MASTER ASSIGNMENT

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

# STAGE 1: CREATE INFRA

1. Navigate to the directory and create the folder ‘terraform’.
2. Inside the folder create a ‘main. tf’ file.
3. Write the code to create a VPC and 2 subnets – Public and Private inside it.

# Define the provider provider "aws" {

region = "ap-southeast-2" # Change the region as needed

}

# Create a VPC

resource "aws\_vpc" "main" { cidr\_block = "10.0.0.0/16" tags = {

Name = "Project VPC"

}

}

# Create a Public Subnet

resource "aws\_subnet" "public\_subnet" { vpc\_id = aws\_vpc.main.id

cidr\_block = "10.0.1.0/24" map\_public\_ip\_on\_launch = true tags = {

Name = "Public Subnet"

}

}

# Create a Private Subnet

resource "aws\_subnet" "private\_subnet" { vpc\_id = aws\_vpc.main.id

cidr\_block = "10.0.4.0/24" tags = {

Name = "Private Subnet"

}

}

# Create an Internet Gateway

resource "aws\_internet\_gateway" "gw" { vpc\_id = aws\_vpc.main.id

tags = {

Name = "Project VPC IG"

}

}

1/2

# Create a Route Table for the Public Subnet resource "aws\_route\_table" "public\_rt" { vpc\_id = aws\_vpc.main.id

route {

cidr\_block = "0.0.0.0/0"

gateway\_id = aws\_internet\_gateway.gw.id

}

tags = {

Name = "Public Route Table"

}

}

# Associate the Public Subnet with the Public Route Table

resource "aws\_route\_table\_association" "public\_subnet\_asso" {

subnet\_id = aws\_subnet.public\_subnet.id route\_table\_id = aws\_route\_table.public\_rt.id

}

# Create a Route Table for the Private Subnet (without a route to the Internet Gateway) resource "aws\_route\_table" "private\_rt" { vpc\_id = aws\_vpc.main.id

tags = {

Name = "Private Route Table"

}

}

# Associate the Private Subnet with the Private Route Table

resource "aws\_route\_table\_association" "private\_subnet\_asso" {

subnet\_id = aws\_subnet.private\_subnet.id route\_table\_id = aws\_route\_table.private\_rt.id

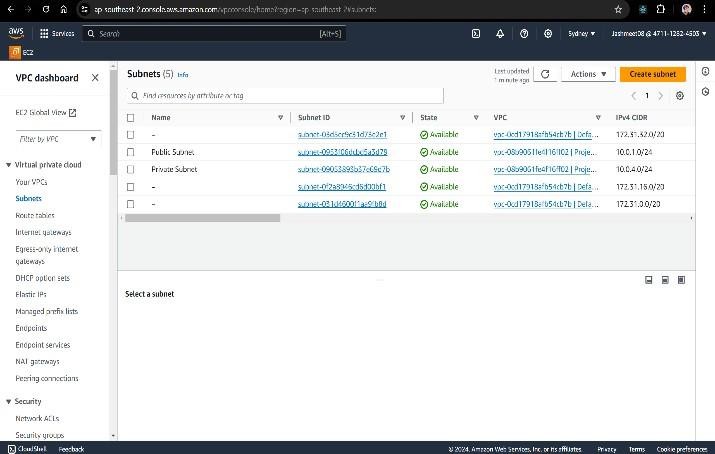
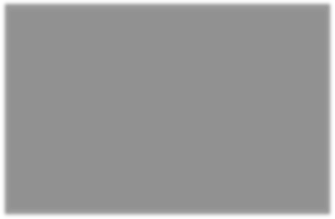
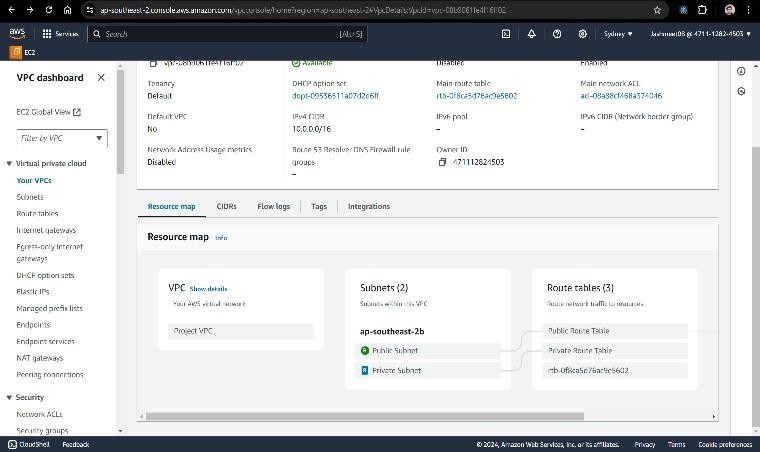
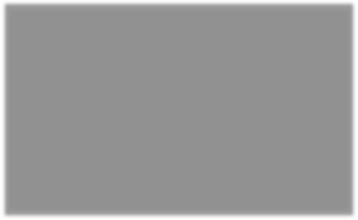
}

2/2

1. Run the following commands –

terraform init terraform plan terraform apply

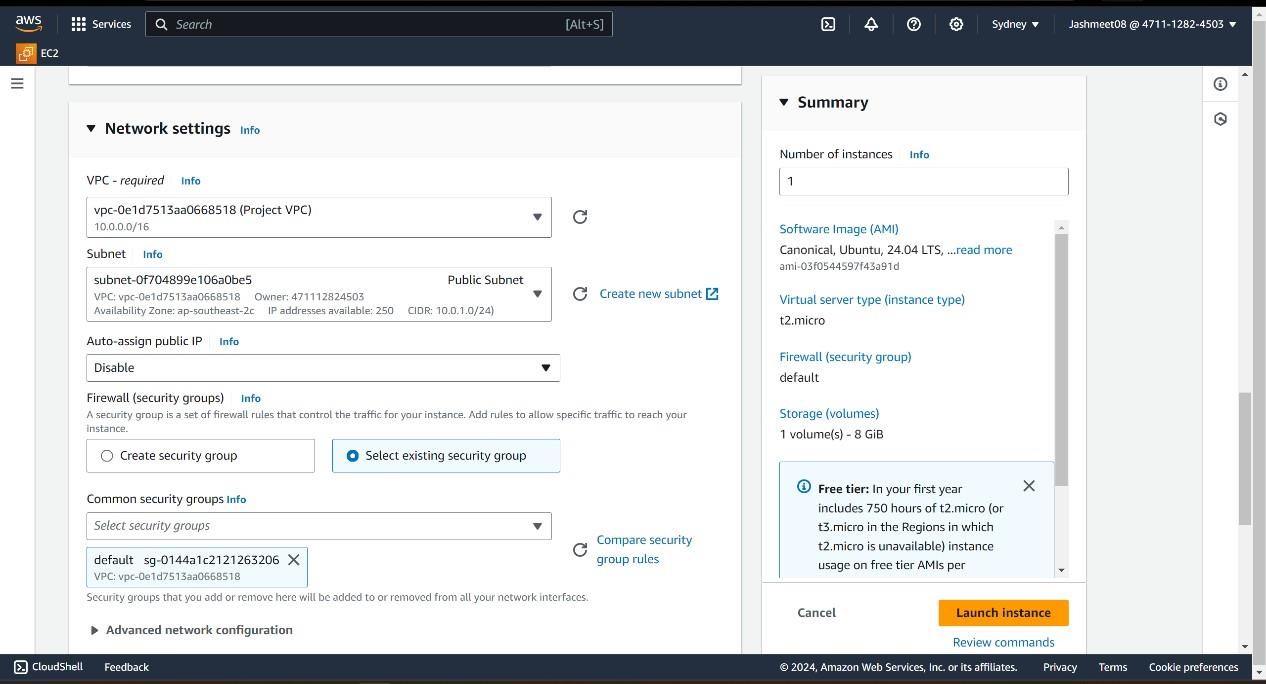
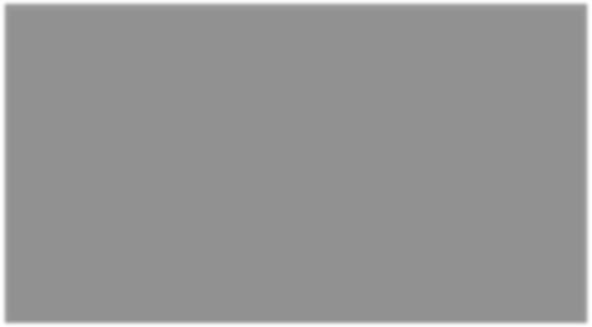
1. Once done, go to the AWS console and check a VPC named “Project VPC” will be created, along with two subnets - Public and Private and two route tables – Public and Private.



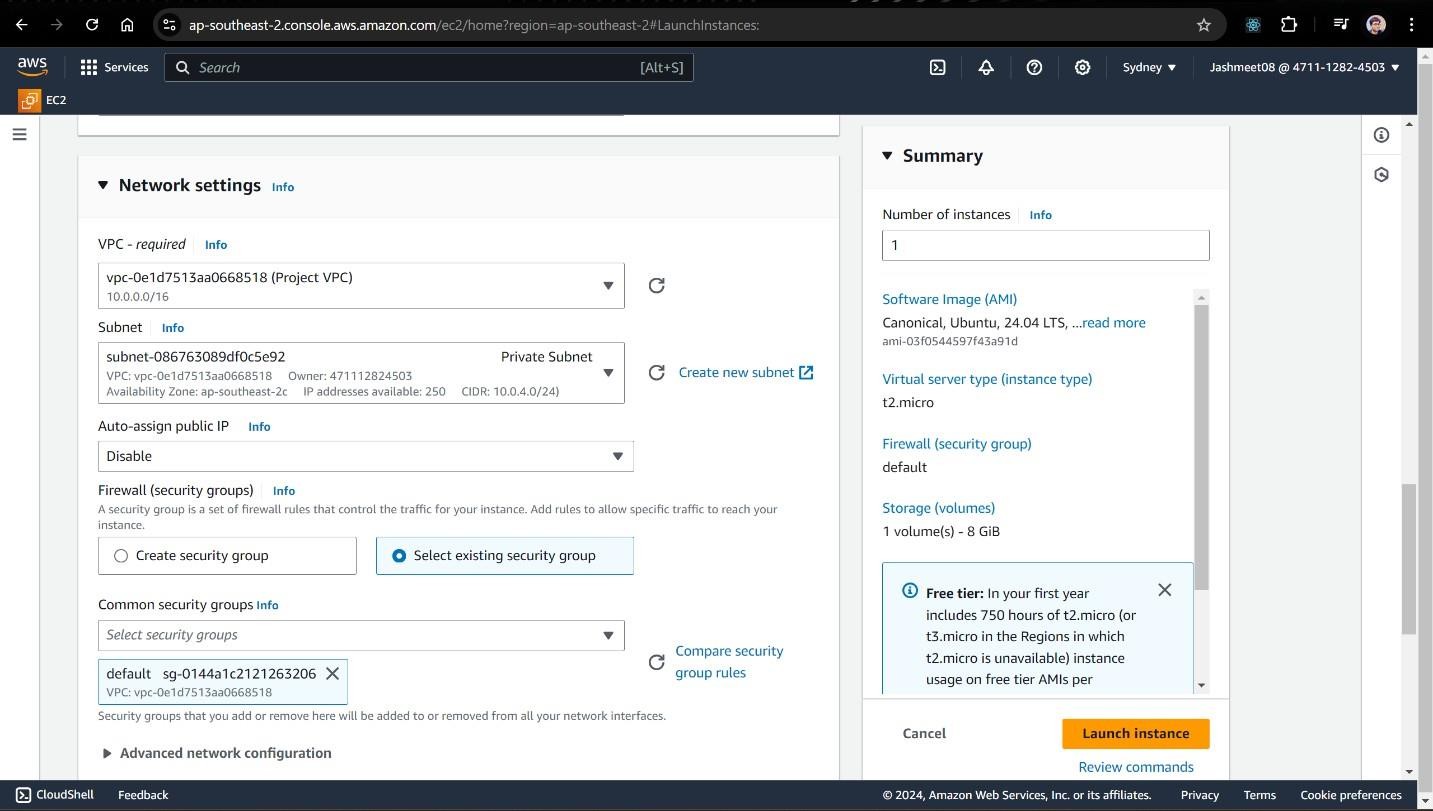
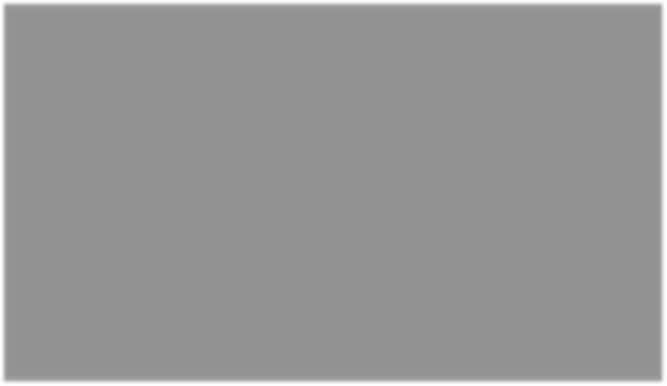
1. Go to EC2 and click Launch Instances – Name it ‘**Frontend**’ > Select ‘**Ubuntu**’ AMI > Select keypair.

In Network settings, click on Edit

* + Under VPC, select ‘Project VPC’
  + Under Subnet, select ‘Public Subnet’
  + Select the existing security group
  + Click on Launch Instances



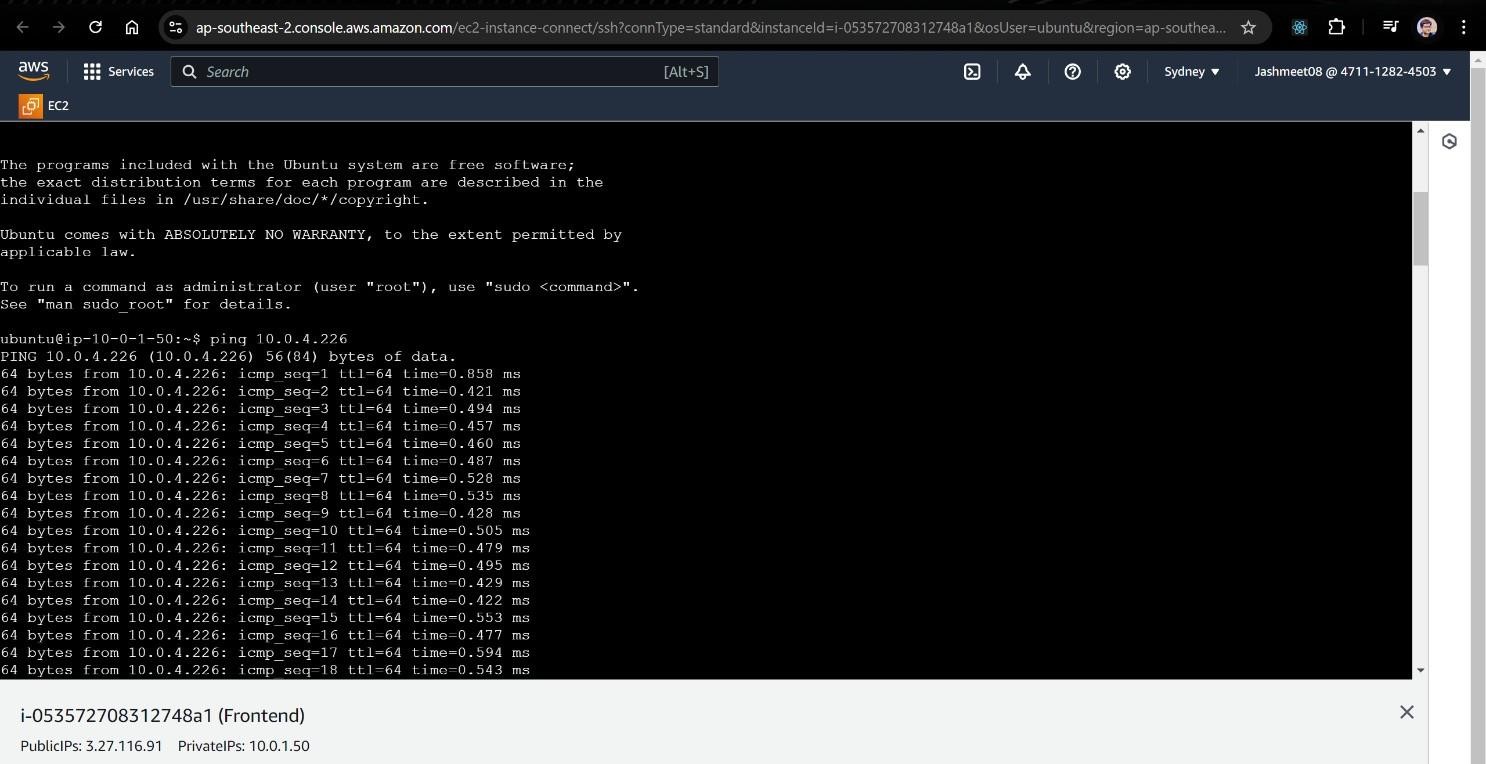
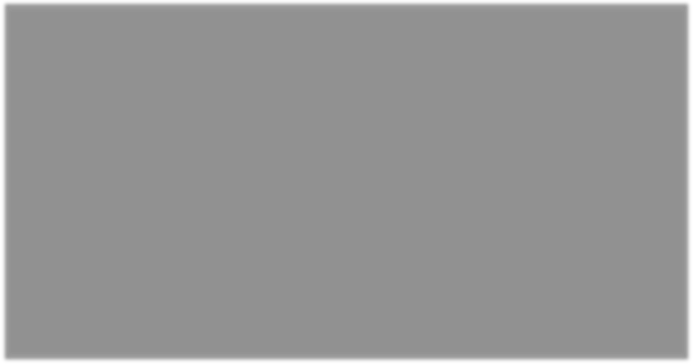
1. Go to EC2 and click Launch Instances – Name it ‘**Backend**’ > Select ‘**Ubuntu**’ AMI > Select keypair. In Network settings, click on Edit
   * Under VPC, select ‘Project VPC’
   * Under Subnet, select ‘Private Subnet’
   * Select the existing security group
   * Click on Launch Instances



1. Go to the Backend Instance, and copy its Private IPv4 address.
2. Go to the Frontend instance SSH it (via EC2 Connect) and type the command

**ping <Backend Private IP>**

1. You will receive a response like this :



This response confirms that **both instances can communicate with each other.** Alternatively, you can also check communication by SSH in the frontend instance with the Public IP of the backend instance.

1. Go to Terminal and create a directory “Terraform-provisioners”, inside it create a script file – **frontend.sh** and

**backend.sh**.

1. Create a file “main.tf” and write the code into it –

provider "aws" {

region = "ap-southeast-2" # Replace with your AWS region

}

# Data block to get information about the existing frontend instance

data "aws\_instance" "frontend" {

instance\_id = "i-0d5900b1d8a43f0b5" # Replace with your frontend instance ID

}

# Data block to get information about the existing backend instance

data "aws\_instance" "backend" {

instance\_id = "i-03f1dbfe0f1b0129e" # Replace with your backend instance ID

}

resource "null\_resource" "frontend" { provisioner "file" {

source = "frontend.sh" destination = "/tmp/frontend.sh"

connection { type = "ssh"

user = "ubuntu" # Replace with your SSH user

private\_key = file("/home/jashmeet/terraform- provisioners/ssh-keys1.pem") # Correct the path to your private key

host = data.aws\_instance.frontend.public\_ip

}

}

provisioner "remote-exec" { inline = [

"chmod +x /tmp/frontend.sh", "/tmp/frontend.sh"

]

connection { type = "ssh"

user = "ubuntu" # Replace with your SSH user

private\_key = file("/home/jashmeet/terraform- provisioners/ssh-keys1.pem") # Correct the path to your private key

host = data.aws\_instance.frontend.public\_ip

}

}

}

resource "null\_resource" "backend" { provisioner "file" {

source = "backend.sh" destination = "/tmp/backend.sh"

connection { type = "ssh"

user = "ubuntu" # Replace with your SSH user private\_key = file("/home/jashmeet/terraform-

provisioners/ssh-keys1.pem") # Correct the path to your private key

host = data.aws\_instance.backend.public\_ip

}

}

provisioner "remote-exec" { inline = [

"chmod +x /tmp/backend.sh", "/tmp/backend.sh"

]

connection { type = "ssh"

user = "ubuntu" # Replace with your SSH user private\_key = file("/home/jashmeet/terraform-

provisioners/ssh-keys1.pem") # Correct the path to your private key

host = data.aws\_instance.backend.public\_ip

}

}

}

**2-TIER APPLICATION IN EC2 (FRONTEND, BACKEND & DATABASE)**

**g**

**D**

1. **CREATE A NEW FOLDER ‘DevOps’**
2. **CREATE ANOTHER FOLDER INSIDE DevOps NAMED AS PUBLIC**
3. **INSIDE PUBLIC – CREATE AN INDEX.HTML FILE.**
4. **INSIDE DevOps FOLDER, CREATE A SERVER.JS FILE.**
5. **GO TO THE AWS CONSOLE, LAUNCH AN EC2 INSTANCE.**
6. **GO TO UBUNTU(LOCAL MACHINE) AND COPY THE FOLDER DevOps to the VM.**

**scp -i ~/DevOps/key-pair.pem -r ~/DevOps ubuntu@\*\*YOUR PUBLIC DNS\*\*:~ ssh -i ~/DevOps/key-pair.pem ubuntu@\*\*YOUR PUBLIC DNS\*\***

INSIDE VM, **ls -l**

1. **IN THE VM, INSTALL NECESSARY PACKAGES**

sudo apt update

sudo apt install nodejs mkdir contact-form cd contact-form

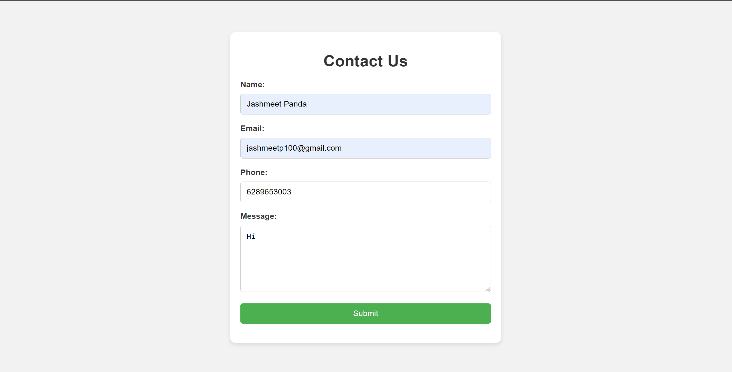
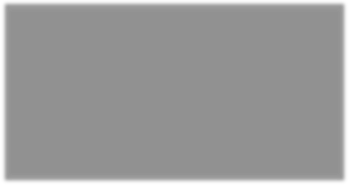
npm init -y

npm install express body-parser sqlite3

1. **GO TO THE TERMINAL, NAVIGATE TO THE DIRECTORY AND TYPE**

sudo apt install sqlite3 node server.js

1. **NAVIGATE TO http://publicdns:3000 IN YOUR BROWSER TO SEE THE FORM. WHEN YOU SUBMIT THE FORM, THE DATA WILL BE SAVED IN THE SQLITE DATABASE.**



1. **ENSURE YOUR PROJECT STRUCTURE IS AS FOLLOWS:**

DevOps > node modules

* Backend > package.json
* Backend > server.js
* Backend > contacts.db
* public –> index.html

**CONTAINERISATION**

1. **GO TO DEVOPS FOLDER > PUBLIC > CREATE A DOCKERFILE AND WRITE**

# Use an official Nginx image as the base image FROM nginx:alpine

# Remove the default Nginx HTML files RUN rm -rf /usr/share/nginx/html/\*

# Copy the HTML file to the Nginx HTML directory COPY index.html /usr/share/nginx/html/

# Expose port 80 EXPOSE 80

# Start Nginx

CMD ["nginx", "-g", "daemon off;"]

1. **NOW BUILD & RUN THE FRONT-END IMAGE USING THE FOLLOWING COMMANDS**

sudo docker build -t public .

sudo docker run -d -p 8080:8080 public

1. **IN TERMINAL, TYPE**

sudo docker login

1. **ENTER YOUR USERID AND PASSWORD OF DOCKERHUB.**
2. **PUSH THE FRONTEND IMAGE TO DOCKER HUB USING THESE COMMANDS.**

sudo docker tag public jashmeet09/frontend docker push jashmeet09/frontend

1. **ONCE PUSHED, SAVE THE PULL COMMAND –**

sudo docker pull jashmeet09/frontend

1. **GO TO DEVOPS > CREATE DOCKERCOMPOSE.YML**

version: '3' services: frontend:

build:

context: ./public dockerfile: Dockerfile ports:

- "8082:80"

backend: build:

context: ./backend dockerfile: Dockerfile ports:

- "3002:3000"

volumes:

- db-data:/usr/src/app

1. **GO TO DEVOPS > BACKEND > CREATE A DOCKER FILE FOR THE BACKEND**

# Use an official Node.js runtime as a parent image FROM node:14

# Set the working directory in the container WORKDIR /usr/src/app

# Copy the package.json and package-lock.json to the working directory COPY package\*.json ./

# Install dependencies RUN npm install

# Copy the rest of the application code to the working directory COPY . .

# Expose the port the app runs on EXPOSE 3000

# Start the application CMD ["node", "server.js"]

1. **NOW BUILD & RUN THE BACKEND IMAGE USING THE FOLLOWING COMMANDS**

sudo docker build -t backend .

sudo docker run -d -p 3000:3000 public

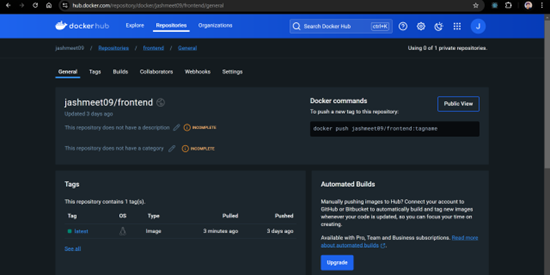
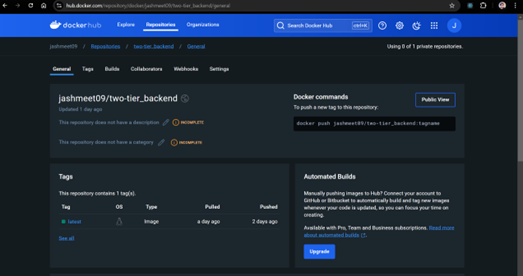
1. **PUSH THE FRONTEND IMAGE TO DOCKER HUB USING THESE COMMANDS.**

sudo docker tag public jashmeet09/two-tier\_backend

docker push jashmeet09/two-tier\_backend

1. **ONCE PUSHED, SAVE THE PULL COMMAND.**

sudo docker pull jashmeet09/two-tier\_backend

**DOCKERHUB AFTER FRONTEND AND BACKEND IMAGES ARE PUSHED**

# STAGE 2: DEPLOY APPS

* 1. **GO TO THE FOLDER TERRAFORM-PROVISIONERS, AND EDIT THE FRONTEND.SH**

#!/bin/sh

sudo apt update

sudo apt install -y docker.io sudo systemctl start docker sudo systemctl enable docker

echo "Pulling the frontend image from dockerhub" sudo docker pull jashmeet09/frontend

echo "Running the front-end image"

sudo docker run -d -p 80:80 jashmeet09/frontend

**BACKEND.SH**

#!/bin/sh

sudo apt update

sudo apt install -y docker.io sudo systemctl start docker sudo systemctl enable docker

echo "Pulling the back-end image from dockerhub" sudo docker pull jashmeet09/two-tier\_backend

echo "Running backend image"

sudo docker run -d -p 3000:3000 jashmeet09/two-tier\_backend

* 1. **GO TO HOME, CREATE A NEW FOLDER TERRAFORM-CHECK DOCKER > MAIN.TF**

provider "aws" {

region = "ap-southeast-2" # Replace with your AWS region

}

data "aws\_instance" "existing\_instance" {

instance\_id = "i-0d5900b1d8a43f0b5" # Replace with your actual instance ID

}

resource "null\_resource" "check\_docker" { depends\_on = [data.aws\_instance.existing\_instance]

provisioner "remote-exec" { inline = [

"usermod -aG docker ubuntu", "sudo systemctl restart docker", # Check if Docker is installed

"if command -v docker >/dev/null 2>&1; then echo 'Docker is installed'; else echo 'Docker is not installed'; fi",

# Check if your application is running

"if docker ps --filter 'ancestor=jashmeet09/frontend' | grep -q 'jashmeet09/frontend'; then echo 'Application is running'; else echo 'Application is not running'; fi"

]

connection { type = "ssh"

user = "ubuntu" # Replace with your SSH user if different

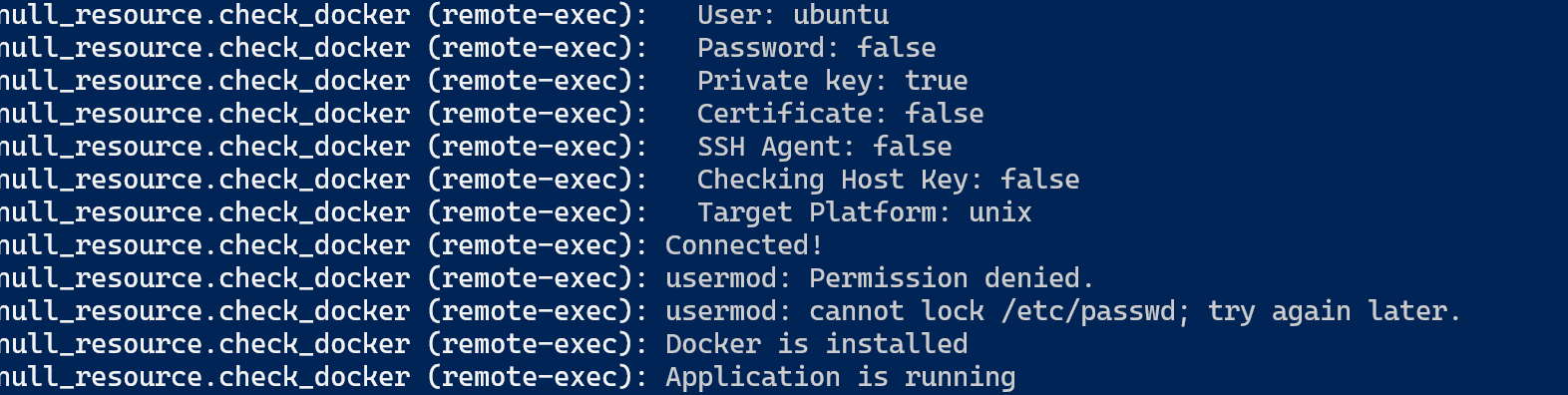
private\_key = file("/home/jashmeet/terraform-provisioners/ssh-keys1.pem") # Replace with the path to your private key

host = data.aws\_instance.existing\_instance.public\_ip

}

}

}



**TERRAFORM CHECKS IF IMAGE IS RUNNING**

**STAGE 3: TEST\_SOLUTION**

1. **CREATE A NEW FOLDER “TERRAFORM-OUTPUTS”**
2. **INSIDE TERRAFORM-OUTPUTS, CREATE MAIN.TF, VARIABLES.TF, OUTPUTS.TF MAIN.TF**

provider "aws" {

region = "ap-southeast-2" # Replace with your AWS region

}

# Data source for existing EC2 instance data "aws\_instance" "frontend" {

instance\_id = "i-0d5900b1d8a43f0b5" # Replace with your actual instance ID

}

**VARIABLES.TF**

# variables.tf

variable "frontend\_instance\_id" {

description = "instance id of frontend ec2 instance" type = string

default = "i-0d5900b1d8a43f0b5"

}

**OUTPUTS.TF**

# outputs.tf

output "frontend\_public\_ip" {

description = "The public IP address of the frontend instance" value = data.aws\_instance.frontend.public\_ip

}

output "frontend\_public\_dns" {

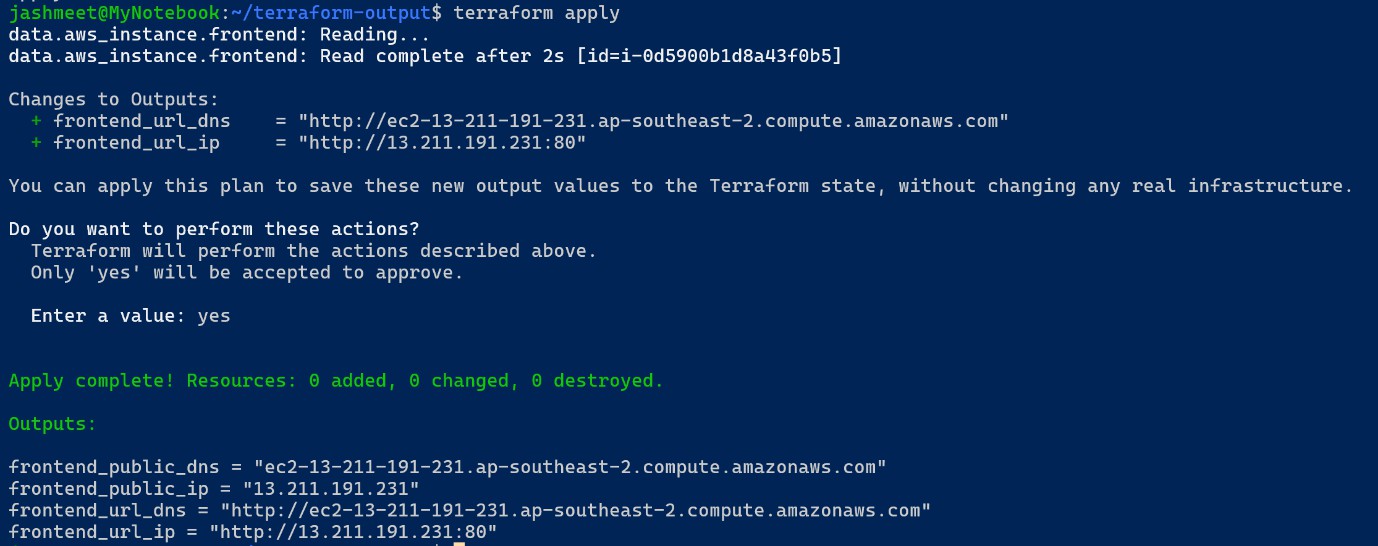
description = "The public DNS of the frontend instance" value = data.aws\_instance.frontend.public\_dns

}

output "frontend\_url\_ip" {

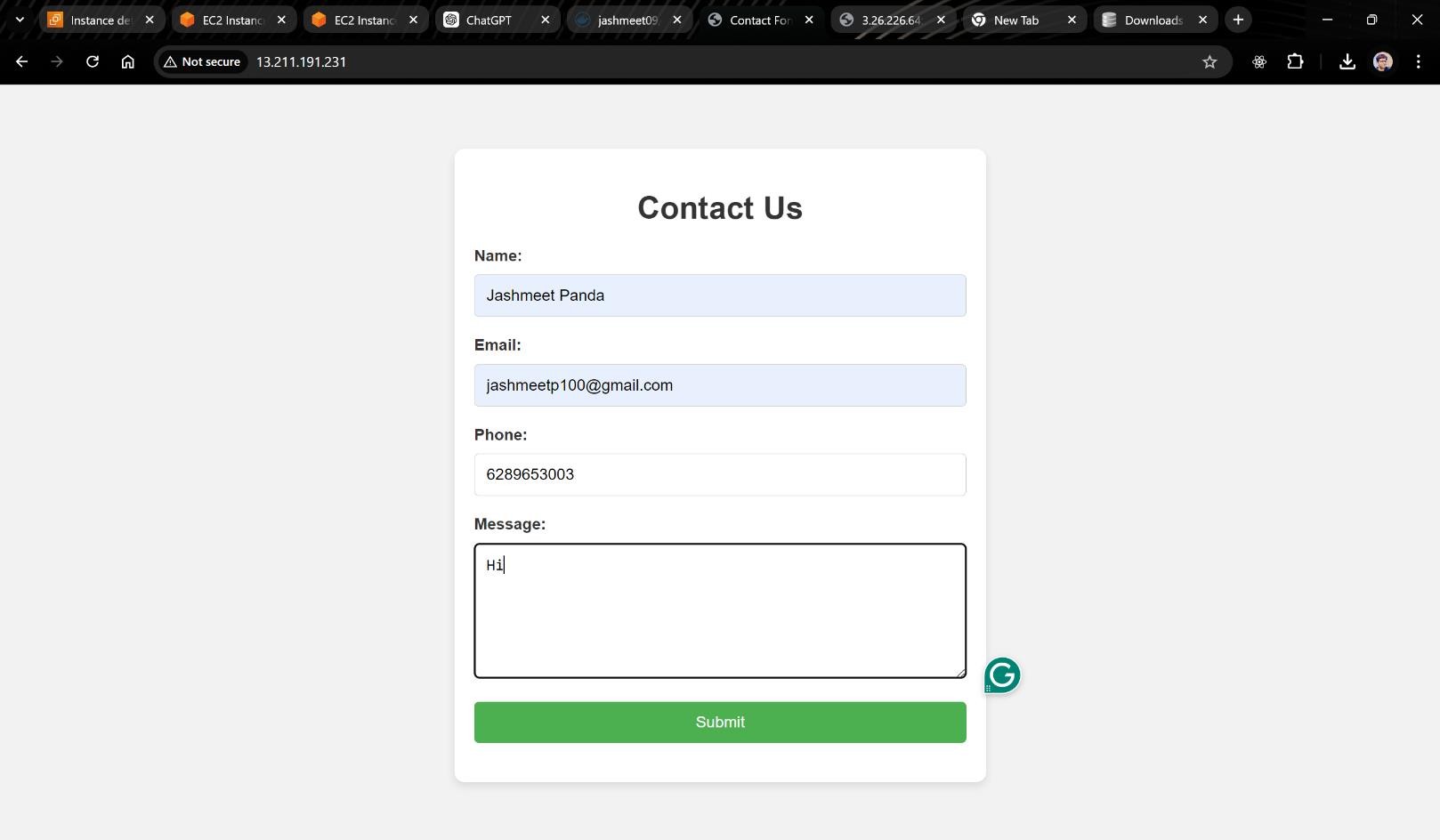
description = "The URL of the frontend instance using public IP" value = "http://${data.aws\_instance.frontend.public\_ip}:80"

}

**OUTPUT**

ON CLICKING ON frontend\_url\_ip = [**https://13.211.191.231:80**](https://13.211.191.231:80)

The following webpage will open in the browser.



**JENKINS FILE**

pipeline {

agent any

parameters {

password(name: 'AIAW3MEDG23TG4BTK6U')

password(name: '0XYjJOcXyyRsL4L6mxnpTH/GLo72rhNfDLpCIjk')

}

environment {

TF\_WORKSPACE = 'dev' // Sets the Terraform Workspace

TF\_IN\_AUTOMATION = 'true'

}

stages {

stage('Terraform init') {

steps {

script {

withCredentials([string(credentialsId: 'AKIAW3MEDG23TG4BTK6U', variable: 'AKIAW3MEDG23TG4BTK6'),

string(credentialsId: '03XYjJOcXyyRsL4L6mxnpTH/GLo72rhNfDLpCIjk', variable: '03XYjJOcXyyRsL4L6mxnpTH/GLo72rhfDLpCIjk')]) {

sh """

cd /home/jashmeet/terraform\_vpc/main.tf

export AWS\_ACCESS\_KEY\_ID=${AKAW3MEDG23TG4BTK6U}

export AWS\_SECRET\_ACCESS\_KEY=${3XYjJOcXyyRsL4L6mxnpTH/GLo72rhNfDLpCIjk}

terraform init -input=false

"""

}

}

}

}

stage('Terraform plan') {

steps {

script {

withCredentials([string(credentialsId: 'AKIW3MEDG23TG4BTK6U', variable: 'AKIAW3MEDG23TG4BTK6U'),

string(credentialsId: '03XYjJOcXyRsL4L6mxnpTH/GLo72rhNfDLpCIjk', variable: '03XYjJOcXyyRsL4L6mxnpTH/GLo72rhNfDLpCIjk')]) {

sh """

cd /home/jashmeet/terraform\_vpc/main.tf

export AWS\_ACCESS\_KEY\_ID=${AIAW3MEDG23TG4BTK6U}

export AWS\_SECRET\_ACCESS\_KEY=${03YjJOcXyyRsL4L6mxnpTH/GLo72rhNfDLpCIjk}

terraform plan -out=tfplan -input=false

"""

}

}

}

}

stage('Terraform apply') {

steps {

script {

withCredentials([string(credentialsId: 'AKIAWMEDG23TG4BTK6UD', variable: 'AKIAW3MEDG23TG4BTK6U'),

string(credentialsId: '03XYjJOcXyyRL4L6mxnpTH/GLo72rhNfDLpCIjk', variable: '03XYjJOcXyyRsL4L6mxnpTH/GLo72rhNfDLpCIjk')]) {

sh """

cd /home/jashmeet/terraform\_vpc/main.tf

export AWS\_ACCESS\_KEY\_ID=${AKIW3MEDG23TG4BTK6U}

export AWS\_SECRET\_ACCESS\_KEY=${3XYjJOcXyyRsL4L6mxnpTH/GLo72rhNfDLpCIjk}

terraform apply -input=false tfplan

}

}

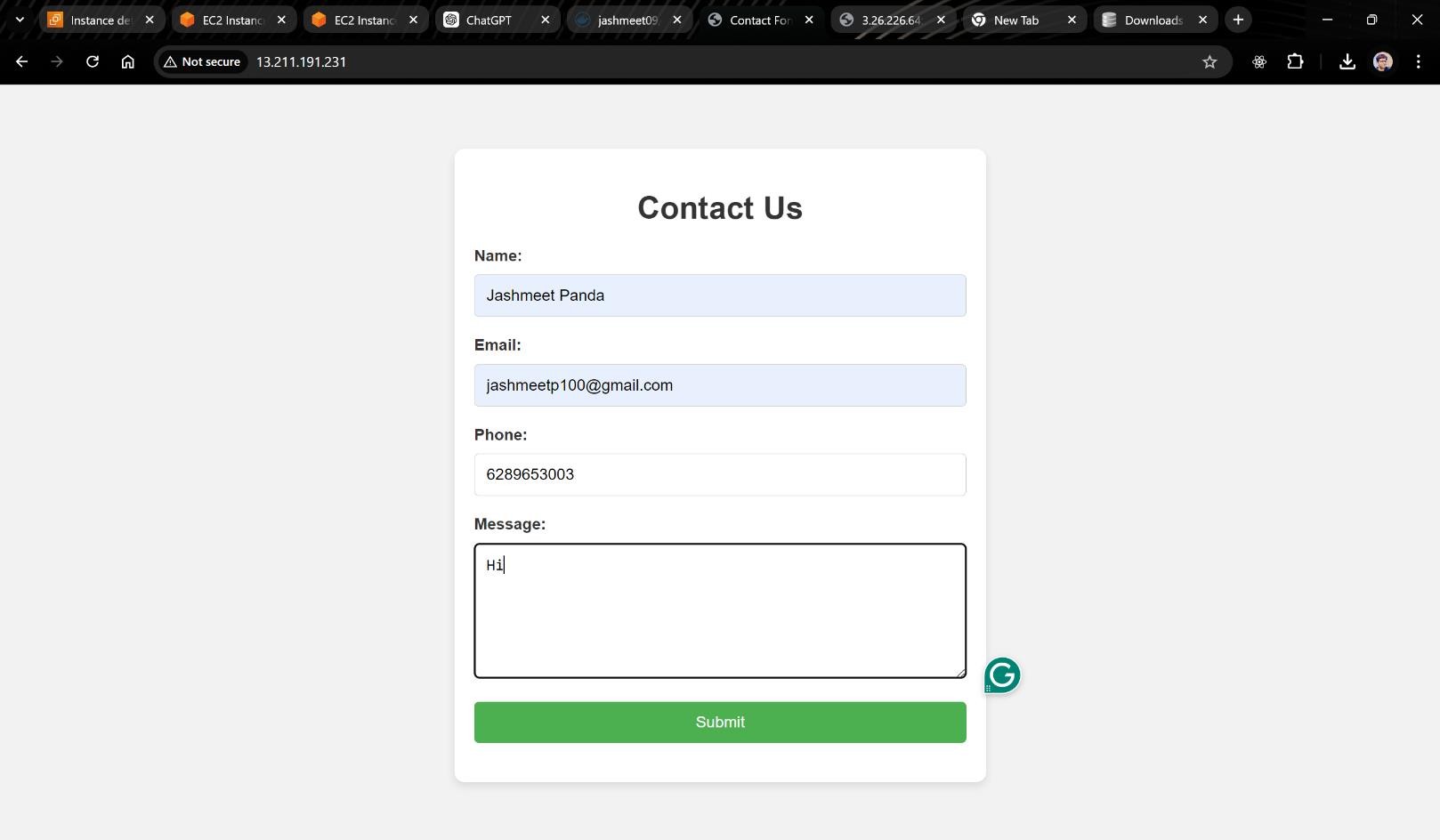
}

}

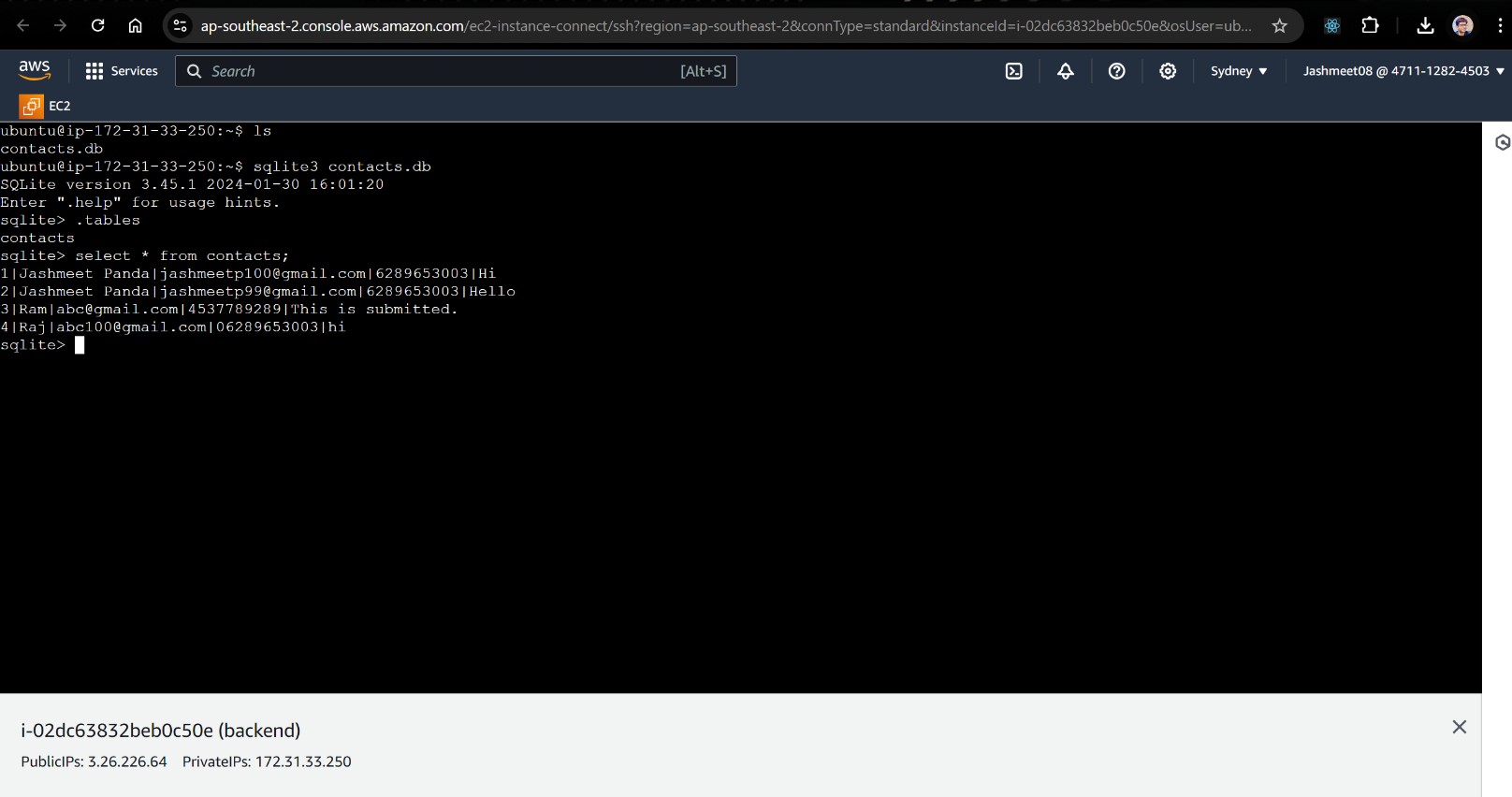
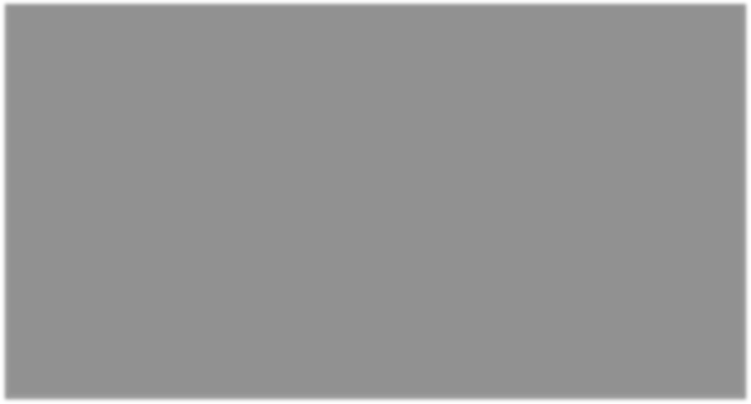
}

}

# OUTPUTS



FRONTEND INSTANCE WITH PUBLIC IP



BACKEND INSTANCE WITH PUBLIC IP



JENKINS PIPELINE SCRIPTS



