Name: Dean Lenard D Perez	**Date Performed:** 04/09/2024
Course/Section: CPE 212-CPE31S21	**Date Submitted:** 04/09/2024
Instructor:	**Semester and SY:** 2024-2025

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.

```
dldperez@workstation:~$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/dldperez/.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:
SHA256:7l6W9UDdZ7wzVkRixSQ+XjOYLnjyUEi5lOOJ3hOkWcc dldperez@workstation
The key's randomart image is:
+---[RSA 4096]----+
|          .+   +==|
|         .B.E +o*.|
|         O.=..o+oB|
|        + =o... +*|
|       . S+.oo..= |
|        o o=o.o. o|
|         . =.  .  |
|        . o       |
|        .o        |
+----[SHA256]-----+
```

ssh

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.

```
dldperez@workstation:~$ ls -la .ssh
total 36
drwx------  2 dldperez dldperez 4096 Sep  4 03:49 .
drwxr-x--- 15 dldperez dldperez 4096 Sep  4 03:46 ..
-rw-------  1 dldperez dldperez  746 Sep  4 03:44 authorized_keys
-rw-------  1 dldperez dldperez  411 Sep  4 03:49 id_ed25519
-rw-r--r--  1 dldperez dldperez  102 Sep  4 03:49 id_ed25519.pub
-rw-------  1 dldperez dldperez 3389 Sep  4 03:09 id_rsa
-rw-r--r--  1 dldperez dldperez  746 Sep  4 03:09 id_rsa.pub
-rw-------  1 dldperez dldperez 1688 Sep  4 03:44 known_hosts
-rw-r--r--  1 dldperez dldperez  142 Sep  4 00:05 known_hosts.old
```

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*

```
dldperez@workstation:~$ ssh-copy-id -i /home/dldperez/id_rsa dldperez@workstation
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/dldperez/id_rsa.pub"
The authenticity of host 'workstation (192.168.56.103)' can't be established.
ED25519 key fingerprint is SHA256:SgiMW8s4YuWRUpL+9uPnuhRXGA5IkqF4ebGm432sERA.
This host key is known by the following other names/addresses:
    ~/.ssh/known_hosts:1: [hashed name]
    ~/.ssh/known_hosts:4: [hashed name]
    ~/.ssh/known_hosts:5: [hashed name]
    ~/.ssh/known_hosts:6: [hashed name]
    ~/.ssh/known_hosts:7: [hashed name]
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 tha
t are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it i
s to install the new keys
dldperez@workstation's password:

Number of key(s) added: 1

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

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?

```
dldperez@workstation:~$ ssh dldperez@server1
Welcome to Ubuntu 24.04.1 LTS (GNU/Linux 6.8.0-41-generic x86_64)

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

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: Wed Sep  4 03:53:31 2024 from 192.168.56.103
dldperez@server1:~$
```

It did not asked for a password because I used the copy id tool of the keygen in workstation into the server1. So, it acknowledges the server1.

Reflections:

Answer the following:

1. How will you describe the ssh-program? What does it do?
   It is a remote connection to other devices over an unsecured network. I
2. How do you know that you already installed the public key to the remote servers?
   You need to issue the command to copy it to your remote server. Or you can access the server and find out if it is installed in the .ssh directory.

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*

