Name: Paz, John Glen L.	**Date Performed: Aug. 24, 2023**
Course/Section: CPE 232 - CPE31S6	**Date Submitted: Aug. 25, 2023**
Instructor: Dr. Jonathan F. Taylar	**Semester and SY: 1st Sem. 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.

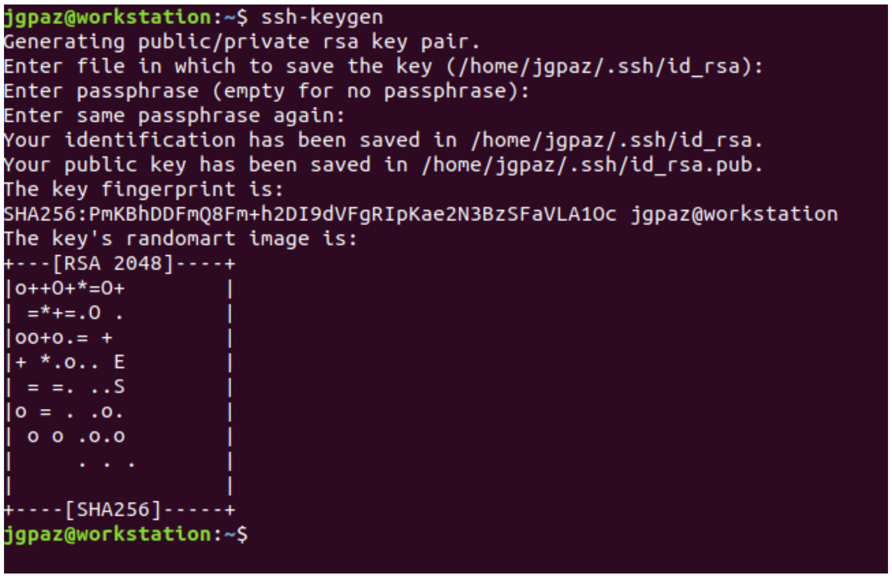
SSH introduced public key authentication as a more secure alternative to the older .rhosts authentication. It improved security by avoiding the need to have password stored in files and eliminated the possibility of a compromised server stealing the user's password.

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

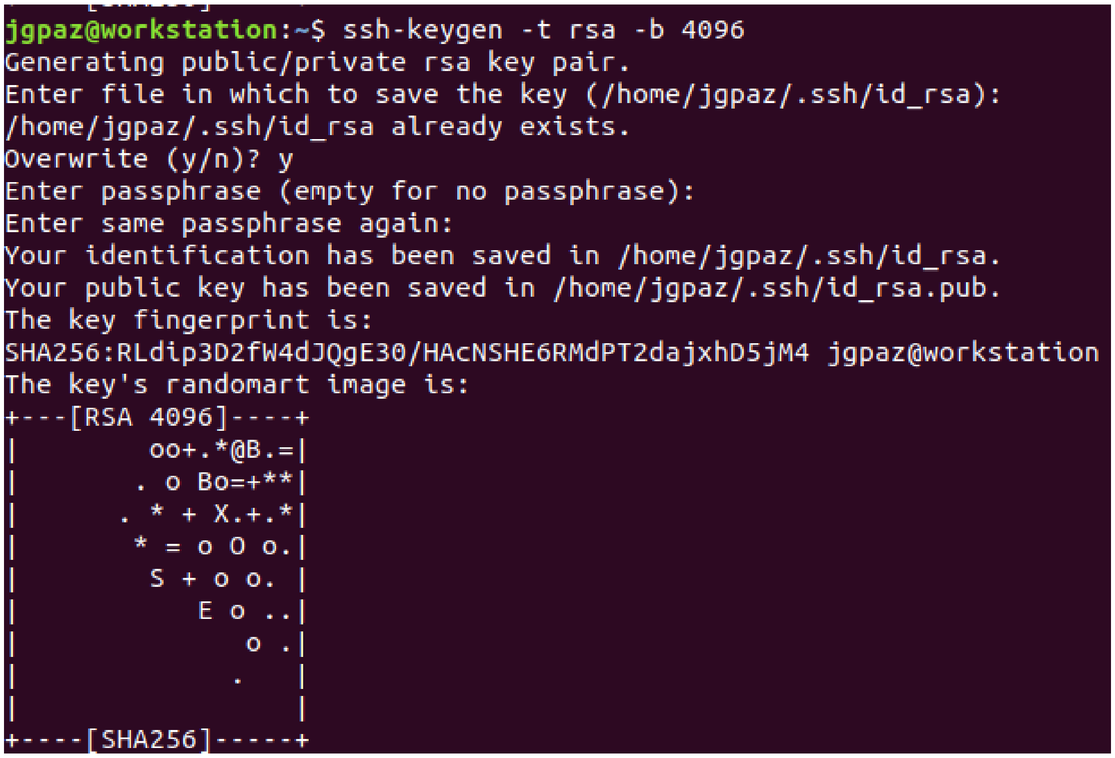
1. The simplest way to generate a key pair is to run *ssh-keygen* without arguments. In this case, it will prompt for the file in which to store keys. First,

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*.



```
jgpaz@workstation:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/jgpaz/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/jgpaz/.ssh/id_rsa.
Your public key has been saved in /home/jgpaz/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:PmKBhDDFmQ8Fm+h2DI9dVFgRIpKae2N3BzSFaVLA1Oc jgpaz@workstation
The key's randomart image is:
+---[RSA 2048]----+
|o++O+*=O+        |
| =*+=.O .        |
|oo+o.= +         |
|+ *.o.. E        |
| = =. ..S        |
|o = . .o.        |
| o o .o.o        |
|     . . .       |
|                 |
+----[SHA256]-----+
jgpaz@workstation:~$
```

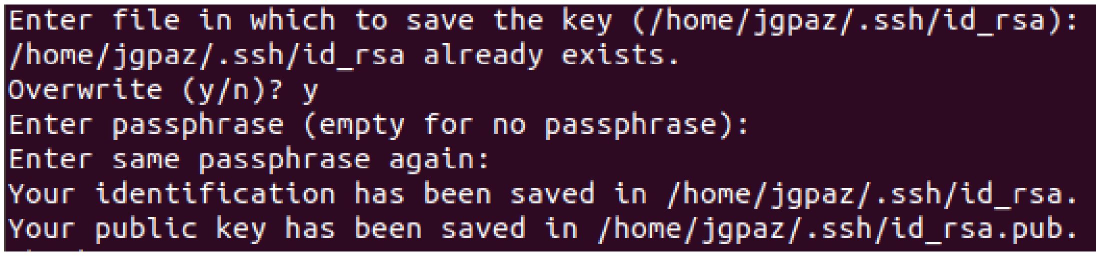
2. Issue the command *ssh-keygen -t rsa -b 4096.* The algorithm is selected using the -t option and key size using the -b option.



```
jgpaz@workstation:~$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/jgpaz/.ssh/id_rsa):
/home/jgpaz/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/jgpaz/.ssh/id_rsa.
Your public key has been saved in /home/jgpaz/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:RLdip3D2fW4dJQgE30/HAcNSHE6RMdPT2dajxhD5jM4 jgpaz@workstation
The key's randomart image is:
+---[RSA 4096]----+
|        oo+.*@B.=|
|       . o Bo=+**|
|      . * + X.+.*|
|       * = o O o.|
|        S + o o. |
|           E o ..|
|              o .|
|             .   |
|                 |
+----[SHA256]-----+
```

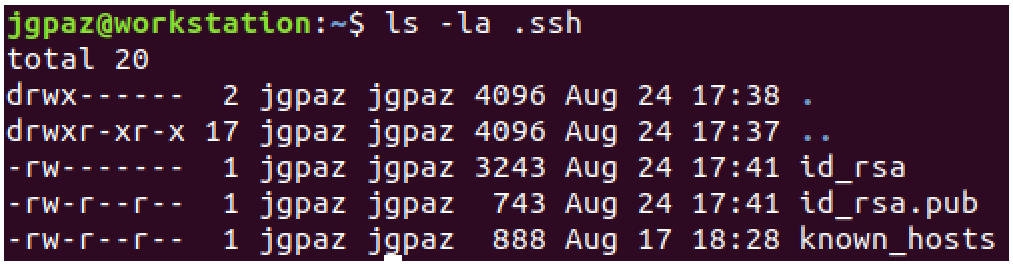
3. When asked for a passphrase, just press enter. The passphrase is used for encrypting the key, so that it cannot be used even if someone obtains the

private key file. The passphrase should be cryptographically strong.



```
Enter file in which to save the key (/home/jgpaz/.ssh/id_rsa):
/home/jgpaz/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/jgpaz/.ssh/id_rsa.
Your public key has been saved in /home/jgpaz/.ssh/id_rsa.pub.
```

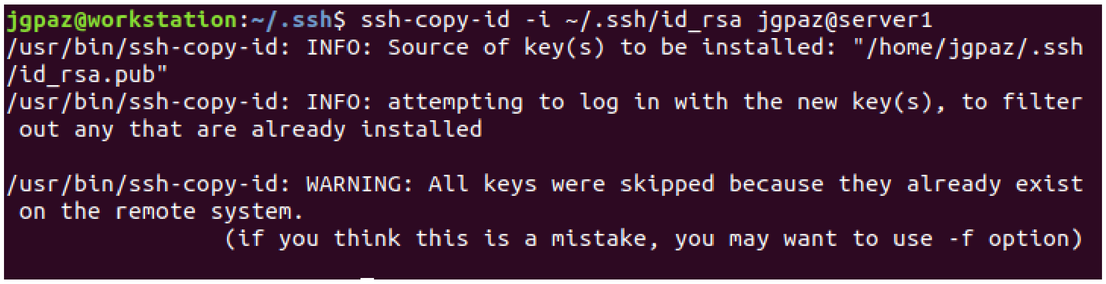
4. Verify that you have created the key by issuing the command *ls -la .ssh.* The command should show the .ssh directory containing a pair of keys. For example, id_rsa.pub and id_rsa.



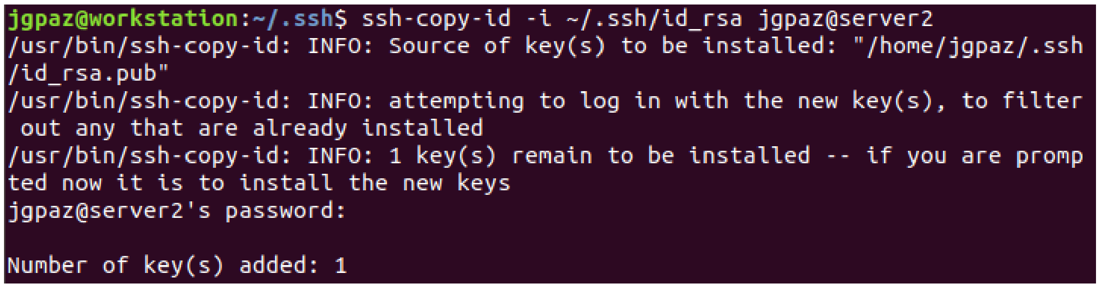
```
jgpaz@workstation:~$ ls -la .ssh
total 20
drwx------  2 jgpaz jgpaz 4096 Aug 24 17:38 .
drwxr-xr-x 17 jgpaz jgpaz 4096 Aug 24 17:37 ..
-rw-------  1 jgpaz jgpaz 3243 Aug 24 17:41 id_rsa
-rw-r--r--  1 jgpaz jgpaz  743 Aug 24 17:41 id_rsa.pub
-rw-r--r--  1 jgpaz jgpaz  888 Aug 17 18:28 known_hosts
```

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*



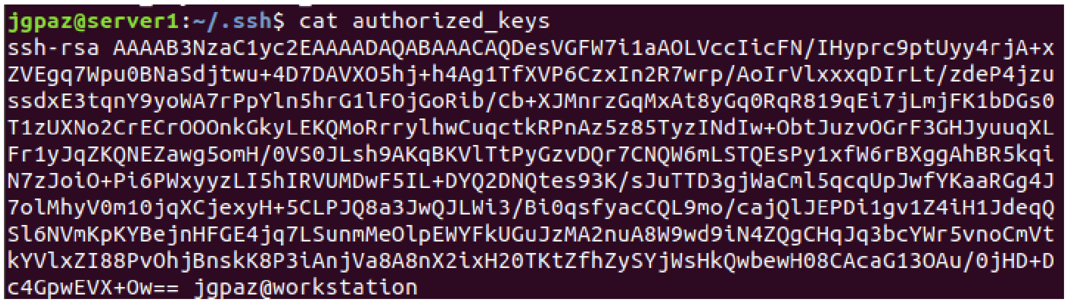
```
jgpaz@workstation:~/.ssh$ ssh-copy-id -i ~/.ssh/id_rsa jgpaz@server1
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/jgpaz/.ssh
/id_rsa.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter
 out any that are already installed

/usr/bin/ssh-copy-id: WARNING: All keys were skipped because they already exist
 on the remote system.
                (if you think this is a mistake, you may want to use -f option)
```



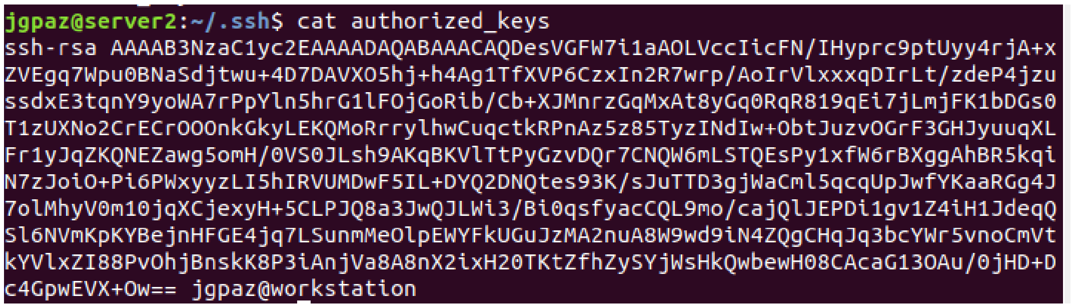
```
jgpaz@workstation:~/.ssh$ ssh-copy-id -i ~/.ssh/id_rsa jgpaz@server2
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/jgpaz/.ssh
/id_rsa.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter
 out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are promp
ted now it is to install the new keys
jgpaz@server2's password:

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

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?



```
jgpaz@server1:~/.ssh$ cat authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAACAQDesVGFW7i1aAOLVccIicFN/IHyprc9ptUyy4rjA+x
ZVEgq7Wpu0BNaSdjtwu+4D7DAVXO5hj+h4Ag1TfXVP6CzxIn2R7wrp/AoIrVlxxxqDIrLt/zdeP4jzu
ssdxE3tqnY9yoWA7rPpYln5hrG1lFOjGoRib/Cb+XJMnrzGqMxAt8yGq0RqR819qEi7jLmjFK1bDGs0
T1zUXNo2CrECrOOOnkGkyLEKQMoRrrylhwCuqctkRPnAz5z85TyzINdIw+ObtJuzvOGrF3GHJyuuqXL
Fr1yJqZKQNEZawg5omH/0VS0JLsh9AKqBKVlTtPyGzvDQr7CNQW6mLSTQEsPy1xfW6rBXggAhBR5kqi
N7zJoiO+Pi6PWxyyzLI5hIRVUMDwF5IL+DYQ2DNQtes93K/sJuTTD3gjWaCml5qcqUpJwfYKaaRGg4J
7olMhyV0m10jqXCjexyH+5CLPJQ8a3JwQJLWi3/Bi0qsfyacCQL9mo/cajQlJEPDi1gv1Z4iH1JdeqQ
Sl6NVmKpKYBejnHFGE4jq7LSunmMeOlpEWYFkUGuJzMA2nuA8W9wd9iN4ZQgCHqJq3bcYWr5vnoCmVt
kYVlxZI88PvOhjBnskK8P3iAnjVa8A8nX2ixH20TKtZfhZySYjWsHkQwbewH08CAcaG13OAu/0jHD+D
c4GpwEVX+Ow== jgpaz@workstation
```

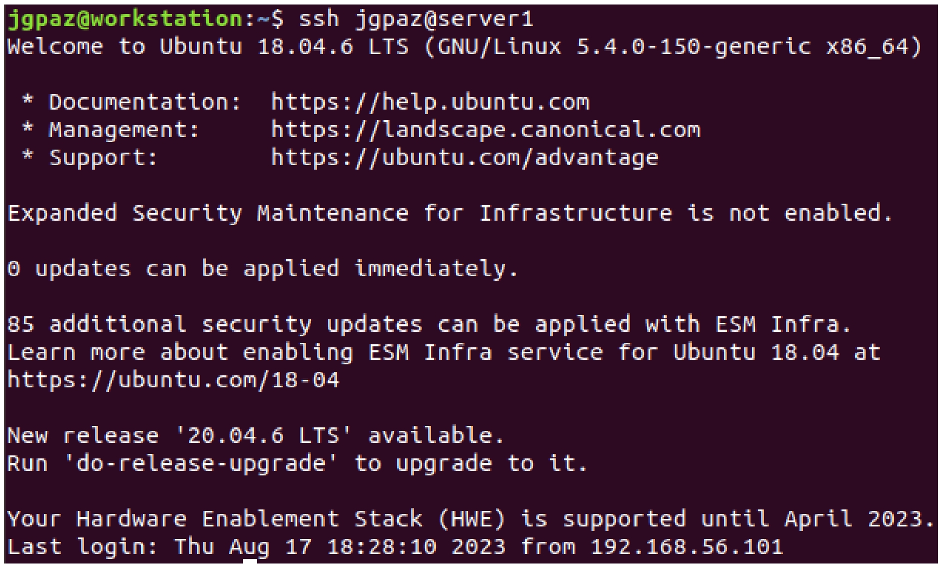


