Getting Started With CI/CD Pipeline Security

CO-UPDATED BY COLLIN CHAU
SENIOR PRINCIPAL PORTFOLIO MARKETING, RED HAT

CO-UPDATED BY MICHAEL FOSTER
PRINCIPAL PRODUCT MARKETING MANAGER, RED HAT

ORIGINAL BY SUDIP SENGUPTA
PRINCIPAL ARCHITECT & TECHNICAL WRITER, JAVELYNN

CONTENTS

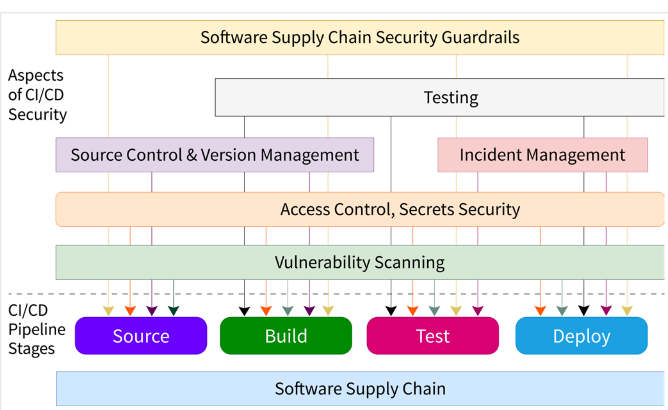
A continuous integration/continuous delivery (CI/CD) pipeline is an agile workflow that automates the code, build, test, and deploy cycles of application delivery. While automated deployment cycles enable developers to release new features and updates rapidly, CI/CD pipelines are commonly targeted by attackers who are looking to exploit vulnerabilities and inject malicious code into application workflows. A compromised pipeline often has severe consequences, such as an attacker gaining access to sensitive data and even controlling the release of new software versions.

In this Refcard, we discuss the key aspects and challenges of securing CI/CD pipelines as well as the fundamental steps to administer security on CI/CD pipelines.

KEY ASPECTS OF SECURING CI/CD PIPELINES

A DevOps workflow is typically characterized by its non-traditional approach to security. This is often because the security of a DevOps workflow is not centralized or does not follow the same approach as other workflows. Instead, securing a DevOps workflow is often distributed among various tools and processes, and across teams.

Figure 1: Key aspects of securing CI/CD pipelines



Securing the CI/CD pipeline at every stage requires a thorough understanding of the core aspects, common threats, and challenges for CI/CD security. Core aspects of CI/CD security include testing, automation, source control, incident management, secrets management, vulnerability scanning, and access control.

TESTING

Continuous application testing helps ensure software security and quality without compromising delivery cycles. Besides inspecting application source code, testing also relies on an iterative cycle of identifying security flaws in third-party libraries, resource-level conflicts, and misconfigurations at code time. It's also important to employ the appropriate testing approaches that inspect flaws across various stages of the CI/CD pipeline. These include:

1. **Static tests** – These tests can be run against code that isn't yet deployed to production, making them extremely fast and easy to automate. However, this testing mechanism

Red Hat
Advanced Cluster
Security Cloud Service

Comprehensive security for cloud-native applications

Red Hat Advanced Cluster Security Cloud Service – a fully managed SaaS solution – enables organizations to securely build, deploy, and run cloud-native applications anywhere.

Supply chain security

Shift security left with developer-friendly guardrails built into the CI/CD pipeline

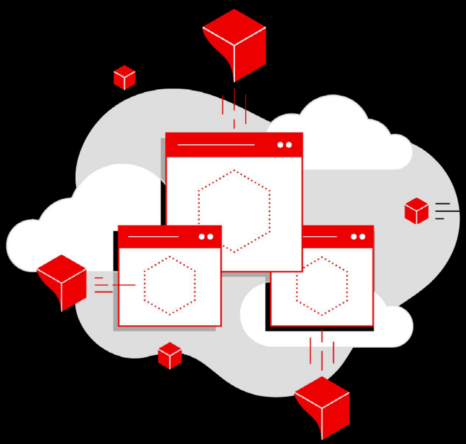
Platform security

Secure the Kubernetes platform leveraging native controls that work with K8s

Workload security

Implement deploy- and runtime policies to protect workloads against threats

Start your trial!



