Name: Ferrer, Joseph Bryan M.	**Date Performed: 08/24/2023**
Course/Section: CPE31S6	**Date Submitted: 08/24/2023**
Instructor: Engr. Jonathan V. Taylar	**Semester and SY: 1st Semester / 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.

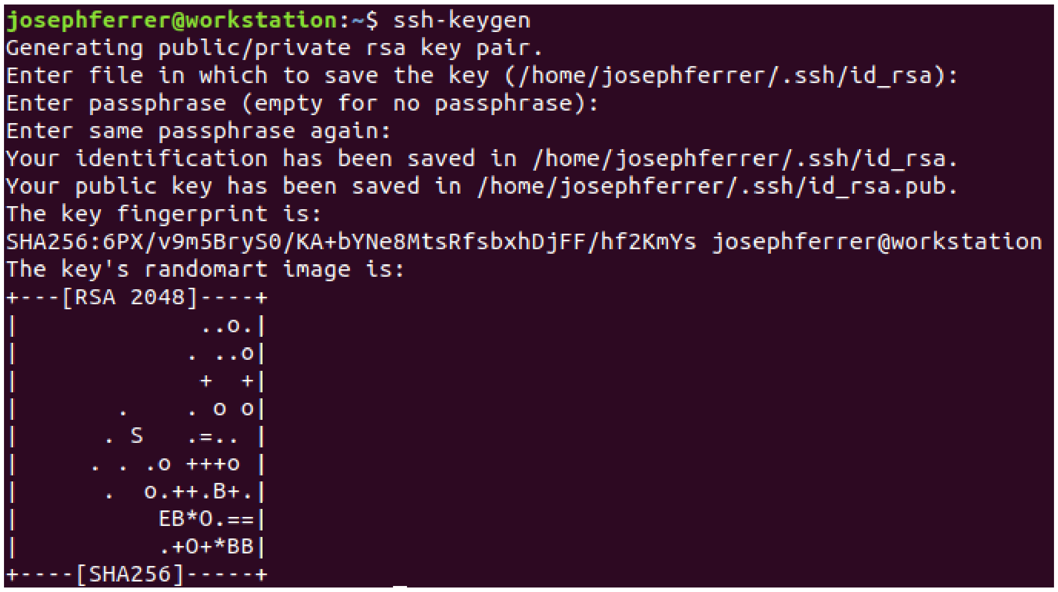
SSH introduced public key authentication as a more secure alternative to the older .rhosts authentication. It improved security by avoiding the need to have password stored in files and eliminated the possibility of a compromised server stealing the user's password.

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

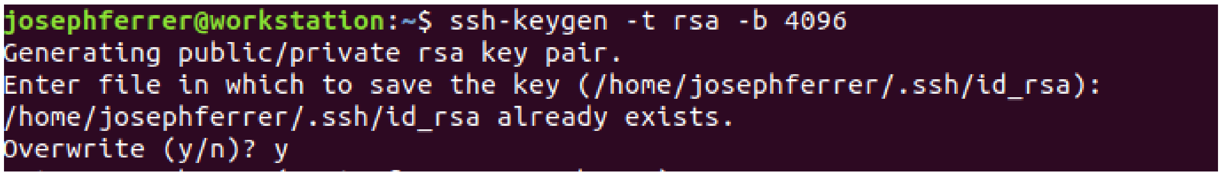
1. The simplest way to generate a key pair is to run *ssh-keygen* without arguments. In this case, it will prompt for the file in which to store keys. First, the tool asked where to save the file. SSH keys for user authentication are usually stored in the users .ssh directory under the home directory.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



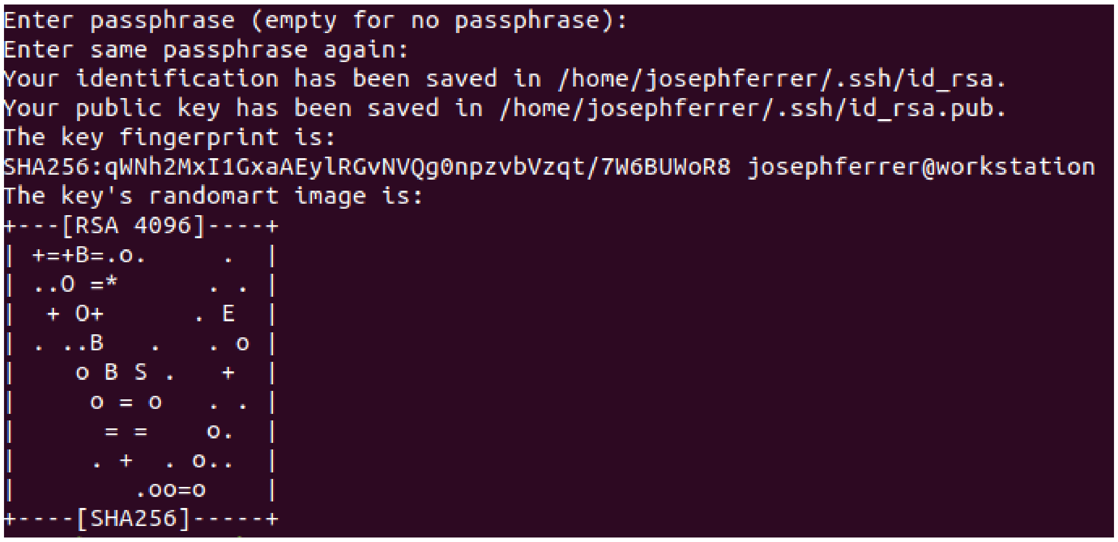
```
josephferrer@workstation:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/josephferrer/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/josephferrer/.ssh/id_rsa.
Your public key has been saved in /home/josephferrer/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:6PX/v9m5BryS0/KA+bYNe8MtsRfsbxhDjFF/hf2KmYs josephferrer@workstation
The key's randomart image is:
+---[RSA 2048]----+
|             ..o.|
|            . ..o|
|             +  +|
|       .    . o o|
|      . S   .=.. |
|     . . .o +++o |
|      .  o.++.B+.|
|          EB*O.==|
|          .+O+*BB|
+----[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.



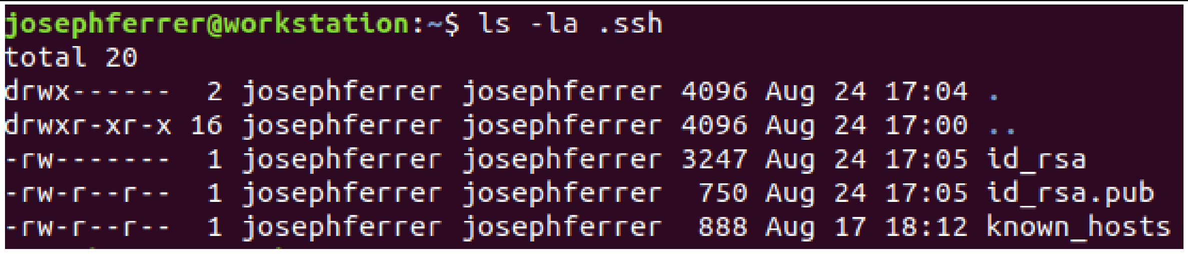
```
josephferrer@workstation:~$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/josephferrer/.ssh/id_rsa):
/home/josephferrer/.ssh/id_rsa already exists.
Overwrite (y/n)? y
```

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 passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/josephferrer/.ssh/id_rsa.
Your public key has been saved in /home/josephferrer/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:qWNh2MxI1GxaAEylRGvNVQg0npzvbVzqt/7W6BUWoR8 josephferrer@workstation
The key's randomart image is:
+---[RSA 4096]----+
| +=+B=.o.     .  |
| ..O =*      . . |
|  + O+      . E  |
| . ..B   .   . o |
|    o B S .   +  |
|     o = o   . . |
|      = =    o.  |
|     . +  . o..  |
|        .oo=o    |
+----[SHA256]-----+
```

