

Name: Flores, Joshua Mico A.	Date Performed: 08/22/23
Course/Section: CPE31S4	Date Submitted: 08/25/23
Instructor: Dr. Jonathan V. Taylar	Semester and SY: 2023-2024
Activity 2: SSH Key-Based Authentication and Setting up Git	
<p>1. Objectives:</p> <ul style="list-style-type: none"> 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 	
<p>Part 1: Discussion</p> <p>It is assumed that you are already done with the last Activity (Activity 1: Configure Network using Virtual Machines). <i>Provide screenshots for each task.</i></p> <p>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.</p> <p>What is ssh-keygen?</p> <p>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.</p> <p>SSH Keys and Public Key Authentication</p> <p>The SSH protocol uses public key cryptography for authenticating hosts and users. The authentication keys, called SSH keys, are created using the keygen program.</p> <p>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.</p> <p>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.</p>	
<p>Task 1: Create an SSH Key Pair for User Authentication</p> <ul style="list-style-type: none"> 1. The simplest way to generate a key pair is to run <i>ssh-keygen</i> 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`.

```
micoflores@workstation:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/micoflores/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/micoflores/.ssh/id_rsa
Your public key has been saved in /home/micoflores/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:n6fED0VRETM4zx5bsYPX6eJ0vuoyVSWwbFia6GcxHCQ micoflores@workstat
The key's randomart image is:
+---[RSA 3072]---+
|                E.o +---+ |
|                + B.o+ . |
|                . B =. o. |
|                .  =.  o |
|                S o .oo = |
|                = o o+=o |
|                * ooo=. |
|                . B..=. |
|                . *+oo. |
+-----[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.

```
micoflores@workstation:~$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
```

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.

```

micoflores@workstation:~$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/micoflores/.ssh/id_rsa):
/home/micoflores/.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/micoflores/.ssh/id_rsa
Your public key has been saved in /home/micoflores/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:Vtc8R0wGWnNFIFLCN5+56AkIIIqRShdbvUUVGBNA5K8Y micoflores@workstati
The key's randomart image is:
+---[RSA 4096]---+
| . ..o+.*o.o =*o|
|o...+ =...oo* =.|
|+o.o.. = ..oo+o.|
|+   E.o . . +o |
|   . ..S. . . |
|   . . . . . |
|   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`.

```

micoflores@workstation:~$ ls -la .ssh
total 24
drwx----- 2 micoflores micoflores 4096 Aug 22 21:31 .
drwxr-x--- 17 micoflores micoflores 4096 Aug 14 23:18 ..
-rw----- 1 micoflores micoflores 3389 Aug 22 21:32 id_rsa
-rw-r--r-- 1 micoflores micoflores 748 Aug 22 21:32 id_rsa.pub
-rw----- 1 micoflores micoflores 2240 Aug 15 00:07 known_hosts
-rw----- 1 micoflores micoflores 1120 Aug 15 00:03 known_hosts.old

```

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.

```

Connection to server2 closed.
micoflores@workstation:~$ ssh-copy-id
Usage: /usr/bin/ssh-copy-id [-h|-?|-f|-n|-s] [-i [identity_file]] [-p port]
alternative ssh_config file] [[-o <ssh -o options>] ...] [user@]hostname
-f: force mode -- copy keys without trying to check if they are al
installed
-n: dry run -- no keys are actually copied
-s: use sftp -- use sftp instead of executing remote-commands. C
useful if the remote only allows sftp
-h|-?: print this help

```

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

```
micoflores@workstation:~$ ssh-copy-id -i ~/.ssh/id_rsa micoflores@workstation
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/micoflores/.ssh/id_rsa.pub"
The authenticity of host 'workstation (127.0.0.1)' can't be established.
ED25519 key fingerprint is SHA256:Wk51+W2pG8UgnJLKNkLez832+I5Z+rFpHcyX2PvHi.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to find out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
micoflores@workstation's password:

Number of key(s) added: 1

Now try logging into the machine, with:  "ssh 'micoflores@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.

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

Number of key(s) added: 1

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

```
micoflores@workstation:~$ ssh-copy-id -i ~/.ssh/id_rsa micoflores@server2
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/micoflores/.ssh/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 prompted now it is to install the new keys
micoflores@server2's password:

Number of key(s) added: 1

Now try logging into the machine, with: "ssh 'micoflores@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?

