Chris Faris

CEN4802C

10/4/24

midterm presentation

Welcome,

This project will demonstrate the following tools:

* IntelliJ IDEA for code development.
  + - IntelliJ is in my opinion, the ultimate IDE for everything Java, worth noting you get the ultimate edition for free on JetBrains if you are a student.
* Git/GitHub for version control.
  + - Git is a version control system that tracks changes in code over time. It allows multiple developers to work on a project simultaneously, maintain different versions (branches), and merge them back together. GitHub is a platform that hosts Git repositories online, making it easy for developers to collaborate and share code.
    - Benefits- collaboration, backup/remote access, integration.
* JUnit for unit testing.
  + - Unit testing is where you take a piece of code from a system and test it by itself to determine if it behaves as designed. Through unit testing you can verify correct behavior and highlight incorrect behavior and be able to modify the code accordingly to eliminate it. I chose JUnit because of my familiarity with it.
    - Benefits- faster debugging, improved code quality, good documentation.
* Jenkins for CI/CD.
  + - Jenkins is an automation tool that helps developers build, test and deploy code automatically.
    - Continuous integration or CI is the process of automatically building and testing code when changes are made, ensuring the code stays functional.
    - Continuous deployment or CD is the process of automatically deploying tested code to chosen environment (chose to keep this process manual).
    - Benefits- automation, fast test feedback, consistent releasing.
* Docker for deployment.
  + - Docker is a tool that packages applications and their dependencies into containers, ensuring they run consistently in different environments.
    - Consistency, isolation, portability.

A quick overview of the workflow and structure of this project:

 **Developer pushes new code to version control (Git/GitHub)**.

 **Jenkins automatically triggers a build**:

* Pulls the latest code.
* Runs the build process (e.g., Maven for a Java project).

 **Automated tests run on Jenkins**:

* Unit tests, integration tests, etc.
* Code quality checks (e.g., static analysis tools).

 **If tests pass, create Docker image through IDE integration**:

* Builds the Docker image with the new code.
* Optionally, tags the image with a version or latest.

 **Pushes the Docker image to a container registry** (e.g., Docker Hub, AWS ECR).

 **Deployment to staging or production**:

* The new Docker image is pulled and deployed in the appropriate environment.

Key takeaways:

* Focuses on 2 main environments (IntelliJ, Jenkins), with some terminal and GitHub usage.
* Immediate project build and testing upon commit/push.
* Access to Docker tools right within the IDE for easy deployment.

Snippets:

**ngrok command**

ngrok http --url=parrot-mutual-grouse.ngrok-free.app 8080

add Main-Class: (class name) to manifest

if can’t find run

jar xf target/(name)-1.0-SNAPSHOT.jar META-INF/MANIFEST.MF