Name: Atienza, Stephen Lhaye C.	**Date Performed:01/22/24**
Course/Section:CPE231S1	**Date Submitted:01/23/24**
Instructor: Dr. Jonathan V. Taylar	**Semester and SY: 2nd and 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.

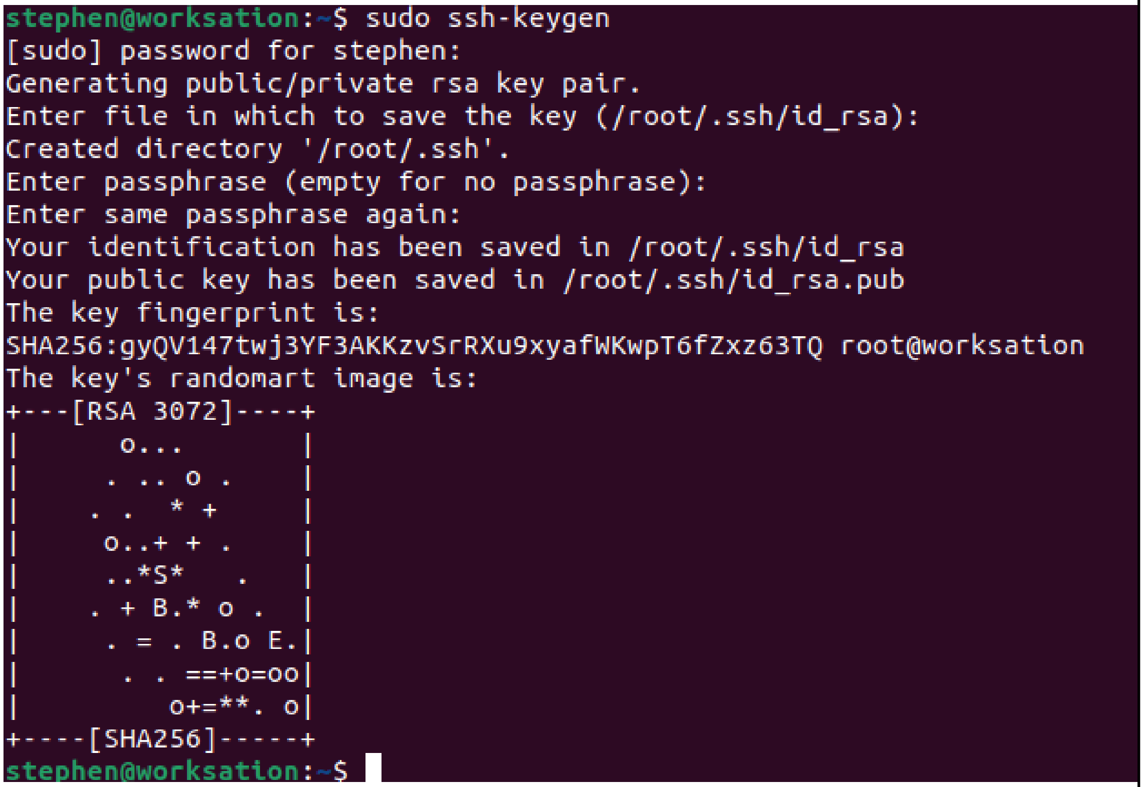
SSH introduced public key authentication as a more secure alternative to the older .rhosts authentication. It improved security by avoiding the need to have password stored in files and eliminated the possibility of a compromised server stealing the user's password.

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

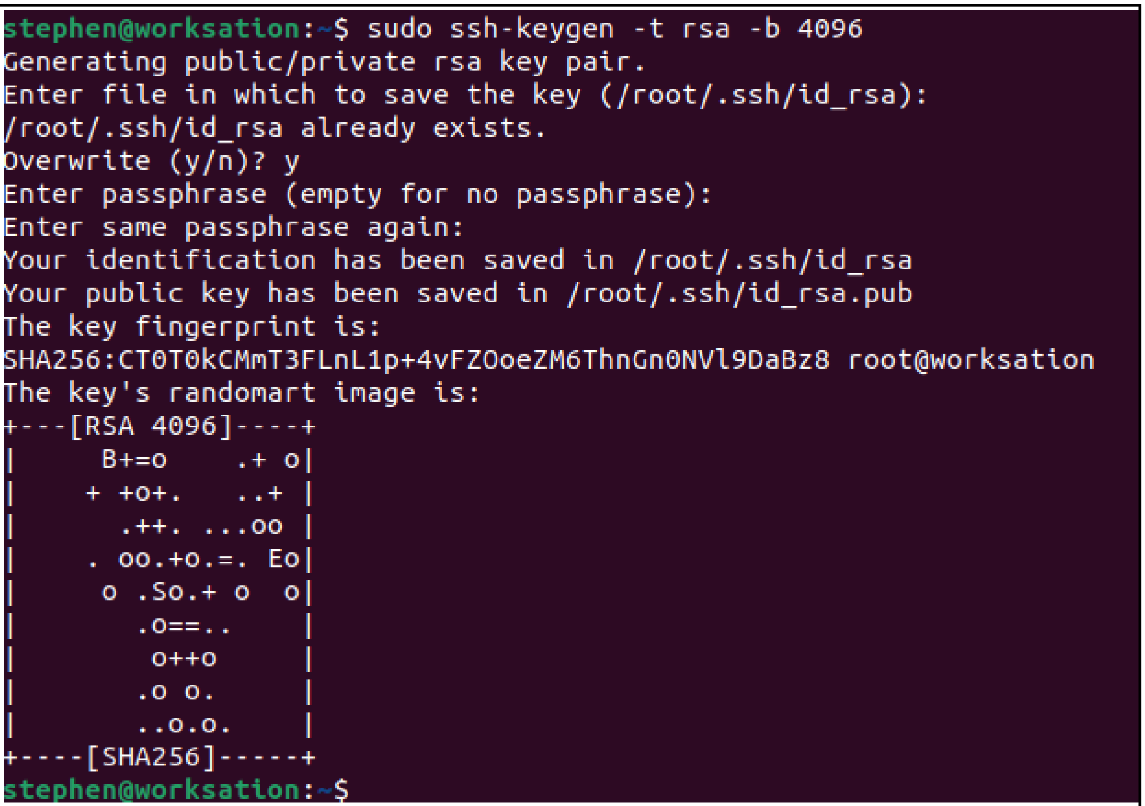
1. The simplest way to generate a key pair is to run *ssh-keygen* without arguments. In this case, it will prompt for the file in which to store keys. First,

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*.



```
stephen@worksation:~$ sudo ssh-keygen
[sudo] password for stephen:
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Created directory '/root/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa
Your public key has been saved in /root/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:gyQV147twj3YF3AKKzvSrRXu9xyafWKwpT6fZxz63TQ root@worksation
The key's randomart image is:
+---[RSA 3072]----+
|      o...       |
|     . .. o .    |
|    . .  * +     |
|     o..+ + .    |
|     ..*S*   .   |
|    . + B.* o .  |
|     . = . B.o E.|
|      . . ==+o=oo|
|         o+=**. o|
+----[SHA256]-----+
stephen@worksation:~$
```

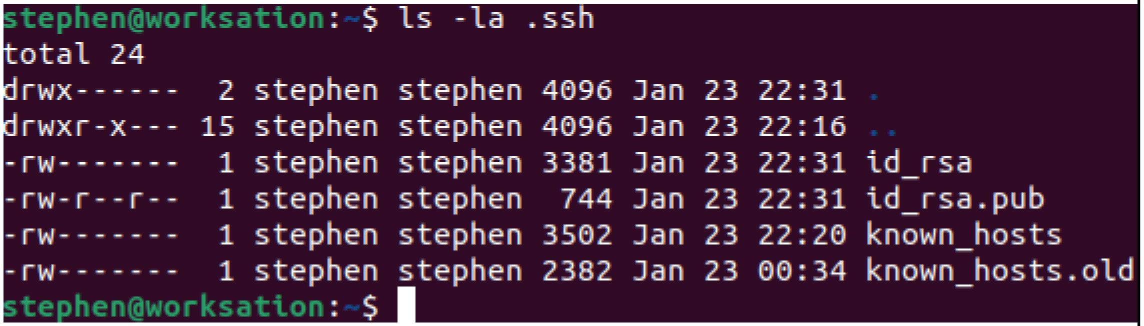
2. Issue the command *ssh-keygen -t rsa -b 4096.* The algorithm is selected using the -t option and key size using the -b option.



```
stephen@worksation:~$ sudo ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
/root/.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 /root/.ssh/id_rsa
Your public key has been saved in /root/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:CT0T0kCMmT3FLnL1p+4vFZOoeZM6ThnGn0NVl9DaBz8 root@worksation
The key's randomart image is:
+---[RSA 4096]----+
|     B+=o    .+ o|
|    + +o+.   ..+ |
|      .++. ...oo |
|    . oo.+o.=. Eo|
|     o .So.+ o  o|
|       .o==..    |
|        o++o     |
|       .o o.     |
|       ..o.o.    |
+----[SHA256]-----+
stephen@worksation:~$
```

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.

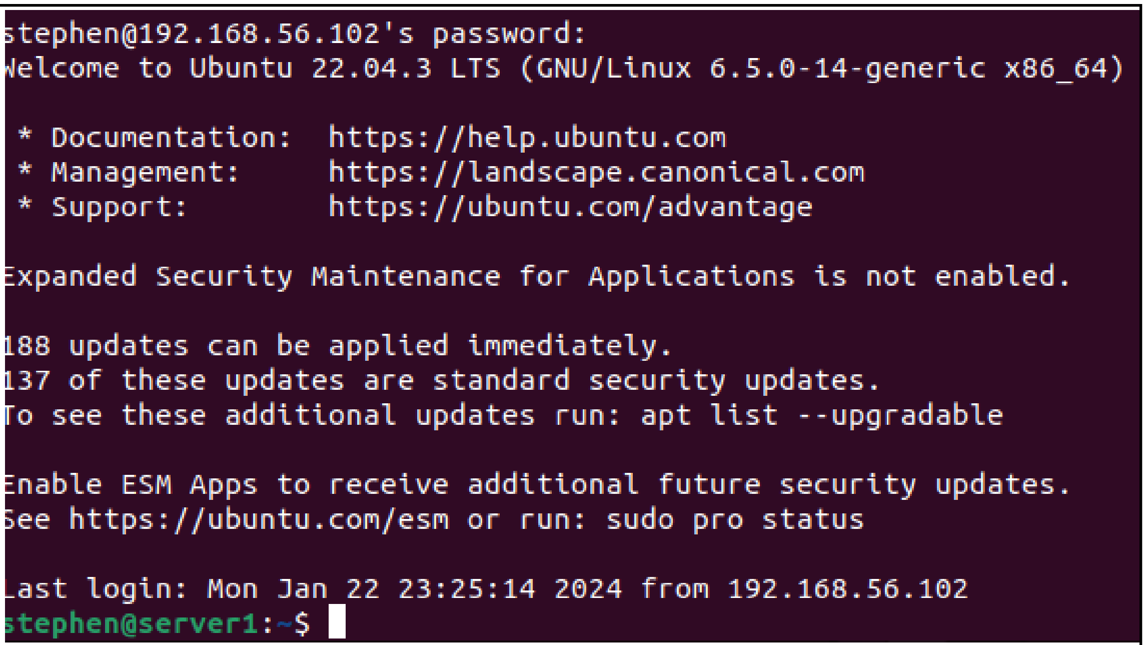
4. Verify that you have created the key by issuing the command *ls -la .ssh.* The command should show the .ssh directory containing a pair of keys. For example, id_rsa.pub and id_rsa.



```
stephen@worksation:~$ ls -la .ssh
total 24
drwx------  2 stephen stephen 4096 Jan 23 22:31 .
drwxr-x--- 15 stephen stephen 4096 Jan 23 22:16 ..
-rw-------  1 stephen stephen 3381 Jan 23 22:31 id_rsa
-rw-r--r--  1 stephen stephen  744 Jan 23 22:31 id_rsa.pub
-rw-------  1 stephen stephen 3502 Jan 23 22:20 known_hosts
-rw-------  1 stephen stephen 2382 Jan 23 00:34 known_hosts.old
stephen@worksation:~$
```

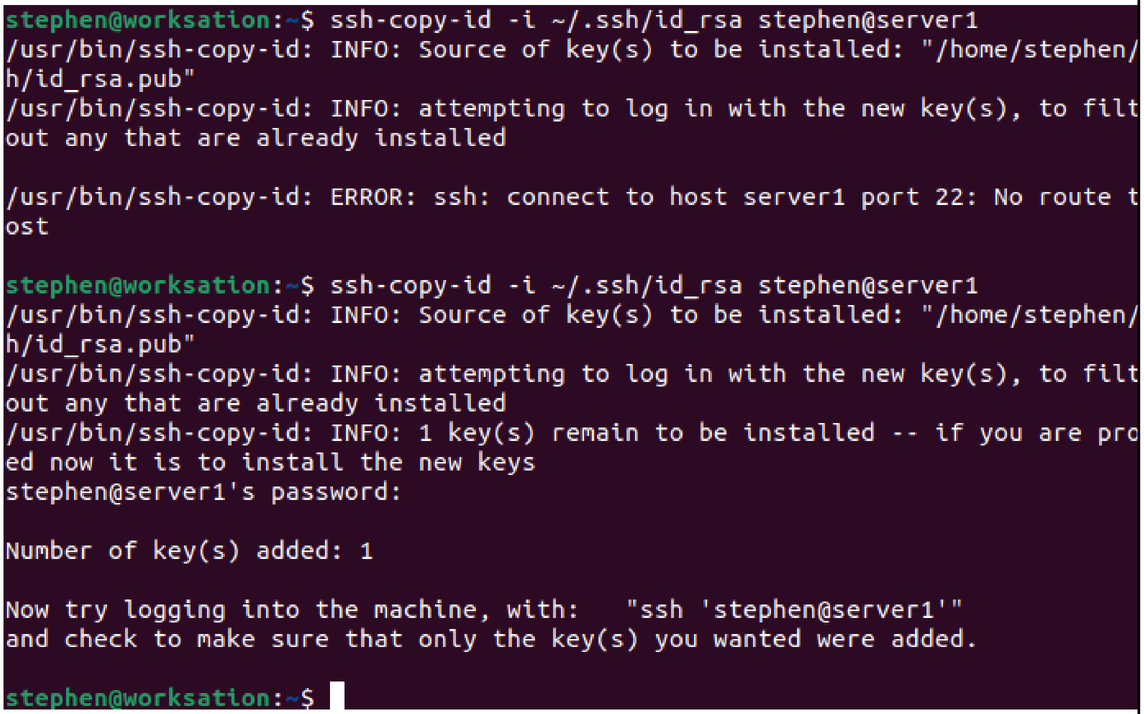
Task 2: Copying the Public Key to the remote servers

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.



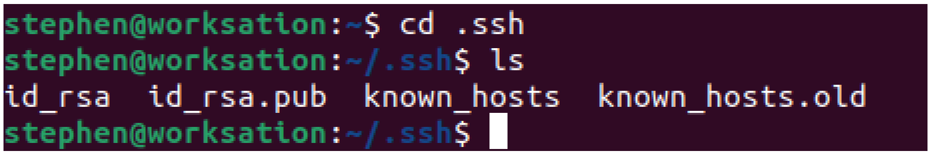
```
stephen@192.168.56.102's password:
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 6.5.0-14-generic x86_64)

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

Expanded Security Maintenance for Applications is not enabled.

188 updates can be applied immediately.
137 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

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

Last login: Mon Jan 22 23:25:14 2024 from 192.168.56.102
stephen@server1:~$
```

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



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

/usr/bin/ssh-copy-id: ERROR: ssh: connect to host server1 port 22: No route t
ost

stephen@worksation:~$ ssh-copy-id -i ~/.ssh/id_rsa stephen@server1
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/stephen/
h/id_rsa.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filt
out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are pro
ed now it is to install the new keys
stephen@server1's password:

Number of key(s) added: 1

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

stephen@worksation:~$
```

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.



```
stephen@worksation:~$ cd .ssh
stephen@worksation:~/.ssh$ ls
id_rsa  id_rsa.pub  known_hosts  known_hosts.old
stephen@worksation:~/.ssh$
```

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?

   It works to connect server 1 and 2

Reflections:

Answer the following:

1. How will you describe the ssh-program? What does it do?
   - Secure Shell (SSH) is a way to securely connect to another computer from your own. It's similar to a private phone connection in that it encrypts all of your communications (including instructions and data) as you control the remote computer from a distance.

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

   - the terminal by using the following command: cat ~/.ssh/authorized_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**).

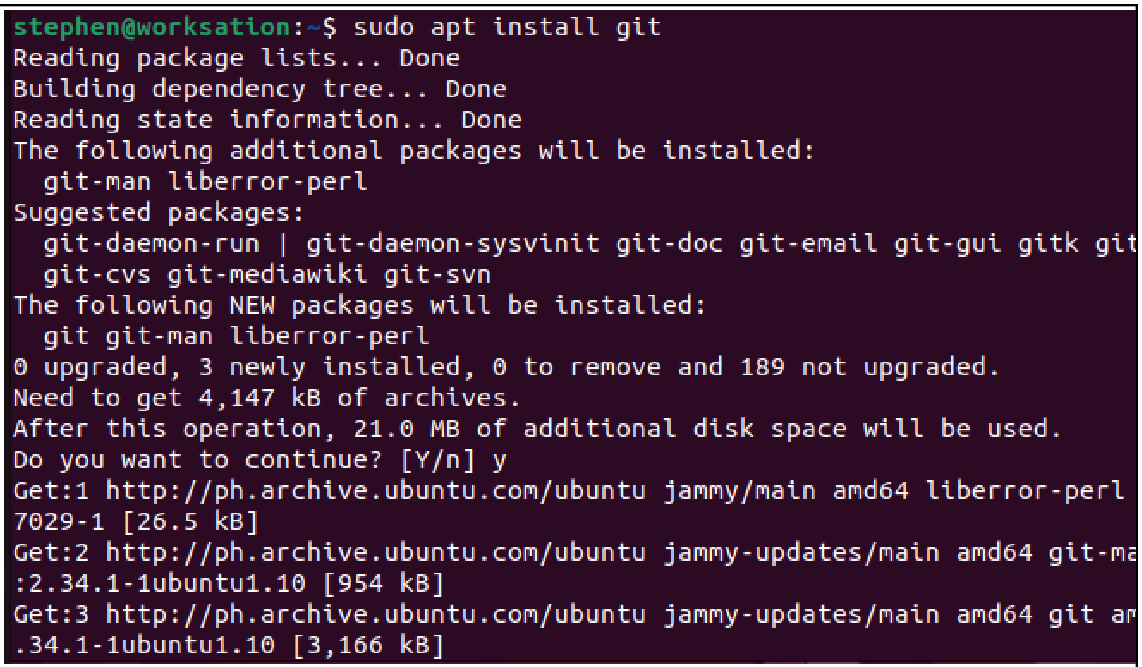
Set up Git

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

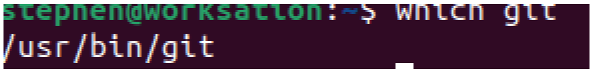
Task 3: Set up the Git Repository

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*



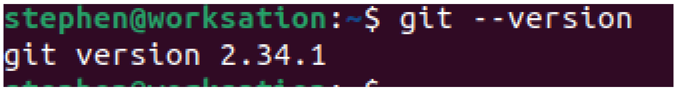
```
stephen@worksation:~$ sudo apt install git
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  git-man liberror-perl
Suggested packages:
  git-daemon-run | git-daemon-sysvinit git-doc git-email git-gui 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 189 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*.



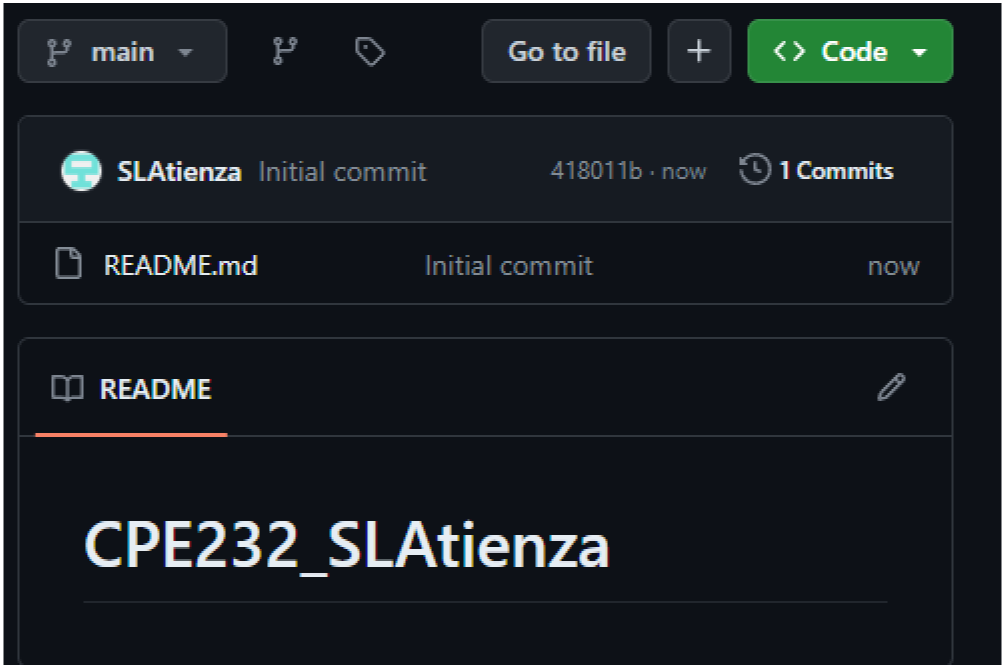
```
stephen@worksation:~$ which git
/usr/bin/git
```

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



```
stephen@worksation:~$ git --version
git version 2.34.1
```

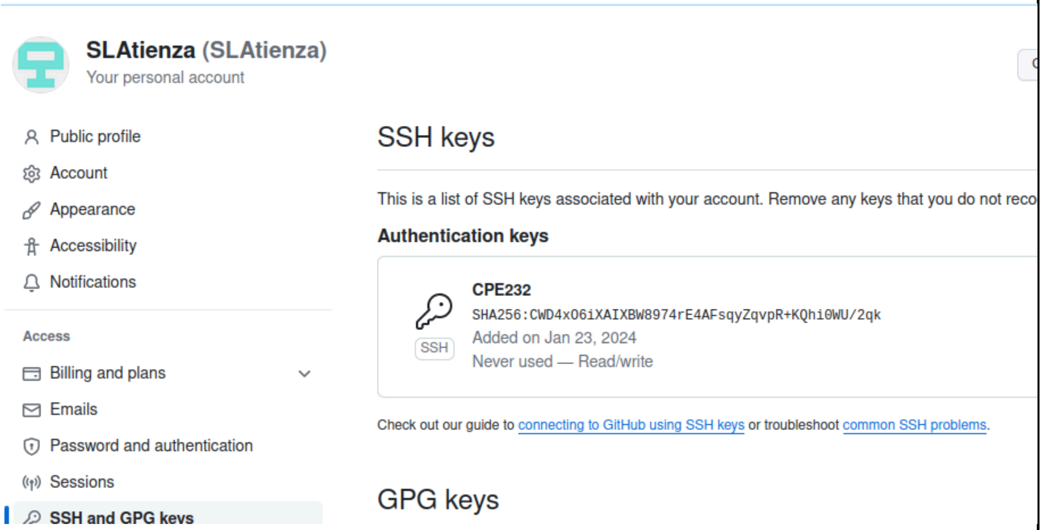
4. Using the browser in the local machine, go to www.github.com.
5. Sign up in case you don't have an account yet. Otherwise, login to your GitHub account.
    a. Create a new repository and name it as CPE232_yourname. Check Add a README file and click Create repository.



CPE232_SLAtienza

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.



