Name: FRIAS, ABEGAIL L.	**Date Performed: Sep 6, 2024**
Course/Section: CPE212 - CPE31S21	**Date Submitted: September 6, 2024**
Instructor: Engr. Robin Valenzuela	**Semester and SY: 1st Sem/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

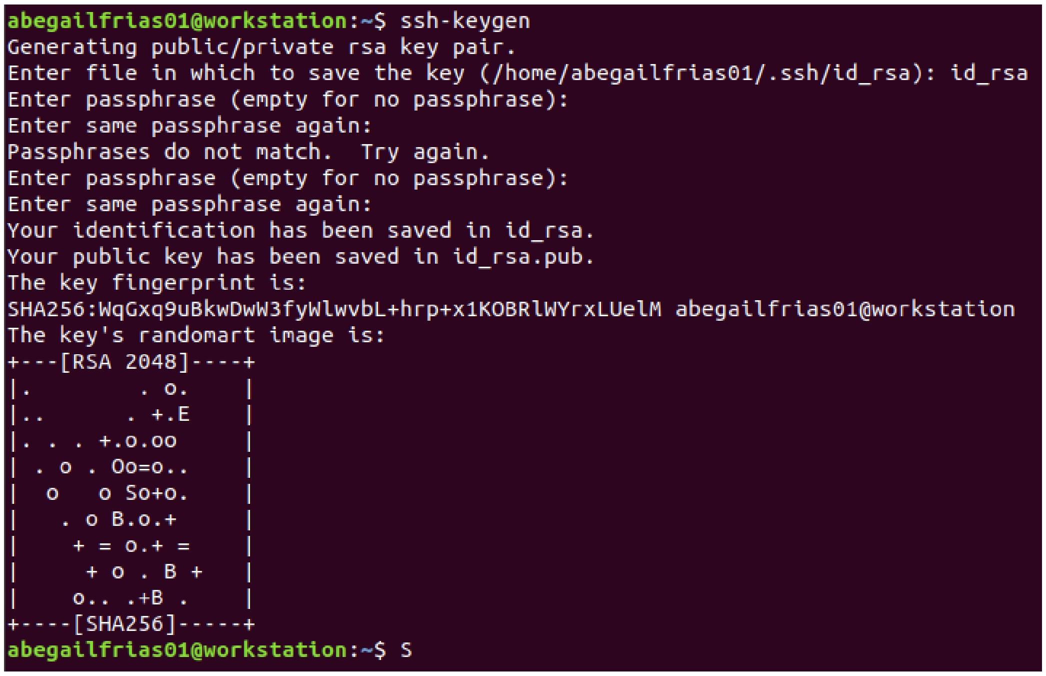
The SSH protocol uses public key cryptography for authenticating hosts and users. The authentication keys, called SSH keys, are created using the keygen program.

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.

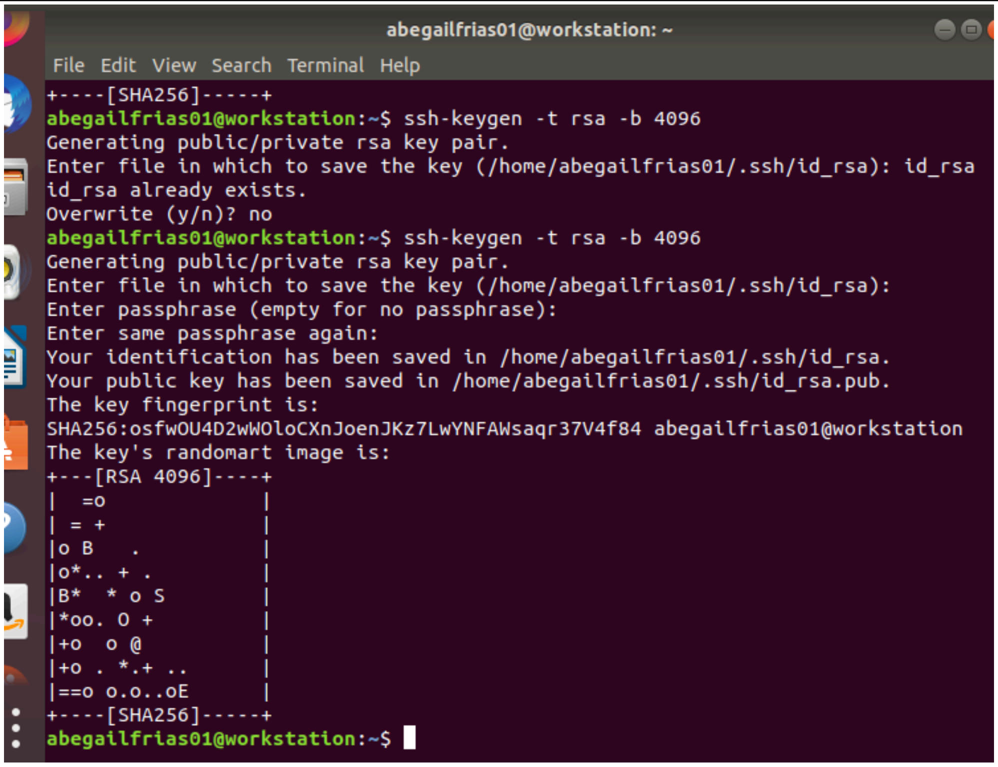
Task 1: Create an SSH Key Pair for User Authentication

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*.



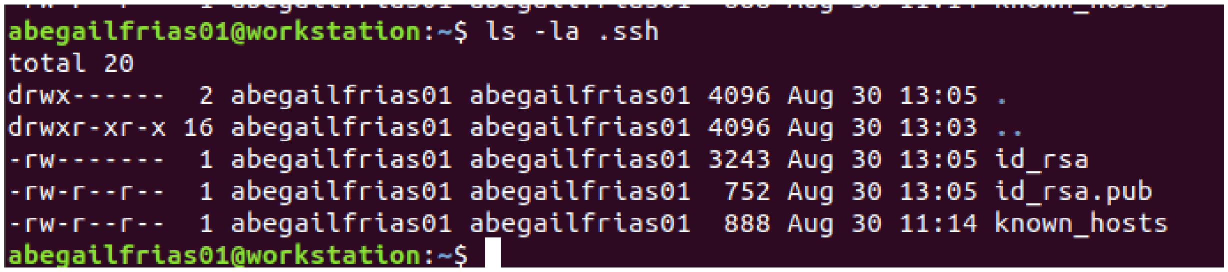
```
abegailfrias01@workstation:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/abegailfrias01/.ssh/id_rsa): id_rsa
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Passphrases do not match.  Try again.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in id_rsa.
Your public key has been saved in id_rsa.pub.
The key fingerprint is:
SHA256:WqGxq9uBkwDwW3fyWlwvbL+hrp+x1KOBRlWYrxLUelM abegailfrias01@workstation
The key's randomart image is:
+---[RSA 2048]----+
|.        . o.    |
|..      . +.E    |
|. . . +.o.oo     |
| . o . Oo=o..    |
|  o   o So+o.    |
|   . o B.o.+     |
|    + = o.+ =    |
|     + o . B +   |
|    o.. .+B .    |
+----[SHA256]-----+
abegailfrias01@workstation:~$ S
```

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



```
+----[SHA256]-----+
abegailfrias01@workstation:~$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/abegailfrias01/.ssh/id_rsa): id_rsa
id_rsa already exists.
Overwrite (y/n)? no
abegailfrias01@workstation:~$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/abegailfrias01/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/abegailfrias01/.ssh/id_rsa.
Your public key has been saved in /home/abegailfrias01/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:osfwOU4D2wWOloCXnJoenJKz7LwYNFAWsaqr37V4f84 abegailfrias01@workstation
The key's randomart image is:
+---[RSA 4096]----+
|  =o             |
| = +             |
|o B   .          |
|o*.. + .         |
|B*  * o S        |
|*oo. O +         |
|+o  o @          |
|+o . *.+ ..      |
|==o o.o..oE      |
+----[SHA256]-----+
abegailfrias01@workstation:~$
```

