

CHECKLIST

# Kubernetes Security Checklist



## 5 Steps to Securing Kubernetes

Kubernetes has swiftly emerged as the de facto operating system of the cloud, revolutionizing how developers package applications into portable microservices. Yet, the complexity of operating Kubernetes often leads DevOps teams to postpone crucial security measures until the brink of production deployment. This delay poses significant risks, as Kubernetes demands a novel approach to security — one that traditional tools and processes, with their limited visibility into dynamic container environments, fail to satisfy.

As the cloud landscape increases in complexity, where 70% of containers in 2024 have a lifespan of five minutes or less, the ability to swiftly detect and investigate anomalous behaviors becomes a formidable challenge. Addressing container security risks at the earliest stage is paramount; procrastination not only hampers cloud adoption momentum but also escalates security and compliance vulnerabilities.

The creation, management, and updates of concise checklists for critical cloud applications as they transition to production falls heavily on the DevOps teams. This adds a substantial burden to their existing duties of maintaining cloud infrastructure and application integrity.

In cloud security, time is the most valuable currency. An attack could tarnish reputations in as little as 10 minutes, underlining the urgency of adopting new strategies, technologies, and mindsets to safeguard business innovation at the cloud's pace. It's imperative now, more than ever, to reimagine security that aligns with the speed of the cloud.

With this critical perspective in mind, we have curated a comprehensive checklist to guide your security strategy as you escalate your utilization of containers and Kubernetes. This resource aims to equip you with the insights necessary for a proactive and informed approach to cloud-native security, ensuring your readiness to tackle challenges at cloud speed.

<sup>1</sup> <https://www.zdnet.com/article/technology-containers-short-lifespans-are-getting-even-sh>

STEP

# 01

## Breaking Down Kubernetes Security Risk

The ability to declaratively provision and configure Kubernetes infrastructure settings, as well as application constraints using an “Infrastructure as Code” (IaC) approach, lets organizations set a security baseline across all key aspects of their Kubernetes clusters, regardless of the specific Kubernetes platform and underlying compute infrastructure environment. Declarative configuration also eliminates operator errors that can result in exploitable misconfigurations.

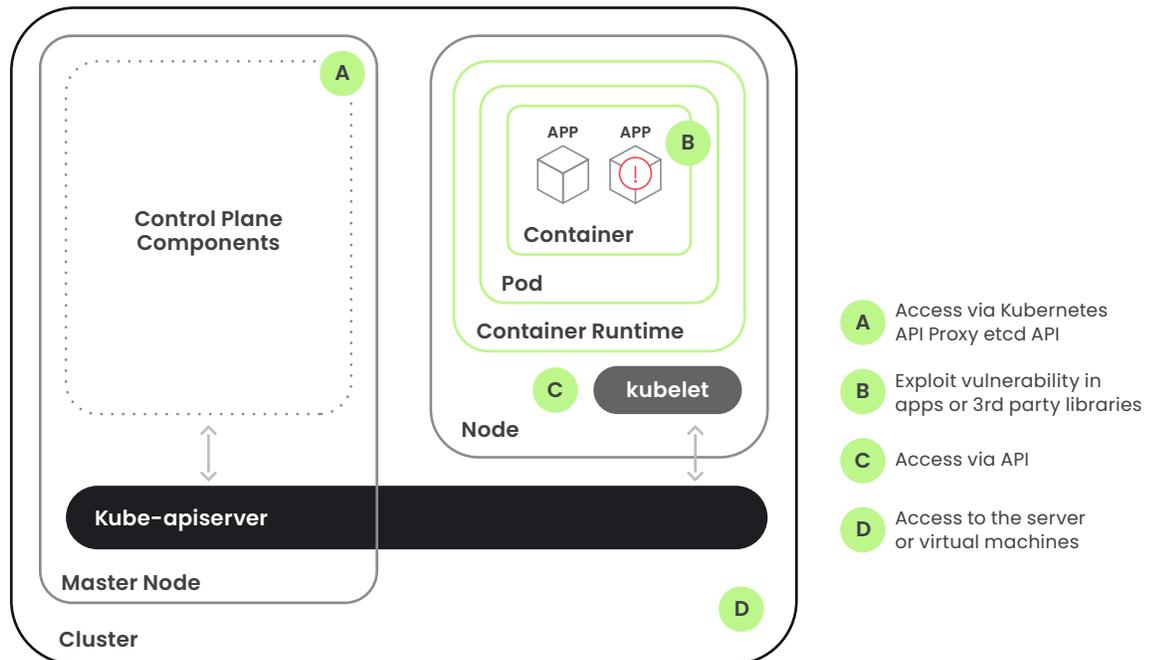
Let’s now take a glance at a Kubernetes cluster to understand which elements you need to protect.

First, you need to protect your applications and libraries. Vulnerabilities in your base OS images for your applications can be exploited to steal data, crash your servers, or scale privileges. Another component you need to secure are third-party libraries.

Often, attackers won’t bother to search for vulnerabilities in your code because it’s easier to use known exploits in your applications libraries.

The next area is the Kubernetes control plane, your cluster brain. Programs like the controller manager, etcd, or kubelet can be accessed via the Kubernetes API. An attacker with access to the API could completely stop your server, deploy malicious containers, or delete your entire cluster. Additionally, your cluster runs on servers, so access to them needs to be protected. Undesired access to these servers, or the virtual machines where the nodes run, will enable an attacker to have access to all of your resources and the ability to create serious security exposures.

Now that we know what to secure, let’s get into the details and review the framework for approaching Kubernetes security.



STEP

## 02

## Securing Infrastructure as Code

### What it is

**Infrastructure as Code (IaC)** is a way to manage both Kubernetes infrastructure and applications within a version control system (e.g., Git). Any changes to infrastructure are achieved through pull requests that change the source files.

Once approved and merged, pull requests will reconfigure and synchronize production infrastructure to match the state defined in the source repository.

This trend is an opportunity for security to shift further left, as part of a secure DevOps workflow to manage risk in Kubernetes.

### Benefits

IaC is rapidly gaining ground because it offers a path towards higher resiliency through better operational control. With IaC security, teams can identify and eliminate configuration risks before infrastructure is deployed in production.

Git commits provide verifiable updates and allow you to also apply auto-remediation using a GitOps workflow. Teams can improve their Kubernetes security posture and close the gap between source and production.

### Approach

DevOps teams can automate compliance and governance using policy as code based on Open Policy Agent (OPA). They can apply policies to scan IaC templates pre-deployment and also detect runtime drift, which can be remediated at the source with a simple pull-request. Fixes can be prioritized based on application context.

### IaC Security and Auto-Remediation

Scan IaC templates

Apply policy as code via OPA

Enforce compliance and governance

Auto-remediate drift

risk-based prioritization

STEP

# 03

## Preventing Threats with Admission Controllers

### What is it

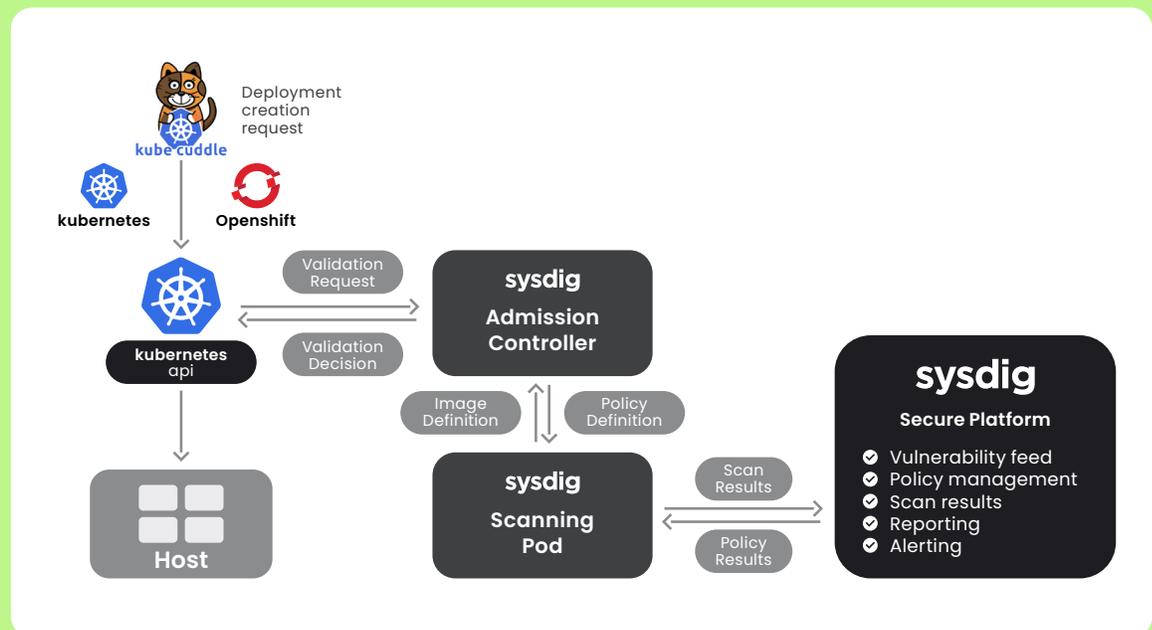
**Kubernetes admission controllers** are pieces of code that intercept Kubernetes API calls before the objects are created. They can be seen as a gatekeeper that intercepts API requests and enforces what can run on the cluster.

CI/CD image scanning is a critical requirement when implementing container and Kubernetes security. But developers might sometimes bypass the CI/CD pipeline and deploy an image directly to the cluster. You can integrate an **admission controller** with a scanning engine to prevent risky images from being deployed if they don't meet your security and compliance requirements.

### Benefits

When **image scanning** is used with the **admission controller**, you can block threats before they reach production. Immediately trigger a scan for every image that is trying to be deployed in the cluster. You can also use additional environment context when defining admission criteria, such as namespace, pod metadata, etc. By triggering **image scanning** via the admission controller, you can:

- ✓ Check your application, its libraries, and other files for well-known vulnerabilities.
- ✓ Analyze the metadata to detect misconfigurations like exposed insecure ports, running as privileged (root) users, or exposed credentials.
- ✓ Define custom checks, like package blacklisting or detecting wrong file permissions.



If these security policies are not met, you can block the image from reaching production and notify your developers to fix the issues.

### Approach

Enable the **Kubernetes admission controller** and integrate with a scanning engine to prevent risky or unscanned images from being deployed.

### What it is

PodSecurityPolicy (PSP) is a cluster-level resource that controls the actions a pod can do or what resources it can access, and can be used to implement least privilege access for pods.

### Benefits

PSP can prevent threats without impacting performance at runtime by enforcing least privilege access for pods in your clusters. You can enforce preventative controls such as disallowing running privileged containers, restricting resources, or limiting access to volumes at this level.

### Approach

PodSecurityPolicy is implemented as an optional (but recommended) admission controller.

STEP 04

# Securing the Kubernetes Control Plane

### What it is

The Kubernetes control plane is the brain of your Kubernetes cluster. It manages all of your cluster resources, can schedule new pods, and can read all of the secrets stored in the cluster.

### Benefits

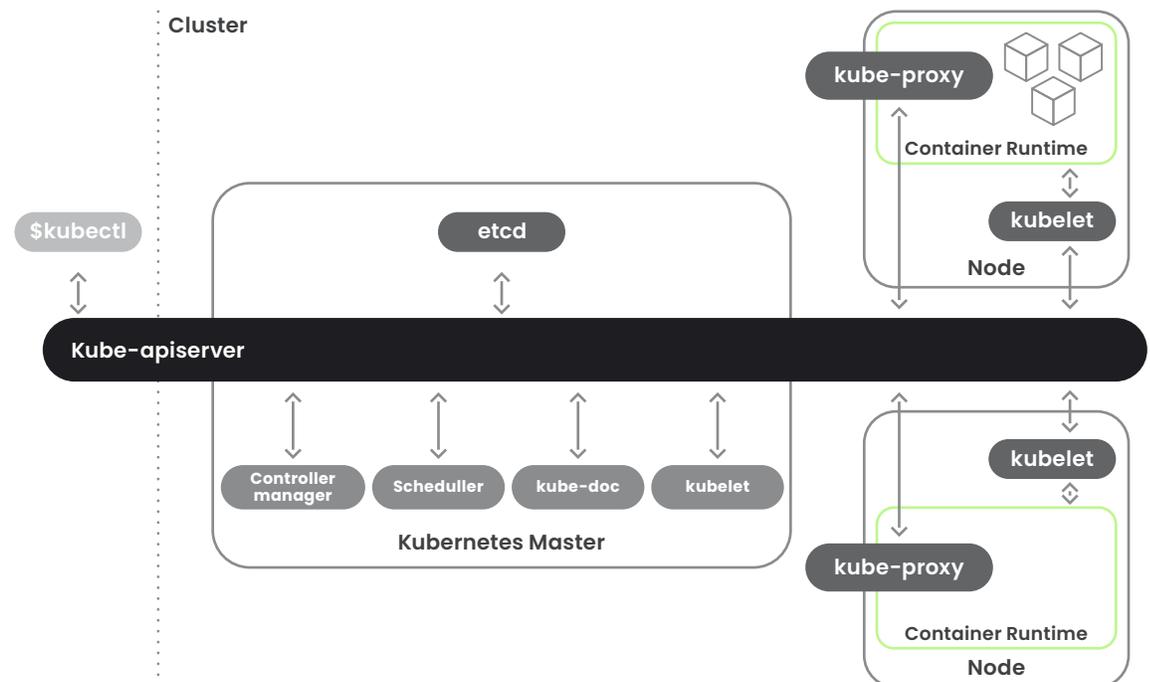
The control plane controls your cluster; securing it will prevent a malicious user from extracting information, crashing your infrastructure or scheduling pods with access to the parent node.

