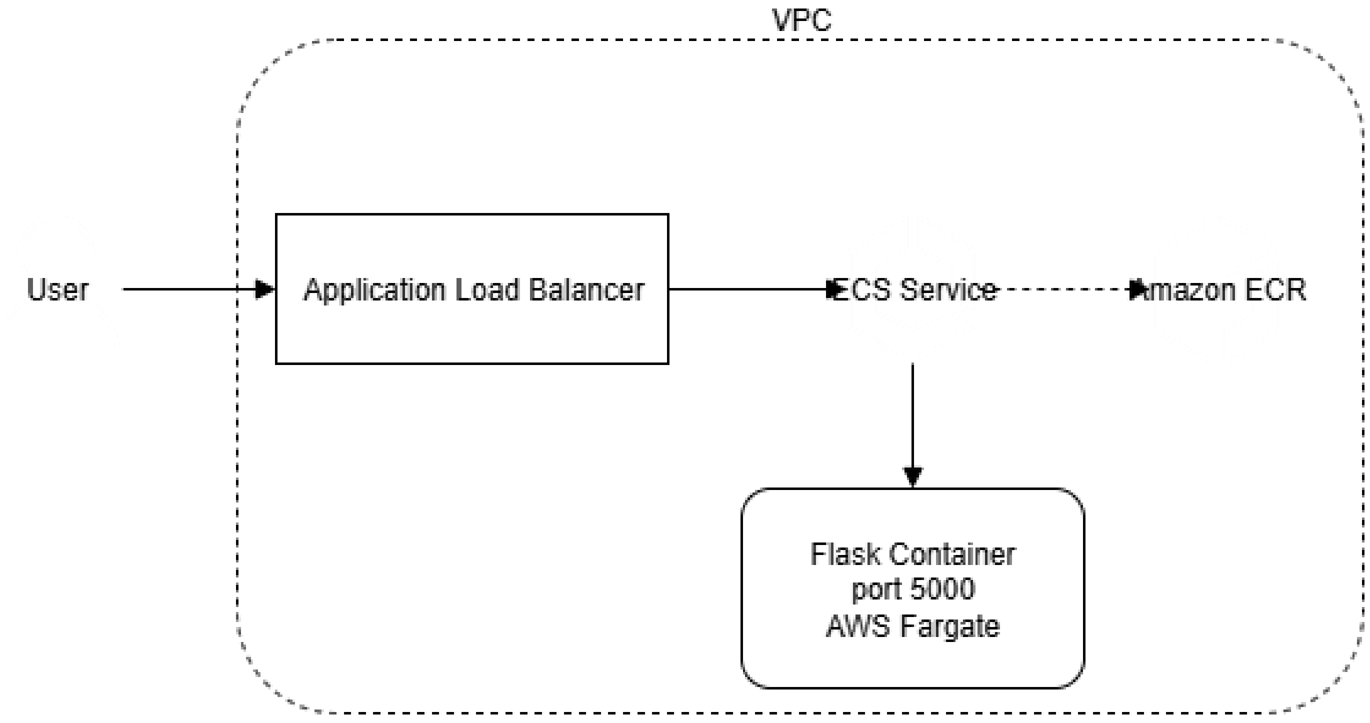
Name: Rakesh Revashetti

Date: 3rd July 2025

Objective: Deploy the Python API using AWS ECR and AWS ECS, Terraform-managed infrastructure, and Docker

This playbook documents the steps taken to provision AWS infrastructure using **Terraform**, build and push the Python API Docker image to **Amazon ECR**, and prepare the environment for deploying an containerized app using Amazon ECS

Architecture Diagram:



User → Application Load Balancer → AWS ECS service → Docker container running Flask API on port 5000

- User request goes to ALB
- ALB forwards to ECS Service
- ECS runs task on Fargate with Flask container
- Flask image is pulled from ECR

Components provisioned:

- VPC with public subnets
- Security groups for ALB and ECS
- Application Load Balancer (ALB)
- ECS Cluster (Fargate)
- ECS Task Definition + Service
- IAM Role for ECS Task Execution

Code repository:

Github URL contains python API code, terraform code and docker build script: https://github.com/rakeshrevashetti/tribal_assessment.git

Folder structure: https://github.com/rakeshrevashetti/tribal_assessment.git

tribal_assessment

```
├── README.md
├── deployment_script.sh
├── python-api
│   ├── Dockerfile
│   ├── README.md
│   ├── app.py
│   └── requirements.txt
└── terraform
    ├── ecs
    │   ├── alb
    │   │   ├── alb.tf
    │   │   ├── output.tf
    │   │   └── variable.tf
    │   ├── ecs_cluster
    │   │   ├── ecs_cluster.tf
    │   │   ├── output.tf
    │   │   └── variable.tf
    │   ├── iam.tf
    │   ├── output.tf
    │   ├── task_definition
    │   │   ├── iam.tf
    │   │   ├── output.tf
    │   │   ├── task_definition.tf
    │   │   └── variable.tf
    │   ├── vpc
    │   │   ├── output.tf
    │   │   ├── variable.tf
    │   │   └── vpc.tf
    │   └── vpc.tf
    ├── main.tf
    ├── output.tf
    ├── terraform.tfstate
    ├── terraform.tfstate.backup
    ├── terraform.tfvars
    └── variable.tf
```

Tools Used

- AWS ECS (Fargate)
- AWS ALB
- AWS IAM, VPC, Subnets
- Docker, Flask
- Terraform

Steps involved in the process:

1. **Containerized the Flask API, we have written a docker file**:

   ```
   FROM python:3.7.11-slim
   WORKDIR  /python-api
   COPY requirements.txt requirements.txt
   RUN pip install -r requirements.txt
   COPY . /python-api
   CMD [ "python3", "-m" , "flask", "run", "--host=0.0.0.0"]
   ```

2. **Bash deployment scrip**t to build docker image and push that docker image to AWS ECR, with repo name as the accountID with image tag as latest and we need to configure aws credentials to authenticate to aws cloud:

   ```
   #!/bin/bash
   AWS_REGION="us-east-1"
   REPO_NAME="flask-api"
   IMAGE_TAG="latest"
   ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
   docker build -t $REPO_NAME ./python-api
   aws ecr get-login-password --region $AWS_REGION | docker login --username AWS --password-
   stdin $ACCOUNT_ID.dkr.ecr.$AWS_REGION.amazonaws.com
   ```

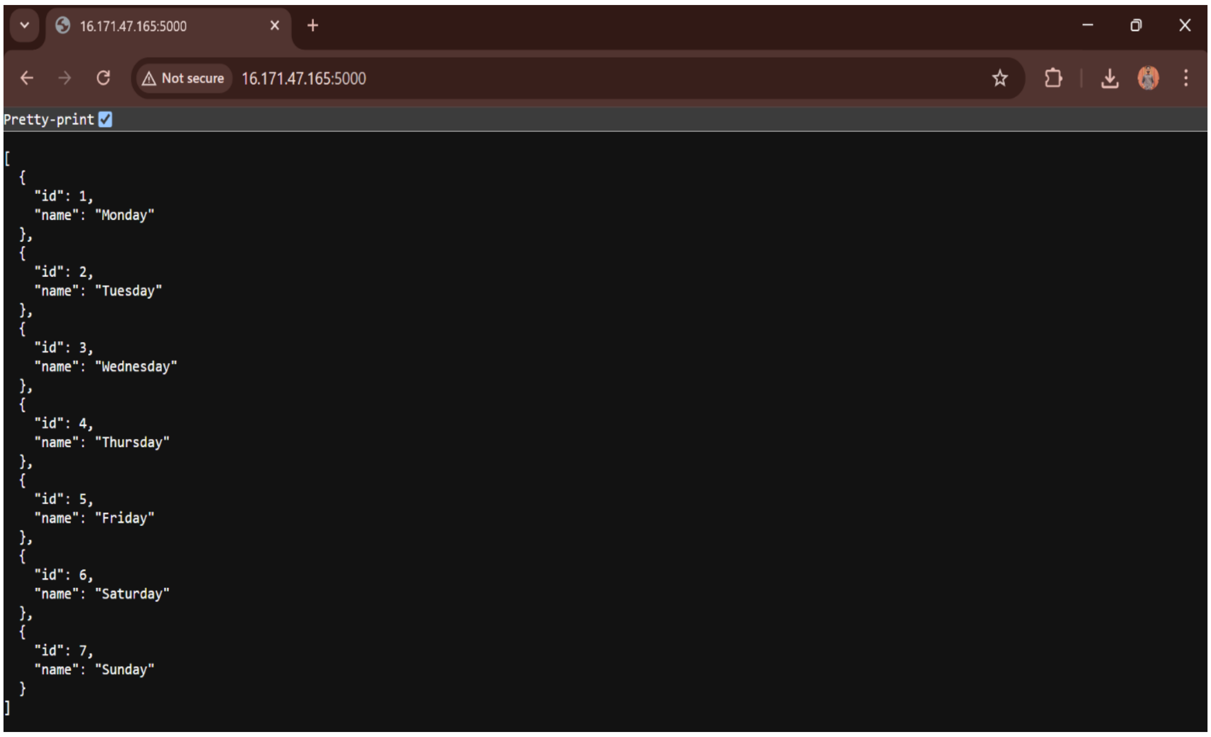
```
aws ecr describe-repositories --repository-names $REPO_NAME --region $AWS_REGION
>/dev/null || aws ecr create-repository --repository-name $REPO_NAME --region $AWS_REGION

docker tag $REPO_NAME:latest
$ACCOUNT_ID.dkr.ecr.$AWS_REGION.amazonaws.com/$REPO_NAME:$IMAGE_TAG
docker push
$ACCOUNT_ID.dkr.ecr.$AWS_REGION.amazonaws.com/$REPO_NAME:$IMAGE_TAG
```

3. **Provisioning the infrastructure resources using IaaC tool terraform**, we have used terraform modules to provision the resources module wise, given input variables through variable.tf file, and made use of output.tf file to output the alb dns link to access the flask api, given values to the input variables through terraform.tfvars file, and provisioned infrastructure with commands like:

   ```
   terraform init
   terraform validate
   terraform plan
   terraform apply
   ```

4. **Health checks**:
   The ALB periodically sends health checks to the container (typically at /) to ensure the app is running.
   If a container fails the health check, ECS automatically replaces it with a new task, ALB will stop routing traffic to it.

5. **Security Groups:**
   ALB and ECS service run in the same VPC (public subnet).
   The security group for the ALB allows HTTP traffic from the internet.
   The security group for ECS tasks allows traffic from the ALB on port 5000.



```
[
  {
    "id": 1,
    "name": "Monday"
  },
  {
    "id": 2,
    "name": "Tuesday"
  },
  {
    "id": 3,
    "name": "Wednesday"
  },
  {
    "id": 4,
    "name": "Thursday"
  },
  {
    "id": 5,
    "name": "Friday"
  },
  {
    "id": 6,
    "name": "Saturday"
  },
  {
    "id": 7,
    "name": "Sunday"
  }
]
```