Name: Yash Mutatkar Date: 28 oct 2024

AWS Infrastructure Setup using Terraform

In this document, I will explain the AWS infrastructure I set up using Terraform. The primary objective of this setup is to create a scalable and secure environment for deploying applications using Amazon ECS (Elastic Container Service) with Fargate, which allows for serverless container management.

1. Creating a Virtual Private Cloud (VPC)

I created a Virtual Private Cloud (VPC) to isolate my resources in a dedicated network. The CIDR block used is '10.0.0.0/16', which allows for 65,536 IP addresses. This provides sufficient IP addresses for subnets and resources in the VPC.

2. Creating Public and Private Subnets

To organize my resources, I created both public and private subnets within the VPC:

Public Subnet: This subnet is configured with 'map_public_ip_on_launch = true', allowingresources launched in this subnet to have public IP addresses. This is ideal for resources that need direct access to the internet.

Private Subnet: This subnet does not assign public IPs to its resources. It is used for resources that should not be directly accessible from the internet.

3. Internet Gateway

I created an Internet Gateway to enable internet access for resources in the public subnet. This allows instances in the public subnet to communicate with external services.

4. NAT Gateway

To allow instances in the private subnet to access the internet (for updates, etc.) without being directly exposed to incoming traffic, I set up a NAT Gateway. This gateway routes outbound traffic to the internet while keeping the private subnet secure.

5. Route Tables

I created separate route tables for the public and private subnets:

Public Route Table: This route table directs outbound traffic to the Internet Gateway, allowing instances in the public subnet to access the internet.

Private Route Table: This route table routes outbound traffic through the NAT Gateway, enabling instances in the private subnet to access the internet securely.

6. Security Group

I created a Security Group to control inbound and outbound traffic:

Inbound Rules: I allowed incoming HTTP traffic (port 80) from any IP address. This enables external users to access the web applications hosted in my public subnet.

Outbound Rules: I permitted all outbound traffic, allowing resources in the security group to communicate freely with other services and the internet.

7. ECS Cluster and Service

I created an ECS Cluster named 'yash-example-cluster' to manage my containerized applications. The cluster will host services that run tasks defined in my ECS Task Definition.

The ECS Service is set up to run one instance of my task, which is defined to use the Nginx container. The service ensures that the specified number of task instances is running and replaces any failed tasks.

8. ECS Task Definition

The ECS Task Definition specifies the configuration for running containers:

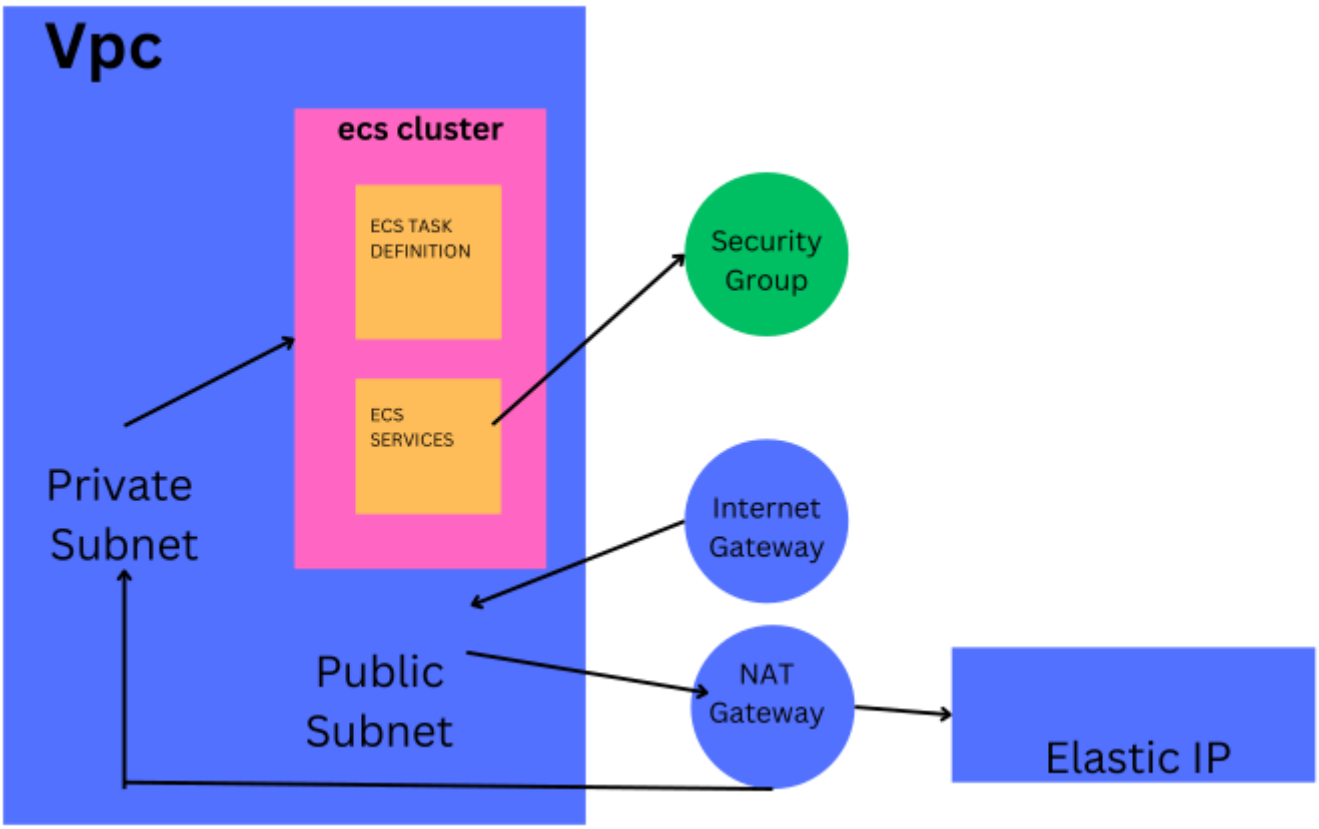
- I set the task to use the Fargate launch type, which allows AWS to manage the infrastructure.

- The container is configured to run the Nginx web server, with port 80 exposed for web traffic.

Conclusion

In summary, this Terraform configuration sets up a secure and efficient environment for hosting applications using AWS services. By utilizing a VPC, subnets, NAT gateways, and ECS.

Diagram


Vpc
ecs cluster
ECS TASK DEFINITION
ECS SERVICES
Security Group
Private Subnet
Internet Gateway
Public Subnet
NAT Gateway
Elastic IP