```
#!/bin/bash
set -e

echo "============================="
echo "Terraform Part 1 Deployment"
echo "============================="

# -------- VARIABLES --------
AWS_REGION="ap-south-1"
INSTANCE_TYPE="m7i-flex.large"
KEY_NAME="AWSASSIGN"
AMI_ID="ami-0ff91eb5c6fe7cc86"
ALLOWED_IP="100.48.47.243/32"
PROJECT_DIR="$HOME/terraform/part1-single-ec2"

# -------- INSTALL DEPENDENCIES --------
echo "[1/6] Installing dependencies..."
sudo apt update -y
sudo apt install -y unzip curl gnupg software-properties-common

if ! command -v terraform &> /dev/null; then
  echo "Installing Terraform..."
  curl -fsSL https://apt.releases.hashicorp.com/gpg | sudo tee /usr/share/keyrings/hashicorp.gpg > /dev/null
  echo "deb [signed-by=/usr/share/keyrings/hashicorp.gpg] https://apt.releases.hashicorp.com $(lsb_release -cs) main" | sudo tee /etc/apt/sources.list.d/hashicorp.list
  sudo apt update && sudo apt install -y terraform
fi

terraform -version

# -------- PROJECT STRUCTURE --------
echo "[2/6] Creating Terraform project..."
mkdir -p "$PROJECT_DIR"
cd "$PROJECT_DIR"
```

```
terraform -version

# -------- PROJECT STRUCTURE --------
echo "[2/6] Creating Terraform project..."
mkdir -p "$PROJECT_DIR"
cd "$PROJECT_DIR"

# -------- provider.tf --------
cat <<EOF > provider.tf
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 5.0"
    }
  }
}

provider "aws" {
  region = var.aws_region
}
EOF

# -------- variables.tf --------
cat <<EOF > variables.tf
variable "aws_region" {}
variable "instance_type" {}
variable "key_name" {}
variable "ami_id" {}
variable "allowed_ip" {}
EOF

# -------- main.tf --------
cat <<EOF > main.tf
resource "aws_security_group" "flask_express_sg" {
  name = "flask-express-sg"
```

```
    from_port   = 22
    to_port     = 22
    protocol    = "tcp"
    cidr_blocks = [var.allowed_ip]
  }

  ingress {
    from_port   = 5000
    to_port     = 5000
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  ingress {
    from_port   = 3000
    to_port     = 3000
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  egress {
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }
}

resource "aws_instance" "single_ec2" {
  ami                    = var.ami_id
  instance_type          = var.instance_type
  key_name               = var.key_name
  vpc_security_group_ids = [aws_security_group.flask_express_sg.id]
  user_data              = file("user-data.sh")

  tags = {
    Name = "Flask-Express-Single-EC2"
```

Once we run bash script written , it create ec2 instance created and gave EC2 ip address , through that we can check url working fine or not.

```
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v5.100.0

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.
[4/6] Terraform plan
aws_security_group.flask_express_sg: Refreshing state... [id=sg-0261dd9116ad896bf]
aws_instance.single_ec2: Refreshing state... [id=i-005a6a7096cea2740]

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.
[5/6] Terraform apply
aws_security_group.flask_express_sg: Refreshing state... [id=sg-0261dd9116ad896bf]
aws_instance.single_ec2: Refreshing state... [id=i-005a6a7096cea2740]

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:

ec2_public_ip = "13.234.10.82"
[6/6] Deployment completed!
ec2_public_ip = "13.234.10.82"
ubuntu@ip-172-31-64-117:~$
```

Hello from Flask Backend

PART 2

```
Last login: Sat Dec 13 17:11:02 2025 from 18.206.107.29
ubuntu@ip-172-31-64-117:~$ cat deploy-2.sh
#!/bin/bash
set -e

echo "========================================"
echo " Terraform PART 2: Separate EC2 Deployment"
echo "========================================"

# ----------- EDIT THESE VALUES -------------
AWS_REGION="us-east-1"
INSTANCE_TYPE="t3.micro"
AMI_ID="ami-00096836009b16a22"        # Ubuntu 22.04 / Amazon Linux (same region)
KEY_NAME="AWSASSIGN"
ALLOWED_IP="100.48.47.243/32"     # Your IP
PROJECT_DIR="$HOME/terraform/part2-separate-ec2"
# -----------------------------------------

# -------- INSTALL TERRAFORM ----------
echo "[1/6] Installing Terraform..."
