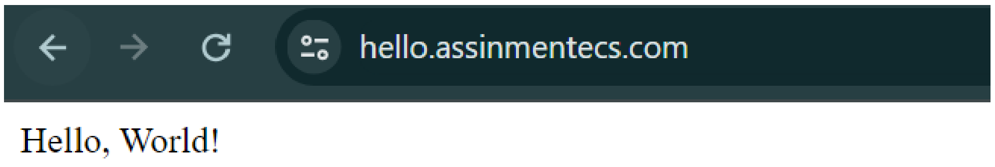
Problem - 5

Create an nginx container hosting any python/javascript application (any sample application) and deploy the container on an ECS cluster.

Part 1 - Create a Docker file configuring Nginx on a latest Ubuntu base image and bundle the application including a process manager (pm2/gunicorn) for the sample application.

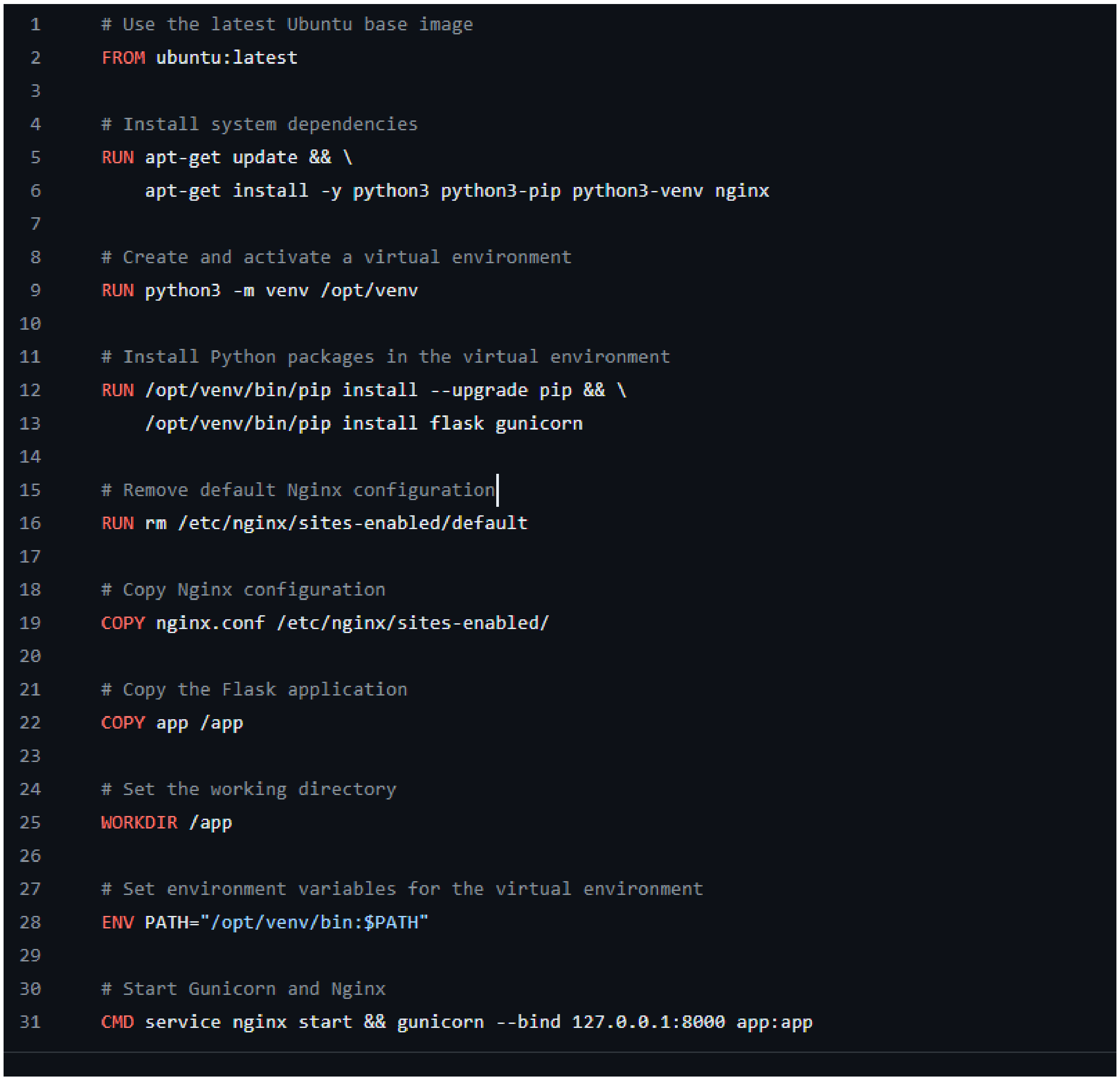
```
Sample Web app :- https://hello.assinmentecs.com

Git repo :- https://github.com/Thimira93/DevOps_task.git
```



hello.assinmentecs.com

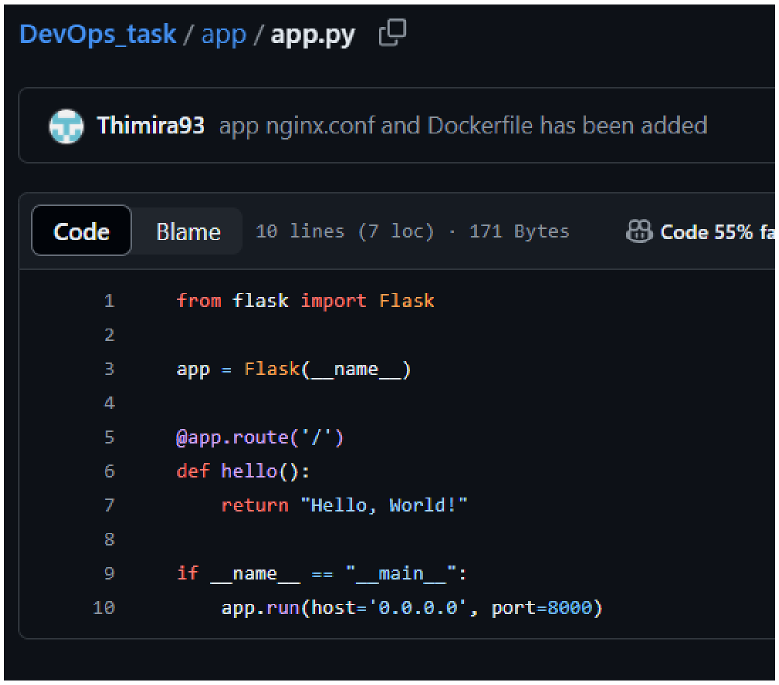
Hello, World!

Dockerfile



```dockerfile
# Use the latest Ubuntu base image
FROM ubuntu:latest

# Install system dependencies
RUN apt-get update && \
    apt-get install -y python3 python3-pip python3-venv nginx

# Create and activate a virtual environment
RUN python3 -m venv /opt/venv

# Install Python packages in the virtual environment
RUN /opt/venv/bin/pip install --upgrade pip && \
    /opt/venv/bin/pip install flask gunicorn

# Remove default Nginx configuration
RUN rm /etc/nginx/sites-enabled/default

# Copy Nginx configuration
COPY nginx.conf /etc/nginx/sites-enabled/

# Copy the Flask application
COPY app /app

# Set the working directory
WORKDIR /app

# Set environment variables for the virtual environment
ENV PATH="/opt/venv/bin:$PATH"

# Start Gunicorn and Nginx
CMD service nginx start && gunicorn --bind 127.0.0.1:8000 app:app
```

Python file

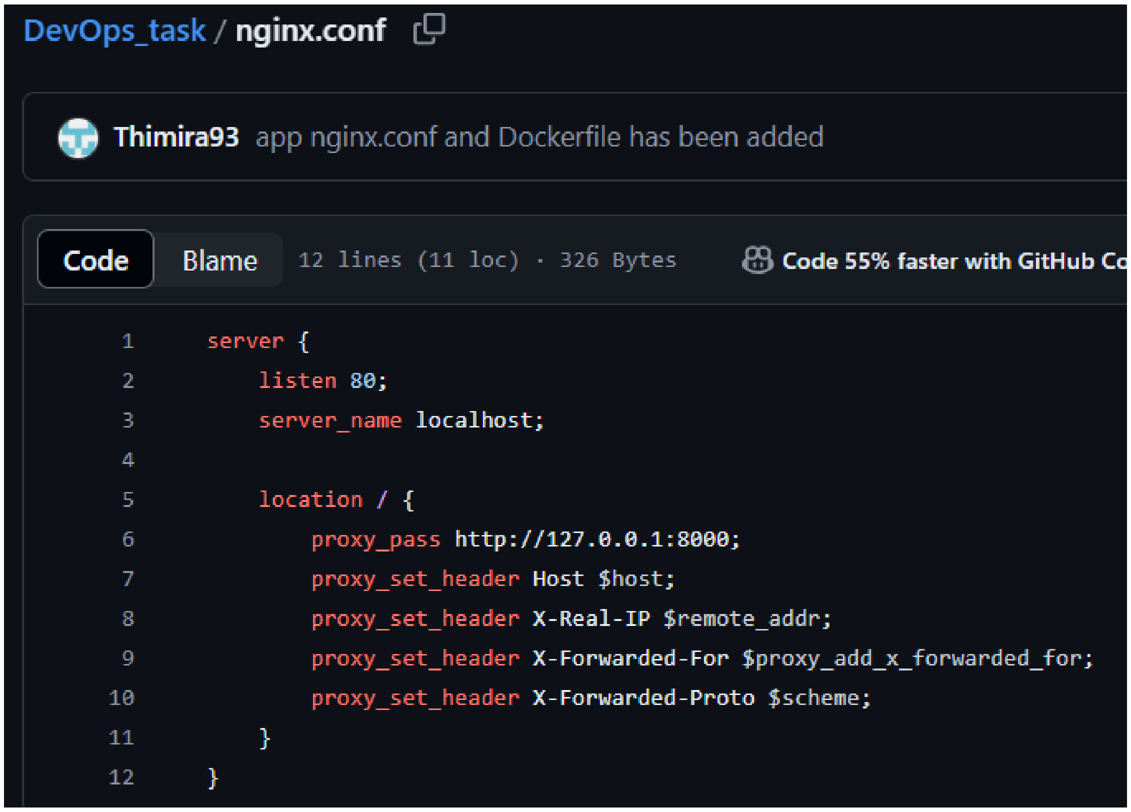

DevOps_task / app / app.py

Thimira93 app nginx.conf and Dockerfile has been added

Code Blame 10 lines (7 loc) · 171 Bytes Code 55% fa

```python
from flask import Flask

app = Flask(__name__)

@app.route('/')
def hello():
    return "Hello, World!"

if __name__ == "__main__":
    app.run(host='0.0.0.0', port=8000)
```

Nginx.conf file


DevOps_task / nginx.conf

Thimira93 app nginx.conf and Dockerfile has been added

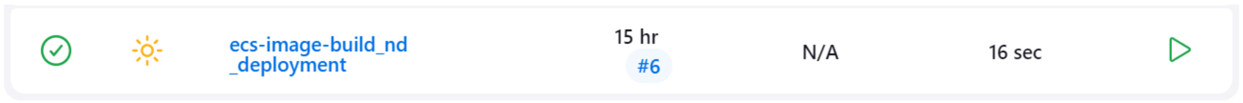
Code Blame 12 lines (11 loc) · 326 Bytes Code 55% faster with GitHub Co

```nginx
server {
    listen 80;
    server_name localhost;

    location / {
        proxy_pass http://127.0.0.1:8000;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
}
```

Part 2 - Create a Jenkins based CI pipeline to build the image and push it to the ECR repository.

