DevOps Course Curriculum

Phase 1: Linux Essentials (30 hours)

1)Linux Fundamentals

- File system structure, basic commands, users, groups
- File permissions, process management, package management (DNF/YUM)

2)Shell Scripting & Automation

- Bash scripting basics, variables, loops, functions
- Automating system tasks
- Cronjob

3)Networking in Linux

- Configuring network interfaces, firewall (firewalld, iptables)
- SSH, SCP, NFS, DNS, SFTP,Rsync

4)RedHat Enterprise Linux (RHEL) Administration

- System monitoring (top, ps, journalctl, systemctl)
- SELinux, system logs, troubleshooting
- Tar Archives

5)Web Servers (Apache & Nginx)

- Installing and configuring Apache (4 hrs)
- Installing and configuring Nginx (4 hrs)
- Load Balancing, Reverse Proxy, Performance tuning (4 hrs)

Phase 2: Networking Essentials

1)Networking Basics(6 hours)

- OSI model, TCP/IP, subnetting, VLANs, routing, NAT
- Ports, protocols (HTTP, HTTPS, FTP, SSH, DNS, DHCP)

2)Networking Tools & Troubleshooting(6 hrs)

- Ping, Traceroute, Netstat, Curl
- Configuring and troubleshooting networking in Linux

3)Advanced Networking Concepts(6 hrs)

- Load balancing, VPNs, DNS management
- Network security and firewall

Phase 3: Git & Version Control (8 hours)

1)Git Fundamentals

- Installing Git, configuring repositories, basic commands

2)Branching & Merging

- Working with branches, resolving merge conflicts

3)Git Workflows

- GitHub/GitLab workflows, CI/CD integration

Phase 4: CI/CD Tool- Jenkins (20 hours)

1)Introduction and Basics

- Introduction to Jenkins
- Basics of SCM
- Basics of CI/CD

2)Jenkins Setup and Interface

- Jenkins Architecture
- Jenkins Installation Options
- Demo: Jenkins Installation

- Jenkins User Interface Overview
- Types of Jenkins Projects
- Working with Freestyle Job
- Chained Freestyle Projects

3)Extending Jenkins

- Jenkins Plugins
- Installing Plugins
- Controller Failure - Freestyle Project
- Jenkins Fingerprints

4)Jenkins Pipelines

- Pipeline and Jenkinsfile
- Additional Pipeline Configuration
- Simple Pipeline Job
- Build and Test via Pipeline

- Pipeline Script from SCM
- Controller Failure - Pipeline Project
- Create Pipeline using Blue Ocean Graphical Editor
- Create Pipeline with Parameters

5)Automation and security

- Automating Jenkins using CLI and APIs
- Jenkins CLI - Build a job
- Jenkins REST API - Install a Plugin
- Jenkins CSRF – CRUMB
- Jenkins Security Overview
- Jenkins Authentication
- Jenkins Authorization - Matrix Authorization Strategy

Phase 5: AWS Cloud Computing (34 hours)

1)AWS Basics

- IAM, EC2, S3, VPC, RDS, Route 53, CloudWatch

2)Compute & Storage Services

- Launching EC2 instances, configuring security groups, S3 storage

3)Networking in AWS

- VPC, Subnets, Security Groups, Elastic Load Balancer (ELB)

4)Automation & Infrastructure as Code in AWS

- CloudFormation, AWS CLI, Terraform basics

Phase 6: Containers & Podman (40 Hours)

1)Introduction and overview of containers

- Describe how containers facilitate application development.

2)Podman basics

- Manage and run containers with Podman.

3)Container images

- Navigate container registries to find and manage container images.

4)Custom container images

- Build custom container images to containerize applications.

5)Persisting data

- Run database containers with persistence.

6)Troubleshooting containers

- Analyze container logs and configure a remote debugger.

7)Multi-container applications with compose

- Run multi-container applications using Compose.

10)Container orchestration with Kubernetes and OpenShift

- Orchestrate containerized applications with Kubernetes and OpenShift**.**

Phase 7: Container Orchestration (70 hours)

1)Introduction to Kubernetes and OpenShift

- Identify the main Kubernetes cluster services and OpenShift platform services, and monitor them from the web console.

2)Kubernetes and OpenShift Command-Line Interfaces and APIs

- Access an Kubernetes cluster from the command line, and query its Kubernetes API resources to assess the health of a cluster.

3)Run Applications as Containers and Pods

- Run and troubleshoot containerized applications as unmanaged Kubernetes pods.

4)Deploy Managed and Networked Applications on Kubernetes

- Deploy applications and expose them to network access from inside and outside a Kubernetes cluster.

5)Manage Storage for Application Configuration and Data

- Externalize application configurations in Kubernetes resources, and provision storage volumes for persistent data files.

6)Configure Applications for Reliability

- Configure applications to work with Kubernetes for high availability and resilience.

7)Declarative Resource Management

- Deploy and update applications from resource manifests that are parameterized for different target environments.

8)Deploy Packaged Applications

- Deploy and update applications from resource manifests that are packaged for sharing and distribution.

9)Authentication and Authorization

- Configure authentication with the HTPasswd identity provider and assign roles to users and groups.

10)Network Security

- Protect network traffic between applications inside and outside the cluster.

11)Expose non-HTTP/SNI Applications

- Expose applications to external access without using an Ingress controller.

12)Enable Developer Self-Service

- Configure clusters for safe self-service by developers from multiple teams and disallow self-service if projects have to be provisioned by the operations staff.

13)Application Security

- Run applications that require elevated or special privileges from the host Operating System or Kubernetes.

Phase 8: Infrastructure as Code (Ansible) (40 hours)

1)Introduce Ansible

- Describe the fundamental concepts of Red Hat Ansible Automation Platform and how it is used, and install Red Hat Ansible Automation Platform.

2)Implement an Ansible playbook

- Create an inventory of managed hosts, write a simple Ansible playbook, and run the playbook to automate tasks on those hosts.

3)Manage variables and facts

- Write playbooks that use variables to simplify management of the playbook and facts to reference information about managed hosts.

4)Implement task control

- Manage task control, handlers, and task errors in Ansible Playbooks.

5)Deploy files to managed hosts

- Deploy, manage, and adjust files on hosts managed by Ansible.

6)Manage complex plays and playbooks

- Write playbooks that are optimized for larger, more complex plays and playbooks.

7)Simplify playbooks with roles

- Use Ansible roles to develop playbooks more quickly and to reuse Ansible code.

8)Troubleshoot Ansible

- Troubleshoot playbooks and managed hosts.

9)Automate Linux administration tasks

- Automate common Linux system administration tasks with Ansible.