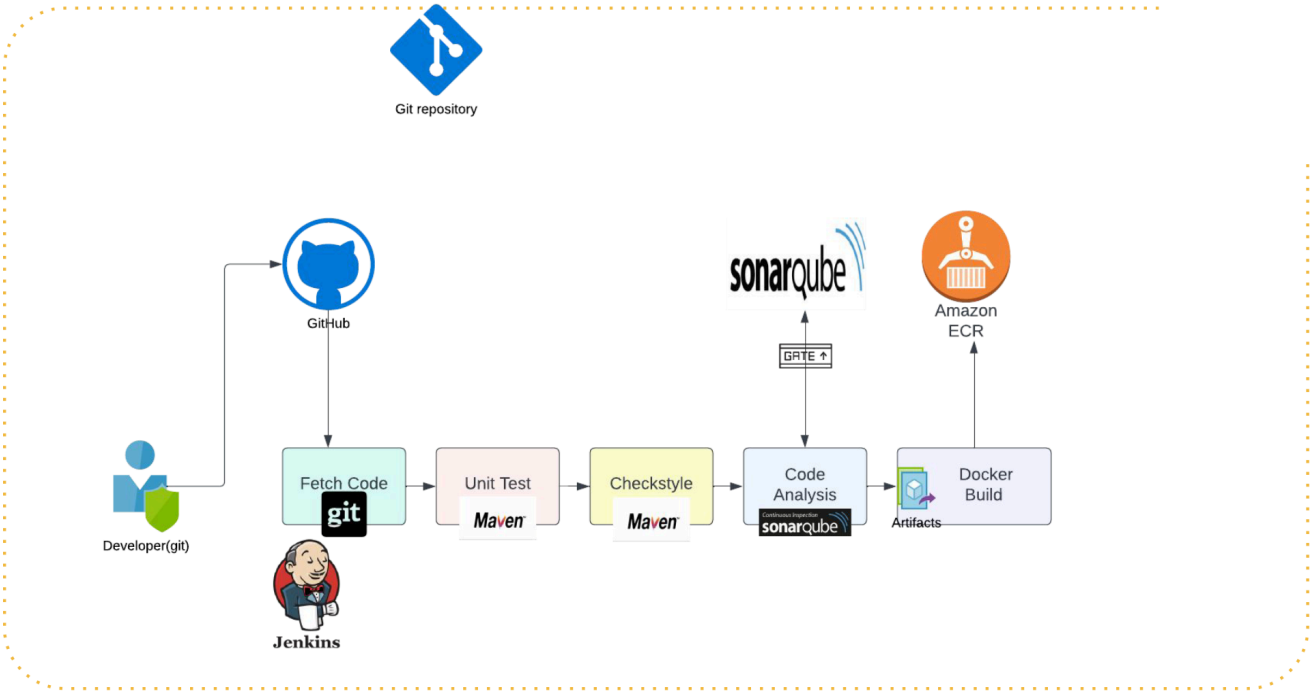
Flow of Continuous Integration Pipeline

The flow of a Continuous Integration (CI) pipeline that integrates code from a Git repository, performs unit testing, code quality checks, and code analysis before finally building a Docker image and pushing it to Amazon Elastic Container Registry (ECR).



CI Pipeline Workflow

1. Developer Commits Code (GitHub)

- The process begins when a developer pushes code to a GitHub repository.
- This triggers the Jenkins job, signaling that new code has been committed and needs to be integrated, tested, and analyzed.
- GitHub acts as the version control system where the project code resides.

2. Fetch Code (Jenkins & Git)

- Jenkins, the automation server, is configured to automatically pull the latest code from GitHub using the Git plugin.
- Once the code is fetched, Jenkins prepares the workspace for the next steps.
- The fetched code is placed in Jenkins' workspace where further actions like testing and building will take place.

3. Unit Test (Maven)

- After fetching the code, Jenkins triggers the Unit Test stage using Maven. Maven is a build automation tool primarily used for Java projects.
- This stage runs unit tests to ensure that the individual components or units of the application are working as expected.
- If any test fails, the pipeline halts, and feedback is provided to the developer. If the tests pass, the process continues.

4. Checkstyle (Maven)

- In this step, Jenkins uses Checkstyle (through Maven) to check the coding standards and style guidelines.
- Checkstyle analyzes the source code for potential quality issues like improper formatting or naming conventions. This ensures that the code adheres to a specific coding standard, improving readability and maintainability.
- If any issues are found, the pipeline might either log a warning or fail depending on the severity of the violations.

5. Code Analysis (SonarQube)

- Once the code passes unit tests and code style checks, it goes through Code Analysis using SonarQube.
- SonarQube performs a deep analysis of the codebase to identify issues like bugs, code smells, vulnerabilities, and potential security risks.
- After analysis, SonarQube provides a report, often referred to as the Quality Gate, which sets thresholds for acceptable code quality. If the code passes the quality gate, the pipeline proceeds to the next step.
- If the code does not pass, feedback is provided, and the developer is notified to address the issues.

6. Docker Build

- After the code is validated through testing and analysis, Jenkins moves on to building a Docker image.
- Docker is used to package the application into containers, ensuring that it runs consistently across different environments.
- In this step, Jenkins compiles the application, packages it into a Docker image, and prepares it for deployment.

7. Artifact Management

- Once the Docker image is built, the generated artifacts (e.g., the Docker image or any other build output) are prepared for shipping.
- The artifacts could include compiled code, logs, or other build outputs, ensuring that they are preserved and ready for future use, deployment, or review.

8. Push to Amazon ECR

- Finally, Jenkins pushes the Docker image to Amazon Elastic Container Registry (ECR), a fully managed container registry.
- ECR serves as the storage for the Docker images, making them available for deployment in production or staging environments.
- Once the image is stored in ECR, it can be pulled by any service that needs to deploy or run the application in a containerized environment, such as AWS ECS or Kubernetes.

SUMMARY

Each step of this pipeline is carefully connected:

- The process starts when new code is committed, triggering Jenkins to fetch the latest version of the code.
- Automated unit tests ensure the integrity of the application, while Checkstyle enforces good coding practices.
- SonarQube ensures that the code is not only functional but also secure and of high quality.
- Once the code quality is assured, the application is packaged into a Docker container for easy and consistent deployment.
- Lastly, the Docker image is stored in Amazon ECR, ready to be deployed in production environments, making the entire process automated, efficient, and repeatable.