sudo apt update -y
sudo apt install -y unzip curl gnupg software-properties-common

if ! command -v terraform &> /dev/null; then
  curl -fsSL https://apt.releases.hashicorp.com/gpg | sudo apt-key add -
  sudo apt-add-repository "deb https://apt.releases.hashicorp.com $(lsb_release -cs) main"
  sudo apt update -y && sudo apt install terraform -y
fi

terraform -version

# -------- CREATE PROJECT ----------
echo "[2/6] Creating project directory..."
mkdir -p $PROJECT_DIR
cd $PROJECT_DIR
```

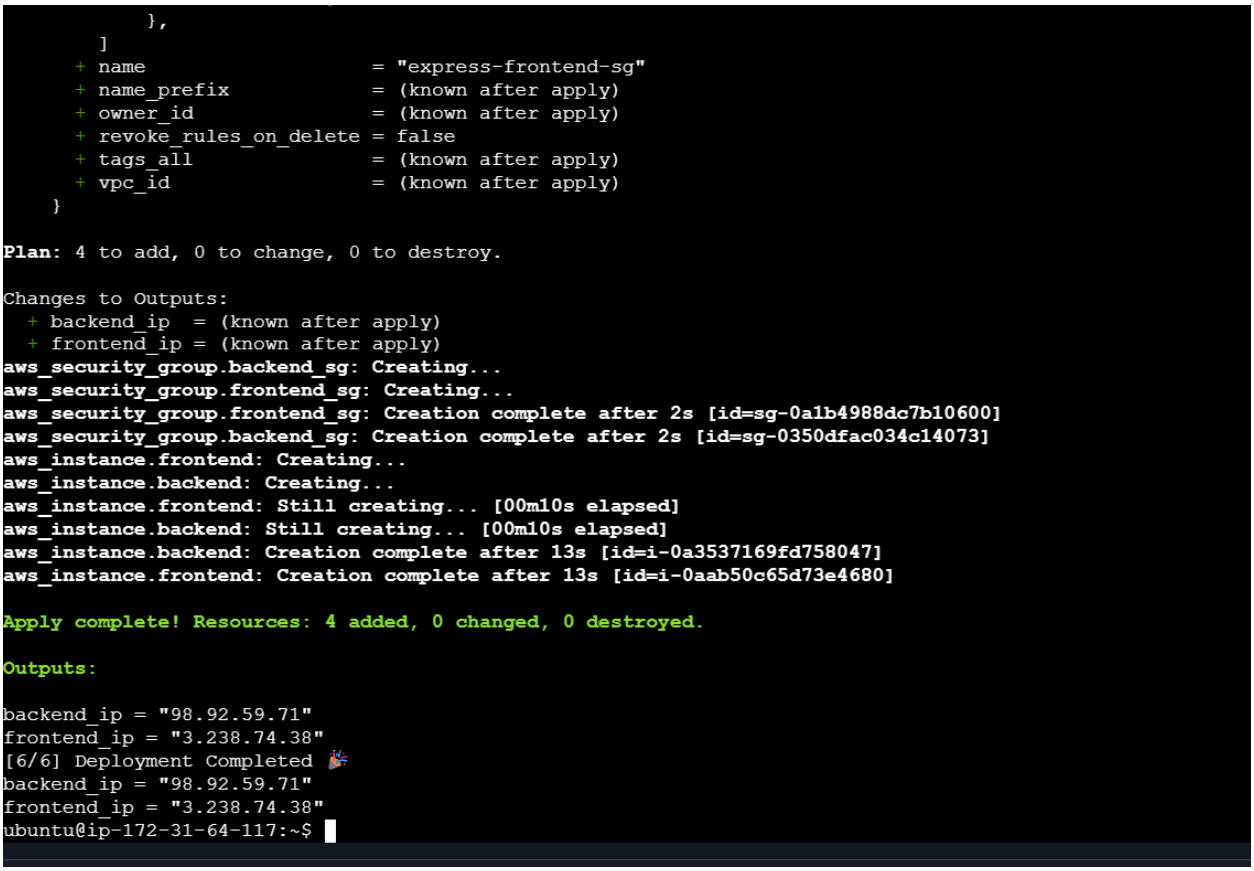
```
# -------- provider.tf ----------
cat <<EOF > provider.tf
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 5.0"
    }
  }
}

provider "aws" {
  region = var.aws_region
}
EOF

# -------- variables.tf ----------
cat <<EOF > variables.tf
variable "aws_region" {}
variable "instance_type" {}
variable "ami_id" {}
variable "key_name" {}
variable "allowed_ip" {}
EOF

# -------- security-groups.tf ----------
cat <<EOF > security-groups.tf
resource "aws_security_group" "backend_sg" {
  name = "flask-backend-sg"

  ingress {
    from_port   = 22
    to_port     = 22
    protocol    = "tcp"
    cidr_blocks = [var.allowed_ip]
  }
```

```
  ingress {
    from_port   = 5000
    to_port     = 5000
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  egress {
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }
}

resource "aws_security_group" "frontend_sg" {
  name = "express-frontend-sg"

  ingress {
    from_port   = 22
    to_port     = 22
    protocol    = "tcp"
    cidr_blocks = [var.allowed_ip]
  }

  ingress {
    from_port   = 3000
    to_port     = 3000
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  egress {
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
```

```
# -------- backend-user-data.sh ----------
cat <<EOF > backend-user-data.sh
#!/bin/bash
apt update -y
apt install -y python3 python3-pip

pip3 install flask

cat <<APP > /home/ubuntu/app.py
from flask import Flask
app = Flask(__name__)

@app.route("/api/health")
def health():
    return {"status":"ok"}

app.run(host="0.0.0.0", port=5000)
APP

nohup python3 /home/ubuntu/app.py &
EOF

chmod +x backend-user-data.sh

# -------- frontend-user-data.sh ----------
cat <<EOF > frontend-user-data.sh
#!/bin/bash
apt update -y
apt install -y nodejs npm

mkdir /home/ubuntu/app
cd /home/ubuntu/app

npm init -y
npm install express
```

```
  user_data              = file("frontend-user-data.sh")

  tags = {
    Name = "Express-Frontend-EC2"