Start your free trial today
red.ht/cloud-native-security-trial

can only test for superficial defects, which lacks offering a comprehensive picture of how the code will actually behave in a production environment.

2. **Dynamic tests** – These tests, on the other hand, inspect code during application runtime. This makes dynamic tests slower and more difficult to automate, but they are efficient at detecting flaws that static tests would normally miss. Dynamic tests are further categorized into:
   - Load testing to ensure that the system can handle heavy traffic
   - Stress testing to identify performance bottlenecks
   - Security testing to check for vulnerabilities
3. **Penetration testing** – This is a proactive approach used in pre-production to simulate real-world attacks and offer valuable insights into an organization's security posture. The testing methodology also helps validate the strength of security controls (e.g., firewalls, antivirus systems).

AUTOMATION

By automating the processes of building, testing, and deploying code, you can ensure that only approved code is deployed to production. Automated enforcement of security controls eliminates errors associated with the manual execution of repetitive tasks while making it easier to track changes and roll back if necessary.

To ensure systems operate on the most secure versions of software, enterprises can also leverage automation for faster roll out of security updates and patches. Besides automatically documenting and recording system vulnerabilities, you can also leverage automation platforms to configure notifications and alerts to flag security threats as soon as they arise.

SOURCE CONTROL

One of the most powerful ways of enforcing code integrity is to use source control systems that enable enterprise teams to securely manage code changes, collaborate with cross-functional teams, and resolve conflicts in code before committing changes. This approach also helps prevent accidental or malicious changes from being introduced into the codebase, which could potentially break the build or cause other problems downstream.

Also commonly referred to as ***version control***, source control involves the configuration of access permissions to the codebase, ensuring only the approved contributors are allowed to make code changes. This guarantees that only authorized users have access to the codebase and that all changes are tracked and audited.

Additionally, by using a centralized source control repository, you can more easily automate code reviews and roll back changes to a safe, declarative state if something does go wrong.

INCIDENT MANAGEMENT

Incidents are unplanned events that disrupt normal operations by compromising the integrity of a system. In the context of CI/CD pipelines, incidents can range from simple build failures to more complex security breaches. Consequently, it's essential to formulate an incident management process that encompasses various procedures and tools to manage and respond to security events.

While the primary purpose of an incident management framework is to reduce the impact of an event, it also helps alleviate the future occurrence of similar incidents by helping recognize identical patterns and fine-tuning alerting systems for expedited response. A typical approach is also to hard-code incident response plans into workflow tools, allowing for the automatic remediation of CI/CD security threats.

SECRETS MANAGEMENT

Managing secrets involves practices and procedures to securely manage, store, and transmit confidential credentials, including encryption keys, API keys, passwords, session tokens, database connection strings and certificates. Effectively administered secrets management maintains a fine balance between the ease of injecting secrets and limiting data exposure. This essentially implies that sensitive data remain confidential, while services can autonomously use secrets to interconnect with other services or tools.

There are a few key things to keep in mind when managing secrets for CI/CD pipelines:

- Always use strong encryption for storing and transmitting secrets. This will help ensure that even if a malicious user gains access to your secrets, they will not be able to read or use them.
- Be sure to rotate your secrets regularly. This will help prevent attackers from using old secrets that they may have discovered.
- Make sure that only authorized users have access to your secrets. This can be accomplished through role-based access control (RBAC) or other authorization mechanisms.
- Use environment variables to store secrets as part of your application code. This approach allows you to keep secrets out of your code repository that prevent deeper compromise of the system.

VULNERABILITY SCANNING

Automated vulnerability scanning helps teams enforce a ***shift-left*** approach for security by identifying and remediating threats from early stages of a development cycle. Remediating vulnerabilities typically involves development teams detecting a flaw, assessing its impact and severity, deploying a fix, and performing a determinative scan to ensure the flaw no longer exists.

Since CI/CD pipelines are composed of numerous components and dependencies, vulnerability scanning for CI/CD is oftentimes broken down into:

- Source code scanning
- Third-party dependency scanning
- Container image scanning
- Infrastructure component scanning

To ensure all misconfigurations are appropriately attributed with their impacts, a common practice is also to leverage databases of known weaknesses. Some popular vulnerability databases include:

- Common Weakness Enumerations
- National Vulnerability Database
- OWASP's Top 10 CI/CD Vulnerabilities List

ACCESS CONTROL

One of the most important aspects of CI/CD security is making sure all cluster endpoints are secured. Access control mechanisms help mitigate the risk of data breaches by determining who has the privileges to access specific data and resources of a pipeline.

Administering stricter policies requires users to verify their identity before they are allowed to access sensitive information. Beyond verifying a user's identity, access control policies also determine the allowed actions by defining permissions granted for each user.

COMMON CI/CD PIPELINE SECURITY THREATS

As per OWASP, although there are emerging practices and tools to avert security incidents, attackers continue to adapt novel techniques that exploit the distributed complexity of a CI/CD framework.

Some common security threats of CI/CD pipelines include:

- **Distributed denial-of-service** (DDoS) attacks are orchestrated by compromising the server, network, or service by overwhelming it with a high number of requests/internet traffic in a given time.
- **Supply chain** attacks focus on weak links in trusted third-party vendors that offer tools and services to the CI/CD pipeline.
- **Dependency confusion** attacks abuse flaws within package managers to replace legitimate private packages with malicious versions in public registries.
- **Injection** attacks are exploited over input validation errors to inject unauthorized code into the application, which ends up interpreting it as part of a command or a query.
- **Remote code execution** attacks are widely exploited attacks executed through malicious code on remote machines by connecting to them over insecure public and private networks.

Table 1

COMMON CI/CD PIPELINE SECURITY THREATS		
THREAT	**TARGET CI/CD PIPELINE STAGE**	**ATTACK PATTERN**
DDoS attacks	Deployment	Leveraging botnets to target the victim server/network, overwhelming it and resulting in a denial of service
Supply chain attacks	Build	Injecting malicious code into an open-source component to compromise the entire tech stack
Dependency confusion attacks	Source and build	Registering a package with a similar name to the target app of a public repository, which gets committed to the pipeline every time a new install occurs
Injection attacks	Deployment	Altering request URLs to change the parameters of the resulting database query, consequently enabling unauthorized access of restricted data
Remote code execution attacks	All	Tricking the target user to install arbitrary scripts on the host machine, which are subsequently executed to orchestrate deeper, system-level attacks

CHALLENGES WITH SECURING CI/CD PIPELINES

Securing CI/CD is a complex practice that encompasses the identification, remediation, and prevention of security risks across each stage of a pipeline. While building a robust security posture is the fundamental objective of the practice, the framework should also continue to maintain the agility and pace of release cycles. As a result, when compared to securing legacy frameworks, there are a number of challenges with administering security on CI/CD pipelines, including:

- Improper secrets management
- Inconsistent approaches to microservices
- Inadequate security automation
- Conflicts between security and velocity
- Unauthorized access to code registries
- Developer and DevOps resistance

ADMINISTERING COMPREHENSIVE SECURITY ON CI/CD PIPELINES

Apart from protecting data and code from potential breaches that traverse through various endpoints of the pipeline, administering security on CI/CD pipelines also helps maintain compliance and prevent accidental issues such as data loss or corruption. In the following section, we discuss the steps for effective CI/CD security implementation and open-source tools to simplify the process.

STEPS TO ENSURE CI/CD PIPELINE SECURITY

While the specifics of a CI/CD pipeline security strategy will differ by use case, the process typically follows a similar workflow.

