**Step 1: Set Up Your Environment**

1. **Update and install required packages**:

bash

Copy code

sudo apt-get update

sudo apt-get install -y python3-pip docker.io

sudo pip3 install boto3 psycopg2-binary

sudo systemctl start docker

sudo systemctl enable docker

1. **Install AWS CLI**:

bash

Copy code

curl "https://awscli.amazonaws.com/awscli-exe-linux-x86\_64.zip" -o "awscliv2.zip"

unzip awscliv2.zip

sudo ./aws/install

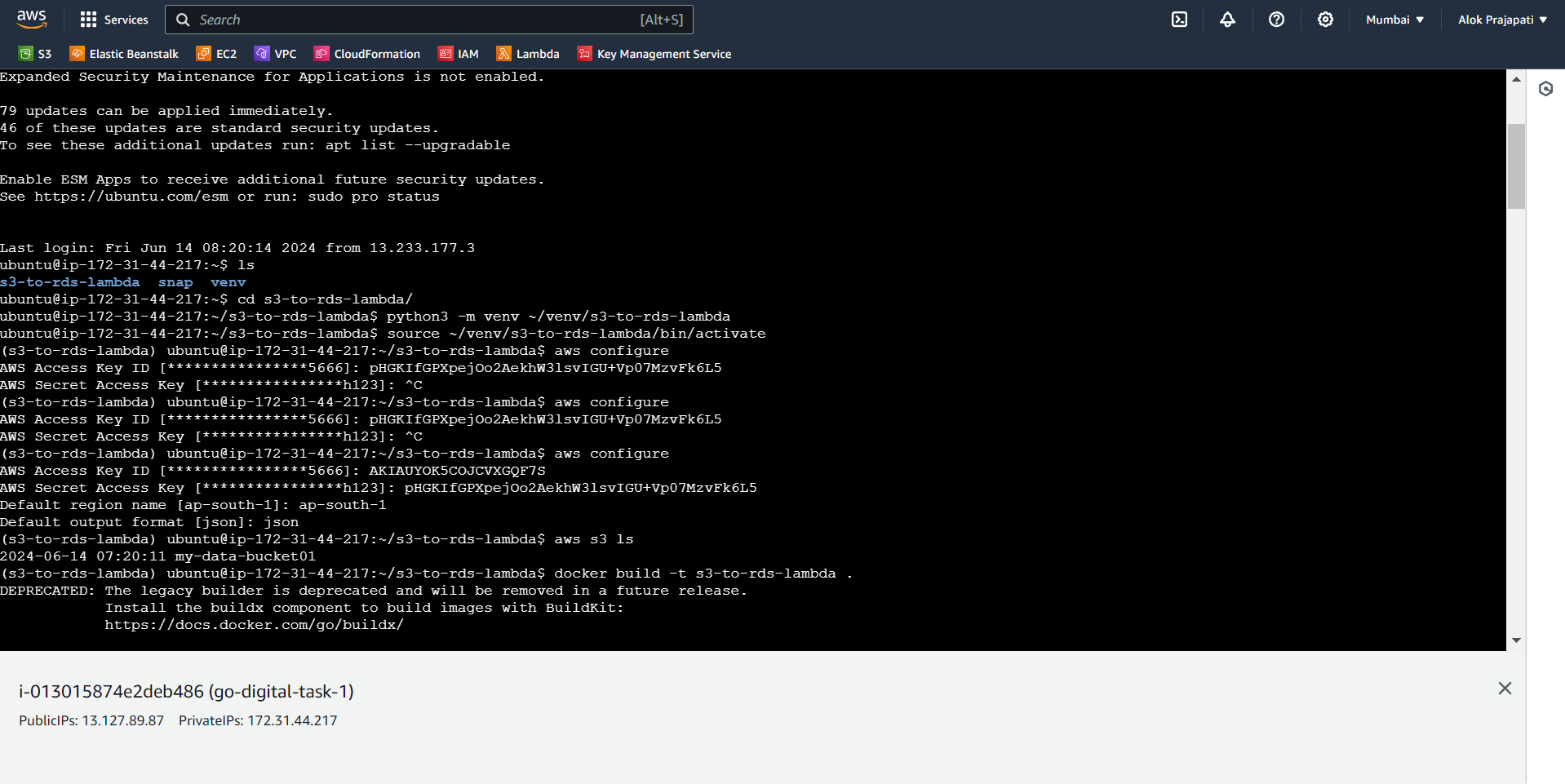
1. **Configure AWS CLI**:

bash

Copy code

aws configure

Enter your AWS Access Key ID, Secret Access Key, region (ap-south-1 for Mumbai), and output format (json).



### Step 2: Create the Python Application

1. **Create a project directory**:

bash

Copy code

mkdir ~/s3-to-rds-lambda

cd ~/s3-to-rds-lambda

1. **Create app.py**:

python

Copy code

import boto3

import psycopg2

import os

S3\_BUCKET = os.getenv('S3\_BUCKET')

S3\_KEY = os.getenv('S3\_KEY')

RDS\_HOST = os.getenv('RDS\_HOST')

RDS\_PORT = os.getenv('RDS\_PORT')

RDS\_DB = os.getenv('RDS\_DB')

RDS\_USER = os.getenv('RDS\_USER')

RDS\_PASSWORD = os.getenv('RDS\_PASSWORD')

def read\_from\_s3(bucket, key):

s3 = boto3.client('s3')

obj = s3.get\_object(Bucket=bucket, Key=key)

data = obj['Body'].read().decode('utf-8')

return data

def write\_to\_rds(data, conn):

with conn.cursor() as cur:

cur.execute("INSERT INTO your\_table (data\_column) VALUES (%s)", (data,))

conn.commit()

def main():

data = read\_from\_s3(S3\_BUCKET, S3\_KEY)

conn = psycopg2.connect(

host=RDS\_HOST,

port=RDS\_PORT,

dbname=RDS\_DB,

user=RDS\_USER,

password=RDS\_PASSWORD

)

write\_to\_rds(data, conn)

conn.close()

if \_\_name\_\_ == '\_\_main\_\_':

main()

1. **Create requirements.txt**:

text

Copy code

boto3

psycopg2-binary

1. **Create Dockerfile**:

dockerfile

Copy code

FROM python:3.9-slim

WORKDIR /app

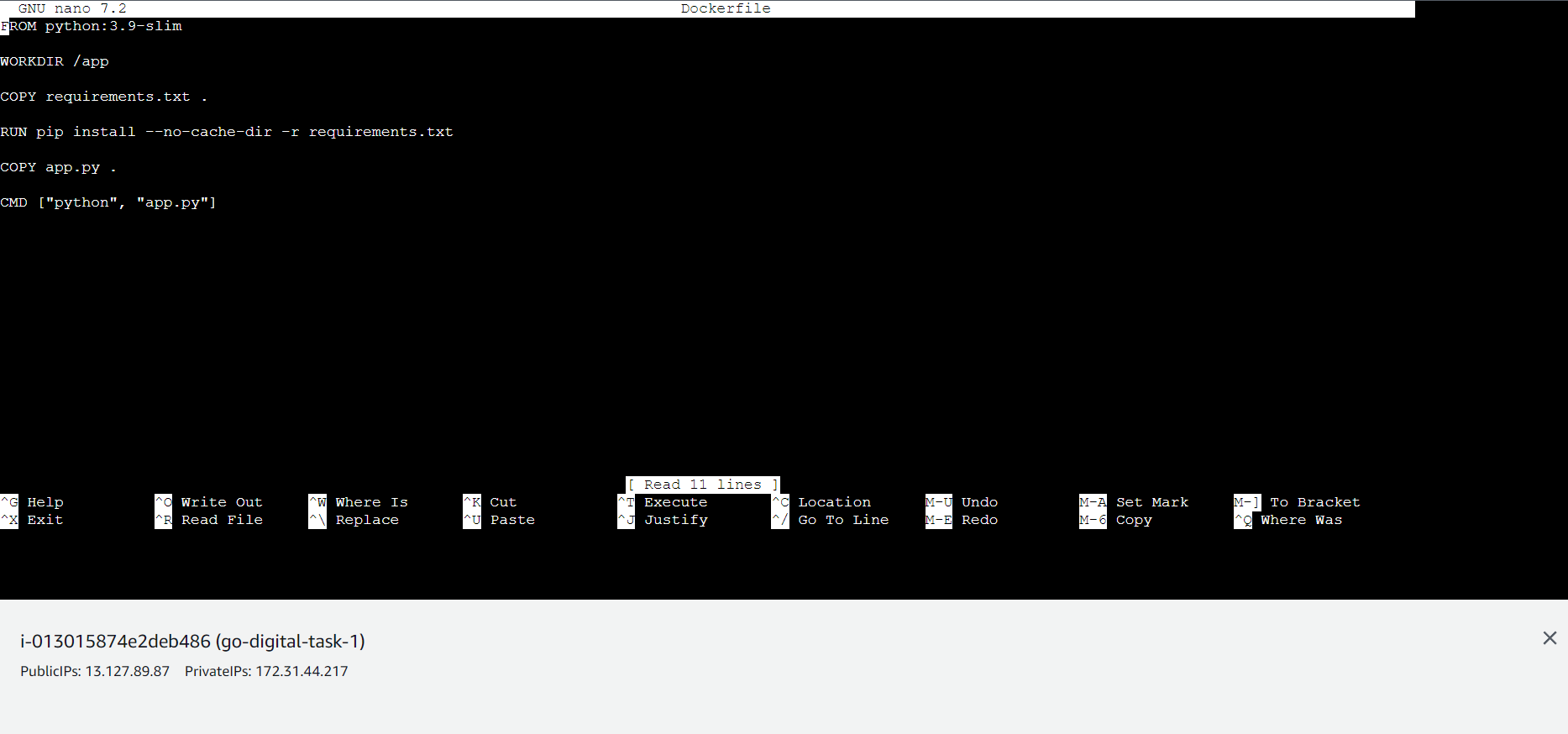
COPY requirements.txt .

