# 9-Phase Plan for DevOps Mastery

## Phase 1: Programming Languages (Python and Java)

**Duration**: 40 Days  
**Goal**: Gain proficiency in Python and Java, focusing on automating tasks, managing backend services, and writing scripts for DevOps tasks.

### Week 1-2 (Java):

* + Focus: Java for backend microservices.
  + Learn:
    - Java syntax, OOP principles (inheritance, polymorphism, encapsulation).
    - Spring Boot framework for building RESTful APIs.
    - Build and deploy simple microservices with Spring Boot.
  + Projects: Create a Java-based microservice that interacts with a database and deploy it locally.

### Week 3-4 (Python):

* + Focus: Basics and Intermediate Python for DevOps Automation
  + Learn:
    - Python syntax, variables, loops, and functions.
    - File handling and working with external libraries (e.g., requests).
    - Scripting for automation (e.g., system administration tasks like monitoring, log parsing).
  + Projects: Write automation scripts to handle file systems and network communication.

### Week 5-6 (Advanced Python and Java):

* + Focus: Advanced use cases in Python and Java for DevOps.
  + Learn:
    - Multithreading, asynchronous tasks in Python.
    - Working with APIs in both languages.
    - Writing tests in Python and Java.
  + Projects: Automate CI/CD pipeline tasks (Python) and build a microservice (Java) that can be integrated into Kubernetes.

## Phase 2: Version Control (Git, GitHub)

**Duration**: 10 Days  
**Goal**: Learn version control best practices, branching strategies, and collaboration with Git and GitHub.

### Day 1-3:

* + Learn:
    - Git basics: commit, push, pull, merge, branch, rebase.
    - Branching strategies (e.g., Gitflow, feature branches).
  + Practice: Create branches, commit changes, resolve merge conflicts, and contribute to an open-source project.

### Day 4-6:

* + Learn:
    - Collaborating with others: Pull Requests, code reviews.
    - Setting up GitHub Actions for CI/CD pipelines.
  + Projects: Implement GitHub Actions for automated testing of your Python and Java projects.

### Day 7-10:

* + Learn: Tagging releases, version control for IaC, and integrating Git into CI/CD pipelines.
  + Project: Manage your personal projects using Git and GitHub for both Python and Java.

## Phase 3: Linux (Basics, Shell Scripting, Networking)

**Duration**: 25 Days  
**Goal**: Understand Linux commands, shell scripting, and networking fundamentals essential for DevOps.

### Week 1 (Basics):

* + Learn:
    - Navigating the Linux filesystem.
    - Essential commands (cp, mv, chmod, grep, etc.).
    - User management and permissions.
  + Practice: Manage users, set file permissions, and schedule jobs with cron.

### Week 2 (Shell Scripting):

* + Learn:
    - Writing shell scripts to automate system administration tasks.
    - Conditional logic, loops, and functions in bash.
    - Handling files and processes with shell scripts.
  + Projects: Write a shell script to automate system monitoring (disk usage, memory, etc.).

### Week 3 (Networking):

* + Learn:
    - Basic networking concepts (IP addresses, DNS, DHCP, routing).
    - Network utilities (ping, netstat, ssh, scp).
    - Managing firewalls (iptables, ufw).
  + Projects: Set up SSH keys for secure remote access and automate backup processes.

## Phase 4: Containerization (Docker, Kubernetes, Helm)

**Duration**: 35 Days  
**Goal**: Master containerization using Docker and orchestration with Kubernetes and Helm for production-ready applications.

### Week 1 (Docker):

* + Learn:
    - Docker basics: images, containers, volumes, networks.
    - Writing Dockerfiles and using Docker Compose.
  + Project: Dockerize a Python and Java application, create Docker Compose files to manage multi-container environments.

### Week 2 (Kubernetes Basics):

* + Learn:
    - Kubernetes architecture (pods, services, deployments).
    - Managing clusters and deploying applications.
  + Project: Deploy your Dockerized applications in a local Kubernetes cluster (minikube).

