Assignment 3

Carter Harms

SID: 12411025

Question 1

Compare and contrast at least three CI tools/platforms (e.g., Jenkins, GitHub Actions, GitLab CI, Azure DevOps, CircleCI).

**Comparing: Jenkins, Gitlab, and Bamboo**

|  |  |  |  |
| --- | --- | --- | --- |
| **Capability** | **Jenkins** | **Gitlab CI/CD** | **Bamboo** |
| **Pricing Model** | - Free and open source for individual users  - Enterprise costs for companies related to infrastructure and support  - Free built agent license pricing | - Free for individual users  - Premium model for $29/month with add-on options  - Enterprise costs for business | - Pricing based on agent subscriptions  - High cost at ~$1,300 / agent |
| **Ease of Setup** | - 3/5  - Requires Java V11 or V17  - Easy to install on most OS’s  - Complex configuration | - 4/5  - CI/CD is a built in feature | - 2/5  - Must create the VM and install agent component then define required tools |
| **Integration capability** | - Over 1,800 free plugins  - Older plugins can break setup  - Need to manually keep plugins up to date through updates | - Works best with GitLab ecosystem  - External integration require additional effort | - 200+ add-ons and possible to develop custom plug-ins  - Uses app links for integrations  - Have to manually define property with password field |
| **Key features** | - Open source  - Handles distributed builds  - Blue Ocean visualization  - Legacy UI  - Docker support  - Lacks version control and code review  - Strong community | - Modern user-friendly interface  - Unified platform dependent on GitLab  - All-in-one solution for streamlined workflows  - Simplified pipelines | - Modern and intuitive UI  - Docker support |
| **Best use cases** | - Building microservices  - Independent projects  - You need an open-source tool to integrate with wide range of plugins | - Paired with GitLab ecosystem  - Simplification of pipeline setup with auto DevOps | - Microservices  - Enterprise builds  - Collaboration and visibility  - Integration with JIRA |

References:

[Katalon](https://katalon.com/resources-center/blog/ci-cd-tools) - comparison metrics

[Saucelabs](https://saucelabs.com/resources/blog/jenkins-vs-bamboo-what-to-know-for-building-your-ci-cd-automated-pipeline) - Bamboo and Jenkins information

[BDCC Global](https://www.bdccglobal.com/blog/jenkins-vs-gitlab-vs-circleci-the-battle-of-ci-cd-tools/) - Gitlab and Jenkins comparison

Question 2

References:

Deliverables

Submit the following components:

1. **CI Pipeline Configuration**
   * Complete pipeline configuration file (Jenkinsfile, .github/workflows/ci.yml, etc.)
   * Pipeline must implement all 5 required tasks above
2. **Application Source Code**
   * Working application with minimum 3 endpoints/features
   * Unit tests with minimum 5 test cases
   * README file with project setup instructions
   * All necessary configuration files
3. **Pipeline Evidence**
   * Screenshots or links showing successful pipeline runs
   * Evidence that all 5 pipeline tasks execute successfully
   * Screenshots showing pipeline failures (e.g., failed tests, code quality issues)

REFERENCES

[Jenkins](https://www.jenkins.io/doc/pipeline/tour/hello-world/) - Jenkins Docs and example pipeline

[Github](https://gist.github.com/merikan/228cdb1893fca91f0663bab7b095757c) - Jenkinsfile examples

[DZone](https://dzone.com/articles/elevate-python-code-quality) - Breakdown on linting tools

[Dev OliverJumpertz](https://dev.to/oliverjumpertz/setting-up-jenkins-to-handle-github-pull-requests-5bjc) - How to use Jenkins to identify pull requests

[HTML Plugin](https://www.jenkins.io/blog/2016/07/01/html-publisher-plugin/) – How to use Jenkins’s HTML plugin

[Archbee](https://www.archbee.com/blog/readme-document-elements) - How to build a simple-app README

[Stackoverflow](https://stackoverflow.com/questions/67608446/declarative-jenkins-pipeline-syntax-for-pylint-or-flake8) - Jenkings Linting

Question 3

Write a short reflection (250-400 words) on:

* Challenges you faced while implementing your CI pipeline
* How CI might change your approach to software development
* One improvement you would make to your pipeline given more time

Jenkins is an incredibly useful and comprehensive tool. However, at times, I did find it a bit archaic and not very user friendly. The challenges I faced with Jenkins fall into three camps: the syntax of the Jenkinsfile, running tests and troubleshooting efficiently, and using particular Plugins. Learning how to write a Jenkinsile and creating all the build artifacts and generating the reports using the correct batch commands could sometimes be frustrating. Additionally, I accomplished troubleshooting by using Pushes to update the Jenkinsfile on the app – something I will not do again (and will instead change the script on the app itself). Finally, learning about all the particular plugins and what to use. Jenkins has so many plugins but it is the Paradox of Choice, where so many options creates uncertainty about what the best one too select is.

Building the proper CI process is hard to setup and get all the elements right. However, once that it is setup I could absolutely see the value in having flows like this setup across different environments. The ability to run automatic tests and print reports on errors is exceptionally useful and would produce more resilient and readable code. Especially the Stages page when the app is working and the Pipeline Steps and Overview pages when the app is not working. It was nice to not have to check the logs each time and I could instead get right to the root of the failure.

If given more time, one improvement I would make to my pipeline would be either to add more in-depth and UI-friendly user reports on errors when they were raised in future builds. This would be tied into the broader improvement I would make of getting more familiar with the different plugins. I heard about Blue Ocean and how that helps create visually appealing outputs but I didn’t take time to explore that. I think if I setup Jenkins more comprehensively as part of a long-term Dev project or my jobs there would be ways to make the automated testing and steps more effective.