Continuous Improvement in Software Releases

Continuous Improvement in Software Releases

Naga Bhushanam Rangisetty

Presenter Designation

Monthly Release Cycle Overview

1

Monthly Updates

Every month, we release updates that include new initiatives, features, and bug fixes.

2

Continuous Improvement

We are consistently working on improving the application through the addition of new functionalities.

3

Bug Fixing

Part of our monthly cycle involves addressing and resolving bugs to enhance application performance.

4

Automated Testing

We automate testing for new changes to ensure that all updates run smoothly and maintain application integrity.

Integration Challenges with Other Applications

Impact of System Changes

Changes in inbound and outbound applications can affect our application, necessitating careful monitoring.

Data Flow Management

Ensuring proper data flow between integrated systems is crucial to maintain functionality.

Preventing Breakage

We must check that modifications in other systems do not disrupt our application's performance.

Test Automation with Selenium and BDD

Utilization of Selenium

We use Selenium for test automation, allowing for efficient testing of web applications.

Integration with BDD Framework

Selenium is integrated with a BDD framework to enhance the testing process.

Automatic Testing of Features

This approach enables automatic testing of new features and changes.

Ensuring Functionality

Automated tests ensure that new changes work correctly and do not disrupt existing functionality.

Continuous Integration Using Jenkins

Continuous Integration (CI)

Continuous Integration is a development practice where code changes are automatically tested.

Role of Jenkins

Jenkins is used to manage changes and ensure that any updates in code are continuously integrated.

Automated Testing

Jenkins automatically runs tests on the updated code, helping to identify issues early.

Early Issue Detection

Catching issues early prevents them from escalating into larger problems.

Continuous Deployment Process with XLR

Transition to Continuous Deployment

Once everything is tested and works well, we move into the Continuous Deployment (CD) phase.

Utilization of XLR

We use XLR for this process, which automatically packages and deploys the software.

Software Readiness

XLR ensures that the software is ready for release at the end of the month.

Reducing Manual Work with Automation Tools

1	**Automation Tools Utilized**	We are using automation tools such as Selenium, Jenkins, and XLR to streamline our processes.
2	**Reduction in Manual Work**	These tools significantly reduce the amount of manual work required for updates.
3	**Thorough Testing**	Every update, whether it's a new feature or a bug fix, is thoroughly tested.
4	**Smooth Integration**	Automation ensures that updates are integrated smoothly into the system.

Efficiency in Building, Testing, and Deployment

1. **Enhanced Efficiency with CI and CD**

   Continuous Integration (CI) and Continuous Deployment (CD) enable more efficient building, testing, and deployment of changes.

2. **Support for Monthly Initiatives**

   The CI/CD process allows for effective management of changes, even with monthly initiatives.

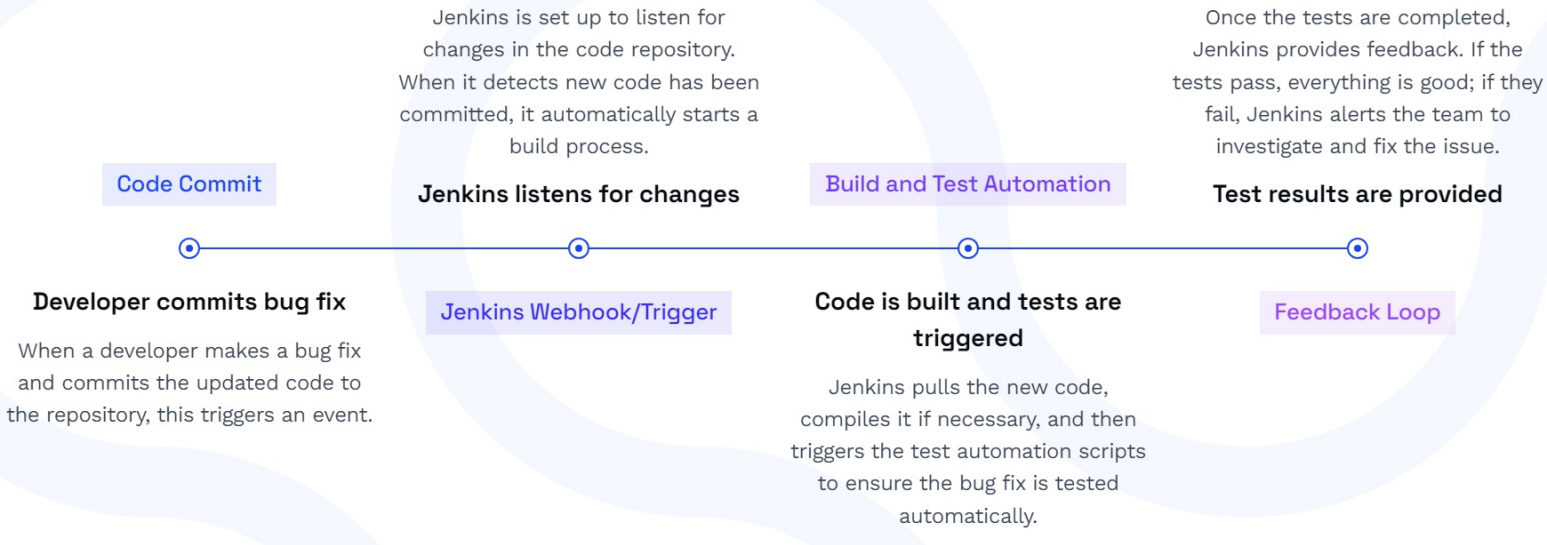
3. **Stability Maintenance**

   CI and CD practices help maintain stability throughout the development and deployment processes.

4. **Smooth Data Flow**

   Ensures a seamless flow of data between integrated applications, enhancing overall operational efficiency.

Jenkins Automation for Bug Fix Deployments



Handling Separate Git Repositories in Jenkins

Integration of Jenkins with Developer Repository

Jenkins can be configured to listen to the developer's code repository, not just the test automation repository.

Notification of Code Commits

When a developer commits new code (e.g., a bug fix), Jenkins is notified through a webhook or polling.

Triggering Test Automation Pipeline

Upon detecting a new code commit, Jenkins triggers the automation test pipeline stored in a different test automation repository.

Pulling Latest Automation Scripts

Jenkins pulls the latest version of the automation scripts from the test automation code repository to run against the updated application code.

End-to-End Jenkins Workflow

The workflow involves developers committing code, Jenkins detecting the commit, triggering the test pipeline, and executing tests on the new code.

Reporting Test Results

After testing, Jenkins reports back the results indicating whether tests passed or failed.

Seamless Workflow for Teams

This setup allows developer and test automation teams to maintain independent workflows while ensuring automated testing of new application changes.

Integrating Test Automation Repository with Jenkins

Configuration Details for Seamless Integration

1

Test

Provide the repository URL, branch information,

2

Build or Test

Specify the command(s) necessary to run the test

3

Test Results

Indicate where the test results will be generated so

4

Build and Test

List any dependencies or setup instructions

5

Environment

Provide a list of any specific environment variables

6

Test Execution

Specify parameters or arguments required for

7

Test Report

Detail the configuration for any test reporting

8

Post-Build

Outline any additional tasks Jenkins should perform

8

Key Configuration Areas

Eight key areas of configuration are essential for integrating the test automation repository with Jenkins.