

Name: Rio Marie G. Suzuki	Date Performed: 08/24/2023
Course/Section: CPE232 S6	Date Submitted: 08/24/2023
Instructor: Dr. Jonathan Taylor	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). <i>Provide screenshots for each task.</i> 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 <i>ssh-keygen</i> 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`.

```
rio@Workstation:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/rio/.ssh/id_rsa): id_rsa
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in id_rsa.
Your public key has been saved in id_rsa.pub.
The key fingerprint is:
SHA256:QJlVByhonbHaZKSXnX7j+CcDkwC133LreZ3GN0Ry6TM rio@Workstation
The key's randomart image is:
+---[RSA 2048]---+
|      + = *.00..      |
|      ++* = ..  .     |
|      ...0.0         . |
|      *.+ . . . +     |
|      . ..S.=      =   |
|      +* o   E       |
|      .oo  o..o      |
|      oo.o =..       |
|      += . ..        |
+-----[SHA256]-----+
rio@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.

```

overwrite (y/n): n
rio@Workstation:~$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/rio/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/rio/.ssh/id_rsa.
Your public key has been saved in /home/rio/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:0rubpX0ldY77FCC85trR+iDvmaKsXQ1/rt5fPukXpM0 rio@Workstation
The key's randomart image is:
+---[RSA 4096]---+
|
|      .
|      o .
|      . o o..
|      . S. o .==
|      . .* o.oEo
|      .o.* = .=
|      o .B= @ .o
|      ..+*o=@o+o+=
+-----[SHA256]-----+
rio@Workstation:~$

```

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/rio/.ssh/id_rsa.
Your public key has been saved in /home/rio/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:0rubpX0ldY77FCC85trR+iDvmaKsXQ1/rt5fPukXpM0 rio@Workstation

```

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.

```
+----[SHA256]-----+
rio@Workstation:~$ ls -la .ssh
total 20
drwx----- 2 rio rio 4096 Aug 24 17:28 .
drwxr-xr-x 16 rio rio 4096 Aug 24 17:25 ..
-rw----- 1 rio rio 3243 Aug 24 17:28 id_rsa
-rw-r--r-- 1 rio rio 741 Aug 24 17:28 id_rsa.pub
-rw-r--r-- 1 rio rio 222 Aug 17 18:08 known_hosts
rio@Workstation:~$
```

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.

```
rio@Workstation:~$ ssh-copy-id
Usage: /usr/bin/ssh-copy-id [-h|-?|-f|-n] [-i [identity_file]] [-p port] [[-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
    -h|-?: print this help
```

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

```
rio@Workstation:~$ ssh-copy-id -i ~/.ssh/id_rsa rio@Workstation
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/rio/.ssh/i
d_rsa.pub"
The authenticity of host 'workstation (127.0.0.1)' can't be established.
ECDSA key fingerprint is SHA256:sydMGH9MZuzCWpeWhvswqt502rdC2e43cgKPJYDg6kA.
Are you sure you want to continue connecting (yes/no)? y
Please type 'yes' or 'no': 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
rio@workstation's password:
```

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.

```
ted now it is to install the new ke
rio@workstation's password:
```

```
ted now it is to install the new keys  
rio@workstation's password:
```

```
Number of key(s) added: 1
```

```
Now try logging into the machine, with:  "ssh 'rio@Workstation'"  
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?

```
rio@Server1:~$ ssh rio@server1  
rio@server1's password:  
Welcome to Ubuntu 18.04.6 LTS (GNU/Linux 5.4.0-150-generic x86_64)  
  
* Documentation:  https://help.ubuntu.com  
* Management:    https://landscape.canonical.com  
* Support:        https://ubuntu.com/advantage  
  
Expanded Security Maintenance for Infrastructure is not enabled.  
  
0 updates can be applied immediately.  
  
7 additional security updates can be applied with ESM Infra.  
Learn more about enabling ESM Infra service for Ubuntu 18.04 at  
https://ubuntu.com/18-04  
  
Your Hardware Enablement Stack (HWE) is supported until April 2023.  
Last login: Thu Aug 17 18:22:51 2023 from 127.0.0.1  
rio@Server1:~$
```

```
rio@Server2:~$ ssh rio@server2
rio@server2's password:
Welcome to Ubuntu 18.04.6 LTS (GNU/Linux 5.4.0-150-generic x86_64)

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

Expanded Security Maintenance for Infrastructure is not enabled.

0 updates can be applied immediately.

