ECS EC2 with ALB, EFS and CloudWatch (Terraform Deployment Guide)

Table of Contents

Prerequisites....1
Objective....2
Project Directory Structure....2
Architecture Diagram....3
Terraform Modular Structure....4
1. Configure the VPC Module....4
2. Configure the Security Module....5
3. Configure the ECR Module....6
4. Configure the EFS Module Persistent Storage....6
5. Configure the ECS Module....7
6. Configure the ALB Module (Public Entry Point)....9
Configure the Root main.tf....9
Terraform Deployment Commands....10
Initialize providers and modules....10
Format code (optional)....10
Validate configuration....10
View changes before applying....10
Deploy infrastructure....10
Custom Docker Image Build and Push....10
Testing and Validation....10
Validate Provisioned Infrastructure (AWS Console)....10
Validate Application Response....12
Cleanup....12

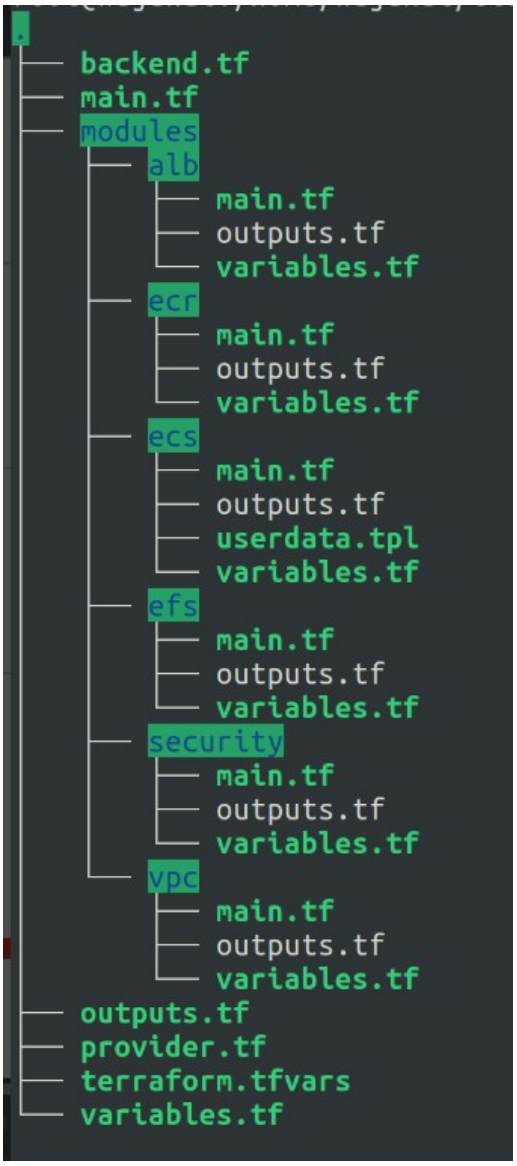
Prerequisites

- AWS account with programmatic access (IAM user with Administrator or equivalent permissions).
- Terraform installed (v1.3+ recommended).
- AWS CLI installed and configured.
- Existing S3 bucket for remote state (backend).
- Basic knowledge of ECS, EC2 networking, and Terraform modules.
- Custom Nginx Docker image pushed to ECR (or ready to push).

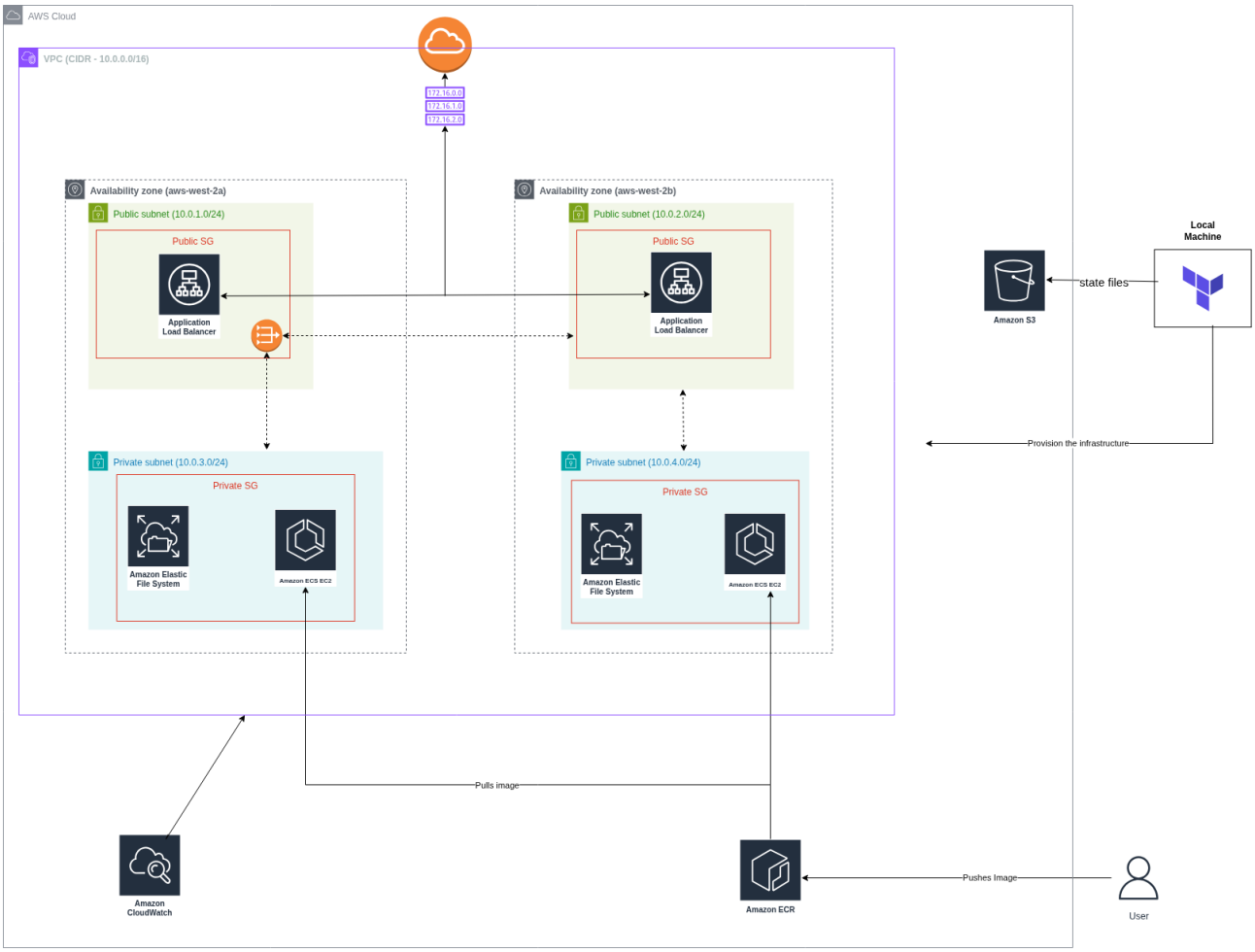
Objective

Deploy a complete ECS Cluster using **EC2 Launch Type** fronted by an **Application Load Balancer (ALB)** with **EFS persistent storage** and **CloudWatch logging**, all managed through Terraform.

Project Directory Structure



Architecture Diagram



Terraform Modular Structure

Below is the correct logical order in which the infrastructure is built and how each module contributes.

1. Configure the VPC Module

Purpose

Create the complete networking backbone required for ECS, ALB, EC2, and EFS.

Resources Created

- **1 VPC**
  - CIDR: `10.1.0.0/16`
  - Reason:
    - Large enough subnet space
    - Avoids overlap with other users (your earlier conflict issue)
- **2 Public Subnets**
  - Example:
    - `10.1.1.0/24`
    - `10.1.2.0/24`
  - Used for:
    - ALB
    - ECS EC2 instances (EC2 launch type requires public access + IMDS)
- **2 Private Subnets**
  - Example:
    - `10.1.3.0/24`
    - `10.1.4.0/24`
  - Used for:
    - EFS Mount Targets
    - Internal-only resources
- **Internet Gateway (IGW)**
  - Required for outbound access from ALB + EC2
- **Public Route Table**
  - Route: `0.0.0.0/0 → IGW`

        - Associated with both public subnets
    - **NAT Gateway**
        - 1 NAT in a public subnet
        - Elastic IP assigned
        - Allows **private** subnets to:
            - Download OS packages
            - Pull ECR images
            - Update CloudWatch agent
    - **Private Route Table**
        - Route: `0.0.0.0/0 → NAT Gateway`
        - Associated with both private subnets

2. Configure the Security Module

Purpose

Create all Security Groups following least-privilege principles.

Resources Created

1. ALB Security Group

- Inbound:
    - `80/tcp` from `0.0.0.0/0`
- Outbound:
    - Allow all (default)
- Reason:
    - ALB must be publicly reachable.

