

<b>Name:</b> Baltazar, Paul Eimar R.	<b>Date Performed:</b> Sep 10, 2024
<b>Course/Section:</b> CPE212 – CPE31S2	<b>Date Submitted:</b> Sep 11, 2024
<b>Instructor:</b> Engr. Robin Valenzuela	<b>Semester and SY:</b> 1st Sem [2024 - 2025]
<b>Activity 2: SSH Key-Based Authentication and Setting up Git</b>	
<b>1. Objectives:</b> <ol style="list-style-type: none"> <li>1.1 Configure remote and local machine to connect via SSH using a KEY instead of using a password</li> <li>1.2 Create a public key and private key</li> <li>1.3 Verify connectivity</li> <li>1.4 Setup Git Repository using local and remote repositories</li> <li>1.5 Configure and Run ad hoc commands from local machine to remote servers</li> </ol>	
<b>Part 1: Discussion</b> <p>It is assumed that you are already done with the last Activity (<b>Activity 1: Configure Network using Virtual Machines</b>). <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><b>What Is ssh-keygen?</b></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><b>SSH Keys and Public Key Authentication</b></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>	
<b>Task 1: Create an SSH Key Pair for User Authentication</b> <ol style="list-style-type: none"> <li>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</li> </ol>	

name depends on the algorithm, in this case *id\_rsa* when using the default RSA algorithm. It could also be, for example, *id\_dsa* or *id\_ecdsa*.

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

```
qperbaltazar@workstation: ~  
qperbaltazar@workstation:~$ ssh-keygen -t rsa -b 4096  
Generating public/private rsa key pair.  
Enter file in which to save the key (/home/qperbaltazar/.ssh/id_rsa):  
Enter passphrase (empty for no passphrase):  
Enter same passphrase again:  
Your identification has been saved in /home/qperbaltazar/.ssh/id_rsa  
Your public key has been saved in /home/qperbaltazar/.ssh/id_rsa.pub  
The key fingerprint is:  
SHA256:CLzS/zjTTmVoJy7nYZKlOGVLYv/Lp9c1MGGu4V05+U qperbaltazar@workstation  
The key's randomart image is:  
+---[RSA 4096]-----+  
|  
| ..  
| .o  
| o o....  
| . o . o +oo o  
| . = S * .o. E  
| o O B = o o  
| o O.* ..  
| .o%..o  
| .+O=  
+-----[SHA256]-----+  
qperbaltazar@workstation:~$
```

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.

```
qperbaltazar@workstation:~$ ls -la .ssh  
total 24  
drwx----- 2 qperbaltazar qperbaltazar 4096 Sep 10 22:02 .  
drwxr-x--- 16 qperbaltazar qperbaltazar 4096 Apr 5 17:36 ..  
-rw----- 1 qperbaltazar qperbaltazar 3389 Sep 10 22:02 id_rsa  
-rw-r--r-- 1 qperbaltazar qperbaltazar 750 Sep 10 22:02 id_rsa.pub
```

## 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?

```
qperbaltazar@workstation:~$ ssh-copy-id -i ~/.ssh/id_rsa qperbaltazar@server1
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/qperbaltazar/.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 already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
qperbaltazar@server1's password:

Number of key(s) added: 1

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

### Copying to server 1

```
qperbaltazar@workstation:~$ ssh-copy-id -i ~/.ssh/id_rsa qperbaltazar@server2
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/qperbaltazar/.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 already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
qperbaltazar@server2's password:

Number of key(s) added: 1

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

### Copying to server 2

```
qperbaltazar@workstation:~$ ssh qperbaltazar@server1
Welcome to Ubuntu 22.04.4 LTS (GNU/Linux 6.5.0-26-generic x86_64)

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

Expanded Security Maintenance for Applications is not enabled.

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

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

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 connection or proxy settings
```

### Workstation to server 1

```
qperbaltazar@workstation:~$ ssh qperbaltazar@server2
Welcome to Ubuntu 22.04.4 LTS (GNU/Linux 6.5.0-26-generic x86_64)

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

Expanded Security Maintenance for Applications is not enabled.

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

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

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 connection or proxy settings

Last login: Sun Aug 25 21:32:13 2024 from 192.168.56.101
```

Workstation to server 2

### Reflections:

Answer the following:

1. How will you describe the ssh-program? What does it do?
  - SH (Secure Shell) is a network protocol designed for secure remote access to computers. It ensures the protection of communications by encrypting the data exchanged between the client and server, safeguarding against unauthorized access and eavesdropping. SSH enables remote login to servers, supports file transfers, and allows for the execution of commands on remote systems.
2. How do you know that you already installed the public key to the remote servers?
  - You can verify the public key installation by checking if the public key is present in the authorization file. You can also check it by testing the SSH connection.

