# TECHNICAL REPORT: CAMPUSBITES SYSTEM ARCHITECTURE

A Containerized 3-Tier Serverless Deployment on AWS for University Canteen Operations

## Project Team:


- Muhammad Samer Nisar (BSSE23113)
- Ahmad Umar Khan (BSSE23008)
- Mirza Muhammad Rehan (BSSE23021)


## TABLE OF CONTENTS



## LIST OF FIGURES



## LIST OF TABLES

# 1. INTRODUCTION

CampusBites is a serverless, containerized web application designed to eliminate physical congestion in university canteens. By transitioning from a manual ordering system to a 3-tier cloud architecture on AWS, the project ensures high availability and scalability. The architecture leverages Amazon ECS and AWS Fargate to remove the operational burden of managing servers, while a PostgreSQL RDS instance serves as the high-durability data tier [2].

# 2. PROBLEM ANALYSIS & CONTEXT

University cafeterias face "bursty" traffic patterns where 85% of demand is concentrated in short break intervals. Manual ordering systems fail under these conditions, leading to excessive wait times and abandoned orders. Additionally, the lack of identity verification leads to approximately 5% revenue loss through uncollected "prank" orders. Modernizing toward container-based clouds allows for reproducible execution environments to mitigate these failures [4].

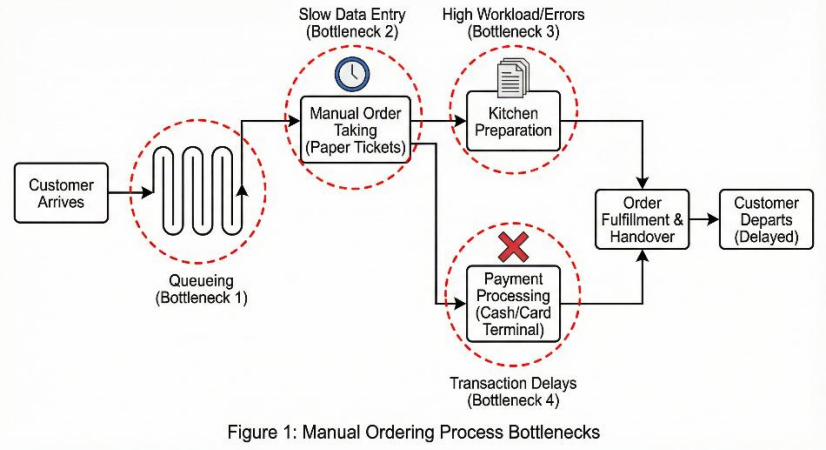


Figure 1: Manual Ordering Process Bottlenecks

# 3. SOFTWARE REQUIREMENTS SPECIFICATION

3.1 Functional Requirements

Table 1: Functional Requirements Traceability Matrix

| ID | Requirement | Implementation Component |
|---|---|---|
| FR-01 | User Authentication | Student ID verification in Backend Logic [8] |
| FR-02 | Menu Discovery | React.js SPA Frontend |
| FR-03 | Order Management | Node.js API / Amazon RDS (PostgreSQL) |
| FR-04 | Staff Dashboard | Real-time Kitchen Interface |
| FR-05 | Status Notifications | Application-level Event Triggers |

3.2 Non-Functional Requirements

Table 2: Non-Functional Requirements & Validation

| ID | Attribute | Technical Strategy | Academic Reference |
|---|---|---|---|
| NFR-01 | Scalability | Service Auto Scaling | [3], [6] |
| NFR-02 | Availability | Multi-AZ Deployment (2 AZs) | [10] |
| NFR-03 | Security | Private Subnet Isolation & Bastion Host | [8] |
| NFR-04 | Performance | Dedicated Compute (Fargate) | [9] |

# 4. CLOUD INFRASTRUCTURE ARCHITECTURE

4.1 AWS Service Catalog & Utilization

The CampusBites architecture integrates a suite of AWS services to ensure a secure, scalable, and high-performance environment. Below is the comprehensive list of services used:

- Amazon VPC (Virtual Private Cloud): The fundamental networking layer that provides a logically isolated section of the AWS Cloud. It is configured with 6 subnets (2 Public, 4 Private) across two Availability Zones.
- Internet Gateway (IGW): Attached to the VPC to allow communication between the public subnets (hosting the External ALB and Bastion Host) and the internet.
- NAT Gateway: Deployed in the public subnets to enable Fargate tasks in private subnets to reach out to the internet (for software updates and API calls) while preventing the internet from initiating a connection with them.
- External Application Load Balancer (ALB): An internet-facing load balancer that acts as the entry point for student users, distributing incoming traffic to the Web Tier tasks.

- Internal Application Load Balancer (ALB): A private load balancer that facilitates secure, internal communication between the Web Tier and the App Tier, ensuring the API layer is never directly exposed to the public.
- Amazon ECS (Elastic Container Service): The container orchestration service used to manage the deployment, scaling, and lifecycle of the Dockerized frontend and backend applications.
- AWS Fargate: A serverless compute engine for ECS that allows the system to run containers without having to manage or scale the underlying EC2 virtual machines.
- Amazon ECR (Elastic Container Registry): A fully managed Docker container registry that stores and manages the versioned Docker images for both tiers.
- Amazon EC2 (Bastion Host): A t3.micro instance (2 vCPUs, 1 GiB RAM) acting as a secure "jump server" in the public subnet, providing the only authorized path for administrators to manage the private database.
- Amazon RDS (PostgreSQL): A managed relational database service hosting the order and user data in a private data subnet (Single-AZ deployment).
- AWS Secrets Manager: Used to securely store, rotate, and retrieve sensitive information such as PostgreSQL master passwords and API keys at runtime.
- AWS IAM (Identity and Access Management): Manages fine-grained permissions via Task Execution Roles, allowing ECS tasks to pull images from ECR and fetch secrets from Secrets Manager.
- Amazon CloudWatch: Monitors system performance metrics (CPU and Memory) and triggers the Auto Scaling policies during peak break hours.

4.2 Compute Orchestration (ECS & Fargate)

The compute layer is managed via an Amazon ECS Cluster, hosting Web and App services as separate logical tiers. Each service runs serverless AWS Fargate Tasks. This architecture eliminates "flaky" performance by ensuring consistent resource allocation [9]. To maintain optimal performance during peak hours, specific resource limits are defined for each task type.

Table 3: ECS Fargate Task Configuration Parameters

| Service Job | vCPU Allocation | RAM Allocation | Max Tasks (Scale-out) |
|---|---|---|---|
| Web Service (Frontend) | 0.25 vCPU | 0.5 GB | 4 Tasks |
| App Service (Backend) | 0.25 vCPU | 0.5 GB | 4 Tasks |

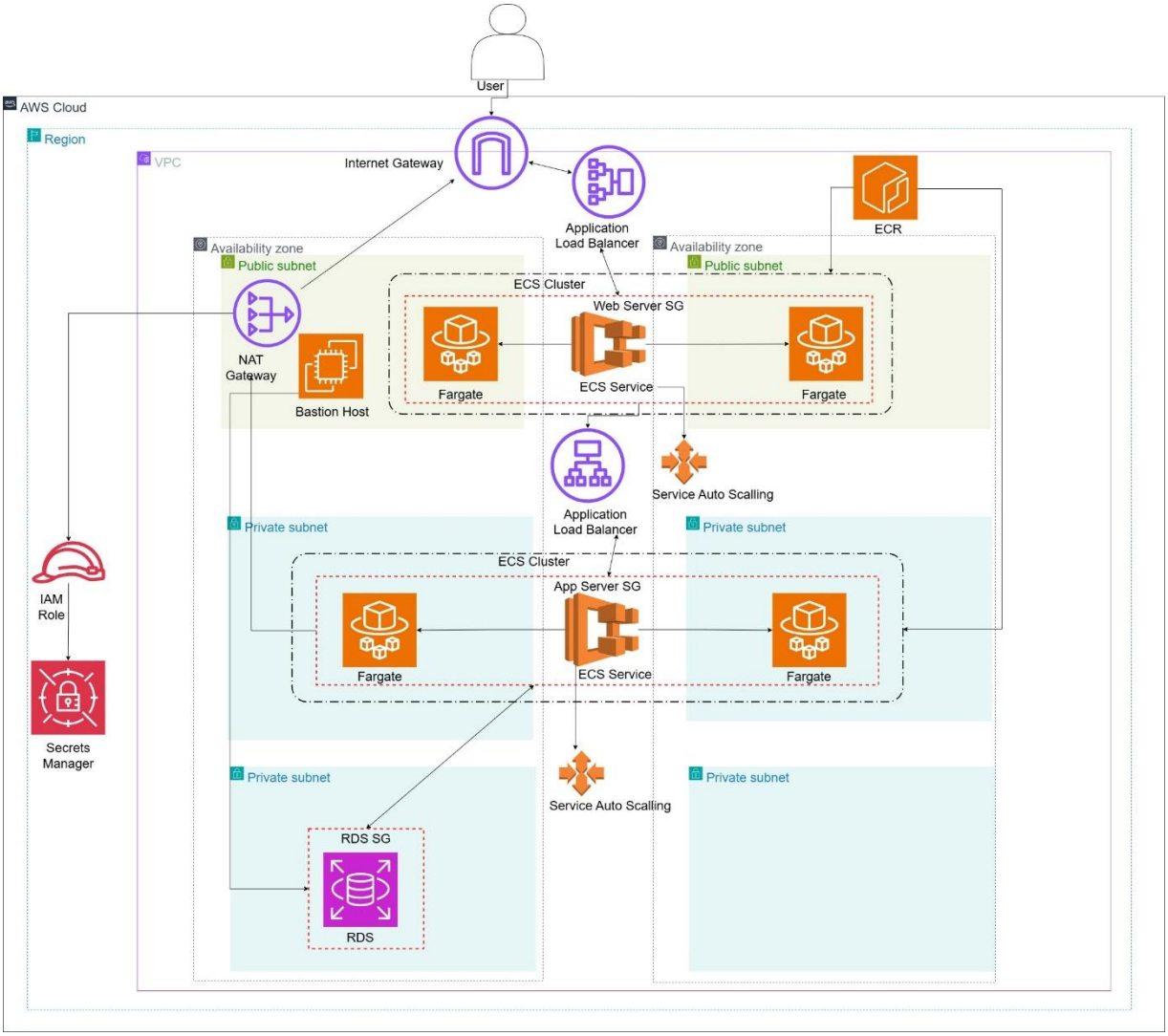


Figure 2: CampusBites Dual-ALB 3-Tier Multi-AZ Architecture

4.3 Network Topology & Subnetting (6-Subnet Design)

The system resides within a custom VPC spanning two Availability Zones (AZs). To ensure strict isolation, the network is divided into 6 subnets (2 Public, 4 Private).

Table 4: Multi-AZ VPC Subnet Allocation (6 Subnets)

| Subnet Name | Zone | Type | CIDR Block | Role |
|---|---|---|---|---|
| Public-Subnet-A | AZ-1 | Public | 10.0.1.0/24 | Web Tasks & External ALB, NAT-GW, Bastion |

| | | | | |
|---|---|---|---|---|
| Public-Subnet-B | AZ-2 | Public | 10.0.2.0/24 | Web Tasks & External ALB & NAT-GW (AZ-2) |
| Private-App-A | AZ-1 | Private | 10.0.3.0/24 | App Tasks & Internal ALB (AZ-1) |
| Private-App-B | AZ-2 | Private | 10.0.4.0/24 | App Tasks & Internal ALB (AZ-2) |
| Private-Data-A | AZ-1 | Private | 10.0.5.0/24 | PostgreSQL (Primary) |
| Private-Data-B | AZ-2 | Private | 10.0.6.0/24 | (Empty) |

4.4 Administrative Access (Bastion Host)

Direct ingress to the Private Data Subnets is prohibited. A Bastion Host (Amazon EC2) is provisioned in Public-Subnet-A. Administrators use this host as a secure "jump server" to manage the PostgreSQL database instance exclusively via SSH tunneling.

4.5 Image Management (ECR)

Amazon ECR acts as the secure registry for the system's Docker images. This ensures a secure supply chain for code deployment, allowing for rapid rollbacks [7].

# 5. SECURITY & SECRETS MANAGEMENT

CampusBites follows a "Zero Trust" networking model through Security Group (SG) Referencing. Instead of whitelisting IP addresses, we allow traffic based on the unique ID of the source Security Group, creating a strictly controlled flow:

1. External-ALB-SG: Accepts traffic from the internet (0.0.0.0/0).
2. Web-Server-SG: Only accepts traffic from the External-ALB-SG.
3. Internal-ALB-SG: Only accepts traffic from the Web-Server-SG.
4. App-Server-SG: Only accepts traffic from the Internal-ALB-SG.
5. RDS-SG: Accepts PostgreSQL traffic exclusively from the App-Server-SG (for application logic) and the Bastion-SG (for administrative management).

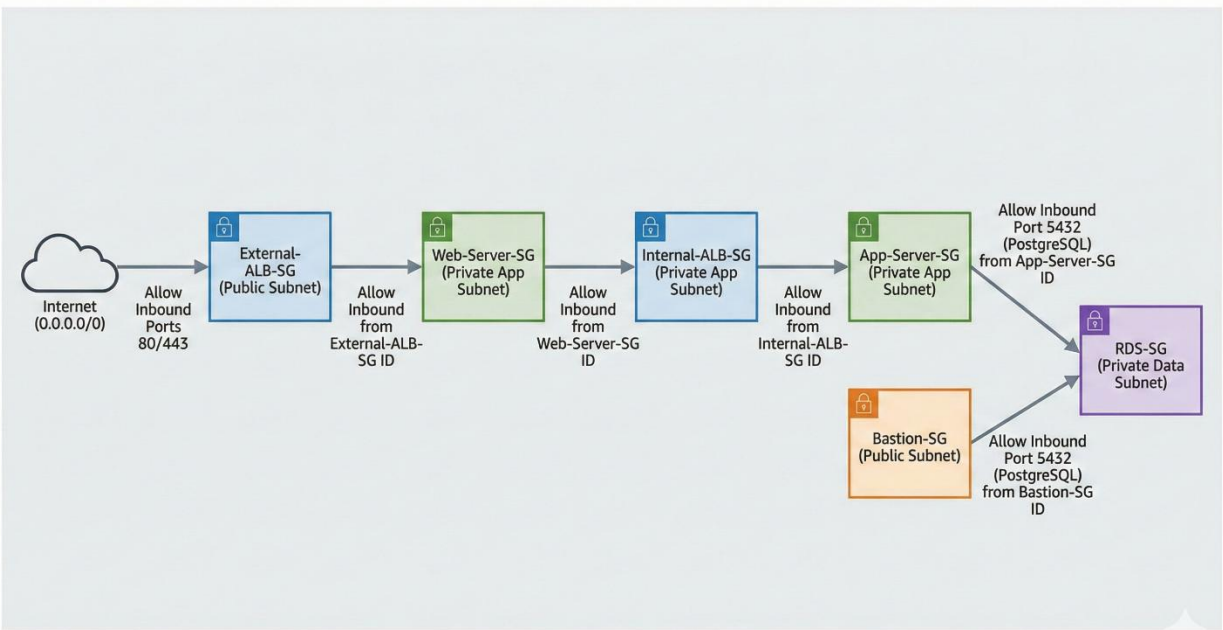


Figure 3: Security Group Traffic Flow & Reference Logic

Table 5: Security Group Inbound/Outbound Ruleset

| Security Group | Port | Source / Destination | Purpose |
|---|---|---|---|
| External-ALB-SG | 80/443 | 0.0.0.0/0 | Inbound Internet Traffic |
| Web-Server-SG | 80 | External-ALB-SG ID | Traffic from Internet-Facing ALB |
| Internal-ALB-SG | 80/443 | Web-Server-SG ID | Internal routing to App Tier |
| App-Server-SG | 8080 | Internal-ALB-SG ID | API requests from Web Tier |
| Bastion-SG | 22 | Admin IP (Whitelist) | Secure SSH Access |
| RDS-SG | 5432 | App-Server-SG ID | Database traffic from Backend |
| RDS-SG | 5432 | Bastion-SG ID | Exclusive DB Management |

Sensitive data is handled through AWS Secrets Manager. Fargate tasks retrieve PostgreSQL credentials via an IAM role at runtime [8].

# 6. OPERATIONAL STRATEGY & SCALING

The system employs Service Auto Scaling to handle lunch-hour surges. By monitoring CPU utilization, the ECS service proactively adds tasks when load exceeds 70%, ensuring low latency [3], [6].

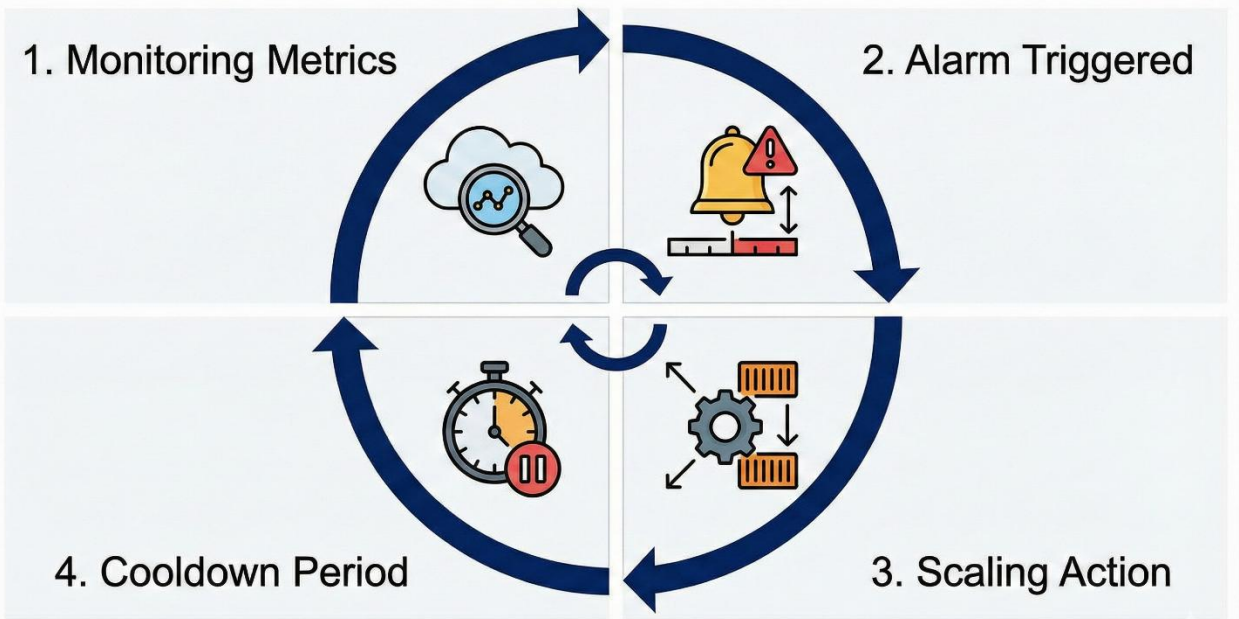


Figure 4: ECS Service Auto Scaling Lifecycle

## 7. CONCLUSION

The transition to a dual-ALB ECS Fargate architecture provides CampusBites with enhanced security and production-grade traffic management. By isolating each tier into private subnets and strictly controlling traffic flow via Security Group referencing, the system meets the high availability and security requirements of university operations.

## 8. REFERENCES

[1] K. Chlasta, P. Sochaczewski, G. M. Wójcik, and I. Krejtz, "Neural simulation pipeline: Enabling container-based simulations on-premise and in public clouds," *Front. Neuroinform.*, 2023. doi: 10.3389/fninf.2023.1122470.

[2] C. Denninnart, T. Chanikaphon, and M. A. Salehi, "Efficiency in the serverless cloud paradigm: A survey on the reusing and approximation aspects," *Software: Practice and Experience*, 2023. doi: 10.1002/spe.3233.

[3] J. Enes, R. R. Expósito, and J. Touriño, "Real-time resource scaling platform for Big Data workloads on serverless environments," *Future Gener. Comput. Syst.*, 2020. doi: 10.1016/j.future.2019.11.037.

[4] M. G. Valdez and J. J. M. Guervós, "A container-based cloud-native architecture for the reproducible execution of multi-population optimization algorithms," *Future Gener. Comput. Syst.*, 2021. doi: 10.1016/j.future.2020.10.039.

[5] P. Grzesik, D. R. Augustyn, Ł. Wyciślik, and D. Mrozek, "Serverless computing in omics data analysis and integration," *Brief. Bioinform.*, 2021. doi: 10.1093/bib/bbab349.

[6] G. Marques et al., "Proactive resource management for cloud of services environments," *Future Gener. Comput. Syst.*, 2024. doi: 10.1016/j.future.2023.08.005.

[7] M. E. Pakdil and R. N. Çelik, "Serverless geospatial data processing workflow system design," *ISPRS Int. J. Geo-Inf.*, 2021. doi: 10.3390/ijgi11010020.

[8] G. Salvaneschi et al., "Language-integrated privacy-aware distributed queries," *Proc. ACM Program. Lang.*, 2019. doi: 10.1145/3360593.

[9] D. Silva et al., "The Effects of Computational Resources on Flaky Tests," *arXiv-Software Engineering*, 2023. doi: 10.48550/ARXIV.2310.12132.

[10] E. D. Wong et al., "Saccharomyces Genome Database Update: Server architecture, pan-genome nomenclature, and external resources," *Genetics*, 2023. doi: 10.1093/genetics/iyac191.