Name: Aaron Martin P. Caro	**Date Performed: August 27, 2023**
Course/Section: CPE232/CPE31S5	**Date Submitted: August 27, 2023**
Instructor: Prof. 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 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.

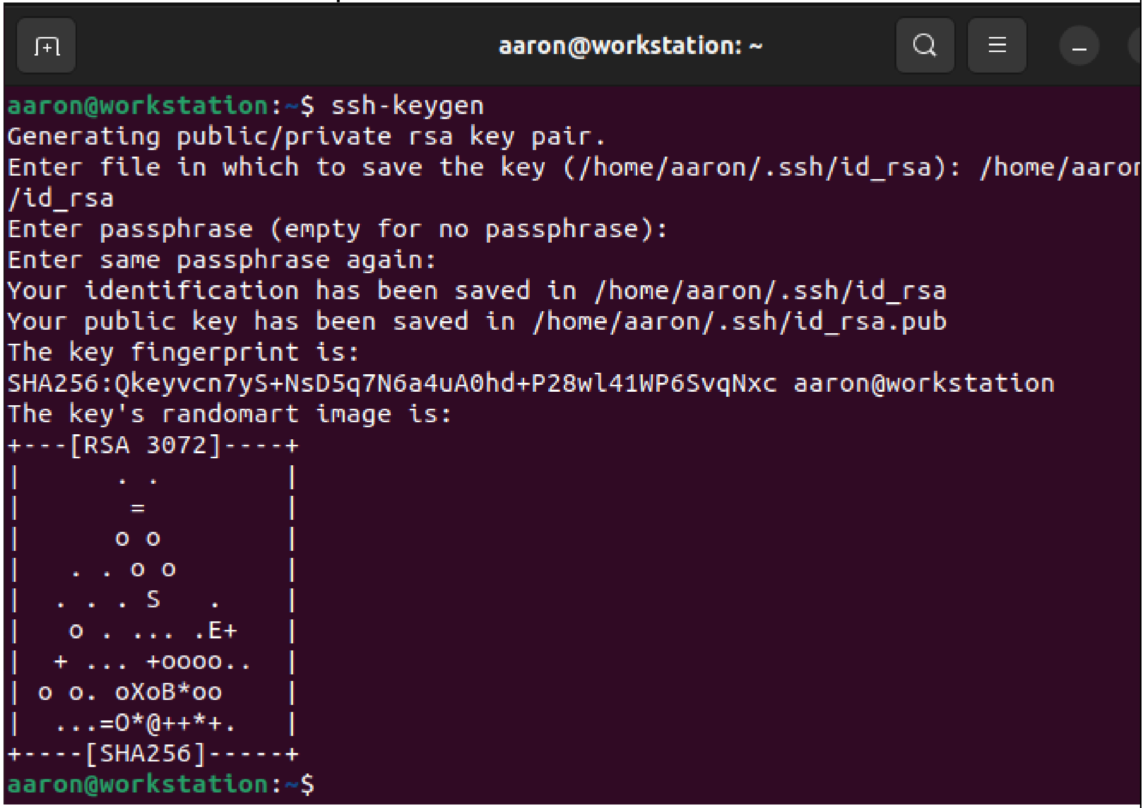
SSH introduced public key authentication as a more secure alternative to the older .rhosts authentication. It improved security by avoiding the need to have password stored in files and eliminated the possibility of a compromised server stealing the user's password.

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

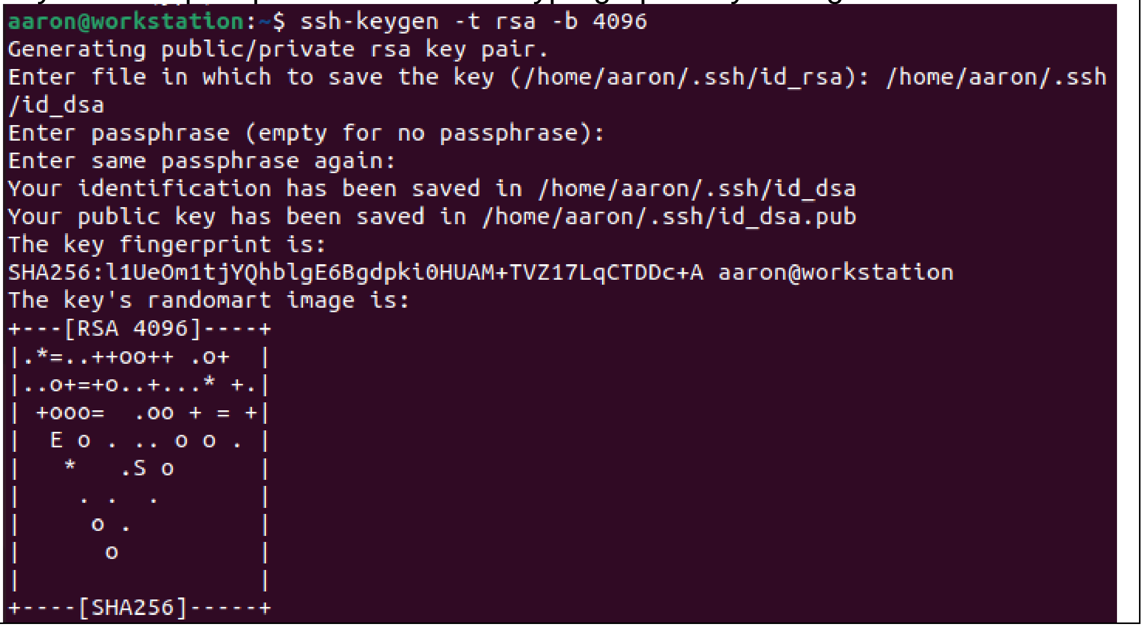
1. The simplest way to generate a key pair is to run *ssh-keygen* without arguments. In this case, it will prompt for the file in which to store keys. First, the tool asked where to save the file. SSH keys for user authentication are usually stored in the users .ssh directory under the home directory.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



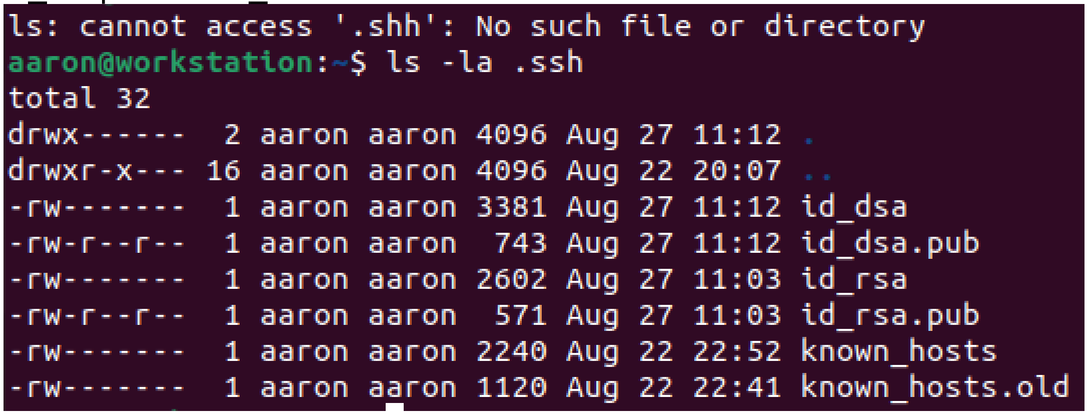
```
aaron@workstation:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/aaron/.ssh/id_rsa): /home/aaron
/id_rsa
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/aaron/.ssh/id_rsa
Your public key has been saved in /home/aaron/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:Qkeyvcn7yS+NsD5q7N6a4uA0hd+P28wl41WP6SvqNxc aaron@workstation
The key's randomart image is:
+---[RSA 3072]----+
|      . .        |
|       =         |
|      o o        |
|   . . o o       |
|  . . . S   .    |
|   o . ... .E+   |
|  + ... +oooo..  |
| o o. oXoB*oo    |
|  ...=O*@++*+.   |
+----[SHA256]-----+
aaron@workstation:~$
```

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.



```
aaron@workstation:~$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/aaron/.ssh/id_rsa): /home/aaron/.ssh
/id_dsa
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/aaron/.ssh/id_dsa
Your public key has been saved in /home/aaron/.ssh/id_dsa.pub
The key fingerprint is:
SHA256:l1UeOm1tjYQhblgE6Bgdpki0HUAM+TVZ17LqCTDDc+A aaron@workstation
The key's randomart image is:
+---[RSA 4096]----+
|.*=..++oo++ .o+  |
|..o+=+o..+...* +.|
| +ooo=  .oo + = +|
|  E o . .. o o . |
|   *   .S o      |
|    . .  .       |
|     o .         |
|      o          |
|                 |
+----[SHA256]-----+
```

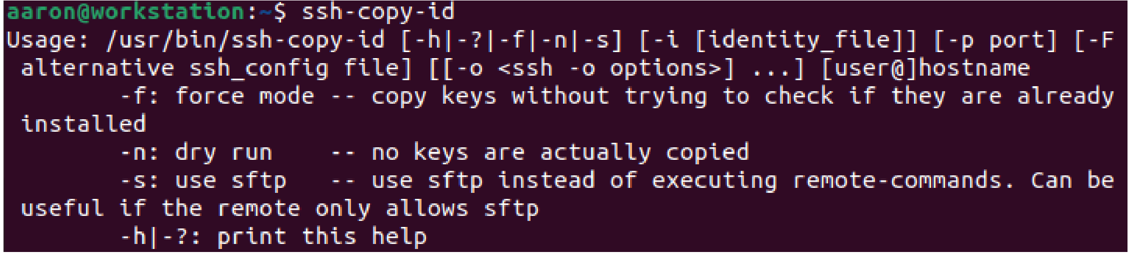
4. Verify that you have created the key by issuing the command *ls -la .ssh.* The command should show the .ssh directory containing a pair of keys. For example, id_rsa.pub and id_rsa.



```
ls: cannot access '.shh': No such file or directory
aaron@workstation:~$ ls -la .ssh
total 32
drwx------  2 aaron aaron 4096 Aug 27 11:12 .
drwxr-x--- 16 aaron aaron 4096 Aug 22 20:07 ..
-rw-------  1 aaron aaron 3381 Aug 27 11:12 id_dsa
-rw-r--r--  1 aaron aaron  743 Aug 27 11:12 id_dsa.pub
-rw-------  1 aaron aaron 2602 Aug 27 11:03 id_rsa
-rw-r--r--  1 aaron aaron  571 Aug 27 11:03 id_rsa.pub
-rw-------  1 aaron aaron 2240 Aug 22 22:52 known_hosts
-rw-------  1 aaron aaron 1120 Aug 22 22:41 known_hosts.old
```

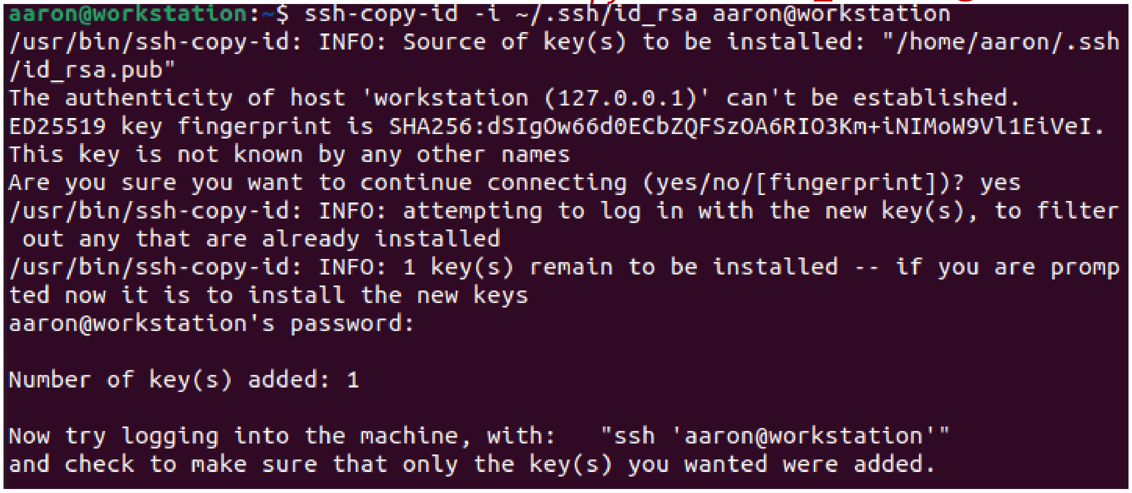
Task 2: Copying the Public Key to the remote servers

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.



