

Name: Torrecampo, Juan Piolo S.	Date Performed: Aug 23, 2022
Course/Section: CPE 232 / CPE31S22	Date Submitted: Aug 23, 2022
Instructor: Dr. Jonathan Taylor	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*.

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.

```
piolo@workstation:~/Desktop$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/piolo/.ssh/id_rsa):
```

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/piolo/.ssh/id_rsa
Your public key has been saved in /home/piolo/.ssh/id_rsa.pub
Your fingerprint is:
SHA256:Bv8293CcYEaqmmri9ZAwk6moSCLJG5z0SKP1rMR8slQ piolo@workstation
The key's randomart image is:
+---[RSA 4096]---+
|
|      .
|     o o o
|  +B E  S . +
|B*=B . . o o o .
|BB*.B . + o +
|*o+*.o o . o +
|oo+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*.

```
piolo@workstation:~$ ls -la .ssh
total 24
drwx----- 2 piolo piolo 4096 Aug 23 08:34 .
drwxr-x--- 15 piolo piolo 4096 Aug 16 12:07 ..
-rw----- 1 piolo piolo 3381 Aug 23 08:34 id_rsa
-rw-r--r-- 1 piolo piolo 743 Aug 23 08:34 id_rsa.pub
-rw----- 1 piolo piolo 2240 Aug 16 12:14 known_hosts
-rw----- 1 piolo piolo 1120 Aug 16 12:10 known_hosts.old
```

Task 1 Full Screenshot

```
Enter same passphrase again:
Your identification has been saved in /home/piolo/.ssh/id_rsa
Your public key has been saved in /home/piolo/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:Bv8293CcYEaqmmri9ZAwk6moSCLJG5z0SKP1rMR8slQ piolo@workstation
The key's randomart image is:
+--[RSA 4096]-----+
|
|      .
|    o   o   o
| +B E   S . +
|B*=B . . o o o .
|BB*.B   . + o +
|*o+*.o o . o +
|oo+o..+      .
+-----[SHA256]-----+
piolo@workstation:~/Desktop$ ls -la .ssh
ls: cannot access '.ssh': No such file or directory
piolo@workstation:~/Desktop$ cd
piolo@workstation:~$ ls -la .ssh
total 24
drwx----- 2 piolo piolo 4096 Aug 23 08:34 .
drwxr-x--- 15 piolo piolo 4096 Aug 16 12:07 ..
-rw----- 1 piolo piolo 3381 Aug 23 08:34 id_rsa
-rw-r--r-- 1 piolo piolo 743 Aug 23 08:34 id_rsa.pub
-rw----- 1 piolo piolo 2240 Aug 16 12:14 known_hosts
-rw----- 1 piolo piolo 1120 Aug 16 12:10 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.
2. Issue the command similar to this: *ssh-copy-id -i ~/.ssh/id_rsa user@host*

Local Host to Server 1

```
piolo@workstation:~$ ssh-copy-id piolo@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 promp
ted now it is to install the new keys
piolo@server1's password:

Number of key(s) added: 1

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

Local Host to Server 2

```
piolo@workstation:~$ ssh-copy-id piolo@server2
/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
piolo@server2's password:
Permission denied, please try again.
piolo@server2's password:

Number of key(s) added: 1

Now try logging into the machine, with:  "ssh 'piolo@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?
 - **Both ssh connections for server1 and server2 from the local host did not ask for the password and had a successful connection.**

Local Host to Server 1

```
piolo@workstation:~$ ssh piolo@server1
Welcome to Ubuntu 22.04.1 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.

Last login: Tue Aug 16 12:14:06 2022 from 192.168.56.102
piolo@server1:~$ exit
logout
Connection to server1 closed.
```

Local Host to Server 2

```

piolo@workstation:~$ ssh piolo@server2
Welcome to Ubuntu 22.04.1 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.

