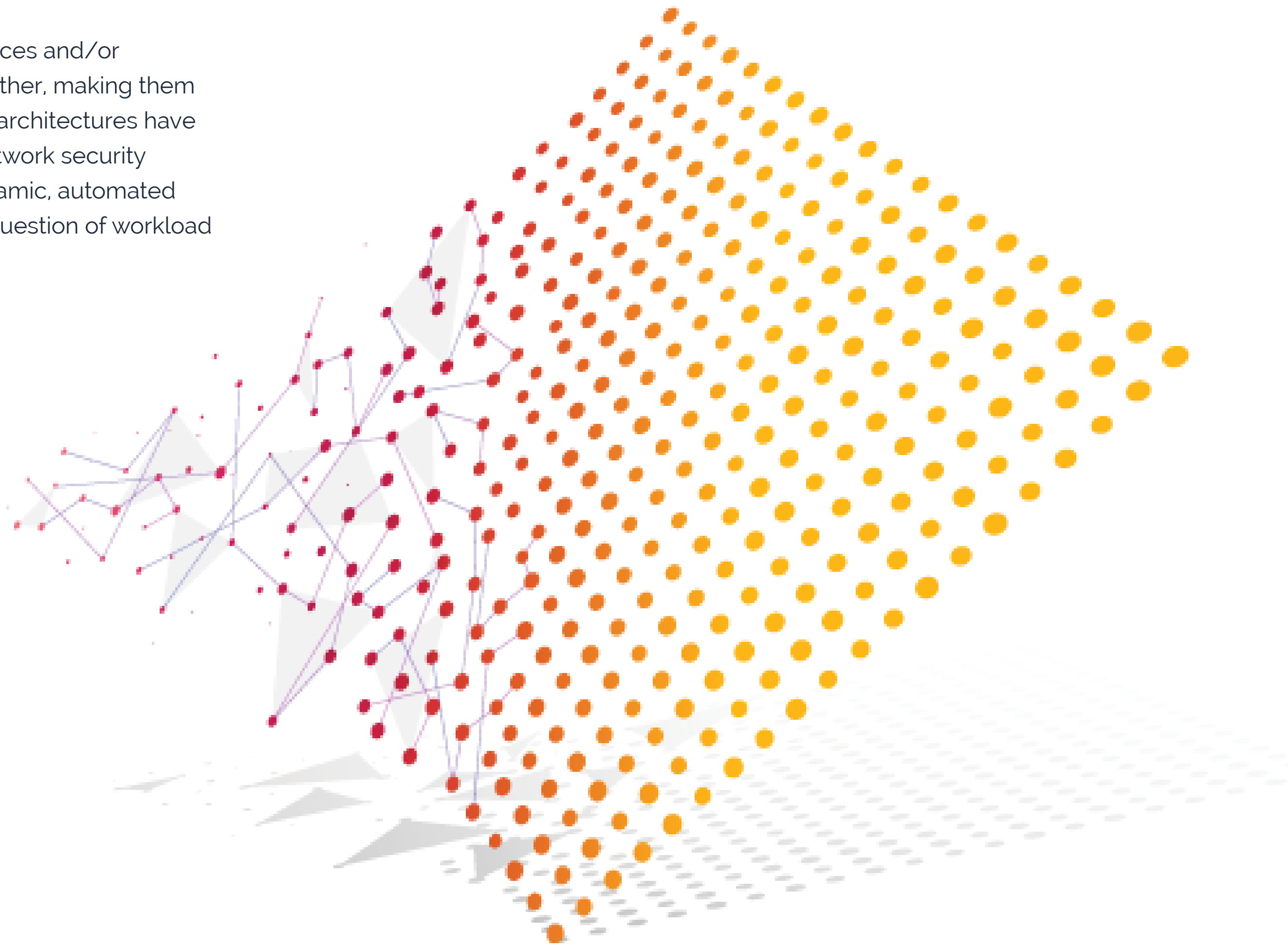
Tigera eBook:

Achieving network security and compliance for Kubernetes on AWS

Table of Contents

Introduction

Modern applications are being broken down into smaller pieces and/or architectures. Now, services are communicating with each other, making them dependent on the network to operate. These microservices architectures have significantly more components, meaning more extensive network security measures are needed. They are also deployed in highly dynamic, automated environments with ephemeral addresses which makes the question of workload identity a critical one for enforcing security.



