Name: San Jose, Kier Justin F.	**Date Performed: 9/5/2024**
Course/Section: CPE212 – CPE31S21	**Date Submitted: 9/5/2024**
Instructor: Engr. Robin Valenzuela	**Semester and SY: 2024 - 2025**

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.

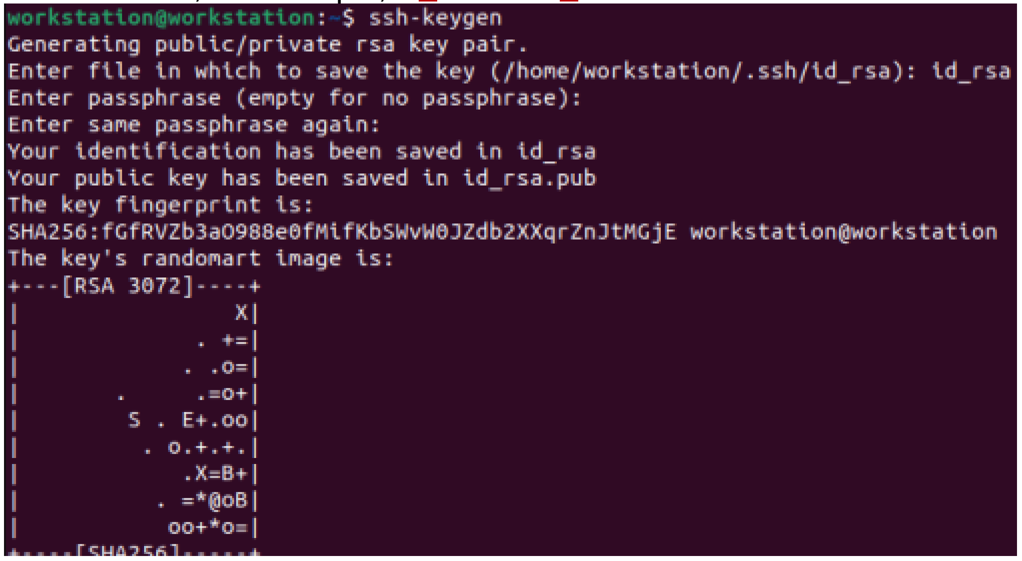
SSH introduced public key authentication as a more secure alternative to the older .rhosts authentication. It improved security by avoiding the need to have password stored in files and eliminated the possibility of a compromised server stealing the user's password.

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

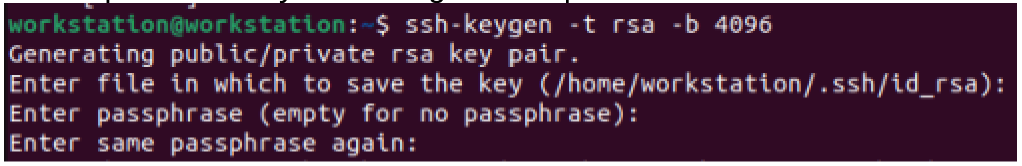
1. The simplest way to generate a key pair is to run *ssh-keygen* without arguments. In this case, it will prompt for the file in which to store keys. First, the tool asked where to save the file. SSH keys for user authentication are usually stored in the users .ssh directory under the home directory.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



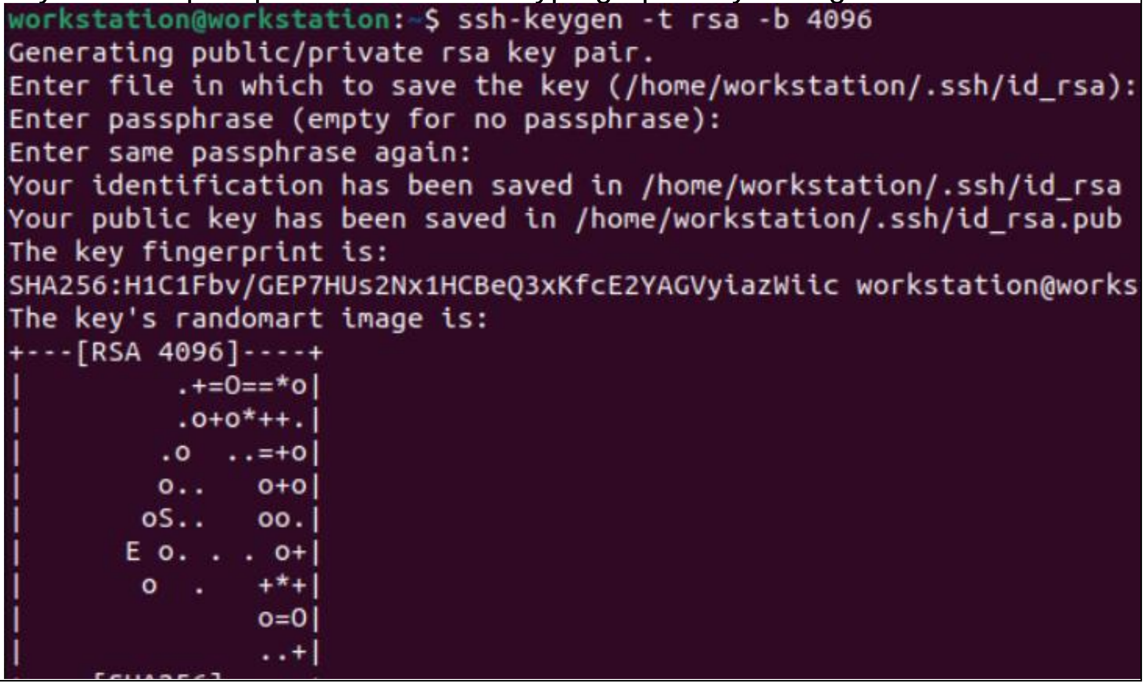
```
workstation@workstation:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/workstation/.ssh/id_rsa): id_rsa
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in id_rsa
Your public key has been saved in id_rsa.pub
The key fingerprint is:
SHA256:fGfRVZb3aO988e0fMifKbSWvW0JZdb2XXqrZnJtMGjE workstation@workstation
The key's randomart image is:
+---[RSA 3072]----+
|                X|
|             . +=|
|            . .o=|
|       .     .=o+|
|        S . E+.oo|
|         . o.+.+.|
|            .X=B+|
|          . =*@oB|
|           oo+*o=|
```

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.



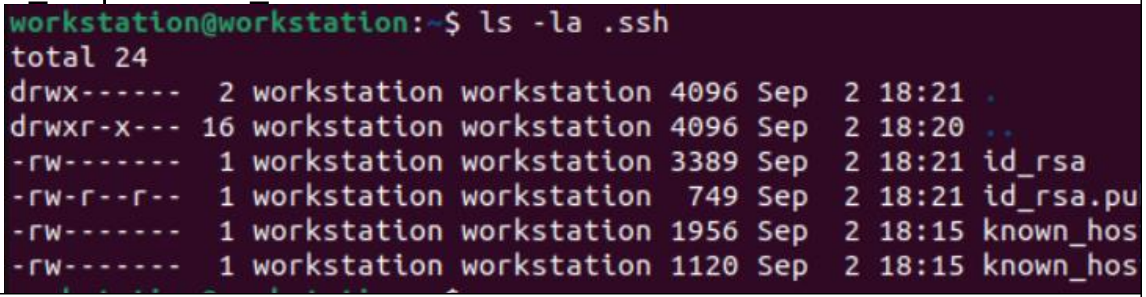
```
workstation@workstation:~$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/workstation/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
```

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.



```
workstation@workstation:~$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/workstation/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/workstation/.ssh/id_rsa
Your public key has been saved in /home/workstation/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:H1C1Fbv/GEP7HUs2Nx1HCBeQ3xKfcE2YAGVyiazWiic workstation@works
The key's randomart image is:
+---[RSA 4096]----+
|         .+=O==*o|
|         .o+o*++.|
|        .o  ..=+o|
|        o..   o+o|
|       oS..   oo.|
|      E o. . . o+|
|       o  .   +*+|
|              o=O|
|              ..+|
```

