DevOps Engineer Roadmap 2026

Linux, cloud, CI/CD, Docker, Kubernetes, Terraform, observability, security and platform engineering roadmap

Goal: Build and operate reliable software delivery platforms: automate infrastructure, deploy safely, monitor systems, secure the supply chain and improve developer productivity.

Prepared: 31 May 2026. **Use-case:** self-study, portfolio building, interview prep and job-readiness.

How to use this roadmap

- Follow the phases in order, but do not wait to build projects. Every topic should end with a public artifact: code, dashboard, demo, README, diagram or case study.
- Use YouTube for intuition and demos; use official documentation for correctness; use practice platforms for repetition; use projects for proof.
- Track progress with a weekly scorecard: hours learned, problems solved, project commits, notes written, demos recorded and applications/interviews completed.
- AI tools are allowed, but every project must show that you understand, test and can debug the output.

2026 market calibration

- DevOps in 2026 is converging with platform engineering, DevSecOps, SRE, cloud cost management and AI-assisted delivery.
- Kubernetes and cloud-native practices remain core for many production environments; reliability, observability, security and developer experience are major hiring signals.
- AI tools can assist operations, but strong fundamentals in Linux, networking, CI/CD, IaC, debugging and incident response remain non-negotiable.

Role title mappings

- DevOps Engineer, Cloud Engineer, Platform Engineer, Site Reliability Engineer, Infrastructure Engineer, DevSecOps Engineer, Build and Release Engineer

2026 hiring-ready stack

- Linux, Bash/Python, networking, Git, CI/CD, Docker, Kubernetes, Helm, Terraform, AWS/Azure/GCP, Prometheus/Grafana/OpenTelemetry, GitOps, security scanning, incident response, FinOps

End-to-end roadmap overview

Phase	Focus	What to learn	Job-ready proof
0-4 weeks	Linux, networking and Git	Shell, processes, services, permissions, TCP/IP, DNS, HTTP, SSH, Git workflows.	Run and troubleshoot services on Linux.
1-2 months	CI/CD and containers	GitHub Actions/GitLab CI, Dockerfiles, Compose, registries, artifact management, release strategies.	Build pipeline that tests, builds and deploys a containerized app.
2-4 months	Cloud and infrastructure as code	AWS/Azure/GCP basics, IAM, VPC, compute, storage, databases, Terraform modules, secrets.	Provision cloud-like infra with Terraform and document it.
4-6 months	Kubernetes and GitOps	Pods, deployments, services, ingress, config, secrets, Helm, autoscaling, Argo CD, policies.	Deploy a microservice stack to Kubernetes with GitOps.
6-12 months	Reliability, security and platforms	Observability, SLOs, incident response, supply chain security, platform portals, FinOps, AI ops.	Ship a production platform portfolio with runbooks and dashboards.

Weekly operating system

- 10-12 hours learning/building: 60% project implementation, 20% docs, 10% YouTube/course intuition, 10% notes and revision.
- Minimum weekly output: 5 commits, 1 written note, 1 demo screenshot/video, 5-20 practice problems depending on role.
- Every Sunday: review blockers, update README, write what you learned, plan next week, and compare against target job descriptions.
- Every month: ship one small project or one major milestone of a bigger capstone.

Complete topic and subtopic checklist

1. Linux and scripting

- Filesystem, permissions, processes, systemd, logs, package managers, shell scripting, Python automation, cron, troubleshooting.
- Proof to create: Debug a failing service from logs to fix.

2. Networking fundamentals

- TCP/IP, DNS, HTTP/TLS, load balancing, proxies, firewalls, CIDR, NAT, service discovery, latency troubleshooting.
- Proof to create: Explain the path of a request through your system.

3. CI/CD

- Pipelines, test stages, artifacts, container builds, deployment strategies, blue/green, canary, feature flags, rollback.
- Proof to create: Automate test-build-deploy with rollback.

4. Containers and Kubernetes

- Docker images, Compose, Pods, Deployments, Services, Ingress, ConfigMaps, Secrets, HPA, volumes, probes, resource limits.
- Proof to create: Run a resilient app on Kubernetes.

5. Infrastructure as code and cloud

- Terraform state, modules, IAM, VPC, compute, object storage, managed databases, secrets, environments, drift.
- Proof to create: Provision reusable environments safely.

6. Observability and SRE

- Metrics, logs, traces, Prometheus, Grafana, OpenTelemetry, SLOs, error budgets, alert fatigue, on-call, postmortems.
- Proof to create: Create dashboards and alerts tied to user impact.

7. Security and supply chain

- OWASP, container scanning, SBOMs, secret scanning, IAM least privilege, policy as code, vulnerability management, incident response.
- Proof to create: Add security gates to CI/CD.

