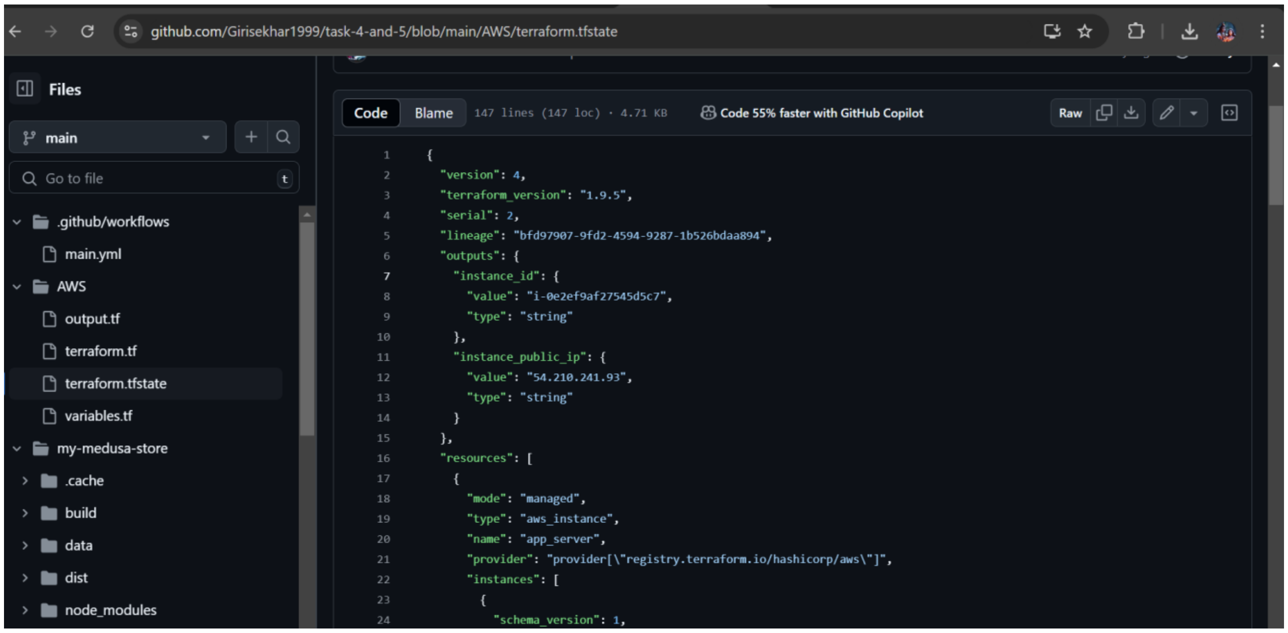
Medusa Setup on EC2 Instance through Terraform and Github actions :)

Step 1: Connect to the EC2 Instance and write code to create ec2 through Terraform.



github.com/Girisekhar1999/task-4-and-5/blob/main/AWS/terraform.tfstate

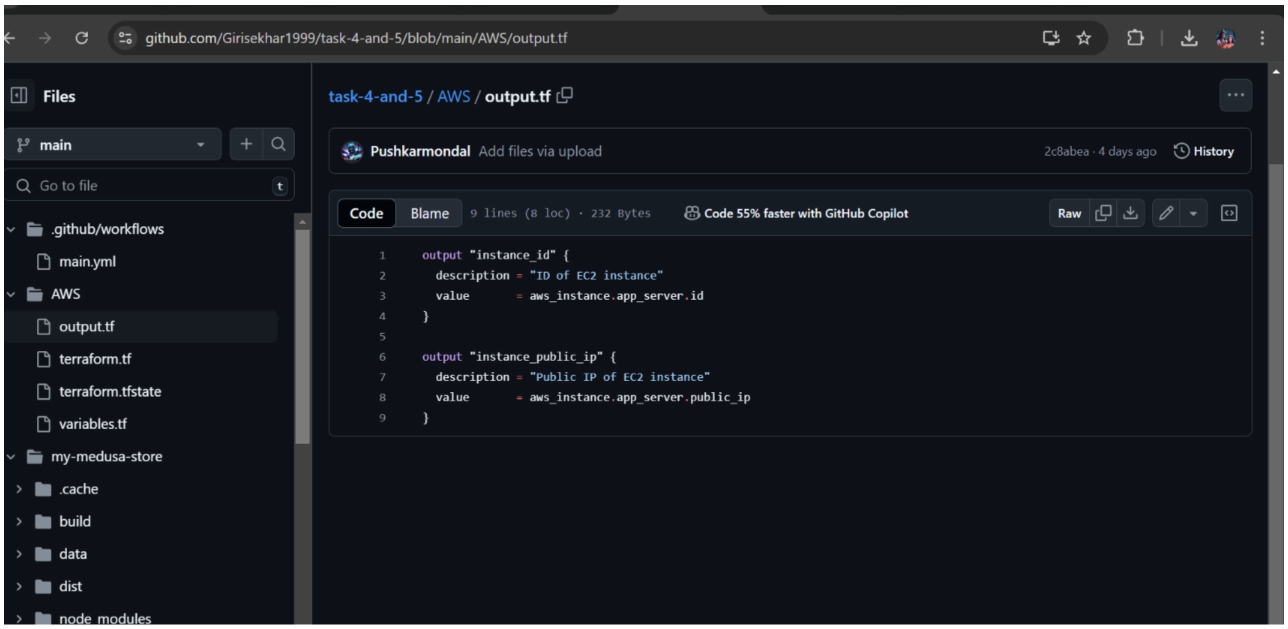
Files

- main
- .github/workflows
  - main.yml
- AWS
  - output.tf
  - terraform.tf
  - terraform.tfstate
  - variables.tf
- my-medusa-store
  - .cache
  - build
  - data
  - dist
  - node_modules

Code | Blame | 147 lines (147 loc) · 4.71 KB | Code 55% faster with GitHub Copilot | Raw

```
{
  "version": 4,
  "terraform_version": "1.9.5",
  "serial": 2,
  "lineage": "bfd97907-9fd2-4594-9287-1b526bdaa894",
  "outputs": {
    "instance_id": {
      "value": "i-0e2ef9af27545d5c7",
      "type": "string"
    },
    "instance_public_ip": {
      "value": "54.210.241.93",
      "type": "string"
    }
  },
  "resources": [
    {
      "mode": "managed",
      "type": "aws_instance",
      "name": "app_server",
      "provider": "provider[\"registry.terraform.io/hashicorp/aws\"]",
      "instances": [
        {
          "schema_version": 1,
```

Outputs and logs.


github.com/Girisekhar1999/task-4-and-5/blob/main/AWS/output.tf
Files
main
Go to file
.github/workflows
main.yml
AWS
output.tf
terraform.tf
terraform.tfstate
variables.tf
my-medusa-store
.cache
build
data
dist
node_modules
task-4-and-5 / AWS / output.tf
Pushkarmondal Add files via upload
2c8abea · 4 days ago
History
Code
Blame
9 lines (8 loc) · 232 Bytes
Code 55% faster with GitHub Copilot
Raw

```
output "instance_id" {
  description = "ID of EC2 instance"
  value       = aws_instance.app_server.id
}

output "instance_public_ip" {
  description = "Public IP of EC2 instance"
  value       = aws_instance.app_server.public_ip
}
```

Step 2: Update the System & Install Dependencies

Once connected to the EC2 instance, run the following commands to update the package manager and install Node.js, npm, and PostgreSQL:

sudo apt update -y
sudo apt install nodejs npm -y
sudo apt install postgresql postgresql-contrib -y

Step 3: Install Medusa CLI

Install Medusa CLI globally using npm:
sudo npm install -g @medusajs/medusa-cli

Step 4: Start PostgreSQL Service

Start the PostgreSQL service:
sudo service postgresql start

Step 5: Set Up PostgreSQL Database

Log in to PostgreSQL as the postgres user and create a new user and database for Medusa:
sudo -u postgres psql

Within the PostgreSQL prompt, run the following SQL commands to create the user and database:

```
CREATE USER medusa_user WITH PASSWORD 'password';
CREATE DATABASE medusa_db OWNER medusa_user;
\q
```

Step 6: Create a New Medusa Project

Clone the already deployed Github repo

Step 7: Configure Environment Variables

Open the .env file for editing:

```
Vim .env
```

Add the following line to configure the database URL:

```
DATABASE_URL=postgres://medusa_user:password@localhost:5432/medusa_db
```

Step 8: Install Node Dependencies

Run the following command to install all necessary dependencies for the Medusa project:

```
npm install
```

Step 9: Seed the Database

Seed the database with initial data:

```
npm run seed
```

Step 10: Create an Admin User

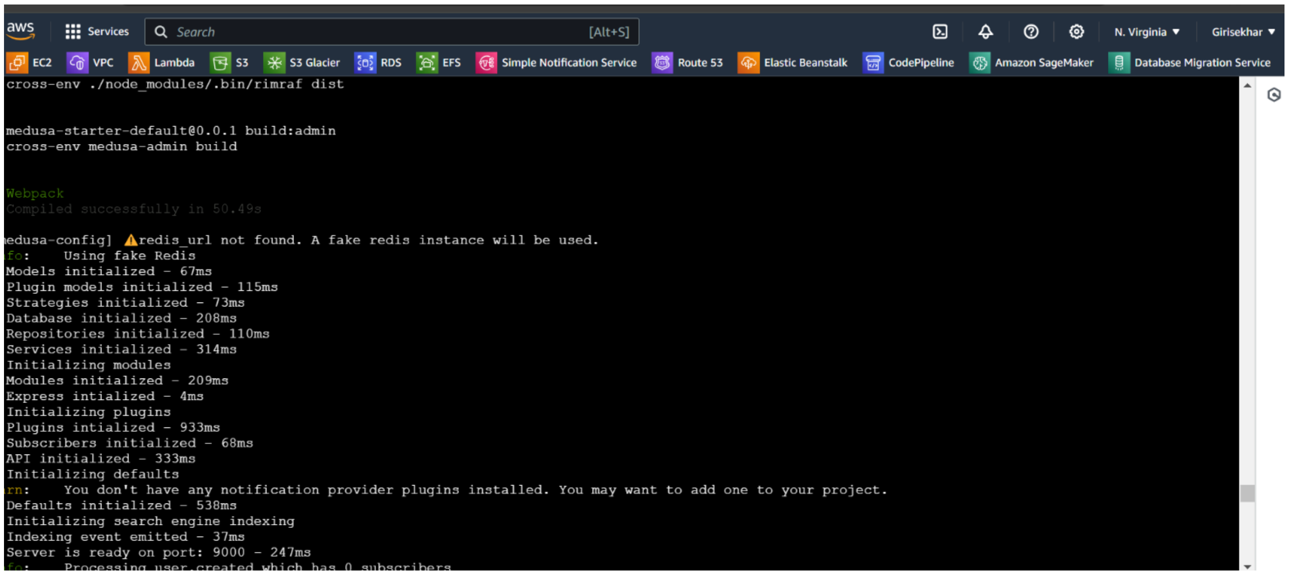
Create an admin user with the following command:

```
npx medusa user -e “your email” -p supersect
```

Step 11: Start the Medusa Server to test

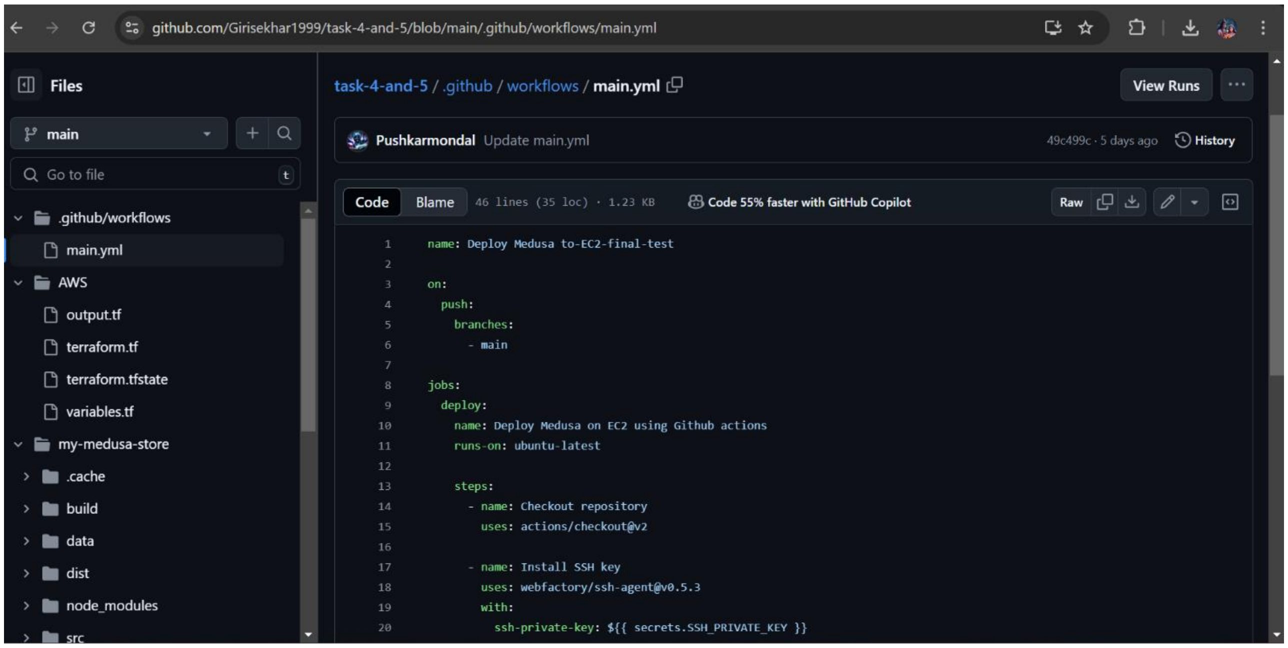
Start the Medusa server:

```
npm run start
```



```
cross-env ./node_modules/.bin/rimraf dist


medusa-starter-default@0.0.1 build:admin
cross-env medusa-admin build


Webpack
Compiled successfully in 50.49s

edusa-config] ⚠redis_url not found. A fake redis instance will be used.
fo:    Using fake Redis
Models initialized – 67ms
Plugin models initialized – 115ms
Strategies initialized – 73ms
Database initialized – 208ms
Repositories initialized – 110ms
Services initialized – 314ms
Initializing modules
Modules initialized – 209ms
Express intialized – 4ms
Initializing plugins
Plugins intialized – 933ms
Subscribers initialized – 68ms
API initialized – 333ms
Initializing defaults
rn:    You don't have any notification provider plugins installed. You may want to add one to your project.
Defaults initialized – 538ms
Initializing search engine indexing
Indexing event emitted – 37ms
Server is ready on port: 9000 – 247ms
fo:    Processing user.created which has 0 subscribers
```

