

|  |                                  |
|--|----------------------------------|
| <b>Name:</b> Aaron Jonathan G. Valencia  | <b>Date Performed:</b> 9/9/2024  |
| <b>Course/Section:</b> CPE 212   | <b>Date Submitted:</b> 9/11/2024 |
| <b>Instructor:</b>   | <b>Semester and SY:</b> 1st sem  |
| <b>Activity 2: SSH Key-Based Authentication and Setting up Git</b>   |                                  |
| <b>1. Objectives:</b> <ul style="list-style-type: none"> <li>1.1 Configure remote and local machine to connect via SSH using a KEY instead of using a password</li> <li>1.2 Create a public key and private key</li> <li>1.3 Verify connectivity</li> <li>1.4 Setup Git Repository using local and remote repositories</li> <li>1.5 Configure and Run ad hoc commands from local machine to remote servers</li> </ul>  |                                  |
| <b>Part 1: Discussion</b> <p>It is assumed that you are already done with the last Activity (<b>Activity 1: Configure Network using Virtual Machines</b>). <i>Provide screenshots for each task.</i></p> <p>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.</p> <p><b>What Is ssh-keygen?</b></p> <p>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.</p> <p><b>SSH Keys and Public Key Authentication</b></p> <p>The SSH protocol uses public key cryptography for authenticating hosts and users. The authentication keys, called SSH keys, are created using the keygen program.</p> <p>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.</p> <p>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.</p> |                                  |
| <b>Task 1: Create an SSH Key Pair for User Authentication</b> <ul style="list-style-type: none"> <li>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</li> </ul>   |                                  |

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*.

```
valencia@workstation:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/valencia/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/valencia/.ssh/id_rsa
Your public key has been saved in /home/valencia/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:mDtZkX8vm2F6qRR00h5Aof0M2FhWLB2UvNukeEJhy/bI valencia@workstation
The key's randomart image is:
+---[RSA 3072]-----+
|
|      +=B+o      |
|    . +X+. =     |
|   o+ B+ .       |
|  ooOo .         |
| o S+O=.         |
|   +E=o+..       |
|   +  + +. .     |
|   . . oo=       |
|   ooo           |
+-----[SHA256]-----+
valencia@workstation:~$ s
```

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.

```
valencia@workstation:~$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/valencia/.ssh/id_rsa):
/home/valencia/.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/valencia/.ssh/id_rsa
Your public key has been saved in /home/valencia/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:gVeE9t2B19F30z4+Jz1sXYuvnvkWTEH97bLh0lWZc+s valencia@workstation
The key's randomart image is:
+---[RSA 4096]-----+
|
|      oo ..++|
|    .o. . o.*|
|   ..o. . o .X|
|    . . . .B=|
|    S      +O=|
|           +*o|
|           o==*|
|           ..*E+|
|           oB*++|
+-----[SHA256]-----+
valencia@workstation:~$ s
```

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`.

```
valencia@workstation:~$ ls -la .ssh
total 24
drwx----- 2 valencia valencia 4096 Sep 13 10:38 .
drwxr-x--- 15 valencia valencia 4096 Sep 13 10:31 ..
-rw----- 1 valencia valencia 3389 Sep 13 10:39 id_rsa
-rw-r--r-- 1 valencia valencia 746 Sep 13 10:39 id_rsa.pub
-rw----- 1 valencia valencia 2098 Sep 13 10:34 known_hosts
-rw----- 1 valencia valencia 1120 Sep 13 10:32 known_hosts.old
valencia@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.

```
valencia@workstation:~$ ssh-copy-id
Usage: /usr/bin/ssh-copy-id [-h|-?|-f|-n|-s] [-i [identity_file]] [-p port] [-F 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 useful if the r
emote only allows sftp
-h|-?: print this help
valencia@workstation:~$ S
```

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

```
valencia@workstation:~$ ssh-copy-id -i ~/.ssh/id_rsa valencia@server1
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/valencia/.ssh/id_rsa.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that ar
e 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
valencia@server1's password:

Number of key(s) added: 1

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

```
valencia@workstation:~$ ssh valencia@server1
Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 6.8.0-40-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

New release '24.04.1 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Fri Sep 13 10:35:00 2024 from 192.168.1.13
valencia@server1:~$
```

```
valencia@workstation:~$ ssh valencia@server2
Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 6.8.0-40-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

New release '24.04.1 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Fri Sep 13 10:32:52 2024 from 192.168.1.13
valencia@server2:~$
```

- No, because it is using the ssh keygen to verify the authentication from workstation to server1 or server2

### Reflections:

Answer the following:

1. How will you describe the ssh-program? What does it do?
  - You are able to login to different connected servers without going to the physical server. You can execute commands in the server using the workstation.
2. How do you know that you already installed the public key to the remote servers?
  - When you are able to switch from server1 or server2 without the need of the server's password.

### Part 2: Discussion

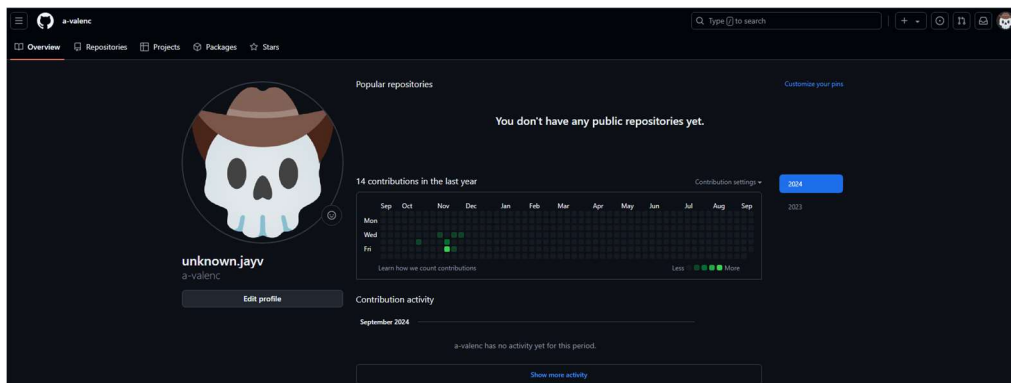
*Provide screenshots for each task.*

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*



```

valencia@workstation:~$ sudo apt install git
[sudo] password for valencia:
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 0 not upgraded.
Need to get 4,146 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://us.archive.ubuntu.com/ubuntu jammy/main amd64 liberror-perl all 0.17029-1 [26.5 kB]
Get:2 http://us.archive.ubuntu.com/ubuntu jammy-updates/main amd64 git-man all 1:2.34.1-1ubuntu1.11 [955 kB]
Get:3 http://us.archive.ubuntu.com/ubuntu jammy-updates/main amd64 git amd64 1:2.34.1-1ubuntu1.11 [3,165 kB]
Fetched 4,146 kB in 3s (1,535 kB/s)
Selecting previously unselected package liberror-perl.
(Reading database ... 172451 files and directories currently installed.)
Preparing to unpack .../liberror-perl_0.17029-1_all.deb ...
Unpacking liberror-perl (0.17029-1) ...
Selecting previously unselected package git-man.
Preparing to unpack .../git-man_1%3a2.34.1-1ubuntu1.11_all.deb ...
Unpacking git-man (1:2.34.1-1ubuntu1.11) ...
Selecting previously unselected package git.
Preparing to unpack .../git_1%3a2.34.1-1ubuntu1.11_amd64.deb ...
Unpacking git (1:2.34.1-1ubuntu1.11) ...
Setting up liberror-perl (0.17029-1) ...
Setting up git-man (1:2.34.1-1ubuntu1.11) ...
Setting up git (1:2.34.1-1ubuntu1.11) ...

```

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

```

valencia@workstation:~$ which git
/usr/bin/git
valencia@workstation:~$

```


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

```

valencia@workstation:~$ git --version
git version 2.34.1
valencia@workstation:~$


```


4. Using the browser in the local machine, go to [www.github.com](https://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.

Owner \*  a-valenc / Repository name \* CPE232\_VALENCIA  
✔ CPE232\_VALENCIA is available.


Great repository names are short and memorable. Need inspiration? How about [shiny-umbrella](#) ?


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.](#)

Add .gitignore  
.gitignore template: **None**   
Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

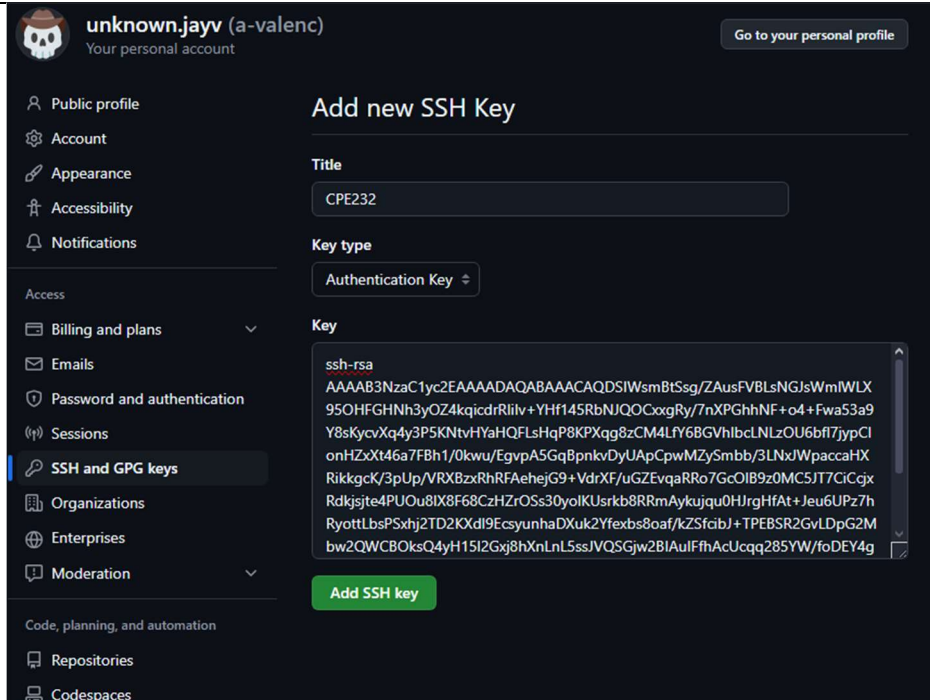
Choose a license  
License: **None**   
A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

This will set  **main** as the default branch. Change the default name in your [settings](#).

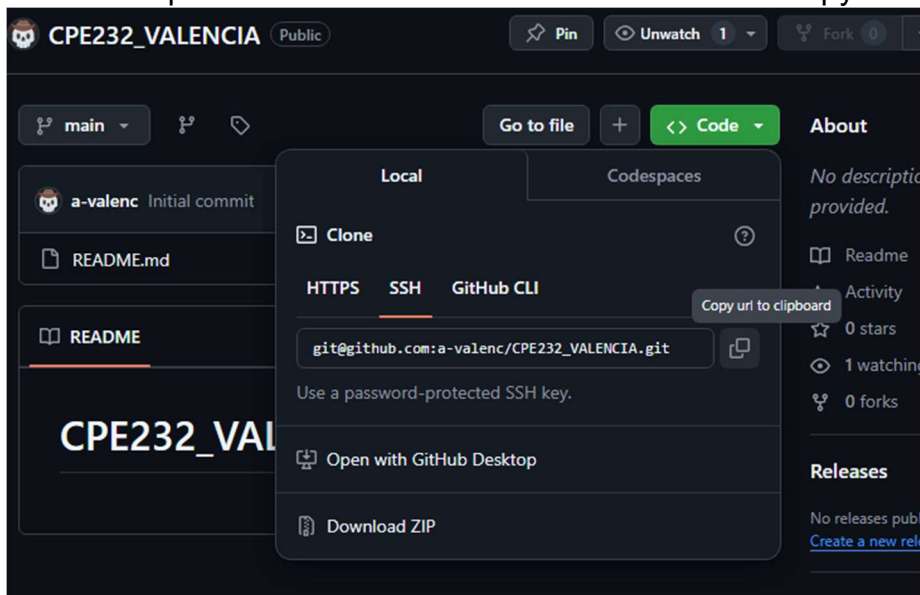
 You are creating a public repository in your personal account.

[Create repository](#)

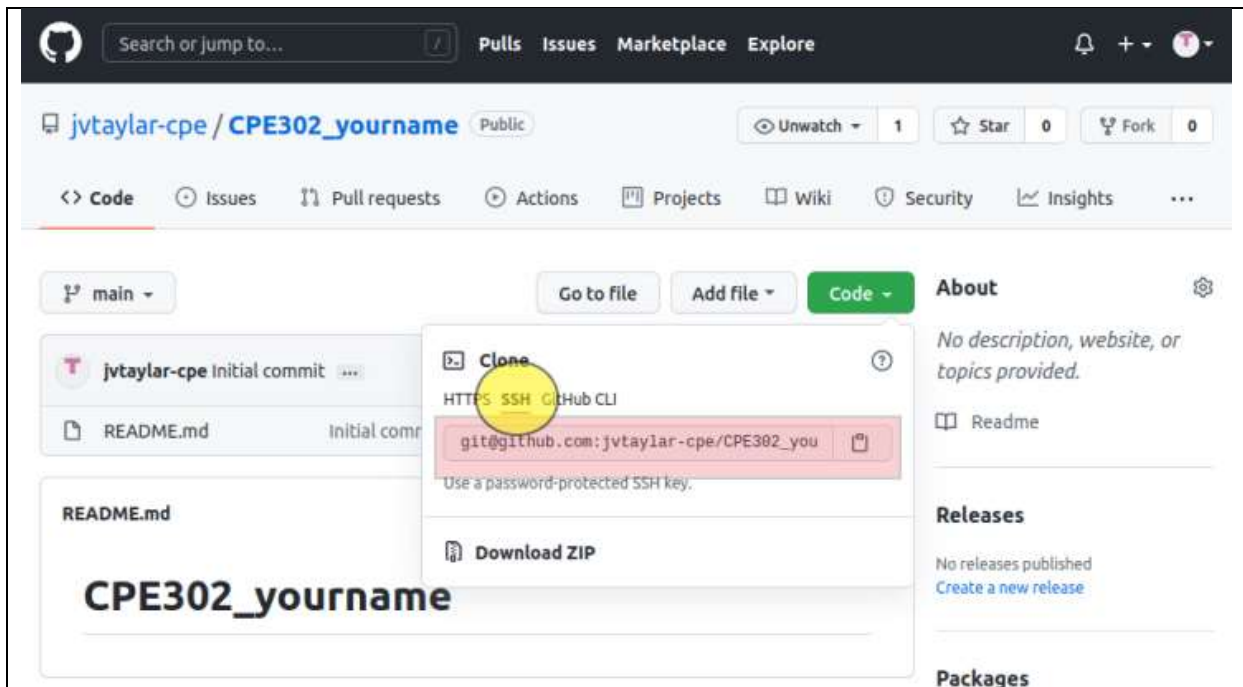
- b. Create a new SSH key on GitHub. Go your profile's setting and click SSH and GPG keys. If there is an existing key, make sure to delete it. To create a new SSH keys, click New SSH Key. Write CPE232 key as the title of the key.
- c. On the local machine's terminal, issue the command `cat .ssh/id_rsa.pub` 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.

```
valencia@workstation:~$ git clone git@github.com:a-valenc/CPE232_VALENCIA.git
Cloning into 'CPE232_VALENCIA'...
The authenticity of host 'github.com (20.205.243.166)' can't be established.
ED25519 key fingerprint is SHA256:+DiY3wvV6TuJJhbpZisF/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 (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`.


```
valencia@workstation:~$ ls
CPE232_VALENCIA  Documents  Music      Public  Templates
Desktop          Downloads  Pictures   snap    Videos
valencia@workstation:~$ cd CPE232_valencia
bash: cd: CPE232_valencia: No such file or directory
valencia@workstation:~$ cd CPE232_VALENCIA
valencia@workstation:~/CPE232_VALENCIA$ ls
README.md
valencia@workstation:~/CPE232_VALENCIA$ s
```

- 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`

```
valencia@workstation:~/CPE232_VALENCIA$ git config --global user.name VALENCIA
valencia@workstation:~/CPE232_VALENCIA$ git config --global user.email qajgvalencia@tip.edu.ph
valencia@workstation:~/CPE232_VALENCIA$ cat ~/.gitconfig
[user]
  name = VALENCIA
  email = qajgvalencia@tip.edu.ph
valencia@workstation:~/CPE232_VALENCIA$
```

- 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.



```
valencia@workstation: ~/CPE232_VALENCIA
GNU nano 6.2 README.md
# CPE232_VALENCIA

## Description
1st created for ACT2.1
```

- 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?

```
valencia@workstation:~/CPE232_VALENCIA$ 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")
valencia@workstation:~/CPE232_VALENCIA$
```

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

```
valencia@workstation:~/CPE232_VALENCIA$ nano README.md
valencia@workstation:~/CPE232_VALENCIA$ git add README.md
valencia@workstation:~/CPE232_VALENCIA$
```

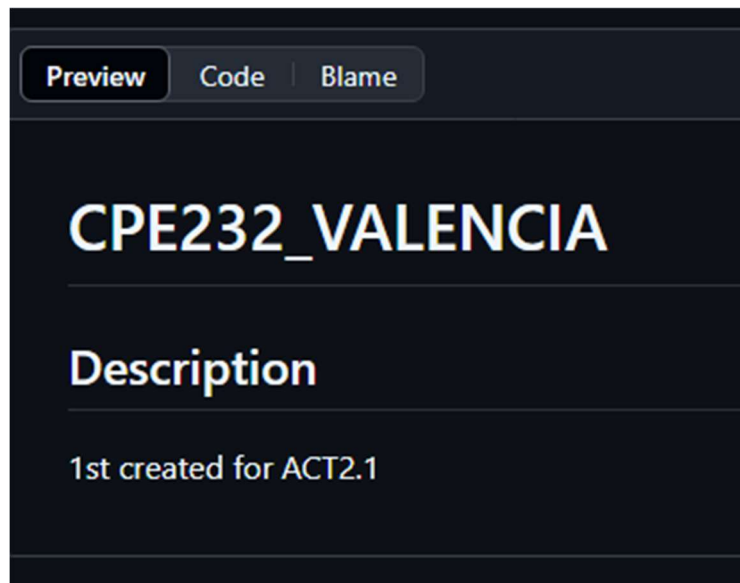
- k. Use the **git commit -m "your message"** to create a snapshot of the staged changes along the timeline of the Git projects history. The use of this command is required to select the changes that will be staged for the next commit.

```
valencia@workstation:~/CPE232_VALENCIA$ git commit -m "1st test"
[main 2aa7a31] 1st test
1 file changed, 3 insertions(+)
```

- 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**.

```
valencia@workstation:~/CPE232_VALENCIA$ git push origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 2 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 305 bytes | 305.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:a-valenc/CPE232_VALENCIA.git
7ec738d..2aa7a31  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?
  - We used git as commit and pushing changes from the repository we made from the github.
4. How important is the inventory file?
  - It will be useful when using and configuring multiple/many devices which needs to have almost the same configuration. This is mostly used for management and flexibility.

**Conclusions/Learnings:**

- We are using ssh to be able to access the server's terminal while using the workstation device. This is useful when doing maintenance and managing changes needed per device. Using git shows that you can store files to the internet or in this case, in Github, for easier access when changing from different system to another system.