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). Create a comparison table that includes:

* Pricing model
* Ease of setup
* Integration capabilities
* Key features
* Best use cases

|  |  |  |  |
| --- | --- | --- | --- |
| 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 | * 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:

<https://katalon.com/resources-center/blog/ci-cd-tools>

<https://saucelabs.com/resources/blog/jenkins-vs-bamboo-what-to-know-for-building-your-ci-cd-automated-pipeline>

https://www.bdccglobal.com/blog/jenkins-vs-gitlab-vs-circleci-the-battle-of-ci-cd-tools/

Question 2

Create a simple web application (you can use any technology stack you're comfortable with - Node.js, Python Flask, Java Spring Boot, etc.) that includes:

* Basic functionality (minimum 3 endpoints/features)
  + /overall dashboard (homepage – recently read, currently reading, reading goal)
  + /add new book – what contents to include (title, author, pages, 7 star review, date complete)
  + /delete book
  + /update – changes details on a book
* Unit tests (minimum 5 test cases)
  + 3 for adding new book
  + 2 for deleting book
  + 3 for updating book
* A README file with setup instructions
  + Create last

Reading List Manager

* App.py file

References:

- <https://www.archbee.com/blog/readme-document-elements>

- <https://stackoverflow.com/questions/67608446/declarative-jenkins-pipeline-syntax-for-pylint-or-flake8> - Jenkings Linting

CI Pipeline Implementation

Set up a CI pipeline using Jenkins (or another CI platform of your choice such as GitHub Actions, GitLab CI, etc.) that:

1. **Triggers appropriately**
   * Configure pipeline to run on push to main/master branch
   * Configure pipeline to run on pull requests targeting main/master
   * Ensure triggers are specific to avoid unnecessary runs
2. **Environment setup**
   * Check out the source code from the repository
   * Install and configure the correct runtime environment (Node.js, Python, Java, etc.)
   * Install project dependencies using appropriate package managers
   * Set up any required environment variables or configuration files
3. **Code quality checks**
   * Configure and run linting tools appropriate for your language (ESLint, flake8, Checkstyle, etc.)
   * Implement at least one code formatting check tool (Prettier, Black, etc.)
   * Configure quality gates that fail the pipeline if standards aren't met
   * Provide clear error messages and suggestions for fixing issues
4. **Testing**
   * Execute all unit tests using appropriate testing framework
   * Generate comprehensive test coverage reports (HTML, XML, or JSON format)
   * Set minimum coverage thresholds (e.g., 80%) that must be met
   * Ensure pipeline fails if any tests fail or coverage is below threshold
   * Include test result artifacts for later analysis
5. **Build process**
   * Compile/build the application using appropriate build tools
   * Verify that all build artifacts are generated successfully

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

<https://www.jenkins.io/doc/pipeline/tour/hello-world/>

<https://gist.github.com/merikan/228cdb1893fca91f0663bab7b095757c> - Jenkinsfile examples

<https://dzone.com/articles/elevate-python-code-quality> - selection of Linters

<https://dev.to/oliverjumpertz/setting-up-jenkins-to-handle-github-pull-requests-5bjc> - pull requests locally

<https://www.jenkins.io/blog/2016/07/01/html-publisher-plugin/> - publish html

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
* PROBLEMS
* Configuration of Global Tools – how to connect with Git/already connected?
* Determining how to tie in Jenkins pipeline to the files within the repo
* Triggers – the nuance behind each of the options; setting up triggers on the public internet
* Learning the syntax of the Jenkins file and then keeping everything straight
* Pull requests efficiently

It’s crazy helpful it’s just a beast to setup properly and initialize.

Improvement: tinkering around with plugins – find one that would be applicable