Red Hat

Whitepaper

A guide to achieving DevSecOps in Kubernetes environments

Introduction

DevOps-driven adoption of new technologies and processes may mean security is an afterthought and can expose new gaps in security coverage and risk management.

Over the past decade, organizations have increasingly embraced modern software development practices, public cloud infrastructure, and cloud-native software such as Kubernetes and containers to fuel their digital transformation and innovation. At the center of these changes is DevOps, a set of practices and tools designed to enable teams to deliver software applications and manage infrastructure environments at high velocity.

DevOps emphasizes principles such as increased collaboration, shared responsibility for development and operations, removing barriers between operational teams, and autonomous decision-making, all in the spirit of achieving greater speed and consistency. DevOps relies on methodologies that use automation and continuous integration and delivery (CI/CD), and treats infrastructure and application components as immutable.

These changes can strain existing security programs. DevOps-driven adoption of new technologies and processes may mean security is an afterthought and can expose new gaps in security coverage and risk management. Security teams must therefore work toward a familiar set of goals for modern computing environments—avoiding security incidents, breaches, and exposures; establishing security best practices and policies to be implemented on an organization-wide basis; and managing resources to minimize operational overhead, alert fatigue, tool sprawl, and manual investigative workflows—in ways that align with the approaches that engineering teams favor.

This has given rise to the concept of DevSecOps: the combination of DevOps practices and security strategies as a means for every organization to increase protection and reduce risk to their modern software environments.

This whitepaper provides an overview of what DevSecOps is and how organizations can adopt its practices in conjunction with technologies such as Kubernetes and containers to achieve strong, scalable security for their cloud-native environments.

Kubernetes security trends and challenges

Organizations continue to rapidly shift their software development efforts to hybrid cloud environments focused on Kubernetes, containers, microservices, and service meshes. These modernization efforts can substantially impact organizations by requiring new personnel skills, tooling, processes, and culture.

The introduction of new technical architectures and operational patterns associated with these efforts also results in security challenges that stem from greater complexity—for example, the number of cloud-native technologies that a single organization uses can easily climb into the dozens.

The practices that DevOps teams use to achieve scale and speed make it harder to coordinate security controls implementation. Automation, APIs, and fast iteration in software releases often means security lags behind.

The sprawl of smaller-sized application components running in containers, microservices, and serverless functions makes it harder to ensure that scalable, strong security controls are applied uniformly across infrastructure and applications.

These challenges are often amplified by existing issues such as misalignment of priorities between engineering and security teams; lack of coordination between development, operations, and security; and a collective knowledge gap in how to effectively prevent security incidents in an emerging landscape of cloud-native threats.

DevOps teams are focused on building and shipping software while security is focused on avoiding incidents and exposures. These different priorities introduce tradeoffs that may result in inadequate attention being given to shared security outcomes. DevOps and security teams must work to ensure that the organization faces an acceptable level of risk while minimizing additional overhead and complexity in software development and operations.

The practices that DevOps teams use to achieve scale and speed make it harder to coordinate security controls implementation. Automation, APIs, and fast iteration in software releases often means security lags behind. Finally, new technologies such as Kubernetes introduce new threats that are not fully clear, meaning the "who, how, and what" teams must work together to protect against frequently remains unclear or is not widely understood by key stakeholders.

What DevSecOps is and why it matters

DevSecOps can enable an organization to secure their software environments with greater speed, at larger scale, and more comprehensively when compared to traditional security strategies.

DevSecOps promises to better align engineering and security teams by addressing the challenges outlined above. At the same time, it is not a panacea. Rather, organizations should focus on integrating DevSecOps principles into the tools and processes they utilize to build, ship, and secure software, to the extent that this approach serves their needs.

DevSecOps is based on the idea that security is everyone's responsibility and that collective attention on security across engineering and security teams can lower risk for the entire organization. To successfully achieve DevSecOps requires an organization to realize it impacts security in three important ways.

First, security controls must be integrated continuously across the software life cycle from the time application components are built to when they are running. Second, security that is implemented earlier in the life cycle following a "shift left" approach can have an outsized impact on improving security and minimizing the overall operational overhead required. Third, development engineers and DevOps end users must be recognized as security users: they must be empowered to implement and make independent decisions regarding security controls.

When adopted effectively, DevSecOps can enable an organization to secure their software environments with greater speed, at larger scale, and more comprehensively when compared to traditional security strategies that were not designed to safeguard modern infrastructure and applications.

