

| | |
|--|--|
| Name: Bolivar, Deo, M. | Date Performed: September 1, 2022 |
| Course/Section: CPE 232-CPE31S24 | Date Submitted: September 2, 2022 |
| Instructor: Engr. Jonathan V. Taylor. | Semester and SY: 1 st semester and SY: 2022-2023 |

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*.

```
Bolivar@DESKTOP-31MIQ57 MINGW64 ~
$ cd .ssh

Bolivar@DESKTOP-31MIQ57 MINGW64 ~/.ssh
$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/Bolivar/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /c/Users/Bolivar/.ssh/id_rsa
Your public key has been saved in /c/Users/Bolivar/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:ZBqoIcay9VTnIquqnF5M65+P7TjyRrhJdkSXk4YHhI Bolivar@DESKTOP-31MIQ57
The key's randomart image is:
+---[RSA 3072]-----+
|
|  E.o
|   o +
|  + o . .
|   . B . .+
|  * So+o
|   o,==,+o
|   . o.B0o. .
|   o +*B=.
|   .o==+=o
+---[SHA256]-----+

Bolivar@DESKTOP-31MIQ57 MINGW64 ~/.ssh
$ ls
id_rsa id_rsa.pub known_hosts known_hosts.old
```

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.

```
Bolivar@DESKTOP-31MIQ57 MINGW64 ~/.ssh
$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/Bolivar/.ssh/id_rsa):
```

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.

```
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /c/Users/Bolivar/.ssh/id_rsa
Your public key has been saved in /c/Users/Bolivar/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:CwcnBf133t9kK/YD8EdcNcfooarPZ31gKPWzS0hd1cE Bolivar@DESKTOP-31MIQ57
The key's randomart image is:
+---[RSA 4096]-----+
|
|  .o.      .B|
|  ..      oEB|
|  o..      o..o|
|  +.  ..+.+.+|
|  . S.  =++o |
|  o .o.o+*.o |
|  ..  ..o+B0|
|  ..  o+00+|
|  .00.  ++. |
+---[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.

```
Bolivar@DESKTOP-31MIQ57 MINGW64 ~/.ssh
$ ls
id_rsa id_rsa.pub known_hosts 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.
2. Issue the command similar to this: `ssh-copy-id -i ~/.ssh/id_rsa user@host`

```
Bolivar@DESKTOP-31MIQ57 MINGW64 ~/.ssh
$ ssh-copy-id -i ~/.ssh/id_rsa bolivar@192.168.56.108

/usr/bin/ssh-copy-id: ERROR: failed to open ID file '/c/Users/Bolivar/.ssh/id_rsa.pub': No such file or directory

Bolivar@DESKTOP-31MIQ57 MINGW64 ~/.ssh
$ ls
id_rsa id_rsa.pub known_hosts known_hosts.old
```

```
Bolivar@DESKTOP-31MIQ57 MINGW64 ~/.ssh
$ ssh-copy-id bolivar@192.168.56.108
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/c/Users/Bolivar/.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
bolivar@192.168.56.108's password:

Number of key(s) added: 1

Now try logging into the machine, with: "ssh 'bolivar@192.168.56.108'"
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?
Server 1:

```

Bolivar@DESKTOP-31MIQ57 MINGW64 ~/.ssh
$ ssh bolivar@192.168.56.108
Welcome to Ubuntu 22.04.1 LTS (GNU/Linux 5.15.0-46-generic x86_64)

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

0 updates can be applied immediately.

Last login: Sat Aug 27 00:42:22 2022 from fe80::a167:96f0:8b57:1e76%enp0s8
bolivar@Server1:~$ exit
logout
Connection to 192.168.56.108 closed.

Bolivar@DESKTOP-31MIQ57 MINGW64 ~/.ssh
$ ssh bolivar@Server1
Welcome to Ubuntu 22.04.1 LTS (GNU/Linux 5.15.0-46-generic x
86_64)

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

0 updates can be applied immediately.

Last login: Thu Sep  1 23:47:58 2022 from 192.168.56.1
bolivar@Server1:~$ |

```

Server 2:

```

Bolivar@DESKTOP-31MIQ57 MINGW64 ~/.ssh
$ ssh bolivar@192.168.56.109
bolivar@192.168.56.109's password:
Welcome to Ubuntu 22.04.1 LTS (GNU/Linux 5.15.0-46-generic x86_64)

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

0 updates can be applied immediately.

Last login: Sat Aug 27 00:30:52 2022 from fe80::a167:96f0:8b57:1e76%enp0s8
bolivar@Server2:~$ exit
logout
Connection to 192.168.56.109 closed.

Bolivar@DESKTOP-31MIQ57 MINGW64 ~/.ssh
$ ssh bolivar@Server2
bolivar@server2's password:
Welcome to Ubuntu 22.04.1 LTS (GNU/Linux 5.15.0-46-generic x86_64)

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

0 updates can be applied immediately.

Last login: Fri Sep  2 00:03:41 2022 from 192.168.56.1
bolivar@Server2:~$ |

```

Reflections:

Answer the following:

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

- For me the SSH program can act as a remote and enables the two computers or servers to communicate to each other.
- The SSH program encrypt all message or data before executing on the server.

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

- I ran the command **ls** to see whether there was a file of permitted keys, and then I followed the file name with the **cat** command to see if I had installed the public key to the distant servers.

Server 1:

```
bolivar@Server1:~$ cd .ssh
bolivar@Server1:~/.ssh$ ls
authorized_keys
bolivar@Server1:~/.ssh$ cat authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQDCgowbsmc5pIHfvH/9Cy2LZMf1WQt0
mchmjzz7Bx9h8ExfD1m3wPMpN5lQbbk9Rr4ji8piZBgwszUVcbkFzMld4dtPbkOxe4Y
LcRBYbEXR8zt/PV0dWuXBjU0H6I6Zd3JqCBKqd27TjS9vpJmi+iqkfz7rMefN6/aXChJ
7SPeFdv6NmR1fQJKcMWXgQ4C9JkRh0DaLGINLrZw+x/KfY9fTmc6TLVyx8sTJcyMjjl
F+KZW+KF07kWI4CYQx60+g1l6clayz2pqD9+9yefWgzsasSpWF7Jwom+Y8smIts3tgzv
nmMTUXw+Se+JnygwI7VSOMgn6wUh9cX8uLZv1ZHW9qSJHEuQZTiVZWwRk/QMURR1sJvV
fknv18n2s6DmoTLKMI fqaAhE+Dg3rpnUBHwp7CgSlBnC5lEgP3INaKgK08PwlkjNEFIj
DpDRbXXWl0lsxJVHgLksYlMEkwsYP9zXt9LZB6LDgNu6jjVEMoyrbNDyNG9+9u+ItasN
I2XWcAUIKnsny3d6PZmkGWubvECqaHDy4Vr7x/gx0terjdLt0KopxqjPPiv9sboXf/CY
ACbZY0mdgaw== Bolivar@DESKTOP-31MIQ57
bolivar@Server1:~/.ssh$
```

Server 2:

```
bolivar@Server2:~$ cd .ssh
bolivar@Server2:~/.ssh$ ls
authorized_keys
bolivar@Server2:~/.ssh$ cat authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQDCgowbsmc5pIHfvH/9Cy2LZMf1WQt0
mchmjzz7Bx9h8ExfD1m3wPMpN5lQbbk9Rr4ji8piZBgwszUVcbkFzMld4dtPbkOxe4
LcRBYbEXR8zt/PV0dWuXBjU0H6I6Zd3JqCBKqd27TjS9vpJmi+iqkfz7rMefN6/aXCh
7SPeFdv6NmR1fQJKcMWXgQ4C9JkRh0DaLGINLrZw+x/KfY9fTmc6TLVyx8sTJcyMjjl
F+KZW+KF07kWI4CYQx60+g1l6clayz2pqD9+9yefWgzsasSpWF7Jwom+Y8smIts3tgz
nmMTUXw+Se+JnygwI7VSOMgn6wUh9cX8uLZv1ZHW9qSJHEuQZTiVZWwRk/QMURR1sJv
fknv18n2s6DmoTLKMI fqaAhE+Dg3rpnUBHwp7CgSlBnC5lEgP3INaKgK08PwlkjNEFI
DpDRbXXWl0lsxJVHgLksYlMEkwsYP9zXt9LZB6LDgNu6jjVEMoyrbNDyNG9+9u+Itas
I2XWcAUIKnsny3d6PZmkGWubvECqaHDy4Vr7x/gx0terjdLt0KopxqjPPiv9sboXf/C
ACbZY0mdgaw== Bolivar@DESKTOP-31MIQ57
bolivar@Server2:~/.ssh$
```

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*

```
Bolivar@DESKTOP-31MIQ57 MINGW64 ~
$ which git
/mingw64/bin/git

Bolivar@DESKTOP-31MIQ57 MINGW64 ~
$ |
```

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

```
Bolivar@DESKTOP-31MIQ57 MINGW64 ~
$ git --version
git version 2.37.2.windows.2

Bolivar@DESKTOP-31MIQ57 MINGW64 ~
$ |
```

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.

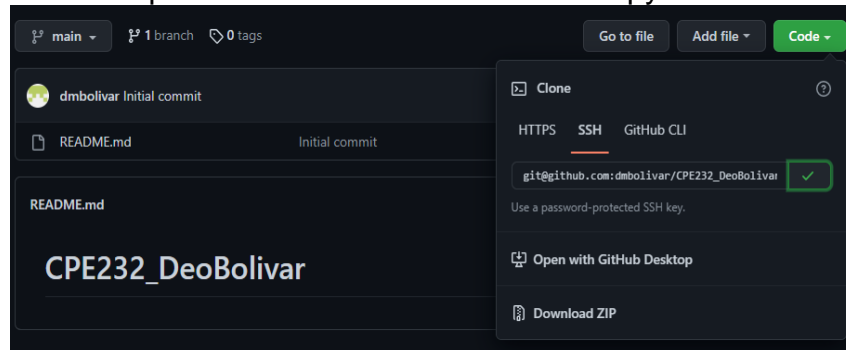
Note: Since there was an error on the output above it only means that I already used the Repository name and I forgot to SS that process that is why on the output above in the Repository name said is I already use the Repository name.

- 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.



- 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.

```
Bolivar@DESKTOP-31MIQ57 MINGW64 ~
$ git clone https://github.com/dmbolivar/CPE232_DeoBolivar.git
Cloning into 'CPE232_DeoBolivar'...
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.

Bolivar@DESKTOP-31MIQ57 MINGW64 ~
$ |
```

- 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`.


```

Bolivar@DESKTOP-31MIQ57 MINGW64 ~
$ ls
-1.14-windows.xml
'3D Objects'/'
AppData/
'Application Data'@
CPE232_DeoBolivar/

Bolivar@DESKTOP-31MIQ57 MINGW64 ~
$ cd CPE232_DeoBolivar

Bolivar@DESKTOP-31MIQ57 MINGW64 ~/CPE232_DeoBolivar (main)
$ ls
README.md

Bolivar@DESKTOP-31MIQ57 MINGW64 ~/CPE232_DeoBolivar (main)
$ |

```

- 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`

```

Bolivar@DESKTOP-31MIQ57 MINGW64 ~/CPE232_DeoBolivar (main)
$ git config --global user.name "Deo Bolivar"

Bolivar@DESKTOP-31MIQ57 MINGW64 ~/CPE232_DeoBolivar (main)
$ git config --global user.email qdmbolivar@tip.edu.ph

Bolivar@DESKTOP-31MIQ57 MINGW64 ~/CPE232_DeoBolivar (main)
$ cat ~/.gitconfig
[user]
    name = Deo Bolivar
    email = qdmbolivar@tip.edu.ph

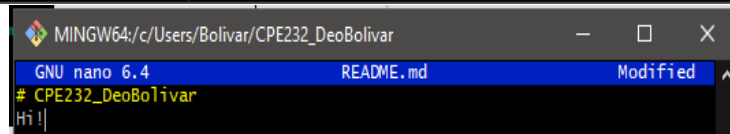
```

- 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.

```

Bolivar@DESKTOP-31MIQ57 MINGW64 ~/CPE232_DeoBolivar (main)
$ nano README.md

```



```

MINGW64:/c/Users/Bolivar/CPE232_DeoBolivar
GNU nano 6.4      README.md      Modified
# CPE232_DeoBolivar
Hi!

```

- 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?

```

Bolivar@DESKTOP-31MIQ57 MINGW64 ~/CPE232_DeoBolivar (main)
$ 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")

```


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

```
Bolivar@DESKTOP-31MIQ57 MINGW64 ~/CPE232_DeoBolivar (main)
$ git add README.md
warning: in the working copy of 'README.md', LF will be replaced by CRLF
the next time Git touches it
```

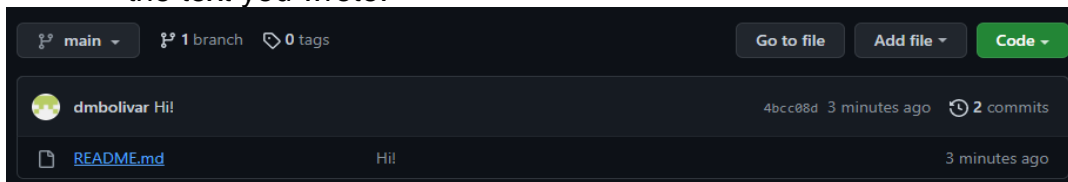
- k. Use the *git commit -m "your message"* to create a snapshot of the staged changes along the timeline of the Git projects history. The use of this command is required to select the changes that will be staged for the next commit.

```
Bolivar@DESKTOP-31MIQ57 MINGW64 ~/CPE232_DeoBolivar (main)
$ git commit -m "Hi!"
[main 4bcc08d] Hi!
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*.

```
Bolivar@DESKTOP-31MIQ57 MINGW64 ~/CPE232_DeoBolivar (main)
$ git push origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Writing objects: 100% (3/3), 264 bytes | 264.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/dmbolivar/CPE232_DeoBolivar.git
0a644ad..4bcc08d 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.



Reflections:

Answer the following:

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

-Using ansible commands, I successfully connected the local machine and servers and also I installed the authorized key on both servers.

4. How important is the inventory file?

- The inventory file is important because it helps us to easily define the hosts to which command in the playbook can operate.

Conclusions/Learnings:

After I finished this activity, I learned that this activity help me in knowing or learning many linux and git commands. I learned how to configure the local machine and remote servers using the SSH. In addition, I was able to configure various instructions to be sent from a local server to remote servers. This knowledge will be useful to me for my upcoming activities.

"I affirm that I will not give or receive any unauthorized help on this activity/exam and that all work will be my own."