Name: Abigail Laxamana	**Date Performed:** August 24, 2023
Course/Section: CPE 232 - CPE31S6	**Date Submitted:** August 24, 2023
Instructor: Dr. Jonathan V. Taylar	**Semester and SY:** 1st Sem - 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*.

```
laxamana_ubuntu@workstation:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/laxamana_ubuntu/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/laxamana_ubuntu/.ssh/id_rsa.
Your public key has been saved in /home/laxamana_ubuntu/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:+gwEwaNqCRBIrTai55xFbZCPnlT+6JfxvzmxcQNBqU4 laxamana_ubuntu@workstation
The key's randomart image is:
+---[RSA 2048]----+
|+o....     ...   |
|o  .=..     o    |
|. ...O     . .   |
|o+. +.=   E .    |
|=.o+ o.oSo   .   |
|oo. +...o . o o  |
|.+ o .o  +   = . |
|  +   .+o . o.   |
|       .o  .+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.

```
laxamana_ubuntu@workstation:~$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/laxamana_ubuntu/.ssh/id_rsa):
/home/laxamana_ubuntu/.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/laxamana_ubuntu/.ssh/id_rsa.
Your public key has been saved in /home/laxamana_ubuntu/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:uwzT6DrxWmPRzH8tDikWXUKP/XYiMDVQ7ZCwrVwZ5GQ laxamana_ubuntu@workstation
The key's randomart image is:
+---[RSA 4096]----+
|          +=Eo   |
|         . @o+.  |
|          * Oo   |
|       + o B ..  |
|      . S + . + .|
|     .  + + . + o |
|     oB = + o .  |
|    .+.* o + .   |
|    o+. o   .    |
+----[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.

```
laxamana_ubuntu@workstation:~$ ls -la .ssh
total 20
drwx------  2 laxamana_ubuntu laxamana_ubuntu 4096 Aug 24 17:30 .
drwxr-xr-x 15 laxamana_ubuntu laxamana_ubuntu 4096 Aug 17 18:36 ..
-rw-------  1 laxamana_ubuntu laxamana_ubuntu 3247 Aug 24 17:32 id_rsa
-rw-r--r--  1 laxamana_ubuntu laxamana_ubuntu  753 Aug 24 17:32 id_rsa.pub
-rw-r--r--  1 laxamana_ubuntu laxamana_ubuntu  888 Aug 17 18:44 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*

```
laxamana_ubuntu@workstation:~/.ssh$ ssh-copy-id -i ~/.ssh/id_rsa laxamana_ubunt
u@workstation
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/laxamana_u
buntu/.ssh/id_rsa.pub"
The authenticity of host 'workstation (127.0.0.1)' can't be established.
ECDSA key fingerprint is SHA256:0fLmuhpPOqrdcoWOgunjzLUpo/kNAy6YAf8dBINIe6A.
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
laxamana_ubuntu@workstation's password:

Number of key(s) added: 1

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

In order to ensure secure communication and remote access to computers and networked devices over potentially insecure networks, like the internet, Secure Shell (SSH), which is both a network protocol and an application, is essential. This utility performs a number of vital tasks.

SSH is essential for system administrators and remote server administration because it offers a very secure route for remote access, enabling users to connect to distant systems and issue instructions. SSH encrypts data as it is being transmitted, ensuring the integrity and confidentiality of data shared between the client and server. In order to guarantee that only authorized users are granted access, it also provides a variety of authentication techniques, such as passwords and public key pairs.

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

We can check it by using ***ls -la .ssh*** in terminal to see if there is **id_rsa.pub**, or **id_rsa** which is the typical sign that the key have been made. We can also check if the public key have been listed by using the command ***cat ~/.ssh/authorized_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

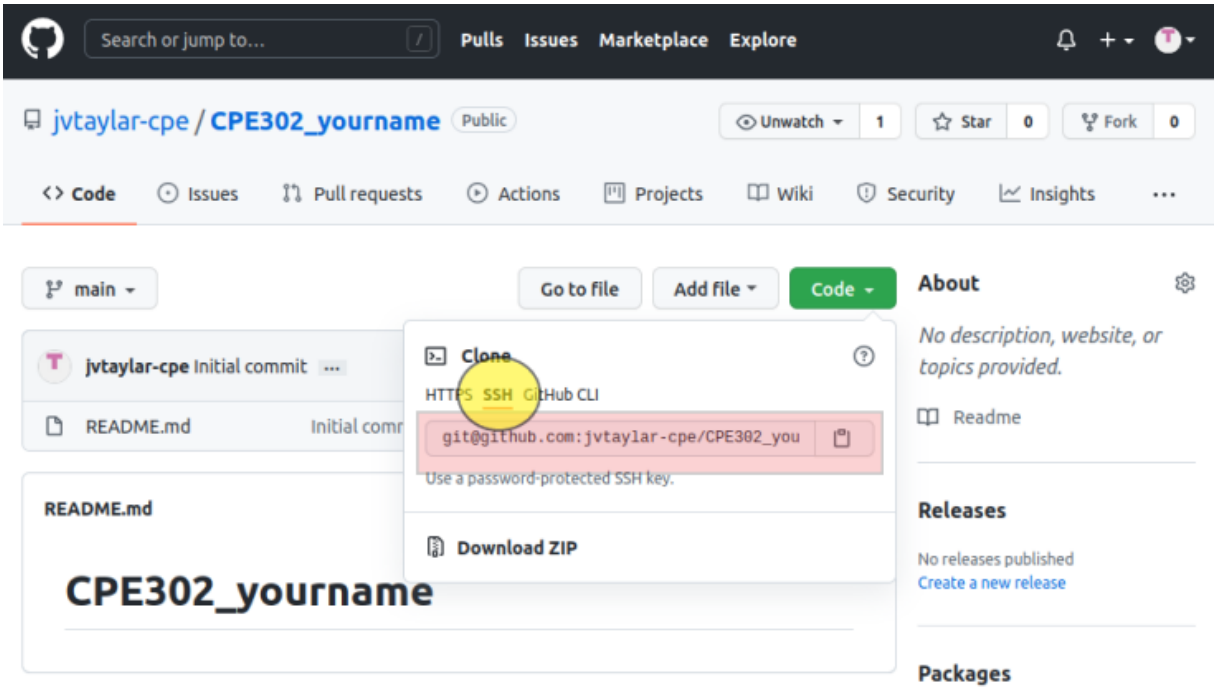
At the heart of GitHub is an open-source version control system (VCS) called Git. Git is responsible for everything GitHub-related that happens locally on your computer. To use Git on the command line, you'll need to download, install, and configure Git on your computer. You can also install GitHub CLI 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.

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.

Reflections:

Answer the following:

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

4. How important is the inventory file?

An inventory file is a common term used in the software and IT industries to describe a list of all hardware and software resources inside a business. Specifications for the device, software licenses, configurations, and maintenance schedules are among the information included. This document is crucial for IT managers and administrators because it facilitates effective resource management, licensing compliance, security management, and

troubleshooting. By detecting and addressing risks, maintaining an accurate inventory file helps improve cybersecurity while saving time and costs.

Conclusions/Learnings:

After completing this assignment, I learnt that we can utilize git to meticulously monitor code changes, enabling streamlined team collaboration. The key takeaway from Git repositories is the effectiveness of organized version control, which empowers teams to confidently manage complicated codebases. Parallel development efforts are encouraged by branching and merging features, while remote repositories provide centralized hubs for backup and communication. Git repositories also act as a safety net by facilitating catastrophe recovery through routine pushes to distant repositories. Git repositories are a crucial tool for every developer or team of developers because they highlight the need of communication, structure, and version history monitoring in the dynamic world of software development.