8. Platform engineering

- Internal developer platforms, golden paths, service catalogs, Backstage, templates, developer experience, self-service, governance.
- Proof to create: Build a simple developer portal or paved-road template.

9. FinOps and cost

- Tagging, budgets, rightsizing, autoscaling, storage lifecycle, cost dashboards, waste cleanup, GPU cost awareness.
- Proof to create: Add cost controls to your cloud architecture.

Portfolio projects and capstones

Do not treat these as toy tutorials. Each project should have a README, architecture diagram, setup steps, screenshots, demo link when possible, tests/checks, limitations and future improvements.

Project 1: CI/CD production pipeline

- Outcome: Test, build, scan, push and deploy a containerized app.
- Suggested stack: GitHub Actions, Docker, registry, cloud/local server
- Stretch goal: Add canary/rollback and security scanning.

Project 2: Terraform cloud platform

- Outcome: Provision VPC, compute, database, object storage and IAM.
- Suggested stack: Terraform, AWS/Azure/GCP
- Stretch goal: Use modules, remote state and environment separation.

Project 3: Kubernetes microservices platform

- Outcome: Deploy services with ingress, HPA, secrets and Helm.
- Suggested stack: Kubernetes, Helm, Argo CD
- Stretch goal: Add network policy and resource limits.

Project 4: Observability stack

- Outcome: Metrics, logs, traces and SLO dashboard for a real app.
- Suggested stack: Prometheus, Grafana, OpenTelemetry
- Stretch goal: Run a simulated incident and postmortem.

Project 5: Internal developer platform

- Outcome: Self-service template to create/deploy a service.
- Suggested stack: Backstage or GitHub templates, CI/CD, Kubernetes
- Stretch goal: Measure developer experience improvements.

Interview preparation checklist

- Linux troubleshooting and shell scripting.
- Design CI/CD for a service with rollback and security gates.
- Explain Kubernetes objects and debug a failing deployment.
- Design cloud infrastructure with IAM, networking and cost controls.
- Discuss observability, SLOs, incidents and postmortems.
- Explain Terraform state, drift and module design.

Portfolio checklist before applying

- One CI/CD pipeline with tests, scans and deployment.
- One Terraform cloud project.
- One Kubernetes/GitOps project.
- One observability incident simulation with dashboard and postmortem.
- Diagrams, runbooks and cost notes.

Resume keywords to include when true

- Linux, Bash/Python, networking, Git, CI/CD, Docker, Kubernetes, Helm, Terraform, AWS/Azure/GCP, Prometheus/Grafana/OpenTelemetry, GitOps, security scanning, incident response, FinOps, DevOps Engineer, Cloud Engineer, Platform Engineer

Best official docs and learning references

Prioritize these over random snippets. Read docs when implementation breaks, when preparing interviews, and when upgrading tools.

- **Docker Docs** - Containers and images. (open)
- **Kubernetes Docs** - Container orchestration. (open)
- **Helm Docs** - Kubernetes packaging. (open)
- **Terraform Docs** - Infrastructure as code. (open)
- **Ansible Docs** - Configuration automation. (open)
- **AWS Docs** - AWS services. (open)
- **Google Cloud Docs** - Google Cloud services. (open)
- **Azure Docs** - Azure services. (open)
- **GitHub Actions Docs** - CI/CD automation. (open)
- **Prometheus Docs** - Metrics and monitoring. (open)
- **Grafana Docs** - Dashboards and observability. (open)
- **OpenTelemetry Docs** - Traces, metrics and logs standard. (open)
- **Argo CD Docs** - GitOps continuous delivery. (open)
- **Backstage Docs** - Internal developer platform portal. (open)
- **SRE Book** - Google SRE principles. (open)
- **OWASP Top 10** - Web security risks. (open)
- **Node.js Learn** - Node runtime and backend basics. (open)
- **Express Docs** - Minimal Node web framework. (open)
- **Git documentation** - Version control fundamentals and workflows. (open)
- **GitHub Docs** - GitHub repos, Issues, PRs, Actions and security features. (open)
- **Linux Journey** - Linux basics. (open)
- **The Linux Documentation Project** - Linux guides. (open)
- **CNCF Landscape** - Cloud-native ecosystem. (open)
- **Kubernetes Patterns** - Kubernetes application patterns. (open)
- **OpenSSF Scorecard** - Supply chain security checks. (open)
- **Trivy Docs** - Container and dependency scanning. (open)

Best YouTube sources and how to use them

Use these for intuition, project walkthroughs and high-level context. Always verify implementation details against official documentation because tools change quickly.

- **freeCodeCamp.org** - Long-form free courses across web, data, cloud and AI. (open)
- **ByteByteGo** - System design, architecture and backend concepts. (open)
- **TechWorld with Nana** - DevOps, Docker, Kubernetes and CI/CD. (open)
- **KodeKloud** - Linux, Kubernetes, Terraform and certifications. (open)
- **Bret Fisher** - Docker and DevOps production practice. (open)
- **CNCF** - Cloud-native talks and projects. (open)
- **Google Cloud Tech** - GCP, Kubernetes and cloud architecture. (open)
- **AWS Developers** - AWS developer and cloud content. (open)
- **Microsoft Developer** - Azure, .NET, AI and cloud development. (open)
- **DevOps Toolkit** - Kubernetes, GitOps and platform engineering. (open)
- **Fireship** - Fast overviews of web tools and trends. (open)
- **Web Dev Simplified** - Practical JavaScript, React and web explanations. (open)
- **DataTalksClub** - Data engineering and MLOps zoomcamps. (open)
- **Seattle Data Guy** - Data engineering career and architecture. (open)

Recommended YouTube search strategy

- **YouTube search: DevOps Engineer roadmap 2026 project based** - Use filters for upload date and length; prefer recent, project-based content. (open)
- **YouTube search: DevOps Engineer interview preparation 2026** - Use filters for upload date and length; prefer recent, project-based content. (open)
- **YouTube search: DevOps Engineer portfolio project end to end** - Use filters for upload date and length; prefer recent, project-based content. (open)
- **YouTube search: DevOps Engineer system design real world project** - Use filters for upload date and length; prefer recent, project-based content. (open)

Best coding and practice platforms

Practice platforms build repetition. They do not replace projects. For each platform, maintain a solved-problems log with mistakes and revised patterns.

- **KodeKloud Free Labs** - Linux, Docker, Kubernetes and DevOps labs. (open)
- **Killercoda** - Kubernetes, Linux and cloud-native labs. (open)
- **Cloud Resume Challenge** - Cloud portfolio challenge. (open)
- **AWS Workshops** - AWS practice. (open)
- **Google Cloud Skills Boost** - GCP labs. (open)
- **Microsoft Learn** - Azure labs. (open)
- **OverTheWire** - Linux and security fundamentals. (open)
- **Terraform Tutorials** - IaC practice. (open)
- **LeetCode** - DSA patterns for interviews. (open)
- **NeetCode Roadmap** - Structured DSA practice. (open)
- **HackerRank** - Algorithms, SQL and language tracks. (open)
- **Exercism** - Language practice with mentoring. (open)
- **ByteByteGo** - System design explanations. (open)
- **System Design Primer** - Open-source system design study. (open)
- **Design Gurus Grokking** - System-design practice. (open)
- **Educative System Design** - Interactive system design course. (open)

30-60-90 day execution plan

Period	Primary objective	Deliverables
Days 1-30	Foundation and first small project	Finish basics, solve starter practice, publish one small project and notes.
Days 31-60	Core role skills and second project	Complete core docs, build a stronger project, add tests/checks and write a case study.
Days 61-90	Production proof and interview prep	Deploy capstone, create architecture diagram, record demo, start mock interviews and applications.

Common mistakes to avoid

- Only watching videos without building public proof.
- Building tutorial clones without explaining decisions, tradeoffs and failure cases.
- Ignoring documentation, testing, security, accessibility, observability or data quality.
- Using AI-generated code without understanding it or adding tests.
- Applying to jobs before your portfolio proves the core responsibilities of the role.

Market research sources consulted

- **World Economic Forum Future of Jobs 2025** - Macro skill demand for 2025-2030; big data, AI/ML, software, cybersecurity and tech literacy signals. (open)
- **Stack Overflow Developer Survey 2025** - Developer technology, AI-tool use, documentation, languages and web framework signals. (open)
- **GitHub Octoverse 2025** - Open-source and language momentum; TypeScript, Python, agents and typed-language shift. (open)
- **DORA State of AI-assisted Software Development 2025** - AI in software delivery; verification, platform maturity and organizational system effects. (open)
- **CNCF Annual Cloud Native Survey** - Kubernetes, cloud-native maturity and AI infrastructure adoption. (open)
- **dbt State of Analytics Engineering 2025** - Analytics engineering trends: AI use, data trust, quality and investment. (open)
- **LangChain State of Agent Engineering** - Agent adoption, production challenges, evaluation and observability trends. (open)
- **McKinsey State of AI 2025** - Enterprise AI adoption, scaling practices, governance and value capture. (open)
- **DORA State of AI-assisted Software Development 2025** - AI-assisted software delivery research. (open)
- **CNCF Annual Cloud Native Survey** - Kubernetes/cloud native adoption. (open)
- **State of Platform Engineering Report Vol 4** - Platform engineering trends. (open)