Name: Andaya, Lyka C.	**Date Performed: August 22, 2023**
Course/Section: CPE31S4	**Date Submitted: August 22, 2023**
Instructor: Dr. Taylar	**Semester and SY: 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). ***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.

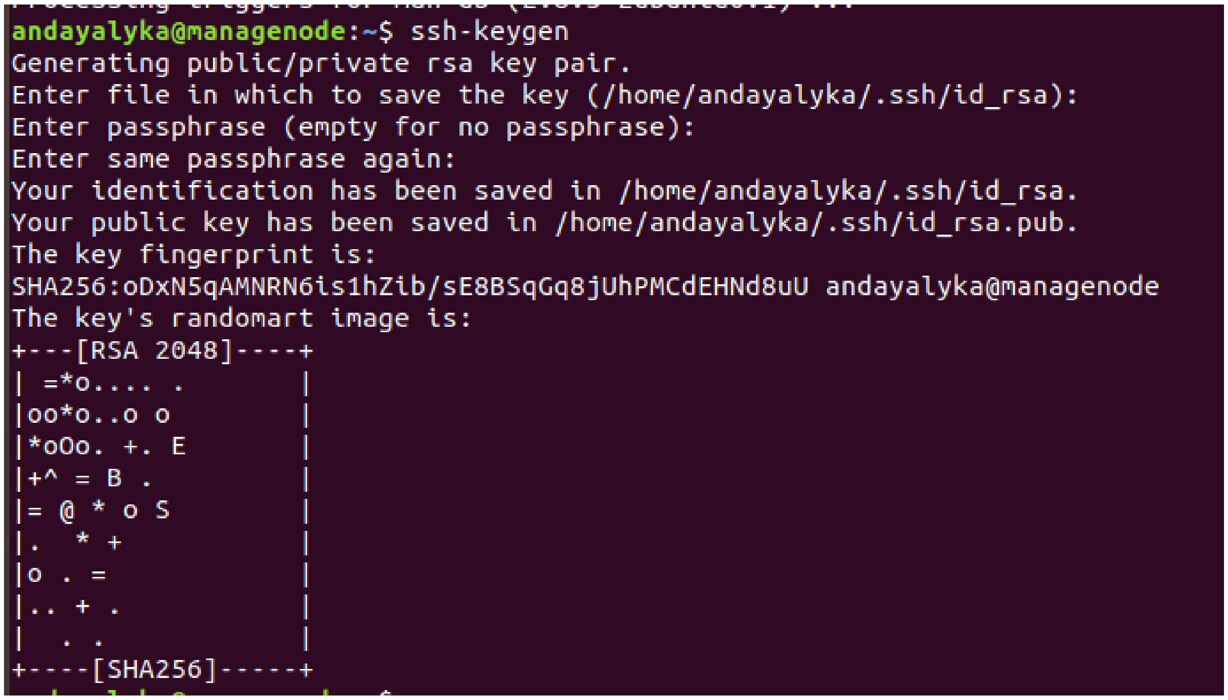
SSH introduced public key authentication as a more secure alternative to the older .rhosts authentication. It improved security by avoiding the need to have password stored in files and eliminated the possibility of a compromised server stealing the user's password.

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

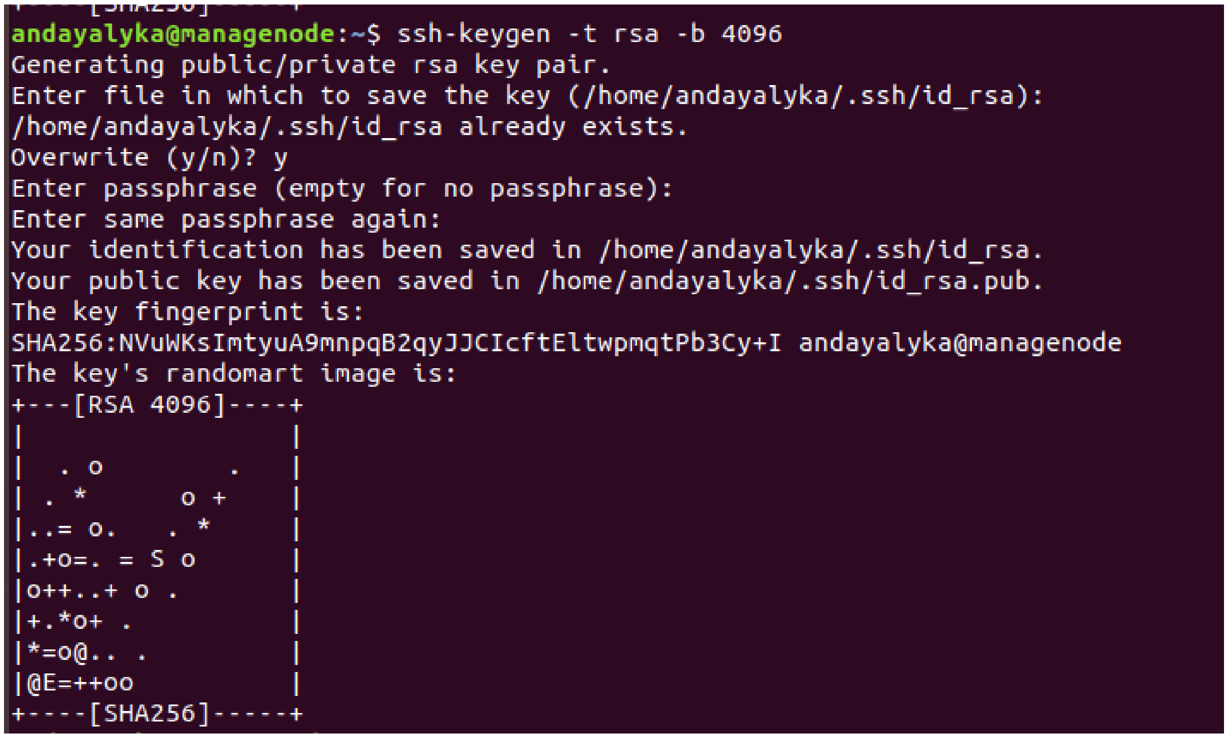
1. The simplest way to generate a key pair is to run *ssh-keygen* without arguments. In this case, it will prompt for the file in which to store keys. First,

