DevOps vs. Traditional IT and Agile

Aspect	DevOps	Traditional IT	Agile
Focus	Collaboration and automation.	Isolated development and operations.	Iterative development.
Speed	Continuous delivery and faster deployment.	Slower, rigid workflows.	Incremental but without operations integration.
Automation	Highly automated processes.	Manual configurations.	Limited focus on automation.
Team Structure	Cross-functional teams.	Separate teams.	Development-focused teams.
Feedback	Continuous integration of feedback.	Delayed feedback.	Rapid development feedback.

Core DevOps Principles

1. **Collaboration and Communication:** Close cooperation between development, operations, and QA.
2. **Automation:** Automating repetitive tasks like testing, builds, and deployments.
3. **Continuous Improvement:** Iteratively enhancing the processes and tools.
4. **Monitoring and Feedback:** Using metrics to track and improve performance.
5. **Infrastructure as Code (IaC):** Managing infrastructure through code to ensure consistency.

DevOps Lifecycle

The **DevOps lifecycle** consists of the following stages:

1. **Plan:** Define business requirements and plan workflows.
2. **Develop:** Write and manage code efficiently.
3. **Build:** Compile code into deployable artifacts.
4. **Test:** Ensure the application works as expected.
5. **Release:** Make the software ready for deployment.
6. **Deploy:** Roll out the software into production.
7. **Operate:** Monitor and maintain the software.
8. **Monitor:** Collect feedback and monitor performance for continuous improvement.

Continuous Integration and Delivery (CI/CD)

CI/CD is a DevOps practice focusing on automating and improving the software release process.

- **Continuous Integration (CI):** Developers frequently merge their code changes into a central repository where automated builds and tests are run to detect issues early.
- **Continuous Delivery (CD):** Ensures that code changes are automatically prepared for release to production after passing through rigorous testing.

Introduction to CI/CD Pipeline

A **CI/CD pipeline** is a set of automated steps to:

1. Fetch code from version control (e.g., Git).
2. Build the application (e.g., with Maven).
3. Test the application automatically.
4. Package and deploy it to production environments.

---

1. **Git:** A distributed version control system for tracking changes in code.
2. **Jenkins:** An open-source automation server for building, testing, and deploying applications.

What is Git?

Git is a free, open-source **distributed version control system (VCS)** designed to handle everything from small to very large projects with speed and efficiency. It helps developers track changes in source code and collaborate on software development.

- **Key Features of Git:**
    1. **Version Control:** Tracks changes to code over time, allowing you to revert to previous versions.
    2. **Branching and Merging:** Allows developers to work on separate features (branches) and later merge them.
    3. **Distributed System:** Each developer has a full copy of the repository, ensuring no single point of failure.
    4. **Efficient:** Optimized for performance with fast operations on local machines.

What is GitHub?

GitHub is a **cloud-based platform** built around Git. It provides a graphical interface and additional collaboration tools to manage Git repositories, enabling developers to host, share, and collaborate on code.

- **Key Features of GitHub:**
    1. **Code Hosting:** Stores Git repositories online.
    2. **Collaboration Tools:** Includes pull requests, code reviews, and discussions.
    3. **Integration:** Works with CI/CD tools, project management tools, and more.
    4. **Versioning and Backup:** Keeps code safe and versioned in the cloud.
    5. **Open Source and Community:** Millions of open-source projects are hosted on GitHub.

Differences Between Git and GitHub

Aspect	Git	GitHub
Definition	A version control system.	A hosting platform for Git repositories.
Usage	Tracks and manages source code changes locally.	Stores repositories online for collaboration.
Installation	Installed on local machines (CLI-based).	Accessed via a web interface or CLI.
Features	Branching, merging, and local version control.	Pull requests, code reviews, and online collaboration.
Hosting	No hosting; files stored locally.	Cloud-hosted repositories.

How Git and GitHub Work Together

1. **Git:** Manages the local repository on your computer, tracks changes, and creates snapshots of your code.
2. **GitHub:** Hosts the remote repository where you can push your local changes to share them with your team.

Common Git Commands

1. **git init:** Initialize a new Git repository.

2. **git clone [URL]:** Clone an existing repository.
3. **git add [file]:** Stage changes for the next commit.
4. **git commit -m "message":** Save changes to the repository with a message.
5. **git push:** Upload changes to a remote repository (e.g., GitHub).
6. **git pull:** Fetch and merge changes from the remote repository.
7. **git status:** View the current state of the repository.
8. **git branch:** List, create, or switch branches.

What is Jenkins?

Jenkins is an **open-source automation server** designed to facilitate Continuous Integration (CI) and Continuous Delivery (CD) of software. It helps automate parts of the software development process, such as building, testing, and deploying applications.

Key Features of Jenkins

1. **Open-Source:** Free to use and supported by an active community.
2. **Extensibility:** Offers over 1,800 plugins to integrate with various tools and platforms (e.g., Git, Maven, Docker, Kubernetes).
3. **Distributed Builds:** Supports distributed workloads to speed up builds by running them across multiple nodes.
4. **Easy Configuration:** Provides a web-based GUI for configuration and management.
5. **Pipeline Automation:** Automates CI/CD pipelines using a scripting language called Jenkinsfile.

Why Use Jenkins?

- **Automation:** Reduces manual intervention in repetitive tasks.
- **Continuous Integration:** Automatically integrates and tests code changes to detect issues early.
- **Continuous Delivery:** Ensures that the software is always ready for deployment to production.
- **Collaboration:** Enables teams to work together efficiently by providing shared pipelines and reporting.

How Jenkins Works

1. **Source Code Management:** Connects to version control systems (e.g., Git, GitHub).
2. **Build Automation:** Uses tools like Maven, docker and package applications.
3. **Testing:** Runs automated tests (unit, integration, etc.) to ensure code quality.
4. **Deployment:** Deploys applications to staging or production environments automatically.
5. **Feedback:** Provides real-time feedback on builds, tests, and deployments.