**GIT Assignment 1**

1. What is a git stash list?

Ans:- In Git, a "stash" is a feature that allows you to temporarily save changes that you have made to your working directory, without committing them to your repository. A Git stash is useful when you need to switch to a different branch or work on a different task, but want to save your current changes.

The git stash list command is used to display a list of all the stashes that you have created in your repository. It shows the stash ID, the branch where the stash was created, and a description of the stash (if you provided one when creating the stash). The most recent stash is listed first, and the oldest stash is listed last.

Here's an example of the output of the git stash list command:

stash@{0}: On master: adding new feature

stash@{1}: On feature-branch: fixing bug

stash@{2}: On develop: refactoring code

In this example, there are three stashes in the repository, with IDs 0, 1, and 2. The first stash was created on the "master" branch, with a description of "adding new feature". The second stash was created on the "feature-branch" branch, with a description of "fixing bug". And the third stash was created on the "develop" branch, with a description of "refactoring code".

1. How do you get a list of all the files that have been updated in a given commit?

Ans:- To get a list of all the files that have been updated in a given commit in Git, you can use the git show command with the --name-only option. Here are the steps:

1.Open your terminal or Git Bash on your computer.

2.Navigate to the Git repository where you want to view the updated files.

3.Type the following command: git show --name-only <commit hash>

4.Replace <commit hash> with the hash of the commit you want to view. You can find the commit hash by running git log.

5.Press Enter to execute the command.

This will display a list of all the files that were updated in the specified commit.

The output will look something like this:

commit 3e7f83d1b1c7b0dc2e92c7c9d5fa5c5a5f5a5f5a

Author: John Smith <john@example.com>

Date: Fri Apr 2 12:34:56 2023 -0500

Updated README and main.py

README.md

src/main.py

In this example, the commit hash is 3e7f83d1b1c7b0dc2e92c7c9d5fa5c5a5f5a5f5a. The author, date, and commit message are displayed first, followed by a list of the updated files. The --name-only option ensures that only the file names are displayed, without any additional information.

1. What is a Git merge conflict?

Ans:- A Git merge conflict occurs when Git is unable to automatically merge changes from different branches of a codebase. This happens when two or more branches have changes that conflict with each other - that is, they modify the same file or lines of code in incompatible ways.

When a merge conflict occurs, Git will halt the merge process and indicate which files have conflicts. You will need to resolve these conflicts manually by editing the conflicting files and choosing which changes to keep. After resolving the conflicts, you will need to commit the changes to complete the merge.

It's important to resolve merge conflicts carefully and thoughtfully, to ensure that the merged code is correct, functional, and free of errors.

1. How do you distinguish between git fetch and git pull? How do you differentiate between Git Merge and Git Rebase?

Ans:- git fetch and git pull are both Git commands used to update your local repository with changes from a remote repository. However, they differ in how they handle the changes.

git fetch downloads the changes from the remote repository to your local repository, but does not automatically merge the changes into your local branch. It only updates the remote tracking branch in your local repository. This means that you can review the changes before merging them into your local branch using git merge or git rebase. You can think of git fetch as a "download-only" command.

git pull, on the other hand, is a combination of git fetch and git merge. It downloads the changes from the remote repository and immediately merges them into your local branch. This means that if there are any merge conflicts, they will need to be resolved at the time of the pull. You can think of git pull as a "download-and-merge" command.

git merge and git rebase are both Git commands used to integrate changes from one branch into another. However, they differ in how they apply the changes.

git merge creates a new commit that incorporates the changes from one branch into another. The commit history of both branches is preserved, and a merge commit is created to record the merge. This is the recommended way to merge branches in Git when there is a need to preserve the branch history.

git rebase, on the other hand, applies the changes from one branch onto another by re-writing the commit history. Instead of creating a new commit, it modifies the existing commits on the branch being rebased, and applies them on top of the branch you are rebasing onto. This results in a linear commit history, with no merge commits. This is useful when you want to create a clean, linear history, and do not need to preserve the original branch history. However, it should be used with caution, as it can result in conflicts and other issues if not used correctly.

1. What command uploads any GitHub repository to your computer using the git command?

Ans:- To upload a GitHub repository to your computer using Git, you need to use the git clone command. Here are the steps:

1.Open your terminal or Git Bash on your computer.

2.Navigate to the directory where you want to clone the repository.

3.Go to the GitHub repository you want to clone and copy the repository URL. You can find the URL on the repository page, under the "Clone or download" button.

4.Type the following command: git clone <repository URL>

Replace <repository URL> with the URL of the repository you want to clone.

5.Press Enter to execute the command.

Git will now download a copy of the repository to your local computer. You can make changes to the files, commit them, and push them back to the repository using the git add, git commit, and git push commands.

1. How do you write a commit message using the command? How do you push code in GitHub?

Ans:-To write a commit message using the command line, follow these steps:

Make sure you have made changes to the code in your local Git repository.

Use the git add command to stage the changes you want to commit. For example, to stage all changes in the repository, use git add .

Use the git commit command with the -m flag to add a commit message. For example, git commit -m "Added new feature"

Write a descriptive and concise commit message that explains the changes you have made in this commit. Make sure to follow any relevant guidelines or conventions for your project.

To push code to a GitHub repository, follow these steps:

Use the git add command to stage the changes you want to push.

Use the git commit command with a commit message to create a new commit with the changes you want to push.

Use the git push command to push the changes to the remote repository on GitHub. You may need to authenticate using your GitHub username and password or access token.

Here's an example of how to push code to a GitHub repository:

Make changes to the code in your local Git repository.

Stage the changes you want to commit using git add.

Create a new commit with a commit message using git commit -m "Added new feature".

Push the changes to the remote repository on GitHub using git push. For example, git push origin main.

Note that in the git push command, origin is the name of the remote repository and main is the name of the branch you want to push the changes to. You may need to replace these with different names depending on your repository configuration.

1. How do you make a Git repository?

Ans:- To make a Git repository, follow these steps:

Open a terminal or command prompt on your computer.

Navigate to the directory where you want to create the repository.

Use the git init command to create a new Git repository in that directory. For example, git init my-repository.

You can now add files to the repository by copying them into the repository directory or creating them using a text editor.

Use the git add command to stage the changes you want to commit. For example, git add my-file.txt.

Use the git commit command with a commit message to create a new commit with the changes you want to save. For example, git commit -m "Initial commit".

After completing these steps, you will have created a new Git repository in the specified directory and made your first commit with the initial changes. You can continue to make changes, add files, and create commits as needed to keep your repository up to date.

1. What is Git, and how does it work?

Ans:- Git is a free and open-source distributed version control system that helps developers track changes to their code and collaborate with others. It was created by Linus Torvalds in 2005 to manage the development of the Linux operating system.

Git works by keeping track of changes to files in a code repository. Each time a developer makes a change, Git creates a new snapshot of the entire repository. These snapshots are stored as "commits" in Git's history and are used to track changes over time.

Git allows developers to work on their own copies of a repository and then merge their changes back into the main repository when they're ready. This allows multiple developers to work on the same codebase at the same time, without overwriting each other's changes.

Git also includes features for branching and merging code. A branch is a copy of a repository at a specific point in time, which developers can work on independently without affecting the main repository. Once changes in a branch are complete, they can be merged back into the main repository using Git's merge or rebase features.

Git is designed to be lightweight, fast, and flexible, and is used by developers of all skill levels and backgrounds. It is widely used in the software development industry and is supported by a large and active community of contributors.