1. IMPLEMENT STRONG ACCESS CONTROLS

The first step toward securing a pipeline is to control and organize access privileges. This essentially requires policy enforcement that restricts every user of the organization to possess similar privileges for accessing tools and resources within the CI/CD pipeline. Additionally, those with permissions to access the pipeline should not be assigned default permissions to view all resources and data within the pipeline.

Some approaches to help enforce access controls include:

- **Configure identity and access management** (IAM) – helps configure digital identities and enforce access permissions at the entity level
- **Enforce role-based access controls** (RBACs) – restricts users to access data and resources based on the functions/tasks associated with their roles
- **Apply the principle of least privilege** – limits a user's access rights to strictly what is required to perform their job

2. SECURE ACCESS TO CODE REPOSITORIES

Since a code repository acts as the central storage, review, and management system of the code used within a DevOps pipeline, securing repos is the next step that requires key consideration. Public code repos, or those lacking secure controls, are often targets of malicious exploits that lead to code tampering and loss of code integrity.

Approaches to securing code repositories include:

- Choose a trusted repository by providers with a reputation for secure infrastructure administration and management
- Enforce the principle of least privilege for repository access
- Secure access credentials and separate them from source code
- Revoke access to the repository when it is no longer required
- Review all code changes before merging to the main branch
- Conceal personally identifying information when using public repositories
- Enforce backup and disaster recovery for all code used within the system
- Perform regular audits against security benchmarks

3. AVOID HARD-CODING SECRETS

Hard-coded passwords and secrets are common attack targets that lead to data breaches and malicious access of pipeline resources. Attackers typically target source codes within public repos and identify hard-coded credentials through code scanning, guessing, and learning. As a recommended practice, security admins should implement policies to regulate the usage of hard-coded secrets into application code. If secrets are to be parsed, they should be included as variables in a `.gitignore` file, which keeps them from being committed into the repository. For instance, before distributing secrets in a Kubernetes cluster, secrets should first be encrypted at rest and then stored in the ETCD server.

A conventional approach for achieving this is by encoding the secrets in Base64 format as shown:

```
$ username=$(echo -n "default" | base64)
$ password=$(echo -n "a62fjbd37942dcs" | base64)
```

And then, defining the secrets:

```
echo "apiVersion: v1
> kind: Secret
> metadata:
>   name: darwin-secret
> type: Opaque
> data:
>   username: $username
>   password: $password" >> secret.yaml
```

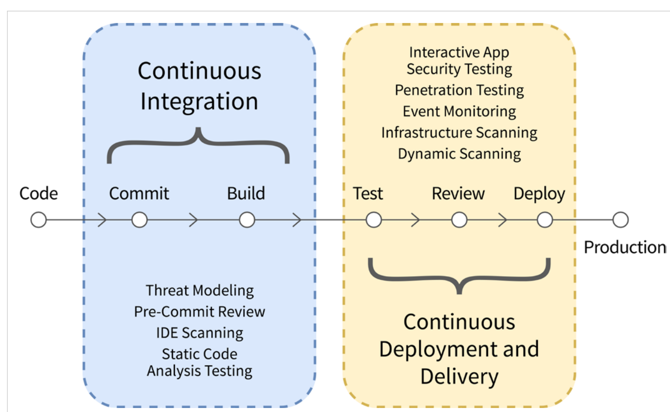
Following the above, you can now create the secret using the `kubectl create` command:

```
$ kubectl create -f secret.yaml
```

4. PERFORM APPLICATION SECURITY TESTS

Once code repositories are secure and secrets are safely managed, developers and security teams should collectively ensure the source code is free of any vulnerabilities. This is accomplished through a combination of tests that are deployed at each layer of CI/CD workflows to automatically notify security teams upon detecting pipeline vulnerabilities.

