Name: Dela cruz, Ivan Kenneth B.	**Date Performed:** Aug 27, 2023
Course/Section: CPE31S5	**Date Submitted:** Aug 28, 2023
Instructor: Engr. Roman Richard	**Semester and SY:** 1st Sem (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 locald 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.

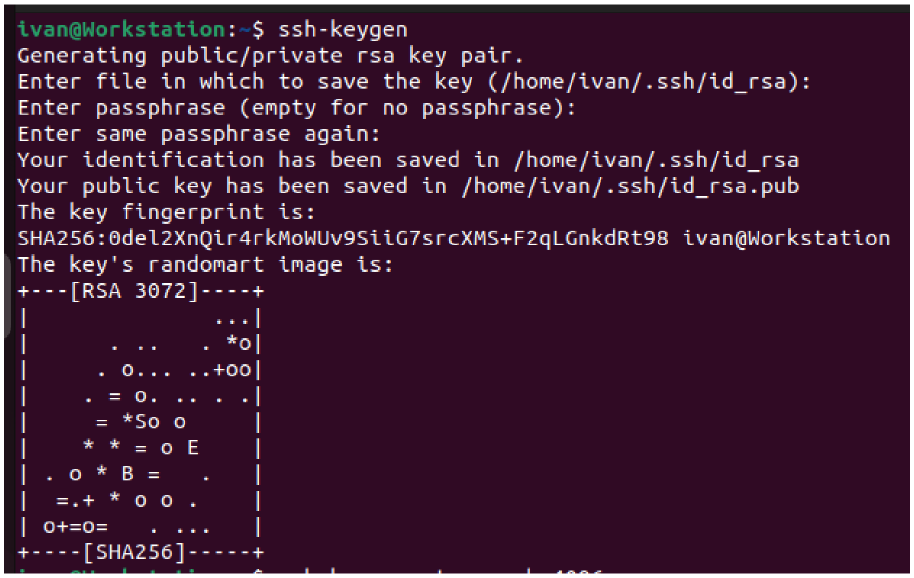
SSH introduced public key authentication as a more secure alternative to the older .rhosts authentication. It improved security by avoiding the need to have password stored in files and eliminated the possibility of a compromised server stealing the user's password.

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

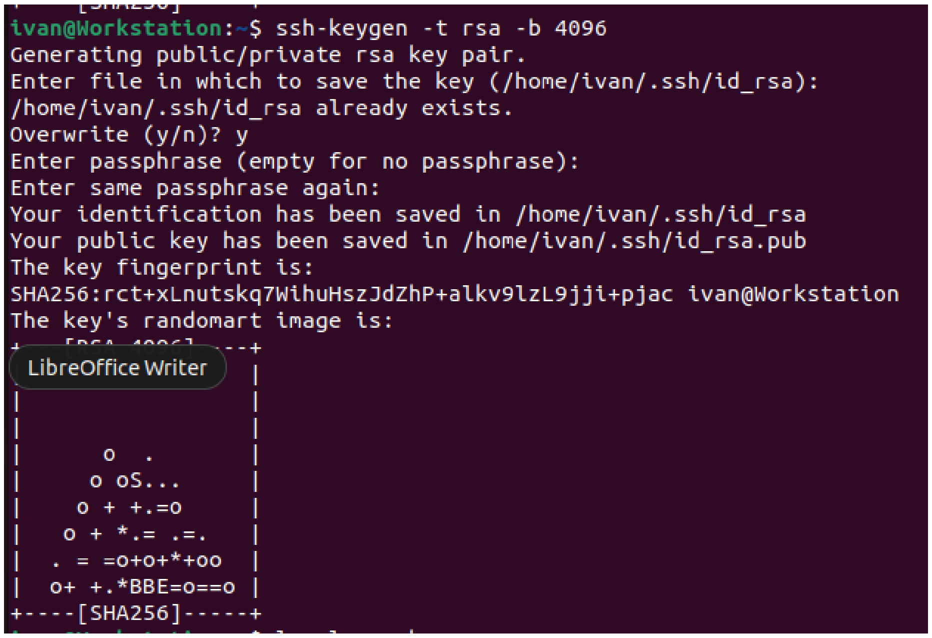
1. The simplest way to generate a key pair is to run *ssh-keygen* without arguments. In this case, it will prompt for the file in which to store keys. First,

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*.