Server 1

```
micoflores@workstation:~$ ssh micoflores@server1
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 6.2.0-26-generic x86_64)

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

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

The list of available updates is more than a week old.
To check for new updates run: sudo apt update
Last login: Tue Aug 15 00:53:48 2023 from 192.168.56.104
micoflores@server1:~$ exit
```

Server 2

```
micoflores@workstation:~$ ssh micoflores@server2
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 6.2.0-26-generic x86_64)

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

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

The list of available updates is more than a week old.
To check for new updates run: sudo apt update
Last login: Tue Aug 15 00:46:20 2023 from 192.168.56.104
micoflores@server2:~$
```

The workstation accessed the 2 servers remotely without having to enter the password since the public key was copied to the servers. Therefore, connecting to servers can be less hassle because of the public key.

Reflections:

Answer the following:

1. How will you describe the ssh-program? What does it do?
 - It is a type of secured connection to have communication between computers and servers. It is secured because of its uniqueness that the only one who can access the servers is the one who is authorized.
2. How do you know that you already installed the public key to the remote servers?
 - By using the ssh-copy-id tool, the key must be copied to the designated servers from the main host. Then, by entering the server's password for the first time, the control node will no longer need to input the password for the second time.

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*

```
micoflores@server2:~$ sudo apt install git
[sudo] password for micoflores:
Sorry, try again.
[sudo] password for micoflores:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and
  libflashrom1 libftdi1-2 libllvm13
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  git-man liberror-perl
Suggested packages:
  git-daemon-run | git-daemon-svsvinit git-doc git-email
```

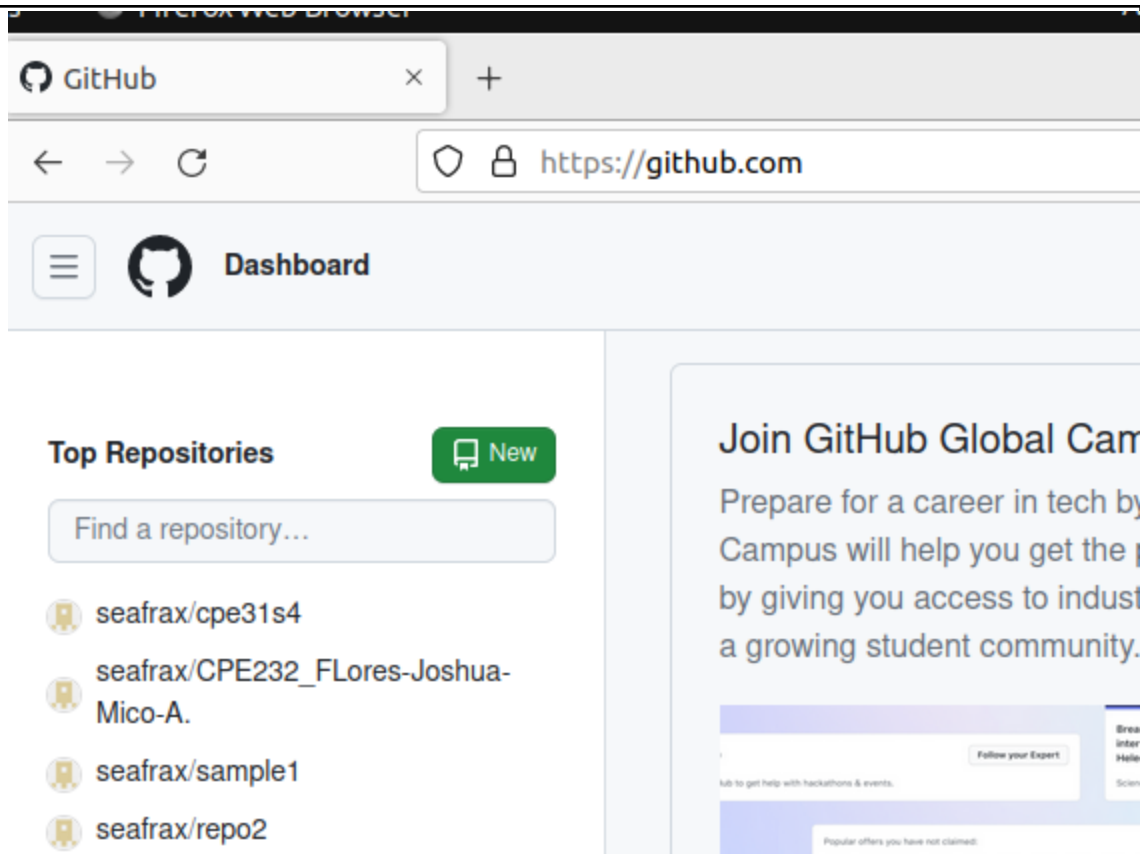
2. After the installation, issue the command *which git* again. The directory of git is usually installed in this location: *user/bin/git*.

```
micoflores@server2:~$ 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.

```
micoflores@server2:~$ git --version
git version 2.34.1
```

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.

https://github.com/new

Owner *

seafrax

Repository name *

CPE232_FLores-Joshua-Mico-A.

✓ CPE232_FLores-Joshua-Mico-A. is available.

Great repository names are short and memorable. Need inspiration? How about [super-duper-enig](#)

Description (optional)

☒  **Public**

Anyone on the internet can see this repository. You choose who can commit.

☐  **Private**

You choose who can see and commit to this repository.

Initialize this repository with:

☒ **Add a README file**

This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

.gitignore template:None

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

License:None

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

This will set `main` as the default branch. Change the default name in your [settings](#).



CPE232_FLores-Joshua-Mico-A.

Public



Pin

 main

 1 branch

 0 tags

Go to file

Add file



seafrax Initial commit

4dd8f64 no



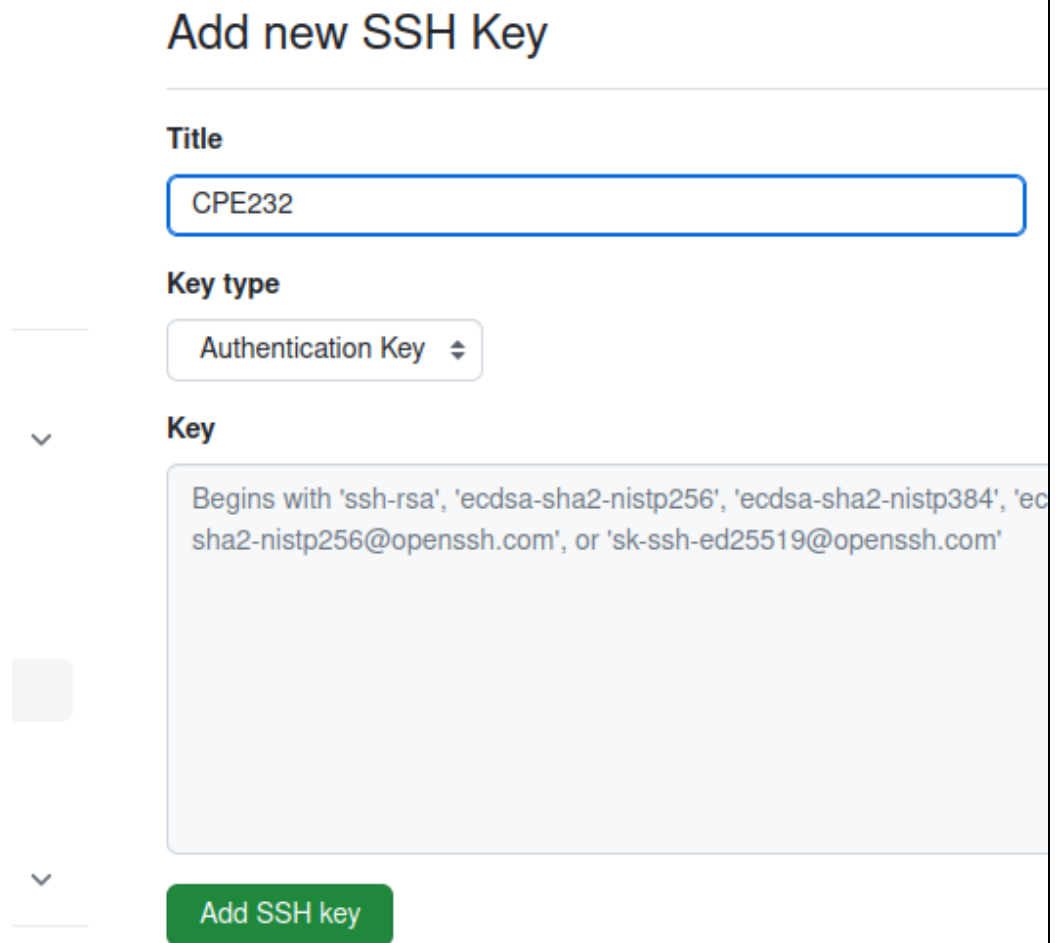
README.md

Initial commit

README.md

CPE232_FLores-Joshua-Mico-A.

- 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.



Add new SSH Key

Title

Key type

Authentication Key

Key

Begins with 'ssh-rsa', 'ecdsa-sha2-nistp256', 'ecdsa-sha2-nistp384', 'ecdsa-sha2-nistp256@openssh.com', or 'sk-ssh-ed25519@openssh.com'

Add SSH 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.

```

micoflores@workstation:~$ cat .ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQCAQCz1BEqbxHuN+UYe+gC1hVSG6X2ytWPB
dn4caBqqSEXadAW0Ur7fnLmCwvrVdwY/6M2CYUT64Uzo1HfyPOq5i9X2NQ0te9DKVnJfE
am5mgmgRrnqqnPoK0zkNPfy587DIP9PitG+eimAMzMfCFAU/s/lRUGtZPG7YbFmvSvKse
cOYJSeTOZI7CrDD+SL9L0ByFXnQUwA9r4kggqtLUhLSeJog0aszjm7Jgz3WeJXg04Jo0x
hfm1hSvf0gCq0+HeCzz/zNZuWOUdRMrYKWWX/AdynKp678qbARSqICuvZ1A6hS+NHLm1Q
248ynhczRwgQ4b0q+9SmpGVA/tuXpmWjyELtyIoeEX3WPP1H9PMqZEjbFmiUFnXI5fg6Y
NZEaJrIs+p73z+NhYe3WoV8TuDIUyYpP/cTYQV4m0+Ny5vXMkiSp5RD3Cw8i6EeCeETyX
X3UBxgM2fCx5S5yeQSkJh18hk3+leHa0oBnu8xY4So7XP82gqDS1CA5qZ3MqXz29sgVch
S4kC7+ypoFypvJht0CLEYqWD9neiHc9ZqRPLjaHxnrahWfta7R0V4HF9txga2juemJVFY
rQ== micoflores@workstation

```

Title

CPE232

Key type

Authentication Key

Key

```

ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAQCAQCz1BEqbxHuN+UYe+gC1hVSG6X2ytWPBCBNxemoB+ZRdn
dAW0Ur7fnLmCwvrVdwY/6M2CYUT64Uzo1HfyPOq5i9X2NQ0te9DKVnJfE+IJoEHUudr1am5mgmgRrnq
y587DIP9PitG+eimAMzMfCFAU
/s/lRUGtZPG7YbFmvSvKsedQaSBZ04skcOYJSeTOZI7CrDD+SL9L0ByFXnQUwA9r4kggqtLUhLSeJogC
eJXgO4Jo0xkvr1F3JmYm5hfm1hSvf0gCq0+HeCzz/zNZuWOUdRMrYKWWX
/AdynKp678qbARSqICuvZ1A6hS+NHLm1QvvdLByNWDJV248ynhczRwgQ4b0q+9SmpGVA
/tuXpmWjyELtyIoeEX3WPP1H9PMqZEjbFmiUFnXI5fg6YmDdbYNUC28SNZEaJrIs+p73z+NhYe3WoV8Tu
/cTYQV4m0+Ny5vXMkiSp5RD3Cw8i6EeCeETyXC5Sxvo

```

- 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.

CPE232_Flores-Joshua-Mico-A Public

Pin Unwatch 1

main 1 branch 0 tags

Go to file Add file <> Code

Local Codespaces New

Clone

HTTPS SSH GitHub CLI

git@github.com:seafraX/CPE232_Flores-Josh

Use a password-protected SSH key.

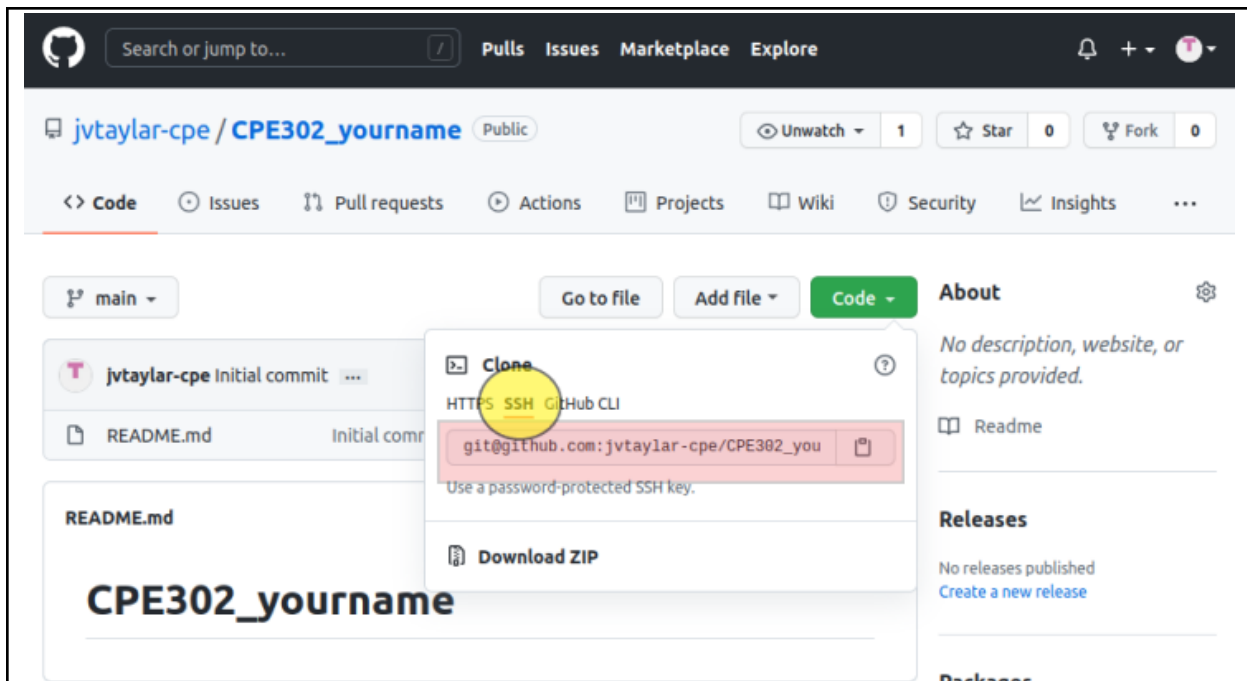
Download ZIP

seafraX Initial commit

README.md Initial commit

README.md

CPE232_Flores-Joshua-Mico-A



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

```
micoFlores@workstation:~$ git clone git@github.com:seafraX/CPE232_Flores-Joshua-
Cloning into 'CPE232_Flores-Joshua-Mico-A.'...
The authenticity of host 'github.com (20.205.243.166)' can't be established.
ED25519 key fingerprint is SHA256:+DiY3wvvV6TuJJhbpZisF/zLDA0zPMSvHdkr4UvCOqU.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'github.com' (ED25519) to the list of known 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`.

```
micoFlores@workstation:~$ ls
CPE232_Flores-Joshua-Mico-A.  Documents  Music      Public  T
Desktop                      Downloads  Pictures   snap    V
micoFlores@workstation:~$ cd CPE232_Flores-Joshua-Mico-A.
micoFlores@workstation:~/CPE232_Flores-Joshua-Mico-A.$ ls
README.md
```

- g. Use the following commands to personalize your git.

- `git config --global user.name "Your Name"`

```
micoflores@workstation: ~/CPE232_Flores-Joshua-Mico-A
micoflores@workstation:~$ git config --global user.name "joshua"
```

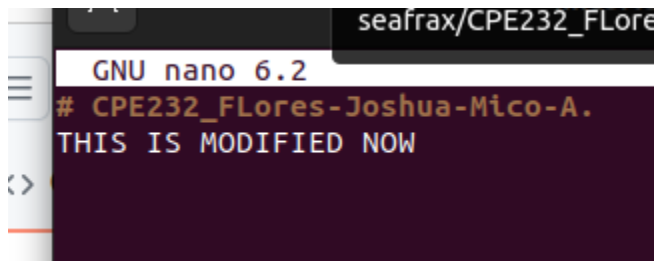
- `git config --global user.email yourname@email.com`

```
micoflores@workstation:~$ git config --global user.email qjmaflores@tip.edu.ph
```

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

```
micoflores@workstation:~/CPE232_Flores-Joshua-Mico-A$ cd
micoflores@workstation:~$ git config --global user.name "joshua"
micoflores@workstation:~$ git config --global user.email qjmaflores@tip.edu.ph
micoflores@workstation:~$ cat ~/.gitconfig
[user]
  name = joshua
  email = qjmaflores@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.



- 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?

```
[sudo] password for micoflores:
micoflores@workstation:~/CPE232_Flores-Joshua-Mico-A$ 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 restore <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.

```
micoflores@workstation:~/CPE232_Flores-Joshua-Mico-A$ 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

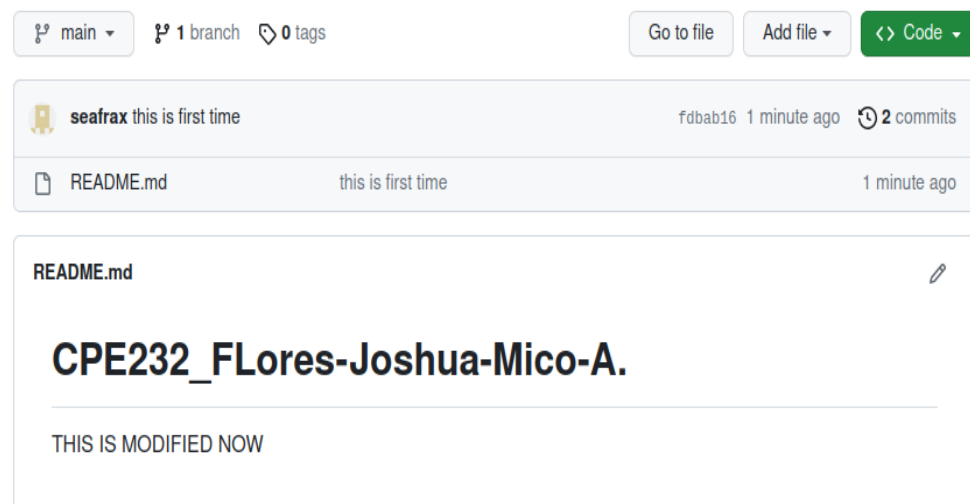
this command is required to select the changes that will be staged for the next commit.

```
micoflores@workstation:~/CPE232_FLores-Joshua-Mico-A.$ git commit -m "this is first time"
[main fdbab16] this is first time
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**.

```
micoflores@workstation:~/CPE232_FLores-Joshua-Mico-A.$ git push origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 2 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 293 bytes | 293.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:seafraX/CPE232_FLores-Joshua-Mico-A..git
4dd8f64..fdbab16 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.



The screenshot shows a GitHub repository interface. At the top, there are buttons for 'main', '1 branch', and '0 tags'. Below this, there are buttons for 'Go to file', 'Add file', and 'Code'. The repository name is 'seafraX' and the commit message is 'this is first time'. The commit hash is 'fdbab16' and it was made '1 minute ago'. There are '2 commits' in total. The file 'README.md' is selected, showing its content: 'CPE232_FLores-Joshua-Mico-A.' and 'THIS IS MODIFIED NOW'.

Reflections:

Answer the following:

3. What sort of things have we so far done to the remote servers using ansible commands?

- First, I've used the command `ssh-keygen` to generate the key so it will be used to connect to other servers. I copied the generated public from my workstation using the command `ssh-copy-id` to the remote servers and checked if the connection was working. Upon testing, it worked properly.

4. How important is the inventory file?

- This is where all the remote servers that are being managed by the host, it displays the ip addresses in order to keep track of resources in the network. So troubleshooting will be easier to do and problems are easier to find. This also shows the servers where you can perform Ansible.

Conclusions/Learnings:

After completing all the tasks that are given in this activity, I've learned how to generate ssh keys, connect to servers using ssh and access them without needing to input passwords. In addition to this, I've also learned how to manipulate files by executing basic commands on git and making changes in my repository. The connection between the linux terminal and the github repository is also implementing ssh.