Terraform AWS ECS Fargate Flask + Express Deployment

🧩 1. Project Overview

This project demonstrates the deployment of two containerized microservices —

Flask App (Python) — backend API

Express App (Node.js) — frontend service

The infrastructure is built and automated using Terraform on AWS ECS (Fargate).
Both applications are containerized using Docker and stored in AWS ECR.
Logs are managed in AWS CloudWatch, and networking is set up via Terraform-managed VPCs.

⚙️ 2. Tasks Completed

🟢 Task 1 — Setting Up Terraform Project Structure

Goal: Initialize Terraform to define and deploy AWS infrastructure.

Files Created:

main.tf — provider, ECS, ECR, and VPC configuration

variables.tf — AWS region, account ID, etc.

outputs.tf — display outputs such as ECR URLs

Key Commands:

```
terraform init
terraform plan -var="aws_account_id=215764923642"
terraform apply -var="aws_account_id=215764923642" -auto-approve
```

Explanation:
Terraform connects to AWS using your credentials and creates all resources (VPC, ECS Cluster, ECR repos, IAM Roles, Security Groups, etc.) automatically.

🟢 Task 2 — Creating ECR Repositories

Goal: Create repositories to store Docker images for both services.

Terraform Resources Used:

```
resource "aws_ecr_repository" "flask" {
  name = "flask-backend-repo"
}

resource "aws_ecr_repository" "express" {
  name = "express-frontend-repo"
}
```

Output Example:

```
express_ecr_repo_url = "215764923642.dkr.ecr.ap-south-1.amazonaws.com/express-frontend-repo"
flask_ecr_repo_url   = "215764923642.dkr.ecr.ap-south-1.amazonaws.com/flask-backend-repo"
```

Explanation:
ECR (Elastic Container Registry) is where Docker images are stored for ECS to pull during deployment.

🟢 Task 3 — Building and Pushing Docker Images

Goal: Build and push both Flask and Express Docker images to ECR.

Commands:

```
# Flask app
cd apps/flask_app
docker build -t flask-backend-repo .
docker tag flask-backend-repo:latest 215764923642.dkr.ecr.ap-south-1.amazonaws.com/flask-backend-repo:latest
docker push 215764923642.dkr.ecr.ap-south-1.amazonaws.com/flask-backend-repo:latest

# Express app
cd apps/express_app
```

docker build -t express-frontend-repo .
docker tag express-frontend-repo:latest 215764923642.dkr.ecr.ap-south-1.amazonaws.com/express-frontend-repo:latest
docker push 215764923642.dkr.ecr.ap-south-1.amazonaws.com/express-frontend-repo:latest

Explanation:
Each app's Dockerfile defines its environment and dependencies. After building, the image is tagged and pushed to AWS ECR for ECS deployment.

🟢 Task 4 — Deploying ECS Cluster and Task Definitions

Goal: Run Flask and Express containers on AWS ECS Fargate.

Terraform Resources:

```
resource "aws_ecs_cluster" "main" {
  name = "flask-express-cluster"
}

resource "aws_ecs_task_definition" "flask" {
  family                   = "flask-task"
  requires_compatibilities = ["FARGATE"]
  container_definitions    = jsonencode([{ ... }])
}

resource "aws_ecs_task_definition" "express" {
  family                   = "express-task"
  requires_compatibilities = ["FARGATE"]
  container_definitions    = jsonencode([{ ... }])
}
```

Explanation:
Each task definition maps to a container in ECS. It specifies:

Docker image from ECR

CPU, memory limits

Network mode

Log configuration

Exposed ports (Flask → 5000, Express → 3000)

🟢 Task 5 — Configuring Networking (VPC, Subnets, Security Groups)

Goal: Allow both ECS services to communicate securely and expose them publicly.

Terraform Resources:

```
resource "aws_vpc" "main" { ... }
resource "aws_subnet" "public_a" { ... }
resource "aws_subnet" "public_b" { ... }
resource "aws_security_group" "ecs_sg" {
 ingress {
  from_port   = 3000
  to_port     = 3000
  protocol    = "tcp"
  cidr_blocks = ["0.0.0.0/0"]
 }
 ingress {
  from_port   = 5000
  to_port     = 5000
  protocol    = "tcp"
  cidr_blocks = ["0.0.0.0/0"]
 }
}
```

Explanation:
This setup ensures both Flask and Express containers are accessible from the internet while maintaining isolation between services.

🟢 Task 6 — Enabling CloudWatch Logs

Goal: Capture logs for each ECS container to debug and monitor apps.

Key Command:

aws logs describe-log-groups --query
"logGroups[?contains(logGroupName,'express')].logGroupName" --output text

Sample Output:

/ecs/express-task

Explanation:
Logs for both services are stored in /ecs/<task-name> log groups inside CloudWatch for centralized monitoring.

🟢 Task 7 — Testing ECS Deployment

Goal: Confirm applications are running and reachable via public IP.

Commands Used:

aws ecs list-tasks --cluster flask-express-cluster
aws ecs describe-tasks --cluster flask-express-cluster --tasks <TASK_ARN>
aws ec2 describe-network-interfaces --network-interface-ids <ENI_ID> --query
"NetworkInterfaces[0].Association.PublicIp" --output text

Example Results:

Express App IP: 3.109.122.174
Flask App IP: 13.235.74.119

Test Commands:

curl http://3.109.122.174:3000
curl http://3.109.122.174:3000/health
curl http://3.109.122.174:3000/api/flask

Output Example:

{"source":"Express App","flask_response":{"message":"Hello from Flask running in Docker on port 5000!"}}

Explanation:
This confirms both containers are deployed successfully, communicating correctly, and publicly accessible.

🟢 Task 8 — GitHub Version Control Integration

Goal: Upload project to GitHub for version control and portfolio visibility.

