**Devops CSYE7220**

**Homework 5**

**Harshitha Somasundar**

**Deploying Docker container on AWS:**

1. Create your vm instance with Docker installed in **AWS** Terraform;  
   Create a providers.tf to enable access to AWS account   
   in Providers.tf with access key and secret key
2. provider "aws" {
3. region     = var.aws\_region
4. access\_key = ""
5. secret\_key = ""
6. }

2)To create a VM Instance first create a base image with AMI (with ubuntu as base image)

1. Install Packer <https://packer.io/downloads.html>. And create a ubuntu.json file  
   Run packer build < ubuntu.json >.json

{

    "variables": {

        "aws\_access\_key": "",

        "aws\_secret\_key": "",

        "aws\_region": "us-east-1",

        "subnet\_id": "subnet-003f0f5ee5c1ada54",

        "source\_ami": "ami-07ebfd5b3428b6f4d",

        "ssh\_username": "ubuntu"

    },

    "builders": [

        {

            "type": "amazon-ebs",

            "access\_key": "{{user `aws\_access\_key`}}",

            "secret\_key": "{{user `aws\_secret\_key`}}",

            "region": "{{user `aws\_region`}}",

            "instance\_type": "t2.micro",

            "subnet\_id": "{{user `subnet\_id`}}",

            "source\_ami": "{{user `source\_ami`}}",

            "ssh\_username": "{{user `ssh\_username`}}",

            "ami\_name": "devops\_{{timestamp}}",

            "ami\_description": "ubuntu AMI with dotnet core and docker",

            "launch\_block\_device\_mappings": [

                {

                    "device\_name": "/dev/sda1",

                    "volume\_size": 8,

                    "volume\_type": "gp2",

                    "delete\_on\_termination": true

                }

            ]

        }

    ],

    "provisioners": [

        {

            "type": "shell",

        "environment\_vars": [

                "FOO=foo"

            ],

            "inline": [

                "wget -q https://packages.microsoft.com/config/ubuntu/19.04/packages-microsoft-prod.deb -O packages-microsoft-prod.deb",

                "sudo dpkg -i packages-microsoft-prod.deb",

                "sudo apt-get -y update",

                "sudo apt-get install -y apt-transport-https",

                "sudo apt-get -y update",

                "sudo apt-get install -y dotnet-sdk-3.1",

                "sudo apt -y update",

                "sudo apt install -y apt-transport-https ca-certificates curl software-properties-common",

                "curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -",

                "sudo add-apt-repository \"deb [arch=amd64] https://download.docker.com/linux/ubuntu bionic stable\"",

                "sudo apt -y update",

                "apt-cache policy docker-ce",

                "sudo apt install -y docker-ce",

                "sudo systemctl status docker"

            ]

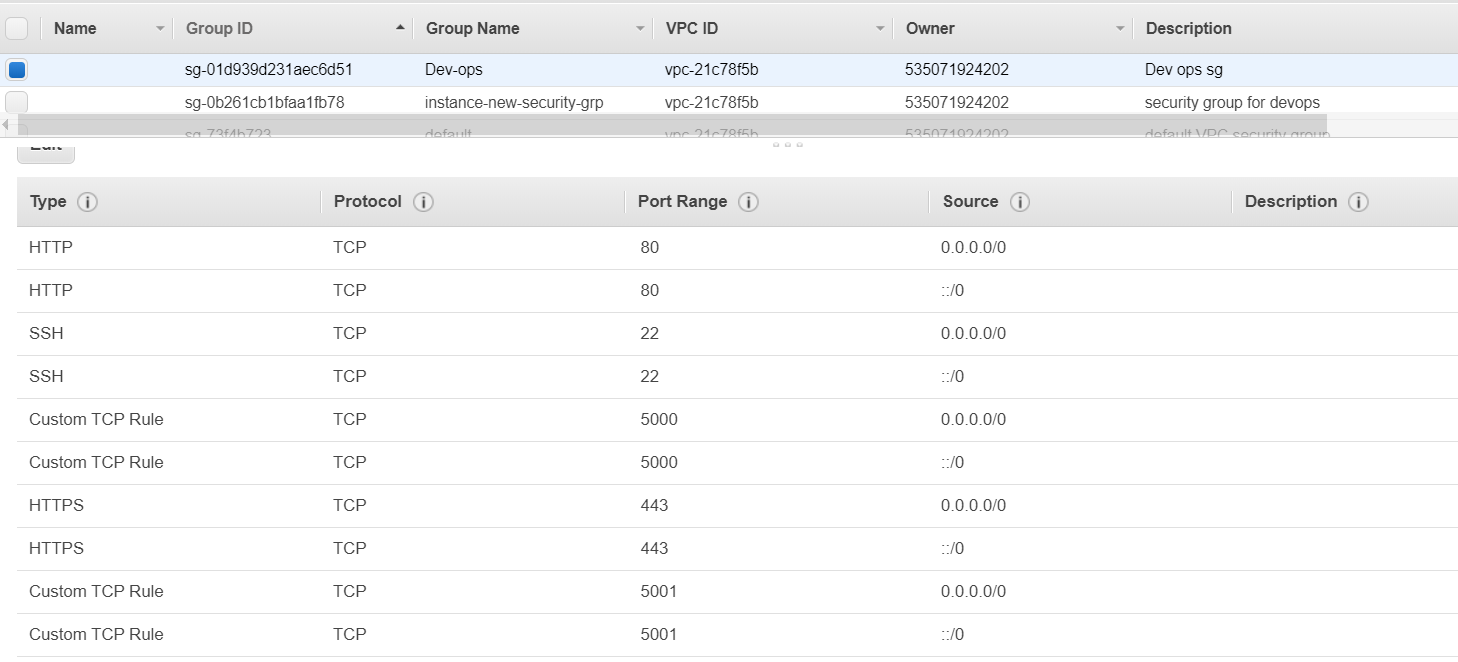
        }

    ]

}

**2. Add this to tf file**

1. data "aws\_ami" "ubuntu" {
2. most\_recent = true
3. filter {
4. name   = "name"
5. values = ["ubuntu/images/hvm-ssd/ubuntu-bionic-18.04-amd64-server-20200112"]
6. }
7. filter {
8. name   = "virtualization-type"
9. values = ["hvm"]
10. }
11. owners = ["099720109477"] # Canonical
12. }

3)Create a security grp in AWS console, for the instance to be accessible across different ports:  
  
  
  
4)Create a main.tf file , which will contain details about which   
🡪 ami to use,  
🡪 what kind of instance we need   
🡪which docker image we need to pull  
🡪security grp required for the instance

resource "aws\_instance" "my-test-instance-harshitha" {

  ami             = "ami-0dc34a024759117e0"

  instance\_type   = "t2.micro"

  security\_groups = ["Dev-ops"]

  key\_name = "devops"

  user\_data = <<-EOF

      #!/bin/bash

      sudo docker pull harshithass/dotnet:v2

      sudo docker run --name test --rm -d -i -t -p 5000:5000 harshithass/dotnet:v2

  EOF

  tags = {

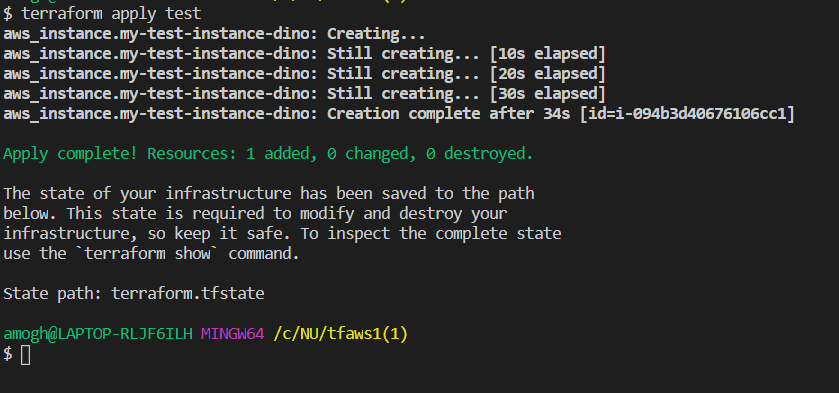
    Name = "test-instance"

  }

}

Once all the tf files are created  
  
  
**1)RUN terraform init  
2)RUN terraform plan**

**3)RUN terraform apply**

  
  
Open a browser and open <public\_ip of your ec2 instance>:5000

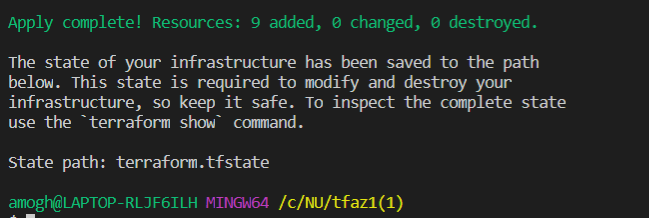
After testing run “Terraform Destroy”

**Deploying Docker container on AZURE:**

1. Insert the azurerm\_virtual\_machine\_extension resource and enter your commands that you need to install docker and pull the image.



1. Run “Terraform init”
2. Run command “Terraform plan -out=test”
3. Run command “Terraform apply test”



1. Open your browser and navigate to <your\_VM\_PUBLIC\_IP>:5000 and you will see the app.
2. Run “Terraform destroy” to destroy your resources.