Placement Empowerment Program

Cloud Computing and DevOps Centre

Use Git Hooks for Automation Create a Git hook that automatically runs tests before committing code to your repository.

Name: ERIC JEEVAN A Department: ADS

Using Git Hooks for Automation

Introduction

Git hooks are scripts that run automatically at certain points in the Git workflow. They allow developers to automate tasks such as running tests, checking code style, or preventing incorrect commits. In this guide, we will set up a **pre-commit** Git hook to automatically run tests before committing code. This helps ensure that only tested and validated code is committed to the repository.

Overview

Git hooks provide an effective way to enforce project standards and improve workflow automation. Pre-commit hooks, in particular, execute scripts before a commit is finalized, allowing developers to catch issues early. These hooks can be customized to run tests, enforce linting, or even prevent commits that contain sensitive data.

Objectives

By the end of this guide, you will:

- Understand the purpose and function of Git hooks.
- Learn how to create and configure a **pre-commit** hook.
- Automate code testing to prevent faulty commits.
- Gain knowledge on how to customize hooks for various project needs.

Expected Outcomes

After following this guide, you will be able to:

- Implement a **pre-commit** hook that runs tests before commits.
- Prevent commits if tests fail, ensuring better code quality.
- Extend hooks for additional automation tasks such as linting or formatting enforcement.

- Improve the overall software development workflow within a Git repository.

Prerequisites

- Basic understanding of Git
- A Git repository initialized (git init)
- A testing framework installed (e.g., pytest for Python projects)

Steps to Set Up a Pre-commit Hook

Step 1: Navigate to the Git Hooks Directory

Each Git repository has a .git/hooks/ directory where hooks are stored. Navigate to this directory using the terminal:

```
cd your-repo/.git/hooks/
```

Step 2: Create a Pre-commit Hook Script

Create a new file named pre-commit in the .git/hooks/ directory:

```
touch pre-commit
chmod +x pre-commit  # Make the script executable
```

Step 3: Write the Pre-commit Script

Edit the pre-commit file using a text editor:

```
nano pre-commit
```

Add the following script to run tests before a commit:

```
#!/bin/sh

# Run tests using pytest
if ! pytest; then
    echo "Tests failed. Commit aborted!"
    exit 1
fi

echo "All tests passed. Proceeding with commit."
exit 0
```

Save and exit the file (CTRL + X, then Y, and ENTER in nano editor).

Step 4: Test the Hook

1. Make some changes to your code.
2. Try committing them:
3. git add .
4. git commit -m "Test commit"
5. If tests fail, the commit will be aborted.
6. If tests pass, the commit will proceed successfully.

Conclusion

Setting up Git hooks can help automate testing and enforce code quality before committing changes. You can customize hooks for various checks like linting, security scanning, or formatting enforcement. By implementing pre-commit hooks, you ensure that only high-quality code is committed, improving collaboration and maintainability within your development team.

Reference

Git Hooks Guide