### Week 3 (Advanced Kubernetes):

* + Learn:
    - Scaling applications, managing secrets and config maps.
    - Kubernetes networking (services, ingress controllers).
  + Project: Set up an ingress controller and deploy a multi-tier microservice application.

### Week 4 (Helm):

* + Learn:
    - Helm charts for Kubernetes.
    - Creating and deploying Helm charts.
  + Project: Convert your Kubernetes deployment to a Helm chart and manage updates using Helm.

## Phase 5: Microservices

**Duration**: 20 Days  
**Goal**: Understand and implement microservices architecture using best practices for DevOps.

### Week 1:

* + Learn:
    - Microservices architecture principles (loose coupling, scalability).
    - API gateways, service discovery, and inter-service communication.
  + Project: Break down a monolithic application into smaller microservices.

### Week 2:

* + Learn:
    - Deploying and managing microservices with Kubernetes.
    - Observability in microservices (monitoring, logging, tracing).
  + Project: Deploy your microservices to Kubernetes with logging and monitoring integrated.

## Phase 6: CI/CD Tools (Jenkins, GitLab, GitHub Actions, CircleCI)

**Duration**: 25 Days  
**Goal**: Build fully automated pipelines for building, testing, and deploying applications.

### Week 1 (Jenkins):

* + Learn:
    - Jenkins setup, pipeline as code, and integration with version control.
    - Automating builds, tests, and deployments with Jenkins.
  + Project: Create a Jenkins pipeline to build and deploy a Java microservice.

### Week 2 (GitLab, GitHub Actions, CircleCI):

* + Learn:
    - GitLab CI/CD pipelines and GitHub Actions for automation.
    - CircleCI for integration and deployment tasks.
  + Project: Implement CI/CD pipelines using different tools for your Python and Java projects.

## Phase 7: Infrastructure as Code (Terraform, Ansible, Pulumi)

**Duration**: 20 Days  
**Goal**: Manage infrastructure with code to improve scalability, consistency, and collaboration.

### Week 1 (Terraform):

* + Learn:
    - Terraform basics: writing and managing infrastructure using HCL.
    - Provisioning infrastructure in AWS/GCP using Terraform.
  + Project: Use Terraform to create a simple AWS/GCP infrastructure (e.g., EC2 instances, S3 buckets).

### Week 2 (Ansible, Pulumi):

* + Learn:
    - Ansible for configuration management and automation.
    - Pulumi for infrastructure as code with programming languages.
  + Project: Automate configuration changes across multiple servers using Ansible and create infrastructure with Pulumi.

## Phase 8: Cloud Providers (AWS, Azure, GCP)

**Duration**: 15 Days  
**Goal**: Gain experience in managing cloud infrastructure and services for DevOps.

### Week 1:

* + Learn:
    - Core services in AWS (EC2, S3, IAM, RDS) or your chosen provider.
    - Deploying applications on AWS.
  + Project: Deploy a microservice on AWS using EC2, configure S3 for storage.

### Week 2:

* + Learn:
    - Load balancing, autoscaling, and security best practices.
    - Monitoring cloud infrastructure with AWS CloudWatch (or equivalents in Azure/GCP).
  + Project: Set up auto-scaling and monitoring for a web application deployed in the cloud.

## Phase 9: Monitoring and Logging (Prometheus, Grafana, ELK)

**Duration**: 15 Days  
**Goal**: Implement robust monitoring and logging systems to ensure application reliability.

### Week 1 (Prometheus, Grafana):

* + Learn:
    - Monitoring metrics with Prometheus and visualizing data with Grafana.
  + Project: Set up a monitoring system for a Kubernetes-deployed microservice using Prometheus and Grafana.

### Week 2 (ELK Stack):

* + Learn:
    - Centralized logging with the ELK stack (Elasticsearch, Logstash, Kibana).
  + Project: Implement an ELK stack to centralize logs for your microservices and Kubernetes clusters.