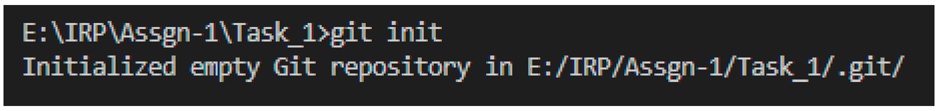
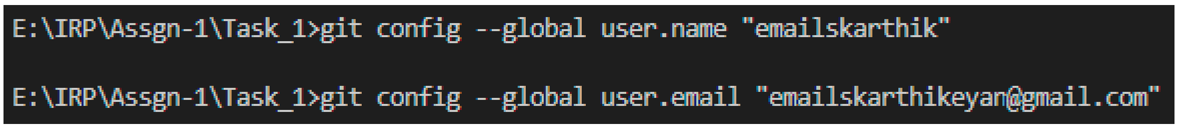
1. git init



```
E:\IRP\Assgn-1\Task_1>git init
Initialized empty Git repository in E:/IRP/Assgn-1/Task_1/.git/
```

The git init command creates a new Git repository. It can be used to convert an existing, unversioned project to a Git repository or initialize a new, empty repository. Most other Git commands are not available outside of an initialized repository, so this is usually the first command you'll run in a new project.

2. git config

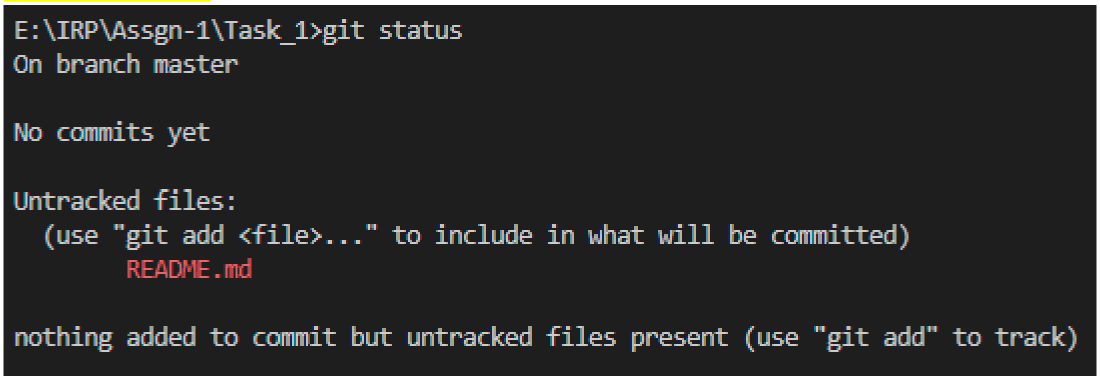


```
E:\IRP\Assgn-1\Task_1>git config --global user.name "emailskarthik"

E:\IRP\Assgn-1\Task_1>git config --global user.email "emailskarthikeyan@gmail.com"
```

The above example writes the value your_email@example.com to the configuration name user.email. It uses the --global flag so this value is set for the current operating system user.

The git config command is a convenience function that is used to set Git configuration values on a global or local project level. These configuration levels correspond to .gitconfig text files. Executing git config will modify a configuration text file. We'll be covering common configuration settings like email, username, and editor.

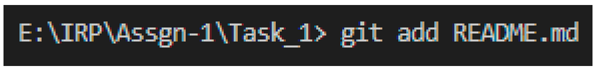
3. git status



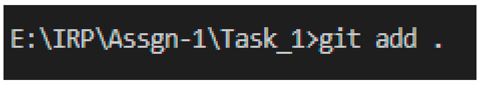
```
E:\IRP\Assgn-1\Task_1>git status
On branch master

No commits yet

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

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

The git status command is a relatively straightforward command. It simply shows you what's been going on with git add and git commit. Status messages also include relevant instructions for staging/unstaging files.

4. git add & git add .



```
E:\IRP\Assgn-1\Task_1> git add README.md
```

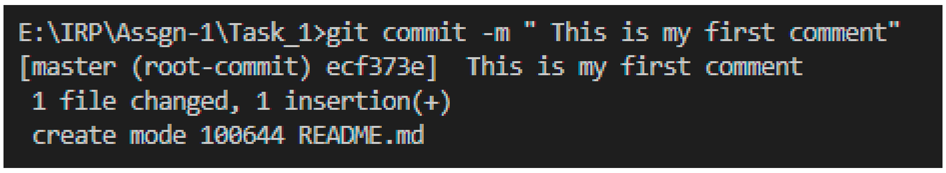


