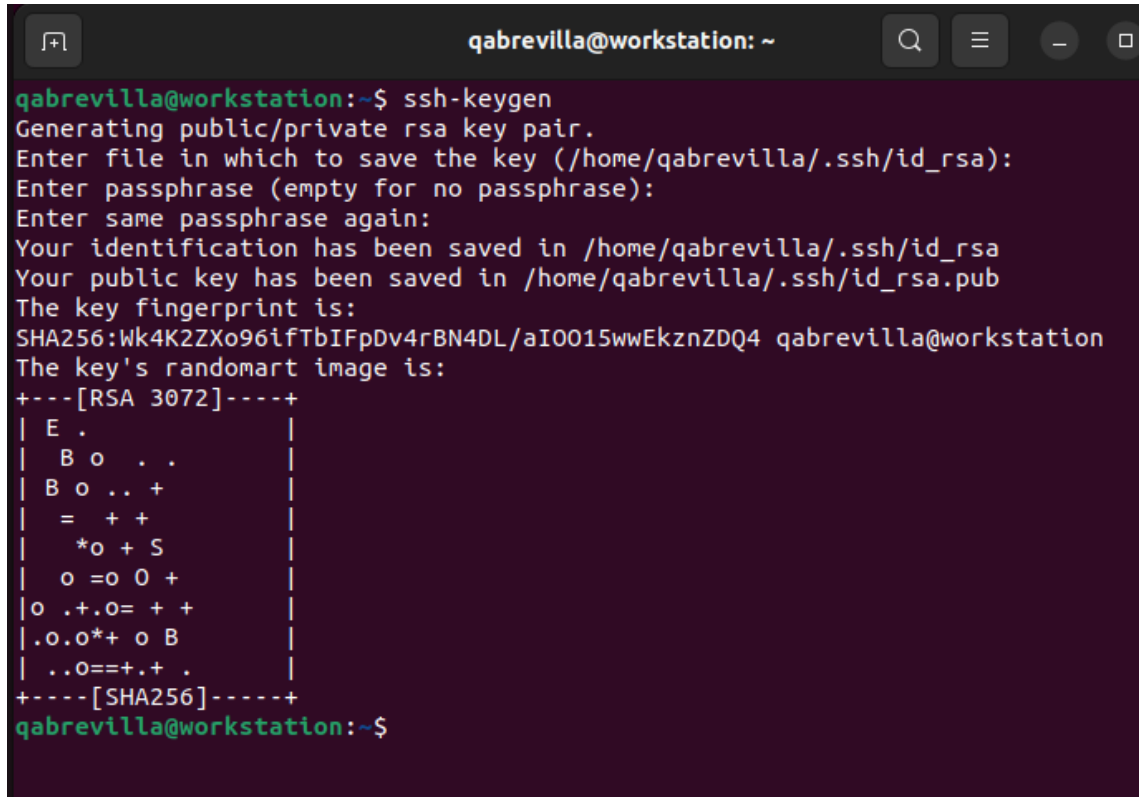


Name: Revilla, Aldwin Joseph B.	Date Performed: 08/27/2023
Course/Section: CPE31S5	Date Submitted: 08/28/2023
Instructor: Engr. Roman Richard	Semester and SY: 2023-2024
Activity 2: SSH Key-Based Authentication and Setting up Git	
1. Objectives: 1.1 Configure remote and local machine to connect via SSH using a KEY instead of using a password 1.2 Create a public key and private key 1.3 Verify connectivity 1.4 Setup Git Repository using local and remote repositories 1.5 Configure and Run ad hoc commands from local machine to remote servers	
Part 1: Discussion It is assumed that you are already done with the last Activity (Activity 1: Configure Network using Virtual Machines). <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`.



```
qabrevilla@workstation: ~  
qabrevilla@workstation:~$ ssh-keygen  
Generating public/private rsa key pair.  
Enter file in which to save the key (/home/qabrevilla/.ssh/id_rsa):  
Enter passphrase (empty for no passphrase):  
Enter same passphrase again:  
Your identification has been saved in /home/qabrevilla/.ssh/id_rsa  
Your public key has been saved in /home/qabrevilla/.ssh/id_rsa.pub  
The key fingerprint is:  
SHA256:Wk4K2ZXo96IfTbIFpDv4rBN4DL/aIO015wwEkznZDQ4 qabrevilla@workstation  
The key's randomart image is:  
+---[RSA 3072]-----+  
| E .  
| B o . .  
| B o .. +  
| = + +  
| *o + S  
| o =o O +  
|o .+.o= + +  
|.o.o*+ o B  
| ..o==+.+.  
+----[SHA256]-----+  
qabrevilla@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.

