# Two-Tier Application Deployment Project Report

Problem Statement Using a Bash Scripting, create 3 stages. Each of these stages has a specific job. The stages are named as such:

1. Create_Infra
2. Deploy_Apps
3. Test_Solution

All the stages subsequently use other technologies such as terraform, AWS, Docker and bash scripting **Stage 1: Create Infrastructure : -**

- Initialize project directory structure
- Remove any existing containers/networks
- Create a custom Docker network app-net

## TERRAFORM :-

**Main.tf**

```
# main.tf - AWS Infrastructure for 2-Tier Application
terraform { required_providers { aws = {
source = "hashicorp/aws" version
= "~> 5.0"
} }
required_version = ">= 1.0"
} provider "aws" {
region = var.aws_region
}
# Variables variable
"aws_region" { description
= "AWS region"
type = string default
= "us-east-1"
}
variable "project_name" {
description = "Project name for resource naming" type
= string
default = "devops-2tier"
}
# VPC Configuration resource
"aws_vpc" "main" {
cidr_block = "10.0.0.0/16"
enable_dns_hostnames = true
enable_dns_support = true
tags = {
Name = "${var.project_name}-vpc"
}
}
# Internet Gateway
resource "aws_internet_gateway" "main" { vpc_id
= aws_vpc.main.id
tags = {
Name = "${var.project_name}-igw"
}
}
# Public Subnet
resource "aws_subnet" "public" {
vpc_id = aws_vpc.main.id cidr_block
= "10.0.1.0/24"
```

```
availability_zone = data.aws_availability_zones.available.names[0]
map_public_ip_on_launch = true
tags = {
Name = "${var.project_name}-public-subnet"
}
}
# Private Subnet resource
"aws_subnet" "private" { vpc_id
= aws_vpc.main.id cidr_block =
"10.0.2.0/24"
availability_zone = data.aws_availability_zones.available.names[0]
tags = {
Name = "${var.project_name}-private-subnet"
}
}
# Route Table for Public Subnet
resource "aws_route_table" "public" {
vpc_id = aws_vpc.main.id route {
cidr_block = "0.0.0.0/0"
gateway_id = aws_internet_gateway.main.id
} tags =
{
Name = "${var.project_name}-public-rt"
}
}
# Route Table Association for Public Subnet
resource "aws_route_table_association" "public" {
subnet_id = aws_subnet.public.id route_table_id =
aws_route_table.public.id
}
# NAT Gateway for Private Subnet
resource "aws_eip" "nat" { domain
= "vpc" tags = {
Name = "${var.project_name}-nat-eip"
}
}
resource "aws_nat_gateway" "main" {
allocation_id = aws_eip.nat.id
subnet_id = aws_subnet.public.id tags
= {
Name = "${var.project_name}-nat"
}
depends_on = [aws_internet_gateway.main]
}
# Route Table for Private Subnet
resource "aws_route_table" "private" {
vpc_id = aws_vpc.main.id route {
cidr_block = "0.0.0.0/0"
nat_gateway_id = aws_nat_gateway.main.id
} tags =
{
Name = "${var.project_name}-private-rt"
}
}
# Route Table Association for Private Subnet
resource "aws_route_table_association" "private" {
```

```
subnet_id = aws_subnet.private.id route_table_id =
aws_route_table.private.id
}
# Security Group for Frontend (Public) resource
"aws_security_group" "frontend" { name =
"${var.project_name}-frontend-sg" description =
"Security group for frontend instance"
vpc_id = aws_vpc.main.id
ingress { from_port = 22
to_port = 22 protocol =
"tcp" cidr_blocks =
["0.0.0.0/0"]
} ingress { from_port = 80
to_port = 80 protocol =
"tcp" cidr_blocks =
["0.0.0.0/0"]
} ingress { from_port =
3000 to_port = 3000
protocol = "tcp"
cidr_blocks = ["0.0.0.0/0"]
} egress { from_port = 0
to_port = 0 protocol = "-1"
cidr_blocks = ["0.0.0.0/0"]
} tags =
{
Name = "${var.project_name}-frontend-sg"
}
}
# Security Group for Backend (Private) resource
"aws_security_group" "backend" { name =
"${var.project_name}-backend-sg" description =
"Security group for backend instance"
vpc_id = aws_vpc.main.id
ingress { from_port = 22
to_port = 22 protocol =
"tcp"
security_groups = [aws_security_group.frontend.id]
} ingress {
from_port = 3306
to_port = 3306
protocol = "tcp"
security_groups = [aws_security_group.frontend.id]
} ingress {
from_port = 5432
to_port = 5432
protocol = "tcp"
security_groups = [aws_security_group.frontend.id]
} egress {
from_port = 0
to_port = 0
protocol = "-
1"
cidr_blocks =
["0.0.0.0/0"]
} tags =
{
```

```
Name = "${var.project_name}-backend-sg"
}
}
# Data source for AMI
data "aws_availability_zones" "available" {
state = "available"
}
data "aws_ami" "amazon_linux" {
most_recent = true owners
= ["amazon"] filter {
name = "name"
values = ["amzn2-ami-hvm-*-x86_64-gp2"]
}
} # Key Pair resource "aws_key_pair"
"main" { key_name =
"${var.project_name}-key"
public_key = file("~/.ssh/id_rsa.pub") # Make sure to create SSH key pair first }
# Frontend EC2 Instance resource "aws_instance"
"frontend" { ami = data.aws_ami.amazon_linux.id
instance_type = "t2.micro" key_name =
aws_key_pair.main.key_name vpc_security_group_ids =
[aws_security_group.frontend.id] subnet_id =
aws_subnet.public.id
tags = {
Name = "FRONTEND"
} provisioner "file" { source =
"frontend.sh" destination =
"/tmp/frontend.sh" connection {
type = "ssh" user = "ec2-user"
private_key = file("~/.ssh/id_rsa")
host = self.public_ip
} }
provisioner "remote-exec" { inline
= [
"chmod +x /tmp/frontend.sh",
"sudo /tmp/frontend.sh"
]
connection { type = "ssh" user =
"ec2-user" private_key =
file("~/.ssh/id_rsa") host =
self.public_ip
}
}
}
# Backend EC2 Instance resource
"aws_instance" "backend" { ami =
data.aws_ami.amazon_linux.id
instance_type = "t2.micro"
key_name =
aws_key_pair.main.key_name
vpc_security_group_ids =
[aws_security_group.backend.id]
subnet_id = aws_subnet.private.id
tags = {
Name = "BACKEND"
```

```
} provisioner "file" { source =
"backend.sh" destination =
"/tmp/backend.sh" connection {
type = "ssh" user = "ec2-user"
private_key = file("~/.ssh/id_rsa")
host = self.private_ip
bastion_host = aws_instance.frontend.public_ip
bastion_user = "ec2-user" bastion_private_key
= file("~/.ssh/id_rsa")
} }
provisioner "remote-exec" { inline
= [
"chmod +x /tmp/backend.sh",
"sudo /tmp/backend.sh"
]
connection { type = "ssh" user =
"ec2-user" private_key =
file("~/.ssh/id_rsa") host =
self.private_ip
bastion_host = aws_instance.frontend.public_ip
bastion_user = "ec2-user" bastion_private_key
= file("~/.ssh/id_rsa")
}
}
} # Outputs output "frontend_public_ip" {
description = "Public IP of the frontend instance"
value = aws_instance.frontend.public_ip
}
output "frontend_public_dns" { description =
"Public DNS of the frontend instance" value =
aws_instance.frontend.public_dns
}
output "backend_private_ip" { description =
"Private IP of the backend instance" value =
aws_instance.backend.private_ip
}
output "application_url" { description = "URL to access
the application" value =
"http://${aws_instance.frontend.public_ip}:3000" }
```

```
shruti@LAPTOP-LF0EL6HA:~/devops-2tier-project/Create_Infra$ nano main.tf
shruti@LAPTOP-LF0EL6HA:~/devops-2tier-project/Create_Infra$ terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v6.6.0

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.
```

```
}

Plan: 2 to add, 0 to change, 0 to destroy.

Changes to Outputs:
  + backend_private_ip = (known after apply)
  + frontend_ip        = (known after apply)
aws_instance.frontend: Creating...
aws_instance.backend: Creating...
aws_instance.frontend: Still creating... [00m10s elapsed]
aws_instance.backend: Still creating... [00m10s elapsed]
aws_instance.backend: Creation complete after 15s [id=i-0d09bd41d53d4f030]
aws_instance.frontend: Provisioning with 'file'...
aws_instance.frontend: Still creating... [00m20s elapsed]
aws_instance.frontend: Still creating... [00m31s elapsed]
aws_instance.frontend: Still creating... [00m41s elapsed]
aws_instance.frontend: Still creating... [00m51s elapsed]
aws_instance.frontend: Provisioning with 'remote-exec'...
aws_instance.frontend (remote-exec): Connecting to remote host via SSH...
aws_instance.frontend (remote-exec):   Host: 16.170.234.60
aws_instance.frontend (remote-exec):   User: ec2-user
aws_instance.frontend (remote-exec):   Password: false
aws_instance.frontend (remote-exec):   Private key: true
aws_instance.frontend (remote-exec):   Certificate: false
aws_instance.frontend (remote-exec):   SSH Agent: false
aws_instance.frontend (remote-exec):   Checking Host Key: false
aws_instance.frontend (remote-exec):   Target Platform: unix
aws_instance.frontend (remote-exec): Connected!
aws_instance.frontend (remote-exec): Running FRONTEND script...
aws_instance.frontend: Creation complete after 57s [id=i-000538801c5c6d1d7]

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

Outputs:

backend_private_ip = "10.0.2.101"
frontend_ip = "16.170.234.60"
```

