Name: Denila, Nikko Bryner G.	**Date Performed:08/28/23**
Course/Section: CPE31S4	**Date Submitted:08/28/23**
Instructor: Dr. Jonathan V. Taylar	**Semester and SY: 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 Authenticationcd

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.

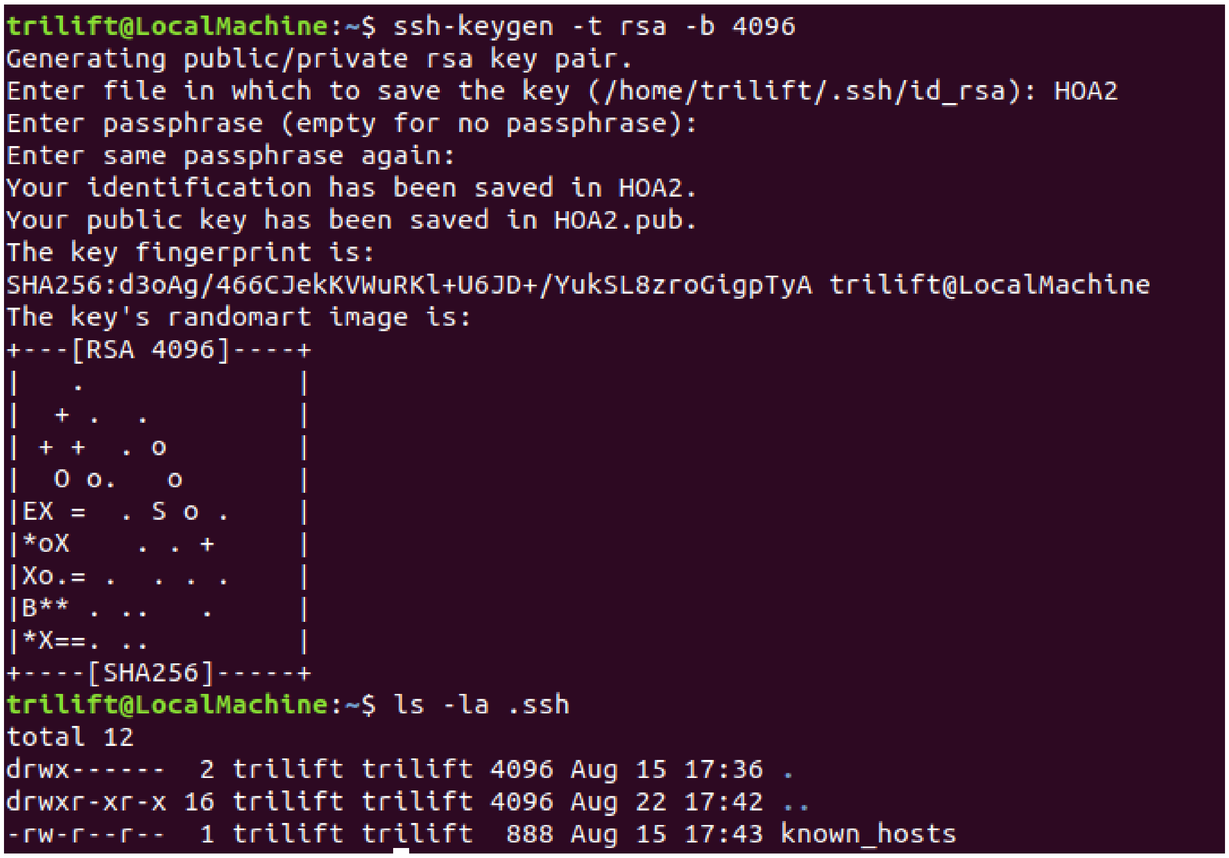
However, SSH keys are authentication credentials just like passwords. Thus, they must be managed somewhat analogously to usernames and passwords. They should have a proper termination process so that keys are removed when no longer needed.

Task 1: Create an SSH Key Pair for User Authentication

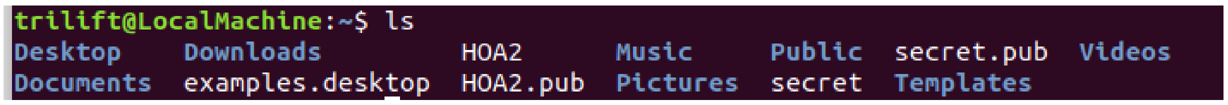
1. The simplest way to generate a key pair is to run *ssh-keygen* without arguments. In this case, it will prompt for the file in which to store keys. First,

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*.

