Name: Agpaoa, Ma.Diane J.	**Date Performed: 08/23/2022**
Course/Section: CPE232-CPE31S22	**Date Submitted: 08/23/2022**
Instructor: Dr. Jonathan Taylar	**Semester and SY: 1st Sem-2022-2023**

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.

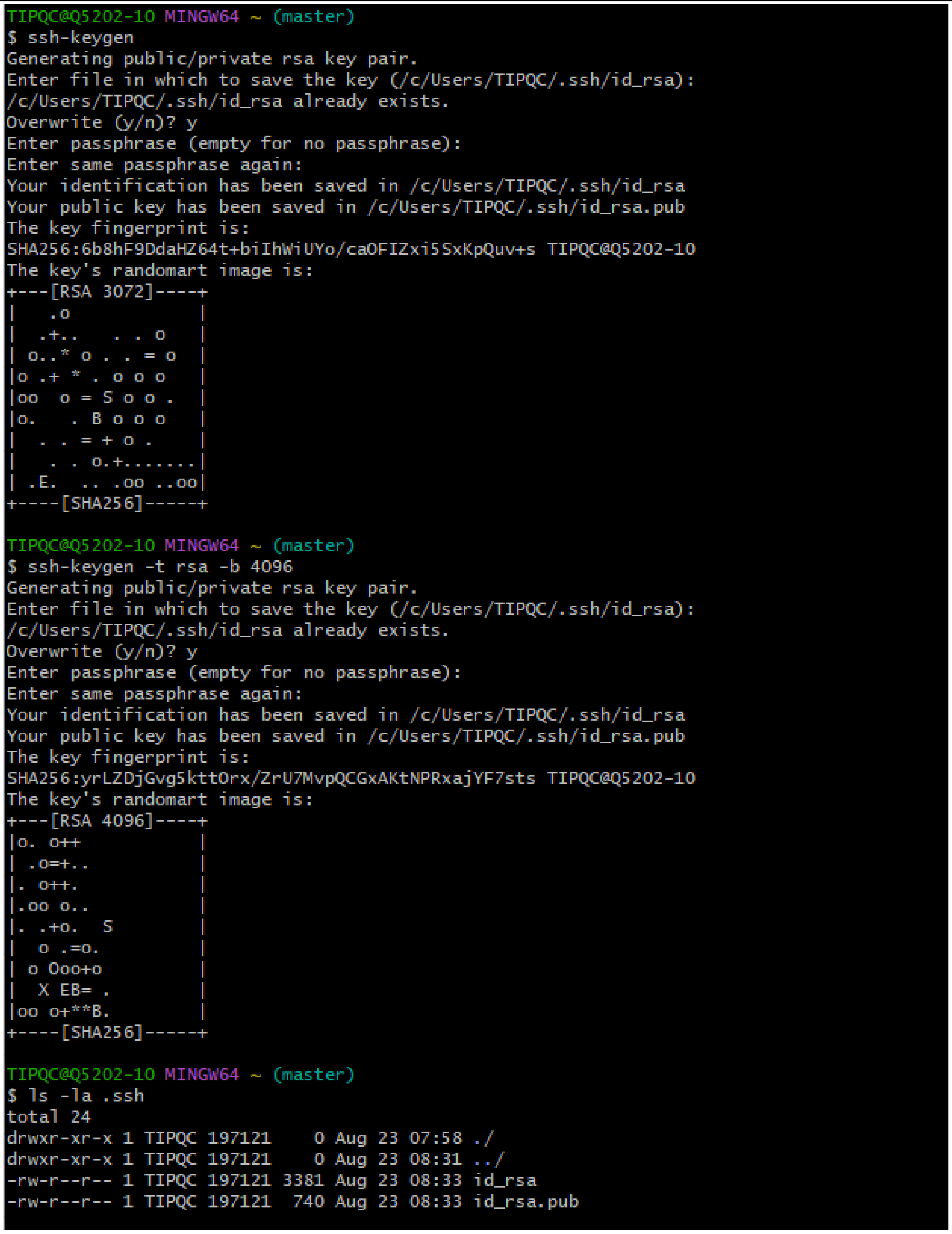
However, SSH keys are authentication credentials just like passwords. Thus, they must be managed somewhat analogously to usernames and passwords. They should have a proper termination process so that keys are removed when no longer needed.

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.
3. When asked for a passphrase, just press enter. The passphrase is used for encrypting the key, so that it cannot be used even if someone obtains the private key file. The passphrase should be cryptographically strong.
4. Verify that you have created the key by issuing the command *ls -la .ssh.* The command should show the .ssh directory containing a pair of keys. For example, id_rsa.pub and id_rsa.



```
TIPQC@Q5202-10 MINGW64 ~ (master)
$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/TIPQC/.ssh/id_rsa):
/c/Users/TIPQC/.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 /c/Users/TIPQC/.ssh/id_rsa
Your public key has been saved in /c/Users/TIPQC/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:6b8hF9DdaHZ64t+biIhWiUYo/caOFIZxi5SxKpQuv+s TIPQC@Q5202-10
The key's randomart image is:
+---[RSA 3072]----+
|   .o            |
|  .+..   . . o   |
| o..* o . . = o  |
|o .+ * . o o o   |
|oo  o = S o o .  |
|o.   . B o o o   |
|  . . = + o .    |
|   . . o.+.......|
| .E.  .. .oo ..oo|
+----[SHA256]-----+

TIPQC@Q5202-10 MINGW64 ~ (master)
$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/TIPQC/.ssh/id_rsa):
/c/Users/TIPQC/.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 /c/Users/TIPQC/.ssh/id_rsa
Your public key has been saved in /c/Users/TIPQC/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:yrLZDjGvg5kttOrx/ZrU7MvpQCGxAKtNPRxajYF7sts TIPQC@Q5202-10
The key's randomart image is:
+---[RSA 4096]----+
|o. o++           |
| .o=+..          |
|. o++.           |
|.oo o..          |
|. .+o.   S       |
|  o .=o.         |
| o Ooo+o         |
|  X EB= .        |
|oo o+**B.        |
+----[SHA256]-----+

TIPQC@Q5202-10 MINGW64 ~ (master)
$ ls -la .ssh
total 24
drwxr-xr-x 1 TIPQC 197121    0 Aug 23 07:58 ./
drwxr-xr-x 1 TIPQC 197121    0 Aug 23 08:31 ../
-rw-r--r-- 1 TIPQC 197121 3381 Aug 23 08:33 id_rsa
-rw-r--r-- 1 TIPQC 197121  740 Aug 23 08:33 id_rsa.pub
```

