Name: Seruelas, Ronn Kristoper H.	**Date Performed:** 08-22-2023
Course/Section: CPE31S4	**Date Submitted:** 08-29-2023
Instructor: Dr. Jonathan V. Taylar	**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.

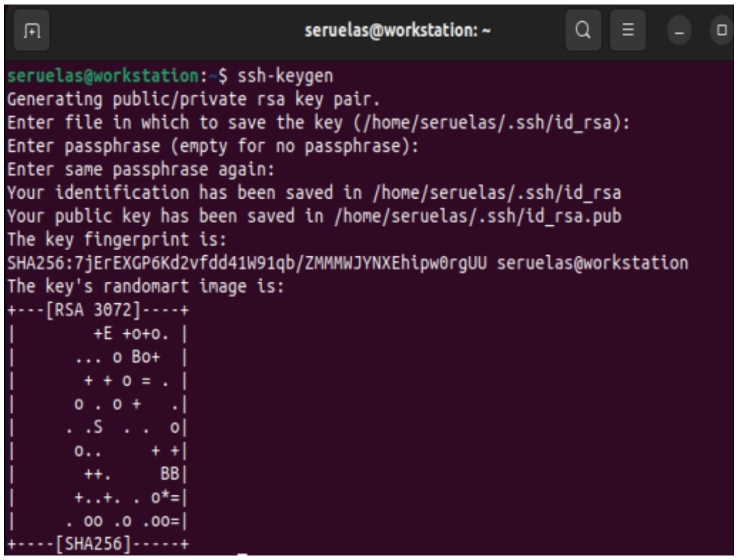
SSH introduced public key authentication as a more secure alternative to the older .rhosts authentication. It improved security by avoiding the need to have password stored in files and eliminated the possibility of a compromised server stealing the user's password.

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,

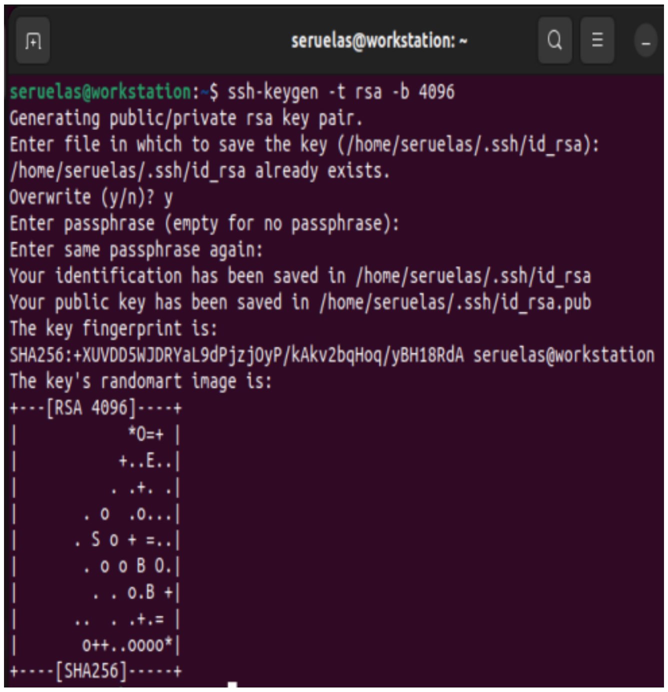
the tool asked where to save the file. SSH keys for user authentication are usually stored in the users .ssh directory under the home directory. However, in enterprise environments, the location is often different. The default key file name depends on the algorithm, in this case *id_rsa* when using the default RSA algorithm. It could also be, for example, *id_dsa* or *id_ecdsa*.



```
seruelas@workstation:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/seruelas/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/seruelas/.ssh/id_rsa
Your public key has been saved in /home/seruelas/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:7jErEXGP6Kd2vfdd41W91qb/ZMMMWJYNXEhipw0rgUU seruelas@workstation
The key's randomart image is:
+---[RSA 3072]----+
|        +E +o+o. |
|      ... o Bo+  |
|       + + o = . |
|      o . o +   .|
|     . .S  . .  o|
|      o..     + +|
|       ++.     BB|
|      +..+. . o*=|
|     . oo .o .oo=|
+----[SHA256]-----+
```

Figure 1.1 - Generating an RSA Encrypted SSH Key.

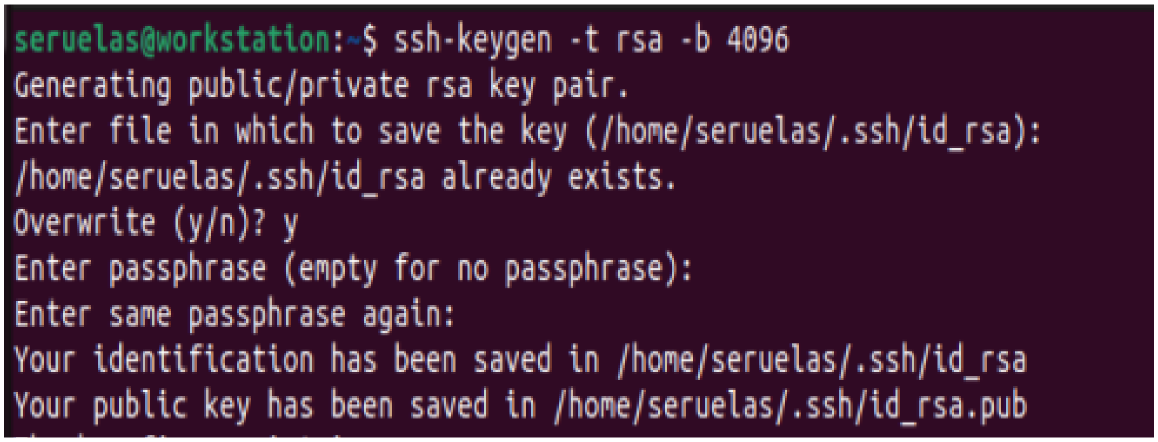
2. Issue the command *ssh-keygen -t rsa -b 4096.* The algorithm is selected using the -t option and key size using the -b option.



```
seruelas@workstation:~$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/seruelas/.ssh/id_rsa):
/home/seruelas/.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/seruelas/.ssh/id_rsa
Your public key has been saved in /home/seruelas/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:+XUVDD5WJDRYaL9dPjzjOyP/kAkv2bqHoq/yBH18RdA seruelas@workstation
The key's randomart image is:
+---[RSA 4096]----+
|            *O=+ |
|           +..E..|
|          . .+. .|
|       . o  .o...|
|      . S o + =..|
|       . o o B O.|
|        . . o.B +|
|      ..  . .+.= |
|       o++..oooo*|
+----[SHA256]-----+
```