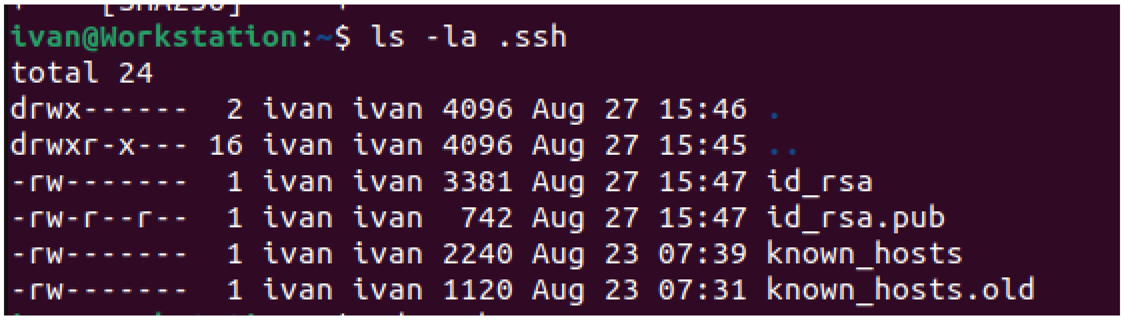
```
ivan@Workstation:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/ivan/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/ivan/.ssh/id_rsa
Your public key has been saved in /home/ivan/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:0del2XnQir4rkMoWUv9SiiG7srcXMS+F2qLGnkdRt98 ivan@Workstation
The key's randomart image is:
+---[RSA 3072]----+
|              ...|
|      . ..   . *o|
|     . o... ..+oo|
|    . = o. .. . .|
|     = *So o     |
|    * * = o E    |
| . o * B =   .   |
|  =.+ * o o .    |
| o+=o=   . ...   |
+----[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.
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.



