Name: Pacinos, Angela Monique A.	**Date Performed:** 08-22-23
Course/Section: CPE232 - CPE31S4	**Date Submitted:** 08-29-23
Instructor: Dr. Jonathan V. Taylar	**Semester and SY:** 1st Sem ‘23 - ‘24

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).**

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.

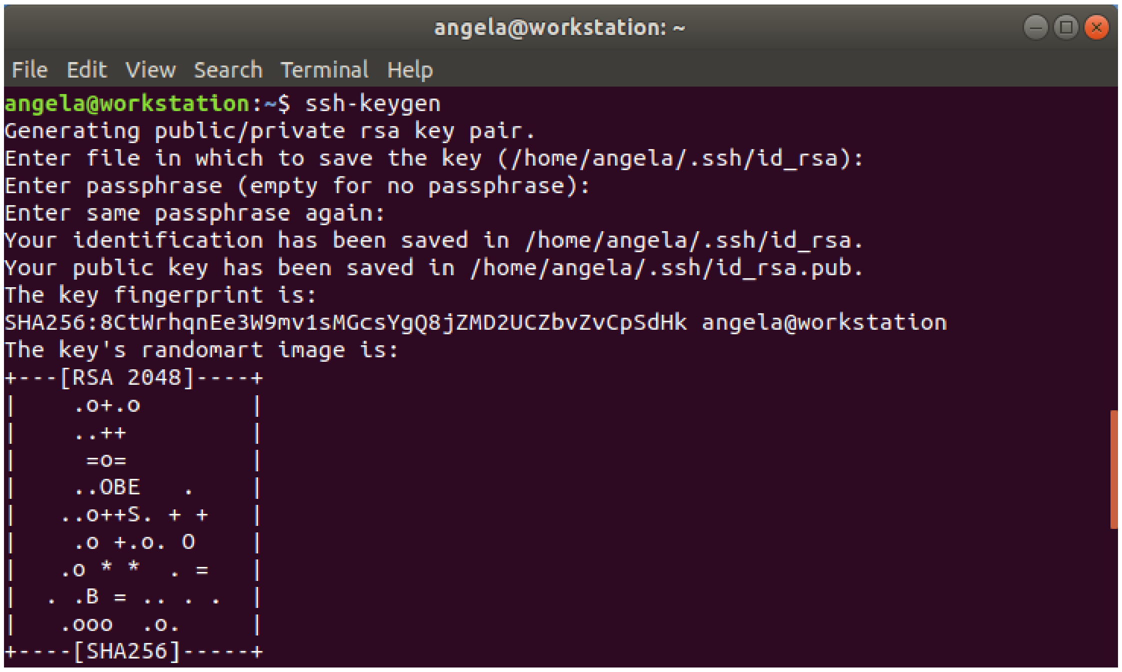
SSH introduced public key authentication as a more secure alternative to the older .rhosts authentication. It improved security by avoiding the need to have password stored in files and eliminated the possibility of a compromised server stealing the user's password.

