

Name: Josiah Miguel B. Darroca	Date Performed: August 22, 2023
Course/Section: CPE31S4	Date Submitted: August 29, 2023
Instructor: Engr. Jonathan Taylar	Semester and SY: 1st Sem 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*.

```
joshxh@managenode:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/joshxh/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/joshxh/.ssh/id_rsa
Your public key has been saved in /home/joshxh/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:rxN9xCtsCzQ+9hunluGxDAWvPoypSTeSOWLq7ZCtu58 joshxh@managenode
The key's randomart image is:
+---[RSA 3072]-----+
|
|      .
|      o o
|     o o o
|    oS* . .
|   o o 0.B o
|  oo.* o* Xo0.
| o+o.=o.=.0+
| o==Eo. .+o.
+---[SHA256]-----+
joshxh@managenode:~$
```

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.

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

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*.

```
joshxh@managenode:~$ ls -la .ssh
total 16
drwx----- 2 joshxh joshxh 4096 Aug 29 00:36 .
drwxr-x--- 16 joshxh joshxh 4096 Aug 29 00:26 ..
-rw----- 1 joshxh joshxh 2602 Aug 29 00:36 id_rsa
-rw-r--r-- 1 joshxh joshxh 571 Aug 29 00:36 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.

```
joshxh@managenode:~$ 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
joshxh@managenode:~$
```

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

```
joshxh@managenode:~$ ssh-copy-id -i ~/.ssh/id_rsa joshxh@controlnode1
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/joshxh/.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
joshxh@controlnode1's password:
```

Number of key(s) added: 1

Now try logging into the machine, with: "ssh 'joshxh@controlnode1'" and check to make sure that only the key(s) you wanted were added.

```
joshxh@managenode:~$ ssh-copy-id -i ~/.ssh/id_rsa joshxh@controlnode2
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/joshxh/.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
joshxh@controlnode2's password:
```

Number of key(s) added: 1

Now try logging into the machine, with: "ssh 'joshxh@controlnode2'" and check to make sure that only the key(s) you wanted were added.

```
joshxh@managenode:~$
```

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?

```
joshxh@managenode:~$ ssh joshxh@controlnode1
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 6.2.0-31-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

Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your
Internet connection or proxy settings

Last login: Tue Aug 29 01:43:56 2023 from 192.168.56.101
joshxh@controlnode1:~$
```

Reflections:

Answer the following:

1. How will you describe the ssh-program? What does it do?
 - The ssh-program in the way I see it is that, it is a tool used for secure remote access and also, a tool to manage various Unix systems just like Ubuntu. The ssh-program provides a good way of interacting with machines which can be remotely accessed and it is also good for privacy's sake since all of its data that is transmitted is protected with integrity.
2. How do you know that you already installed the public key to the remote servers?
 - You can check to verify if your public key has been installed on the server by logging onto the remote server and looking in the `authorized_keys` file. The `authorized_keys` file is located in the `/.ssh` directory on the remote server.

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

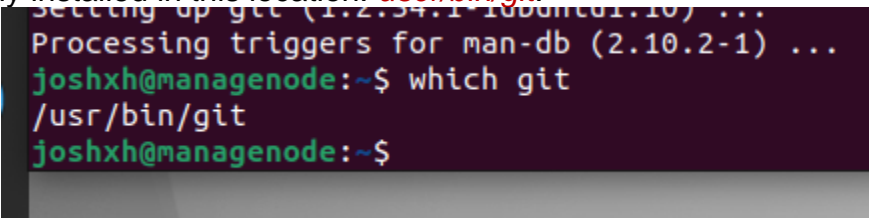
At the heart of GitHub is an open-source version control system (VCS) called Git. Git is responsible for everything GitHub-related that happens locally on your computer. To use Git on the command line, you'll need to download, install, and configure Git on your computer. You can also install GitHub CLI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

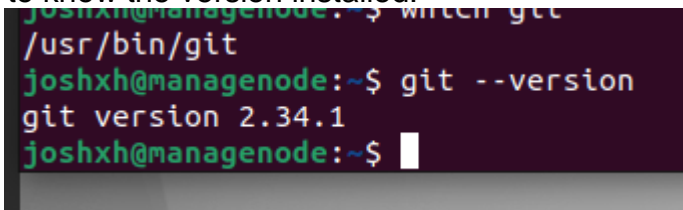
Task 3: Set up the Git Repository

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*.



