

Name: Davonn P. Escobilla	Date Performed: 09/02/2022
Course/Section: CPE31S24	Date Submitted: 09/02/2022
Instructor: Dr. Jonathan Taylar	Semester and SY: 1st Sem, 2022-2023
Activity 2: SSH Key-Based Authentication and Setting up Git	
1. Objectives: <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 	
Part 1: Discussion <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>	
Task 1: Create an SSH Key Pair for User Authentication <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`.

```
davonn@workstation:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/davonn/.ssh/id_rsa): /home/davonn/.ssh/id_rsa
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/davonn/.ssh/id_rsa
Your public key has been saved in /home/davonn/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:dEC2UcxkAJhdSokhV0LI98bKsXS1C9XWG3/+2nyTM34 davonn@workstation
The key's randomart image is:
+---[RSA 3072]-----+
| ..o=B+*BB+      |
| oo=o+o+=        |
| . +.+O .+       |
| + *.. . .       |
| o B .S   o       |
| + .             |
|                 |
|                ..|
|               BE |
|              ooO  |
+----[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.

```
davonn@workstation:~$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/davonn/.ssh/id_rsa): /home/davonn/.ssh/id_rsa
/home/davonn/.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/davonn/.ssh/id_rsa
Your public key has been saved in /home/davonn/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:qLC5g7JGoh/tjAH0AKeR9HDvYdJos3Ei0s6hk55BiC4 davonn@workstation
The key's randomart image is:
+---[RSA 4096]---+
| o .           |
|+o+ +         |
|Bo+O *        |
|=Oo.X..       |
|Eoo.o. S      |
|B+o+ . .      |
|=== + .       |
|. + B o        |
|o.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`.

```
davonn@workstation:~$ ls -la .ssh
total 20
drwx----- 2 davonn davonn 4096 Sep  2 16:23 .
drwxr-xr-x 16 davonn davonn 4096 Sep  2 15:40 ..
-rw----- 1 davonn davonn 3381 Sep  2 16:24 id_rsa
-rw-r--r-- 1 davonn davonn 744 Sep  2 16:24 id_rsa.pub
```

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.

```
davonn@workstation:~$ ssh-copy-id
Usage: /usr/bin/ssh-copy-id [-h|-?|-f|-n] [-i [identity_file]] [-p port] [[-o <ssh -o options>] ...] [user@]hostname
    -f: force mode -- copy keys without trying to check if they are already 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`

Server 1

```
davonn@workstation:~$ ssh-copy-id -i ~/.ssh/id_rsa davonn@server1
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/davonn/.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
davonn@server1's password:

Number of key(s) added: 1

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

Server 2

```
davonn@workstation:~$ ssh-copy-id -i ~/.ssh/id_rsa davonn@server2
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/davonn/.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
davonn@server2's password:

Number of key(s) added: 1

Now try logging into the machine, with:  "ssh 'davonn@server2'"
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.
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?

The connection did not ask for a password since the public key is already configured between the remote hosts. It is identified and verified the access of the connection.

Server 1 and Server 2

```
davonn@workstation:~$ ssh davonn@server1
Welcome to Ubuntu 20.04.5 LTS (GNU/Linux 5.15.0-46-generic x86_64)

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

0 updates can be applied immediately.

New release '22.04.1 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Your Hardware Enablement Stack (HWE) is supported until April 2025.
Last login: Fri Sep  2 16:15:31 2022 from 192.168.56.102
davonn@workstation:~$ ssh davonn@server2
Welcome to Ubuntu 20.04.5 LTS (GNU/Linux 5.15.0-46-generic x86_64)

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

0 updates can be applied immediately.