SSH keys

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

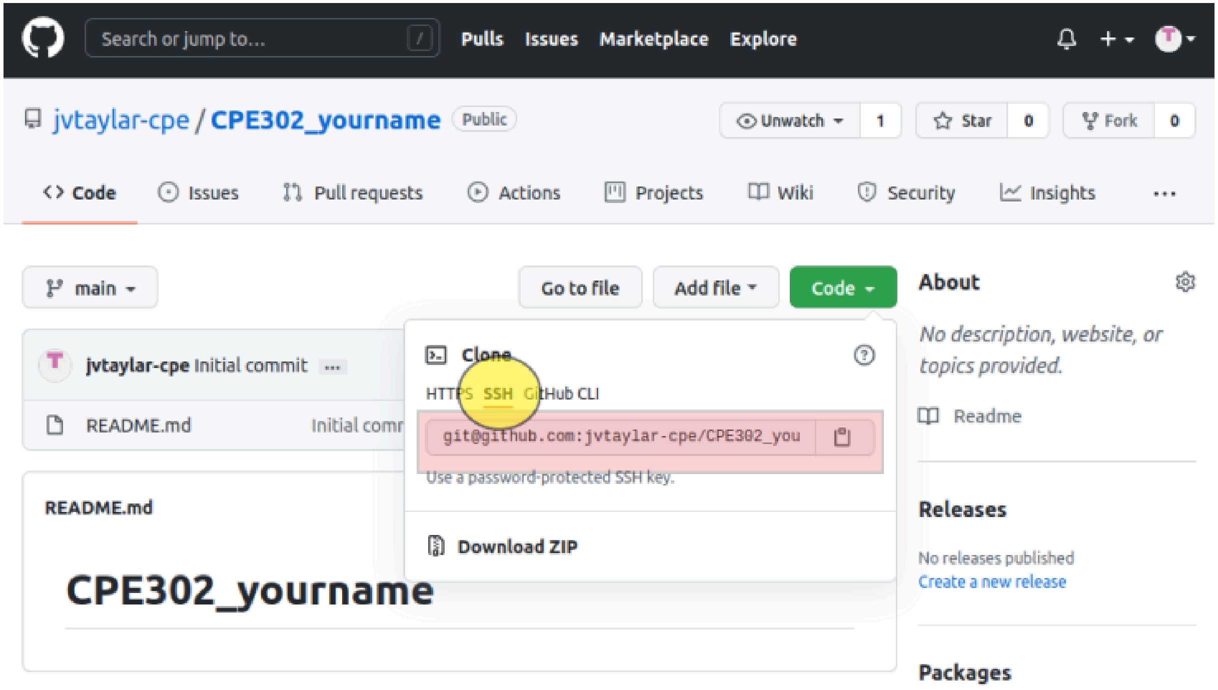
Authentication keys

CPE232
SHA256:CWD4x06iXAIXBW8974rE4AFsqyZqvpR+KQhi0WU/2qk
Added on Jan 23, 2024
Never used — Read/write

Check out our guide to connecting to GitHub using SSH keys or troubleshoot common SSH problems.

GPG keys

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.

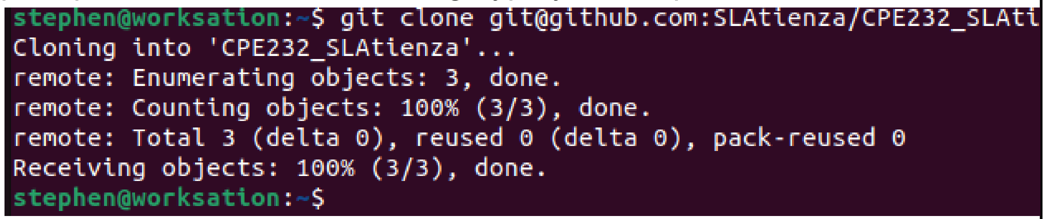


git@github.com:jvtaylar-cpe/CPE302_you

Use a password-protected SSH key.

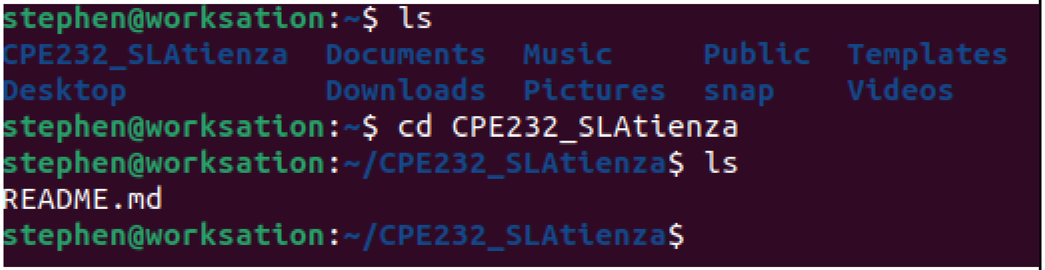
No description, website, or topics provided.

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.



```
stephen@worksation:~$ git clone git@github.com:SLAtienza/CPE232_SLAti
Cloning into 'CPE232_SLAtienza'...
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.
stephen@worksation:~$
```

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.

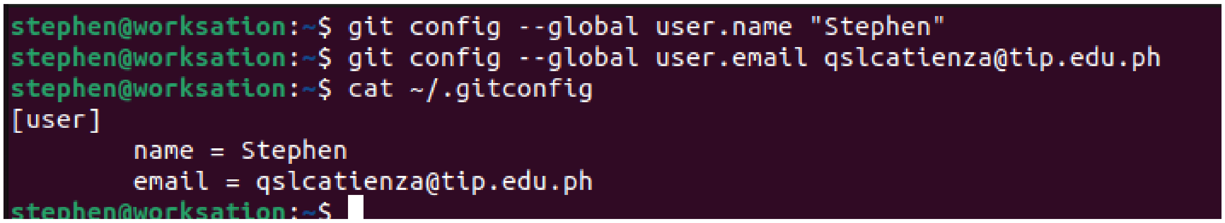


```
stephen@worksation:~$ ls
CPE232_SLAtienza  Documents  Music     Public  Templates
Desktop           Downloads  Pictures  snap    Videos
stephen@worksation:~$ cd CPE232_SLAtienza
stephen@worksation:~/CPE232_SLAtienza$ ls
README.md
stephen@worksation:~/CPE232_SLAtienza$
```

g. Use the following commands to personalize your git.

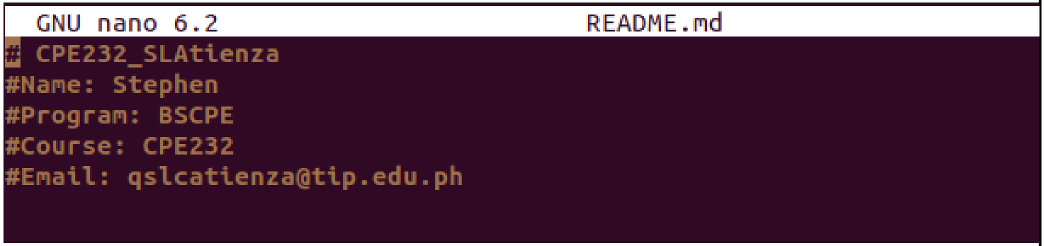
- *git config --global user.name "Your Name"*
- *git config --global user.email yourname@email.com*

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



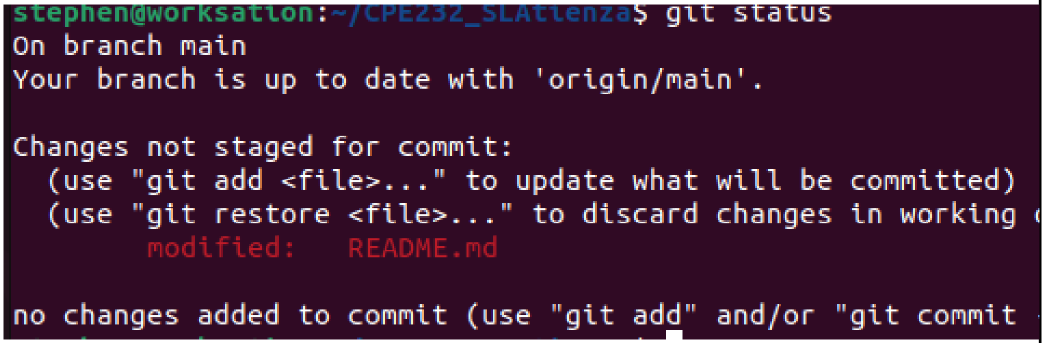
```
stephen@worksation:~$ git config --global user.name "Stephen"
stephen@worksation:~$ git config --global user.email qslcatienza@tip.edu.ph
stephen@worksation:~$ cat ~/.gitconfig
[user]
        name = Stephen
        email = qslcatienza@tip.edu.ph
stephen@worksation:~$
```

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



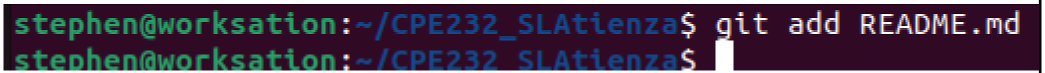
```
  GNU nano 6.2                                    README.md
# CPE232_SLAtienza
#Name: Stephen
#Program: BSCPE
#Course: CPE232
#Email: qslcatienza@tip.edu.ph
```

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?



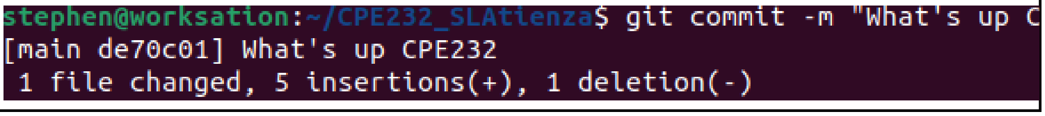
```
stephen@worksation:~/CPE232_SLAtienza$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working 
        modified:   README.md

no changes added to commit (use "git add" and/or "git commit 
```

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



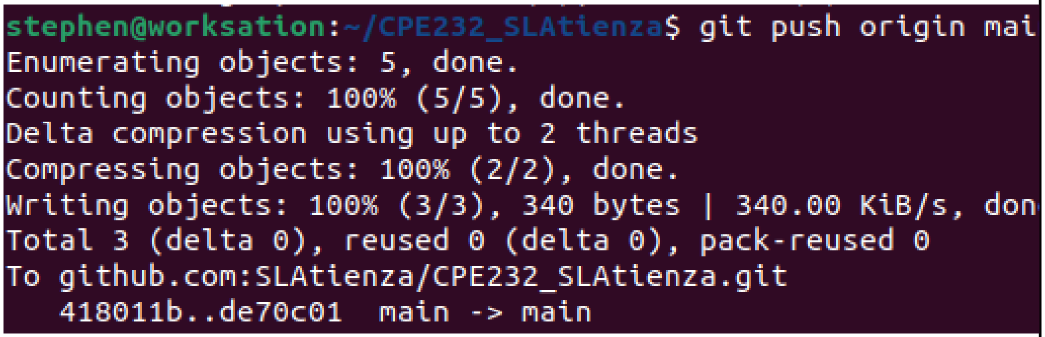
```
stephen@worksation:~/CPE232_SLAtienza$ git add README.md
stephen@worksation:~/CPE232_SLAtienza$
```

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.



```
stephen@worksation:~/CPE232_SLAtienza$ git commit -m "What's up C
[main de70c01] What's up CPE232
 1 file changed, 5 insertions(+), 1 deletion(-)
```

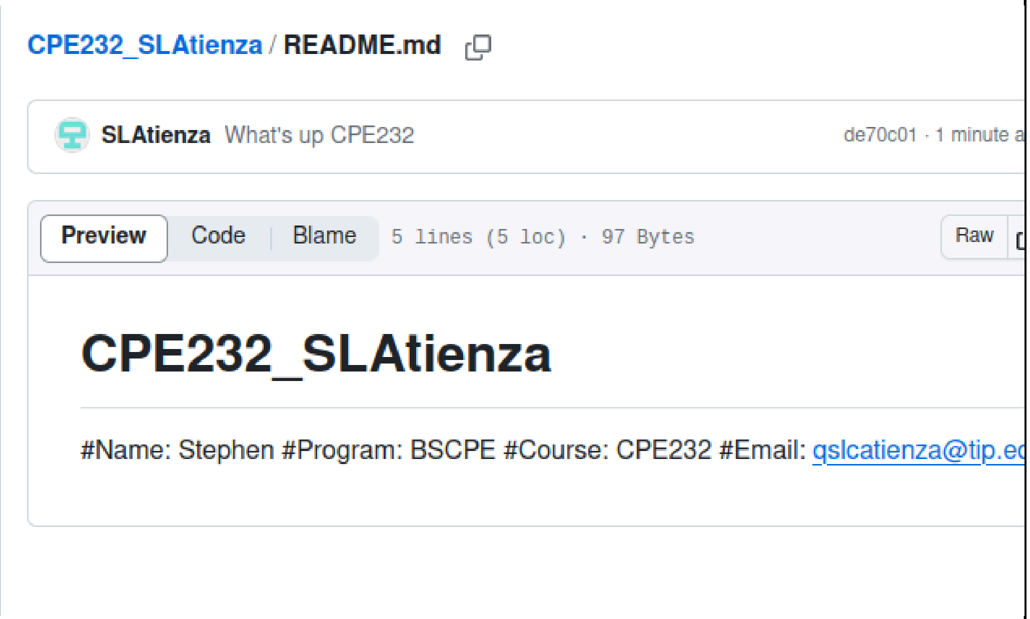
l. Use the command *git push <remote><branch>* to upload the local repository content to GitHub repository. Pushing means to transfer commits from the local repository to the remote repository. As an example, you may issue *git push origin main*



```
stephen@worksation:~/CPE232_SLAtienza$ git push origin mai
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 2 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 340 bytes | 340.00 KiB/s, don
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:SLAtienza/CPE232_SLAtienza.git
   418011b..de70c01  main -> main
```

.

m. On the GitHub repository, verify that the changes have been made to README.md by refreshing the page. Describe the README.md file. You can notice the how long was the last commit. It should be some minutes ago and the message you typed on the git commit command should be there. Also, the README.md file should have been edited according to the text you wrote.



CPE232_SLAtienza / README.md

SLAtienza What's up CPE232 de70c01 · 1 minute a

5 lines (5 loc) · 97 Bytes

CPE232_SLAtienza

#Name: Stephen #Program: BSCPE #Course: CPE232 #Email: qslcatienza@tip.ed

Reflections:

Answer the following:

3. What sort of things have we so far done to the remote servers using ansible commands?
    - Remote hosting and connecting to other stations are topics we go into and master.

4. How important is the inventory file?
    - Its significance lies in furnishing the user with a list that enables the administrator to oversee and host the other stations.

Conclusions/Learnings:

- Our lessons have prepared you to work safely and efficiently in remote and collaborative settings by teaching you how to connect to distant computers without using passwords, how to create a pair of keys to unlock secure communication, how to check that your connection is stable, how to create secure spaces for your computer code to collaborate, and how to send short instructions to distant computers using those same keys.