

Name:Edgardo Kenneth D. Cordero	Date Performed: 8/23/2022
Course/Section:CPE 232 - CPE31S22	Date Submitted: 8/23/2022
Instructor: Dr. Jonathan Taylar	Semester and SY: 1ST SEMESTER
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*.

```
TIPQC@Q5202-01 MINGW64 ~
$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/TIPQC/.ssh/id_rsa):
/c/Users/TIPQC/.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 /c/Users/TIPQC/.ssh/id_rsa
Your public key has been saved in /c/Users/TIPQC/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:EjvSU5uzZ2N6AVxPUriFhDcb6Fws+FEhmr4mLm8V8bA TIPQC@Q5202-01
The key's randomart image is:
+---[RSA 3072]-----+
|  ..*+=. |
|    +o=.@ o |
|    +X.* X |
|    oE+Boo . |
|    . B.S. |
|    ..= o. |
|    ..o . =. |
|    ...o  =.. |
|    +o    .. |
+-----[SHA256]-----+
TIPQC@Q5202-01 MINGW64 ~
$ .....
```

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.

```
TIPQC@Q5202-01 MINGW64 ~
$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/TIPQC/.ssh/id_rsa):
/c/Users/TIPQC/.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 /c/Users/TIPQC/.ssh/id_rsa
Your public key has been saved in /c/Users/TIPQC/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:qGLPg2ahw4ILa+1Hw1WSI5v1s5gbfA/AaJzhJXagATs TIPQC@Q5202-01
The key's randomart image is:
+---[RSA 4096]-----+
|+..  .|
|.B B .|
|E % +|
|. @ + .|
|+ = o . S|
| B. = .|
|+.Xo+|
|*B+=..|
|** . o.|
+-----[SHA256]-----+
TIPQC@Q5202-01 MINGW64 ~
$
```

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.

```
TIPQC@Q5202-01 MINGW64 ~
$ ls -la .ssh
total 25
drwxr-xr-x 1 TIPQC 197121  0 Aug 23 08:01 ./
drwxr-xr-x 1 TIPQC 197121  0 Aug 23 08:20 ../
-rw-r--r-- 1 TIPQC 197121 3381 Aug 23 09:43 id_rsa
-rw-r--r-- 1 TIPQC 197121  740 Aug 23 09:43 id_rsa.pub
-rw-r--r-- 1 TIPQC 197121   96 Aug 18 11:53 known_hosts
TIPQC@Q5202-01 MINGW64 ~
$
```

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`
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?

Server1

```
TIPQC@Q5202-01 MINGW64 ~
$ ssh-copy-id -i ~/.ssh/id_rsa khien@192.168.56.102
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/c/Users/TIPQ
C/.ssh/id_rsa.pub"
The authenticity of host '192.168.56.102 (192.168.56.102)' can't be establis
hed.
ED25519 key fingerprint is SHA256:Q0wFH0nIPtRqCNFhtphT3XdTCv3FGR1s7a8PYfgPM2
8.
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 fil
ter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are pr
ompted now it is to install the new keys
khien@192.168.56.102's password:

Number of key(s) added: 1

Now try logging into the machine, with:  "ssh 'khien@192.168.56.102'"
and check to make sure that only the key(s) you wanted were added.

TIPQC@Q5202-01 MINGW64 ~
$ ssh khien@192.168.56.102
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

9 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

Last login: Tue Aug 23 09:12:28 2022 from 192.168.56.108
khien@server1:~$
```

Server2

```

TIPQC@Q5202-01 MINGW64 ~
$ ssh-copy-id -i ~/.ssh/id_rsa khien@192.168.56.101
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/c/Users/TIPQ
C/.ssh/id_rsa.pub"
The authenticity of host '192.168.56.101 (192.168.56.101)' can't be establis
hed.
ED25519 key fingerprint is SHA256:SKGMbTFNVtz1084DhuDbTA/3i75Qv6+TcrNjcyMFUi
c.
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 fil
ter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are pr
ompted now it is to install the new keys
khien@192.168.56.101's password:
Number of key(s) added: 1

Now try logging into the machine, with:  "ssh 'khien@192.168.56.101'"
and check to make sure that only the key(s) you wanted were added.

TIPQC@Q5202-01 MINGW64 ~
$ ssh khien@192.168.56.101
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

13 updates can be applied immediately.
4 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

Last login: Tue Aug 23 09:15:56 2022 from 192.168.56.108
khien@server52:~$

```

Observation:

What did you notice? Did the connection ask for a password? If not, why?

What I notice is that I am able to access the server by using the command `ssh-copy-id -i ~/.ssh/id_rsa user@host` in each server. The connection did not ask for a password because in the copy process it asked for the password already .

Reflections:

Answer the following:

1. How will you describe the ssh-program? What does it do?

I describe the ssh-program in a way of easy ticket because once you are able to have an access to a specific server you are able to control it and once you are able to enter the he password it once you installed the public key to remote the server it will be an easy ticket to have access and it will not ask for the password. The SSH is used in SFTP that can be used on securing the remote login from one

computer to another.

2. How do you know that you already installed the public key to the remote servers?

To be able to know if you are able to install the public key to the remote servers you need to type the command `ssh user@host` and it will show if you are able to have access to it.

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*

```
khien@workstation:~$ sudo apt install git
[sudo] password for khien:
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 18 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] y
Get:1 http://ph.archive.ubuntu.com/ubuntu jammy/main amd64 liberror-perl a
17029-1 [26.5 kB]
Get:2 http://ph.archive.ubuntu.com/ubuntu jammy-updates/main amd64 git-man
1:2.34.1-1ubuntu1.4 [952 kB]
Get:3 http://ph.archive.ubuntu.com/ubuntu jammy-updates/main amd64 git amd
2.34.1-1ubuntu1.4 [3,131 kB]
Fetched 4,110 kB in 25s (165 kB/s)
Selecting previously unselected package liberror-perl.
(Reading database ... 198501 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.
```

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

```
Processing triggers for man-db (2.
khien@workstation:~$ which git
/usr/bin/git
khien@workstation:~$
```

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

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


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.

Create a new repository

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

Owner *

 kennethcordero

Repository name *

CPE232_edgardokennethcordero ✓

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

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.



kennethcordero


Your personal account


[Go to your personal profile](#)

 Public profile


 Account

 Appearance


 Accessibility


 Notifications

Access


 Billing and plans

 Emails

 Password and authentication

 **SSH and GPG keys**

 Organizations

 Moderation

SSH keys / Add new

Title

CPE232 key

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.

SSH keys / Add new

Title

CPE232 Key

Key

ssh-rsa

```
AAAAB3NzaC1yc2EAAAADAQABAAQCuYTBpagxflMtMCS6j+TopEi8cBknVNzYsKpDv
5xZzUq7UkqsmZo6ZSgZ4xksd+Ew0umzlziaq/SSGaoHfXIEWuBGMLZzLhLOTbVfpZH6Q
4y/piysjirYUYeF3BEns7Usgad29AmJMyrnXftUU40jiK63Ntebtp1vw7AZd4a4607Olytz4a
nVdKjPhl4tNlCx7ZI5N7DPS67ep8BrY4T1D6X6HGnWe/CFIL1o1TRMWJtzO4IQfYvrNvs8L
hgcXir3zRKbX11ZAXknzouFpAQ21zkLxPIbl6MxGsLWHjQPjo2NsHUzNscAMPGXdtUIGw
UxmPlorXnpr+UyW4JfSrwHAyn6rXET2sVBkdmAvc0s5xBxCIDlhzwWzWOLWAnLRNrqsl
FjsvosgdKXiiLElZ892TWSLBmb60b16Lf2E+03DPM/fIOQ/4Xc1q9P6tXDd2sodyg2iQmiaZ
154jtYzBAA/Imoh64R3gLyvotx3P8QPjVg1CPg8wWVuQep7O5gYdS8/MekGbGqY3uXV
WE5ic5Sv6HxQIKWPaYbPa8AeQHAYMMs8sudea878iK6e8eYKH878OC844OKdce9M5C
```

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.

Search or jump to... Pulls Issues Marketplace Explore

jvtaylor-cpe / CPE302_yourname Public

Unwatch 1 Star 0 Fork 0

Code Issues Pull requests Actions Projects Wiki Security Insights

main

Go to file Add file Code

About

No description, website, or topics provided.

Readme

Releases

No releases published
Create a new release

Packages

Clone

HTTPS SSH GitHub CLI

git@github.com:jvtaylor-cpe/CPE302_you

Use a password-protected SSH key.

Download ZIP

README.md

CPE302_yourname

- 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.

```
TIPQC@Q5202-01 MINGW64 ~
$ git clone git@github.com:kennethcordero/CPE232_edgardokennethcordero.git
Cloning into 'CPE232_edgardokennethcordero'...
The authenticity of host 'github.com (140.82.112.4)' can't be established.
ED25519 key fingerprint is SHA256:+DiY3wvV6TuJJhbpZisF/zLDA0zPMSvHdkr4UvC
U.
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 hos
.
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.

TIPQC@Q5202-01 MINGW64 ~
$
```