### Approach

Isolate the cluster network, secure the API, and audit kubectl commands.

Control plane components communicate via the Kubernetes API, and kubectl instructions also translate into API calls. To secure it:

- ✓ Check the **kubelet** config: Disable `anonymous-auth`, set a `client-ca-file`, ensure `authorization-mode` delegates to the API server, and disable the `read-only-port`.
- ✓ Enable `NodeRestriction` in your API so kubelets are only allowed to perform modifications in their own node.
- ✓ Enable authorization via RBAC.



## STEP

## 05

## Securing Workloads at Runtime

### What it is

Managing security risk at runtime in containers and Kubernetes environments. **Runtime security** detects abnormal behavior that could indicate a container has been compromised.

### Benefits

Flag owners and respond quickly to newly discovered **vulnerabilities** before they are exploited. Detect and remediate **attacks** when they happen, before they cause major damage. Protect from software **bugs** or **misconfigurations** that cause erratic behavior and resource leaking.

### Approach

Scan continuously so you can detect issues as soon as possible. Also, place automatic incident responses so action can occur right away. Finally, capture forensic data when an incident happens so you can investigate the root cause and prevent it from happening again. Let's expand a bit on each of those strategies.

#### > Runtime vulnerability reporting

After an image is initially scanned, new vulnerabilities may be found or your policies may change. You need to keep scanning your images to ensure that they're secure over time. Some image scanners would require you to do a full re-scan each time, while others will save the metadata and will be able to warn you of new issues without a new scan. You need to be able to map critical vulnerabilities (e.g., CVEs with a fix available in images that are running longer than 30 days) to specific applications and identify teams responsible to fix them. This requires mapping CVEs back to the Kubernetes asset landscape (specific namespaces, deployments, clusters, pods, etc.).

#### > Abnormal behavior detection

Is your container doing what it's supposed to do? Is it accessing files it shouldn't? Does it have strange network connections? Did anyone spawn a terminal shell? By monitoring your container's activity, you can detect abnormal behavior.

You'll need **instrumentation** to detect these issues. Does your instrumentation cover just your apps, or also the system calls? The more data you have, the more behaviors you'll be able to detect. How many **resources** does your instrumentation need? Some solutions will need a lot of memory, while others will tax your CPU.

**Falco** is the de facto Kubernetes threat detection engine; it detects unexpected application behavior and alerts on threats at runtime. Falco captures system calls using eBPF\* (among other sources), which provides visibility into runtime system activity with Kubernetes application context, and also makes it ready for high performance production environments.

Creating rules for all of your pods can be a time consuming task. Having a wide library of **out-of-the-box rules** available can make a difference here. With so many images it's easy to miss something, so being able to use **machine learning** to profile expected behaviors is a nice safety net.

#### > Threat detection via operational security

Log services like AWS CloudTrail can enable governance, compliance, operational auditing, and risk auditing for your cloud account. With it, you can log, monitor, and retain account activity related to actions (configuration changes, events created/deleted/modified) across your entire cloud infrastructure. The out-of-the-box set of Falco rules for CloudTrail, a source of truth for operational audit, can minimize the setup effort, response time, and the resources needed. For runtime visibility you need to implement AWS audit log threat detection in order to properly investigate security events.