New release '22.04.1 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Your Hardware Enablement Stack (HWE) is supported until April 2025.
Last login: Fri Sep  2 15:15:55 2022 from fe80::6190:aee8:1083:9409%enp0s8
```

Reflections:

Answer the following:

1. How will you describe the ssh-program? What does it do?
ssh-program lets the user have privacy since the local host has the control over the keys and it can be removed or added to the remote servers. It does create that extra layer of protection to avoid the password to be stolen. A good tool to strengthen the security authentication.
2. How do you know that you already installed the public key to the remote servers?
You can go use the remote servers and use `ls .ssh` command to enter the ssh directory, then open the `authorized_keys` in order to check if the public key is already installed.

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**).

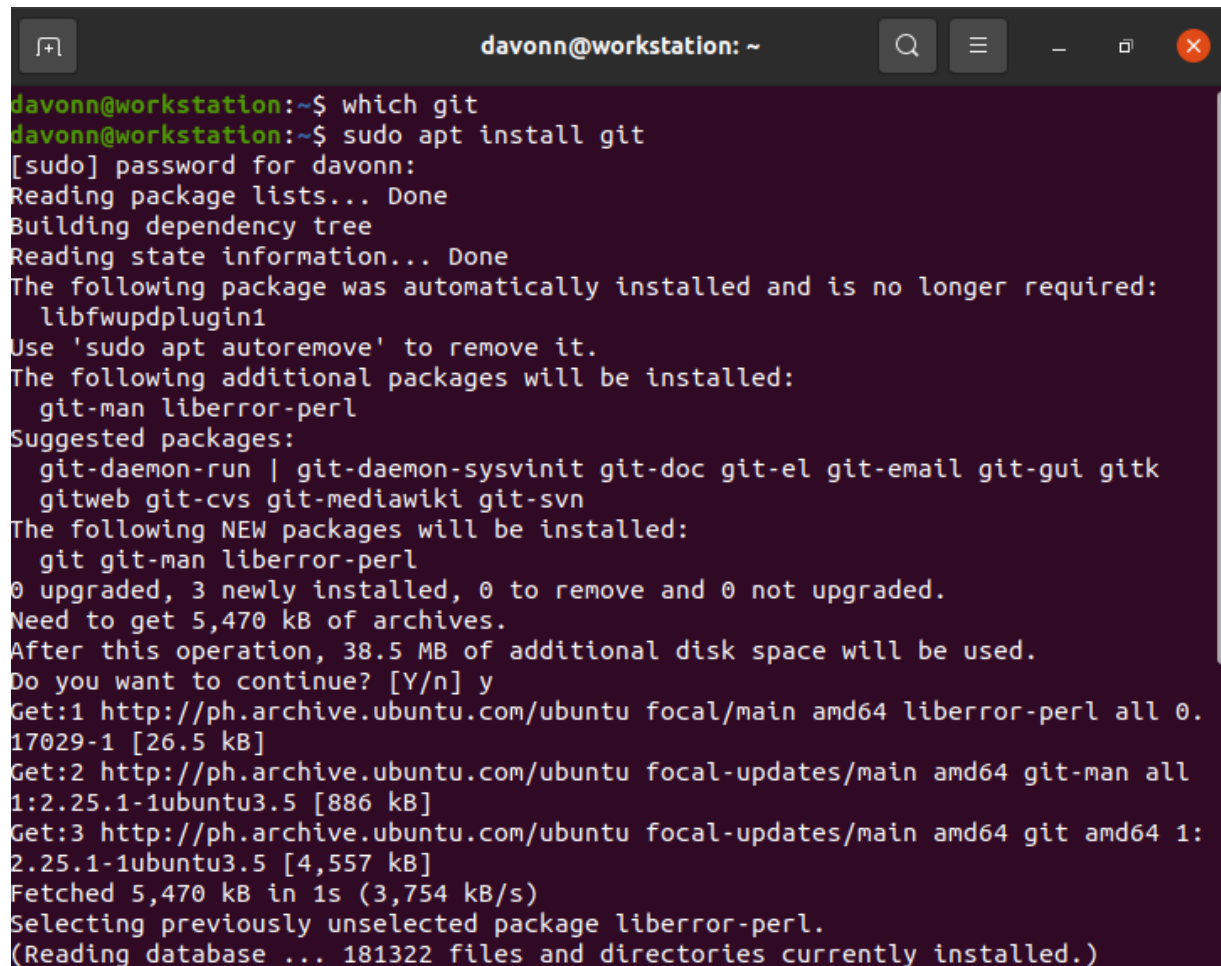
Set up Git

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*



```
davonn@workstation: ~  
davonn@workstation:~$ which git  
davonn@workstation:~$ sudo apt install git  
[sudo] password for davonn:  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
The following package was automatically installed and is no longer required:  
  libfwupdplugin1  
Use 'sudo apt autoremove' to remove it.  
The following additional packages will be installed:  
  git-man liberror-perl  
Suggested packages:  
  git-daemon-run | git-daemon-sysvinit git-doc git-el git-email git-gui gitk  
  gitweb git-cvs git-mediawiki git-svn  
The following NEW packages will be installed:  
  git git-man liberror-perl  
0 upgraded, 3 newly installed, 0 to remove and 0 not upgraded.  
Need to get 5,470 kB of archives.  
After this operation, 38.5 MB of additional disk space will be used.  
Do you want to continue? [Y/n] y  
Get:1 http://ph.archive.ubuntu.com/ubuntu focal/main amd64 liberror-perl all 0.  
17029-1 [26.5 kB]  
Get:2 http://ph.archive.ubuntu.com/ubuntu focal-updates/main amd64 git-man all  
1:2.25.1-1ubuntu3.5 [886 kB]  
Get:3 http://ph.archive.ubuntu.com/ubuntu focal-updates/main amd64 git amd64 1:  
2.25.1-1ubuntu3.5 [4,557 kB]  
Fetched 5,470 kB in 1s (3,754 kB/s)  
Selecting previously unselected package liberror-perl.  
(Reading database ... 181322 files and directories currently installed.)
```