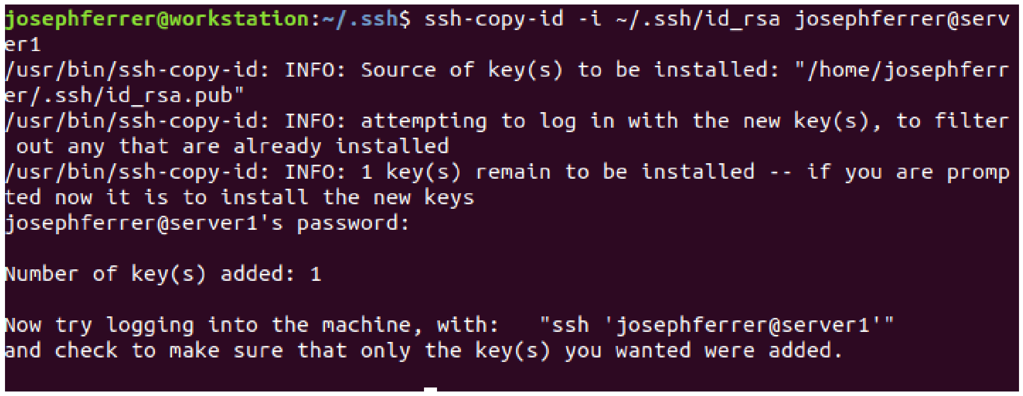
4. Verify that you have created the key by issuing the command *ls -la .ssh.* The command should show the .ssh directory containing a pair of keys. For example, id_rsa.pub and id_rsa.



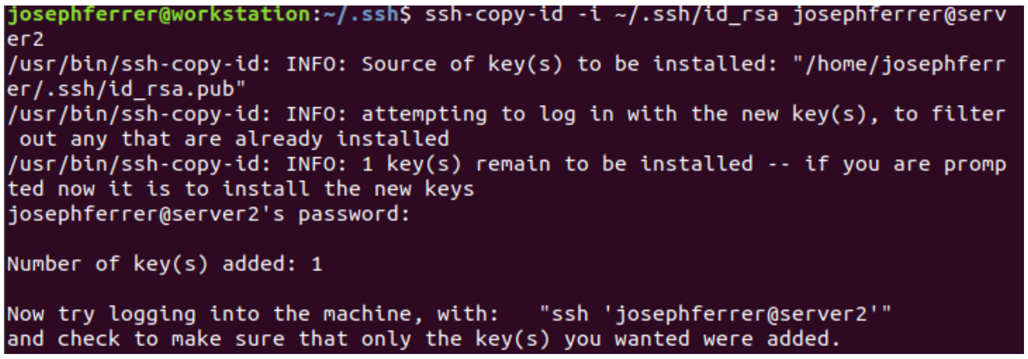
```
josephferrer@workstation:~$ ls -la .ssh
total 20
drwx------  2 josephferrer josephferrer 4096 Aug 24 17:04 .
drwxr-xr-x 16 josephferrer josephferrer 4096 Aug 24 17:00 ..
-rw-------  1 josephferrer josephferrer 3247 Aug 24 17:05 id_rsa
-rw-r--r--  1 josephferrer josephferrer  750 Aug 24 17:05 id_rsa.pub
-rw-r--r--  1 josephferrer josephferrer  888 Aug 17 18:12 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*

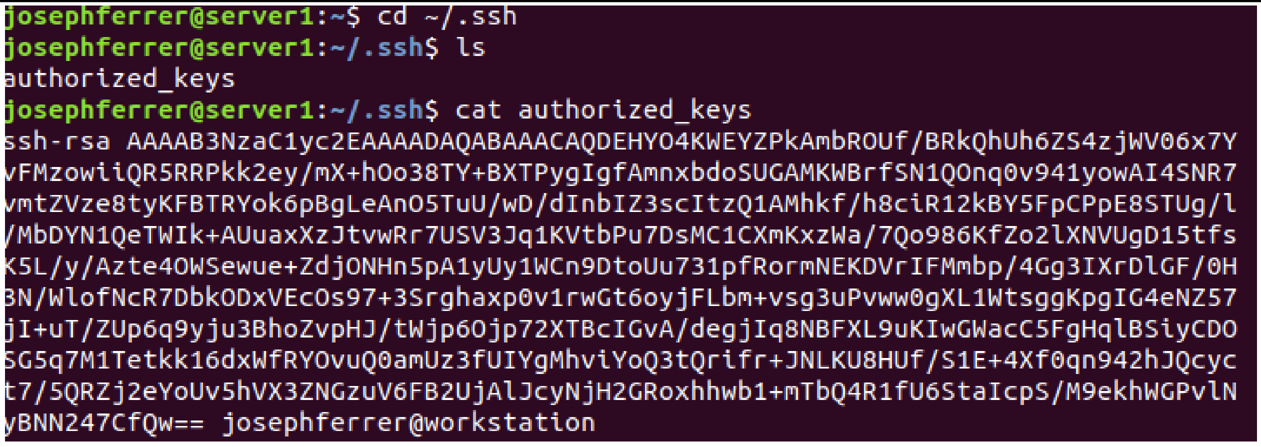


```
josephferrer@workstation:~/.ssh$ ssh-copy-id -i ~/.ssh/id_rsa josephferrer@serv
er1
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/josephferr
er/.ssh/id_rsa.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter
 out any that are already installed
/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
josephferrer@server1's password:

Number of key(s) added: 1

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

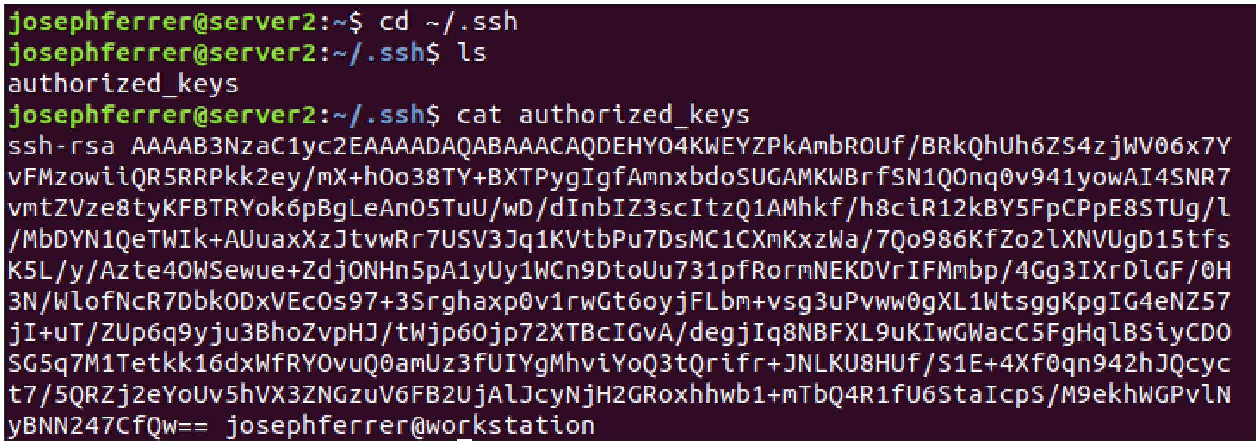


```
josephferrer@workstation:~/.ssh$ ssh-copy-id -i ~/.ssh/id_rsa josephferrer@serv
er2
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/josephferr
er/.ssh/id_rsa.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter
 out any that are already installed
/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
josephferrer@server2's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'josephferrer@server2'"
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.

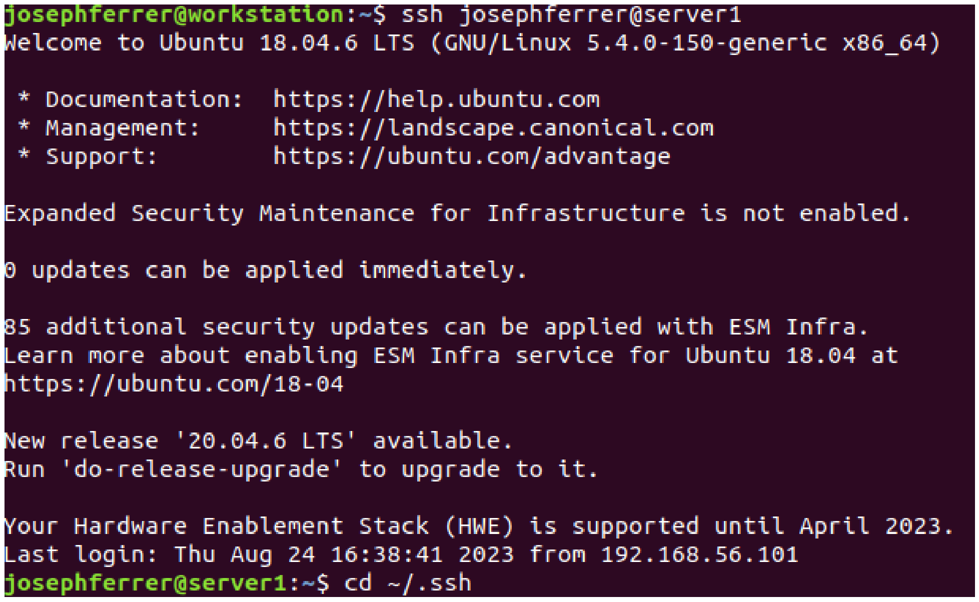


```
josephferrer@server1:~$ cd ~/.ssh
josephferrer@server1:~/.ssh$ ls
authorized_keys
josephferrer@server1:~/.ssh$ cat authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAACAQDEHYO4KWEYZPkAmbROUf/BRkQhUh6ZS4zjWV06x7Y
vFMzowiiQR5RRPkk2ey/mX+hOo38TY+BXTPygIgfAmnxbdoSUGAMKWBrfSN1QOnq0v941yowAI4SNR7
vmtZVze8tyKFBTRYok6pBgLeAnO5TuU/wD/dInbIZ3scItzQ1AMhkf/h8ciR12kBY5FpCPpE8STUg/l
/MbDYN1QeTWIk+AUuaxXzJtvwRr7USV3Jq1KVtbPu7DsMC1CXmKxzWa/7Qo986KfZo2lXNVUgD15tfs
K5L/y/Azte4OWSewue+ZdjONHn5pA1yUy1WCn9DtoUu731pfRormNEKDVrIFMmbp/4Gg3IXrDlGF/0H
3N/WlofNcR7DbkODxVEcOs97+3Srghaxp0v1rwGt6oyjFLbm+vsg3uPvww0gXL1WtsggKpgIG4eNZ57
jI+uT/ZUp6q9yju3BhoZvpHJ/tWjp6Ojp72XTBcIGvA/degjIq8NBFXL9uKIwGWacC5FgHqlBSiyCDO
SG5q7M1Tetkk16dxWfRYOvuQ0amUz3fUIYgMhviYoQ3tQrifr+JNLKU8HUf/S1E+4Xf0qn942hJQcyc
t7/5QRZj2eYoUv5hVX3ZNGzuV6FB2UjAlJcyNjH2GRoxhhwb1+mTbQ4R1fU6StaIcpS/M9ekhWGPvlN
yBNN247CfQw== josephferrer@workstation
```

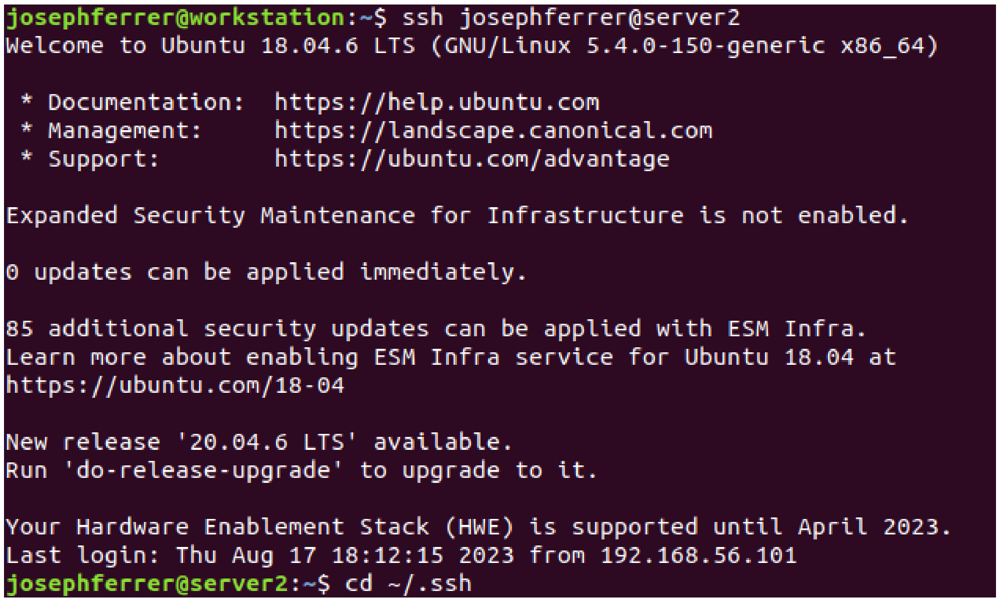


