Name: Shaniah Rose Hope M. Sumaoang	**Date Performed: August 24, 2023**
Course/Section: CPE 232-CPE31S5	**Date Submitted: August 28, 2023**
Instructor: Engr. Roman Richard	**Semester and SY: 1st Semester 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 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.

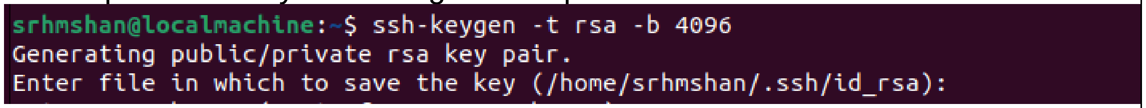
However, SSH keys are authentication credentials just like passwords. Thus, they must be managed somewhat analogously to usernames and passwords. They should have a proper termination process so that keys are removed when no longer needed.

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

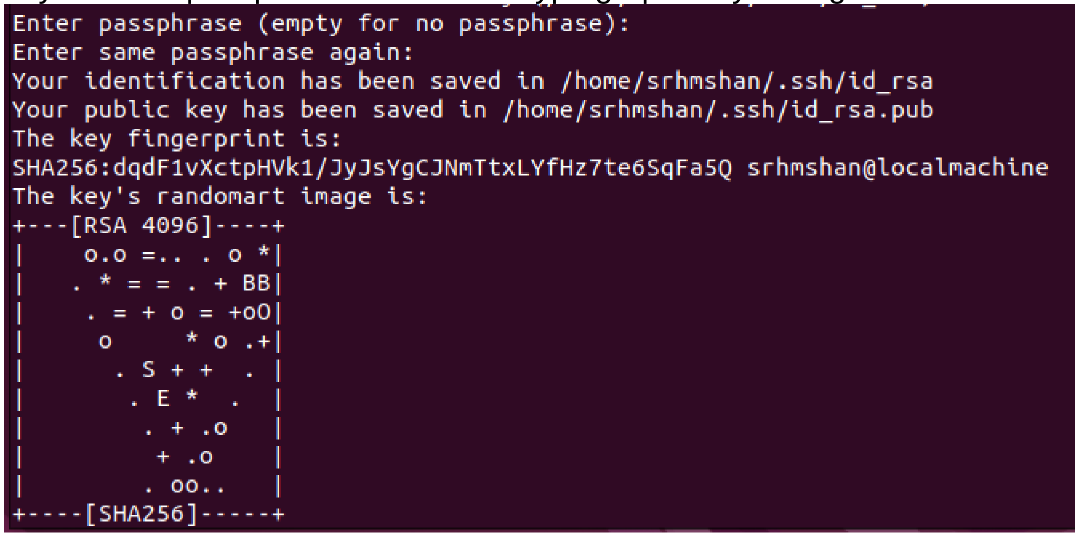
users .ssh directory under the home directory. However, in enterprise environments, the location is often different. The default key file name depends on the algorithm, in this case *id_rsa* when using the default RSA algorithm. It could also be, for example, *id_dsa* or *id_ecdsa*.

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.



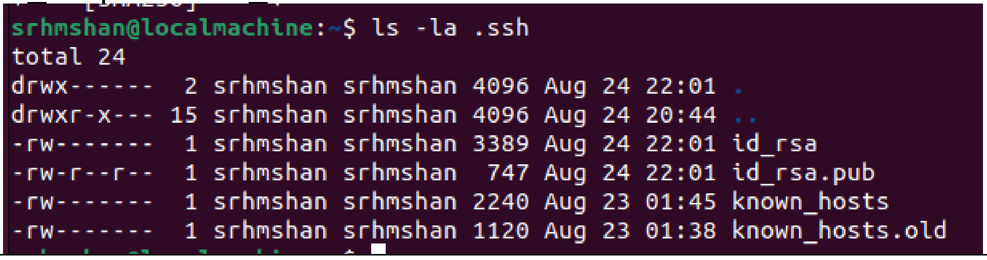
```
srhmshan@localmachine:~$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/srhmshan/.ssh/id_rsa):
```

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.



```
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/srhmshan/.ssh/id_rsa
Your public key has been saved in /home/srhmshan/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:dqdF1vXctpHVk1/JyJsYgCJNmTtxLYfHz7te6SqFa5Q srhmshan@localmachine
The key's randomart image is:
+---[RSA 4096]----+
|    o.o =.. . o *|
|   . * = = . + BB|
|    . = + o = +oO|
|     o     * o .+|
|      . S + +  . |
|       . E *   . |
|        . + .o   |
|         + .o    |
|        . oo..   |
+----[SHA256]-----+
```

