**Aux Project 1: Shell Scripting**

A shell script is a text [file](https://whatis.techtarget.com/definition/file) that contains a sequence of commands for a [UNIX](https://searchdatacenter.techtarget.com/definition/Unix)-based [operating system](https://whatis.techtarget.com/definition/operating-system-OS). It is called a shell script because it combines a sequence of commands, that would otherwise have to be typed into the keyboard one at a time, into a single script. The [shell](https://searchdatacenter.techtarget.com/definition/shell) is the operating system's command-line interface ([CLI](https://searchwindowsserver.techtarget.com/definition/command-line-interface-CLI)) and interpreter for the set of commands that are used to communicate with the system. A shell interfaces between the user and the operating system.

A shell script is usually created for command sequences in which a user has a need to use repeatedly in order to save time. Like other programs, the shell script can contain [parameters](https://whatis.techtarget.com/definition/parameter), comments and subcommands that the shell must follow. Users initiate the sequence of commands in the shell script by simply entering the file name on a command line. Typical operations performed by shell scripts include file manipulation, program execution, and printing text.

In the [DOS](https://searchsecurity.techtarget.com/definition/DOS) operating system, a shell script is called a [batch file](https://searchwindowsserver.techtarget.com/definition/batch-file). In IBM's mainframe VM operating systems, it's called an EXEC.

### **How shell scripting works**

The basic steps involved with shell scripting are writing the script, making the script accessible to the shell and giving the shell execute permission.

Shell scripts contain [ASCII](https://whatis.techtarget.com/definition/ASCII-American-Standard-Code-for-Information-Interchange) text and are written using a [text editor](https://whatis.techtarget.com/definition/text-editor), [word processor](https://searchwindowsserver.techtarget.com/definition/word-processor) or graphical user interface ([GUI](https://searchwindevelopment.techtarget.com/definition/GUI)). The content of the script is a series of commands in a language that can be interpreted by the shell. Functions that shell scripts support include [loops](https://whatis.techtarget.com/definition/loop), variables, if/then/else statements, arrays and shortcuts. Once complete, the file is saved typically with a .txt or .sh extension and in a location that the shell can access.

### **Types of shells**

In Unix and [Linux](https://searchdatacenter.techtarget.com/definition/Linux-operating-system), the two major types of shell scripts are:

1. Bourne again shells ([BASH](https://searchdatacenter.techtarget.com/definition/bash-Bourne-Again-Shell))- BASH is the default shell for Unix version 7. The character for prompting a bourne again shell is $.
2. [C shells](https://whatis.techtarget.com/definition/C-shell)- A C shell is run in a text terminal window and is able to easily read file commands. The character for prompting a C shell is %.

**CSV File**

A Comma Separated Values (CSV) file is a plain text file that contains a list of data. These files are often used for exchanging data between different applications. For example, databases and contact managers often support CSV files. These files may sometimes be called Character Separated Values or Comma Delimited files

It is a delimited text file that uses a comma to separate values. A CSV file stores tabular data in plain text format. Each line of the file is a data record. You can use ‘[while shell loop](https://www.cyberciti.biz/faq/shell-script-while-loop-examples/)’ to read comma-separated CSV file. IFS variable will set CSV separated to , (comma). The read command will read each line and store data into each field.

They mostly use the comma character to separate (or delimit) data, but sometimes use other characters, like semicolons. The idea is that you can export complex data from one application to a CSV file, and then import the data in that CSV file into another application.

**Kernel**

A kernel interfaces between hardware and software. The Linux kernel is the main component of a Linux operating system (OS) and is the core interface between a computer’s hardware and its processes. It communicates between the 2, managing resources as efficiently as possible.

The kernel is so named because—like a seed inside a hard shell—it exists within the OS and controls all the major functions of the hardware, whether it’s a phone, laptop, server, or any other kind of computer.

**Functions of the Kernel**

1. Memory management: Keep track of how much memory is used to store what, and where
2. Process management: Determine which processes can use the central processing unit (CPU), when, and for how long
3. Device drivers: Act as mediator/interpreter between the hardware and processes
4. System calls and security: Receive requests for service from the processes

The kernel, if implemented properly, is invisible to the user, working in its own little world known as kernel space, where it allocates memory and keeps track of where everything is stored. What the user sees—like web browsers and [files](https://www.redhat.com/en/topics/data-storage/file-block-object-storage)—are known as the user space. These applications interact with the kernel through a system call interface (SCI).

The kernel is more like a busy personal assistant for a powerful executive (the hardware). It’s the assistant’s job to relay messages and requests (processes) from employees and the public (users) to the executive, to remember what is stored where (memory), and to determine who has access to the executive at any given time and for how long.

### Where the kernel fits within the OS

To put the kernel in context, you can think of a [Linux](https://www.redhat.com/en/topics/linux) machine as having 3 layers:

1. The hardware: The physical machine—the bottom or base of the system, made up of memory (RAM) and the processor or central processing unit (CPU), as well as input/output (I/O) devices such as [storage](https://www.redhat.com/en/topics/data-storage), [networking](https://www.redhat.com/en/topics/hyperconverged-infrastructure/what-is-software-defined-networking), and graphics. The CPU performs computations and reads from, and writes to, memory.
2. The Linux kernel: The core of the OS. (See? It’s right in the middle.) It’s software residing in memory that tells the CPU what to do.
3. User processes: These are the running programs that the kernel [manages](https://www.redhat.com/en/topics/management). User processes are what collectively make up user space. User processes are also known as just processes. The kernel also allows these processes and servers to communicate with each other (known as inter-process communication, or IPC).

In this project, we will onboard 20 new Linux users onto a server. Create a shell script that reads a csv file that contains the first name of the users to be onboarded.

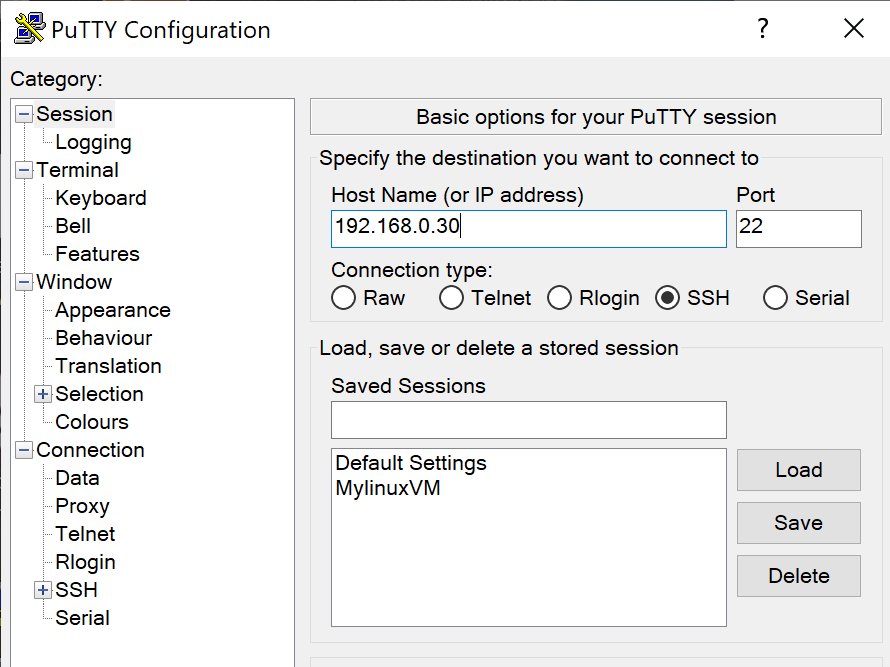
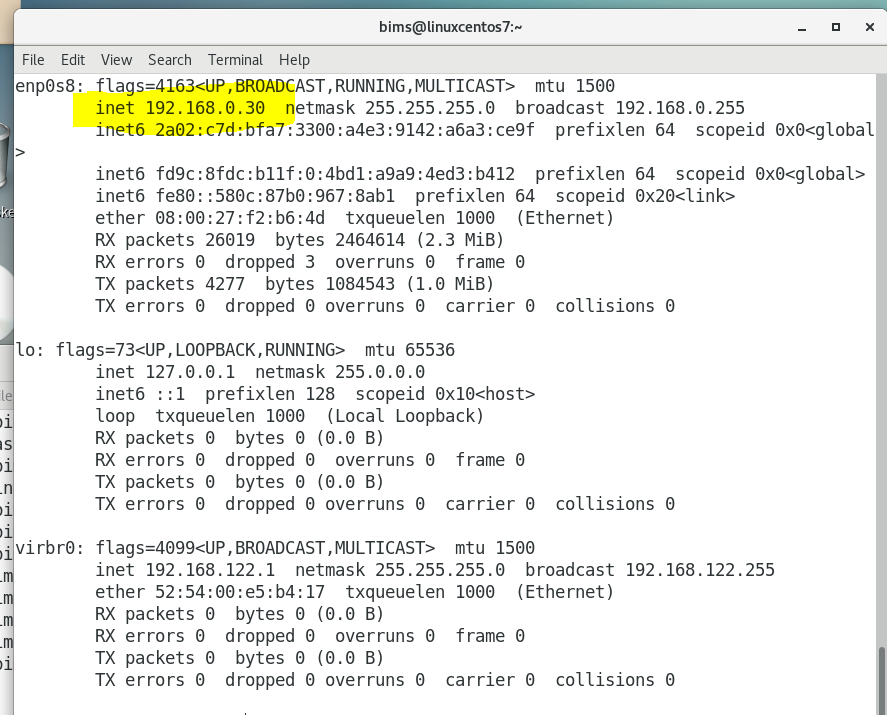
Created Virtual Box

Created virtual machine for Linux

Completed Guest Addition for Linux to allow copy and paste

Locate my IP address in Linux: ifconfig

Input IP address in Putty and press enter in order to open a session in Putty.



Steps taken include:

1. Create the project folder called Shell

mkdir Shell

1. Move into the Shell folder

cd Shell

1. Create a csv file name names.csv

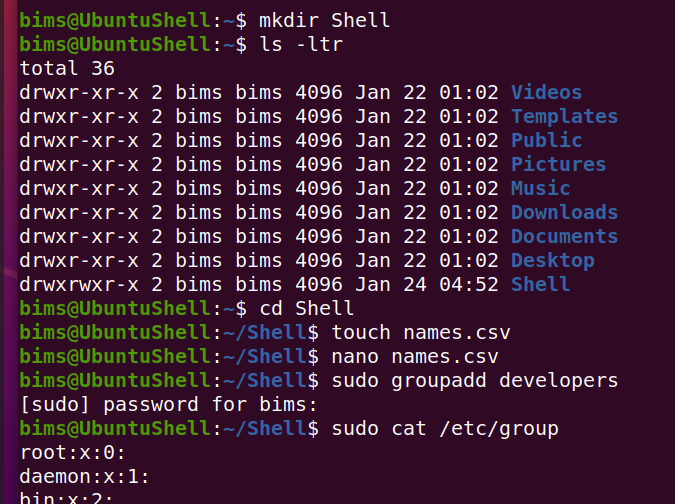
Touch names.csv

1. Open the names.csv file

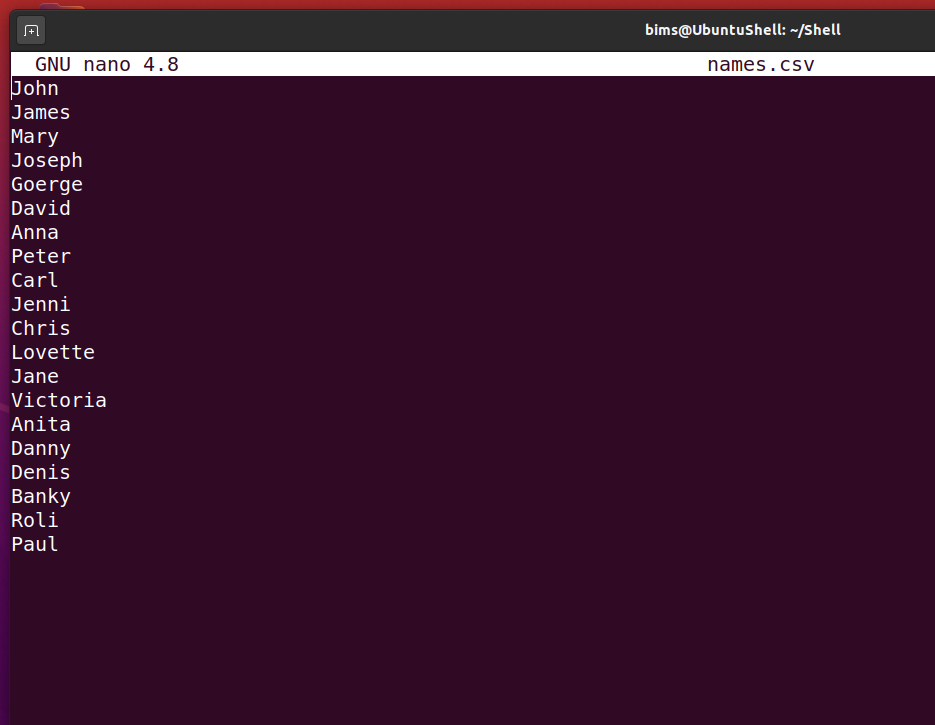
nano names.csv

Create a group called developers: sudo groupadd developers

Confirm the group has been created: sudo cat /etc/group



1. Insert some random names into it. (One name per line)



Create a script that will read the CSV file, create each user on the server, and add to an existing group called developers (You will need to manually create this group ahead).

Create a file called users: touch users

Create the script and add into the users file:- sudo nano users

#!/bin/bash

userfile=$(cat names.csv)

if [ $(id -u) -eq 0 ]; then

# Reading the CSV file

for user in $userfile;

do

echo $user

if id "$user" &>/dev/null

then

echo "User Exist"

else

# This will create a new user

useradd -m -d /home/$user -s /bin/bash -g developers $user

echo "New User Created"

echo

# This will create a ssh folder in the user home folder

su - -c "mkdir ~/.ssh" $user

echo ".ssh directory created for new user"

echo

# We need to set the user permission for the ssh dir

su - -c "chmod 700 ~/.ssh" $user

echo "user permission for .ssh directory set"

echo

# This will create an authorized-key file

su - -c "touch ~/.ssh/authorized\_keys" $user

echo "Authorized Key File Created"

echo

# We need to set permission for the key file

su - -c "chmod 600 ~/.ssh/authorized\_keys" $user

echo "user permission for the Authorized Key File set"

echo

# We need to create and set public key for users in the server

cp -R "/home/bims/.ssh/id\_rsa.pub" "/home/$user/.ssh/authorized\_keys"

echo "Copyied the Public Key to New User Account on the server"

echo

echo

echo "USER CREATED"

fi

done

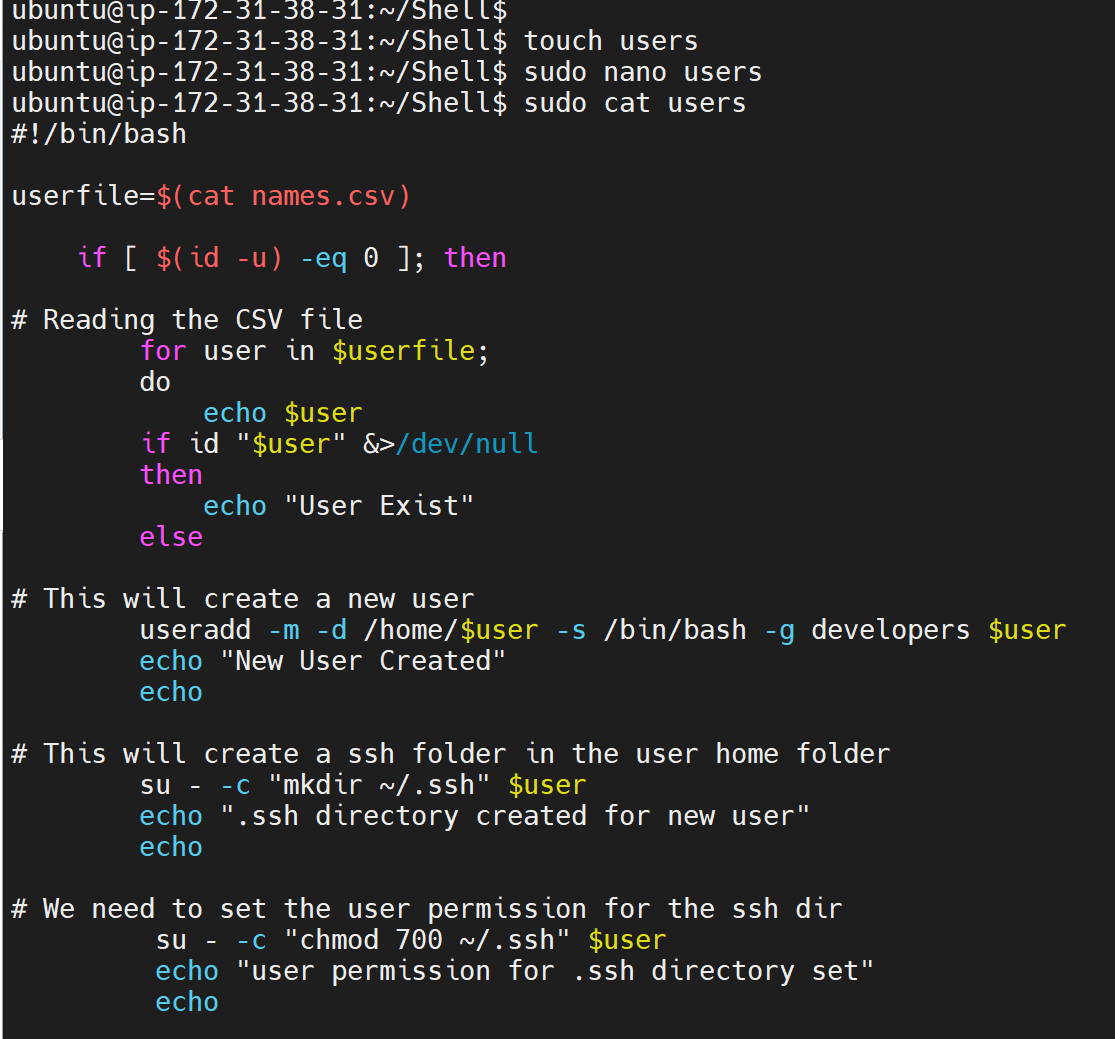
else

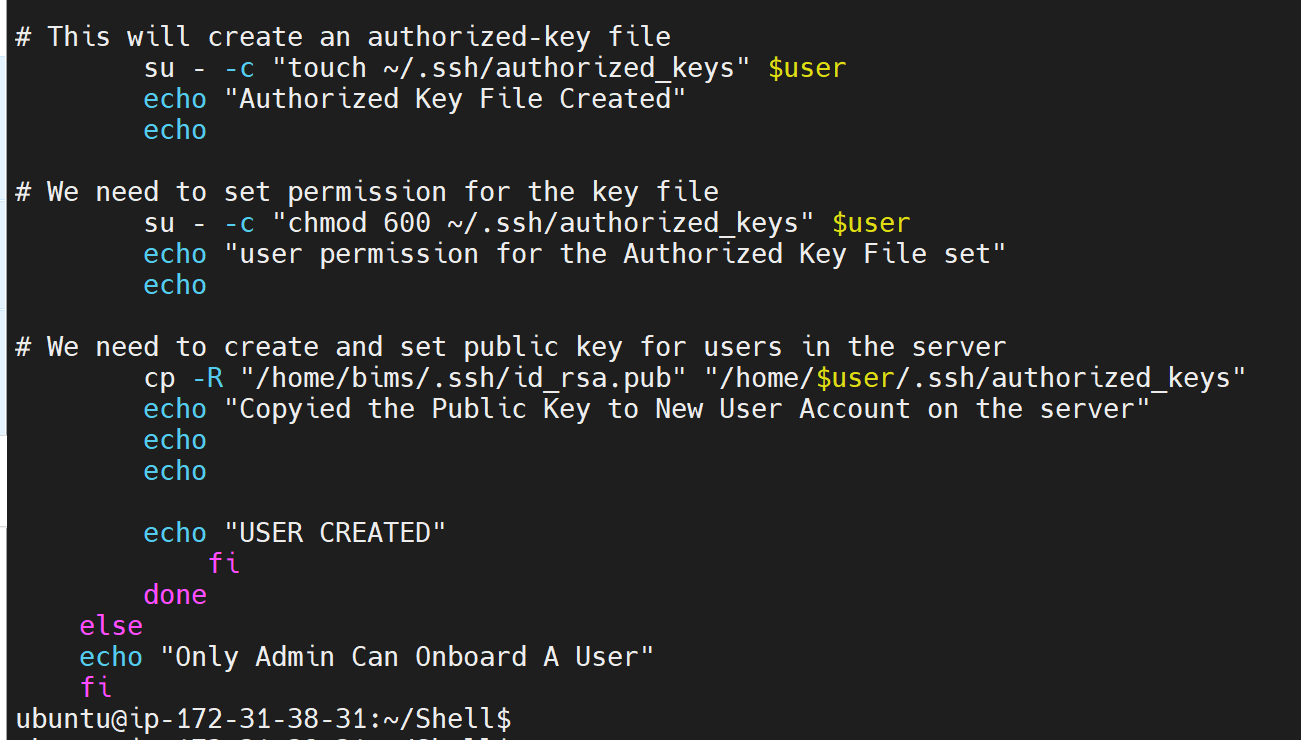
echo "Only Admin Can Onboard A User"

fi

Change ysf to bims to allow for connection to my name

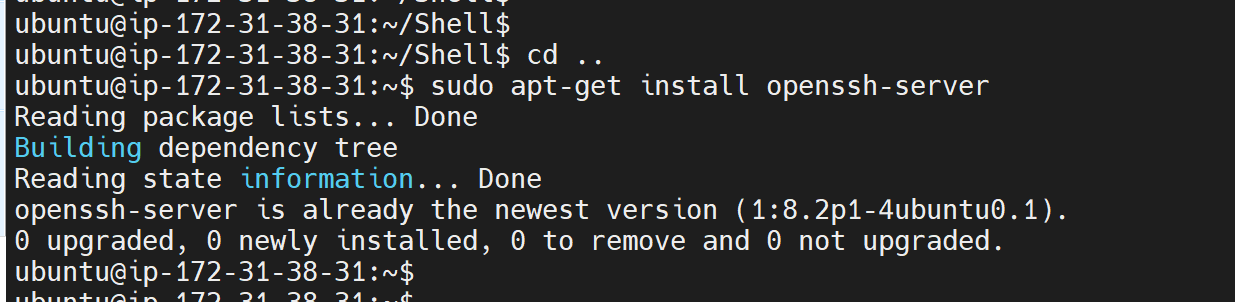
Check the users file: sudo cat users





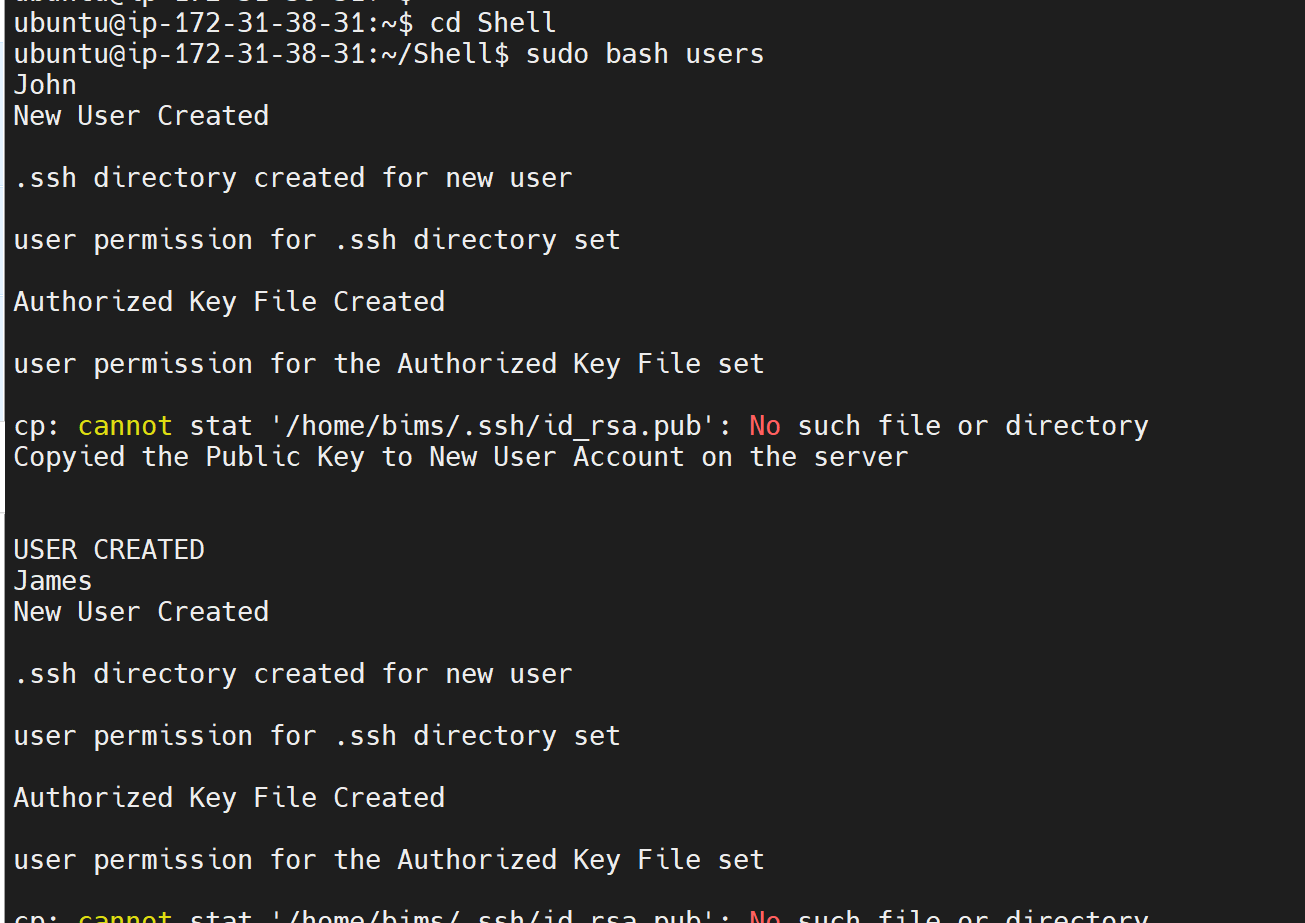
Install SSH: sudo apt-get install openssh-server

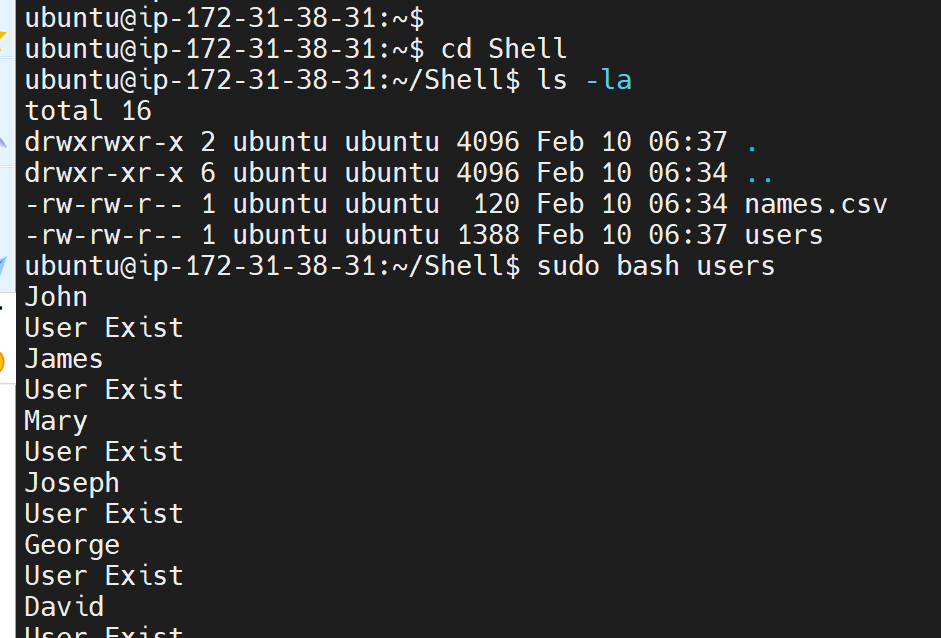
SSH folder: sudo nano /etc/ssh/sshd\_config



Then bashed the files so all the users’ files can be edited/ configured at the same time. This will create the users in the users file as a bash

sudo bash users





Ensure that your script will first check for the existence of the user on the system, before it will attempt to create that in.

Ensure that the user that is being created also has a default home folder

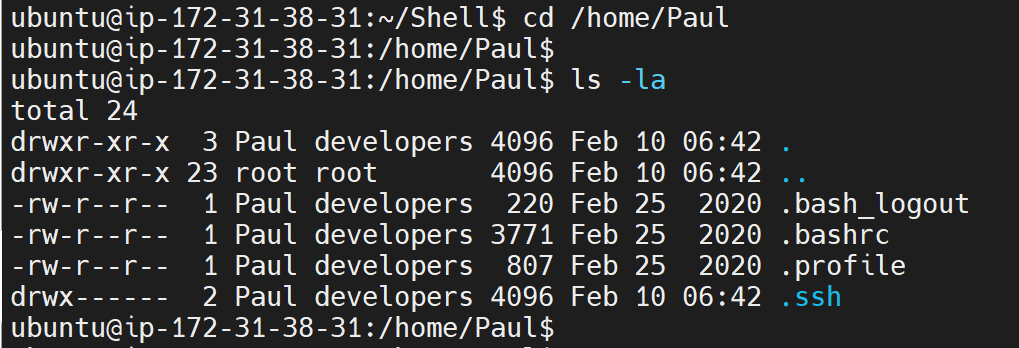
Ensure that each user has a .ssh folder within its HOME folder. If it does not exist, then create it.

For each user’s SSH configuration, create an authorized\_keys file and add the below public key.

Random user’s file to show that the file has been created inside of home directory and confirm the existence of .ssh file.

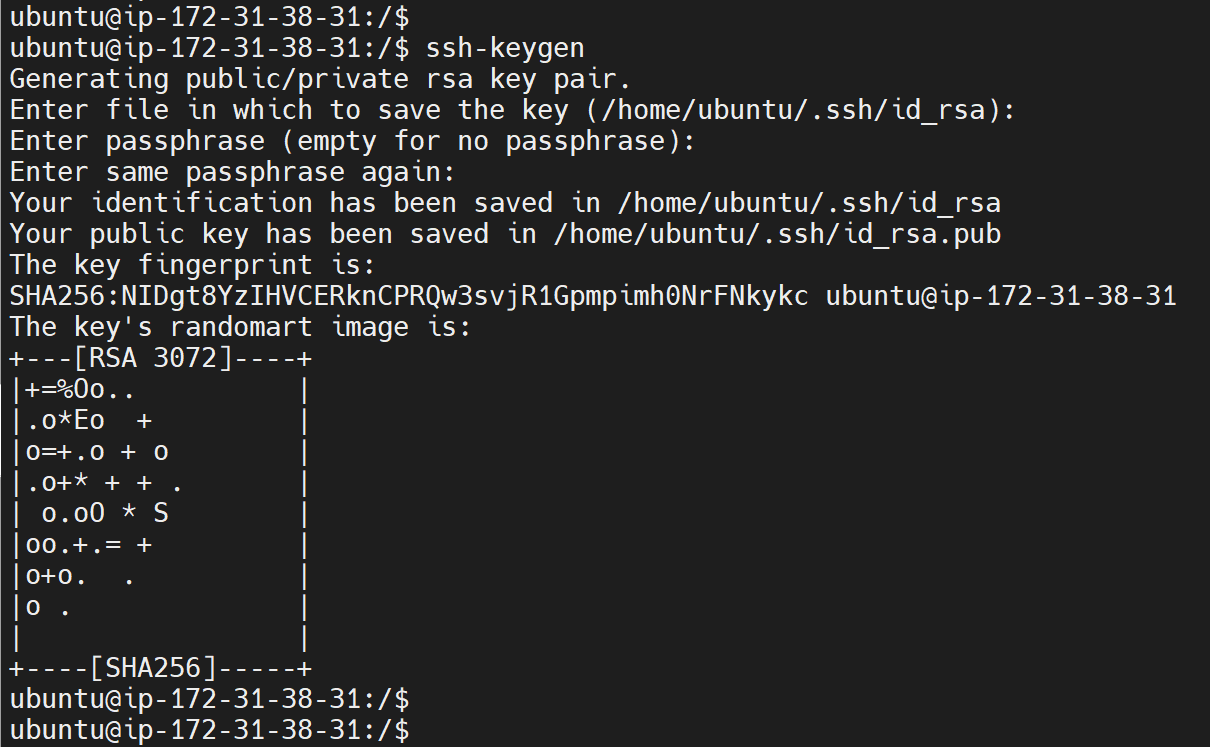
Go into the home directory of the user: $ cd /home/Paul

Then do ls -la



Next is to generate the authorisation keys both private and public. The public authorisation key will be added to each user’s folder in order to permit them access into the server.

Generate public/private rsa key pair for bims to enable the users to connect: ssh-keygen



cd into home and then cd into .ssh file

$ cd

$ cd .ssh

$ ls -la

Go into the private and public authorisation key files, delete and replace with the keys below

$ vi id\_rsa

$ vi id\_rsa.pub

**PUBLIC AUTHORIZATION KEY: vi id\_rsa.pub**

ssh-rsa 

**PRIVATE AUTHORIZATION KEY: $ vi id\_rsa**

-----BEGIN OPENSSH PRIVATE KEY-----

b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAABlwAAAAdzc2gtcn

NhAAAAAwEAAQAAAYEAsymconB8SJJJUqzZcbCcsk63CHQSLfOGdHEG90eJNIx+d5MZGk+Y

T/LX1k3uti14u71fOn2Pkl+0wTTO+lxgm8P+r6zBnoJuidcdotQIizcKuZ6k4zhp9ACMJL

uCJQgiwAe7XX0clbgmq5gZcJ8t5c0csmyFAFl0fw+r6TdPTLAyC2ynYWsvkP6fTlATEiz6

l3WP0CIfznt61b9ZcGNwKITX6u1d7TvPMyrMHwOYonsTYkmF9Zsk9Dq1/HARO+9Y/arqDW

YUIGiyNZiDLQL5x0Aoa5/ZNp1GjvwTAQkBqJeV8xr9WnaO4YKFT6JbiFA/hRjaP6R+M3Jg

yyAh+na12PwvTBEp5DFDyxcQVyjvzorQKWO9b0WqPZwIQMKVTLMuhaojm+bmsIb4aB2CB0

1f3+BLWoFWoETD9j8WDGEeo8pCFeDDyU5ApvpRoU8jJUbvqkUI3S5/AALR3+GhFfHSJKBo

dPPAyO3gP+KNMM+DleLqR+XsEhUvuSB5kRAhFuTXAAAFgIuJ0uiLidLoAAAAB3NzaC1yc2

EAAAGBALMpnKJwfEiSSVKs2XGwnLJOtwh0Ei3zhnRxBvdHiTSMfneTGRpPmE/y19ZN7rYt

eLu9Xzp9j5JftME0zvpcYJvD/q+swZ6CbonXHaLUCIs3CrmepOM4afQAjCS7giUIIsAHu1

19HJW4JquYGXCfLeXNHLJshQBZdH8Pq+k3T0ywMgtsp2FrL5D+n05QExIs+pd1j9AiH857

etW/WXBjcCiE1+rtXe07zzMqzB8DmKJ7E2JJhfWbJPQ6tfxwETvvWP2q6g1mFCBosjWYgy

0C+cdAKGuf2TadRo78EwEJAaiXlfMa/Vp2juGChU+iW4hQP4UY2j+kfjNyYMsgIfp2tdj8

L0wRKeQxQ8sXEFco786K0CljvW9Fqj2cCEDClUyzLoWqI5vm5rCG+GgdggdNX9/gS1qBVq

BEw/Y/FgxhHqPKQhXgw8lOQKb6UaFPIyVG76pFCN0ufwAC0d/hoRXx0iSgaHTzwMjt4D/i

jTDPg5Xi6kfl7BIVL7kgeZEQIRbk1wAAAAMBAAEAAAGAPf8KOpOeDibAxKEXZWXt8y2V3J

D9sXTxc92gwXS5n7t2D76REy+zzwaDdZ7mGZhGjQCMsVq9kbMYgzrY3H2W2I/L09J99XHA

+mW71Zp1kmbriSvCdvYQg+SkmhlggZv9GmISjdk7SPu+Nead9wC+CyUc5wjyRRqvW0B7Bm

qjQDBAQP/KM8W5Yf0Z9ylyT/nMhRijOSx1wSeta8WZF3DxYLQHWz3kILFvk48dryW5bZAV

Nw+mEUUsVm7yhnXpIMpDdl7wlDlqAWcuEQKJ7WJ7swuZM/FTQW4rFMmpDO8Q8PgijqOFDQ

jl8XfCPCkOhI9JOFTbmImTxfbRZ/NYYF09cFcqhKyvEi/Egx2oUZq4M81EGpP+EZnWgZtG

/PHqrSqIW166fixe/47eGCSt+AlyeR8SZCA1jjMRf7WB1RjANUHgC59tNTMQiFg+T5c2Yj

ORmPT0PpzEtQ+WMSMI5hGoklmqXuS5iiyJx7HyLOnK7wNloj7oqboz91wcCYnYWCORAAAA

wQDUbuGf0dAtJ4Qr2vdHiIi4dHAlMQMMsw/12CmpuSoqeEIWHVpAEBpqzx67qDZ+AMpBDV

BU9KbXe7IIzzfwUvxl1WCycg/pJM0OMjyigvz4XziuSVmSuy10HNvECvpxI3Qx8iF/HgAP

eyYe369FHEBsNZ5M5KhZ4oHI/XgZB5OGOaxErJd3wXhGASHnsWcmIswIjat7hH9WlAeWAk

/aeMz92iSDnYBOr+gAycsBm/skEDrN7dD45ilSvLZ6DQ2hbKAAAADBAOhLy9Tiki1IM2Gg

ma8KkUiLrqqx8IexPd580n7KsL32U2iu6Y88+skC8pkZQmIVG2UQhjiVLpNBgrzKKDJciK

/lyen21npQjuYaJPUgVUG0sjMtTpgGwbN/IVyHO28KSOogB6MclRBW7Z2SJggSAJaQmO9g

u7kieXbtf+5A7gUSb7icD629OiYCEJMTKTpVS/Pk7NDx/ZXQVzGrkJMKdPFU8nDoOjFLSP

jdbbddYe6zuB/HwabV3Lpaxl38tNG78wAAAMEAxXHS2IRABAvX7+OmZO2JU7+9Gxh/gudJ

eXf76c10kKvUztoe8Mskw79yVq6LtYd0JGOVx0oNgMeZJHmwUc2qVPKaFGEhSG6MuFn3J2

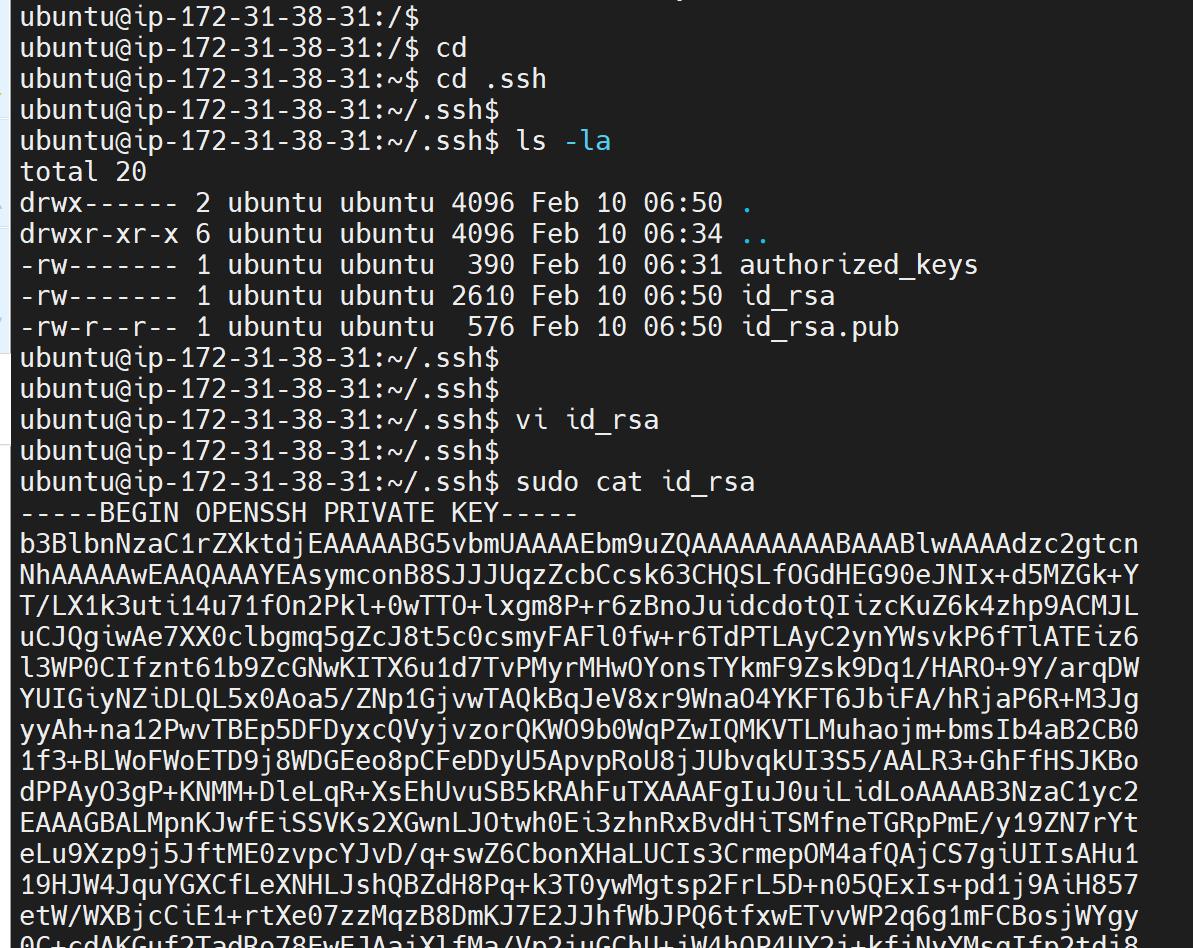
O5+Kt+KfU5M9uAN7tob3+yG18ZJt9FY+5FTK1TV5LmF5OTGBN9XyehT2Miqa8sSu80rwpN

nhe+U/XswAp9KEVYkSIjFeoy/amsOP+qvRke1dKWBsU12IbhnMgjDHVggkYV52l7d9S2bx

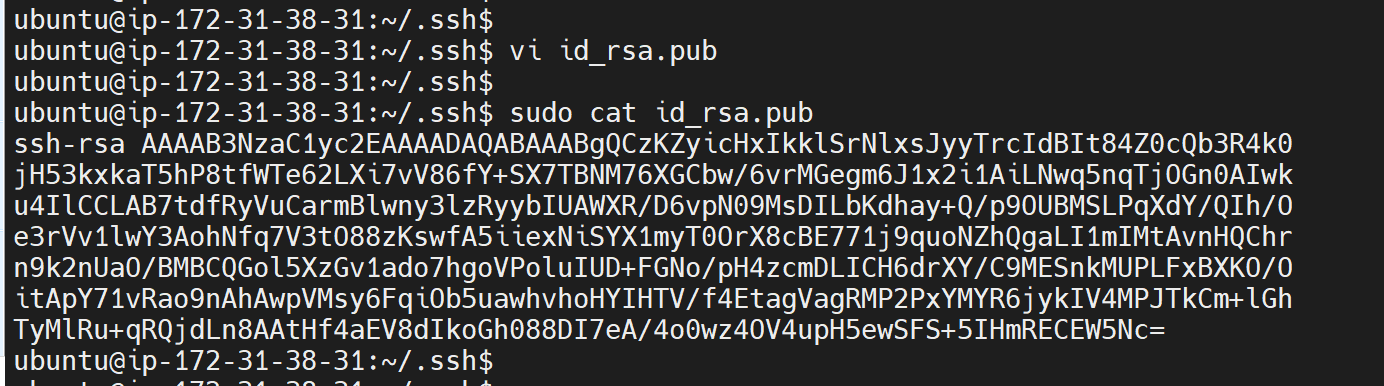
kmaSGj362OnCCNAAAACWRhcmVARGFyZQE=

-----END OPENSSH PRIVATE KEY-----

Private Key



Public key



Test a few of the users randomly, and ensure that you are able to connect to the server using the above public key.

Using Paul:

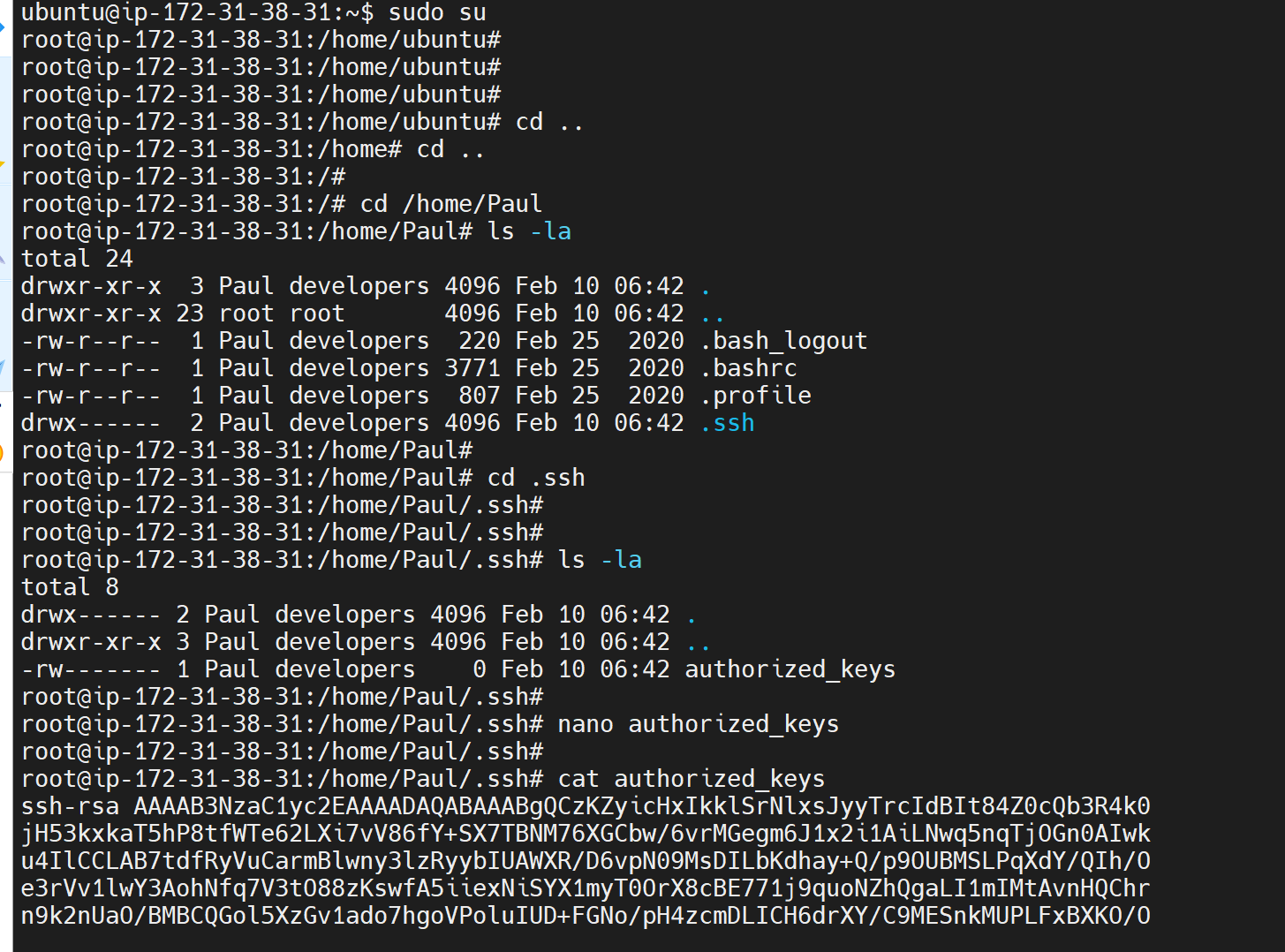
Need to add the Server’s public authorization key to Paul’s authorisation Key folder to enable access into the server.

Go into Pauls’s folder: cd /home/Paul

cd into ssh: cd .ssh

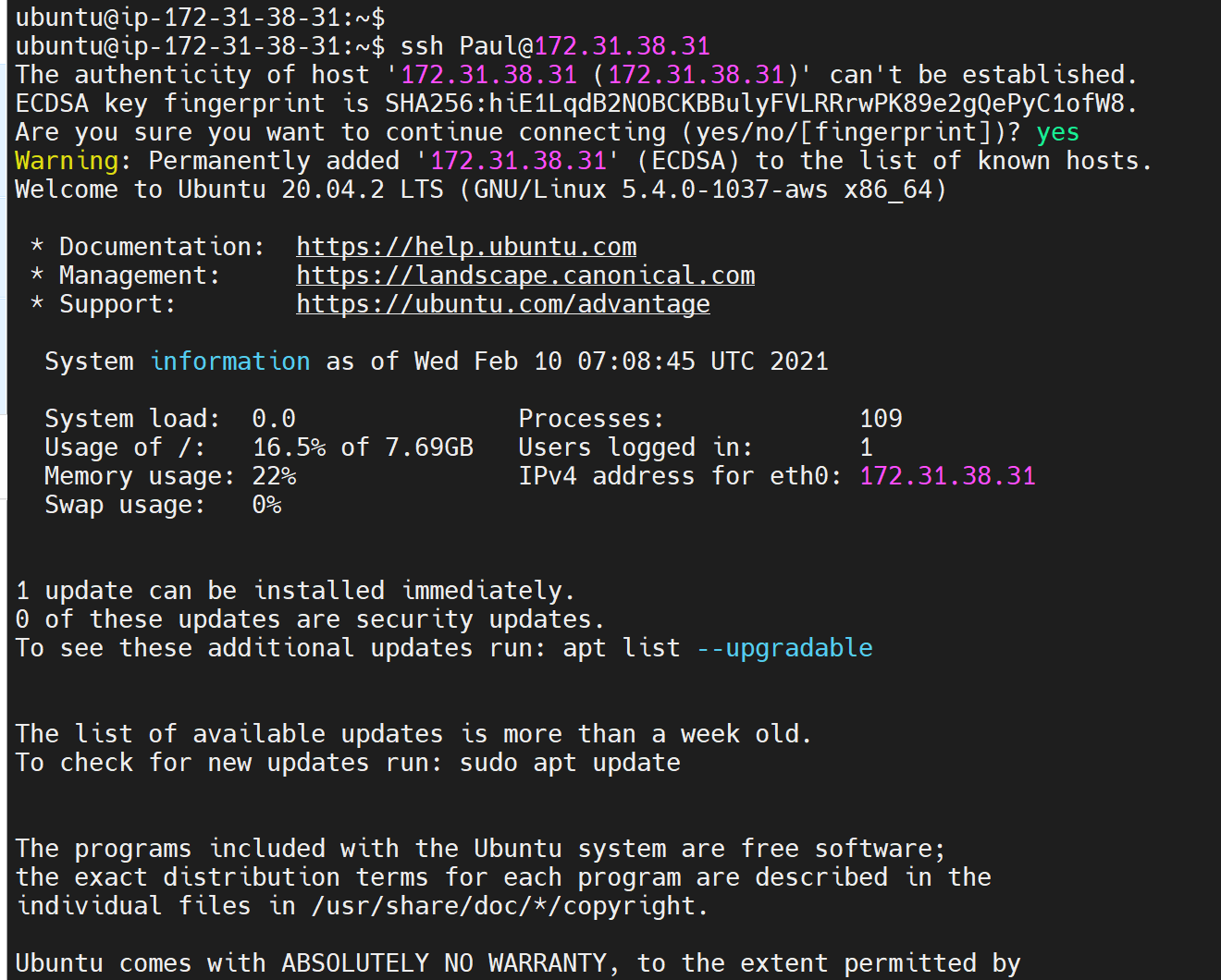
Access denied, then work as root user: sudo su

Then add the server’s public key (above) into Paul’s authorisation key folder: nano authorized\_keys

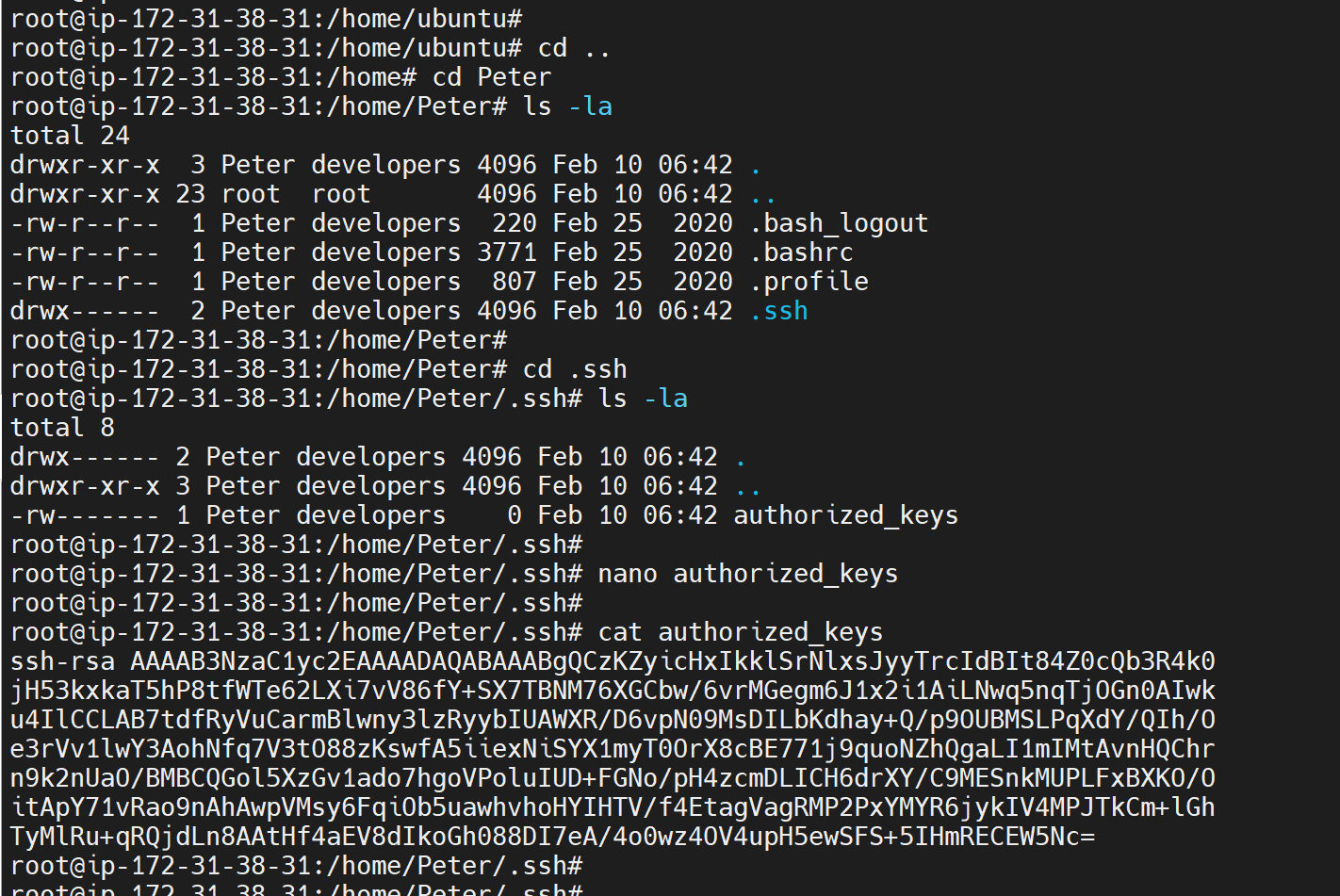


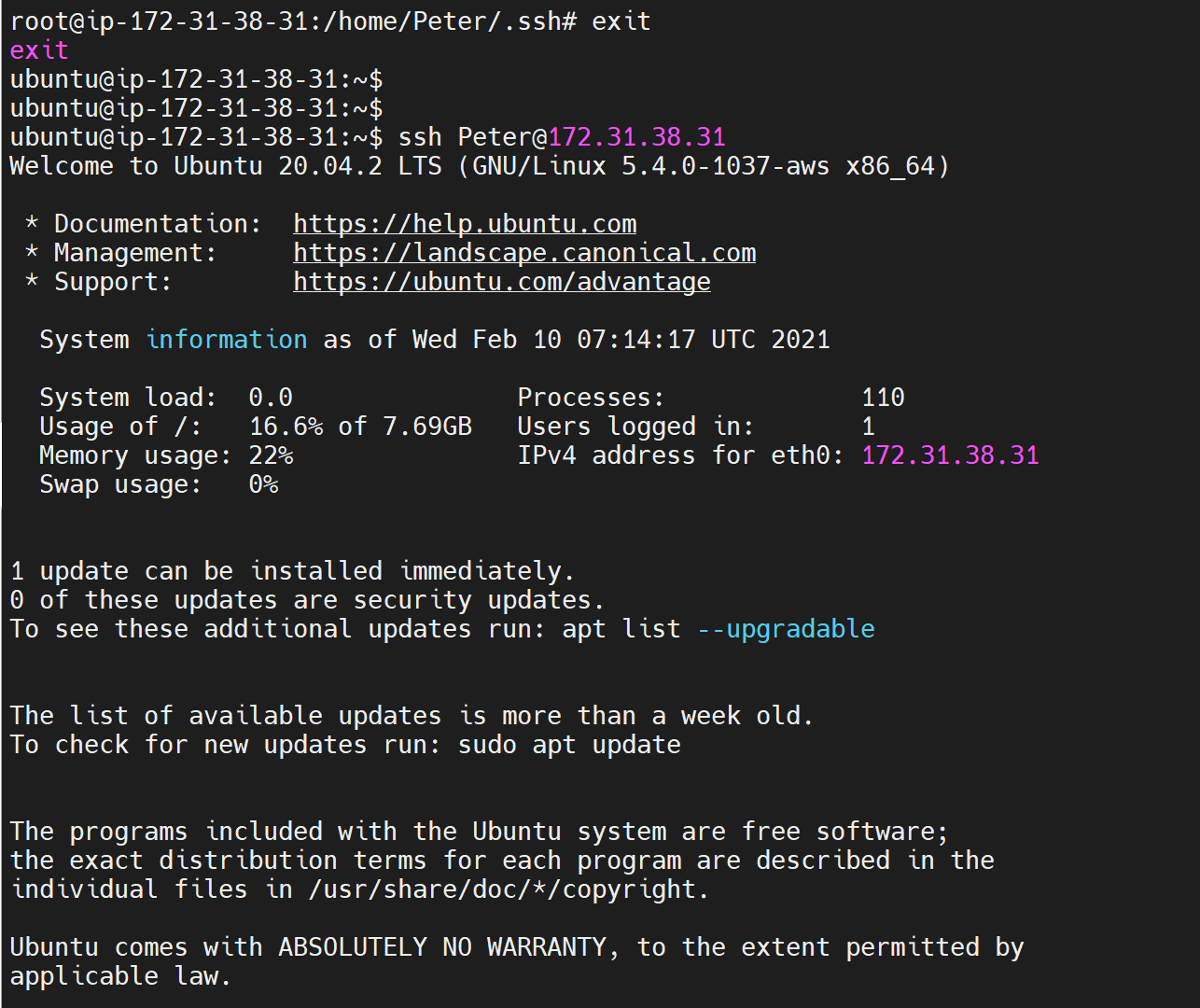
Log out from the root.

Enable Paul access the server: $ ssh Paul@172.31.38.31



Repeat the same for another user - Peter





Users - Paul and Peter have been able to connect to the server successfully.

**Created a Repository on Github to move my project with Ubuntu**

**Name:auxiliary-project**

**Description: Shell Scripting**

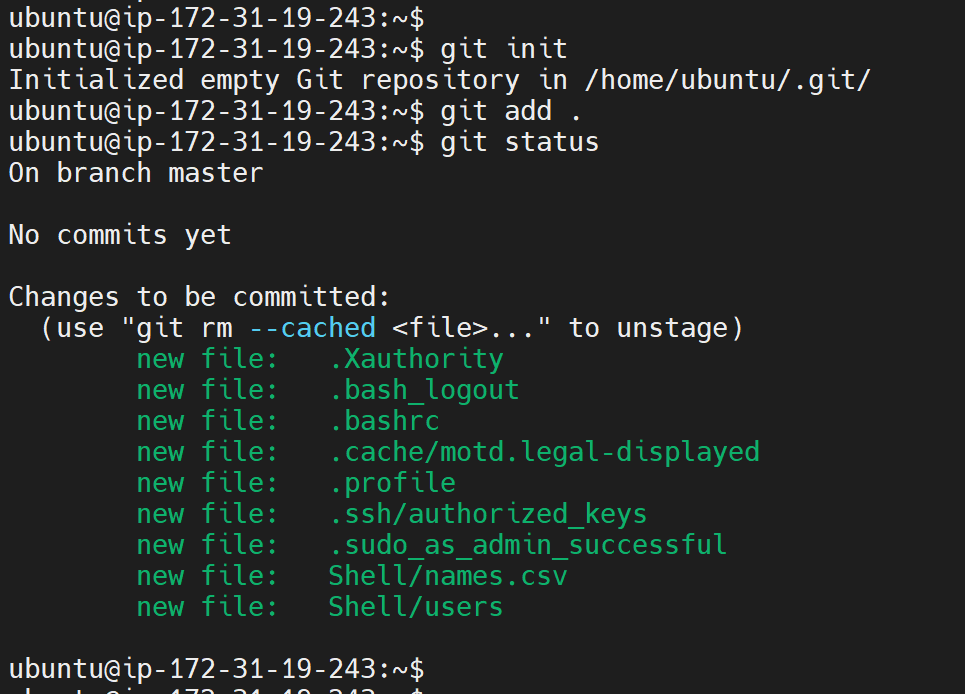
Steps:

Opened an account with Github and created a repository

Install git into terminal: git init

Add the folders you want to move: git add .

Check the status: git status

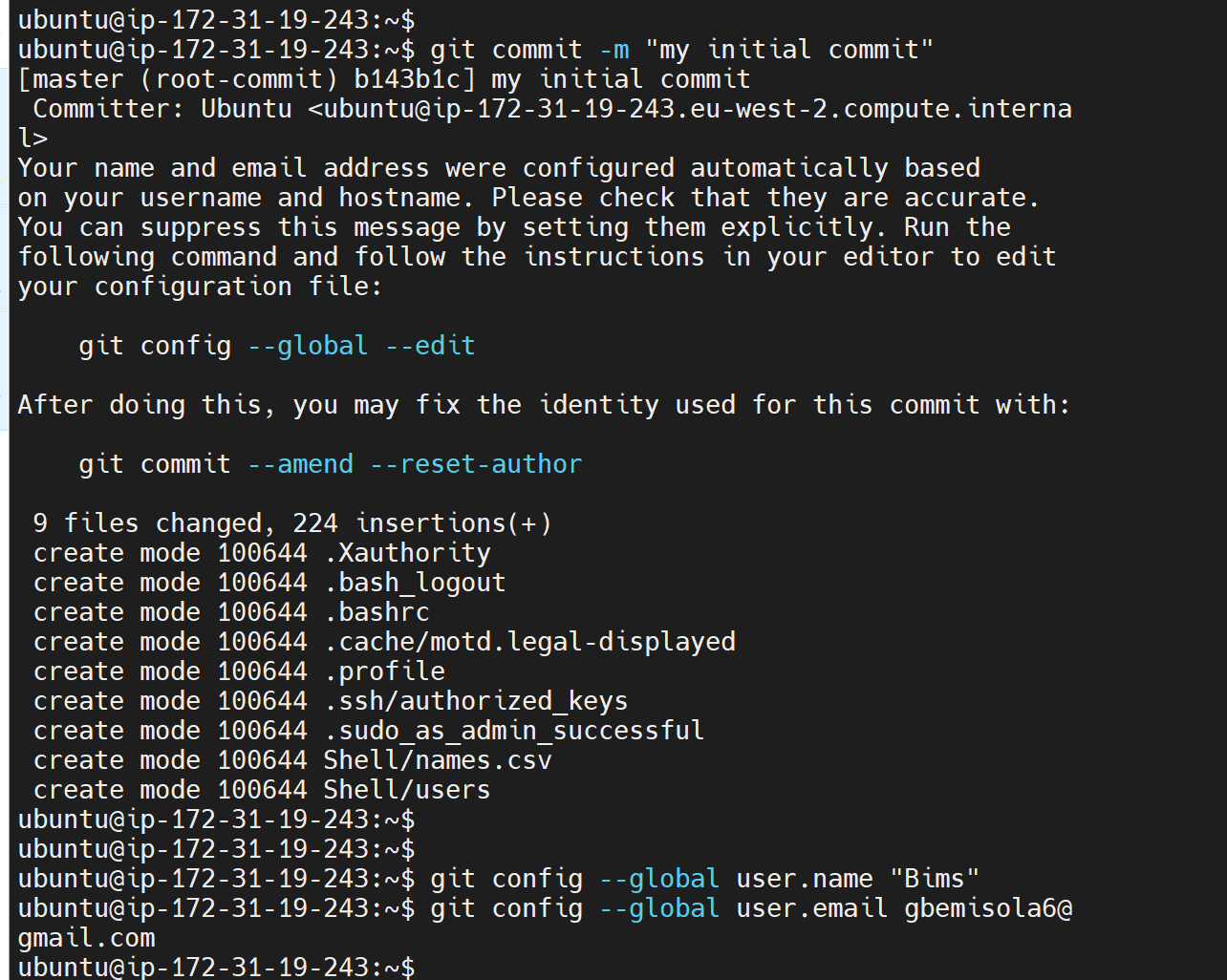


Commit: git commit -m "my initial commit"

Add your name and email for identification purposes:

git config --global user.name "Bims"

git config --global user.email [gbemisola6@gmail.com](mailto:gbemisola6@gmail.com)

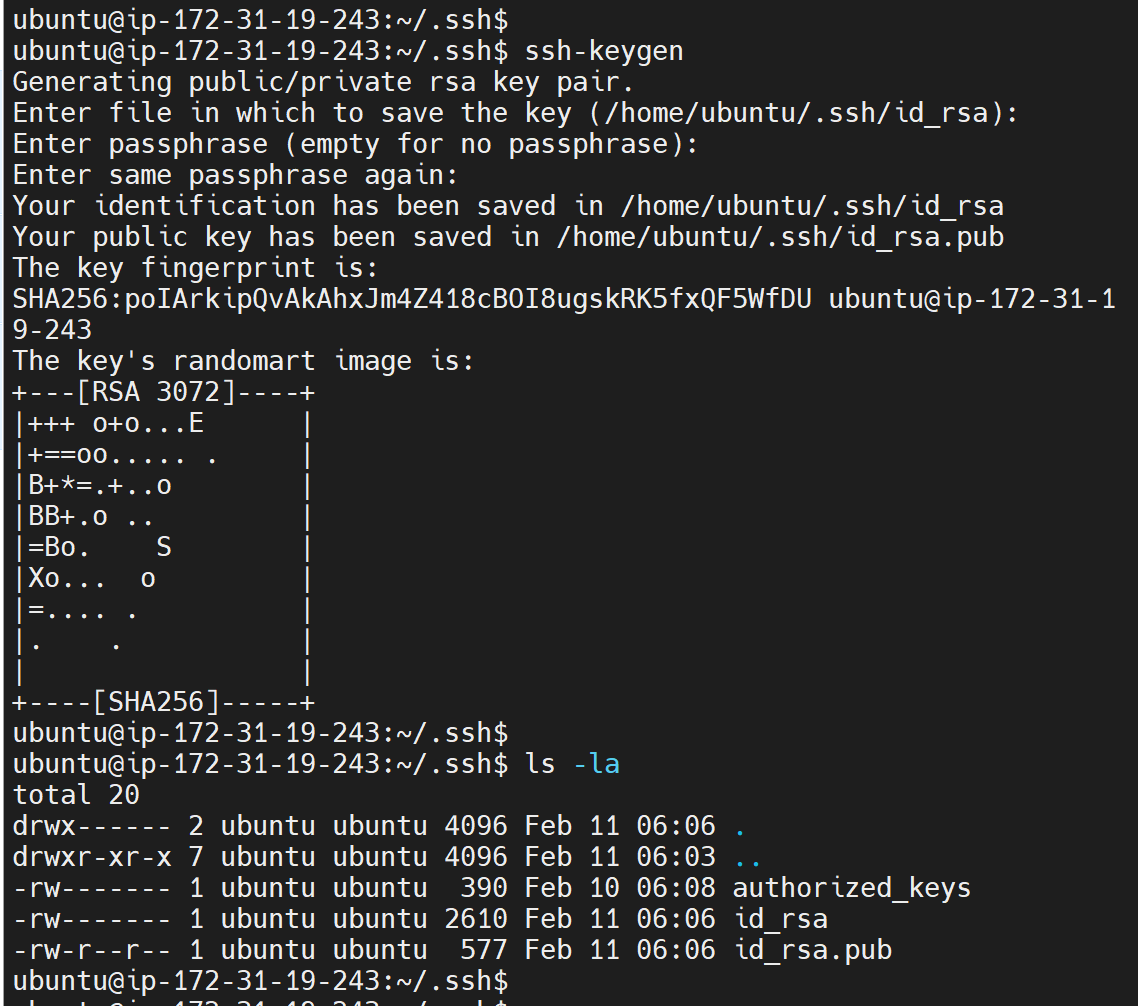


Generate ssh key:

$ cd .ssh

$ ssh-keygen

$ ls -la



$ cat ~/.ssh/id\_rsa.pub

Copy the public file: vi ~/.ssh/id\_rsa.pub

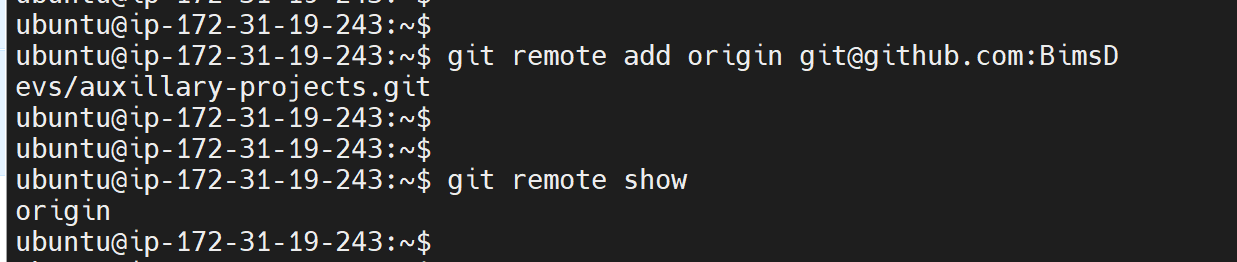
Exit .ssh folder: cd ..



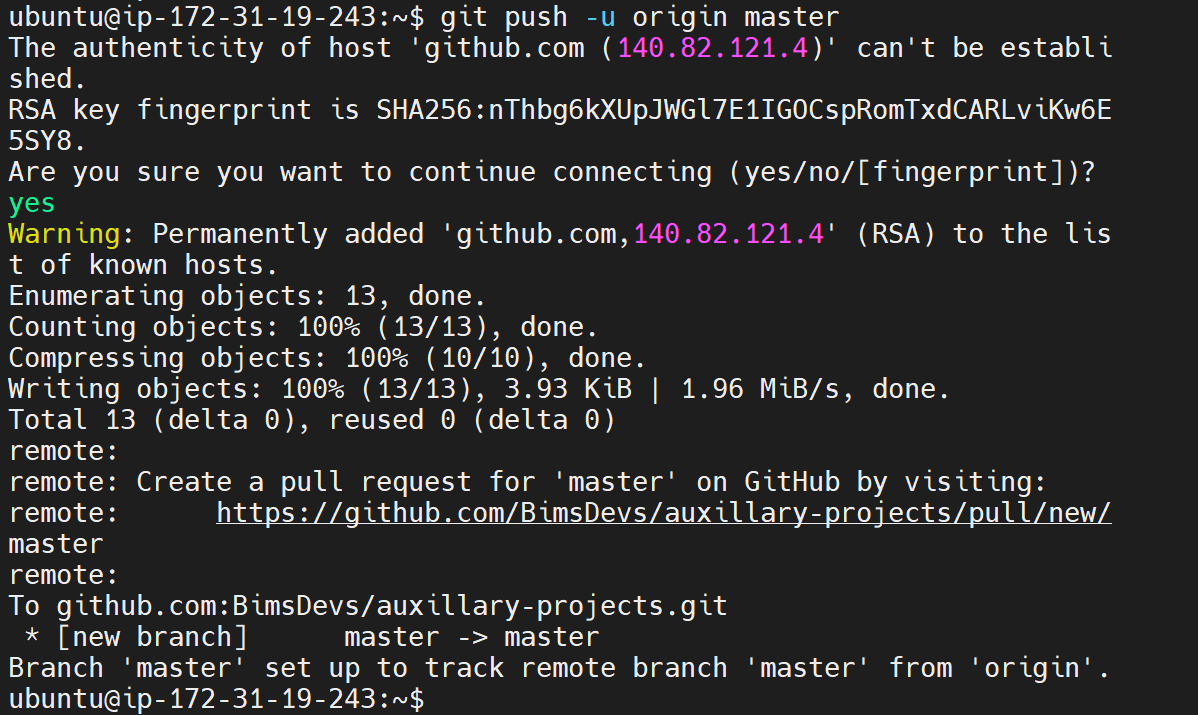
Connect local repository to the remote github

$ git remote add origin git@github.com:BimsDevs/auxillary-projects.git

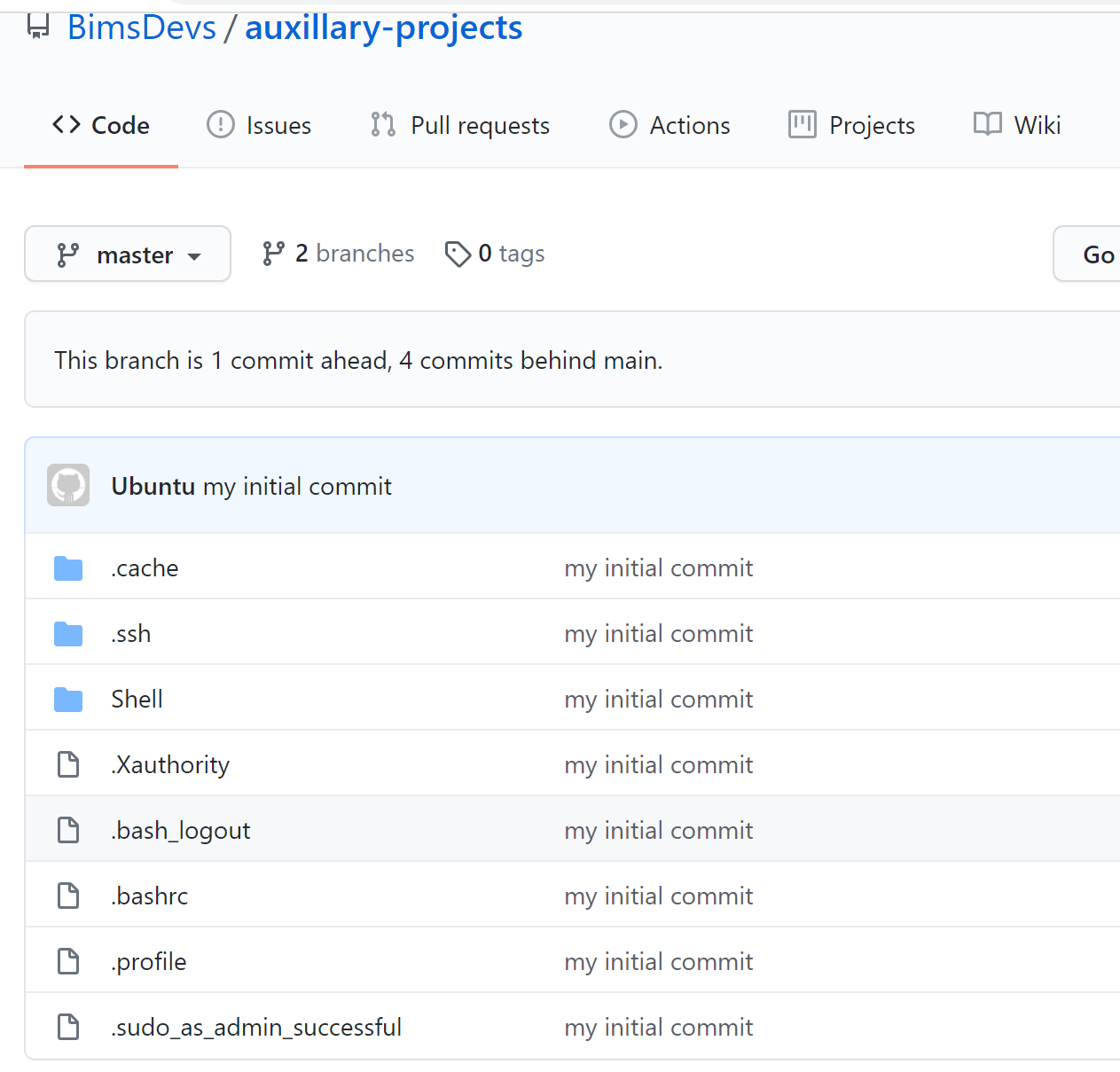
Check if connected: git remote show



Push to the remote repository: $ git push -u origin master



Check Github to confirm Push has been done successfully.



GitHub Links:

Gitbash: <https://github.com/BimsDevs/Aux-Documentation/tree/master>

Command line: <https://github.com/BimsDevs/auxillary-projects/tree/master>

Credit:

Shell Script: [*https://searchdatacenter.techtarget.com/definition/shell-script*](https://searchdatacenter.techtarget.com/definition/shell-script)

Kernel:[*https://www.redhat.com/en/topics/linux/what-is-the-linux-kernel*](https://www.redhat.com/en/topics/linux/what-is-the-linux-kernel)

[*https://www.cyberciti.biz/faq/unix-linux-bash-read-comma-separated-cvsfile/*](https://www.cyberciti.biz/faq/unix-linux-bash-read-comma-separated-cvsfile/)

[*http://www.beginninglinux.com/home/server-administration/openssh-keys-certificates-authentication-pem-pub-crt*](http://www.beginninglinux.com/home/server-administration/openssh-keys-certificates-authentication-pem-pub-crt)

[*https://www.digitalocean.com/community/tutorials/ssh-essentials-working-with-ssh-servers-clients-and-keys*](https://www.digitalocean.com/community/tutorials/ssh-essentials-working-with-ssh-servers-clients-and-keys)

Youtube Channels:

[*https://www.youtube.com/watch?v=ju9loeXNVW0*](https://www.youtube.com/watch?v=ju9loeXNVW0)

[*https://www.youtube.com/watch?v=CKBv9sn0Dz4*](https://www.youtube.com/watch?v=CKBv9sn0Dz4)

[*https://www.youtube.com/watch?v=SWYqp7iY\_Tc&feature=youtu.be*](https://www.youtube.com/watch?v=SWYqp7iY_Tc&feature=youtu.be)