1. Linux Basics

- **Introduction to Linux**
    - What is Linux? Distribution types (Ubuntu, CentOS, RHEL, etc.)
    - Linux kernel and shell overview.
- **Filesystem Basics**
    - File and directory structure (e.g., `/home`, `/etc`, `/var`, `/usr`, `/opt`).
    - Absolute vs relative paths.
- **Basic Commands**
    - File operations: `ls`, `cd`, `cp`, `mv`, `rm`, `mkdir`, `rmdir`.
    - Viewing files: `cat`, `more`, `less`, `head`, `tail`.
    - File statistics: `stat`, `du`, `df`.

2. User Management

- **Users and Groups**
    - Adding/removing users: `useradd`, `userdel`.
    - Managing groups: `groupadd`, `groupdel`, `usermod`.
- **Permissions**
    - File ownership: `chown`, `chgrp`.
    - File permissions: `chmod`, understanding `rwx` and octal notation.
    - Special permissions: SUID, SGID, Sticky Bit.
- **Switching Users**
    - `su` and `sudo` usage.
    - Configuring `sudoers` file.

3. File Management

- **File Compression and Archiving**
  - `tar`, `gzip`, `gunzip`, `zip`, `unzip`.
- **File Search**
  - Finding files: `find`, `locate`.
  - Searching file contents: `grep`, `egrep`, `awk`.
- **Disk Management**
  - Partitioning: `fdisk`, `parted`.
  - Filesystem creation: `mkfs`, `mount`, `umount`.
  - Disk space monitoring: `df`, `du`.

---

4. Process and System Monitoring

- **Process Management**
  - Viewing processes: `ps`, `top`, `htop`, `pgrep`.
  - Managing processes: `kill`, `pkill`, `killall`, `nice`, `renice`.
- **System Monitoring**
  - Resource usage: `vmstat`, `iostat`, `free`.
  - Logs: `/var/log`, `journalctl`, `dmesg`.
  - Performance monitoring: `iotop`, `sar`.

---

5. Networking

- **Network Configuration**
  - Basics of IP addressing and subnetting.
  - Viewing network interfaces: `ifconfig`, `ip`.
  - Configuring interfaces: `nmcli`, `nmtui`, `ip link`.
- **Network Troubleshooting**
  - Connectivity: `ping`, `traceroute`.

    - DNS: `nslookup`, `dig`.
    - Monitoring: `netstat`, `ss`.
- **Secure Shell (SSH)**
    - Setting up SSH: `ssh-keygen`, `ssh-copy-id`.
    - Secure file transfer: `scp`, `rsync`.
    - SSH configuration: `/etc/ssh/sshd_config`.

---

6. Shell Scripting

- **Scripting Basics**
    - Writing and executing scripts.
    - Shebang (`#!/bin/bash`) usage.
    - Variables, conditionals (`if`, `else`, `elif`), loops (`for`, `while`).
- **Advanced Scripting**
    - Functions, arrays.
    - Reading input (`read`), command substitution.
    - Error handling and debugging (`set -x`).
- **Cron Jobs and Automation**
    - Scheduling tasks: `crontab`, `at`.
    - Automating backups and system maintenance.

---

7. Package Management

- **Package Installation**
    - Debian-based: `apt`, `dpkg`.
    - RHEL-based: `yum`, `dnf`, `rpm`.
- **Repository Management**
    - Adding custom repositories.

    - Updating and upgrading software.

---

8. Security

- **Firewall Management**
    - Using `iptables` and `ufw`.
    - Basics of security groups (related to AWS).
- **User Security**
    - Locking accounts, password policies.
    - Restricting SSH access (IP whitelisting, disabling root login).
- **File Security**
    - Encrypting files: `gpg`, `openssl`.
    - Verifying file integrity: `md5sum`, `sha256sum`.

---

9. System Administration

- **Boot Process**
    - Understanding GRUB, systemd, and init.
    - Troubleshooting boot issues.
- **Service Management**
    - Managing services: `systemctl`, `service`.
    - Enable/disable services on boot.
- **Backup and Restore**
    - Using `rsync`, `tar`, `dd` for backups.
    - Snapshot and AMI backups (AWS-specific).

---

1. GitHub Basics

- **What is GitHub?**
  - Introduction to version control and GitHub as a platform.
  - Key concepts: repositories, branches, commits, pull requests.
- **Getting Started**
  - Creating and cloning repositories.
  - Navigating the GitHub web interface.

---

2. Git Basics for GitHub

- **Version Control with Git**
  - Initializing a repository: `git init`.
  - Adding files: `git add`.
  - Committing changes: `git commit`.
  - Viewing history: `git log`, `git diff`.
- **Branching and Merging**
  - Creating and switching branches: `git branch`, `git checkout`.
  - Merging branches: `git merge`.
  - Resolving merge conflicts.
- **Remote Repositories**
  - Connecting to GitHub: `git remote`.
  - Pushing changes: `git push`.
  - Pulling updates: `git pull`.
  - Fetching changes: `git fetch`.

---

3. Repository Management

- **Repository Settings**
    - Configuring repository settings (visibility, branch protection).
    - Setting up webhooks and integrations.
- **Collaborating on Repositories**
    - Inviting collaborators and managing permissions.
    - Creating and managing teams in GitHub organizations.
- **Managing Large Repositories**
    - Using `.gitignore` to exclude files.
    - Cleaning up history with `git rebase`.
    - Git Large File Storage (LFS) for handling large files.

---

4. Pull Requests and Code Reviews

- **Creating Pull Requests**
    - Opening a pull request from a feature branch.
    - Writing effective pull request descriptions.
- **Code Review**
    - Commenting on code in pull requests.
    - Approving or requesting changes.
    - Understanding pull request statuses (mergeable or not).
- **Merging Pull Requests**
    - Squash and merge, rebase and merge, or create a merge commit.
    - Automating pull request merges with rules.