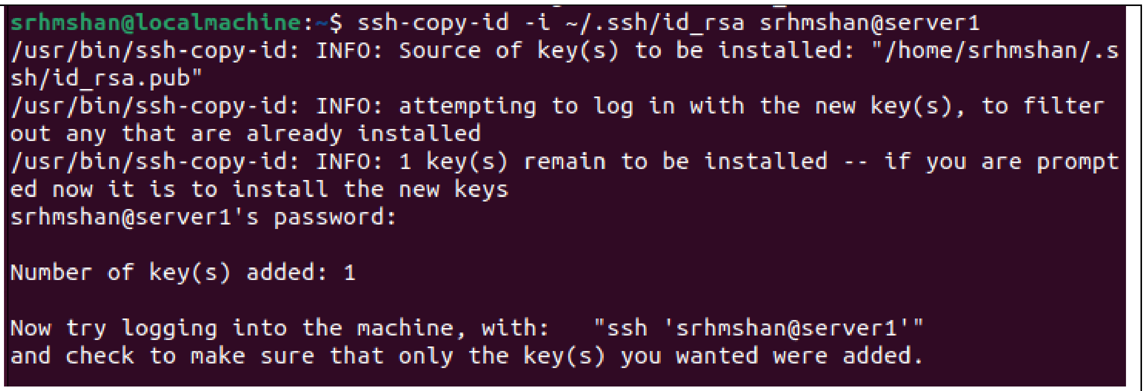
4. Verify that you have created the key by issuing the command *ls -la .ssh.* The command should show the .ssh directory containing a pair of keys. For example, id_rsa.pub and id_rsa.



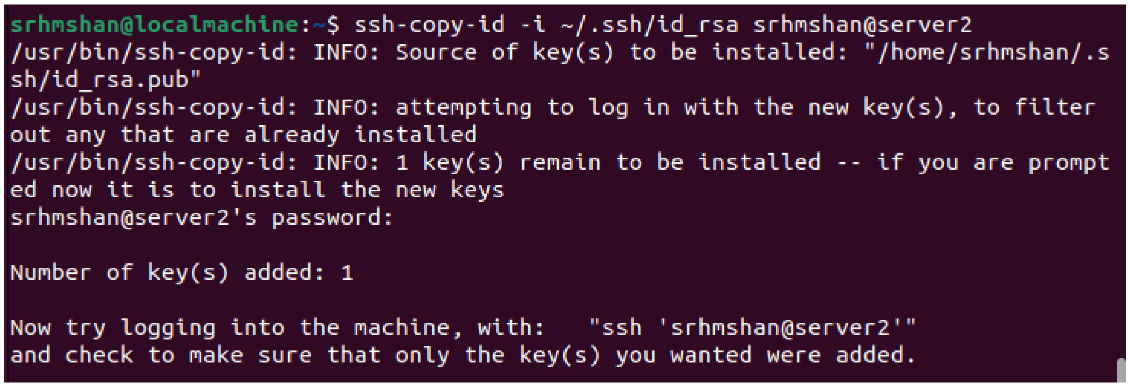
```
srhmshan@localmachine:~$ ls -la .ssh
total 24
drwx------  2 srhmshan srhmshan 4096 Aug 24 22:01 .
drwxr-x--- 15 srhmshan srhmshan 4096 Aug 24 20:44 ..
-rw-------  1 srhmshan srhmshan 3389 Aug 24 22:01 id_rsa
-rw-r--r--  1 srhmshan srhmshan  747 Aug 24 22:01 id_rsa.pub
-rw-------  1 srhmshan srhmshan 2240 Aug 23 01:45 known_hosts
-rw-------  1 srhmshan srhmshan 1120 Aug 23 01:38 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*
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.

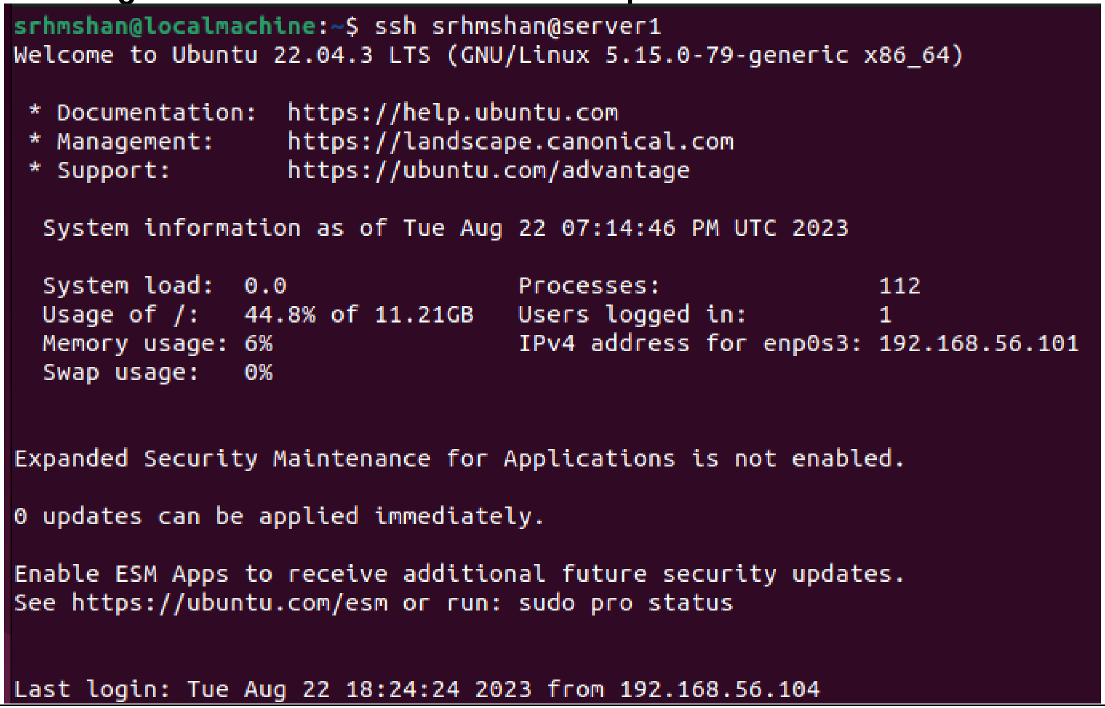


