

Name: Reyes, Alexzander J.	Date Performed: 15/08/2025
Course/Section: CPE212 -CPE31S2	Date Submitted: 15/08/2025
Instructor: Engr. Robin Valenzuela	Semester and SY: 1st sem & 2025-2026

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*.

```
vboxuser@Workstation: ~
File Edit View Search Terminal Help
vboxuser@Workstation:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/vboxuser/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Passphrases do not match. Try again.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/vboxuser/.ssh/id_rsa.
Your public key has been saved in /home/vboxuser/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:1IjcdM0UZNYVON+ttWyTRfNlX/uYqkCW7tBzeVvIXw4 vboxuser@Workstation
The key's randomart image is:
+---[RSA 2048]----+
|       . =*o.oo|
|   . + + ooo..=|
|   o + .   ooX|
|       . .oB|
|       S     .==|
|   =   o .+*.|
|   . = o +.E 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.
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.

```
vboxuser@Workstation:~$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/vboxuser/.ssh/id_rsa):
/home/vboxuser/.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/vboxuser/.ssh/id_rsa.
Your public key has been saved in /home/vboxuser/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:+xUpoBYuH5pU0+BCzrXZ6kVTslUDP90e+Tpx9tzk3kU vboxuser@Workstation
The key's randomart image is:
+---[RSA 4096]---+
|          .... |
|          . . . |
| . o + + + .. |
| + o B 0 . . +o|
| + * % S . o oE|
| o 0 + . . . =|
| + o . . B=|
| . . . . .B|
| . . . . oo|
+---[SHA256]---+
```

- 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.

```
vboxuser@Workstation:~$ ls -la .ssh
total 20
drwx----- 2 vboxuser vboxuser 4096 Aug 15 16:57 .
drwxr-xr-x 15 vboxuser vboxuser 4096 Aug 15 16:50 ..
-rw------- 1 vboxuser vboxuser 3243 Aug 15 16:58 id_rsa
-rw-r--r-- 1 vboxuser vboxuser 746 Aug 15 16:58 id_rsa.pub
-rw-r--r-- 1 vboxuser vboxuser 888 Aug 8 18:35 known_hosts
```

Task 2: Copying the Public Key to the remote servers

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

```
vboxuser@Workstation:~$ ssh-copy-id -i ~/.ssh/id_rsa vboxuser@192.168.56.105
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/vboxuser/.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
vboxuser@192.168.56.105's password:

Number of key(s) added: 1

Now try logging into the machine, with: "ssh 'vboxuser@192.168.56.105'"
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?

```
vboxuser@Workstation:~$ ssh vboxuser@192.168.56.105
Welcome to Ubuntu 18.04.6 LTS (GNU/Linux 5.4.0-150-generic x86_64)

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

Expanded Security Maintenance for Infrastructure is not enabled.

0 updates can be applied immediately.

Enable ESM Infra to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

New release '20.04.6 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Your Hardware Enablement Stack (HWE) is supported until April 2023.
Last login: Fri Aug  8 18:33:33 2025 from 192.168.56.107
```

Reflections:

Answer the following:

1. How will you describe the ssh-program? What does it do?
 - The ssh program is a secure protocol used to remotely access and manage computers over a network. It provides encrypted communication between the client and the server, ensuring data privacy and security. SSH allows users to log into remote machines, execute commands, and transfer files securely without exposing sensitive information like passwords to interception. It replaces older, less secure protocols by using encryption and authentication methods such as password or key-based authentication.
2. How do you know that you already installed the public key to the remote servers?
 - You can confirm that the public key has been successfully installed on the remote server when you can SSH into the server without being prompted for a password. This means that the server recognizes the public key in its `~/.ssh/authorized_keys` file and allows access based on the private key stored on the local machine. Additionally, running `ssh-copy-id` and verifying the presence of the public key inside the `authorized_keys` file on the remote server confirms that the key has been correctly installed.

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**

```
root@Workstation:/home/vboxuser# sudo apt install git
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following package was automatically installed and is no longer required:
  libllvm7
Use 'sudo apt autoremove' to remove it.
The following additional packages will be installed:
  git-man liberror-perl
Suggested packages:
  git-daemon-run | git-daemon-sysvinit git-doc git-el 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 0 not upgraded.
Need to get 4,817 kB of archives.
After this operation, 34.3 MB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 http://ph.archive.ubuntu.com/ubuntu bionic/main amd64 liberror-perl all 0
.17025-1 [22.8 kB]
Get:2 http://ph.archive.ubuntu.com/ubuntu bionic-updates/main amd64 git-man all
 1:2.17.1-1ubuntu0.18 [804 kB]
Get:3 http://ph.archive.ubuntu.com/ubuntu bionic-updates/main amd64 git amd64 1
  :2.17.1-1ubuntu0.18 [3,990 kB]
Fetched 4.817 kB in 8s (582 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**.

```
root@Workstation:/home/vboxuser# 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.

```
root@Workstation:/home/vboxuser# 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_yourusername. Check Add a README file and click Create repository.

The screenshot shows a GitHub repository page for 'CPE232_Reyes_Alexzander'. The repository is public. It displays a commit from 'AlexzanderReyes' creating a README.md file 10 minutes ago. It also shows a LICENSE file and an initial commit. The README file is open, showing the Apache-2.0 license. Below the repository details, there is a section titled 'ACTIVITY 2'.

- b. Create a new SSH key on GitHub. Go to 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 key, 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:+xUpoBYuH5pU0+BCzrXZ6kVTs1UdP90e+Tpx9tzk3kU

Added on Aug 15, 2025

Never used — Read/write

Delete

Check out our guide to [connecting to GitHub using 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.

The screenshot shows a GitHub repository page for 'jvtaylor-cpe / CPE302_yourname'. The 'Code' dropdown menu is open, and the 'SSH' option is highlighted with a yellow circle. The URL 'git@github.com:jvtaylor-cpe/CPE302_yourname.git' is displayed in the dropdown menu. The repository has 1 pull request, 0 stars, and 0 forks.

- 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.

```
root@Workstation:/home/vboxuser# git clone git@github.com:AlexzanderReyes/CPE232_Reyes_Alexzander.git
Cloning into 'CPE232_Reyes_Alexzander'...
The authenticity of host 'github.com (4.237.22.38)' can't be established.
ECDSA key fingerprint is SHA256:p2QAMXNIC1TJYWeI0ttrVc98/R1BUFWu3/LiyKgUfQM.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'github.com,4.237.22.38' (ECDSA) to the list of known hosts.
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (3/3), 4.70 KiB | 4.70 MiB/s, 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.

```
root@Workstation:/home/vboxuser# ls
a.out          Documents      Music    Templates
CPE232_Reyes_Alexzander Downloads    Pictures   Videos
Desktop        examples.desktop Public
```



```
vboxuser@Workstation:~/CPE232_Reyes_Alexzander$ ls
LICENSE  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`

```
vboxuser@Workstation:~/CPE232_Reyes_Alexzander$ git config --global user.name "Alexzander Reyes"
vboxuser@Workstation:~/CPE232_Reyes_Alexzander$ git config --global user.email qajreyes01@tip.edu.ph
vboxuser@Workstation:~/CPE232_Reyes_Alexzander$ cat ~/.gitconfig
[user]
  name = Alexzander Reyes
  email = qajreyes01@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.

File Edit View Search Terminal Help

GNU nano 2.9.3

README.md

CPE232_REYES

- 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?

```
vboxuser@Workstation:~/CPE232_Reyes_Alexzander$ 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.

```
[no changes added to commit (use "git add" and/or "git commit -a")]
vboxuser@Workstation:~/CPE232_Reyes_Alexzander$ 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.

```
vboxuser@Workstation:~/CPE232_Reyes_Alexzander$ git commit -m "YAHUUUU!"
[main a2e79d1] YAHUUUU!
 1 file changed, 1 insertion(+), 1 deletion(-)
```

- 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*.

```
vboxuser@Workstation:~/CPE232_Reyes_Alexzander$ git push origin main
Counting objects: 3, done.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 297 bytes | 297.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To github.com:AlexzanderReyes/CPE232_Reyes_Alexzander.git
 54cfa3f..a2e79d1  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.

The screenshot shows a GitHub repository interface. At the top, it displays the repository name 'CPE232 Reyes_Alexzander' and its status as 'Public'. To the right are buttons for 'Pin', 'Watch' (with a count of 0), and a dropdown menu. Below this is a navigation bar with 'main' (a dropdown arrow), a search icon, and a refresh icon. To the right of the search are buttons for 'Go to file', '+', and 'Code' (with a dropdown arrow). The main area shows a commit history:

File	Commit Message	Time Ago
LICENSE	Initial commit	32 minutes ago
README.md	YAHUUUU!	1 minute ago

Below the commit history is the content of the README file:

```
Apache-2.0 license
CPE232_REYES
```

Reflections:

Answer the following:

3. What sort of things have we so far done to the remote servers using ansible commands?
 - Using Ansible commands, we have automated tasks such as configuring network settings, deploying SSH keys for passwordless authentication, installing necessary software packages, and managing configurations across multiple remote servers efficiently. Ansible allows us to perform ad hoc commands and execute playbooks that ensure consistency, reduce manual errors, and save time by orchestrating complex tasks on several machines simultaneously.
4. How important is the inventory file?
 - The inventory file is crucial in Ansible as it defines the list of remote hosts and groups that Ansible will manage. It tells Ansible where the commands and playbooks should be executed. Without a properly configured inventory file,

Ansible wouldn't know which machines to connect to. It also helps organize hosts into groups, enabling targeted deployments and configurations, which is essential for managing large and diverse environments effectively.

Conclusions/Learnings:

- In this activity, I learned how to create and use SSH keys for secure, passwordless login to remote servers. This method improves security and makes remote access easier and faster. I also understood how to set up Git locally and connect it with a remote GitHub repository using SSH keys. This allows for secure and efficient version control and collaboration. Additionally, I saw how automation tools like Ansible simplify managing multiple remote servers by running commands and configurations all at once. The inventory file is very important because it tells Ansible which servers to manage. Overall, this activity helped me understand secure remote access, version control, and basic automation, which are essential skills for managing servers and development projects.