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*.



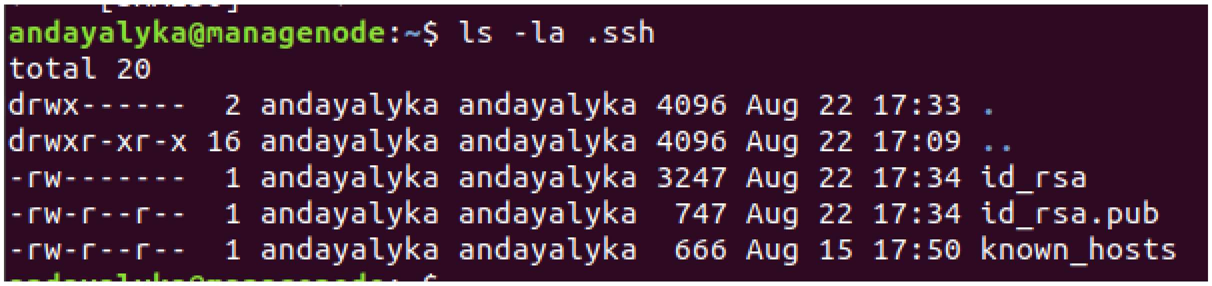
```
andayalyka@managenode:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/andayalyka/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/andayalyka/.ssh/id_rsa.
Your public key has been saved in /home/andayalyka/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:oDxN5qAMNRN6is1hZib/sE8BSqGq8jUhPMCdEHNd8uU andayalyka@managenode
The key's randomart image is:
+---[RSA 2048]----+
| =*o.... .       |
|oo*o..o o        |
|*oOo. +. E       |
|+^ = B .         |
|= @ * o S        |
|.  * +           |
|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.



```
andayalyka@managenode:~$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/andayalyka/.ssh/id_rsa):
/home/andayalyka/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/andayalyka/.ssh/id_rsa.
Your public key has been saved in /home/andayalyka/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:NVuWKsImtyuA9mnpqB2qyJJCIcftEltwpmqtPb3Cy+I andayalyka@managenode
The key's randomart image is:
+---[RSA 4096]----+
|                 |
|  . o       .    |
| . *      o +    |
|..= o.   . *     |
|.+o=. = S o      |
|o++..+ o .       |
|+.*o+ .          |
|*=o@.. .         |
|@E=++oo          |
+----[SHA256]-----+
```

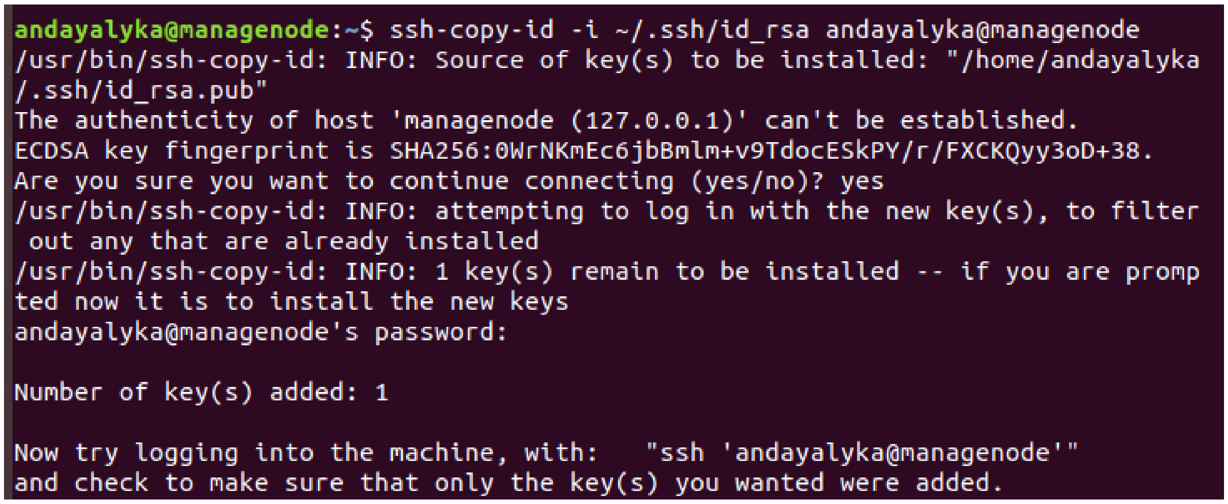
3. When asked for a passphrase, just press enter. The passphrase is used for encrypting the key, so that it cannot be used even if someone obtains the private key file. The passphrase should be cryptographically strong.
4. Verify that you have created the key by issuing the command *ls -la .ssh.* The command should show the .ssh directory containing a pair of keys. For example, id_rsa.pub and id_rsa.



```
andayalyka@managenode:~$ ls -la .ssh
total 20
drwx------  2 andayalyka andayalyka 4096 Aug 22 17:33 .
drwxr-xr-x 16 andayalyka andayalyka 4096 Aug 22 17:09 ..
-rw-------  1 andayalyka andayalyka 3247 Aug 22 17:34 id_rsa
-rw-r--r--  1 andayalyka andayalyka  747 Aug 22 17:34 id_rsa.pub
-rw-r--r--  1 andayalyka andayalyka  666 Aug 15 17:50 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.
2. Issue the command similar to this: *ssh-copy-id -i ~/.ssh/id_rsa user@host*



```
andayalyka@managenode:~$ ssh-copy-id -i ~/.ssh/id_rsa andayalyka@managenode
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/andayalyka
/.ssh/id_rsa.pub"
The authenticity of host 'managenode (127.0.0.1)' can't be established.
ECDSA key fingerprint is SHA256:0WrNKmEc6jbBmlm+v9TdocESkPY/r/FXCKQyy3oD+38.
Are you sure you want to continue connecting (yes/no)? 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 promp
ted now it is to install the new keys
andayalyka@managenode's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'andayalyka@managenode'"
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?

Reflections:

Answer the following:

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

- **SSH, known as Secure Shell, functions as both a network protocol and software application utilized to enable safe remote entry to systems across a potentially insecure network, such as the internet. It facilitates the creation of encrypted communication channels connecting two devices, affording users the ability to securely log into remote systems, carry out commands, move files, and oversee network devices. SSH was developed with the intention of supplanting outdated and less secure protocols like Telnet and rlogin. These protocols conveyed information in plain text, rendering them susceptible to interception and various types of cyber assaults. To counteract these security vulnerabilities, SSH encrypts all exchanged data between the client and server, thereby ensuring the privacy and integrity of the communication.**

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

   - **To ascertain whether your public key has been set up on distant servers to enable SSH access, begin by verifying your local public key. Following that, establish connections with the remote servers. Subsequently, examine the authentication method in use, and finally, inspect the authorized_keys file. It's important to note that SSH public keys are unique to individual user accounts on remote servers. Moreover, certain systems might possess varying paths or configurations for SSH key files, so it's essential to adapt the commands as per your specific setup.**

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**).

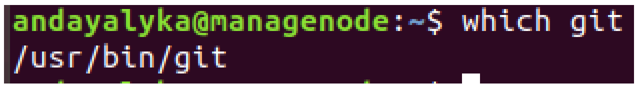
Set up Git

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

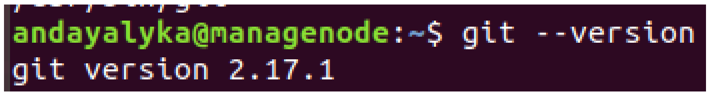
Task 3: Set up the Git Repository

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*



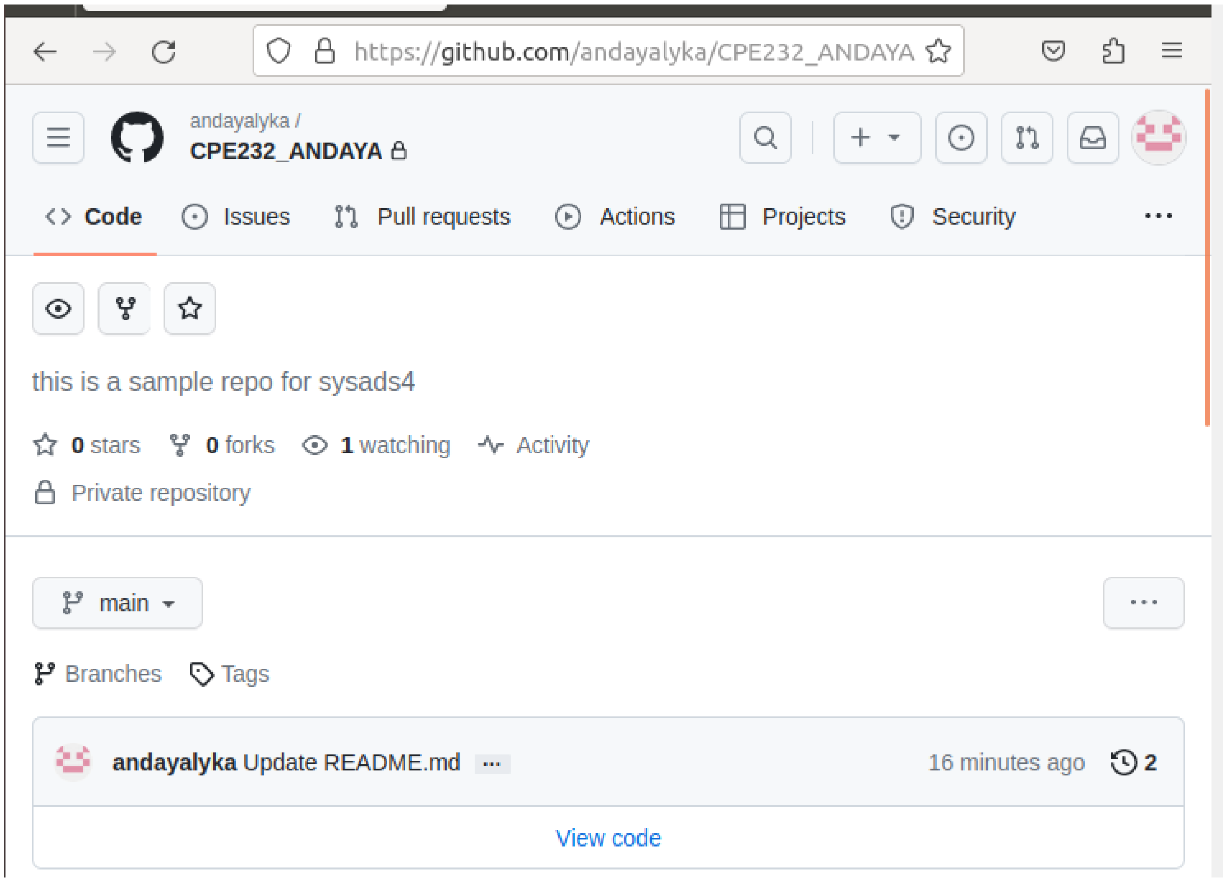
```
andayalyka@managenode:~$ which git
/usr/bin/git
```

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.



```
andayalyka@managenode:~$ git --version
git version 2.17.1
```

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.

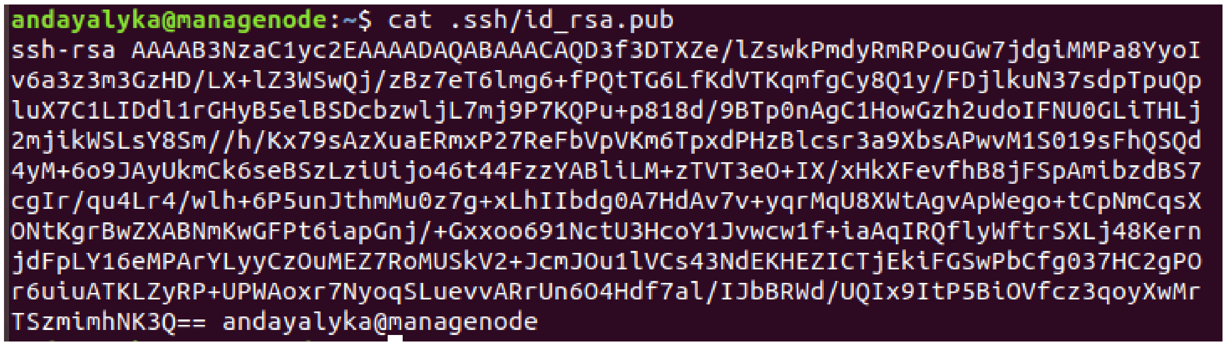


this is a sample repo for sysads4

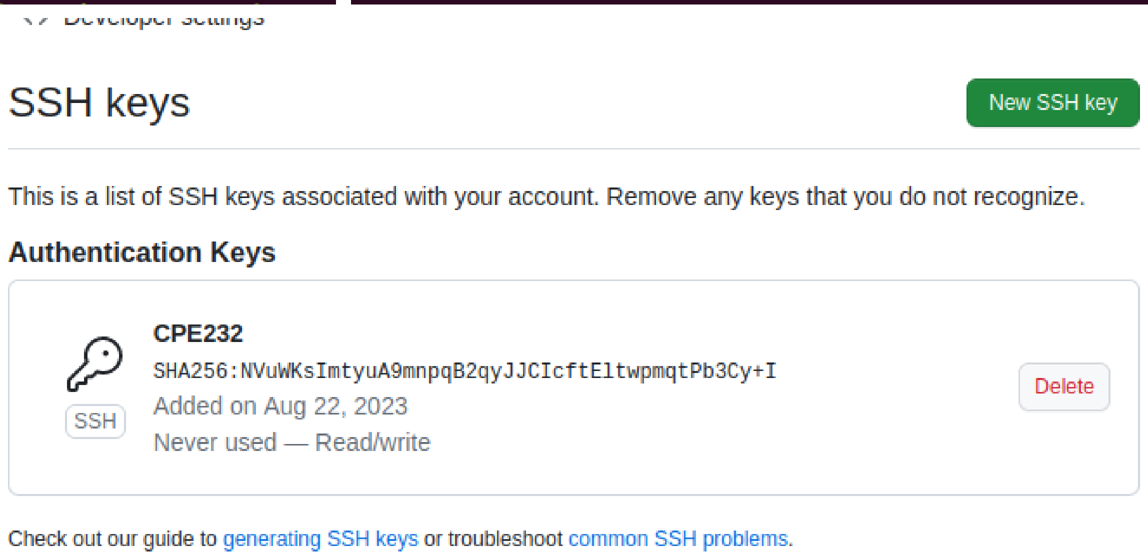
andayalyka Update README.md ··· 16 minutes ago 2

    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.



```
andayalyka@managenode:~$ cat .ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAACAQD3f3DTXZe/lZswkPmdyRmRPouGw7jdgiMMPa8YyoI
v6a3z3m3GzHD/LX+lZ3WSwQj/zBz7eT6lmg6+fPQtTG6LfKdVTKqmfgCy8Q1y/FDjlkuN37sdpTpuQp
luX7C1LIDdl1rGHyB5elBSDcbzwljL7mj9P7KQPu+p818d/9BTp0nAgC1HowGzh2udoIFNU0GLiTHLj
2mjikWSLsY8Sm//h/Kx79sAzXuaERmxP27ReFbVpVKm6TpxdPHzBlcsr3a9XbsAPwvM1S019sFhQSQd
4yM+6o9JAyUkmCk6seBSzLziUijo46t44FzzYABliLM+zTVT3eO+IX/xHkXFevfhB8jFSpAmibzdBS7
cgIr/qu4Lr4/wlh+6P5unJthmMu0z7g+xLhIIbdg0A7HdAv7v+yqrMqU8XWtAgvApWego+tCpNmCqsX
ONtKgrBwZXABNmKwGFPt6iapGnj/+Gxxoo691NctU3HcoY1Jvwcw1f+iaAqIRQflyWftrSXLj48Kern
jdFpLY16eMPArYLyyCzOuMEZ7RoMUSkV2+JcmJOu1lVCs43NdEKHEZICTjEkiFGSwPbCfg037HC2gPO
r6uiuATKLZyRP+UPWAoxr7NyoqSLuevvARrUn6O4Hdf7al/IJbBRWd/UQIx9ItP5BiOVfcz3qoyXwMr
TSzmimhNK3Q== andayalyka@managenode
```



SSH keys

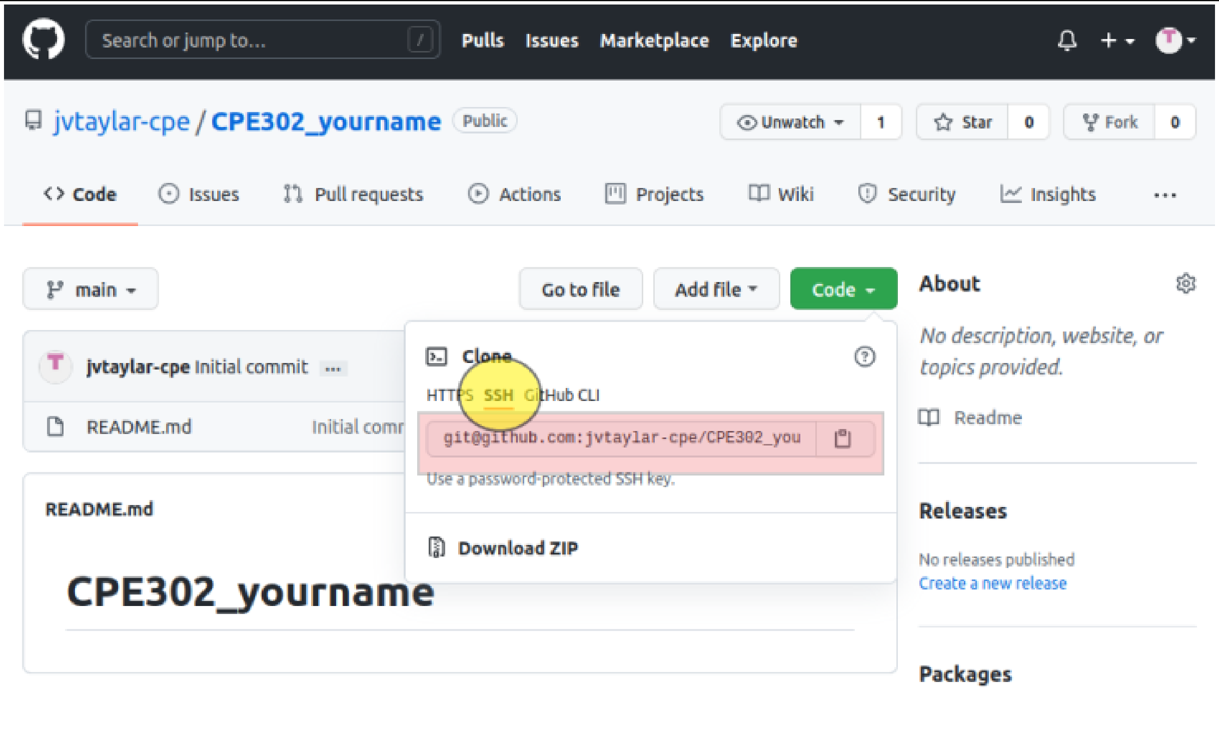
This is a list of SSH keys associated with your account. Remove any keys that you do not recognize.

Authentication Keys

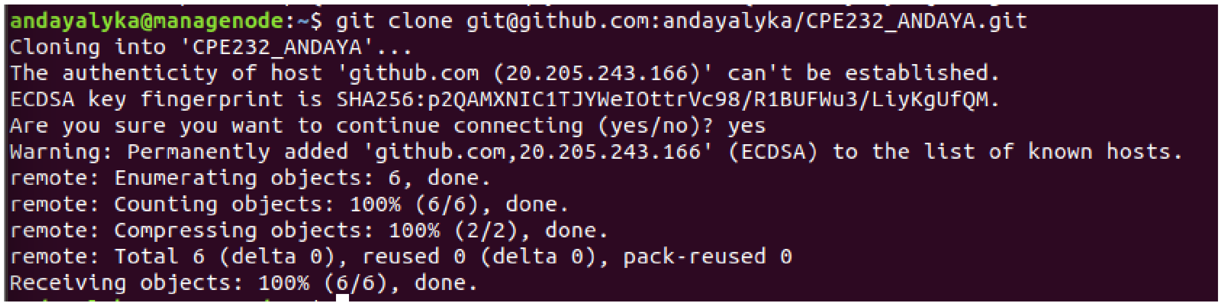
CPE232
SHA256:NVuWKsImtyuA9mnpqB2qyJJCIcftEltwpmqtPb3Cy+I
Added on Aug 22, 2023
Never used — Read/write

Check out our guide to generating 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.

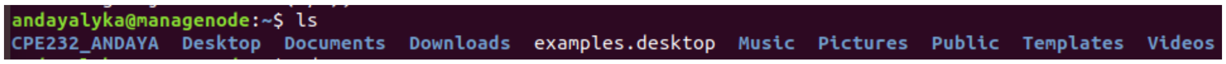


e. Issue the command git clone followed by the copied link. For example, *git clone git@github.com:jvtaylar-cpe/CPE232_yourname.git*. When prompted to continue connecting, type yes and press enter.

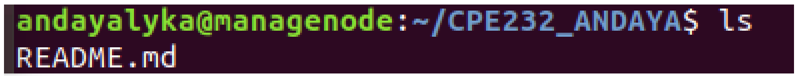


```
andayalyka@managenode:~$ git clone git@github.com:andayalyka/CPE232_ANDAYA.git
Cloning into 'CPE232_ANDAYA'...
The authenticity of host 'github.com (20.205.243.166)' can't be established.
ECDSA key fingerprint is SHA256:p2QAMXNIC1TJYWeIOttrVc98/R1BUFWu3/LiyKgUfQM.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'github.com,20.205.243.166' (ECDSA) to the list of known hosts.
remote: Enumerating objects: 6, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 6 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (6/6), 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.



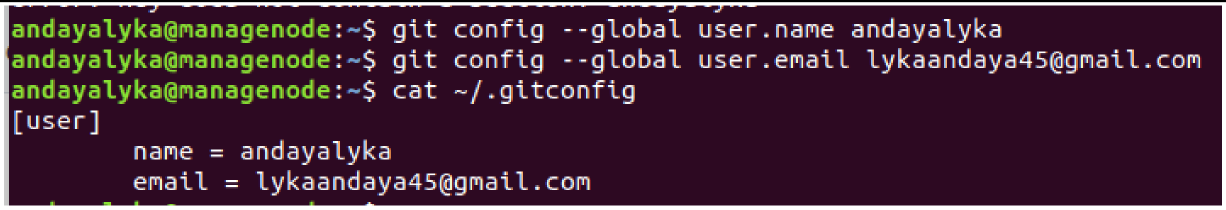
```
andayalyka@managenode:~$ ls
CPE232_ANDAYA  Desktop  Documents  Downloads  examples.desktop  Music  Pictures  Public  Templates  Videos
```



```
andayalyka@managenode:~/CPE232_ANDAYA$ ls
README.md
```

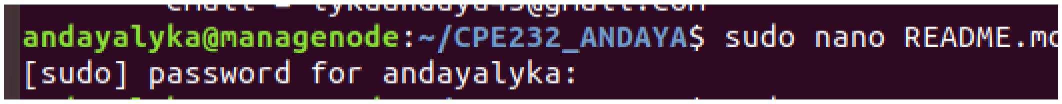
g. Use the following commands to personalize your git.

- *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*

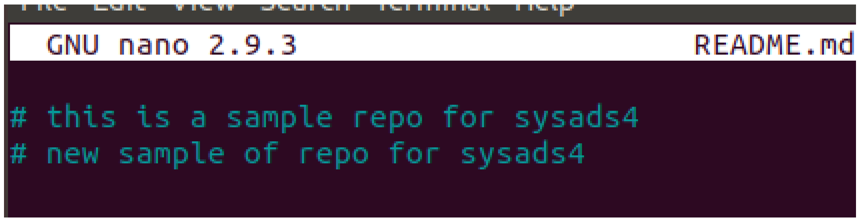


```
andayalyka@managenode:~$ git config --global user.name andayalyka
andayalyka@managenode:~$ git config --global user.email lykaandaya45@gmail.com
andayalyka@managenode:~$ cat ~/.gitconfig
[user]
        name = andayalyka
        email = lykaandaya45@gmail.com
```

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.

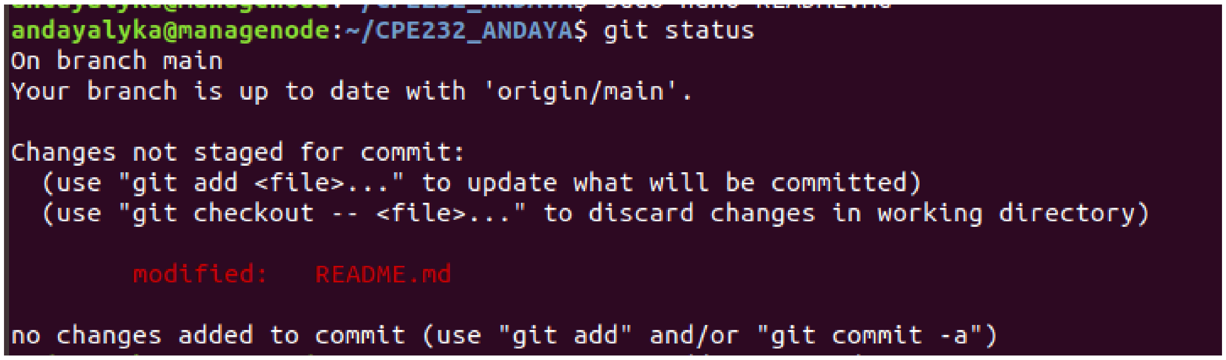


```
andayalyka@managenode:~/CPE232_ANDAYA$ sudo nano README.md
[sudo] password for andayalyka:
```



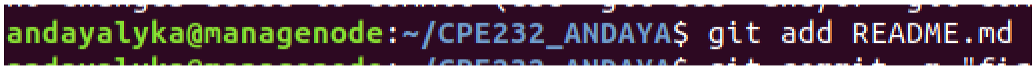
```
  GNU nano 2.9.3                             README.md

# this is a sample repo for sysads4
# new sample of repo for sysads4
```

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?



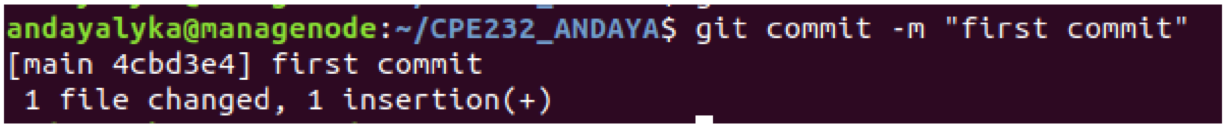
```
andayalyka@managenode:~/CPE232_ANDAYA$ git status
On branch main
Your branch is up to date with 'origin/main'.

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.



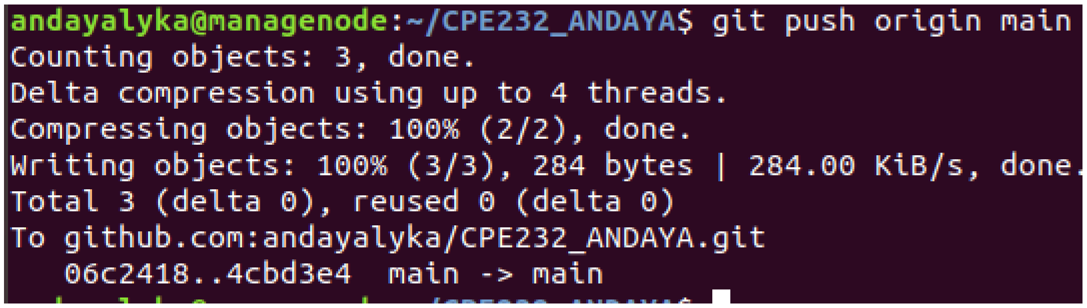
```
andayalyka@managenode:~/CPE232_ANDAYA$ git add README.md
```

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.



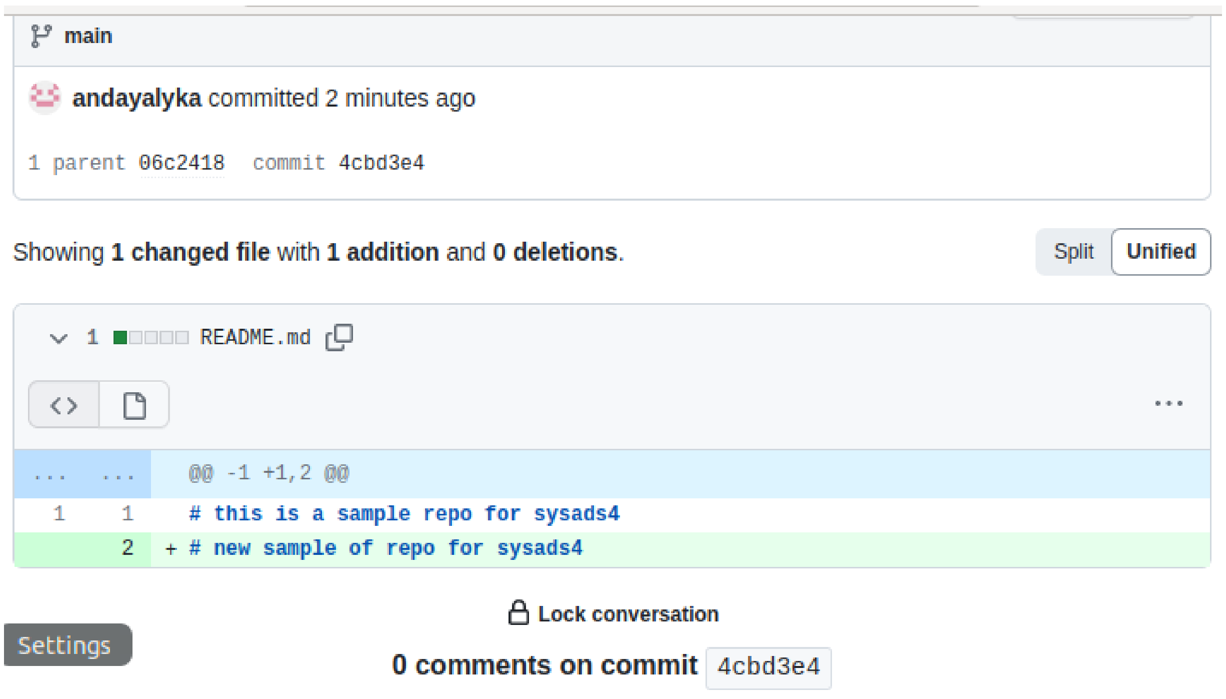
```
andayalyka@managenode:~/CPE232_ANDAYA$ git commit -m "first commit"
[main 4cbd3e4] first commit
 1 file changed, 1 insertion(+)
```

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.*



```
andayalyka@managenode:~/CPE232_ANDAYA$ git push origin main
Counting objects: 3, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 284 bytes | 284.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To github.com:andayalyka/CPE232_ANDAYA.git
   06c2418..4cbd3e4  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.



main

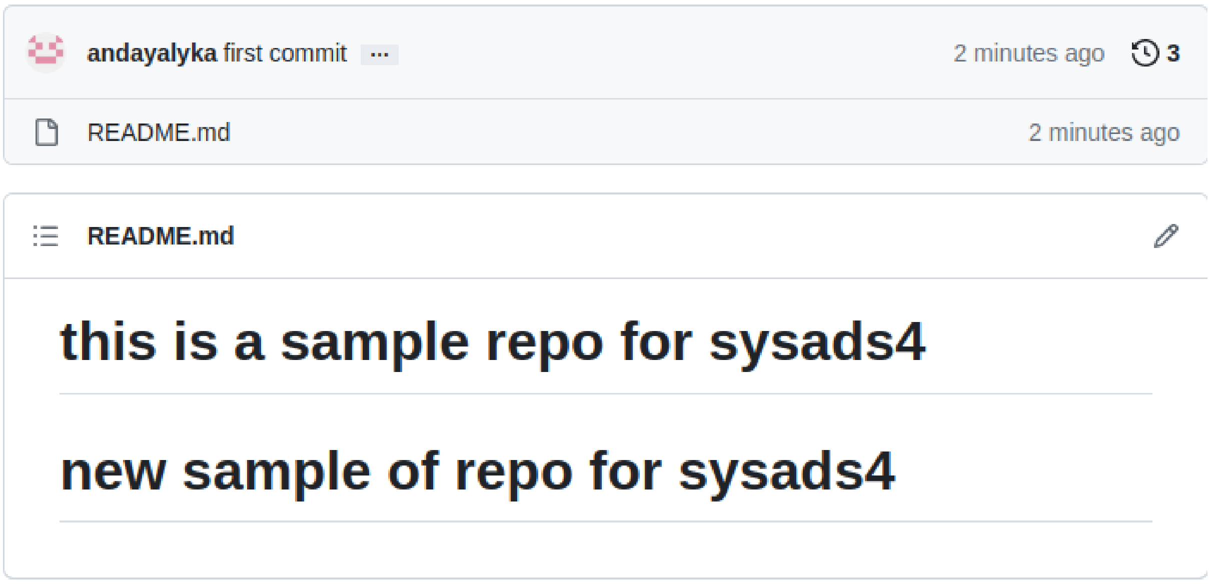
andayalyka committed 2 minutes ago

1 parent 06c2418   commit 4cbd3e4

Showing **1 changed file** with **1 addition** and **0 deletions**.

```
1 README.md
@@ -1 +1,2 @@
1  1    # this is a sample repo for sysads4
   2  + # new sample of repo for sysads4
```

0 comments on commit 4cbd3e4


andayalyka first commit ··· 2 minutes ago 3

README.md 2 minutes ago

README.md

this is a sample repo for sysads4

new sample of repo for sysads4

Reflections:
Answer the following:

3. What sort of things have we so far done to the remote servers using ansible commands?
   - **Commands in Ansible for overseeing distant servers. Ansible serves as an automation utility that grants the capability to supervise and set up remote servers in a uniform and replicable fashion. It offers the capability to deploy or upgrade packages and applications on remote servers. Additionally, Ansible is equipped to regulate configuration files on remote servers, ensuring alignment with specified settings. This encompasses assignments such as revising configuration parameters within files. The tool further enables the movement of files between your local system and remote servers, proving beneficial for actions like duplicating configuration files or disseminating scripts.**
4. How important is the inventory file?
   - **The inventory file holds significant importance within Ansible, as it outlines the hosts and host groups that Ansible will oversee and set up. This file functions as a means to structure and group the destination systems with which Ansible engages. Essentially, the inventory file stands as a pivotal aspect of Ansible's capabilities. It empowers you to set the boundaries of automation, pinpoint the locations of actions, and tailor the configuration of each system to its distinct needs. It plays a vital role in upholding uniformity, overseeing configurations, and automating operations across a variety of servers and environments.**

Conclusions/Learnings:

Enabling SSH-based connectivity between a local and remote machine through a key pair (comprising a public and a private key) instead of relying on passwords encompasses a series of steps to ensure secure and automated authentication. This procedure heightens security by eliminating the necessity of transmitting passwords across the network. Upon completing this sequence, you will have established a secure and streamlined approach for linking to distant machines using SSH keys. This authentication mechanism bolsters security, streamlines the login process, and constitutes a pivotal practice for individuals overseeing remote systems.

Creating a Git repository entails setting up a local repository on your computer and connecting it to an external repository hosted on platforms such as GitHub, GitLab, or Bitbucket. Upon completing this process, you will possess a strong grasp of how to initiate and collaborate within both local and remote Git repositories. This understanding is crucial for proficient version management, collaborative work, and overseeing code in software development projects.