Name: Christian Jaraula	**Date Performed:**
Course/Section:CPE	**Date Submitted:**
Instructor: Prof. 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).** *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.

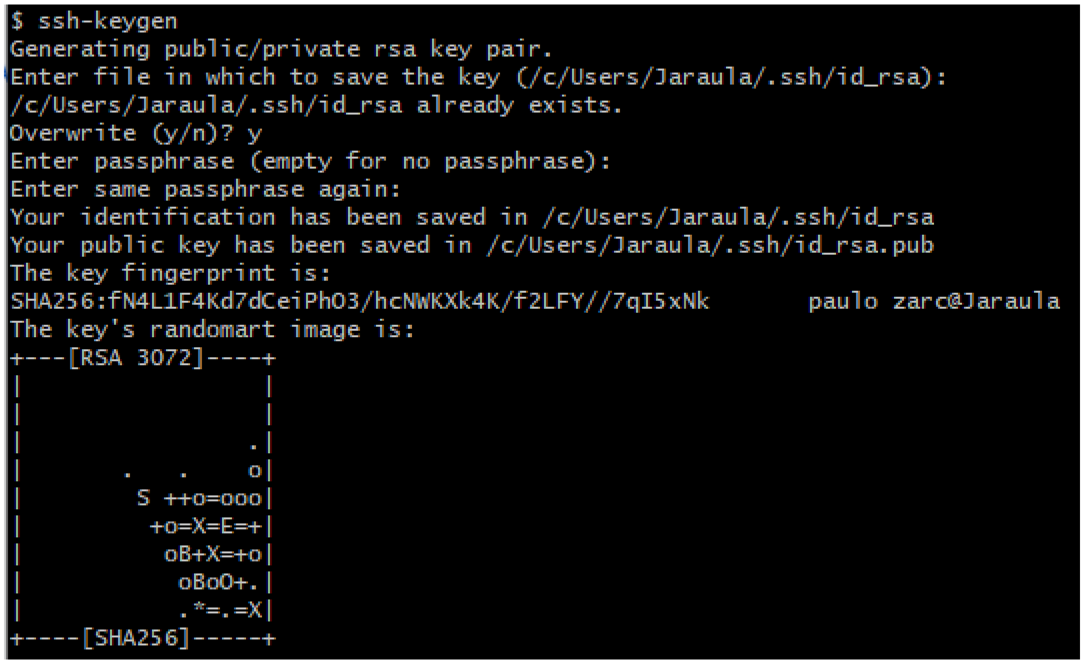
SSH introduced public key authentication as a more secure alternative to the older .rhosts authentication. It improved security by avoiding the need to have password stored in files and eliminated the possibility of a compromised server stealing the user's password.

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.

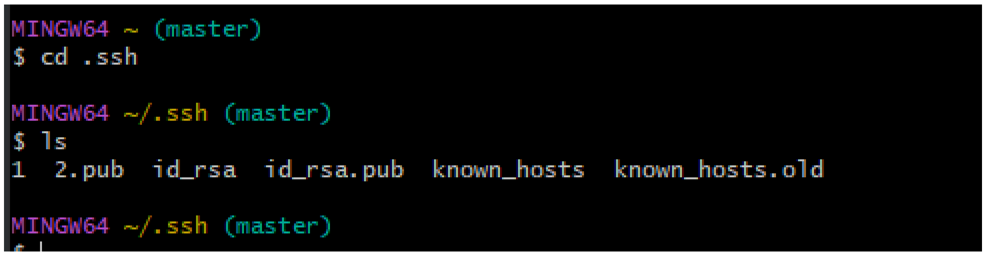
Task 1: Create an SSH Key Pair for User Authentication

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*.

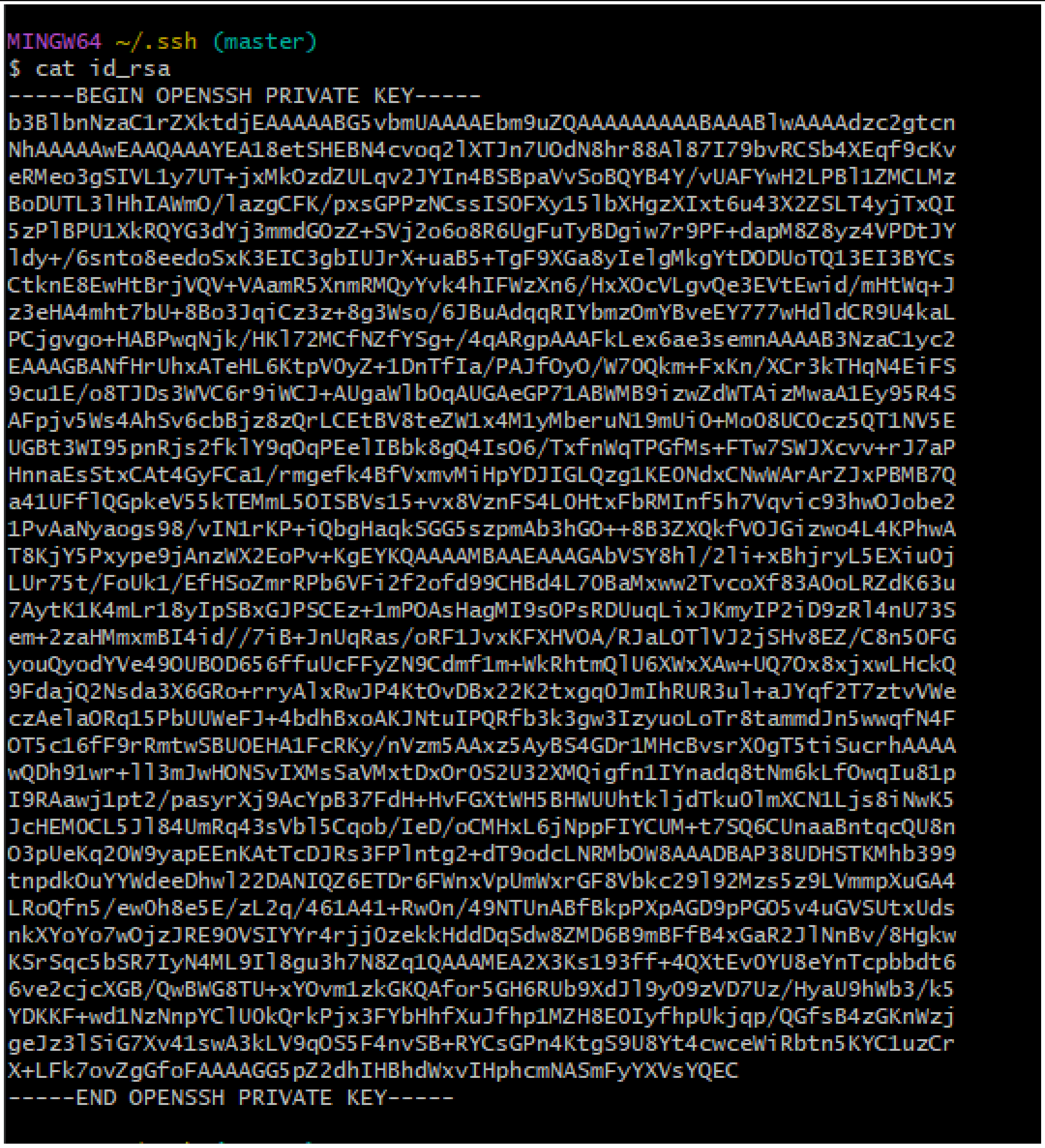


```
$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/Jaraula/.ssh/id_rsa):
/c/Users/Jaraula/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /c/Users/Jaraula/.ssh/id_rsa
Your public key has been saved in /c/Users/Jaraula/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:fN4L1F4Kd7dCeiPhO3/hcNWKXk4K/f2LFY//7qI5xNk       paulo zarc@Jaraula
The key's randomart image is:
+---[RSA 3072]----+
|                 |
|                 |
|                .|
|       .   .    o|
|        S ++o=ooo|
|         +o=X=E=+|
|          oB+X=+o|
|           oBoO+.|
|           .*=.=X|
+----[SHA256]-----+
```

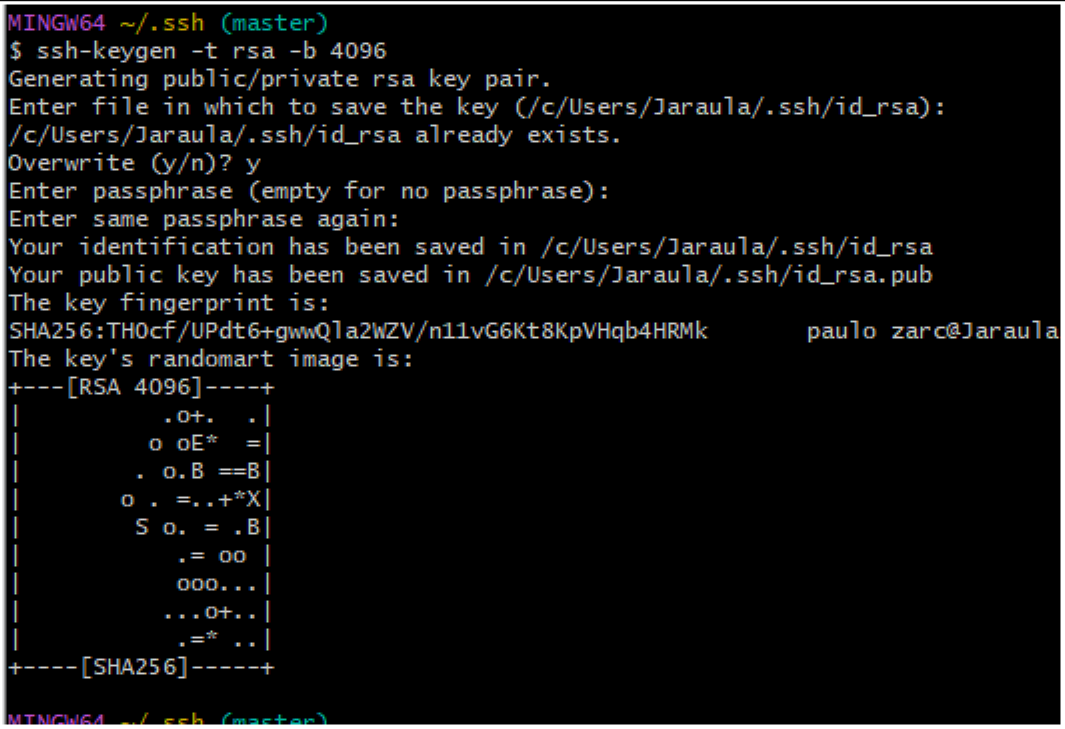


```
MINGW64 ~ (master)
$ cd .ssh

MINGW64 ~/.ssh (master)
$ ls
1  2.pub  id_rsa  id_rsa.pub  known_hosts  known_hosts.old

MINGW64 ~/.ssh (master)
```

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.

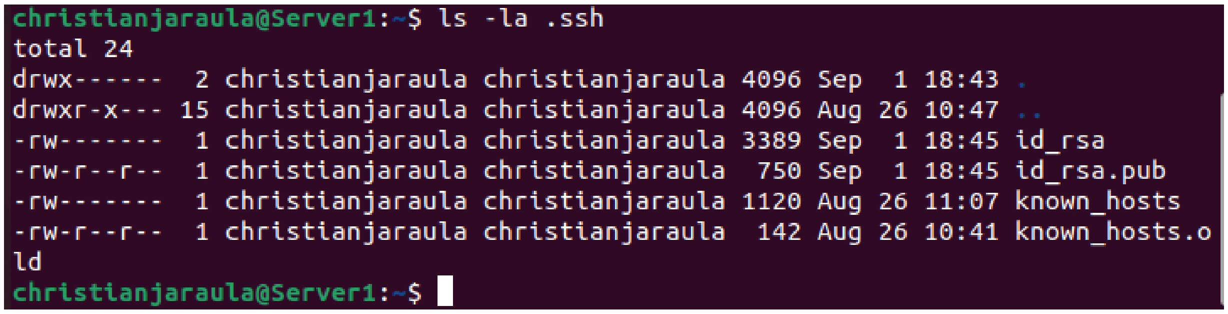


```
MINGW64 ~/.ssh (master)
$ cat id_rsa
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAABlwAAAAdzc2gtcn
NhAAAAAwEAAQAAAYEA18etSHEBN4cvoq2lXTJn7UOdN8hr88Al87I79bvRCSb4XEqf9cKv
eRMeo3gSIVL1y7UT+jxMkOzdZULqv2JYIn4BSBpaVvSoBQYB4Y/vUAFYwH2LPBl1ZMCLMz
BoDUTL3lHhIAWmO/lazgCFK/pxsGPPzNCssISOFXy15lbXHgzXIxt6u43X2ZSLT4yjTxQI
5zPlBPU1XkRQYG3dYj3mmdGOzZ+SVj2o6o8R6UgFuTyBDgiw7r9PF+dapM8Z8yz4VPDtJY
ldy+/6snto8eedoSxK3EIC3gbIUJrX+uaB5+TgF9XGa8yIelgMkgYtDODUoTQ13EI3BYCs
CtknE8EwHtBrjVQV+VAamR5XnmRMQyYvk4hIFWzXn6/HxXOcVLgvQe3EVtEwid/mHtWq+J
z3eHA4mht7bU+8Bo3JqiCz3z+8g3Wso/6JBuAdqqRIYbmzOmYBveEY777wHdldCR9U4kaL
PCjgvgo+HABPwqNjk/HKl72MCfNZfYSg+/4qARgpAAAFkLex6ae3semnAAAAB3NzaC1yc2
EAAAGBANfHrUhxATeHL6KtpVOyZ+1DnTfIa/PAJfOyO/W7OQkm+FxKn/XCr3kTHqN4EiFS
9cu1E/o8TJDs3WVC6r9iWCJ+AUgaWlbOqAUGAeGP71ABWMB9izwZdWTAizMwaA1Ey95R4S
AFpjv5Ws4AhSv6cbBjz8zQrLCEtBV8teZW1x4M1yMberuN19mUiO+MoO8UCOcz5QT1NV5E
UGBt3WI95pnRjs2fklY9qOqPEelIBbk8gQ4IsO6/TxfnWqTPGfMs+FTw7SWJXcvv+rJ7aP
HnnaEsStxCAt4GyFCa1/rmgefk4BfVxmvMiHpYDJIGLQzg1KEONdxCNwWArArZJxPBMB7Q
a41UFflQGpkeV55kTEMmL5OISBVs15+vx8VznFS4LOHtxFbRMInf5h7Vqvic93hwOJobe2
1PvAaNyaogs98/vIN1rKP+iQbgHaqkSGG5szpmAb3hGO++8B3ZXQkfVOJGizwo4L4KPhwA
T8KjY5Pxype9jAnzWX2EoPv+KgEYKQAAAAMBAAEAAAGAbVSY8hl/2li+xBhjryL5EXiuOj
LUr75t/FoUk1/EfHSoZmrRPb6VFi2f2ofd99CHBd4L7OBaMxww2TvcoXf83AOoLRZdK63u
7AytK1K4mLr18yIpSBxGJPSCEz+1mPOAsHagMI9sOPsRDUuqLixJKmyIP2iD9zRl4nU73S
em+2zaHMmxmBI4id//7iB+JnUqRas/oRF1JvxKFXHVOA/RJaLOTlVJ2jSHv8EZ/C8n5OFG
youQyodYVe49OUBOD656ffuUcFFyZN9Cdmf1m+WkRhtmQlU6XWxXAw+UQ7Ox8xjxwLHckQ
9FdajQ2Nsda3X6GRo+rryAlxRwJP4KtOvDBx22K2txgqOJmIhRUR3ul+aJYqf2T7ztvVWe
czAelaORq15PbUUWeFJ+4bdhBxoAKJNtuIPQRfb3k3gw3IzyuoLoTr8tammdJn5wwqfN4F
OT5c16fF9rRmtwSBUOEHA1FcRKy/nVzm5AAxz5AyBS4GDr1MHcBvsrXOgT5tiSucrhAAAA
wQDh91wr+ll3mJwHONSvIXMsSaVMxtDxOrOS2U32XMQigfn1IYnadq8tNm6kLfOwqIu81p
I9RAawj1pt2/pasyrXj9AcYpB37FdH+HvFGXtWH5BHWUUhtkljdTkuOlmXCN1Ljs8iNwK5
JcHEMOCL5Jl84UmRq43sVbl5Cqob/IeD/oCMHxL6jNppFIYCUM+t7SQ6CUnaaBntqcQU8n
O3pUeKq2OW9yapEEnKAtTcDJRs3FPlntg2+dT9odcLNRMbOW8AAADBAP38UDHSTKMhb399
tnpdkOuYYWdeeDhwl22DANIQZ6ETDr6FWnxVpUmWxrGF8Vbkc29l92Mzs5z9LVmmpXuGA4
LRoQfn5/ewOh8e5E/zL2q/461A41+RwOn/49NTUnABfBkpPXpAGD9pPGO5v4uGVSUtxUds
nkXYoYo7wOjzJRE9OVSIYYr4rjjOzekkHddDqSdw8ZMD6B9mBFfB4xGaR2JlNnBv/8Hgkw
KSrSqc5bSR7IyN4ML9Il8gu3h7N8Zq1QAAAMEA2X3Ks193ff+4QXtEvOYU8eYnTcpbbdt6
6ve2cjcXGB/QwBWG8TU+xYOvm1zkGKQAfor5GH6RUb9XdJl9yO9zVD7Uz/HyaU9hWb3/k5
YDKKF+wd1NzNnpYClUOkQrkPjx3FYbHhfXuJfhp1MZH8EOIyfhpUkjqp/QGfsB4zGKnWzj
geJz3lSiG7Xv41swA3kLV9qOS5F4nvSB+RYCsGPn4KtgS9U8Yt4cwceWiRbtn5KYC1uzCr
X+LFk7ovZgGfoFAAAAGG5pZ2dhIHBhdWxvIHphcmNNASmFyYXVsYQEC
-----END OPENSSH PRIVATE KEY-----
```



```
MINGW64 ~/.ssh (master)
$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/Jaraula/.ssh/id_rsa):
/c/Users/Jaraula/.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 /c/Users/Jaraula/.ssh/id_rsa
Your public key has been saved in /c/Users/Jaraula/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:THOcf/UPdt6+gwwQla2WZV/n11vG6Kt8KpVHqb4HRMk       paulo zarc@Jaraula
The key's randomart image is:
+---[RSA 4096]----+
|          .o+.  .|
|         o oE*  =|
|        . o.B ==B|
|       o . =..+*X|
|        S o. = .B|
|          .= oo  |
|          ooo... |
|        ...o+..  |
|          .=* .. |
+----[SHA256]-----+
```

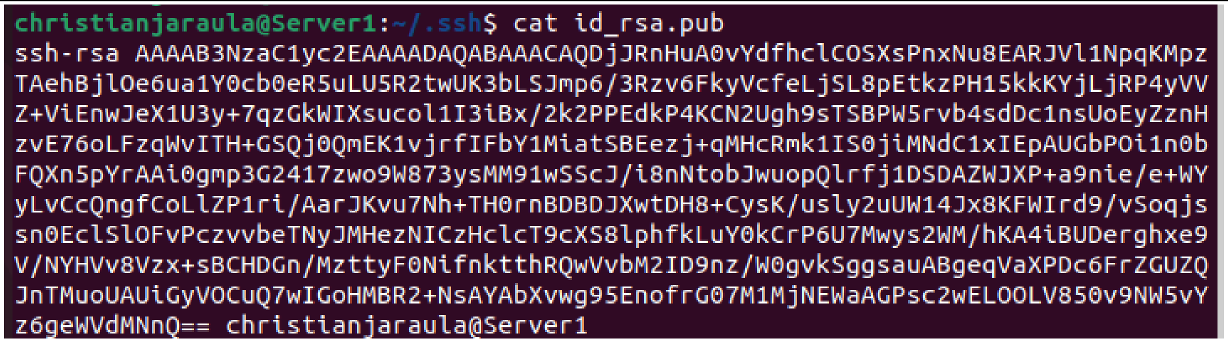
3. When asked for a passphrase, just press enter. The passphrase is used for encrypting the key, so that it cannot be used even if someone obtains the private key file. The passphrase should be cryptographically strong.
4. Verify that you have created the key by issuing the command *ls -la .ssh.* The command should show the .ssh directory containing a pair of keys. For example, id_rsa.pub and id_rsa.



```
christianjaraula@Server1:~$ ls -la .ssh
total 24
drwx------  2 christianjaraula christianjaraula 4096 Sep  1 18:43 .
drwxr-x--- 15 christianjaraula christianjaraula 4096 Aug 26 10:47 ..
-rw-------  1 christianjaraula christianjaraula 3389 Sep  1 18:45 id_rsa
-rw-r--r--  1 christianjaraula christianjaraula  750 Sep  1 18:45 id_rsa.pub
-rw-------  1 christianjaraula christianjaraula 1120 Aug 26 11:07 known_hosts
-rw-r--r--  1 christianjaraula christianjaraula  142 Aug 26 10:41 known_hosts.o
ld
christianjaraula@Server1:~$ 
```

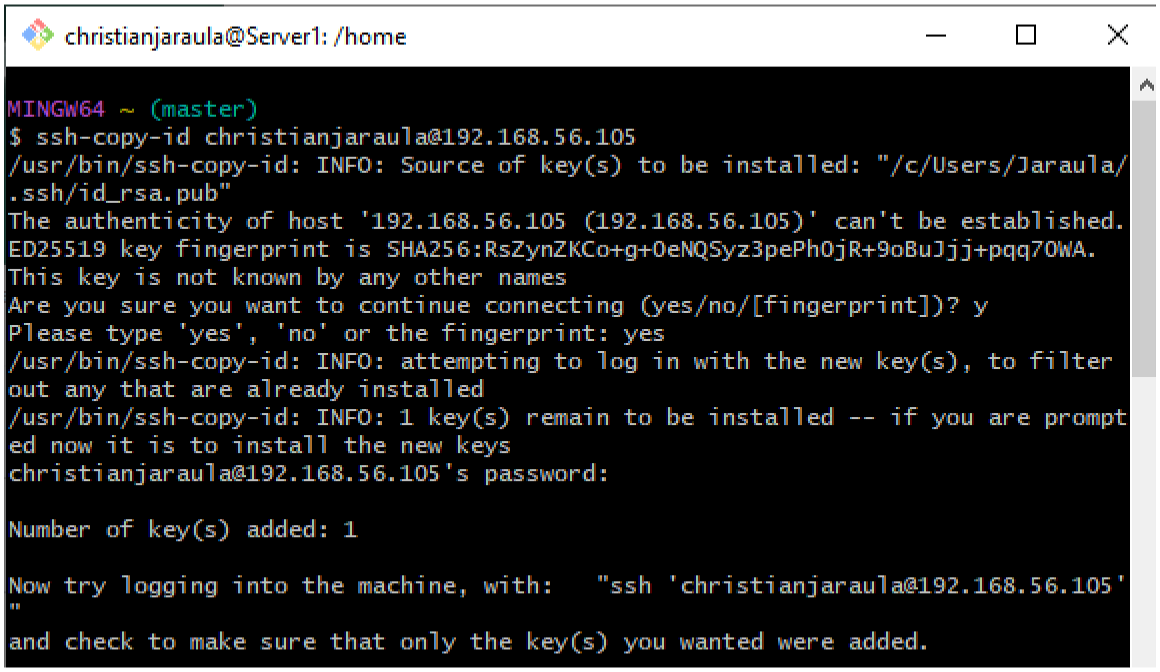
Task 2: Copying the Public Key to the remote servers

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.



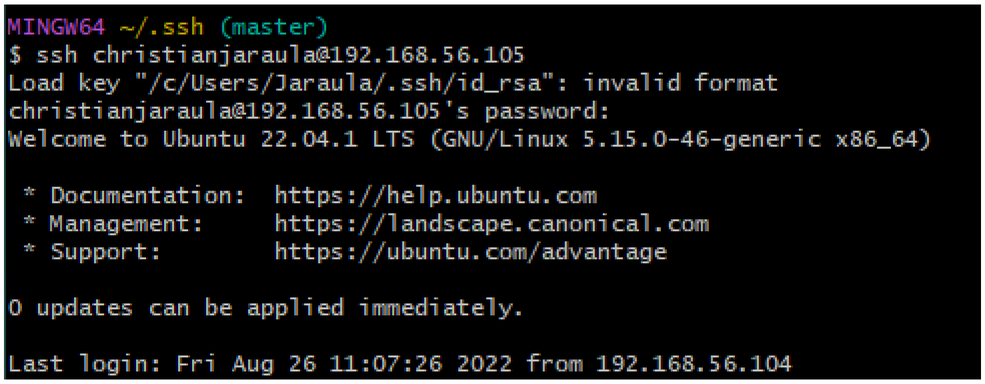
```
christianjaraula@Server1:~/.ssh$ cat id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAACAQDjJRnHuA0vYdfhclCOSXsPnxNu8EARJVl1NpqKMpz
TAehBjlOe6ua1Y0cb0eR5uLU5R2twUK3bLSJmp6/3Rzv6FkyVcfeLjSL8pEtkzPH15kkKYjLjRP4yVV
Z+ViEnwJeX1U3y+7qzGkWIXsucol1I3iBx/2k2PPEdkP4KCN2Ugh9sTSBPW5rvb4sdDc1nsUoEyZznH
zvE76oLFzqWvITH+GSQj0QmEK1vjrfIFbY1MiatSBEezj+qMHcRmk1IS0jiMNdC1xIEpAUGbPOi1n0b
FQXn5pYrAAi0gmp3G2417zwo9W873ysMM91wSScJ/i8nNtobJwuopQlrfj1DSDAZWJXP+a9nie/e+WY
yLvCcQngfCoLlZP1ri/AarJKvu7Nh+TH0rnBDBDJXwtDH8+CysK/usly2uUW14Jx8KFWIrd9/vSoqjs
sn0EclSlOFvPczvvbeTNyJMHezNICzHclcT9cXS8lphfkLuY0kCrP6U7Mwys2WM/hKA4iBUDerghxe9
V/NYHVv8Vzx+sBCHDGn/MzttyF0NifnktthRQwVvbM2ID9nz/W0gvkSggsauABgeqVaXPDc6FrZGUZQ
JnTMuoUAUiGyVOCuQ7wIGoHMBR2+NsAYAbXvwg95EnofrG07M1MjNEWaAGPsc2wELOOLV850v9NW5vY
z6geWVdMNnQ== christianjaraula@Server1
```

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

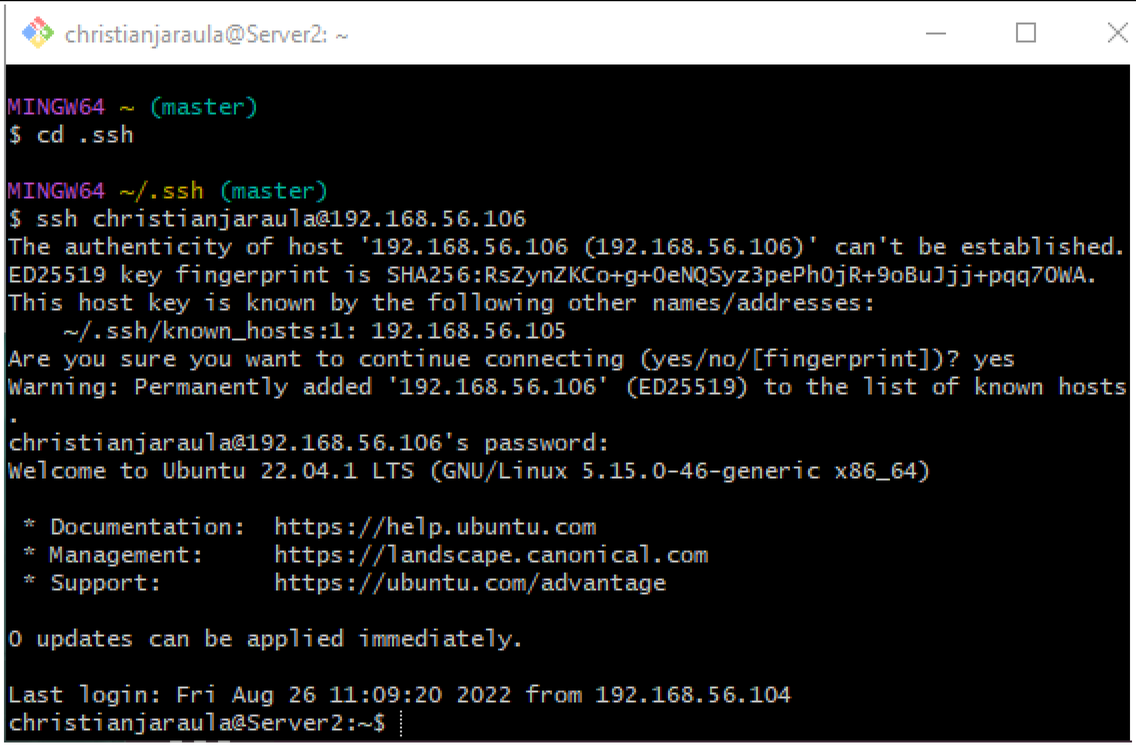


```
MINGW64 ~ (master)
$ ssh-copy-id christianjaraula@192.168.56.105
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/c/Users/Jaraula/
.ssh/id_rsa.pub"
The authenticity of host '192.168.56.105 (192.168.56.105)' can't be established.
ED25519 key fingerprint is SHA256:RsZynZKCo+g+0eNQSyz3pePh0jR+9oBuJjj+pqq70WA.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? y
Please type 'yes', 'no' or the 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 prompt
ed now it is to install the new keys
christianjaraula@192.168.56.105's password:

Number of key(s) added: 1

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



```
MINGW64 ~/.ssh (master)
$ ssh christianjaraula@192.168.56.105
Load key "/c/Users/Jaraula/.ssh/id_rsa": invalid format
christianjaraula@192.168.56.105'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 Aug 26 11:07:26 2022 from 192.168.56.104
```



```
MINGW64 ~ (master)
$ cd .ssh

MINGW64 ~/.ssh (master)
$ ssh christianjaraula@192.168.56.106
The authenticity of host '192.168.56.106 (192.168.56.106)' can't be established.
ED25519 key fingerprint is SHA256:RsZynZKCo+g+0eNQSyz3pePh0jR+9oBuJjj+pqq7OWA.
This host key is known by the following other names/addresses:
    ~/.ssh/known_hosts:1: 192.168.56.105
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.56.106' (ED25519) to the list of known hosts
.
christianjaraula@192.168.56.106'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 Aug 26 11:09:20 2022 from 192.168.56.104
christianjaraula@Server2:~$
```

No, because we duplicated the servers' authorisation keys so that the local host would recognize the server and not need to ask for the password.

Reflections:

Answer the following:

1. How will you describe the ssh-program? What does it do?
   Using the ssh-program makes it simpler to connect to and command servers. The local host that can link the two servers together and allow connections without requesting a password in the ssh-program.

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

   If you attempt to log in without providing a password, we will be able to determine if the public key has been deployed on the remote server.

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**).

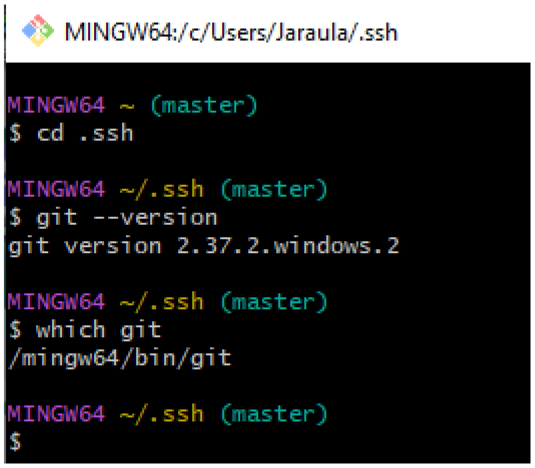
Set up Git

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:

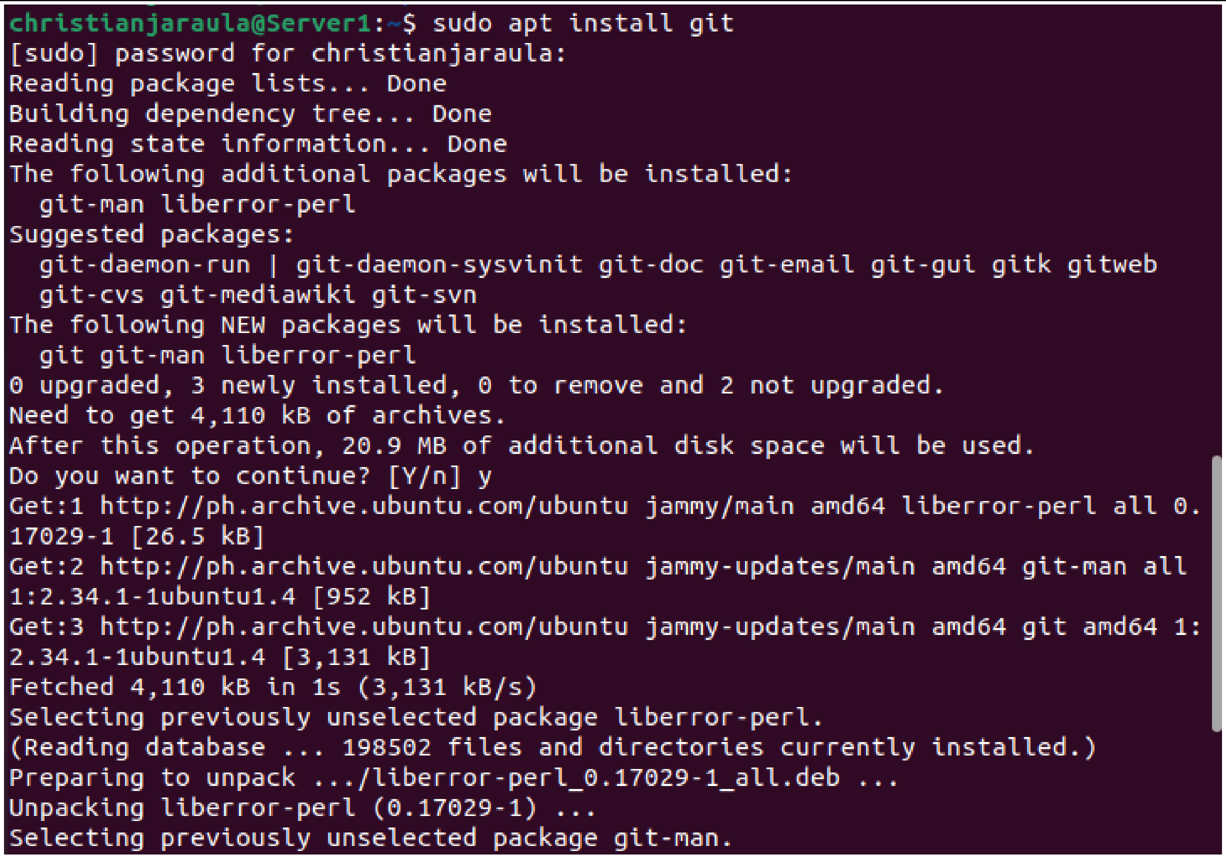
- Creating a repository
- Forking a repository
- Managing files
- Being social

Task 3: Set up the Git Repository

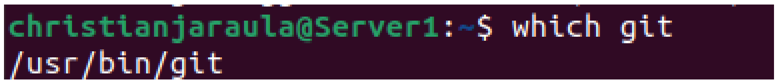
1. On the local machine, verify the version of your git using the command *which git.* If a directory of git is displayed, then you don't need to install git. Otherwise, to install git, use the following command: *sudo apt install git*



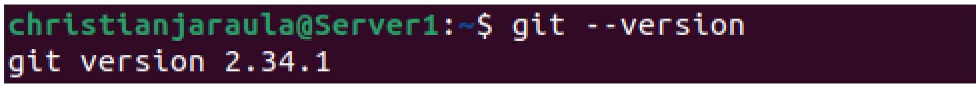
```
MINGW64:/c/Users/Jaraula/.ssh

MINGW64 ~ (master)
$ cd .ssh

MINGW64 ~/.ssh (master)
$ git --version
git version 2.37.2.windows.2

MINGW64 ~/.ssh (master)
$ which git
/mingw64/bin/git

MINGW64 ~/.ssh (master)
$
```


```
christianjaraula@Server1:~$ sudo apt install git
[sudo] password for christianjaraula:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
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 2 not upgraded.
Need to get 4,110 kB of archives.
After this operation, 20.9 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://ph.archive.ubuntu.com/ubuntu jammy/main amd64 liberror-perl all 0.
17029-1 [26.5 kB]
Get:2 http://ph.archive.ubuntu.com/ubuntu jammy-updates/main amd64 git-man all
1:2.34.1-1ubuntu1.4 [952 kB]
Get:3 http://ph.archive.ubuntu.com/ubuntu jammy-updates/main amd64 git amd64 1:
2.34.1-1ubuntu1.4 [3,131 kB]
Fetched 4,110 kB in 1s (3,131 kB/s)
Selecting previously unselected package liberror-perl.
(Reading database ... 198502 files and directories currently installed.)
Preparing to unpack .../liberror-perl_0.17029-1_all.deb ...
Unpacking liberror-perl (0.17029-1) ...
Selecting previously unselected package git-man.
```

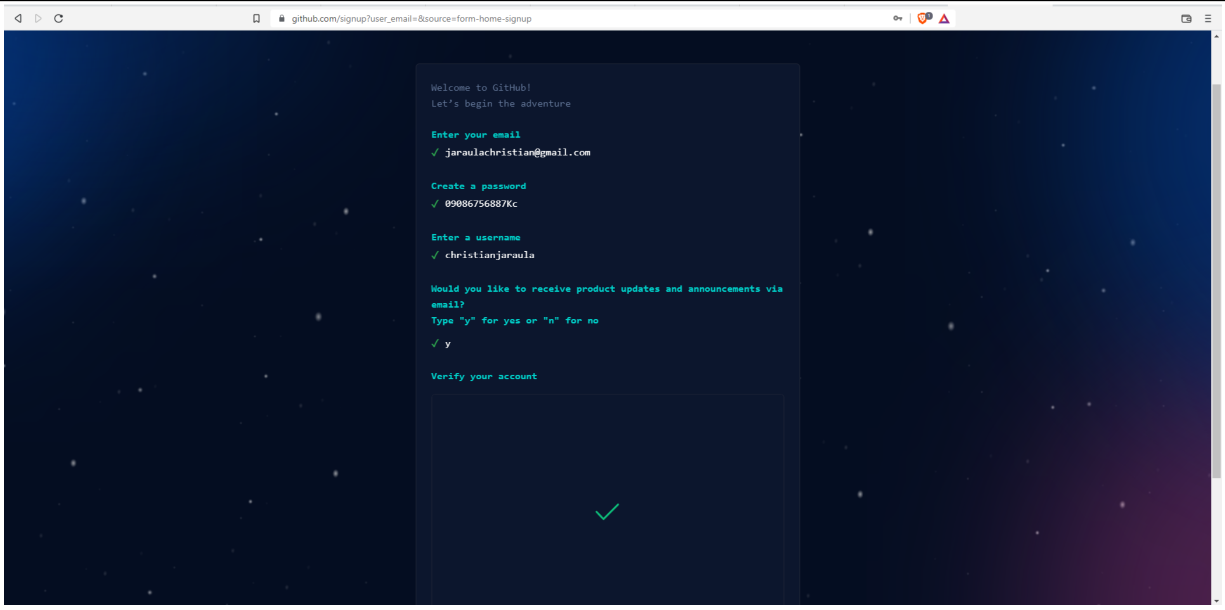
2. After the installation, issue the command *which git* again. The directory of git is usually installed in this location: *user/bin/git.*


```
christianjaraula@Server1:~$ 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.


```
christianjaraula@Server1:~$ git --version
git version 2.34.1
```

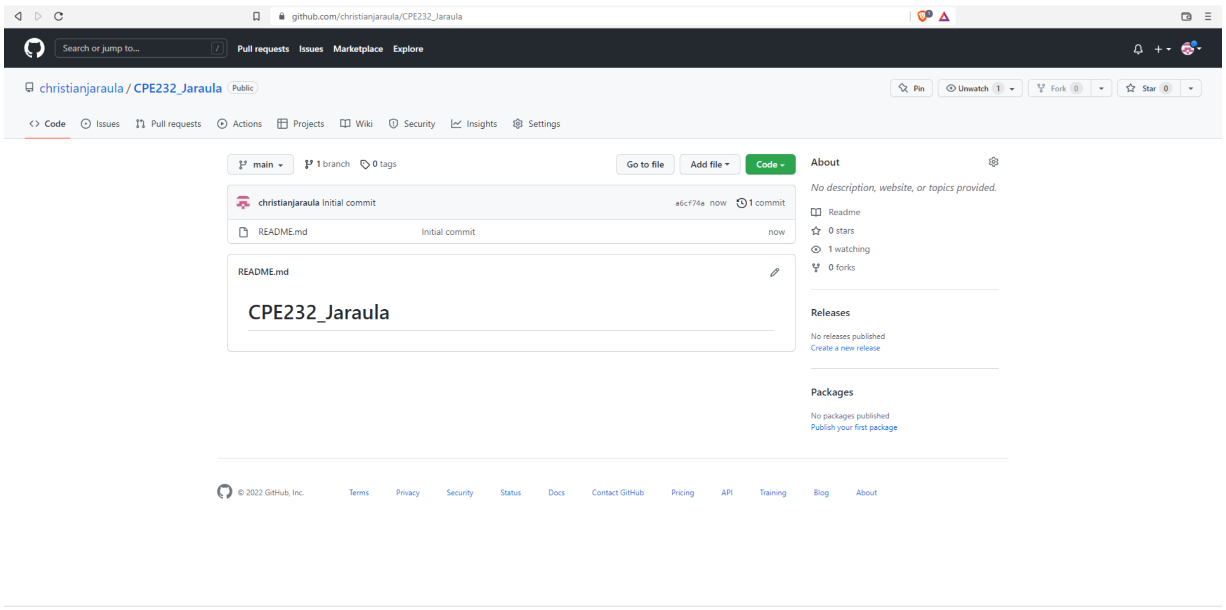
4. Using the browser in the local machine, go to www.github.com.
5. Sign up in case you don't have an account yet. Otherwise, login to your GitHub account.



github.com/signup?user_email=&source=form-home-signup

```
Welcome to GitHub!
Let's begin the adventure

Enter your email
✓ jaraulachristian@gmail.com

Create a password
✓ 09086756887Kc

Enter a username
✓ christianjaraula

Would you like to receive product updates and announcements via
email?
Type "y" for yes or "n" for no
✓ y

Verify your account
```

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



github.com/christianjaraula/CPE232_Jaraula

christianjaraula / CPE232_Jaraula Public

main · 1 branch · 0 tags

christianjaraula Initial commit · a6cf74a now · 1 commit

README.md · Initial commit · now

README.md

CPE232_Jaraula

About

No description, website, or topics provided.

Readme · 0 stars · 1 watching · 0 forks

Releases

No releases published
Create a new release

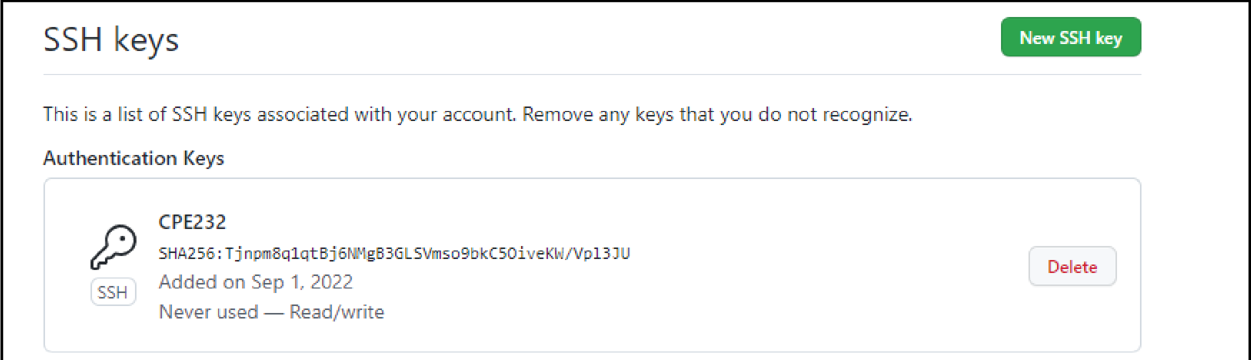
Packages

No packages published
Publish your first package

© 2022 GitHub, Inc.

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.



SSH keys

New SSH key

This is a list of SSH keys associated with your account. Remove any keys that you do not recognize.

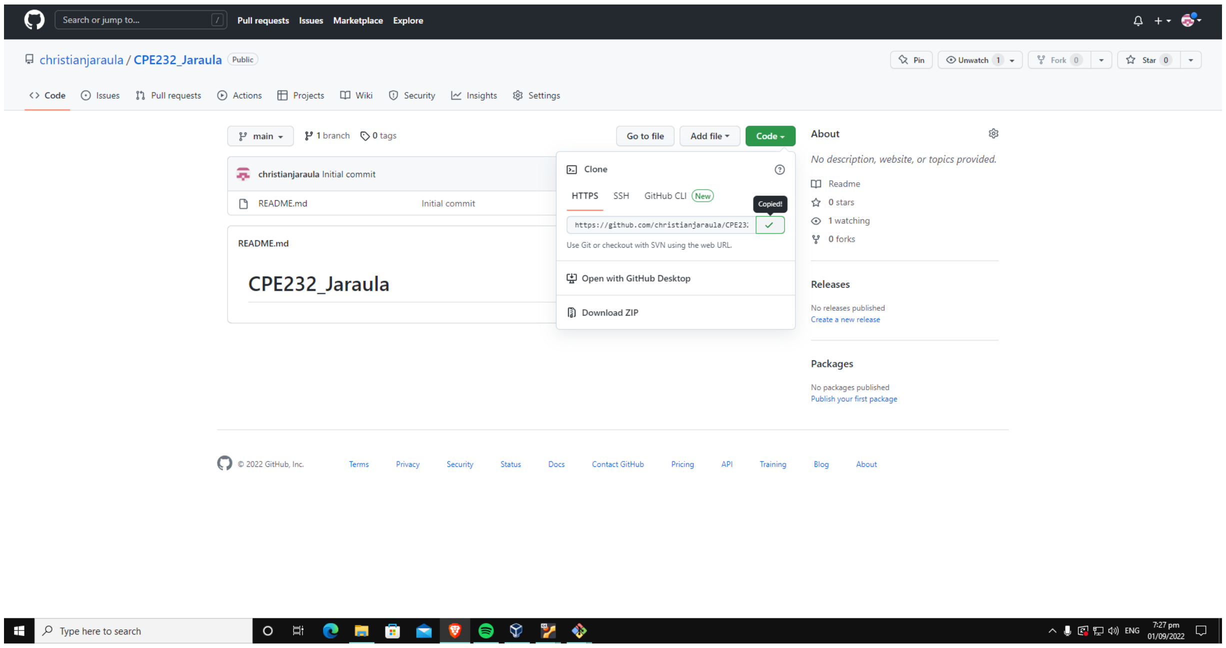
Authentication Keys

CPE232
SHA256:Tjnpm8q1qtBj6NMgB3GLSVmso9bkC5OiveKW/Vp13JU
Added on Sep 1, 2022
Never used — Read/write

SSH

Delete

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.



christianjaraula / CPE232_Jaraula Public

christianjaraula Initial commit

README.md Initial commit

README.md

CPE232_Jaraula

Clone

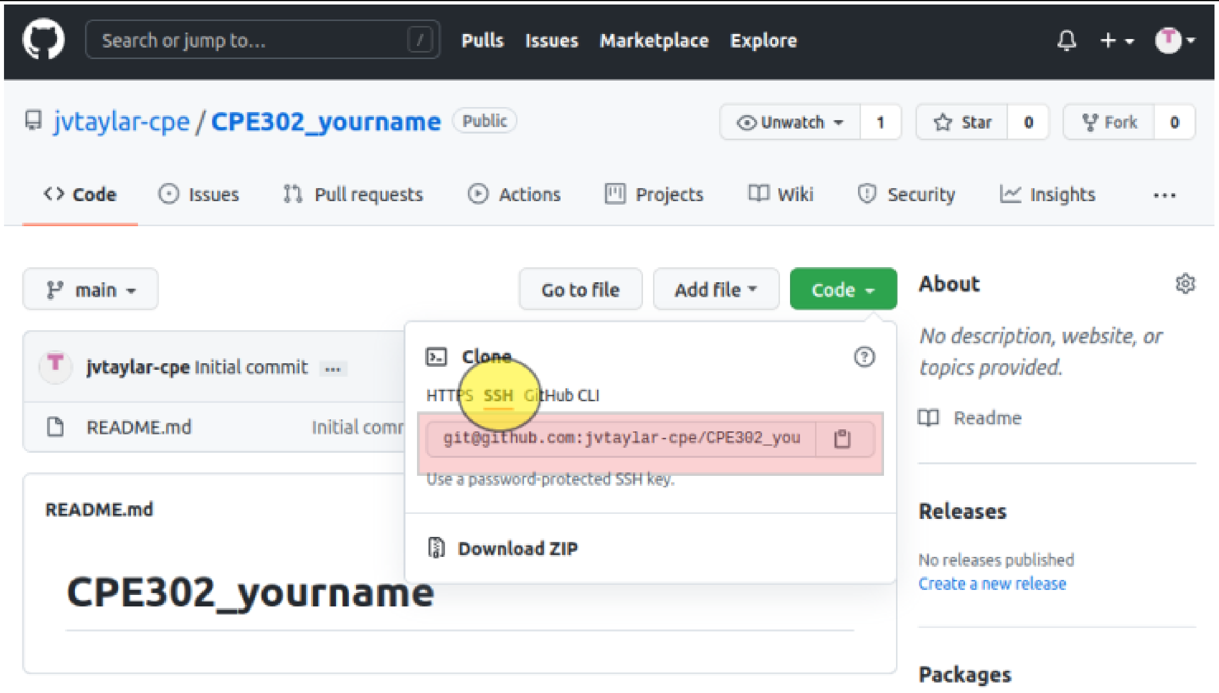
HTTPS SSH GitHub CLI New

```
https://github.com/christianjaraula/CPE23:
```

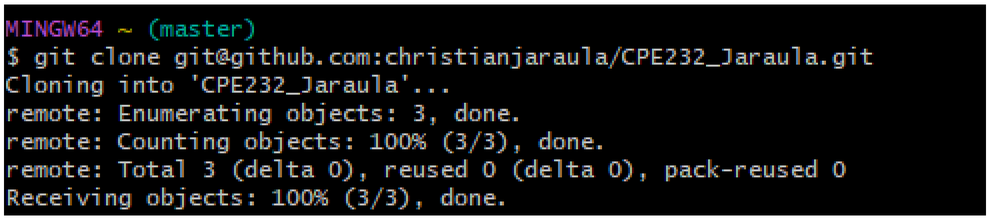
Use Git or checkout with SVN using the web URL.

About

No description, website, or topics provided.

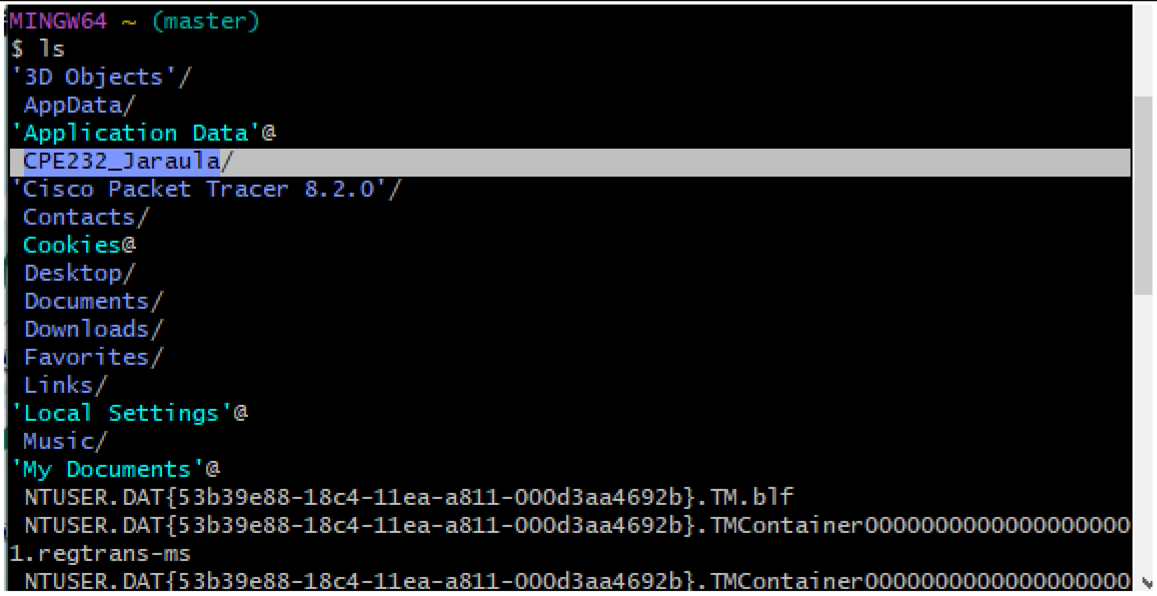


e. Issue the command git clone followed by the copied link. For example, *git clone* *git@github.com:jvtaylar-cpe/CPE232_yourname.git*. When prompted to continue connecting, type yes and press enter.



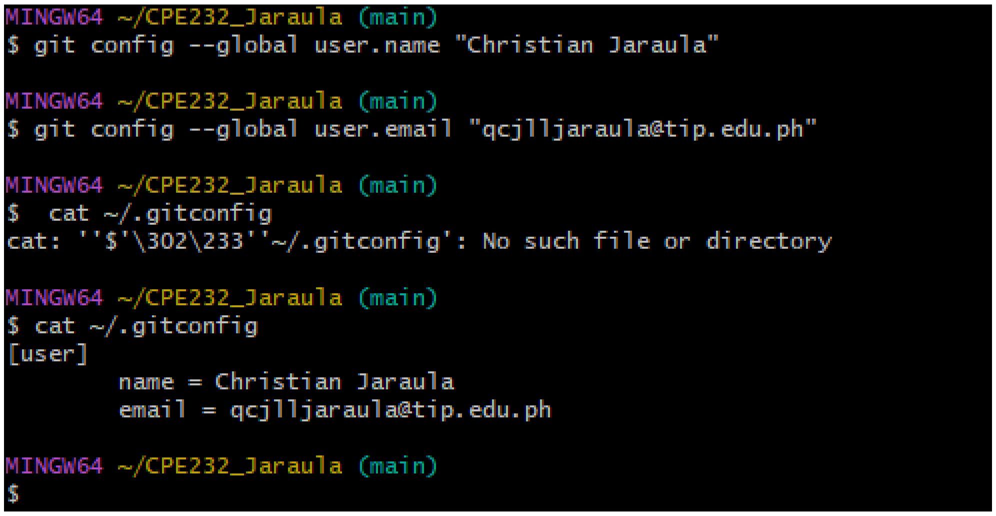
```
MINGW64 ~ (master)
$ git clone git@github.com:christianjaraula/CPE232_Jaraula.git
Cloning into 'CPE232_Jaraula'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.
```

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.



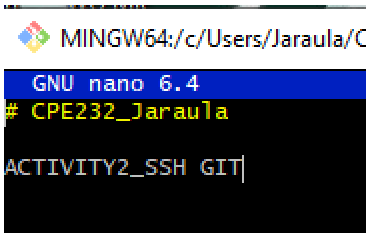
```
MINGW64 ~ (master)
$ ls
'3D Objects'/
 AppData/
'Application Data'@
 CPE232_Jaraula/
'Cisco Packet Tracer 8.2.0'/
 Contacts/
 Cookies@
 Desktop/
 Documents/
 Downloads/
 Favorites/
 Links/
'Local Settings'@
 Music/
'My Documents'@
 NTUSER.DAT{53b39e88-18c4-11ea-a811-000d3aa4692b}.TM.blf
 NTUSER.DAT{53b39e88-18c4-11ea-a811-000d3aa4692b}.TMContainer0000000000000000000
1.regtrans-ms
 NTUSER.DAT{53b39e88-18c4-11ea-a811-000d3aa4692b}.TMContainer0000000000000000000
```

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*



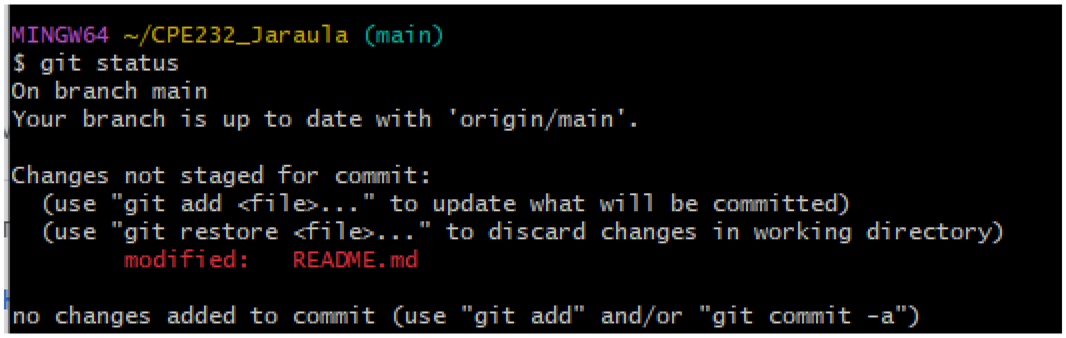
```
MINGW64 ~/CPE232_Jaraula (main)
$ git config --global user.name "Christian Jaraula"

MINGW64 ~/CPE232_Jaraula (main)
$ git config --global user.email "qcjlljaraula@tip.edu.ph"

MINGW64 ~/CPE232_Jaraula (main)
$  cat ~/.gitconfig
cat: ''$'\302\233''~/.gitconfig': No such file or directory

MINGW64 ~/CPE232_Jaraula (main)
$ cat ~/.gitconfig
[user]
        name = Christian Jaraula
        email = qcjlljaraula@tip.edu.ph

MINGW64 ~/CPE232_Jaraula (main)
$
```

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.



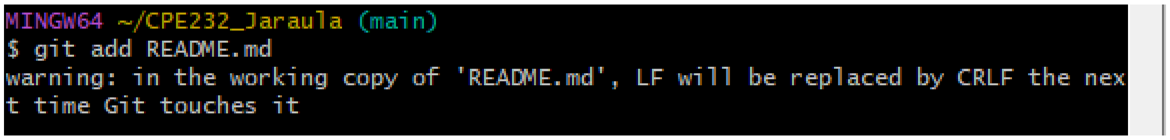
```
MINGW64:/c/Users/Jaraula/C
 GNU nano 6.4
# CPE232_Jaraula

ACTIVITY2_SSH GIT
```

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?



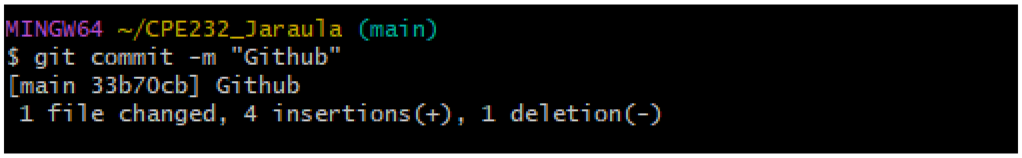
```
MINGW64 ~/CPE232_Jaraula (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

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.



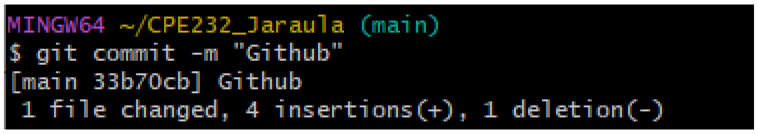
```
MINGW64 ~/CPE232_Jaraula (main)
$ git add README.md
warning: in the working copy of 'README.md', LF will be replaced by CRLF the nex
t time Git touches it
```

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.

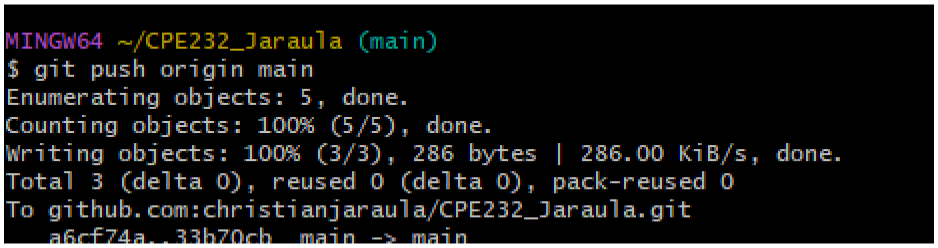


```
MINGW64 ~/CPE232_Jaraula (main)
$ git commit -m "Github"
[main 33b70cb] Github
 1 file changed, 4 insertions(+), 1 deletion(-)
```

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.*



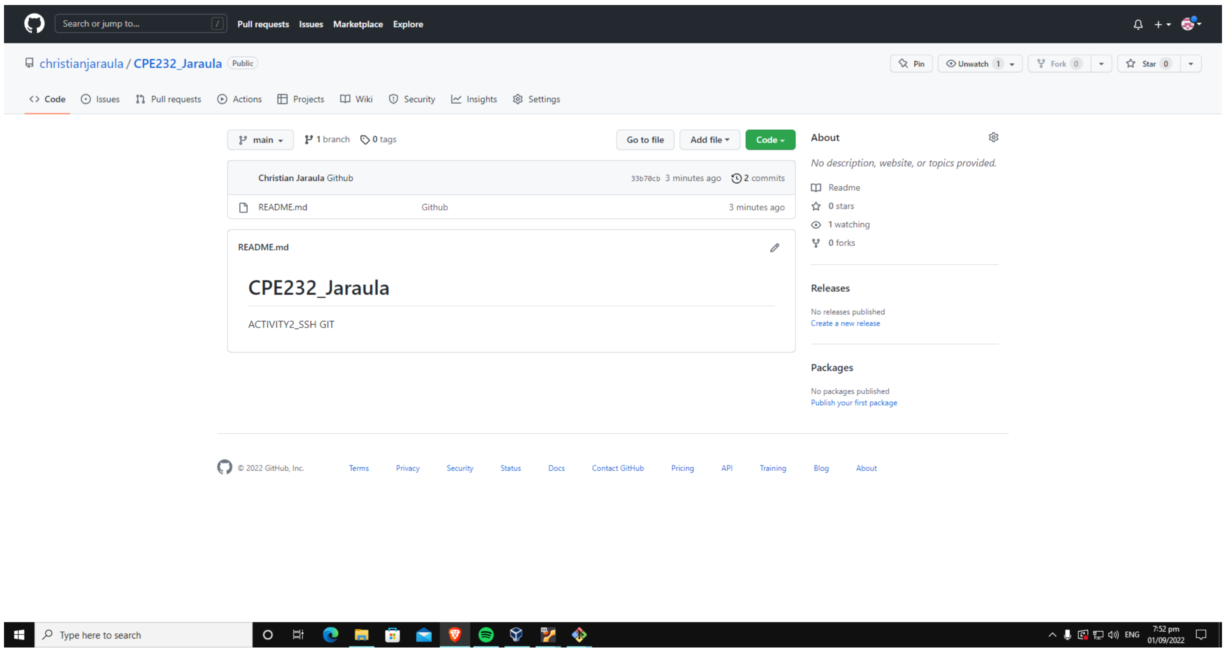
```
MINGW64 ~/CPE232_Jaraula (main)
$ git commit -m "Github"
[main 33b70cb] Github
 1 file changed, 4 insertions(+), 1 deletion(-)
```



```
MINGW64 ~/CPE232_Jaraula (main)
$ git push origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Writing objects: 100% (3/3), 286 bytes | 286.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:christianjaraula/CPE232_Jaraula.git
   a6cf74a..33b70cb  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.



Christian Jaraula Github 33b70cb 3 minutes ago 2 commits

README.md Github 3 minutes ago

README.md

CPE232_Jaraula

ACTIVITY2_SSH GIT

Reflections:

Answer the following:

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

establishing a connection between the local host and Github, creating a Github authorization key that may be used to prevent the server from demanding a password, and choosing to edit rather than read with the nano tool.

4. How important is the inventory file?

It is crucial to keep an inventory file so that you have a backup of anything you might accidentally modify or delete. If editing is done by group, having an inventory will make it simpler to keep track of what is being changed on the file.

Conclusions/Learnings:

I learned a lot about ssh keys, I know how to configure both local and distant computers using ssh using a key instead of a password and to establish a public key and a private key, and I know how to configure and use the git command.