Git and GitHub Projects

Git Branching

Use a branch to isolate development work without affecting other branches in the repository. Each repository has one default branch and can have multiple other branches. You can merge a branch into another branch using a pull request.

Branches allow you to develop features, fix bugs, or safely experiment with new ideas in a contained area of your repository.

Git Revert and Reset

Two commonly used tools that git users will encounter are git reset and git revert. The benefit of both of these commands is that you can use them to **remove or edit changes** you’ve made in the code in previous commits.

Git Rebase and Merge

What Is Git Rebase?

Git rebase is a command that lets **users integrate changes from one branch to another**, and the logs are modified once the action is complete. Git rebase was developed to overcome merging’s shortcomings, specifically regarding logs.

What Is Git Merge?

Git merge is a command that allows **developers to merge Git branches** while the logs of commits on branches remain intact.

The merge wording can be confusing because we have two methods of merging branches, and one of those ways is actually called “merge,” even though both procedures do essentially the same thing.

Git Stash:

Git stash is a command that allows you to temporarily save changes you have made in your working directory, without committing them. This is useful when you need to switch to a different branch to work on something else, but you don’t want to commit the changes you’ve made in your current branch yet.

To use Git stash, you first create a new branch and make some changes to it. Then you can use the command git stash to save those changes. This will remove the changes from your working directory and record them in a new stash. You can apply these changes later. git stash list command shows the list of stashed changes.

You can also use git stash drop to delete a stash and git stash clear to delete all the stashes.

Cherry-pick:

Git cherry-pick is a command that allows you to select specific commits from one branch and apply them to another. This can be useful when you want to selectively apply changes that were made in one branch to another.

To use git cherry-pick, you first create two new branches and make some commits to them. Then you use the git cherry-pick <commit_hash> command to select the specific commits from one branch and apply them to the other.

Resolving Conflicts:

Conflicts can occur when you merge or rebase branches that have diverged, and you need to manually resolve the conflicts before git can proceed with the merge/rebase. git status command shows the files that have conflicts, the git diff command shows the difference between the conflicting versions and the git add command is used to add the resolved files.

Task 1: Clone/Download code in local, Do the changes and push the code to github repo

Steps to be follow

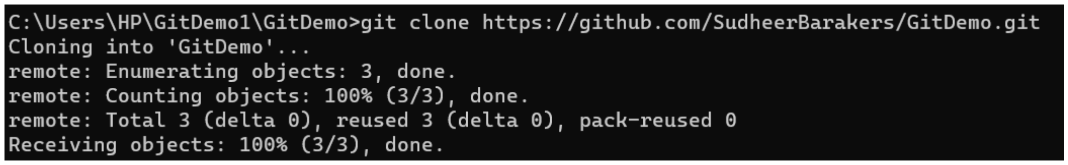
1. Install git on system (Windows/Linux)

    1.1. Download git on windows and install.

    1.2 Linux: Run the command – **sudo apt-get install git**

2. Clone the code from central repo (GitHub) to local repo

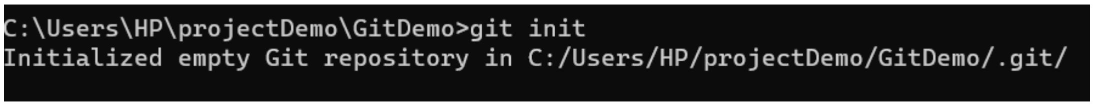
    $ git clone https://github.com/SudheerBarakers/GitDemo.git



```
C:\Users\HP\GitDemo1\GitDemo>git clone https://github.com/SudheerBarakers/GitDemo.git
Cloning into 'GitDemo'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 3 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.
```

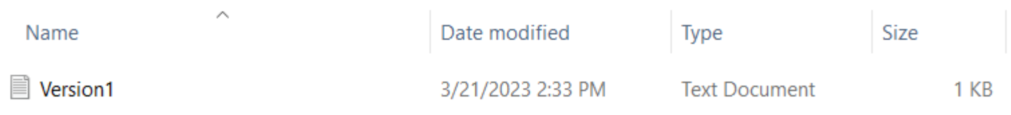
3. Make project folder into local repo by initializing git

    $ git init



```
C:\Users\HP\projectDemo\GitDemo>git init
Initialized empty Git repository in C:/Users/HP/projectDemo/GitDemo/.git/
```

4**.** Create a new file- **Version.txt**

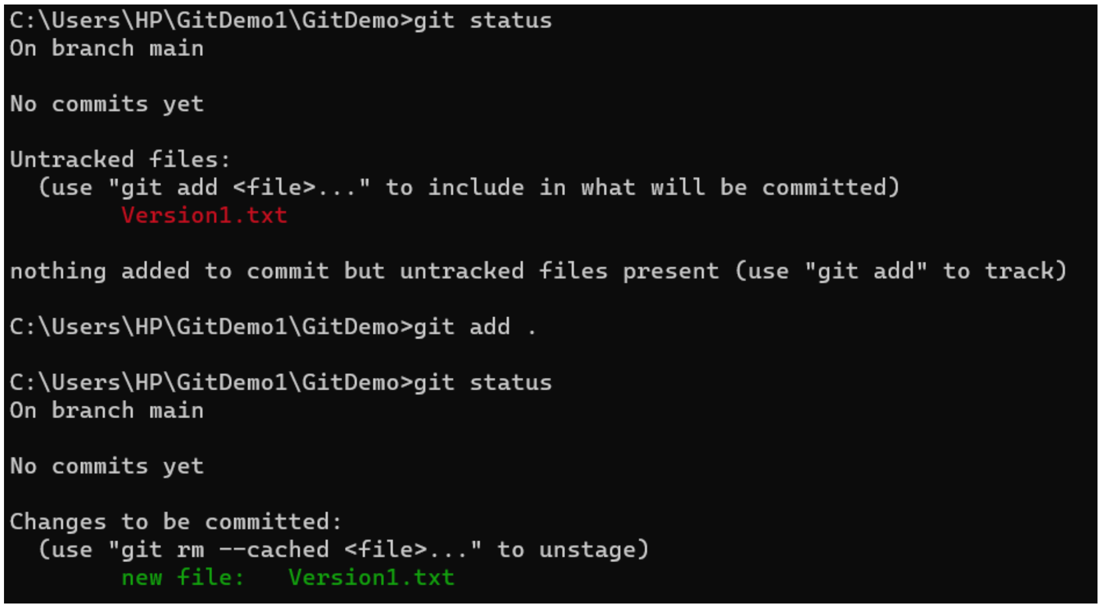


Name	Date modified	Type	Size
Version1	3/21/2023 2:33 PM	Text Document	1 KB

5. Add project from working area to staging area.

    $ git add *

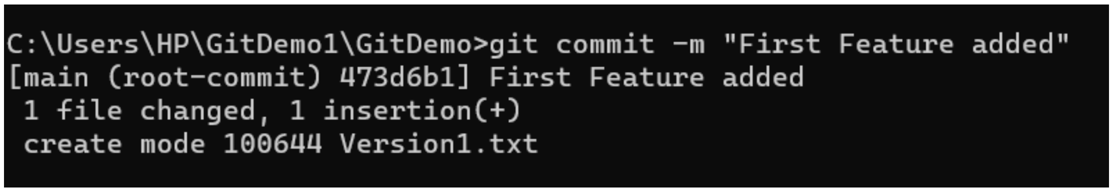
    $ git status ---- To check the status



```
C:\Users\HP\GitDemo1\GitDemo>git status
On branch main

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        Version1.txt

nothing added to commit but untracked files present (use "git add" to track)

C:\Users\HP\GitDemo1\GitDemo>git add .

C:\Users\HP\GitDemo1\GitDemo>git status
On branch main

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   Version1.txt
```

6. Commit the changes done in local repo.

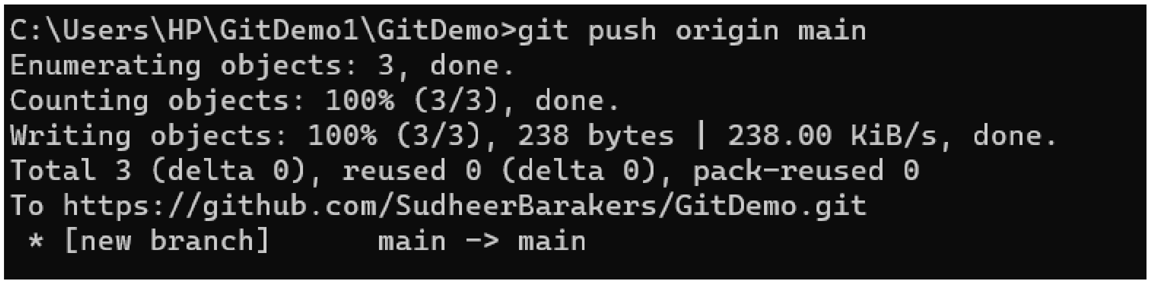
$ git commit -m "Added lines to project"



```
C:\Users\HP\GitDemo1\GitDemo>git commit -m "First Feature added"
[main (root-commit) 473d6b1] First Feature added
 1 file changed, 1 insertion(+)
 create mode 100644 Version1.txt
```

7. Push the project to Central repo (GitHub).

$ git push origin main



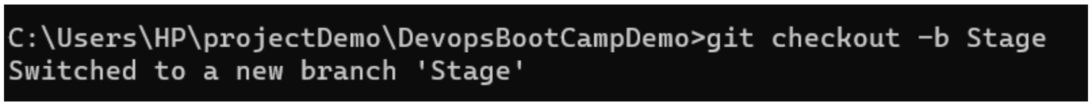
```
C:\Users\HP\GitDemo1\GitDemo>git push origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 238 bytes | 238.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/SudheerBarakers/GitDemo.git
 * [new branch]      main -> main
```

Task 2: Add a text file called Version1.txt inside the project in local repo with "This is first feature of our application" written inside. This should be in a branch coming from master/main, switch to dev branch (Make sure your commit message will reflect as "Added new feature").

Steps to be follow

1. Change to dev branch

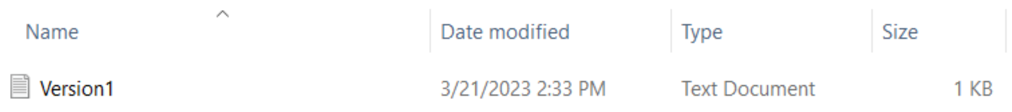
   $ git checkout –b Stage (Create dev branch with –b and checkout to dev branch)



```
C:\Users\HP\projectDemo\DevopsBootCampDemo>git checkout -b Stage
Switched to a new branch 'Stage'
```

2. Create a file in to dev branch

   Version1.txt

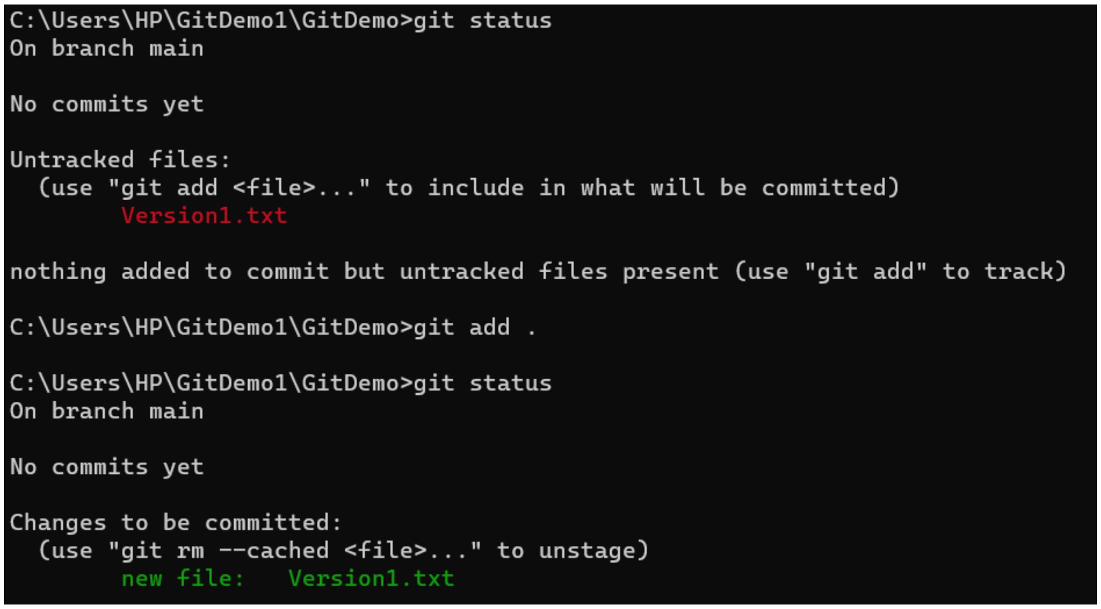


Name	Date modified	Type	Size
Version1	3/21/2023 2:33 PM	Text Document	1 KB

3. Add code from working area to staging area.

   $ git add *

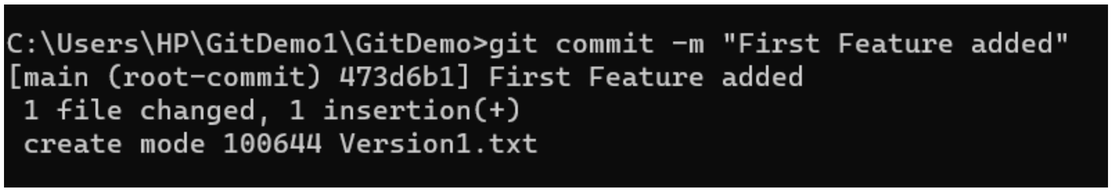
   $ git status ---- To check the status



```
C:\Users\HP\GitDemo1\GitDemo>git status
On branch main

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        Version1.txt

nothing added to commit but untracked files present (use "git add" to track)

C:\Users\HP\GitDemo1\GitDemo>git add .

C:\Users\HP\GitDemo1\GitDemo>git status
On branch main

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   Version1.txt
```

4. Commit the changes done in local repo.

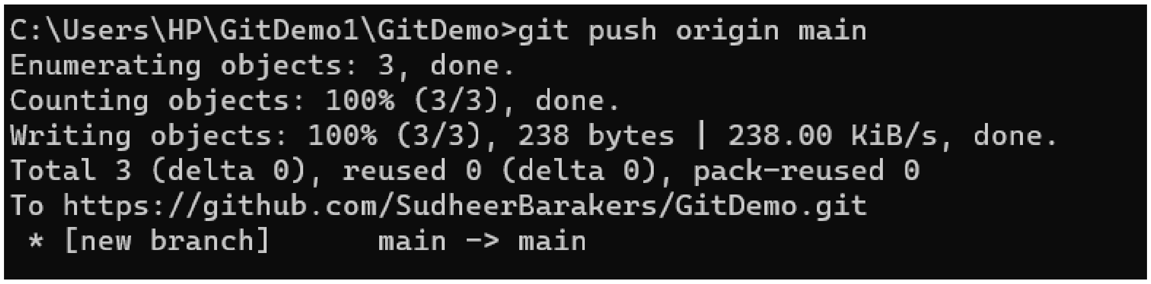
$ git commit -m "First Feature added"



```
C:\Users\HP\GitDemo1\GitDemo>git commit -m "First Feature added"
[main (root-commit) 473d6b1] First Feature added
 1 file changed, 1 insertion(+)
 create mode 100644 Version1.txt
```

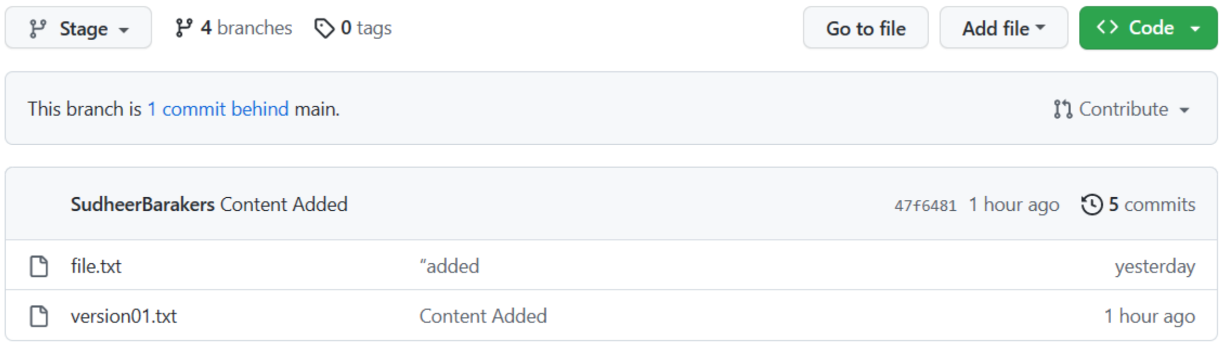
5. Push the code to Central repo (GitHub).

$ git push origin master



```
C:\Users\HP\GitDemo1\GitDemo>git push origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 238 bytes | 238.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/SudheerBarakers/GitDemo.git
 * [new branch]      main -> main
```

6. Verify the changes in central repo (GitHub)



This branch is 1 commit behind main.

SudheerBarakers Content Added 47f6481 1 hour ago 5 commits

file.txt	"added	yesterday
version01.txt	Content Added	1 hour ago