Last login: Tue Aug 16 12:14:45 2022 from 192.168.56.102
piolo@server2:~$ exit
logout
Connection to server2 closed.

```

Reflections:

Answer the following:

1. How will you describe the ssh-program? What does it do?
 - The ssh program is a tool for connecting to a remote pc. It provides two methods of connection for the host and server. The first SSH program provides logging in using password but there are exploits that can penetrate to the system whenever we use this method. The second command that is mostly used because of its security is connection using SSH key. These two methods are built into this ssh program which we can say that SSH connection is more secure than we thought.
 -
2. How do you know that you already installed the public key to the remote servers?
 - We can verify if the public SSH is installed to the system, by examining if the "authorized_keys" file exists under the "~/.ssh" directory in remote servers.

Verification of Public Key in Server 1

```

piolo@server1:~/.ssh$ ll
total 20
drwx----- 2 piolo piolo 4096 Aug 23 08:41 ./
drwxr-x--- 15 piolo piolo 4096 Aug 16 12:09 ../
-rw----- 1 piolo piolo 743 Aug 23 08:41 authorized_keys

```

Verification of Public Key in Server 2

```

piolo@server2:~/.ssh$ ll
total 12
drwx----- 2 piolo piolo 4096 Aug 23 08:43 ./
drwxr-x--- 16 piolo piolo 4096 Aug 23 08:29 ../
-rw----- 1 piolo piolo 743 Aug 23 08:43 authorized_keys
piolo@server2:~/.ssh$ █

```

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*

```
piolo@workstation:~$ sudo apt install git
[sudo] password for piolo:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  git-man liberror-perl
Suggested packages:
  git-daemon-run | git-daemon-sysvinit git-doc 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 3 not upgraded.
Need to get 4,110 kB of archives.
After this operation, 20.9 MB of additional disk space will be used.
Do you want to continue? [Y/n]
Get:1 http://ph.archive.ubuntu.com/ubuntu jammy/main amd64 liberror-perl all 0.17029-1 [26.5 kB]
Get:2 http://ph.archive.ubuntu.com/ubuntu jammy-updates/main amd64 git-man all 1:2.34.1-1ubuntu1.4 [952 kB]
Get:3 http://ph.archive.ubuntu.com/ubuntu jammy-updates/main amd64 git amd64 1:2.34.1-1ubuntu1.4 [3,131 kB]
Fetched 4,110 kB in 5s (846 kB/s)
Selecting previously unselected package liberror-perl.
(Reading database ... 201044 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%3a2.34.1-1ubuntu1.4_all.deb ...
Unpacking git-man (1:2.34.1-1ubuntu1.4) ...
Selecting previously unselected package git.
Preparing to unpack .../git_1%3a2.34.1-1ubuntu1.4_amd64.deb ...
Unpacking git (1:2.34.1-1ubuntu1.4) ...
Setting up liberror-perl (0.17029-1) ...
Setting up git-man (1:2.34.1-1ubuntu1.4) ...
Setting up git (1:2.34.1-1ubuntu1.4) ...
Processing triggers for man-db (2.10.2-1) ...
```

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

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

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

```
piolo@workstation:~$ git --version
git version 2.34.1
piolo@workstation:~$
```

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

4. Sign up in case you don't have an account yet. Otherwise, login to your GitHub account.

Welcome to GitHub!
Let's begin the adventure

Enter your email

✓ qjpstorrecampo@tip.edu.ph

Create a password

✓

Enter a username

✓ piolotorrecampo

Would you like to receive product updates and
announcements via email?

Type "y" for yes or "n" for no

→ n

Continue

You're almost done!
We sent a launch code to `qjpstorrecampo@tip.edu.ph`


→ Enter code

- a. Create a new repository and name it as CPE232_yourname. Check Add a README file and click Create repository.

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner *

 piolotorrecampo ▾

Repository name *

/ CPE232_piolo ✓

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

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.](#)

Add .gitignore


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

.gitignore template: None ▾

Choose a license

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

License: None ▾

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

① You are creating a public repository in your personal account.

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.

SSH keys / Add new

Title

CPE232

Key

Begins with 'ssh-rsa', 'ecdsa-sha2-nistp256', 'ecdsa-sha2-nistp384', 'ecdsa-sha2-nistp521', 'ssh-ed25519', 'sk-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.