```
srhmshan@localmachine:~$ ssh-copy-id -i ~/.ssh/id_rsa srhmshan@server1
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/srhmshan/.s
sh/id_rsa.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter
out any that are already installed
/usr/bin/ssh-copy-id: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

Number of key(s) added: 1

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

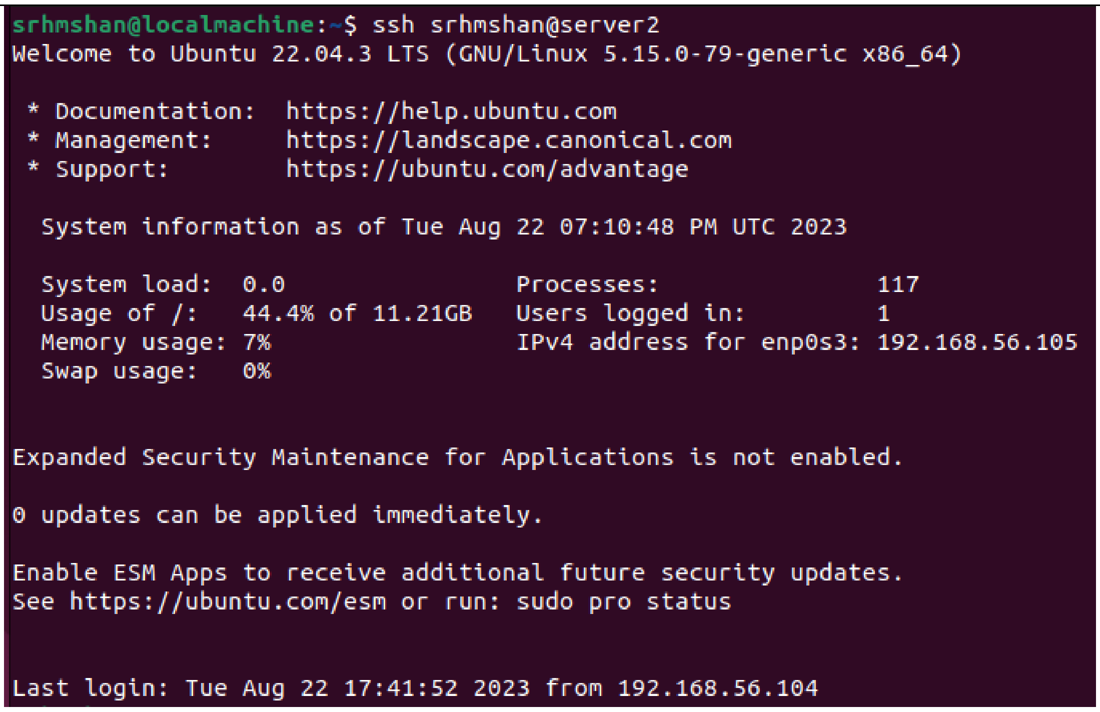


```
srhmshan@localmachine:~$ ssh-copy-id -i ~/.ssh/id_rsa srhmshan@server2
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/srhmshan/.s
sh/id_rsa.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter
out any that are already installed
/usr/bin/ssh-copy-id: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

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'srhmshan@server2'"
and check to make sure that only the key(s) you wanted were added.
```

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? **The connection did not ask for a password because the command I used copies the SSH public key to the remote server's "authorized_keys", enabling it to be authenticated without a password.**



```
srhmshan@localmachine:~$ ssh srhmshan@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 Tue Aug 22 07:14:46 PM UTC 2023

  System load:  0.0               Processes:               112
  Usage of /:   44.8% of 11.21GB   Users logged in:         1
  Memory usage: 6%                IPv4 address for enp0s3: 192.168.56.101
  Swap usage:   0%


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


Last login: Tue Aug 22 18:24:24 2023 from 192.168.56.104
```



```
srhmshan@localmachine:~$ ssh srhmshan@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 Tue Aug 22 07:10:48 PM UTC 2023

  System load:  0.0               Processes:               117
  Usage of /:   44.4% of 11.21GB   Users logged in:         1
  Memory usage: 7%                IPv4 address for enp0s3: 192.168.56.105
  Swap usage:   0%


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


Last login: Tue Aug 22 17:41:52 2023 from 192.168.56.104
```

Reflections:
Answer the following:
1. How will you describe the ssh-program? What does it do? **The SSH-program serves as a secure communication protocol for encrypted connections and remote login functions between systems.**
2. How do you know that you already installed the public key to the remote servers? **To confirm public key integration on remote servers, you can attempt passwordless login using the private key for authentication.**

Part 2: Discussion

Provide screenshots for each task.

