Name: Khelvin P. Nicolas	Date Performed: 9/4/2024
Course/Section: CPE 212 - CPE31S2	Date Submitted: 9/4/2024
Instructor: Engr. Robin Valenzuela	Semester and SY: 1st Sem - 3rd Year

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.
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.

Task 1 Documentation

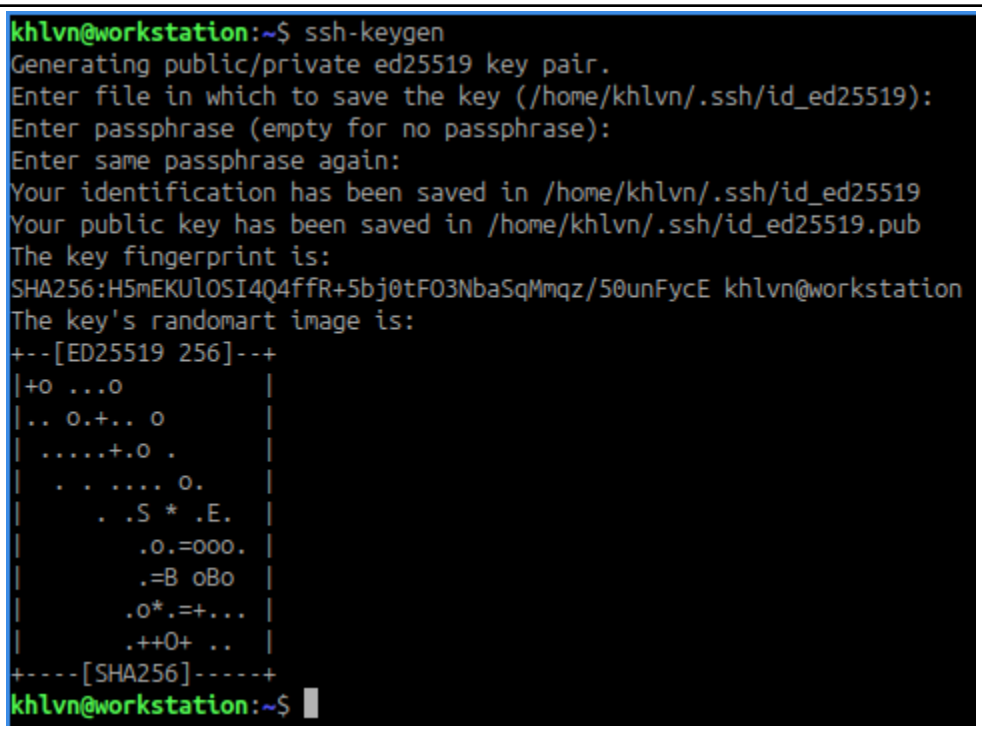


figure 1.1: running ssh-keygen command without arguments

```
khlvn@workstation:~$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/khlvn/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/khlvn/.ssh/id_rsa
Your public key has been saved in /home/khlvn/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:rNl0ZupjrnAAxQcFiFpGeWIkDU67FRzk7iimi4gwWVs khlvn@workstation
The key's randomart image is:
+---[RSA 4096]----+
|oB=B=+.          |
|+.@o+ .          |
|.*.= .           |
|. +.   .         |
| ...E   S +      |
| ooo . = =       |
|=o... + o        |
|O.   o .o        |
|*.    .+o.       |
+----[SHA256]-----+
khlvn@workstation:~$
```

figure 1.2: issuing ssh-keygen -t rsa -b 4096 command

```
khlvn@workstation:~$ ls -la .ssh
total 32
drwx------  2 khlvn khlvn 4096 Set  4 08:51 .
drwxr-x--- 14 khlvn khlvn 4096 Ago 25 17:58 ..
-rw-------  1 khlvn khlvn  411 Set  4 08:49 id_ed25519
-rw-r--r--  1 khlvn khlvn   99 Set  4 08:49 id_ed25519.pub
-rw-------  1 khlvn khlvn 3381 Set  4 08:51 id_rsa
-rw-r--r--  1 khlvn khlvn  743 Set  4 08:51 id_rsa.pub
-rw-------  1 khlvn khlvn 2240 Ago 25 18:15 known_hosts
-rw-------  1 khlvn khlvn 1120 Ago 25 18:03 known_hosts.old
khlvn@workstation:~$
```

figure 1.3: verifying the created ssh-key

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*
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?

Task 2 Documentation

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

Number of key(s) added: 1

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

khlvn@workstation:~$ ssh khlvn@server1
Welcome to Ubuntu 24.04 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
Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your Internet connection or proxy settings

Last login: Wed Sep  4 09:08:39 2024 from 192.168.56.10
khlvn@server1:~$
```

```
khlvn@workstation:~$ ssh-copy-id -i ~/.ssh/id_rsa khlvn@server2
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/khlvn/.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
khlvn@server2's password:

Number of key(s) added: 1

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

khlvn@workstation:~$ ssh khlvn@server2
Welcome to Ubuntu 24.04 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
Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your Internet connection or proxy settings

Last login: Sun Aug 25 18:15:53 2024 from 192.168.56.10
khlvn@server2:~$
```

figure 2.1: copying the public key to remote servers

Reflections:

Answer the following:

1. **How will you describe the ssh-program? What does it do?**
   - It is a software that initiates connection to a remote server, usually comes with a GUI or through a command line interface.
2. **How do you know that you already installed the public key to the remote servers?**
   - By logging in the remote server and locating the authorized_keys file in ~/.ssh/authorized_keys directory, and checking the public key inside by using nano or vim.

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

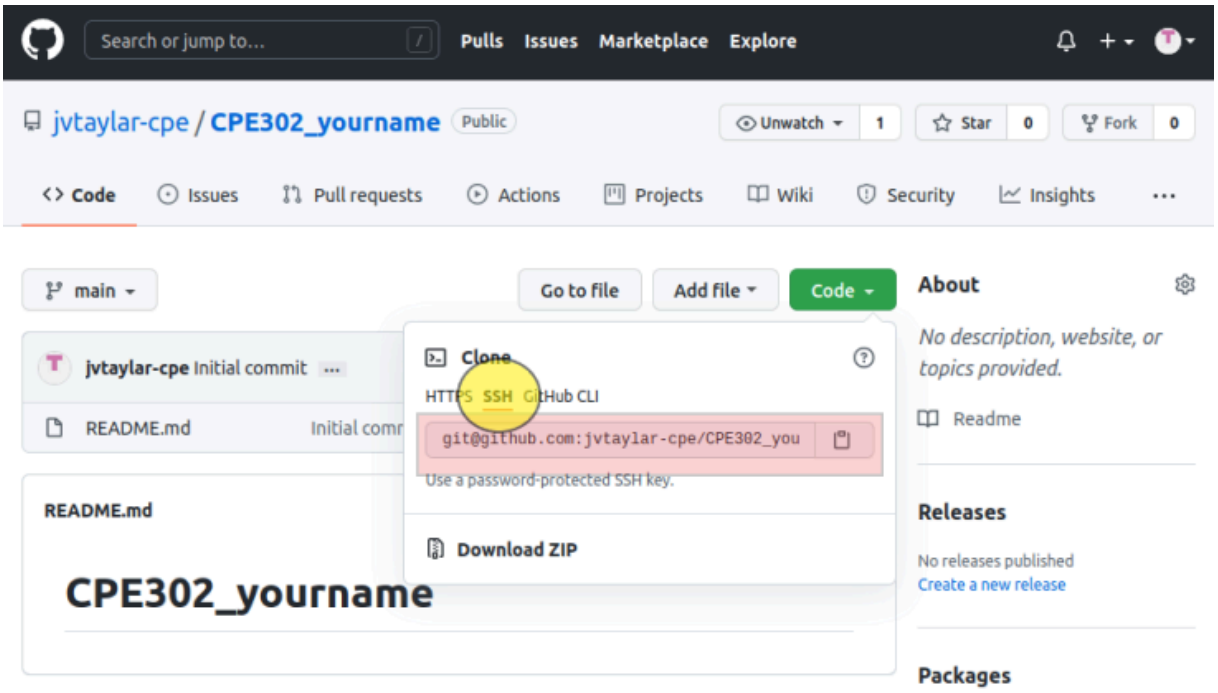
At the heart of GitHub is an open-source version control system (VCS) called Git. Git is responsible for everything GitHub-related that happens locally on your computer. To use Git on the command line, you'll need to download, install, and configure Git on your computer. You can also install GitHub CLI 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*
2. After the installation, issue the command *which git* again. The directory of git is usually installed in this location: *user/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.
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.
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.
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*
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?
j. Use the command *git add README.md* to add the file into the staging area.
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.
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*.
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.

Task 3 Documentation

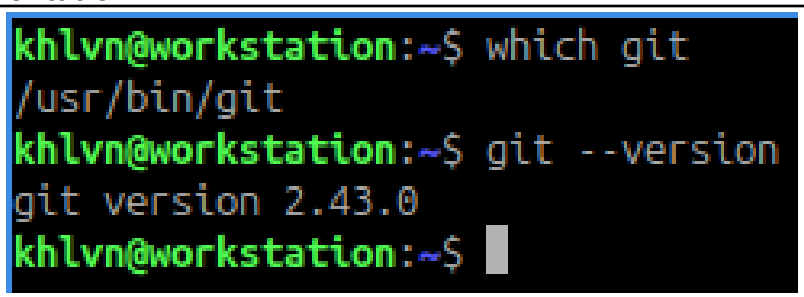


figure 3.1: verify git version

figure 3.2: creating new repository

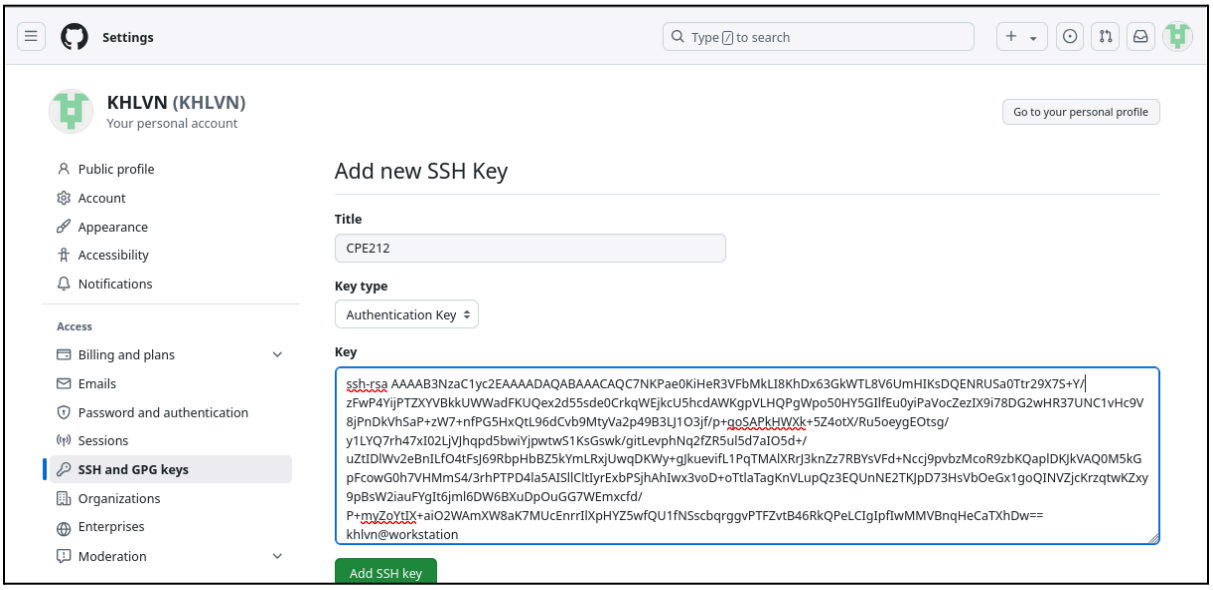


figure 3.3: setting SSH key

```
khlvn@workstation:~$ git clone git@github.com:KHLVN/CPE212_KHELVIN.git
Cloning into 'CPE212_KHELVIN'...
The authenticity of host 'github.com (20.205.243.166)' can't be established.
ED25519 key fingerprint is SHA256:+DiY3wvvV6TuJJhbpZisF/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.
khlvn@workstation:~$ ls
CPE212_KHELVIN  Desktop  Documents  Downloads  Music  Pictures  Public  snap
khlvn@workstation:~$
```

```
khlvn@workstation:~$ cd CPE212_*
khlvn@workstation:~/CPE212_KHELVIN$ ls
README.md
khlvn@workstation:~/CPE212_KHELVIN$
```

figure 3.4: cloning the created repository

```
khlvn@workstation:~/CPE212_KHELVIN$ git config --global user.name "Khelvin"
khlvn@workstation:~/CPE212_KHELVIN$ git config --global user.email qkpnicolas@tip.edu.ph
khlvn@workstation:~/CPE212_KHELVIN$ cat ~/.gitconfig
[user]
        name = Khelvin
        email = qkpnicolas@tip.edu.ph
khlvn@workstation:~/CPE212_KHELVIN$
```

figure 3.5: personalizing the git config file

```
khlvn@workstation:~/CPE212_KHELVIN$ nano README.md
khlvn@workstation:~/CPE212_KHELVIN$ more README.md
# CPE212_KHELVIN

Repository for academic purposes | CPE212 - CPE31S2
khlvn@workstation:~/CPE212_KHELVIN$ 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")
khlvn@workstation:~/CPE212_KHELVIN$ git add README.md
khlvn@workstation:~/CPE212_KHELVIN$ git commit -m "Updated README.md"
[main 8078c92] Updated README.md
 1 file changed, 3 insertions(+), 1 deletion(-)
khlvn@workstation:~/CPE212_KHELVIN$ git push origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 313 bytes | 313.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:KHLVN/CPE212_KHELVIN.git
   c186858..8078c92  main -> main
khlvn@workstation:~/CPE212_KHELVIN$
```

figure 3.6: editing the README.md using nano and using various git commands to push the edited README.md file into the repository.

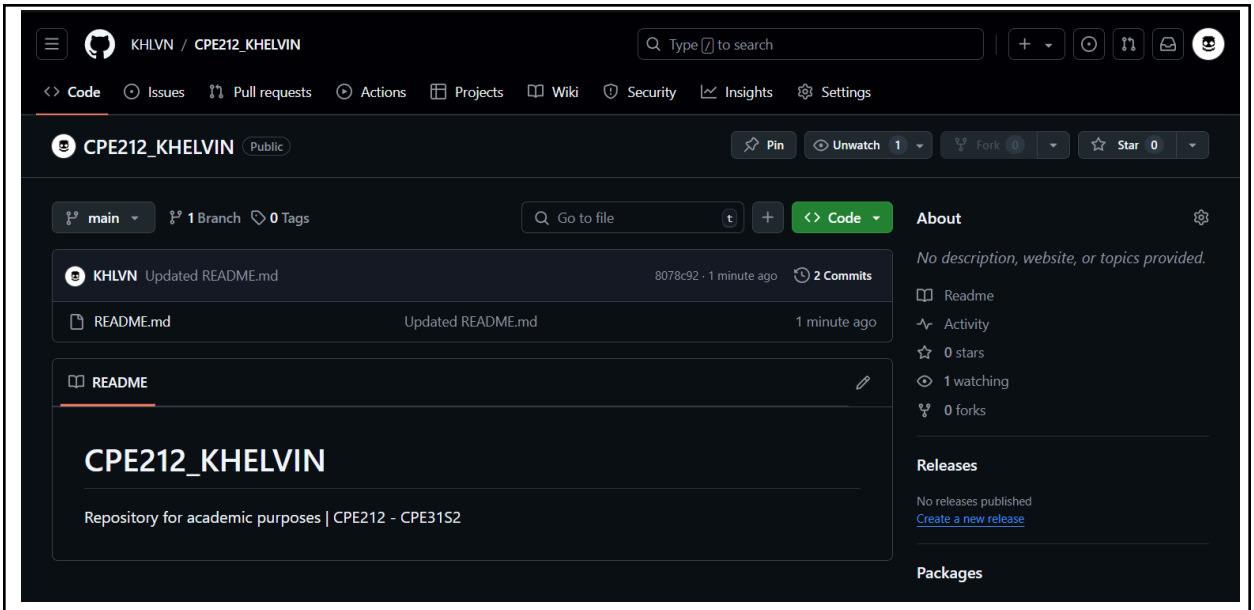


figure 3.7: verifying changes and the last commit to the repository

Reflections:

Answer the following:

3. What sort of things have we so far done to the remote servers using ansible commands?
   - Not yet discussed
4. How important is the inventory file?
   - Not yet discussed

Conclusions/Learnings:

In this activity, we have learned how to use GitHub and set up a repository. We have also used several git commands such as add, push, and commit. Git is a version control system that is used to track changes in files and coordinate work among multiple developers on a project. It allows users to manage code, track revisions, and revert to previous versions whenever needed.