DevOps Syllabus

RHCSA

1) Understanding and Using Essential Tools:
   - Accessing the command line.
   - Managing files from the command line.
   - Using input-output redirection.
   - Managing text files.
2) Operating Running Systems:
   - Booting into different run levels.
   - Identifying processes and controlling them.
   - Managing system services (start, stop, enable, disable).
   - Configuring systems to boot automatically.
3) Configuring Local Storage:
   - Partitioning and formatting storage devices.
   - Creating and managing filesystems.
   - Mounting and unmounting filesystems.
   - Extending logical volumes.
4) Managing Users and Groups:
   - Creating, modifying, and deleting users.
   - Creating, modifying, and deleting groups.
   - Managing user accounts and group memberships.
5) Configuring Network Services:
   - Configuring network interfaces.
   - Configuring hostname resolution (DNS).
   - Configuring network time synchronization (NTP).
   - Configuring basic firewall settings (firewalld).
6) Managing Security:
   - Configuring firewall settings.
   - Configuring SELinux settings.
   - Setting file permissions and ownership.
   - Creating and using access control lists (ACLs).
7) Working with Containers:
   - Understanding container technology.
   - Managing containers with Podman or Docker.
8) Automation with Ansible:
   - Understanding Ansible concepts and architecture.
   - Writing and using Ansible playbooks for system configuration.

Ansible

1) Introduction to Ansible:
   - Understanding configuration management and automation.
   - Introduction to Ansible: architecture, components, and terminology.
   - Installing Ansible and configuring the control node.
2) Ansible Basics:
   - Ansible inventory: defining hosts and groups.
   - Writing and executing ad-hoc commands.
   - Working with playbooks: YAML syntax, structure, and best practices.
   - Ansible roles: organizing and reusing playbooks.
3) Managing Inventory and Variables:
   - Dynamic inventory: integrating with cloud providers, external sources, and custom scripts.
   - Host and group variables: defining variables for specific hosts or groups.
   - Using facts: gathering information about managed hosts.
4) Working with Modules:
   - Understanding Ansible modules: core, custom, and community modules.
   - Commonly used modules: file, package, service, user, template, command, shell, and copy.
   - Using modules in playbooks to perform tasks on managed hosts.
5) Managing Files and Directories:
   - Working with files and directories: creating, copying, moving, and deleting files.
   - Managing file permissions and ownership.
   - Using templates to dynamically generate configuration files.
6) Managing Packages and Software:
   - Installing, updating, and removing packages.
   - Managing software repositories.
   - Managing application services: starting, stopping, and restarting services.
7) Managing Users and Groups:
   - Managing user accounts: creating, modifying, and deleting users.
   - Managing user groups: creating, modifying, and deleting groups.
   - Managing SSH keys and authentication.
8) Working with Roles:
   - Understanding Ansible roles: structure, tasks, handlers, variables, and templates.
   - Creating and using roles to organize and modularize playbooks.
9) Ansible Vault:

- Securing sensitive data with Ansible Vault.
- Encrypting and decrypting files using Vault.
- Integrating Vault with playbooks and roles.

10) Advanced Ansible Features:
    - Ansible loops and conditionals: iterating over tasks and applying conditional logic.
    - Using Ansible tags: selectively executing tasks within playbooks.
    - Ansible Galaxy: discovering and using pre-built roles from the Ansible community.
11) Best Practices and Troubleshooting:
    - Ansible best practices: organizing playbooks, writing clean and maintainable code.
    - Troubleshooting common issues and errors in Ansible automation.

AWS

1) AWS Fundamentals:
    - Understanding the AWS Global Infrastructure: regions, availability zones, edge locations.
    - AWS Services Overview: Compute, Storage, Database, Networking, Security, and Management services.
2) Identity and Access Management (IAM):
    - IAM Users, Groups, Roles, and Policies.
    - IAM best practices and security principles.
3) Compute Services:
    - Amazon Elastic Compute Cloud (EC2): Instance types, AMIs, Launch configurations, Auto Scaling, and Elastic Load Balancing (ELB).
    - AWS Lambda: Serverless compute service for running code without provisioning or managing servers.
    - AWS Elastic Beanstalk: Platform as a Service (PaaS) for deploying and managing web applications.
4) Storage Services:
    - Amazon Simple Storage Service (S3): Object storage service for storing and retrieving data.
    - Amazon Elastic Block Store (EBS): Block-level storage volumes for EC2 instances.
    - Amazon Elastic File System (EFS): Managed file storage service for EC2 instances.
    - Amazon Glacier: Low-cost storage service for data archiving and long-term backup.
5) Database Services:
    - Amazon Relational Database Service (RDS): Managed relational database service for MySQL, PostgreSQL, Oracle, SQL Server, and Aurora.
    - Amazon DynamoDB: Fully managed NoSQL database service.
    - Amazon Redshift: Fully managed data warehouse service.
6) Networking and Content Delivery:

- Amazon Virtual Private Cloud (VPC): Networking service for creating isolated virtual networks.
- AWS Direct Connect: Dedicated network connection between on-premises data centers and AWS.
- Amazon Route 53: Scalable Domain Name System (DNS) web service.
- Amazon CloudFront: Content Delivery Network (CDN) service for delivering static and dynamic web content.

7) Monitoring and Management:
   - Amazon CloudWatch: Monitoring and observability service for AWS resources.
   - AWS CloudTrail: Auditing and logging service for tracking user activity and API usage.
8) Best Practices and Troubleshooting:
   - Troubleshooting common issues and errors in AWS

GIT & GitHub

1) Introduction to Version Control:
   - Understanding the need for version control.
   - Overview of centralized vs. distributed version control systems.
   - Introduction to Git: history, features, and advantages.
2) Getting Started with Git:
   - Installing Git: setup and configuration.
   - Basic Git commands: init, clone, add, commit, status, diff, log, branch, checkout, merge, and reset.
   - Creating and managing Git repositories.
3) Working with Branches and Merging:
   - Understanding branches: creation, deletion, listing.
   - Branching strategies: feature branches, release branches, hotfix branches.
   - Merging changes: fast-forward, recursive, and conflict resolution.
4) Collaborating with Remote Repositories:
   - Introduction to remote repositories.
   - Configuring remote connections: adding, renaming, and removing remotes.
   - Pushing and pulling changes to and from remote repositories.
   - Resolving conflicts during collaboration.
5) GitHub Overview:
   - Introduction to GitHub: features, benefits, and use cases.
   - Creating and managing GitHub repositories.
   - Collaborating on GitHub: forking, cloning, pull requests, code reviews.
   - GitHub workflows: issues, milestones, labels, and projects.
6) Advanced Git Topics:
   - Git configuration: global vs. local settings, aliases.
   - Rewriting history: git rebase, interactive rebase.
   - Cherry-picking and reverting commits.
   - Git hooks: pre-commit, post-commit, pre-push, post-receive.

7) Git Best Practices:
    - Branching strategies and workflows: GitFlow, GitHub Flow.
    - Commit message conventions: format, length, clarity.
    - Code review best practices: etiquette, constructive feedback.
8) GitHub Pages and GitHub Actions:
    - Hosting static websites with GitHub Pages.
    - Automating workflows with GitHub Actions.
    - Creating custom workflows: CI/CD pipelines, automated testing, and deployment.
9) Git Tips and Tricks:
    - Git stash: saving and applying temporary changes.
    - Git bisect: finding the commit that introduced a bug.
    - Git blame: identifying the author of specific lines of code.
    - Gitignore: ignoring files and directories in Git repositories.
10) Git and GitHub Integration:
    - Integrating Git with IDEs and text editors.
    - Using Git with Continuous Integration (CI) tools like Jenkins, Travis CI, or CircleCI

Jenkins

1) Introduction to Jenkins:
    - Overview of continuous integration and continuous delivery (CI/CD).
    - Introduction to Jenkins: history, features, and advantages.
    - Understanding Jenkins architecture: master and agent nodes, executors, and workspaces.
2) Installing and Configuring Jenkins:
    - Installing Jenkins: setup and initial configuration.
    - Configuring Jenkins plugins: installing and managing plugins.
    - Configuring global settings: system configuration, security settings, and notification settings.
3) Creating Jenkins Jobs:
    - Understanding Jenkins jobs: freestyle projects, pipeline projects, and multi-branch pipeline projects.
    - Creating and configuring freestyle projects: defining source code management, build triggers, build steps, and post-build actions.
    - Introduction to Jenkinsfile: defining pipelines as code.
4) Working with Jenkins Pipelines:
    - Understanding Jenkins pipeline: syntax, stages, steps, and directives.
    - Writing and executing declarative pipelines.
    - Writing and executing scripted pipelines.
    - Parameterized builds: passing parameters to Jenkins pipelines.
5) Artifact Management and Deployment:
    - Artifact management with Jenkins: archiving build artifacts.

