EE 6254: DEVOPS ENGINEERING

ASSIGNMENT 3

Group Number : 44

Group Members : EG/2020/4291 : Wijerathna W.M.M.T

EG/2020/4149 : Randula R.D.

Date : 27/06/2024

Part 1: CICD Design Diagram

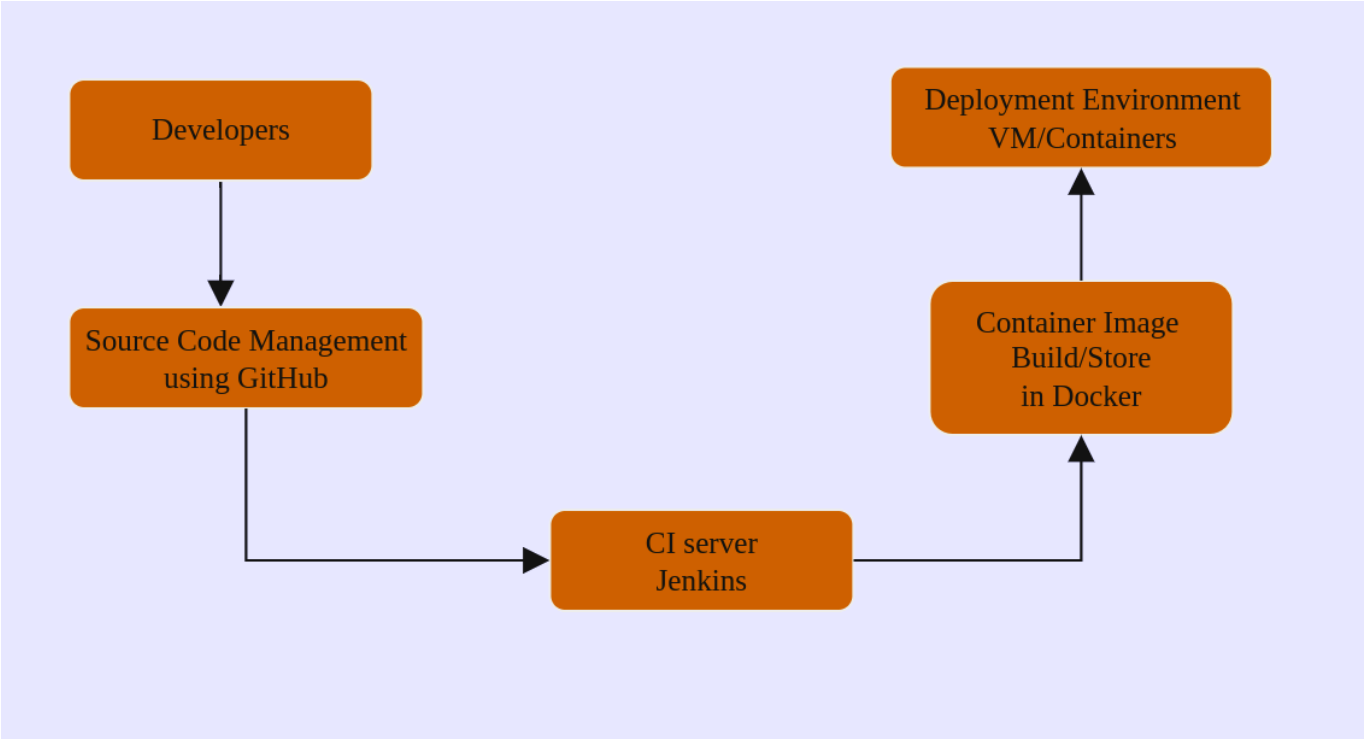

Figure 1: CICD Design Diagram

Explanation

Developers

Developers are responsible for writing and maintaining application code. They use various programming languages and tools to implement new features, resolve issues, and enhance the application's performance. When making updates, they save their code in a Git repository such as GitLab or GitHub, accompanied by a commit message that details the changes. By using version control systems like Git, developers can track modifications, collaborate with teammates, and revert to previous versions when needed. To ensure the code meets high standards and adheres to best practices, developers frequently conduct peer code reviews.

Git Repository

A Git repository serves as a central hub for storing code, offering version control and effective code management. It tracks every modification made to the code, enabling developers to revert to earlier versions if necessary. Git's branching and merging capabilities allow multiple developers to work on different features or fixes simultaneously without disrupting each other’s work. Pull requests facilitate code reviews and discussions before incorporating changes into the main branch. Additionally, the repository integrates with CI/CD pipelines to automate the building and deployment processes.

CI Server (Jenkins)

Jenkins is a primary tool for Continuous Integration (CI), automating the process of building, testing, and integrating code changes to ensure the code is consistently deployment-ready. Upon detecting a commit push in the Git repository, Jenkins pulls the latest code, compiles it, and executes various tests to verify its functionality. It then packages the application into a deployable format, such as a Docker image or a JAR file. Jenkins also offers detailed logs and reports, enabling developers to swiftly identify and resolve issues.

Container Image Build/Store (Docker)

We use Docker to containerize our application, bundling it and its dependencies into a single, portable container. Docker images are created with Docker files, which define the application's environment, dependencies, and startup commands. After being built, these images are stored in a Docker registry like Docker Hub or a private registry. The Docker registry manages image versions and ensures the correct images are used during deployment. It also supports image scanning to detect vulnerabilities and enforce security policies. Containerization ensures consistency across different environments, from development to production.

Deployment Environment (VMs/Containers)

The deployment environment utilizes virtual machines (VMs) or containers to run the application. VMs offer an isolated environment with their own operating system, while containers share the host OS kernel but provide isolated user space. Kubernetes oversees the deployment of these containers on VMs or container hosts, ensuring efficient resource utilization and seamless updates. It schedules containers according to their resource requirements and availability, and can automatically scale the number of containers up or down based on demand. This allows the application to manage varying traffic levels without manual intervention.

Connectivity in the CI/CD Pipeline

- Developers push code to the Git repository.
- Jenkins retrieves the code from the Git repository, compiles it, and uploads the build artifacts to the artifact storage.

Jenkins initiates Docker image builds, which are then stored in a Docker registry.

Part 2: Automation Approach

Application Tools and Dependencies

Node

- Version: v20.15.0.

React

- Version: 18.2.0
- Purpose: Frontend JavaScript library for Web application Development

Express

- Purpose: Backend using Nodejs for Web application Development

MongoDB

- MongoDB Atlas(online)
- Purpose: Database management system for storing and managing application data.

Automated Application Deployment

DevOps Tools

GitHub

- Git – Version 2.34.1
- Purpose: GitHub serves as both a version control system and a code repository. It monitors code modifications and facilitates collaboration among developers. GitHub includes features for branching, pull requests, and code reviews to maintain code quality and streamline the incorporation of new functionalities. Additionally, it integrates seamlessly with CI/CD pipelines to automate tasks such as builds and deployments upon code commits.

Jenkins

- Version: 2.440.3
- Purpose: Jenkins is employed for Continuous Integration (CI) and Continuous Deployment (CD), automating the processes of building, testing, and deploying applications. It retrieves the most recent code from GitHub, compiles it, conducts tests for functionality validation, and prepares it for deployment in various formats. Jenkins is highly extensible with numerous plugins that enhance its capabilities and enable seamless integration with a wide range of DevOps tools.

Docker

- Version: 27.0.1
- Purpose: Docker is a tool that bundles applications and their dependencies into portable containers, ensuring uniform performance across various environments, from development to production. It simplifies setting up development environments, testing, and deployment processes. These containers are lightweight, isolated, and provide a secure and efficient way to execute applications.

Automated Deployment Process Using Jenkins CI/CD Pipeline

Our application deployment is completely automated through a Continuous Integration/Continuous Deployment (CI/CD) pipeline overseen by Jenkins. Here's a comprehensive overview of its operation:

Code Commit

1. Developer Actions:
    - Developers submit code changes to GitHub.
    - Each submission includes a commit message detailing the modifications made to the code.
2. Triggering the Pipeline:
    - When new code changes are committed to GitHub, a webhook configured within GitHub activates the CI/CD pipeline in Jenkins.
    - Jenkins then starts the build process.

Build Stage

1. Code Retrieval:
    - Jenkins fetches the most recent code from the GitHub repository to ensure it possesses the latest version of the application.
2. Build Process:
    - Jenkins utilizes build tools such as Maven for compiling the code.
    - Throughout this phase, it resolves all dependencies and compiles the application into executable binaries or packages.

Containerization

1. Building Docker Images:

    • Docker generates images using the application artifacts.

    • Each image contains the application and its runtime environment, ensuring uniformity across diverse deployment environments.

2. Pushing to Docker Registry:

    • Docker tags the built images with version numbers and uploads them to a Docker registry, such as Docker Hub or a private registry.

    • This registry serves as a centralized storage for all container images, enabling their availability for deployment.

After, combining these tools and methods, our CI/CD pipeline ensures a strong, streamlined, and automated deployment process, maintaining high benchmarks for code quality, reliability, and performance.