2. ECS EC2 Security Group

- Inbound:
    - `80/tcp` **from ALB SG** only
    - `22/tcp` only from **your IP** (optional)
- Outbound:
    - Allow all
- Reason
    - EC2 should only accept web traffic from ALB, not the public.

3. EFS Security Group

- Inbound:
    - `2049/tcp` (NFS) from ECS EC2 SG
- Outbound:
    - Allow all
- Why?
    - Ensures only ECS instances can mount EFS.

3. Configure the ECR Module

Purpose

Host your custom Nginx Docker image.

Resource Created

- **ECR Repository**
    - Name: e.g., `nginx-ecs-repo`
    - Lifecycle (optional): remove old images

4. Configure the EFS Module Persistent Storage

Purpose

Allow the Nginx containers to store shared data (e.g., HTML, logs, assets, uploaded content).

Resources Created

- **EFS Filesystem**
    - Highly available across AZs
    - Persistent even if ECS tasks restart
- **EFS Mount Targets**
    - One in each **private subnet**
- **Access Point**
    - Optional but recommended for ECS
    - Enforces correct permissions inside containers

Values

- Mount targets deployed in:
    - `10.1.3.0/24`
    - `10.1.4.0/24`
- SG attached:

- EFS SG → allows NFS from ECS EC2 SG

5. Configure the ECS Module

Purpose

Deploy the compute layer that will run your Nginx containers.

This module is unique because it orchestrates:

- EC2 autoscaling
- ECS registration
- Task definitions
- CloudWatch logging
- EFS mounts
- ALB integration

A. ECS Cluster

- Name: `ecs-cluster`
- Manages all ECS EC2 instances and tasks.

B. Launch Template

Includes:

- Data source AMI:
    - **Amazon Linux 2023 ECS-Optimized AMI**
- Instance type:
    - `t3.medium`
- Key pair (optional)
- EC2 SG attached
- User data:
    - Sets ECS_CLUSTER name
    - Ensures ECS agent auto-registers
    - No need to install Docker (AMI already includes it)

C. Auto Scaling Group

- Desired capacity: 2 EC2 instances
- Min: 1
- Max: 3
- Subnets: **public subnets**

- Why in public?
    - ECS EC2 must pull images from ECR via NAT if private;
      we keep them public to reduce complexity and avoid earlier issues.

D. ECS Instance IAM Role

Policies:

- `AmazonEC2ContainerServiceforEC2Role`
- `AmazonEC2ContainerRegistryReadOnly`
- EFS access via:
    - `AmazonElasticFileSystemClientFullAccess`
      (or a least-privilege version)

E. Task Execution Role

Used by ECS agent to:

- pull images from ECR
- push logs to CloudWatch

Policies:

- `AmazonECSTaskExecutionRolePolicy`

F. Task Definition

Defines:

- Container name
- ECR image URI
- Port: `80`
- CPU/memory
- Log configuration:
    - CloudWatch log group
- EFS mount:
    - Access point
    - `/usr/share/nginx/html` inside container

G. ECS Service

Configures:

- Desired count: 2 tasks
- Placement on EC2 instances

- Integration with ALB target group
- Deployment controller:
    - Rolling updates with zero downtime

6. Configure the ALB Module (Public Entry Point)

Purpose

Expose Nginx to the internet and route traffic to ECS tasks.

Resources Created

Application Load Balancer

- Subnets:
    - Public Subnets 1 & 2
- SG:
    - ALB SG
- Scheme:
    - Internet-facing

Listener

- Port `80`
- Default action → forward to target group

Target Group

- Protocol: HTTP
- Port: 80
- Target type: **instance** (because EC2 launch type)
- Health check:
    - Path: `/`
    - Interval: ~30 seconds

Listener Rule

- Forwards traffic to ECS service tasks automatically

Configure the Root `main.tf`

The root `main.tf` file orchestrates all Terraform modules and providers, acting as the central hub for your ECS + ALB + EFS infrastructure**.**

Module Calls (In Order):

- **VPC**: Creates networking (VPC, subnets, IGW, NAT).
- **Security**: Creates security groups for ALB, ECS, and EFS.
- **EFS**: Creates EFS filesystem + mount targets.
- **ECR**: Creates ECR repository for Nginx image.
- **ECS**: Deploys ECS cluster, EC2 instances, tasks, and services.
- **ALB**: Creates ALB, target group, and listener.