---

5. Automation in GitHub

- **GitHub Webhooks**

    - Setting up webhooks for triggering events.
    - Integrating GitHub with Jenkins, Terraform, or other DevOps tools.
- **Integrations**
    - Connecting GitHub with third-party tools like Slack, Jira, or .
    - Using GitHub API for custom automations.

---

6. Collaboration Best Practices

- **Commit Guidelines**
    - Writing meaningful commit messages.
    - Using tools like `git commit --amend` to refine commits.
- **Reviewing and Approving PRs**
    - Best practices for collaborative reviews.
    - Using labels and milestones to organize work.
- **Documenting Repositories**
    - Writing effective `README.md` files.
    - Adding contribution guides (`CONTRIBUTING.md`).
    - Using issue templates and PR templates.

---

1. Maven for AWS/DevOps

1.1. Basics of Maven

- **What is Maven?**
  - Introduction to Maven as a build automation and dependency management tool.
  - Key concepts: POM (Project Object Model), plugins, and lifecycle.
- **Installing Maven**
  - Installing on different operating systems.
  - Verifying installation with `mvn -version`.

1.2. Core Maven Concepts

- **Project Structure**
  - Standard directory layout (`src/main/java`, `src/test/java`).
- **Understanding POM File**
  - `<groupId>`, `<artifactId>`, `<version>`: GAV coordinates.
  - Managing dependencies using `<dependencies>` and `<dependency>` tags.
  - Using `<repositories>` for custom Maven repositories.
- **Maven Lifecycle**
  - Default lifecycle: `validate`, `compile`, `test`, `package`, `verify`, `install`, `deploy`.
  - Clean and site lifecycles.
- **Dependency Management**
  - Understanding scopes (`compile`, `provided`, `runtime`, `test`, `system`).
  - Excluding transitive dependencies.
  - Dependency conflict resolution.

1.3. Maven Plugins

- **Common Plugins**
  - Compiler plugin (`maven-compiler-plugin`): Customizing `javac` options.
  - Surefire plugin: Running unit tests.
  - Assembly plugin: Creating archives (ZIP, TAR, etc.).
  - Shade plugin: Creating Uber/Fat JARs.
- **Custom Plugins**
  - Adding custom Maven plugins for specific tasks.
  - Using AWS SDK plugins for Maven.

1.4. Advanced Maven Features

- **Profiles**
  - Defining build profiles in POM files for different environments (dev, test, prod).
  - Activating profiles with `P` flag.
- **Parent POM and Multi-Module Projects**
  - Using parent POM for centralized configuration.
  - Setting up multi-module projects.
- **Using Private Maven Repositories**
  - Setting up Artifactory, Nexus, or AWS CodeArtifact as a Maven repository.
  - Configuring `settings.xml` for private repositories.
- **Customizing Build Process**
  - Adding custom goals to the lifecycle.
  - Running pre- and post-build scripts.

1.5. Maven for CI/CD in DevOps

- **Integration with Jenkins**

- Setting up Maven jobs in Jenkins.
  - Automating builds and deployments using Maven goals.
- **Integration with AWS**
  - Deploying artifacts to AWS S3 using Maven plugins.
  - Building and deploying Java-based AWS Lambda functions.
- **Dockerizing Maven Projects**
  - Building Docker images for Java applications built with Maven.
  - Running Maven commands inside Docker containers.

---

2. npm for AWS/DevOps

2.1. Basics of npm

- **What is npm?**
  - npm as a package manager for Node.js.
  - Understanding npm registry and package.json.
- **Installing npm**
  - Installing Node.js and npm.
  - Verifying installation with `npm -v`.

2.2. Core npm Concepts

- **Package Management**
  - Installing packages locally (`npm install`) and globally (`npm install -g`).
  - Adding packages as dependencies or devDependencies.
  - Updating and removing packages.
- **package.json**
  - Creating and understanding `package.json`.

    - Configuring scripts under the `"scripts"` section.
    - Semantic versioning for dependencies.
- **Lock Files**
    - Purpose of `package-lock.json`.
    - Managing consistent dependency trees.

2.3. Common npm Commands

- **Dependency Management**
    - Installing specific versions: `npm install package@version`.
    - Viewing outdated packages: `npm outdated`.
    - Updating dependencies: `npm update`.
- **Project Lifecycle Commands**
    - Running scripts: `npm run <script>`.
    - Building and testing projects: `npm build`, `npm test`.

2.4. Advanced npm Features

- **Private npm Repositories**
    - Using private registries (e.g., Verdaccio, AWS CodeArtifact).
    - Configuring `.npmrc` for authentication and custom registries.
- **Monorepo Management**
    - Managing monorepos using `npm workspaces`.
- **npm Hooks**
    - Pre- and post-scripts for automating workflows.

2.5. npm for CI/CD in DevOps

- **Integration with Jenkins**
    - Running npm build and test commands in Jenkins pipelines.

    - Automating deployments for Node.js applications.
- **Integration with AWS**
    - Deploying Node.js applications to AWS Lambda.
    - Using AWS SDK for JavaScript to interact with AWS services.
- **Dockerizing npm Projects**
    - Building Docker images for Node.js applications.
    - Running npm commands inside Docker containers.

---

1. Jenkins Basics

1.1. Introduction to Jenkins

- **What is Jenkins?**
    - Jenkins as a CI/CD automation server.
    - Key features: open-source, extensibility with plugins, and distributed builds.
- **Installing Jenkins**
    - Installation on Linux, Windows, and macOS.
    - Running Jenkins in Docker containers.
    - Initial setup and unlocking Jenkins.

1.2. Jenkins Architecture

- Master-agent architecture.
- Understanding the Jenkins pipeline and workspace.
- Distributed builds and scalability.

2. Jenkins Configuration

2.1. Configuring Jenkins

- Configuring system settings: global tools (JDK, Maven, Git).
- Setting up credentials for secure access (SSH keys, AWS keys).
- Customizing user roles and permissions using Role-Based Access Control (RBAC).

