Name: Go, Meljune Royette G.	**Date Performed: 08/25/2022**
Course/Section: CPE – 31S23	**Date Submitted: 08/25/2022**
Instructor: Engr. Jonathan V. Taylar	**Semester and SY: 1st Semester – 2022 - 2023**

Activity 2: SSH Key-Based Authentication and Setting up Git

1. **Objectives:**

1.1 Configure remote and local machine to connect via SSH using a KEY instead of using a password
1.2 Create a public key and private key
1.3 Verify connectivity
1.4 Setup Git Repository using local and remote repositories
1.5 Configure and Run ad hoc commands from local machine to remote servers

Part 1: Discussion

It is assumed that you are already done with the last Activity (**Activity 1: Configure Network using Virtual Machines).** *Provide screenshots for each task*.

It is also assumed that you have VMs running that you can SSH but requires a password. Our goal is to remotely login through SSH using a key without using a password. In this activity, we create a public and a private key. The private key resides in the local machine while the public key will be pushed to remote machines. Thus, instead of using a password, the local machine can connect automatically using SSH through an authorized key.

What Is ssh-keygen?

Ssh-keygen is a tool for creating new authentication key pairs for SSH. Such key pairs are used for automating logins, single sign-on, and for authenticating hosts.

SSH Keys and Public Key Authentication

The SSH protocol uses public key cryptography for authenticating hosts and users. The authentication keys, called SSH keys, are created using the keygen program.

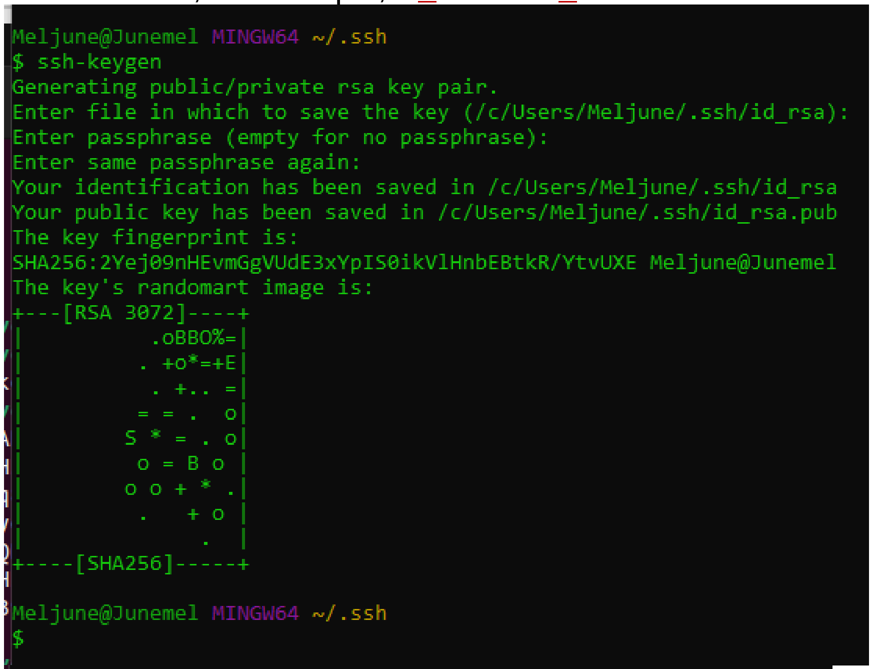
SSH introduced public key authentication as a more secure alternative to the older .rhosts authentication. It improved security by avoiding the need to have password stored in files and eliminated the possibility of a compromised server stealing the user's password.

However, SSH keys are authentication credentials just like passwords. Thus, they must be managed somewhat analogously to usernames and passwords. They should have a proper termination process so that keys are removed when no longer needed.

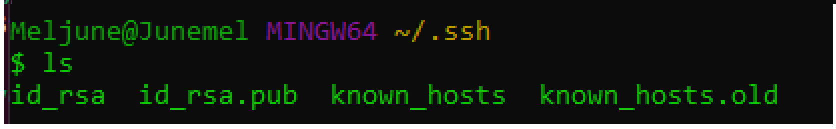
Task 1: Create an SSH Key Pair for User Authentication

1. The simplest way to generate a key pair is to run *ssh-keygen* without arguments. In this case, it will prompt for the file in which to store keys. First, the tool asked where to save the file. SSH keys for user authentication are usually stored in the users .ssh directory under the home directory. However, in enterprise environments, the location is often different. The default key file name depends

on the algorithm, in this case *id_rsa* when using the default RSA algorithm. It could also be, for example, *id_dsa* or *id_ecdsa*.

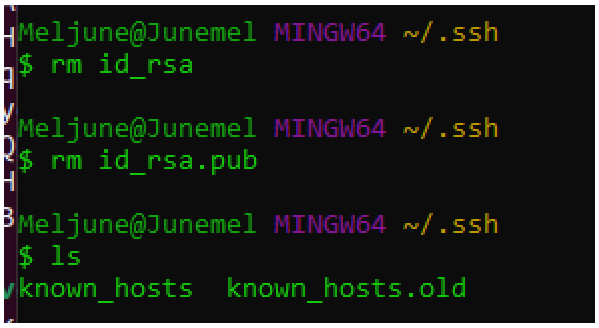


```
Meljune@Junemel MINGW64 ~/.ssh
$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/Meljune/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /c/Users/Meljune/.ssh/id_rsa
Your public key has been saved in /c/Users/Meljune/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:2Yej09nHEvmGgVUdE3xYpIS0ikVlHnbEBtkR/YtvUXE Meljune@Junemel
The key's randomart image is:
+---[RSA 3072]----+
|          .oBBO%=|
|         . +o*=+E|
|          . +.. =|
|         = = .  o|
|        S * = . o|
|         o = B o |
|        o o + * .|
|         .   + o |
|              .  |
+----[SHA256]-----+

Meljune@Junemel MINGW64 ~/.ssh
$
```



```
Meljune@Junemel MINGW64 ~/.ssh
$ ls
id_rsa  id_rsa.pub  known_hosts  known_hosts.old
```

By doing the command ssh-keygen, A key fingerprint was generated.

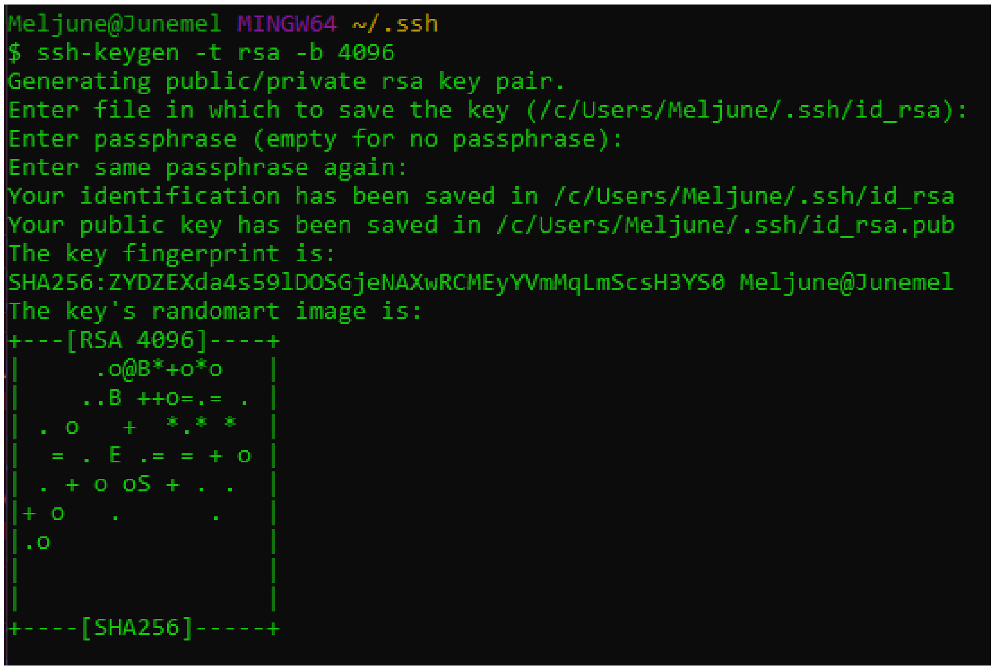


```
Meljune@Junemel MINGW64 ~/.ssh
$ rm id_rsa

Meljune@Junemel MINGW64 ~/.ssh
$ rm id_rsa.pub

Meljune@Junemel MINGW64 ~/.ssh
$ ls
known_hosts  known_hosts.old
```

Since we will create an ssh keygen using the -t and -b option we will delete these files.

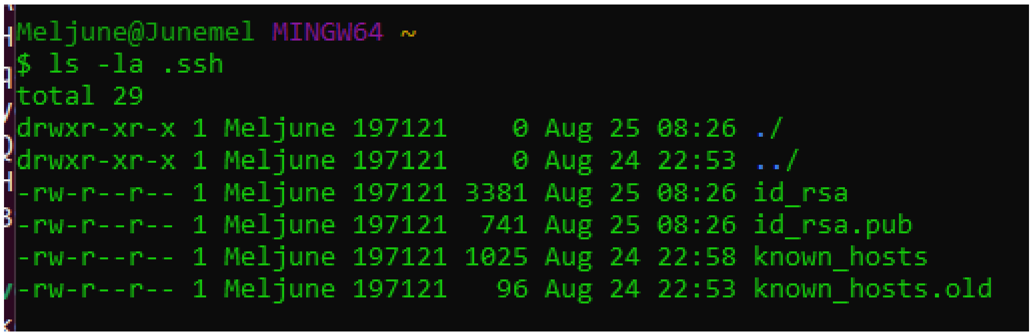
2. Issue the command *ssh-keygen -t rsa -b 4096.* The algorithm is selected using the -t option and key size using the -b option.

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.



```
Meljune@Junemel MINGW64 ~/.ssh
$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/Meljune/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /c/Users/Meljune/.ssh/id_rsa
Your public key has been saved in /c/Users/Meljune/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:ZYDZEXda4s59lDOSGjeNAXwRCMEyYVmMqLmScsH3YS0 Meljune@Junemel
The key's randomart image is:
+---[RSA 4096]----+
|      .o@B*+o*o  |
|     ..B ++o=.= .|
| . o   +  *.* *  |
|  = . E .= = + o |
| . + o oS + . .  |
|+ 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.



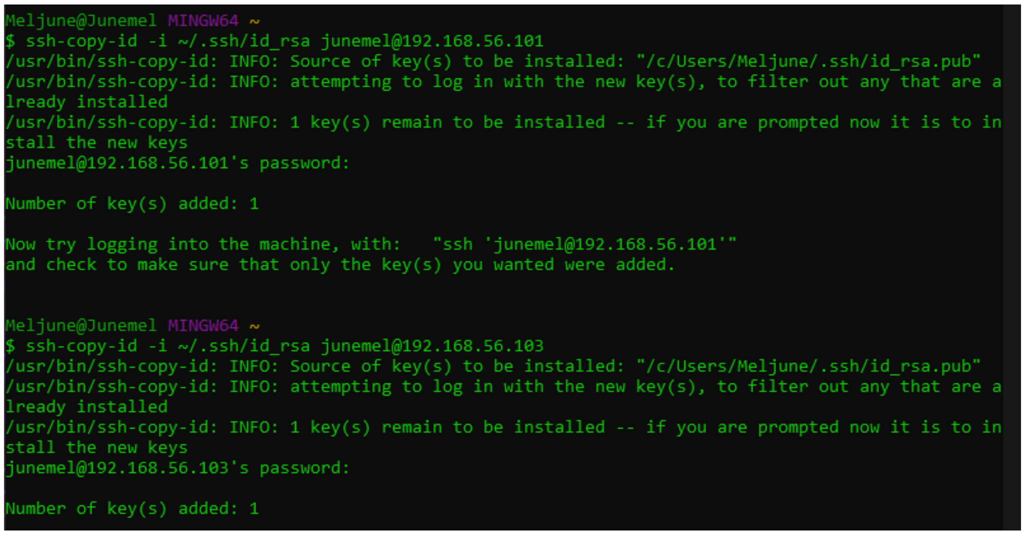
```
Meljune@Junemel MINGW64 ~
$ ls -la .ssh
total 29
drwxr-xr-x 1 Meljune 197121    0 Aug 25 08:26 ./
drwxr-xr-x 1 Meljune 197121    0 Aug 24 22:53 ../
-rw-r--r-- 1 Meljune 197121 3381 Aug 25 08:26 id_rsa
-rw-r--r-- 1 Meljune 197121  741 Aug 25 08:26 id_rsa.pub
-rw-r--r-- 1 Meljune 197121 1025 Aug 24 22:58 known_hosts
-rw-r--r-- 1 Meljune 197121   96 Aug 24 22:53 known_hosts.old
```

We have verified that a pair of keys is inside of the .ssh folder.

Task 2: Copying the Public Key to the remote servers

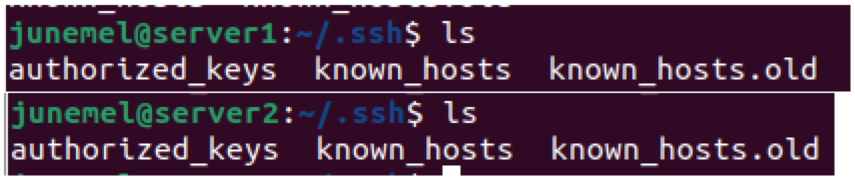
1. To use public key authentication, the public key must be copied to a server and installed in an *authorized_keys* file. This can be conveniently done using the *ssh-copy-id* tool.
2. Issue the command similar to this: *ssh-copy-id -i ~/.ssh/id_rsa user@host*

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.



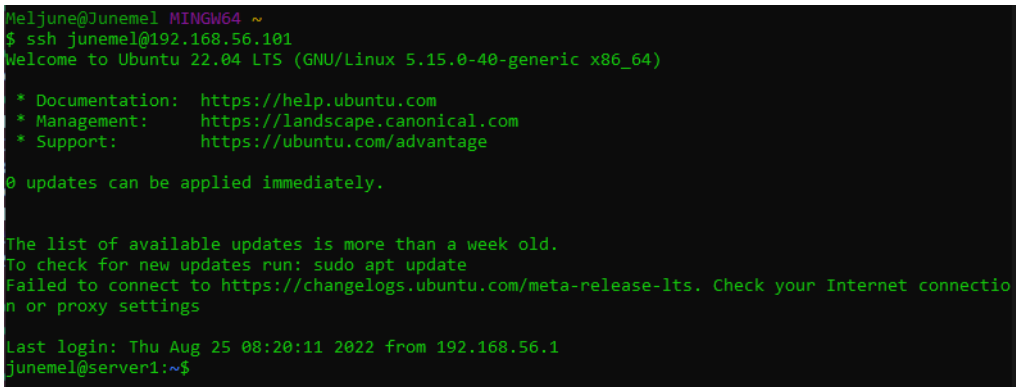
```
Meljune@Junemel MINGW64 ~
$ ssh-copy-id -i ~/.ssh/id_rsa junemel@192.168.56.101
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/c/Users/Meljune/.ssh/id_rsa.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are a
lready installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to in
stall the new keys
junemel@192.168.56.101's password:

Number of key(s) added: 1

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


Meljune@Junemel MINGW64 ~
$ ssh-copy-id -i ~/.ssh/id_rsa junemel@192.168.56.103
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/c/Users/Meljune/.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 a
lready installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to in
stall the new keys
junemel@192.168.56.103's password:

Number of key(s) added: 1
```

The public keys were copied to the .ssh folder of each servers.

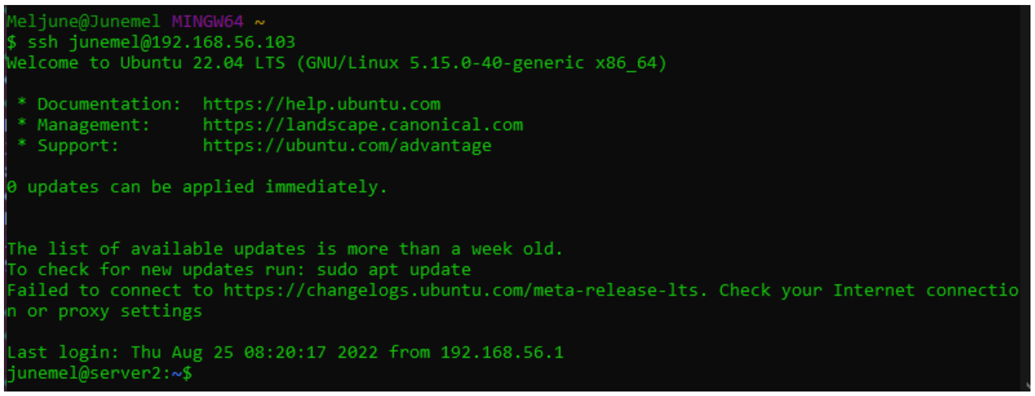


```
junemel@server1:~/.ssh$ ls
authorized_keys  known_hosts  known_hosts.old
junemel@server2:~/.ssh$ ls
authorized_keys  known_hosts  known_hosts.old
```

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?



```
Meljune@Junemel MINGW64 ~
$ ssh junemel@192.168.56.101
Welcome to Ubuntu 22.04 LTS (GNU/Linux 5.15.0-40-generic x86_64)

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

0 updates can be applied immediately.


The list of available updates is more than a week old.
To check for new updates run: sudo apt update
Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your Internet connectio
n or proxy settings

Last login: Thu Aug 25 08:20:11 2022 from 192.168.56.1
junemel@server1:~$
```



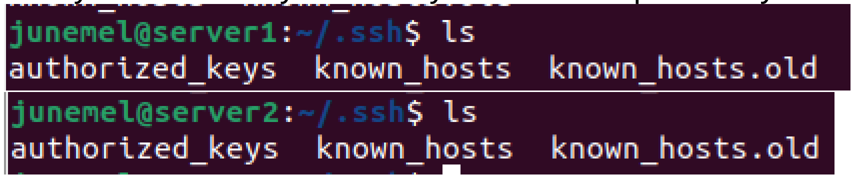
```
Meljune@Junemel MINGW64 ~
$ ssh junemel@192.168.56.103
Welcome to Ubuntu 22.04 LTS (GNU/Linux 5.15.0-40-generic x86_64)

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

0 updates can be applied immediately.


The list of available updates is more than a week old.
To check for new updates run: sudo apt update
Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your Internet connectio
n or proxy settings

Last login: Thu Aug 25 08:20:17 2022 from 192.168.56.1
junemel@server2:~$
```

Upon logging in to the ssh servers, there are no passwords that were asked because the public keys are inside of the servers.

Reflections:

Answer the following:

1. How will you describe the ssh-program? What does it do?
   The ssh-program uses a secure shell to connect to a remote system. Remote command line login and execution is the most common use of it.
2. How do you know that you already installed the public key to the remote servers?



```
junemel@server1:~/.ssh$ ls
authorized_keys  known_hosts  known_hosts.old
```

```
junemel@server2:~/.ssh$ ls
authorized_keys  known_hosts  known_hosts.old
```

   By checking the files inside the .ssh folder of each servers, you can verify if the public key is there when there is a "authorized_keys" file in there.

Part 2: Discussion

The screenshots were posted in the tasks.

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

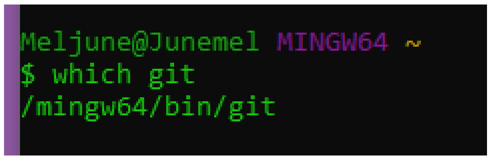
Set up Git

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

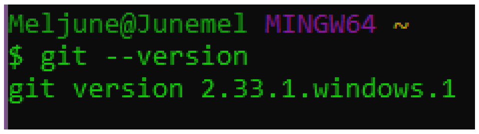
Task 3: Set up the Git Repository

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



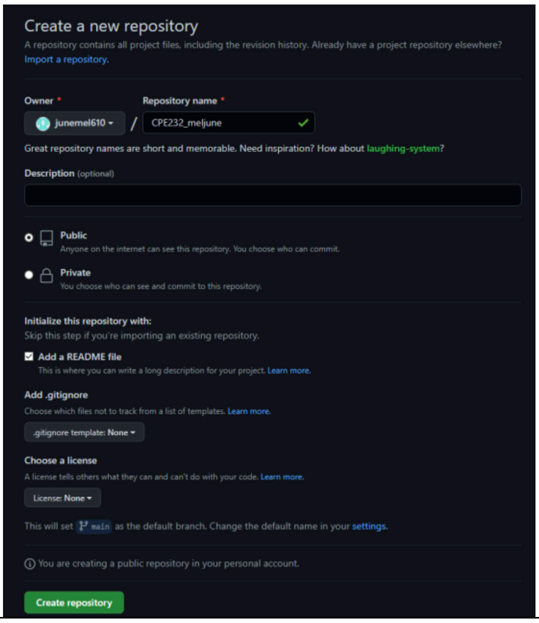
```
Meljune@Junemel MINGW64 ~
$ which git
/mingw64/bin/git
```

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



```
Meljune@Junemel MINGW64 ~
$ git --version
git version 2.33.1.windows.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.



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

Owner * junemel610 / Repository name * CPE232_meljune

Great repository names are short and memorable. Need inspiration? How about laughing-system?

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

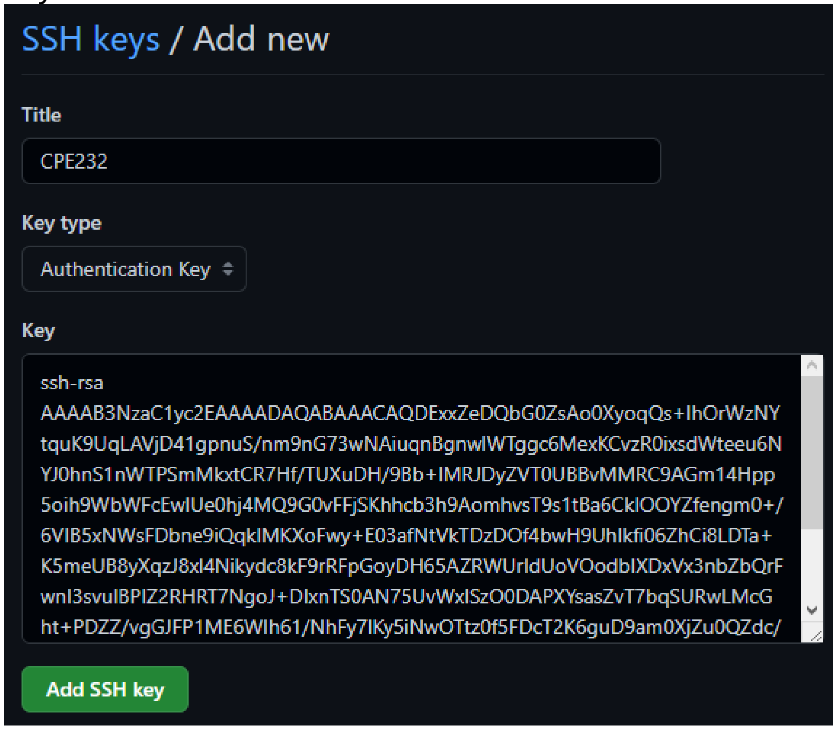
Choose a license
A license tells others what they can and can't do with your code. Learn more.
License: None

This will set main as the default branch. Change the default name in your settings.

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

b. Create a new SSH key on GitHub. Go your profile's setting and click SSH and GPG keys. If there is an existing key, make sure to delete it. To create a new SSH keys, click New SSH Key. Write CPE232 key as the title of the key.

c. On the local machine's terminal, issue the command cat .ssh/id_rsa.pub and copy the public key. Paste it on the GitHub key and press Add SSH key.



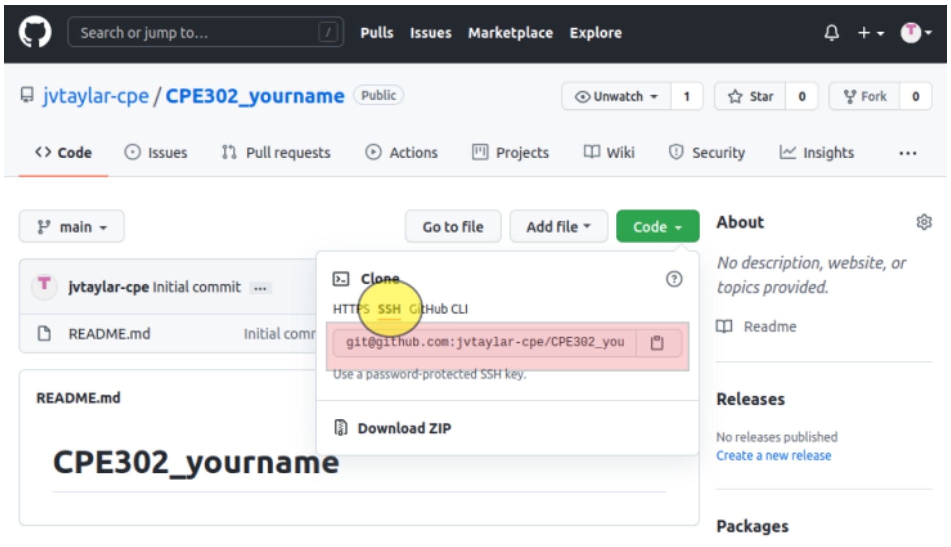
Title: CPE232

Key type: Authentication Key

Key:

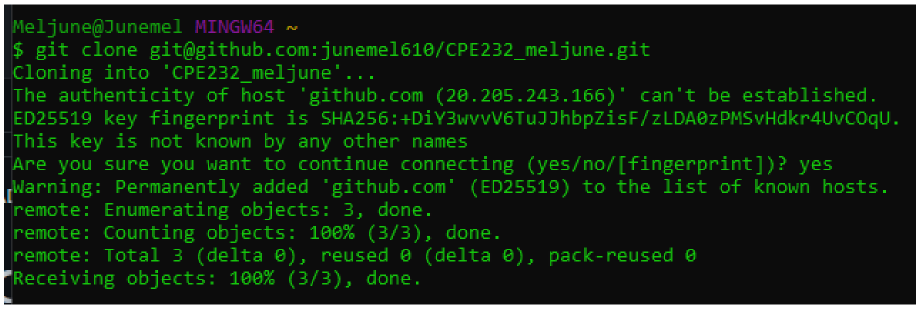
```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAACAQDExxZeDQbG0ZsAo0XyoqQs+IhOrWzNY
tquK9UqLAVjD41gpnuS/nm9nG73wNAiuqnBgnwlWTggc6MexKCvzR0ixsdWteeu6N
YJ0hnS1nWTPSmMkxtCR7Hf/TUXuDH/9Bb+IMRJDyZVT0UBBvMMRC9AGm14Hpp
5oih9WbWFcEwlUe0hj4MQ9G0vFFjSKhhcb3h9AomhvsT9s1tBa6CklOOYZfengm0+/
6VIB5xNWsFDbne9iQqkIMKXoFwy+E03afNtVkTDzDOf4bwH9Uhlkfi06ZhCi8LDTa+
K5meUB8yXqzJ8xl4Nikydc8kF9rRFpGoyDH65AZRWUrldUoVOodblXDxVx3nbZbQrF
wnl3svulBPIZ2RHRT7NgoJ+DlxnTS0AN75UvWxlSzO0DAPXYsasZvT7bqSURwLMcG
ht+PDZZ/vgGJFP1ME6WIh61/NhFy7IKy5iNwOTtz0f5FDcT2K6guD9am0XjZu0QZdc/
```

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.



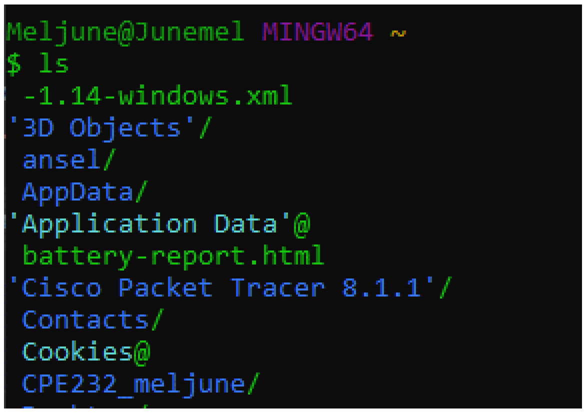
```
git@github.com:jvtaylar-cpe/CPE302_you
```

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

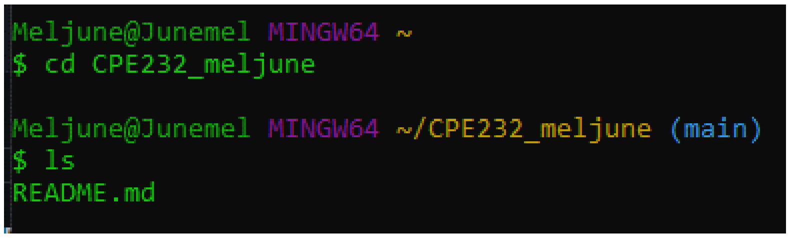


```
Meljune@Junemel MINGW64 ~
$ git clone git@github.com:junemel610/CPE232_meljune.git
Cloning into 'CPE232_meljune'...
The authenticity of host 'github.com (20.205.243.166)' can't be established.
ED25519 key fingerprint is SHA256:+DiY3wvvV6TuJJhbpZisF/zLDA0zPMSvHdkr4UvCOqU.
This key is not known by 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.



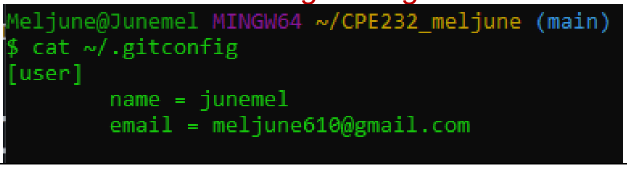
```
Meljune@Junemel MINGW64 ~
$ ls
 -1.14-windows.xml
'3D Objects'/
 ansel/
 AppData/
'Application Data'@
 battery-report.html
'Cisco Packet Tracer 8.1.1'/
 Contacts/
 Cookies@
 CPE232_meljune/
```



```
Meljune@Junemel MINGW64 ~
$ cd CPE232_meljune

Meljune@Junemel MINGW64 ~/CPE232_meljune (main)
$ ls
README.md
```

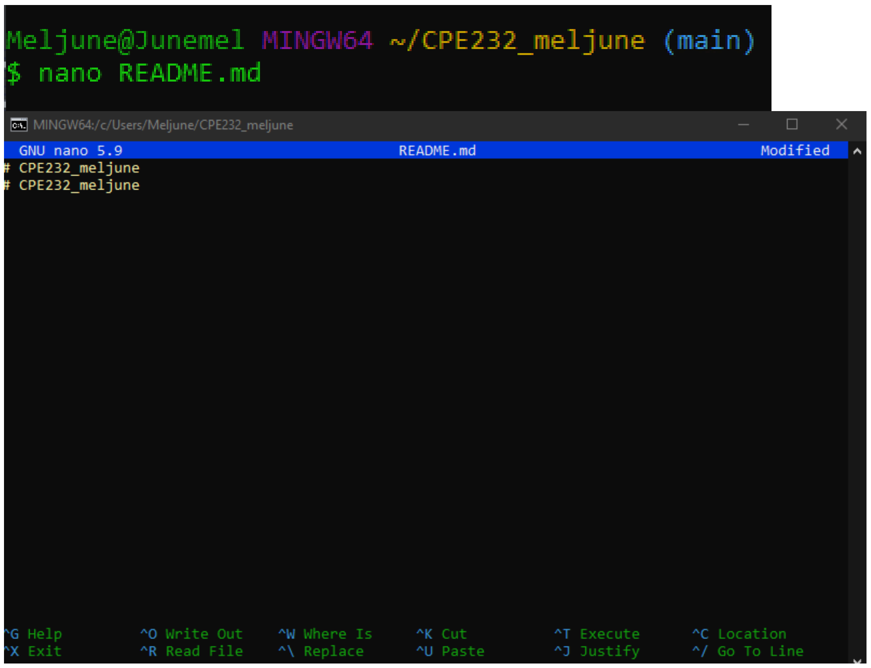
g. Use the following commands to personalize your git.

- *git config --global user.name “Your Name”*
- *git config --global user.email yourname@email.com*
- Verify that you have personalized the config file using the command *cat ~/.gitconfig*



```
Meljune@Junemel MINGW64 ~/CPE232_meljune (main)
$ cat ~/.gitconfig
[user]
        name = junemel
        email = meljune610@gmail.com
```

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.

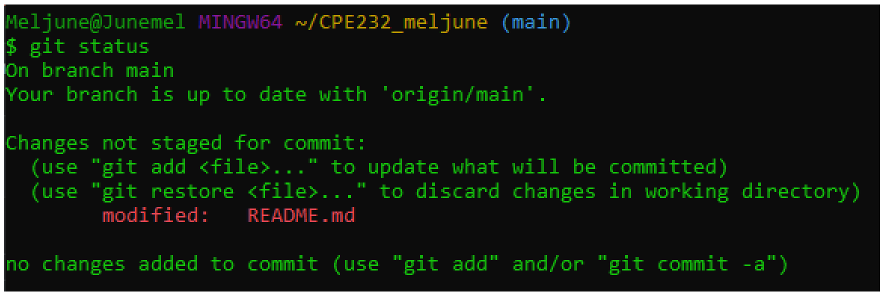


```
Meljune@Junemel MINGW64 ~/CPE232_meljune (main)
$ nano README.md
```

MINGW64:/c/Users/Meljune/CPE232_meljune

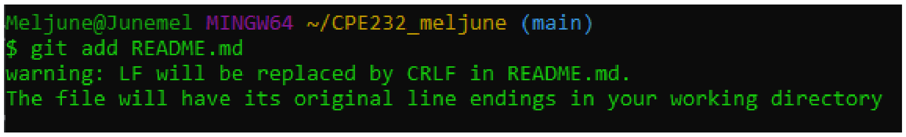
```
GNU nano 5.9                                  README.md                                  Modified
# CPE232_meljune
# CPE232_meljune

^G Help         ^O Write Out   ^W Where Is    ^K Cut          ^T Execute     ^C Location
^X Exit         ^R Read File   ^\ Replace     ^U Paste        ^J Justify     ^/ Go To Line
```

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?



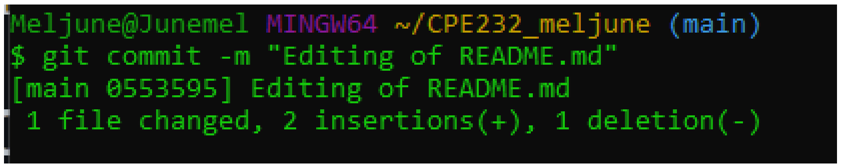
```
Meljune@Junemel MINGW64 ~/CPE232_meljune (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

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")
```

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



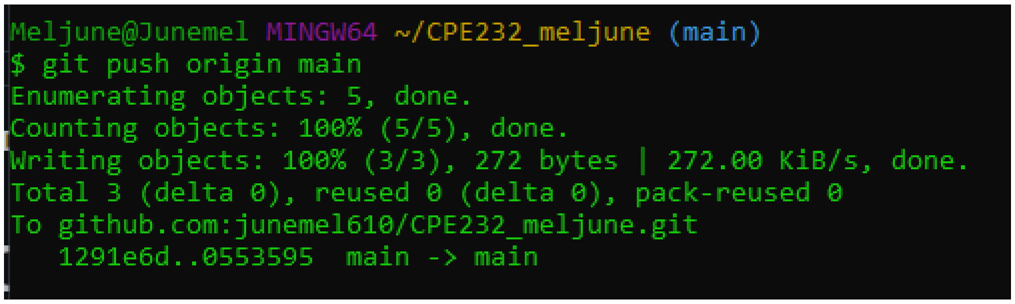
```
Meljune@Junemel MINGW64 ~/CPE232_meljune (main)
$ git add README.md
warning: LF will be replaced by CRLF in README.md.
The file will have its original line endings in your working directory
```

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.



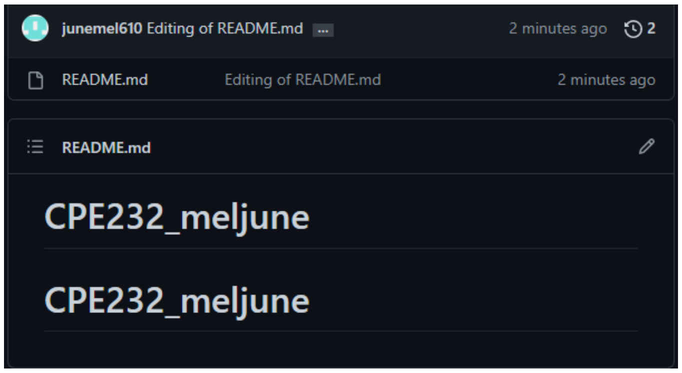
```
Meljune@Junemel MINGW64 ~/CPE232_meljune (main)
$ git commit -m "Editing of README.md"
[main 0553595] Editing of README.md
 1 file changed, 2 insertions(+), 1 deletion(-)
```

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.*



```
Meljune@Junemel MINGW64 ~/CPE232_meljune (main)
$ git push origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Writing objects: 100% (3/3), 272 bytes | 272.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:junemel610/CPE232_meljune.git
   1291e6d..0553595  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.



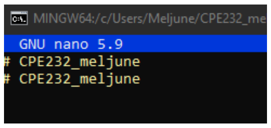
junemel610 Editing of README.md ... 2 minutes ago 2

README.md Editing of README.md 2 minutes ago

README.md

CPE232_meljune

CPE232_meljune



```
MINGW64:/c/Users/Meljune/CPE232_me
 GNU nano 5.9
# CPE232_meljune
# CPE232_meljune
```

The README.md was edited successfully.

Reflections:

Answer the following:

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

   So far, we have set up the 2 ssh servers. In those 2 servers a password is no longer asked when logging in because there is a Key in the .ssh folder of those servers.

4. How important is the inventory file?

   Inventory file is very important because it can serve as a backup of your files

Conclusions/Learnings:

In this activity, I've learned how to set up servers and log in to them without typing a password using a KEY. I've also learned on how to sync my github in my local computer. Mastering these tasks is very important because these will be the foundation of managing an enterprise server.