Commands:

```
git init
git add .
git commit -m "Initial commit - Terraform AWS Flask Express deployment project"
git branch -M main
git remote add origin https://github.com/0079567603568sahal/terraform-aws-flask-express-deployment.git
git push -u origin main
```

Result:
✅ Project uploaded at:
👉 https://github.com/0079567603568sahal/terraform-aws-flask-express-deployment

🟢 Task 9 — Validation and Maintenance

Goal: Ensure infrastructure consistency and maintain resources.

Commands:

```
terraform plan -var="aws_account_id=215764923642"
terraform apply -var="aws_account_id=215764923642" -auto-approve
```

Result:

```
No changes. Your infrastructure matches the configuration.
Apply complete! Resources: 0 added, 0 changed, 0 destroyed.
```

Explanation:
Indicates that Terraform state and AWS resources are perfectly synchronized.

🧠 10. Learning Outcomes

Skill, Description

Terraform, Infrastructure as Code (IaC) with ECS, ECR, IAM, SG, VPC

Docker, Containerization and image management

AWS ECS & Fargate, Serverless container orchestration

AWS CLI, Service management and troubleshooting

CloudWatch, Centralized logging and monitoring

Git & GitHub, Version control and repository management

Networking, Configuring public access, ENIs, and subnets

🧱 11. Project Directory Structure

```
terraform-flask-express/
│
├── apps/
│   ├── flask_app/
│   │   ├── app.py
│   │   ├── Dockerfile
│   │   └── requirements.txt
│   ├── express_app/
│   │   ├── index.js
│   │   ├── package.json
│   │   └── Dockerfile
│
├── part3_ecr_ecs/
│   ├── main.tf
│   ├── variables.tf
│   ├── outputs.tf
│   └── terraform.tfstate
│
├── .gitignore
```

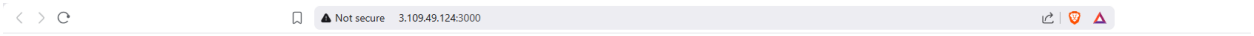
```
├── README.md
└── terraform.lock.hcl
```

🧩 12. Final Verification Checklist

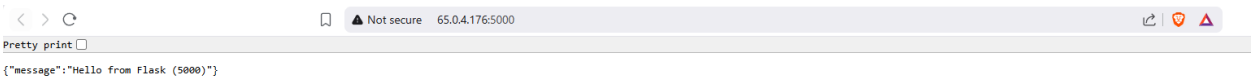
- ✅ Terraform applied successfully
- ✅ ECS Cluster active
- ✅ Both Flask & Express tasks in RUNNING state
- ✅ Public IP accessible
- ✅ Logs streaming to CloudWatch
- ✅ Project pushed to GitHub

13.screenshots

```
Mohds@Mohdsahal MINGW64 ~/terraform-flask-express/part1_single_ec2 (main)
$ terraform init -reconfigure | tee terraform_init_part1.txt
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v6.18.0...
- Installed hashicorp/aws v6.18.0 (signed by HashiCorp)
Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.

Mohds@Mohdsahal MINGW64 ~/terraform-flask-express/part1_single_ec2 (main)
$
```



Hello from Express on port 3000




```
Mohds@Mohdsahal MINGW64 ~/terraform-flask-express/part2_separate_ec2 (main)
$ terraform -v
Terraform v1.13.4
on windows_386
+ provider registry.terraform.io/hashicorp/aws v6.18.0
+ provider registry.terraform.io/hashicorp/random v3.7.2

Mohds@Mohdsahal MINGW64 ~/terraform-flask-express/part2_separate_ec2 (main)
$ aws sts get-caller-identity
{
    "UserId": "215764923642",
    "Account": "215764923642",
    "Arn": "arn:aws:iam::215764923642:root"
}


Mohds@Mohdsahal MINGW64 ~/terraform-flask-express/part2_separate_ec2 (main)
$ docker --version
Docker version 28.3.3, build 980b856

Mohds@Mohdsahal MINGW64 ~/terraform-flask-express/part2_separate_ec2 (main)
$ git --version
git version 2.50.0.windows.1

Mohds@Mohdsahal MINGW64 ~/terraform-flask-express/part2_separate_ec2 (main)
$
```

```
Mohds@Mohdsahal MINGW64 ~/terraform-flask-express/part2_separate_ec2 (main)
$ cd ~/terraform-flask-express

Mohds@Mohdsahal MINGW64 ~/terraform-flask-express (main)
$ ls
apps/  part1_single_ec2/  part2_separate_ec2/  part3_ecr_ecs/  README.md.txt  scripts/

Mohds@Mohdsahal MINGW64 ~/terraform-flask-express (main)
$ mkdir part1_single_ec2
cd part1_single_ec2
```

```
Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.

Mohds@Mohdsahal MINGW64 ~/terraform-flask-express/part1_single_ec2 (main)
$ terraform plan -var="ami=ami-08e5424edfe926b43" -var="key_name=mohdsahal924"
terraform apply -var="ami=ami-08e5424edfe926b43" -var="key_name=mohdsahal924" -auto-approve
var.tfstate_bucket
  S3 bucket for terraform state

  Enter a value:
```

```
Mohds@Mohdsahal MINGW64 ~/terraform-flask-express/part1_single_ec2 (main)
$ terraform plan -var="ami=ami-08e5424edfe926b43" -var="key_name=mohdsahal924"
data.aws_vpc.default: Reading...
data.aws_ami.ubuntu: Reading...
data.aws_ami.ubuntu: Read complete after 1s [id=ami-087d1c9a513324697]
data.aws_vpc.default: Read complete after 1s [id=vpc-022b0466a1cbd6e07]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # aws_instance.single_instance will be created
  + resource "aws_instance" "single_instance" {
      + ami                                  = "ami-08e5424edfe926b43"
      + arn                                  = (known after apply)
      + associate_public_ip_address          = (known after apply)
      + availability_zone                    = (known after apply)
      + disable_api_stop                     = (known after apply)
      + disable_api_termination              = (known after apply)
      + ebs_optimized                        = (known after apply)
      + enable_primary_ipv6                  = (known after apply)
      + force_destroy                        = false
      + get_password_data                    = false
      + host_id                              = (known after apply)
      + host_resource_group_arn              = (known after apply)
      + iam_instance_profile                 = (known after apply)
      + id                                   = (known after apply)
      + instance_initiated_shutdown_behavior = (known after apply)
      + instance_lifecycle                   = (known after apply)
      + instance_state                       = (known after apply)
      + instance_type                        = "t3.micro"
      + ipv6_address_count                   = (known after apply)
      + ipv6_addresses                       = (known after apply)
      + key_name                             = "mohdsahal924"
      + monitoring                           = (known after apply)
      + outpost_arn                          = (known after apply)
      + password_data                        = (known after apply)
      + placement_group                      = (known after apply)
      + placement_group_id                   = (known after apply)
      + placement_partition_number           = (known after apply)
      + primary_network_interface_id         = (known after apply)
      + private_dns                          = (known after apply)
      + private_ip                           = (known after apply)
      + public_dns                           = (known after apply)
      + public_ip                            = (known after apply)
      + region                               = "ap-south-1"
      + secondary_private_ips                = (known after apply)
      + security_groups                      = (known after apply)
      + source_dest_check                    = true
```

```
+ spot_instance_request_id             = (known after apply)
+ subnet_id                            = (known after apply)
+ tags                                 = {
    + "Name" = "flask-express-single"
  }
+ tags_all                             = {
    + "Name" = "flask-express-single"
  }
+ tenancy                              = (known after apply)
+ user_data                            = <<-EOT
      #!/bin/bash
      set -xe

      # update and install basics
      apt-get update -y
      apt-get upgrade -y
      apt-get install -y python3 python3-pip curl git

      # install Node.js 18.x
      curl -fsSL https://deb.nodesource.com/setup_18.x | bash -
      apt-get install -y nodejs build-essential

      # install pm2 to manage node
      npm install -g pm2

      # create app directories
      mkdir -p /opt/apps/flask
      mkdir -p /opt/apps/express

      # create sample flask app
      cat > /opt/apps/flask/app.py <<'PY'
      from flask import Flask
      app = Flask(__name__)

      @app.route('/')
      def hello():
          return 'Hello from Flask on port 5000'

      if __name__ == '__main__':
          app.run(host='0.0.0.0', port=5000)