2.2. Jenkins Plugins

- **Essential Plugins for DevOps**

    - Git plugin for version control.
    - Pipeline plugin for declarative pipelines.
    - Blue Ocean plugin for a modern UI.
    - Build tools plugins (e.g., Maven, npm).

---

3. Jenkins Pipelines

3.1. Introduction to Pipelines

- **Types of Pipelines**
    - Freestyle projects.
    - Declarative pipelines.
    - Scripted pipelines.
- Advantages of using pipelines over freestyle jobs.

3.2. Declarative Pipelines

- Writing a `Jenkinsfile`.
- Stages and steps: `pipeline`, `agent`, `stages`, `steps`.
- Parallel stages for concurrent execution.

3.3. Scripted Pipelines

- Using Groovy syntax for complex workflows.
- Dynamic stages and advanced scripting.

---

4. Integrating Jenkins with AWS

4.1. Setting Up Jenkins on AWS

- Running Jenkins on EC2 instances.
- Setting up auto-scaling Jenkins agents on AWS using the EC2 plugin.

- Storing Jenkins backups on S3.

4.2. CI/CD Integration

- **AWS CodeDeploy Integration**
    - Automating deployments to EC2 instances, or on-premise servers.
- **Terraform**
    - Managing AWS infrastructure from Jenkins pipelines.
    - Automating infrastructure deployment.

---

5. Job Management

5.1. Creating and Configuring Jobs

- Freestyle jobs: configuring build steps, triggers, and post-build actions.
- Pipeline jobs: linking to `Jenkinsfile` in Git repositories.
- Multibranch pipelines for managing multiple Git branches.

5.2. Triggers

- **Build Triggers**
    - Poll SCM for changes.
    - Webhooks for real-time builds (e.g., GitHub or Bitbucket triggers).
    - Scheduled builds using CRON syntax.

---

6. Continuous Integration

6.1. Source Code Management

- Configuring Git repositories in Jenkins.
- Integrating with GitHub, GitLab, or Bitbucket.
- Handling branches and tags in SCM.

6.2. Testing Automation

- Running unit and integration tests in Jenkins pipelines.
- Using testing tools like JUnit, Selenium, or Cypress.
- Publishing test results and code coverage reports.

7. Continuous Delivery/Deployment

7.1. Artifact Management

- Storing artifacts in Jenkins workspace.
- Publishing artifacts to S3, Nexus, or Artifactory.

7.2. Deployment Automation

- Deploying applications to AWS ECS, EKS, or Lambda.
- Rolling updates and canary deployments using Jenkins pipelines.
- Integrating with Docker for containerized deployments.

8. Jenkins and Docker

8.1. Running Jenkins in Docker

- Installing and running Jenkins in a Docker container.
- Managing Jenkins data persistence with volumes.
- Networking and scaling Jenkins with Docker Compose.

8.2. Docker Integration

- Using Docker plugins to build and push images.
- Automating container builds for Kubernetes deployment.

9. Jenkins Security

9.1. Security Best Practices

- Enabling HTTPS for Jenkins.
- Configuring authentication with LDAP or Active Directory.
- Managing user roles and permissions.

9.2. Credential Management

- Storing and using sensitive data securely.
- Using Jenkins credentials in pipelines (`withCredentials` block).

---

10. Monitoring and Maintenance

10.1. Monitoring Jenkins

- Monitoring Jenkins logs and usage metrics.
- Integrating with monitoring tools (e.g., Prometheus, Grafana).
- Using the Jenkins Monitoring plugin.

10.2. Backup and Restore

- Automating Jenkins backups to AWS S3.
- Restoring Jenkins from backup files.

---

11. Advanced Jenkins Topics

11.1. Distributed Builds

- Setting up Jenkins agents for distributed builds.
- Configuring agents on EC2 or Kubernetes.

---

12. Troubleshooting Jenkins

- Debugging pipeline failures.
- Resolving plugin compatibility issues.

- Scaling Jenkins for high availability.

---

1. Introduction to Docker

- **What is Docker?**
  - Overview of Docker and containerization
  - Differences between Virtual Machines (VMs) and Containers
- **Docker Components**
  - Docker Engine
  - Docker Images
  - Docker Containers
  - Docker Registries (Docker Hub, private registries)

2. Setting Up Docker

- **Installing Docker**
  - Installing Docker on Windows, Linux, and macOS
  - Verifying installation and checking Docker version
- **Basic Docker Commands**
  - `docker --version`
  - `docker info`
  - `docker help`

3. Docker Images and Containers

- **Working with Docker Images**
  - What is a Docker Image?
  - Creating Docker Images from Dockerfiles
  - Docker Hub and Pulling Images

    - Pushing Images to Docker Hub
    - `docker pull`, `docker push`, `docker build`, `docker images`
- **Working with Docker Containers**
    - What is a Container?
    - Creating and Running Containers (`docker run`, `docker create`, `docker start`)
    - Interacting with Containers (`docker exec`, `docker attach`, `docker logs`)
    - Stopping and Removing Containers (`docker stop`, `docker rm`, `docker ps`)

4. Dockerfile and Docker Image Building

- **Introduction to Dockerfile**
    - Structure of Dockerfile
    - Dockerfile Instructions: `FROM`, `RUN`, `CMD`, `COPY`, `ADD`, `EXPOSE`, `WORKDIR`, `ENTRYPOINT`
- **Building Custom Docker Images**
    - Writing Dockerfiles for custom applications
    - Caching and optimization strategies for building images
    - Multi-stage builds
- **Tagging and Versioning Images**
    - Understanding tags (`latest`, specific tags)
    - Versioning images with semantic versioning

5. Docker Networking

- **Docker Networking Basics**
    - What is Docker Networking?
    - Network types: bridge, host, overlay, and none
    - `docker network ls`, `docker network inspect`, `docker network create`
- **Port Mapping**