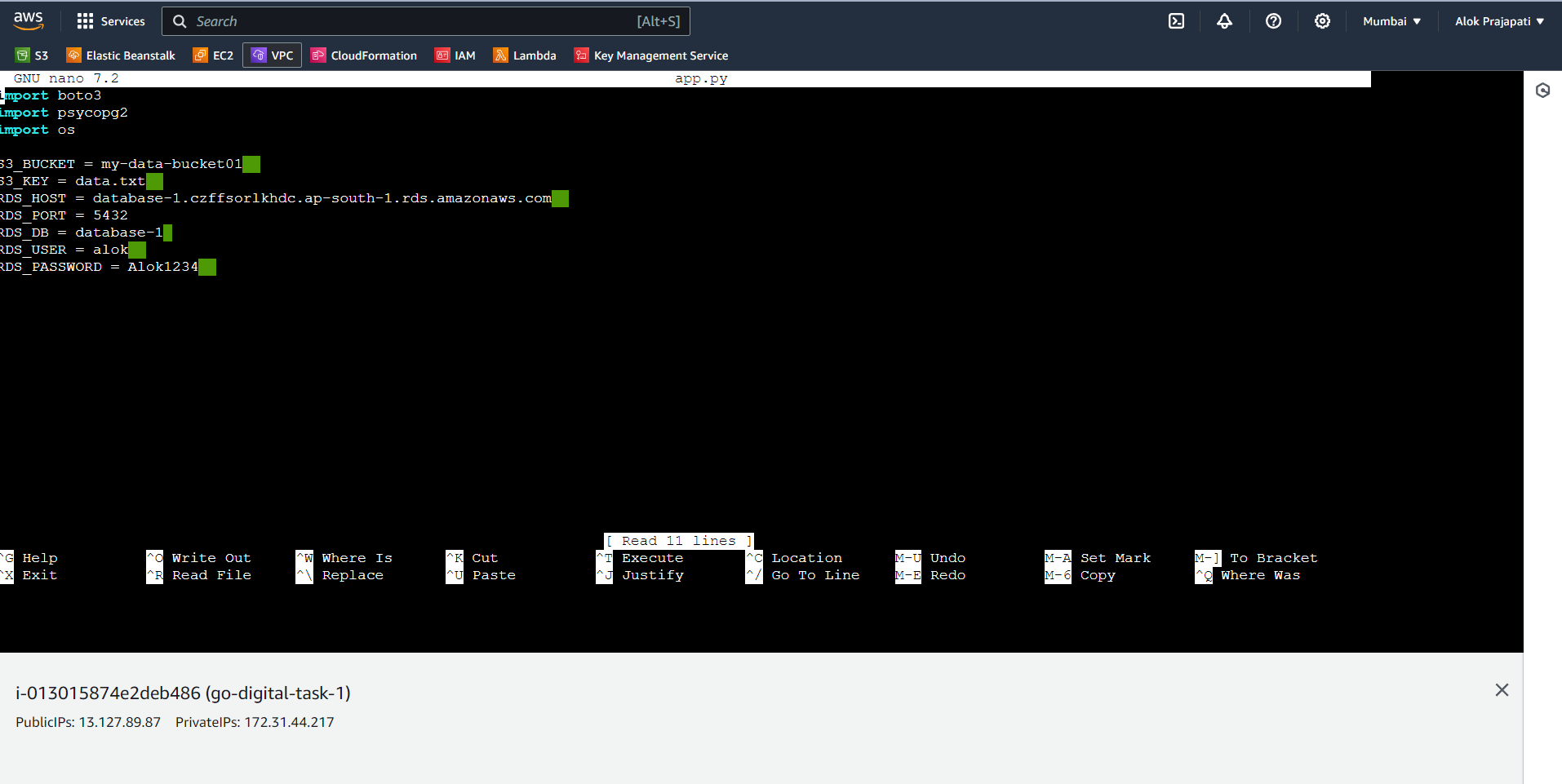
RUN pip install --no-cache-dir -r requirements.txt