- 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`.

```

TIPQC@Q5202-01 MINGW64 ~
$ ls
'3D Objects'/'
AppData/'
'Application Data'@
CPE232_edgardokennethcordero/
'Cisco Packet Tracer 7.2'/'
'Cisco Packet Tracer 7.3.0'/'
'Cisco Packet Tracer 8.1.1'/'
'Cisco Packet Tracer 8.2.0'/'
Contacts/
Cookies@
DNSCache/
Desktop/
Documents/
Downloads/
Favorites/
GNS3/
IntelGraphicsProfiles/
Links/
'Local Settings'@
MicrosoftEdgeBackups/
Music/
'My Documents'@
NTUSER.DAT
NTUSER.DAT{$3b39e88-18c4-11ea-a811-000d3aa4692b}.TM.b1f
NTUSER.DAT{$3b39e88-18c4-11ea-a811-000d3aa4692b}.TMCContainer0000000000000000
00001.regtrans-ms
NTUSER.DAT{$3b39e88-18c4-11ea-a811-000d3aa4692b}.TMCContainer0000000000000000
00002.regtrans-ms
NetHood@
OneDrive/
Pictures/
PrintHood@
Recent@
'Saved Games'/'
Searches/
SendTo@
'Start Menu'@
Templates@
Videos/
'VirtualBox VMs'/'
birador/
bluej/
gayla/
ntuser.dat.LOG1
ntuser.dat.LOG2
ntuser.ini

```

```

TIPQC@Q5202-01 MINGW64 ~
$ cd CPE232_edgardokennethcordero

TIPQC@Q5202-01 MINGW64 ~/CPE232_edgardokennethcordero (main)
$ ls
README.md

TIPQC@Q5202-01 MINGW64 ~/CPE232_edgardokennethcordero (main)
$

```

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`

```
TIPQC@Q5202-01 MINGW64 ~/CPE232_edgardokennethcordero (main)
$ git config --global user.name "Edgardo Kenneth Cordero"

TIPQC@Q5202-01 MINGW64 ~/CPE232_edgardokennethcordero (main)
$ git config --global user.email qekdccordero@tip.edu.ph

TIPQC@Q5202-01 MINGW64 ~/CPE232_edgardokennethcordero (main)
$ cat ~/.gitconfig
[user]
    name = Edgardo Kenneth Cordero
    email = qekdccordero@tip.edu.ph

TIPQC@Q5202-01 MINGW64 ~/CPE232_edgardokennethcordero (main)
$
```

- 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.

```
MINGW64:/c/Users/TIPQC/CPE232_edgardokennethcordero
GNU nano 6.4 README.md
# CPE232_edgardokennethcordero
It created 8/23/2022 11:13 AM
```

- 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?

```
TIPQC@Q5202-01 MINGW64 ~/CPE232_edgardokennethcordero (main)
$ 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")

TIPQC@Q5202-01 MINGW64 ~/CPE232_edgardokennethcordero (main)
$
```

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

```
TIPQC@Q5202-01 MINGW64 ~/CPE232_edgardokennethcordero (main)
$ git add README.md
warning: in the working copy of 'README.md', LF will be replaced by CRLF
next time Git touches it

TIPQC@Q5202-01 MINGW64 ~/CPE232_edgardokennethcordero (main)
$
```

- 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.

```
TIPQC@Q5202-01 MINGW64 ~/CPE232_edgardokennethcordero (main)
$ git commit -m "For three"
[main 275751c] For three
1 file changed, 2 insertions(+), 1 deletion(-)

TIPQC@Q5202-01 MINGW64 ~/CPE232_edgardokennethcordero (main)
$
```

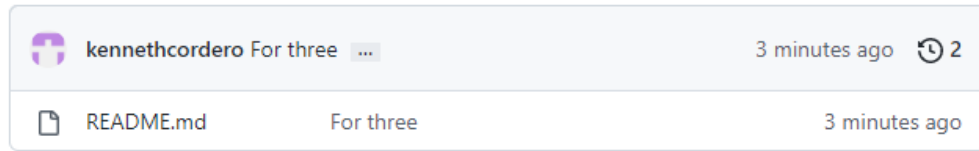
- l. Use the command `git push <remote><branch>` to upload the local repository content to GitHub repository. Pushing means to transfer commits from the local repository to the remote repository. As an example, you may issue `git push origin main`.

```
TIPQC@Q5202-01 MINGW64 ~/CPE232_edgardokennethcordero (main)
$ git push origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 315 bytes | 315.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:kennethcordero/CPE232_edgardokennethcordero.git
e9f4e4e..275751c main -> main

TIPQC@Q5202-01 MINGW64 ~/CPE232_edgardokennethcordero (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 are able to modify by using nano, send messages using the git to github and are able to connect the git to github and most importantly we are able to connect git to github. I learned how to configure remote and local machines.

4. How important is the inventory file?

The importance of the inventory file is to see what's added to the inventory and we can leave a note. We can get the link of the github in the inventory file. It can keep files on the github and use it as a host or groups so that it will lessen the hassle.

Conclusions/Learnings:

In conclusion I learned how to configure remote and local machines to connect via SSH using a key instead of using a password. I am able to create a public key and a private key and am able to verify the connectivity. I am able to set up the Git repository using local and remote repositories and configure and run ad hoc commands from local machine to remote servers. I discovered how to send a message from using git to github and being able to connect the git and github. I learned the importance of the inventory file of github. I learned how to set up git and the importance of ssh-keygen.