SESSION UNDERSTANDING DOCUMENT ON CI/CD

DevOps:

Overview on DevOps

DevOps is a set of practices that combines software development (Dev) and IT operations (Ops). The goal of DevOps is to shorten the system development life cycle and provide continuous delivery with high software quality. It emphasizes collaboration, communication, and integration between software developers and IT operations.

Key concepts include:

CI/CD: Continuous Integration (CI) involves integrating code changes into a shared repository frequently, while Continuous Deployment (CD) involves the automated release of validated code changes to production.

Automation: Automating repetitive tasks to improve efficiency, consistency, and speed of delivery.

Collaboration: Encouraging cooperation between development and operations teams to achieve shared goals.

Version Control - GIT

GIT is a distributed version control system that tracks changes in source code during software development. It allows multiple developers to work on a project simultaneously without overwriting each other's changes.

Repositories: Central places where code is stored.

Continuous Integration - Jenkins

Jenkins is an open-source automation server used to build, deploy, and automate projects.

Build Automation: Automatically compiling and testing code.

Deploy Automation: Automatically deploying code to different environments.

Pipelines: Represent the series of stages in the software delivery process. Two types are:

Declarative Pipeline: Simplified, structured syntax for defining your pipeline.

Scripted Pipeline: More flexible, allows for complex scripting.

Configuration Management - Ansible

Ansible is an open-source tool for IT configuration management, deployment, and orchestration. YAML files is used for defining a series of tasks to be executed on remote machines.

Dockers - Build and Run Apps Inside Containers

Docker is a platform that allows developers to package applications and their dependencies into containers.

Containers: Lightweight, portable, and self-sufficient units that include everything needed to run a piece of software.

Images: Immutable files that contain the source code, libraries, and tools necessary for an application to run.

Dockerfile: A script containing instructions on how to build a Docker image.

Kubernetes - Orchestration

Kubernetes is an open-source system for automating the deployment, scaling, and management of containerized applications.

Clusters: Groups of hosts running Kubernetes.

Nodes: Individual machines in a cluster.

Pods: The smallest deployable units in Kubernetes, which can contain one or more containers.

Prometheus and Grafana - Monitoring and Alerting

Prometheus is an open-source monitoring and alerting toolkit.

Metrics: Numeric data points representing system performance.

Alert Manager: Handles alerts triggered by Prometheus.

Terraform – Managing Infrastructure as Code

Terraform is an open-source tool developed by HashiCorp for defining and provisioning infrastructure using a high-level configuration language called HashiCorp Configuration Language (HCL). It enables you to manage your infrastructure in a declarative way, ensuring consistency and reproducibility.

Infrastructure as Code (IaC): Use code to define and manage infrastructure, allowing version control and collaboration.

Providers: Plugins that allow Terraform to interact with various services (e.g., AWS, Azure, Google Cloud).

VERSION CONTROL SYSTEMS:

A version control system (VCS) is a tool that helps manage changes to source code or other collections of files over time. It allows multiple developers to work on a project simultaneously without interfering with each other's work, provides a history of changes, and facilitates collaboration.

Types of Version Control Systems

1. **Local Version Control Systems:**

    - These systems keep track of file changes on a local machine.
    - A simple database keeps all the changes to files under revision control.
    - Example: Revision Control System (RCS).

2. **Centralized Version Control Systems (CVCS):**

    - A single central repository is used to store all the versioned files.
    - Developers check out files from the central server to work on them and then check them back in when finished.
    - Advantages: Easier administration, centralized backup.
    - Disadvantages: Single point of failure, limited offline work capabilities.
    - Examples: Concurrent Versions System (CVS), Subversion (SVN), Perforce.

3. **Distributed Version Control Systems (DVCS):**

    - Each developer has a complete copy of the repository, including the full history of changes.
    - Changes can be committed locally and later pushed to a central server or shared with other developers.
    - Advantages: No single point of failure, enhanced collaboration, better performance, and the ability to work offline.
    - Examples: Git, Mercurial, Bazaar.

GIT:

Git is a distributed version control system that helps developers track changes in their code over time, collaborate on projects, and manage different versions of their software. It is widely used in software development to maintain a history of changes, branch out code for different features or fixes, and merge these changes back into the main codebase.

Key features of Git include:

Version Tracking: Keeps a history of changes made to files and directories.

Branching and Merging: Allows developers to work on different features or fixes in isolated branches, and then merge these changes back into the main codebase.

Distributed System: Every developer has a complete copy of the repository, including its history, allowing for offline work and better collaboration.

Commit History: Each change is recorded as a commit, which includes a message describing the change and metadata about the author and timestamp.

Common commands in Git include:

- git init: Initializes a new Git repository.
- git clone: Clones an existing repository.
- git add: Stages changes for the next commit.
- git commit: Records staged changes to the repository.
- git push: Uploads local commits to a remote repository.
- git pull: Fetches and integrates changes from a remote repository.
- git merge: Combines changes from different branches.
- git branch: Lists, creates, or deletes branches.

STASH:

Stash in Git refers to a feature that allows developers to temporarily save changes that are not yet ready to be committed. This is useful when you need to switch branches or perform other tasks without losing your current work.

Key commands for using stash include:

git stash: Stashes the current changes in the working directory and index.

git stash list: Lists all stashed changes.

git stash apply: Applies the stashed changes back to the working directory.

git stash pop: Applies the stashed changes and removes them from the stash list.

git stash drop: Discards a specific stash from the stash list.

Using Stash with GIT:

Steps:

- Setting up a repository in Stash
- Cloning the repository
- Pushing changes to Stash
- Pulling updates from Stash

Advantages of Using Stash:

- Code Review
- Pull Requests
- Permission Management
- Integration with other Atlassian tools (e.g., JIRA)

Code Review

- **Facilitated Code Reviews:** Bitbucket Server provides built-in tools for conducting code reviews, allowing teams to discuss changes, comment on specific lines of code, and ensure quality before merging.
- **Improved Code Quality:** Regular code reviews help catch bugs and maintain coding standards, leading to higher-quality codebases.

Pull Requests

- **Streamlined Collaboration:** Pull requests enable developers to propose changes to the codebase, which can be reviewed, discussed, and approved by team members before being merged.
- **Controlled Merging:** This process ensures that only reviewed and approved changes are integrated into the main codebase, reducing the risk of introducing errors.

Permission Management

- **Granular Access Control:** Bitbucket Server allows administrators to set fine-grained permissions, ensuring that only authorized users can access or modify certain repositories and branches.
- **Enhanced Security:** This level of control helps protect the integrity of the codebase by preventing unauthorized changes.

Integration with Other Atlassian Tools (e.g., JIRA)

- **Seamless Integration:** Bitbucket Server integrates tightly with JIRA, Atlassian's issue and project tracking tool. This integration allows developers to link commits, branches, and pull requests to JIRA issues.
- **Improved Workflow:** Developers can view the status of related JIRA issues directly within Bitbucket, facilitating better tracking and management of development progress.
- **Automated Processes:** Automation rules can be set up to transition JIRA issues based on actions in Bitbucket, streamlining the workflow and reducing manual updates.

These advantages make Bitbucket Server a powerful tool for managing Git repositories, especially in organizations that rely on Atlassian's ecosystem for software development and project management.