Why organizations are adopting containers on the cloud

Containers provide a standard way to package your application's code, configurations, and dependencies into a single object. This object can be used to execute quick, reliable, and consistent deployments, regardless of the environment.

Ship more software faster

Standardize operations

Save money

Why leverage Kubernetes?

Kubernetes allows you to deploy and manage containerized applications at scale, using the same toolset for your on-premises and cloud environments. It manages clusters of compute instances and scheduling containers, so you can run containers with automated processes for deployment, maintenance, and scaling.

Run applications at scale

Kubernetes allows you to define complex containerized applications and run them across a cluster of servers to scale alongside your business

Run anywhere

Kubernetes empowers you to run highly available and scalable Kubernetes clusters on Amazon Web Services (AWS), while maintaining full compatibility with your on-premises deployments.

Seamlessly move applications

You can utilize the same tooling on-premises and on the cloud to move containerized applications from local development machines into production.

Add new functionality

As an open source project, new functionality is constantly being added to the platform by its large community of developers and technology companies.

Running Kubernetes on AWS

AWS provides infrastructure resources designed to run containers, as well as a set of orchestration services that make it easy for you to build and run containerized applications in production. Whether you want to manage the Kubernetes infrastructure yourself, or take advantage of fully managed services, you gain the security, scalability, and high-availability of AWS, with integrations to its powerful native services..

Amazon Elastic Compute Cloud (Amazon EC2)

For users who want to fully manage their own Kubernetes deployment, you can provision and run Kubernetes of your choice of powerful instance types. There are many open source projects that enable you to more easily run Kubernetes on Amazon EC2, such as Kubernetes Operations (kops).

Amazon Elastic Container Service for Kubernetes (Amazon EKS)

Amazon EKS provides you with fully managed Kubernetes infrastructure, so you can run Kubernetes without needing to provision or manage master instances and etcd.

Amazon Elastic Container Registry (Amazon ECR)

Amazon ECR enables you to store, encrypt, and manage container images for faster deployments

Use cases:

- Microservices
- Hybrid container deployments
- Batch processing
- Application migration

Why Kubernetes requires a new approach to network security and compliance

The vision behind Kubernetes is rapid application deployment, where services mesh development and test with production. This cannot truly be realized if security and networking teams must be brought into the fold for every application deployment.

Complex network security

Kubernetes heavily relies on the network and generates significant east-west traffic, creating a greater attack surface than traditional architectures, requiring more dynamic security and compliance controls.

Limited of visibility

Applications running on Kubernetes platforms are constantly changing IP addresses and locations, making it difficult to use traditional log flows to debug issues and investigate anomalous activity.

Complicated compliance measures

Dynamic application environments render periodic audits insufficient, as auditors require proof of network and security policy enforcement amid ever-changing conditions.

Enabling network security and compliance for Kubernetes on AWS with Tigera

Organizations running their workloads on AWS are responsible for anything running on top of cloud infrastructure, whereas AWS is responsible for the base-line infrastructure itself. Tigera complements native AWS services, delivering automated orchestration for containerized microservices, so developers can create scalable, secure, and reliable applications.

Zero trust network security

Visibility and traceability

Enterprise control and continuous compliance

Zero trust network security

Tigera operates under the assumption that your services, network, users – and anything else related to your environment – are potentially compromised. It delivers a layered defense model on top of your existing network to lock down your Kubernetes workloads, without requiring any code or configuration changes.

Multiple sources of identity

Tigera's Zero Trust Security model authenticates the identity of every request based on multiple sources, including the L3 network identity and x509 certificate-based cryptographic identity.

Multiple enforcements points

Tigera's declarative, intent-based policies are enforced at multiple points – including the host, container, and edge layers of the application.

Multiple layers of encryption

Encryption can be enabled for all traffic within and across all environments, leveraging mutual Transport Layer Security (mTLS) for application and edge layers and IPsec for traffic between hosts.

Zero trust network security

Business use case: AWS security groups and Kubernetes policy integration

When deploying Kubernetes on AWS, all Kubernetes pods have the same security groups as the host/node they are on (and vice versa).