85 additional security updates can be applied with ESM Infra.
Learn more about enabling ESM Infra service for Ubuntu 18.04 at
https://ubuntu.com/18-04

Your Hardware Enablement Stack (HWE) is supported until April 2023.
Last login: Thu Aug 17 18:24:16 2023 from 192.168.56.102
rio@Server2:~$
```

Reflections:

Answer the following:

1. How will you describe the ssh-program? What does it do?
 - The SSH program allows secure communication between computers over unsecured networks. It enables remote system access, file transfers, and more. SSH establishes a secure channel, ensuring data privacy and integrity during tasks like remote command execution and system management.
2. How do you know that you already installed the public key to the remote servers?
 - If the key was properly installed, the local computer will switch to the server without requesting the server's password, allowing you to determine whether the public key was installed on the distant servers by first attempting to log in with the command "ssh user@hostname".

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*

```
rio@Suzuki:~$ which git
rio@Suzuki:~$ sudo apt install git
[sudo] password for rio:
rio is not in the sudoers file. This incident will be reported.
rio@Suzuki:~$ sudo su-
[sudo] password for rio:
rio is not in the sudoers file. This incident will be reported.
rio@Suzuki:~$ su
Password:
root@Suzuki:/home/rio# which git
root@Suzuki:/home/rio# sudo apt install git
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  git-man liberror-perl
Suggested packages:
  git-daemon-run | git-daemon-sysvinit git-doc git-email git-gui gitk gitweb
  git-cvs git-mediawiki git-svn
```

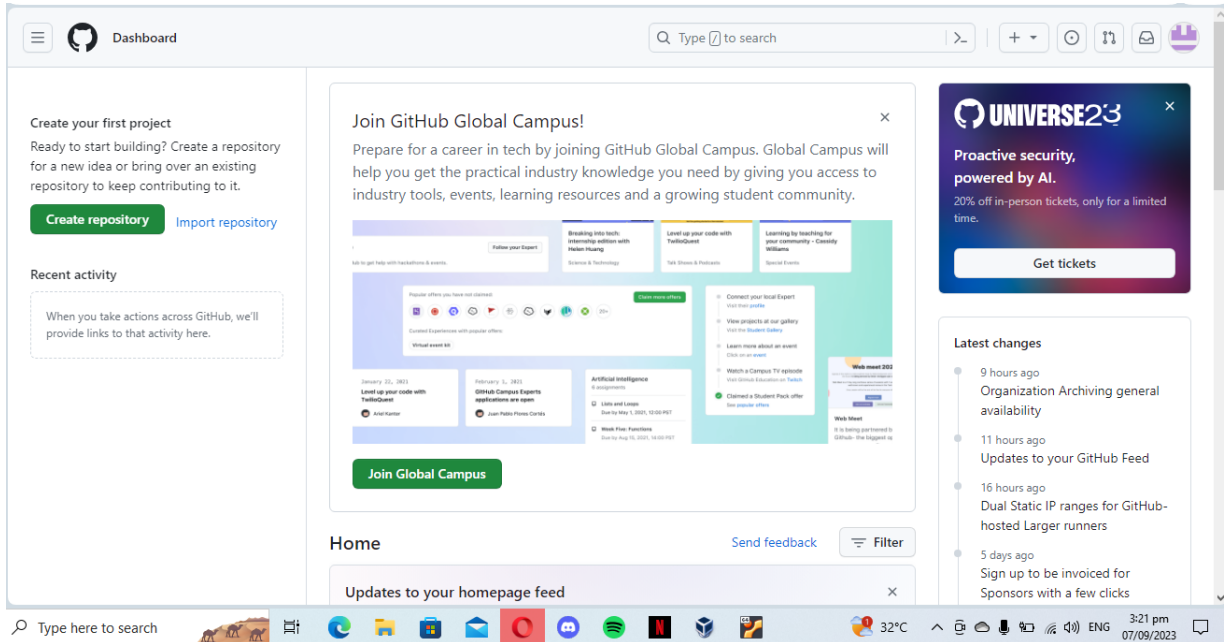
2. After the installation, issue the command *which git* again. The directory of git is usually installed in this location: *user/bin/git*.

```
exit
rio@Suzuki:~$ which git
/usr/bin/git
rio@Suzuki:~$ user/bin/git
bash: user/bin/git: No such file or directory
rio@Suzuki:~$
```

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

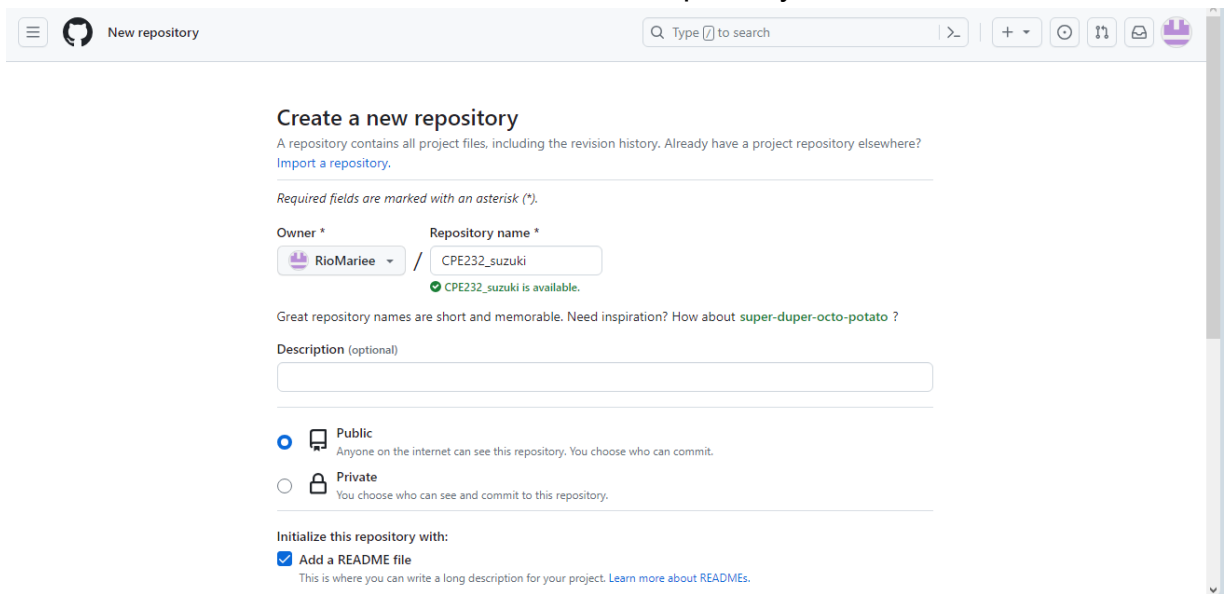

```
rio is not in the sudoers file.
rio@Suzuki:~$ git --version
git version 2.34.1
rio@Suzuki:~$
```

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

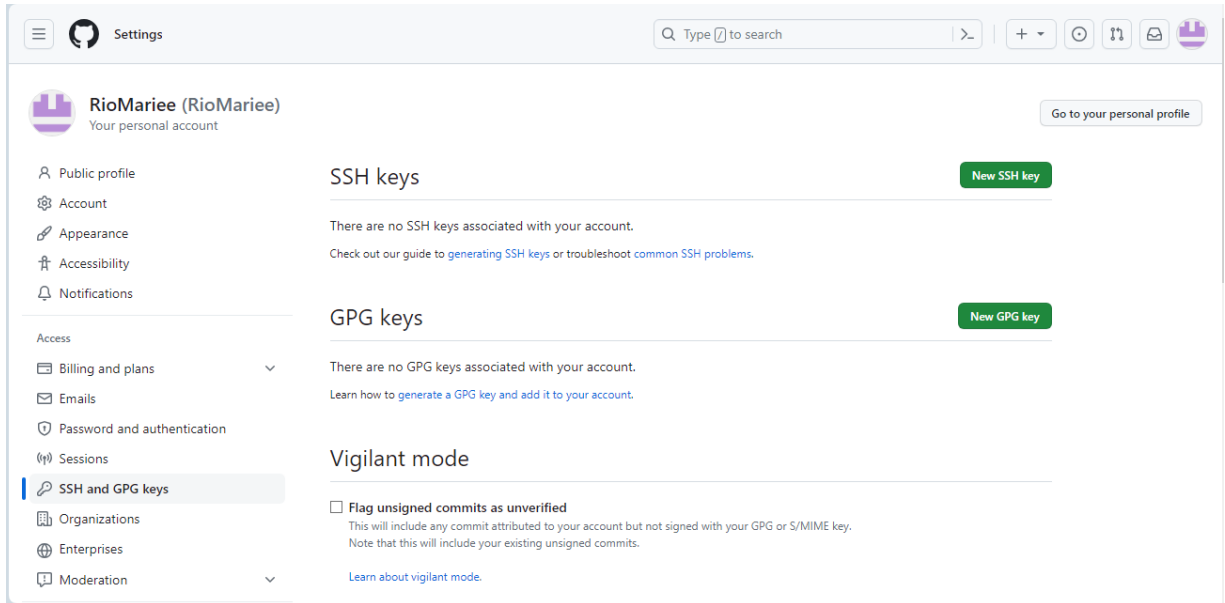


5. Sign up in case you don't have an account yet. Otherwise, login to your GitHub account.

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

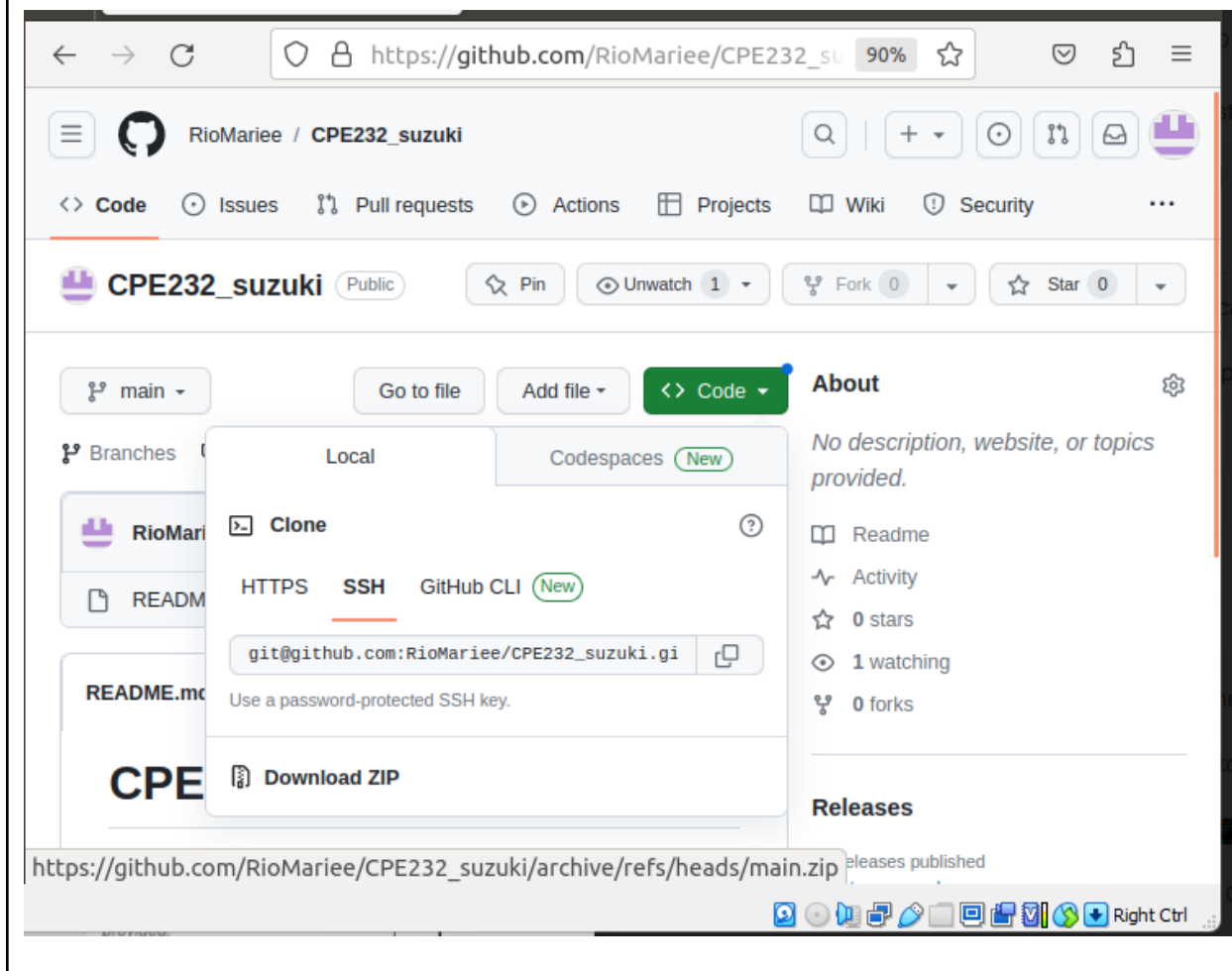
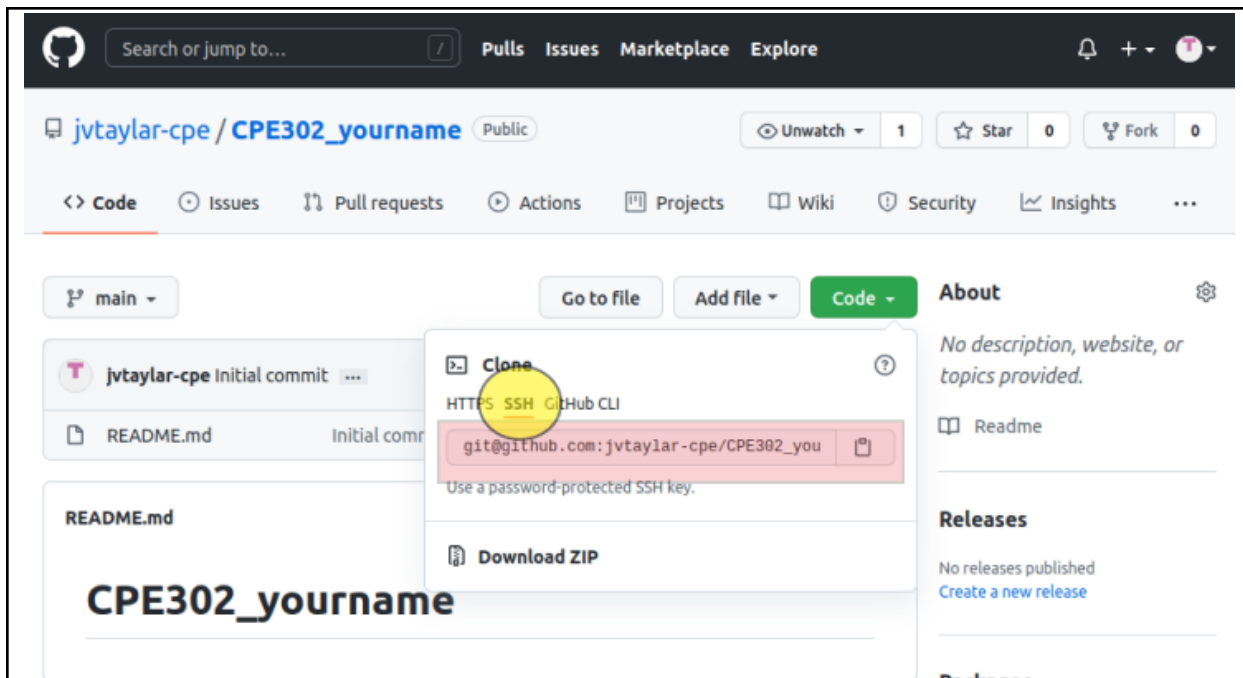


CPE232
SHA256: jGDM0tD0zcqcxgv0Mr1E8GxnstYWcn1vkrRUsLq5Tz0
Added on Sep 7, 2023
Never used — Read/write

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:jvtaylor-cpe/CPE232_yourname.git`. When prompted to continue connecting, type yes and press enter.

```
rio@Workstation:~$ git clone git@github.com:RioMariee/CPE232_suzuki.git
Cloning into 'CPE232_suzuki'...
The authenticity of host 'github.com (20.87.225.212)' can't be established.
ECDSA key fingerprint is SHA256:p2QAMXNIC1TJYWeIOttrVc98/R1BUFWu3/LiyKgUfQM.
Are you sure you want to continue connecting (yes/no)?
```

```
Processing triggers for man-db (2.8.3-2ubuntu0.1) ...
rio@Workstation:~$ git clone git@github.com:RioMariee/CPE232_suzuki.git
Cloning into 'CPE232_suzuki'...
The authenticity of host 'github.com (20.87.225.212)' can't be established.
ECDSA key fingerprint is SHA256:p2QAMXNIC1TJYWeIOttrVc98/R1BUFWu3/LiyKgUfQM.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'github.com,20.87.225.212' (ECDSA) 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.
rio@Workstation:~$
```

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

```
Receiving objects: 100% (3/3), done.
rio@Workstation:~$ ls
CPE232_suzuki  Downloads      'id_rsa\'      Music          Templates
Desktop        examples.desktop id_rsa.pub     Pictures       Videos
Documents      id_rsa         'id_rsa\'.pub  Public         Workstation
rio@Workstation:~$
```

```
bash: cd: /CPE232_suzuki: No such file or directory
rio@Workstation:~$ cd CPE232_suzuki
rio@Workstation:~/CPE232_suzuki$ ls
README.md
rio@Workstation:~/CPE232_suzuki$
```

- g. Use the following commands to personalize your git.

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

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

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

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

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

```
rio@Workstation:~$ cat ~/.gitconfig
[user]
    name = rio
    email = qrmgsuzuki@tip.edu.ph
rio@Workstation:~$
```

- 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.

```
rio@Workstation:~$ ls
CPE232_suzuki  Downloads      'id_rsa\'      Music          Templates
Desktop        examples.desktop id_rsa.pub     Pictures       Videos
Documents      id_rsa         'id_rsa\.pub' Public         Workstation

rio@Workstation:~$ cd CPE232_suzuki
rio@Workstation:~/CPE232_suzuki$ ls
README.md
rio@Workstation:~/CPE232_suzuki$ nano README.md
rio@Workstation:~/CPE232_suzuki$
```

File Edit View Search Terminal Help

GNU nano 2.9.3

README.md

```
# CPE232_suzuki
Activity 2 - SSH Key-Based Authentication and GIT Setup
```

- 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?

```

rio@Workstation:~/CPE232_suzuki$ 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 checkout -- <file>..." to discard changes in working directory)

        modified:   README.md

no changes added to commit (use "git add" and/or "git commit -a")
rio@Workstation:~/CPE232_suzuki$

```

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

```

rio@Workstation:~/CPE232_suzuki$ git add README.md
rio@Workstation:~/CPE232_suzuki$

```

- 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.

```

rio@Workstation:~/CPE232_suzuki$ git commit -m "GIT Setup"
[main 967325f] GIT Setup
 1 file changed, 2 insertions(+), 1 deletion(-)
rio@Workstation:~/CPE232_suzuki$

```

- 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*.

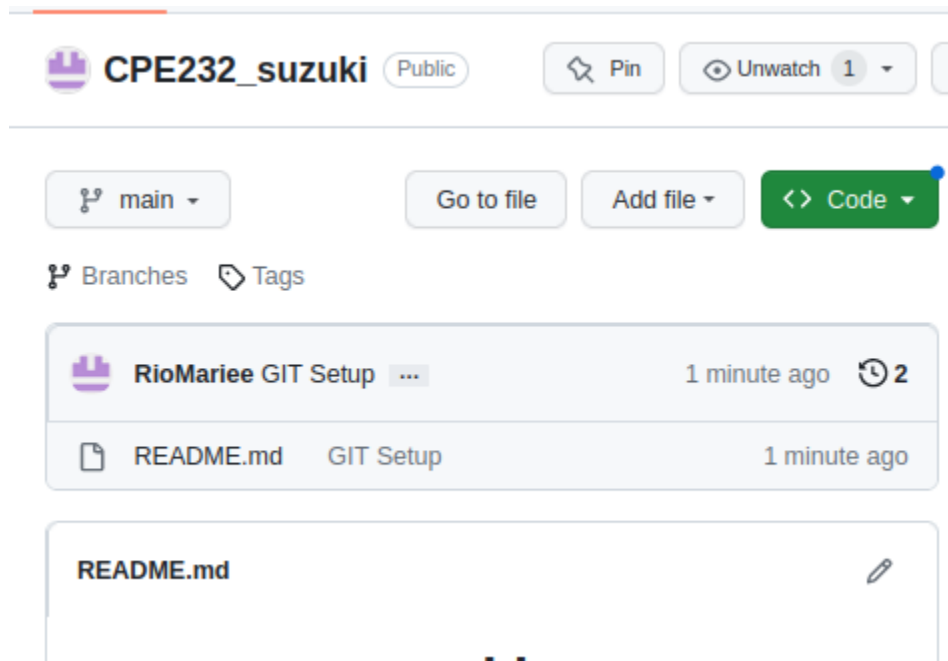
```

rio@Workstation:~/CPE232_suzuki$ git push origin main
Counting objects: 3, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 313 bytes | 313.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To github.com:RioMarieee/CPE232_suzuki.git
 cc34c29..967325f  main -> main
rio@Workstation:~/CPE232_suzuki$

```

- 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.



Reflections:

Answer the following:

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

Conclusions/Learnings:

I learned a lot about how to create and utilise ssh keys and how they work from this assignment. I am familiar with the fundamentals of ssh keys and how to utilise them. Keys may be used as a password to get access to the server using only ssh commands. I also learnt how to set up a github account, link it to the local computer, and edit local files that affect the github repository.