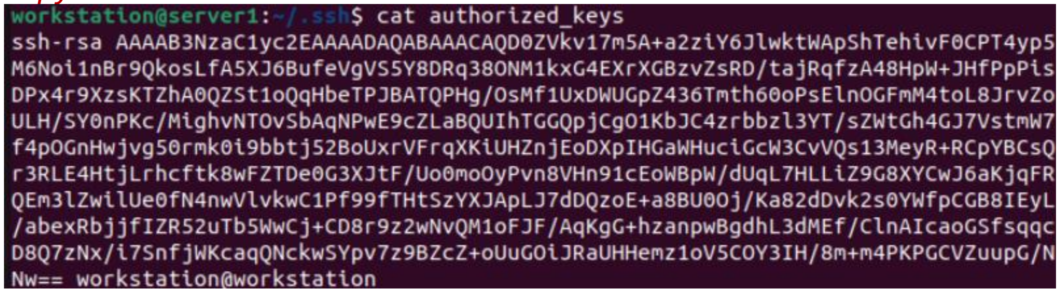
4. Verify that you have created the key by issuing the command *ls -la .ssh.* The command should show the .ssh directory containing a pair of keys. For example, id_rsa.pub and id_rsa.



```
workstation@workstation:~$ ls -la .ssh
total 24
drwx------  2 workstation workstation 4096 Sep  2 18:21 .
drwxr-x--- 16 workstation workstation 4096 Sep  2 18:20 ..
-rw-------  1 workstation workstation 3389 Sep  2 18:21 id_rsa
-rw-r--r--  1 workstation workstation  749 Sep  2 18:21 id_rsa.pu
-rw-------  1 workstation workstation 1956 Sep  2 18:15 known_hos
-rw-------  1 workstation workstation 1120 Sep  2 18:15 known_hos
```

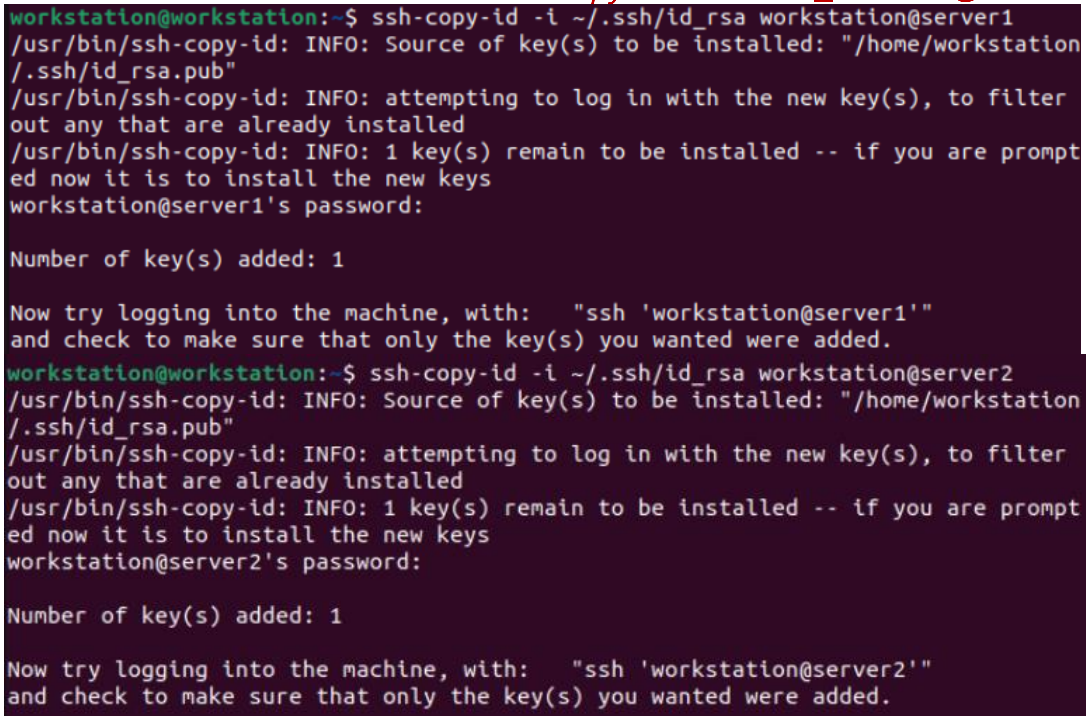
Task 2: Copying the Public Key to the remote servers

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.



```
workstation@server1:~/.ssh$ cat authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAACAQD0ZVkv17m5A+a2ziY6JlwktWApShTehivF0CPT4yp5
M6Noi1nBr9QkosLfA5XJ6BufeVgVS5Y8DRq38ONM1kxG4EXrXGBzvZsRD/tajRqfzA48HpW+JHfPpPis
DPx4r9XzsKTZhA0QZSt1oQqHbeTPJBATQPHg/OsMf1UxDWUGpZ436Tmth60oPsElnOGFmM4toL8JrvZo
ULH/SY0nPKc/MighvNTOvSbAqNPwE9cZLaBQUIhTGGQpjCgO1KbJC4zrbbzl3YT/sZWtGh4GJ7VstmW7
f4pOGnHwjvg50rmk0i9bbtj52BoUxrVFrqXKiUHZnjEoDXpIHGaWHuciGcW3CvVQs13MeyR+RCpYBCsQ
r3RLE4HtjLrhcftk8wFZTDe0G3XJtF/Uo0moOyPvn8VHn91cEoWBpW/dUqL7HLLiZ9G8XYCwJ6aKjqFR
QEm3lZwilUe0fN4nwVlvkwC1Pf99fTHtSzYXJApLJ7dDQzoE+a8BU0Oj/Ka82dDvk2s0YWfpCGB8IEyL
/abexRbjjfIZR52uTb5WwCj+CD8r9z2wNvQM1oFJF/AqKgG+hzanpwBgdhL3dMEf/ClnAIcaoGSfsqqc
D8Q7zNx/i7SnfjWKcaqQNckwSYpv7z9BZcZ+oUuGOiJRaUHHemz1oV5COY3IH/8m+m4PKPGCVZuupG/N
Nw== workstation@workstation
```

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



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

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'workstation@server1'"
and check to make sure that only the key(s) you wanted were added.
workstation@workstation:~$ ssh-copy-id -i ~/.ssh/id_rsa workstation@server2
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/workstation
/.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 prompt
ed now it is to install the new keys
workstation@server2's password:

Number of key(s) added: 1

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

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?

Reflections:
Answer the following:
1. How will you describe the ssh-program? What does it do?
   - SSH allows secure remote access to a computer or server or an unsecured network, such as the internet.
2. How do you know that you already installed the public key to the remote servers?
   - You can check your local machine's know_hosts file, it is typically found in ~/ .ssh/ directory, which stores records of known hosts and their public keys.

Part 2: Discussion

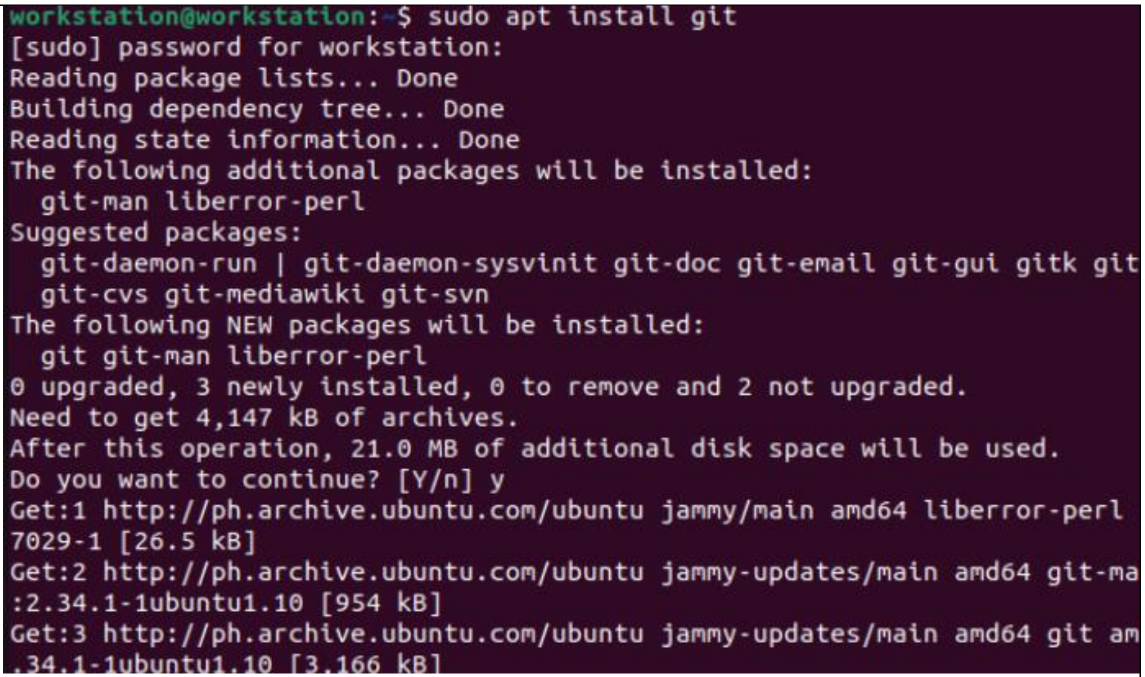
Provide screenshots for each task.