- Exposing Ports (`p` option in `docker run`)
- Linking Containers and Networking
- Connecting multiple containers via networks

6. Docker Volumes and Persistent Storage

- **Understanding Volumes**
  - What are Volumes in Docker?
  - Volume vs Bind Mounts
  - Creating and Using Volumes (`docker volume create`, `docker volume ls`, `docker volume inspect`)
- **Data Persistence in Containers**
  - Storing data outside containers
  - Sharing data between containers using volumes

7. Docker Compose

- **Introduction to Docker Compose**
  - What is Docker Compose?
  - Benefits of using Docker Compose
  - Writing `docker-compose.yml`
- **Managing Multi-Container Applications**
  - Starting, stopping, and managing services (`docker-compose up`, `docker-compose down`)
  - Defining multiple services in Compose files (web, db, etc.)
  - Environment variables and configuration
- **Docker Compose Networking**
  - Defining custom networks in `docker-compose.yml`
  - Linking containers across services

8. Docker Swarm and Orchestration

- **Introduction to Docker Swarm**
    - What is Docker Swarm?

9. Advanced Docker Concepts

- **Docker Registry**
    - What is Docker Registry?
    - Using Docker Hub
    - Setting up a private Docker Registry
    - Pushing and Pulling images from private registries
- **Docker Security**
    - Managing Docker user privileges
    - Understanding security risks with Docker containers
    - Docker security best practices (e.g., running containers as non-root users, image scanning)

10. Docker and CI/CD Integration

- **Integrating Docker with CI/CD Pipelines**
    - Using Docker in Jenkins
    - Building Docker Images in a CI/CD pipeline
    - Deploying applications in containers via CI/CD
- **Docker in Testing and Development**
    - Using Docker to create test environments
    - Running unit tests in containers
    - Using Docker Compose in testing workflows

11. Troubleshooting Docker

- **Common Docker Issues**

- Diagnosing container startup issues
- Logs, events, and troubleshooting commands
- Network troubleshooting with Docker

- **Docker Metrics and Monitoring**
  - Monitoring Docker containers with `docker stats`
  - Using third-party tools like Prometheus, Grafana, and cAdvisor
  - Logging with Docker (e.g., using Fluentd, ELK Stack)

12. Best Practices

- **Docker Image Optimization**
  - Writing minimal and efficient Dockerfiles
  - Using smaller base images (e.g., Alpine Linux)
  - Reducing image layers
- **Managing Docker at Scale**
  - Handling container orchestration with Kubernetes (optional)
  - Managing container lifecycle at scale
  - Using Docker in cloud environments (AWS, Azure, Google Cloud)

---

1. Introduction to Kubernetes

- **What is Kubernetes?**
    - Overview of Kubernetes and container orchestration
    - Benefits of Kubernetes for deploying containerized applications
    - Kubernetes vs Docker Swarm: Key differences
- **Kubernetes Components**
    - Nodes, Pods, Containers
    - Control Plane vs Worker Nodes
    - Overview of Kubernetes Architecture
- **Kubernetes Cluster Structure**
    - Master Node (API Server, Controller Manager, Scheduler)
    - Worker Nodes (Kubelet, Kube Proxy, Container Runtime)
    - Understanding Namespaces, Labels, and Annotations

---

2. Setting Up Kubernetes

- **Installing Kubernetes**
    - Installing Kubernetes on different environments (Windows, macOS, Linux)
    - Minikube for local development
    - Installing Kubernetes with kubeadm (for multi-node clusters)
    - Installing Kubernetes with managed services (EKS, GKE, AKS)
- **Kubernetes CLI - kubectl**
    - Installing and configuring kubectl
    - Basic kubectl commands: `kubectl version`, `kubectl get`, `kubectl describe`
    - Interacting with a Kubernetes cluster using kubectl

- Using `kubectl` for troubleshooting and debugging

---

3. Kubernetes Pods and Containers

- **Understanding Pods**
  - What are Pods and how do they relate to containers?
  - Pod lifecycle (Pending, Running, Succeeded, Failed)
  - Multi-container Pods and use cases
  - Pod networking: Container communication inside a Pod
- **Running Containers in Pods**
  - Specifying containers within Pods
  - Managing environment variables and resources (CPU, Memory)
  - Using ConfigMaps and Secrets in Pods
  - Pods vs Containers: Understanding the difference

---

4. Kubernetes Services and Networking

- **Kubernetes Networking Basics**
  - Cluster Networking overview (CNI)
  - Pod-to-Pod communication
  - Service discovery and DNS
- **Kubernetes Services**
  - What are Services in Kubernetes?
  - Types of Services: ClusterIP, NodePort, LoadBalancer, ExternalName
  - Exposing applications via services (`kubectl expose`)
  - Endpoints and routing traffic
- **Ingress Controllers and Resources**
  - Introduction to Ingress

- Setting up Ingress controllers
- Defining Ingress Resources and routing HTTP traffic

---

5. Kubernetes Deployments

- **What are Deployments?**
  - Understanding the role of Deployments in Kubernetes
  - Creating Deployments using `kubectl apply -f`
  - Rolling updates and Rollbacks in Deployments
  - Scaling Deployments (horizontal scaling)
- **ReplicaSets and Pods**
  - Role of ReplicaSets in Kubernetes
  - Creating and managing ReplicaSets
  - Scaling Pods via ReplicaSets and Deployments

---

6. Kubernetes Configurations and Secrets

- **ConfigMaps**
  - Storing and using configuration data with ConfigMaps
  - Using ConfigMaps in Pods as environment variables, volumes, or command-line arguments
- **Secrets**
  - Managing sensitive information using Secrets
  - Using Secrets in Pods for database credentials or API keys
  - Encoding and decoding secrets
- **Environment Variables and Resource Limits**
  - Setting environment variables for Pods and containers
  - Managing resource requests and limits (CPU, memory)

---

7. Kubernetes Volumes and Persistent Storage

- **Kubernetes Volumes**
  - Understanding Kubernetes Volumes and their lifecycle
  - Different types of Volumes: `emptyDir`, `hostPath`, `nfs`, etc.
  - Sharing data between containers in Pods
- **Persistent Volumes and Persistent Volume Claims**
  - Understanding Persistent Volumes (PVs) and Persistent Volume Claims (PVCs)
  - Setting up dynamic provisioning for persistent storage
  - Access Modes: ReadWriteOnce, ReadOnlyMany, ReadWriteMany
- **Storage Classes**
  - What are Storage Classes?
  - Setting up and using different storage classes for dynamic provisioning

---

8. Advanced Scheduling and Resource Management

- **Scheduling in Kubernetes**
  - How Kubernetes schedules Pods to nodes
  - Affinity, Anti-Affinity, Taints, and Tolerations
  - Resource Requests and Limits for efficient scheduling
- **Pod Disruption Budgets**
  - Managing availability during updates or node failures

---

9. Kubernetes Security

- **Role-Based Access Control (RBAC)**
  - Understanding RBAC in Kubernetes
  - Defining Roles, RoleBindings, ClusterRoles, and ClusterRoleBindings

    - Securing API access and permissions

---

10. Kubernetes Monitoring and Logging

- **Monitoring Kubernetes**
    - Introduction to Kubernetes monitoring tools (Prometheus, Grafana, etc.)
    - Setting up monitoring for Pods, Nodes, and Services
    - Collecting and visualizing metrics with Grafana
- **Logging in Kubernetes**
    - Centralized logging with Elasticsearch, Fluentd, and Kibana (EFK stack)
    - Using `kubectl logs` for accessing Pod logs
    - Setting up logging solutions for multi-cluster environments

---

11. Kubernetes High Availability and Scaling

- **High Availability in Kubernetes**
    - Understanding Kubernetes High Availability (HA) architecture
    - Setting up a highly available Kubernetes cluster (multi-master nodes)
- **Scaling Kubernetes Applications**
    - Horizontal Pod Autoscaling (HPA)
    - Vertical Pod Autoscaling (VPA)
    - Cluster Autoscaler and Auto-scaling nodes in cloud environments
    - Scaling Deployments and StatefulSets automatically

---

12. Kubernetes State Management

- **StatefulSets**
    - What are StatefulSets and why are they important?
    - Use cases of StatefulSets (databases, queues, etc.)

    - StatefulSet management and scaling
    - Persistent Storage and StatefulSets
- **DaemonSets**
    - What is a DaemonSet?
    - Running a DaemonSet to ensure a copy of a Pod runs on each node
    - Use cases for DaemonSets (monitoring agents, log collectors)

---

13. Kubernetes CI/CD and Automation

- **Integrating Kubernetes with CI/CD Pipelines**
    - Using Kubernetes for automated application deployment
    - Integrating with CI/CD tools (Jenkins, GitLab CI, CircleCI, etc.)
    - Continuous delivery with Kubernetes and Helm
- **Helm Charts**
    - What is Helm and why is it used?
    - Installing and managing Helm on Kubernetes
    - Using Helm to deploy applications and manage configurations
    - Creating and maintaining Helm charts for application deployments

---

14. Kubernetes Best Practices

- **Kubernetes Best Practices for Developers**
    - Creating efficient and scalable Kubernetes workloads
    - Organizing and structuring Kubernetes configurations (YAML files)
- **Kubernetes Best Practices for Operations**
    - Cluster monitoring, scaling, and management
    - Backup and disaster recovery strategies
    - Handling rolling updates, blue-green deployments, and canary releases

---

Advanced Topics

- **Kubernetes and Service Mesh**
    - Introduction to Service Mesh (e.g., Istio)
    - Managing microservices with Service Mesh
    - Traffic management and observability in Kubernetes with Service Mesh
- **Kubernetes on Cloud Platforms**
    - Deploying and managing Kubernetes on AWS (EKS), GCP (GKE), and Azure (AKS)
    - Hybrid and multi-cloud Kubernetes management

---

1. AWS Identity and Access Management (IAM)

- **Overview of IAM**
    - What is IAM and its role in managing access to AWS resources
    - Users, Groups, Roles, and Policies in IAM
    - IAM best practices for managing permissions securely
- **IAM Users**
    - Creating IAM users and assigning permissions
    - Configuring MFA (Multi-Factor Authentication) for IAM users
    - Best practices for managing IAM user access (least privilege)
- **IAM Groups**
    - Creating IAM groups and assigning users to groups
    - Applying IAM policies to groups for centralized permission management
    - Managing group permissions and roles
- **IAM Roles**
    - What are IAM roles and how are they different from users
    - Creating IAM roles for EC2 instances and Lambda functions
    - Attaching policies to roles and using them with AWS services
- **IAM Policies**
    - Understanding IAM policy syntax (JSON format)
    - AWS Managed Policies vs Custom Policies
    - Granting permissions to resources using policies (IAM Policy Simulator)
- **IAM Access Analyzer**

    - Using IAM Access Analyzer to identify resources shared with external entities

---

2. AWS Key Management Service (KMS)

- **Overview of KMS**
    - What is AWS KMS and how it helps with managing encryption keys
    - Overview of symmetric vs asymmetric encryption
    - Key management best practices
- **Creating and Managing KMS Keys**
    - Creating customer-managed keys (CMKs)
    - Key policies and access control for KMS keys
    - Rotating encryption keys and managing key lifecycle
- **Using KMS with AWS Services**
    - Encrypting data using KMS with S3, EBS, RDS, DynamoDB, etc.
    - Using KMS for envelope encryption
    - Managing encryption of data at rest and in transit
- **Key Policies and IAM Integration**
    - Managing key permissions using IAM and KMS key policies
    - Configuring key access for different AWS services and users