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.
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.



```
trilift@LocalMachine:~$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/trilift/.ssh/id_rsa): HOA2
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in HOA2.
Your public key has been saved in HOA2.pub.
The key fingerprint is:
SHA256:d3oAg/466CJekKVWuRKl+U6JD+/YukSL8zroGigpTyA trilift@LocalMachine
The key's randomart image is:
+---[RSA 4096]----+
|   .             |
|  + .  .         |
| + +  . o        |
|  O o.   o       |
|EX =  . S o .    |
|*oX    . . +     |
|Xo.= .  . . .    |
|B** . ..   .     |
|*X==. ..         |
+----[SHA256]-----+
trilift@LocalMachine:~$ ls -la .ssh
total 12
drwx------  2 trilift trilift 4096 Aug 15 17:36 .
drwxr-xr-x 16 trilift trilift 4096 Aug 22 17:42 ..
-rw-r--r--  1 trilift trilift  888 Aug 15 17:43 known_hosts
```

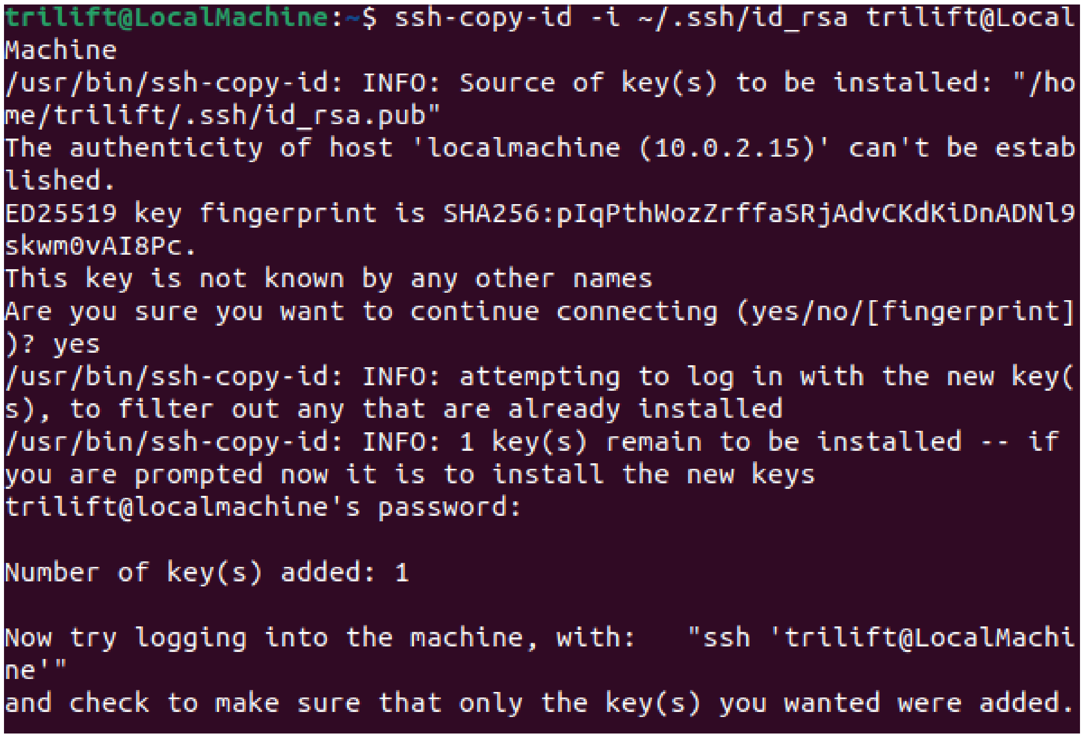


```
trilift@LocalMachine:~$ ls
Desktop    Downloads        HOA2     Music     Public   secret.pub  Videos
Documents  examples.desktop HOA2.pub Pictures  secret   Templates
```

