

Name: Gabiano, Chris Leonard A.	Date Performed: 08/24/2023
Course/Section: CPE232-s6	Date Submitted:
Instructor: Engr. Taylar	Semester and SY: 3rd year - 1st sem
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`.

```
leonard@workstation:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/leonard/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/leonard/.ssh/id_rsa.
Your public key has been saved in /home/leonard/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:FWluY/4VfNwAlXkGTCpyYcmOPoyP7FDEcBo/KwFrgS0 leonard@workstation
The key's randomart image is:
+---[RSA 2048]---+
|=.      o....=++ |
|E0o    ....o. .= o|
|+o=   o ..o+ .. =.|
|.o o. . . +=.  o +|
|. o+   S+ .  o   |
|o. +    .    .   |
|.. o .    .    .  |
|.o .    .    .   |
|..      .    .   |
+-----[SHA256]-----+
leonard@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.

```
leonard@workstation:~$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/leonard/.ssh/id_rsa):
/home/leonard/.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/leonard/.ssh/id_rsa.
Your public key has been saved in /home/leonard/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:0Qd9kyZ0z0Xre3mMBL+j9atnk0WXhcUb3/JF9g+19hQ leonard@workstation
The key's randomart image is:
+---[RSA 4096]---+
|          .+. |
|          . o.E*|
|          . + * o+@|
|          = * *o=0|
|          S o o =*=|
|          o . =*|
|          *.*|
|          o O.|
|          ..+.+|
+-----[SHA256]-----+
leonard@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`.

```
leonard@workstation:~$ ls -la .ssh
total 20
drwx----- 2 leonard leonard 4096 Aug 24 17:25 .
drwxr-xr-x 16 leonard leonard 4096 Aug 24 17:00 ..
-rw----- 1 leonard leonard 3243 Aug 24 17:28 id_rsa
-rw-r--r-- 1 leonard leonard 745 Aug 24 17:28 id_rsa.pub
-rw-r--r-- 1 leonard leonard 888 Aug 17 18:11 known_hosts
leonard@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.
2. Issue the command similar to this: `ssh-copy-id -i ~/.ssh/id_rsa user@host`

```
leonard@workstation:~$ ssh-copy-id -i ~/.ssh/id_rsa leonard@SERVER1
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/leonard/.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
leonard@server1's password:

Number of key(s) added: 1

Now try logging into the machine, with: "ssh 'leonard@SERVER1'"
and check to make sure that only the key(s) you wanted were added.

leonard@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.

```

Leonard@SERVER1:~$ cd ~/.ssh
bash: cd: too many arguments
Leonard@SERVER1:~$ cd ~/.ssh
Leonard@SERVER1:~/.ssh$ ls
authorized_keys
Leonard@SERVER1:~/.ssh$ cat authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQDE30Lyx7k7ZooUdpB0CwLPgeDHNXWHgDhFIdkB8R7
s55BAad69Hq1bfIU2ZyRMk5xQFNSL133ZVVWJ6+Mkhoq3/MWFEwwH+SvbX/kLqohIGxbEvwvQisKo5C
OCZIGP4EANvWrBhKU2nJf8vexefkonIE7JHbio60deuLqLPDlEx4048Vf5P6L5tRdWze5AR0ff6lB+J
uPbMVNaLD1ZUoBuyT9j4AWZgbU+z8LYudusq0pd9SWoNRkrRUxb+GcL7N8pyQa5Ii+bYjcFg3j783D
kMrQK2VuVnQZTJ2bIRVaFLJpQrzAaHqx/e4ysskey5er+c4oAdoGDvroitIY0Fs8lrNQ6VgGLu/GNbP
eA008YXH8i0n2EW0DopGldgsRkMawr5ohyEW2Gg8ZkzfVwD5chHmGWgjwRpezd1tMsHbLkAgCNxi2cS
VvrTa4Md4CtP4to6BvC10Tkg2rBqR3buWjGpMVjZk0QTNQs+KCsc+o/S1BsBwZXf+HzfMoFyNoEbTxI
Omqt2Xh/BclJ2xDwqgFuqR/0aZXGEORyEdpb5fokCyzTRDz2tkSRiLPCqTvtK9S1lqWBy4bgwP8BcIs
AHmmfyymBE4uh799bhZ+9+8YYl6rStV/tn9+s0RoW3fUnSelYqud0v0iAczad05mRzuqDtaq+702+W9
wCgdvhm3udQ== leonard@workstation
Leonard@SERVER1:~/.ssh$

```

```

Leonard@SERVER2:~$ cd ~/.ssh
Leonard@SERVER2:~/.ssh$ ls
authorized_keys
Leonard@SERVER2:~/.ssh$ cat authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQDE30Lyx7k7ZooUdpB0CwLPgeDHNXWHgDhFIdkB8R7
s55BAad69Hq1bfIU2ZyRMk5xQFNSL133ZVVWJ6+Mkhoq3/MWFEwwH+SvbX/kLqohIGxbEvwvQisKo5C
OCZIGP4EANvWrBhKU2nJf8vexefkonIE7JHbio60deuLqLPDlEx4048Vf5P6L5tRdWze5AR0ff6lB+J
uPbMVNaLD1ZUoBuyT9j4AWZgbU+z8LYudusq0pd9SWoNRkrRUxb+GcL7N8pyQa5Ii+bYjcFg3j783D
kMrQK2VuVnQZTJ2bIRVaFLJpQrzAaHqx/e4ysskey5er+c4oAdoGDvroitIY0Fs8lrNQ6VgGLu/GNbP
eA008YXH8i0n2EW0DopGldgsRkMawr5ohyEW2Gg8ZkzfVwD5chHmGWgjwRpezd1tMsHbLkAgCNxi2cS
VvrTa4Md4CtP4to6BvC10Tkg2rBqR3buWjGpMVjZk0QTNQs+KCsc+o/S1BsBwZXf+HzfMoFyNoEbTxI
Omqt2Xh/BclJ2xDwqgFuqR/0aZXGEORyEdpb5fokCyzTRDz2tkSRiLPCqTvtK9S1lqWBy4bgwP8BcIs
AHmmfyymBE4uh799bhZ+9+8YYl6rStV/tn9+s0RoW3fUnSelYqud0v0iAczad05mRzuqDtaq+702+W9
wCgdvhm3udQ== leonard@workstation
Leonard@SERVER2:~/.ssh$

```

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?

```
leonard@workstation:~$ ssh leonard@server1
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.

91 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 18:12:13 2023 from 192.168.56.101
leonard@server1:~$
```

```
leonard@workstation:~$ ssh leonard@SERVER2
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

New release '20.04.6 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Your Hardware Enablement Stack (HWE) is supported until April 2023.
Last login: Thu Aug 17 18:11:40 2023 from 192.168.56.101
leonard@server2:~$
```

Reflections:

Answer the following:

1. How will you describe the ssh-program? What does it do?
2. How do you know that you already installed the public key to the remote servers?

Part 2: Discussion

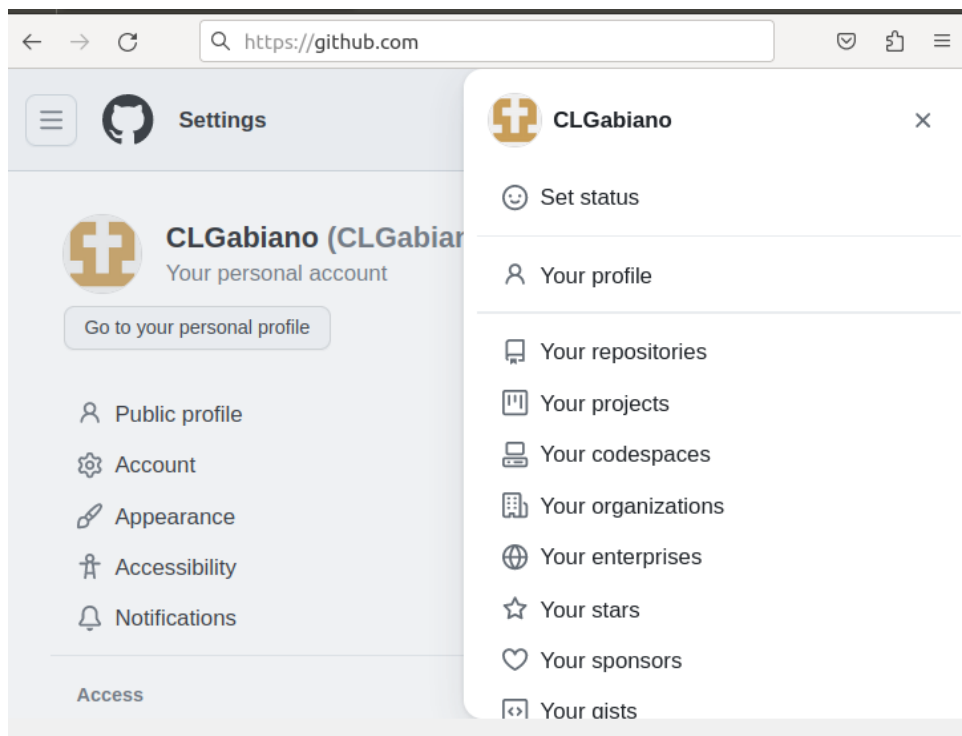
Provide screenshots for each task.