```
piolo@workstation:~$ cat .ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQCS7QayLLvIL3a42x8MUX6eDH1nwq5qCWQGpxLTGasm
kPxw60KUAQoonlKc5qANoZtRMED4cC88mH/E5o1a140mKL33YM9dnP/8mh2+B0u54ZquHICqe4uLi3Gw
9KGRnulaOAVladjErUztXIPr6F0JUXYbXXL3V6d5IjAhsbj0dWT6LRxLMjwxpU3KxzZaAhhdMnmBOIKT
u6T4a8K5Ir0gccQD+JRFxsVIMR+Kj8x7ACqV3X+xxQu6mesplF4BnL52ugmYM18OckstmQul83lOQm79
3AiSpVMd2o7c5M2aENyWSDvC69xdy2FmoyI2YdfwfiIJJm1mbH2o79hetzyDBYifhAdgdvUbmciXbz4
Qvv2SQ+9d0MaV/Q9QWzLO99vaMBWoakJiByzn+QDHxvcwEgFYtRQX9fWScdtYyl1ISbhf5hzNdTadmd
60Ixui0MwXHpzRj8kvNVlz6nQzBPVF2tw0Vt4HA3kUouLNPazLypZUnEuW1YsNPQixd1nJbvWJhmF6nE
ZCgbx6VT6WCDXKRisrTKqxfc8jjW2QtK/7UcpMtApbr6Yr0StS6RS9Vr//wkHECph5y4+Bln1D2d/ZWQ
nnvOp+XzdjJ6LvZHqA5asChgq04TF4JnfnmEqw/UCADZPwtdLQJEOWOUcCcjhswKpbMCKcb0cLkFtJJ
DQ== piolo@workstation
piolo@workstation:~$
```

After Adding the SSH Key

SSH keys

New SSH key

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



CPE232

SHA256:Bv8293CcYEaqmmr19ZAwk6moSCLJG5z0SKP1rMR8s1Q

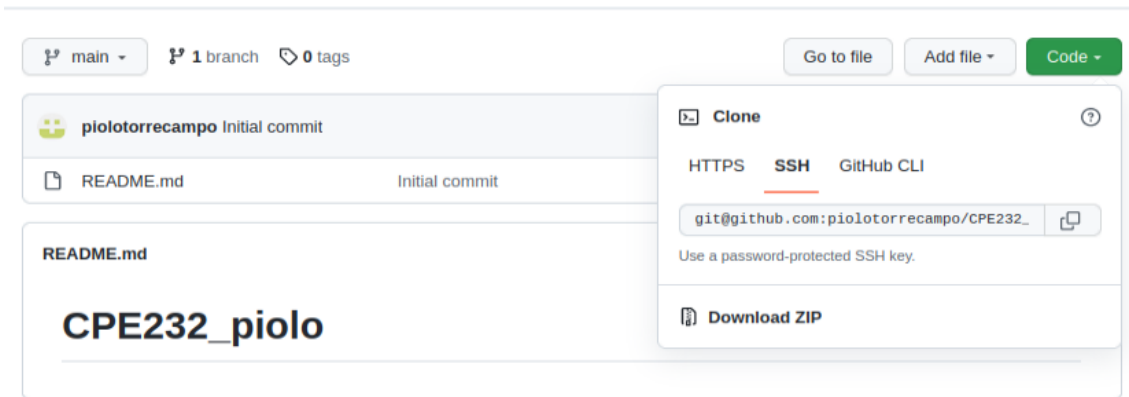
Added on Aug 23, 2022

Never used — Read/write

Delete

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

- 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.

```
piolo@workstation:~$ git clone git@github.com:piolotorrecampo/CPE232_piolo.git
Cloning into 'CPE232_piolo'...
The authenticity of host 'github.com (140.82.114.3)' can't be established.
ED25519 key fingerprint is SHA256:+DiY3wvV6TuJJhbpZisF/zLDA0zPMSvHdkr4UvC0qU.
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.
piolo@workstation:~$
```

- 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`.

```
piolo@workstation:~$ ls
CPE232_piolo  Documents  Music      Public     Templates
Desktop       Downloads  Pictures   snap       Videos
piolo@workstation:~$
```

```
piolo@workstation:~$ cd CPE232_piolo
piolo@workstation:~/CPE232_piolo$ ls
README.md
piolo@workstation:~/CPE232_piolo$
```

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`

```
piolo@workstation:~/CPE232_piolo$ git config --global user.name "piolotorrecampo"
piolo@workstation:~/CPE232_piolo$ git config --global user.email "qjpsstorrecampo@tip.edu.ph"
piolo@workstation:~/CPE232_piolo$ cat ~/.gitconfig
[user]
  name = piolotorrecampo
  email = qjpsstorrecampo@tip.edu.ph
piolo@workstation:~/CPE232_piolo$
```

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.

```
piolo@workstation:~/CPE232_piolo$ nano README.md
```

```
GNU nano 6.2                                README.md
# CPE232_piolo

# THIS IS A SAMPLE README FILE
### Name: Juan Piolo S. Torrecampo
### Age: 20
```

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?

- The command displays that the README.md has been modified.

```
piolo@workstation:~/CPE232_piolo$ 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")
piolo@workstation:~/CPE232_piolo$
```

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

```
piolo@workstation:~/CPE232_piolo$ git add README.md
piolo@workstation:~/CPE232_piolo$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   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.


```
piolo@workstation:~/CPE232_piolo$ git commit -m "First Commit"
[main 80c1419] First Commit
 1 file changed, 5 insertions(+), 1 deletion(-)
piolo@workstation:~/CPE232_piolo$
```


- 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*.



```
piolo@workstation:~/CPE232_piolo$ 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), 335 bytes | 335.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:piolotorrecampo/CPE232_piolo.git
   94a2d40..80c1419  main -> main
piolo@workstation:~/CPE232_piolo$
```

- 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.

 main  1 branch  0 tags Go to file Add file Code

 piolotorrecampo First Commit 80c1419 4 minutes ago 2 commits

 README.md First Commit 4 minutes ago

 README.md 

CPE232_piolo

THIS IS A SAMPLE README FILE

Name: Juan Piolo S. Torrecampo

Age: 20

Reflections:

Answer the following:

1. What sort of things have we so far done to the remote servers using ansible commands?
 - The first command that I have encountered in this activity is the "ssh-keygen" which enables us to create a key pair for our local machine and for remote machines. The second command is the "ssh-copy-id" which copies the public key of the local machine to the remote machine. Then after configuring the key pairs in both end devices, we can verify the SSH connection using the key pair by using SSH command in connection to remote servers.

2. How important is the inventory file?

- The inventory file is very important when doing big projects. It enables us to back track our progress and by this we can easily fix the bugs in the program.

Conclusions/Learnings:

To sum up, this activity aims to introduce and perform ssh authentication using key pairs. The first task is where I created a ssh key pair using the “ssh-keygen” command and specifying the bits and encryption method. The second task is where we are copying the public key to the remote server to allow remote access without entering the machine’s password. Lastly, the task three where I perform the creation of Github account and repository. Also, I perform the cloning and pushing in the cloud git repository. Overall, this activity gives me confidence in performing SSH commands and executing git commands in cloning and pushing to a remote repository.