Task: Using aws terraform code to create an EC2 instance and deploy sample application.

Step1:

—Some requriment need for creating docker image and push into ECR .

These are docker and aws cli.

— docker: I am installing the docker using aws ec2-instance.

**sudo yum install docker -y**

****

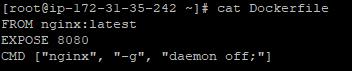
— after that enable the docker and restart the docker with docker

commands

**systemctl enable docker**

**systemctl start docker**

— after that create docker file for build a nginx image.



— AWScli : I am installing the aws cli . for using access the aws console

**curl "https://awscli.amazonaws.com/awscli-exe-linux-x86\_64.zip" -o "awscliv2.zip"**

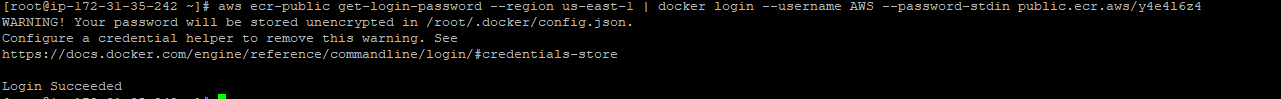
**unzip awscliv2.zip**

— these commands install the aws cli dependencies and packages. It downloads the zip file by using the unzip command to unzip the file.

— then go to aws ECR service and create a new repository.its public or private repositories of our choices.

— In ECR is there one an authentication token and authenticate your Docker client to your registry Use the AWS CLI

**aws ecr-public get-login-password --region us-east-1 | docker login --username AWS --password-stdin public.ecr.aws/y4e4l6z4**



— build your docker image with using following command.if already image will be

there This step will be skip.

– **docker build -t mkreddy .**

— after that your build will be completed . tag your image to ecr repository

— **docker tag mkreddy:latest public.ecr.aws/y4e4l6z4/mkreddy:latest**

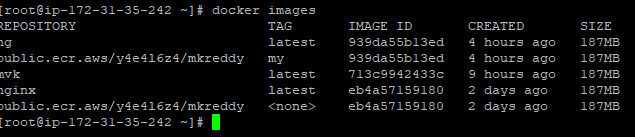
— after run the following command is push the image to newly created ecr

repository.

– **docker push public.ecr.aws/y4e4l6z4/mkreddy:latest**

— if you want show the what images are created using following common

— **docker images**



— now you can go to ECR repository and refresh it shows the what image will be

pushed

Step2:

— in this step create a VPC and subnets and a gateway and ec2-instance in using terraform.

— terraform is infrastructure as code . it's used to create and delete and manage the resources.in terraform so many providers in there like aws and azure etc..

— terraform installation in first install the packages in zip archive

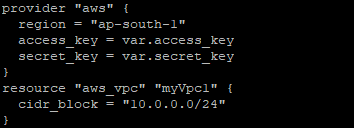
— **sudo yum install -y yum-utils — its manages yum-config-manager**

– **sudo yum-config-manager --add-repo** <https://rpm.releases.hashicorp.com/AmazonLinux/hashicorp.repo> —

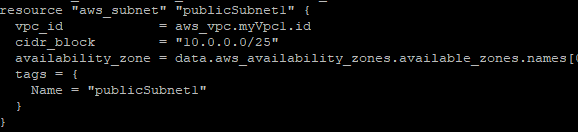
To add the official hashicorp to your linux machine

— **sudo yum -y install terraform** – install the terraform.

Create VPC: VPC is virtual private cloud is a virtual network thar you connect aws cloud. It provided an isolated and customized network environment where you can launch and manage aws resources like ec2 and rds etc..

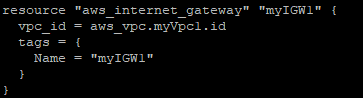


Create Public subnets: public subnets used to resources that need directly access the internets. Such as web services etc..



Crete private subnets: private subnets used to resources that do not require direct internet access.such as backend applications and servers.

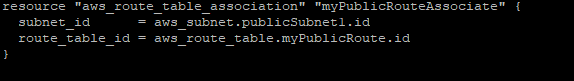
Create Internet gateway: its horizontally scalable, its highly available and redundant components to communicate with VPC and internet.its act as gateway between your VPC and public internet.



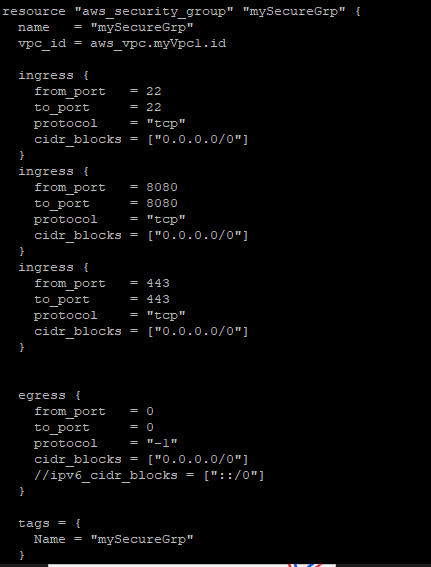
Create route table: a route table is a virtual network component that controls the traffic workflow between subnets and VPC. It determines how network traffic routed in VPC and defines the rules where traffic should be directed.



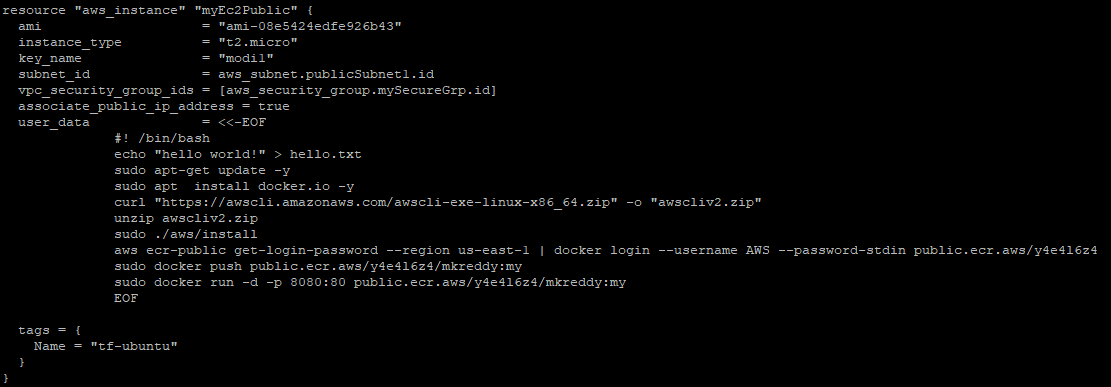
associate subnet with route table its envolves specifying which route table should be used for routing traffic within that subnet.



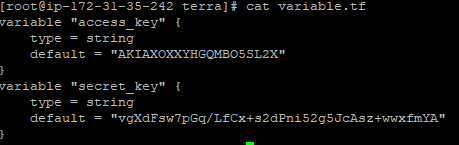
Create security groups: a security group is fundamental component network security that acts as virtual firewall for your ec2-instances.its controls the inbound and outbound traffic rules.



Create ec2-instance : EC2 instances are a core component of AWS and provide flexible and scalable compute resources in the cloud. They enable you to run a wide range of applications, from simple web servers to complex distributed systems, and offer various configuration options to meet your specific needs.

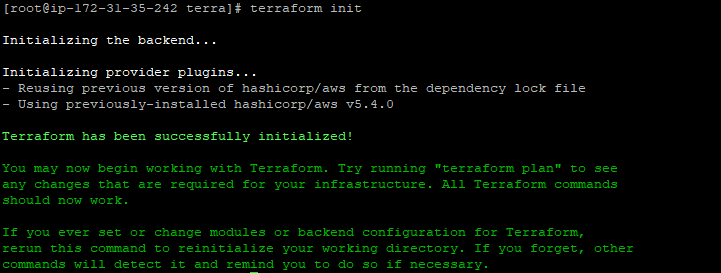


— these all resources are stored in the main.tf file and variables are stored in the variable.tf file.

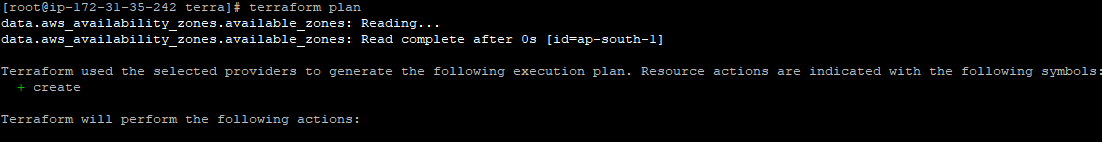


— now run the following commands

– **terraform init —** its setup necessary components and download provided plugins



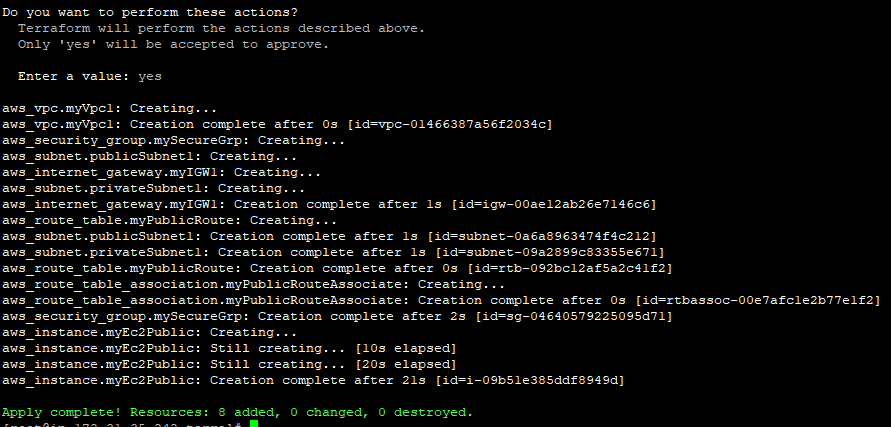
– **terraform plan —** its creates a execution plan



– **terraform validate —** its validate the syntax and configuration of terraform file without creating and modifying infrastructure



– **terraform apply –** its used to apply the changes to define your terraform configuration file create and modify the associated infrastructure resources.

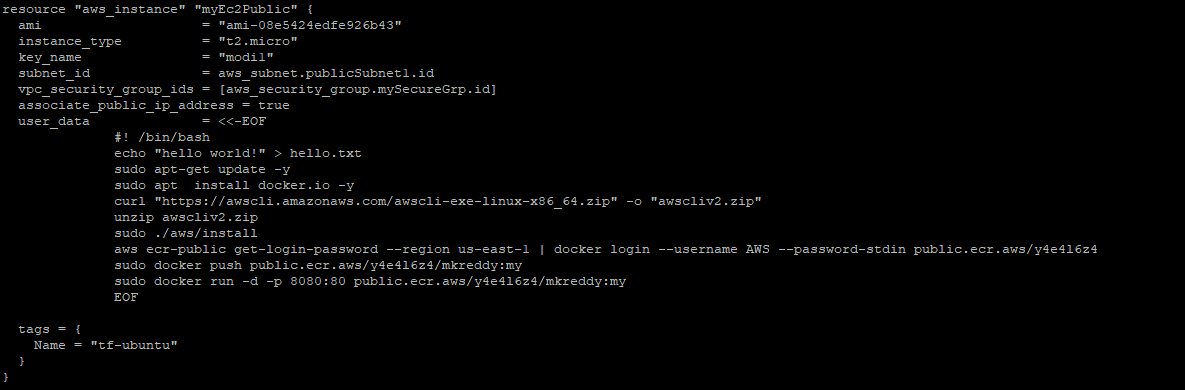


— now create all the resources whatever your mentioned terraform file.

Step3:

— Deploy Nginx Application as Docker Container using User Data script using terraform. Image should be pulled from ECR.

Userdate: user data refers to the script or configuration data that can be provided to an Amazon EC2 instance during its launch. It allows you to run commands or configure the instance automatically upon startup.



— the ec2-instance tag is ‘tf-ubuntu’ its creates the used data automatically upon startup.

— first it update the server.

— then install the docker in ec2-instance

— then install the aws cli its command line for aws console.

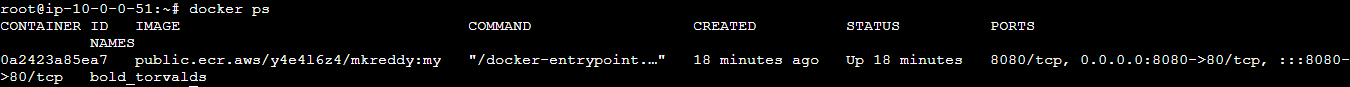
— after unzipping the aws cli archive file.

— after getting the authentication token from aws ECR repository. It's using docker login and connecting with aws cli.

— then pull the nginx image from ECR repository to local repository



— now run the nginx image to create a container with a certain port.



Step4: Now explode the nginx image using ‘tf-ubuntu’ instance public ip address and nginx image port number.

— <http://35.154.36.91:8080/>



Step5: Now whatever file I create is pushing and pulling nginx images to ECR . all files are pushed to github.

https://github.com/reddymodi/finalEvluation.git