```

qabrevilla@workstation:~$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/qabrevilla/.ssh/id_rsa):
/home/qabrevilla/.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/qabrevilla/.ssh/id_rsa
Your public key has been saved in /home/qabrevilla/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:j3ah2H7tQW+pAogYx+9u6aBqxrHLMcbNqtvqHW++Pqo qabrevilla@workstation
The key's randomart image is:
+---[RSA 4096]---+
|
|.
|. o
|+ o .S . .
|.. . oo.+ o . .
|o =.....=.o.. +
|.O++o++o .o .+
|  o= .. oo
|_SHA256_]-----+
Help
qabrevilla@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.
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`.

```

qabrevilla@workstation:~$ ls -la .ssh
total 24
drwx----- 2 qabrevilla qabrevilla 4096 Aug 27 16:14 .
drwxr-x--- 16 qabrevilla qabrevilla 4096 May 17 05:40 ..
-rw----- 1 qabrevilla qabrevilla 3389 Aug 27 16:16 id_rsa
-rw-r--r-- 1 qabrevilla qabrevilla 748 Aug 27 16:16 id_rsa.pub
-rw----- 1 qabrevilla qabrevilla 1956 Aug 23 14:15 known_hosts
-rw----- 1 qabrevilla qabrevilla 1120 Aug 23 14:15 known_hosts.old
qabrevilla@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.

```

qabrevilla@workstation:~$ ssh-copy-id
Usage: /usr/bin/ssh-copy-id [-h|-?|-f|-n|-s] [-i [identity_file]] [-p port]
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 used if the remote only allows sftp
    -h|-?: print this help
qabrevilla@workstation:~$

```

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

```

qabrevilla@workstation:~$ ssh-copy-id -i ~/.ssh/id_rsa qabrevilla@workstation
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/qabrevilla/.ssh/id_rsa.pub"
The authenticity of host 'workstation (127.0.0.1)' can't be established.
ED25519 key fingerprint is SHA256:ZqyUlgF1cnLUCxSPEVOE0/uX1Ak1GimD+H9RWuW2l0
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 prompted
now it is to install the new keys
qabrevilla@workstation's password:

Number of key(s) added: 1

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

qabrevilla@workstation:~$

```

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.

Server 1:

```

qabrevilla@workstation:~$ ssh-copy-id -i ~/.ssh/id_rsa qabrevilla@server1
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/qabrevilla/.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 prompted
now it is to install the new keys
qabrevilla@server1's password:

Number of key(s) added: 1

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

qabrevilla@workstation:~$

```

Server 2:

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

Number of key(s) added: 1

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

qabrevilla@workstation:~$
```

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?

Server 1:

```
qabrevilla@server1: ~
and check to make sure that only the key(s) you wanted were added.

qabrevilla@workstation:~$ ssh qabrevilla@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 08:26:17 AM UTC 2023

System load:  0.0537109375      Processes:           121
Usage of /:   25.7% of 11.21GB   Users logged in:    1
Memory usage: 5%                IPv4 address for enp0s3: 192.168.56.102
Swap usage:   0%

 * Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s
   just raised the bar for easy, resilient and secure K8s cluster deployment.

https://ubuntu.com/engage/secure-kubernetes-at-the-edge

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 08:24:26 2023
qabrevilla@server1:~$
```

Server 2:

```
qabrevilla@workstation:~$ ssh qabrevilla@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 08:37:50 AM UTC 2023

System load:  0.0               Processes:            112
Usage of /:   25.7% of 11.21GB  Users logged in:     1
Memory usage: 5%               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
Internet connection or proxy settings

Last login: Sun Aug 27 08:24:38 2023
qabrevilla@server2:~$
```

It does not ask for a password because we are able to set the public key to the server. With this method we are able to access safely login in to our server without needing a password.

Reflections:

Answer the following:

1. How will you describe the ssh-program? What does it do?
ssh-program is a network protocol used for configuring computers and servers. It offers a remotely safe access over the computers and is commonly used to establish a secure and encrypted connection between workstation and servers.
2. How do you know that you already installed the public key to the remote servers?
After using "ssh-copy-id -i ~/.ssh/id_rsa user@[servername]", there will be a prompt below that says "Number of keys added" meaning the public key is added 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**).

