Name: Bencito, Sonny Jay	**Date Performed: 08/31/2022**
Course/Section: CPE31S24	**Date Submitted: 09/01/2022**
Instructor: Dr. Jonathan Taylar	**Semester and SY: 1st – 22-2023**

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.

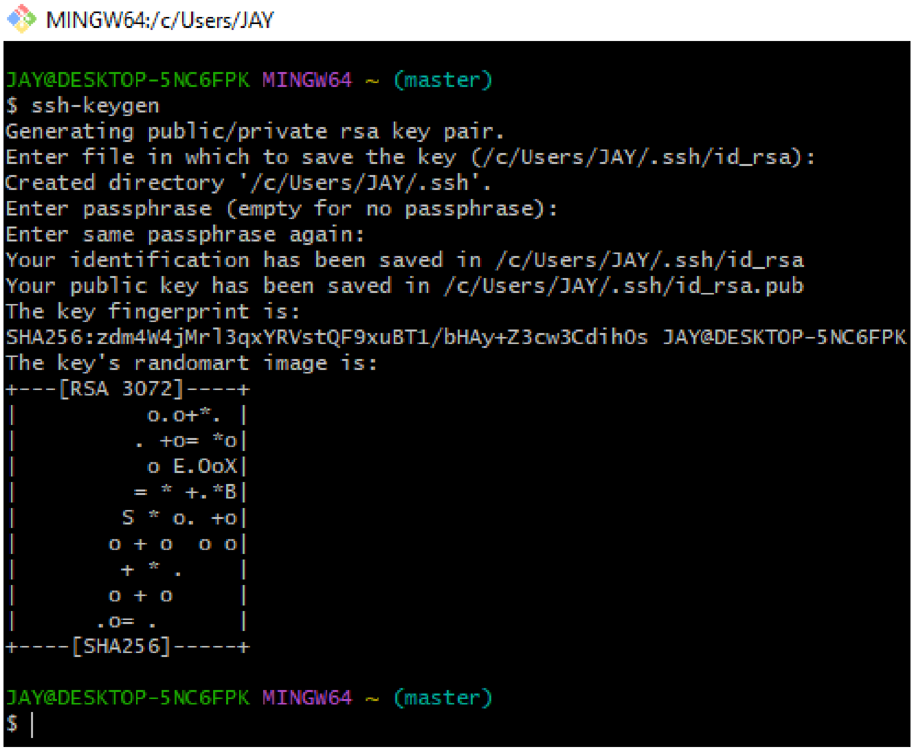
SSH introduced public key authentication as a more secure alternative to the older .rhosts authentication. It improved security by avoiding the need to have password stored in files and eliminated the possibility of a compromised server stealing the user's password.

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

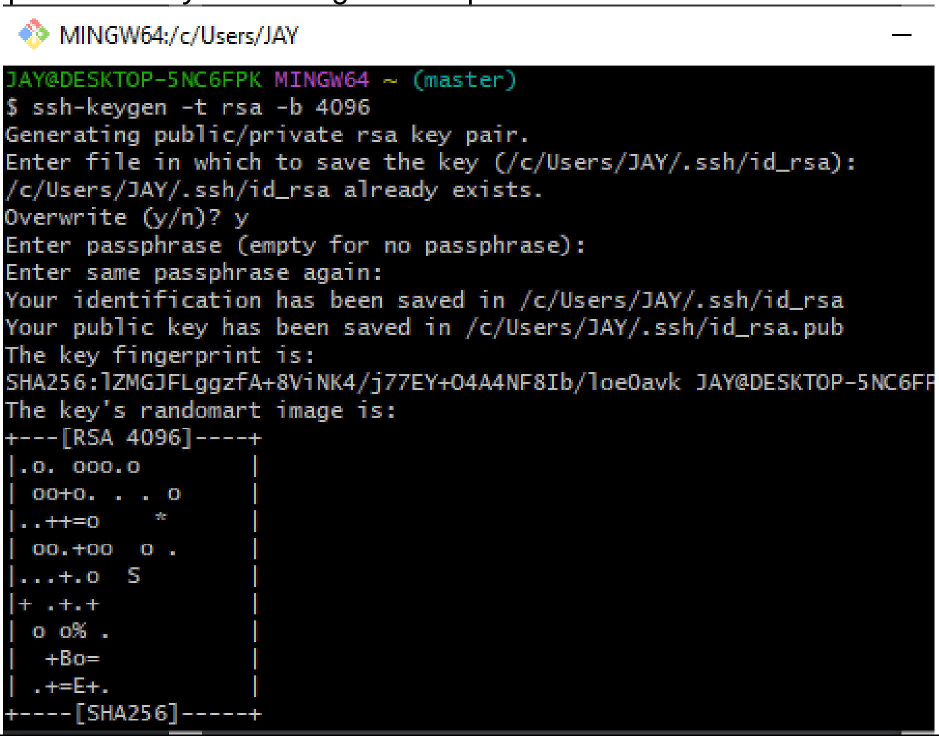
1. The simplest way to generate a key pair is to run *ssh-keygen* without arguments. In this case, it will prompt for the file in which to store keys. First, the tool asked where to save the file. SSH keys for user authentication are usually stored in the users .ssh directory under the home directory.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



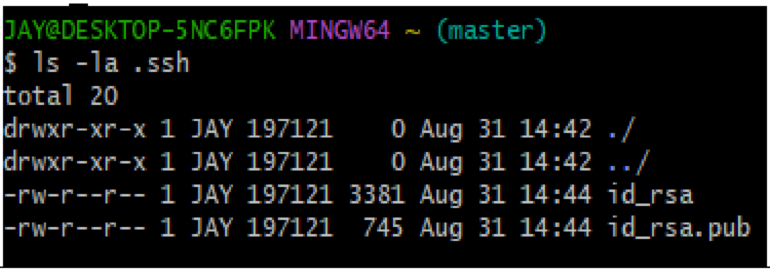
```
JAY@DESKTOP-5NC6FPK MINGW64 ~ (master)
$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/JAY/.ssh/id_rsa):
Created directory '/c/Users/JAY/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /c/Users/JAY/.ssh/id_rsa
Your public key has been saved in /c/Users/JAY/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:zdm4W4jMrl3qxYRVstQF9xuBT1/bHAy+Z3cw3CdihOs JAY@DESKTOP-5NC6FPK
The key's randomart image is:
+---[RSA 3072]----+
|          o.o+*. |
|         . +o= *o|
|          o E.OoX|
|         = * +.*B|
|        S * o. +o|
|       o + o  o o|
|        + * .    |
|       o + o     |
|      .o= .      |
+----[SHA256]-----+

JAY@DESKTOP-5NC6FPK MINGW64 ~ (master)
$ |
```

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.



```
JAY@DESKTOP-5NC6FPK MINGW64 ~ (master)
$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/JAY/.ssh/id_rsa):
/c/Users/JAY/.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 /c/Users/JAY/.ssh/id_rsa
Your public key has been saved in /c/Users/JAY/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:lZMGJFLggzfA+8ViNK4/j77EY+O4A4NF8Ib/loeOavk JAY@DESKTOP-5NC6FP
The key's randomart image is:
+---[RSA 4096]----+
|.o. ooo.o        |
| oo+o. . . o     |
|..++=o    *      |
| oo.+oo  o .     |
|...+.o  S        |
|+ .+.+           |
| o o% .          |
|  +Bo=           |
| .+=E+.          |
+----[SHA256]-----+
```

