**# August Terraform - AWS Infrastructure**

This Terraform project provisions a complete AWS infrastructure for hosting a containerized student portal application using ECS Fargate, with RDS PostgreSQL database backend.

**## Architecture Overview**

This infrastructure deploys a highly available, scalable web application with the following components:

- **\*\*Region\*\***: ap-south-1

- **\*\*Organization\*\***: livingdevops

- **\*\*Team\*\***: august bootcamp

**## Infrastructure Components**

**### Network (network.tf)**

- **\*\*VPC\*\***: Custom VPC with CIDR 10.0.0.0/16

- **\*\*Subnets\*\***:

  - 2 Public Subnets (10.0.1.0/24, 10.0.2.0/24) across AZ a & b - for ECS tasks

  - 2 Private Subnets (10.0.3.0/24, 10.0.4.0/24) across AZ a & b - for ALB

  - 2 RDS Subnets (10.0.5.0/24, 10.0.6.0/24) across AZ a & b - for database

- **\*\*Internet Gateway\*\***: For public subnet internet access

- **\*\*NAT Gateway\*\***: With Elastic IP for private subnet outbound traffic

- **\*\*Route Tables\*\***: Separate routing for public and private subnets

**### Application Layer (ecs.tf)**

- **\*\*ECS Cluster\*\***: Fargate-based cluster for running containers

- **\*\*ECS Task Definition\*\***:

  - Container: Student Portal application (ECR image)

  - Port: 8000

  - Resources: 256 CPU units, 512 MB memory

  - Environment: Database connection string injected via env vars

- **\*\*ECS Service\*\***:

  - Desired count: 2 tasks

  - Launch type: Fargate

  - Deployed in private subnets

  - Integrated with ALB

- **\*\*Security Group\*\***: Allows inbound on port 8000 from ALB only

**### Database Layer (rds.tf)**

- **\*\*RDS PostgreSQL\*\***:

  - Engine: PostgreSQL 14.15

  - Instance: db.t3.micro

  - Storage: 30 GB (auto-scaling up to 50 GB), encrypted with KMS

  - Backup retention: 7 days

  - Multi-AZ deployment via subnet group

  - Not publicly accessible

- **\*\*DB Subnet Group\*\***: Spans both RDS subnets

- **\*\*Security Group\*\***: Allows inbound on port 5432 from ECS tasks only

- **\*\*Secrets Manager\*\***: Stores database connection string securely

- **\*\*Random Password\*\***: Generated for RDS master user

**### Load Balancer (alb.tf)**

- **\*\*Application Load Balancer\*\***:

  - Deployed in public subnets

  - Deletion protection: disabled

- **\*\*Target Group\*\***: Routes traffic to ECS tasks on port 8000

- **\*\*Listeners\*\***:

  - HTTP (port 80): Forwards to target group

  - HTTPS (port 443): SSL termination with ACM certificate

- **\*\*Health Check\*\***: Endpoint `/login`, 90s interval

- **\*\*Security Group\*\***: Allows inbound HTTP/HTTPS from internet

**### DNS & SSL (route53.tf)**

- **\*\*Route53 Hosted Zone\*\***: hemantdesale.tech

- **\*\*DNS Record\*\***: august.hemantdesale.tech pointing to ALB

- **\*\*ACM Certificate\*\***: SSL certificate for august.hemantdesale.tech

- **\*\*DNS Validation\*\***: Automated via Route53

**### Monitoring (clowdwatch.tf)**

- **\*\*CloudWatch Log Group\*\***: `/aws/ecs/august-ecs` (30 day retention)

- **\*\*Log Query Definition\*\***: Pre-configured query to filter ECS logs

**### IAM (iam.tf)**

- **\*\*ECS Task Execution Role\*\***: Allows ECS to pull ECR images and write CloudWatch logs

- **\*\*Policy Attachment\*\***: AmazonECSTaskExecutionRolePolicy

**### Data Sources (data.tf)**

- Existing Elastic IP allocation

- Existing KMS key for RDS encryption

- Current AWS region and account identity

**## State Management**

- **\*\*Backend\*\***: S3 bucket `state-bucket-307946636515`

- **\*\*State file\*\***: `august-bootcamp25/terraform.tfstate`

- **\*\*Region\*\***: ap-south-1

- **\*\*Encryption\*\***: Enabled

**## Prerequisites**

1. AWS Account (ID: 059941826920)

2. Terraform version 1.5.7

3. AWS provider ~> 6.0.0

4. Existing resources:

   - Elastic IP allocation: `eipalloc-0e0fac707feec10ea`

   - KMS key: `alias/dev-august-batch-rds`

   - Route53 hosted zone: `hemantdesale.tech`

   - ECR repository with image: ` 059941826920.dkr.ecr.ap-south-1.amazonaws.com/ecs-studentportal-1.0

   - S3 bucket for state: `state-bucket-059941826920`

**## Usage**

**### Initialize Terraform**

```bash

terraform init

```

**### Plan Infrastructure**

```bash

terraform plan

```

**### Apply Infrastructure**

```bash

terraform apply

```

**### Destroy Infrastructure**

```bash

terraform destroy

```

**## Application Access**

Once deployed, the application is accessible at:

- **\*\*HTTP\*\***: http://dev.hemantdesale.tech

- **\*\*HTTPS\*\***: https://dev.hemantdesale.tech

**## Security Features**

- ✅ All resources tagged with repository, organization, and team

- ✅ Private subnets for application tier

- ✅ Database isolated in dedicated subnets

- ✅ Security groups with least privilege access

- ✅ RDS encryption at rest using KMS

- ✅ SSL/TLS encryption in transit via ACM

- ✅ Database credentials stored in Secrets Manager

- ✅ No public access to RDS instance

- ✅ NAT Gateway for secure outbound traffic from private subnets

**## Notes**

- The infrastructure uses implicit and explicit dependencies to ensure proper resource creation order

- Database connection string is automatically generated and injected into ECS containers

- Auto-scaling storage for RDS ensures database can grow as needed

- CloudWatch logging enabled for ECS tasks for monitoring and debugging

**## Pre-requisites to run this terraform code:**

- Create S3 bucket named " state-bucket-059941826920"

- Create public hosted zone named "hemantdesale.tech"

- Make sure the NS entries are correct in godaddy domain settings

- During RDS creation, do skip\_final\_snapshot   = true (For Non-Prod)

**## Terraform execution**

**### Dev**

`terraform init -backend-config=vars/dev.tfbackend`

`terraform plan -var-file=vars/dev.tfvars`

`terraform apply -var-file=vars/dev.tfvars`

**### destroy**

`terraform destroy -var-file=vars/dev.tfvars`

**### prod**

`terraform init -backend-config=vars/prod.tfbackend`

`terraform plan -var-file=vars/prod.tfvars`

`terraform apply -var-file=vars/prod.tfvars`

**## Switching the environment & statefile safely**

**### Step 1: Initialize dev backend**

terraform init -backend-config=vars/dev.tfbackend -reconfigure

**### Step 2: Apply dev environment**

terraform apply -var-file=vars/dev.tfvars

**### Step 3: Initialize prod backend**

**### Step 4: Apply prod environment**

terraform apply -var-file=vars/prod.tfvars

**### ECR Login command (For Testing purpose):**

Execute: `aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin 059941826920.dkr.ecr.ap-south-1.amazonaws.com` from cli

**### Load testing using docker container**

docker run --rm williamyeh/hey \

  -n 10000 \    # Total requests

  -c 200 \      # Concurrent workers

  https://your-ecs-app-url.com/

  docker run --rm williamyeh/hey \

  -n 1000 \

  -c 200 \

  https://dev.studentportal.akhileshmishra.tech/login

  <!-- docker run fjudith/load-test -h [host] -c [number of clients] -r [number of requests] -->

  docker run fjudith/load-test \

   -h https://dev.studentportal.akhileshmishra.tech/login \

   -c 10 \

   -r 1000







