Name: Nowell Gabriel C. Quizon	**Date Performed:** 08/28/2023
Course/Section: CPE31S5	**Date Submitted:** 08/28/2023
Instructor: Engr. Roman Richard	**Semester and SY:** 1st and 2023-2024

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.

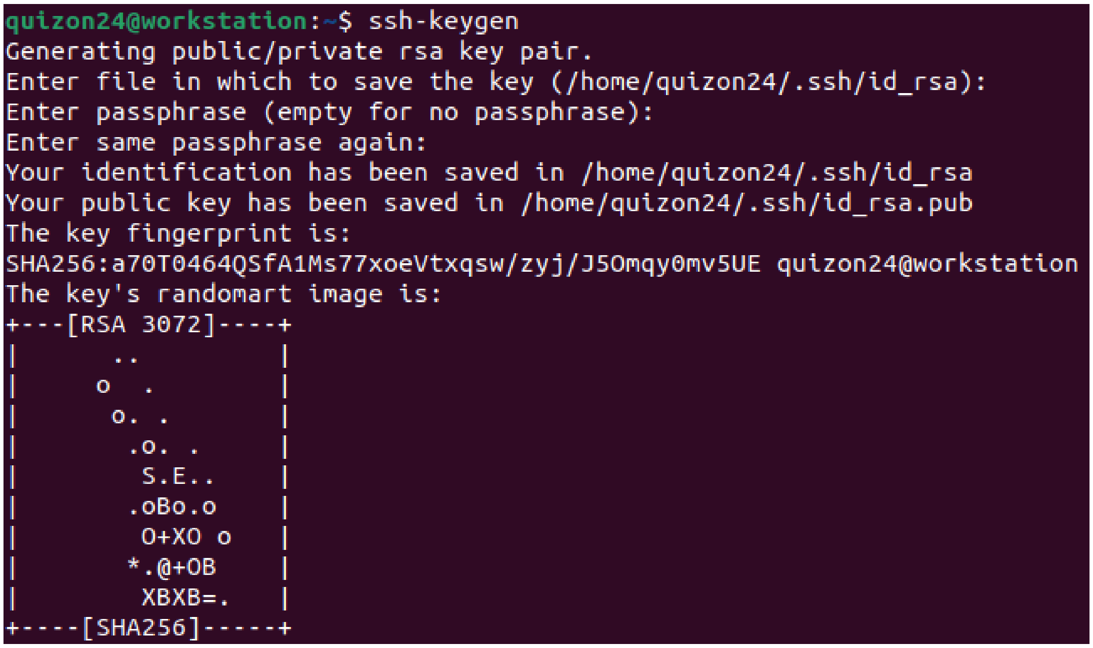
SSH introduced public key authentication as a more secure alternative to the older .rhosts authentication. It improved security by avoiding the need to have password stored in files and eliminated the possibility of a compromised server stealing the user's password.

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

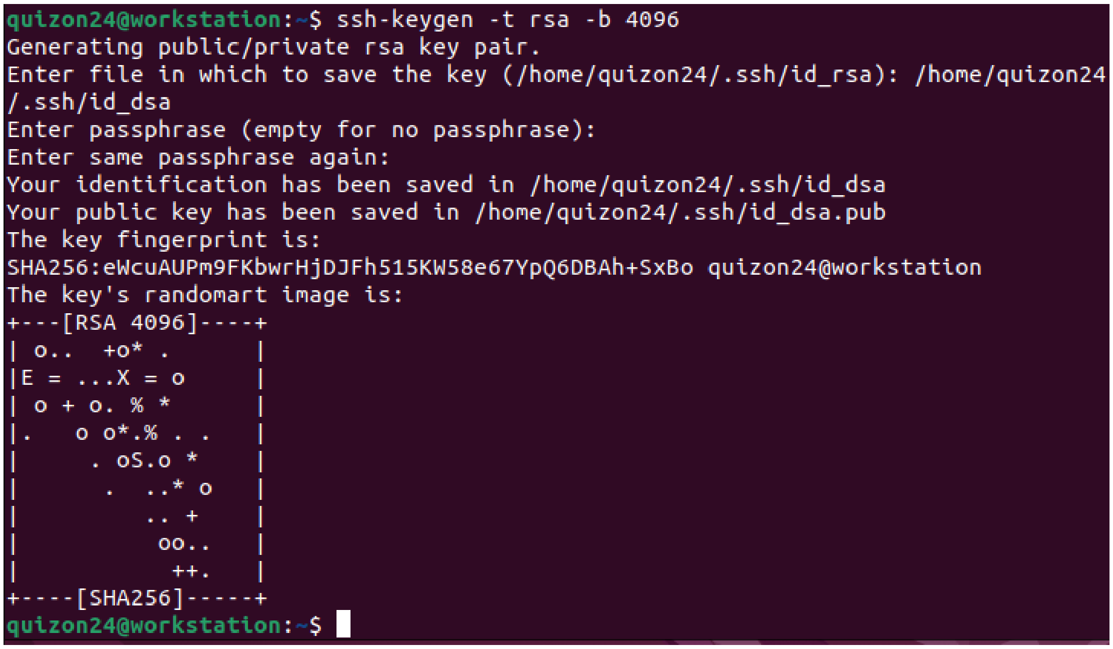
1. The simplest way to generate a key pair is to run *ssh-keygen* without arguments. In this case, it will prompt for the file in which to store keys. First, the tool asked where to save the file. SSH keys for user authentication are usually stored in the users .ssh directory under the home directory.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