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:

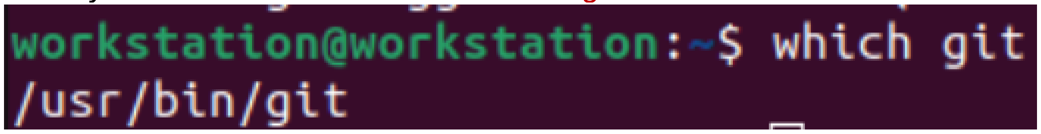
- Creating a repository
- Forking a repository
- Managing files
- Being social

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*



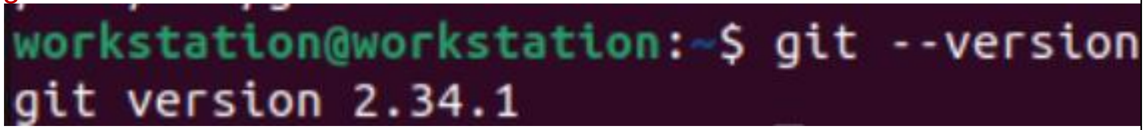
```
workstation@workstation:~$ sudo apt install git
[sudo] password for workstation:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  git-man liberror-perl
Suggested packages:
  git-daemon-run | git-daemon-sysvinit git-doc git-email git-gui gitk git
  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
7029-1 [26.5 kB]
Get:2 http://ph.archive.ubuntu.com/ubuntu jammy-updates/main amd64 git-ma
:2.34.1-1ubuntu1.10 [954 kB]
Get:3 http://ph.archive.ubuntu.com/ubuntu jammy-updates/main amd64 git am
.34.1-1ubuntu1.10 [3,166 kB]
```

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



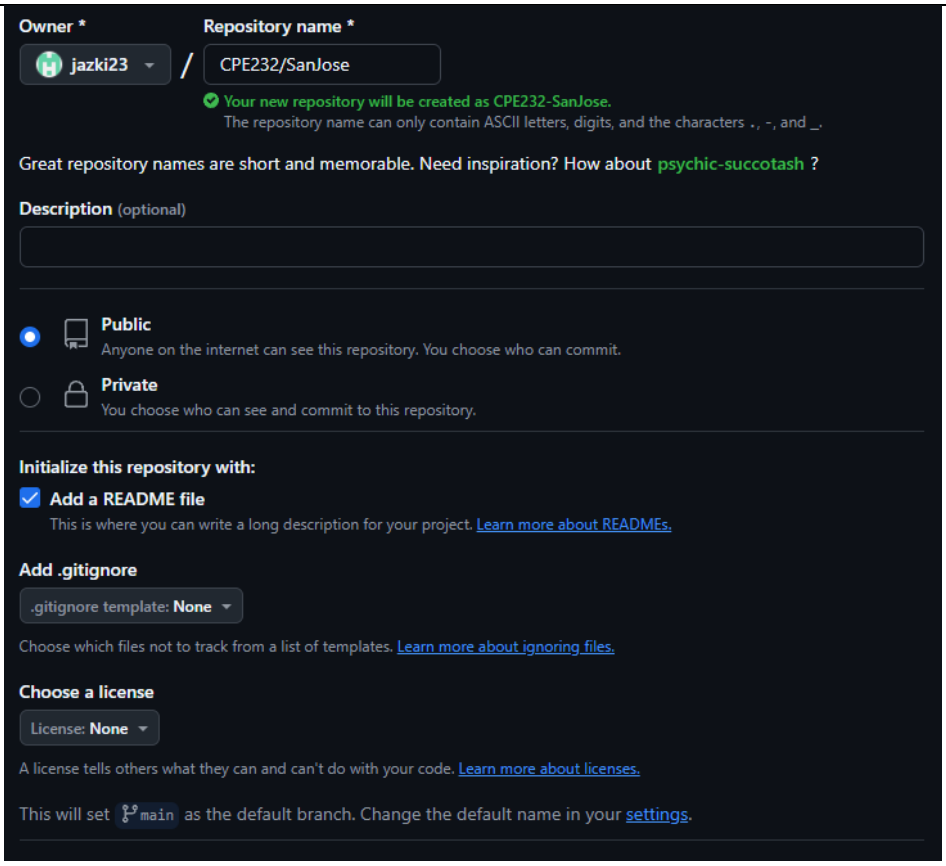
```
workstation@workstation:~$ which git
/usr/bin/git
```

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



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

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


Owner * jazki23 / Repository name * CPE232/SanJose

Your new repository will be created as CPE232-SanJose.
The repository name can only contain ASCII letters, digits, and the characters ., -, and _.

Great repository names are short and memorable. Need inspiration? How about psychic-succotash ?

Description (optional)

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

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

Initialize this repository with:
Add a README file
This is where you can write a long description for your project. Learn more about READMEs.

Add .gitignore
.gitignore template: None
Choose which files not to track from a list of templates. Learn more about ignoring files.

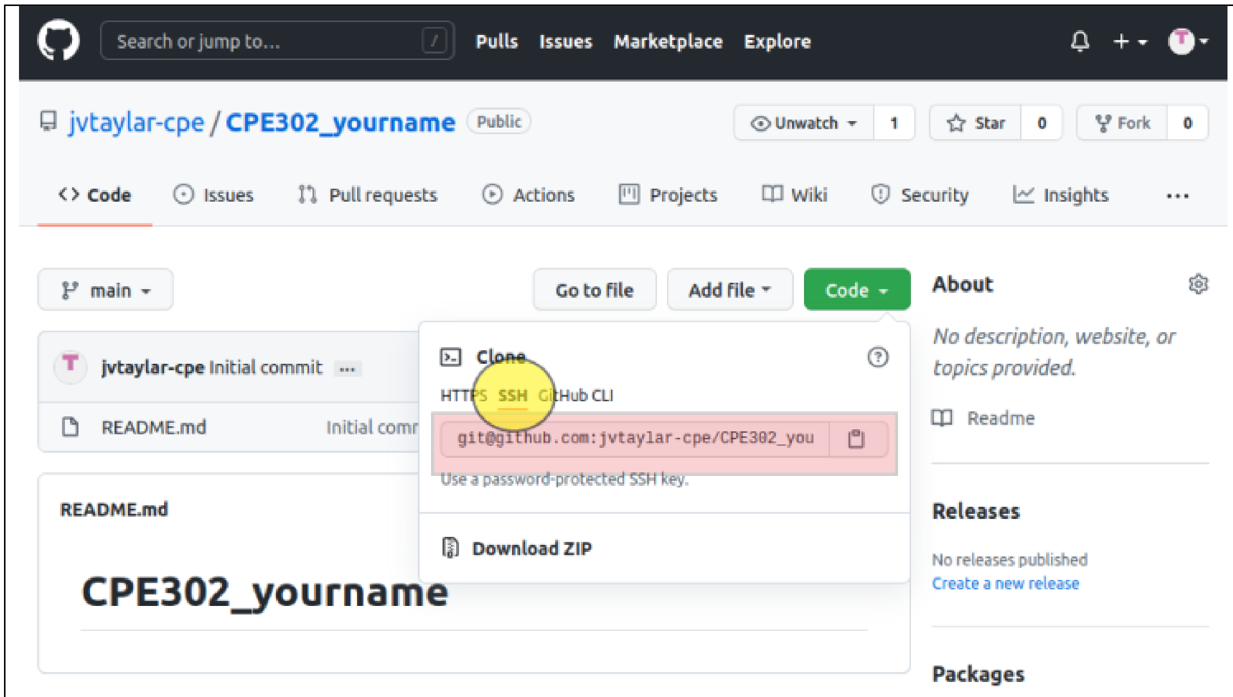
Choose a license
License: None
A license tells others what they can and can't do with your code. Learn more about licenses.

This will set `main` as the default branch. Change the default name in your settings.

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.



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

Use a password-protected SSH key.

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.

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.

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*

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?

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.

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.*
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?
4. How important is the inventory file?

Conclusions/Learnings: