THE COMPLETE DEVOPS ENGINEER ROADMAP

Go From Zero to a DevOps Engineer in 12 Months

Mosh Hamedani

Hi! I am Mosh Hamedani, a software engineer with over 20 years of experience.

Over the past 10 years, I've had the privilege of teaching millions of people how to code and become professional software engineers through my YouTube channel and online courses.

It's my mission to make software engineering accessible to everyone. Join me on this journey and unlock your potential in the world of coding!

https://codewithmosh.com

Table of Content

Introduction

This guide is designed to help you navigate the essential skills needed to become a successful DevOps engineer. Whether you're just starting out or looking to enhance your existing skills, this roadmap will provide a clear and structured path.

Target Audience

This guide is for:

- **Beginners** who want to know what they need to learn to land a DevOps job.
- **Experienced individuals** looking to level up their skills and fill in the gaps in their knowledge.

Resources

For detailed tutorials and full courses, check out the following resources:

- **YouTube Channel**: https://www.youtube.com/c/programmingwithmosh
- **Full Courses**: https://codewithmosh.com

Roadmap Overview

Below is a comprehensive table listing all the essential skills needed to become a proficient DevOps engineer, along with the estimated time required to learn each skill.

Keep in mind that the time needed to learn each skill can vary for everyone. These estimates are based on dedicating 3 to 5 hours of study every day.

Use this roadmap to guide your learning journey and track your progress as you build a strong foundation in data analysis.

Skill	Est. Time	Learning Phase
Linux Fundamentals	2-3 weeks	Beginner
Networking Concepts	2 weeks	Beginner
Version Control (Git)	1-2 weeks	Beginner
Programming Languages (Python)	4-6 weeks	Beginner
Cloud Providers (AWS)	4-6 weeks	Intermediate
Containerization (Docker)	3-4 weeks	Intermediate
CI/CD (Jenkins)	3-4 weeks	Intermediate
Container Orchestration (Kubernetes)	4-6 weeks	Intermediate
Networking Services (Nginx)	3-4 weeks	Intermediate
Configuration Management (Ansible)	3-4 weeks	Advanced
Infrastructure as Code (Terraform)	3-4 weeks	Advanced
Monitoring and Logging (Prometheus)	3-4 weeks	Advanced
Total	**10-14 months**	

Linux Fundamentals

Linux is an open-source operating system that's super popular for servers and development. As a DevOps engineer, you'll be setting up and maintaining the infrastructure where applications run. Most servers use Linux, so getting comfortable with it, especially the command line, is a must. Start with learning Bash, which is the most commonly used shell and scripting language in Linux.

Estimated Time: 2-3 weeks

Essential Concepts

- Basic Linux commands (ls, cp, mv, rm, etc.)
- File system hierarchy (/, /home, /etc, /var, etc.)
- Permissions and ownership (chmod, chown)
- Processes and signals (ps, top, kill)
- Package management (apt, yum)
- Shell scripting basics (variables, loops, conditionals, functions)

Networking Concepts

Networking is all about how computers talk to each other, covering things like IP addresses and network protocols. You need to understand how data moves around, how to secure it, and how to troubleshoot network issues. This foundation is key to managing your infrastructure effectively. A great tool to get hands-on experience is Wireshark, which lets you analyze network traffic.

Estimated Time: 2 weeks

Essential Concepts

- OSI and TCP/IP models
- IP addressing and subnetting (IPv4, IPv6)
- DNS and DHCP
- Network protocols (HTTP, HTTPS, FTP, SSH, etc.)
- Firewalls and security groups (iptables, UFW)
- Basic network troubleshooting (ping, traceroute, netstat)

Version Control (Git)

Git is a version control system that lets you keep track of changes in your code and collaborate with others. It's essential for working on projects with a team, managing your code, tracking changes, and collaborating without overwriting each other's work.

Estimated Time: 1-2 weeks

Learning resources: YouTube Tutorial | Full Course

Essential Concepts

- Git basics and commands (clone, commit, push, pull)
- Branching and merging
- Working with remote repositories (GitHub, GitLab)
- Pull requests and code reviews
- Git workflows (Git Flow, GitHub Flow)
- Resolving merge conflicts

Programming Languages

Programming languages like Python, Ruby, and Go are used to automate tasks and manage configurations. You'll need to write scripts to automate repetitive tasks, handle configurations, and integrate different tools. While there are several popular languages, I recommend Python for its simplicity, powerful libraries, and versatility.

Estimated Time: 4-6 weeks

Learning resources: YouTube Tutorial | Full Course

Essential Concepts

- Python syntax and data structures (lists, dictionaries, sets, tuples)
- Modules and packages (importing and using libraries)
- Writing and executing Python scripts
- File handling (reading from and writing to files)
- Error handling (try, except)
- Automation scripts (using libraries like subprocess, os, sys)
- Basic web scraping (using libraries like BeautifulSoup, requests)

Cloud Providers

Cloud providers like AWS, Azure, and GCP offer a range of services like computing, storage, and networking that you can use to build and deploy applications. If you're just starting out, I recommend picking one cloud provider to focus on, and AWS is a great choice because it's the most widely used. Once you get comfortable with one, you can learn the others as needed.