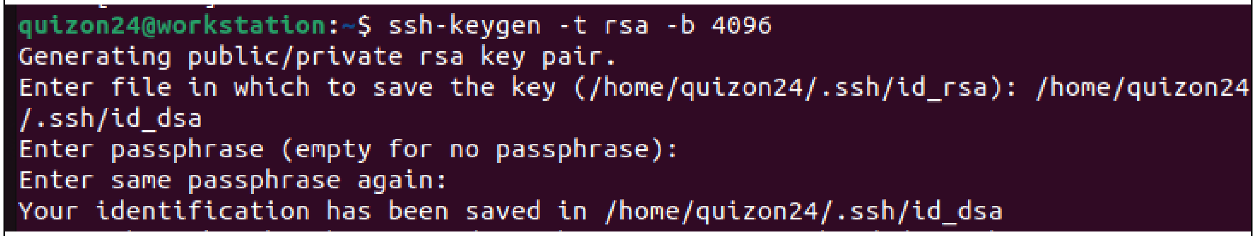
```
quizon24@workstation:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/quizon24/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/quizon24/.ssh/id_rsa
Your public key has been saved in /home/quizon24/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:a70T0464QSfA1Ms77xoeVtxqsw/zyj/J5Omqy0mv5UE quizon24@workstation
The key's randomart image is:
+---[RSA 3072]----+
|      ..         |
|     o  .        |
|      o. .       |
|       .o. .     |
|        S.E..    |
|       .oBo.o    |
|        O+XO o   |
|       *.@+OB    |
|        XBXB=.   |
+----[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.



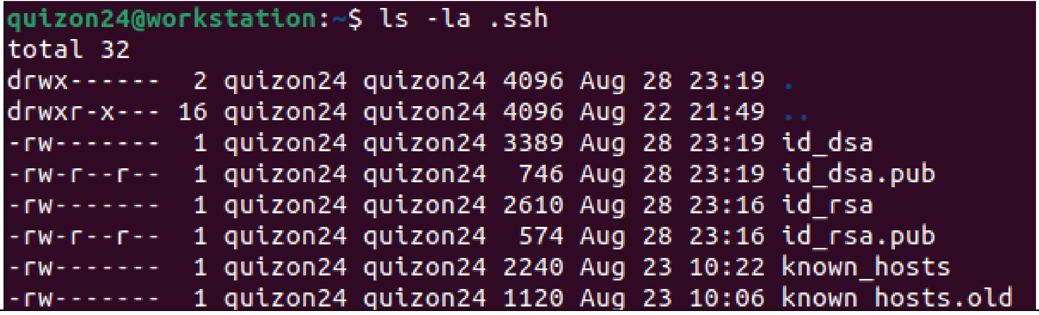
```
quizon24@workstation:~$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/quizon24/.ssh/id_rsa): /home/quizon24
/.ssh/id_dsa
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/quizon24/.ssh/id_dsa
Your public key has been saved in /home/quizon24/.ssh/id_dsa.pub
The key fingerprint is:
SHA256:eWcuAUPm9FKbwrHjDJFh515KW58e67YpQ6DBAh+SxBo quizon24@workstation
The key's randomart image is:
+---[RSA 4096]----+
| o..  +o* .      |
|E = ...X = o     |
| o + o. % *      |
|.   o o*.% . .   |
|     . oS.o *    |
|      .  ..* o   |
|         .. +    |
|          oo..   |
|           ++.   |
+----[SHA256]-----+
quizon24@workstation:~$ 
```

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.



```
quizon24@workstation:~$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/quizon24/.ssh/id_rsa): /home/quizon24
/.ssh/id_dsa
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/quizon24/.ssh/id_dsa
```

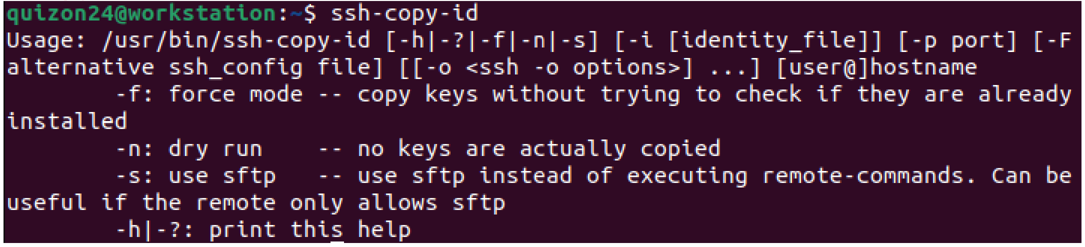
4. Verify that you have created the key by issuing the command *ls -la .ssh.* The command should show the .ssh directory containing a pair of keys. For example, id_rsa.pub and id_rsa.



