DEVSECOPS Syllabus

LINUX

- **Fundamentals of Linux**

  - Core Components of Linux Machine
  - Linux Distributions
  - Setup Linux Distributions in Virtual Machine, Docker
  - Package Management
  - Folder Structure of Linux
  - Basic Linux Networking

- **User Management**

  - Introduction to User Management in Linux
  - Creating Users in Linux
  - Managing User Passwords
  - Enforcing Password Policies
  - Modifying Users
  - Deleting Users
  - Working with Groups
  - Sudo Access and Privilege Escalation
  - Granting Specific Commands with sudo

- **File Management**

  - File and Directory Management
  - File Viewing and Editing
  - Basic Navigation
  - Different types of editor in linux
  - Insert Mode Shortcuts
  - Editing Text
  - Search and Replace
  - Working with Multiple Files

GIT Version Control System

- **Introduction to Version Control**
  - What is version control?
  - Benefits of using version control
  - Centralized vs Distributed version control
  - Overview of Git and GitHub/GitLab/Bitbucket

- **GitHub/GitLab Basics**
  - Creating a GitHub/GitLab account
  - Creating a new repository on the web interface
  - Linking local repo to remote
  - Public vs Private repositories

- **Installing and Setting Up Git**
  - Installing Git on Windows/Mac/Linux
  - Verifying installation: git --version
  - Configuring Git for the first time.

- **Basic Git Workflow**
  - Creating a Git repository: git init
  - Cloning a repository: git clone
  - Checking status: git status

- **Tracking Changes**
  - Adding files to staging: git add
  - Committing changes: git commit -m "message"
  - Viewing commit history: git log

- **Working with Files**

  - Ignoring files with .gitignore
  - Removing files: git rm
  - Renaming files: git mv

- **Undoing Changes (Safe Reverts)**

  - Unstaging a file: git reset <file>
  - Amending the last commit: git commit --amend
  - Discarding changes in a file: git checkout -- <file>
  - Viewing previous states: git show <commit_id>

- **Branching Basics**

  - Creating a branch: git branch <branch-name>
  - Switching branches: git checkout <branch-name>
  - Creating and switching: git checkout -b <branch-name>
  - Merging branches: git merge <branch-name>
  - Deleting a branch: git branch -d <branch-name>
  - Reseting changes: Git reset –hard origin/<branch>

- **Working with Remote Repositories**

  - Adding a remote: git remote add origin <url>
  - Viewing remotes: git remote -v
  - Pushing changes: git push -u origin main
  - Pulling updates: git pull origin main
  - Cloning repositories from GitHub/GitLab

- **Best Collaboration Practices**

  - Pull before push
  - Avoid committing to main directly
  - Use clear and concise commit messages

Databases

- **Core SQL Concepts**
  - Basics of RDBMS (PostgreSQL, MySQL)
  - CRUD operations (SELECT, INSERT, UPDATE, DELETE)
  - Joins, Subqueries, and Aggregations
- **Core NoSQL Concepts**
  - Types: Document (MongoDB), Key-Value (Redis)
  - Understanding when to use NoSQL over SQL
  - CRUD operations in MongoDB
  - Data modeling for NoSQL
- **Database Security Essentials**
  - Role-based access control (RBAC)
  - Least privilege principle for users
  - SQL Injection detection and prevention
  - Password encryption and SSL connections
  - Audit logging (e.g., PostgreSQL pgaudit)
  - Remote Connection for Developers
- **Monitoring & Compliance**
  - Enable and monitor slow query logs
  - Track permission changes
  - Automated backup & restore testing (e.g., pg_dump, mysqldump, mongoexport, mongodump)
  - Encryption at rest and in transit.

