account ID – the AWS account ID of the source network
- Pending actions – the next step in the source network replication workflow
- Last recovery result – **Not started**, **Pending**, **Successful**, **Failed**, and **Partial success** (meaning the network was deployed, but the source servers were not configured as part of the recovered network)

- Launched VPC – the recovered network
- CFN stack name – the name of the CloudFormation stack which was used to deploy the launched VPC
- Source network ID – the ID of the source network

Use the top navigation to select an S3 bucket, which is required to enable recovery or to initiate a recovery job.

Use the **Actions** menu to perform various actions including:

- Start replication – Use this option if you want to start replicating your network configuration.
- Stop replication – Use this option if you want to stop replicating your network configuration.
- Export CloudFormation (CFN) template – This option allows you to export the CloudFormation template to your selected S3 bucket. This allows you to verify that the configurations match your preferences and conduct security checks.

> **Note**
>
> If you choose to make changes to the CloudFormation template, it cannot be reuploaded to AWS Elastic Disaster Recovery.

- Manage tags – This option will open the **Manage tags** page which allows you to add or remove tags from your selected network resource.
- Select S3 bucket – This option allows you to save network CFN stacks in your account's Amazon S3 bucket. You must specify the S3 bucket before you initiate network replication. It is recommended that you employ security best practices for Amazon S3.

Adding source networks to Elastic Disaster Recovery

Available source networks are presented automatically on the **Source networks** page, along with their details: replication status, pending action, CloudFormation stack name, and more.

When adding a source server to AWS Elastic Disaster Recovery, and after an agent is installed, the VPC network will be automatically identified and created.

To replicate and recover your network configurations, take the following steps:

1. Install the AWS Replication agent on your source servers. Alternatively, source networks can be added manually by calling the CreateSourceNetwork API.
2. Create the required role.
3. Select the relevant network.
4. Start replication.
5. Select an S3 bucket.

> **Important**
>
> You only need to configure your S3 bucket once. Configurations will apply to all existing and newly added source networks.

6. Test or recover your network configurations by initiating a recovery job. This will include creating or updating your CloudFormation stack.

Installing the AWS Replication Agent

In order to use the network replication feature, you must first install the AWS Replication Agent on each source server that you want to add to AWS Elastic Disaster Recovery.

Linux installation instructions

Windows installation instructions

Creating the required role for Elastic Disaster Recovery

In order to replicate network configurations between different accounts, you need to go to the source account and create the **Network role** from the **Trusted accounts** page. This will automatically create the role and attach the required policies.

> **Note**
>
> This is only required if your target account is different from the source account.

To create the required role, take the following steps:

1. Go to your source account.

2. Go to the **Trusted accounts** page.
3. Click **Add trusted accounts and create roles**.
4. Click **Add new trusted account**.
5. Enter the target account ID and choose **Network role**.
6. Click **Add trusted accounts and roles**. A success message will appear at the top of the screen.

This action will create the DRSSourceNetworkRole role that is required to utilize the feature.

This role includes the AWSElasticDisasterRecoverySourceNetworkPolicy policy and the following trust policy permissions:

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
  {
  "Effect": "Allow",
  "Principal": {
  "Service": "drs.amazonaws.com"
  },
  "Action": "sts:AssumeRole",
  "Condition": {
  "StringLike": {
  "aws:SourceAccount": "{{target_account}}"
  },
  "ArnLike": {
  "aws:SourceArn": "arn:aws:drs:*:*:source-network/*"
  }
  }
  }
  ]
  }
```

After you install the agent and create the relevant role, you can start replicating your network configurations.

Replicating your network configurations in Elastic Disaster Recovery

Once you install your agent and create the required role, go to the **Source networks** page and take the following steps:

1. Select the network you want to replicate from the list.
2. Click **Actions** and select **Start replication** from the drop-down menu.
3. Click **Select S3 bucket**. This will allow you to save the CloudFormation stack in your account's S3 bucket. You must specify the S3 bucket before you initiate network recovery. It is recommended that you employ S3 bucket security and access management policies.

   You can choose between selecting an existing S3 bucket and creating a new bucket using the S3 bucket console.

   > **Note**
   >
   > You must enable S3 versioning.

4. To test or recover your network configurations, click **Initiate recovery job** and the **Initiate recovery job** prompt will appear.

   If this is the first time you are replicating network configurations, you will need to create a new stack.

   If you already created a stack, you can choose between 3 options:

   a. **Update a recommended stack** – The recommended stack is always the last stack you used.

      > **Note**
      >
      > If the update is not successful, simply create a new stack.

   b. **Create new stack**

   c. **Use a previously created stack** – if you want to choose a stack that you have previously used, select your preferred stack from the drop-down. This will only update the launch templates. The selected stack will then become the recommended stack, allowing you to update it.

Once the recovery job is marked as **Successful**, the network (VPC) is launched in the target Region. All the EC2 launch templates of the source servers in the relevant network will be automatically

updated and will feature the new values. This means that when you perform a recovery, those source servers will be launched as part of the new network and the correct subnet.

Trusted accounts

Trusted accounts provide enhanced account management capabilities and visibility, including the ability to easily create multiple IAM roles for different users. Use this feature to quickly add the roles you need to use various AWS Elastic Disaster Recovery features and see the permissions of different accounts from a single screen.

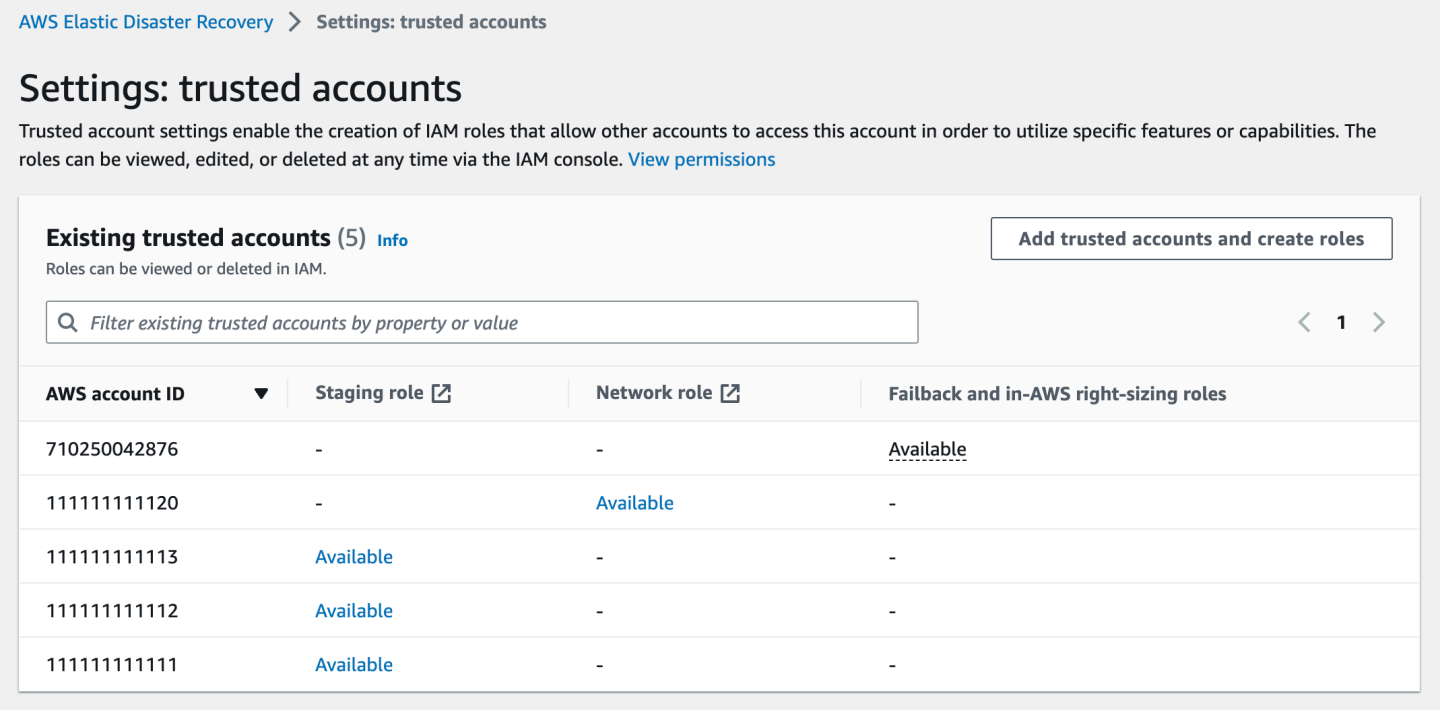
Roles created via CloudFormation (Failback and in-AWS right-sizing roles), should be deleted from the CloudFormation console.

AWS DRS trusted account page

The **Trusted accounts** page allows you to automatically create IAM roles that are required in order to utilize specific features and capabilities.

This page provides visibility into the existing roles assigned to each trusted account.

To edit or delete these roles, go to the IAM console. Deleting the IAM role will automatically remove the trusted account from the AWS Elastic Disaster Recovery console.



> **Note**
>
> Commercial AWS accounts can only be trusted to other Commercial AWS accounts and GovCloud AWS accounts can only be trusted to other GovCloud AWS accounts.

Adding a trusted account in AWS DRS

To add a trusted account, take the following steps:

1. Click **Add trusted accounts and create roles**.
2. Click **Add new trusted account**.
3. Enter an account ID and choose the relevant role or roles. There are 3 available options: Staging role, Network role, and Failback and in-AWS right-sizing roles.
4. Click **Add trusted accounts and roles**. A success message will appear at the top of the screen.

> **Note**
>
> Up to 10 accounts can be added in a single batch and up to 100 accounts for a single AWS DRS account.

Creating the Staging role

The **Staging role** is required to utilize various AWS Elastic Disaster Recovery capabilities, including the multi-account feature. To automatically create the role and the attached required policies, simply create it for a specific account via the **Trusted accounts** page.

This action will create the DRSStagingAccountRole role which includes the AWSElasticDisasterRecoveryStagingAccountPolicy_v2 policy and the following trust policy permissions:

JSON

```
{
  "Version":"2012-10-17",
```

```
"Statement": [
{
"Effect": "Allow",
"Principal": {
"Service": "drs.amazonaws.com"
},
"Action": [
"sts:AssumeRole",
"sts:SetSourceIdentity"
],
"Condition": {
"StringLike": {
"sts:SourceIdentity": "{{target_account}}",
"aws:SourceAccount": "{{target_account}}"
},
"ArnLike": {
"aws:SourceArn": "arn:aws:drs:*:*:source-server/*"
}
}
}
]
}
```

Creating the Network role

The **Network role** is required to utilize various AWS Elastic Disaster Recovery capabilities, including the network replication feature. To automatically create the role and the attached required policies, simply create it for a specific account via the **Trusted accounts** page.

This action will create the DRSSourceNetworkRole role which includes the AWSElasticDisasterRecoverySourceNetworkPolicy policy and the following trust policy permissions:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
    {
    "Effect": "Allow",
    "Principal": {
```

```
"Service": "drs.amazonaws.com"
},
"Action": "sts:AssumeRole",
"Condition": {
"StringLike": {
"aws:SourceAccount": "{{target_account}}"
},
"ArnLike": {
"aws:SourceArn": "arn:aws:drs:*:*:source-network/*"
}
}
}
]
}
```

Creating the Failback and in-AWS right-sizing roles

The **Failback and in-AWS right-sizing roles** are required to utilize various AWS Elastic Disaster Recovery capabilities, including cross account failback and in-AWS features. Each Trusted AWS Account will need a set of these IAM roles for functionality. You can automatically create these roles, and their attached policies, via the **Trusted accounts** section of the AWS Elastic Disaster Recovery console. The roles required are:

1. **DRSCrossAccountReplicationRole**
2. **DRSCrossAccountAgentRole**
3. **DRSCrossAccountAgentAuthorizedRole**

If you intend to create these roles manually, please ensure they are placed in the `service-role` path, with the Role name ending in an underscore and the trusted Account ID, as specified below:

```
arn:aws:iam::account-id:role/service-role/
DRSCrossAccountReplicationRole_trustedAccountID
```

DRSCrossAccountReplicationRole

The **DRSCrossAccountReplicationRole** contains the following trust policy. If you plan to use the policy as a template, replace the `account-id` with the Trusted AWS Account ID.

The **DRSCrossAccountReplicationRole** has the AWS Managed Policy **AWSElasticDisasterRecoveryCrossAccountReplicationPolicy** attached.

DRSCrossAccountAgentRole

The **DRSCrossAccountAgentRole** contains the following trust policy. If you plan to use the policy as a template, replace the `trustedAccount` with the Trusted AWS Account ID, and replace `sourceAccount` with the source AWS Account ID.

The **DRSCrossAccountAgentRole** has the AWS Managed Policy **AWSElasticDisasterRecoveryEc2InstancePolicy** attached.

DRSCrossAccountAgentAuthorizedRole

The **DRSCrossAccountAgentAuthorizedRole** contains the following trust policy. If you plan to use the policy as a template, replace the `account-id` with the Trusted AWS Account ID.

The **DRSCrossAccountAgentAuthorizedRole** has the following inline policy attached. If you plan to use the policy as a template, replace the `trustedAccount` with the Trusted AWS Account ID, and replace `sourceAccount` with the source AWS Account ID.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Action": [
                "sts:AssumeRole",
                "sts:TagSession"
            ],
            "Resource": "arn:aws:iam::111122223333:role/service-role/
DRSCrossAccountAgentRole_sourceAccount",
            "Effect": "Allow"
        },
        {
            "Condition": {
                "StringLike": {
                    "sts:SourceIdentity": "i-*"
                }
            },
```

```
            "Action": [
                "sts:SetSourceIdentity"
            ],
            "Resource": "arn:aws:iam::111122223333:role/service-role/
DRSCrossAccountAgentRole_sourceAccount",
            "Effect": "Allow"
        }
    ]
}
```

Configuring launch settings in AWS Elastic Disaster Recovery

Launch settings determine how your drill and recovery instances are launched in AWS. They are composed of DRS launch settings and an EC2 launch template, allowing you to fully customize your drill and recovery instances by configuring key metrics, such as the subnet within which the instance will be launched, the instance type to be used, license transfers, replication status, and a variety of other settings. AWS Elastic Disaster Recovery ensures that your drill and recovery instances constantly abide by the latest AWS security, instance, and other updates by utilizing EC2 launch templates. EC2 launch templates always use the latest EC2 instance and technology. EC2 launch templates integrate with AWS Elastic Disaster Recovery in order to give you full control over every single setting within your drill and recovery instance.

Preparing for drill and recovery instance launch

Prior to launching your instances, make sure that your environment is set up properly to ensure successful launches. Check the following prior to continuing:

- Prepare your subnets for launch – Plan which subnets you will use to launch your drill and recovery instances. You will use these subnets in your EC2 launch template when you configure launch settings.
- Create security groups within the subnets – Create the security groups you want to use within your prepared subnets. You will set these security groups in your EC2 launch template when you configure launch settings.

> **Note**
>
> If you want to run a proof of concept, you can skip this step. AWS Elastic Disaster Recovery will automatically use the default subnet and security groups. Ensure that you have not deleted your default subnet.

Important

When launching a drill, recovery, or an in-AWS failback, you can launch up to 100 source servers in a single operation. Additional source servers can be launched in subsequent operations.

Launch settings

Launch settings are a set of instructions that are comprised of two sections: DRS launch settings and the EC2 launch template, that determine how a drill or recovery instance is launched for each source server in AWS.

Launch settings, including the EC2 launch template, are automatically created each time you add a source server to AWS Elastic Disaster Recovery.

You can modify the launch settings at any time, including before the source server has completed its initial sync.

Note

- Any changes made to the launch settings only affect newly launched drill and recovery instances.
- For many customers, there is no need to modify the DRS launch settings or the EC2 launch template in order to launch drill or recovery instances.

You can change launch settings for a single server or for multiple servers in the AWS DRS console. This allows you to make changes to multiple servers at once. You can also modify launch settings for multiple servers via the AWS Elastic Disaster Recovery API.

To access the launch settings of a specific source server, go to the **Source servers** page and click the server's hostname. In the individual server view navigate to the **Launch settings** tab.

You can also access the launch settings of a single server by selecting a single source server on the **Source servers** page and choosing **Actions > Edit DRS launch settings** or **Actions > Edit EC2 launch template**.

The **Launch settings** tab is divided into two sections:

- DRS launch settings
- EC2 launch template

Learn more about Amazon EC2 launch templates in Amazon EC2 User Guide.

DRS launch settings

The **DRS launch settings** section allows you to control server-specific settings.

To edit these settings for a single server, take the following steps:

1. Go to the **Source servers** page.
2. Select a source server to update.
3. Under the **Actions** menu, select **Edit DRS launch settings** and you will be navigated to the **Edit DRS launch template** page within the AWS DRS console.
4. Change the settings according to your preferences.
5. Click **Save settings**.

Alternatively:

- Go to the **Source servers** page.
- Select a specific source server.
- Go to the **Launch settings** tab.
- Click **Edit** in the DRS launch settings section.

DRS launch settings parameters

The DRS launch settings include the following parameters:

- **Instance type right-sizing –** choose whether to allow AWS Elastic Disaster Recovery to launch a drill, recovery, or failback instance type that best matches the hardware configuration of the source server. If you activate this feature, any modification you make to the instance type in the EC2 launch template will be overwritten by the service.

- If you select the **Active (basic)** option, AWS Elastic Disaster Recovery will launch an AWS instance type that best matches the OS, CPU, and RAM of your source server. AWS Elastic Disaster Recovery will launch a new instance type after every change of configuration on the source server (for example, added/removed disks, added/removed RAM). Instance types are only chosen from the C5 family.

  > **Note**
  >
  > If your default launch settings use **Active (basic)**, Elastic Disaster Recovery applies **Active (in-aws)** to new `arm64` source servers. You cannot select **Active (basic)** for an `arm64` source server.

- If you select the **Active (in-aws)** option, AWS Elastic Disaster Recovery will periodically update the EC2 launch template based on the hardware configuration of the EC2 instance source server. For source servers replicated from another account, the failback and in-AWS right-sizing roles must exist in the source account. Otherwise, Elastic Disaster Recovery uses the recommended instance type. See Creating the Failback and in-AWS right-sizing roles.
- If you select **Inactive**, AWS Elastic Disaster Recovery will launch the AWS instance type as configured in your EC2 launch template. Select this option if you want to determine the instance type that will be launched in AWS for all your drill or recovery servers.

> **Important**
>
> - The AWS instance type selected by AWS Elastic Disaster Recovery when this feature is activated will overwrite the instance type defined in your EC2 launch template.
> - Hardware changes and the resulting AWS instance type change may take up to 90 minutes to be processed by AWS Elastic Disaster Recovery.

The right-sizing instance type selected by AWS Elastic Disaster Recovery will be featured on the **Server details** tab.

- **Start instance upon launching –** Choose whether you want to start your drill and recovery instances automatically upon launch or whether you want to launch them in a stopped state.

  If you choose **No**, you will have to start the drill or recovery AWS instance manually from the EC2 Console.

- **Copy Private IP –** Choose whether you want AWS Elastic Disaster Recovery to ensure that the private IP used by the drill or recovery instance matches the private IP used by the source server. AWS Elastic Disaster Recovery will monitor the source server on an hourly basis to identify the private IP and will use the private IP of the primary network interface.

  The **No** option is chosen by default. Choose **No** if you do not want the private IP of the drill or recovery instance to match that of the source machine.

  Choose **Yes** if you want to use a private IP. The IP will be shown in brackets next to the option.

  - If you choose **Yes**, ensure that the IP range of the subnet you set in the EC2 launch template includes the private IP address.
  - If both the source server and the drill or recovery instance share the same subnet through a VPN, then the source private IP is already in use, and the **Copy private IP** option should not be used.
  - Removing a private IP from a specific server's settings does not remove it from the launch template.

  Copy private IP will be deactivated if you set a value for **Launch into instance ID**, as this setting cannot affect an already launched instance.
- **Transfer server tags –** Choose whether you want AWS Elastic Disaster Recovery to transfer any user-configured custom tags from your source servers onto your drill or recovery instance. These tags are attached to all source servers, all launched drill and recovery instances, and all of the ephemeral resources that are created on your AWS Account during the normal operation of AWS Elastic Disaster Recovery. Transfer server tags only copies tags associated with the source servers in the AWS Elastic Disaster Recovery console, and does not copy the EC2 source server tags (in case of AWS to AWS disaster recovery implementation).

  These resources include:

  - EC2 instances
  - Conversion groups
  - Security groups
  - EBS volumes
  - Snapshots

> **Note**
>
> - AWS Elastic Disaster Recovery automatically adds system tags to all resources.
> - Tags that are added on the EC2 launch template will take precedence over tags that are transferred directly from the source server.

You can always add tags from the Amazon EC2 console as described in this Amazon EC2 article.

Transfer server tags is deactivated if you set a value for **Launch into instance ID**, because this setting cannot affect an already launched instance.

- **Launch into instance ID** - Configure an existing instance ID to launch into, instead of creating a new instance. This field allows you to select an EC2 instance from the list of EC2 instances available in this region. The EC2 instance to launch into must have a tag with key *AWSDRS* and value *AllowLaunchingIntoThisInstance* to appear in the list, and it must be stopped prior to launching into it. When this value is set, the **Transfer server tags** and **Copy private IP** settings will be deactivated, as they cannot apply to an already launched instance.

> **Note**
>
> For the instance to appear and perform as a recovery instance in DRS and allow you to run post-launch actions on it, it needs to have an instance profile that includes the policies **AWSElasticDisasterRecoveryRecoveryInstancePolicy** and **AmazonSSMManagedInstanceCore**. The role **AWSElasticDisasterRecoveryRecoveryInstanceWithLaunchActionsRole**, installed from the Default post-launch actions settings page if not already present, contains these policies and can be used as an instance profile.

The launch into an instance will fail if the following pre-requisites are not met:

1. The instance to launch into must have the required tag with key *AWSDRS* and value *AllowLaunchingIntoThisInstance*.
2. The instance to launch into has been stopped.
3. The instance to launch into must have the same operating system platform (Linux or Windows) as that of the server it is protecting.

4. The instance to launch into must support the boot mode of the server it is protecting. If the protected server uses the UEFI boot mode, the instance to launch into must also use the UEFI boot mode. If the protected server uses the legacy BIOS boot mode, the instance to launch into can use either boot mode. This applies to both Linux and Windows.
5. The instance to launch into must meet these requirements:
   - The same processor architecture as the source server: `x86_64` or `arm64`.
   - Hardware virtual machine (HVM) virtualization.
   - An Amazon Elastic Block Store (Amazon EBS) root device.
6. **OS licensing** can only be **Bring Your Own License (BYOL)** if the instance's platform is Linux or if the instance's **tenancy** is **dedicated host**.
7. **Transfer server tags** and **Copy private IP** must be deactivated (this is done automatically when **Launch into instance ID** is set via the console).

- **OS licensing –** Choose whether you want to Bring Your Own Licenses (BYOL) from the source server into the drill or recovery instance.

  The **Use default** option will use the default licensing mechanism for your operating system.

  Choose **BYOL** if:

  - You are migrating a Linux server. All Linux licenses are BYOL by default. Any RHEL, SUSE, or Debian licenses will be transferred in their current form to the recovered instance. Make sure that the terms of your licenses allow this license transfer.
  - You want to BYOL your Windows licenses. This will set up a dedicated host through which all the licenses from the Windows source server will be automatically transferred to the drill or recovery instance.

  > **Important**
  >
  > If you activate BYOL licensing for Windows, you have to change the **Placement.tenancy** type in the EC2 launch template to **Host**. Otherwise, instance launch will fail.

  > **Note**
  >
  > - Windows Desktop Editions require BYOL – note the specific restrictions for AWS Provided Licenses.

- If you are using Windows Servers datacenter: Azure addition, note the specified restrictions for BYOL.

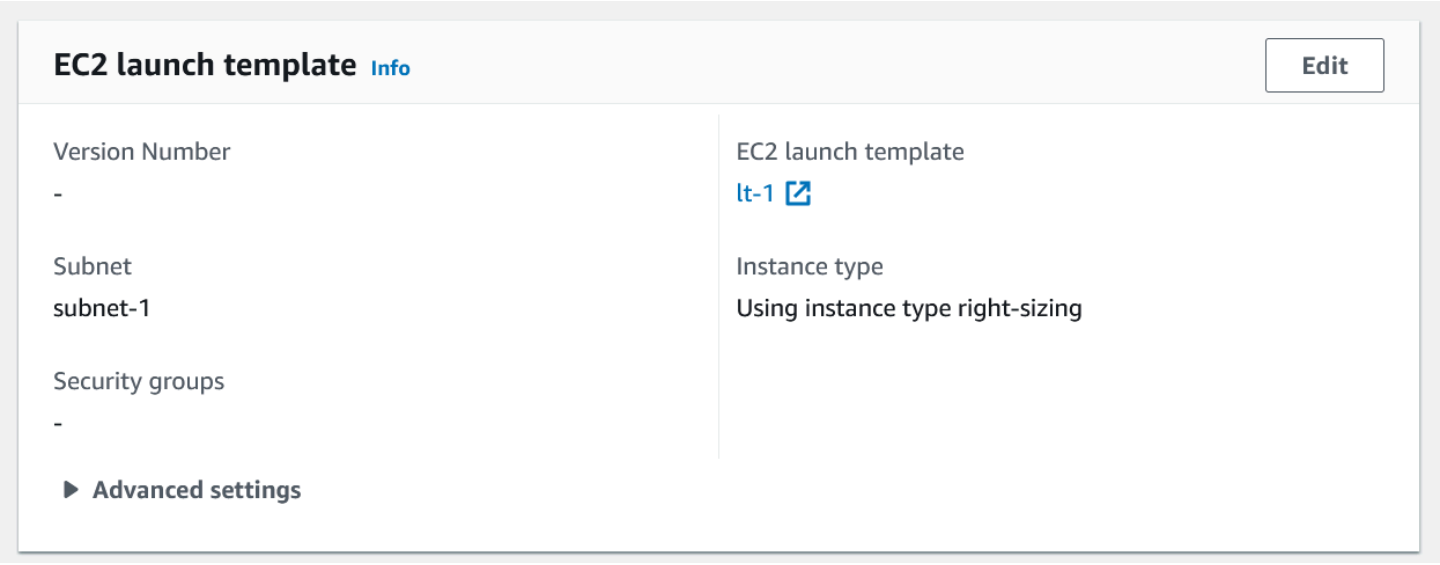
- **Recovery mode –** Choose the recovery mode for launching recovery instances:
  - **Optimal** (default) – Runs the full conversion process before launch. The process includes driver injection, boot configuration, and filesystem modifications. Use this mode when you are uncertain about source server compatibility, or for cross-platform scenarios.
  - **Fast** – Skips the conversion process and launches instances directly from replicated snapshots, which reduces recovery time. Use this mode for AWS-to-AWS disaster recovery scenarios where source servers already have AWS-compatible drivers and configurations.

> **Note**
>
> Fast recovery mode requires DRS agent version 6.42.20 or later.

EC2 launch template

AWS Elastic Disaster Recovery (AWS DRS) utilizes EC2 launch templates to launch drill and recovery EC2 instances for each source server. You can edit those templates for each source server directly from the AWS DRS console.

The EC2 launch template is created automatically for each source server that is added to AWS DRS upon the installation of the AWS Replication Agent.



Topics

- EC2 launch template parameters
- Key considerations for EC2 launch templates
- Full launch template setting review

> **Note**
>
> - In most use cases, the EC2 launch template does not need to be edited.
> - You cannot use the same template for multiple servers.
> - Many EC2 launch template parameters can be changed, but some may not be used by the AWS DRS launch process and some may interfere with the AWS Elastic Disaster Recovery launch process.
> - You must set the EC2 launch template you want to use with AWS DRS as the **default** launch template.

To edit the EC2 template for a single server, take the following steps:

1. Go to the **Source servers** page.
2. Select a source server to update.
3. Under the **Actions** menu, select **Edit EC2 launch settings** and you will be navigated to the **Edit EC2 launch template** page within the AWS DRS console.
4. Change the settings according to your preferences.
5. Click **Save settings**.

Alternatively:

- Go to the **Source servers** page.
- Select a specific source server.
- Go to the **Launch settings** tab.
- Click **Edit** in the EC2 launch template section.

EC2 launch template parameters

AWS Elastic Disaster Recovery (AWS DRS) EC2 launch settings are divided into basic and advanced settings.

The basic settings include:

- **Subnet –** When you specify a subnet, this field defines where the instance will be launched. When selecting a subnet, only the default network interface will be updated. If you do not include a subnet, the launched instance will use the Region's default subnet located in the default VPC.

  > **Note**
  >
  > - If you do not have a default VPC, you must modify the EC2 launch template and explicitly define the subnet in which to launch. Failure to do so will result in errors when launching drill or recovery instances.
  > - For cross-AZ recovery, ensure that the staging area subnet and the subnets that you configure your recovery instances to launch in are not in the same AZ as your source EC2 instances.
  > - If you have multiple on-premises servers that represent the same resources you can use different AZs for recovery instances to increase resiliency.

- **Security groups –** The selected security groups to assign to the instance, applied to the subnet selected for the default network interface. If no security group is selected, there is no default value and no group will be used. Security groups can only be selected if a subnet is included.
- **Instance type –** The default instance type to use when launching. If instance type right-sizing is active, the system will disregard this setting. If no instance type is included, a default value will be used. You can either select an instance type, or you can specify instance attributes and let Amazon EC2 identify the instance types with those attributes.

  > **Note**
  >
  > Select an instance type that uses the same processor architecture as the source server. For `arm64` source servers, AWS Elastic Disaster Recovery uses `arm64`-compatible recovery instance types. See Key considerations for EC2 launch templates.

Instance type attributes:

- **Number of vCPUs:** Enter the minimum and maximum number of vCPUs for your compute requirements. To indicate no limits, mark the no minimum or no maximum checkboxes, or leave them blank.
- **Amount of memory (MiB):** Enter the minimum and maximum amount of memory, in MiB, for your compute requirements. To indicate no limits, mark the no minimum or no maximum checkboxes or leave the fields blank.
- Expand **Optional instance type attributes**: Select an attribute from the **Choose attribute** dropdown and press **Add attribute** to express your compute requirements in more detail. For information about each attribute, see InstanceRequirementsRequest in the *Amazon EC2 API Reference*.
- **Preview matching instance types:** You can preview the instance types that match the specified attributes. To exclude instance types, you can select the instance types you want to exclude from the previewed list of instance types, but only if you did not allow instance types, as you can either exclude or allow instance types but not both.

See more about these attributes here: How attribute-based instance type selection works. EC2 uses fleets to launch your instances and applies default price protection settings to avoid selecting expensive instance types. For current price protection defaults, see Price Protection.

To learn more about using instance type attributes in DRS, visit Flexible Instance Types.

Advanced settings include additional parameters that add specific features to the EC2 template. If you choose not to include these parameters in the template, the specific capabilities will not be added.

The advanced settings include:

- **IAM instance profile –** Attach a specific profile to the instance that will be launched. Make sure the instance profile has the AWSElasticDisasterRecoveryRecoveryInstancePolicy IAM policy attached in addition to any other policy.
- **Auto assign public IP –** Automatically assign a public IP to the launched instance.
- **Termination protection –** Protect the launched instance from accidental termination using the EC2 console.
- **Tenancy –** Set tenancy information, such as dedicated host needed in conjunction with setting BYOL for Windows servers and Windows Home.

- **Capacity reservation –** Apply reservation consideration to the launched instances.
- **Key pair –** Associate a key pair with launched instances that are based on EC2 instances.

> **Note**
>
> AWS DRS only supports major EC2 template parameters. If you want to change values that are not supported by this feature, you can still do so by editing the EC2 launch template via the Amazon EC2 console:
>
> - Create a new EC2 template version with the required changes.
> - Mark it as default.

> **Important**
>
> Every time you modify an EC2 launch template on the Amazon EC2 console, a new version is created. AWS DRS uses the version that is marked as the default. If you prefer to use the EC2 launch template you just modified, make sure to mark it as the default. Changes made through the AWS DRS console are automatically set as the default version.

Amazon EBS volume initialization rate – You can set a `VolumeInitializationRate` on Amazon EBS block device mappings in the EC2 launch template to accelerate volume initialization during drill and recovery launches. Set this value directly on the launch template through the Amazon EC2 console or the AWS Command Line Interface (AWS CLI). AWS Elastic Disaster Recovery preserves the rate across its own launch template updates and passes it to volume creation at recovery time. If volume creation fails because of the specified rate, AWS Elastic Disaster Recovery retries without the rate to prevent the rate from blocking recovery.

> **Note**
>
> This feature is available only in AWS Regions and environments that support Amazon EBS volume initialization rate. You are charged per GB based on the full snapshot size and the initialization rate you specified. For more information, see Initialize Amazon EBS volumes and Amazon EBS pricing.

EC2 launch template tags – In addition to the basic and advanced settings, you can also add up to 50 tags. These will be transferred to your drill and recovery instances. Note that these tags may interfere with other tags that have already been added to the source server. Launch template tags always take precedence over tags set in the AWS DRS Console or tags manually added to the server.

Learn more about EC2 launch template settings and configuration options in this EC2 article.

Key considerations for EC2 launch templates

Review the following key rules and interactions before you modify an EC2 launch template for use with AWS Elastic Disaster Recovery.

1. **Instance type** – AWS Elastic Disaster Recovery uses the instance type set on the launch template unless Instance type right-sizing is activated. If right-sizing is active, it overrides the launch template value.

   > **Note**
   >
   > The recovery instance type must use the same processor architecture as the source server. For an `arm64` source server, choose an AWS Graviton instance type. Elastic Disaster Recovery validates the architecture before launch. The launch fails if the instance type, instance requirements, or Amazon Machine Image (AMI) do not match.

2. **Subnet** – If you do not have a default VPC, you must explicitly define the subnet. Failure to do so results in errors when launching drill or recovery instances.
3. **Private IP and subnet** – If you use the Copy private IP feature, ensure that the IP is included in the subnet's CIDR block range. Otherwise, instance launch fails.
4. **Network interfaces** – AWS Elastic Disaster Recovery manages the primary network interface (device index 0). Additional network interfaces defined in the launch template are preserved and passed through at launch time.
5. **Custom device name** – Do not alter this field. AWS Elastic Disaster Recovery uses the device name as defined on the source server to map disks on the drill or recovery instance.
6. **Disks** – You cannot add disks to the EC2 launch template. Any disks added that do not exist on the source server are ignored.
7. **Launch template name** – Do not alter this field. AWS Elastic Disaster Recovery automatically generates this name.

8. **Volumes** – For each EBS volume, AWS Elastic Disaster Recovery uses the user-selected values. If no matching volume exists in the launch template, AWS Elastic Disaster Recovery uses default values. If the launch template includes a volume that does not exist on the source server, the system disregards it. If you delete the EC2 launch template, the service creates a new one with default values.
9. **Volume initialization rate** – AWS Elastic Disaster Recovery preserves `VolumeInitializationRate` values that you set on launch template Amazon EBS block device mappings. AWS Elastic Disaster Recovery passes the rate to volume creation during drill and recovery launches to accelerate volume initialization. If volume creation fails because of the rate, AWS Elastic Disaster Recovery retries without it to prevent the rate from blocking recovery. Possible failure reasons include quota limits exceeded or an unsupported Availability Zone. This feature is available only in AWS Regions and environments that support Amazon EBS volume initialization rate. For more information, see Initialize Amazon EBS volumes.
10. **KMS key** – Encryption for recovery volumes is controlled through the EBS Encryption section of the replication settings within the AWS Elastic Disaster Recovery console. The launch template's KMS key field is not used by AWS Elastic Disaster Recovery during volume creation. To encrypt recovery volumes with a specific KMS key, configure it in the replication settings.
11. **Automatic cleanup** – AWS Elastic Disaster Recovery deletes the EC2 launch template for source servers that have been disconnected from AWS Elastic Disaster Recovery or for which recovery has been finalized.
12. **Tags** – Launch template tags always take precedence over tags set in the AWS Elastic Disaster Recovery console or tags manually added to the server.
13. **Launch template AMI** – Make sure your launch template AMI matches the boot mode of your source server. If the source uses Unified Extensible Firmware Interface (UEFI), the chosen AMI must support UEFI. For an `arm64` source server, use an `arm64` AMI, or remove the AMI override so that Elastic Disaster Recovery creates an architecture-matched AMI. An `arm64` AMI must use UEFI boot mode; if you leave the AMI boot mode unset, Amazon EC2 uses UEFI on Graviton instance types.

Full launch template setting review

The following sections describe each EC2 launch template field and indicate whether you can safely change it. Fields marked **Do not change** must remain at their default values for AWS Elastic Disaster Recovery to function properly.

Top-level fields

Launch template name – Do not change

AWS Elastic Disaster Recovery automatically generates the launch template name to maintain the association between the template and its source server. Changing this name breaks the link between the source server and its launch configuration.

Template version description – You can edit

You can update the version description to track changes you make to the template. This field is informational and does not affect launch behavior.

AMI – Do not typically set

AWS Elastic Disaster Recovery creates the appropriate AMI from the replicated source server data during drill or recovery. Setting an AMI manually overrides this behavior and can result in an instance that does not match your source server.

Instance type – You can edit

Set the instance type to use when launching drill or recovery instances. If Instance type right-sizing is active, the right-sizing value overrides this setting. If no instance type is specified, AWS Elastic Disaster Recovery uses a default value. For source-server architecture requirements, see Key considerations for EC2 launch templates.

Storage (volumes)

Device name – Do not change

AWS Elastic Disaster Recovery uses the device name from the source server to map disks to the drill or recovery instance. Changing this value causes disk mapping failures.

Snapshot – Do not change

AWS Elastic Disaster Recovery manages snapshots as part of the replication process. Specifying a different snapshot causes the launched instance to use data that does not reflect the source server state.

Size – You can increase

Do not decrease the volume size below the source disk size, as this causes launch failures. You can increase the size if you need additional capacity on the recovery instance. Be aware that

AWS Elastic Disaster Recovery automatically increases this value if the source disk grows larger than the value set in the launch template.

Volume type – You can edit

You can change the EBS volume type. The default is `gp3`. Valid options include `gp2`, `gp3`, `io1`, `io2`, `st1`, `sc1`, and `standard`.

IOPS – You can edit

You can set the provisioned IOPS for volume types that support this setting (`io1`, `io2`, and `gp3`).

Throughput – You can edit

You can set the throughput (in MiB/s) for `gp3` volumes only.

Delete on termination – You can set

Specify whether to delete the volume when the instance terminates. Set this based on your data retention requirements.

Encrypted – Not used by DRS

The launch template Encrypted field is not used by AWS Elastic Disaster Recovery. Recovery volume encryption is determined by the EBS Encryption section of the replication settings.

KMS key – Not used by DRS

The launch template KMS key field is not used by AWS Elastic Disaster Recovery. Recovery volume encryption is determined by the EBS Encryption section of the replication settings. To use a customer managed key for recovery volumes, configure it in the replication settings rather than the launch template.

> **⚠ Important**
>
> Do not add additional disks to the launch template. Any volumes defined in the launch template that do not exist on the source server are ignored by AWS Elastic Disaster Recovery.

Network interfaces

Device index – Do not change

The device index must remain set to `0` for the primary network interface. AWS Elastic Disaster Recovery requires the primary interface to be at index 0 for proper instance configuration.

Subnet – You can edit

Specify the subnet in which to launch the drill or recovery instance. If you do not have a default VPC, you must set this value. For cross-AZ recovery, ensure that the subnet is not in the same Availability Zone as your source instances.

Auto-assign public IP – You can edit

Specify whether to automatically assign a public IP address to the launched instance. Set this based on your network access requirements.

Primary IP – You can edit

The Copy private IP feature uses this field. If you enable Copy private IP, ensure the specified IP is within the subnet's CIDR block range. Otherwise, instance launch fails.

Security groups – You can edit

Specify the security groups to assign to the launched instance. Security groups can only be set if a subnet is defined.

Delete on termination – You can set

Set this to **Yes** so that network interfaces are cleaned up when you terminate drill or recovery instances.

System configuration fields – Do not change

The following advanced detail fields must remain unchanged for AWS Elastic Disaster Recovery to function properly. Changing these values can cause launch failures or unexpected instance behavior.

- RAM disk ID
- Kernel
- Nitro Enclave

- Metadata accessible – AWS Elastic Disaster Recovery uses instance metadata to determine whether an instance is a recovery instance so that it can complete final recovery tasks.

Flexible Instance Types

AWS Elastic Disaster Recovery (AWS DRS) uses EC2 Launch Templates to define how each source server will be launched as an EC2 instance. For each source server, you can define an EC2 instance type that will be used to launch, if **Instance-type right-sizing** is not activated. However, you might want to be flexible with the instance type definitions as this can help in certain situations, such as reducing the risk of not finding enough resources to support recovery which results in an **Insufficient Capacity Error (ICE)**.

How it works

Using attribute based instance type selection allows you to specify attributes to define a set of instance types that can be used to launch an EC2 instance for the source server. These instance types will only include those that are offered in the AWS Region and Availability Zone specified by the subnet in your launch template. To learn more about attribute based instance type selection, visit: How attribute-based instance type selection works.

> **Note**
>
> Elastic Disaster Recovery matches only instance types that use the processor architecture of the source server. For an `arm64` source server, the launch uses AWS Graviton instance types. If your instance requirements exclude every `arm64` instance type, the launch fails. Elastic Disaster Recovery does not fall back to an `x86_64` instance type.

DRS will use EC2 fleets to launch instances using the attribute settings defined in the EC2 launch template. This will apply price protection to ensure the fleet picks the most cost effective instance types for you. We use the default protection settings, so we will protect against selecting instance types that are 20% more expensive than the lowest cost instance type. To learn more about price protection using fleets, visit: Price protection.

When launching an instance, EC2 fleets will attempt to launch an instance type listed in **Preview matching instance types**. It will start from the lowest cost instance type. If an instance type has run out of capacity in the current AWS Region and Availability Zone, EC2 fleets will attempt to

launch with another instance type from the list. It is expected each attempt will take up to a few minutes, so if most of the instance types are not available, it could have an impact on your Recovery Time Objective (RTO).

A large and diverse list of possible instance types listed in **Preview matching instance types** increases the chances of finding an available instance type.

How to use

To use this setting, you can modify the **Default launch settings** (editing the **Default EC2 launch template**) or the **Launch settings** of one or more source servers, and edit the **EC2 launch template**.

Modifying the default launch template settings will only apply to servers added after the change. Modifying the settings of existing source servers will apply to the next launch of those servers.

In the **Instance type** section of the page, select **Specify instance type attributes**.

Attribute selection

These are the attributes that you can set:

- **Number of vCPUs:** Enter the minimum and maximum number of vCPUs for your compute requirements. To indicate no limits, mark the no minimum or no maximum checkboxes.
- **Amount of memory (MiB):** Enter the minimum and maximum amount of memory, in MiB, for your compute requirements. To indicate no limits, mark the no minimum or no maximum checkboxes.
- Expand **Additional instance type attributes** and choose **Add attribute** to express your compute requirements in more detail. For information about each attribute, see InstanceRequirementsRequest in the Amazon EC2 API Reference. For example, you can set **Allowed instance types** to include additional specific instance types, or alternatively, you can use **Excluded instance types** to exclude specific instance types out of the resulting instance types. Note that allowed and excluded instance types cannot be specified together. You can also exclude instance types by selecting instance types from the **Preview matching instance types** and clicking the **Exclude selected instance types**.
- **Preview matching instance types:** You can preview the instance types that match the specified attributes. To exclude instance types, you can select the instance types you want to exclude from the previewed list of instance types, but only if you did not allow instance types, as you can

either exclude or allow instance types but not both. You can exclude instance types by selecting instance types from the **Preview matching instance types** and clicking the **Exclude selected instance types**. If this list is empty, try to expand the attribute values to allow more options.

You can use **Allowed instance types** to add most of the supported instance types to the **Preview matching instance types** list. This way you can create a limited set of instance types, which may be useful in situations where you are constrained by the instance type options you can use, and can only use a small list of those.

Note

Adding burstable instance types, such as t2 or t3, to the allow attribute will not affect the list in **Preview matching instance types**. However, if you select a burstable instance type and update the **EC2 launch template** on the EC2 console by selecting *Burstable performance support* from the **Attribute** dropdown, setting the **Attribute value** to *Include* (or *Required*), and set the new template version as a default, these instance types will affect the preview list.

Note

You can define attributes directly on the EC2 launch template. If you have defined an attribute that is not supported on the DRS console, that attribute will still affect the list of instance types in **Preview matching instance types**, even if the DRS console will not display it.

Note

Note: Flexible instance types cannot be used if **instance type right sizing** is active, or if the subnet selected for launch is on an AWS Outpost rack.

Using Elastic Disaster Recovery for recovery and failback

In the event of a disaster, AWS Elastic Disaster Recovery facilitates the recovery of your workloads by launching recovery instances in AWS. Once the disaster has been mitigated, you can also use AWS Elastic Disaster Recovery to perform failback to the original source infrastructure.

Key terminology

The following terms are used throughout the AWS Elastic Disaster Recovery documentation. Understanding the distinction between these terms is important for using the service effectively.

Recovery

The process of launching recovery instances on AWS using AWS Elastic Disaster Recovery. This is the action you perform within the Elastic Disaster Recovery Console or API (using `start-recovery` or **Initiate recovery job** in the Console). Recovery creates new Amazon EC2 instances from your replicated data.

Recovery drill

A non-disruptive test that launches drill instances to validate your disaster recovery readiness. Drills use the same process as recovery but do not affect your source servers or ongoing replication.

Failover

The act of redirecting production traffic from your primary (source) environment to your recovery instances on AWS. Failover is performed outside of AWS Elastic Disaster Recovery, typically using a DNS routing service such as Amazon Route 53 or your organization's traffic management solution.

Failback

The process of returning your workloads from the recovery environment on AWS back to your original source infrastructure after the disaster has been resolved. AWS Elastic Disaster Recovery assists with failback by replicating data from recovery instances back to your source servers.

In summary: AWS Elastic Disaster Recovery handles **recovery** (launching instances) and **failback** (returning to source). The **failover** step (redirecting traffic) is performed by you, outside of AWS Elastic Disaster Recovery.

Recovery and Failback overview

AWS Elastic Disaster Recovery provides scalable resilience to your existing infrastructure, coupled with low Recovery Point Objectives (RPO) and Recovery Time Objectives (RTO). Learn more about how AWS Elastic Disaster Recovery (DRS) can meet your team's RPO and RTO.

Understanding recovery

Recovery allows you to orchestrate launch of your workload within AWS EC2 Instances. After initial sync is completed, you are able to customize the configuration of the recovery environment in preparation of a business continuity event.

AWS Elastic Disaster Recovery allows you to launch Drill and Recovery instances for your source servers in AWS once they are in **Continuous Data Protection**. While Drill Instances and Recovery Instances are launched similarly, they serve different purposes. During normal operations, we recommend periodically testing your ability to recover using DRS by using Drill Instances.

To recover a group of interdependent source servers in a defined order, with wait times between groups, use a recovery plan. For more information, see Orchestrating recovery with recovery plans.

Understanding failback

Failback allows you to restore your Recovery Instances back to your source infrastructure. Depending on the source infrastructure, performing a failback uses differing mechanisms.

Source Infrastructure	Failback Mechanism	More Information
On-Premise	Use the Failback Client ISO or the DRS Failback Automation.	On-Premise Failback
AWS - Same Account	Start Reverse Replication on the Protected Recovery Instance.	Same Account Failback
AWS - Cross Account	Start Reverse Replication on the Protected Recovery Instance in Failover Account.	Cross Account Failback

Source Infrastructure	Failback Mechanism	More Information
Other Cloud	Configuration varies per provider.	Other Cloud Failback

Preparing for recovery

After installing the AWS Elastic Disaster Recovery Agent on your Source Servers, we recommend validating your Source Server settings and testing (drilling) frequently in preparation of a recovery event. Configuration of the recovery environment includes DRS Launch Settings, EC2 Launch Template, and Post-Launch Actions.

Valid and up-to-date configuration and drilling facilitates lowering the RTO.

Validate launch settings

After successful installation, we recommend validating your individual Source Server Settings to ensure they meet your recovery requirements. These settings can even be modified during the **Initial Sync** phase.

Launch Setting	Example Settings	More Information
DRS Launch Settings	• Automated Instance type right-sizing. • Start instance on launch. • Operating System Licensing.	DRS Launch Settings
EC2 Launch Template	• Instance profile (IAM role attached to the instance). • Recovery Instance VPC, Subnet, and Security Group configuration.	EC2 Launch Template
Post Launch Actions	• Install CloudWatch agent. Validate HTTP/HTTPS connectivity.	Post Launch Actions

Recovery drill overview

A Recovery Drill is a non-disruptive test that performs all the same steps as an actual recovery. Recovery Drills run with the same Source Server Launch Settings and Point in Time snapshots that a Recovery would. As a result, we recommend adjusting any Source Server Launch Settings to isolate Drill Instances when necessary to avoid production or business impact. You can use verification post-launch actions when performing a drill to ensure that Launch Settings are accurate. A Recovery Drill can be performed with an individual source server, or it can include as many source servers as necessary to simulate the recovery of an application.

Recovery Drills will create EC2 resources in your Target AWS Account upon completion; these resources will be billed by the respective service until deleted. Recovery Drill EC2 resources will automatically be cleaned up if a Recovery Drill is performed again with the same Source Server.

Recovery drill objectives

Performing a Recovery Drill will assist in ensuring DRS can fulfill your Recovery Objectives during a failover event. Some Recovery Objectives can include:

- Ensuring Recovery Instances obtain Healthy System and Instance Status Checks.
- Ensuring all components in an application can communicate with one another.
- Ensuring users can interact successfully with the application.

Frequent and successful Recovery Drills will ensure your team can meet RTO/RPO goals during a failover event. We recommend performing a drill on at least a quarterly basis; individual compliance needs may necessitate more frequent drills.

Performing recovery drills

Once a Source Server has reached **Healthy**, a recovery drill can be performed. Recovery Drills should also be performed whenever the last recovery result was not Successful, or it has been a significant amount of time since a Successful Recovery Drill has been performed.

As long as Initial Sync has completed, a Recovery Drill can be performed, even if a Source Server is in **Lag** or **Stall** status.

DRS Console

Performing a Recovery Drill

1. Navigate to the AWS Elastic Disaster Recovery Console. In the left navigation pane, select **Source Servers**
2. Select one or more source servers, then select **Initiate Recovery Job**.
3. Select **Initiate recovery drill**
4. Select a Point in Time to recover to:
   - Select "Use most recent data" to attempt to create a sub-second RPO snapshot from the source server(s).
   - Select a specific time to use snapshots created at that timestamp, or slightly before if a snapshot was unavailable for a particular source server(s).
5. Select **Initiate drill**.
6. (Optional) Monitor Recovery Drill progress from the AWS Elastic Disaster Recovery Console **Recovery Job History**.

Command Line

Performing a Recovery Drill

Recovery Drills can be started via command line.

1. (optional) Obtain Recovery (PIT) Snapshot to recover to:
   - describe-recovery-snapshots (AWS CLI)

     ```
     aws drs describe-recovery-snapshots --source-server-id s-123456789abcdefgh
     ```

   - Get-EDRSRecoverySnapshot (DRS Tools for Windows PowerShell)

     ```
     Get-EDRSRecoverySnapshot -SourceServerID s-123456789abcdefgh
     ```

2. Perform a Recovery Drill, specifying IsDrill:
   - start-recovery (AWS CLI)

     With Recovery Snapshot

```
aws drs start-recovery --source-servers
 recoverySnapshotID=pit-123456789abcdefgh,sourceServerID=s-123456789abcdefgh
 --is-drill
```

Attempt to Use Latest Snapshot

```
aws drs start-recovery --source-servers sourceServerID=s-123456789abcdefgh
 --is-drill
```

- Start-EDRSRecovery (DRS Tools for Windows PowerShell)

  With Recovery Snapshot

```
$sourceServer = new-object Amazon.Drs.Model.StartRecoveryRequestSourceServer
$sourceServer.RecoverySnapshotID = 'pit-123456789abcdefgh'
$sourceServer.SourceServerID = 's-123456789abcdefgh'
Start-EDRSRecovery -SourceServer $sourceServer
```

  Attempt to Use Latest Snapshot

```
$sourceServer = new-object
 Amazon.Drs.Model.StartRecoveryRequestSourceServer;
$sourceServer.SourceServerID = 's-123456789abcdefgh'
Start-EDRSRecovery -SourceServer $sourceServer
```

Post recovery drill actions

Once a Recovery Drill has been successfully completed, we recommend cleaning up the recovery environment. Leaving Recovery Drill resources running may result in increased AWS charges. We recommend cleaning up your environment via AWS Elastic Disaster Recovery to ensure all resources created during the drill are removed.

DRS Console

Performing a Recovery Drill

1. Navigate to the AWS Elastic Disaster Recovery Console. In the left navigation pane, select **Recovery instances**.

2. Select one or more source servers, then select **Actions**.
3. Select **Terminate recovery instances**.
4. Select **Terminate** on any dialog boxes.

Command Line

Cleaning up Recovery Drill

Cleaning up Drills can be started via command line.

1. Identify any Recovery Instances.
   - describe-recovery-instances (AWS CLI)

     ```
     aws drs describe-recovery-instances
     ```

   - Get-EDRSRecoveryInstance (DRS Tools for Windows PowerShell)

     ```
     Get-EDRSRecoveryInstance
     ```

2. Terminate the Recovery Instances.
   - terminate-recovery-instances (AWS CLI)

     ```
     aws drs terminate-recovery-instances --recovery-instance-
     ids i-123456789abcdefgh
     ```

   - Stop-EDRSRecoveryInstance (DRS Tools for Windows PowerShell)

     ```
     Stop-EDRSRecoveryInstance -RecoveryInstanceIDs 'i-123456789abcdefgh'
     ```

Orchestrating recovery with recovery plans

Multi-tier applications usually have to be recovered in a specific order. A database tier must be running before the application tier starts, and the application tier must be running before the web tier accepts traffic. Recovering these servers one at a time, in the right order, during a disaster is slow and error-prone.

A **recovery plan** captures that order once so that you can run it on demand. You group your source servers into ordered steps, optionally insert wait times between steps, and then run the whole plan

with a single action. AWS Elastic Disaster Recovery recovers each step in sequence. Within a step it recovers all of the servers in parallel, and waits for every server to reach a terminal state before it starts the next step.

You can run a recovery plan as a drill at any time to validate your disaster recovery readiness without affecting your source servers, and run the same plan in recovery mode during an actual event.

> **Note**
>
> Recovery plans recover source servers within a single AWS account and AWS Region. All of the source servers in a plan must be in the same account and Region as the plan.

Topics

- Recovery plan concepts
- How a recovery plan runs
- Validation that runs before a server is recovered
- Controlling how server failures affect a plan
- Creating a recovery plan
- Adding and ordering steps
- Running a recovery plan
- Monitoring an execution
- Retrying, skipping, and canceling
- Editing and deleting recovery plans
- Recovery plan quotas

Recovery plan concepts

The following terms are used throughout the recovery plan documentation:

Recovery plan

An ordered list of steps that defines how a group of source servers is recovered. A plan belongs to a single AWS account and Region, and its name must be unique within that account and Region.

Step

One entry in the ordered list. A step is either a **server step** or a **wait step**. Steps run one at a time, in order.

Server step

A step that contains one or more source servers. AWS Elastic Disaster Recovery starts recovery for every server in the step at the same time and considers the step finished only after every server has finished.

Wait step

A step that pauses the plan for a fixed number of minutes before the next step starts. Use a wait step to give an application time to initialize after its instances launch. A wait step cannot be the first step in a plan.

Impact level

A per-server setting on a server step that determines whether the failure of that server stops the plan. A **Critical** server (the default) fails the step and the execution. An **Optional** server does not. For more information, see Controlling how server failures affect a plan.

Execution

A single run of a recovery plan. An execution is a point-in-time copy of the plan: it records the steps and servers as they were when the execution started, so later edits to the plan do not change a run that is already in progress.

Execution mode

Whether the run is a **Drill** or a **Recovery**. The mode applies to every server step in the plan and is equivalent to choosing between a drill and a recovery for an individual source server.

How a recovery plan runs

When you start an execution, AWS Elastic Disaster Recovery does the following:

1. Takes a point-in-time copy of the plan, including its steps, its servers, and each server's impact level.
2. Validates every source server across all of the server steps in the plan. If any server fails validation, the execution fails immediately and AWS Elastic Disaster Recovery recovers no

servers. Validation applies to every server regardless of its impact level. For what is checked, see Validation that runs before a server is recovered.

3. Runs each step in order. For a server step, AWS Elastic Disaster Recovery starts a recovery job for every server in the step at the same time. For a wait step, AWS Elastic Disaster Recovery pauses for the configured number of minutes.
4. Revalidates the servers in a server step at the moment that step begins. A server that passed validation when the execution started can still fail here if something changed in the meantime, such as replication stalling or a chosen recovery point being deleted.
5. Waits for every server in the current step to finish before evaluating the step's outcome and moving to the next step.
6. Marks the execution `COMPLETED` after the last step finishes, or `FAILED` as soon as a step fails.

Because each server step launches its servers in parallel, a step takes as long as its slowest server. Recovery plans use the same recovery process, launch settings, and launch templates as an individual recovery. A recovery that a plan launches behaves exactly like one that you start yourself. The resulting recovery instances appear on the **Recovery instances** page and each server's recovery job appears in Recovery job history.

> **Important**
>
> An execution must finish within 24 hours of the time it started. If it does not, AWS Elastic Disaster Recovery stops advancing the plan and marks the step that was running as `TIMED_OUT`. AWS Elastic Disaster Recovery does not cancel the recovery jobs that are already in progress, and those jobs run to completion. The steps after the timed-out step do not run, and you cannot resume the execution. To finish the recovery, start a new execution of the plan, or recover the remaining servers individually.
> Size your plans so that they finish well inside this limit. A step takes as long as its slowest server, so the practical ceiling is the sum of your server steps plus the sum of your wait steps.

Validation that runs before a server is recovered

AWS Elastic Disaster Recovery validates the source servers in a plan twice: once for the whole plan when an execution starts, and again for the servers in a server step at the moment that step begins. The checks are the same as the ones that run when you recover a source server on its own.

A server fails validation for any of the following reasons:

- The source server no longer exists in the account, for example because it was deleted after it was added to the plan.
- The server has no consistent recovery point yet. This is the check that catches a server whose initial sync has not finished, and a server whose replication has never produced a usable snapshot.
- Another job is already running for the server. A server that is in the middle of a recovery, drill, or other operation cannot be recovered again until that job finishes.
- The server is in a lifecycle state that cannot be recovered, such as disconnected, stopped, or archived.
- The server's disks are missing volume attributes, or those attributes are still pending. This affects servers that need marketplace license information.
- The server's launch template conflicts with the server. For example, a Windows server using its own license is not set to host tenancy, or the instance type does not support the server's boot mode.
- A recovery point that you pinned for the server no longer exists.
- The same source server appears more than once in the request.

> **Important**
>
> Validation is all or nothing, and impact levels do not apply to it. If any server in the plan fails the validation that runs when the execution starts, the whole execution fails. AWS Elastic Disaster Recovery then recovers no servers at all, not even the servers that you marked **Optional**. If any server fails the revalidation at the start of a server step, that whole step fails, again regardless of impact level. AWS Elastic Disaster Recovery recovers none of the servers in that step.

Before you run a plan in recovery mode, confirm that every server in it shows a healthy replication state and has recovery points available. Running the plan as a drill first is the most reliable way to find these problems early.

Controlling how server failures affect a plan

Every server in a server step has an impact level that determines whether its failure stops the plan:

Impact level	Effect when the server fails to recover
Critical (default)	The step fails, the execution fails, and the remaining steps do not run. Servers in the same step that already started continue to completion.
Optional	The server fails, but the step still completes and the plan continues to the next step. The failed server is reported on the step so that you can see which server did not recover.

Mark a server **Optional** when your application can start without it, for example, a reporting server or a secondary worker node. Leave a server **Critical** when a later tier depends on it.

If a step contains only **Optional** servers and all of them fail, the step still completes and the plan continues.

> **Note**
>
> Impact levels apply only to servers that actually started recovering and then failed. They do not apply to validation. If a server fails the validation that runs when the execution starts, the whole execution fails; if a server fails the revalidation at the start of its step, that whole step fails. Both happen even if the server is marked **Optional**, because AWS Elastic Disaster Recovery starts recovery for all of a step's servers in a single request. For more information, see Validation that runs before a server is recovered.

Creating a recovery plan

Create an empty plan, then add the steps that define your recovery order. A plan cannot be run until it contains at least one server step:

To create a recovery plan (console)

1. Open the Elastic Disaster Recovery console at https://console.aws.amazon.com/drs/home.
2. In the navigation pane, choose **Recovery plans**.
3. Choose **Create recovery plan**.
4. Enter a **Name** for the plan. The name must be unique within your account and Region.
5. (Optional) Enter a **Description**.

6. Add your steps in the order that you want them to run. For each server step, select the source servers to recover in that step and set the impact level for each one. For each wait step, enter the number of minutes to pause.
7. Review the plan and choose **Create recovery plan**.

To create a plan with the AWS CLI, use the `create-recovery-plan` command.

```
aws drs create-recovery-plan \
    --name "Tier1-ecommerce" \
    --description "Database, application, and web tiers for the storefront"
```

The response includes the `recoveryPlanArn` that you use to add steps and to start an execution.

A plan name must be 1 to 256 characters long, must start with a letter or a number, and can contain letters, numbers, spaces, underscores, and hyphens.

Adding and ordering steps

Steps are numbered from 1 and run in ascending order. A plan can contain up to 20 steps and up to 100 source servers in total across all of its steps.

The following rules apply to steps:

- A source server can appear in only one step within a single plan. The same source server can appear in more than one plan.
- A wait step cannot be the first step in a plan, because there is nothing to wait for.
- A wait step must be between 1 and 120 minutes.
- You cannot change a step's type after you create it. To change a server step into a wait step, delete it and create a new one.

To add a server step with the AWS CLI, use the `create-recovery-plan-step` command with a `serverStepConfiguration`. Servers that you do not give an `impactLevel` default to `CRITICAL`.

```
aws drs create-recovery-plan-step \
    --recovery-plan-arn PLAN_ARN \
    --step-name "Database tier" \
    --configuration '{
```

```
        "serverStepConfiguration": {
            "servers": [
                {"serverArn": "SERVER_ARN_1", "impactLevel": "CRITICAL"},
                {"serverArn": "SERVER_ARN_2", "impactLevel": "OPTIONAL"}
            ]
        }
    }'
```

To add a wait step, use a `waitStepConfiguration` instead.

```
aws drs create-recovery-plan-step \
    --recovery-plan-arn PLAN_ARN \
    --step-name "Wait for database startup" \
    --configuration '{"waitStepConfiguration": {"waitDurationMinutes": 10}}'
```

By default, a new step is added to the end of the plan. To insert a step at a specific position, include `--step-order`; the steps at and after that position move down by one. To reorder the steps that a plan already has, use `reorder-recovery-plan-steps` and pass the complete list of step ARNs in the order that you want.

```
aws drs reorder-recovery-plan-steps \
    --recovery-plan-arn PLAN_ARN \
    --ordered-step-arns STEP_ARN_2 STEP_ARN_1 STEP_ARN_3
```

To change the servers in a server step, use `update-recovery-plan-step`. The server list is replaced in full, so include every server that you want the step to keep.

Finding a source server ARN

The step APIs identify a source server by its ARN, in the `serverArn` field. A source server ARN has the following format.

```
arn:partition:drs:region:account-id:source-server/s-EXAMPLE1
```

To find the ARNs of your source servers, use `describe-source-servers` and read the `arn` field of each entry in the response.

```
aws drs describe-source-servers \
    --query 'items[].{arn:arn,hostname:sourceProperties.identificationHints.hostname}'
```

> **Note**
>
> The step APIs take a full ARN in `serverArn`, but `start-recovery-plan-execution` takes the short source server ID in `sourceServerID` instead, for example `s-EXAMPLE1`. The short ID is the last part of the ARN, after `source-server/`.

Running a recovery plan

You can run a plan as a drill or as a recovery:

Drill

Launches drill instances to validate your recovery readiness. Drills do not affect your source servers or ongoing replication. Run drills regularly so that your plan is proven before you need it.

Recovery

Launches recovery instances for an actual recovery event.

A plan recovers each server the same way an individual recovery does. Any post-launch actions that are active for a source server run for that server's recovery instance exactly as they would for an individual recovery. For more information, see Post-launch action settings.

> **Important**
>
> AWS Elastic Disaster Recovery does not terminate the instances that a plan launches, including drill instances, and it does not clean them up when an execution completes, is canceled, or times out. Every instance a plan launches keeps incurring Amazon EC2, Amazon EBS, and associated charges until you terminate it. Because a plan can launch up to 100 instances at once, clean up after every drill. You can terminate recovery instances from the **Recovery instances** page; see Managing recovery instances and Post recovery drill actions. For charge details, see AWS Elastic Disaster Recovery pricing.

To run a recovery plan (console)

1. Open the Elastic Disaster Recovery console at https://console.aws.amazon.com/drs/home.

2. In the navigation pane, choose **Recovery plans**, and then choose the plan that you want to run.
3. Choose **Execute recovery plan**.
4. Choose **Drill** or **Recovery**.
5. (Optional) Choose a specific recovery point for individual servers. If you do not choose a recovery point for a server, AWS Elastic Disaster Recovery recovers that server from the latest data available. That is the same behavior as recovering the server on its own without choosing a recovery point.
6. Confirm your choices to start the execution.

To start an execution with the AWS CLI, use the `start-recovery-plan-execution` command.

```
aws drs start-recovery-plan-execution \
    --recovery-plan-arn PLAN_ARN \
    --mode DRILL
```

To recover specific servers from a specific recovery point, add the `--source-servers` parameter. Every server that you list must belong to the plan and must include a `recoverySnapshotID`. You do not have to list every server in the plan. Any server that you omit is recovered from the latest data available.

```
aws drs start-recovery-plan-execution \
    --recovery-plan-arn PLAN_ARN \
    --mode RECOVERY \
    --source-servers '[
        {"sourceServerID": "s-EXAMPLE1", "recoverySnapshotID": "pit-EXAMPLE1"}
    ]'
```

A `recoverySnapshotID` is an AWS Elastic Disaster Recovery recovery point ID. It is always 21 characters long and begins with the `pit-` prefix, for example `pit-1234567890abcdef1`.

Important

A recovery point ID is not an Amazon EBS snapshot ID. An Amazon EBS snapshot ID begins with `snap-`, and although a recovery point lists the Amazon EBS snapshots that back it in its `ebsSnapshots` field, those `snap-` IDs are not valid values for `recoverySnapshotID`. Passing one is rejected as a validation error.

To find the recovery point IDs that are available for a source server, use `describe-recovery-snapshots`. Ordering by DESC returns the most recent recovery point first.

```
aws drs describe-recovery-snapshots \
    --source-server-id s-EXAMPLE1 \
    --order DESC \
    --max-results 5
```

Each entry in the response `items` list contains the recovery point ID in the `snapshotID` field, along with its `timestamp`. Take the value of `snapshotID` and pass it as `recoverySnapshotID` when you start the execution; the field is named differently in the two APIs.

```
{
    "items": [
        {
            "snapshotID": "pit-1234567890abcdef1",
            "sourceServerID": "s-1234567890abcdef1",
            "expectedTimestamp": "2026-08-18T09:00:00Z",
            "timestamp": "2026-08-18T09:00:12Z",
            "ebsSnapshots": [
                "snap-0123456789abcdef0",
                "snap-0123456789abcdef1"
            ]
        }
    ]
}
```

In this example, `pit-1234567890abcdef1` is the value to pass as `recoverySnapshotID`. The two `snap-` IDs are the Amazon EBS snapshots that back that recovery point, and are not valid values.

To narrow the results to a time range, add the `--filters` parameter with `fromDateTime` and `toDateTime`.

Important

A recovery point that you pin is validated again at the moment its step begins, not only when the execution starts. In a long plan, a step can run many hours after you started the execution. A recovery point that existed then might be deleted, or age out of your point-in-time retention window, before the step runs. If that happens, the step fails and the plan

stops. Pin recovery points only when you need a specific point in time, and prefer recovery points that will still be inside your retention window when the step is expected to run. Omit `--source-servers` to always use the latest data available.

The following conditions prevent an execution from starting. Each one returns the error shown:

Condition	Error
The plan already has an execution in progress. A plan can have only one execution running at a time.	`ConflictException` — *Another execution is already active for plan `planId`*
A source server in this plan also belongs to another plan that has an execution in progress.	`ConflictException` — names the plan and the execution that are holding the server, so that you can wait for that execution or cancel it.
The plan status is not `ACTIVE`. A plan becomes `INVALID` when it has no server steps.	`ConflictException` — *Recovery plan `planId` is in status `status` and cannot be executed. Only ACTIVE plans can be executed.*
The plan has no steps.	`ValidationException` — *Recovery plan `planId` has no steps. Add at least one step to the plan before starting an execution.*
The plan is being deleted.	`ConflictException` — *Cannot start execution: recovery plan `planId` is being deleted.*
A server listed in `--source-servers` is not part of the plan, or is missing a `recoverySnapshotID`.	`ValidationException` — *Source server IDs not found in plan* or *Missing recoverySnapshotID for servers*, listing the servers concerned.
The AWS account has not been initialized for AWS Elastic Disaster Recovery.	`UninitializedAccountException`

Monitoring an execution

To follow an execution in the console, open the plan and choose the execution from its execution history. The execution details show the status of the execution, the status of each step, and the status of each server within a step, including the recovery job that AWS Elastic Disaster Recovery created for it.

With the AWS CLI, use `get-recovery-plan-execution` for the overall status and `list-recovery-plan-execution-steps` for per-step progress. Use `list-recovery-plan-executions` to see the history for a plan, optionally filtered by status.

```
aws drs get-recovery-plan-execution \
    --recovery-plan-execution-arn EXECUTION_ARN

aws drs list-recovery-plan-execution-steps \
    --recovery-plan-execution-arn EXECUTION_ARN
```

An execution reports one of the following statuses:

Execution status	Description
`CREATED`	The execution was accepted and its source servers are being validated. No servers have been recovered yet.
`IN_PROGRESS`	AWS Elastic Disaster Recovery is running the steps of the plan.
`COMPLETED`	Every step reached a terminal state without a critical failure. An execution that skipped one or more steps also reports `COMPLETED` .
`FAILED`	Either the validation at the start of the execution failed, or a step failed because a critical server did not recover. The remaining steps did not run. If a step failed, you can retry or skip that step to continue. If the validation failed, no step ran, so there is nothing to retry or skip; correct the problem the error reports and start a new execution.
`TIMED_OUT`	The execution reached its 24-hour limit while a step was still running. The remaining steps did not run, and you cannot retry, skip, or resume the execution. Start a new execution of the plan to finish the recovery.

Execution status	Description
CANCELLING	You canceled the execution and AWS Elastic Disaster Recovery is waiting for the step that was already running to finish.
CANCELLED	The execution stopped after you canceled it. The steps that had not started were canceled.

Note

If an execution completes with skipped steps, the status is still exactly `COMPLETED`. There is no separate status value for it. The detail appears as a message on the execution: *Completed with skipped steps. View execution details for more information.* If you script against these APIs, branch on the `status` field and treat the execution's `errorDetail` as informational text only. In particular, do not branch on `errorDetail.code`, which is not a reliable signal of failure — a successful execution that skipped a step still returns a code value.

Each step within an execution reports one of the following statuses:

Step status	Description
NOT_STARTED	The step is waiting for the steps before it to finish.
EXECUTING	A server step is recovering its servers.
WAITING	A wait step is counting down its wait duration.
COMPLETED	The step finished. A server step can complete even if optional servers failed.
FAILED	At least one critical server in the step did not recover.
TIMED_OUT	The step was still running when the execution reached its 24-hour limit.
SKIPPED	You skipped the step, or the step was not run because the execution was canceled.

Recovery plan API calls are recorded by AWS CloudTrail, so you can audit who created, changed, or ran a plan. For more information, see Logging AWS Elastic Disaster Recovery API calls using AWS CloudTrail.

Retrying, skipping, and canceling

When a step fails, you do not have to rerun the whole plan. You can retry the failed step, skip it, or cancel the rest of the execution.

Topics

- Retrying a failed step
- Skipping a step
- Canceling an execution

Retrying a failed step

Retrying a step recovers only the servers in that step that failed or timed out. Servers in the step that already recovered stay as they are, and AWS Elastic Disaster Recovery does not recover them a second time. After the retried step completes, the execution continues automatically with the steps that follow it.

To retry a failed step with the AWS CLI, use the `retry-recovery-plan-execution-step` command and pass the ARN of the execution step that you want to retry:

```
aws drs retry-recovery-plan-execution-step \
    --recovery-plan-execution-step-arn EXECUTION_STEP_ARN
```

The following conditions apply to a retry:

- The execution must be `FAILED`. You cannot retry a step while the execution is still in progress.
- The step must be `FAILED`.
- You cannot retry a wait step. Skip it instead.
- No other execution of the same plan may be running.

> **Important**
>
> An execution that stopped because it reached the 24-hour limit reports `TIMED_OUT`, not `FAILED`. You cannot retry or resume it. To finish the recovery in that case, start a new execution of the plan, or recover the remaining servers individually.

Each retry increments the step's attempt count, which is returned with the step.

Skipping a step

Skip a step to move past it without recovering its servers.

To skip a step with the AWS CLI, use the `update-recovery-plan-execution-step` command with `--status SKIPPED` and pass the ARN of the execution step that you want to skip:

```
aws drs update-recovery-plan-execution-step \
    --recovery-plan-execution-step-arn EXECUTION_STEP_ARN \
    --status SKIPPED
```

The following conditions apply to a skip:

- The step must be `NOT_STARTED`, `FAILED`, or `TIMED_OUT`. You cannot skip a step that is currently `EXECUTING` or `WAITING`, and you cannot skip one that already reached `COMPLETED`.
- You can skip a `NOT_STARTED` step while the execution is `IN_PROGRESS` and an earlier step is still running. The skip is recorded immediately and the plan steps over that step when it reaches it. Skipping ahead does not start any step early; only one server step runs at a time.
- Skipping a `FAILED` step in a `FAILED` execution resumes the execution at the following step.

An execution that finishes with skipped steps reports `COMPLETED`.

Canceling an execution

Canceling an execution stops AWS Elastic Disaster Recovery from starting any more steps.

To cancel an execution with the AWS CLI, use the `cancel-recovery-plan-execution` command and pass the ARN of the execution that you want to cancel:

```
aws drs cancel-recovery-plan-execution \
```

```
    --recovery-plan-execution-arn EXECUTION_ARN
```

> **Important**
>
> Canceling does not stop recovery jobs that have already started. The step that is running when you cancel continues until all of its servers finish, and any instances that it launches are created. The execution moves to `CANCELLING` until that step finishes, and then to `CANCELLED`. Steps that had not started are canceled and never run. To clean up instances that were launched before you canceled, terminate them from the **Recovery instances** page.

You can cancel an execution that is `CREATED` or `IN_PROGRESS`. You cannot cancel one that has already completed, failed, or been canceled.

Editing and deleting recovery plans

Use `update-recovery-plan` to change a plan's name or description, and the step commands to change its steps. Editing a plan does not affect an execution that is already running, because an execution runs against the copy of the plan that was taken when it started. Your edits apply to the next execution that you start.

Deleting a step or a plan has the following effects:

- Deleting a step removes its servers from the plan and renumbers the remaining steps so that there are no gaps.
- Deleting the last server step in a plan leaves the plan with no servers to recover. The plan's status becomes `INVALID` and it cannot be run until you add a server step, which returns it to `ACTIVE`. Adding a wait step does not make a plan valid.
- Deleting a plan deletes its steps and its server assignments. You cannot delete a plan while it has an execution in progress.

Deleting a plan does not delete any recovery instances that its executions launched, and it does not affect your source servers or their replication.

> **Note**
>
> You cannot delete a source server from AWS Elastic Disaster Recovery while it belongs to a recovery plan. Remove the server from its plans first, then delete it.

Recovery plan quotas

The following quotas apply to recovery plans in each AWS account and Region:

Resource	Quota
Recovery plans per account, per Region	1,000
Steps per recovery plan	20
Source servers per recovery plan, across all steps	100
Concurrent executions per recovery plan	1
Wait step duration	1–120 minutes
Maximum duration of a single execution	24 hours

The AWS Elastic Disaster Recovery service quotas that apply to an individual recovery, such as the number of concurrent recovery jobs and the number of source servers per account, also apply to the recoveries that a plan starts. A plan does not raise or bypass those quotas.

Performing a failover with Elastic Disaster Recovery

A failover is the redirection of traffic from a primary system to a secondary system. It's a network operation that's performed outside of AWS Elastic Disaster Recovery. AWS Elastic Disaster Recovery helps you perform a failover by launching recovery instances in AWS. Once the Recovery instances are launched, you will need to redirect the traffic from your primary systems to the launched recovery instances.

> **Note**
>
> These instructions also apply to the cross-Region or cross-AZ failover process.

Launching recovery instances

Ready for launch indicators

Prior to launching a Recovery instance, ensure that your source servers are ready for testing by looking for these indicators on the **Source Servers** page:

1. Under the **Ready for recovery** column, the server should show **Ready**
2. Under the **Data replication status** column, the server should show the **Healthy** status.
3. Under the **Last recovery result** column, there should be an indication of a successful Drill instance launch sometime in the past. The column should state **Successful** and show when the last successful launch occurred. This column may be empty if a significant amount of time passed since your last drill instance launch.

Launching recovery instances

To launch a recovery instance for a single source server or multiple source servers:

1. Go to the **Source servers** page and select each server for which you want to launch a recovery instance.
2. Open the **Initiate recovery job** menu and select **Initiate recovery**.
3. Select the Point in time snapshot from which to launch the recovery instance for the selected source server. You can either select the **Use most recent data** option to use the latest snapshot available or select an earlier specific Point-in-time snapshot. You may opt to select an earlier snapshot in case you wish to return to a specific server configuration before a disaster occurred.
4. Choose **Initiate recovery**.

Learn more about Point in Time snapshots.

The AWS Elastic Disaster Recovery Console will indicate **Recovery job is creating drill instance for X source servers** when the drill has started.

Click **View job details** on the dialog to view the specific job for the test launch in the **Recovery job history** tab.

Successful recovery instance launch indicators

You can tell that the recovery instance launch started successfully through several indicators on the **Source servers** page.

1. The **Last recovery result** column will show the status of the recovery launch and the time of the launch. A successful recovery instance launch will show the **Successful** status. A launch that is still in progress will show the **Pending** status.
2. The launched recovery instance will appear on the **Recovery instances** page. Learn more about the Recovery instances page.
3. You can now redirect traffic from your primary systems to the launched recovery instances.

> **Note**
>
> Launch of a new recovery instance from the same source server will clean up all the previous recovery instances, regardless if they have been disconnected and deleted from DRS

Performing a failback with Elastic Disaster Recovery

Failback is the act of redirecting traffic from your recovery system to your primary system. This is an operation that is performed outside of AWS Elastic Disaster Recovery. AWS Elastic Disaster Recovery assists you in performing the failback by ensuring that the state of your primary system is up to date with the state of your recovery system.

Failback is only supported to AWS and non-AWS environments that can boot up from an ISO. For non-AWS environments which do not support ISO boot, we recommend that you convert the ISO to a suitable format. Examples - Building a disaster recovery site on AWS for workloads on Microsoft Azure and Building a disaster recovery site on AWS for workloads on Google Cloud. These blog posts are not maintained or supported by &AWS; Premium Support, and guidance for these are provided on a best effort basis.

> **Note**
>
> `arm64` source servers support failback within AWS, including reversed replication to the original Amazon EC2 instance. The Failback Client supports `x86_64` servers only. Elastic Disaster Recovery does not support failback of an `arm64` server to an on-premises environment or another cloud.

Before performing a failback, make sure that any data that was written to your failover systems during the failover is replicated back to your original systems before you perform the actual failback and before redirecting users to your primary systems. AWS Elastic Disaster Recovery helps you prepare for failback by replicating the data from your Recovery instances on &AWS; back to your source servers with the aid of the Failback Client.

Failback to on-premises environment

Using the Failback Client

Failback replication allows you to replicate data from AWS back to your original source server. To initiate this process, the Failback Client is booted directly on the source server that will receive the replicated data.

> **Note**
>
> The Failback Client supports `x86_64` servers only. For `arm64` source servers, fail back within AWS. See Performing a failback with Elastic Disaster Recovery.

Before you begin

Before starting failback replication, ensure you have completed the following:

- Meet the failback prerequisites.
- Generate failback AWS credentials.

Monitoring failback progress

Once failback replication is underway, you can track its progress in the AWS Elastic Disaster Recovery Console on the **Recovery instances** page. Learn more about the Recovery instances page.

Failback prerequisites

Prior to performing a failback, ensure that you meet all replication network requirements and these failback-specific requirements:

- Ensure that the volumes on the server you are failing back to are the same size, or larger, than the Recovery instance.
- The Failback Client must be able to communicate with the Recovery instance on TCP 1500, this can be done either via a private route (VPN/DX) or a public route (public IP assigned to the recovery instance)
- TCP Port 1500 inbound and TCP Port 443 outbound must be open on the recovery instance for the pairing to succeed.
- You must allow traffic to S3 from the server you are failing back to.

  If the server cannot reach public Amazon S3, use an Amazon S3 interface endpoint with private DNS, so that `s3.{REGION}.amazonaws.com` resolves to the endpoint. See Private DNS in the Amazon S3 User Guide.
- The server on which the Failback Client is run must have at least 4 GB of dedicated RAM.
- The recovery instance used as a source for failback must have permissions to access the DRS service via API calls. This is done using instance profile for the underlying EC2 instance. The instance profile must include the AWSElasticDisasterRecoveryRecoveryInstancePolicy in addition to any other policy you require the EC2 instance to have. By default, the launch settings that DRS creates for source servers already have an instance profile defined that includes that policy and that instance profile will be used when launching a Recovery Instance.
- Be sure to deactivate secure boot on the server on which the Failback Client is run.
- Make sure that the hardware clock on the server on which the Failback Client is run is set to UTC rather than Local Time.
- The Failback Client uses chrony for time synchronization, which requires outbound UDP port 123 connectivity to a Network Time Protocol (NTP) server. If your network firewall blocks UDP port 123, make sure that the system clock is accurate before booting the Failback Client. Time skew can cause Transport Layer Security (TLS) certificate validation failures.

Failback AWS credentials

In order to perform a failback with the Elastic Disaster Recovery Failback Client, you must first generate the required AWS credentials. You can create temporary credentials with AWS Security Token Service. These credentials are only used during Failback Client initialization.

You will need to enter your credentials into the Failback Client when prompted.

Generating temporary failback credentials

In order to generate the temporary credentials required to install the AWS Elastic Disaster Recovery Failback Client, take these steps:

1. Create a new IAM Role with the **AWSElasticDisasterRecoveryFailbackInstallationPolicy** policy.
2. Request temporary security credentials via AWS STS using the AssumeRole API. For example:

   ```
   aws sts assume-role \
     --role-arn arn:aws:iam::<account-id>:role/<role-name> \
     --role-session-name drs-failback-session
   ```

   This command returns temporary credentials consisting of an **AccessKeyId**, **SecretAccessKey**, and **SessionToken**.
3. When prompted by the Failback Client, enter:
   - **AWS Access Key ID** – the `AccessKeyId` value
   - **AWS Secret Access Key** – the `SecretAccessKey` value
   - **AWS Session Token** – the `SessionToken` value
   - **AWS Region** – the Region where your Recovery Instance resides

> **Note**
>
> Temporary credentials expire after a default session duration of 1 hour. Ensure you complete the Failback Client initialization before they expire.

Learn more about creating a role to delegate permissions to an AWS service in the IAM documentation. Attach this policy to the role: **AWSElasticDisasterRecoveryFailbackInstallationPolicy**.

Failback Client detailed walkthrough

Once you are ready to perform a failback to your original source servers or to different servers, take these steps:

> **Note**
>
> Replication from the source instance to the source server (in the target AWS Region) will continue when you perform failback on a test machine.

1. Complete the recovery as described above.
2. Configure your failback replication settings on the recovery instances you want to fail back. Learn more about failback replication settings.
3. Download the AWS Elastic Disaster Recovery Failback Client ISO (aws-failback-livecd-64bit.iso) from the S3 bucket that corresponds to the AWS Region in which your recovery instances are located.
   a. Direct download link: Failback Client ISO: `https://aws-elastic-disaster-recovery-{REGION}.s3.{REGION}.amazonaws.com/latest/failback_livecd/aws-failback-livecd-64bit.iso`
   b. Failback Client ISO hash link: `https://aws-elastic-disaster-recovery-hashes-{REGION}.s3.{REGION}.amazonaws.com/latest/failback_livecd/aws-failback-livecd-64bit.iso.sha512`
4. Boot the Failback Client ISO on the server you want to fail back to. This can be the original source server that is paired with the recovery instance, or a different server.

> **Important**
>
> Ensure that the server you are failing back to has the same number of volumes or more than the Recovery Instance and that the volume sizes are equal to or larger than the ones on the recovery instance.

> **Note**
>
> - When performing a failback **for a Linux server**, you must boot the Failback Client with the same boot mode (BIOS or UEFI) as the recovery instance that you are failing back

from. For information about checking the boot mode of the recovery instance, see Determine the boot mode of an EC2 instance in the *Amazon EC2 User Guide*.

- When performing a failback **for a Windows server**, you must boot the Failback Client with the same boot mode (BIOS or UEFI) as the Windows source server.

5. If you plan on using a static IP for the Failback Client, run the following once the Failback Client ISO boots:

   ```
   IPADDR="enter IPv4 address" NETMASK="subnet mask" GATEWAY="default
   gateway" DNS="DNS server IP address" CONFIG_NETWORK=1 /usr/bin/start.sh
   ```

   For example,

   ```
   IPADDR="192.168.10.20" NETMASK="255.255.255.0" GATEWAY="192.168.10.1"
   DNS="192.168.10.10" CONFIG_NETWORK=1 /usr/bin/start.sh
   ```

6. Enter your AWS credentials, including your **AWS Access Key ID** and **AWS Secret Access Key** that you created for Failback Client installation, the **AWS Session Token** (if you are using temporary credentials – users who are not using temporary credentials can leave this field blank), and the **AWS Region** in which your Recovery instance resides. You can attach the Elastic Disaster Recovery Failback Client credentials policy to a user or create a role and attach the policy to that role to obtain temporary credentials. Learn more about Elastic Disaster Recovery credentials.



7. Enter the custom endpoint or press Enter to use the default endpoint. You should enter a custom endpoint if you want to use a VPC Endpoint (PrivateLink).



8. If you are failing back to the original source machine, the Failback Client will automatically choose the correct corresponding recovery instance.



9. If the Failback Client is unable to automatically map the instance, then you will be prompted to select the recovery instance to fail back from. The Failback Client displays a list with all recovery instances. Select the correct recovery instance by either entering the numerical choice from the list that corresponds to the correct recovery instance or by typing in the full recovery instance ID.

```
Failback Client automated instance detection...
Cannot automatically detect the Recovery Instance that matches the configuration of this server, asking for manual instance mapping.
Which Recovery Instance would you like to fail back from? Select a number from the list or enter the full Recovery Instance ID (e.g: i-xxxx).
1: i-08
Enter input: 1
Recovery instance i-08b                matched with Failback Client 4221c
```

> **Note**
>
> The Failback Client will only display recovery instances whose volume sizes are equal to or smaller than the volume sizes of the server you're failing back to. If the recovery instance has volume sizes that are larger than that of the server you are failing back to, then these Recovery instances will not be displayed.

10. If you are failing back to the original source server, then the Failback Client will attempt to automatically map the volumes of the instance.

```
Trying to get recovery volumes for i-02
Local devices:
        /dev/sda 8.0 GB
Remote devices:
        /dev/xvda 8.0 GB
```

11. If the Failback Client is unable to automatically map the volumes, you will need to manually enter a local block device (example /dev/sdg) to replicate to from the remote block device. Enter the EXCLUDE command to specifically exclude Recovery Instance volumes from replication.

    Optionally, you can also enter the complete volume mapping in the same CSV or JSON format used by --device-mapping Failback Client argument. For example: `ALL="/dev/nvme2n1=/dev/sda,/dev/nvme0n1=EXCLUDE, . . ."`.

    The full volume mapping should be provided as single CSV or JSON line in the format of --device-mapping Failback Client argument.

    Learn more about using --device-mapping program argument

```
Cannot detect volume mapping automatically, asking for manual volume mapping
Enter local block device (e.g., /dev/sdg) to replicate from remote block device /dev/xvda or EXCLUDE to exclude: /dev/sda
Successfully mapped volumes
```

> **Important**
>
> The local volumes must be the same in size or larger than the recovery instance volumes. The valid special case is when original local volume has fractional GiB size (e.g. 9.75 GiB). Then the recovery instance volume size will be larger because of rounding to nearest GiB (e.g. 10 GiB).

12. The Failback Client will verify connectivity between the recovery instance and AWS Elastic Disaster Recovery.

```
Verifying Recovery Instance connectivity to the Dirrus service...
Connectivity established
```

13. The Failback Client will download the replication software from a public S3 bucket onto the source server.

```
Downloading AWS Replication Software [113 MB]. This may take a few minutes, please wait...
Downloaded the AWS Replication Software successfully
```

> **Important**
>
> You must allow traffic to S3 from the source server for this step to succeed.

14. The Failback Client will configure the replication software.

```
Configuring AWS Replication Software...
Finished AWS Replication Software configuration
```

15. The Failback Client will pair with the AWS Replication Agent running on the recovery instance and will establish a connection.

```
Pairing the AWS Replication Agent running on the Recovery Instance with the Failback Client...
Pairing completed
Establishing a connection between the Failback Client and the AWS Replication Agent running on the Recovery Instance over port 1500...
```

> **Important**
>
> TCP Port 1500 inbound must be open on the recovery instance for the pairing to succeed.

16. Data replication will begin.

```
Connection established. Replication in progress...
```

You can monitor data replication progress on the **Recovery instances** page in the AWS Elastic Disaster Recovery Console.

17. Once data replication has been completed, the Recovery instance on the **Recovery instances** page will show the **Ready** status under the **Failback state** column and the **Healthy** status under the **Data replication status** column.

18. Once all of the recovery instances you are planning to fail back show the statuses above, select each Instance and choose **Failback**. This will stop data replication and will start the conversion process. This will finalize the failback process and create a replica of each recovery instance on the corresponding source server.

Select one or more recovery instances that are in the **Ready** state and click **Failback** to continue the failback process after performing a failback with the Elastic Disaster Recovery Failback Client. This action will stop data replication and will start the conversion process. This will finalize the failback process and will create a replica of each recovery instance on the corresponding source server.

When the **Continue with failback for X instances** dialog appears, click **Failback**.

This action will create a Job, which you can follow on the **Recovery job history** page. Learn more about the recovery job history page.

19. Once the failback is complete, the Failback Client will show that the failback has been completed successfully. You can reboot the server and check that it has the needed data, before proceeding.

> **Note**
>
> The server client iso should not be in the boot order when you want to recover into the original OS.

20. You can opt to either terminate, delete, or disconnect the Recovery instance. Learn more about each action.

Failback Client program arguments

The arguments supported by Failback Client LiveCD process are:

- --aws-access-key-id AWS_ACCESS_KEY_ID
- --aws-secret-access-key AWS_SECRET_ACCESS_KEY
- --aws-session-token AWS_SESSION_TOKEN
- --region REGION
- --endpoint ENDPOINT
- --default-endpoint
- --recovery-instance-id RECOVERY_INSTANCE_ID
- --dm-value-format {dev-name,by-path,by-id,by-uuid,all-strict}
- --device-mapping DEVICE_MAPPING] [--no-prompt
- --log-console

- --log-file LOG_FILE

All arguments are optional.

[--device-mapping DEVICE_MAPPING]

`--device-mapping` argument will skip mapping auto-detection and manual mapping and use the mapping provided in this parameter.

There are three formats supported:

1. Classic CE format of key-value CSV string as one line.

   You may use either ":" or "=" as CSV fields separator which is more suitable for Windows drive letters. Examples are:

   ```
   recovery_device1=local_device1,recovery_device2=local_device2,recovery_device3=EXCLUDE, . . .
   ```

   ```
   recovery_device1:local_device1,recovery_device2:local_device2, . . .
   ```

2. JSON format:

   ```
   '{"/dev/xvdb":"/dev/sdb","/dev/xvdc":"/dev/sdc","recovery_device3":"local_device3"}'
   ```

3. JSON list DRS API format:

   ```
   '[{"recoveryInstanceDeviceName": "recovery_device1","failbackClientDeviceName":
    "local_device1"},{"recoveryInstanceDeviceName" . . .: }]'
   ```

No matter which format you choose, you need to provide either valid Failback Client device name or EXCLUDE for each Recovery Instance device.

[dm-value-format DM_VALUE_FORMAT]

`--dm-value-format` allows to use Failback Client persistent block devices identifiers in --device-mapping argument.

Such persistent identifiers will always refer to the same block devices after Failback Client reboot.

Possible --dm-value-format choices are:

1. "dev-name" - default format for using /dev/sda, /dev/xvda, /dev/nvme3n1 etc
2. "by-path" - from ls -l /dev/disk/by-id/ e.g. pci-0000:00:10.0-scsi-0:0:3:0, pci-0000:00:1e.0-nvme-1, pci-0000:02:01.0-ata-1, xen-vbd-768 etc
3. "by-id" - from ls -l /dev/disk/by-id/ e.g. device serial numbers
4. "by-uuid" - UUIDs from ls -l /dev/disk/by-uuid/
5. "all-strict" - all of the above mixed

We will use the example of SCSI identifiers from the command output below:

```
# root@ubuntu:~# ls -l /dev/disk/by-path/
total 0
lrwxrwxrwx 1 root root  9 Jun 27 12:25 pci-0000:00:10.0-scsi-0:0:0:0 -> ./../sda
lrwxrwxrwx 1 root root 10 Jun 27 12:25 pci-0000:00:10.0-scsi-0:0:0:0-part1 -> ../../
sda1
lrwxrwxrwx 1 root root  9 Jun 27 12:25 pci-0000:00:10.0-scsi-0:0:1:0 -> ../../sdb
lrwxrwxrwx 1 root root  9 Jun 27 12:25 pci-0000:00:10.0-scsi-0:0:2:0 -> ../../sdc
lrwxrwxrwx 1 root root  9 Jun 27 12:25 pci-0000:00:10.0-scsi-0:0:3:0 -> ../../sdd
```

To use block device SCSI identifiers like 'pci-0000:00:10.0-scsi-0:0:0:0' you need to add to command line:`--dm-value-format by-path`

The examples of valid --device-mapping for `--dm-value-format by-path` are:

```
/dev/nvme2n1=pci-0000:00:10.0-scsi-0:0:0:0,/dev/nvme0n1=pci-0000:00:10.0-scsi-0:0:1:0,/
dev/nvme3n1=pci-0000:00:10.0-scsi-0:0:2:0...
```

```
'{"/dev/nvme2n1":"pci-0000:00:10.0-scsi-0:0:0:0","/dev/nvme0n1":"pci-0000:00:10.0-
scsi-0:0:1:0","/dev/nvme3n1":"pci-0000:00:10.0-scsi-0:0:2:0", . . .}'
```

No matter which format you choose, you need to provide either valid Failback Client device name or EXCLUDE for each Recovery Instance device.

Performing a failback with the DRS Mass Failback Automation Client

The DRS Mass Failback Automation Client (DRSFA client) is a command-line tool that automates the failback process for vCenter source servers. Instead of manually initiating failback for each Recovery instance individually, the DRSFA client orchestrates failback across multiple machines

simultaneously — handling ISO attachment, network configuration, and replication initiation automatically.

The DRSFA client supports two failback modes:

- **One-click failback** — Fails back all Recovery instances in your account using automatic configuration. Best for environments where the default network and device settings are acceptable.
- **Custom failback** — Generates a configuration file that you can edit to set specific network settings, device mappings, and other parameters for each machine before initiating failback. Best for environments requiring per-machine control.

The client runs on a dedicated Ubuntu host that has network access to both your vCenter environment and AWS. It communicates with the AWS Elastic Disaster Recovery API to identify Recovery instances and with vCenter to attach ISOs and manage VMs during the failback process.

To get started, complete the following steps in order:

1. Verify that you meet the prerequisites.
2. Install the DRSFA client on your Ubuntu host.
3. Generate IAM credentials for the client.
4. Run the client and choose your failback mode.

DRSFA prerequisites

Before using the DRSFA client, verify that you meet the following requirements.

> **Note**
>
> The DRSFA client works only with vCenter source servers. It was tested on vCenter versions 6.7, 7.0, 8.0, and 9.0.

You can fail back all Recovery instances in a single AWS Region simultaneously with the DRSFA client, as long as your vCenter hardware supports the failback load.

Network and AWS prerequisites

- Meet all network requirements for replication.
- Initialize AWS Elastic Disaster Recovery in your account.
- Open inbound TCP port 1500 on the Recovery instance in AWS.
- The Recovery instance used as a source for failback must have permissions to access AWS Elastic Disaster Recovery through API calls. Attach an instance profile that includes the AWSElasticDisasterRecoveryRecoveryInstancePolicy policy. By default, the launch settings that DRS creates for source servers already include an instance profile with this policy.

DRSFA client host requirements

The DRSFA client host must meet the following requirements:

- At least 4 GB of RAM
- Ubuntu 22.04, 24.04, or 26.04 (x86_64)
- Python 3.13 with pip installed
- Network connectivity to your vCenter environment

The following system packages are required. The installer attempts to install them if they are not already present:

```
build-essential curl genisoimage git libbz2-dev libffi-dev
liblzma-dev libncurses5-dev libncursesw5-dev libreadline-dev
libsqlite3-dev libssl-dev llvm make tk-dev unzip wget
xz-utils zlib1g-dev
```

For required Python libraries, see the `requirements.txt` file at `https://drsfa-us-west-2.s3.us-west-2.amazonaws.com/requirements.txt`. These libraries are installed automatically by the DRSFA client.

vCenter requirements

- Each server being failed back must have at least 4 GB of RAM.
- Each server being failed back must have the hardware clock set to UTC (not Local Time).

- Each vCenter source server must have two CD-ROM devices with IDE controllers attached — one for the DRS Failback Client and one for `drs_failback_automation_seed.iso`. If no attached CD-ROM devices are found, the DRSFA client attempts to add them.
- Upload the DRS Failback Client ISO to your vCenter Datastore. We recommend using the latest version. Download the latest DRS Failback Client and upload it to your datastore.
- The vCenter credentials must include the following permissions:

Category	Required permissions
Virtual machine	Change Settings, Guest operation queries, Guest operation program execution, Connect devices, Power off, Power on, Add or remove device, Configure CD media
Datastore	Browse datastore

- Constrain vCenter credentials to only the VMs you plan to fail back.

Recommendations

- Run SHA-512 checksum verification on the DRS Failback Client ISO before use. Verify the checksum at: `https://aws-elastic-disaster-recovery-hashes-{REGION}.s3.amazonaws.com/latest/failback_livecd/aws-failback-livecd-64bit.iso.sha512`
- Run SHA-512 checksum verification on `drs_failback_automation_seed.iso` before use. The hash is exported to `drs_failback_automation_seed.iso.sha512` when the seed ISO is created.
- Run the DRSFA client with low privileges. The client does not require root access.
- Follow the least privilege principle for the folder where JSON output and ISO files are stored.

Security best practices

Follow these security best practices when using the DRSFA client:

- Always use the latest version of the DRSFA client. The client automatically checks for updates at startup.

- Verify DRSFA client hashes manually when automatic hash verification is not performed. The hash verification hint appears during installation.
- Do not grant additional permissions to the DRSFA client beyond those listed in the prerequisites.
- Follow the AWS recommended password policy when setting the password for the VM that hosts the DRS Failback Client.
- Restrict access to the vCenter environment to trusted administrators only. The DRSFA client treats the executing user and anyone with datastore access as a single trust entity.

Installing the DRSFA Client

Installing the DRSFA client is a one-time operation.

The DRSFA client is supported on Ubuntu 22.04, 24.04, and 26.04. Launch an Ubuntu 22.04, 24.04, or 26.04 x86_64 instance in EC2 or use a public ISO to run the client locally in your vCenter environment. Follow the Create your EC2 resources and launch your EC2 instance guidelines in the EC2 documentation.

After your instance is ready, connect to it and download the DRSFA client installer:

```
wget https://drsfa-us-west-2.s3.us-west-2.amazonaws.com/
drs_failback_automation_installer.sh
```

> **Note**
>
> Verify the hash of the installer: `https://drsfa-hashes-us-west-2.s3.us-west-2.amazonaws.com/drs_failback_automation_installer.sh.sha512`

Run the installation script:

```
bash drs_failback_automation_installer.sh
```

> **Note**
>
> This command might ask for a sudo password if you use the Ubuntu ISO. Enter the password but **do not** run this command as sudo.

After installation completes, reload your shell profile:

```
source ~/.profile
```

The DRSFA client is installed in the `drs_failback_automation_client` directory. You can optionally delete the installer:

```
rm drs_failback_automation_installer.sh
```

Generating the seed ISO

Generate a `drs_failback_automation_seed.iso` file that contains the password for the Failback Client VM. Upload this file to your datastore.

```
bash drs_failback_automation_seed_creator.sh
```

Enter a unique password that follows the AWS recommended password policy.

Two files are generated: `drs_failback_automation_seed.iso` and `drs_failback_automation_seed.iso.sha512`. Upload the seed ISO to the same datastore where the DRS Failback Client ISO is stored.

After generating the seed ISO, you can optionally delete the seed creator:

```
rm drs_failback_automation_seed_creator.sh
```

Generating IAM credentials and configuring CloudWatch logging

The DRSFA client requires AWS credentials to operate.

> **Important**
>
> Temporary credentials are the recommended option. They do not require rotation, cannot be reused after expiration, and provide enhanced security. You can specify how long they remain valid, up to a maximum limit.

Temporary credentials

To create temporary credentials:

1. Create a new IAM Role with the AWSElasticDisasterRecoveryFailbackInstallationPolicy policy.
2. Request temporary security credentials through AWS STS using the AssumeRole API.

Configuring CloudWatch logging

After generating credentials, create a CloudWatch log group named **DRS_Mass_Failback_Automation**. If this log group does not exist or has the wrong name, the DRSFA client still works but does not send logs to CloudWatch. Learn more about working with log groups in the Amazon CloudWatch Logs documentation.

Running the DRSFA client

Navigate to the `drs_failback_automation_client` directory, set the required environment variables, and run the client.

The following table describes the environment variables:

Variable	Required	Description
`AWS_REGION`	Yes	The AWS Region where your Recovery instances are located.
`AWS_ACCESS_KEY`	Yes	The AWS access key for the DRSFA client.
`AWS_SECRET_ACCESS_KEY`	Yes	The AWS secret access key for the DRSFA client.
`AWS_SESSION_TOKEN`	Conditional	Required when using temporary credentials.
`DRS_FAILBACK_CLIENT_PASSWORD`	Yes	The password you set in the `drs_failback_automation_seed.iso` file.
`VCENTER_HOST`	Yes	The IP address of the vCenter host.
`VCENTER_PORT`	Yes	The vCenter port (usually 443).
`VCENTER_USER`	Yes	The vCenter username (for example, `admin@vsphere.local`).
`VCENTER_PASSWORD`	Yes	The vCenter password.

Variable	Required	Description
VCENTER_DATASTORE	Yes	The datastore where the Failback Client ISO and seed ISO are stored.
VCENTER_FAILBACK_CLIENT_PATH	Yes	Path to aws-failback-livecd-64bit.iso in the datastore.
VCENTER_SEED_ISO_PATH	Yes	Path to drs_failback_automation_seed.iso in the datastore.
DISABLE_SSL_VERIFICATION	No	Set to true to deactivate SSL verification. Active by default.
THREAD_POOL_SIZE	No	Number of servers to process concurrently. Default is 10.

You can set the variables inline or export them individually.

Option 1: Export variables individually

```
export AWS_REGION=us-west-2
export AWS_ACCESS_KEY=XXXX
export AWS_SECRET_ACCESS_KEY=XXXX
export AWS_SESSION_TOKEN=XXXX
export DRS_FAILBACK_CLIENT_PASSWORD=XXXX
export VCENTER_HOST=10.0.1.100
export VCENTER_PORT=443
export VCENTER_USER=admin@vsphere.local
export VCENTER_PASSWORD=XXXX
export VCENTER_DATASTORE=Datastore1
export VCENTER_FAILBACK_CLIENT_PATH='path/aws-failback-livecd-64bit.iso'
export VCENTER_SEED_ISO_PATH='path/drs_failback_automation_seed.iso'

python3 drs_failback_automation_init.pyc
```

Option 2: Inline variables

```
AWS_REGION=us-west-2 AWS_ACCESS_KEY=XXXX AWS_SECRET_ACCESS_KEY=XXXX \
DRS_FAILBACK_CLIENT_PASSWORD=XXXX VCENTER_HOST=10.0.1.100 VCENTER_PORT=443 \
```

```
VCENTER_USER=admin@vsphere.local VCENTER_PASSWORD=XXXX \
VCENTER_DATASTORE=Datastore1 \
VCENTER_FAILBACK_CLIENT_PATH='path/aws-failback-livecd-64bit.iso' \
VCENTER_SEED_ISO_PATH='path/drs_failback_automation_seed.iso' \
python3 drs_failback_automation_init.pyc
```

One-click failback

After the client connects and completes verification, the main menu displays:

```
What would you like to do?
1. One-Click Failback
2. Custom Failback
3. Generate a default failback configuration file
4. Find servers in vCenter
5. Exit
```

Select **1** for One-Click Failback.

1. Enter a custom prefix for the results output file.
2. If failback replication has already started for some Recovery instances, the client prompts you to skip those instances or restart replication for them.
3. The client lists the Recovery instances in your AWS account. Enter **Y** to continue.
4. The client initiates failback. Monitor progress on the **Recovery instances** page in the DRS console.

Failback results

After the failback completes, the client displays a summary showing how many servers succeeded and how many failed.

The full results are exported as a JSON file to `/drs_failback_automation_client/results/Failback` with the naming convention: `{prefix}_{account_id}_{region}_{timestamp}.json`

The JSON file contains the following fields for each server:

- The AWS ID of the Recovery instance
- The status of the failback (succeeded, skipped, or failed)

- A message (provides the cause for failure if applicable)
- The vCenter VM UUID
- The vCenter UUID of the original source server

If failback failed for any machines, review the `failback_hosts_settings.json` file in the same folder to see the exact configuration used. Fix any problems and use the custom failback flow to retry those specific machines.

Custom failback

The custom failback option provides more control over the failback process. You first generate a configuration file, edit settings for individual machines, and then perform failback using that file.

Find servers in vCenter

Before configuring custom failback, use the **Find servers in vCenter** menu option to discover the disks and volumes of your vCenter machines. This information is needed when configuring custom device mapping.

You can enter a name filter or press Enter to list all machines. The results include the following details for each server:

- Name
- UUID
- Disk and volume information

Results are exported to the `Results/VMFinder` folder with the naming convention: `{vcenter_host}_{timestamp}.txt`

Generating the configuration file

You can create a custom configuration JSON file manually or generate a default configuration file through the client.

To generate a default file, select **Generate a default failback configuration file** from the main menu. Enter a custom prefix for the file name. The configuration file is created as a JSON file in the `/drs_failback_automation_client/Configurations/` folder with the name: `{prefix}_{account_id}_{region}.json`

The file contains these fields for each machine:

- NETMASK
- VCENTER_MACHINE_UUID
- PROXY
- DNS
- CONFIG_NETWORK
- IPADDR
- GATEWAY
- SOURCE_SERVER_ID
- DEVICE_MAPPING

Edit any field to control the failback configuration for each machine, then save your changes.

Note

- Set CONFIG_NETWORK to "DHCP" for DHCP or "STATIC" for manual network configuration. When set to "DHCP", the DNS, IPADDR, GATEWAY, NETMASK, and PROXY parameters are ignored but must not be deleted.
- If you are not using a proxy server, leave the PROXY field as an empty string. Do not remove it.
- If a source server does not have an attached recovery instance, the **SOURCE_SERVER_ID** field is empty.

Custom device mapping parameter

The DEVICE_MAPPING field is passed to the LiveCD failback process as the `--device-mapping` argument. Learn more about using the --device-mapping program argument.

Use the Find servers in vCenter utility to discover disk names for your machines.

Three formats are supported:

1. Classic CE format — a key-value CSV string on one line. Use either ":" or "=" as the field separator:

```
"DEVICE_MAPPING":
 "recovery_device1=local_device1,recovery_device2=local_device2,recovery_device3=EXCLUDE"
```

```
"DEVICE_MAPPING": "recovery_device1:local_device1,recovery_device2:local_device2"
```

2. JSON format:

```
"DEVICE_MAPPING": {
    "/dev/xvdb":"/dev/sdb",
    "/dev/xvdc":"/dev/sdc",
    "recovery_device3":"local_device3"
}
```

3. JSON list DRS API format:

```
[
    {
        "recoveryInstanceDeviceName": "recovery_device1",
        "failbackClientDeviceName": "local_device1"
    },
    {
        "recoveryInstanceDeviceName": "recovery_device2",
        "failbackClientDeviceName": "local_device2"
    }
]
```

Regardless of format, provide either a valid Failback Client device name or EXCLUDE for each Recovery Instance device.

Performing the custom failback

After editing your configuration file, rerun the DRSFA client and select **Custom Failback** from the main menu.

1. Select your configuration file. You can define a custom path or use the default path displayed by the client.
2. Enter a custom prefix for the results output file.
3. If failback replication has already started for some Recovery instances, choose whether to skip those instances or restart replication.

4. The client lists the Recovery instances to be failed back. Enter **Y** to continue.
5. The client initiates failback. Monitor progress on the **Recovery instances** page in the DRS console.

Results are exported in the same format described in Failback results.

Upgrading the DRSFA Client

Most DRSFA components upgrade automatically on execution. If the client displays a message requiring a manual upgrade, complete these steps:

1. Navigate to the directory where the original installation took place.
2. Download the DRSFA installer:

```
wget https://drsfa-us-west-2.s3.us-west-2.amazonaws.com/
drs_failback_automation_installer.sh
```

> **Note**
>
> Verify the installer hash: `https://drsfa-hashes-us-west-2.s3.us-west-2.amazonaws.com/drs_failback_automation_installer.sh.sha512`

3. Run the installer:

```
bash drs_failback_automation_installer.sh
```

4. Remove the installer:

```
rm drs_failback_automation_installer.sh
```

Troubleshooting the DRSFA Client

The following table describes common issues and resolutions when using the DRSFA client.

Issue	Possible cause	Resolution
Cannot connect to vCenter	vCenter host is unreachable from the	Verify network connectivity between the DRSFA client host and vCenter.

Issue	Possible cause	Resolution
	DRSFA client host, or the specified port is blocked.	Confirm that the `VCENTER_PORT` (usually 443) is open.
Replication fails to start	TCP port 1500 is blocked on the Recovery instance.	Verify that inbound TCP port 1500 is open on the Recovery instance security group in AWS.
Permission errors on vCenter operations	The vCenter credentials lack required permissio ns.	Confirm that the vCenter user has all required permissions listed in the prerequisites (Virtual machine and Datastore permissions).
AWS API permission errors	The IAM credentials lack the required policy.	Verify that the IAM role or user includes the AWSElasticDisasterRecoveryF ailbackInstallationPolicy policy.
CD-ROM attachment failures	The VM does not have two available IDE CD-ROM slots, or the datastore path is incorrect.	Verify that the VM has available IDE controller slots. Confirm that `VCENTER_FAILBACK_CLIENT_PAT H` and `VCENTER_SEED_ISO_PATH` are correct.
Replication stalls or fails	Network connectivity issues between the source server and the Recovery instance.	Verify that the source server can reach the Recovery instance on TCP port 1500. Check the `failback.log` on the source VM for details.

Log locations

- **DRSFA client log**: `drs_failback_automation_client/drs_failback_automation.log` on the host where the client runs.
- **Per-server failback log**: `failback.log` on the individual source VM being failed back.
- **CloudWatch**: If configured, logs are sent to the `DRS_Mass_Failback_Automation` log group.

Using the failback client to perform a failback to the original source server

When using the failback client, you can fail back to the original source server or a different source server using AWS Elastic Disaster Recovery.

To ensure that the original source server has not been deleted and still exists, check its status in the AWS DRS console. Source servers that have been deleted or no longer exist will show as having **Lag** and being **Stalled**.

> **Note**
>
> After failing back to the original source server, you don't need to reinstall the DRS agent to start replication back to AWS.

When you fail back to the original source server, replication includes only the delta changes made after recovery. As a result, the failback completes faster. When failing back to a new server that was not the original source, the Failback Client fully synchronizes all data. This process takes longer depending on the total volume size.

If the original source server is healthy and you decide to fail back to it, it will undergo a rescan until it reaches the **Ready** status.

You can tell whether you are failing back to the original or a new source server in the recovery instance details view under **Failback status**.

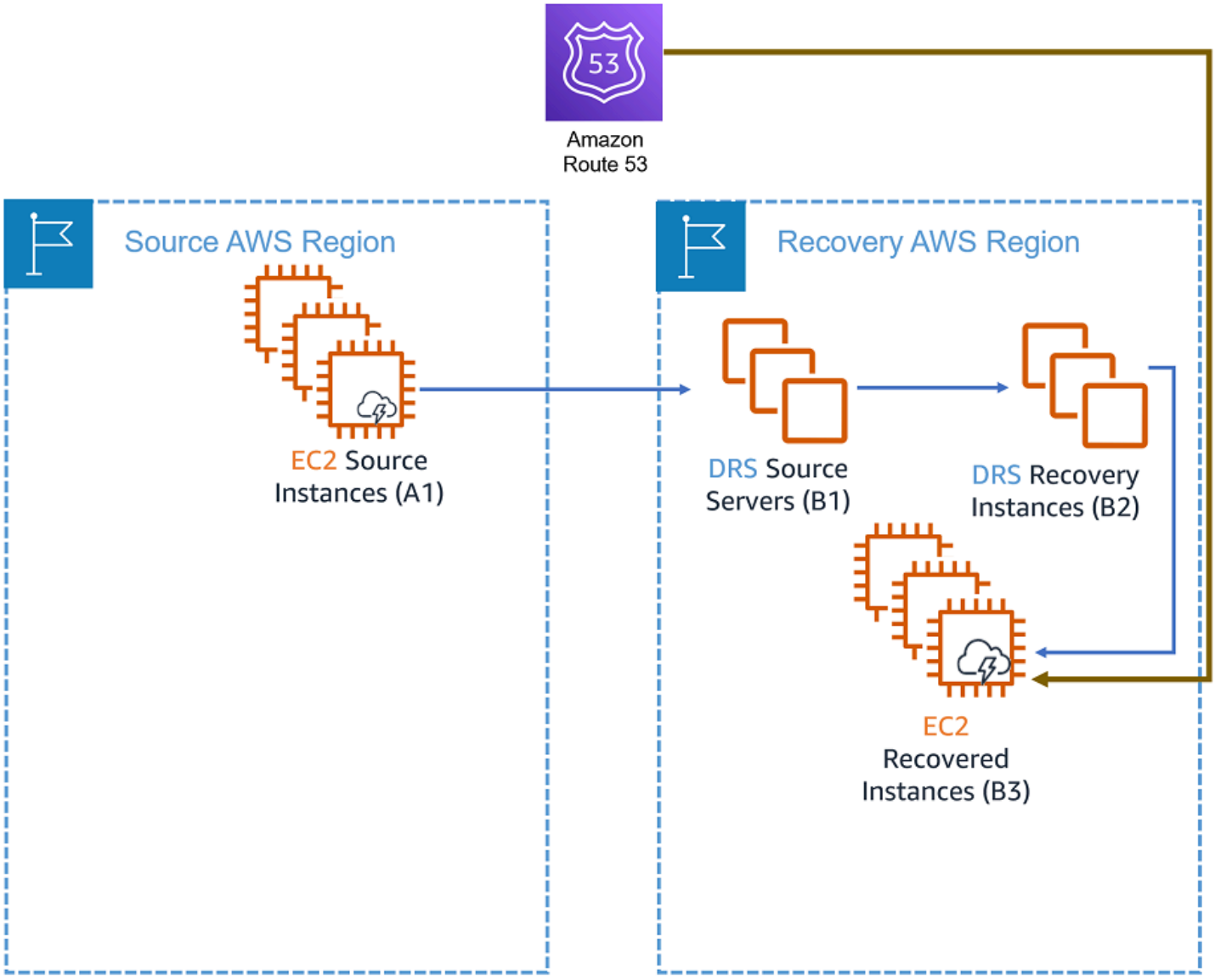
Performing a cross-Region failback

AWS Elastic Disaster Recovery (AWS DRS) allows you to perform failover and failback on your EC2-based applications from one AWS Region to another AWS Region. The failover process is the same as failing over into an AWS Region from a source outside of AWS, but the failback process is different. The instructions below describe the complete cross-Region failover and failback process. In the examples, we use us-east-1 as the source AWS Region and us-east-2 as the recovery AWS Region, but any combination of AWS Regions that are supported by DRS will work.

> **Note**
>
> Cross-Partition failback features between commercial, and AWS GovCloud partitions are not supported. Cross-Region failback features within the AWS GovCloud partition are available between AWS GovCloud Regions (us-gov-west-1 and us-gov-east-1)

Overview and prerequisites

The failback process starts after the failover process ends. During failover, AWS DRS allows you to replace the EC2 source instance (A1) with the EC2 recovered instance (B3). The current AWS resource state is illustrated in this diagram:



After performing a recovery, your applications are running on EC2 instances in the recovery region. However, these recovered instances (marked B3 in the diagram above) are not protected against other potential outages. In order to avoid data loss, you should start a reversed replication immediately. Starting reversed replication involves copying the data from the EC2 recovered instances (B3) to the original region, an operation that takes time and incurs cross-Region data transfer costs.

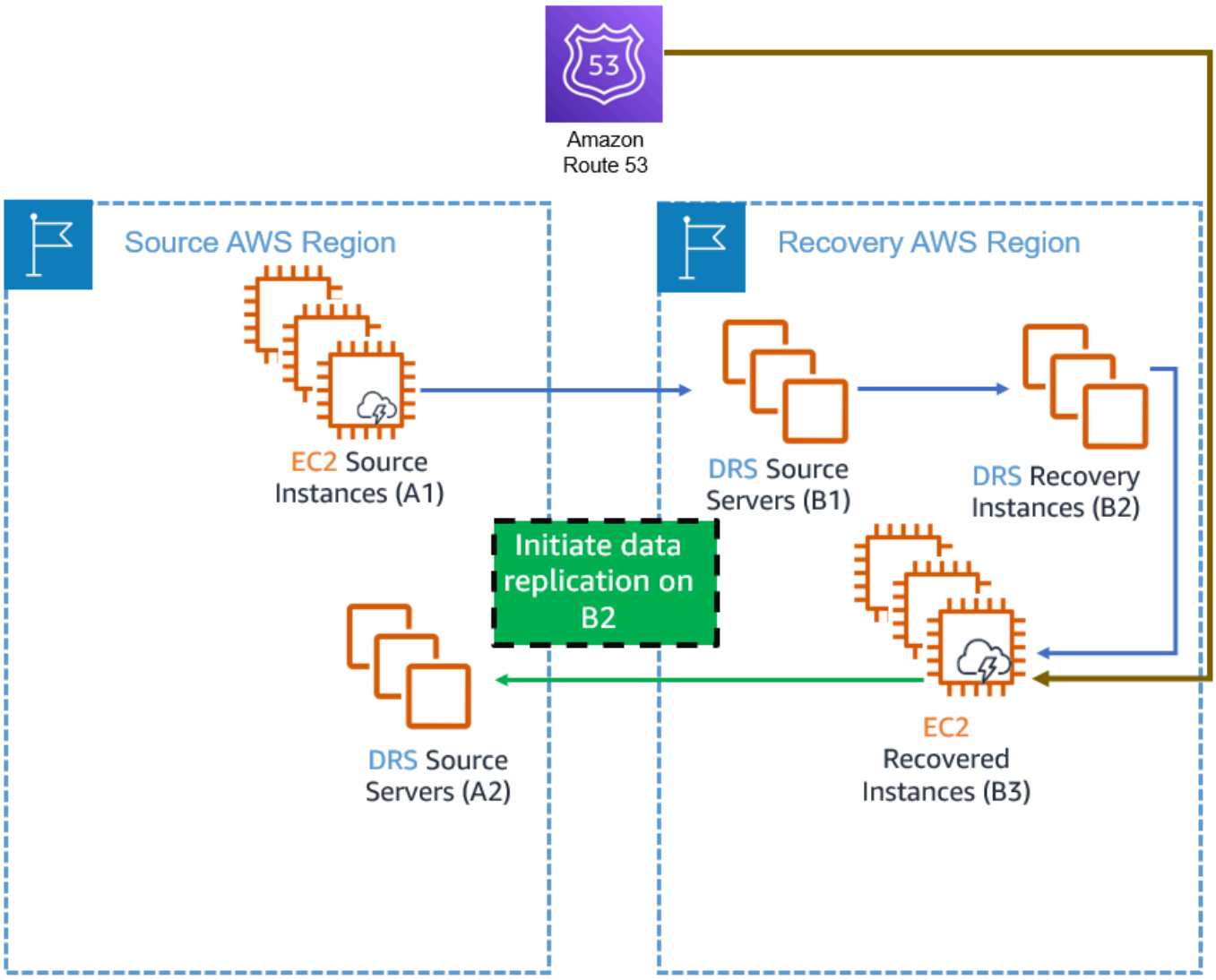
Once replication has reached a healthy state, failing back to the source region is possible using the DRS console on that region, assuming DRS has been initialized in the source region.

Important

- To ensure operational continuity, initialize the AWS DRS in advance in both the source and target AWS Regions, and conduct regular failover and failback drills.
- Before starting a failback, make sure the EC2 recovered instances (B3) have a network interface while meeting the specified network requirements.
- Access to EC2 instance metadata is required. If you have a custom network setup that modifies the operating system route, ensure that access to metadata is intact. Learn how to verify metadata access for Linux and for Windows.
- EC2 Instances that have failed over must resolve via DNS the regional DRS endpoint of the failback region. The resolved endpoint must be accessible from the EC2 Instance via TCP 443.

Performing cross-region failback

1. **Start reversed replication.**
   a. Go to the recovery AWS Region (in this example, us-east-2).
   b. Choose the **AWS Elastic Disaster Recovery** service.
   c. Navigate to the **Recovery instances** page.
   d. Select the servers that you want to protect and click **Start reversed replication**.
   e. A Source server (A2) will be created in the source region, as shown in this diagram.



Note

All server data is transferred over the wire during this step. This process could take some time and will result in cross-Region data transfer costs. Moreover, starting reversed replication creates additional replication resources (A2). To avoid double billing, you can stop replicating the source instances (A1) by navigating to the AWS DRS source server in the recovery region (B1) and clicking **Stop replication** in the replication drop-down menu. Make sure that you only stop the replication after validating the failover instances because once replication is stopped, all previous points in time are deleted.

> **Important**
>
> Once replication is stopped, all previous points in time are deleted. This is done to minimize costs.

2. **Launch, validate, and redirect traffic.**

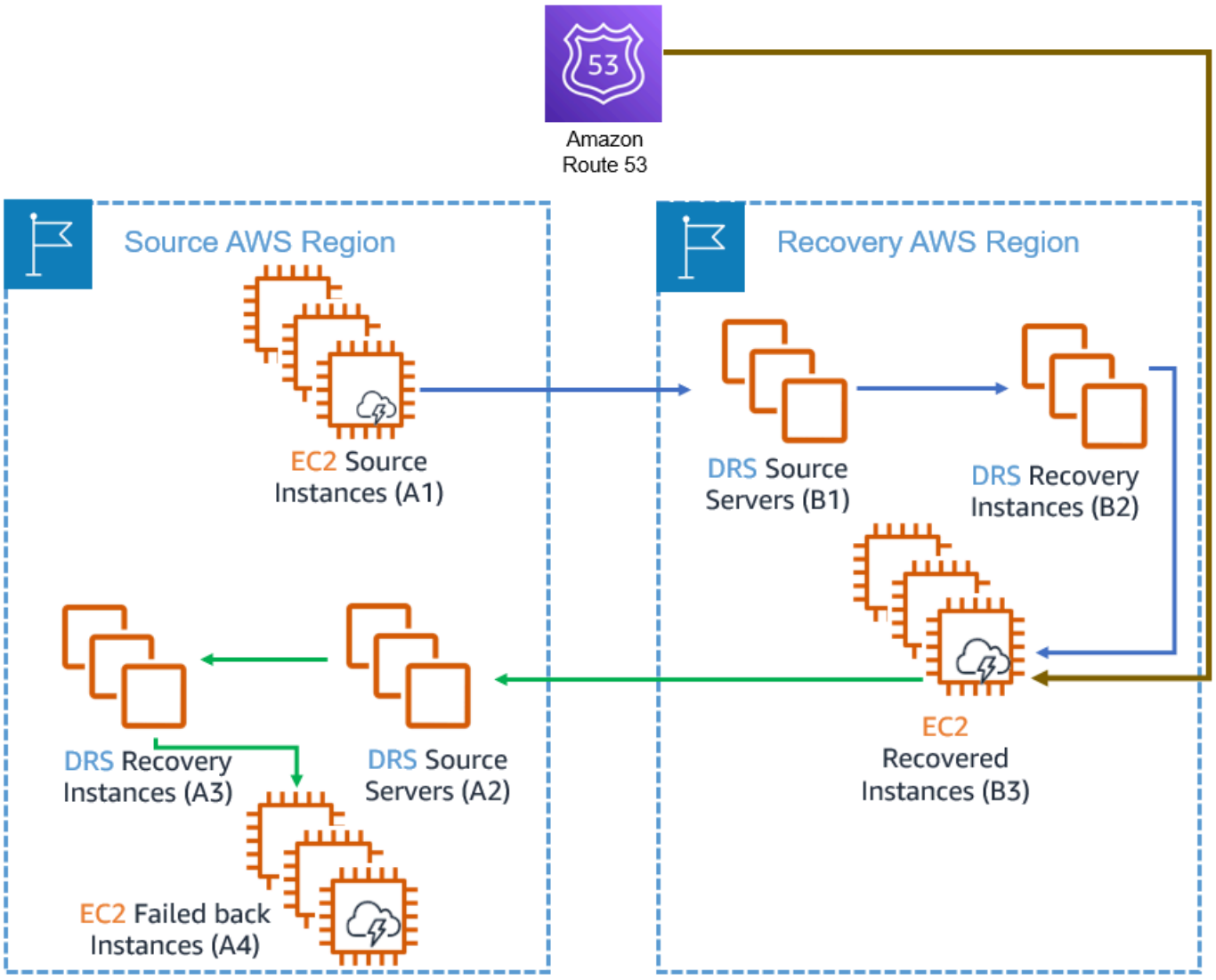
   After the **Reversed direction launch state** is marked as **Ready**, take these steps to complete the failback:

   a. Find the relevant source servers (A2) in the source region by clicking the **Replicating to source server** link in the recovery instance (B2).

   > **Note**
   >
   > You can also find it directly on the **Source servers** page in AWS DRS console at the source region.

   b. If the state is **Ready** (or **Ready with lag**), click **Launch for failback** under **Initiate recovery job**.

   > **Important**
   >
   > Make sure that your applications (A4) are working as expected. If you run into any issues, you can relaunch the instances and try again. Until you opt to failback, your recovery instances (B3) will continue to run in your recovery AWS Region to ensure business continuity.



c. Redirect traffic to failed back instances (A4), which will now become your new primary instances. Traffic redirection is not conducted using DRS. Choose a service according to your preferences (consider using Amazon Route 53).

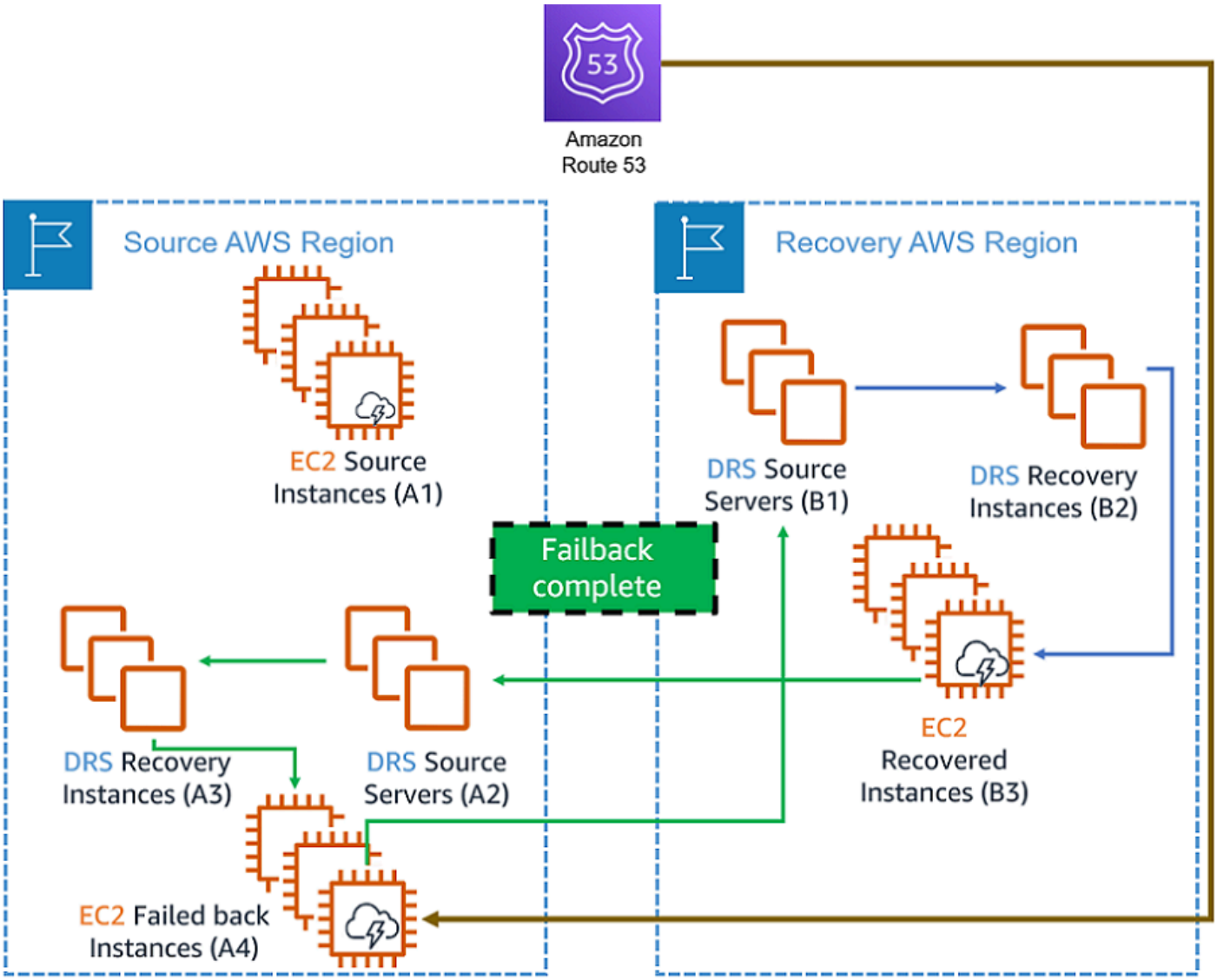
3. **Protect your new failed back instances.**

> **Important**
>
> Do not perform this step when performing a drill. This step replaces the instances that AWS DRS replicates (from the Source instances, A1, to the failed back instances, A4). In a drill, the source instances (A1) are still your production environment.

The newly launched failed-back instances (A4) are not protected. In order to protect them, follow these steps:

a. Navigate to the recovery instance (A3) in the source region.

b. Click **Start reversed replication**. This step will replace the Instances that the Source Server (B1) protects (A4 instead of A1).



4. **Clean your environment.**

After the failover to failback cycle is complete, you may be left with multiple AWS resources that you no longer need and that are costly to maintain. These include the source and failover EC2 instances (A1,B3), the recovery instances (B2, A3), and the Source servers (A2). Consider removing them.

Cleanup steps:

a. **Stop replication on the source servers (A2) of the source region.**

Navigate to the source server in the source region (A2), and click on **Stop replication** under the **Replication menu**. This step is required before terminating the recovery instance (B2).

b. **Terminate the recovery instances (B2).**

These instances, launched in your recovery AWS Region, are no longer needed now that you have launched new primary instances in your original source AWS Region. To terminate these instances, navigate to the AWS DRS Console in your recovery AWS Region (B2). After termination, those instances will no longer appear in the **Recovery Instances** page of the DRS Console. This process also terminates the recovered EC2 instances (B3).

c. **Terminate the source region EC2 instances (A1).**

These have now been replaced by the new instances launched in step 2 above (EC2 failed back instances, A4). You might have stopped these instances after the failover, and you can now terminate them using the AWS EC2 Console.

d. **Remove the recovery instance (A3) in the source region.**

Navigate to the **Recovery instances** in the AWS DRS console. Select the relevant recovery instance and click **Delete server** under the **Action** drop-down menu.

> **Note**
>
> If you have started reversed replication for the recovery instance (A3), you will not be able to disconnect it. To remove the recovery instances (A3) in the source region, simply delete the server. This will ensure that the newly launched failed-back instances (A4) remain protected.

e. **Remove the source servers (A2) in the source region.**

Navigate to the **Source servers** in the AWS DRS console. Select the relevant source server and select **Disconnect from AWS** under the **Actions** drop-down menu. Then, select **Delete server** under the same **Actions** menu.

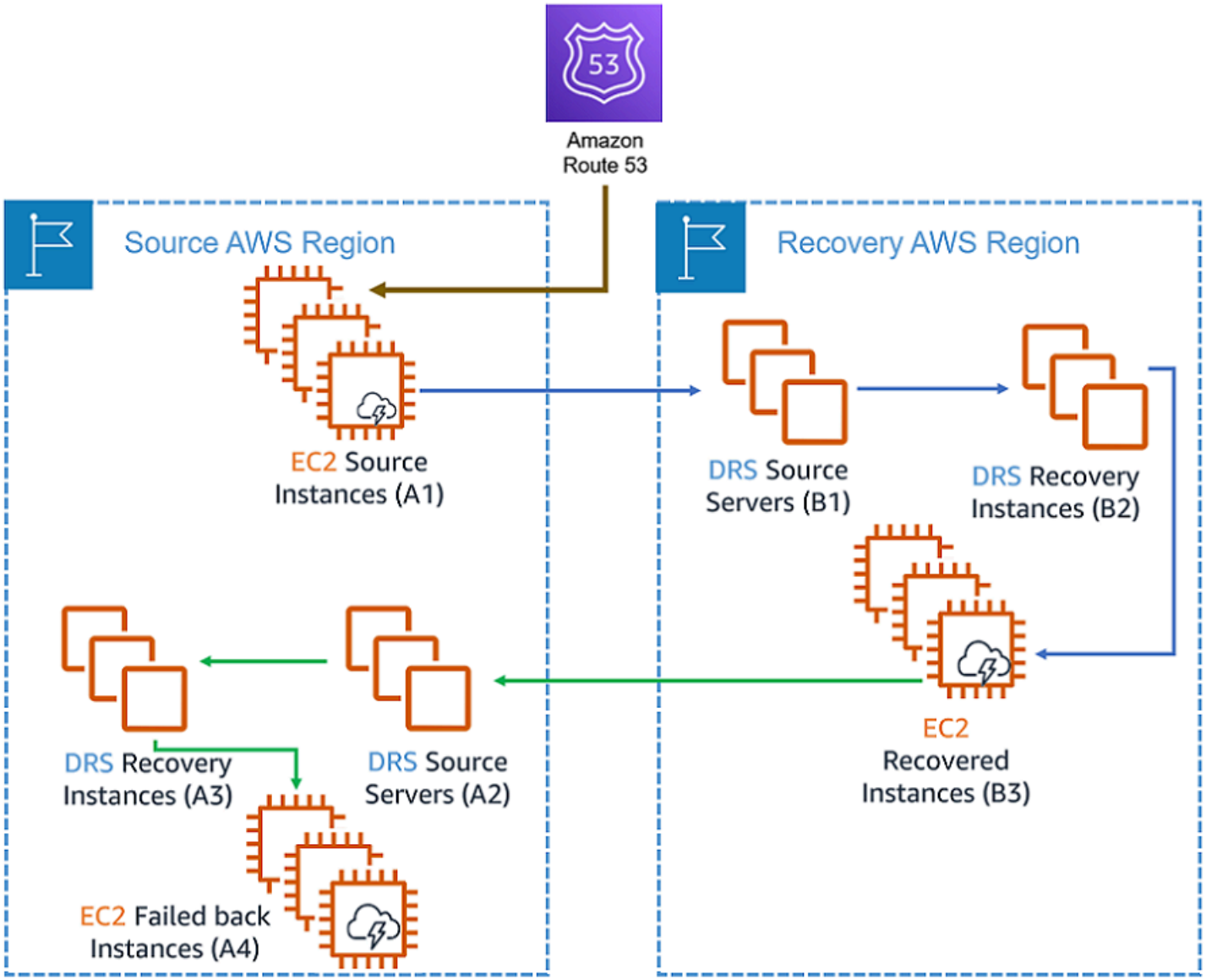
Performing a drill

To conduct a drill, follow the steps 1 and 2 as described above, and then perform a different cleanup process as described below.

> **Note**
>
> 1. Do not stop the source server (B1) in the recovery AWS region as recommended in the note of step 1-e.
> 2. Do not perform step 3, Protecting the failed back instances would affect your production data.

Cleaning up after a drill

After a successful drill your AWS environment should look like this:



The only two AWS resources that need to remain are your actual production environment (A1) and its replication backup (B1). Since DRS protects replication servers, you must stop the replication first.

1. Stop the replication of the Source servers (A2) in the Source region.

> **Important**
>
> Make sure you don't stop replicating the Source servers (B1) in the recovery region.

2. Terminate the recovery instances (A3) in the source region and the recovery instances (B2) in the recovery region. As a result of this action, both the recovered instances (B3) and the failback instances (A4) are terminated as well.

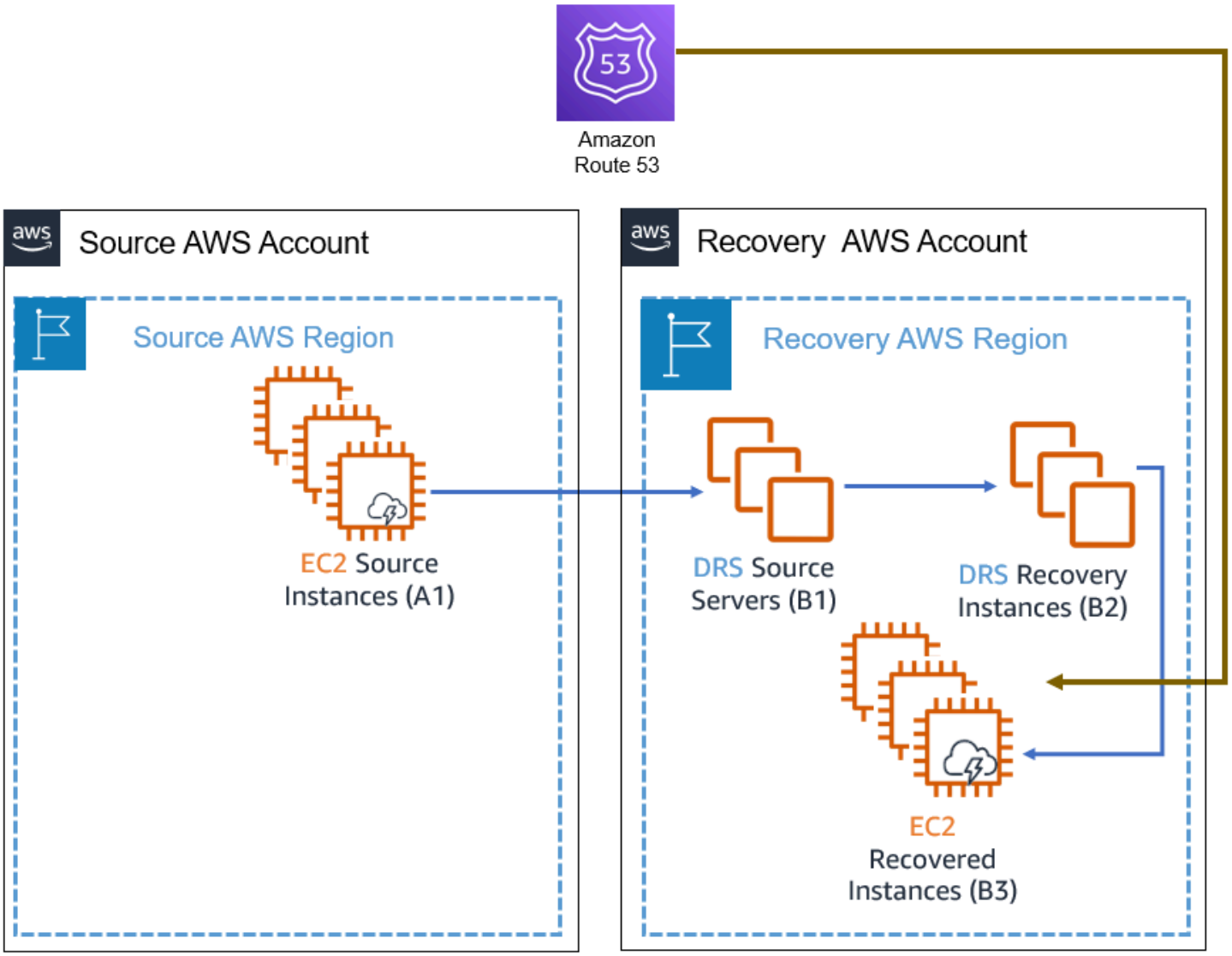
> **Note**
>
> Performing cross-region replication, failover and failback accrues additional costs, not detailed in the AWS DRS pricing examples. These additional costs consist of cross-Region data transfer costs during initial data replication, ongoing data replication, and failback replication; as well as the cost of replication resources (such as Amazon EBS volumes, snapshots, and more), used for failback replication; and also the DRS hourly billing for failback source servers.

Performing a cross-account failback

AWS Elastic Disaster Recovery (AWS DRS) allows you to perform failover and failback on your EC2-based applications from one AWS account to another AWS account. The failover process is the same as failing over into an AWS account from a source outside of AWS, but the failback process is different. The instructions below describe the complete cross-account failover and failback process.

Overview and prerequisites

The failback process starts after the failover process ends. During failover, AWS DRS allows you to replace the EC2 source instance (A1) with the EC2 recovered instance (B3). The current AWS resource state is illustrated in this diagram:



After performing a recovery, your applications are running on EC2 instances in the recovery account and region. However, these recovered instances (marked B3 in the diagram above) are not protected against other potential outages. In order to avoid data loss, you should start a reversed replication immediately. Starting reversed replication is only possible if the service is initialized in the recovery account and region. See initialize the AWS DRS.

Starting reversed replication involves copying the data from the EC2 recovered instances (B3) to the original account and region, an operation that takes time and possibly incurs cross-Region data transfer costs if the source region differs from the recovery region.

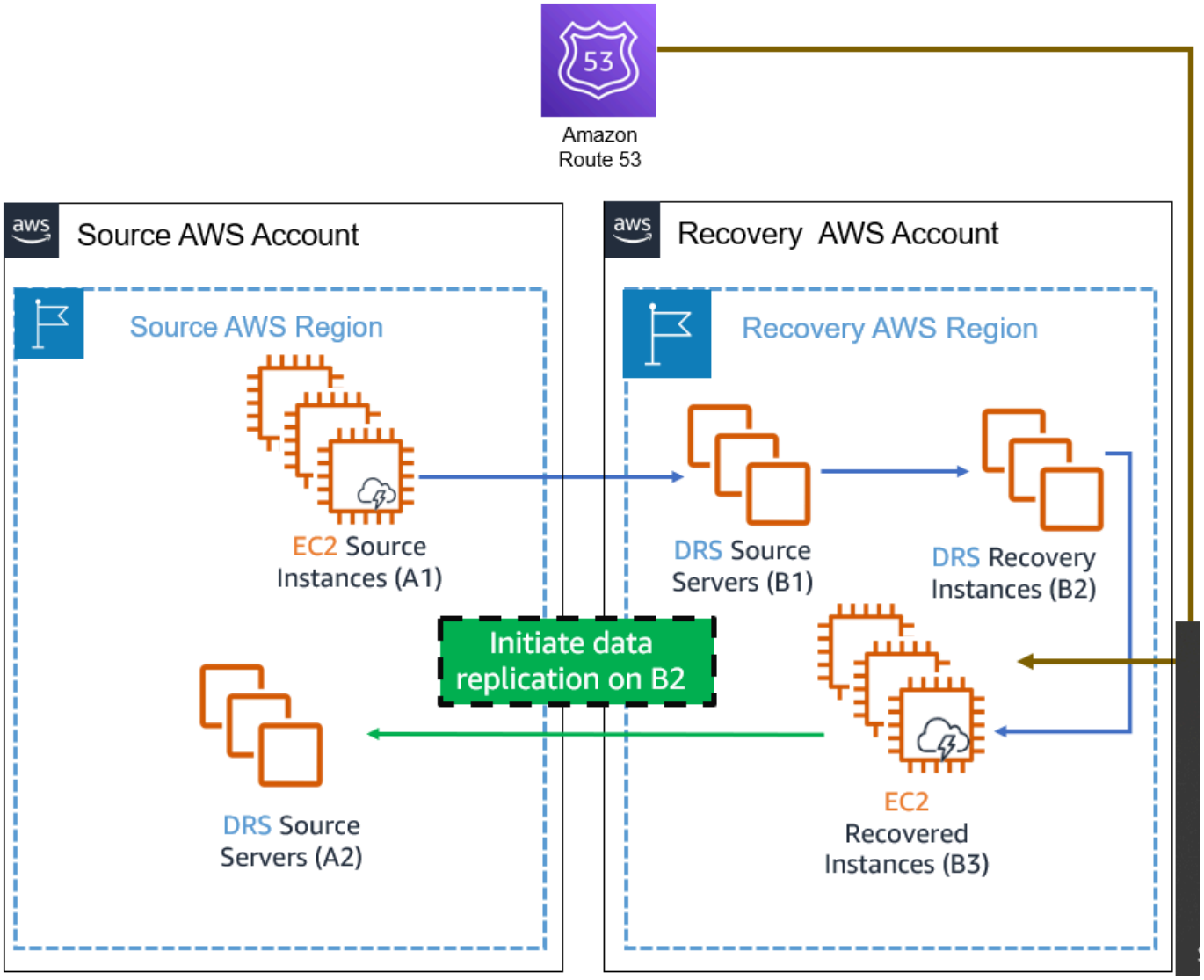
Once replication has reached a healthy state, failing back to the source account (after starting reversed replication) is possible using the DRS console on the source account and region, assuming DRS has been initialized in the source account and region.

Important

- To ensure operational continuity, initialize the AWS DRS in advance in both the source and target AWS accounts and regions, and conduct regular failover and failback drills.
- If the source region is different from the recovery region, and at least one of the involved regions is an opt-in region, it is mandatory that the opt-in region be enabled in both accounts. If both regions are opt-in regions, then both regions must be enabled in both the source account and the recovery account.
- Create the roles, identified as **Failback and in-AWS right-sizing roles** via Trusted Account page in advance, for both directions: from source account to recovery account and from recovery account to source account.
- Before starting a failback, make sure the EC2 recovered instances (B3) have a network interface while meeting the specified network requirements.
- Access to EC2 instance metadata is required. If you have a custom network setup that modifies the operating system route, ensure that access to metadata is intact. Learn how to verify metadata access for Linux and for Windows.

Performing cross-account failback

1. **Start reversed replication.**
   a. Log in to the recovery account and select the recovery region (the account and region where the recovery instances were launched in).
   b. Open the **AWS Elastic Disaster Recovery** service console.
   c. Navigate to the **Recovery instances** page.
   d. Select the servers that you want to protect and click **Start reversed replication**.
   e. A Source server (A2) will be created in the source account and region, as shown in this diagram.



Note

All server data is transferred over the wire during this step. This process could take some time and possibly result in cross-Region data transfer costs if the source region differs from the recovery region. Moreover, starting reversed replication creates additional replication resources (A2). To avoid double billing, you can stop replicating the source instances (A1) by navigating to the AWS DRS source server in the recovery account and region (B1) and clicking **Stop replication** in the replication drop-down menu. Make sure that you only stop the replication after validating the recovery instances because once replication is stopped, all previous points in time are deleted.

> **Important**
>
> Once replication is stopped, all previous points in time are deleted. This is done to minimize costs.

2. **Launch, validate, and redirect traffic.**

   After the **Reversed direction launch state** is marked as **Ready**, take these steps to complete the failback:

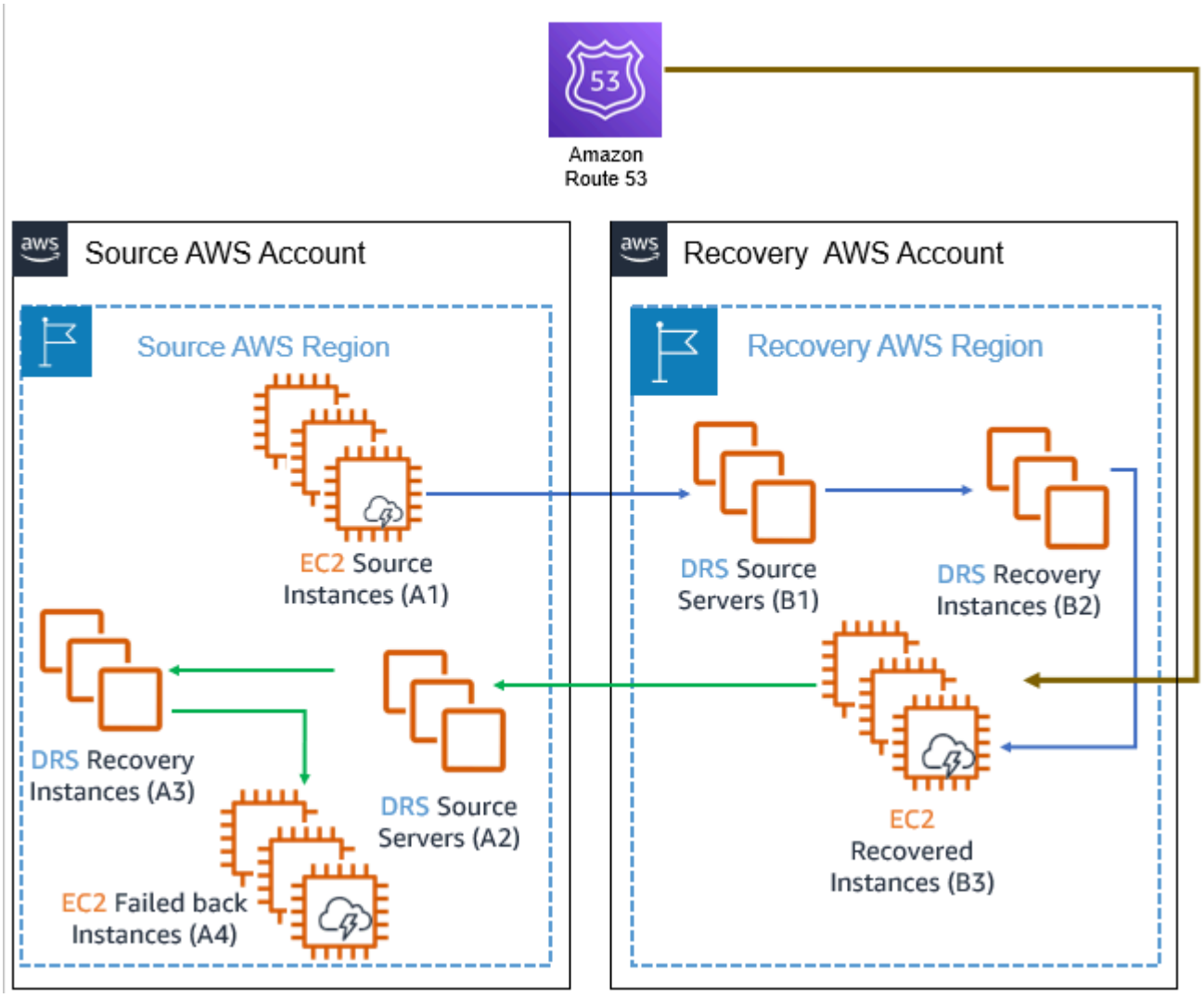
   a. Find the relevant source servers (A2) in the source account and region by using information in the **Replicating to source server** and **Replicating to account** columns of the recovery instance (B2)

   > **Note**
   >
   > You can also find it directly on the **Source servers** page in AWS DRS console at the source account and region.

   > **Note**
   >
   > The **Replicating to account** column is not visible by default and can be made visible in the preferences of the Recovery instances page.

   b. If the state is **Ready** (or **Ready with lag**), click **Launch for failback** under **Initiate recovery job**.

   > **Important**
   >
   > Make sure that your applications (A4) are working as expected. If you run into any issues, you can relaunch the instances and try again. Until you opt to failback, your recovery instances (B3) will continue to run in your recovery account and region to ensure business continuity.



c. Redirect traffic to failed back instances (A4), which will now become your new primary instances. Traffic redirection is not conducted using DRS -> You need to perform traffic redirection either using your systems, or by utilizing a custom post-launch action. Choose a service according to your preferences (consider using Amazon Route 53).

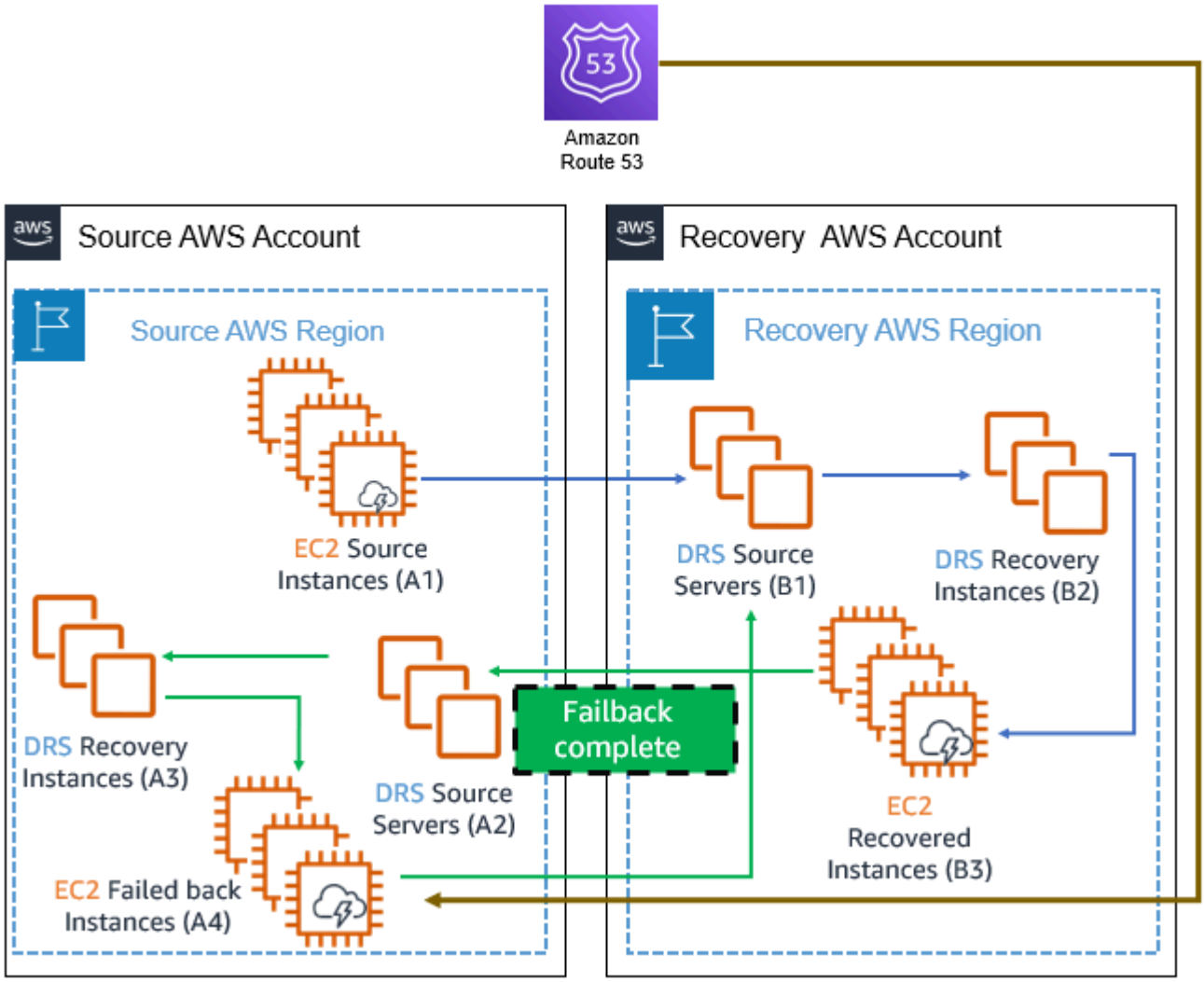
3. **Protect your new failed back instances.**

> **Important**
>
> Do not perform this step when performing a drill. This step replaces the instances that AWS DRS replicates (from the Source instances, A1, to the failed back instances, A4). In a drill, the source instances (A1) are still your production environment.

The newly launched failed-back instances (A4) are not protected. In order to protect them, follow these steps:

a. Navigate to the recovery instance (A3) in the source account and region.

b. Click **Start reversed replication**. This step will replace the Instances that the Source Server (B1) protects (A4 instead of A1).



4. **Clean your environment.**

After the failover to failback cycle is complete, you may be left with multiple AWS resources that you no longer need and that are costly to maintain. These include the source and failover EC2 instances (A1,B3), the recovery instances (B2, A3), and the Source servers (A2). Consider removing them.

Cleanup steps:

a. **Stop replication on the source servers (A2) of the source account and region.**

   Navigate to the source server in the source account and region (A2), and click on **Stop replication** under the **Replication menu**. This step is required before terminating the recovery instance (B2).

b. **Terminate the recovery instances (B2).**

   These instances, launched in your recovery account and region, are no longer needed now that you have launched new primary instances in your original source account and region. To terminate these instances, navigate to the AWS DRS Console in your recovery account and region (B2). After termination, those instances will no longer appear in the **Recovery Instances** page of the DRS Console. This process also terminates the recovered EC2 instances (B3).

c. **Terminate the EC2 instances (A1) on the source account and region.**

   These have now been replaced by the new instances launched in step 2 above (EC2 failed back instances, A4). You might have stopped these instances after the failover, and you can now terminate them using the AWS EC2 Console.

d. **Remove the recovery instance (A3) in the source account and region.**

   Navigate to the **Recovery instances** in the AWS DRS console. Select the relevant recovery instance and click **Delete server** under the **Action** drop-down menu.

   > **Note**
   >
   > If you have started reversed replication for the recovery instance (A3), you will not be able to disconnect it. To remove the recovery instances (A3) in the source account and region, simply delete the server. This will ensure that the newly launched failed-back instances (A4) remain protected.

e. **Remove the source servers (A2) in the source account and region**

   Navigate to the **Source servers** in the AWS DRS console. Select the relevant source server and select **Disconnect from AWS** under the **Actions** drop-down menu. Then, select **Delete server** under the same **Actions** menu.

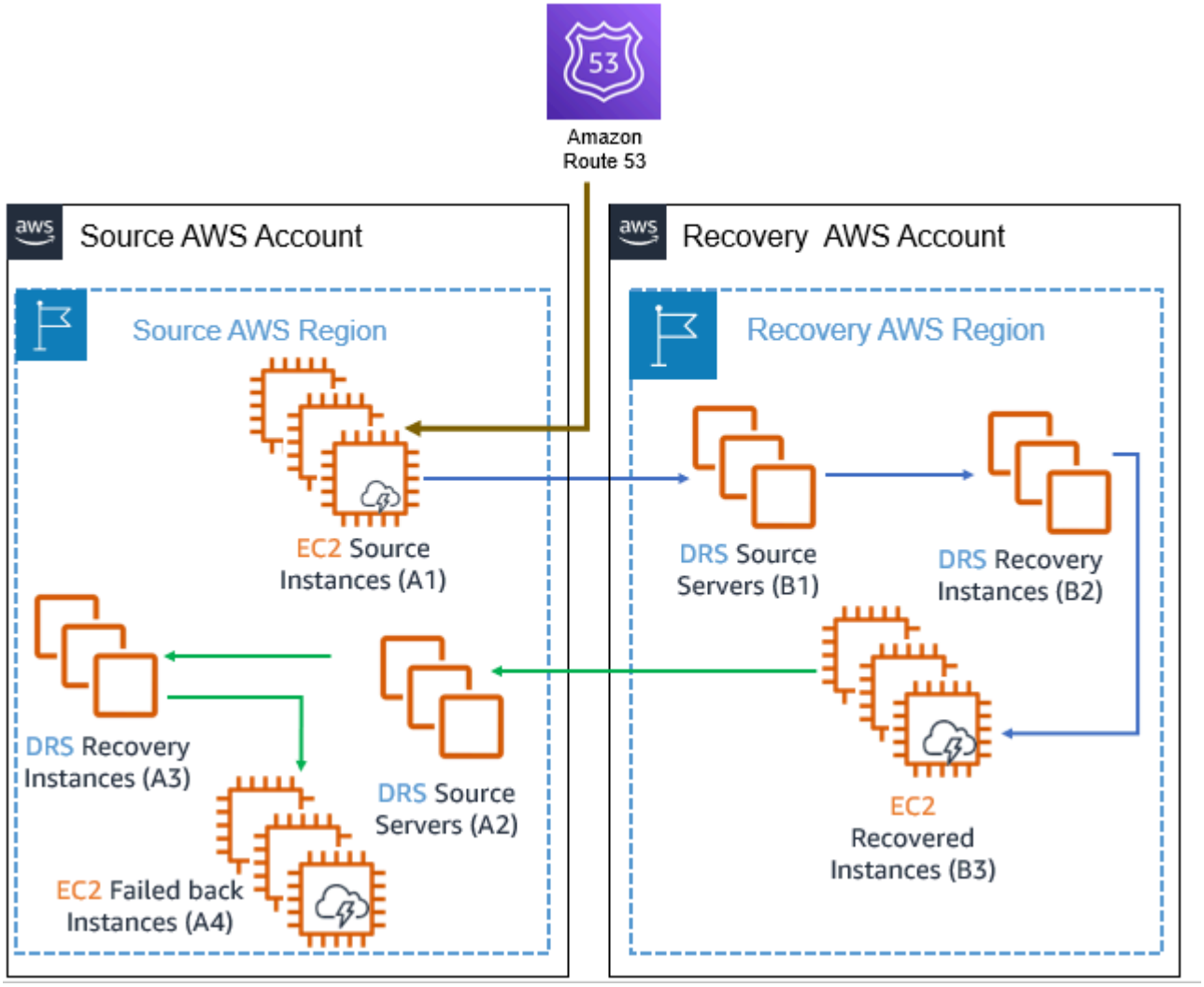
Performing a drill

To conduct a drill, follow the steps 1 and 2 as described above, and then perform a different cleanup process as described below.

> **Note**
>
> 1. Do not to stop the source server (B1) in the recovery account and region as recommended in the note of step 1-e.
> 2. Do not perform step 3, Protecting the failed back instances would affect your production data.

Cleaning up after a drill

After a successful drill your AWS environment should look like this:



The only two AWS resources that need to remain are your actual production environment (A1) and its replication backup (B1). Since DRS protects replication servers, you must stop the replication first.

1. Stop the replication of the Source servers (A2) in the source account and region.

> **Important**
>
> Make sure you don't stop replicating the Source servers (B1) in the recovery account and region.

2. Terminate the recovery instances (A3) in the source account and region and the recovery instances (B2) in the recovery account and region. As a result of this action, both the recovered instances (B3) and the failback instances (A4) are terminated as well.

Note

Performing cross-account replication, failover and failback accrues additional costs, not detailed in the AWS DRS pricing examples. These additional costs consist of cross-Region data transfer costs during initial data replication, ongoing data replication, and failback replication if the source region differs from the recovery region; as well as the cost of replication resources (such as Amazon EBS volumes, snapshots, and more), used for failback replication; and also the DRS hourly billing for failback source servers.

Cross Availability Zone recovery

You can use DRS to replicate and recover EC2 instances across Availability Zones.

Cross Availability Zone (AZ) setup

Initial settings

In order to replicate an EC2 instance across availability zones, the replication settings and launch settings should be set to replicate into an availability zone different from the one hosting your protected EC2 instance. To find out which availability zone hosts an instance, visit the AWS EC2 console.

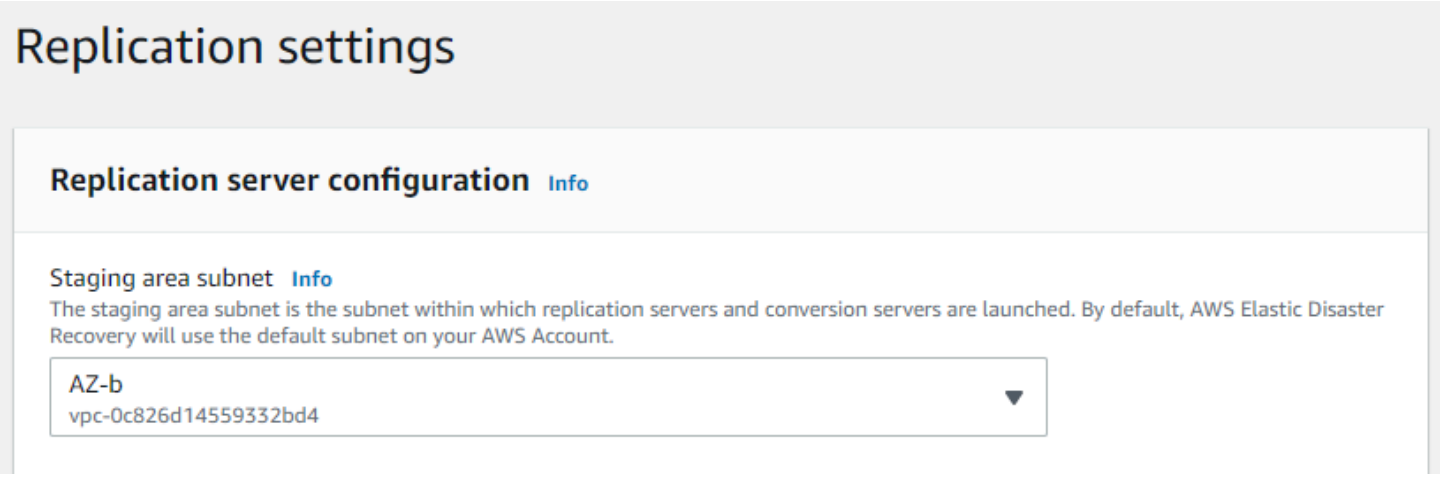
Configure the replication settings and launch template to use a subnet hosted on an availability zone different from the one hosting the EC2 instance being protected.

Example

If the protected EC2 is hosted on availability zone eu-west-1a, the replication settings subnet (and launch template subnet) are hosted on another availability zone in the same region, for example, eu-west-1b.

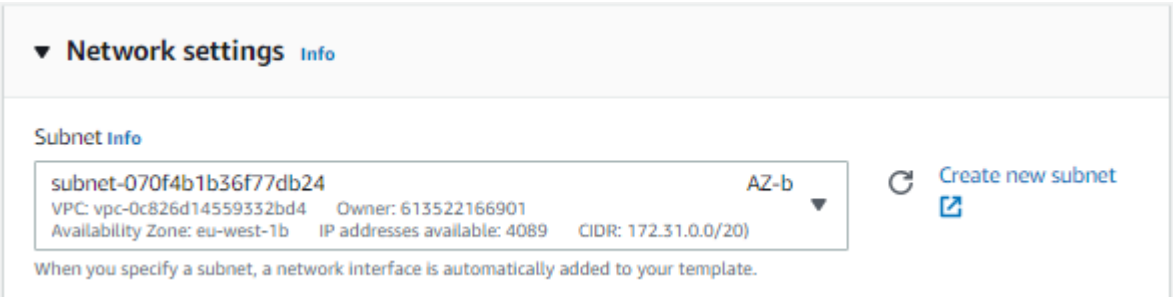
Select a subnet for replication from the replication settings page for the source server. Information about each subnet, including which availability zone hosts it, can be found on the Amazon VPC console.

Replication settings



Launch settings

Learn how to modify the launch template.



Launching a Recovery Instance

To recover the protected EC2 instance, follow these instructions.

Protecting your Recovered Instance

Once a recovery instance has been successfully launched inside a target availability zone and failed over, this recovery instance should be protected by DRS.

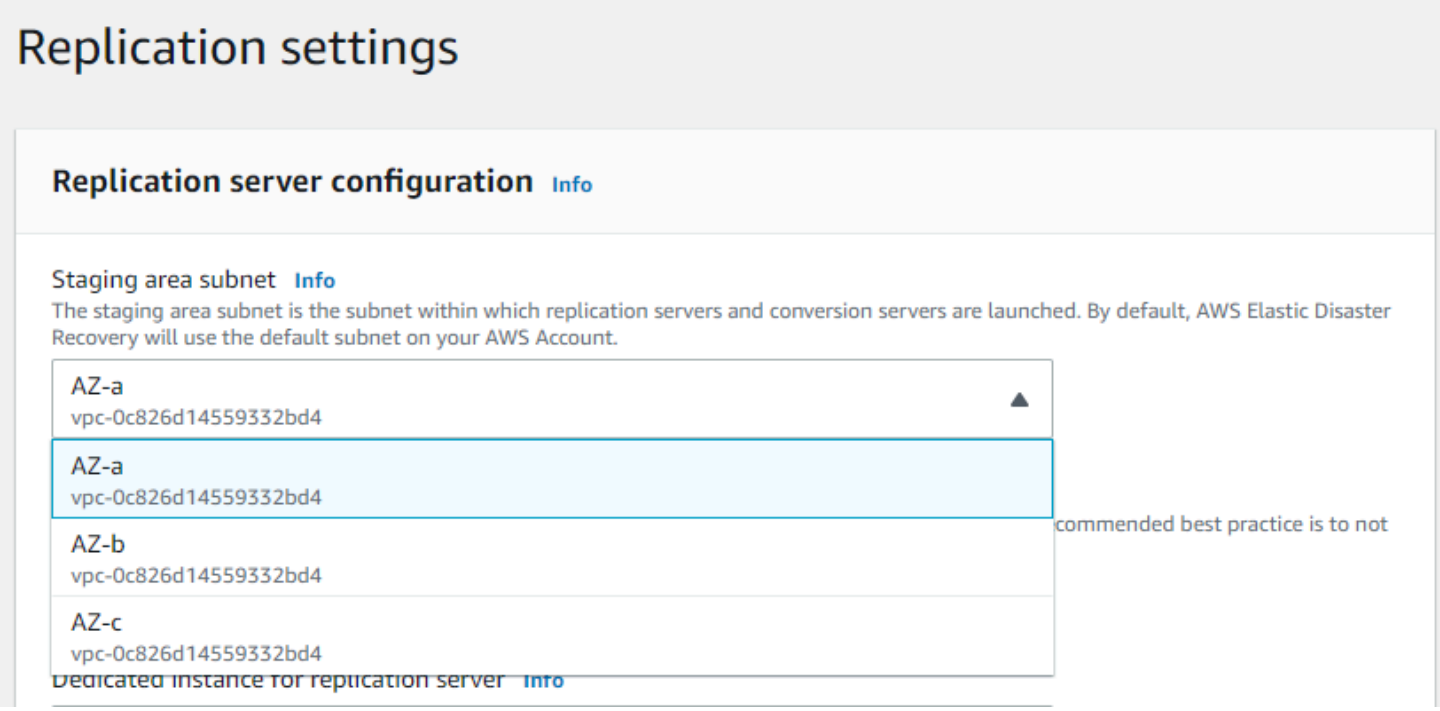
To protect this recovery instance:

- Replication settings and launch template subnets should be changed to a subnet hosted on an availability zone different from the one hosting the EC2 instance that is associated with the recovery instance.
- You must start the replication from the new Recovery EC2 Instance instead of the original EC2 instance.

Example

If a recovery instance was created and the underlying EC2 instance is hosted on availability zone "eu-west-1**b**", the replication settings and launch template can be modified to use a subnet hosted on availability zone "eu-west-1**a**".

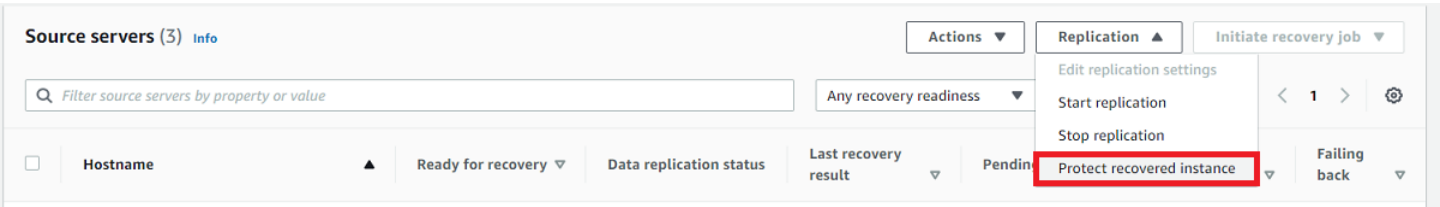
Modify the replication settings to replicate to the original availability zone.



Modify the launch settings to the original availability zone.

In order to modify the launch template follow these instructions.

Protect your recovered instance.



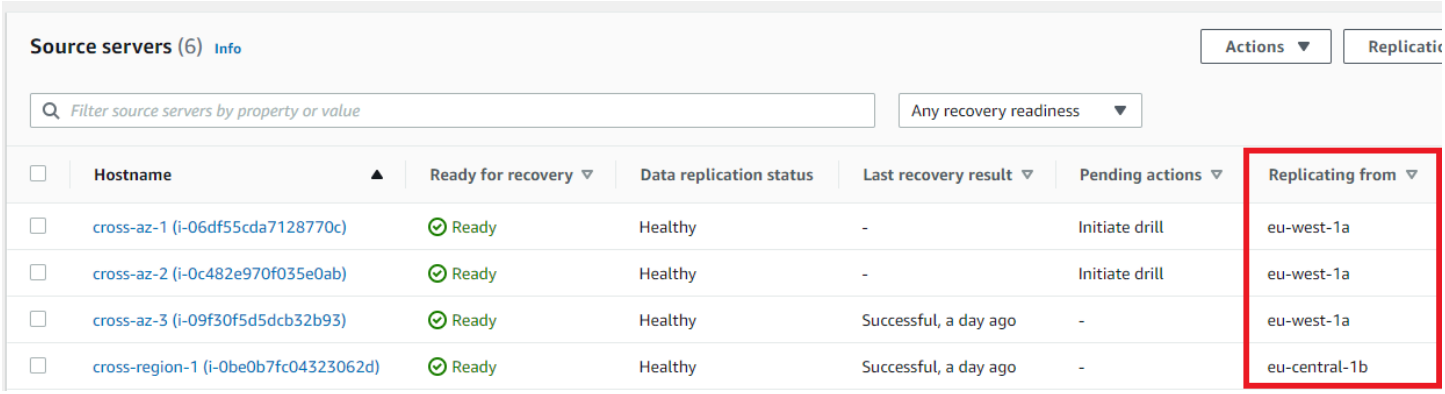
Protecting your recovered instance also stops the replication of the original EC2 instance. For example, if the original EC2 instance is hosted in availability zone "eu-west-1a" and is recovered to a subnet hosted in availability zone eu-west-1b, starting the replication on the recovered instance back to eu-west-1a also stops the replication of the original instance hosted in eu-west-1a.

Starting the replication for a recovered instance only initiates a rescan (to apply the new instance's changes on the last snapshot) instead of a full synchronization. The reason is that all the replication resources associated with the original instance, such as point in time snapshots, configuration, and job logs are retained. After the replication has started, there is no need to keep the original instance for replication purposes.

The availability zone hosting the EC2 instance that is being protected can be viewed on the **Source servers** list (**Replicating from** column).

> **Note**
>
> One of the major benefits of cross AZ replication is that the replication agent only needs to rescan the differences between the latest point in time snapshot and the current source server data. This saves both time and resources. All points-in-time snapshots, configuration, and job logs will be retained. You can now terminate the original EC2 instance in eu-west-**1a**. Your recovered instances are now protected.

You can view the source environment availability zone from the **Source servers** list.



AWS DRS Recovery Instances page

Recovery instances overview

You can manage your recovery instances on the **Recovery instances** page.

This page displays all of the recovery instances that you launched in AWS for your source servers and those that you added directly to Elastic Disaster Recovery.

It allows you to monitor the data replication status of your recovery instances, view recovery instance details, start reversed replication, edit recovery instance failback settings for on-premises failback, view post-launch actions run results, and terminate recovery instances.

Monitoring recovery instances

You can monitor your recovery instances on the **Recovery instances** page. It displays all of your recovery instances and sorts them by **Instance ID**, **Reversed direction launch state**, **Data replication status**, **Pending actions**, **Replicating to source server**, **Last launch result**, and **Launched from source server.**

You can sort your recovery instances alphabetically in descending or ascending order by choosing the arrow next to the various category headers (with the exception of data replication status).

You can filter the recovery instances page by the properties in the **Filter by property or value** box.

Recovery instance categories

Here is a breakdown of each category header:

Instance ID

The **Instance ID** category displays the ID of the recovery instance. Choose an Instance ID to open the recovery instance details view. Learn more about the recovery instance details view.

Reversed direction launch state

The **Reversed direction launch state** displays the current state of the reversed direction launch for the recovery instance. Possible states include:

- **Not started** – Reversed replication has not been started for the recovery instance.
- **Synchronizing** – Reversed replication has been started for the recovery instance and is currently in process.
- **Ready** – Reversed replication has completed initiation and is ready to be launched.
- **Completed** – Failback process to the on-premises server has been successfully completed. This value does not appear for in-AWS launch flows.
- **Error** – There was an error during the reversed replication process. You can learn more about the cause of the error in the **Data replication status** and **Pending actions** columns.

Data replication status

The **Data replication status** category displays the current data replication status of the recovery instance. Possible states include:

- **Not started** – Data replication has not started for the recovery instance. This indicates that failback has not started for the instance.
- **Initiating** – Data replication is initiating. This indicates that reversed replication has been initiated for the instance.
- **Initial sync** – The recovery instance is undergoing the initial sync process after reversed replication has been initiated. The Elastic Disaster Recovery Console displays the percentage completed and the time left.
- **Rescanning** – The recovery instance is undergoing a rescan. The AWS Elastic Disaster Recovery Console displays the percentage completed and the time left.
- **Healthy** – The data replication process has been completed and the recovery instance is ready for launch.
- **Lag** – The recovery instance is currently experiencing lag. Open the recovery instance details view to learn more.
- **Stalled** – The recovery instance is experiencing a stall. Open the **Recovery instance details** view to learn more.
- **Completed** – The failback process has been completed and as a result data replication has been successfully completed and stopped. This value is only relevant to on-premises failback and does not appear in in-AWS flows.
- **Disconnected** – The recovery instance has been disconnected from AWS Elastic Disaster Recovery. As a result, data replication has stopped.

Pending actions

The **Pending actions** column provides additional details, when relevant, about the next actions that should be performed to progress the current flow or to initiate the reversed replication. Possible values include:

- **Launch for failback on {region}** – This status indicates that reversed replication has reached a healthy state. To launch for failback, choose the link under **replicating to source server**.
- **Use failback client** – To start the replication back to the on-premises server, use the Failback Client. This value is only relevant to on-premises failback.
- **Start reversed replication to {region name}** – Choose **start reversed replication** to initiate reversed replication to the specified region. This value only applies to in-AWS and cross-region replications.

Replicating to source server

The **Replicating to source server** category identifies the source server to which the recovery instance is replicating. When you start reversed replication it is managed through this source server. Launch operations are performed by navigating to this source server and initiating the operation from that screen.

These are displayed in order:

- The source server region
- The source server's ID

Choose the source server links to view the source server details of the source server that is associated with the specific recovery instance. Learn more about the server details view. If the source server is located in another region (marked by an external icon), choosing the link opens the source server's details page in a different tab.

Last launch results

This category indicates the results of the last launch. Possible values include:

- Launch successful
- Failback successful

- Launch failed
- Failback failed

Launched from source server

The **Launched from source server** column identifies the source server from which the recovery instance was launched.

These are displayed in order:

- The source server hostname
- The source server's ID

Choose the source server links to view the source server details of the source server that is associated with the specific recovery instance. Learn more about the server details view.

Recovery instances actions

The recovery instances page allows you to perform actions that include viewing recovery instance details, adding recovery instances, editing the failback replication settings, terminating recovery instances, and continuing the failback process.

Actions menu

Actions available on the **Actions** menu:

View instance details

Select a recovery instance and choose the **View instance details** option under the **Actions** menu to open the **Recovery instance details** view. Learn more about the recovery instance details view.

Edit failback replication settings

Select one or more recovery instances and choose the **Edit failback replication settings** option under the **Actions** menu to edit the failback replication settings for the selected recovery instances. The failback replication settings configure the replication to the on-premises servers during an on-premises failback process. This does not apply to in-AWS replication, which is managed on the **replicating to source server** source servers. Learn more about Failback replication settings.

Stop failback

Select one or more recovery instances that are in the **Synchronizing** state and choose the **Stop** option under the **Actions** menu to stop the failback process for the selected recovery instance or instances. This returns the instances' **Reversed replication launch state** to **Not started** and stops any ongoing failback process. The Failback client indicates that the failback has been stopped. To restart failback, reboot the machine in the Failback Client. Note that the **Stop failback** state is only relevant to on-premises flows.

On the **Stop failback for recovery instances** dialog choose **Stop failback**.

Terminate recovery instances

Select one or more recovery instances and choose the **Terminate recovery instances** option under the **Actions** menu to terminate the recovery instance or instances. This removes all of the resources associated with the selected recovery instance or instances from Elastic Disaster Recovery and terminates all related EC2 resources. Perform this action if you no longer need the recovery instance, for example, if it was for a drill.

On the **Terminate recovery instances** dialog choose **Terminate**.

Disconnect from AWS

Select one or more recovery instances and choose the **Disconnect from AWS** option under the **Actions** menu to disconnect the recovery instance or instances from AWS. This deletes the AWS Replication Agent from the recovery instance or instances, but keeps the recovery instance Elastic Disaster Recovery resources and the EC2 resources intact. You may want to disconnect from AWS if you do not want to perform a launch for the specific recovery instance or instances and do not want to accrue additional costs for data replication, but still want the recovery instance to appear in the Elastic Disaster Recovery Console.

On the **Disconnect X recovery instances from service** dialog choose **Disconnect**.

Delete recovery instances

Select one or more recovery instances and choose the **Delete recovery instances** option under the **Actions** menu to delete the recovery instance or instances. This removes all of resources associated with the selected recovery instance or instances from Elastic Disaster Recovery but does not terminate all related EC2 resources and the instance keeps running on Amazon EC2.

You may want to delete the recovery instance or instances if you already failed over into AWS, but then decided to permanently keep your workload in AWS instead of failing back to your original

source servers and do not want to incur any more costs associated with Elastic Disaster Recovery resources. You may also want to delete the recovery instance or instances if you performed an in-AWS launch but do not want to start reversed replication back to the original region. Note that you can only delete recovery instances that have already been disconnected from AWS.

On the **Delete recovery instance** dialog choose **Delete**.

> **Note**
>
> Launch of a new recovery instance from the same source server cleans up all the previous recovery instances, regardless if they have been disconnected and deleted from DRS.

Failback

Select one or more recovery instances that are in the **Ready** state and choose the **Complete failback** option to continue the failback process after performing a failback with the Elastic Disaster Recovery Failback Client. This action stops data replication and starts the conversion process. This finalizes the failback process and creates a replica of each recovery instance on the corresponding source server.

> **Important**
>
> Ensure that you complete the entire failback process with the Elastic Disaster Recovery Failback Client prior to choosing the Failback option.

On the **Continue with failback for X instances** dialog choose **Failback.**

Recovery instance details view

The recovery instance details view provides an overview of the recovery instance, including the instance's reversed direction launch process, post-launch action runs and data replication status. It allows you to control instance tags and the instance's failback settings.

You can access the recovery instance details view by choosing the instance ID of a recovery instance under the **Instance ID** column.

You can also access the recovery instance details view by selecting a recovery instance and choosing the **View instance details** option under the **Actions** menu.

The recovery instance information page displays the **Instance ID** at the top.

The **Overview** panel provides an overview of the failback process, including:

- **EC2 instance** – the ID of the recovery instance in EC2. Choose **View in EC2** to open the AWS EC2 Console.
- **Pending actions** – information derived from the **Pending actions** column (for example, **Launch for failback on {region}**).
- **Replicating to source server** – the source server to which the recovery instance is replicating. Choose the source server ID to open the **Source server details view** page for the specific source server.
- **Launched from source server** – the source server from which the recovery instance was launched. Choose the source server ID to open the **Source server details view** page for the specific source server.
- **Post-launch actions status** – displays the status of the last post-launch actions run on this instance.

The recovery instance details page is divided into these sections:

- **Launch dashboard** - see the current status of failback or reversed direction replication and launch.
- **Instance information** - view information about the underlying EC2 instance.
- **Tags** - manage the tags of the recovery instance.
- **Failback replication settings** - configure settings for failback to on-premises servers. Not relevant for in-AWS launches.
- **Post-launch actions status** - the progress or result of the last post-launch actions run.

Launch dashboard

The **Launch** dashboard provides a detailed overview of the reversed direction launch process.

Reversed direction launch state

The **Reversed direction launch state** panel provides an overview of the reversed direction launch process, including:

- The current state of the failback.

- **Launch target** – the server into which the recovery instance is launching. This indicates whether the recovery instance is launching into the original server or to a new server. Note that for in-AWS flows this value is always a new server, unless you are using the **Launch into existing instance** capability.
- **Last job ID** – the ID of the last failback job started for the recovery instance.
- **Last job started** – the date and time the last failback job was started for the recovery instance.
- **Last job finished** – the date and time the last failback job was finished for the recovery instance.
- **Last launch results** – the results of the most recent launch.

Data replication status

The Data replication status panel displays the current data replication status state for the recovery instance, including:

- **Replication progress** – the progress of the replication of the recovery instance in percent completed.
- **Total replicated storage** – the total amount of storage replicated in GiB.
- **Lag** – the total Lag time, if any.
- **Backlog** – the total backlog amount and time to clear, if any.
- **Elapsed replication time** – time elapsed since replication began.
- **AWS replication agent last seen** – the date and time connectivity was last established between the recovery instance and the AWS Replication Agent.
- **Failback client last seen** – the date and time connectivity was last established between the recovery instance and the Failback client.
- **Replication start time** – the date and time replication was started for the recovery instance.

Events and metrics

The Events and metrics section contains external links to monitor your recovery instance in AWS CloudTrail. Learn more about monitoring DRS with CloudTrail.

Instance information

The **Instance information** tab displays general server information, hardware, and network information:

- **Last updated**
- **AWS recovery instance ID**
- **Created in recovery job**
- **Hostname**
- **Fully qualified domain name**
- **CPUs**
- **Disks**
- **Primary network interface**
- **Operating system** information

Tags

The Tags section displays tags that have been assigned to the server. A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs. Learn more about AWS tags in Tag your Amazon EC2 resources.

Choose **Manage tags** to add or remove tags. On the **Manage tags** page choose **Add new tag** to add a new tag. Add a tag **Key** and an optional tag **Value**. **Save** your added tags.

To remove a tag, choose **Remove** to the right of the tag you want to remove, and then choose **Save**.

Failback replication settings

The Failback replication settings tab allows you to edit various failback replication settings for the recovery instance prior to performing a failback.

Choose **Edit** to edit the settings.

You can configure the failback replication settings for multiple recovery instances at once. The **Selected recovery instances** box displays the recovery instances for which you are updating the settings.

Network bandwidth throttling

You can control the amount of network bandwidth used for data replication per server. By default, Elastic Disaster Recovery uses all available network bandwidth on five concurrent connections.

Choose **Throttle bandwidth** if you want to control the transfer rate over TCP Port 1500 of data sent from your recovery instances to your source servers during failback. On the **Throttle network bandwidth (per instance, in Mbps)** enter the bandwidth in Mbps.

Otherwise, choose **Do not throttle bandwidth**.

Use private IP

By default, data is sent from the recovery instance to the source servers over the public internet, using the public IP that was automatically assigned to the replication servers. Transferred data is always encrypted in transit.

Choose the **Use private IP** option if you want to route the replicated data from your recovery instance to your source servers through a private network with a VPN, AWS Direct Connect, VPC peering, or another type of existing private connection. Use this option if you want to:

- Allocate a dedicated bandwidth for replication
- Use another level of encryption
- Add another layer of security by transferring the replicated data from one private IP address (source) to another private IP address (on AWS)

> **Important**
>
> Data replication does not work unless you have already set up the VPN, AWS Direct Connect, or VPC peering in the AWS Console.

> **Note**
>
> - If you selected the default subnet, it is highly unlikely that the private IP is activated for that subnet. Ensure that Private IP (VPN, AWS Direct Connect, or VPC peering) is activated for your chosen subnet to use this option.
> - Choosing the **Use Private IP** option does not create a new private connection.

Saving failback replication settings

Once you have configured your failback replication settings, choose **Save failback replication settings.**

Post-launch actions status

The **Post-launch actions** view displays the current run status of post-launch actions.

The status includes:

- **Order** - the running order of the action.
- **Name** - the name of the action is a link to the detailed run status in the AWS Systems Manager console.
- **Run result** – provides the current action run status.
- **Start time** – the time when the action script started to run. This column is empty for actions that have not yet started running.
- **End time** – the time when the action script run ended. This column is empty for actions that have not yet completed running.
- **Details** – error messages are displayed in this column.
- **Link** – provides a link to resources created by this action if there are any, or to the action run logs in the AWS Systems Manager console.

Recovery job history

The Recovery launch history page provides an in-depth overview for operations (Jobs) performed in Elastic Disaster Recovery.

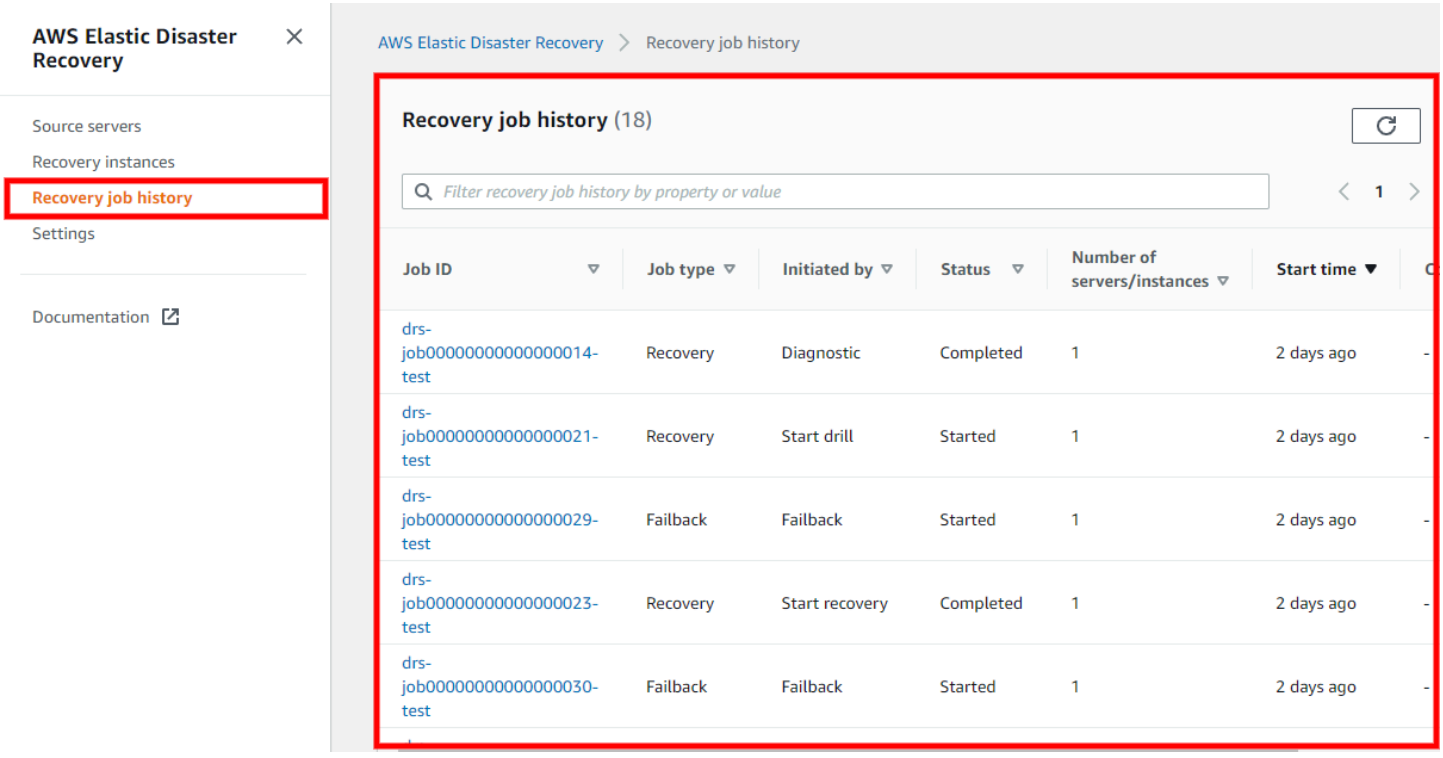
Topics

- Recovery job history

Recovery job history

The **Recovery job history** page allows you to track and manage all operations performed in Elastic Disaster Recovery.

You can access the Recovery job history page by choosing **Recovery job history** on the left-hand navigation menu.

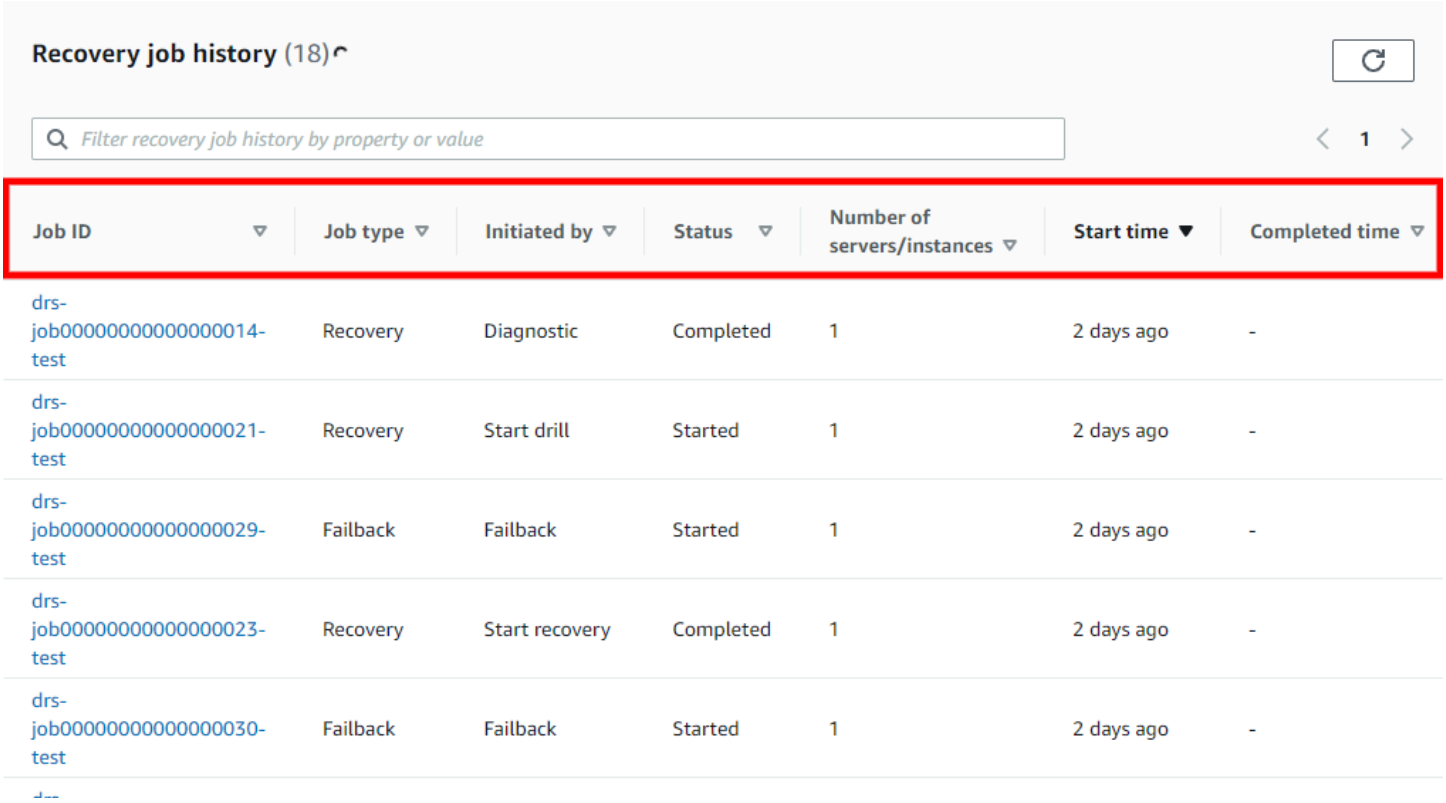


Topics

- Overview
- Job Details

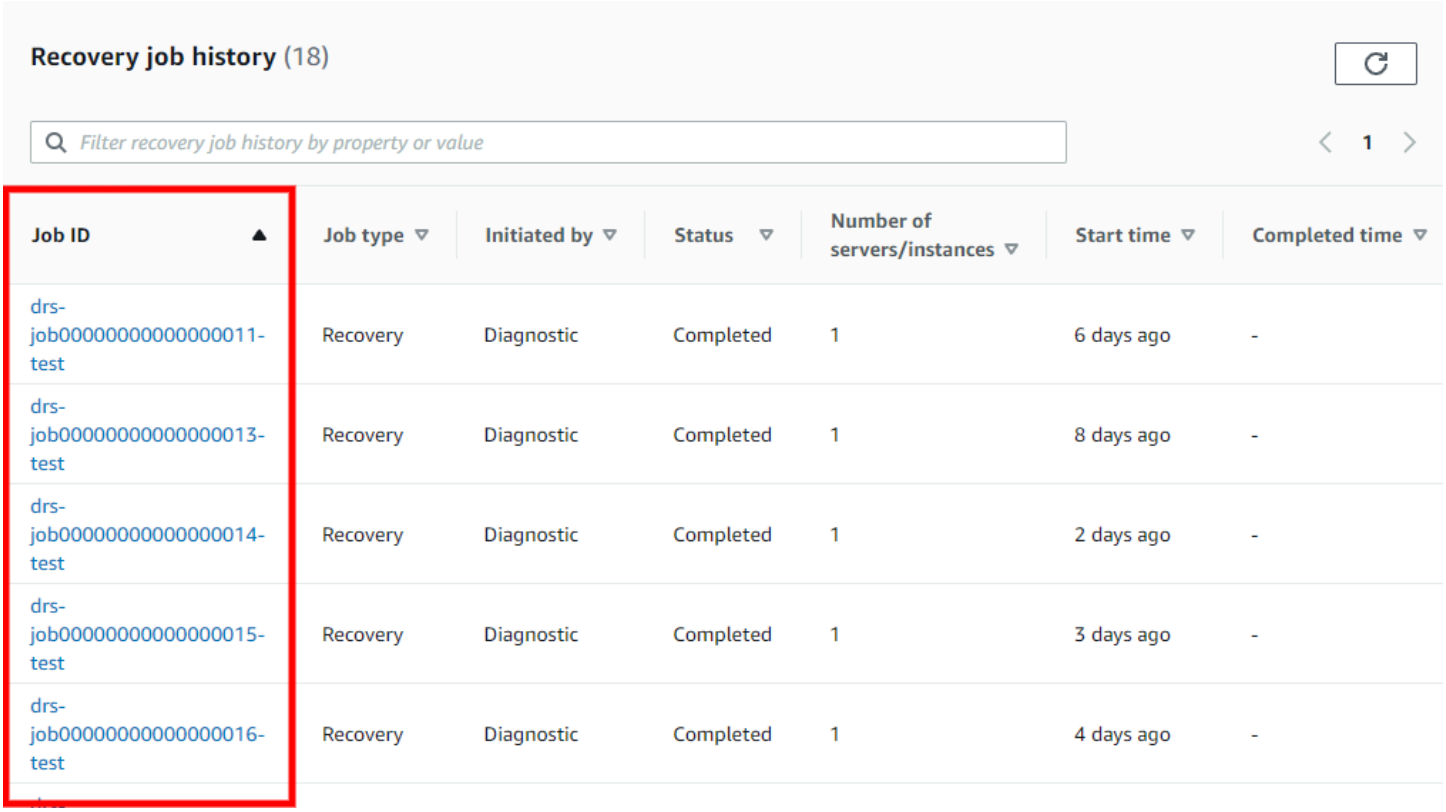
Overview

The Recovery job history tab shows all of the operations (referred to as "Jobs") performed on your account. Each Job corresponds to a single operation (ex. Launch Recovery instance, Launch Drill instance, etc.) Each Job is composed of one or more servers. The main Recovery job history view allows you to easily identify all key Job parameters, including:

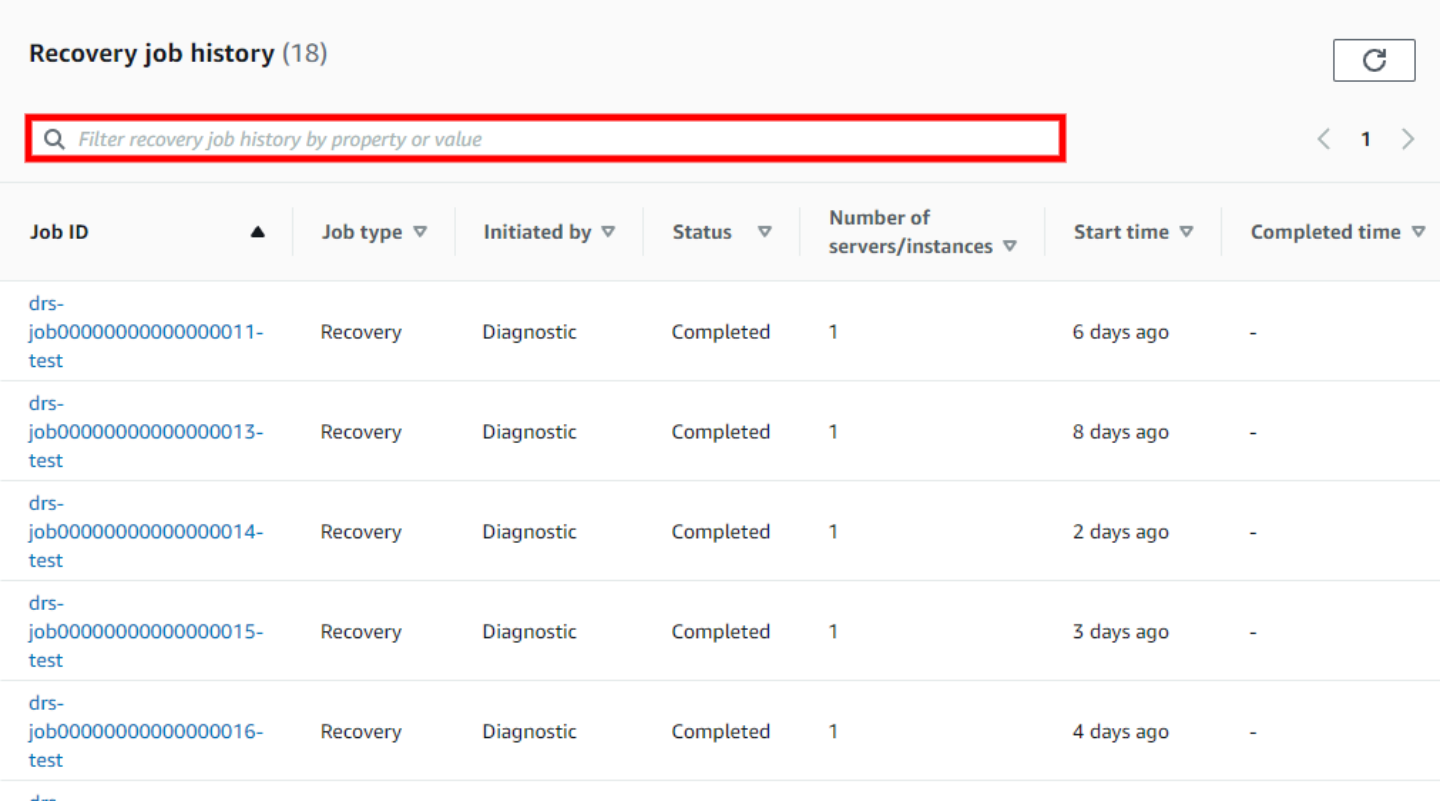


- **Job ID** - The unique ID of the Job.
- **Job Type** - The type of Job (Recovery, Failback, or Terminate)
- **Initiated By** - The command or action that initiated the Job (ex. Drill, Recovery, Failback)
- **Status** - The status of the Job (Pending, Completed, or Started)
- **Servers** - The number of servers that are included in the Job.
- **Start Time** - The time the job was started.
- **Completed Time** - The time the Job was completed (blank if the job was not completed)

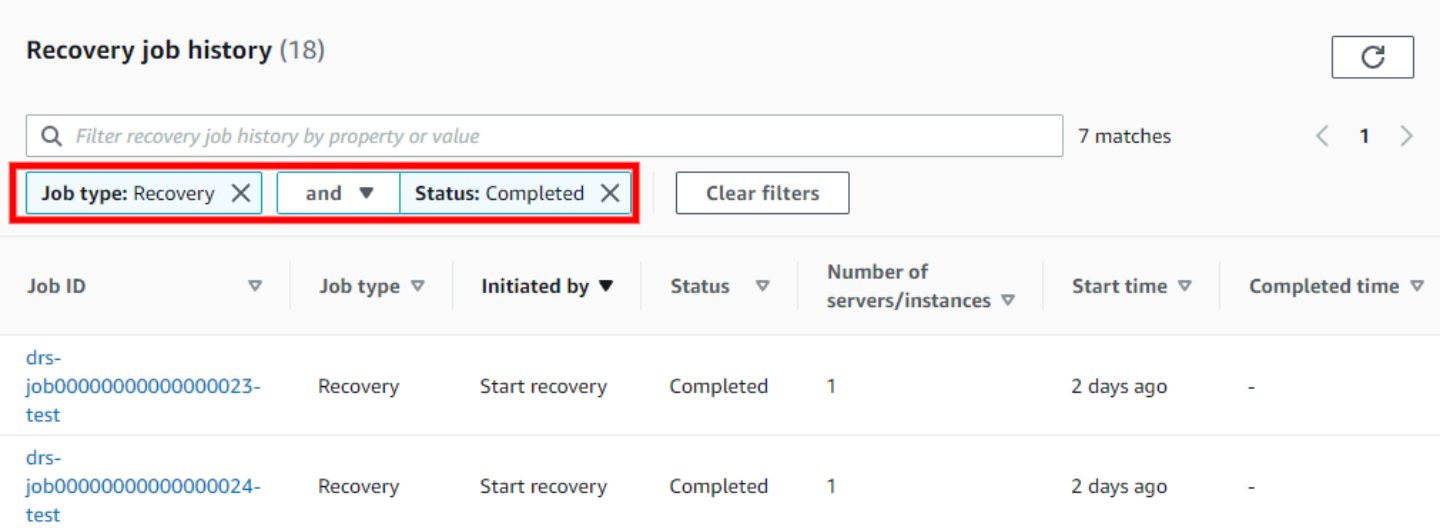
To sort the Recovery job history by any column (for example, **Job ID**), click the column header.



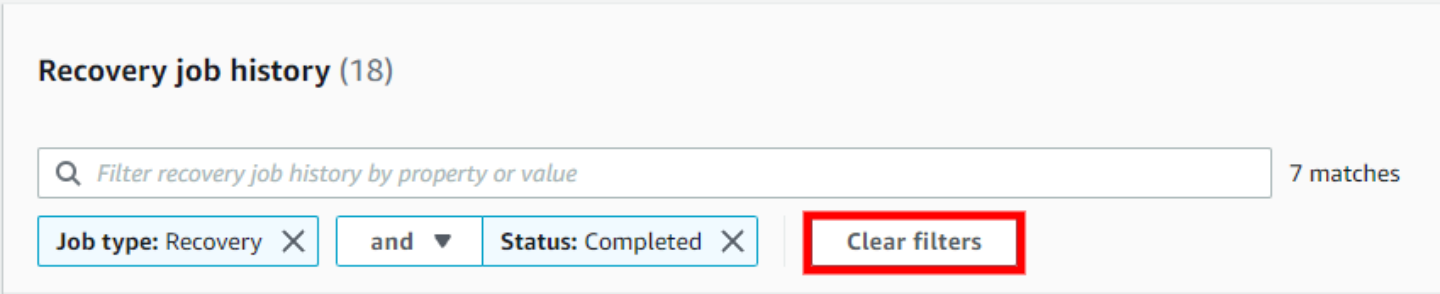
You can search for specific Jobs by any of the available fields within the **Find launch history by property or value** search bar.



Example: Filtered search for the values **Job type: Recovery** and **Status: Completed**, only showing completed Recovery Jobs.



Choose **Clear filters** to clear the search results and return to the default Job History view.

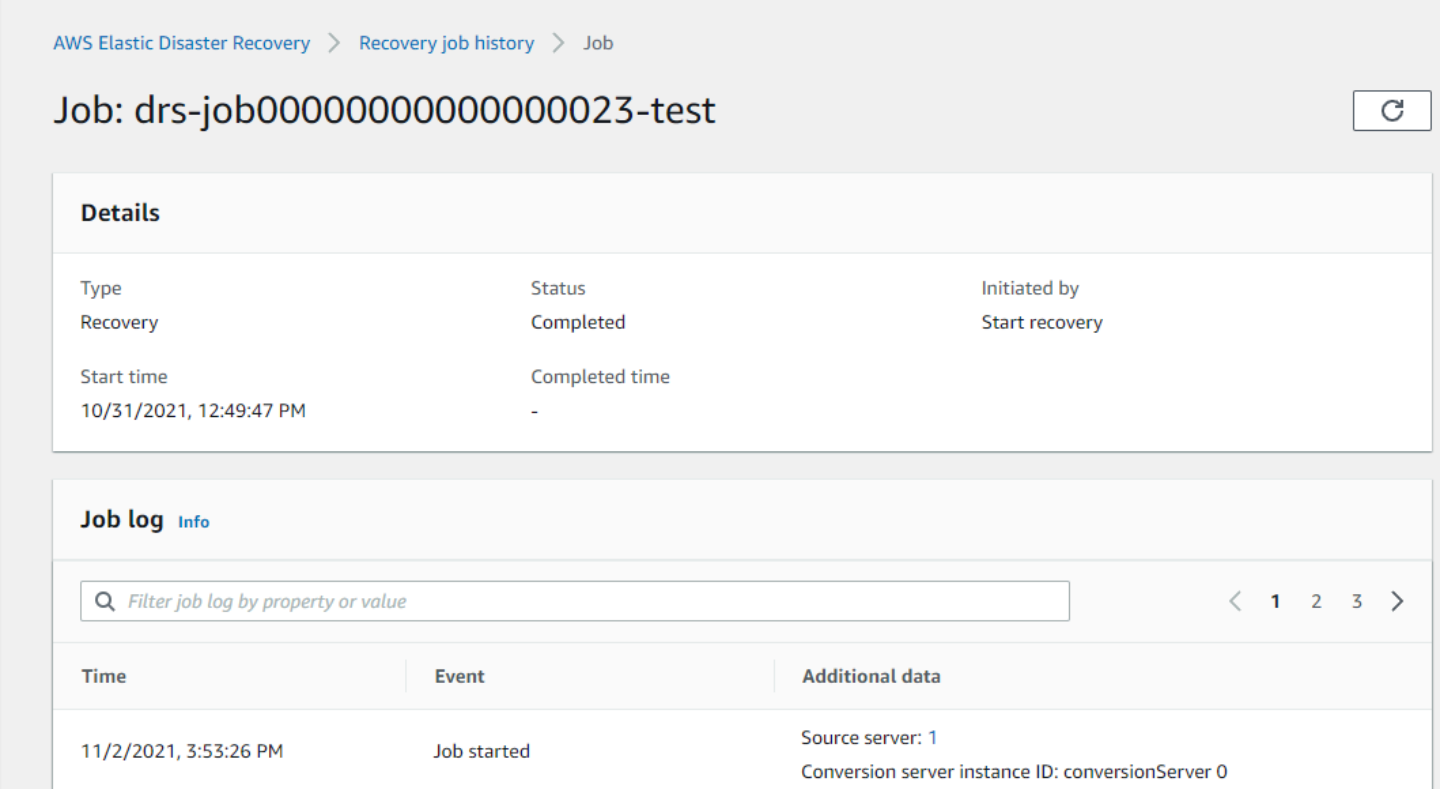


Job Details

You can view a detailed breakdown of each individual job by choosing the Job ID. Choose the **Job ID** of any Job to open the Job details view.

Job ID	Job type	Initiated by	Status	Number of servers/instances	Start time
drs-job00000000000000014-test	Recovery	Diagnostic	Completed	1	3 hours ago
drs-job00000000000000028-test	Failback	Continue failback	Started	1	3 hours ago

The Job details view is composed of three sections:

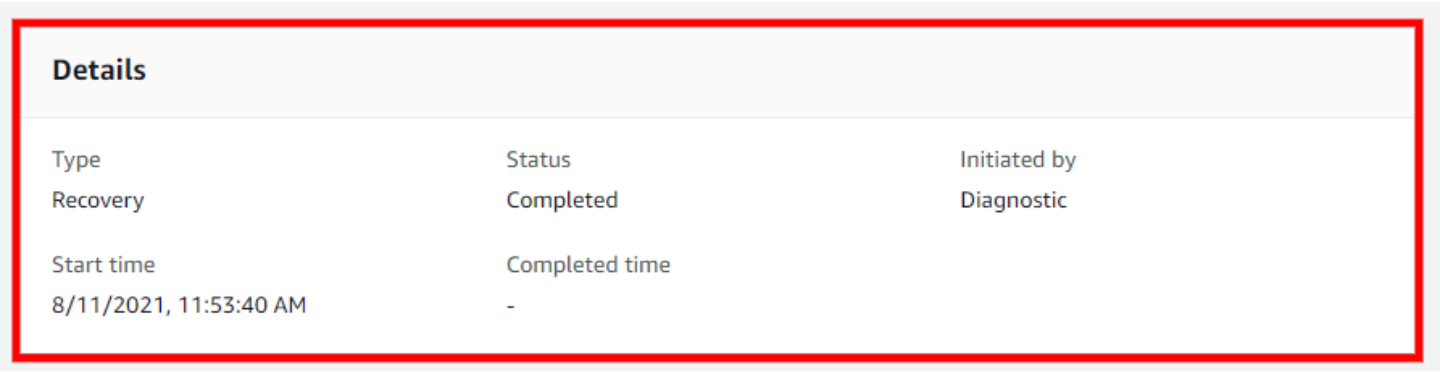


Topics

- Details
- Job log
- Jobs - Source servers

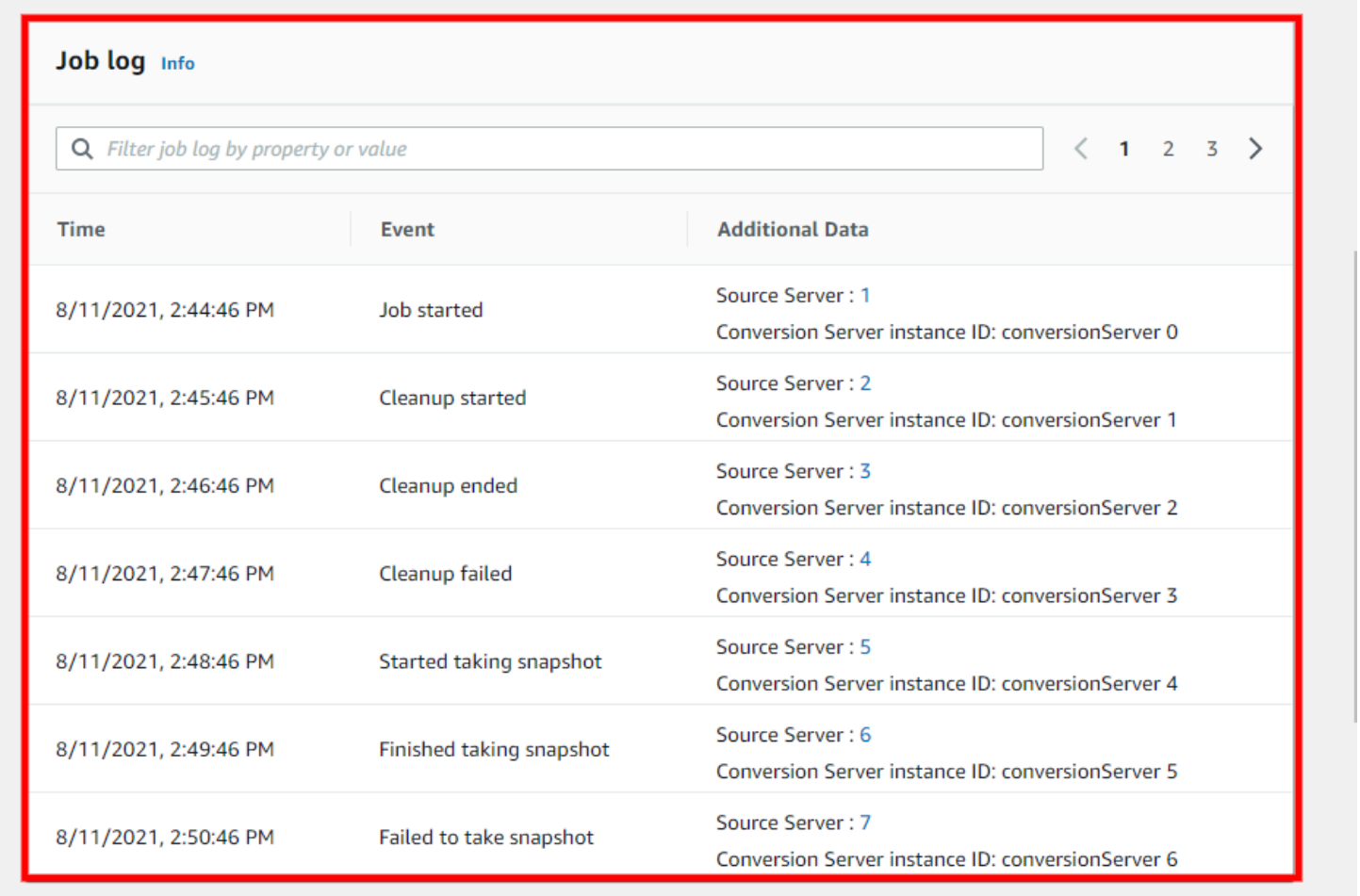
Details

The **Details** section shows the same information as the main Job log page, including the **Type, Status, Initiated By, Start time,** and **Completed time**.



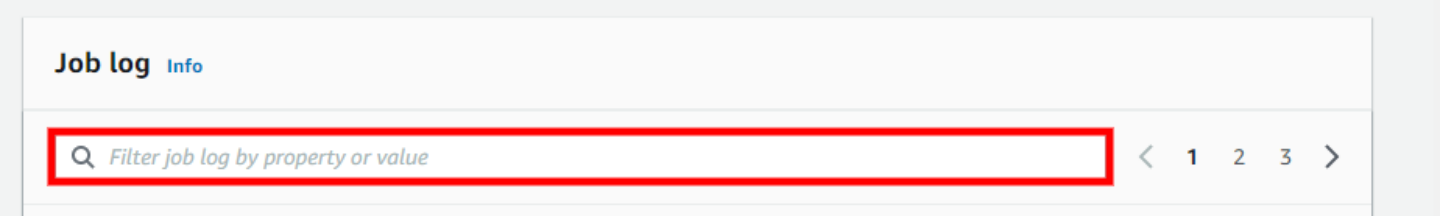
Job log

The Job log section shows a detailed log of all of the operations performed during the Job.

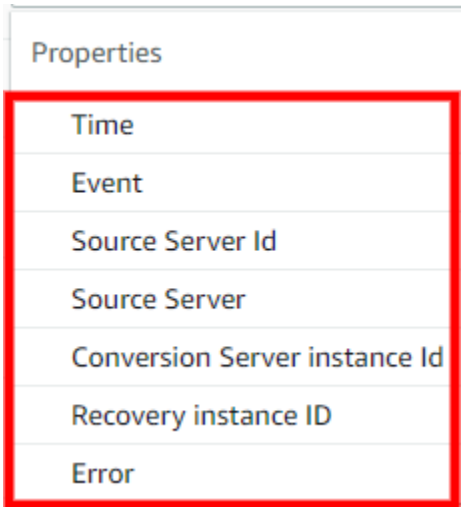


You can use this section to troubleshoot any potential issues and determine in which step of the launch process they occurred.

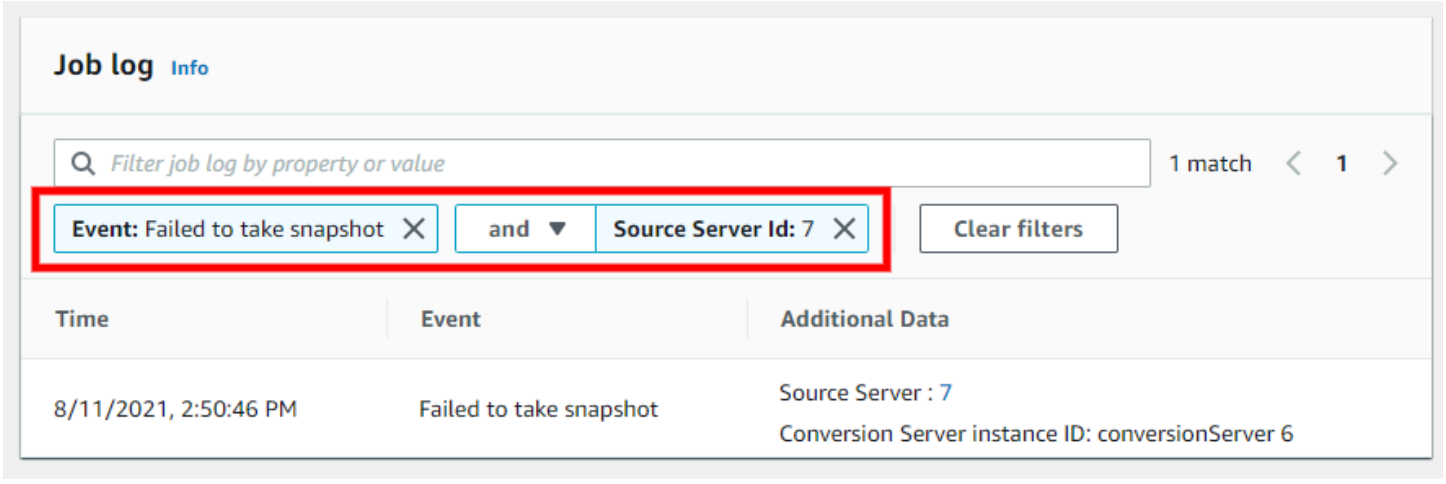
You can use the **Filter job log by property or value** search bar to filter the Job log.



You can filter by a variety of properties, including **Time, Event, Source Server Id, Source server hostname, Conversion Server instance Id, Drill/Recovery instance ID,** and **Error.**



You can filter by multiple values at once (for example, Job log filtered by **Event: Failed to take snapshot** and a specific **Source Server Id: 7**).



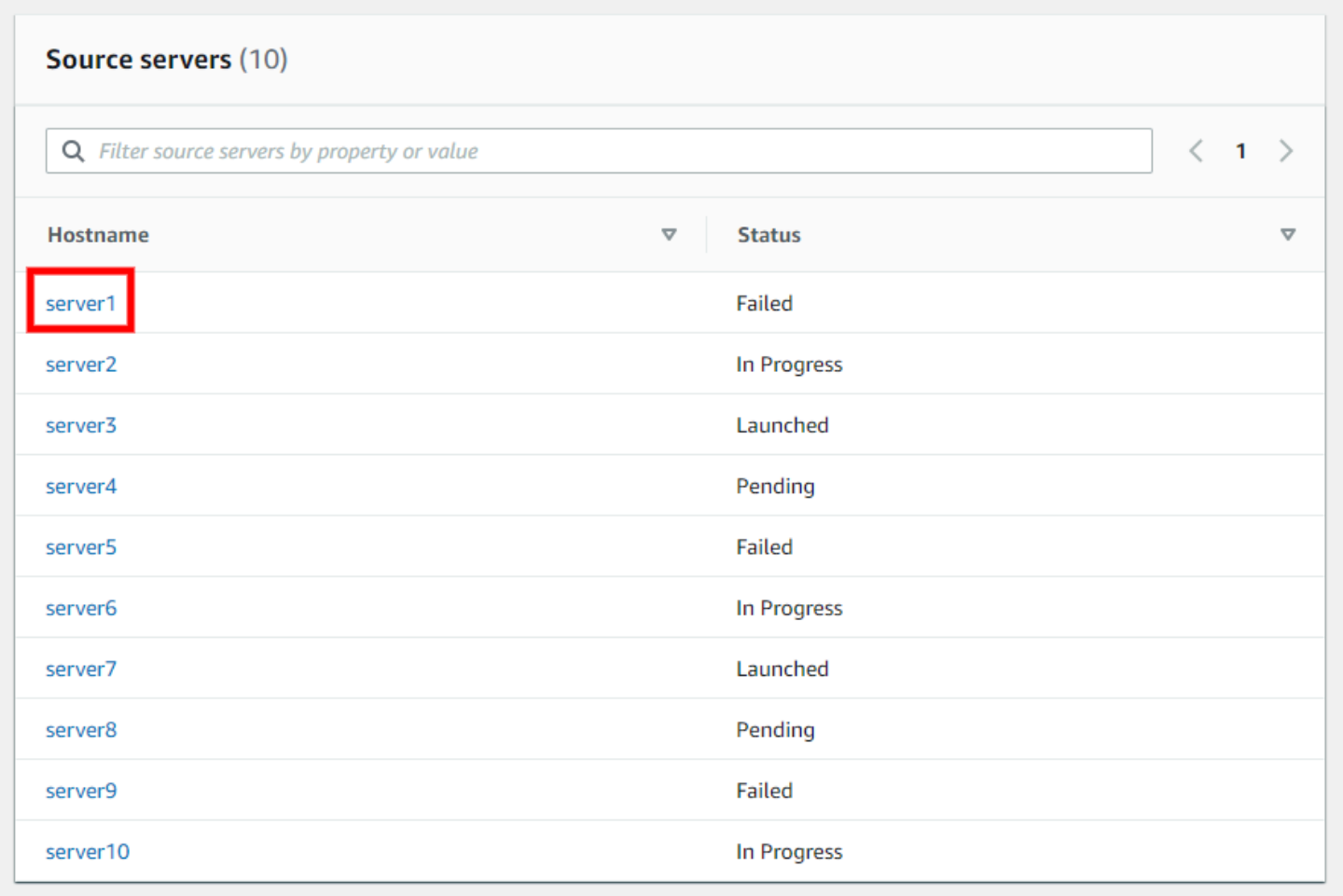
Jobs - Source servers

The Source servers section shows a list of all source servers involved in the Job and their status.

You can use the **Filter source servers by property or value** search bar to filter by **Hostname** or **Status**.



Choose the Hostname of any of Source server from the list to open the Server Details view for that server. Learn more about the Source Server details view.



Using multiple staging accounts with AWS DRS

AWS Elastic Disaster Recovery (AWS DRS) currently limits customers to 300 replicating source servers per account per AWS Region, due to various storage and API limitations. Customers who want to replicate and protect more than 300 source servers can use multiple staging accounts for replication, and recover their source servers into these accounts or into a single target AWS account. Customers who want to recover into a target account can manage the recovery for all the source servers in the staging accounts from that target account.

Use cases for this feature:

- You have more than 300 replicating servers and want to manage them from a single account.
- You have multiple AWS accounts with any number of servers and want to manage these servers from a single account.
- You want to manage your source servers in different AWS accounts for various business or security reasons and want to manage them from a single account.
- You have replicating servers that you would like to be able to recover to multiple different AWS accounts.

Overview

The multiple staging account feature is configured similarly to the standard AWS DRS configuration, but includes several extra steps required to configure the target AWS account.

For each staging account, you must first:

1. Initialize AWS DRS.
2. Define your replication configuration template.
3. Install the AWS Replication Agent on each source server.
4. Configure the individual source server replication settings.
5. Share the EBS encryption key with the target account.
6. Create a role to allow access into the staging account from the target account

For each target account, you must first:

1. Initialize the target account.

Once all of your source servers have been added to your staging accounts and are replicating successfully (are in the **Healthy** data replication state), you can use the target AWS account to launch Drill and Recovery instances for each server.

> **Note**
>
> You can only update the default replication template for the source servers from the staging account and not from the target account. Also, disconnection and deletion of the staging account's source servers are done from the staging account (to stop replication and save on resource usage). Source servers can be extended into many target accounts, or deleted from them.
>
> Source servers that reside in the staging account but are managed in the target account are called "extended source servers". An extended source server for which the staging source server has been deleted, or the role revoked, will remain in the target account, but will be marked with an extension error. An extended source server can be deleted at any time from the target account.
>
> Source servers that are EC2 instances, and have one or more marketplace licenses associated, cannot be extended into the target AWS account, unless the source AWS account (the AWS account that owns the EC2 instance) creates a failback and in-AWS account role for the staging account. This is required to provide permissions to get the marketplace license data from the source account when the server is extended. Create a Failback and in-AWS right-sizing role for trusted account for any staging account on the source account (the AWS account that owns the EC2 instance).

On a target account, the source servers list view shows all the source servers that were extended into the account, or those that are replicating in it.

Extending source servers from a staging account into a target AWS account

You can extend source servers from both new and existing AWS DRS accounts into a target AWS account.

Onboarding a new staging account

To use an account as a staging account in any AWS Region, you must first initialize AWS DRS in the AWS Region of the staging account, and add roles for the target account or accounts you plan to use.

During initialization, you will need to define the default replication settings, as described in the quick start guide.

> **Note**
>
> If your volumes are encrypted, you must use a custom encryption using a customer managed key when defining the EBS encryption. This key must be shared with the target account (see instructions below), to facilitate recovery in the target account.

After the initialization of the staging account, add IAM roles for the target accounts on the **Settings: trusted accounts** page of AWS DRS in the staging account. The roles are used to allow the target account to extend source servers from the staging account and to recover them in the target account.

> **Note**
>
> Commercial AWS accounts can only be extended to other Commercial AWS accounts and GovCloud AWS accounts can only be extended to other GovCloud AWS accounts.

Under trusted accounts settings, you will find the **Existing trusted accounts** pane. Here, you can manage existing staging account IAM roles. These IAM roles are used to associate the staging account with the target account.

Use **Add trusted accounts and create roles** to add roles for any trusted account you plan to use:

1. On the **Add trusted accounts and create roles** page, choose **Add new trusted account**.
2. Enter the AWS account IDs of the trusted account and select **staging role**. This automatically generates a service IAM role that will allow the use of Amazon EC2, Amazon EBS and AWS DRS resources in the staging account on behalf of a trusted account's user.

3. Choose **Add new trusted account** to add more than one trusted account at once. You can add up to 10 trusted accounts at once.
4. Once you have added your accounts, choose **Add trusted accounts and create roles**.

Using an existing account as a staging account

To use an account as a staging account, the default replication settings and replication settings of each source server that is to be extended into a target account should be reviewed, and EBS encryption must be set to use custom encryption using a customer managed key.

> **Note**
>
> This may trigger a full resync of the replicated data for a source server that had the default key, if that source server's encryption key was modified.

Share the customer managed key (or keys) with the target account (as described below).

IAM roles are automatically created for the target accounts on the **Settings: trusted account** page of AWS DRS in the staging account. These roles are used to allow the target account to extend source servers from the staging account and to recover them in the target account.

Under trusted accounts settings, you will find the **Existing trusted accounts** category. Here, you can manage existing staging accounts with links to IAM roles associated for each account. These IAM roles are used to associate the staging account with the trusted account.

Use **Add trusted accounts and create roles** to add roles for any trusted account you plan to use:

1. On the **Add trusted accounts and create roles** page, choose **Add new trusted account**.
2. Enter the AWS account IDs of the trusted account and select staging role. This will automatically generate a service IAM role that will allow the use of Amazon EC2, Amazon EBS and AWS DRS resources in the staging account on behalf of a trusted account's user.
3. Choose **Add new trusted account** to add more than one trusted account at once. You can add up to 10 trusted accounts at once.
4. Once you have added your accounts, choose **Add trusted accounts and create roles**.

Share the EBS encryption key with the target account

Sharing the EBS encryption key is mandatory only if your volumes are encrypted.

In order for the target account to be able to successfully read the EBS snapshots of the replication servers in the staging account, you must share the EBS encryption key configured in the staging account with the target account. This can be done by following the instructions in the Allowing users in other accounts to use a KMS key documentation.

You must set the following statement policies on your staging account's KMS key in order to be able to recover extended source servers on a specific target account. Ensure that you properly assign the `$STAGING_ACCOUNT_ID` and `$TARGET_ACCOUNT_ID` and `$REGION` variables.

Note that if this is a key you already have been using, you will need to attach this policy in addition to the existing one.

