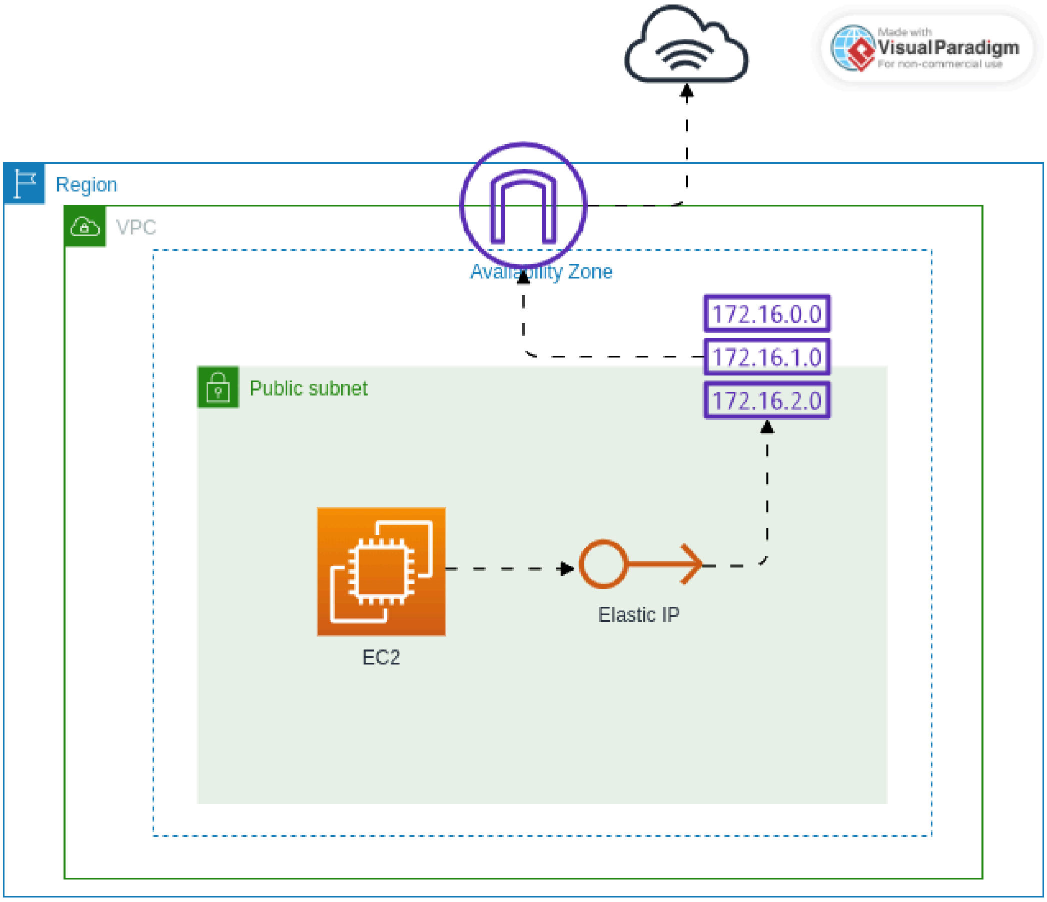
Interview Tasks

Task1 Cloud Infrastructure & Deployment on AWS

Architecture Diagram



Steps to Deploy the Application

1. Create an AWS Account

2. Create an Cloudformation template

```
# structure from here

#
```

```
https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/template-formats.html#template-comments



AWSTemplateFormatVersion: 2010-09-09

Description: Interview Test file



Parameters:

EC2InstanceSizeInput:

Description: The supoorted instances sizes for EC2

Type: String

Default: t2.micro

AllowedValues:

- t3.micro

- t2.micro

Resources:

# VPC , subnet , igw , route table , router table to igw rule, security group , security group rules , eip, nic , ec2 , user data , ssm role , ssm policy, role assumption



#VPC

TestVPC:

Type: AWS::EC2::VPC
```

```yaml
Properties:
CidrBlock: 10.0.0.0/16
Tags:
- Key: ProjectNumber
Value: 4
- Key: ProjectName
Value: interviewData

TestIGW:
Type: AWS::EC2::InternetGateway
Properties:
Tags:
- Key: ProjectNumber
Value: 4
- Key: ProjectName
Value: interviewData

TestAttachGateway:
Type: AWS::EC2::VPCGatewayAttachment
Properties:
```

```yaml
VpcId: !Ref TestVPC
InternetGatewayId: !Ref TestIGW


#Subnet
TestPublicSubnet:
Type: AWS::EC2::Subnet
Properties:
VpcId: !Ref TestVPC
AvailabilityZone: "ap-south-1a"
CidrBlock: 10.0.0.1/24
Tags:
- Key: ProjectNumber
Value: 4
- Key: ProjectName
Value: interviewData


TestRouteTable:
Type: AWS::EC2::RouteTable
Properties:
VpcId: !Ref TestVPC
```

```yaml
Tags:
- Key: ProjectNumber
Value: 4
- Key: ProjectName
Value: interviewData
TestInternetPublicRoute:
Type: AWS::EC2::Route
DependsOn: TestIGW
Properties:
RouteTableId: !Ref TestRouteTable
DestinationCidrBlock: 0.0.0.0/0
GatewayId: !Ref TestIGW
TestRouteTableToTestSubnetAssociation:
Type: AWS::EC2::SubnetRouteTableAssociation
Properties:
RouteTableId: !Ref TestRouteTable
SubnetId: !Ref TestPublicSubnet

TestInstanceSecurityGroup:
Type: AWS::EC2::SecurityGroup
```

```yaml
Properties:

GroupDescription: Allow EC2 traffic

VpcId: !Ref TestVPC

SecurityGroupIngress:

- Description: Allow ssh

IpProtocol: tcp

FromPort: 22

ToPort: 22

CidrIp: 0.0.0.0/0

- Description: Allow http

IpProtocol: tcp

FromPort: 80

ToPort: 80

CidrIp: 0.0.0.0/0

- Description: Allow https

IpProtocol: tcp

FromPort: 443

ToPort: 443

CidrIp: 0.0.0.0/0

- Description: Allow all
```

```
IpProtocol: -1

CidrIp: 0.0.0.0/0

SecurityGroupEgress:

- Description: Allow ssh

IpProtocol: tcp

FromPort: 22

ToPort: 22

CidrIp: 0.0.0.0/0

- Description: Allow http

IpProtocol: tcp

FromPort: 80

ToPort: 80

CidrIp: 0.0.0.0/0

- Description: Allow https

IpProtocol: tcp

FromPort: 443

ToPort: 443

CidrIp: 0.0.0.0/0

- Description: Allow all

IpProtocol: -1
```

```yaml
CidrIp: 0.0.0.0/0
Tags:
- Key: ProjectNumber
Value: 4
- Key: ProjectName
Value: interviewData
TestEIP:
Type: AWS::EC2::EIP
Properties:
Tags:
- Key: ProjectNumber
Value: 4
- Key: ProjectName
Value: interviewData

TestNetworkInterface:
Type: AWS::EC2::NetworkInterface
Properties:
Description: A External Network Interface for the EC2
SubnetId: !Ref TestPublicSubnet
```

```
GroupSet:
- !Ref TestInstanceSecurityGroup
Tags:
- Key: ProjectNumber
Value: 4
- Key: ProjectName
Value: interviewData


TestEIPAssociation:
Type: AWS::EC2::EIPAssociation
Properties:
AllocationId: !GetAtt TestEIP.AllocationId
NetworkInterfaceId: !Ref TestNetworkInterface
DependsOn: TestNetworkInterface
DependsOn: TestEIP

TestEC2Instance:
Type: 'AWS::EC2::Instance'
Properties:
```

```
ImageId: ami-002f6e91abff6eb96 # ami-053b12d3152c0cc71 for ubuntu

InstanceType: !Ref EC2InstanceSizeInput

IamInstanceProfile : !Ref InstanceProfileOfRoleToEC2

# SubnetId : !Ref TestPublicSubnet

# SecurityGroupIds:

# - !Ref TestInstanceSecurityGroup

NetworkInterfaces:

- Description: A Network interface made externally with AWS EIP attached at start up as primary

DeviceIndex: 0

NetworkInterfaceId: !Ref TestNetworkInterface

KeyName : myEC2KeyForInterview

Tags:

- Key: ProjectNumber

Value: 4

- Key: ProjectName

Value: interviewData

UserData:

Fn::Base64: !Sub |

#!/bin/bash
```

```
dnf update -y

dnf install httpd git python pip -y

yum install docker -y

systemctl start docker

systemctl enable docker

usermod -aG docker $USER

mkdir -p /usr/local/lib/docker/cli-plugins

curl -SL https://github.com/docker/compose/releases/latest/download/docker-compose-linux-x86_64 -o /usr/local/lib/docker/cli-plugins/docker-compose

chmod +x /usr/local/lib/docker/cli-plugins/docker-compose

cd /home/ec2-user

git clone https://github.com/AryanGitHub/a-very-simple-webapp-for-assignment.git 2> error.log

cd a-very-simple-webapp-for-assignment

bash bash.sh 2> error_bash.log

DependsOn: TestEIPAssociation

SSMEC2ControlRole:

Type: AWS::IAM::Role

Properties:

Description: SSM Role for Test EC2
```

```yaml
AssumeRolePolicyDocument:
Version: "2012-10-17"
Statement:
- Effect: Allow
Principal:
Service:
- ec2.amazonaws.com
Action:
- 'sts:AssumeRole'
ManagedPolicyArns:
- arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore
MaxSessionDuration: 3600
RoleName: Test_EC2_Role
Policies: # Adding inline policy for CloudWatch Logs
- PolicyName: CloudWatchLogsPolicy
PolicyDocument:
Version: "2012-10-17"
Statement:
- Effect: Allow
Action:
```

```
- logs:CreateLogGroup

- logs:CreateLogStream

- logs:PutLogEvents

- logs:DescribeLogStreams

Resource: "*"

Tags:

- Key: ProjectNumber

Value: 4

- Key: ProjectName

Value: interviewData



InstanceProfileOfRoleToEC2:

Type: AWS::IAM::InstanceProfile

Properties:

InstanceProfileName: SSMEC2Role

Roles:

- !Ref SSMEC2ControlRole
```

