**Jenkins vs. GitLab CI**

**Enterprise Systems Integration**

Assignment 2

ACIT4850

Jeremy Lam - A01253371

Adel Kuanysheva - A01258780

October 26th, 2022

# Introduction

The choice between continuous integration tools is a largely debated topic amongst developers. The two most considerable of these tools are Jenkins and GitLab CI, both open-source management platforms that help with the DevOps stages of developing, testing, and deploying. Both hold important benefits and drawbacks which are discussed below.

# GitLab CI

## **Benefits** – Setup, Scalability & Integration

We can conclude, after setting up both integration tools, that setting up GitLab CI is less demanding. This tool is self-contained and has the ability to integrate everything in the same product under the same browser tab [1]. From the help of runner binaries, configuration and scaling take a short amount of time. Along with the graphical interface and a declarative language, GitLab CI can orchestrate many types of pipelines (chains, runs, monitors, etc). The tool naturally integrates with GitLab source code, we simply create pipelines using ***gitlab-ci.yml*** files and change them with the provided GUI.

## **Drawbacks** – Complexity & Analytics

When the setup is finished, GitLab CI becomes not all that easy. The company has numerous unfinished features which creates many problems for developers needing a solution for unclear tasks. Files created for build processes with important data become redundant since they need to be created for each job that is run. There is no easy way to share the same ***gitlab-ci.yml*** script between repositories either. To add, this tool does not contain analytics for pipeline tracking. Hence, important KPIs are lost during builds and cannot be retrieved.

# Jenkins

## **Benefits** – Customization & Support

Being the most popular CI tool, Jenkins does much more than just pipelines. The platform possesses an extensive library of plug-ins are supplement specific actions that developers are trying to accomplish. There are countless options on how to display outputs, such as having a pass/fail test list. Jenkins provides parallel execution, REST API support, and work distribution, which makes this a better integration tool for larger projects. Another reason that Jenkins is chosen over GitLab CI so frequently is for the technical support. The tool itself has a sizeable documentation as well as many community forums and patrons who are willing to provide guidance, this is what paints Jenkins as the more advantageous CI tool.

## **Drawbacks** – Setup & Maintenance

The setup process of Jenkins compared to GitLab CI feels more manual. The imperative programming language has a not-so easy syntax to read or understand. As well, Jenkins jobs and views can look very messy with the legacy UI it uses. After running through the entire setup process, it seems as though Jenkins makes it too easy to do the wrong thing. With its ecosystem of plug-ins, there are many paths for a project to go south quickly. Maintenance will be tough for developers that are not extremely well versed in the platform.

# Recommendation

Learning both CI tools is the best way to interpret what will work best for a certain project. However, the tool that has the overall best benefits to drawbacks ratio is Jenkins. Being the most popular in the tech community, it is easy to see why. This tool does not require a large budget or people to get it working efficiently and is best suited for small to medium companies. Once the overhead of the setup process is finished, the tool provides easy to deploy code, a large plugin library, and tons of support. Solely for Continuous Integration, Jenkins wins the competition.

GitLab CI should be used more for collaboration and version control. It should not be chosen due to its complexity with code structures and lack of analytics. These drawbacks create a lot of problems down the line when it is time to refactor code. This tool, despite its flaws, are great for larger projects based off its granularity and centralized setup and configuration. To avoid the complexity issue, perhaps the best action would be to only use GitLab CI for quick tests. Jenkins can be used for larger builds and deployments.

Overall, Jenkins provides the most benefits when considering the price and training that it requires. It fits into many different projects due to its customization abilities. As well, support within the Jenkins community provides new Jenkins users with lots of documentation to reference during integration.

# References

[1] GitLab. (n.d.). *Gitlab Runner*. GitLab. Retrieved October 26, 2022, from <https://docs.gitlab.com/runner/>

[2] *Make Better Quality Software using Jenkins for your CI/CD pipeline*. CloudIQ Tech. (2020, January 31). Retrieved October 26, 2022, from <https://www.cloudiqtech.com/jenkins-pipeline-an-introduction/>

[3] *Gitlab CI/CD*. GitLab. (n.d.). Retrieved October 26, 2022, from <https://docs.gitlab.com/ee/ci/>

[4] Stephan Avenwedde (Correspondent) January 31, Stephan Avenwedde (Correspondent), *How to set up a CI pipeline on Gitlab*. Opensource.com. Retrieved October 26, 2022, from <https://opensource.com/article/22/2/setup-ci-pipeline-gitlab>