Experiment No :1

1. Setting up and Basic Commands

Initialize a new Git Repository in a directory. Create a new file and add it to the staging area and commit the changes with an appropriate commit message.

Step 1: Initialize a new Git Repository

- Open a terminal or command prompt.
- Create a Directory
  mkdir first
- Navigate to the directory where you want to create your new Git repository.
  cd path/to/your/directory
- Initialize a new Git repository.
  git init

Step 2: Create a New File

- Use any text editor of your choice to create a new file in the repository directory. For example, let's create a file named sample.txt.
  vim sample.txt
- Open sample.txt in your text editor and add some content.

Step 3: Add the File to the Staging Area

- Add the newly created file to the staging area.
  git add sample.txt
  This command stages the changes in sample.txt for the next commit.

Step 4: Commit the Changes

- Commit the changes to the repository with a meaningful commit message.
  git commit -m "Initial commit: Added sample.txt"

Experiment No :2

Creating and Managing Branches

Create a new branch named "feature-branch." Switch to the "master" branch. Merge the "feature-branch" into "master."

Step 1: Initialize a new Git Repository

- Open a terminal or command prompt.
- Create a Directory

  mkdir second
  - Navigate to the directory where you want to create your new Git repository.

    cd path/to/your/directory
  - Initialize a new Git repository.

    git init

Step 2: Create a new branch named "feature-branch"

- **git branch feature-branch**

  This command creates a new branch named "feature-branch".

Step 3: Switch to the feature-branch

- **git checkout feature-branch**

  This command switches to feature-branch from master branch.

Step 4: Make changes in the "feature-branch":

- Make the necessary changes to your code in the "feature-branch."
- Edit the file using the editor and make appropriate changes to the source code.

Step 5: Commit the changes in "feature-branch":

- **git add .**
- **git commit -m "Implement new feature in feature-branch"**

  This command stages and commits your changes in the "feature-branch."

Step 6: Switch back to the "master" branch:

- **git checkout master**

  This command switches back to the "master" branch.

Step 7: Update the "master" branch (optional):

- It's a good practice to ensure your "master" branch is up-to-date before merging changes. Fetch the latest changes from the remote repository:

  git pull origin master

Step 8: Merge "feature-branch" into "master":

- **git merge feature-branch**

  This command merges the changes from "feature-branch" into the "master" branch.

Experiment No: 3

Creating and Managing Branches

Write the commands to stash your changes, switch branches, and then apply the stashed changes.

In Git, the git stash command is used to temporarily save changes that are not ready to be committed but need to be set aside. This is helpful when you want to switch branches or perform some other operation that requires a clean working directory.

Step 1: Initialize a new Git Repository

- Open a terminal or command prompt.
- Create a Directory

  mkdir second
  - Navigate to the directory where you want to create your new Git repository.
    cd path/to/your/directory
  - Initialize a new Git repository.
    git init

Step 2: Create a new branch named "feature-branch"

- **git branch feature-branch**
  This command creates a new branch named "feature-branch".

Step 3: Switch to the feature-branch

- **git checkout feature-branch**
  This command switches to feature-branch from master branch.

Step 4: To stash your changes:

- **git stash save "Your stash message"**
  This command saves your local changes in a "stash" and provides an optional message describing the changes.

Step- 5: Switch branches:

- **git checkout <branch_name>**
  Replace <branch_name> with the name of the branch you want to switch to.

Step 6: Apply the stashed changes:

- **git stash apply**
  This command will apply the latest stash. If you have multiple stashes, you may need to specify the stash you want to apply by using git stash apply stash@{n}, where n is the index of the stash you want to apply.

Experiment No : 4

Collaboration and Remote Repositories

Clone a remote Git repository to your local machine

Step 1: Create the remote repository in Github

Step 2: Clone the repository
To clone a remote Git repository to your local machine, you can use the git clone command. Here's the basic syntax:

git clone <repository_url>

Replace **<repository_url>** with the URL of the Git repository you want to clone. For example:

git clone https://github.com/example/repository.git

This command will create a copy of the entire remote repository, including all branches and commit history, in a new directory on your local machine. The directory will have the same name as the repository.

Optional -

If you want to clone the repository into a specific directory, you can specify the destination directory after the repository URL:

git clone https://github.com/example/repository.git <destination_directory>

Experiment No : 5

Collaboration and Remote Repositories

Fetch the latest changes from a remote repository and rebase your local branch onto the updated remote branch.

To fetch the latest changes from a remote repository and rebase your local branch onto the updated remote branch, you can follow these steps:

Step 1: Fetch the latest changes from the remote repository:

git fetch

This command fetches the latest changes from the remote repository without merging them into your local branches.

Step 2: Rebase your local branch onto the updated remote branch:

git rebase origin/<branch_name>

Replace **<branch_name>** with the name of the branch you want to rebase. This command applies your local changes on top of the fetched changes from the remote branch.

Step 3: Rebase the changes

```
git add .            # Stage the resolved changes
git rebase --continue  # Continue with the rebase
```

Step 4: Push the rebased changes to the remote repository:

git push origin <branch_name> --force

Be cautious when using --force with git push as it overwrites the remote branch with your local changes.

Ubuntu Support for Authentication was removed on 2021

Download the Key from your github Account:

git remote set-url origin https:// KEY@github.com/username/repo