Step 12: Setup the github actions yaml file


github.com/Girisekhar1999/task-4-and-5/blob/main/.github/workflows/main.yml

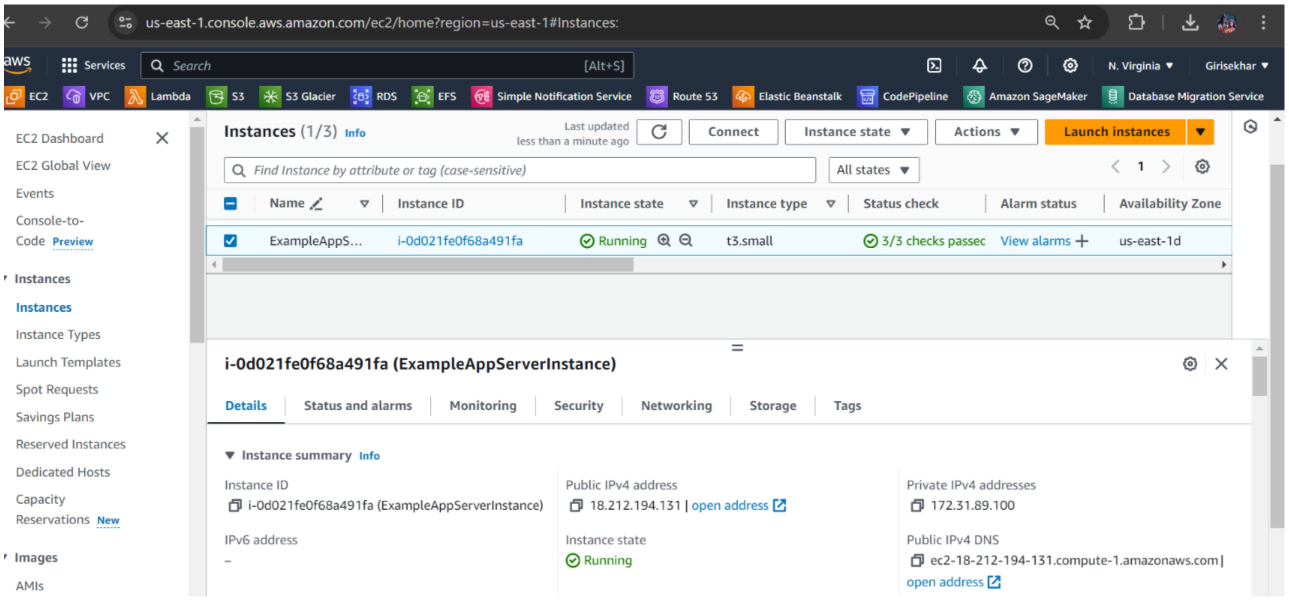
task-4-and-5 / .github / workflows / main.yml

Pushkarmondal Update main.yml 49c499c · 5 days ago History

46 lines (35 loc) · 1.23 KB

```yaml
name: Deploy Medusa to-EC2-final-test

on:
  push:
    branches:
      - main

jobs:
  deploy:
    name: Deploy Medusa on EC2 using Github actions
    runs-on: ubuntu-latest

    steps:
      - name: Checkout repository
        uses: actions/checkout@v2

      - name: Install SSH key
        uses: webfactory/ssh-agent@v0.5.3
        with:
          ssh-private-key: ${{ secrets.SSH_PRIVATE_KEY }}
```

Step 13: check using some small tweak into the ec2.


us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#Instances:

Instances (1/3)

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
ExampleAppS...	i-0d021fe0f68a491fa	Running	t3.small	3/3 checks passec	View alarms +	us-east-1d

i-0d021fe0f68a491fa (ExampleAppServerInstance)

Instance summary

Instance ID: i-0d021fe0f68a491fa (ExampleAppServerInstance)

Public IPv4 address: 18.212.194.131 | open address

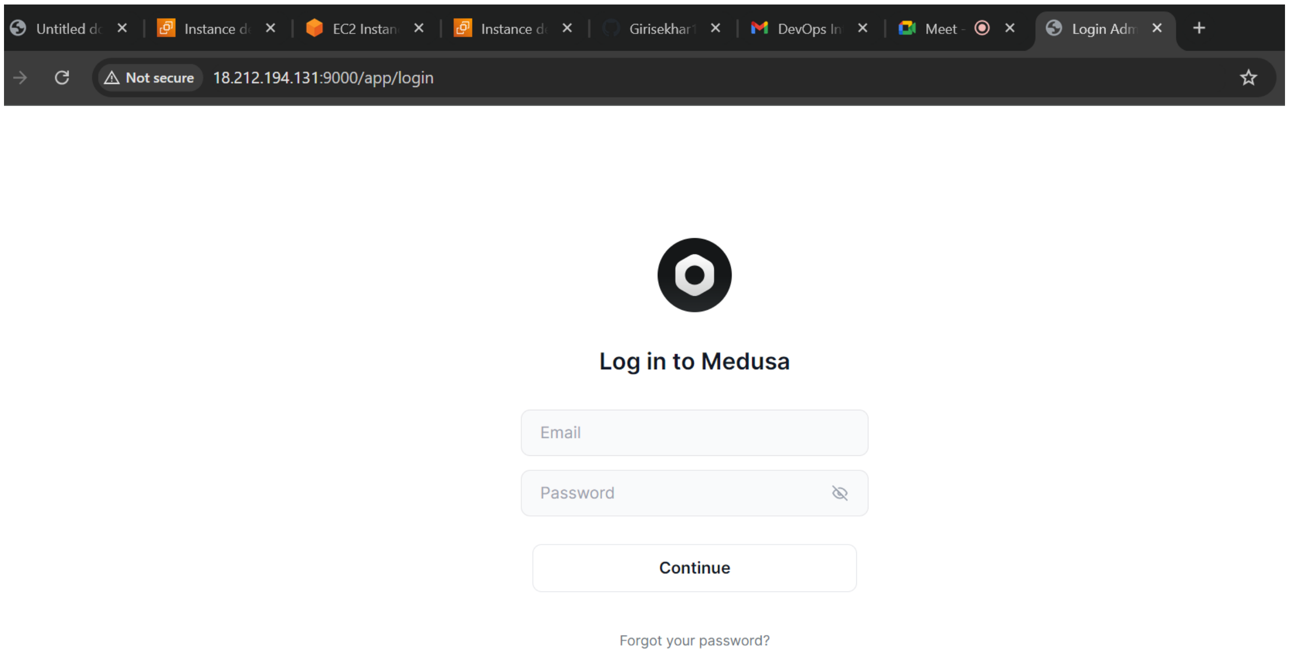
Private IPv4 addresses: 172.31.89.100

IPv6 address: –

Instance state: Running

Public IPv4 DNS: ec2-18-212-194-131.compute-1.amazonaws.com | open address

Finally login medusa :



Loom Video : https://www.loom.com/share/862a16bce07740789af5aa3d19dfe463?sid=a5b47dc5-a061-41ec-a704-4288ca89322a