AWS security groups are the standard approach to network security for Amazon Virtual Private Cloud (Amazon VPC) resources. Tigera serves as the model for the standard Kubernetes Network Policy, and natively integrates with AWS security groups, providing more fine-grained policy controls on top of your existing AWS policies. Combined, AWS and Tigera enable you to achieve a universal approach to security policies, securing your applications end-to-end.

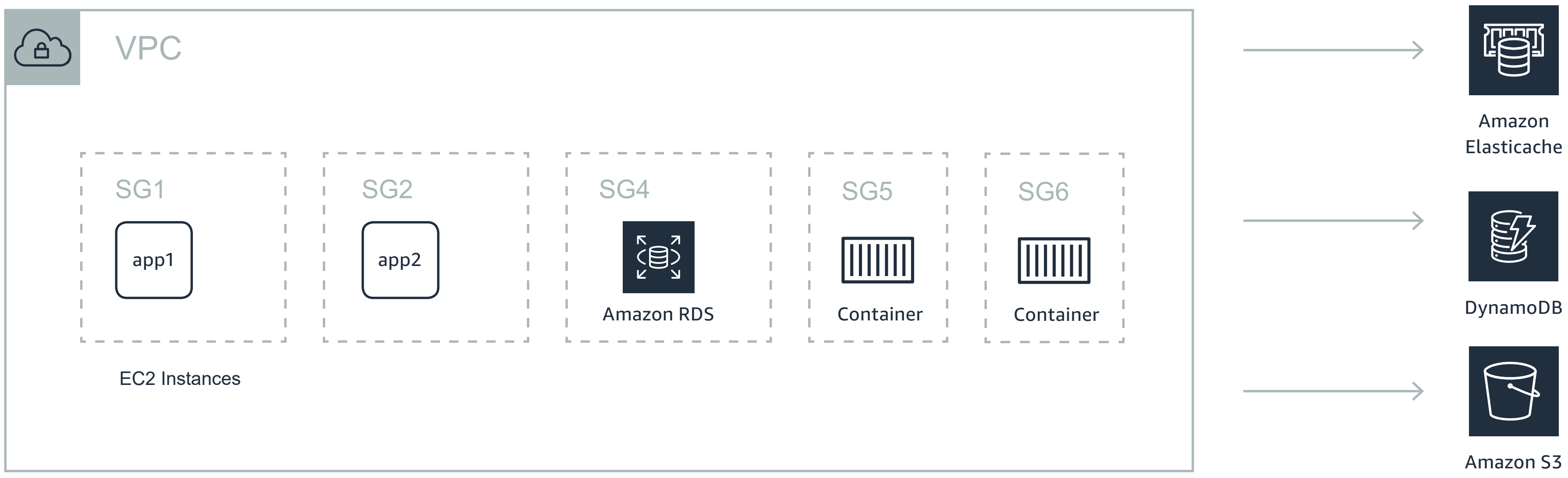


Figure 1: AWS security groups and Kubernetes policy integration

Visibility and traceability

Tigera integrates to Amazon CloudWatch, and other AWS services, for comprehensive visibility across your containerized environments. Some of the key capabilities including network visibility, compliance monitoring, and scanning of the network for illegitimate traffic and indicators of compromise (IoC).

Accurate network flow logging

Tigera captures flow logs with full context, including Kubernetes labels at the application and container interface, providing you with the data required for PCI, HIPAA, GDPR, and other compliance frameworks.

Security forensics for Kubernetes

Modern search and visualization capabilities provide real-time enterprise-wide visibility into Kubernetes traffic, empowering you to discover and identify communications between Kubernetes microservices to better support DevOps and Security teams' objectives.

