Two-Month DevOps/SRE Learning Plan A Comprehensive Guide to Mastering DevOps and SRE

Skills July 8, 2025

Contents

1 Introduction

This 8-week plan prepares you for a DevOps/SRE role involving Kubernetes, GitOps, observability, CI/CD, and cloud technologies. It covers all job requirements (mandatory, preferred, nice-to-have), addresses gaps in DevOps/SRE practices (e.g., performance optimization, security, incident management), and includes hands-on projects and interview preparation.

2 Assumptions

- **Background**: Basic knowledge of Linux, Python/Bash, and Docker.
- **Resources**: Linux machine (or WSL2), AWS/GCP free-tier accounts, GitHub account.
- **Time Commitment**: 4–6 hours/day (2–3 hours theory, 2–3 hours practice).
- **Goal**: Master technical skills, prepare for interviews (coding, system design, troubleshooting), and excel in the job.

3 Plan Structure

- **Weeks 1–2**: Kubernetes, Docker, GitOps, Kustomize, Helm, scripting, networking, security.
- **Weeks 3–4**: Observability (Prometheus, Grafana, Loki, Tempo, Jaeger/OpenTelemetry).
- **Weeks 5–6**: Service mesh (Istio), CI/CD (Jenkins, ArgoCD, Argo Workflows), SRE principles.
- **Weeks 7–8**: Cloud (AWS/EKS, GCP/GKE), Terraform, databases, chaos engineering, interview prep.
- **Tools**: Minikube/Kind, Docker, kubectl, Kustomize, Helm, ArgoCD, Prometheus, Grafana, Loki, Tempo, Jaeger, OpenTelemetry, Istio, Jenkins, Argo Workflows, AWS CLI, eksctl, GCP SDK, Terraform, Velero, LitmusChaos, Trivy.

4 Week 1: Kubernetes, Docker, Networking, and Security

Goal: Build a strong foundation in Kubernetes, Docker, scripting, and security.

4.1 Day 1: Kubernetes Basics

- **Duration**: 6 hours
- **Learn**: Kubernetes architecture (control plane, nodes, pods, services, deployments).
  - Resources: Kubernetes Docs, FreeCodeCamp Kubernetes Course.
- **Practice**:
  - Install Minikube or Kind.
  - Deploy an Nginx pod using `kubectl apply -f nginx.yaml`.

  - Expose as a service, access via `curl`.
- **Task**: Document `kubectl` commands in README.

4.2 Day 2: Docker Deep Dive

- **Duration**: 6 hours
- **Learn**: Docker images, containers, Dockerfile, registries.
  - Resources: Docker Get Started.
- **Practice**:
  - Build a Docker image for a Python Flask app.
  - Push to Docker Hub, run locally, verify with `docker ps`.
- **Task**: Add Dockerfile and build script to Git repo.

4.3 Day 3: Kubernetes Networking and Security

- **Duration**: 6 hours
- **Learn**: CNI plugins (e.g., Calico), Network Policies, RBAC, Pod Security Standards, image scanning.
  - Resources: Kubernetes Networking, Calico Docs, Trivy Docs.
- **Practice**:
  - Configure a Network Policy to restrict pod traffic.
  - Set up RBAC for a user.
  - Scan a Docker image with Trivy.
- **Task**: Document Network Policy and RBAC configs in README.

4.4 Day 4: Python and Bash Scripting

- **Duration**: 6 hours
- **Learn**: Automation scripts for DevOps tasks.
  - Resources: Automate the Boring Stuff, Bash Guide.
- **Practice**:
  - Write a Bash script to automate Docker builds.
  - Write a Python script to list pods using `kubernetes` client.
- **Task**: Add scripts to Git repo.

4.5 Day 5: Performance Optimization

- **Duration**: 6 hours
- **Learn**: Kubernetes resource quotas, limits, Horizontal Pod Autoscaler (HPA).
  - Resources: Kubernetes Resource Management.
- **Practice**:
  - Set CPU/memory limits for a pod.
  - Configure HPA based on CPU usage.
- **Task**: Document resource configs in README.

