**Answers:**

**What is ci/cd?**

CI/CD stands for continuous integration and continuous deployment (or Continuous Delivery). It’s a set of practices and tools designed to automate the process of integrating code changes and deploying applications.

**What do you use it for?**

CI/CD is used to:

Automate testing of code changes.

Streamline integration of code changes into a shared repository.

Deploy applications automatically to various environments (staging, production).

**What are three common ci/cd stages? Explain each one in details**

First stage – continuous integration (CI):

Automatically build and test code changes to ensure they integrate well with the existing code base. every time a code is committed, automated tests run to check for errors. This helps catch issues early and improves code quality.

Secon stage – continuous deployment (CD):

automatically prepare code changes for deployment but requires manual approval to go live. The code is automatically deployed to a staging environment where it can be tested and reviewed before manual approval is needed to release it to production.

Third stage – continuous deployment (CD):

Automatically deploy code changes to a production environment. After passing CI stages, the application is deployed to production without manual intervention. This ensures that new features and fixes are delivered quickly.

**What is ci pipeline?**

A CI pipeline is a series of automated processes that code changes go through from integration to deployment. It typically includes stages like build, test, and deploy, ensuring code quality and functionality throughout the development lifecycle.

**Which platforms are there for ci/cd?**

Common CI/CD platforms include:

Jenkins

GitLab CI/CD

Travis CI

CircleCI

GitHub Actions

Azure DevOps

Bitbucket Pipelines

**What is gitlab-ci.yml file? What is it used for?**

The gitlab-ci.yml file is a configuration file used by GitLab CI/CD. It defines the pipeline stages, jobs, and scripts that automate the build, test, and deployment processes for a project. It specifies how the CI/CD pipeline should run, including the sequence of tasks and conditions under which they execute.

**What is IAC?**

Infrastructure as Code (IAC) is a practice in which infrastructure—such as servers, databases, networks, and other IT resources—is managed and provisioned through code rather than manual processes. It involves writing configuration files that define the desired state of the infrastructure.

**What are providers in terraform?**

Providers in Terraform are plugins that interact with various cloud platforms, services, or APIs. They enable Terraform to create, read, update, and delete resources on those platforms.

**What are modules in tf?**

Modules in Terraform are containers for multiple resources that are used together. They allow you to organize and reuse your Terraform configurations by grouping related resources into a single, reusable unit.

**What are the advantages of using tf?**

**Consistency**: Ensures consistent infrastructure deployment across environments.

**Version Control**: Infrastructure changes are managed and versioned in code. **Reusability**: Modules and configurations can be reused, reducing duplication.

**Automation**: Automates infrastructure provisioning and management.

**Scalability**: Easily manage and scale infrastructure across various providers.

**What is a state file in tf?**

State File in Terraform tracks the current state of the infrastructure managed by Terraform. It maps the resources defined in the configuration files to their actual state in the cloud or on-premises environment.

**List 5 tf commands and explain what they do**

**terraform init**: Initializes a Terraform configuration directory. It downloads the necessary provider plugins and sets up the working directory.

**terraform plan**: Creates an execution plan showing what actions Terraform will take to achieve the desired state defined in the configuration files.

**terraform apply**: Applies the changes required to reach the desired state of the configuration. It creates, updates, or deletes resources as necessary.

**terraform destroy**: Removes all the resources defined in the configuration files. It is used to tear down the infrastructure.

**terraform state**: Provides commands to manage and inspect the state file, such as listing resources, moving resources between states, or removing resources.