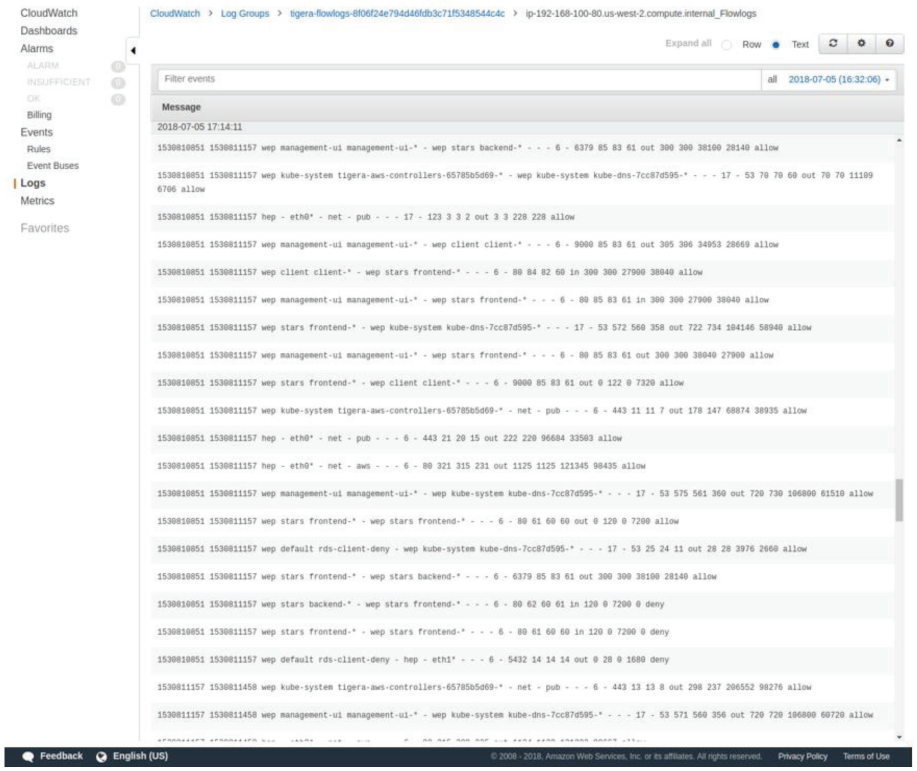
Automated remediation

Tigera is constantly monitoring traffic for anomalies, with built in alerting. In case of a compromise, it automatically remediates the issue by applying a quarantine to the container, prohibiting any lateral movement.

Visibility and traceability

Business use case: Visualization and CloudWatch network flow log

Tigera goes beyond traditional 5-tuple flow logs, which require additional context and correlation. It captures denied traffic at the container level, and appends workload metadata into the flow logs. For Kubernetes environments, such as Amazon EKS, Tigera generates bi-directional flow logs for all pods, and host connections, each including workload identity, pod labels, and host labels.


CloudWatch
Dashboards
Alarms
ALARM
INSUFFICIENT
OK
Billing
Events
Rules
Event Buses
Logs
Metrics
Favorites
CloudWatch > Log Groups > tigera-flowlogs-8f06f24e794d46fdb3c71f5348544c4c > ip-192-168-100-80.us-west-2.compute.internal_Flowlogs
Expand all
Row
Text
Filter events
all
2018-07-05 (16:32:06)
Message
Feedback
English (US)
© 2008 - 2018, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use

```
2018-07-05 17:14:11
1530810851 1530811157 wep management-ui management-ui-* - wep stars backend-* - - - 6 - 6379 85 83 61 out 300 300 38100 28140 allow
1530810851 1530811157 wep kube-system tigera-aws-controllers-65785b5d69-* - wep kube-system kube-dns-7cc87d595-* - - - 17 - 53 70 70 60 out 70 70 11109
6706 allow
1530810851 1530811157 hep - eth0* - net - pub - - - 17 - 123 3 3 2 out 3 3 228 228 allow
1530810851 1530811157 wep management-ui management-ui-* - wep client client-* - - - 6 - 9000 85 83 61 out 305 306 34953 28669 allow
1530810851 1530811157 wep client client-* - wep stars frontend-* - - - 6 - 80 84 82 60 in 300 300 27900 38040 allow
1530810851 1530811157 wep management-ui management-ui-* - wep stars frontend-* - - - 6 - 80 85 83 61 in 300 300 27900 38040 allow
1530810851 1530811157 wep stars frontend-* - wep kube-system kube-dns-7cc87d595-* - - - 17 - 53 572 560 358 out 722 734 104146 58940 allow
1530810851 1530811157 wep management-ui management-ui-* - wep stars frontend-* - - - 6 - 80 85 83 61 out 300 300 38040 27900 allow
1530810851 1530811157 wep stars frontend-* - wep client client-* - - - 6 - 9000 85 83 61 out 0 122 0 7320 allow
1530810851 1530811157 wep kube-system tigera-aws-controllers-65785b5d69-* - net - pub - - - 6 - 443 11 11 7 out 178 147 68874 38935 allow
1530810851 1530811157 hep - eth0* - net - pub - - - 6 - 443 21 20 15 out 222 220 96684 33503 allow
1530810851 1530811157 hep - eth0* - net - aws - - - 6 - 80 321 315 231 out 1125 1125 121345 98435 allow
1530810851 1530811157 wep management-ui management-ui-* - wep kube-system kube-dns-7cc87d595-* - - - 17 - 53 575 561 360 out 720 730 106800 61510 allow
1530810851 1530811157 wep stars frontend-* - wep stars frontend-* - - - 6 - 80 61 60 60 out 0 120 0 7200 allow
1530810851 1530811157 wep default rds-client-deny - wep kube-system kube-dns-7cc87d595-* - - - 17 - 53 25 24 11 out 28 28 3976 2660 allow
1530810851 1530811157 wep stars frontend-* - wep stars backend-* - - - 6 - 6379 85 83 61 out 300 300 38100 28140 allow
1530810851 1530811157 wep stars backend-* - wep stars frontend-* - - - 6 - 80 62 60 61 in 120 0 7200 0 deny
1530810851 1530811157 wep stars frontend-* - wep stars frontend-* - - - 6 - 80 61 60 60 in 120 0 7200 0 deny
1530810851 1530811157 wep default rds-client-deny - hep - eth1* - - - 6 - 5432 14 14 14 out 0 28 0 1680 deny
1530811157 1530811458 wep kube-system tigera-aws-controllers-65785b5d69-* - net - pub - - - 6 - 443 13 13 8 out 298 237 206552 98276 allow
1530811157 1530811458 wep management-ui management-ui-* - wep kube-system kube-dns-7cc87d595-* - - - 17 - 53 571 560 356 out 720 720 106800 60720 allow
```

Figure 2: Integration with CloudWatch for metrics and audit logs

Enterprise control and continuous compliance

Tigera delivers ongoing compliance and cross-functional collaboration capabilities, including the ability to extract data required for IT audits of Amazon EKS and other Kubernetes environments running on AWS.

Workload identity

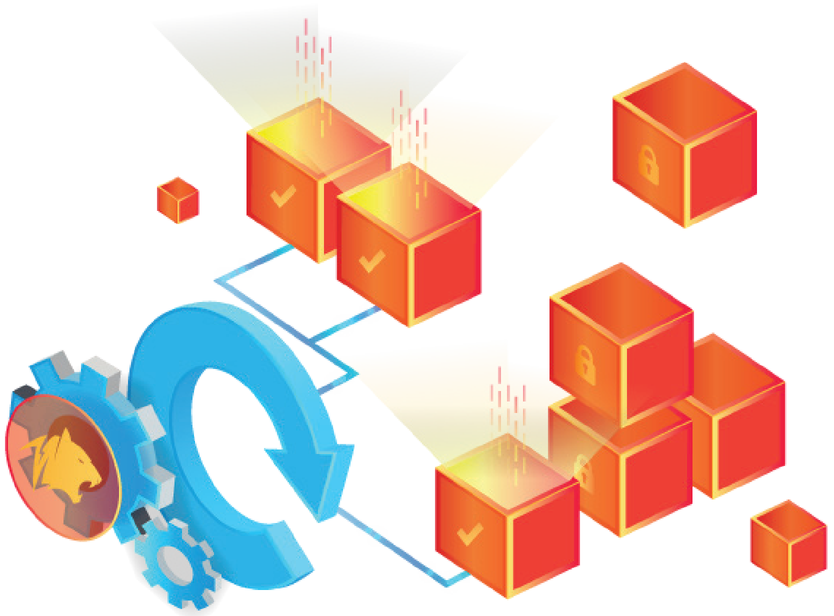
Workloads authenticate and authorize based on a series of attributes, including network and cryptographic identity (equivalent to two-factor authentication for workloads). All network flows are logged with the necessary workload identity and metadata information to demonstrate compliance with security policies.

Audit logging

All changes to security policies are logged. Combined with Tigera flow logging, you can quickly and easily demonstrate what policies are in place, and the history of enforcement.

Tiered security policies

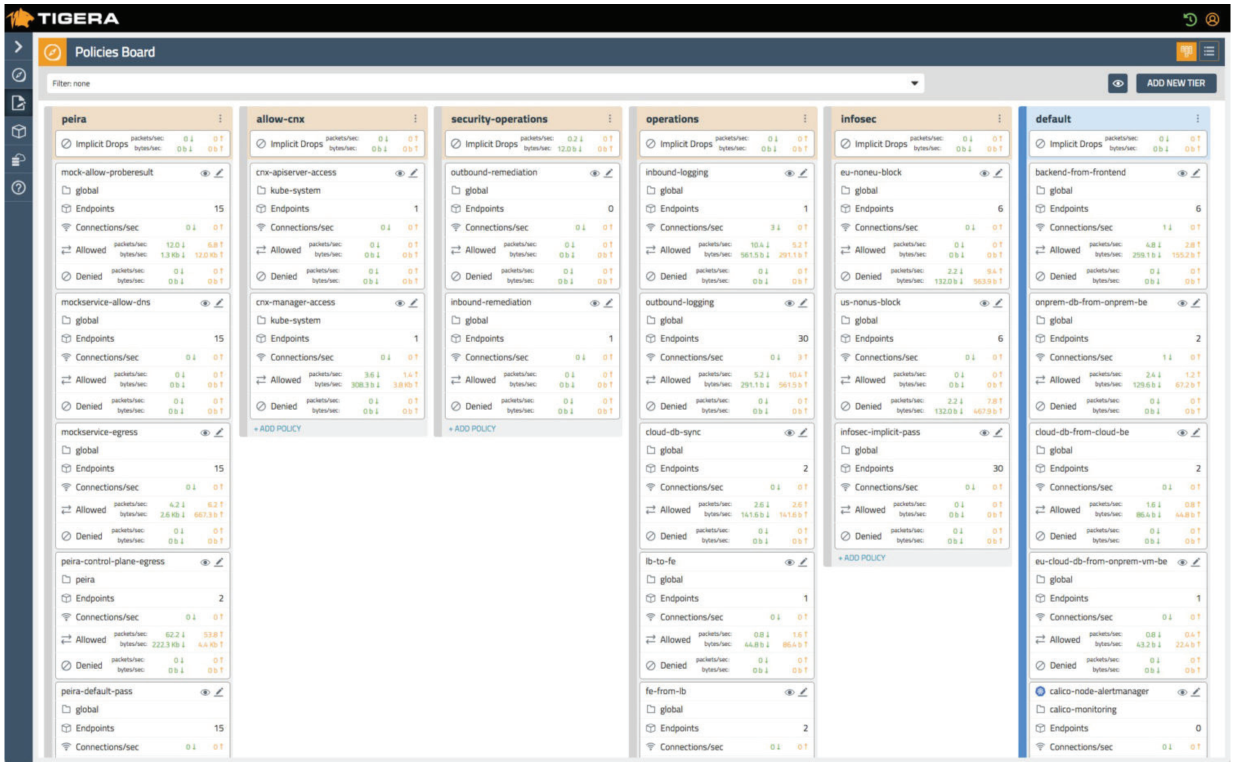
Tiered policy capabilities enable teams to collaborate on defining and implementing policies, without introducing dependencies on each other.



Enterprise control and continuous compliance

Business use case: Taking a tiered approach for security and compliance across teams

The flexibility of Tigera's tiered security approach makes it possible for your teams to concurrently set policies without changing others' workflows. For example, while your Information Security (InfoSec) team is creating policies preventing access for known bad actors, your networking team can prevent access between production and development nodes, and your application teams can define which services have access to other URLs and HTTP Methods. This yields more agile operations, with a greater security and compliance posture.



peira
Implicit Drops — packets/sec: 0↓ 0↑ — bytes/sec: 0 b↓ 0 b↑

mock-allow-proberesult
global
Endpoints 15
Connections/sec 0↓ 0↑
Allowed — packets/sec: 12.0↓ 6.8↑ — bytes/sec: 1.3 Kb↓ 12.0 Kb↑
Denied — packets/sec: 0↓ 0↑ — bytes/sec: 0 b↓ 0 b↑

mockservice-allow-dns
global
Endpoints 15
Connections/sec 0↓ 0↑
Allowed — packets/sec: 0↓ 0↑ — bytes/sec: 0 b↓ 0 b↑
Denied — packets/sec: 0↓ 0↑ — bytes/sec: 0 b↓ 0 b↑

mockservice-egress
global
Endpoints 15
Connections/sec 0↓ 0↑
Allowed — packets/sec: 4.2↓ 6.2↑ — bytes/sec: 2.6 Kb↓ 667.3 b↑
Denied — packets/sec: 0↓ 0↑ — bytes/sec: 0 b↓ 0 b↑

peira-control-plane-egress
peira
Endpoints 2
Connections/sec 0↓ 0↑
Allowed — packets/sec: 62.2↓ 53.8↑ — bytes/sec: 222.3 Kb↓ 4.4 Kb↑
Denied — packets/sec: 0↓ 0↑ — bytes/sec: 0 b↓ 0 b↑

peira-default-pass
global
Endpoints 15
Connections/sec 0↓ 0↑

allow-cnx
Implicit Drops — packets/sec: 0↓ 0↑ — bytes/sec: 0 b↓ 0 b↑

cnx-apiserver-access
kube-system
Endpoints 1
Connections/sec 0↓ 0↑
Allowed — packets/sec: 0↓ 0↑ — bytes/sec: 0 b↓ 0 b↑
Denied — packets/sec: 0↓ 0↑ — bytes/sec: 0 b↓ 0 b↑

cnx-manager-access
kube-system
Endpoints 1
Connections/sec 0↓ 0↑
Allowed — packets/sec: 3.6↓ 1.4↑ — bytes/sec: 308.3 b↓ 3.8 Kb↑
Denied — packets/sec: 0↓ 0↑ — bytes/sec: 0 b↓ 0 b↑

security-operations
Implicit Drops — packets/sec: 0.2↓ 0↑ — bytes/sec: 12.0 b↓ 0 b↑

outbound-remediation
global
Endpoints 0
Connections/sec 0↓ 0↑
Allowed — packets/sec: 0↓ 0↑ — bytes/sec: 0 b↓ 0 b↑
Denied — packets/sec: 0↓ 0↑ — bytes/sec: 0 b↓ 0 b↑

inbound-remediation
global
Endpoints 1
Connections/sec 0↓ 0↑
Allowed — packets/sec: 0↓ 0↑ — bytes/sec: 0 b↓ 0 b↑
Denied — packets/sec: 0↓ 0↑ — bytes/sec: 0 b↓ 0 b↑

operations
Implicit Drops — packets/sec: 0↓ 0↑ — bytes/sec: 0 b↓ 0 b↑

inbound-logging
global
Endpoints 1
Connections/sec 3↓ 0↑
Allowed — packets/sec: 10.4↓ 5.2↑ — bytes/sec: 561.5 b↓ 291.1 b↑
Denied — packets/sec: 0↓ 0↑ — bytes/sec: 0 b↓ 0 b↑

outbound-logging
global
Endpoints 30
Connections/sec 0↓ 3↑
Allowed — packets/sec: 5.2↓ 10.4↑ — bytes/sec: 291.1 b↓ 561.5 b↑
Denied — packets/sec: 0↓ 0↑ — bytes/sec: 0 b↓ 0 b↑

cloud-db-sync
global
Endpoints 2
Connections/sec 0↓ 0↑
Allowed — packets/sec: 2.6↓ 2.6↑ — bytes/sec: 141.6 b↓ 141.6 b↑
Denied — packets/sec: 0↓ 0↑ — bytes/sec: 0 b↓ 0 b↑

lb-to-fe
global
Endpoints 1
Connections/sec 0↓ 0↑
Allowed — packets/sec: 0.8↓ 1.6↑ — bytes/sec: 44.8 b↓ 86.4 b↑
Denied — packets/sec: 0↓ 0↑ — bytes/sec: 0 b↓ 0 b↑

fe-from-lb
global
Endpoints 2
Connections/sec 0↓ 0↑

infosec
Implicit Drops — packets/sec: 0↓ 0↑ — bytes/sec: 0 b↓ 0 b↑

eu-noneu-block
global
Endpoints 6
Connections/sec 0↓ 0↑
Allowed — packets/sec: 0↓ 0↑ — bytes/sec: 0 b↓ 0 b↑
Denied — packets/sec: 2.2↓ 9.4↑ — bytes/sec: 132.0 b↓ 563.9 b↑

us-nonus-block
global
Endpoints 6
Connections/sec 0↓ 0↑
Allowed — packets/sec: 0↓ 0↑ — bytes/sec: 0 b↓ 0 b↑
Denied — packets/sec: 2.2↓ 7.8↑ — bytes/sec: 132.0 b↓ 467.9 b↑

infosec-implicit-pass
global
Endpoints 30
Connections/sec 0↓ 0↑
Allowed — packets/sec: 0↓ 0↑ — bytes/sec: 0 b↓ 0 b↑
Denied — packets/sec: 0↓ 0↑ — bytes/sec: 0 b↓ 0 b↑

default
Implicit Drops — packets/sec: 0↓ 0↑ — bytes/sec: 0 b↓ 0 b↑

backend-from-frontend
global
Endpoints 6
Connections/sec 1↓ 0↑
Allowed — packets/sec: 4.8↓ 2.8↑ — bytes/sec: 259.1 b↓ 155.2 b↑
Denied — packets/sec: 0↓ 0↑ — bytes/sec: 0 b↓ 0 b↑

onprem-db-from-onprem-be
global
Endpoints 2
Connections/sec 1↓ 0↑
Allowed — packets/sec: 2.4↓ 1.2↑ — bytes/sec: 129.6 b↓ 67.2 b↑
Denied — packets/sec: 0↓ 0↑ — bytes/sec: 0 b↓ 0 b↑

cloud-db-from-cloud-be
global
Endpoints 2
Connections/sec 0↓ 0↑
Allowed — packets/sec: 1.6↓ 0.8↑ — bytes/sec: 86.4 b↓ 44.8 b↑
Denied — packets/sec: 0↓ 0↑ — bytes/sec: 0 b↓ 0 b↑

eu-cloud-db-from-onprem-vm-be
global
Endpoints 1
Connections/sec 0↓ 0↑
Allowed — packets/sec: 0.8↓ 0.4↑ — bytes/sec: 43.2 b↓ 22.4 b↑
Denied — packets/sec: 0↓ 0↑ — bytes/sec: 0 b↓ 0 b↑

calico-node-alertmanager
calico-monitoring
Endpoints 0
Connections/sec 0↓ 0↑

Figure 3: Tired security policy implementation

Customer success story: Atlassian

Founded in 2002, Atlassian serves more than 125,000 customers around the world. The company delivers a suite of Software-as-a-Service (SaaS) products that enable efficient communication and collaboration across teams, with popular services such as JIRA, Confluence, and BitBucket.

Challenge

Atlassian's traditional approach was not efficient or scalable enough to continue, with deployments taking more than 24 hours and full data centers of unused equipment and large amounts of unused capacity. They chose to move operations to the cloud, seeking to consolidate its compute resources and re-architect its applications for multi-tenancy with Kubernetes. An added layer of complexity is that it hosts arbitrary code execution – a difficult, high-risk security scenario to manage – which is even riskier with multi-tenancy and shared blast radii between customers.

Solution

AWS enabled Atlassian to consolidate its compute resources, and run their workloads on managed clusters, simplifying operations for other teams. They leveraged Tigera Calico to define and enforce policies across their Kubernetes workloads, with security groups on each node to restrict access to sensitive areas of the platform. This helped to reduce the blast radius of their workloads, while decreasing application downtime for their customers.

Results

- Stopped a bitcoin mining cyber-attack in less than 15 minutes
- Improved deployment efficiency, up to 800% with some services
- Security and compliance policy changes can be made in minutes, instead of hours
- Improved IT agility by streamlining the application of network security policies on-premises and on AWS

Next steps

All in all, as Atlassian continues to adopt microservices, they are looking to implement some of the richer functionality of Tigera. Atlassian is working closely with Tigera to extend its functionality to Windows-based workloads, and built out more intelligent traffic management on a per-application basis.

"We can programmatically alter and enlarge or add new IPs, or remove IPs from our Tigera rules. That's a real benefit to using Tigera for us, it plays real nicely with our CI/CD pipeline… we can make changes in minutes"

– Chris Johnston, Kubernetes Platform Senior Team Lead at Atlassian

Resources and getting started

Do you have a question about network security for Kubernetes? Are you interested in setting up a demo, or free trial? We'd love to hear from you!

Contact us

30-day free trial

Tigera Secure Cloud Edition has a 30-day free trial for Kubernetes on AWS, and Amazon EKS.

Sign up

Getting started

Tigera Secure Cloud Edition can be deployed within minutes, via AWS Marketplace.

Get started