

<b>Name: Lance Gebrielle A. Santos</b>	<b>Date Performed: 01/23/2024</b>
<b>Course/Section: CPE232 - CPE31S1</b>	<b>Date Submitted: 01/23/2024</b>
<b>Instructor: Dr. Jonathan V. Taylor</b>	<b>Semester and SY: 2nd Semester 2023-2024</b>
<b>Activity 2: SSH Key-Based Authentication and Setting up Git</b>	
<b>1. Objectives:</b> 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	
<b>Part 1: Discussion</b>  It is assumed that you are already done with the last Activity ( <b>Activity 1: Configure Network using Virtual Machines</b> ). <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.  <b>What is ssh-keygen?</b>  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.  <b>SSH Keys and Public Key Authentication</b>  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`.

```
santos@workstation:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/santos/.ssh/id_rsa): /home/santos/.ssh/id_rsa
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/santos/.ssh/id_rsa.
Your public key has been saved in /home/santos/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:v/ziap0NmIPs2LqHtMpm/5tJys/A6uwPF6egs4BUGw santos@workstation
The key's randomart image is:
+---[RSA 2048]---+
| ..                |
| .E                |
| ..               |
| .                |
| .      S         |
| . ....          |
|O= += +         |
| =+#+X O....     |
|O#=#*O@O.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.

```
santos@workstation:~$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/santos/.ssh/id_rsa): /home/santos/.ssh/id_rsa
/home/santos/.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/santos/.ssh/id_rsa.
Your public key has been saved in /home/santos/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:vIr+l/pQxXoIVL31Q0Kkh7LWh/F7W0Xht61IA166Szc santos@workstation
The key's randomart image is:
+---[RSA 4096]---+
| .... oo          |
| . . .oo .        |
| . . *O.+ .       |
| . B.* oo .       |
| * + S o.o o .    |
| o . . * + o .o   |
| .   + = E ..     |
| . o . 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`.

```
santos@workstation:~$ ls -la .ssh
total 20
drwx----- 2 santos santos 4096 Jan 23 17:25 .
drwxr-xr-x 15 santos santos 4096 Jan 22 19:58 ..
-rw----- 1 santos santos 3243 Jan 23 17:28 id_rsa
-rw-r--r-- 1 santos santos  744 Jan 23 17:28 id_rsa.pub
-rw-r--r-- 1 santos santos  888 Jan 23 17:08 known_hosts
```

## 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.

```
santos@workstation:~$ ssh-copy-id
Usage: /usr/bin/ssh-copy-id [-h|-?|-f|-n] [-i [identity_file]] [-p port] [[-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
    -h|-?: print this help
```

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

```
santos@workstation:~$ ssh-copy-id -i ~/.ssh/id_rsa santos@workstation
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/santos/.ssh/id_rsa.pub"
The authenticity of host 'workstation (127.0.0.1)' can't be established.
ECDSA key fingerprint is SHA256:XCjdm+LuZT8IHqMhAIZXiXWBQgtsevHtsViKkfMS6tI.
Are you sure you want to continue connecting (yes/no)? 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
santos@workstation's password:

Number of key(s) added: 1

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

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

Number of key(s) added: 1

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

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

Number of key(s) added: 1

Now try logging into the machine, with: "ssh 'santos@server2'"
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?

```
santos@workstation:~$ ssh santos@server1
Welcome to Ubuntu 18.04.6 LTS (GNU/Linux 4.18.0-15-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.
*** System restart required ***
Last login: Tue Jan 23 17:06:42 2024 from 192.168.56.124
santos@server1:~$ logout
Terminal
Connection to server1 closed.
```

```
santos@workstation:~$ ssh santos@server2
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.

145 additional security updates can be applied with ESM Infra.
Learn more about enabling ESM Infra service for Ubuntu 18.04 at
https://ubuntu.com/18-04

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: Tue Jan 23 17:08:07 2024 from 192.168.56.124
santos@server2:~$ logout
Connection to server2 closed.
```

I notice that it doesn't ask for a password because of the `authorized_keys` that was input in `managedNode`.

### Reflections:

Answer the following:

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

It is a method for securely sending commands to a computer over an unsecured network. It is protocol and program that allows secure remote access to a computer or server over a potentially unsecured network. It provides a secure encrypted communication channel, enabling users to log in and execute commands on a remote machine as if they were directly interacting with it.

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

You can try to SSH into a remote server using the corresponding private key. If the public key is installed, and the private key matches, you should be granted access without entering a password.

### 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*

```
santos@workstation:~$ sudo apt install git
[sudo] password for santos:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following package was automatically installed and is no longer required:
  libllvm7
```

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

```
santos@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.

```
santos@workstation:~$ git --version
git version 2.17.1
```

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.

**Create a new repository**

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository](#).

Required fields are marked with an asterisk (\*).

Owner \* Repository name \*

santoslance / CPE232\_SantosLance

✔ CPE232\_SantosLance is available.

Great repository names are short and memorable. Need inspiration? How about [miniature-goggles](#) ?

Description (optional)

☒ **Public**  
Anyone on the internet can see this repository. You choose who can commit.

☐ **Private**  
You choose who can see and commit to this repository.

Initialize this repository with:

☒ **Add a README file**  
This is where you can write a long description for your project. [Learn more about READMEs](#).

**Add .gitignore**

.gitignore template: **None**

Choose which files not to track from a list of templates. [Learn more about ignoring files](#).

**Choose a license**

License: **None**

A license tells others what they can and can't do with your code. [Learn more about licenses](#).

This will set **main** as the default branch. Change the default name in your [settings](#).

① You are creating a public repository in your personal account.

**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.

**Add new SSH Key**

Title

CPE232

Key type

Authentication 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.

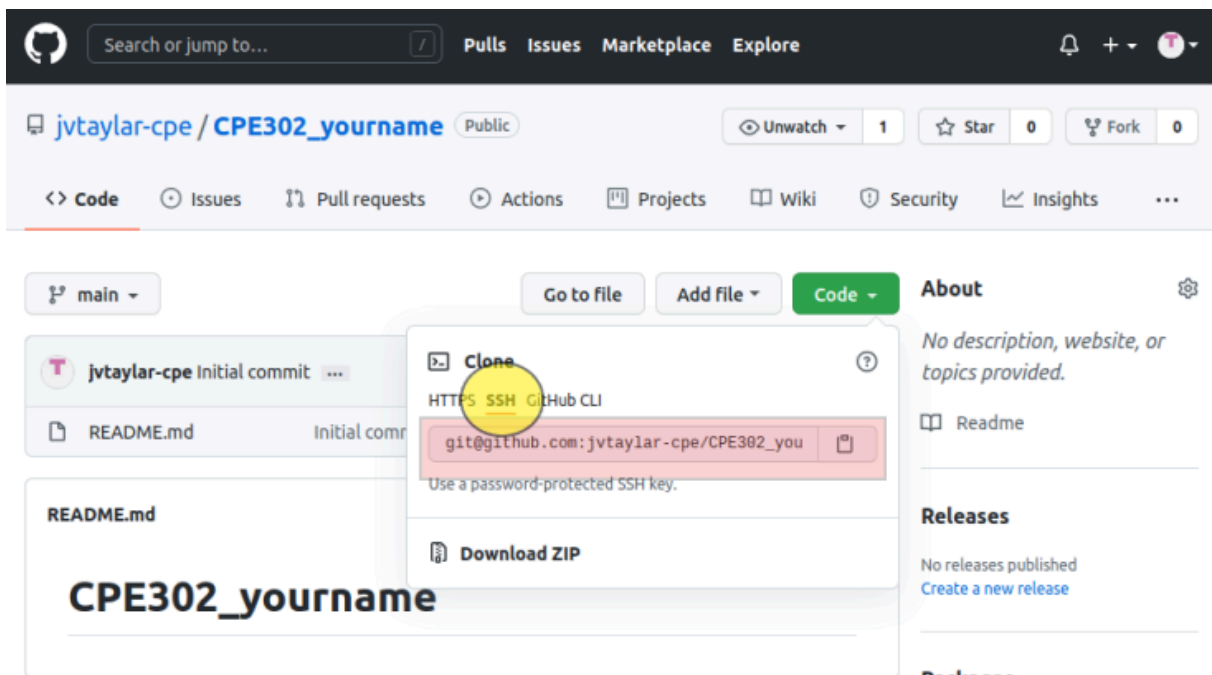
### Authentication keys

**CPE232**  
SHA256: vIr+i/pQxXoIVL31Q0Kkh7LWh/F7WOXht61IA166SZc  
Added on Jan 23, 2024  
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 the GitHub interface for a repository named 'CPE302\_yourname' by user 'jvtaylor-cpe'. The repository is public and has 1 watch, 0 stars, and 0 forks. The 'Code' button is highlighted with a green circle, and the 'SSH' option in the dropdown menu is highlighted with a red box. The repository name 'CPE302\_yourname' is visible in the main content area.

- 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.



```
santos@workstation:~$ git clone git@github.com:santoslance/CPE232_SantosLance.git
Cloning into 'CPE232_SantosLance'...
The authenticity of host 'github.com (20.205.243.166)' can't be established.
ECDSA key fingerprint is SHA256:p2QAMXNIC1TJYWeIOttrVc98/R1BUFWu3/LiyKgufQM.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'github.com,20.205.243.166' (ECDSA) 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.
```

- 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.

```
santos@workstation:~$ ls
CPE232_SantosLance  Documents  examples.desktop  Pictures  Templates
Desktop             Downloads  Music             Public    Videos

santos@workstation:~$ cd CPE232_SantosLance
santos@workstation:~/CPE232_SantosLance$ ls
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`

```
santos@workstation:~$ git config --global user.name "Lance"
santos@workstation:~$ git config --global user.email "qlgasantos@tip.edu.ph"
santos@workstation:~$ cat ~/.gitconfig
[user]
    name = Lance
    email = qlgasantos@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 2.9.3 README.md

# CPE232_SantosLance
God is Good!
```

- 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?

```
santos@workstation:~/CPE232_SantosLance$ 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.
- 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.

```
santos@workstation:~/CPE232_SantosLance$ git add README.md
santos@workstation:~/CPE232_SantosLance$ git commit -m "Enjoy"
[main 39025d6] Enjoy
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`.

```
santos@workstation:~/CPE232_SantosLance$ git push origin main
Counting objects: 3, done.
Writing objects: 100% (3/3), 272 bytes | 272.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To github.com:santoslance/CPE232_SantosLance.git
8e0a71f..39025d6  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.

CPE232\_SantosLance / README.md

santoslance Enjoy 2 minutes ago

2 lines (2 loc) · 34 Bytes

Preview Code Blame Raw Copy Download Edit

## CPE232\_SantosLance

God is Good!

**Reflections:**

Answer the following:

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

The use of the private and public key allows the host to SSH into the server without requiring a password with the help of authorized keys.

4. How important is the inventory file?

The inventory file is essential for effective management, especially in larger infrastructure where you need to orchestrate tasks across multiple servers or devices.

**Conclusions/Learnings:**

This Hands-on Activity 2.1 teaches me how to set up local and remote machines so that they can connect over SSH with KEYS rather than passwords. In addition, I learnt how to generate the public and private keys needed for connectivity verification and authentication. In addition to configuring Github and linking my Ubuntu server. In general, any network can be connected to another, and we can do a wide range of tasks using Ubuntu, including managing connections and networks.