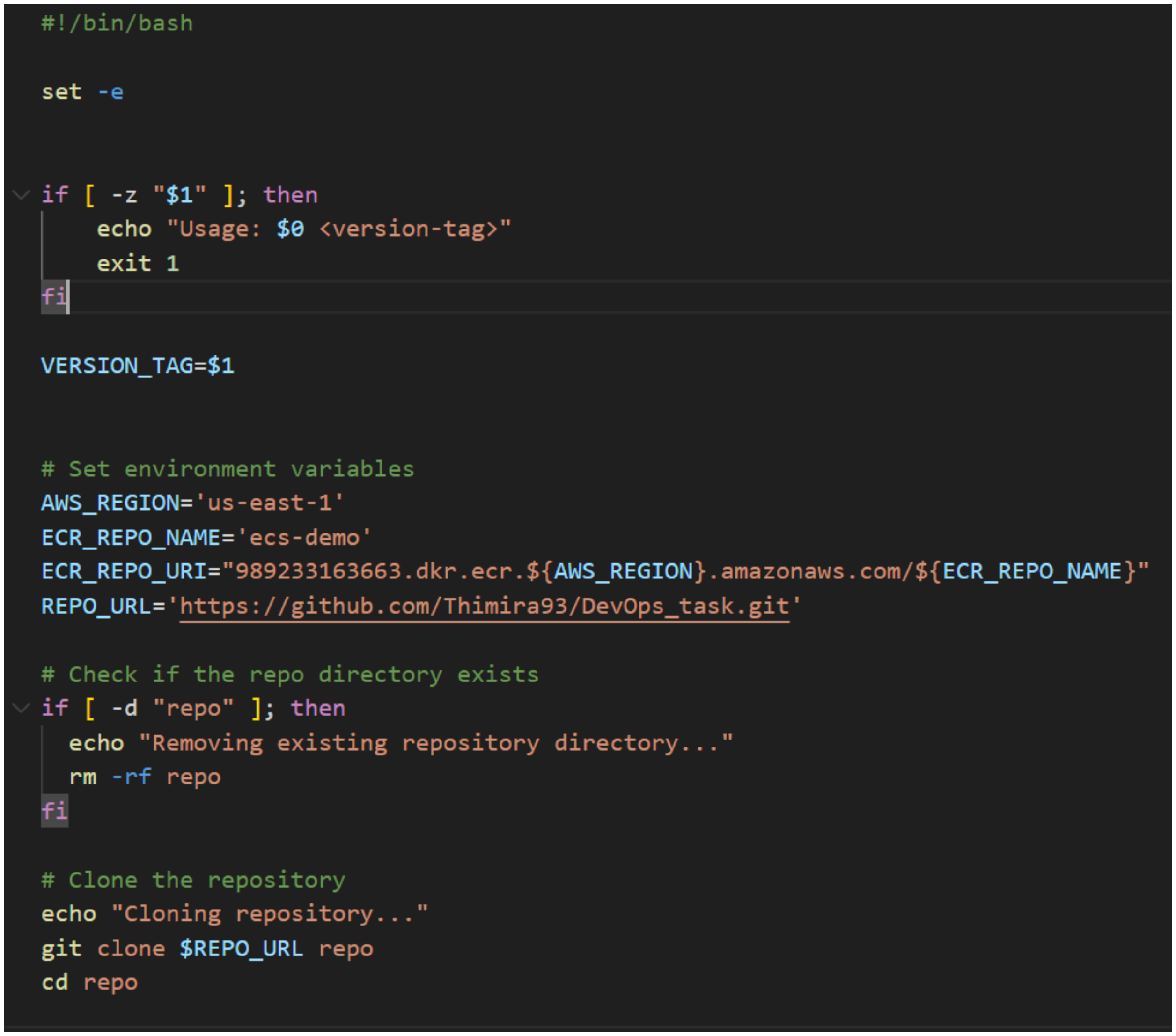
```
Jenkins server :- https://jenkins.assinmentecs.com
```

Jenkins pipeline



✓	☀	ecs-image-build_nd_deployment	15 hr #6	N/A	16 sec	▷

Bash script file (This script is run by Jenkins pipeline to clone the repo and push image to ecr with a tag)



```bash
#!/bin/bash

set -e


if [ -z "$1" ]; then
    echo "Usage: $0 <version-tag>"
    exit 1
fi

VERSION_TAG=$1


# Set environment variables
AWS_REGION='us-east-1'
ECR_REPO_NAME='ecs-demo'
ECR_REPO_URI="989233163663.dkr.ecr.${AWS_REGION}.amazonaws.com/${ECR_REPO_NAME}"
REPO_URL='https://github.com/Thimira93/DevOps_task.git'

# Check if the repo directory exists
if [ -d "repo" ]; then
  echo "Removing existing repository directory..."
  rm -rf repo
fi

# Clone the repository
echo "Cloning repository..."
git clone $REPO_URL repo
cd repo
```

```
# Build Docker image
echo "Building Docker image..."
docker build -t ${ECR_REPO_URI}:${VERSION_TAG} .

# Login to AWS ECR
echo "Logging in to AWS ECR..."
aws ecr get-login-password --region $AWS_REGION | docker login --username AWS --password-stdin $ECR_REPO_URI

# Push Docker image to ECR
echo "Pushing Docker image to ECR..."
docker push ${ECR_REPO_URI}:${VERSION_TAG}

echo "Done."
```

Part 3 - Provision the Terraform infrastructure necessary to deploy the container to a ECS Cluster. Use the following configuration as the basis for the infrastructure. Create any other resources as required. Utilise a S3 backend with DynamoDB as state locking mechanism.

● VPC – 10.0.0.0/16

● Subnets

○ EC2 Private 1 – 10.0.10.0/24

○ EC2 Private 2 – 10.0.11.0/24

○ ELB Public 1 – 10.0.20.0/24

○ ELB Public 2 – 10.0.21.0/24

● Route Table

○ EC2 Private RT

○ ELB Public RT

● NACL

○ EC2 Private NACL

○ ELB Public NACL

●

● ALB

○ Listener Rules

■ HTTP redirected to HTTPS.

■ HTTPS pointed to the ECS.

● ECS Cluster

○ Capacity Provider - EC2

● WAF

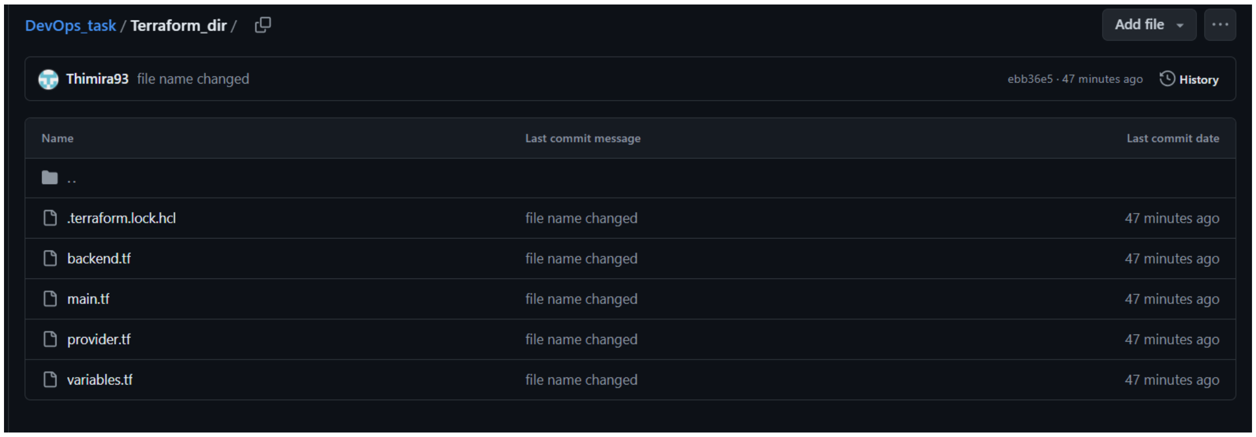
○ Rule sets

■ AWSManagedRulesCommonRuleSet

■ AWSManagedRulesAmazonIpReputationList

● IAM roles/policies

Note - Provision any other resource that may be required to facilitate a web application.



DevOps_task / Terraform_dir /

Add file

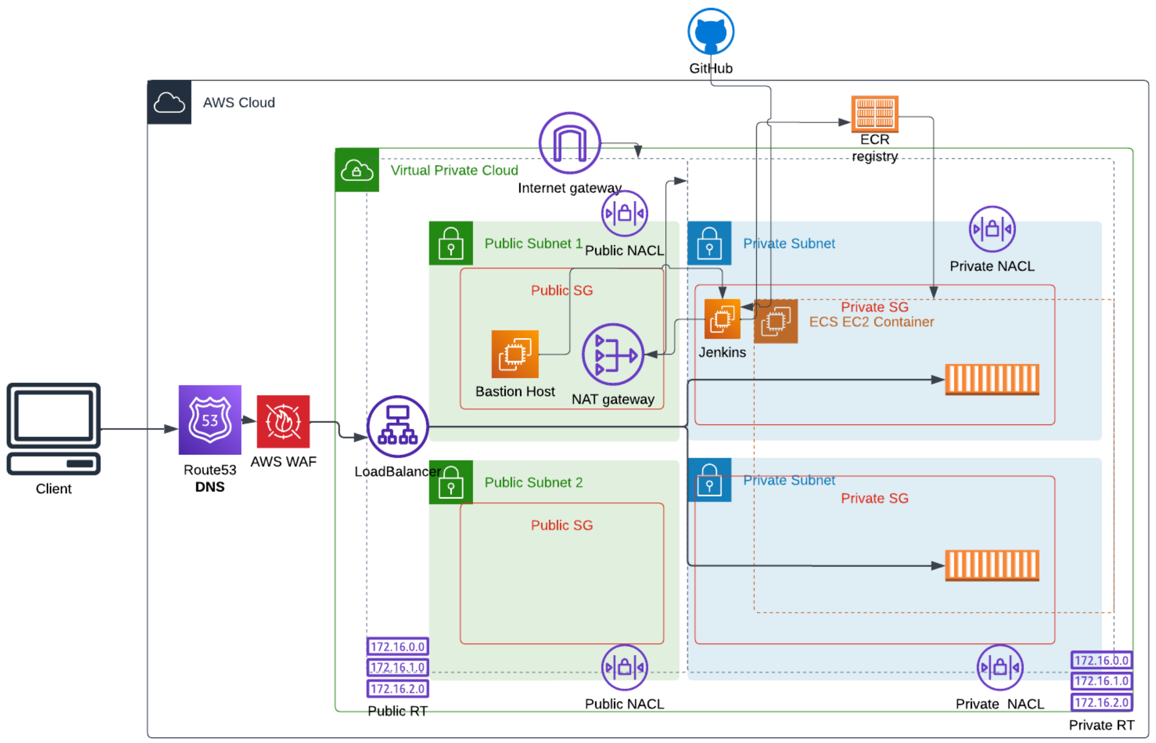
Thimira93 file name changed

ebb36e5 · 47 minutes ago

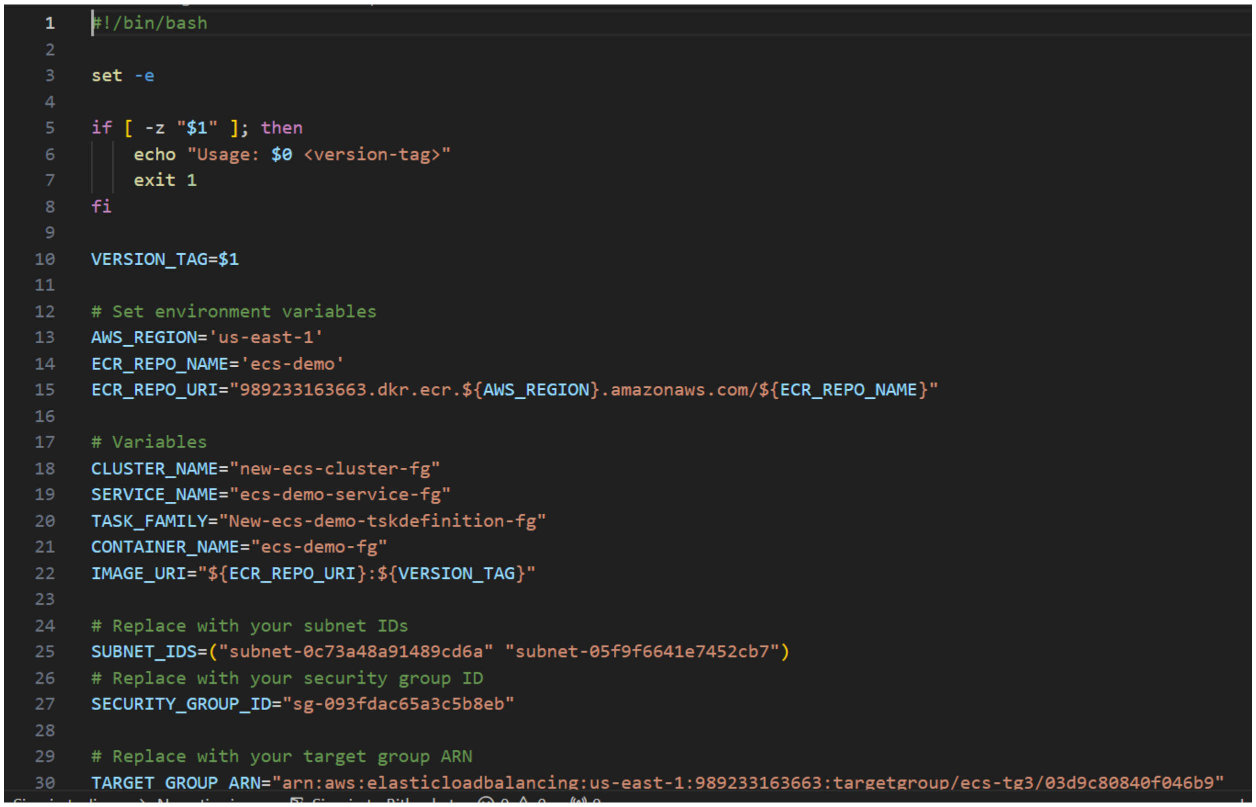
History

Name	Last commit message	Last commit date
..		
.terraform.lock.hcl	file name changed	47 minutes ago
backend.tf	file name changed	47 minutes ago
main.tf	file name changed	47 minutes ago
provider.tf	file name changed	47 minutes ago
variables.tf	file name changed	47 minutes ago

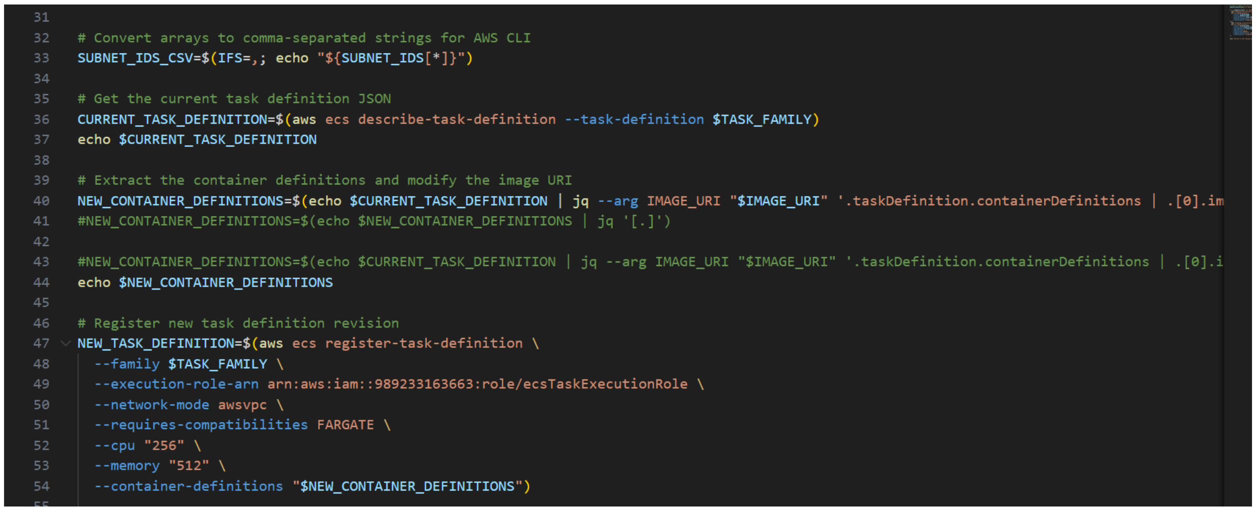
Part 4 - Create an Infrastructure diagram for the provisioned resources.



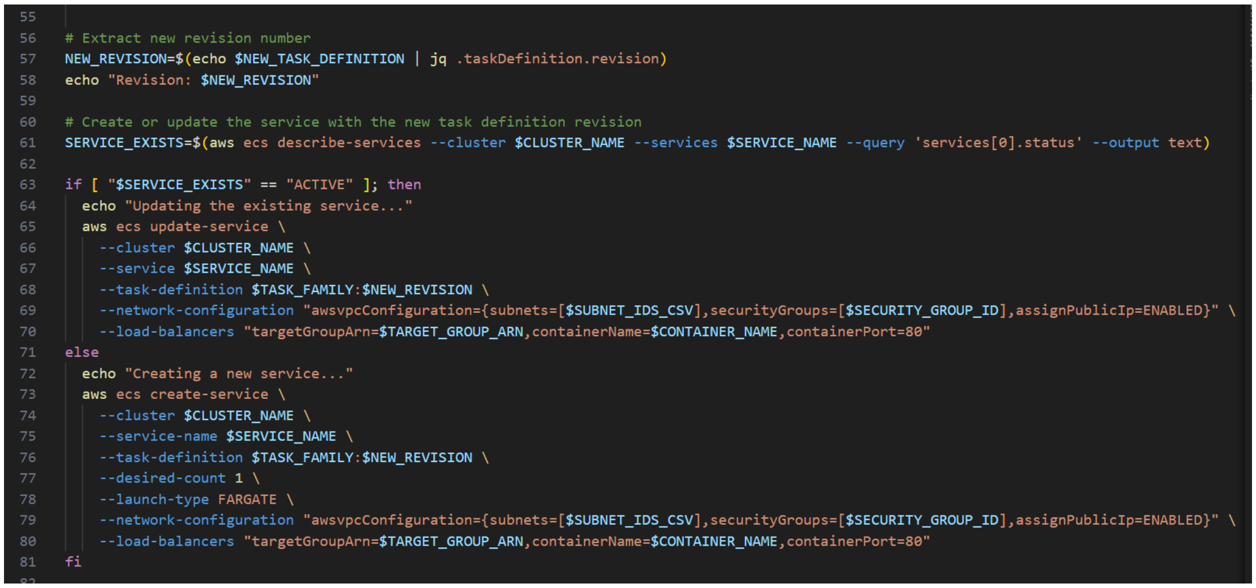
Part 5 - Create a Jenkins based CD pipeline to deploy the container to the ECS cluster



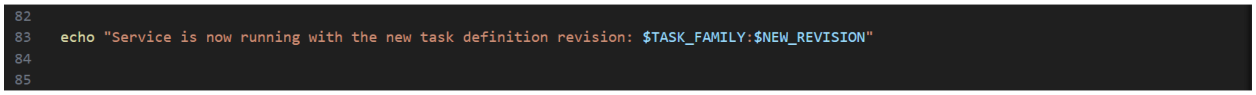
```bash
#!/bin/bash

set -e

if [ -z "$1" ]; then
    echo "Usage: $0 <version-tag>"
    exit 1
fi

VERSION_TAG=$1

# Set environment variables
AWS_REGION='us-east-1'
ECR_REPO_NAME='ecs-demo'
ECR_REPO_URI="989233163663.dkr.ecr.${AWS_REGION}.amazonaws.com/${ECR_REPO_NAME}"

# Variables
CLUSTER_NAME="new-ecs-cluster-fg"
SERVICE_NAME="ecs-demo-service-fg"
TASK_FAMILY="New-ecs-demo-tskdefinition-fg"
CONTAINER_NAME="ecs-demo-fg"
IMAGE_URI="${ECR_REPO_URI}:${VERSION_TAG}"

# Replace with your subnet IDs
SUBNET_IDS=("subnet-0c73a48a91489cd6a" "subnet-05f9f6641e7452cb7")
# Replace with your security group ID
SECURITY_GROUP_ID="sg-093fdac65a3c5b8eb"

# Replace with your target group ARN
TARGET_GROUP_ARN="arn:aws:elasticloadbalancing:us-east-1:989233163663:targetgroup/ecs-tg3/03d9c80840f046b9"
```



```bash

# Convert arrays to comma-separated strings for AWS CLI
SUBNET_IDS_CSV=$(IFS=,; echo "${SUBNET_IDS[*]}")

# Get the current task definition JSON
CURRENT_TASK_DEFINITION=$(aws ecs describe-task-definition --task-definition $TASK_FAMILY)
echo $CURRENT_TASK_DEFINITION

# Extract the container definitions and modify the image URI
NEW_CONTAINER_DEFINITIONS=$(echo $CURRENT_TASK_DEFINITION | jq --arg IMAGE_URI "$IMAGE_URI" '.taskDefinition.containerDefinitions | .[0].im
#NEW_CONTAINER_DEFINITIONS=$(echo $NEW_CONTAINER_DEFINITIONS | jq '[.]')

#NEW_CONTAINER_DEFINITIONS=$(echo $CURRENT_TASK_DEFINITION | jq --arg IMAGE_URI "$IMAGE_URI" '.taskDefinition.containerDefinitions | .[0].i
echo $NEW_CONTAINER_DEFINITIONS

# Register new task definition revision
NEW_TASK_DEFINITION=$(aws ecs register-task-definition \
  --family $TASK_FAMILY \
  --execution-role-arn arn:aws:iam::989233163663:role/ecsTaskExecutionRole \
  --network-mode awsvpc \
  --requires-compatibilities FARGATE \
  --cpu "256" \
  --memory "512" \
  --container-definitions "$NEW_CONTAINER_DEFINITIONS")
```



```bash

# Extract new revision number
NEW_REVISION=$(echo $NEW_TASK_DEFINITION | jq .taskDefinition.revision)
echo "Revision: $NEW_REVISION"

# Create or update the service with the new task definition revision
SERVICE_EXISTS=$(aws ecs describe-services --cluster $CLUSTER_NAME --services $SERVICE_NAME --query 'services[0].status' --output text)

if [ "$SERVICE_EXISTS" == "ACTIVE" ]; then
  echo "Updating the existing service..."
  aws ecs update-service \
    --cluster $CLUSTER_NAME \
    --service $SERVICE_NAME \
    --task-definition $TASK_FAMILY:$NEW_REVISION \
    --network-configuration "awsvpcConfiguration={subnets=[$SUBNET_IDS_CSV],securityGroups=[$SECURITY_GROUP_ID],assignPublicIp=ENABLED}" \
    --load-balancers "targetGroupArn=$TARGET_GROUP_ARN,containerName=$CONTAINER_NAME,containerPort=80"
else
  echo "Creating a new service..."
  aws ecs create-service \
    --cluster $CLUSTER_NAME \
    --service-name $SERVICE_NAME \
    --task-definition $TASK_FAMILY:$NEW_REVISION \
    --desired-count 1 \
    --launch-type FARGATE \
    --network-configuration "awsvpcConfiguration={subnets=[$SUBNET_IDS_CSV],securityGroups=[$SECURITY_GROUP_ID],assignPublicIp=ENABLED}" \
    --load-balancers "targetGroupArn=$TARGET_GROUP_ARN,containerName=$CONTAINER_NAME,containerPort=80"
fi
```



```bash

echo "Service is now running with the new task definition revision: $TASK_FAMILY:$NEW_REVISION"


```