  }
}
EOF

# -------- outputs.tf ----------
cat <<EOF > outputs.tf
output "backend_ip" {
  value = aws_instance.backend.public_ip
}

output "frontend_ip" {
  value = aws_instance.frontend.public_ip
}
EOF

# -------- terraform.tfvars ----------
cat <<EOF > terraform.tfvars
aws_region    = "$AWS_REGION"
instance_type = "$INSTANCE_TYPE"
ami_id        = "$AMI_ID"
key_name      = "$KEY_NAME"
allowed_ip    = "$ALLOWED_IP"
EOF

# -------- DEPLOY ----------
echo "[3/6] Terraform init"
terraform init

echo "[4/6] Terraform plan"
terraform plan

echo "[5/6] Terraform apply"
terraform apply -auto-approve
```

```
npm init -y
npm install express

cat <<JS > server.js
const express = require("express");
const app = express();

app.get("/", (req, res) => {
  res.send("Hello from Express Frontend");
});

app.listen(3000);
JS

nohup node server.js &
EOF

chmod +x frontend-user-data.sh

# -------- main.tf ----------
cat <<EOF > main.tf
resource "aws_instance" "backend" {
  ami                    = var.ami_id
  instance_type          = var.instance_type
  key_name               = var.key_name
  vpc_security_group_ids = [aws_security_group.backend_sg.id]
  user_data              = file("backend-user-data.sh")

  tags = {
    Name = "Flask-Backend-EC2"
  }
}

resource "aws_instance" "frontend" {
  ami                    = var.ami_id
  instance_type          = var.instance_type
  key_name               = var.key_name
  vpc_security_group_ids = [aws_security_group.frontend_sg.id]
```


},
]
+ name = "express-frontend-sg"
+ name_prefix = (known after apply)
+ owner_id = (known after apply)
+ revoke_rules_on_delete = false
+ tags_all = (known after apply)
+ vpc_id = (known after apply)
}
Plan: 4 to add, 0 to change, 0 to destroy.
Changes to Outputs:
+ backend_ip = (known after apply)
+ frontend_ip = (known after apply)
aws_security_group.backend_sg: Creating...
aws_security_group.frontend_sg: Creating...
aws_security_group.frontend_sg: Creation complete after 2s [id=sg-0a1b4988dc7b10600]
aws_security_group.backend_sg: Creation complete after 2s [id=sg-0350dfac034c14073]
aws_instance.frontend: Creating...
aws_instance.backend: Creating...
aws_instance.frontend: Still creating... [00m10s elapsed]
aws_instance.backend: Still creating... [00m10s elapsed]
aws_instance.backend: Creation complete after 13s [id=i-0a3537169fd758047]
aws_instance.frontend: Creation complete after 13s [id=i-0aab50c65d73e4680]
Apply complete! Resources: 4 added, 0 changed, 0 destroyed.
Outputs:
backend_ip = "98.92.59.71"
frontend_ip = "3.238.74.38"
[6/6] Deployment Completed
backend_ip = "98.92.59.71"
frontend_ip = "3.238.74.38"
ubuntu@ip-172-31-64-117:~$

PART 3

```
Dockerfile  server.js

/home/ubuntu/terraform/part3-ecs/flask-backend:
Dockerfile  app.py
ubuntu@ip-172-31-64-117:~/terraform/part3-ecs$ cd ~/terraform/part3-ecs
ubuntu@ip-172-31-64-117:~/terraform/part3-ecs$ cat <<EOF > provider.tf
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 5.0"
    }
  }
}

provider "aws" {
  region = "ap-south-1"
}
EOF
ubuntu@ip-172-31-64-117:~/terraform/part3-ecs$ cat <<EOF > ecr.tf
resource "aws_ecr_repository" "flask_repo" {
  name = "flask-backend"
}

resource "aws_ecr_repository" "express_repo" {
  name = "express-frontend"
}
EOF
ubuntu@ip-172-31-64-117:~/terraform/part3-ecs$ cat <<EOF > outputs.tf
output "flask_ecr_url" {
  value = aws_ecr_repository.flask_repo.repository_url
}

output "express_ecr_url" {
  value = aws_ecr_repository.express_repo.repository_url
```

```
  + resource "aws_ecr_repository" "flask_repo" {
      + arn                  = (known after apply)
      + id                   = (known after apply)
      + image_tag_mutability = "MUTABLE"
      + name                 = "flask-backend"
      + registry_id          = (known after apply)
      + repository_url       = (known after apply)
      + tags_all             = (known after apply)
    }

Plan: 2 to add, 0 to change, 0 to destroy.

Changes to Outputs:
  + express_ecr_url = (known after apply)
  + flask_ecr_url   = (known after apply)
aws_ecr_repository.flask_repo: Creating...
aws_ecr_repository.express_repo: Creating...
aws_ecr_repository.express_repo: Creation complete after 1s [id=express-frontend]
aws_ecr_repository.flask_repo: Creation complete after 1s [id=flask-backend]

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.

Outputs:

express_ecr_url = "401448503438.dkr.ecr.ap-south-1.amazonaws.com/express-frontend"
flask_ecr_url = "401448503438.dkr.ecr.ap-south-1.amazonaws.com/flask-backend"
ubuntu@ip-172-31-64-117:~/terraform/part3-ecs$
```

```
 ---> Removed intermediate container d569ca82fa17
 ---> 3a53d47c501d
Successfully built 3a53d47c501d
Successfully tagged flask-backend:latest
ubuntu@ip-172-31-64-117:~/terraform/part3-ecs/flask-backend$
```

```
ubuntu@ip-172-31-64-117:~/terraform/part3-ecs/flask-backend$ docker images
REPOSITORY                   TAG         IMAGE ID       CREATED          SIZE
flask-backend                latest      3a53d47c501d   51 seconds ago   134MB
python                       3.10-slim   53ef2abd7dc9   4 days ago       122MB
gcr.io/k8s-minikube/kicbase  v0.0.48     c6b5532e987b   3 months ago     1.31GB
ubuntu@ip-172-31-64-117:~/terraform/part3-ecs/flask-backend$
```

```
Sending build context to Docker daemon  3.072kB
Step 1/6 : FROM python:3.10-slim
 ---> 53ef2abd7dc9
Step 2/6 : WORKDIR /app
 ---> Using cache
 ---> 6d808b882e28
Step 3/6 : COPY app.py .
 ---> Using cache
 ---> a446d22e67c4
Step 4/6 : RUN pip install flask
 ---> Using cache
 ---> fa8bb05b2d88
Step 5/6 : EXPOSE 5000
 ---> Using cache
 ---> d3af0c2905bb
Step 6/6 : CMD ["python", "app.py"]
 ---> Using cache
 ---> 3a53d47c501d
Successfully built 3a53d47c501d
Successfully tagged flask-backend:1.0
ubuntu@ip-172-31-64-117:~/terraform/part3-ecs/flask-backend$ docker tag flask-backend:1.0 \
 \03438.dkr.ecr.ap-south-1.amazonaws.com/flask-backend:1.0
401448503438.dkr.ecr.ap-south-1.amazonaws.com/flask-backend:1.0
ubuntu@ip-172-31-64-117:~/terraform/part3-ecs/flask-backend$ docker push 401448503438.dkr.ecr.ap-so
.ecr.ap-south-1.amazonaws.com/flask-backend:1.0
The push refers to repository [401448503438.dkr.ecr.ap-south-1.amazonaws.com/flask-backend]
ea5a2d3a65c4: Pushed
f2b036a3287a: Pushed
1734833dac0c: Pushed
17bbb46fa426: Pushed
dc27ff3b7431: Pushed
557139ca12a0: Pushed
77a2b55fbe8b: Pushed                                                          1.0: digest: sh
: 1783
ubuntu@ip-172-31-64-117:~/terraform/part3-ecs/flask-backend$
```