## 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

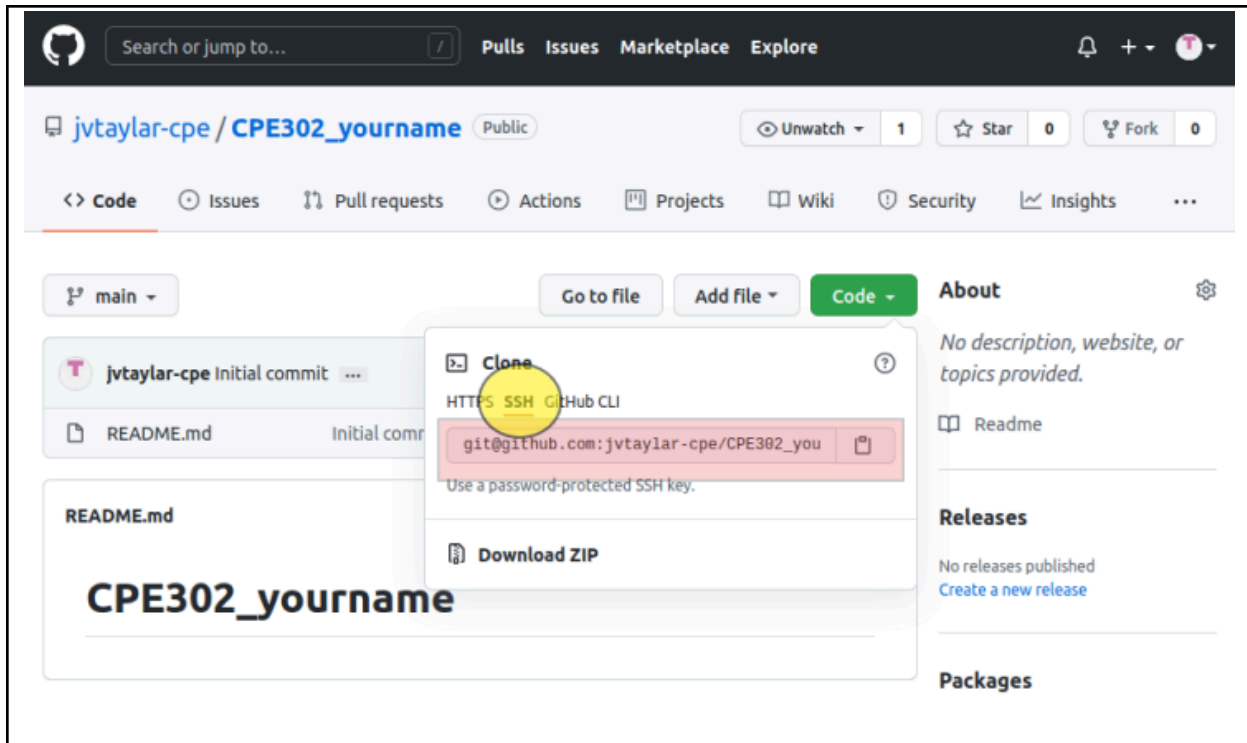
- Being social

### 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*

```
paul_etmar@Workstation:~$ sudo apt install git
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following package was automatically installed and is no longer required:
  libllvm7
Use 'sudo apt autoremove' to remove it.
The following additional packages will be installed:
  git-man liberror-perl
Suggested packages:
  git-daemon-run | git-daemon-sysvinit git-doc git-el 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 0 not upgraded.
Need to get 4,817 kB of archives.
After this operation, 34.3 MB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 http://ph.archive.ubuntu.com/ubuntu bionic/main amd64 liberror-perl all 0.17025-1 [22.8 kB]
Get:2 http://ph.archive.ubuntu.com/ubuntu bionic-updates/main amd64 git-man all 1:2.17.1-1ubuntu0.18 [804 kB]
Get:3 http://ph.archive.ubuntu.com/ubuntu bionic-updates/main amd64 git amd64 1:2.17.1-1ubuntu0.18 [3,990 kB]
Fetched 4,817 kB in 2s (2,038 kB/s)
Selecting previously unselected package liberror-perl.
(Reading database ... 170408 files and directories currently installed.)
Preparing to unpack .../liberror-perl_0.17025-1_all.deb ...
Unpacking liberror-perl (0.17025-1) ...
Selecting previously unselected package git-man.
Preparing to unpack .../git-man_1%3a2.17.1-1ubuntu0.18_all.deb ...
Unpacking git-man (1:2.17.1-1ubuntu0.18) ...
Selecting previously unselected package git.
Preparing to unpack .../git_1%3a2.17.1-1ubuntu0.18_amd64.deb ...
Unpacking git (1:2.17.1-1ubuntu0.18) ...
Setting up git-man (1:2.17.1-1ubuntu0.18) ...
Setting up liberror-perl (0.17025-1) ...
Setting up git (1:2.17.1-1ubuntu0.18) ...
Processing triggers for man-db (2.8.3-2ubuntu0.1) ...
```

