

Name: KEVIN SUMAYA	Date Performed:
Course/Section: CPE232 31S6	Date Submitted:
Instructor: Engr. Taylar	Semester and SY:
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`.

```
kevin@Workstation:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/kevin/.ssh/id_rsa): id_
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in id_rsa.
Your public key has been saved in id_rsa.pub.
The key fingerprint is:
SHA256:wV3YhFRR9bejUAPlqgl+I/uarq88e48U97x9ncSzn6s kevin@Workstati
The key's randomart image is:
+---[RSA 2048]---+
|      ..B*+...|
|      . .O.+  .|
|      o .  +  o|
|      .  o  . o|
|      oS. o  .o|
|      . + = . .+|
|      + = o  . .|
|      ....* . o  +o|
|      =0B+o  . Eoo+|
+-----[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.

```

kevin@Workstation:~$ ssh-keygen -t rsa -b 2048
Generating public/private rsa key pair.
^[[3~Enter file in which to save the key (/home/kevin/.ssh/id_rsa)
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/kevin/.ssh/id_rsa.
Your public key has been saved in /home/kevin/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:bSrBwidNR0siqGUGNQ6vtPoXa5fKK7H/8jnXGLZbXmg kevin@Workstation
The key's randomart image is:
+---[RSA 2048]---+
|. o .o          |
| = . . .        |
|..o .o          |
|.+. . = .       |
|o.= = * S o     |
|o+...= .oo .    |
|o o o.o.=E .    |
|.o.= +o++..     |
| .==B+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`.

```

kevin@Workstation:~$ ls -la .ssh
total 12
drwx----- 2 kevin kevin 4096 Aug 17 18:07 .
drwxr-xr-x 16 kevin kevin 4096 Aug 24 17:36 ..
-rw-r--r-- 1 kevin kevin 888 Aug 24 16:41 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.

```

kevin@Workstation:~$ ls -la .ssh
total 12
drwx----- 2 kevin kevin 4096 Aug 17 18:07 .
drwxr-xr-x 16 kevin kevin 4096 Aug 24 17:36 ..
-rw-r--r-- 1 kevin kevin 888 Aug 24 16:41 known_hosts

```

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.

```
kevin@Workstation:~/.ssh$ ssh-copy-id -i ~/.ssh/id_rsa kevin@192.168.56.102
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/kevin/.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
kevin@192.168.56.102's password:
```

Number of key(s) added: 1

Now try logging into the machine, with: "ssh 'kevin@192.168.56.102'" and check to make sure that only the key(s) you wanted were added.

```
kevin@Workstation:~/.ssh$ ssh-copy-id -i ~/.ssh/id_rsa kevin@192.168.56.103
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/kevin/.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
kevin@192.168.56.103's password:
```

Number of key(s) added: 1

Now try logging into the machine, with: "ssh 'kevin@192.168.56.103'" 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?

```
File Edit View Search Terminal Help
kevin@server1:~$ cd .ssh
kevin@server1:~/.ssh$ ls
authorized_keys  known hosts
kevin@server1:~/.ssh$
```

```
kevin@server2:~$ cd .ssh
kevin@server2:~/.ssh$ ls
authorized_keys  known hosts
kevin@server2:~/.ssh$
```

Reflections:

Answer the following:

1. How will you describe the ssh-program? What does it do?
2. How do you know that you already installed the public key to the remote servers?

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*

```
kevin@Workstation:~$ sudo apt install git
[sudo] password for kevin:
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:
```

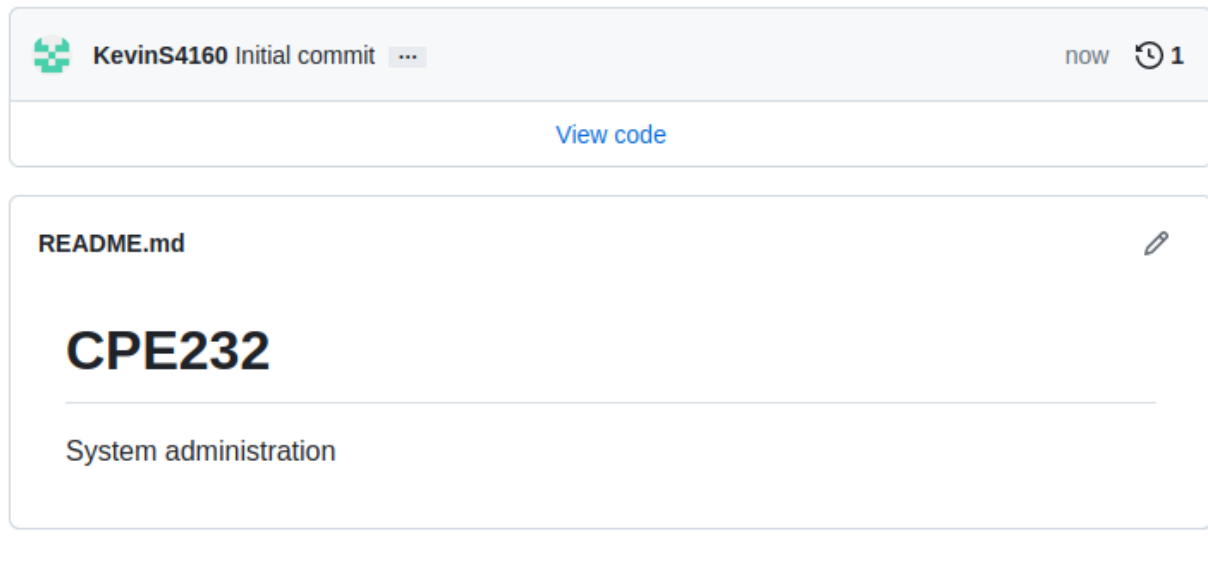
2. After the installation, issue the command *which git* again. The directory of git is usually installed in this location: *user/bin/git*.

```
kevin@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.

```
kevin@Workstation:~$ 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_yourname. Check Add a README file and click Create repository.



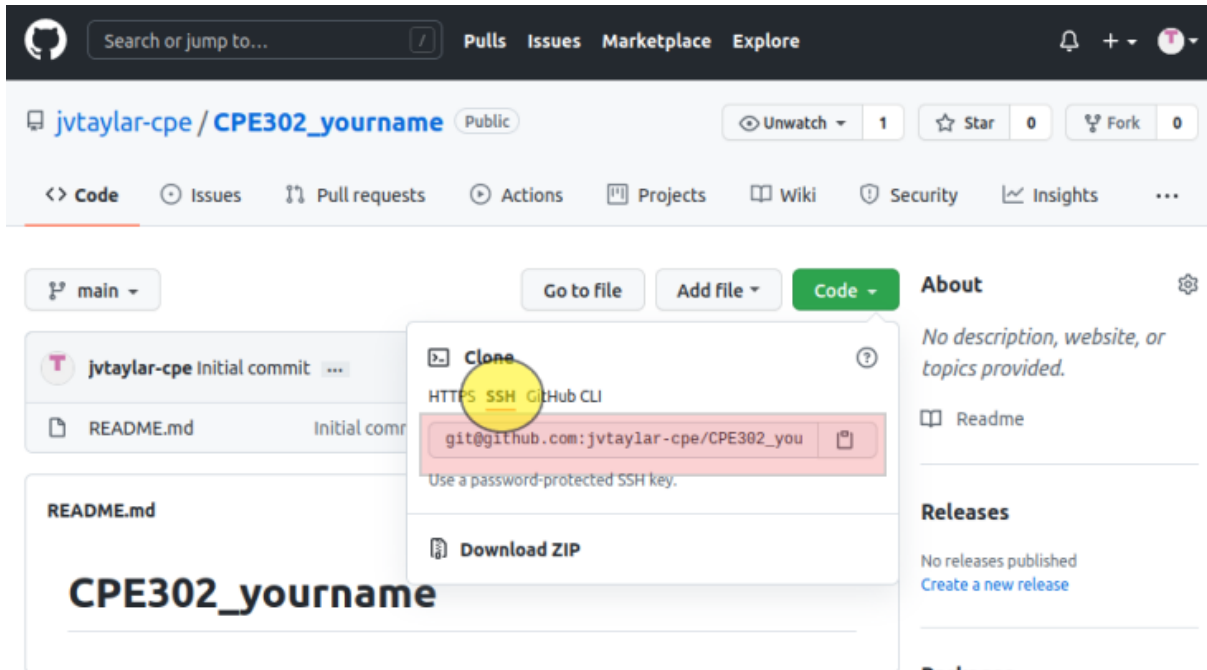
- b. Create a new SSH key on GitHub. Go your profile's setting and click SSH and GPG keys. If there is an existing key, make sure to delete it. To create a new SSH keys, click New SSH Key. Write CPE232 key as the title of the key.

```
kevin@Workstation:~$ cat .ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQDQyXw1e+KU09RRG4ixUyv2s7wFJnYewg02wyvsC
B6Zu5j8zzhuwIt7XiJb9znWr63GDwK78nId7ypU4JTlTpGIIUQw73XgH6lxum4dAe7Z1vr7UX7sRV
Fukvo1qjhuHEGUi53Hp/DvXW9HawviBQPCKZCMqA7SjLEDtsklLUs6VBPKUGNzHi8REiZI/g1uS2c
5KkC7kEf8mp1qGUwK00+bCGywkgLviIkg/Nk806C4TiHH07A0LSZdHec9pQ1j9rDU1xpaQfxabnew
gJcaXAR1PjLwwVgZDeL08bvDbYFZl+qwYwCZu3eyskVCvsp2VS+bewNdhr9NEjjoz kevin@Workst
ion
```

- 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.

```
kevin@Workstation:~$ cat .ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQDQyXw1e+KU09RRG4ixUyv2s7wFJnYewg02wyvsC
B6Zu5j8zzhuwIt7XiJb9znWr63GDwK78nId7ypU4JTlTpGIIUQw73XgH6lxum4dAe7Z1vr7UX7sRV
Fukvo1qjhuHEGUi53Hp/DvXW9HawviBQPCKZCMqA7SjLEDtsklLUs6VBPKUGNzHi8REiZI/g1uS2c
5KkC7kEf8mp1qGUwK00+bCGywkgLviIkg/Nk806C4TiHH07A0LSZdHec9pQ1j9rDU1xpaQfxabnew
gJcaXAR1PjLwwVgZDeL08bvDbYFZl+qwYwCZu3eyskVCvsp2VS+bewNdhr9NEjjoz kevin@Workst
ion
```

- 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.
- 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.
- 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`
- 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.
- 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?

- 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`.
- 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?
- 4. How important is the inventory file?

Conclusions/Learnings: