

Name: Victor B. Ortega	Date Performed: 08/27/23
Course/Section: CPE31S5	Date Submitted: 08/28/23
Instructor: Engr. Roman Richard	Semester and SY: 2023-2024
Activity 2: SSH Key-Based Authentication and Setting up Git	
1. Objectives: <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 	
Part 1: Discussion <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>	
Task 1: Create an SSH Key Pair for User Authentication <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`.

```
victor@workstation:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/victor/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/victor/.ssh/id_rsa
Your public key has been saved in /home/victor/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:PV06uj7JGNfA3fafR9SyCUBewb/tEYLjn1PaopjuCNC victor@workstation
The key's randomart image is:
+---[RSA 3072]-----+
|      ...O.      |
|      ...      |
|    . .OO. .    |
|    o..=+.o.o   |
|    So*..o.B.   |
|    . o . =  =oo |
|    . * E . =+o  |
|    + =.o  * o+  |
|    o*B .. o .   |
+-----[SHA256]-----+
victor@workstation:~$
```

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.

```
victor@workstation:~$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/victor/.ssh/id_rsa): /home/victor/.ssh/id_dsa
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/victor/.ssh/id_dsa
Your public key has been saved in /home/victor/.ssh/id_dsa.pub
The key fingerprint is:
SHA256:z4zJNHPoEAd4HUowrGH2Xkaxho6KB94e0aLfIjivN5M victor@workstation
The key's randomart image is:
+---[RSA 4096]-----+
|  .oo+O..      |
|  + o+O+.      |
|  o +.O= .      |
|  .oo.OO .      |
|  .+.+. S .      |
|  ooo +  = X      |
|  oo.+  = +      |
|  .oE +          |
|  +=.* .          |
+-----[SHA256]-----+
victor@workstation:~$
```

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.

```
victor@workstation:~$ ls -la .ssh
total 24
drwx----- 2 victor victor 4096 Aug 27 10:58 .
drwxr-x--- 15 victor victor 4096 Aug 27 10:22 ..
-rw----- 1 victor victor 3381 Aug 27 11:00 id_rsa
-rw-r--r-- 1 victor victor 744 Aug 27 11:00 id_rsa.pub
-rw----- 1 victor victor 2240 Aug 22 22:54 known_hosts
-rw----- 1 victor victor 1120 Aug 22 22:45 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.

```
victor@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 remote only allows sftp
-h|-?: print this help
```

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

```
victor@workstation:~$ ssh-copy-id -i ~/.ssh/id_rsa victor@workstation
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/victor/.ssh/id_rsa.pub"
The authenticity of host 'workstation (10.0.2.15)' can't be established.
ED25519 key fingerprint is SHA256:2LLwMPj6kFvLYl8ilNxRns2r/aQqIfXdHOPAAuqTQ+w.
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
victor@workstation's password:

Number of key(s) added: 1

Now try logging into the machine, with: "ssh 'victor@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.

Live Server 1:

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

Number of key(s) added: 1

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

Live Server 2:

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

Number of key(s) added: 1

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

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?

Live Server 1:

```
victor@workstation:~$ ssh ortega@server1
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 5.15.0-79-generic x86_64)

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

System information as of Sun Aug 27 03:30:32 AM UTC 2023

System load:  0.0          Processes:           113
Usage of /:   44.6% of 11.21GB Users logged in:        1
Memory usage: 5%          IPv4 address for enp0s3: 192.168.56.103
Swap usage:   0%

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: Sun Aug 27 03:22:39 2023
```

Live Server 2:

```
Connection to server2 closed.
victor@workstation:~$ ssh ortega@server2
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 5.15.0-79-generic x86_64)

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

System information as of Sun Aug 27 03:31:59 AM UTC 2023

System load:  0.0224609375    Processes:           115
Usage of /:   44.8% of 11.21GB Users logged in:        1
Memory usage: 6%            IPv4 address for enp0s3: 192.168.56.102
Swap usage:   0%

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: Sun Aug 27 03:22:48 2023
ortega@server2:~$
```

The host didn't ask for the password; because of "ssh-copy-id -i ~/.ssh/id_rsa user@host," the key became a public key.

Reflections:

Answer the following:

1. How will you describe the ssh-program? What does it do?

-A network protocol and tool called SSH (Secure Shell) offers a safe means to access and control distant computers via an insecure network, such as the internet. It creates a secure connection that is encrypted and authenticated, enabling users to safely log in, issue instructions to distant computers, and transfer data. SSH guarantees the privacy, consistency, and validity of any data sent between local and distant workstations.

2. How do you know that you already installed the public key to the remote servers?

If you type the command "ssh-copy-id -i ~/.ssh/id_rsa user@host," a log detailing the installation of the public key will appear.

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*

```

ortega@server2:~$ sudo apt install git
[sudo] password for ortega:
Sorry, try again.
[sudo] password for ortega:
Sorry, try again.
[sudo] password for ortega:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
git is already the newest version (1:2.34.1-1ubuntu1.10).
git set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
ortega@server2:~$

```

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

```

ortega@server2:~$ which git
/usr/bin/git

```

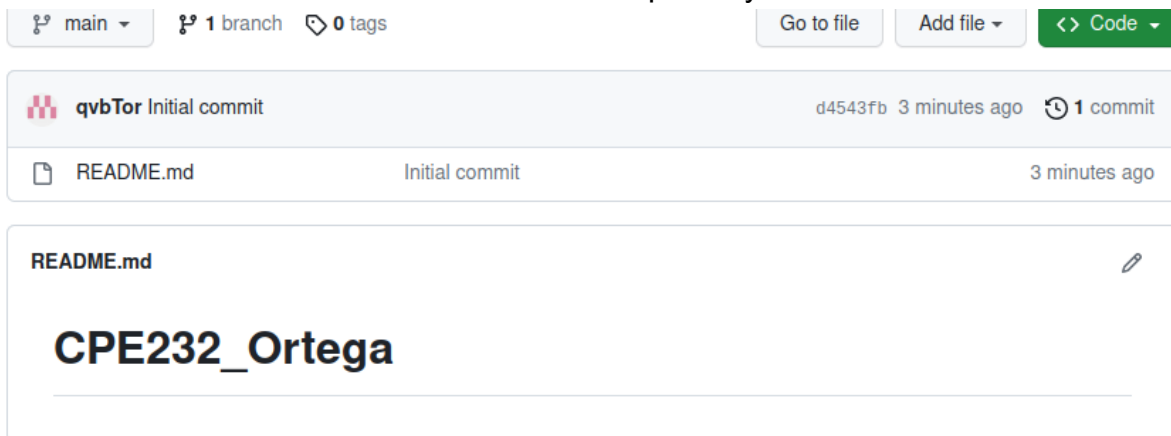
3. The version of git installed in your device is the latest. Try issuing the command *git --version* to know the version installed.

```

ortega@server2:~$ git --version
git version 2.34.1

```

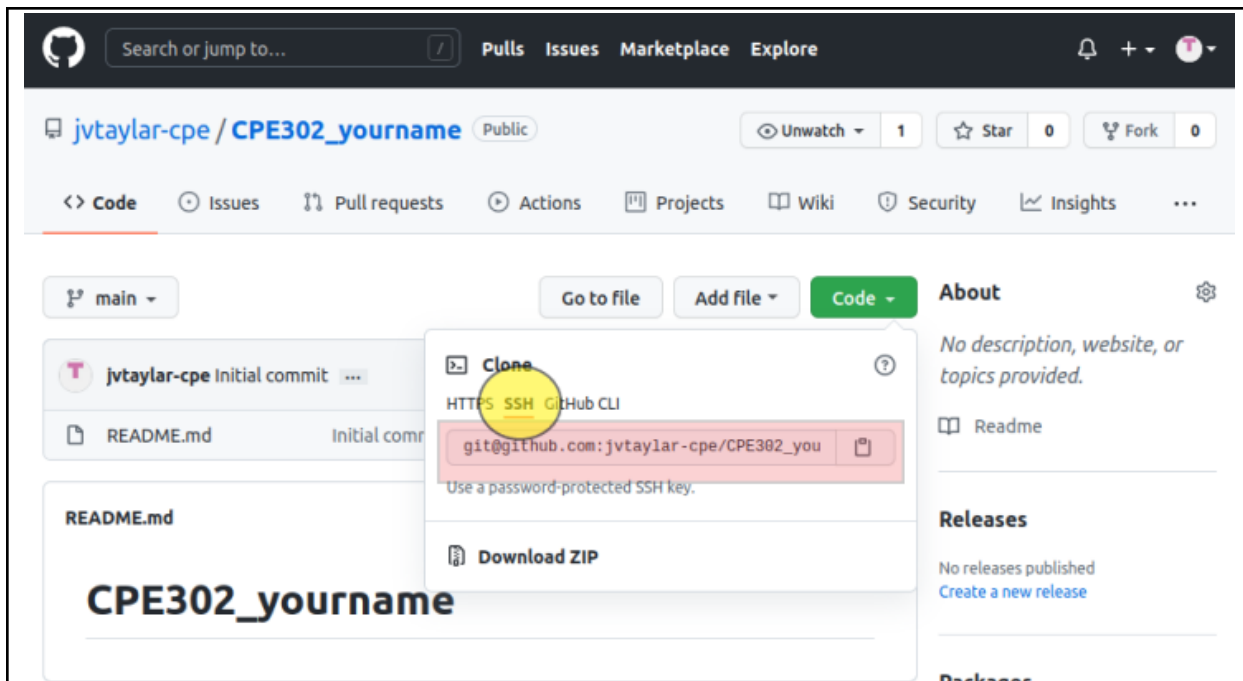
4. Using the browser in the local machine, go to www.github.com.
5. Sign up in case you don't have an account yet. Otherwise, login to your GitHub account.
 - a. Create a new repository and name it as CPE232_yourname. Check Add a README file and click Create repository.



- 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.

```
victor@workstation:~$ git clone git@github.com:qvbTor/CPE232_Ortega.git
Cloning into 'CPE232_Ortega'...
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
Receiving objects: 100% (3/3), done.
victor@workstation:~$
```

- 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`.

```
victor@workstation:~$ ls
CPE232_Ortega Desktop Documents Downloads id_rsa id_rsa.pub Music Pictures Public snap Templates Videos
victor@workstation:~$ cd CPE232_Ortega
victor@workstation:~/CPE232_Ortega$ ls
README.md
victor@workstation:~/CPE232_Ortega$
```

- g. Use the following commands to personalize your git.

- `git config --global user.name "Your Name"`

```
victor@workstation:~$ git config --global user.name "Victor"
```


- `git config --global user.email yourname@email.com`

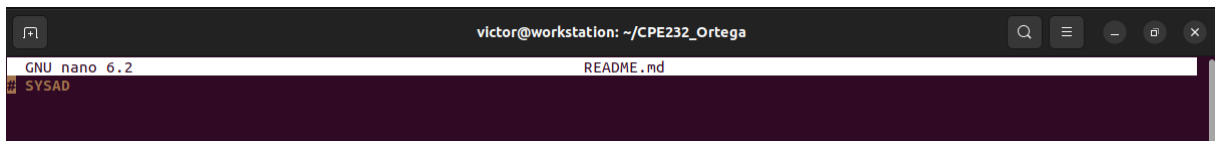
```
victor@workstation:~$ git config --global user.email "qvbortega@tip.edu.ph"
victor@workstation:~$ cat ~/.gitconfig
```

- Verify that you have personalized the config file using the command `cat ~/.gitconfig`

```
victor@workstation:~$ cat ~/.gitconfig
[user]
    name = Victor
    email = qvbortega@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.

```
victor@workstation:~$ cd CPE232_Ortega
victor@workstation:~/CPE232_Ortega$ ls
README.md
victor@workstation:~/CPE232_Ortega$ sudo nano README.md
```



- 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?

```
victor@workstation:~/CPE232_Ortega$ 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")
victor@workstation:~/CPE232_Ortega$
```

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

```
victor@workstation:~/CPE232_Ortega$ git add README.md
```

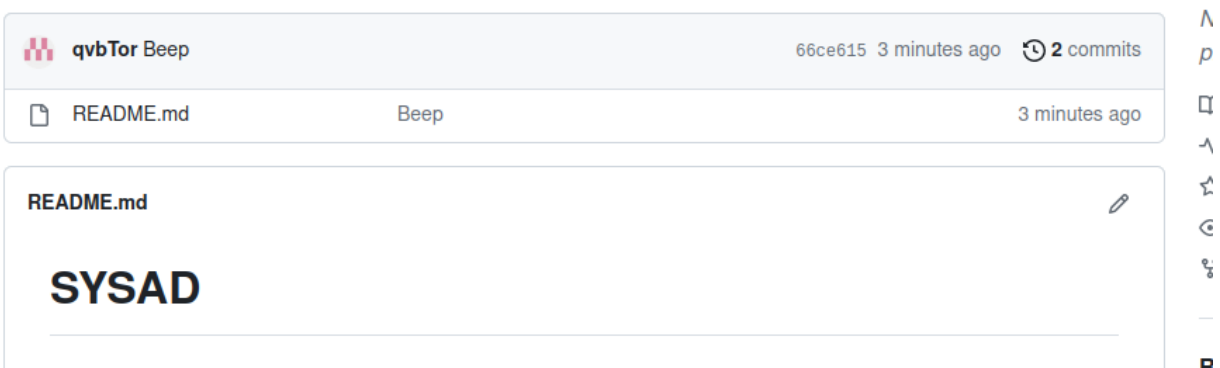
- k. Use the `git commit -m "your message"` to create a snapshot of the staged changes along the timeline of the Git projects history. The use of this command is required to select the changes that will be staged for the next commit.

```
victor@workstation:~/CPE232_Ortega$ git commit -m "Beep"
[main 66ce615] Beep
1 file changed, 1 insertion(+), 1 deletion(-)
victor@workstation:~/CPE232_Ortega$
```

- 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`.

```
victor@workstation:~/CPE232_Ortega$ git commit -m "Beep"
[main 66ce615] Beep
1 file changed, 1 insertion(+), 1 deletion(-)
victor@workstation:~/CPE232_Ortega$ git push origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Writing objects: 100% (3/3), 243 bytes | 243.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:qvbTor/CPE232_Ortega.git
d4543fb..66ce615  main -> main
victor@workstation:~/CPE232_Ortega$
```

- 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.



The screenshot shows a GitHub repository page for 'qvbTor Beep'. The repository has a commit hash of 66ce615, made 3 minutes ago, with 2 commits in total. The file 'README.md' is highlighted, showing a commit message 'Beep' made 3 minutes ago. The content of the README.md file is visible, starting with the word 'SYSAD' in large, bold letters. The interface includes a sidebar with navigation icons and a main content area with a text editor icon.

Reflections:

Answer the following:

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

In essence, Ansible commands enable you to automate a wide range of tasks related to server provisioning, configuration, maintenance, and management in a consistent and repeatable manner.

4. How important is the inventory file?

The inventory file is a crucial component in the context of managing servers using SSH and tools like Ansible. It helps streamline the process of connecting to remote servers, organizing them, and applying consistent configurations and tasks across your infrastructure.

Conclusions/Learnings:

I learned throughout the activity how to use SSH to create secure server connections without entering a password by using public and private key pairs. Along with that, I connected my Ubuntu server and GitHub successfully. As a result, I was able to easily edit the README.md file on my GitHub account from my Ubuntu desktop. The experience exceeded my initial expectations and made me aware of Ubuntu's enormous potential.