ENG1 - Assessment 2

Continuous Integration

CI2.pdf

Group 6

Freya Goodger	sg1967
Mikolaj Wyrzykowski	mw2179
Barnaby Matthews	bm1287
Cooper Love	cl2702
Oliver Cassey	oc854
Anna Hrynyshyn	ah2886
Oliver Thompson	ot699

In order to enhance collaboration and streamline development cycles in our project we employed Continuous Integration(CI). This methodology is aligned with the practices recommended by Martin Fowler, which mitigate the risks associated with delayed integration and maximises project transparency. Our CI strategy focuses on integrated work frequently to detect integration issues early, ensuring that the software remains in a state where it can be released at any time[1]. This approach is particularly applicable in the team environment where multiple developers interact with the same codebase, ensuring that at every point everything is compatible and correctly functioning.

Key Practices that we have implemented:

- Automated build: Using Github Actions, we automated the compilation, unit testing testing, and integration testing, ensuring that build is always in a deployable state.
- Self-Testing builds: Each build undergoes the automated tests to validate the code's functionality before the build can be considered successful. This is crucial to identify any bugs in the early development stage.
- Regular Commits: Developers of the product were integrating their work with the main branch frequently , thus facilitating regular updates and minimising integration challenges.

Our CI pipeline consists of two main jobs designed to handle testing and releasing processes effectively:

1. **Test Job:**

   This job is crucial as it ensures that every change pushed to the repository or every pull request is thoroughly tested, maintaining the integrity and quality of the codebase.

- Code Checkout: Utilises **'actions/checkout@v3'** for checking out the latest code, ensuring that the tests run on the most recent version.
- Java Environment Setup: Sets up Java using **'actions/setup-java@v4'**, specifying the Temurin distribution and Java version 11, preparing the environment for Java builds.
- Code Format Check: Runs **'./gradlew spotlessCheck'** to ensure that the code adheres to predefined formatting standards, which helps in maintaining consistency across the codebase.
- Local Formatting Command: Running **'./gradlew spotlessApply'** locally before committing reformats all Java files according to predefined style guidelines, ensuring that commits meet the CI pipeline's formatting check requirements.
- Unit Testing: Executes './gradlew clean test' to perform unit testing on the code, cleaning any previous builds and ensuring all tests pass with the latest changes.
-

2. **Release Job:**

   This job is manually triggered, providing the team with control over the timing of releases, which is critical for aligning releases with project milestones and readiness.

- Environment Preparation: Similar steps as the test job, ensuring the environment is ready for building the application.

- Application Building: Compiles the application using **‘./gradlew desktop:dist’**, which generates the distributable JAR files for the desktop version.
- Version Tagging: A custom script extracts the current version from the Gradle properties, logs it, and ensures that the GitHub release is tagged accurately.
- Automatic Release Creation: Utilises **‘marvinpinto/action-automatic-releases@latest’** for creating a release on GitHub, ensuring precise tagging and review of each release.
- Version Increment: Updates the project version for ongoing development by editing and cleaning the **‘build.gradle’** file.
- Version Commit: Uses ‘**stefanzweifel/git-auto-commit-action@v5’** to automatically commit the version changes to the repository.

Permissions and Security:

- Configured with **‘write-all’** permissions to ensure that the pipeline can perform necessary operations such as committing changes and managing releases securely.

Repository Documentation and In-line Comments further enhanced our CI process, providing clear guidance and detailed explanations within our GitHub repository. The ‘README.md’ file provides an overview of our product, making it easier and more accessible to run it. Meanwhile, extensive comments within our workflow files facilitates modifications, ensuring that our CI practices remain adaptable to changes

To sum up, by implementing CI best practices to ensure a smooth integration of our work.

References

[1] Fowler, M. (2006). Continuous Integration. [online] martinfowler.com. Available at: https://martinfowler.com/articles/continuousIntegration.html