sysdig

30-60-90 Day Checklist for a DevOps Engineer

Who is this guide for?

Target Audience:

- Senior/Lead DevOps Engineer
- DevOps Manager
- Director of Engineering
- VP of Cloud Platforms
- CTO

Your team is responsible for ensuring faster application delivery. You're probably starting to adopt open-source tools that support the "shift left" in security best practices, as well as evaluating containerized and serverless architectures to deliver applications. Your team's responsibilities range from monitoring performance and availability, to troubleshooting incidents and managing security and compliance.

As your team grows, this guide will provide a well-defined framework to successfully onboard and make a new DevOps/DevSecOps engineer productive within the first 90 days.

Let's get started!

30 DAY GOAL

See the Big Picture

Crawl

Milestone 1

Gain an overview of the cloud-native landscape

- Understand the Software Delivery Lifecycle (SDLC) framework and how code gets pushed to production (*see appendix for cloud-native tool landscape*).
- Understand the application architecture.
  - Be aware of the application tech stack (*e.g., Java, Go, Cassandra, Redis, etc.*). This is important to know even for non-developers.
  - Be aware of the technical debt in the engineering organization.
- Be aware of container technologies used by your organization. Examples below:
  - Container runtimes: (e.g., Docker, crio, containerd, etc.)
  - Orchestrators: (e.g., Kubernetes, Red Hat OpenShift, Rancher, etc)
  - Cloud Providers (*e.g., AWS, Azure, IBM etc*) and managed Kubernetes services (*ex. EKS, GKE, AKS etc*)
  - Serverless frameworks: (ex. AWS Fargate, Google Cloud Run etc)

Related Resources:

- CNCF Landscape
- Densify's CI/CD framework
- 2019 Container Usage Report

Milestone 2

Develop a strong grasp of the business initiatives and how they translate to cloud and cloud-native project goals

- Understand what applications are currently containerized in your environment and what is the next 6-18 month transformation plan to containers/Kubernetes

Milestone 3

Align on shared goals with both internal and cross-functional stakeholders

- Make sure you shadow/spend time with the following:
  - DevOps teammates (shadow someone that is in charge of a production release)
  - Developers
  - Security team
  - Infrastructure operations
  - Product Managers
  - Vendor management and procurement

Milestone 4

Get access to container monitoring and security tools

- Monitoring tool access: Prometheus, Sysdig Monitor
  - Dig into what dashboards are currently used for visualization
  - Understand the alerting process
- Security tool access: Falco, Open Policy Agent (OPA) Gatekeeper, Sysdig Secure

Bonus Milestone
Make a contribution if you can!

- ☐ Open a pull request (PR) in the first 30 days
- ☐ Start contributing ideas to generate new project ideas

A few projects on GitHub:

- https://github.com/kubernetes/kubernetes
- https://github.com/prometheus/prometheus
- https://github.com/falcosecurity/falco

Key members of the community to follow:

- https://twitter.com/kelseyhightower
- https://twitter.com/ahmetb
- https://twitter.com/jbeda
- https://twitter.com/craigmcl
- https://twitter.com/brendandburns
- https://twitter.com/brendangregg

60 DAY GOAL

Understand the Current State

Walk

Milestone 1
Start taking on day-to-day responsibilities

- [ ] Deliver features to production (e.g., bug fixes, minor changes/improvements, leveraging previous expertise to make prominent contributions to production).
- [ ] Become an active participant during sprint planning and sizing, and aim to run at least one feature demo and/or brown bag (assuming these initiatives exist, if not, suggest starting regular lunch and learn sessions!).
- [ ] Become heavily involved in day-to-day operations (e.g., ongoing health and maintenance of the container and K8s platform, bug fixes, upgrade/patch process, etc.).
- [ ] Participate in design and strategy meetings, including:
  - [ ] Improvements to overall infrastructure architecture
  - [ ] Scaling systems (e.g., networking, applications, load balancers, etc.)
- [ ] Regularly be on call/respond to outages.
- [ ] Contribute to documentation.
- [ ] Conduct peer review/code reviews on pull requests (PR).

Related Resources:

- Building Secure and Reliable Systems
- The DevOps Handbook
- Clean Architecture
- The Unicorn Project
- Running containers in production
- Linux Observability with BPF

Milestone 2
Understand the current state of automation and immutable infrastructure

- ☐ Understand what automation tools are being used in production (e.g., Puppet, Chef, Ansible, etc.).
- ☐ Understand the Infra as code (IAC) and security as code (SAC) process.
- ☐ Consider any potential improvements around immutability of any components.
- ☐ Validate the immutability of containers in use (e.g., they aren't being changed in-place over time instead of versioned and redeployed)

Related Resources:

- The Top 7 Infrastructure-As-Code Tools For Automation
- GitHub Actions
- GitOps Security with k8s-security-configwatch

Milestone 3
Understand how containers/Kubernetes environments are monitored

- ☐ Be involved in benchmarking as well as load testing and performance testing. Consider including highly complex systems for performance testing.
- ☐ Observe troubleshooting methodologies being used to respond to an incident.
- ☐ Get comfortable with the incident management process and be aware of the process for remediation/post-mortem analysis.
- ☐ Get familiar with tools to monitor cloud providers (*e.g.*, *AWS*, *Azure*, *GCP*, *etc.*).