Cloud-native technologies require a DevSecOps approach

Cloud-native technologies such as Kubernetes, containers, microservices, and service meshes have become tremendously popular because they provide the building blocks, or "primitives," necessary for organizations to build, deploy, and run applications more dynamically, reliably, and at greater scale than was previously possible. In particular, Kubernetes, the standard in container orchestration,

has been widely adopted with 78% of the Cloud Native Computing Foundation (CNCF) community running it in production today. It allows businesses to benefit from the full potential of cloud environments with faster time to market, cost savings, and greater operational flexibility.

However, those benefits also come with associated security demands. Kubernetes exposes organizations to additional risks due to the complexity involved in operating the system itself, which may result in administrator errors; new infrastructure components that are not secured by default; and the need to apply new security controls that were never previously required. Security and DevOps teams must consider it their responsibility to jointly address these new challenges.

Security teams need to understand Kubernetes and cloud-native technologies sufficiently to establish relevant preventative measures and controls. DevOps teams have to incorporate strong security protections in the workflows and toolchains they use to provision infrastructure and build software applications in Kubernetes environments.

Cloud-native technologies generally share several attributes that are key to fulfilling these aims and therefore present a strategic opportunity to accomplish security in new and better ways. These technologies are based on configuring environments and their resources declaratively, they emphasize treating infrastructure and application components as immutable, they expose user-friendly abstractions, and they are extensible and flexible in how they can be used.

These attributes can be used as part of a DevSecOps approach and make it easier to integrate security earlier and throughout the entire application life cycle. Organizations can also take advantage of orchestration, automation, and declarative configuration to achieve highly scalable security and ultimately spend less time remediating security issues by putting stronger controls in place and identifying security risks earlier, before incidents arise.

The following sections cover approaches that organizations can take to apply DevSecOps practices to securing their cloud-native applications running on Kubernetes.

Applying DevSecOps practices to Kubernetes

DevOps users: The new security users

Kubernetes, in particular, creates the potential for DevOps end users to impact security and help organizations shift toward DevSecOps rapidly

As part of the shift to a DevSecOps approach, teams must delineate roles and responsibilities for achieving organization-wide security goals. Security teams must account for DevOps-driven initiatives as part of their overall strategies and can no longer unilaterally block them.

Because DevOps teams will likely take on significant responsibility for operationalizing security tools and practices, they should also be recognized as a new class of security users that is critical to ensuring security is delivered at the speed and scale needed to operate. In turn, security teams should turn their attention to setting holistic security policies and defining best practices that can be easily and repeatedly adhered to by DevOps users.

Kubernetes, in particular, creates the potential for DevOps end users to impact security and help organizations shift toward DevSecOps rapidly. This is because a number of native Kubernetes security controls, such as network policies (used to apply ingress and egress restrictions on pod-to-pod traffic), role-based access control (used to define user and service account roles and privileges), and pod security policies and security contexts (used to set access and permission constraints on pods within Kubernetes clusters), can be configured and operated by these users rather than relying on

separate security tools that are managed by security operators. This presents a significant opportunity for security teams to help these end users more efficiently improve the organization's security posture for their Kubernetes infrastructure and applications.

Kubernetes security is only as good as where you start: The software supply chain

Because the software supply chain makes up the primary area where DevOps methodologies are typically applied (e.g., using CI/CD pipelines), it is also arguably the easiest area to start extending DevOps practices to DevSecOps practices.

Achieving DevSecOps in Kubernetes environments starts with focusing on how applications are built. Container images are the standard delivery format for cloud-native and Kubernetes applications and so the initial area for getting DevOps teams involved in security should be the software supply chain because cloud-native technologies redefine its importance.

Traditionally, applications would be subject to updates, patches, and manual changes while running in production; instead in cloud-native environments, developers and DevOps teams rebuild and redeploy containers while treating running containers as immutable. This means that the software supply chain acts as a centralized place for an organization to apply all production changes, whether it means building applications from trusted sources including base operating system images or unit testing or continuous integration. By securing the software supply chain, containerized applications can be better protected before those containers are ever running.

Because the software supply chain makes up the primary area where DevOps methodologies are typically applied (e.g., using CI/CD pipelines), it is also arguably the easiest area to start extending DevOps practices to DevSecOps practices. Security teams can use this to their advantage when protecting cloud-native applications. For example, one of the most valuable steps organizations can take to improve their security hygiene for Kubernetes environments is to avoid introducing vulnerabilities in their applications into production environments. DevOps teams can accomplish this by incorporating container image scanning into CI/CD pipelines to identify operating system and language-specific vulnerabilities in their application images and set criteria to be used to appropriately fail builds or restrict deployment of pods into production clusters.

Other security functions that a DevSecOps approach can improve across the software supply chain include removing unnecessary tools and components in images to ensure they are only as minimal as needed in order for applications to run, ensuring secrets are not embedded in images, and using immutable tags (sometimes supported by the image registry) to track specific versions of images that are used to deploy containerized applications.

Declarative configuration: Use the power of DevOps for Kubernetes security advantage

Organizations looking to adopt DevSecOps can also extend existing DevOps-centric workflows and concepts to security as well. The majority of DevOps teams use an "Infrastructure as Code" (IaC) model that is based on interfacing with declarative APIs to ensure configurations are specified once upfront and implemented consistently across environments, a so-called "configure once, run everywhere" model. This is typically used for provisioning cloud infrastructure, configuring cloud services, setting up recurring API interactions or jobs, and a number of other workflows.

Extending IaC to security functions as part of a DevSecOps initiative can enable a "Security as Code" approach that allows for proactive, automated, and repeatable configuration of security controls across infrastructure and applications.

For example, most production Kubernetes environments are made up of multiple clusters that may or may not be provisioned across multicloud or hybrid cloud environments. Organizations must figure out how to configure these clusters more securely and consistently both at the time their infrastructure is provisioned and when applications are deployed.

Significant differences exist between Kubernetes platforms, especially when considering their default security settings (or the lack thereof)—some expose weaknesses by including and deploying the Kubernetes dashboard, some only support older versions of Kubernetes that contain known vulnerabilities, and others have weak access control (authentication and authorization) and network traffic restrictions. A DevSecOps approach can bolster consistent security to compensate for these differences.

The ability to declaratively configure Kubernetes infrastructure settings as well as application constraints (with a pod specification or deployment manifest) using a DevSecOps model 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. For example, users can specify a single configuration, such as a Kubernetes network policy, to be propagated across all pods in a deployment rather than having to configure system-level controls on every host in a cluster. This baseline then enables security operators to more easily identify unexpected, anomalous, and malicious activity that warrants further investigation and analysis, streamlining the entire security life cycle.

Declarative configuration also eliminates operator errors that can result in misconfigurations that can be exploited. It can also reduce overall complexity associated with ongoing configuration management.

Standardization: Align teams with a single, familiar operational framework

DevOps practices often promote standardization as a way of accelerating collaboration and interaction between various stakeholders. DevSecOps extends this by looking for opportunities to streamline workflows and consolidate tooling for security functions. This can be achieved by minimizing the number of new interfaces, configurations, and resource models that users have to learn to implement preventative security measures. Instead, aligning security with open, standardized abstractions and tooling already used by DevOps teams reduces complexity and mitigates potential security issues from arising.

Applied to Kubernetes, a DevSecOps approach has a rich set of useful standardized abstractions. To build and ship containerized applications, developers and operators already utilize standards such as image formats and YAML manifests. Additionally, Kubernetes has a robust object and resource model that includes concepts such as deployments, ReplicaSets, DaemonSets, and others. Many of these are the primary way that pods, the smallest provisionable unit of computing in Kubernetes, are deployed and scaled.

Because developers and operations engineers already utilize these Kubernetes abstractions, security should align with these as well rather than implementing a custom framework of its own. This means basing security functions such as visibility, configuration management, and runtime detection on existing Kubernetes objects and resources to give all stakeholders a single, consistent understanding of security issues. This helps organizations realize greater cross-team alignment and simplify the Kubernetes learning curve for teams, thereby saving time and costs when executing cloud-native security strategies.

By taking a DevSecOps approach that incorporates and analyzes application context introduced in the software supply chain alongside security and policy violations at runtime, security and engineering teams can more quickly determine their relevant risk levels and prioritization concerning how their production environments are impacted.

DevOps context drives better, faster Kubernetes security analysis and decision-making

Most organizations believe they have to do too much work to resolve security incidents. This only gets harder when containerized applications are dynamically orchestrated, scale to large numbers, and are ephemeral. As previously discussed, the software supply chain acts as a centralized place to make any software changes that will propagate through the rest of the application life cycle and into production environments.

The software supply chain also serves as a problem area where users can incorporate additional application context that can be highly valuable when security issues end up arising later in the application life cycle, at runtime. A DevSecOps approach to Kubernetes security seeks to use this context to speed investigations of security incidents and subsequent issue remediation.

Application context in Kubernetes can take many forms including metadata and attributes that are added to Kubernetes artifacts such as manifests and images (for example, immutable tags or cryptographic signatures). This context from Kubernetes may include information across a deployment about what processes will execute within containers, whether any resource limits exist, the system-level privileges and capabilities granted to individual containers, whether the container's root filesystem is read-only or not, and what block devices and secrets are present. It also may include valuable metadata such as labels and annotations. Labels help establish what an application is, while annotations add descriptions about the application.

Evaluating this context across pod replicas can produce a quicker, clearer understanding of application baselines as they apply to expected behavior of running containers—for example, network traffic patterns or execution of specific container processes—to more easily identify anomalies and threats with greater accuracy, reduce alert fatigue, and eliminate manual workflows. By taking a DevSecOps approach that incorporates and analyzes application context introduced in the software supply chain alongside security and policy violations at runtime, security and engineering teams can more quickly determine their relevant risk levels and prioritization concerning how their production environments are impacted.

Bringing the best of DevOps and security together with full life-cycle policies

DevSecOps is meant to enable practices that integrate security end-to-end from the time applications are built through the time they are running in production. This requires ensuring the policy frameworks that are adopted are able to incorporate criteria across the entire application life cycle.

Teams need to consider the feasibility, ease of operation, and overall impact on DevOps processes and workflows as part of complying with and enforcing these policies on an ongoing basis. Ideally, a DevSecOps model can empower individuals to operate autonomously in applying these policies across an organization's environment that has many applications deployed and running in multiple production clusters.

For containerized applications, policies should be enforced on images to fail builds based on parameters such as vulnerabilities or whether unnecessary tooling and packages are embedded within the images. Policies also need to be enforced to restrict the deployment of pods if their levels of access or permissions exceed what is necessary, for example in pods that run privileged containers or mount container filesystems that have both read and write access.

In Kubernetes environments, a DevSecOps approach to policy enforcement is best realized using the orchestration system, Kubernetes itself, to carry out enforcement actions such as killing pods, preventing containers from being launched, or restricting system-level activities that an application is allowed to perform.

Finally, running applications should be subject to policies if runtime activity deviates from what is expected, if a known malicious process is executed or an unusual network communication is attempted outside of typical patterns.

As application owners, DevOps teams should set policies that incorporate security criteria across different phases of the application life cycle (build, deploy, run) that match how these applications are intended to operate. This approach allows important aspects of security to be encapsulated within end-to-end policies that all stakeholders can monitor rather than working with policies that are implemented in isolation by engineering and security teams.

How immutability supports policy enforcement when things go wrong

A core tenet of cloud-native software is that infrastructure and applications should be considered immutable—once running, they are not updated or patched. Instead, any changes to images, configuration files, or anything else are made at their source and these components should be torn down and subsequently re-deployed. Similarly, DevOps principles emphasize fast iteration, frequent updates, and high release velocity. Together, these can support a DevSecOps operational model for scalable, orchestrated security enforcement.

In Kubernetes environments, a DevSecOps approach to policy enforcement is best realized using the orchestration system, Kubernetes itself, to carry out enforcement actions such as killing pods, preventing containers from being launched, or restricting system-level activities that an application is allowed to perform. This approach minimizes operational risk to running applications, ensures greater scalability, and eliminates the need for DevOps teams to run and maintain additional tooling.

Enforcement should not be viewed as applicable to runtime only. In Kubernetes environments, organizations can implement multiple points of enforcement across the application life cycle: throughout CI/CD pipelines, at deployment time using Kubernetes admission controllers such as pod security policies, and at runtime. This allows users to "gate" certain activities based on potential security issues and make necessary changes as early as possible.

This DevSecOps-friendly approach to policy enforcement was previously not possible when infrastructure platforms lacked the security controls that exist natively in Kubernetes—controls such as network policies for network segmentation; admission controllers for intercepting and possibly rejecting requests to the Kubernetes API server; secrets for storing sensitive credentials; role-based access control for granting authorization to users and services accounts; and security contexts, pod security policies, and support for Linux® security modules such as seccomp for setting system-level constraints at the granularity of individual containers.

By utilizing this rich set of controls, security and engineering teams can apply DevSecOps practices to achieve a faster, more iterative, fine-grained framework for enforcing security policies in Kubernetes environments.