```
jgpaz@server2:~/.ssh$ cat authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAACAQDesVGFW7i1aAOLVccIicFN/IHyprc9ptUyy4rjA+x
ZVEgq7Wpu0BNaSdjtwu+4D7DAVXO5hj+h4Ag1TfXVP6CzxIn2R7wrp/AoIrVlxxxqDIrLt/zdeP4jzu
ssdxE3tqnY9yoWA7rPpYln5hrG1lFOjGoRib/Cb+XJMnrzGqMxAt8yGq0RqR819qEi7jLmjFK1bDGs0
T1zUXNo2CrECrOOOnkGkyLEKQMoRrrylhwCuqctkRPnAz5z85TyzINdIw+ObtJuzvOGrF3GHJyuuqXL
Fr1yJqZKQNEZawg5omH/0VS0JLsh9AKqBKVlTtPyGzvDQr7CNQW6mLSTQEsPy1xfW6rBXggAhBR5kqi
N7zJoiO+Pi6PWxyyzLI5hIRVUMDwF5IL+DYQ2DNQtes93K/sJuTTD3gjWaCml5qcqUpJwfYKaaRGg4J
7olMhyV0m10jqXCjexyH+5CLPJQ8a3JwQJLWi3/Bi0qsfyacCQL9mo/cajQlJEPDi1gv1Z4iH1JdeqQ
Sl6NVmKpKYBejnHFGE4jq7LSunmMeOlpEWYFkUGuJzMA2nuA8W9wd9iN4ZQgCHqJq3bcYWr5vnoCmVt
kYVlxZI88PvOhjBnskK8P3iAnjVa8A8nX2ixH20TKtZfhZySYjWsHkQwbewH08CAcaG13OAu/0jHD+D
c4GpwEVX+Ow== jgpaz@workstation
```

Reflections:

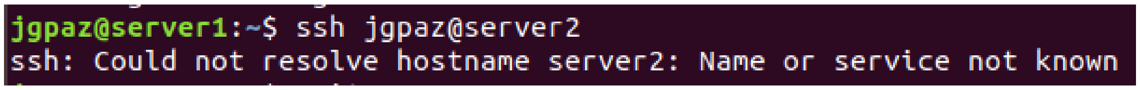
Answer the following:

1. How will you describe the ssh-program? What does it do?
2. How do you know that you already installed the public key to the remote servers?



```
jgpaz@workstation:~$ ssh jgpaz@server1
Welcome to Ubuntu 18.04.6 LTS (GNU/Linux 5.4.0-150-generic x86_64)

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

Expanded Security Maintenance for Infrastructure is not enabled.

0 updates can be applied immediately.

85 additional security updates can be applied with ESM Infra.
Learn more about enabling ESM Infra service for Ubuntu 18.04 at
https://ubuntu.com/18-04

New release '20.04.6 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Your Hardware Enablement Stack (HWE) is supported until April 2023.
Last login: Thu Aug 17 18:28:10 2023 from 192.168.56.101
```

- I generate the public key in workstation to connect to server 1.



