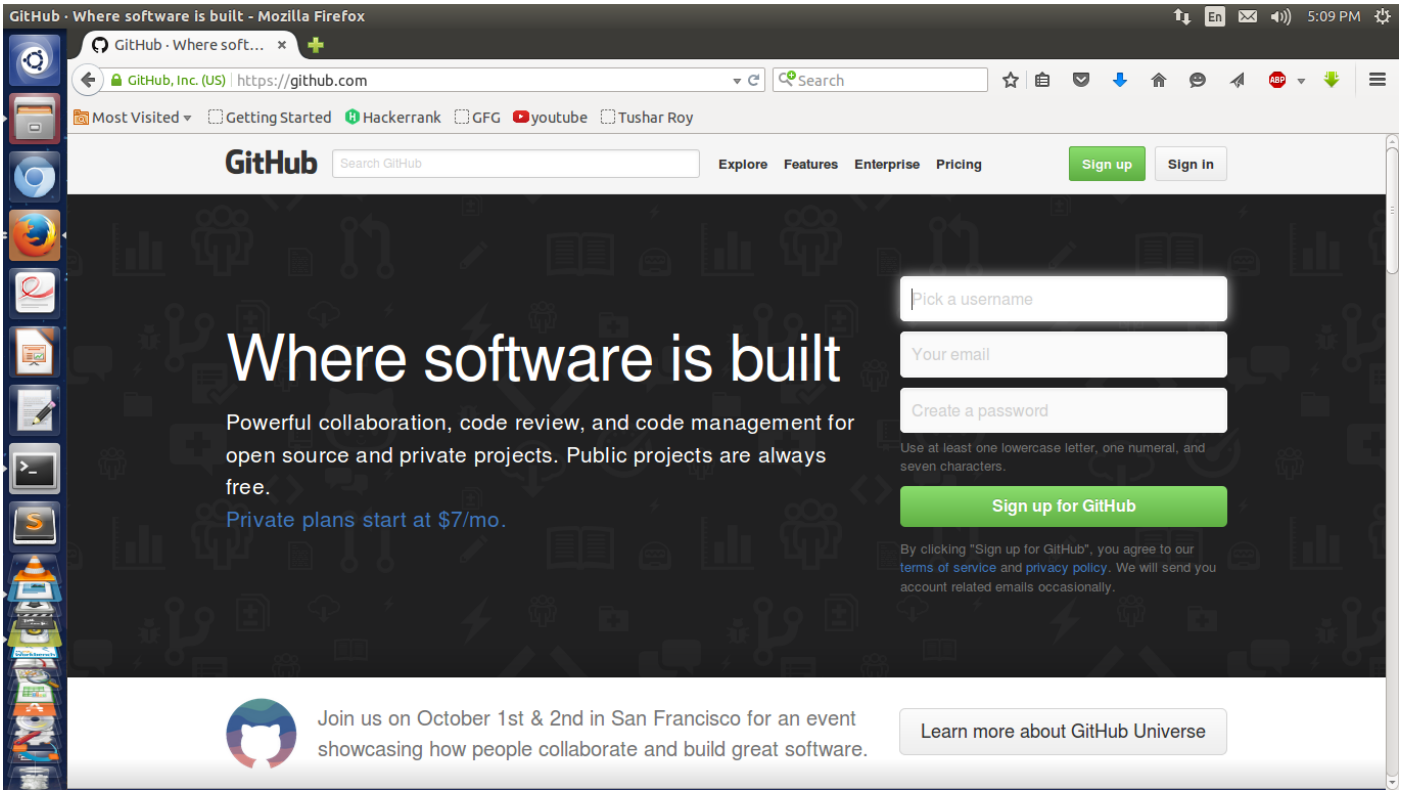
How to install, configure and use GIT on Ubuntu?