Related Resources:

- Systems Performance: Enterprise and the Cloud
- Cloud Native Infrastructure: Patterns for Scalable Infrastructure and Applications in a Dynamic Environment
- Prometheus: Up & Running
- Prometheus for the enterprise
- Kubernetes Up and Running
- Kubernetes Monitoring Guide

Milestone 4
Understand how containers/Kubernetes environments are secured

- Be knowledgeable about the key metrics/KPIs that are important to the security team.
- Be familiar with how security and compliance policies are implemented.
  - What are existing security policies in place?
  - Who sets security and compliance policies? Infosec? Central IT? DevSecOps engineers?
- Start getting training on container and Kubernetes security:
  - Learn the image scanning process pre-deployment and reporting of new vulnerabilities at runtime.
  - Understand the compliance regulations (*e.g.*, *PCI*, *SOC2*, *FedRAMP*, *NIST*, *HIPAA*, *GDPR*, *etc.*) that your containers and clusters need to meet.
  - Learn what network security policies are enforced for both east-west and north-south traffic.
  - Understand the runtime security mechanism to detect threats.
  - Learn the incident response playbook to respond to container specific attacks.
- Educate other team members about the above security best practices to promote a strong security first mindset.

Related Resources:

- Five Things CISOs Can Do To Make Containers Secure And Compliant
- The Kubernetes Ship Has Set Sail: Is Your Security Team On Board?
- 5 Keys to a Secure DevOps Workflow

90 DAY GOAL

Define a Secure DevOps Workflow

Run

Milestone 1

Define a framework for automation and immutable infrastructure

- Focus on repeatability of builds, deploys, etc.
  - Be aware of the negative consequences: How long will it take you to rebuild your production environment if it was tampered with?
- Standardize on an infrastructure-as-code tool (e.g., Terraform, CloudFormation (CFT), etc.)
- Adopt secure GitOps principles:
  - Use security as code in your infrastructure to define alerts, rules and policies that are committed into a git repository.
  - Trigger image scanning directly from your github repository and prevent vulnerable images from even going into a registry.
  - Ensure a PR triggers a sanity check of the Kubernetes security configuration early to avoid misconfigurations.

Related Resources:

- Announcing the enhanced Sysdig Terraform Provider
- Image Scanning with Github Actions
- GitOps Security with k8s-security-configwatch

Milestone 2
Define a framework for maximizing performance and meeting availability SLA's

- Understand the Service Level Indicators (SLIs) and other metrics/KPIs that inform the objectives to define the Service Level Agreement (SLA) held accountable:
  - Understand the Operational Level Agreement (OLAs) between the different teams working with one another in order to meet the SLAs.
- Become adept with observability techniques: performance, infrastructure and application monitoring, logging, tracing, troubleshooting, capacity management, etc.
- Be able to diagnose a severity one customer issue or breach with existing monitoring and security tools, and resolve issues quickly.
- Look for opportunities to improve overall infrastructure observability.
  - Become comfortable with monitoring/alerting systems - defining alerts, build dashboards, etc.
  - What are the signals to look for when it comes to Kubernetes?
  - Identify all areas of friction that slows code delivery to production.
- Contribute and learn from the community.
  - Stay up to date with the latest Prometheus best practices here: https://promcat.io

Milestone 3
Define a framework to manage security risk and validate compliance

- Define RBAC controls for specific teams/users.
- Build a security mentality early on for containers:
  - Before a developer pushes code to production, ensure security checks (e.g., vulnerabilities, misconfigurations and best practices) are embedded into the CI/CD pipeline.
  - Work very closely with the InfoSec, Operations and Engineering teams to craft the security policies.
  - DevOps teams need to implement these security policies, even if they are not pushing code to production.
- Ensure your team is meeting compliance guidelines.
  - Regularly sync with the compliance team to ensure you have a roadmap to achieve compliance attestations (SOC2, PCI, NIST, GDPR, etc.).
- Ensure you have robust runtime security controls (e.g., threat detection, network security controls and prevention mechanisms) in place and review/improve periodically.
- Do we have an incident response framework to follow?
  - Ensure a mitigation plan is in place to ensure the same issue doesn't happen twice.
  - Develop methods to track, prioritize, escalate and report security findings.
  - Take ownership of and drive resolution of security issues.
- Contribute and learn from the community.
  - Stay up to date with the latest cloud-native security best practices here: https://securityhub.dev

Related Resources:

- Kubernetes Security Guide
- PCI Compliance for containers and Kubernetes
- IDC's guide to evaluating container security tools

Appendix

Cloud-Native Tool Landscape