"Heaven's Light is our Guide."

Rajshahi University of Engineering & Technology, Rajshahi

Lab Report

Title: Study of different Git Commands

Name: Humaira Tasnim Adiba

Roll Number: 2010002

Lab Session: 02

Submission Date: October 20, 2024

Course Code: ECE 3113

Department: Electrical & Computer Engineering (ECE)
University: Rajshahi University of Engineering & Technology (RUET)

Submitted To:
Instructor: Oishi Joyti
Position: Assistant Professor
Department: Electrical & Computer Engineering (ECE)
University: Rajshahi University of Engineering & Technology (RUET)

📖 Git Command Quick Reference Guide

This guide offers a complete overview of the most important Git commands, with examples for each. It's perfect for both beginners and experienced developers looking for quick access to essential Git operations.

✨ What is Git?

> Git is a powerful distributed version control system that can manage everything from small projects to enterprise-level repositories. It allows multiple developers to collaborate simultaneously without conflicts.

⚡ Basic Git Commands

◇ git init

Initializes a new Git repository in the current directory.

```
git init
```

Example: Starting a new project in a folder? Run `git init` to create a Git repository.

◇ git clone

Creates a local copy of a remote repository.

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

Example: Clone a GitHub repository to your local machine using `git clone` to start contributing.

◇ git status

Shows the current status of your working directory and staging area.

```
git status
```

Example: After editing files, check which changes have been staged or not by running `git status`.

◇ git add

Stages changes in the working directory for the next commit.

```
git add filename
# Add all changes
git add .
```

Example: Use `git add .` to stage all changes made to the files before committing.

◇ git commit

Commits the staged changes with a message.

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

Example: Once changes are staged, commit them with a meaningful message like `git commit -m "Added new feature"`.

◇ git push

Pushes your committed changes to a remote repository.

```
git push origin main
```

Example: After committing, push your changes to GitHub using `git push`.

◇ git pull

Pulls changes from a remote repository and integrates them with your local branch.

```
git pull origin main
```

Example: Keep your local branch updated by pulling the latest changes from the remote repository with `git pull`.

🌿 Branching & Merging

◇ git branch

Lists, creates, or deletes branches.

```
# List all branches
git branch

# Create a new branch
git branch feature-branch

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

Example: Use `git branch new-feature` to create a branch dedicated to developing a new feature.

◇ git checkout

Switches to a different branch.

```
# Switch to a branch
git checkout branch-name

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

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

◇ git merge

Merges changes from one branch into another.

```
git merge feature-branch
```

Example: Integrate changes from the `feature-branch` into your current branch using `git merge`.

🔧 Advanced Git Commands

◇ git stash

Temporarily stores your uncommitted changes.

```
git stash
git stash apply
```

Example: Use `git stash` to save your progress without committing, so you can switch branches without losing work.

◇ git log

Shows the commit history.

```
git log
```

Example: Use `git log` to view all previous commits and their messages.

◇ git reset

Resets the current HEAD to a previous state.

```
# Soft reset (keeps changes)
git reset --soft HEAD~1

# Hard reset (discards changes)
git reset --hard HEAD~1
```

Example: Undo your last commit but keep changes with `git reset --soft`.

◇ git revert

Reverts a previous commit without modifying the commit history.

```
git revert commit_id
```

Example: Revert a specific commit using `git revert` if it introduced an error.

◇ git rebase

Reapplies commits on top of another base tip.

```
git rebase branch-name
```

Example: Rebase your feature branch onto the main branch to keep a cleaner history.

🌐 Collaborating with Git

◇ git remote

Manages connections to remote repositories.

```
git remote add origin https://github.com/username/repository.git
git remote -v
```

Example: Add a new remote repository to push or pull changes with `git remote add origin`.

◇ git fetch

Fetches changes from a remote repository without merging them.

```
git fetch origin
```

Example: Get the latest changes from the remote repository using `git fetch`.

◇ git pull request

Creates a pull request for code review.
Note: Typically done via platforms like GitHub or GitLab.

```
# GitHub CLI Example
gh pr create --base main --head feature-branch --title "New Feature" --body
"Feature description"
```

Example: Create a pull request for team review using the GitHub CLI.

⚙ Git Configuration

◇ git config

Sets user preferences for your Git installation.

```
# Set user details
git config --global user.name "adibaruet"
git config --global user.email "adibaruet@gmail.com"

# Check current config
git config --list
```

Example: Configure your Git username and email using `git config --global`.