3. Build the app and push it on github

main.py

```
from fastapi import FastAPI, Form, Request
from fastapi.responses import HTMLResponse, RedirectResponse
from fastapi.templating import Jinja2Templates
from prometheus_fastapi_instrumentator import Instrumentator
app = FastAPI()
templates = Jinja2Templates(directory="templates")

todos = []
Instrumentator().instrument(app).expose(app)
@app.get("/", response_class=HTMLResponse)
async def read_root(request: Request):
    return templates.TemplateResponse("index.html",
{"request": request, "todos": todos})

@app.post("/add", response_class=HTMLResponse)
async def add_todo(request: Request, task: str = Form(...)):
    todos.append(task)
    return RedirectResponse(url="/", status_code=303)
```

and templates/index.html

```
<!DOCTYPE html>
<html>
<head>
    <title>FastAPI ToDo App</title>
</head>
<body>
    <h1>📝 ToDo List</h1>

    <form action="/add" method="post">
        <input type="text" name="task" placeholder="Enter a
task" required>
        <button type="submit">Add.</button>
    </form>

    <ul>
        {% for todo in todos %}
            <li>{{ todo }}</li>
        {% else %}
            <li>No tasks yet!</li>
        {% endfor %}
    </ul>
</body>
</html>
```

add a bash file to host it

```
#!/bin/bash
python -m venv .venv
source .venv/bin/activate
pip install -r ./requirements.txt
pip install prometheus-fastapi-instrumentator
uvicorn main:app --host 0.0.0.0 --port 80 --reload
```

3. Deploy Cloudformation template using aws cli command

and This themplate contains USERDATA, so it will automatically pull app from github repo

```
#!/bin/bash

aws cloudformation deploy --region ap-south-1 \

--template-file ./main.yaml \

--stack-name ec2forinterview \

--tags madeFromCLI=yeah anotherTagForAllStackResources=okay \

--capabilities CAPABILITY_NAMED_IAM

#--no-execute-changeset
```

4.AWS configurations used (Resource Groups, Networking, etc

CloudFormation Resources Summary

VPC and Networking Resources

1. **VPC**
   - **Logical ID:** `TestVPC`
   - **Type:** `AWS::EC2::VPC`
   - **Properties:**
     - CidrBlock: `10.0.0.0/16`
2. **Internet Gateway**
   - **Logical ID:** `TestIGW`
   - **Type:** `AWS::EC2::InternetGateway`
3. **VPC Gateway Attachment**
   - **Logical ID:** `TestAttachGateway`
   - **Type:** `AWS::EC2::VPCGatewayAttachment`
   - **Properties:**
     - VpcId: `!Ref TestVPC`
     - InternetGatewayId: `!Ref TestIGW`
4. **Subnet**
   - **Logical ID:** `TestPublicSubnet`
   - **Type:** `AWS::EC2::Subnet`
   - **Properties:**
     - VpcId: `!Ref TestVPC`
     - AvailabilityZone: `ap-south-1a`
     - CidrBlock: `10.0.0.1/24`
5. **Route Table**
   - **Logical ID:** `TestRouteTable`
   - **Type:** `AWS::EC2::RouteTable`
   - **Properties:**
     - VpcId: `!Ref TestVPC`

6. **Route**
   - **Logical ID:** `TestInternetPublicRoute`
   - **Type:** `AWS::EC2::Route`
   - **Properties:**
     - RouteTableId: `!Ref TestRouteTable`
     - DestinationCidrBlock: `0.0.0.0/0`
     - GatewayId: `!Ref TestIGW`
7. **Subnet Route Table Association**
   - **Logical ID:** `TestRouteTableToTestSubnetAssociation`
   - **Type:** `AWS::EC2::SubnetRouteTableAssociation`
   - **Properties:**
     - RouteTableId: `!Ref TestRouteTable`
     - SubnetId: `!Ref TestPublicSubnet`

Security Resources

8. **Security Group**
   - **Logical ID:** `TestInstanceSecurityGroup`
   - **Type:** `AWS::EC2::SecurityGroup`
   - **Properties:**
     - VpcId: `!Ref TestVPC`
     - Ingress and Egress rules defined for SSH, HTTP, HTTPS, and all traffic.

EC2 Resources

9. **Elastic IP**
   - **Logical ID:** `TestEIP`
   - **Type:** `AWS::EC2::EIP`
10. **Network Interface**
    - **Logical ID:** `TestNetworkInterface`
    - **Type:** `AWS::EC2::NetworkInterface`
    - **Properties:**
      - SubnetId: `!Ref TestPublicSubnet`

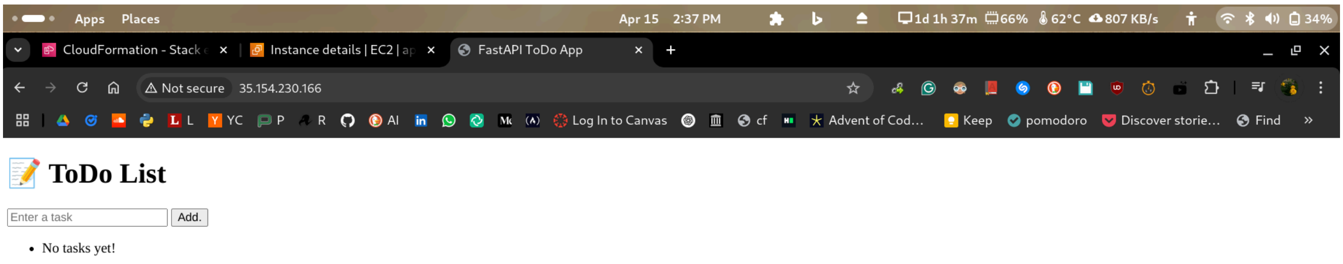
    - GroupSet: `!Ref TestInstanceSecurityGroup`

11. **EIP Association**
    - **Logical ID:** `TestEIPAssociation`
    - **Type:** `AWS::EC2::EIPAssociation`
    - **Properties:**
        - AllocationId: `!GetAtt TestEIP.AllocationId`
        - NetworkInterfaceId: `!Ref TestNetworkInterface`
12. **EC2 Instance**
    - **Logical ID:** `TestEC2Instance`
    - **Type:** `AWS::EC2::Instance`
    - **Properties:**
        - ImageId: `ami-002f6e91abff6eb96`
        - InstanceType: `!Ref EC2InstanceSizeInput`
        - NetworkInterfaces: `!Ref TestNetworkInterface`
        - UserData: Script for instance initialization.

IAM Resources

13. **IAM Role**
    - **Logical ID:** `SSMEC2ControlRole`
    - **Type:** `AWS::IAM::Role`
    - **Properties:**
        - AssumeRolePolicyDocument for EC2
        - ManagedPolicyArns: `arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore`
        - Inline policy for CloudWatch Logs.
14. **IAM Instance Profile**
    - **Logical ID:** `InstanceProfileOfRoleToEC2`
    - **Type:** `AWS::IAM::InstanceProfile`
    - **Properties:**
        - Roles: `!Ref SSMEC2ControlRole`

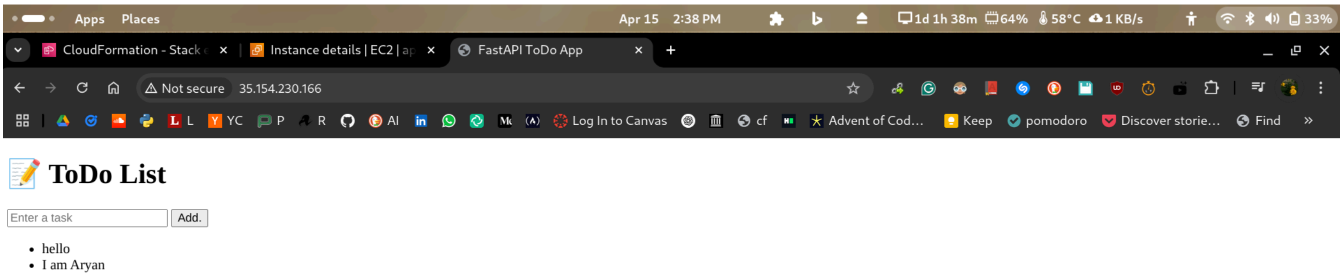
Screenshots



📝 ToDo List

Enter a task [Add.]

- No tasks yet!



📝 ToDo List

Enter a task [Add.]