2. After the installation, issue the command *which git* again. The directory of git is usually installed in this location: *user/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.
4. Using the browser in the local machine, go to [www.github.com](http://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.
  - 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/CPE302\_yourname.git`. When prompted to continue connecting, type yes and press enter.

f. To verify that you have cloned the GitHub repository, issue the command `ls`. Observe that you have the `CPE302_yourname` in the list of your directories. Use `CD` command to go to that directory and `LS` command to see the file `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`

```
qperbaltazar@workstation: ~/CPE212_Baltazar1
GNU nano 6.2 README.md *
# CPE212_Baltazar1
Owner Name: Paul Eimar R. Baltazar

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

- 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.
- 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?

```
paul_einar@workstation:~$ git --version
git version 2.17.1
paul_einar@workstation:~$ cat .ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQCT4qLk1MuTRV29eap7U9Hj5Tbk1To5zj6ldV3Svo1IogW5ptw+twEFHwYnsd8C+52FuyWMOtNCBHTcyPPDFPLVKHQQuP9KweAvG4oF06xM+/q8Dn3wkl3stgcExeFl3rtuuVGSYdQ9otv2EekQzm7Y88met36dnldHRHgF3
3Zt85Rw8XfY7ZkqW2UdZ53jyWQW7fSR20xVYD05D7f7Z88n/2FufZevB08K8NLMLtTEUOVXl8vltLV8odnSA3J8KXW7fncrjYF25vYnu4EFLWNSPZyulVhFud30TCD2LmqeLP2XvYtqY7ZE7AuscJfmskw2dVQ0KMDvnrKtd604180dWQ3PMQDEwVSElLXDCnzR059cbZADf
VtPq0W1T0159vZnPRC3XGdV8V9P3E1ZTQvPefjooFIICLMD7z+7p3Xch9Pw+7Cq5dH+ZnBcofIndrYqFOepuKqYD9L+2Gygs5ewtcl2jZnDvNkKCE4f/+Hw0rPGVBoP3Frj34Tgukw85009Lxp+9dCvE3j5clqgaQn0K8actgK5eHt/Q1s3chQLbzysKdhr0Lkr
4d5Kqec9151772w740joZ5y2/z0/64EQBIwz+PFA+PHDDMNK+e5CYUUMHHRq8Gu18H0JacoFGCJTTLGd3F8Q== paul_einar@workstation
paul_einar@workstation:~$ git clone git@github.com:rp1dpaul/CPE212_Baltazar.git
Cloning into 'CPE212_Baltazar'...
The authenticity of host 'github.com (20.205.243.166)' can't be established.
ECDSA key fingerprint is SHA256:p2QAMXNIC1JYMe10trVC98/R1BUFWu3/LyKguFQM.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added github.com (20.205.243.166) (ECDSA) to the list of known hosts.
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
paul_einar@workstation:~$ ls
CPE212_Baltazar Desktop Documents Downloads examples.desktop Music Pictures Public Templates Videos
paul_einar@workstation:~$ cd CPE212_Baltazar
paul_einar@workstation:~/CPE212_Baltazar$ ls
README.md
paul_einar@workstation:~/CPE212_Baltazar$ git --global user.name "Paul Einar"
unknown option: --global
usage: git [--version] [--help] [-C <path>] [-c <name>=<value>]
[--exec-path<=path>] [--html-path] [--man-path] [--info-path]
[-p | --paginate] | --no-pager] [--no-replace-objects] [--bare]
[--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
<command> [<args>]
paul_einar@workstation:~/CPE212_Baltazar$ git config --global user.name "Paul Einar"
paul_einar@workstation:~/CPE212_Baltazar$ git config --global user.email "pau1einar8904@gmail.com"
paul_einar@workstation:~/CPE212_Baltazar$ cat ~/.gitconfig
[user]
    name = Paul Einar
    email = pau1einar8904@gmail.com
paul_einar@workstation:~/CPE212_Baltazar$ sudo nano README
[sudo] password for paul_einar:
paul_einar@workstation:~/CPE212_Baltazar$ ls
README README.md
paul_einar@workstation:~/CPE212_Baltazar$ sudo nano README.md
paul_einar@workstation:~/CPE212_Baltazar$ 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 checkout -- <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.
- 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.



```
qperbaltazar@workstation:~/CPE212_Baltazar1$ git status
On branch main
Your branch is up to date with 'origin/main'.

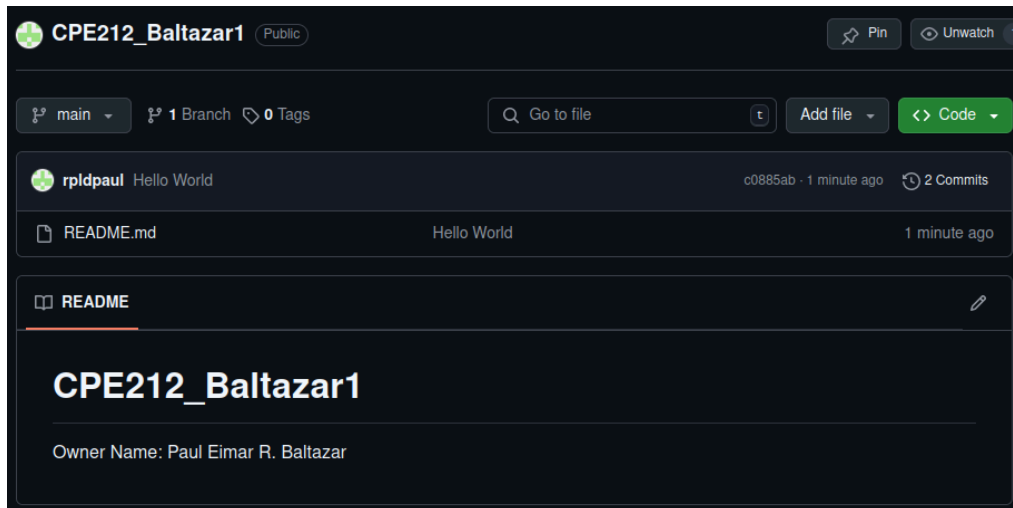
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   README.md

no changes added to commit (use "git add" and/or "git commit -a")
qperbaltazar@workstation:~/CPE212_Baltazar1$ git add README.md
qperbaltazar@workstation:~/CPE212_Baltazar1$ git commit -m "Hello World"
[main c0885ab] Hello World
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`.

```
qperbaltazar@workstation:~/CPE212_Baltazar1$ 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), 294 bytes | 294.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:rpldpaul/CPE212_Baltazar1.git
d078a29..c0885ab  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.



The screenshot shows the GitHub interface for a repository named 'CPE212\_Baltazar1' (Public). The repository is owned by 'rpldpaul'. The main branch is selected, showing 1 branch and 0 tags. A commit titled 'Hello World' by 'rpldpaul' is displayed, with commit hash 'c0885ab' and a timestamp of '1 minute ago'. Below the commit, the 'README' file is shown, containing the text 'CPE212\_Baltazar1' and 'Owner Name: Paul Eimar R. Baltazar'.



## Reflections:

Answer the following:

3. What sort of things have we so far done to the remote servers using ansible commands?
  - In this activity, we have created a public key in order to gain access to remote servers without the need for a password prompt and also linked a git repository into a Virtual Machine. We also demonstrated the capabilities of git commands in Ubuntu.
4. How important is the inventory file?
  - The inventory file uses SSH for authentication and it serves as an area for storing remote server IP addresses together with their hostnames.

## Conclusions/Learnings:

Using public key authentication makes SSH connections secure and lets you directly access any device in the network where SSH is being used. The public and private keys are important because they ensure only authorized people can access the remote computers. Setting up a Git repository on both local and remote machines is a useful tool for managing versions and working together. It also shows that SSH is great for running commands from your local computer to a remote server for easy management.