L00170246 Question 7

Continous Assessment Conclusion

Table of Contents

About The Assessment

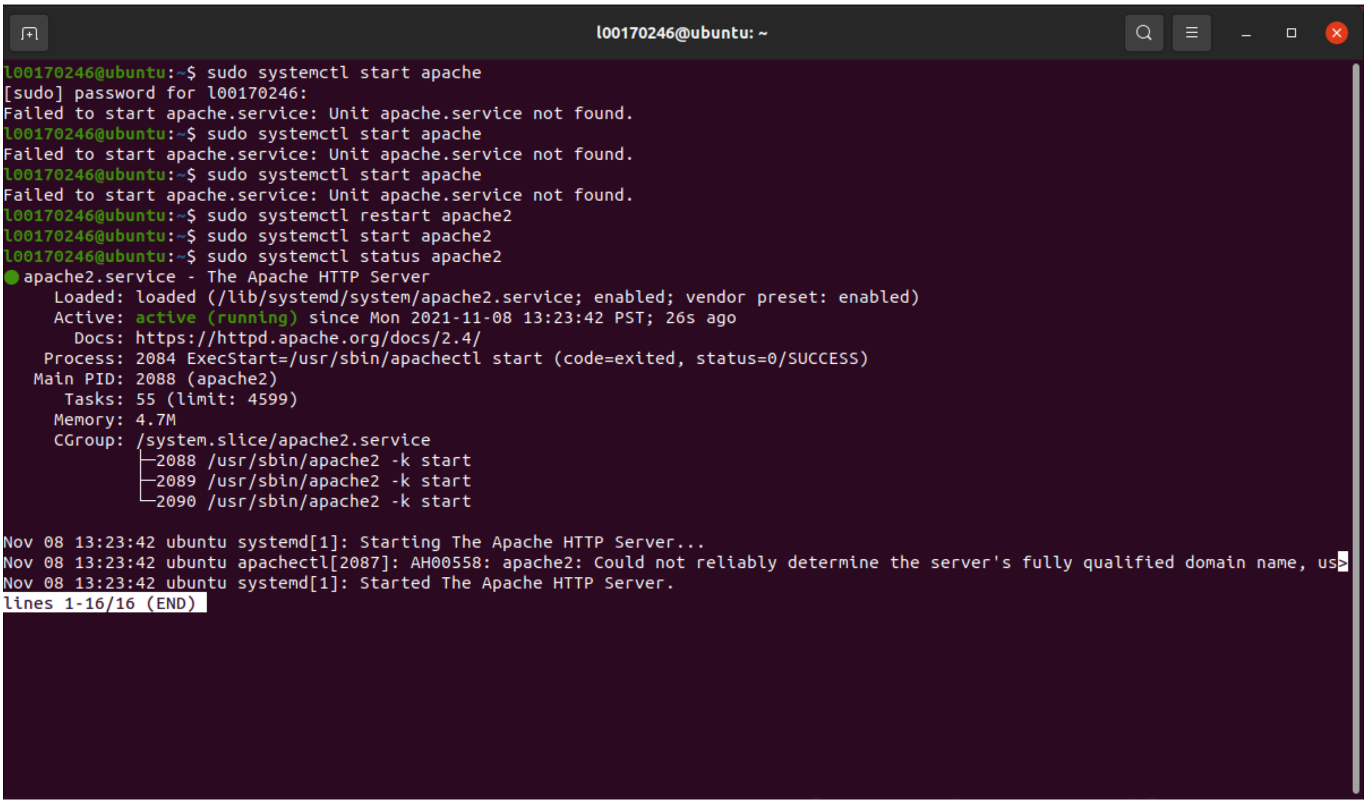
In this continuous assessment for Objected Orientated Programming for Server Administration, we had six tasks to carry out in our Programming Assignment. The details of all the files and documentation can be found on Github at OOPR-Assignment_DCM2021

(back to top)

## Task 01 - Labelled as L00170246_Q1

Task1 was a straightforward task, going through thr motions of installing Apache2 on the VM with Ubuntu 20.4 we were using for this assignment. I used my college account ID L00170246 to create account, and ran the command 'sudo apt-get install apache2' followed by 'sudo ufw allow 'Apache Full'.

By running 'sudo start apache2' and the 'sudo restart apache2' we can see the status of our website we have created by running 'sudo systemctl status apache' we can see the details of our webpage as per figure 1 below. `

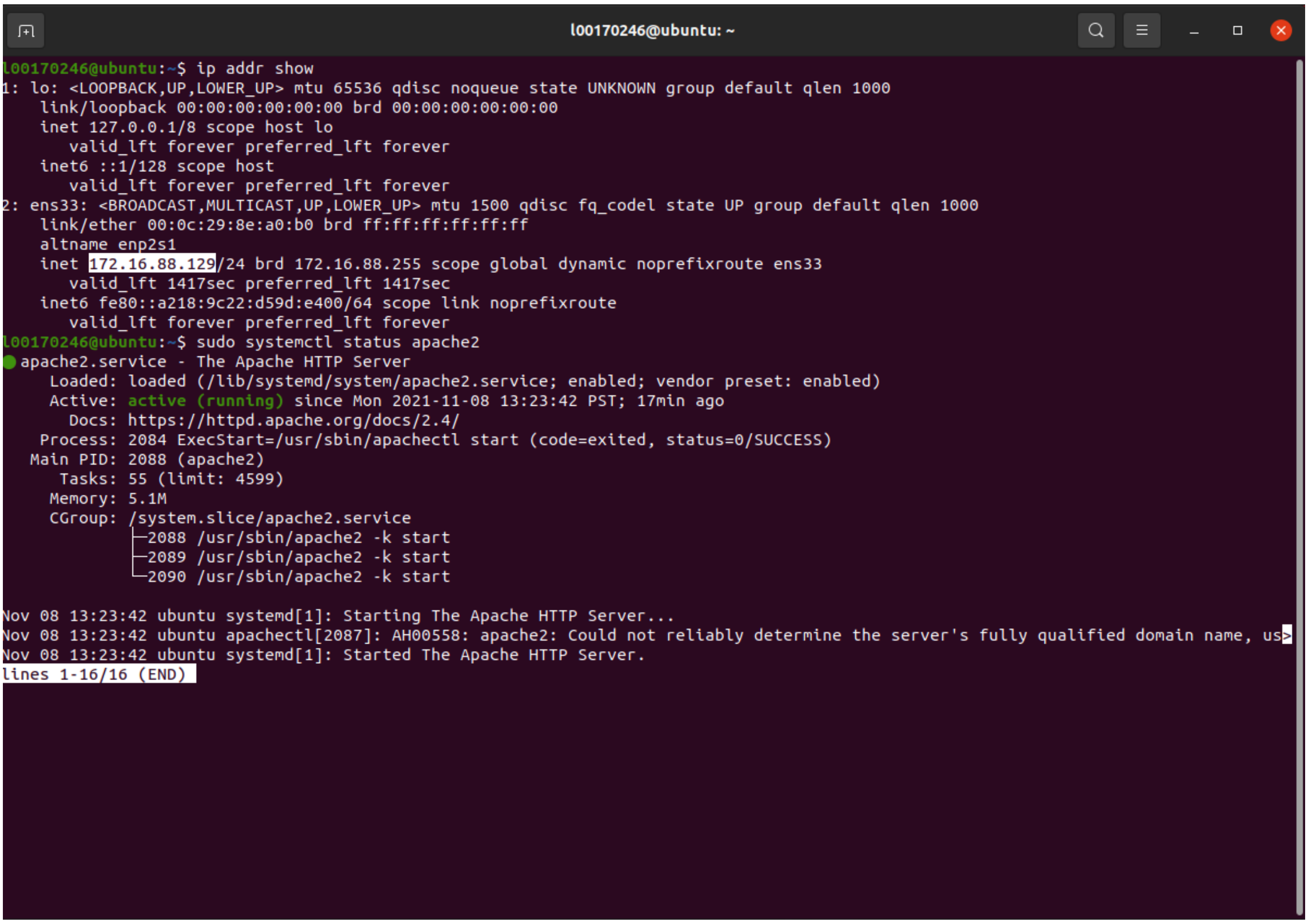


```
l00170246@ubuntu:~$ sudo systemctl start apache
[sudo] password for l00170246:
Failed to start apache.service: Unit apache.service not found.
l00170246@ubuntu:~$ sudo systemctl start apache
Failed to start apache.service: Unit apache.service not found.
l00170246@ubuntu:~$ sudo systemctl start apache
Failed to start apache.service: Unit apache.service not found.
l00170246@ubuntu:~$ sudo systemctl restart apache2
l00170246@ubuntu:~$ sudo systemctl start apache2
l00170246@ubuntu:~$ sudo systemctl status apache2
● apache2.service - The Apache HTTP Server
     Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor preset: enabled)
     Active: active (running) since Mon 2021-11-08 13:23:42 PST; 26s ago
       Docs: https://httpd.apache.org/docs/2.4/
    Process: 2084 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/SUCCESS)
   Main PID: 2088 (apache2)
      Tasks: 55 (limit: 4599)
     Memory: 4.7M
     CGroup: /system.slice/apache2.service
             ├─2088 /usr/sbin/apache2 -k start
             ├─2089 /usr/sbin/apache2 -k start
             └─2090 /usr/sbin/apache2 -k start

Nov 08 13:23:42 ubuntu systemd[1]: Starting The Apache HTTP Server...
Nov 08 13:23:42 ubuntu apachectl[2087]: AH00558: apache2: Could not reliably determine the server's fully qualified domain name, us>
Nov 08 13:23:42 ubuntu systemd[1]: Started The Apache HTTP Server.
lines 1-16/16 (END)
```

Figure 1

To see what IP address our webpage is running on we can run 'ip addr show' which will reveal the address which in this case is '172.16.88.129' as shown below in figure 2:



```
l00170246@ubuntu:~$ ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:0c:29:8e:a0:b0 brd ff:ff:ff:ff:ff:ff
    altname enp2s1
    inet 172.16.88.129/24 brd 172.16.88.255 scope global dynamic noprefixroute ens33
       valid_lft 1417sec preferred_lft 1417sec
    inet6 fe80::a218:9c22:d59d:e400/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
l00170246@ubuntu:~$ sudo systemctl status apache2
● apache2.service - The Apache HTTP Server
     Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor preset: enabled)
     Active: active (running) since Mon 2021-11-08 13:23:42 PST; 17min ago
       Docs: https://httpd.apache.org/docs/2.4/
    Process: 2084 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/SUCCESS)
   Main PID: 2088 (apache2)
      Tasks: 55 (limit: 4599)
     Memory: 5.1M
     CGroup: /system.slice/apache2.service
             ├─2088 /usr/sbin/apache2 -k start
             ├─2089 /usr/sbin/apache2 -k start
             └─2090 /usr/sbin/apache2 -k start

Nov 08 13:23:42 ubuntu systemd[1]: Starting The Apache HTTP Server...
Nov 08 13:23:42 ubuntu apachectl[2087]: AH00558: apache2: Could not reliably determine the server's fully qualified domain name, us>
Nov 08 13:23:42 ubuntu systemd[1]: Started The Apache HTTP Server.
lines 1-16/16 (END)
```

Figure 2

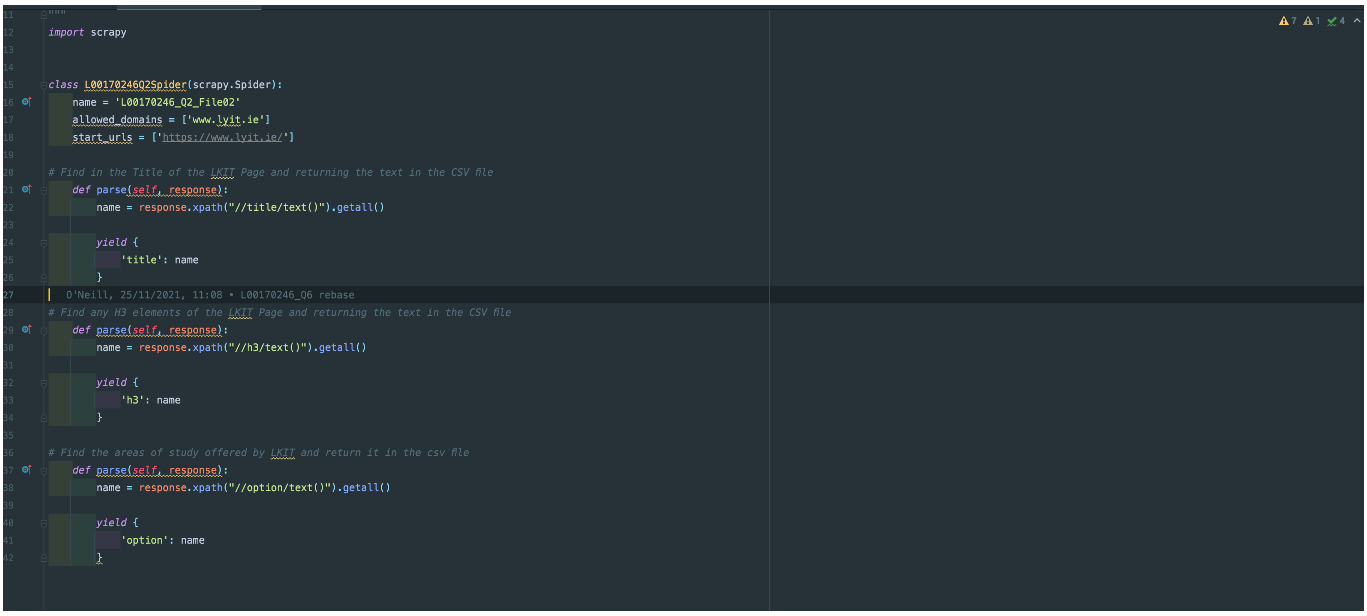
Task 02 - Labelled as L00170246_Q2

In the task 2, we were challenged with scraping the apache web page we created or the Letterkenny IT page. On this occasion I chose the LKIT website, Letterkenny I.T..

As I have used the Beautifulsoup module before, I decided to try out Scrapy for this task. The module was very easy to install using the command

```
pip install scrapy
```

Scrapy is an application framework for writing web spiders that crawl web sites and extract data from them. Using scrapy we generate spiders which can be setup and run to pull information. My file L00170246_Q2_File01.py, below in figure 3, shows an example which pulls "h3" elements from the website.



```
"""
import scrapy


class L00170246Q2Spider(scrapy.Spider):
    name = 'L00170246_Q2_File02'
    allowed_domains = ['www.lyit.ie']
    start_urls = ['https://www.lyit.ie/']

# Find in the Title of the LKIT Page and returning the text in the CSV file
    def parse(self, response):
        name = response.xpath("//title/text()").getall()

        yield {
            'title': name
        }
  O'Neill, 25/11/2021, 11:08 • L00170246_Q6 rebase
# Find any H3 elements of the LKIT Page and returning the text in the CSV file
    def parse(self, response):
        name = response.xpath("//h3/text()").getall()

        yield {
            'h3': name
        }

# Find the areas of study offered by LKIT and return it in the csv file
    def parse(self, response):
        name = response.xpath("//option/text()").getall()

        yield {
            'option': name
        }
```

Figure 3

Task 03 - Labelled as L00170246_Q3

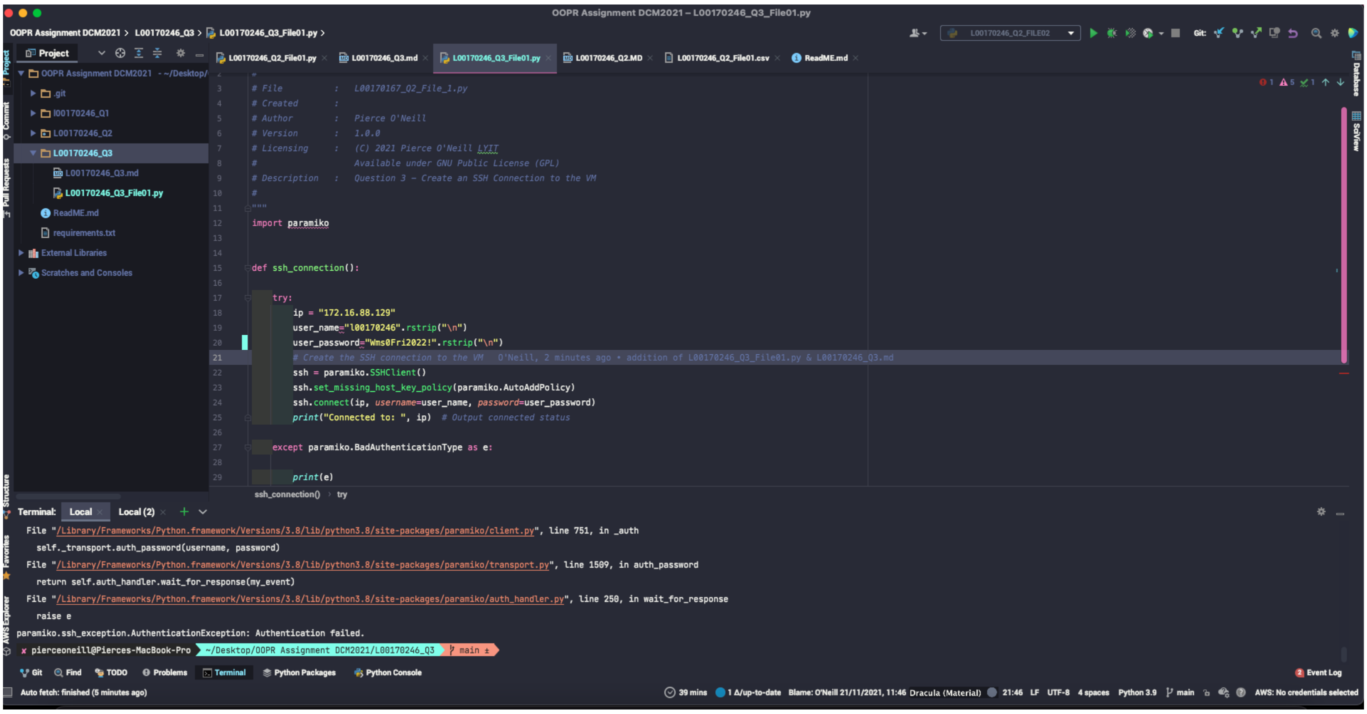
In Task 03, we have been asked to our virtual machine using a python script using the ssh port via modifying a previous script and verify that the connection was successful.

To test the connection to the VM from my host I used Iterm on my Mac OSX with the following command:

```
ssh l00170246@172.16.88.129
```

This includes my username "l00170246" & VM web hosting ip address "172.16.88.29"

I constructed the python script in Pycharm as per figure 4 below.



```python
#
# File          :   L00170167_Q2_File_1.py
# Created       :
# Author        :   Pierce O'Neill
# Version       :   1.0.0
# Licensing     :   (C) 2021 Pierce O'Neill LYIT
#                   Available under GNU Public License (GPL)
# Description   :   Question 3 - Create an SSH Connection to the VM
#
"""
import paramiko


def ssh_connection():

    try:
        ip = "172.16.88.129"
        user_name="l00170246".rstrip("\n")
        user_password="Wms0Fri2022!".rstrip("\n")
        # Create the SSH connection to the VM   O'Neill, 2 minutes ago • addition of L00170246_Q3_File01.py & L00170246_Q3.md
        ssh = paramiko.SSHClient()
        ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy)
        ssh.connect(ip, username=user_name, password=user_password)
        print("Connected to: ", ip)  # Output connected status

    except paramiko.BadAuthenticationType as e:

        print(e)
```

```
  File "/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/paramiko/client.py", line 751, in _auth
    self._transport.auth_password(username, password)
  File "/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/paramiko/transport.py", line 1509, in auth_password
    return self.auth_handler.wait_for_response(my_event)
  File "/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/paramiko/auth_handler.py", line 250, in wait_for_response
    raise e
paramiko.ssh_exception.AuthenticationException: Authentication failed.
✘ pierceoneill@Pierces-MacBook-Pro  ~/Desktop/OOPR Assignment DCM2021/L00170246_Q3  main ±
```

Figure 4

To prove that the connection will not work with the wrong security credentials, I updated the password and it produced the error

```
Authentication Failed
```

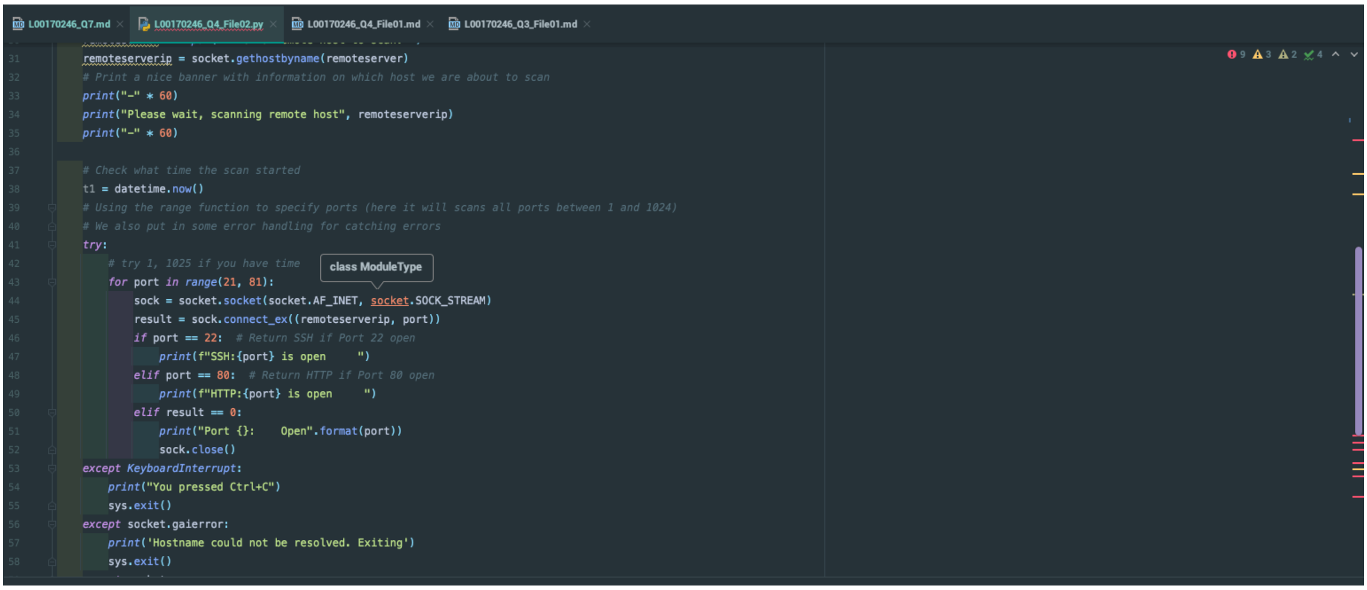
This task was interesting using python to form the script. I have enjoyed learning python and its light but powerful code.

(back to top)

Task 04 - Labelled as L00170246_Q4

Similar to the previous task, we had to create a python script but this time to determine which ports are open, in particular Port 22 & Port 80.

I constructed the python script in Pycharm as per figure 5, modifying the code from our lab with Ruth Lennon LYIT.

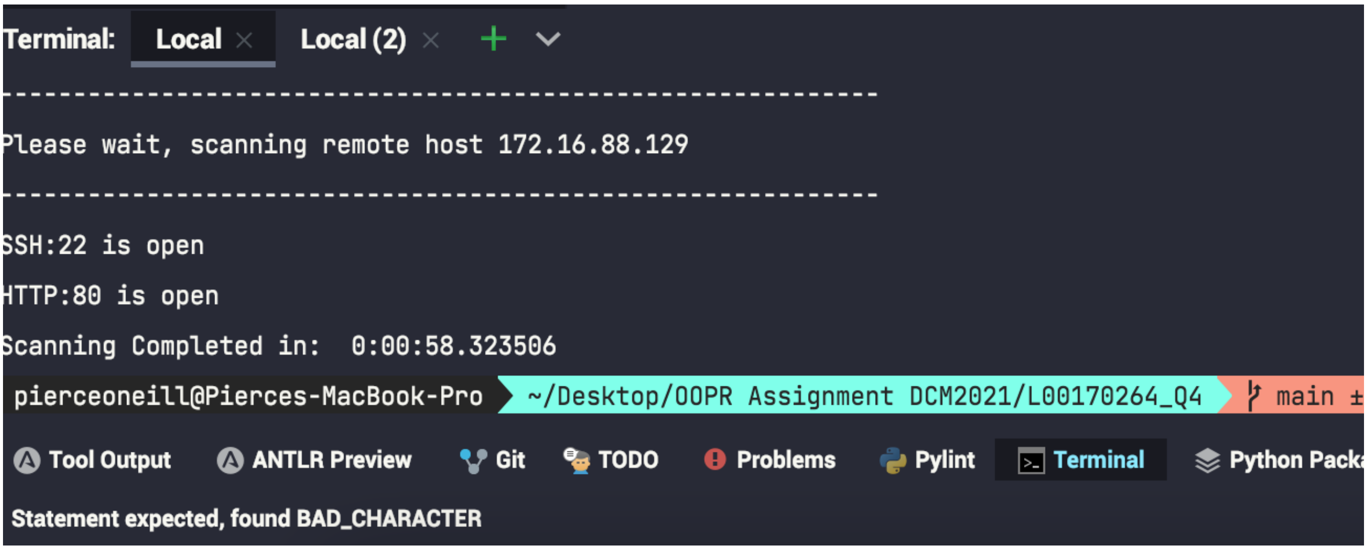


```python
remoteserverip = socket.gethostbyname(remoteserver)
# Print a nice banner with information on which host we are about to scan
print("-" * 60)
print("Please wait, scanning remote host", remoteserverip)
print("-" * 60)

# Check what time the scan started
t1 = datetime.now()
# Using the range function to specify ports (here it will scans all ports between 1 and 1024)
# We also put in some error handling for catching errors
try:
    # try 1, 1025 if you have time
    for port in range(21, 81):
        sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        result = sock.connect_ex((remoteserverip, port))
        if port == 22:  # Return SSH if Port 22 open
            print(f"SSH:{port} is open     ")
        elif port == 80:  # Return HTTP if Port 80 open
            print(f"HTTP:{port} is open     ")
        elif result == 0:
            print("Port {}:    Open".format(port))
            sock.close()
except KeyboardInterrupt:
    print("You pressed Ctrl+C")
    sys.exit()
except socket.gaierror:
    print('Hostname could not be resolved. Exiting')
    sys.exit()
```

Figure 5

The script checks the posts in range from 21 to 81 and checks if port 22 or port 80 are open using the 'elif' statement which is pythons short form for 'else if' . This means the script checks for open ports on 22 or 80 and returns a message to confirm the status as per figure 6.



```
------------------------------------------------------------
Please wait, scanning remote host 172.16.88.129
------------------------------------------------------------
SSH:22 is open
HTTP:80 is open
Scanning Completed in:  0:00:58.323506
pierceoneill@Pierces-MacBook-Pro  ~/Desktop/OOPR Assignment DCM2021/L00170264_Q4  main ±
```

Statement expected, found BAD_CHARACTER

Figure 6

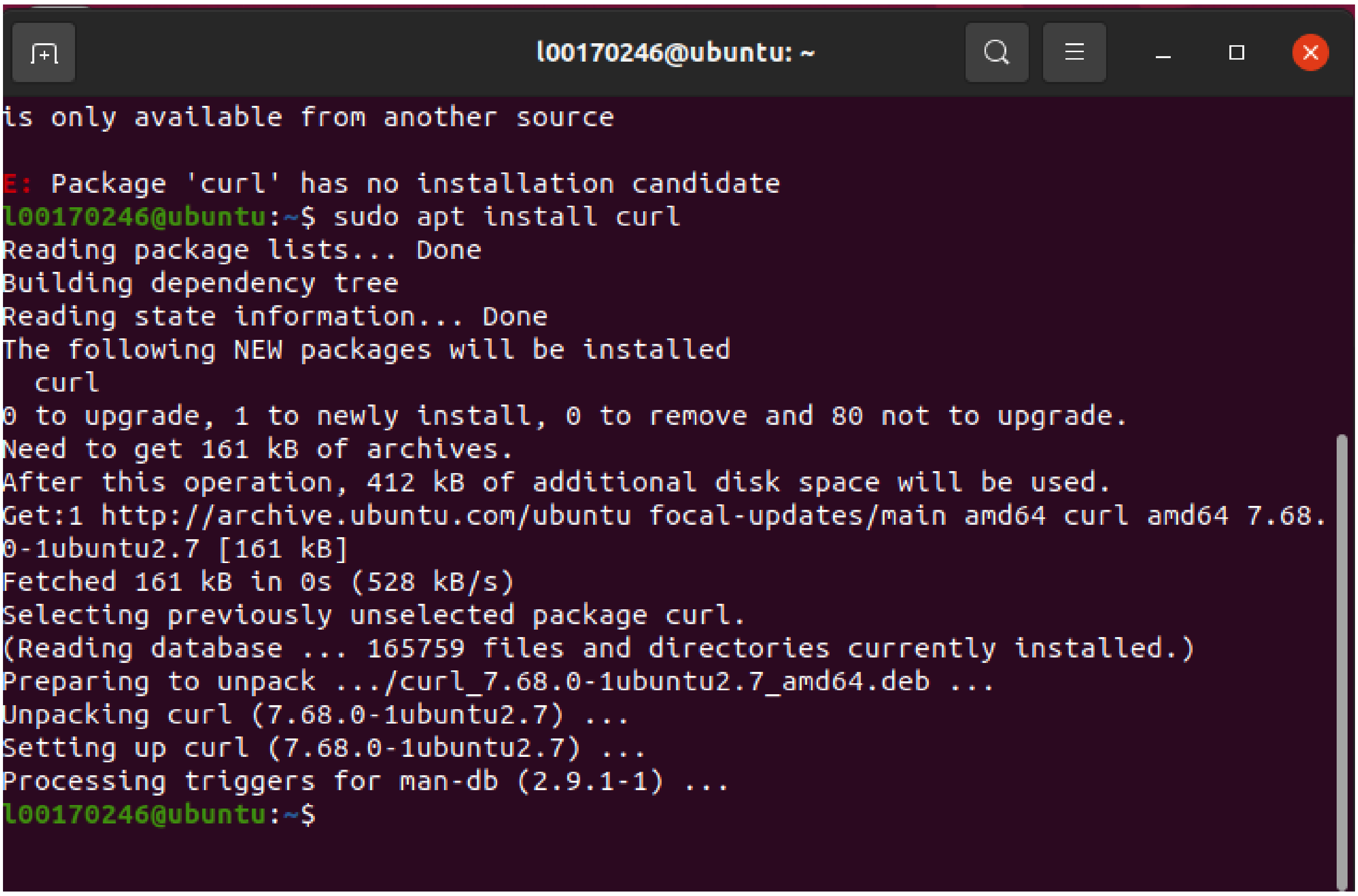
([back to top](#))

Task 05 - Labelled as L00170246_Q5

In task 5, code was provided which had to be modified in such a way as to (1) install curl, (2) create a directory structure & (3) find out when files were last accessed. To install curl, I reopened the VM and ran the command:

```
sudo apt install curl
```

curl installs successfully as per figure 1 below.

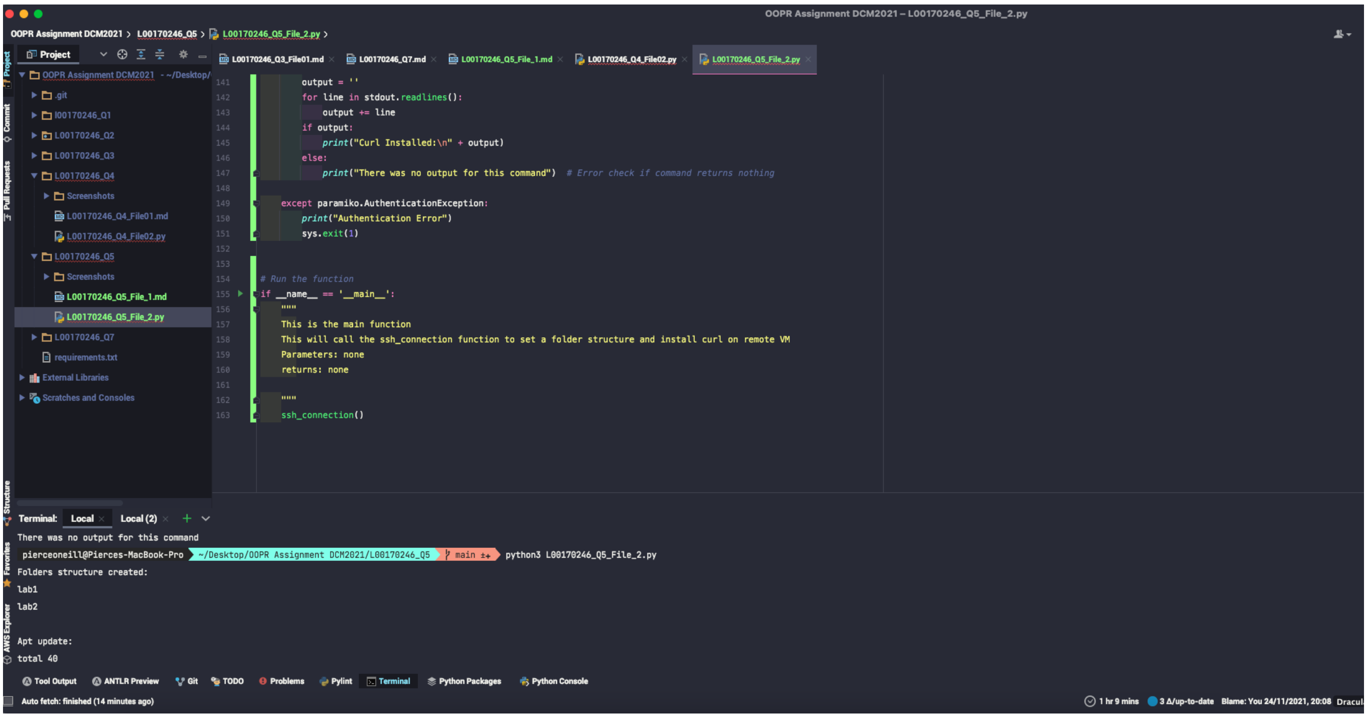


```
is only available from another source

E: Package 'curl' has no installation candidate
l00170246@ubuntu:~$ sudo apt install curl
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed
  curl
0 to upgrade, 1 to newly install, 0 to remove and 80 not to upgrade.
Need to get 161 kB of archives.
After this operation, 412 kB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 curl amd64 7.68.
0-1ubuntu2.7 [161 kB]
Fetched 161 kB in 0s (528 kB/s)
Selecting previously unselected package curl.
(Reading database ... 165759 files and directories currently installed.)
Preparing to unpack .../curl_7.68.0-1ubuntu2.7_amd64.deb ...
Unpacking curl (7.68.0-1ubuntu2.7) ...
Setting up curl (7.68.0-1ubuntu2.7) ...
Processing triggers for man-db (2.9.1-1) ...
l00170246@ubuntu:~$
```

Figure 7

We construct the python script in Pycharm as per figure 8 below.



```python
        output = ''
        for line in stdout.readlines():
            output += line
        if output:
            print("Curl Installed:\n" + output)
        else:
            print("There was no output for this command")  # Error check if command returns nothing

    except paramiko.AuthenticationException:
        print("Authentication Error")
        sys.exit(1)


# Run the function
if __name__ == '__main__':
    """
    This is the main function
    This will call the ssh_connection function to set a folder structure and install curl on remote VM
    Parameters: none
    returns: none

    """
    ssh_connection()
```

```
There was no output for this command
pierceoneill@Pierces-MacBook-Pro ~/Desktop/OOPR Assignment DCM2021/L00170246_Q5 main ±+ python3 L00170246_Q5_File_2.py
Folders structure created:
lab1
lab2

Apt update:
total 40
```

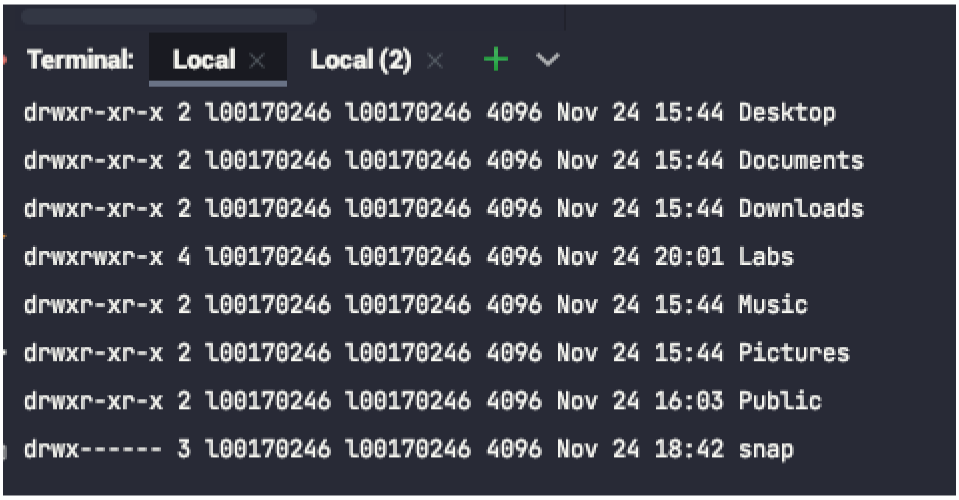
Figure 8

As can be seen from Figure 2, the script has created the Folder Labs with Lab1 & Lab2

By running the command

```
ls -l --time=atime
```

We can see a list of the directories and the last time they were accessed in figure 9 below:



```
drwxr-xr-x 2 l00170246 l00170246 4096 Nov 24 15:44 Desktop
drwxr-xr-x 2 l00170246 l00170246 4096 Nov 24 15:44 Documents
drwxr-xr-x 2 l00170246 l00170246 4096 Nov 24 15:44 Downloads
drwxrwxr-x 4 l00170246 l00170246 4096 Nov 24 20:01 Labs
drwxr-xr-x 2 l00170246 l00170246 4096 Nov 24 15:44 Music
drwxr-xr-x 2 l00170246 l00170246 4096 Nov 24 15:44 Pictures
drwxr-xr-x 2 l00170246 l00170246 4096 Nov 24 16:03 Public
drwx------ 3 l00170246 l00170246 4096 Nov 24 18:42 snap
```

Figure 9

(back to top)

Task 06 - Labelled as L00170246_Q6

In task 6, we were asked to construct a Terraform Script to create a sample infrastructure in Amazon Web Services. I used a virtual machine running Ubuntu 20.4 and the terminal to run this part of the assignment.

I validated my connection to Amazon Web Services (AWS) using the command:

```
aws configure
```

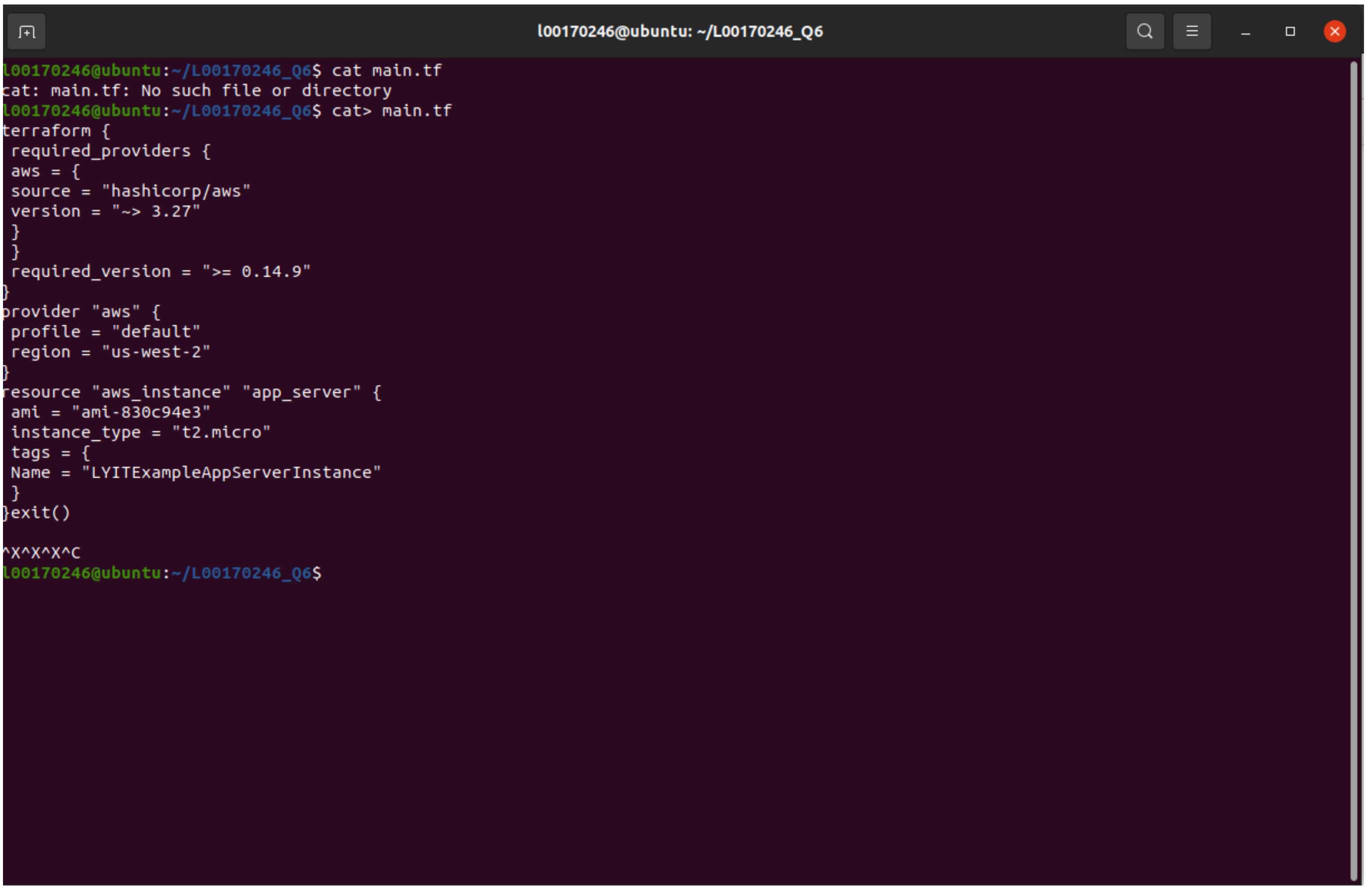
From this command I am prompted to add the following details:

```
AWS Access Key ID [****************PIXE]:
AWS Secret Access Key [****************lioS]:
Default region name [eu-west-1]:
Default output format [None]:
```

These details are available to the user from the Identity and Access Management module of the AWS account. (Warning: it is important to be careful with you Secret Access ID & Secret Access Key)

Inside my Folder L00170246_Q6, I create a file called 'main.tf'

Inside the main.tf file, we produce the following code as per figure 10 below. This code comes from the demo on tha Hashicorp Website.



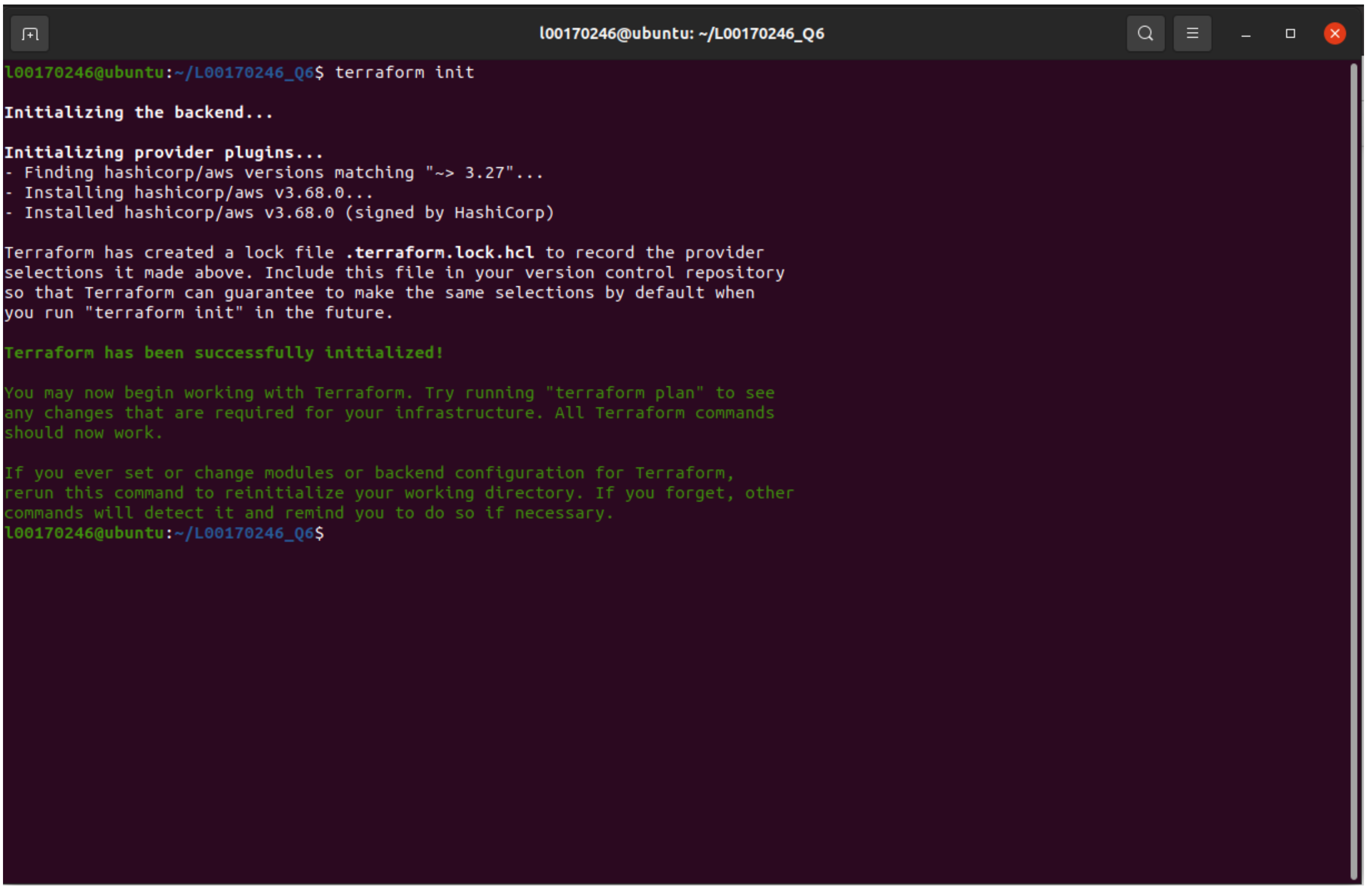
```
l00170246@ubuntu:~/L00170246_Q6$ cat main.tf
cat: main.tf: No such file or directory
l00170246@ubuntu:~/L00170246_Q6$ cat> main.tf
terraform {
 required_providers {
 aws = {
 source = "hashicorp/aws"
 version = "~> 3.27"
 }
 }
 required_version = ">= 0.14.9"
}
provider "aws" {
 profile = "default"
 region = "us-west-2"
}
resource "aws_instance" "app_server" {
 ami = "ami-830c94e3"
 instance_type = "t2.micro"
 tags = {
 Name = "LYITExampleAppServerInstance"
 }
}exit()

^X^X^X^C
l00170246@ubuntu:~/L00170246_Q6$
```

Figure 10

Now that the terraform configuration file is created run init to initialize terraform using the command:

```
terraform init
```



```
l00170246@ubuntu:~/L00170246_Q6$ terraform init

Initializing the backend...

Initializing provider plugins...
- Finding hashicorp/aws versions matching "~> 3.27"...
- Installing hashicorp/aws v3.68.0...
- Installed hashicorp/aws v3.68.0 (signed by HashiCorp)

Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
l00170246@ubuntu:~/L00170246_Q6$
```

Figure 11

As can be seen from Figure 11, Terraform has been successfully initialized! `

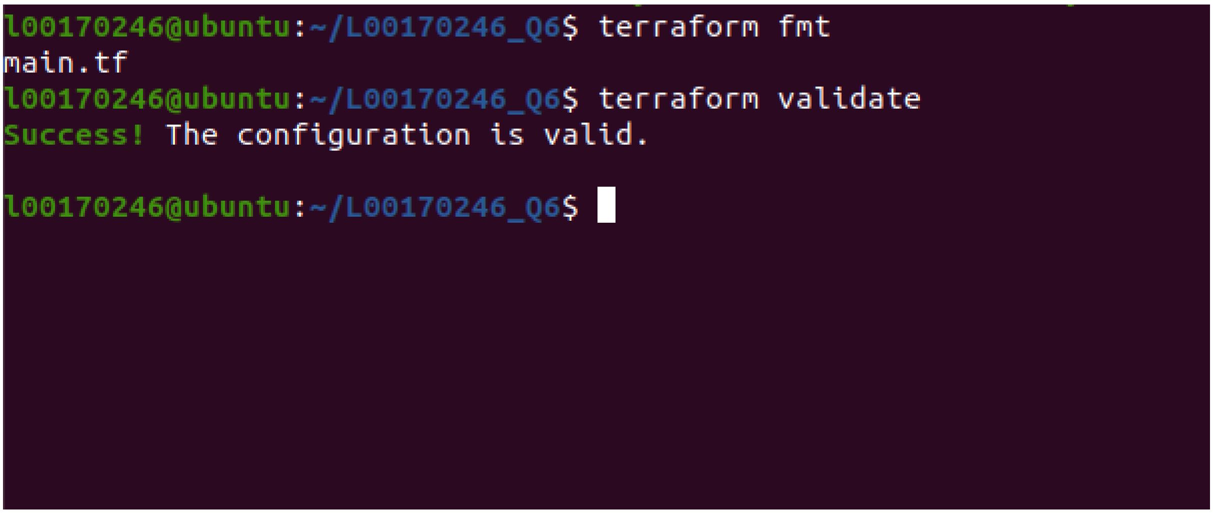
To run the file we need to run the command

```
terraform fmt
```

which should return 'main.tf' and also the command

```
terraform fmt
```

which should produce the success message as per figure 12 below

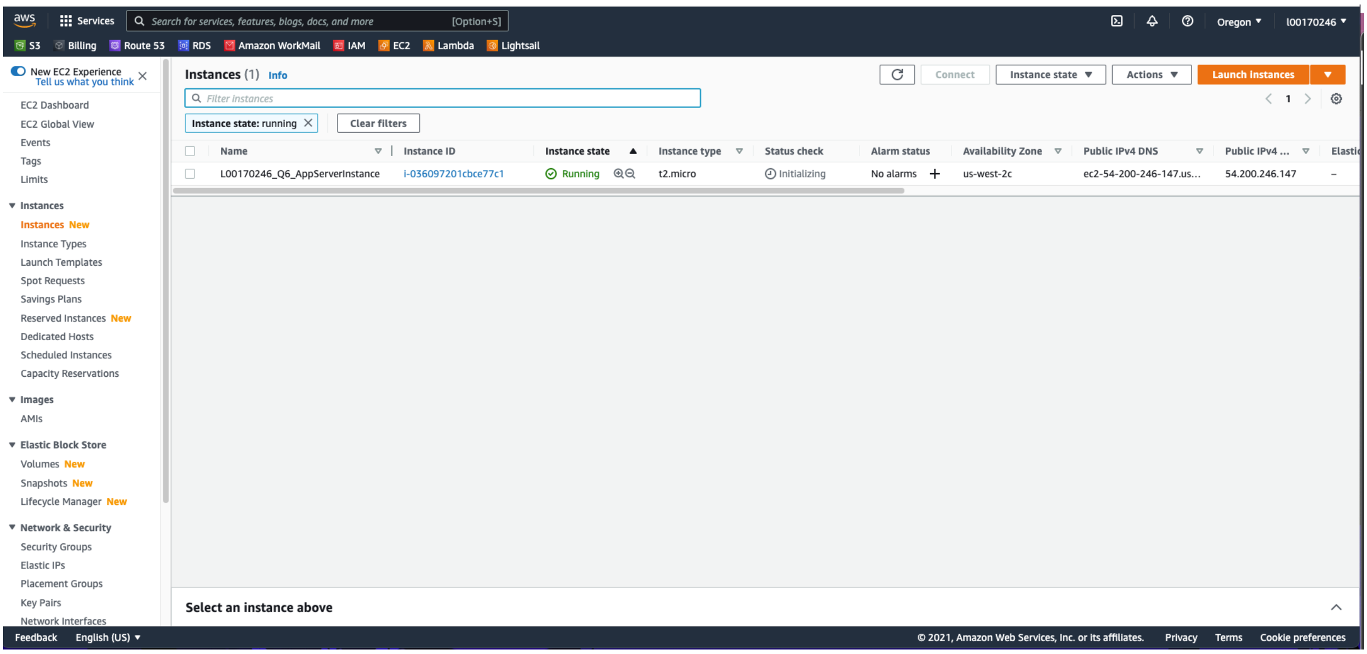


```
l00170246@ubuntu:~/L00170246_Q6$ terraform fmt
main.tf
l00170246@ubuntu:~/L00170246_Q6$ terraform validate
Success! The configuration is valid.

l00170246@ubuntu:~/L00170246_Q6$ 
```

Figure 12

When we run the commands above, we should be able to see our EC-2 instance created in AWS as per figure 13 below



Instances (1)

Instance state: running

	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elasti
□	L00170246_Q6_AppServerInstance	i-036097201cbce77c1	Running	t2.micro	Initializing	No alarms	us-west-2c	ec2-54-200-246-147.us...	54.200.246.147	–

Figure 13

To test the settings inside terraform, we are going to modify the settings inside the 'main.tf' file. We are going to change the version of Linux used by changing the ami from

```
ami = "ami-830c94e3"
```

to

```
ami = "ami-08d70e59c07c61a3a"
```

We must use the commands

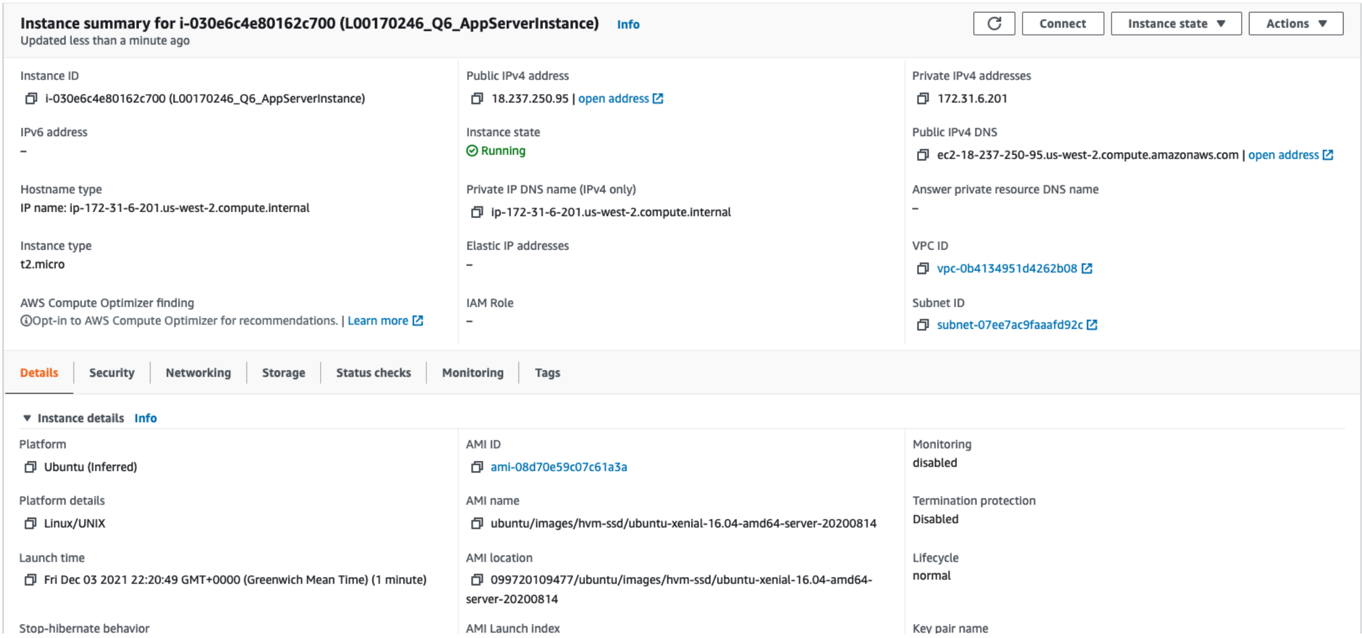
```
terraform validate
```

```
terraform plan
```

to ensure the code is still valid and to describe the tasks to be carried out. By running

```
terraform apply
```

we enact this plan and if refresh the AWS EC-2 dashboard, our 1st instance 'i-03609720cbce77c1' is shut down and terminates. Our new Instance 'i-030e6c4e80162c700' starts with our new ami id 'ami-08d70e59c07c61a3a'



Instance summary for i-030e6c4e80162c700 (L00170246_Q6_AppServerInstance)
Updated less than a minute ago

Instance ID
i-030e6c4e80162c700 (L00170246_Q6_AppServerInstance)

IPv6 address
–

Hostname type
IP name: ip-172-31-6-201.us-west-2.compute.internal

Instance type
t2.micro

AWS Compute Optimizer finding
Opt-in to AWS Compute Optimizer for recommendations. | Learn more

Public IPv4 address
18.237.250.95 | open address

Instance state
Running

Private IP DNS name (IPv4 only)
ip-172-31-6-201.us-west-2.compute.internal

Elastic IP addresses
–

IAM Role
–

Private IPv4 addresses
172.31.6.201

Public IPv4 DNS
ec2-18-237-250-95.us-west-2.compute.amazonaws.com | open address

Answer private resource DNS name
–

VPC ID
vpc-0b4134951d4262b08

Subnet ID
subnet-07ee7ac9faaafd92c

▼ Instance details

Platform
Ubuntu (Inferred)

Platform details
Linux/UNIX

Launch time
Fri Dec 03 2021 22:20:49 GMT+0000 (Greenwich Mean Time) (1 minute)

Stop-hibernate behavior

AMI ID
ami-08d70e59c07c61a3a

AMI name
ubuntu/images/hvm-ssd/ubuntu-xenial-16.04-amd64-server-20200814

AMI location
099720109477/ubuntu/images/hvm-ssd/ubuntu-xenial-16.04-amd64-server-20200814

AMI Launch index

Monitoring
disabled

Termination protection
Disabled

Lifecycle
normal

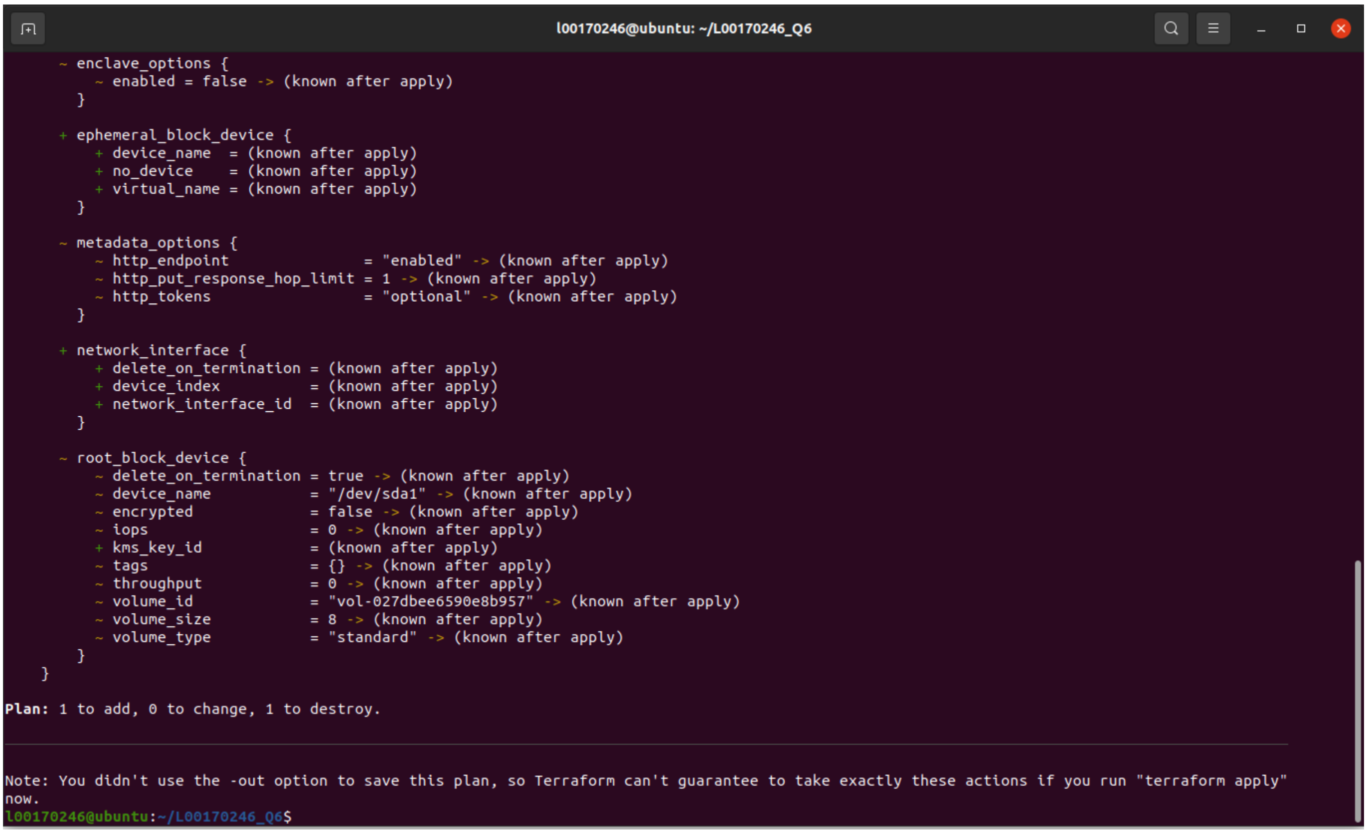
Key pair name

Figure 14

By running the command

```
terraform destroy
```

we will destroy all resources managed by terraform for a give project.



```
      ~ enclave_options {
          ~ enabled = false -> (known after apply)
        }

      + ephemeral_block_device {
          + device_name  = (known after apply)
          + no_device    = (known after apply)
          + virtual_name = (known after apply)
        }

      ~ metadata_options {
          ~ http_endpoint               = "enabled" -> (known after apply)
          ~ http_put_response_hop_limit = 1 -> (known after apply)
          ~ http_tokens                 = "optional" -> (known after apply)
        }

      + network_interface {
          + delete_on_termination = (known after apply)
          + device_index          = (known after apply)
          + network_interface_id  = (known after apply)
        }

      ~ root_block_device {
          ~ delete_on_termination = true -> (known after apply)
          ~ device_name           = "/dev/sda1" -> (known after apply)
          ~ encrypted             = false -> (known after apply)
          ~ iops                  = 0 -> (known after apply)
          + kms_key_id            = (known after apply)
          ~ tags                  = {} -> (known after apply)
          ~ throughput            = 0 -> (known after apply)
          ~ volume_id             = "vol-027dbee6590e8b957" -> (known after apply)
          ~ volume_size           = 8 -> (known after apply)
          ~ volume_type           = "standard" -> (known after apply)
        }
    }

Plan: 1 to add, 0 to change, 1 to destroy.

Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these actions if you run "terraform apply"
now.
l00170246@ubuntu:~/L00170246_Q6$
```

Figure 15

In this example we take the variable from another file as needed an populate the variables file for use by terraform. This way we can keep the file secret or we can modify the contents of the file as often as needed. The variables held in the terraform.tfvars therefore populate the variables.tf which then go into the main.tf file.

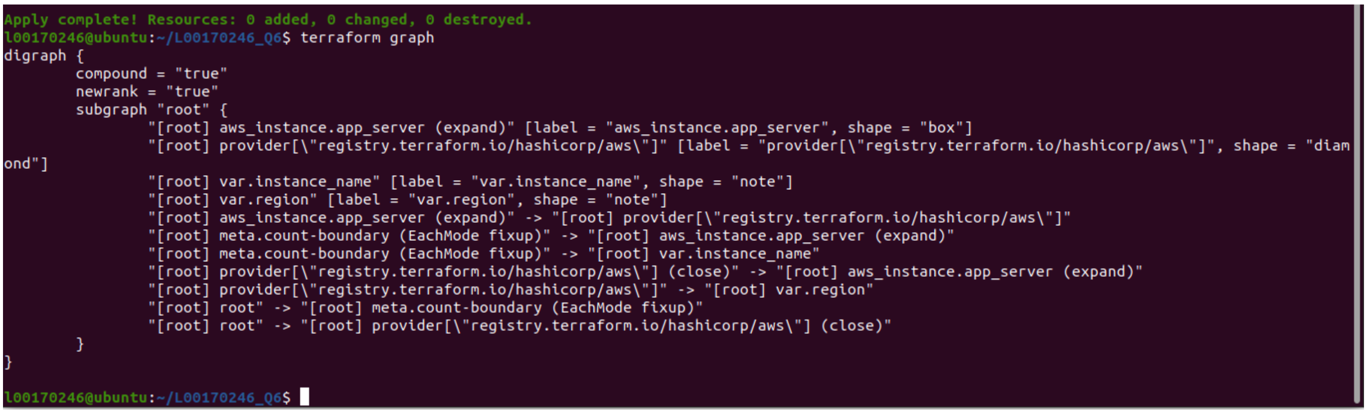
Variables.tf might show details such as the type and description even if the value is not provided.

```
variable “aws_secret_key” {
 type = string
 description = “Secret key for authorization”
}
```

We can run terraform graph to get a summarized, graphic version of the steps to be carried out.

```
terraform graph
```

This produces the below output, figure 16 from our Instance



```
Apply complete! Resources: 0 added, 0 changed, 0 destroyed.
l00170246@ubuntu:~/L00170246_Q6$ terraform graph
digraph {
        compound = "true"
        newrank = "true"
        subgraph "root" {
                "[root] aws_instance.app_server (expand)" [label = "aws_instance.app_server", shape = "box"]
                "[root] provider[\"registry.terraform.io/hashicorp/aws\"]" [label = "provider[\"registry.terraform.io/hashicorp/aws\"]", shape = "diam
ond"]
                "[root] var.instance_name" [label = "var.instance_name", shape = "note"]
                "[root] var.region" [label = "var.region", shape = "note"]
                "[root] aws_instance.app_server (expand)" -> "[root] provider[\"registry.terraform.io/hashicorp/aws\"]"
                "[root] meta.count-boundary (EachMode fixup)" -> "[root] aws_instance.app_server (expand)"
                "[root] meta.count-boundary (EachMode fixup)" -> "[root] var.instance_name"
                "[root] provider[\"registry.terraform.io/hashicorp/aws\"] (close)" -> "[root] aws_instance.app_server (expand)"
                "[root] provider[\"registry.terraform.io/hashicorp/aws\"]" -> "[root] var.region"
                "[root] root" -> "[root] meta.count-boundary (EachMode fixup)"
                "[root] root" -> "[root] provider[\"registry.terraform.io/hashicorp/aws\"] (close)"
        }
}

l00170246@ubuntu:~/L00170246_Q6$ 
```

Figure 16

The value of fields from a plugin can be stored in a similar way to the variables.tf file. In this example two values are stored. When we originally apply a change to our resource the following output is shown below. A simple ‘apply complete’ is indicated.

However, the aws provider has access to additional information. To access it we can create a file to convert the values for use by terraform.

We create a file to convert the values for storage called outputs.tf

```
output "instance_id" {
description = "ID of the EC2 instance"
value = aws_instance.app_server.id
}
output "instance_public_ip" {
description = "Public IP address of the EC2 instance"
value = aws_instance.app_server.public_ip
}
```

To tidy up and finish our lab we must use the command

```
    terraform destroy
```

Conclusion

I found this assessment challenging, but I also learned a lot by using python scripts to automate tasks. It was very satisfying to get the end results and to be able to unserstand how and where they came from.

The final task with terraform was new to me. I have experience with AWS but not this area. I feel that what I learned will help me with my career and studies. I will need to study Terraform more, however I found it a very "user-friendly" to understand. Task 6 probably took me the most amount of time to finish. Going through each task for the first time and learning was enjoyable.

I really enjoyed the coding side of this project being able to use short concise bits of code to produce an end result. I used Git and committed as best I could. I did have an issue with my computer after completing the first 5 tasks but resolved it. I enjoy finding solutions to problems and also like the backup of Git and Github.

(back to top)