Figure 2: Security testing of a CI/CD pipeline



Automated tests can also be combined with automated remediation tools that use the findings of security checks to safeguard pull requests from attack vectors. In production-grade pipelines, a common approach is also to engage external penetration testers to provide an unbiased view of the pipeline's security posture and help identify flaws that may have been missed by automated tests.

5. IMPLEMENT SECURITY-FOCUSED CI/CD WORKFLOWS

With pipeline steps and tasks stored in the same repository as their code, DevOps practices automate continuous deployment to an auditable, declarative state to avoid human error and misconfiguration. Building security into container images is an integral part of a CI/CD pipeline as it continuously scans for code integrity. Auto-generated software bill of materials (SBOMs) for provenance can now attest and verify open-source software components and their transitive dependencies and take away manual toil.

Development teams need to stay consistent with industry standards like Supply-chain Levels for Software Artifacts (SLSA). It is essential to use default pipeline definitions with policy as code to prevent deployment of suspicious build activity to production.

6. USE ROLLBACKS TO ENFORCE SECURITY IN PRODUCTION PIPELINES

Once policies are framed to secure pipelines, the next stage focuses on minimizing the consequences of a successful attack. This requires the formulation of controls that help revert to earlier, stable versions of an application if the current one is compromised. The ability to quickly roll back an insecure application version also helps reduce application downtimes while expediting patch cycles for faster remediation.

7. DETECT AND REMEDIATE THREATS AT RUNTIME

Beyond securing the pipeline and hardening the deployment, the next stage focuses on detecting and thwarting a successful attack. This requires having granular visibility into all process executions, network flows, etc. to detect anomalous activity indicative of a threat. By combining policy-based detection for common threats — such as cryptomining, privilege escalations, and other exploits — with process baselining of known, validated behavior, you can improve detection fidelity and minimize false positives.

Since containers are immutable and applying patches to the running environment will be temporary, it's critical that remediation efforts happen in the pipeline that you're rebuilding and redeploying to your containerized applications.

8. OUTLINE AN INCIDENT RESPONSE PLAN

Incident response plans strengthen a continuous testing process by shortening the feedback loop of identifying and addressing CI/CD security threats. Once potential security threats have been mapped with their respective attack vectors, the incident response plan should outline tools and processes to be used to restore normal operations.

Besides reducing the response time for a security event, response plans should also tag a summary of related non-critical incidents that may signal potential issues within the application, thereby helping developers to fine-tune their code for security and performance.

9. LEVERAGE A SECURITY INFORMATION AND EVENT MANAGEMENT TOOL

Security information and event management (SIEM) tools go beyond incident response plans by offering granular indicators of various events. For CI/CD security, SIEM tools perform three critical capabilities: threat detection, event investigation, and response time reduction.

These tools aggregate and analyze telemetry data from different resources of the CI/CD pipeline. The composite data is then stored, normalized, and analyzed for threat detection and trend analysis. When configuring an SIEM solution, security testers and developers should also integrate a continuous testing and monitoring framework for faster discovery of security breaches and remediation.

GETTING STARTED: BUILD AN OPEN-SOURCE STACK FOR CI/CD PIPELINE SECURITY

Securing a CI/CD pipeline is a multi-pronged process that requires an in-depth understanding of the tech stack's core aspects, changing threat patterns, and inherent vulnerabilities.

Out of the number of tools available, below is a list of popular open-source tools that are free, simplify the implementation of CI/CD security, and offer comprehensive hardening solutions:

Table 2

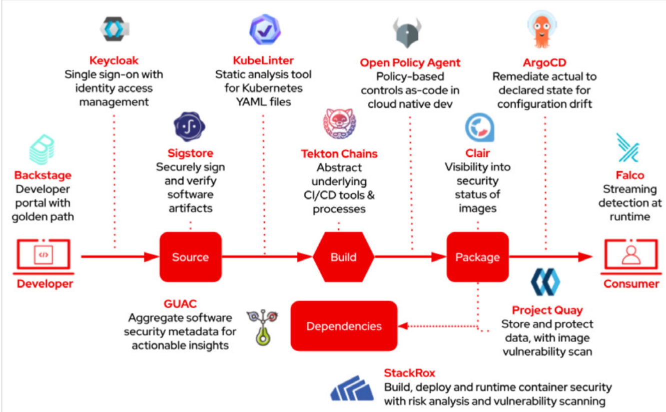
OPEN-SOURCE TOOLS TO SECURE CI/CD PIPELINES

TOOL	DESCRIPTION
Argo CD	• Ensures robust security measures throughout the CI/CD process, specified as code in the same git repo • Automates continuous deployment to an auditable, declarative stage without human error • Follows GitOps patterns to monitor running apps from their live state to a desired state
Backstage	• Build self-service developer portals with a centralized software catalog and community plug-ins • Make use of standards based templates as golden paths to auto-create security-focused microservices
Clair	• Monitor container security with static analysis of vulnerabilities in apps and docker containers • Understand the impact radius of emerging CVEs to alert on which existing layers are vulnerable
Falco	• Threat detection in hosts and containers, and across the cloud, to remain regulatory compliance • Cloud-native security tools for Linux systems that employ custom rules and metadata for real-time alerts • Streaming detection at runtime that monitors for abnormal behavior, configurations, and attacks
GUAC	• Aggregates and synthesizes software security metadata at scale, making it meaningful and actionable • Identifies gaps and threats in the software supply chain and provides a path to remediation
HashiCorp Vault	• Secrets management and data protection tool • Securely stores and manages sensitive information (e.g., credentials, encryption keys, API tokens)

OPEN-SOURCE TOOLS TO SECURE CI/CD PIPELINES	
TOOL	**DESCRIPTION**
Keycloak	• IAM built on top of industry-security-standard protocols for modern apps/services • Authentication using single sign-on with user federation and management, and fine-grained authorization
KubeLinter	• Static analysis tool for Kubernetes YAML files • Identifies misconfigurations and security issues in Kubernetes configurations • Proactively identifies and rectifies security risks in Kubernetes deployments, improving overall cluster security
Open Policy Agent	• Flexible policy-driven control plane for cloud-native environments • Enables fine-grained policy enforcement across entire cloud-native stack • Helps enforce security policies, compliance, and access control
OWASP Dependecy-Check	• Assesses vulnerabilities in software project dependencies • Scans and identifies known security issues in third-party libraries and frameworks, aiding in informed decision-making to mitigate risks
Project Quay	• Includes the Clair vulnerability scanner and provides a secure container registry • Offers container image vulnerability analysis • Ensures container integrity and safety • Reduces the risk of deploying vulnerable containers into prod environments
Project Calico	• Networking solution for container-native deployments • Enforces zero-trust, endpoint-level security through `GlobalNetworkPolicies` to help secure containerized hosts and workloads • Helps secure in-cluster pod traffic with on-the-wire encryption, enforcing data integrity without requiring specialized hardware
Sigstore Cosign	• Enhances container image and software artifact security via digital signing • Allows developers to sign and verify code and container image integrity and authenticity • Alleviates concerns of tampering and ensures software's trustworthiness
Sigstore Rekor	• Digital notary service for CI/CD pipeline code • Ensures code authenticity and security, verifies origins of third-party dependencies, offers transparency, and automates verification • With Gitsign, implements keyless Sigstore to sign Git commits with a valid OpenID Connect identity to tamper proof code • Stores signing details in the Rekor transparency log for subsequent verification

