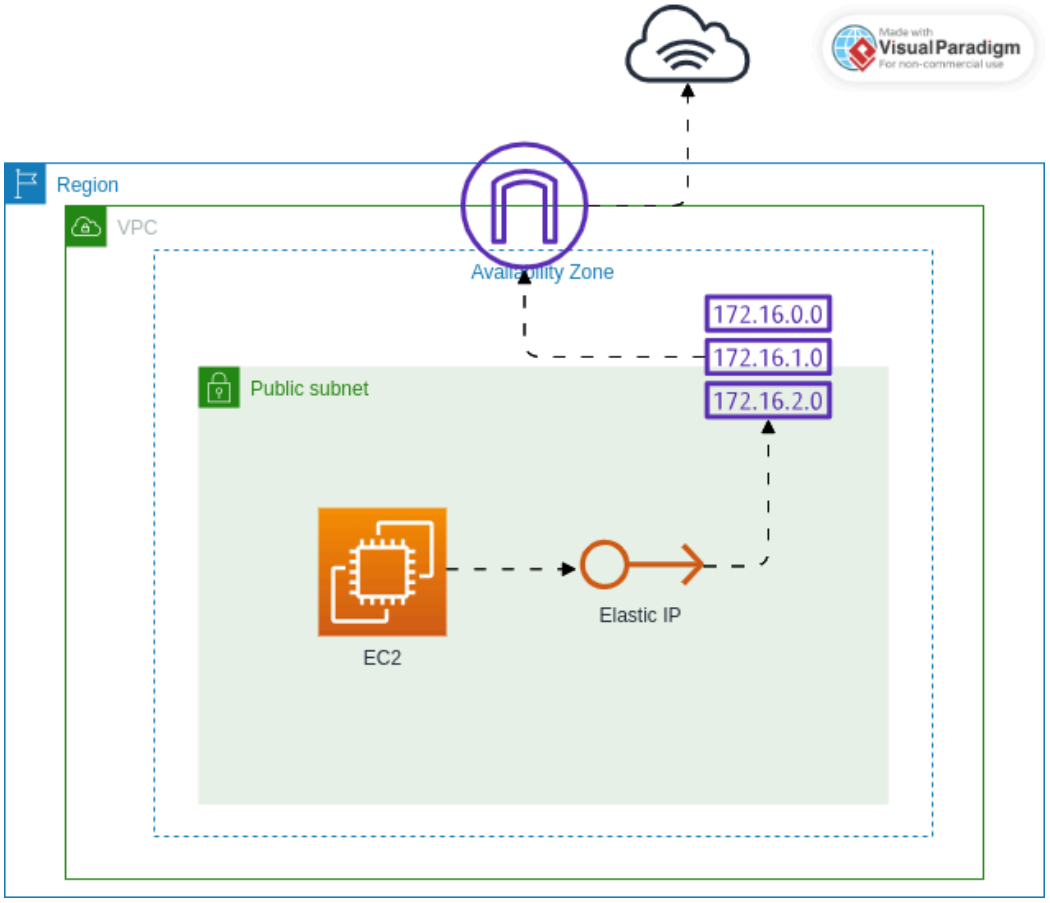
Task1

Task 1: Cloud Infrastructure & Deployment on AWS

Architecture Diagram



Steps to Deploy the Application

1. **Create an AWS Account**
2. **Create a CloudFormation Template**

   Refer to the structure from the AWS documentation: [CloudFormation Template Formats](#).

```
AWSTemplateFormatVersion: 2010-09-09
Description: Interview Test file
```

```yaml
Parameters:
  EC2InstanceSizeInput:
    Description: The supported instance sizes for EC2
    Type: String
    Default: t2.micro
    AllowedValues:
      - t3.micro
      - t2.micro

Resources:
  # VPC, subnet, IGW, route table, route 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
    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:
      VpcId: !Ref TestVPC
      InternetGatewayId: !Ref TestIGW
```

```yaml
  # 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
      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
    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
          CidrIp: 0
```

```
          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
      - TestEIP

  TestEC2Instance:
```

```
    Type: 'AWS::EC2::Instance'
    Properties:
      ImageId: ami-002f6e91abff6eb96 # ami-
053b12d3152c0cc71 for Ubuntu
      InstanceType: !Ref EC2InstanceSizeInput
      IamInstanceProfile: !Ref InstanceProfileOfRoleToEC2
      NetworkInterfaces:
        - Description: A Network interface made externally
with AWS EIP attached at startup 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
```

```yaml
  SSMEC2ControlRole:
    Type: AWS::IAM::Role
    Properties:
      Description: SSM Role for Test EC2
      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)
```

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>
```

Bash Script to Host the Application

```
#!/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
```

4. **Deploy CloudFormation Template Using AWS CLI Command**

This template contains USERDATA, so it will automatically pull the app from the GitHub repository.

```
#!/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
```

5. **AWS Configurations Used (Resource Groups, Networking, etc.)**

CloudFormation Resources Summary

VPC and Networking Resources

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

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

Security Resources

- **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`

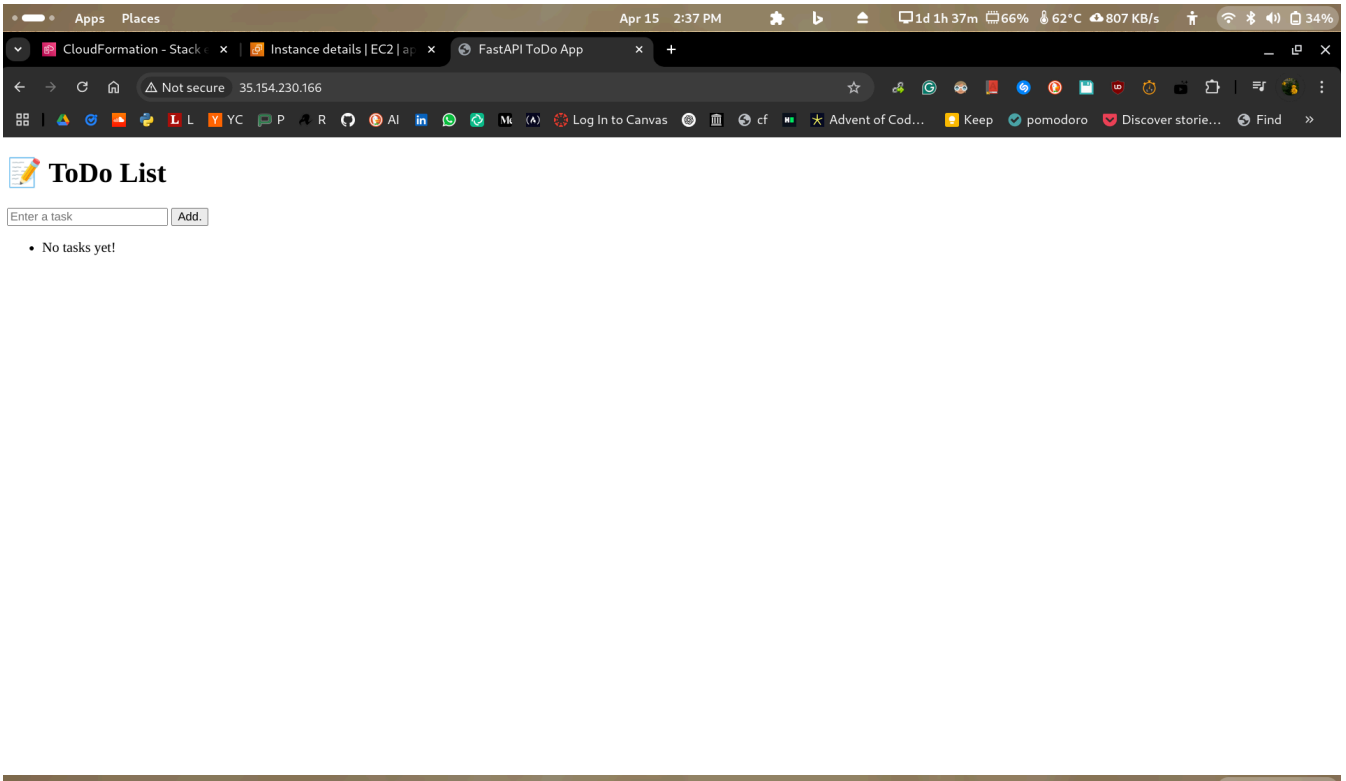
11. **EIP Association**
    - **Logical ID:** `TestEIPAssociation`
    - **Type:** `AWS::EC2::EIPAssociation`
    - **Properties:**
      - AllocationId: `!GetAtt TestEIP.AllocationId`
      - NetworkInterfaceId: `!Ref TestNetworkInterface`
12. **EC2 Instance**
    - **Logical ID:** `TestEC2Instance`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

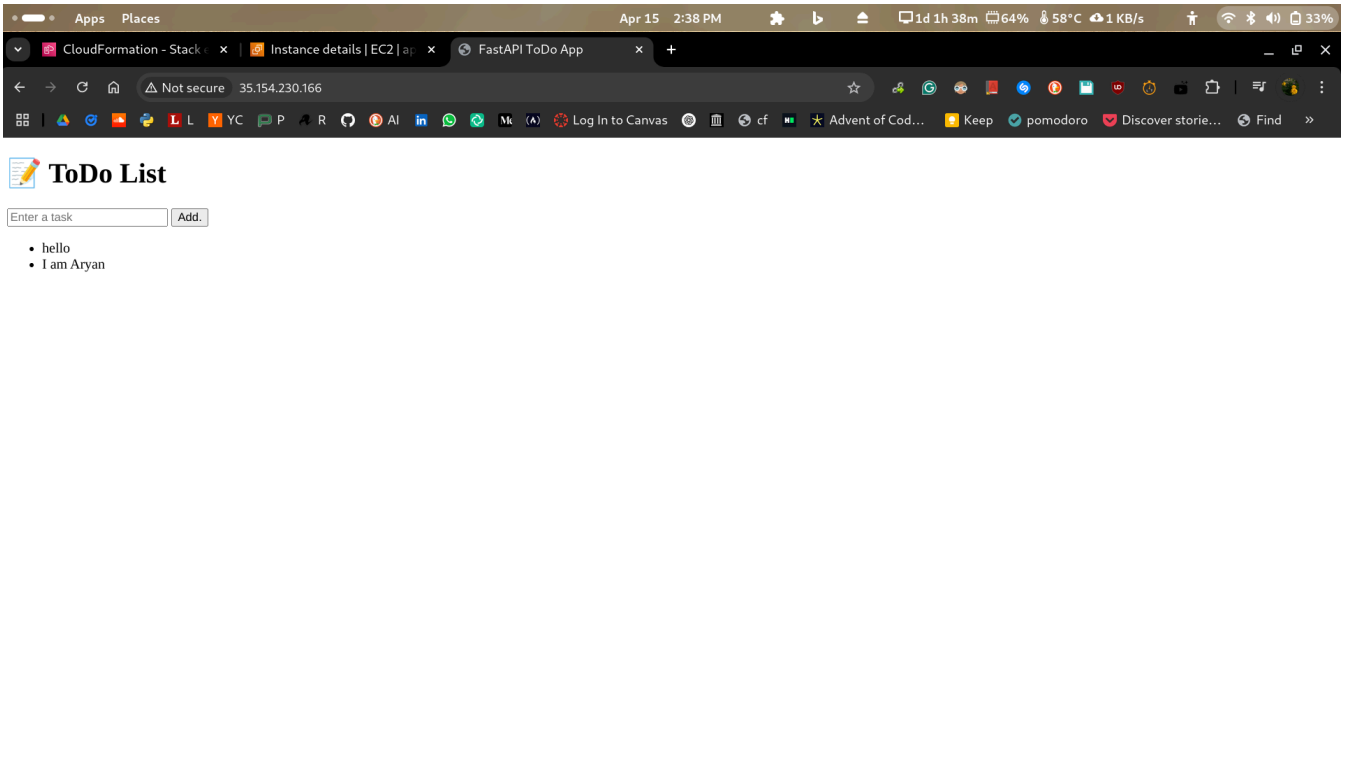
IAM Resources

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





Task2

Task 2: 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

In this implementation, I have used a single stage to deploy the changes to the EC2 instance that is hosting the web application. The GitHub Action `appleboy/ssh-action@v1` is utilized to SSH into the EC2 instance. It uses the contents of the PEM file stored in the GitHub repository secrets for deployment.

The script runs `git pull` from the origin because `uvicorn --reload` is used to automatically reload the application when changes in the repository are detected.

How Environment Variables/Secrets Are Managed

Environment variables are managed using GitHub Secrets for the entire repository. I have saved the following secrets:

- The public IP address of the EC2 instance (HOST).
- The contents of the PEM file used to log in to the EC2 machine (EC2_SSH_KEY).

TASK 3 Security & Compliance (ISO, GDPR, SOC 2)

Overview

Integrating operations into a continuous development process introduces various security challenges in guaranteeing a threat-free system. Addressing these compromises requires strategies consistent with internationally recognized frameworks, such as ISO 27001, GDPR, and 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 management process.

Best Practices in AWS Cloud Deployments

- **Data Encryption:** Encrypt data in transit using TLS/SSL and at rest using AWS KMS.
- **Logging & Monitoring:** Implement effective monitoring using AWS CloudWatch, X-Ray, and CloudTrail.
- **Patch Management:** Regularly apply patches using services like AWS Systems Manager.
- **Network Segmentation:** Use Amazon VPC, security groups, and NACLs to isolate critical resources.

Task4

Task 4: Monitoring & Logging

Steps to Configure Monitoring/Logging Tools

For this task, I have used **Prometheus** and **Grafana**. To set them up on the web application used for Task 1 and Task 2, I created a `USERDATA` script for the EC2 instance to install Docker.

Prometheus Configuration

1. **Add the Configuration File for Prometheus**

   Create a file named `prometheus-config.yml` with the following content:

   ```yaml
   global:
     scrape_interval: 4s

   scrape_configs:
     - job_name: prometheus
       static_configs:
         - targets: ["10.0.0.36:80"]
   ```

2. **Create a Docker Compose File**

   Create a `docker-compose.yml` file to run Prometheus with the given configurations:

   ```yaml
   version: "3"

   services:
     prom-server:
       image: prom/prometheus
       ports:
   ```

```
      - 9090:9090
    volumes:
      - ./prometheus-
config.yml:/etc/prometheus/prometheus.yml
```

3. **Start Prometheus**

   Run the following command to start Prometheus:

   ```
   docker-compose up
   ```

4. **Verify Prometheus Setup**

   Check if Prometheus is scraping metrics by running:

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

   You can access Prometheus at `http://<public-ip>:9090`.

Grafana Setup

1. **Run the Grafana Docker Container**

   Execute the following command to run Grafana:

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

2. **Add Data Source to Prometheus**

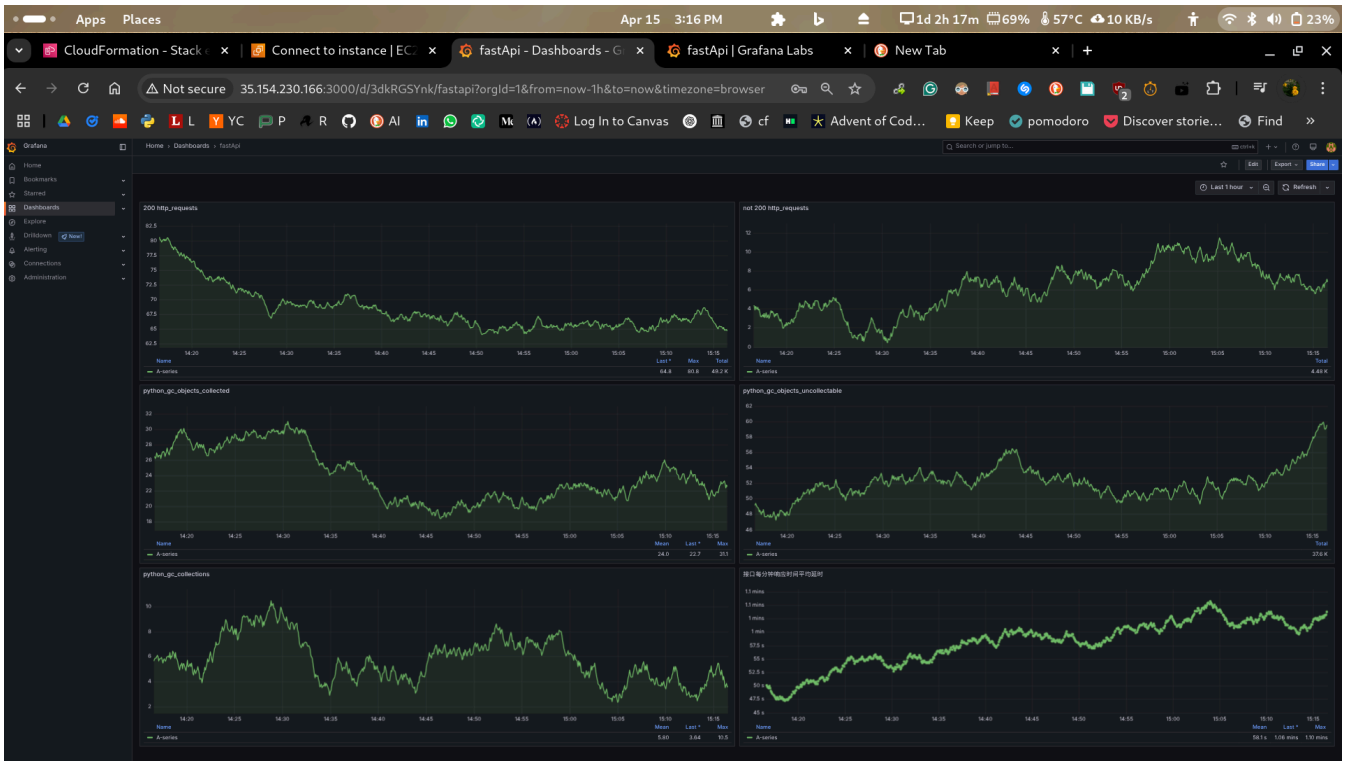
   In Grafana, add a data source by specifying the following URL:

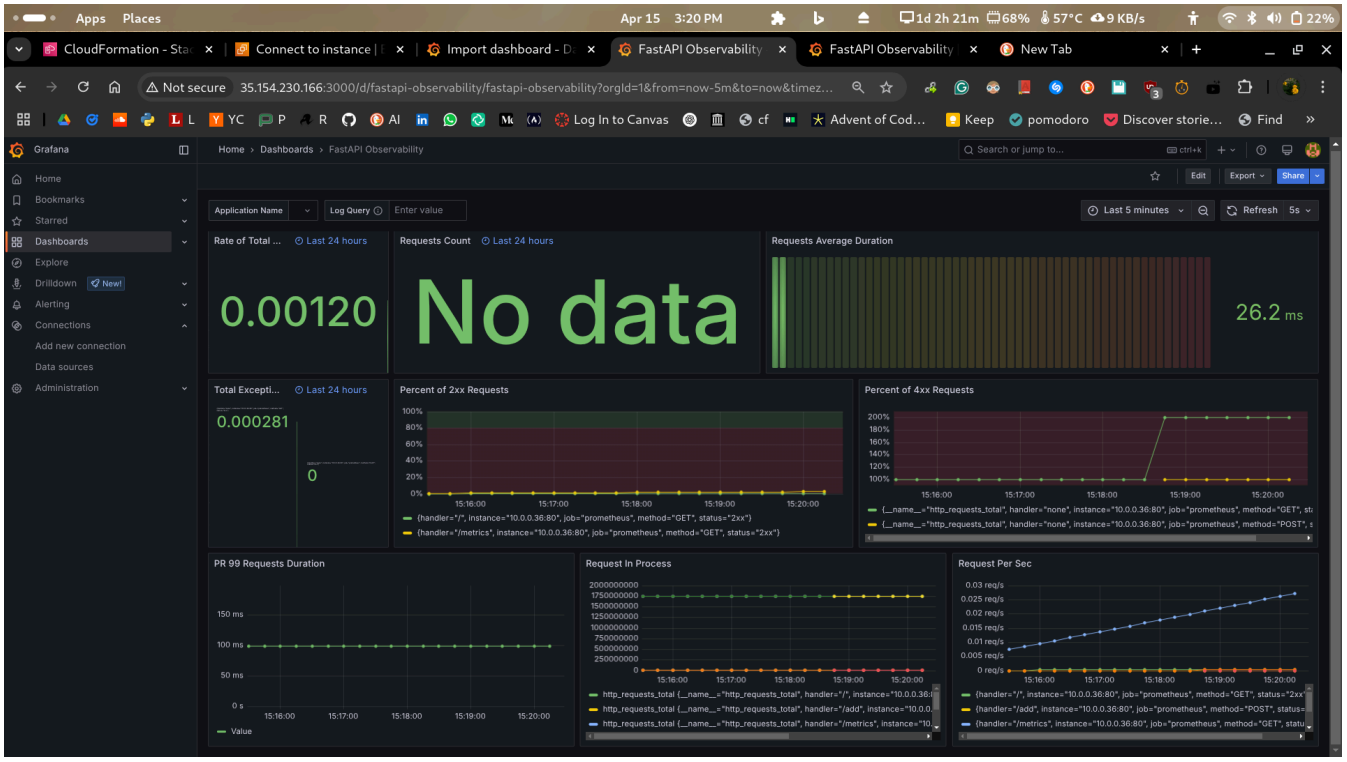
   ```
   http://10.0.0.36:9090
   ```

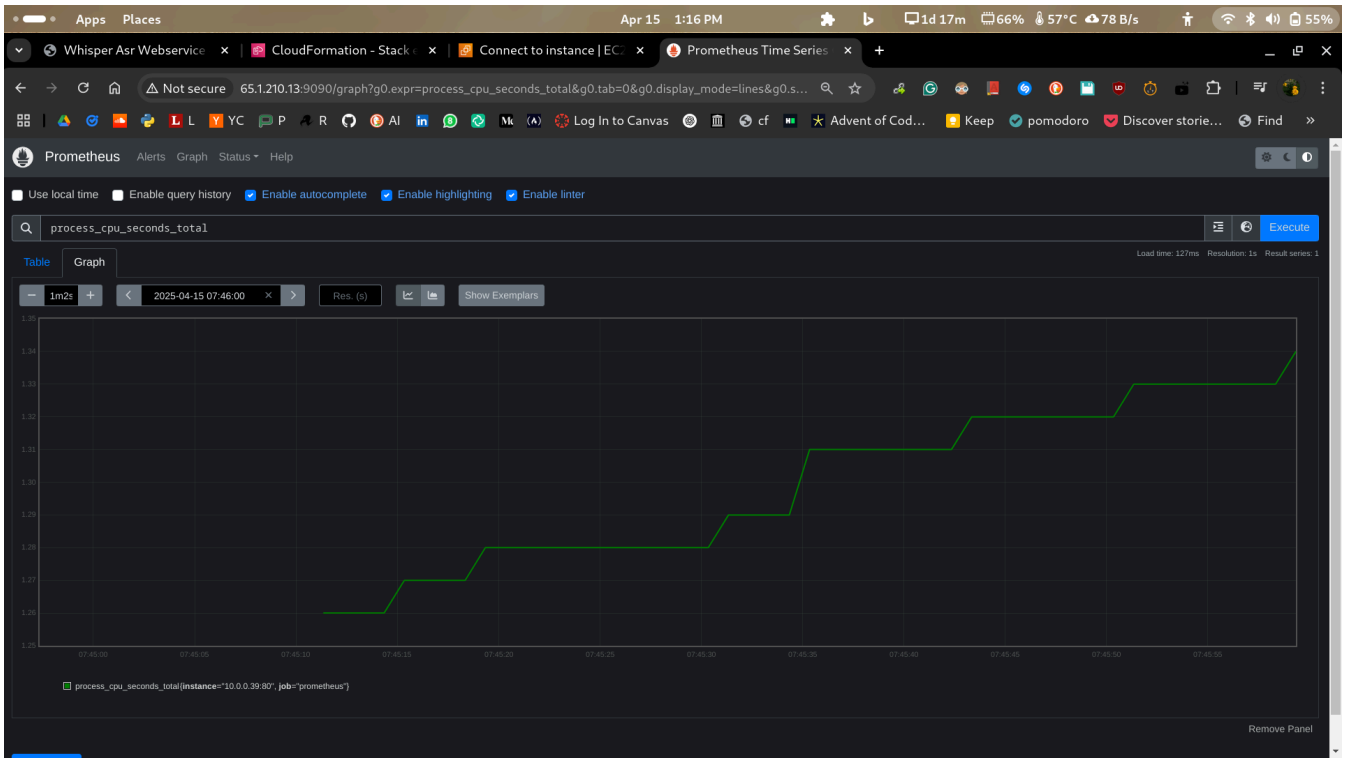
3. **Create a New Dashboard**
   - Log in to Grafana.
   - Navigate to **Dashboard > New Dashboard**.

- Add the panel with ID `15834`.
- You can also add another dashboard with ID `18739`.

Dashboard Screenshots Showing Application Metrics






Apps Places
Apr 15 1:16 PM
1d 17m 66% 57°C 78 B/s
55%
Whisper Asr Webservice
CloudFormation - Stack
Connect to instance | EC2
Prometheus Time Series
Not secure
65.1.210.13:9090/graph?g0.expr=process_cpu_seconds_total&g0.tab=0&g0.display_mode=lines&g0.s...
Log In to Canvas
Advent of Cod...
Keep
pomodoro
Discover storie...
Find
Prometheus
Alerts
Graph
Status
Help
Use local time
Enable query history
Enable autocomplete
Enable highlighting
Enable linter
process_cpu_seconds_total
Execute
Table
Graph
Load time: 127ms
Resolution: 1s
Result series: 1
1m2s
2025-04-15 07:46:00
Res. (s)
Show Exemplars
process_cpu_seconds_total{instance="10.0.0.39:80", job="prometheus"}
Remove Panel

Task5: Database and 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:

sql

```
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.

Task6

Script File (.sh or .py)

CloudFormation Deployment Script

```
#!/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
```

Explanation of the Deployment Script

This script builds and deploys the CloudFormation template. The template contains the user data for the EC2 instance.

