Assignment

Submitted to:
Sona Maria Sebastian

Submitted By:

Vidhu Krishnan Vinod

Roll No:58

Summary

Introduction

Git is a distributed version control system used to track changes in source code during software development. It allows multiple developers to collaborate on a project and keep track of changes made to the codebase over time. In this response, we will cover the basic steps for installing Git on a Windows or Mac computer and give a brief introduction to using Git.

Throughout the course various Git commands are taught, such as git clone, git add, git commit, git push, and git pull, and how to manage branches.It also shows how to collaborate with other developers using Git, including how to merge changes and handle conflicts.

Chapter wise commands learnt

Chapter 1: Git Introduction and Installation

Introduction to version control and Git

Installing Git on different operating systems:

- Windows: https://git-scm.com/download/win
- Mac: https://git-scm.com/download/mac
- Linux: sudo apt-get install git

Chapter 2: Working with Staging area and Snapshots

Adding and removing files from the staging area:

- Add all files: git add .
- Remove a file from staging area: git reset filename
- Making commits to the repository: git commit -m "commit message"

Viewing commit history and changes:

- View commit history: git log

- View changes made in a file: git diff filename

Once Git is installed on your computer, you can use the Git command line interface or a Git client such as GitHub Desktop to interact with Git.

Creating a new repository

To create a new Git repository on your local machine, open a terminal window and navigate to the directory where you want to create the repository. Then run the command **git init**. This will initialize a new Git repository in the current directory.

Adding files to a repository

Once you have created a Git repository, you can add files to it by running the command **git add <file>**. This will stage the file for commit.

To commit changes to the repository, run the command **git commit -m "Commit message"**. This will create a new commit with the changes you have made.

Chapter 3: Branching and Merging

Creating and switching branches:

- Create a new branch: git branch branch-name
- Switch to a branch: git checkout branch-name
- Merging changes from one branch to another: git merge branch-name
- Handling merge conflicts: git merge tool

Chapter 4: Remote Repository

- Cloning a remote repository: git clone remote-repository-url
- Pushing and pulling changes from a remote repository:
- Push changes: git push origin branch-name
- Pull changes: git pull origin branch-name
- Resolving conflicts with remote changes: git merge tool

"git fetch": This command downloads new changes from a remote repository but does not merge them into your local branch. It is useful for getting an overview of changes made by other developers without modifying your own work. The syntax of this command is "git fetch [remote]".

"git push": This command uploads your local changes to a remote repository. It is used to share your work with other developers and to save changes made to the repository. The syntax of this command is "git push [remote] [branch]".

"git pull": This command downloads new changes from a remote repository and merges them into your local branch. It is used to update your local repository with changes made by other developers. The syntax of this command is "git pull [remote] [branch]".

Chapter 5: Other Topics (fork, stash)

Fork

- Clone the forked repository: git clone forked-repo-url
- Make changes and push to forked repository: git push origin branch-name
- Create pull request on GitHub website

Stash

- Using Git stash to temporarily store changes:
- Stash changes: git stash save "stash message"
- Apply stashed changes: git stash apply

“git stash" is a Git command used to temporarily save changes that are not ready to be committed or pushed to a remote repository. This is useful when you want to switch to a different branch or work on a different feature without losing your current work.

To use "git stash", open your terminal or Git bash and navigate to the repository where you want to stash your changes. Then, run the command "git stash". This will save your changes in a stack of stashes, allowing you to revert to them later.

You can also add a message to the stash to make it easier to identify later by using the command "git stash save [message]". Replace [message] with a short description of the changes you are stashing.

Conclusion:

Git is a powerful tool that is essential for any software developer. This tutorial provides a basic introduction to Git and its core functionalities. By using Git, you can track changes, collaborate with others, and manage your codebase with ease.