Tutorial Report

Runpeng Li
rli3@oxy.edu
Occidental College

1 Introduction to Git

Git is a distributed version control system essential for developers. It helps track changes in your code, allowing multiple people to work together on the same project efficiently. Version control systems like Git are fundamental in software development because they help manage different versions of project files and allow you to revert to earlier versions if needed.

2 What is Git?

Git is not just a tool but a solution to common problems in collaborative coding projects. It ensures that the history of your project is well-organized and accessible, enabling seamless teamwork. Using Git, you can track each change made to the code, who made it, and why. This clarity helps prevent conflicts and eases the process of integrating different parts of a project.

2.1 Installation and Configuration

Install Git: Download and install Git from the official website or use package managers like Homebrew (for macOS) or Chocolatey (for Windows).

Configure Git: Set up your identity with Git using the following commands:

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

2.2 Creating a Repository

A repository or "repo" is where your project's files and their history are stored.

Initialize a New Repository: To start tracking a new project with Git, navigate to your project folder and run:

```
git init
```

Cloning an Existing Repository: To work on an existing project, clone it using:

```
git clone <repository_url>
```

2.3 Basic Workflow

Understanding the Git workflow is key to managing your code effectively.

Adding Changes: Use the following command to select changes you want to commit:

```
git add <file1> <file2>
```

Committing Changes: Save your changes with a meaningful message using:

```
git commit -m "Add feature X"
```

Pushing and Pulling: Share your commits with the remote repository and update your local repository using:

```
git push origin <branch_name>
git pull origin <branch_name>
```

3 Branches and Merging

Branches allow you to work on different versions of your project simultaneously.

Creating a Branch: Start a new feature or fix a bug without affecting the main project using:

```
git checkout -b feature/my-feature
```

Switching Branches: Switch your working branch with:

```
git checkout <branch_name>
```

Merging Branches: Combine changes from one branch to another with:

```
git merge <source_branch>
```

Handling Conflicts: Sometimes merging can cause conflicts. Resolve these manually, then commit the resolved changes.

4 Stashing

Stashing is useful for saving your uncommitted changes temporarily without committing them to the branch.

Stash Changes: Save your uncommitted changes temporarily with:

```
git stash save "Work in progress"
```

Apply Stash: Retrieve your stashed changes with:

```
git stash apply
```

4.1 Additional Resources

GitHub: Explore GitHub for hosting your repositories and collaborating with others. Practice: Create a sample project, experiment with branches, and practice merging to gain confidence and proficiency.

5 Tips and Advice

With this guide, beginners can understand the purpose and basic operations of Git, setting a strong foundation for more advanced practices in version control and collaborative software development.