From the Node Up:
The Complete Guide
to Kubernetes Security
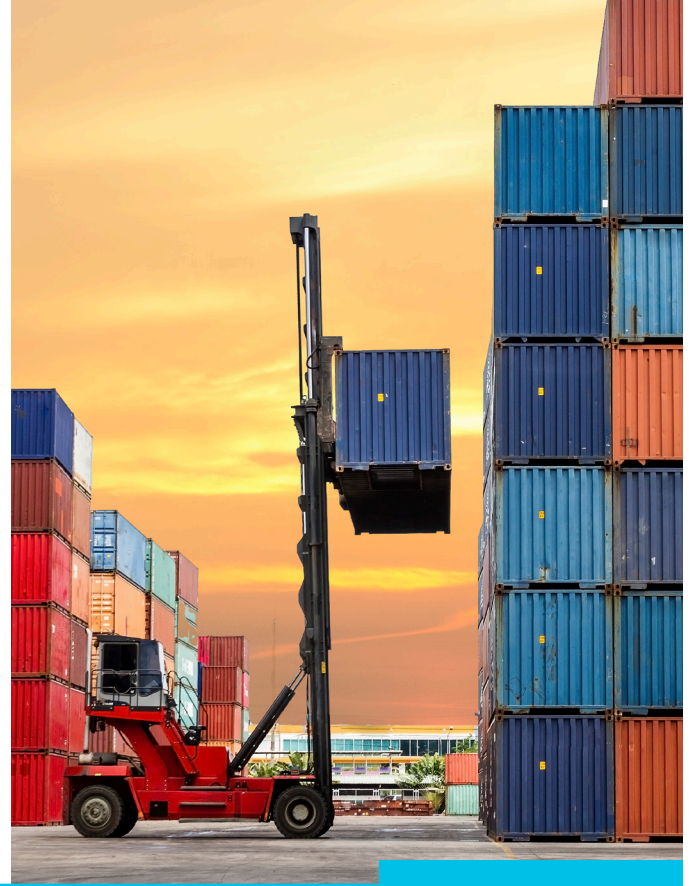
PRISMA® CLOUD | paloalto® NETWORKS

Table of Contents

Introduction

Most discussions of Kubernetes® security focus on how challenging it is to secure a cluster. Kubernetes offers only a handful of native security features, we're told, which makes it exceedingly difficult to secure every layer of a Kubernetes environment.

It's true that Kubernetes provides few built-in security tools and that securing Kubernetes requires addressing multiple types of potential vulnerabilities across multiple layers of infrastructure. That doesn't mean, however, that you need to think of Kubernetes security as being hopelessly challenging.

On the contrary, the fact that Kubernetes is such a sprawling platform with so many integrations presents an opportunity: it makes it easy to build an automated, systematic set of processes that bakes security into the core of the Kubernetes build and deployment process. The result is a tightly integrated security strategy that mitigates threats at all layers and levels of your stack.

This e-book explains how to design a security strategy that reinforces, rather than hinders, the rest of your Kubernetes-based processes. It identifies Kubernetes security challenges from the node up and pinpoints specific solutions for addressing each of them, with a focus on automated, scalable approaches that will keep Kubernetes-based workloads secure, no matter how large your cluster is or which type of infrastructure you use to host it—on-premises, public cloud, or managed service.

Chapter 1: Kubernetes Security Basics

Before delving into specific types of security challenges in Kubernetes or strategies for addressing them, let's briefly walk through the essential high-level considerations about Kubernetes security.

Kubernetes Is a Many-Layered Beast

First and foremost, it's important to understand that Kubernetes is a complex platform that consists of more than half a dozen different components. It has an API server for enabling communications between different parts of a cluster, a scheduler that manages how workloads are distributed, and controllers that manage the state of Kubernetes itself. It also includes an agent that runs on each node, or server, within a cluster; and a key-value store, where cluster configuration data is housed.
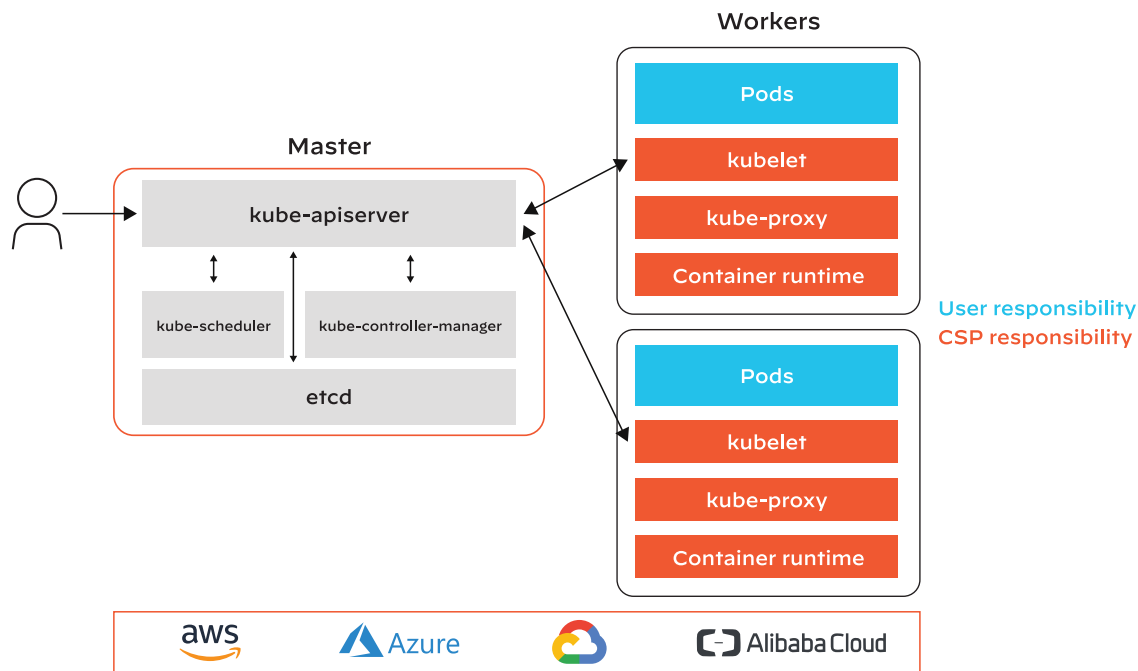

Figure 1: Managed Kubernetes services architecture

Those are only the main components of Kubernetes itself. A functional cluster also relies on many other moving parts, including a container runtime for executing containers, some type of persistent storage solution, a logging tool, operating systems to power each node, and more.

Each of these various pieces of a Kubernetes cluster brings its own set of potential vulnerabilities. Container runtimes may be subject to coding flaws that enable privilege escalation within a container. The Kubernetes API server could be improperly configured, giving attackers the opportunity to access resources that are supposed to be locked down. Vulnerabilities could exist within a containerized application, or within the operating systems running on Kubernetes nodes, that enable privilege escalation attacks or access to sensitive data. These are just a few examples.

In short, securing Kubernetes requires securing a broad set of different components, each subject to its own security needs. There is no single set of tools or processes that can easily secure all aspects of a Kubernetes cluster against all types of vulnerabilities. You need a multi-pronged defense.

Kubernetes Native Security Protections

Complicating Kubernetes security further is the fact that, although Kubernetes provides certain built-in security features, it is hardly capable of securing itself without help from external tools.

Kubernetes allows administrators to define role-based access control (RBAC) policies to help guard against unauthorized access to cluster resources. They can also configure pod security policies and network policies to prevent certain types of abuse on pods and the network that connects them. They can impose resource quotas to mitigate the disruption caused by an attacker who compromises one part of the cluster. With resource quotas in place, the attacker won't be able to execute a denial-of-service attack by depriving the rest of the cluster of sufficient resources to run (assuming, of course, that the breach isn't escalated beyond the part of the cluster where it originates).

Kubernetes allows administrators to define role-based access control (RBAC) policies to help guard against unauthorized access to cluster resources.

Native Kubernetes security features like these plug certain security gaps within a Kubernetes cluster. However, they are of little or no use in addressing many other types of security risks, such as exploits that impact node operating systems or container runtimes.

