TSU en Tecnologías de la Información Área Desarrollo de Software Multiplataforma

Nombre:

García Arreola Howard Isaí

Subject:

Software Development Process Management

Actividad:

Development environment for continuous integration

Grupo: 10-B

Profesor:

Ray Brunett Parra

Galaviz

Fecha de Realización:

January 7th, 2025

What is CI

It’s a software development practice where developers frequently merge their code changes into a shared repository, typically multiple times a day. Each integration is verified through an automated build and testing process to detect issues early.

What is a development environment

A development environment is a setup or workspace designed to enable developers to write, test, and debug code efficiently. It includes the tools, software, and configurations required to create and maintain applications during the development phase.

Considerations and best practices

Setting up an effective development environment for continuous integration (CI) is crucial for streamlining software development and ensuring high-quality code. Here are key considerations and best practices:

1. Version Control System (VCS):

Centralized Repository: Utilize a robust VCS like Git to maintain a central repository where all code changes are tracked. This facilitates collaboration and ensures that all team members work with the latest codebase.

2. Automated Build Processes:

Build Automation Tools: Implement tools that automate the process of compiling and building the codebase. This ensures consistency and reduces manual errors.

3. Automated Testing:

Unit and Integration Tests: Develop a comprehensive suite of automated tests to validate code changes. This helps in early detection of defects and maintains code quality.

4. Continuous Integration Servers:

CI Tools: Set up CI servers (e.g., Jenkins, Travis CI) to automatically run builds and tests whenever code is committed. This provides immediate feedback on the impact of changes.

5. Environment Configuration:

Consistent Development Environments: Use containerization tools like Docker to create consistent development and testing environments, minimizing discrepancies between different stages of development.

6. Continuous Deployment Pipelines:

Automated Deployment: Establish pipelines that automate the deployment process to various environments (e.g., staging, production), ensuring that code changes can be delivered efficiently and reliably.

7. Monitoring and Feedback:

Real-time Monitoring: Implement monitoring tools to track the health and performance of applications post-deployment, allowing for proactive issue resolution.

Continuous Delivery (CD)

Continuous Delivery (CD) is a software development practice in which code changes are automatically built, tested, and prepared for release to a production environment. It is an extension of Continuous Integration (CI), ensuring that new code can be reliably deployed at any time with minimal manual intervention.

References:

Fowler, M. (n.d.). *Continuous integration*. martinfowler.com. https://martinfowler.com/articles/continuousIntegration.html?utm_source=chatgpt.com

Alipour, R. (2023, October 1). *A simple local development environment with a complete CI/CD pipeline*. DEV Community. https://dev.to/rapour/a-simple-localdevelopment-environment-with-a-complete-cicd-pipeline-35pg?utm_source=chatgpt.com

Team, C. O. (2024, December 11). *CI/CD Concepts*. Codefresh. https://codefresh.io/learn/ci-cd/