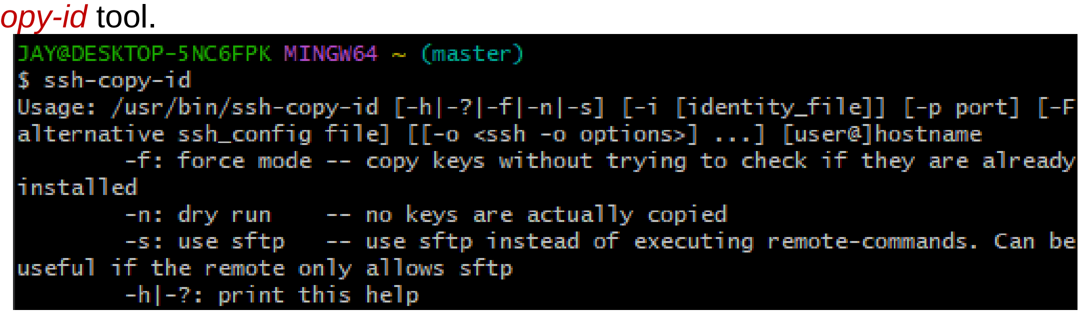
3. When asked for a passphrase, just press enter. The passphrase is used for encrypting the key, so that it cannot be used even if someone obtains the private key file. The passphrase should be cryptographically strong.
4. Verify that you have created the key by issuing the command *ls -la .ssh.* The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



```
JAY@DESKTOP-5NC6FPK MINGW64 ~ (master)
$ ls -la .ssh
total 20
drwxr-xr-x 1 JAY 197121    0 Aug 31 14:42 ./
drwxr-xr-x 1 JAY 197121    0 Aug 31 14:42 ../
-rw-r--r-- 1 JAY 197121 3381 Aug 31 14:44 id_rsa
-rw-r--r-- 1 JAY 197121  745 Aug 31 14:44 id_rsa.pub
```

Task 2: Copying the Public Key to the remote servers

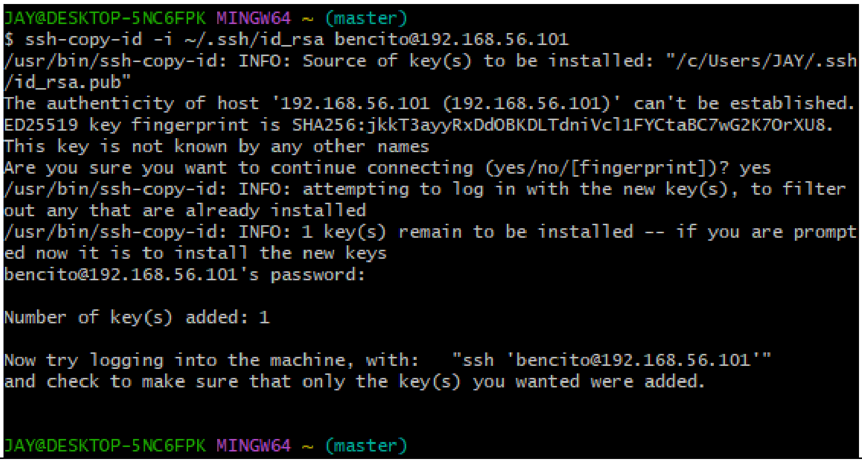
1. To use public key authentication, the public key must be copied to a server and installed in an *authorized_keys* file. This can be conveniently done using the *ssh-copy-id* tool.



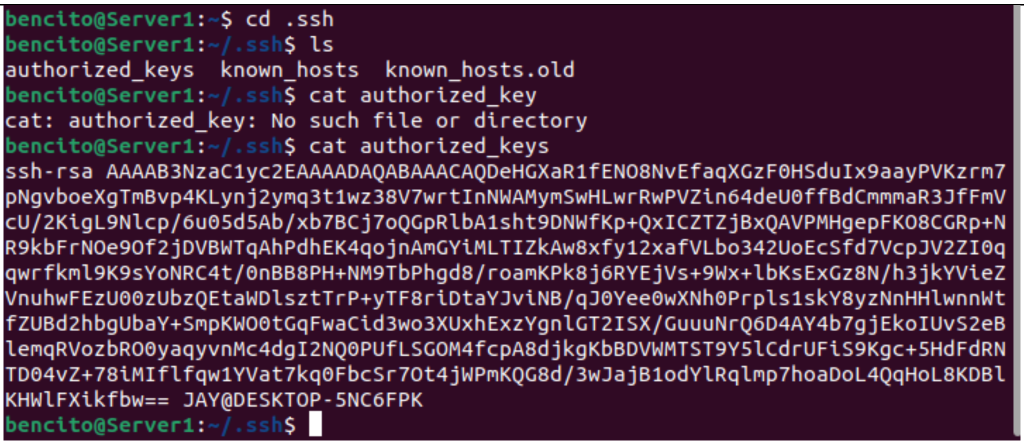
```
JAY@DESKTOP-5NC6FPK MINGW64 ~ (master)
$ ssh-copy-id
Usage: /usr/bin/ssh-copy-id [-h|-?|-f|-n|-s] [-i [identity_file]] [-p port] [-F
alternative ssh_config file] [[-o <ssh -o options>] ...] [user@]hostname
        -f: force mode -- copy keys without trying to check if they are already
installed
        -n: dry run    -- no keys are actually copied
        -s: use sftp   -- use sftp instead of executing remote-commands. Can be
useful if the remote only allows sftp
        -h|-?: print this help
```

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

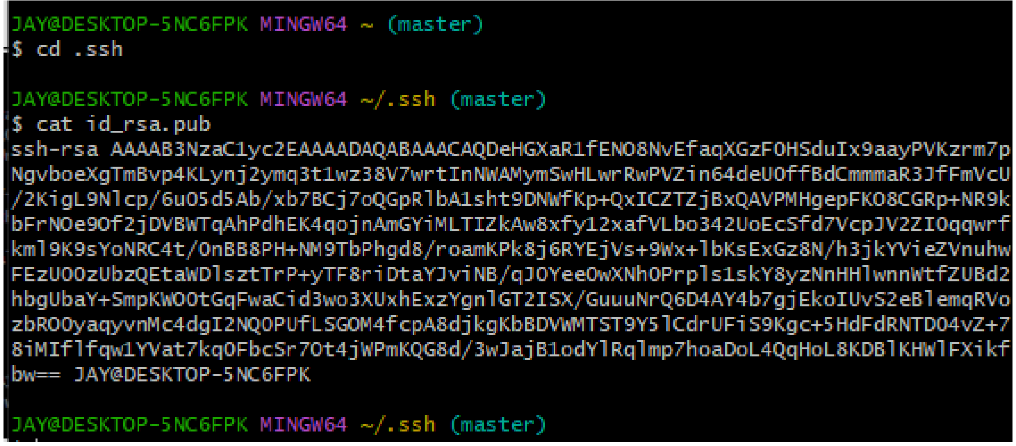
Server 1



```
JAY@DESKTOP-5NC6FPK MINGW64 ~ (master)
$ ssh-copy-id -i ~/.ssh/id_rsa bencito@192.168.56.101
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/c/Users/JAY/.ssh
/id_rsa.pub"
The authenticity of host '192.168.56.101 (192.168.56.101)' can't be established.
ED25519 key fingerprint is SHA256:jkkT3ayyRxDdOBKDLTdniVcl1FYCtaBC7wG2K7OrXU8.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? 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 prompt
ed now it is to install the new keys
bencito@192.168.56.101's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'bencito@192.168.56.101'"
and check to make sure that only the key(s) you wanted were added.


JAY@DESKTOP-5NC6FPK MINGW64 ~ (master)
```

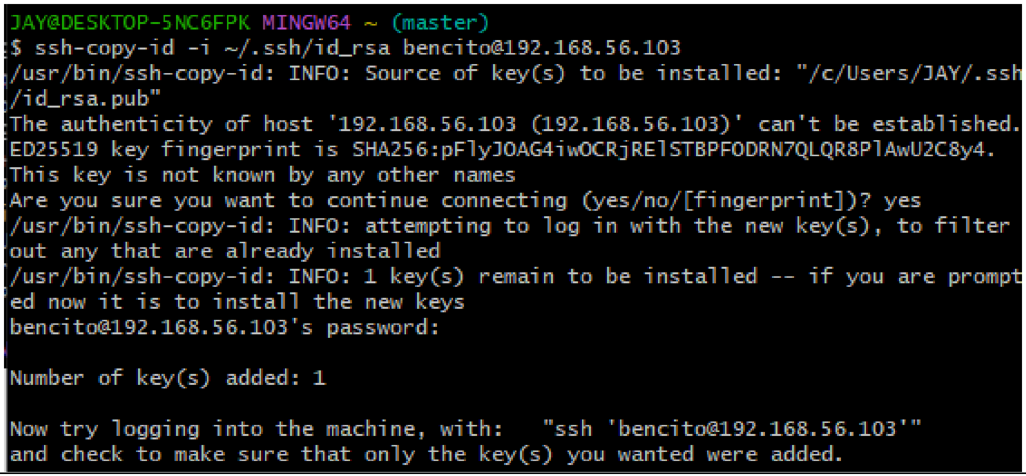


```
bencito@Server1:~$ cd .ssh
bencito@Server1:~/.ssh$ ls
authorized_keys  known_hosts  known_hosts.old
bencito@Server1:~/.ssh$ cat authorized_key
cat: authorized_key: No such file or directory
bencito@Server1:~/.ssh$ cat authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAACAQDeHGXaR1fENO8NvEfaqXGzF0HSduIx9aayPVKzrm7
pNgvboeXgTmBvp4KLynj2ymq3t1wz38V7wrtInNWAMymSwHLwrRwPVZin64deU0ffBdCmmmaR3JfFmV
cU/2KigL9Nlcp/6u05d5Ab/xb7BCj7oQGpRlbA1sht9DNWfKp+QxICZTZjBxQAVPMHgepFKO8CGRp+N
R9kbFrNOe9Of2jDVBWTqAhPdhEK4qojnAmGYiMLTIZkAw8xfy12xafVLbo342UoEcSfd7VcpJV2ZI0q
qwrfkml9K9sYoNRC4t/0nBB8PH+NM9TbPhgd8/roamKPk8j6RYEjVs+9Wx+lbKsExGz8N/h3jkYVieZ
VnuhwFEzU00zUbzQEtaWDlsztTrP+yTF8riDtaYJviNB/qJ0Yee0wXNh0Prpls1skY8yzNnHHlwnnWt
fZUBd2hbgUbaY+SmpKWO0tGqFwaCid3wo3XUxhExzYgnlGT2ISX/GuuuNrQ6D4AY4b7gjEkoIUvS2eB
lemqRVozbRO0yaqyvnMc4dgI2NQ0PUfLSGOM4fcpA8djkgKbBDVWMTST9Y5lCdrUFiS9Kgc+5HdFdRN
TD04vZ+78iMIflfqw1YVat7kq0FbcSr7Ot4jWPmKQG8d/3wJajB1odYlRqlmp7hoaDoL4QqHoL8KDBl
KHWlFXikfbw== JAY@DESKTOP-5NC6FPK
bencito@Server1:~/.ssh$ 
```

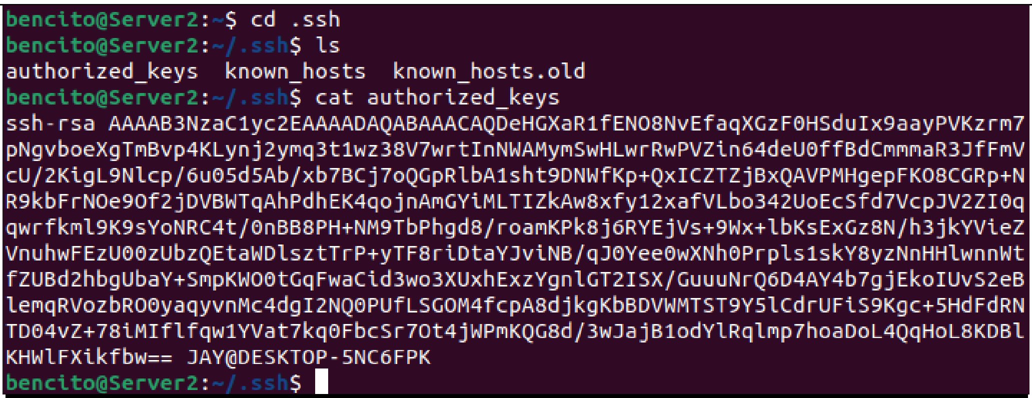


```
JAY@DESKTOP-5NC6FPK MINGW64 ~ (master)
$ cd .ssh

JAY@DESKTOP-5NC6FPK MINGW64 ~/.ssh (master)
$ cat id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAACAQDeHGXaR1fENO8NvEfaqXGzF0HSduIx9aayPVKzrm7p
NgvboeXgTmBvp4KLynj2ymq3t1wz38V7wrtInNWAMymSwHLwrRwPVZin64deU0ffBdCmmmaR3JfFmVcU
/2KigL9Nlcp/6u05d5Ab/xb7BCj7oQGpRlbA1sht9DNWfKp+QxICZTZjBxQAVPMHgepFKO8CGRp+NR9k
bFrNOe9Of2jDVBWTqAhPdhEK4qojnAmGYiMLTIZkAw8xfy12xafVLbo342UoEcSfd7VcpJV2ZI0qqwrf
kml9K9sYoNRC4t/0nBB8PH+NM9TbPhgd8/roamKPk8j6RYEjVs+9Wx+lbKsExGz8N/h3jkYVieZVnuhw
FEzU00zUbzQEtaWDlsztTrP+yTF8riDtaYJviNB/qJ0Yee0wXNh0Prpls1skY8yzNnHHlwnnWtfZUBd2
hbgUbaY+SmpKWO0tGqFwaCid3wo3XUxhExzYgnlGT2ISX/GuuuNrQ6D4AY4b7gjEkoIUvS2eBlemqRVo
zbRO0yaqyvnMc4dgI2NQ0PUfLSGOM4fcpA8djkgKbBDVWMTST9Y5lCdrUFiS9Kgc+5HdFdRNTD04vZ+7
8iMIflfqw1YVat7kq0FbcSr7Ot4jWPmKQG8d/3wJajB1odYlRqlmp7hoaDoL4QqHoL8KDBlKHWlFXikf
bw== JAY@DESKTOP-5NC6FPK

JAY@DESKTOP-5NC6FPK MINGW64 ~/.ssh (master)
```