```
ivan@Workstation:~$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/ivan/.ssh/id_rsa):
/home/ivan/.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 /home/ivan/.ssh/id_rsa
Your public key has been saved in /home/ivan/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:rct+xLnutskq7WihuHszJdZhP+alkv9lzL9jji+pjac ivan@Workstation
The key's randomart image is:
+---[RSA 4096]----+
|                 |
|                 |
|                 |
|      o  .       |
|     o oS...     |
|    o + +.=o     |
|   o + *.= .=.   |
|  . = =o+o+*+oo  |
|  o+ +.*BBE=o==o |
+----[SHA256]-----+
```

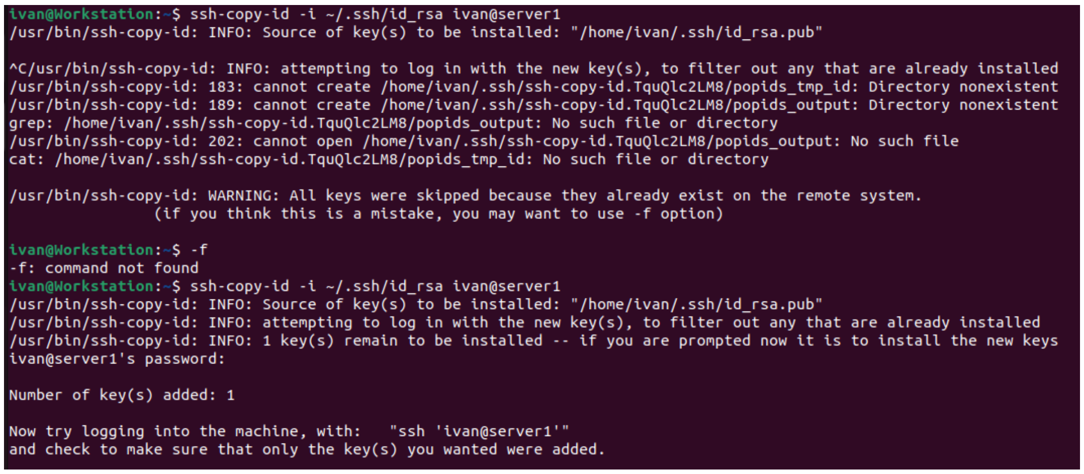
4. Verify that you have created the key by issuing the command *ls -la .ssh.* The command should show the .ssh directory containing a pair of keys. For example, id_rsa.pub and id_rsa.



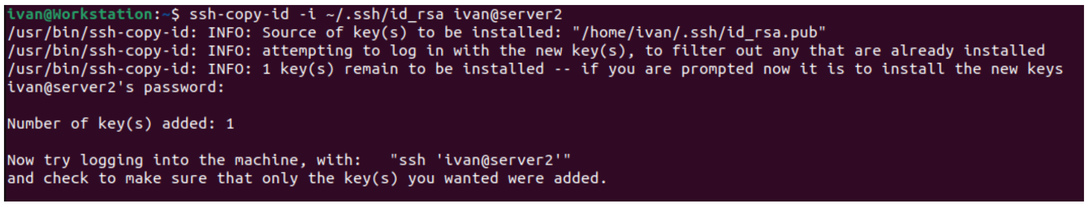
```
ivan@Workstation:~$ ls -la .ssh
total 24
drwx------  2 ivan ivan 4096 Aug 27 15:46 .
drwxr-x--- 16 ivan ivan 4096 Aug 27 15:45 ..
-rw-------  1 ivan ivan 3381 Aug 27 15:47 id_rsa
-rw-r--r--  1 ivan ivan  742 Aug 27 15:47 id_rsa.pub
-rw-------  1 ivan ivan 2240 Aug 23 07:39 known_hosts
-rw-------  1 ivan ivan 1120 Aug 23 07:31 known_hosts.old
```

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*

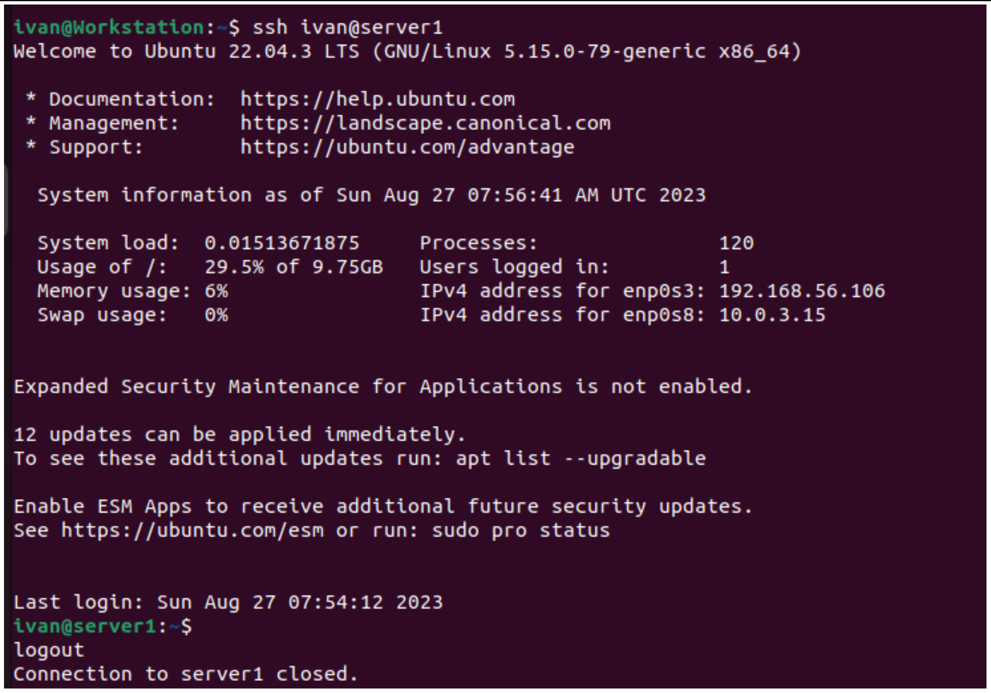


```
ivan@Workstation:~$ ssh-copy-id -i ~/.ssh/id_rsa ivan@server1
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/ivan/.ssh/id_rsa.pub"

^C/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: 183: cannot create /home/ivan/.ssh/ssh-copy-id.TquQlc2LM8/popids_tmp_id: Directory nonexistent
/usr/bin/ssh-copy-id: 189: cannot create /home/ivan/.ssh/ssh-copy-id.TquQlc2LM8/popids_output: Directory nonexistent
grep: /home/ivan/.ssh/ssh-copy-id.TquQlc2LM8/popids_output: No such file or directory
/usr/bin/ssh-copy-id: 202: cannot open /home/ivan/.ssh/ssh-copy-id.TquQlc2LM8/popids_output: No such file
cat: /home/ivan/.ssh/ssh-copy-id.TquQlc2LM8/popids_tmp_id: No such file or directory

/usr/bin/ssh-copy-id: WARNING: All keys were skipped because they already exist on the remote system.
                (if you think this is a mistake, you may want to use -f option)

ivan@Workstation:~$ -f
-f: command not found
ivan@Workstation:~$ ssh-copy-id -i ~/.ssh/id_rsa ivan@server1
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/ivan/.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 prompted now it is to install the new keys
ivan@server1's password:

Number of key(s) added: 1

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

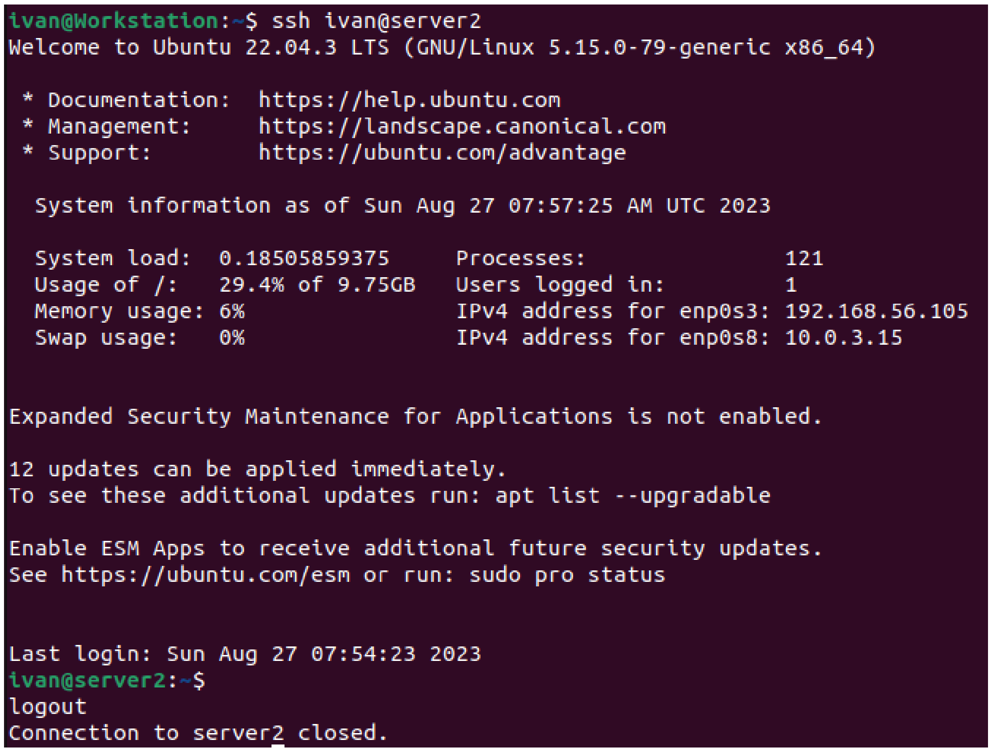


```
ivan@Workstation:~$ ssh-copy-id -i ~/.ssh/id_rsa ivan@server2
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/ivan/.ssh/id_rsa.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
ivan@server2's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'ivan@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?



```
ivan@Workstation:~$ ssh ivan@server1
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 5.15.0-79-generic x86_64)

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

  System information as of Sun Aug 27 07:56:41 AM UTC 2023

  System load:  0.01513671875     Processes:             120
  Usage of /:   29.5% of 9.75GB   Users logged in:       1
  Memory usage: 6%                IPv4 address for enp0s3: 192.168.56.106
  Swap usage:   0%                IPv4 address for enp0s8: 10.0.3.15


Expanded Security Maintenance for Applications is not enabled.

12 updates can be applied immediately.
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: Sun Aug 27 07:54:12 2023
ivan@server1:~$
logout
Connection to server1 closed.
```



```
ivan@Workstation:~$ ssh ivan@server2
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 5.15.0-79-generic x86_64)

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

  System information as of Sun Aug 27 07:57:25 AM UTC 2023

  System load:  0.18505859375     Processes:             121
  Usage of /:   29.4% of 9.75GB   Users logged in:       1
  Memory usage: 6%                IPv4 address for enp0s3: 192.168.56.105
  Swap usage:   0%                IPv4 address for enp0s8: 10.0.3.15


Expanded Security Maintenance for Applications is not enabled.

12 updates can be applied immediately.
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: Sun Aug 27 07:54:23 2023
ivan@server2:~$
logout
Connection to server2 closed.
```

Reflections:

Answer the following:

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

- Remote servers are protected, and users can control them. In a secure context, it also offers an encrypted connection for remote login, command execution, and file transfers between servers.

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

   - I verified that the public keys were installed when I used the command ssh-copy-id -i /.ssh/id_rsa user@host and was able to access the distant servers without being prompted for a password.

Part 2: Discussion

Provide screenshots for each task.

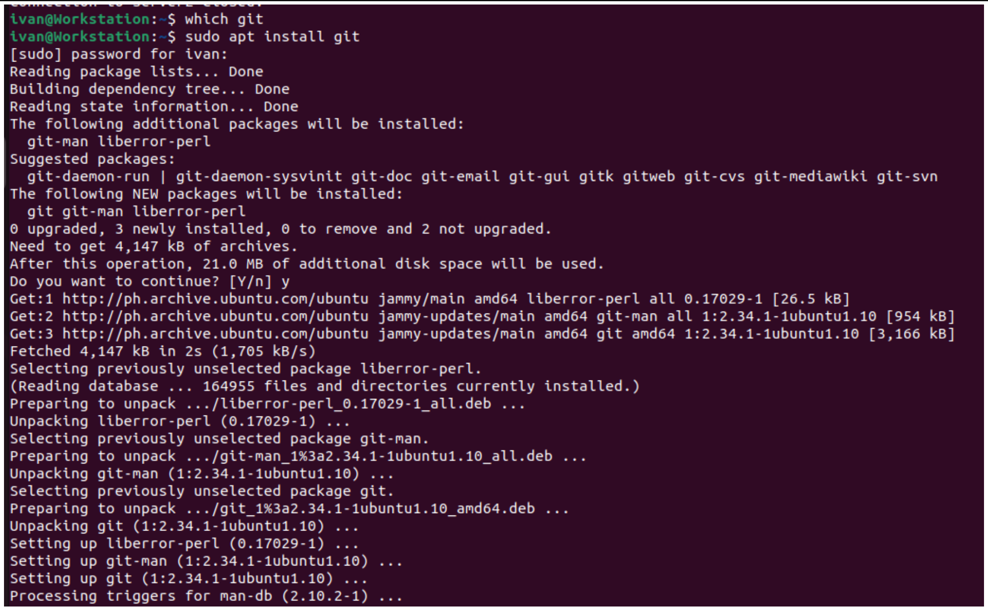
It is assumed that you are done with the last activity (**Activity 2: SSH Key-Based Authentication**).

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

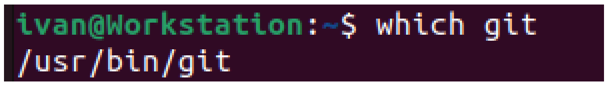
Task 3: Set up the Git Repository

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*



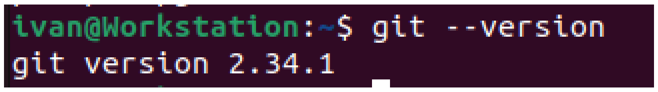
```
ivan@Workstation:~$ which git
ivan@Workstation:~$ sudo apt install git
[sudo] password for ivan:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  git-man liberror-perl
Suggested packages:
  git-daemon-run | 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 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 all 0.17029-1 [26.5 kB]
Get:2 http://ph.archive.ubuntu.com/ubuntu jammy-updates/main amd64 git-man all 1:2.34.1-1ubuntu1.10 [954 kB]
Get:3 http://ph.archive.ubuntu.com/ubuntu jammy-updates/main amd64 git amd64 1:2.34.1-1ubuntu1.10 [3,166 kB]
Fetched 4,147 kB in 2s (1,705 kB/s)
Selecting previously unselected package liberror-perl.
(Reading database ... 164955 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.
Preparing to unpack .../git-man_1%3a2.34.1-1ubuntu1.10_all.deb ...
Unpacking git-man (1:2.34.1-1ubuntu1.10) ...
Selecting previously unselected package git.
Preparing to unpack .../git_1%3a2.34.1-1ubuntu1.10_amd64.deb ...
Unpacking git (1:2.34.1-1ubuntu1.10) ...
Setting up liberror-perl (0.17029-1) ...
Setting up git-man (1:2.34.1-1ubuntu1.10) ...
Setting up git (1:2.34.1-1ubuntu1.10) ...
Processing triggers for man-db (2.10.2-1) ...
```

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



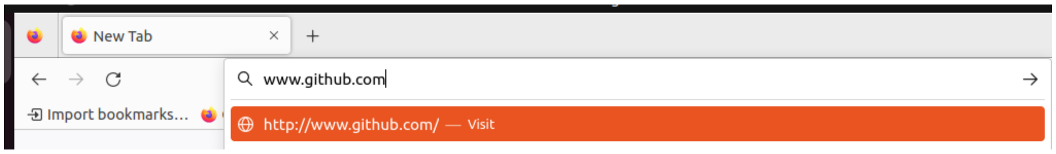
```
ivan@Workstation:~$ which git
/usr/bin/git
```

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

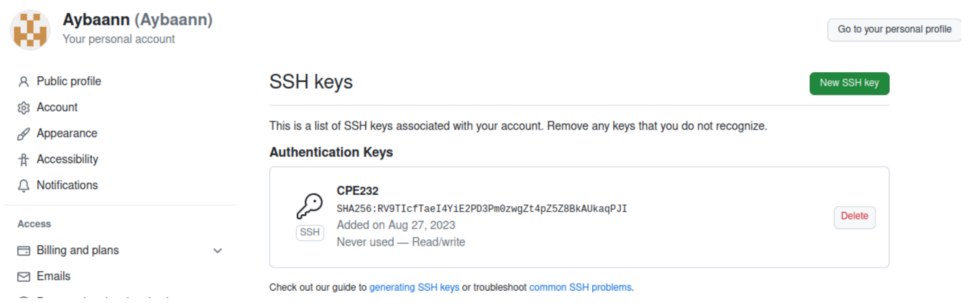