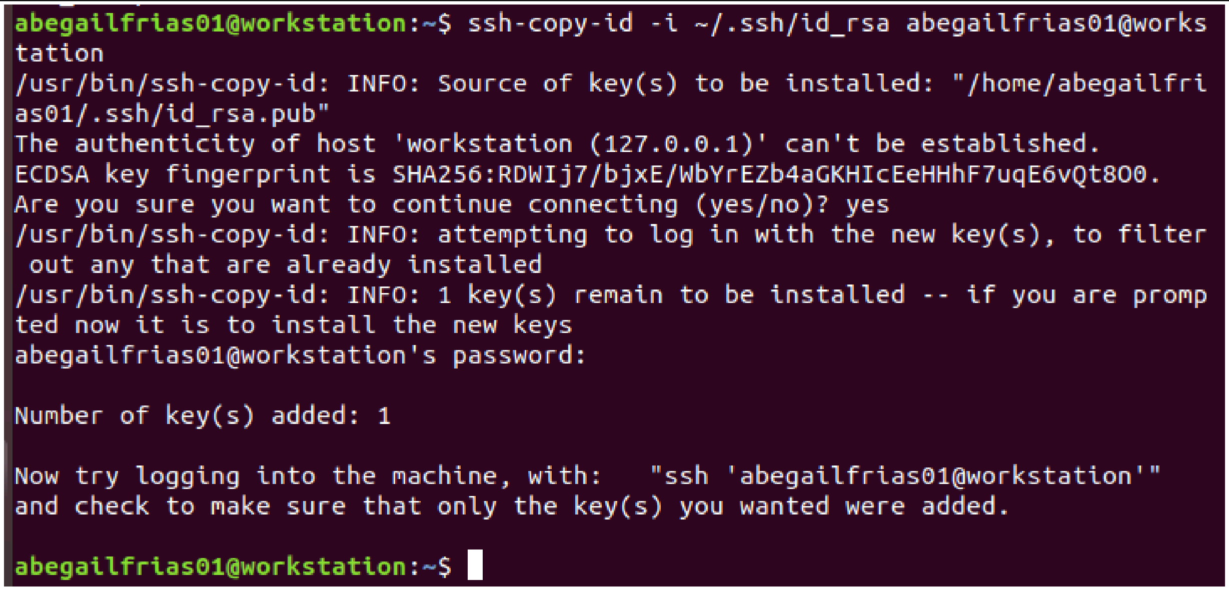
3. When asked for a passphrase, just press enter. The passphrase is used for encrypting the key, so that it cannot be used even if someone obtains the private key file. The passphrase should be cryptographically strong.
4. Verify that you have created the key by issuing the command *ls -la .ssh.* The command should show the .ssh directory containing a pair of keys. For example, id_rsa.pub and id_rsa.



```
abegailfrias01@workstation:~$ ls -la .ssh
total 20
drwx------  2 abegailfrias01 abegailfrias01 4096 Aug 30 13:05 .
drwxr-xr-x 16 abegailfrias01 abegailfrias01 4096 Aug 30 13:03 ..
-rw-------  1 abegailfrias01 abegailfrias01 3243 Aug 30 13:05 id_rsa
-rw-r--r--  1 abegailfrias01 abegailfrias01  752 Aug 30 13:05 id_rsa.pub
-rw-r--r--  1 abegailfrias01 abegailfrias01  888 Aug 30 11:14 known_hosts
abegailfrias01@workstation:~$
```

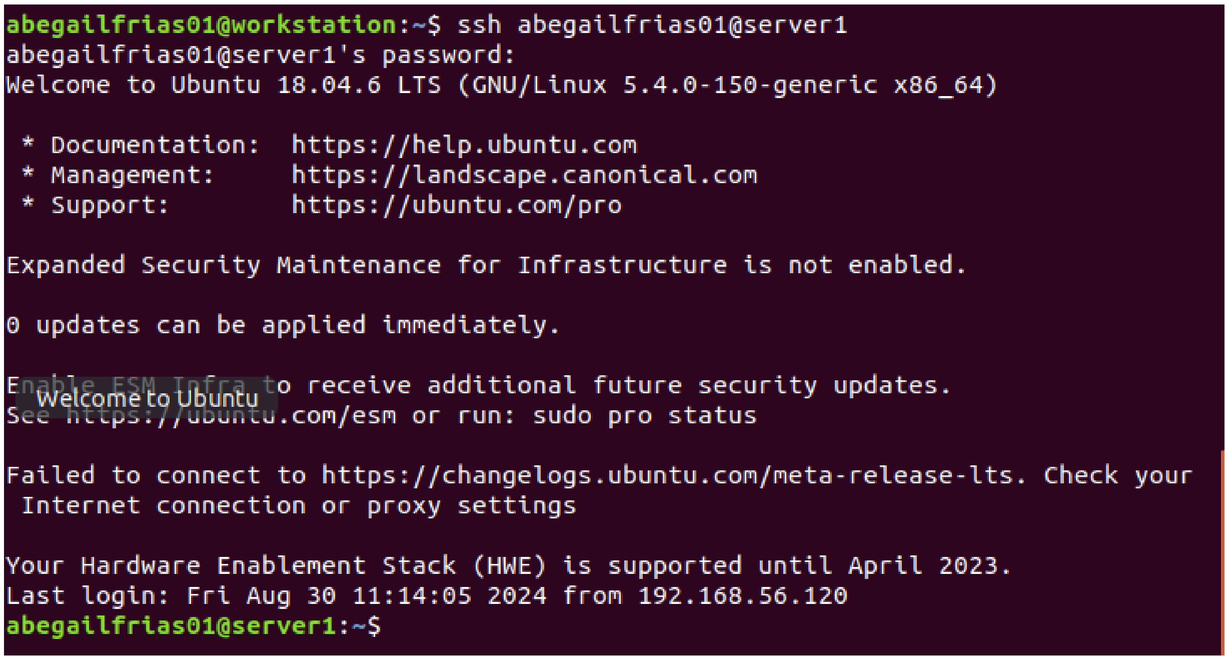
Task 2: Copying the Public Key to the remote servers

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*

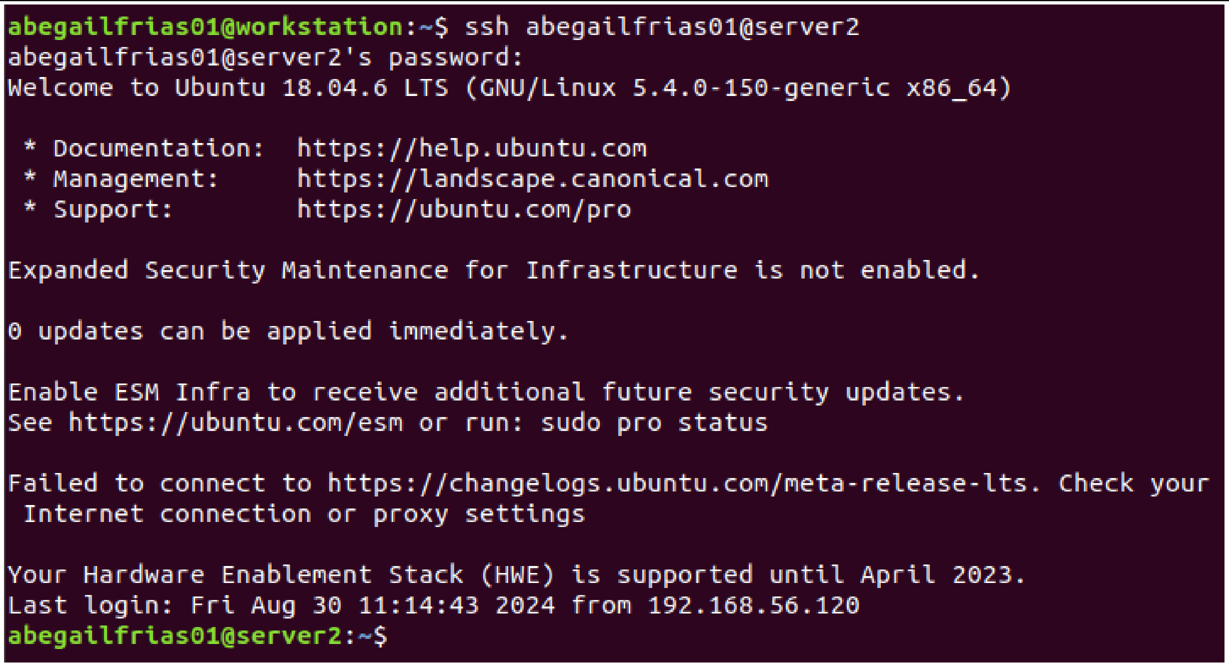


```
abegailfrias01@workstation:~$ ssh-copy-id -i ~/.ssh/id_rsa abegailfrias01@works
tation
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/abegailfri
as01/.ssh/id_rsa.pub"
The authenticity of host 'workstation (127.0.0.1)' can't be established.
ECDSA key fingerprint is SHA256:RDWIj7/bjxE/WbYrEZb4aGKHIcEeHHhF7uqE6vQt8O0.
Are you sure you want to continue connecting (yes/no)? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter
 out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are promp
ted now it is to install the new keys
abegailfrias01@workstation's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'abegailfrias01@workstation'"
and check to make sure that only the key(s) you wanted were added.

abegailfrias01@workstation:~$
```