Closing the loop: Streamline remediation across the full life cycle

When engineering and security teams adopt a mindset that infrastructure and applications are immutable, traditional approaches to incident response and remediation are rendered obsolete. Security operators may no longer be the primary personnel responsible for remediation. Instead, because changes to address root causes must be made upstream in image builds, DevOps teams must increasingly focus on remediating vulnerabilities, misconfigurations, and other sources of security incidents. Therefore, applying DevSecOps to cloud-native environments requires that DevOps users tackle significant aspects of response and remediation.

However, this DevSecOps-driven approach to remediation requires establishing new workflows and practices for DevOps users to effectively accomplish remediation goals. These users must be given a clear prioritization framework for security issues that arise and desired changes to be made must be clearly communicated. Security and engineering stakeholders need to collaborate to outline prioritization criteria that corresponds to risk levels associated with each security issue. These can be applied to rank issues in priority order to guide DevOps users regarding which ones require immediate attention.

For example, a vulnerability that requires certain privileges to exploit may or may not be ranked as high priority depending on whether those privileges exist for containers in a given application. Or, a vulnerability that is exploited by writing to a container's filesystem may or may not be considered critical if the filesystem is configured as read-only.

One fundamental workflow that can fuel efficient collaboration regarding remediation is to configure alerts and notifications on security issues to be delivered to specific DevOps teams based on the impacted application. As an example, container images are typically made up of multiple layers, with contributions made from multiple individuals or teams, and often built from a base operating system image owned by a particular team. In this scenario, alerts on vulnerabilities or issues with components that exist in particular layers can be directed in a targeted manner to the team that owns the base image, a specific image layer, or certain open source components.

Another example concerns compliance. If checks against industry-standard benchmarks such as payment card industry (PCI) or Health Insurance Portability and Accountability Act (HIPAA) fail, DevOps teams must be given clear remediation guidance to resolve outstanding issues and ensure their applications conform to compliance requirements.

Conclusion

In today's DevOps-driven software environments, security does not fall only under the purview of a centralized security team. Security is everyone's responsibility. Cloud-native technologies, which encompass containers, microservices, immutable infrastructure, and standardized APIs, are the fundamental building blocks driving accelerated software release cycles in combination with the rise of DevOps, CI/CD automation, and agile methodologies. These technologies are driving the software innovations at the heart of business transformation. They also create new challenges when it comes to security.

The changes introduced by cloud-native technologies require organizations to evolve their security toward a DevSecOps model. This means security and engineering teams must work together to develop strategies that successfully help their organizations build and run modern, scalable applications, with "shift left" practices that incorporate security earlier in the software development life cycle and workflows that implement "security as code." The goals of an effective cloud-native security strategy are to allow teams to achieve greater levels of software delivery while building more secure systems. DevOps and software engineers stand to greatly improve security functions in collaboration with security teams that specify policies for tooling, processes, and metrics.

Security and engineering stakeholders can work together to build a shared understanding of key objectives and priorities to protect their organizations' cloud-native applications. Continuous improvement and iteration is critical to any successful DevSecOps effort. Success can be gauged based on key metrics such as trends in the number of security issues, time to remediation, and others. Like the platforms it is designed to secure, DevSecOps is meant to provide a flexible, extensible

approach to enabling organizations to transform how they build, run, and secure software. DevSecOps is a valuable starting point for enabling organizations to more effectively and efficiently use cloud-native technologies such as Kubernetes with significantly greater confidence.

Further reading: Implementing Kubernetes-native security with Red Hat

Security platforms purpose-built to protect Kubernetes offer powerful security and operational advantages. Kubernetes-native security applies controls at the Kubernetes layer, ensuring consistency, automation, and scale. Organizations successfully deploy security as code, enabling security that is built in, not bolted on.

Download this whitepaper—Kubernetes-native Security: what is it and why it matters—to find out more about the key features and benefits of Kubernetes-native security and how it is different from existing container security approaches to deliver protections that are purpose-built for Kubernetes environments.

North America
1 888 REDHAT1
www.redhat.com

Europe, Middle East, and Africa
00800 7334 2835
europe@redhat.com

Asia Pacific
+65 6490 4200
apac@redhat.com

Latin America
+54 11 4329 7300
info-latam@redhat.com

facebook.com/redhatinc
@RedHat
linkedin.com/company/red-hat

redhat.com
#F28077_0421