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*

```
leonard@workstation:~$ sudo apt install git -y
[sudo] password for leonard:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following package was automatically installed and is no longer required:
  libllvm7
Use 'sudo apt autoremove' to remove it.
```

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


```
leonard@workstation:~$ which git
/usr/bin/git
leonard@workstation:~$ git --version
```


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

```
leonard@workstation:~$ git --version
git version 2.17.1
leonard@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.


Required fields are marked with an asterisk ().*


Owner *  CLGabiano / Repository name *

 CPE232_Gabiano is available.

Great repository names are short and memorable. Need inspiration? How about [studious-octo-adventure](#) ?

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.](#)

- 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'

Add 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.

```
leonard@workstation:~$ cat .ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQCAQDHmVN9p0S3b0Wc9dGKaXkj4CvRUH5gGUEN48zd3b5
rfVS96Rnx30/rzZfc57PPjt1MftmTqBdcECK+5fCn8qE2Ljs6UzljlSD+oHTz+kkyQ5cRWeUQG4MC/c
uHevosMc5uYLz1B+/MO+0+Ey9gULAagpbHvQgy+47Wdo1JgV/So7mYMhGngTtpMECb+b370y5/PPCB
q8JkrcZXVdk1icumBr9XRGMbn17T5wAnjTS8aGW5xf0cbA0v8fLLBkSUMltRqand7Cqh23AEppMo/L0
2GQ4wD896F4JZQ3kdV8Z66rMA1c2GuNg2ogEVVHAGPERgGMII3nr0ZZ7+bVBr6nIGJ0BRBUyobfkbEK
IOjss9ykG3kemqvr9lAYvw2fxoFp2M9VwttVJH1SwLSsRSyyQ+G/lgbwOfihIXRrMTC9TfIiXMY57jx
oMsYLU/i743xKkxyKL2u32kAI/7VrAF/tbP9JMLAZc1VrV1H+IlQ1ykJIxoz91IZXqZ8BCE0anQf1fe
cbYnWG+g/VeL22BLGK06UnKNR5/GAN0ynPULfZyM4zWG5HK/d+mIP9sc1f0tND0k2M7I2gRfNVix0iV
mJ6evACsuHFz8LZTNnTSDCzVQd64K1tWmgsV8wvSM/5fohwnYV44wvJWXNKbcAi8q0j22R8Rz2gr+x
La0P/x6AXvw== leonard@workstation
leonard@workstation:~$
```

Authentication Keys



cp232

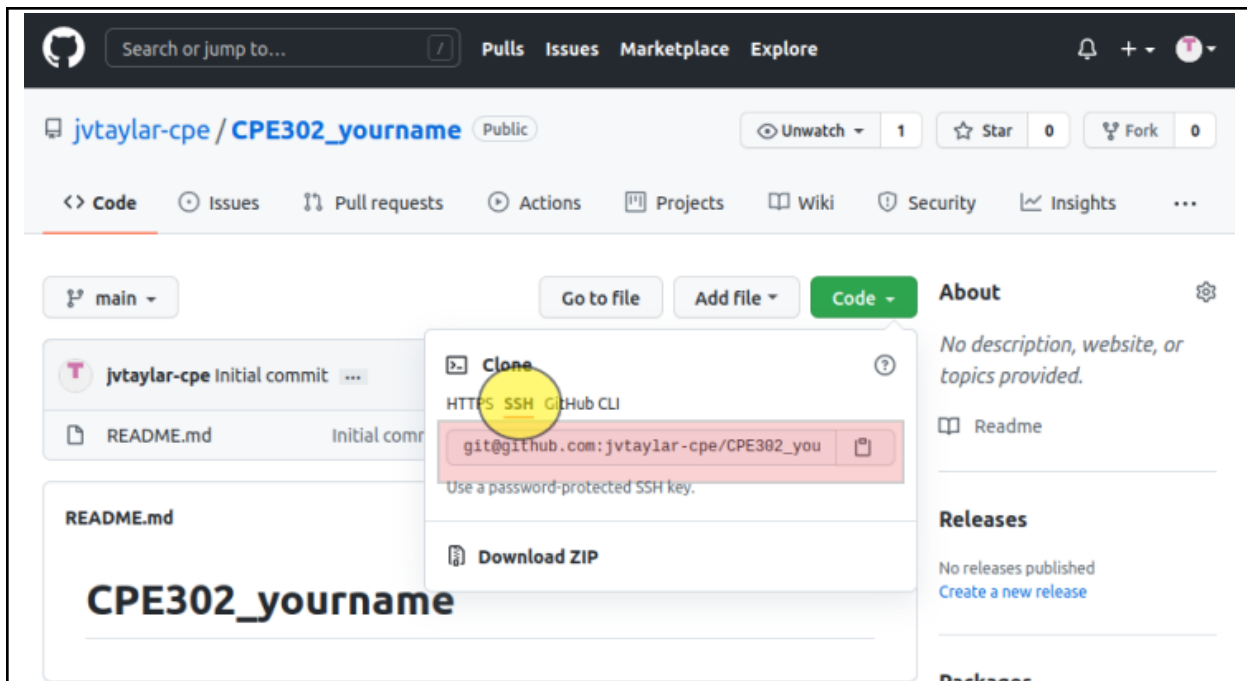
SHA256:odqYdsi65dJaq+9EBwZQt15/pPFUWXbaps/b0rwiTk8

Added on Sep 7, 2023

Never used — Read/write

Delete

- 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.