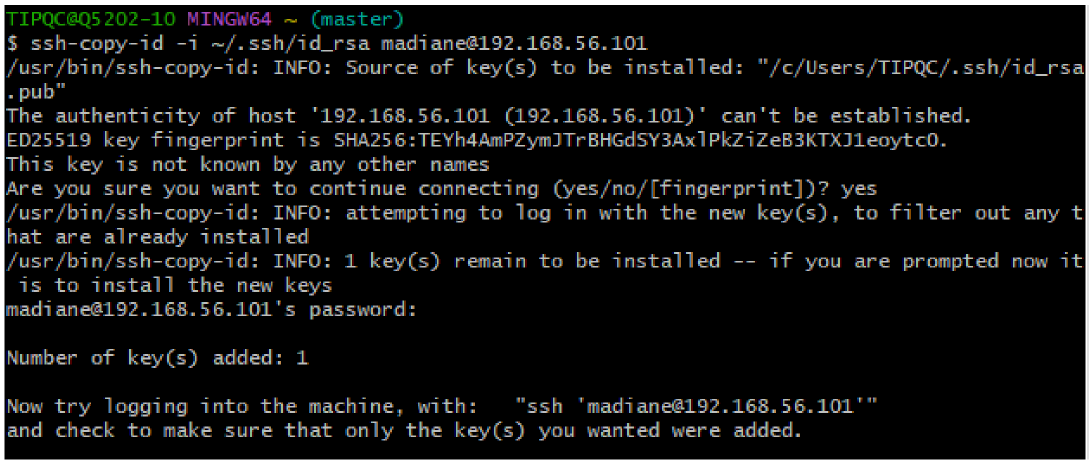
Figure 1. Created an SSH Key Pair for User Authentication

I created an SSH Key Pair for User authentication by following the instructions written in Task 1.

Task 2: Copying the Public Key to the remote servers

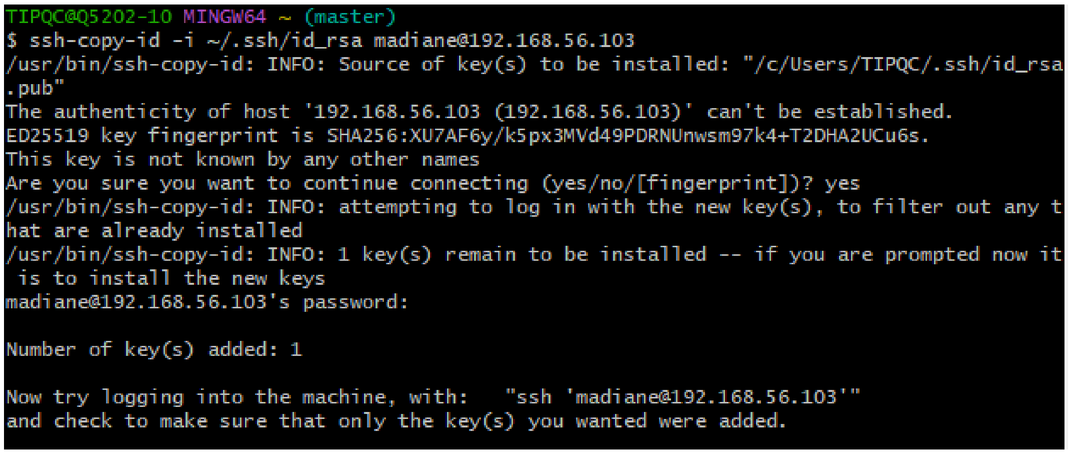
1. To use public key authentication, the public key must be copied to a server and installed in an *authorized_keys* file. This can be conveniently done using the *ssh-copy-id* tool.

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.



```
TIPQC@Q5202-10 MINGW64 ~ (master)
$ ssh-copy-id -i ~/.ssh/id_rsa madiane@192.168.56.101
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/c/Users/TIPQC/.ssh/id_rsa
.pub"
The authenticity of host '192.168.56.101 (192.168.56.101)' can't be established.
ED25519 key fingerprint is SHA256:TEYh4AmPZymJTrBHGdSY3AxlPkZiZeB3KTXJ1eoytc0.
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 t
hat are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it
 is to install the new keys
madiane@192.168.56.101's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'madiane@192.168.56.101'"
and check to make sure that only the key(s) you wanted were added.
```

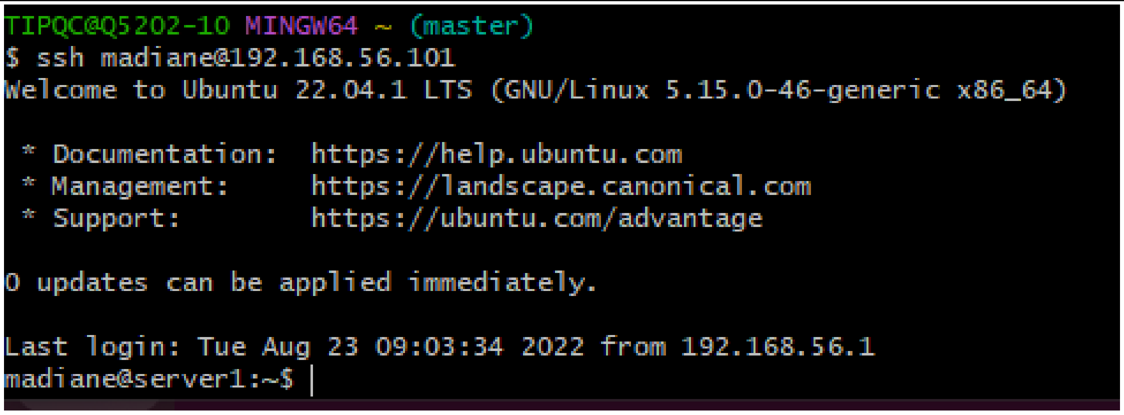
Figure 2.1 Copying the Public Key to the Server 1



```
TIPQC@Q5202-10 MINGW64 ~ (master)
$ ssh-copy-id -i ~/.ssh/id_rsa madiane@192.168.56.103
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/c/Users/TIPQC/.ssh/id_rsa
.pub"
The authenticity of host '192.168.56.103 (192.168.56.103)' can't be established.
ED25519 key fingerprint is SHA256:XU7AF6y/k5px3MVd49PDRNUnwsm97k4+T2DHA2UCu6s.
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 t
hat are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it
 is to install the new keys
madiane@192.168.56.103's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'madiane@192.168.56.103'"
and check to make sure that only the key(s) you wanted were added.
```

