Name: Hazel Aillson T. Manuel	**Date Performed:** Aug. 15, 2025
Course/Section: CPE31S4	**Date Submitted:** Aug. 15, 2025
Instructor: Engr. Robin Valenzuela	**Semester and SY:** 1st Sem 25-26

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.

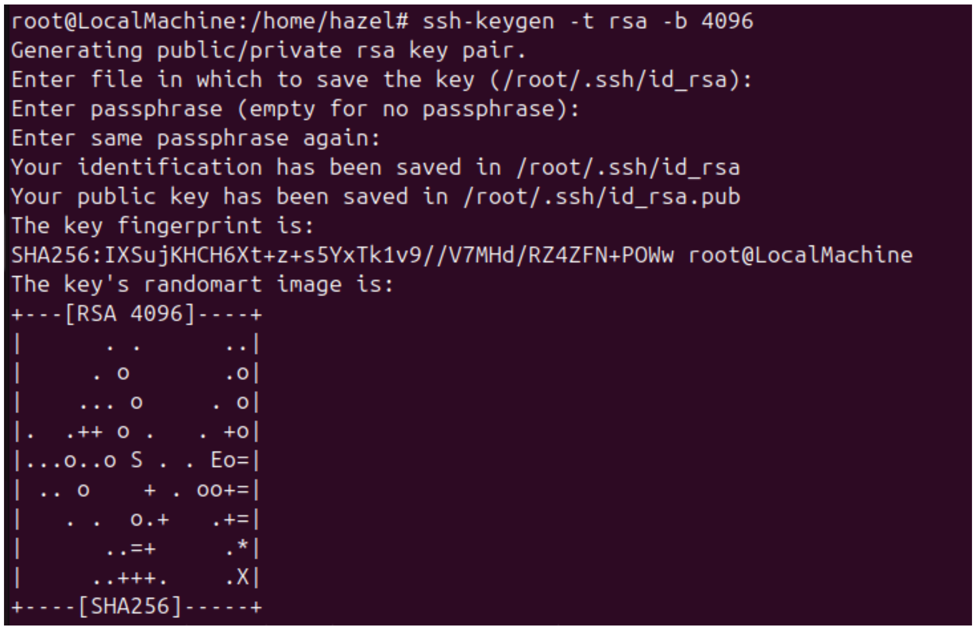
However, SSH keys are authentication credentials just like passwords. Thus, they must be managed somewhat analogously to usernames and passwords. They should have a proper termination process so that keys are removed when no longer needed.

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,

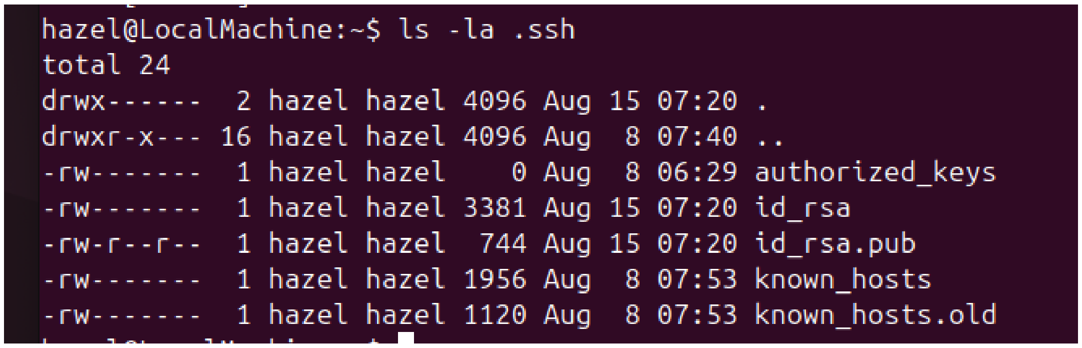
the tool asked where to save the file. SSH keys for user authentication are usually stored in the users .ssh directory under the home directory. However, in enterprise environments, the location is often different. The default key file name depends on the algorithm, in this case *id_rsa* when using the 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.



```
root@LocalMachine:/home/hazel# ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa
Your public key has been saved in /root/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:IXSujKHCH6Xt+z+s5YxTk1v9//V7MHd/RZ4ZFN+POWw root@LocalMachine
The key's randomart image is:
+---[RSA 4096]----+
|      . .      ..|
|     . o       .o|
|    ... o     . o|
|.  .++ o .   . +o|
|...o..o S . . Eo=|
| .. o    + . oo+=|
|   . .  o.+   .+=|
|      ..=+     .*|
|     ..+++.    .X|
+----[SHA256]-----+
```