```
E:\IRP\Assgn-1\Task_1>git add .
```

The git add command adds a change in the working directory to the staging area. It tells Git that you want to include updates to a particular file in the next commit. However, git add doesn't really affect the repository in any significant way—changes are not actually recorded until you run git commit.

The git add . Command applies to all the files in the working directory.

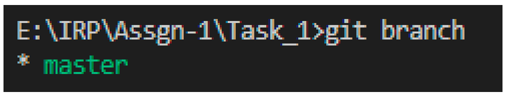
5. git commit –m “This is my first comment”



```
E:\IRP\Assgn-1\Task_1>git commit -m " This is my first comment"
[master (root-commit) ecf373e]  This is my first comment
 1 file changed, 1 insertion(+)
 create mode 100644 README.md
```

The git commit command captures a snapshot of the project's currently staged changes. Committed snapshots can be thought of as “safe” versions of a project—Git will never change them unless you explicitly ask it to. Prior to the execution of git commit, The git add command is used to promote or 'stage' changes to the project that will be stored in a commit. These two commands git commit and git add are two of the most frequently used.

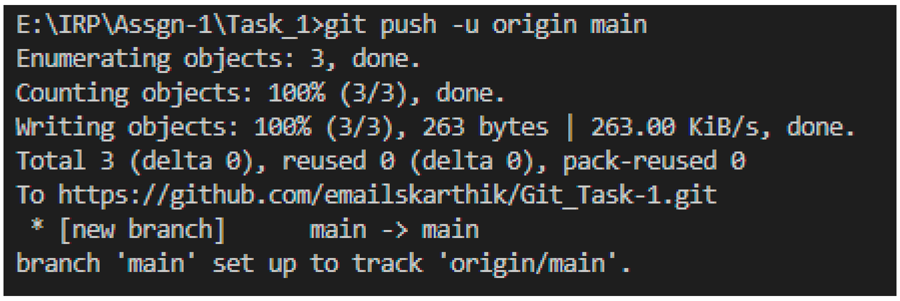
6. git branch



```
E:\IRP\Assgn-1\Task_1>git branch
* master
```

List all of the branches in your repository. This is synonymous with git branch --list.

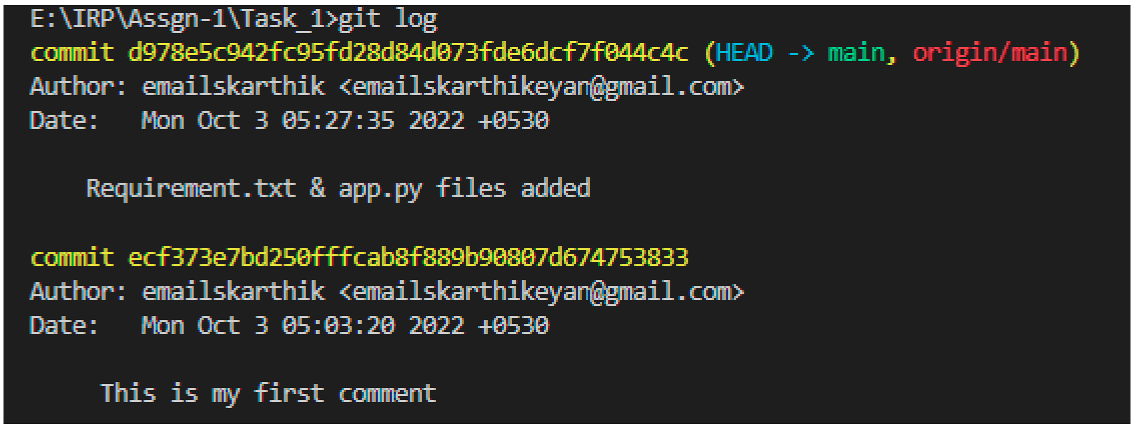
7. git push –u origin main



```
E:\IRP\Assgn-1\Task_1>git push -u origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 263 bytes | 263.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/emailskarthik/Git_Task-1.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
```

The git push –u origin main command says "push the commits in the local branch named main to the remote named origin". Once this is executed, all the stuff that we last synchronised with origin will be sent to the remote repository and other people will be able to see them there.

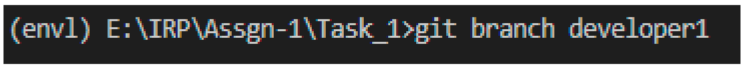
8. git log



```
E:\IRP\Assgn-1\Task_1>git log
commit d978e5c942fc95fd28d84d073fde6dcf7f044c4c (HEAD -> main, origin/main)
Author: emailskarthik <emailskarthikeyan@gmail.com>
Date:   Mon Oct 3 05:27:35 2022 +0530

    Requirement.txt & app.py files added

commit ecf373e7bd250fffcab8f889b90807d674753833
Author: emailskarthik <emailskarthikeyan@gmail.com>
Date:   Mon Oct 3 05:03:20 2022 +0530

     This is my first comment
```

The git log command displays all of the commits in a repository's history.

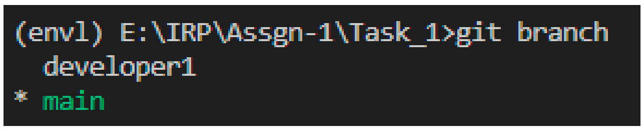
9. git branch developer1



```
(envl) E:\IRP\Assgn-1\Task_1>git branch developer1
```

Create a new branch called < branch > . This does *not* check out the new branch

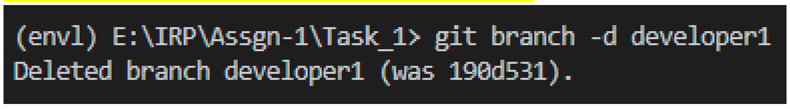
10.git branch



```
(envl) E:\IRP\Assgn-1\Task_1>git branch
  developer1
* main
```

List all of the branches in your repository.

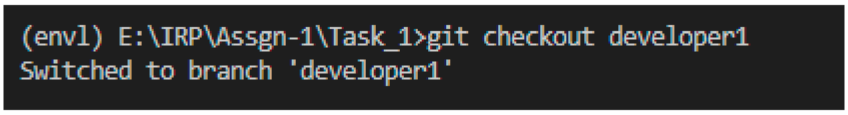
11. git branch –d developer1



```
(envl) E:\IRP\Assgn-1\Task_1> git branch -d developer1
Deleted branch developer1 (was 190d531).
```

Delete the specified branch. This is a “safe” operation in that Git prevents you from deleting the branch if it has unmerged changes.

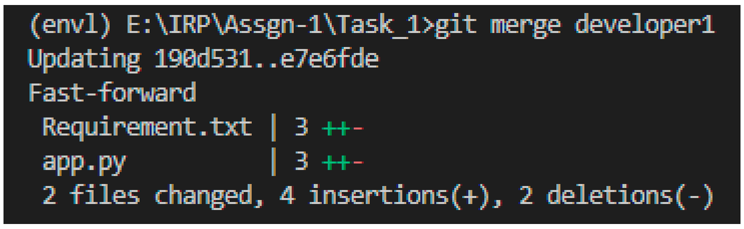
12. git checkout developer1



```
(envl) E:\IRP\Assgn-1\Task_1>git checkout developer1
Switched to branch 'developer1'
```

Enables to shift from main branch to sub branch developer1

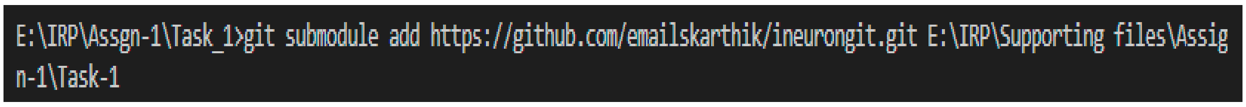
13. git merge developer1



```
(envl) E:\IRP\Assgn-1\Task_1>git merge developer1
Updating 190d531..e7e6fde
Fast-forward
 Requirement.txt | 3 ++-
 app.py          | 3 ++-
 2 files changed, 4 insertions(+), 2 deletions(-)
```

This command merges the specified branch into the current branch, but always generates a merge commit .

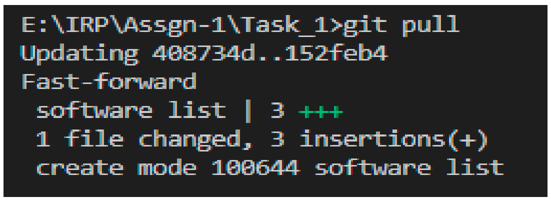
14. git submodule add git@mygithost:billboard lib/billboard



```
E:\IRP\Assgn-1\Task_1>git submodule add https://github.com/emailskarthik/ineurongit.git E:\IRP\Supporting files\Assig
n-1\Task-1
```

- git submodule add – This simply tells Git that we are adding a submodule. This syntax will always remain the same.
- git@mygithost:billboard – This is the external repository that is to be added as a submodule. The exact syntax will vary depending on the setup of the Git repository you are connecting to. You need to ensure that you have the ability to clone the given repository.
- lib/billboard – This is the path where the submodule repository will be added to the main repository.

15. git pull



```
E:\IRP\Assgn-1\Task_1>git pull
Updating 408734d..152feb4
Fast-forward
 software list | 3 +++
 1 file changed, 3 insertions(+)
 create mode 100644 software list
```

The git pull command is used to fetch and download content from a remote repository and immediately update the local repository to match that content. Merging remote upstream changes into your local repository is a common task in Git-based collaboration work flows. The git pull command is actually a combination of two other commands, git fetch followed by git merge. In the first stage of operation git pull will execute a git fetch scoped to the local branch that HEAD is pointed at. Once the content is downloaded, git pull will enter a merge workflow. A new merge commit will be-created and HEAD updated to point at the new commit.