OPEN-SOURCE TOOLS TO SECURE CI/CD PIPELINES	
TOOL	**DESCRIPTION**
SonarQube	• Tests against the most critical risk categories in app code • Performs a static analysis of pull requests to ensure all code entering the pipeline is free of threats found on the OWASP Top 10 list of vulnerabilities • Relies on a taint analysis mechanism to track and detect malicious inputs in the DevOps workflow • Offers an issue visualizer to inspect how vulnerabilities flow within pipelines and guidance to identify root causes and enforce stricter controls
StackRox	• Integrates security directly into Kubernetes ecosystems and CI/CD pipelines, providing real-time threat detection, risk assessment, and policy enforcement • Provides comprehensive security visibility and automated responses to protect from vulnerabilities, misconfigurations, and runtime threats • Adheres to compliance standards (e.g., CIS Benchmarks, PCI-DSS, HIPAA, NIST) • Maps its functionalities to the MITRE ATTt&CK Framework for Kubernetes
Syft	• Analyzes container images and filesystems • Performs comprehensive software component inspections to identify vulnerabilities and report on software bill of materials • Shores up containerized app security posture, ensuring containers are free from known vulnerabilities
Tekton Chains	• Cloud- and Kubernetes-native CI/CD framework • Isolates and segments pipeline stages, enforces immutability to prevent unauthorized changes, and maintains audit trails • Manages secrets securely, offers fine-grained access control, and integrates security scanning

Figure 3: Leverage open-source projects for security and compliance



CONCLUSION

Using CI/CD systems for production releases is one of the most commonly established practices in modern application delivery. With increasing adoption of DevOps practices, the foundational security of CI/CD pipelines has come under greater scrutiny because they are often the gateway to an organization's codebase and deployments, making them a common target for attackers, especially within their software supply chain. There is clear evidence that security is a key concern in cloud-native environments and Kubernetes, especially when it comes to vulnerabilities and misconfigurations. Traditional security measures to administer security on CI/CD-based workflows are often insufficient. Consequently, a DevOps practice relies on the implementation of granular policies across every stage of pipelines for comprehensive security. Taking steps to integrate security guardrails at every phase of the SDLC will help you audit and act on security vulnerabilities early before it impacts user trust.

Additional resources:

- *Continuous Delivery Pipeline Security Essentials* Refcard by Sudip Sengupta
- *Advanced Cloud Security* Refcard by Samir Behara
- *Threat Detection for Containers* Refcard by Boris Zaikin
- *Cloud-Native Application Security Patterns* and Anti-Patterns Refcard by Samir Behara
- *IaC Security Core Practices* Refcard by Payton O'Neal

CO-UPDATED BY COLLIN CHAU,

SENIOR PRINCIPAL PORTFOLIO MARKETING, RED HAT

Collin has scaled DevOps engineers in their continuous testing and application release automation tooling, while enabling ITOps teams to bridge and broker their hybrid cloud to deliver features faster. While running AI/ML-assisted app workloads over a data-driven platform, Collin has helped SREs monitor and direct their IT estate's health and performance to predict and prevent service failures at runtime.

CO-UPDATED BY MICHAEL FOSTER,

PRINCIPAL PRODUCT MARKETING MANAGER, RED HAT

Michael Foster is the Community Lead for the open source StackRox project and Principal Product Marketing Manager for Red Hat. Michael wants organizations to leverage the open source StackRox project in their Kubernetes environments and join the open source community through stackrox.io. He holds a B.S. in Chemical Engineering from Northeastern University and CKAD, CKA, and CKS certifications.

WRITTEN BY SUDIP SENGUPTA,

PRINCIPAL ARCHITECT & TECHNICAL WRITER, JAVELYNN

Sudip Sengupta is a TOGAF Certified Solutions Architect with more than 18 years of experience working for global majors such as CSC, Hewlett Packard Enterprise, and DXC Technology. Sudip now works as a full-time tech writer, focusing on Cloud, DevOps, SaaS, and cybersecurity. When not writing or reading, he's likely on the squash court or playing chess.

3343 Perimeter Hill Dr, Suite 100
Nashville, TN 37211
888.678.0399 | 919.678.0300

At DZone, we foster a collaborative environment that empowers developers and tech professionals to share knowledge, build skills, and solve problems through content, code, and community. We thoughtfully — and with intention — challenge the status quo and value diverse perspectives so that, as one, we can inspire positive change through technology.