```
josephferrer@server2:~$ cd ~/.ssh
josephferrer@server2:~/.ssh$ ls
authorized_keys
josephferrer@server2:~/.ssh$ cat authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAACAQDEHYO4KWEYZPkAmbROUf/BRkQhUh6ZS4zjWV06x7Y
vFMzowiiQR5RRPkk2ey/mX+hOo38TY+BXTPygIgfAmnxbdoSUGAMKWBrfSN1QOnq0v941yowAI4SNR7
vmtZVze8tyKFBTRYok6pBgLeAnO5TuU/wD/dInbIZ3scItzQ1AMhkf/h8ciR12kBY5FpCPpE8STUg/l
/MbDYN1QeTWIk+AUuaxXzJtvwRr7USV3Jq1KVtbPu7DsMC1CXmKxzWa/7Qo986KfZo2lXNVUgD15tfs
K5L/y/Azte4OWSewue+ZdjONHn5pA1yUy1WCn9DtoUu731pfRormNEKDVrIFMmbp/4Gg3IXrDlGF/0H
3N/WlofNcR7DbkODxVEcOs97+3Srghaxp0v1rwGt6oyjFLbm+vsg3uPvww0gXL1WtsggKpgIG4eNZ57
jI+uT/ZUp6q9yju3BhoZvpHJ/tWjp6Ojp72XTBcIGvA/degjIq8NBFXL9uKIwGWacC5FgHqlBSiyCDO
SG5q7M1Tetkk16dxWfRYOvuQ0amUz3fUIYgMhviYoQ3tQrifr+JNLKU8HUf/S1E+4Xf0qn942hJQcyc
t7/5QRZj2eYoUv5hVX3ZNGzuV6FB2UjAlJcyNjH2GRoxhhwb1+mTbQ4R1fU6StaIcpS/M9ekhWGPvlN
yBNN247CfQw== josephferrer@workstation
```

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?



```
josephferrer@workstation:~$ ssh josephferrer@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 24 16:38:41 2023 from 192.168.56.101
josephferrer@server1:~$ cd ~/.ssh
```