---

3. Amazon SNS (Simple Notification Service)

- **Overview of SNS**
    - What is Amazon SNS and its use cases (message delivery, pub/sub, alerts)
    - Different protocols supported by SNS (SMS, Email, HTTP/S, Lambda, SQS)
- **Creating and Managing Topics**
    - Creating SNS topics for publishing messages
    - Managing topic policies and permissions

- **Subscribing to Topics**
  - Subscribing endpoints to SNS topics (Email, SQS, Lambda, etc.)
  - Verifying subscription and managing delivery protocols
- **Publishing Messages to Topics**
  - Publishing messages to SNS topics using AWS SDKs and the console
  - Using SNS to trigger Lambda functions, workflows, or send notifications
- **SNS Use Cases**
  - Application monitoring and alerting
  - Mobile push notifications
  - Integrating SNS with CloudWatch and CloudTrail for event-driven workflows

---

4. Amazon EC2 (Elastic Compute Cloud)

- **Overview of EC2**
  - What is EC2 and its purpose in AWS
  - EC2 Instance types, families, and use cases
  - EC2 Pricing models: On-Demand, Reserved, Spot, and Savings Plans
- **EC2 Instance Lifecycle**
  - Launching, stopping, and terminating EC2 instances
  - Instance states: running, pending, stopped, terminated
  - EC2 Instance Configuration and AMIs (Amazon Machine Images)
- **Security and Access**
  - Key pairs for SSH access
  - EC2 security groups and network access control
  - IAM roles for EC2 instances
- **Elastic IPs and Public IPs**

- What are Elastic IPs and how to assign them to EC2 instances
- **Auto Scaling for EC2 Instances**

---

5. Amazon EFS (Elastic File System)

- **Overview of EFS**
    - What is EFS and its use cases (shared storage, file systems)
    - Benefits of EFS for scalable file storage
- **Creating and Managing EFS**
    - Mounting and unmounting EFS on EC2 instances
    - NFS-based access and mounting EFS on Linux/Windows instances
- **Performance and Scaling**
    - Performance modes and throughput modes
    - Scaling EFS automatically with workloads

---

6. Amazon EBS (Elastic Block Store)

- **Overview of EBS**
    - What is EBS and its use cases (persistent block storage)
    - EBS volume types: General Purpose SSD (gp3), Provisioned IOPS SSD (io2), Magnetic, etc.
- **EBS Snapshots and Backups**
    - Creating, restoring, and sharing EBS snapshots
    - Automating backups with AWS Backup
- **EBS Performance and Encryption**
    - EBS performance considerations and monitoring (IOPS, throughput, latency)
    - Enabling encryption for EBS volumes
- **Attaching and Detaching EBS Volumes**

- Attaching EBS volumes to EC2 instances and mounting them

---

7. Amazon S3 (Simple Storage Service)

- **Overview of S3**
  - What is Amazon S3 and its use cases (object storage)
  - S3 Storage Classes: Standard, Intelligent-Tiering, Glacier, etc.
- **Creating and Managing Buckets**
  - Bucket policies, permissions, and encryption
  - S3 Versioning and lifecycle policies
- **Access Control for S3**
  - IAM policies, bucket policies, and ACLs (Access Control Lists)
  - Signed URLs and pre-signed URLs
- **S3 Data Management and Security**
  - Data encryption at rest and in transit
  - Event notifications and Lambda integrations

---

8. Amazon VPC (Virtual Private Cloud)

- **Overview of VPC**
  - What is VPC and its components (subnets, route tables, internet gateway)
  - VPC CIDR block, IPv4, IPv6 addressing
  - VPC Peering and Transit Gateway
- **Subnets and Route Tables**
  - Creating public and private subnets
  - Configuring route tables and routing traffic between subnets
- **NAT Gateway and Internet Gateway**
  - Configuring Internet Gateway and NAT Gateway for private subnet internet access

- **Security with VPC**
  - Security groups and NACLs (Network Access Control Lists)

---

9. Amazon ELB (Elastic Load Balancing)

- **Overview of ELB**
  - What is Elastic Load Balancer and its types (Application Load Balancer, Network Load Balancer, Classic Load Balancer)
  - Use cases for each type of load balancer
- **Configuring and Managing Load Balancers**
  - Setting up listeners and target groups
  - Registering EC2 instances with load balancers
  - SSL termination with ELB
- **Health Checks and Auto Scaling**
  - Health check configuration for target instances
  - Integration with Auto Scaling Groups

---

10. Amazon ASG (Auto Scaling Groups)

- **Overview of ASG**
  - What is Auto Scaling and its purpose in AWS
  - Launch configurations and launch templates
- **Auto Scaling Policies**
  - Configuring scaling policies (CPU utilization, custom metrics)
  - Scaling based on load or time of day
- **Scaling Strategies**
  - Horizontal scaling vs vertical scaling
  - Scaling EC2 instances in and out based on demand

---

11. AWS GuardDuty

- **Overview of GuardDuty**
    - What is Amazon GuardDuty and its role in threat detection
    - Types of threats GuardDuty can detect (malicious activity, anomalous behavior)
- **Configuring GuardDuty**
    - Enabling GuardDuty across AWS accounts
    - GuardDuty findings and alerts
- **Integration with CloudWatch and CloudTrail**
    - Automating responses to GuardDuty findings

---

12. AWS Shield

- **Overview of AWS Shield**
    - What is AWS Shield and its purpose in DDoS protection
    - AWS Shield Standard vs AWS Shield Advanced
- **Protecting Resources with Shield**
    - Enabling Shield for CloudFront distributions and ELBs
    - Responding to DDoS events with Shield Advanced

---

13. AWS WAF (Web Application Firewall)

- **Overview of WAF**
    - What is AWS WAF and how it protects web applications
    - WAF rules, conditions, and actions
- **Configuring WAF**
    - Creating custom WAF rules for blocking malicious traffic
    - AWS Managed Rules for common threats
    - Integrating WAF with CloudFront and ALB