User Data Script

```
#!/bin/bash


dnf update -y
dnf install httpd git python3 python3-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/do
cker-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
```

Explanation of the User Data Script

This script performs the following actions:

1. **System Update and Package Installation**:
   - Updates the system and installs `httpd`, `git`, `python3`, `python3-pip`, and `docker`.
2. **Docker Setup**:
   - Starts and enables the Docker service.
   - Adds the current user to the Docker group to allow running Docker commands without `sudo`.
3. **Docker Compose Installation**:
   - Downloads and installs Docker Compose.
4. **Clone the Web Application Repository**:
   - Clones the specified GitHub repository containing the web application.
5. **Run the Web Application**:
   - Navigates to the cloned repository and executes the `bash.sh` script.

Bash Script in the Web Application Folder

```
#!/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
```

Explanation of the Bash Script

This script performs the following actions:

1. **Create a Python Virtual Environment**:
   - Sets up a virtual environment in the current directory.
2. **Activate the Virtual Environment**:
   - Activates the virtual environment to isolate package installations.
3. **Install Required Packages**:
   - Installs the packages listed in `requirements.txt` and the `prometheus-fastapi-instrumentator` package.
4. **Run the Web Application**:
   - Starts the web application using `uvicorn`, listening on all interfaces at port 80 with auto-reload enabled.

Task7 : Disaster Recovery & High Availability

Key Elements of a DR Strategy

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).

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).

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.

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:

bash

```
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):

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

Create the backup plan:

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

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/AWSBackupDefaultServi
ceRole",
        "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:

Create a Backup Vault:

```
aws backup create-backup-vault \
```

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

1. *This command creates a backup vault in the specified region to store your backup data.*

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>"]
  }'
```

2. *This setup schedules a daily full backup for the specified EC2 instance, using AWS Backup with defined retention and scheduling, ensuring compliance with defined RTO and RPO.*

Task8

Task 8: AI Model Deployment & MLOps

We have used ECS to host the Docker task and added an ALB (Application Load Balancer) to the publicly exposed Docker service. This is all achieved using a CloudFormation template and ECS Fargate.

The Docker service takes a voice sound file and converts it into text.

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:
```

```yaml
    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-
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
    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"]]
```

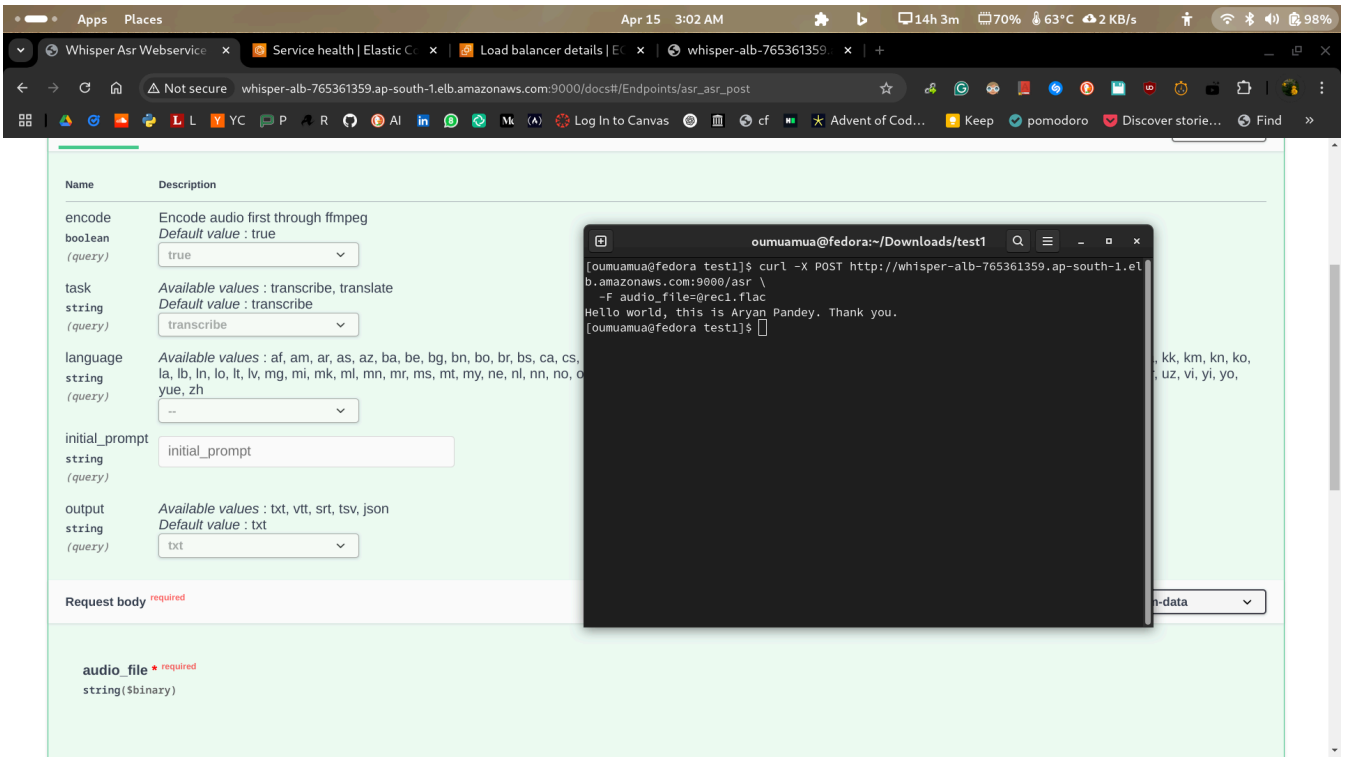
Steps to Deploy the Model

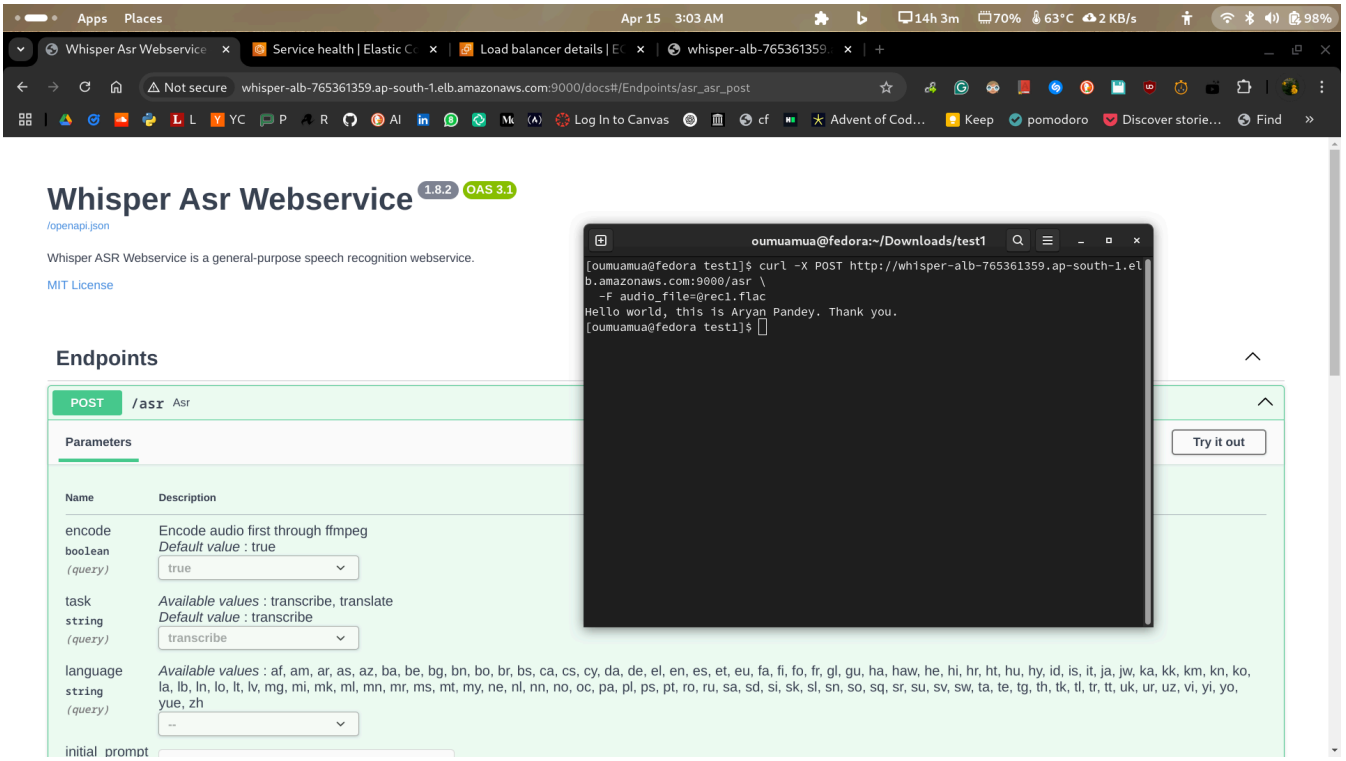
Deploy this using the AWS CLI deploy command:

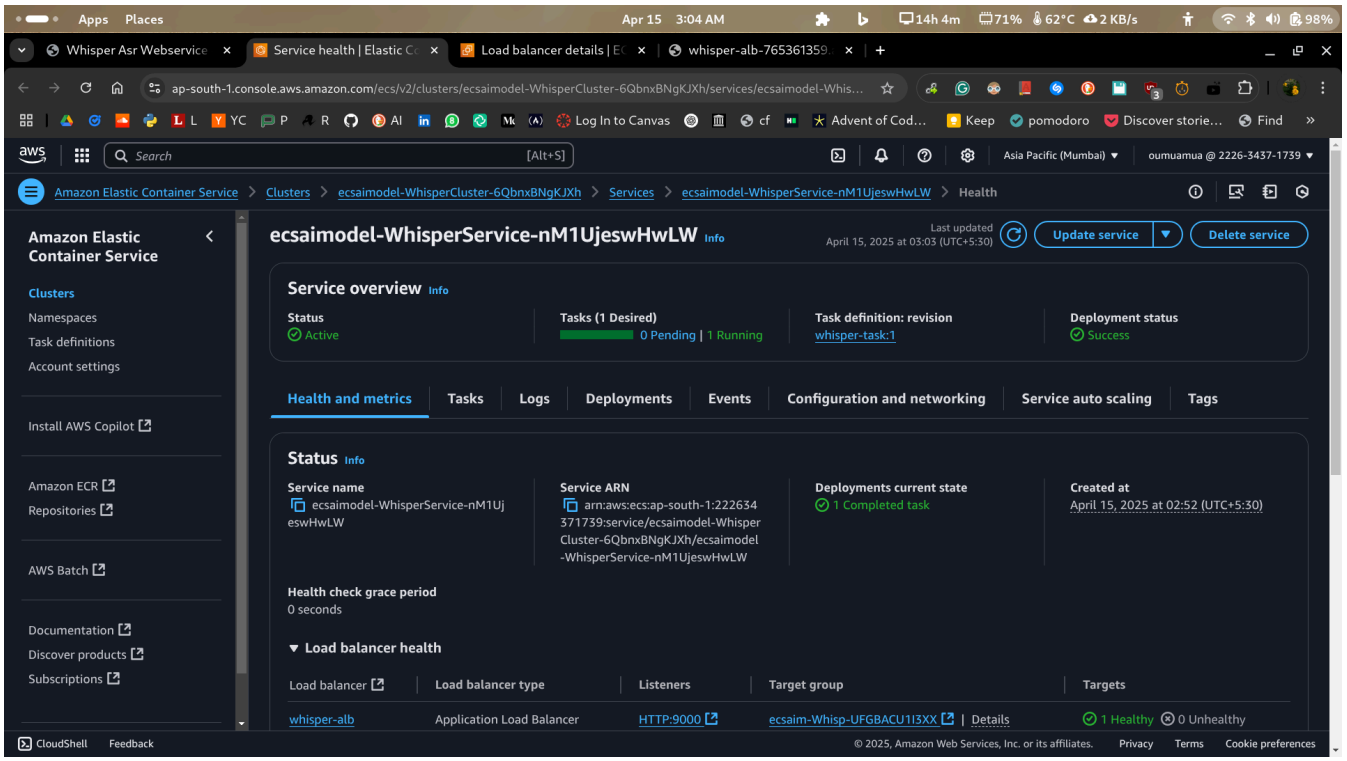
```
#!/bin/bash

aws cloudformation deploy --region ap-south-1 \
  --template-file ./main.yaml \
  --stack-name ecsaimodel \
  --tags madeFromCLI=yeah anotherTagForAllStackResources=okay
\
  --capabilities CAPABILITY_NAMED_IAM
```

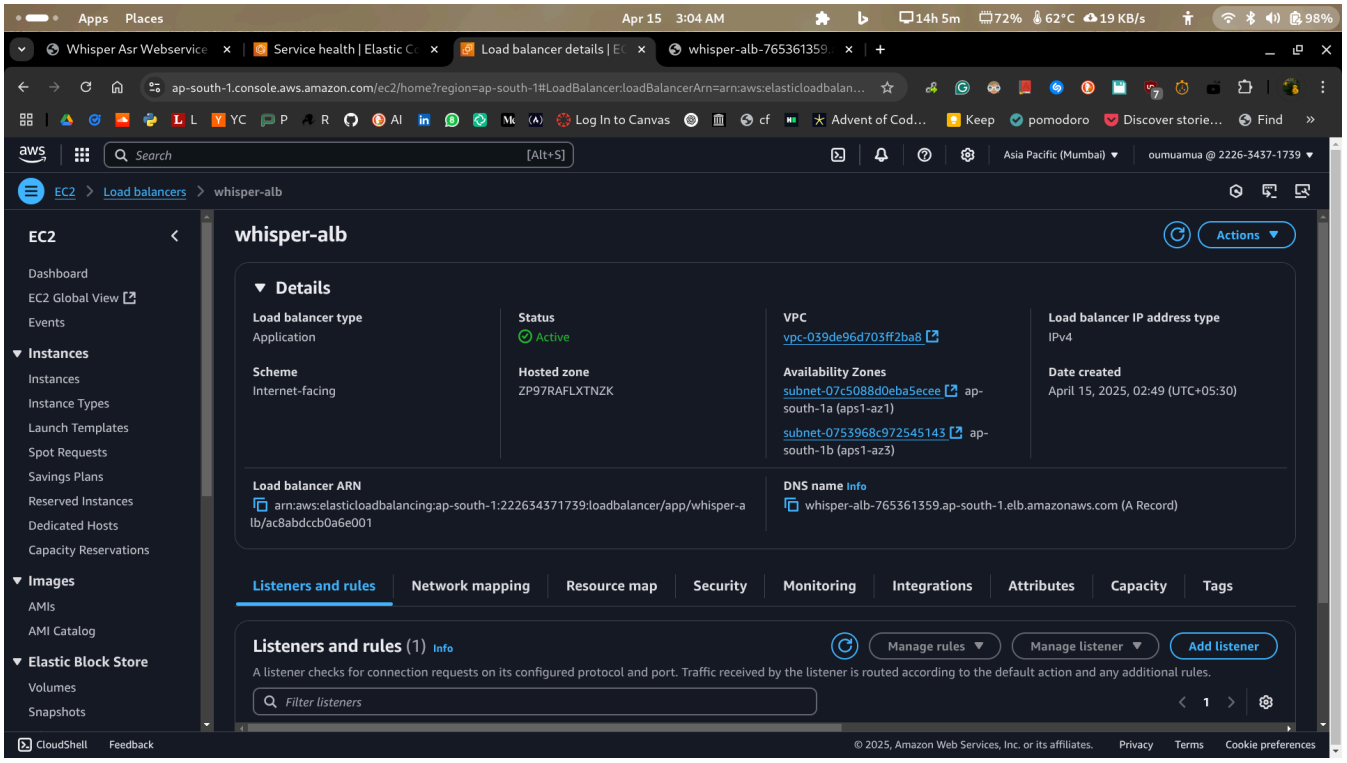
```
# --no-execute-changeset
```

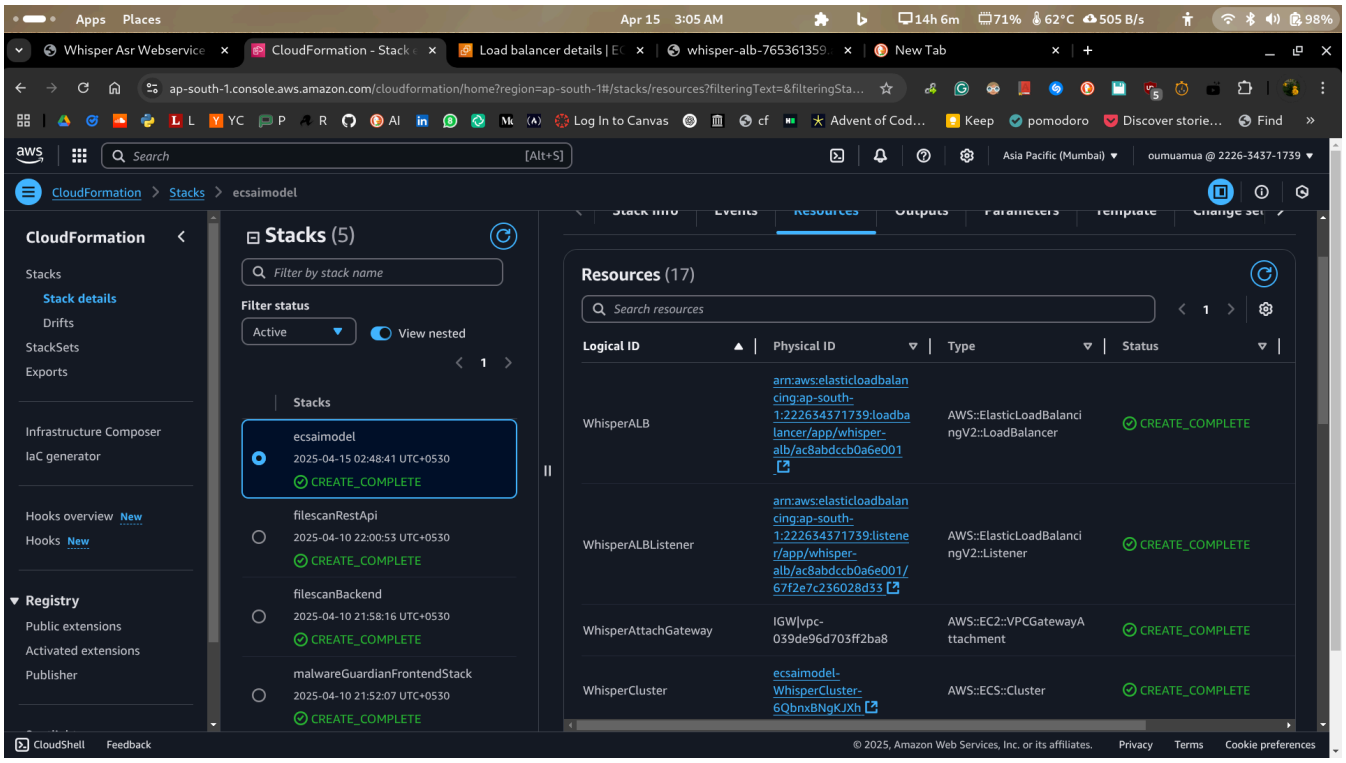
Screenshot of the model running on ECS

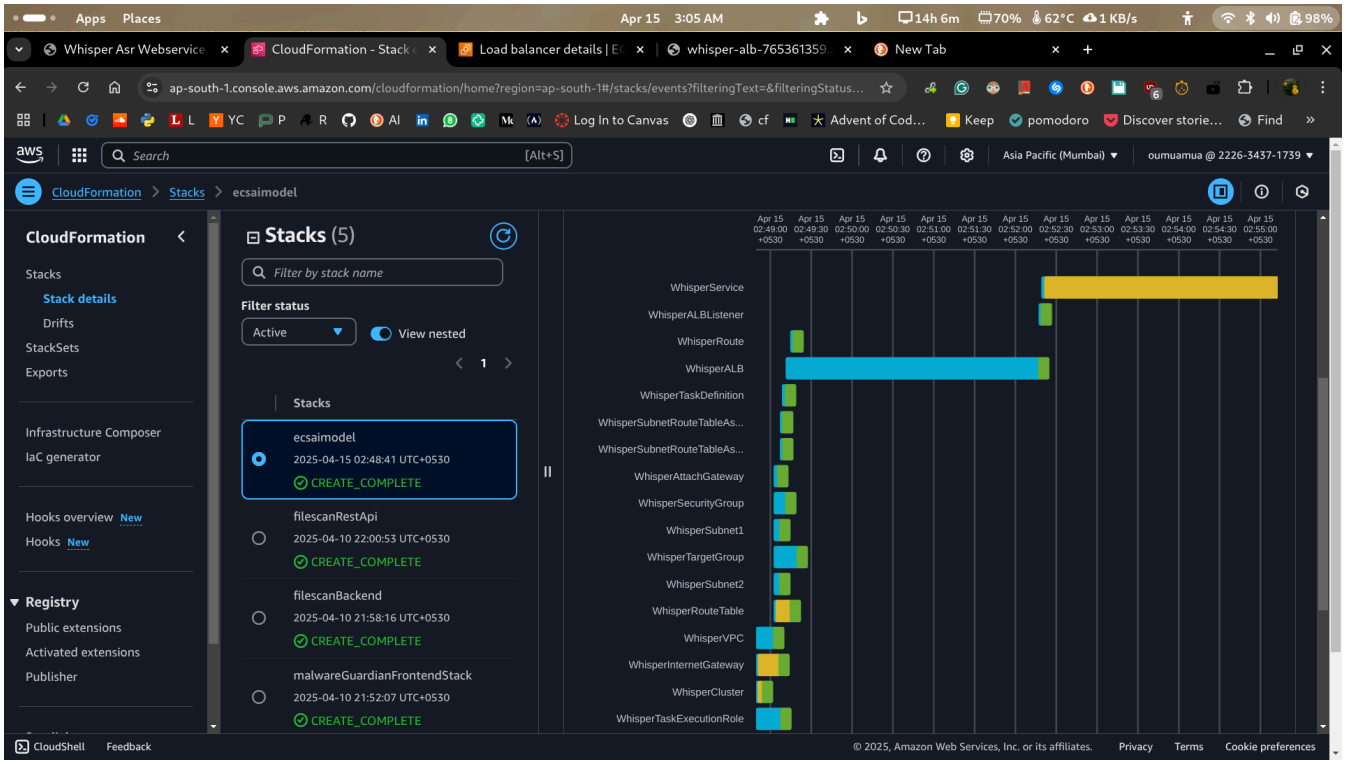





Apps Places Apr 15 3:04 AM 14h 4m 71% 62°C 2 KB/s 98%
Whisper Asr Webservice
Service health | Elastic C
Load balancer details | E
whisper-alb-765361359
ap-south-1.console.aws.amazon.com/ecs/v2/clusters/ecsaimodel-WhisperCluster-6QbnxBNgKJXh/services/ecsaimodel-Whis...
Log In to Canvas
Advent of Cod...
Keep
pomodoro
Discover storie...
Find
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
ecsaimodel-WhisperService-nM1UjeswHwLW Info
Last updated April 15, 2025 at 03:03 (UTC+5:30)
Update service
Delete service
Service overview Info
Status
Active
Tasks (1 Desired)
0 Pending | 1 Running
Task definition: revision
whisper-task:1
Deployment status
Success
Health and metrics
Tasks
Logs
Deployments
Events
Configuration and networking
Service auto scaling
Tags
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
CloudShell Feedback
© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences


Apps Places Apr 15 3:04 AM 14h 5m 72% 62°C 19 KB/s 98%
Whisper Asr Webservice
Service health | Elastic C
Load balancer details | E
whisper-alb-765361359
ap-south-1.console.aws.amazon.com/ec2/home?region=ap-south-1#LoadBalancer:loadBalancerArn=arn:aws:elasticloadbalan...
Log In to Canvas
Advent of Cod...
Keep
pomodoro
Discover storie...
Find
Search [Alt+S]
Asia Pacific (Mumbai)
oumuamua @ 2226-3437-1739
EC2 > Load balancers > whisper-alb
EC2
Dashboard
EC2 Global View
Events
Instances
Instances
Instance Types
Launch Templates
Spot Requests
Savings Plans
Reserved Instances
Dedicated Hosts
Capacity Reservations
Images
AMIs
AMI Catalog
Elastic Block Store
Volumes
Snapshots
whisper-alb
Actions
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
Network mapping
Resource map
Security
Monitoring
Integrations
Attributes
Capacity
Tags
Listeners and rules (1) Info
Manage rules
Manage listener
Add listener
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.
Filter listeners
CloudShell Feedback
© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences


Apps Places Apr 15 3:05 AM 14h 6m 71% 62°C 505 B/s 98%
Whisper Asr Webservice | CloudFormation - Stack | Load balancer details | whisper-alb-765361359 | New Tab
ap-south-1.console.aws.amazon.com/cloudformation/home?region=ap-south-1#/stacks/resources?filteringText=&filteringSta...
Search [Alt+S] Asia Pacific (Mumbai) oumuamua @ 2226-3437-1739
CloudFormation > Stacks > ecsaimodel
CloudFormation
Stacks
Stack details
Drifts
StackSets
Exports
Infrastructure Composer
IaC generator
Hooks overview New
Hooks New
Registry
Public extensions
Activated extensions
Publisher
Stacks (5)
Filter by stack name
Filter status
Active
View nested
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
CloudShell Feedback
© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences


Apps Places Apr 15 3:05 AM 14h 6m 70% 62°C 1 KB/s 98%
Whisper Asr Webservice | CloudFormation - Stack | Load balancer details | whisper-alb-765361359 | New Tab
ap-south-1.console.aws.amazon.com/cloudformation/home?region=ap-south-1#/stacks/events?filteringText=&filteringStatus...
Search [Alt+S] Asia Pacific (Mumbai) oumuamua @ 2226-3437-1739
CloudFormation > Stacks > ecsaimodel
CloudFormation
Stacks
Stack details
Drifts
StackSets
Exports
Infrastructure Composer
IaC generator
Hooks overview New
Hooks New
Registry
Public extensions
Activated extensions
Publisher
Stacks (5)
Filter by stack name
Filter status
Active
View nested
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
WhisperService
WhisperALBListener
WhisperRoute
WhisperALB
WhisperTaskDefinition
WhisperSubnetRouteTableAs...
WhisperSubnetRouteTableAs...
WhisperAttachGateway
WhisperSecurityGroup
WhisperSubnet1
WhisperTargetGroup
WhisperSubnet2
WhisperRouteTable
WhisperVPC
WhisperInternetGateway
WhisperCluster
WhisperTaskExecutionRole
CloudShell Feedback
© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences