DEVOPS PROJECT REPORT - 2025

Deploying a 2-tier Application on AWS Cloud Using Terraform

Submitted By -

Name: DIPTESH SINGH

Brannch: CSE

Registration Number: 2241019485

1. Introduction

In today's evolving software development landscape, the integration of automation, cloud infrastructure, and containerization has become essential for scalable, reliable, and maintainable deployments. This project, undertaken as part of a one-month DevOps training program, aims to build and deploy a containerized 2-tier application using a combination of **Terraform, Docker, Bash scripting, and AWS Cloud Infrastructure**. The project follows a structured, multi-stage automation workflow that emphasizes Infrastructure as Code (IaC), secure network segmentation, and inter-container communication across separate EC2 instances.

2. Project Statement

The objective of this project is to design, deploy, and test a 2-tier web application in a cloud environment using DevOps principles. The application comprises a frontend interface accessible to users and a backend service connected to a PostgreSQL database. The project involves:

- Automating infrastructure creation on AWS using Terraform.
- Running Docker containers on separate EC2 instances placed in public and private subnets.
- Verifying secure communication between the frontend and backend across different network layers.
- Performing automated provisioning and manual testing to validate end-to-end functionality.

3. Project Overview

The project is divided into three major stages:

- **Stage 1: Create_Infra**
  Terraform is used to provision a custom VPC with public and private subnets. A frontend EC2 instance is launched in the public subnet, and a backend EC2 instance is launched in the private subnet. Terraform provisioners are used to transfer and prepare the necessary setup scripts (`frontend.sh` and `backend.sh`) to each instance.
- **Stage 2: Deploy_Apps**
  Docker is installed on both instances, and the application containers are pulled from Docker Hub and executed. The frontend container runs a static web interface allowing user input, while the backend container hosts an API server connected to a PostgreSQL database, handling data persistence and logic.
- **Stage 3: Test_Solution**
  Outputs from Terraform are used to access the frontend application via a public IP or DNS. The application is tested manually by submitting feedback through the frontend and verifying its presence in the backend database. Additional testing is done using `curl` and Docker commands to confirm container health and service availability.

This project demonstrates a practical application of DevOps methodologies, enabling end-to-end automation of infrastructure deployment, application provisioning, and system validation in a cloud-native environment.

Stage 1: Create_Infra & Build 2-Tier App

1.1 Objective

- **Infra:** Use Terraform to create a Public VPC and 2 subnets in it, Namely PUBLIC and PRIVATE subnet. Add all other AWS services in such a way that resources in the PUBLIC subnet are accessible through routes and the PRIVATE subnet resources are restricted.

  - Inside the PUBLIC subnet launch and instance called FRONTEND. In the PRIVATE subnet launch another instance called BACKEND.

  - Test if the instances can communicate with each other (Although BACKEND is in private subnet, instances within a VPC are able to communicate). Using Terraform Provisioner send a script named frontend.sh to FRONTEND and backend.sh to BACKEND.

- **App:** Create a 2-tier application (preferably on git) using any language and tools, run and test the application.

  - The application must have a frontend and a database connected to it in the backend. It must allow the user to enter some details in the frontend and store the same in a row in the database.

  - Containerize the application in such a way that the frontend and the backend can be connected on different systems (test this using ec2 instances first then containerize).Upload the application to DockerHub and save the pull request command.

1.2 Application Development & Containerization

```
diptesh-singh@ASUS-TUF:~/Desktop/Diptesh_Singh_Devops/Project/2-tier-app$ ls
backend  db  docker-compose.yml  frontend  pull-commands.txt
```

1.2.1 Frontend

- **Tech:** HTML, CSS, Javascript
- **Function:** Collects `name` + `feedback`, calls `POST /feedback` and lists via `GET /feedback`.

  frontend/index.html →

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>Feedback App</title>
  <link rel="stylesheet" href="style.css" />
  </head>
<body>
  <div class="container">
    <h1>Give Your Feedback</h1>
    <form id="feedbackForm">
      <input type="text" id="name" placeholder="Your Name" required />
      <textarea id="feedback" placeholder="Your Feedback" required></textarea>
      <button type="submit">Submit</button>
    </form>
    <h2>Previous Feedback</h2>
    <ul id="feedbackList"></ul>
  </div>
  <script src="script.js"></script>
</body>
</html>
```

frontend/script.js →

```
const form = document.getElementById('feedbackForm');
const feedbackList = document.getElementById('feedbackList');
const API_BASE_PATH = '/api';
form.addEventListener('submit', async (e) => {
  e.preventDefault();
  const name = document.getElementById('name').value;
  const feedback = document.getElementById('feedback').value;

  await fetch(`${API_BASE_PATH}/feedback`, {
    method: 'POST',
    headers: { 'Content-Type': 'application/json' },
    body: JSON.stringify({ name, feedback })
  });

  document.getElementById('name').value = '';
  document.getElementById('feedback').value = '';
  loadFeedbacks();
});

async function loadFeedbacks() {
  try {
    const res = await fetch(`/api/feedback`);
    if (!res.ok) {
      const errorData = await res.json();
      throw new Error(errorData.message || 'Failed to fetch feedbacks');
    }

    const data = await res.json();
    feedbackList.innerHTML = '';
    if (Array.isArray(data)) {
      data.forEach(fb => {
        const li = document.createElement('li');
        li.textContent = `${fb.name}: ${fb.feedback}`;
        feedbackList.appendChild(li);
      });
    } else {
      console.error('Received data is not an array:', data);
    }
  } catch (error) {
    console.error('Error loading feedbacks:', error);
    feedbackList.innerHTML = `<li>Error: ${error.message}</li>`;
  }
}
loadFeedbacks();
```

frontend/entrypoint.sh →

```
#!/bin/sh
set -e
# Replace the placeholder in the template and create the final config file
# inside the /etc/nginx/conf.d/ directory.
sed "s|__BACKEND_URL__|${BACKEND_URL}|g" /etc/nginx/conf.d/default.conf.template > /etc/nginx/conf.d/default.conf
# Start Nginx in the foreground
nginx -g "daemon off;"
```

frontend/dockerfile →

```
FROM nginx:alpine
COPY . /usr/share/nginx/html
# Copy the Nginx configuration template to the correct directory
# This will become /etc/nginx/conf.d/default.conf after the entrypoint script runs
COPY nginx.conf /etc/nginx/conf.d/default.conf.template
COPY entrypoint.sh /entrypoint.sh
RUN chmod +x /entrypoint.sh
# The entrypoint script will generate the final config and start Nginx
CMD ["/entrypoint.sh"]
```

frontend/nginx.conf →

```
server {
    listen 80;

    # Serve static files from the root
    location / {
        root   /usr/share/nginx/html;
        index  index.html;
        # This line is important for single-page applications
        try_files $uri $uri/ /index.html;
    }

    # Proxy API requests to the backend
    location /api/ {
        # The __BACKEND_URL__ placeholder will be replaced by the entrypoint script.
        # The trailing slash is important! It maps /api/ to the root of the backend URL.
        # e.g., /api/feedback becomes http://<backend-ip>:5000/feedback
        proxy_pass http://__BACKEND_URL__/;

        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;
    }
}
```

1.2.2 Backend

- **Tech:** Node.js + Express + PostgreSQL
- **Code:** `backend/index.js` with `POST /feedback`, `GET /feedback`, `GET /`

```
diptesh-singh@ASUS-TUF:~/Desktop/Diptesh_Singh_Devops/Project/2-tier-app/backend$ ls
dockerfile  index.js  package.json
```

backend/dockerfile →

```
FROM node:18
ENV PGUSER=postgres
ENV PGPASSWORD=postgres
ENV PGDATABASE=feedbackdb

WORKDIR /app
COPY package*.json ./
RUN npm install
COPY . .
EXPOSE 5000
CMD ["node", "index.js"]
```

backend/package.json →

```
{
  "name": "feedback-backend",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "start": "node index.js"
  },
  "dependencies": {
    "express": "^4.18.2",
    "pg": "^8.10.0",
    "cors": "^2.8.5"
  }
}
```

backend/index.js →

```
const express = require('express');
const cors = require('cors');
const { Pool } = require('pg');

const app = express();
app.use(cors());
app.use(express.json());

const pool = new Pool({
  host: process.env.PGHOST,
  user: process.env.PGUSER,
  password: process.env.PGPASSWORD,
  database: process.env.PGDATABASE,
});

app.post('/feedback', async (req, res) => {
  const { name, feedback } = req.body;
  try {
    await pool.query('INSERT INTO feedbacks (name, feedback) VALUES ($1, $2)', [name, feedback]);
    res.status(201).json({ message: 'Feedback received successfully' });
  } catch (err) {
    console.error(err);
    res.status(500).json({ message: 'Error saving feedback' });
  }
});

app.get("/", (req, res) => {
  res.json({ message: "Backend is running" });
});

app.get('/feedback', async (req, res) => {
  try {
    const result = await pool.query('SELECT * FROM feedbacks');
    res.json(result.rows);
  } catch (err) {
    console.error(err);
    res.status(500).json({ message: 'Error retrieving feedbacks' });
  }
});

app.listen(5000, '0.0.0.0', () => console.log('Backend running on port 5000'));
```

1.2.3 Database

db/init.sql →

```
CREATE TABLE IF NOT EXISTS feedbacks (
  id SERIAL PRIMARY KEY,
  name VARCHAR(100),
  feedback TEXT
);
```

db/dockerfile →

```
FROM postgres:latest
ENV POSTGRES_USER=postgres
ENV POSTGRES_PASSWORD=postgres
ENV POSTGRES_DB=feedbackdb
COPY init.sql /docker-entrypoint-initdb.d/
```

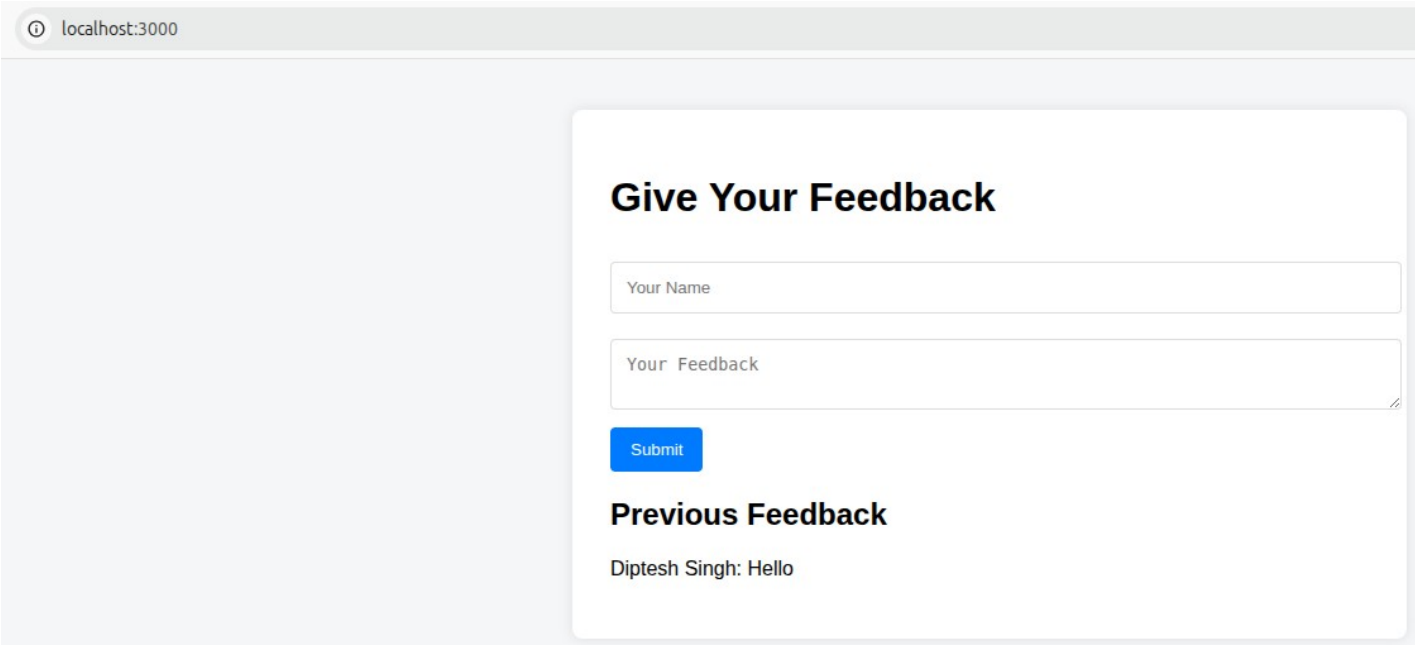
docker-compose.yml -

```
version: '3.8'

services:
  db:
    build: ./db
    image: feedback-db
    container_name: db
    volumes:
      - pgdata:/var/lib/postgresql/data
    ports:
    - "5432:5432"

  backend:
    build: ./backend
    image: feedback-backend
    container_name: backend
    environment:
      - PGHOST=db
    ports:
      - "5000:5000"
    depends_on:
      - db
  frontend:
    build: ./frontend
    image: feedback-frontend
    container_name: frontend
    ports:
      - "3000:80"
    depends_on:
      - backend
    # Provide the backend URL to Nginx for the reverse proxy
    environment:
      BACKEND_URL: backend:5000

volumes:
  pgdata:
```

1.2.4 Building Images and Running Containers using `docker-compose.yml` -

```
diptesh-singh@ASUS-TUF:~/Desktop/Diptesh_Singh_Devops/Project/2-tier-app$ sudo docker compose up
[sudo] password for diptesh-singh:
WARN[0000] /home/diptesh-singh/Desktop/Diptesh_Singh_Devops/Project/2-tier-app/docker-compose.yml: the attribute `version` is obsolete, it will be ignored, please remove it to avoid po
tential confusion
[+] Running 3/3
 ! frontend Warning pull access denied for feedback-frontend, repository does not exist or may require 'docker login': denied: requested access to the resour...                3.4s
 ! backend Warning  pull access denied for feedback-backend, repository does not exist or may require 'docker login': denied: requested access to the resource...          3.9s
 ! db Warning       pull access denied for feedback-db, repository does not exist or may require 'docker login': denied: requested access to the resource is denied          3.9s
[+] Building 1.2s (31/31) FINISHED
 => [internal] load local bake definitions                                                                                                                              0.0s
 => => reading from stdin 1.18kB                                                                                                                                        0.0s
```

1.2.5 Accessing the Frontend Application via Browser



1.2.6 Verifying Frontend Data Submission in Database

```
diptesh-singh@ASUS-TUF:~/Desktop/Diptesh_Singh_Devops/Project/2-tier-app$ sudo docker ps
[sudo] password for diptesh-singh:
CONTAINER ID   IMAGE              COMMAND                  CREATED              STATUS              PORTS                                                  NAMES
4a4a642c0eda   feedback-frontend  "/docker-entrypoint.…"   About a minute ago   Up About a minute   0.0.0.0:3000->80/tcp, [::]:3000->80/tcp                frontend
6152b20f214c   feedback-backend   "docker-entrypoint.s…"   About a minute ago   Up About a minute   0.0.0.0:5000->5000/tcp, [::]:5000->5000/tcp            backend
2c893d3b6027   feedback-db        "docker-entrypoint.s…"   About a minute ago   Up About a minute   0.0.0.0:5432->5432/tcp, [::]:5432->5432/tcp            db
diptesh-singh@ASUS-TUF:~/Desktop/Diptesh_Singh_Devops/Project/2-tier-app$ sudo docker images
REPOSITORY              TAG           IMAGE ID       CREATED         SIZE
feedback-backend        latest        57f85bac5a89   4 minutes ago   1.1GB
feedback-frontend       latest        ed082bbc33e0   2 hours ago     52.5MB
feedback-db             latest        1995214f8da6   37 hours ago    438MB
getting-started         latest        215a0faedd7d   4 days ago      251MB
java-app                latest        5791bd033b29   7 days ago      471MB
dipteshsingh/java-app   latest        5791bd033b29   7 days ago      471MB
nginx                   latest        2cd1d97f893f   2 weeks ago     192MB
dipteshsingh/nginx-d    latest        634374cbc450   3 months ago    240MB
nginx                   stable-perl   634374cbc450   3 months ago    240MB
hello-world             latest        74cc54e27dc4   6 months ago    10.1kB
diptesh-singh@ASUS-TUF:~/Desktop/Diptesh_Singh_Devops/Project/2-tier-app$ sudo docker exec -it db psql -U postgres -d feedbackdb
psql (17.5 (Debian 17.5-1.pgdg120+1))
Type "help" for help.

feedbackdb=# SELECT * FROM feedbacks;
 id |     name      | feedback
----+---------------+----------
  1 | Diptesh Singh | Hello   +
    |               |
(1 row)

feedbackdb=# exit
```

1.2.7 Uploading Application Images to DockerHub

```
diptesh-singh@ASUS-TUF:~/Desktop/Diptesh_Singh_Devops/Project/2-tier-app$ sudo docker tag feedback-backend dipteshsingh/backend
diptesh-singh@ASUS-TUF:~/Desktop/Diptesh_Singh_Devops/Project/2-tier-app$ sudo docker tag feedback-frontend dipteshsingh/frontend
diptesh-singh@ASUS-TUF:~/Desktop/Diptesh_Singh_Devops/Project/2-tier-app$ sudo docker tag feedback-db dipteshsingh/feedback-db
diptesh-singh@ASUS-TUF:~/Desktop/Diptesh_Singh_Devops/Project/2-tier-app$ sudo docker login
Authenticating with existing credentials... [Username: dipteshsingh]

i Info → To login with a different account, run 'docker logout' followed by 'docker login'


Login Succeeded
diptesh-singh@ASUS-TUF:~/Desktop/Diptesh_Singh_Devops/Project/2-tier-app$ sudo docker push dipteshsingh/frontend
Using default tag: latest
The push refers to repository [docker.io/dipteshsingh/frontend]
a727a6880386: Pushed
b24609a5c521: Pushed
4f9da84de2fe: Pushed
c0b087e1f2ee: Pushed
57fb2e22a07a: Mounted from library/nginx
c38bee0b0d28: Mounted from library/nginx
26081059fc81: Mounted from library/nginx
daa8ffa7606a: Mounted from library/nginx
95a6190cfaec: Mounted from library/nginx
430a7aa99a19: Mounted from library/nginx
77a17eed5d29: Mounted from library/nginx
418dccb7d85a: Mounted from library/nginx
latest: digest: sha256:c34e6e0dff94c3665cb336a3f9a5275e43c5b32844041a27e8f516f2f864c412 size: 2818
diptesh-singh@ASUS-TUF:~/Desktop/Diptesh_Singh_Devops/Project/2-tier-app$ sudo docker push dipteshsingh/backend
Using default tag: latest
The push refers to repository [docker.io/dipteshsingh/backend]
```

```
diptesh-singh@ASUS-TUF:~/Desktop/Diptesh_Singh_Devops/Project/2-tier-app$ sudo docker push dipteshsingh/backend
Using default tag: latest
The push refers to repository [docker.io/dipteshsingh/backend]
0f71309c8299: Pushed
c425ba96311a: Pushed
639da4a7bfba: Pushed
5d4902b180a8: Pushed
d2a991bcab4d: Pushed
b624aa2d5ea2: Pushed
d399c9dc306f: Pushed
84f9fa179c1b: Pushed
ce84ba212e49: Pushed
e4dc8cd9ecc8: Pushed
6428cc293366: Pushed
2f7436e79a0b: Pushed
latest: digest: sha256:d0661b05ac09dca7969671436052abce5e46ff2ae29aaba9aee15b9b176aef6a size: 2835
diptesh-singh@ASUS-TUF:~/Desktop/Diptesh_Singh_Devops/Project/2-tier-app$ sudo docker push dipteshsingh/feedback-db
Using default tag: latest
The push refers to repository [docker.io/dipteshsingh/feedback-db]
93d78a4a4b7e: Pushed
8d34d0bc792a: Mounted from library/postgres
d4f5cc113c6c: Mounted from library/postgres
5253dc3f8436: Mounted from library/postgres
af4e3ef2fba7: Mounted from library/postgres
4a0102318d7f: Mounted from library/postgres
f83032ba7173: Mounted from library/postgres
f6496299669a: Mounted from library/postgres
0efc75ba8fa6: Mounted from library/postgres
aee8bdcf1c88: Mounted from library/postgres
51769ad1eaeb: Mounted from library/postgres
355851f35662: Mounted from library/postgres
225e82c94175: Mounted from library/postgres
bf168e8d96de: Mounted from library/postgres
7cc7fe68eff6: Mounted from dipteshsingh/nginx-d
latest: digest: sha256:aef18085e7ba35e101b9d9e2c269648fac830650c1463343271e30b9cd05debb size: 3455
```

1.2.8 Saving the DockerHub Pull Commands

```
diptesh-singh@ASUS-TUF:~/Desktop/Diptesh_Singh_Devops/Project/2-tier-app$ echo "docker pull dipteshsingh/frontend:latest" > pull-commands.txt
diptesh-singh@ASUS-TUF:~/Desktop/Diptesh_Singh_Devops/Project/2-tier-app$ echo "docker pull dipteshsingh/backend:latest" >> pull-commands.txt
diptesh-singh@ASUS-TUF:~/Desktop/Diptesh_Singh_Devops/Project/2-tier-app$ echo "docker pull dipteshsingh/feedback-db:latest" >> pull-commands.txt
diptesh-singh@ASUS-TUF:~/Desktop/Diptesh_Singh_Devops/Project/2-tier-app$ vim pull-commands.txt
```

```
docker pull dipteshsingh/frontend:latest
docker pull dipteshsingh/backend:latest
docker pull dipteshsingh/feedback-db:latest
~
```

1.3 Terraform Files & Folder Structure

```
infra/
├── main.tf
├── variables.tf
├── outputs.tf
├── provider.tf
└── files/
    ├── frontend.sh
    └── backend.sh
```

```
diptesh-singh@ASUS-TUF:~/Desktop/Diptesh_Singh_Devops/Project/Create_Infra$ ls
files  main.tf  output.tf  provider.tf  script.sh  terraform.tfstate  terraform.tfstate.backup  variables.tf
```

variables.tf →

```
variable "frontend_port" {
  description = "Port number for the frontend application"
  type        = number
  default     = 3000
}

variable "backend_port" {
  description = "Port number for the backend application"
  type        = number
  default     = 5000
}
```

provider.tf →

```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 6.0"
    }
  }
}
```

main.tf →(I) -

```
resource "null_resource" "check_backend_sh_exists" {
  provisioner "local-exec" {
    command = "test -f files/backend.sh || (echo 'backend.sh not found!' && exit 1)"
  }
}

resource "null_resource" "check_frontend_sh_exists" {
  provisioner "local-exec" {
    command = "test -f files/frontend.sh || (echo 'frontend.sh not found!' && exit 1)"
  }
}
```

(II) -

```
resource "aws_vpc" "MyVpc" {
   cidr_block = "10.0.0.0/16"
   instance_tenancy = "default"
   tags = {
     Name = "diptesh-VPC"
   }
}

resource "aws_subnet" "public" {
   vpc_id = aws_vpc.MyVpc.id
   cidr_block = "10.0.1.0/24"
   tags = {
     Name = "PUBLIC"
    }
}

resource "aws_subnet" "private" {
   vpc_id = aws_vpc.MyVpc.id
   cidr_block = "10.0.2.0/24"
   tags = {
     Name = "PRIVATE"
    }
}

resource "aws_internet_gateway" "igw" {
    vpc_id = aws_vpc.MyVpc.id
    tags = {
      Name = "diptesh-igw"
     }
}

resource "aws_route_table" "rt-ig" {
  vpc_id = aws_vpc.MyVpc.id
  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = aws_internet_gateway.igw.id
  }

  tags = {
    Name = "diptesh-rt-ig"
   }
```

(III) -

```
resource "aws_route_table_association" "rt-a-public" {
  subnet_id = aws_subnet.public.id
  route_table_id = aws_route_table.rt-ig.id
}

resource "aws_eip" "eip" {
  tags = {
    Name = "nat-gateway-eip"
  }
}

resource "aws_nat_gateway" "ngw" {
  allocation_id = aws_eip.eip.id
  subnet_id     = aws_subnet.public.id
  tags = {
    Name = "diptesh-ngw"
  }

  # To ensure proper ordering, it is recommended to add
  # on the Internet Gateway for the VPC.
  depends_on = [
    aws_internet_gateway.igw,
    null_resource.check_frontend_sh_exists,
    null_resource.check_backend_sh_exists
  ]

}

resource "aws_route_table" "rt-nat" {
  vpc_id = aws_vpc.MyVpc.id
  route {
    cidr_block = "0.0.0.0/0"
    nat_gateway_id = aws_nat_gateway.ngw.id
   }
  tags = {
    Name = "diptesh-rt-nat"
   }
}


resource "aws_route_table_association" "rt-a-private" {
   subnet_id = aws_subnet.private.id
   route_table_id = aws_route_table.rt-nat.id
}
```

(IV) -

```
data "aws_ami" "ubuntu" {
  most_recent = true
  filter {
    name   = "name"
    values = ["ubuntu/images/hvm-ssd/ubuntu-jammy-22.04-amd64-server-*"]
       }
  filter {
    name   = "virtualization-type"
    values = ["hvm"]
  }
  owners = ["099720109477"]
}
```

(V) -

```
resource "aws_security_group" "sg_public_ins" {
   name = "allow_ssh_public"
   description = "Allow SSH inbound traffic to public instance"
   vpc_id = aws_vpc.MyVpc.id
   ingress {
     description = "SSH from anywhere"
     from_port = 22
     to_port = 22
     protocol = "tcp"
     cidr_blocks = ["0.0.0.0/0"]
   }
   ingress {
     description     = "Allow ping from everywhere"
     from_port       = -1
     to_port         = -1
     protocol        = "icmp"
     cidr_blocks = ["0.0.0.0/0"]
   }
   ingress {
     description = "Allow HTTP from everywhere on port 3000"
     from_port   = 3000
     to_port     = 3000
     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 = "diptesh-sg-public"
   }
}
```

(VI) -

```
resource "aws_security_group" "sg_private_ins" {
   name = "allow_ssh_private"
   description = "Allow SSH inbound traffic to private instance"
   vpc_id = aws_vpc.MyVpc.id
   ingress {
     description = "Allow SSH from within the VPC"
     from_port = 22
     to_port = 22
     protocol = "tcp"
     cidr_blocks = [aws_subnet.public.cidr_block]
   }
   ingress {
     description     = "Allow ping from within the VPC"
     from_port       = -1
     to_port         = -1
     protocol        = "icmp"
     cidr_blocks = [aws_subnet.public.cidr_block]
   }
   egress {
     from_port   = 0
     to_port     = 0
     protocol    = "-1"
     cidr_blocks = ["0.0.0.0/0"]
   }
   tags = {
     Name = "diptesh-sg-private"
   }
}

resource "aws_security_group_rule" "allow_frontend_to_backend" {
  type                     = "ingress"
  from_port                = 5000
  to_port                  = 5000
  protocol                 = "tcp"
  security_group_id        = aws_security_group.sg_private_ins.id
  source_security_group_id = aws_security_group.sg_public_ins.id
  description              = "Allow frontend SG - backend on port 5000"
}

resource "tls_private_key" "private_key_pair" {
  algorithm = "RSA"
}

resource "local_file" "private_key_pair" {
   content = tls_private_key.private_key_pair.private_key_pem
   filename = "backend.pem"
   file_permission = "0400"
}

resource "aws_key_pair" "private_key_pair" {
   key_name = "diptesh-private-key"
   public_key = tls_private_key.private_key_pair.public_key_openssh
}
```

(VII) -

```
resource "aws_instance" "backend" {
   ami = data.aws_ami.ubuntu.id
   instance_type = "t3.micro"
   key_name = aws_key_pair.private_key_pair.key_name
   subnet_id = aws_subnet.private.id
   vpc_security_group_ids = [aws_security_group.sg_private_ins.id]
   depends_on = [aws_key_pair.private_key_pair]

   tags = {
     Name = "BACKEND"
    }
}

resource "tls_private_key" "public_key_pair" {
  algorithm = "RSA"
}

resource "local_file" "public_key_pair" {
   content = tls_private_key.public_key_pair.private_key_pem
   filename = "frontend.pem"
   file_permission = "0400"
}

resource "aws_key_pair" "public_key_pair" {
   key_name = "diptesh-public-key"
   public_key = tls_private_key.public_key_pair.public_key_openssh
}

resource "aws_instance" "frontend" {
   ami = data.aws_ami.ubuntu.id
   instance_type = "t3.micro"
   key_name = aws_key_pair.public_key_pair.key_name
   subnet_id = aws_subnet.public.id
   vpc_security_group_ids = [aws_security_group.sg_public_ins.id]
   associate_public_ip_address = true
   depends_on = [
     aws_key_pair.public_key_pair,
   ]

   tags = {
     Name = "FRONTEND"
    }
}
```

(VIII) -

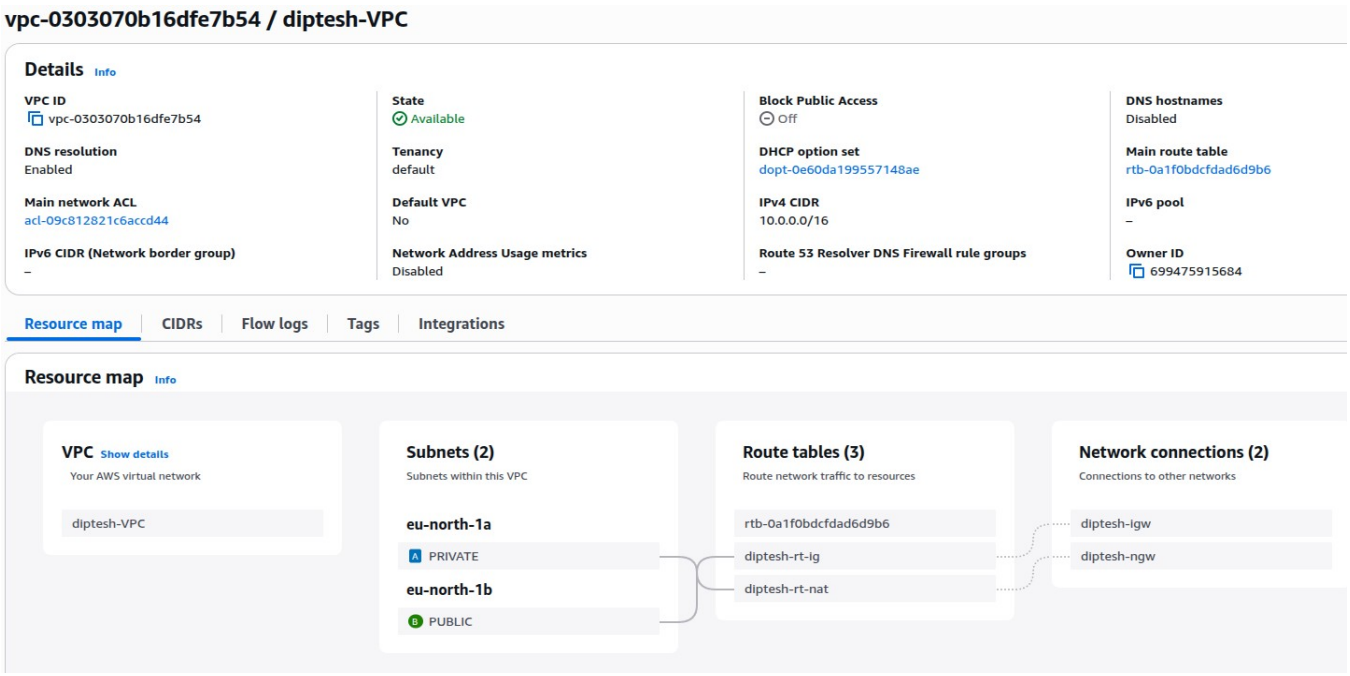
```
resource "null_resource" "backend_provisioner" {
   depends_on = [aws_instance.backend,
                 aws_instance.frontend,
                 null_resource.check_backend_sh_exists
                ]
   provisioner "file" {
     source = "files/backend.sh"
     destination = "backend.sh"
   }
   provisioner "remote-exec" {
     inline = [
          "chmod +x backend.sh",
          "sudo ./backend.sh ${var.backend_port}",
          "sudo docker --version",
          "sudo docker ps",
         ]
    }
   connection {
     type = "ssh"
     user = "ubuntu"
     private_key = tls_private_key.private_key_pair.private_key_pem
     host = aws_instance.backend.private_ip
     bastion_host = aws_instance.frontend.public_ip
     bastion_user = "ubuntu"
     bastion_private_key = tls_private_key.public_key_pair.private_key_pem
   }
}
```

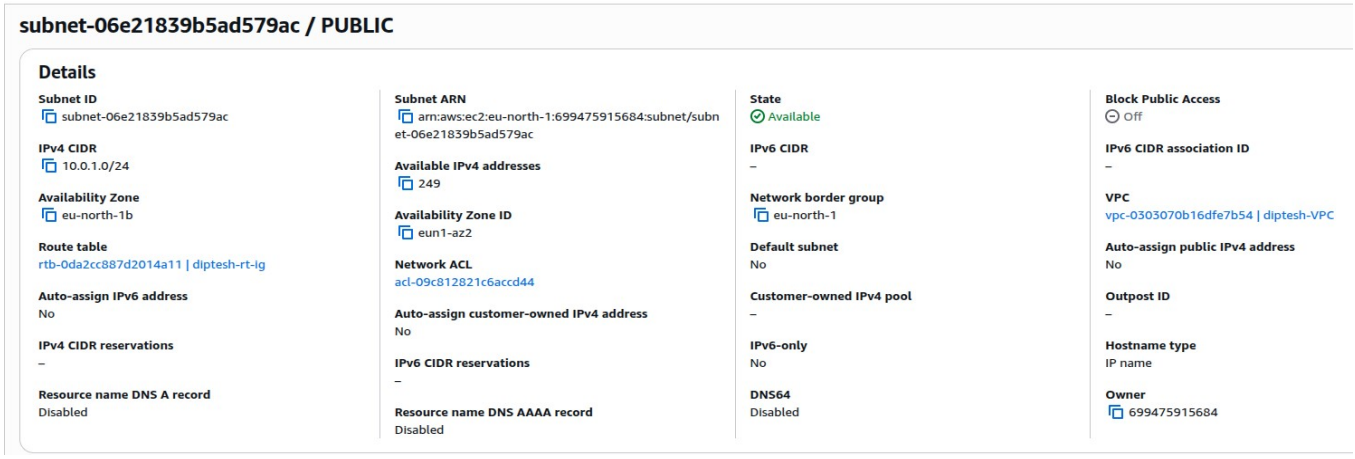
(IX) -

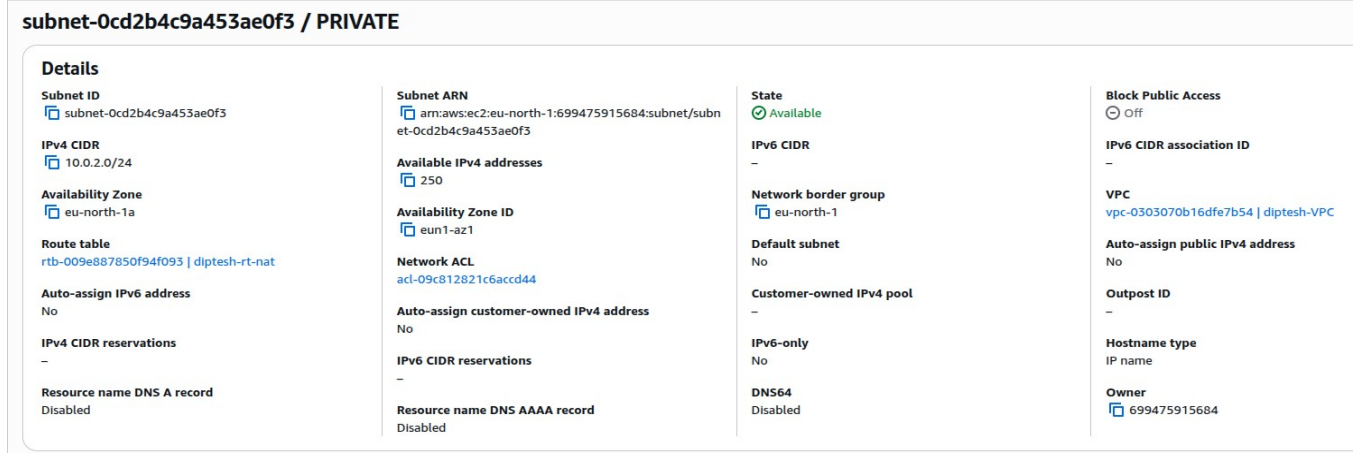
```
resource "null_resource" "frontend_provisioner" {
   depends_on = [aws_instance.backend,
                 aws_instance.frontend,
                 null_resource.check_frontend_sh_exists,
                 null_resource.backend_provisioner
                ]
   provisioner "file" {
     source = "backend.pem"
     destination = "backend.pem"
   }
   provisioner "file" {
     source = "files/frontend.sh"
     destination = "frontend.sh"
   }
   provisioner "remote-exec" {
     inline = [
          "chmod +x frontend.sh",
          "sudo ./frontend.sh ${aws_instance.backend.private_ip} ${var.frontend_port}",
          "chmod 400 backend.pem",
          "sudo docker --version",
          "sudo docker ps",
          "curl -I http://${aws_instance.backend.private_ip}:${var.backend_port}"
         ]
    }
   connection {
     type = "ssh"
     user = "ubuntu"
     private_key = tls_private_key.public_key_pair.private_key_pem
     host = aws_instance.frontend.public_ip
   }
}
```

1.4 VPC & Subnets :-

- **VPC CIDR:** `10.0.0.0/16`
- **Public Subnet:** `10.0.1.0/24`
- **Private Subnet:** `10.0.2.0/24`
- **IGW** attached to VPC; Route Table for public subnet directs `0.0.0.0/0` → IGW.
- **NAT Gateway** in public subnet; private subnet route table sends `0.0.0.0/0` → NAT.

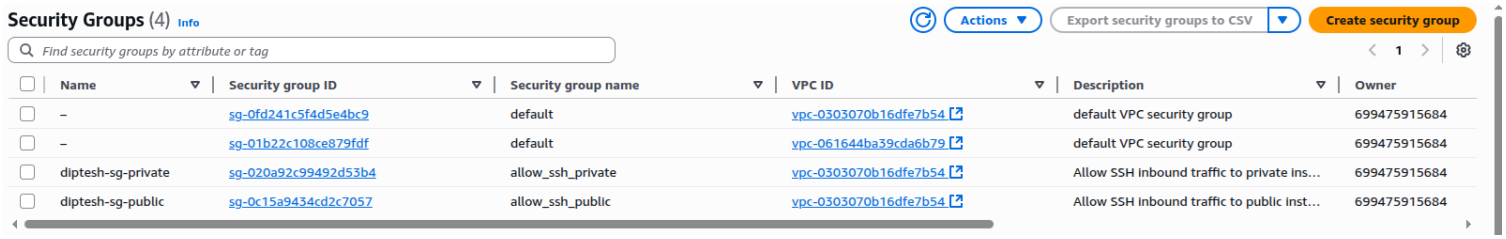


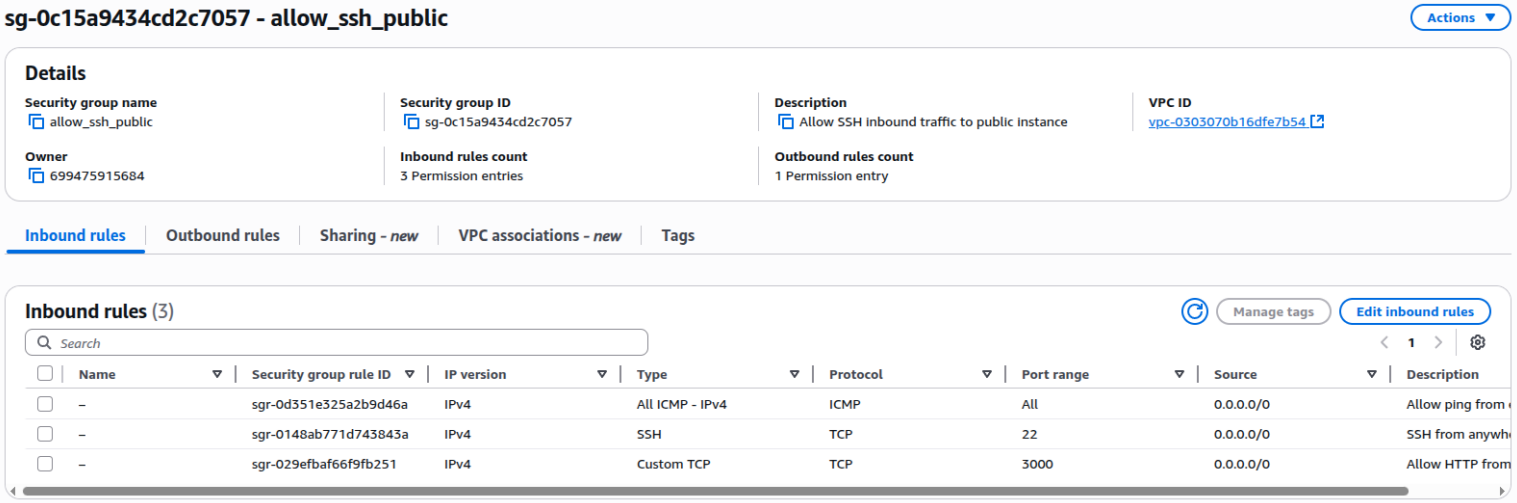


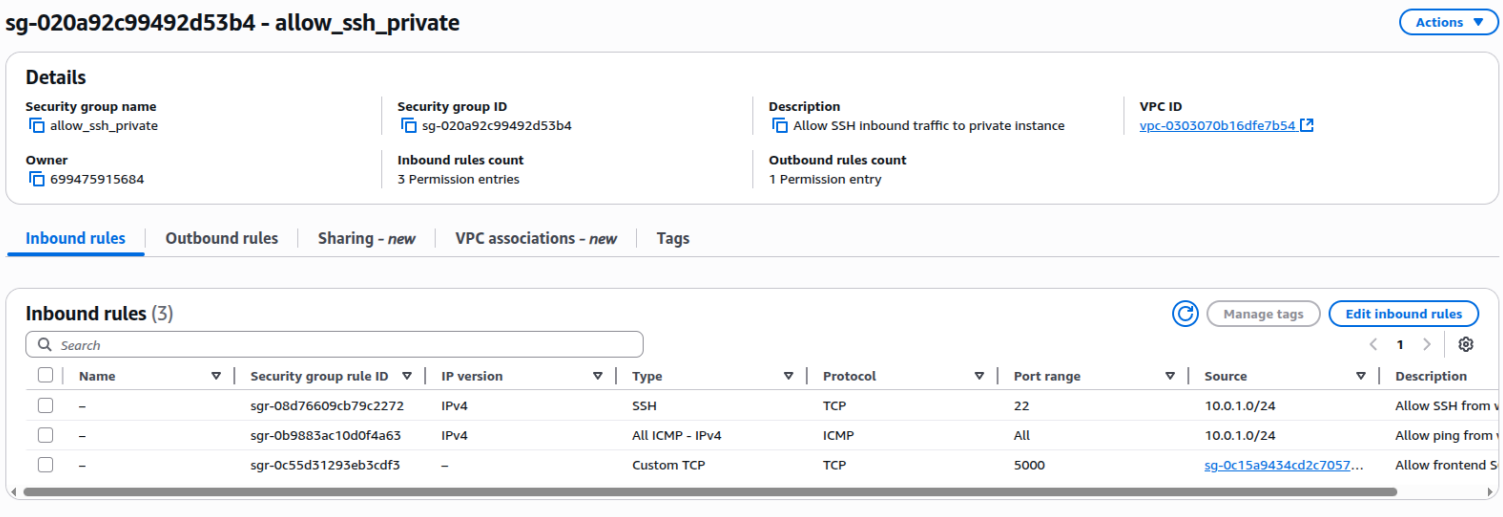


1.5 Security Groups

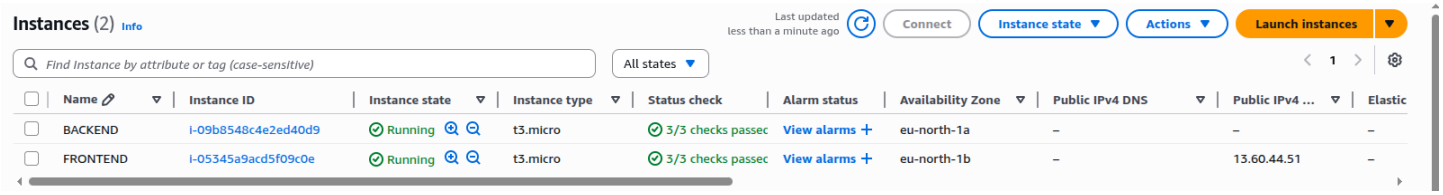
- **sg_public_ins** (for frontend):
    - Ingress: TCP 22, TCP 3000, ICMP from `0.0.0.0/0`
- **sg_private_ins** (for backend):
    - Ingress: TCP 22 & ICMP from public subnet
    - Ingress: TCP 5000 from `sg_public_ins`
- Egress: All outbound



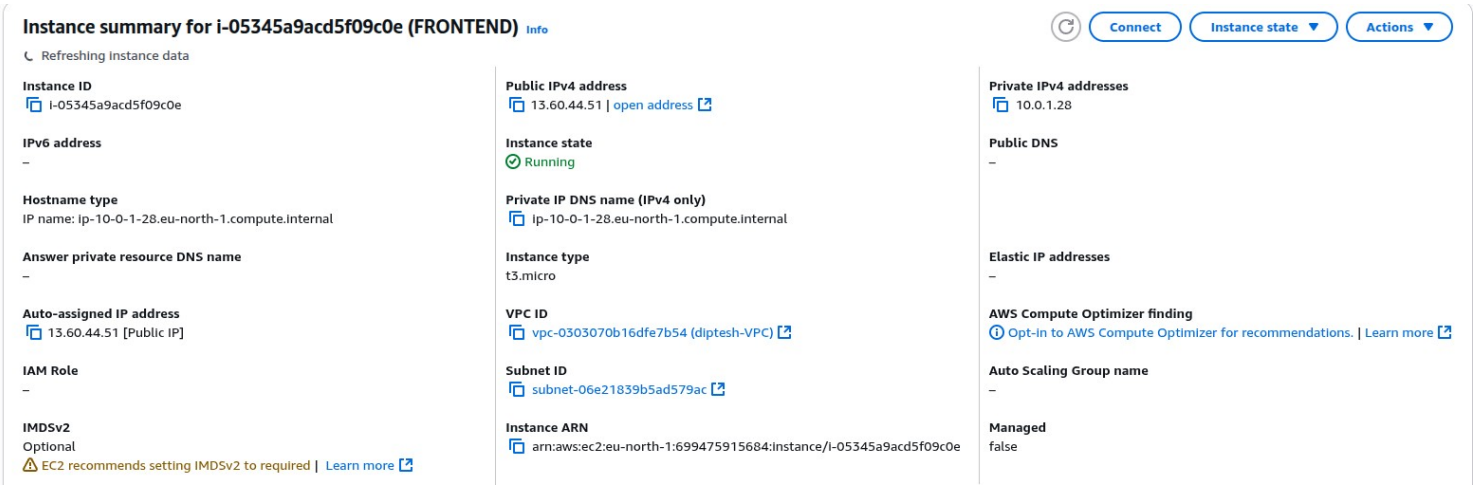




1.5 EC2 Instances



- **FRONTEND**
  - AMI: Ubuntu 22.04, Type: t3.micro, Public IP: Yes
  - Security Group: `sg_public_ins`



- **BACKEND**
  - AMI: Ubuntu 22.04, Type: t3.micro, Private IP only
  - Security Group: `sg_private_ins`

Stage 2: Deploy_Apps

2.1 Objective

Using terraform provisioners execute the scripts in respective systems:

- frontend.sh: Installs and Configures docker in the FRONTEND instance and runs the containerized frontend in it using the pull request command in previous slide.

- backend.sh: Installs and Configures docker in BACKEND instance and runs the containerized backend in it using the pull request command in previous slide.

- Use remote-exec provisioner to find out if docker has been installed and the application is running in the local system.

- **FRONTEND** container listens on port 3000→80
- **BACKEND** container listens on port 5000; connects to `db` container

2.2 frontend.sh -

```
#!/bin/bash
set -e
backend_private_ip=$1
port_no=$2
echo "Backend IP is: $backend_private_ip"
echo "Installing Docker"
# Add Docker's official GPG key:
sudo apt-get update
sudo apt-get install -y ca-certificates curl
sudo install -m 0755 -d /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc
sudo chmod a+r /etc/apt/keyrings/docker.asc

# Add the repository to Apt sources:
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/ubuntu \
  $(. /etc/os-release && echo "${UBUNTU_CODENAME:-$VERSION_CODENAME}") stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update
sudo apt-get install -y docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
sudo apt install -y git

sudo docker pull dipteshsingh/frontend
sudo docker images
sudo docker run -d -p ${port_no}:80 -e BACKEND_URL=${backend_private_ip}:5000 --name frontend dipteshsingh/frontend

echo "Waiting for frontend service to start..."
for i in {1..24}; do
  # Use curl to check if the service is responding successfully
  if curl -s --head --fail http://localhost:${port_no} > /dev/null; then
    echo " Frontend service is up and running!"
    exit 0
  fi
  printf "."
  sleep 5
done

echo "Frontend service failed to start within the timeout period."
exit 1
```

NOTE – Here, the `port_no` and `backend_private_ip` are being passed from the `main.tf` file.

2.3 backend.sh -

```
#!/bin/bash
set -e
port_no=$1
echo "Installing Docker"
sudo rm -rf /var/lib/apt/lists/*
sudo apt-get clean
# Add Docker's official GPG key:
sudo apt-get update
sudo apt-get install -y ca-certificates curl
sudo install -m 0755 -d /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc
sudo chmod a+r /etc/apt/keyrings/docker.asc

# Add the repository to Apt sources:
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/ubuntu \
  $(. /etc/os-release && echo "${UBUNTU_CODENAME:-$VERSION_CODENAME}") stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update
sudo apt-get install -y docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
sudo apt install -y git

sudo docker network create myapp-network || true
sudo docker pull dipteshsingh/feedback-db
sudo docker pull dipteshsingh/backend
sudo docker images
sudo docker volume create pgdata
sudo docker run -d -p 5432:5432 -v pgdata:/var/lib/postgresql/data --name db  --network myapp-network dipteshsingh/feedback-db
sudo docker run -d -p ${port_no}:5000 -e PGHOST=db --name backend  --network myapp-network dipteshsingh/backend

echo "Waiting for backend service to start..."
for i in {1..24}; do
  # Use curl to check if the service is responding successfully
  if curl -s --head --fail http://localhost:${port_no} > /dev/null; then
    echo " Backend service is up and running!"
    exit 0
  fi
  printf "."
  sleep 10
done

echo " Backend service failed to start within the timeout period."
exit 1
```

NOTE – Here, the `port_no` is being sent from the frontend. Both the backend and database containers must be on the same Docker network (`myapp-network`). Additionally, the database container name and the value of the `PGHOST` environment variable should both be set to `db`.

2.4 Verification via remote-exec

script.sh -

```
#!/bin/bash
set -e
terraform init
terraform plan
terraform apply -auto-approve
```

2.4.1 Connecting to the backend instance, executing the `backend.sh` script, and verifying if Docker is installed using `docker --version`, and confirming that the application is running locally using `docker ps` and `curl` -

```
diptesh-singh@ASUS-TUF:~/Desktop/Diptesh_Singh_Devops/Project/Create_Infra$ ./script.sh
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/tls from the dependency lock file
- Reusing previous version of hashicorp/local from the dependency lock file
- Reusing previous version of hashicorp/aws from the dependency lock file
- Reusing previous version of hashicorp/null from the dependency lock file
- Using previously-installed hashicorp/tls v4.1.0
- Using previously-installed hashicorp/local v2.5.3
- Using previously-installed hashicorp/aws v6.6.0
- Using previously-installed hashicorp/null v3.2.4

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
data.aws_ami.ubuntu: Reading...
data.aws_ami.ubuntu: Read complete after 3s [id=ami-0b8e4d801c75b0f0d]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
  + create
```

```
aws_instance.frontend: Still creating... [00m10s elapsed]
aws_instance.backend: Still creating... [00m10s elapsed]
aws_instance.frontend: Creation complete after 14s [id=i-05345a9acd5f09c0e]
aws_nat_gateway.ngw: Still creating... [00m20s elapsed]
aws_instance.backend: Creation complete after 16s [id=i-09b8548c4e2ed40d9]
null_resource.backend_provisioner: Creating...
null_resource.backend_provisioner: Provisioning with 'file'...
aws_nat_gateway.ngw: Still creating... [00m30s elapsed]
null_resource.backend_provisioner: Still creating... [00m10s elapsed]
null_resource.backend_provisioner: Provisioning with 'remote-exec'...
null_resource.backend_provisioner (remote-exec): Connecting to remote host via SSH...
null_resource.backend_provisioner (remote-exec):   Host: 10.0.2.57
null_resource.backend_provisioner (remote-exec):   User: ubuntu
null_resource.backend_provisioner (remote-exec):   Password: false
null_resource.backend_provisioner (remote-exec):   Private key: true
null_resource.backend_provisioner (remote-exec):   Certificate: false
null_resource.backend_provisioner (remote-exec):   SSH Agent: true
null_resource.backend_provisioner (remote-exec):   Checking Host Key: false
null_resource.backend_provisioner (remote-exec):   Target Platform: unix
null_resource.backend_provisioner (remote-exec): Using configured bastion host...
null_resource.backend_provisioner (remote-exec):   Host: 13.60.44.51
null_resource.backend_provisioner (remote-exec):   User: ubuntu
null_resource.backend_provisioner (remote-exec):   Password: false
null_resource.backend_provisioner (remote-exec):   Private key: true
null_resource.backend_provisioner (remote-exec):   Certificate: false
null_resource.backend_provisioner (remote-exec):   SSH Agent: true
null_resource.backend_provisioner (remote-exec):   Checking Host Key: false
aws_nat_gateway.ngw: Still creating... [00m40s elapsed]
null_resource.backend_provisioner: Still creating... [00m20s elapsed]
null_resource.backend_provisioner (remote-exec): Connected!
null_resource.backend_provisioner (remote-exec): Installing Docker
```

```
null_resource.backend_provisioner (remote-exec): git is already the newest version (1:2.34.1-1ubuntu1.15).
null_resource.backend_provisioner (remote-exec): git set to manually installed.
null_resource.backend_provisioner (remote-exec): 0 upgraded, 0 newly installed, 0 to remove and 16 not upgraded.
null_resource.backend_provisioner (remote-exec): 7b1fb6a050d30f1d5117c0a39d1a054b1f0cbbe0c330ea6773eeed9f18f7f8ce
null_resource.backend_provisioner (remote-exec): Using default tag: latest
null_resource.backend_provisioner (remote-exec): latest: Pulling from dipteshsingh/feedback-db

null_resource.backend_provisioner (remote-exec): 59e22667830b: Pulling fs layer

null_resource.backend_provisioner (remote-exec): c2922dd5c76b: Pulling fs layer
```

```
null_resource.backend_provisioner (remote-exec): c89b43fc7596: Pull complete
null_resource.backend_provisioner (remote-exec): Digest: sha256:aef18085e7ba35e101b9d9e2c269648fac830650c1463343271e30b9cd05debb
null_resource.backend_provisioner (remote-exec): Status: Downloaded newer image for dipteshsingh/feedback-db:latest
null_resource.backend_provisioner (remote-exec): docker.io/dipteshsingh/feedback-db:latest
null_resource.backend_provisioner (remote-exec): Using default tag: latest
null_resource.backend_provisioner (remote-exec): latest: Pulling from dipteshsingh/backend

null_resource.backend_provisioner (remote-exec): 5c9256e0f3ca: Pulling fs layer

null_resource.backend_provisioner (remote-exec): 37927ed901b1: Pulling fs layer
```

```
null_resource.backend_provisioner (remote-exec): ab9e7705db7e: Pull complete
null_resource.backend_provisioner (remote-exec): Digest: sha256:d0661b05ac09dca7969671436052abce5e46ff2ae29aaba9aee15b9b176aef6a
null_resource.backend_provisioner (remote-exec): Status: Downloaded newer image for dipteshsingh/backend:latest
null_resource.backend_provisioner (remote-exec): docker.io/dipteshsingh/backend:latest
null_resource.backend_provisioner (remote-exec): REPOSITORY                TAG       IMAGE ID       CREATED          SIZE
null_resource.backend_provisioner (remote-exec): dipteshsingh/backend      latest    57f85bac5a89   21 minutes ago   1.1GB
null_resource.backend_provisioner (remote-exec): dipteshsingh/feedback-db   latest    1995214f8da6   37 hours ago     438MB
null_resource.backend_provisioner (remote-exec): pgdata
null_resource.backend_provisioner (remote-exec): 11f2cf8d753b9a4fa2bf61e027765eaff43cb80c3c3168858a0d6d508946ab41
null_resource.backend_provisioner (remote-exec): 56653a68fc1c9fb44de7cc1ac369c04e89a84aedb9ba7850339a1c2fcebf7756
null_resource.backend_provisioner (remote-exec): Waiting for backend service to start...
null_resource.backend_provisioner (remote-exec): .
null_resource.backend_provisioner: Still creating... [02m20s elapsed]
null_resource.backend_provisioner (remote-exec):  Backend service is up and running!
null_resource.backend_provisioner (remote-exec): Docker version 28.3.3, build 980b856
null_resource.backend_provisioner (remote-exec): CONTAINER ID   IMAGE                     COMMAND                  CREATED          STATUS          PORTS
          NAMES
null_resource.backend_provisioner (remote-exec): 56653a68fc1c   dipteshsingh/backend      "docker-entrypoint.s…"   11 seconds ago   Up 10 seconds   0.0.0.0:5000->5000/tcp, [::]:5000->
5000/tcp   backend
null_resource.backend_provisioner (remote-exec): 11f2cf8d753b   dipteshsingh/feedback-db   "docker-entrypoint.s…"   11 seconds ago   Up 10 seconds   0.0.0.0:5432->5432/tcp, [::]:5432->
5432/tcp   db
null_resource.backend_provisioner: Creation complete after 2m28s [id=1712291981652303100]
```

2.4.2 Connecting to the frontend instance, executing the `frontend.sh` script, verifying Docker installation using `docker --version`, and checking if the application is running locally using `docker ps` and `curl`.

```
null_resource.frontend_provisioner (remote-exec): Connecting to remote host via SSH...
null_resource.frontend_provisioner (remote-exec):   Host: 13.60.44.51
null_resource.frontend_provisioner (remote-exec):   User: ubuntu
null_resource.frontend_provisioner (remote-exec):   Password: false
null_resource.frontend_provisioner (remote-exec):   Private key: true
null_resource.frontend_provisioner (remote-exec):   Certificate: false
null_resource.frontend_provisioner (remote-exec):   SSH Agent: true
null_resource.frontend_provisioner (remote-exec):   Checking Host Key: false
null_resource.frontend_provisioner (remote-exec):   Target Platform: unix
null_resource.frontend_provisioner: Still creating... [00m10s elapsed]
null_resource.frontend_provisioner (remote-exec): Connected!
null_resource.frontend_provisioner (remote-exec): Backend IP is: 10.0.2.57
null_resource.frontend_provisioner (remote-exec): Installing Docker
null_resource.frontend_provisioner (remote-exec): 0% [Working]
```

```
null_resource.frontend_provisioner (remote-exec): git is already the newest version (1:2.34.1-1ubuntu1.15).
null_resource.frontend_provisioner (remote-exec): git set to manually installed.
null_resource.frontend_provisioner (remote-exec): 0 upgraded, 0 newly installed, 0 to remove and 16 not upgraded.
null_resource.frontend_provisioner (remote-exec): Using default tag: latest
null_resource.frontend_provisioner: Still creating... [00m50s elapsed]
null_resource.frontend_provisioner (remote-exec): latest: Pulling from dipteshsingh/frontend

null_resource.frontend_provisioner (remote-exec): 9824c27679d3: Pulling fs layer

null_resource.frontend_provisioner (remote-exec): a5585638209e: Pulling fs layer
```

```
null_resource.frontend_provisioner (remote-exec): 9053b5c14689: Pull complete
null_resource.frontend_provisioner (remote-exec): Digest: sha256:c34e6e0dff94c3665cb336a3f9a5275e43c5b32844041a27e8f516f2f864c412
null_resource.frontend_provisioner (remote-exec): Status: Downloaded newer image for dipteshsingh/frontend:latest
null_resource.frontend_provisioner (remote-exec): docker.io/dipteshsingh/frontend:latest
null_resource.frontend_provisioner (remote-exec): REPOSITORY               TAG       IMAGE ID       CREATED       SIZE
null_resource.frontend_provisioner (remote-exec): dipteshsingh/frontend   latest    ed082bbc33e0   3 hours ago   52.5MB
null_resource.frontend_provisioner (remote-exec): 2093dfb7da0d01270294b5aff98e1ee37e420cccac81344e8899b0c6c8ad3b1a
null_resource.frontend_provisioner (remote-exec): Waiting for frontend service to start...
null_resource.frontend_provisioner (remote-exec):  Frontend service is up and running!
null_resource.frontend_provisioner (remote-exec): Docker version 28.3.3, build 980b856
null_resource.frontend_provisioner (remote-exec): CONTAINER ID   IMAGE                  COMMAND                  CREATED         STATUS                  PORTS
          NAMES
null_resource.frontend_provisioner (remote-exec): 2093dfb7da0d   dipteshsingh/frontend   "/docker-entrypoint.…"   1 second ago   Up Less than a second   0.0.0.0:3000->80/tcp, [::]:3000
->80/tcp   frontend
null_resource.frontend_provisioner (remote-exec): HTTP/1.1 200 OK
null_resource.frontend_provisioner (remote-exec): X-Powered-By: Express
null_resource.frontend_provisioner (remote-exec): Access-Control-Allow-Origin: *
null_resource.frontend_provisioner (remote-exec): Content-Type: application/json; charset=utf-8
null_resource.frontend_provisioner (remote-exec): Content-Length: 32
null_resource.frontend_provisioner (remote-exec): ETag: W/"20-kE1OX//ZQQmoRd7s/KVIwrSC3cM"
null_resource.frontend_provisioner (remote-exec): Date: Thu, 31 Jul 2025 09:30:01 GMT
null_resource.frontend_provisioner (remote-exec): Connection: keep-alive
null_resource.frontend_provisioner (remote-exec): Keep-Alive: timeout=5

null_resource.frontend_provisioner: Creation complete after 55s [id=387496641052291792]

Apply complete! Resources: 25 added, 0 changed, 0 destroyed.

Outputs:

Backend-Private-Ip = "10.0.2.57"
Frontend-Private-Ip = "10.0.1.28"
Frontend-Public-Ip = "13.60.44.51"
Frontend-application-url = "http://13.60.44.51:3000"
```

Stage 3: Test_Solution

3.1 Objective

- Using terraform output save the public DNS or Public IP of the FRONTEND and display it as the stage is executed.
- Using terraform outputs and variables display the exact address with port number for the frontend form application.
- Curl the public DNS or IP to check if the frontend containerized application is working.

3.2 Terraform Outputs

Output.tf →

```
output "Frontend-Public-Ip"{
  value = aws_instance.frontend.public_ip
}

output "Frontend-Private-Ip"{
  value = aws_instance.frontend.private_ip
}

output "Backend-Private-Ip"{
 value = aws_instance.backend.private_ip
}

output "Frontend-application-url" {
 description = "Access URL for the frontend form application"
 value= "http://${aws_instance.frontend.public_ip}:${var.frontend_port}"
}
```

```
Outputs:

Backend-Private-Ip = "10.0.2.57"
Frontend-Private-Ip = "10.0.1.28"
Frontend-Public-Ip = "13.60.44.51"
Frontend-application-url = "http://13.60.44.51:3000"
```

3.3 Automated `curl` Test

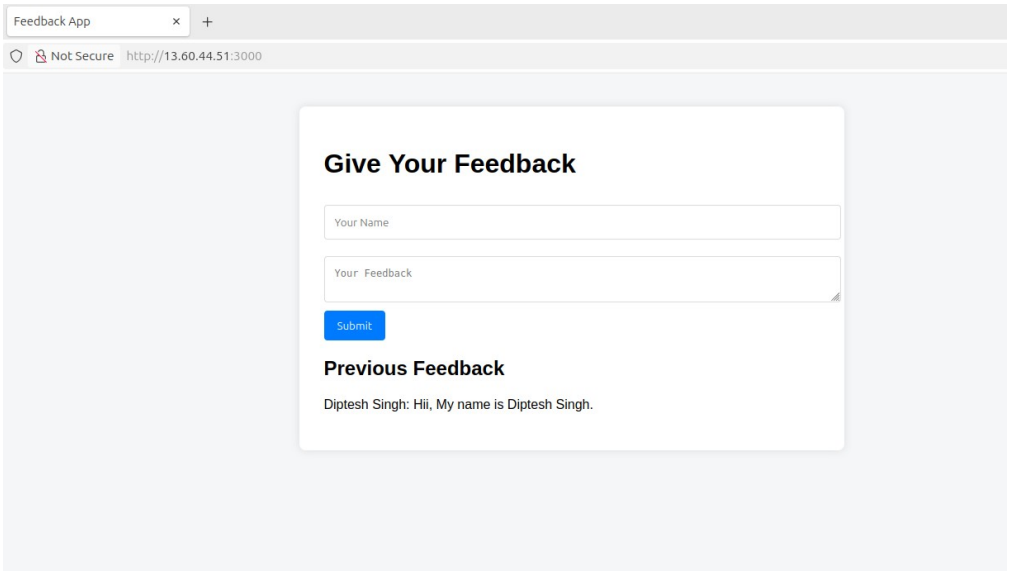
```
ubuntu@ip-10-0-1-28:~$ curl http://localhost:3000
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>Feedback App</title>
  <link rel="stylesheet" href="style.css" />
  </head>
<body>
  <div class="container">
    <h1>Give Your Feedback</h1>
    <form id="feedbackForm">
      <input type="text" id="name" placeholder="Your Name" required />
      <textarea id="feedback" placeholder="Your Feedback" required></textarea>
      <button type="submit">Submit</button>
    </form>
    <h2>Previous Feedback</h2>
    <ul id="feedbackList"></ul>
  </div>
  <script src="script.js"></script>
</body>
</html>
```

Manual `Testing`

- Manually test if the application is running correctly. In the frontend enter some details for an user (Capture a screenshot of it)

- Now dive into the database and check if the line has been added to the database in the BACKEND instance or not (Capture a screenshot of it)

1. **Frontend UI**



2. **Backend DB Verification**

SSH into frontend instance (instance in the public subnet) →

```
diptesh-singh@ASUS-TUF:~/Desktop/Diptesh_Singh_Devops/Project/Create_Infra$ ssh -o StrictHostKeyChecking=no -i frontend.pem ubuntu@13.60.44.51
Warning: Permanently added '13.60.44.51' (ED25519) to the list of known hosts.
Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 6.8.0-1031-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

 System information as of Thu Jul 31 09:34:11 UTC 2025

  System load:  0.0               Processes:             117
  Usage of /:   32.7% of 7.57GB   Users logged in:       0
  Memory usage: 30%               IPv4 address for ens5: 10.0.1.28
  Swap usage:   0%


Expanded Security Maintenance for Applications is not enabled.

22 updates can be applied immediately.
22 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

New release '24.04.2 LTS' available.
Run 'do-release-upgrade' to upgrade to it.


Last login: Thu Jul 31 09:29:20 2025 from 49.37.115.24
```

SSH into backend instance (instance in the private subnet) →

```
ubuntu@ip-10-0-1-28:~$ ssh -o StrictHostKeyChecking=no -i backend.pem ubuntu@10.0.2.57
Warning: Permanently added '10.0.2.57' (ED25519) to the list of known hosts.
Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 6.8.0-1031-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

 System information as of Thu Jul 31 09:34:52 UTC 2025

  System load:  0.0               Processes:             120
  Usage of /:   52.9% of 7.57GB   Users logged in:       0
  Memory usage: 37%               IPv4 address for ens5: 10.0.2.57
  Swap usage:   0%


Expanded Security Maintenance for Applications is not enabled.

22 updates can be applied immediately.
22 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

New release '24.04.2 LTS' available.
Run 'do-release-upgrade' to upgrade to it.


Last login: Thu Jul 31 09:27:04 2025 from 10.0.1.28
ubuntu@ip-10-0-2-57:~$ sudo docker ps
CONTAINER ID   IMAGE                      COMMAND                  CREATED         STATUS         PORTS                                                NAMES
56653a68fc1c   dipteshsingh/backend       "docker-entrypoint.s…"   6 minutes ago   Up 6 minutes   0.0.0.0:5000->5000/tcp, [::]:5000->5000/tcp          backend
11f2cf8d753b   dipteshsingh/feedback-db   "docker-entrypoint.s…"   6 minutes ago   Up 6 minutes   0.0.0.0:5432->5432/tcp, [::]:5432->5432/tcp          db
ubuntu@ip-10-0-2-57:~$ sudo docker exec -it db psql -U postgres -d feedbackdb
psql (17.5 (Debian 17.5-1.pgdg120+1))
Type "help" for help.

feedbackdb=# SELECT * FROM feedbacks;
 id |     name      |            feedback
----+---------------+--------------------------------
  1 | Diptesh Singh | Hii, My name is Diptesh Singh.
(1 row)
```

Containerized Application Link :-

- **Frontend Image:** https://hub.docker.com/r/dipteshsingh/frontend
- **Backend Image:** https://hub.docker.com/r/dipteshsingh/backend
- **Database Image:** https://hub.docker.com/r/dipteshsingh/feedback-db

---

Conclusion

This project successfully demonstrates the deployment of a 2-tier containerized application using Docker and Terraform on AWS. By automating the infrastructure setup with Terraform and managing application containers through Docker, we ensured consistent, repeatable, and scalable deployment. The frontend allowed user interaction, while the backend securely stored data in a PostgreSQL database. Networking, provisioning, and testing were thoroughly implemented and verified, fulfilling all project requirements. This hands-on implementation provided a strong foundation in Infrastructure as Code (IaC), container orchestration, and cloud deployment practices.

End of Report