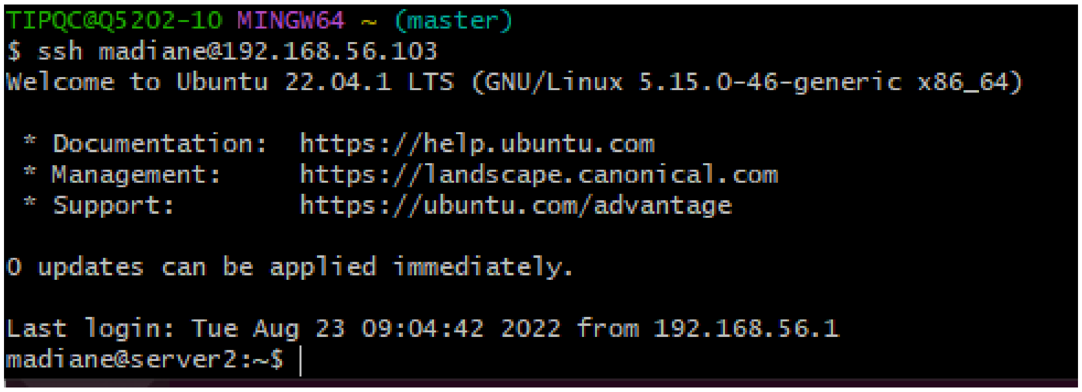
2.2 Copying the Public Key to the Server 2

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?



```
TIPQC@Q5202-10 MINGW64 ~ (master)
$ ssh madiane@192.168.56.101
Welcome to Ubuntu 22.04.1 LTS (GNU/Linux 5.15.0-46-generic x86_64)

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

0 updates can be applied immediately.

Last login: Tue Aug 23 09:03:34 2022 from 192.168.56.1
madiane@server1:~$ |
```

Figure 2.2 SSH with Server 1



```
TIPQC@Q5202-10 MINGW64 ~ (master)
$ ssh madiane@192.168.56.103
Welcome to Ubuntu 22.04.1 LTS (GNU/Linux 5.15.0-46-generic x86_64)

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

0 updates can be applied immediately.

Last login: Tue Aug 23 09:04:42 2022 from 192.168.56.1
madiane@server2:~$ |
```

Figure 2.3 SSH with Server 2

I noticed that the connection with Server 1 and Server 2 did not ask for a password. It did not ask for password because the authentication it used to verify the user was the generated SSH Key Pair.

Reflections:

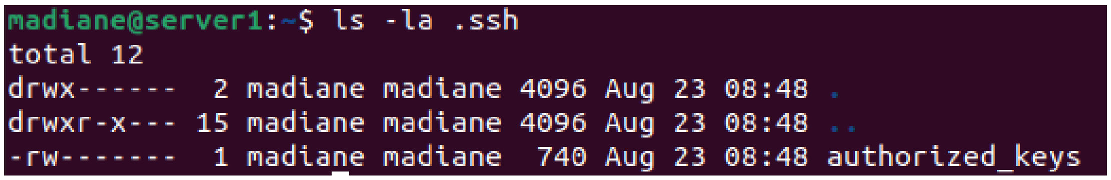
Answer the following:

1. How will you describe the ssh-program? What does it do?

   I will describe the ssh-program as a secure way to configure and connect through Linux remote servers. Based on our professor, an SSH-program is like a key and a lock; this represents the SSH key pair it generates for authentication (public and private keys). The SSH-program also uses password for client authentication however this is less secure than the SSH-keys. In addition, you can execute commands, make changes and configure services safely using the SSH-program.

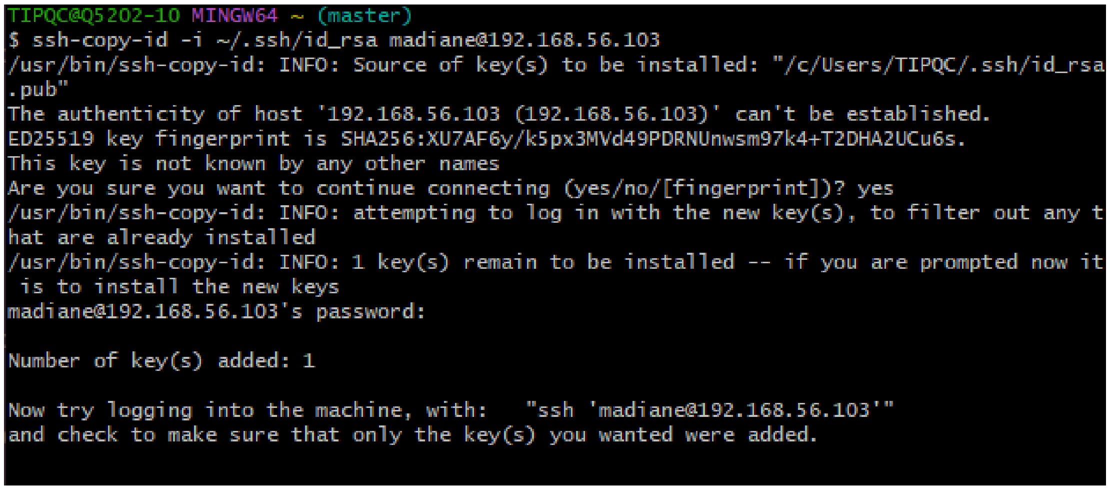
2. How do you know that you already installed the public key to the remote servers?

You can know that you already installed the public key to the remote servers, by issuing the command ls -la .ssh on the remote server. You will notice the autorized_keys file that was installed in the server, this file contains the public key that was copied using the ssh-copy-id tool.



```
madiane@server1:~$ ls -la .ssh
total 12
drwx------  2 madiane madiane 4096 Aug 23 08:48 .
drwxr-x--- 15 madiane madiane 4096 Aug 23 08:48 ..
-rw-------  1 madiane madiane  740 Aug 23 08:48 authorized_keys
```

Figure 2.3 Verifying the installation of public key on remote servers



```
TIPQC@Q5202-10 MINGW64 ~ (master)
$ ssh-copy-id -i ~/.ssh/id_rsa madiane@192.168.56.103
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/c/Users/TIPQC/.ssh/id_rsa
.pub"
The authenticity of host '192.168.56.103 (192.168.56.103)' can't be established.
ED25519 key fingerprint is SHA256:XU7AF6y/k5px3MVd49PDRNUnwsm97k4+T2DHA2UCu6s.
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 t
hat are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it
 is to install the new keys
madiane@192.168.56.103's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'madiane@192.168.56.103'"
and check to make sure that only the key(s) you wanted were added.
```

Figure 2.4 Verifying the installation of public key on remote servers

Another simple way to know that you already installed the public key to the remote servers is the confirmation it will display after you enter the password of the user@host in the ssh-copy-id tool. This confirmation also means that you can SSH with the Server 1 and Server 2, if you were able to connect without entering password, it verifies that you already installed the public key to the remote server.

Part 2: Discussion

Provide screenshots for each task.

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

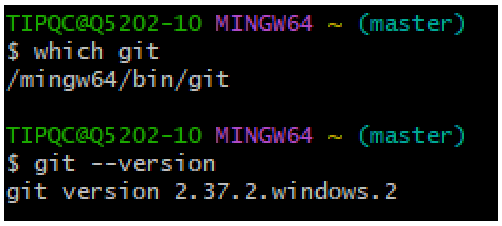
Set up Git

At the heart of GitHub is an open-source version control system (VCS) called Git. Git is responsible for everything GitHub-related that happens locally on your computer. To use Git on the command line, you'll need to download, install, and configure Git on your computer. You can also install GitHub CLI to use GitHub from the command line. If you don't need to work with files locally, GitHub lets you complete many Git-related actions directly in the browser, including:

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

Task 3: Set up the Git Repository

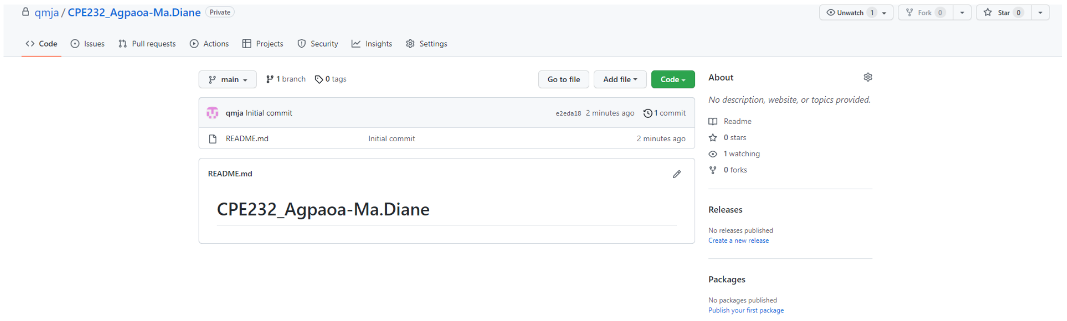
1. On the local machine, verify the version of your git using the command *which git.* If a directory of git is displayed, then you don't need to install git. Otherwise, to install git, use the following command: *sudo apt install git*
2. After the installation, issue the command *which git* again. The directory of git is usually installed in this location: *user/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.



```
TIPQC@Q5202-10 MINGW64 ~ (master)
$ which git
/mingw64/bin/git

TIPQC@Q5202-10 MINGW64 ~ (master)
$ git --version
git version 2.37.2.windows.2
```

Figure 3.1 Issuing the command git —version

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.



qmja / CPE232_Agpaoa-Ma.Diane Private

qmja Initial commit e2eda18 2 minutes ago 1 commit

README.md Initial commit 2 minutes ago

README.md

CPE232_Agpaoa-Ma.Diane

About

No description, website, or topics provided.

Readme
0 stars
1 watching
0 forks

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Figure 3.2 Created a new repository

I created a new repository by following the instructions and named it as CPE232_Agpaoa, Ma.Diane.

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.

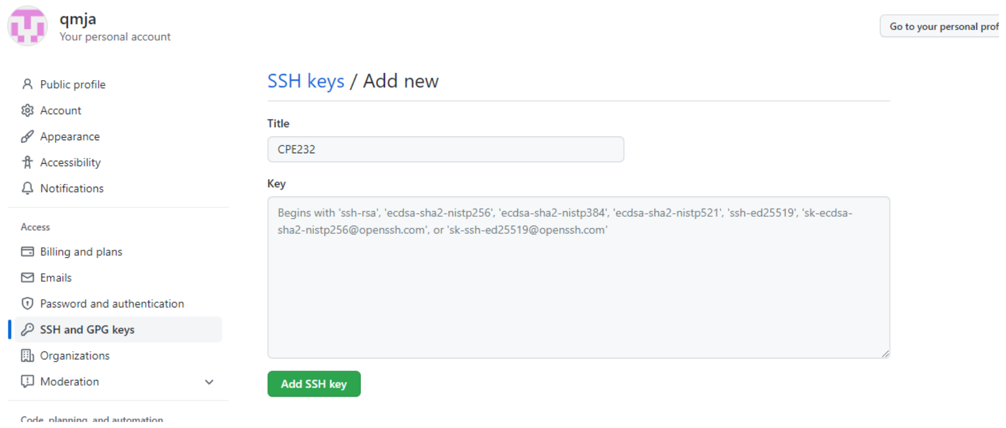
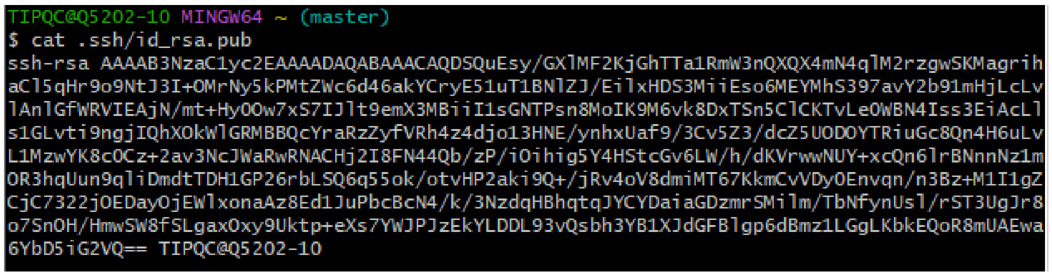

qmja
Your personal account
Go to your personal prof
Public profile
Account
Appearance
Accessibility
Notifications
Access
Billing and plans
Emails
Password and authentication
SSH and GPG keys
Organizations
Moderation
Code, planning, and automation
SSH keys / Add new
Title
CPE232
Key
Begins with 'ssh-rsa', 'ecdsa-sha2-nistp256', 'ecdsa-sha2-nistp384', 'ecdsa-sha2-nistp521', 'ssh-ed25519', 'sk-ecdsa-sha2-nistp256@openssh.com', or 'sk-ssh-ed25519@openssh.com'
Add SSH key