```
leonard@workstation:~$ git clone git@github.com:CLGabiano/CPE232_Gabiano.git
Cloning into 'CPE232_Gabiano'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.
```

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

```
leonard@workstation:~$ cd CPE232_Gabiano
leonard@workstation:~/CPE232_Gabiano$ ls
README.md
leonard@workstation:~/CPE232_Gabiano$ cat README.md
# CPE232_Gabiano
leonard@workstation:~/CPE232_Gabiano$
```

- 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`

```
leonard@workstation:~$ cat ~/.gitconfig
[user]
    name = leonard
    email = qclagabiano@tip.edu.ph
leonard@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 2.9.3 README.md
#CPE232_Gabiano

print(I love TIP)
print(CPE3156)
```

- 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?

```
leonard@workstation:~/CPE232_Gabiano$ 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")
```

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

```
leonard@workstation:~/CPE232_Gabiano$ git commit -m "Created"
[main e4577a3] Created
1 file changed, 4 insertions(+), 1 deletion(-)
```

- 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.

```
Leonard@workstation:~/CPE232_Gabiano$ git commit -m "Created"
[main e4577a3] Created
1 file changed, 4 insertions(+), 1 deletion(-)
```

- I. 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`.

```
Leonard@workstation:~/CPE232_Gabiano$ 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), 282 bytes | 282.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To github.com:CLGabiano/CPE232_Gabiano.git
fe8f0e6..e4577a3  main -> main
```

- m. On the GitHub repository, verify that the changes have been made to README.md by refreshing the page. Describe the README.md file. You can notice the how long was the last commit. It should be some minutes ago and the message you typed on the git commit command should be there. Also, the README.md file should have been edited according to the text you wrote.

Reflections:

Answer the following:

3. What sort of things have we so far done to the remote servers using ansible commands?
4. How important is the inventory file?

Conclusions/Learnings: