# AOS-Note Project-1

Name:[N Narendra]

Project:[Static Website-Docker]

Project: **Host Static Website on Docker Container thru Dockerfile and Push Docker Image to Docker Hub (Normal)**
[EC2 + Github + Docker]
Reference Videos: 1. https://youtu.be/uEfUxFnlxgM (for Normal)
2. https://youtu.be/eEU6gae494Y (thru Terraform)

Step 1: Create Public Repo in Docker Hub to store image & Github Repo for website files
Step 2: Create EC2 with 22,80 ports and install docker and create Dockerfile and run needed commands for httpd
Step 3: build the docker image from Dockerfile and push to docker hub
Step 4: Run the container from the above docker image which contains our needed website files
Step 5: check in browser for website with public IP of container running EC2

Note: Prepare all Terraform code for Infra setup and run Terraform commands to host website **`[thru Terraform]`**

**Task-1 (Git & Docker) on EC2**

```
1. sudo yum update -y
2. Sudo yum install git -y
3. sudo amazon-linux-extras install docker -y
4. sudo systemctl start docker, sudo systemctl enable docker, sudo systemctl status docker
5. sudo usermod -a -G docker ec2-user
6. /var/run/docker.sock
7. sudo vi Dockerfile                [ create Dockerfile in VS Code]
8. docker build -t . nn-techmax      [# to build image from Dockerfile]
9. docker login --username narian318, password:
10. docker tag nn-techmax narian318/nn-techmax
11. docker push narian318/nn-techmax
12. docker run -d --name nn-container -p 80:80 narian318/nn-techmax
```

`check the image in dockerhub and check website in browser with publicIP`

**Task-2 Thru Terraform**

```
1. Create build_image.sh file with commands inside
2. Create my_password.txt file on Desktop with our dockerhub password inside it
3. Create Dockerfile same as in Task-1
4. Create ec2.tf file [# with vpc, subnet, azone, ec2, SG script]
5. Run the Terraform commands in integrated Terminal in VS Code
```

`check the image in dockerhub and check website in browser`

**1. Dockerfile:**

```
FROM amazonlinux:latest

# Install dependencies
RUN yum update -y && \
yum install -y httpd && \
yum search wget && \
yum install wget -y && \
yum install unzip -y

RUN cd /var/www/html  [#change directory]
RUN wget https://github.com/azeezsalu/jupiter/archive/refs/heads/main.zip [# download webfiles]
RUN unzip main.zip    [#unzip folder]
RUN cp -r jupiter-main/* /var/www/html/     [#copy files into html directory]
RUN rm -rf jupiter-main main.zip     [# remove unwanted folder]
EXPOSE 80             [# exposes port 80 on the container]

# set the default application that will start when the container start
ENTRYPOINT ["/usr/sbin/httpd", "-D", "FOREGROUND"]
```

**2. build_image.sh :**

# create a repository to store the docker image in docker hub

# launch an ec2 instance. open port 80 and port 22

# install and configure docker on the ec2 instance
```
sudo yum update -y
sudo amazon-linux-extras install docker -y
sudo service docker start
sudo systemctl enable docker
sudo usermod -a -G docker ec2-user
docker info
```

# create a dockerfile
```
sudo vi Dockerfile
```

# build the docker image
```
docker build -t nn-techmax .
```

# login to your docker hub account
```
docker login --username narian318
```

# use the docker tag command to give the image a new name
```
docker tag techmax narian318/nn-techmax
```

# push the image to your docker hub repository
```
docker push narian318/nn-techmax
```

# start the container to test the image
```
docker run -dp 80:80 --name nn-container narian318/nn-techmax
```

# references
# https://docs.aws.amazon.com/AmazonECS/latest/developerguide/create-container-image.html
# https://docs.docker.com/get-started/02_our_app/

**3. ec2.tf :**

```
# configured aws provider with proper credentials
provider "aws" {
  region  = "us-east-1"
  profile = "nn-terraform"
}
# create default vpc if one does not exit
resource "aws_default_vpc" "default_vpc" {

  tags = {
    Name = "default vpc"
  }
}

# use data source to get all avalablility zones in region
data "aws_availability_zones" "available_zones" {}

# create default subnet if one does not exit
resource "aws_default_subnet" "default_az1" {
  availability_zone = data.aws_availability_zones.available_zones.names[0]

  tags = {
    Name = "default subnet"
  }
}
# create security group for the ec2 instance
resource "aws_security_group" "ec2_security_group" {
  name        = "docker server SG"
  description = "allow access on ports 80 and 22"
  vpc_id      = aws_default_vpc.default_vpc.id
  ingress {
    description = "http access"
    from_port   = 80
    to_port     = 80
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }
  ingress {
    description = "ssh access"
    from_port   = 22
    to_port     = 22
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }
  egress {
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }
  tags = {
    Name = "docker server sg"
  }
}
# use data source to get a registered amazon linux 2 ami
data "aws_ami" "amazon_linux_2" {
  most_recent = true
  owners      = ["amazon"]

  filter {
    name   = "owner-alias"
    values = ["amazon"]
  }

  filter {
    name   = "name"
    values = ["amzn2-ami-hvm*"]
  }
}

# launch the ec2 instance
resource "aws_instance" "ec2_instance" {
  ami                    = data.aws_ami.amazon_linux_2.id
  instance_type          = "t2.micro"
  subnet_id              = aws_default_subnet.default_az1.id
  vpc_security_group_ids = [aws_security_group.ec2_security_group.id]
  key_name               = "nar*****"

  tags = {
    Name = "docker server"
  }
}
# an empty resource block
resource "null_resource" "name" {

  # ssh into the ec2 instance
  connection {
    type        = "ssh"
    user        = "ec2-user"
    private_key = file("~/Downloads/nar****.pem")
    host        = aws_instance.ec2_instance.public_ip
  }
  # copy the password file for your docker hub account
  # from your computer to the ec2 instance
  provisioner "file" {
    source      = "~/Downloads/my-dhub-password.txt"
    destination = "/home/ec2-user/my-dhub-password.txt"
  }
  # copy the dockerfile from your computer to the ec2 instance
  provisioner "file" {
    source      = "Dockerfile"
    destination = "/home/ec2-user/Dockerfile"
  }
  # copy the build_image.sh from your computer to the ec2 instance
  provisioner "file" {
    source      = "techmax-docker-tf.sh"
    destination = "/home/ec2-user/techmax-docker-tf.sh"
  }
  # set permissions and run the build_image.sh file
  provisioner "remote-exec" {
    inline = [
      "sudo chmod +x /home/ec2-user/techmax-docker-tf.sh",
      "sh /home/ec2-user/techmax-docker-tf.sh",
    ]
  }
  # wait for ec2 to be created
  depends_on = [aws_instance.ec2_instance]

}

# print the url of the container
output "container_url" {
  value = join("", ["http://", aws_instance.ec2_instance.public_dns])
}
```