Python

- **Python Fundamentals**

  - Variables, data types (str, int, float, bool)
  - Control structures (if, for, while)
  - Functions and scopes
  - List, Tuple, Set, Dictionary operations
  - Exception handling (try, except, finally)
  - File I/O operations

- **Python for Automation**

  - Reading/writing config files (.ini, .yaml, .json)
  - Automating command-line tools (subprocess, os)
  - Working with logs and text parsing (re, csv, json)
  - Scheduled tasks (schedule, crontab)

- **Working with APIs & Web**

  - Making API requests (requests, httpx)
  - REST API consumption and automation
  - Web scraping (optional)

DOCKER

- **Introduction to Docker**

  - What is Docker and why use it?
  - Difference between VMs and containers
  - Docker architecture: Engine, Daemon, CLI, Images, Containers
  - Installing docker

- **Docker Basics**

  - docker run, docker ps, docker stop, docker start
  - Understanding container lifecycle
  - Detached vs foreground mode
  - Using docker exec and docker logs

- **Docker Registry**

  - Using Docker Hub, AWS ECR
  - Setting up a private registry
  - docker login, push, pull

- **Docker Images**

  - What is a Docker image?
  - Docker Hub and official images
  - docker pull, docker images, docker rmi
  - Creating a basic image with a Dockerfile
  - docker build

- **Dockerfile Basics**

  - FROM, RUN, CMD, EXPOSE, COPY, ENV
  - Building a custom image
  - Layer caching

- **Docker volumes and Persistence**

  - What are docker volumes?
  - docker volume create, docker run -v
  - Bind mounts vs named volumes
  - Data persistence across containers

- **Docker Networking**

  - Bridge, host, none networks
  - docker network create, docker network inspect
  - Inter-container communication
  - Exposing ports with -p and -P

- **Docker Compose**

  - Why Docker Compose?
  - docker-compose.yml syntax
  - docker-compose up, down, logs, build
  - Defining services, networks, and volumes
  - Environment Variables and Secrets

- **Dockerfile Optimization**

  - Reducing image size
  - Multi-stage builds
  - Ignoring files with .dockerignore

- **Building for Production**

  - Best practices for Docker in production
  - Stateless containers and 12-Factor App principles
  - Health checks

- **Debugging and Troubleshooting**
    - Inspecting containers (docker inspect)
    - Checking logs and stats (docker logs, docker stats)
    - Debugging build issues
- **Security Best Practices**
    - Least privilege container user
    - Read-only file systems
    - Non-root user
    - Distroless images
    - Multi stage build
- **Docker Cleanup**
    - Uninstalling docker
    - Docker system prune, volume prune, image prune commands

Cloud Computing

- **Introduction to Cloud Computing**

  - Definition and benefits of cloud computing
  - Types of cloud models: IaaS, PaaS, SaaS
  - Deployment models: Public, Private, Hybrid, Multi-cloud
  - AWS Free Tier and pricing models
  - Overview of major cloud providers (focus on AWS)

- **Introduction to AWS**

  - What is AWS?
  - Global infrastructure (Regions, Availability Zones, Edge Locations)
  - AWS Management Console, CLI, and SDKs

- **Compute Services**

  - Amazon EC2: Instances, AMIs, Key Pairs, Security Groups
  - AWS Lambda and serverless computing
  - AWS Elastic Beanstalk

- **Storage Services**

  - Amazon S3: Buckets, Objects, Storage Classes, Versioning
  - Amazon EBS: Volumes, Snapshots
  - Amazon EFS: Elastic File System

- **AWS Networking**

  - Amazon VPC: Subnets, Route Tables, Internet Gateways, NAT
  - Security Groups vs. NACLs
  - Elastic IPs, VPC Peering, Transit Gateway
  - AWS CloudFront (CDN)
  - Route 53 (DNS Management)

- **AWS Databases**

  - Amazon RDS: MySQL, PostgreSQL, Oracle, SQL Server
  - Amazon DynamoDB (NoSQL)
  - Amazon Redshift (Data Warehousing)
  - Amazon Aurora

- **Identity and Access Management (IAM)**

  - AWS IAM: Users, Groups, Roles, Policies
  - MFA (Multi-Factor Authentication)
  - IAM best practices

- **Security and Compliance**

  - Shared Responsibility Model
  - AWS Secret Manager
  - AWS Firewall Manager, AWS Inspector
  - AWS Shield and WAF
  - Data encryption at rest and in transit

- **Cost Management**

  - AWS Pricing Calculator
  - AWS Budgets and Cost Explorer
  - Reserved vs. On-Demand vs. Spot Instances

Web Servers

- **Introduction to Web Servers**

    - What is a web server?
    - Difference between static and dynamic content
    - Comparison: Apache vs. Nginx
    - Use cases and popularity