4.6 Day 6: Mini-Project

- **Duration**: 6 hours
- **Practice**:
  - Deploy a Flask app to Minikube with resource limits, Network Policy, and RBAC.
  - Scan image with Trivy, automate deployment with a Bash script.
- **Task**: Create a Git repo section for the project, document in README.

4.7 Day 7: Review and Troubleshooting

- **Duration**: 6 hours
- **Practice**:
  - Simulate a pod failure (e.g., crash loop).
  - Debug using `kubectl logs/describe`, fix issue.
- **Task**: Add troubleshooting log to README.

Tools: Minikube/Kind, Docker, kubectl, Python, Bash, Git, Trivy.

5 Week 2: GitOps, Kustomize, Helm, and Developer Experience

Goal: Master GitOps workflows, Kustomize, Helm, and developer collaboration.

5.1 Day 8: GitOps and ArgoCD

- **Duration**: 6 hours
- **Learn**: GitOps principles, ArgoCD for continuous deployment.
  - Resources: Weaveworks GitOps, ArgoCD Docs.
- **Practice**:
  - Install ArgoCD on Minikube.

    - Deploy a sample app via a Git repo, simulate updates.
- **Task**: Document ArgoCD setup in README.

5.2 Day 9: Kustomize

- **Duration**: 6 hours
- **Learn**: Declarative configuration with Kustomize.
    - Resources: Kustomize Docs.
- **Practice**:
    - Create Kustomize manifests for a multi-environment (dev/staging) Flask app.
- **Task**: Add manifests to Git repo, document in README.

5.3 Day 10: Helm

- **Duration**: 6 hours
- **Learn**: Helm charts and package management.
    - Resources: Helm Docs.
- **Practice**:
    - Package a Flask app as a Helm chart.
    - Deploy to Minikube, customize with values.yaml.
- **Task**: Add Helm chart to Git repo, document in README.

5.4 Day 11: Developer Experience and CNCF Tools

- **Duration**: 6 hours
- **Learn**: Writing developer guides, Fluentd vs. Loki comparison.
    - Resources: Fluentd Docs.
- **Practice**:
    - Write a guide for developers to deploy apps with Kustomize.
    - Experiment with Fluentd for log collection.
- **Task**: Add guide and Fluentd notes to README.

5.5 Day 12: Mini-Project

- **Duration**: 6 hours
- **Practice**:
    - Deploy a Flask + Redis app using Kustomize, Helm, and ArgoCD in a GitOps workflow.
    - Include a developer guide.

- **Task**: Document the project in README.

5.6 Day 13–14: Review and Troubleshooting

- **Duration**: 8–12 hours
- **Practice**:
  - Simulate a deployment failure (e.g., invalid Helm chart).
  - Debug using ArgoCD UI and `kubectl`.
- **Task**: Add troubleshooting notes to README.

Tools: ArgoCD, Kustomize, Helm, Fluentd.

6 Week 3: Observability Foundations (Prometheus, Grafana)

Goal: Learn monitoring and metrics.

6.1 Day 15–16: Prometheus

- **Duration**: 12 hours
- **Learn**: Prometheus architecture, metrics, PromQL, alerting.
  - Resources: Prometheus Docs, Robust Perception blog.
- **Practice**:
  - Install Prometheus via Helm.
  - Configure metrics scraping for a Flask app.
  - Set up alerts for high CPU usage.
- **Task**: Document Prometheus setup and alerts in README.

6.2 Day 17: Grafana

- **Duration**: 6 hours
- **Learn**: Grafana dashboards and visualization.
  - Resources: Grafana Docs.
- **Practice**:
  - Install Grafana, create dashboards for app metrics (CPU, memory, HTTP requests).
- **Task**: Add dashboard screenshots to README.

6.3 Day 18: Performance Optimization and Troubleshooting

- **Duration**: 6 hours
- **Learn**: Troubleshooting with metrics, HPA.
  - Resources: Kubernetes HPA Docs.

- **Practice**:
  - Configure HPA for a Flask app.
  - Simulate high load with `stress`, debug with metrics.
- **Task**: Document HPA setup and troubleshooting in README.

6.4 Day 19: Mini-Project

- **Duration**: 6 hours
- **Practice**:
  - Deploy a microservices app, monitor with Prometheus/Grafana, set up alerts and HPA.
- **Task**: Document the project in README.

6.5 Day 20–21: Review and Catch-Up

- **Duration**: 8–12 hours
- **Practice**:
  - Debug a pod memory leak using Prometheus/Grafana.
- **Task**: Add troubleshooting log to README.

Tools: Prometheus, Grafana.

7 Week 4: Advanced Observability (Loki, Tempo, Jaeger/OpenTelemetry)

Goal: Master logging, tracing, and secrets management.

7.1 Day 22: Loki

- **Duration**: 6 hours
- **Learn**: Log aggregation with Loki.
  - Resources: Grafana Loki Docs.
- **Practice**:
  - Install Loki, configure a Flask app to send logs, query in Grafana.
- **Task**: Document Loki setup in README.

7.2 Day 23: Tempo and Jaeger

- **Duration**: 6 hours
- **Learn**: Distributed tracing with Tempo and Jaeger.
  - Resources: Tempo Docs, Jaeger Docs.
- **Practice**:

  - Instrument a Python app with OpenTelemetry.
  - Send traces to Jaeger/Tempo, visualize in Grafana.
- **Task**: Add tracing setup to README.

7.3 Day 24: OpenTelemetry and Secrets Management

- **Duration**: 6 hours
- **Learn**: OpenTelemetry for metrics/logs/traces, Kubernetes Secrets, HashiCorp Vault.
  - Resources: OpenTelemetry Docs, Vault Docs.
- **Practice**:
  - Add OpenTelemetry auto-instrumentation.
  - Store a secret in Kubernetes Secrets.
- **Task**: Document OpenTelemetry and Secrets setup in README.

7.4 Day 25: Mini-Project

- **Duration**: 6 hours
- **Practice**:
  - Deploy a multi-service app with Loki, Tempo/Jaeger, Prometheus, and Secrets.
  - Create a unified Grafana dashboard.
- **Task**: Document the project in README.

7.5 Day 26–27: Troubleshooting and Review

- **Duration**: 8–12 hours
- **Practice**:
  - Simulate issues (missing logs, broken traces), debug using observability tools.
- **Task**: Add troubleshooting notes to README.

Tools: Loki, Tempo, Jaeger, OpenTelemetry, Vault.

8 Week 5: Service Mesh (Istio)

Goal: Learn Istio for microservices and security.

8.1 Day 29–30: Istio Basics

- **Duration**: 12 hours
- **Learn**: Istio architecture, virtual services, destination rules, gateways, mTLS.
  - Resources: Istio Docs, "Istio Up and Running" (summary).
- **Practice**:

  - Install Istio on Minikube, deploy Bookinfo app.
  - Configure routing and mTLS.
- **Task**: Document Istio setup in README.

8.2 Day 31: Advanced Istio

- **Duration**: 6 hours
- **Learn**: Traffic mirroring, A/B testing, circuit breakers.
- **Practice**:
  - Configure mirroring and circuit breakers for Bookinfo.
- **Task**: Add configs to README.

8.3 Day 32: Troubleshooting Istio and CNCF Tools

- **Duration**: 6 hours
- **Learn**: Debug Istio issues, compare Istio vs. Linkerd.
  - Resources: Linkerd Docs.
- **Practice**:
  - Simulate a misconfigured virtual service, debug with `istioctl analyze`.
  - Compare Istio/Linkerd features.
- **Task**: Add troubleshooting and comparison to README.

8.4 Day 33: Mini-Project

- **Duration**: 6 hours
- **Practice**:
  - Deploy a microservices app with Istio, routing, circuit breakers, mTLS, and Prometheus/Grafana monitoring.
- **Task**: Document the project in README.

8.5 Day 34–35: Review and Catch-Up

- **Duration**: 8–12 hours
- **Practice**:
  - Debug a complex Istio issue (e.g., traffic routing failure).
- **Task**: Add troubleshooting notes to README.

Tools: Istio, istioctl, Linkerd.

9 Week 6: CI/CD and SRE Principles

Goal: Master CI/CD pipelines and SRE concepts.

