"Heaven's Light is Our Guide"

Rajshahi University of Engineering & Technology, Rajshahi

Lab Report

Title: Basic Git Commands

Name: Sirajum Munir
Roll: 2010013
Lab Report Number: 01
Submission Date: 23-06-2024
Course Code: ECE 3113
Department: Department of Electrical & Computer Engineering (ECE)
University: Rajshahi University of Engineering & Technology (RUET)

Submitted To:
Oishi Joyti
Lecturer
Department of Electrical & Computer Engineering (ECE)
Rajshahi University of Engineering & Technology (RUET)

Git Command Reference Guide

> This document provides a comprehensive overview of essential Git commands along with examples and descriptions. Suitable for beginners and experienced developers alike, this guide facilitates easy navigation through Git functionalities.

Table of Contents

Introduction to Git

> Git is a distributed version control system designed to handle everything from small to very large projects with speed and efficiency. It allows multiple developers to work on a project simultaneously without interfering with each other's work.

Basic Git Commands

git init

> Initializes a new Git repository in the current directory.

```
git init
```

🎉 Example: If we want to start a new project in an empty directory. Simply we run `git init` to create a new Git repository.

git clone

> Creates a copy of an existing Git repository from a remote source.

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

📥 Example: Cloning a repository from GitHub to local machine so we can start working on it.

git status

> Displays the state of the working directory and the staging area. Shows which changes have been staged, which haven't, and which files aren't being tracked by Git.

```
git status
```

🔍 Example: After making changes to your files, run `git status` to see which files are staged for commit and which are not.

git add

> Adds changes in the working directory to the staging area.

```
git add filename
# or to add all changes
git add .
```

➕ Example: After modifying a file, use git add to stage it for the next commit. Use `git add .` to stage all changes.

git commit

> Records changes to the repository with a descriptive message.

```
git commit -m "Your commit message here"
```

📝 Example: Once changes are staged, commit is done with a descriptive message like `git commit -m "Added new feature"`.

git push

> Uploads local repository content to a remote repository.

```
git push origin main
```

🚀 Example: After committing your changes, push them to the remote repository using `git push`.

git pull

> Fetches from and integrates with another repository or a local branch.

```
git pull origin main
```

⬇️ Example: Before starting new work, pull the latest changes from the remote repository to ensure local copy is up-to-date.

Branching and Merging

git branch

> Lists, creates, or deletes branches.

```
# List all branches
git branch

# Create a new branch
git branch new-feature

# Delete a branch
git branch -d old-branch
```

🌿 Example: Use branches to develop features in isolation. Create a new branch with `git branch new-feature`.

git checkout

> Switches branches or restores working tree files.

```
# Switch to a branch
git checkout new-feature

# Create and switch to a new branch
git checkout -b new-feature
```

🔄 Example: Switch to the new-feature branch using `git checkout new-feature`.

git merge

> Joins two or more development histories together.

```
# Merge branch into the current branch
git merge feature-branch
```

🔗 Example: Merge changes from feature-branch into current branch with `git merge`.

Advanced Git Commands

git stash

> Temporarily stores all modified tracked files.

```
# Save changes to stash
git stash

# Apply stashed changes
git stash apply
```

📦 Example: Stash your current changes with `git stash` if need to switch branches without committing.

git log

> Shows the commit logs.

```
git log
```

📜 Example: View the commit history of repository with `git log`.

git reset

> Resets the current HEAD to the specified state.

```
# Soft reset (keeps changes in the working directory)
git reset --soft HEAD~1

# Mixed reset (keeps changes in the working directory but unstages them)
git reset --mixed HEAD~1

# Hard reset (discards changes in the working directory)
git reset --hard HEAD~1
```

♻️ Example: Undo your last commit but keep the changes with `git reset --soft HEAD~1`.

git revert

> Creates a new commit that undoes the changes made by an earlier commit.

```
git revert commit_id
```

🔙 Example: Revert changes introduced by a specific commit without rewriting history using `git revert`.

git rebase

> Reapplies commits on top of another base tip.

```
git rebase branch_name
```

🔄 Example: Rebase feature branch onto the latest commit of the main branch to keep a linear history.

Collaboration Commands

git remote

> Manages set of tracked repositories.

```
# Add a new remote
git remote add origin https://github.com/username/repository.git

# List all remotes
git remote -v
```

🌐 Example: Add a new remote repository with `git remote add origin`.

git fetch

> Downloads objects and refs from another repository.

```
git fetch origin
```

⬇️ Example: Fetch the latest changes from the remote repository without merging them with `git fetch`.

git pull request

> Creates a pull request to merge changes.

Note: This command is typically done through platforms like GitHub, GitLab, or Bitbucket.

```
# Example command for GitHub CLI
gh pr create --base main --head feature-branch --title "Feature Title" --body
"Description of the feature"
```

🔀 Example: Create a pull request to propose changes for review using `gh pr create`.

Git Configuration

git config

> Sets configuration values for your Git installation.

```
# Set user name
git config --global user.name "Sirajum Munir"

# Set user email
git config --global user.email "2010013@student.ruet.ac.bd"

# List all settings
git config --list
```

⚙️ Example: Configure your Git user name and email with `git config --global user.name "Your Name"`.