Heaven's light is our guide

Department of Electrical & Computer Engineering

LAB REPORT

Course Title
Software Engineering & Information System Design Sessional

Course Code: ECE 3118

Submitted By
Md. Nazat Kabir
Roll: 2010026
Session: 2020-21

Experiment Date: 21.04.2024
Submission Date: 15.11.2024

Submitted To
Oishi Jyoti
Assistant Professor
ECE-RUET

Experiment No. 3
Experiment Name: Study of Different Git Commands

Objectives:

- To understand the purpose and usage of Git commands.
- To learn how to use Git for version control and collaboration.
- To explore different Git commands for managing repositories.

Theory:

Git is a distributed version control system that tracks changes in source code during software development. It allows multiple developers to work on the same project without overwriting each other's contributions. Key concepts in Git include repositories, commits, branches, and merges. This experiment focuses on studying different Git commands, their uses, and how they contribute to efficient version control.

Key Git Commands:

1. `git init`: Initializes a new Git repository.
2. `git clone`: Clones an existing repository into a new directory.
3. `git status`: Displays the state of the working directory and the staging area.
4. `git add`: Adds changes to the staging area for the next commit.
5. `git commit`: Records the changes in the repository.
6. `git log`: Shows the commit history of the repository.
7. `git push`: Uploads local repository content to a remote repository.
8. `git pull`: Fetches changes from a remote repository and merges them into the local repository.
9. `git branch`: Lists, creates, or deletes branches in a repository.

10. `git merge`: Merges changes from one branch into another.

Procedure:

1. Initialize a Git repository:

- Use `git init` to initialize a new Git repository in the current directory.
- Example:

```
git init
```

2. Clone an existing repository:

- Use `git clone` to clone a remote repository.
- Example:

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

3. Check repository status:

- Use `git status` to check the current status of the repository.
- Example:

```
git status
```

4. Add files to the staging area:

- Use `git add` to add specific files or all changes to the staging area.
- Example:

```
git add filename.txt
git add .
```

5. Commit changes:

- Use `git commit` to save changes to the repository with a message.
- Example:

```
git commit -m "Initial commit"
```

6. View commit history:

- Use `git log` to view the commit history.
- Example:

```
git log
```

7. Push changes to a remote repository:

- Use `git push` to upload changes to a remote repository (like GitHub).
- Example:

```
git push origin main
```

8. Pull updates from a remote repository:

- Use `git pull` to fetch and merge changes from the remote repository.
- Example:

```
git pull origin main
```

9. Branch management:

- Use `git branch` to create and list branches.
- Example:

```
git branch new-feature
git checkout new-feature
```

10. Merge branches:

- Use `git merge` to merge changes from one branch into another.
- Example:

```
git checkout main
git merge new-feature
```

Source Code and Output:

Commands:

```
# Initialize a new repository
git init

# Clone a repository
git clone https://github.com/username/repository.git

# Check status
git status

# Add files
git add .

# Commit changes
git commit -m "Initial commit"

# Push changes
git push origin main

# Pull updates
git pull origin main

# Create a new branch
git branch feature-branch

# Merge branches
git merge feature-branch
```

Discussion:

In this experiment, we studied key Git commands essential for managing code.

1. **Repository Setup**: We used `git init` to start a new

repository and `git clone` to copy an existing one. This allows us to either begin a new project or work with an existing codebase.

2. **Tracking and Staging:** `git status` helps check the current state of our files, and `git add` stages changes for the next commit. This keeps our work organized and ready for version control.

3. **Committing and Sharing:** With `git commit`, we save our changes with a message. `git push` uploads these changes to a remote repository, while `git pull` updates our local copy with changes from others.

4. **Branching:** We used `git branch` to manage different lines of development and `git merge` to combine changes from different branches.

Overall, these commands help us keep track of changes, collaborate with others, and manage different versions of our code efficiently.