9.1 Day 36–37: Jenkins and CI/CD

- **Duration**: 12 hours
- **Learn**: Jenkins pipelines, Groovy syntax, pipeline optimization.
  - Resources: Jenkins Docs, CloudBees tutorials.
- **Practice**:
  - Set up Jenkins on Kubernetes.
  - Create a pipeline with parallel stages to build/test/deploy a Python app.
- **Task**: Document Jenkins pipeline in README.

9.2 Day 38: Argo Workflows

- **Duration**: 6 hours
- **Learn**: Argo Workflows for task orchestration.
  - Resources: Argo Workflows Docs.
- **Practice**:
  - Create a workflow to automate build/test/deploy, integrate with ArgoCD.
- **Task**: Add workflow to README.

9.3 Day 39: SRE Concepts and Incident Management

- **Duration**: 6 hours
- **Learn**: SLOs, SLIs, error budgets, PagerDuty workflows.
  - Resources: Google SRE Book, PagerDuty Docs.
- **Practice**:
  - Define SLOs/SLIs for a sample app.
  - Calculate an error budget, simulate an on-call response.
- **Task**: Document SLOs and incident response in README.

9.4 Day 40: Mini-Project

- **Duration**: 6 hours
- **Practice**:
  - Build a CI/CD pipeline with Jenkins/ArgoCD, deploy a microservices app, monitor SLOs.

- **Task**: Document the project in README.

9.5 Day 41–42: Review and Troubleshooting

- **Duration**: 8–12 hours
- **Practice**:
  - Simulate a pipeline failure (e.g., failed build), debug.
- **Task**: Add troubleshooting notes to README.

Tools: Jenkins, Argo Workflows, ArgoCD, PagerDuty.

10 Week 7: Cloud (AWS/EKS, GCP/GKE) and Terraform

Goal: Learn cloud-native Kubernetes and IaC.

10.1 Day 43–44: AWS, EKS, and Cost Optimization

- **Duration**: 12 hours
- **Learn**: AWS basics (EC2, S3, IAM), EKS, AWS Cost Explorer.
  - Resources: AWS EKS Docs, AWS Cloud Practitioner.
- **Practice**:
  - Create an EKS cluster with `eksctl`.
  - Deploy a sample app, analyze costs with Cost Explorer.
- **Task**: Document EKS setup and cost analysis in README.

10.2 Day 45: GCP, GKE, and Advanced Cloud Services

- **Duration**: 6 hours
- **Learn**: GCP basics, GKE, Cloud SQL.
  - Resources: GCP GKE Docs, Google Cloud free tier.
- **Practice**:
  - Set up a GKE cluster, deploy an app, experiment with Cloud SQL.
- **Task**: Add GKE and Cloud SQL setup to README.

10.3 Day 46: Terraform

- **Duration**: 6 hours
- **Learn**: Infrastructure as code with Terraform.
  - Resources: Terraform Docs.
- **Practice**:
  - Write Terraform scripts to provision an EKS cluster, deploy an app.

- **Task**: Add Terraform scripts to Git repo, document in README.

10.4 Day 47: Mini-Project

- **Duration**: 6 hours
- **Practice**:
  - Provision an EKS cluster with Terraform.
  - Deploy a microservices app with ArgoCD, use Cloud SQL.
- **Task**: Document the project in README.

10.5 Day 48–49: Review and Multicloud

- **Duration**: 8–12 hours
- **Practice**:
  - Experiment with a multicloud setup (EKS + GKE), debug issues.
- **Task**: Add multicloud notes to README.

Tools: AWS CLI, eksctl, GCP SDK, Terraform, Cloud SQL.

11 Week 8: Databases, Chaos Engineering, and Interview Prep

Goal: Learn stateful apps, chaos engineering, and interview skills.

11.1 Day 50: Databases on Kubernetes

- **Duration**: 6 hours
- **Learn**: MongoDB/Postgres with Operators, backups.
  - Resources: Bitnami Helm Charts, Velero Docs.
- **Practice**:
  - Deploy Postgres with an Operator, configure backups with Velero.
- **Task**: Document Postgres setup and backups in README.

11.2 Day 51: Chaos Engineering

- **Duration**: 6 hours
- **Learn**: Chaos engineering, LitmusChaos.
  - Resources: LitmusChaos Docs.
- **Practice**:
  - Install LitmusChaos, simulate pod failures, analyze with Prometheus/Grafana.
- **Task**: Add chaos experiment results to README.

11.3 Day 52: System Design and Team Dynamics

- **Duration**: 6 hours
- **Learn**: System design for DevOps, DevOps culture.
  - Resources: "System Design Interview" (Alex Xu), "The Phoenix Project" (summary).
- **Practice**:
  - Design a scalable Kubernetes system.
  - Practice explaining SLOs to a team.
- **Task**: Document system design in README.

11.4 Day 53: Capstone Project

- **Duration**: 6 hours
- **Practice**:
  - Deploy a Flask + Postgres app to EKS/GKE with Terraform, ArgoCD, Istio, observability, and LitmusChaos.
  - Optimize costs.
- **Task**: Document the project in README.

11.5 Day 54–55: Interview Preparation

- **Duration**: 12 hours
- **Learn**: DevOps/SRE interview questions, system design, behavioral questions.
  - Resources: KodeKloud Labs, Pramp, "Cracking the Coding Interview."
- **Practice**:
  - Solve Kubernetes troubleshooting scenarios.
  - Practice Python/Bash coding on HackerRank.
  - Conduct mock interviews (technical + behavioral).
  - Design a cost-optimized, secure Kubernetes cluster.
- **Task**: Add interview notes and diagrams to README.

11.6 Day 56: Final Review and Runbook

- **Duration**: 6 hours
- **Practice**:
  - Write a runbook for the capstone project, explain it to a developer.
  - Review weak areas (e.g., Istio, SLOs).

- **Task**: Finalize README with runbook and summary.

Tools: Postgres/MongoDB Operators, Velero, LitmusChaos.

12 README Document Template

Below is a Markdown template for a GitHub repository to document your learning and showcase your work.

```
# DevOps/SRE Learning Portfolio

This repository documents my 8-week journey to master DevOps/SRE skills for a role in

## Table of Contents
1. [Week 1: Kubernetes, Docker, Networking, Security](#week-1)
2. [Week 2: GitOps, Kustomize, Helm, Developer Experience](#week-2)
3. [Week 3: Observability (Prometheus, Grafana)](#week-3)
4. [Week 4: Advanced Observability](#week-4)
5. [Week 5: Service Mesh (Istio)](#week-5)
6. [Week 6: CI/CD and SRE Principles](#week-6)
7. [Week 7: Cloud and Terraform](#week-7)
8. [Week 8: Databases, Chaos Engineering, Interview Prep](#week-8)
9. [Capstone Project](#capstone-project)
10. [Runbook](#runbook)
11. [Interview Notes](#interview-notes)

## Week 1: Kubernetes, Docker, Networking, Security
- **Objective**: Master Kubernetes/Docker, networking, security, scripting.
- **Tasks**:
  - Deployed Nginx to Minikube.
  - Built a Flask Docker image, scanned with Trivy.
  - Configured Network Policy, RBAC, resource limits, HPA.
  - Wrote automation scripts.
- **Artifacts**:
  - [Dockerfile](./week1/Dockerfile)
  - [Network Policy](./week1/network-policy.yaml)
  - [Python Script](./week1/k8s_query.py)
  - [Troubleshooting Log](./week1/troubleshooting.md)
- **Mini-Project**: Flask app with security and resource optimization.

## Week 2: GitOps, Kustomize, Helm, Developer Experience
- **Objective**: Master GitOps and developer collaboration.
- **Tasks**:
  - Installed ArgoCD, deployed via GitOps.
  - Created Kustomize manifests and Helm charts.
  - Wrote a developer guide, compared Fluentd vs. Loki.
- **Artifacts**:
  - [Kustomize Manifests](./week2/kustomize/)
  - [Helm Chart](./week2/helm/)
  - [Developer Guide](./week2/developer-guide.md)
- **Mini-Project**: Flask + Redis app with GitOps.

## Week 3: Observability (Prometheus, Grafana)
- **Objective**: Learn monitoring and metrics.
- **Tasks**:
  - Installed Prometheus/Grafana, configured metrics and alerts.
  - Set up HPA, debugged high-load issues.
- **Artifacts**:
  - [Prometheus Config](./week3/prometheus.yml)
  - [Grafana Dashboard JSON](./week3/dashboard.json)
- **Mini-Project**: Microservices app with monitoring.

## Week 4: Advanced Observability
- **Objective**: Master logging, tracing, secrets.
- **Tasks**:
  - Configured Loki, Tempo/Jaeger, OpenTelemetry.
  - Managed secrets with Kubernetes Secrets.
- **Artifacts**:
  - [OpenTelemetry Config](./week4/otel-config.yaml)
  - [Secrets YAML](./week4/secrets.yaml)
- **Mini-Project**: Multi-service app with unified observability.

## Week 5: Service Mesh (Istio)
- **Objective**: Learn Istio and microservices.
- **Tasks**:
  - Deployed Bookinfo with Istio, configured routing, mTLS, circuit breakers.
  - Compared Istio vs. Linkerd.
- **Artifacts**:
  - [Virtual Service](./week5/virtual-service.yaml)
  - [Comparison Table](./week5/istio-vs-linkerd.md)
- **Mini-Project**: Microservices app with Istio.

## Week 6: CI/CD and SRE Principles
- **Objective**: Master CI/CD and SRE concepts.
- **Tasks**:
  - Built Jenkins pipeline with parallel stages.
  - Created Argo Workflows.
  - Defined SLOs/SLIs, simulated on-call response.
- **Artifacts**:
  - [Jenkinsfile](./week6/Jenkinsfile)
  - [SLO Definition](./week6/slo.md)
- **Mini-Project**: CI/CD pipeline with SLO monitoring.

## Week 7: Cloud and Terraform
- **Objective**: Learn cloud-native Kubernetes and IaC.
- **Tasks**:
  - Created EKS/GKE clusters, analyzed costs.
  - Used Terraform for EKS provisioning.
  - Deployed to Cloud SQL.
- **Artifacts**:
  - [Terraform Scripts](./week7/terraform/)
  - [Cost Analysis](./week7/cost-analysis.md)
- **Mini-Project**: App on EKS with Terraform and Cloud SQL.

## Week 8: Databases, Chaos Engineering, Interview Prep
- **Objective**: Learn stateful apps, chaos engineering, interview skills.
- **Tasks**:
  - Deployed Postgres with Operator, backups with Velero.
  - Ran chaos experiments with LitmusChaos.
  - Designed a Kubernetes system, practiced interviews.
- **Artifacts**:
  - [Postgres YAML](./week8/postgres.yaml)
  - [Chaos Log](./week8/chaos.md)
  - [System Design Diagram](./week8/design.png)
- **Capstone Project**: Flask + Postgres app on EKS/GKE with full observability and r

## Runbook
- **Purpose**: Guide for capstone project deployment.
- **Steps**:
  1. Provision EKS with Terraform.
  2. Deploy with ArgoCD and Istio.
  3. Configure observability (Prometheus/Loki/Tempo).
  4. Set up SLOs and alerts.
  5. Test with LitmusChaos.
- **Troubleshooting**: Pod crashes, routing issues.
- [Full Runbook](./runbook.md)

## Interview Notes
- **Technical**: Kubernetes debugging, Istio, SLOs.
- **System Design**: Scalable Kubernetes architecture.
- **Behavioral**: Incident response, collaboration stories.
- [Full Notes](./interview-notes.md)

## About
- DevOps/SRE learning journey, July 2025.
- Contact: Your Email/LinkedIn
```

13 Tips for Success

- **GitHub Repo**: Host the README in a public repo, include scripts, manifests, screenshots.
- **Troubleshooting**: Practice debugging daily (`kubectl`, `istioctl`).
- **Interviews**: Prepare STAR-method stories, demo the capstone project.

- **Community**: Join CNCF Slack, X DevOps/SRE groups (@kelseyhightower).
- **Time Management**: Stick to 4–6 hours/day, use catch-up days.

14 Resources

- **Books**: Google SRE Book, “Kubernetes Patterns,” “System Design Interview.”
- **Courses**: FreeCodeCamp, ACloudGuru, CNCF CKA prep.
- **Labs**: KodeKloud, Killercoda, Qwiklabs.