Name: Flores, Marc Oliver U.	Date Performed: 08/22/23
Course/Section: CPE 232 - CPE31S4	Date Submitted: 08/22/23
Instructor: Dr. Jonathan V. Taylar	Semester and SY: 1st Semester | SY 2023-2024

Activity 2: SSH Key-Based Authentication and Setting up Git

1. **Objectives:**

1.1 Configure remote and local machine to connect via SSH using a KEY instead of using a password
1.2 Create a public key and private key
1.3 Verify connectivity
1.4 Setup Git Repository using local and remote repositories
1.5 Configure and Run ad hoc commands from local machine to remote servers

Part 1: Discussion

It is assumed that you are already done with the last Activity (**Activity 1: Configure Network using Virtual Machines).** *Provide screenshots for each task*.

It is also assumed that you have VMs running that you can SSH but requires a password. Our goal is to remotely login through SSH using a key without using a password. In this activity, we create a public and a private key. The private key resides in the local machine while the public key will be pushed to remote machines. Thus, instead of using a password, the local machine can connect automatically using SSH through an authorized key.

What Is ssh-keygen?

Ssh-keygen is a tool for creating new authentication key pairs for SSH. Such key pairs are used for automating logins, single sign-on, and for authenticating hosts.

SSH Keys and Public Key Authentication

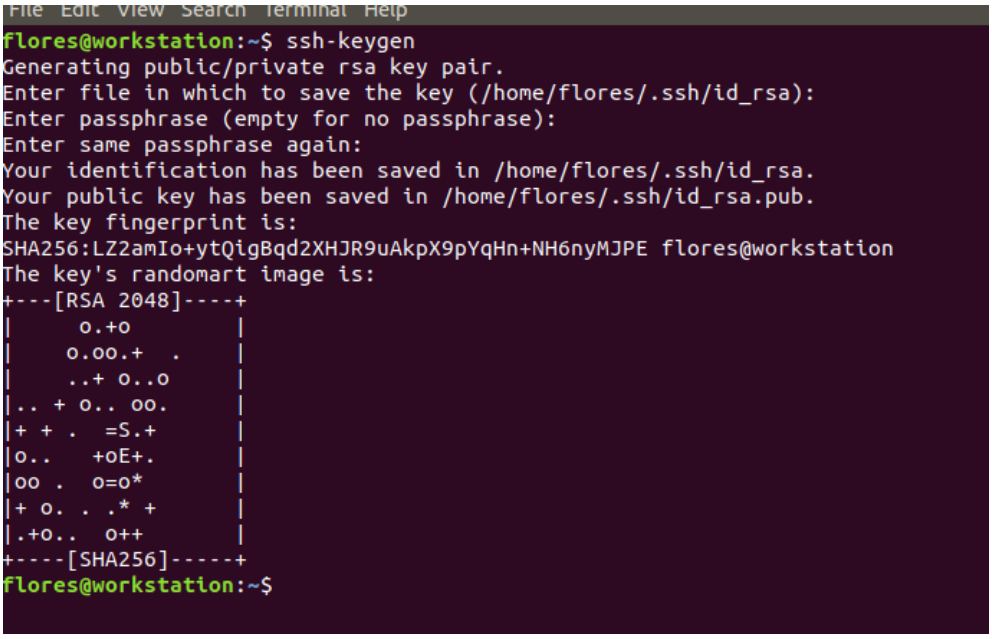
The SSH protocol uses public key cryptography for authenticating hosts and users. The authentication keys, called SSH keys, are created using the keygen program.

SSH introduced public key authentication as a more secure alternative to the older .rhosts authentication. It improved security by avoiding the need to have password stored in files and eliminated the possibility of a compromised server stealing the user's password.

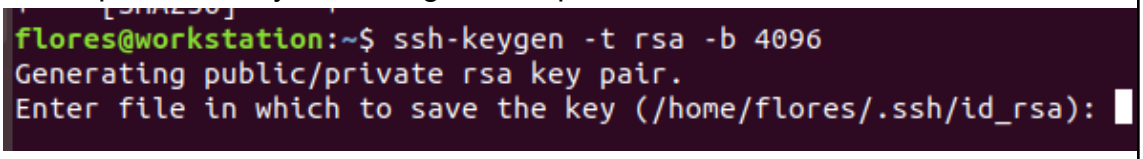
However, SSH keys are authentication credentials just like passwords. Thus, they must be managed somewhat analogously to usernames and passwords. They should have a proper termination process so that keys are removed when no longer needed.