```
davonn@workstation: ~  
git git-man liberror-perl  
0 upgraded, 3 newly installed, 0 to remove and 0 not upgraded.  
Need to get 5,470 kB of archives.  
After this operation, 38.5 MB of additional disk space will be used.  
Do you want to continue? [Y/n] y  
Get:1 http://ph.archive.ubuntu.com/ubuntu focal/main amd64 liberror-perl all 0.17029-1 [26.5 kB]  
Get:2 http://ph.archive.ubuntu.com/ubuntu focal-updates/main amd64 git-man all 1:2.25.1-1ubuntu3.5 [886 kB]  
Get:3 http://ph.archive.ubuntu.com/ubuntu focal-updates/main amd64 git amd64 1:2.25.1-1ubuntu3.5 [4,557 kB]  
Fetched 5,470 kB in 1s (3,754 kB/s)  
Selecting previously unselected package liberror-perl.  
(Reading database ... 181322 files and directories currently installed.)  
Preparing to unpack .../liberror-perl_0.17029-1_all.deb ...  
Unpacking liberror-perl (0.17029-1) ...  
Selecting previously unselected package git-man.  
Preparing to unpack .../git-man_1%3a2.25.1-1ubuntu3.5_all.deb ...  
Unpacking git-man (1:2.25.1-1ubuntu3.5) ...  
Selecting previously unselected package git.  
Preparing to unpack .../git_1%3a2.25.1-1ubuntu3.5_amd64.deb ...  
Unpacking git (1:2.25.1-1ubuntu3.5) ...  
Setting up liberror-perl (0.17029-1) ...  
Setting up git-man (1:2.25.1-1ubuntu3.5) ...  
Setting up git (1:2.25.1-1ubuntu3.5) ...  
Processing triggers for man-db (2.9.1-1) ...
```

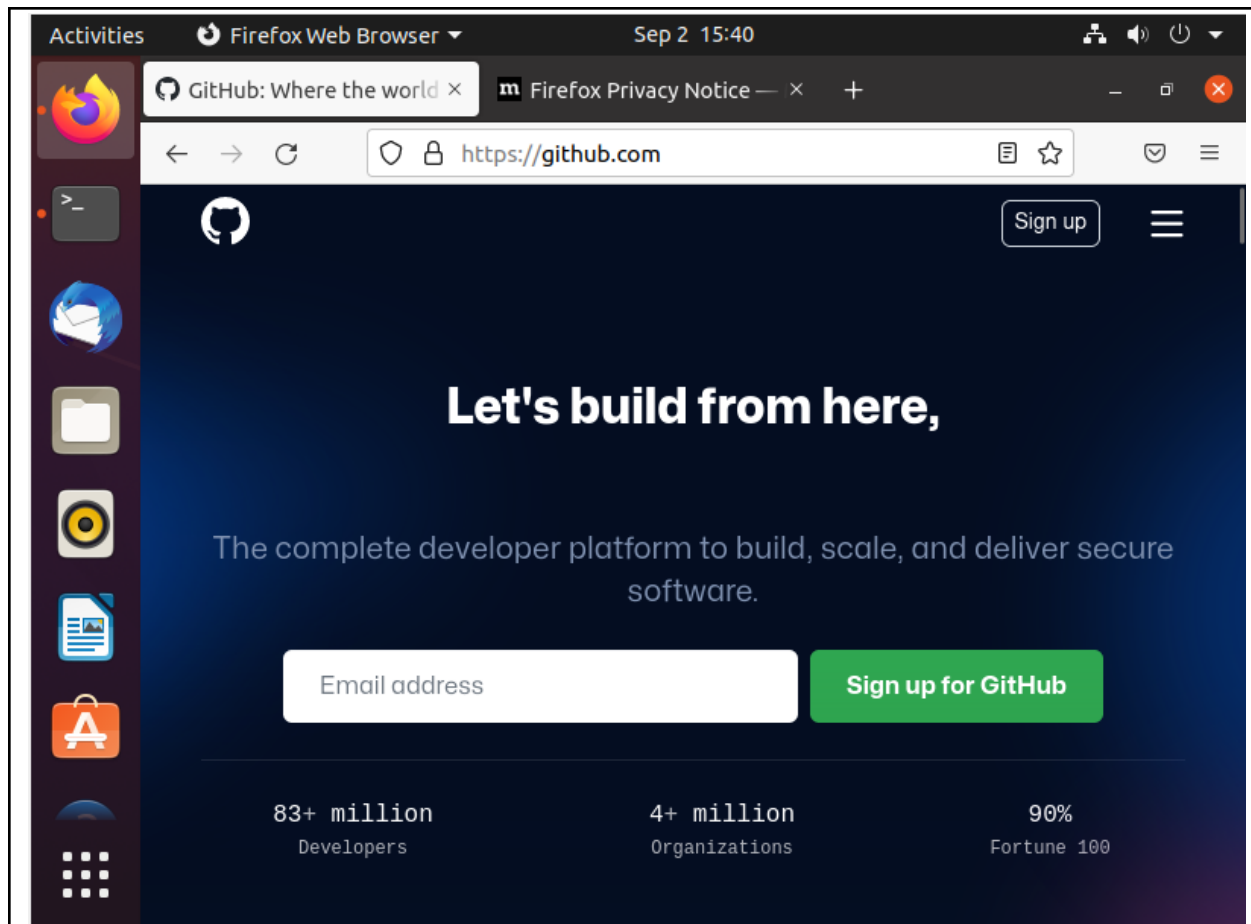
2. After the installation, issue the command *which git* again. The directory of git is usually installed in this location: *user/bin/git*.

```
davonn@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.

```
davonn@workstation:~$ git --version  
git version 2.25.1
```

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.


Activities Firefox Web Browser Sep 2 15:47

Create a New Repository × Firefox Privacy Notice ×

https://github.com/new

[Import a repository.](#)

Owner * **Repository name ***

 DavonnEscobilla / CPE232_Davonn ✓

Great repository names are short and memorable. Need inspiration? How about [glowing-garbanzo?](#)

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:
Skip this step if you're importing an existing repository.