Terraform Deployment Commands

Initialize providers and modules

```
terraform init
```

Format code (optional)

```
terraform fmt
```

Validate configuration

```
terraform validate
```

View changes before applying

```
terraform plan
```

Deploy infrastructure

```
terraform apply
```

Custom Docker Image Build and Push

The deployment expects a custom NGINX image located in ECR. Build and push using standard Docker and AWS CLI commands.

Follow the commands and make sure you put the right directory (where you created your Dockerfile) while building the custom image.

Testing and Validation

Validate Provisioned Infrastructure (AWS Console)

After `terraform apply` completes successfully, validate the following components directly from the AWS Console:

VPC

- Verify the VPC, public subnets, and private subnets exist with correct CIDR ranges.
- Confirm public subnets are associated with the public route table and have IGW access.
- Confirm private subnets are associated with the private route table and route through NAT Gateway.

Security Groups

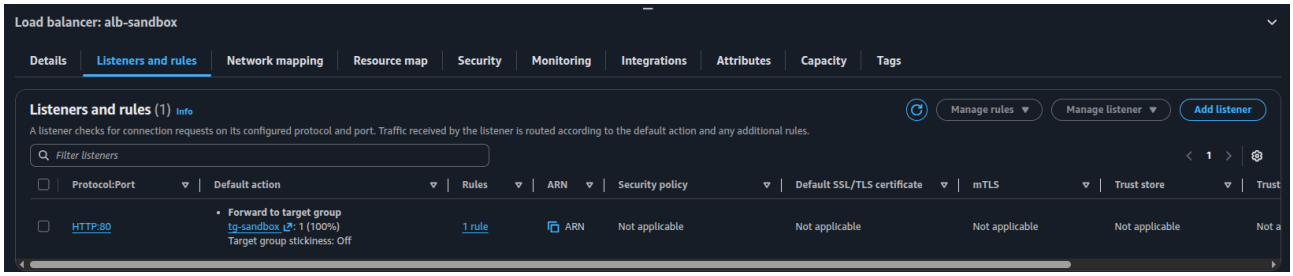
- ALB security group allows inbound HTTP 80 (or your defined port).
- ECS EC2 security group allows inbound traffic from ALB SG.
- EFS security group allows NFS (2049) from ECS EC2 SG.

ECS Cluster

- “EC2 Instances” tab shows your EC2 container instances registered and healthy.
- “Services” tab shows your ECS service in RUNNING state with desired tasks running.
- “Tasks” tab shows task status as RUNNING (not PENDING, STOPPED, or DRAINING).

ALB

- ALB listener (HTTP 80) is active.
- Target group associated with the ECS service shows:
    - 1 or more targets (your EC2 instances)
    - Health status: **healthy**



EFS

- File system exists.
- Mount targets are created for each private subnet.
- Access Point exists (one access point for the ECS tasks).

CloudWatch Logs

- A log group with your configured name (e.g., `/ecs/nginx`) exists.
- Log streams for each ECS task appear and show Nginx boot messages.

Validate Application Response

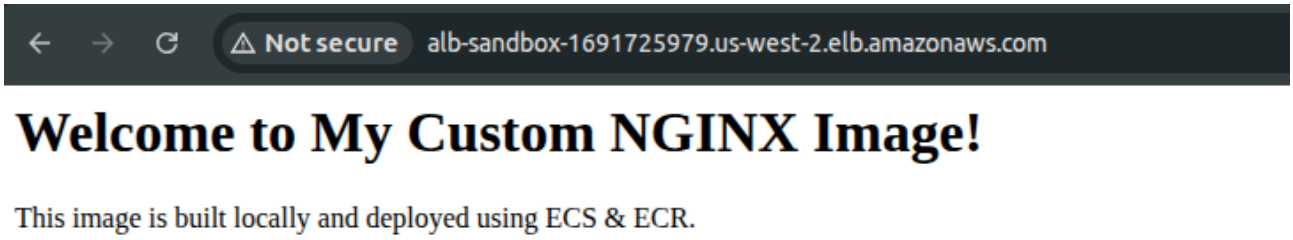
Step 1: Access the ALB DNS Name

From AWS Console → EC2 → Load Balancers → your ALB → copy the **DNS name**, visit in browser:

`http://<alb-dns-name>`

Expected behavior:

- Nginx responds with the custom page hosted on EFS.
- No 403, no 502 bad gateway errors.
- The page is consistent across refreshes (showing persistence via EFS).



Cleanup

To remove all resources:

`terraform destroy`

Type `yes` when prompted.