```
quizon24@workstation:~$ ls -la .ssh
total 32
drwx------  2 quizon24 quizon24 4096 Aug 28 23:19 .
drwxr-x--- 16 quizon24 quizon24 4096 Aug 22 21:49 ..
-rw-------  1 quizon24 quizon24 3389 Aug 28 23:19 id_dsa
-rw-r--r--  1 quizon24 quizon24  746 Aug 28 23:19 id_dsa.pub
-rw-------  1 quizon24 quizon24 2610 Aug 28 23:16 id_rsa
-rw-r--r--  1 quizon24 quizon24  574 Aug 28 23:16 id_rsa.pub
-rw-------  1 quizon24 quizon24 2240 Aug 23 10:22 known_hosts
-rw-------  1 quizon24 quizon24 1120 Aug 23 10:06 known_hosts.old
```

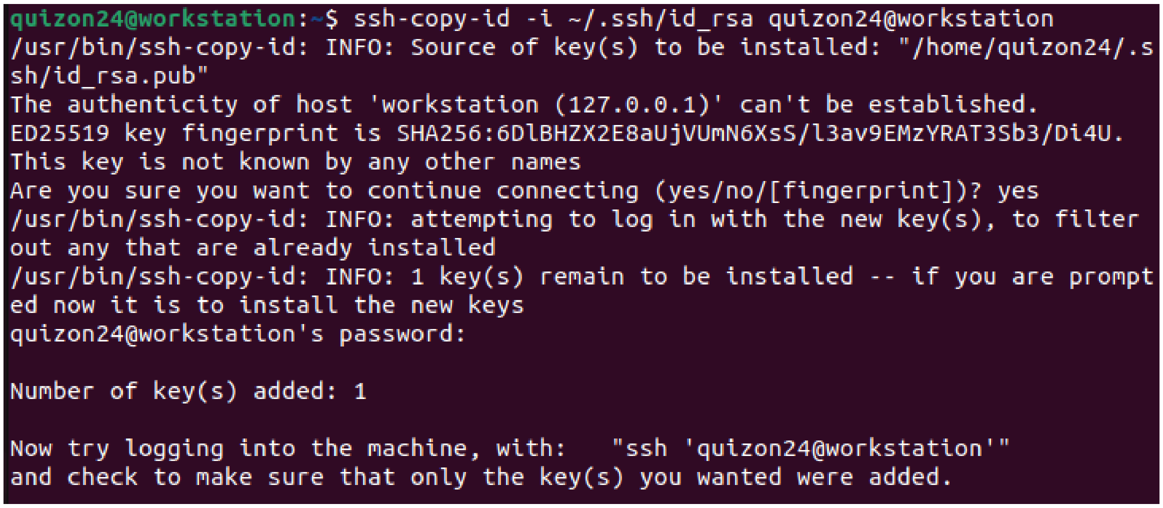
Task 2: Copying the Public Key to the remote servers

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.



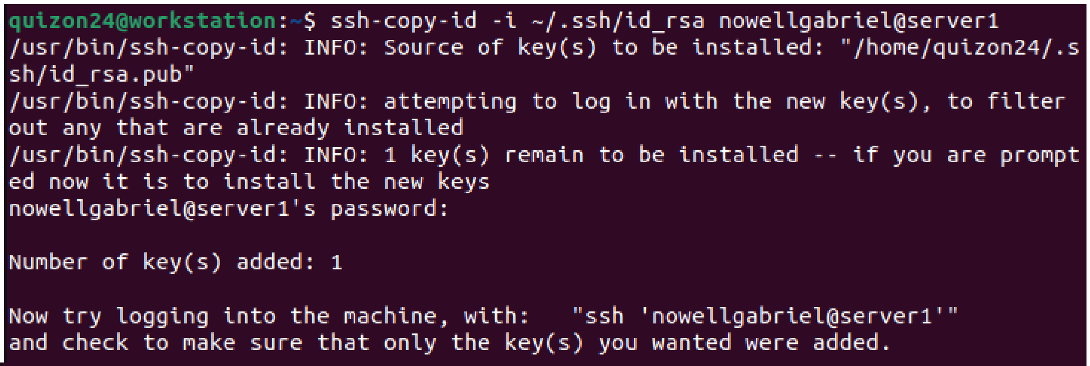
```
quizon24@workstation:~$ ssh-copy-id
Usage: /usr/bin/ssh-copy-id [-h|-?|-f|-n|-s] [-i [identity_file]] [-p port] [-F
alternative ssh_config file] [[-o <ssh -o options>] ...] [user@]hostname
        -f: force mode -- copy keys without trying to check if they are already
installed
        -n: dry run    -- no keys are actually copied
        -s: use sftp   -- use sftp instead of executing remote-commands. Can be
useful if the remote only allows sftp
        -h|-?: print this help
```

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

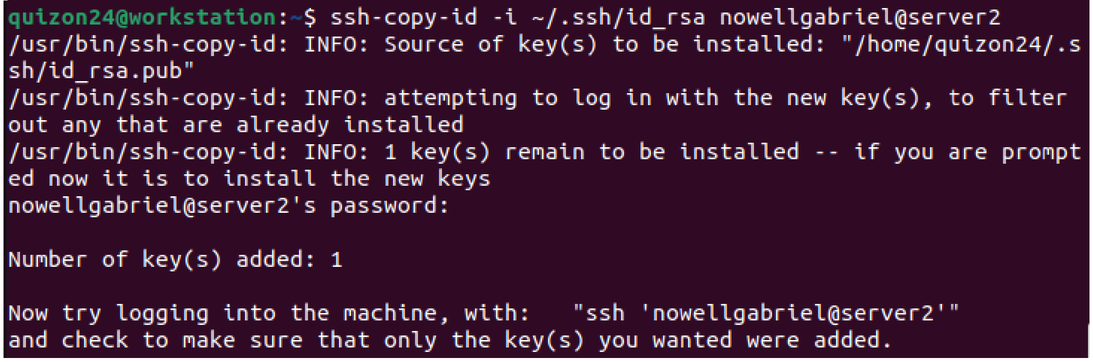


```
quizon24@workstation:~$ ssh-copy-id -i ~/.ssh/id_rsa quizon24@workstation
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/quizon24/.s
sh/id_rsa.pub"
The authenticity of host 'workstation (127.0.0.1)' can't be established.
ED25519 key fingerprint is SHA256:6DlBHZX2E8aUjVUmN6XsS/l3av9EMzYRAT3Sb3/Di4U.
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 filter
out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompt
ed now it is to install the new keys
quizon24@workstation's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'quizon24@workstation'"
and check to make sure that only the key(s) you wanted were added.
```

3. Once the public key has been configured on the server, the server will allow any connecting user that has the private key to log in. During the login process, the client proves possession of the private key by digitally signing the key exchange.



```
quizon24@workstation:~$ ssh-copy-id -i ~/.ssh/id_rsa nowellgabriel@server1
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/quizon24/.s
sh/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 prompt
ed now it is to install the new keys
nowellgabriel@server1's password:

Number of key(s) added: 1

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



```
quizon24@workstation:~$ ssh-copy-id -i ~/.ssh/id_rsa nowellgabriel@server2
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/quizon24/.s
sh/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 prompt
ed now it is to install the new keys
nowellgabriel@server2's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'nowellgabriel@server2'"
and check to make sure that only the key(s) you wanted were added.
```

4. On the local machine, verify that you can SSH with Server 1 and Server 2. What did you notice? Did the connection ask for a password? If not, why?
After I SSH with the two servers, I noticed that it did not ask for the password anymore. This is because of the created authentication key pairs. The created key pairs made logging in quicker and simpler.

Reflections:

Answer the following:

1. How will you describe the ssh-program? What does it do?
   It automates logins. It authenticates hosts.
2. How do you know that you already installed the public key to the remote servers?
   You can find out by trying to login and see if it asks for a password. If it does not, then that means you already installed the public key.

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**).

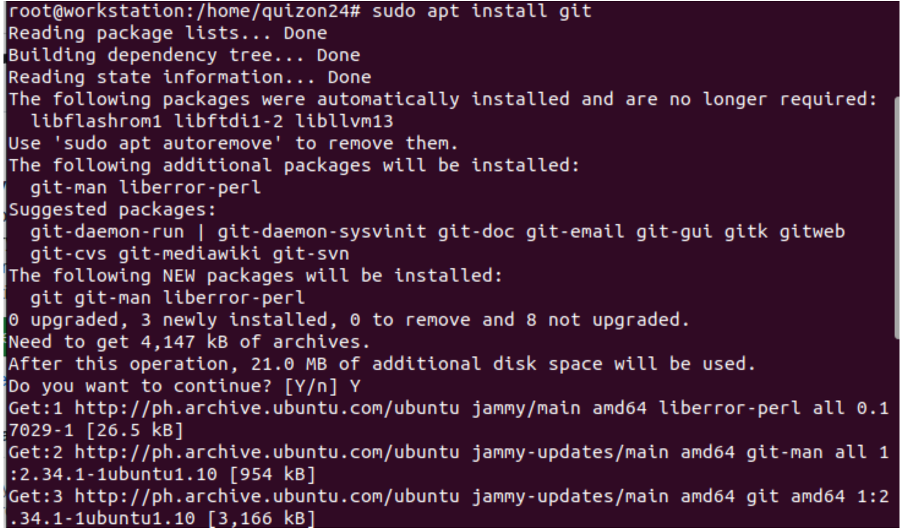
Set up Git

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:

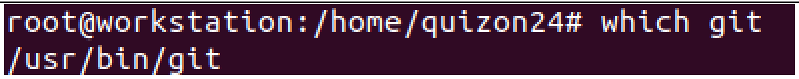
- Creating a repository
- Forking a repository
- Managing files
- Being social

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*

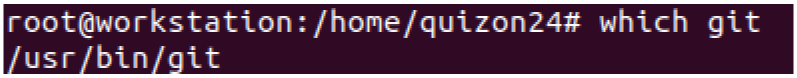


```
root@workstation:/home/quizon24# sudo apt install git
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  libflashrom1 libftdi1-2 libllvm13
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  git-man liberror-perl
Suggested packages:
  git-daemon-run | git-daemon-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 8 not upgraded.
Need to get 4,147 kB of archives.
After this operation, 21.0 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 all 0.1
7029-1 [26.5 kB]
Get:2 http://ph.archive.ubuntu.com/ubuntu jammy-updates/main amd64 git-man all 1
:2.34.1-1ubuntu1.10 [954 kB]
Get:3 http://ph.archive.ubuntu.com/ubuntu jammy-updates/main amd64 git amd64 1:2
.34.1-1ubuntu1.10 [3,166 kB]
```



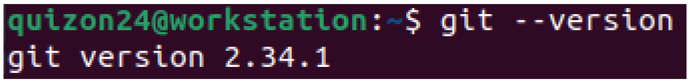
```
root@workstation:/home/quizon24# which git
/usr/bin/git
```

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



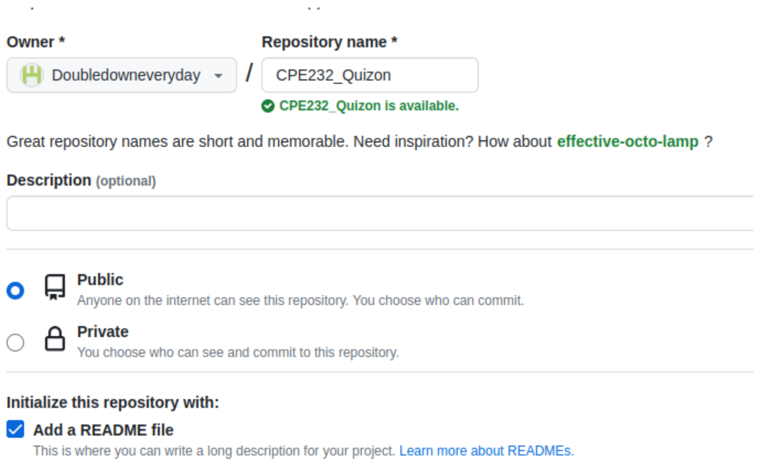
```
root@workstation:/home/quizon24# which git
/usr/bin/git
```

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



```
quizon24@workstation:~$ git --version
git version 2.34.1
```

4. Using the browser in the local machine, go to www.github.com.
5. Sign up in case you don't have an account yet. Otherwise, login to your GitHub account.
   a. Create a new repository and name it as CPE232_yourname. Check Add a README file and click Create repository.



Owner * Doubledowneveryday / Repository name * CPE232_Quizon

✓ CPE232_Quizon is available.

Great repository names are short and memorable. Need inspiration? How about **effective-octo-lamp** ?

Description (optional)

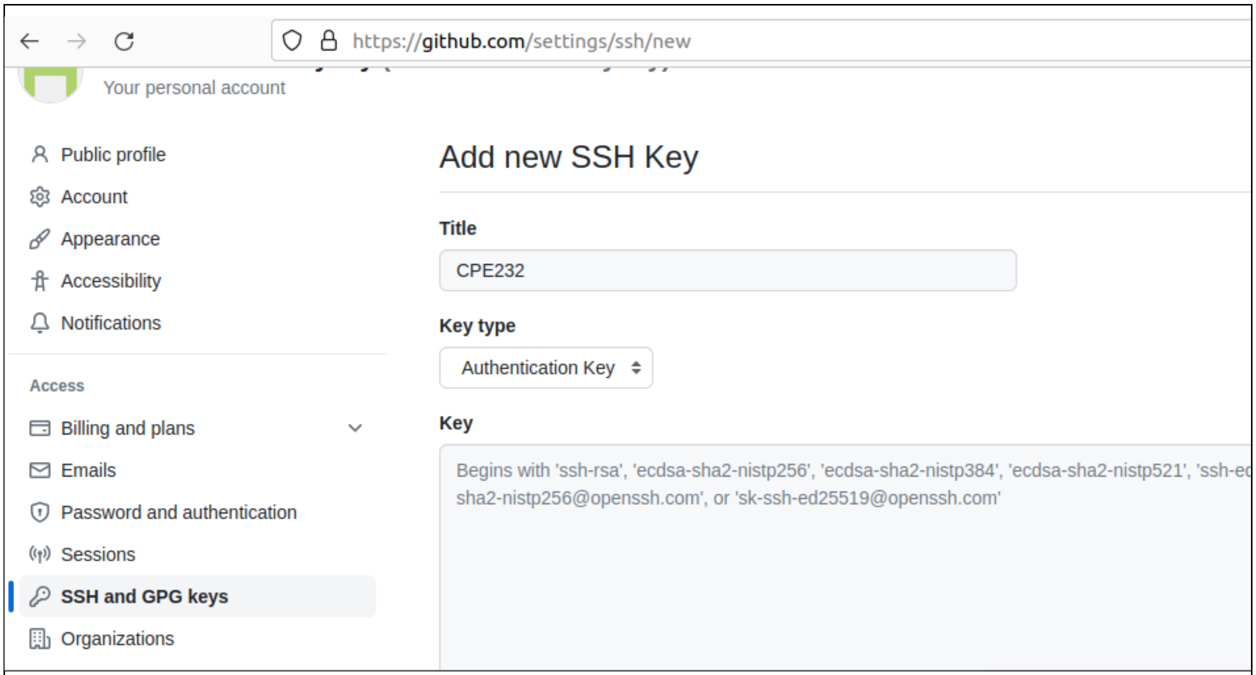
● **Public**
Anyone on the internet can see this repository. You choose who can commit.

○ **Private**
You choose who can see and commit to this repository.

Initialize this repository with:

☑ **Add a README file**
This is where you can write a long description for your project. Learn more about READMEs.

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



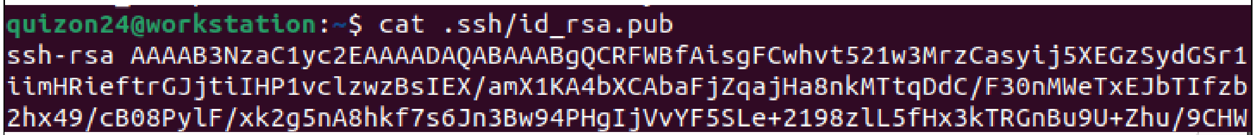
Add new SSH Key

Title: CPE232

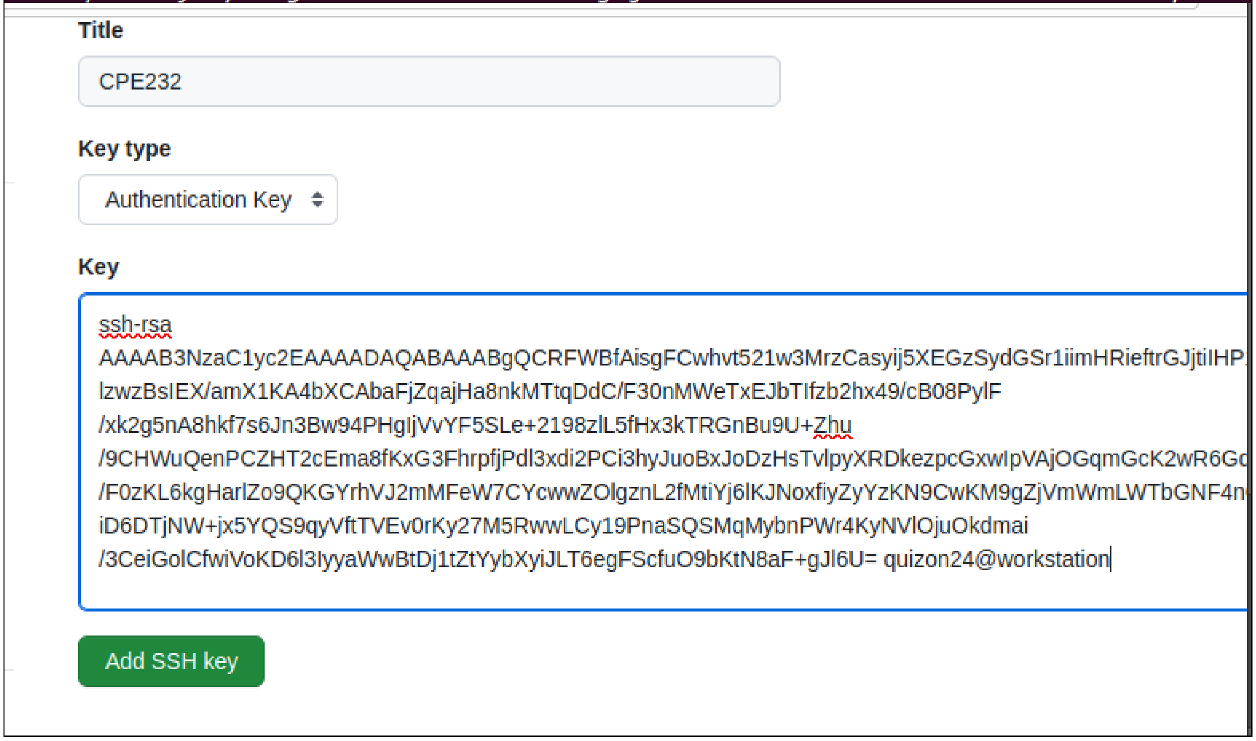
Key type: Authentication Key

Key: Begins with 'ssh-rsa', 'ecdsa-sha2-nistp256', 'ecdsa-sha2-nistp384', 'ecdsa-sha2-nistp521', 'ssh-ed
sha2-nistp256@openssh.com', or 'sk-ssh-ed25519@openssh.com'

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.



```
quizon24@workstation:~$ cat .ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQCRFWBfAisgFCwhvt521w3MrzCasyij5XEGzSydGSr1
iimHRieftrGJjtiIHP1vclzwzBsIEX/amX1KA4bXCAbaFjZqajHa8nkMTtqDdC/F30nMWeTxEJbTIfzb
2hx49/cB08PylF/xk2g5nA8hkf7s6Jn3Bw94PHgIjVvYF5SLe+2198zlL5fHx3kTRGnBu9U+Zhu/9CHW
```



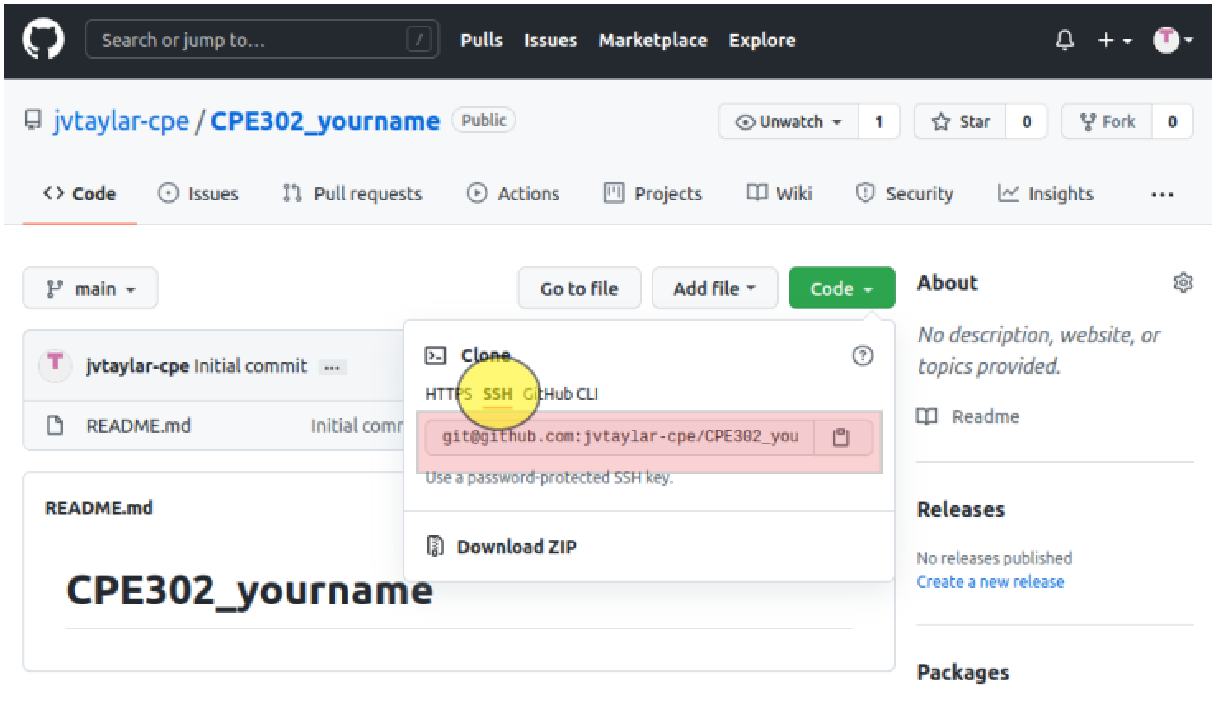
Title: CPE232

Key type: Authentication Key

Key:

```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABgQCRFWBfAisgFCwhvt521w3MrzCasyij5XEGzSydGSr1iimHRieftrGJjtiIHP
lzwzBsIEX/amX1KA4bXCAbaFjZqajHa8nkMTtqDdC/F30nMWeTxEJbTIfzb2hx49/cB08PylF
/xk2g5nA8hkf7s6Jn3Bw94PHgIjVvYF5SLe+2198zlL5fHx3kTRGnBu9U+Zhu
/9CHWuQenPCZHT2cEma8fKxG3FhrpfjPdl3xdi2PCi3hyJuoBxJoDzHsTvlpyXRDkezpcGxwIpVAjOGqmGcK2wR6Gc
/F0zKL6kgHarlZo9QKGYrhVJ2mMFeW7CYcwwZOlgznL2fMtiYj6lKJNoxfiyZyYzKN9CwKM9gZjVmWmLWTbGNF4n
iD6DTjNW+jx5YQS9qyVftTVEv0rKy27M5RwwLCy19PnaSQSMqMybnPWr4KyNVlOjuOkdmai
/3CeiGolCfwiVoKD6l3IyyaWwBtDj1tZtYybXyiJLT6egFScfuO9bKtN8aF+gJl6U= quizon24@workstation
```

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.



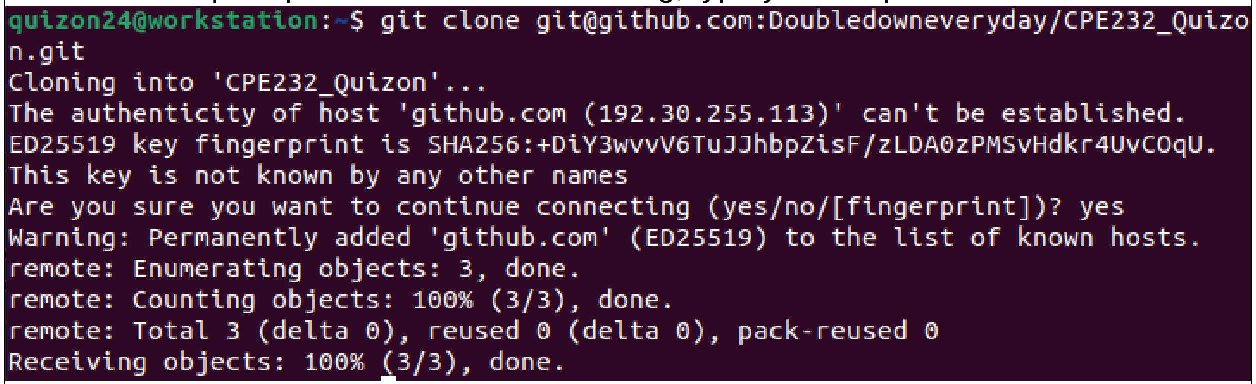
```
git@github.com:jvtaylar-cpe/CPE302_you
```

Use a password-protected SSH key.

No description, website, or topics provided.

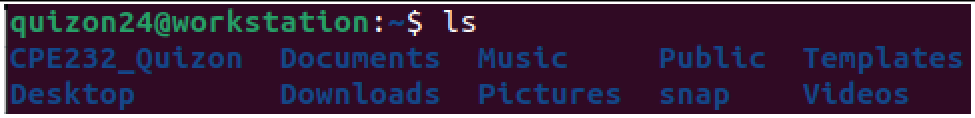
No releases published

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.



```
quizon24@workstation:~$ git clone git@github.com:Doubledowneveryday/CPE232_Quizo
n.git
Cloning into 'CPE232_Quizon'...
The authenticity of host 'github.com (192.30.255.113)' 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.
```

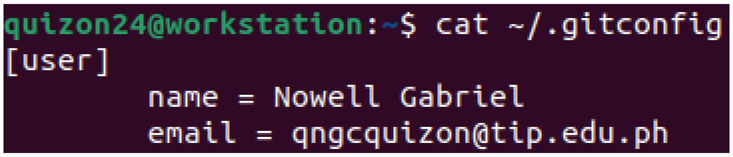
f. To verify that you have cloned the GitHub repository, issue the command *ls*. Observe that you have the CPE232_yourname in the list of your directories. Use CD command to go to that directory and LS command to see the file README.md.



```
quizon24@workstation:~$ ls
CPE232_Quizon  Documents  Music     Public  Templates
Desktop        Downloads  Pictures  snap    Videos
```

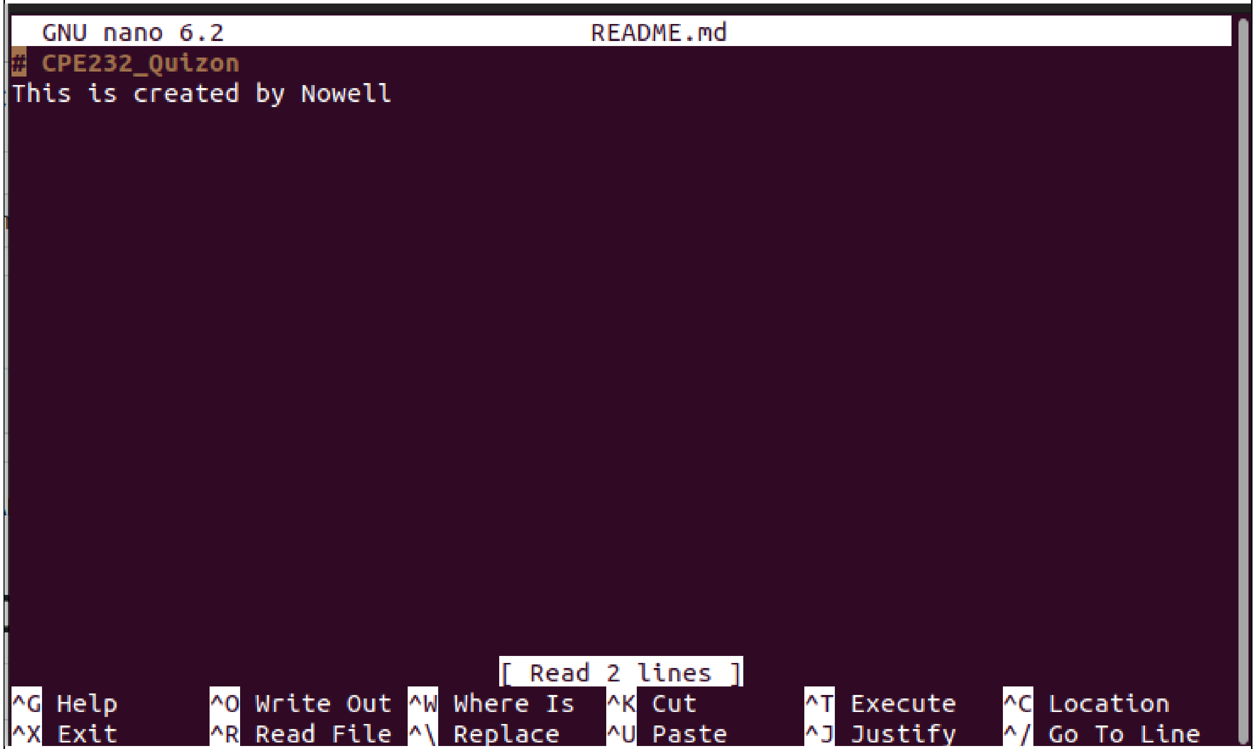
g. Use the following commands to personalize your git.

- *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*



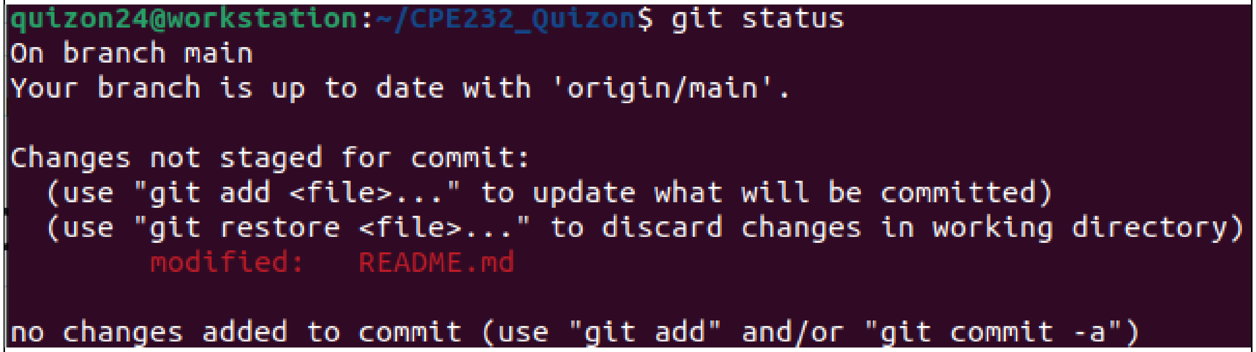
```
quizon24@workstation:~$ cat ~/.gitconfig
[user]
        name = Nowell Gabriel
        email = qngcquizon@tip.edu.ph
```

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.



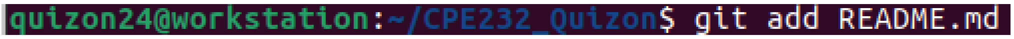
```
  GNU nano 6.2                          README.md
# CPE232_Quizon
This is created by Nowell








                                 [ Read 2 lines ]
^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?



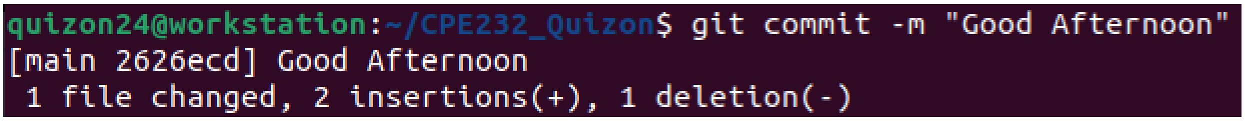
```
quizon24@workstation:~/CPE232_Quizon$ git status
On branch main
Your branch is up to date with 'origin/main'.

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.



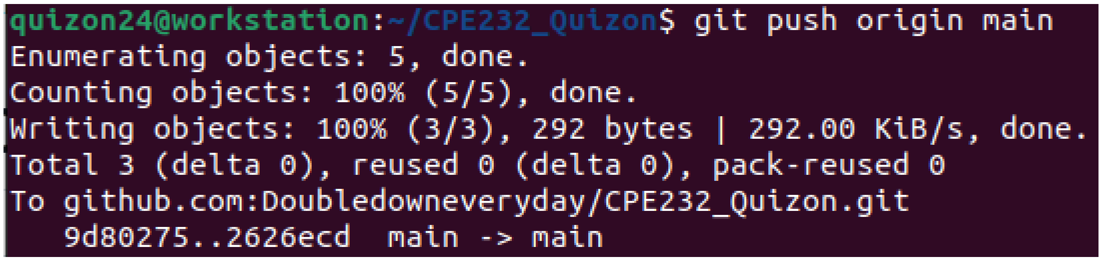
```
quizon24@workstation:~/CPE232_Quizon$ git add README.md
```

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.



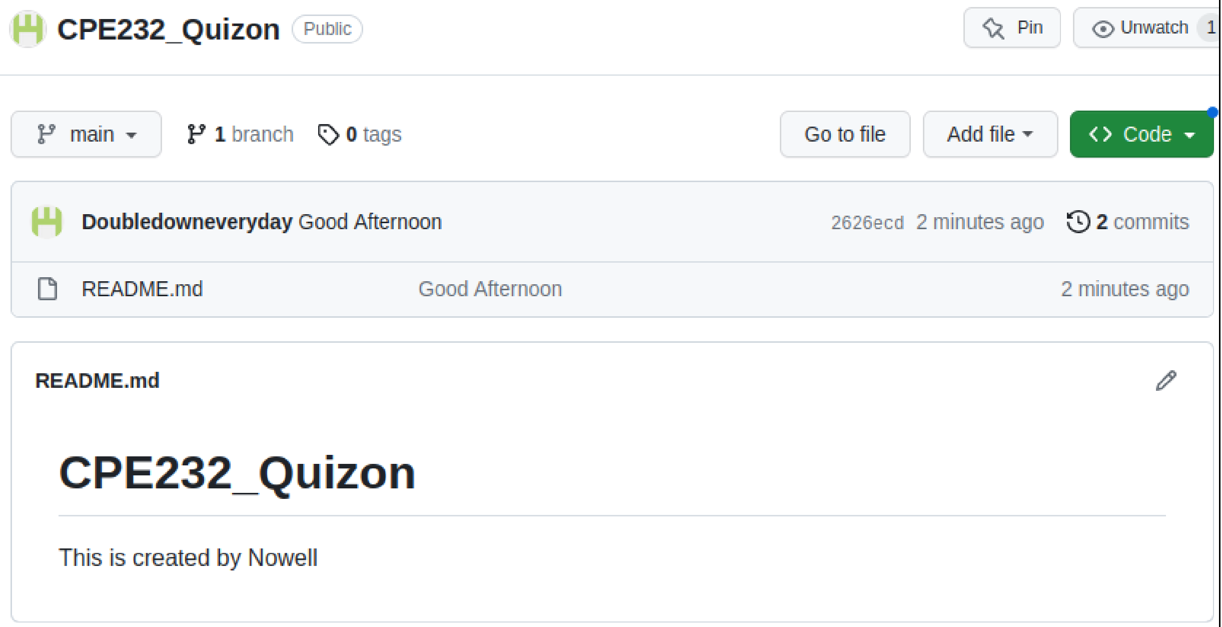
```
quizon24@workstation:~/CPE232_Quizon$ git commit -m "Good Afternoon"
[main 2626ecd] Good Afternoon
 1 file changed, 2 insertions(+), 1 deletion(-)
```

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.*



```
quizon24@workstation:~/CPE232_Quizon$ git push origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Writing objects: 100% (3/3), 292 bytes | 292.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:Doubledowneveryday/CPE232_Quizon.git
   9d80275..2626ecd  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.



CPE232_Quizon

Doubledowneveryday Good Afternoon		2626ecd 2 minutes ago 2 commits
README.md	Good Afternoon	2 minutes ago

README.md

CPE232_Quizon

This is created by Nowell

Reflections:

Answer the following:

3. What sort of things have we so far done to the remote servers using ansible commands?
   We were able to create authorized keys for the remote servers through SSH for quicker authentication without the use of passwords.
4. How important is the inventory file?
   The inventory file is where administrators can keep track of the remote servers that they manage and is where all ansible commands are stored.

Conclusions/Learnings:

In this activity I learned how to create public and private keys that helped in making authentication and logins in different servers easier and more secure. Also, I learned how to use git and git commands.