```
ivan@Workstation:~$ git --version
git version 2.34.1
```

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.
    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

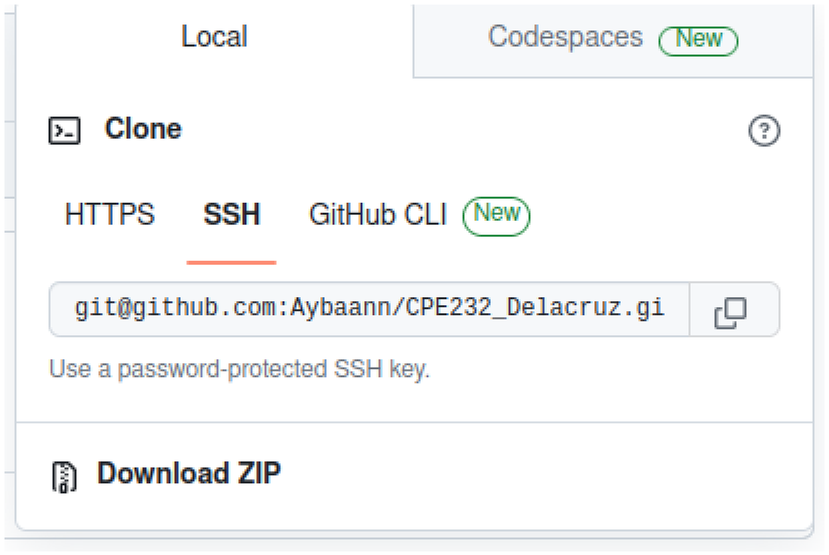
This is a list of SSH keys associated with your account. Remove any keys that you do not recognize.

Authentication Keys

CPE232
SHA256:RV9TIcfTaeI4YiE2PD3Pm0zwgZt4pZ5Z8BkAUkaqPJI
Added on Aug 27, 2023
Never used — Read/write

Check out our guide to generating SSH keys or troubleshoot common SSH problems.

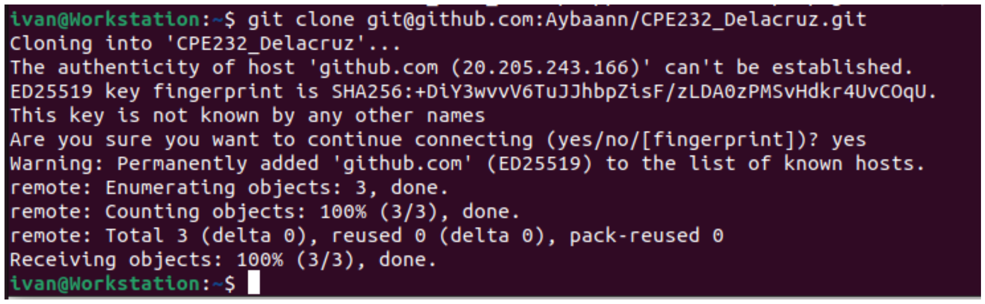
d. Clone the repository that you created. In doing this, you need to get the link from GitHub. Browse to your repository as shown below. Click on the Code drop down menu. Select SSH and copy the link.



git@github.com:Aybaann/CPE232_Delacruz.gi

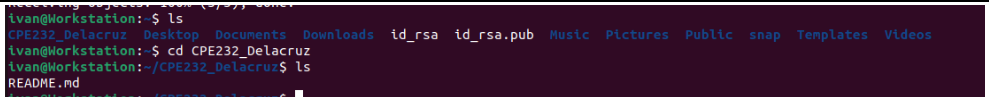
Use a password-protected SSH key.

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.



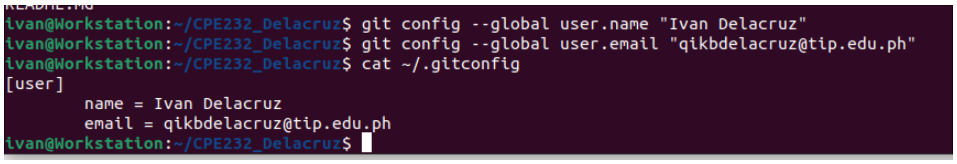
```
ivan@Workstation:~$ git clone git@github.com:Aybaann/CPE232_Delacruz.git
Cloning into 'CPE232_Delacruz'...
The authenticity of host 'github.com (20.205.243.166)' can't be established.
ED25519 key fingerprint is SHA256:+DiY3wvvV6TuJJhbpZisF/zLDA0zPMSvHdkr4UvCOqU.
This key is not known by any other 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: 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.
ivan@Workstation:~$
```

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.



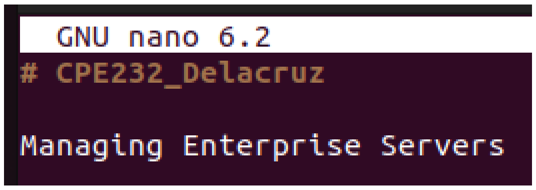
```
ivan@Workstation:~$ ls
CPE232_Delacruz  Desktop  Documents  Downloads  id_rsa  id_rsa.pub  Music  Pictures  Public  snap  Templates  Videos
ivan@Workstation:~$ cd CPE232_Delacruz
ivan@Workstation:~/CPE232_Delacruz$ ls
README.md
```

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 thcommand *cat ~/.gitconfig*



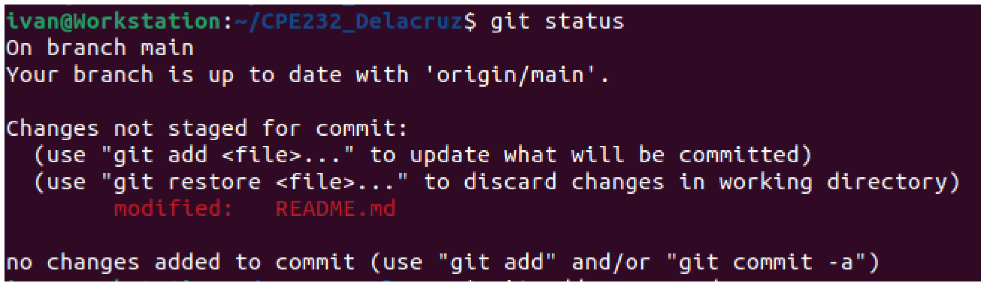
```
ivan@Workstation:~/CPE232_Delacruz$ git config --global user.name "Ivan Delacruz"
ivan@Workstation:~/CPE232_Delacruz$ git config --global user.email "qikbdelacruz@tip.edu.ph"
ivan@Workstation:~/CPE232_Delacruz$ cat ~/.gitconfig
[user]
        name = Ivan Delacruz
        email = qikbdelacruz@tip.edu.ph
ivan@Workstation:~/CPE232_Delacruz$
```

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.



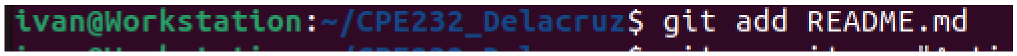
```
  GNU nano 6.2
# CPE232_Delacruz

Managing Enterprise Servers
```

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' being tracked by Git. Status output does not show any information regarding the committed project history. What is the result of issuing this command?



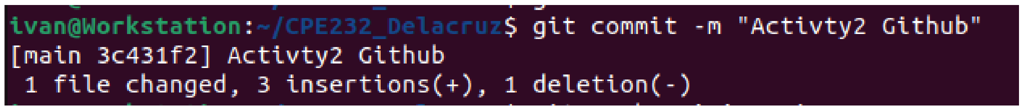
```
ivan@Workstation:~/CPE232_Delacruz$ git status
On branch main
Your branch is up to date with 'origin/main'.

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.



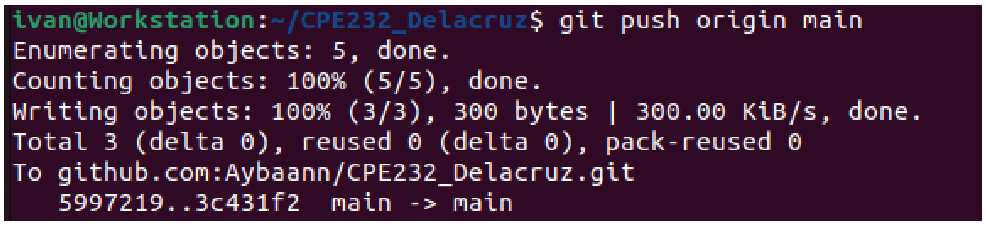
```
ivan@Workstation:~/CPE232_Delacruz$ git add README.md
```

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.



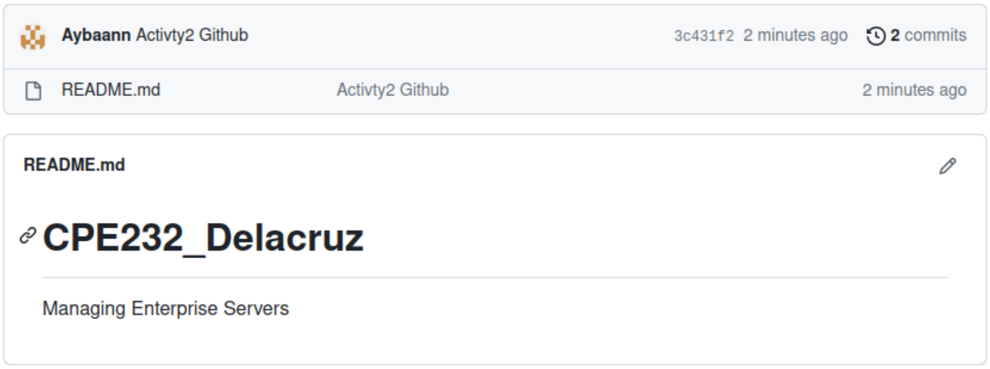
```
ivan@Workstation:~/CPE232_Delacruz$ git commit -m "Activty2 Github"
[main 3c431f2] Activty2 Github
 1 file changed, 3 insertions(+), 1 deletion(-)
```

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.*



```
ivan@Workstation:~/CPE232_Delacruz$ git push origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Writing objects: 100% (3/3), 300 bytes | 300.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:Aybaann/CPE232_Delacruz.git
   5997219..3c431f2  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.



Aybaann Activty2 Github 3c431f2 2 minutes ago **2** commits

README.md Activty2 Github 2 minutes ago

README.md

CPE232_Delacruz

Managing Enterprise Servers

Reflections:

Answer the following:

3. What sort of things have we so far done to the remote servers using ansible commands?
   - Creating a connection between the servers and the GitHub repository was the major usage of the Ansible command. To do this, an SSH key was created and then used to automatically log into the servers and the repository. The files inside the repository were also edited and updated using the Ansible and Nano commands.

4. How important is the inventory file?
   - The inventory file is important because it is good at controlling and keeping track of stock levels. It can save inventory carrying costs and reduce stockouts.

Accurate inventory management depends on keeping an up-to-date inventory file.

Conclusions/Learnings:

- In this activity, the SSH key has a variety of uses. We can say that SSH can be used to access a server with a password and it can be used to access a server without a password by simply generating a public and private key and adding authorization to the server that we want to access. I was able to configure my repository using an Ubuntu terminal by generating an SSH key which gives my terminal access to the repository. Lastly, creating a GitHub was also a great help especially to us. It can be used in collaborative works and can help us to see the changes happening in the working process.