

Name: Bernardo, Christian Emmanuel	Date Performed: 23/01/2024
Course/Section: CPE232 CPE31S1	Date Submitted: 23/01/2024
Instructor: Dr. Jonathan Taylar	Semester and SY: 2nd 2023-2024
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). <i>Provide screenshots for each task.</i> 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 <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`.

```
christian@workstation: ~  
File Edit View Search Terminal Help  
christian@workstation:~$ ssh-keygen  
Generating public/private rsa key pair.  
Enter file in which to save the key (/home/christian/.ssh/id_rsa):  
Enter passphrase (empty for no passphrase):  
Enter same passphrase again:  
Your identification has been saved in /home/christian/.ssh/id_rsa.  
Your public key has been saved in /home/christian/.ssh/id_rsa.pub.  
The key fingerprint is:  
SHA256:vpEe0mk3SHePHHM0BZ8P37L641c28uCoe67yDmLWA4E christian@workstation  
The key's randomart image is:  
+---[RSA 2048]---+  
|                 ...|  
|      .           o.|  
| E .             +..|  
|      .           +o|  
|      . S . + o +|  
|      o+ = o O +o|  
|      +.X o = *.o|  
|      o .o= + o...|  
|      ***. .ooo |  
+---[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.

```
christian@workstation:~$ ssh-keygen -t rsa -b 4096  
Generating public/private rsa key pair.  
Enter file in which to save the key (/home/christian/.ssh/id_rsa):  
/home/christian/.ssh/id_rsa already exists.  
Overwrite (y/n)?
```

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`.

```
christian@workstation:~$ ls -la .ssh  
total 20  
drwx----- 2 christian christian 4096 Jan 23 18:34 .  
drwxr-xr-x 16 christian christian 4096 Jan 23 18:32 ..  
-rw----- 1 christian christian 1679 Jan 23 18:34 id_rsa  
-rw-r--r-- 1 christian christian 403 Jan 23 18:34 id_rsa.pub  
-rw-r--r-- 1 christian christian 888 Jan 23 18:17 known_hosts
```

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.

```
christian@workstation: ~  
File Edit View Search Terminal Help  
christian@workstation:~$ ssh-copy-id  
Usage: /usr/bin/ssh-copy-id [-h|-?|-f|-n] [-i [identity_file]] [-p port] [[-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  
-h|-?: print this help  
christian@workstation:~$
```

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

```
christian@workstation: ~  
File Edit View Search Terminal Help  
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/christian/  
.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 promp  
ted now it is to install the new keys  
christian@server1's password:  
  
Number of key(s) added: 1  
  
Now try logging into the machine, with: "ssh 'christian@server1'"  
and check to make sure that only the key(s) you wanted were added.  
  
christian@workstation:~$ ssh-copy-id -i ~/.ssh/id_rsa christian@server2  
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/christian/  
.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 promp  
ted now it is to install the new keys  
christian@server2's password:  
  
Number of key(s) added: 1  
  
Now try logging into the machine, with: "ssh 'christian@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?

```
christian@server1: ~  
File Edit View Search Terminal Help  
christian@workstation:~$ ssh christian@server1  
Welcome to Ubuntu 18.04.2 LTS (GNU/Linux 4.18.0-15-generic x86_64)  
  
* Documentation:  https://help.ubuntu.com  
* Management:    https://landscape.canonical.com  
* Support:        https://ubuntu.com/advantage  
  
* Canonical Livepatch is available for installation.  
  - Reduce system reboots and improve kernel security. Activate at:  
    https://ubuntu.com/livepatch  
  
696 packages can be updated.  
506 updates are security updates.  
  
New release '20.04.6 LTS' available.  
Run 'do-release-upgrade' to upgrade to it.  
  
Your Hardware Enablement Stack (HWE) is supported until April 2023.  
Last login: Tue Jan 23 18:16:03 2024 from 192.168.56.123
```

```
christian@server2: ~  
File Edit View Search Terminal Help  
christian@workstation:~$ ssh christian@server2  
Welcome to Ubuntu 18.04.2 LTS (GNU/Linux 4.18.0-15-generic x86_64)  
  
* Documentation:  https://help.ubuntu.com  
* Management:    https://landscape.canonical.com  
* Support:        https://ubuntu.com/advantage  
  
* Canonical Livepatch is available for installation.  
  - Reduce system reboots and improve kernel security. Activate at:  
    https://ubuntu.com/livepatch  
  
696 packages can be updated.  
506 updates are security updates.  
  
New release '20.04.6 LTS' available.  
Run 'do-release-upgrade' to upgrade to it.  
  
Your Hardware Enablement Stack (HWE) is supported until April 2023.  
Last login: Tue Jan 23 18:17:10 2024 from 192.168.56.123
```

Reflections:

Answer the following:

1. How will you describe the ssh-program? What does it do?
The SSH program or Secure Shell program is a network protocol that is used for secure access and management of remote systems for unsecured networks. This protocol provides a secure way to log into a system and execute commands on remote systems in secure transferring files between each other.
2. How do you know that you already installed the public key to the remote servers?

you can check the connection between the server using ssh and to see if you are able to put in a password, if you didn't get a prompt for a password you have successfully made a public key.

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*

```

christian@workstation: ~
File Edit View Search Terminal Help
christian@workstation:~$ which git
christian@workstation:~$ sudo apt install git
[sudo] password for christian:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following package was automatically installed and is no longer required:
  libllvm7
Use 'sudo apt autoremove' to remove it.
The following additional packages will be installed:
  git-man liberror-perl
Suggested packages:
  git-daemon-run | git-daemon-sysvinit git-doc git-el 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 4,817 kB of archives.
After this operation, 34.3 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://ph.archive.ubuntu.com/ubuntu bionic/main amd64 liberror-perl all 0
.17025-1 [22.8 kB]
Get:2 http://ph.archive.ubuntu.com/ubuntu bionic-updates/main amd64 git-man all
1:2.17.1-1ubuntu0.18 [804 kB]
Get:3 http://ph.archive.ubuntu.com/ubuntu bionic-updates/main amd64 git amd64 1
:2.17.1-1ubuntu0.18 [3,990 kB]
Fetched 4,817 kB in 19s (253 kB/s)
Selecting previously unselected package liberror-perl.
(Reading database ... 165232 files and directories currently installed.)

```

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

```

es Terminal ▼ Tue 18:54
christian@workstation: ~
File Edit View Search Terminal Help
christian@workstation:~$ which git
/usr/bin/git
christian@workstation:~$

```

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

```

christian@workstation: ~
File Edit View Search Terminal Help
christian@workstation:~$ git --version
git version 2.17.1
christian@workstation:~$

```

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.

The image shows two screenshots from a web browser. The top screenshot is the 'New repository' page on GitHub. The 'Owner' is 'Xerxes000' and the 'Repository name' is 'CPE232_Bernardo'. A green checkmark indicates the name is available. The 'Description' field is empty. Under 'Initialize this repository with:', the 'Add a README file' checkbox is checked. The '.gitignore' template is set to 'None'. The bottom screenshot shows the newly created repository 'CPE232_Bernardo' by 'Xerxes000'. It displays the 'README.md' file with the content 'CPE232_Bernardo'.

Owner * Xerxes000 / Repository name * CPE232_Bernardo
CPE232_Bernardo is available.

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

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.](#)

Xerxes000 / CPE232_Bernardo

Code Issues Pull requests Actions Projects Wiki

Files main

CPE232_Bernardo / README.md

Xerxes000 Initial commit now

1 lines (1 loc) · 17 Bytes

Preview Code Blame Raw

CPE232_Bernardo

- 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.

Your profile x + Private browsing -

← → ↻ https://github.com/settings/ssh/new ☆

<> Developer settings

Add new SSH Key

Title

Key type

Authentication Key ▾

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.

Title

Key type

Authentication Key ▾

Key

```
ssh-rsa |
AAAAB3NzaC1yc2EAAAADAQABAAQAC1r2MWnfNGNXCA3wv4V/icOkcCJ37ah9cdnlkoKgAAbI
oMeW9+60X3JNbSfW2
/epxDBkX2Q0j1PFSA8L3bnfuSugndbsD3lFXDrkJhNZWRzB3rc70lro9DUVlItf3yAinOV5q+Ugkfx
/LK7MUzaJeuRt+VOd1VJrwl5HbllLegvGBpM/NZf2LCkb0z3lafLAq8y5Dz
/rxepHXqm6VUwCtECNzypys6iFNx+0HAQdASD0GsQAZ79huc63R3SmpkzEd6Md34egTJ5fuUJnmV
lexaB9Mk+RFZw/s3OFpgfd9BCsDcaEnW6bMQHQTgnt2rJXumA/KsE/X9NfS8SILCnUecz
christian@workstation
```


Add SSH key

SSH keys

[New SSH key](#)

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

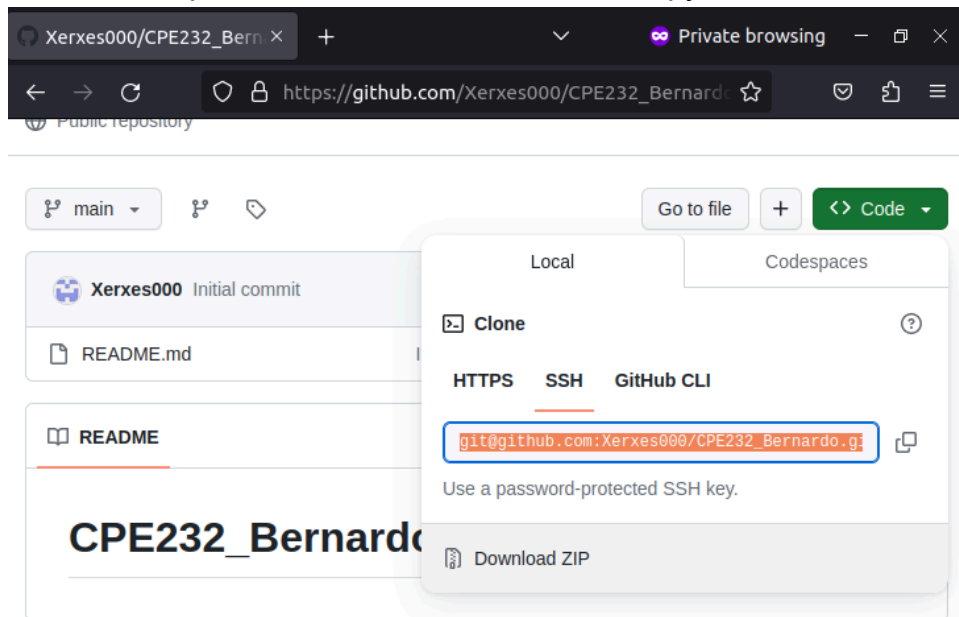
Authentication keys

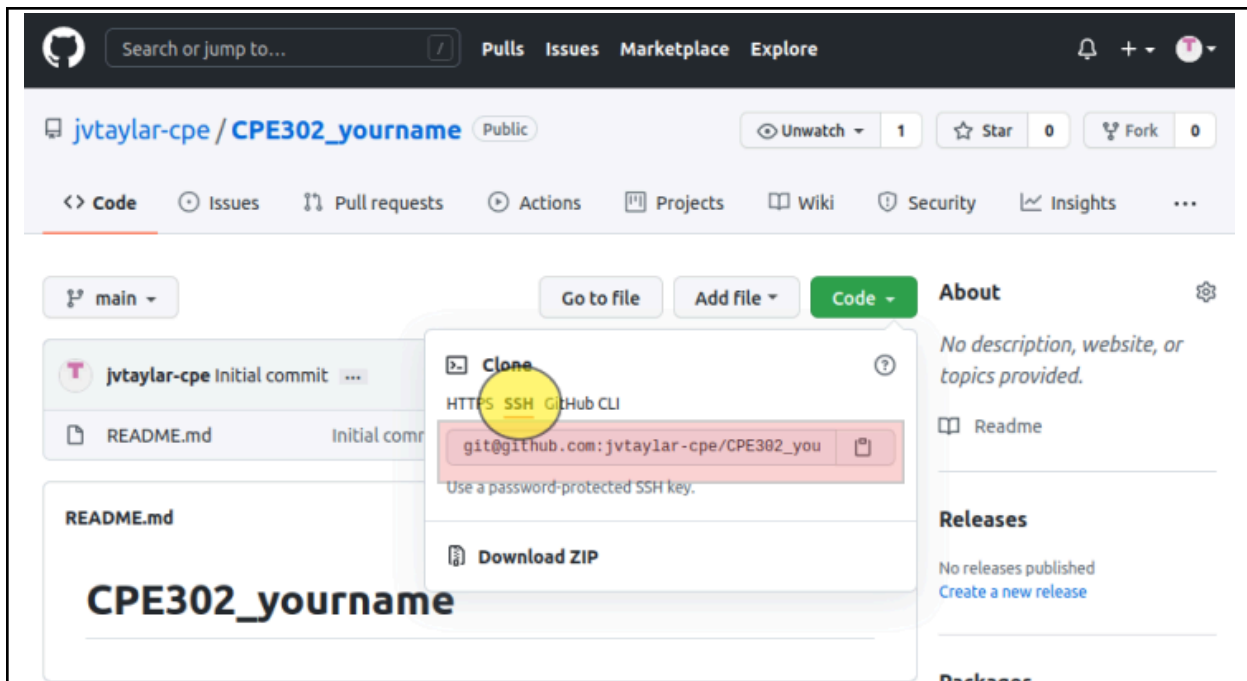
**CPE232 key**
SHA256:vpEe0mk3SHePHM0BZ8P37L641c28uCoe67yDmLWA4E
Added on Jan 23, 2024
Never used — Read/write

Delete

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:jvtaylor-cpe/CPE232_yourname.git`. When prompted to continue connecting, type yes and press enter.

```
christian@workstation: ~
File Edit View Search Terminal Help
christian@workstation:~$ git clone git@github.com:Xerxes000/CPE232_Bernardo.git
Cloning into 'CPE232_Bernardo'...
The authenticity of host 'github.com (20.205.243.166)' can't be established.
ECDSA key fingerprint is SHA256:p2QAMXNIC1TJYWeI0ttrVc98/R1BUFWu3/LiyKgUfQM.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'github.com,20.205.243.166' (ECDSA) to the list of k
nown 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`.

```
christian@workstation:~$ ls
CPE232_Bernardo  Documents  examples.desktop  Pictures  Templates
Desktop          Downloads  Music             Public   Videos
```

- 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`

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

- 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.

```
christian@workstation: ~
File Edit View Search Terminal Help
GNU nano 2.9.3 CPE232_Bernardo/README.md

# CPE232_Bernardo
#NAME: Christian
#Program: BSCPE
#Course: CPE232
#Email: Christianbaenardo@gmail.com
```

- 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?

```
christian@workstation:~/CPE232_Bernardo$ 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 checkout -- <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.

```
christian@workstation: ~/CPE232_Bernardo
File Edit View Search Terminal Help
christian@workstation:~/CPE232_Bernardo$ git add README.md
christian@workstation:~/CPE232_Bernardo$
```

- 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.

```
christian@workstation: ~/CPE232_Bernardo
File Edit View Search Terminal Help
christian@workstation:~/CPE232_Bernardo$ git commit -m "Hello"
[main 29835e0] Hello
1 file changed, 5 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`.

```
christian@workstation:~/CPE232_Bernardo$ git push origin main
Counting objects: 3, done.
Delta compression using up to 3 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 326 bytes | 326.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To github.com:Xerxes000/CPE232_Bernardo.git
1d252bd..29835e0  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.

← Files
main

CPE232_Bernardo / README.md

Xerxes000 Hello
1 minute ago

5 lines (5 loc) · 103 Bytes

Preview
Code
Blame

Raw
Copy
Download
Edit

CPE232_Bernardo

#NAME: Christian #Program: BSCPE #Course: CPE232 #Email: Christianbaenardo@gmail.com

Reflections:

Answer the following:

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

Ansible is a powerful tool that automates the management and configuration of multiple servers simultaneously. here are some ansible commands examples:

System Configuration:

Install or update packages and software on remote servers.
Configure system settings, such as network configurations, hostname, etc.

User and Permissions Management:

Create or delete user accounts.
Manage user permissions and group memberships.

File Operations:

Copy files and directories to remote servers.
Manage file permissions and ownership.

Service Management:

Start, stop, or restart services on remote servers.
Configure and manage services, such as web servers, databases, etc.

Security Configuration:

Update system security settings.
Apply security patches.

Monitoring:

Retrieve system information and monitor server health.

Automation of Custom Tasks:

Execute custom scripts or commands on remote servers.
Automate complex workflows.

4. How important is the inventory file?

the ansible inventory is very crucial for the servers in defining the hosts and groups of other hosts that ansible manages

Conclusions/Learnings:

In this activity I learned that SSH keys can help in making the servers in Linux more secure, they can be both public and private to make sure that people can see what they have the credentials for. In the SSH it is a great tool to make sure that not everyone can see the server that you can control and to only give the credentials or keys when it is needed. The Github setup with the Linux operating system can be a good for the user because of the extra security it can give for the privacy.