☒ **Add a README file**
This is where you can write a long description for your project. [Learn more.](#)

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.

SSH and GPG keys

Firefox Privacy Notice

https://github.com/settings/keys

<> Developer settings

SSH keys

New SSH key

There are no SSH keys associated with your account.

Check out our guide to [generating SSH keys](#) or troubleshoot [common SSH problems](#).

GPG keys

New GPG key

There are no GPG keys associated with your account.

Learn how to [generate a GPG key and add it to your account](#).

Vigilant mode

☐ Flag unsigned commits as unverified

This will include any commit attributed to your account but not signed with your GPG or S/MIME key.

Note that this will include your existing unsigned commits.

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.

```
davonn@workstation:~$ cat .ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQCAQDe069u8a2Ph+lisP6eQLRrfJ+KONLet6z/FjlQ8QT
lW5HUH0p0/yctWbiCDxJmNFjcKaVs0b2ZLEchJUo00U02F64lGd63WRIFNwF9+Cf/bGuJStZBp8sQkZ
x8Y6Z3k/+d2tPFF0MDlk8Qd8UX2cUtm7J4VD1VBh3Xtd059nDHF07XzPfsCywzFk1MMkqad0PkUoaxy
cuL/nyGi8qMqjU2/JhbSC5guVzS3E1tqQBKTbfbhggAke3m7L44QAhGFoZeAuI7iOp6SPqNQzmNhKg
fdvLRw4++enLguU+yH1sJtOY71WW451J1tPn8c1BHK2u/FBzWnXkIwQAeyxcqeGLappdKuNoBGTqcNQ
kixAVCwLOGrzE3rlVd6dMzuAH2ZxwhtwvLIxcS8BSgQBy9C7ELnYR1Qm5IZpSr4o18pYz+srW9W98yD
HiZeWJ9dPcSMN2B7RPzSQtkTGkNVpt7utmpUHm8VdGcdbNcnrsLLxCcB99ETr13z0STvxCLZ0FFtKgV
kuC8p0Uw0cgsjpV9v80PKzfSHJ/8yZWmyWPQy5+gmoCjvWbjccWWceRNrBTeKB5jPh/cn/41PEMJJWk
+na/fNtJfmQ67fKqBtDvmNxE/A5o5/7uHr+uWXBk0YAaqsx7aRdNXnzMPe+qo2xHrK0avPYxai05nFF
b1CvFRcfbGQ== davonn@workstation
```

Add new SSH keys

https://github.com/settings/ssh/new

SSH keys / Add new

Title

CPE232

Key type

Authentication Key

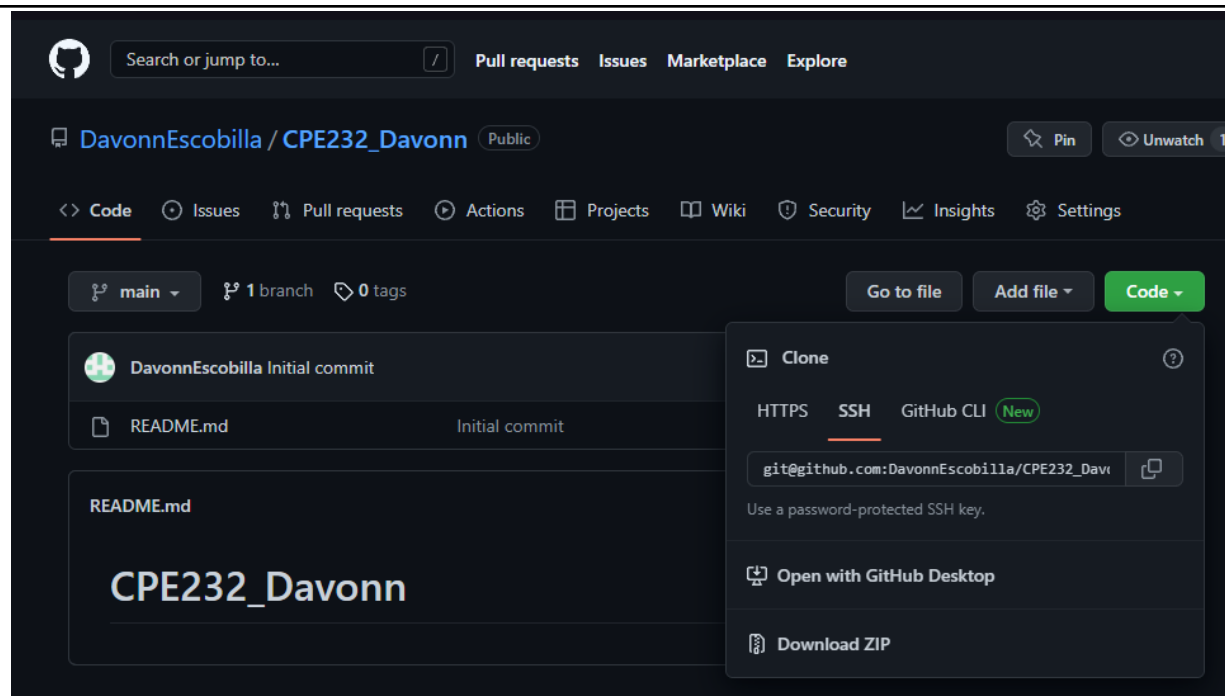
Key

```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAQDeO69u8a2Ph+lisP6eQLRrfJ+KONLet6z/FjlQ8QTIW5H
UH0pO/ycTWbiCDxJmNFjcKaVs0b2ZIEchJUoO0U02F64IGd63WRIFNwF9+Cf
/bGujStZBp8sQkZx8Y6Z3k
/+d2tPFF0MDIk8Qd8UX2cUtm7J4VD1VBh3XtdO59nDHFO7XzPfsCywzFk1MMkqad0PkUoaxcul
/nyGi8qMqjU2
/JhbSC5guVzS3E1tqQBKTbfbhggAke3m7L44QAhGFoZeAul7iOp6SPqNQzmNhHkgfdvLRw4++enL
guU+yH1sJtOY71WW451J1tPn8c1BHK2u
/FBzWnXklwQAeyxcqeGLappdKuNoBGTqcNQkixAVCwIOGrzE3rIVd6dMzuAH2ZxwhtwvllXcS8BSg
```

Add SSH key

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.

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:jvtaylor-cpe/CPE232_yourname.git`. When prompted to continue connecting, type yes and press enter.

```
davonn@workstation:~$ git clone git@github.com:DavonnEscobilla/CPE232_Davonn.git
Cloning into 'CPE232_Davonn'...
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/[fingerprint])? yes
Warning: Permanently added 'github.com,20.205.243.166' (ECDSA) 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`.

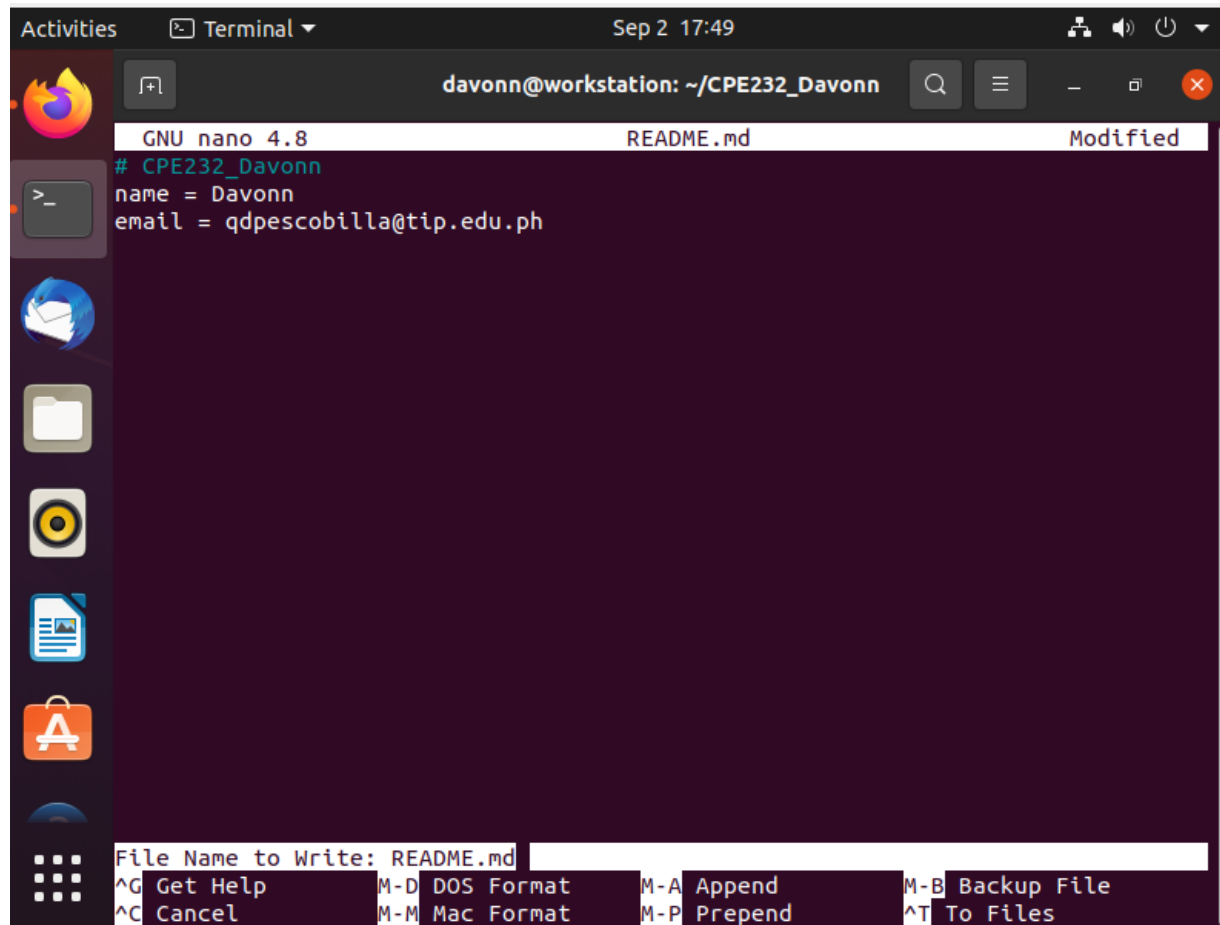
```
davonn@workstation:~$ ls
CPE232_Davonn  Documents  Music      Pictures  Templates
Desktop        Downloads  nano.save  Public    Videos
```

- 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`

```
davonn@workstation:~$ git config --global user.name "Davonn"
davonn@workstation:~$ git config --global user.email qdpescobilla@tip.edu.ph
davonn@workstation:~$ cat ~/.gitconfig
[user]
    name = Davonn
    email = qdpescobilla@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?

```
davonn@workstation:~/CPE232_Davonn$ 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.