- hello
- I am Aryan

Task2 CI/CD Pipeline Implementation

CI/CD pipeline YAML using Github actions

```yaml
name: remote ssh command

on: [push]

jobs:
  build:
    name: Build
    runs-on: ubuntu-latest
    steps:
      - name: Executing remote SSH commands using pem file
        uses: appleboy/ssh-action@v1
        with:
          host: ${{ secrets.HOST }}
          username: ec2-user
          key : ${{secrets.EC2_SSH_KEY}}
          port: 22
          script: |
            whoami
            echo "Deploying on EC2, logged IN"
            sudo chown -R ec2-user:ec2-user /home/ec2-user/a-very-simple-webapp-for-assignment
            git config --global --add safe.directory /home/ec2-user/a-very-simple-webapp-for-assignment
            cd /home/ec2-user/a-very-simple-webapp-for-assignment
            git pull
            echo "Deployment script ran successfully!"
```

Explanation of different pipeline stages

I have used single stage, to deploy the changes into the EC2 which is hosting the webapp
I used github action `appleboy/ssh-action@v1` its used to ssh into the ec2. it uses pem file contents into github repo secretes to deploy
I just run git pull from origin cus, uvicorn --reload is used to automatically reload the contents as the chags in the repo are detected.

How environment variables/secrets are managed

Env Vars ares managed using the github secrets for complete repo. there i have saved

the HOST public IP address for the ec2
the contents of pem file used to login into ec2 machine.

Task3 Security & Compliance (ISO, GDPR, SOC 2)

Security Risks Identification and Mitigation Strategies

1. **Insecure CI/CD Pipeline & Secrets Management**
   Risk: Exposure of sensitive information (API keys, tokens, database credentials, etc.) due to insecure storage of environment variables or incorrect configurations of the CI/CD pipeline.
   Mitigation Strategies:
   - **Secret Management:** Manage secrets securely using AWS Secrets Manager or AWS Systems Manager Parameter Store.
   - **Encryption:** Ensure secrets stored in AWS services are encrypted in transit and at rest using AWS Key Management Service (KMS).
   - **Isolation of Environments:** Limit exposure by using separate AWS accounts or Virtual Private Clouds (VPCs) for production, staging, and development environments.
   - **Regular Auditing:** Use AWS CloudTrail and AWS Config to continuously monitor and audit CI/CD configurations.
     Compliance Alignment:
   - **ISO 27001:** Aligns with requirements for secure access controls and encryption policies.
   - **SOC 2:** Addresses the security and confidentiality of information.
   - **GDPR:** Mandates a high level of protection for personal data.
2. **Inadequate Access Control & Privilege Escalation**
   Risk: Overly permissive roles or policies can allow unauthorized access or privilege escalation, increasing the attack surface.
   Mitigation Strategies:
   - **Role-Based Access Control (RBAC):** Implement AWS IAM to enforce the principle of least privilege.

    - **Multi-Factor Authentication (MFA):** Enforce MFA for users accessing AWS environments.
    - **Regular Reviews:** Regularly review IAM roles and permissions to ensure they meet evolving security requirements.
      Compliance Alignment:
    - **ISO 27001:** Requires strict access management and regular review of permissions.
    - **SOC 2:** Mandates controls to protect operational environments from unauthorized access.
    - **GDPR:** Least-privilege access helps mitigate the risk of unauthorized personal data exposure.
3. **Third-Party & Supply Chain Vulnerabilities**
   Risk: Using untrusted third-party libraries, container images, or external plugins can introduce vulnerabilities.
   Mitigation Strategies:
    - **Vulnerability Scanning:** Regularly scan dependencies and container images using tools like AWS Inspector or Amazon ECR.
    - **Trusted Registries and Code Signing:** Use trusted registries for container images and implement code signing.
    - **Update Policies:** Maintain an effective patch and update

Task 4 Monitoring & Logging

Steps to configure monitoring/logging tools

I have used Prometheus & Grafana
To set it up on the webapp used for task 1 and task 2, I made USERDATA of EC@ to install docker

add the configuration file for Promethus

```
global:
  scrape_interval: 4s

scrape_configs:
  - job_name: prometheus
```

```
    static_configs:
      - targets: ["10.0.0.36:80"]
```

and docker-compose file to run it with the given configurations

```
version: "3"

services:
  prom-server:
    image: prom/prometheus
    ports:
      - 9090:9090
    volumes:
      - ./prometheus-
config.yml:/etc/prometheus/prometheus.yml
```

then docker compose up and check

```
curl http://10.0.0.36:80/metrics
```

user http://:9090 to open Prometheus

For Grafana
run docker container

```
docker run -d -p 3000:3000 --name=grafana grafana/grafana-oss
```