Figure 2.1 - Generating a new RSA Encrypted Key with 4096 bits

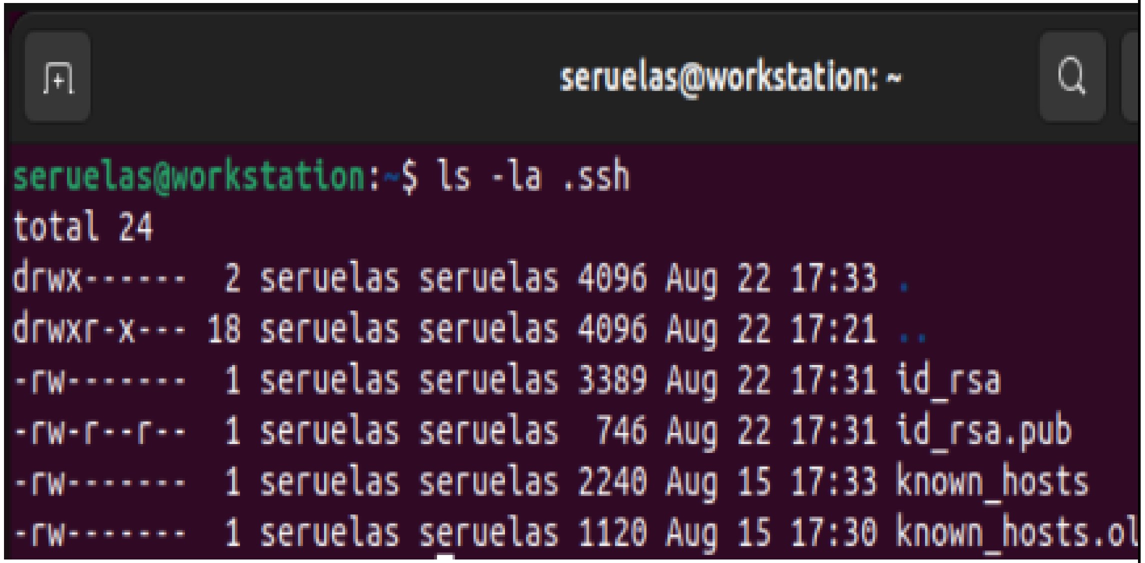
3. When asked for a passphrase, just press enter. The passphrase is used for encrypting the key, so that it cannot be used even if someone obtains the private key file. The passphrase should be cryptographically strong.



```
seruelas@workstation:~$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/seruelas/.ssh/id_rsa):
/home/seruelas/.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/seruelas/.ssh/id_rsa
Your public key has been saved in /home/seruelas/.ssh/id_rsa.pub
```

Figure 3.1 - Skipping or pressing enter when a passphrase is asked.

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.

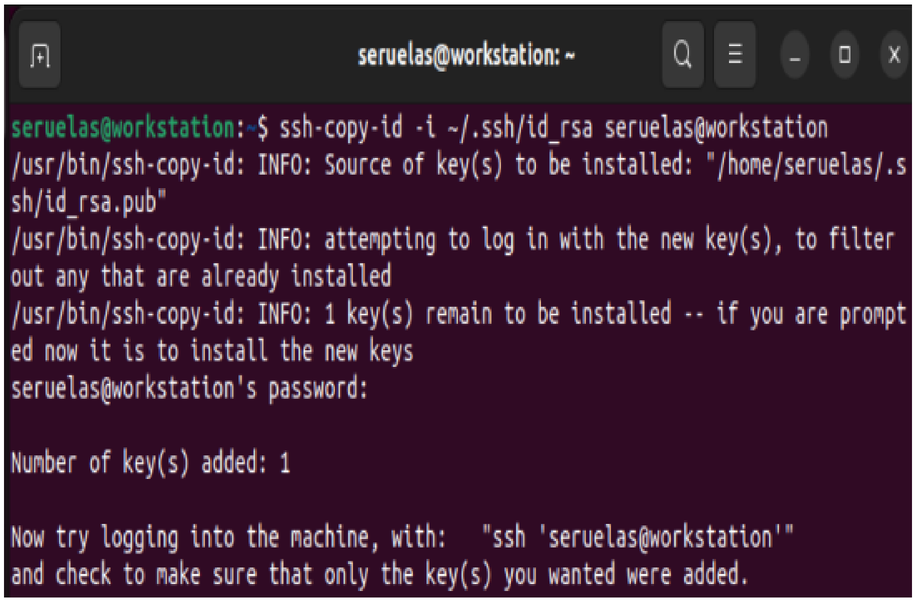


```
seruelas@workstation:~$ ls -la .ssh
total 24
drwx------  2 seruelas seruelas 4096 Aug 22 17:33 .
drwxr-x--- 18 seruelas seruelas 4096 Aug 22 17:21 ..
-rw-------  1 seruelas seruelas 3389 Aug 22 17:31 id_rsa
-rw-r--r--  1 seruelas seruelas  746 Aug 22 17:31 id_rsa.pub
-rw-------  1 seruelas seruelas 2240 Aug 15 17:33 known_hosts
-rw-------  1 seruelas seruelas 1120 Aug 15 17:30 known_hosts.ol
```

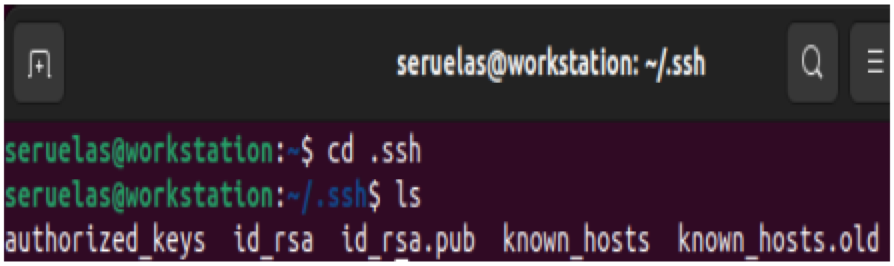
Figure 4.1 - Verification of key creation.

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.*



```
seruelas@workstation:~$ ssh-copy-id -i ~/.ssh/id_rsa seruelas@workstation
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/seruelas/.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
seruelas@workstation's password:

Number of key(s) added: 1

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

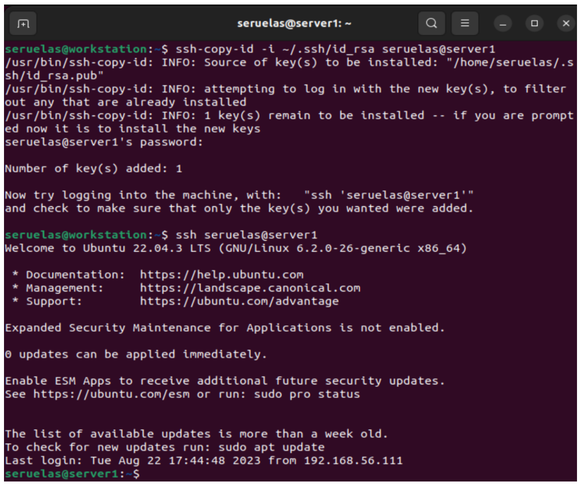