INSTALLING GIT:

Step 1: Open the Terminal and type sudo apt-get install git

```
harshit@harshit-System-Product-Name: ~
harshit@harshit-System-Product-Name:~$ sudo apt-get install git
Reading package lists... Done
Building dependency tree
Reading state information... Done
Suggested packages:
  git-daemon-run git-daemon-sysvinit git-doc git-el git-email git-gui gitk
  gitweb git-arch git-bzr git-cvs git-mediawiki git-svn
The following NEW packages will be installed:
  git
0 upgraded, 1 newly installed, 0 to remove and 23 not upgraded.
Need to get 0 B/2,511 kB of archives.
After this operation, 20.3 MB of additional disk space will be used.
Selecting previously unselected package git.
(Reading database ... 274644 files and directories currently installed.)
Preparing to unpack .../git_1%3a1.9.1-1ubuntu0.1_i386.deb ...
Unpacking git (1:1.9.1-1ubuntu0.1) ...
Setting up git (1:1.9.1-1ubuntu0.1) ...
harshit@harshit-System-Product-Name:~$
```

Step 2: Go to **www.github.com** and sign into your account. If you're a new user, you can simply sign-up. (You can also use www.bitbucket.org as an alternative, but we will use github here). You'll have a username from here. Let us say that it's your username



CONFIGURING GIT:

Step 1: Go back to the terminal and type this to configure git

git config –global user.name “your username”

(or)

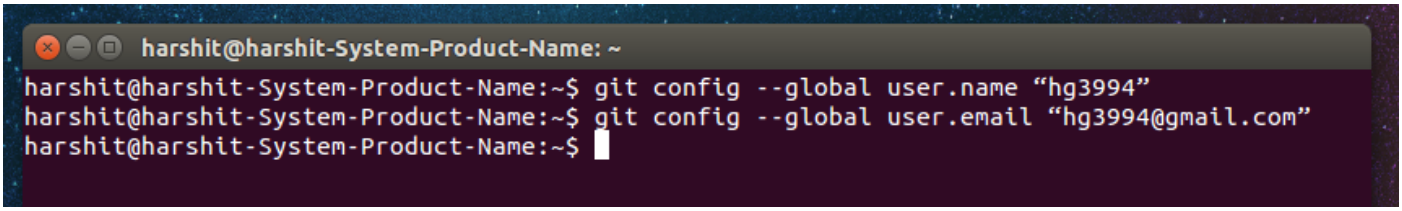
git config user.name “your username”

Step 2: Now type this to link your email too.

git config –global user.email “your email ID”

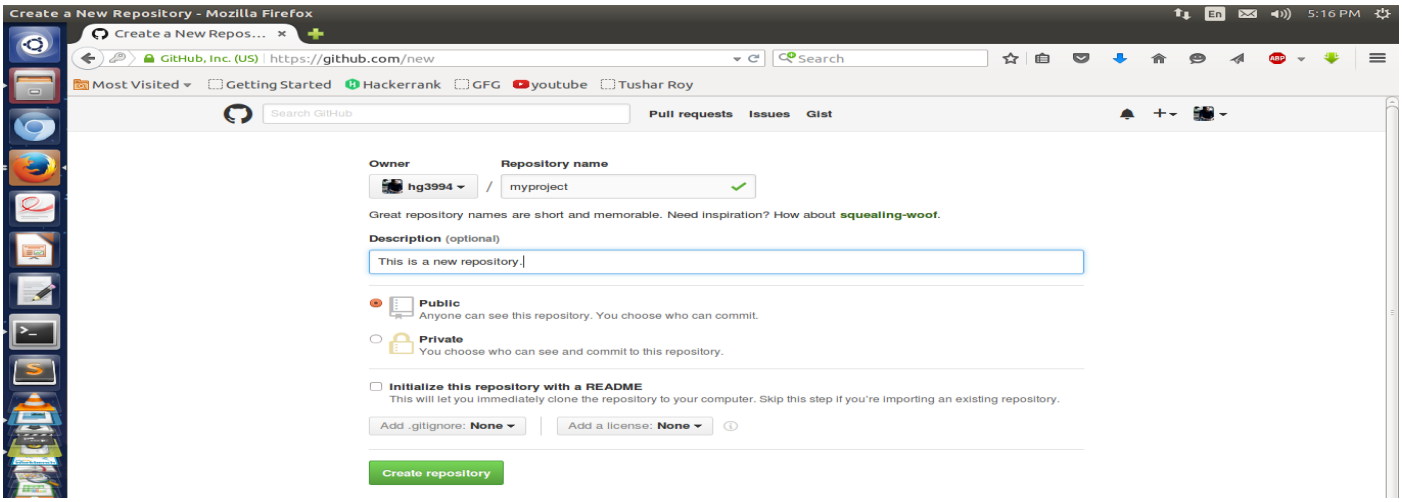
(or)

git config user.email “your email ID”

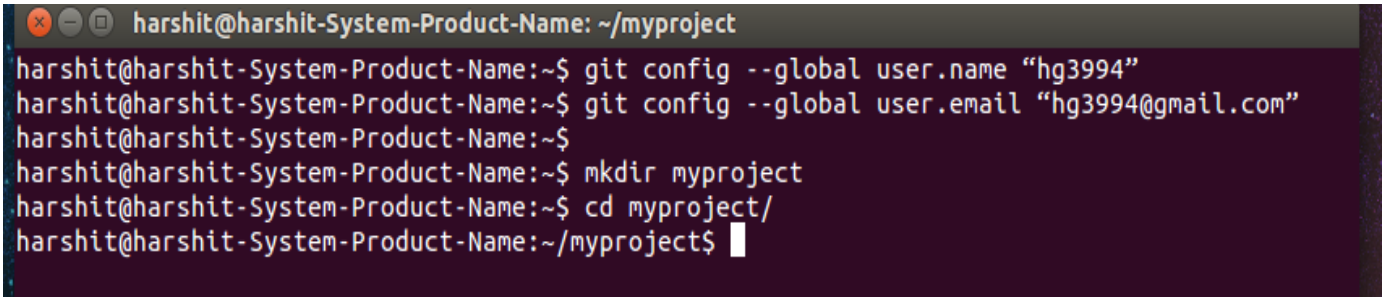


USING GIT:

Step 1: Go to your github account and create a repository with a name (let’s say name of your project). We are creating a repository with the name myproject



Step 2: Make a folder with the name of your project and change your current directory to that Directory. 1) mkdir myproject 2) cd myproject



Step 3: Now we want to initiate Git for this folder

```
git init
```

```
harshit@harshit-System-Product-Name: ~/myproject
harshit@harshit-System-Product-Name:~/myproject$ git init
Initialized empty Git repository in /home/harshit/myproject/.git/
harshit@harshit-System-Product-Name:~/myproject$
```

Step 4: Now we will set up the remote, which tells git where the repository is located.

```
git remote add origin  https://github.com/your_username/myproject.git
```

```
harshit@harshit-System-Product-Name: ~/myproject
harshit@harshit-System-Product-Name:~/myproject$ git remote add origin  https://github.com/hg3994/myproject.git
harshit@harshit-System-Product-Name:~/myproject$
```

We have now configured and installed git and, created and configured a repository. Let's say we have a simple file in the myproject folder helloworld.c and we want it to share it with a friend who is working on the same project.

```
~/myproject/helloworld.c • - Sublime Text 2 (UNREGISTERED)
helloworld.c
1  #include <stdio.h>
2
3  int main()
4  {
5      printf("Hello World! This is Geeksforgeeks.org ")
6      return 0;
7  }
```

Step 5: To add this file we will type

```
git add helloworld.c
```

if we have a lot of files to be transferred from the folder to our git account, then we can use the command.

```
git add .
```

```
harshit@harshit-System-Product-Name: ~/myproject
harshit@harshit-System-Product-Name:~/myproject$ subl helloworld.c
harshit@harshit-System-Product-Name:~/myproject$ git add .
harshit@harshit-System-Product-Name:~/myproject$
```

This would transfer the file(s) in the list which we will later commit.