Estimated Time: 3-4 weeks

Essential Concepts

- **AWS (Recommended)**
  - EC2 (virtual servers)
  - S3 (object storage)
  - IAM (Identity and Access Management)
  - VPC (Virtual Private Cloud)
  - RDS (Relational Database Service)
  - Lambda (serverless compute)
  - CloudWatch (monitoring and logging)
- **Azure**
  - VMs (virtual machines)
  - Storage accounts
  - Networking (VNets, NSGs)
  - Azure DevOps (CI/CD pipelines)
- **GCP**
  - Compute Engine (virtual machines)

- Cloud Storage
- Networking (VPCs, Firewalls)
- Stackdriver (monitoring and logging)

Containerization

Containerization is about packaging an application and its dependencies into a container, ensuring it runs the same everywhere. Containers are the new standard for packaging software. They make sure your application runs consistently across different environments, which simplifies deployment and management. Docker is the go-to tool for containerization.

Estimated Time: 3-4 weeks

Learning resources: YouTube Tutorial | Full Course

Essential Concepts

- Docker installation and setup
- Creating and managing Docker images
- Running containers
- Dockerfile basics (writing Dockerfiles)
- Docker Compose (defining and running multi-container applications)
- Container networking
- Data persistence (volumes and bind mounts)

Continuous Integration/Continuous Deployment (CI/CD)

CI/CD is a method to automatically integrate and deploy code changes, allowing for frequent and reliable releases. CI/CD pipelines make sure that your code changes are automatically tested and deployed. This speeds up the development process and maintains high software quality. Jenkins is a powerful tool for setting up CI/CD pipelines, but other popular tools include GitLab CI/CD, CircleCI, and Travis CI. I recommend starting with Jenkins for its versatility and strong community support.

Estimated Time: 3-4 weeks

Essential Concepts

- CI/CD concepts and principles
- Jenkins installation and setup
- Creating and configuring Jenkins pipelines
- Jenkinsfile basics (declarative and scripted syntax)
- Automated testing (unit tests, integration tests)
- Build automation
- Deployment automation
- Monitoring and reporting

Container Orchestration

Orchestration tools like Kubernetes and Helm help automate the deployment, scaling, and management of containerized applications. These tools are essential for managing complex applications in production, ensuring they run smoothly, scale easily, and recover quickly from failures. Kubernetes is the leading orchestration tool you should learn.

Estimated Time: 4-6 weeks

Essential Concepts

- Kubernetes architecture (master and worker nodes)
- Kubernetes components (pods, services, deployments)
- Managing resources in Kubernetes (namespaces, ConfigMaps, Secrets)
- Scaling applications (Horizontal Pod Autoscaler)
- Helm basics (charts, repositories, releases)
- Kubernetes networking (services, ingress)
- Security in Kubernetes (RBAC, network policies)

Networking and Infrastructure Services

This involves setting up and managing services like reverse proxies, forward proxies, caching servers, firewalls, and load balancers. Understanding these services is crucial for optimizing application performance, managing network traffic, and ensuring security. They play a vital role in the architecture and operational stability of your applications. Nginx is a versatile tool for handling reverse proxies and load balancing.

Estimated Time: 3-4 weeks

Essential Concepts

- Setting up and configuring Nginx as a reverse proxy
- Setting up forward proxies
- Implementing caching with Redis or Varnish
- Configuring firewalls and security groups (iptables, UFW)
- Load balancing with HAProxy or AWS ELB
- SSL/TLS configuration for secure communications
- Troubleshooting network issues

Configuration Management

Configuration management tools like Ansible, Puppet, and Chef automate the deployment, configuration, and management of servers and applications. These tools are critical for maintaining consistency across your infrastructure. They enable you to automate repetitive tasks, ensuring that all servers and applications are configured correctly and consistently. I recommend starting with Ansible due to its simplicity and powerful features.

Estimated Time: 3-4 weeks

Essential Concepts

- **Ansible (Recommended)**
  - Writing Ansible playbooks
  - Roles and modules
  - Managing variables and templates
- **Puppet**
  - Manifests
  - Modules
  - Classes
- **Chef**
  - Cookbooks
  - Recipes
  - Resources

Infrastructure as Code (IaC)

IaC involves managing and provisioning computing infrastructure through machine-readable configuration files, rather than through physical hardware configuration or interactive configuration tools. IaC tools like Terraform and CloudFormation enable you to automate the setup and management of your infrastructure. This approach increases efficiency, reduces errors, and ensures that infrastructure is scalable and repeatable. I highly recommend starting with Terraform for its flexibility and widespread use.

Estimated Time: 3-4 weeks

Essential Concepts

- **Terraform (Recommended)**
  - Basics (installation, providers, resources)
  - Writing Terraform configuration files
  - Managing Terraform state
  - Using Terraform modules
  - Advanced concepts (workspaces, remote state)
- **AWS CloudFormation**
  - Basics (stacks, templates)
  - Writing CloudFormation templates
  - Managing infrastructure lifecycle with IaC

Monitoring and Logging

Monitoring and logging tools, such as Prometheus, Grafana, ELK Stack, and Fluentd, track the performance and health of your applications and infrastructure. Effective monitoring and logging are essential for maintaining the reliability and performance of your applications. These tools help you detect and troubleshoot issues quickly, ensuring that your applications run smoothly and efficiently. Prometheus is a great tool for monitoring, paired with Grafana for visualization.

Estimated Time: 3-4 weeks

Essential Concepts

- **Prometheus (Recommended)**
  - Basics (installation, metrics, exporters)
  - Alerting
  - PromQL (Prometheus Query Language)
- **Grafana**
  - Basics (installation, dashboard)
  - Virtualization (integration with Prometheus, creating virtualizations)
- **ELK Stack**
  - Basics (installation, data collection)
  - Elasticsearch (indexing, searching)
  - Logstash (data pipeline, filters)
  - Kibana (creating virtualizations, dashboards)

Becoming a DevOps engineer is a journey. Be patient with yourself and stay persistent, even when things get tough.

- Mosh