STEP

# 06

## Implementing Kubernetes Native Network Segmentation

### What it is

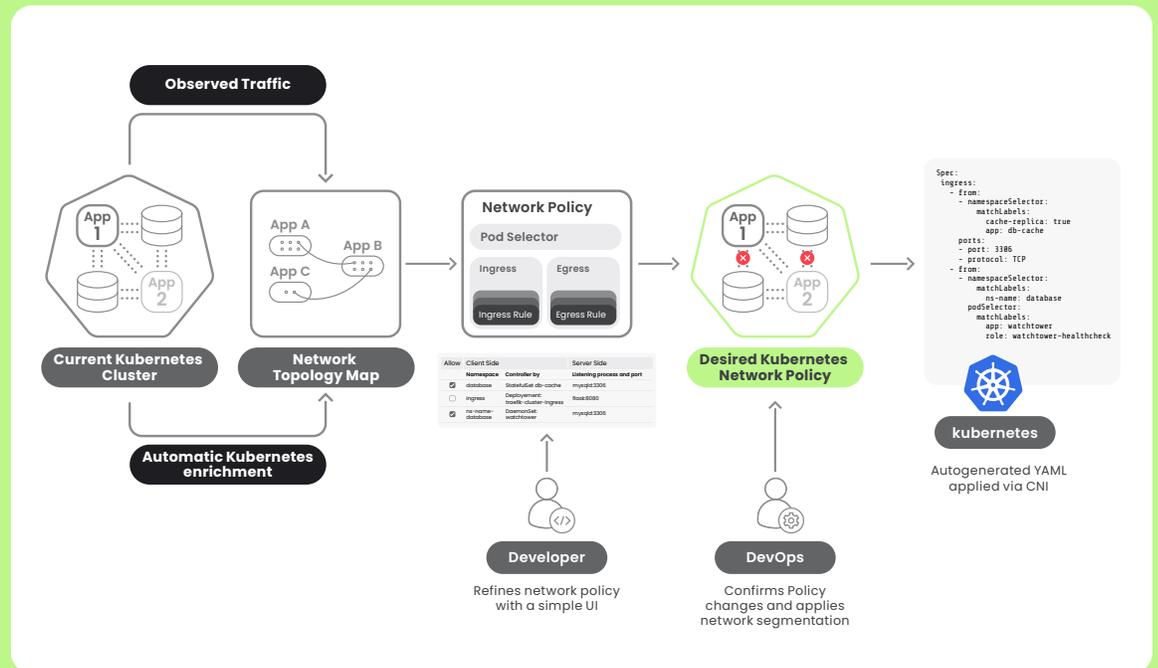
By default, Kubernetes pods are non-isolated, meaning they accept traffic from any source. Kubernetes network policies are a native security control of the platform that can be used to define how apps and services explicitly communicate with one another.

### Benefits

Protect against threats like lateral movement across containers, privilege escalation, and data exfiltration. Your teams can also meet compliance requirements (NIST, PCI, etc.) that require network segmentation.

### Approach

- ✓ Monitor all network connections made between apps and services running in Kubernetes.
- ✓ Use application context and Kubernetes metadata. Use native controls of Kubernetes network policies to implement least privilege policies.
- ✓ Configure Kubernetes native Network Policies to segment and restrict traffic between, to, and from pods.
- ✓ Simplify network policy management and integrate in your policy as code framework.



## STEP

## 07

## Automating Incident Response and Capturing Forensics

**What it is**

When an unauthorized event occurs, you'll need a full audit trail that describes what and when it changed, and by who. You will also need to know how sensitive files were modified and who made those changes.

**Benefits**

You can become operationally efficient and audit-ready with your container environment. It will also help you resolve issues quickly and validate compliance requirements for PCI, NIST, SOC2, etc.

**Automatic incident response**

React to incidents right away before they become a bigger issue. Nothing is faster than an automatic response. Critical incidents will require you to stop the affected pods, but for other incidents, a notification is enough. Being able to notify the relevant people for further investigation through the appropriate channels is crucial.

**Auditing and forensic tools**

You need to capture all of the information you possibly can around an incident since, by the time you're going to investigate it, the containers may already be gone. Besides the captures, you'll need a way to browse the data so you can correlate events and find the source of the issue faster. For example, you should be able to identify unusual network activity, correlate it to shell commands executed around that time, and see what files changed.

**Audit Tap**

Track every network connection to or from a specific process, even if the connection is not successful. Capture a record of all accepted / failed net connections to identify suspicious or unusual processes.

**File integrity monitoring (FIM)**

Gives you visibility into all of your sensitive file related activity. It's used to detect tampering of critical system files, directories, and unauthorized changes, regardless of whether the activity is a malicious attack or an unplanned operational activity.

- ✓ Bake FIM checks into your image scanning policy.
- ✓ Create Runtime Policies to monitor for Filesystem Changes.
- ✓ Implement an automated response mechanism.
- ✓ Ensure you have comprehensive forensics data.

**Automated response**

- Notify incidents
- Pause / Kill containers

**Forensics data**

- What changed
- When?
- By **Who**?

STEP

## 08

## Comparing Kubernetes Security Options: DIY or Turnkey

## Steps to Securing Kubernetes

## Open source (DIY)

sysdig  
(Turnkey)

## Threat prevention with admission controllers

Kubernetes admission controller can integrate with a scanning engine to validate if images are vulnerability free.

kube-psp-advisor is a tool that makes it easier to create K8s Pod Security Policies (PSPs) from either a live K8s environment or from a single .yaml file containing a pod specification.

Sysdig Secure embeds scanning into the CI/CD pipeline. It provides out-of-the-box policies covering best security practices and compliance standards.

Prevents risky images from ever being deployed (via Kubernetes admission control).

You can scan directly in the pipeline and prevent risky images from going into the registry.

You get out-of-the-box integrations and alerts with tools like Slack, SNS, PagerDuty, etc.

## Securing Kubernetes control plane

Validate cluster configuration is compliant based on CIS Benchmarks for Kubernetes (kube-bench).

K8-security-configwatch can review the changes in your Kubernetes config files, and highlight those that can affect the security of the cluster.

Use Falco to detect unexpected Kubernetes control plane activity.

Gain deep visibility across hundreds of thousands of nodes with out-of-the-box dashboards to monitor Kubernetes control plane activity.

Detect anomalous activity faster with curated Falco rules based on Kubernetes audit logs, with automatic remediation, alerting, and notification integrations.

Schedule continuous compliance assessments and generate reports based on CIS benchmarks for Kubernetes.

## Steps to Securing Kubernetes

## Open source (DIY)

## sysdig (Turnkey)

### Securing workloads at runtime

Falco, the open source cloud-native runtime security project, is the de facto Kubernetes threat detection engine.

Falco detects unexpected application behavior and alerts on threats at runtime.

Detects new vulnerabilities at runtime and ties the risky image to a specific namespace, cluster, deployment, pod, etc.

Save time detecting anomalous activity by extending Falco with curated out-of-the-box rules.

Improve DevOps productivity by using ML-based image profiling.

Gain deeper visibility into all network traffic across containers running on hybrid/multi-cloud environments.

### Kubernetes native network segmentation

Kubernetes Network Policies are a native resource that allow you to specify how a pod is allowed to communicate with various “entities” over the network.

Network policies are implemented by a network plugin like Calico.

Reduce risk with network visibility that enables microsegmentation in minutes.

Deep visibility provides guardrails for teams without Kubernetes security expertise.

Implement the right policies with a unified view and shared context across teams.

Simplify network policy management by automating K8s network policies.

### Incident response and forensics

Sysdig is an open source Linux system exploration and troubleshooting tool for containers.

Speed up incident response with comprehensive audit trails and deep forensics data.

Respond faster via auto-remediation and alerting.

Validate runtime compliance with policies mapped to various compliance standards (NIST, PCI, SOC2).

# Secure Every Second.

Dig deeper into how Sysdig provides  
Kubernetes Security.

GET PERSONALIZED DEMO →

---

**sysdig**

---

KUBERNETES SECURITY  
CHECKLIST

---

COPYRIGHT © 2021-2024  
SYSDIG, INC.  
ALL RIGHTS RESERVED.  
CL-004 REV. D 3/24

---