```
[
 {
  "Sid": "Allow access to share snapshots with a target account",
  "Effect": "Allow",
  "Principal": {
   "AWS": [
    "arn:aws:iam::$STAGING_ACCOUNT_ID:role/service-role/DRSStagingAccountRole_
$TARGET_ACCOUNT_ID"
   ]
  },
  "Action": "kms:ReEncrypt*",
  "Resource": "*",
  "Condition": {
   "StringEquals": {
    "kms:CallerAccount": "$STAGING_ACCOUNT_ID",
    "kms:ViaService": "ec2.$REGION.amazonaws.com"
   }
  }
 },
 {
  "Sid": "Allow a target account to use this KMS key via EC2",
  "Effect": "Allow",
  "Principal": {
   "AWS": "arn:aws:iam::$TARGET_ACCOUNT_ID:root"
```

```
  },
  "Action": "kms:ReEncrypt*",
  "Resource": "*",
  "Condition": {
   "StringEquals": {
    "kms:CallerAccount": "$TARGET_ACCOUNT_ID",
    "kms:ViaService": "ec2.$REGION.amazonaws.com"
   }
  }
 }
]
```

Managing extended source servers within the target AWS account

In order to manage extended source servers within the target accounts, you should extend source servers you wish to recover in the target account into that account from any staging account.

Initializing the target account

If you plan on using an AWS account and AWS Region in which AWS DRS has not been initialized, the service can be initialized either from the AWS DRS console or from the API. If you choose to initialize the service from the API, using the InitializeService API, you can skip creating the default replication settings if you plan to use the service only from the API and do not plan to have source servers replicating on this account. If you initialize the service through the AWS DRS console, the initialization wizard still creates the default replication settings, and the wizard will also run if you use the console after initializing a service without creating the default replication settings.

Create extended source servers

To add your source servers from your staging accounts into your target account you must extend the source servers from the staging account to the target account.

> **Important**
>
> Repeat the steps for every staging account you want to associate with the target account.

1. Navigate to the **Source servers** view within the target account, open the **Actions** menu, and choose **Create extended source servers** to start the **Create extended source servers** wizard.
2. Under **Configure access > Staging account configuration**, enter the ID of the staging account in which you created the IAM roles in the previous step.
3. Select the source servers you want to extend from the staging account into the target account **Hostname**. This will create a new source server resource that will inherit the replication configuration and points in time from the base source server in the staging account. Only source servers that have not already been extended will be shown. Once you have selected your source servers, choose **Next**.

   > **Note**
   >
   > The Extend source servers page will only show 30 source servers per page. If you have many source servers in your staging account and want to extend them all to your target account, then choose the **Select all X source servers** option.
   > You can filter the source servers shown by Hostname or Source Server ID through the **Filter....** box.

   Optional: Add **Tags**. Learn more about adding tags in AWS DRS.
4. **Review and create** the extended source servers. Review the information on the page and then choose **Create extended source servers**.

The AWS DRS console shows the **Successfully created X extended source servers** message and you will see your extended source servers in your target account.

> **Note**
>
> Extended source servers will show **Extended** under **Extension status** in the **Source server details > Overview section.**

Manage source servers

Once you have extended your source servers from every staging account into the target account, you can manage the source servers from the target account.

Source servers are grouped by staging account. You can choose the staging account under the **Source servers** header.

> **Note**
>
> If you want to see the Source Server ID and Staging Account ID of each source server in the **source servers** view, you can add those columns by choosing the **Preferences** wheel. From **Preferences**, toggle the **Source server ID** and **Staging account ID** options and choose **Confirm**.

You can now perform Elastic Disaster Recovery operations for the source servers, including:

- Configuring individual source server settings
- Configure launch settings
- Launching Drill and Recovery instances
- Performing a failover and failback

> **Note**
>
> You will not be able to edit the replication settings and disks for individual extended source servers from the target account. You must edit these from the staging account.

> **Note**
>
> The AWS Replication Agent will stop replicating automatically after failing back from a recovery instance of an extended source server to the original server.

Removing an extended source server

If you need to delete an extended source server, do this from the account it was extended to (and where it is no longer needed). Deleting an extended source server has no effect on the replication of the source server into the staging account.

You can always recreate an extended source server after it was deleted, using 'Create extended source server' on the same staging source server.

> **Note**
>
> To delete the staging account source server (the source server that is used to replicate data into the staging account), it must first be disconnected, and then it can be deleted from the staging account.

> **Note**
>
> You cannot change the staging account (the account where replication takes place) for a server that has been extended. For example, if a source server is replicating into account A, and has been extended into account B, you cannot reinstall the agent on the source server to replicate into an account that is different than A while it is still extended into account B.

Troubleshooting

If your source server shows **Extension error** under the **Ready for recovery** category in the target account, then the source server was most likely deleted from the staging account.

Navigate to the source server details page by choosing the server's hostname in order to see the extension error details.

Working with AWS DRS and AWS Outposts

AWS Elastic Disaster Recovery supports AWS Outpost racks. AWS Outposts require specific configurations in the default replication settings, individual source server replication settings, default launch templates and source server launch templates to work. The following sections explain how to use AWS Outpost racks with DRS, how to troubleshoot key AWS Outpost issues, and how to monitor AWS Outposts with DRS. Learn more about AWS Outposts.

> **Note**
>
> AWS Elastic Disaster Recovery does not support AWS Outpost servers.

Considerations when using AWS Outposts:

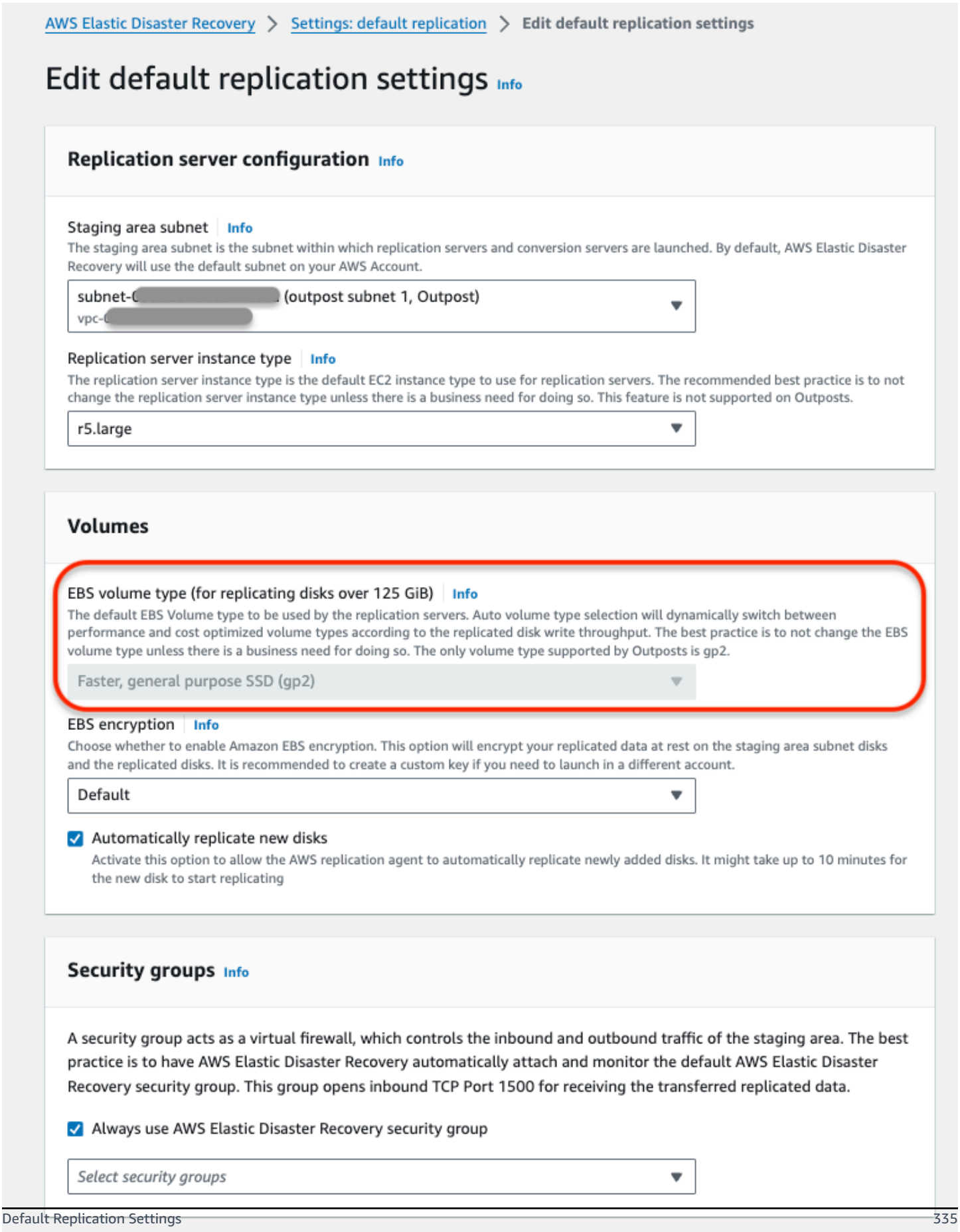
- To use an AWS Outpost, you need to have a subnet within the Outpost selected to be used for replication or recovery. If you select an Outpost subnet for replication, a subnet in the same Outpost must be selected for recovery.
- Your Outpost must be in direct VPC routing mode. For more information see Direct VPC routing in the *AWS Outposts user guide*.
- Once a subnet within an AWS Outpost is selected for replication, all replication resources (including replication servers, conversion servers, EBS volumes, and snapshots) will be created and saved within the Outpost.
- If a subnet within an AWS Outpost is selected for launch, then the recovery instances, EBS volumes and snapshots used for recovery are created and saved within the Outpost.
- Your S3 storage must be locally deployed on the Outpost rack. Remote S3 storage is not supported.

Default Replication Settings

Selecting an Outpost subnet on the *Settings: default replication* page means that newly added source servers will automatically start replicating to the Outpost.

Subnets that are within Outposts will have the word "Outpost" appended after the subnet name in the subnet selector drop down.

Once the Subnet is chosen, you will have to select the replication server instance type. Only instance types that are supported by the chosen Outpost will be shown.


AWS Elastic Disaster Recovery > Settings: default replication > Edit default replication settings
Edit default replication settings Info
Replication server configuration Info
Staging area subnet Info
The staging area subnet is the subnet within which replication servers and conversion servers are launched. By default, AWS Elastic Disaster Recovery will use the default subnet on your AWS Account.
subnet- (outpost subnet 1, Outpost)
vpc-
Replication server instance type Info
The replication server instance type is the default EC2 instance type to use for replication servers. The recommended best practice is to not change the replication server instance type unless there is a business need for doing so. This feature is not supported on Outposts.
r5.large
Volumes
EBS volume type (for replicating disks over 125 GiB) Info
The default EBS Volume type to be used by the replication servers. Auto volume type selection will dynamically switch between performance and cost optimized volume types according to the replicated disk write throughput. The best practice is to not change the EBS volume type unless there is a business need for doing so. The only volume type supported by Outposts is gp2.
Faster, general purpose SSD (gp2)
EBS encryption Info
Choose whether to enable Amazon EBS encryption. This option will encrypt your replicated data at rest on the staging area subnet disks and the replicated disks. It is recommended to create a custom key if you need to launch in a different account.
Default
Automatically replicate new disks
Activate this option to allow the AWS replication agent to automatically replicate newly added disks. It might take up to 10 minutes for the new disk to start replicating
Security groups Info
A security group acts as a virtual firewall, which controls the inbound and outbound traffic of the staging area. The best practice is to have AWS Elastic Disaster Recovery automatically attach and monitor the default AWS Elastic Disaster Recovery security group. This group opens inbound TCP Port 1500 for receiving the transferred replicated data.
Always use AWS Elastic Disaster Recovery security group
Select security groups

Outposts only support GP2 disks. As such, you will not be able to change the default disk type in the replication settings.

All Outpost volumes must be encrypted, and so you must select an encryption key when a subnet within an Outpost is selected. You will not be able to select not to encrypt. Ensure that you choose the correct volume encryption key you want to use. You can use the default KMS key for your account, or select a customer managed key (CMK).

Other replication settings can be set normally. Learn more about default replication settings.

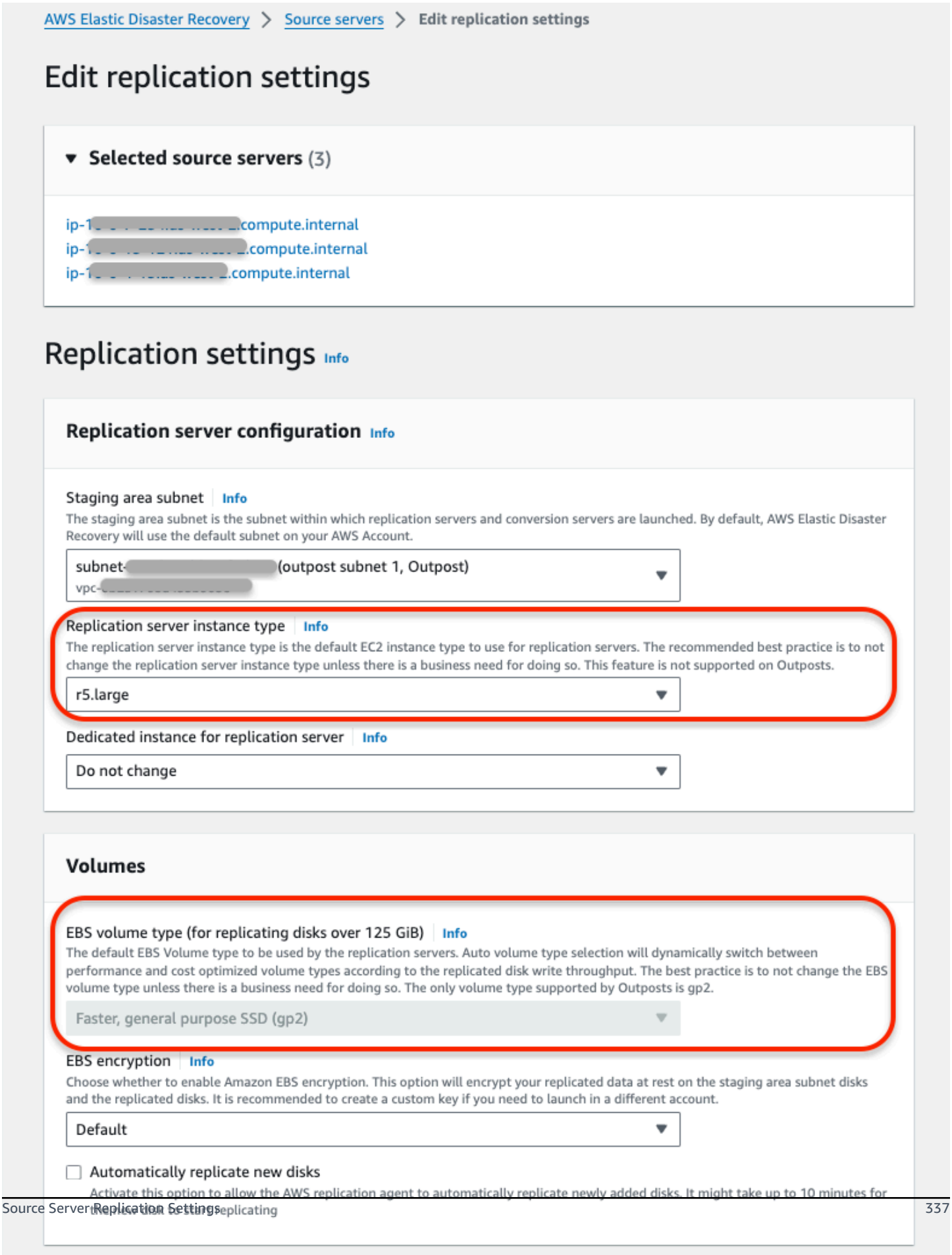
> **Note**
>
> When selecting an Outpost subnet in your replication settings, ensure that your launch template specifies a subnet on the same Outpost. Not doing so may result in an error during recovery.

Source Server Replication Settings

Use these settings to have a specific source server or multiple source servers replicate into an AWS Outpost by selecting a subnet within an AWS Outpost.

Subnets that are within Outposts will have the word "Outpost" appended after the subnet name in the subnet selector drop down.

Once the Subnet is chosen, you will have to select the replication server instance type. Only instance types that are supported by the chosen Outpost will be shown.


AWS Elastic Disaster Recovery > Source servers > Edit replication settings
Edit replication settings
Selected source servers (3)
ip-1[redacted].compute.internal
ip-1[redacted].compute.internal
ip-1[redacted].compute.internal
Replication settings Info
Replication server configuration Info
Staging area subnet Info
The staging area subnet is the subnet within which replication servers and conversion servers are launched. By default, AWS Elastic Disaster Recovery will use the default subnet on your AWS Account.
subnet-[redacted] (outpost subnet 1, Outpost)
vpc-[redacted]
Replication server instance type Info
The replication server instance type is the default EC2 instance type to use for replication servers. The recommended best practice is to not change the replication server instance type unless there is a business need for doing so. This feature is not supported on Outposts.
r5.large
Dedicated instance for replication server Info
Do not change
Volumes
EBS volume type (for replicating disks over 125 GiB) Info
The default EBS Volume type to be used by the replication servers. Auto volume type selection will dynamically switch between performance and cost optimized volume types according to the replicated disk write throughput. The best practice is to not change the EBS volume type unless there is a business need for doing so. The only volume type supported by Outposts is gp2.
Faster, general purpose SSD (gp2)
EBS encryption Info
Choose whether to enable Amazon EBS encryption. This option will encrypt your replicated data at rest on the staging area subnet disks and the replicated disks. It is recommended to create a custom key if you need to launch in a different account.
Default
Automatically replicate new disks
Activate this option to allow the AWS replication agent to automatically replicate newly added disks. It might take up to 10 minutes for
[redacted] replicating

Outposts only support GP2 disks. As such, you will not be able to change the default disk type in the replication settings.

All Outpost volumes must be encrypted and you must select an encryption key when a subnet within an Outpost is selected. You will not be able to select *not to encrypt*. Ensure that you choose the correct volume encryption key you want to use. You can use the default KMS key for your account or select a customer managed key (CMK).

Other replication settings can be set normally. Learn more about replication settings.

Note

You cannot edit the replication settings of multiple source servers if some of the source servers are replicating to Outpost subnets and others are replicating to non-Outpost subnets.

Note

When selecting an Outpost subnet in your replication settings, ensure that your launch template specifies a subnet on the same Outpost. Not doing so may result in an error during recovery.

Default Launch Template

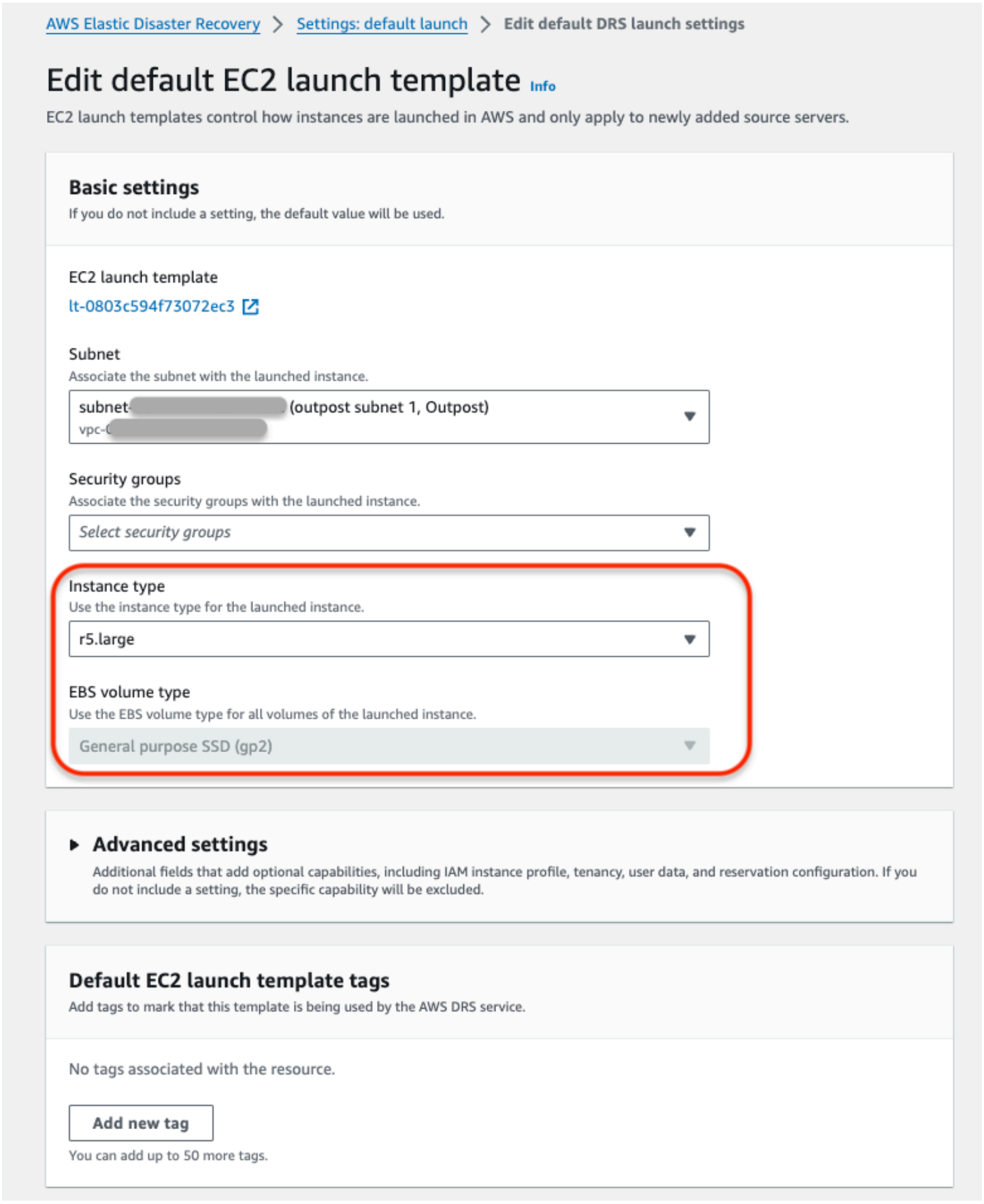
Selecting an Outpost subnet in the default launch template means that newly added source servers will launch into the Outpost. You must select the instance type from the list of available instance types on your Outpost.

Note

When working with Outposts, if you selected a subnet within an AWS Outpost in the default replication settings, you must also choose a subnet within the same Outpost in the default launch template. Otherwise, newly added source servers will replicate into an Outpost but launch outside of an Outpost which may result in an error.

> **Note**
>
> Network replication and recovery does not create a subnet within the Outpost.

Outposts only support GP2 disks. As such, you will not be able to change the default disk type per volume in the default launch template.

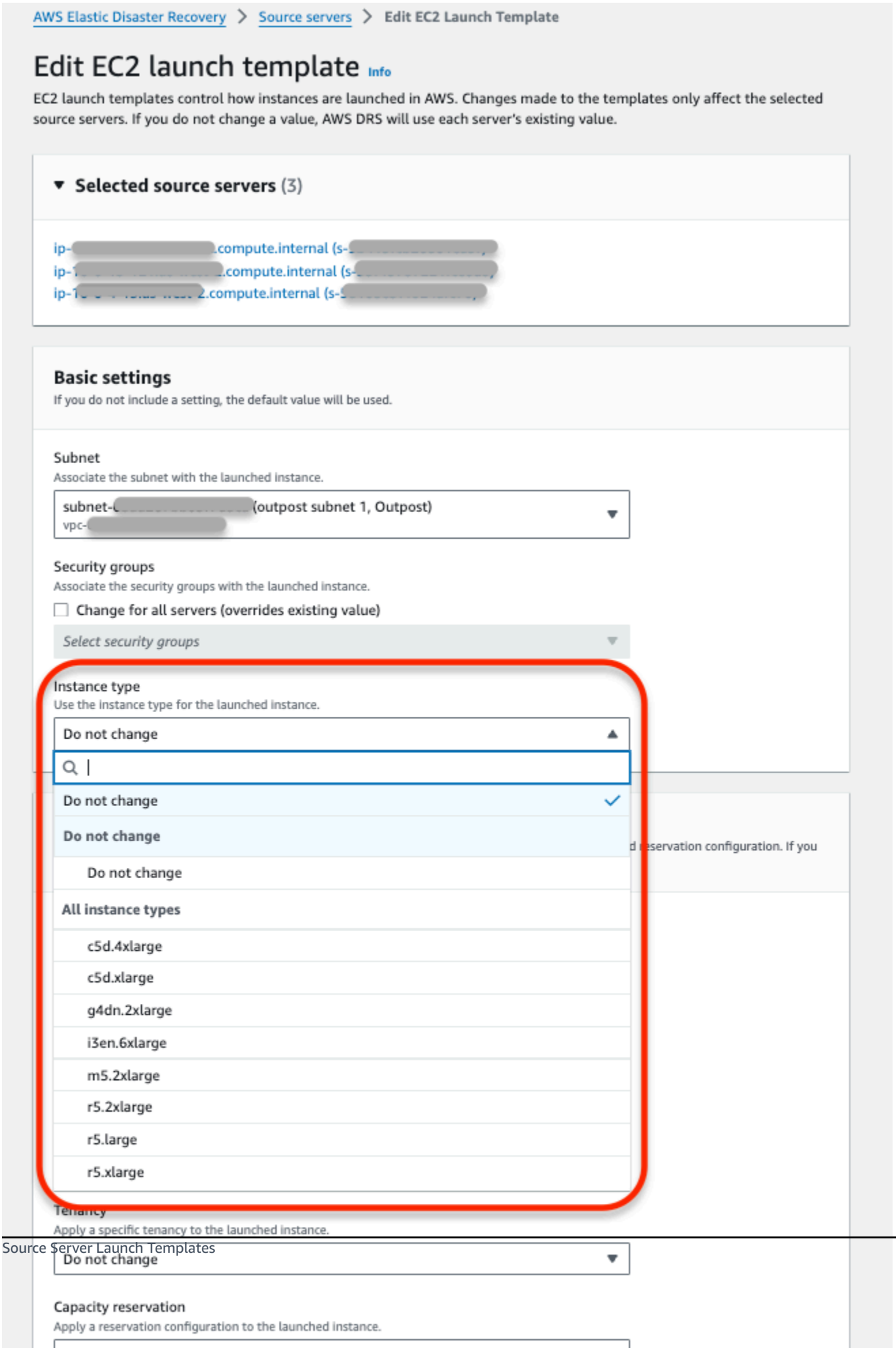

AWS Elastic Disaster Recovery > Settings: default launch > Edit default DRS launch settings
Edit default EC2 launch template Info
EC2 launch templates control how instances are launched in AWS and only apply to newly added source servers.
Basic settings
If you do not include a setting, the default value will be used.
EC2 launch template
lt-0803c594f73072ec3
Subnet
Associate the subnet with the launched instance.
subnet- (outpost subnet 1, Outpost)
vpc-
Security groups
Associate the security groups with the launched instance.
Select security groups
Instance type
Use the instance type for the launched instance.
r5.large
EBS volume type
Use the EBS volume type for all volumes of the launched instance.
General purpose SSD (gp2)
Advanced settings
Additional fields that add optional capabilities, including IAM instance profile, tenancy, user data, and reservation configuration. If you do not include a setting, the specific capability will be excluded.
Default EC2 launch template tags
Add tags to mark that this template is being used by the AWS DRS service.
No tags associated with the resource.
Add new tag
You can add up to 50 more tags.

Source Server Launch Templates

When working with Outposts, if you selected a subnet within an AWS Outpost in the replication settings for any source server, you must also choose a subnet within the same Outpost in the launch template of that source server. You must select the instance type to launch into from the list of available instance types on your Outpost. Not doing so may result in an error during recovery.

> **Note**
>
> Network replication and recovery does not create a subnet within the Outpost.

Outposts only support GP2 disks. As such, you will not be able to change the default disk type per volume in the default launch template.

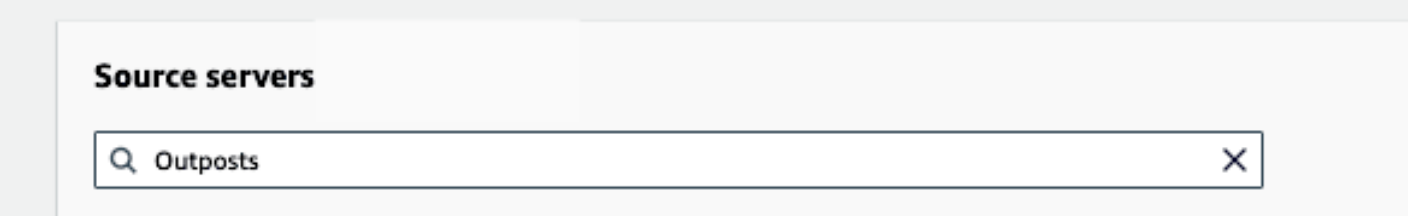

AWS Elastic Disaster Recovery > Source servers > Edit EC2 Launch Template
Edit EC2 launch template Info
EC2 launch templates control how instances are launched in AWS. Changes made to the templates only affect the selected source servers. If you do not change a value, AWS DRS will use each server's existing value.
Selected source servers (3)
Basic settings
If you do not include a setting, the default value will be used.
Subnet
Associate the subnet with the launched instance.
(outpost subnet 1, Outpost)
Security groups
Associate the security groups with the launched instance.
Change for all servers (overrides existing value)
Select security groups
Instance type
Use the instance type for the launched instance.
Do not change
Do not change
Do not change
Do not change
All instance types
c5d.4xlarge
c5d.xlarge
g4dn.2xlarge
i3en.6xlarge
m5.2xlarge
r5.2xlarge
r5.large
r5.xlarge
Apply a specific tenancy to the launched instance.
Do not change
Capacity reservation
Apply a reservation configuration to the launched instance.

> **Note**
>
> You cannot edit the launch templates of multiple source servers if some of the source servers are configured to launch to Outpost subnets and the others are configured to launch to non-Outpost subnets.

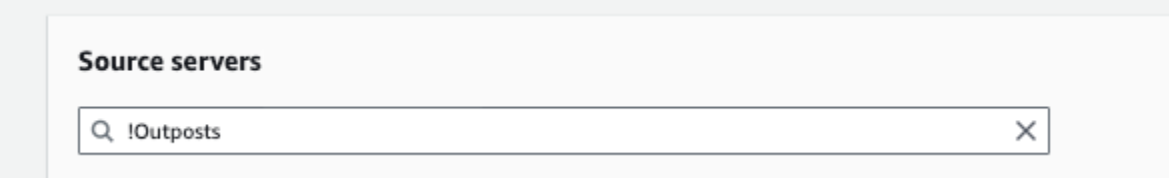
Source Server Page

You can find which of your servers are replicating into an AWS Outpost rack, as these are marked with the value (Outpost) in the **Replicating to** field. The following are search options for source servers replicating to an Outpost:

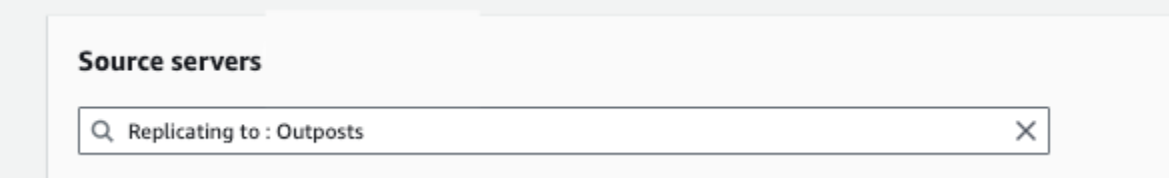
- You can enter the text "*Outpost*" in the search field to only display source servers that are replicating to an AWS Outpost rack.



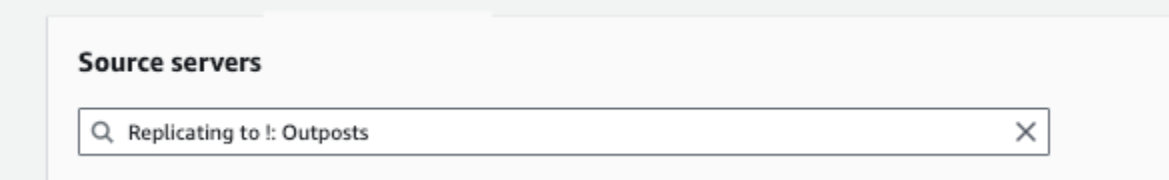
- You can enter the text "*!Outpost*" to only display source servers that are not replicating to an AWS Outpost rack.



- You can enter "*Replicating to: Outpost*" in the search field to only display source servers that are replicating to an AWS Outpost rack.



- You can enter "*Replicating to !: Outpost*" in the search field to only display source servers that are not replicating to an AWS Outpost rack.



Important Outpost Notes

Outpost Storage

AWS Outposts have a certain storage capacity. You should actively monitor available storage capacity in your Outpost. **If you run out of storage capacity, you may not be able to use DRS with that Outpost environment.** For example, if S3 runs out of capacity on the Outpost then DRS won't be able to create any new snapshots, so no new points in time will be created.

Be aware of EBS storage capacity on the Outpost. **DRS requires storage capacity roughly on a 2:1 ratio (excluding any launched instances on the Outpost).** For example, if you have source volumes amounting to a total of 1 TiB of storage, then DRS will require at least 1 TiB for replication resources, and another 1 TiB for conversion or recovery instance creation.

Replication and Launch Subnets

When using DRS with Outposts it is imperative that the subnet in replication settings and launch settings are on the same Outpost. If the subnets are not on the same Outpost, it may result in a failure during recovery.

Instance Types and Operating Systems (OSs)

> **Note**
>
> AWS Outposts only support Nitro instance types.

- **Linux** - RHEL 7.0+ and CentOS 7.0+
- **Windows** - Windows Server 2008 R2, Windows Server 2012 R2, Windows Server 2016, and higher.

Monitoring

You can monitor Outpost metrics for S3, EC2, and EBS capacity through CloudWatch. You can create a dashboard with these metrics:

- EBS Metrics: EBSVolumeTypeCapacityAvailability, EBSVolumeTypeCapacityUtilization
- EC2 Metrics: InstanceTypeCapacityUtilization, AvailableInstanceType_Count
- S3 Metrics: OutpostTotalBytes, OutpostFreeBytes

Learn more about CloudWatch S3 monitoring.

Learn more about CloudWatch Metrics for AWS Outposts.

Security in AWS Elastic Disaster Recovery

Overview

Cloud security at AWS is the highest priority. As an AWS customer, you benefit from a data center and network architecture that is built to meet the requirements of the most security-sensitive organizations.

Security is a shared responsibility between AWS and you. The shared responsibility model describes this as security of the cloud and security in the cloud:

- **Security of the cloud** – AWS is responsible for protecting the infrastructure that runs AWS services in the AWS Cloud. AWS also provides you with services that you can use securely. Third-party auditors regularly test and verify the effectiveness of our security as part of the AWS Compliance Programs . To learn about the compliance programs that apply to AWS Elastic Disaster Recovery (AWS DRS), see AWS Services in Scope by Compliance Program .
- **Security in the cloud** – Your responsibility is determined by the AWS service that you use. You are also responsible for other factors including the sensitivity of your data, your company's requirements, and applicable laws and regulations

This documentation helps you understand how to apply the shared responsibility model when using AWS DRS. It shows you how to configure AWS DRS to meet your security and compliance objectives. You also learn how to use other AWS services that help you to monitor and secure your AWS Elastic Disaster Recovery resources.

The customer is responsible for making sure that no misconfigurations are present during and after the recovery process, including:

1. The replication server should be accessed only from the CIDR range of the source servers. Proper security groups rules should be assigned to the replication server after it is created.
2. After the recovery, the customer should make sure that on the recovery instances only allowed ports are exposed to the public internet.
3. Hardening of OS packages and other software deployed in the recovery instances is completely under the customer's responsibility and we recommend the following:

   a. Packages should be up to date and free of known vulnerabilities.

b. Only necessary OS/application services should be up and running.

4. Activating the Anti-DDOS protection (AWS Shield) in the customer's AWS Account to eliminate the risk of denial of service attacks on the replication servers as well as the migrated servers.

Identity and access management for AWS Elastic Disaster Recovery

AWS Identity and Access Management (IAM) is an AWS service that helps an administrator securely control access to AWS resources. IAM administrators control who can be authenticated (signed in) and authorized (have permissions) to use AWS resources. IAM allows you to create users and groups under your AWS account. You control the permissions that users have to perform tasks using AWS resources. You can use IAM for no additional charge.

By default, IAM users don't have permissions for AWS Elastic Disaster Recovery (AWS DRS) resources and operations. To allow IAM users to manage AWS DRS resources, you must create an IAM policy that explicitly grants them permissions, and attach the policy to the users or groups that require those permissions.

When you attach a policy to a user or group of users, it allows or denies the users permission to perform the specified tasks on the specified resources. For more information, see Policies and Permissions in the *IAM User Guide* guide.

Federated identity

As a best practice, require human users to use federation with an identity provider to access AWS services using temporary credentials.

A *federated identity* is a user from your enterprise directory, web identity provider, or Directory Service that accesses AWS services using credentials from an identity source. Federated identities assume roles that provide temporary credentials.

For centralized access management, we recommend AWS IAM Identity Center. For more information, see What is IAM Identity Center? in the *AWS IAM Identity Center User Guide*.

Authenticating with identities in AWS Elastic Disaster Recovery

Authentication is how you sign in to AWS using your identity credentials. You must be authenticated as the AWS account root user, an IAM user, or by assuming an IAM role.

You can sign in as a federated identity using credentials from an identity source like AWS IAM Identity Center (IAM Identity Center), single sign-on authentication, or Google/Facebook credentials. For more information about signing in, see How to sign in to your AWS account in the *AWS Sign-In User Guide*.

For programmatic access, AWS provides an SDK and CLI to cryptographically sign requests. For more information, see AWS Signature Version 4 for API requests in the *IAM User Guide*.

AWS account root user

When you create an AWS account, you begin with one sign-in identity called the AWS account *root user* that has complete access to all AWS services and resources. We strongly recommend that you don't use the root user for everyday tasks. For tasks that require root user credentials, see Tasks that require root user credentials in the *IAM User Guide*.

IAM users and groups

An *IAM user* is an identity with specific permissions for a single person or application. We recommend using temporary credentials instead of IAM users with long-term credentials. For more information, see Require human users to use federation with an identity provider to access AWS using temporary credentials in the *IAM User Guide*.

An *IAM group* specifies a collection of IAM users and makes permissions easier to manage for large sets of users. For more information, see Use cases for IAM users in the *IAM User Guide*.

IAM roles

An *IAM role* is an identity with specific permissions that provides temporary credentials. You can assume a role by switching from a user to an IAM role (console) or by calling an AWS CLI or AWS API operation. For more information, see Methods to assume a role in the *IAM User Guide*.

IAM roles are useful for federated user access, temporary IAM user permissions, cross-account access, cross-service access, and applications running on Amazon EC2. For more information, see Cross account resource access in IAM in the *IAM User Guide*.

Grant permission to tag resources during creation

Some resource-creating Amazon DRS API actions allow you to specify tags when you create the resource. You can use resource tags to implement attribute-based access control (ABAC).

To allow users to tag resources on creation, they must have permissions to use the action that creates the resource, such as:

- `drs:CreateSourceServerForDrs` – for creating a source server
- `drs:CreateRecoveryInstanceForDrs` – for creating a Recovery instance
- `drs:TagResource` – if tags are specified in the resource-creating action

If tags are specified in the resource-creating action, Amazon DRS performs additional authorization on the `drs:TagResource` action to verify that users have permissions to create tags. Therefore, users must also have explicit permissions to use the `drs:TagResource` action.

In the IAM policy definition for the `drs:TagResource` action, use the Condition element with the `drs:CreateAction` condition key to give tagging permissions to the action that creates the resource.

The following example demonstrates a policy that allows an agent installer to create a source server or recovery instance and apply any tags to the resource on creation. The installer is not permitted to tag any existing resources (it cannot call the `drs:TagResource` action directly).

JSON

```
{
 "Version":"2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "drs:GetAgentInstallationAssetsForDrs",
    "drs:SendClientLogsForDrs",
    "drs:CreateSourceServerForDrs",
    "drs:CreateRecoveryInstanceForDrs",
    "drs:DescribeRecoveryInstances"
   ],
   "Resource": "*"
  },
  {
   "Effect": "Allow",
   "Action": "drs:TagResource",
   "Resource": "arn:aws:drs:*:*:source-server/*",
```

