TASK 3 : Infrastructure as Code (IaC) with Terraform

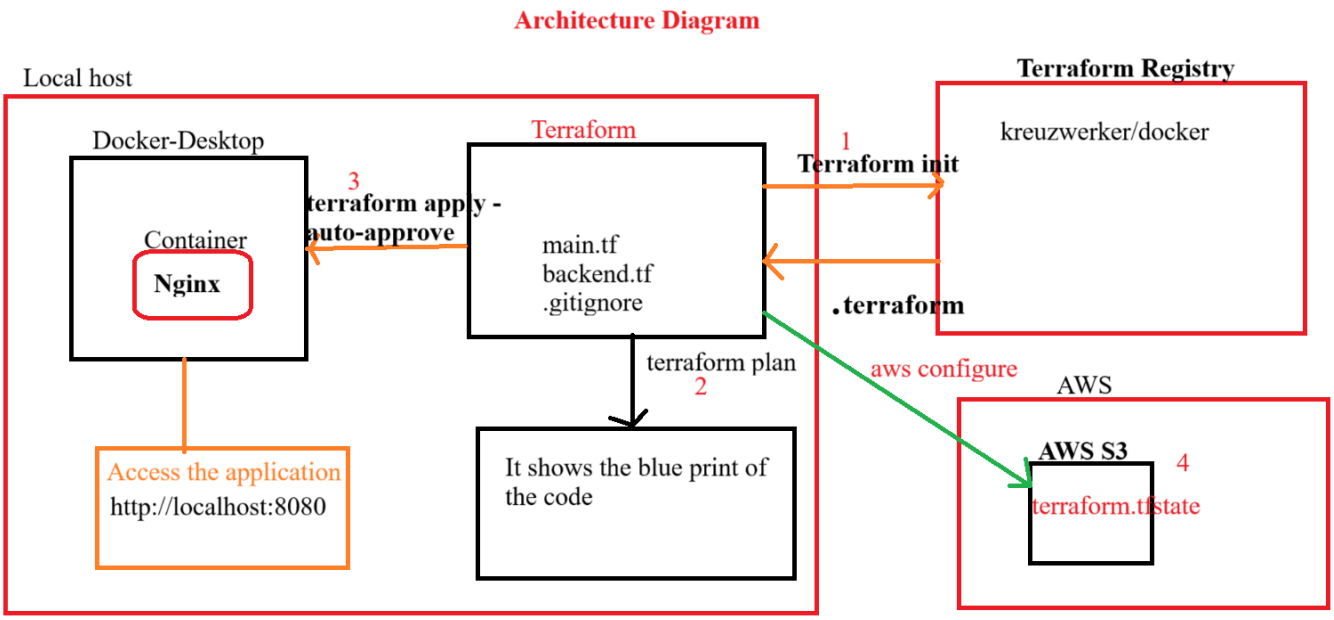
Objective

- Provision and manage a Docker container on your local system using Terraform

Tools Required:

- Terraform – to define and manage infrastructure using code.
- Docker – to run the containerized application locally.

Architecture Diagram



Step-by-step:

- Create a custom workspace in terraform
- Define the code first
  - Main.tf
  - Backend.tf
  - .gitignore

Main.tf

```
terraform {
 required_providers {
  docker = {
   source  = "kreuzwerker/docker"
   version = "~> 3.0.2"
  }
 }
}
```

```
provider "docker" {}

resource "docker_image" "nginx" {
  name = "nginx:latest"
}

resource "docker_container" "nginx" {
  name  = "my-nginx"
  image = docker_image.nginx.name
  ports {
    internal = 80
    external = 81
  }
}
```

Backend.tf

```
# Run first terraform-10-30am/day-4-resources_for_backend_s3_dynamodb to create reources
# This backend configuration instructs Terraform to store its state in an S3 bucket.
 terraform {
backend "s3" {
    bucket         = "bucket_name"  # Name of the S3 bucket where the state will be stored.
    region         =  "region_name"
    key            = "terraform.tfstate" # Path within the bucket where the state will be read/written.
    encrypt        = true  # Ensures the state is encrypted at rest in S3.
}
}
```

.gitignore

```
# Local .terraform directories
**/.terraform/*
# Terraform state files
*.tfstate
*.tfstate.*
```

Terraform lock file

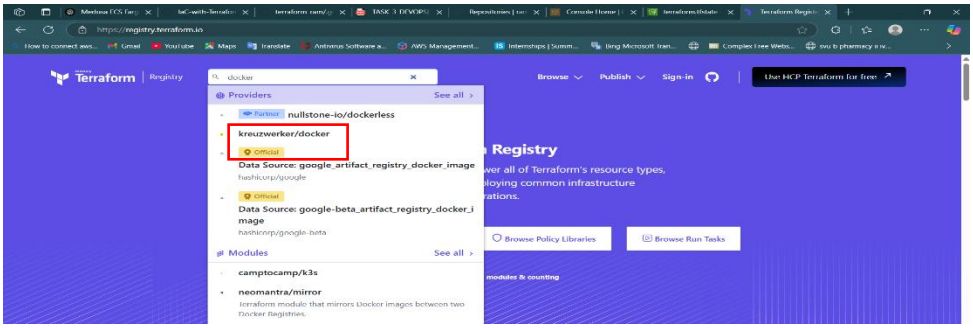
.terraform.lock.hcl

documents

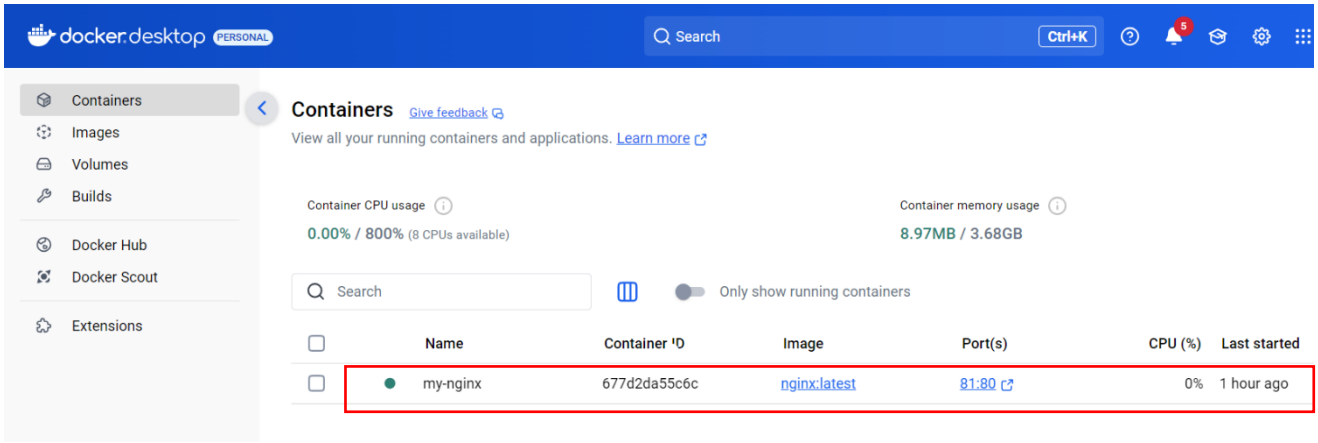
*.doc

```

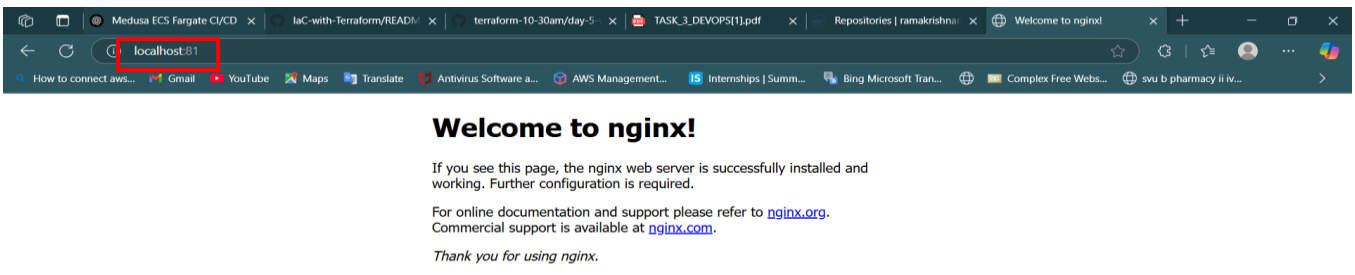
- Initialize Terraform where the main.tf file located ---> terraform init --------> .terraform (plugins)



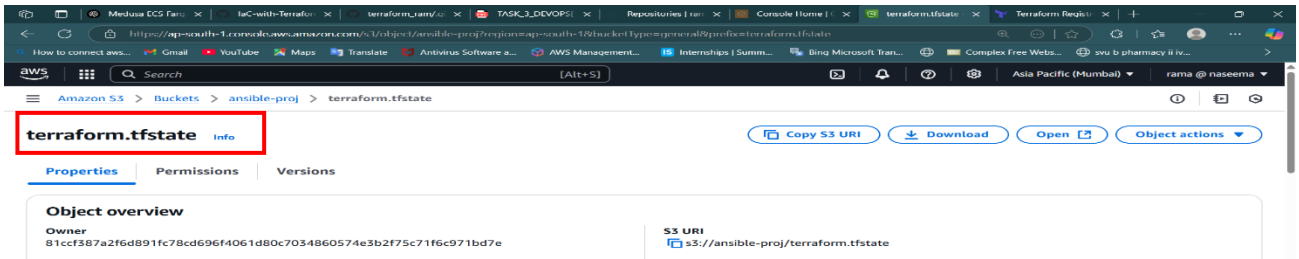

- Check the blue print of the code ----> terraform plan
- To actually create the Docker image and container --> terraform apply -auto-approve




- The container is running in localhost
- Access the application ------> http://localhost:8080




- Check the terraform.tfstate file in s3


```

- Check once plan and before destroy

```
        # (1 unchanged block hidden)
    }

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.
```

- Check list of the docker containers

```
ramakrishna@ASUS MINGW64 /d/IaC-with-Terraform
$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS          PORTS                  NAMES
677d2da55c6c   nginx:latest   "/docker-entrypoint.…"   12 minutes ago   Up 12 minutes   0.0.0.0:81->80/tcp   my-nginx
```

- Check the terraform state

```
ramakrishna@ASUS MINGW64 /d/IaC-with-Terraform
$ terraform state list
docker_container.nginx
docker_image.nginx
```

- Then do destroy the resources ----> terraform destroy -auto-approve

Benefits:

- Automate container management with Terraform