PROJECT NAME

CONFIGURE FTP SERVER AND APACHE SERVER ON VIRTUAL MACHINE USING TERRAFORM

(TEAM 11)

CELEBEL PROJECT

7th floor Corporate Tower, JLN Marg, Jaipur-302017

DATE

08/05/20 – 14/05/2020

MENTOR NAME

(TUSHAR MITTAL)

TEAM LEADER NAME

(SHAILLY SHAH)

TEAM MEMBERS NAME

1. SHAILLY SHAH
2. RIYA GUPTA
3. SALONI KARAMWANI
4. RAGHUVEER SINGH
5. RISHABH KALYANI

TABLE OF CONTENTS :-

PAGE NO

1.OBJECTIVE :- CONFIGURE FTP SERVER AND APACHE SERVER ON VIRTUAL MACHINE USING TERRAFORM.

2. WHAT IS TERRAFORM ? & WHY WE USE TERRAFORM :-

Terraform is a tool for building, changing, and versioning infrastructure safely and efficiently. Terraform can manage existing and popular service providers as well as custom in-house solutions. Configuration files describe to Terraform the components needed to run a single application or your entire datacenter.

Terraform generates an execution plan describing what it will do to reach the desired state, and then executes it to build the described infrastructure.

As the configuration changes, Terraform is able to determine what changed and create incremental execution plans which can be applied.

The infrastructure Terraform can manage includes low-level components such as compute instances, storage, and networking, as well as high-level components such as DNS entries, SaaS features, etc.

3. Problems in Provisioning but with the help of terraform we can resolve it :-

There are two major problems everyone faces when trying to improve their provisioning practices :- Technical complexity and Organizational complexity.

1. ***Technical complexity*** *— Different infrastructure providers use different interfaces to provision new resources, and the inconsistency between these interfaces imposes extra costs on daily operations. These costs get worse as you add more infrastructure providers and more collaborators.*

   Terraform addresses this complexity by separating the provisioning workload. It uses a single core engine to read infrastructure as code configurations and determine the relationships between resources, then uses many provider plugins to create, modify, and destroy resources on the infrastructure providers.

   In other words, Terraform uses a model of workflow-level abstraction, rather than resource-level abstraction. It lets you use a single workflow for managing infrastructure, but acknowledges the uniqueness of each provider instead of imposing generic concepts on non-equivalent resources.

2. ***Organizational complexity*** *— As infrastructure scales, it requires more teams to maintain it. For effective collaboration, it's important to delegate ownership of infrastructure across these teams and empower them to work in parallel without conflict. Terraform and Terraform Cloud can help delegate infrastructure in the same way components of a large application are delegated.*

> *To delegate a large application, companies often split it into small, focused microservice components that are owned by specific teams. Each microservice provides an API, and as long as those APIs don't change, microservice teams can make changes in parallel despite relying on each others' functionality.*
>
> *This is how Terraform Cloud solves the organizational complexity of provisioning: by providing a centralized run environment for Terraform that supports and enforces your organization's access control decisions across all workspaces. This helps you delegate infrastructure ownership to enable parallel development.*

So here is the main reason why we use terraform because it solves both the problems we face.

NOW, let's continue to our installation process of TERRAFORM.

4.Prerequisites for the installation :-

Azure subscription: *If you don't have an Azure subscription, create a free account before you begin.*

Install Terraform

By default, the latest version of Terraform is installed for use in the Azure Cloud Shell. If you choose to install Terraform locally, complete this step; otherwise, continue to Configure Terraform access to Azure.

1. Install Terraform specifying the appropriate package for your operating system.

2. The download contains a single executable file. Define a global path to the executable based on your operating system:

Linux or MacOS , Windows.

1.PROCEDURE FOR TERRAFORM INSTALLATION :

Installing and configuring terraform in windows

1. Install terraform from hashicorp website to the system and extract the file to the new directory
2. After extracting the file set the environment variable of the terraform
   this pc -> properties-> advanced-> environment variables-> set system variables-> select path-> edit-> add the path of terraform directory-> ok
3. now terraform is installed and configured to run on the local system

Installing azure cli

1. Install azure cli to interact with azure though local system using cmd
2. open poweshell as administrator
3. type Invoke-WebRequest -Uri https://aka.ms/installazurecliwindows -OutFile .\AzureCLI.msi; Start-Process msiexec.exe -Wait -ArgumentList '/I AzureCLI.msi /quiet'; rm .\AzureCLI.msi
4. run the command and wait for azure cli to install

Authenticating terraform using azure cli

1. Open cmd from the terraform project directory
2. run az login-> a browser window open to login into azure account
3. login to azure account and close the window
4. detail of the azure account will be displayed on the cmd
5. set the subscription id for the project which to be used with terraform

```
az account set --subscription="${SUBSCRIPTION_ID}"
```

Authenticating using service principal to make role-based provisions

1. open cmd and run the following command

```
az ad sp create-for-rbac --role="Contributor"--scopes="/subscriptions/${SUBSCRIPTION_ID}"
```

2. detailed information regarding tenant id, password will be displayed on cmd

Writing IaC

1. open any ide and name it main.tf
This main.tf file contain all the main configuration of the project
2. write the following code

```
provider "azurerm" {
subscription_id = "<your_subscription_id>"
  version       = "~>2.0"
  features{}
```

This is the basic config for the terraform which provide the name of the provider and your subscription id to authenticate while run time.

3. To make the terraform modules dynamic use variables which can be pas through run time in console

To make these changes create a new file name variable.tf

Write the following code in variable.tf

```
        Variable “subscriptionId”{....
        Type=string
        }
```

4. To pass this variable during run time create a new file named terraform.tfvars

Write the following code

```
        subscriptionId=”<your_subscription_id>”
```

5. Now change the main.tf file to pass this variable

```
                provider "azurerm" {
        subscription_id =var.subscriptionId
         version       = "~>2.0"
         features{}
        }
```

Running the code

1. open the cmd from the terraform project location
2. run terraform init

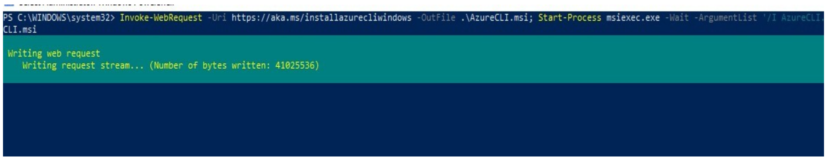
After successful initialization of terraform plan the project

3. run terraform plan

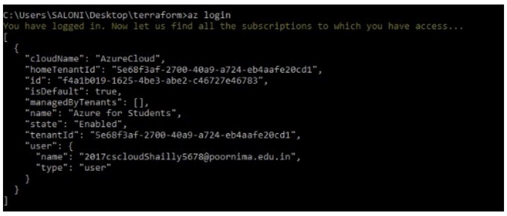
All the created resources and other things should display on the console. Type yes to execute the plan

3. run terraform apply.

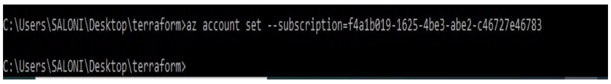
SCREENSHOT OF TERRAFORM INSTALLATION



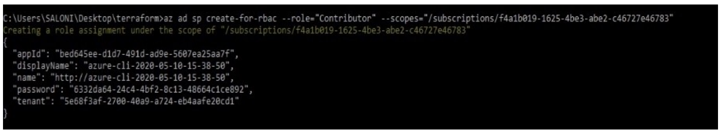
```
PS C:\WINDOWS\system32> Invoke-WebRequest -Uri https://aka.ms/installazurecliwindows -OutFile .\AzureCLI.msi; Start-Process msiexec.exe -Wait -ArgumentList '/I AzureCLI.
CLI.msi

 Writing web request
    Writing request stream... (Number of bytes written: 41025536)
```



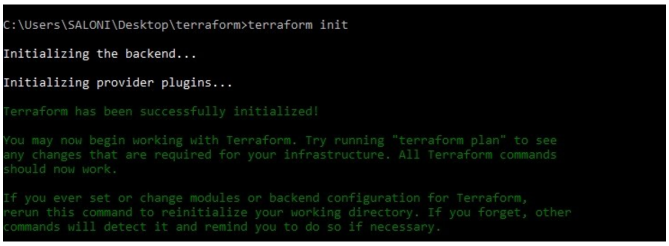
```
C:\Users\SALONI\Desktop\terraform>az login
You have logged in. Now let us find all the subscriptions to which you have access...
[
  {
    "cloudName": "AzureCloud",
    "homeTenantId": "5e68f3af-2700-40a9-a724-eb4aafe20cd1",
    "id": "f4a1b019-1625-4be3-abe2-c46727e46783",
    "isDefault": true,
    "managedByTenants": [],
    "name": "Azure for Students",
    "state": "Enabled",
    "tenantId": "5e68f3af-2700-40a9-a724-eb4aafe20cd1",
    "user": {
      "name": "2017cscloudShailly5678@poornima.edu.in",
      "type": "user"
    }
  }
]
```



```
C:\Users\SALONI\Desktop\terraform>az account set --subscription=f4a1b019-1625-4be3-abe2-c46727e46783

C:\Users\SALONI\Desktop\terraform>
```



```
C:\Users\SALONI\Desktop\terraform>az ad sp create-for-rbac --role="Contributor" --scopes="/subscriptions/f4a1b019-1625-4be3-abe2-c46727e46783"
Creating a role assignment under the scope of "/subscriptions/f4a1b019-1625-4be3-abe2-c46727e46783"
{
  "appId": "bed645ee-d1d7-491d-ad9e-5607ea25aa7f",
  "displayName": "azure-cli-2020-05-10-15-38-50",
  "name": "http://azure-cli-2020-05-10-15-38-50",
  "password": "6332da64-24c4-4bf2-8c13-48664c1ce892",
  "tenant": "5e68f3af-2700-40a9-a724-eb4aafe20cd1"
}
```