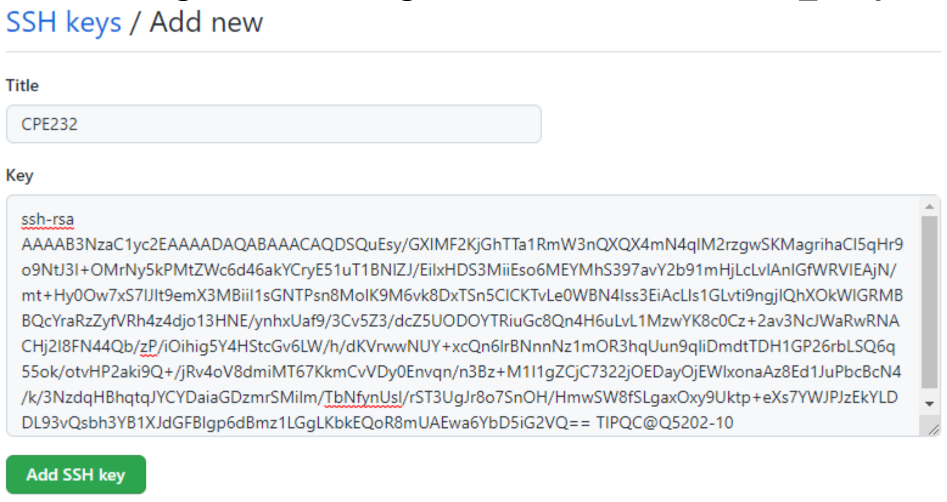
Figure 3.3 Creating a new SSH Keys

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.



```
TIPQC@Q5202-10 MINGW64 ~ (master)
$ cat .ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAACAQDSQuEsy/GXlMF2KjGhTTa1RmW3nQXQX4mN4qlM2rzgwSKMagrih
aCl5qHr9o9NtJ3I+OMrNy5kPMtZWc6d46akYCryE51uT1BNlZJ/EilxHDS3MiiEso6MEYMhS397avY2b91mHjLcLv
lAnlGfWRVIEAjN/mt+Hy0Ow7xS7IJlt9emX3MBiiI1sGNTPsn8MoIK9M6vk8DxTSn5ClCKTvLe0WBN4Iss3EiAcLl
s1GLvti9ngjIQhXOkWlGRMBBQcYraRzZyfVRh4z4djo13HNE/ynhxUaf9/3Cv5Z3/dcZ5UODOYTRiuGc8Qn4H6uLv
L1MzwYK8c0Cz+2av3NcJWaRwRNACHj2I8FN44Qb/zP/iOihig5Y4HStcGv6LW/h/dKVrwwNUY+xcQn6lrBNnnNz1m
OR3hqUun9qliDmdtTDH1GP26rbLSQ6q55ok/otvHP2aki9Q+/jRv4oV8dmiMT67KkmCvVDy0Envqn/n3Bz+M1I1gZ
CjC7322jOEDayOjEWlxonaAz8Ed1JuPbcBcN4/k/3NzdqHBhqtqJYCYDaiaGDzmrSMilm/TbNfynUsl/rST3UgJr8
o7SnOH/HmwSW8fSLgaxOxy9Uktp+eXs7YWJPJzEkYLDDL93vQsbh3YB1XJdGFBlgp6dBmz1LGgLKbkEQoR8mUAEwa
6YbD5iG2VQ== TIPQC@Q5202-10
```

Figure 3.4 Issuing the command cat .ssh/id_rsa.pub



SSH keys / Add new

Title

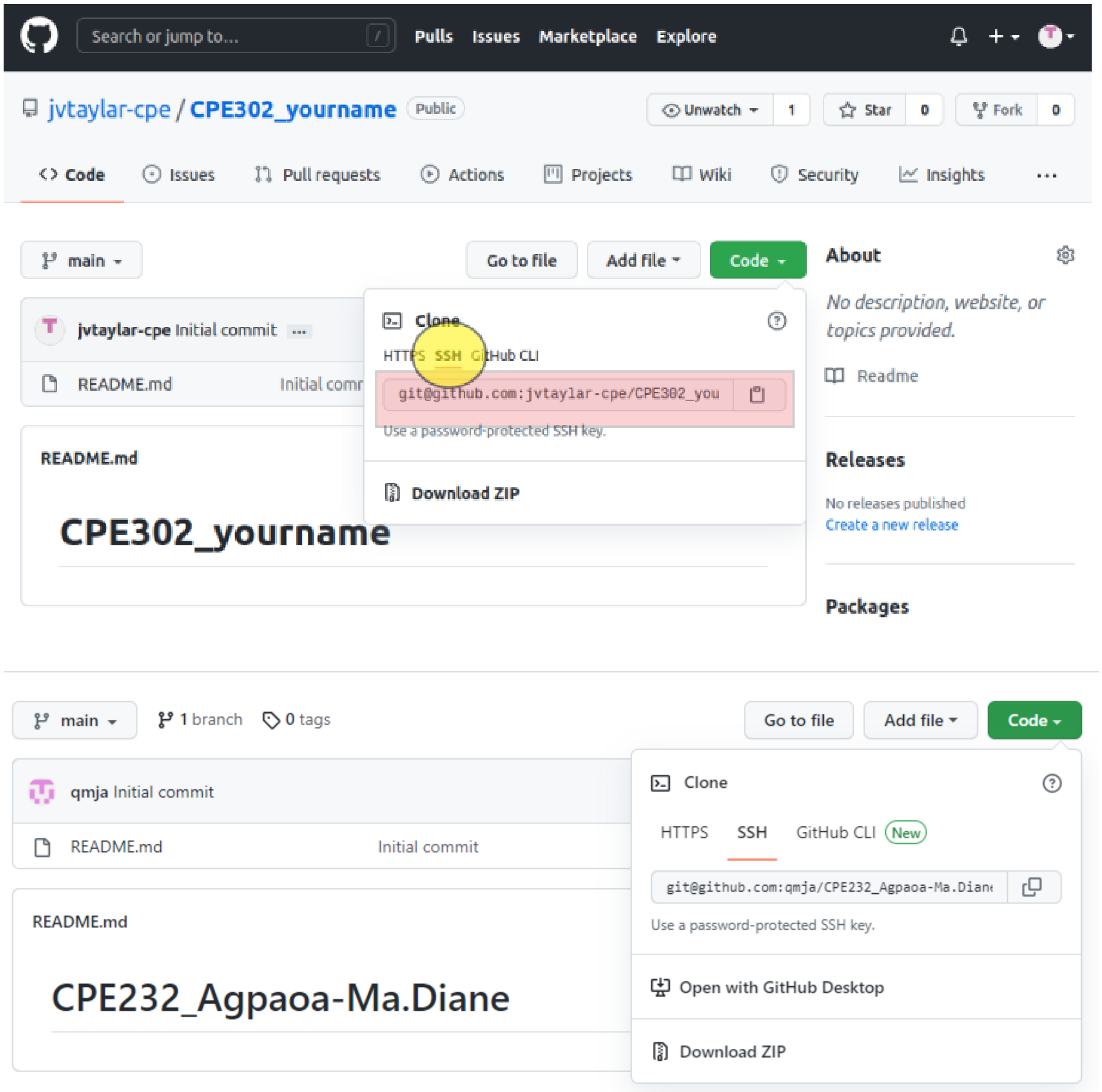
CPE232

Key

```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAACAQDSQuEsy/GXlMF2KjGhTTa1RmW3nQXQX4mN4qlM2rzgwSKMagrihaCl5qHr9
o9NtJ3I+OMrNy5kPMtZWc6d46akYCryE51uT1BNlZJ/EilxHDS3MiiEso6MEYMhS397avY2b91mHjLcLvlAnlGfWRVIEAjN/
mt+Hy0Ow7xS7IJlt9emX3MBiiI1sGNTPsn8MoIK9M6vk8DxTSn5ClCKTvLe0WBN4Iss3EiAcLls1GLvti9ngjIQhXOkWlGRMB
BQcYraRzZyfVRh4z4djo13HNE/ynhxUaf9/3Cv5Z3/dcZ5UODOYTRiuGc8Qn4H6uLvL1MzwYK8c0Cz+2av3NcJWaRwRNA
CHj2I8FN44Qb/zP/iOihig5Y4HStcGv6LW/h/dKVrwwNUY+xcQn6lrBNnnNz1mOR3hqUun9qliDmdtTDH1GP26rbLSQ6q
55ok/otvHP2aki9Q+/jRv4oV8dmiMT67KkmCvVDy0Envqn/n3Bz+M1I1gZCjC7322jOEDayOjEWlxonaAz8Ed1JuPbcBcN4
/k/3NzdqHBhqtqJYCYDaiaGDzmrSMilm/TbNfynUsl/rST3UgJr8o7SnOH/HmwSW8fSLgaxOxy9Uktp+eXs7YWJPJzEkYLD
DL93vQsbh3YB1XJdGFBlgp6dBmz1LGgLKbkEQoR8mUAEwa6YbD5iG2VQ== TIPQC@Q5202-10
```

Figure 3.5 Copying and Pasting the public key on the GitHub Key

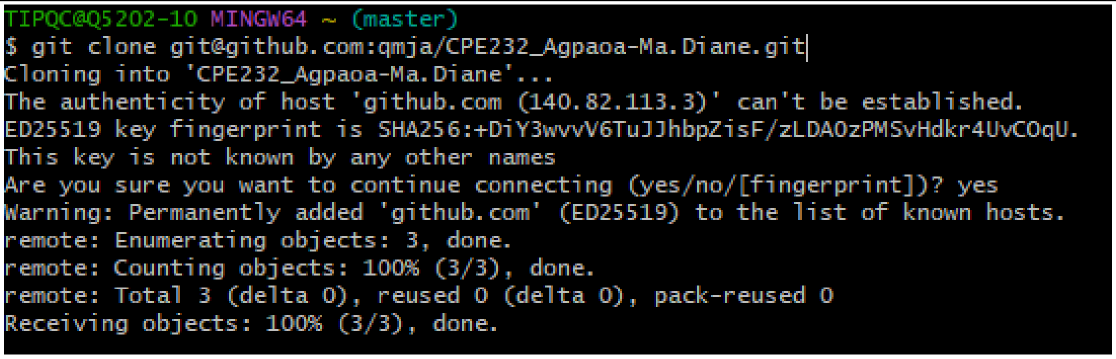
d. Clone the repository that you created. In doing this, you need to get the link from GitHub. Browse to your repository as shown below. Click on the Code drop down menu. Select SSH and copy the link.



```
git@github.com:jvtaylar-cpe/CPE302_you
git@github.com:qmja/CPE232_Agpaoa-Ma.Dian
```

Figure 3.6 Copying the link of SSH

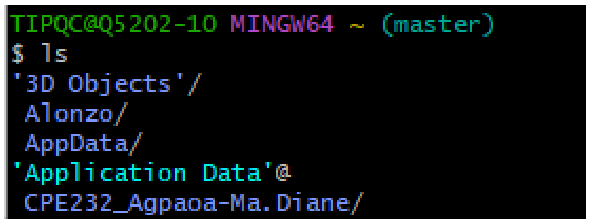
e. Issue the command git clone followed by the copied link. For example, *git clone git@github.com:jvtaylar-cpe/CPE232_yourname.git*. When prompted to continue connecting, type yes and press enter.



```
TIPQC@Q5202-10 MINGW64 ~ (master)
$ git clone git@github.com:qmja/CPE232_Agpaoa-Ma.Diane.git
Cloning into 'CPE232_Agpaoa-Ma.Diane'...
The authenticity of host 'github.com (140.82.113.3)' 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.
```

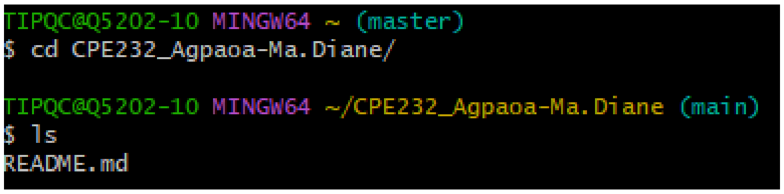
Figure 3.7 Issuing the command git clone followed by the copied link.

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.



```
TIPQC@Q5202-10 MINGW64 ~ (master)
$ ls
'3D Objects'/
 Alonzo/
 AppData/
'Application Data'@
 CPE232_Agpaoa-Ma.Diane/
```

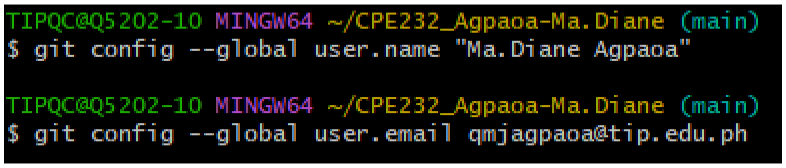
Figure 3.8 Issuing the command ls



```
TIPQC@Q5202-10 MINGW64 ~ (master)
$ cd CPE232_Agpaoa-Ma.Diane/

TIPQC@Q5202-10 MINGW64 ~/CPE232_Agpaoa-Ma.Diane (main)
$ ls
README.md
```

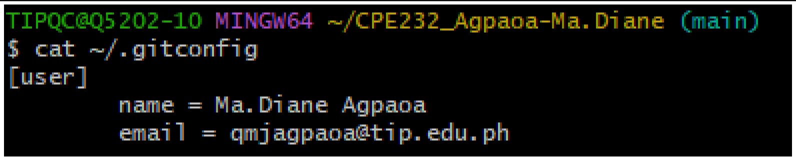
Figure 3.9 Issuing the command ls

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*



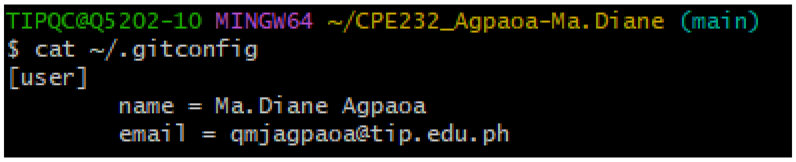
```
TIPQC@Q5202-10 MINGW64 ~/CPE232_Agpaoa-Ma.Diane (main)
$ git config --global user.name "Ma.Diane Agpaoa"

TIPQC@Q5202-10 MINGW64 ~/CPE232_Agpaoa-Ma.Diane (main)
$ git config --global user.email qmjagpaoa@tip.edu.ph
```

Figure 3.10 Issuing the command git config —global user.name



```
TIPQC@Q5202-10 MINGW64 ~/CPE232_Agpaoa-Ma.Diane (main)
$ cat ~/.gitconfig
[user]
        name = Ma.Diane Agpaoa
        email = qmjagpaoa@tip.edu.ph
```

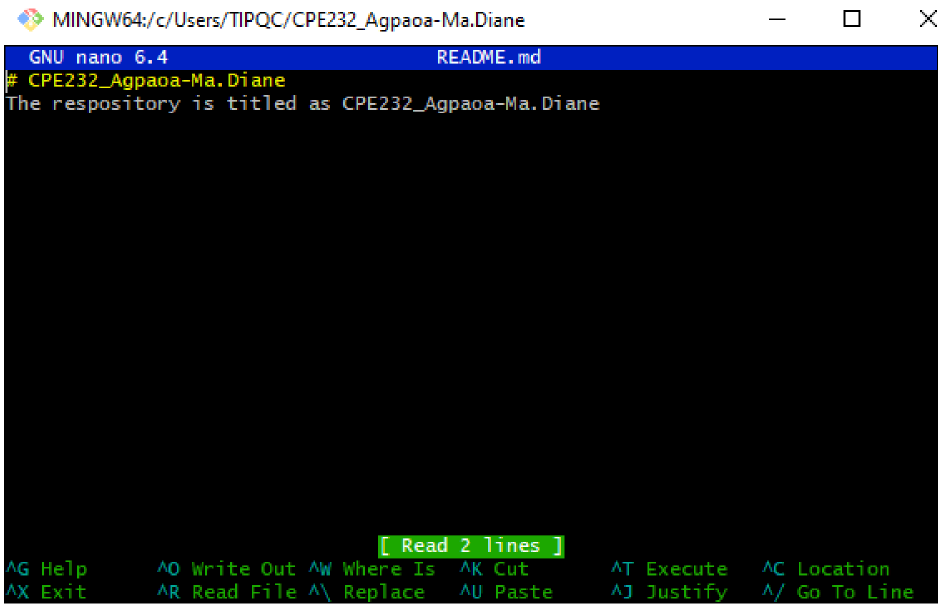
Figure 3.11 Issuing the command git config —global user.email



```
TIPQC@Q5202-10 MINGW64 ~/CPE232_Agpaoa-Ma.Diane (main)
$ cat ~/.gitconfig
[user]
        name = Ma.Diane Agpaoa
        email = qmjagpaoa@tip.edu.ph
```

Figure 3.12 Verifying that I have personalized the config file

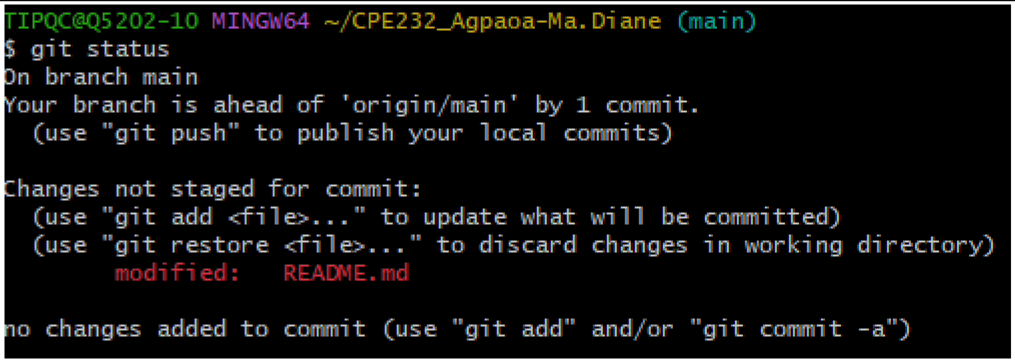
h. Edit the README.md file using nano command. Provide any information on the markdown file pertaining to the repository you created. Make sure to write out or save the file and exit.



```
MINGW64:/c/Users/TIPQC/CPE232_Agpaoa-Ma.Diane
  GNU nano 6.4                       README.md
# CPE232_Agpaoa-Ma.Diane
The respository is titled as CPE232_Agpaoa-Ma.Diane

                                  [ Read 2 lines ]
^G Help          ^O Write Out ^W Where Is  ^K Cut          ^T Execute     ^C Location
^X Exit          ^R Read File ^\ Replace   ^U Paste        ^J Justify     ^/ Go To Line
```

Figure 3.13 Providing any information on the markdown file pertaining to the repository I created.

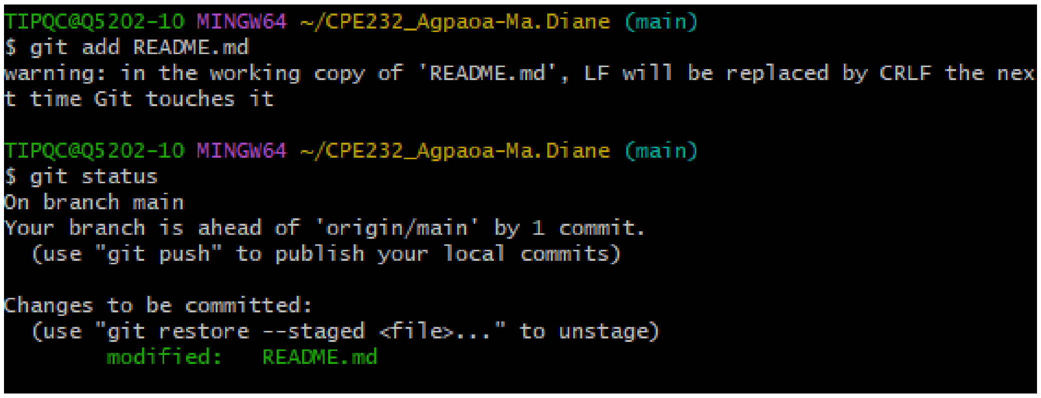
i. Use the *git status* command to display the state of the working directory and the staging area. This command shows which changes have been staged, which haven't, and which files aren't being tracked by Git. Status output does not show any information regarding the committed project history. What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



```
TIPQC@Q5202-10 MINGW64 ~/CPE232_Agpaoa-Ma.Diane (main)
$ git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

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 3.14 Displaying the state of the working directory

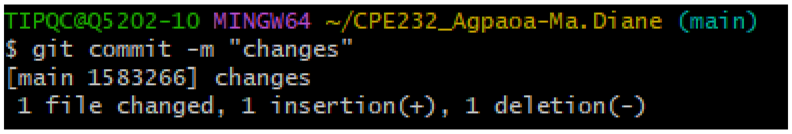
j. Use the command *git add README.md* to add the file into the staging area.



```
TIPQC@Q5202-10 MINGW64 ~/CPE232_Agpaoa-Ma.Diane (main)
$ git add README.md
warning: in the working copy of 'README.md', LF will be replaced by CRLF the nex
t time Git touches it

TIPQC@Q5202-10 MINGW64 ~/CPE232_Agpaoa-Ma.Diane (main)
$ git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

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

Figure 3.15 Adding the README.md file to the staging area

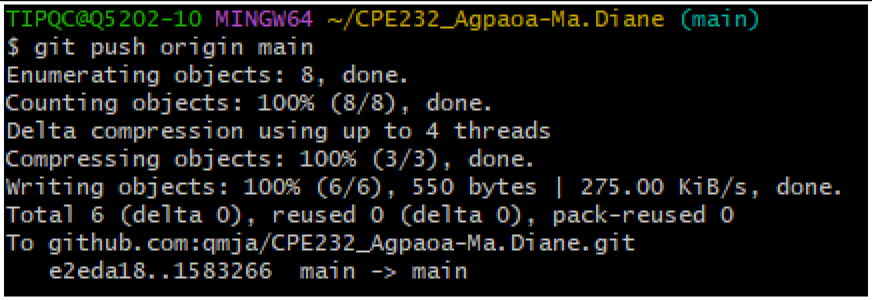
k. Use the *git commit -m "your message"* to create a snapshot of the staged changes along the timeline of the Git projects history. The use of this command is required to select the changes that will be staged for the next commit.



```
TIPQC@Q5202-10 MINGW64 ~/CPE232_Agpaoa-Ma.Diane (main)
$ git commit -m "changes"
[main 1583266] changes
 1 file changed, 1 insertion(+), 1 deletion(-)
```

Figure 3.16 Creating a snapshot of the staged changes along the timeline of the Git projects history

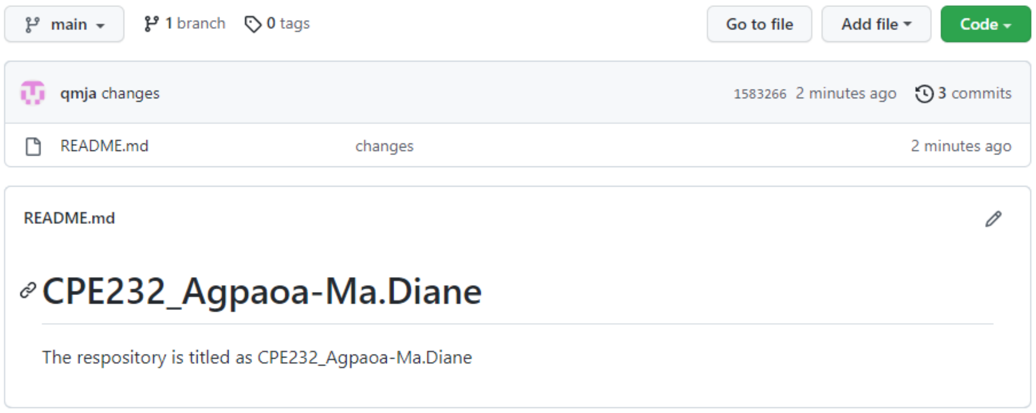
l. Use the command *git push <remote><branch>* to upload the local repository content to GitHub repository. Pushing means to transfer commits from the local repository to the remote repository. As an example, you may issue *git push origin main.*



```
TIPQC@Q5202-10 MINGW64 ~/CPE232_Agpaoa-Ma.Diane (main)
$ git push origin main
Enumerating objects: 8, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (6/6), 550 bytes | 275.00 KiB/s, done.
Total 6 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:qmja/CPE232_Agpaoa-Ma.Diane.git
   e2eda18..1583266  main -> main
```

Figure 3.17 Uploading the local repository content to GitHub repository

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_Agpaoa-Ma.Diane

The respository is titled as CPE232_Agpaoa-Ma.Diane

Figure 3.18 Verifying the changes that have been made to README.md

I noticed that the last commit was 2 minutes ago, also the git commit command "changes" is seen in the figure above. Lastly, the README.md file has been edited based on the text I wrote.

Reflections:

Answer the following:

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

   We used the command ls -la .ssh to verify that we created the key, we also used the command ls in order to verify if we successfully cloned the GitHub.

4. How important is the inventory file?

Inventory files are very important to display and monitor the individual hosts or the user-defined groups of hosts.

Conclusions/Learnings:

In conclusion, this activity helped me to learn and understand how to configure remote and local machines to connect via SSH using a KEY instead of using the password. This activity also helped us to learn how to verify connectivity to remote servers. In addition, I learned how to set up a Git Repository using local and remote repositories. Lastly, this activity is very helpful to learn how to configure and run ad hoc commands from local machines to remote servers. I was able to practice and improve my skills in using the SSH-program.