```
davonn@workstation:~/CPE232_Davonn$ git add README.md
```

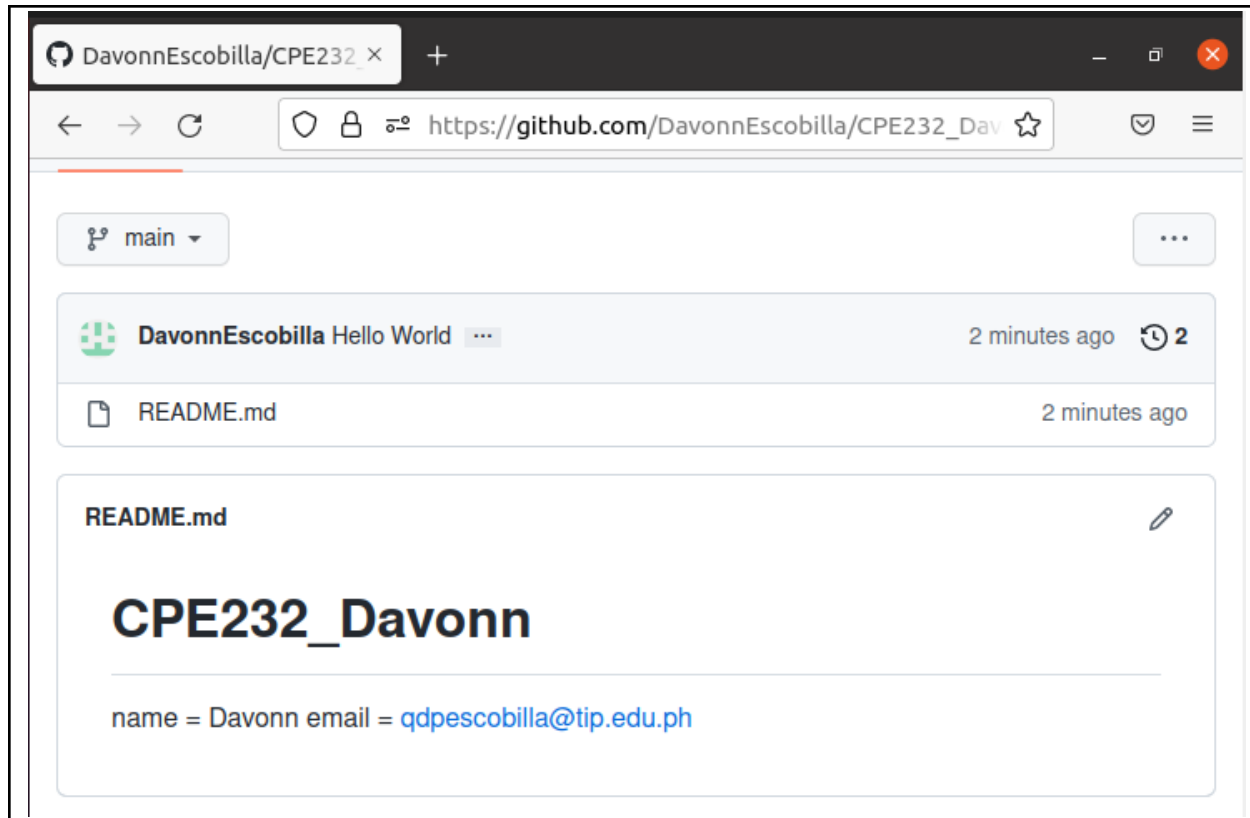
- k. Use the `git commit -m "your message"` to create a snapshot of the staged changes along the timeline of the Git projects history. The use of this command is required to select the changes that will be staged for the next commit.

```
davonn@workstation:~/CPE232_Davonn$ git commit -m "Hello World"
[main 715c0b1] Hello World
1 file changed, 4 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`.

```
davonn@workstation:~/CPE232_Davonn$ git push origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 303 bytes | 303.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To github.com:DavonnEscobilla/CPE232_Davonn.git
fbb79aa..715c0b1  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 have performed the ssh commands that manipulated the public key in order to install it on the remote servers, in this way there is a proper authentication process between the local machine and the servers. These ansible commands are composed of locating the directories, installing, and creating keys.

4. How important is the inventory file?

It does create the development of organizing the file locations, size, last time edit, and etc. It can serve as a guide to identify problems on managing the data and resources.

Conclusions/Learnings:

Securing the servers using the private and public key is another essential way to protect the data from compromising techniques. Since it is more secure in automated login and authentication hosts, ssh-program is much more optimal to implement and install. Using the ansible commands, one must understand the functionality of these things in order to execute the right process and avoid small mistakes.

**I affirm that I shall not give or receive any unauthorized help on this assignment
and that all work shall be my own. - Davonn P. Escobilla**