Step 6: Next, when we are finished adding the files, then we will have to commit adding.

git commit -m 'your message'

```
harshit@harshit-System-Product-Name: ~/myproject
harshit@harshit-System-Product-Name:~/myproject$ git commit -m 'First Commit'
[master (root-commit) 391d57c] First Commit
 2 files changed, 14 insertions(+)
 create mode 100644 helloworld.c
 create mode 100644 helloworld.c~
harshit@harshit-System-Product-Name:~/myproject$
```

Step 7: Next, we need to push the commit that we just made on to the repository at github

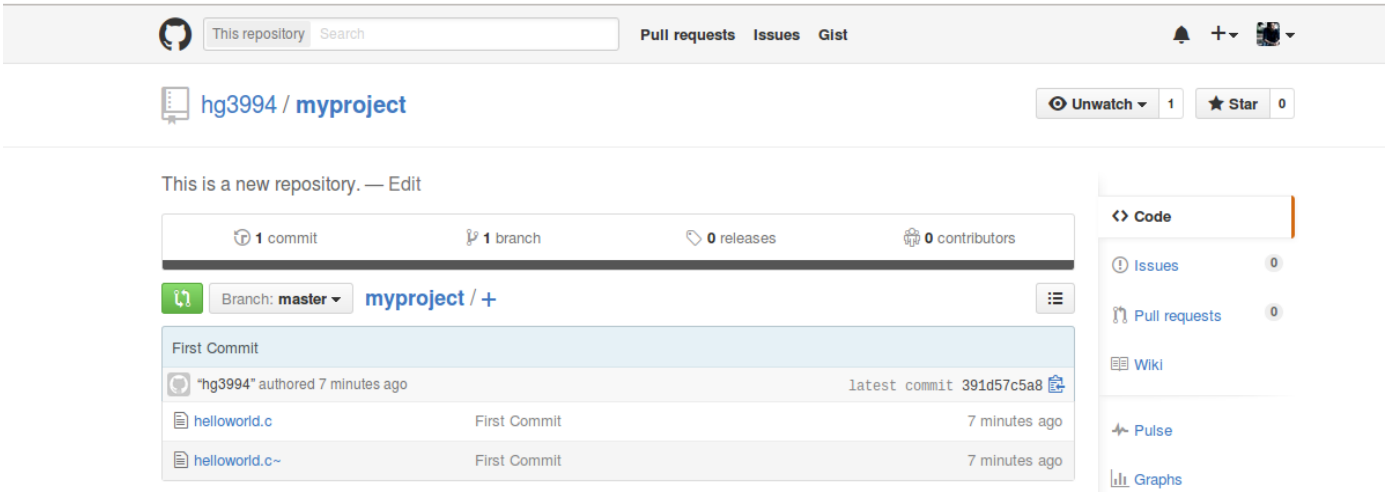
git push origin master

It would automatically ask you for your username and password for github. After entering the details, go to github and refresh. The files would get added there.

Username for ' https://github.com ': your username

Password for ' https://your_username@github.com ' : *******

```
root@Techno-251:~/Desktop/testing# ls
file1  file2  user_files.txt  wel.py
root@Techno-251:~/Desktop/testing# git add .
root@Techno-251:~/Desktop/testing# git commit -m 'additing new files'
[master cefd1d6] additing new files
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 file1
 create mode 100644 file2
root@Techno-251:~/Desktop/testing# git push origin master
Username for 'https://github.com': syed-salman-technoforte
Password for 'https://syed-salman-technoforte@github.com':
To https://github.com/syed-salman-technoforte/testing.git
```

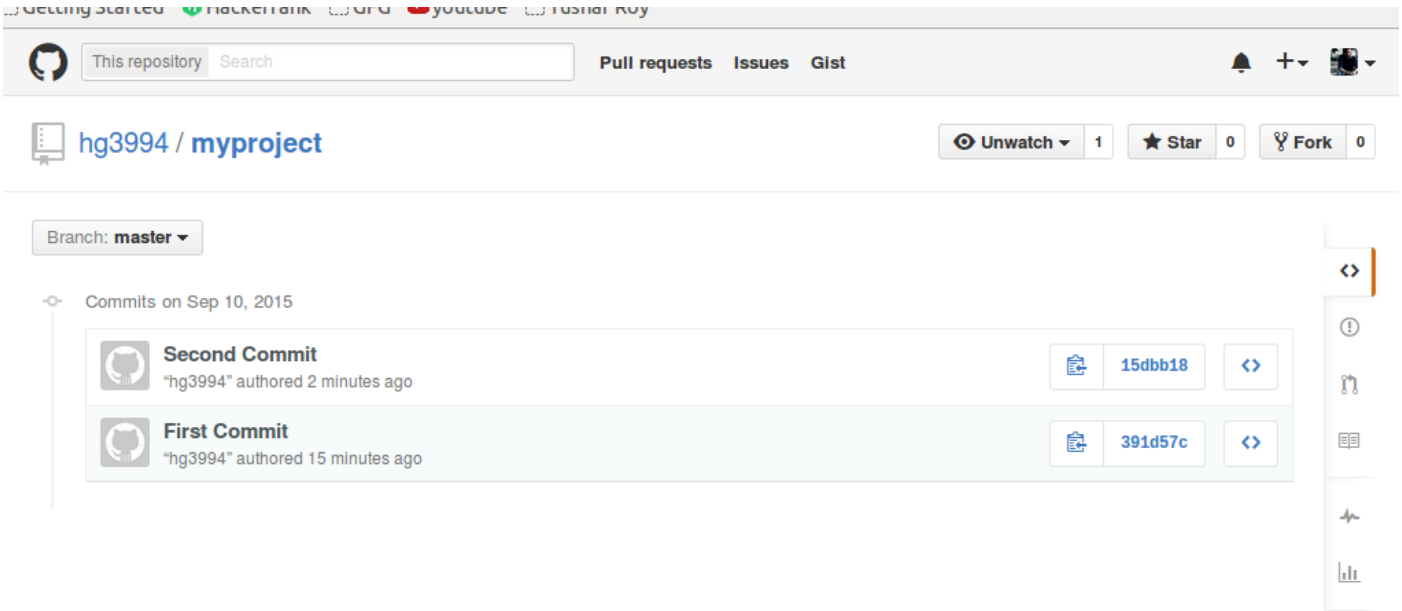


Step 8: We have successfully transferred a file on your github account. Now let's add one more file aboutme.txt and edit our file helloworld.c. Following the same procedure, we will first add the files, commit and then push them to the github account.

- git add . (or) git add -A
- git commit -m 'your message' (or) git commit –a –m "your message"
- git push origin master

```
harshit@harshit-System-Product-Name:~/myproject$ gedit aboutme.txt
harshit@harshit-System-Product-Name:~/myproject$ gedit helloworld.c
harshit@harshit-System-Product-Name:~/myproject$ git add .
harshit@harshit-System-Product-Name:~/myproject$ git commit -m 'Second Commit'
[master 15dbb18] Second Commit
 4 files changed, 4 insertions(+)
 create mode 100644 aboutme.txt
 create mode 100644 aboutme.txt~
harshit@harshit-System-Product-Name:~/myproject$ git push origin master
Username for 'https://github.com': hg3994
Password for 'https://hg3994@github.com':
Counting objects: 7, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (5/5), 550 bytes | 0 bytes/s, done.
Total 5 (delta 0), reused 0 (delta 0)
To https://github.com/hg3994/myproject.git
   391d57c..15dbb18  master -> master
harshit@harshit-System-Product-Name:~/myproject$
```

Step 9: When we would go to our GitHub account, we would see the entire hierarchy of the modification of the file. Here, we would see the changes we made to the helloworld.c file in the respective commits.

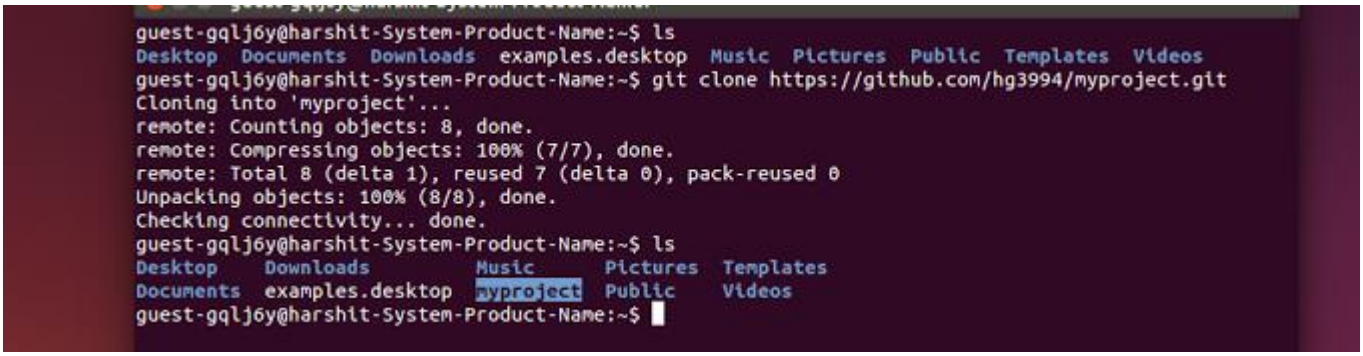


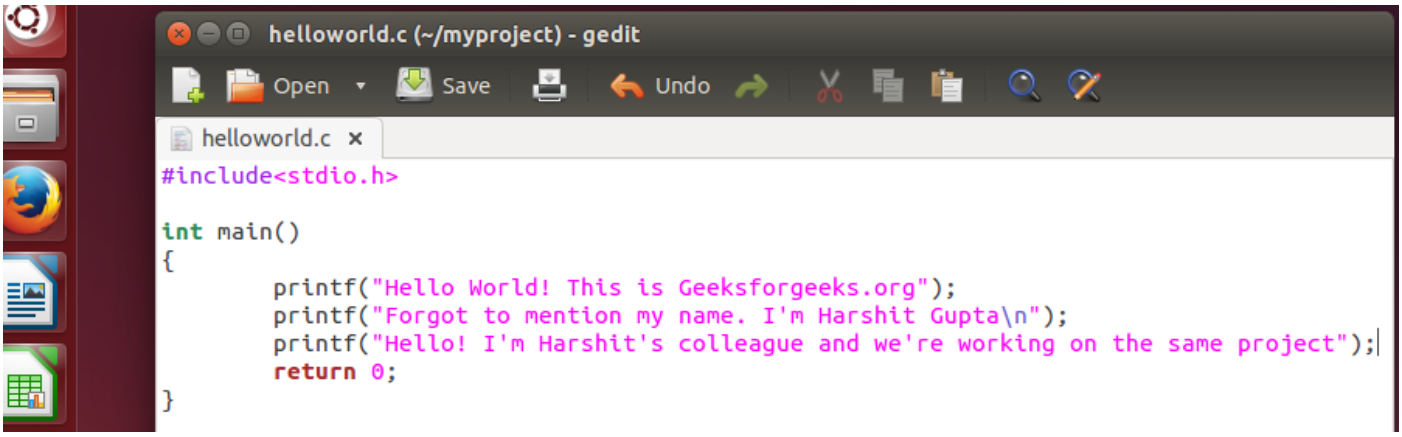
Now, Let's say one of the co-worker of the project needs to work on helloworld.c. After making some changes, he wants to update the file on GitHub.

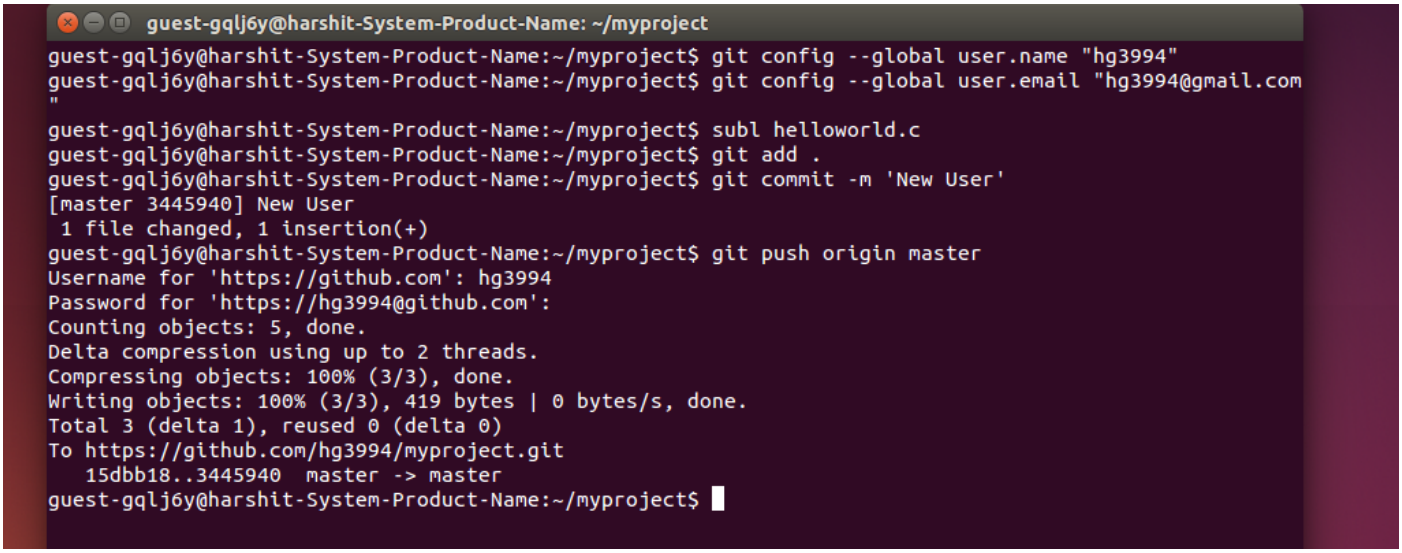
Step 10: First he would have to download the whole repository in which the file helloworld.c is present into his system.

git clone https://github.com/your_username/myproject.git

A folder named myproject gets downloaded with all the files in it. The necessary changes are made and then the file is similarly added, committed and pushed similarly as above.







Step 11: If the any user wishes to see the changes, then he can see it by typing:

git pull origin master // master is the central branch name

Important Git commands:

- **Git user configuration (First Step):**

```
git --version (to check git version)
git config --global user.name "your name here"
git config --global user.email "your email here"
```

These are the information attached to commits.

- **Initialize directory:**

```
git init
```

(Initializes your directory to work with git and makes a local repository. .git folder is made)

(OR)

```
git clone http_url
```

This is done if we have an existing git repository.

- **Connecting to repository:**

```
git remote add origin http_url/ssh_url
```

(connect to central repo to push/pull)

pull means transferring the changes on central repository to your local repository. push is the vice versa of pull.

```
git pull origin master
```

Always first pull contents from central repo before pushing so that you are updated with other team members work. Here, master means the master branch (in Git).

- **Steps to add a file to central Repository:**

  First your file is in your working directory, move it to the staging area by typing:

```
git add -A (for all files and folders)
```

  git status: here, untracked files mean files which you haven't added to the staging area. Changes not staged for commit means you have staged the file earlier then you have made changes in that files in your working directory and the changes need to be staged once more. Changes ready to be committed : these are files which have been committed and ready to be pushed to central repository.

```
git commit -a -m "message for commit"
```

-a: commit all files and for files which have been staged earlier need not to be git add once more -a options does that automatically.

```
git push origin master;         pushes your files to gitHub master branch
git push origin anyOtherBranch; pushes any other branch to gitHub.
git log;                        to see all your commits
```

```
git checkout hashcode_of_versions (first 8 bits) file.txt
(revert back to this previous commit for file file.txt)
```

Branching in Git

Create Branch

```
git branch myBranch
or
git checkout -b myBranch
(make and switch to the branch myBranch)
```

Then,

```
git checkout master; to switch back to master branch
```

Merge:

merge contents with your myBranch By:

```
git merge myBranch (writing in master branch)
```

This merger makes a new commit.

Rebase:

Another way to merge branches are

```
git rebase myBranch
```

This merges the branch with master in a serial fashion.

Now,

```
git push origin master
```

Contributing to open source by: fork a project and do some work (add new features) in your branch and then do a pull request on github.