```
    "Condition": {
     "StringEquals": {
      "drs:CreateAction": "CreateSourceServerForDrs"
     }
    }
   },
   {
    "Effect": "Allow",
    "Action": "drs:TagResource",
    "Resource": "arn:aws:drs:*:*:recovery-instance/*",
    "Condition": {
     "StringEquals": {
      "drs:CreateAction": "CreateRecoveryInstanceForDrs"
     }
    }
   },
   {
    "Effect": "Allow",
    "Action": "drs:IssueAgentCertificateForDrs",
    "Resource": "arn:aws:drs:*:*:source-server/*"
   }
  ]
}
```

The `drs:TagResource` action is only evaluated if tags are applied during the resource-creating action. Therefore, an installer that has permissions to create a resource (assuming there are no tagging conditions) does not require permissions to use the `drs:TagResource` action if no tags are specified in the request. However, if the installer attempts to create a resource with tags, the request fails if the installer does not have permissions to use the `drs:TagResource` action.

AWS managed policies for AWS Elastic Disaster Recovery

An AWS managed policy is a standalone policy that is created and administered by AWS. AWS managed policies are designed to provide permissions for many common use cases so that you can start assigning permissions to users, groups, and roles.

Keep in mind that AWS managed policies might not grant least-privilege permissions for your specific use cases because they're available for all AWS customers to use. We recommend that you reduce permissions further by defining customer managed policies that are specific to your use cases.

You cannot change the permissions defined in AWS managed policies. If AWS updates the permissions defined in an AWS managed policy, the update affects all principal identities (users, groups, and roles) that the policy is attached to. AWS is most likely to update an AWS managed policy when a new AWS service is launched or new API operations become available for existing services.

For more information, see AWS managed policies in the *IAM User Guide*.

AWS managed policy: AWSElasticDisasterRecoveryAgentPolicy

This policy gives the AWS Replication Agent, which is used with AWS Elastic Disaster Recovery (AWS DRS) to replicate source servers to AWS, permissions to communicate with AWS DRS to receive instructions and to send logs and metrics.

> **Important**
>
> This policy is designed exclusively for the AWS Replication Agent. We do not recommend that you attach this policy to your IAM users or roles.

Permissions details

To view the policy permission details see AWSElasticDisasterRecoveryAgentPolicy in the AWS Managed Policy Reference Guide.

AWS managed policy: AWSElasticDisasterRecoveryAgentInstallationPolicy

This policy allows installing the AWS Replication Agent, which is used with AWS Elastic Disaster Recovery (AWS DRS) to recover external servers to AWS. Attach this policy to your users or roles whose credentials you provide during the installation step of the AWS Replication Agent.

Permissions details

To view the policy permission details see AWSElasticDisasterRecoveryAgentInstallationPolicy in the AWS Managed Policy Reference Guide.

AWS managed policy: AWSElasticDisasterRecoveryConversionServerPolicy

This policy is attached to the AWS Elastic Disaster Recovery Conversion Server's instance role. This policy allows AWS Elastic Disaster Recovery (AWS DRS) Conversion Servers, which are EC2 instances launched by AWS DRS, to communicate with the DRS service. An IAM role with this

policy is attached (as an EC2 Instance Profile) by DRS to the DRS Conversion Servers, which are automatically launched and terminated by DRS when needed. DRS Conversion Servers are used by AWS Elastic Disaster Recovery when users choose to recover source servers using the AWS DRS console, CLI, or API. We do not recommend that you attach this policy to your users or roles.

Permissions details

To view the policy permission details see AWSElasticDisasterRecoveryConversionServerPolicy in the AWS Managed Policy Reference Guide.

AWS managed policy: AWSElasticDisasterRecoveryFailbackPolicy

This policy allows using the AWS Elastic Disaster Recovery Failback Client, which is used to fail back Recovery Instances to your original source infrastructure. This policy is also used by AWS Elastic Disaster Recovery to refresh credentials for the Failback Client. We do not recommend that you attach this policy to your users or roles.

Permissions details

To view the policy permission details see AWSElasticDisasterRecoveryFailbackPolicy in the AWS Managed Policy Reference Guide.

AWS managed policy: AWSElasticDisasterRecoveryFailbackInstallationPolicy

You can attach the AWSElasticDisasterRecoveryFailbackInstallationPolicy policy to your IAM identities.

This policy allows installing the AWS Elastic Disaster Recovery Failback Client, which is used to failback Recovery Instances back to your original source infrastructure. Attach this policy to your users or roles whose credentials you provide when running the AWS Elastic Disaster Recovery Failback Client.

Permissions details

To view the policy permission details see AWSElasticDisasterRecoveryFailbackInstallationPolicy in the AWS Managed Policy Reference Guide.

AWS managed policy: AWSElasticDisasterRecoveryConsoleFullAccess

This policy provides full access to all public APIs of AWS Elastic Disaster Recovery (AWS DRS), as well as permissions to read KMS key, License Manager, Resource Groups, Elastic Load Balancing,

IAM, and EC2 information. It also includes EC2 actions that allow you to launch, delete, or modify replication servers and recovery instances. These EC2 actions are limited only to resources which the service creates with a specific AWS-only tag. Attach this policy to your users or roles.

AWSElasticDisasterRecoveryConsoleFullAccess includes access to your AWS managed keys. However, it does not include access to your customer managed keys, so if you use CMK you will need to add a policy statement to allow the usage of your KMS keys.

Permissions details

To view the policy permission details see AWSElasticDisasterRecoveryConsoleFullAccess in the AWS Managed Policy Reference Guide.

AWS managed policy: AWSElasticDisasterRecoveryReadOnlyAccess

You can attach the AWSElasticDisasterRecoveryReadOnlyAccess policy to your IAM identities.

This policy provides permissions to all read-only public APIs of AWS Elastic Disaster Recovery (AWS DRS), as well as some read-only APIs of other AWS services that are required to make full read-only use of the DRS console. This includes:

- **AWS Elastic Disaster Recovery (read-only)** – View all DRS resources such as Source Servers, Recovery Instances, Recovery Snapshots, recovery plans, recovery plan executions, and post-launch actions.
- **IAM (read-only)** – List IAM roles in your account.
- **EC2 (read-only)** – View EC2 instance details, launch templates, security groups, and subnets related to your recovery environment.
- **SSM (read-only)** – View Systems Manager configurations such as post-launch action settings and automation executions.

Attach this policy to your users or roles. This policy is ideal for team members who need visibility into your disaster recovery setup, such as auditors or monitoring teams, without the ability to make changes.

Permissions details

To view the policy permission details see AWSElasticDisasterRecoveryReadOnlyAccess in the AWS Managed Policy Reference Guide.

AWS managed policy: AWSElasticDisasterRecoveryReplicationServerPolicy

This policy is attached to the AWS Elastic Disaster Recovery replication server's instance role.

This policy allows the AWS Elastic Disaster Recovery (AWS DRS) replication servers, which are Amazon EC2 instances launched by Elastic Disaster Recovery, to communicate with the DRS service, and to create EBS snapshots in your AWS account. An IAM role with this policy is attached (as an EC2 instance profile) by AWS DRS to the AWS DRS replication servers which are automatically launched and terminated by AWS DRS, as needed. AWS DRS replication servers are used to facilitate data replication from your external servers to AWS, as part of the recovery process managed by AWS DRS. We do not recommend that you attach this policy to your users or roles.

Permissions details

To view the policy permission details see AWSElasticDisasterRecoveryReplicationServerPolicy in the AWS Managed Policy Reference Guide.

AWS managed policy: AWSElasticDisasterRecoveryRecoveryInstancePolicy

This policy is attached to the instance role of AWS Elastic Disaster Recovery's recovery instance.

This policy allows the AWS Elastic Disaster Recovery (AWS DRS) recovery instances, which are EC2 instances launched by AWS DRS - to communicate with the AWS DRS service, and to be able to failback to their original source infrastructure. An IAM role with this policy is attached (as an Amazon EC2 Instance Profile) by AWS DRS to the AWS DRS recovery instances. We do not recommend that you attach this policy to your users or roles.

Permissions details

To view the policy permission details see AWSElasticDisasterRecoveryRecoveryInstancePolicy in the AWS Managed Policy Reference Guide.

AWS managed policy: AWSElasticDisasterRecoveryServiceRolePolicy

This policy allows AWS Elastic Disaster Recovery to manage AWS resources on your behalf.

This policy is attached to the AWSServiceRoleForElasticDisasterRecovery role.

Permissions details

This policy includes permissions to do the following:

- ec2 – Retrieve and modify resources needed to support failover and failback of source servers and source networks.
- cloudwatch – Retrieve disk usage to allow cost optimization
- iam – Acquire the permissions required for recovery
- kms – Allow using encrypted volumes
- drs – Retrieve tags and set tags for DRS resources, create DRS resources on failover

Permissions details

To view the policy permission details see AWSElasticDisasterRecoveryServiceRolePolicy in the AWS Managed Policy Reference Guide.

AWS managed policy: AWSElasticDisasterRecoveryStagingAccountPolicy

This policy allows read-only access to AWS Elastic Disaster Recovery (AWS DRS) resources such as source servers and jobs. It also allows creating a converted snapshot and sharing that EBS snapshot with a specified account.

Permissions details

To view the policy permission details see AWSElasticDisasterRecoveryStagingAccountPolicy in the AWS Managed Policy Reference Guide.

AWS managed policy: AWSElasticDisasterRecoveryStagingAccountPolicy_v2

This policy is used by AWS Elastic Disaster Recovery (AWS DRS) to recover source servers into a separate target account and to allow failing back. We do not recommend that you attach this policy to your users or roles.

Permissions details

To view the policy permission details see AWSElasticDisasterRecoveryStagingAccountPolicy_v2 in the AWS Managed Policy Reference Guide.

AWS managed policy: AWSElasticDisasterRecoveryEc2InstancePolicy

This policy allows installing and using the AWS Replication Agent, which is used by AWS Elastic Disaster Recovery (AWS DRS) to recover source servers that run on EC2 (cross-Region, cross-AZ or

cross-Account). An IAM role with this policy should be attached (as an EC2 Instance Profile) to the EC2 Instances.

Permissions details

To view the policy permission details, see AWSElasticDisasterRecoveryEc2InstancePolicy in the AWS Managed Policy Reference Guide.

AWS managed policy: AWSElasticDisasterRecoveryCrossAccountReplicationPolicy

This policy allows AWS Elastic Disaster Recovery (DRS) to support cross-account replication and cross-account failback.

Permissions details

To view the policy permission details see AWSElasticDisasterRecoveryCrossAccountReplicationPolicy in the AWS Managed Policy Reference Guide.

AWS managed policy: AWSElasticDisasterRecoveryNetworkReplicationPolicy

This policy allows AWS Elastic Disaster Recovery (DRS) to support network replication.

Permissions details

To view the policy permission details see AWSElasticDisasterRecoveryNetworkReplicationPolicy in the AWS Managed Policy Reference Guide.

AWS managed policy: AWSElasticDisasterRecoveryLaunchActionsPolicy

You can attach the AWSElasticDisasterRecoveryLaunchActionsPolicy policy to your IAM identities.

This policy allows you to use Amazon SSM and additional services required permissions to run post-launch actions in AWS Elastic Disaster Recovery (AWS DRS). Attach this policy to your IAM roles or users.

Permissions details

To view the policy permission details see AWSElasticDisasterRecoveryLaunchActionsPolicy in the AWS Managed Policy Reference Guide.

AWS managed policy: AWSElasticDisasterRecoveryConsoleFullAccess_v2

You can attach the **AWSElasticDisasterRecoveryConsoleFullAccess_v2** policy to your IAM identities.

Allows full administrative access to AWS Elastic Disaster Recovery (AWS DRS) Console. Attach this policy to your users or roles.

Permissions details

This policy includes permissions to do the following:

- `drs` – All APIs.
- `kms` – List aliases and describe keys.
- `ec2` – Describe account attributes, availability zones, images, instance (including types, statuses, type offerings), subnets, volumes, ebs encryption by default, ebs default kms key id, key/pairs, capacity reservations and hosts. Describe, create and delete snapshots. Describe and create launch templates. Start, run, stop and terminate instances. Describe and modify instance attributes. Create, attach and detach volumes. Describe, create, modify and delete launch template version. Create and delete tags. Get console output and screenshots. Describe and create security groups. Authorize and revoke security group egress. Authorize security group ingress.
- `license manager` – List license configurations.
- `resource groups` – List groups.
- `elastic load balancing` – Describe load balancers.
- `iam` – List instance profiles and roles, passRole.
- `cloudformation` – Describe and list stacks.
- `s3` – Get bucket location and list all my buckets.
- `ssm` – Describe instance information, send command, start automation execution. List documents and command invocations. Get and put parameters. Describe and get document. Get automation executions.

Permissions details

To view the policy permission details see AWSElasticDisasterRecoveryConsoleFullAccess_v2 in the AWS Managed Policy Reference Guide.

Elastic Disaster Recovery updates for AWS managed policies

View details about updates to AWS managed policies for AWS Elastic Disaster Recovery since March 1, 2021.

AWS Elastic Disaster Recovery policy updates

Change	Description	Date
AWSElasticDisasterRecoveryReadOnlyAccess – Updated policy.	We updated the AWSElasticDisasterRecoveryReadOnlyAccess policy with new read-only permissions for recovery plans and recovery plan executions.	August 18, 2026
AWSElasticDisasterRecoveryConsoleFullAccess_v2– Updated policy. AWSElasticDisasterRecoveryLaunchActionsPolicy– Updated policy.	AWS Elastic Disaster Recovery updated these policies to add permissions for Amazon EC2 Systems Manager documents with the `AWSDRS-` prefix.	June 18, 2026
AWSElasticDisasterRecoveryConsoleFullAccess_v2– Updated policy. AWSElasticDisasterRecoveryLaunchActionsPolicy_v2– Updated policy.	Updated policies to reflect changes in SSM.	July 3, 2025
AWSElasticDisasterRecoveryServiceRolePolicy– Updated policy AWSElasticDisasterRecoveryConsoleFullAccess_v2– Updated policy	Created new revisions of AWSElasticDisasterRecoveryServiceRolePolicy, AWSElasticDisasterRecoveryConsoleFullAccess_v2 and AWSElasticDisasterRecoveryConsoleFullAccess managed policies to	January 6, 2025

Change	Description	Date
AWSElasticDisasterRecoveryConsoleFullAccess– Updated policy	support a change in authentication with EBS APIs.	
• AWSElasticDisasterRecoveryConsoleFullAccess_v2– Updated policy • AWSElasticDisasterRecoveryLaunchActionsPolicy– Updated policy	Created new revisions of AWSElasticDisasterRecoveryConsoleFullAccess_v2 and AWSElasticDisasterRecoveryLaunchActionsPolicy managed policies, to support additional parameter types in SSM Parameter Store for post-launch actions.	May 19, 2024
AWSElasticDisasterRecoveryServiceRolePolicy – Updated policy	Created revision of the AWSElasticDisasterRecoveryServiceRolePolicy policy, to support replicating marketplace licenses to launched instances.	January 28, 2024
AWSElasticDisasterRecoveryCrossAccountReplicationPolicy – Updated policy	Created revision of the AWSElasticDisasterRecoveryCrossAccountReplicationPolicy policy, to support replicating marketplace licenses to launched instances.	January 28, 2024
AWSElasticDisasterRecoveryNetworkReplicationPolicy AWSElasticDisasterRecoveryServiceRolePolicy	Created new revisions of managed policies to support managed prefix lists for DRS network replication and recovery.	January 3rd, 2024

Change	Description	Date
AWSElasticDisasterRecoveryAgentPolicy AWSElasticDisasterRecoveryAgentInstallationPolicy AWSElasticDisasterRecoveryEc2InstancePolicy AWSElasticDisasterRecoveryConsoleFullAccess AWSElasticDisasterRecoveryLaunchActionsPolicy AWSElasticDisasterRecoveryNetworkReplicationPolicy AWSElasticDisasterRecoveryRecoveryInstancePolicy AWSElasticDisasterRecoveryServiceRolePolicy AWSElasticDisasterRecoveryConversionServerPolicy AWSElasticDisasterRecoveryFailbackPolicy AWSElasticDisasterRecoveryFailbackInstallationPolicy AWSElasticDisasterRecoveryStagingAccountPolicy_v2	Created new revisions of managed policies to support DRS to GovCloud and added Sid to statements in managed policies	November 27, 2023

Change	Description	Date
AWSElasticDisasterRecoveryStagingAccountPolicy AWSElasticDisasterRecoveryReplicationServerPolicy		
AWSElasticDisasterRecoveryCrossAccountReplicationPolicy – Updated policy	Created revision of AWSElasticDisasterRecoveryCrossAccountReplicationPolicy to support DRS in GovCloud	November 27, 2023
AWSElasticDisasterRecoveryReadOnlyAccess – Updated policy	AWS Elastic Disaster Recovery updated the policy with additional read-only permissions for post-launch actions.	November 27, 2023
AWSElasticDisasterRecoveryConsoleFullAccess_v2 New policy	AWS Elastic Disaster Recovery added a new policy. This policy provides access to use DRS console. Attach this policy to your IAM roles or users.	November 27, 2023
AWSElasticDisasterRecoveryConsoleFullAccess – Updated policy	AWS Elastic Disaster Recovery updated the policy to allow launching into an existing instance.	October 15, 2023
AWSElasticDisasterRecoveryConsoleFullAccess – Updated policy	AWS Elastic Disaster Recovery updated the policy to allow launching into an existing instance.	October 15, 2023

Change	Description	Date
AWSElasticDisasterRecoveryLaunchActionsPolicy – Updated policy	AWS Elastic Disaster Recovery updated the policy to allow launching into an existing instance tagged with a specific AWS-only key-value pair.	October 15, 2023
AWSElasticDisasterRecoveryEc2InstancePolicy – Updated policy	AWS Elastic Disaster Recovery updated the policy to allow sending installation result metrics to AWS Elastic Disaster Recovery.	October 10, 2023
AWSElasticDisasterRecoveryAgentInstallationPolicy – Updated policy	AWS Elastic Disaster Recovery updated the policy to allow sending installation result metrics to AWS Elastic Disaster Recovery.	October 10, 2023
AWSElasticDisasterRecoveryLaunchActionsPolicy New policy	AWS Elastic Disaster Recovery added a new policy. This policy provides access to use post-launch actions. Attach this policy to your IAM roles or users.	September 13, 2023
AWSElasticDisasterRecoveryReadOnlyAccess – Updated policy	AWS Elastic Disaster Recovery updated the policy with new read-only APIs for post-laun ch actions.	September 13, 2023
AWSElasticDisasterRecoveryAgentInstallationPolicy – Updated policy	AWS Elastic Disaster Recovery updated the policy to allow network replication and recovery.	June 13, 2023

Change	Description	Date
AWSElasticDisasterRecoveryEc2InstancePolicy – Updated policy	This policy was updated to allow network replication and recovery.	June 13, 2023
AWSElasticDisasterRecoveryConsoleFullAccess– Updated policy	This policy was updated to support network replication and recovery.	June 13, 2023
AWSElasticDisasterRecoveryNetworkReplicationPolicy – New policy	This policy is used by AWS Elastic Disaster Recovery (DRS) to support network replication.	June 13, 2023
AWSElasticDisasterRecoveryServiceRolePolicy – Updated policy	This policy was updated to support network replication and recovery.	June 13, 2023
AWSElasticDisasterRecoveryCrossAccountReplicationPolicy – New policy	This policy is used by AWS Elastic Disaster Recovery (DRS) to support replication and failback.	May 14, 2023
AWSElasticDisasterRecoveryRecoveryInstancePolicy – Updated policy	This policy was updated to support failback by the agent after reverse replication.	May 14, 2023
AWSElasticDisasterRecoveryEc2InstancePolicy – Updated policy	This policy was updated to support replication by the agent.	May 14, 2023
AWSElasticDisasterRecoveryConsoleFullAccess– Updated policy	This policy was updated to support default EC2 launch templates and bulk editing of source server EC2 launch templates.	April 19, 2023

Change	Description	Date
AWSElasticDisasterRecoveryCrossAccountReplicationPolicy – New policy	This policy is used by AWS Elastic Disaster Recovery (DRS) to support cross-account replication and cross-account failback.	May 7, 2023
AWSElasticDisasterRecoveryRecoveryInstancePolicy – Updated policy	This policy was updated to support cross-account failback by the agent after reverse replication.	May 7, 2023
AWSElasticDisasterRecoveryEc2InstancePolicy – Updated policy	This policy was updated to support cross-account replication by the agent.	May 7, 2023
AWSElasticDisasterRecoveryConsoleFullAccess– Updated policy	This policy was updated to support default EC2 launch templates and bulk editing of source server EC2 launch templates.	April 16, 2023
AWSElasticDisasterRecoveryAgentPolicy – Updated policy	This policy was updated to support the kernel upgrade feature.	April 1, 2023
AWSElasticDisasterRecoveryStagingAccountPolicy_v2 – New policy	This policy was updated to support the kernel upgrade feature.	December 11, 2022

Change	Description	Date
AWSElasticDisasterRecoveryAgentInstallationPolicy – Updated policy	AWS Elastic Disaster Recovery updated the policy to properly support agent installation on Recovery Instances. This policy allows installing the AWS Replicati on Agent, which is used with AWS Elastic Disaster Recovery (AWS DRS) to recover external servers to AWS. Attach this policy to your users or roles whose credentials you provide during the installation step of the AWS Replication Agent.	November 14, 2022
AWSElasticDisasterRecoveryRecoveryInstancePolicy – Updated policy	AWS Elastic Disaster Recovery updated this policy to include permissions which allow DRS Recovery Instances that originated from EC2 instances to replicate back to their origin locations in a failback scenario. As an additional security mechanism, Elastic Disaster Recovery will block requests that are not targeted at the source server the EC2 instance is associated with.	October 24, 2022

Change	Description	Date
AWSElasticDisasterRecoveryAgentInstallationPolicy – Updated policy	AWS Elastic Disaster Recovery updated the policy to include resource tagging. This policy allows installing the AWS Replication Agent, which is used with AWS Elastic Disaster Recovery (AWS DRS) to recover external servers to AWS. Attach this policy to your users or roles whose credentials you provide during the installation step of the AWS Replication Agent.	June 28, 2022
AWSElasticDisasterRecoveryFailbackInstallationPolicy – Updated policy	AWS Elastic Disaster Recovery updated this policy to include a new permission (drs:UpdateAgentReplicationInfoForDrs). This permission is needed to complete the failback process in some cases.	June 22, 2022
AWSElasticDisasterRecoveryServiceRolePolicy – Updated policy	AWS Elastic Disaster Recovery updated the policy to allow DRS to call cloudwatch:GetMetricData and also ec2:ModifyVolume on EBS volumes of the replication server in order to support the automatic volume type selection feature.	June 21st, 2022

Change	Description	Date
AWSElasticDisasterRecoveryReplicationServerPolicy – Updated policy	AWS Elastic Disaster Recovery updated the policy to allow replication servers to call drs:NotifyVolumeEventForDrs and drs:SendVolumeStatsForDrs.	June 21st, 2022
AWSElasticDisasterRecoveryConsoleFullAccess – Updated policy	AWS Elastic Disaster Recovery updated the policy to allow listing IAM roles.	May 26th, 2022
AWSElasticDisasterRecoveryReadOnlyAccess – Updated policy	AWS Elastic Disaster Recovery updated the policy with new read-only APIs of DRS and also added a permission that allows to list IAM roles.	May 26th, 2022
AWSElasticDisasterRecoveryEc2InstancePolicy – Updated policy	AWS Elastic Disaster Recovery added a new policy. This policy allows installing and using the AWS Replication Agent, which is used by AWS Elastic Disaster Recovery (DRS) to recover source servers that run on EC2 (cross-region or cross-AZ). An IAM role with this policy should be attached (as an EC2 Instance Profile) to the EC2 Instances.	April 6, 2022
AWSElasticDisasterRecoveryReadOnlyAccess – Updated policy	AWS Elastic Disaster Recovery updated this policy.	April 3, 2022

Change	Description	Date
AWSElasticDisasterRecoveryStagingAccountPolicy – New policy	AWS Elastic Disaster Recovery added a new policy. This policy allows read-only access to AWS Elastic Disaster Recovery (DRS) resources such as source servers and jobs. It also allows creating a converted snapshot and sharing that EBS snapshot with a specified account.	February 24, 2022
AWSElasticDisasterRecoveryAgentPolicy – New policy	AWS Elastic Disaster Recovery added a new policy. This policy allows using the AWS Replication Agent, which is used with AWS Elastic Disaster Recovery to recover source servers to AWS. We do not recommend that you attach this policy to your users or roles.	November 17, 2021

Change	Description	Date
AWSElasticDisasterRecoveryConversionServerPolicy New policy	AWS Elastic Disaster Recovery added a new policy. This policy is attached to the AWS Elastic Disaster Recovery Conversion server's instance role. This policy allows Elastic Disaster Recovery (DRS) Conversion Servers, which are EC2 instances launched by Elastic Disaster Recovery, to communicate with the DRS service. An IAM role with this policy is attached (as an EC2 Instance Profile) by DRS to the DRS Conversion Servers, which are automatically launched and terminated by DRS, when needed. We do not recommend that you attach this policy to your users or roles. AWS DRS conversio n servers are used by AWS Elastic Disaster Recovery when users choose to recover source servers using the Elastic Disaster Recovery console, CLI, or API.	November 17, 2021

Change	Description	Date
AWSElasticDisasterRecoveryFailbackPolicy - New policy	AWS Elastic Disaster Recovery added a new policy. This policy allows using the AWS Elastic Disaster Recovery Failback Client, which is used to failback Recovery Instances back to your original source infrastructure. We do not recommend that you attach this policy to your users or roles.	November 17, 2021
AWSElasticDisasterRecoveryFailbackInstallationPolicy – New policy	AWS Elastic Disaster Recovery added a new policy. You can attach the AWSElasticDisasterRecoveryFailbackInstallationPolicy policy to your IAM identities. This policy allows installing the AWS Elastic Disaster Recovery Failback Client, which is used to failback recovery instances back to your original source infrastructure. Attach this policy to your users or roles whose credentials you provide when running the AWS Elastic Disaster Recovery Failback Client.	November 17, 2021

Change	Description	Date
AWSElasticDisasterRecoveryConsoleFullAccess – New policy	AWS Elastic Disaster Recovery added a new policy. This policy provides full access to all public APIs of AWS Elastic Disaster Recovery (AWS DRS), as well as permissions to read KMS key, License Manager, Resource Groups, Elastic Load Balancing, IAM, and Amazon EC2 information. Attach this policy to your users or roles.	November 17, 2021

Change	Description	Date
AWSElasticDisasterRecoveryReplicationServerPolicy – New policy	AWS Elastic Disaster Recovery added a new policy. This policy is attached to the Elastic Disaster Recovery Replication server's instance role. This policy allows the Elastic Disaster Recovery (DRS) Replication Servers, which are EC2 instances launched by Elastic Disaster Recovery, to communicate with the DRS service, and to create EBS snapshots in your AWS account. An IAM role with this policy is attached (as an EC2 Instance Profile) by Elastic Disaster Recovery to the DRS Replication Servers which are automatically launched and terminated by DRS, as needed. DRS Replication Servers are used to facilitat e data replication from your external servers to AWS, as part of the recovery process managed by DRS. We do not recommend that you attach this policy to your users or roles.	November 17, 2021

Change	Description	Date
AWSElasticDisasterRecoveryRecoveryInstancePolicy – New policy	AWS Elastic Disaster Recovery added a new policy. This policy is attached to the instance role of Elastic Disaster Recovery's Recovery Instance. This policy allows the Elastic Disaster Recovery (DRS) Recovery Instance, which are EC2 instances launched by Elastic Disaster Recovery - to communicate with the DRS service, and to be able to failback to their original source infrastru cture. An IAM role with this policy is attached (as an EC2 Instance Profile) by Elastic Disaster Recovery to the DRS recovery instances. We do not recommend that you attach this policy to your users or roles.	November 17, 2021
AWSElasticDisasterRecoveryServiceRolePolicy – New policy	AWS Elastic Disaster Recovery added a new policy. This policy allows Elastic Disaster Recovery to manage AWS resources on your behalf.	November 17, 2021
AWS Elastic Disaster Recovery started tracking changes	AWS Elastic Disaster Recovery started tracking changes for AWS managed policies.	November 17, 2021

Managing access using policies

You control access in AWS by creating policies and attaching them to AWS identities or resources. A policy defines permissions when associated with an identity or resource. AWS evaluates these policies when a principal makes a request. Most policies are stored in AWS as JSON documents. For more information about JSON policy documents, see Overview of JSON policies in the *IAM User Guide*.

Using policies, administrators specify who has access to what by defining which **principal** can perform **actions** on what **resources**, and under what **conditions**.

By default, users and roles have no permissions. An IAM administrator creates IAM policies and adds them to roles, which users can then assume. IAM policies define permissions regardless of the method used to perform the operation.

Identity-based policies

Identity-based policies are JSON permissions policy documents that you can attach to an identity, such as a user, role, or group. These policies control what actions that identity can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see Creating IAM Policies in the IAM User Guide.

Identity-based policies can be further categorized as inline policies or managed policies. Inline policies are embedded directly into a single user, group, or role. Managed policies are standalone policies that you can attach to multiple users, groups, and roles in your AWS account. Managed policies include AWS managed policies and customer managed policies. To learn how to choose between a managed policy or an inline policy, see Choosing Between Managed Policies and Inline Policies in the IAM User Guide.

Using identity-based policies

By default, IAM users and roles don't have permission to create or modify AWS Elastic Disaster Recovery resources. They also can't perform tasks using the AWS Management Console, AWS CLI, or AWS API. An IAM administrator must create IAM policies that grant users and roles permission to perform specific API operations on the specified resources they need. The administrator must then attach those policies to the users or groups that require those permissions. To learn how to attach policies to a user or group, see Adding and removing IAM identity permissions in the IAM User Guide. To learn how to create an IAM identity-based policy using example JSON policy documents, see Creating policies on the JSON tab in the IAM User Guide.

Topics

- Customer-managed policies in AWS Elastic Disaster Recovery
- Console Full Access Policy - AWSElasticDisasterRecoveryConsoleFullAccess
- Console Full Access Policy - AWSElasticDisasterRecoveryConsoleFullAccess_v2
- Launch Actions Policy - AWSElasticDisasterRecoveryLaunchActionsPolicy
- Console Read-Only Access Policy - AWSElasticDisasterRecoveryReadOnlyAccess

Customer-managed policies in AWS Elastic Disaster Recovery

You can create your own custom IAM policies to allow permissions for AWS Elastic Disaster Recovery actions and resources. You can attach these custom policies to the users, roles, or groups that require those permissions. You can also create your own custom IAM policies for integration between AWS Elastic Disaster Recovery and other AWS services. The following example IAM policies grant permissions for various AWS Elastic Disaster Recovery actions. Use them to limit AWS Elastic Disaster Recovery access for your users and roles.

Console Full Access Policy - AWSElasticDisasterRecoveryConsoleFullAccess

This policy provides full access to all public APIs of AWS Elastic Disaster Recovery (AWS DRS), as well as permissions to read KMS key, License Manager, Resource Groups, Elastic Load Balancing, IAM, and Amazon EC2 information. Attach this policy to your users or roles.

Permissions details

This policy includes the following permissions.

JSON

```
{
 "Version":"2012-10-17",
 "Statement": [
  {
   "Sid": "ConsoleFullAccess1",
   "Effect": "Allow",
   "Action": [
    "drs:*"
   ],
   "Resource": "*"
```

```
},
{
 "Sid": "ConsoleFullAccess2",
 "Effect": "Allow",
 "Action": [
  "kms:ListAliases",
  "kms:DescribeKey"
 ],
 "Resource": "*"
},
{
 "Sid": "ConsoleFullAccess3",
 "Effect": "Allow",
 "Action": [
  "ec2:DescribeAccountAttributes",
  "ec2:DescribeAvailabilityZones",
  "ec2:DescribeImages",
  "ec2:DescribeInstances",
  "ec2:DescribeInstanceTypes",
  "ec2:DescribeInstanceAttribute",
  "ec2:DescribeInstanceStatus",
  "ec2:DescribeInstanceTypeOfferings",
  "ec2:DescribeLaunchTemplateVersions",
  "ec2:DescribeLaunchTemplates",
  "ec2:DescribeSecurityGroups",
  "ec2:DescribeSnapshots",
  "ec2:DescribeSubnets",
  "ec2:DescribeVolumes",
  "ec2:GetEbsEncryptionByDefault",
  "ec2:GetEbsDefaultKmsKeyId",
  "ec2:DescribeKeyPairs",
  "ec2:DescribeCapacityReservations",
  "ec2:DescribeHosts"
 ],
 "Resource": "*"
},
{
 "Sid": "ConsoleFullAccess4",
 "Effect": "Allow",
 "Action": "license-manager:ListLicenseConfigurations",
 "Resource": "*"
},
{
 "Sid": "ConsoleFullAccess5",
```

```
   "Effect": "Allow",
   "Action": "resource-groups:ListGroups",
   "Resource": "*"
  },
  {
   "Sid": "ConsoleFullAccess6",
   "Effect": "Allow",
   "Action": "elasticloadbalancing:DescribeLoadBalancers",
   "Resource": "*"
  },
  {
   "Sid": "ConsoleFullAccess7",
   "Effect": "Allow",
   "Action": [
    "iam:ListInstanceProfiles",
    "iam:ListRoles"
   ],
   "Resource": "*"
  },
  {
   "Sid": "ConsoleFullAccess8",
   "Effect": "Allow",
   "Action": "iam:PassRole",
   "Resource": [
    "arn:aws:iam::*:role/service-role/
AWSElasticDisasterRecoveryConversionServerRole",
    "arn:aws:iam::*:role/service-role/
AWSElasticDisasterRecoveryRecoveryInstanceRole"
   ],
   "Condition": {
    "StringEquals": {
     "iam:PassedToService": "ec2.amazonaws.com"
    }
   }
  },
  {
   "Sid": "ConsoleFullAccess9",
   "Effect": "Allow",
   "Action": [
    "ec2:DeleteSnapshot"
   ],
   "Resource": "arn:aws:ec2:*:*:snapshot/*",
   "Condition": {
    "Null": {
```

```
      "aws:ResourceTag/AWSElasticDisasterRecoveryManaged": "false"
     },
     "Bool": {
      "aws:ViaAWSService": "true"
     }
    }
   },
   {
    "Sid": "ConsoleFullAccess10",
    "Effect": "Allow",
    "Action": [
     "ec2:CreateLaunchTemplateVersion",
     "ec2:ModifyLaunchTemplate",
     "ec2:DeleteLaunchTemplateVersions",
     "ec2:CreateTags",
     "ec2:DeleteTags"
    ],
    "Resource": "arn:aws:ec2:*:*:launch-template/*",
    "Condition": {
     "Null": {
      "aws:ResourceTag/AWSElasticDisasterRecoveryManaged": "false"
     }
    }
   },
   {
    "Sid": "ConsoleFullAccess11",
    "Effect": "Allow",
    "Action": [
     "ec2:CreateLaunchTemplate"
    ],
    "Resource": "arn:aws:ec2:*:*:launch-template/*",
    "Condition": {
     "Null": {
      "aws:RequestTag/AWSElasticDisasterRecoveryManaged": "false"
     }
    }
   },
   {
    "Sid": "ConsoleFullAccess12",
    "Effect": "Allow",
    "Action": [
     "ec2:DeleteVolume"
    ],
    "Resource": "arn:aws:ec2:*:*:volume/*",
```

```
   "Condition": {
    "Null": {
     "aws:ResourceTag/AWSElasticDisasterRecoveryManaged": "false"
    },
    "Bool": {
     "aws:ViaAWSService": "true"
    }
   }
  },
  {
   "Sid": "ConsoleFullAccess13",
   "Effect": "Allow",
   "Action": [
    "ec2:StartInstances",
    "ec2:StopInstances",
    "ec2:TerminateInstances",
    "ec2:ModifyInstanceAttribute",
    "ec2:GetConsoleOutput",
    "ec2:GetConsoleScreenshot"
   ],
   "Resource": "arn:aws:ec2:*:*:instance/*",
   "Condition": {
    "Null": {
     "aws:ResourceTag/AWSElasticDisasterRecoveryManaged": "false"
    },
    "Bool": {
     "aws:ViaAWSService": "true"
    }
   }
  },
  {
   "Sid": "ConsoleFullAccess14",
   "Effect": "Allow",
   "Action": [
    "ec2:RevokeSecurityGroupEgress",
    "ec2:AuthorizeSecurityGroupIngress",
    "ec2:AuthorizeSecurityGroupEgress"
   ],
   "Resource": "arn:aws:ec2:*:*:security-group/*",
   "Condition": {
    "Null": {
     "aws:ResourceTag/AWSElasticDisasterRecoveryManaged": "false"
    },
    "Bool": {
```

```
      "aws:ViaAWSService": "true"
     }
    }
   },
   {
    "Sid": "ConsoleFullAccess15",
    "Effect": "Allow",
    "Action": [
     "ec2:CreateVolume"
    ],
    "Resource": "arn:aws:ec2:*:*:volume/*",
    "Condition": {
     "Null": {
      "aws:RequestTag/AWSElasticDisasterRecoveryManaged": "false"
     },
     "Bool": {
      "aws:ViaAWSService": "true"
     }
    }
   },
   {
    "Sid": "ConsoleFullAccess16",
    "Effect": "Allow",
    "Action": "ec2:CreateSecurityGroup",
    "Resource": "arn:aws:ec2:*:*:vpc/*"
   },
   {
    "Sid": "ConsoleFullAccess17",
    "Effect": "Allow",
    "Action": [
     "ec2:CreateSecurityGroup"
    ],
    "Resource": "arn:aws:ec2:*:*:security-group/*",
    "Condition": {
     "Null": {
      "aws:RequestTag/AWSElasticDisasterRecoveryManaged": "false"
     },
     "Bool": {
      "aws:ViaAWSService": "true"
     }
    }
   },
   {
    "Sid": "ConsoleFullAccess18",
```

```
   "Effect": "Allow",
   "Action": [
    "ec2:CreateSnapshot"
   ],
   "Resource": "arn:aws:ec2:*:*:volume/*",
   "Condition": {
    "Null": {
     "ec2:ResourceTag/AWSElasticDisasterRecoveryManaged": "false"
    },
    "Bool": {
     "aws:ViaAWSService": "true"
    }
   }
  },
  {
   "Sid": "ConsoleFullAccess19",
   "Effect": "Allow",
   "Action": [
    "ec2:CreateSnapshot"
   ],
   "Resource": "arn:aws:ec2:*:*:snapshot/*",
   "Condition": {
    "Null": {
     "aws:RequestTag/AWSElasticDisasterRecoveryManaged": "false"
    },
    "Bool": {
     "aws:ViaAWSService": "true"
    }
   }
  },
  {
   "Sid": "ConsoleFullAccess20",
   "Effect": "Allow",
   "Action": [
    "ec2:DetachVolume",
    "ec2:AttachVolume"
   ],
   "Resource": "arn:aws:ec2:*:*:instance/*",
   "Condition": {
    "Null": {
     "ec2:ResourceTag/AWSElasticDisasterRecoveryManaged": "false"
    },
    "Bool": {
     "aws:ViaAWSService": "true"
```

```
    }
   }
  },
  {
   "Sid": "ConsoleFullAccess21",
   "Effect": "Allow",
   "Action": [
    "ec2:DetachVolume",
    "ec2:AttachVolume",
    "ec2:StartInstances",
    "ec2:GetConsoleOutput",
    "ec2:GetConsoleScreenshot"
   ],
   "Resource": "arn:aws:ec2:*:*:instance/*",
   "Condition": {
    "StringEquals": {
     "ec2:ResourceTag/AWSDRS": "AllowLaunchingIntoThisInstance"
    },
    "ForAnyValue:StringEquals": {
     "aws:CalledVia": [
      "drs.amazonaws.com"
     ]
    }
   }
  },
  {
   "Sid": "ConsoleFullAccess22",
   "Effect": "Allow",
   "Action": [
    "ec2:AttachVolume"
   ],
   "Resource": "arn:aws:ec2:*:*:volume/*",
   "Condition": {
    "Null": {
     "ec2:ResourceTag/AWSElasticDisasterRecoveryManaged": "false"
    },
    "Bool": {
     "aws:ViaAWSService": "true"
    }
   }
  },
  {
   "Sid": "ConsoleFullAccess23",
   "Effect": "Allow",
```

```
   "Action": [
    "ec2:DetachVolume"
   ],
   "Resource": "arn:aws:ec2:*:*:volume/*",
   "Condition": {
    "Bool": {
     "aws:ViaAWSService": "true"
    }
   }
  },
  {
   "Sid": "ConsoleFullAccess24",
   "Effect": "Allow",
   "Action": [
    "ec2:RunInstances"
   ],
   "Resource": "arn:aws:ec2:*:*:instance/*",
   "Condition": {
    "Null": {
     "aws:RequestTag/AWSElasticDisasterRecoveryManaged": "false"
    },
    "Bool": {
     "aws:ViaAWSService": "true"
    }
   }
  },
  {
   "Sid": "ConsoleFullAccess25",
   "Effect": "Allow",
   "Action": [
    "ec2:RunInstances"
   ],
   "Resource": [
    "arn:aws:ec2:*:*:security-group/*",
    "arn:aws:ec2:*:*:volume/*",
    "arn:aws:ec2:*:*:subnet/*",
    "arn:aws:ec2:*:*:image/*",
    "arn:aws:ec2:*:*:network-interface/*",
    "arn:aws:ec2:*:*:launch-template/*"
   ],
   "Condition": {
    "Bool": {
     "aws:ViaAWSService": "true"
    }
```

```
    }
   },
   {
    "Sid": "ConsoleFullAccess26",
    "Effect": "Allow",
    "Action": "ec2:CreateTags",
    "Resource": [
     "arn:aws:ec2:*:*:security-group/*",
     "arn:aws:ec2:*:*:volume/*",
     "arn:aws:ec2:*:*:snapshot/*",
     "arn:aws:ec2:*:*:instance/*"
    ],
    "Condition": {
     "StringEquals": {
      "ec2:CreateAction": [
       "CreateSecurityGroup",
       "CreateVolume",
       "CreateSnapshot",
       "RunInstances"
      ]
     },
     "Bool": {
      "aws:ViaAWSService": "true"
     }
    }
   },
   {
    "Sid": "ConsoleFullAccess27",
    "Effect": "Allow",
    "Action": "ec2:CreateTags",
    "Resource": "arn:aws:ec2:*:*:launch-template/*",
    "Condition": {
     "StringEquals": {
      "ec2:CreateAction": [
       "CreateLaunchTemplate"
      ]
     }
    }
   },
   {
    "Sid": "ConsoleFullAccess28",
    "Effect": "Allow",
    "Action": [
     "cloudformation:DescribeStacks",
```

```
    "cloudformation:ListStacks"
   ],
   "Resource": "*"
  },
  {
   "Sid": "ConsoleFullAccess29",
   "Effect": "Allow",
   "Action": [
    "s3:GetBucketLocation",
    "s3:ListAllMyBuckets"
   ],
   "Resource": "*"
  },
  {
   "Sid": "ConsoleFullAccess30",
   "Effect": "Allow",
   "Action": [
    "ec2:CreateVolume"
   ],
   "Resource": "arn:aws:ec2:*:*:snapshot/*",
   "Condition": {
    "Null": {
     "aws:ResourceTag/AWSElasticDisasterRecoveryManaged": "false"
    },
    "Bool": {
     "aws:ViaAWSService": "true"
    }
   }
  }
 ]
}
```

Console Full Access Policy - AWSElasticDisasterRecoveryConsoleFullAccess_v2

Allows full administrative access to AWS Elastic Disaster Recovery (AWS DRS) Console. Attach this policy to your users or roles.

Permissions details

This policy includes permissions to do the following:

- `drs` – All APIs.
- `kms` – List aliases and describe keys.
- `ec2` – Describe account attributes, availability zones, images, instance (including types, statuses, type offerings), subnets, volumes, ebs encryption by default, ebs default kms key id, key/pairs, capacity reservations and hosts. Describe, create and delete snapshots. Describe and create launch templates. Start, run, stop and terminate instances. Describe and modify instance attributes. Create, attach and detach volumes. Describe, create, modify and delete launch template version. Create and delete tags. Get console output and screenshots. Describe and create security groups. Authorize and revoke security group egress. Authorize security group ingress.
- `license manager` – List license configurations.
- `resource groups` – List groups.
- `elastic load balancing` – Describe load balancers.
- `iam` – List instance profiles and roles, passRole.
- `cloudformation` – Describe and list stacks.
- `s3` – Get bucket location and list all my buckets.
- `ssm` – Describe instance information, send command, start automation execution. List documents and command invocations. Get and put parameters. Describe and get document. Get automation executions.

JSON

```
{
 "Version":"2012-10-17",
 "Statement": [
  {
   "Sid": "ConsoleFullAccess1",
   "Effect": "Allow",
   "Action": [
    "drs:*"
   ],
   "Resource": "*"
  },
  {
   "Sid": "ConsoleFullAccess2",
   "Effect": "Allow",
```

```
 "Action": [
  "kms:ListAliases",
  "kms:DescribeKey"
 ],
 "Resource": "*"
},
{
 "Sid": "ConsoleFullAccess3",
 "Effect": "Allow",
 "Action": [
  "ec2:DescribeAccountAttributes",
  "ec2:DescribeAvailabilityZones",
  "ec2:DescribeImages",
  "ec2:DescribeInstances",
  "ec2:DescribeInstanceTypes",
  "ec2:DescribeInstanceAttribute",
  "ec2:DescribeInstanceStatus",
  "ec2:DescribeInstanceTypeOfferings",
  "ec2:DescribeLaunchTemplateVersions",
  "ec2:DescribeLaunchTemplates",
  "ec2:DescribeSecurityGroups",
  "ec2:DescribeSnapshots",
  "ec2:DescribeSubnets",
  "ec2:DescribeVolumes",
  "ec2:GetEbsEncryptionByDefault",
  "ec2:GetEbsDefaultKmsKeyId",
  "ec2:DescribeKeyPairs",
  "ec2:DescribeCapacityReservations",
  "ec2:DescribeHosts"
 ],
 "Resource": "*"
},
{
 "Sid": "ConsoleFullAccess4",
 "Effect": "Allow",
 "Action": "license-manager:ListLicenseConfigurations",
 "Resource": "*"
},
{
 "Sid": "ConsoleFullAccess5",
 "Effect": "Allow",
 "Action": "resource-groups:ListGroups",
 "Resource": "*"
},
```

```
  {
   "Sid": "ConsoleFullAccess6",
   "Effect": "Allow",
   "Action": "elasticloadbalancing:DescribeLoadBalancers",
   "Resource": "*"
  },
  {
   "Sid": "ConsoleFullAccess7",
   "Effect": "Allow",
   "Action": [
    "iam:ListInstanceProfiles",
    "iam:ListRoles"
   ],
   "Resource": "*"
  },
  {
   "Sid": "ConsoleFullAccess8",
   "Effect": "Allow",
   "Action": "iam:PassRole",
   "Resource": [
    "arn:aws:iam::*:role/service-role/
AWSElasticDisasterRecoveryConversionServerRole",
    "arn:aws:iam::*:role/service-role/
AWSElasticDisasterRecoveryRecoveryInstanceRole",
                "arn:aws:iam::*:role/service-role/
AWSElasticDisasterRecoveryRecoveryInstanceWithLaunchActionsRole"
   ],
   "Condition": {
    "StringEquals": {
     "iam:PassedToService": "ec2.amazonaws.com"
    }
   }
  },
  {
   "Sid": "ConsoleFullAccess9",
   "Effect": "Allow",
   "Action": [
    "ec2:DeleteSnapshot"
   ],
   "Resource": "arn:aws:ec2:*:*:snapshot/*",
   "Condition": {
    "Null": {
     "aws:ResourceTag/AWSElasticDisasterRecoveryManaged": "false"
    },
```

```
    "Bool": {
     "aws:ViaAWSService": "true"
    }
   }
  },
  {
   "Sid": "ConsoleFullAccess10",
   "Effect": "Allow",
   "Action": [
    "ec2:CreateLaunchTemplateVersion",
    "ec2:ModifyLaunchTemplate",
    "ec2:DeleteLaunchTemplateVersions",
    "ec2:CreateTags",
    "ec2:DeleteTags"
   ],
   "Resource": "arn:aws:ec2:*:*:launch-template/*",
   "Condition": {
    "Null": {
     "aws:ResourceTag/AWSElasticDisasterRecoveryManaged": "false"
    }
   }
  },
  {
   "Sid": "ConsoleFullAccess11",
   "Effect": "Allow",
   "Action": [
    "ec2:CreateLaunchTemplate"
   ],
   "Resource": "arn:aws:ec2:*:*:launch-template/*",
   "Condition": {
    "Null": {
     "aws:RequestTag/AWSElasticDisasterRecoveryManaged": "false"
    }
   }
  },
  {
   "Sid": "ConsoleFullAccess12",
   "Effect": "Allow",
   "Action":
   [
    "ec2:DeleteVolume"
   ],
   "Resource": "arn:aws:ec2:*:*:volume/*",
   "Condition": {
```

```
    "Null": {
     "aws:ResourceTag/AWSElasticDisasterRecoveryManaged": "false"
    },
    "Bool": {
     "aws:ViaAWSService": "true"
    }
   }
  },
  {
   "Sid": "ConsoleFullAccess13",
   "Effect": "Allow",
   "Action": [
    "ec2:StartInstances",
    "ec2:StopInstances",
    "ec2:TerminateInstances",
    "ec2:ModifyInstanceAttribute",
    "ec2:GetConsoleOutput",
    "ec2:GetConsoleScreenshot"
   ],
   "Resource": "arn:aws:ec2:*:*:instance/*",
   "Condition": {
    "Null": {
     "aws:ResourceTag/AWSElasticDisasterRecoveryManaged": "false"
    },
    "Bool": {
     "aws:ViaAWSService": "true"
    }
   }
  },
  {
   "Sid": "ConsoleFullAccess14",
   "Effect": "Allow",
   "Action": [
    "ec2:RevokeSecurityGroupEgress",
    "ec2:AuthorizeSecurityGroupIngress",
    "ec2:AuthorizeSecurityGroupEgress"
   ],
   "Resource": "arn:aws:ec2:*:*:security-group/*",
   "Condition": {
    "Null": {
     "aws:ResourceTag/AWSElasticDisasterRecoveryManaged": "false"
    },
    "Bool": {
     "aws:ViaAWSService": "true"
```

```
    }
   }
  },
  {
   "Sid": "ConsoleFullAccess15",
   "Effect": "Allow",
   "Action": [
    "ec2:CreateVolume"
   ],
   "Resource": "arn:aws:ec2:*:*:volume/*",
   "Condition": {
    "Null": {
     "aws:RequestTag/AWSElasticDisasterRecoveryManaged": "false"
    },
    "Bool": {
     "aws:ViaAWSService": "true"
    }
   }
  },
  {
   "Sid": "ConsoleFullAccess16",
   "Effect": "Allow",
   "Action": "ec2:CreateSecurityGroup",
   "Resource": "arn:aws:ec2:*:*:vpc/*"
  },
  {
   "Sid": "ConsoleFullAccess17",
   "Effect": "Allow",
   "Action":
   [
    "ec2:CreateSecurityGroup"
   ],
   "Resource": "arn:aws:ec2:*:*:security-group/*",
   "Condition": {
    "Null": {
     "aws:RequestTag/AWSElasticDisasterRecoveryManaged": "false"
    },
    "Bool": {
     "aws:ViaAWSService": "true"
    }
   }
  },
  {
   "Sid": "ConsoleFullAccess18",
```

```
   "Effect": "Allow",
   "Action": [
    "ec2:CreateSnapshot"
   ],
   "Resource": "arn:aws:ec2:*:*:volume/*",
   "Condition": {
    "Null": {
     "ec2:ResourceTag/AWSElasticDisasterRecoveryManaged": "false"
    },
    "Bool": {
     "aws:ViaAWSService": "true"
    }
   }
  },
  {
   "Sid": "ConsoleFullAccess19",
   "Effect": "Allow",
   "Action": [
    "ec2:CreateSnapshot"
   ],
   "Resource": "arn:aws:ec2:*:*:snapshot/*",
   "Condition": {
    "Null": {
     "aws:RequestTag/AWSElasticDisasterRecoveryManaged": "false"
    },
    "Bool": {
     "aws:ViaAWSService": "true"
    }
   }
  },
  {
   "Sid": "ConsoleFullAccess20",
   "Effect": "Allow",
   "Action": [
    "ec2:DetachVolume",
    "ec2:AttachVolume"
   ],
   "Resource": "arn:aws:ec2:*:*:instance/*",
   "Condition": {
    "Null": {
     "ec2:ResourceTag/AWSElasticDisasterRecoveryManaged": "false"
    },
    "Bool": {
     "aws:ViaAWSService": "true"
```

```
    }
   }
  },
  {
   "Sid": "ConsoleFullAccess21",
   "Effect": "Allow",
   "Action": [
    "ec2:DetachVolume",
    "ec2:AttachVolume",
    "ec2:StartInstances",
    "ec2:GetConsoleOutput",
    "ec2:GetConsoleScreenshot"
   ],
   "Resource": "arn:aws:ec2:*:*:instance/*",
   "Condition": {
    "StringEquals": {
     "ec2:ResourceTag/AWSDRS": "AllowLaunchingIntoThisInstance"
    },
    "ForAnyValue:StringEquals": {
     "aws:CalledVia": [
      "drs.amazonaws.com"
     ]
    }
   }
  },
  {
   "Sid": "ConsoleFullAccess22",
   "Effect": "Allow",
   "Action":
   [
    "ec2:AttachVolume"
   ],
   "Resource": "arn:aws:ec2:*:*:volume/*",
   "Condition": {
    "Null": {
     "ec2:ResourceTag/AWSElasticDisasterRecoveryManaged": "false"
    },
    "Bool":
    {
     "aws:ViaAWSService": "true"
    }
   }
  },
  {
```

```
 "Sid": "ConsoleFullAccess23",
 "Effect": "Allow",
 "Action": [
  "ec2:DetachVolume"
 ],
 "Resource": "arn:aws:ec2:*:*:volume/*",
 "Condition":
 {
  "Bool": {
   "aws:ViaAWSService": "true"
  }
 }
},
{
 "Sid": "ConsoleFullAccess24",
 "Effect": "Allow",
 "Action": [
  "ec2:RunInstances"
 ],
 "Resource": "arn:aws:ec2:*:*:instance/*",
 "Condition": {
  "Null": {
   "aws:RequestTag/AWSElasticDisasterRecoveryManaged": "false"
  },
  "Bool": {
   "aws:ViaAWSService": "true"
  }
 }
},
{
 "Sid": "ConsoleFullAccess25",
 "Effect": "Allow",
 "Action": [
  "ec2:RunInstances"
 ],
 "Resource": [
  "arn:aws:ec2:*:*:security-group/*",
  "arn:aws:ec2:*:*:volume/*",
  "arn:aws:ec2:*:*:subnet/*",
  "arn:aws:ec2:*:*:image/*",
  "arn:aws:ec2:*:*:network-interface/*",
  "arn:aws:ec2:*:*:launch-template/*"
 ],
 "Condition":
```

```
   {
    "Bool": {
     "aws:ViaAWSService": "true"
    }
   }
  },
  {
   "Sid": "ConsoleFullAccess26",
   "Effect": "Allow",
   "Action": "ec2:CreateTags",
   "Resource": [
    "arn:aws:ec2:*:*:security-group/*",
    "arn:aws:ec2:*:*:volume/*",
    "arn:aws:ec2:*:*:snapshot/*",
    "arn:aws:ec2:*:*:instance/*"
   ],
   "Condition": {
    "StringEquals": {
     "ec2:CreateAction": [
      "CreateSecurityGroup",
      "CreateVolume",
      "CreateSnapshot",
      "RunInstances"
     ]
    },
    "Bool": {
     "aws:ViaAWSService": "true"
    }
   }
  },
  {
   "Sid": "ConsoleFullAccess27",
   "Effect": "Allow",
   "Action": "ec2:CreateTags",
   "Resource": "arn:aws:ec2:*:*:launch-template/*",
   "Condition": {
    "StringEquals": {
     "ec2:CreateAction": [
      "CreateLaunchTemplate"
     ]
    }
   }
  },
  {
```

```
    "Sid": "ConsoleFullAccess28",
    "Effect": "Allow",
    "Action": [
     "cloudformation:DescribeStacks",
     "cloudformation:ListStacks"
    ],
    "Resource": "*"
   },
   {
    "Sid": "ConsoleFullAccess29",
    "Effect": "Allow",
    "Action": [
     "s3:GetBucketLocation",
     "s3:ListAllMyBuckets"
    ],
    "Resource": "*"
   },
   {
    "Sid": "ConsoleFullAccess30",
    "Effect": "Allow",
    "Action": [
     "ssm:DescribeInstanceInformation",
     "ssm:DescribeParameters"
    ],
    "Resource": [
     "*"
    ],
    "Condition": {
     "ForAnyValue:StringEquals": {
      "aws:CalledVia": [
       "drs.amazonaws.com"
      ]
     }
    }
   },
   {
    "Sid": "ConsoleFullAccess31",
    "Effect": "Allow",
    "Action": [
     "ssm:SendCommand",
     "ssm:StartAutomationExecution"
    ],
    "Resource": [
     "arn:aws:ssm:*:*:document/AWS-CreateImage",
```

```
    "arn:aws:ssm:*:*:document/AWSMigration-ValidateNetworkConnectivity",
    "arn:aws:ssm:*:*:document/AWSMigration-VerifyMountedVolumes",
    "arn:aws:ssm:*:*:document/AWSMigration-ValidateHttpResponse",
    "arn:aws:ssm:*:*:document/AWSMigration-ValidateDiskSpace",
    "arn:aws:ssm:*:*:document/AWSMigration-VerifyProcessIsRunning",
    "arn:aws:ssm:*:*:document/AWSMigration-LinuxTimeSyncSetting",
    "arn:aws:ssm:*:*:document/AWSEC2-
ApplicationInsightsCloudwatchAgentInstallAndConfigure",
    "arn:aws:ssm:*:*:automation-execution/*"
   ],
   "Condition": {
    "ForAnyValue:StringEquals": {
     "aws:CalledVia": [
      "drs.amazonaws.com"
     ]
    }
   }
  },
  {
   "Sid": "ConsoleFullAccess32",
   "Effect": "Allow",
   "Action": [
    "ssm:SendCommand"
   ],
   "Resource": [
    "arn:aws:ec2:*:*:instance/*"
   ],
   "Condition": {
    "ForAnyValue:StringEquals": {
     "aws:CalledVia": [
      "drs.amazonaws.com"
     ]
    },
    "Null": {
     "aws:ResourceTag/AWSElasticDisasterRecoveryManaged": "false"
    }
   }
  },
  {
   "Sid": "ConsoleFullAccess33",
   "Effect": "Allow",
   "Action": [
    "ssm:ListDocuments",
    "ssm:ListCommandInvocations"
```

```
    ],
    "Resource": "*"
   },
   {
    "Sid": "ConsoleFullAccess34",
    "Effect": "Allow",
    "Action": [
     "ssm:GetParameter",
     "ssm:PutParameter"
    ],
    "Resource": "arn:aws:ssm:*:*:parameter/ManagedByAWSElasticDisasterRecovery-*",
    "Condition": {
     "StringEquals": {
         "aws:ResourceAccount": "${aws:PrincipalAccount}"
     }
    }
   },
   {
    "Sid": "ConsoleFullAccess35",
    "Effect": "Allow",
    "Action": [
     "ssm:DescribeDocument",
     "ssm:GetDocument"
    ],
    "Resource": "arn:aws:ssm:*:*:document/*"
   },
   {
    "Sid": "ConsoleFullAccess36",
    "Effect": "Allow",
    "Action": [
     "ssm:GetParameters"
    ],
    "Resource": [
     "arn:aws:ssm:*:*:parameter/ManagedByAWSElasticDisasterRecovery-*"
    ],
    "Condition": {
     "ForAnyValue:StringEquals": {
      "aws:CalledVia": "ssm.amazonaws.com"
     }
    }
   },
   {
    "Sid": "ConsoleFullAccess37",
    "Effect": "Allow",
```

```
   "Action": [
    "ssm:GetAutomationExecution"
   ],
   "Resource": "arn:aws:ssm:*:*:automation-execution/*",
   "Condition": {
    "Null": {
     "aws:ResourceTag/AWSElasticDisasterRecoveryManaged": "false"
    }
   }
  }
 ]
}
```

Launch Actions Policy - AWSElasticDisasterRecoveryLaunchActionsPolicy

This policy allows you to use Amazon SSM and additional services required permissions to run post-launch actions in AWS Elastic Disaster Recovery (AWS DRS). Attach this policy to your IAM roles or users.

Permissions details

This policy includes the following permissions.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "LaunchActionsPolicy1",
            "Effect": "Allow",
            "Action": [
                "ssm:DescribeInstanceInformation"
            ],
            "Resource": [
                "*"
            ],
            "Condition": {
                "ForAnyValue:StringEquals": {
```

```
                "aws:CalledVia": [
                    "drs.amazonaws.com"
                ]
            }
        }
    },
    {
        "Sid": "LaunchActionsPolicy2",
        "Effect": "Allow",
        "Action": [
            "ssm:SendCommand",
            "ssm:StartAutomationExecution"
        ],
        "Resource": [
            "arn:aws:ssm:*:*:document/*",
            "arn:aws:ssm:*:*:automation-execution/*"
        ],
        "Condition": {
            "ForAnyValue:StringEquals": {
                "aws:CalledVia": [
                    "drs.amazonaws.com"
                ]
            },
            "StringEquals": {
                "aws:ResourceAccount": "${aws:PrincipalAccount}"
            }
        }
    },
    {
        "Sid": "LaunchActionsPolicy3",
        "Effect": "Allow",
        "Action": [
            "ssm:SendCommand",
            "ssm:StartAutomationExecution"
        ],
        "Resource": [
            "arn:aws:ssm:*::document/AWS-*",
            "arn:aws:ssm:*::document/AWSCodeDeployAgent-*",
            "arn:aws:ssm:*::document/AWSConfigRemediation-*",
            "arn:aws:ssm:*::document/AWSConformancePacks-*",
            "arn:aws:ssm:*::document/AWSDisasterRecovery-*",
            "arn:aws:ssm:*::document/AWSDistroOTel-*",
            "arn:aws:ssm:*::document/AWSDocs-*",
            "arn:aws:ssm:*::document/AWSEC2-*",
```

```
                "arn:aws:ssm:*::document/AWSEC2Launch-*",
                "arn:aws:ssm:*::document/AWSFIS-*",
                "arn:aws:ssm:*::document/AWSFleetManager-*",
                "arn:aws:ssm:*::document/AWSIncidents-*",
                "arn:aws:ssm:*::document/AWSKinesisTap-*",
                "arn:aws:ssm:*::document/AWSMigration-*",
                "arn:aws:ssm:*::document/AWSNVMe-*",
                "arn:aws:ssm:*::document/AWSNitroEnclavesWindows-*",
                "arn:aws:ssm:*::document/AWSObservabilityExporter-*",
                "arn:aws:ssm:*::document/AWSPVDriver-*",
                "arn:aws:ssm:*::document/AWSQuickSetupType-*",
                "arn:aws:ssm:*::document/AWSQuickStarts-*",
                "arn:aws:ssm:*::document/AWSRefactorSpaces-*",
                "arn:aws:ssm:*::document/AWSResilienceHub-*",
                "arn:aws:ssm:*::document/AWSSAP-*",
                "arn:aws:ssm:*::document/AWSSAPTools-*",
                "arn:aws:ssm:*::document/AWSSQLServer-*",
                "arn:aws:ssm:*::document/AWSSSO-*",
                "arn:aws:ssm:*::document/AWSSupport-*",
                "arn:aws:ssm:*::document/AWSSystemsManagerSAP-*",
                "arn:aws:ssm:*::document/AmazonCloudWatch-*",
                "arn:aws:ssm:*::document/AmazonCloudWatchAgent-*",
                "arn:aws:ssm:*::document/AmazonECS-*",
                "arn:aws:ssm:*::document/AmazonEFSUtils-*",
                "arn:aws:ssm:*::document/AmazonEKS-*",
                "arn:aws:ssm:*::document/AmazonInspector-*",
                "arn:aws:ssm:*::document/AmazonInspector2-*",
                "arn:aws:ssm:*::document/AmazonInternal-*",
                "arn:aws:ssm:*::document/AwsEnaNetworkDriver-*",
                "arn:aws:ssm:*::document/AwsVssComponents-*",
                "arn:aws:ssm:*:*:automation-execution/*"
            ],
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "aws:CalledVia": [
                        "drs.amazonaws.com"
                    ]
                }
            }
        },
        {
            "Sid": "LaunchActionsPolicy4",
            "Effect": "Allow",
            "Action": [
```

```
                "ssm:SendCommand"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:instance/*"
            ],
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "aws:CalledVia": [
                        "drs.amazonaws.com"
                    ]
                },
                "Null": {
                    "aws:ResourceTag/AWSElasticDisasterRecoveryManaged": "false"
                }
            }
        },
        {
            "Sid": "LaunchActionsPolicy5",
            "Effect": "Allow",
            "Action": [
                "ssm:SendCommand"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:instance/*"
            ],
            "Condition": {
                "StringEquals": {
                    "ec2:ResourceTag/AWSDRS": "AllowLaunchingIntoThisInstance"
                },
                "ForAnyValue:StringEquals": {
                    "aws:CalledVia": [
                        "drs.amazonaws.com"
                    ]
                }
            }
        },
        {
            "Sid": "LaunchActionsPolicy6",
            "Effect": "Allow",
            "Action": [
                "ssm:ListDocuments",
                "ssm:ListCommandInvocations"
            ],
            "Resource": "*"
```

```
        },
        {
            "Sid": "LaunchActionsPolicy7",
            "Effect": "Allow",
            "Action": [
                "ssm:ListDocumentVersions",
                "ssm:GetDocument",
                "ssm:DescribeDocument"
            ],
            "Resource": "arn:aws:ssm:*:*:document/*"
        },
        {
            "Sid": "LaunchActionsPolicy8",
            "Effect": "Allow",
            "Action": [
                "ssm:GetAutomationExecution"
            ],
            "Resource": "arn:aws:ssm:*:*:automation-execution/*",
            "Condition": {
                "Null": {
                    "aws:ResourceTag/AWSElasticDisasterRecoveryManaged": "false"
                }
            }
        },
        {
            "Sid": "LaunchActionsPolicy9",
            "Effect": "Allow",
            "Action": [
                "ssm:GetParameters"
            ],
            "Resource": "arn:aws:ssm:*:*:parameter/
ManagedByAWSElasticDisasterRecoveryService-*",
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "aws:CalledVia": "ssm.amazonaws.com"
                }
            }
        },
        {
            "Sid": "LaunchActionsPolicy10",
            "Effect": "Allow",
            "Action": [
                "ssm:GetParameter",
                "ssm:PutParameter"
```

```
            ],
            "Resource": "arn:aws:ssm:*:*:parameter/
ManagedByAWSElasticDisasterRecoveryService-*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceAccount": "${aws:PrincipalAccount}"
                }
            }
        },
        {
            "Sid": "LaunchActionsPolicy11",
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": [
                "arn:aws:iam::*:role/service-role/
AWSElasticDisasterRecoveryRecoveryInstanceWithLaunchActionsRole"
            ],
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": "ec2.amazonaws.com"
                },
                "ForAnyValue:StringEquals": {
                    "aws:CalledVia": "drs.amazonaws.com"
                }
            }
        }
    ]
}
```

Console Read-Only Access Policy - AWSElasticDisasterRecoveryReadOnlyAccess

You can attach the AWSElasticDisasterRecoveryReadOnlyAccess policy to your IAM identities.

This policy provides permissions to all read-only public APIs of AWS Elastic Disaster Recovery (AWS DRS), as well as some read-only APIs of IAM, EC2 and SSM in order to list and view installed roles, Recovery Instances, Source Servers and post-launch actions. Attach this policy to your users or roles.

Permissions details

This policy includes the following permissions.

JSON

```
{
 "Version":"2012-10-17",
 "Statement": [
  {
   "Sid": "DRSReadOnlyAccess1",
   "Effect": "Allow",
   "Action": [
    "drs:DescribeJobLogItems",
    "drs:DescribeJobs",
    "drs:DescribeRecoveryInstances",
    "drs:DescribeRecoverySnapshots",
    "drs:DescribeReplicationConfigurationTemplates",
    "drs:DescribeSourceServers",
    "drs:GetFailbackReplicationConfiguration",
    "drs:GetLaunchConfiguration",
    "drs:GetReplicationConfiguration",
    "drs:ListExtensibleSourceServers",
    "drs:ListStagingAccounts",
    "drs:ListTagsForResource",
    "drs:ListLaunchActions"
   ],
   "Resource": "*"
  },
  {
   "Sid": "DRSReadOnlyAccess2",
   "Effect": "Allow",
   "Action": [
    "ec2:DescribeInstances",
    "ec2:DescribeLaunchTemplateVersions",
    "ec2:DescribeSecurityGroups",
    "ec2:DescribeSubnets"
   ],
   "Resource": "*"
  },
  {
   "Sid": "DRSReadOnlyAccess4",
   "Effect": "Allow",
   "Action": "iam:ListRoles",
   "Resource": "*"
  },
  {
```

```
    "Sid": "DRSReadOnlyAccess5",
    "Effect": "Allow",
    "Action": "ssm:ListCommandInvocations",
    "Resource": "*"
   },
   {
    "Sid": "DRSReadOnlyAccess6",
    "Effect": "Allow",
    "Action": "ssm:GetParameter",
    "Resource": "arn:aws:ssm:*:*:parameter/ManagedByAWSElasticDisasterRecovery-*"
   },
   {
    "Sid": "DRSReadOnlyAccess7",
    "Effect": "Allow",
    "Action": [
     "ssm:DescribeDocument",
     "ssm:GetDocument"
    ],
    "Resource": [
     "arn:aws:ssm:*:*:document/AWS-CreateImage",
     "arn:aws:ssm:*:*:document/AWSMigration-ValidateNetworkConnectivity",
     "arn:aws:ssm:*:*:document/AWSMigration-VerifyMountedVolumes",
     "arn:aws:ssm:*:*:document/AWSMigration-ValidateHttpResponse",
     "arn:aws:ssm:*:*:document/AWSMigration-ValidateDiskSpace",
     "arn:aws:ssm:*:*:document/AWSMigration-VerifyProcessIsRunning",
     "arn:aws:ssm:*:*:document/AWSMigration-LinuxTimeSyncSetting",
     "arn:aws:ssm:*:*:document/AWSEC2-
ApplicationInsightsCloudwatchAgentInstallAndConfigure"
    ]
   },
   {
    "Sid": "DRSReadOnlyAccess8",
    "Effect": "Allow",
    "Action": [
     "ssm:GetAutomationExecution"
    ],
    "Resource": "arn:aws:ssm:*:*:automation-execution/*",
    "Condition": {
     "Null": {
      "aws:ResourceTag/AWSElasticDisasterRecoveryManaged": "false"
     }
    }
   }
  ]
```

```
}
```

Resource-based policies

Resource-based policies are JSON policy documents that you attach to a resource. Examples include IAM *role trust policies* and Amazon S3 *bucket policies*. In services that support resource-based policies, service administrators can use them to control access to a specific resource. You must specify a principal in a resource-based policy.

Resource-based policies are inline policies that are located in that service. You can't use AWS managed policies from IAM in a resource-based policy.

Access control lists (ACLs)

Access control lists (ACLs) control which principals (account members, users, or roles) have permissions to access a resource. ACLs are similar to resource-based policies, although they do not use the JSON policy document format.

Amazon S3, AWS WAF, and Amazon VPC are examples of services that support ACLs. To learn more about ACLs, see Access control list (ACL) overview in the *Amazon Simple Storage Service Developer Guide*.

Other policy types

AWS supports additional policy types that can set the maximum permissions granted by more common policy types:

- **Permissions boundaries** – Set the maximum permissions that an identity-based policy can grant to an IAM entity. For more information, see Permissions boundaries for IAM entities in the *IAM User Guide*.
- **Service control policies (SCPs)** – Specify the maximum permissions for an organization or organizational unit in AWS Organizations. For more information, see Service control policies in the *AWS Organizations User Guide*.
- **Resource control policies (RCPs)** – Set the maximum available permissions for resources in your accounts. For more information, see Resource control policies (RCPs) in the *AWS Organizations User Guide*.
- **Session policies** – Advanced policies passed as a parameter when creating a temporary session for a role or federated user. For more information, see Session policies in the *IAM User Guide*.

Multiple policy types

When multiple types of policies apply to a request, the resulting permissions are more complicated to understand. To learn how AWS determines whether to allow a request when multiple policy types are involved, see Policy evaluation logic in the *IAM User Guide*.

Using service-linked roles for AWS Elastic Disaster Recovery

AWS Elastic Disaster Recovery uses AWS Identity and Access Management (IAM)service-linked roles. A service-linked role is a unique type of IAM role that is linked directly to AWS Elastic Disaster Recovery. Service-linked roles are predefined by AWS Elastic Disaster Recovery and include all the permissions that the service requires to call other AWS services on your behalf.

A service-linked role makes setting up AWS Elastic Disaster Recovery easier because you don't have to manually add the necessary permissions. AWS Elastic Disaster Recovery defines the permissions of its service-linked roles, and unless defined otherwise, only AWS Elastic Disaster Recovery can assume its roles. The defined permissions include the trust policy and the permissions policy, and that permissions policy cannot be attached to any other IAM entity.

You can delete a service-linked role only after first deleting its related resources. This protects your AWS Elastic Disaster Recovery resources because you can't inadvertently remove permission to access the resources.

For information about other services that support service-linked roles, see AWS Services That Work with IAM and look for the services that have **Yes** in the **Service-Linked Role** column. Choose a **Yes** with a link to view the service-linked role documentation for that service.

Service-linked role permissions for AWS Elastic Disaster Recovery

AWS Elastic Disaster Recovery uses the service-linked role named **AWSServiceRoleForElasticDisasterRecovery**. This role includes a managed IAM policy AWSElasticDisasterRecoveryServiceRolePolicy with scoped permissions that AWS Elastic Disaster Recovery needs to run in your account.

The AWSServiceRoleForElasticDisasterRecovery service-linked role trusts the following services to assume the role: `drs.amazonaws.com`

The role permissions policy allows AWS Elastic Disaster Recovery to complete the following actions on the specified resources. For the full policy definition, see AWSElasticDisasterRecoveryServiceRolePolicy.

You must configure permissions to allow an IAM entity (such as a user, group, or role) to create, edit, or delete a service-linked role. For more information, see Service-Linked Role Permissions in the *IAM User Guide*.

Creating a service-linked role for AWS Elastic Disaster Recovery

You don't need to manually create a service-linked role. When you configure the Replication Configuration Template for AWS Elastic Disaster Recovery, a service-linked role is automatically created. AWS Elastic Disaster Recovery automatically creates the IAM service-linked role, which you can see in the IAM console. You don't need to manually create or configure this role.

If you delete this service-linked role, and then need to create it again, you can use the same process to recreate the role in your account. When you create the first new replication configuration template in AWS Elastic Disaster Recovery, it creates the service-linked role for you again.

In the AWS CLI or the AWS API, create a service-linked role with the AWS Elastic Disaster Recovery service name. For more information, see Creating a Service-Linked Role in the *IAM User Guide*. If you delete this service-linked role, you can use this same process to create the role again.

Editing a service-linked role for AWS Elastic Disaster Recovery

AWS Elastic Disaster Recovery does not allow you to edit the AWSServiceRoleForElasticDisasterRecovery service-linked role. After you create a service-linked role, you cannot change the name of the role because various entities might reference the role. However, you can edit the description of the role using IAM. For more information, see Editing a Service-Linked Role in the *IAM User Guide*.

Deleting a service-linked role for AWS Elastic Disaster Recovery

If you no longer need to use a feature or service that requires a service-linked role, we recommend that you delete that role. That way you don’t have an unused entity that is not actively monitored or maintained. However, you must clean up the resources for your service-linked role before you can manually delete it.

> **Note**
>
> If AWS Elastic Disaster Recovery is using the role when you try to delete the resources, the deletion might fail. If that happens, wait for a few minutes and try the operation again.

To clean up AWS Elastic Disaster Recovery resources used by AWSServiceRoleForElasticDisasterRecovery

Resources can be cleaned up without stopping any AWS Elastic Disaster Recovery services. Cleaning up AWS Elastic Disaster Recovery resources will cause AWS Elastic Disaster Recovery to stop working. To clean up resources, you should disconnect all source servers, terminate all Recovery Instances, and delete all replication and launch configuration templates from AWS Elastic Disaster Recovery. For more information, see Cleaning up a service-linked role in the *IAM User Guide*.

To manually delete the service-linked role using IAM

Use the IAM console, the AWS CLI, or the AWS API to delete the AWSServiceRoleForElasticDisasterRecovery service-linked role. For more information, see Deleting a service-linked role in the *IAM User Guide*.

Supported AWS Regions for AWS Elastic Disaster Recovery service-linked roles

AWS Elastic Disaster Recovery supports using service-linked roles in all of the AWS Regions where the service is available.

Policy structure

An IAM policy is a JSON document that consists of one or more statements. Each statement is structured as follows:

```
{
 "Statement": [
  {
   "Effect": "effect",
   "Action": "action",
   "Resource": "arn",
   "Condition": {
    "condition": {
     "key": "value"
    }
   }
  }
 ]
}
```

There are various elements that make up a statement:

- **Effect:** The effect can be `Allow` or `Deny`. By default, IAM users don't have permission to use resources and API actions, so all requests are denied. An explicit allow overrides the default. An explicit deny overrides any allows.
- **Action**: The action is the specific AWS Elastic Disaster Recovery API action for which you are granting or denying permission.
- **Resource**: The resource that's affected by the action. For AWS Elastic Disaster Recovery, you must specify "*" as the resource.
- **Condition**: Conditions are optional. They can be used to control when your policy is in effect.

Resilience in AWS Elastic Disaster Recovery

The AWS global infrastructure is built around AWS Regions and Availability Zones. Regions provide multiple physically separated and isolated Availability Zones, which are connected through low-latency, high-throughput, and highly redundant networking. With Availability Zones, you can design and operate applications and databases that automatically fail over between zones without interruption. Availability Zones are more highly available, fault tolerant, and scalable than traditional single or multiple data center infrastructures.

For more information about AWS Regions and Availability Zones, see AWS Global Infrastructure.

Infrastructure security in AWS Elastic Disaster Recovery

As a managed service, AWS Elastic Disaster Recovery is protected by the AWS global network security procedures that are described in the Amazon Web Services: Overview of Security Processes whitepaper.

You use AWS published API calls to access AWS Elastic Disaster Recovery through the network. Clients must support Transport Layer Security (TLS) 1.2 or later. Clients must also support cipher suites with perfect forward secrecy (PFS) such as Ephemeral Diffie-Hellman (DHE) or Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). Most modern systems such as Java 7 and later support these modes.

All parties involved in the communication authenticate each other using TLS, IAM policies and tokens. The communication between the Agents and the replication server is based on TLS 1.2 only with the highest standard of cipher suite (PFS, ECDHE). Requests between the agent and AWS Elastic Disaster Recovery as well as between the replication server and AWS Elastic Disaster

Recovery are signed using an access key ID and a secret access key that is associated with an IAM principal.

Additionally, requests must be signed using an access key ID and a secret access key that is associated with an IAM principal. Or you can use the AWS Security Token Service (AWS STS) to generate temporary security credentials to sign requests.

AWS Elastic Disaster Recovery customers must ensure that they manually delete their access keys after installing the AWS Replication Agent and successful recovery. AWS does not delete these keys automatically. AWS Elastic Disaster Recovery does delete the keys from source servers after they are disconnected from the service. If you want your keys to automatically stop working at a certain date after you have finished using them so that you do not have to worry about manually deleting them, you can do so through the IAM permissions boundary and the aws:CurrentTime global context key.

AWS Elastic Disaster Recovery customers should use Amazon EBS encryption.

AWS Elastic Disaster Recovery customers should secure their replication servers by reducing their exposure to the public internet. This can be done through:

1. Using Security Groups to only allow permitted IP addresses to connect to the replication servers. Learn more about Security Groups.
2. Using a VPN to connect to the replication servers, such as the AWS site-to-site VPN. Learn more about the AWS Site-to-site VPN.

AWS Elastic Disaster Recovery creates and uses the "aws-replication" user within the Source server. The AWS Elastic Disaster Recovery replication server and AWS Replication Agent run under this user. Although this is not a root user, this user needs to be part of the disk group that grants this user full read and write permissions to block devices.

> **Note**
>
> AWS Elastic Disaster Recovery only uses these permissions to read from block devices.

AWS Elastic Disaster Recovery customers should only grant access to the AWS Elastic Disaster Recovery Failback Client to trusted administrators in order to prevent unauthorized entities from gaining access to your systems through the client.

AWS GovCloud

AWS GovCloud (US) are isolated AWS Regions designed to allow U.S. government agencies and customers to move sensitive workloads into the cloud.

- AWS GovCloud (US) uses FIPS 140-2 approved cryptographic modules for all AWS service API endpoints, unless otherwise indicated in the Service Endpoints section.
- AWS GovCloud (US) is appropriate for all types of Controlled Unclassified Information (CUI) and unclassified data. For more details, see Maintaining U.S. International Traffic in Arms Regulations (ITAR) Compliance .
- The AWS GovCloud (US) Regions are physically isolated and have logical network isolation from all other AWS Regions.
- AWS restricts all physical and logical access for those staff supporting AWS GovCloud (US) to US Citizens. AWS allows only vetted U.S. citizens with distinct access controls separate from other AWS Regions to administer AWS GovCloud (US). Any customer data fields that are defined as outside of the ITAR boundary (such as S3 bucket names) are explicitly documented in the service-specific section as not permitted to contain export-controlled data.
- AWS GovCloud (US) authentication is completely isolated from commercial regions.

Compliance validation for AWS Elastic Disaster Recovery

Third-party auditors assess the security and compliance of AWS Elastic Disaster Recovery as part of multiple AWS compliance programs.

For a list of AWS services in scope of specific compliance programs, see AWS Services in Scope by Compliance Program . For general information, see AWS Compliance Programs .

You can download third-party audit reports using AWS Artifact. For more information, see Downloading Reports in AWS Artifact .

Your compliance responsibility when using AWS Elastic Disaster Recovery is determined by the sensitivity of your data, your company's compliance objectives, and applicable laws and regulations. AWS provides the following resources to help with compliance:

- Security and Compliance Quick Start Guides – These deployment guides discuss architectural considerations and provide steps for deploying security- and compliance-focused baseline environments on AWS.

- Architecting for HIPAA Security and Compliance Whitepaper – This whitepaper describes how companies can use AWS to create HIPAA-compliant applications.
- AWS Compliance Resources – This collection of workbooks and guides might apply to your industry and location.
- Evaluating Resources with Rules in the *AWS Config Developer Guide* – AWS Config; assesses how well your resource configurations comply with internal practices, industry guidelines, and regulations.
- AWS Security Hub CSPM – This AWS service provides a comprehensive view of your security state within AWS that helps you check your compliance with security industry standards and best practices.

Cross-service confused deputy prevention

The confused deputy problem is a security issue where an entity that doesn't have permission to perform an action can coerce a more-privileged entity to perform the action. In AWS, cross-service impersonation can result in the confused deputy problem. Cross-service impersonation can occur when one service (the *calling service*) calls another service (the *called service*). The calling service can be manipulated to use its permissions to act on another customer's resources in a way it should not otherwise have permission to access. To prevent this, AWS provides tools that help you protect your data for all services with service principals that have been given access to resources in your account.

We recommend using the `aws:SourceArn` and `aws:SourceAccount` global condition context keys in resource policies to limit the permissions that AWS Elastic Disaster Recovery gives another service to the resource. If you use both global condition context keys, the `aws:SourceAccount` value and the account in the `aws:SourceArn` value must use the same account ID when used in the same policy statement.

The value of `aws:SourceArn` must be "arn:aws:drs:*:123456789012:source-server/*"

The most effective way to protect against the confused deputy problem is to use the `aws:SourceArn` global condition context key with the full ARN of the resource. If you don't know the full ARN of the resource or if you are specifying multiple resources, use the `aws:SourceArn` global context condition key with wildcards (*) for the unknown portions of the ARN. For example, `arn:aws:servicename::123456789012:*` .

The following example shows how you can use the `aws:SourceArn` and `aws:SourceAccount` global condition context keys in AWS Elastic Disaster Recovery to prevent the confused deputy problem.

IAM Roles that are created by AWS Elastic Disaster Recovery in your account already contain the confused deputy mitigation.

JSON

```
{
 "Version":"2012-10-17",
 "Statement": {
 "Sid": "ConfusedDeputyPreventionExamplePolicy",
 "Effect": "Allow",
 "Principal": {
 "Service": "drs.amazonaws.com"
 },
 "Action": "sts:AssumeRole",
 "Condition": {
 "StringLike": {
 "aws:SourceAccount": "123456789012"
 },
 "ArnLike": {
 "aws:SourceArn": "arn:aws:drs:*:123456789012:source-server/*"
 }
 }
 }
 }
```

Monitoring with Elastic Disaster Recovery

Logging AWS Elastic Disaster Recovery API calls using AWS CloudTrail

AWS Elastic Disaster Recovery is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service in AWS Elastic Disaster Recovery. CloudTrail captures all API calls for AWS Elastic Disaster Recovery as events. The calls captured include calls from the AWS Elastic Disaster Recovery console and code calls to the AWS Elastic Disaster Recovery API operations. If you create a trail, you can activate continuous delivery of CloudTrail events to an Amazon S3 bucket, including events for AWS Elastic Disaster Recovery. If you don't configure a trail, you can still view the most recent events in the CloudTrail console in **Event history**. Using the information collected by CloudTrail, you can determine the request that was made to AWS Elastic Disaster Recovery, the IP address from which the request was made, who made the request, when it was made, and additional details.

To learn more about CloudTrail, see the AWS CloudTrail User Guide.

AWS Elastic Disaster Recovery information in CloudTrail

CloudTrail is activated on your AWS account when you create the account. When activity occurs in AWS Elastic Disaster Recovery, that activity is recorded in a CloudTrail event along with other AWS service events in **Event history**. You can view, search, and download recent events in your AWS account. For more information, see Viewing events with CloudTrail Event history.

For an ongoing record of events in your AWS account, including events for AWS Elastic Disaster Recovery, create a trail. A *trail* enables CloudTrail to deliver log files to an Amazon S3 bucket. By default, when you create a trail in the console, the trail applies to all AWS Regions. The trail logs events from all Regions in the AWS partition and delivers the log files to the Amazon S3 bucket that you specify. Additionally, you can configure other AWS services to further analyze and act upon the event data collected in CloudTrail logs. For more information, see the following:

- Overview for creating a trail
- CloudTrail supported services and integrations
- Configuring Amazon SNS notifications for CloudTrail

- Receiving CloudTrail log files from multiple regions and Receiving CloudTrail log files from multiple accounts

All AWS Elastic Disaster Recovery actions are logged by CloudTrail and are documented in the AWS Elastic Disaster Recovery API. For example, calls to the `DescribeSourceServers` action to generate entries in the CloudTrail log files.

Every event or log entry contains information about who generated the request. The identity information helps you determine the following:

- Whether the request was made with root or AWS Identity and Access Management (IAM) user credentials.
- Whether the request was made with temporary security credentials for a role or federated user.
- Whether the request was made by another AWS service.

For more information, see the CloudTrail userIdentity element.

Understanding AWS Elastic Disaster Recovery log file entries

A trail is a configuration that allows delivery of events as log files to an Amazon S3 bucket that you specify. CloudTrail log files contain one or more log entries. An event represents a single request from any source and includes information about the requested action, the date and time of the action, request parameters, and so on. CloudTrail log files aren't an ordered stack trace of the public API calls, so they don't appear in any specific order.

The following example shows a CloudTrail log entry that demonstrates the DescribeSourceServers.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AAAAAAAAAAAAAAAAAAA",
        "arn": "arn:aws:sts::1234567890:assumed-role/Admin/user-Isengard",
        "accountId": "1234567890",
        "accessKeyId": "BBBBBBBBBBBBBBBBBBB",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
```

```
                "principalId": "AAAAAAAAAAAAAAAAAAA",
                "arn": "arn:aws:iam::1234567890:role/Admin",
                "accountId": "1234567890",
                "userName": "Admin"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2021-10-20T14:19:17Z",
                "mfaAuthenticated": "false"
            }
        }
    },
    "eventTime": "2021-10-20T14:19:59Z",
    "eventSource": "drs.amazonaws.com",
    "eventName": "DescribeSourceServers",
    "awsRegion": "eu-west-1",
    "sourceIPAddress": "54.240.197.234",
    "userAgent": "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like
 Gecko) Chrome/94.0.4606.81
    Safari/537.36",
    "requestParameters": {
        "maxResults": 1000,
        "filters": {}
    },
    "responseElements": null,
    "requestID": "d7618669-db08-4b53-bf6e-8a2cd57a677d",
    "eventID": "436c17a7-3a54-4f4e-815d-4d980339744e",
    "readOnly": true,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "1234567890",
    "eventCategory": "Management"
}
```


Amazon CloudWatch Metrics for DRS

The following are CloudWatch metrics for DRS:

- **TotalSourceServerCount** - number of source servers
- **LagDuration** - the age of the latest consistent snapshot, in seconds
- **Backlog** - the amount of data yet to be synced, in bytes.

- **DurationSinceLastSuccessfulRecoveryLaunch** - the amount of time that has passed since the last Drill or Recovery instance launch in seconds.
- **ElapsedReplicationDuration** - the cumulative amount of time this server has been replicating for in seconds.

Alarm events and EventBridge

Sample events for Elastic Disaster Recovery

The following are sample events for Elastic Disaster Recovery:

Source server data replication status

These events are triggered when source servers' data replication state changes from Stalled (replication not functioning properly) and Not Stalled (replication is functioning as expected).

STALLED

```
{
 "version": "0",
 "id": "9da9af57-9253-4406-87cb-7cc400e43465",
 "detail-type": "DRS Source Server Data Replication Stalled Change",
 "source": "aws.drs",
 "account": "111122223333",
 "time": "2016-08-22T20:12:19Z",
 "region": "us-west-2",
 "resources": [
  "arn:aws:drs:us-west-2:111122223333:source-server/s-12345678901234567"
 ],
 "detail": {
  "state": "STALLED"
 }
}
```

NOT_STALLED

```
{
```

```
 "version": "0",
 "id": "9da9af57-9253-4406-87cb-7cc400e43465",
 "detail-type": "DRS Source Server Data Replication Stalled Change",
 "source": "aws.drs",
 "account": "111122223333",
 "time": "2016-08-22T20:12:19Z",
 "region": "us-west-2",
 "resources": [
  "arn:aws:drs:us-west-2:111122223333:source-server/s-12345678901234567"
 ],
 "detail": {
  "state": "NOT_STALLED"
 }
}
```

Source server launch result

These events are triggered when a drill or recovery instance is launched for a source server and indicate whether the launch succeeded or failed.

RECOVERY_LAUNCH_SUCCEEDED

```
{
    "version": "0",
    "id": "9da9af57-9253-4406-87cb-7cc400e43465",
    "detail-type": "DRS Source Server Launch Result",
    "source": "aws.drs",
    "account": "111122223333",
    "time": "2016-08-22T20:12:19Z",
    "region": "us-west-2",
    "resources": [
        "arn:aws:drs:us-west-2:111122223333:source-server/s-12345678901234567"
    ],
    "detail": {
        "state": "RECOVERY_LAUNCH_SUCCEEDED",
        "job-id": "drsjob-04ca7d0d3fb6afa3e",
        "is-drill": "FALSE"
    }
}
```

RECOVERY_LAUNCH_FAILED

```
{
    "version": "0",
    "id": "9da9af57-9253-4406-87cb-7cc400e43465",
    "detail-type": "DRS Source Server Launch Result",
    "source": "aws.drs",
    "account": "111122223333",
    "time": "2016-08-22T20:12:19Z",
    "region": "us-west-2",
    "resources": [
        "arn:aws:drs:us-west-2:111122223333:source-server/s-12345678901234567"
    ],
    "detail": {
        "state": "RECOVERY_LAUNCH_FAILED",
        "job-id": "drsjob-04ca7d0d3fb6afa3e",
        "is-drill": "FALSE"
    }
}
```

Recovery instance failback State Change

These events are triggered as part of the failback process and indicate if failback is in progress, completed or failed.

FAILBACK_IN_PROGRESS

```
{
    "version": "0",
    "id": "9da9af57-9253-4406-87cb-7cc400e43465",
    "detail-type": "DRS Recovery Instance Failback State Change",
    "source": "aws.drs",
    "account": "111122223333",
    "time": "2016-08-22T20:12:19Z",
    "region": "us-west-2",
    "resources": [
        "arn:aws:drs:us-west-2:111122223333:recovery-instance/ri-12345678901234567"
    ],
    "detail": {
    "state": "FAILBACK_IN_PROGRESS"
```

```
    }
}
```

FAILBACK_COMPLETED

```
{
    "version": "0",
    "id": "9da9af57-9253-4406-87cb-7cc400e43465",
    "detail-type": "DRS Recovery Instance Failback State Change",
    "source": "aws.drs",
    "account": "111122223333",
    "time": "2016-08-22T20:12:19Z",
    "region": "us-west-2",
    "resources": [
        "arn:aws:drs:us-west-2:111122223333:recovery-instance/ri-12345678901234567"
    ],
    "detail": {
        "state": "FAILBACK_COMPLETED"
    }
}
```

FAILBACK_ERROR

```
{
    "version": "0",
    "id": "9da9af57-9253-4406-87cb-7cc400e43465",
    "detail-type": "DRS Recovery Instance Failback State Change",
    "source": "aws.drs",
    "account": "111122223333",
    "time": "2016-08-22T20:12:19Z",
    "region": "us-west-2",
    "resources": [
        "arn:aws:drs:us-west-2:111122223333:recovery-instance/ri-12345678901234567"
    ],
    "detail": {
    "state": "FAILBACK_ERROR"
    }
}
```

PIT Snapshot Taken

This event is triggered whenever a point in time snapshot is taken and includes its identifiers.

PIT Snapshot Taken

```
{
    "account": "111122223333",
    "detail": {
        "DrsSnapshotID": "112233",
        "EbsSnapshotIDs": "445566,778899"
    },
    "detail-type": "DRS PIT Snapshot Taken",
    "id": "9da9af57-9253-4406-87cb-7cc400e43465",
    "region": "us-west-2",
    "resources": [
        "arn:aws:drs:us-west-2:111122223333:source-server/s-12345678901234567"
    ],
    "source": "aws.drs",
    "time": "2016-08-22T20:12:19Z",
    "version": "0"
}
```

Registering event rules

You create EventBridge rules that capture events coming from your Elastic Disaster Recovery resources.

> **Note**
>
> When you use the AWS Management Console to create an event rule, the console automatically adds the IAM permissions necessary to grant EventBridge Event permissions to call your desired target type. If you are creating an event rule using the AWS CLI, you must grant permissions explicitly. For more information, see Event Patterns in the *Amazon EventBridge User Guide*.

To create Amazon EventBridge rules

1. Open the Amazon EventBridge console at https://console.aws.amazon.com/events/.
2. Using the following values, create an EventBridge rule that captures events coming from Elastic Disaster Recovery resources:

   - For **Rule type**, choose **Rule with an event pattern**.
   - For **Event source**, choose **Other**.
   - For **Event pattern**, choose **Custom patterns (JSON editor)**, and paste one of the following event pattern examples into the text area:
     - To catch all Elastic Disaster Recovery events:

```
{
 "source": [
  "aws.drs"
 ]
}
```

     - To catch all Recovery instance failback state changes:

```
{
 "detail-type": [
  "DRS Recovery Instance Failback State Change"
 ],
 "source": [
  "aws.drs"
 ]
}
```

     - To catch all events relating to a given Source server:

```
{
 "source": [
  "aws.drs"
 ],
 "resources": [
  "arn:aws:drs:us-west-2:111122223333:source-server/s-12345678901234567"
 ]
}
```

- For **Target types**, choose **AWS service**, and for **Select a target** choose your desired target.

For details about creating rules, see Creating Amazon EventBridge rules that react to events in the **Amazon EventBridge User Guide**.

Troubleshooting Elastic Disaster Recovery

Use this section to troubleshoot issues with AWS Elastic Disaster Recovery. Find the category that matches your issue below.

If you are experiencing...	See
Agent installer failures, prerequisites not met, or agent not starting	Agent installation
Replication stalled, disconnected, lagging, or showing error codes in the console	Replication errors
Recovery drill or failover launch failures, conversion errors, or post-launch issues	Launch errors
Failback Client errors, reverse replication failures, or failback connectivity issues	Failback errors

If you cannot resolve an issue using the guidance in this section, create a case with AWS Support.

Topics

- Troubleshooting agent installation
- Troubleshooting replication errors
- Troubleshooting launch errors
- Troubleshooting failback errors

Troubleshooting agent installation

Use this section to troubleshoot issues with installing or running the AWS Replication Agent on your source servers.

If you need to...	See
Check agent status, view logs, or gather diagnostics for a support case	Agent logs and diagnostics
Verify your server meets requirements before installing	Installation prerequisites
Fix errors during installation on Linux (kernel headers, SELinux, permissions, disk issues)	Linux installation errors
Fix errors during installation on Windows (TLS, BitLocker, PATH, signature issues)	Windows installation errors
Fix credential, IAM, or account configuration errors (any platform)	Common installation errors

Topics

- Agent logs and diagnostics
- Agent installation prerequisites
- Linux agent installation errors
- Windows agent installation errors
- Common agent installation errors

Agent logs and diagnostics

When troubleshooting Elastic Disaster Recovery agent issues, start by checking the service status and reviewing recent log entries. This page provides commands for checking agent health and gathering diagnostic information for support cases.

Topics

- Check agent status and logs
- Common log patterns
- Gather diagnostic information for support

Check agent status and logs

Use the following commands to verify that the replication agent is running and to review recent log output.

Linux

Check the agent service status:

```
systemctl status aws-replication-agent
```

View the most recent log entries:

```
tail -100 /var/lib/aws-replication-agent/agent.log.0
```

Search for errors in the log file:

```
grep -i "ERROR\|FATAL" /var/lib/aws-replication-agent/agent.log.0
```

> **Note**
>
> The agent log file is located at `/var/lib/aws-replication-agent/agent.log.0`. The service name is `aws-replication-agent`.

Windows

Check the agent service status:

```
Get-Service AwsReplicationService
```

View the most recent log entries:

```
Get-Content "C:\Program Files (x86)\AWS Replication Agent\agent.log.0" -Tail 100
```

Search for errors in the log file:

```
Select-String -Pattern "ERROR|FATAL" -Path "C:\Program Files (x86)\AWS Replication
 Agent\agent.log.0"
```

> **Note**
>
> The agent log file is located at `C:\Program Files (x86)\AWS Replication Agent\agent.log.0`. The service name is `AwsReplicationService`.

Common log patterns

Look for the following patterns in agent logs to identify the root cause of issues.

- **FATAL or ERROR entries** — Indicate agent failures that require immediate attention.
- **Connection refused** — Indicates a network connectivity issue. Verify that the source server can reach the AWS replication endpoints.
- **Certificate errors** — Indicates TLS or certificate trust issues. Verify that the source server trusts the AWS root certificates.
- **Timeout** — Indicates a firewall blocking traffic or an unreachable endpoint. Check security group rules and network ACLs.

Gather diagnostic information for support

Use the following tools to collect system details before opening a support case.

Linux

Run the AWS Linux system details tool:

https://github.com/awslabs/mgn-drs-linux-system-details-tool

Check the agent version:

```
cat /var/lib/aws-replication-agent/agent.version
```

Windows

Run the AWS Windows support tool:

https://github.com/awslabs/aws-support-tools/tree/master/MGN/Windows

Include the following information in your support case:

- Agent log files (`agent.log.0`)
- Diagnostic tool output
- Operating system version
- Agent version

Agent installation prerequisites

Before installing the AWS Replication Agent, verify that your source server meets the following requirements.

Topics

- Common requirements
- Linux prerequisites
- Windows prerequisites

Common requirements

The following requirements apply to both Linux and Windows source servers.

- **64-bit operating system** — The AWS Replication Agent supports `x86_64` source servers and `arm64` Linux source servers that run on Amazon EC2. It does not support 32-bit systems or Windows on `arm64`.
- **Supported operating system version** — Verify that your OS is on the supported list. See Supported Linux operating systems or Supported Windows operating systems.
- **AWS credentials** — You must have an IAM user or role with the `AWSElasticDisasterRecoveryAgentInstallationPolicy` managed policy attached. The installer uses these credentials to register the source server with Elastic Disaster Recovery.
- **Network connectivity** — The source server must allow outbound TCP 443 to the Elastic Disaster Recovery, Amazon S3, and Amazon EC2 endpoints in your target Region. The agent uses these connections to communicate with the replication infrastructure.
- **Elastic Disaster Recovery initialized in target Region** — You must initialize AWS Elastic Disaster Recovery in the target Region before installing the agent. See Initializing Elastic Disaster Recovery.
- **Volume limits** — Each volume can be a maximum of 16 TiB. Each source server can have a maximum of 63 volumes. Volumes that exceed these limits are not replicated.

Linux prerequisites

Verify the following requirements on Linux source servers before running the installer.

- **Root or sudo access** — The installer must run as root or with sudo privileges to install kernel modules and system services.
- **Disk space: 4 GB free on /** — The agent binaries and replication data require at least 4 GB on the root filesystem. Verify with:

```
df -h /
```

- **Disk space: 500 MB free on /tmp** — The installer extracts temporary files to `/tmp` during installation. If `/tmp` is not a separate mount, this space is part of the 4 GB requirement on `/`. Verify with:

```
df -h /tmp
```

- **Kernel headers matching running kernel** — The kernel headers must match the running kernel version exactly.

> **Note**
>
> The agent builds a kernel driver at install time to capture block-level writes. Without matching headers, the driver compilation fails and the agent cannot replicate data.

Verify the running kernel version:

```
uname -r
```

Then verify matching headers are installed. On RHEL/CentOS/SUSE:

```
rpm -qa | grep 'kernel.*devel'
```

On Debian/Ubuntu:

```
dpkg -l | grep linux-headers
```

- **Required packages: gcc, make** — These packages are required for kernel driver compilation. Verify with:

```
which gcc && which make
```

- **/tmp must allow execution** — The installer runs executables from `/tmp`. If `/tmp` is mounted with `noexec`, the installation fails. Verify with:

```
mount | grep /tmp
```

  If the output shows `noexec`, remount temporarily with:

```
sudo mount -o remount,exec /tmp
```

- **No conflicting disk drivers** — Certain disk filter drivers conflict with the replication driver.
  - *Oracle ASM Filter Driver (ASMFD)* — You must deactivate ASMFD before installation. The replication driver cannot coexist with ASMFD on the same volumes.
  - *PowerPath* — PowerPath requires special handling. Contact AWS Support before installing the agent on servers that use PowerPath.

Windows prerequisites

Verify the following requirements on Windows source servers before running the installer.

- **Administrator access** — The installer must run with Administrator privileges to install the replication driver and system services.
- **.NET Framework 3.5 or later** — The agent requires .NET Framework for its Windows components. Verify on Windows Server:

```
Get-WindowsFeature Net-Framework-Core
```

  On desktop Windows, check Programs and Features for Microsoft .NET Framework 3.5 or later.

- **Disk space: 1 GB free on C:** — The agent binaries and service files require at least 1 GB on the system drive. Verify with:

```
Get-PSDrive C
```

- **net.exe and sc.exe in system PATH** — The installer uses these utilities to manage Windows services. They are located in `C:\Windows\System32` by default. Verify with:

```
where.exe net.exe
```

- **BitLocker must be disabled** — BitLocker encryption prevents the replication driver from reading block-level data. You must disable BitLocker on all volumes that you want to replicate. Verify with:

```
manage-bde -status
```

- **TLS 1.2 must be enabled** — Elastic Disaster Recovery endpoints require TLS 1.2. Older Windows versions (such as Windows Server 2012) might default to TLS 1.0, which Elastic Disaster Recovery endpoints reject.

> **Note**
>
> If your server uses Windows Server 2012 or earlier, verify that TLS 1.2 is enabled in the Windows Registry before running the installer.

- **Use PowerShell to run the installer** — Run the installer from PowerShell, not CMD. CMD has known issues with credential pasting that can cause authentication failures during installation.

Linux agent installation errors

This topic covers errors that you might encounter during or after installing the AWS Elastic Disaster Recovery agent on Linux source servers. Each section describes an error message, its cause, and the resolution.

Topics

- Error: Invalid disk path format
- Error: Kernel headers version mismatch
- Error: GLIBC version not found
- Error: Unsupported Linux kernel version
- Error: gcc not found
- Error: Permission denied when loading kernel driver
- Error: Failed to create system user

- Error: Failed to map segment from shared object
- Error: Multipath disk detection failure
- Error: PowerPath multipath detected
- Error: Driver compiled for a different kernel
- Error: Agent driver build configuration failed
- Error: Agent driver compilation failed
- Error: SUSE kernel headers package not found
- Error: Oracle ASM Filter Driver requires a restart
- Error: Invalid driver state location

Error: Invalid disk path format

Error message: Installation fails when you specify disks to replicate.

Cause: Apostrophes, brackets, or non-existent paths were used in the disk list.

Resolution: Use comma-separated paths only. Do not include quotes, brackets, or spaces. For example:

```
/dev/sda,/dev/sdb
```

Error: Kernel headers version mismatch

Error message: Cannot find kernel headers or the driver build fails during installation.

Cause: The agent builds a kernel driver at install time. This requires `kernel-devel` (RHEL/CentOS/SUSE) or `linux-headers` (Debian/Ubuntu) matching the exact running kernel version.

Resolution: Complete the following steps to install the correct kernel headers.

1. Check the running kernel version:

   ```
   $ uname -r
   ```

2. Check installed headers:

   - **RHEL/CentOS/SUSE:**

     ```
     $ rpm -qa | grep 'kernel.*devel'
     ```

- **Debian/Ubuntu:**

```
$ dpkg -l | grep linux-headers
```

3. Check whether the header directory is a symlink. If it is a symlink, remove it:
   - **RHEL/CentOS/SUSE:**

```
$ ls -l /usr/src/kernels
```

   - **Debian/Ubuntu:**

```
$ ls -l /usr/src
```

   If a symlink exists, remove it with `rm`.
4. Install the correct headers:
   - **RHEL/CentOS:**

```
$ sudo yum install kernel-devel-$(uname -r)
```

   - **SUSE:**

```
$ sudo zypper install kernel-default-devel=$(uname -r | sed "s/-default//")
```

   - **Debian/Ubuntu:**

```
$ sudo apt-get install linux-headers-$(uname -r)
```

5. If the headers are not available in your configured repositories, download them manually from one of the following sources:
   - **RHEL/CentOS/SUSE:** rpm.pbone.net
   - **Debian:** packages.debian.org
   - **Ubuntu:** packages.ubuntu.com

> **Note**
>
> Multiple kernel-headers versions can coexist safely. Installing new headers does not affect the running kernel.

Error: GLIBC version not found

Error message: version 'GLIBC_2.7' not found (required by ./aws-replication-installer-64bit)

Cause: The operating system is unsupported or too old for the agent binary.

Resolution: Verify that your operating system is supported. For more information, see Supported Linux operating systems.

Error: Unsupported Linux kernel version

Error message: Your Linux kernel version is not supported

Cause: The running kernel is not compatible with the AWS Elastic Disaster Recovery replication driver.

Resolution: Check the supported kernels list. If your kernel is listed but the error persists, ensure that the matching kernel headers are installed. For more information, see the Error: Kernel headers version mismatch section.

Error: gcc not found

Error message: gcc was not found and could not be automatically fetched from the configured repositories

Cause: The `gcc` compiler is required to compile the replication driver, and the installer could not install it.

Resolution: Identify which condition prevented the installation and apply the corresponding fix. The following conditions are the most common.

Unreachable package repositories

Check whether the repositories respond:

```
$ sudo apt-get update
```

On RHEL, CentOS, and Amazon Linux, use the following command instead:

```
$ sudo yum makecache
```

On SUSE, use the following command instead:

```
$ sudo zypper refresh
```

If the command fails, restore repository access and run the installer again. If the server reaches the internet through a web proxy, configure that proxy for the package manager as well.

gcc missing from the repositories

Check whether the configured repositories offer `gcc`:

```
$ apt-cache policy gcc
```

On RHEL, CentOS, and Amazon Linux, use the following command instead:

```
$ yum info gcc
```

On SUSE, use the following command instead:

```
$ zypper info gcc
```

If no candidate version is listed, and this server is not permitted to reach an external repository, install `gcc` and `make` from local media or from an internal repository, and then run the installer again.

Package manager lock contention

Check whether another package operation is running:

```
$ pgrep -af 'apt|dpkg|yum|dnf|zypper|unattended'
```

The pattern covers the package managers and the unattended upgrade service, which is a common holder of the lock. Because the command matches full command lines, it can also list itself; disregard that entry. If it lists a package operation, wait for that operation to finish and then run the installer again.

Insufficient free space

Check the file systems that hold the package cache and the installation target:

```
$ df -h /var /usr
```

If either is full, delete unneeded files and run the installer again.

If none of the preceding conditions applies, install `gcc` and `make` manually and then run the installer again:

- **RHEL/CentOS/Amazon Linux:**

```
$ sudo yum install gcc make
```

- **Debian/Ubuntu:**

```
$ sudo apt-get install gcc make
```

- **SUSE:**

```
$ sudo zypper install gcc make
```

Error: Permission denied when loading kernel driver

Error message: insmod: ERROR: could not insert module ./aws-replication-driver.ko: Permission denied or Operation not permitted

Cause: SELinux, Secure Boot, or endpoint protection software is blocking kernel module insertion.

Resolution: Identify which mechanism is blocking the module and apply the corresponding fix.

SELinux

Check SELinux status:

```
$ sestatus
```

Fix the security context on the driver module:

```
$ restorecon /lib/modules/*/extra/aws-replication-driver.ko
```

Secure Boot

Check Secure Boot status:

```
$ mokutil --sb-state
```

If Secure Boot is enabled, you must disable it to use the agent.

> **Important**
>
> Consult your security team before disabling Secure Boot.

Antivirus or endpoint protection

Check your endpoint protection software and add AWS Elastic Disaster Recovery components to the allow list.

Error: Failed to create system user

Error message: Failed to set system user permissions: followed by one of several messages, for example: "getpwnam(): name not found: aws-replication", Unable to change permissions of /var/lib/aws-replication-agent, or sudoers file failed verification ...

Cause: The installer cannot create or configure the `aws-replication` system user because of a file-level restriction.

Resolution: Identify which restriction is blocking user creation and apply the corresponding fix.

Immutable user database files

Check for the immutable attribute:

```
$ lsattr /etc/passwd /etc/group /etc/shadow
```

Remove the immutable attribute:

```
$ sudo chattr -i /etc/passwd /etc/group /etc/shadow
```

Run the agent installer. Re-apply the immutable attribute after installation completes.

> **Important**
>
> The immutable attribute might be intentional security hardening. Consult your system administrator before removing it. Re-apply the attribute after installation.

Installation directory not writable or immutable

Check whether the file system is mounted read-only:

```
$ findmnt -T /var/lib/aws-replication-agent
```

If the OPTIONS column includes `ro`, remount the file system with write permissions and run the installer again.

Check whether the directory has the immutable attribute:

```
$ lsattr -d /var/lib/aws-replication-agent
```

If the output includes `i`, remove the immutable attribute:

```
$ sudo chattr -i /var/lib/aws-replication-agent
```

Run the installer again after removing the attribute.

Invalid sudoers file

Validate the existing `/etc/sudoers` file:

```
$ sudo visudo -c
```

Correct any reported syntax errors, then run the installer again. Your existing `/etc/sudoers` file remains unchanged.

Error: Failed to map segment from shared object

Error message: error while loading shared libraries: libz.so.1: failed to map segment from shared object

Cause: The `/tmp` directory is mounted with the `noexec` option.

Resolution: Use one of the following options:

- **Option 1:** Temporarily remount `/tmp` with execute permissions:

```
$ sudo mount /tmp -o remount,exec
```

- **Option 2:** Set the `TMPDIR` environment variable to a directory with execute permissions:

```
$ TMPDIR='/path/to/exec/dir' sudo ./aws-replication-installer-init
```

> **Note**
>
> The `noexec` option on `/tmp` is a common security hardening measure. The `TMPDIR` alternative avoids modifying mount options.

Error: Multipath disk detection failure

Error message: The installer does not correctly identify disks on a multipath-configured server.

Cause: Automatic disk detection cannot resolve multipath device mappings.

Resolution: Specify disks explicitly with the `--devices` and `--no-prompt` options:

```
$ sudo ./aws-replication-installer-init \
    --region region \
    --aws-access-key-id key \
    --aws-secret-access-key secret \
    --devices /dev/sda,/dev/mapper/mpatha \
    --no-prompt
```

If the installation still fails, add the `--force-volumes` option.

> **Important**
>
> The `--force-volumes` option disables automatic disk detection. Manually verify that all required disks are included in the `--devices` list.

Error: PowerPath multipath detected

Error message: The installation detects EMC PowerPath.

Cause: PowerPath block-level I/O management conflicts with the AWS Elastic Disaster Recovery replication driver.

Resolution: Contact AWS Support for guidance on installing the agent on PowerPath-configured servers. Use the `--force-volumes` option as a potential workaround.

Error: Driver compiled for a different kernel

Error message: The agent log shows that the driver was compiled for a different kernel version and cannot load.

Cause: The kernel was updated (for example, through `yum update` or automated patching) after the agent driver was originally built. This commonly occurs when significant time passes between failover and failback.

Resolution: Reboot the server to load the current kernel, then reinstall the agent. On a recovery instance, reboot and reinstall the agent as a recovery instance.

Error: Agent driver build configuration failed

Error message: Could not configure the AWS Replication Agent kernel driver build.

Cause: The agent builds a kernel driver at install time. The kernel headers are present, but the `configure` step failed before compilation. This usually means the kernel development package (`kernel-devel` or `linux-headers`) for the running kernel is incomplete. It can also mean the kernel build tree at `/lib/modules/$(uname -r)/build` is missing or invalid.

Resolution: Complete the following steps.

1. Confirm the kernel build tree exists and resolves to a valid directory:

```
$ ls -l /lib/modules/$(uname -r)/build
```

2. Reinstall the complete kernel development package that matches the running kernel version:

- **RHEL/CentOS:**

```
$ sudo yum install kernel-devel-$(uname -r)
```

- **Debian/Ubuntu:**

```
$ sudo apt-get install --reinstall linux-headers-$(uname -r)
```

- **SUSE/SLES:**

```
$ sudo zypper install kernel-default-devel=$(uname -r | sed "s/-default//") kernel-
source
```

3. Run the AWS Replication Agent installer again.
4. If the error persists, collect the installation log (`aws_replication_agent_installer.log`), which contains the `configure` output, and contact AWS Support.

Error: Agent driver compilation failed

Error message: Failed to compile the AWS Replication Agent kernel driver.

Cause: The kernel headers are present, but the driver module failed to compile. This error usually occurs when the compiler or toolset is incompatible with the compiler that built the running kernel. For example, a mismatched `gcc` version can cause this error. Insufficient free disk space is another common cause.

Resolution: Complete the following steps.

1. Confirm that `gcc` and `make` are installed, and compare the `gcc` version with the compiler that built the running kernel:

```
$ gcc --version
$ cat /proc/version
```

2. If the versions differ, install a compatible compiler toolchain, then run the installer again.
3. Confirm that there is sufficient free disk space:

```
$ df -h /
```

4. If the error persists, collect the installation log (`aws_replication_agent_installer.log`), which contains the `make` output, and contact AWS Support.

Error: SUSE kernel headers package not found

Error message: Could not find a matching kernel headers package for your SUSE kernel version.

Cause: On SUSE Linux Enterprise Server, the installer searches for the `kernel-source` package that matches the running kernel. It then installs the matching `kernel-default-devel`

headers. The `zypper` search returned no matching package, which usually means the configured repositories do not include the required package.

Resolution: Complete the following steps.

1. Check the running kernel version:

```
$ uname -r
```

2. Refresh the repositories and install the matching kernel development packages:

```
$ sudo zypper refresh
$ sudo zypper install kernel-default-devel=$(uname -r | sed "s/-default//") kernel-
source
```

3. If your repositories do not include the required packages, register the server with the SUSE Customer Center (SCC) to enable the required module. Then run the installer again. Alternatively, contact your SUSE support representative to obtain the correct packages before running the installer.
4. If the error persists, collect the installation log (`aws_replication_agent_installer.log`) and contact AWS Support.

Error: Oracle ASM Filter Driver requires a restart

Error message: Oracle ASM Filter Driver detected. Please reboot to start replication.

Cause: The Oracle ASM Filter Driver (ASMFD) is loaded, but the AWS Elastic Disaster Recovery replication driver has not loaded yet. Installation completes successfully, but replication cannot start until the replication driver loads.

Resolution: Restart the source server. The replication driver loads during startup and replication begins automatically. You do not have to deactivate ASMFD.

Error: Invalid driver state location

Error message: The value "*`location`*" provided for parameter "--driver-state-location" has a value for which the location does not exist, or is not on a supported file system type.

Cause: The path that you provided for the `--driver-state-location` parameter does not exist. Alternatively, the agent cannot write to that file system early in the boot process. For example, the agent cannot write to a Btrfs subvolume at that stage.

Resolution: Use one of the following options:

- Provide an existing directory on a supported file system, such as ext4 or xfs.
- Do not point the `--driver-state-location` parameter at a Btrfs subvolume.
- Omit the `--driver-state-location` parameter so that the installer chooses the location automatically.

Windows agent installation errors

This page covers errors that you might encounter when you install the AWS Elastic Disaster Recovery agent on Windows source servers. Each section describes the error message, its cause, and the resolution.

Topics

- Error: This app can't run on your PC
- Error: Cannot open self or archive
- Error: TLS connection error downloading installer
- Error: The directory is not empty
- Error: Certificate verification failed
- Error: BitLocker is not supported
- Error: net.exe or sc.exe not found

Error: This app can't run on your PC

Error message: This app can't run on your PC when you run `AwsReplicationWindowsInstaller.exe`.

Cause: Windows is running in 32-bit mode. AWS Elastic Disaster Recovery requires a 64-bit operating system.

Resolution:

1. Verify the system architecture. Open **Settings**, choose **System**, and then choose **About**. Check the **System type** field.
2. If the value is `32-bit operating system, x64-based processor`, you are running 32-bit Windows on 64-bit hardware. Reinstall Windows as 64-bit.
3. If the value is `32-bit operating system, x86-based processor`, the hardware does not support 64-bit. You cannot use AWS Elastic Disaster Recovery on this server.
4. If the system is 64-bit but the error still appears, check for AppLocker policies, SmartScreen, or Windows Defender Application Control blocking the executable. Right-click `AwsReplicationWindowsInstaller.exe` and select **Run as administrator**.

Error: Cannot open self or archive

Error message: Cannot open self AwsReplicationWindowsInstaller.exe or archive

Cause: The installer file is corrupted or the download was incomplete due to a network interruption or browser timeout.

Resolution:

1. Re-download the installer from the AWS Elastic Disaster Recovery console or direct URL.
2. Verify the file hash matches the published hash:
   a. Download the published hash:

```
https://aws-elastic-disaster-recovery-hashes-REGION.s3.REGION.amazonaws.com/
latest/windows/AwsReplicationWindowsInstaller.exe.sha512
```

   b. Compute the local hash in PowerShell:

```
Get-FileHash .\AwsReplicationWindowsInstaller.exe -Algorithm SHA512
```

   c. Compare the two values. If they differ, download the installer again.
3. If the hash matches but the error persists, check if antivirus software quarantined or modified the file.

Error: TLS connection error downloading installer

Error message: The underlying connection was closed: An unexpected error occurred on a send

Cause: The Windows server is using TLS 1.0 or 1.1 by default. AWS Elastic Disaster Recovery endpoints require TLS 1.2. This is common on Windows Server 2012 and earlier.

Resolution:

- **Per-session fix (PowerShell):**

```
[System.Net.ServicePointManager]::SecurityProtocol =
 [System.Net.SecurityProtocolType]::Tls12
```

- **Permanent fix:** Enable TLS 1.2 system-wide through the registry:

```
Set-ItemProperty -Path 'HKLM:\SOFTWARE\Microsoft\.NETFramework\v4.0.30319' -Name
 'SchUseStrongCrypto' -Value 1 -Type DWord
Set-ItemProperty -Path 'HKLM:\SOFTWARE\Wow6432Node\Microsoft\.NETFramework
\v4.0.30319' -Name 'SchUseStrongCrypto' -Value 1 -Type DWord
```

- Restart PowerShell after applying registry changes.

Error: The directory is not empty

Error message: OSError: [WinError 145] The directory is not empty

Cause: A previous installation attempt was interrupted, leaving temporary files that cannot be cleaned up. This can also be caused by antivirus locking files or Group Policy restrictions.

Resolution:

1. Run the installer as Administrator. Right-click `AwsReplicationWindowsInstaller.exe` and select **Run as administrator**.
2. Verify the user has Full Control permissions on the temporary directory shown in the error.
3. Temporarily disable antivirus or endpoint protection software.
4. Check that no Group Policy Object (GPO) blocks deletion of temporary files.
5. If a previous partial installation exists, uninstall it first through Add/Remove Programs.

Error: Certificate verification failed

Error message: CERTIFICATE_VERIFY_FAILED or certificate verify failed

Cause: The Windows certificate store does not contain the Amazon Trust Services root CA certificates required to validate AWS Elastic Disaster Recovery endpoint TLS connections. This occurs on air-gapped servers or systems that cannot reach Windows Update.

Resolution:

- **Option 1 (internet access):** Run Windows Update, or open Microsoft Edge once. Windows auto-downloads trusted root CAs when a browser first connects to HTTPS sites.
- **Option 2 (no internet or air-gapped):** Download Amazon root certificates from https://www.amazontrust.com/repository/ and import them:

```
Import-Certificate -FilePath .\AmazonRootCA1.cer -CertStoreLocation Cert:
\LocalMachine\Root
```

> **Important**
>
> Modifying the trusted certificate store is typically managed by system administrators. Consult your admin before importing certificates on production servers.

Error: BitLocker is not supported

Error message: BitLocker is not supported. Please disable BitLocker and try again

Cause: AWS Elastic Disaster Recovery performs block-level replication and cannot read data from BitLocker-encrypted volumes. The replication driver requires direct block access.

Resolution:

1. Disable BitLocker on all volumes to be replicated:

```
manage-bde -off C:
```

2. Wait for decryption to complete. Check the status:

```
manage-bde -status C:
```

3. Run the installer again after decryption completes.

> **Note**
>
> BitLocker must remain disabled while using AWS Elastic Disaster Recovery.

> **Important**
>
> Disabling BitLocker decrypts the drive. Verify this is acceptable per your organization's security policy before proceeding.

Error: net.exe or sc.exe not found

Error message: Installation fails because `net.exe` or `sc.exe` cannot be located.

Cause: The `C:\Windows\System32` directory is not included in the system PATH environment variable.

Resolution:

1. Verify in PowerShell:

```
$env:Path -split ';' | Where-Object { $_ -like '*System32*' }
```

2. If System32 is missing from PATH, add it permanently:

```
$machinePath = [System.Environment]::GetEnvironmentVariable('Path', 'Machine')
[System.Environment]::SetEnvironmentVariable('Path', $machinePath + ';C:\Windows
\System32', 'Machine')
```

3. Restart the PowerShell session and run the installer again.

Common agent installation errors

This topic covers platform-agnostic installation errors related to AWS credentials, account configuration, IAM, and agent lifecycle. These errors occur on both Linux and Windows.

Topics

- Error: Outdated agent installer version

- Error: Account not initialized
- Error: Failed to validate AWS credentials
- Error: Request signature mismatch
- Error: Missing agent installation policy
- Error: Agent IAM role missing
- Error: Server registered to different Region or account
- Error: Reboot required after uninstallation
- Error: Volume exceeds size limit
- Error: Source server already exists
- Error: Missing marketplace license permissions
- Error: Secure connection failed while downloading installation files
- Error: DRS cannot confirm that the arm64 source is an EC2 instance
- Error: Operating system is not supported
- Error: Operating system has reached its end of life
- Error: Invalid endpoint
- Error: Connection attempt failed on port 443
- Error: Root or administrator privileges required

Error: Outdated agent installer version

Error: Installation fails with version-related errors or unexpected behavior.

Cause: You are using an old version of the installer. The installer does not self-update in all scenarios.

Resolution:

1. Download the latest installer from the AWS Elastic Disaster Recovery console. Choose **Source servers**, then choose **Add server**.
2. On Linux, check the installed agent version:

```
cat /var/lib/aws-replication-agent/agent.version
```

3. If a previous agent is installed, uninstall it first, reboot the server, then install the new version.
4. For download instructions, see Adding source servers.

Error: Account not initialized

Error: `AWS Replication Agent installation failed due to the account not being initialized`

Cause: AWS Elastic Disaster Recovery has not been initialized in the target Region. Initialization creates the required service-linked roles and replication infrastructure.

Resolution:

1. Initialize AWS Elastic Disaster Recovery by following Elastic Disaster Recovery initialization and permissions.
2. Run the installer again after initialization completes.

Error: Failed to validate AWS credentials

Error: `Failed to validate AWS credentials`

Cause: The AWS Access Key ID or Secret Access Key provided during installation is incorrect, expired, or malformed.

Resolution:

1. Verify that the credentials are active in the IAM console.
2. If you are using temporary credentials (STS), check the expiration time.
3. Test the credentials independently:

```
aws sts get-caller-identity --region region
```

4. On Windows, use PowerShell instead of CMD to avoid special character pasting issues with secret keys.
5. Ensure that the Region specified during installation matches the Region where AWS Elastic Disaster Recovery is initialized.

Error: Request signature mismatch

Error: InvalidSignatureException... The request signature we calculated does not match the signature you provided

Cause: The installer signed the request with a corrupted AWS Secret Access Key, so the computed signature does not match. This commonly occurs when a command shell interprets special characters in the key and alters it during entry. A significantly skewed system clock can also cause this error.

Resolution:

1. Re-enter the Secret Access Key and make sure you paste it exactly. On Windows, use PowerShell instead of CMD, or enclose the key in double quotes.
2. Make sure the system clock is accurate. For example, synchronize with Network Time Protocol (NTP).
3. Verify the credentials with the following command:

   ```
   aws sts get-caller-identity
   ```

   Then run the installer again.

Error: Missing agent installation policy

Error: `User is not authorized to perform` or access denied errors during installation.

Cause: The IAM user or role used for installation does not have the required permissions.

Resolution:

1. Attach the `AWSElasticDisasterRecoveryAgentInstallationPolicy` managed policy to the IAM user or role.

   Policy ARN: `arn:aws:iam::aws:policy/AWSElasticDisasterRecoveryAgentInstallationPolicy`
2. Verify the policy attachment for an IAM user:

   ```
   aws iam list-attached-user-policies --user-name username
   ```
3. If you are using an IAM role, verify the policy attachment:

   ```
   aws iam list-attached-role-policies --role-name role-name
   ```

Error: Agent IAM role missing

Error: Installation fails because required service roles do not exist.

Cause: The AWS Elastic Disaster Recovery service roles were not created during initialization or were manually deleted. These roles are:

- `AWSElasticDisasterRecoveryReplicationServerRole`
- `AWSElasticDisasterRecoveryConversionServerRole`
- `AWSElasticDisasterRecoveryRecoveryInstanceRole`

Resolution:

1. Reinitialize AWS Elastic Disaster Recovery from the console. This recreates the roles.
2. For more information, see Elastic Disaster Recovery initialization and permissions.

Error: Server registered to different Region or account

Error: `Cannot install agent, as this server was previously installed to replicate into another region or account`

Cause: The source server's agent configuration file references a previous AWS Elastic Disaster Recovery Region or account.

Resolution:

> **Important**
>
> Resolving this requires disconnecting and deleting the existing source server from the AWS Elastic Disaster Recovery console. This removes the server from AWS Elastic Disaster Recovery and terminates its replication resources. Consult your DR administrator before proceeding.

1. Disconnect and delete the source server from the AWS Elastic Disaster Recovery console in the previously configured Region or account.
2. Run the installer again with the correct Region and credentials.

Error: Reboot required after uninstallation

Error: `The server has not been restarted since agent uninstallation`

Cause: The previous agent was uninstalled but the kernel driver is still loaded in memory. A reboot is required to fully remove it.

Resolution:

1. Reboot the source server.
2. Run the installer again after the reboot completes.

Error: Volume exceeds size limit

Error: Error indicating a volume is too large for replication.

Cause: A source volume exceeds the AWS Elastic Disaster Recovery size limits.

AWS Elastic Disaster Recovery volume limits:

- Maximum volume size: 16 TiB per volume
- Maximum boot volume size: 16 TiB
- Maximum volumes per source server: 63

Resolution:

1. Exclude the oversized volume by using the `--devices` installer parameter (Linux) or replication settings.
2. Alternatively, reduce the volume size on the source server before installing.

Error: Source server already exists

Error: `already exists` during installation.

Cause: The source server is already registered with AWS Elastic Disaster Recovery in this Region and account.

Resolution:

- If reinstalling on the same server: run the installer again without providing tags. You cannot update tags during installation. Use the AWS Elastic Disaster Recovery console or API to modify tags.
- If registering as a new source server: disconnect and delete the existing source server from the AWS Elastic Disaster Recovery console first.

Error: Missing marketplace license permissions

Error: `Missing permissions to retrieve marketplace licenses from the source account`

Cause: The IAM credentials used for installation do not have permission to access AWS Marketplace product codes. This error only occurs when replicating Amazon EC2 instances that use Marketplace AMIs.

Resolution:

1. Add the `ec2:DescribeInstances` permission to the IAM user or role.
2. This permission is needed to retrieve Marketplace product codes from the source instance for license compliance.

Error: Secure connection failed while downloading installation files

Error: Failed to establish a secure connection while downloading the AWS Replication Agent installation files.

Cause: The installer could not complete a TLS handshake when it downloaded the agent installation files from Amazon S3. An intercepting proxy or a TLS-inspecting firewall might present a certificate that the installer does not trust. The certificate authority (CA) trust store on the source server might also be missing or outdated.

Resolution: Complete the following steps:

1. Allow direct HTTPS egress from the source server to the Amazon S3 endpoints in your target Region, without TLS inspection.
2. If you use a proxy, verify that the operating system trust store on the source server trusts the proxy CA certificate.

3. Verify that the system time on the source server is correct. Clock skew invalidates certificates that are otherwise valid.

Error: DRS cannot confirm that the arm64 source is an EC2 instance

Error: ARM64 recovery is only supported for verified EC2 instances

A verified Amazon EC2 instance is one that Elastic Disaster Recovery can confirm is an `arm64` Amazon EC2 instance with the instance ID that the agent reports. When you install the agent on an `arm64` source server, Elastic Disaster Recovery confirms this from the instance's signed Amazon EC2 instance identity document, which the agent reads from the instance metadata service.

Cause: The `arm64` source server is not an Amazon EC2 instance, Elastic Disaster Recovery could not read its signed identity document, or the document reports a different instance ID or architecture.

Resolution: Install the agent on the `arm64` Amazon EC2 instance itself. AWS Elastic Disaster Recovery does not support `arm64` source servers in on-premises, vCenter, or other cloud environments. Make sure that the instance can reach the Amazon EC2 instance metadata service. For more information, see AWS DRS supported Linux operating systems.

Error: Operating system is not supported

Error: The operating system is not supported by the AWS Replication Agent.

Cause: The source operating system is not supported. For `arm64`, this includes Windows on `arm64`. Elastic Disaster Recovery checks the operating system during installation.

Resolution: `arm64` source servers must run a 64-bit Linux operating system on an Amazon EC2 instance. Windows on `arm64` is not supported. For `x86_64` source servers, verify that Elastic Disaster Recovery supports the operating system, then run the installer again. For the supported versions, see AWS DRS supported Linux operating systems or Supported Windows operating systems.

Error: Operating system has reached its end of life

Error: The operating system has reached its vendor end of life and is not supported for new source servers.

Cause: The operating system of the source server has passed the end-of-life date published by its vendor. AWS Elastic Disaster Recovery does not accept new source servers on an operating system

that the vendor no longer supports. Unsupported operating systems can no longer receive security or compatibility fixes.

Resolution: Upgrade the source server to an operating system version that its vendor still supports. Then run the installer again. Source servers that were already registered before the end-of-life date continue to replicate, and can be recovered while they have an approved exception.

Error: Invalid endpoint

Error: The AWS SDK returns this error: Invalid endpoint: *`value`*

Cause: The value that you provided for the `--endpoint` parameter, or a malformed `--region` value, is not a well-formed endpoint URL.

Resolution: Use one of the following options:

- Omit the `--endpoint` parameter so that the installer uses the default Regional endpoint.
- If you must set the `--endpoint` parameter, provide a valid HTTPS URL.
- Verify that the `--region` value is a valid AWS Region code, for example `us-east-1`.

Error: Connection attempt failed on port 443

Error: The installer cannot reach the AWS Elastic Disaster Recovery endpoint and returns this error: Connection attempt to *`region`* on port 443 failed.

Cause: The source server cannot open an outbound TCP connection on port 443 to the AWS Elastic Disaster Recovery endpoints. This error is usually caused by one of the following:

- A missing route from the source server to the AWS Elastic Disaster Recovery endpoints
- A firewall that blocks outbound connections, either on the source server or on a network appliance
- An incorrect web proxy configuration on the source server, for example a proxy that does not pass HTTPS traffic

Resolution: Verify that your firewall, security group, and web proxy configuration allow outbound traffic on port 443 to the following endpoints:

- `drs.`*`region`*`.amazonaws.com`

- s3.*region*.amazonaws.com

To test connectivity on Linux, run the following command on the source server:

```
$ curl -v https://drs.region.amazonaws.com
```

To test connectivity on Windows, run the following command on the source server:

```
Test-NetConnection drs.region.amazonaws.com -Port 443
```

To avoid this error, verify connectivity to these endpoints from the source server before you run the AWS Elastic Disaster Recovery agent installer.

Error: Root or administrator privileges required

Error: The installer exits immediately without making any changes. On Linux, the installer returns the following error message:

You do not have enough privileges to run this application installer. Run the installer again, using root privileges.

On Windows, the installer returns the following error message:

Please run this script as Administrator.

Cause: You ran the installer without the privileges it requires. The installer checks for sufficient privileges before it does any work, so it does not modify the source server.

Resolution: On Linux, run the installer with `sudo`:

```
$ sudo ./aws-replication-installer-init
```

On Windows, open a Command Prompt or PowerShell window with **Run as administrator** and run the installer from there.

Troubleshooting replication errors

Use this section to troubleshoot issues with data replication between your source servers and the AWS Elastic Disaster Recovery staging area.

If you see...	See
Connection timeouts, port 443/1500 blocked, or need to verify network paths	Verifying network connectivity
"Agent not seen", "Disconnected", "Failed to authenticate", or "Failed to connect" errors	Agent communication errors
"Failed to launch replication server", "Failed to create staging disks", or firewall rule errors	Replication infrastructure errors
"Not converging", replication lag growing, or unknown replication errors	Replication performance errors
Errors in the agent log on the source server (identity, driver, connection, credentials)	Agent runtime errors
Need to calculate bandwidth requirements or measure source server write speed	Bandwidth requirements

Topics

- Verifying network connectivity
- Replication errors: agent communication
- Replication infrastructure errors
- Replication performance errors
- Replication errors: agent runtime errors
- Replication bandwidth requirements

Verifying network connectivity

AWS Elastic Disaster Recovery requires two network paths from the staging area for replication to succeed. TCP port 443 provides API communication with AWS service endpoints. TCP port 1500 provides data replication between source servers and replication servers. Use this page to verify both paths and resolve connectivity issues.

Topics

- Network requirements
- Verifying TCP port 443
- Verifying TCP port 1500
- Resolving port 1500 issues
- Common network configuration issues

Network requirements

Elastic Disaster Recovery replication servers require the following network connectivity:

- **TCP port 443 outbound** from the staging area subnet to the following endpoints:
  - `drs.region.amazonaws.com`
  - `s3.region.amazonaws.com`
  - `ec2.region.amazonaws.com`
- **TCP port 1500 inbound** to the staging area from source servers, or outbound from source servers to the replication server private or public IP address.

For a complete list of regional endpoints, see AWS Elastic Disaster Recovery endpoints.

Verifying TCP port 443

Use the following procedures to verify that the staging area networking configuration permits outbound TCP port 443 traffic to AWS service endpoints.

Console

To verify outbound port 443 in the VPC console

1. Open the Amazon VPC console and navigate to **Route Tables**.
2. Find the route table associated with your staging area subnet. Verify that a route to `0.0.0.0/0` exists with a target of an internet gateway, NAT gateway, or VPN gateway.
3. Choose **Network ACLs** in the navigation pane. Verify that the network ACL associated with the staging area subnet allows outbound traffic on TCP port 443 and allows inbound traffic on ephemeral ports (1024–65535).

4. Choose **Security Groups** in the navigation pane. Find the replication server security group and verify that it allows outbound traffic on TCP port 443 to `0.0.0.0/0` or to the specific service endpoint IP ranges.

CLI

To verify outbound port 443 with the AWS CLI

1. Verify the route table for the staging area subnet:

```
aws ec2 describe-route-tables \
    --filters Name=association.subnet-id,Values=subnet-id \
    --query 'RouteTables[0].Routes[*].
{Dest:DestinationCidrBlock,GatewayId:GatewayId,NatGatewayId:NatGatewayId,State:State}'
```

2. Check the network ACL rules for the staging area subnet:

```
aws ec2 describe-network-acls \
    --filters Name=association.subnet-id,Values=subnet-id \
    --query 'NetworkAcls[0].Entries[*].
{RuleNum:RuleNumber,Protocol:Protocol,Action:RuleAction,CIDR:CidrBlock,PortRange:PortRan
```

3. Retrieve the security group IDs assigned to the replication servers:

```
aws drs get-replication-configuration \
    --source-server-id server-id \
    --query 'replicationServersSecurityGroupsIDs'
```

4. Verify the outbound rules for the replication server security group:

```
aws ec2 describe-security-groups \
    --group-ids sg-id \
    --query 'SecurityGroups[0].IpPermissionsEgress[*].
{Port:ToPort,CIDR:IpRanges[0].CidrIp}'
```

Verifying TCP port 1500

Run connectivity tests from your source server to confirm that TCP port 1500 is reachable on the replication server.

Linux

Use nc (netcat) to test connectivity:

```
nc -zv replication-server-ip 1500
```

If nc is not available, use `telnet`:

```
telnet replication-server-ip 1500
```

A successful connection confirms that TCP port 1500 is open between your source server and the replication server.

Windows

Use PowerShell to test connectivity:

```
Test-NetConnection -ComputerName replication-server-ip -Port 1500
```

A result of `TcpTestSucceeded : True` confirms that TCP port 1500 is open between your source server and the replication server.

Resolving port 1500 issues

Check your network configuration to identify and resolve port 1500 connectivity issues.

Console

To verify port 1500 network configuration in the VPC console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose **Network ACLs**. Verify that the inbound rules for the staging area subnet allow TCP port 1500.
3. Verify that the outbound rules for the network ACL allow the ephemeral port range (1024–65535).
4. In the navigation pane, choose **Route tables**. Verify that the route table for the staging area subnet has a route for inbound traffic from the source server network.
5. In the navigation pane, choose **Security groups**. Locate the security group attached to the replication servers and verify that it allows inbound TCP port 1500.

CLI

To verify port 1500 network configuration with the AWS CLI

1. Check that the network ACL allows inbound TCP port 1500 and outbound ephemeral ports:

```
aws ec2 describe-network-acls \
    --filters "Name=association.subnet-id,Values=staging-subnet-id"
```

   Verify that the inbound rules include an allow entry for TCP port 1500 and that the outbound rules include an allow entry for TCP ports 1024–65535.

2. Get the security group IDs for your replication servers:

```
aws drs get-replication-configuration \
    --source-server-id source-server-id \
    --query 'replicationServersSecurityGroupsIDs'
```

3. Verify that the security group allows inbound TCP port 1500:

```
aws ec2 describe-security-groups \
    --group-ids sg-id \
    --query 'SecurityGroups[0].IpPermissions[?ToPort==`1500`]'
```

4. Check the source server firewall rules. Do one of the following:

   - **Linux** — Run one of the following commands to check firewall rules:

```
iptables -L -n | grep 1500
```

```
firewall-cmd --list-all
```

   - **Windows** — Run the following PowerShell command to check firewall rules:

```
Get-NetFirewallRule | Where-Object {$_.Enabled -eq 'True'} |
  Get-NetFirewallPortFilter | Where-Object {$_.RemotePort -eq '1500' -or
 $_.LocalPort -eq '1500'}
```

Common network configuration issues

The following issues commonly affect TCP port 443 or TCP port 1500 connectivity for Elastic Disaster Recovery replication:

- **DHCP options set misconfigured** — The VPC DHCP options set specifies incorrect DNS servers, preventing endpoint name resolution.
- **Route table missing internet route** — The staging area subnet route table has no route to `0.0.0.0/0` through an internet gateway or NAT gateway.
- **Network ACL denying traffic** — A network ACL rule denies outbound TCP port 443, inbound TCP port 1500, or denies ephemeral port traffic for return packets.
- **Security group restricting traffic** — The replication server security group does not allow outbound TCP port 443 or inbound TCP port 1500.
- **NAT gateway issues** — The NAT gateway is in a private subnet without internet access, or all associated Elastic IP addresses are exhausted.
- **VPC endpoint policy blocking access** — If you use VPC endpoints, the endpoint policy might deny the required API calls for AWS Elastic Disaster Recovery, Amazon S3, or Amazon EC2.
- **Source server firewall blocking outbound 1500** — A host-based firewall on the source server blocks outbound TCP port 1500 connections.
- **"Use private IP" setting misconfigured** — The **Use private IP for data replication** setting does not match your network topology. Enable this setting only when the source server can reach the replication server private IP address directly.

Replication errors: agent communication

The following errors occur when network connectivity issues prevent communication between the AWS Replication Agent, replication server, and AWS Elastic Disaster Recovery service endpoints. Each section describes the error, its cause, and resolution steps.

Topics

- Error: Agent not seen
- Error: Failed to authenticate with service
- Error: Failed to connect agent to replication software
- Error: Failed to establish communication with replication software

- Error: Failed to connect agent to replication server
- Error: Unstable network
- Error: Failback client not seen (replication)

Error: Agent not seen

Error code: AGENT_NOT_SEEN

The source server shows a **Disconnected** status in the AWS Elastic Disaster Recovery console.

Cause: The AWS Elastic Disaster Recovery service has lost communication with the AWS Replication Agent on the source server.

Resolution:

Console

To resolve agent not seen errors (console)

1. Open the AWS Elastic Disaster Recovery console and check the source server status.
2. Verify that the AWS Replication Agent is running on the source server.
   - **Linux:** Run `systemctl status aws-replication-agent`.
   - **Windows:** Open Services and verify that the **AWS Replication Agent** service is running.
3. Verify that the source server can reach the AWS Elastic Disaster Recovery endpoint on TCP port 443.

CLI

To resolve agent not seen errors (CLI)

1. Run the following command to check the source server state:

```
aws drs describe-source-servers \
    --filters sourceServerIDs=source-server-id \
    --query "items[0].
{State:dataReplicationInfo.dataReplicationState,Error:dataReplicationInfo.dataReplicatio
```

2. Verify that the agent is running on the source server.
   - **Linux:**

```
systemctl status aws-replication-agent
```

- **Windows:**

```
Get-Service -Name AwsReplicationService
```

3. Test TCP 443 connectivity to the AWS Elastic Disaster Recovery endpoint.
   - **Linux:**

```
curl -v https://drs.region.amazonaws.com
```

   - **Windows:**

```
Test-NetConnection -ComputerName drs.region.amazonaws.com -Port 443
```

> **Note**
>
> If this error appears on the recovery dashboard, verify that the `AWSElasticDisasterRecoveryRecoveryInstancePolicy` managed policy is associated with the recovery instance IAM role.

Error: Failed to authenticate with service

Error code: FAILED_TO_AUTHENTICATE_WITH_SERVICE

Cause: The replication server cannot reach the AWS Elastic Disaster Recovery endpoint on TCP port 443. This is a staging area network issue.

Resolution:

Console

To resolve authentication errors (console)

1. Open the AWS Elastic Disaster Recovery console and check the staging area subnet configuration for the affected source server.
2. Launch a test Amazon EC2 instance in the staging area subnet.

3. From the test instance, verify TCP 443 connectivity to the AWS Elastic Disaster Recovery endpoint (`drs.region.amazonaws.com`).

CLI

To resolve authentication errors (CLI)

1. Run the following command to retrieve the staging area network configuration:

```
aws drs get-replication-configuration \
    --source-server-id source-server-id \
    --query "{Subnet:stagingAreaSubnetId,SGs:stagingAreaTags}"
```

2. Launch a test instance in the staging area subnet and verify TCP 443 connectivity to `drs.region.amazonaws.com`.

Error: Failed to connect agent to replication software

Error code: FAILED_TO_PAIR_AGENT_WITH_REPLICATION_SOFTWARE

Cause: AWS Elastic Disaster Recovery cannot provide the replication server and agent with the information they need to communicate. This indicates a network connectivity issue between the agent, replication server, and the AWS Elastic Disaster Recovery endpoint.

Resolution:

To resolve agent pairing errors

1. Verify network connectivity between the source server (where the agent runs) and the replication server in the staging area.
2. Verify that both the source server and the replication server can reach the AWS Elastic Disaster Recovery endpoint on TCP port 443.
3. If the issue persists, contact AWS Support.

Error: Failed to establish communication with replication software

Error code: FAILED_TO_ESTABLISH_AGENT_REPLICATOR_SOFTWARE_COMMUNICATION

Cause: Network connectivity issues exist between the agent and the replication server.

Resolution:

To resolve communication errors

1. Verify network connectivity between the agent on the source server, the replication server in the staging area, and the AWS Elastic Disaster Recovery endpoint.
2. Verify that TCP port 1500 is open between the source server and the replication server.

> **Important**
>
> During failback, verify that TCP port 1500 is open for inbound traffic on the recovery instance security group.

Error: Failed to connect agent to replication server

Error code: FAILED_TO_CONNECT_AGENT_TO_REPLICATION_SERVER

Cause: The agent cannot establish a data replication connection with the replication server over TCP port 1500.

Resolution:

Console

To resolve replication server connection errors (console)

1. Open the Amazon EC2 console and locate the security group associated with the staging area replication server.
2. Verify that the security group allows inbound TCP traffic on port 1500.
3. Check the network ACL for the staging area subnet to confirm it allows inbound TCP traffic on port 1500.

CLI

To resolve replication server connection errors (CLI)

1. Retrieve the security group IDs for the replication configuration:

```
aws drs get-replication-configuration \
    --source-server-id source-server-id \
    --query "replicationServersSecurityGroupsIDs"
```

2. Verify that the security group allows inbound TCP port 1500:

```
aws ec2 describe-security-groups \
    --group-ids sg-id \
    --query "SecurityGroups[].IpPermissions[?FromPort==`1500`]"
```

3. Test connectivity to the replication server on port 1500.
   - **Linux:**

     ```
     nc -zv replication-server-ip 1500
     ```

   - **Windows:**

     ```
     Test-NetConnection -ComputerName replication-server-ip -Port 1500
     ```

Error: Unstable network

Error code: UNSTABLE_NETWORK

Cause: Network connectivity between the source server and the replication server is intermittent.

Resolution:

To resolve unstable network errors

1. Verify that network connectivity between the source server and replication server is stable.
2. Run the network bandwidth test to identify bandwidth or latency issues.

Error: Failback client not seen (replication)

Error code: FAILBACK_CLIENT_NOT_SEEN

Cause: A network connectivity issue is preventing the Failback Client from communicating with the AWS Elastic Disaster Recovery endpoint.

Resolution:

To resolve Failback Client connectivity errors

1. Verify that the Failback Client can reach the AWS Elastic Disaster Recovery endpoint on TCP port 443 (`drs.region.amazonaws.com`).
2. Check security group rules and network ACLs to confirm outbound TCP 443 traffic is allowed.

For more information, see Failback Client troubleshooting.

Replication infrastructure errors

This topic covers errors related to the replication server infrastructure, including launching, booting, disk operations, and firewall rules. Each section describes the error message, cause, and resolution.

Topics

- Error: Failed to launch replication server
- Error: Failed to boot replication server
- Error: Failed to create staging disks
- Error: Failed to attach staging disks
- Error: Failed to create firewall rules
- Error: Failed to start data transfer
- Error: arm64 replication server instance type is not supported
- Error: Snapshot failure

Error: Failed to launch replication server

Error message: `FAILED_TO_LAUNCH_REPLICATION_SERVER`

Cause: Elastic Disaster Recovery was unable to launch a replication server in the staging area. Common causes include Amazon EC2 instance limits, IAM permissions, or subnet capacity.

Resolution:

Console

- Check the staging area subnet and instance type in replication settings.
- Check Amazon EC2 Service Quotas for the instance type.
- Verify IAM prerequisites are met.

CLI

Run the following command to check the instance type and subnet configured for replication:

```
aws drs get-replication-configuration \
    --source-server-id SOURCE_SERVER_ID
```

Run the following command to check the Amazon EC2 running instances quota:

```
aws service-quotas get-service-quota \
    --service-code ec2 \
    --quota-code L-1216C47A
```

Compare the quota value against the number of running instances in the staging area.

Error: Failed to boot replication server

Error message: `FAILED_TO_BOOT_REPLICATION_SERVER`

Cause: The replication server was launched but failed to boot. This is usually a staging area network issue that prevents the server from reaching Elastic Disaster Recovery endpoints during startup.

Resolution:

- Verify the staging area subnet has outbound TCP 443 access to the Elastic Disaster Recovery endpoint.
- Check the security group and network ACL associated with the staging area subnet.
- If the issue persists, contact AWS Support.

Error: Failed to create staging disks

Error message: `Failed to create staging disks`

Cause: The AWS account might be configured to encrypt EBS volumes by default, and the IAM user or role lacks permissions to use the selected KMS key.

Resolution:

- Verify IAM prerequisites are met.
- If using default EBS encryption, ensure Elastic Disaster Recovery service roles have permissions on the KMS key (`kms:CreateGrant`, `kms:DescribeKey`).
- Check EBS volume limits in the staging area Region.

Error: Failed to attach staging disks

Error message: `FAILED_TO_ATTACH_STAGING_DISKS`

Cause: Elastic Disaster Recovery could not attach staging disks to the replication server. This can be caused by IAM permission issues or EBS volume attachment limits.

Resolution:

- Verify IAM permissions for Amazon EC2 volume operations.
- Check EBS volume attachment limits in the staging area Region.
- If the issue persists, contact AWS Support.

Error: Failed to create firewall rules

Error message: `Firewall rules creation failed`

Cause: Elastic Disaster Recovery could not configure security group rules for the replication server. This can be caused by missing IAM permissions or invalid replication settings.

Resolution:

- Verify IAM permission prerequisites.
- Review the replication settings of the associated source server.
- Ensure the security group specified in replication settings exists and is in the correct VPC.

Error: Failed to start data transfer

Error message: `FAILED_TO_START_DATA_TRANSFER`

Cause: The agent and replication server were paired but data transfer could not begin. This is usually a network throughput or connectivity issue.

Resolution:

- Check network connectivity and bandwidth between the source server and the replication server.
- Check replication agent logs for details. For more information, see Agent logs.
- If the issue persists, contact AWS Support.

Error: arm64 replication server instance type is not supported

Error message: Instance type '*`instance-type`*' is not supported. Only instance types with 'x86_64' architecture are supported.

Cause: You selected an `arm64` instance type for a replication server.

Resolution: Select an `x86_64` replication server instance type. Replication servers use `x86_64` even when the source server uses `arm64`.

Error: Snapshot failure

Error message: `SNAPSHOTS_FAILURE`

Cause: Elastic Disaster Recovery is unable to take a consistent snapshot. Common causes include inadequate IAM permissions or API throttling.

Resolution:

- Verify IAM permissions and ensure required roles are correctly configured.
- Check if you have activated throttling. For more information, see route control.
- Check CloudTrail logs for throttling errors.

Replication performance errors

This topic covers replication errors related to performance, convergence, and lag in AWS Elastic Disaster Recovery.

Topics

- Error: Replication not converging
- Replication lag increasing
- Error: Unknown data replication error
- Error: Failed to configure replication software
- Error: Failed to download replication software
- Error: Failed to establish communication with recovery instance
- Error: Failed to pair replication agent with replication server

Error: Replication not converging

Error: `NOT_CONVERGING` status

Cause: The rate of data changes on the source server exceeds the available replication bandwidth. Elastic Disaster Recovery cannot catch up with the ongoing writes.

Resolution:

Console

- Check **Replication lag** and **ETA** in the Elastic Disaster Recovery console.
- Check the **Disk settings** tab. Consider upgrading the staging disk type for higher throughput (for example, gp2 to gp3).

CLI

Run the following command to check replication lag and ETA:

```
aws drs describe-source-servers \
    --filters sourceServerIDs=SOURCE_SERVER_ID
```

Run the following command to check disk type, IOPS, and throughput configuration:

```
aws drs get-replication-configuration \
    --source-server-id SOURCE_SERVER_ID
```

Additionally:

- Calculate required bandwidth. For more information, see bandwidth requirements.
- Run a bandwidth test. For more information, see network bandwidth test.

> **Note**
>
> Changing the staging disk type might affect replication costs. Review Amazon EBS pricing before making changes.

Replication lag increasing

Symptom: Replication lag grows over time or spikes unexpectedly.

Causes:

- Source server is down or the agent is not running.
- TCP port 1500 is blocked outbound from the source server to the replication server.
- Source server MAC address changed. This requires agent reinstallation.
- Source server recently rebooted or Elastic Disaster Recovery services restarted. Disks are re-read, and lag grows temporarily until the process completes.
- Source server experienced a spike in write operations. Lag grows until Elastic Disaster Recovery flushes the backlog.
- Insufficient bandwidth for the combined write throughput of all source servers.

Resolution:

- Verify the agent is running and the source server is connected.
- Verify TCP port 1500 connectivity from the source server to the replication server.
- If the lag is temporary (post-reboot or write spike), wait for replication to converge.
- If lag persists, check bandwidth. For more information, see bandwidth requirements.
- Consider upgrading the staging disk type for higher throughput.

Error: Unknown data replication error

Error: `unknown_error`

Cause: An unclassified replication error. Multiple root causes are possible.

Resolution:

- Check connectivity between the source server and the replication server.
- Check AWS CloudTrail for API throttling errors.
- Monitor replication server performance (CPU, memory, disk I/O) in Amazon CloudWatch.
- Verify network bandwidth is adequate. For more information, see bandwidth requirements.
- Check agent logs. For more information, see Agent logs and diagnostics.
- If the error persists, contact AWS Support with the agent logs and source server ID.

Error: Failed to configure replication software

Error: `FAILED_TO_CONFIGURE_REPLICATION_SOFTWARE`

Cause: An internal error occurred during replication software configuration. This is typically transient.

Resolution: Retry the operation. If the error persists, contact AWS Support.

Error: Failed to download replication software

Error: `FAILED_TO_DOWNLOAD_REPLICATION_SOFTWARE_TO_FAILBACK_CLIENT`

Cause: The Failback Client cannot download replication software from Amazon S3. This indicates connectivity issues to the Amazon S3 endpoint, or a proxy or network security appliance is filtering traffic.

Resolution:

- Verify connectivity to the Amazon S3 endpoint. For more information, see TCP port 443 troubleshooting.
- Check for a proxy or network security appliance intercepting or blocking the download.
- Retry the operation.

Error: Failed to establish communication with recovery instance

Error: `FAILED_TO_ESTABLISH_RECOVERY_INSTANCE_COMMUNICATION`

Cause: The Failback Client cannot communicate with the recovery instance.

Resolution:

- If you use a public network (no VPN or Direct Connect), ensure the recovery instance has a public IP address. By default, Elastic Disaster Recovery launch templates disable public IP assignment.
- If you use a private network, verify routing between the Failback Client and the recovery instance.
- Check that the security group on the recovery instance allows inbound traffic on the required ports.

Error: Failed to pair replication agent with replication server

Error: Failed to pair replication agent with replication server

Cause: The replication agent, replication server, and Elastic Disaster Recovery endpoint cannot establish a three-way communication channel.

Resolution:

- Verify connectivity between the agent, the replication server, and the Elastic Disaster Recovery endpoint.
- If the error persists, contact AWS Support.

Replication errors: agent runtime errors

This topic covers messages that the AWS Replication Agent writes to its own log on the source server or recovery instance. The other topics in this section cover replication errors that AWS Elastic Disaster Recovery reports in the console. Use this topic when the console shows no specific error, or to find the cause behind one. To find the agent log, see the section called “Check agent status and logs”.

For each error in this topic, the agent logs a line that starts with Error code: DRS-REPL- and a three-digit number. The line also includes a short summary and a *For troubleshooting, see* link to the matching section.

Topics

- DRS-REPL-001: Server identity changed and no matching recovery instance was found
- DRS-REPL-002: Replication driver failed to load at agent startup
- DRS-REPL-003: Cannot connect to the replication server on TCP port 1500
- DRS-REPL-004: Authentication with the replication server failed
- DRS-REPL-005: Cannot reach AWS Elastic Disaster Recovery or load agent credentials
- DRS-REPL-006: AWS Elastic Disaster Recovery rejected the agent credentials
- DRS-REPL-007: Timed out waiting for the failback connection on TCP port 1500
- DRS-REPL-008: Handshake with the replication server failed
- DRS-REPL-009: Agent ran out of memory and restarted

DRS-REPL-001: Server identity changed and no matching recovery instance was found

Error message: Error code: DRS-REPL-001 - Server identity changed and no matching recovery instance was found

Cause: At startup, the agent detected that the server identity changed and found no matching recovery instance. This happens when:

- A disk with the agent installed was cloned, migrated, or restored onto a different server.
- On the same server, the agent cannot detect or read the identifier that it recorded at installation: the MAC address, the VMware UUID, or the Google Cloud instance ID. For example, it cannot read the VMware UUID from the system BIOS information or reach the Google Cloud metadata server.
- On a recovery instance, the agent could not confirm with AWS Elastic Disaster Recovery that the instance is a recovery instance. For example, the instance has no instance profile with the required permissions, cannot reach AWS Elastic Disaster Recovery over TCP port 443, or is not a recovery instance in this AWS account and Region.

The process exits after 60 seconds, and the service manager restarts it. Replication does not start.

Resolution:

- On a recovery instance, verify the instance profile policy and TCP port 443 access to AWS Elastic Disaster Recovery. See the section called “AWSElasticDisasterRecoveryRecoveryInstancePolicy”. The agent checks again after it restarts. If the error persists, reinstall the agent as a recovery

instance. This procedure first disconnects the instance from AWS. See the section called "Reinstalling the agent on a recovery instance".

- On a server that was cloned, migrated, or restored, or whose identifier changed, reinstall the agent so that it registers as its own source server. See the section called "Reinstalling the agent".
- If the identity changed unexpectedly, contact AWS Support.

DRS-REPL-002: Replication driver failed to load at agent startup

Error message: Error code: DRS-REPL-002 - Replication driver failed to load at agent startup

Cause: At startup, the agent could not load the replication driver or set up the driver device. The process exits, and the service manager restarts it. Common causes include the following:

- On Linux, the agent could not rebuild the driver after a kernel upgrade. The agent rebuilds it automatically, which requires kernel headers that match the new kernel and access to download the driver sources from AWS.
- On Linux, kernel headers that do not match the running kernel
- On Linux, Secure Boot or SELinux settings that block the driver
- Endpoint protection software that blocks the driver

Resolution:

1. On Linux, install kernel headers that match the running kernel, and then reinstall the agent to rebuild the driver. See the section called "Error: Kernel headers version mismatch".
2. On Linux, resolve Secure Boot, SELinux, or endpoint protection settings that block the driver. See the section called "Error: Permission denied when loading kernel driver".
3. On Windows, check the driver service with `sc query AwsReplicationDriver`, confirm that endpoint protection software is not blocking the driver, and then restart the agent.
4. If the driver still fails to load, collect diagnostic information and contact AWS Support. See the section called "Gather diagnostic information for support".

DRS-REPL-003: Cannot connect to the replication server on TCP port 1500

Error message: Error code: DRS-REPL-003 - Cannot connect to the replication server on TCP port 1500

Cause: The agent could not connect to the replication server on TCP port 1500 because the connection was refused, timed out, or had no route. The agent retries every 30 seconds until it connects.

Resolution:

1. Verify that the firewall, route table, and network ACL that you manage allow outbound TCP traffic on port 1500 from the source server to the replication server. See the section called "Verifying TCP port 1500".
2. AWS Elastic Disaster Recovery opens inbound TCP port 1500 on the security group that it creates for the replication server. If you replaced that security group, verify that your security group allows inbound TCP traffic on port 1500. See the section called "Resolving port 1500 issues".
3. If the error persists, contact AWS Support.

DRS-REPL-004: Authentication with the replication server failed

Error message: Error code: DRS-REPL-004 - Authentication with the replication server failed

Cause: The agent connected to the replication server on TCP port 1500, but the connection failed while the agent sent or received the authentication message. For example, a firewall, proxy, or other network device between the source server and the replication server interrupted or inspected the TLS connection. The agent retries every 30 seconds.

Resolution:

1. Make sure that no firewall, proxy, or inspection device decrypts or interrupts TCP port 1500 traffic between the source server and the replication server. To verify connectivity, see the section called "Verifying TCP port 1500".
2. If the log shows No available tokens for peer, earlier connection attempts in the same session failed. Fix the network issues first. If this is the only error in the log, contact AWS Support.
3. If authentication still fails, reinstall the agent. If the error returns, contact AWS Support.

DRS-REPL-005: Cannot reach AWS Elastic Disaster Recovery or load agent credentials

Error message: Error code: DRS-REPL-005 - Cannot reach AWS Elastic Disaster Recovery or load agent credentials

Cause: A request to AWS Elastic Disaster Recovery failed before the service responded, because the agent could not reach the endpoint or load its credentials. Common causes are DNS resolution failures, blocked outbound TCP port 443, incorrect proxy settings, or an unavailable credential source. The agent keeps running and retries.

Resolution:

1. Verify DNS resolution and outbound TCP port 443 connectivity from the source server to the AWS Elastic Disaster Recovery endpoint for the target Region. See the section called “Solving communication problems over TCP port 443 between the source servers and AWS Elastic Disaster Recovery”.
2. If the source server uses a proxy, verify the proxy settings.
3. If the agent uses an instance profile, confirm that an instance profile with the required IAM policy is attached. For an agent that you installed with an instance profile, see the section called “Using an instance profile for agent installation in AWS”. For a recovery instance, see the section called “AWSElasticDisasterRecoveryRecoveryInstancePolicy”.
4. If the error persists, contact AWS Support.

DRS-REPL-006: AWS Elastic Disaster Recovery rejected the agent credentials

Error message: Error code: DRS-REPL-006 - AWS Elastic Disaster Recovery rejected the agent credentials

Cause: AWS Elastic Disaster Recovery does not recognize the agent's credentials. For example, AWS access keys stored in the agent configuration were deleted or deactivated. Agents installed with the current installer use a certificate instead of stored access keys. After seven days of rejected requests, the agent stops its regular requests and checks the credentials once every 24 hours until they work.

Resolution:

1. Reinstall the agent to configure new credentials. See the section called “Reinstalling the agent”.
2. Confirm that the agent log no longer shows the invalid security token warning and that replication resumes.

> **Note**
>
> With cross-account roles, a missing or misconfigured `DRSCrossAccountAgentAuthorizedRole` role in the account where the instance runs, or `DRSCrossAccountAgentRole` role in the account that the agent replicates to, causes an AWS STS error for the `AssumeRole` request instead of this message. See the section called “Creating the Failback and in-AWS right-sizing roles”.

DRS-REPL-007: Timed out waiting for the failback connection on TCP port 1500

Error message: Error code: DRS-REPL-007 - Timed out waiting for the failback connection on TCP port 1500

Cause: During failback, the agent on the recovery instance listens on TCP port 1500 for the Failback Client. The agent starts to listen only after the Failback Client finishes configuring the replication software. The Failback Client did not connect within 5 minutes, so the wait timed out. The agent waits 30 seconds and then listens again.

Resolution:

1. On the computer that runs the Failback Client, confirm that the Failback Client is still running and check its status. While it connects, it shows Establishing a connection between the Failback Client and the AWS Replication Agent running on the Recovery Instance over port 1500.... When it succeeds, it shows Connection established. Replication in progress.... If it reports a failure, follow the message and see the section called “Failback Client errors”.
2. Confirm that the Failback Client can reach the recovery instance on TCP port 1500. The Failback Client connects to the private IP address of the recovery instance when you choose **Use private IP** in the failback replication settings. Otherwise, it connects to the public IP address. See the section called “Use private IP”. From the Failback Client network, test the connection to that address on TCP port 1500. On Linux:

   ```
   nc -zv recovery-instance-ip 1500
   ```

   On Windows:

   ```
   Test-NetConnection -ComputerName recovery-instance-ip -Port 1500
   ```

If the test connects, the agent can log the DRS-REPL-004 error once. You can ignore that entry. If the test fails, continue with the next step.

3. Confirm that the security group and network ACL for the recovery instance allow inbound TCP traffic on port 1500 from the Failback Client. See the section called “Error: Failed to get recovery instance volumes”. Also verify that your environment meets the requirements in the section called “Failback prerequisites”.
4. If the error persists, contact AWS Support.

DRS-REPL-008: Handshake with the replication server failed

Error message: Error code: DRS-REPL-008 - Handshake with the replication server failed

Cause: The TCP port 1500 connection opened and the authentication message was sent. While the agent exchanged version, session, and volume information, the connection closed or reset, or returned no response for 5 minutes. A firewall, proxy, or other network device can reset or drop the connection during this exchange. The replication server can also end the session when AWS Elastic Disaster Recovery does not accept the agent or its replication session. During failback, this error can also occur between a recovery instance and the Failback Client. The agent retries every 30 seconds.

On Windows, this error also occurs if the agent cannot map a replicated volume to a local disk. The log then shows runWindowsDeviceMappingCommand failed.

Resolution:

1. If the connection keeps dropping and reconnecting, the agent can log this error repeatedly. First, check whether replication is affected. For a source server, open the **Source servers** page and confirm that **Data replication status** is **Healthy** and that lag is not growing. See the section called “Interacting with the Source Servers page”. For failback, open the recovery instance details page and check the **Data replication status** panel. See the section called “Data replication status”. You can also search the agent log for Connected to replicator after the error. If replication is healthy and lag is not growing, no action is needed.
2. This error happens after the TCP port 1500 connection opens, so a test that only checks whether the port is reachable passes. Instead, check your firewall, proxy, and intrusion prevention logs for TCP port 1500 connections that were reset or dropped after they opened. Exclude TCP port 1500 traffic from deep packet inspection on those devices.

3. If the replication server ended the session, the agent gets new connection details from AWS Elastic Disaster Recovery on its next attempt. If the error continues and replication does not resume, contact AWS Support.
4. On Windows, if the log shows runWindowsDeviceMappingCommand failed, the agent could not map a replicated volume to a local disk. Collect diagnostic information and contact AWS Support. See the section called “Gather diagnostic information for support”.

DRS-REPL-009: Agent ran out of memory and restarted

Error message: Error code: DRS-REPL-009 - Agent ran out of memory and restarted

Cause: The agent process ran out of memory and exited. The service manager restarts the agent after 30 seconds, and replication resumes. The agent sets its maximum memory based on the memory of the server. On some out-of-memory errors, the Java runtime can write a `client.hprof` file to the agent directory. This file contains a copy of the agent memory and can include sensitive data.

Resolution:

1. The agent restarts after this error, and replication resumes. To count how many times the error occurred, search the agent log for the error code. On Linux:

   ```
   sudo grep "DRS-REPL-009" /var/lib/aws-replication-agent/agent.log.0
   ```

   On Windows, in PowerShell run as an administrator:

   ```
   Select-String -Pattern "DRS-REPL-009" -Path "C:\Program Files (x86)\AWS Replication
    Agent\agent.log.0"
   ```

   If the agent does not log error codes, search for OutOfMemoryError instead. On Linux, you can also run `systemctl status aws-replication-agent` to see when the agent last started. If the error appears once and the console shows replication as **Healthy**, no action is needed.
2. If the error appears more than once, collect diagnostic information and contact AWS Support. See the section called “Gather diagnostic information for support”. In your case, include the kind of OutOfMemoryError from the agent log, for example Java heap space or unable to create new native thread.

3. If a `client.hprof` file is present, tell AWS Support in your case and wait for instructions on how to share it. The file is at `/var/lib/aws-replication-agent/client.hprof` on Linux and `C:\Program Files (x86)\AWS Replication Agent\client.hprof` on Windows, in the same folder as `agent.log.0`. The Java runtime does not overwrite an existing `client.hprof` file, so check the file timestamp to confirm that it is from the latest error. Because this file can include sensitive data, delete it after you share it.

Replication bandwidth requirements

AWS Elastic Disaster Recovery replicates data changes from your source servers to AWS over TCP port 1500. The required network bandwidth must exceed the combined write throughput of all source servers that replicate through the same network path. If your available bandwidth is lower than the total write rate, replication can't keep up and data won't converge.

If replication is not converging, verify that your actual network throughput matches or exceeds the calculated requirement. To measure actual throughput, see the section called “Network bandwidth test”.

Topics

- Calculating required bandwidth
- Measuring source server write speed
- Running a network bandwidth test to an EC2 endpoint

Calculating required bandwidth

Use the following formula to determine the minimum bandwidth you need:

```
Minimum bandwidth >= sum of average write speed of all source servers
```

For example, if you replicate two source servers through the same network path and their average write speeds are 5 MBps and 7 MBps, you need at least 12 MBps of available bandwidth.

> **Note**
>
> We recommend adding at least 20% headroom to account for burst writes. If available bandwidth is insufficient, replication lag increases and convergence might fail.

Measuring source server write speed

To determine the write speed of each source server, use the following operating system-specific tools. We recommend collecting data for at least 24 hours to capture both peak and average write patterns.

Linux

Use `iostat` from the `sysstat` package to measure disk write speed.

Install `sysstat`:

```
$ sudo yum install sysstat
```

On Debian or Ubuntu, run the following command instead:

```
$ sudo apt-get install sysstat
```

Run `iostat` with extended statistics, reporting every 3 seconds:

```
$ iostat -x 3
```

In the output, look at the `wkB/s` column (write kilobytes per second) for each disk. The following example shows sample output:

```
Device         rrqm/s wrqm/s   r/s    w/s   rkB/s    wkB/s  %util
sda              0.00   5.00  1.00  15.00   4.00   5120.00  12.50
sdb              0.00   2.00  0.00  10.00   0.00   3072.00   8.00
```

In this example, `sda` writes at 5,120 KB/s (5 MBps) and `sdb` writes at 3,072 KB/s (3 MBps).

Windows

Use Performance Monitor (`perfmon`) to measure disk write speed.

To measure write speed with Performance Monitor

1. Open Performance Monitor.
2. Add the counter **PhysicalDisk** > **Disk Write Bytes/sec** for all disk instances.
3. Monitor for at least 24 hours to identify peak write patterns.

Alternatively, use the following PowerShell command to collect write speed samples:

```
Get-Counter '\PhysicalDisk(*)\Disk Write Bytes/sec' -SampleInterval 3 -MaxSamples 10
```

Running a network bandwidth test to an EC2 endpoint

This procedure measures throughput over TLS on TCP port 1500 between a source server and an Amazon Elastic Compute Cloud instance that you launch as a disposable endpoint. The test exercises the same encrypted path that replication uses. As a result, it reveals TLS interception by a middlebox and the maximum transmission unit (MTU) effects that a plain TCP test can hide. You run the test from one of two clients that drive the same endpoint, on the same port, with the same certificate: a command-line client or a browser. Use the command-line client; it produces the figure you should quote and is the only client that lets you set the number of concurrent streams. Choose the browser instead when the source server has a desktop, or when you want to watch the test run.

> **Note**
>
> This procedure uses a self-signed certificate with a manually supplied trust anchor. Treat the result as a measurement of the network path, not as confirmation that Elastic Disaster Recovery certificate validation will succeed.

Replace every value shown as a *`placeholder`* with one from your own account. Each task sets the shell variables it needs at the point where you use them, on the host that uses them.

Security considerations for the test endpoint

> **Important**
>
> The test endpoint is unauthenticated. There is no login, token, or per-client authorization. Anyone who can reach port 1500 can run a test against it and consume your bandwidth. The only access control is the security group.

Keep the following in mind before you launch the endpoint:

- **Scope ingress on port 1500 to the addresses you test from.** Because the endpoint is unauthenticated, every address the security group admits can run tests and spend your

bandwidth. Narrower ranges mean less exposure. We recommend against opening the rule to a wide range such as `0.0.0.0/0`: an internet-open, unauthenticated, bandwidth-generating endpoint is an abuse target. If no narrower range is workable, make that decision knowingly, and tear the endpoint down as soon as you have your numbers.

- **The instance is disposable and must be terminated.** The main cost risk is forgetting to terminate the instance, along with the data transfer that each run moves. See the section called "Tear down the test rig" and the section called "Costs".
- **Your organization's security tooling might isolate an internet-reachable endpoint.** Many organizations run automated detection that stops or isolates instances that expose an unauthenticated service, and it can act within hours.

Choosing your connectivity path

The endpoint binds to `0.0.0.0:1500` and serves on every address the instance has, so the shape of the procedure is the same in all three cases. What changes is which address you put in the security group and which name you connect by.

Your source server is	Endpoint needs a public IP?	Notes
In the same VPC as the endpoint	No	The complete procedure, including full certificate verification, runs against an endpoint that has no public IP in a subnet with no internet route.
On-premises over VPN or Direct Connect	No	The endpoint serves on its private address, and VPN or Direct Connect delivers to that same private address through Amazon VPC routing. Path MTU and any middlebox on your tunnel are exactly what this test exists to reveal, so they show up in your result. Connect by the endpoint's private IP address, not a DNS name; see the section called "Write the server list".

Your source server is	Endpoint needs a public IP?	Notes
On the public internet	Yes	The endpoint is reachable from the internet. See the section called “Security considerations for the test endpoint”.

The private path is the default in this procedure. The only step that needs internet access from the endpoint is the software download in the section called “Install the endpoint software”, and that step is separable.

> **Note**
>
> From inside the VPC, an instance's public DNS name resolves to its private IP. That is a convenience when you want the private path and an obstacle when you want the public one. The name you connect by has to match the path you intend to measure, not only the certificate.

Prerequisites

You need	Notes
An AWS account and an AWS Region to test into	Use the Region you replicate to.
Permission to launch and terminate an Amazon EC2 instance and to edit its security group	The exact actions depend on how you launch and connect to the instance, so this list cannot be complete for every method. Grant the permissions that your chosen launch and access method require.
The address each source server reaches the endpoint from	For a source server in the same VPC, this is its private IP. For a source server on-premises over VPN or Direct Connect, it is the private address it presents inside the VPC. Only for a source server on the public internet is it the public egress address, which you get from the source server with `curl -sS`

You need	Notes
	`https://checkip.amazonaws.com` . Getting this wrong is the most common reason the pre-flight check in the section called "Run the measurement" reports that the server is not responding.
A VPC and subnet	No internet route is needed unless a source server is on the public internet.
A way to open a shell on the new instance	You need interactive shell access to run the section called "Install the endpoint software". Use whichever method suits your network. For more information, see Connect to your EC2 instance in the *Amazon EC2 User Guide*.

Topics

- Launch the endpoint instance
- Install the endpoint software
- Test with the command line
- Test with the browser
- Tear down the test rig
- Interpreting the result
- Costs
- Troubleshooting the bandwidth test

Launch the endpoint instance

Launch a dedicated Amazon EC2 instance to act as the test endpoint, and allow TCP port 1500 to it from your source servers. For instructions on launching and connecting to an instance, see Get started with Amazon EC2 in the *Amazon EC2 User Guide*. When you launch it, make the following choices:

- Choose the standard Amazon Linux 2023 AMI, not the minimal AMI. You can resolve it from its public SSM parameter: `/aws/service/ami-amazon-linux-latest/al2023-ami-kernel-default-x86_64` for an x86_64 instance, or `/aws/service/ami-amazon-linux-latest/`

`al2023-ami-kernel-default-arm64` for a Graviton (arm64) instance. For the AMI and its SSM parameter names, see AL2023 on Amazon EC2 in the *Amazon Linux 2023 User Guide*.

- Launch it in the staging area subnet from your replication settings, and match those settings as closely as you can, including the **Use private IP for data replication** choice. See the section called "Choosing your connectivity path".
- To test from inside the VPC without an existing source server, launch a second instance in the same VPC. Use its private IP as the source address in the port 1500 rule.
- Choose an instance type large enough that the endpoint is not the bottleneck. For more information, see Amazon EC2 instance network bandwidth in the *Amazon EC2 User Guide*. If you choose a Graviton type, use the arm64 AMI and set `ARCH_TARGET` as described in the section called "Download and unpack the endpoint software".
- Make sure that you can open a shell on the instance, by whichever method you normally use. The method you choose has consequences for the rest of this procedure, because each method has its own setup that you provide at launch time and remove at teardown. SSH, for example, needs a key pair that you select at launch and an inbound rule of its own (such as TCP port 22); Session Manager instead needs an instance profile on the instance and outbound HTTPS to the Session Manager endpoints, and no inbound rule. Provide whatever your chosen method requires. For the options, see Connect to your EC2 instance in the *Amazon EC2 User Guide*, which documents each one. If your egress address rotates across a NAT pool, any inbound access rule needs every address in the pool, the same as the port 1500 rule.
- Give the instance outbound HTTPS access to download the endpoint software in the section called "Download and unpack the endpoint software", or plan to copy the archive to it.

After launch, add an inbound rule for TCP port 1500 from the address of each source server. Put this rule in a security group you create for this test, rather than editing a shared or default security group: the endpoint is unauthenticated, so every instance that shares the group would inherit the open port. Attach that dedicated group to the endpoint alongside whatever security group your shell-access method already requires (for example, one that allows TCP port 22 for SSH). You do not need to move or remove the access rule; only keep the port 1500 rule out of a shared group. For instructions, see Configure security group rules in the *Amazon EC2 User Guide*. Every address that this rule allows can run tests against the endpoint, which has no authentication. Allow only the addresses you test from. See the section called "Security considerations for the test endpoint".

A stop and start changes the public address, so if you stop the instance after you generate the certificate, generate it again in the section called "Generate the TLS certificate".

Install the endpoint software

Everything in this task runs on the Amazon EC2 instance, as an ordinary user rather than root. The procedure needs no elevated privilege. Installing the optional diagnostic tools, such as `sysstat` for the `sar` command in the section called “Do not compare headline figures between runs”, does need `sudo`. The whole task takes about 20 seconds after you have a shell.

Download and unpack the endpoint software

The endpoint is speedtest-rust, an open-source Rust implementation of the LibreSpeed server. It ships as a single binary with a built-in browser test page, which is why this procedure needs no web server, no runtime, and no `sudo`. This task downloads speedtest-rust from librespeed/speedtest-rust on the GitHub website. Review the project before you run it in your account.

> **Important**
>
> This procedure uses third-party, open-source software (librespeed/speedtest-rust here, and librespeed/speedtest-cli in the section called “Test with the command line”) that is not developed, maintained, or supported by AWS. It is provided on an "AS IS" basis, with no guarantee of completeness, accuracy or timeliness, and without warranty or representations of any kind, expressed or implied. In no event will AWS and/or its subsidiaries and/or their employees or service providers be liable to you or anyone else for any decision made or action taken in reliance on this software or for any direct, indirect, consequential, special or similar damages (including any kind of loss), even if advised of the possibility of such damages. AWS is not responsible for the update, validation or support of this software. Review each project and confirm it meets your organization's security and licensing requirements before you run it on servers in your account.

```
export LIBRESPEED_RS_VERSION=1.4.0
export ARCH_TARGET=x86_64-unknown-linux-musl        # aarch64-unknown-linux-gnu on
 Graviton.
export PORT=1500                                   # the TLS port the endpoint listens
 on.
export CERT_DAYS=7                                 # certificate lifetime; the rig
 should not outlive it.

mkdir -p ~/bwtest && cd ~/bwtest
```

```
curl -sSLO "https://github.com/librespeed/speedtest-rust/releases/download/v
${LIBRESPEED_RS_VERSION}/librespeed-rs-${ARCH_TARGET}.tar.xz"
```

Note

The project's GitHub release page lists a SHA-256 digest beside each archive. The digest covers the archive, not the extracted binary, so hash the archive and compare it with that digest before you unpack it:

```
sha256sum "librespeed-rs-${ARCH_TARGET}.tar.xz"
```

```
tar -xJf "librespeed-rs-${ARCH_TARGET}.tar.xz"
chmod +x librespeed-rs
```

The archive is about 11 MB and unpacks to roughly 48 MB, of which a 42 MB ASN database is never used by this procedure.

Note

Downloading the archive is the only step that needs internet access from the endpoint. Use one of the following three ways to satisfy it, in order of preference:

1. Download the archive on a machine that has egress and copy it in. An endpoint in a subnet with no internet route can be staged this way, and everything downstream behaves identically. Then continue from the `tar -xJf` command.

   ```
   scp -i /path/to/key.pem librespeed-rs-*.tar.xz USER@PRIVATE_IP:~/bwtest/
   ```

2. Put a NAT gateway on the subnet, if one is not already there.
3. Give the instance a public IP for the install, and accept the exposure described in the section called “Security considerations for the test endpoint”.

Delete the shipped config file

```
rm configs.toml
```

> **Important**
>
> The shipped file points at an assets directory that does not exist at your install path. Left in place, the program logs Config assets directory failed ! and then serves 404 on /, so the browser page is dead although the CLI endpoints keep working, which is easy to misdiagnose.

Deleting the file also stops the program from creating a `speedtest.db` in your working directory, and it removes the file as a thing to get wrong: with nothing to load, the defaults are used deterministically. Editing the file instead of deleting it is not a reliable substitute: depending on how the program is invoked, a partial or hand-edited config is either rejected with a parse error or silently ignored in favor of defaults. With no config file present, startup logs `Configs initialized with defaults` and `Config default assets directory.`, and / returns 200.

Generate the TLS certificate

The certificate's SAN must cover the exact address the client connects to, and it must cover it as the right kind of entry: a DNS name needs a `DNS:` entry, and a bare IP address needs an `IP Address:` entry.

Derive the addresses from the instance metadata service (IMDSv2):

```
TOKEN=$(curl -sS -X PUT "http://169.254.169.254/latest/api/token" \
  -H "X-aws-ec2-metadata-token-ttl-seconds: 300")

# -f is essential. See the following warning.
IMDS() { curl -fsS -H "X-aws-ec2-metadata-token: $TOKEN" \
  "http://169.254.169.254/latest/meta-data/$1" 2>/dev/null; }

PUB_DNS=$(IMDS public-hostname)
PUB_IP=$(IMDS public-ipv4)
PRIV_DNS=$(IMDS local-hostname)
PRIV_IP=$(IMDS local-ipv4)
printf 'PUB_DNS=[%s]\nPUB_IP=[%s]\nPRIV_DNS=[%s]\nPRIV_IP=[%s]\n' \
  "$PUB_DNS" "$PUB_IP" "$PRIV_DNS" "$PRIV_IP"
```

Important

Use `curl -fsS`, not `curl -sS`. On an instance with no public address, the `public-hostname` and `public-ipv4` metadata paths return a 404 HTML error page with exit status 0. Without `-f`, `curl` prints that page to stdout and succeeds, so PUB_DNS and PUB_IP fill with an XHTML document instead of being empty, and `openssl` then fails and writes no certificate. With `-f`, the same request fails cleanly and the variables are left empty, which is correct and expected. If those two variables contain HTML, you dropped the `-f`.

```
cd ~/bwtest && mkdir -p certs

SAN="DNS:${PRIV_DNS},IP:${PRIV_IP}"
[ -n "$PUB_IP" ] && SAN="IP:${PUB_IP},${SAN}"
[ -n "$PUB_DNS" ] && SAN="DNS:${PUB_DNS},${SAN}"
CN=${PUB_DNS:-$PRIV_DNS}
echo "CN=$CN"; echo "SAN=$SAN"

openssl req -x509 -newkey rsa:2048 -nodes -sha256 -days "$CERT_DAYS" \
  -keyout certs/server.key -out certs/server.crt \
  -subj "/CN=${CN}/O=DRS-Bandwidth-Test/OU=throwaway" \
  -addext "subjectAltName=${SAN}" \
  -addext "basicConstraints=critical,CA:TRUE" \
  -addext "keyUsage=digitalSignature,keyEncipherment,keyCertSign" \
  -addext "extendedKeyUsage=serverAuth"

chmod 600 certs/server.key
chmod 644 certs/server.crt
openssl x509 -in certs/server.crt -noout -subject -dates -ext subjectAltName
```

`basicConstraints=critical,CA:TRUE` is deliberate: it makes this one certificate act as its own trust anchor, which is what the CLI uses to run with full verification in the command-line test instead of disabling it. Add `,DNS:localhost,IP:127.0.0.1` to SAN if you also want to test from the instance itself.

> **Note**
>
> The certificate expires after CERT_DAYS. The rig should be torn down long before that. If it is still up when the certificate expires, verification starts failing, and the correct response is teardown, not a new certificate.

Start the endpoint

```
cd ~/bwtest
nohup ./librespeed-rs -p "$PORT" --enable-tls true \
  --tls-cert-file "$HOME/bwtest/certs/server.crt" \
  --tls-key-file  "$HOME/bwtest/certs/server.key" > ~/bwtest/server.log 2>&1 &
```

Starting it with nohup ... & keeps it in the same shell you used for the earlier steps, so the PORT and certificate variables are still in scope, and the endpoint keeps running after you log out. Read its output from the log file rather than a live foreground:

```
sleep 1 && cat ~/bwtest/server.log
```

The log opens with an ASCII-art banner. Each line after it is timestamped and prefixed `INFO librespeed_rs`, so it will not match the following literally:

```
Config default assets directory.
Configs initialized with defaults
Database disabled
Server started on 0.0.0.0:1500
Server base url : /backend/
Server TLS successfully configured
```

Verify TLS with a cleartext probe that must fail

> **Important**
>
> Do not confirm TLS by reading the log. The log misleads in both directions: it prints `Server started` before dying on a bad certificate, and it prints a healthy log while serving plaintext. If the certificate flags are supplied but `--enable-tls` is omitted, the

endpoint serves plaintext HTTP 200 with a working page and no error output, and the throughput number it produces is inflated because no encryption was performed.

The only reliable check is a negative one. Run both of these from the instance:

```
# NEGATIVE control - cleartext on the TLS port. This MUST FAIL.
curl -sS -m 5 "http://127.0.0.1:${PORT}/" -o /dev/null -w 'http_rc=%{http_code}\n';
 echo "curl_rc=$?"

# POSITIVE control - TLS on the same port. This MUST return 200.
curl -sSk -m 5 "https://127.0.0.1:${PORT}/" -o /dev/null -w 'http_rc=%{http_code}\n';
 echo "curl_rc=$?"
```

The following table lists the verdict for each pair of results:

Negative probe	Positive probe	Verdict
Non-zero `curl_rc`, no page	`http_rc=200`	TLS is on. Continue.
`http_rc=200`	Non-zero `curl_rc`	TLS is off, and you are about to measure plaintext. Stop, check `--enable-tls true` , and restart.
Non-zero `curl_rc`, no page	Non-zero `curl_rc`, no page	Both probes failed, so this is not a TLS pass. Nothing is listening on the port you probed, which happens when PORT is unset in this shell and both probes went to a port with no server. The negative probe failing here does not confirm TLS. Confirm PORT is set to 1500, confirm the endpoint is running, and run both probes again.

Against a correctly configured endpoint, the negative probe returns immediately rather than burning its five-second timeout, reporting `http_rc=000` and `curl_rc=1`. For a second confirmation, the following command reports `wrong version number` against a plaintext endpoint:

```
openssl s_client -connect "127.0.0.1:${PORT}" -brief </dev/null
```

Expect `Protocol version: TLSv1.3` and `Verification error: self-signed certificate`. The verification error is the expected shape here; the trust path is established on the client side in the section called “Copy the certificate to the client”, not in this probe. You might also see `Can't use SSL_get_servername`: this message is benign, because it only reports that no Server Name Indication (SNI) value was sent on this probe, and it does not affect the result.

Test with the command line

In this test you can set the stream count to match what Elastic Disaster Recovery actually does. Except where a step says otherwise, run this task on the source server, against the endpoint from the section called “Install the endpoint software”.

> **Important**
>
> Use five concurrent streams. The Elastic Disaster Recovery agent's replication channel is exactly five TCP sockets to one replication server. Five streams reproduces that socket count. A single-stream test measures something Elastic Disaster Recovery never does and can report a fast link as much slower than it is.
> Five streams are not equivalent to what the agent does with them. The agent sends one ordered stream and rotates successive chunks across the five sockets from a single sender, and the receiver reassembles them in strict order, so a socket that stalls stalls the whole channel. LibreSpeed runs five independent streams that cannot block one another. Treat the result as the path's upper limit. See the section called “Interpreting the result”.

> **Important**
>
> Five parallel streams can saturate the source server's uplink for about a minute. On a production source server, that can slow business traffic and raise replication lag on other servers that replicate over the same link. Run the test off-peak, coordinate with whoever operates the network, and test one source server at a time.

Install the client

The client is librespeed/speedtest-cli, the LibreSpeed command-line client.

```
export LIBRESPEED_CLI_VERSION=1.0.14
export CLI_ARCH=linux_amd64                 # see the release page for other platforms
export PORT=1500                            # must match the port the endpoint listens
 on
export STREAMS=5                            # matches the DRS agent; do not change
 without reading this task
export DURATION=30                          # seconds per direction

mkdir -p ~/bwclient && cd ~/bwclient
curl -sSLO "https://github.com/librespeed/speedtest-cli/releases/download/v
${LIBRESPEED_CLI_VERSION}/librespeed-cli_${LIBRESPEED_CLI_VERSION}_${CLI_ARCH}.tar.gz"
```

Note

The project's GitHub release page lists a SHA-256 digest beside each archive. The digest covers the archive, not the extracted binary, so hash the archive and compare it with that digest before you unpack it:

```
sha256sum "librespeed-cli_${LIBRESPEED_CLI_VERSION}_${CLI_ARCH}.tar.gz"
```

```
tar -xzf "librespeed-cli_${LIBRESPEED_CLI_VERSION}_${CLI_ARCH}.tar.gz" librespeed-cli
chmod +x librespeed-cli
./librespeed-cli --version
```

Note

The release page carries builds for other operating systems and architectures. The Windows builds are `.zip` archives, such as `librespeed-cli_1.0.14_windows_amd64.zip`, that contain `librespeed-cli.exe`. For Windows, see Install on Windows in the project's README. The commands on this page are for a Linux shell.

Note

The published Linux client binaries for x86 and Arm are UPX-compressed, which some antivirus and endpoint detection and response (EDR) tools flag or quarantine as a heuristic. If that happens on a production source server, the download is not corrupt: add an

exception for the binary, or run the client from a non-production host on the same network path.

Copy the certificate to the client

Prefer real certificate verification over a bypass: copy the endpoint's certificate to the client and pass it as the trust anchor. This is the better option because the certificate is its own CA (`CA:TRUE` in the section called “Generate the TLS certificate”), so verification genuinely checks that you are talking to the endpoint you built. A bypass checks nothing and also hides a TLS-intercepting middlebox, which is one of the things this test exists to detect.

Display the certificate on the endpoint instance, and copy its full output:

```
cat ~/bwtest/certs/server.crt
```

On the source server, write the copied certificate to `~/bwclient/target-ca.crt` with the following command. Paste the certificate, including the `-----BEGIN CERTIFICATE-----` and `-----END CERTIFICATE-----` lines, then press `Ctrl+D` on an empty line to end the input:

```
cat > ~/bwclient/target-ca.crt
```

The certificate is public material, so it is safe to display and paste; only the private key is secret, and it never leaves the instance. The paste travels over the shell session you already use, not over the measured TLS path on port 1500, so a TLS-intercepting middlebox on that path cannot substitute it. A paste error makes verification fail rather than pass silently.

Note

If you genuinely cannot move the file, for example on an air-gapped source server with no SSH and no Session Manager path, substitute `--skip-cert-verify` for `--ca-cert` in the section called “Run the measurement”. The client then accepts any certificate, including one presented by something intercepting the connection, and TLS interception is one of the conditions this test is meant to expose.

Write the server list

The client takes its target from a small JSON file. Field names are case-sensitive: note `dlURL`, `ulURL`, and `pingURL`, but `getIpURL` with a lowercase p.

```
ENDPOINT_HOST=ENDPOINT_NAME        # the CN= value the certificate step printed; over
 VPN or Direct Connect, its PRIV_IP value instead

cat > ~/bwclient/servers-1500.json <<JSON
[
  {
    "id": 1,
    "name": "DRS bandwidth test endpoint (TLS :${PORT})",
    "server": "https://${ENDPOINT_HOST}:${PORT}/backend/",
    "dlURL": "garbage",
    "ulURL": "empty",
    "pingURL": "empty",
    "getIpURL": "getIP"
  }
]
JSON

cat ~/bwclient/servers-1500.json
```

> **Important**
>
> The trailing `/backend/` on `server` is required. This endpoint serves its throughput routes under `backend/`, not at the root, and it has no `.php` route aliases. Without the suffix, every endpoint returns 404 and the client aborts at its pre-flight check, reporting that the server is not responding, before taking any measurement.

The host in `server` must match a SAN entry of the same kind: a DNS name must match a `DNS:` entry, and a bare IP address must match an `IP Address:` entry. In the section called "Generate the TLS certificate" on the endpoint, the `echo "CN=$CN"` line printed a value; set `ENDPOINT_HOST` on this source server to that same value and the kinds match automatically.

The host must also match the path you intend to measure. A source server on-premises over VPN or Direct Connect has to use the endpoint's private IP literal, the `PRIV_IP` value from the same step, which the SAN lists as an `IP Address:` entry. On-premises DNS usually cannot resolve the private DNS name. The public DNS name resolves to the public address, which bypasses the tunnel.

An in-VPC client measuring the internet path has to use the endpoint's public IP literal here, because the public DNS name resolves to the private address from inside the VPC and the traffic would never leave.

Run the measurement

```
cd ~/bwclient
./librespeed-cli \
  --local-json servers-1500.json \
  --server 1 \
  --concurrent "$STREAMS" \
  --duration "$DURATION" \
  --ca-cert target-ca.crt \
  --no-icmp
```

- `--concurrent 5` matches the Elastic Disaster Recovery agent's five replication sockets. Without this flag, the client uses three.
- `--duration 30` sets the length of each direction, not of the whole run. Expect about 62 seconds of wall-clock time: 30 seconds of download, then 30 of upload, plus pre-flight.
- `--no-icmp` skips the ICMP ping probe, which many networks block regardless of TCP reachability. The client can still print a `Ping` line measured over HTTP, so a reported ping value does not mean ICMP was used.

Check that the client reports a client IP that is the source server's address, which proves the request really originated off-instance. Verification succeeded if the run completed without a certificate error and without `--skip-cert-verify`.

Run the tests one at a time unless you specifically want to measure contention.

> **Note**
>
> Cross-Availability Zone paths add latency and are metered differently from same-Zone paths, so a figure measured within one Availability Zone should not be assumed to carry over to a cross-Zone path. See the section called “Interpreting the result”.

Test with the browser

This test is an alternative to the command-line test in the section called "Test with the command line". The endpoint serves its own single-page test UI at `/`. Use the browser instead of the command-line client when you want a quick interactive result from a source server that has a desktop, or when you want a person to watch the test run.

> **Note**
>
> The browser is a convenience surface. It does not let you set the number of concurrent streams, which is why the command-line client in the section called "Test with the command line" produces the figure you should quote. Against a fast private path, the browser can understate the CLI figure substantially. Do not report a browser number as your link speed. It is, however, a useful second instrument; see the section called "Optionally compare the browser and the CLI to find the bottleneck".

Open the page from the source server

```
https://PRIV_DNS:1500/          # source server in the same VPC
https://PUB_DNS:1500/           # source server on the public internet
```

Use whichever name or address you put in the certificate SAN in the section called "Generate the TLS certificate". Pick the name that matches the path you want to measure, not only the certificate. From inside the VPC, a public DNS name resolves to the instance's private IP, so an in-VPC client that uses *PUB_DNS* stays on the private path. An in-VPC client that must cross the internet gateway has to use the public IP literal. A source server over VPN or Direct Connect has to use `https://PRIV_IP:1500/`, because on-premises DNS usually cannot resolve the private DNS name.

Accept the certificate warning

The certificate is self-signed, so every browser shows an interstitial warning. Chrome reports NET::ERR_CERT_AUTHORITY_INVALID under "Your connection is not private", and Firefox reports "Warning: Potential Security Risk Ahead". Choose the advanced or details option and proceed (in Chrome, **Advanced**, **Proceed**; in Firefox, **Advanced**, **Accept the Risk and Continue**; in Safari, **Show Details**, **visit this website**).

> **Important**
>
> Accepting this warning is a deliberate bypass of certificate validation. It is acceptable here because the endpoint is a throwaway instance you launched minutes ago and the test carries no sensitive data. Confirm that the hostname in the warning is your own instance before you proceed. You should not see a name-mismatch complaint; if you do, the SAN in the section called "Generate the TLS certificate" does not cover the name you connected by.

Run the test

Choose the start button on the page and let it run to completion. It reports download, upload, ping, and jitter, and it shows the address it saw your request arrive from, which is a useful confirmation that the traffic really came from where you think. The run takes about 18 seconds, fixed by the page's own configuration and not operator-settable.

Tear down the test rig

As soon as you have your numbers, delete everything that you created for the test, such as the endpoint instance and the inbound TCP port 1500 rule.

Interpreting the result

Before you quote a figure, be clear about what it represents and what it does not.

What the figure represents, and what it does not

> **Important**
>
> If a proxy sits between this source server and AWS, treat the result as provisional. The test client might be routed through the proxy while the Elastic Disaster Recovery agent's replication connection is not, so a passing result does not confirm that replication can reach the replication server over that path.

- The figure is the upper limit of the path, not the replication rate to expect. The Elastic Disaster Recovery agent compresses replicated data with LZ4 before sending it, so a compressible workload can move more disk bytes per second than this figure suggests. At the same time, the agent sends one ordered stream that the receiver reassembles in strict order across the five

sockets, so a single stalled socket stalls the whole channel and replication usually runs below this figure.

- The replication server is itself a limit, and it is not the instance type of your test endpoint. By default the replication server is a `t3.small`. This test measures the path to an endpoint whose size you chose.
- A short run on a small instance can briefly burst past what the path sustains, so do not quote it as a link speed.
- A pass here covers only TCP port 1500. Replication also needs outbound TCP port 443 to the Elastic Disaster Recovery service endpoint and to Amazon S3, which this test does not exercise. For information about verifying port 443, see the section called “Verifying network connectivity”.

Optionally compare the browser and the CLI to find the bottleneck

If you did run both the command-line test and the browser against the same endpoint, comparing them is inexpensive and tells you which way to look.

- If they disagree materially, the instrument is the limit, not the path. Against a fast private endpoint, the browser can report figures well below the CLI, for example roughly 37% low on download and 45% low on upload. Quote the CLI figure.
- If they agree closely, something in the path is limiting both. Two independent implementations converging on the same number is the signature of a shaped or saturated path, and that number is the one to investigate.

Do not compare headline figures between runs

A genuine allowance problem shows throughput declining within a single run, not between runs. Comparing one run's headline number against another's is unreliable: back-to-back 30-second runs on an unchanged idle path can decline by 20% or more and then recover on the next run.

To look for a real decline, watch one long run's own time series. On the endpoint, during the run, use the Linux `sar -n DEV 1` command, or read `/proc/net/dev` in a one-second loop:

```
sar -n DEV 1
```

Compare the first third of a phase against the last third. A decline that persists inside a single run is real; a difference between two runs' headline numbers is noise.

Costs

This procedure has two cost line items, and the second is the larger one.

- The instance-hour is minor for the lifetime of a test, and an in-use public IPv4 address adds a small hourly charge on top of it.
- Data transfer out dominates. Each measurement run moves real data (a 30-second download phase at 5 Gbps is roughly 18 GB), and data transfer out to the internet is billed per GB beyond the monthly free allowance. Traffic that stays inside the VPC is billed differently and is generally far cheaper, which is one more argument for the private path.

Check current rates for your Region on the AWS Pricing and Amazon EC2 On-Demand pricing pages rather than relying on the figures quoted here.

Troubleshooting the bandwidth test

The connection is refused, or the page will not load

For information about general port 1500 reachability, including firewalls and network ACLs, see the section called “Verifying network connectivity”. Then check the following, which are specific to this test endpoint:

1. Confirm the source address has not changed. An internet egress address can rotate. On the testing machine, compare `curl -sS https://checkip.amazonaws.com` against the address in your security group.
2. Confirm you are using the right kind of address for your case. A rule naming a public address does not admit the same host's private address, and the reverse. An in-VPC client reaching the endpoint's public IP hairpins through the internet gateway and arrives bearing its public source address, so a rule that allows only its private `/32` refuses it.
3. Confirm the rule is the shape you think it is: a `tcp` rule from port `1500` to port `1500` for only the source addresses you intended, with no rule on any other port and no unexpected IPv6, security-group-reference, or prefix-list entry. Any of those can admit traffic you did not intend, and an allow-all IPv6 rule in particular can render as an empty list that looks harmless.
4. Confirm the endpoint process is still running. It does not survive a reboot, and nothing relaunches it. Restart it from the section called “Start the endpoint”.
5. Confirm the instance has not been isolated. If its security group is no longer the one you created, see the section called “Security considerations for the test endpoint”.

The client reports that the server is not responding

The client prints the following message and then exits with status 0:

```
Selected server name (host) is not responding at the moment, try again later
```

A zero exit code from this client does not mean the test succeeded, so read the output rather than trusting the exit code. The client reports the same "not responding" message for several distinct failures, including a malformed server URL and a rejected certificate. Check these causes:

1. The port in the `server` URL. If `PORT` was empty when you wrote the server list, the URL contains an empty port (`https://host:/backend/`) and the client cannot connect. Confirm `PORT` is set to 1500 in the shell where you run the command-line test, then write the server list again. See the section called “Write the server list”.
2. The `/backend/` suffix on the `server` URL. Without it, every route returns 404 and the client reports the server as not responding. See the section called “Write the server list”.
3. The security group. Confirm it admits this source server's address on TCP port 1500. On a newly launched endpoint, the endpoint's public DNS name might resolve to its public address rather than the private one, so a rule that allows only the source server's private address refuses the connection. In one observed case, a rerun a minute later passed. Check what `ENDPOINT_HOST` resolves to with `getent hosts "$ENDPOINT_HOST"`, and retry.
4. Certificate verification. A certificate the client rejects, because of a name that is not in the SAN or a trust anchor that does not match, produces the same message and the same exit code 0. Check the certificate SAN against the exact host string in the server list, and confirm you copied the right certificate. See the section called “Copy the certificate to the client”.

The browser page returns 404

The `configs.toml` file was not deleted. Return to the section called “Delete the shipped config file”. The CLI endpoints keep working in this state, which makes it easy to misdiagnose.

Verification fails with a name mismatch

The name you connected by is not in the certificate SAN as the right kind of entry, or the instance was stopped and started and its public address changed. Re-run the section called “Generate the TLS certificate” against the current addresses.

Troubleshooting launch errors

Use this section to troubleshoot issues that occur when launching drill or recovery instances.

If you see...	See
"Conversion failed", "Failed to connect using HTTP channel", or snapshot errors	Conversion errors
Lifecycle state errors, BYOL conflicts, EBS key issues, IAM permission errors, or instance store conflicts	Configuration errors
Windows license activation failures, drive letter changes, or Dynamic Disk showing as Foreign	Windows post-launch issues

Topics

- Launch conversion errors
- Launch configuration errors
- Windows post-launch issues

Launch conversion errors

The following errors can occur during the conversion phase when you launch a drill or recovery instance in AWS Elastic Disaster Recovery (Elastic Disaster Recovery). The conversion phase prepares your replicated volumes for use in the target AWS environment.

Topics

- Error: Failed to connect using HTTP channel
- Error: Conversion server launch failed
- Error: Conversion failed
- Error: Failed to take snapshot

Error: Failed to connect using HTTP channel

Error message: Failed to connect using HTTP channel

Cause: The Conversion server cannot communicate with AWS endpoints on TCP port 443. The Conversion server is a separate instance from the replication server and might be affected by network changes made after replication was established.

Resolution:

Console

To resolve using the console

1. Open the Amazon VPC console and verify the staging area subnet's route table allows outbound traffic on TCP port 443.
2. Verify that the security group associated with the staging area subnet allows outbound TCP port 443 traffic.
3. Verify that the network ACL for the staging area subnet allows outbound TCP port 443 traffic.
4. Check for recent network changes that might affect connectivity from the staging area to AWS service endpoints.

CLI

To resolve using the CLI

1. Test connectivity from the staging area to the Elastic Disaster Recovery, Amazon S3, and Amazon EC2 endpoints. Run one of the following commands depending on your operating system:
   - Linux:

     ```
     curl -v https://drs.region.amazonaws.com
     ```

   - Windows:

     ```
     Test-NetConnection -ComputerName drs.region.amazonaws.com -Port 443
     ```

2. Verify the route tables for the staging area subnet:

   ```
   aws ec2 describe-route-tables \
       --filters "Name=association.subnet-id,Values=subnet-id"
   ```

3. Confirm that a route to `0.0.0.0/0` exists and points to an internet gateway, NAT gateway, or VPC endpoint that allows traffic to AWS service endpoints on port 443.

If the issue persists after verifying network configuration, create an AWS Support case with details about your staging area network configuration.

Error: Conversion server launch failed

Error message: Conversion server launch failed

Cause: The conversion server could not be launched or did not become available. This can be caused by Amazon EC2 capacity constraints, IAM permission issues, or subnet configuration problems.

Resolution:

To resolve this error

1. Retry the launch from the Elastic Disaster Recovery console.
2. If the error persists, create an AWS Support case. Include the recovery job ID, which you can find in the Elastic Disaster Recovery console.

Error: Conversion failed

Error message: Conversion failed

Cause: The volume conversion process did not complete. This can occur due to disk corruption on the source, unsupported filesystem types, or internal conversion errors.

Resolution:

To resolve this error

1. Retry the launch from the Elastic Disaster Recovery console.
2. If the error persists, create an AWS Support case. Include the recovery job ID from the Elastic Disaster Recovery console.

Error: Failed to take snapshot

Error message: Failed to take snapshot

Cause: Elastic Disaster Recovery could not create a point-in-time snapshot. If you use a custom AWS KMS key for EBS encryption, the key might be missing, disabled, or the Elastic Disaster Recovery service roles might lack permissions.

Resolution:

To resolve this error

1. Verify that the AWS KMS key exists and is enabled.
2. Verify that the Elastic Disaster Recovery service roles have `kms:CreateGrant` and `kms:DescribeKey` permissions on the key.
3. Retry the launch from the Elastic Disaster Recovery console.
4. If the error persists, create an AWS Support case. Include the recovery job ID from the Elastic Disaster Recovery console.

Launch configuration errors

The following errors occur when launch configuration settings, IAM permissions, or resource limits prevent AWS Elastic Disaster Recovery from launching recovery instances.

Topics

- Error: Instance not launched due to lifecycle state
- Error: OS BYOL requires Dedicated Hosts
- Error: EBS encryption key not found
- Error: Missing IAM permissions for launch
- Error: Instance store volume device name conflict
- Error: Basic right-sizing is not supported for arm64
- Error: In-AWS right-sizing cannot describe the source instance
- Error: Source instance not found for in-AWS right-sizing
- Error: Source instance type is unavailable in the recovery Region
- Error: Launch template specifies an x86_64 instance type for arm64
- Error: Instance requirements exclude arm64 instance types
- Error: AMI architecture does not match the arm64 source
- Error: Invalid arm64 recovery settings

- Error: Launch-into instance architecture does not match

Error: Instance not launched due to lifecycle state

Error message

instance not launched because server lifecycle state is not READY_FOR_TEST

Cause

The source server has not completed initial sync or is not in the correct lifecycle state for the requested operation.

Resolution

To resolve this error, complete the following steps:

1. Verify that the source server is in the *Ready for recovery* state. For recovery drills, the server must have completed initial sync.
2. Check the data replication status in the AWS Elastic Disaster Recovery console.
3. If the server is in a *Stalled* or *Disconnected* state, resolve the replication issue before attempting to launch.

Error: OS BYOL requires Dedicated Hosts

Error message

OS BYOL can only be used with EC2 Dedicated Hosts

Cause

Bring Your Own License (BYOL) is enabled in the launch settings, but the Amazon EC2 Launch Template is not configured to use a Dedicated Host.

Resolution

Use one of the following options to resolve this error:

- Configure the Amazon EC2 Launch Template to use a Dedicated Host.
- Disable BYOL in the AWS Elastic Disaster Recovery launch settings for the source server.

Error: EBS encryption key not found

Error message

The EBS encryption key could not be found in this account

Cause

The AWS KMS key specified in the replication settings does not exist or is not accessible from the target account.

Resolution

To resolve this error, verify the following:

- The KMS key ARN in the replication settings is correct.
- The key has not been deleted or disabled.
- The AWS Elastic Disaster Recovery service roles have `kms:CreateGrant` and `kms:DescribeKey` permissions on the key.

Error: Missing IAM permissions for launch

Error message

Your IAM user do not have permission for ec2:CreateSecurityGroup

Cause

The IAM credentials used for the launch operation lack required permissions.

Resolution

Verify that the required AWS Elastic Disaster Recovery IAM policies are attached to your user or role. For more information, see Identity-based policies for AWS Elastic Disaster Recovery.

Error: Instance store volume device name conflict

Error message

Launch fails with device name conflicts when the source server has instance store volumes.

Cause

The Amazon EC2 Launch Template specifies instance store volumes that collide with device names used by AWS Elastic Disaster Recovery for replicated EBS volumes.

Resolution

Use one of the following options to resolve this error:

- **If you need the instance store data** – Change the device name in the Amazon EC2 Launch Template to avoid the collision. For example, use `/dev/xvdc1`.
- **If you don't need instance store data** – Exclude instance store volumes from replication by using the `--devices` installation parameter. AWS Elastic Disaster Recovery does not populate excluded volumes in the Launch Template. For more information, see installation parameters.

Error: Basic right-sizing is not supported for arm64

Error message

Cannot set BASIC right sizing on source server *`source-server-id`*. BASIC right sizing is not supported for arm64/Graviton source servers. Use IN_AWS or NONE instead.

Cause

You explicitly set **Active (basic)** right-sizing on an `arm64` source server.

Resolution

Set right-sizing to **Active (in-aws)** or **Inactive**. If the account default launch settings specify **Active (basic)** right-sizing, AWS Elastic Disaster Recovery applies **Active (in-aws)** right-sizing when it creates an `arm64` source server.

Error: In-AWS right-sizing cannot describe the source instance

Error message

Could not apply in-AWS right sizing for source server *`source-server-id`* because AWS Elastic Disaster Recovery is unable to successfully make the DescribeInstances API call against the source EC2 instance *`instance-id`*.

Cause

AWS Elastic Disaster Recovery does not have permission to call `DescribeInstances` for the source Amazon EC2 instance. This error appears only when you change the right-sizing method for an `arm64` source server. Otherwise Elastic Disaster Recovery uses its recommended AWS Graviton instance type.

Resolution

Restore permission to describe the source instance, then retry the launch-settings update. For source servers replicated from another account, create the failback and in-AWS right-sizing roles in the source account. See Creating the Failback and in-AWS right-sizing roles.

Error: Source instance not found for in-AWS right-sizing

Error message

Could not apply in-AWS right sizing for source server *`source-server-id`* because source instance *`instance-id`* was not found in its reported account and region.

Cause

The source Amazon EC2 instance no longer exists in the reported account and Region, or the recorded instance identity is incorrect. For when Elastic Disaster Recovery checks the source instance, see Error: In-AWS right-sizing cannot describe the source instance.

Resolution

Verify the source account, Region, and instance ID. Restore the source instance or reinstall the agent on the correct instance, then retry.

Error: Source instance type is unavailable in the recovery Region

Error message

Could not apply in-AWS right sizing for source server *`source-server-id`* because source instance type *`instance-type`* is not available in the recovery region.

Cause

The source Graviton instance type is not offered in the recovery Region. Elastic Disaster Recovery runs this check only in the situations described in Error: In-AWS right-sizing cannot describe the source instance.

Resolution

Use **Inactive** right-sizing and configure a compatible Graviton type, or keep the right-sizing method unchanged so that Elastic Disaster Recovery selects a Graviton recommendation that is offered in the recovery Region.

Error: Launch template specifies an x86_64 instance type for arm64

Error message

Source server *source-server-id* is an arm64 (Graviton) source, but the launch template specifies instance type '*instance-type*', which is not an arm64 instance type. An aarch64 root volume cannot boot on an x86_64 instance. Use a Graviton instance type (such as c6g, m6g, r6g, c7g, or t4g) in your launch template.

Cause

The launch template pins an `x86_64` instance type for an `arm64` source server.

Resolution

Select a Graviton instance type that is offered in the recovery location.

Error: Instance requirements exclude arm64 instance types

Error message

Source server *source-server-id* is an arm64 (Graviton) source, but the launch template's InstanceRequirements cannot be satisfied by any arm64 instance type. An aarch64 root volume cannot boot on an x86_64 instance. Widen AllowedInstanceTypes to include a Graviton family (for example c6g.*, m6g.*, r6g.*, c7g.*, or t4g.*), or relax the attribute constraints so at least one arm64 instance type qualifies.

Cause

The allowed, excluded, or attribute requirements match no `arm64` instance type offered in the recovery Region.

Resolution

Widen the allowed instance types or relax the requirements so that they match at least one Graviton type offered in the recovery Region.

Error: AMI architecture does not match the arm64 source

Error message

Source server *source-server-id* is an arm64 (Graviton) source, but the AMI '*ami-id*' specified for launch has architecture '*ami-architecture*'. An aarch64 root volume cannot boot from an x86_64 AMI. Use an arm64 AMI, or remove the AMI override to let DRS create an architecture-matched AMI.

Cause

The launch template or usage operation specifies an `x86_64` Amazon Machine Image (AMI) for an `arm64` source server.

Resolution

Use an `arm64` AMI, or remove the AMI override so that AWS Elastic Disaster Recovery creates an architecture-matched AMI.

Error: Invalid arm64 recovery settings

Error message

Action could not be completed. Invalid ARM64 recovery settings: *violations* in the launch configuration template.

Cause

The launch template contains one or more of these violations:

- one of InstanceType or InstanceRequirements is required
- instance type '*instance-type*' is not an arm64 (Graviton) type

At launch time, the messages described in Error: AMI architecture does not match the arm64 source and Error: Instance requirements exclude arm64 instance types can also appear inside this error.

Resolution

Specify a Graviton instance type or satisfiable instance requirements in the launch template.

Error: Launch-into instance architecture does not match

Error message

You cannot define the EC2 instance *ec2-instance-id* as a launch into instance ID as is has conflicting instance settings to those of the source server (*source-server-id*). The following fields have a configuration mismatch: Architecture.

Cause

The target instance does not use the same architecture as the protected server.

Resolution

Select a stopped target Amazon EC2 instance that uses the same architecture as the protected server and meets the other launch-into requirements.

Windows post-launch issues

This topic covers Windows-specific issues that can occur after you launch drill or recovery instances with AWS Elastic Disaster Recovery.

Topics

- Windows license activation failure
- Windows drive letter reassignment
- Windows Dynamic Disk shows as Foreign

Windows license activation failure

Cause: AWS Elastic Disaster Recovery converts Windows OS licenses to AWS licenses and activates them against AWS KMS. Activation can fail if the recovery instance cannot reach the AWS KMS endpoint.

Resolution: Follow the AWS guide to resolve this issue: Troubleshooting Windows activation.

> **Important**
>
> During failback, AWS Elastic Disaster Recovery does not have access to customer licenses and cannot activate them. After failback completes, activate licenses manually or by using post-launch scripts.

Windows drive letter reassignment

Symptom: Drive letters change after launch (for example, D: becomes E:).

Cause: Windows reconfigures drive letter assignments when a machine starts on new infrastructure, especially if the source server had drive letters mapped to disks that were not replicated (such as network drives or excluded volumes).

Resolution: Remap the drive letters on the recovery instance after launch by using **Disk Management** (diskmgmt.msc) or diskpart. This is a one-time post-launch step.

Windows Dynamic Disk shows as Foreign

Symptom: After launch, a Dynamic Disk shows status `Foreign` in **Disk Management**.

Cause: Moving a Windows Dynamic Disk between computers changes the disk status to Foreign. This is normal Windows behavior when disks are presented to a different machine.

Resolution: Import the foreign disk by using **Disk Management**:

1. Open diskmgmt.msc.
2. Right-click the disk showing `Foreign` and choose **Import Foreign Disks**.

For more information, see Microsoft Dynamic Disk troubleshooting.

Troubleshooting failback errors

Use this section to troubleshoot issues that occur during the failback process — from Failback Client setup through reverse replication.

If you see...	See
"Could not associate failback client", "Failback client not seen", credential errors, or connectivity errors during failback setup	Failback Client errors
"Agent is not connected to DRS", credential retrieval errors (clock skew), or "Recovery instances could not be processed"	Failback replication errors

Topics

- Failback Client errors
- Failback replication errors

Failback Client errors

The following topics cover errors that you might encounter when you set up or connect the Failback Client in AWS Elastic Disaster Recovery (Elastic Disaster Recovery).

Topics

- Error: Could not associate failback client to recovery instances
- Error: Could not discover account ID
- Error: Failed to get recovery instance volumes
- Error: Failback client not seen
- Error: Could not verify recovery instance connectivity to DRS

Error: Could not associate failback client to recovery instances

Error message: Could not associate failback client to recovery instances

Cause: The IAM credentials used for the Failback Client do not have the required failback policy attached.

Resolution:

To resolve this error

1. Attach the `AWSElasticDisasterRecoveryFailbackInstallationPolicy` managed policy to the IAM user or role.
2. Restart the failback process.

For more information, see Learn more about Failback Client credentials.

Error: Could not discover account ID

Error message: Could not discover account id from describe

Cause: The AWS credentials or Region entered into the Failback Client are incorrect.

Resolution:

To resolve this error

1. Verify that the AWS Access Key ID and Secret Access Key are correct and active.
2. Verify that the Region matches the Region where Elastic Disaster Recovery is configured.
3. Test the credentials by running the following command:

```
aws sts get-caller-identity --region region
```

Error: Failed to get recovery instance volumes

Error message: Failed to get recovery instance volumes, please check the network configuration of your recovery instance

Cause: The Failback Client cannot communicate with the recovery instance on TCP port 1500.

Resolution:

To resolve this error

1. Verify that TCP port 1500 is open between the failback server and the recovery instance.
2. If you are not using a private route (VPN or), verify that the recovery instance has a public IP address.
3. Test connectivity to the recovery instance on port 1500:
   - **Linux:**

     ```
     nc -zv recovery-instance-ip 1500
     ```

   - **Windows (PowerShell):**

     ```
     Test-NetConnection -ComputerName recovery-instance-ip -Port 1500
     ```

Error: Failback client not seen

Error message: Failback client not seen status in the Elastic Disaster Recovery console.

Cause: Communication between the Failback Client and the Elastic Disaster Recovery endpoint has been interrupted. Common causes include:

- Network issues between the Failback Client and the Elastic Disaster Recovery endpoint (TCP 443)
- The Failback Client process was terminated or interrupted
- The Failback Client runs in the foreground — closing the shell session terminates the process

Resolution:

To resolve this error

1. Verify network connectivity to `drs.region.amazonaws.com` on port 443.
2. Use `screen` or `tmux` to keep the Failback Client session alive.
3. Restart the Failback Client if needed.

> **Note**
>
> Replication might continue in the background because the AWS Replication Agent runs independently of the Failback Client process.

Error: Could not verify recovery instance connectivity to DRS

Error message: Could not verify recovery instance connectivity to Elastic Disaster Recovery

Cause: The recovery instance cannot communicate with the AWS Elastic Disaster Recovery endpoint on TCP port 443.

Resolution:

Console

To resolve this error (console)

1. Verify that the recovery instance has a public IP address. If you use a VPN or , a public IP is not required.
2. Verify that the security group attached to the recovery instance allows outbound traffic on TCP port 443.

3. Check the agent logs for connectivity errors.

CLI

To resolve this error (CLI)

1. Verify the recovery instance network configuration:

```
aws ec2 describe-instances --instance-ids instance-id \
    --query "Reservations[*].Instances[*].[PublicIpAddress,SecurityGroups]"
```

2. Test connectivity from the recovery instance to the Elastic Disaster Recovery endpoint:
   - **Linux:**

     ```
     curl -v https://drs.region.amazonaws.com
     ```

   - **Windows (PowerShell):**

     ```
     Test-NetConnection -ComputerName drs.region.amazonaws.com -Port 443
     ```

3. Check the agent logs for errors:

```
tail /var/lib/aws-replication-agent/agent.log.0 | grep error
```

If the agent log shows a "driver compiled for different kernel" error, see Troubleshoot driver compilation errors.

Failback replication errors

The following errors can occur during reverse replication (failback data sync) in AWS Elastic Disaster Recovery. Each section describes the error message, its cause, and the resolution steps.

Topics

- Error: AWS Replication Agent is not connected to DRS
- Error: Credential retrieval failed (clock skew)
- Error: Some recovery instances could not be processed

Error: AWS Replication Agent is not connected to DRS

Error message

AWS Replication agent is not connected to DRS. Verify the agent is installed and running, and that it has connectivity to the service

Cause

After you initiate reverse replication, the agent on the recovery instance cannot communicate with the AWS Elastic Disaster Recovery service.

Resolution

To resolve this error, complete the following steps:

1. Verify that the agent is installed and running on the recovery instance. Run the following command for your operating system:
   - Linux:

     ```
     systemctl status aws-replication-agent
     ```

   - Windows:

     ```
     Get-Service AwsReplicationService
     ```

2. Verify that the recovery instance has internet connectivity or a NAT gateway connection.
3. If both the agent and connectivity are confirmed working, reinstall the agent as a recovery instance and retry reverse replication.

Error: Credential retrieval failed (clock skew)

Error message

botocore.exceptions.CredentialRetrievalError: Error when retrieving credentials from cert

Cause

The Failback Client uses Amazon Linux 2 and certificate-based authentication. Amazon Linux 2 assumes the hardware clock is set to UTC. If the BIOS or EFI clock is set to local time instead of UTC, authentication fails because of time skew.

Resolution

To resolve this error, complete the following steps:

1. Access the BIOS or EFI Shell of the failback target server.
2. Set the hardware clock to UTC (not local time).
3. Restart the Failback Client.

> **Note**
>
> This error commonly occurs when failing back to physical servers or hypervisors where the hardware clock was configured for local time.

Error: Some recovery instances could not be processed

Error message

Some Recovery instances could not be processed: *recovery-instance-id*

Cause

This error occurs during reverse replication when both of the following conditions are true:

- The **Launch into source instance** setting is enabled in the source Region's default launch settings.
- The source Amazon EC2 instance is missing the required tag `AWSDRS:AllowLaunchingIntoThisInstance`.

Resolution

To resolve this error, use one of the following options:

- **Option 1** – Disable the **Launch into source instance** setting in the source Region's default launch settings.
- **Option 2** – Add the tag `AWSDRS:AllowLaunchingIntoThisInstance` to the source Amazon EC2 instance.

Note

Review the impact before you disable **Launch into source instance**. This setting affects all future recovery launches for the account in this Region.

Elastic Disaster Recovery FAQ

What source infrastructure does AWS Elastic Disaster Recovery support?

With AWS Elastic Disaster Recovery, you can recover applications on AWS from physical infrastructure, virtual machines, and other clouds. These source servers must use the `x86_64` architecture. You must be able to install the AWS Replication Agent and run the Failback Client. AWS Elastic Disaster Recovery also supports 64-bit Arm (`arm64`) Linux source servers on AWS Graviton-based Amazon EC2 instances. Elastic Disaster Recovery supports `arm64` only for AWS-to-AWS recovery. See AWS DRS supported Linux operating systems.

Can I upgrade from CloudEndure Disaster Recovery to AWS Elastic Disaster Recovery?

CloudEndure Disaster Recovery was discontinued on September 29, 2025, in all AWS Regions except the China Regions. There is no upgrade path from CloudEndure Disaster Recovery to AWS Elastic Disaster Recovery. To protect your servers with Elastic Disaster Recovery, follow the getting started procedure.

Can AWS Elastic Disaster Recovery protect physical servers?

Because AWS Elastic Disaster Recovery works at the OS layer it can protect not only virtual servers but physical ones as well.

What data is stored on and transmitted through AWS Elastic Disaster Recovery servers?

AWS Elastic Disaster Recovery stores only configuration and log data on the AWS Elastic Disaster Recovery Console's encrypted database. Replicated data is always stored on the customer's own cloud VPC. The replicated data is encrypted in transit.

What is the Recovery Time Objective (RTO) of AWS Elastic Disaster Recovery?

The Recovery Time Objective (RTO) of Elastic Disaster Recovery is typically measured in minutes. The RTO is highly dependent on the OS boot time.

What is the Recovery Point Objective (RPO) of AWS Elastic Disaster Recovery?

The Recovery Point Objective (RPO) of AWS Elastic Disaster Recovery is typically in the sub-second range.

What to consider when replicating Active Directory

There are two main approaches when it comes to migrating Active Directory or domain controllers from a disaster:

1. Replicating the entire environment, including the AD server(s) - in this approach it is recommended to launch the drill or recovery AD servers first, wait until it's up and running and then launch the other drill or recovery instances, to make sure the AD servers are ready to authenticate them.
2. Leaving the AD server(s) in the Source environment - in this approach, the drill or recovery instances will communicate back to the AD server in the source environment and will take the source server's place in the AD automatically.

   In this case, it is important to conduct any drills using an isolated subnet in the AWS cloud, so to avoid having the drill or recovery instances communicate into the source AD server outside of a recovery.

Does AWS Elastic Disaster Recovery work with LVM and RAID configurations?

AWS Elastic Disaster Recovery works with any hardware RAID configuration and LVM configuration.

Note

Boot partitions that span or mirror, using software over multiple physical disks, are not supported by AWS Elastic Disaster Recovery and are not recommended for use in EC2. If your source server's configuration contains mirrored boot partition (e.g. Windows Mirrored Disks), we recommend installing the agent to replicate only one physical disk of the mirrored boot partition using the `--devices` parameter.

- Windows EC2 RAID Documentation.
- Linux EC2 RAID Documentation.

Do Replication Servers have the SSM agent installed on them?

The SSM agent is intentionally not installed on these servers as part of our security architecture to maintain isolation and prevent any potential unauthorized access paths.

What is there to note regarding SAN/NAS Support?

If the disks are represented as block devices on the machine, as most SAN are, Elastic Disaster Recovery will replicate them transparently, just like actual local disks.

If the disks are mounted over the network, such as an NFS share, as most NAS implementations are, the AWS Replication Agent would need to be installed on the actual NFS server in order to replicate the disk.

Does AWS Elastic Disaster Recovery support Windows License Migration?

AWS Elastic Disaster Recovery conforms to the Microsoft Licensing on AWS guidelines.

Can you perform an OS (Operating System) upgrade with AWS Elastic Disaster Recovery?

No. AWS Elastic Disaster Recovery copies the entire machine as-is. However, you can copy the data disks exclusively and attach them to a new machine with an upgraded OS.

What are the private APIs used by AWS DRS to define actions in the IAM Policy?

AWS Elastic Disaster Recovery (AWS DRS) utilizes the following private API resources as actions in the IAM Policy. Learn more about actions, resources, and condition keys for Elastic Disaster Recovery.

- BatchCreateVolumeSnapshotGroupForDRS – Grants permission to create volume snapshot group.
- BatchDeleteSnapshotRequestForDRS – Grants permission to batch delete snapshot request.
- DescribeReplicationServerAssociationsForDRS – Grants permission to describe replication server associations.
- DescribeSnapshotRequestsForDRS – Grants permission to describe snapshots requests.
- GetAgentCommandForDRS – Grants permission to get agent command.
- GetAgentConfirmedResumeInfoForDRS – Grants permission to get agent confirmed resume info.
- GetAgentInstallationAssetsForDRS – Grants permission to get agent installation assets.
- GetAgentReplicationInfoForDRS – Grants permission to get agent replication info.
- GetAgentRuntimeConfigurationForDRS – Grants permission to get agent runtime configuration.
- GetAgentSnapshotCreditsForDRS – Grants permission to get agent snapshots credits.
- GetChannelCommandsForDRS – Grants permission to get channel commands.
- NotifyAgentAuthenticationForDRS – Grants permission to notify agent authentication.
- NotifyAgentConnectedForDRS – Grants permission to notify agent is connected.
- NotifyAgentDisconnectedForDRS – Grants permission to notify agent is disconnected
- NotifyAgentReplicationProgressForDRS – Grants permission to notify agent replication progress.
- RegisterAgentForDRS – Grants permission to register agent.
- SendAgentLogsForDRS – Grants permission to send agent logs.
- SendAgentMetricsForDRS – Grants permission to send agent metrics.
- SendChannelCommandResultForDRS – Grants permission to send channel command result.
- SendClientLogsForDRS – Grants permission to send client logs.
- SendClientMetricsForDRS – Grants permission to send client metrics.
- UpdateAgentBacklogForDRS – Grants permission to update agent backlog.
- UpdateAgentConversionInfoForDRS – Grants permission to update agent conversion info.

- UpdateAgentReplicationInfoForDRS – Grants permission to update agent replication info.
- UpdateAgentSourcePropertiesForDRS – Grants permission to update agent source properties.
- IssueAgentCertificateForDrs – Grants permission to issue an agent certificate.
- CreateConvertedSnapshotForDrs – Grants permission to create converted snapshot.

What post-launch scripts does AWS Elastic Disaster Recovery support?

DRS can run scripts on a launched drill or recovery instance. This is done by creating the following folder on the source server and placing the scripts within that folder.

Linux: /boot/post_launch (any files that are marked as executable)

Windows: C:\Program Files (x86)\AWS Replication Agent\post_launch\ (any .exe, .cmd, or .bat files)

Once you put these scripts in the above folders on the source server, the folder will be replicated to the drill or recovery instance and be executed once after the instance boots for the first time.

> **Note**
>
> Post-launch scripts on Windows run under the Local System context. Post-launch scripts on Linux run under the 'root' user.

Is BitLocker encryption supported?

DRS does not support OS-based disk encryption features such as BitLocker. These should be deactivated before using AWS Elastic Disaster Recovery.

Can I set instance metadata on my launched instance to support IMDSv2 only?

You can easily set Instance Metadata Service Version 2 (IMDSv2) on your recovery instances using the EC2 launch template associated with your DRS source server.

Follow the instructions on the EC2 launch template page.

When you are redirected to the EC2 console to modify your template, take the following steps

- Click **Advanced details > Metadata version**.
- Select **V2 only (token required)**.

You can then set this launch template as your default version.

Elastic Disaster Recovery Concepts

What is the Recovery Time Objective (RTO) of Elastic Disaster Recovery?

The Recovery Time Objective (RTO) of Elastic Disaster Recovery is typically measured in minutes. The RTO is highly dependent on the OS boot time.

A: When launching a recovery job, the AWS DRS orchestration process creates cloned volumes by using the replicated volumes in the replication staging area. During this process, AWS DRS also initiates a process that converts all volumes that originated outside of AWS into AWS-compatible volumes, which are attached to EC2 instances that can boot natively on AWS. The job and boot time depend on the following environment conditions:

1. OS type: The average recovered Linux server normally boots within 5 minutes, while the average recovered Windows server normally boots within 20 minutes because it is tied to the more resource-intensive Windows boot process.
2. OS configuration: The OS configuration and application components it runs can impact the boot time. For example, some servers run heavier workloads and start additional services when booted, which may increase their total boot time.
3. Target instance performance: AWS DRS sets a default instance type based on the CPU and RAM provisioned on the source server. Changing to a lower performance instance type will result in a slower boot time than that of a higher performance instance type.
4. Target volume performance: Using a lower performance volume type will result in a slower boot time than that of a higher performance volume type with more provisioned IOPS.

What is the Recovery Point Objective (RPO) of Elastic Disaster Recovery?

The Recovery Point Objective (RPO) of Elastic Disaster Recovery is typically in the sub-second range.

AWS Elastic Disaster Recovery (AWS DRS) provides continuous block-level replication, recovery orchestration, and automated server conversion capabilities. These allow customers to achieve a crash-consistent recovery point objective (RPO) of seconds, and a recovery time objective (RTO) typically ranging between 5–20 minutes. Below is an explanation of how RPO and RTO are measured, how AWS DRS supports these RPOs and RTOs, and what common environment conditions can impact RPO and RTO.

Recovery Point Objective (RPO)

How is RPO measured?

RPO is measured based on the latest point in time in which block data was written to the source server volume(s) and successfully copied in a crash-consistent state into the replication staging area located in the customer's target AWS account.

How does AWS DRS allow an RPO of seconds?

The AWS Replication Agent continuously monitors the blocks written to the source server volume(s), and immediately attempts to copy the blocks across the network and into the replication staging area subnet located in the customer's target AWS account. This continuous replication approach allows an RPO of seconds as long as the written data can be immediately copied across the network and into the replication Staging Area volumes.

> **Important**
>
> A crash-consistent recovery point allows the successful recovery of crash-consistent applications, such as databases. The recovery point will include any data that has been successfully written to the source server volume(s). Application data that is kept in memory is not replicated to the target replication Staging Area until it is written to the source server volume(s). Therefore, if a disruption occurs before in-memory application data is written to the volume(s), this data will not be available on the target server when launched for test or recovery purposes.

What environment conditions can impact the ability to achieve a typical RPO of seconds?

To achieve an RPO of seconds, AWS Elastic Disaster Recovery primarily requires that the outbound network, inbound network, and staging area resources must allow data to be copied across the network and written to the target environment faster than the rate at which it is written to the source volume(s). In the case that block writes burst at faster rates than these components can support, the RPO will temporarily increase until the data replication can catch up, at which point the RPO will return to seconds. Examples:

1. Outbound network: If a source server writes block data at a rate of 10 MB/second, the outbound network bandwidth must also support a rate of at least 10 MB/second in order to maintain a seconds RPO. If the source network contains 10 servers that each write at an average rate of 10 MB/second, the total bandwidth will need to support a rate of at least 100 MB/second in order to allow a seconds RPO.
2. Inbound network: Once the replicated data is sent from the source network, it must enter the target network at a rate greater to that at which the data is written to the source servers and sent from the source network in order to maintain a seconds RPO.
3. Staging area resources: When the data arrives to the target network, it is received by the AWS DRS replication server instance(s), which in turn writes the replicated data to attached EBS volumes. Both the replication server instance(s) and attached Amazon EBS volumes must allow the data to be written at a rate faster than that at which it is written to the source servers and sent by the source network in order to maintain an RPO of seconds.

What happens if the block data written to the source volume(s) cannot be sent immediately to the target replication Staging Area Subnet?

If the block data written on the source volume(s) cannot be sent immediately to the target replication Staging Area, the RPO will increase until the data can be flushed across the network. During this time, you will still be able to recover your server(s), but to a recovery point older than seconds, in accordance with the increase in RPO. The RPO represents the latest crash-consistent point in time during which data was replicated.

What are Point in Time Snapshots?

Point in Time (PIT) Snapshots are an AWS Elastic Disaster Recovery feature which allows launching a Recovery Instance of a Source Server from a set of EBS Snapshots captured at a specific moment in time. The PIT Snapshot is a crash-consistent recovery point of your Source Server, and represent

your Recovery Point Objective (RPO). After Source Servers complete **Initial Sync** and maintain **Healthy** replication status, Point in Time states are automatically created and stored in accordance to your snapshot retention policy.

Each PIT Snapshot for a Source Server consists of one or more EBS Snapshots; one EBS snapshot for each volume being replicated. See below for where the EBS snapshots are stored:

Replication Strategy	Replication Target	EBS Snapshot S3 Region	EBS Snapshot Account
Single Account	Any Region	Same Region as Replication Target	Same AWS Account
Extended Account	Any Region	Same Region as Replication Target	Staging Account
Multi-Account	Any Region	Same Region as Replication Target	Target Account
Reverse Replication	Any Region	Same Region as Source	Source Account
Any	Outpost	Stored locally on Outpost	Outpost Account

What is the PIT Snapshot Retention Rate?

Elastic Disaster Recovery has the following default PIT Snapshot frequency and retention schedule:

- **Minute** - 1 PIT Snapshot per 10 minutes for the prior 1 hour.
- **Hour** - 1 PIT Snapshot per 1 hour for the prior 24 hours.
- **Day** - 1 PIT Snapshot per 1 day for the prior 7 days.

Can I adjust the PIT Snapshot Retention Rate?

You can only adjust the **Day** PIT Snapshot retention limit from 1 day through 365 days in the replication settings. As each PIT Snapshot consists of one or more EBS snapshots, increasing the PIT Snapshot retention rate can result in additional EBS costs. The frequency (i.e. how often) that

AWS Elastic Disaster Recovery creates snapshots are not configurable. Learn more about managing Point in Time retention.

What is "Use most recent data"?

"**Use most recent data**" is a feature available when selecting a PIT Snapshot from the AWS Elastic Disaster Recovery console during a **Recovery Drill** or **Recovery**. It is implicitly used when a **Recovery Drill** or **Recovery** is started programmatically (e.g. AWS CLI) without specifying a PIT Snapshot. When used, AWS Elastic Disaster Recovery will attempt to create an on-demand PIT Snapshot of all Source Servers within the Recovery Job, representing a sub-second RPO of the Source Server. This PIT Snapshot will be consistent to the time the Recovery Job was submitted.

DRS requires an active network connection to the Source Server to successfully create this new PIT Snapshot. AWS Elastic Disaster Recovery may be unable to create this PIT Snapshot for various reasons, and will wait for up to 10 minutes for this new PIT Snapshot to be created. If DRS is unable to create this PIT Snapshot, it will use a snapshot based on the last consistent state from data on the Replication Server. Reasons why "**Use most recent data**" may fail to successfully create a PIT Snapshot include:

- Unable to communicate with the Source Server.
- Unable to transfer all changes present on the Source Server within the timeout window.

As "**Use most recent data**" requires an active network connection to the Source Server to create a new PIT Snapshot, there may be circumstances (e.g Source Server is offline) where your RTO will be shortened by selecting an existing PIT Snapshot rather than waiting for "**Use most recent data**" to timeout.

What is "Any" and "All" in Point in Time Snapshot selection?

The **Any** and **All** selection criteria are available in the AWS Elastic Disaster Recovery Console when selecting a Point in Time Snapshot for a job that contains multiple Source Servers.

- **Any** - Displays all of the PIT Snapshots available for all of the selected source servers. AWS Elastic Disaster Recovery will launch a drill instance for each source server that has a PIT snapshot taken at the chosen time. For any Source Server that does not have a corresponding PIT snapshot taken at the chosen time, a previous PIT Snapshot will be used.
- **All** - Displays only PIT Snapshots that all selected Source Servers share. If there are no points in time that include all Source Servers, the list will be empty.

Agent related

What does the AWS Replication Agent do?

The AWS Replication Agent performs an initial block-level read of the content of any volume attached to the server and replicates it to the replication server. The Agent then acts as an OS-level read filter to capture writes and synchronizes any block level modifications to the Elastic Disaster Recovery replication server, ensuring near-zero RPO.

What kind of data is transferred between the Agent and the AWS Elastic Disaster Recovery Service Manager?

The AWS Replication Agent sends the following types of information to the AWS Elastic Disaster Recovery Service Manager:

- Monitoring metrics of the Agent itself
- Replication status (started, stalled, resumed)
- Backlog information
- OS and hardware information.

When an Agent is installed on a source server, it collects the following information on the machine:

- Host name and ID.
- List of CPUs including models and number of cores
- Amount of RAM
- Hardware and OS information.
- Number of disks and their size – in Windows, disk letters; in Linux, block device names.
- Installed applications (Windows)
- Installed Packages (Linux)
- Running services.
- Machine's Private IP address.

Can a proxy server be used between the source server and the Elastic Disaster Recovery Console?

Yes. You can configure the proxy either by using an environment variable prior to the installation (Linux and Windows), or by using the `--proxy-address` flag in the Linux installer.

Using the installer: `./aws-replication-installer-init --proxy-address https://PROXY:PORT/`

Using environment variable: `export https_proxy=https://PROXY:PORT/; ./aws-replication-installer-init`

Make sure the proxy has a trailing forward slash (/).

What are the pre-requisites needed to install the AWS Replication Agent?

The installation requirements for source server depend on the type of OS that the server runs – either Linux or Windows.

View the prerequisites.

What ports does the AWS Replication Agent utilize?

The Agent utilizes TCP Port 443 to communicate to the Elastic Disaster Recovery Service Manager and TCP Port 1500 for replication to AWS.

What kind of resources does the AWS Replication Agent utilize?

The AWS Replication Agent is lightweight and nondisruptive. The agent utilizes approximately 5% CPU and 300 MB of RAM.

Can Elastic Disaster Recovery migrate containers?

Elastic Disaster Recovery only supports the replication of full servers. Nevertheless, Elastic Disaster Recovery replicates on a server level and therefore any containers within the selected servers will be replicated.

Does the AWS Replication Agent cache any data to disk?

Elastic Disaster Recovery does not write any cache or do any sort of journalling to disk. The Agent holds a buffer which is large enough to map all volume's blocks ~250 MB in memory.

The Agent then acts as a sort of write filter and will replicate changed blocks directly from memory to the replication server. In cases where the data is no longer in memory, the Agent will read the block from the volume directly. This is the case where you may see backlog in the Elastic Disaster Recovery Console. The cause of this is the volume of change is greater than the bandwidth available.

How is communication between the AWS Replication Agent and the Elastic Disaster Recovery Service Manager secured?

All communication is encrypted using SSL. In addition, each Agent is assigned a key during installation which is used to encrypt all traffic. All keys are unique and are not shared across multiple Agents.

Is it possible to change the port the AWS Replication Agent utilizes from TCP Port 1500 to a different port?

No. The Elastic Disaster Recovery Agent can only utilize TCP Port 1500 for replication.

How do I manually uninstall the Elastic Disaster Recovery Agent from a server?

Please refer to:the section called “Uninstalling the agent”.

When do I need to reinstall the Agent?

Agent re-installations are required in these cases:

- After adding new volumes if the Automatic replication of new disks option is not activated for the source server where the volume is added.
- Windows OS upgrades (ex. Windows Server 2012 to Windows Server 2016)
- Some new features require a re-installation to apply. In this case, the feature documentation will specifically state that this is a requirement for the feature to be activated.

How much bandwidth does the AWS Replication Agent consume?

The AWS Replication Agent opens up to five connections and will attempt to maximize available bandwidth.

Throttling can be activated by selecting the specific server and selecting the **Settings** page in the Elastic Disaster Recovery Console.

How many disks can the AWS Replication Agent replicate?

The Agent can replicate up to 50 disks from a single server. Ensure that the replication server instance type supports at least the number of disks being replicated.

Is it possible to add a disk to replication without a complete resync of any disks that have already been replicated?

When you add a disk to a source server, AWS Elastic Disaster Recovery will automatically identify it and add it to the **Disk settings** tab in the console.

This feature is activated automatically for newly added servers. Learn how to deactivate or reactivate this feature.

Which Windows and Linux OSs support no-rescan upon reboot?

A shutdown (from the OS menu or CLI) of any supported Linux or Windows source server no longer causes a rescan in DRS once the source server is restarted. A rescan means that the agent on the source server rereads all blocks on all replicated disks and transmits blocks that are different from the previously replicated data. A rescan is similar to the initial sync but is faster because only blocks that are different need to be transmitted.

Rescans can still happen following a hard reboot, crashes, or when you add or remove disks to or from the source server. In addition, a rescan will occur if the underlying Storage types do not use static DUIDs (such as 3PARdata). Supported OSs include:

Windows Server

- 2012 and newer

Linux

- CentOS 6–8
- Oracle 6–10
- RHEL 6–10
- Rocky 8 and 9
- SLES 12 and 15
- Debian 9–11
- Ubuntu 16, 18, 20, and 22
- Amazon Linux 2

No-Rescan Upon Reboot Limitations

A rescan may still occur in certain circumstances, including:

Hard Reboot or Power Loss.

> **Important**
>
> **A rescan duration may impact your RPO**
>
> - While a rescan is conducted, a point of time recovery cannot be made.
> - If a disaster occurs during the rescan you will only be able to restore point of time from before the rescan began. This could affect your ability to meet your RPO.

How do temporary credentials work?

The temporary credential mechanism was developed specifically to provide an easy and secure way to install AWS DRS Agents. The main flow of the temporary credentials' creation process relies on generating a x509 certificate per agent and then using this x509 certificate to receive temporary IAM credentials. This process utilizes a similar mechanism to the one used by IAM Roles Anywhere.

Where can I find the AWS DRS Replication Agent logs

The AWS DRS agent logs are stored in agent.log.0:

- **Linux:** /var/lib/aws-replication-agent/agent.log.0

- **Windows 64 bit:** C:\Program Files (x86)\AWS Replication Agent\agent.log.0

In addition, you can review the installation log located in: <install_path>\aws_replication_agent_installer.log

Replication related

Can we set specific IP addresses for the replication server or conversion server in the AWS DRS staging area?

You cannot specify or assign static IP addresses for the replication server or conversion server in AWS DRS. These servers are managed and maintained by the DRS service.

What do Lag and Backlog mean during replication?

During replication you may see a server fall out of Continuous Data Protection (CDP) mode. This may occur for various reasons, typically related to the network throughput or interruption.

- **Lag** – The amount of time since the server was last in CDP mode.
- **Backlog** – The amount of data that was written to the disk and still needs to be replicated in order to reach CDP mode.
- **ETA** – The estimated time remaining to return to CDP.

Is the replicated data encrypted?

Elastic Disaster Recovery encrypts all the data in transit.

How is the replication server provisioned and managed in the Staging Area?

AWS Elastic Disaster Recovery automatically provisions replication servers in your staging area subnet when source servers are added. The service manages the full lifecycle of replication servers, including launching new servers when additional replication capacity is needed, removing servers that are no longer in use, and recycling servers every 14 days to ensure they run the latest AMI with up-to-date security patches. Each replication server can handle replication of volumes from multiple source servers. You can configure the replication server instance type and subnet in the replication settings.

What type of replication server is utilized in the Elastic Disaster Recovery Staging Area?

AWS Elastic Disaster Recovery provisions a t3.small server by default. The typical ratio of volumes to replication servers is 15:1.

What happens if the configured replication server instance type is unavailable?

If the replication server instance type that you configured in your replication settings is unavailable in the staging area subnet's Availability Zone, AWS Elastic Disaster Recovery automatically launches the replication server using an alternative instance type. This can occur, for example, when the configured instance type returns an `InsufficientInstanceCapacity` error. This behavior allows continuous data replication to proceed without interruption so that your recovery point objective (RPO) is maintained.

Replication servers are managed by AWS Elastic Disaster Recovery throughout their lifecycle and are automatically recycled approximately every 14 days. When a replication server is replaced, AWS Elastic Disaster Recovery provisions the new server using your current replication settings.

Does AWS Elastic Disaster Recovery compress data during replication?

Yes, AWS Elastic Disaster Recovery utilizes LZ4 compression during transit resulting in 60-70% compression depending on the type of data.

Are events that are generated by the AWS Elastic Disaster Recovery servers logged in Cloudtrail in AWS?

Yes, AWS Elastic Disaster Recovery is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service in AWS Elastic Disaster Recovery (AWS DRS). CloudTrail captures all API calls for AWS DRS as events. The calls captured include calls from the AWS DRS console and code calls to the AWS DRS API operations. Learn more about AWS DRS and Cloudtrail.

How many snapshots does Elastic Disaster Recovery create?

Point in Time (PIT) is a disaster recovery feature which allows launching an instance from a snapshot captured at a specific Point In Time. As source servers are replicated, Point in Time states

are chronicled over time, while a retention policy will determine which Points in Time are not required after a defined duration.

Elastic Disaster Recovery has the following PIT state schedule:

- Every 10 minutes for the last hour
- Once an hour for the last 24 hours
- Once a day for the last 7 days (or a different retention period, as configured)

You can increase or decrease the default 7 day snapshot retention rate from anywhere between 1 day and 365 days in the replication settings. Learn more about managing Point in Time retention.

Does Elastic Disaster Recovery delete snapshots?

AWS Elastic Disaster Recovery automatically deletes snapshots that are no longer used (such as those left over after source servers have been removed from the Elastic Disaster Recovery Console) or those that are past the designated retention setting.

How much capacity is allocated to the staging area?

A volume is created for each volume in the source infrastructure of the same size. The EBS volumes will be a 1:1 match for the source machines provisioned size.

Why is 0.0.0.0:1500 added to inbound rules in the Staging Area?

AWS Elastic Disaster Recovery uses TCP Port 1500 for replication between the source agents and the replication server. The connection is open for all IPs and can be managed by ACLs or network controls to limit inbound IPs.

What should I know about rescans?

During a rescan, the agent on the source server rereads the data on all replicated disks and transmits any differences from the previously replicated data. A rescan is similar to the initial sync but is faster because only changes are replicated. The rescan speed depends on factors such as the size of the source disks and disk performance. A rescan is functioning normally as long as it's progressing.

Causes of rescans include:

- Hard reboots/crashes.
- Adding or removing disks or modifying the size of disks on the source server.
- The OS is not supported for the no-rescan feature. Learn more in Supported Windows operating systems and Supported Linux operating systems.
- Writing data to disks while the Replication Agent driver is unhooked.

How does AWS Elastic Disaster Recovery ensure that replication servers are patched and follow security best practices?

AWS Elastic Disaster Recovery automatically recycles replication servers every 14 days. When a replication server reaches the end of its lifecycle, the service terminates it and launches a new replication server using the latest AMI, which includes the most recent security patches and updates. During the recycling process, you may observe temporary replication lag while the new replication server is launched and the AWS Replication Agent reconnects. Replication resumes automatically once the new server is ready.

Is the Elastic Disaster Recovery replication crash consistent?

Yes, AWS Elastic Disaster Recovery's replication is crash consistent.

How can I run a network bandwidth test to an Amazon EC2 endpoint?

You can measure the bandwidth available for replication by running a test over TCP port 1500 with TLS, using a disposable Amazon EC2 endpoint that you launch and then terminate. For the full procedure, see Running a network bandwidth test to an EC2 endpoint.

AWS related

What does the Elastic Disaster Recovery machine conversion server do?

The machine conversion server converts the disks to boot and run on AWS.

Specifically, it makes bootloader changes, injects hypervisor drivers, and installs cloud tools.

How do I change the server AMI on AWS after recovery?

After the machine has been launched by AWS Elastic Disaster Recovery, switching the AMI can be done by launching a vanilla machine from the required AMI, stopping that machine, detaching

all the disks (including the root) and then attaching the disks from the drill or recovery instance created by Elastic Disaster Recovery.

Which AWS services are automatically installed when launching a drill or recovery instance?

AWS Elastic Disaster Recovery (AWS DRS) automatically installs EC2Config. After installation, EC2Config automatically installs the SSM EC2 Configuration Service.

CloudWatch, AWS Powershell or CLI are not automatically installed. This can be done by combining the AWS DRS APIs and the AWS APIs - you can use the AWS DRS APIs to determine the EC2 instance IDs of the machines and then use AWS API/CLI to turn on the detailed monitoring. An alternative approach would be to do it via AWS API only based on the tags you associate with the machine. A third approach would be to do so from the post-launch script.

AWS DRS installs EC2Launch (Windows 2016 only). Customers need to configure EC2Launch based on the specific requirements explained here. This configuration step needs to be performed post Recovery using the wizard in C:\ProgramData\Amazon\EC2-Windows\Launch\Settings \Ec2LaunchSettings.exe on the drill or recovery instance.

How long does it take to copy a disk from the AWS Elastic Disaster Recovery staging area to production?

AWS Elastic Disaster Recovery uses internal cloud provider snapshots. This process typically takes less than a minute and the size of the volume does not impact the time.

What are the differences between conversion servers and replication servers?

Replication servers run on Linux and conversion servers (for Windows machines) run on Windows.

The conversion is done by AWS Elastic Disaster Recovery automatically bringing up a vanilla Windows conversion server machine in the same subnet with the replication servers as part of the launch job.

Both conversion and replication servers have Public IPs

The conversion servers will use the same security groups as the replication server.

The conversion servers must be able to access the AWS Elastic Disaster Recovery Service Manager.

The conversion server machines, just like the Replication servers are managed automatically by Elastic Disaster Recovery. Any attempt to disrupt their automated functionality will result in failed conversions.

Can I prevent Elastic Disaster Recovery from cleaning up drill instance resources in AWS?

AWS Elastic Disaster Recovery will, by default, remove any resources created during the drill process either when requested by the user or when a new drill instance is launched.

To prevent this in AWS, you can Activate Termination Protection for the drill or recovery instance, and the resources will not be removed upon a new instance launch.

Why are my Windows Server disks read-only after launching the drill or recovery instance?

When launching drill or recovery instances Windows Server may boot with all the disks as read-only.

This is a common issue that occurs when detaching and attaching data disks. It can be resolved by following the steps in this Microsoft TechNet article.

What impacts the conversion and boot time of drill and recovery instances?

Prior to launching the drill or recovery instance, AWS Elastic Disaster Recovery goes through a machine conversion server process on the boot volume. The conversion process is fairly quick.

While the actual conversion process itself is quick, the time to boot the drill or recovery instance varies depending on many factors unrelated to any Elastic Disaster Recovery processes. Some of these are controllable and should be taken into account when drill or recovery times are of importance.

- Operating system - The amount of time required to boot the operating system is dependent on the OS itself. While Linux servers typically boot quickly, Windows servers may take additional time, due to the nature of the Windows OS. If opportunity permits, drill the boot time of the source server. If Linux OS takes a long time to boot, ensure to check that dhclient (Dynamic Host Configuration Protocol Client) is installed on the system so it can pull an IP.

- Scheduled Windows Updates - If the Windows server has pending patches, ensure those are installed prior to launching the drill or recovery instance. If pending patches remain, the boot time in the cloud may be severely impacted as the patch process may commence upon the initial boot.
- Boot volume type - Depending on services/applications, boot time may be impacted by disk performance. It is recommended that boot volumes be drilled with a higher performance SSD and even by provisioning IOPs to ensure throughput. This may be more critical during the first initial boot of the server in the cloud, as all initial settings are applied. In many cases, the boot volume type may be scaled back after the initial boot and should be drilled.

> **Note**
>
> The first boot of Windows machines on AWS may take up to 45 minutes due to Windows adjusting to the AWS virtual hardware.

How is the AWS Licensing Model Tenancy chosen for Elastic Disaster Recovery?

Elastic Disaster Recovery conforms to the Microsoft Licensing on AWS guidelines.

How does Elastic Disaster Recovery interact with interface VPC endpoints?

If you use Amazon Virtual Private Cloud (Amazon VPC) to host your AWS resources, you can establish a private connection between your Amazon VPC and AWS Elastic Disaster Recovery. You can use this connection to allow AWS Elastic Disaster Recovery to communicate with your resources on your VPC without going through the public internet.

Amazon VPC is an AWS service that you can use to launch AWS resources in a virtual network that you define. With a VPC, you have control over your network settings, such as the IP address range, subnets, route tables, and network gateways. With VPC endpoints, the routing between the Amazon VPC and AWS services is handled by the AWS network, and you can use IAM policies to control access to service resources.

To connect your VPC to Elastic Disaster Recovery, you define an *interface VPC endpoint* for Elastic Disaster Recovery. An interface endpoint is an elastic network interface with a private IP address

that serves as an entry point for traffic destined to a supported AWS service. The endpoint provides reliable, scalable connectivity to Elastic Disaster Recovery without requiring an internet gateway, network address translation (NAT) instance, or VPN connection. For more information, see What is Amazon VPC in the *Amazon VPC User Guide*.

Interface VPC endpoints are powered by AWS PrivateLink, an AWS technology that facilitates private communication between AWS services using an elastic network interface with private IP addresses. For more information, see AWS PrivateLink.

For more information, see Getting Started in the *Amazon VPC User Guide*.

If the AWS replication agents are installed with a principal using AWSElasticDisasterRecoveryAgentInstallationPolicy and a VPCE policy is used (to scope down access), add the following statement to your policy:

```
{
            "Effect": "Allow",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": "arn:aws:execute-api:<region>:*.*/POST/CreateSessionForDrs"
            }
```

Will AWS Elastic Disaster Recovery reserve EC2 capacity for recovery?

AWS Elastic Disaster Recovery relies on Amazon EC2 On-Demand pools by default. If a specific Amazon EC2 instance type is unavailable to support your recovery, DRS will automatically attempt to scale up the instance repeatedly until an available instance type is found, but in extreme circumstances, instances may not always be available. To ensure the availability of the required instance types you need for your most critical applications, you may purchase EC2 Capacity Reservations. You can specifically designate which applications you want to use the EC2 Capacity Reservations for by using launch templates.

Advanced FAQ

Does AWS DRS support Nutanix?

Nutanix hypervisor is supported along with other hypervisor vendors. The AWS Replication Agent is installed on the virtual machine (VM) and performs block level replication. In addition, the client ISO is booted for failback on the same VM itself.

Does AWS DRS support VMware vSphere?

VMware vSphere is supported (both on-premises as well as VMware on AWS). Examples of detailed walkthroughs: Disaster recovery for VMware Cloud on AWS using AWS Elastic Disaster Recovery. Performing a failback with the DRS Mass Failback Automation client.

Does AWS DRS support Microsoft Hyper-V?

Both Hyper-V and Microsoft Azure are supported. The AWS Replication Agent installation and replication follows the same process described in Adding source servers. For failback to Azure, review the Building a disaster recovery site on AWS for workloads on Microsoft Azure blog post.

Release Notes

AWS Elastic Disaster Recovery performs regular Service and Client releases of new features and capabilities.

Service Release Notes

Client Release Notes

AWS Elastic Disaster Recovery Service Release Notes

September 2026

- AWS Elastic Disaster Recovery now supports source servers that run on AWS Graviton (`arm64`) Amazon EC2 instances. You can replicate a 64-bit Linux `arm64` Amazon EC2 instance and recover it to an `arm64` instance type. Install the architecture-specific `arm64` AWS Replication Agent installer on the source server. Recovery instances boot with UEFI, and replication servers continue to use `x86_64` instance types. `arm64` source servers support recovery and failback within AWS only. On-premises and other cloud sources, Windows on `arm64`, and the Failback Client are not supported. For more information, see AWS Graviton source servers.

August 2026

- AWS Elastic Disaster Recovery now supports **recovery plans**. With recovery plans, you can recover groups of source servers in a defined order, with wait times between groups. A recovery plan contains up to 20 ordered steps and up to 100 source servers. Each server step recovers its servers in parallel and must finish before the next step begins, and each server can be marked critical or optional to control whether its failure stops the plan. You can run a plan as a drill or as a recovery, follow its progress per step and per server, and retry, skip, or cancel steps while it runs. Recovery plans are available in the AWS Elastic Disaster Recovery console and through the AWS Elastic Disaster Recovery API. For more information, see Orchestrating recovery with recovery plans.
- AWS managed policy update – We updated the AWSElasticDisasterRecoveryReadOnlyAccess policy with new read-only permissions for recovery plans and recovery plan executions.
- AWS Elastic Disaster Recovery automatically preserves Unified Extensible Firmware Interface (UEFI) boot mode for Linux source servers that boot using UEFI firmware. Previously, AWS Elastic

Disaster Recovery converted Linux UEFI source servers to legacy Basic Input/Output System (BIOS) during recovery and drill launches. With this change, your recovered instances launch with the same UEFI boot mode as your source servers. This eliminates the need for post-launch boot configuration adjustments.

- With AWS Elastic Disaster Recovery, you can now protect source servers that have volumes of up to 64 TiB, increased from 16 TiB. To replicate a volume larger than 16 TiB, choose the gp3 staging disk type.
- For gp3 staging disks that you configure manually, you can now provision up to 80,000 IOPS and 2,000 MiB/s of throughput, increased from 16,000 IOPS and 1,000 MiB/s.

July 2026

- With AWS Elastic Disaster Recovery, you can now choose a **Recovery mode** in your launch configurations. Choose **Fast** to skip the conversion process and reduce recovery time. This mode launches instances directly from replicated snapshots. Choose **Optimal** (default) to run the full conversion process before launch. To use Fast recovery mode, you must have version 6.42.20 or later of the agent. For more information, see Editing the default AWS DRS launch settings.
- AWS Elastic Disaster Recovery now preserves and passes through `VolumeInitializationRate` values set on EC2 launch template block device mappings during drill and recovery launches. This accelerates Amazon EBS volume initialization without requiring AWS Elastic Disaster Recovery configuration changes. For more information, see Key considerations for EC2 launch templates.
- AWS Elastic Disaster Recovery launched in the following AWS Regions: Asia Pacific (Malaysia), Asia Pacific (New Zealand), Asia Pacific (Taiwan), Asia Pacific (Thailand), Canada West (Calgary), and Mexico (Central).

June 2026

- Added support for Debian 12 and Debian 13 as source server operating systems. Debian 13 requires AWS Elastic Disaster Recovery Agent version 6.42.21 or later. See Supported Linux operating systems.
- AWS managed policy update – Updated policies AWSElasticDisasterRecoveryConsoleFullAccess_v2 and AWSElasticDisasterRecoveryLaunchActionsPolicy to add permissions for SSM documents with the `AWSDRS-` prefix.

April 2025

- AWS Elastic Disaster Recovery is now authorized for Department of Defense Cloud Computing Security Requirements Guide Impact Levels 4 and 5 (DoD CC SRG IL4 and IL5) in the AWS GovCloud (US-East and US-West) Regions.

February 2025

- Added support for RHEL 9.5.
- Added ability to add tags to Amazon EBS snapshots.

October 2024

- Added support for Oracle 9.0-9.4.
- You can use AWS Elastic Disaster Recovery with workloads that require FedRAMP High categorization level in the AWS GovCloud (US-East and US-West) Regions. Learn more in Federal Risk and Authorization Management Program.

September 2024

- AWS Elastic Disaster Recovery now supports recovery to AWS Local Zones.

July 2024

- AWS Elastic Disaster Recovery now supports Flexible Instance Types.
- AWS managed policy updates – Created managed policy revisions to support FlexibleInstances feature for DRS. The following managed policies were updated:

  AWSElasticDisasterRecoveryConsoleFullAccess_v2

  AWSElasticDisasterRecoveryReadOnlyAccess

May 2024

- AWS Elastic Disaster Recovery now supports protecting Source Servers with up to 60 volumes.

April 2024

- AWS Elastic Disaster Recovery now supports AWS Outposts. For more information see: Working with AWS DRS and Outposts.
- Source Networks – Added support for replicating Security Groups with references to other Security Groups.

January 2024

- AWS managed policy update – Updated AWSElasticDisasterRecoveryServiceRolePolicy and AWSElasticDisasterRecoveryCrossAccountReplicationPolicy policies to support replicating marketplace licenses to launched instances.
- Source Networks – Added support for replicating Security Groups with Prefix Lists.

November 2023

- AWS Elastic Disaster Recovery is now generally available in the AWS GovCloud (US) Regions. This launch gives customers in both the public and commercial sectors, as well as their partners, access to AWS DRS capabilities in the AWS GovCloud (US) Regions.
- Introduced disaster recovery drill validation automation for AWS Elastic Disaster Recovery, this allows you to automate validations when launching EC2 instances for recovery and drills.
- AWS managed policy update – updated AWSElasticDisasterRecoveryReadOnlyAccess to support describing additional post-launch actions.
- New AWS managed policy – Added new policy: AWSElasticDisasterRecoveryConsoleFullAccess_v2.
- AWS managed policy updates – Created new revisions to support DRS in AWS GovCloud and added Statement ID (SID) to managed policy statements. The following managed policies were updated:

  AWSElasticDisasterRecoveryAgentPolicy

  AWSElasticDisasterRecoveryAgentInstallationPolicy

  AWSElasticDisasterRecoveryEc2InstancePolicy

  AWSElasticDisasterRecoveryConsoleFullAccess

  AWSElasticDisasterRecoveryLaunchActionsPolicy

AWSElasticDisasterRecoveryNetworkReplicationPolicy

AWSElasticDisasterRecoveryRecoveryInstancePolicy

AWSElasticDisasterRecoveryServiceRolePolicy

AWSElasticDisasterRecoveryConversionServerPolicy

AWSElasticDisasterRecoveryFailbackPolicy

AWSElasticDisasterRecoveryFailbackInstallationPolicy

AWSElasticDisasterRecoveryStagingAccountPolicy_v2

AWSElasticDisasterRecoveryStagingAccountPolicy

AWSElasticDisasterRecoveryReplicationServerPolicy

- New revision of AWSElasticDisasterRecoveryCrossAccountReplicationPolicy policy to support DRS in GovCloud

October 2023

- Introduced a new feature: Recover into existing instance, allowing you to set an existing EC2 instance as the target of a drill, recovery or failback launch, instead of launching a new instance.
- AWS managed policy update – Updated policies AWSElasticDisasterRecoveryConsoleFullAccess and AWSElasticDisasterRecoveryLaunchActionsPolicy to support launching into existing instance.

September 2023

- Introduced a new feature: Post-launch actions framework for automating any action needed to be performed on recovery instances after launch.
- Service launch in Israel (Tel Aviv) Region.
- AWS managed policy update – added policies AWSElasticDisasterRecoveryRecoveryInstancePolicy and AWSElasticDisasterRecoveryLaunchActionsPolicy to support post-launch actions.

August 2023

- Added support for Amazon Linux 2023.
- Source Networks – Added support for replicating Route Tables.

July 2023

- Service launch in the following regions: Europe (Zurich), Europe (Spain), Asia Pacific (Hyderabad), Australia (Melbourne), and Middle East (UAE) regions.
- Introduced a new feature: In-AWS Right Sizing, allowing you to easily replicate your EC2 instance and EBS volume types between AWS regions.

June 2023

- Introduced a new feature: Trusted accounts, allowing to quickly create roles for multiple accounts and providing visibility into existing permissions.
- AWS managed policy update – updated AWSElasticDisasterRecoveryAgentInstallationPolicy to support network replication and recovery.

May 2023

- Introduced a new feature: Network replication configurations, allowing you to easily replicate your existing source network configurations, saving time and resources and preventing security risks.
- New AWS managed policy – Added new policies: AWSElasticDisasterRecoveryCrossAccountReplicationPolicy policy and AWSElasticDisasterRecoveryNetworkReplicationPolicy policy.
- AWS managed policy updates – Updated the AWSElasticDisasterRecoveryRecoveryInstancePolicy policy, the AWSElasticDisasterRecoveryEc2InstancePolicy policy, the AWSElasticDisasterRecoveryAgentPolicy policy, the AWSElasticDisasterRecoveryServiceRolePolicy policy, and the AWSElasticDisasterRecoveryConsoleFullAccess policy.

April 2023

- Introducing a new feature: Launch settings management, allowing to configure default launch settings that apply to newly added source servers and the ability to update multiple servers' settings.
- AWS managed policy updates – AWSElasticDisasterRecoveryAgentPolicy and AWSElasticDisasterRecoveryConsoleFullAccess.

March 2023

- Introduced a new feature: automated replication of new disks Introduced a new feature: support for Oracle ASM Filter Driver

February 2023

- Introduced a new feature: MAP 2.0 Auto Tagging

December 2022

- New AWS managed policy – Added the AWSElasticDisasterRecoveryStagingAccountPolicy_v2 policy.

November 2022

- Added support for cross-Region failback and cross-Availability-Zone recovery. Learn more about cross-Region failback and cross-Availability-Zone recovery.
- AWS managed policy update – updated AWSElasticDisasterRecoveryAgentInstallationPolicy for Replication Agent reinstallation on recovery instance.

October 2022

- AWS managed policy update – AWSElasticDisasterRecoveryRecoveryInstancePolicy.

September 2022

- Service launch in Asia Pacific (Jakarta) Region.

June 2022

- Service launch in the following regions: US West (N. California), Africa (Cape Town), Asia Pacific (Hong Kong), Asia Pacific (Mumbai), Asia Pacific (Osaka), Asia Pacific (Seoul), Canada (Central),

Europe (Milan), Europe (Paris), Europe (Stockholm), Middle East (Bahrain), and South America (São Paulo).

- AWS managed policy update – Updated several policies: AWSElasticDisasterRecoveryAgentInstallationPolicy, AWSElasticDisasterRecoveryFailbackInstallationPolicy, AWSElasticDisasterRecoveryServiceRolePolicy, and AWSElasticDisasterRecoveryReplicationServerPolicy.

May 2022

- AWS managed policy update – Updates the AWSElasticDisasterRecoveryConsoleFullAccess policy and the AWSElasticDisasterRecoveryReadOnlyAccess policy.

April 2022

- AWS managed policy update – Updates the AWSElasticDisasterRecoveryAgentPolicy policy.

March 2022

- Added support for no rescan for all Windows operating systems and certain Linux operating systems. Learn more about the no-rescan feature.

February 2022

- New AWS managed policy – Added the AWSElasticDisasterRecoveryStagingAccountPolicy.

January 2022

- Added support for failback automation.

November 2021

- New AWS managed policy – Added several policies:
  - AWSElasticDisasterRecoveryStagingAccountPolicy

- AWSElasticDisasterRecoveryAgentPolicy
- AWSElasticDisasterRecoveryConversionServerPolicy
- AWSElasticDisasterRecoveryFailbackPolicy
- AWSElasticDisasterRecoveryFailbackInstallationPolicy
- AWSElasticDisasterRecoveryConsoleFullAccess
- AWSElasticDisasterRecoveryReplicationServerPolicy
- AWSElasticDisasterRecoveryRecoveryInstancePolicy
- AWSElasticDisasterRecoveryServiceRolePolicy

AWS Elastic Disaster Recovery Client Release Notes

AWS Elastic Disaster Recovery is constantly releasing new agent versions; the versions have major and minor version numbers. We recommend to upgrade when a new major version is out, and also once every 1-2 months, to catch up on important fixes deployed through the minor versions. Releases are staggered, and may not be available in all supported regions simultaneously.

What's in a Release?

AWS Elastic Disaster Recovery releases can contain changes of varying scale, impact, and severity.

- **miscellaneous bug fixes** - contain non-material bug fixes or enhancements within the software components.
- **miscellaneous security enhancements** - provide tangible security enhancements that address vulnerabilities within the software package or subcomponents/dependencies.
- **miscellaneous performance enhancements** - provide non-material improvements to component responsiveness, reliability, and/or resilience.

Agent Version History

The following table describes the released versions of the AWS Elastic Disaster Recovery Agent. See the agent reinstallation instructions on how to upgrade the Replication Agent to the latest version.

What's in a Release?

Agent Version	Details	Release date
6.42.25	• Added support for Oracle Linux 10. See Supported Linux operating systems. • The Failback Client no longer prompts for a custom S3 endpoint. It resolves Amazon S3 using the DNS configuration of the server that you are failing back to. • Updates the bundled OpenSSL version to 3.0.22 to address security vulnerabilities: CVE-2026-63072, CVE-2026-63076, CVE-2026-54874, CVE-2026-63074, and CVE-2026-75803. • Updates the bundled Bouncy Castle libraries to version 1.85 to address security vulnerabilities, including CVE-2026-8763, CVE-2026-59650, CVE-2026-58062, CVE-2026-59639, CVE-2026-12802, CVE-2026-59642, and CVE-2026-59645.	9 October 2026
6.42.24	• Updates the bundled OpenSSL version to 3.0.21 to address security vulnerabilities: CVE-2026-45447, CVE-2026-45446, CVE-2026-34180, CVE-2026-7383, and CVE-2026-9076. • Updates the bundled Jackson libraries to version 2.18.8 to address security vulnerabilities: CVE-2026-54512 and CVE-2026-54513. • Updates the bundled Apache Log4j libraries to version 2.25.5 to address security vulnerability CVE-2026-49844.	July 2026
6.42.23	• Added support for RHEL 10.1. See Supported Linux operating systems. • Fixes a security issue to mitigate CVE-2025-66566. • Updates the Python for Windows version to 3.14.6 to address security vulnerabilities.	25 June 2026
6.42.21	• Added support for Microsoft Windows Server 2025 64-bit as a source server operating system. See Supported Windows operating systems.	5 June 2026

Agent Version	Details	Release date
6.42.20	• Added support for Fast recovery mode. For more informati on, see Editing the default AWS DRS launch settings.	30 April 2026
6.42.19	• Security fix to mitigate CVE-2025-15467 for Windows	17 April 2026
6.42.15	• Security fix to mitigate CVE-2025-15467 for Linux	11 February 2026
6.42.14	• Security fix to mitigate CVE-2025-9230 for Windows	9 February 2026
6.42.13	• Added support for Amazon Linux 2023 kernel 6.12 • Added support for RHEL 9.7 • Added support for Ubuntu 24.04 LTS • Added support for Rocky Linux 9.7	14 January 2026
6.41		17 February 2025
6.40	• Added support for RHEL 9.5	25 January 2025
6.39.3	• Optimized Failback workflow when performing multi-acc ount recovery between opt-in and default AWS Regions. • Miscellaneous performance enhancements.	17 December 2024
6.39.2	• Miscellaneous performance enhancements	9 December 2024
6.39.1	• Miscellaneous performance enhancements	24 November 2024
6.39.0	• Added support for Oracle 9.0-9.4.	22 October 2024

Agent Version	Details	Release date
6.37.0	• Automatic disk detection - improved detection for some edge cases. • Fixed issues preventing AWS Replication Agent installation on CentOS 5.11 with `scl` enabled • Miscellaneous performance enhancements	24 September 2024
6.34.0	• Miscellaneous performance enhancements	28 August 2024
6.32.0	• Miscellaneous bug fixes.	21 August 2024
6.31.0	• The AWS Replication Agent prevents reinstallation to a different AWS Region if already replicating to an AWS Region. • To change the AWS Region a Source Server is replicating to, first disconnect, then delete the original Source Server.	18 August 2024
6.29.0	• Fixed issue preventing the AWS Replication Agent from starting after a kernel upgrade was performed.	11 August 2024
6.28.0	• Fixed issue resulting in incorrect Instance type right sizing recommendations in legacy Windows Source Servers.	7 August 2024
6.27.0	• Miscellaneous security enhancements.	4 August 2024
6.26.0	• Miscellaneous security enhancements.	29 July 2024
6.25.0	• Fixed issue preventing the AWS Replication Agent from identifying the correct primary IP Address of a Source Server when Docker was installed. • Miscellaneous security enhancements.	25 July 2024
6.23.0		18 July 2024

Agent Version	Details	Release date
6.21.0	• Miscellaneous security enhancements.	15 July 2024
6.20.0	• Miscellaneous security enhancements.	6 July 2024
6.19.0	• Miscellaneous performance enhancements.	3 July 2024
6.18.0	• Miscellaneous security enhancements.	30 June 2024
6.17.0	• Miscellaneous security enhancements.	26 June 2024
6.16.0	• Miscellaneous security enhancements.	24 June 2024
6.16.0	• Miscellaneous security enhancements. • Miscellaneous performance enhancements.	23 June 2024
6.14.0	• Miscellaneous security enhancements.	19 June 2024
6.11.0	• Miscellaneous security enhancements.	15 June 2024
6.10.0	• Fixed an issue preventing the installation of the AWS Replication Agent on Linux Source Servers when over 100 storage devices were present. • Miscellaneous security enhancements.	8 June 2024
6.8.0	• Added support for installing the AWS Replication Agent on Ubuntu 24.04 with kernel 6.8. • Miscellaneous security enhancements.	2 June 2024
6.7.0	• Fixed an issue preventing failback to on-premises when the Recovery Instance was using a Linux Operating System. • Miscellaneous bug fixes.	29 May 2024
6.6.0	• Added support for protecting up to 63 volumes on a Source Server.	18 May 2024

Agent Version	Details	Release date
6.5.0	• Fixed issue preventing Replication Agent installation on Linux Source Servers when a debug kernel is present.	14 May 2024
6.3.0	• Miscellaneous security enhancements.	8 May 2024
6.2.0	• (Re-Release) Fixed issue preventing in-place agent upgrades from installations prior to v5.31.	24 April 2024
6.1.0	• Miscellaneous bug fixes. • Miscellaneous security enhancements.	18 April 2024
6.0.0	• Miscellaneous bug fixes. • Fixed an issue preventing some services from correctly starting after a drill in CentOS 6.	12 April 2024
5.37.0	• Fixed issue preventing in-place agent upgrades from installations prior to v5.31.	30 March 2024
5.32.0	• Miscellaneous bug fixes.	11 March 2024

Agent Version	Details	Release date
5.31.0	• Miscellaneous security enhancements. • Fixed issue that would lead to a BSOD in Windows Server if the Source Server was migrated to AWS with Application Migration Service. • Removed support for Windows Server 2008 R2 with the `AwsReplicationWindowsInstaller.exe` installer. **Important** New agent installations and existing agent upgrades for Windows Server 2008 R2 should use `AwsReplicationWindowsLegacyInstaller.exe` going forward.	8 March 2024
5.28.0	• Fixed an issue preventing the unmounting of LVM devices on source servers after the AWS Replication Agent was installed.	27 February 2024
5.26.1	• Fixed issue preventing installation when CIFS mounts contained spaces on source servers.	21 February 2024
5.26.0	• Miscellaneous bug fixes.	18 February 2024
5.25.0	• Miscellaneous security enhancements.	12 February 2024
5.24.0	• Miscellaneous bug fixes.	7 February 2024

Agent Version	Details	Release date
5.22.0	• Added support for protecting EC2 Instances with Marketplace Licenses. • Added support for Linux Kernel 6.6 and 6.7. • Fixed issue preventing installation on Linux when using UEFI boot with no grub.cfg present.	28 January 2024
5.20.0	• Miscellaneous bug fixes.	8 January 2024
5.19.0	• Fixed an issue preventing AWS-to-AWS failback from starting when using both `--devices` and `--force-volumes` flags.	27 December 2023
5.18.0	• Fixed issue preventing Agent installation when over 50 disks were attached to the Source Server. • Note: DRS can only replicate up to 50 disks.	19 December 2023
5.17.0	• Minor bug fixes.	18 December 2023
5.15.0	• Added support for replicating to GovCloud Regions. • Fixed issue preventing Copy private IP from functioning in certain Windows Server Portuguese localizations. • Fixed issue that would result in a timeout starting post-launch services on certain Linux configurations. • Minor bug fixes.	10 December 2023
5.14.1	• Fixed issue preventing a Source Server from entering a Healthy data replication status after a reboot.	22 November 2023
5.14.0	• Fixed issue preventing in-AWS failback from succeeding in multi-account scenarios. • Fixed issue preventing SuSE 15 SP3 from successfully launching in Xen-based EC2 Instance Types.	18 November 2023

Agent Version	Details	Release date
5.13.0	• Fixed issue preventing no-rescan-on-reboot from functioning in Ubuntu 22.04.	18 November 2023
5.12.0	• Fixed issue preventing post boot conversion scripts from executing in RHEL 6 installations. • Fixed issue resulting in multiple initrd files being present on Source Servers.	8 November 2023
5.11.0	• Miscellaneous bug fixes.	2 November 2023
5.10.0	• Fixed issue preventing in-place Linux agent upgrades when proxy settings are present.	24 October 2023
5.9.0	• Replicating EC2 instances that were launched with marketplace product codes is no longer supported. • Fixed issue impacting initramfs generation in some Linux versions.	22 October 2023
5.8.0	• Miscellaneous bug fixes.	8 October 2023
5.7.0	• Miscellaneous bug fixes.	27 September 2023
5.6.0	• Installation fixes for SLES15 with UEFI. • Added support for no-rescan-on-reboot on Amazon Linux 2023.	14 September 2023
5.5.0	• Added support for Amazon Linux 2023 (See Supported Operating Systems). • Miscellaneous bug fixes.	7 September 2023

Agent Version	Details	Release date
5.4.0	• Fixed a rare issue preventing OS boot if a reboot was performed immediately after a kernel update. • Added support for Oracle Linux 8.7/8.8 (See Supported Operating Systems). • Miscellaneous bug fixes.	7 September 2023
5.2.0	• Fixed rare issue that could result in a kernel panic in older Linux Kernels.	2 August 2023
5.1.0	• Fixed issue where multipath devices may not be properly detected. • Miscellaneous bug fixes.	26 July 2023
5.0.0	• Fixed issue preventing replication after a kernel upgrade was performed. • Miscellaneous bug fixes and security enhancements.	26 July 2023
4.11.0	• Enhancements for operating systems with btrfs. • Installer Enhancements.	22 June 2023
4.10.0	• Added support for kernel upgrades. • Fixed issue preventing replication after reinstallation on certain older operating systems. • Miscellaneous bug fixes.	19 May 2023
4.8.0	• Added support for no-rescan-on-reboot for btrfs. • Miscellaneous bug fixes. • Installer Enhancements.	22 April 2023
4.7.0	• Introduced auto detection of new disks. • Miscellaneous bug fixes.	22 March 2023

Agent Version	Details	Release date
4.6.0	• Introduced support for Oracle ASM Filter Driver. • Miscellaneous bug fixes and security enhancements.	22 March 2023
4.5.0	• Miscellaneous bug fixes and security enhancements.	22 February 2023
4.4.0	• Added support for RHEL 8.7 (See Supported Operating Systems).	16 February 2023
4.3.0	• Miscellaneous performance enhancements.	12 February 2023
4.1.0	• Fixed issue preventing installation on RHEL 6 Operating Systems when using UEFI. • Fixed issue with replication stalling in Windows Servers with teamed network adapters. • Fixed issue with multipath drive detection in Ubuntu 20.04 and 22.04.	24 January 2023
3.7.0	• Miscellaneous bug fixes and performance enhancements.	28 December 2022
3.6.0	• Miscellaneous bug fixes and security enhancements.	13 December 2022

Agent Version	Details	Release date
3.4.0	• Fixed issue preventing syncing Source Servers in some foreign language Windows Server installation. • Fixed issue allowing installation without necessary flags present. • Fixed issue preventing installation via SSM. • Fixed issue preventing installation on SLES12 SP1. • Added support for no-rescan-on-reboot for RHEL 9. • Fixed issue resulting in occasional rescan after agent reinstall ation on CentOS 6. • Fixed issue with no-rescan-on-reboot on CentOS 6.	20 October 2022
3.3.0	• Miscellaneous performance enhancements.	19 September 2022
3.2.0	• Added support for RHEL 5 and CentOS 5 (See Supported Operating Systems). • Miscellaneous bug fixes and performance enhancements.	6 September 2022
3.1.0	• Fixed issue preventing agent startup on CentOS. • Miscellaneous bug fixes and performance enhancements.	6 September 2022
3.0.0	• Added support for legacy Windows Server operating systems (See Supported Operating Systems). • Added proxy support for Linux. • Miscellaneous bug fixes and performance enhancements.	6 September 2022

Failback Client Version History

The following table describes the released versions of the AWS Elastic Disaster Recovery Failback Client. The latest version of the Failback Client can be downloaded by following the failback instructions.

What's in a Release?

Client Version	Details	Release date
6.42.25	• Removed support for specifying a custom S3 endpoint. The Failback Client now resolves Amazon S3 using the DNS configuration of the server that you are failing back to.	9 October 2026
6.21.0	• Miscellaneous security enhancements.	15 July 2024
5.21.0	• Fixed issue preventing custom DRS Endpoints from being used during failback.	13 January 2024
5.20.0	• Fixed an issue preventing failback from starting when using static IP configurations and specifying an S3 Endpoint. • Fixed an issue where the failback client may return `ERROR: Manifest download timed out.` during initial configura tion.	8 January 2024
5.13.0	• Fixed issue preventing failback client from starting when configured with 4GB of memory.	18 November 2023
5.8.0	• Fixed issue preventing downloading replicator software to the failback client.	8 October 2023
5.5.0	• Fixed issues preventing specifying S3 endpoints while using static IP addresses.	7 September 2023
5.3.0	• Miscellaneous bug fixes.	7 September 2023
5.2.0	• Fixed issue preventing automatic disk mappings during failback. • Miscellaneous manual disk mapping improvements.	2 August 2023
4.12.0	• Fixed issue preventing failback to volumes of different sizes.	6 July 2023

Client Version	Details	Release date
4.11.0	• Fixed issue preventing the failback client from detecting some physical disk configurations. • Fixed issue preventing failback to volumes larger than the replica. • Miscellaneous manual disk mapping improvements.	22 June 2023
4.5.0	• Miscellaneous bug fixes.	22 February 2023
4.1.0	• Fixed issue preventing VMware tools from being enabled on failback.	24 January 2023
3.6.0	• Miscellaneous bug fixes.	13 December 2022

DRSFA Version History

The following table describes the released versions of the DRSFA Client. The latest version of the DRSFA Client can be downloaded by following the DRSFA installation instructions.

What's in a Release?

Client Version	Details	Release date
5.2.0	• Fixed issue preventing automatic disk mappings during failback. • Miscellaneous manual disk mapping improvements.	2 August 2023
3.6.0	• Miscellaneous performance enhancements.	13 December 2022
3.1.0	• Fixed issue with failback where VMware VMs would not eject ISOs via cdrom.	20 August 2022

Client Version	Details	Release date
	• Fixed issue with failback when a VMware VM had an ISO already mounted via cdrom.	

Document history for the AWS Elastic Disaster Recovery User Guide

The following are the latest documentation updates for AWS Elastic Disaster Recovery. We update the documentation frequently to address the feedback that you send us.

Change	Description	Date
CloudEndure Disaster Recovery discontinued	AWS discontinued CloudEndure Disaster Recovery on September 29, 2025. This update removes the upgrade guidance from CloudEndure Disaster Recovery to AWS Elastic Disaster Recovery. For more information about CloudEndure Disaster Recovery, see Can I upgrade from CloudEndure Disaster Recovery to AWS Elastic Disaster Recovery?.	October 14, 2026
AWS Graviton source servers	Added documentation for replicating and recovering source servers that run on AWS Graviton (`arm64`) Amazon EC2 instances. For more information, see AWS Graviton source servers.	September 25, 2026

Change	Description	Date
Recovery plans	Added documentation for Orchestrating recovery with recovery plans, which recover groups of source servers in a defined order with wait times between groups.	August 18, 2026
Updated AWS managed policy	Updates the AWSElasticDisasterRecoveryReadOnlyAccess policy with new read-only permissions for recovery plans and recovery plan executions.	August 18, 2026
Amazon EBS volume initialization rate passthrough	AWS Elastic Disaster Recovery now preserves and passes through `VolumeInitializationRate` values set on EC2 launch template block device mappings during drill and recovery launches. For more information, see Key considerations for EC2 launch templates.	July 10, 2026

Change	Description	Date
Updated AWS managed policies	Updated the AWSElasticDisasterRecoveryConsoleFullAccess_v2 and AWSElasticDisasterRecoveryLaunchActionsPolicy policies to add permissions for SSM documents with the `AWSDRS-` prefix.	June 18, 2026
Updated AWS managed policies	Created new revisions of managed policies to support a change in SSM. AWSElasticDisasterRecoveryConsoleFullAccess_v2 AWSElasticDisasterRecoveryLaunchActionsPolicy_v2	July 3, 2025

Change	Description	Date
Updated AWS managed policies	Created new revisions of managed policies to support a change in authentication with EBS APIs. AWSElasticDisasterRecoveryServiceRolePolicy AWSElasticDisasterRecoveryConsoleFullAccess_v2 AWSElasticDisasterRecoveryConsoleFullAccess	January 6, 2025
AWS Elastic Disaster Recovery - Flexible Instance Types feature	Release of Flexible Instance Types in AWS Elastic Disaster Recovery (AWS DRS). For more information, see Flexible Instance Types	July 30, 2024

Change	Description	Date
Updated AWS managed policies	Created new revisions of managed policies to support additiona l parameter types in SSM Parameter Store for post-launch actions. The following managed policies were updated: AWSElasticDisaster RecoveryConsoleFul lAccess_v2 AWSElasticDisaster RecoveryLaunchActi onsPolicy	May 19, 2024
AWS Elastic Disaster Recovery Support for AWS Outposts	AWS Elastic Disaster Recovery now supports AWS Outposts. For more information see: Working with AWS DRS and Outposts.	April 18, 2024
Updated AWS managed policy	Created a new revision of the AWSElasticDisaster RecoveryCrossAccou ntReplicationPolic y policy, to support replicating marketpla ce licenses to launched instances.	January 28, 2024

Change	Description	Date
Updated AWS managed policy	Created new revision of the AWSElasticDisasterRecoveryServiceRolePolicy policy, to support replicating marketplace licenses to launched instances.	January 28, 2024
Updated AWS managed policies	Updated policies to support managed prefix lists for network replication and recovery. The following managed policies were updated: AWSElasticDisasterRecoveryNetworkReplicationPolicy AWSElasticDisasterRecoveryServiceRolePolicy	January 3rd, 2024

Change	Description	Date
Updated AWS managed policies	Created new revisions of managed policies to support DRS to GovCloud and added Sid to statements in managed policies. The following managed policies were updated: AWSElasticDisasterRecoveryAgentPolicy AWSElasticDisasterRecoveryAgentInstallationPolicy AWSElasticDisasterRecoveryEc2InstancePolicy AWSElasticDisasterRecoveryConsoleFullAccess AWSElasticDisasterRecoveryLaunchActionsPolicy AWSElasticDisasterRecoveryNetworkReplicationPolicy AWSElasticDisasterRecoveryRecoveryInstancePolicy AWSElasticDisasterRecoveryServiceRolePolicy	November 27, 2023

Change	Description	Date
	AWSElasticDisasterRecoveryConversionServerPolicy AWSElasticDisasterRecoveryFailbackPolicy AWSElasticDisasterRecoveryFailbackInstallationPolicy AWSElasticDisasterRecoveryStagingAccountPolicy_v2 AWSElasticDisasterRecoveryStagingAccountPolicy AWSElasticDisasterRecoveryReplicationServerPolicy	
Updated AWS managed policy	Created new revision of AWSElasticDisasterRecoveryCrossAccountReplicationPolicy policy to support DRS in GovCloud	November 27, 2023
Updated AWS managed policy	Updated the AWSElasticDisasterRecoveryReadOnlyAccess policy to support describing additional post-launch actions.	November 27, 2023

Change	Description	Date
New AWS managed policy	Added the AWSElasticDisasterRecoveryConsoleFullAccess_v2 policy to support running predefined post-launch actions from the console.	November 27, 2023
Updated AWS managed policy	Updated the AWSElasticDisasterRecoveryServiceRolePolicy policy to support managed prefix lists for network replication and recovery.	October 15, 2023
Updated AWS managed policy	Updated the AWSElasticDisasterRecoveryConsoleFullAccess and AWSElasticDisasterRecoveryLaunchActionsPolicy policies to support recovery into an existing instance.	October 15, 2023
Updated AWS managed policy	Updated the AWSElasticDisasterRecoveryEc2InstancePolicy policy to allow sending installation result metrics to AWS Elastic Disaster Recovery.	October 10, 2023

Change	Description	Date
Updated AWS managed policy	Updated the AWSElasticDisasterRecoveryAgentInstallationPolicy policy to allow sending installation result metrics to AWS Elastic Disaster Recovery.	October 10, 2023
New AWS managed policy	Added the AWSElasticDisasterRecoveryLaunchActionsPolicy policy to support post-launch actions.	September 13, 2023
Updated AWS managed policy	Updated the AWSElasticDisasterRecoveryReadOnlyAccess policy to support post-launch actions.	September 13, 2023
Updated AWS managed policy	Updated the AWSElasticDisasterRecoveryEc2InstancePolicy policy to support network replication and recovery.	June 13, 2023
Updated AWS managed policy	Updated the AWSElasticDisasterRecoveryAgentInstallationPolicy policy to support network replication and recovery.	June 13, 2023

Change	Description	Date
Updated AWS managed policy	Updated the AWSElasticDisasterRecoveryConsoleFullAccess policy to support network replication and recovery.	June 13, 2023
New AWS managed policy	Updated the AWSElasticDisasterRecoveryNetworkReplicationPolicy policy to support network replication and recovery.	June 13, 2023
Updated AWS managed policy	Updated the AWSElasticDisasterRecoveryServiceRolePolicy policy to support network replication and recovery.	June 13, 2023
New AWS managed policy	Added the AWSElasticDisasterRecoveryCrossAccountReplicationPolicy policy to support cross-account failback.	May 7, 2023
Updated AWS managed policy	Updated the AWSElasticDisasterRecoveryRecoveryInstancePolicy policy to support cross-account failback by the agent after reverse replication.	May 7, 2023

Change	Description	Date
Updated AWS managed policy	Updated the AWSElasticDisasterRecoveryEc2InstancePolicy policy to support cross-account replication by the agent.	May 7, 2023
Updated AWS managed policy	Updated the AWSElasticDisasterRecoveryConsoleFullAccess policy to support default EC2 launch templates and bulk editing of source server EC2 launch templates.	April 19, 2023
Updated AWS managed policy	Updated the AWSElasticDisasterRecoveryAgentPolicy policy to support the kernel upgrade feature.	April 1, 2023
New AWS managed policy	Added the AWSElasticDisasterRecoveryStagingAccountPolicy_v2 policy to support the recovery of source servers into a separate target account and to allow failing back.	December 11, 2022
Cross-Region failback and cross-Availability-Zone recovery	Added support for cross-Region failback and cross-Availability-Zone recovery.	November 27, 2022