- **Basics of Apache HTTP Server**

    - Apache architecture and process model
    - Installing Apache on Linux (Debian/Ubuntu/CentOS)
    - Configuration files: httpd.conf, apache2.conf, sites-available, sites-enabled
    - Starting, stopping, and restarting Apache
    - Virtual Hosts (Name-based and IP-based hosting)

- **Apache Core Configuration**

    - Directory structure and default document root
    - Setting up custom error pages
    - Directory Indexing and .htaccess files
    - MIME types and content types
    - Logging: access.log and error.log

- **Apache Modules and Features**

    - Enabling and disabling modules
    - Common modules: mod_rewrite, mod_ssl, mod_proxy, mod_headers
    - URL rewriting and redirection
    - Basic authentication and access control
    - Enabling HTTPS with Let's Encrypt or self-signed SSL

- **Basics of Nginx**

  - Nginx architecture and event-driven model
  - Installing Nginx on Linux
  - Configuration files: nginx.conf, sites-available/, sites-enabled/
  - Starting, stopping, and restarting Nginx
  - Serving static files

- **Nginx Core Configuration**

  - Server blocks (similar to Apache virtual hosts)
  - Location blocks and URI matching
  - Root vs. alias directives
  - Custom error pages
  - Logging: access.log and error.log

- **Nginx Reverse Proxy and Load Balancing**

  - Reverse proxy configuration for a backend service
  - Load balancing with round robin, IP hash, least connections
  - Health checks
  - Caching static and dynamic content

- **Nginx Security and SSL**

  - Setting up SSL with Let's Encrypt or self-signed certs
  - HTTP to HTTPS redirection
  - Rate limiting and connection limiting
  - IP whitelisting/blacklisting

CI/CD Workflows

- **Introduction to CI/CD**

  - What is Continuous Integration (CI)?
  - What is Continuous Delivery (CD) and Continuous Deployment?
  - Benefits of CI/CD in software development
  - Overview of CI/CD tools (Jenkins, GitLab CI, GitHub Actions, etc.)

- **Basics of Jenkins**

  - Jenkins architecture and components (Master-Agent model)
  - Installing Jenkins on Linux
  - Jenkins Web UI overview

- **Jenkins Configuration and Setup**

  - User management and security
  - Global tool configuration (JDK, Git, Maven, Docker, etc.)
  - Installing and managing plugins
  - Backup and restore Jenkins configurations

- **Building Projects with Jenkins**

  - Freestyle project setup
  - Configuring source code repository (Git/GitHub/GitLab/Bitbucket)
  - Running shell/batch commands in build steps

- **Jenkins Pipelines (Core of CI/CD)**

  - Introduction to Jenkins Pipeline (Declarative vs Scripted)
  - Creating and running a simple Jenkinsfile
  - Stages, steps, and post actions
  - Pipeline syntax and best practices
  - Using input, when, and parallel directives.

Kubernetes

- **Introduction to Kubernetes**

  - Evolution from virtual machines to containers to Kubernetes
  - Kubernetes vs Docker Swarm
  - Core concepts and terminology
  - Kubernetes architecture overview (Master, Node, etcd, Kubelet, Kube-Proxy, API Server)

- **Setting Up Kubernetes**

  - Installing and Setting up Kubernetes
  - Installing and using kubectl
  - Cluster configuration and kubeconfig file
  - Namespaces and resource isolation

- **Core Kubernetes Objects**

  - Pods
  - ReplicaSets
  - Deployments
  - Services
  - DaemonSets
  - StatefulSets
  - ConfigMaps & Secrets – Managing configuration and sensitive data
  - Volumes , Storage Class & PersistentVolumes – Volume types, PVCs, dynamic provisioning

- **Application Management**

  - Creating and managing YAML manifests
  - Labels, selectors, and annotations
  - Health checks: liveness and readiness probes
  - Taints, tolerations, node selectors, and affinities
  - Init containers and sidecars

- **Networking in Kubernetes**

  - Kubernetes networking model explained
  - Pod-to-Pod communication
  - Service discovery and DNS in Kubernetes
  - Network policies (isolation and access control)
  - Ingress and Ingress Controllers (NGINX Ingress)
  - TLS termination and HTTPS via Ingress

- **Storage in Kubernetes**

  - Volume types: emptyDir, hostPath, NFS, CSI
  - Persistent Volumes (PV) and Persistent Volume Claims (PVC)
  - StorageClasses and dynamic provisioning
  - StatefulSets for stateful applications
  - Volume lifecycle and data persistence

- **Helm – Kubernetes Package Manager**

  - What is Helm and why use it
  - Installing Helm and configuring repositories
  - Helm charts: structure and customization
  - Deploying applications using Helm
  - Creating and packaging your own Helm charts
  - Using values.yaml and templating

- **Kubernetes Security**

  - Role-Based Access Control (RBAC)
  - Service Accounts and Permissions
  - Network policies for pod-level security
  - Securing secrets and config maps
  - Pod security standards and admission controllers
  - Image scanning and signing

- **CI/CD Integration with Kubernetes**

  - Integrating Jenkins with Kubernetes
  - Building and deploying container images
  - Using Kubernetes for canary and blue/green deployments

- **Kubernetes Scaling and High Availability**

  - Horizontal Pod Autoscaler (HPA)
  - Vertical Pod Autoscaler (VPA)
  - Cluster Autoscaler

Istio Service Mesh

- **Service-Mesh Fundamentals**

  - What is service mesh is and why it exists
  - Sidecar proxy pattern (Envoy)
  - Control plane vs. data plane

- **Istio Architecture Overview**

  - Core components: istiod, Envoy proxy, Pilot, Citadel, Galley (conceptual)
  - How Istio integrates with Kubernetes resources (Pods, Services, Deployments)
  - High-level traffic flow through an Envoy sidecar

- **Installation & First Cluster Setup**

  - Prerequisites (Kubernetes ≥ 1.27, kubectl, istioctl)
  - Installing with istioctl install (default profile)
  - Enabling automatic sidecar injection via namespace label
  - Verifying installation with istioctl verify-install and basic health checks

- **Traffic Management Basics**

  - VirtualService, Gateways and Destination Rules
  - Layer-7 routing scenarios: Simple path-based routing

Observability (Monitoring, Logging, Alerting and Auditing)

- **Introduction to Observability**
  - Difference between Monitoring, Logging, and Alerting
  - Importance in DevOps and SRE practices
  - Key terms: Metrics, Logs, Traces, Events

- **Monitoring with Prometheus**
  - What is Prometheus and its use case
  - Prometheus architecture: time-series DB, exporters, pull model
  - Installing Prometheus in Kubernetes (Helm or YAML)
  - Configuring prometheus.yml scrape configs
  - Using node-exporter, kube-state-metrics, and cAdvisor
  - Writing basic PromQL queries
  - Visualizing metrics from Prometheus UI
  - Setting up Prometheus retention and storage limits

- **Visualization with Grafana**
  - What is Grafana and its role in observability
  - Connecting Grafana to Prometheus as a data source
  - Creating and importing dashboards
  - Using Grafana templating, variables, and panel types
  - Dashboard best practices for microservices

- **Centralized Logging with EFK Stack (Elasticsearch, Fluentd, Kibana)**
  - Introduction to the EFK stack
  - Setting up Elasticsearch in Kubernetes
  - Installing Fluentd as a log forwarder
  - Parsing and filtering logs in Fluentd
  - Deploying Kibana and accessing dashboards
  - Creating visualizations and search patterns in Kibana
  - Index management and retention in Elasticsearch
  - Best practices for log rotation and log size management

- **Alerting with Prometheus Alertmanager**

  - Overview of Alertmanager architecture
  - Defining alerting rules in Prometheus
  - Routing alerts to different receivers (email, Slack, etc.)
  - Setting up silence, inhibition, and grouping
  - Configuring alert thresholds and severity levels
  - Integrating Alertmanager with Grafana
  - Managing alert history and status

- **Runtime Security Monitoring with Falco**

  - Runtime Security Monitoring with Falco
  - What is Falco and why it's used for Kubernetes security
  - Installing Falco using Helm or daemonset
  - Falco rule structure and writing custom rules
  - Common rule examples (exec shell, privilege escalation, etc.)
  - Logging Falco events to stdout, file, or external systems
  - Integrating Falco with Alertmanager or Slack