Task 1: Create an SSH Key Pair for User Authentication

1. The simplest way to generate a key pair is to run *ssh-keygen* without arguments. In this case, it will prompt for the file in which to store keys. First, the tool asked where to save the file. SSH keys for user authentication are usually stored in the users .ssh directory under the home directory. However, in enterprise environments, the location is often different. The default key file name depends on the algorithm, in this case *id_rsa* when using the default RSA algorithm. It could also be, for example, *id_dsa* or *id_ecdsa*.



2. Issue the command *ssh-keygen -t rsa -b 4096.* The algorithm is selected using the -t option and key size using the -b option.



3. When asked for a passphrase, just press enter. The passphrase is used for encrypting the key, so that it cannot be used even if someone obtains the private key file. The passphrase should be cryptographically strong.

```
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/flores/.ssh/id_rsa.
Your public key has been saved in /home/flores/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:oz+lZehcI8t1zIeeSdWy5L0G1BvXRZe77qSlqb50tmE flores@workstation
The key's randomart image is:
+---[RSA 4096]----+
|               .+|
|               .o|
|              ..+|
|             .+++|
|        S. o.= +=|
|       .o.B *.+o.|
|      .+ X * Eoo.|
|       .B . B O+ |
|        ...+o=o. |
+----[SHA256]-----+
```

4. Verify that you have created the key by issuing the command *ls -la .ssh.* The command should show the .ssh directory containing a pair of keys. For example, id_rsa.pub and id_rsa.

```
flores@workstation:~$ ls -la .ssh
total 20
drwx------  2 flores flores 4096 Aug 22 17:01 .
drwxr-xr-x 17 flores flores 4096 Aug 22 16:43 ..
-rw-------  1 flores flores 3243 Aug 22 17:03 id_rsa
-rw-r--r--  1 flores flores  744 Aug 22 17:03 id_rsa.pub
-rw-r--r--  1 flores flores 1110 Aug 15 17:50 known_hosts
```

Task 2: Copying the Public Key to the remote servers

1. To use public key authentication, the public key must be copied to a server and installed in an *authorized_keys* file. This can be conveniently done using the *ssh-copy-id* tool.

```
flores@workstation:~$ ssh-copy-id
Usage: /usr/bin/ssh-copy-id [-h|-?|-f|-n] [-i [identity_file]] [-p port] [[
ssh -o options>] ...] [user@]hostname
        -f: force mode -- copy keys without trying to check if they are alr
 installed
        -n: dry run    -- no keys are actually copied
        -h|-?: print this help
```

2. Issue the command similar to this: *ssh-copy-id -i ~/.ssh/id_rsa user@host*

```
flores@workstation:~$ ssh-copy-id -i ~/.ssh/id_rsa flores@workstation
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/flores/.ss
h/id_rsa.pub"
The authenticity of host 'workstation (10.0.2.15)' can't be established.
ECDSA key fingerprint is SHA256:Qv2fASqW0DdrA8UmlkJhW3+RJbSIp8TQaCJKkN90eSI.
Are you sure you want to continue connecting (yes/no)? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter
 out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are promp
ted now it is to install the new keys
flores@workstation's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'flores@workstation'"
and check to make sure that only the key(s) you wanted were added.
```

3. Once the public key has been configured on the server, the server will allow any connecting user that has the private key to log in. During the login process, the client proves possession of the private key by digitally signing the key exchange.

```
flores@workstation:~$ ssh-copy-id -i ~/.ssh/id_rsa flores@Server1
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/flores/.ss
h/id_rsa.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter
 out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are promp
ted now it is to install the new keys
flores@server1's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'flores@Server1'"
and check to make sure that only the key(s) you wanted were added.
```

```
flores@workstation:~$ ssh-copy-id -i ~/.ssh/id_rsa flores@Server2
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/flores/.ss
h/id_rsa.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter
 out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are promp
ted now it is to install the new keys
flores@server2's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'flores@Server2'"
and check to make sure that only the key(s) you wanted were added.
```

4. On the local machine, verify that you can SSH with Server 1 and Server 2. What did you notice? Did the connection ask for a password? If not, why?

   I notice that it doesn't need any password and just straight connecting to the other server I think this is because they have the same key which applies as an authentication that this user has the same key and allows the user to enter different server.

```
flores@workstation:~$ ssh flores@Server1
Welcome to Ubuntu 18.04.6 LTS (GNU/Linux 5.4.0-150-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

Expanded Security Maintenance for Infrastructure is not enabled.

0 updates can be applied immediately.

85 additional security updates can be applied with ESM Infra.
Learn more about enabling ESM Infra service for Ubuntu 18.04 at
https://ubuntu.com/18-04

New release '20.04.6 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Your Hardware Enablement Stack (HWE) is supported until April 2023.
Last login: Tue Aug 22 17:25:58 2023 from 192.168.164.7
```

```
flores@workstation:~$ ssh flores@Server2
Welcome to Ubuntu 18.04.6 LTS (GNU/Linux 5.4.0-150-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

Expanded Security Maintenance for Infrastructure is not enabled.

0 updates can be applied immediately.

85 additional security updates can be applied with ESM Infra.
Learn more about enabling ESM Infra service for Ubuntu 18.04 at
https://ubuntu.com/18-04

New release '20.04.6 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Your Hardware Enablement Stack (HWE) is supported until April 2023.
Last login: Tue Aug 15 17:50:43 2023 from 192.168.164.7
flores@Server2:~$
```

Reflections:
Answer the following:

1. How will you describe the ssh-program? What does it do?
   SSH is a software package that provides secure system administration and file transfers over insecure networks. It is widely used by network administrators and developers to manage remote systems & applications, execute commands, share files, etc. SSH is a cryptographic network protocol for operating network services securely over an unsecured network. Its most notable applications are remote login and command-line execution.
2. How do you know that you already installed the public key to the remote servers?
   Based on what I checked on the internet when troubleshooting during the activity you can use the cat command to view the contents of the authorized_keys file if the public key is added correctly or not.

Part 2: Discussion

Provide screenshots for each task.

It is assumed that you are done with the last activity (**Activity 2: SSH Key-Based Authentication**).

Set up Git
At the heart of GitHub is an open-source version control system (VCS) called Git. Git is responsible for everything GitHub-related that happens locally on your computer. To use Git on the command line, you'll need to download, install, and configure Git on your computer. You can also install GitHub CLI to use GitHub from the command line. If you don't need to work with files locally, GitHub lets you complete many Git-related actions directly in the browser, including:

- Creating a repository
- Forking a repository
- Managing files
- Being social

Task 3: Set up the Git Repository

1. On the local machine, verify the version of your git using the command *which git*. If a directory of git is displayed, then you don't need to install git. Otherwise, to install git, use the following command: *sudo apt install git*

```
flores@workstation:~$ sudo apt install git -y
[sudo] password for flores:
Reading package lists... Done
Building dependency tree
Reading state information... Done
git is already the newest version (1:2.17.1-1ubuntu0.18).
The following package was automatically installed and is no longer required:
  libllvm7
Use 'sudo apt autoremove' to remove it.
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
```

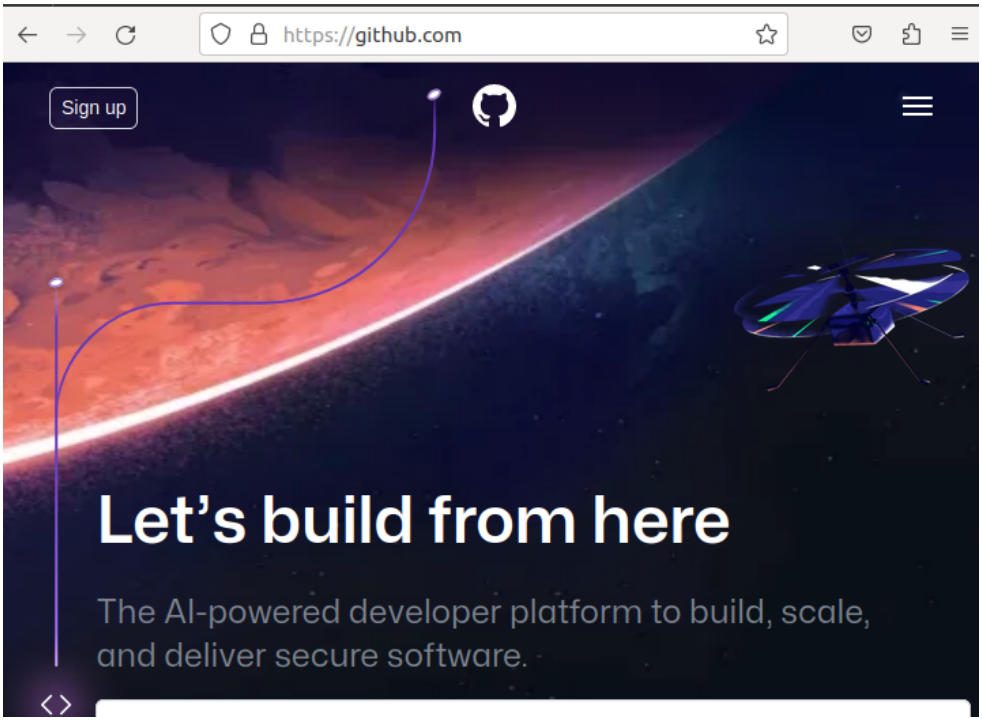
2. After the installation, issue the command *which git* again. The directory of git is usually installed in this location: *user/bin/git.*

```
flores@workstation:~$ which git
/usr/bin/git
```

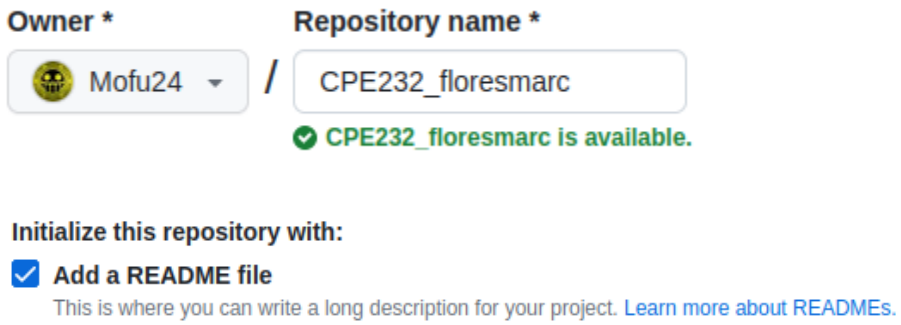
3. The version of git installed in your device is the latest. Try issuing the command *git --version* to know the version installed.

```
flores@workstation:~$ git --version
git version 2.17.1
```

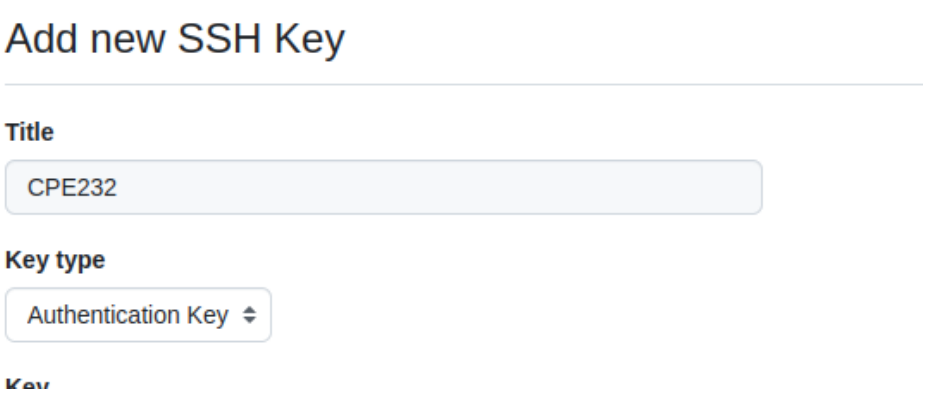
4. Using the browser in the local machine, go to www.github.com.



5. Sign up in case you don't have an account yet. Otherwise, login to your GitHub account.
    a. Create a new repository and name it as CPE232_yourname. Check Add a README file and click Create repository.



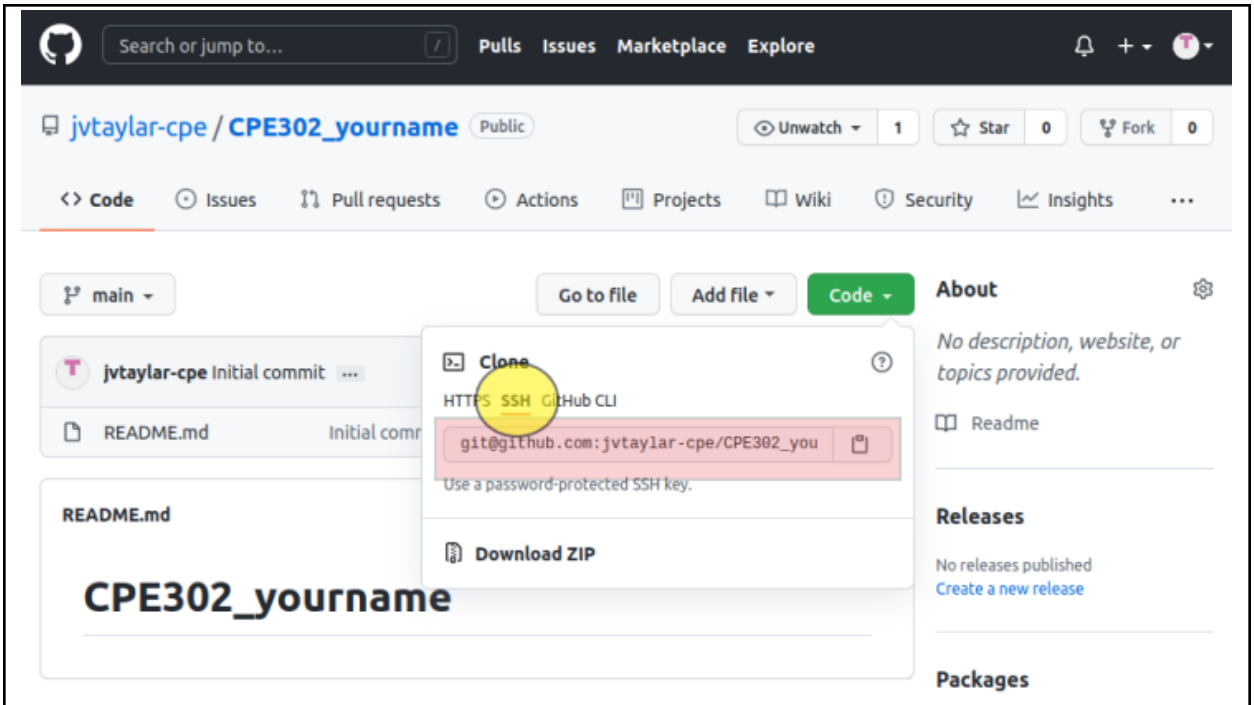
    b. Create a new SSH key on GitHub. Go your profile's setting and click SSH and GPG keys. If there is an existing key, make sure to delete it. To create a new SSH keys, click New SSH Key. Write CPE232 key as the title of the key.



    c. On the local machine's terminal, issue the command cat .ssh/id_rsa.pub and copy the public key. Paste it on the GitHub key and press Add SSH key.

```
flores@workstation:~$ cat .ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAACAQDBsUqLPXrwNNlYHc2bf964FNyugL/wRgOTkCdCquL
CqayPANthO39us+cIjI8Y0YaRAxebZjkoPgAqW7D9X4Dj7AO+FeXWcdic47RlH6SqL16t+5Pf5iVs5l
/QyBNU877BbtittrXRfSNLHDtYMTFMEi0lkYkLXVwNy5o+i6Vii93feXzQSaFDuCjND4m286R6FVSLo
faBQkIbLZg7YYQ7DJfFMus1TPxYLVXbqOK7DklaRvC/3O2U+8Z0oEV+F/sIjf8WwRi8OkdW+sc2qVkY
KS7l9zSNsGr+IP8fDE9Um253G6zRuFq13rY32BoWHsHm6dNtZMQeJdwviCo3DLZIiWON68TESkeu3Dt
H0fC+VepWBzTj2Dw8mUdxvIwIWlzvrkU0DyFhRXrB20O053NH689AWsD/SK88VW8TeHyshpqL1uKPn8
ykJdyq/Y0Z8UteP+82TRjw048ea5hJ6UOfdZ5wtOXlo2BMiVOwrnFdwL4FJWkplbBA84jIGFgNQTO+k
GAUbSrnGgIs9e1h86dT326Y0nmeWQDyuHmdUC/hiVIv03nlfAIkUDBXUGMpqG7zQv+D9NZan9xpdNh8
hNIQDLBvA4aEnRXzxZII2/2cMwylrYeQe5l0gP/xwOX58ZByVPgAH2vQGQbzX3ZguIShahLapF9/dhm
TmShMbPhWIQ== flores@workstation
```

    d. Clone the repository that you created. In doing this, you need to get the link from GitHub. Browse to your repository as shown below. Click on the Code drop down menu. Select SSH and copy the link.



e. Issue the command git clone followed by the copied link. For example, *git clone git@github.com:jvtaylar-cpe/CPE232_yourname.git*. When prompted to continue connecting, type yes and press enter.

```
flores@workstation:~$ git clone git@github.com:Mofu24/CPE232_floresmarc.git
Cloning into 'CPE232_floresmarc'...
The authenticity of host 'github.com (20.205.243.166)' can't be established.
ECDSA key fingerprint is SHA256:p2QAMXNIC1TJYWeIOttrVc98/R1BUFWu3/LiyKgUfQM.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'github.com,20.205.243.166' (ECDSA) to the list of k
nown hosts.
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.
```

f. To verify that you have cloned the GitHub repository, issue the command *ls*. Observe that you have the CPE232_yourname in the list of your directories. Use CD command to go to that directory and LS command to see the file README.md.

```
flores@workstation:~$ ls
CPE232_floresmarc  Documents  examples.desktop  Music     Project  Templates
Desktop            Downloads  main.py           Pictures  Public   Videos
flores@workstation:~$ cd CPE232_floresmarc
flores@workstation:~/CPE232_floresmarc$ ls
README.md
flores@workstation:~/CPE232_floresmarc$
```

g. Use the following commands to personalize your git.
   - *git config --global user.name "Your Name"*
   - *git config --global user.email yourname@email.com*

- Verify that you have personalized the config file using the command *cat ~/.gitconfig*

```
flores@workstation:~/CPE232_floresmarc$ git config --global user.name "floresma
rc"
flores@workstation:~/CPE232_floresmarc$ git config --global user.email "qmouflo
res@tip.edu.ph"
flores@workstation:~/CPE232_floresmarc$ cat ~/.gitconfig
[user]
        name = floresmarc
        email = qmouflores@tip.edu.ph
flores@workstation:~/CPE232_floresmarc$
```

h. Edit the README.md file using nano command. Provide any information on the markdown file pertaining to the repository you created. Make sure to write out or save the file and exit.

```
 GNU nano 2.9.3                    README.md

# CPE232_floresmarc
# Activity 2 - Git

                      [ Wrote 2 lines ]
^G Get Help  ^O Write Out  ^W Where Is  ^K Cut Text  ^J Justify
^X Exit      ^R Read File  ^\ Replace   ^U Uncut Text ^T To Spell
```

i. Use the *git status* command to display the state of the working directory and the staging area. This command shows which changes have been staged, which haven't, and which files aren't being tracked by Git. Status output does not show any information regarding the committed project history. What is the result of issuing this command?

```
flores@workstation:~/CPE232_floresmarc$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   README.md

no changes added to commit (use "git add" and/or "git commit -a")
```

j. Use the command *git add README.md* to add the file into the staging area.

```
flores@workstation:~/CPE232_floresmarc$ git add README.md
```

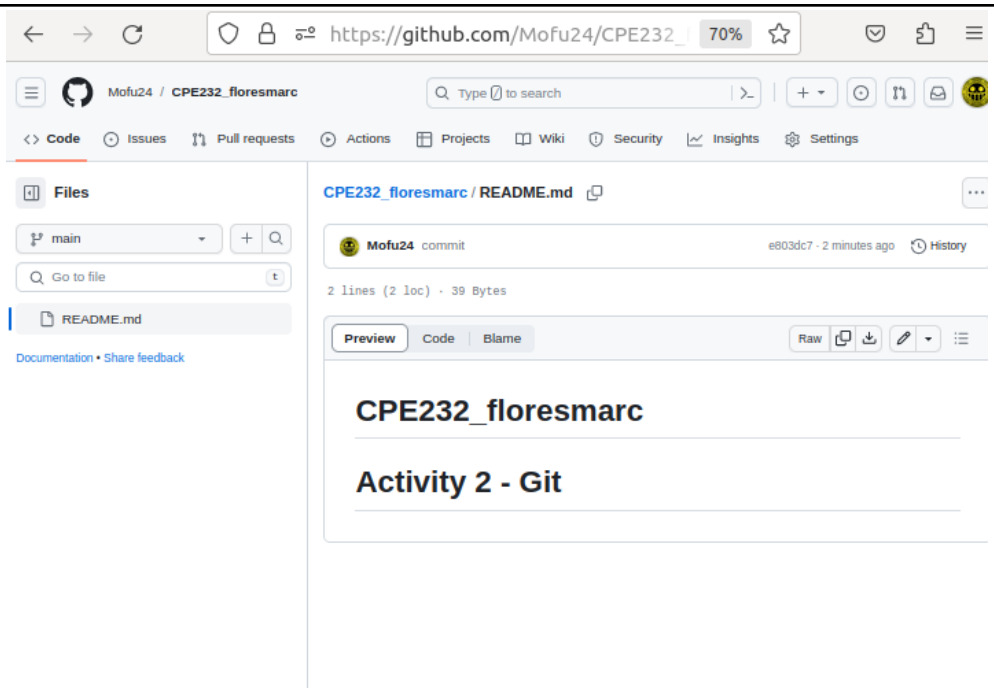
k. Use the *git commit -m "your message"* to create a snapshot of the staged changes along the timeline of the Git projects history. The use of this command is required to select the changes that will be staged for the next commit.

```
flores@workstation:~/CPE232_floresmarc$ git commit -m "commit"
[main e803dc7] commit
 1 file changed, 2 insertions(+), 1 deletion(-)
```

l. Use the command *git push <remote><branch>* to upload the local repository content to GitHub repository. Pushing means to transfer commits from the local repository to the remote repository. As an example, you may issue *git push origin main*.

```
flores@workstation:~/CPE232_floresmarc$ git push origin main
Counting objects: 3, done.
Writing objects: 100% (3/3), 273 bytes | 273.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To github.com:Mofu24/CPE232_floresmarc.git
   03ff608..e803dc7  main -> main
```

m. On the GitHub repository, verify that the changes have been made to README.md by refreshing the page. Describe the README.md file. You can notice the how long was the last commit. It should be some minutes ago and the message you typed on the git commit command should be there. Also, the README.md file should have been edited according to the text you wrote.



Reflections:

Answer the following:

3. What sort of things have we so far done to the remote servers using ansible commands?
   - We add new inputs into the README.md file that is created in the repository in github and modified it using the ubuntu terminal.

4. How important is the inventory file?
   - It is very important because this is a file that lists all of the hosts or devices that Ansible can manage. It can be a text, a ile, or a dynamic source that Ansible can query

Conclusions/Learnings:

I learned how to configure remote and local machines to connect via ssh using keys instead of using passwords, manage to create a public and private key, verify their connectivities, set up a github repository, and also configure and run commands from local machines to remote servers.