```
dldperez@workstation:~$ sudo apt install git
[sudo] password for dldperez:
Sorry, try again.
[sudo] password for dldperez:
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 4,804 kB of archives.
After this operation, 24.5 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://archive.ubuntu.com/ubuntu noble/main amd64 liberror-perl all 0.17029-2
 [25.6 kB]
Get:2 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 git-man all 1:2.43.
0-1ubuntu7.1 [1,100 kB]
Get:3 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 git amd64 1:2.43.0-
1ubuntu7.1 [3,679 kB]
Fetched 4,804 kB in 3s (1,629 kB/s)
Selecting previously unselected package liberror-perl.
(Reading database ... 150951 files and directories currently installed.)
Preparing to unpack .../liberror-perl_0.17029-2_all.deb ...
Unpacking liberror-perl (0.17029-2) ...
Selecting previously unselected package git-man.
Preparing to unpack .../git-man_1%3a2.43.0-1ubuntu7.1_all.deb ...
Unpacking git-man (1:2.43.0-1ubuntu7.1) ...
```

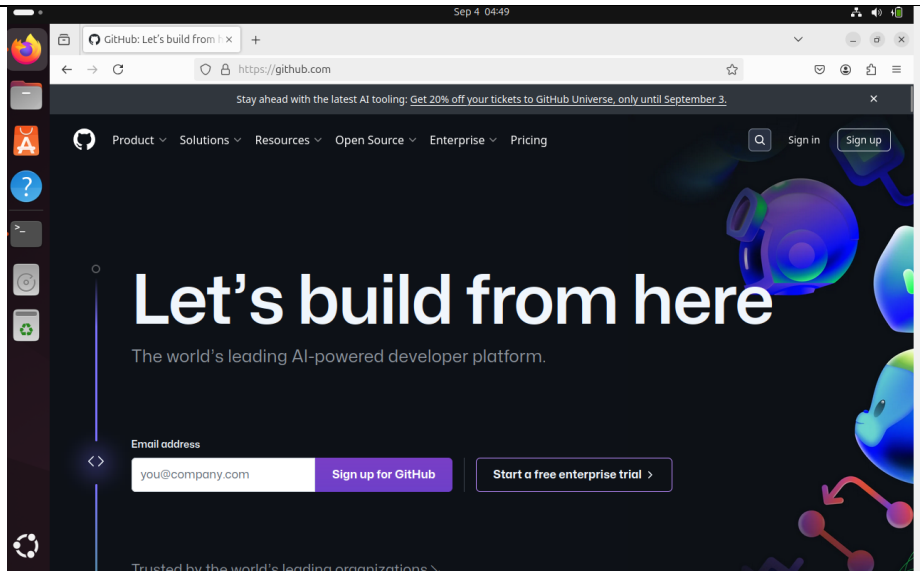
2. After the installation, issue the command *which git* again. The directory of git is usually installed in this location: *user/bin/git.*

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

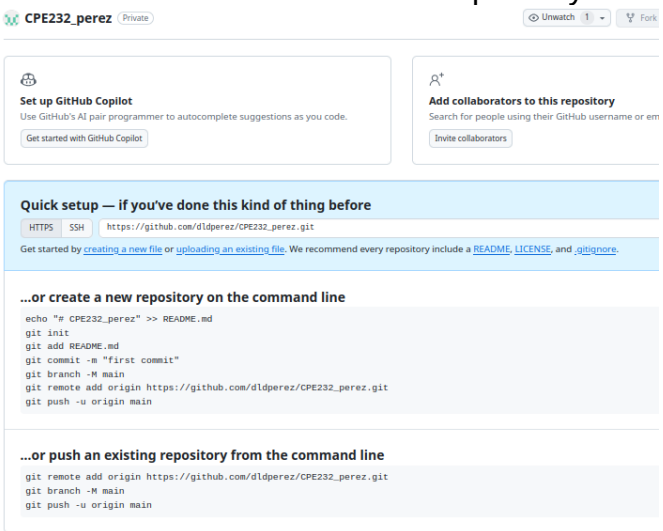
3. The version of git installed in your device is the latest. Try issuing the command *git --version* to know the version installed.