```
josephferrer@workstation:~$ ssh josephferrer@server2
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 18:12:15 2023 from 192.168.56.101
josephferrer@server2:~$ cd ~/.ssh
```

When using SSH with Server 1 and Server 2, I noticed that it didn't ask for a password.

Reflections:
Answer the following:

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

   SSH is a versatile and essential tool for securely accessing and managing remote systems. It provides a layer of encryption and authentication that makes it possible to manage servers and perform tasks on remote computers without exposing sensitive information to potential security risks on the open network.

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

   You will know that you already installed the public key to the remote servers if your public key is present in the authorized_keys file. The authorized_keys file is where you should see the list of public keys that are allowed to connect to the server using key-based authentication. If your public key is present in the authorized_keys file, you have successfully installed your key on the remote server.

Part 2: Discussion

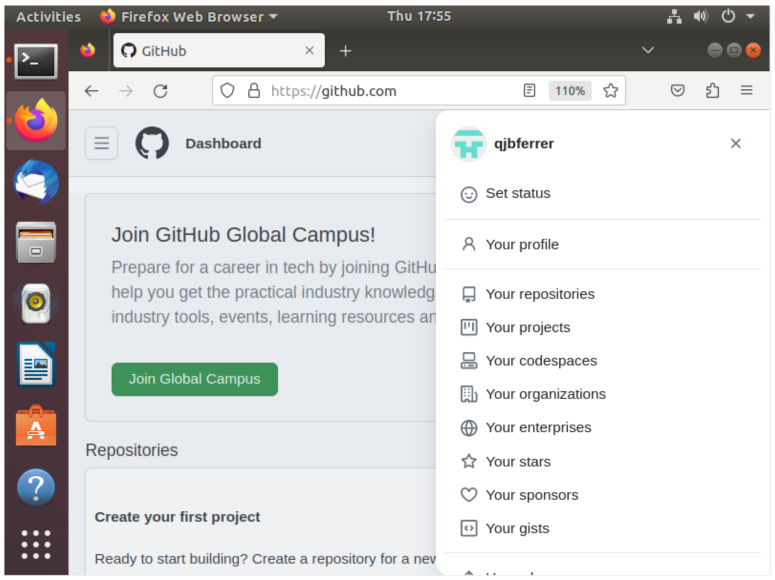
Provide screenshots for each task.

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

Set up Git

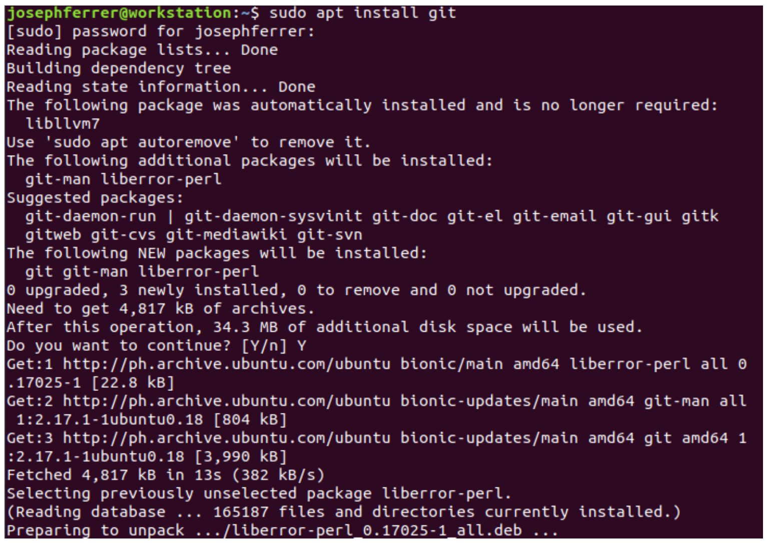
At the heart of GitHub is an open-source version control system (VCS) called Git. Git is responsible for everything GitHub-related that happens locally on your computer. To use Git on the command line, you'll need to download, install, and configure Git on your computer. You can also install GitHub CLI 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



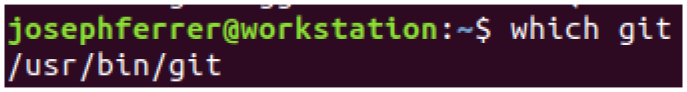
Task 3: Set up the Git Repository

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*



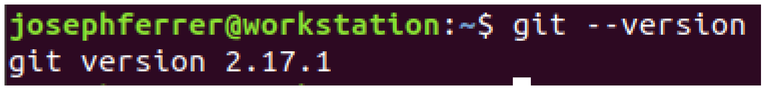
```
josephferrer@workstation:~$ sudo apt install git
[sudo] password for josephferrer:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following package was automatically installed and is no longer required:
  libllvm7
Use 'sudo apt autoremove' to remove it.
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 13s (382 kB/s)
Selecting previously unselected package liberror-perl.
(Reading database ... 165187 files and directories currently installed.)
Preparing to unpack .../liberror-perl_0.17025-1_all.deb ...
```

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



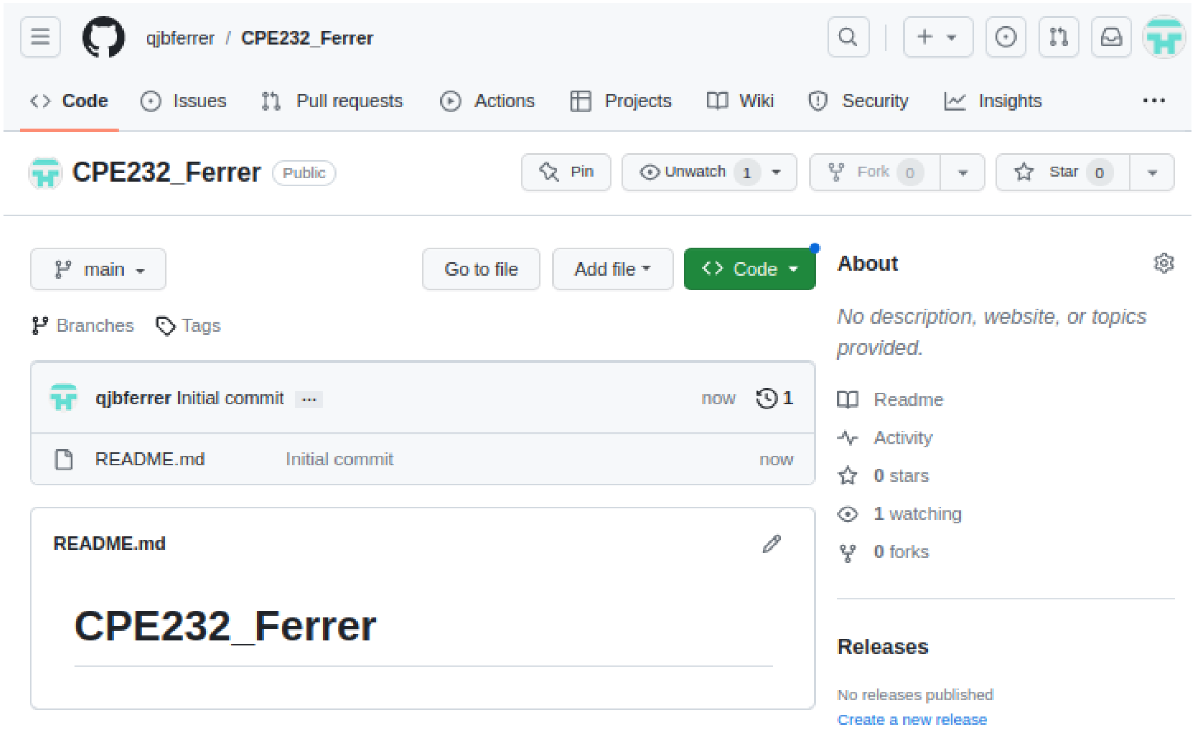
```
josephferrer@workstation:~$ which git
/usr/bin/git
```

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



```
josephferrer@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.



No description, website, or topics provided.

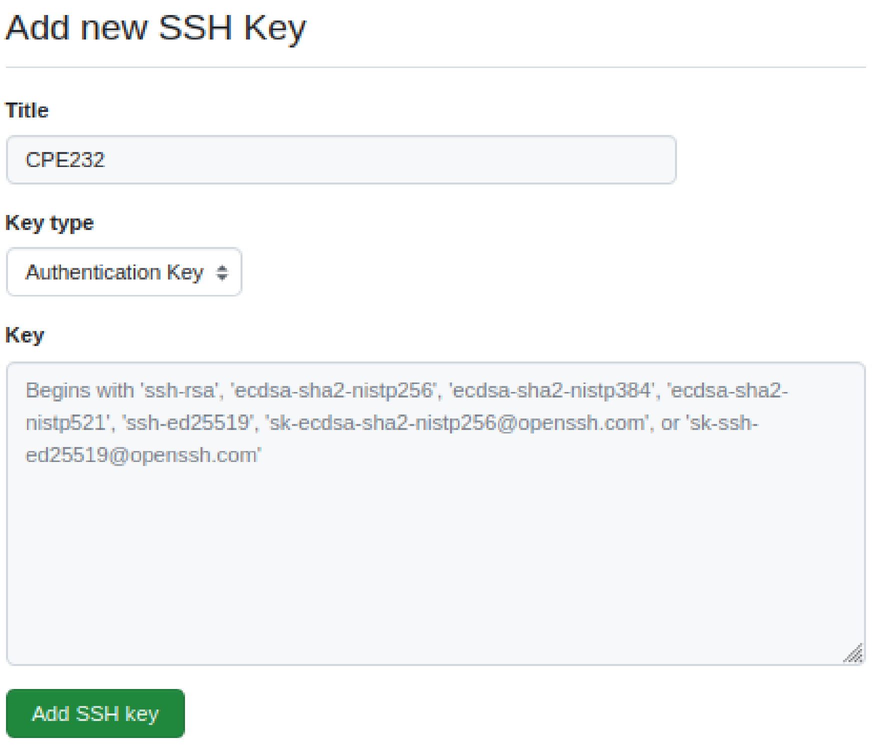
qjbferrer Initial commit		now 1
README.md	Initial commit	now

README.md

CPE232_Ferrer

No releases published

   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

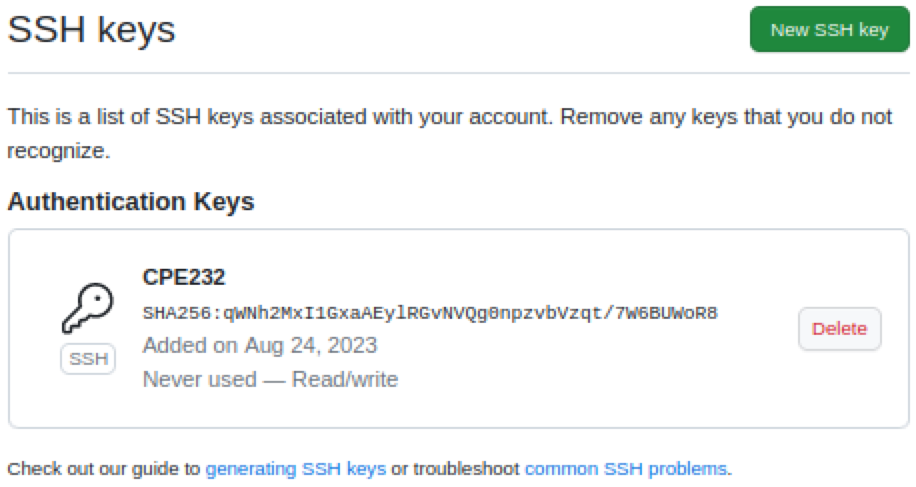
Key type

Authentication Key

Key

Begins with 'ssh-rsa', 'ecdsa-sha2-nistp256', 'ecdsa-sha2-nistp384', 'ecdsa-sha2-nistp521', 'ssh-ed25519', 'sk-ecdsa-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.



SSH keys

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

Authentication Keys

CPE232

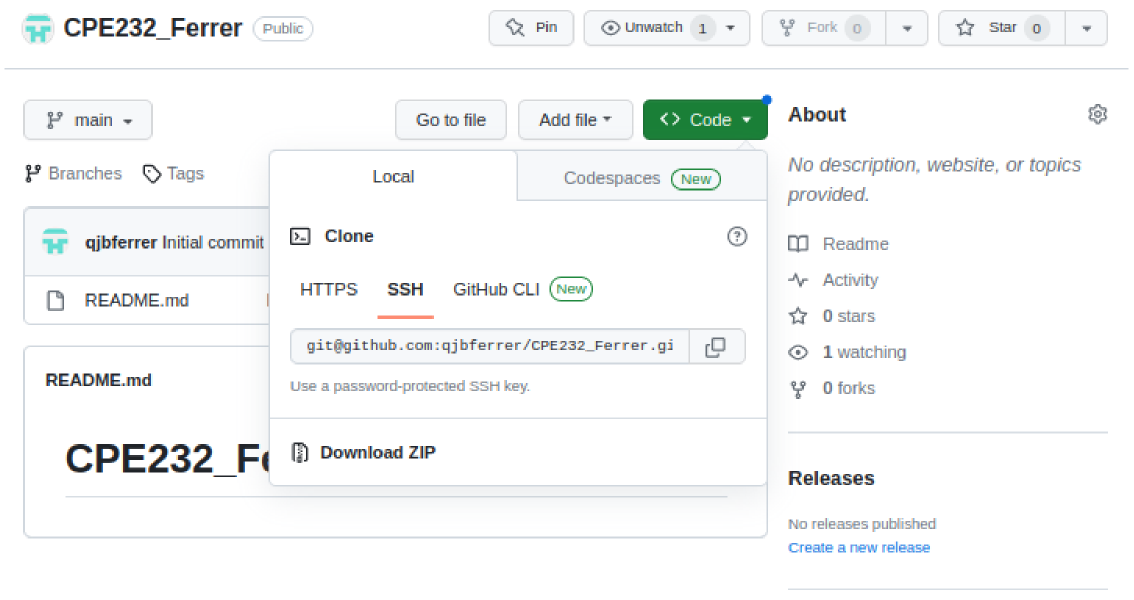
SHA256:qWNh2MxI1GxaAEylRGvNVQg0npzvbVzqt/7W6BUWoR8

Added on Aug 24, 2023

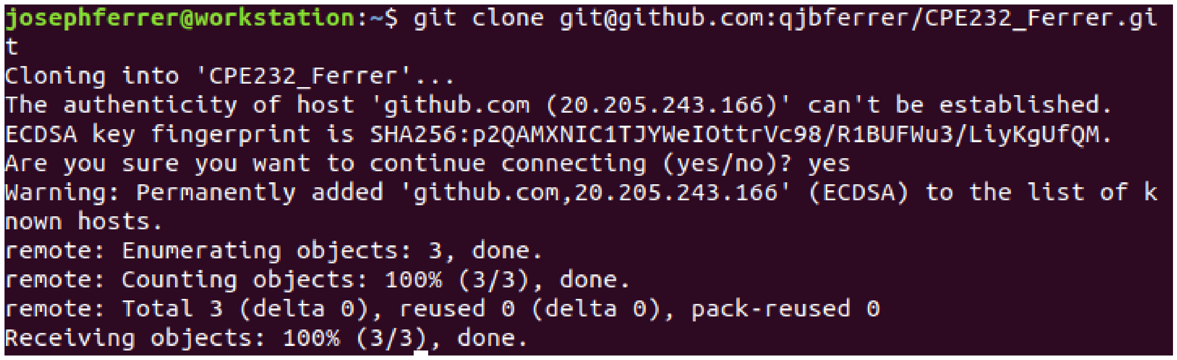
Never used — Read/write

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.

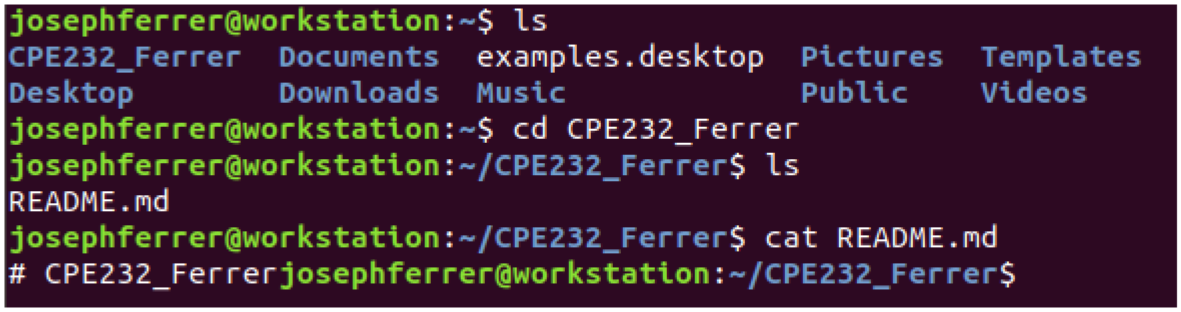


e. Issue the command git clone followed by the copied link. For example, *git clone git@github.com:jvtaylar-cpe/CPE232_yourname.git*. When prompted to continue connecting, type yes and press enter.



```
josephferrer@workstation:~$ git clone git@github.com:qjbferrer/CPE232_Ferrer.gi
t
Cloning into 'CPE232_Ferrer'...
The authenticity of host 'github.com (20.205.243.166)' can't be established.
ECDSA key fingerprint is SHA256:p2QAMXNIC1TJYWeIOttrVc98/R1BUFWu3/LiyKgUfQM.
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 k
nown 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.
```