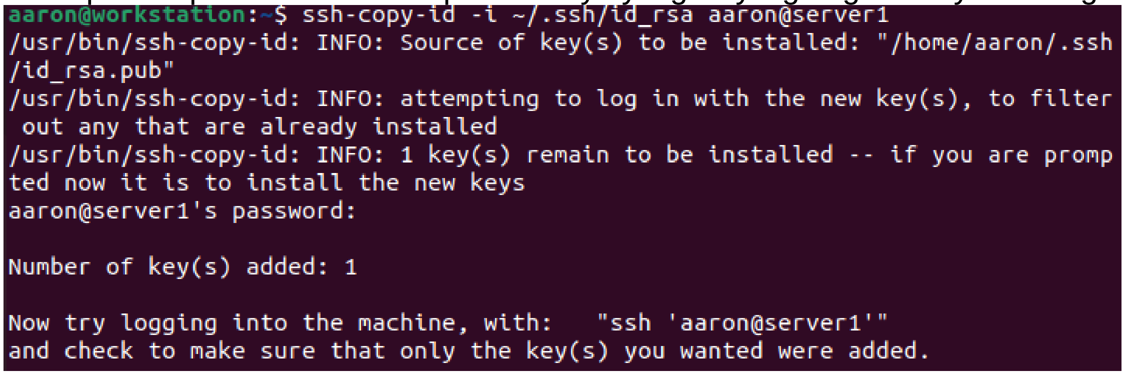
```
aaron@workstation:~$ ssh-copy-id
Usage: /usr/bin/ssh-copy-id [-h|-?|-f|-n|-s] [-i [identity_file]] [-p port] [-F
 alternative ssh_config file] [[-o <ssh -o options>] ...] [user@]hostname
        -f: force mode -- copy keys without trying to check if they are already
 installed
        -n: dry run    -- 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*

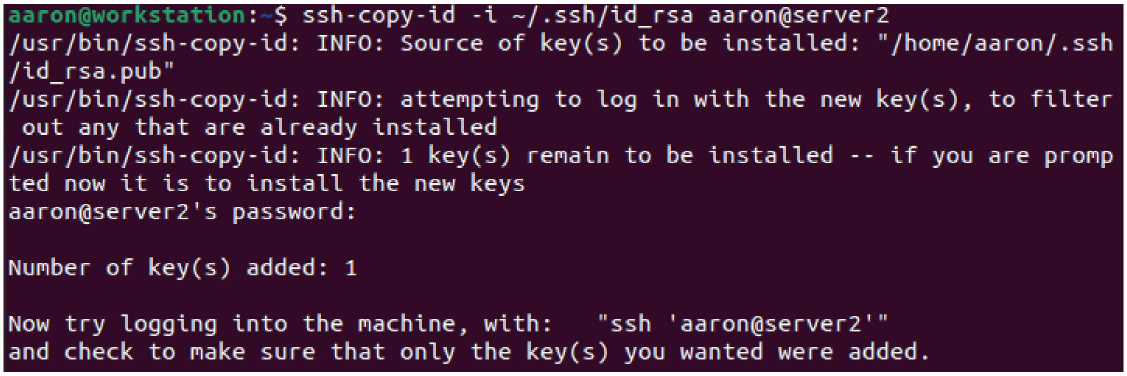


```
aaron@workstation:~$ ssh-copy-id -i ~/.ssh/id_rsa aaron@workstation
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/aaron/.ssh
/id_rsa.pub"
The authenticity of host 'workstation (127.0.0.1)' can't be established.
ED25519 key fingerprint is SHA256:dSIgOw66d0ECbZQFSzOA6RIO3Km+iNIMoW9Vl1EiVeI.
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 promp
ted now it is to install the new keys
aaron@workstation's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'aaron@workstation'"
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.

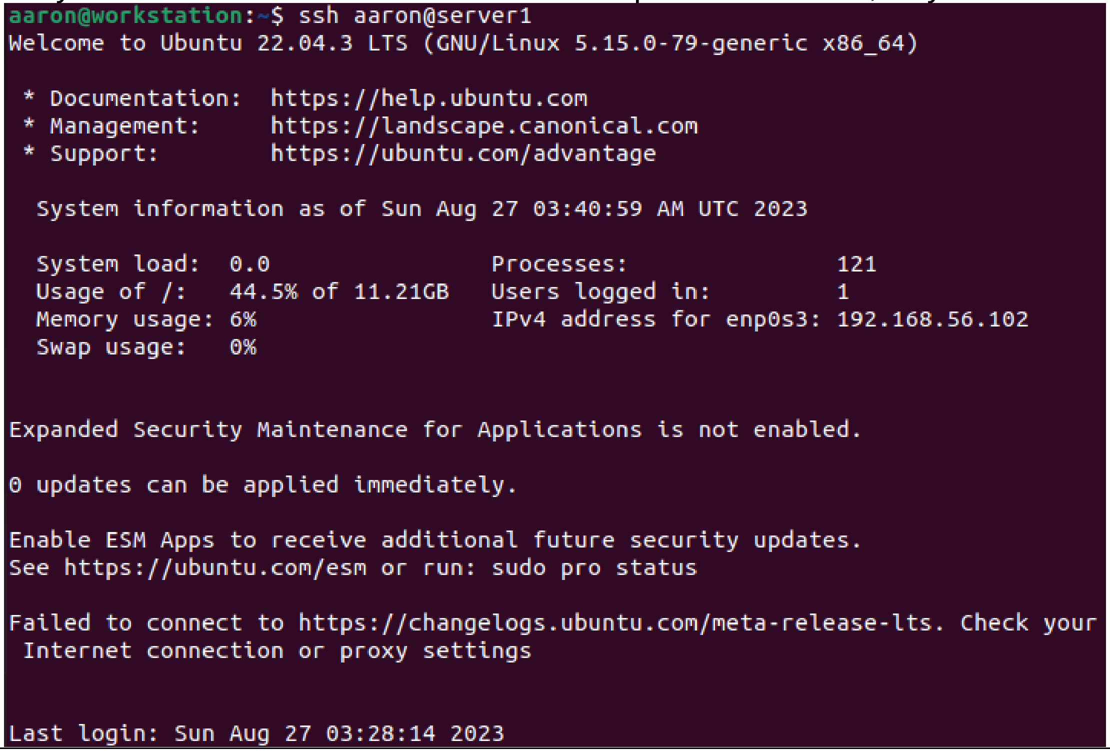


```
aaron@workstation:~$ ssh-copy-id -i ~/.ssh/id_rsa aaron@server1
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/aaron/.ssh
/id_rsa.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter
 out any that are already installed
/usr/bin/ssh-copy-id: INFO: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

Number of key(s) added: 1

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

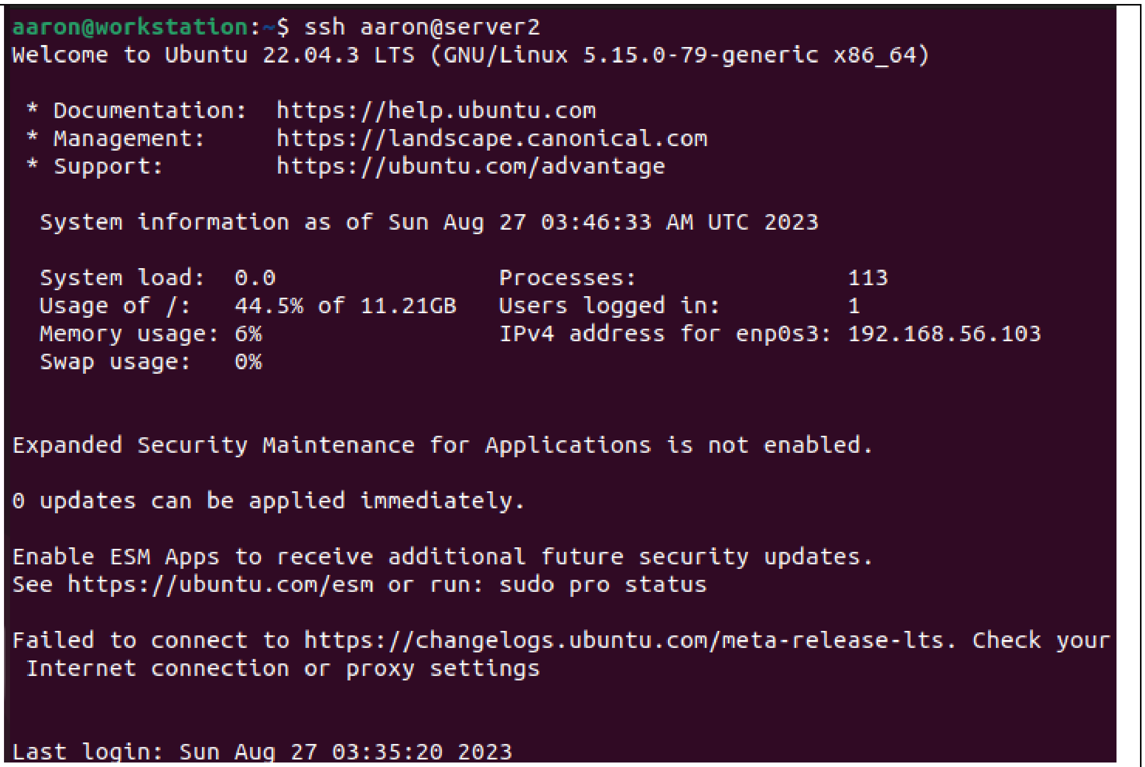


```
aaron@workstation:~$ ssh-copy-id -i ~/.ssh/id_rsa aaron@server2
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/aaron/.ssh
/id_rsa.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter
 out any that are already installed
/usr/bin/ssh-copy-id: INFO: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

Number of key(s) added: 1

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



```
aaron@workstation:~$ ssh aaron@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 03:40:59 AM UTC 2023

  System load:  0.0                Processes:               121
  Usage of /:   44.5% of 11.21GB   Users logged in:         1
  Memory usage: 6%                 IPv4 address for enp0s3: 192.168.56.102
  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

Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your
 Internet connection or proxy settings


Last login: Sun Aug 27 03:28:14 2023
```



```
aaron@workstation:~$ ssh aaron@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 03:46:33 AM UTC 2023

  System load:  0.0               Processes:               113
  Usage of /:   44.5% of 11.21GB  Users logged in:         1
  Memory usage: 6%                IPv4 address for enp0s3: 192.168.56.103
  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

Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your
 Internet connection or proxy settings


Last login: Sun Aug 27 03:35:20 2023
```

No password was needed because of the inputted authorized_keys in the workstation

Reflections:

Answer the following:

1. **How will you describe the ssh-program? What does it do?**
   SSH (Secure Shell) is a network protocol and program that provides a secure way to access and manage remote computers or servers over a potentially unsecured network. It does this by encrypting the data transmitted between the client and server, ensuring confidentiality and integrity of the communication.

2. **How do you know that you already installed the public key to the remote servers?**
   You can verify that you've installed a public key on a remote server by attempting to connect to the server using SSH. If you're able to log in without being prompted for a password and instead authenticate with your private key, then your public key is correctly installed on the server. You can also check the server's authorized_keys file in the ~/.ssh directory to ensure your public key is listed there 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**).

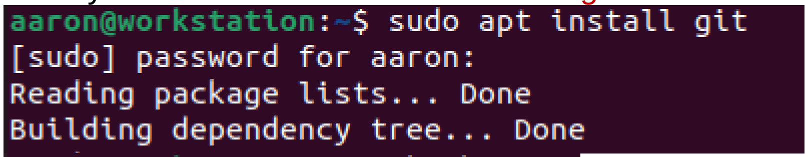
Set up Git

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:

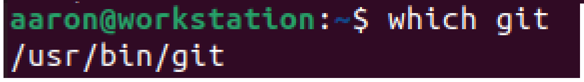
- Creating a repository
- Forking a repository
- Managing files
- Being social

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

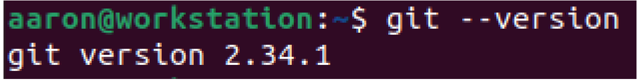


```
aaron@workstation:~$ sudo apt install git
[sudo] password for aaron:
Reading package lists... Done
Building dependency tree... Done
```



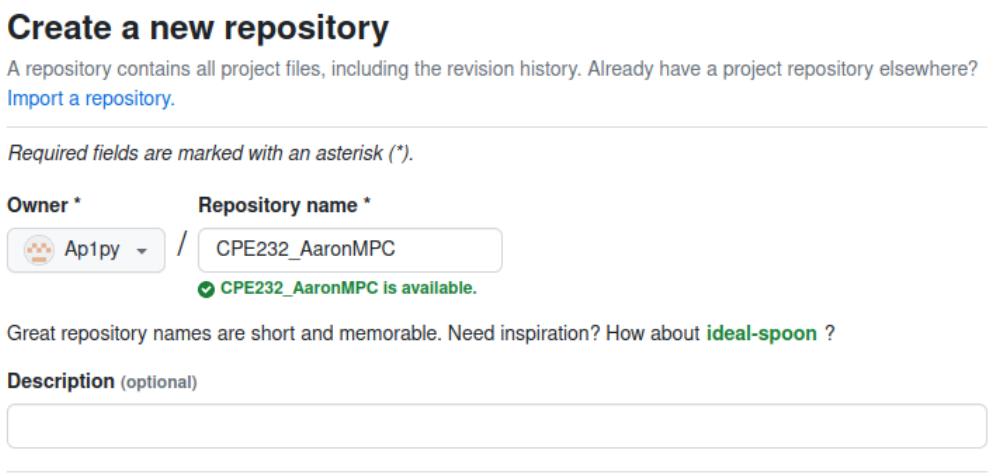
```
aaron@workstation:~$ which git
/usr/bin/git
```

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



```
aaron@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.



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 * / Repository name *

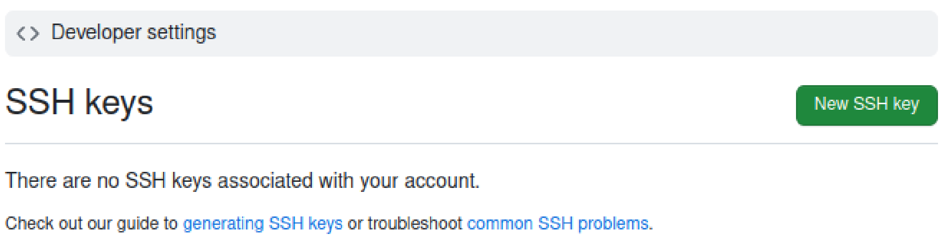
Ap1py / CPE232_AaronMPC

CPE232_AaronMPC is available.

Great repository names are short and memorable. Need inspiration? How about **ideal-spoon** ?

Description (optional)

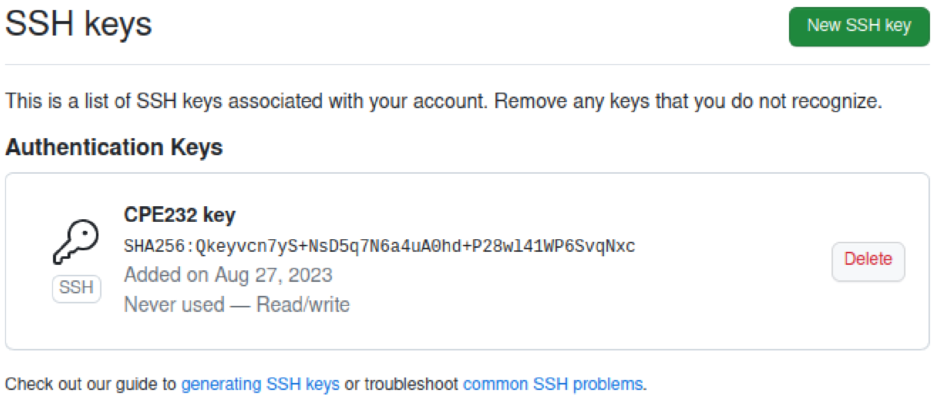
b. Create a new SSH key on GitHub. Go your profile's setting and click SSH and GPG keys. If there is an existing key, make sure to delete it. To create a new SSH keys, click New SSH Key. Write CPE232 key as the title of the key.



SSH keys

There are no SSH keys associated with your account.

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



SSH keys

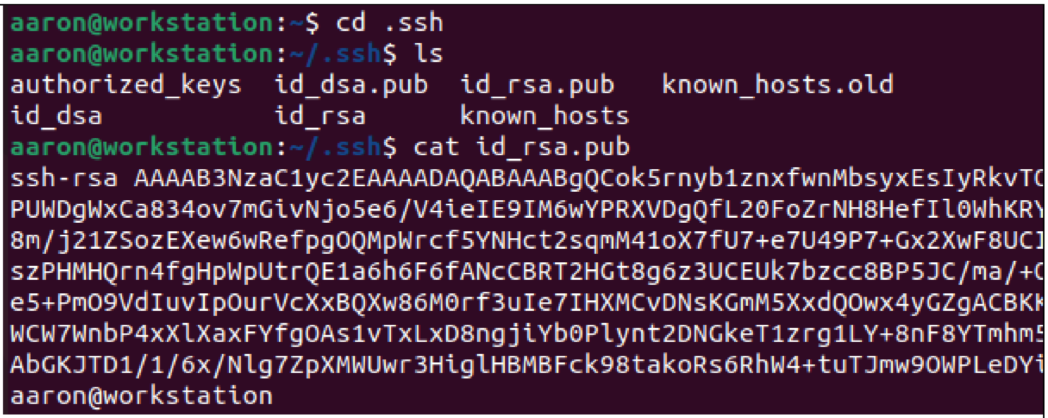
This is a list of SSH keys associated with your account. Remove any keys that you do not recognize.

Authentication Keys

CPE232 key
SHA256:Qkeyvcn7yS+NsD5q7N6a4uA0hd+P28wl41WP6SvqNxc
Added on Aug 27, 2023
Never used — Read/write

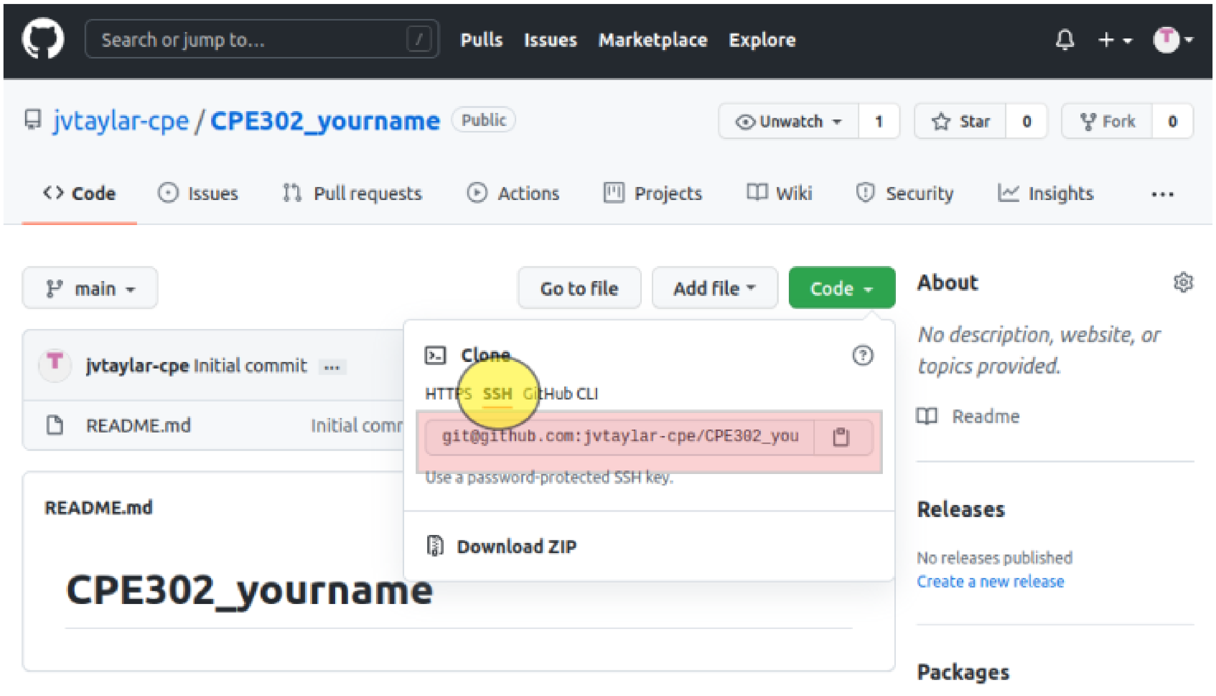
Check out our guide to generating SSH keys or troubleshoot common SSH problems.

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.



```
aaron@workstation:~$ cd .ssh
aaron@workstation:~/.ssh$ ls
authorized_keys  id_dsa.pub  id_rsa.pub   known_hosts.old
id_dsa           id_rsa      known_hosts
aaron@workstation:~/.ssh$ cat id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQCok5rnyb1znxfwnMbsyxEsIyRkvTC
PUWDgWxCa834ov7mGivNjo5e6/V4ieIE9IM6wYPRXVDgQfL20FoZrNH8HefIl0WhKRY
8m/j21ZSozEXew6wRefpgOQMpWrcf5YNHct2sqmM41oX7fU7+e7U49P7+Gx2XwF8UCI
szPHMHQrn4fgHpWpUtrQE1a6h6F6fANcCBRT2HGt8g6z3UCEUk7bzcc8BP5JC/ma/+C
e5+PmO9VdIuvIpOurVcXxBQXw86M0rf3uIe7IHXMCvDNsKGmM5XxdQOwx4yGZgACBKK
WCW7WnbP4xXlXaxFYfgOAs1vTxLxD8ngjiYb0Plynt2DNGkeT1zrg1LY+8nF8YTmhm5
AbGKJTD1/1/6x/Nlg7ZpXMWUwr3HiglHBMBFck98takoRs6RhW4+tuTJmw9OWPLeDYi
aaron@workstation
```

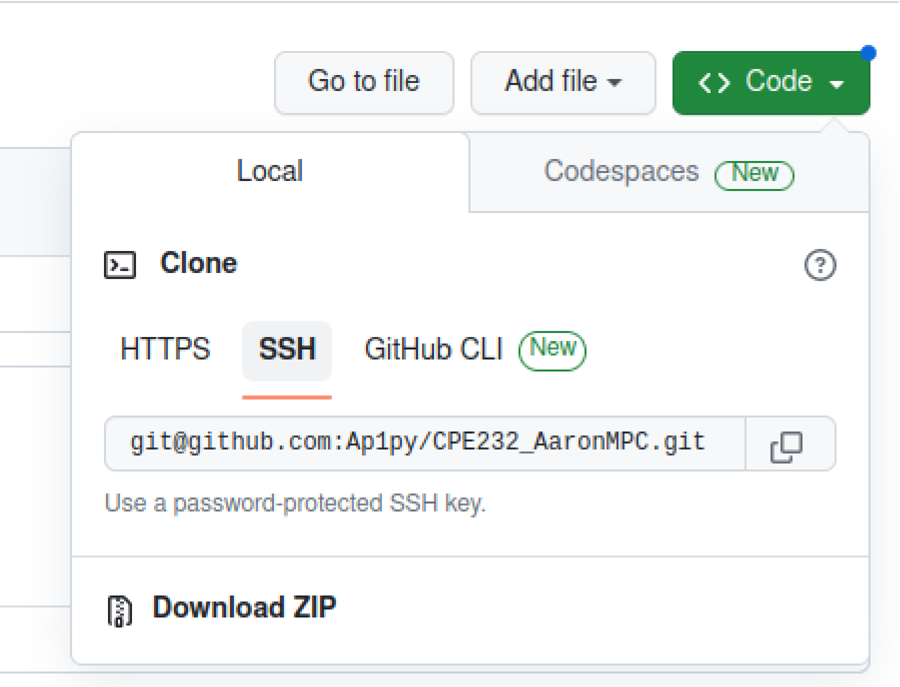
d. Clone the repository that you created. In doing this, you need to get the link from GitHub. Browse to your repository as shown below. Click on the Code drop down menu. Select SSH and copy the link.



git@github.com:jvtaylar-cpe/CPE302_you

Use a password-protected SSH key.

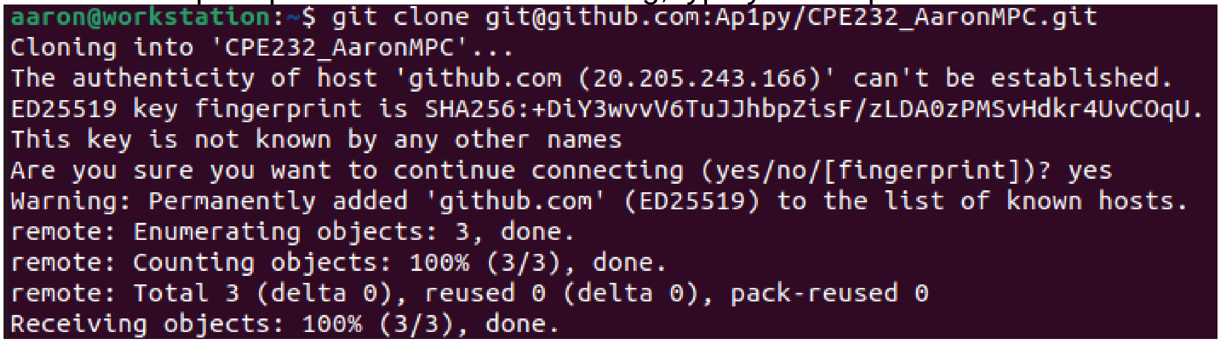
No description, website, or topics provided.



```
git@github.com:Ap1py/CPE232_AaronMPC.git
```

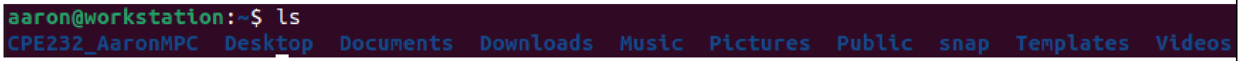
Use a password-protected SSH key.

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.



```
aaron@workstation:~$ git clone git@github.com:Ap1py/CPE232_AaronMPC.git
Cloning into 'CPE232_AaronMPC'...
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.
```

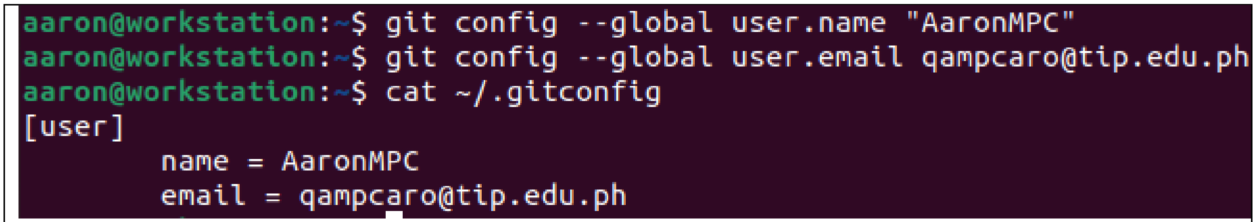
f. To verify that you have cloned the GitHub repository, issue the command *ls*. Observe that you have the CPE232_yourname in the list of your directories. Use CD command to go to that directory and LS command to see the file README.md.



```
aaron@workstation:~$ ls
CPE232_AaronMPC  Desktop  Documents  Downloads  Music  Pictures  Public  snap  Templates  Videos
```

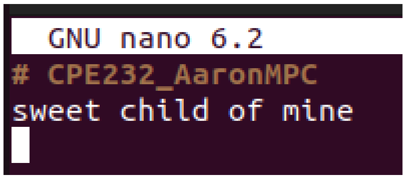
g. Use the following commands to personalize your git.

- *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*



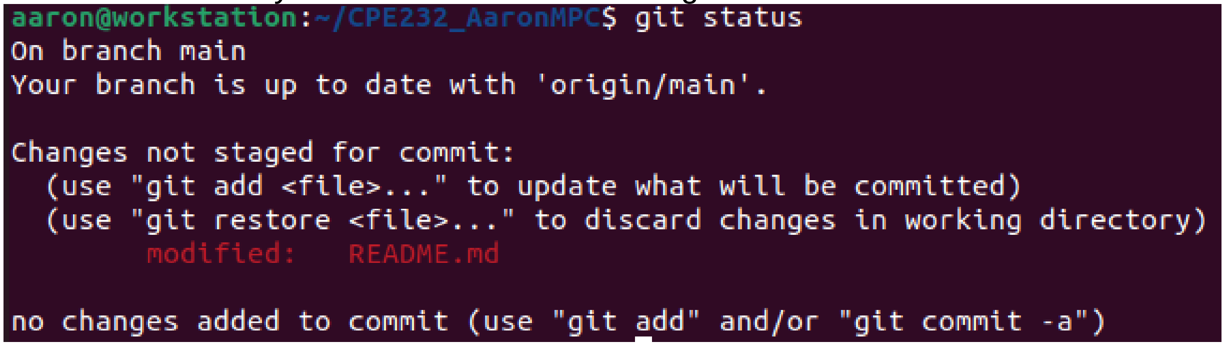
```
aaron@workstation:~$ git config --global user.name "AaronMPC"
aaron@workstation:~$ git config --global user.email qampcaro@tip.edu.ph
aaron@workstation:~$ cat ~/.gitconfig
[user]
        name = AaronMPC
        email = qampcaro@tip.edu.ph
```

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
# CPE232_AaronMPC
sweet child of mine
```

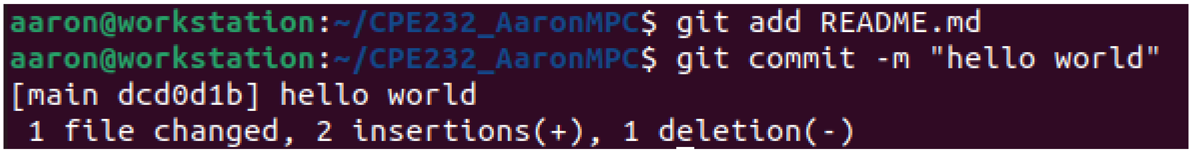
i. Use the *git status* command to display the state of the working directory and the staging area. This command shows which changes have been staged, which haven't, and which files aren't being tracked by Git. Status output does not show any information regarding the committed project history. What is the result of issuing this command?



```
aaron@workstation:~/CPE232_AaronMPC$ 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")
```

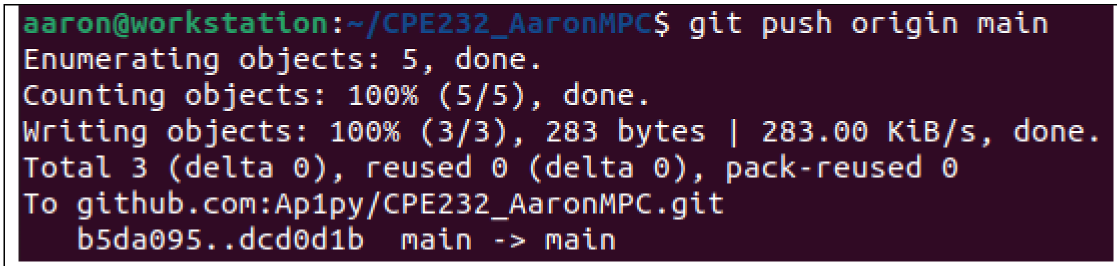
j. Use the command *git add README.md* to add the file into the staging area.

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.



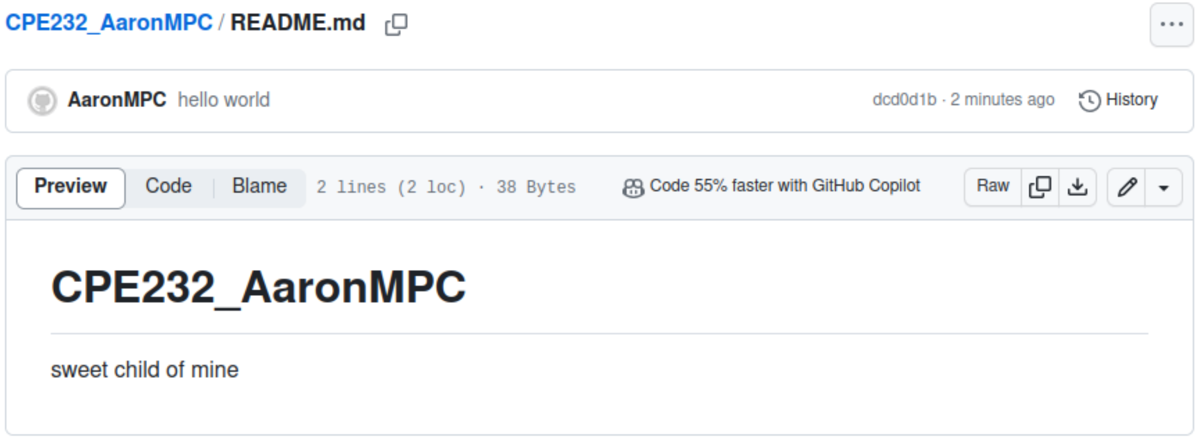
```
aaron@workstation:~/CPE232_AaronMPC$ git add README.md
aaron@workstation:~/CPE232_AaronMPC$ git commit -m "hello world"
[main dcd0d1b] hello world
 1 file changed, 2 insertions(+), 1 deletion(-)
```

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.*



```
aaron@workstation:~/CPE232_AaronMPC$ git push origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Writing objects: 100% (3/3), 283 bytes | 283.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:Ap1py/CPE232_AaronMPC.git
   b5da095..dcd0d1b  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_AaronMPC / README.md
History
Preview
Code
Blame
Code 55% faster with GitHub Copilot
Raw

AaronMPC hello world

dcd0d1b · 2 minutes ago

2 lines (2 loc) · 38 Bytes

CPE232_AaronMPC

sweet child of mine

Reflections:

Answer the following:

3. **What sort of things have we so far done to the remote servers using ansible commands?**
   The utilization of a pair of private and public keys enables the host to access the server via SSH without the need for a password, facilitated by the use of authorized keys.
4. **How important is the inventory file?**
   The inventory file serves as a reference for ansible commands, specifying the hosts to connect with and execute actions on. It acts as a repository for commands, allowing for reuse, and once activated, the commands can be readily identified.

Conclusions/Learnings:

During this exercise, I gained the knowledge of establishing passwordless SSH connections through servers using both public and private keys. Additionally, I successfully linked my Ubuntu server to GitHub, enabling me to edit the README.md file directly from the Ubuntu Desktop. This experience has shown me the extensive capabilities of Ubuntu, exceeding my initial expectations.