

<b>Name: Ilagan, Carlo Hideki D.</b>	<b>Date Performed: 9/8/2024</b>
<b>Course/Section: CPE31S2</b>	<b>Date Submitted: 9/11/2024</b>
<b>Instructor: Mr. Robin Valenzuela</b>	<b>Semester and SY: 1<sup>st</sup> Sem/2024 – 2025</b>

### **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*.

```
hideki@workstation: ~  
hideki@workstation:~$ ssh-keygen  
Generating public/private ed25519 key pair.  
Enter file in which to save the key (/home/hideki/.ssh/id_ed25519):  
Enter passphrase (empty for no passphrase):  
Enter same passphrase again:  
Your identification has been saved in /home/hideki/.ssh/id_ed25519  
Your public key has been saved in /home/hideki/.ssh/id_ed25519.pub  
The key fingerprint is:  
SHA256:E+8MZVIDpbaUsiIundS0+hrJ8kxgW/oYP8dSQzqNHos hideki@workstation  
The key's randomart image is:  
+--[ED25519 256]--+  
|      .o+      |  
|      + .      |  
|    . . B o     |  
|    o o = B     |  
|..o.O . S .    |  
|. *+X = =       |  
|o+% * . o       |  
| E=B 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.
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.

```
hideki@workstation:~$ ssh-keygen -t rsa -b 4096  
Generating public/private rsa key pair.  
Enter file in which to save the key (/home/hideki/.ssh/id_rsa):  
Enter passphrase (empty for no passphrase):  
Enter same passphrase again:  
Your identification has been saved in /home/hideki/.ssh/id_rsa  
Your public key has been saved in /home/hideki/.ssh/id_rsa.pub  
The key fingerprint is:  
SHA256:qW5u1Csvgfv6DyNuPziWGOrJ9QQA7gSiDbrrpCkt+S+c hideki@workstation  
The key's randomart image is:  
+----[RSA 4096]-----+  
|=                  |  
|Bo                 |  
|o+.                |  
|oo.                |  
|o. . . .S          |  
|. . o o..          |  
|. + =.B+. .        |  
|B.=o%o*o.          |  
|+=.++E**o          |  
+-----[SHA256]-----+
```

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`.

```
hideki@workstation:~$ ls -ls .ssh
total 24
0 -rw----- 1 hideki hideki    0 Aug 25 10:13 authorized_keys
4 -rw----- 1 hideki hideki  411 Sep  8 08:21 id_ed25519
4 -rw-r--r-- 1 hideki hideki  100 Sep  8 08:21 id_ed25519.pub
4 -rw----- 1 hideki hideki 3381 Sep  8 08:21 id_rsa
4 -rw-r--r-- 1 hideki hideki   744 Sep  8 08:21 id_rsa.pub
4 -rw----- 1 hideki hideki 1404 Aug 25 11:42 known_hosts
4 -rw-r--r-- 1 hideki hideki   142 Aug 25 11:11 known_hosts.old
hideki@workstation:~$
```

## 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`
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.

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

Number of key(s) added: 1

Now try logging into the machine, with:  "ssh 'hideki@server1'"
and check to make sure that only the key(s) you wanted were added.
```

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?
  - The local machine or the main workstation ssh works with the Server 1, the connection did not ask for password since server 1 and local machine had the same SSH keygen.

```
hideki@workstation:~$ ssh hideki@server1
Welcome to Ubuntu 24.04 LTS (GNU/Linux 6.8.0-41-generic x86_64)

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

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
Last login: Sun Aug 25 11:41:16 2024 from 192.168.56.101
hideki@server1:~$
```

### Reflections:

Answer the following:

1. How will you describe the ssh-program? What does it do?
  - SSH program is an authentication process for a server and local machine to connect with each other without using any password. This program can or will automatically provide keygen that will be shared throughout the servers if needed with the right queries.
2. How do you know that you already installed the public key to the remote servers?
  - If the server has the .pub file when using the command `ls -ls .ssh` since .pub stands for public.

## 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

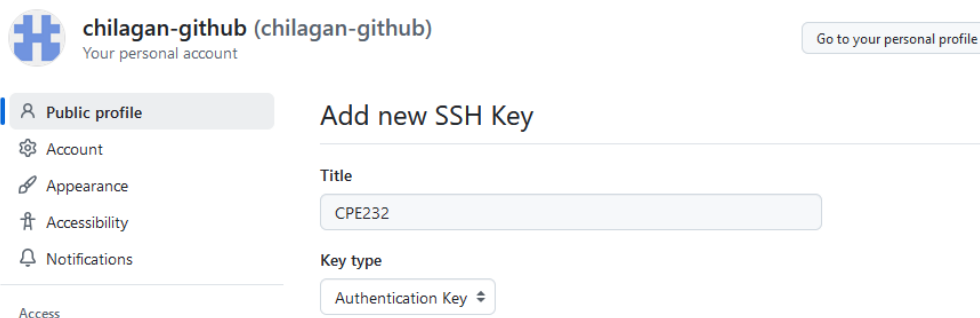
1. On the local machine, verify the version of your git using the command *which git*. If a directory of git is displayed, then you don't need to install git. Otherwise, to install git, use the following command: *sudo apt install git*
2. After the installation, issue the command *which git* again. The directory of git is usually installed in this location: *user/bin/git*.

```
hideki@workstation:~$ which git
/usr/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.

```
hideki@workstation:~$ git -v
git version 2.43.0
```

4. Using the browser in the local machine, go to [www.github.com](https://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.

## 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

SHA256:qW5u1Csvgfv6DyNuPziWG0rJ9QQA7gSiDbrpCkt+S+c

Added on Sep 8, 2024

Never used — Read/write

Delete

Search or jump to... Pulls Issues Marketplace Explore

jvtaylor-cpe / CPE302\_yourname Public

Unwatch 1 Star 0 Fork 0

Code Issues Pull requests Actions Projects Wiki Security Insights

main

Go to file Add file Code About

Clone

HTTPS SSH GitHub CLI

git@github.com:jvtaylor-cpe/CPE302\_you

Use a password-protected SSH key.

Download ZIP

Initial commit

README.md

Initial commit

README.md

CPE302\_yourname

No description, website, or topics provided.

Readme

Releases

No releases published

Create a new release

Packages

- 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.
- 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`.

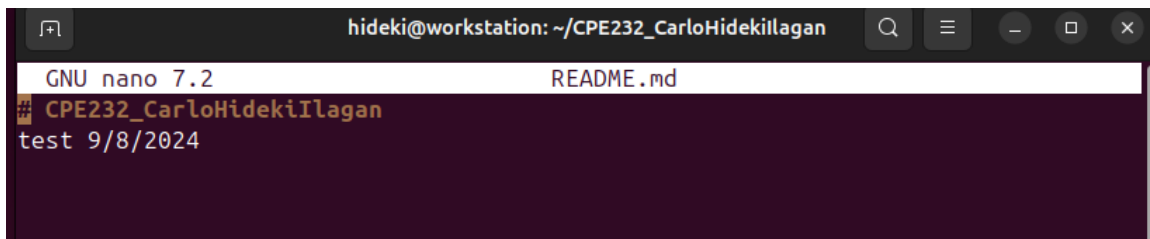
```
hideki@workstation:~$ git clone git@github.com:chilagan-github/CPE232_CarloHidekiIlagan.git
Cloning into 'CPE232_CarloHidekiIlagan'...
The authenticity of host 'github.com (20.205.243.166)' can't be established.
ED25519 key fingerprint is SHA256:+DiY3wvV6TuJJhbpZisF/zLDA0zPMSvHdkr4UvCOqU.
This key 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 (from 0)
Receiving objects: 100% (3/3), done.
hideki@workstation:~$ ls
CPE232_CarloHidekiIlagan  Documents  Music      Public  Templates
Desktop                  Downloads  Pictures   snap    Videos
hideki@workstation:~$ ls CPE232_CarloHidekiIlagan
README.md
hideki@workstation:~$
```

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`

```
hideki@workstation:~/CPE232_CarloHidekiIlagan$ git config --global user.name "hideki"
hideki@workstation:~/CPE232_CarloHidekiIlagan$ git config --global user.email "qchdilagan@tip.edu.ph"
hideki@workstation:~/CPE232_CarloHidekiIlagan$ cat ~/.gitconfig
[user]
    name = hideki
    email = qchdilagan@tip.edu.ph
```

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.



```
hideki@workstation: ~/CPE232_CarloHidekiIlagan
GNU nano 7.2                                README.md
# CPE232_CarloHidekiIlagan
test 9/8/2024
```

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?



- Your branch is up to date with 'origin/main'.

```
hideki@workstation:~/CPE232_CarloHidekiIlagan$ 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
```

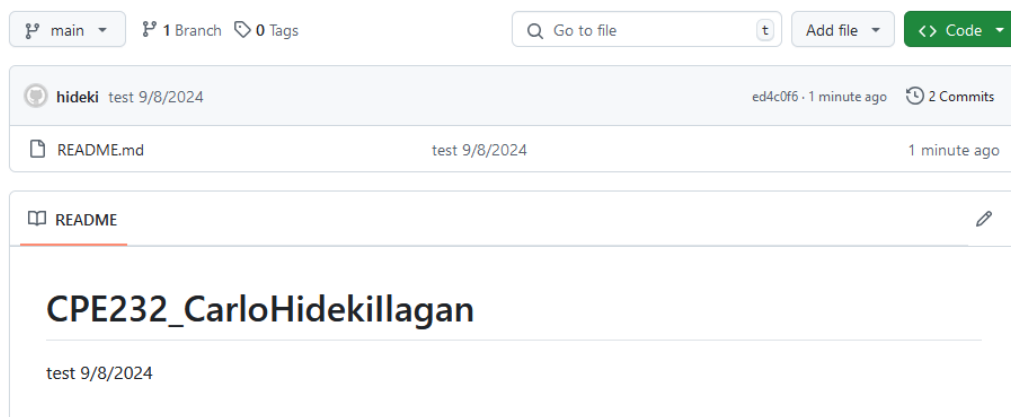
- j. Use the command *git add README.md* to add the file into the staging area.
- k. Use the *git commit -m "your message"* to create a snapshot of the staged changes along the timeline of the Git projects history. The use of this command is required to select the changes that will be staged for the 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*.

```
hideki@workstation:~/CPE232_CarloHidekiIlagan$ git commit -m "test 9/8/2024"
[main ed4c0f6] test 9/8/2024
 1 file changed, 2 insertions(+), 1 deletion(-)
hideki@workstation:~/CPE232_CarloHidekiIlagan$ git push origin/main
fatal: 'origin/main' does not appear to be a git repository
fatal: Could not read from remote repository.

Please make sure you have the correct access rights
and the repository exists.
hideki@workstation:~/CPE232_CarloHidekiIlagan$ git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Writing objects: 100% (3/3), 281 bytes | 281.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:chilagan-github/CPE232_CarloHidekiIlagan.git
 a61751a..ed4c0f6  main -> main
hideki@workstation:~/CPE232_CarloHidekiIlagan$
```

- 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?
  - In this activity, we utilize pairing of SSH keys from local machine to servers. I was able to access Server1 using SSH command. I was also able to connect the local machine in my GitHub account in order for me to modify files from the GitHub using Ubuntu.
4. How important is the inventory file?
  - Inventory file is important since can be used to define the SSH key to use for authentication when connecting to host.

### Conclusions/Learnings:

In this activity, I was able to learn the importance, the use, and the commands that an SSH can utilize. Where SSH is a program that can set-up key based authentication for servers and GitHub. There are new commands tackled as the activity progress, for example, the command `ls -la .ssh` that was used to check what are the keygens inside the local machine. In order to copy these ssh files into the servers, `ssh-copy-id -i ~/.ssh/id_rsa` was queried. Although there were confusions while performing the activity, I had fun, and I also gain additional knowledge in this course.