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

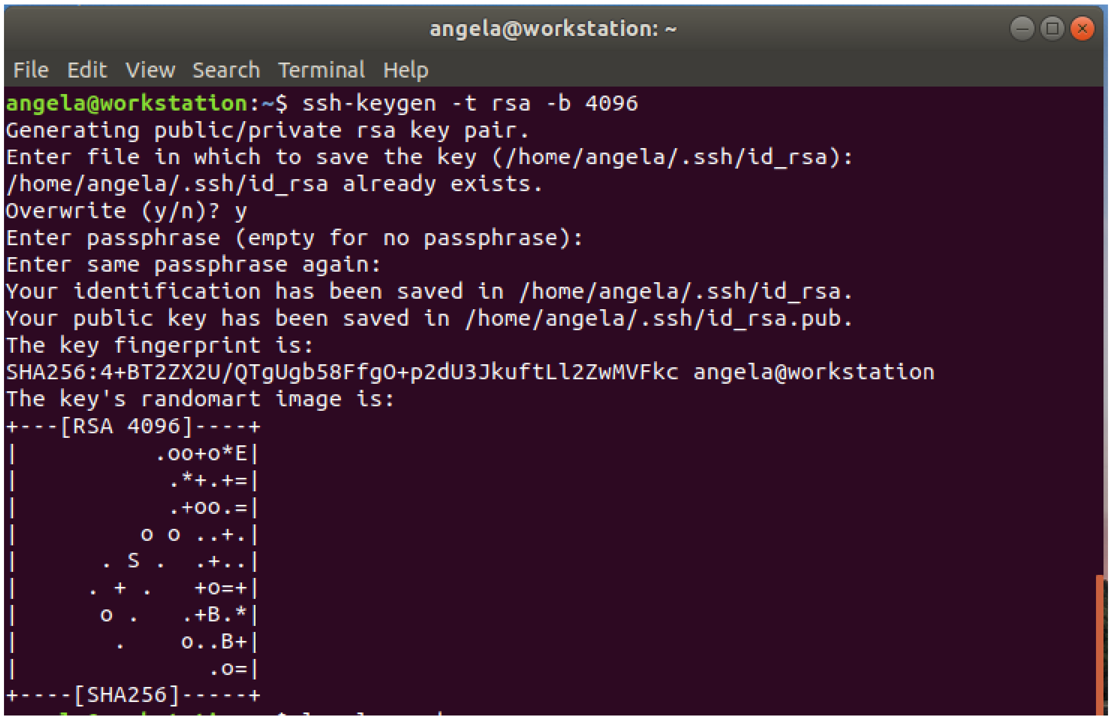
1. The simplest way to generate a key pair is to run *ssh-keygen* without arguments. In this case, it will prompt for the file in which to store keys. First,

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*.



```
angela@workstation:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/angela/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/angela/.ssh/id_rsa.
Your public key has been saved in /home/angela/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:8CtWrhqnEe3W9mv1sMGcsYgQ8jZMD2UCZbvZvCpSdHk angela@workstation
The key's randomart image is:
+---[RSA 2048]----+
|    .o+.o        |
|    ..++         |
|     =o=         |
|    ..OBE   .    |
|   ..o++S. + +   |
|    .o +.o. O    |
|   .o * *  . =   |
|  . .B = .. . .  |
|   .ooo  .o.     |
+----[SHA256]-----+
```

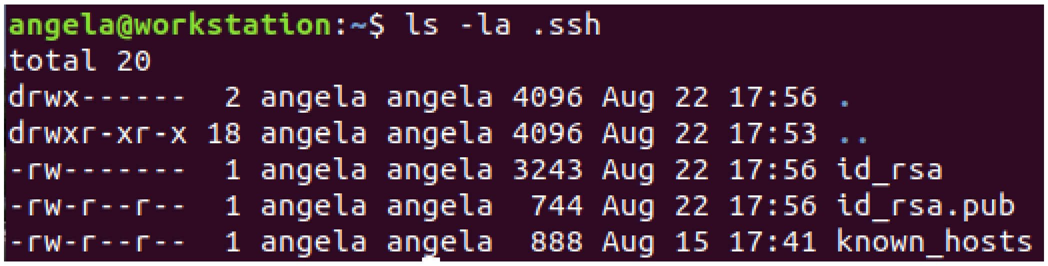
2. Issue the command *ssh-keygen -t rsa -b 4096.* The algorithm is selected using the -t option and key size using the -b option.



```
angela@workstation:~$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/angela/.ssh/id_rsa):
/home/angela/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/angela/.ssh/id_rsa.
Your public key has been saved in /home/angela/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:4+BT2ZX2U/QTgUgb58FfgO+p2dU3JkuftLl2ZwMVFkc angela@workstation
The key's randomart image is:
+---[RSA 4096]----+
|          .oo+o*E|
|           .*+.+=|
|           .+oo.=|
|         o o ..+.|
|      . S .  .+..|
|     . + .   +o=+|
|      o .    .+B.*|
|       .    o..B+|
|              .o=|
+----[SHA256]-----+
```

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.

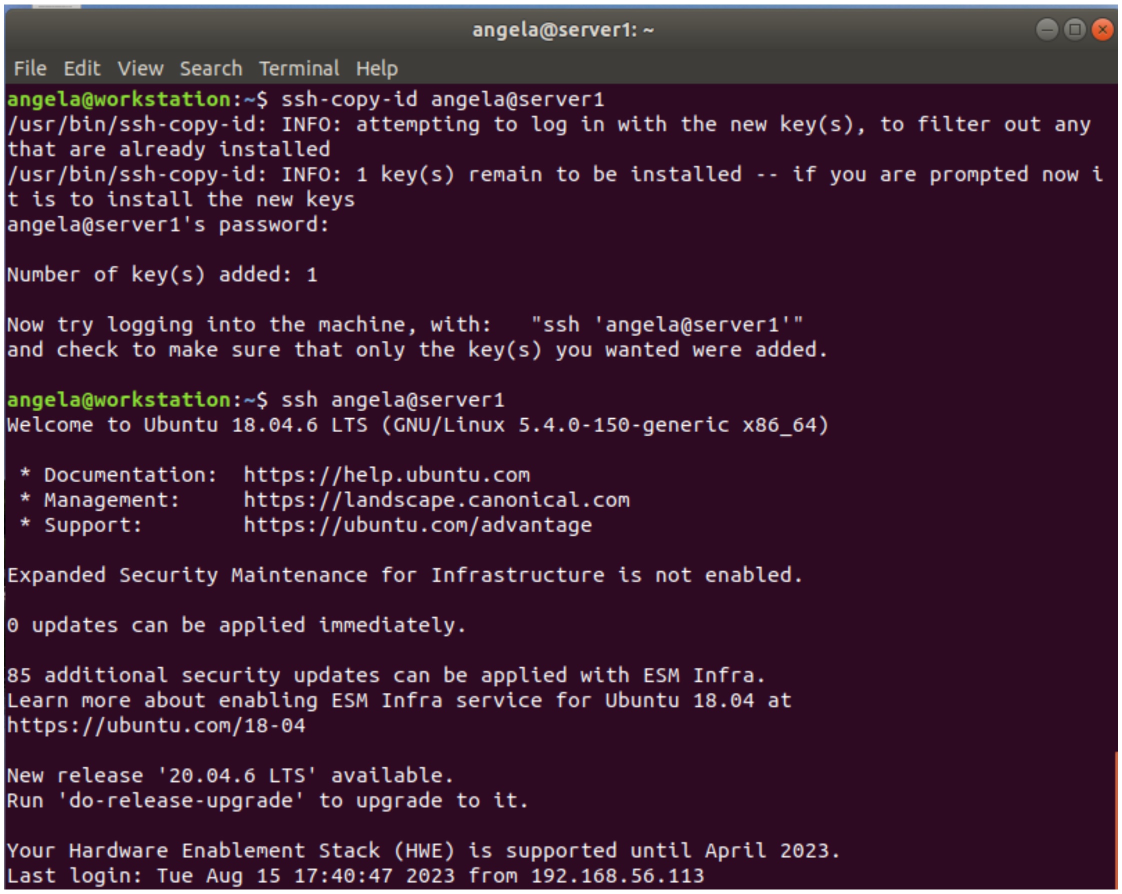
4. Verify that you have created the key by issuing the command *ls -la .ssh.* The command should show the .ssh directory containing a pair of keys. For example, id_rsa.pub and id_rsa.



```
angela@workstation:~$ ls -la .ssh
total 20
drwx------  2 angela angela 4096 Aug 22 17:56 .
drwxr-xr-x 18 angela angela 4096 Aug 22 17:53 ..
-rw-------  1 angela angela 3243 Aug 22 17:56 id_rsa
-rw-r--r--  1 angela angela  744 Aug 22 17:56 id_rsa.pub
-rw-r--r--  1 angela angela  888 Aug 15 17:41 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.



```
angela@workstation:~$ ssh-copy-id angela@server1
/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 prompted now i
t is to install the new keys
angela@server1's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'angela@server1'"
and check to make sure that only the key(s) you wanted were added.

angela@workstation:~$ ssh angela@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 15 17:40:47 2023 from 192.168.56.113
```

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

Reflections:
Answer the following:

- How will you describe the ssh-program? What does it do?
  - Ssh is a network protocol that provides security on systems to ensure the safety of access and management. It encrypted the sensitive information that the administrator wants to hide from the public eye as well as to not easily be hacked. This is done by command executions, enabling security for the system and other ways. This is to make sure that the systems data, confidentiality, and devices are all well secure from possible unauthorized access.

- How do you know that you already installed the public key to the remote servers?
  - The easiest way to check if we already installed the public keys is to use the ls -la .ssh command. We should be able to see there the files ending with id_rsa or id_rsa.pub. These files means that the authorized_key file is already there and installed.

Part 2: Discussion

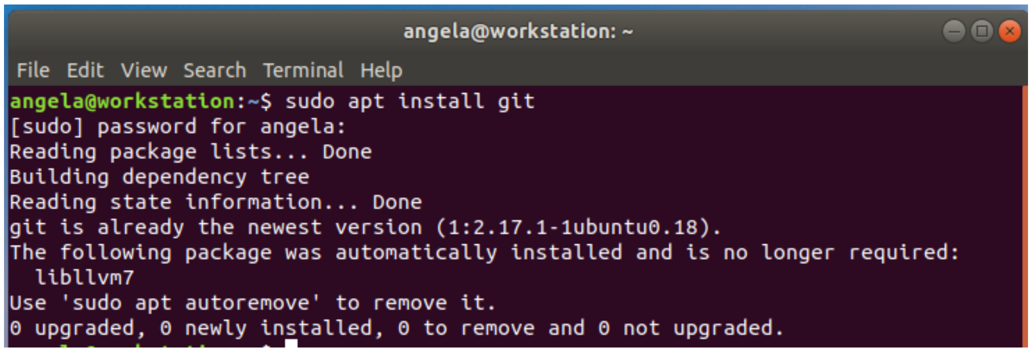
It is assumed that you are done with the last activity (**Activity 2: SSH Key-Based Authentication**).

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

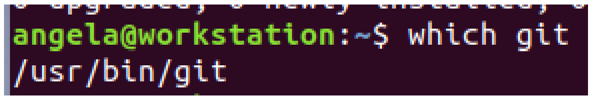
Task 3: Set up the Git Repository

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*



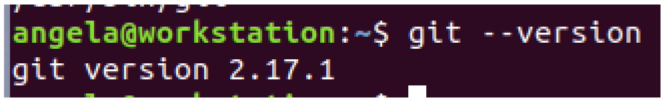
```
angela@workstation:~$ sudo apt install git
[sudo] password for angela:
Reading package lists... Done
Building dependency tree
Reading state information... Done
git is already the newest version (1:2.17.1-1ubuntu0.18).
The following package was automatically installed and is no longer required:
  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*.



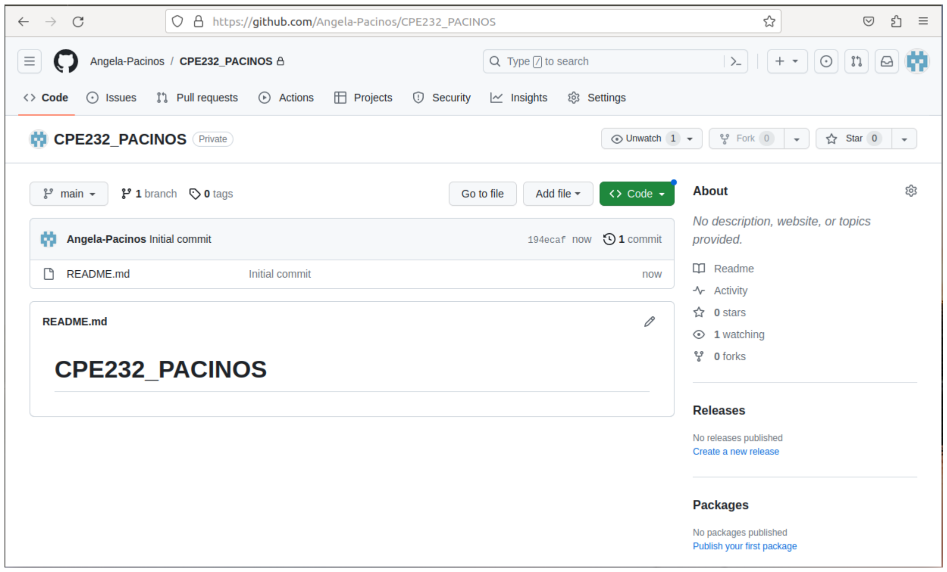
```
angela@workstation:~$ which git
/usr/bin/git
```

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



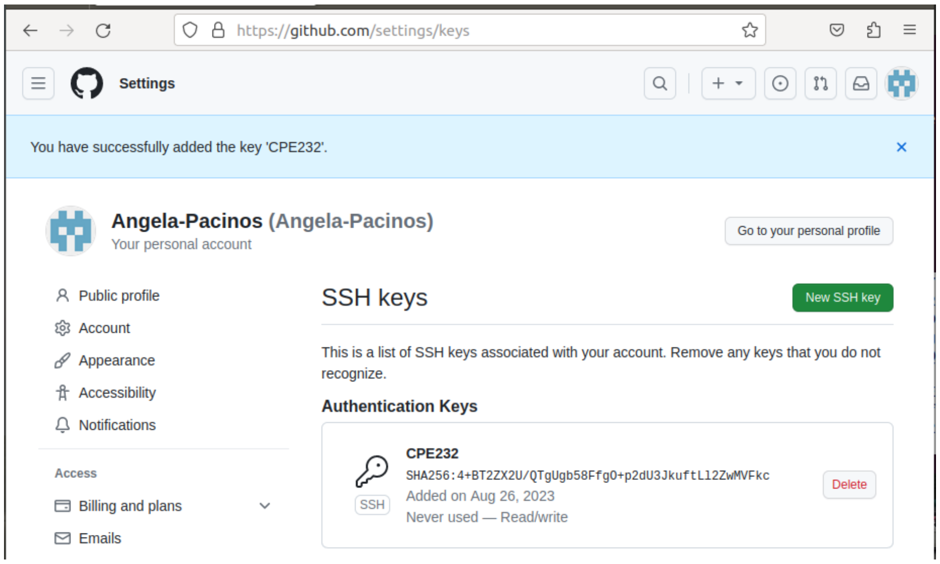
```
angela@workstation:~$ git --version
git version 2.17.1
```

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.



You have successfully added the key 'CPE232'.

Angela-Pacinos (Angela-Pacinos)
Your personal account

SSH keys

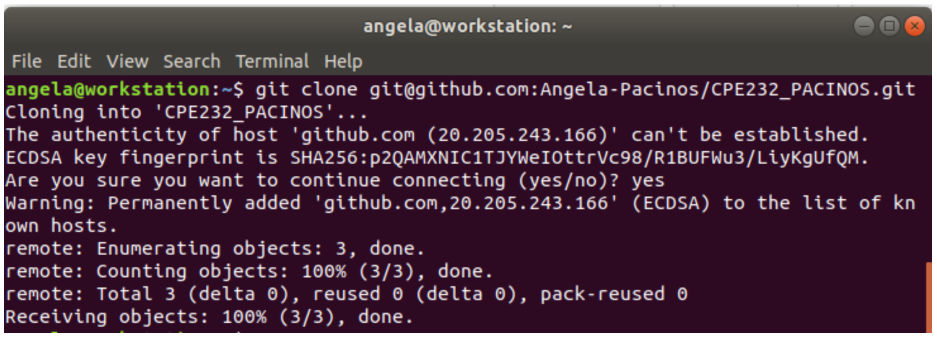
This is a list of SSH keys associated with your account. Remove any keys that you do not recognize.

Authentication Keys

CPE232
SHA256:4+BT2ZX2U/QTgUgb58Ffg0+p2dU3JkuftLl2ZwMVFkc
Added on Aug 26, 2023
Never used — Read/write

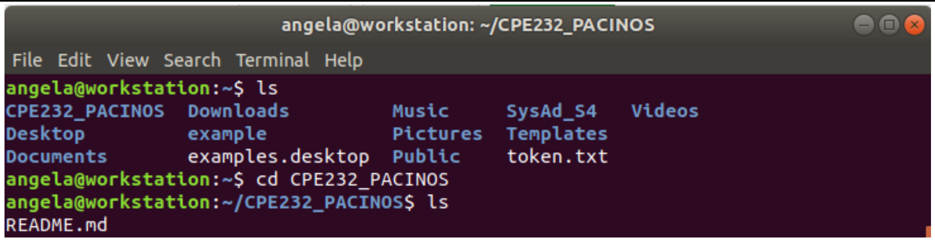
d. Clone the repository that you created. In doing this, you need to get the link from GitHub. Browse to your repository as shown below. Click on the Code drop down menu. Select SSH and copy the link.

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.



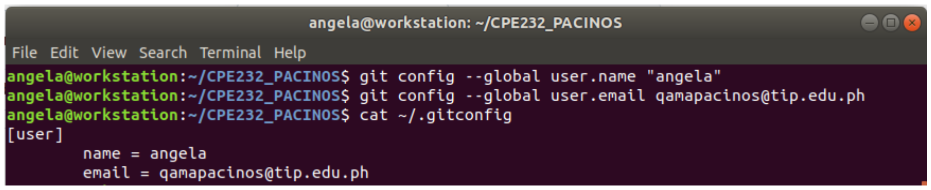
```
angela@workstation:~$ git clone git@github.com:Angela-Pacinos/CPE232_PACINOS.git
Cloning into 'CPE232_PACINOS'...
The authenticity of host 'github.com (20.205.243.166)' can't be established.
ECDSA key fingerprint is SHA256:p2QAMXNIC1TJYWeIOttrVc98/R1BUFWu3/LiyKgUfQM.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'github.com,20.205.243.166' (ECDSA) to the list of kn
own 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.



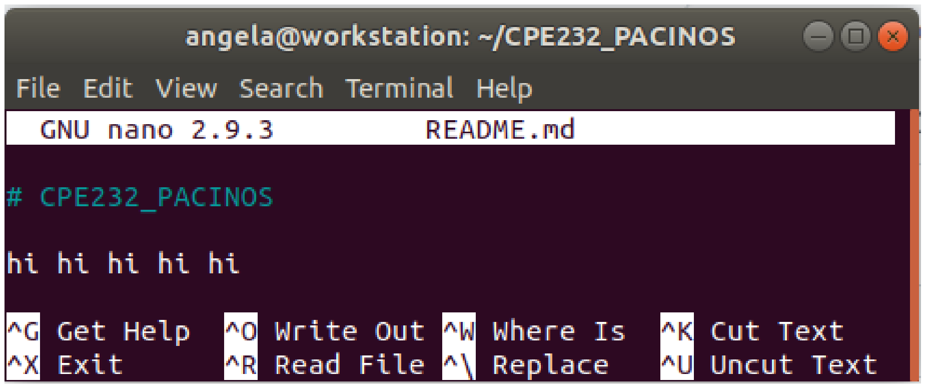
```
angela@workstation:~$ ls
CPE232_PACINOS  Downloads          Music      SysAd_S4   Videos
Desktop         example            Pictures   Templates
Documents       examples.desktop   Public     token.txt
angela@workstation:~$ cd CPE232_PACINOS
angela@workstation:~/CPE232_PACINOS$ ls
README.md
```

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*



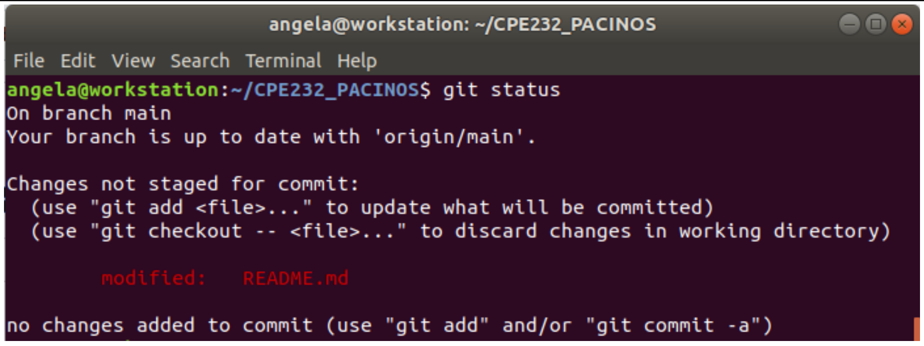
```
angela@workstation:~/CPE232_PACINOS$ git config --global user.name "angela"
angela@workstation:~/CPE232_PACINOS$ git config --global user.email qamapacinos@tip.edu.ph
angela@workstation:~/CPE232_PACINOS$ cat ~/.gitconfig
[user]
        name = angela
        email = qamapacinos@tip.edu.ph
```

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.



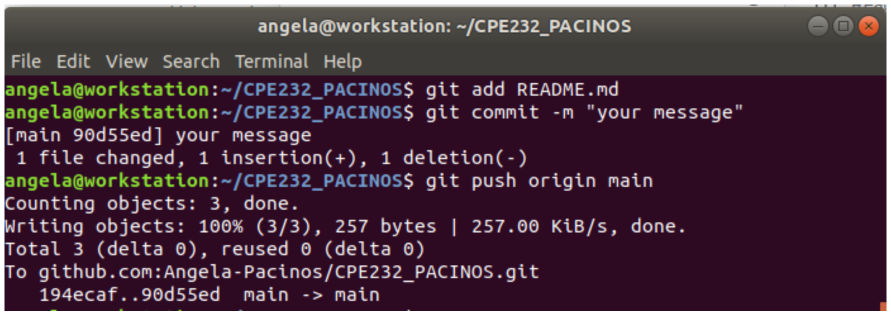
```
  GNU nano 2.9.3              README.md

# CPE232_PACINOS

hi hi hi hi hi

^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text
^X Exit       ^R Read File  ^\ Replace    ^U Uncut Text
```

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?



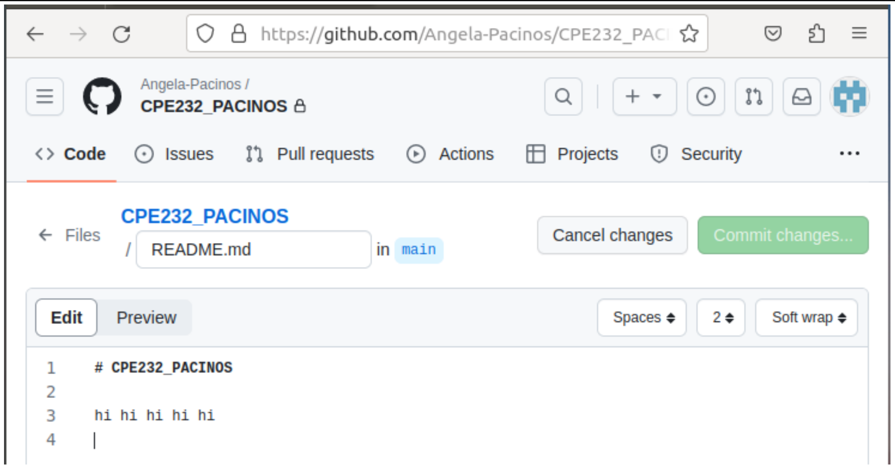
```
angela@workstation:~/CPE232_PACINOS$ git status
On branch main
Your branch is up to date with 'origin/main'.

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.
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.
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.*



```
angela@workstation:~/CPE232_PACINOS$ git add README.md
angela@workstation:~/CPE232_PACINOS$ git commit -m "your message"
[main 90d55ed] your message
 1 file changed, 1 insertion(+), 1 deletion(-)
angela@workstation:~/CPE232_PACINOS$ git push origin main
Counting objects: 3, done.
Writing objects: 100% (3/3), 257 bytes | 257.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To github.com:Angela-Pacinos/CPE232_PACINOS.git
   194ecaf..90d55ed  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 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.



```
# CPE232_PACINOS

hi hi hi hi hi

```

Reflections:
Answer the following:

- What sort of things have we so far done to the remote servers using ansible commands?
  - Some of the things that we have already created for the remote server is the public and private key. These are ways we can authenticate when we switch from servers to another one without compromising the security and in a much easier way. Another one, is that we were also able to connect the github account and repository from the website directly to the CLI. We can now access the different repositories on our github account because of the local connection that we did.

- How important is the inventory file?
  - Managing your system inventory is quite important especially in managing and storing the remote hosts that we can connect to. Having all the hostnames, usernames, etc. stored in an inventory allows us to be efficient in managing the servers.

Conclusions/ Learnings:

The very first step that we did is to create a public and private key for us to have easier access on our systems, remote and local machine without needing to always use a password. These keys are encrypted for security and can be managed by an administrator easily. Next is we connect our github repository to our local machine. With this we can easily make changes in everything that we have to on our github using the CLI and if committed properly the changes we make appear on the git repository website. There are errors at first for me but the simple omit and use of sudo solve it. Overall this module taught us how to use the different SSH keys and the connection of the git repository that we will surely be using for the next modules.