

Name: Bueno, Joshua C.	Date Performed: 08/15/25
Course/Section: CPE31S2	Date Submitted: 08/15/25
Instructor: Engr. Robin Valenzuela	Semester and SY: 1st Semester
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`.

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.

```
joshuabueno@workstation:~$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/joshuabueno/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/joshuabueno/.ssh/id_rsa
Your public key has been saved in /home/joshuabueno/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:dNl8cLzvCWKpoBBpWknXtXd4GPQH9U96Qx7Cfv2lIP9E joshuabueno@workstation
The key's randomart image is:
+---[RSA 4096]---+
|      . ...o.+=+ |
|      . . . .+=o*o+|
|      . +      ..o+o+o=.|
|      *      . .. o.o..|
|      + .      S      . oo|
|      . .      + o .E|
|      . . . 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`.

```
joshuabueno@workstation:~$ ls -la ~/.ssh
total 24
drwx----- 2 joshuabueno joshuabueno 4096 Aug 15 09:12 .
drwxr-x--- 15 joshuabueno joshuabueno 4096 Aug  8 09:01 ..
-rw----- 1 joshuabueno joshuabueno    0 Aug  8 08:50 authorized_keys
-rw----- 1 joshuabueno joshuabueno 3389 Aug 15 09:12 id_rsa
-rw-r--r-- 1 joshuabueno joshuabueno  749 Aug 15 09:12 id_rsa.pub
-rw----- 1 joshuabueno joshuabueno 1262 Aug  8 10:21 known_hosts
-rw-r--r-- 1 joshuabueno joshuabueno  142 Aug  8 10:15 known_hosts.old
```

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.

```
joshuabueno@workstation:~$ ssh-copy-id
Usage: /usr/bin/ssh-copy-id [-h|-?|-f|-n|-s|-x] [-i [identity_file]] [-p port] [-F alternative_ssh_config_file] [-t target_path] [[-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
    -x: debug       -- enables -x in this shell, for debugging
    -h|-?: print this help
```

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

```
joshuabueno@workstation:~$ ssh-copy-id -i ~/.ssh/id_rsa joshuabueno@workstation
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/joshuabueno/.ssh/id_rsa.pub"
The authenticity of host 'workstation (127.0.0.1)' can't be established.
ED25519 key fingerprint is SHA256:dxuj1z+HgOqKys/2mvuQLvDbR7PEkevsRZBciz+ITTQ.
This host key is known by the following other names/addresses:
  ~/.ssh/known_hosts:1: [hashed name]
  ~/.ssh/known_hosts:4: [hashed name]
  ~/.ssh/known_hosts:5: [hashed name]
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
joshuabueno@workstation's password:

Number of key(s) added: 1

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

```
joshuabueno@workstation:~$ ssh joshuabueno@server1
Welcome to Ubuntu 24.04.3 LTS (GNU/Linux 6.14.0-27-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

Last login: Fri Aug 15 09:34:01 2025 from 127.0.0.1
```

-The server no longer asks for a password because it identifies my public key, and my local machine confirms it holds the corresponding private key.

Reflections:

Answer the following:

1. How will you describe the ssh-program? What does it do?
-SSH is a secure protocol to remotely access and manage servers over encrypted connections.
2. How do you know that you already installed the public key to the remote servers?
-If you can SSH into the server without a password prompt, your public key is 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

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*

```
joshuabueno@workstation:~$ sudo apt install git
[sudo] password for joshuabueno:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  libgl1-amber-dri libglapi-mesa
Use 'sudo apt autoremove' to remove them.
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 1 not upgraded.
Need to get 4,806 kB of archives.
After this operation, 24.5 MB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 http://ph.archive.ubuntu.com/ubuntu noble/main amd64 liberror-perl all 0.1
7029-2 [25.6 kB]
Get:2 http://ph.archive.ubuntu.com/ubuntu noble-updates/main amd64 git-man all 1
:2.43.0-1ubuntu7.3 [1,100 kB]
Get:3 http://ph.archive.ubuntu.com/ubuntu noble-updates/main amd64 git amd64 1:2
```

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

```
joshuabueno@workstation:~$ which git
/usr/bin/git
```

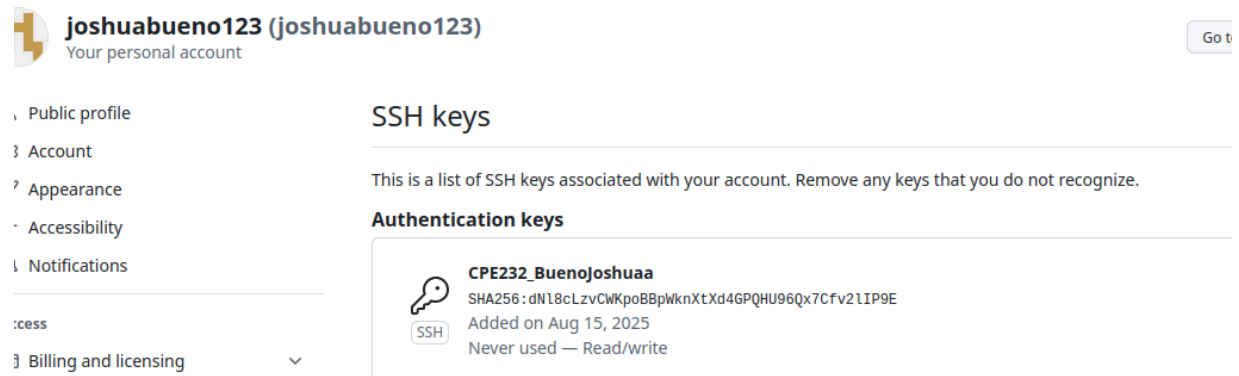
3. The version of git installed in your device is the latest. Try issuing the command *git --version* to know the version installed.

```
joshuabueno@workstation:~$ git --version
git version 2.43.0
```

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.

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

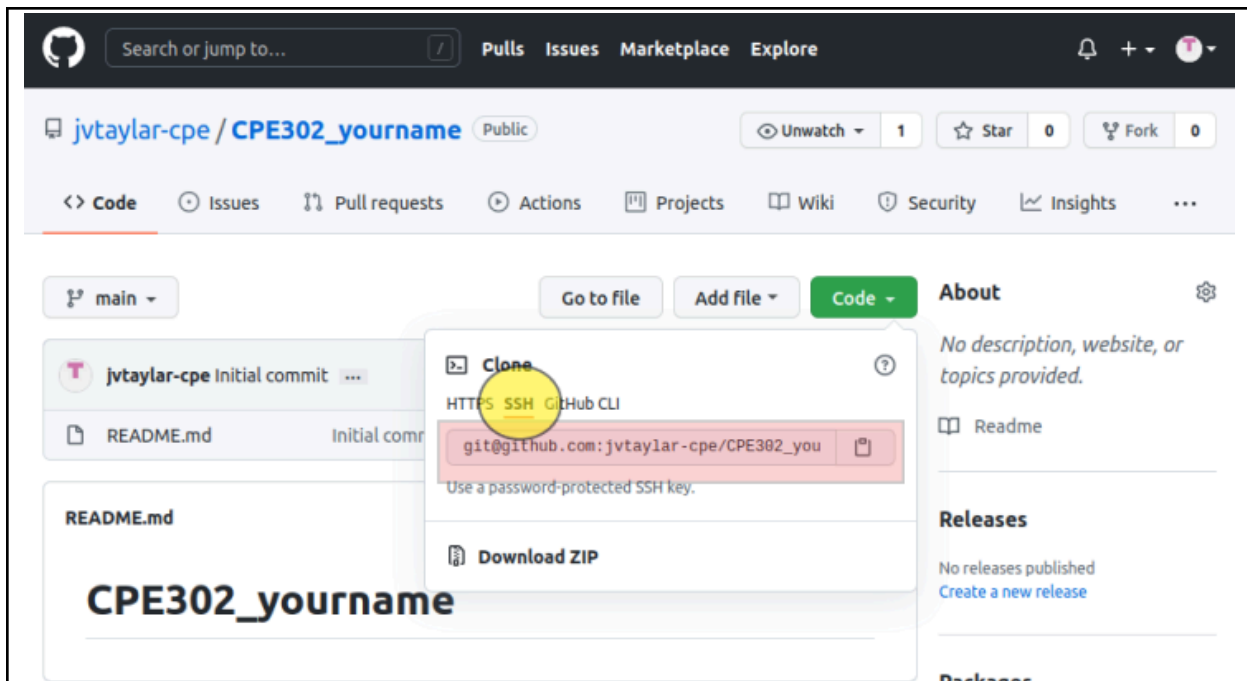


The screenshot shows the GitHub profile settings for user **joshuabueno123**. The left sidebar contains navigation links: Public profile, Account, Appearance, Accessibility, Notifications, and Billing and licensing. The main content area is titled "SSH keys" and includes a note: "This is a list of SSH keys associated with your account. Remove any keys that you do not recognize." Under the "Authentication keys" section, there is one key listed: **CPE232_BuenoJoshuaa**. The key's details include the SHA256 hash: `dN18cLzvCwKpoBBpWknXtXd46PQH96Qx7Cfv21IP9E`, the date added: "Added on Aug 15, 2025", and the status: "Never used — Read/write".

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. cat

```
joshuabueno@workstation:~$ cat .ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQDAuwftTZeCEA1Pe2C0M02U8bevWkT9DoYsyHxz150J
gTEiH4S4nnjrAYDT0zzVk5qpzSEDr3qRo7kqM2WkrQ3f/9lstp+TbsVMHkUyoaDFWhzz8jk9w3jCl6wd
zny0wfEV6oHAUjeqnmnWr6sEtB1a5FwylU0aBwWQHTPDxBY6Xk34uQ3PgWWUoXtbHDfoLATH/QKazMhH
oQj1u+tgZK8tLEzApM8ZHc51DIL7X8FJmYUMp40KnGe8dVeuVupoT6+va37RCdsfHv4ULs6Q73lkG+t
P1IqDe2mQ470c52LNPOPl5YIOAB5myzOhSugiEhXFK4NzDwjxPCp+bJhgmLKCwZ3//cP9GYKut2RlW0y
EFkgQ3HI0fJb60/Q08ss9piI1E7Ek0oWVUt4jZiAIGf60HoQd2ArxWIu/2/D0fA3LkIx7zTFZfKwnsR7
G8NXqDzT7z4ZYuwj3VHgc29mtWB4tB9tqBQAWukTjGqeWny5fYOMXP3R5A8m2XdCGYHAaB/aREv72ZG5
+cZhA4sQpIqi09dzPIu/ZyACcaMjnBkwWr8+KL3ZxZOWUN2v4VnHZV8Vmp2VmwDazVvi6BaZP8zirLrj
jYI8ozJSrZhV9k8sRi5iGM12Rif6F4Gk+/n6PtWW/JTTJEFniHq3t3G1FbGucTsZTvk7E/wGp7J8o33R
KQ== joshuabueno@workstation
```

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.

```
joshuabueno@workstation:~$ git clone git@github.com:joshuabueno123/CPE232_Bueno-Joshua.git
Cloning into 'CPE232_Bueno-Joshua'...
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.
```

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`.

```
joshuabueno@workstation:~$ ls
CPE232_Bueno-Joshua  Documents  Music      Public  Templates
Desktop              Downloads  Pictures   snap    Videos
```

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`

```
joshuabueno@workstation:~/CPE232_Bueno-Joshua$ git config --global user.name "BuenoJoshua"
> "
joshuabueno@workstation:~/CPE232_Bueno-Joshua$ git config --global user.email qjbueno@tip.edu.ph
joshuabueno@workstation:~/CPE232_Bueno-Joshua$ cat ~/.gitconfig
[user]
    name = BuenoJoshua\n
    email = qjbueno@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.

```
GNU nano 7.2
# CPE232_Bueno-Joshua
Green
```

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?

```
joshuabueno@workstation:~/CPE232_Bueno-Joshua$ git status
On branch main
Your branch is up to date with 'origin/main'.
```

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.

```
joshuabueno@workstation:~/CPE232_Bueno-Joshua$ git add README.md
joshuabueno@workstation:~/CPE232_Bueno-Joshua$ git commit -m "POGIAKO"
[main 8028181] POGIAKO
1 file changed, 2 insertions(+), 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

```
joshuabueno@workstation:~/CPE232_Bueno-Joshua$ git push origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Writing objects: 100% (3/3), 274 bytes | 274.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:joshuabueno123/CPE232_Bueno-Joshua.git
99fe306..8028181  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 the GitHub interface for a repository named 'CPE232_Bueno-Joshua' by user 'joshuabueno123'. The repository is public and has 1 branch and 0 tags. The main branch is selected. A search bar and a 'Code' button are visible. Below the repository header, a commit by 'POGIAKO' is shown, dated '1 minute ago' with '2 Commits'. The file 'README.md' is listed. The README content is displayed, showing the title 'CPE232_Bueno-Joshua' and the word 'GREEN'.

Reflections:

Answer the following:

3. What sort of things have we so far done to the remote servers using ansible commands?
 - We've used Ansible to check if servers are connected, manage users and SSH keys, install or update software, set up configuration files, control services, and get system info like disk space. The inventory file is important because it tells Ansible which servers to manage and how to organize them.

4. How important is the inventory file?

-The inventory file is very important because it tells Ansible which servers to manage. Without it, Ansible wouldn't know where to connect or how to organize the servers. It's like a list or map that guides Ansible to do its tasks on the right machines.

Conclusions/Learnings:

- In this activity, setting up SSH key-based authentication and Git is useful because it makes connecting to remote servers and managing code easier and safer. By creating SSH keys and adding the public key to the remote server, I can connect without typing a password each time, and the connection is secure. Setting up Git with my user info helps keep track of changes and work well with others. These skills are important for modern software development because they make access safe and help with organizing code.