```
dldperez@workstation:~$ git --version
git version 2.43.0
```

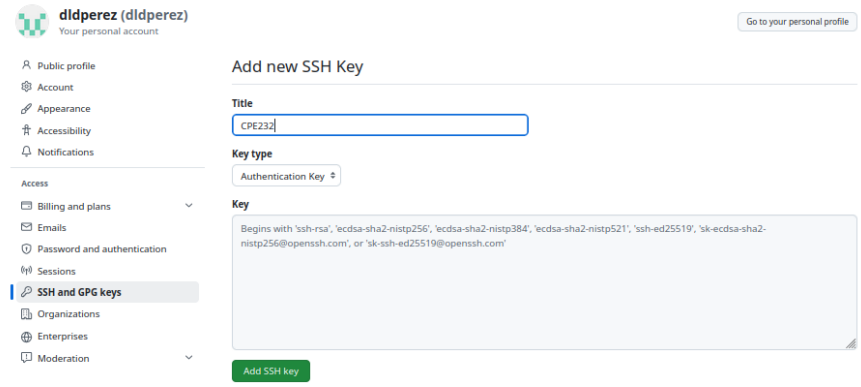
4. Using the browser in the local machine, go to www.github.com.



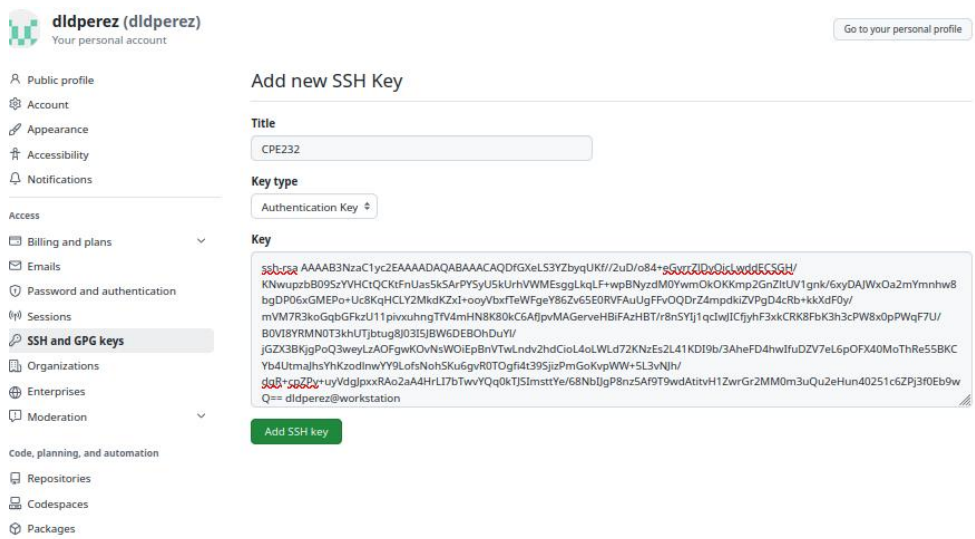
5. Sign up in case you don't have an account yet. Otherwise, login to your GitHub account.
    a. Create a new repository and name it as CPE232_yourname. Check Add a README file and click Create repository.

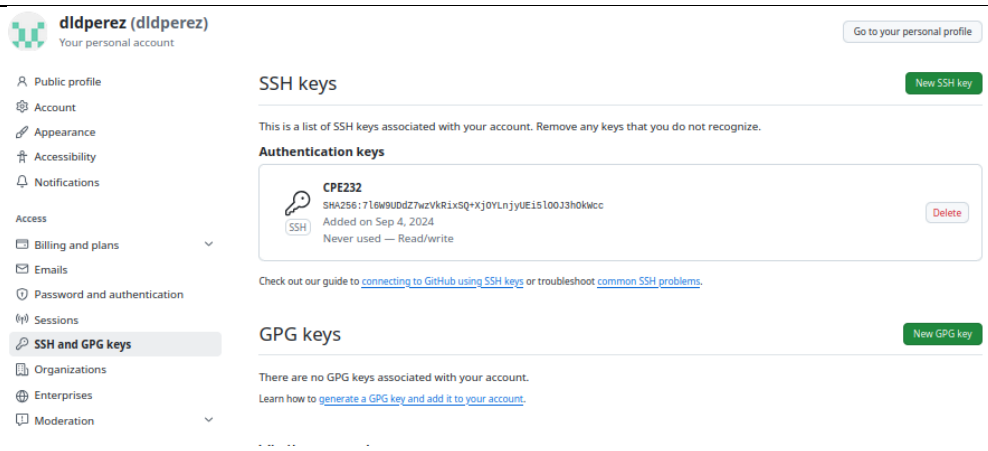


    b. Create a new SSH key on GitHub. Go your profile's setting and click SSH and GPG keys. If there is an existing key, make sure to delete it. To create a new SSH keys, click New SSH Key. Write CPE232 key as the title of the key.

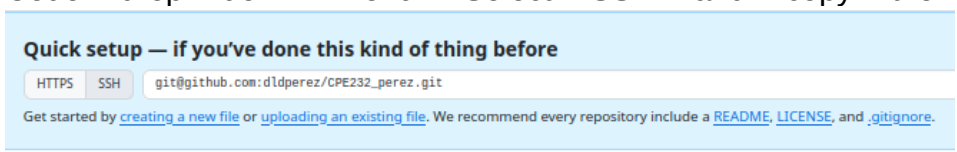


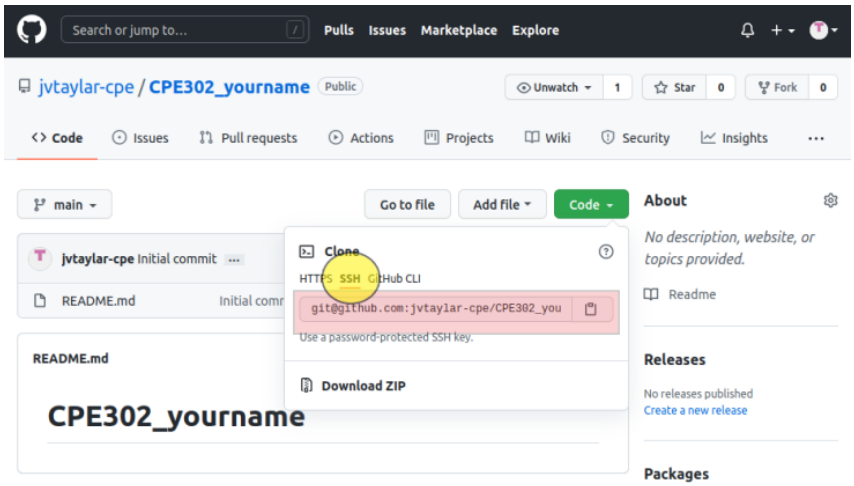
c. On the local machine's terminal, issue the command cat .ssh/id_rsa.pub and copy the public key. Paste it on the GitHub key and press Add SSH key.





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

