

Name: Christian Dave L. Berja	Date Performed:28/08/2023
Course/Section: CPE31S5	Date Submitted:
Instructor: Engr. Roman	Semester and SY: 1st sem 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). <i>Provide screenshots for each task.</i> 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 <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*.

```
Enter same passphrase again:
Passphrases do not match. Try again.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Passphrases do not match. Try again.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
sYour identification has been saved in id_rsa
Your public key has been saved in id_rsa.pub
The key fingerprint is:
SHA256:FG6C24v36geDkF9Xh2u0elfe0kUqizD19HSCnn8Tub8 berja@localmachine
The key's randomart image is:
+---[RSA 3072]---+
|      .      |
|    . . .O .  |
| .. . +. O .   |
| o  o.+ . + o o |
| o.o..S= + + =. |
| o.o.+ . * oo.  |
| . o+ o + * .o  |
| ...O o + +o.   |
| .o+.. . E=     |
+-----[SHA256]-----+
berja@localmachine:~$ s
```

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.

```
berja@localmachine:~$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/berja/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/berja/.ssh/id_rsa
Your public key has been saved in /home/berja/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:R2sWXPvOuqCxtbH/c8H0w53poK8am789VaDKeYDQjy4 berja@localmachine
The key's randomart image is:
+---[RSA 4096]-----+
|
| . . . . .
| . ++ . . .
| o.OO.. ..
| .S.=+ = *
| E .++ .+ Oo|
| ...+.. * o|
| =+*.O...|
| o+==B=oo|
+-----[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.

```
berja@localmachine:~$ ls -la .ssh
total 24
drwx----- 2 berja berja 4096 Aug 28 14:07 .
drwxr-x--- 15 berja berja 4096 Aug 28 14:01 ..
-rw----- 1 berja berja 3381 Aug 28 14:07 id_rsa
-rw-r--r-- 1 berja berja 744 Aug 28 14:07 id_rsa.pub
-rw----- 1 berja berja 978 Aug 23 23:48 known_hosts
-rw-r--r-- 1 berja berja 142 Aug 23 23:48 known_hosts.old
berja@localmachine:~$
```

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.

```
berja@localmachine:~$ 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 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
berja@localmachine:~$
```

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.

```
berja@localmachine:~$ ssh-copy-id -i ~/.ssh/id_rsa berja@bserver1
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/berja/.ssh/
id_rsa.pub"
The authenticity of host 'bserver1 (192.168.56.109)' can't be established.
ED25519 key fingerprint is SHA256:YU2zYAmfEhMKhhnQBYL6NmJlEnTlVUnFvpo58eZ08Ps.
This host key is known by the following other names/addresses:
  ~/.ssh/known_hosts:1: [hashed name]
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
berja@bserver1's password:

Number of key(s) added: 1

Now try logging into the machine, with:  "ssh 'berja@bserver1'"
and check to make sure that only the key(s) you wanted were added.
```

```
berja@localmachine:~$ ssh-copy-id -i ~/.ssh/id_rsa berja2@bserver2
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/berja/.ssh/id_rsa.pub"
The authenticity of host 'bserver2 (192.168.56.110)' can't be established.
ED25519 key fingerprint is SHA256:s9k8cv0uqQfW1ezfZ05kJnN02oGJT4/iY4PY5WMcrSk.
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 al
ady installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to ins
ll the new keys
berja2@bserver2's password:

Number of key(s) added: 1

Now try logging into the machine, with:  "ssh 'berja2@bserver2'"
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?

```
berja@localmachine:~$ ssh bserver1
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 5.15.0-78-generic x86_64)

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

System information as of Mon Aug 28 14:12:04 UTC 2023

System load:  0.0               Processes:    104
Usage of /:   36.9% of 8.02GB   Users logged in: 1
Memory usage: 5%              IPv4 address for enp0s3: 192.168.56.109
Swap usage:   0%

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your Internet connection o
r proxy settings

Last login: Mon Aug 28 13:51:51 2023
berja@bserver1:~$
```

```
berja@localmachine:~$ ssh berja2@server2
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 5.15.0-78-generic x86_64)

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

System information as of Mon Aug 28 14:50:56 UTC 2023

System load:  0.0205078125      Processes:            104
Usage of /:   36.6% of 8.02GB   Users logged in:     1
Memory usage: 5%               IPv4 address for enp0s3: 192.168.56.110
Swap usage:   0%

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status
```

When using the ssh to check the server1 the key in the ssh did not ask for the password to enter but when I used the server 2 it asked for the password but error occurred during the process, by typing the right key it inputted "permission denied". Since I inputted the whole user and hosts it let me in inside the server 2.

Reflections:

Answer the following:

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

The ssh specially known as Secure shell, it is a network protocol gives the users more secure way to access the computer over on an unsecured network, and system administration. It has very good access more on the encryption and good connection to the client to server. It allows users to use public-key cryptography for authentication. This method is more secure and convenient as it eliminates the need to transmit passwords over the network.

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

We can know that it is already installed public key to the remote servers, is by inputting "ls -la .ssh" command to show if it is already in the system. Do a test to check if it can be detected.

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*

```
berja@localmachine:~$ sudo apt install git
[sudo] password for berj:
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 2 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 2
:2.34.1-1ubuntu1.10 [954 kB]
Get:3 http://ph.archive.ubuntu.com/ubuntu jammy-updates/main amd64 git amd64 2:
.34.1-1ubuntu1.10 [3,166 kB]
Fetched 4,147 kB in 5s (780 kB/s)
Selecting previously unselected package liberror-perl.
```

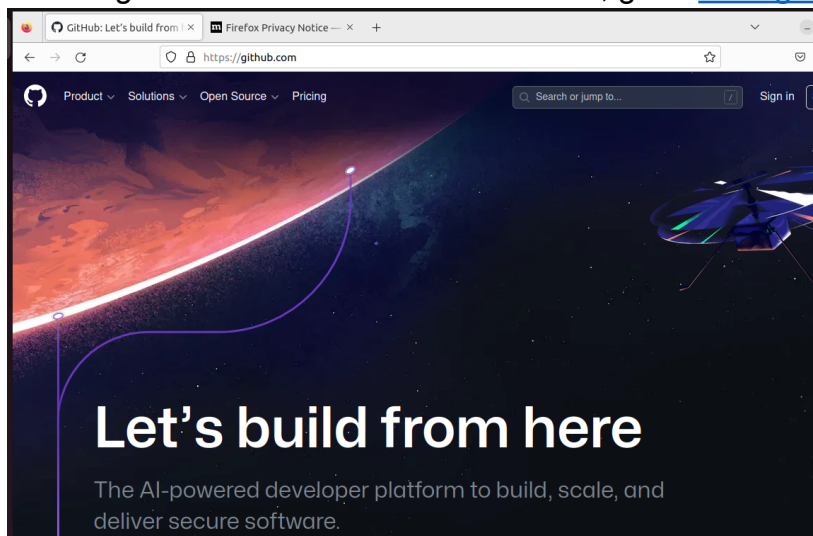
2. After the installation, issue the command *which git* again. The directory of git is usually installed in this location: *user/bin/git*.

```
berja@localmachine:~$ which git
/usr/bin/git
berja@localmachine:~$
```

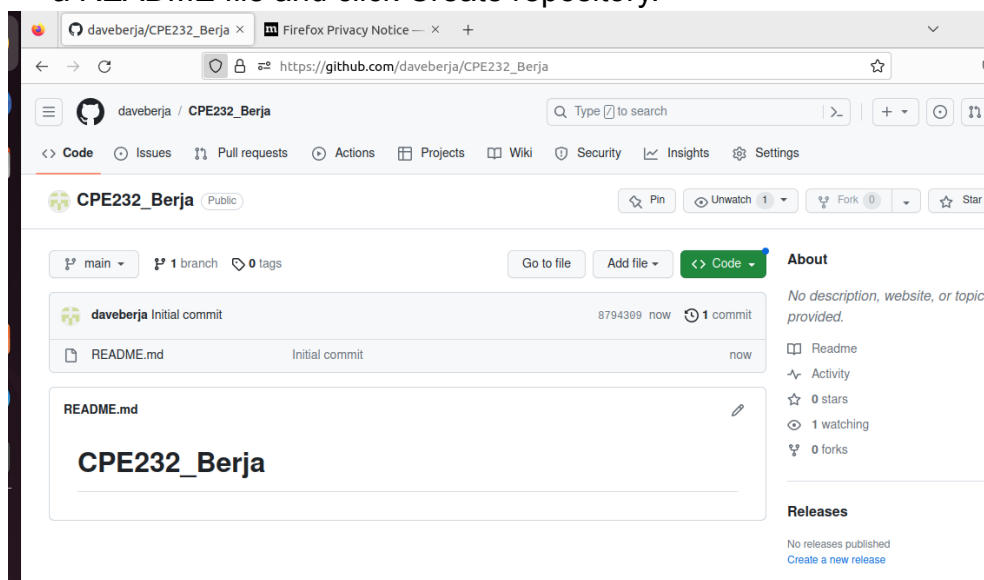
3. The version of git installed in your device is the latest. Try issuing the command *git --version* to know the version installed.


```
berja@localmachine:~$ git --version  
git version 2.34.1  
berja@localmachine:~$
```

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.




- 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.

SSH keys

[New SSH key](#)

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

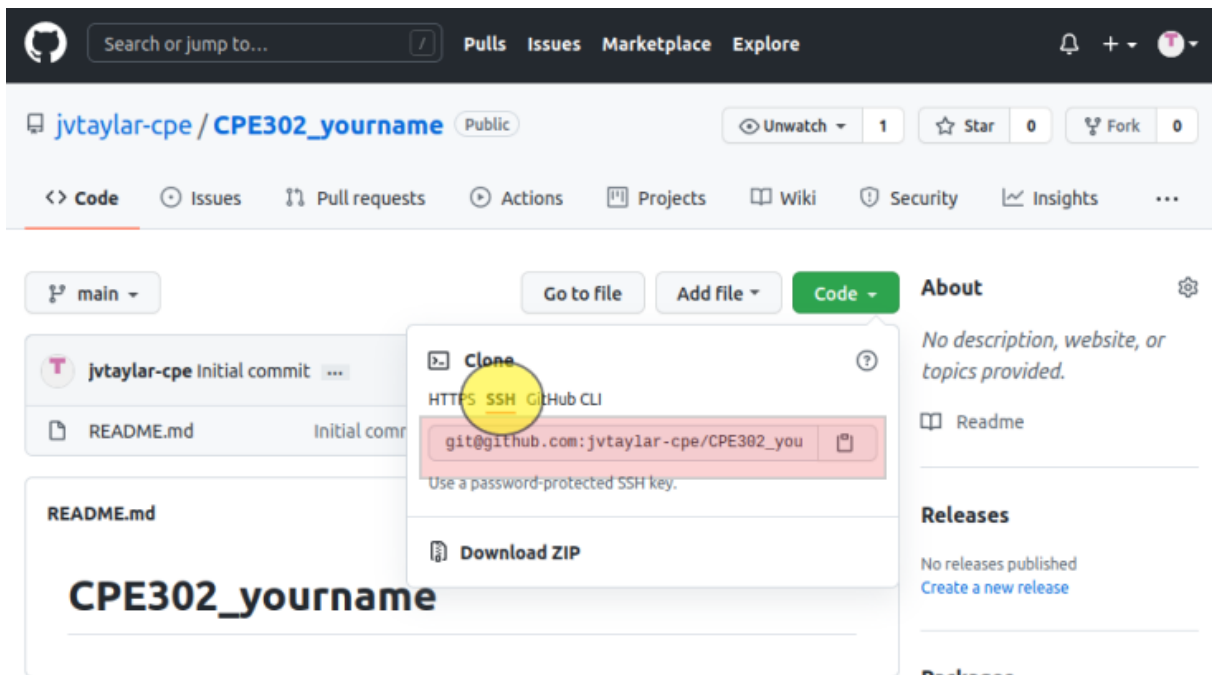
Authentication Keys

**CPE232**
SHA256:Uet7gKa/71KobNXd70fWRjW0wmLuCJNVHwuH9DfP67w
Added on Aug 29, 2023
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.


```
berja@localmachine:~$ git clone git@github.com:daveberja/CPE232_Berja.git
Cloning into 'CPE232_Berja'...
The authenticity of host 'github.com (192.30.255.112)' 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.
berja@localmachine:~$
```

- 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.

```
berja@localmachine:~$ ls
CPE232_Berja  Desktop  Downloads  id_dsa  id_rsa  id_rsa.pub  Music  Pictures  snap  Templates  Videos
berja@localmachine:~$
```

- 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`

```
berja@localmachine:~$ git config --global user.name "Berja"
berja@localmachine:~$ git config --global user.email "qcdlberja@tip.edu.ph"
berja@localmachine:~$ cat ~/.gitconfig
[user]
    name = Berja
    email = qcdlberja@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.

```
berja@localmachine:~/CPE232_Berja
GNU nano 6.2 README.md
# CPE232_Berja

ggwp
```

- 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?

```
berja@localmachine:~/CPE232_Berja$ 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")
berja@localmachine:~/CPE232_Berja$
```

You can think of using the `git status` command to encapsulate the current state of your repository. It disassembles the files created for those who have changed but are not yet saved, those who are committing, and the novel's files elude detection. Armed with this information, you can Use caution when choosing your commits and designing your subsequent moves, keeping an eye out for any changes that might be occurring within your project.

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

```
berja@localmachine:~/CPE232_Berja$ git add README.md
```

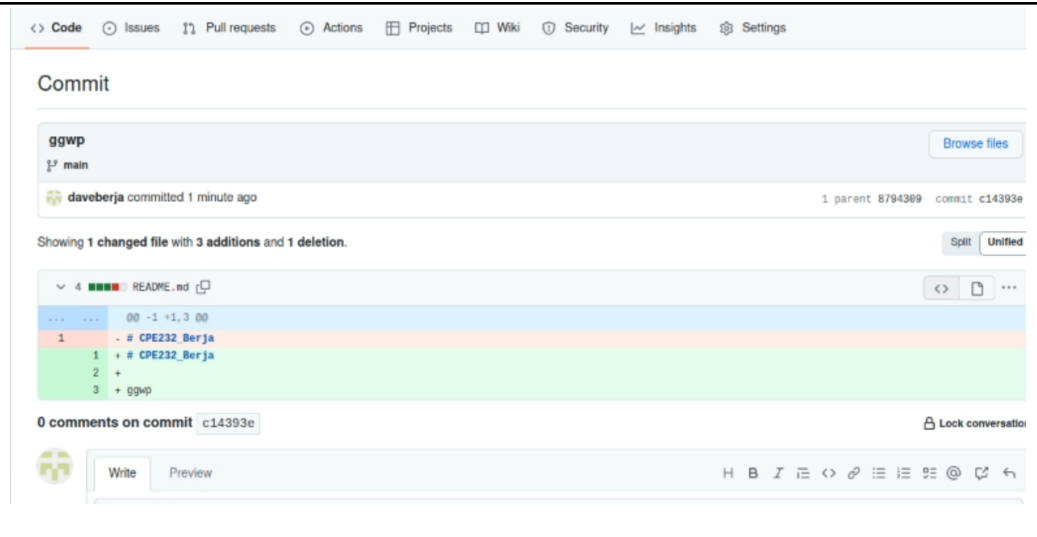
- k. Use the `git commit -m "your message"` to create a snapshot of the staged changes along the timeline of the Git projects history. The use of this command is required to select the changes that will be staged for the next commit.

```
berja@localmachine:~/CPE232_Berja$ git commit -m "ggwp"
[main c14393e] ggwp
1 file changed, 3 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`.

```
berja@localmachine:~/CPE232_Berja$ git push origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Writing objects: 100% (3/3), 257 bytes | 257.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:daveberja/CPE232_Berja.git
8794309..c14393e main -> main
```

- m. On the GitHub repository, verify that the changes have been made to README.md by refreshing the page. Describe the README.md file. You can notice the how long was the last commit. It should be some minutes ago and the message you typed on the git commit command should be there. Also, the README.md file should have been edited according to the text you wrote.



Reflections:

Answer the following:

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

We were able to access the ssh and access other servers using the ssh. I can connect to several servers remotely by Creating an SSH key that will serve as a key for them while also knowing their IP addresses permanent authentication, so you do not need to enter a password each time. remote access to the server.

4. How important is the inventory file?

The inventory file is crucial when utilizing ansible commands and remote servers. This forms the basis for references that define the properties of the target servers, including hostnames, IP addresses, and other details. The inventory component is crucial because it gives Ansible the context it needs to perform activities across the entire server fleet more effectively and efficiently, enabling automation and clear administration.

Conclusions/Learnings:

In this activity we have learned on how we create an SSH key pair for the User, and also by copying the public key. This lesson helped me to improve my knowledge know more on how it can be created and the authentication keys. By creating the github i manage to connect this to my terminal inside the Ubuntu and after the results is shown we can see the progress in the github.