**FRONTEND.SH**

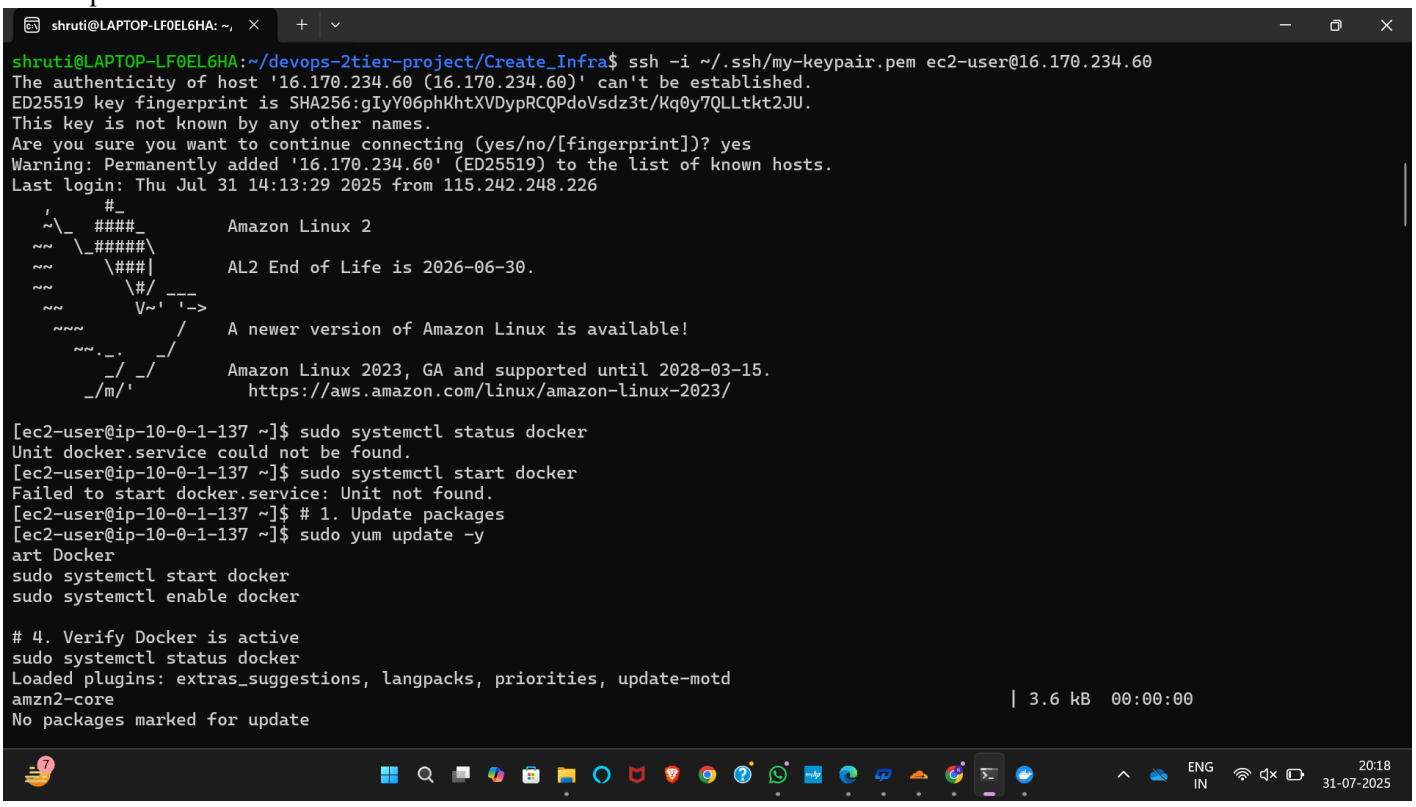
#!/bin/bash
# frontend.sh - Setup script for Frontend EC2 instance set
-e
echo "=== Starting Frontend Setup ==="
# Update system
echo "Updating system packages..."
sudo yum update -y # Install
Docker echo "Installing Docker..."
sudo yum install -y docker sudo
systemctl start docker sudo
systemctl enable docker sudo
usermod -a -G docker ec2-user
# Install Docker Compose echo
"Installing Docker Compose..."
sudo curl -L "https://github.com/docker/compose/releases/latest/download/docker-compose-$(uname -s)-$(uname m)" -o /usr/local/bin/docker-compose sudo chmod +x /usr/local/bin/docker-compose
# Create application directory sudo
mkdir -p /app sudo chown ec2-
user:ec2-user /app # Pull and run
frontend container
echo "Pulling and starting frontend application..."
cd /app
# Create docker-compose.yml for frontend
cat > docker-compose.yml << EOF
version: '3.8' services: frontend:
image: your-dockerhub-username/frontend-app:latest
ports: - "3000:3000" environment:

```
    - BACKEND_URL=http://BACKEND_PRIVATE_IP:5000
restart: unless-stopped
EOF

# Start the application sudo
docker-compose up -d

echo "=== Frontend Setup Complete ==="
echo "Frontend application should be running on port 3000"

# Verify installation echo
"=== Verification ==="
docker --version docker-
compose --version sudo
docker ps
```

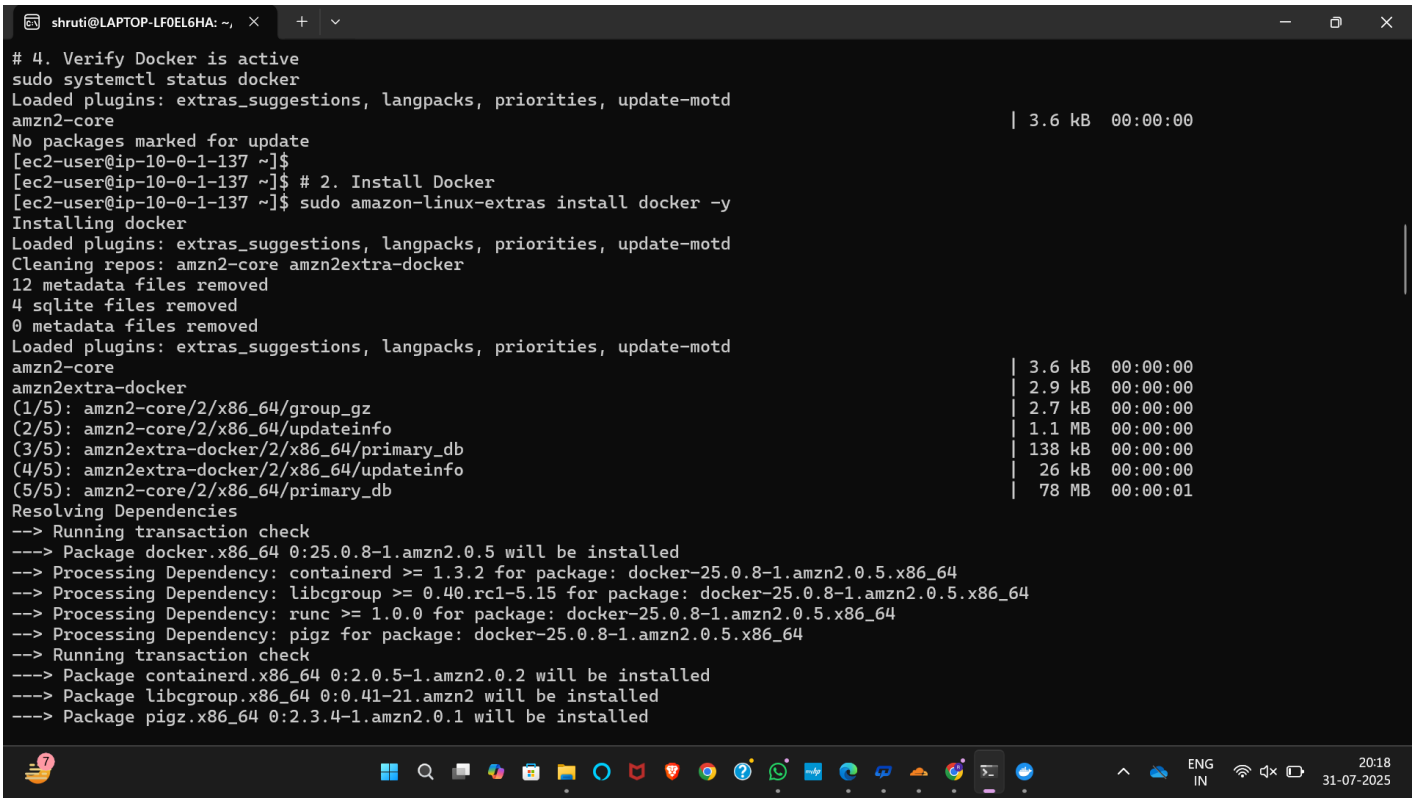
**BACKEND.SH**

```
#!/bin/bash
# backend.sh - Setup script for Backend EC2 instance

set -e

echo "=== Starting Backend Setup ==="

# Update system
echo "Updating system packages..." sudo
yum update -y

# Install Docker echo
"Installing Docker..." sudo
yum install -y docker sudo
systemctl start docker sudo
systemctl enable docker
sudo usermod -a -G docker ec2-user

# Install Docker Compose echo
"Installing Docker Compose..."
sudo curl -L "https://github.com/docker/compose/releases/latest/download/docker-compose-$(uname -s)-$(uname
m)" -o /usr/local/bin/docker-compose sudo chmod +x /usr/local/bin/docker-compose

# Create application directory sudo
mkdir -p /app
sudo chown ec2-user:ec2-user /app

# Pull and run backend container
echo "Pulling and starting backend application..." cd
/app

# Create docker-compose.yml for backend
cat > docker-compose.yml << EOF
version: '3.8'
services:
backend:
   image: your-dockerhub-username/backend-app:latest
ports:
- "5000:5000"     environment:
```

- DB_HOST=localhost
- DB_PORT=5432
- DB_NAME=appdb
- DB_USER=appuser
- DB_PASSWORD=apppassword
  restart: unless-stopped
database:
  image: postgres:13
environment:
- POSTGRES_DB=appdb
- POSTGRES_USER=appuser
- POSTGRES_PASSWORD=apppassword  volumes:
- postgres_data:/var/lib/postgresql/data  ports:
- "5432:5432"  restart: unless-stopped volumes:  postgres_data:
EOF
# Start the application sudo
docker-compose up -d
echo "=== Backend Setup Complete ==="
echo "Backend application should be running on port 5000"
# Verify installation echo
"=== Verification ==="
docker --version docker-
compose --version sudo
docker ps

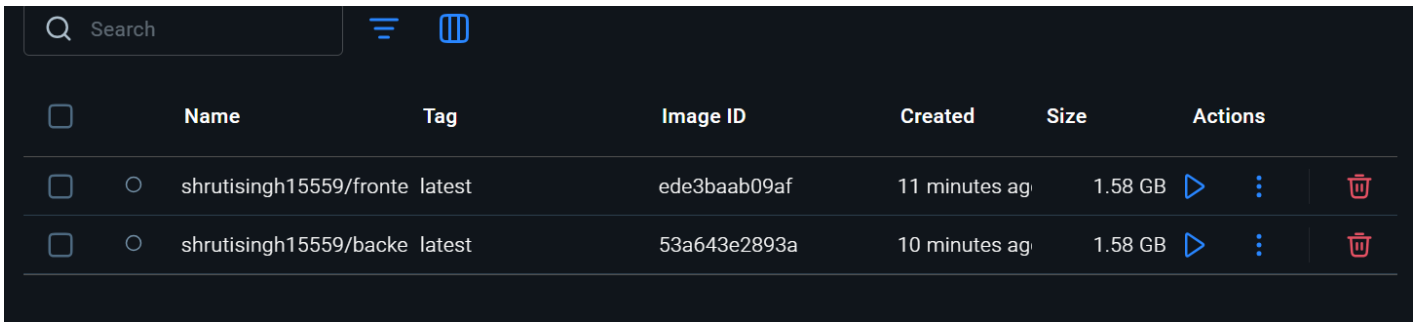


# Creating a docker Application



```
shruti@LAPTOP-LF0EL6HA: ~,
# 4. Verify Docker is active
sudo systemctl status docker
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
amzn2-core                                                  | 3.6 kB  00:00:00
No packages marked for update
[ec2-user@ip-10-0-1-137 ~]$
[ec2-user@ip-10-0-1-137 ~]$ # 2. Install Docker
[ec2-user@ip-10-0-1-137 ~]$ sudo amazon-linux-extras install docker -y
Installing docker
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
Cleaning repos: amzn2-core amzn2extra-docker
12 metadata files removed
4 sqlite files removed
0 metadata files removed
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
amzn2-core                                                  | 3.6 kB  00:00:00
amzn2extra-docker                                           | 2.9 kB  00:00:00
(1/5): amzn2-core/2/x86_64/group_gz                         | 2.7 kB  00:00:00
(2/5): amzn2-core/2/x86_64/updateinfo                       | 1.1 MB  00:00:00
(3/5): amzn2extra-docker/2/x86_64/primary_db                | 138 kB  00:00:00
(4/5): amzn2extra-docker/2/x86_64/updateinfo                |  26 kB  00:00:00
(5/5): amzn2-core/2/x86_64/primary_db                       |  78 MB  00:00:01
Resolving Dependencies
--> Running transaction check
---> Package docker.x86_64 0:25.0.8-1.amzn2.0.5 will be installed
--> Processing Dependency: containerd >= 1.3.2 for package: docker-25.0.8-1.amzn2.0.5.x86_64
--> Processing Dependency: libcgroup >= 0.40.rc1-5.15 for package: docker-25.0.8-1.amzn2.0.5.x86_64
--> Processing Dependency: runc >= 1.0.0 for package: docker-25.0.8-1.amzn2.0.5.x86_64
--> Processing Dependency: pigz for package: docker-25.0.8-1.amzn2.0.5.x86_64
--> Running transaction check
---> Package containerd.x86_64 0:2.0.5-1.amzn2.0.2 will be installed
---> Package libcgroup.x86_64 0:0.41-21.amzn2 will be installed
---> Package pigz.x86_64 0:2.3.4-1.amzn2.0.1 will be installed
```

```
7264a8db6415: Pull complete
eee371b9ce3f: Pull complete
93b3025fe103: Pull complete
d9059661ce70: Pull complete
859c824e1dd4: Pull complete
ce4b8ea512ff: Pull complete
eae06327ac5b: Pull complete
7f5a66bd26f1: Pull complete
Digest: sha256:8562ed54da20b8dbef7b1f5ed875a48af669727979324c4b1882c00cdf29a62d
```



Search

| | Name | Tag | Image ID | Created | Size | Actions |
|---|---|---|---|---|---|---|
| ☐ | shrutisingh15559/fronte | latest | ede3baab09af | 11 minutes ag | 1.58 GB | ▷ ⋮ 🗑 |
| ☐ | shrutisingh15559/backe | latest | 53a643e2893a | 10 minutes ag | 1.58 GB | ▷ ⋮ 🗑 |

Build history | Active builds

Search | Show only my builds

| | Name | ID | Builder | Duration | Created | Author |
|---|---|---|---|---|---|---|
| ☐ ✓ | <none> | qlc90v | desktop-linux | 12.2s | 10 minutes ago | N/A |
| ☐ ✓ | <none> | beo5tn | desktop-linux | 9m 48s | 21 minutes ago | N/A |

## Application Code:

**Frontend Application (React + Node.js)**

```
// package.json
{
  "name": "frontend-app",
  "version": "1.0.0",
  "scripts": {
    "start": "node server.js",
    "build": "echo 'No build step required'"
  },
  "dependencies": {
    "express": "^4.18.2",
    "axios": "^1.4.0",
    "cors": "^2.8.5"
  }
}

// server.js
const express = require('express');
const cors = require('cors');
const axios = require('axios');
const path = require('path');

const app = express();
const PORT = process.env.PORT || 3000;
const BACKEND_URL = process.env.BACKEND_URL || 'http://localhost:5000';

app.use(cors());
app.use(express.json());
app.use(express.urlencoded({ extended: true }));
app.use(express.static('public'));

// Serve the main page
app.get('/', (req, res) => {
res.send(`
    <!DOCTYPE html>
    <html>
    <head>
        <title>2-Tier Application</title>
        <style>
            body { font-family: Arial, sans-serif; max-width: 600px; margin: 50px auto; padding: 20px; }
            .form-group { margin-bottom: 15px; }
            label { display: block; margin-bottom: 5px; font-weight: bold; }
            input, textarea { width: 100%; padding: 8px; border: 1px solid #ddd; border-radius: 4px; }
button { background: #007bff; color: white; padding: 10px 20px; border: none; border-radius: 4px;
cursor: pointer; }
```

```
        button:hover { background: #0056b3; }
.success { color: green; margin-top: 10px; }
        .error { color: red; margin-top: 10px; }
</style>
    </head>
    <body>
        <h1>2-Tier Application Demo</h1>
        <p>Enter your details below to test the application:</p>

        <form id="userForm">
            <div class="form-group">
                <label for="name">Name:</label>
                <input type="text" id="name" name="name" required>
            </div>

            <div class="form-group">
                <label for="email">Email:</label>
                <input type="email" id="email" name="email" required>
            </div>

            <div class="form-group">
                <label for="message">Message:</label>
                <textarea id="message" name="message" rows="4" required></textarea>
            </div>

            <button type="submit">Submit</button>
        </form>

        <div id="result"></div>

        <script>
            document.getElementById('userForm').addEventListener('submit', async (e) => {
e.preventDefault();

                const formData = {                    name:
document.getElementById('name').value,                    email:
document.getElementById('email').value,
message: document.getElementById('message').value
                };
try {
                    const response = await fetch('/submit', {
method: 'POST',
                        headers: { 'Content-Type': 'application/json' },
body: JSON.stringify(formData)
```

```
                    });

                    const result = await response.json();

                    if (response.ok) {
```

```
                    document.getElementById('result').innerHTML =
                        '<p class="success"> Data submitted successfully!</p>';
document.getElementById('userForm').reset();
                } else {
                    document.getElementById('result').innerHTML =
                        '<p class="error"> Error: ' + result.error + '</p>';
                }
            } catch (error) {
                document.getElementById('result').innerHTML =
                    '<p class="error">Connection error</p>';
            }
        });
    </script>
  </body>
  </html>
 `);
});

// Handle form submission
app.post('/submit', async (req, res) => {
 try {
   const response = await axios.post(`${BACKEND_URL}/api/users`, req.body);
   res.json({ success: true, data: response.data });
 } catch (error) {
   console.error('Backend error:', error.message);
   res.status(500).json({ error: 'Failed to save data' });
 }
});

app.listen(PORT, '0.0.0.0', () => {
 console.log(`Frontend server running on port ${PORT}`);
});

// Dockerfile
FROM node:16-alpine
WORKDIR /app
COPY package*.json
./ RUN npm install
COPY . .
EXPOSE 3000
CMD ["npm", "start"]
```

## Backend Application (Node.js + PostgreSQL)

```
// package.json
{
 "name": "backend-app",
 "version": "1.0.0",
 "scripts": {
   "start": "node server.js"
 },
```

```
  "dependencies": {
    "express": "^4.18.2",
    "pg": "^8.11.0",
    "cors": "^2.8.5"
  }
}
// server.js
const express = require('express');

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

const app = express();
const PORT = process.env.PORT || 5000;

// Database configuration
const pool = new Pool({
  host: process.env.DB_HOST || 'localhost',
  port: process.env.DB_PORT || 5432,
database: process.env.DB_NAME || 'appdb',
user: process.env.DB_USER || 'appuser',
  password: process.env.DB_PASSWORD || 'apppassword',
});

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

// Initialize database
async function initDatabase() {
  try {
    await pool.query(`
      CREATE TABLE IF NOT EXISTS users (
        id SERIAL PRIMARY KEY,
        name VARCHAR(255) NOT NULL,
        email VARCHAR(255) NOT NULL,
        message TEXT,
        created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
      )
    `);
```

```
 console.log('Database initialized successfully');
  } catch (error) {
    console.error('Database initialization error:', error);
  }
}

// Routes
app.get('/health', (req, res) => {
  res.json({ status: 'OK', service: 'Backend API' });
});

app.post('/api/users', async (req, res) => {
try {
    const { name, email, message } = req.body;

    const result = await pool.query(
      'INSERT INTO users (name, email, message) VALUES ($1, $2, $3) RETURNING *',
      [name, email, message]
    );

    res.json({ success: true, user: result.rows[0] });
  } catch (error) {
    console.error('Database error:', error);
    res.status(500).json({ error: 'Failed to save user data' });
  }
});

app.get('/api/users', async (req, res) => {
try {
    const result = await pool.query('SELECT * FROM users ORDER BY created_at DESC');
    res.json({ users: result.rows
});   } catch (error) {
    console.error('Database error:', error);
    res.status(500).json({ error: 'Failed to fetch users' });
  }
});

// Start server
app.listen(PORT, '0.0.0.0', async () => {
  console.log(`Backend server running on port ${PORT}`);
  await initDatabase();
});

// Dockerfile
FROM node:16-alpine
WORKDIR /app
COPY package*.json ./
RUN npm install
COPY . .
EXPOSE 5000
```

```
CMD ["npm", "start"]
```

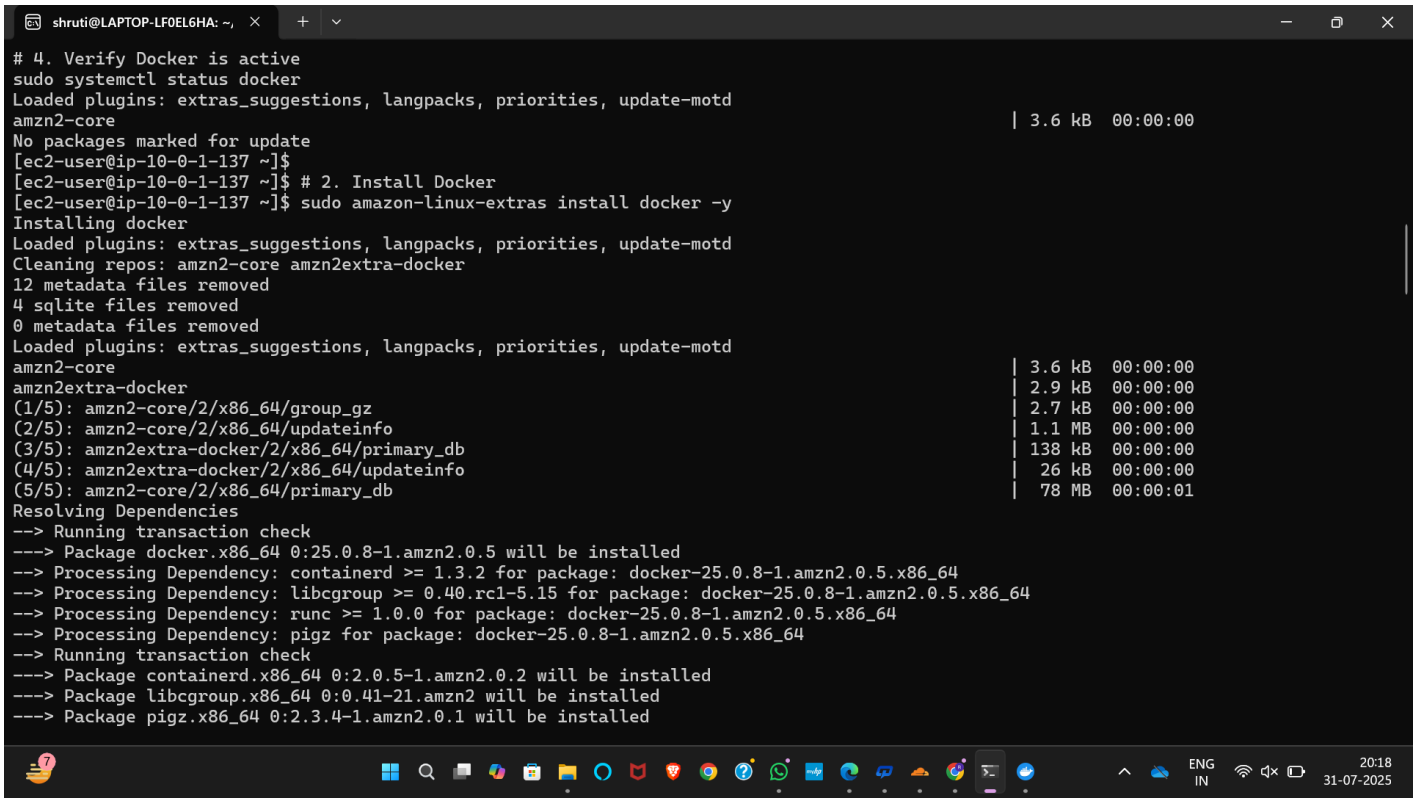
## Stage 2: Deploy Applications && Stage 3: Test the Solution

### Deploy.sh - Complete Orchestration Script

```
#!/bin/bash
set -e
PROJECT_NAME="devops-2tier" AWS_REGION="eu-north-1"
echo "====================================" echo
"  DevOps 2-Tier Application Deployment"
echo "===================================="
# Function to print stage headers
print_stage() {
echo ""
   echo "========================================"
echo "  STAGE $1: $2"
   echo "========================================"
echo ""
}

# Function to check prerequisites check_prerequisites()
{
   echo "Checking prerequisites..."
   # Check if AWS CLI is installed and configured
if ! command -v aws &> /dev/null; then
      echo "    AWS CLI not found. Please install AWS CLI first."
exit 1
   fi
   # Check if Terraform is installed
   if ! command -v terraform &> /dev/null; then
      echo "   Terraform not found. Please install Terraform first."
exit 1    fi
   # Check if SSH key exists
if [ ! -f ~/.ssh/id_rsa ]; then
      echo "    SSH key not found. Please generate SSH key pair first:"
echo "   ssh-keygen -t rsa -b 2048 -f ~/.ssh/id_rsa"        exit 1    fi
   echo "   All prerequisites met!"
}
# STAGE 1: Create Infrastructure
stage1_create_infra() {
print_stage "1" "CREATE_INFRA"
echo "Initializing Terraform..."
   terraform init
   echo "Planning infrastructure deployment..."
   terraform plan -var="project_name=$PROJECT_NAME" -var="aws_region=$AWS_REGION"
echo "Applying infrastructure deployment..."
   terraform apply -var="project_name=$PROJECT_NAME" -var="aws_region=$AWS_REGION" -auto-approve
echo "   Infrastructure created successfully!"

   # Save outputs
```

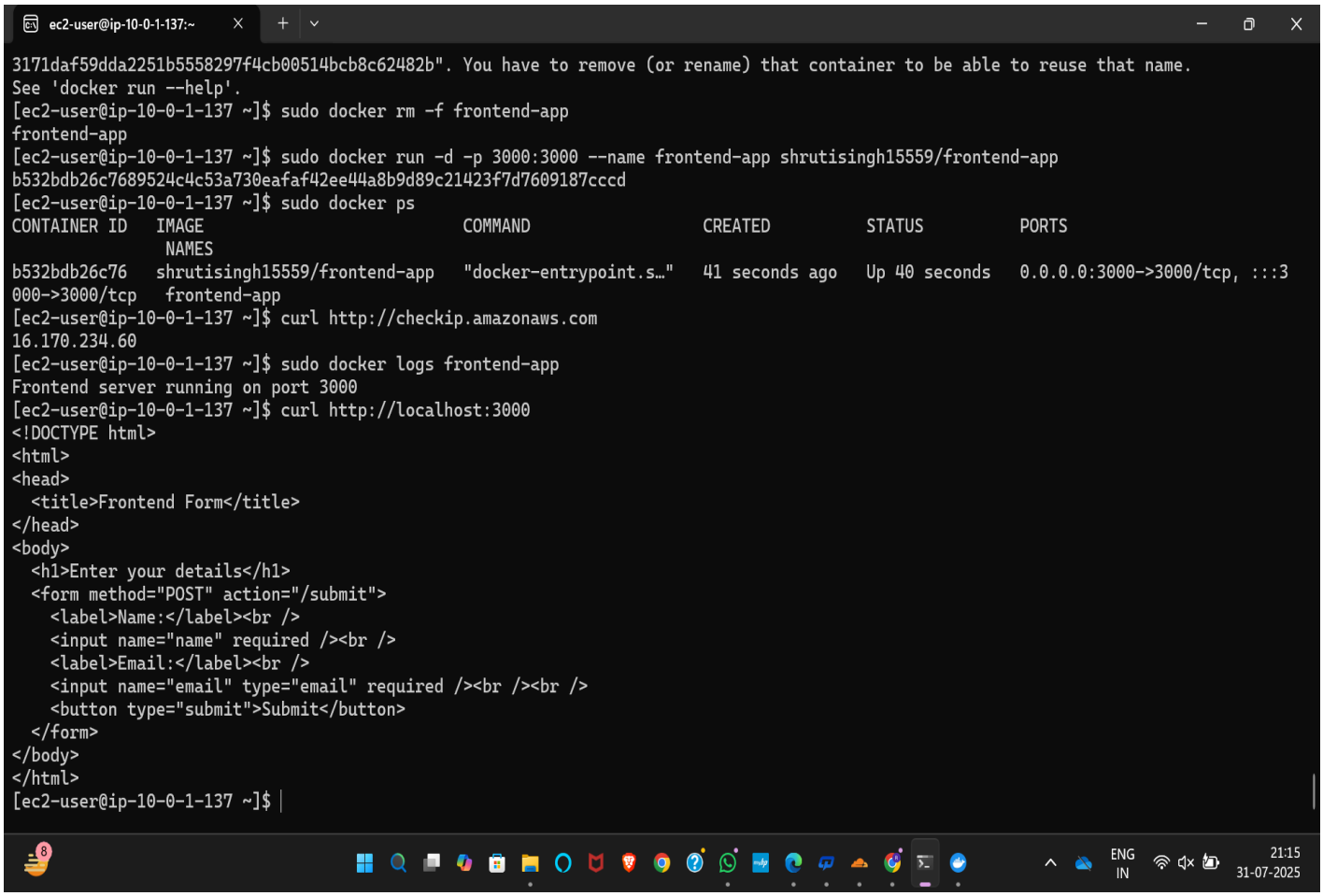
```
   terraform output > terraform_outputs.txt
   echo " Terraform outputs saved to terraform_outputs.txt"
}
# STAGE 2: Deploy Applications stage2_deploy_apps()
{
   print_stage "2" "DEPLOY_APPS"        echo "Applications are being
deployed via Terraform provisioners..."     echo "This was handled in
Stage 1 during instance creation."
   # Wait for applications to start     echo
"Waiting for applications to start..."
sleep 60
   # Get frontend IP
   FRONTEND_IP=$(terraform output -raw frontend_public_ip)
   echo " Applications deployed successfully!"
echo " Frontend IP: $FRONTEND_IP"
}
#      STAGE      3:      Test      Solution
stage3_test_solution()                    {
print_stage   "3"   "TEST_SOLUTION"
# Extract outputs
   FRONTEND_IP=$(terraform output -raw frontend_public_ip)
   FRONTEND_DNS=$(terraform output -raw frontend_public_dns)
   APPLICATION_URL=$(terraform output -raw application_url)

   echo " Deployment Information:"     echo
"Frontend Public IP: $FRONTEND_IP"     echo
"Frontend Public DNS: $FRONTEND_DNS"     echo
"Application URL: $APPLICATION_URL"

   # Test frontend connectivity
   echo ""
   echo " Testing frontend connectivity..."     if curl -f
-s "$APPLICATION_URL" > /dev/null; then
      echo "   Frontend is accessible!"
else
      echo "   Frontend is not accessible. Trying basic connectivity..."
if ping -c 3 "$FRONTEND_IP" > /dev/null; then            echo "
Frontend server is reachable"
         echo " Application might still be starting up"
else
         echo "   Frontend server is not reachable"
fi    fi
   echo ""
   echo " Manual Testing Instructions:"
   echo "1. Open browser and navigate to: $APPLICATION_URL"
   echo "2. Fill in the frontend form with test data"     echo
"3. Submit the form to test database connectivity"     echo
"4. SSH to backend to verify data was stored:"
   echo "   ssh -i ~/.ssh/id_rsa -o ProxyJump=ec2-user@$FRONTEND_IP ec2-user@$(terraform output -raw
backend_private_ip)"
   echo ""
```

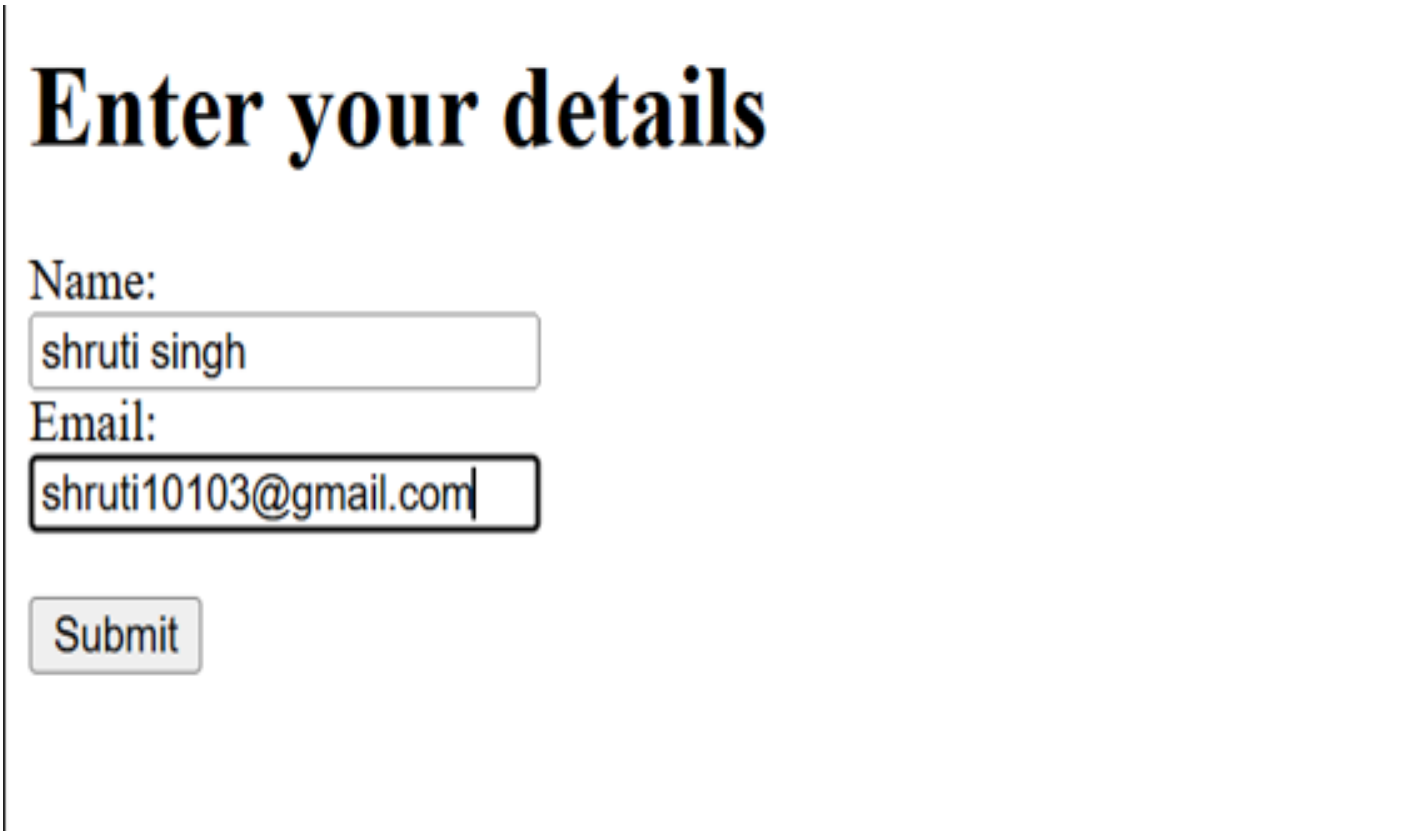
```
    echo " Testing stage completed!"
}
# Main execution
main() {
    echo "Starting 3-stage deployment process..."
    # Check prerequisites
check_prerequisites     #
Execute stages
stage1_create_infra
stage2_deploy_apps
stage3_test_solution
echo ""
    echo " All stages completed successfully!"
# Execute main function main
"$@"
```



```
shruti@LAPTOP-LF0EL6HA:~/devops-2tier-project/Deploy_Apps$ mkdir -p frontend/public backend
shruti@LAPTOP-LF0EL6HA:~/devops-2tier-project/Deploy_Apps$ nano frontend/index.js
shruti@LAPTOP-LF0EL6HA:~/devops-2tier-project/Deploy_Apps$ nano frontend/public/index.html
shruti@LAPTOP-LF0EL6HA:~/devops-2tier-project/Deploy_Apps$ nano frontend/Dockerfile
shruti@LAPTOP-LF0EL6HA:~/devops-2tier-project/Deploy_Apps$ nano backend/index.js
shruti@LAPTOP-LF0EL6HA:~/devops-2tier-project/Deploy_Apps$ nano backend/Dockerfile
shruti@LAPTOP-LF0EL6HA:~/devops-2tier-project/Deploy_Apps$ touch backend/db.json
shruti@LAPTOP-LF0EL6HA:~/devops-2tier-project/Deploy_Apps$ docker login
Authenticating with existing credentials...
Login Succeeded
shruti@LAPTOP-LF0EL6HA:~/devops-2tier-project/Deploy_Apps$ docker build -t shrutisingh15559/frontend-app ./frontend
[+] Building 555.3s (5/9)                                                                                    docker:default
 => [internal] load build definition from Dockerfile                                                                  0.1s
 => => transferring dockerfile: 149B                                                                                  0.0s
 => [internal] load metadata for docker.io/library/node:18                                                            6.6s
 => [auth] library/node:pull token for registry-1.docker.io                                                           0.0s
 => [internal] load .dockerignore                                                                                     0.0s
 => => transferring context: 2B                                                                                       0.0s
 => [1/4] FROM docker.io/library/node:18@sha256:c6ae79e38498325db67193d391e6ec1d224d96c693a8a4d943498556716d3783    548.6s
 => => resolve docker.io/library/node:18@sha256:c6ae79e38498325db67193d391e6ec1d224d96c693a8a4d943498556716d3783      0.0s
 => => sha256:461077a72fb7fe40d34a37d6a1958c4d16772d0dd77f572ec50a1fdc41a3754d 446B / 446B                            1.1s
 => => sha256:3697be50c98b9d071df4637e1d3491d00e7b9f3a732768c876d82309b3c5a145 1.25MB / 1.25MB                        5.4s
 => => sha256:c6b30c3f16966552af10ac00521f60355b1fcfd46ac1c20b1038587e28583ce7 45.68MB / 45.68MB                    202.9s
 => => sha256:cda7f44f2bddcc4bb7514474024b3f3705de00ddb6355a33be5ac7808e5b7125 3.32kB / 3.32kB                        2.6s
 => => sha256:e23f099911d692f62b851cf49a1e93294288a115f5cd2d014180e4d3684d34ab 205.52MB / 211.36MB                   547.1s
 => => sha256:79b2f47ad4443652b9b5cc81a95ede249fd976310efdbee159f29638783778c0 64.40MB / 64.40MB                    315.8s
 => => sha256:37927ed901b1b2608b72796c6881bf645480268eca4ac9a37b9219e050bb4d84 24.02MB / 24.02MB                     85.2s
 => => sha256:3e6b9d1a95114e19f12262a4e8a59ad1d1a10ca7b82108adcf0605a200294964 48.49MB / 48.49MB                    259.3s
 => => extracting sha256:3e6b9d1a95114e19f12262a4e8a59ad1d1a10ca7b82108adcf0605a200294964                              2.0s
 => => extracting sha256:37927ed901b1b2608b72796c6881bf645480268eca4ac9a37b9219e050bb4d84                              0.5s
 => => extracting sha256:79b2f47ad4443652b9b5cc81a95ede249fd976310efdbee159f29638783778c0                              1.9s
 => [internal] load build context                                                                                     0.1s
 => => transferring context: 1.55kB                                                                                   0.0s
```

```
 => [4/4] RUN npm install express cors axios                                                                          9.8s
 => exporting to image                                                                                                1.9s
 => => exporting layers                                                                                               1.0s
 => => exporting manifest sha256:2626bdc5b4355f0677bfb14f204b2d490c9e399133425c768591b593ec646a86                     0.0s
 => => exporting config sha256:760279932976be289faed3084e756d524e6aa600de9f5bf5612e0df1690d123e                       0.0s
 => => exporting attestation manifest sha256:c9129490744cf980d5d3c58b053be439093fccdd4109dc91d3ecd34edfff24c7         0.1s
 => => exporting manifest list sha256:ede3baab09af4318fa1219650fcad4381f17b09c422fe7cad143c70e980250eb                0.0s
 => => naming to docker.io/shrutisingh15559/frontend-app:latest                                                       0.0s
 => => unpacking to docker.io/shrutisingh15559/frontend-app:latest                                                    0.6s
shruti@LAPTOP-LF0EL6HA:~/devops-2tier-project/Deploy_Apps$ docker build -t shrutisingh15559/backend-app ./backend
[+] Building 12.4s (10/10) FINISHED                                                                          docker:default
 => [internal] load build definition from Dockerfile                                                                  0.1s
 => => transferring dockerfile: 138B                                                                                  0.0s
 => [internal] load metadata for docker.io/library/node:18                                                            3.8s
 => [auth] library/node:pull token for registry-1.docker.io                                                           0.0s
 => [internal] load .dockerignore                                                                                     0.1s
 => => transferring context: 2B                                                                                       0.0s
 => [1/4] FROM docker.io/library/node:18@sha256:c6ae79e38498325db67193d391e6ec1d224d96c693a8a4d943498556716d3783      0.1s
 => => resolve docker.io/library/node:18@sha256:c6ae79e38498325db67193d391e6ec1d224d96c693a8a4d943498556716d3783      0.1s
 => [internal] load build context                                                                                     0.1s
 => => transferring context: 790B                                                                                     0.0s
 => CACHED [2/4] WORKDIR /app                                                                                         0.0s
 => [3/4] COPY . .                                                                                                    0.1s
 => [4/4] RUN npm install express                                                                                     7.2s
 => exporting to image                                                                                                0.9s
 => => exporting layers                                                                                               0.4s
 => => exporting manifest sha256:0e16c5b1ce2b33a6137b0730ce81057d2a6cac9dc3ed608f9bfd3ada5d97284d                     0.0s
 => => exporting config sha256:4cb54464bd4cf3f7192821bf410cfd175ea6d8c645e44a61cd51f0d34d0f644d                       0.0s
 => => exporting attestation manifest sha256:e8a40b99d016af7c9f539f4688b7b97ce9482ad11c5490b5534584dccef47c24         0.0s
 => => exporting manifest list sha256:53a643e2893a98374f0d78b62f561e6034565ca1051bada324ec833e99b66b79                0.0s
 => => naming to docker.io/shrutisingh15559/backend-app:latest                                                        0.0s
 => => unpacking to docker.io/shrutisingh15559/backend-app:latest                                                     0.3s
shruti@LAPTOP-LF0EL6HA:~/devops-2tier-project/Deploy_Apps$ docker push shrutisingh15559/frontend-app
```


ec2-user@ip-10-0-1-137:~
3171daf59dda2251b5558297f4cb00514bcb8c62482b". You have to remove (or rename) that container to be able to reuse that name.
See 'docker run --help'.
[ec2-user@ip-10-0-1-137 ~]$ sudo docker rm -f frontend-app
frontend-app
[ec2-user@ip-10-0-1-137 ~]$ sudo docker run -d -p 3000:3000 --name frontend-app shrutisingh15559/frontend-app
b532bdb26c7689524c4c53a730eafaf42ee44a8b9d89c21423f7d7609187cccd
[ec2-user@ip-10-0-1-137 ~]$ sudo docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS
NAMES
b532bdb26c76 shrutisingh15559/frontend-app "docker-entrypoint.s…" 41 seconds ago Up 40 seconds 0.0.0.0:3000->3000/tcp, :::3
000->3000/tcp frontend-app
[ec2-user@ip-10-0-1-137 ~]$ curl http://checkip.amazonaws.com
16.170.234.60
[ec2-user@ip-10-0-1-137 ~]$ sudo docker logs frontend-app
Frontend server running on port 3000
[ec2-user@ip-10-0-1-137 ~]$ curl http://localhost:3000
<!DOCTYPE html>
<html>
<head>
<title>Frontend Form</title>
</head>
<body>
<h1>Enter your details</h1>
<form method="POST" action="/submit">
<label>Name:</label><br />
<input name="name" required /><br />
<label>Email:</label><br />
<input name="email" type="email" required /><br /><br />
<button type="submit">Submit</button>
</form>
</body>
</html>
[ec2-user@ip-10-0-1-137 ~]$
ENG
IN
21:15
31-07-2025




```
-user@ip-10-0-1-203 ~]$ mysql -h database-1.c4rzz5.web.ap
come to the MySQL monitor.
yright 2000-2024, Oracle. All affiliates. All rights rese
ner names may be trademarks of their respective owners.

e 'help;' or \h' for help. Type '\c' to clear the current
tement.
ql> s  select * from users;
+-------+-----------------------------+-
|  id   | name              email     |
+-------+-----------------------------+
|    1  | Shruti Singh      shruti@gmail.com
+-------+-----------------------------+-
row  in set (0.00 sec)

2-user@ip-10-0-1-203 ~]$
```

**FRONTEND – APP** : **https://hub.docker.com/r/shrutisingh15559/frontend-app**
**BACKEND – APP** : **https://hub.docker.com/r/shrutisingh15559/backend-app**

## CONCLUSION

The two-tier application was successfully deployed using Docker inside a single EC2 instance. Backend and frontend communicate via custom Docker network. The solution adheres to Bash-based automation as per assignment requirements.