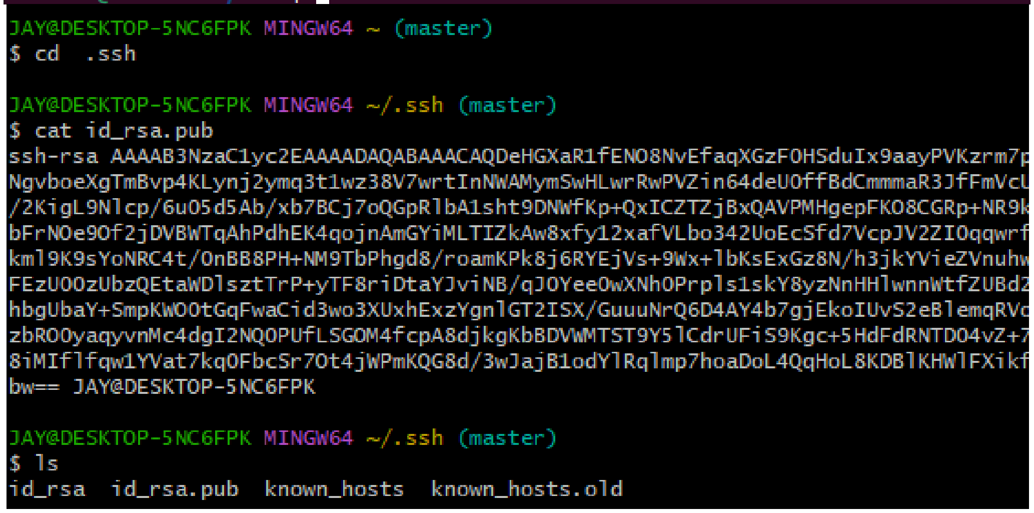
Server 2



```
JAY@DESKTOP-5NC6FPK MINGW64 ~ (master)
$ ssh-copy-id -i ~/.ssh/id_rsa bencito@192.168.56.103
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/c/Users/JAY/.ssh
/id_rsa.pub"
The authenticity of host '192.168.56.103 (192.168.56.103)' can't be established.
ED25519 key fingerprint is SHA256:pFlyJOAG4iwOCRjRElSTBPFODRN7QLQR8PlAwU2C8y4.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? 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 prompt
ed now it is to install the new keys
bencito@192.168.56.103's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'bencito@192.168.56.103'"
and check to make sure that only the key(s) you wanted were added.
```



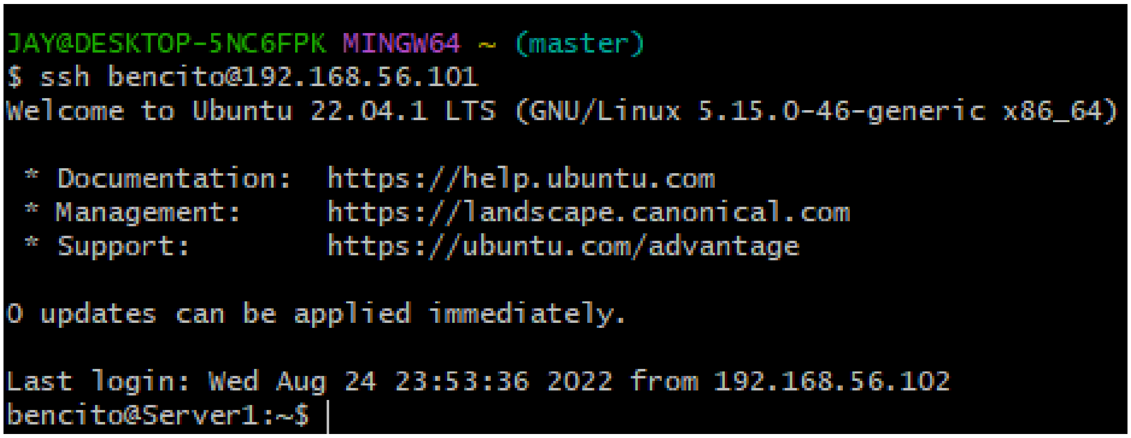
```
bencito@Server2:~$ cd .ssh
bencito@Server2:~/.ssh$ ls
authorized_keys  known_hosts  known_hosts.old
bencito@Server2:~/.ssh$ cat authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAACAQDeHGXaR1fENO8NvEfaqXGzF0HSduIx9aayPVKzrm7
pNgvboeXgTmBvp4KLynj2ymq3t1wz38V7wrtInNWAMymSwHLwrRwPVZin64deU0ffBdCmmmaR3JfFmV
cU/2KigL9Nlcp/6u05d5Ab/xb7BCj7oQGpRlbA1sht9DNWfKp+QxICZTZjBxQAVPMHgepFKO8CGRp+N
R9kbFrNOe9Of2jDVBWTqAhPdhEK4qojnAmGYiMLTIZkAw8xfy12xafVLbo342UoEcSfd7VcpJV2ZI0q
qwrfkml9K9sYoNRC4t/0nBB8PH+NM9TbPhgd8/roamKPk8j6RYEjVs+9Wx+lbKsExGz8N/h3jkYVieZ
VnuhwFEzU00zUbzQEtaWDlsztTrP+yTF8riDtaYJviNB/qJ0Yee0wXNh0Prpls1skY8yzNnHHlwnnWt
fZUBd2hbgUbaY+SmpKWO0tGqFwaCid3wo3XUxhExzYgnlGT2ISX/GuuuNrQ6D4AY4b7gjEkoIUvS2eB
lemqRVozbRO0yaqyvnMc4dgI2NQ0PUfLSGOM4fcpA8djkgKbBDVWMTST9Y5lCdrUFiS9Kgc+5HdFdRN
TD04vZ+78iMIflfqw1YVat7kq0FbcSr7Ot4jWPmKQG8d/3wJajB1odYlRqlmp7hoaDoL4QqHoL8KDBl
KHWlFXikfbw== JAY@DESKTOP-5NC6FPK
bencito@Server2:~/.ssh$
```



```
JAY@DESKTOP-5NC6FPK MINGW64 ~ (master)
$ cd  .ssh

JAY@DESKTOP-5NC6FPK MINGW64 ~/.ssh (master)
$ cat id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAACAQDeHGXaR1fENO8NvEfaqXGzFOHSduIx9aayPVKzrm7p
NgvboeXgTmBvp4KLynj2ymq3t1wz38V7wrtInNWAMymSwHLwrRwPVZin64deUOffBdCmmmaR3JfFmVcU
/2KigL9Nlcp/6u05d5Ab/xb7BCj7oQGpRlbA1sht9DNWfKp+QxICZTZjBxQAVPMHgepFKO8CGRp+NR9k
bFrNOe9Of2jDVBWTqAhPdhEK4qojnAmGYiMLTIZkAw8xfy12xafVLbo342UoEcSfd7VcpJV2ZIOqqwrf
kml9K9sYoNRC4t/OnBB8PH+NM9TbPhgd8/roamKPk8j6RYEjVs+9Wx+lbKsExGz8N/h3jkYVieZVnuhw
FEzUOOzUbzQEtaWDlsztTrP+yTF8riDtaYJviNB/qJOYeeOwXNhOPrpls1skY8yzNnHHlwnnWtfZUBd2
hbgUbaY+SmpKWOOtGqFwaCid3wo3XUxhExzYgnlGT2ISX/GuuuNrQ6D4AY4b7gjEkoIUvS2eBlemqRVo
zbROOyaqyvnMc4dgI2NQOPUfLSGOM4fcpA8djkgKbBDVWMTST9Y5lCdrUFiS9Kgc+5HdFdRNTDO4vZ+7
8iMIflfqw1YVat7kqOFbcSr7Ot4jWPmKQG8d/3wJajB1odYlRqlmp7hoaDoL4QqHoL8KDBlKHWlFXikf
bw== JAY@DESKTOP-5NC6FPK

JAY@DESKTOP-5NC6FPK MINGW64 ~/.ssh (master)
$ ls
id_rsa  id_rsa.pub  known_hosts  known_hosts.old
```

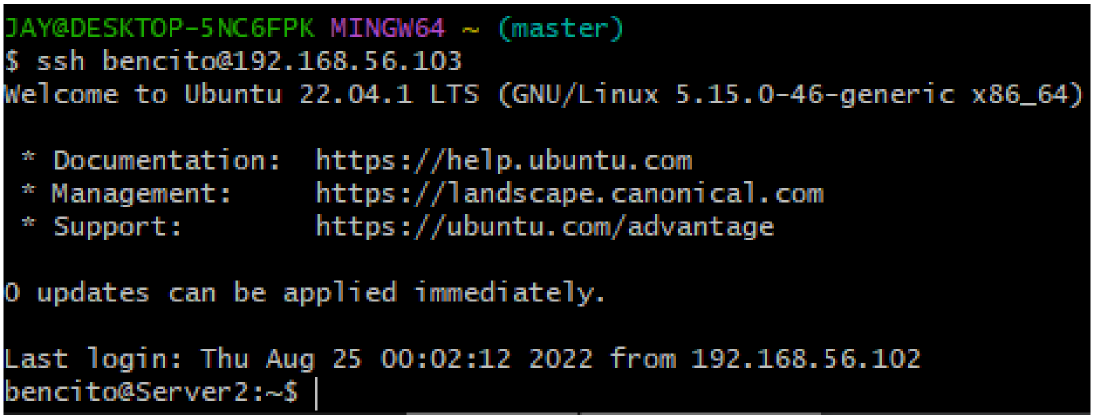
3. Once the public key has been configured on the server, the server will allow any connecting user that has the private key to log in. During the login process, the client proves possession of the private key by digitally signing the key exchange.
4. On the local machine, verify that you can SSH with Server 1 and Server 2. What did you notice? Did the connection ask for a password? If not, why?

Server 1



```
JAY@DESKTOP-5NC6FPK MINGW64 ~ (master)
$ ssh bencito@192.168.56.101
Welcome to Ubuntu 22.04.1 LTS (GNU/Linux 5.15.0-46-generic x86_64)

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

0 updates can be applied immediately.

Last login: Wed Aug 24 23:53:36 2022 from 192.168.56.102
bencito@Server1:~$
```

Server 2



```
JAY@DESKTOP-5NC6FPK MINGW64 ~ (master)
$ ssh bencito@192.168.56.103
Welcome to Ubuntu 22.04.1 LTS (GNU/Linux 5.15.0-46-generic x86_64)

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

0 updates can be applied immediately.

Last login: Thu Aug 25 00:02:12 2022 from 192.168.56.102
bencito@Server2:~$ |
```

Reflections:

Answer the following:

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

   Ssh is like a private encryptment tool that helps you to secure your network with the key.

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

   You just need to go to your root ssh to locate the private keypair.

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**).

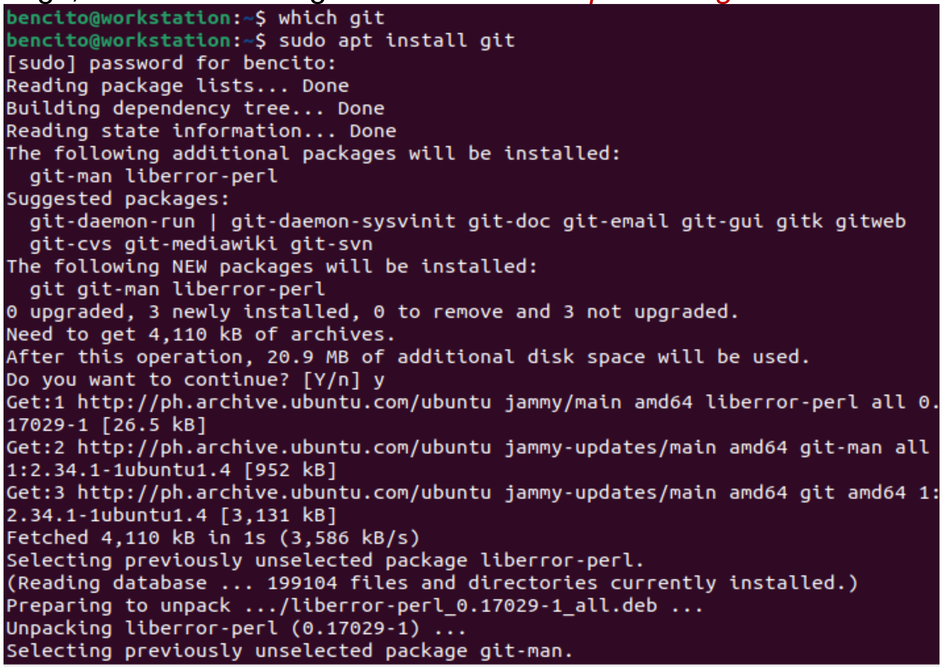
Set up Git

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

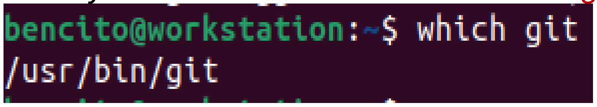
Task 3: Set up the Git Repository

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*



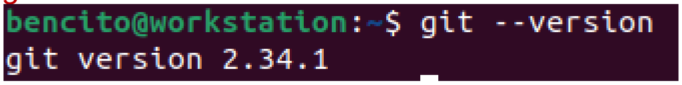
```
bencito@workstation:~$ which git
bencito@workstation:~$ sudo apt install git
[sudo] password for bencito:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  git-man liberror-perl
Suggested packages:
  git-daemon-run | git-daemon-sysvinit git-doc git-email git-gui gitk 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 3 not upgraded.
Need to get 4,110 kB of archives.
After this operation, 20.9 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 all 0.
17029-1 [26.5 kB]
Get:2 http://ph.archive.ubuntu.com/ubuntu jammy-updates/main amd64 git-man all
1:2.34.1-1ubuntu1.4 [952 kB]
Get:3 http://ph.archive.ubuntu.com/ubuntu jammy-updates/main amd64 git amd64 1:
2.34.1-1ubuntu1.4 [3,131 kB]
Fetched 4,110 kB in 1s (3,586 kB/s)
Selecting previously unselected package liberror-perl.
(Reading database ... 199104 files and directories currently installed.)
Preparing to unpack .../liberror-perl_0.17029-1_all.deb ...
Unpacking liberror-perl (0.17029-1) ...
Selecting previously unselected package git-man.
```

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



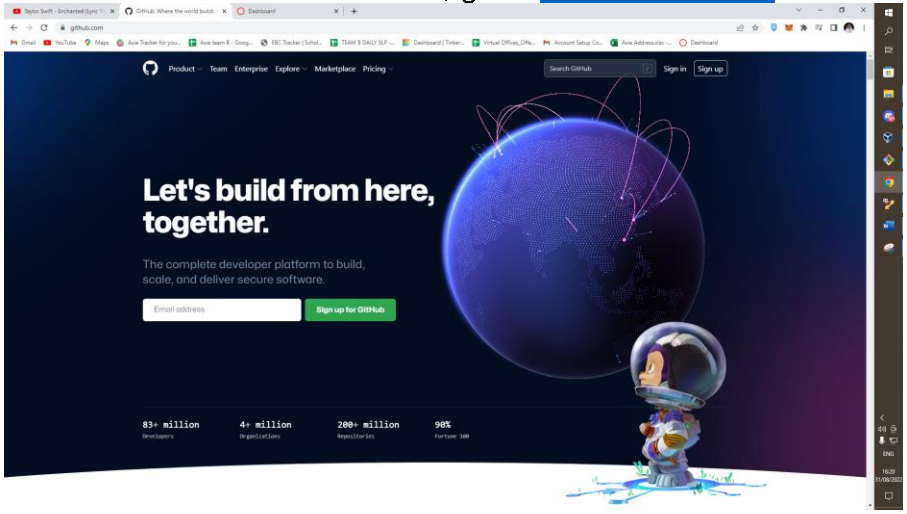
```
bencito@workstation:~$ which git
/usr/bin/git
```