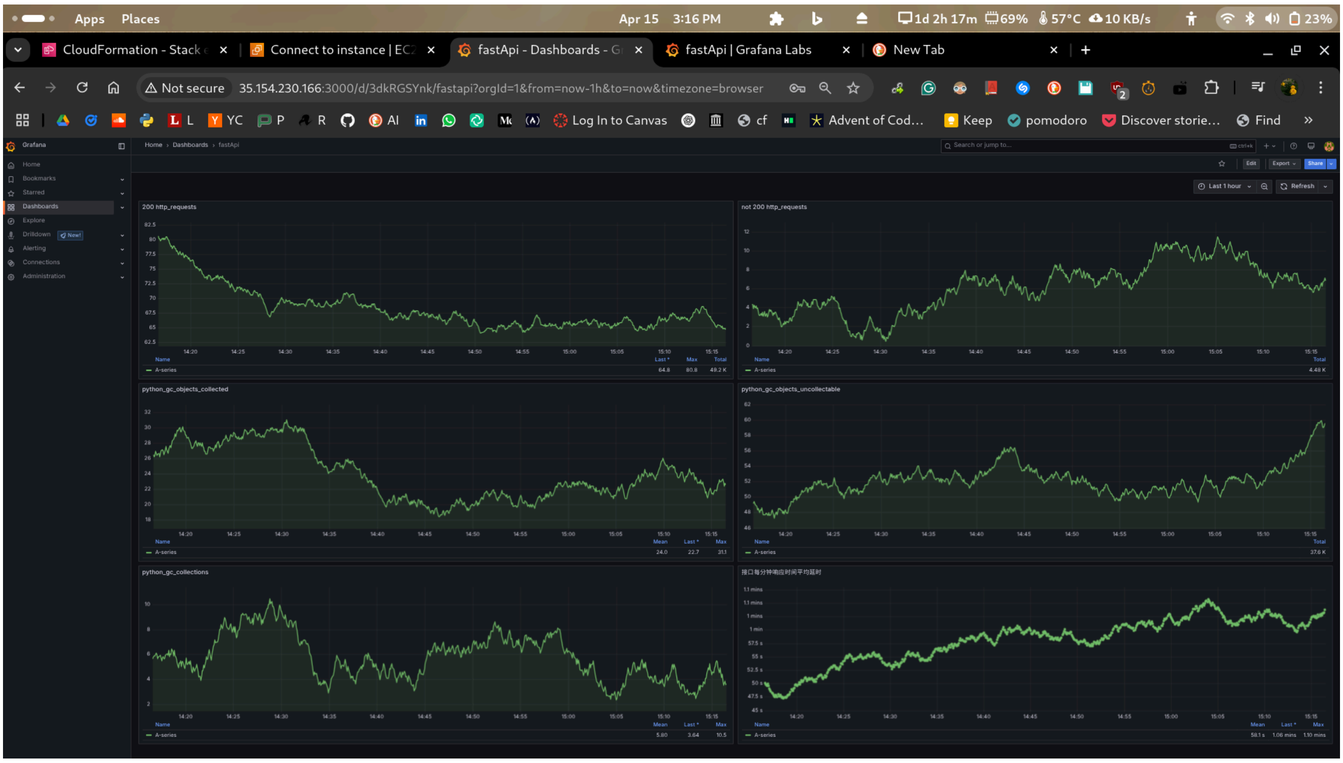
add data source to Prometheus
by adding

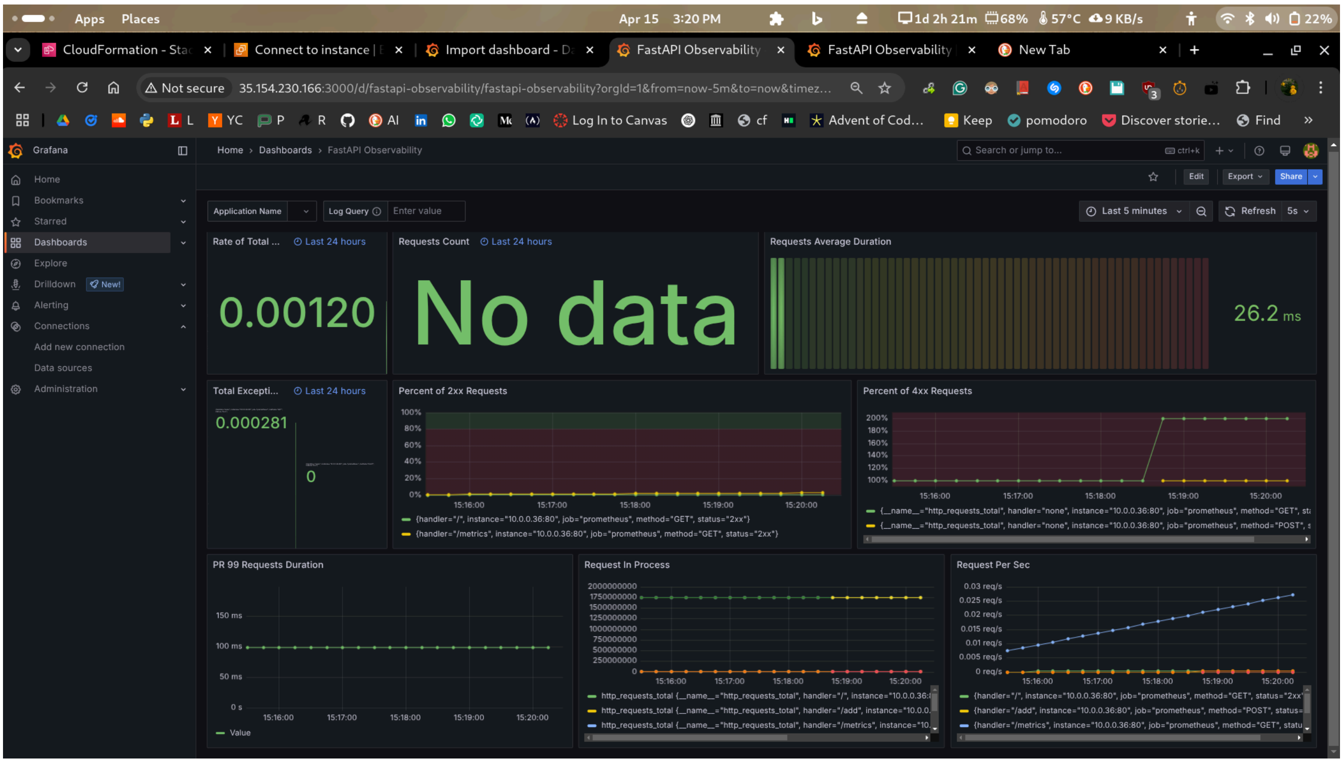
```
http://10.0.0.36:9090
```

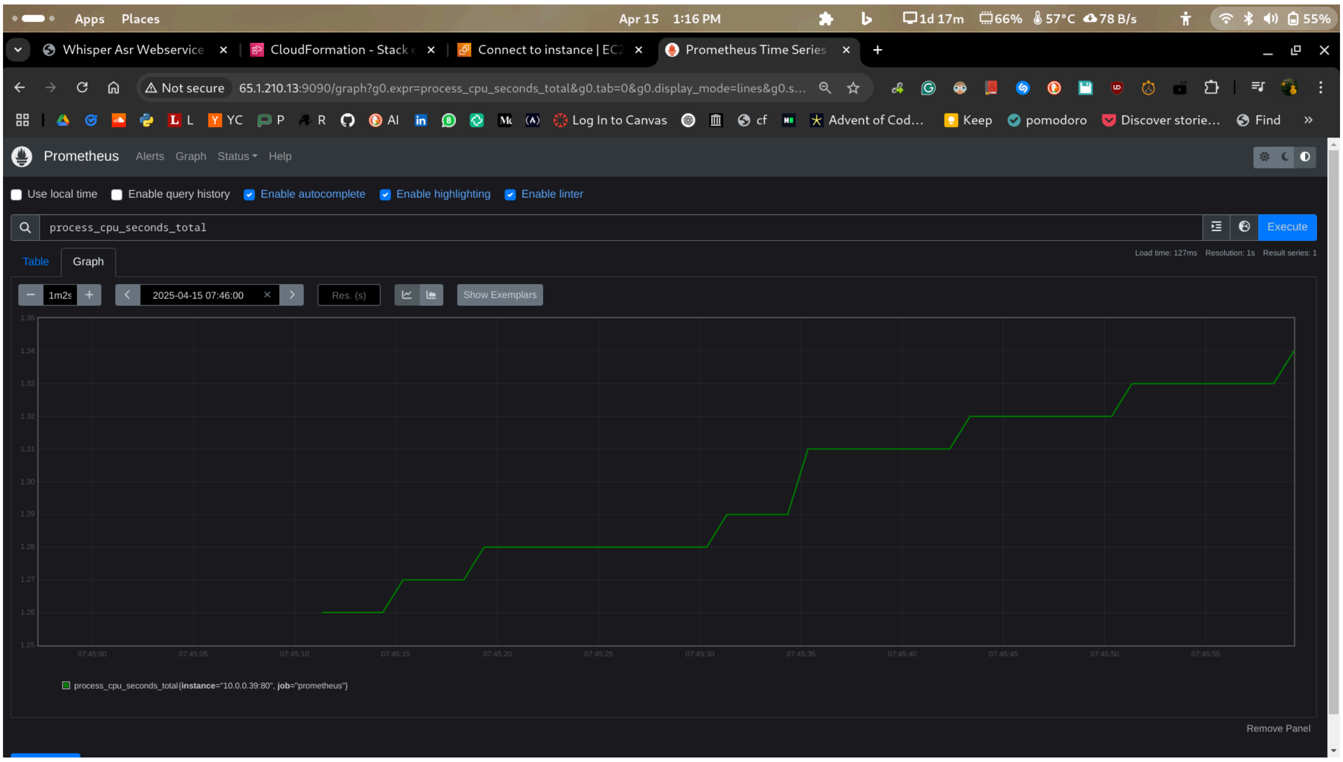
then login to grafana , and goto Dashboard > New dashboard > add id 15834

we can add another dashboard, add id 18739

Dashboard screenshots showing application metrics


fastApi - Dashboards - Grafana
35.154.230.166:3000/d/3dkRGSYnk/fastapi?orgId=1&from=now-1h&to=now&timezone=browser
Home › Dashboards › fastApi
Last 1 hour
Refresh
200 http_requests
not 200 http_requests
python_gc_objects_collected
python_gc_objects_uncollectable
python_gc_collections
接口每分钟响应时间平均耗时
A-series





Not secure 65.1.210.13:9090/graph?g0.expr=process_cpu_seconds_total&g0.tab=0&g0.display_mode=lines&g0.s...

□ Use local time □ Enable query history ☑ Enable autocomplete ☑ Enable highlighting ☑ Enable linter

```
process_cpu_seconds_total
```

Load time: 127ms Resolution: 1s Result series: 1

1m2s 2025-04-15 07:46:00

🟩 process_cpu_seconds_total{**instance**="10.0.0.39:80", **job**="prometheus"}

Task5 Database & Storage Optimization

Overview

Database optimization enhances query performance, resource utilization, and overall reliability. These techniques are applicable to AWS-managed databases like Amazon RDS or Amazon DocumentDB.

Optimization Techniques

1. **Indexing**
   Purpose: Improve data retrieval times by reducing the amount of data scanned during query execution.
   Implementation: Identify columns frequently used in query `WHERE` or `ORDER BY` clauses and create indexes on them.
   Example (PostgreSQL):
   Before Indexing:

   ```
   SELECT * FROM orders WHERE customer_id = '12345' ORDER BY
   order_date DESC;
   ```

   After Indexing:

```sql
CREATE INDEX idx_orders_customer_date ON orders
(customer_id, order_date DESC);
```

Benefit: This composite index allows direct access to relevant rows, improving query efficiency.

2. **Query Optimization**

   Purpose: Rewrite queries to eliminate redundant computations and optimize join logic.
   Implementation: Use tools like PostgreSQL's `EXPLAIN ANALYZE` to identify performance bottlenecks.
   Example:
   Before (Using Subquery):

   ```sql
   SELECT * FROM orders
   WHERE customer_id IN (SELECT id FROM customers WHERE
   region = 'West');
   ```

   After (Using JOIN):

   ```sql
   SELECT o.*
   FROM orders o
   INNER JOIN customers c ON o.customer_id = c.id
   WHERE c.region = 'West';
   ```

   Benefit: Better performance due to more efficient join algorithm selection.

3. **Data Partitioning**

   Purpose: Divide large tables into smaller, more manageable pieces to improve performance.
   Implementation: Partition data based on a specific key, such as date ranges or categorical values.
   Example (PostgreSQL - Range Partitioning):

   ```sql
   -- Define the parent partitioned table
   CREATE TABLE orders (
     order_id serial NOT NULL,
   ```

```
  customer_id int NOT NULL,
  order_date date NOT NULL,
  -- other columns
  PRIMARY KEY (order_id, order_date)
) PARTITION BY RANGE (order_date);

-- Create partitions
CREATE TABLE orders_2024 PARTITION OF orders
FOR VALUES FROM ('2024-01-01') TO ('2025-01-01');

CREATE TABLE orders_2025 PARTITION OF orders
FOR VALUES FROM ('2025-01-01') TO ('2026-01-01');
```

Benefit: Queries can scan only relevant partitions, significantly improving performance.

