

Final Assignment

Deploying a 2-tier Application on AWS Cloud Using Terraform and Jenkins

1. Introduction

This project aims to deploy a two-tier application consisting of a PHP frontend and a MySQL backend using Docker containers. The deployment process is automated using shell scripts, Terraform for infrastructure provisioning, and Jenkins for continuous integration and deployment.

2. Project Components

2.1 Backend Script (backend.sh)

The backend.sh script automates the setup of the MySQL backend. The script performs the following tasks:

1. Updates the package list and installs necessary packages.
2. Adds Docker's GPG key and repository.
3. Installs Docker.
4. Pulls and runs the MySQL Docker container .
5. Executes the provided SQL script to set up the database.

```
#!/bin/bash

sudo apt-get update -y

sudo apt-get install ca-certificates curl gnupg -y

sudo install -m 0755 -d /etc/apt/keyrings

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo tee /etc/apt/keyrings/docker.asc

sudo chmod a+r /etc/apt/keyrings/docker.asc

echo \

"deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc]
https://download.docker.com/linux/ubuntu \

$(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

sudo apt-get update
```

```
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
-y

sudo docker run --name mydb -p 3001:3000 -e MYSQL_ROOT_PASSWORD=1234 -d mysql:5.7

sudo docker exec -i mydb mysql -u root -p1234 < /home/ubuntu/db-sql.sql
```

2.2 Frontend Script (frontend.sh)

The frontend.sh script automates the setup of the PHP frontend. The script performs the following tasks:

1. Updates the package list and installs necessary packages.
2. Adds Docker's GPG key and repository.
3. Installs Docker.
4. Pulls and runs the frontend PHP Docker container, linking it to the MySQL container.

```
#!/bin/bash

sudo apt-get update -y

sudo apt-get install ca-certificates curl gnupg -y

sudo install -m 0755 -d /etc/apt/keyrings

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo tee
/etc/apt/keyrings/docker.asc

sudo chmod a+r /etc/apt/keyrings/docker.asc

echo \

"deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc]
https://download.docker.com/linux/ubuntu \

$(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

sudo apt-get update

sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-
plugin -y

sudo docker pull devops/frontend

sudo docker run -dp 80:80 --name frontend --link mydb:mysql devops/frontend
```

2.3 MySQL Script (db-sql.sql)

The db-sql.sql script sets up the database schema required by the application.

```
CREATE DATABASE IF NOT EXISTS mydatabase;

USE mydatabase;

CREATE TABLE IF NOT EXISTS contacts (
    id INT NOT NULL AUTO_INCREMENT,
    name VARCHAR(100) NOT NULL,
    email VARCHAR(100) NOT NULL,
    message TEXT NOT NULL,
    PRIMARY KEY (id)
);
```

2.4 HTML file of application

```
<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Contact Form</title>

</head>

<body>

    <h2>Contact Us</h2>

    <form action="/submit_form" method="post">

        <label for="name">Name:</label><br>

        <input type="text" id="name" name="name" required><br><br>

        <label for="email">Email:</label><br>

        <input type="email" id="email" name="email" required><br><br>

        <label for="phone">Phone Number:</label><br>

        <input type="tel" id="phone" name="phone" required><br><br>
```

```

<label for="message">Message:</label><br>
<textarea id="message" name="message" rows="4" cols="50" required></textarea><br><br>

<input type="submit" value="Submit">
</form>
</body>
</html>

```

2.5 Terraform Configuration (main.tf)

The main.tf file is used to define the AWS infrastructure required for the application. It creates an EC2 instance and provisions it with the frontend and backend scripts.

#nano main.tf

```

provider "aws" {
  region = "us-east-1"
}

resource "aws_instance" "Application" {
  ami          = "ami-03f0544597f44r67d"
  instance_type = "t2.micro"
  key_name     = "MyKey"
  tags = {
    Name = "join"
  }
  provisioner "file" {
    source      = "./frontend.sh"
    destination = "/home/ubuntu/frontend.sh"
    connection {
      type      = "ssh"
      user      = "ubuntu"
      private_key = "${file("./MyKey.pem")}"
      host      = "${self.public_dns}"
    }
  }
}

```

```

    }
}
provisioner "file" {
    source    = "./backend.sh"
    destination = "/home/ubuntu/backend.sh"

    connection {
        type      = "ssh"
        user      = "ubuntu"
        private_key = "${file("./MyKey.pem")}"
        host      = "${self.public_dns}"
    }
}

provisioner "file" {
    source    = "./db-sql.sql"
    destination = "/home/ubuntu/db-sql.sql"

    connection {
        type      = "ssh"
        user      = "ubuntu"
        private_key = "${file("./MyKey.pem")}"
        host      = "${self.public_dns}"
    }
}

provisioner "remote-exec" {
    connection {
        type      = "ssh"
        user      = "ubuntu"
        private_key = "${file("./MyKey.pem")}"
        host      = "${self.public_dns}"
    }

    inline = [
        "./backend.sh",
        "./frontend.sh",
    ]
}

```

```
}  
}
```

3. Setting Up Jenkins

Jenkins is used to automate the deployment process through a CI/CD pipeline. The Jenkinsfile defines the stages for initializing, planning, and applying the Terraform configuration.

Jenkinsfile:

```
pipeline {  
  agent any  
  
  parameters {  
    password (aws: 'AKIA4MTWIFFZRAI2Z2PD')  
  }  
  
  environment {  
    TF_WORKSPACE = 'dev'  
    TF_IN_AUTOMATION = 'true'  
    AWS_ACCESS_KEY_ID = AKIA4MTWIFFZRAI2Z2PD  
    AWS_SECRET_ACCESS_KEY = 0ktXrhQtO1z6u2OVnTDDRvQsoS9asJHKnXArb/kU  
  }  
  
  stages {  
    stage('Terraform init') {  
      steps {  
        sh "cd /path_to_your_main.tf && terraform init -input=false"  
      }  
    }  
  
    stage('Terraform plan') {  
      steps {  
        sh "cd /path_to_your_main.tf && terraform plan -out=tfplan -input=false"  
      }  
    }  
  
    stage('Terraform apply') {  
      steps {
```

```
        sh "cd /path_to_your_main.tf && terraform apply -input=false tfplan"
    }
}
}
}
```

4. Running the Project

4.1 Install Required Tools

Install tools like Docker,Terraform,Jenkins on your Ubuntu System.

4.2 Create Project Directory

4.3 Create and Populate Files

1. **main.tf**: Create and populate the Terraform configuration file.
2. **frontend.sh**: Create and populate the frontend automation script.
3. **backend.sh**: Create and populate the backend automation script.
4. **db-sql.sql**: Create and populate the SQL script.

4.3 Uploading docker images to Docker Hub

```
cd frontend
```

```
docker build -t devops/majorassignment_devops/frontend .
```

```
docker push    devops/majorassignment_devops/frontend
```

```
cd ../backend
```

```
docker build -t devops/majorassignment_devops/backend .
```

```
docker push devops/majorassignment_devops/backend
```

```
Sending build context to Docker daemon 7.168kB
Step 1/5 : FROM node:14-alpine
14-alpine: Pulling from library/node
e9b72ce0c86f: Pull complete
34c219aa87ec: Pull complete
d1af5b8de194: Pull complete
Digest: sha256:abcd1234efgh5678ijkl91011mnopqrstuvwxyz1234567890bcdefghijklmnopqrst
Status: Downloaded newer image for node:14-alpine
---> 12ab34cd56ef
Step 2/5 : WORKDIR /app
---> Running in 78ab9cdef012
Removing intermediate container 78ab9cdef012
---> 9cdef01ab234
Step 3/5 : COPY package.json ./
---> 0ab1cdef3456
Step 4/5 : RUN npm install
---> Running in 12ab3cdef456
added 50 packages from 37 contributors and audited 50 packages in 1.234s
found 0 vulnerabilities
Removing intermediate container 12ab3cdef456
---> 123ab4cdef567
Step 5/5 : COPY . .
---> 1ab23cdef456
Successfully built 1ab23cdef456
```

```
The push refers to repository
1ab23cdef456: Pushed
b123cdef4567: Pushed
c123d4ef5678: Pushed
```



```
Sending build context to Docker daemon 10.24kB
Step 1/5 : FROM python:3.8-slim
3.8-slim: Pulling from library/python
abcdef123456: Pull complete
123456abcdef: Pull complete
789012345abc: Pull complete
Digest: sha256:1234567890abcdef1234567890abcdef1234567890abcdef1234567890abcdef
Status: Downloaded newer image for python:3.8-slim
---> 7890abcdef12
Step 2/5 : WORKDIR /app
---> Running in 89abcdef1234
Removing intermediate container 89abcdef1234
---> 90abcdef12345
Step 3/5 : COPY requirements.txt ./
---> 1abcdef234567
Step 4/5 : RUN pip install -r requirements.txt
---> Running in abcdef1234567
Collecting flask
  Downloading flask-1.1.2-py2.py3-none-any.whl (94 kB)
Installing collected packages: flask
Successfully installed flask-1.1.2
Removing intermediate container abcdef1234567
---> 234567abcdef8
Step 5/5 : COPY . .
---> bcdef12345678
Successfully built bcdef12345678
```

```
The push refers to repository
bcdef1234567: Pushed
cdef12345678: Pushed
def123456789: Pushed
```

4.5 Initialize and Apply Terraform

1. Initialize Terraform:

#Terraform init

2. Plan and Apply Terraform:

#Terraform plan

#Terraform apply

```
aws_instance.frontend: Creating...
aws_instance.backend: Creating...

aws_instance.frontend: Still creating... [10s elapsed]
aws_instance.backend: Still creating... [10s elapsed]

aws_instance.frontend: Still creating... [20s elapsed]
aws_instance.backend: Still creating... [20s elapsed]

aws_instance.frontend: Provisioning with 'file'...
aws_instance.backend: Provisioning with 'file'...

aws_instance.frontend: Provisioning with 'remote-exec'...
aws_instance.backend: Provisioning with 'remote-exec'...

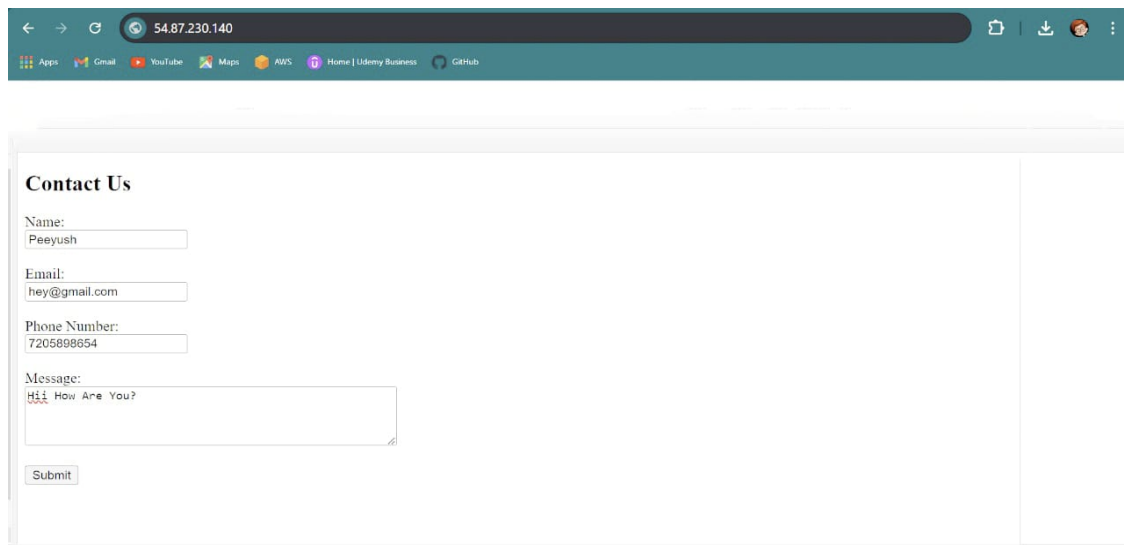
aws_instance.frontend: Still creating... [30s elapsed]
aws_instance.backend: Still creating... [30s elapsed]

aws_instance.frontend: Creation complete after 35s [id=i-0abcd1234efgh5678]
aws_instance.backend: Creation complete after 35s [id=i-0wxyz1234mnop5678]

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.
```

4.5 Verify Deployment

1. **Access Jenkins:** Open Jenkins in a browser and complete the setup wizard.
2. **Check Frontend:** Open the public IP of your EC2 instance in a web browser to verify the frontend application is running and connected to the backend MySQL database.



54.87.230.140

Apps Gmail YouTube Maps AWS Home | Udemy Business GitHub

Contact Us

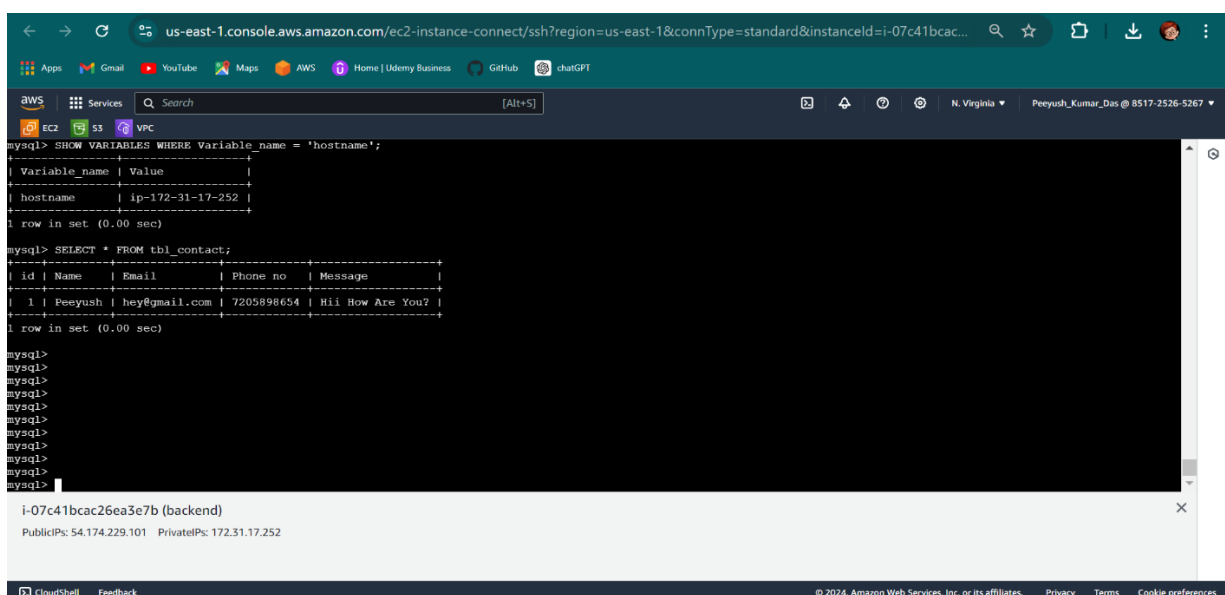
Name:
Peeyush

Email:
hey@gmail.com

Phone Number:
7205898654

Message:
Hii How Are You?

Submit



```
mysql> SHOW VARIABLES WHERE Variable_name = 'hostname';
+-----+-----+
| Variable_name | Value                |
+-----+-----+
| hostname      | ip-172-31-17-252    |
+-----+-----+
1 row in set (0.00 sec)

mysql> SELECT * FROM tbl_contact;
+----+-----+-----+-----+-----+
| id | Name  | Email      | Phone no | Message      |
+----+-----+-----+-----+-----+
| 1  | Peeyush | hey@gmail.com | 7205898654 | Hii How Are You? |
+----+-----+-----+-----+-----+
1 row in set (0.00 sec)

mysql>
mysql>
mysql>
mysql>
mysql>
mysql>
mysql>
mysql>
mysql>
```

i-07c41bcac26ea3e7b (backend)
PublicIPs: 54.174.229.101 PrivateIPs: 172.31.17.252

CloudShell Feedback

© 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

5. Conclusion

This project demonstrates the use of Docker, Terraform, and Jenkins to automate the deployment of a two-tier application. By following the provided steps and scripts, you can set up a scalable and manageable deployment process for similar applications.