

Egress Security Checklist

This checklist summarizes key considerations and best practices for implementing robust egress access controls in your Kubernetes environment using Calico Enterprise or Calico Cloud, drawing on the principles of secure policy design.

I. Mandatory Security Posture

Establish a strong default security baseline.

- ☐ **Implement Global Default-Deny:** Establish a **global default-deny policy for all Ingress and egress traffic** as a first-order security and compliance requirement. Ensure it explicitly excludes critical system pods and allows necessary DNS traffic.
- ☐ **Adopt Policy Tiers:** Utilize Calico's hierarchical policy tiers (e.g., **security** tier for global egress rules, **application** tier for microsegmentation) to manage precedence and delegate granular controls effectively.
- ☐ **Centralize Egress Management:** Centralize the implementation and management of egress security policies to optimize the cluster's security posture and simplify operations.
- ☐ **Design for Native Compliance:** Build a scalable security design that accommodates Day-1 and Day-N enforcement, ensures native compliance of applications, and supports real-time enforcement.

II. Policy Design & Implementation

Build scalable, precise policies using labels, network sets, and FQDN-based rules.

- ☐ **Establish Label Standard:** Define and enforce a **robust label standard** across your environment. Ensure all pods, hosts, and network sets are consistently and appropriately labeled to enable precise policy targeting.
- ☐ **Leverage Network Sets:** Group external IP addresses and domain names into **NetworkSet** or **GlobalNetworkSet** resources for centralized management, improved visibility in logs, and simplified policy creation.
- ☐ **Implement Granular Controls:** Define granular egress controls for your application microservices, aligning with their communication patterns.
- ☐ **Utilize FQDN-based Policies (Calico Enterprise/Cloud):** Employ Fully Qualified Domain Name (FQDN) based egress security policies to target external services that use dynamic IP addresses, ensuring scalability and preventing backdoors.
- ☐ **Integrate Threat Intelligence (Calico Enterprise/Cloud):** Configure threat intelligence feeds and network policies to automatically **deny egress access to known malicious destinations**, enhancing defense against cyber threats.
- ☐ **Ensure Automated Enforcement:** Design for automatic enforcement of egress policies, especially for dynamically provisioned resources like VMs during autoscaling.

III. Egress Gateways (Calico Enterprise/Cloud Specific)

Manage outbound IPs and control external access using egress gateways for compliance and visibility.

- ☐ **Assign Static Egress IPs:** Use egress gateways to assign **fixed, controlled source IP addresses** to specific applications, namespaces, or deployments for outbound traffic, particularly for integration with external firewalls or services with strict IP whitelisting.
- ☐ **Provision Dedicated IP Pools:** Create dedicated IP pools for egress gateways, including **AWS-backed IP pools** for native VPC integration, to control the range of source IPs used. Ensure CIDRs are reserved in the AWS fabric to avoid conflicts.
- ☐ **Configure ENI Modes (AWS):** Choose between **ENI-per-workload** and **Secondary-IP-per-workload** modes based on your instance types, workload churn rates, and IP address allocation needs.
- ☐ **Plan for Load Balancing & Failover:** Deploy **multiple egress gateways** per group for high availability, load balancing of outbound traffic, and fast failover. Optionally enable ECMP load balancing on client nodes if required.
- ☐ **Understand Policy Enforcement with Gateways:** Recognize that policy applied on the **client pod is most powerful** (sees original source/destination). Policies on the egress gateway pod have limitations regarding domain-based matching and source IP information.

IV. Operational Aspects

Govern, validate, and monitor egress policies to ensure secure, consistent operations.

- ☐ **Define Policy Lifecycle:** Establish a clear process and lifecycle for egress policies, covering access request, enforcement, and ongoing monitoring.
- ☐ **Design for Minimal Changes:** Create policies that require minimal updates by effectively using network sets and dynamic labels. Update network sets as external resource requirements change, and label pods as new ones need access.
- ☐ **Validate Policies Before Enforcement:** Always simulate and validate policies using tools like `netcat` or `curl`. For Calico Enterprise/Cloud, utilize staged policies to preview impact; for Calico Open Source, use the `log` action.
- ☐ **Implement Continuous Monitoring:** Monitor egress traffic and policy effectiveness using Calico's Policies board, Dynamic Service and Threat Graph, and custom Kibana dashboards to identify misconfigurations, denied flows, and traffic patterns.
- ☐ **Implement RBAC for Policy Management:** Define Role-Based Access Control (RBAC) to enable different personas (Security, Platform, Developers) to provision policies according to their privileges and responsibilities.
- ☐ **Ensure Security Governance:** Implement pull request policies and admission controllers in your CI/CD pipeline to ensure application manifests and container deployments adhere to authorized labels and security contexts.
- ☐ **Standardize and Centralize Policies:** Adopt a centralized and standard Kubernetes security policy model, potentially using a GitOps approach, to apply consistent controls across multiple cloud and on-premises clusters.

Now that you've reviewed the checklist, here's how to start putting it into practice:

✓ Your Next Steps Toward Egress Security

- 1 Review your current egress policies using this checklist to identify gaps in enforcement, DNS handling, and FQDN-based rules.
- 2 Identify and prioritize quick wins like implementing a global default-deny policy or standardizing labels to make immediate improvements.
- 3 Engage your security, platform, and development teams to align on roles, responsibilities, and ownership of policy enforcement.
- 4 Roll out changes in stages using Calico's policy tiers and simulation tools to validate policies before enforcement.
- 5 Monitor and optimize policies over time with observability tools like Calico's Dynamic Service & Threat Graph or custom Kibana dashboards to detect misconfigurations, denied flows, and traffic patterns.
- 6 Download the full eBook for step-by-step implementation guides, detailed examples, and advanced egress security strategies.

[Download the eBook](#)

About Tigera

[Tigera](#) provides Calico, a unified network security and observability platform to prevent, detect and mitigate security breaches in Kubernetes clusters. Tigera's open-source offering, [Calico Open Source](#), is the most widely adopted container networking and security solution.

Powering more than 100M containers across 8M+ nodes in 166 countries, Calico software is supported across all major cloud providers and Kubernetes distributions, and is used by leading companies including Discover, Chipotle, NBCUniversal, HanseMerkur, Box, Siemens Healthineers, Playtech, Royal Bank of Canada, and Bell Canada.

Questions or want to learn more?

Contact us to speak with a Kubernetes security expert.

[Contact Us](#)

Explore more at [tigera.io](#)

[Visit Website](#)