ASSIGNMENT 1

Name : Gunjan Kadu
Matriculation Number : 11012377

TASK 1 - GIT CLI ASSIGNMENT

1. An SSH key is a way to identify yourself rather than typing your username and password for each time you commit to a repository. It works on Secure Shell network protocol which is an authenticated and encrypted secure network protocol used for remote communications between machined on an unsecured open network.

SSH keys generally com in pairs of public key that get shared with services like version controlling system like Git Hub and a private key that is only stored on your computer.

To Generate a SSH Key the following steps needs to be followed.

Generate Your SSH Key (1)-

To generate an SSH key use the following command.

ssh-kedge -t rsa - b 4069 -C "your mail address"

After this we get the address of the key stored on the local machine.

Use the Key (2)-

After the key is generated we need to use it by starting the ssh agent by the command

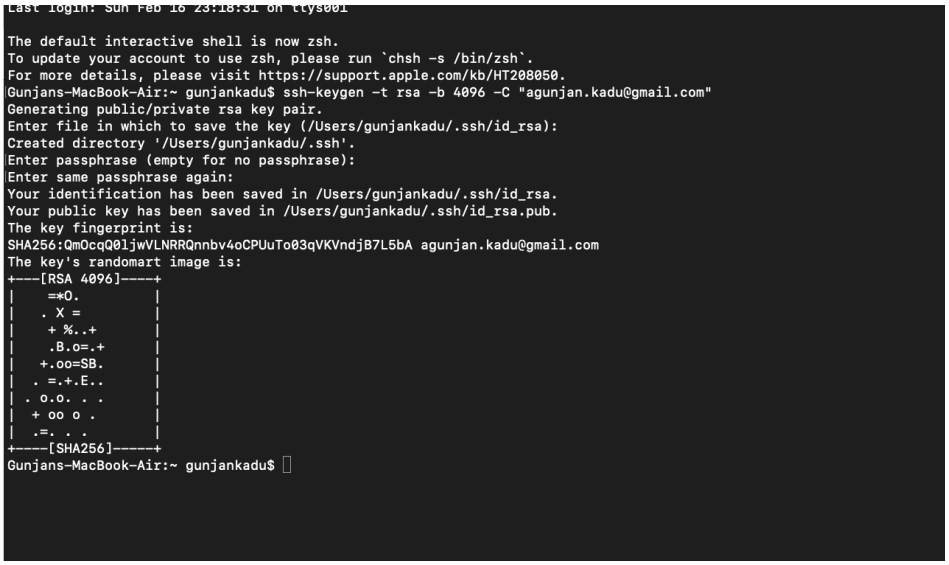
eval $(ssh-agent -s)

Now we need to add the key generated to by using the command.

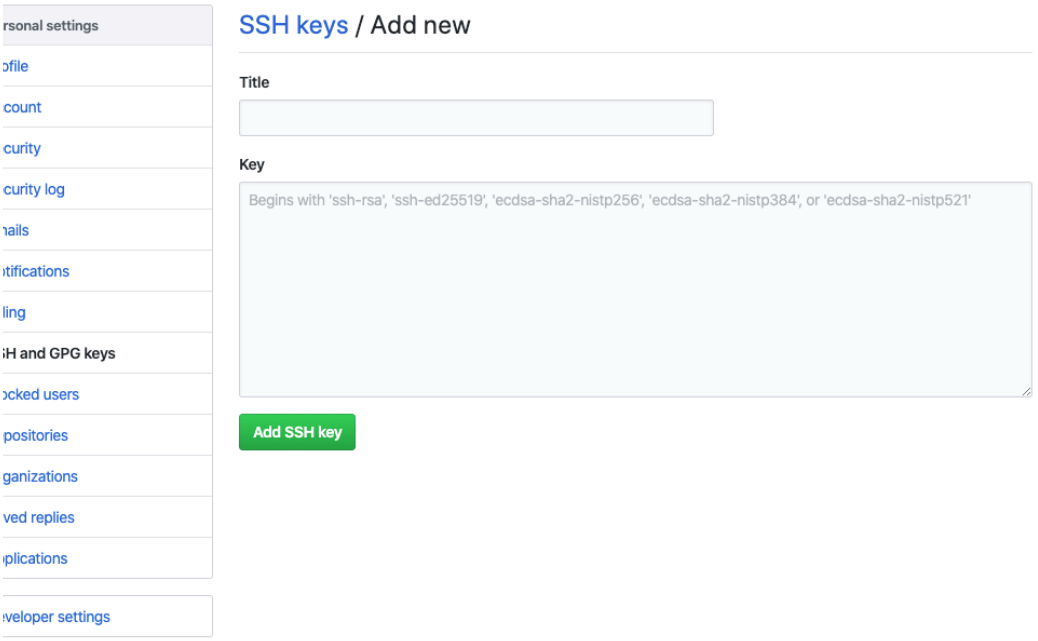
ssh-add ~/.ssh/id_rsa

Add the Ssh Key on Github (3)-

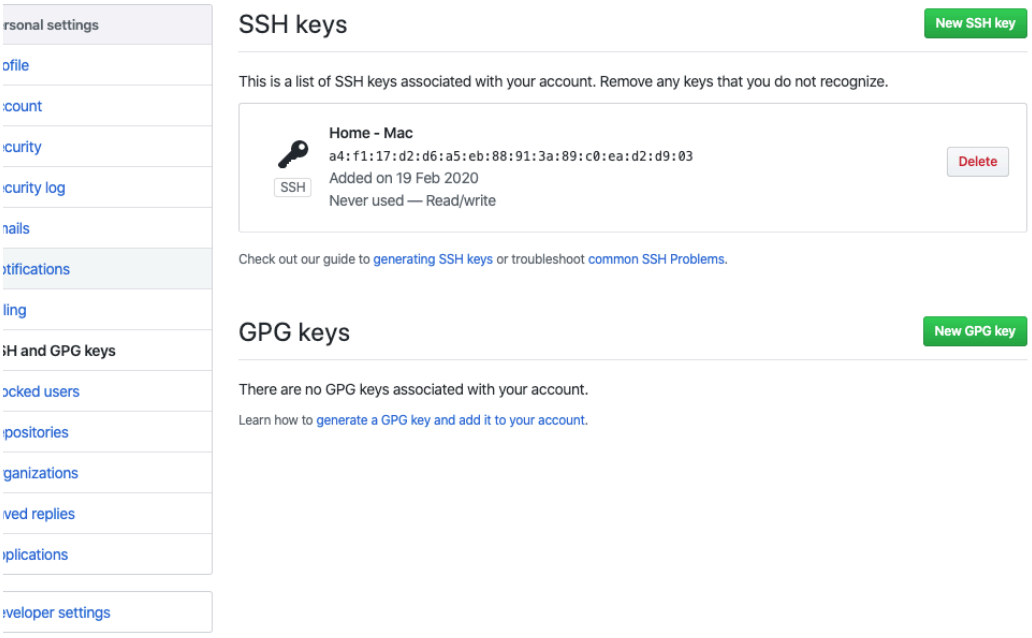
Go to your GitHub account and in the meeting click SSH and GPG keys. Click New SSH key and give it a title. Then paste the key into the box or drag the file directly from your local machine to the box area.



RESULTS OF STEP 1 & 2



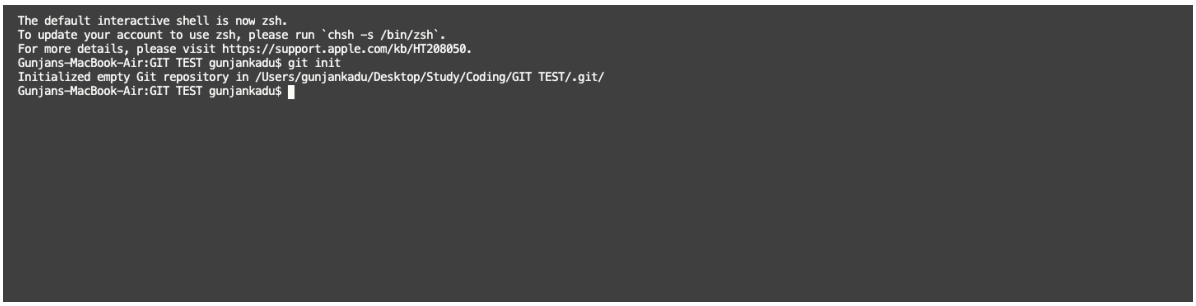
RESULT OF STEP 3 -1



RESULT OF STEP 3 -2

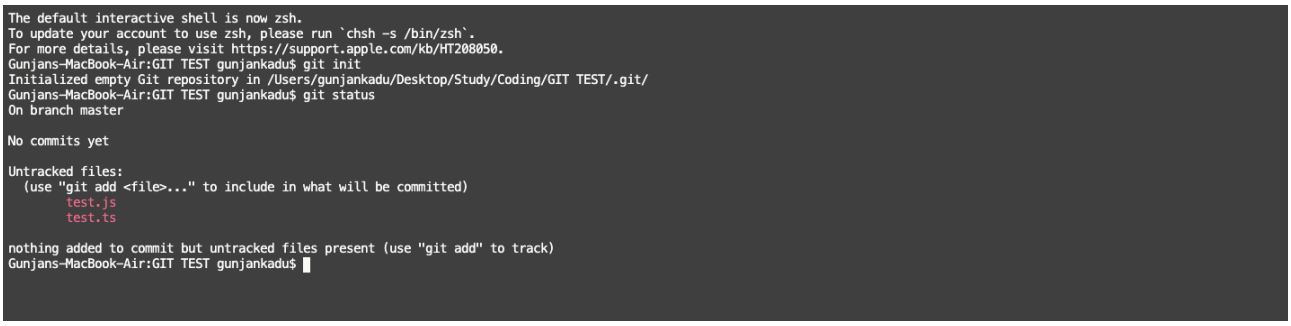
2. To create a local repository we need to install git in our system.

- To create a local repository navigate to folder using CLI and then type the command
  - ***git init***



STEP 1

- Create some files in the folder and we can see that the git has started tracking the files in the folder.
- To check the status of the folder type the command
  - ***git -status***



STEP - 2

- Now we need to add the files to the staging area which is done by the following command
  - ***git add test.js && git add test.ts***

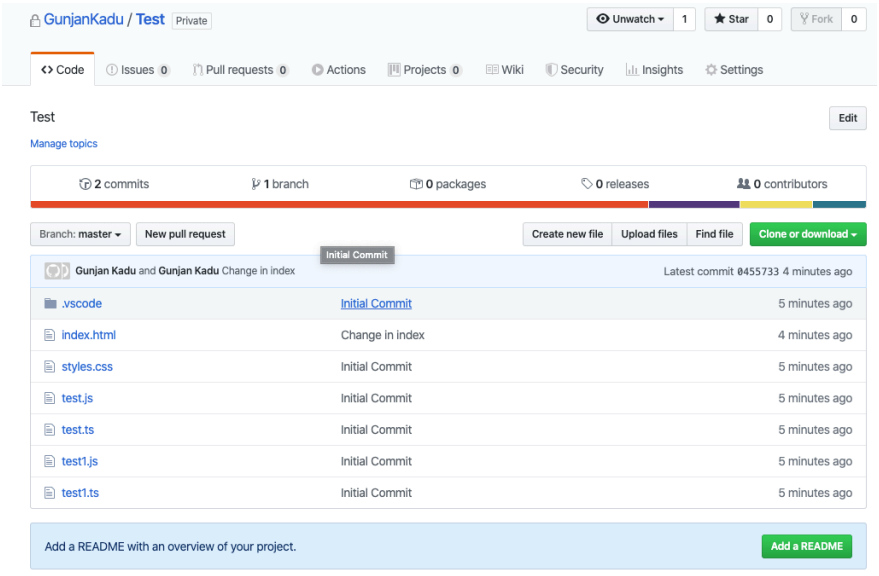
Or ***git add .*** **-** To add all the files in the workspace.

- After adding the files we need to commit the files which is done by using the command.
  - ***git commit -m "Initial Commit"***

- To push the local folder to the remote git server we need to create a repository on the git and follow the following commands.
  - ***git remote add origin git@github.com:GunjanKadu/Test.git***
  - ***git push -u origin master***

```
Gunjans-MacBook-Air:Test gunjankadu$ git add index.html
Gunjans-MacBook-Air:Test gunjankadu$ git commit -m "Change in index"
[master 0455733] Change in index
 Committer: Gunjan Kadu <gunjankadu@Gunjans-MacBook-Air.local>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly. Run the
following command and follow the instructions in your editor to edit
your configuration file:

    git config --global --edit

After doing this, you may fix the identity used for this commit with:

    git commit --amend --reset-author

 1 file changed, 2 insertions(+), 2 deletions(-)
Gunjans-MacBook-Air:Test gunjankadu$ git push
Enter passphrase for key '/Users/gunjankadu/.ssh/id_rsa':
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 317 bytes | 158.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To github.com:GunjanKadu/Test.git
   8bf58e6..0455733  master -> master
```

Then we can see that our local files and the git is now available on the git server.



OUTPUT

- We now will create a new branch in our repository and edit our code there. For that first we used
    - ***git branch test-branch***

- No we will checkout on this branch using
    - ***git checkout test-branch***

```
Gunjans-MacBook-Air:Test gunjankadu$ git branch
* master
  test-branch
Gunjans-MacBook-Air:Test gunjankadu$ git checkout test-branch
Switched to branch 'test-branch'
Gunjans-MacBook-Air:Test gunjankadu$ git branch
  master
* test-branch
Gunjans-MacBook-Air:Test gunjankadu$
```

OUTPUT

```
Gunjans-MacBook-Air:Test gunjankadu$ git branch test-branch1
Gunjans-MacBook-Air:Test gunjankadu$ git branch
  master
* test-branch
  test-branch1
Gunjans-MacBook-Air:Test gunjankadu$
```

CREATE OTHER BRANCH

- On test branch now we edit a file and commit it to git. And then checkout to master and merge the branch using.
    - ***git checkout master***
    - ***git merge test-branch***

- Now we checkout to test-branch1 and make edits in the same file and on the same line and checkout to master and merge the branch

    - ***git checkout master***
    - ***git merge test-branch1***

- Here we get a merge conflicts. Which means that if 2 developers are working on 2 different branch and they edit the same code then a merge conflict could occur.

- Conflicts generally arise when two people have changed the same lines in a file, or if one developer deleted a file while another developer was modifying it. In these cases, Git cannot automatically determine what is correct.

- Conflicts only affect the developer conducting the merge, the rest of the team is unaware of the conflict. Git will mark the file as being conflicted and halt the merging process. It is then the developers' responsibility to resolve the conflict.

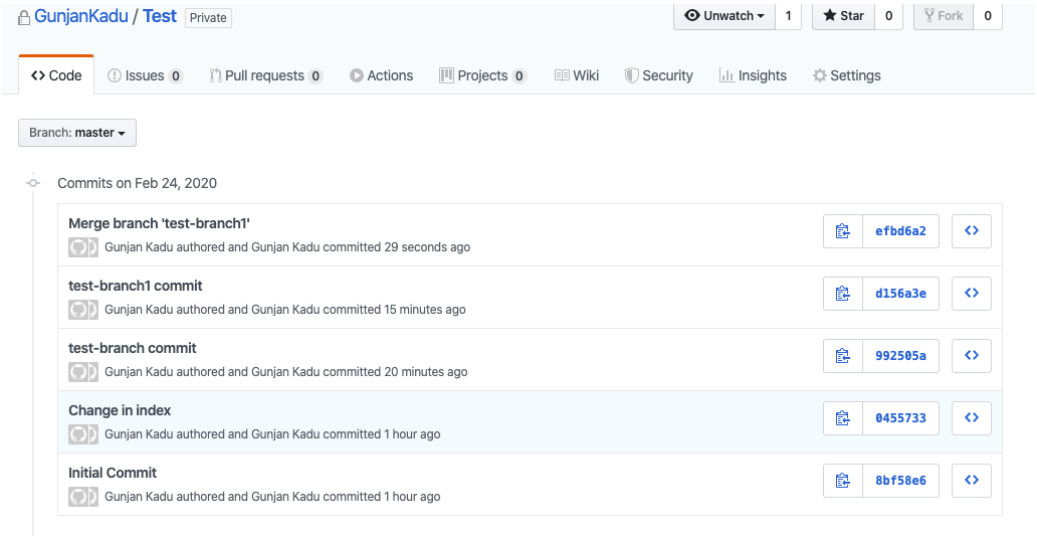
```
                <button
                  className='btn btn-success'
Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
<<<<<<< HEAD (Current Change)
                  onMouseOver={this.handleHoverChange}
=======
                  onMouseOver={this.handleHoverOnChange}
>>>>>>> test-branch1 (Incoming Change)
                >
                  Hover Me
                </button>
              </div>
            );
          }
        }
```

OUTPUT 1

```
Gunjans-MacBook-Air:Test gunjankadu$ git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.
Gunjans-MacBook-Air:Test gunjankadu$ git merge test-branch1
Auto-merging index.html
CONFLICT (content): Merge conflict in index.html
Automatic merge failed; fix conflicts and then commit the result.
```

OUTPUT 2

- To Resolve this merge conflict we will accept the incoming changes as we had deliberately made those changes and resolve that conflict.
- After solving the merge conflicts we can now push the changed file to the master branch.



OUTPUT