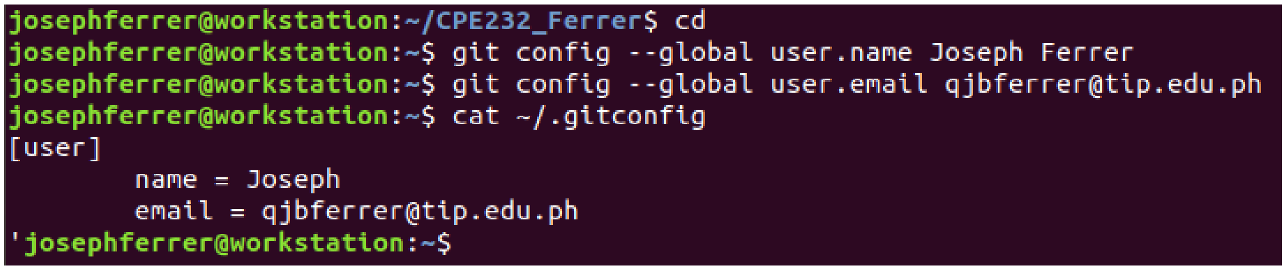
f. To verify that you have cloned the GitHub repository, issue the command *ls*. Observe that you have the CPE232_yourname in the list of your directories. Use CD command to go to that directory and LS command to see the file README.md.



```
josephferrer@workstation:~$ ls
CPE232_Ferrer  Documents  examples.desktop  Pictures  Templates
Desktop        Downloads  Music             Public    Videos
josephferrer@workstation:~$ cd CPE232_Ferrer
josephferrer@workstation:~/CPE232_Ferrer$ ls
README.md
josephferrer@workstation:~/CPE232_Ferrer$ cat README.md
# CPE232_Ferrerjosephferrer@workstation:~/CPE232_Ferrer$
```

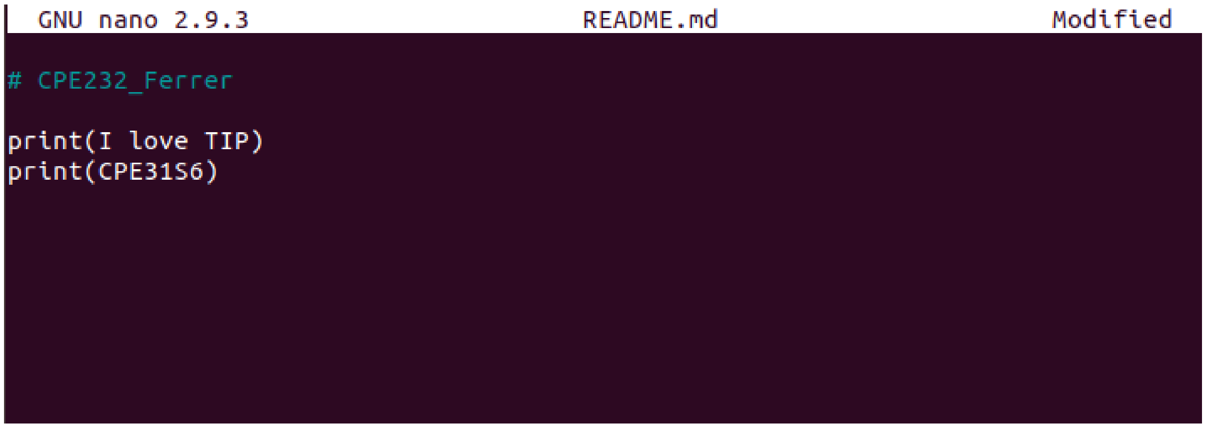
g. Use the following commands to personalize your git.
- *git config --global user.name "Your Name"*
- *git config --global user.email yourname@email.com*

- Verify that you have personalized the config file using the command *cat ~/.gitconfig*



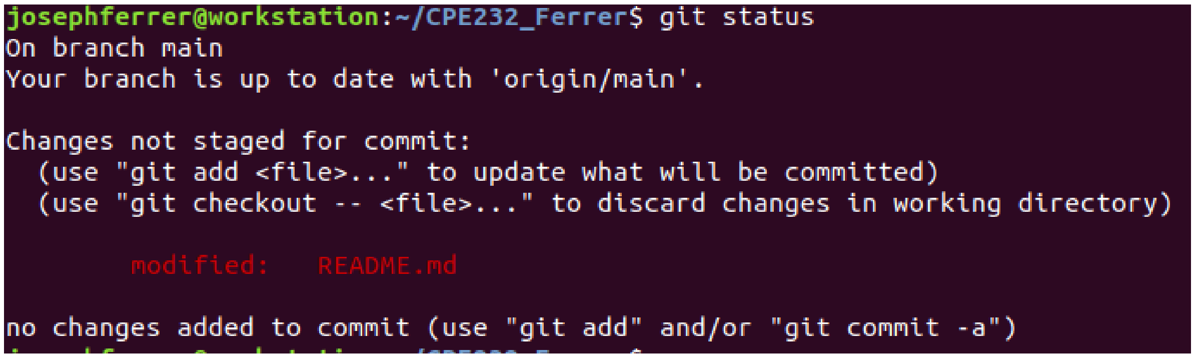
```
josephferrer@workstation:~/CPE232_Ferrer$ cd
josephferrer@workstation:~$ git config --global user.name Joseph Ferrer
josephferrer@workstation:~$ git config --global user.email qjbferrer@tip.edu.ph
josephferrer@workstation:~$ cat ~/.gitconfig
[user]
        name = Joseph
        email = qjbferrer@tip.edu.ph
'josephferrer@workstation:~$
```

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.
nano



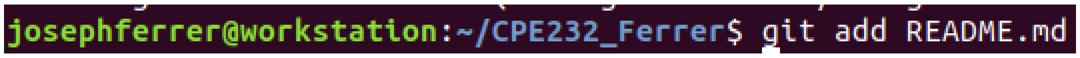
```
  GNU nano 2.9.3                    README.md                         Modified

# CPE232_Ferrer

print(I love TIP)
print(CPE31S6)
```

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?



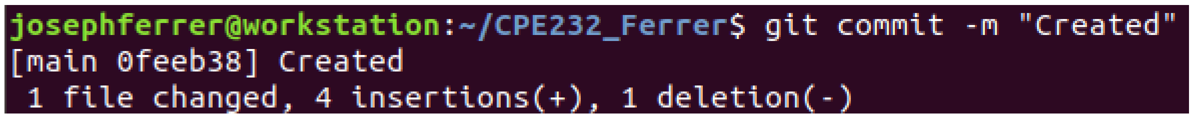
```
josephferrer@workstation:~/CPE232_Ferrer$ git status
On branch main
Your branch is up to date with 'origin/main'.

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.



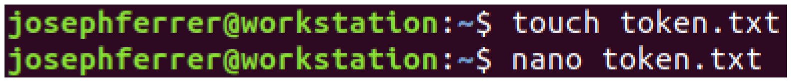
```
josephferrer@workstation:~/CPE232_Ferrer$ git add README.md
```

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.

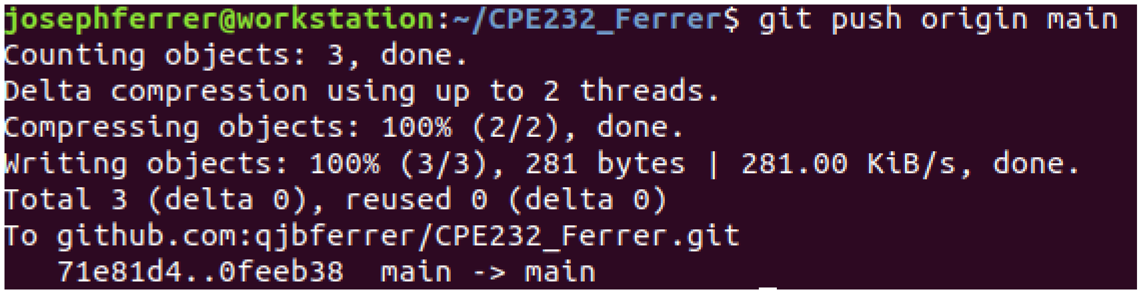


```
josephferrer@workstation:~/CPE232_Ferrer$ git commit -m "Created"
[main 0feeb38] Created
 1 file changed, 4 insertions(+), 1 deletion(-)
```

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.*

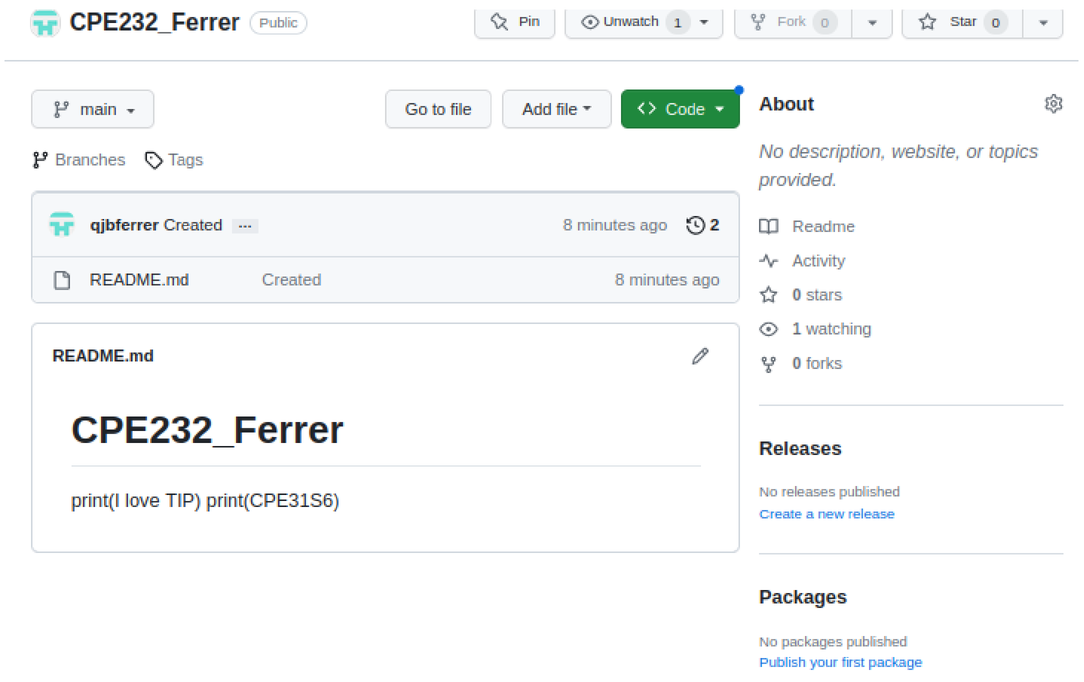


```
josephferrer@workstation:~$ touch token.txt
josephferrer@workstation:~$ nano token.txt
```



```
josephferrer@workstation:~/CPE232_Ferrer$ git push origin main
Counting objects: 3, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 281 bytes | 281.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To github.com:qjbferrer/CPE232_Ferrer.git
   71e81d4..0feeb38  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.



qjbferrer Created ··· 8 minutes ago 2

README.md Created 8 minutes ago

README.md

CPE232_Ferrer

print(I love TIP) print(CPE31S6)

Reflections:
Answer the following:
What sort of things have we so far done to the remote servers using ansible commands?
Some of the things that we have done to the remote servers using ansible commands are the following:
Package Installation and Updates - Ansible can install, update, or remove packages on remote servers using package managers like apt
File Operations - Ansible can copy files from the local machine to remote servers or vice versa. Managing file permissions, ownership, and content.
Security Hardening - Ansible can implement security measures by configuring firewalls and other security-related settings.
SSH Key Distribution - Ansible can distribute SSH public keys to remote servers to enable key-based authentication.

Conclusions/Learnings:

In this activity, there are four objectives to accomplish. The first objective is to configure remote and local machines to connect via SSH using a KEY instead of using a password. I have been able to accomplish the first objective by following the instruction given from the activity. Also, I used syntaxes such as ssh-keygen and ssh-copy-id in order to produce public and private keys for the servers. The second objective is to create a public key and private key. I completed this by using SSH keys and public key authentication. The third objective is to verify connectivity. I achieved this objective by troubleshooting the connectivity and making sure that every syntax is correct. The fourth objective is to set up a Git Repository using local and remote repositories. By the help of GitHub, I have been able to create a GIT Repository and connect it to both of my servers. Configuring and running ad hoc commands from local machine to remote servers is the last objective which is accomplished by doing ansible commands such as ssh key distribution and configuring firewall.