```
jgpaz@server1:~$ ssh jgpaz@server2
ssh: Could not resolve hostname server2: Name or service not known
```

- I did not install the public key in server 2 that's why it is service error.

Part 2: Discussion

Provide screenshots for each task.

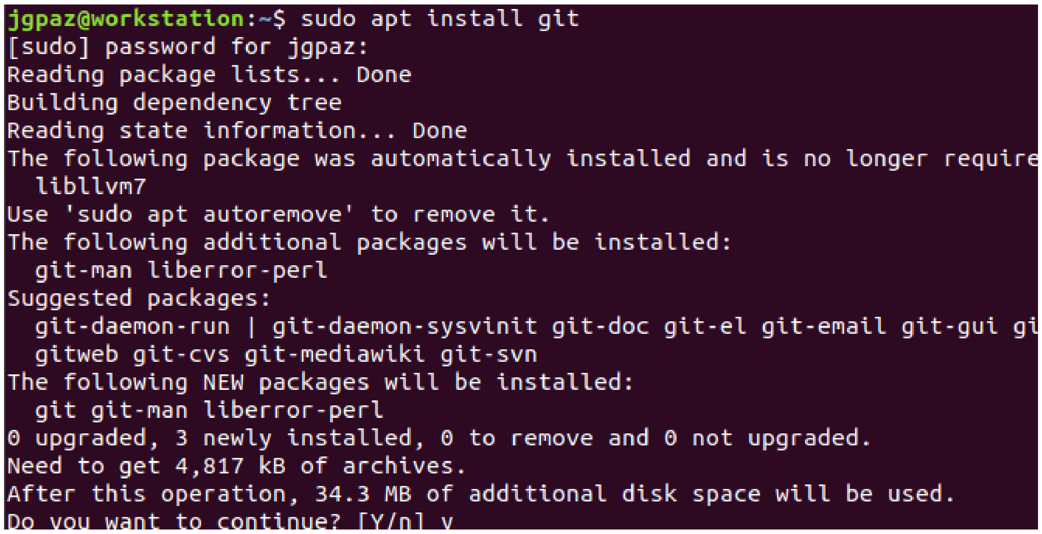
It is assumed that you are done with the last activity (**Activity 2: SSH Key-Based Authentication**).

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

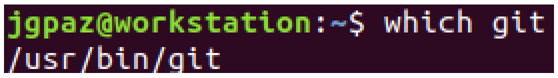
Task 3: Set up the Git Repository

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*



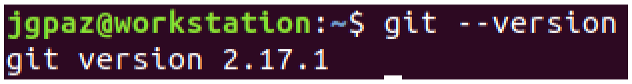
```
jgpaz@workstation:~$ sudo apt install git
[sudo] password for jgpaz:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following package was automatically installed and is no longer require
  libllvm7
Use 'sudo apt autoremove' to remove it.
The following 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 gi
  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
```

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



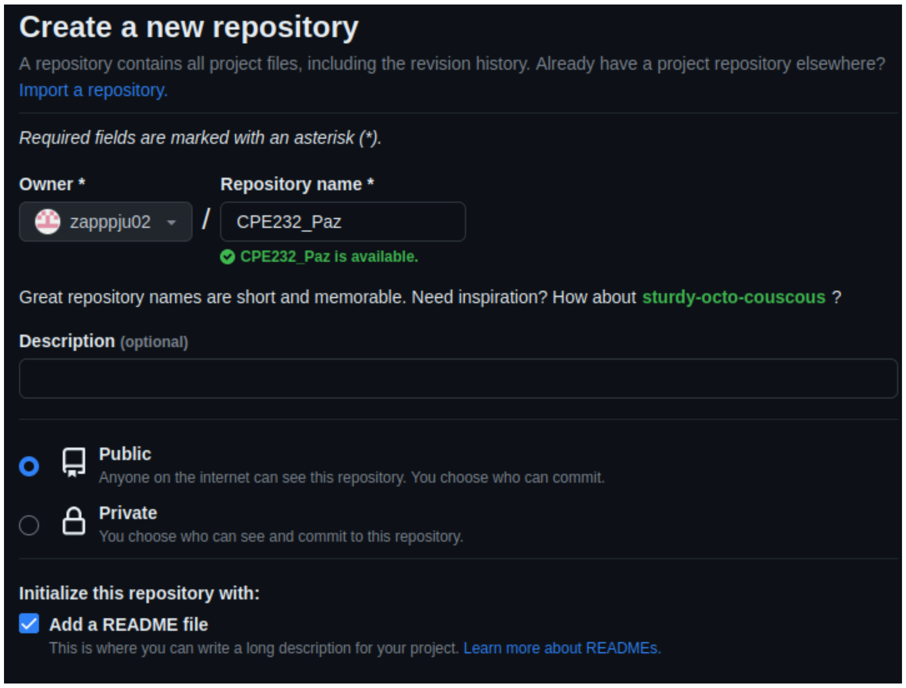
```
jgpaz@workstation:~$ which git
/usr/bin/git
```

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



```
jgpaz@workstation:~$ git --version
git version 2.17.1
```

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.

Required fields are marked with an asterisk (*).

Owner *: zapppju02 / Repository name *: CPE232_Paz

CPE232_Paz is available.

Great repository names are short and memorable. Need inspiration? How about sturdy-octo-couscous ?

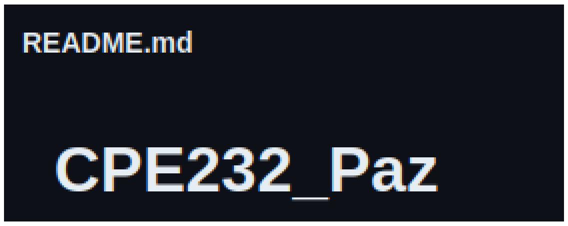
Description (optional)

● Public — Anyone on the internet can see this repository. You choose who commits.

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

Initialize this repository with:

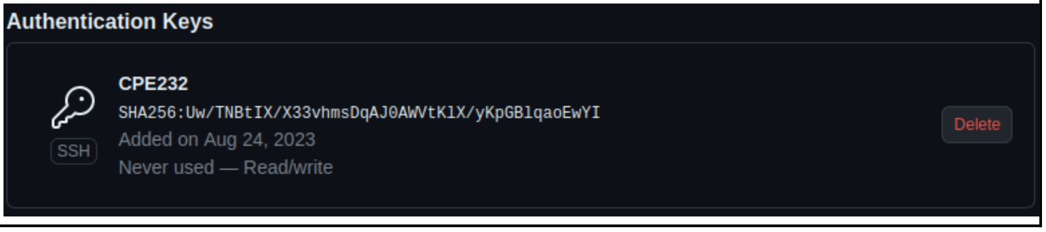
✓ Add a README file — This is where you can write a long description for your project. Learn more about READMEs.



README.md

CPE232_Paz

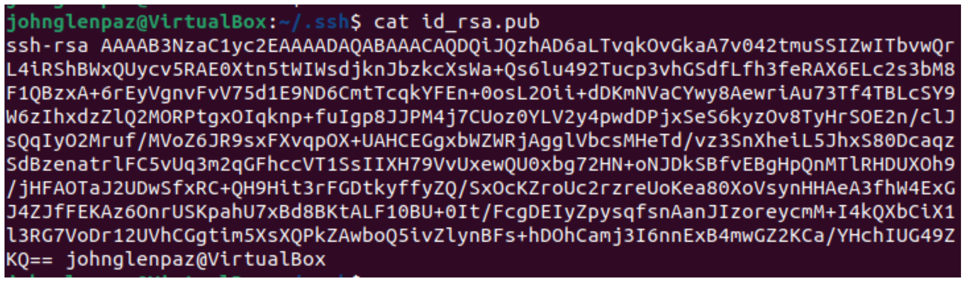
   b. Create a new SSH key on GitHub. Go your profile's setting and click SSH and GPG keys. If there is an existing key, make sure to delete it. To create a new SSH keys, click New SSH Key. Write CPE232 key as the title of the key.



Authentication Keys

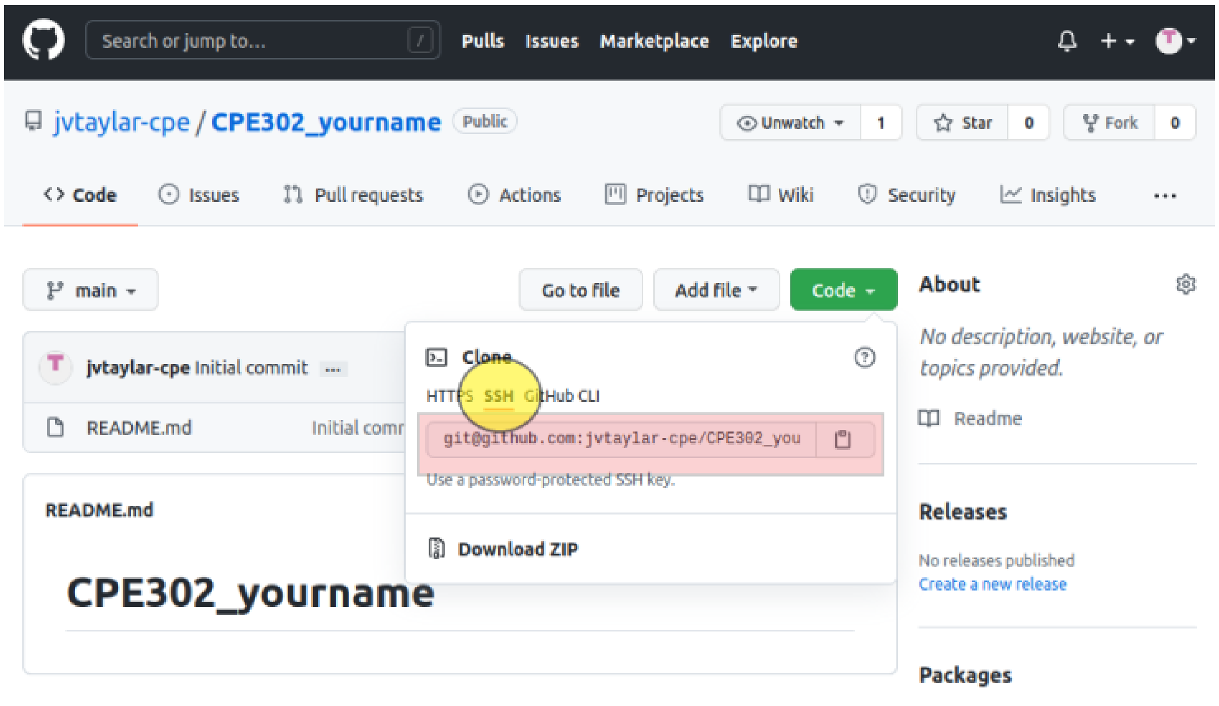
CPE232
SHA256:Uw/TNBtIX/X33vhmsDqAJ0AWVtKlX/yKpGBlqaoEwYI
Added on Aug 24, 2023
Never used — Read/write

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.



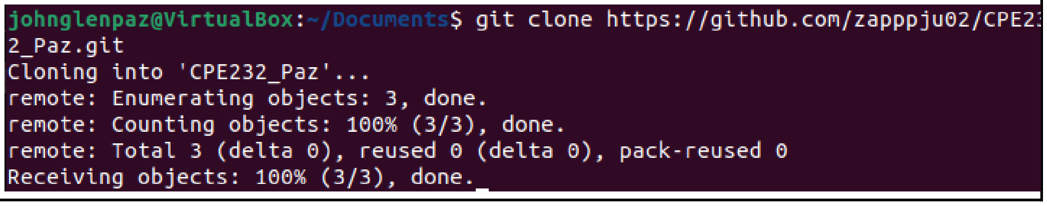
```
johnglenpaz@VirtualBox:~/.ssh$ cat id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAACAQDQiJQzhAD6aLTvqkOvGkaA7v042tmuSSIZwITbvwQr
L4iRShBWxQUycv5RAE0Xtn5tWIWsdjknJbzkcXsWa+Qs6lu492Tucp3vhGSdfLfh3feRAX6ELc2s3bM8
F1QBzxA+6rEyVgnvFvV75d1E9ND6CmtTcqkYFEn+0osL2Oii+dDKmNVaCYwy8AewriAu73Tf4TBLcSY9
W6zIhxdzZlQ2MORPtgxOIqknp+fuIgp8JJPM4j7CUoz0YLV2y4pwdDPjxSeS6kyzOv8TyHrSOE2n/clJ
sQqIyO2Mruf/MVoZ6JR9sxFXvqpOX+UAHCEGgxbWZWRjAgglVbcsMHeTd/vz3SnXheiL5JhxS80Dcaqz
SdBzenatrlFC5vUq3m2qGFhccVT1SsIIXH79VvUxewQU0xbg72HN+oNJDkSBfvEBgHpQnMTlRHDUXOh9
/jHFAOTaJ2UDwSfxRC+QH9Hit3rFGDtkyffyZQ/SxOcKZroUc2rzreUoKea80XoVsynHHAeA3fhW4ExG
J4ZJfFEKAz6OnrUSKpahU7xBd8BKtALF10BU+0It/FcgDEIyZpysqfsnAanJIzoreycmM+I4kQXbCiX1
l3RG7VoDr12UVhCGgtim5XsXQPkZAwboQ5ivZlynBFs+hDOhCamj3I6nnExB4mwGZ2KCa/YHchIUG49Z
KQ== johnglenpaz@VirtualBox
```

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.



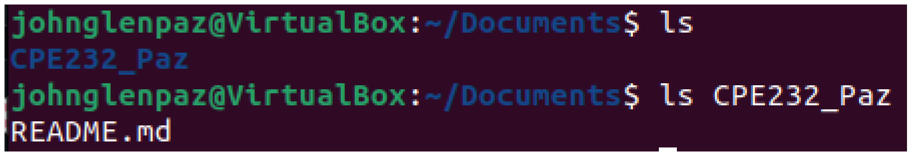
```
git@github.com:jvtaylar-cpe/CPE302_you
```

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.



```
johnglenpaz@VirtualBox:~/Documents$ git clone https://github.com/zapppju02/CPE23
2_Paz.git
Cloning into 'CPE232_Paz'...
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.
```

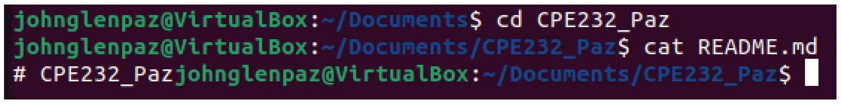
f. To verify that you have cloned the GitHub repository, issue the command *ls*. Observe that you have the CPE232_yourname in the list of your directories. Use CD command to go to that directory and LS command to see the file README.md.



```
johnglenpaz@VirtualBox:~/Documents$ ls
CPE232_Paz
johnglenpaz@VirtualBox:~/Documents$ ls CPE232_Paz
README.md
```

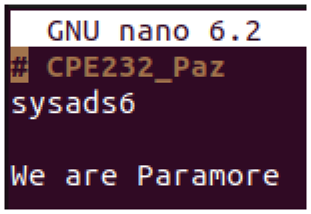
g. Use the following commands to personalize your git.

- *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*



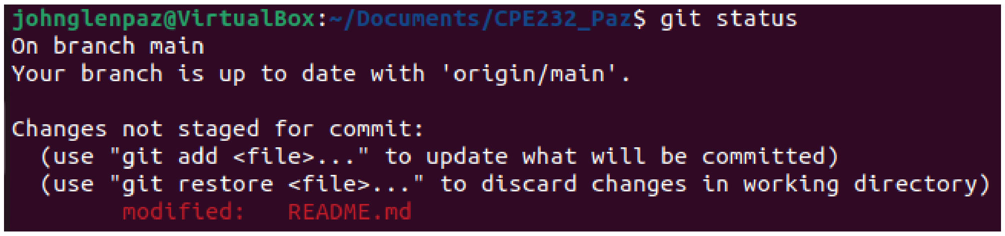
```
johnglenpaz@VirtualBox:~/Documents$ cd CPE232_Paz
johnglenpaz@VirtualBox:~/Documents/CPE232_Paz$ cat README.md
# CPE232_Pazjohnglenpaz@VirtualBox:~/Documents/CPE232_Paz$ 
```

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.



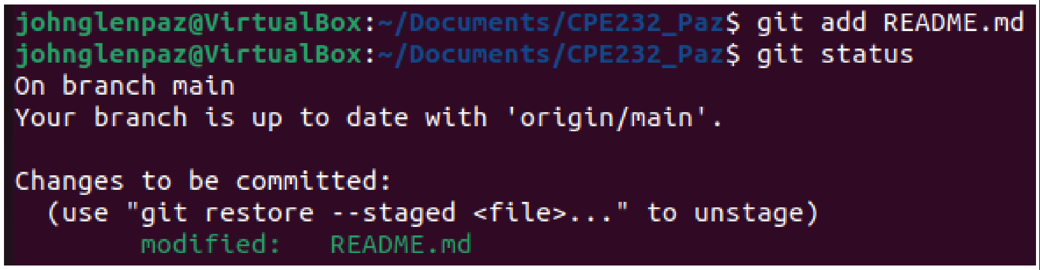
```
  GNU nano 6.2
# CPE232_Paz
sysads6

We are Paramore
```

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?



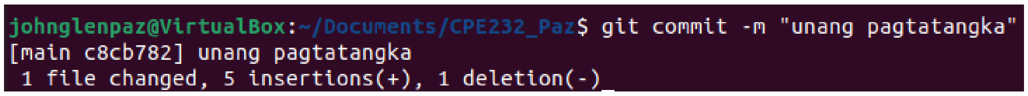
```
johnglenpaz@VirtualBox:~/Documents/CPE232_Paz$ git status
On branch main
Your branch is up to date with 'origin/main'.

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
```

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



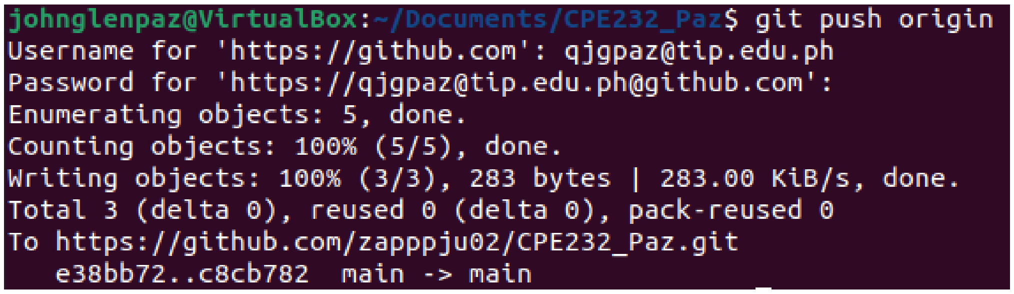
```
johnglenpaz@VirtualBox:~/Documents/CPE232_Paz$ git add README.md
johnglenpaz@VirtualBox:~/Documents/CPE232_Paz$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   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.



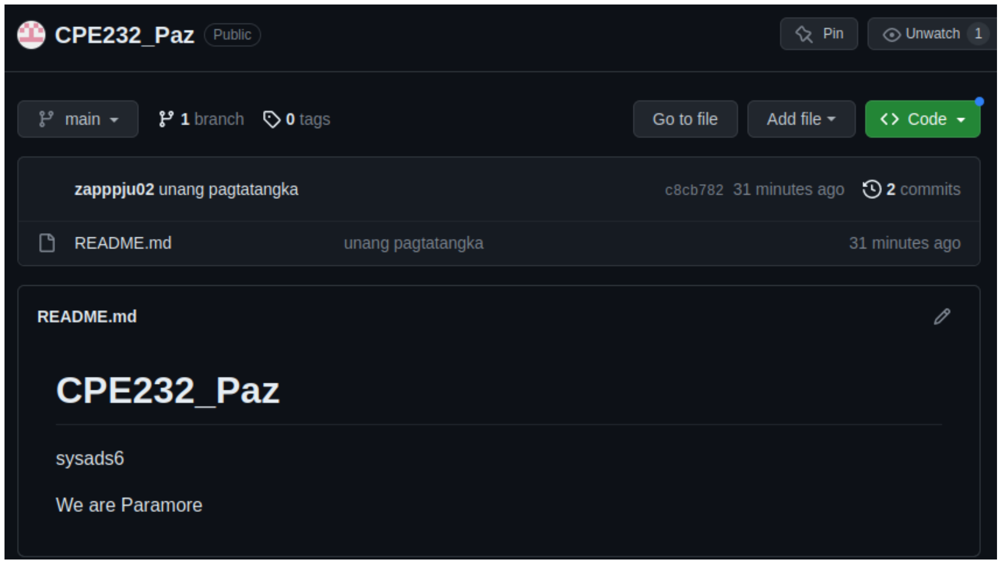
```
johnglenpaz@VirtualBox:~/Documents/CPE232_Paz$ git commit -m "unang pagtatangka"
[main c8cb782] unang pagtatangka
 1 file changed, 5 insertions(+), 1 deletion(-)
```

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.*



```
johnglenpaz@VirtualBox:~/Documents/CPE232_Paz$ git push origin
Username for 'https://github.com': qjgpaz@tip.edu.ph
Password for 'https://qjgpaz@tip.edu.ph@github.com':
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Writing objects: 100% (3/3), 283 bytes | 283.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/zapppju02/CPE232_Paz.git
   e38bb72..c8cb782  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_Paz Public

zapppju02 unang pagtatangka c8cb782 31 minutes ago 2 commits

README.md unang pagtatangka 31 minutes ago

README.md

CPE232_Paz

sysads6

We are Paramore

Reflections:

Answer the following:

3. What sort of things have we so far done to the remote servers using ansible commands?
   - The consistent application of configurations made possible by Ansible's playbooks has allowed for effective scaling and maintenance. Ansible's remote server management capabilities may have also simplified duties like user management, service orchestration, and security hardening.

4. How important is the inventory file?

Conclusions/Learnings:

In this activity, I learned a lot starting from creating a ssh public and private key to have access to other servers and to control them remotely. Next is setting up git repositories and remote repositories, the process is slightly difficult for me because I did the process again when I got home and I was confused because the instruction in the given document is not updated but luckily I remembered the process that our prof. told us before when I was still in school that's why I finished the task within the day. Overall doing this activity is very fun and challenging, and I hope the next activity will be the same.