

Cloud-Native Hostel Management System: Architecture and Optimization

Instructor:
Dr. Zunnurain

Student Names:
Asma Haider BSSE23051
Uzair Abdullah BSSE23075

Table of Contents

Cloud-Native Hostel Management System: Architecture and Optimization	1
Project Summary.....	3
Aim and Objectives	4
Architecture Diagram.....	4
Core Components & Purpose	5
Infrastructure & Networking	5
Application & Data Layer	5
DevOps & Integration	5
Technology Stack Summary	5
DevOps & CI/CD Pipeline	6
AWS Architecture Design, Implementation and Screenshots	6
AWS account setup	7
IAM roles and policies.....	8
VPC/subnets/security groups	8
VPC.....	8
SUBNETS (PRIVATE AND PUBLIC).....	9
SECURITY GROUPS (ALB – APP – CLOUD9 – DB).....	10
EC2, S3, RDS or any used service	11
S3.....	11
RDS	11
Deployment process	12
ECS-CLUSTER	12
ECR- REPOSITORY.....	13
Testing workflow.....	14
Monitoring / logging (CloudWatch).....	16
Final output.....	16
List of Figures:	17
List of Tables:	17
References:	17

Executive Summary

This project involved the development and deployment of a containerized web application designed to modernize student record management and room allocation systems. By migrating from a legacy monolithic architecture to AWS Fargate, the solution leverages serverless container orchestration to ensure high availability and automated scaling without the overhead of managing underlying infrastructure.

A critical component of this modernization was addressing the challenge of ephemeral storage inherent in containerized environments; this was achieved by integrating cloud-native persistence layers, ensuring that critical student data and allocation records remain durable and consistent across container lifecycles.

Introduction

The administration of university hostels is a data-intensive domain requiring the efficient handling of student records, room allocations, fee collection, and facility management. In the context of modern educational institutions, the demand for real-time access to these services has grown exponentially. Historically, hostel management has relied on **legacy monolithic applications** hosted on on-premise physical servers. While these traditional systems served their purpose initially, they lack the agility required for today's dynamic user base. As institutions scale, the "monolithic" approach creates significant bottlenecks. A single server failure can render the entire system inaccessible, and maintenance often requires significant downtime, disrupting administrative workflows. Furthermore, the global shift towards **Cloud-Native computing** highlights the obsolescence of managing physical infrastructure manually. This project introduces a **Cloud-Native Hostel Management System (HMS)** designed to address these legacy limitations. By migrating the application logic to **AWS Fargate (Serverless ECS)** and decoupling the data layer using **Amazon RDS**, this project aims to demonstrate how modern containerization and cloud orchestration can transform a static, fragile administrative tool into a highly available, secure, and auto-scaling platform.

Problem Statement

The primary problem this project addresses is the **inherent rigidity and fragility of on-premise monolithic architectures** in high-demand educational environments.

Current legacy systems face several critical real-world challenges:

- **Inability to Scale (Traffic Spikes):** During the start of a semester, thousands of students attempt to log in simultaneously to book rooms. Traditional on-premise servers often crash under this load because they cannot auto-scale. They require manual hardware upgrades (vertical scaling) which is costly and slow.
- **Single Point of Failure (SPOF):** Legacy architectures often rely on a single physical server. If this hardware fails or experiences a power outage, the entire hostel management system goes offline, halting all administrative processes.
- **Ephemeral Storage Risks:** As organizations move to basic containerization (Docker) without cloud orchestration, they face the "Ephemeral Storage" problem. If a container crashes or restarts, any data (such as student profile pictures or logs) stored locally is permanently lost.
- **Security Vulnerabilities:** Many legacy systems expose their database ports directly to the local network or internet for ease of access, creating significant security loopholes. Additionally, reliance on static passwords without Multi-Factor Authentication (MFA) leaves sensitive student data vulnerable to unauthorized access.

- **Operational Overhead:** The manual effort required to patch OS security, manage server cooling, and handle hardware maintenance diverts IT resources away from feature development and improving the student experience.

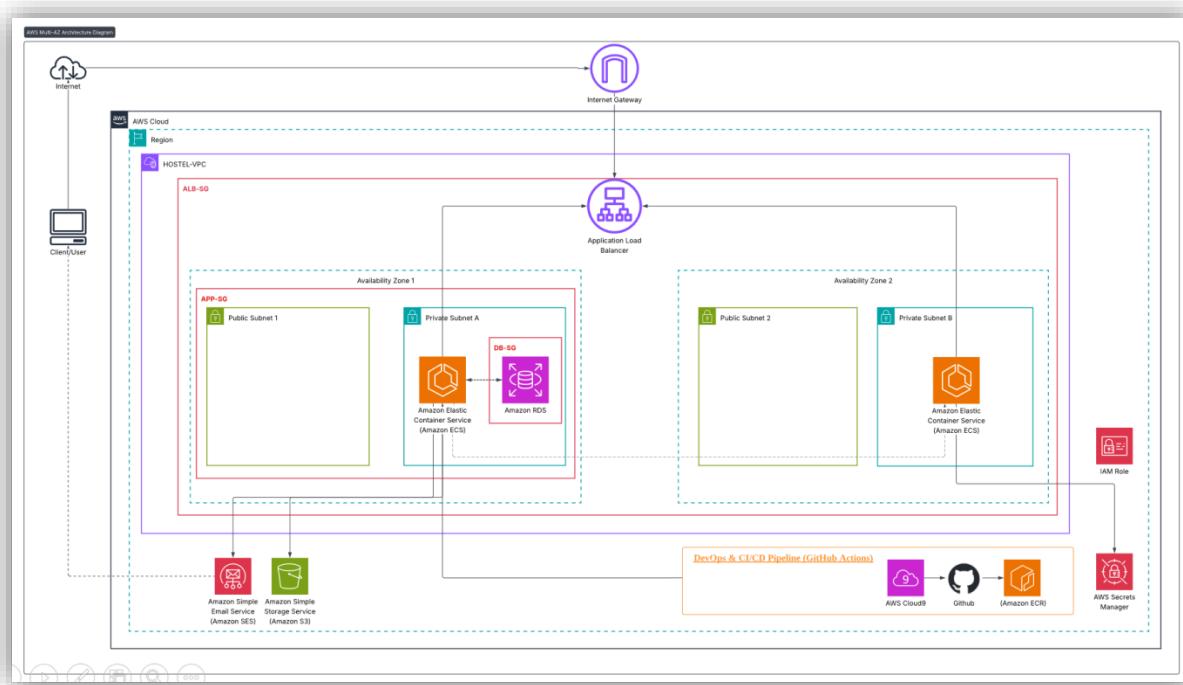
This project seeks to resolve these issues by re-architecting the system into a **Stateless, Serverless, and Multi-AZ (Availability Zone)** solution on AWS.

Aim and Objectives

This system leverages a serverless container strategy using AWS ECS Fargate, which eliminates manual infrastructure management while providing automated auto-scaling to meet fluctuating demand. The architecture is built for high availability and security, deploying resources across Multi-Availability Zones and isolating the database within private subnets to ensure it remains inaccessible from the public internet.

Media storage is externalized to Amazon S3 for durable object management, and application security is bolstered by a custom OTP logic integrated via SMTP to prevent unauthorized access. Additionally, the platform supports administrative efficiency through a bulk upload feature for processing large datasets via Excel.

Architecture Diagram



This AWS architecture represents a highly available, multi-tiered web application deployed across two Availability Zones (AZs) within a single Region. It utilizes a containerized approach for the application layer and a managed relational database for the data layer, all supported by a modern CI/CD pipeline.

Core Components & Purpose

- Frontend/Backend: Flask Application (Dockerized).
- Database: Amazon RDS (MySQL).
- Storage: Amazon S3 (Profile Images).
- Traffic: Application Load Balancer (ALB).
- Auth: Google SMTP.

Infrastructure & Networking

- VPC (Virtual Private Cloud): Provides a logically isolated section of the AWS Cloud where you can launch AWS resources in a defined virtual network.
- Public Subnets: Host the Application Load Balancer (ALB), which acts as the single point of contact for clients and distributes incoming application traffic across multiple targets.
- Private Subnets: Host the application containers and database instances, shielding them from direct internet access for enhanced security.
- Availability Zones (AZ1 & AZ2): Ensure high availability and fault tolerance; if one AZ fails, the application remains operational in the other.

Component	Resource Name	CIDR/Setting	Purpose
VPC	Hostel-VPC	10.0.0.0/16	Network Isolation
Public Subnet	subnet-public-1/2	10.0.0.0/20	Hosts Load Balancer
Private Subnet	subnet-private-1/2	10.0.128.0/20	Hosts App Containers
Availability Zones	us-east-1a, us-east-1b	Multi-AZ	High Availability

Application & Data Layer

- Amazon ECS (Elastic Container Service): Orchestrates and runs containerized applications (often using AWS Fargate for serverless management).
- Amazon RDS (Relational Database Service): A managed database service that, in this "Multi-AZ" setup, maintains a primary instance in one AZ and a standby in another for automatic failover.
- Amazon S3: Used for scalable object storage, typically for static assets or backups.
- AWS Secrets Manager: Securely stores and manages sensitive information like database credentials and API keys.

DevOps & Integration

- GitHub Actions: Automates the CI/CD workflow, triggering builds and deployments whenever code changes are pushed to the repository.
- Amazon ECR (Elastic Container Registry): A fully managed Docker container registry used to store and manage container images.
- AWS Cloud9: A cloud-based IDE used by developers to write, run, and debug code directly within the AWS environment.
- SMTP (Simple Mail Transfer Protocol): A standard email sending service used by the application to send notifications or transactional emails through an email server.

Technology Stack Summary

Layer	Technology
Language	Python 3.11 (Flask Framework)
Frontend	HTML5, CSS3 (Particles.js), Bootstrap 5
Container Engine	Docker
Orchestrator	AWS Fargate (Serverless ECS)
Database	Amazon RDS (MySQL 8.0)
Cloud Storage	Amazon S3
Email Service	Google SMTP
Infrastructure	VPC, Internet Gateway, ALB
CI/CD	GitHub Actions

DevOps & CI/CD Pipeline

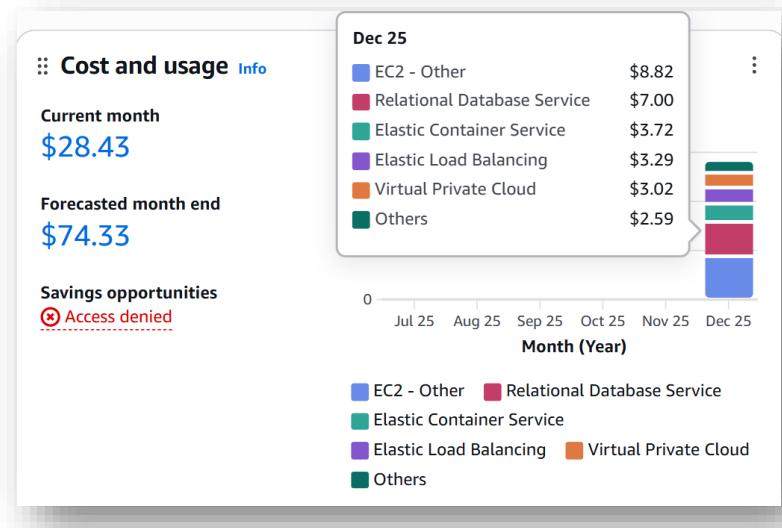
We moved away from manual deployments to a modern GitOps workflow.

- Source Control: GitHub (main branch).
- Pipeline Orchestrator: GitHub Actions.
- Container Registry: Amazon ECR.

The Pipeline (deploy.yml) Workflow:

1. Trigger: Developer pushes code to GitHub.
2. Checkout: GitHub Runner pulls the latest code.
3. Auth: Runner authenticates with AWS using secure Secrets (AWS_ACCESS_KEY_ID, etc.).
4. Build: Docker image is built from the Dockerfile.
5. Push: Image is tagged and pushed to Amazon ECR.
6. Deploy: The runner commands AWS ECS to update the service. ECS drains old connections and spins up new Fargate tasks with the updated image.

AWS Architecture Design, Implementation and Screenshots



AWS account setup

awsacademy.instructure.com/login/canvas

aws academy

Username
bsse23051@itu.edu.pk

Password

Stay signed in
[Forgot Password?](#)

Log In

Help Privacy Policy Cookie Notice Acceptable Use Policy
Facebook X.com

INSTRUCTURE
Meet the Instructure Learning Platform:
Canvas LMS Mastery Connect Elevate Analytics Impact

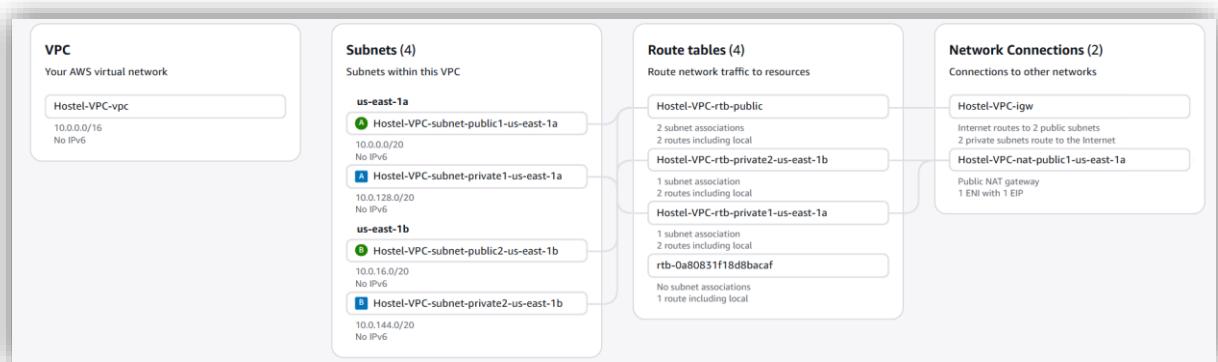
IAM roles and policies

The screenshot shows the AWS IAM Roles page with 24 entries. The columns are Role name, Trusted entities, and Last activity. The roles listed include AWS Service roles like AWSLambdaRoleForCloudWatchEvents, AWSServiceRoleForAmazonCloudWatchLogs, and AWSServiceRoleForAmazonCloudWatchMetrics, as well as custom roles like c184191a47750351131067651w891377334-LambdaSLRRole-8pTE3qXY3jHw.

Role name	Trusted entities	Last activity
AWSLambdaRoleForCloudWatchEvents	AWS Service: cloudwatchlogs (Service-Linked)	1 hour ago
AWSServiceRoleForAmazonCloudWatchLogs	AWS Service: events (Service-Linked)	-
AWSServiceRoleForAmazonCloudWatchMetrics	AWS Service: cloudwatchmetrics (Service-Linked)	9 minutes ago
AWSServiceRoleForAmazonCloudWatchStreams	AWS Service: elasticache (Service-Linked)	-
AWSServiceRoleForAmazonCloudWatchStreams	AWS Service: elasticloadbalancing (Service-Linked)	6 days ago
AWSServiceRoleForAmazonCloudWatchStreams	AWS Service: organizations (Service-Linked)	-
AWSServiceRoleForAmazonCloudWatchStreams	AWS Service: rds (Service-Linked)	12 minutes ago
AWSServiceRoleForAmazonCloudWatchStreams	AWS Service: support (Service-Linked)	-
AWSServiceRoleForAmazonCloudWatchStreams	AWS Service: trustedadvisor (Service-Linked)	-
c184191a47750351131067651w891377334-LambdaSLRRole-8pTE3qXY3jHw	AWS Service: eks	-
c184191a47750351131067651w891377334-LambdaSLRRole-8pTE3qXY3jHw	AWS Service: ec2	-
c184191a47750351131067651w891377334-LambdaSLRRole-8pTE3qXY3jHw	AWS Service: lambda	7 days ago
EMR_AutoScaling_DefaultRole	AWS Service: application-autoscaling (Service-Linked)	-

VPC/subnets/security groups

VPC



SUBNETS (PRIVATE AND PUBLIC)

subnet-00a7ca485b6ee6a07 / Hostel-VPC-subnet-private1-us-east-1a

Details		Flow logs	Route table	Network ACL	CIDR reservations	Sharing	Tags
Details							
Subnet ID	subnet-00a7ca485b6ee6a07	State	Available	Block Public Access	<input type="radio"/> Off		
IPv4 CIDR	10.0.128.0/20	IPv6 CIDR	-	IPv6 CIDR association ID	-		
Availability Zone	use1-az1 (us-east-1a)	VPC	vpc-0bc7115f56e0762db Hostel-VPC-vpc	Route table	rtb-0724a8933bf65cc35 Hostel-VPC-rtb-private1-us-east-1a		
Network ACL	aci-035c244ff43c7c3d	Auto-assign public IPv4 address	No	Auto-assign IPv6 address	No		
Auto-assign customer-owned IPv4 address	No	Outpost ID	-	IPv4 CIDR reservations	-		
IPv6 CIDR reservations	-	Hostname type	IP name	Resource name DNS A record	Disabled		
Resource name DNS AAAA record	Disabled	Owner	891377334911				

subnet-0f506e39f264fba8d / Hostel-VPC-subnet-private2-us-east-1b

Details		Flow logs	Route table	Network ACL	CIDR reservations	Sharing	Tags
Details							
Subnet ID	subnet-0f506e39f264fba8d	State	Available	Block Public Access	<input type="radio"/> Off		
IPv4 CIDR	10.0.144.0/20	IPv6 CIDR	-	IPv6 CIDR association ID	-		
Availability Zone	use1-az2 (us-east-1b)	VPC	vpc-0bc7115f56e0762db Hostel-VPC-vpc	Route table	rtb-0c54a4bd2b1e112 Hostel-VPC-rtb-private2-us-east-1b		
Network ACL	aci-033c244ff43c7c3d	Auto-assign public IPv4 address	No	Auto-assign IPv6 address	No		
Auto-assign customer-owned IPv4 address	No	Outpost ID	-	IPv4 CIDR reservations	-		
IPv6 CIDR reservations	-	Hostname type	IP name	Resource name DNS A record	Disabled		
Resource name DNS AAAA record	Disabled	Owner	891377334911				

subnet-0bca727cbace55dd3 / Hostel-VPC-subnet-public1-us-east-1a

Details		Flow logs	Route table	Network ACL	CIDR reservations	Sharing	Tags
Details							
Subnet ID	subnet-0bca727cbace55dd3	State	Available	Block Public Access	<input type="radio"/> Off		
IPv4 CIDR	10.0.0.0/20	IPv6 CIDR	-	IPv6 CIDR association ID	-		
Availability Zone	use1-az1 (us-east-1a)	VPC	vpc-0bc7115f56e0762db Hostel-VPC-vpc	Route table	rtb-005249f5f5028dd810 Hostel-VPC-rtb-public		
Network ACL	aci-035c244ff43c7c3d	Auto-assign public IPv4 address	No	Auto-assign IPv6 address	No		
Auto-assign customer-owned IPv4 address	No	Outpost ID	-	IPv4 CIDR reservations	-		
IPv6 CIDR reservations	-	Hostname type	IP name	Resource name DNS A record	Disabled		
Resource name DNS AAAA record	Disabled	Owner	891377334911				

subnet-0f70eb0fd22d4cc6 / Hostel-VPC-subnet-public2-us-east-1b

Details		Flow logs	Route table	Network ACL	CIDR reservations	Sharing	Tags
Details							
Subnet ID	subnet-0f70eb0fd22d4cc6	State	Available	Block Public Access	<input type="radio"/> Off		
IPv4 CIDR	10.0.16.0/20	IPv6 CIDR	-	IPv6 CIDR association ID	-		
Availability Zone	use1-az2 (us-east-1b)	VPC	vpc-0bc7115f56e0762db Hostel-VPC-vpc	Route table	rtb-005249f5f5028dd810 Hostel-VPC-rtb-public		
Network ACL	aci-035c244ff43c7c3d	Auto-assign public IPv4 address	No	Auto-assign IPv6 address	No		
Auto-assign customer-owned IPv4 address	No	Outpost ID	-	IPv4 CIDR reservations	-		
IPv6 CIDR reservations	-	Hostname type	IP name	Resource name DNS A record	Disabled		
Resource name DNS AAAA record	Disabled	Owner	891377334911				

SECURITY GROUPS (ALB - APP - CLOUD9 - DB)

sg-0bbfe85bd16b6cfef - ALB-SG

Details			
Security group name ALB-SG	Security group ID sg-0bbfe85bd16b6cfef	Description Allow traffic from internet	VPC ID vpc-0bc7115f56e0762db
Owner 891377334911	Inbound rules count 1 Permission entry	Outbound rules count 1 Permission entry	

[Inbound rules](#) | [Outbound rules](#) | [Sharing](#) | [VPC associations](#) | [Tags](#)

Inbound rules (1)

Search									
<input type="checkbox"/>	Name	Security group rule ID	IP version	Type	Protocol	Port range	Source		Description
<input type="checkbox"/>	-	sgr-0c5a6deb48bd9242a	IPv4	HTTP	TCP	80	0.0.0.0/0	< 1 >	Edit inbound rules

sg-092a1f714a380c72d - APP-SG

Details			
Security group name APP-SG	Security group ID sg-092a1f714a380c72d	Description Allow traffic from ALB	VPC ID vpc-0bc7115f56e0762db
Owner 891377334911	Inbound rules count 1 Permission entry	Outbound rules count 1 Permission entry	

[Inbound rules](#) | [Outbound rules](#) | [Sharing](#) | [VPC associations](#) | [Tags](#)

Inbound rules (1)

Search									
<input type="checkbox"/>	Name	Security group rule ID	IP version	Type	Protocol	Port range	Source		Description
<input type="checkbox"/>	-	sgr-01c8691c54e95a0ae	-	Custom TCP	TCP	5051	sg-0bbfe85bd16b6cfef ...	< 1 >	Edit inbound rules

sg-0d754da10960cee5c - aws-cloud9-Hostel-Builder-c9-f5068dd389354ce4a1dddc6806e99118-InstanceSecurityGroup-q6peG5oxlcNm

Details			
Security group name aws-cloud9-Hostel-Builder-c9-f5068dd389354ce4a1dddc6806e99118-InstanceSecurityGroup-q6peG5oxlcNm	Security group ID sg-0d754da10960cee5c	Description Security group for AWS Cloud9 environment aws-clou...	VPC ID vpc-0bc7115f56e0762db
Owner 891377334911	Inbound rules count 2 Permission entries	Outbound rules count 1 Permission entry	

[Inbound rules](#) | [Outbound rules](#) | [Sharing](#) | [VPC associations](#) | [Tags](#)

Inbound rules (2)

Search									
<input type="checkbox"/>	Name	Security group rule ID	IP version	Type	Protocol	Port range	Source		Description
<input type="checkbox"/>	-	sgr-0a45e5386022d91fc	IPv4	SSH	TCP	22	35.172.155.192/27	-	
<input type="checkbox"/>	-	sgr-033ec528cea06f19	IPv4	SSH	TCP	22	35.172.155.96/27	-	

sg-0aceaf7ad48b3855b - DB-SG

Details			
Security group name DB-SG	Security group ID sg-0aceaf7ad48b3855b	Description Allow traffic from app to DB	VPC ID vpc-0bc7115f56e0762db
Owner 891377334911	Inbound rules count 2 Permission entries	Outbound rules count 1 Permission entry	

[Inbound rules](#) | [Outbound rules](#) | [Sharing](#) | [VPC associations](#) | [Tags](#)

Inbound rules (2)

Search									
<input type="checkbox"/>	Name	Security group rule ID	IP version	Type	Protocol	Port range	Source		Description
<input type="checkbox"/>	-	sgr-048058fb4525a1f05	-	MySQL/Aurora	TCP	3306	sg-0d754da10960cee5c...	-	
<input type="checkbox"/>	-	sgr-0c2b0ba630f4295ab	-	MySQL/Aurora	TCP	3306	sg-092a1f714a380c72d...	-	

Used services

S3

The screenshot shows the AWS S3 console under the 'General purpose buckets' tab. It displays one bucket named 'hostel-images-51-75' located in 'US East (N. Virginia)' (us-east-1) with a creation date of December 14, 2025, at 11:10:33 (UTC+05:00). The bucket is currently empty. The interface includes a search bar, sorting options by Name, AWS Region, and Creation date, and buttons for creating a new bucket or deleting the existing one.

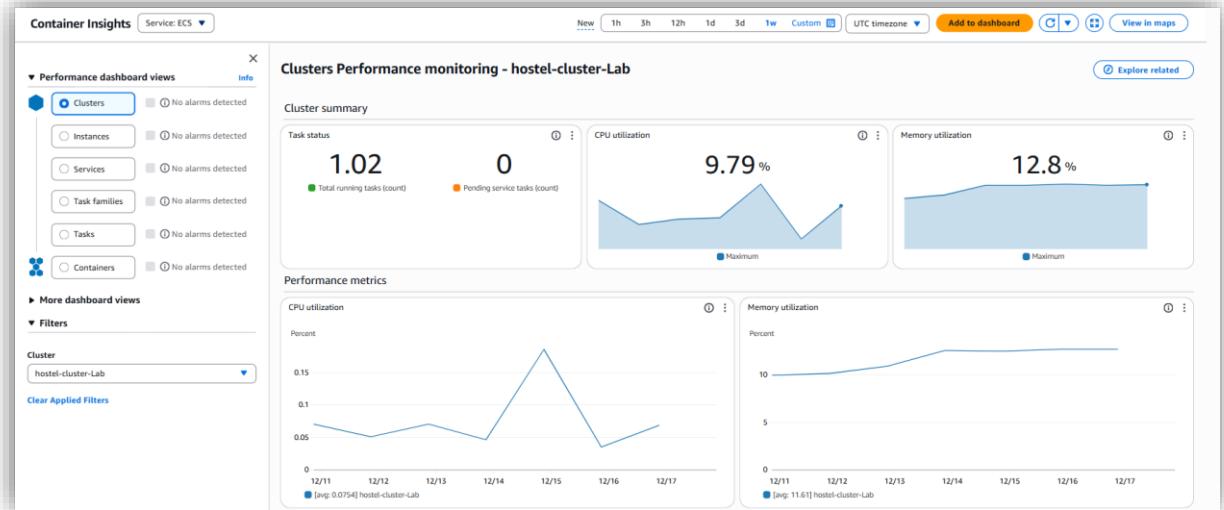
RDS

The screenshot shows the AWS RDS console for a DB instance named 'hostel-database'. The 'Connectivity & security' tab is selected. Key details include:

- Summary:** DB identifier: 'hostel-database', Status: Available, Role: Instance, Engine: MySQL Community, Region & AZ: us-east-1a.
- Connectivity & security:** Endpoint: 'hostel-database.ck66ac60mul.us-east-1.rds.amazonaws.com', Port: 3306. Networking: Availability Zone: us-east-1a, VPC: 'Hostel-VPC-vpc (vpc-0bc7115f56e0762db)', Subnet group: 'default-vpc-0bc7115f56e0762db'. Subnets: 'subnet-0f506e39f264fba8d', 'subnet-0bca727bcac55d3', 'subnet-0f70eb0fd2d2d4cc6', 'subnet-09a7c4a85b66e6a07'. Network type: IPv4.
- Security:** VPC security groups: 'DB-SG (sg-0aarea7fa048b3855b)' (Active). Publicly accessible: No. Certificate authority: 'rds-ca-rsa2048-g1'. Certificate authority date: May 26, 2051, 04:34 (UTC+05:00). DB instance certificate expiration date: December 11, 2026, 17:54 (UTC+05:00).

Deployment process

ECS-CLUSTER



Amazon Elastic Container Service > Clusters > **hostel-cluster-Lab** > Services > **hostel-service** > Configuration

hostel-service Info

Last updated December 18, 2025, 21:19 (UTC+0500)

Status Active

Tasks (1 Desired) 0 Pending | 1 Running

Task definition: revision [hostel-task-lab:4](#)

Deployment status [Success](#)

Health and metrics **Tasks** **Logs** **Deployments** **Events** **Configuration and networking** **Service auto scaling** **Event history** **Tags**

Service configuration Info

Service ARN arn:aws:ecr:us-east-1:891377334911:service/hostel-cluster-Lab/hostel-service

Created by arn:aws:iam:891377334911:role/vocabs

Propagate tags from None

Platform version 1.4.0

Availability Zone rebalancing Turned on

Task definition: revision [hostel-task-lab:4](#)

Launch type Fargate

Platform family Linux

CloudFormation stack ECS-Console-V2-Service-hostel-service-hostel-cluster-Lab-696be875

Scheduling strategy REPLICAS

Amazon ECS managed tags Turned on

ECS Exec [Info](#) Turned off

Fargate ephemeral storage

Network configuration

VPC vpc-0bc7115f56e0762db

Subnets [List](#)

- subnet-0f506e319f264ff08d
- subnet-00a7ca48506e6fa07

Security groups [List](#)

- sg-092a1f714a380c72d

Auto-assign public IP Turned on

Service role [ARN](#) [ServiceRoleForECS](#)

Health check grace period 0 seconds

DNS names

- hostel-alb-569684094.us-east-1.elb.amazonaws.com | [Open address](#)

ECR- REPOSITORY

hostel-app

Summary **Images** **Repository tags**

Repository details

Repository name hostel-app

Created at December 11, 2025, 18:16:08 (UTC+05)

Repository ARN arn:aws:ecr:us-east-1:891377334911:repository/hostel-app

Repository URI 891377334911.dkr.ecr.us-east-1.amazonaws.com/hostel-app

Tag mutability Mutable

Encryption type AES-256

Scan frequency Manual

Tag mutability exclusions -

hostel-app

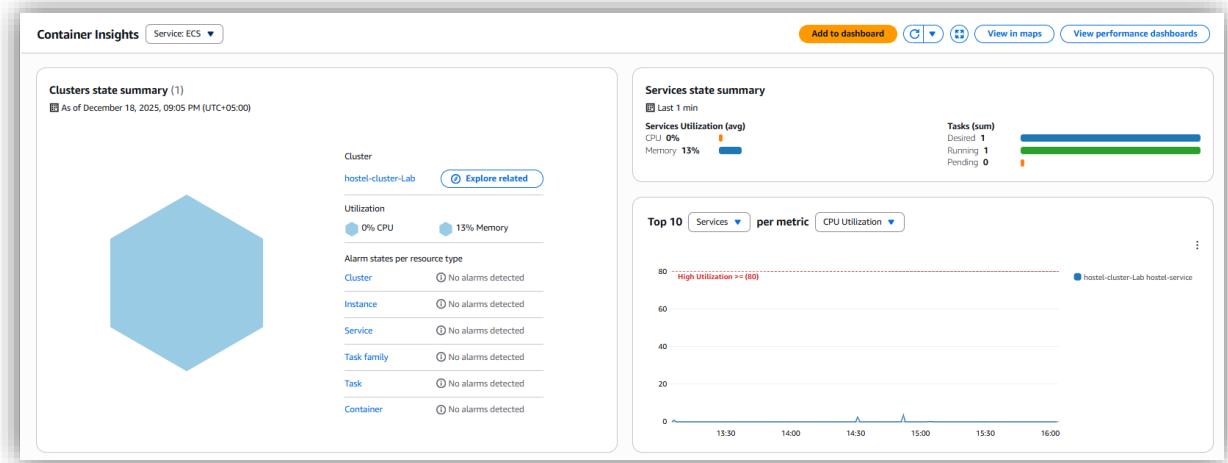
Summary **Images** **Repository tags**

Images (54) Info

Filter active images

<input type="checkbox"/> Image tags	Type	Created at	Image size	Image digest	Last pulled at
<input type="checkbox"/> latest	Image	December 18, 2025, 19:50:45 (UTC+05)	509.56	sha256:b735ccc237abfcfc62...	December 18, 2025, 19:51:31 (UTC+05)
<input type="checkbox"/> -	Image	December 18, 2025, 19:29:34 (UTC+05)	509.56	sha256:fd943f2149add447a...	December 18, 2025, 19:30:59 (UTC+05)
<input type="checkbox"/> -	Image	December 18, 2025, 16:58:46 (UTC+05)	507.30	sha256:ee10a9fe3b5b165c6...	December 18, 2025, 16:59:40 (UTC+05)
<input type="checkbox"/> -	Image	December 18, 2025, 16:51:59 (UTC+05)	507.30	sha256:17dd68b43e5d0c8e...	December 18, 2025, 16:52:46 (UTC+05)
<input type="checkbox"/> -	Image	December 18, 2025, 16:45:59 (UTC+05)	507.30	sha256:546e6d04e63f784b...	December 18, 2025, 16:46:53 (UTC+05)
<input type="checkbox"/> -	Image	December 18, 2025, 16:21:57 (UTC+05)	507.30	sha256:14ced8de1b84a3f4d...	December 18, 2025, 16:22:53 (UTC+05)
<input type="checkbox"/> -	Image	December 16, 2025, 03:48:24 (UTC+05)	507.30	sha256:b5914d4af3e0ccfa...	December 16, 2025, 03:49:17 (UTC+05)
<input type="checkbox"/> -	Image	December 16, 2025, 03:32:54 (UTC+05)	507.41	sha256:90a58cba26b9ca46e...	December 16, 2025, 03:33:43 (UTC+05)
<input type="checkbox"/> -	Image	December 16, 2025, 03:29:08 (UTC+05)	507.41	sha256:c867a8b9da50c2c7a...	December 16, 2025, 03:29:54 (UTC+05)
<input type="checkbox"/> -	Image	December 16, 2025, 03:23:47 (UTC+05)	507.41	sha256:ce7d4a78b2f7a0c08...	December 16, 2025, 03:24:40 (UTC+05)

Testing workflow



The screenshot shows a VS Code interface with multiple tabs open: requirements.txt, Dockerfile, and app.py. The terminal tab displays the output of a Docker build command:

```
PS C:\Users\hh\Desktop\HMS> docker build -t hostel-app .
[+] Building 204.1s (12/12) FINISHED
   => [internal] load build definition from Dockerfile
   => [internal] load metadata for docker.io/library/python:3.12-slim
   => [auth] library/python:pull token for registry-1.docker.io
   => [internal] load .dockerignore
   => [internal] transfer context: 1408
   => [1/6] FROM docker.io/library/python:3.12-slim@sha256:590cad70271b6c1795c6a11fb5c1 8.6s
   => [2/6] resolve docker.io/library/python:3.12-slim@sha256:590cad70271b6c1795c6a11fb5c1 0.1s
   => sha256:1f384a3df5003cc3a739008d6e3c2b2afc752887e9ce09757747c0bbb6e 250B / 250B 0.9s
   => sha256:89933f7805059f29cf8b5a9c0b6df0fe9d96c388b99215881bf6 12.11MB / 12.11MB 7.0s
   => sha256:dff024aded812f05863f68d31b4030038e01017329961ea2df37e6a1c 0.8s
   => sha256:89933f7805059f29cf8b5a9c0b6df0fe9d96c388b99215881bf653ed6f1 1.3s
   => sha256:1f384a3df5003cc3a739008d6e3c2b2afc752887e9ce09757747c0bbb6e 0.0s
   => [internal] load build context
   => [internal] transfer context: 4.24kB
   => [2/6] WORKDIR /app
   => [3/6] RUN apt-get update && apt-get install -y build-essential libpq-dev 69.0s
   => [4/6] COPY requirements.txt .
   => [5/6] RUN pip install --no-cache-dir -r requirements.txt 57.2s
   => [6/6] COPY . .
   => exporting to image 0.5s
   => 64.0s
```

The screenshot shows a Visual Studio Code interface with the following details:

- File Explorer:** Shows a project structure for "HMS" containing files like requirements.txt, Dockerfile, app.py, forms.py, models.py, Pipfile, Pipfile.lock, Procfile, Readme.md, and SECURITY.md.
- Terminal:** Displays the output of a Docker build process:

```
=> => exporting attestation manifest sha256:10432ed21c162c906f42a58fb2fd1afcd1dc1359 0.1s
=> => exporting manifest list sha256:8fe00aadb02ec8ad82783f8ddc352fac2d0c0df92cb589a 0.0s
=> => naming to docker.io/library/hostel-app:latest
=> => unpacking to docker.io/library/hostel-app:latest
0.6s
```
- Output:** Shows PS C:\Users\hh\Desktop\HMS> docker tag hostel-app:latest 891377334911.dkr.ecr.us-east-1.amazonaws.com/hostel-app
- Terminal:** Shows PS C:\Users\hh\Desktop\HMS> docker push 891377334911.dkr.ecr.us-east-1.amazonaws.com/hostel-app
- Bottom Status Bar:** Shows Ln 30, Col 1, Spaces: 4, UTF-8, CRLF, Python, Finish Setup, 3.12.10, Go Live, Continue (NE), Prettier.

The screenshot shows a GitHub Actions run summary for the repository "BSSE23051/HOSTEL-APP".

Run Details: The run was triggered via push 1 minute ago by user BSSE23051 pushing to branch main. The status is Success and the total duration was 1m 22s.

Actions Tab: The "Actions" tab is selected, showing a single job named "Update deploy.yml #2" which has completed successfully.

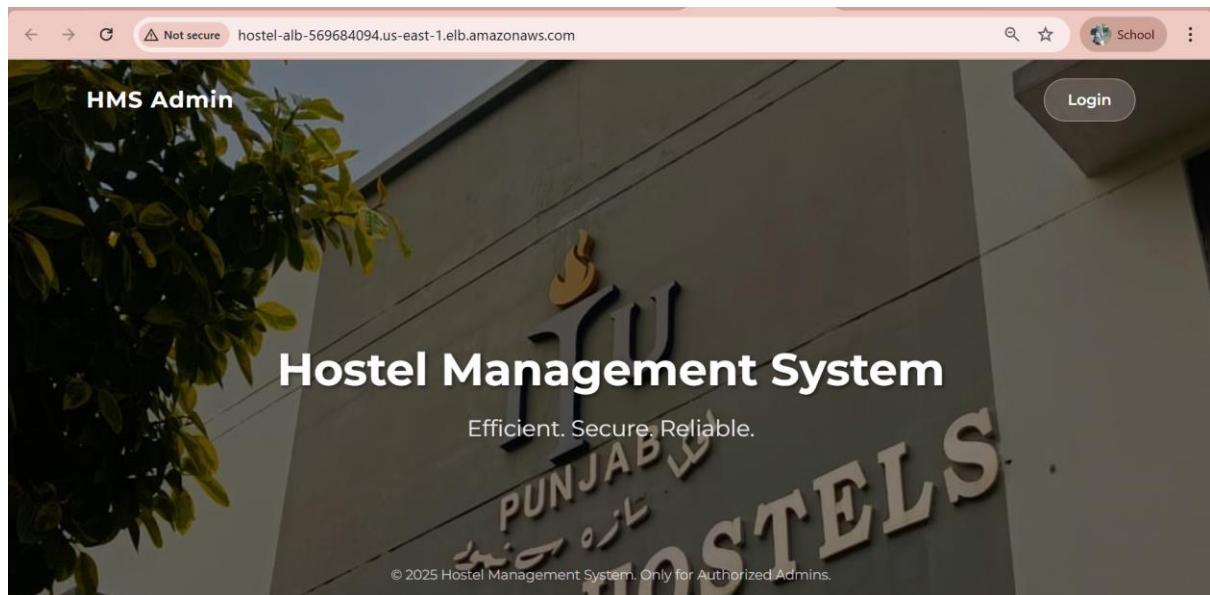
Summary Card: A summary card for the "Deploy" job shows it was triggered by a push and completed 1m 17s ago.

Monitoring / logging (CloudWatch)

Rules on default event bus (9)								
	Name	Status	Type	Event bus	ARN	Description		
	MonitoringRule	Enabled	Standard	default	arn:aws:events:us-east-1:8913773349:rule/MonitoringRule	MonitoringRule		
	resourcefunctionrule	Enabled	Standard	default	arn:aws:events:us-east-1:8913773349:rule/resourceFunctionRule	-		
	voc-bedrock-cw-rule	Enabled	Standard	default	arn:aws:events:us-east-1:8913773349:rule/voc-bedrock-cw-rule	bedrock job state change events		
	voc-bedrockapi-cw-rule	Enabled	Standard	default	arn:aws:events:us-east-1:8913773349:rule/voc-bedrockapi-cw-rule	bedrock api events		
	voc-codebuild-cw-rule	Enabled	Standard	default	arn:aws:events:us-east-1:8913773349:rule/voc-codebuild-cw-rule	codebuild build state change events		
	voc-ec2-cw-rule	Enabled	Standard	default	arn:aws:events:us-east-1:8913773349:rule/voc-ec2-cw-rule	ec2 state change events		
	voc-redshift-cw-rule	Enabled	Standard	default	arn:aws:events:us-east-1:8913773349:rule/voc-redshift-cw-rule	redshift events		
	voc-redshiftapi-cw-rule	Enabled	Standard	default	arn:aws:events:us-east-1:8913773349:rule/voc-redshiftapi-cw-rule	redshift api events		
	voc-redshiftserverlessapi-cw-rule	Enabled	Standard	default	arn:aws:events:us-east-1:8913773349:rule/voc-redshiftserverlessapi-cw-rule	redshift serverless api events		

Log groups (5)								
By default, we only load up to 10,000 log groups.								
Actions ▾								
Filter log groups or try pattern search <input type="text"/> Exact match								
	Log group	Log class	Anomaly d...	Deletion protection	Data protection	Sensitive data count		Retention
	/aws/ecs/containerinsights/hostel-cluster-Lab/perfor...	Standard	Configure	Off	-	-		1 day
	/aws/lambda/RedshiftEventSubscription	Standard	Configure	Off	-	-		Never expire
	/aws/lambda/RedshiftOverwatch	Standard	Configure	Off	-	-		Never expire
	/aws/lambda/RoleCreationFunction	Standard	Configure	Off	-	-		Never expire
	/ecs/hostel-task-lab	Standard	Configure	Off	-	-		Never expire

Final output



List of Figures:

- [Figure 1: High-Level AWS Architecture Diagram \(Multi-AZ Deployment\)](#)
- [Figure 2: Cost and usage](#)
- [Figure 3: AWS Account and Setup](#)
- [Figure 4: IAM Roles and Policy Configuration](#)
- [Figure 5: VPC Resource Map and Network Topology](#)
- [Figure 6: Subnet Configuration Details \(Private and Public\)](#)
- [Figure 7: Security Group Inbound Rules \(ALB, App, and Database\)](#)
- [Figure 9: Amazon S3 Bucket for Media Storage](#)
- [Figure 10: Amazon RDS Instance Connectivity Status](#)
- [Figure 11: ECS Cluster Overview and Service Health](#)
- [Figure 12: Amazon ECR Repository and Image Tags](#)
- [Figure 13: Local Testing Workflow in VS Code](#)
- [Figure 14: Amazon CloudWatch Log Groups and Event Bus](#)
- [Figure 15: Final Hostel Management System Dashboard](#)

List of Tables:

[Table 1: Technology Stack Summary](#)

[Table 2: AWS Network Configuration](#)

References:

Adzic, G., & Chatley, R. (2017). Serverless computing: Economic and architectural impact. *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*, 11(1), 884–889. <https://doi.org/10.1145/3106237.3117767>

Di Francesco, P., Malavolta, I., & Lago, P. (2019). Architecting microservices: A systematic mapping study. *Journal of Systems and Software*, 150(1), 77–97. <https://doi.org/10.1016/j.jss.2019.01.001>

Merkel, D. (2014). Docker: Lightweight Linux containers for consistent development and deployment. *Linux Journal*, 2014(239), 2–2. <https://dl.acm.org/doi/10.5555/2600239.2600241>

Shahin, M., Babar, M. A., & Zhu, L. (2017). Continuous integration, delivery and deployment: A systematic review on approaches, tools, challenges and practices. *IEEE Access*, 5(1), 3909–3943. <https://doi.org/10.1109/ACCESS.2017.2685629>

Varghese, B., & Buyya, R. (2018). Next generation cloud computing: New trends and research directions. *Future Generation Computer Systems*, 79(1), 849–861. <https://doi.org/10.1016/j.future.2017.09.020>