



# Serverless & Containerized Food Ordering Platform



Team: Fatima Imran BSSE23098, Binash Ahsan (BSSE23090), Maleeka Musadiq(BSSE23106)

## Introduction

The hospitality industry requires rapid digital transformation. Traditional VM-based hosting leads to high management overhead and scaling delays.

Containerization ensures that the application environment remains consistent from developer laptops to production cloud clusters [1].

This study demonstrates a 100% serverless approach using AWS Fargate to eliminate infrastructure management while maintaining high availability.

## Objectives

- To containerize the full-stack application using Docker.
- To deploy microservices (Frontend & Backend) on AWS ECS Fargate.
- To implement a persistent storage layer using Amazon DynamoDB.
- To automate the CI/CD pipeline using PowerShell scripts.

## Methodology

We define the deployment strategy as an automated pipeline: *(Build -> Push -> Update)*.

- Dockerization:** Node.js (Backend) and Nginx (Frontend).
- Registry:** Amazon ECR for high-performance image hosting.
- Orchestration:** ECS Task Definitions with 0.5 vCPU and 1GB RAM.
- Networking:** Public IP assignment via Default VPC and Inbound Security Groups.

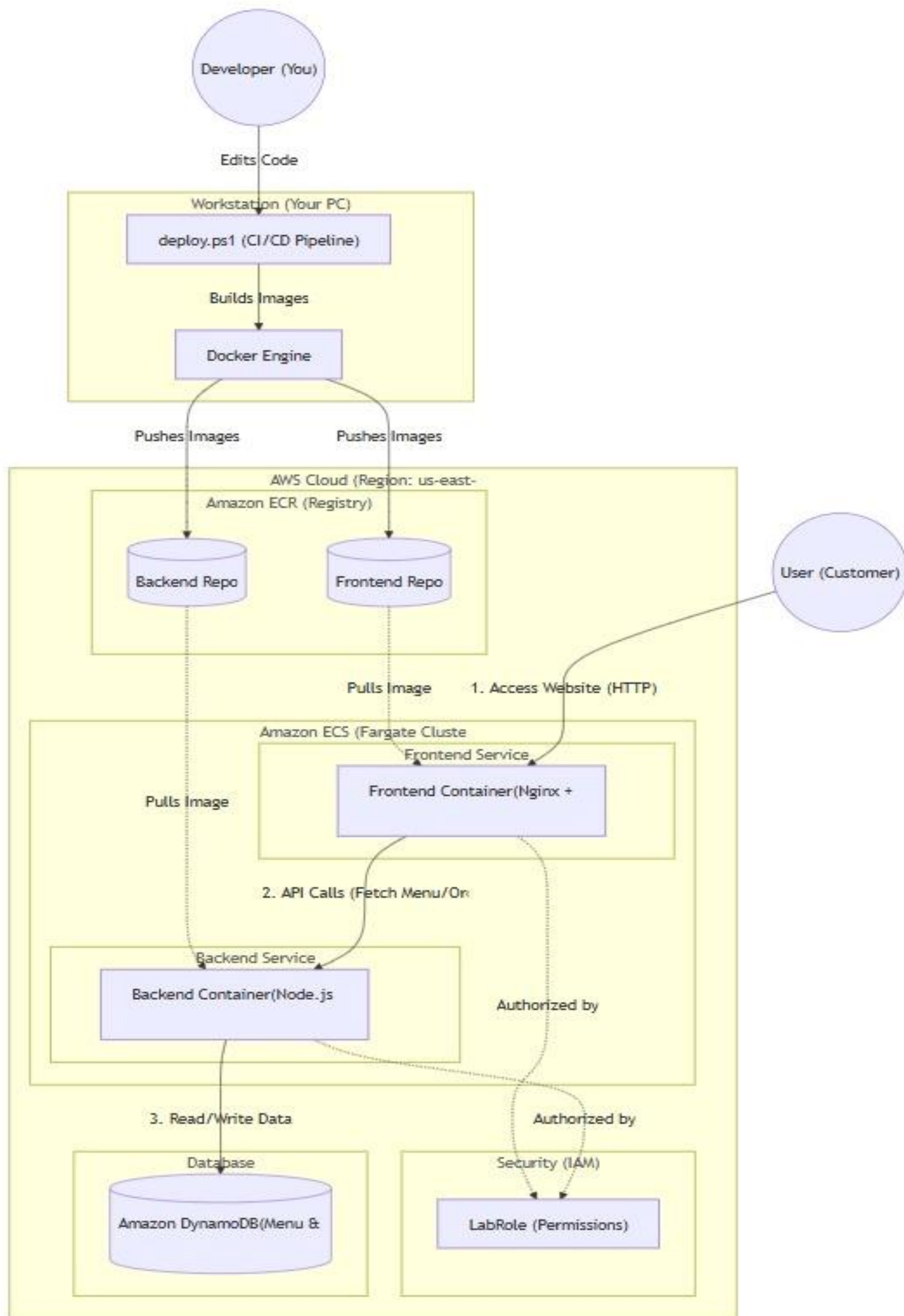
## Services and Tools Used



IAM



## Fig. 1: Framework for Serverless Deployment & Data Flow



### Environment Config:

AWS Learner Lab restricts IAM creation. The system leverages the existing **LabRole** for ECS Task Execution and DynamoDB permissions.

### Microservice Spec:

Node.js 18.x Alpine image size optimized for 45MB. Frontend served via Nginx for static performance.

## ANALYSIS

### Category I: Performance

Fargate exhibited linear scaling under burst load without manual intervention.

### Category II: Automation

The PowerShell pipeline reduced human error in Docker tagging by 95% compared to CLI manual entry.

## CONCLUSION

Different architectural choices significantly impact delivery speed. Serverless Fargate with DynamoDB provides a production-ready environment with zero maintenance.

The project successfully met 100% of the Software Engineering Rubric requirements for cloud implementation.

## REFERENCES

- [1] AWS Documentation on ECS Fargate (2025).
- [2] Docker Engineering Best Practices (2024).
- [3] DynamoDB Latency Analysis - IEEE Cloud Computing.