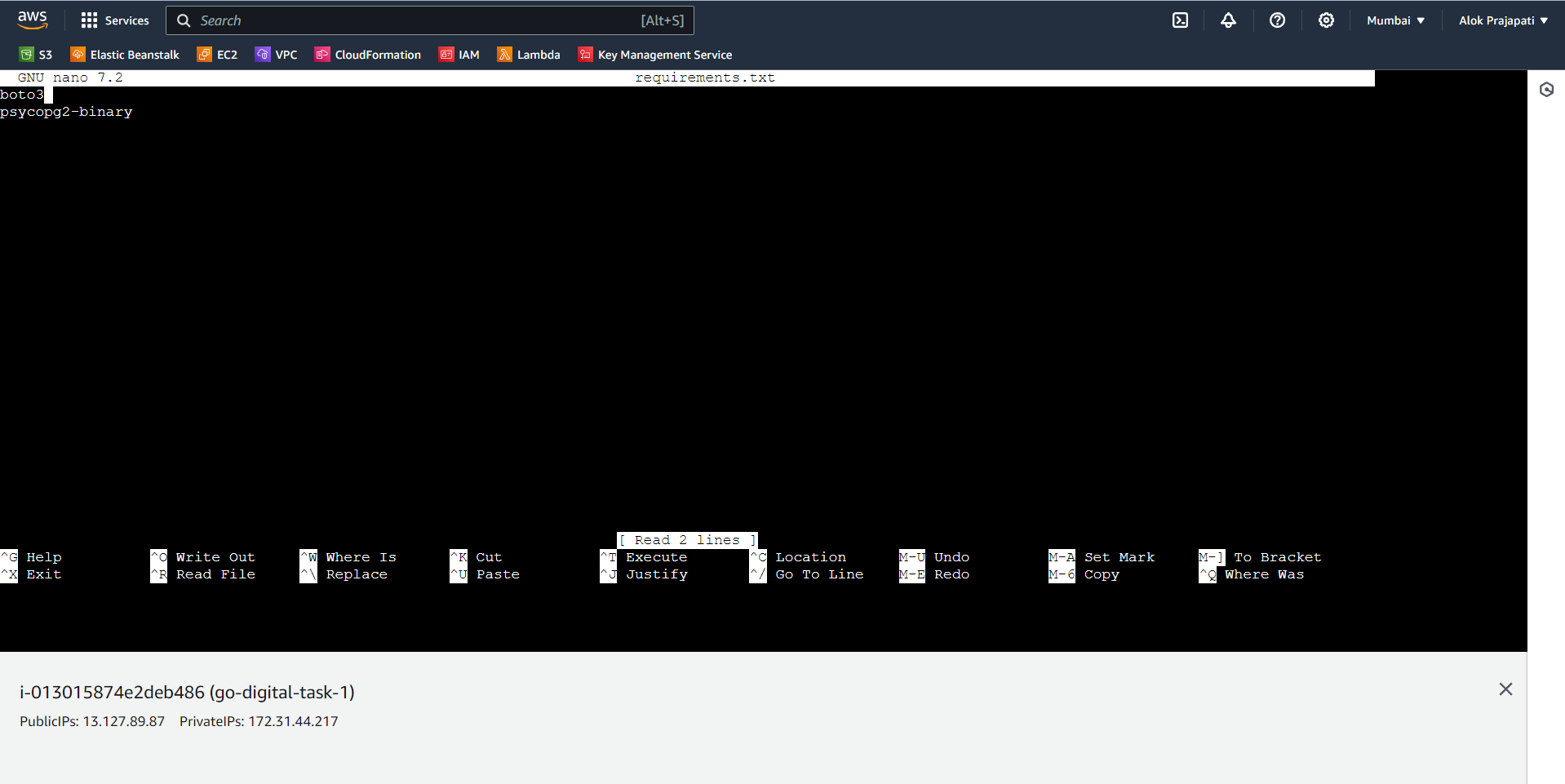
COPY app.py .

CMD ["python", "app.py"]









### Step 3: Build and Push Docker Image to ECR

1. **Create ECR Repository**:

bash

Copy code

aws ecr create-repository --repository-name s3-to-rds-lambda --region ap-south-1

1. **Authenticate Docker to ECR**:

bash

Copy code

aws ecr get-login-password --region ap-south-1 | docker login --username AWS --password-stdin <account-id>.dkr.ecr.ap-south-1.amazonaws.com

1. **Build Docker Image**:

bash

Copy code

docker build -t s3-to-rds-lambda .

1. **Tag Docker Image**:

bash

Copy code

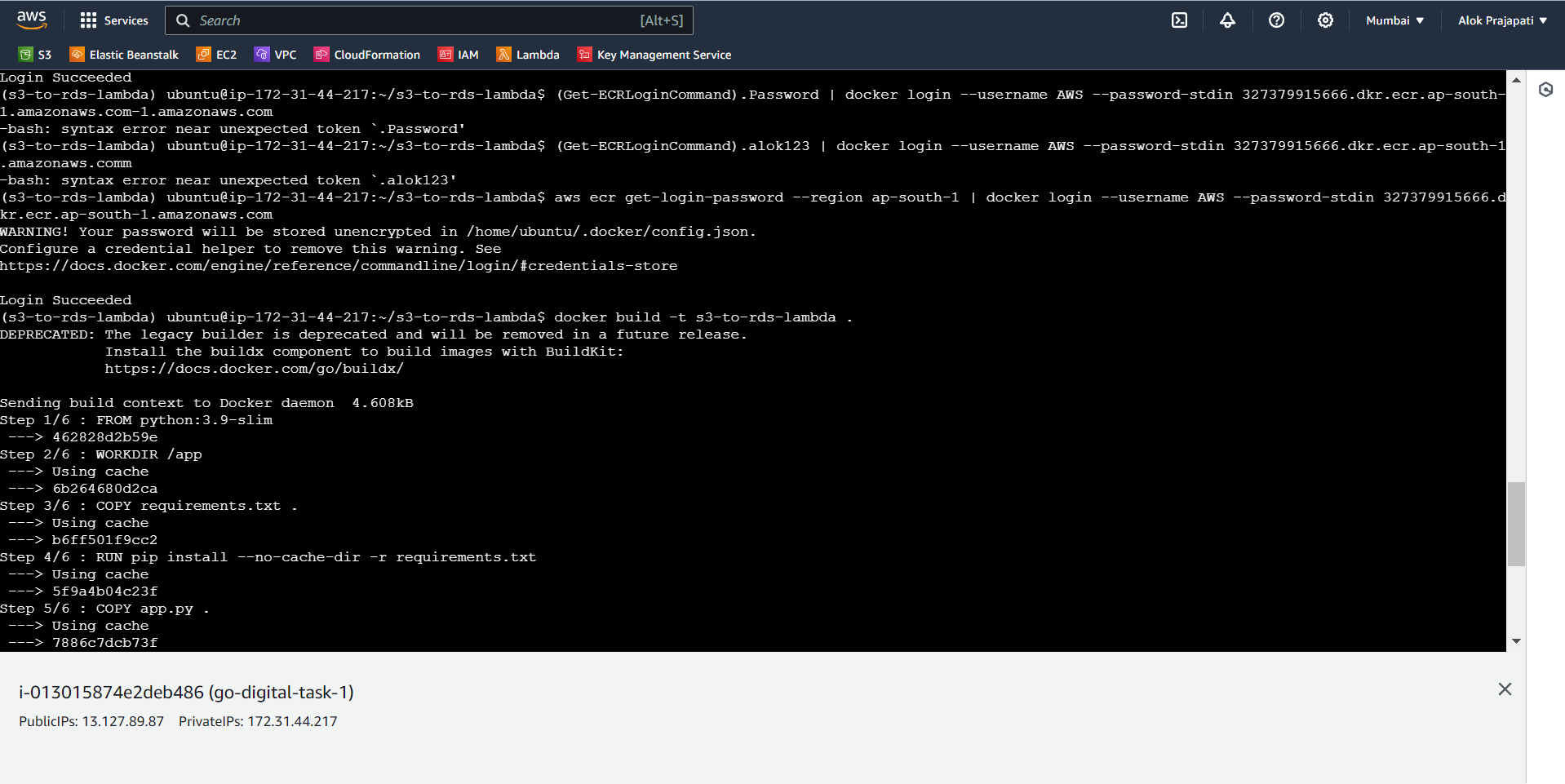
docker tag s3-to-rds-lambda:latest <account-id>.dkr.ecr.ap-south-1.amazonaws.com/s3-to-rds-lambda:latest

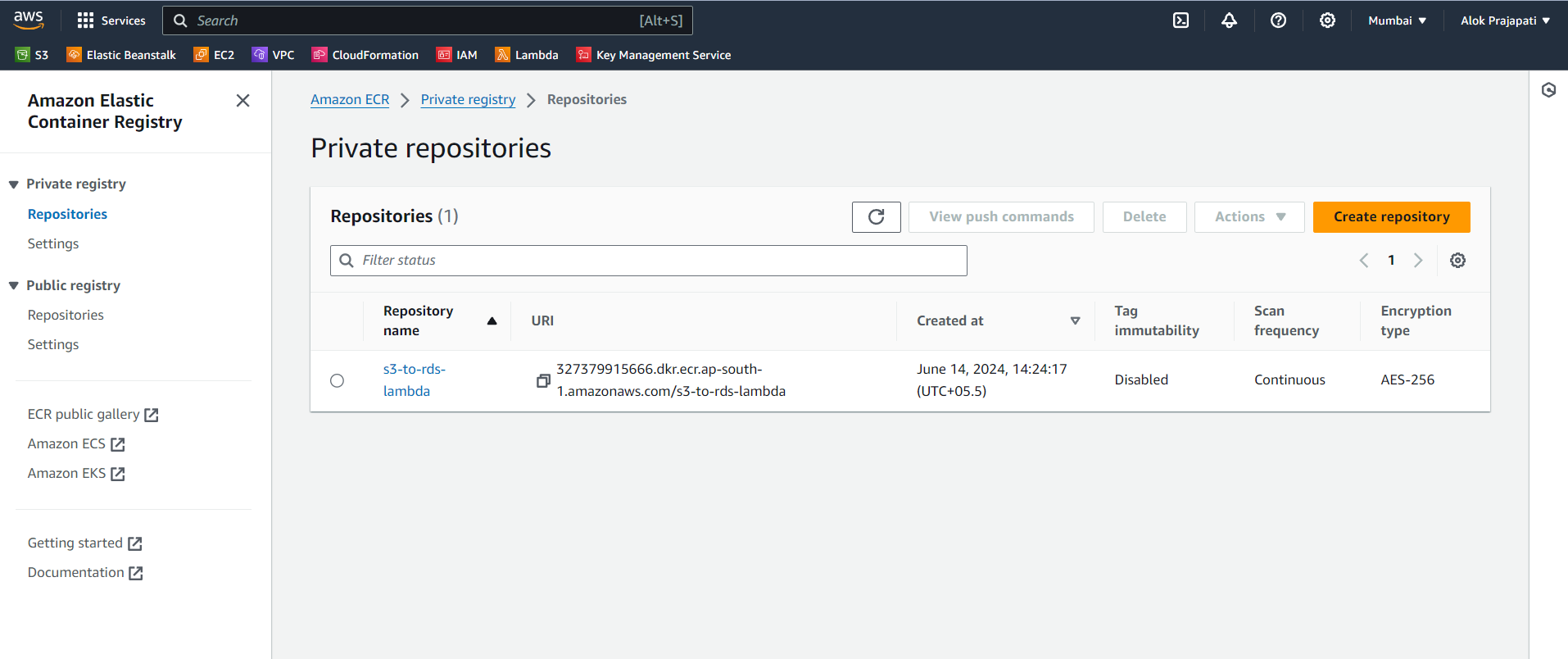
1. **Push Docker Image**:

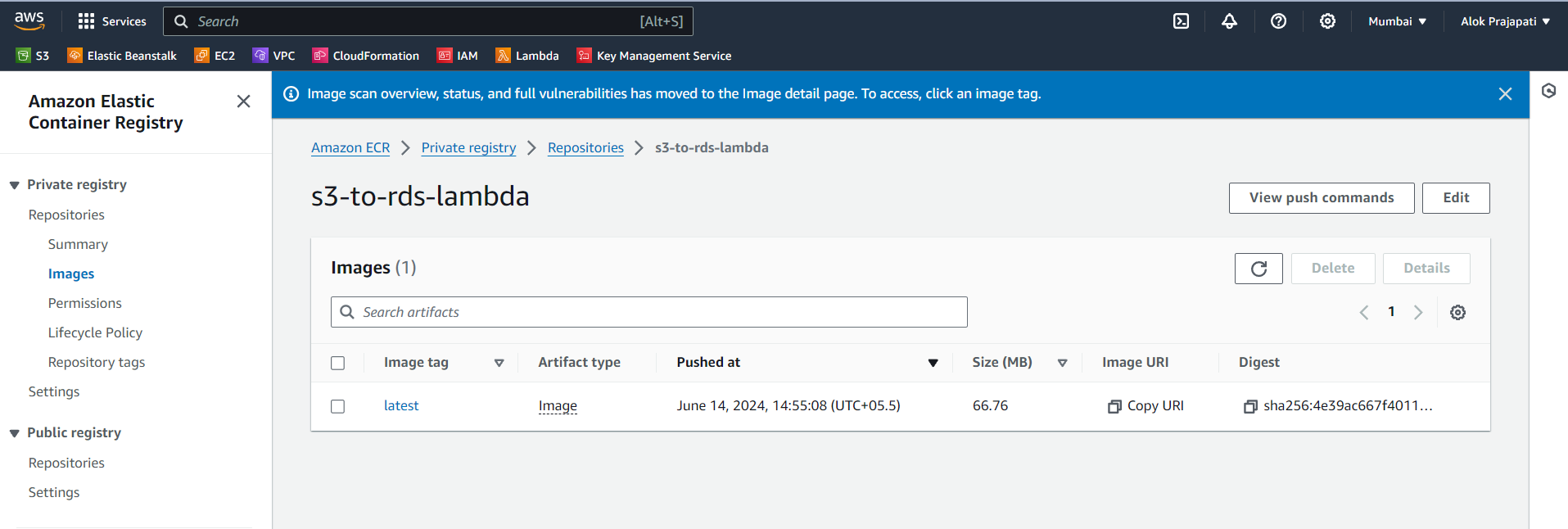
bash

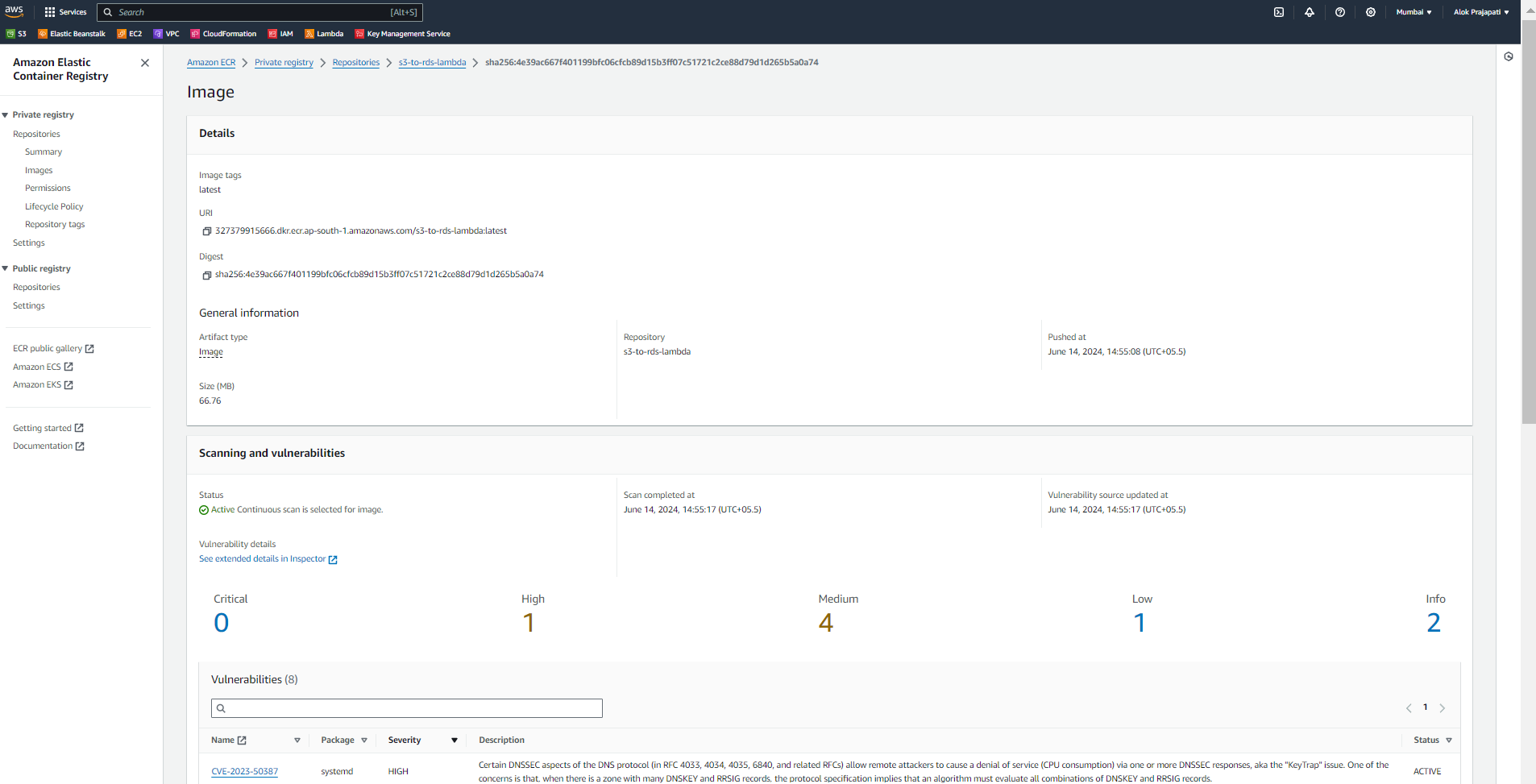
Copy code

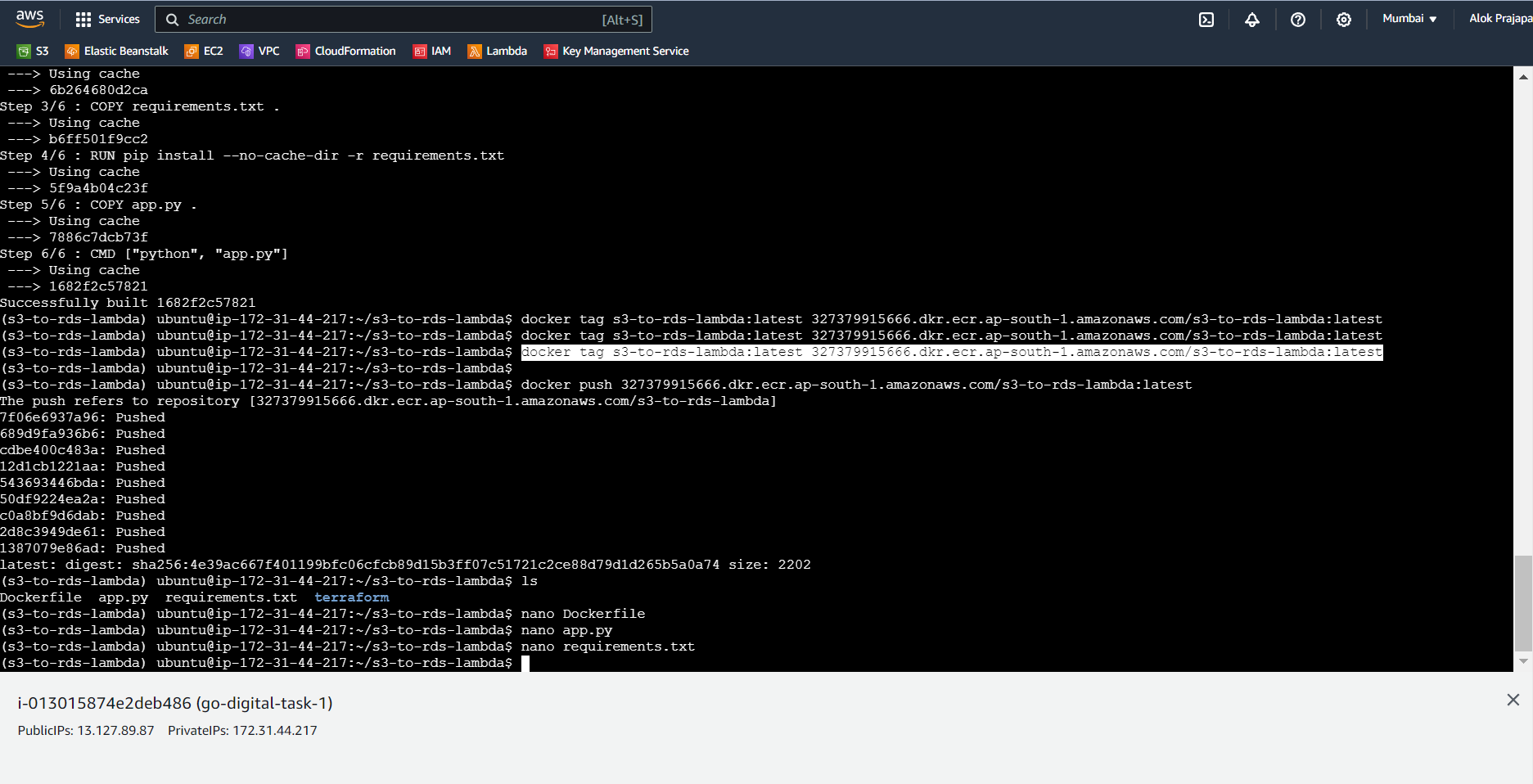
docker push <account-id>.dkr.ecr.ap-south-1.amazonaws.com/s3-to-rds-lambda:latest











### Step 4: Create Terraform Configuration for Lambda and Other Resources

1. **Navigate to the Terraform directory**:

bash

Copy code

mkdir ~/s3-to-rds-lambda/terraform

cd ~/s3-to-rds-lambda/terraform

1. **Create main.tf**:

hcl

Copy code

provider "aws" {

region = var.aws\_region

}

resource "aws\_ecr\_repository" "repository" {

name = var.ecr\_repository\_name

}

resource "aws\_iam\_role" "lambda\_role" {

name = "lambda\_role"

assume\_role\_policy = jsonencode({

Version = "2012-10-17"

Statement = [

{

Action = "sts:AssumeRole"

Effect = "Allow"

Principal = {

Service = "lambda.amazonaws.com"

}

}

]

})

}

resource "aws\_iam\_role\_policy" "lambda\_policy" {

name = "lambda\_policy"

role = aws\_iam\_role.lambda\_role.id

policy = jsonencode({

Version = "2012-10-17"

Statement = [

{

Action = [

"logs:CreateLogGroup",

"logs:CreateLogStream",

"logs:PutLogEvents",

"s3:GetObject",

"rds-db:connect"

]

Effect = "Allow"

Resource = "\*"

}

]

})

}

resource "aws\_lambda\_function" "lambda\_function" {

function\_name = var.lambda\_function\_name

role = aws\_iam\_role.lambda\_role.arn

image\_uri = var.lambda\_image\_uri

package\_type = "Image"

environment {

variables = {

S3\_BUCKET = var.s3\_bucket

S3\_KEY = var.s3\_key

RDS\_HOST = var.rds\_host

RDS\_PORT = var.rds\_port

RDS\_DB = var.rds\_db

RDS\_USER = var.rds\_user

RDS\_PASSWORD = var.rds\_password

}

}

}

1. **Create variables.tf**:

hcl

Copy code

variable "aws\_region" {

description = "The AWS region to deploy to"

type = string

default = "ap-south-1"

}

variable "ecr\_repository\_name" {

description = "The name of the ECR repository"

type = string

default = "s3-to-rds-lambda"

}

variable "lambda\_function\_name" {

description = "The name of the Lambda function"

type = string

default = "s3-to-rds-lambda"

}

variable "lambda\_image\_uri" {

description = "The URI of the Docker image for the Lambda function"

type = string

default = "<account-id>.dkr.ecr.ap-south-1.amazonaws.com/s3-to-rds-lambda:latest"

}

variable "s3\_bucket" {

description = "The S3 bucket name"

type = string

default = "my-data-bucket01"

}

variable "s3\_key" {

description = "The S3 key"

type = string

default = "data.txt"

}

variable "rds\_host" {

description = "The RDS host"

type = string

default = "database-1.czffsorlkhdc.ap-south-1.rds.amazonaws.com"

}

variable "rds\_port" {

description = "The RDS port"

type = string

default = "5432"

}

variable "rds\_db" {

description = "The RDS database name"

type = string

default = "database-1"

}

variable "rds\_user" {

description = "The RDS user"

type = string

default = "alok"

}

variable "rds\_password" {

description = "The RDS password"

type = string

sensitive = true

default = "Alok1234"

}

1. **Create outputs.tf**:

hcl

Copy code

output "lambda\_function\_name" {

value = aws\_lambda\_function.lambda\_function.function\_name

}

1. **Initialize and apply Terraform configuration**:

bash

Copy code

terraform init

terraform apply -auto-approve

