

Name:Calderon Ricardo B.	Date Performed:01-23-24
Course/Section:CPE31S1	Date Submitted:01-23-24
Instructor: Dr. Jonathan Taylar	Semester and SY: 2nd 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). <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`.

```
calderon@workstation:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/calderon/.ssh/id_rsa):
Thunderbird Mail /home/calderon/.ssh
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/calderon/.ssh/id_rsa
Your public key has been saved in /home/calderon/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:DCeEdMtW3NSoBl/tGqRrDBXzkM3GqADCFj5/VgjdJBQ calderon@workstation
The key's randomart image is:
+---[RSA 3072]-----+
|o.o++E*o=X.+|
|. + +=+=X o |
|.o o**.*.. |
| o .+=. . |
|. o +S. o |
| o + . |
| . |
+-----[SHA256]-----+
calderon@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.

```
calderon@workstation:~$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.

Enter file in which to save the key (/home/calderon/.ssh/id_rsa): /home/calderon/.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/calderon/.ssh/id_rsa
Your public key has been saved in /home/calderon/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:T7z8KzzriT00qpX5jdliY4p39As5fFer44NUZUPL3LQ calderon@workstation
The key's randomart image is:
+---[RSA 4096]-----+
| .. |
| o++ . |
| o+.E |
| .. |
|.S + |
| ..+. = o |
|.Oo....= |
|. +.oBo==o+o |
|o+o=oo*+*+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.

```
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/calderon/.ssh/id_rsa
Your public key has been saved in /home/calderon/.ssh/id_rsa.pub
```

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`.

```
calderon@workstation:~$ ls -la .ssh
total 16
drwx----- 2 calderon calderon 4096 Jan 23 15:28 .
drwxr-x--- 15 calderon calderon 4096 Jan 23 15:28 ..
-rw----- 1 calderon calderon 3389 Jan 23 15:30 id_rsa
-rw-r--r-- 1 calderon calderon 746 Jan 23 15:30 id_rsa.pub
```

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`

```
calderon@workstation:~$ ssh-copy-id -i ~/.ssh/id_rsa calderon@workstation
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/calderon/.ssh/id_rsa.pub"
The authenticity of host 'workstation (127.0.0.1)' can't be established.
ED25519 key fingerprint is SHA256:4hWWCqsAMZMBTFzuA/xwSTh5QsHsCyEl51wGGbde77g.
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
calderon@workstation's password:

Number of key(s) added: 1

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

Server 1

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

Server 2

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

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?

Server1

```
calderon@workstation:~$ ssh 'calderon@server1'
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 6.2.0-26-generic x86_64)

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

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

The list of available updates is more than a week old.
To check for new updates run: sudo apt update
```

Server2

```
calderon@workstation:~$ ssh 'calderon@server2'
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 6.2.0-26-generic x86_64)

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

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
```

Reflections:

Answer the following:

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

SSH (Secure Shell) is a cryptographic network protocol used to securely communicate with remote servers. It is typically used to remotely access and manage a computer or server over a network.

SSH provides strong encryption and authentication mechanisms, which ensure that all communications between a client and a server are secure and cannot be intercepted by unauthorized parties. It is widely used by system administrators and developers to securely transfer files, run commands, and securely access remote resources.

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

If you have installed the public key to the remote server, you can test it by attempting to SSH to the server using your private key. If you are able to connect to the server without being prompted for a password, then the public key is installed correctly. You can also check the `authorized_keys` file on the remote server to verify that your public key is listed there.

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*

```
calderon@Workstation:~$ sudo apt install git
[sudo] password for calderon:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  git-man liberror-perl
Suggested packages:
  git-daemon-run | git-daemon-sysvinit git-doc 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 188 not upgraded.
Need to get 4,147 kB of archives.
After this operation, 21.0 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://us.archive.ubuntu.com/ubuntu jammy/main amd64 liberror-perl all
7029-1 [26.5 kB]
Get:2 http://us.archive.ubuntu.com/ubuntu jammy-updates/main amd64 git-man al
:2.34.1-1ubuntu1.10 [954 kB]
Get:3 http://us.archive.ubuntu.com/ubuntu jammy-updates/main amd64 git amd64
.34.1-1ubuntu1.10 [3,166 kB]
Fetched 4,147 kB in 6s (732 kB/s)
```

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

```
calderon@Workstation:~$ which git
/usr/bin/git
calderon@Workstation:~$
```

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

```
calderon@Workstation:~$ git --version
git version 2.34.1
calderon@Workstation:~$
```

4. Using the browser in the local machine, go to www.github.com.

←

→

↻

🔒 https://github.com

🏠

🔍

☆


📄

☆

🔖

🔔

Sign in



☰

0

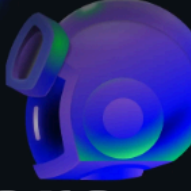
Let's build from here

The world's leading AI-powered developer platform.

Email address

Sign up for GitHub

<> Start a free enterprise trial >



5. Sign up in case you don't have an account yet. Otherwise, login to your GitHub account.



Sign in to GitHub

Username or email address

Password

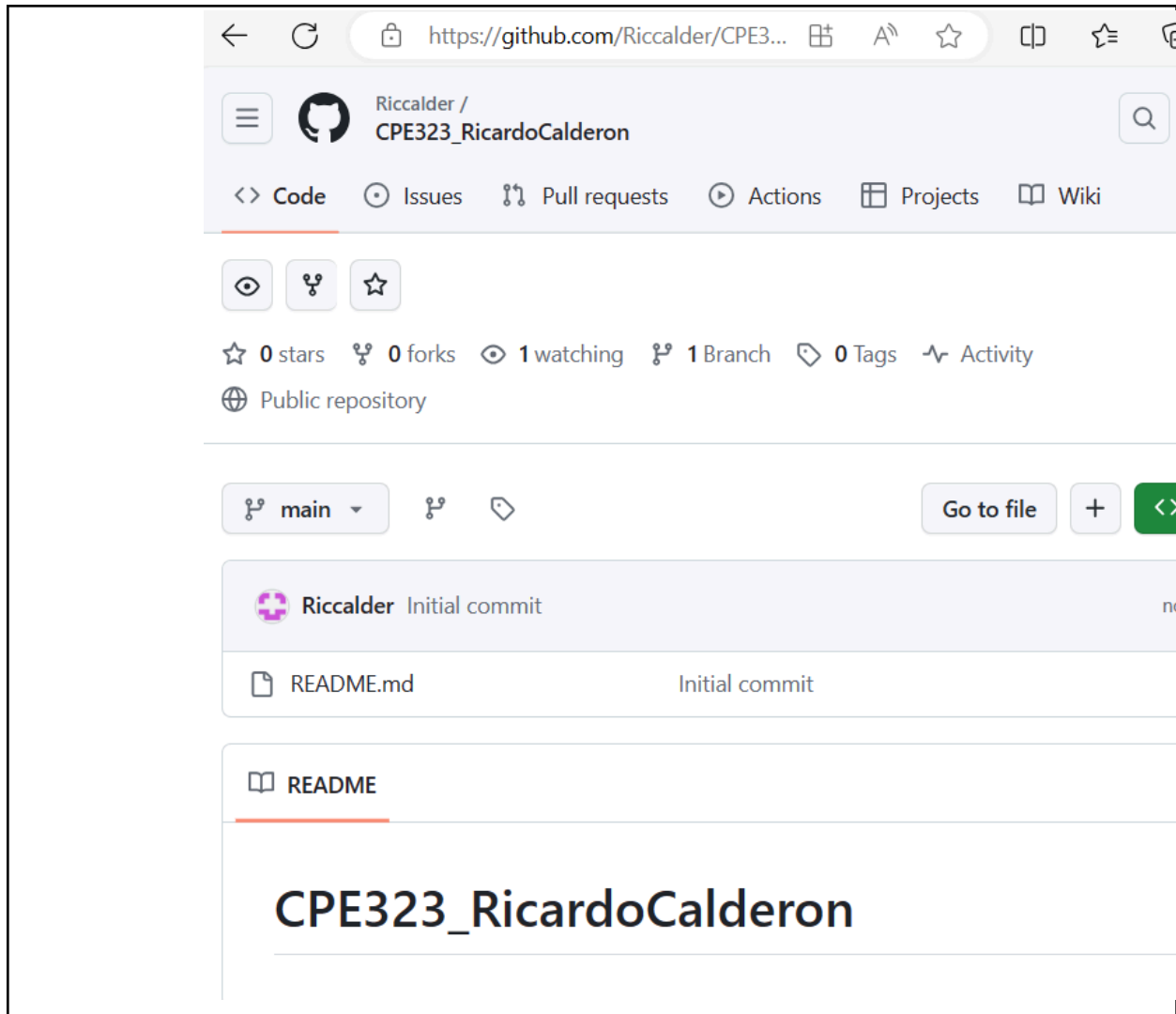
[Forgot password?](#)

Sign in

[Sign in with a passkey](#)

New to GitHub? [Create an account](#)

- a. Create a new repository and name it as CPE232_yourname. Check Add a README file and click Create repository.



The screenshot displays a GitHub repository page for the user Riccalder, specifically the repository named CPE323_RicardoCalderon. The browser's address bar shows the URL https://github.com/Riccalder/CPE3... The repository page includes a navigation bar with tabs for Code, Issues, Pull requests, Actions, Projects, and Wiki. Below the navigation bar, there are icons for repository visibility, a fork icon, and a star icon. The repository statistics are shown as 0 stars, 0 forks, 1 watching, 1 Branch, and 0 Tags. The repository is marked as a Public repository. The main content area shows the repository's main branch, main, with a 'Go to file' button. Below this, the commit history is displayed, showing an initial commit by Riccalder. The README file is selected, and the repository name, CPE323_RicardoCalderon, is prominently displayed.

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.

Add new SSH Key

Title

CPE232

Key type

Authentication Key

Key

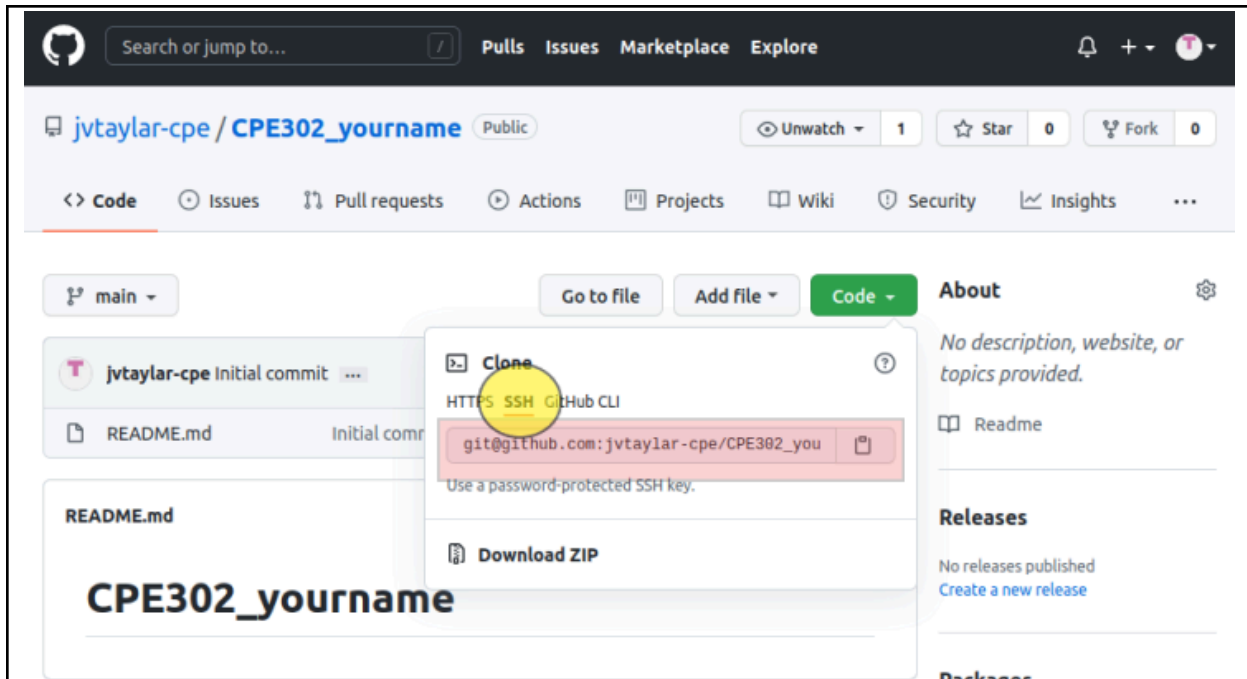
Begins with 'ssh-rsa', 'ecdsa-sha2-nistp256', 'ecdsa-sha2-nistp384', 'ecdsa-sha2-nistp521', 'ssh-ed25519', 'sk-ecdsa-sha2-nistp256' or 'sk-ssh-ed25519@openssh.com'

Add SSH 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.

```
calderon@Workstation: ~/.ssh$ cat id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQgQDDNap4aEAQFX6prtLs1JqSxGrP9ivDRCZFjy9E24dmpw4
6uADfHAGo+HnAVKZIhccDQmZWj1hFrJ9rON0C7SCb593mP/8LyVDp0ivbc8FreTW/n/Z3xtkPNsLA+UgJtg
S9/e/IQJgp8NGQwelBBY6qG87GbIbz2H5QQv9P/C1a6235DYqa4uFKWjBrKDvfNFZ12s2v67g8P9ZNvJmRu
Su5TRiPKizHthytx8TubUxOzaaDnBRdBbAjEsSnllm0L2pSct2nU0tbIeOdoKMaRwM/xgpxHfV9iuntm00L
faGiX+rz09a0zjiQ0UXTagsElr19fYIPzydG4BT3y3DHC83FfHQADGKjnD6FiqN+E5R80VmEs1F8KyKpha
aYLp1hJqUDZwhMXAQZdb7umxqSI3/tJVFkwCoNYDZkcXnVHSS8rDbClzyQiqUwaEsGEa5xs= calderon@W
calderon@Workstation: ~/.ssh$
```

- 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.
- 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?

With Ansible you can automate tasks such as installing and configuring software, managing packages and updates, creating and managing user accounts, managing files and directories, configuring network settings and firewall rules, and much more.

4. How important is the inventory file?

In Ansible, the inventory file is essential since it specifies the list of distant hosts that you can trust Ansible. This file includes remote connection details. SSH users, SSH keys, and IP addresses or hostnames of the servers and further parameters pertaining to the connection. Hosting is arranged via the inventory file, to facilitate targeting particular server groups with Ansible by grouping them into playbooks and assignments.

Conclusions/Learnings:

I learnt to set up SSH key-based authentication and the Git version control system, both of which are essential tools for development workflows. SSH key-based authentication provides added security when accessing remote servers, while Git streamlines collaboration among team members working on the same codebase. By implementing these tools, development teams can ensure both security and productivity in their workflows.