```
seruelas@workstation:~$ cd .ssh
seruelas@workstation:~/.ssh$ ls
authorized_keys  id_rsa  id_rsa.pub  known_hosts  known_hosts.old
```

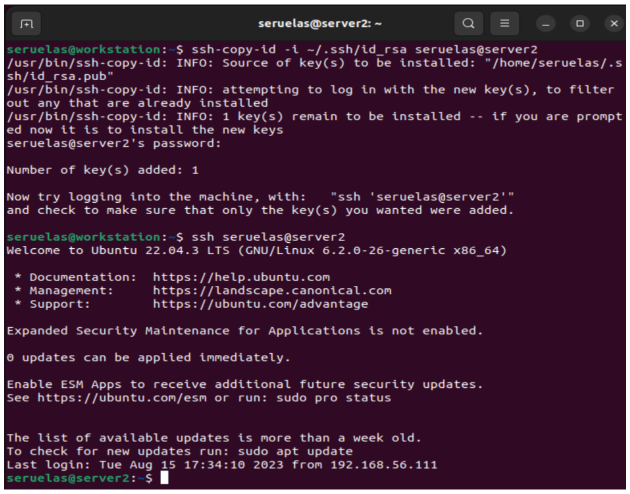
Figure 2.1-2.2 - Copying of public key to the server in an authorized_keys file.

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?



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

Number of key(s) added: 1

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

seruelas@workstation:~$ ssh seruelas@server1
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 list of available updates is more than a week old.
To check for new updates run: sudo apt update
Last login: Tue Aug 22 17:44:48 2023 from 192.168.56.111
seruelas@server1:~$
```



```
seruelas@workstation:~$ ssh-copy-id -i ~/.ssh/id_rsa seruelas@server2
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/seruelas/.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
seruelas@server2's password:

Number of key(s) added: 1

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

seruelas@workstation:~$ ssh seruelas@server2
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 list of available updates is more than a week old.
To check for new updates run: sudo apt update
Last login: Tue Aug 15 17:34:10 2023 from 192.168.56.111
seruelas@server2:~$
```

Figure 4.1-4.2 - Copying of SSH ID to Server1 and Server 2 and successfully logging in without authentication.

It was made possible to connect via SSH to the other servers after copying the ssh id as the SSH keys are authentication keys that once the host or server owns or has a copy of it, it allows the hostname or the host with the other pair of the authentication key without any other ways of authentication, allowing them to easily access.

Reflections:

Answer the following:

1. How will you describe the ssh-program? What does it do?
    - The ssh-program or secure shell program allows the users to remotely control or to remotely host another device by connecting via the ip addresses or network of each device. SSH-program or the secure shell also offers a deeper encryption for the users as to protect the user's data from any disruption or interference from other malicious users.
2. How do you know that you already installed the public key to the remote servers?
    - The way that a user can know when they have installed the public key to the remote servers is to verify or test it by connecting to them. The user can use the ***ssh*** command to connect to the remote servers, if they are able to connect, the users are able to install the public key to the remote servers successfully.

Part 2: Discussion

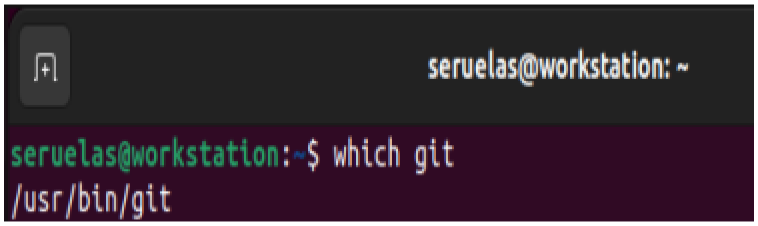
It is assumed that you are done with the last activity (**Activity 2: SSH Key-Based Authentication**).

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

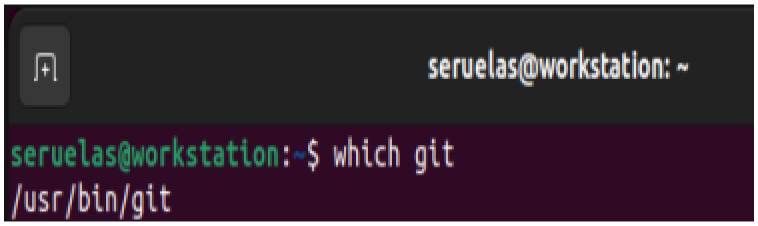
1. On the local machine, verify the version of your git using the command *which git.* If a directory of git is displayed, then you don't need to install git. Otherwise, to install git, use the following command: *sudo apt install git*



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

Figure 1.1 - Verification of installation of the git program.

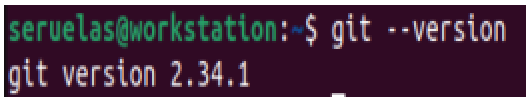
2. After the installation, issue the command *which git* again. The directory of git is usually installed in this location: *user/bin/git*.



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

Figure 2.1 - Verification of the location or the directory of installation of git.

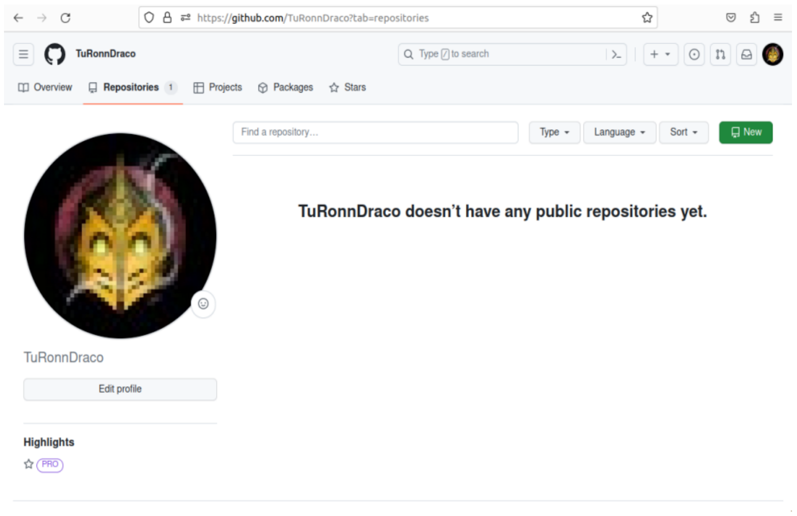
3. The version of git installed in your device is the latest. Try issuing the command *git --version* to know the version installed.



```
seruelas@workstation:~$ git --version
git version 2.34.1
```

Figure 3.1 - Verifying the version of git.