---

14. Security Groups

- **Overview of Security Groups**
    - What is a Security Group and how it controls access to EC2 instances
    - Inbound and outbound rules in security groups
- **Managing Security Groups**
    - Associating Security Groups with EC2 instances, ELBs, etc.
    - Best practices for managing and auditing Security Groups

---

15. AWS Inspector

- **Overview of AWS Inspector**
    - What is AWS Inspector and its use in security assessment
    - Types of assessments: Network Reachability, Host Assessment, and Custom Assessment
- **Running Assessments**
    - Configuring and running security assessments
    - Viewing and acting upon findings

---

16. AWS CloudTrail

- **Overview of CloudTrail**
    - What is AWS CloudTrail and its purpose in monitoring AWS API calls
    - Tracking user activity and changes across AWS resources
- **Configuring CloudTrail**
    - Creating trails, enabling CloudTrail across accounts
    - Storing CloudTrail logs in S3 and integrating with CloudWatch Logs

---

17. AWS CloudWatch

- **Overview of CloudWatch**
    - What is AWS CloudWatch and its use in monitoring and logging
    - CloudWatch Metrics, Alarms, Logs, and Events
- **Setting up CloudWatch Monitoring**
    - Creating custom CloudWatch Metrics and Alarms
    - Using CloudWatch Logs to aggregate and monitor log data
    - Integration with Lambda, SNS, and other AWS services

---

18. Amazon Route 53

- **Overview of Route 53**
    - What is Route 53 and its purpose in DNS management
    - Registering domain names and configuring hosted zones
- **Route 53 Routing Policies**
    - Routing traffic based on latency, geolocation, and weighted policies
    - Using health checks and failover routing
- **Integration with Other AWS Services**
    - Integrating Route 53 with S3, CloudFront, ELB, and more

---

19. Amazon EKS (Elastic Kubernetes Service)

- **Overview of EKS**
    - What is Amazon EKS and its purpose in managing Kubernetes clusters
    - Kubernetes architecture and components
- **Setting up EKS Clusters**
    - Creating and managing EKS clusters
    - Integrating EKS with EC2 instances
- **EKS Networking and Security**

    - Setting up VPC, subnets, and IAM roles for EKS
    - Securing EKS with RBAC, Service Accounts, and Network Policies

---

20. Amazon ECR (Elastic Container Registry)

- **Overview of ECR**
    - What is Amazon ECR and its purpose in storing Docker container images
    - Creating and managing ECR repositories
- **Pushing and Pulling Docker Images**
    - Pushing Docker images to ECR
    - Integrating ECR with ECS, EKS, or other container services
- **ECR Security**
    - IAM permissions for ECR access
    - Enabling image scanning for vulnerabilities

---

21. Amazon RDS (Relational Database Service)

- **Overview of RDS**
    - What is Amazon RDS and its supported database engines (MySQL, PostgreSQL, SQL Server, Oracle, MariaDB, Aurora)
    - Benefits of using RDS over self-managed databases
- **Creating and Managing RDS Instances**
    - Launching and configuring RDS instances
    - RDS backups, snapshots, and Multi-AZ deployments
- **Scaling and Performance Optimization**
    - Read replicas and automatic scaling with RDS
    - Managing database security with IAM and VPC security groups

---

22. Amazon DynamoDB

- **Overview of DynamoDB**
  - What is DynamoDB and its use cases (NoSQL database)
  - Differences between DynamoDB and RDS

---

1. Introduction to Terraform

- **What is Terraform?**
    - Overview of Infrastructure as Code (IaC)
    - Benefits of using Terraform for managing cloud infrastructure
    - Understanding the Terraform workflow (Write, Plan, Apply, Destroy)
- **Terraform vs Other IaC Tools**
    - Terraform vs CloudFormation
    - Terraform vs Ansible
    - Terraform vs Puppet and Chef
- **Installing Terraform**
    - Installing Terraform on various platforms (Linux, macOS, Windows)
    - Verifying installation (`terraform version`)
    - Setting up Terraform CLI and environment

2. Terraform Basics

- **Terraform Configuration Files**
    - Overview of Terraform configuration files (`.tf` files)
    - Understanding the basic structure of a Terraform configuration (Provider, Resources, Outputs)
    - Basic configuration syntax and language
    - Using HCL (HashiCorp Configuration Language)
- **Terraform Providers**
    - Introduction to Providers in Terraform
    - Understanding Provider configuration (e.g., AWS, Azure, Google Cloud)

- Setting up and configuring a Provider (e.g., AWS Access Keys)

- **Resources in Terraform**
    - Defining resources (e.g., EC2, S3, VPC, subnets)
    - Creating and managing resources with Terraform
    - Modifying existing resources using Terraform (`terraform apply`)

---

3. Terraform Variables and Outputs

- **Using Variables**
    - Declaring variables in Terraform (`variable` block)
    - Types of variables (strings, integers, lists, maps)
    - Default values and variable validation
    - Passing variable values (`var`, `var-file`)
- **Output Values**
    - Defining and using output variables (`output` block)
    - Outputting resource attributes (e.g., public IP, instance ID)
    - Using outputs to pass data between modules and configurations

---

4. Terraform State and Remote Backends

- **Understanding Terraform State**
    - What is Terraform state and why is it important?
    - The role of the `.tfstate` file
    - Local state vs remote state
    - Managing and securing Terraform state files
- **Working with Remote Backends**
    - Setting up remote backends (e.g., AWS S3, Azure Blob Storage)
    - Configuring remote state with versioning and locking

    - Using backend configuration for state storage
    - Benefits of using remote backends (team collaboration, state consistency)

---

5. Terraform Modules

- **Introduction to Modules**
    - What are modules in Terraform?
    - Creating and organizing reusable Terraform modules
    - Using modules from the Terraform Registry
    - Importing and using local and remote modules
- **Module Inputs and Outputs**
    - Defining and passing variables to modules
    - Output values from modules
    - Best practices for organizing and structuring modules
- **Terraform Module Structure**
    - Folder structure for a Terraform module
    - Best practices for writing reusable and maintainable modules

---

6. Terraform Provisioners and Taints

- **Provisioners**
    - What are provisioners and how are they used?
    - Types of provisioners: `local-exec`, `remote-exec`, `file`
    - When to use provisioners vs when to avoid them
    - Example: Using `remote-exec` for configuring an EC2 instance
- **Tainting Resources**
    - What is tainting in Terraform?
    - Manually tainting resources using `terraform taint`

    - Understanding the impact of tainting and how it affects resource lifecycle

---

7. Terraform Cloud and Workspaces

- **Introduction to Terraform Cloud**
    - What is Terraform Cloud and how it differs from local execution?
    - Setting up a Terraform Cloud account and organization
    - Using Terraform Cloud for remote runs, collaboration, and team workflows
- **Terraform Workspaces**
    - What are workspaces in Terraform?
    - Creating and using workspaces for managing different environments (development, staging, production)
    - Switching between workspaces and managing state files across environments

---

---

8. Advanced Terraform Features

- **Terraform Graphs**
    - Visualizing infrastructure relationships with `terraform graph`
    - Understanding resource dependencies in large infrastructures
- **Working with Data Sources**
    - Using Terraform data sources to fetch data from external systems
    - Example: Fetching existing resources from AWS (e.g., VPCs, AMIs)
- **Terraform CLI and Automation**
    - Automating Terraform runs using CI/CD tools (e.g., Jenkins
    - Integrating Terraform into automated workflows for provisioning
    - Using Terraform with pipelines for continuous deployment and infrastructure management

---

Advanced Topics

- **Terraform Enterprise**
  - Introduction to Terraform Enterprise and its benefits over Terraform Cloud
  - Managing Terraform workspaces, policies, and teams in Terraform Enterprise
- **Integrating Terraform with Service Meshes**
  - Managing service mesh infrastructure (e.g., Istio) with Terraform
  - Using Terraform for service mesh configurations and deployments

1. Introduction to Ansible

- **What is Ansible?**
  - Overview of Ansible and its role in automation
  - Configuration management vs orchestration vs provisioning
  - Benefits of using Ansible for automation
- **Ansible Architecture**
  - Ansible's agentless architecture (SSH-based communication)
  - Inventory (static and dynamic)
  - Control Node vs Managed Nodes
  - How Ansible works (Modules, Playbooks, Tasks, Variables, etc.)
- **Setting Up Ansible**
  - Installing Ansible on various platforms (Linux, macOS, Windows)
  - Verifying installation (`ansible --version`)
  - Configuring Ansible (ansible.cfg, inventory)
  - Understanding the inventory structure (INI format, YAML format)

---

2. Ansible Basics

- **Understanding Inventory Files**
  - Static inventory file format (INI-style)
  - Dynamic inventory and writing custom scripts
  - Organizing hosts in groups and variables
- **Ad-Hoc Commands**
  - Running ad-hoc commands using `ansible` command
  - Common ad-hoc command examples: `ping`, `shell`, `copy`, `service`

  - Debugging with `ansible -m debug`

---

3. Ansible Playbooks

- **Introduction to Playbooks**
  - What is a Playbook? (YAML format)
  - Understanding Plays, Tasks, and Hosts
  - Writing and Running Playbooks (`ansible-playbook`)
  - Basic structure of a Playbook (hosts, tasks, vars, handlers)
- **Tasks and Modules**
  - Using Ansible modules in Playbooks
  - Common modules: `command`, `shell`, `copy`, `template`, `file`, `package`, `service`
  - Task execution flow (serial, parallel, retries)
- **Variables and Facts**
  - Defining variables in Playbooks (`vars`, `vars_files`)
  - Using built-in facts and custom facts
  - Accessing variables within Playbooks (`{{ variable_name }}`)
  - Registering variables from tasks and using them later
- **Conditionals and Loops**
  - Using `when` for conditional execution of tasks
  - Using loops (`with_items`, `loop`, `with_dict`)
  - Handling loops with `loop_control` (e.g., `index`, `item`)

---

4. Ansible Templates and Files

- **Using Jinja2 Templates**
  - What is Jinja2 and how it integrates with Ansible
  - Creating and using template files (`.j2` files)

  - Substituting variables and logic in templates
- **File Management**
  - Managing files with the `copy`, `template`, and `fetch` modules
  - Working with directories and permissions (`file`, `stat`, `acl`)

---

5. Ansible Handlers and Notifications

- **Handlers**
  - What are Handlers? (Special tasks that only run when notified)
  - Creating Handlers in Playbooks
  - Notifying Handlers (`notify`, `triggered`)
  - Use cases for handlers (e.g., restarting a service after a configuration change)

---

6. Ansible Vault and Security

- **Using Ansible Vault**
  - Introduction to Ansible Vault for encrypting sensitive data
  - Creating and editing encrypted files with `ansible-vault`
  - Encrypting and decrypting Playbooks and variable files
  - Using Vault variables in Playbooks

---

7. Ansible Advanced Features

- **Ansible Facts and Dynamic Variables**
  - Using system facts to gather information from managed nodes
  - Writing custom dynamic facts and using them
  - Using `gather_facts` in Playbooks

- **Ansible Lookup Plugins**
    - Introduction to Lookup Plugins
    - Common Lookup Plugins: `file`, `env`, `password`, `pipe`, `query`
    - Using Lookups in Playbooks to fetch data or files
- **Ansible Filters**
    - Introduction to Filters in Ansible (Jinja2 Filters)
    - Common Filters: `default`, `selectattr`, `map`, `json_query`
    - Using filters to modify and format data in Playbooks

---

8. Ansible Error Handling and Debugging

- **Error Handling in Ansible**
- Conditional execution with `when` and `when not`
- **Debugging Playbooks**
    - Debugging tasks with the `debug` module
    - Verbose output (`v`, `vv`, `vvv`)
    - Checking Playbook syntax using `ansible-playbook --syntax-check`
