Software Build Automation Tools

Laboratory Exercises 1

Subject
Automation of Version Control Processes in GIT

Author:
PhD Hubert Zarzycki

The scope of laboratories

This laboratory introduces the fundamental concepts of GIT and its automation capabilities. By completing these tasks, one can gain practical experience in version control, branching, resolving conflicts, and implementing automation tools to streamline software development workflows.

Introduction to the GIT Version Control System

Version control systems (VCS) are essential tools in modern software development, allowing teams to track changes, collaborate effectively, and maintain code integrity. Among various VCS, GIT is one of the most widely used due to its distributed architecture, efficiency, and robustness. It enables developers to maintain different versions of code, manage branches for feature development, and seamlessly integrate changes into a shared repository.
GIT operates on a local-first principle, meaning every developer has a complete history of the project. This ensures that work is not lost even if remote repositories become unavailable. Additionally, GIT allows for parallel development by facilitating the creation of branches, which helps in isolating changes before merging them into the main codebase.

Installation and Configuration of GIT

1. Installing GIT

GIT can be installed on multiple operating systems:

- Windows: Download the installer from https://git-scm.com/download/win and follow the installation steps.
- Linux (Debian/Ubuntu):
  ```
  sudo apt update
  sudo apt install git
  ```
- macOS:
  ```
  brew install git
  ```

After installation, verify GIT by running:

```
git --version
```

2. Configuring GIT

Before using GIT, user information must be configured:

```
git config --global user.name "Your Name"
git config --global user.email "your.email@example.com"
```

These settings ensure that all commits are attributed correctly.

Creating and Managing Repositories

1. Initializing a Repository

A new repository can be created using:

```
git init my_project
cd my_project
```

This initializes a GIT repository in the specified folder.

2. Cloning an Existing Repository

To work on an existing project, clone the remote repository:

```
git clone https://github.com/username/repository.git
```

Branch Management in GIT

1. Creating a New Branch

Branching allows for parallel development without affecting the main branch.

```
git branch feature-branch
```

2. Switching to a Branch

```
git checkout feature-branch
```

Alternatively, the branch can be created and switched to simultaneously:

```
git checkout -b feature-branch
```

3. Merging Branches

After completing changes, merge the branch into the main branch:

```
git checkout main
git merge feature-branch
```

Handling Conflicts

When merging branches, conflicts may arise if two users modify the same lines in a file. GIT marks the conflict, allowing developers to resolve it manually.

```
git status
```

After resolving conflicts, the changes should be staged and committed:

```
git add resolved_file.txt
```

```
git commit -m "Resolved merge conflict"
```

Automating GIT Processes with Hooks and CI/CD

1. GIT Hooks

GIT provides hooks to automate tasks such as running tests or enforcing commit message formats. Hooks are stored in `.git/hooks/`.

Example of a pre-commit hook to prevent committing files with debugging statements:

```
#!/bin/sh
if grep -q "console.log" $(git diff --cached --name-only); then
  echo "Error: Remove console.log statements before committing."
  exit 1
fi
```

Save this script as `.git/hooks/pre-commit` and make it executable:

```
chmod +x .git/hooks/pre-commit
```

2. Continuous Integration and Deployment (CI/CD)

CI/CD automates code integration, testing, and deployment. Services like GitHub Actions, GitLab CI/CD, and Jenkins provide automated workflows.

Example GitHub Actions workflow (`.github/workflows/main.yml`):

```
name: CI
on: [push]

jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout code
        uses: actions/checkout@v2
      - name: Set up Node.js
        uses: actions/setup-node@v2
        with:
          node-version: '14'
      - name: Install dependencies
        run: npm install
      - name: Run tests
        run: npm test
```

This workflow runs tests on every push, ensuring code quality.

Tasks

1. Initialize a GIT Repository and Commit Changes
   - Create a new directory and initialize a GIT repository.
   - Create a `README.md` file with a short project description.
   - Stage and commit the file.

- Modify the file and commit the changes.

2. Work with Branches
   - Create a new branch named `feature-branch`.
   - Modify `README.md` by adding a new section.
   - Commit the changes and merge `feature-branch` into `main`.
   - Push the changes to a remote repository (GitHub).
3. Handle Merge Conflicts
   - Clone an existing repository.
   - Create two separate branches (`branch1` and `branch2`).
   - Modify the same line in `README.md` in both branches and commit the changes.
   - Merge `branch1` into `main`, then attempt to merge `branch2`.
   - Resolve the conflict manually and commit the resolution.
4. Implement a GIT Hook
   - Create a pre-commit hook that prevents committing `.log` files.
   - Test the hook by attempting to commit a `.log` file.
5. Set Up a CI/CD Pipeline
   - Configure a GitHub Actions workflow to run tests on push events.
   - Push the changes and verify that the workflow runs successfully.

6. Please submit to the e-learning platform a short report summarizing the work, including challenges faced and solutions applied. Include a link to the GitHub repository.

Literature

Loeliger, J., & McCullough, M. (2012). *Version Control with Git: Powerful tools and techniques for collaborative software development.* O'Reilly Media.
Chacon, S., & Straub, B. (2014). *Pro Git.* Apress.
Rubalcaba, C. (2021). Git for Programmers*: Master Git for Effective Development and Deployment.* Apress.
GIT Documentation - *https://git-scm.com/doc*
GitHub Learning Lab - *https://lab.github.com/*
Atlassian GIT Tutorials - *https://www.atlassian.com/git/tutorials*
GitHub Actions Documentation - *https://docs.github.com/en/actions*
Jenkins CI/CD Documentation - *https://www.jenkins.io/doc/*