Task 2: Copying the Public Key to the remote servers

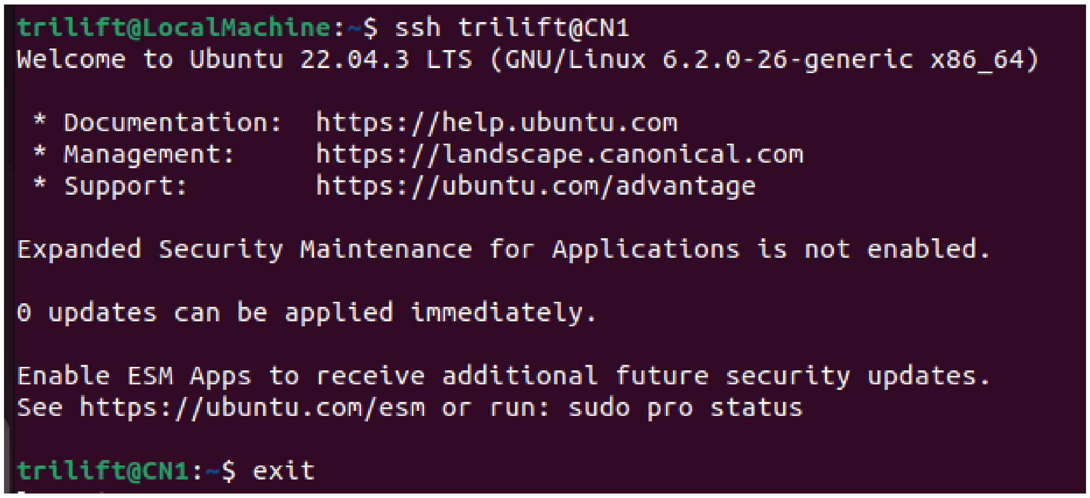
1. To use public key authentication, the public key must be copied to a server and installed in an *authorized_keys* file. This can be conveniently done using the *ssh-copy-id* tool.

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

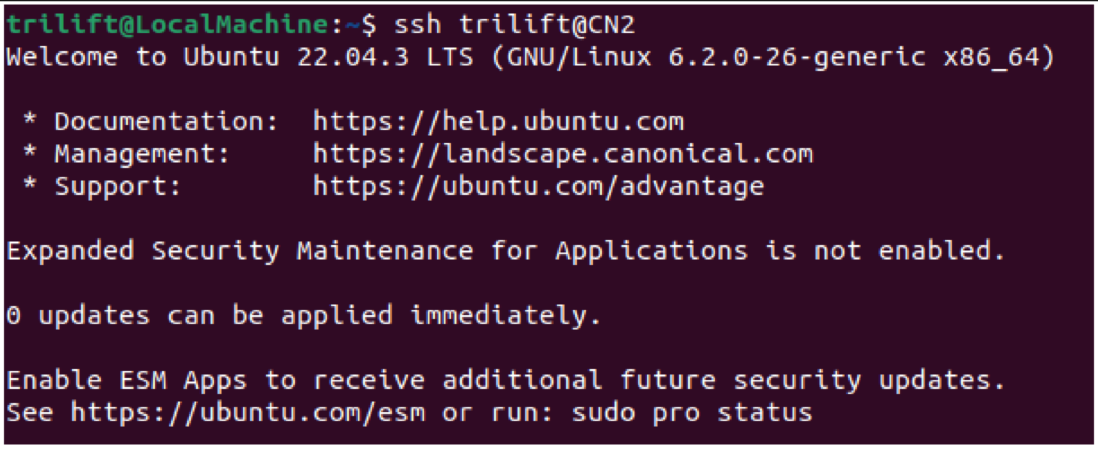


```
trilift@LocalMachine:~$ ssh-copy-id -i ~/.ssh/id_rsa trilift@Local
Machine
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/ho
me/trilift/.ssh/id_rsa.pub"
The authenticity of host 'localmachine (10.0.2.15)' can't be estab
lished.
ED25519 key fingerprint is SHA256:pIqPthWozZrffaSRjAdvCKdKiDnADNl9
skwm0vAI8Pc.
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 prompted now it is to install the new keys
trilift@localmachine's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'trilift@LocalMachi
ne'"
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?



```
trilift@LocalMachine:~$ ssh trilift@CN1
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 6.2.0-26-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.

0 updates can be applied immediately.

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

trilift@CN1:~$ exit
```



```
trilift@LocalMachine:~$ ssh trilift@CN2
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 6.2.0-26-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.

0 updates can be applied immediately.

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

The connection did not ask for a password once I copied the key to server1 and server2. It did not ask for a password since the workstation already has a key that can be used to access the servers.

Reflections:

Answer the following:

1. **How will you describe the ssh-program? What does it do?**
   The ssh-program allows the system admin to access the different systems that is saved in the local machine's hosts list with ease without having to manually go to the place itself, they can just do it remotely.

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

   We can use the ls command to see if the key is successfully copied to the next server.