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



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

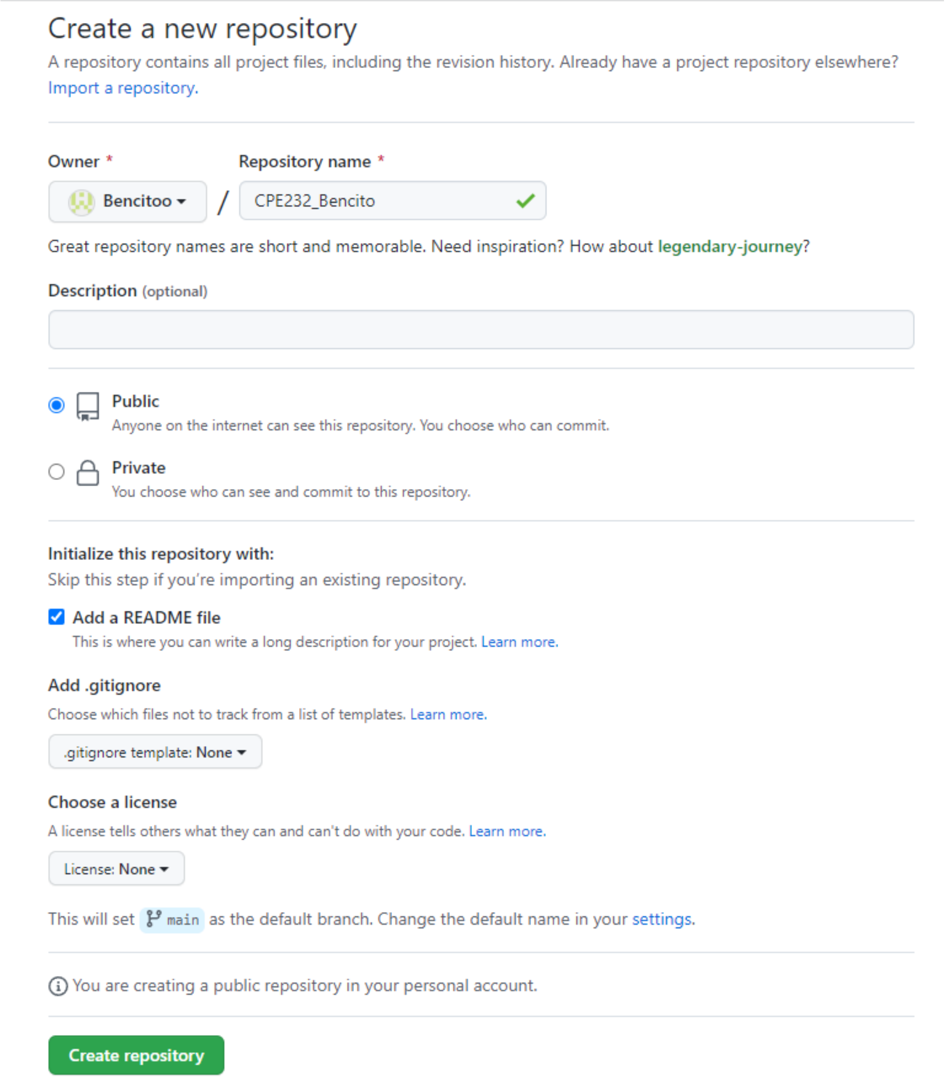
4. Using the browser in the local machine, go to www.github.com.



Let's build from here, together.

The complete developer platform to build, scale, and deliver secure software.

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.

Owner * Repository name *

Bencitoo ▾ / CPE232_Bencito ✓

Great repository names are short and memorable. Need inspiration? How about legendary-journey?

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:
Skip this step if you're importing an existing repository.

☑ Add a README file
This is where you can write a long description for your project. Learn more.

Add .gitignore
Choose which files not to track from a list of templates. Learn more.

.gitignore template: None ▾

Choose a license
A license tells others what they can and can't do with your code. Learn more.

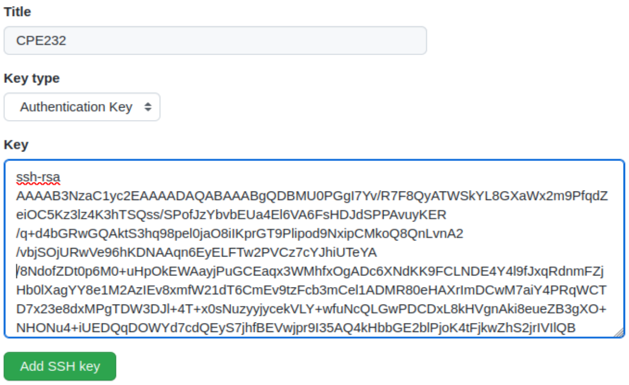
License: None ▾

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

ⓘ You are creating a public repository in your personal account.

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.



Title

CPE232

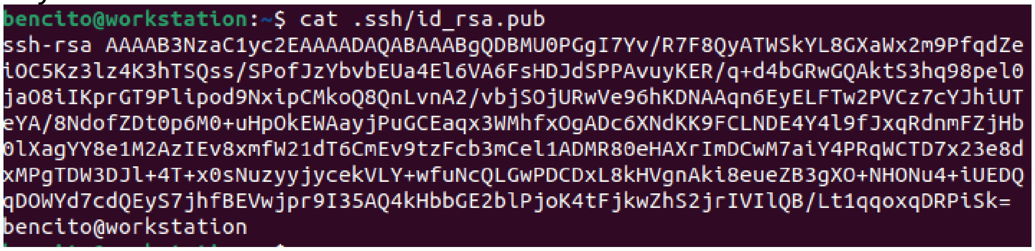
Key type

Authentication Key

Key

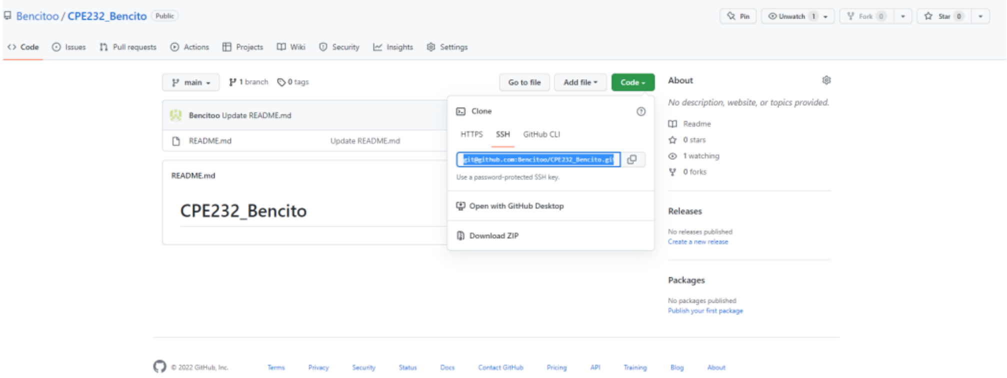
```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABgQDBMU0PGgI7Yv/R7F8QyATWSkYL8GXaWx2m9PfqdZ
eiOC5Kz3lz4K3hTSQss/SPofJzYbvbEUa4El6VA6FsHDJdSPPAvuyKER
/q+d4bGRwGQAktS3hq98pel0jaO8iIKprGT9Plipod9NxipCMkoQ8QnLvnA2
/vbjSOjURwVe96hKDNAAqn6EyELFTw2PVCz7cYJhiUTeYA
/8NdofZDt0p6M0+uHpOkEWAayjPuGCEaqx3WMhfxOgADc6XNdKK9FCLNDE4Y4l9fJxqRdnmFZj
Hb0lXagYY8e1M2AzIEv8xmfW21dT6CmEv9tzFcb3mCel1ADMR80eHAXrImDCwM7aiY4PRqWCT
D7x23e8dxMPgTDW3DJl+4T+x0sNuzyyjycekVLY+wfuNcQLGwPDCDxL8kHVgnAki8eueZB3gXO+
NHONu4+iUEDQqDOWYd7cdQEyS7jhfBEVwjpr9I35AQ4kHbbGE2blPjoK4tFjkwZhS2jrIVIlQB
```

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.