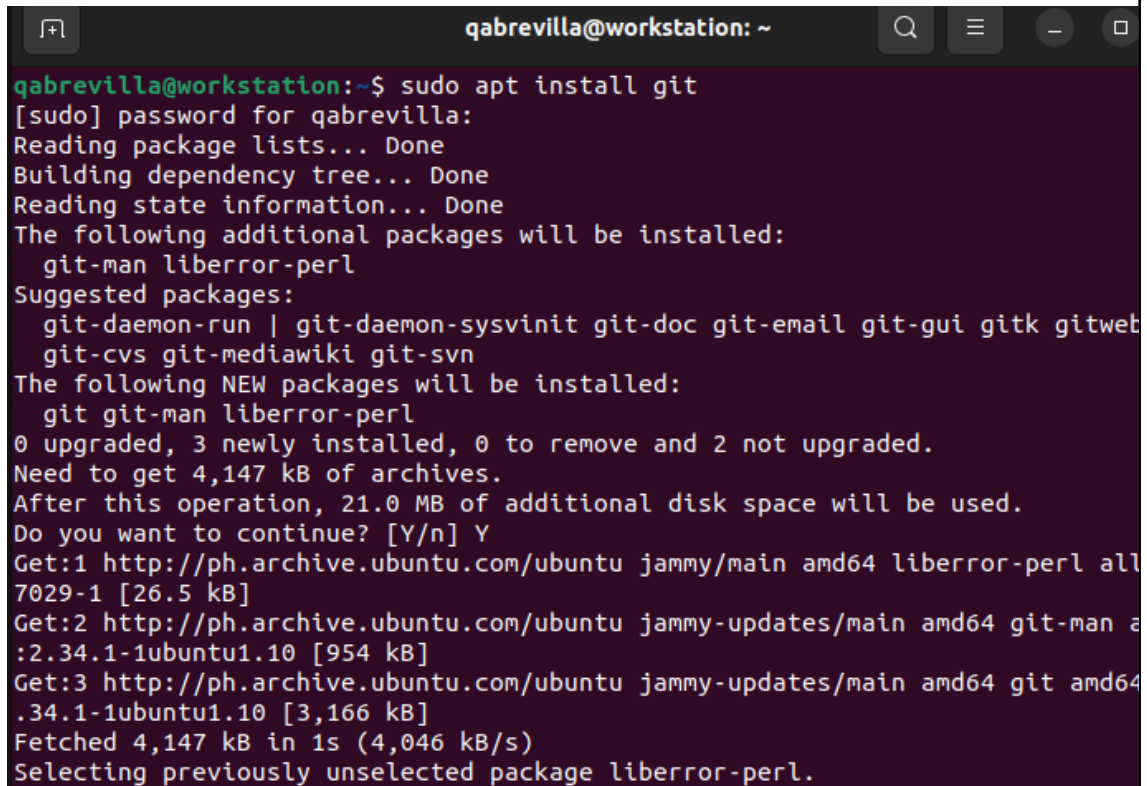
Set up Git

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*



```
qabrevilla@workstation: ~  
qabrevilla@workstation:~$ sudo apt install git  
[sudo] password for qabrevilla:  
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  
The following NEW packages will be installed:  
  git git-man liberror-perl  
0 upgraded, 3 newly installed, 0 to remove and 2 not upgraded.  
Need to get 4,147 kB of archives.  
After this operation, 21.0 MB of additional disk space will be used.  
Do you want to continue? [Y/n] Y  
Get:1 http://ph.archive.ubuntu.com/ubuntu jammy/main amd64 liberror-perl all  
7029-1 [26.5 kB]  
Get:2 http://ph.archive.ubuntu.com/ubuntu jammy-updates/main amd64 git-man a  
:2.34.1-1ubuntu1.10 [954 kB]  
Get:3 http://ph.archive.ubuntu.com/ubuntu jammy-updates/main amd64 git amd64  
.34.1-1ubuntu1.10 [3,166 kB]  
Fetched 4,147 kB in 1s (4,046 kB/s)  
Selecting previously unselected package liberror-perl.
```

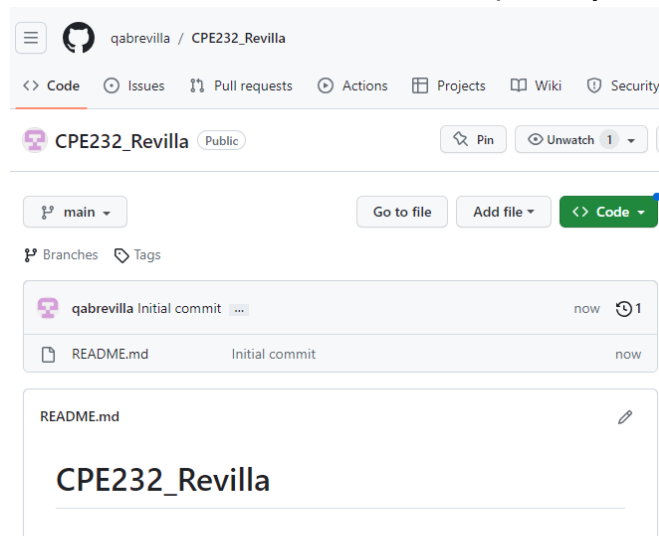
2. After the installation, issue the command *which git* again. The directory of git is usually installed in this location: *user/bin/git*.

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

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

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

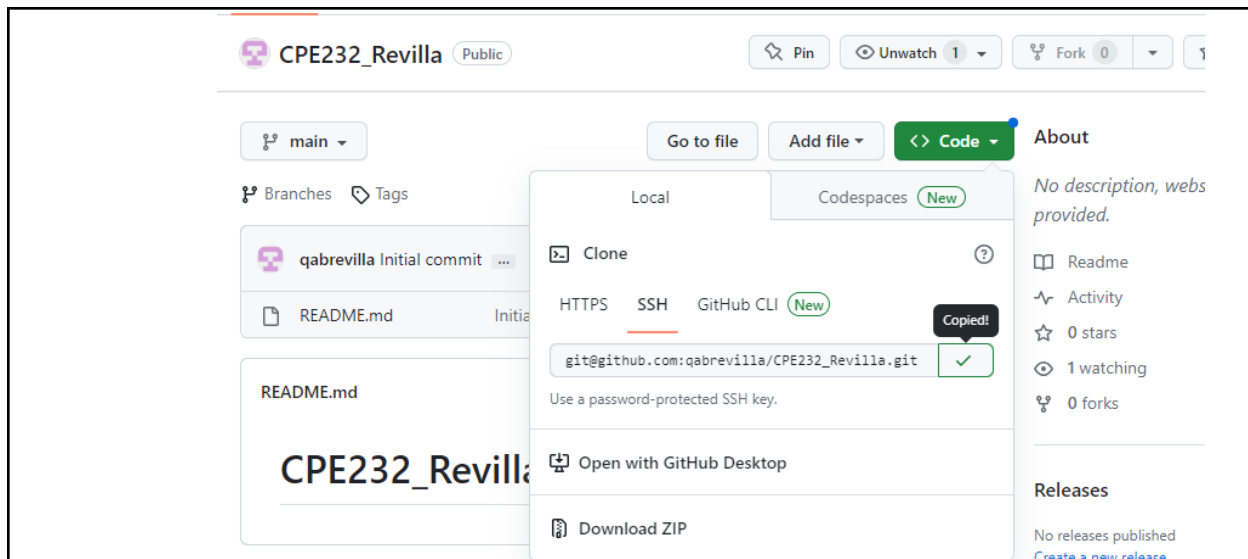
4. Using the browser in the local machine, go to www.github.com.
5. Sign up in case you don't have an account yet. Otherwise, login to your GitHub account.
 - a. Create a new repository and name it as CPE232_yourname. Check Add a README file and click Create repository.



- 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.

- 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.

```
qabrevilla@workstation:~$ git clone git@github.com:qabrevilla/CPE232_
Cloning into 'CPE232_Revilla'...
The authenticity of host 'github.com (20.205.243.166)' can't be estab
ED25519 key fingerprint is SHA256:+DiY3wvvV6TuJJhbpZisF/zLDA0zPMSvHdk
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])?
Warning: Permanently added 'github.com' (ED25519) to the list of know
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
Receiving objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
qabrevilla@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`.

```
qabrevilla@workstation:~$ ls
CPE232_Revilla  Documents  ha        Music     Public  Templ
Desktop         Downloads  ha.sh     Pictures  snap    Video
qabrevilla@workstation:~$ cd ~/CPE232_Revilla
qabrevilla@workstation:~/CPE232_Revilla$ ls
README.md
qabrevilla@workstation:~/CPE232_Revilla$
```

- g. Use the following commands to personalize your git.
- `git config --global user.name "Your Name"`

```
qabrevilla@workstation:~$ git config --global user.name "Revilla"
qabrevilla@workstation:~$
```

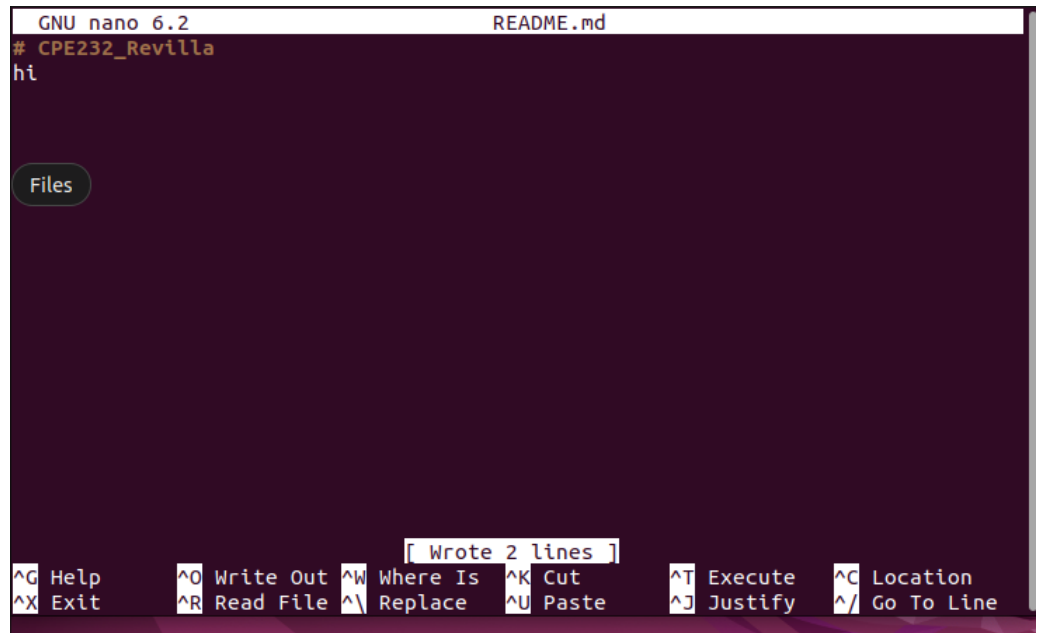
- *git config --global user.email yourname@email.com*