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

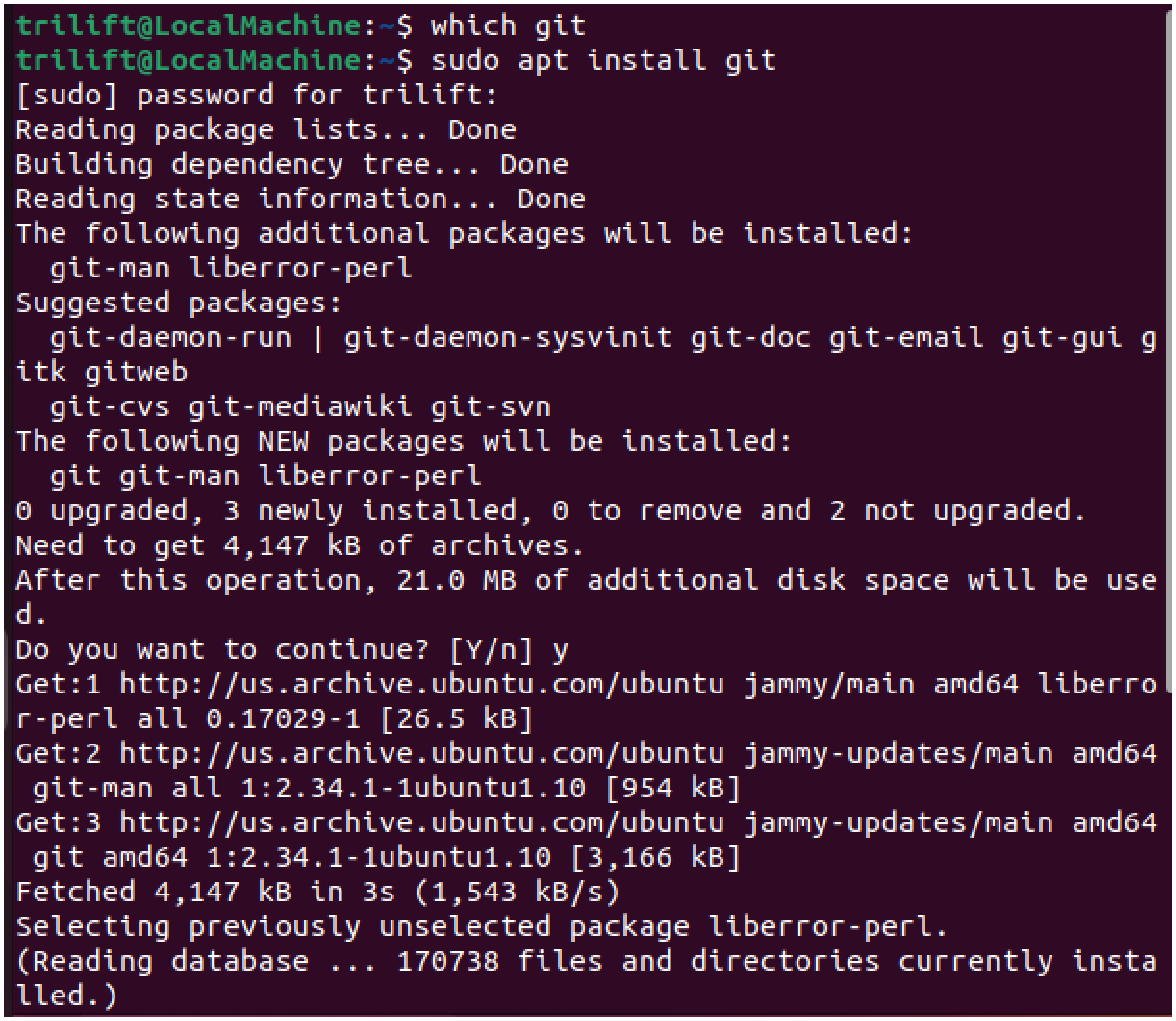
At the heart of GitHub is an open-source version control system (VCS) called Git. Git is responsible for everything GitHub-related that happens locally on your computer. To use Git on the command line, you'll need to download, install, and configure Git on your computer. You can also install GitHub CLI 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

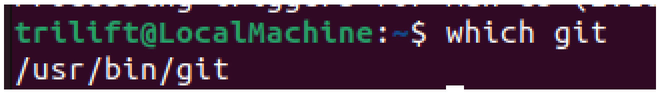
Task 3: Set up the Git Repository

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*



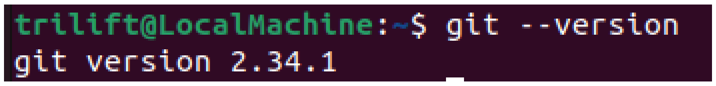
```
trilift@LocalMachine:~$ which git
trilift@LocalMachine:~$ sudo apt install git
[sudo] password for trilift:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  git-man liberror-perl
Suggested packages:
  git-daemon-run | git-daemon-sysvinit git-doc git-email git-gui g
itk 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 use
d.
Do you want to continue? [Y/n] y
Get:1 http://us.archive.ubuntu.com/ubuntu jammy/main amd64 liberro
r-perl all 0.17029-1 [26.5 kB]
Get:2 http://us.archive.ubuntu.com/ubuntu jammy-updates/main amd64
 git-man all 1:2.34.1-1ubuntu1.10 [954 kB]
Get:3 http://us.archive.ubuntu.com/ubuntu jammy-updates/main amd64
 git amd64 1:2.34.1-1ubuntu1.10 [3,166 kB]
Fetched 4,147 kB in 3s (1,543 kB/s)
Selecting previously unselected package liberror-perl.
(Reading database ... 170738 files and directories currently insta
lled.)
```

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



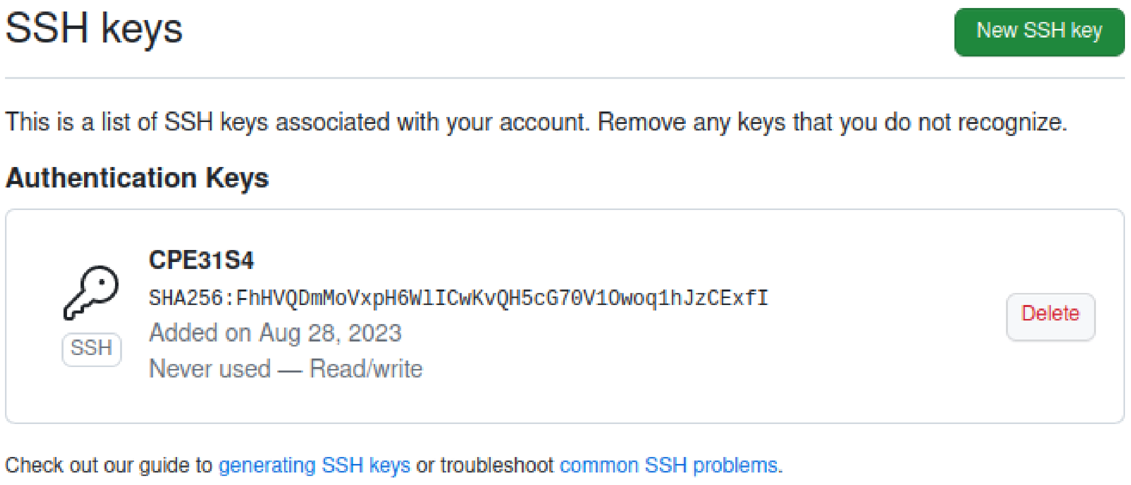
```
trilift@LocalMachine:~$ which git
/usr/bin/git
```