      PY

      cat > /opt/apps/flask/requirements.txt <<'REQ'
      Flask==2.2.5
      REQ

      # create sample express app
      cat > /opt/apps/express/index.js <<'JS'
      const express = require('express')
      const app = express()
      const port = 3000

      app.get('/', (req, res) => res.send('Hello from Express on port 3000'))
```

```
            cd /opt/apps/express && npm install --production

            # start flask (nohup so it survives)
            nohup python3 /opt/apps/flask/app.py > /var/log/flask.log 2>&1 &

            # start express with pm2
            pm2 start /opt/apps/express/index.js --name express-app
            pm2 save
            pm2 startup systemd -u root --hp /root || true
          EOT
        + user_data_base64                    = (known after apply)
        + user_data_replace_on_change         = false
        + vpc_security_group_ids              = (known after apply)

        + capacity_reservation_specification (known after apply)

        + cpu_options (known after apply)

        + ebs_block_device (known after apply)

        + enclave_options (known after apply)

        + ephemeral_block_device (known after apply)

        + instance_market_options (known after apply)

        + maintenance_options (known after apply)

        + metadata_options (known after apply)

        + network_interface (known after apply)

        + primary_network_interface (known after apply)

        + private_dns_name_options (known after apply)

        + root_block_device (known after apply)
      }

    # aws_security_group.web_sg will be created
    + resource "aws_security_group" "web_sg" {
        + arn                    = (known after apply)
        + description            = "Allow SSH, Flask and Express ports"
        + egress                 = [
            + {
                + cidr_blocks      = [
                    + "0.0.0.0/0",
                  ]
                + from_port        = 0
                + ipv6_cidr_blocks = []
                + prefix_list_ids  = []
                + protocol         = "-1"
                + security_groups  = []
```

```
            + self             = false
            + to_port          = 0
              # (1 unchanged attribute hidden)
          },
      ]
    + id                     = (known after apply)
    + ingress                = [
        + {
            + cidr_blocks      = [
                + "0.0.0.0/0",
              ]
            + description      = "Express"
            + from_port        = 3000
            + ipv6_cidr_blocks = []
            + prefix_list_ids  = []
            + protocol         = "tcp"
            + security_groups  = []
            + self             = false
            + to_port          = 3000
          },
        + {
            + cidr_blocks      = [
                + "0.0.0.0/0",
              ]
            + description      = "Flask"
            + from_port        = 5000
            + ipv6_cidr_blocks = []
            + prefix_list_ids  = []
            + protocol         = "tcp"
            + security_groups  = []
            + self             = false
            + to_port          = 5000
          },
        + {
            + cidr_blocks      = [
                + "0.0.0.0/0",
              ]
            + description      = "SSH"
            + from_port        = 22
            + ipv6_cidr_blocks = []
            + prefix_list_ids  = []
            + protocol         = "tcp"
            + security_groups  = []
            + self             = false
            + to_port          = 22
          },
      ]
    + name                   = "flask-express-single-sg"
    + name_prefix            = (known after apply)
    + owner_id               = (known after apply)
    + region                 = "ap-south-1"
    + revoke_rules_on_delete = false
    + tags                   = {
```

```
Mohds@Mohdsahal MINGW64 ~/terraform-flask-express/part1_single_ec2 (main)
$ mkdir ~/terraform-flask-express/part2_separate_ec2
cd ~/terraform-flask-express/part2_separate_ec2
mkdir: cannot create directory ‘/c/Users/Mohds/terraform-flask-express/part2_separate_ec2’: File exists

Mohds@Mohdsahal MINGW64 ~/terraform-flask-express/part2_separate_ec2 (main)
$ terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v6.18.0
Terraform has made some changes to the provider dependency selections recorded
in the .terraform.lock.hcl file. Review those changes and commit them to your
version control system if they represent changes you intended to make.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.

Mohds@Mohdsahal MINGW64 ~/terraform-flask-express/part2_separate_ec2 (main)
$
```

```
Mohds@Mohdsahal MINGW64 ~/terraform-flask-express/part2_separate_ec2 (main)
$ terraform apply \
  -var="ami=ami-08e5424edfe926b43" \
  -var="key_name=mohdsahal924" \
  -auto-approve
data.aws_vpc.default: Reading...
data.aws_vpc.default: Read complete after 1s [id=vpc-022b0466a1cbd6e07]
aws_security_group.flask_sg: Refreshing state... [id=sg-0d4e818161a07dfdb]
aws_security_group.express_sg: Refreshing state... [id=sg-0d6adea5e019083d5]
aws_instance.flask_instance: Refreshing state... [id=i-0cc055b7c34894ed4]
aws_instance.express_instance: Refreshing state... [id=i-0a08abf51e8e9c368]

No changes. Your infrastructure matches the configuration.

Terraform has compared your real infrastructure against your configuration and found no differences, so no changes are needed.

Apply complete! Resources: 0 added, 0 changed, 0 destroyed.

Outputs:

express_public_ip = "3.109.49.124"
flask_public_ip = "65.0.4.176"
```

Pretty print

{"message":"Hello from Flask (5000)"}

Hello from Express on port 3000

```
Mohds@Mohdsahal MINGW64 ~/terraform-flask-express/part2_separate_ec2 (main)
$ terraform show
# data.aws_vpc.default:
data "aws_vpc" "default" {
    arn                                 = "arn:aws:ec2:ap-south-1:215764923642:vpc/vpc-022b0466a1cbd6e07"
    cidr_block                          = "172.31.0.0/16"
    cidr_block_associations             = [
        {
            association_id = "vpc-cidr-assoc-06ee438479ad22f10"
            cidr_block     = "172.31.0.0/16"
            state          = "associated"
        },
    ]
    default                             = true
    dhcp_options_id                     = "dopt-0c7f48040ca141fdf"
    enable_dns_hostnames                = true
    enable_dns_support                  = true
    enable_network_address_usage_metrics = false
    id                                  = "vpc-022b0466a1cbd6e07"
    instance_tenancy                    = "default"
    ipv6_association_id                 = null
    ipv6_cidr_block                     = null
    main_route_table_id                 = "rtb-0a030fa80a971c36b"
    owner_id                            = "215764923642"
    region                              = "ap-south-1"
    tags                                = {}
}

# aws_instance.express_instance:
resource "aws_instance" "express_instance" {
    ami                                  = "ami-08e5424edfe926b43"
    arn                                  = "arn:aws:ec2:ap-south-1:215764923642:instance/i-0a08abf51e8e9c368"
    associate_public_ip_address          = true
    availability_zone                    = "ap-south-1a"
    disable_api_stop                     = false
    disable_api_termination              = false
    ebs_optimized                        = false
    force_destroy                        = false
    get_password_data                    = false
    hibernation                          = false
    host_id                              = null
    iam_instance_profile                 = null
    id                                   = "i-0a08abf51e8e9c368"
    instance_initiated_shutdown_behavior = "stop"
    instance_lifecycle                   = null
    instance_state                       = "running"
    instance_type                        = "t3.micro"
    ipv6_address_count                   = 0
    ipv6_addresses                       = []
    key_name                             = "mohdsahal924"
```

```
    monitoring                           = false
    outpost_arn                          = null
    password_data                        = null
    placement_group                      = null
    placement_group_id                   = null
    placement_partition_number           = 0
    primary_network_interface_id         = "eni-0ca5eb28915092640"
    private_dns                          = "ip-172-31-40-207.ap-south-1.compute.internal"
    private_ip                           = "172.31.40.207"
    public_dns                           = "ec2-3-109-49-124.ap-south-1.compute.amazonaws.com"
    public_ip                            = "3.109.49.124"
    region                               = "ap-south-1"
    secondary_private_ips                = []
    security_groups                      = [
        "express-sg",
    ]
    source_dest_check                    = true
    spot_instance_request_id             = null
    subnet_id                            = "subnet-083bcd2c3732b79e0"
    tags                                 = {
        "Name" = "express-instance"
    }
    tags_all                             = {
        "Name" = "express-instance"
    }
    tenancy                              = "default"
    user_data                            = <<-EOT
        #!/bin/bash
        set -xe

        apt-get update -y
        apt-get upgrade -y
        apt-get install -y curl nodejs npm git build-essential

        # install Node 18.x (safer if using deb nodesource; but older ubuntu may include nodejs)
        curl -fsSL https://deb.nodesource.com/setup_18.x | bash -
        apt-get install -y nodejs

        npm install -g pm2

        mkdir -p /opt/apps/express
        cat > /opt/apps/express/index.js <<'JS'
        const express = require('express')
        const app = express()
        const port = 3000

        app.get('/', (req, res) => res.send('Hello from Express on port 3000'))
        app.get('/health', (req, res) => res.send('OK'))

        app.listen(port, '0.0.0.0', () => console.log(`Express listening on ${port}`))
        JS

        cat > /opt/apps/express/package.json <<'PJ'
```

```
        cat > /opt/apps/express/index.js <<'JS'
        const express = require('express')
        const app = express()
        const port = 3000

        app.get('/', (req, res) => res.send('Hello from Express on port 3000'))
        app.get('/health', (req, res) => res.send('OK'))

        app.listen(port, '0.0.0.0', () => console.log(`Express listening on ${port}`))
        JS

        cat > /opt/apps/express/package.json <<'PJ'
        {
          "name": "express-sample",
          "version": "1.0.0",
          "main": "index.js",
          "dependencies": {
            "express": "^4.18.2"
          }
        }
        PJ

        cd /opt/apps/express && npm install --production
        pm2 start /opt/apps/express/index.js --name express-app
        pm2 save
        pm2 startup systemd -u root --hp /root || true
    EOT
    user_data_replace_on_change          = false
    vpc_security_group_ids               = [
        "sg-0d6adea5e019083d5",
    ]

    capacity_reservation_specification {
        capacity_reservation_preference = "open"
    }

    cpu_options {
        amd_sev_snp      = null
        core_count       = 1
        threads_per_core = 2
    }

    credit_specification {
        cpu_credits = "unlimited"
    }

    enclave_options {
        enabled = false
    }

    maintenance_options {
        auto_recovery = "default"
    }
```

```
    metadata_options {
        http_endpoint               = "enabled"
        http_protocol_ipv6          = "disabled"
        http_put_response_hop_limit = 1
        http_tokens                 = "optional"
        instance_metadata_tags      = "disabled"
    }

    primary_network_interface {
        delete_on_termination = true
        network_interface_id  = "eni-0ca5eb28915092640"
    }

    private_dns_name_options {
        enable_resource_name_dns_a_record    = false
        enable_resource_name_dns_aaaa_record = false
        hostname_type                        = "ip-name"
    }

    root_block_device {
        delete_on_termination = true
        device_name           = "/dev/sda1"
        encrypted             = false
        iops                  = 100
        kms_key_id            = null
        tags                  = {}
        tags_all              = {}
        throughput            = 0
        volume_id             = "vol-012bdf9a5d30dc16c"
        volume_size           = 8
        volume_type           = "gp2"
    }
}

# aws_instance.flask_instance:
resource "aws_instance" "flask_instance" {
    ami                                  = "ami-08e5424edfe926b43"
    arn                                  = "arn:aws:ec2:ap-south-1:215764923642:instance/i-0cc055b7c34894ed4"
    associate_public_ip_address          = true
    availability_zone                    = "ap-south-1a"
    disable_api_stop                     = false
    disable_api_termination              = false
    ebs_optimized                        = false
    force_destroy                        = false
    get_password_data                    = false
    hibernation                          = false
    host_id                              = null
    iam_instance_profile                 = null
    id                                   = "i-0cc055b7c34894ed4"
    instance_initiated_shutdown_behavior = "stop"
    instance_lifecycle                   = null
    instance_state                       = "running"
    instance_type                        = "t3.micro"
    ipv6_address_count                   = 0
    ipv6_addresses                       = []
    key_name                             = "mohdsahal924"
    monitoring                           = false
    outpost_arn                          = null
    password_data                        = null
    placement_group                      = null
    placement_group_id                   = null
    placement_partition_number           = 0
    primary_network_interface_id         = "eni-08176168821b3a84a"
    private_dns                          = "ip-172-31-41-43.ap-south-1.compute.internal"
    private_ip                           = "172.31.41.43"
    public_dns                           = "ec2-65-0-4-176.ap-south-1.compute.amazonaws.com"
    public_ip                            = "65.0.4.176"
    region                               = "ap-south-1"
    secondary_private_ips                = []
    security_groups                      = [
        "flask-sg",
    ]
    source_dest_check                    = true
    spot_instance_request_id             = null
    subnet_id                            = "subnet-083bcd2c3732b79e0"
    tags                                 = {
        "Name" = "flask-instance"
    }
    tags_all                             = {
        "Name" = "flask-instance"
    }
    tenancy                              = "default"
    user_data                            = <<-EOT
        #!/bin/bash
        set -xe

        apt-get update -y
        apt-get upgrade -y
        apt-get install -y python3 python3-pip git

        # create flask app
        mkdir -p /opt/apps/flask
        cat > /opt/apps/flask/app.py <<'PY'
        from flask import Flask, jsonify
        app = Flask(__name__)

        @app.route('/')
        def hello():
            return jsonify(message='Hello from Flask (5000)')

        @app.route('/health')
        def health():
            return 'OK', 200

        if __name__ == '__main__':
```

```
Mohds@Mohdsahal MINGW64 ~/terraform-flask-express/part3_ecr_ecs/scripts (main)
$ aws sts get-caller-identity --query Account --output text
215764923642
```

```
Mohds@Mohdsahal MINGW64 ~/terraform-flask-express/part3_ecr_ecs/scripts (main)
$ export AWS_ACCOUNT_ID=123456789012
export AWS_REGION=ap-south-1
```

```
Mohds@Mohdsahal MINGW64 ~/terraform-flask-express/part3_ecr_ecs/scripts (main)
$ echo $AWS_ACCOUNT_ID
echo $AWS_REGION
123456789012
ap-south-1
```

```
Mohds@Mohdsahal MINGW64 ~/terraform-flask-express/part3_ecr_ecs/scripts (main)
$ ./build_push_flask.sh
./build_push_express.sh
Login Succeeded
[+] Building 0.0s (0/0)
ERROR: failed to build: unable to prepare context: path "../apps/flask_app" not found
Login Succeeded
[+] Building 0.0s (0/0)
ERROR: failed to build: unable to prepare context: path "../apps/express_app" not found
```

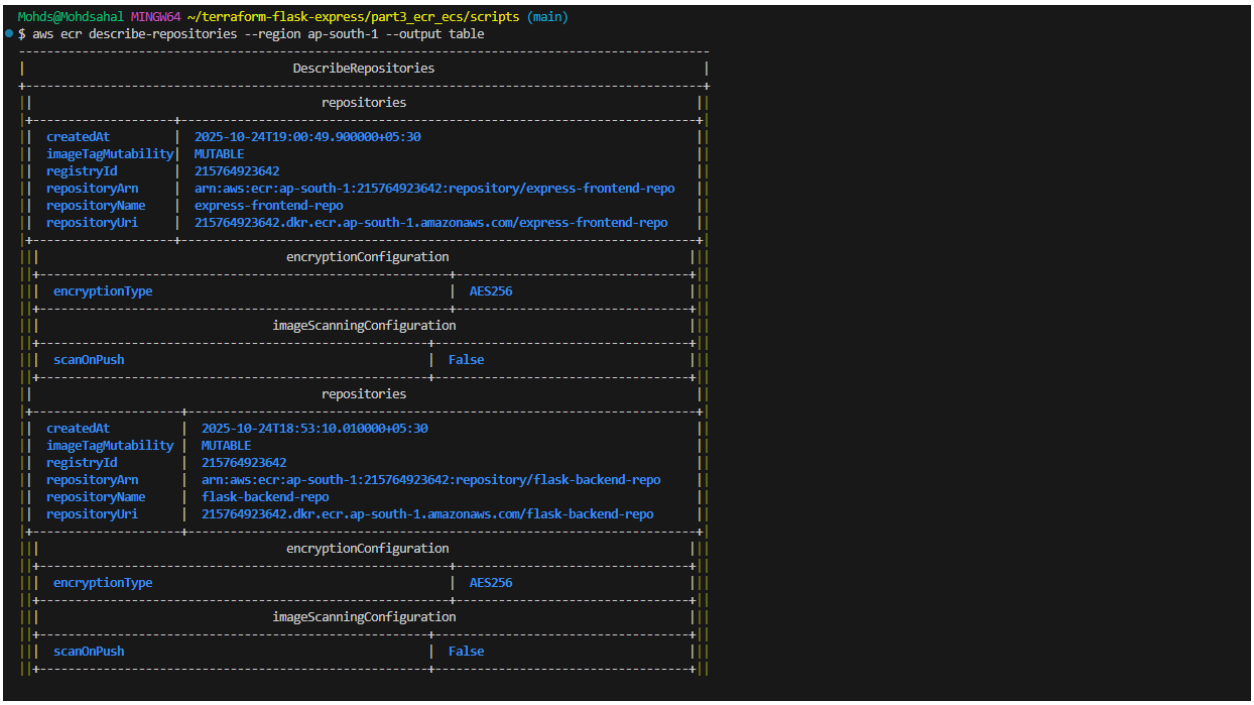
```
Mohds@Mohdsahal MINGW64 ~/terraform-flask-express/part3_ecr_ecs/scripts (main)
$ cd ~/terraform-flask-express/part3_ecr_ecs/scripts
export AWS_ACCOUNT_ID=215764923642
export AWS_REGION=ap-south-1
./build_push_flask.sh
./build_push_express.sh
Login Succeeded
[+] Building 20.3s (10/10) FINISHED                                          docker:desktop-linux
 => [internal] load build definition from Dockerfile                                         0.1s
 => => transferring dockerfile: 153B                                                         0.0s
 => [internal] load metadata for docker.io/library/python:3.9-slim                           4.4s
 => [auth] library/python:pull token for registry-1.docker.io                                0.0s
 => [internal] load .dockerignore                                                            0.1s
 => => transferring context: 2B                                                              0.0s
 => [1/4] FROM docker.io/library/python:3.9-slim@sha256:545badebace9a958b98d3e272f0f0d46c0a1a389ac77e24c33f2e7b548ce1b6b  9.3s
 => => resolve docker.io/library/python:3.9-slim@sha256:545badebace9a958b98d3e272f0f0d46c0a1a389ac77e24c33f2e7b548ce1b6b  0.0s
 => => sha256:6f010d693edcb2d965ff3c2a741fc92cded69746aa616f746e7813896fdd443b 13.87MB / 13.87MB  2.7s
 => => sha256:c57466e36deb99f355eab1131bc21d799709189ecdd7fb41cf4c17d5f53724e2 1.29MB / 1.29MB  2.3s
 => => sha256:21ed5e7e522581efdab14b4fde1d7d493901b4d25736b947520695c7911da66b 250B / 250B  1.7s
 => => sha256:38513bd7256313495cdd83b3b0915a633cfa475dc2a07072ab2c8d191020ca5d 29.78MB / 29.78MB  7.2s
 => => extracting sha256:38513bd7256313495cdd83b3b0915a633cfa475dc2a07072ab2c8d191020ca5d  0.9s
 => => extracting sha256:c57466e36deb99f355eab1131bc21d799709189ecdd7fb41cf4c17d5f53724e2  0.1s
 => => extracting sha256:6f010d693edcb2d965ff3c2a741fc92cded69746aa616f746e7813896fdd443b  0.6s
 => => extracting sha256:21ed5e7e522581efdab14b4fde1d7d493901b4d25736b947520695c7911da66b  0.0s
 => [internal] load build context                                                            0.1s
 => => transferring context: 548B                                                            0.0s
 => [2/4] WORKDIR /app                                                                       0.2s
 => [3/4] COPY . .                                                                           0.1s
 => [4/4] RUN pip install -r requirements.txt                                               50.6s
 => exporting to image                                                                       0.0s
 => => exporting layers                                                                      0.0s
 => => exporting manifest sha256:dfd0a3bab1db159878aafddb99771bf48c509f841e0c66cffe1b0f6020c26e7f  0.0s
 => => exporting config sha256:0c52490b8280b20414251032c155b5ee48b55b8a1f70c236b53a24ee28b17953  0.0s
 => => exporting attestation manifest sha256:d6c4eaaf802d1d1707506bf61df2574ccbb5905d44d4a125828d619575f846db  0.0s
 => => exporting manifest list sha256:33ed6376474b4d71cfc8b91cd4cf151d84556eee939a874235b048fc5df8948d  0.0s
 => => naming to docker.io/library/flask-backend-repo:latest                                 0.0s
 => => unpacking to docker.io/library/flask-backend-repo:latest                              0.4s
{
    "repository": {
        "repositoryArn": "arn:aws:ecr:ap-south-1:215764923642:repository/flask-backend-repo",
        "registryId": "215764923642",
        "repositoryName": "flask-backend-repo",
        "repositoryUri": "215764923642.dkr.ecr.ap-south-1.amazonaws.com/flask-backend-repo",
        "createdAt": "2025-10-24T18:53:10.010000+05:30",
        "imageTagMutability": "MUTABLE",
        "imageScanningConfiguration": {
            "scanOnPush": false
        },
        "encryptionConfiguration": {
-- More  --
```

```
Mohds@Mohdsahal MINGW64 ~/terraform-flask-express/part3_ecr_ecs/scripts (main)
$ cd ~/terraform-flask-express/part3_ecr_ecs/scripts
export AWS_ACCOUNT_ID=215764923642
export AWS_REGION=ap-south-1
./build_push_flask.sh
./build_push_express.sh
Login Succeeded
[+] Building 20.3s (10/10) FINISHED                                                docker:desktop-linux
 => [internal] load build definition from Dockerfile                                               0.1s
 => => transferring dockerfile: 153B                                                               0.0s
 => [internal] load metadata for docker.io/library/python:3.9-slim                                 4.4s
 => [auth] library/python:pull token for registry-1.docker.io                                      0.0s
 => [internal] load .dockerignore                                                                  0.1s
 => => transferring context: 2B                                                                    0.0s
 => [1/4] FROM docker.io/library/python:3.9-slim@sha256:545badebace9a958b98d3e272f0f0d46c0a1a389ac77e24c33f2e7b548ce1b6b  9.3s
 => => resolve docker.io/library/python:3.9-slim@sha256:545badebace9a958b98d3e272f0f0d46c0a1a389ac77e24c33f2e7b548ce1b6b  0.0s
 => => sha256:6f010d693edcb2d965ff3c2a741fc92cded69746aa616f746e7813896fdd443b 13.87MB / 13.87MB    2.7s
 => => sha256:c57466e36deb99f355eab1131bc21d799709189ecdd7fb41cf4c17d5f53724e2 1.29MB / 1.29MB      2.3s
 => => sha256:21ed5e7e522581efdab14b4fde1d7d493901b4d25736b947520695c7911da66b 250B / 250B          1.7s
 => => sha256:38513bd7256313495cdd83b3b0915a633cfa475dc2a07072ab2c8d191020ca5d 29.78MB / 29.78MB    7.2s
 => => extracting sha256:38513bd7256313495cdd83b3b0915a633cfa475dc2a07072ab2c8d191020ca5d          0.9s
 => => extracting sha256:c57466e36deb99f355eab1131bc21d799709189ecdd7fb41cf4c17d5f53724e2          0.1s
 => => extracting sha256:6f010d693edcb2d965ff3c2a741fc92cded69746aa616f746e7813896fdd443b          0.6s
 => => extracting sha256:21ed5e7e522581efdab14b4fde1d7d493901b4d25736b947520695c7911da66b          0.0s
 => [internal] load build context                                                                  0.1s
 => => transferring context: 548B                                                                  0.0s
 => [2/4] WORKDIR /app                                                                             0.2s
 => [3/4] COPY . .                                                                                 0.1s
 => [4/4] RUN pip install -r requirements.txt                                                     50.6s
 => exporting to image                                                                             0.0s
 => => exporting layers                                                                            0.0s
 => => exporting manifest sha256:dfd0a3bab1db159878aafddb99771bf48c509f841e0c66cffe1b0f6020c26e7f  0.0s
 => => exporting config sha256:0c52490b8280b20414251032c155b5ee48b55b8a1f70c236b53a24ee28b17953    0.0s
 => => exporting attestation manifest sha256:d6c4eaaf802d1d1707506bf61df2574ccbb5905d44d4a125828d619575f846db  0.0s
 => => exporting manifest list sha256:33ed6376474b4d71cfc8b91cd4cf151d84556eee939a874235b048fc5df8948d  0.0s
 => => naming to docker.io/library/flask-backend-repo:latest                                       0.0s
 => => unpacking to docker.io/library/flask-backend-repo:latest                                    0.4s
{
    "repository": {
        "repositoryArn": "arn:aws:ecr:ap-south-1:215764923642:repository/flask-backend-repo",
        "registryId": "215764923642",
        "repositoryName": "flask-backend-repo",
        "repositoryUri": "215764923642.dkr.ecr.ap-south-1.amazonaws.com/flask-backend-repo",
        "createdAt": "2025-10-24T18:53:10.010000+05:30",
        "imageTagMutability": "MUTABLE",
        "imageScanningConfiguration": {
            "scanOnPush": false
        },
        "encryptionConfiguration": {
^C
Login Succeeded
```

```
Mohds@Mohdsahal MINGW64 ~/terraform-flask-express/part3_ecr_ecs/scripts (main)
$ aws ecr describe-repositories --region ap-south-1 --output table
-----------------------------------------------------------------------------------------------
|                                    DescribeRepositories                                     |
+---------------------------------------------------------------------------------------------+
||                                        repositories                                       ||
|+--------------------+----------------------------------------------------------------------+|
||  createdAt         |  2025-10-24T18:53:10.010000+05:30                                    ||
||  imageTagMutability|  MUTABLE                                                             ||
||  registryId        |  215764923642                                                        ||
||  repositoryArn     |  arn:aws:ecr:ap-south-1:215764923642:repository/flask-backend-repo   ||
||  repositoryName    |  flask-backend-repo                                                  ||
||  repositoryUri     |  215764923642.dkr.ecr.ap-south-1.amazonaws.com/flask-backend-repo    ||
|+--------------------+----------------------------------------------------------------------+|
|||                                 encryptionConfiguration                                 |||
||+---------------------------------------------------------+-------------------------------+||
|||  encryptionType                                         |  AES256                       |||
||+---------------------------------------------------------+-------------------------------+||
|||                                imageScanningConfiguration                               |||
||+------------------------------------------------------+----------------------------------+||
|||  scanOnPush                                          |  False                           |||
||+------------------------------------------------------+----------------------------------+||

Mohds@Mohdsahal MINGW64 ~/terraform-flask-express/part3_ecr_ecs/scripts (main)
$
```

```
Mohds@Mohdsahal MINGW64 ~/terraform-flask-express/part3_ecr_ecs/scripts (main)
$ cd ~/terraform-flask-express/part3_ecr_ecs/scripts
export AWS_ACCOUNT_ID=215764923642
export AWS_REGION=ap-south-1
./build_push_express.sh
Login Succeeded
[+] Building 21.6s (11/11) FINISHED                                                   docker:desktop-linux
 => [internal] load build definition from Dockerfile                                                  0.0s
 => => transferring dockerfile: 175B                                                                  0.0s
 => [internal] load metadata for docker.io/library/node:18-alpine                                     5.1s
 => [auth] library/node:pull token for registry-1.docker.io                                           0.0s
 => [internal] load .dockerignore                                                                     0.1s
 => => transferring context: 2B                                                                       0.0s
 => [1/5] FROM docker.io/library/node:18-alpine@sha256:8d6421d663b4c28fd3ebc498332f249011d118945588d0a35cb9bc4b8ca09d9e  10.9s
 => => resolve docker.io/library/node:18-alpine@sha256:8d6421d663b4c28fd3ebc498332f249011d118945588d0a35cb9bc4b8ca09d9e  0.1s
 => => sha256:25ff2da83641908f65c3a74d80409d6b1b62ccfaab220b9ea70b80df5a2e0549 446B / 446B           0.7s
 => => sha256:1e5a4c89cee5c0826c540ab06d4b6b491c96eda01837f430bd47f0d26702d6e3 1.26MB / 1.26MB       1.4s
 => => sha256:dd71dde834b5c203d162902e6b8994cb2309ae049a0eabc4efea161b2b5a3d0e 40.01MB / 40.01MB     9.4s
 => => sha256:f18232174bc91741fdf3da96d85011092101a032a93a388b79e99e69c2d5c870 3.64MB / 3.64MB       2.6s
 => => extracting sha256:f18232174bc91741fdf3da96d85011092101a032a93a388b79e99e69c2d5c870            0.2s
 => => extracting sha256:dd71dde834b5c203d162902e6b8994cb2309ae049a0eabc4efea161b2b5a3d0e            1.1s
 => => extracting sha256:1e5a4c89cee5c0826c540ab06d4b6b491c96eda01837f430bd47f0d26702d6e3            0.1s
 => => extracting sha256:25ff2da83641908f65c3a74d80409d6b1b62ccfaab220b9ea70b80df5a2e0549            0.0s
 => [internal] load build context                                                                     0.1s
 => => transferring context: 691B                                                                     0.0s
 => [2/5] WORKDIR /app                                                                                0.2s
 => [3/5] COPY package.json ./                                                                        0.1s
 => [4/5] RUN npm install --production                                                                3.2s
 => [5/5] COPY . .                                                                                    0.1s
 => exporting to image                                                                                1.5s
 => => exporting layers                                                                               0.7s
 => => exporting manifest sha256:10d1e24e94a577dd69b2eb8615167289dbd8a81e1e9835fed1d6ab396398c44d     0.0s
 => => exporting config sha256:fa9af14d0df29ec7ee74c30bf23ba2560f90e443de2997e7d9de7cea454a3c71       0.0s
 => => exporting attestation manifest sha256:ed190f716494acee2c5e2e7ec07499c77516bc0e7d7adcb074c81be9d18e699e  0.1s
 => => exporting manifest list sha256:90b4ddaf55d7d08e16f5a4479f35b0dfbb86df7508f3dbf4d303b82d8bbe68ea  46.1s
 => => naming to docker.io/library/express-frontend-repo:latest                                       0.0s
 => => unpacking to docker.io/library/express-frontend-repo:latest                                    0.0s
{
    "repository": {
        "repositoryArn": "arn:aws:ecr:ap-south-1:215764923642:repository/express-frontend-repo",
        "registryId": "215764923642",
        "repositoryName": "express-frontend-repo",
        "repositoryUri": "215764923642.dkr.ecr.ap-south-1.amazonaws.com/express-frontend-repo",
        "createdAt": "2025-10-24T19:00:49.900000+05:30",
        "imageTagMutability": "MUTABLE",
        "imageScanningConfiguration": {
            "scanOnPush": false
        },
        "encryptionConfiguration": {
            "encryptionType": "AES256"
        }
    }
}
```

```
The push refers to repository [215764923642.dkr.ecr.ap-south-1.amazonaws.com/express-frontend-repo]
f18232174bc9: Pushed
dd71dde834b5: Pushed
f37ca24cefaa: Pushed
25ff2da83641: Pushed
a289c5d27f66: Pushed
a798ead800a6: Pushed
d60e09391626: Pushed
c172ce7349dc: Pushed
1e5a4c89cee5: Pushed
latest: digest: sha256:90b4ddaf55d7d08e16f5a4479f35b0dfbb86df7508f3dbf4d303b82d8bbe68ea size: 856
Pushed 215764923642.dkr.ecr.ap-south-1.amazonaws.com/express-frontend-repo:latest

Mohds@Mohdsahal MINGW64 ~/terraform-flask-express/part3_ecr_ecs/scripts (main)
```


Mohds@Mohdsahal MINGW64 ~/terraform-flask-express/part3_ecr_ecs/scripts (main)
$ aws ecr describe-repositories --region ap-south-1 --output table
DescribeRepositories
repositories
createdAt | 2025-10-24T19:00:49.900000+05:30
imageTagMutability | MUTABLE
registryId | 215764923642
repositoryArn | arn:aws:ecr:ap-south-1:215764923642:repository/express-frontend-repo
repositoryName | express-frontend-repo
repositoryUri | 215764923642.dkr.ecr.ap-south-1.amazonaws.com/express-frontend-repo
encryptionConfiguration
encryptionType | AES256
imageScanningConfiguration
scanOnPush | False
repositories
createdAt | 2025-10-24T18:53:10.010000+05:30
imageTagMutability | MUTABLE
registryId | 215764923642
repositoryArn | arn:aws:ecr:ap-south-1:215764923642:repository/flask-backend-repo
repositoryName | flask-backend-repo
repositoryUri | 215764923642.dkr.ecr.ap-south-1.amazonaws.com/flask-backend-repo
encryptionConfiguration
encryptionType | AES256
imageScanningConfiguration
scanOnPush | False