AWS Deployment Guide

Overview:

This guide details an optimal AWS deployment strategy for a dockerized application leveraging a Large Language Model (LLM) integrated with LangGraph and LangChain for node-based or agent-based reasoning.

Recommended AWS Architecture

- **Container**: Dockerized application including:
  - Integration with LangGraph and LangChain for reasoning logic.
  - LLM directly packaged and hosted within the container.
- **Elastic Container Registry (ECR)**:
  - Docker images are built and uploaded to AWS Elastic Container Registry.
  - **ECS Fargate** pulls container images directly from ECR for deployment.
- **Load Balancing**: Application Load Balancer (ALB) attached to ECS Fargate for secure and efficient distribution of HTTP(S) traffic.
- **Compute Layer**: ECS Fargate clusters, automatically provisioned and managed by AWS.
- **External API Access**: AWS API Gateway to manage and securely interact with external internet-based APIs or services

Environment Variable Management

- Securely store sensitive configuration and secrets using **AWS Secrets Manager**.
- Retrieve at runtime via ECS Task Definitions, enhancing security and flexibility.

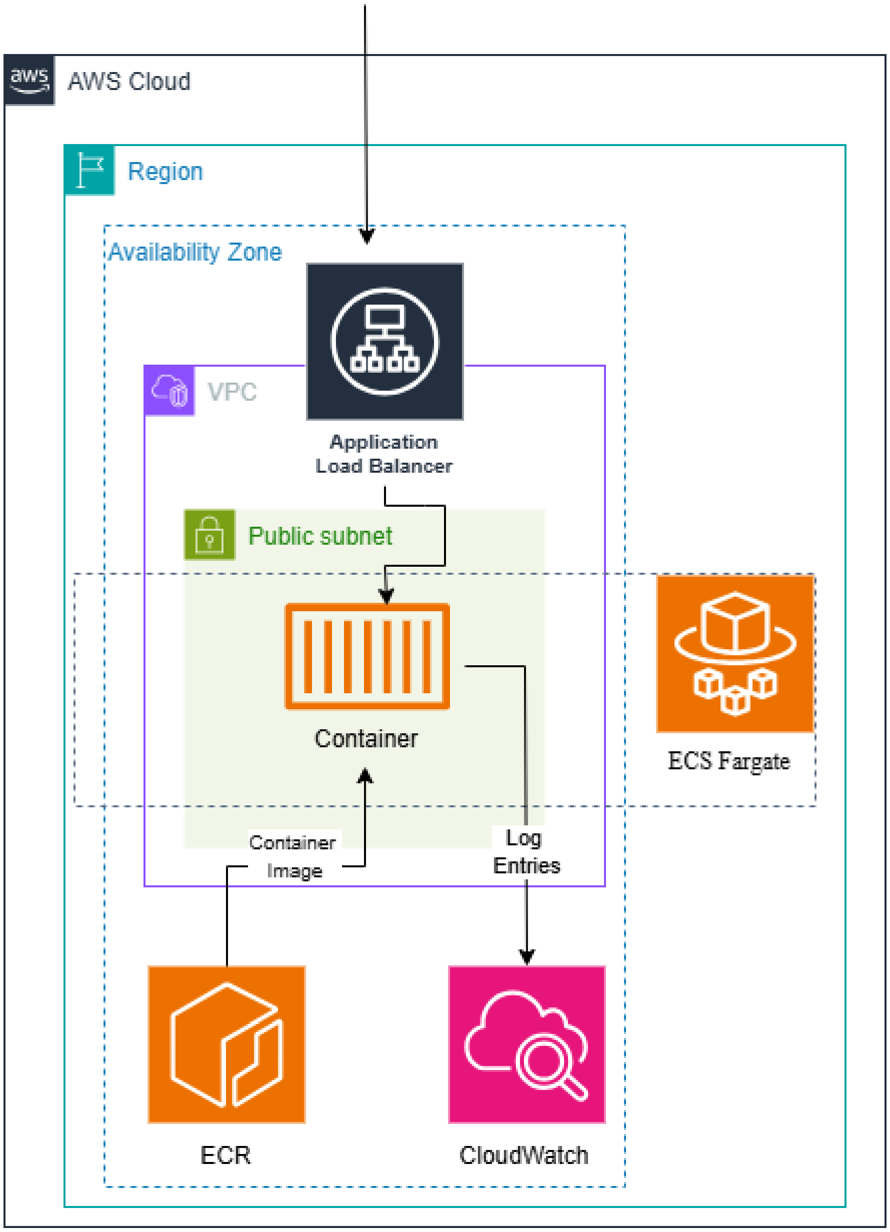
Scaling Strategy

- Implement automatic horizontal scaling via ECS Task Auto Scaling policies triggered by:
  - CPU and memory usage.
  - Traffic volume metrics from ALB.

Monitoring & Logging

- **Monitoring**:
  - AWS CloudWatch for metrics tracking (CPU, memory, latency, requests).
  - Configure CloudWatch Alarms for performance anomalies.
- **Logging**:
  - ECS container logs streamed to CloudWatch Logs.

Architecture Diagram


AWS Cloud
Region
Availability Zone
VPC
Application
Load Balancer
Public subnet
Container
ECS Fargate
Container
Image
Log
Entries
ECR
CloudWatch