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



```
trilift@LocalMachine:~$ 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.



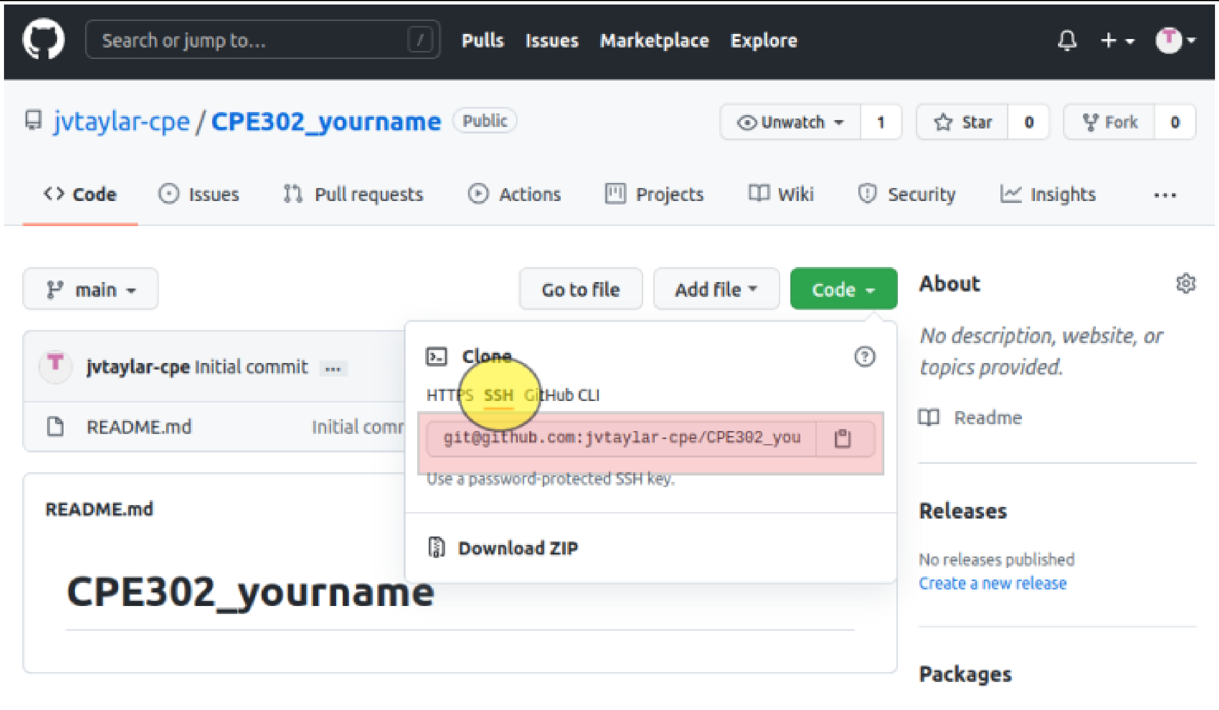
This is a list of SSH keys associated with your account. Remove any keys that you do not recognize.

Authentication Keys

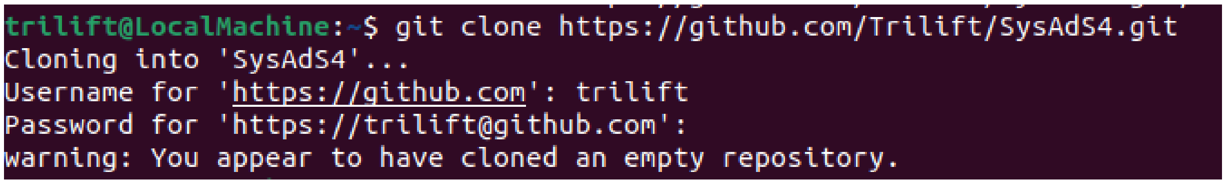
CPE31S4
SHA256:FhHVQDmMoVxpH6WlICwKvQH5cG70V10woq1hJzCExfI
Added on Aug 28, 2023
Never used — Read/write

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.

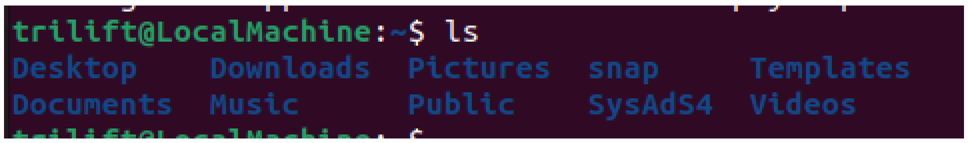


e. Issue the command git clone followed by the copied link. For example, *git clone git@github.com:jvtaylar-cpe/CPE232_yourname.git*. When prompted to continue connecting, type yes and press enter.



```
trilift@LocalMachine:~$ git clone https://github.com/Trilift/SysAdS4.git
Cloning into 'SysAdS4'...
Username for 'https://github.com': trilift
Password for 'https://trilift@github.com':
warning: You appear to have cloned an empty repository.
```

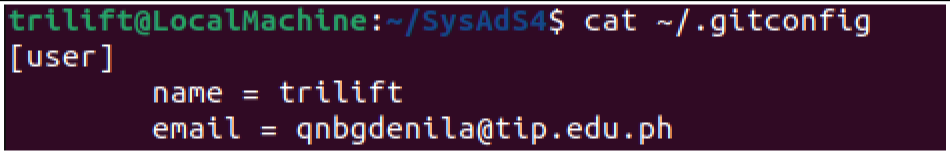
f. To verify that you have cloned the GitHub repository, issue the command *ls*. Observe that you have the CPE232_yourname in the list of your directories. Use CD command to go to that directory and LS command to see the file README.md.



```
trilift@LocalMachine:~$ ls
Desktop     Downloads   Pictures   snap     Templates
Documents   Music       Public     SysAdS4  Videos
```

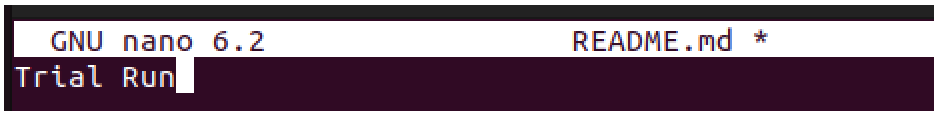
I named the repository as "SysAdS4", as seen in the home directory folder name.

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*



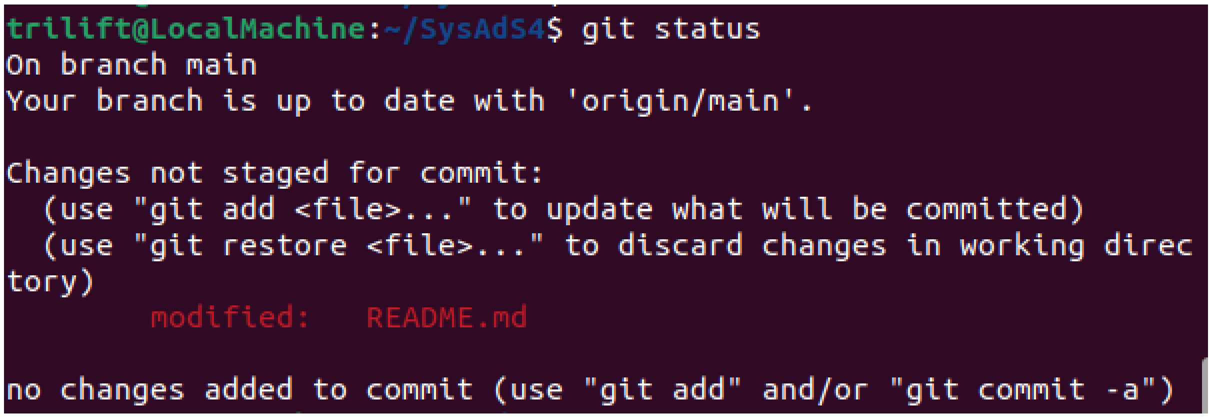
```
trilift@LocalMachine:~/SysAdS4$ cat ~/.gitconfig
[user]
        name = trilift
        email = qnbgdenila@tip.edu.ph
```

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.



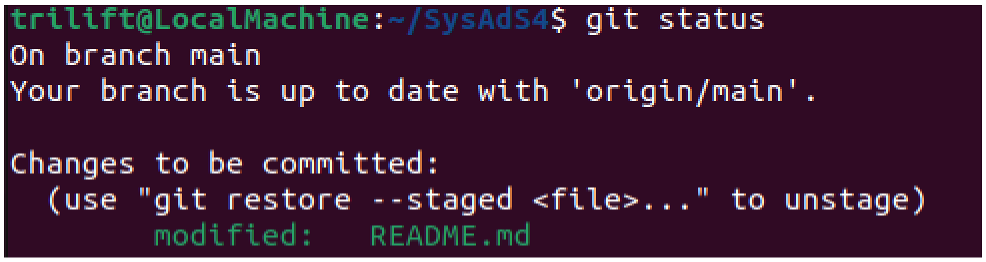
```
  GNU nano 6.2                    README.md *
Trial Run
```

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?



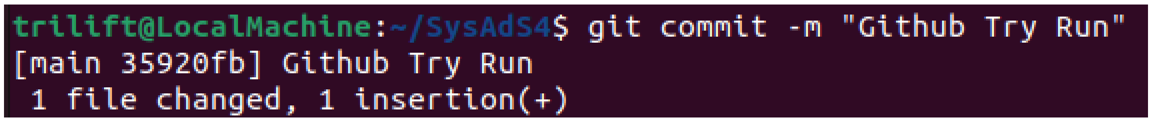
```
trilift@LocalMachine:~/SysAdS4$ git status
On branch main
Your branch is up to date with 'origin/main'.

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



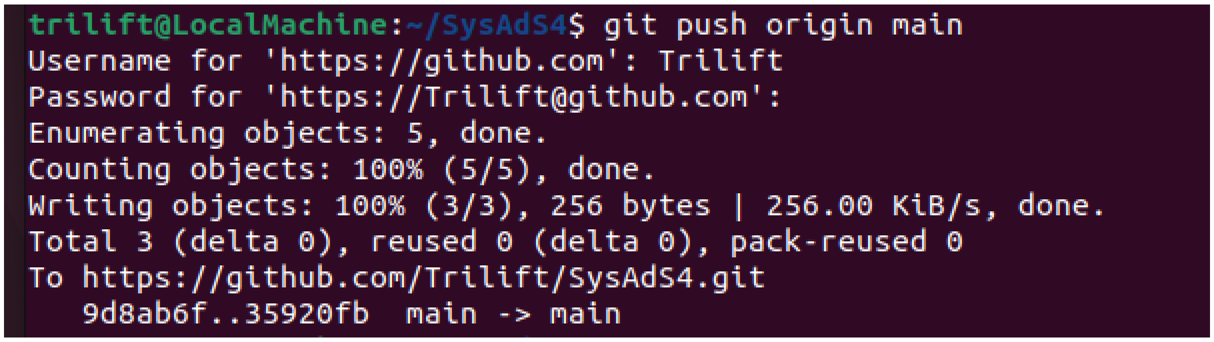
```
trilift@LocalMachine:~/SysAdS4$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   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.



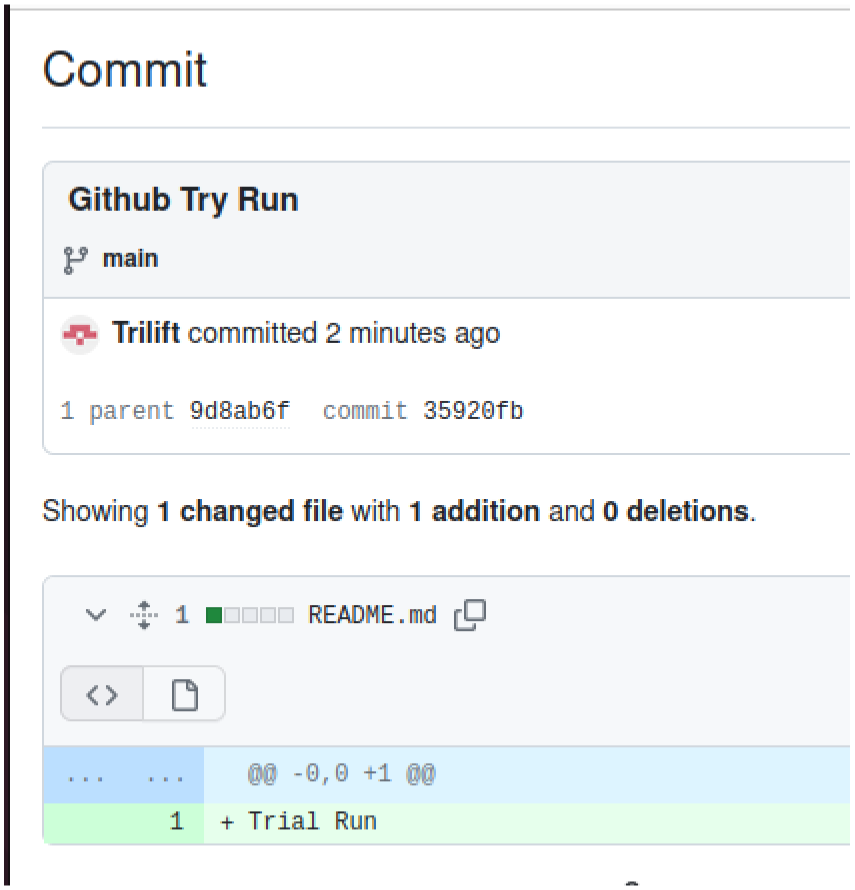
```
trilift@LocalMachine:~/SysAdS4$ git commit -m "Github Try Run"
[main 35920fb] Github Try Run
 1 file changed, 1 insertion(+)
```

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.*



```
trilift@LocalMachine:~/SysAdS4$ git push origin main
Username for 'https://github.com': Trilift
Password for 'https://Trilift@github.com':
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Writing objects: 100% (3/3), 256 bytes | 256.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/Trilift/SysAdS4.git
   9d8ab6f..35920fb  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.



Commit

Github Try Run

main

Trilift committed 2 minutes ago

1 parent 9d8ab6f   commit 35920fb

Showing **1 changed file** with **1 addition** and **0 deletions**.

1 README.md

```
...  ...   @@ -0,0 +1 @@
       1  + Trial Run
```

Reflections:

Answer the following:

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

   So far, we have used ssh, git commands and other basic linux commands to use ansible commands.

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

   The inventory file is important because it keeps track of what is to be committed and to be pushed onto the repository.

Conclusions/Learnings:

In this laboratory activity, I have learned to use the SSH in setting up passphrases and storing them in files in order for my pc to be able to save them for future references. I also learned about the connection of Git and Github in order to record and compile all the works that I have done in my Linux Ubuntu desktop to my Github profile to access them in different platforms, not just in my oracle virtual desktop. This made it easier to look back and check for any mistakes or additional improvements I can give in this line of work.