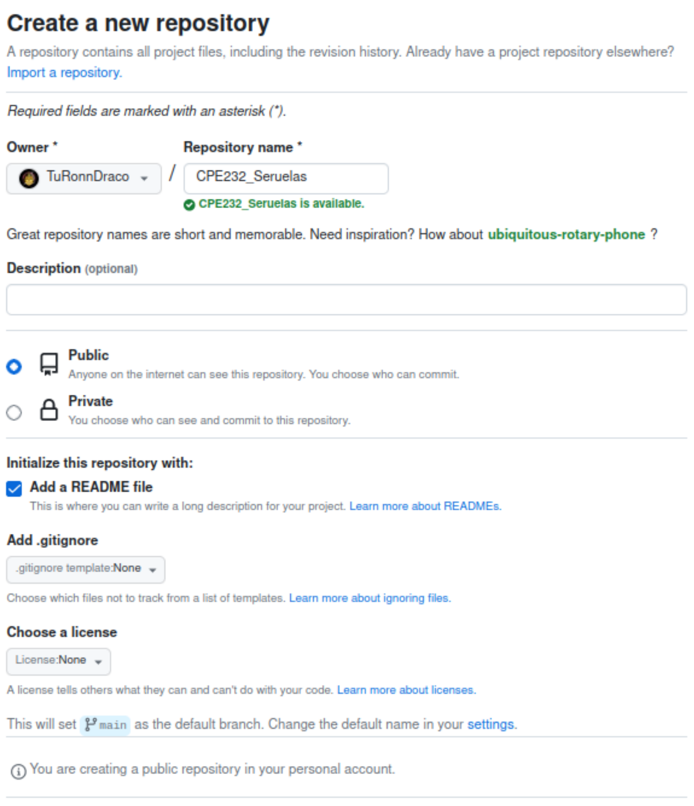
4. Using the browser in the local machine, go to www.github.com.



TuRonnDraco doesn't have any public repositories yet.

Figure 4.1 - Personal Github Page.

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 *: TuRonnDraco / Repository name *: CPE232_Seruelas

CPE232_Seruelas is available.

Great repository names are short and memorable. Need inspiration? How about **ubiquitous-rotary-phone** ?

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

Choose which files not to track from a list of templates. Learn more about ignoring files.

Choose a license

License: None

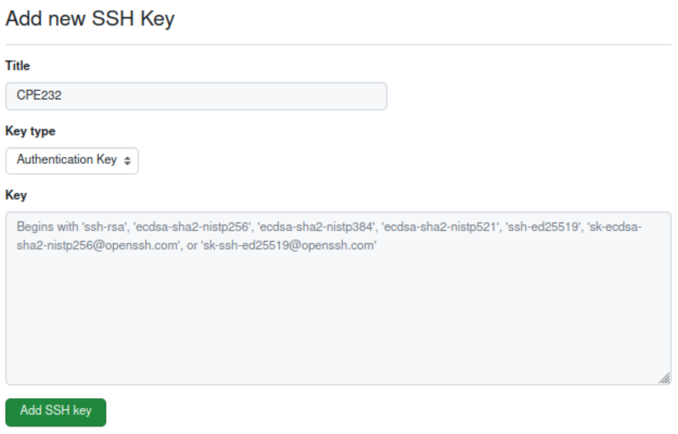
A license tells others what they can and can't do with your code. Learn more about licenses.

This will set `main` as the default branch. Change the default name in your settings.

You are creating a public repository in your personal account.

Figure 5a.1 - Creation of the CPE232 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.



Add new SSH Key

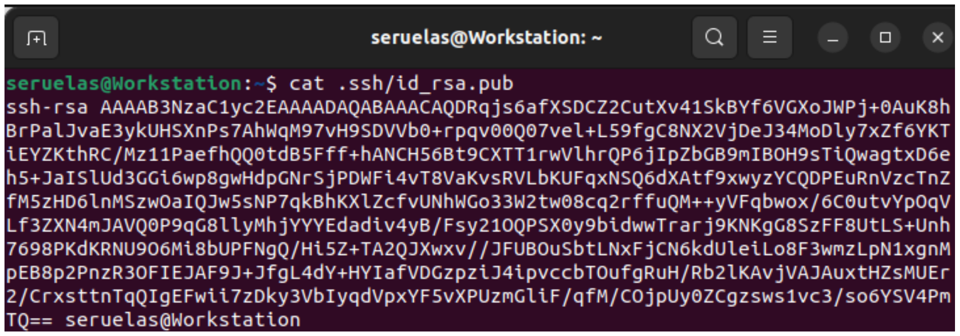
Title: CPE232

Key type: Authentication Key

Key: Begins with 'ssh-rsa', 'ecdsa-sha2-nistp256', 'ecdsa-sha2-nistp384', 'ecdsa-sha2-nistp521', 'ssh-ed25519', 'sk-ecdsa-sha2-nistp256@openssh.com', or 'sk-ssh-ed25519@openssh.com'

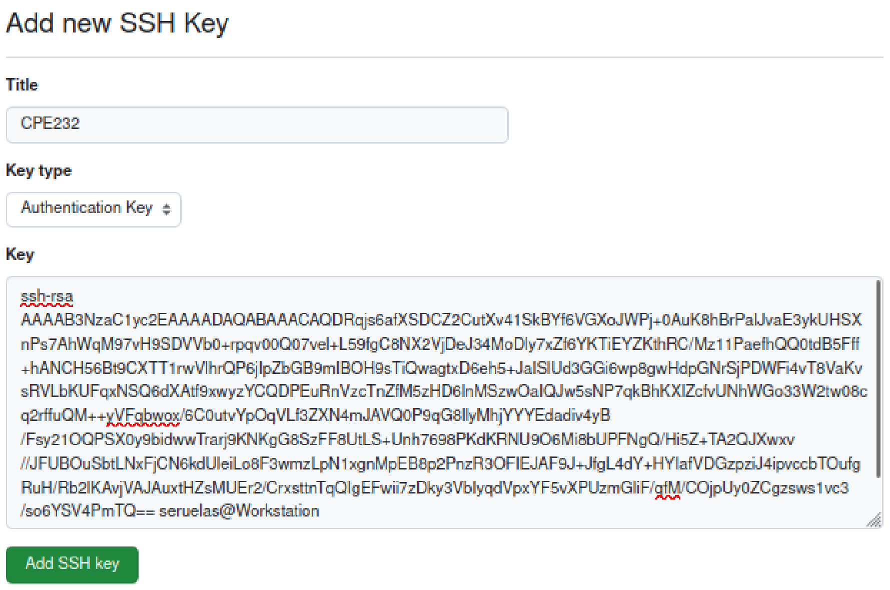
Figure 5b.1 - Creation of the CPE232 SSH Key.

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.



```
seruelas@Workstation:~$ cat .ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAACAQDRqjs6afXSDCZ2CutXv41SkBYf6VGXoJWPj+0AuK8h
BrPalJvaE3ykUHSXnPs7AhWqM97vH9SDVVb0+rpqv00Q07vel+L59fgC8NX2VjDeJ34MoDly7xZf6YKT
iEYZKthRC/Mz11PaefhQQ0tdB5Fff+hANCH56Bt9CXTT1rwVlhrQP6jIpZbGB9mIBOH9sTiQwagtxD6e
h5+JaISlUd3GGi6wp8gwHdpGNrSjPDWFi4vT8VaKvsRVLbKUFqxNSQ6dXAtf9xwyzYCQDPEuRnVzcTnZ
fM5zHD6lnMSzwOaIQJw5sNP7qkBhKXlZcfvUNhWGo33W2tw08cq2rffuQM++yVFqbwox/6C0utvYpOqV
Lf3ZXN4mJAVQ0P9qG8llyMhjYYYEdadiv4yB/Fsy21OQPSX0y9bidwwTrarj9KNKgG8SzFF8UtLS+Unh
7698PKdKRNU9O6Mi8bUPFNgQ/Hi5Z+TA2QJXwxv//JFUBOuSbtLNxFjCN6kdUleiLo8F3wmzLpN1xgnM
pEB8p2PnzR3OFIEJAF9J+JfgL4dY+HYIafVDGzpziJ4ipvccbTOufgRuH/Rb2lKAvjVAJAuxtHZsMUEr
2/CrxsttnTqQIgEFwii7zDky3VbIyqdVpxYF5vXPUzmGliF/qfM/COjpUy0ZCgzsws1vc3/so6YSV4Pm
TQ== seruelas@Workstation
```

Figure 5c.1 - Finding the public key of the workstation.



Add new SSH Key

Title

CPE232

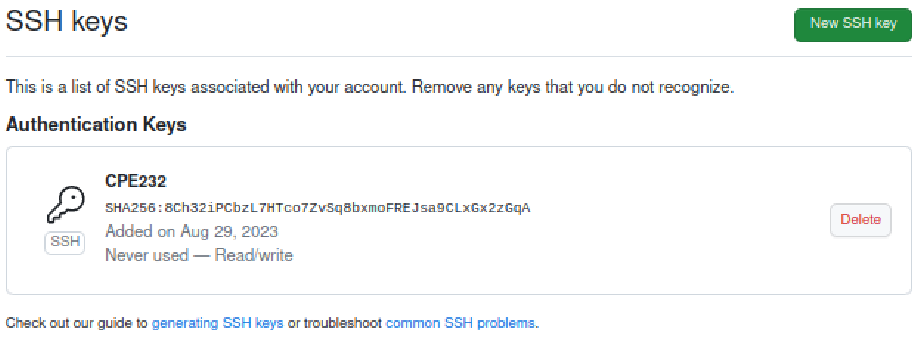
Key type

Authentication Key

Key

```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAACAQDRqjs6afXSDCZ2CutXv41SkBYf6VGXoJWPj+0AuK8hBrPalJvaE3ykUHSX
nPs7AhWqM97vH9SDVVb0+rpqv00Q07vel+L59fgC8NX2VjDeJ34MoDly7xZf6YKTiEYZKthRC/Mz11PaefhQQ0tdB5Fff
+hANCH56Bt9CXTT1rwVlhrQP6jlpZbGB9mIBOH9sTiQwagtxD6eh5+JaISlUd3GGi6wp8gwHdpGNrSjPDWFi4vT8VaKv
sRVLbKUFqxNSQ6dXAtf9xwyzYCQDPEuRnVzcTnZfM5zHD6lnMSzwOalQJw5sNP7qkBhKXIZcfvUNhWGo33W2tw08c
q2rffuQM++yVFqbwox/6C0utvYpOqVLf3ZXN4mJAVQ0P9qG8llyMhjYYYEdadiv4yB
/Fsy21OQPSX0y9bidwwTrarj9KNKgG8SzFF8UtLS+Unh7698PKdKRNU9O6Mi8bUPFNgQ/Hi5Z+TA2QJXwxv
//JFUBOuSbtLNxFjCN6kdUleiLo8F3wmzLpN1xgnMpEB8p2PnzR3OFIEJAF9J+JfgL4dY+HYlafVDGzpziJ4ipvccbTOufg
RuH/Rb2lKAvjVAJAuxtHZsMUEr2/CrxsttnTqQIgEFwii7zDky3VblyqdVpxYF5vXPUzmGliF/qfM/COjpUy0ZCgzsws1vc3
/so6YSV4PmTQ== seruelas@Workstation
```

Figure 5c.2 - Pasting public key in CPE232 SSH Key.



SSH keys

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

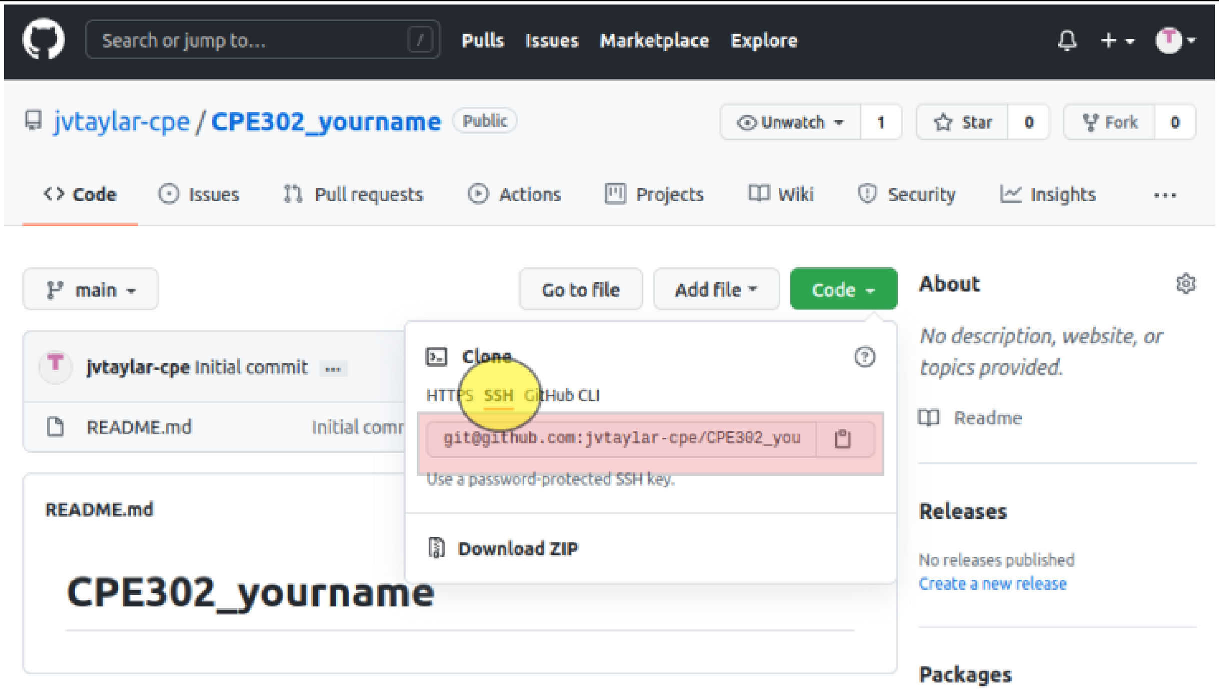
Authentication Keys

CPE232
`SHA256:8Ch32iPCbzL7HTco7ZvSq8bxmoFREJsa9CLxGx2zGqA`
Added on Aug 29, 2023
Never used — Read/write

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

Figure 5c.3 - Creation of CPE232 SSH Key.

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.



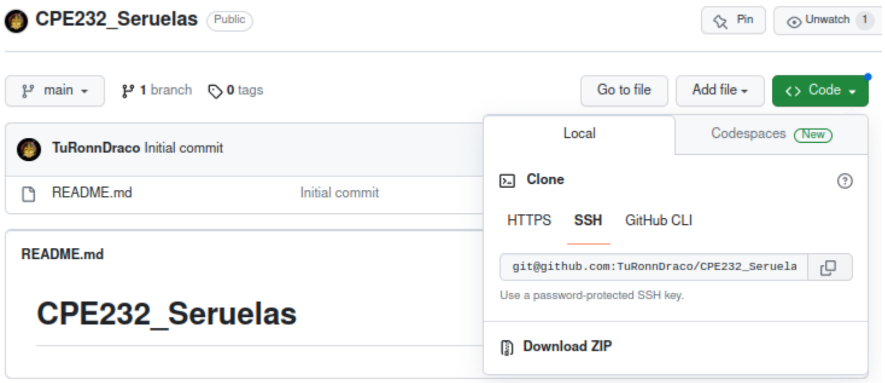
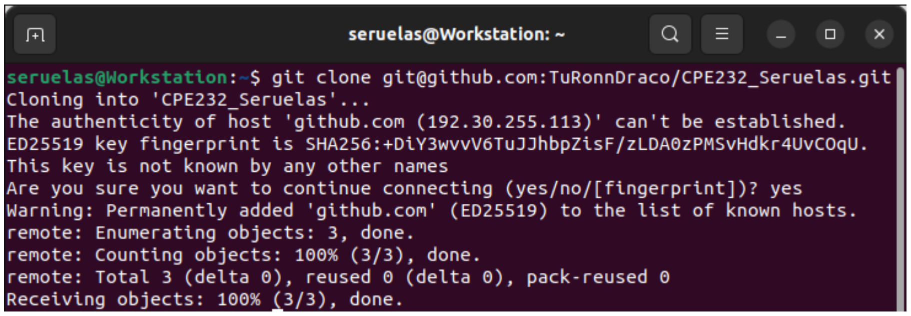


Figure 5d.1 - Getting SSH Code for Cloning from Repository.

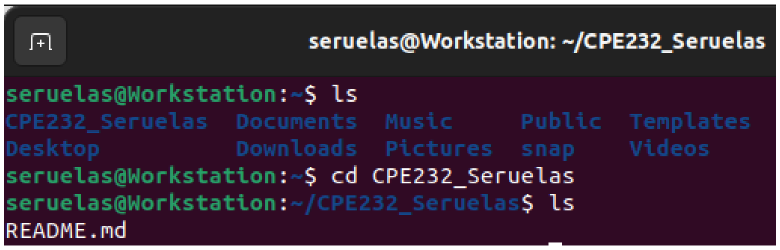
e. Issue the command git clone followed by the copied link. For example, *git clone* *git@github.com:jvtaylar-cpe/CPE232_yourname.git*. When prompted to continue connecting, type yes and press enter.



```
seruelas@Workstation:~$ git clone git@github.com:TuRonnDraco/CPE232_Seruelas.git
Cloning into 'CPE232_Seruelas'...
The authenticity of host 'github.com (192.30.255.113)' 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.
```

Figure 5e.1 - Cloning the Repository with the Workstation terminal.

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.

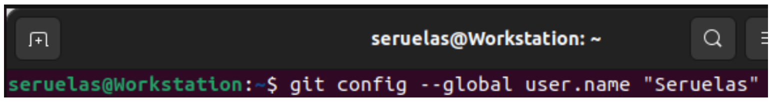


```
seruelas@Workstation: ~/CPE232_Seruelas
seruelas@Workstation:~$ ls
CPE232_Seruelas  Documents  Music     Public  Templates
Desktop          Downloads  Pictures  snap    Videos
seruelas@Workstation:~$ cd CPE232_Seruelas
seruelas@Workstation:~/CPE232_Seruelas$ ls
README.md
```

Figure 5f.1 - Verifying the successful cloning via **ls** and **cd**.

g. Use the following commands to personalize your git.

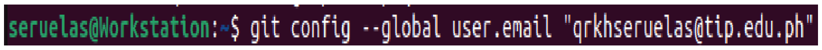
- *git config --global user.name "Your Name"*



```
seruelas@Workstation: ~
seruelas@Workstation:~$ git config --global user.name "Seruelas"
```

Figure 5g.1 - Configuring username to student name.

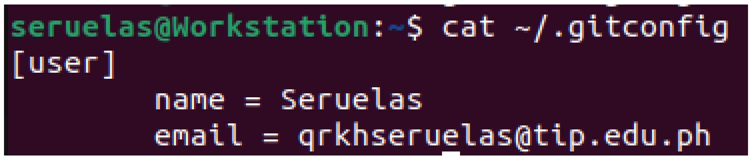
- *git config --global user.email yourname@email.com*



```
seruelas@Workstation:~$ git config --global user.email "qrkhseruelas@tip.edu.ph"
```

Figure 5g.2 - Configure user email to github email.

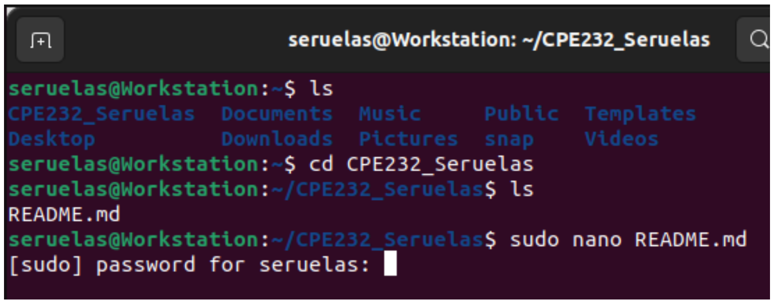
- Verify that you have personalized the config file using the command *cat ~/.gitconfig*



```
seruelas@Workstation:~$ cat ~/.gitconfig
[user]
        name = Seruelas
        email = qrkhseruelas@tip.edu.ph
```

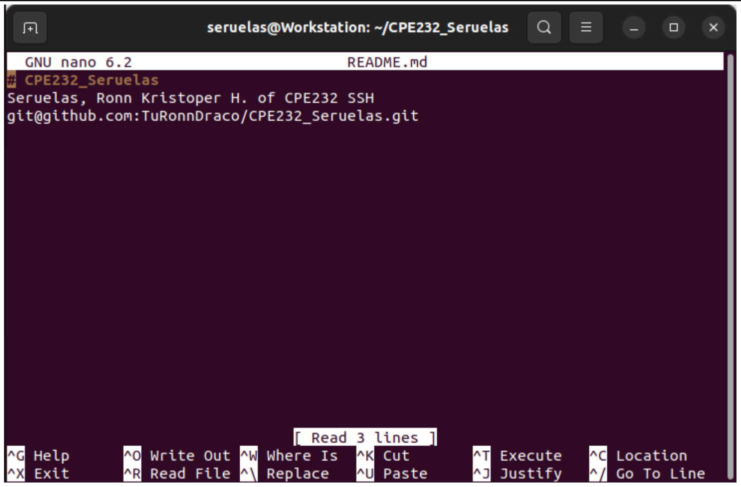
Figure 5g.3 - Verifying the personalized config file via **cat** command.

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.



```
seruelas@Workstation: ~/CPE232_Seruelas
seruelas@Workstation:~$ ls
CPE232_Seruelas  Documents  Music     Public  Templates
Desktop          Downloads  Pictures  snap    Videos
seruelas@Workstation:~$ cd CPE232_Seruelas
seruelas@Workstation:~/CPE232_Seruelas$ ls
README.md
seruelas@Workstation:~/CPE232_Seruelas$ sudo nano README.md
[sudo] password for seruelas: 
```

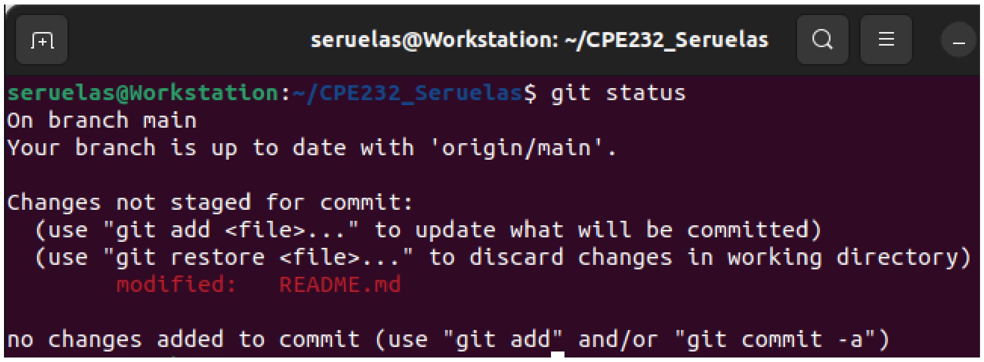
Figure 5h.1 - Editing the README.md via **nano** command.



```
GNU nano 6.2                              README.md
# CPE232_Seruelas
Seruelas, Ronn Kristoper H. of CPE232 SSH
git@github.com:TuRonnDraco/CPE232_Seruelas.git
```

Figure 5h.2 - Providing information in the README.md and saving file.

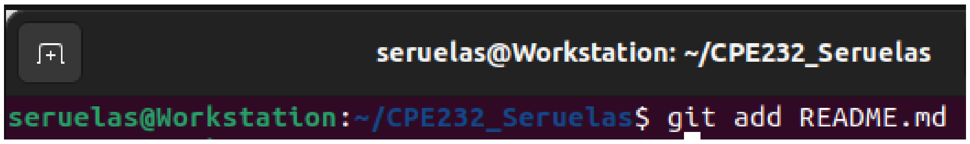
i. Use the *git status* command to display the state of the working directory and the staging area. This command shows which changes have been staged, which haven't, and which files aren't being tracked by Git. Status output does not show any information regarding the committed project history. What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



```
seruelas@Workstation:~/CPE232_Seruelas$ 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")
```

Figure 5i.1 - Upon issuing the **git status** command, it shows the users the history and the branching of the working directory or staging area, pertaining to the cloned directory from the repository.

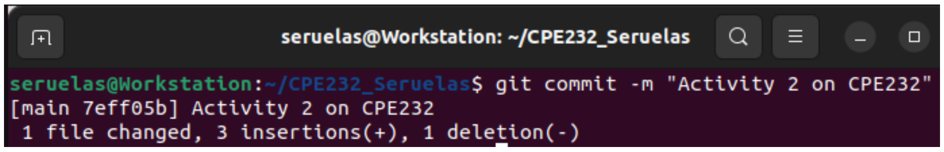
j. Use the command *git add README.md* to add the file into the staging area.



```
seruelas@Workstation:~/CPE232_Seruelas$ git add README.md
```

Figure 5j.1 - Using the **git add** to add the file into the staging area.

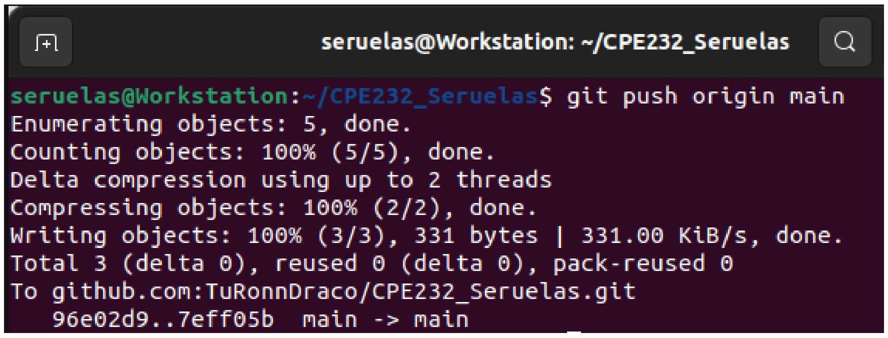
k. Use the *git commit -m “your message”* to create a snapshot of the staged changes along the timeline of the Git projects history. The use of this command is required to select the changes that will be staged for the next commit.



```
seruelas@Workstation: ~/CPE232_Seruelas
seruelas@Workstation:~/CPE232_Seruelas$ git commit -m "Activity 2 on CPE232"
[main 7eff05b] Activity 2 on CPE232
 1 file changed, 3 insertions(+), 1 deletion(-)
```

Figure 5k.1 - Using the **git commit** command to create a snapshot for the staging area.

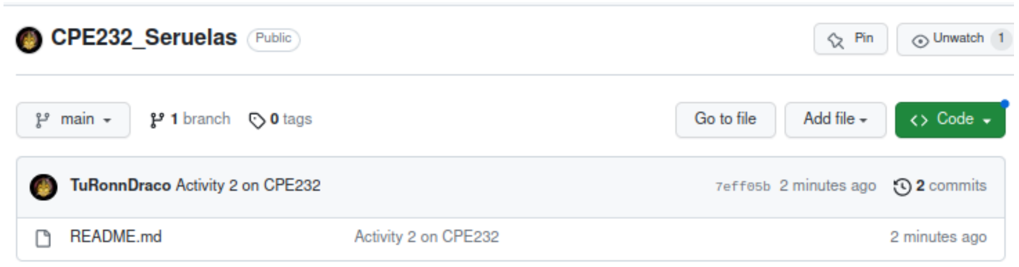
l. Use the command *git push <remote><branch>* to upload the local repository content to GitHub repository. Pushing means to transfer commits from the local repository to the remote repository. As an example, you may issue *git push origin main.*



```
seruelas@Workstation: ~/CPE232_Seruelas
seruelas@Workstation:~/CPE232_Seruelas$ 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), 331 bytes | 331.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:TuRonnDraco/CPE232_Seruelas.git
   96e02d9..7eff05b  main -> main
```

Figure 5l.1 - Uploading or transferring the committed files into the local repository, CPE232_Seruelas via **git push**.

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.

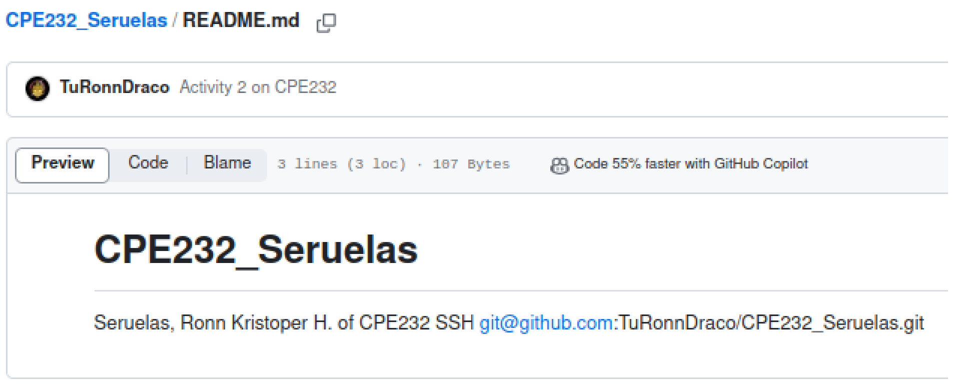


CPE232_Seruelas Public

TuRonnDraco Activity 2 on CPE232 7eff05b 2 minutes ago 2 commits

README.md Activity 2 on CPE232 2 minutes ago

Figure 5m.1 - Verifying the changes made from the README.md file, committed 2 minutes and 2 commits after.


CPE232_Seruelas / README.md

TuRonnDraco Activity 2 on CPE232

Preview
Code
Blame
3 lines (3 loc) · 107 Bytes
Code 55% faster with GitHub Copilot

CPE232_Seruelas

Seruelas, Ronn Kristoper H. of CPE232 SSH git@github.com:TuRonnDraco/CPE232_Seruelas.git

Figure 5k.2 - Verifying the message provided and given from the workstation terminal, successfully committed and transferred to the github repository.

Reflections:

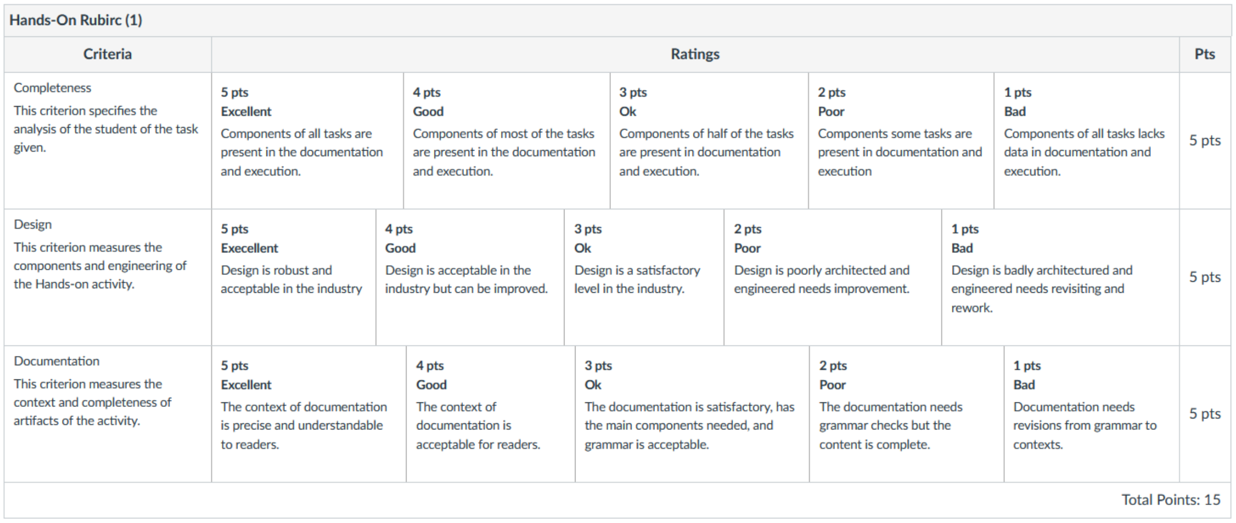
Answer the following:

1. What sort of things have we so far done to the remote servers using ansible commands?
    - We are able to explore and to learn the basics of remote hosting or remote accessing of other stations. With learning the basics of remote hosting, we are able to learn on how we are able to connect to github with the ease of using our own terminal. We are also able to learn on how we can modify the local repository on github by using various commands that allows us to create, modify and to apply changes to the repository.

2. How important is the inventory file?
    - The inventory file allows the user to have a list or an infrastructure that allows the main user or the administrator to manage the files and the hosts in a flexible manner.

Conclusions/Learnings:

In this activity, we are able to learn on how keys are a fundamental or an essential in remote access or remote hosting, as it allows the user to have a more encrypted and secured authentication. We were able to learn on how we can generate our own keys and how we can use the keys for easier authentication from accessing other terminals or stations or servers. We were able to learn on how we can access a local repository from our github accounts via various commands. We were able to learn the importance of SSH and how it allows the user or the administrators to manage inventories.



Hands-On Rubirc (1)

Criteria	Ratings					Pts
Completeness This criterion specifies the analysis of the student of the task given.	**5 pts** **Excellent** Components of all tasks are present in the documentation and execution.	**4 pts** **Good** Components of most of the tasks are present in the documentation and execution.	**3 pts** **Ok** Components of half of the tasks are present in documentation and execution.	**2 pts** **Poor** Components some tasks are present in documentation and execution	**1 pts** **Bad** Components of all tasks lacks data in documentation and execution.	5 pts
Design This criterion measures the components and engineering of the Hands-on activity.	**5 pts** **Exececllent** Design is robust and acceptable in the industry	**4 pts** **Good** Design is acceptable in the industry but can be improved.	**3 pts** **Ok** Design is a satisfactory level in the industry.	**2 pts** **Poor** Design is poorly architected and engineered needs improvement.	**1 pts** **Bad** Design is badly architectured and engineered needs revisiting and rework.	5 pts
Documentation This criterion measures the context and completeness of artifacts of the activity.	**5 pts** **Excellent** The context of documentation is precise and understandable to readers.	**4 pts** **Good** The context of documentation is acceptable for readers.	**3 pts** **Ok** The documentation is satisfactory, has the main components needed, and grammar is acceptable.	**2 pts** **Poor** The documentation needs grammar checks but the content is complete.	**1 pts** **Bad** Documentation needs revisions from grammar to contexts.	5 pts
						Total Points: 15

"I affirm that I have not received or given any unauthorized help on this activity and that all work is my own."