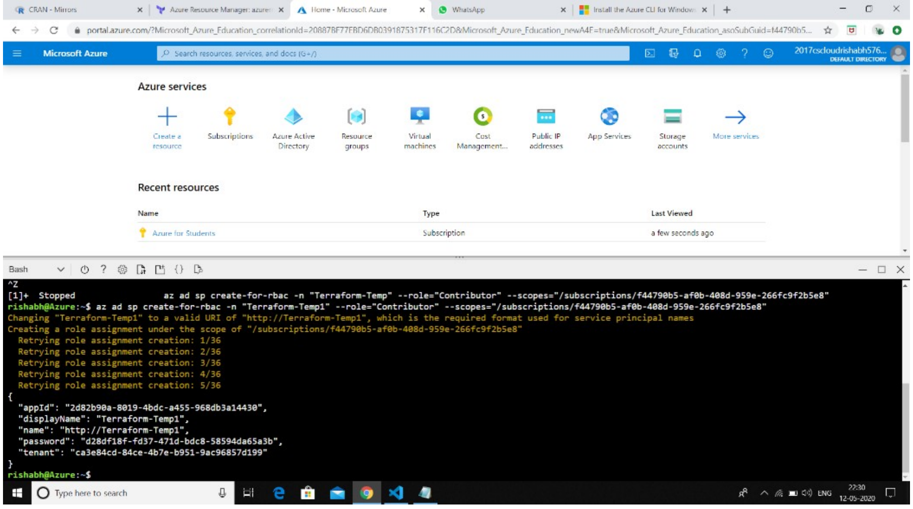


```
C:\Users\SALONI\Desktop\terraform>terraform init

Initializing the backend...

Initializing provider plugins...

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.
```

Configure Terraform environment variables

To configure Terraform to use your Azure AD service principal, set the following environment variables, which are then used by the Azure Terraform modules. You can also set the environment if working with an Azure cloud other than Azure public.

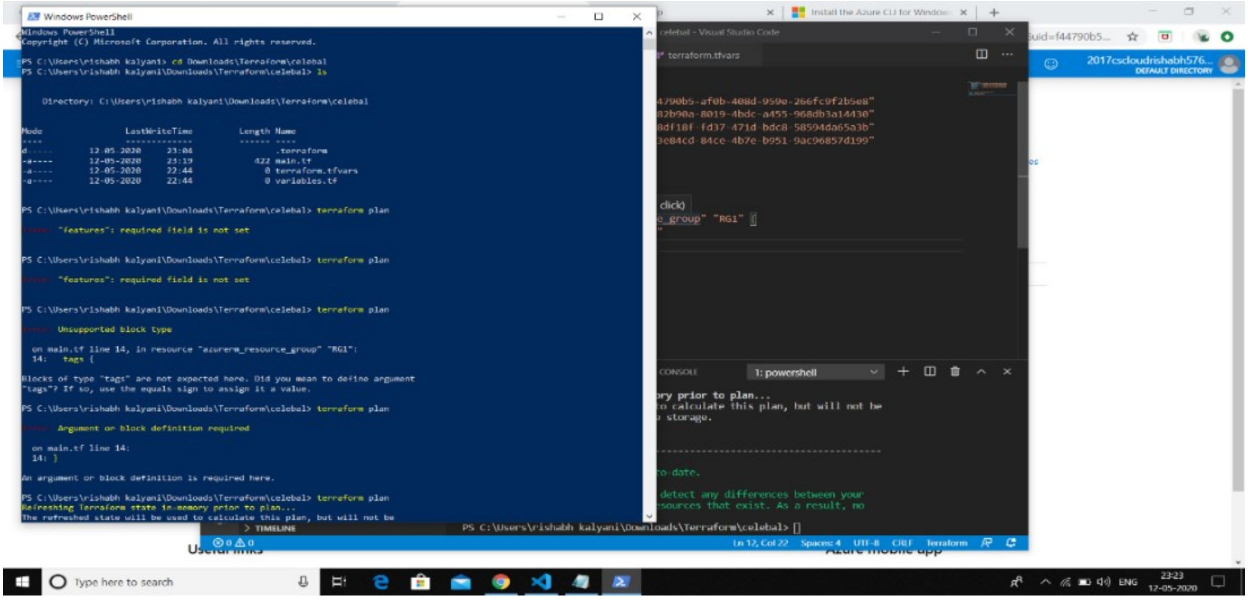
- *USER_SUBSCRIPTION_ID*
- *USER_CLIENT_ID*
- *USER_CLIENT_SECRET*
- *USER_TENANT_ID*
- *USER_ENVIRONMENT*



```
^Z
[1]+  Stopped                 az ad sp create-for-rbac -n "Terraform-Temp" --role="Contributor" --scopes="/subscriptions/f44790b5-af0b-408d-959e-266fc9f2b5e8"
rishabh@Azure:~$ az ad sp create-for-rbac -n "Terraform-Temp1" --role="Contributor" --scopes="/subscriptions/f44790b5-af0b-408d-959e-266fc9f2b5e8"
Changing "Terraform-Temp1" to a valid URI of "http://Terraform-Temp1", which is the required format used for service principal names
Creating a role assignment under the scope of "/subscriptions/f44790b5-af0b-408d-959e-266fc9f2b5e8"
  Retrying role assignment creation: 1/36
  Retrying role assignment creation: 2/36
  Retrying role assignment creation: 3/36
  Retrying role assignment creation: 4/36
  Retrying role assignment creation: 5/36
{
  "appId": "2d82b90a-8019-4bdc-a455-968db3a14430",
  "displayName": "Terraform-Temp1",
  "name": "http://Terraform-Temp1",
  "password": "d28df18f-fd37-471d-bdc8-58594da65a3b",
  "tenant": "ca3e84cd-84ce-4b7e-b951-9ac96857d199"
}
rishabh@Azure:~$
```

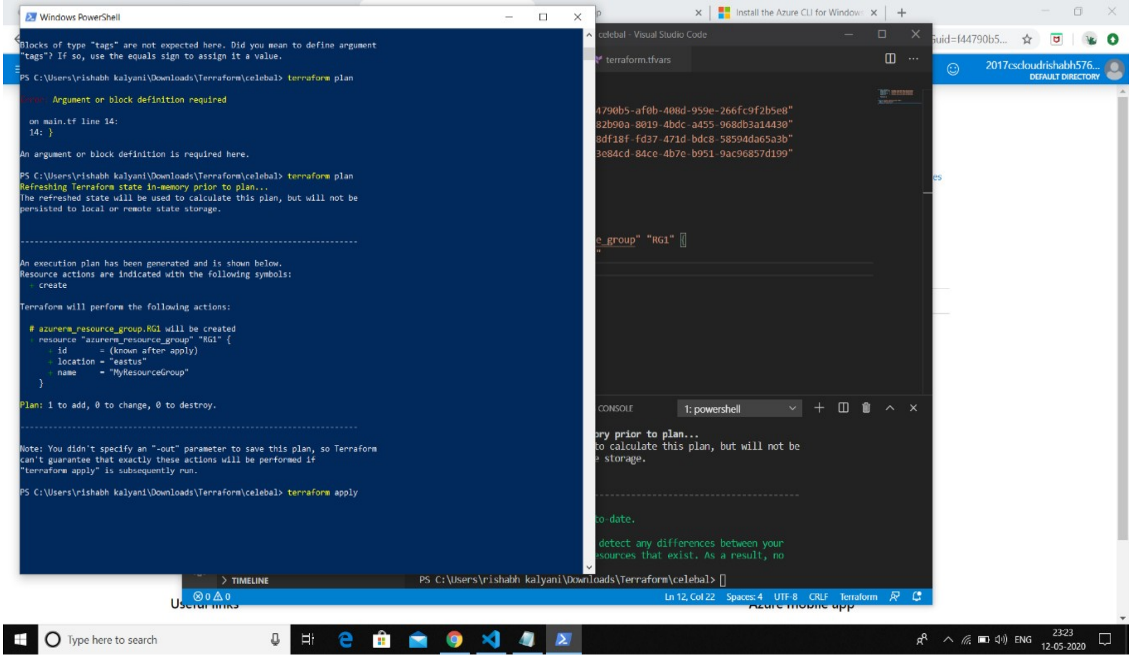
2.CREATION OF VM USING TERRAFORM :-

NOW , You can preview the actions to be completed by the Terraform script with terraform plan. When ready to create the resource group, apply your Terraform plan as follows:



