DevOps Engineering

Total Duration: 40 hours

This course is delivered as instructor-led, hands-on training. Each module pairs conceptual grounding with a guided lab, and every lab builds on the artifacts produced in the previous module — the repository from Module 02 feeds the pipeline in Module 03, the image from Module 05 is deployed in Module 06, and the infrastructure built by hand in Module 07 is reproduced as code in Module 08.

Course Schedule

#	Module	Duration
01	DevOps Fundamentals	2 hrs
02	GIT Version Controlling	3 hrs
03	Jenkins CI/CD Pipeline	6 hrs
04	Prometheus and Grafana Monitoring	4 hrs
05	Docker Containerization	8 hrs
06	Kubernetes Orchestration of Docker Container	10 hrs
07	AWS Cloud Computing	4 hrs
08	Infrastructure as a Code (IaaC)	3 hrs
	Total	**40 hrs**

Detailed Module Breakdown

Module 01: DevOps Fundamentals

Duration: 2 hours

Learning Objectives

- Explain what DevOps is and why organizations adopt it.
- Describe the DevOps lifecycle and how it differs from traditional SDLC.
- Set up a working lab environment for the rest of the course.

Topics Covered

1. What is DevOps? Culture, practices, and tools (the CALMS framework)
2. The DevOps lifecycle: Plan, Code, Build, Test, Release, Deploy, Operate, Monitor
3. Traditional SDLC vs. DevOps — where the bottlenecks disappear
4. Roles and responsibilities in a DevOps team
5. Preparing the lab environment: Linux VMs, networking, and required tooling

Hands-on Lab: Provision two Linux VMs (a control node and a target node), install baseline packages, and verify SSH connectivity between them.

Module 02: GIT Version Controlling

Duration: 3 hours

Learning Objectives

- Use Git to track and manage source code changes.
- Apply branching and merging strategies in a team workflow.
- Recover from mistakes using stash, reset, and checkout.

Topics Covered

1. Git features and the distributed version control model
2. Three-tier architecture: working directory, staging area, repository
3. Core workflow: clone, add, commit, push, pull, fetch
4. Merge vs. rebase — when to use each
5. Stash, reset (soft/mixed/hard), checkout, and revert
6. Resolving merge conflicts
7. Branching workflows used in real teams (feature branching, GitFlow overview)

Hands-on Lab: Create a shared repository, clone it, commit changes on a feature branch, merge it back, deliberately create and resolve a conflict, then practise rebase and stash.

Module 03: Jenkins CI/CD Pipeline

Duration: 6 hours

Learning Objectives

- Install, configure, and secure a Jenkins server.
- Author declarative pipelines using a Jenkinsfile.
- Scale Jenkins with distributed build agents.

Topics Covered

1. Installing and configuring Jenkins on Linux
2. Jenkins job types: Freestyle, Pipeline, Multibranch Pipeline
3. Writing a Jenkinsfile: declarative vs. scripted syntax, sequential build stages
4. Configuring Jenkins Master and Agent (slave) nodes for distributed builds
5. Securing Jenkins: authentication, authorization matrix, credentials, and user management
6. Essential plugins: SCM (Git) plugin, build and test integrations
7. Connecting Jenkins to a Git repository to trigger automated builds

Hands-on Lab: Install Jenkins, connect it to the Git repository from Module 02, write a Jenkinsfile with build/test/deploy stages, attach an agent node, and configure role-based access for a second user.

Module 04: Prometheus and Grafana Monitoring

Duration: 4 hours

Learning Objectives

- Collect infrastructure metrics using Prometheus.
- Visualize metrics and build alerts using Grafana.

Topics Covered

1. Monitoring concepts: metrics, time-series data, and exporters
2. Installing and configuring Prometheus on a Linux server
3. Deploying Node Exporter to collect system metrics from multiple nodes
4. Prometheus configuration: scrape configs, targets, and service discovery
5. Querying metrics with PromQL basics

6. Installing and configuring Grafana
7. Connecting Grafana to a Prometheus data source and building dashboards

Hands-on Lab: Install Prometheus and Node Exporter on two nodes, configure scrape targets, then build a Grafana dashboard visualizing CPU, memory, and disk usage across both nodes.

Module 05: Docker Containerization

Duration: 6 hours

Learning Objectives

- Explain containerization and how it differs from virtualization.
- Build, tag, and manage Docker images and containers.
- Package a multi-service application with Docker Compose.

Topics Covered

1. Containers vs. virtual machines
2. Installing and configuring Docker Engine on Linux
3. Docker architecture: daemon, client, images, containers, registries
4. Docker Hub and essential Docker CLI commands
5. Writing Dockerfiles: layers, caching, and best practices
6. Building, tagging, and versioning images
7. Docker networking and volumes
8. Docker Compose for multi-container applications

Hands-on Lab: Write a Dockerfile for a sample application, build and push the image to Docker Hub, then containerize a multi-service application (app + database) using docker-compose.

Module 06: Kubernetes Orchestration of Docker Container

Duration: 6 hours

Learning Objectives

- Explain why container orchestration is needed at scale.
- Deploy, scale, and update applications on a Kubernetes cluster.

Topics Covered

1. Why Kubernetes? The problems it solves beyond plain Docker
2. Kubernetes architecture: control plane (API server, scheduler, controller manager, etcd) and worker node components (kubelet, kube-proxy, container runtime)
3. Installing and configuring a Kubernetes cluster
4. Core objects: Pods, ReplicaSets, Deployments, Services
5. Writing and applying YAML manifests
6. ConfigMaps, Secrets, and namespaces
7. Deploying a real project to the cluster, scaling replicas, and performing rolling updates

Hands-on Lab: Stand up a Kubernetes cluster, deploy the Dockerized application from Module 05 using a Deployment and Service, scale it to multiple replicas, and perform a rolling update.

Module 07: AWS Cloud Computing

Duration: 8 hours

Learning Objectives

- Explain core cloud computing concepts and AWS's global infrastructure.
- Provision networking, compute, storage, and database resources on AWS.

Topics Covered

1. Cloud computing models: IaaS, PaaS, SaaS; public, private, and hybrid cloud
2. AWS global infrastructure: regions and availability zones
3. AWS account setup and IAM basics: users, roles, and policies
4. Creating a custom VPC: subnets, route tables, internet gateway, security groups
5. Launching and configuring an EC2 instance
6. Storage: EBS volumes and snapshots; creating and managing S3 buckets
7. Introduction to RDS: engines, Multi-AZ, and common use cases
8. Introduction to Route 53: hosted zones and routing policies

Hands-on Lab: Design and build a custom VPC with public and private subnets, launch an EC2 instance inside it, attach an EBS volume, create an S3 bucket, provision an RDS instance, and configure a Route 53 hosted zone.

Module 08: Infrastructure as a Code (IaaC)

Duration: 5 hours

Learning Objectives

- Explain the benefits of Infrastructure as Code over manual provisioning.
- Provision and manage AWS resources using Terraform.

Topics Covered

1. Introduction to Infrastructure as Code: declarative vs. imperative approaches
2. Getting started with Terraform: architecture, providers, and state
3. Terraform workflow: init, plan, apply, destroy
4. Working with Terraform: variables, outputs, and modules
5. Terraform with AWS: provisioning a VPC, EC2 instance, and S3 bucket as code
6. Terraform provisioners: local-exec and remote-exec

Hands-on Lab: Write Terraform configuration to reproduce the VPC and EC2 instance built manually in Module 07, parameterize it with variables, apply it, then tear it down with a single command.