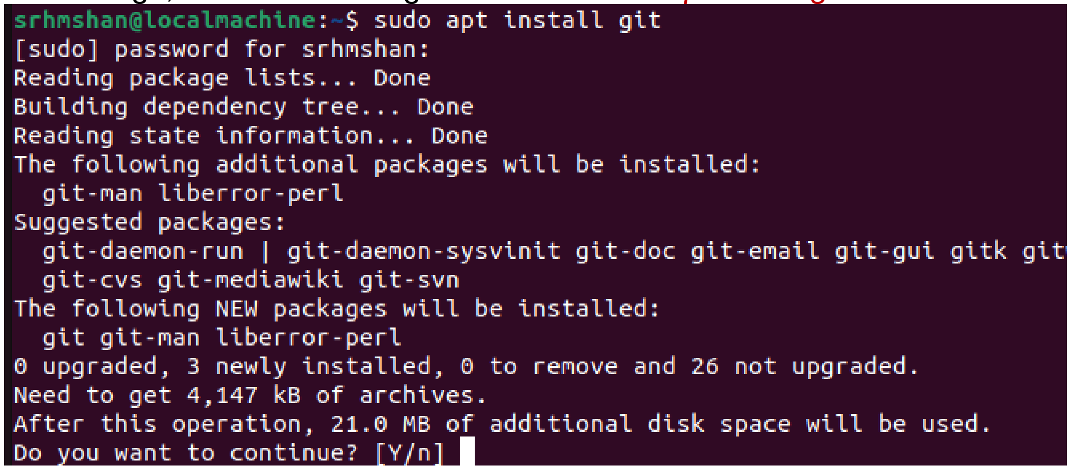
It is assumed that you are done with the last activity (**Activity 2: SSH Key-Based Authentication**).

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

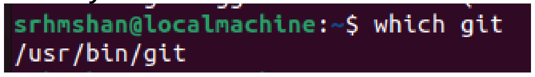
Task 3: Set up the Git Repository

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*



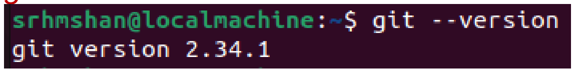
```
srhmshan@localmachine:~$ sudo apt install git
[sudo] password for srhmshan:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  git-man liberror-perl
Suggested packages:
  git-daemon-run | git-daemon-sysvinit 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 26 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]
```

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



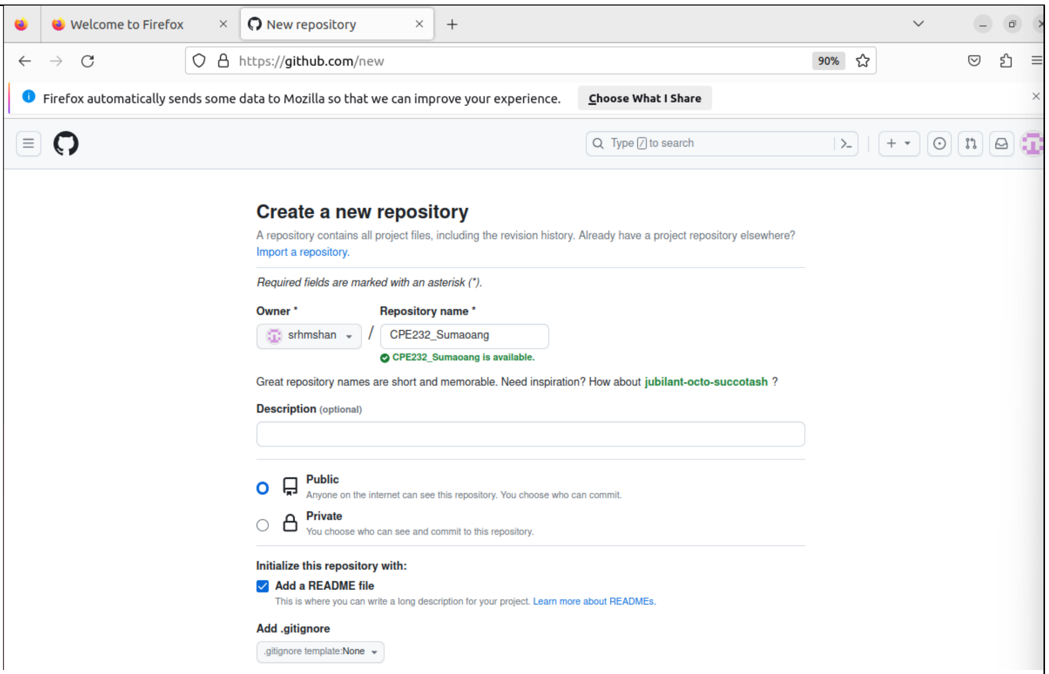
```
srhmshan@localmachine:~$ which git
/usr/bin/git
```