- Automated deployment pipelines: deploying applications to test, staging, and production environments.
- Integration with deployment tools like Docker, Kubernetes, or Ansible.

6) Monitoring and Reporting:
   - Monitoring Jenkins builds and pipelines: viewing build logs, console output, and build trends.
   - Generating and viewing reports: test reports, code coverage reports, and trend analysis.
7) Scaling and High Availability:
   - Scaling Jenkins: configuring distributed builds with master/slave architecture.
   - High availability and fault tolerance: setting up Jenkins clusters and failover mechanisms.
8) Best Practices and Troubleshooting:
   - Jenkins best practices: pipeline design patterns, efficient job configuration, and optimization.
   - Troubleshooting common issues: diagnosing build failures, pipeline errors, and performance bottlenecks.

Terraform

1) Introduction to Terraform and AWS:
   - Understanding infrastructure as code (IaC) concepts.
   - Introduction to Terraform: features, benefits, and advantages.
   - Overview of AWS services and their role in Terraform infrastructure.
2) Installing and Configuring Terraform for AWS:
   - Installing Terraform: setup and configuration.
   - Configuring AWS credentials and authentication for Terraform.
3) Terraform Configuration Basics:
   - Understanding Terraform configuration files: main.tf, variables.tf, outputs.tf.
   - Terraform providers and resources: AWS provider, EC2 instance, VPC, subnet, etc.
   - Declaring infrastructure using the HashiCorp Configuration Language (HCL).
4) Managing AWS Infrastructure with Terraform:
   - Creating and managing EC2 instances with Terraform.
   - Configuring networking components: VPC, subnet, route table, internet gateway.
   - Managing security groups and access control with Terraform.
5) Terraform State Management:
   - Understanding Terraform state files: local vs. remote state.
   - Terraform state locking: preventing concurrent modifications.
   - Configuring and using remote state backends with AWS S3 and DynamoDB.
6) Advanced Terraform Features for AWS:
   - Using Terraform modules for code reuse and organization.
   - Dynamic infrastructure with Terraform: loops, conditionals, and expressions.
   - Managing multiple environments (dev, stage, prod) with workspaces.
7) Managing AWS Services with Terraform:
   - Provisioning AWS managed services: RDS, S3, DynamoDB, etc.

- Creating AWS Lambda functions and API Gateway endpoints.
- Managing IAM roles, policies, and users with Terraform.

8) Terraform Best Practices for AWS:
   - Terraform project structure and organization.
   - Using Terraform variables, locals, and outputs effectively.
   - Implementing modular and reusable Terraform code with modules.
   - Deployment Strategies with Terraform:
   - Deploying infrastructure changes with Terraform apply.
   - Automating infrastructure deployment with CI/CD pipelines.
   - Blue/green deployments and rollback strategies with Terraform.
9) Security and Compliance with Terraform:
   - Implementing security best practices in Terraform configurations.
   - Using AWS IAM policies for least privilege access.
   - Ensuring compliance with AWS security standards and regulations.
10) Terraform and AWS Integration:
   - Integrating Terraform with AWS services and APIs.
   - Leveraging AWS CloudFormation with Terraform for resource provisioning.
   - Orchestrating AWS infrastructure with Terraform and other AWS services (e.g., CodeDeploy, CodePipeline).

Docker

1) Introduction to Containerization:
   - Understanding containerization and its benefits.
   - Comparison of containers vs. virtual machines.
   - Introduction to Docker: history, features, and advantages.
2) Installing and Configuring Docker:
   - Installing Docker Engine: setup and configuration on different platforms.
   - Configuring Docker environment variables and settings.
   - Docker Editions: Community Edition (CE) and Enterprise Edition (EE).
3) Docker Architecture:
   - Understanding Docker architecture: Docker daemon, client, images, containers, and registries.
   - Docker Engine components: containerd, runc, and Docker CLI.
4) Working with Docker Images:
   - Understanding Docker images: layers, tags, and registries.
   - Docker Hub: exploring and pulling images from the public repository.
   - Building Docker images: creating Dockerfiles, defining dependencies, and best practices.
5) Managing Docker Containers:
   - Creating Docker containers: running containers from images, specifying container configurations.
   - Managing container lifecycle: starting, stopping, pausing, and removing containers.

- Docker container networking: connecting containers, exposing ports, and linking containers.

6) Docker Volumes and Data Management:
   - Understanding Docker volumes: persistent storage for containers.
   - Managing data in Docker containers: mounting volumes, volume drivers, and volume plugins.
   - Data persistence and backup strategies for Docker containers.
7) Docker Compose:
   - Introduction to Docker Compose: defining and running multi-container applications.
   - Writing Docker Compose YAML files: defining services, networks, volumes, and environment variables.
   - Managing multi-container applications with Docker Compose.
8) Docker Networking:
   - Understanding Docker networking models: bridge, host, overlay, and MACVLAN networks.
   - Configuring container networking: exposing ports, creating custom networks, and network isolation.
   - Advanced networking features: Docker Swarm networking, service discovery, and load balancing.
9) Docker Orchestration with Docker Swarm:
   - Introduction to Docker Swarm: container orchestration and clustering.
   - Setting up a Docker Swarm cluster: manager and worker nodes.
   - Deploying and managing services with Docker Swarm: scaling, updating, and rolling updates.
10) Monitoring and Logging with Docker:
   - Monitoring Docker containers and hosts: collecting metrics and logs.
   - Docker logging drivers: configuring log options and forwarding logs to external systems.
   - Container observability tools: Docker Stats, Docker Events, and third-party monitoring solutions.

Kubernetes

1) Introduction to Kubernetes:
   - Understanding container orchestration and its benefits.
   - Introduction to Kubernetes: history, features, and advantages.
   - Comparison of Kubernetes with other container orchestration tools.
2) Kubernetes Architecture:
   - Understanding Kubernetes architecture: master components, node components, and the Kubernetes API.
   - Kubernetes components: kube-apiserver, kube-controller-manager, kube-scheduler, kubelet, and kube-proxy.
   - Kubernetes networking model: pod networking, services, and ingress.
3) Installing and Configuring Kubernetes:

- Installing Kubernetes: setup and configuration on different platforms (local, cloud, on-premises).
- Configuring Kubernetes cluster components: etcd, API server, kubelet, kube-proxy, and container runtime.
- Choosing a Kubernetes distribution: Kubernetes, k3s, Minikube, kops, etc.

4) Working with Kubernetes Objects:
   - Understanding Kubernetes objects: pods, deployments, services, ingress, secrets, configmaps, persistent volume claims, etc.
   - Creating and managing Kubernetes objects using YAML manifests and imperative commands.
   - Managing Kubernetes resources: labeling, annotating, and organizing resources.
5) Pods and Containers:
   - Understanding pods: containers, volumes, and pod lifecycle.
   - Creating and managing pods: running single and multi-container pods.
   - Configuring pod specifications: resource requests, limits, environment variables, and secrets.
6) Deployments and ReplicaSets:

- Introduction to Deployments and ReplicaSets: managing application deployments and scaling.
- Creating and managing Deployments: rolling updates, rollbacks, and scaling applications.
- Configuring Deployment strategies: rolling updates, blue-green deployments, and canary deployments.

7) Services and Networking:

- Understanding Kubernetes services: service types, selectors, and endpoints.
- Configuring service networking: cluster IP, node port, load balancer, and external name services.
- Network policies: controlling traffic between pods and enforcing network policies.

8) Storage and Volumes:

- Understanding Kubernetes storage options: volumes, persistent volume claims (PVCs), and storage classes.
- Configuring persistent storage for stateful applications.
- Using storage providers: local storage, cloud storage, and storage plugins.

9) Cluster Management:

- Managing Kubernetes clusters: adding and removing nodes, upgrading clusters, and cluster maintenance.
- Managing cluster resources: quotas, limits, and resource allocation.