```
dldperez@workstation:~$ git clone git@github.com:dldperez/CPE232_perez.git
Cloning into 'CPE232_perez'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 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.

```
dldperez@workstation:~$ ls
CPE232_perez  Documents  Music     Public  Templates
Desktop       Downloads  Pictures  snap    Videos
dldperez@workstation:~$ cd CPE232_perez
dldperez@workstation:~/CPE232_perez$ ls
README.md
```

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*

```
dldperez@workstation:~/CPE232_perez$ git config --global user.name "Dean Lenard Perez"
dldperez@workstation:~/CPE232_perez$ git config --global user.email qdldperez@tip.edu.ph
dldperez@workstation:~/CPE232_perez$ cat ~/.gitcongih
cat: /home/dldperez/.gitcongih: No such file or directory
dldperez@workstation:~/CPE232_perez$ cat ~/.gitconfig
[user]
        name = Dean Lenard Perez
        email = qdldperez@tip.edu.ph
dldperez@workstation:~/CPE232_perez$
```

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.

```
dldperez@workstation: ~/CPE232_perez
  GNU nano 7.2                          README.md *
# CPE232_perez
```

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?

```
dldperez@workstation:~/CPE232_perez$ 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")
```

It shows the status/changes done on repository.

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

```
dldperez@workstation:~/CPE232_perez$ git add README.md
dldperez@workstation:~/CPE232_perez$ git status
On branch main
Your branch is up to date with 'origin/main'.

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

dldperez@workstation:~/CPE232_perez$
```

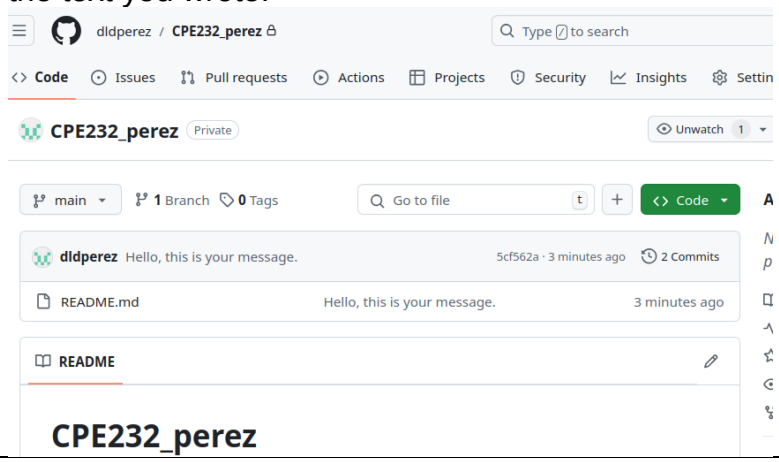
k. Use the *git commit -m "your message"* to create a snapshot of the staged changes along the timeline of the Git projects history. The use of this command is required to select the changes that will be staged for the next commit.

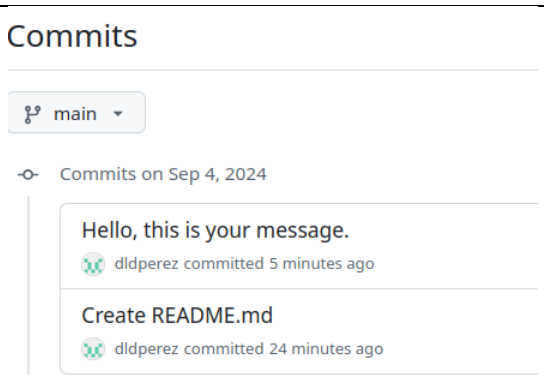
```
dldperez@workstation:~/CPE232_perez$ git commit -m "Hello, this is you
[main 5cf562a] Hello, this is your message.
 1 file changed, 1 insertion(+)
```

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.*

```
dldperez@workstation:~/CPE232_perez$ git push origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Writing objects: 100% (3/3), 282 bytes | 282.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:dldperez/CPE232_perez.git
   211c67b..5cf562a  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.





It showed the changes I made to the repository that I made.

Reflections:

Answer the following:

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

   We created changes on the files remotely, executed the commands remotely, we add and edit files, we created username and email.

4. How important is the inventory file?

   It is important because it shows the servers that Ansible should manage. It also shows the necessary details like IP address, ssh ports to connect with the servers.

Conclusions/Learnings:

This activity taught us how to connect remotely using ssh in different ways like using a keygen instead of a password to connect, using a web browser to connect remotely using the terminal and to create and modify the files we created remotely. The skills to which remotely connect to a server is vital because small businesses or big enterprises use this. Understanding this helps us to efficiently manage a remote server and securing a good and secure connection.