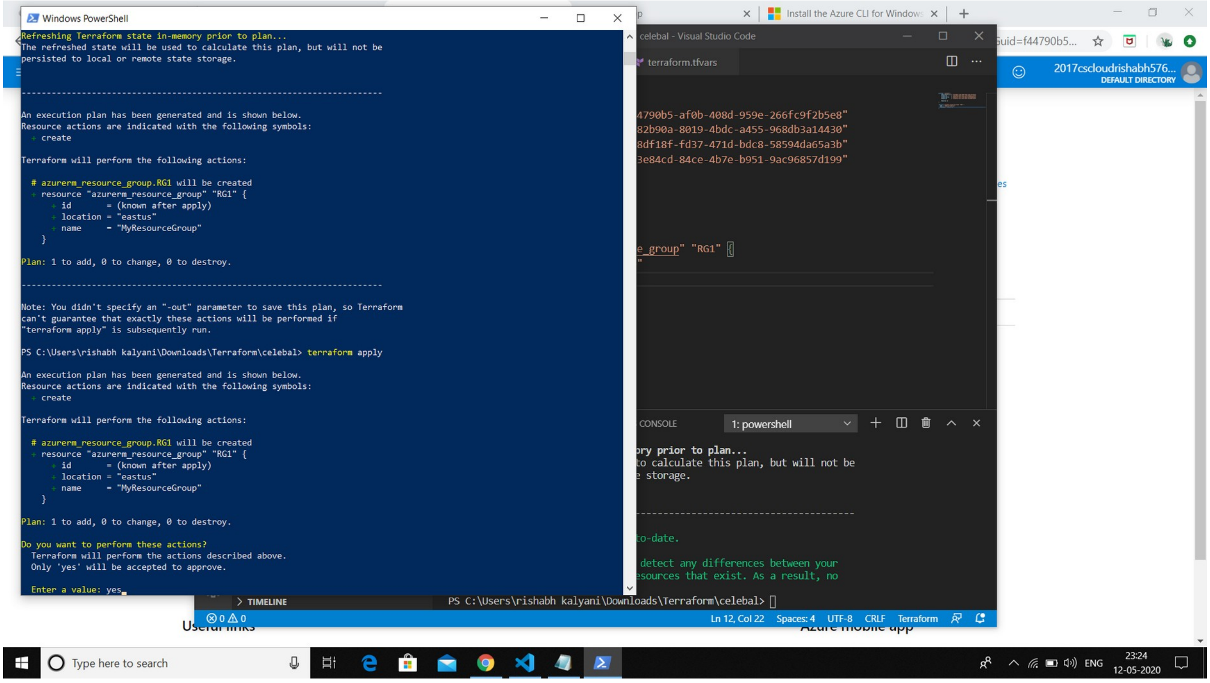
```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Users\rishabh kalyani> cd Downloads\Terraform\celebal
PS C:\Users\rishabh kalyani\Downloads\Terraform\celebal> ls


    Directory: C:\Users\rishabh kalyani\Downloads\Terraform\celebal


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
d-----       12-05-2020     23:04                .terraform
-a----       12-05-2020     23:19            422 main.tf
-a----       12-05-2020     22:44              0 terraform.tfvars
-a----       12-05-2020     22:44              0 variables.tf


PS C:\Users\rishabh kalyani\Downloads\Terraform\celebal> terraform plan

Error: "features": required field is not set


PS C:\Users\rishabh kalyani\Downloads\Terraform\celebal> terraform plan

Error: "features": required field is not set


PS C:\Users\rishabh kalyani\Downloads\Terraform\celebal> terraform plan

Error: Unsupported block type

  on main.tf line 14, in resource "azurerm_resource_group" "RG1":
  14:   tags {

Blocks of type "tags" are not expected here. Did you mean to define argument
"tags"? If so, use the equals sign to assign it a value.

PS C:\Users\rishabh kalyani\Downloads\Terraform\celebal> terraform plan

Error: Argument or block definition required

  on main.tf line 14:
  14: }

An argument or block definition is required here.

PS C:\Users\rishabh kalyani\Downloads\Terraform\celebal> terraform plan
Refreshing Terraform state in-memory prior to plan...
The refreshed state will be used to calculate this plan, but will not be
```

1. *Terraform is performing the actions :-*



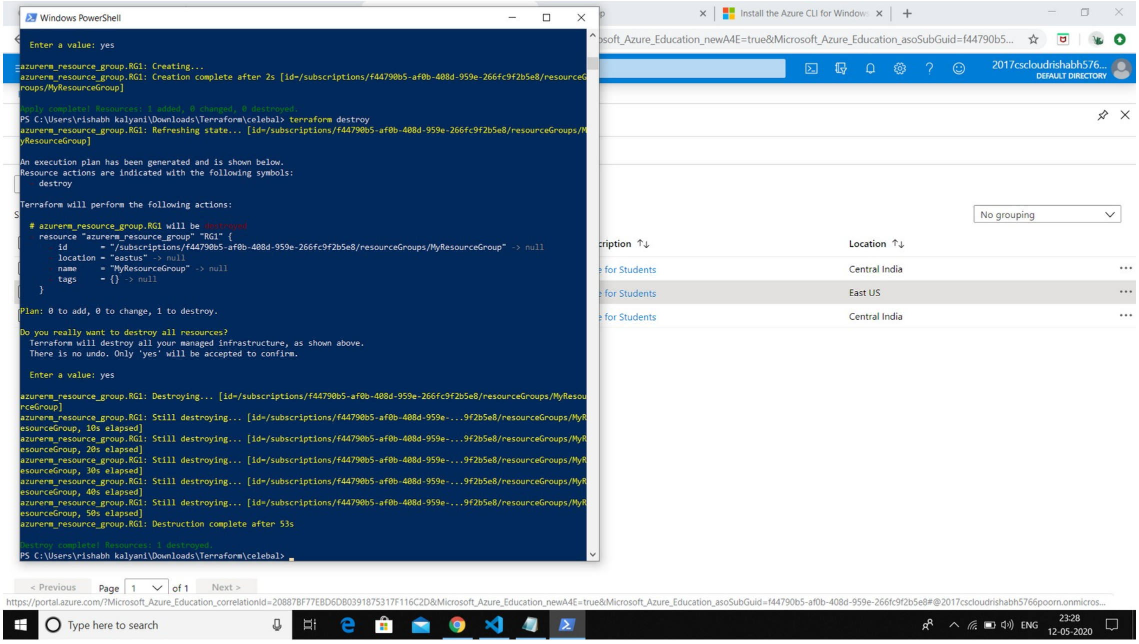
```
Blocks of type "tags" are not expected here. Did you mean to define argument
"tags"? If so, use the equals sign to assign it a value.

PS C:\Users\rishabh kalyani\Downloads\Terraform\celebal> terraform plan

Error: Argument or block definition required

  on main.tf line 14:
  14: }

An argument or block definition is required here.

PS C:\Users\rishabh kalyani\Downloads\Terraform\celebal> terraform plan
Refreshing Terraform state in-memory prior to plan...
The refreshed state will be used to calculate this plan, but will not be
persisted to local or remote state storage.


------------------------------------------------------------------------

An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # azurerm_resource_group.RG1 will be created
  + resource "azurerm_resource_group" "RG1" {
      + id       = (known after apply)
      + location = "eastus"
      + name     = "MyResourceGroup"
    }

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

------------------------------------------------------------------------

Note: You didn't specify an "-out" parameter to save this plan, so Terraform
can't guarantee that exactly these actions will be performed if
"terraform apply" is subsequently run.

PS C:\Users\rishabh kalyani\Downloads\Terraform\celebal> terraform apply
```

2. *Now ,in this output, resource group , is created and it asking for approval to continue:-*



```
Refreshing Terraform state in-memory prior to plan...
The refreshed state will be used to calculate this plan, but will not be
persisted to local or remote state storage.


------------------------------------------------------------------------

An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # azurerm_resource_group.RG1 will be created
  + resource "azurerm_resource_group" "RG1" {
      + id       = (known after apply)
      + location = "eastus"
      + name     = "MyResourceGroup"
    }

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

------------------------------------------------------------------------

Note: You didn't specify an "-out" parameter to save this plan, so Terraform
can't guarantee that exactly these actions will be performed if
"terraform apply" is subsequently run.

PS C:\Users\rishabh kalyani\Downloads\Terraform\celebal> terraform apply

An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # azurerm_resource_group.RG1 will be created
  + resource "azurerm_resource_group" "RG1" {
      + id       = (known after apply)
      + location = "eastus"
      + name     = "MyResourceGroup"
    }

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

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes
```

3. *It asking for the value to enter , we just simply write the yes for further resources and installation :-*

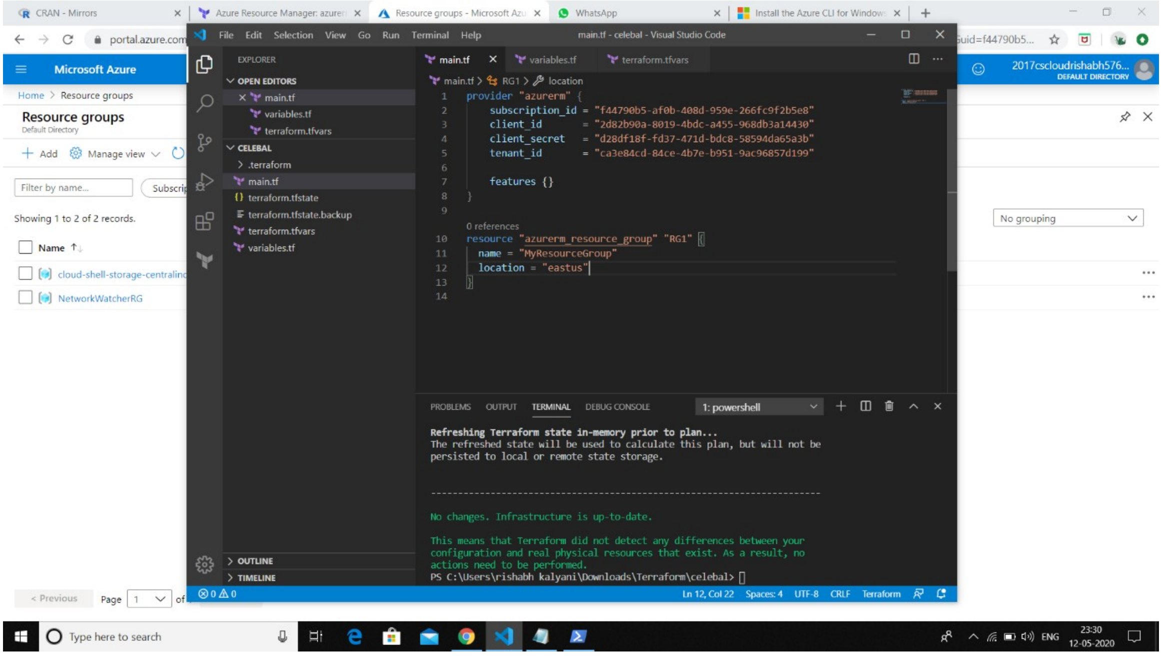


```
  Enter a value: yes

azurerm_resource_group.RG1: Creating...
azurerm_resource_group.RG1: Creation complete after 2s [id=/subscriptions/f44790b5-af0b-408d-959e-266fc9f2b5e8/resourceG
roups/MyResourceGroup]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
PS C:\Users\rishabh kalyani\Downloads\Terraform\celebal> terraform destroy
azurerm_resource_group.RG1: Refreshing state... [id=/subscriptions/f44790b5-af0b-408d-959e-266fc9f2b5e8/resourceGroups/M
yResourceGroup]

An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
  - destroy

Terraform will perform the following actions:

  # azurerm_resource_group.RG1 will be destroyed
  - resource "azurerm_resource_group" "RG1" {
      - id       = "/subscriptions/f44790b5-af0b-408d-959e-266fc9f2b5e8/resourceGroups/MyResourceGroup" -> null
      - location = "eastus" -> null
      - name     = "MyResourceGroup" -> null
      - tags     = {} -> null
    }

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

Do you really want to destroy all resources?
  Terraform will destroy all your managed infrastructure, as shown above.
  There is no undo. Only 'yes' will be accepted to confirm.

  Enter a value: yes

azurerm_resource_group.RG1: Destroying... [id=/subscriptions/f44790b5-af0b-408d-959e-266fc9f2b5e8/resourceGroups/MyResou
rceGroup]
azurerm_resource_group.RG1: Still destroying... [id=/subscriptions/f44790b5-af0b-408d-959e-...9f2b5e8/resourceGroups/MyR
esourceGroup, 10s elapsed]
azurerm_resource_group.RG1: Still destroying... [id=/subscriptions/f44790b5-af0b-408d-959e-...9f2b5e8/resourceGroups/MyR
esourceGroup, 20s elapsed]
azurerm_resource_group.RG1: Still destroying... [id=/subscriptions/f44790b5-af0b-408d-959e-...9f2b5e8/resourceGroups/MyR
esourceGroup, 30s elapsed]
azurerm_resource_group.RG1: Still destroying... [id=/subscriptions/f44790b5-af0b-408d-959e-...9f2b5e8/resourceGroups/MyR
esourceGroup, 40s elapsed]
azurerm_resource_group.RG1: Still destroying... [id=/subscriptions/f44790b5-af0b-408d-959e-...9f2b5e8/resourceGroups/MyR
esourceGroup, 50s elapsed]
azurerm_resource_group.RG1: Destruction complete after 53s

Destroy complete! Resources: 1 destroyed.
PS C:\Users\rishabh kalyani\Downloads\Terraform\celebal>
```

✚ *Create the terraform file*

Let's create our terraform file and name it ***main.tf***

4. *Authenticating using service principal to make role-based provisions*

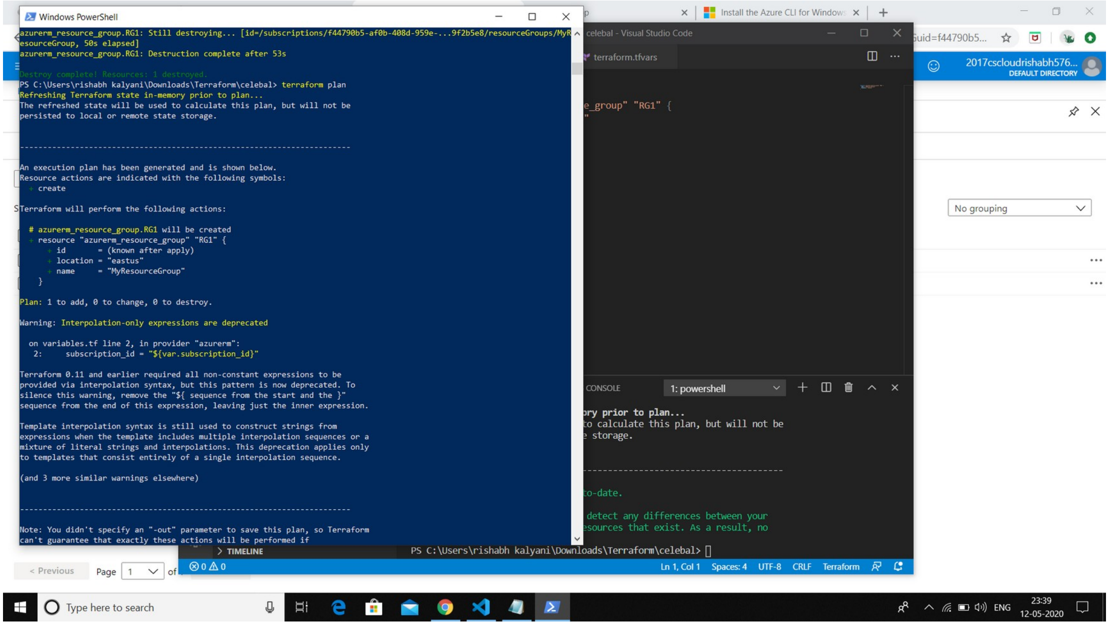


```
provider "azurerm" {
    subscription_id = "f44790b5-af0b-408d-959e-266fc9f2b5e8"
    client_id       = "2d82b90a-8019-4bdc-a455-968db3a14430"
    client_secret   = "d28df18f-fd37-471d-bdc8-58594da65a3b"
    tenant_id       = "ca3e84cd-84ce-4b7e-b951-9ac96857d199"

    features {}
}

resource "azurerm_resource_group" "RG1" {
  name = "MyResourceGroup"
  location = "eastus"
}
```

```
Refreshing Terraform state in-memory prior to plan...
The refreshed state will be used to calculate this plan, but will not be
persisted to local or remote state storage.


------------------------------------------------------------------------

No changes. Infrastructure is up-to-date.

This means that Terraform did not detect any differences between your
configuration and real physical resources that exist. As a result, no
actions need to be performed.
PS C:\Users\rishabh kalyani\Downloads\Terraform\celebal>
```

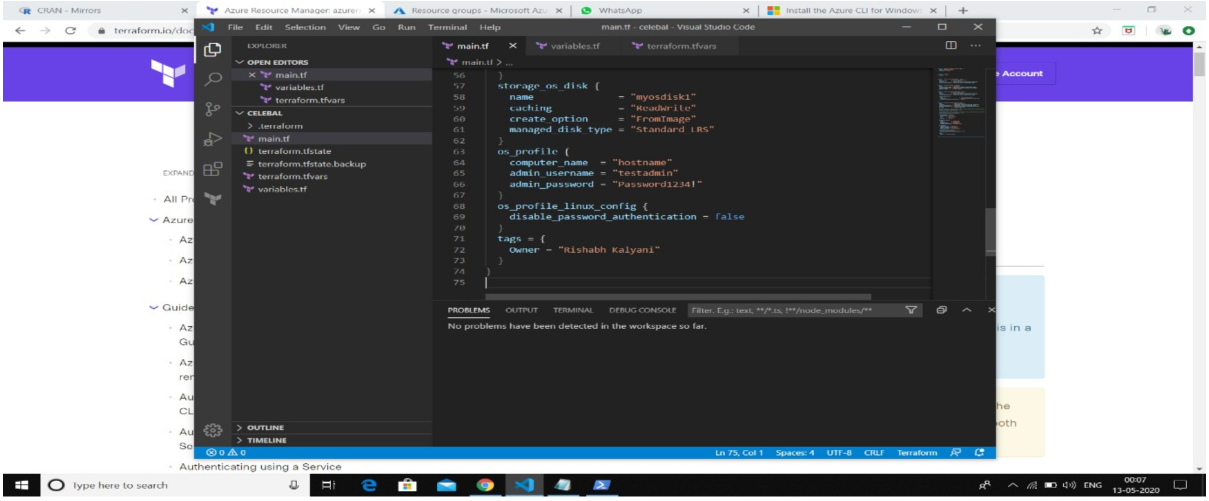
5. ***Define the Azure resource group***

Now let's create our new resource group that everything will live inside



```
azurerm_resource_group.RG1: Still destroying... [id=/subscriptions/f44790b5-af0b-408d-959e-...9f2b5e8/resourceGroups/MyResourceGroup, 50s elapsed]
azurerm_resource_group.RG1: Destruction complete after 53s

Destroy complete! Resources: 1 destroyed.
PS C:\Users\rishabh kalyani\Downloads\Terraform\celebal> terraform plan
Refreshing Terraform state in-memory prior to plan...
The refreshed state will be used to calculate this plan, but will not be
persisted to local or remote state storage.


------------------------------------------------------------------------

An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # azurerm_resource_group.RG1 will be created
  + resource "azurerm_resource_group" "RG1" {
      + id       = (known after apply)
      + location = "eastus"
      + name     = "MyResourceGroup"
    }

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

Warning: Interpolation-only expressions are deprecated

  on variables.tf line 2, in provider "azurerm":
   2:     subscription_id = "${var.subscription_id}"

Terraform 0.11 and earlier required all non-constant expressions to be
provided via interpolation syntax, but this pattern is now deprecated. To
silence this warning, remove the "${ sequence from the start and the }"
sequence from the end of this expression, leaving just the inner expression.

Template interpolation syntax is still used to construct strings from
expressions when the template includes multiple interpolation sequences or a
mixture of literal strings and interpolations. This deprecation applies only
to templates that consist entirely of a single interpolation sequence.

(and 3 more similar warnings elsewhere)


------------------------------------------------------------------------

Note: You didn't specify an "-out" parameter to save this plan, so Terraform
can't guarantee that exactly these actions will be performed if
```

6. *The following section creates a os disk named myosdisk1 and provide the permission of read/write. A user named “TESTADMIN” is created with admin password authentication.*



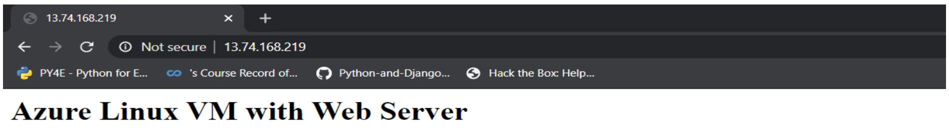
```
    }
    storage_os_disk {
      name              = "myosdisk1"
      caching           = "ReadWrite"
      create_option     = "FromImage"
      managed_disk_type = "Standard_LRS"
    }
    os_profile {
      computer_name  = "hostname"
      admin_username = "testadmin"
      admin_password = "Password1234!"
    }
    os_profile_linux_config {
      disable_password_authentication = false
    }
    tags = {
      Owner = "Rishabh Kalyani"
    }
}
```

No problems have been detected in the workspace so far.

NOW IN THE , main.tf FILE WE DO OUR APACHE INSTALLATION :-

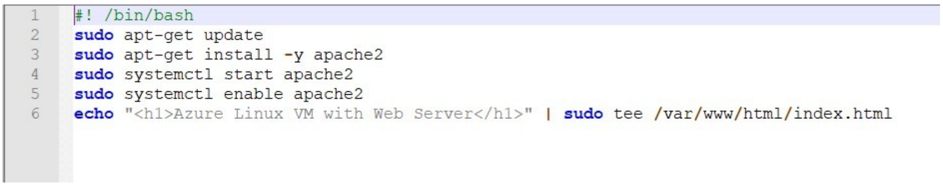
❖ *Now let’s see the procedure of apache installation.*

1. *Make a file with .sh extension.*



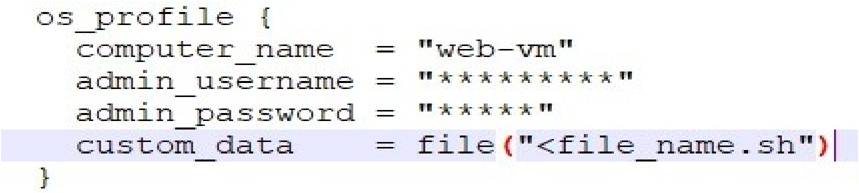
Azure Linux VM with Web Server

2. *Now ,write the apache installation commands.*



```
#! /bin/bash
sudo apt-get update
sudo apt-get install -y apache2
sudo systemctl start apache2
sudo systemctl enable apache2
echo "<h1>Azure Linux VM with Web Server</h1>" | sudo tee /var/www/html/index.html
```

3. *The following section creates a os profile named “web-vm” . A user named is created with admin password authentication.*



```
os_profile {
   computer_name   = "web-vm"
   admin_username = "**********"
   admin_password = "******"
   custom_data     = file("<file_name.sh")
}
```

Hence, we are done here for our apache configuration.

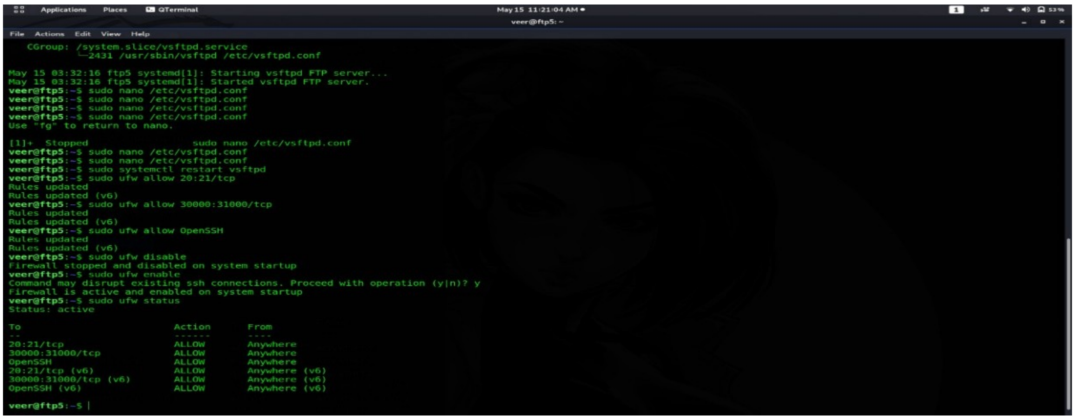
- ***NOW IN THE , main.tf FILE WE DO OUR FTP SERVER INSTALLATION :-***

❖ *Now let’s see the procedure of FTP SERVER installation.*

1. *First, we need to update the system package sources list and then install VSFTPD binary package.*

2. *Once the installation completes, the service will be disabled initially, therefore, we need to start it manually for the mean time and also enable it to start automatically from the next system boot.*

3. *Next, if you have UFW firewall enabled (its not enabled by default) on the server, you have to open ports 21 and 20 where the FTP daemons are listening, in order to allow access to FTP services from remote machines, then add the new firewall rules.*

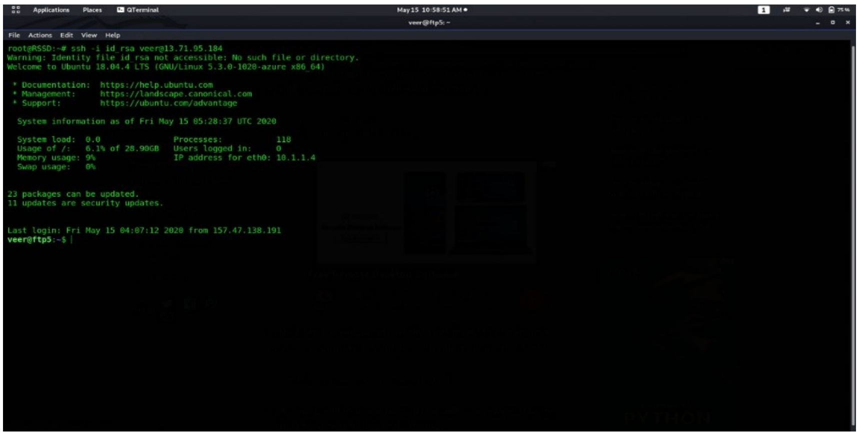


```
   CGroup: /system.slice/vsftpd.service
           └─2431 /usr/sbin/vsftpd /etc/vsftpd.conf

May 15 03:32:16 ftp5 systemd[1]: Starting vsftpd FTP server...
May 15 03:32:16 ftp5 systemd[1]: Started vsftpd FTP server.
veer@ftp5:~$ sudo nano /etc/vsftpd.conf
veer@ftp5:~$ sudo nano /etc/vsftpd.conf
veer@ftp5:~$ sudo nano /etc/vsftpd.conf
veer@ftp5:~$ sudo nano /etc/vsftpd.conf
Use "fg" to return to nano.

[1]+  Stopped                 sudo nano /etc/vsftpd.conf
veer@ftp5:~$ sudo nano /etc/vsftpd.conf
veer@ftp5:~$ sudo nano /etc/vsftpd.conf
veer@ftp5:~$ sudo systemctl restart vsftpd
veer@ftp5:~$ sudo ufw allow 20:21/tcp
Rules updated
Rules updated (v6)
veer@ftp5:~$ sudo ufw allow 30000:31000/tcp
Rules updated
Rules updated (v6)
veer@ftp5:~$ sudo ufw allow OpenSSH
Rules updated
Rules updated (v6)
veer@ftp5:~$ sudo ufw disable
Firewall stopped and disabled on system startup
veer@ftp5:~$ sudo ufw enable
Command may disrupt existing ssh connections. Proceed with operation (y|n)? y
Firewall is active and enabled on system startup
veer@ftp5:~$ sudo ufw status
Status: active

To                         Action      From
--                         ------      ----
20:21/tcp                  ALLOW       Anywhere
30000:31000/tcp            ALLOW       Anywhere
OpenSSH                    ALLOW       Anywhere
20:21/tcp (v6)             ALLOW       Anywhere (v6)
30000:31000/tcp (v6)       ALLOW       Anywhere (v6)
OpenSSH (v6)               ALLOW       Anywhere (v6)

veer@ftp5:~$ |
```

4. *Let’s now perform a few configurations to setup and secure our FTP server, first we will create a backup of the original config file /etc/vsftpd/vsftpd.conf .*

5. *Next, let’s open the vsftpd config file.*

6. *Now, configure VSFTPD to allow/deny FTP access to users based on the user list file /etc/vsftpd.userlist.*

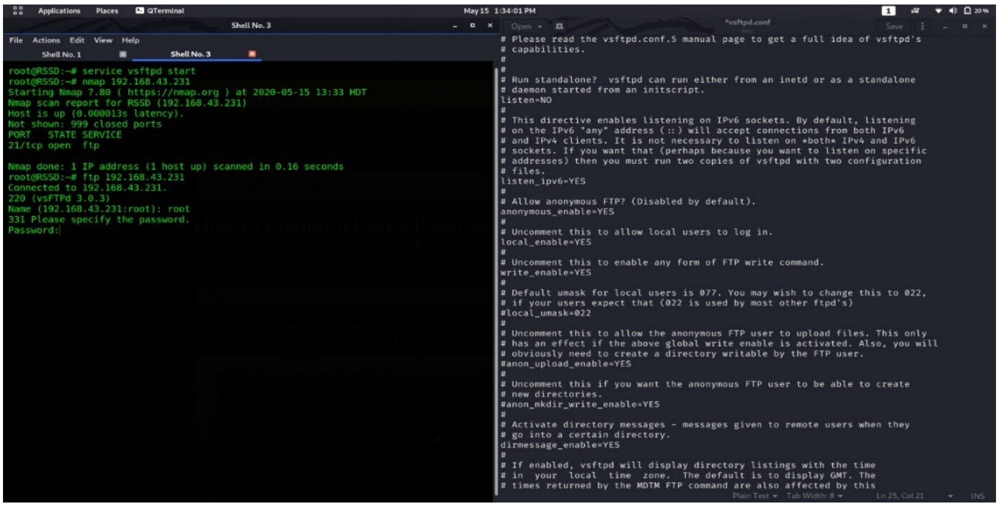


```
root@RSSD:~# ssh -i id_rsa veer@13.71.95.184
Warning: Identity file id_rsa not accessible: No such file or directory.
Welcome to Ubuntu 18.04.4 LTS (GNU/Linux 5.3.0-1020-azure x86_64)

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

  System information as of Fri May 15 05:28:37 UTC 2020

  System load:  0.0               Processes:           118
  Usage of /:   6.1% of 28.90GB   Users logged in:     0
  Memory usage: 9%                IP address for eth0: 10.1.1.4
  Swap usage:   0%


23 packages can be updated.
11 updates are security updates.


Last login: Fri May 15 04:07:12 2020 from 157.47.138.191
veer@ftp5:~$ 
```

7. *Save the file and close it. Then we have to restart VSFTPD services for the changes above to take effect.*

8. *Next, let’s test if a user not listed in the file /etc/vsftpd.userlist will be granted permission to login.*

9. *Now we will carry out a final test to determine whether a user listed in the file /etc/vsftpd.userlist, is actually placed in his/her home directory after login.*



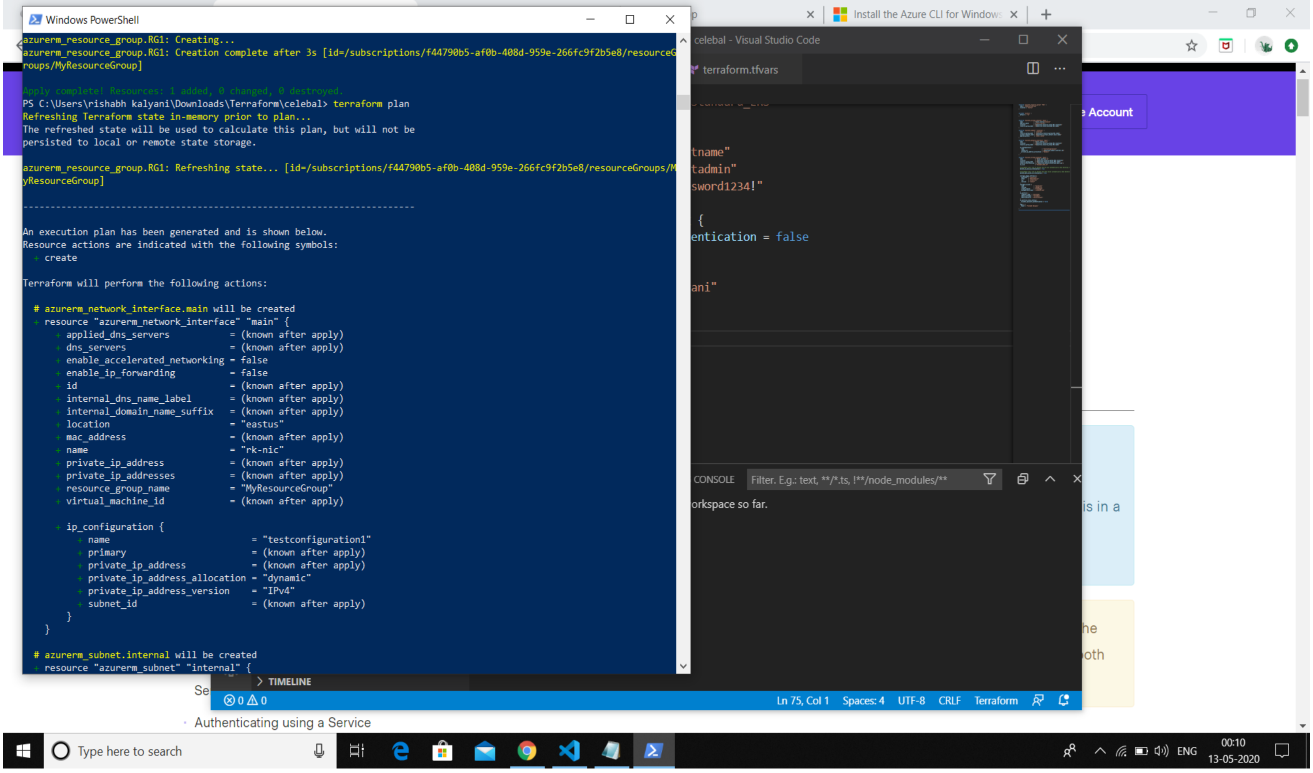
```
root@RSSD:~# service vsftpd start
root@RSSD:~# nmap 192.168.43.231
Starting Nmap 7.80 ( https://nmap.org ) at 2020-05-15 13:33 HDT
Nmap scan report for RSSD (192.168.43.231)
Host is up (0.000013s latency).
Not shown: 999 closed ports
PORT   STATE SERVICE
21/tcp open  ftp

Nmap done: 1 IP address (1 host up) scanned in 0.16 seconds
root@RSSD:~# ftp 192.168.43.231
Connected to 192.168.43.231.
220 (vsFTPd 3.0.3)
Name (192.168.43.231:root): root
331 Please specify the password.
Password:
```

```
# Please read the vsftpd.conf.5 manual page to get a full idea of vsftpd's
# capabilities.
#
#
# Run standalone?  vsftpd can run either from an inetd or as a standalone
# daemon started from an initscript.
listen=NO
#
# This directive enables listening on IPv6 sockets. By default, listening
# on the IPv6 "any" address (::) will accept connections from both IPv6
# and IPv4 clients. It is not necessary to listen on *both* IPv4 and IPv6
# sockets. If you want that (perhaps because you want to listen on specific
# addresses) then you must run two copies of vsftpd with two configuration
# files.
listen_ipv6=YES
#
# Allow anonymous FTP? (Disabled by default).
anonymous_enable=YES
#
# Uncomment this to allow local users to log in.
local_enable=YES
#
# Uncomment this to enable any form of FTP write command.
write_enable=YES
#
# Default umask for local users is 077. You may wish to change this to 022,
# if your users expect that (022 is used by most other ftpd's)
#local_umask=022
#
# Uncomment this to allow the anonymous FTP user to upload files. This only
# has an effect if the above global write enable is activated. Also, you will
# obviously need to create a directory writable by the FTP user.
#anon_upload_enable=YES
#
# Uncomment this if you want the anonymous FTP user to be able to create
# new directories.
#anon_mkdir_write_enable=YES
#
# Activate directory messages - messages given to remote users when they
# go into a certain directory.
dirmessage_enable=YES
#
# If enabled, vsftpd will display directory listings with the time
# in  your  local  time  zone.  The default is to display GMT. The
# times returned by the MDTM FTP command are also affected by this
```

Now, you should have installed an FTP server.

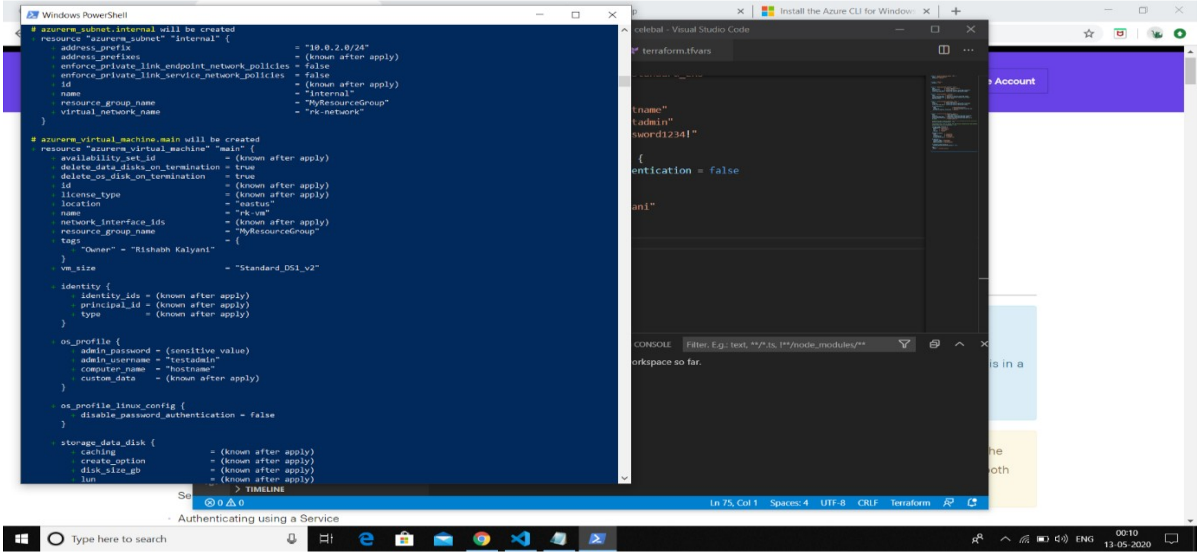
***You should now be able to configure your user lists and accounts, and connect to your new FTP server. We also detailed the risks of the FTP protocol, and how to mitigate them**.*

7. *Here , our azurerm main network interface will define for VM:-*



```
azurerm_resource_group.RG1: Creating...
azurerm_resource_group.RG1: Creation complete after 3s [id=/subscriptions/f44790b5-af0b-408d-959e-266fc9f2b5e8/resourceG
roups/MyResourceGroup]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
PS C:\Users\rishabh kalyani\Downloads\Terraform\celebal> terraform plan
Refreshing Terraform state in-memory prior to plan...
The refreshed state will be used to calculate this plan, but will not be
persisted to local or remote state storage.

azurerm_resource_group.RG1: Refreshing state... [id=/subscriptions/f44790b5-af0b-408d-959e-266fc9f2b5e8/resourceGroups/M
yResourceGroup]

------------------------------------------------------------------------

An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # azurerm_network_interface.main will be created
  + resource "azurerm_network_interface" "main" {
      + applied_dns_servers           = (known after apply)
      + dns_servers                   = (known after apply)
      + enable_accelerated_networking = false
      + enable_ip_forwarding          = false
      + id                            = (known after apply)
      + internal_dns_name_label       = (known after apply)
      + internal_domain_name_suffix   = (known after apply)
      + location                      = "eastus"
      + mac_address                   = (known after apply)
      + name                          = "rk-nic"
      + private_ip_address            = (known after apply)
      + private_ip_addresses          = (known after apply)
      + resource_group_name           = "MyResourceGroup"
      + virtual_machine_id            = (known after apply)

      + ip_configuration {
          + name                          = "testconfiguration1"
          + primary                       = (known after apply)
          + private_ip_address            = (known after apply)
          + private_ip_address_allocation = "dynamic"
          + private_ip_address_version    = "IPv4"
          + subnet_id                     = (known after apply)
        }
    }

  # azurerm_subnet.internal will be created
  + resource "azurerm_subnet" "internal" {
```

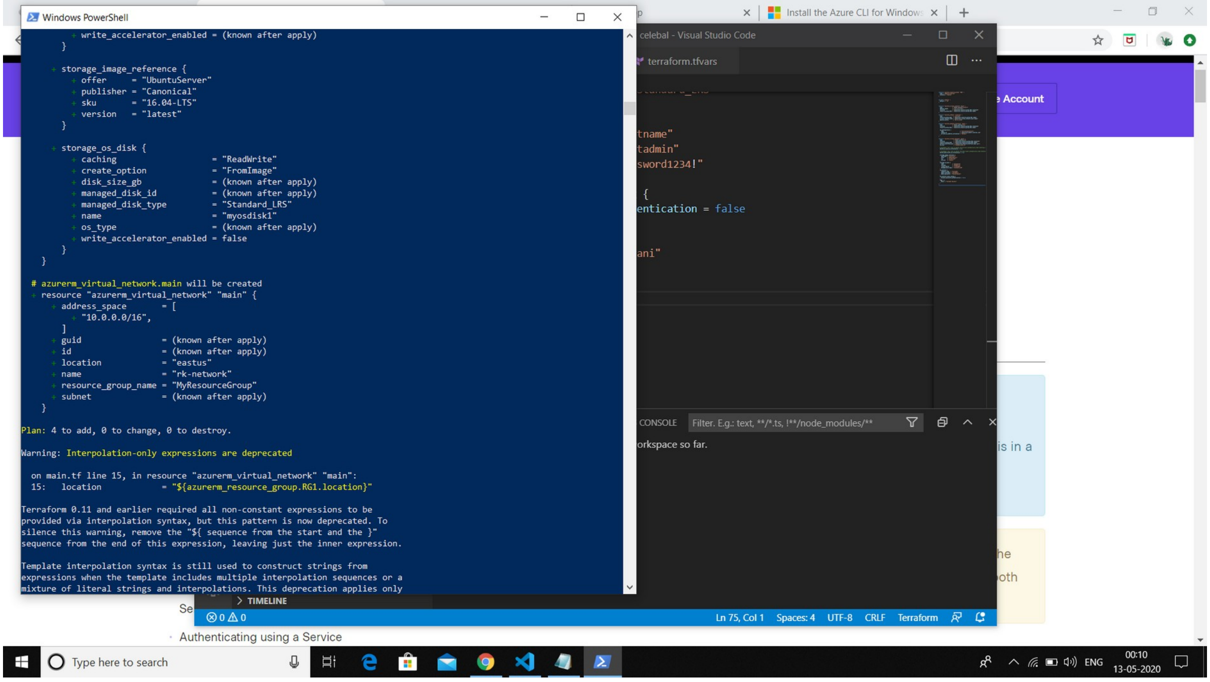
8. ***The final step is to define the VM and use all the resources.,***



```
# azurerm_subnet.internal will be created
+ resource "azurerm_subnet" "internal" {
    + address_prefix                                 = "10.0.2.0/24"
    + address_prefixes                               = (known after apply)
    + enforce_private_link_endpoint_network_policies = false
    + enforce_private_link_service_network_policies  = false
    + id                                             = (known after apply)
    + name                                           = "internal"
    + resource_group_name                            = "MyResourceGroup"
    + virtual_network_name                           = "rk-network"
  }

# azurerm_virtual_machine.main will be created
+ resource "azurerm_virtual_machine" "main" {
    + availability_set_id              = (known after apply)
    + delete_data_disks_on_termination = true
    + delete_os_disk_on_termination    = true
    + id                               = (known after apply)
    + license_type                     = (known after apply)
    + location                         = "eastus"
    + name                             = "rk-vm"
    + network_interface_ids            = (known after apply)
    + resource_group_name              = "MyResourceGroup"
    + tags                             = {
        + "Owner" = "Rishabh Kalyani"
      }
    + vm_size                          = "Standard_DS1_v2"

    + identity {
        + identity_ids = (known after apply)
        + principal_id = (known after apply)
        + type         = (known after apply)
      }

    + os_profile {
        + admin_password = (sensitive value)
        + admin_username = "testadmin"
        + computer_name  = "hostname"
        + custom_data    = (known after apply)
      }

    + os_profile_linux_config {
        + disable_password_authentication = false
      }

    + storage_data_disk {
        + caching                   = (known after apply)
        + create_option             = (known after apply)
        + disk_size_gb              = (known after apply)
        + lun                       = (known after apply)
```

celebal - Visual Studio Code
terraform.tfvars
Install the Azure CLI for Windows
CONSOLE
Filter. E.g.: text, **/*.ts, !**/node_modules/**
orkspace so far.
TIMELINE
Ln 75, Col 1
Spaces: 4
UTF-8
CRLF
Terraform
Authenticating using a Service
Type here to search
ENG
00:10
13-05-2020

9.*Define a virtual network and subnet :-*



```
        + write_accelerator_enabled = (known after apply)
      }

    + storage_image_reference {
        + offer     = "UbuntuServer"
        + publisher = "Canonical"
        + sku       = "16.04-LTS"
        + version   = "latest"
      }

    + storage_os_disk {
        + caching                   = "ReadWrite"
        + create_option             = "FromImage"
        + disk_size_gb              = (known after apply)
        + managed_disk_id           = (known after apply)
        + managed_disk_type         = "Standard_LRS"
        + name                      = "myosdisk1"
        + os_type                   = (known after apply)
        + write_accelerator_enabled = false
      }
  }

# azurerm_virtual_network.main will be created
+ resource "azurerm_virtual_network" "main" {
    + address_space       = [
        + "10.0.0.0/16",
      ]
    + guid                = (known after apply)
    + id                  = (known after apply)
    + location            = "eastus"
    + name                = "rk-network"
    + resource_group_name = "MyResourceGroup"
    + subnet              = (known after apply)
  }

Plan: 4 to add, 0 to change, 0 to destroy.

Warning: Interpolation-only expressions are deprecated

  on main.tf line 15, in resource "azurerm_virtual_network" "main":
  15:   location            = "${azurerm_resource_group.RG1.location}"

Terraform 0.11 and earlier required all non-constant expressions to be
provided via interpolation syntax, but this pattern is now deprecated. To
silence this warning, remove the "${ sequence from the start and the }"
sequence from the end of this expression, leaving just the inner expression.

Template interpolation syntax is still used to construct strings from
expressions when the template includes multiple interpolation sequences or a
mixture of literal strings and interpolations. This deprecation applies only
```

celebal - Visual Studio Code
terraform.tfvars
Install the Azure CLI for Windows
CONSOLE
Filter. E.g.: text, **/*.ts, !**/node_modules/**
orkspace so far.
TIMELINE
Ln 75, Col 1
Spaces: 4
UTF-8
CRLF
Terraform
Authenticating using a Service
Type here to search
ENG
00:10
13-05-2020

Running the code

1. open the cmd from the terraform project location

2. run terraform init

After successful initialization of terraform plan the project

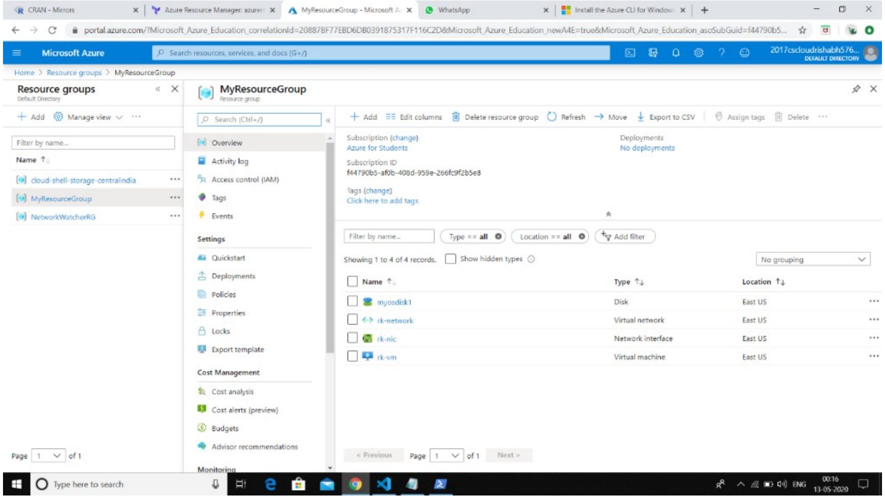
3. run terraform plan

All the created resources and other things should display on the console. Type yes to execute the plan

- *run terraform apply.*

10. *After you execute the previous command, you should see something like the following screen:*

Here , our creation of VM using terraform is successfully completed.



MyResourceGroup
Resource group

Subscription (change): Azure for Students

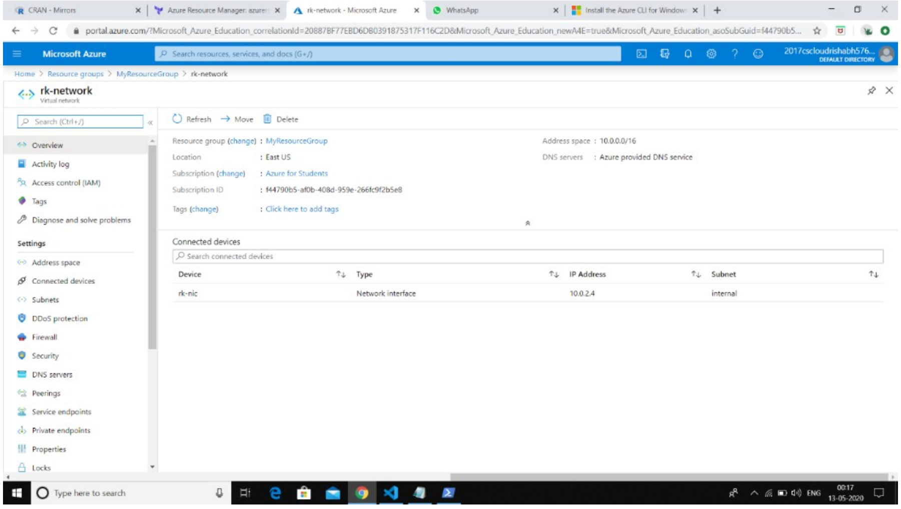
Subscription ID: f44790b5-af0b-408d-959e-266fc9f2b5e8

Tags (change): Click here to add tags

Deployments: No deployments

Showing 1 to 4 of 4 records.

Name	Type	Location
myosdisk1	Disk	East US
rk-network	Virtual network	East US
rk-nic	Network interface	East US
rk-vm	Virtual machine	East US



rk-network
Virtual network

Resource group (change) : MyResourceGroup

Location : East US

Subscription (change) : Azure for Students

Subscription ID : f44790b5-af0b-408d-959e-266fc9f2b5e8

Tags (change) : Click here to add tags

Address space : 10.0.0.0/16

DNS servers : Azure provided DNS service

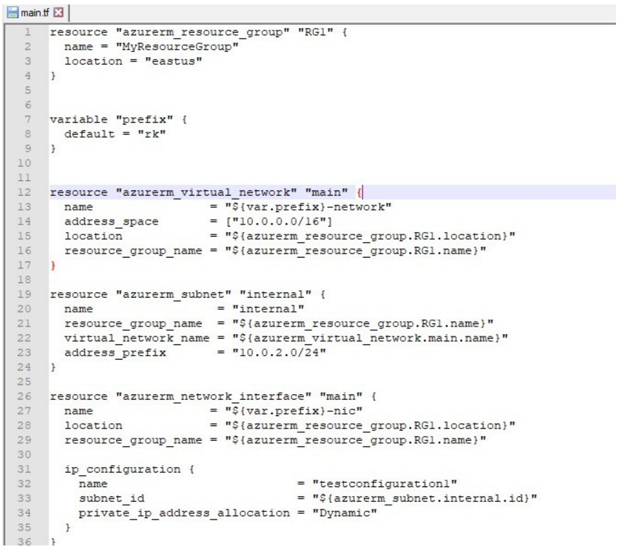
Connected devices

Device	Type	IP Address	Subnet
rk-nic	Network interface	10.0.2.4	internal

The CREATION of VM is completed .

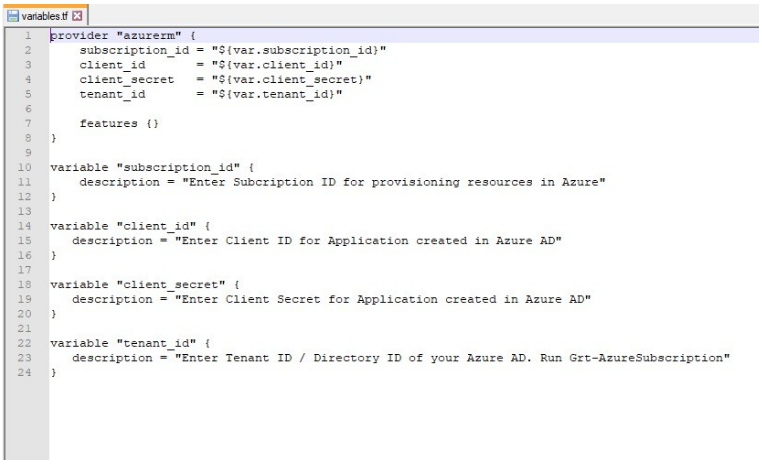
So there we have it, a new Virtual Machine built in Azure using terraform.

RESOURCE CONFIGURATION FILE FOR VM:-



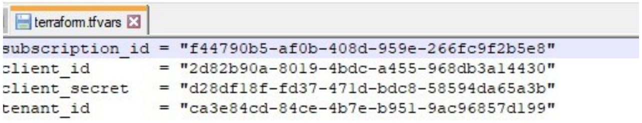
```
resource "azurerm_resource_group" "RG1" {
  name = "MyResourceGroup"
  location = "eastus"
}


variable "prefix" {
  default = "rk"
}


resource "azurerm_virtual_network" "main" {
  name                = "${var.prefix}-network"
  address_space       = ["10.0.0.0/16"]
  location            = "${azurerm_resource_group.RG1.location}"
  resource_group_name = "${azurerm_resource_group.RG1.name}"
}

resource "azurerm_subnet" "internal" {
  name                 = "internal"
  resource_group_name  = "${azurerm_resource_group.RG1.name}"
  virtual_network_name = "${azurerm_virtual_network.main.name}"
  address_prefix       = "10.0.2.0/24"
}

resource "azurerm_network_interface" "main" {
  name                = "${var.prefix}-nic"
  location            = "${azurerm_resource_group.RG1.location}"
  resource_group_name = "${azurerm_resource_group.RG1.name}"

  ip_configuration {
    name                          = "testconfiguration1"
    subnet_id                     = "${azurerm_subnet.internal.id}"
    private_ip_address_allocation = "Dynamic"
  }
}
```

Figure 1 main.tf



```
provider "azurerm" {
    subscription_id = "${var.subscription_id}"
    client_id       = "${var.client_id}"
    client_secret   = "${var.client_secret}"
    tenant_id       = "${var.tenant_id}"

    features {}
}

variable "subscription_id" {
    description = "Enter Subcription ID for provisioning resources in Azure"
}

variable "client_id" {
   description = "Enter Client ID for Application created in Azure AD"
}

variable "client_secret" {
   description = "Enter Client Secret for Application created in Azure AD"
}

variable "tenant_id" {
   description = "Enter Tenant ID / Directory ID of your Azure AD. Run Grt-AzureSubscription"
}
```

Figure 2 variable.tf



```
subscription_id = "f44790b5-af0b-408d-959e-266fc9f2b5e8"
client_id       = "2d82b90a-8019-4bdc-a455-968db3a14430"
client_secret   = "d28df18f-fd37-471d-bdc8-58594da65a3b"
tenant_id       = "ca3e84cd-84ce-4b7e-b951-9ac96857d199"
```

Figure 3 terraform.tfvar

5.RESULT :-

We learn to configure ftp and apache using terraform.

6.CONCLUSION:-

***TERRAFORM** is the configuration orchestration tool that works with any cloud, be it private on-prem or public system, and allows safe and convenient design, management and improvement for infrastructure as code. As a part of Hashicorp stack, including also Vagrant, Packer, Consul, Vault, and Nomad, Terraform helps provision any application written in any language to any infrastructure.*

Terraform is the example of next generation of configuration orchestration systems bringing a new layer of features and functionalities to the table.

7.REFRENCES:-

1. https://learn.hashicorp.com/terraform
2. https://docs.microsoft.com/en-us/azure/virtual-machines/windows/quick-create-portal
3. https://owendavies.net/articles/create-azure-virtual-machine-with-terraform/

Name of Use Case:	MICROSOFT AZURE , TERRAFORM		
Created By:	TEAM 11	**Last Updated By:**	SHAILLY SHAH
Date Created:	13/05/2020	**Last Revision Date:**	14/05/2020