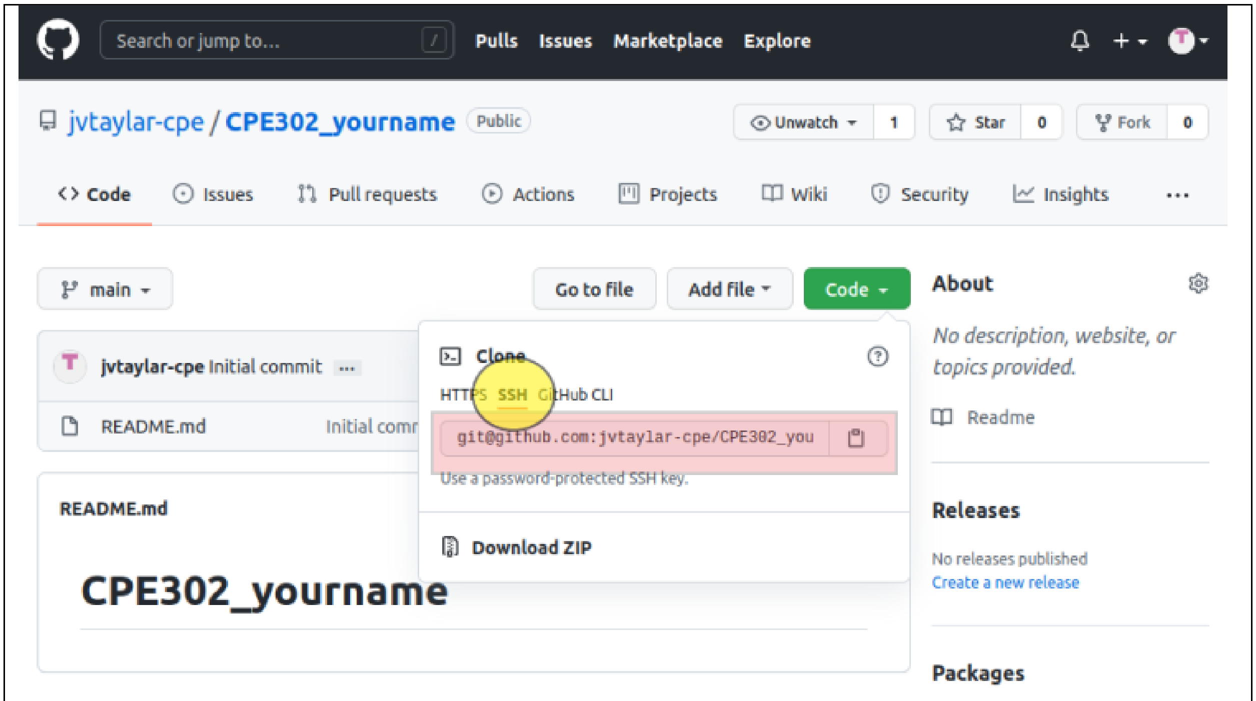
```
bencito@workstation:~$ cat .ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQDBMU0PGgI7Yv/R7F8QyATWSkYL8GXaWx2m9PfqdZe
iOC5Kz3lz4K3hTSQss/SPofJzYbvbEUa4El6VA6FsHDJdSPPAvuyKER/q+d4bGRwGQAktS3hq98pel0
jaO8iIKprGT9Plipod9NxipCMkoQ8QnLvnA2/vbjSOjURwVe96hKDNAAqn6EyELFTw2PVCz7cYJhiUT
eYA/8NdofZDt0p6M0+uHpOkEWAayjPuGCEaqx3WMhfxOgADc6XNdKK9FCLNDE4Y4l9fJxqRdnmFZjHb
0lXagYY8e1M2AzIEv8xmfW21dT6CmEv9tzFcb3mCel1ADMR80eHAXrImDCwM7aiY4PRqWCTD7x23e8d
xMPgTDW3DJl+4T+x0sNuzyyjycekVLY+wfuNcQLGwPDCDxL8kHVgnAki8eueZB3gXO+NHONu4+iUEDQ
qDOWYd7cdQEyS7jhfBEVwjpr9I35AQ4kHbbGE2blPjoK4tFjkwZhS2jrIVIlQB/Lt1qqoxqDRPiSk=
bencito@workstation
```

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:Bencitoo/CPE232_Bencito.gi
```

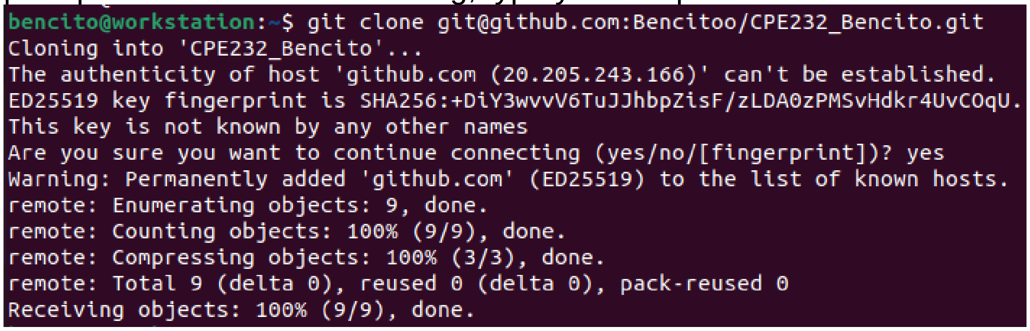
Use a password-protected SSH key.



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

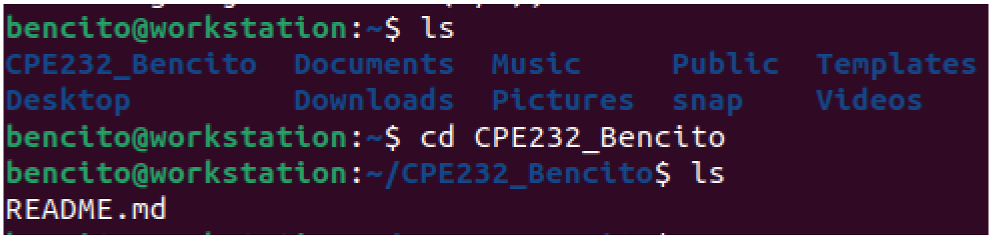
Use a password-protected SSH key.

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.



```
bencito@workstation:~$ git clone git@github.com:Bencitoo/CPE232_Bencito.git
Cloning into 'CPE232_Bencito'...
The authenticity of host 'github.com (20.205.243.166)' can't be established.
ED25519 key fingerprint is SHA256:+DiY3wvvV6TuJJhbpZisF/zLDA0zPMSvHdkr4UvCOqU.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'github.com' (ED25519) to the list of known hosts.
remote: Enumerating objects: 9, done.
remote: Counting objects: 100% (9/9), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 9 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (9/9), 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.

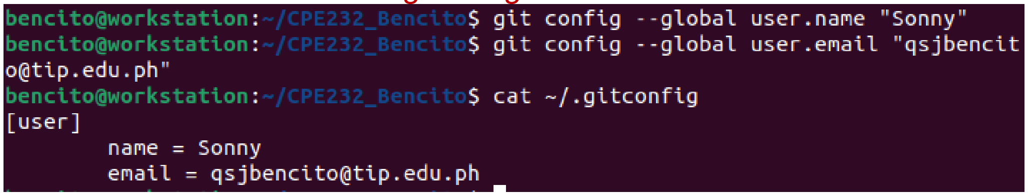


```
bencito@workstation:~$ ls
CPE232_Bencito  Documents  Music     Public  Templates
Desktop         Downloads  Pictures  snap    Videos
bencito@workstation:~$ cd CPE232_Bencito
bencito@workstation:~/CPE232_Bencito$ ls
README.md
```

g. Use the following commands to personalize your git.

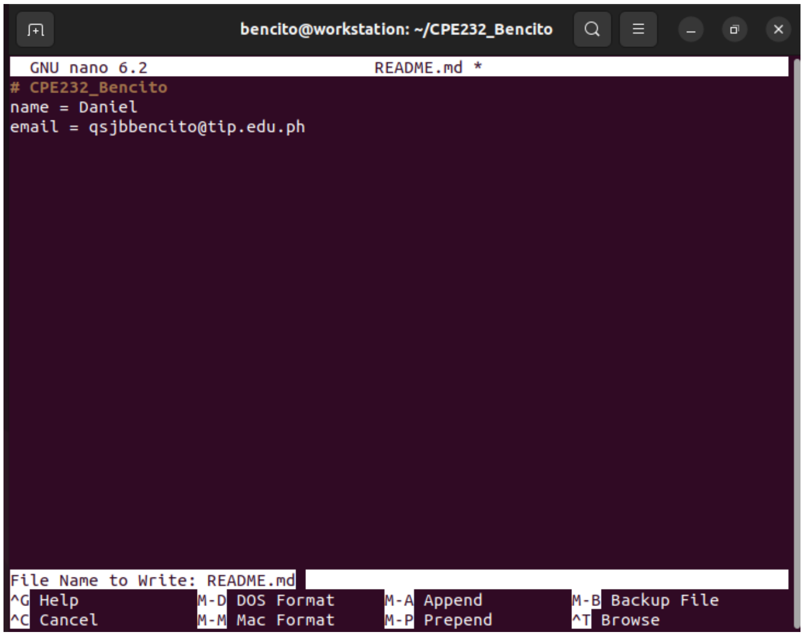
- *git config --global user.name “Your Name”*
- *git config --global user.email yourname@email.com*

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



```
bencito@workstation:~/CPE232_Bencito$ git config --global user.name "Sonny"
bencito@workstation:~/CPE232_Bencito$ git config --global user.email "qsjbencit
o@tip.edu.ph"
bencito@workstation:~/CPE232_Bencito$ cat ~/.gitconfig
[user]
        name = Sonny
        email = qsjbencito@tip.edu.ph
```

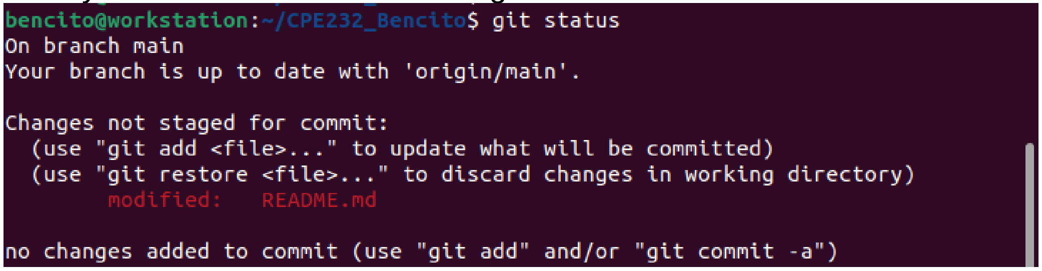
h. Edit the README.md file using nano command. Provide any information on the markdown file pertaining to the repository you created. Make sure to write out or save the file and exit.



bencito@workstation: ~/CPE232_Bencito

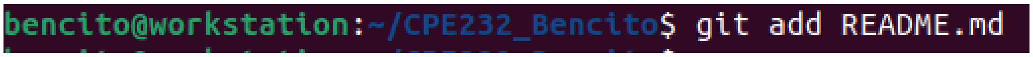
```
  GNU nano 6.2                         README.md *
# CPE232_Bencito
name = Daniel
email = qsjbbencito@tip.edu.ph










File Name to Write: README.md
^G Help              M-D DOS Format       M-A Append           M-B Backup File
^C Cancel            M-M Mac Format       M-P Prepend          ^T Browse
```

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?



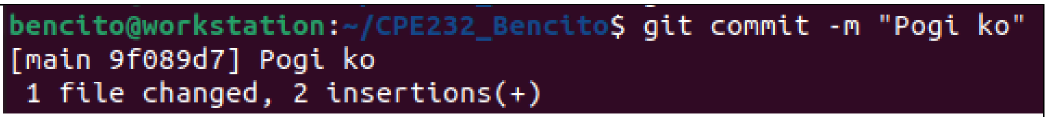
```
bencito@workstation:~/CPE232_Bencito$ git status
On branch main
Your branch is up to date with 'origin/main'.

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")
```

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



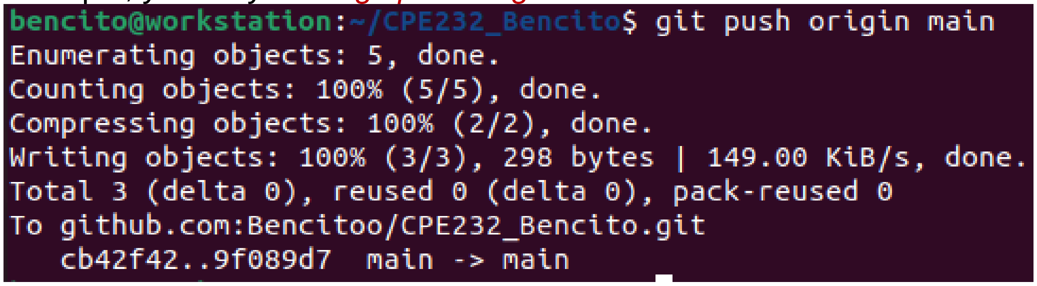
```
bencito@workstation:~/CPE232_Bencito$ git add README.md
```

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.



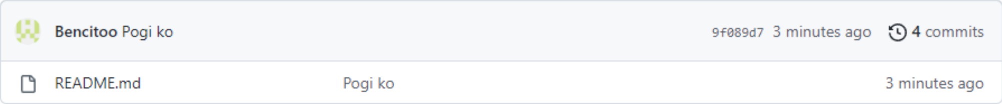
```
bencito@workstation:~/CPE232_Bencito$ git commit -m "Pogi ko"
[main 9f089d7] Pogi ko
 1 file changed, 2 insertions(+)
```

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.*



```
bencito@workstation:~/CPE232_Bencito$ git push origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 298 bytes | 149.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:Bencitoo/CPE232_Bencito.git
   cb42f42..9f089d7  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.



Bencitoo Pogi ko		9f089d7 3 minutes ago 4 commits
README.md	Pogi ko	3 minutes ago

Reflections:

Answer the following:

3. What sort of things have we so far done to the remote servers using ansible commands?

   I maintaining the remote servers and I do some network key that help does server to be private.

4. How important is the inventory file?

   It is important because it can list all the individual host or that helps you to locate the different inventory files in your system.

Conclusions/Learnings:

SSH is very important because it helps you to remote with your own pc. It also helps to protect your data because of the generate keys. I learn in this activity to how generate and control the server's using git and ubuntu. Sometimes I do some errors but when I finally know it, I feel happy. I only need to familiarize some code that need.

I affirm that I shall not give or receive any unauthorized help on this assignment and that all work shall be my own.