To build a holistic security strategy for Kubernetes, then, you need to look beyond the handful of built-in Kubernetes security features. The latter can and should be used where appropriate to mitigate security risks, but on their own, they don't even come close to providing all the functionality you need to secure a cluster.

In the following chapters, we'll look at what it takes to build a comprehensive security strategy for Kubernetes that goes above and beyond the platform's limited built-in security functionality.

Chapter 2: Securing Kubernetes Infrastructure

The first overarching layer to secure in a Kubernetes-based environment is the build layer: the set of tools that developers use to build code that will run in a Kubernetes environment.

These tools are not part of Kubernetes itself. However, because a Kubernetes cluster is only as secure as the code that runs on it, taking steps to secure code before it is even deployed to a cluster is a prerequisite for securing all aspects of Kubernetes.

This chapter explains how to secure Kubernetes builds, with a focus on the three main junctions where you integrate security into your automated build pipeline: integrated development environments (IDEs), configuration management, and continuous integration.
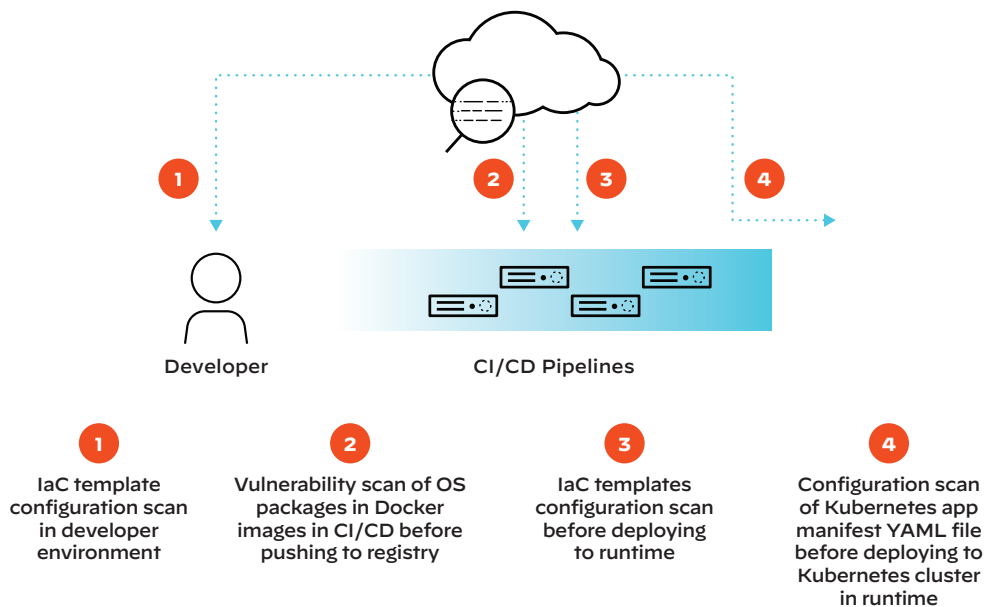
Developer

CI/CD Pipelines

1. IaC template configuration scan in developer environment

2. Vulnerability scan of OS packages in Docker images in CI/CD before pushing to registry

3. IaC templates configuration scan before deploying to runtime

4. Configuration scan of Kubernetes app manifest YAML file before deploying to Kubernetes cluster in runtime

Figure 2: Security in the build pipeline

Integrated Development Environments

IDEs are the tools developers typically use to write source code for applications. As the tool that starts the application deployment pipeline, an IDE is the place where vulnerability scanning should begin. Most IDEs can integrate with a variety of third-party source code vulnerability scanners to detect potential security flaws in application code.
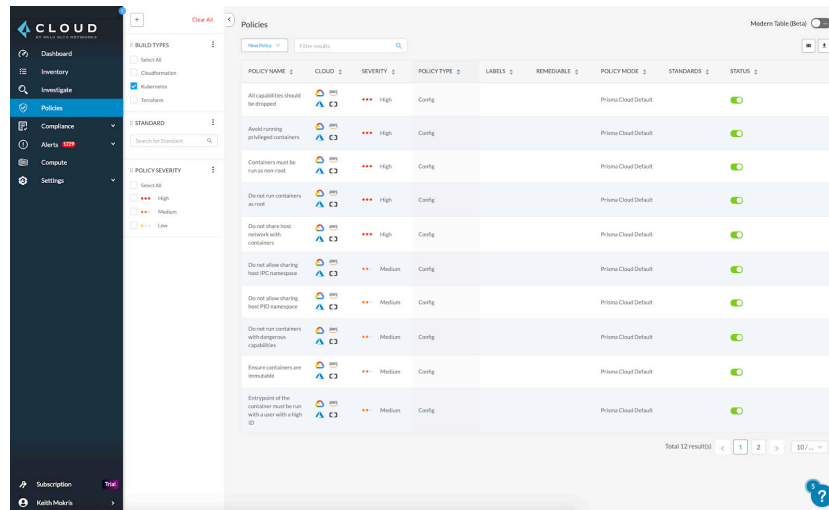
Continuous Integration

Continuous integration (CI) tools host source code and turn it into binaries that can be deployed into Kubernetes. They represent another stage at which code should be scanned for vulnerabilities. Like IDEs, CI servers are compatible with a variety of vulnerability scanners.

Configuration Management

Today, most Kubernetes application build and deployment pipelines rely on automated, policy-based configuration management in the form of infrastructure as code (IaC) and YAML files.

These approaches let Kubernetes administrators write code to define how a cluster (and the infrastructure that hosts it) should be configured, and then apply that code automatically.

In addition to streamlining the process of provisioning a Kubernetes environment, configuration management tools offer an opportunity to scan configuration files for security problems before they are applied. Tools like Prisma® Cloud can do this automatically by comparing your IaC and YAML files to those known to be secure. Prisma Cloud integrates directly with your source code management system, such as GitHub® or GitLab®, making it easy to build a fully automated process for securing Kubernetes configuration files that works with existing build pipelines.

Figure 3: Kubernetes policies in Prisma Cloud

Chapter 3: Securing Container Images to Run on Kubernetes

In most cases, applications are deployed on Kubernetes as container images. (It's possible to manage other types of deployment objects on Kubernetes, including virtual machines, but this is less common.) Container images are checked for vulnerabilities that can exist within the container code itself as well as in any upstream dependencies on which the image is based.

Developer Desktop

There are two ways to go about scanning container images for security issues. The first is to scan individual images manually, using a tool such as twistcli. This is useful if you need to perform a one-off security check of an image.

When container images are built using automated workflows, teams will likely need to integrate vulnerability and compliance scanning.

Continuous Integration

When container images are built using automated workflows on platforms like Jenkins®, CircleCI®, or Azure® DevOps, developers and DevOps teams will likely need to integrate vulnerability and compliance scanning into these workflows. Security platforms like Prisma Cloud can scan these container images to identify issues against frameworks like the Docker CIS Benchmark and enforce standards based on organization or application requirements.

Container Registries

For automated, scalable vetting of container images, however, you should regularly scan all images inside a container registry. Registries are repositories that store container images. By integrating a vulnerability scanner into your registry, you gain full visibility into any threats that may exist within the container images stored in that registry.

One challenge in scanning container registries is that there are a variety of different container registries available. Some Kubernetes distributions, such as Red Hat® OpenShift® and managed Kubernetes services hosted in public clouds, come with their own built-in registries. Others allow administrators to choose from a variety of third-party registries.

This diversity of registry options and configurations makes it important to choose a container image scanning tool that can integrate with any type of registry. Prisma Cloud provides this flexibility, allowing administrators a one-stop image scanning solution no matter how their Kubernetes cluster is configured.

Figure 4: Container image registry scan results in Prisma Cloud

Chapter 4: Kubernetes Runtime Security

Securing an application once it has been deployed into a cluster is the most complex aspect of Kubernetes security. This is because there are so many types of vulnerabilities that can impact an application when it is running, and those vulnerabilities have the potential to be exploited via the application itself as well as through Kubernetes.

There are several steps you can take (beyond securing applications before deployment, using the guidance offered in preceding chapters) to mitigate the security risks of an application after it has been deployed.

Visibility

Above all, it's critical to maintain continuous visibility into your Kubernetes services and resources. Breaches can happen in a variety of ways. The more data you collect about the application environment, the greater your chances of detecting an anomaly that will alert you to a breach.

Security teams may not know where all their clusters are running, and they may therefore lack cohesive information on overall security posture. With Prisma Cloud, security teams gain continuous visibility into cluster locations using API data from the public cloud service providers (CSPs) in addition to configuration and compliance status.

Runtime Protection

Collecting environment data from Kubernetes is relatively straightforward. A key challenge in using that data to monitor for security issues, however, is that Kubernetes clusters tend to change constantly as nodes come offline or shut down, applications are scaled up and down in response to shifts in demand, and so on. As a result, it's impossible to establish a baseline of "normal" activity and measure anomalies against it.

Instead, look for a Kubernetes runtime protection solution like Prisma Cloud, which automatically learns how applications deployed in Kubernetes behave under different conditions. With this insight, users can effectively distinguish normal shifts in application behavior from those that reflect a security problem.

Network Protection

Network-based security threats can impact Kubernetes in two distinct ways: via public-facing networks that connect applications to the internet, and via internal networks that Kubernetes containers use to exchange data within each other.

Detecting signs of malicious activity on both types of networks is therefore crucial for securing Kubernetes network resources. Because network activity, like the rest of a Kubernetes environment, fluctuates constantly—as do the IP addresses of containers—you need a container-aware network scanner that understands the nuances of network traffic in a Kubernetes cluster. You also need a firewall tool that allows you to define rules to protect against network-based threats, and then either alerts you or blocks threats automatically when rules are violated. Prisma Cloud's container-aware network scanning and Layer 4 firewall functionality does all of this.
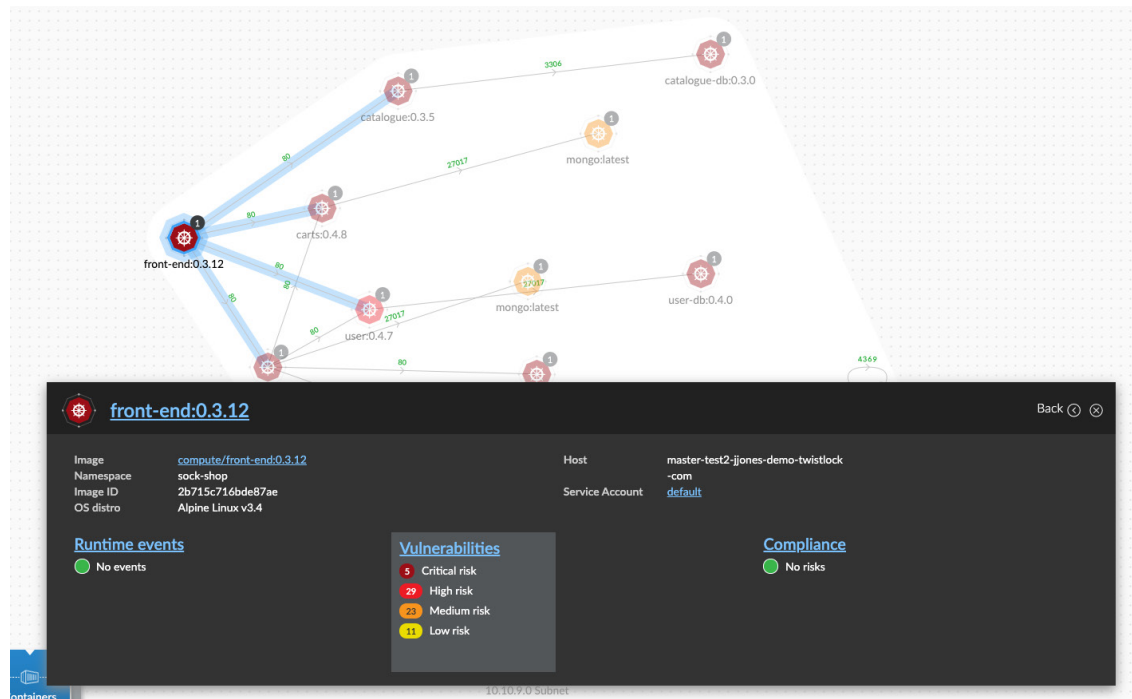

Figure 5: Network topology and container security visualization within Prisma Cloud

Node OS Monitoring

An attacker who takes control of the operating system running on a node within your Kubernetes cluster can wreak havoc of all kinds. Thus, it's critical to monitor not just the internal components of Kubernetes, but also the operating systems that power each of your nodes.

Ideally, you'll be able to do this using the same monitoring solution that tracks your Kubernetes applications so that you won't need to rely on multiple tools and monitor multiple dashboards to detect threats that originate from internal and external sources. Prisma Cloud, which can monitor any type of operating system or cloud infrastructure as well as Kubernetes, offers holistic monitoring functionality.

Kubernetes Security Audits and Compliance

Finally, implement a regular auditing process that scans all layers of your Kubernetes cluster and configurations to ensure they align with industry standards and best practices. Audits won't necessarily detect threats in real time, but they
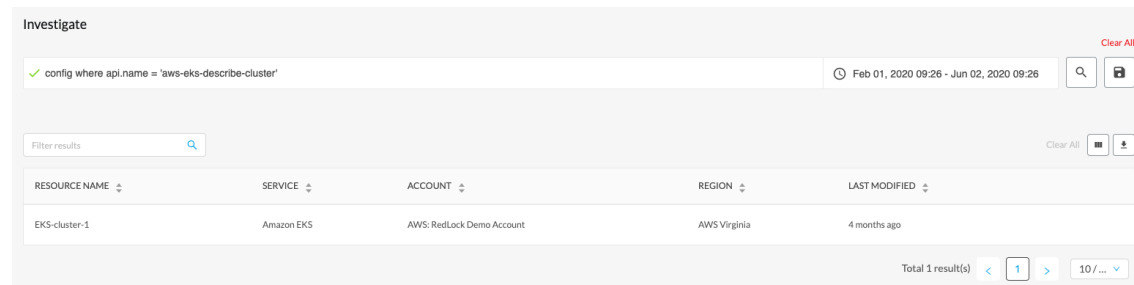

Figure 6: Kubernetes cluster investigation detail in Prisma Cloud

will help you stay ahead of security problems or misconfigurations you may be overlooking that could give attackers an entry point into your cluster or applications.

Prisma Cloud enables top-to-bottom Kubernetes security audits that check all components of the cluster for deviation from established benchmarks and best practices, such as the Kubernetes CIS Benchmark. It includes more than 100 built-in customizable checks for configurations, communications, and more, set for each application or environment.

You can then map those checks to pre-built compliance templates for common frameworks like PCI DSS, HIPAA, GDPR, and NIST SP 800-190. You can also write your own compliance checks, with support for cloud native technologies like Open Policy Agent or Kubernetes AuditSink. This can be useful if you need to meet industry-specific compliance rules or if you are deploying a custom line-of-business application with security needs that generic audit policies do not address.

Conclusion

Kubernetes security can seem daunting, not least because there are so many different moving parts to secure. It's possible, however, to leverage a central tool set to address the various aspects of Kubernetes security. Kubernetes native security features will get you partway there, but it's also essential to leverage external tools that can address vulnerabilities such as malicious code within container images, audit Kubernetes configurations for security risks, and provide ongoing monitoring to detect threats in real time.

Prisma Cloud provides the comprehensive set of security features that teams need to address all facets of Kubernetes security. It also integrates natively with Kubernetes and a variety of associated tools, making it easy to bake security into the Kubernetes build, deployment, and management processes your team already has in place.

To learn more about using Prisma Cloud for Kubernetes security, visit our website.

Prisma Cloud by Palo Alto Networks

Prisma® Cloud is the industry's most comprehensive cloud native security platform (CNSP) with the industry's broadest security and compliance coverage—for applications, data, and the entire cloud native technology stack—throughout the development lifecycle and across hybrid and multi-cloud environments.

The platform offers an integrated approach that enables security operations and DevOps teams to collaborate effectively and accelerate secure cloud native application development.

Prisma Cloud protects and integrates with cloud native architectures and toolkits to ensure complete security coverage while breaking the security operational silos across the entire application lifecycle, allowing DevSecOps adoption and enhanced responsiveness to changing the security needs of cloud native architectures.

For more information, check out paloaltonetworks.com/prisma/cloud.