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



```
srhmshan@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.



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

Required fields are marked with an asterisk (*).

Owner * srhmshan / Repository name * CPE232_Sumaoang

CPE232_Sumaoang is available.

Great repository names are short and memorable. Need inspiration? How about jubilant-octo-succotash ?

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:

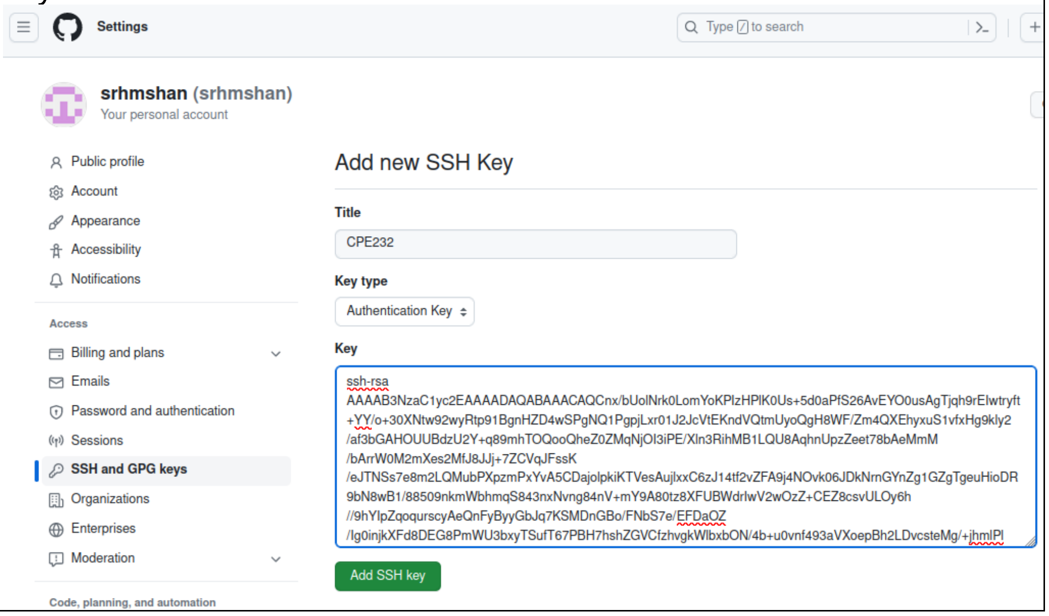
✓ Add a README file
This is where you can write a long description for your project. Learn more about READMEs.

Add .gitignore

.gitignore template: None

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.



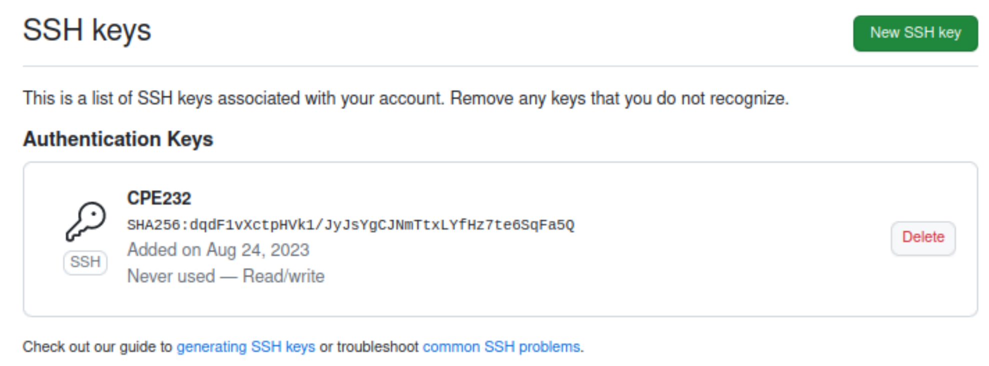
Add new SSH Key

Title: CPE232

Key type: Authentication Key

Key:

```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAACAQCnx/bUolNrk0LomYoKPlzHPlK0Us+5d0aPfS26AvEYO0usAgTjqh9rElwtryft
+YY/o+30XNtw92wyRtp91BgnHZD4wSPgNQ1PgpjLxr01J2JcVtEKndVQtmUyoQgH8WF/Zm4QXEhyxuS1vfxHg9kly2
/af3bGAHOUUBdzU2Y+q89mhTOQooQheZ0ZMqNjOI3iPE/Xln3RihMB1LQU8AqhnUpzZeet78bAeMmM
/bArrW0M2mXes2MfJ8JJj+7ZCVqJFssK
/eJTNSs7e8m2LQMubPXpzmPxYvA5CDajolpkiKTVesAujlxxC6zJ14tf2vZFA9j4NOvk06JDkNrnGYnZg1GZgTgeuHioDR
9bN8wB1/88509nkmWbhmqS843nxNvng84nV+mY9A80tz8XFUBWdrlwV2wOzZ+CEZ8csvULOy6h
//9hYIpZqourscyAeQnFyByyGbJq7KSMDnGBo/FNbS7e/EFDaOZ
/Ig0injkXFd8DEG8PmWU3bxyTSufT67PBH7hshZGVCfzhvgkWlbxbON/4b+u0vnf493aVXoepBh2LDvcsteMg/+jhmIPl
```


SSH keys

New SSH key

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

Authentication Keys

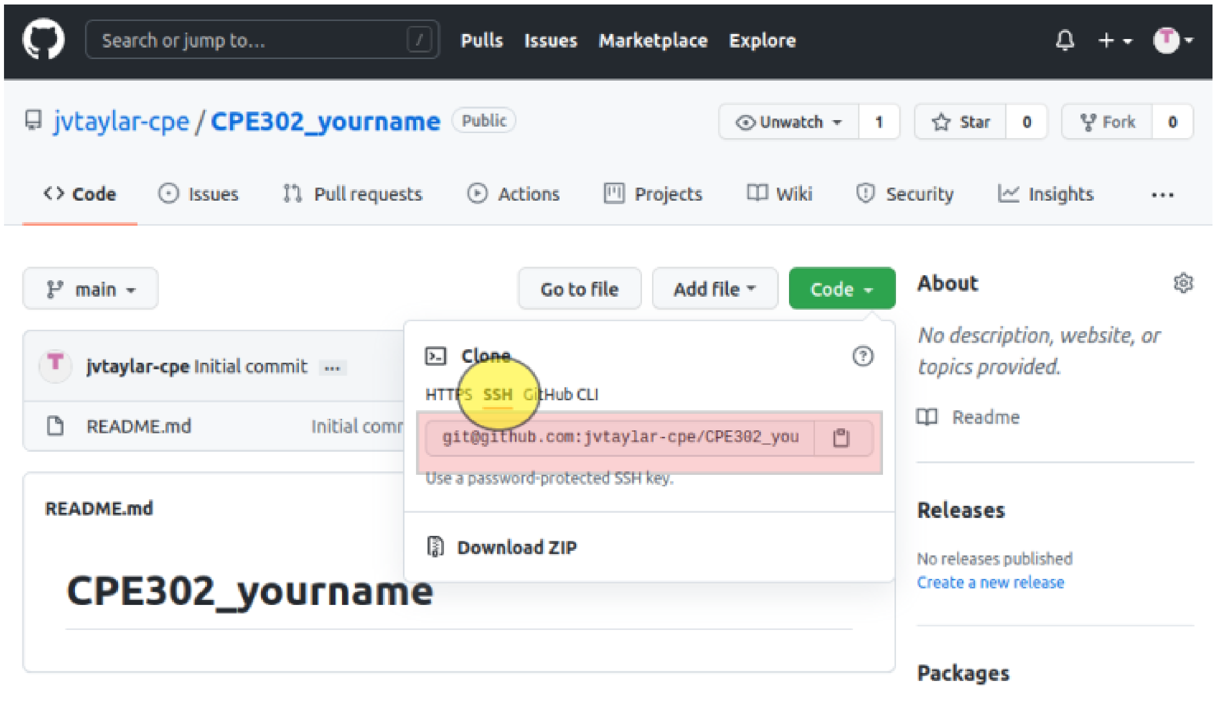
CPE232
SHA256:dqdF1vXctpHVk1/JyJsYgCJNmTtxLYfHz7te6SqFa5Q
Added on Aug 24, 2023
Never used — Read/write

SSH

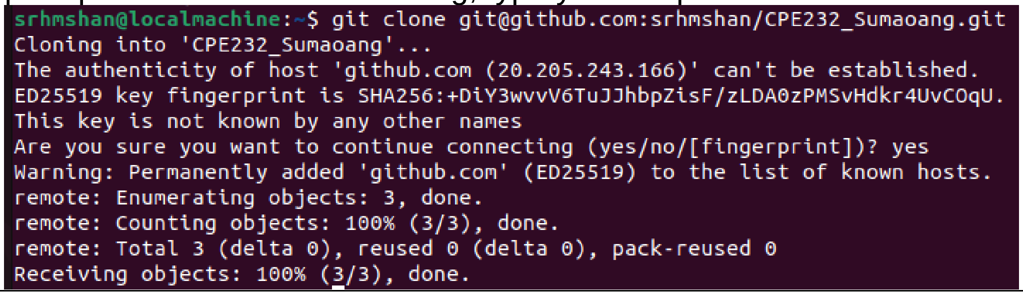
Delete

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

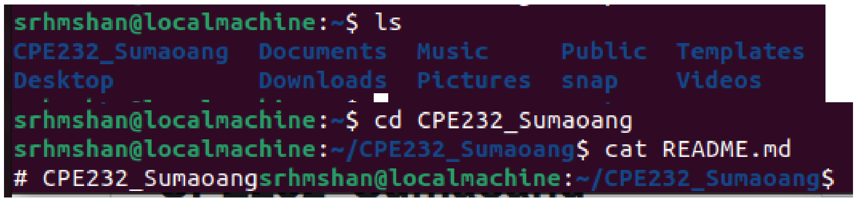
d. Clone the repository that you created. In doing this, you need to get the link from GitHub. Browse to your repository as shown below. Click on the Code drop down menu. Select SSH and copy the link.



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.


```
srhmshan@localmachine:~$ git clone git@github.com:srhmshan/CPE232_Sumaoang.git
Cloning into 'CPE232_Sumaoang'...
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.
```

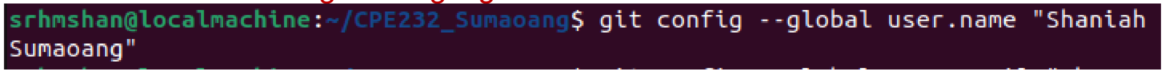
f. To verify that you have cloned the GitHub repository, issue the command *ls*. Observe that you have the CPE232_yourname in the list of your directories. Use CD command to go to that directory and LS command to see the file README.md.



```
srhmshan@localmachine:~$ ls
CPE232_Sumaoang  Documents  Music     Public  Templates
Desktop          Downloads  Pictures  snap    Videos
srhmshan@localmachine:~$ cd CPE232_Sumaoang
srhmshan@localmachine:~/CPE232_Sumaoang$ cat README.md
# CPE232_Sumaoangsrhmshan@localmachine:~/CPE232_Sumaoang$
```

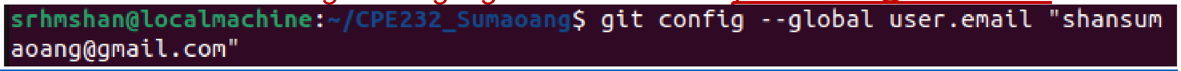
g. Use the following commands to personalize your git.

- *git config --global user.name "Your Name"*



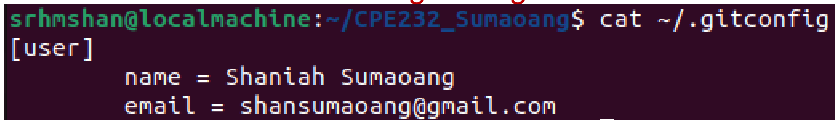
```
srhmshan@localmachine:~/CPE232_Sumaoang$ git config --global user.name "Shaniah
Sumaoang"
```

- *git config --global user.email yourname@email.com*



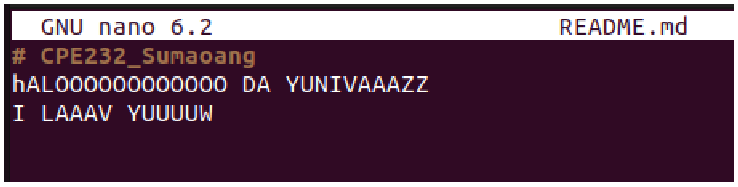
```
srhmshan@localmachine:~/CPE232_Sumaoang$ git config --global user.email "shansum
aoang@gmail.com"
```

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



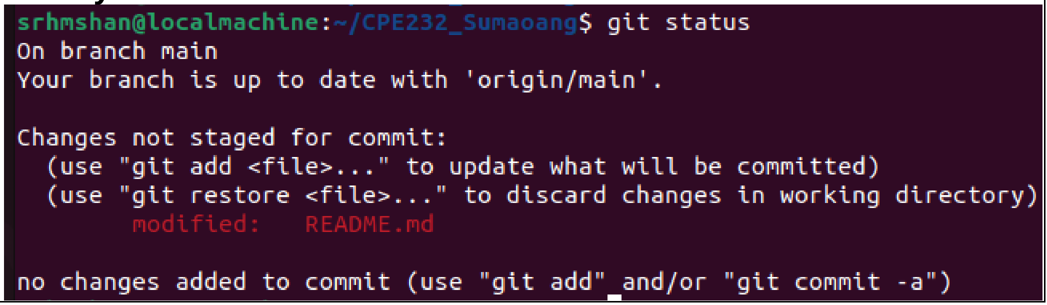
```
srhmshan@localmachine:~/CPE232_Sumaoang$ cat ~/.gitconfig
[user]
        name = Shaniah Sumaoang
        email = shansumaoang@gmail.com
```

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.



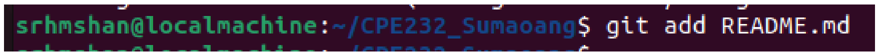
```
  GNU nano 6.2                          README.md
# CPE232_Sumaoang
hALOOOOOOOOOOOO DA YUNIVAAAZZ
I LAAAV YUUUUW
```

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? **The "git status" command reveals the status of changes in the working directory and staging area, including staged and unstaged modifications as well as untracked files, without showing details about committed project history.**



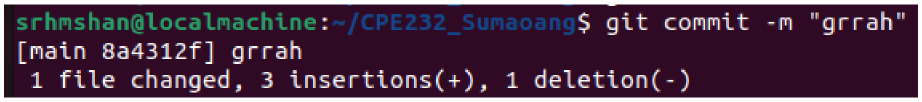
```
srhmshan@localmachine:~/CPE232_Sumaoang$ git status
On branch main
Your branch is up to date with 'origin/main'.

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.



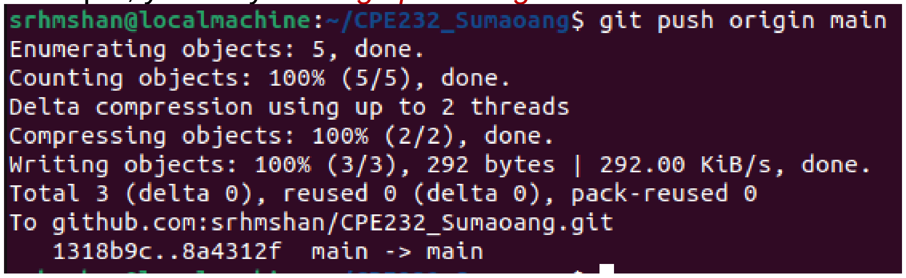
```
srhmshan@localmachine:~/CPE232_Sumaoang$ git add README.md
```

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.



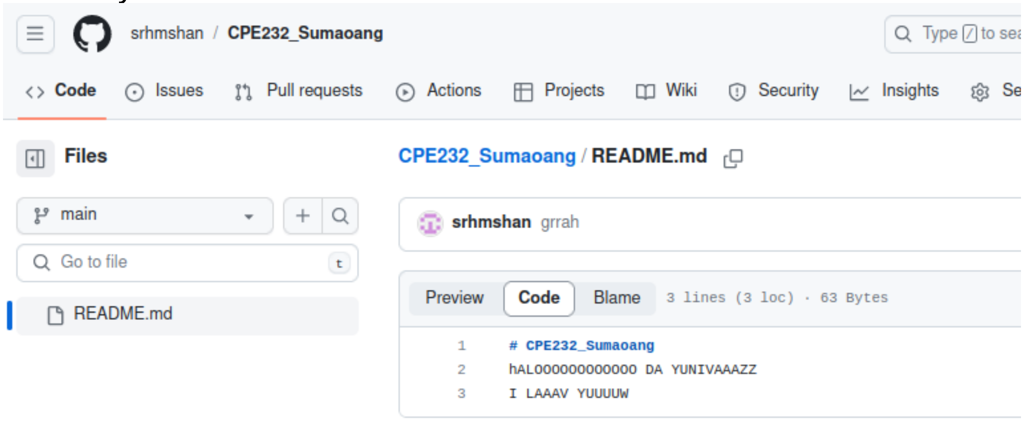
```
srhmshan@localmachine:~/CPE232_Sumaoang$ git commit -m "grrah"
[main 8a4312f] grrah
 1 file changed, 3 insertions(+), 1 deletion(-)
```

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.*



```
srhmshan@localmachine:~/CPE232_Sumaoang$ git push origin main
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), 292 bytes | 292.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:srhmshan/CPE232_Sumaoang.git
   1318b9c..8a4312f  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.



srhmshan grrah

3 lines (3 loc) · 63 Bytes

```
# CPE232_Sumaoang
hAL000000000000 DA YUNIVAAAZZ
I LAAAV YUUUUW
```

Reflections:

Answer the following:

3. What sort of things have we so far done to the remote servers using ansible commands? **So far, I've used ansible commands to manage and fine-tune servers from a remote standpoint, accomplishing operations such as software installation, configuration updates, and service administration.**

4. How important is the inventory file? **The significance of the inventory file is it outlines host details and host groups. By furnishing data about server IP addresses and connectivity specifics, this file enables efficient communication and execution of operations on the designated servers by the commands used.**

Conclusions/Learnings:

In this activity, I learned how to make remote logins safer and easier by using special keys instead of passwords. I created a pair of keys, shared one key with the remote server, and kept the other key on my computer. This helped me connect securely without typing passwords. As I learned, SSH is like a safe chat for computers online. With SSH-keygen, I made a key pair–one for unlocking and one for locking. By sharing a key using ssh-copy-id, I connected easily without typing passwords.

I've acquired practical skills in creating Git repositories, securing communication with SSH keys, and using fundamental Git commands. The process of repository creation, file management, and GitHub interaction is now more understandable. Furthermore, I've comprehended the importance of the Ansible inventory file, enhancing my grasp of version control and server management, aligning with the learning objectives.

In the end, I realized that this activity taught me a lot about using keys for safe and easy connections. It also helped me understand how computers securely talk to each other.