Infrastructure as Code (IaC)

- **Intro to IaC**
  - What is IaC and why it's important
  - Declarative vs. imperative IaC tools
  - Overview of Terraform

- **Terraform Basics**
  - Installation of Terraform (CLI)
  - Understanding providers and resources
  - Basic folder/project structure
  - Terraform workflow: init, plan, apply, destroy

- **Terraform Configuration**
  - Writing your first .tf file
  - Creating basic resources: AWS EC2 instance, S3 bucket
  - Understanding Terraform state (terraform.tfstate)
  - Using .terraform.lock.hcl and .terraform/ folder

- **Variables and Outputs**
  - Declaring input variables (variable)
  - Assigning variable values via CLI, .tfvars, and environment
  - Output values using output block
  - Sensitive data handling

- **Terraform State Management**
  - Local state vs. remote state
  - Common issues with state and how to fix them
  - State locking and state backend overview (e.g., AWS S3 + DynamoDB)

- **Terraform Modules (Intro)**

  - What are modules and why use them
  - Creating a simple reusable module
  - Using official modules from the Terraform Registry
  - Calling modules with inputs and outputs

- **Basic Provisioners and Data Sources**

  - Using provisioner (e.g., remote-exec, local-exec) – best used for demo
  - Using data sources to reference existing resources
  - Example: Fetching latest AMI ID using data "aws_ami"

- **Best Practices for Beginners**

  - Use .gitignore for state files
  - Don’t hardcode credentials (use environment variables or IAM roles)
  - Keep configurations modular
  - Always run terraform plan before apply

Ansible

- **Introduction to Configuration Management**
    - What is Configuration Management?
    - Push vs Pull models
    - Why Ansible? (vs Chef, Puppet, SaltStack)
    - Use cases of Ansible in DevOps and automation

- **Ansible Architecture Basics**
    - How Ansible works: agentless over SSH
    - Key components: Controller Node, Managed Nodes
    - Inventory: static vs dynamic
    - Modules, Tasks, Playbooks, Roles (high-level overview)

- **Installing Ansible**
    - Installing and Setting up Ansible
    - Verifying Ansible installation
    - Setting up SSH access to managed nodes

- **Working with Inventories**
    - Static inventory file format (hosts)
    - Grouping hosts
    - Using ansible-inventory command
    - Introduction to dynamic inventories (concept only)

- **Running Ad-Hoc Commands**
    - Using ansible command-line tool
    - Running shell, ping, and package modules
    - Common modules: ping, yum, apt, copy, file, service

- **Writing Ansible Playbooks**

  - Playbook structure (YAML format)
  - Hosts, tasks, and handlers
  - Using ansible-playbook command

- **Variables and Facts**

  - Defining variables: in playbooks, inventory, CLI
  - Using vars, vars_files, and host_vars/group_vars
  - Gathering and using facts (ansible_facts)

- **Conditionals and Loops**

  - Using when to apply conditions
  - Looping with with_items, loop, with_dict
  - Registering and using output from tasks

- **Templates with Jinja2**

  - Creating dynamic config files using .j2 templates
  - Using variables in templates
  - Deploying templates with the template module

- **Basic File and Package Management**

  - Using modules: file, copy, template, get_url
  - Managing packages with apt, yum, dnf
  - Managing services with service, system

- **Introduction to Roles**

- What are roles?
- Role directory structure
- Mention how roles promote reusability and modularity

- **Basic Error Handling and Debugging**

  - Using debug module
  - Ignoring errors with ignore_errors
  - Checking exit codes with failed_when, changed_when

Disaster Backup and Recovery

- Snapshots of Volumes and Instances Regulary
- Database backup scripts
- Cross Zone Replication of S3 buckets
- EC2 instances AMI images
- Recovery Strategies for Minimum downtime for EC2 instances, Kubernetes cluster etc.

Additional Security

- Sonarqube (SAST)
- Dependency Checker
- Trivy Scan
- Kubesec, Kubehunter
- Kubernetes Security Specialist Course
- CIS Bechmarking for Linux, Kubernetes Cluster, AWS etc.
- Penetration Testing (OWASP ZAP, Burp Suite)