```
qabrevilla@workstation:~$ git config --global user.email qabrevilla@tip.edu.ph
qabrevilla@workstation:~$
```

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

```
qabrevilla@workstation:~$ cat ~/.gitconfig
[user]
    name = Revilla
    email = qabrevilla@tip.edu.ph
qabrevilla@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.



```
GNU nano 6.2                                README.md
# CPE232_Revilla
hi

Files

[ Wrote 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
```

- 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?

```
qabrevilla@workstation:~/CPE232_Revilla$ 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

Untracked files:
  (use "git add" to track)
        .

Help:  Changes added to commit (use "git add" and/or "git commit -a")
qabrevilla@workstation:~/CPE232_Revilla$
```

"git status" command displays the state of the current working directory and the staging area.

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

```
qabrevilla@workstation:~/CPE232_Revilla$ git add README.md
qabrevilla@workstation:~/CPE232_Revilla$
```

- 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.

```
qabrevilla@workstation:~/CPE232_Revilla$ git commit -m "Lopit men"
[main 79aace1] Lopit men
1 file changed, 2 insertions(+), 1 deletion(-)
qabrevilla@workstation:~/CPE232_Revilla$
```

- 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*.

```
qabrevilla@workstation:~/CPE232_Revilla$ git push origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Writing objects: 100% (3/3), 261 bytes | 261.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:qabrevilla/CPE232_Revilla.git
   87efc2f..79aace1  main -> main
qabrevilla@workstation:~/CPE232_Revilla$
```

- 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.

A screenshot of a GitHub repository interface. At the top, the repository name 'qabrevilla' is visible, along with the commit hash '79aace1' and the time '3 minutes ago'. Below this, there are tabs for 'Preview', 'Code', and 'Blame'. The 'Preview' tab is selected, showing a file named 'CPE232_Revilla'. The file content is 'hi'. There are also icons for 'Raw', 'Copy', 'Download', and 'Edit'.

Reflections:

Answer the following:

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

We are able to learn a few basic commands to configure a local machine to connect it to servers using SSH protocol. I was able to learn how to generate a public key and use it instead of using a password to login into the servers. I also learned to set up a Git Repository using local and remote repositories. I find it amazing to control a git repository using a local machine and configuring it was quite difficult for a beginner like me.

4. How important is the inventory file?

Inventory files are an important tool for system administration. It allows administrators to manage, organize, and keep the remote system working. Ansible inventory is also a tool for integrating security and improving optimal efficiency of the local machine and servers.

Conclusions/Learnings:

In this activity, we are able to move 1 step forward in learning more about server management and operating in Linux. The lesson is about SSH, Git Repository, and Ansible. I learned to use a public key to connect into a server without using a password. Using keys is more secure than using passwords to deny hackers and unauthorized users. In the next lesson, we set up a Git Repository as we connect it in local and remote repositories. I was able to edit and configure the Git Repository in the local system remotely. I really enjoyed this activity because I was able to control servers remotely. I know that learning this skill will help me more in the elective that I choose.