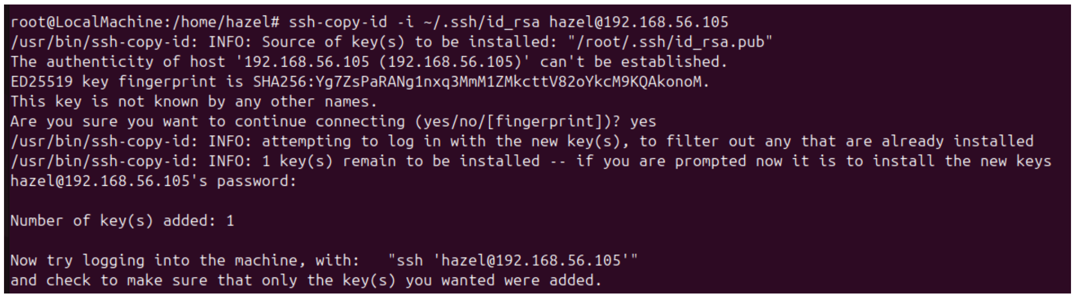
4. Verify that you have created the key by issuing the command *ls -la .ssh.* The command should show the .ssh directory containing a pair of keys. For example, id_rsa.pub and id_rsa.



```
hazel@LocalMachine:~$ ls -la .ssh
total 24
drwx------  2 hazel hazel 4096 Aug 15 07:20 .
drwxr-x--- 16 hazel hazel 4096 Aug  8 07:40 ..
-rw-------  1 hazel hazel    0 Aug  8 06:29 authorized_keys
-rw-------  1 hazel hazel 3381 Aug 15 07:20 id_rsa
-rw-r--r--  1 hazel hazel  744 Aug 15 07:20 id_rsa.pub
-rw-------  1 hazel hazel 1956 Aug  8 07:53 known_hosts
-rw-------  1 hazel hazel 1120 Aug  8 07:53 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*
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?

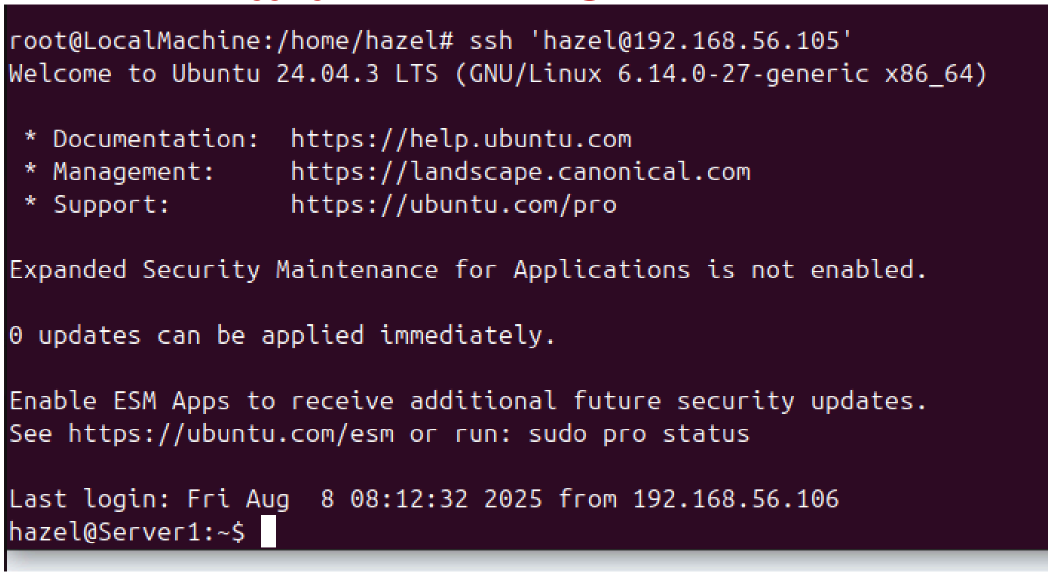


```
root@LocalMachine:/home/hazel# ssh-copy-id -i ~/.ssh/id_rsa hazel@192.168.56.105
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/root/.ssh/id_rsa.pub"
The authenticity of host '192.168.56.105 (192.168.56.105)' can't be established.
ED25519 key fingerprint is SHA256:Yg7ZsPaRANg1nxq3MmM1ZMkcttV82oYkcM9KQAkonoM.
This key is not known by any other names.
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 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
hazel@192.168.56.105's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'hazel@192.168.56.105'"
and check to make sure that only the key(s) you wanted were added.
```

logging in with ssh hazel@192.168.56.105



```
root@LocalMachine:/home/hazel# ssh 'hazel@192.168.56.105'
Welcome to Ubuntu 24.04.3 LTS (GNU/Linux 6.14.0-27-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

Last login: Fri Aug  8 08:12:32 2025 from 192.168.56.106
hazel@Server1:~$
```

What I noticed is that it is basically an exchange of ssh keys between the local machine and server 1. The connection did ask for the password of the remote hose. What is basically happening is that it copied the public key to the authorized_keys file of the remote host. By doing so, we are able to log in to the remote host without the need to input its password

Reflections:
Answer the following:

1. How will you describe the ssh-program? What does it do?
   SSH allows us to access and manage different computers over an unsecured network. With the help of ssh, it basically allows us to have a secure communication between the local machine and a remote host. With this, we can easily connect to servers and modify within it with the terminal of a local machine.
2. How do you know that you already installed the public key to the remote servers?

   We can determine if a public key is installed by using the command ls -la .ssh. This shows the list of .ssh directory which should include the pair of keys.

Part 2: Discussion

Provide screenshots for each task.

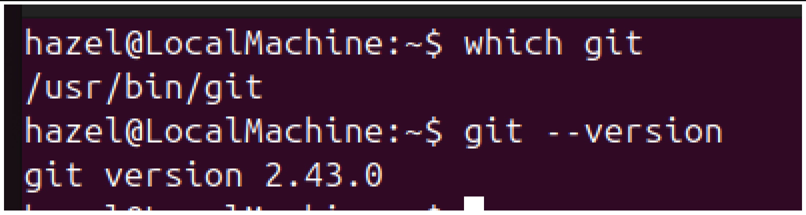
It is assumed that you are done with the last activity (**Activity 2: SSH Key-Based Authentication**).

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

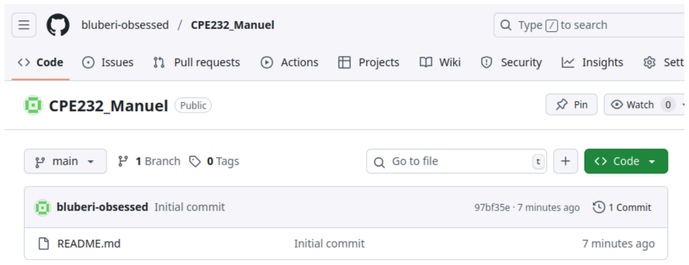
Task 3: Set up the Git Repository

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.



```
hazel@LocalMachine:~$ which git
/usr/bin/git
hazel@LocalMachine:~$ git --version
git version 2.43.0
```

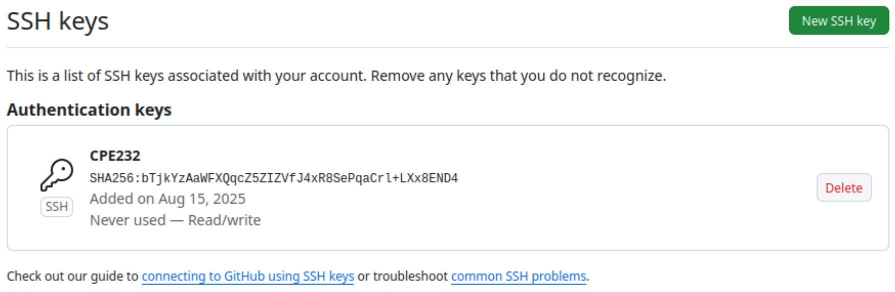
4. Using the browser in the local machine, go to www.github.com.
5. Sign up in case you don't have an account yet. Otherwise, login to your GitHub account.
    a. Create a new repository and name it as CPE232_yourname. Check Add a README file and click Create repository.



bluberi-obsessed Initial commit 97bf35e · 7 minutes ago 1 Commit
README.md Initial commit 7 minutes ago

    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.



SSH keys

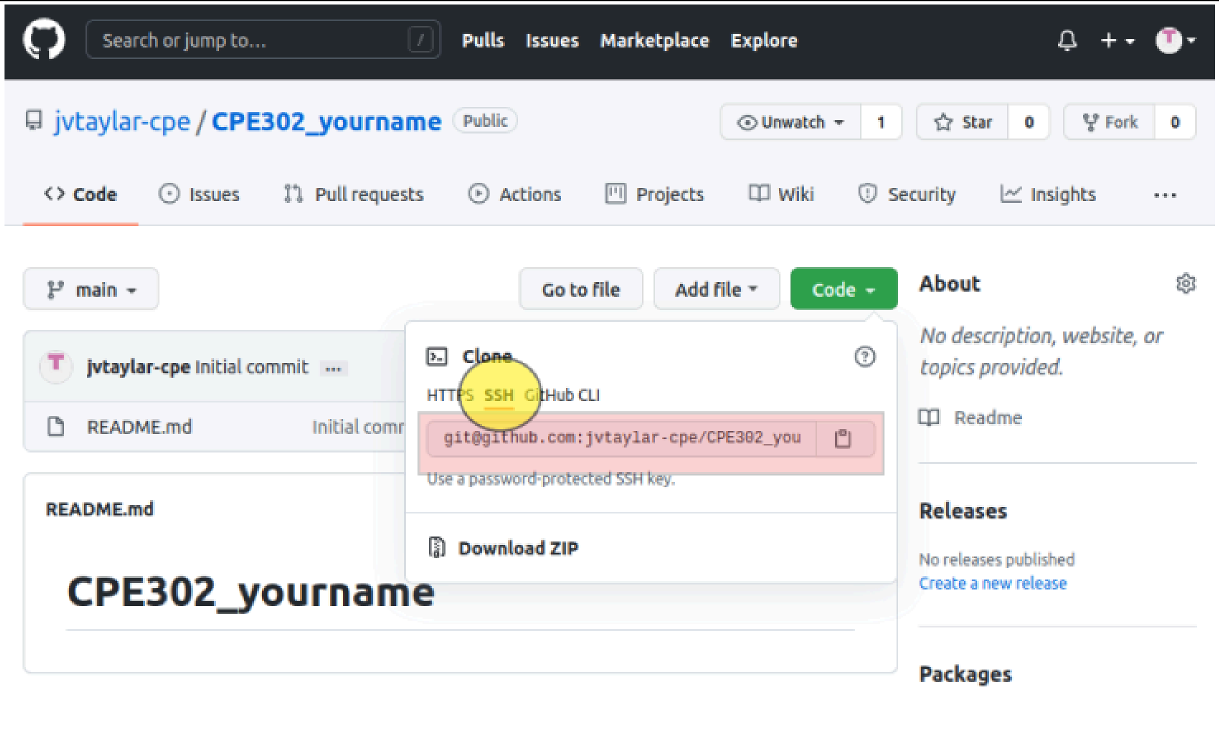
This is a list of SSH keys associated with your account. Remove any keys that you do not recognize.

Authentication keys

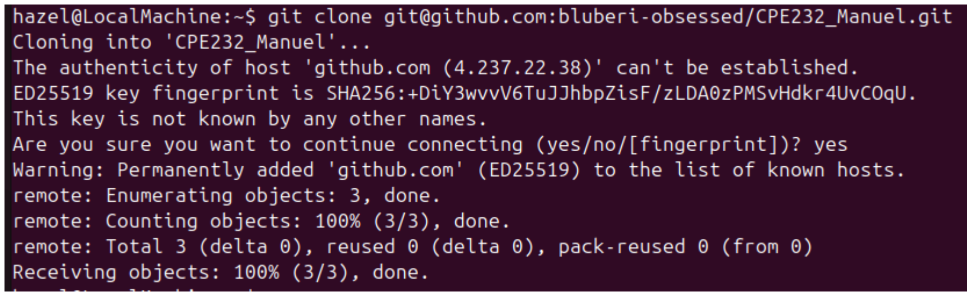
CPE232
SHA256:bTjkYzAaWFXQqcZ5ZIZVfJ4xR8SePqaCrl+LXx8END4
Added on Aug 15, 2025
Never used — Read/write

Check out our guide to connecting to GitHub using SSH keys or troubleshoot common SSH problems.

    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.

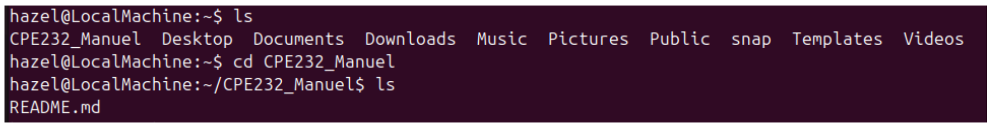


e. Issue the command git clone followed by the copied link. For example, git clone git@github.com:jvtaylar-cpe/CPE232_yourname.git. When prompted to continue connecting, type yes and press enter.



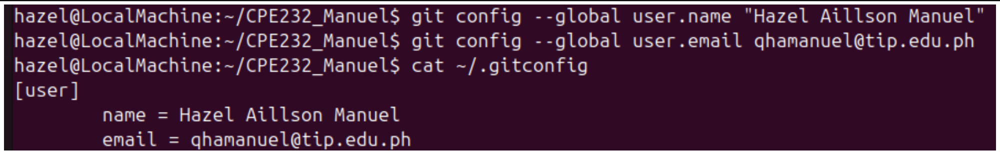
```
hazel@LocalMachine:~$ git clone git@github.com:bluberi-obsessed/CPE232_Manuel.git
Cloning into 'CPE232_Manuel'...
The authenticity of host 'github.com (4.237.22.38)' can't be established.
ED25519 key fingerprint is SHA256:+DiY3wvvV6TuJJhbpZisF/zLDA0zPMSvHdkr4UvCOqU.
This key is not known by any other 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.
```

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.



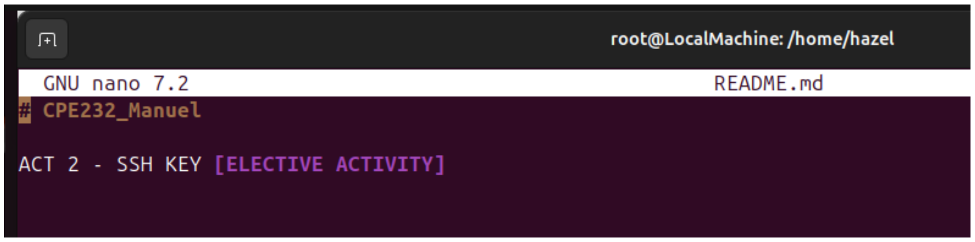
```
hazel@LocalMachine:~$ ls
CPE232_Manuel  Desktop  Documents  Downloads  Music  Pictures  Public  snap  Templates  Videos
hazel@LocalMachine:~$ cd CPE232_Manuel
hazel@LocalMachine:~/CPE232_Manuel$ ls
README.md
```

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*



```
hazel@LocalMachine:~/CPE232_Manuel$ git config --global user.name "Hazel Aillson Manuel"
hazel@LocalMachine:~/CPE232_Manuel$ git config --global user.email qhamanuel@tip.edu.ph
hazel@LocalMachine:~/CPE232_Manuel$ cat ~/.gitconfig
[user]
        name = Hazel Aillson Manuel
        email = qhamanuel@tip.edu.ph
```

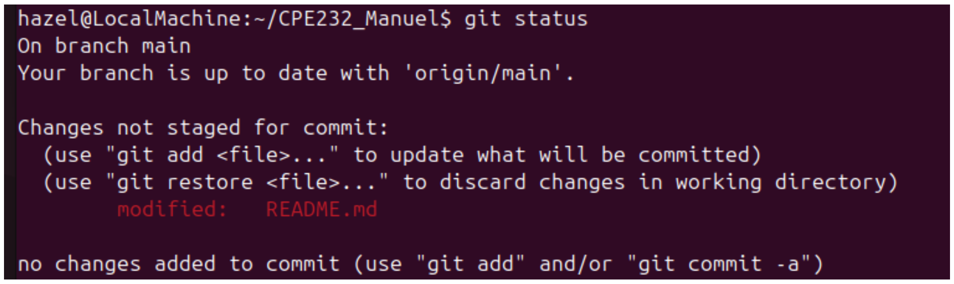
h. Edit the README.md file using nano command. Provide any information on the markdown file pertaining to the repository you created. Make sure to write out or save the file and exit.



```
root@LocalMachine: /home/hazel
GNU nano 7.2                                         README.md
# CPE232_Manuel

ACT 2 - SSH KEY [ELECTIVE ACTIVITY]
```

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?



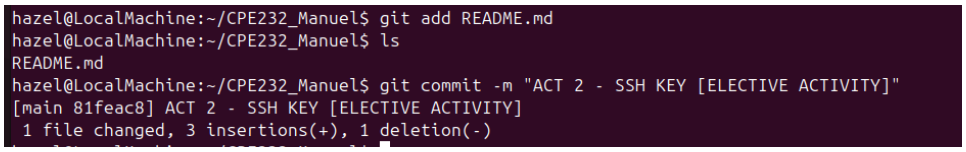
```
hazel@LocalMachine:~/CPE232_Manuel$ git status
On branch main
Your branch is up to date with 'origin/main'.

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")
```

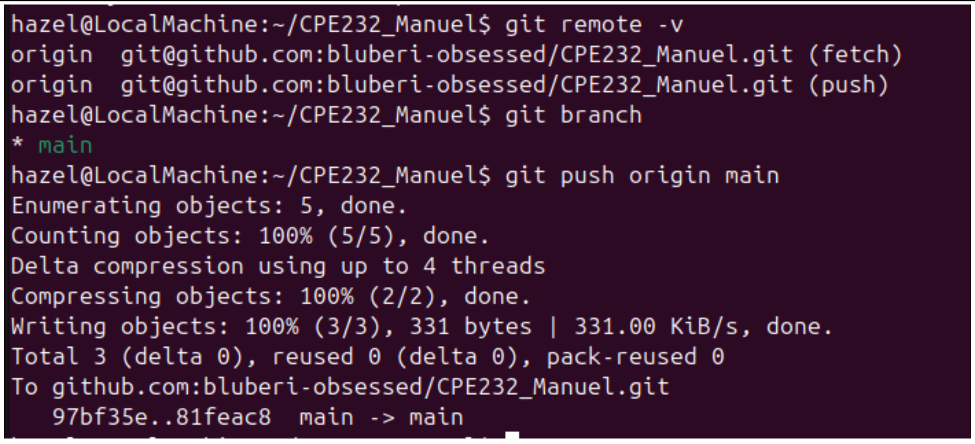
j. Use the command *git add README.md* to add the file into the staging area.

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.



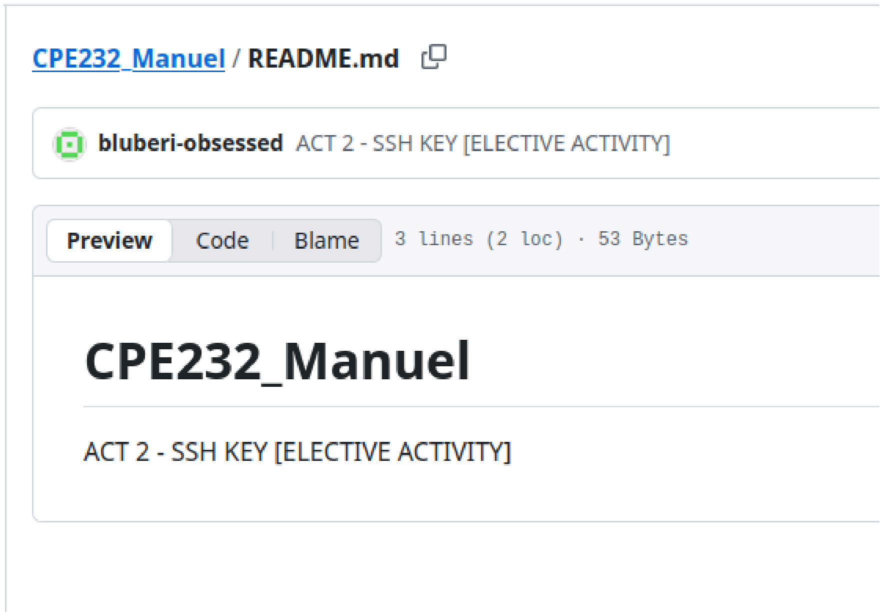
```
hazel@LocalMachine:~/CPE232_Manuel$ git add README.md
hazel@LocalMachine:~/CPE232_Manuel$ ls
README.md
hazel@LocalMachine:~/CPE232_Manuel$ git commit -m "ACT 2 - SSH KEY [ELECTIVE ACTIVITY]"
[main 81feac8] ACT 2 - SSH KEY [ELECTIVE ACTIVITY]
 1 file changed, 3 insertions(+), 1 deletion(-)
```

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.*



```
hazel@LocalMachine:~/CPE232_Manuel$ git remote -v
origin  git@github.com:bluberi-obsessed/CPE232_Manuel.git (fetch)
origin  git@github.com:bluberi-obsessed/CPE232_Manuel.git (push)
hazel@LocalMachine:~/CPE232_Manuel$ git branch
* main
hazel@LocalMachine:~/CPE232_Manuel$ 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), 331 bytes | 331.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:bluberi-obsessed/CPE232_Manuel.git
   97bf35e..81feac8  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.



CPE232_Manuel / README.md

bluberi-obsessed ACT 2 - SSH KEY [ELECTIVE ACTIVITY]

3 lines (2 loc) · 53 Bytes

CPE232_Manuel

ACT 2 - SSH KEY [ELECTIVE ACTIVITY]

Reflections:
Answer the following:

3. What sort of things have we so far done to the remote servers using ansible commands?
   The ansible commands that we have used to configure the remote servers were mainly installing a package and managing services. Another thing is that we are able to set up the users of the servers and configure/edit some

files or directories. Finally, the ssh key creation, cloning repositories and many more.

4. How important is the inventory file?

   Inventory File is crucial this allows us to create and manage many hosts in a single command. The inventory file also helps us to utilize the ansible command efficiently as it lessens the need of specific line options. Overall, it help us simplify executing ansible commands

Conclusions/Learnings:

In this activity, we recalled the creation of ssh key pairs which are the public key and private key. The key moment here is that we are able to log in to a remote host with this ssh key. Furthermore, it also allows us to access our github and modify it. Overall, this has been an insightful activity and learned how we can create or modify files in github. I also just realized that this is another way we can save our scripts and share it to the public.