Task 6 Automation & Scripting

Script file (.sh or .py)

```
#!/bin/bash
# deploy the cloudformation file
aws cloudformation deploy --region ap-south-1 \

--template-file ./main.yaml \

--stack-name ec2forinterview \

--tags madeFromCLI=yeah anotherTagForAllStackResources=okay \

--capabilities CAPABILITY_NAMED_IAM

#--no-execute-changeset
```

Explanation of what the script does

This simple commands build and deploy the cloudfortaion template, the templete contains the user data

```bash
#!/bin/bash

dnf update -y

dnf install httpd git python pip -y

yum install docker -y

systemctl start docker

systemctl enable docker

usermod -aG docker $USER

mkdir -p /usr/local/lib/docker/cli-plugins

curl -SL https://github.com/docker/compose/releases/latest/download/docker-compose-linux-x86_64 -o /usr/local/lib/docker/cli-plugins/docker-compose

chmod +x /usr/local/lib/docker/cli-plugins/docker-compose

cd /home/ec2-user

git clone https://github.com/AryanGitHub/a-very-simple-webapp-for-assignment.git 2> error.log

cd a-very-simple-webapp-for-assignment

bash bash.sh 2> error_bash.log
```

bash file in the webapp folder

```bash
#!/bin/bash
python -m venv .venv
source .venv/bin/activate
pip install -r ./requirements.txt
pip install prometheus-fastapi-instrumentator
uvicorn main:app --host 0.0.0.0 --port 80 --reload
```

it installs the python , git , pip, docker and docker compose
then it clones the repo
and starts running the bash script which runs the webapp, the script download all the packages from requiremet.txt file in an virtual environment and then runs the uvicorn command

Task7 Disaster Recovery & High Availability

Key Elements of a DR Strategy

1. **Recovery Time Objective (RTO)**
   Definition: Maximum acceptable downtime duration before service restoration.
   Example Target: 2 hours (system must be fully functional within 2 hours of incident).
2. **Recovery Point Objective (RPO)**
   Definition: Maximum acceptable data loss measured in time before failure.
   Example Target: 15 minutes (data loss should not exceed the last 15 minutes of transactions).
3. **Backup Strategy**
   - **Automated Backups:** Schedule regular backups using AWS Backup or native service features.
   - **Geographical Redundancy:** Store critical backups in a separate AWS Region.
   - **Testing and Validation:** Regularly test restore procedures to validate RTO and RPO targets.
   - **Backup Security:** Encrypt all backups using AWS KMS and restrict access using IAM policies.
4. **High Availability (HA) Implementation**
   - **Multi-AZ Deployments:** Deploy across multiple Availability Zones within an AWS Region.

- **Replication:** Utilize synchronous or near-synchronous replication for critical data.
- **Load Balancing:** Use Elastic Load Balancing to distribute traffic across healthy instances.
- **Automated Failover:** Configure services like Amazon RDS Multi-AZ for automatic failover.

Example: Automated Backup Setup using AWS Backup

1. **Create a Backup Vault:**

```
aws backup create-backup-vault \
    --backup-vault-name MyBackupVault \
    --region us-east-1
```

2. **Define and Assign a Backup Plan:**
   Create a JSON file (backup-plan.json):

```
{
    "BackupPlanName": "DailyBackupPlan",
    "Rules": [
        {
            "RuleName": "DailyFullBackup",
            "TargetBackupVaultName": "MyBackupVault",
            "ScheduleExpression": "cron(0 2 * * ? *)",
            "StartWindowMinutes": 60,
            "CompletionWindowMinutes": 180,
            "Lifecycle": {
                "DeleteAfterDays": 30
            }
        }
    ]
}
```

Create the backup plan:

```
aws backup create-backup-plan --backup-plan file://backup-plan.json
```

Assign resources:

```
aws backup create-backup-selection \
    --backup-plan-id <your-backup-plan-id> \
    --backup-selection '{
        "SelectionName": "EC2Selection",
        "IamRoleArn": "arn:aws:iam::<account-id>:role/service-role/AWSBackupDefaultServiceRole",
        "Resources": [
            "arn:aws:ec2:us-east-1:<account-id>:instance/<instance-id>"
        ]
    }' \
    --creator-request-id $(uuidgen)
```

Example: Setting Up Automated Backups in AWS

AWS offers automated backup solutions integrated with many of its services. For instance, using AWS Backup you can centralize the backup of EC2 instances, RDS databases, and more. Below is an example using AWS CLI commands:

1. **Create a Backup Vault:**

   ```
   aws backup create-backup-vault \
       --backup-vault-name MyBackupVault \
       --region us-east-1
   ```

   This command creates a backup vault in the specified region to store your backup data.
2. **Create and Assign a Backup Plan:**
   First, create a JSON file (e.g., backup-plan.json) with your backup plan details:

```
{
    "BackupPlanName": "DailyBackupPlan",
    "Rules": [
        {
            "RuleName": "DailyFullBackup",
            "TargetBackupVaultName": "MyBackupVault",
            "ScheduleExpression": "cron(0 2 * * ? *)",
            "StartWindowMinutes": 60,
            "CompletionWindowMinutes": 180,
            "Lifecycle": {
                "DeleteAfterDays": 30
            }
        }
    ]
}
```

Then, apply the backup plan and assign resources (for example, an EC2 instance):

```
# Create the backup plan
aws backup create-backup-plan --backup-plan file://backup-plan.json

# Assign resources (example for EC2, ensuring the resource ARN is correct)
aws backup create-backup-selection \
    --backup-plan-id <your-backup-plan-id> \
    --backup-selection '{
        "SelectionName": "EC2Selection",
        "IamRoleArn": "arn:aws:iam::<account-id>:role/AWSBackupDefaultServiceRole",
        "Resources": ["arn:aws:ec2:us-east-1:<account-id>:instance/<instance-id>"]
    }'
```

This setup schedules a daily full backup for the

Task8 AI Model Deployment & MLOps

We have used ECS to host the docker task, and added ALB (Application load balancer to the publically exposed docker service)
This all is achieved using cloudformation template and the ECS Fargate.

The docker service takes the voice sound file and convet it into test.

The famous docker image used for this is onerahmet/openai-whisper-asr-webservice:latest

Dockerfile & Kubernetes YAML files

Cloudformation File

```
AWSTemplateFormatVersion: '2010-09-09'

Description: Deploy Whisper ASR API to ECS Fargate
Parameters:

WhisperModel:

Type: String

Default: tiny

AllowedValues: [tiny, base, small, medium, large]

Description: Whisper model to use



Resources:



WhisperVPC:

Type: AWS::EC2::VPC

Properties:
```

```
CidrBlock: 10.0.0.0/16
EnableDnsSupport: true
EnableDnsHostnames: true
Tags: [{ Key: Name, Value: WhisperVPC }]

WhisperSubnet1:
Type: AWS::EC2::Subnet
Properties:
VpcId: !Ref WhisperVPC
CidrBlock: 10.0.1.0/24
AvailabilityZone: !Select [0, !GetAZs '']
MapPublicIpOnLaunch: true

WhisperSubnet2:
Type: AWS::EC2::Subnet
Properties:
VpcId: !Ref WhisperVPC
CidrBlock: 10.0.2.0/24
AvailabilityZone: !Select [1, !GetAZs '']
MapPublicIpOnLaunch: true
```

```
WhisperInternetGateway:
Type: AWS::EC2::InternetGateway

WhisperAttachGateway:
Type: AWS::EC2::VPCGatewayAttachment
Properties:
VpcId: !Ref WhisperVPC
InternetGatewayId: !Ref WhisperInternetGateway

WhisperRouteTable:
Type: AWS::EC2::RouteTable
Properties:
VpcId: !Ref WhisperVPC

WhisperRoute:
Type: AWS::EC2::Route
DependsOn: WhisperAttachGateway
Properties:
RouteTableId: !Ref WhisperRouteTable
```

```
DestinationCidrBlock: 0.0.0.0/0
GatewayId: !Ref WhisperInternetGateway

WhisperSubnetRouteTableAssoc1:
Type: AWS::EC2::SubnetRouteTableAssociation
Properties:
SubnetId: !Ref WhisperSubnet1
RouteTableId: !Ref WhisperRouteTable

WhisperSubnetRouteTableAssoc2:
Type: AWS::EC2::SubnetRouteTableAssociation
Properties:
SubnetId: !Ref WhisperSubnet2
RouteTableId: !Ref WhisperRouteTable

WhisperSecurityGroup:
Type: AWS::EC2::SecurityGroup
Properties:
GroupDescription: Allow HTTP access
VpcId: !Ref WhisperVPC
```

```
SecurityGroupIngress:
- IpProtocol: tcp
FromPort: 9000
ToPort: 9000
CidrIp: 0.0.0.0/0

WhisperCluster:
Type: AWS::ECS::Cluster

WhisperTaskExecutionRole:
Type: AWS::IAM::Role
Properties:
AssumeRolePolicyDocument:
Statement:
- Effect: Allow
Principal:
Service: ecs-tasks.amazonaws.com
Action: sts:AssumeRole
ManagedPolicyArns:
- arn:aws:iam::aws:policy/service-
```

```yaml
role/AmazonECSTaskExecutionRolePolicy

WhisperTaskDefinition:
Type: AWS::ECS::TaskDefinition
Properties:
Family: whisper-task
RequiresCompatibilities: [FARGATE]
Cpu: 512
Memory: 1024
NetworkMode: awsvpc
ExecutionRoleArn: !GetAtt WhisperTaskExecutionRole.Arn
ContainerDefinitions:
- Name: whisper
Image: onerahmet/openai-whisper-asr-webservice:latest
PortMappings:
- ContainerPort: 9000
Environment:
- Name: ASR_MODEL
Value: !Ref WhisperModel
```

```
WhisperService:
Type: AWS::ECS::Service
DependsOn: WhisperALBListener
Properties:
Cluster: !Ref WhisperCluster
LaunchType: FARGATE
DesiredCount: 1
NetworkConfiguration:
AwsvpcConfiguration:
AssignPublicIp: ENABLED
SecurityGroups: [!Ref WhisperSecurityGroup]
Subnets: [!Ref WhisperSubnet1, !Ref WhisperSubnet2]
TaskDefinition: !Ref WhisperTaskDefinition
LoadBalancers:
- ContainerName: whisper
ContainerPort: 9000
TargetGroupArn: !Ref WhisperTargetGroup

WhisperALB:
Type: AWS::ElasticLoadBalancingV2::LoadBalancer
```

```yaml
Properties:

Name: whisper-alb

Subnets: [!Ref WhisperSubnet1, !Ref WhisperSubnet2]

SecurityGroups: [!Ref WhisperSecurityGroup]

Scheme: internet-facing

Type: application



WhisperTargetGroup:

Type: AWS::ElasticLoadBalancingV2::TargetGroup

Properties:

Port: 9000

Protocol: HTTP

VpcId: !Ref WhisperVPC

TargetType: ip

HealthCheckPath: /docs



WhisperALBListener:

Type: AWS::ElasticLoadBalancingV2::Listener

Properties:

LoadBalancerArn: !Ref WhisperALB
```

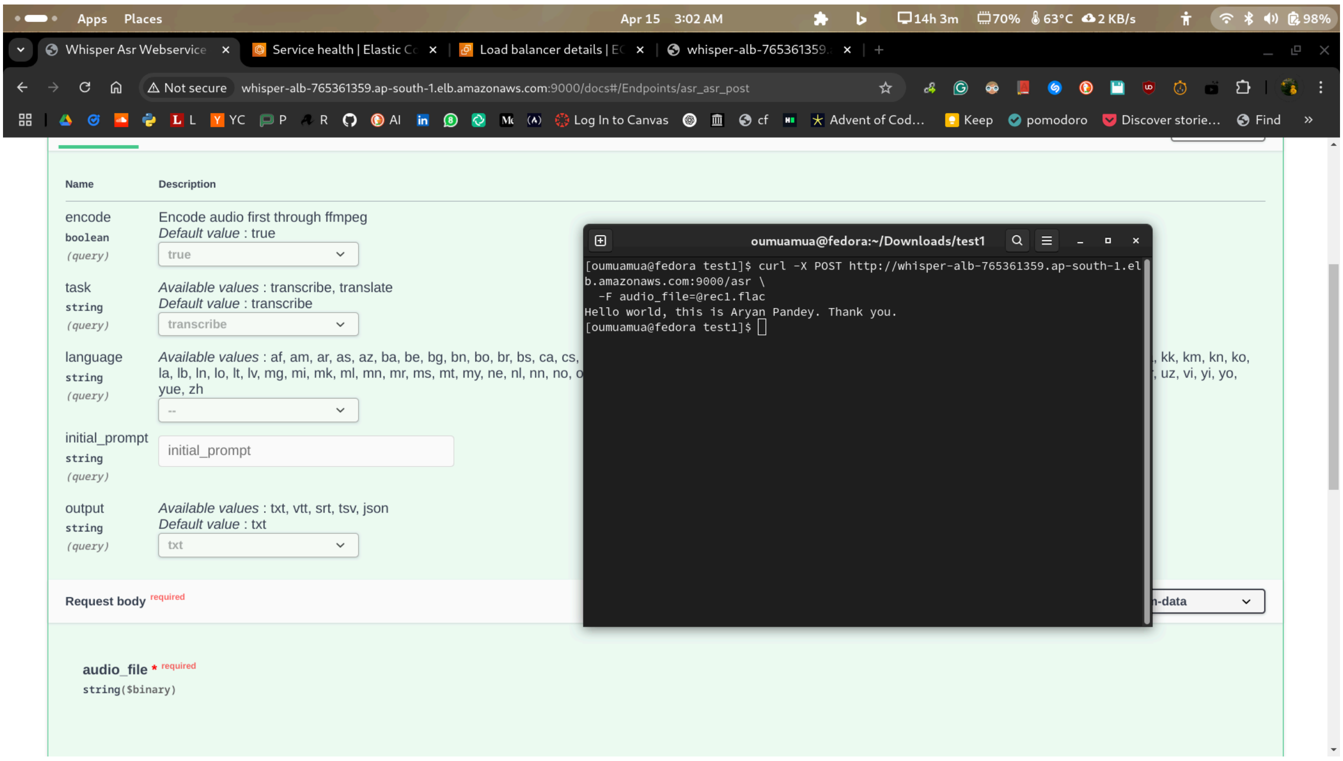
```
Port: 9000

Protocol: HTTP

DefaultActions:

- Type: forward

TargetGroupArn: !Ref WhisperTargetGroup



Outputs:

WhisperAPIURL:

Description: Whisper REST API URL

Value: !Join ["", ["http://", !GetAtt WhisperALB.DNSName,
":9000"]]
```

Steps to deploy the model

deploy this using AWS CLI deploy command

```
#!/bin/bash

aws cloudformation deploy --region ap-south-1 \

--template-file ./main.yaml \

--stack-name ecsaimodel \

--tags madeFromCLI=yeah anotherTagForAllStackResources=okay \

--capabilities CAPABILITY_NAMED_IAM

#--no-execute-changeset
```

Screenshot of the model running on ECS

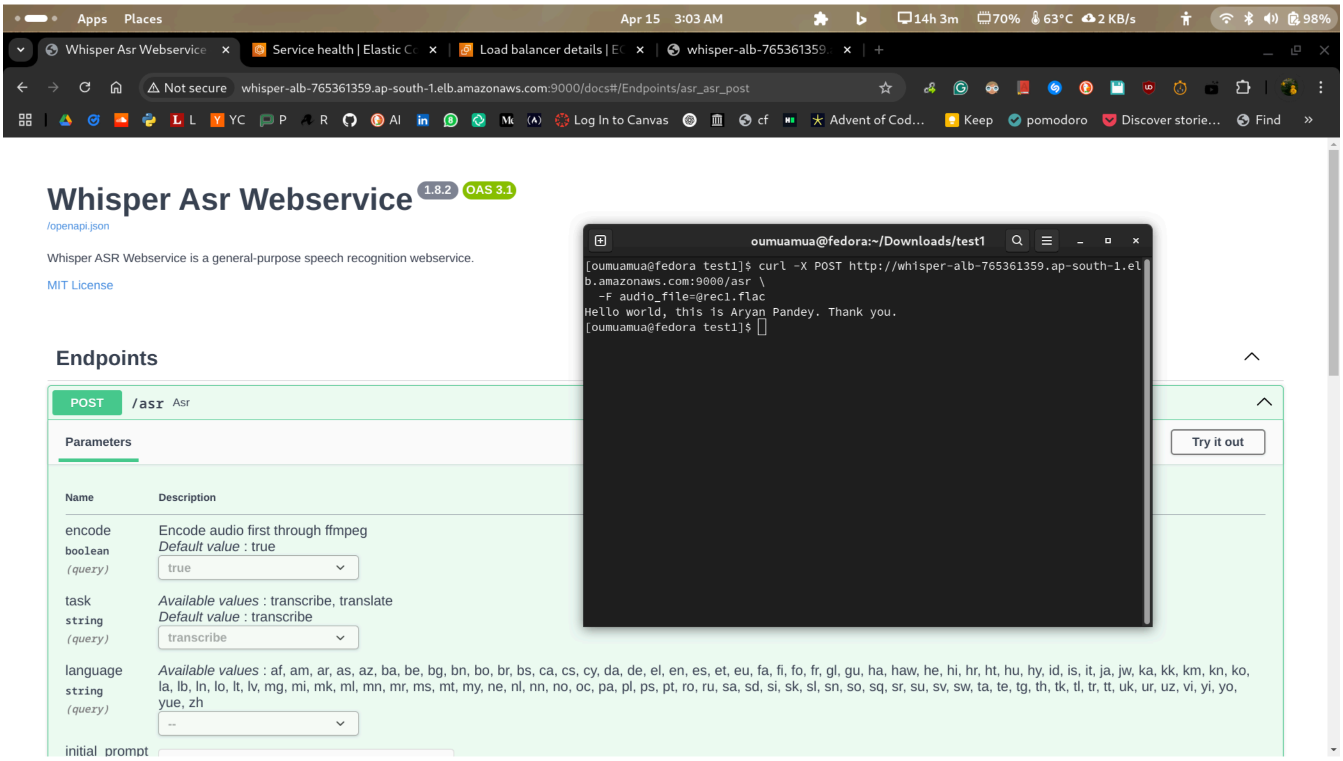


```
[oumuamua@fedora test1]$ curl -X POST http://whisper-alb-765361359.ap-south-1.elb.amazonaws.com:9000/asr \
  -F audio_file=@rec1.flac
Hello world, this is Aryan Pandey. Thank you.
[oumuamua@fedora test1]$
```

Name	Description
encode boolean *(query)*	Encode audio first through ffmpeg *Default value* : true true
task string *(query)*	*Available values* : transcribe, translate *Default value* : transcribe transcribe
language string *(query)*	*Available values* : af, am, ar, as, az, ba, be, bg, bn, bo, br, bs, ca, cs, … , kk, km, kn, ko, la, lb, ln, lo, lt, lv, mg, mi, mk, ml, mn, mr, ms, mt, my, ne, nl, nn, no, o… , uz, vi, yi, yo, yue, zh --
initial_prompt string *(query)*	initial_prompt
output string *(query)*	*Available values* : txt, vtt, srt, tsv, json *Default value* : txt txt

Request body required

audio_file * required
string($binary)



Whisper Asr Webservice 1.8.2 OAS 3.1

/openapi.json

Whisper ASR Webservice is a general-purpose speech recognition webservice.

MIT License

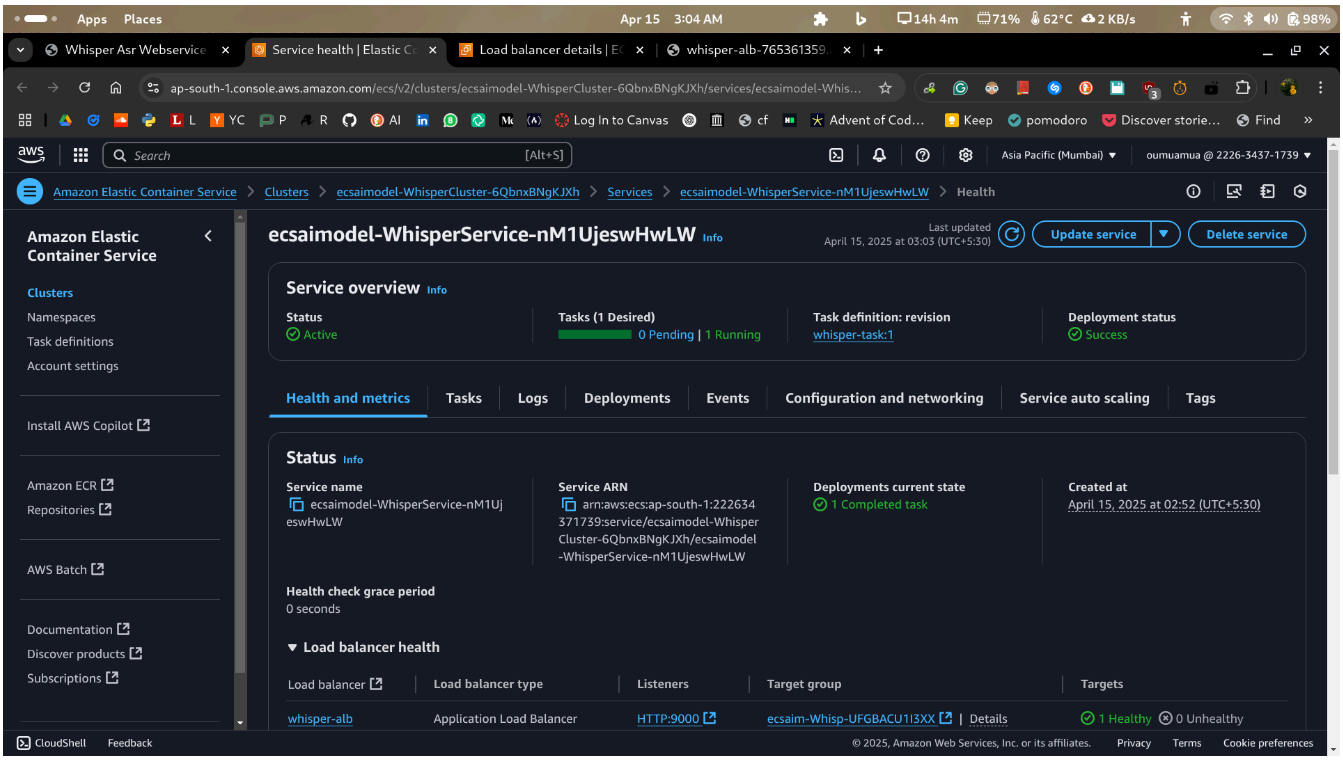
```
[oumuamua@fedora test1]$ curl -X POST http://whisper-alb-765361359.ap-south-1.elb.amazonaws.com:9000/asr \
  -F audio_file=@rec1.flac
Hello world, this is Aryan Pandey. Thank you.
[oumuamua@fedora test1]$
```

Endpoints

POST /asr Asr

Parameters

Name	Description
encode boolean *(query)*	Encode audio first through ffmpeg *Default value* : true true
task string *(query)*	*Available values* : transcribe, translate *Default value* : transcribe transcribe
language string *(query)*	*Available values* : af, am, ar, as, az, ba, be, bg, bn, bo, br, bs, ca, cs, cy, da, de, el, en, es, et, eu, fa, fi, fo, fr, gl, gu, ha, haw, he, hi, hr, ht, hu, hy, id, is, it, ja, jw, ka, kk, km, kn, ko, la, lb, ln, lo, lt, lv, mg, mi, mk, ml, mn, mr, ms, mt, my, ne, nl, nn, no, oc, pa, pl, ps, pt, ro, ru, sa, sd, si, sk, sl, sn, so, sq, sr, su, sv, sw, ta, te, tg, th, tk, tl, tr, tt, uk, ur, uz, vi, yi, yo, yue, zh --
initial_prompt	


Apps Places Apr 15 3:04 AM 14h 4m 71% 62°C 2 KB/s 98%
Whisper Asr Webservice
Service health | Elastic Co
Load balancer details | EC
whisper-alb-765361359.
ap-south-1.console.aws.amazon.com/ecs/v2/clusters/ecsaimodel-WhisperCluster-6QbnxBNgKJXh/services/ecsaimodel-Whis...
Search [Alt+S]
Asia Pacific (Mumbai)
oumuamua @ 2226-3437-1739
Amazon Elastic Container Service > Clusters > ecsaimodel-WhisperCluster-6QbnxBNgKJXh > Services > ecsaimodel-WhisperService-nM1UjeswHwLW > Health
Amazon Elastic Container Service
Clusters
Namespaces
Task definitions
Account settings
Install AWS Copilot
Amazon ECR
Repositories
AWS Batch
Documentation
Discover products
Subscriptions
Update service
Delete service
Health and metrics
Tasks
Logs
Deployments
Events
Configuration and networking
Service auto scaling
Tags
CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

ecsaimodel-WhisperService-nM1UjeswHwLW Info

Last updated
April 15, 2025 at 03:03 (UTC+5:30)

Service overview Info

Status
Active

Tasks (1 Desired)
0 Pending | 1 Running

Task definition: revision
whisper-task:1

Deployment status
Success

Status Info

Service name
ecsaimodel-WhisperService-nM1UjeswHwLW

Service ARN
arn:aws:ecs:ap-south-1:222634371739:service/ecsaimodel-WhisperCluster-6QbnxBNgKJXh/ecsaimodel-WhisperService-nM1UjeswHwLW

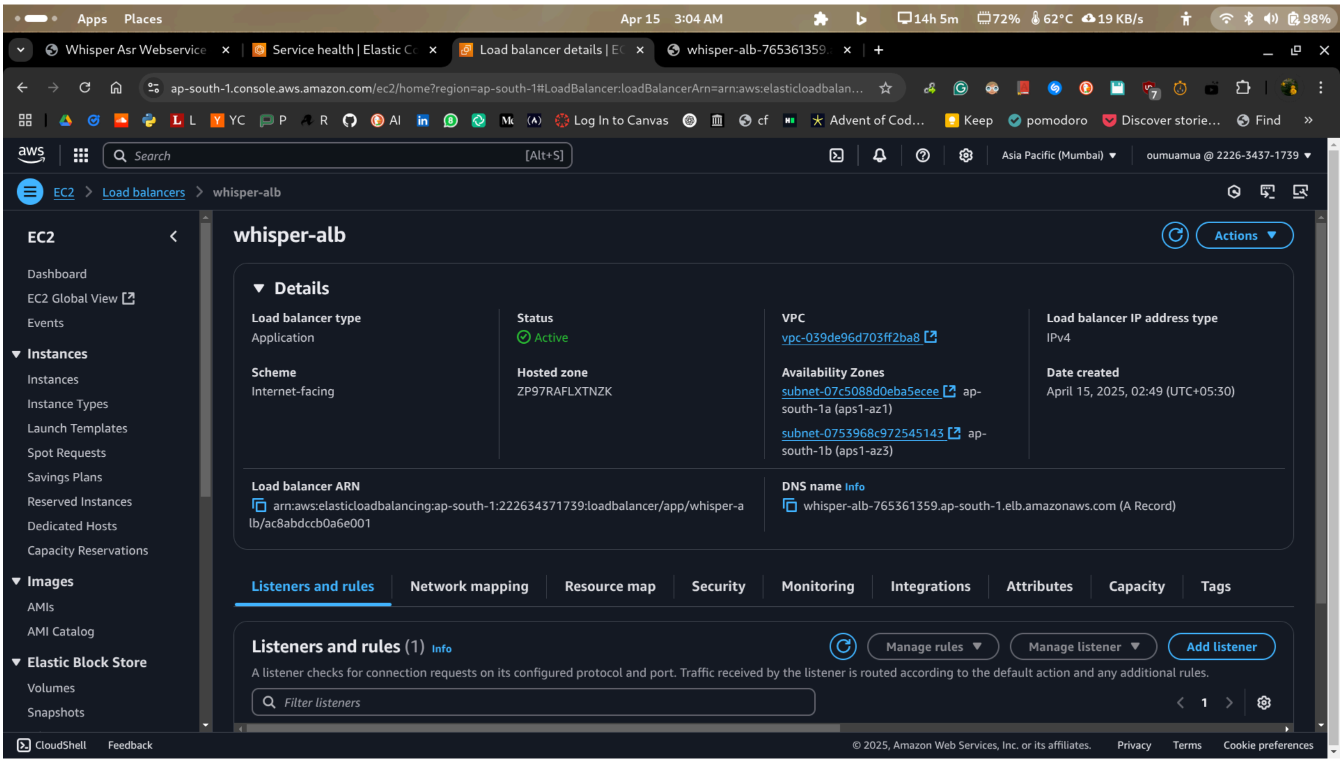
Deployments current state
1 Completed task

Created at
April 15, 2025 at 02:52 (UTC+5:30)

Health check grace period
0 seconds

Load balancer health

Load balancer	Load balancer type	Listeners	Target group	Targets
whisper-alb	Application Load Balancer	HTTP:9000	ecsaim-Whisp-UFGBACU1I3XX | Details	1 Healthy 0 Unhealthy



whisper-alb

Details

Load balancer type
Application

Status
Active

VPC
vpc-039de96d703ff2ba8

Load balancer IP address type
IPv4

Scheme
Internet-facing

Hosted zone
ZP97RAFLXTNZK

Availability Zones
subnet-07c5088d0eba5ecee ap-south-1a (aps1-az1)
subnet-0753968c972545143 ap-south-1b (aps1-az3)

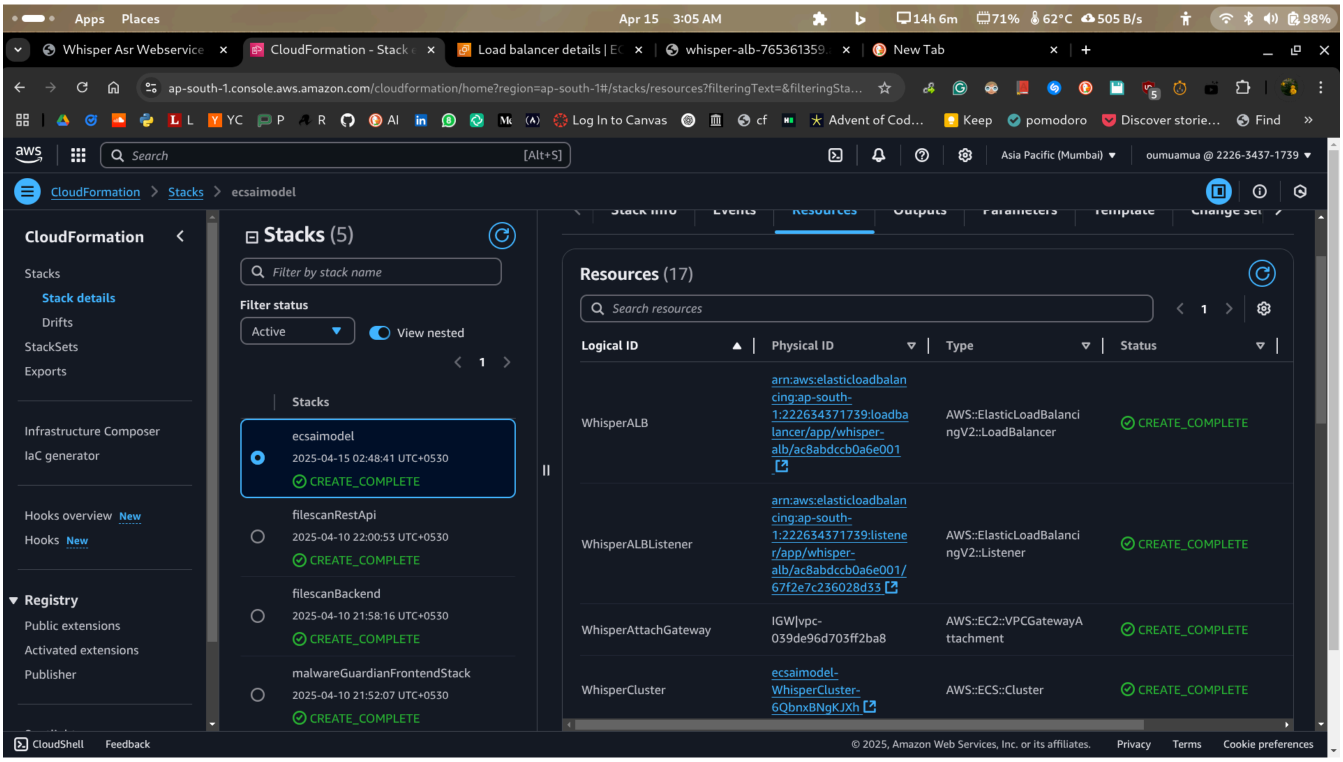
Date created
April 15, 2025, 02:49 (UTC+05:30)

Load balancer ARN
arn:aws:elasticloadbalancing:ap-south-1:222634371739:loadbalancer/app/whisper-alb/ac8abdccb0a6e001

DNS name Info
whisper-alb-765361359.ap-south-1.elb.amazonaws.com (A Record)

Listeners and rules (1) Info

A listener checks for connection requests on its configured protocol and port. Traffic received by the listener is routed according to the default action and any additional rules.



CloudFormation > Stacks > ecsaimodel

Stacks (5)

Filter by stack name

Filter status: Active

View nested

Stacks

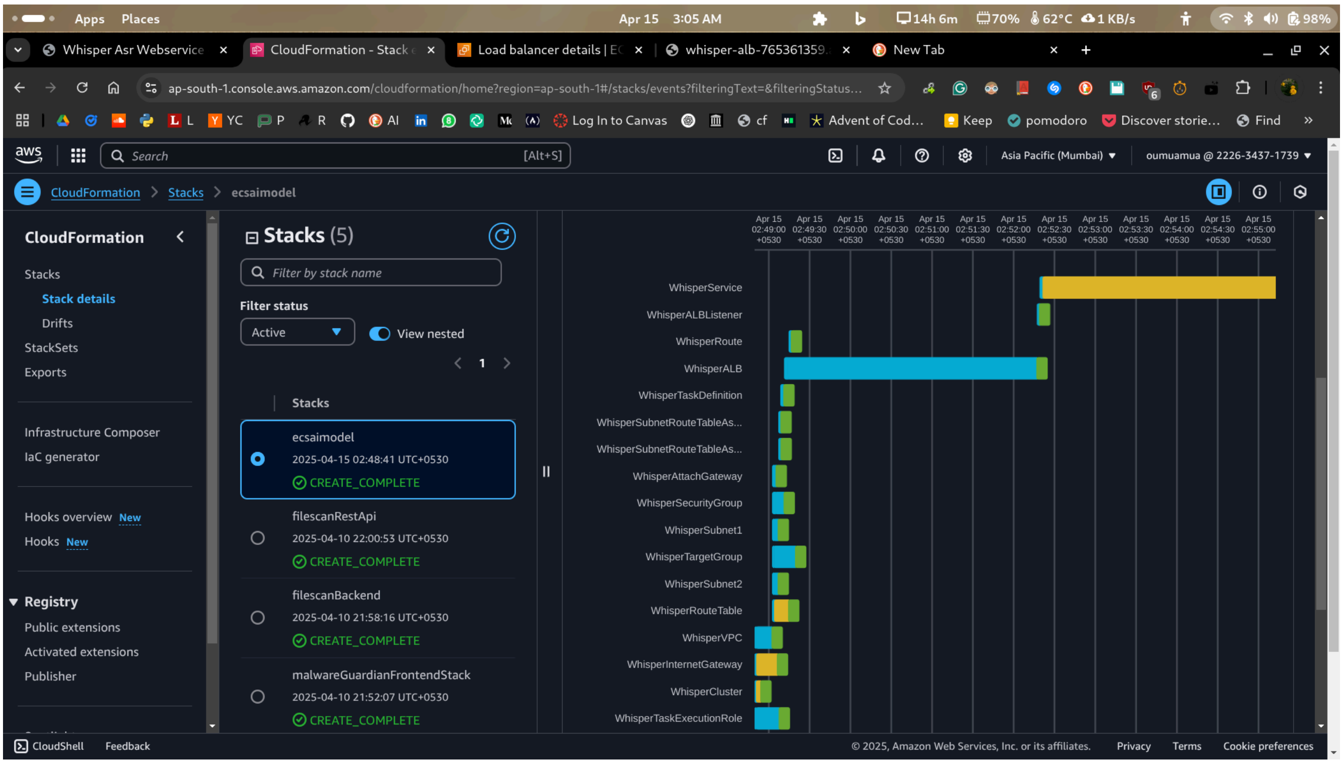
Stacks
● ecsaimodel 2025-04-15 02:48:41 UTC+0530 CREATE_COMPLETE
○ filescanRestApi 2025-04-10 22:00:53 UTC+0530 CREATE_COMPLETE
○ filescanBackend 2025-04-10 21:58:16 UTC+0530 CREATE_COMPLETE
○ malwareGuardianFrontendStack 2025-04-10 21:52:07 UTC+0530 CREATE_COMPLETE

Resources (17)

Search resources

Logical ID	Physical ID	Type	Status
WhisperALB	arn:aws:elasticloadbalancing:ap-south-1:222634371739:loadbalancer/app/whisper-alb/ac8abdccb0a6e001	AWS::ElasticLoadBalancingV2::LoadBalancer	CREATE_COMPLETE
WhisperALBListener	arn:aws:elasticloadbalancing:ap-south-1:222634371739:listener/app/whisper-alb/ac8abdccb0a6e001/67f2e7c236028d33	AWS::ElasticLoadBalancingV2::Listener	CREATE_COMPLETE
WhisperAttachGateway	IGW|vpc-039de96d703ff2ba8	AWS::EC2::VPCGatewayAttachment	CREATE_COMPLETE
WhisperCluster	ecsaimodel-WhisperCluster-6QbnxBNgKJXh	AWS::ECS::Cluster	CREATE_COMPLETE

© 2025, Amazon Web Services, Inc. or its affiliates.



CloudFormation > Stacks > ecsaimodel

Stacks (5)

Filter by stack name

Filter status: Active

View nested

Stacks

Stacks
● ecsaimodel 2025-04-15 02:48:41 UTC+0530 CREATE_COMPLETE
○ filescanRestApi 2025-04-10 22:00:53 UTC+0530 CREATE_COMPLETE
○ filescanBackend 2025-04-10 21:58:16 UTC+0530 CREATE_COMPLETE
○ malwareGuardianFrontendStack 2025-04-10 21:52:07 UTC+0530 CREATE_COMPLETE

© 2025, Amazon Web Services, Inc. or its affiliates.