Linked In : https://www.linkedin.com/in/satyajeet-desai-001962180/

1. Git (Version Control)

Purpose: Git is a version control system. It helps developers track and manage changes to their code over time.

Workflow:

Developers write code locally (on their machines) and save their changes in a Git repository.

They can commit changes, create branches, and merge code to keep track of different versions.

When a developer is happy with their changes, they "push" them to a remote Git repository (like GitHub, GitLab, or Bitbucket) so others can access it.

2. Jenkins (Continuous Integration and Continuous Delivery)

Purpose: Jenkins is an automation server used to automate tasks in the software development lifecycle. It helps with Continuous Integration (CI) and Continuous Delivery (CD).

Workflow:

When changes are pushed to a Git repository (for example, code is updated), Jenkins can automatically detect this change using webhooks.

Jenkins then fetches the code from Git and runs tests to make sure the code is working properly (this is the Continuous Integration part).

After testing, Jenkins can automatically package the code and create a deployable artifact (like a Docker image) and push it to a container registry or deploy it to a server (this is the Continuous Delivery part).

3. Docker (Containerization)

Purpose: Docker is used to create, deploy, and run applications in containers. A container is a lightweight, portable, and consistent way to package an application along with its dependencies (libraries, configurations, etc.).

Workflow:

After the code passes tests and is ready, Jenkins can use Docker to package the application into a Docker image.

The image contains everything needed to run the application, such as code, dependencies, environment variables, etc.

Developers can then push the Docker image to a Docker registry (like Docker Hub or a private registry).

Docker ensures that the application runs the same way on any machine, whether it's your laptop or a production server.

4. Kubernetes (Container Orchestration)

Purpose: Kubernetes is a platform for automating the deployment, scaling, and management of containerized applications (like Docker containers).

Workflow:

Once Docker containers are created and pushed to a registry, Kubernetes takes over to manage how these containers run in a production environment.

Kubernetes can automatically scale applications (e.g., run more containers if traffic increases) and ensure the application stays up and running by restarting failed containers.

It uses Pods (groups of containers) to deploy applications, and Kubernetes controls how containers communicate, get deployed, and are exposed to users.

All in a Simple Workflow:

Code Development (Git):

Developers write code on their local machines and commit the changes to Git.

CI/CD Pipeline (Jenkins):

Jenkins detects the changes in the Git repository and automatically starts the process (CI/CD).

Jenkins pulls the code, runs tests, and builds a Docker image if the tests pass.

Containerization (Docker):

The Docker image contains the application and all its dependencies.

Jenkins pushes the Docker image to a container registry.

Deployment and Scaling (Kubernetes):

Kubernetes pulls the Docker image from the registry and deploys the application to a set of machines. Kubernetes ensures the application is running and scales it up or down based on demand.

Scenario:

A developer commits code to GitHub.

Jenkins detects the commit and pulls the code to run tests.

If tests pass, Jenkins builds a Docker image for the application and pushes it to Docker Hub.

Kubernetes automatically deploys the Docker container to a cloud environment, ensuring it runs smoothly and scales as needed.

Summary:

Git: Code version control (developers push and pull code).

Jenkins: CI/CD automation (builds and tests the code, creates Docker images).

Docker: Containerization (packages the application with everything it needs).

Kubernetes: Orchestration (manages the deployment, scaling, and running of Docker containers).

IMP Differences:

Git tracks code changes and allows collaboration.

Jenkins automates the process of testing, building, and deploying software.

Docker packages and runs applications in containers, making them portable.

Kubernetes manages how containers run in production, ensuring reliability and scalability.

In simple words:

Git is where you store your code.

Jenkins makes sure your code is always tested and ready to go.

Docker puts your code in a neat box so it can run anywhere.

Kubernetes organizes and controls all the boxes (containers) in production.