

Name: Niemo, Christian Al C.	Date Performed: Sep 2, 2023
Course/Section: CPE 232 - CPE31S6	Date Submitted: Sep 2, 2023
Instructor: Dr. Jonathan Taylor	Semester and SY: 1st sem; 2023-2024
Activity 2: SSH Key-Based Authentication and Setting up Git	
<p>1. Objectives:</p> <ul style="list-style-type: none"> 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 	
<p>Part 1: Discussion</p> <p>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></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>What is ssh-keygen?</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>SSH Keys and Public Key Authentication</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>	
<p>Task 1: Create an SSH Key Pair for User Authentication</p> <ul style="list-style-type: none"> 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*.

```
workstation@workstation:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/workstation/.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:fGfRVZb3a0988e0fMifKbSWvW0JZdb2XXqrZnJtMGjE workstation@workstation
The key's randomart image is:
+---[RSA 3072]---+
|                 X|
|                .+=|
|               .o=|
|              .o+|
|             S . E+.oo|
|            . o.+.+.|
|           .X=B+|
|          . =*@oB|
|         oo+*o=|
+-----[SHA256]-----+
```

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.

```
workstation@workstation:~$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/workstation/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
```

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.

```
workstation@workstation:~$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/workstation/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/workstation/.ssh/id_rsa
Your public key has been saved in /home/workstation/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:H1C1Fbv/GEp7HUs2Nx1HCBBeQ3xKfcE2YAGVyiazWiic workstation@workstation
The key's randomart image is:
+---[RSA 4096]----+
|      .+=0==*o|
|      .o+o*++|
|      .o ..+=o|
|      o..  o+o|
|      oS..  oo.|
|      E o. . . o+|
|      o .  +*+|
|      o=0|
|      ..+|
+----[SHA256]-----+
```

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`.

```
workstation@workstation:~$ ls -la .ssh
total 24
drwx----- 2 workstation workstation 4096 Sep  2 18:21 .
drwxr-x--- 16 workstation workstation 4096 Sep  2 18:20 ..
-rw----- 1 workstation workstation 3389 Sep  2 18:21 id_rsa
-rw-r--r-- 1 workstation workstation 749 Sep  2 18:21 id_rsa.pub
-rw----- 1 workstation workstation 1956 Sep  2 18:15 known_hosts
-rw----- 1 workstation workstation 1120 Sep  2 18:15 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.

```
workstation@server1:~/.ssh$ cat authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQCAQD0ZVkv17m5A+a2ziY6JlwktWApShTehivF0CPT4yp5
M6Noi1nBr9QkosLFA5XJ6BuFeVgVS5Y8DRq380NM1kxG4EXrXGBzvZsRD/tajRqfZA48HpW+JHfPpPis
DPx4r9XzsKTZhA0QZSt1oQqHbeTPJBATQPHg/OsMf1UxDWUGpZ436Tmth60oPsElnOGFmM4toL8JrvZo
ULH/SY0nPKc/MighvNTovSbAqNPWE9cZLaBQUiHTGGQpJcG01KbJC4zrbbzL3YT/sZwtGh4GJ7VstmW7
f4p0GnHwjvg50rmk0i9bbtj52BoUxrVFrqXKiUHZnjEoDXpIHGaWHuciGcw3CvVQs13MeyR+RCpYBCsQ
r3RLE4HtjLrhcfTk8wFZTDe0G3XJtF/Uo0moOyPvn8VHn91cEoWBpw/dUqL7HLLiZ9G8XYCwJ6aKjqFR
QEm3lZwilUe0fN4nwVlvkwC1Pf99fTHtSzYXJApLJ7dDQzoE+a8BU00j/Ka82dDvk2s0YWfpcGB8IEyL
/abexRbjjfIZR52uTb5WwCj+CD8r9z2wNvQM1oFJF/AqKgG+hzanpwBgdhL3dMEf/ClnAICaoGSfsqqc
D8Q7zNx/i7SnfjWKcaqQNckwsYpv7z9BZcZ+oUUG0iJRaUHHenz1oV5COY3IH/8m+m4PKPGCVZuupG/N
Nw== workstation@workstation
```

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

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

Number of key(s) added: 1

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

```
workstation@workstation:~$ ssh-copy-id -i ~/.ssh/id_rsa workstation@server2
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/workstation
/.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 prompt
ed now it is to install the new keys
workstation@server2's password:

Number of key(s) added: 1

Now try logging into the machine, with: "ssh 'workstation@server2'"
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?

```
workstation@workstation:~$ ssh workstation@server1
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 6.2.0-31-generic x86_64)

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

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

Last login: Sat Sep  2 18:15:32 2023 from 192.168.56.105
```

```
workstation@workstation:~$ ssh workstation@server2
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 6.2.0-31-generic x86_64)

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

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

Last login: Sat Sep  2 18:15:45 2023 from 192.168.56.105
```

- No. Because it means that the SSH key is already set up for authentication on Server 1 and Server 2.

Reflections:

Answer the following:

1. How will you describe the ssh-program? What does it do?
 - SSH, which stands for Secure Shell, is a network protocol and a program that allows secure remote access to a computer or server over an unsecured network, such as the internet. SSH is widely used in the field of information technology and system administration for securely managing and interacting with remote systems.
2. How do you know that you already installed the public key to the remote servers?
 - First, you can check your local machine's `known_hosts` file, typically found in the `~/.ssh/` directory, which stores records of known hosts and their public keys. If you find an entry for the remote server's hostname or IP address, it suggests your public key is already stored locally. Next, you can attempt to establish an SSH connection to the remote server. If your public key is installed on the server, and it's configured for public key authentication, you should be able to log in without being prompted for a password. Additionally, review your SSH client configurations in the `~/.ssh/config` file to ensure there are no custom settings that might affect public key authentication. If you're uncertain, contacting the server administrator or responsible person is a good option to confirm whether your public key is added to the server's `authorized_keys`. Lastly, if you have access to the remote server's file system, you can check your user's `authorized_keys` file within the `.ssh` directory on the server to see if your public key is present there. Checking these elements will help you determine the status of your public key on 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*

```
workstation@workstation:~$ sudo apt install git
[sudo] password for workstation:
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 0.1
7029-1 [26.5 kB]
Get:2 http://ph.archive.ubuntu.com/ubuntu jammy-updates/main amd64 git-man all 1
:2.34.1-1ubuntu1.10 [954 kB]
Get:3 http://ph.archive.ubuntu.com/ubuntu jammy-updates/main amd64 git amd64 1:2
.34.1-1ubuntu1.10 [3,166 kB]
```

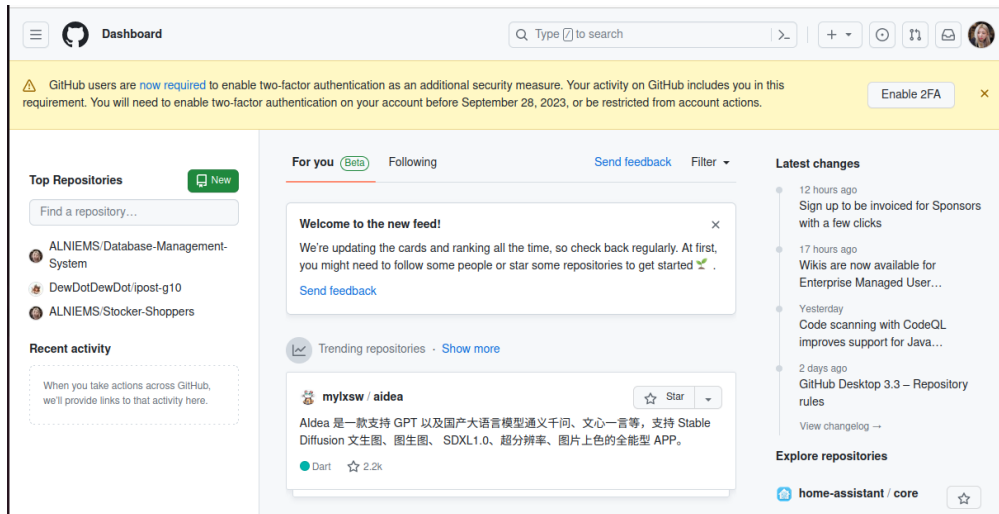
2. After the installation, issue the command *which git* again. The directory of git is usually installed in this location: *user/bin/git*.

```
workstation@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.


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

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.


Owner *  ALNIEMS / Repository name * CPE232/Niemo

⚠ Your new repository will be created as CPE232-Niemo.
The repository name can only contain ASCII letters, digits, and the characters ., -, and _.

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

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', 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.

Add new SSH Key

Title

CPE232

Key type

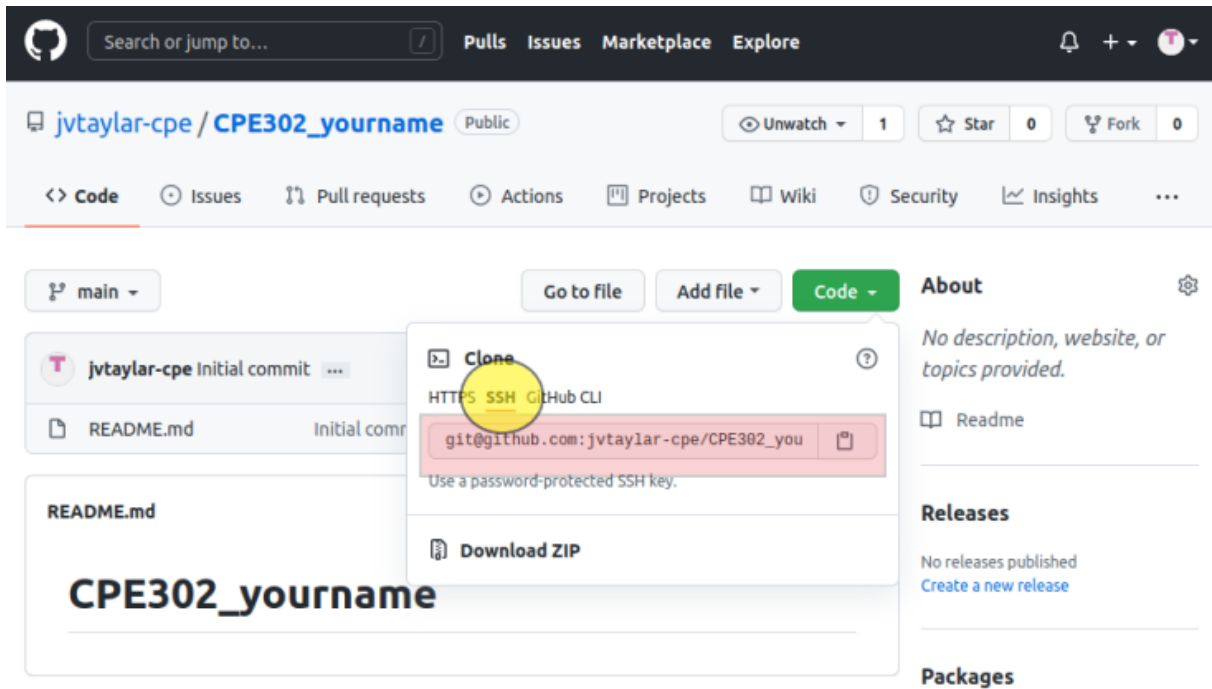
Authentication Key ↕

Key

```
XJ6BufeVgVS5Y8DRq38ONM1kxG4EXrXGBzvZsRD/tajRqfzA48HpW+JHfPpPisDPx4r9XzsKTZhA0QZSt1oQqHbeTPJ
BATQPHg/OsMf1UxDWUGpZ436Tmth60oPsEInOGFmM4toL8JrvZoULH/SY0nPKc
/MighvNTOvSbAqNPwE9cZLaBQUlhTGGQpJCgO1KbJC4zrbbzI3YT
/sZWtGh4GJ7VstmW7f4pOGnHwjvg50rmk0i9bbij52BoUxrVFrqXKiUHZnjEoDXplHGawHuciGcW3CvVQs13MeyR+RCp
YBCsQr3RLE4HijLrhcfk8wFZTDe0G3XJtF/Uo0moOyPvn8VHn91cEoWBpW
/dUqL7HLLIZ9G8XYCwJ6aKjqFRQEm3IZwilUe0fN4nwVlvkwC1Pf99fTHtSzYXJApLJ7dDQzoE+a8BU0Oj
/Ka82dDvk2s0YWfpCGB8IEyL/abexRbjfIZR52uTb5WwCj+CD8r9z2wNvQM1oFJF/AqKgG+hzanpwBgdhL3dMEf
/ClnAlcaoGSfsqqcD8Q7zNx/i7SnfjWKcaqQNckwSYpv7z9BZcZ+oUuGOiJRaUHHemz1oV5COY3IH
/8m+m4PKPGCVZuupG/NNw== workstation@workstation|
```

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.

```
workstation@workstation:~$ git clone git@github.com:ALNIEMS/CPE232-Niemo.git
Cloning into 'CPE232-Niemo'...
The authenticity of host 'github.com (20.205.243.166)' can't be established.
ED25519 key fingerprint is SHA256:+DiY3wvV6TuJJhpZisF/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.
```

- 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.

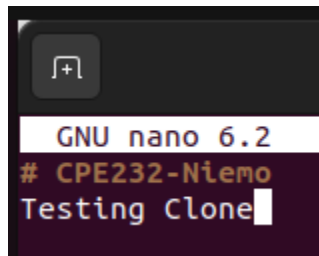
```
workstation@workstation:~$ ls
CPE232-Niemo  Documents  id_rsa      Music      Public  Templates
Desktop      Downloads  id_rsa.pub  Pictures   snap    Videos
workstation@workstation:~$ cd CPE232-Niemo
workstation@workstation:~/CPE232-Niemo$ ls
README.md
```

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`

```
workstation@workstation:~$ git config --global user.name "Niemo"
workstation@workstation:~$ git config --global user.email christianniemo@gmail.com
workstation@workstation:~$ cat ~/.gitconfig
[user]
  name = Niemo
  email = christianniemo@gmail.com
```

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.



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?

```
workstation@workstation:~/CPE232-Niemo$ 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
```

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

```
workstation@workstation:~/CPE232-Niemo$ git add README.md
```

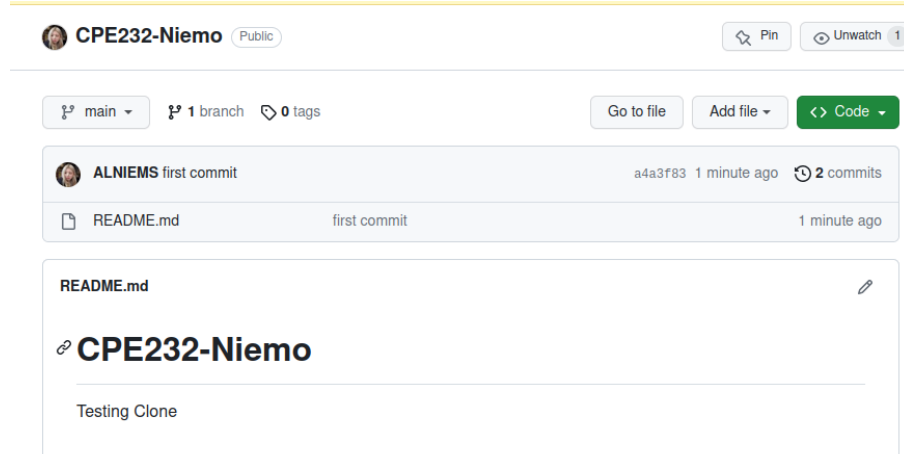
k. Use the `git commit -m "your message"` to create a snapshot of the staged changes along the timeline of the Git projects history. The use of this command is required to select the changes that will be staged for the next commit.

```
workstation@workstation:~/CPE232-Niemo$ git commit -m "first commit"
[main a4a3f83] first commit
1 file changed, 2 insertions(+), 1 deletion(-)
```

- 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`.

```
workstation@workstation:~/CPE232-Niemo$ git push origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Writing objects: 100% (3/3), 270 bytes | 270.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:ALNIEMS/CPE232-Niemo.git
d2f19a4..a4a3f83  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 successfully generated keys that will generate the connection to another machine, and we managed to set up git using the linux terminal.

4. How important is the inventory file?

- In the context of Linux, it's important to clarify that the concept of an inventory file is not an inherent aspect of the Linux operating system itself. Instead, the significance of an inventory file primarily arises within specific Linux-related tools and systems, such as Ansible. Within Ansible, the inventory file serves a crucial role as it allows users to specify target hosts, group them logically, and define how Ansible should connect to and manage these hosts. This inventory file is essential for orchestrating tasks and configurations across multiple Linux servers and other systems.

Conclusions/Learnings:

In summary, the objectives outlined are focused on enhancing security and efficiency in remote machine management, particularly in a Git-based workflow. The primary goals include configuring SSH key-based authentication for improved security, creating key pairs, ensuring connectivity between local and remote machines, establishing Git repositories for version control, and enabling the execution of ad hoc commands from the local machine to remote servers.

By achieving these objectives, users can strengthen the authentication process by replacing passwords with SSH keys, enhancing security. Additionally, setting up Git repositories streamlines collaborative software development and version control. Finally, the ability to configure and execute ad hoc commands simplifies server management tasks, offering greater flexibility and control in the Linux environment. These objectives collectively contribute to a more secure, efficient, and productive workflow for remote machine management and software development.