```
Setting up gcc (4:4.8.3-1ubuntu1) ...  
Processing triggers for man-db (2.10.2-1) ...  
joshxh@managenode:~$ which git  
/usr/bin/git  
joshxh@managenode:~$
```

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



```
joshxh@managenode:~$ which git  
/usr/bin/git  
joshxh@managenode:~$ git --version  
git version 2.34.1  
joshxh@managenode:~$
```

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.

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:rxN9xCtsCzQ+9hun1u6xDAWvPoypSTeS0WLq7ZCtu58

Added on Aug 29, 2023

Never used — Read/write

[Delete](#)

- 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.


```

joshxh@managenode:~$ git clone git@github.com:joshiah0521/CPE232_DARROCA.git
Cloning into 'CPE232_DARROCA'...
The authenticity of host 'github.com (192.30.255.112)' can't be established.
ED25519 key fingerprint is SHA256:+DiY3wvvV6TuJJhbpZisF/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.
joshxh@managenode:~$ S

```

- 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.

```

joshxh@managenode:~$ ls
CPE232_DARROCA  Documents  Music      Public  Templates
Desktop         Downloads  Pictures   snap    Videos
joshxh@managenode:~$ cd /CPE232_DARROCA
bash: cd: /CPE232_DARROCA: No such file or directory
joshxh@managenode:~$ cd CPE232_DARROCA
joshxh@managenode:~/CPE232_DARROCA$ ls
README.md
joshxh@managenode:~/CPE232_DARROCA$

```

- 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`

```

joshxh@managenode:~/CPE232_DARROCA$ ls
README.md
joshxh@managenode:~/CPE232_DARROCA$ git config --global user.name "Josiah Darroca"
joshxh@managenode:~/CPE232_DARROCA$ git config --global user.email "jmbdarroca@tip.edu.ph"
joshxh@managenode:~/CPE232_DARROCA$ cat ~/.gitconfig
[user]
    name = Josiah Darroca
    email = jmbdarroca@tip.edu.ph
joshxh@managenode:~/CPE232_DARROCA$

```

- 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.

```
joshxh@manageno
GNU nano 6.2
# CPE232_DARROCA
# CPE232_DARROCA GITHUB REPOSITORY
```

- 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?

```
joshxh@managenode:~/CPE232_DARROCA$ 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")
joshxh@managenode:~/CPE232_DARROCA$
```

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

```
no changes added to commit (use "git add" and/or "git commit -a")
joshxh@managenode:~/CPE232_DARROCA$ git add README.md
joshxh@managenode:~/CPE232_DARROCA$
```

- 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.

```
joshxh@managenode:~/CPE232_DARROCA$ git add README.md
joshxh@managenode:~/CPE232_DARROCA$ git commit -m "sheeeesh"
[main cd4bfee] sheeeesh
 1 file changed, 4 insertions(+), 1 deletion(-)
joshxh@managenode:~/CPE232_DARROCA$
```

- 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*.


```
joshxh@managenode:~/CPE232_DARROCA$ git push origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 285 bytes | 285.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:josiah0521/CPE232_DARROCA.git
   ed000ff..cd4bfee  main -> main
joshxh@managenode:~/CPE232_DARROCA$
```

- 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 displays the GitHub web interface for the repository 'josiah0521 / CPE232_DARROCA'. The top navigation bar includes links for Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. The left sidebar shows the 'Files' tab with a search bar and a list of files, including 'README.md'. The main content area shows the 'CPE232_DARROCA / README.md' file view. The commit history shows a single commit by 'Josiah Darroca' with the message 'sheeeesh'. The file content is displayed in a 'Preview' mode, showing the text 'CPE232_DARROCA' and 'CPE232_DARROCA GITHUB REPOSITORY'.

Reflections:

Answer the following:

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

- One of the things we've done is by altering the contents of the README.md file inside the github repository. We modified its content by utilizing the connection of the repository to the machine's terminal itself. Through the use of sudo nano, we have gained access to modify the file and we have tested its results by comparing the output of both files from both terminal and github website,

4. How important is the inventory file?

- An inventory file lists the hosts or devices that Ansible can control, hence it is crucial. Among other things, it can define host variable groups. Additionally, it tells Ansible which hosts to connect to and use for job execution.

Conclusions/Learnings:

- In this activity, I have learned how to work with ssh, ssh-keys, and also github. Not only that, because of this activity I have learned how to: Configure remote and local machine to connect via SSH using a KEY instead of using a password. Create a public key and private key. Verify connectivity. Setup Git Repository using local and remote repositories. Configure and Run ad hoc commands from local machine to remote servers. These things that I have learned from finishing this activity helped me realize that there I am miles away from mastering systems administrations. Despite all that, I am glad to learn about all these things and I hope I can learn more in the future.