CI/CD Pipeline - Phase 1

By Team 31

In this phase, we explored linting and code style enforcement, code quality via tool, code quality via human review, unit tests via automation, and documentation generation via automation. Our team's pipeline is a trunk-based development that was developed in order to foster communication, norm and storm, and avoid bad merges. Here, we will be explaining the intricacies and specifics of our pipeline we have developed.

First, the developer will develop their code locally in the VisualStudio Code IDE. The developer will get the initial code by pulling the code from the master branch of our team's GitHub repository. Next, the developer will have to check the code for programmatic and stylistic errors through a process called code linting using ESLint inside the Visual Studio Code IDE. Then, the developer will have to push their code to the team's repository.

Second, we will be checking the code's quality and style using a tool called Codeclimate. Afterwards, the whole team will participate in a code review by reviewing the code on GitHub and discussing the newly pushed code over Zoom. Everyone will discuss their concerns, questions, and suggestions, so we can improve and edit the code accordingly.

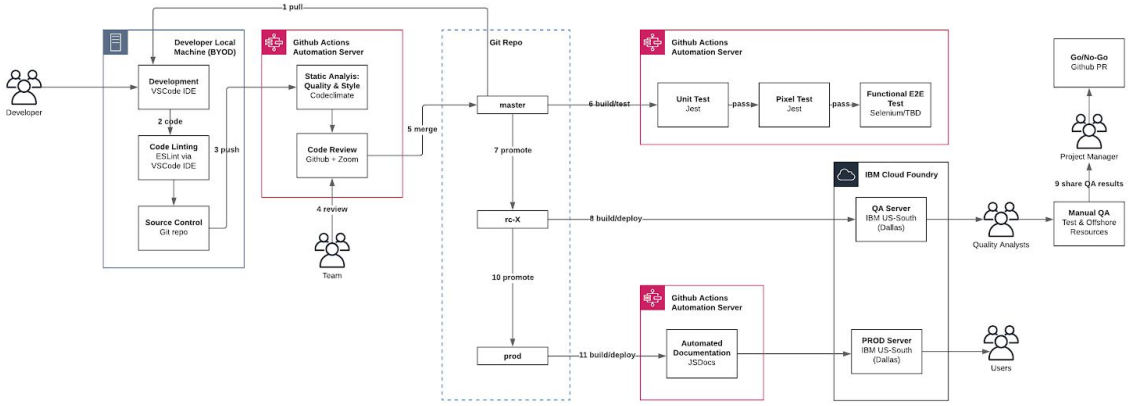
After all the code has been successfully reviewed and approved by the entire team, we will then merge the code to the master branch of our team's GitHub repository and resolve any pull requests. The code will then be built and it will be put through a series of automated tests using GitHub Actions. There will be a total of three tests that the code will have to go through: Unit Test via Jest, Pixel Test via Jest, and a Functional E2E Test via Selenium.

Next, the code will be promoted using rc-X. In the IBM Cloud Foundry, the code will be deployed to the QA Server using IBM US-South (Dallas). Next, the code will have to go through a quality analysis and a manual quality analysis through testing and offshore resources. The results from the quality analyses will be reviewed by the Project

Manager who will then ultimately determine whether the product will be given a Go or No-Go.

Finally, from the rc-X the code will then be promoted to production through automation and will thus be eligible for building and deploying. Then, the code will go through documentation generation via automation using JSDocs tool. The final product will then be deployed to the IBM Cloud Foundry in a PROD Server using the IBM US-South (Dallas). This final step will allow the product to be available for users to use and enjoy.

The figure below shows the process of our pipeline that we have described in this document.



By developing our pipeline, we have successfully completed exploring linting and code style enforcement, code quality via tool, code quality via human review, unit tests via automation, and documentation generation via automation. All of these processes can be seen in our pipeline diagram above and also the written description. Also, all five of these things have been completed on our GitHub and are successfully working. However, we have yet to implement our other tasks.

Overall, our team was able to create a pipeline that is a trunk-based development that was developed in order to foster communication. It allows the team to norm and storm, and most importantly, our pipeline makes sure that our team does not implement bad merges on our GitHub repository.