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?



```
abegailfrias01@workstation:~$ ssh abegailfrias01@server1
abegailfrias01@server1's password:
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/pro

Expanded Security Maintenance for Infrastructure is not enabled.

0 updates can be applied immediately.

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

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

Your Hardware Enablement Stack (HWE) is supported until April 2023.
Last login: Fri Aug 30 11:14:05 2024 from 192.168.56.120
abegailfrias01@server1:~$
```



```
abegailfrias01@workstation:~$ ssh abegailfrias01@server2
abegailfrias01@server2's password:
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/pro

Expanded Security Maintenance for Infrastructure is not enabled.

0 updates can be applied immediately.

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

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

Your Hardware Enablement Stack (HWE) is supported until April 2023.
Last login: Fri Aug 30 11:14:43 2024 from 192.168.56.120
abegailfrias01@server2:~$
```

Reflections:
Answer the following:

1. How will you describe the ssh-program? What does it do?
    - **With the help of the SSH application users can safely transfer files, log into distant computers and execute commands across a secure network connection between a client and a server. In order to improve security, key pairs are used in place of passwords for authentication using public key cryptography.**

2. How do you know that you already installed the public key to the remote servers?
    - **When the public key is added to the server's ~/.ssh/authorized_keys file and you are able to SSH into the remote machine without being required for a password, you may be certain that the key is installed on the distant server. Using ssh -v to check if the key is used during the connection is another way to confirm this.**

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**).

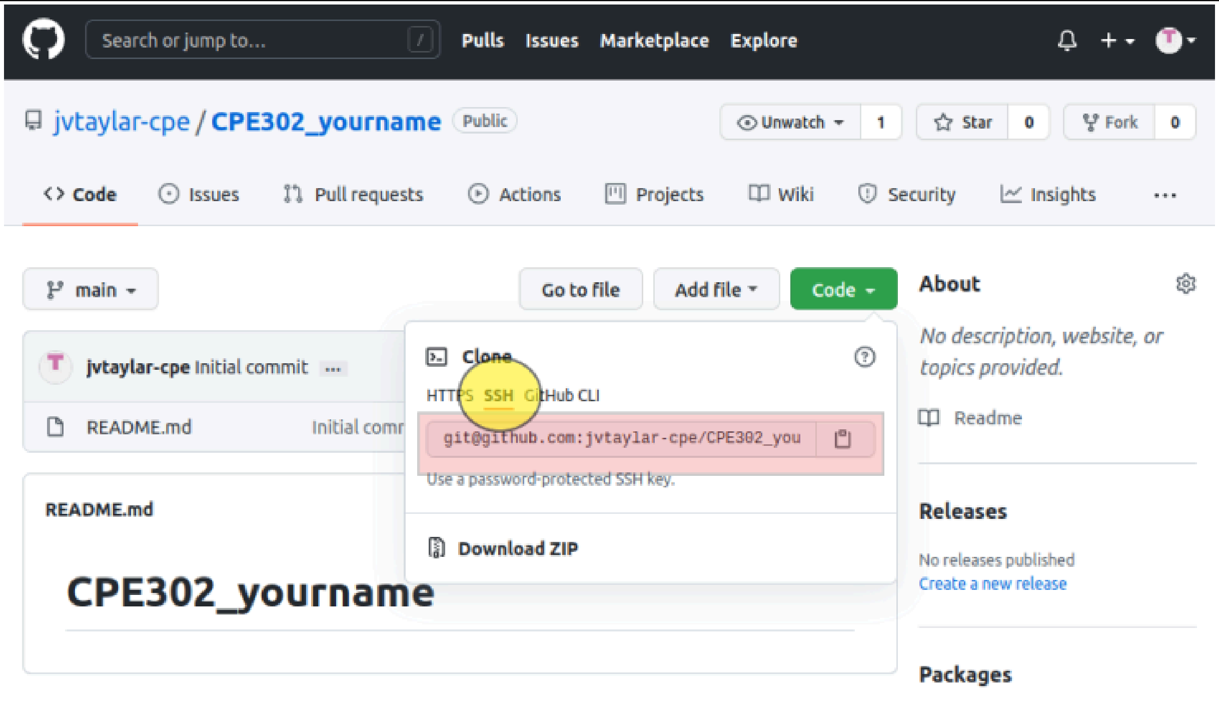
Set up Git

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*
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.
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.



```
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.

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?
    - **As of now I have employed Ansible commands to automate diverse processes on distant servers, including but not limited to software installation, administrative chores and uptime verification. Ansible makes server management simpler and more scalable by enabling us to effectively manage several machines in parallel by executing commands or distributing updates.**

4. How important is the inventory file?

- **The inventory file is essential since it lists the groups and hosts that will be under management. It facilitates the grouping of hosts and guarantees that commands are run on the appropriate systems which makes it simpler to scale automation operations across big infrastructure.**

Conclusions/Learnings:

-