

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
Executive Summary.....	3
Introduction	3
Problem Statement.....	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	6
DevOps & CI/CD Pipeline	6
AWS Architecture Design, Implementation and Screenshots	7
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
Used services	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.

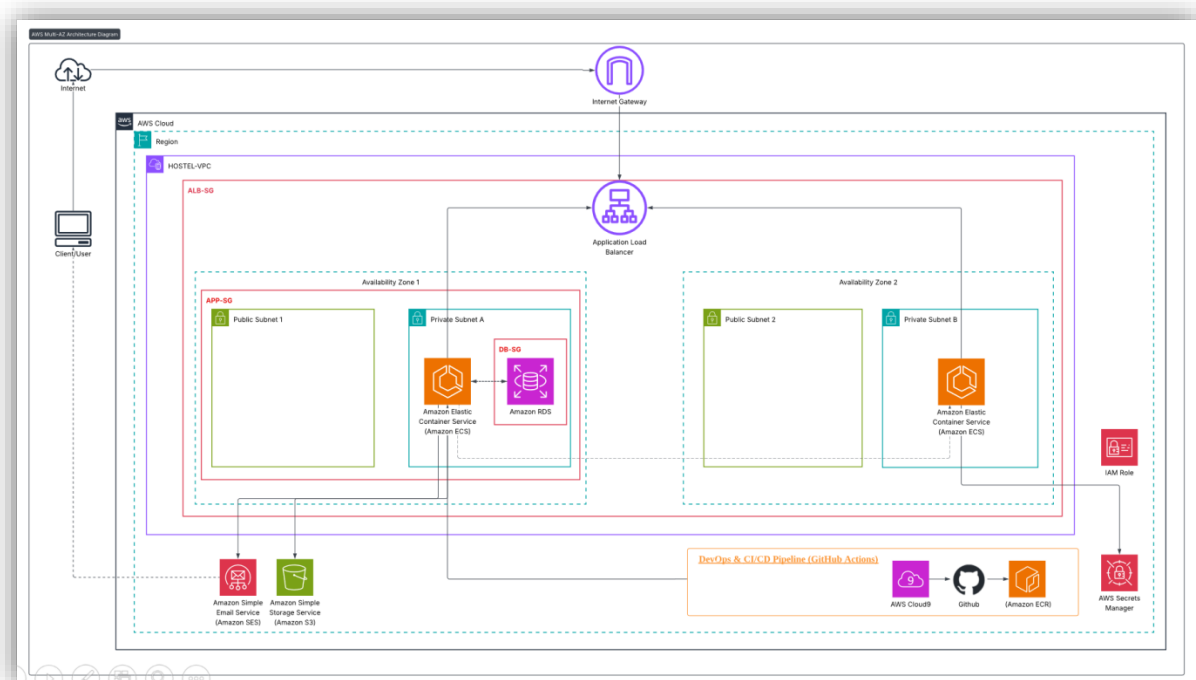
This project seeks to resolve these issues by re-architecting the system into a **Stateless, Serverless, and Multi-AZ (Availability Zone)** solution on AWS.

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

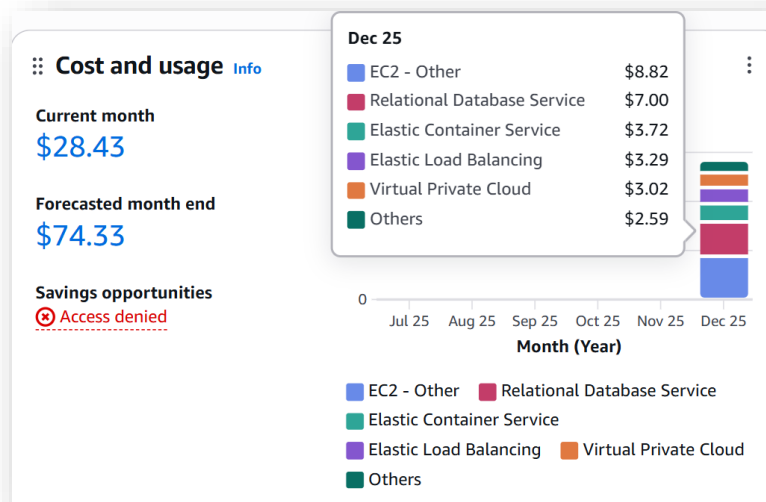
We moved away from manual deployments to a modern GitOps workflow.

- 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

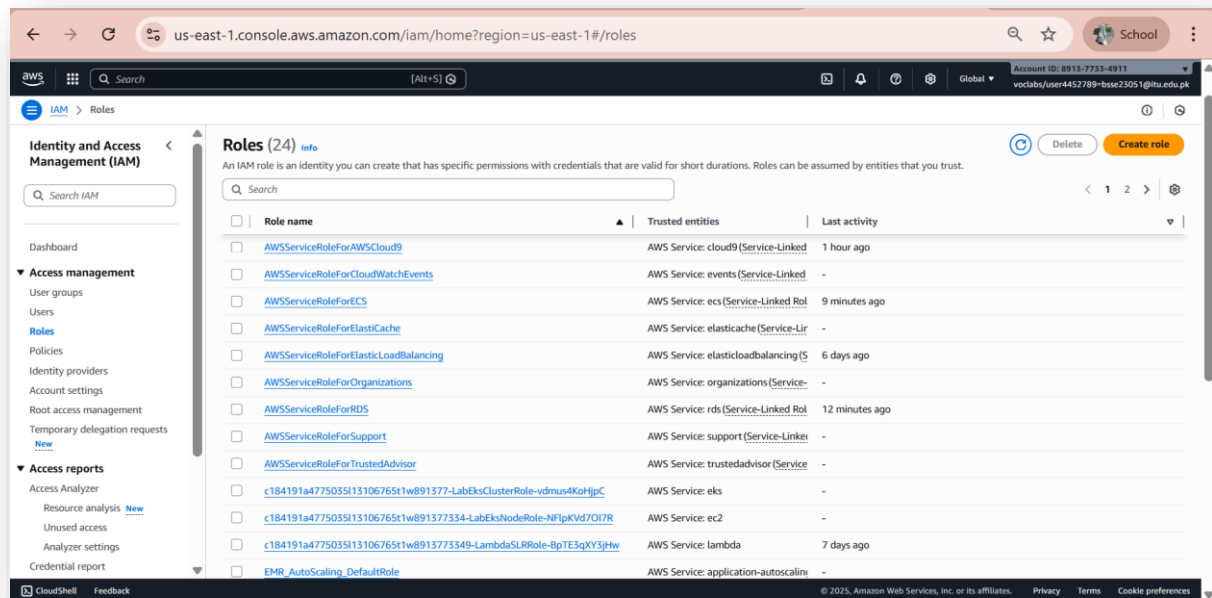
Password
.....

☐ Stay signed in [Forgot Password?](#) [Log In](#)

[Help](#) [Privacy Policy](#) [Cookie Notice](#) [Acceptable Use Policy](#)
[Facebook](#) [X.com](#)

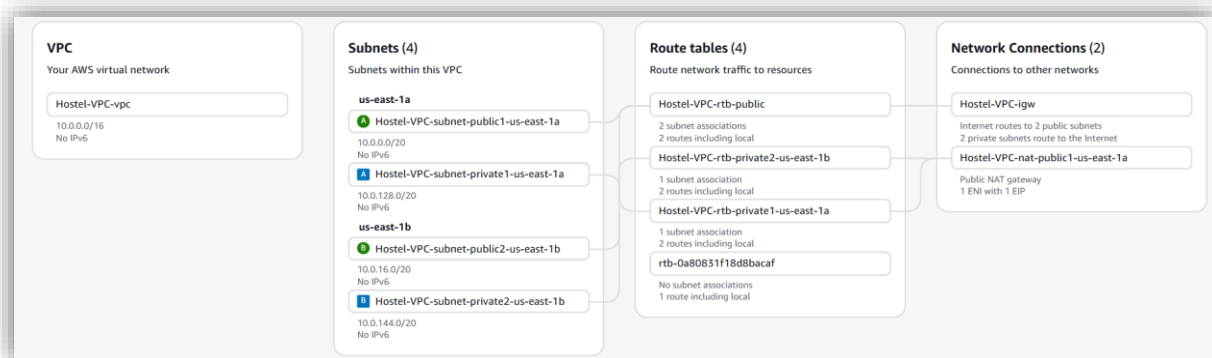
INSTRUCTURE
Meet the Instructure Learning Platform:
[Canvas LMS](#) [Mastery Connect](#) [Elevate Analytics](#) [Impact](#)

IAM roles and policies



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	Subnet ARN arn:aws:ec2:us-east-1:891377354911:subnet/subnet-00a7ca485b6ee6a07	State Available	Block Public Access Off
IPv4 CIDR 10.0.128.0/20	Available IPv4 addresses 4089	IPv6 CIDR -	IPv6 CIDR association ID -
Availability Zone us-east-1a	Network border group us-east-1	VPC vpc-0bc7115f56e0762db Hostel-VPC-vpc	Route table rtb-0724a89338f65cc33 Hostel-VPC-rtb-private1-us-east-1a
Network ACL acl-033c2c44ff43c7c3d	Default subnet No	Auto-assign public IPv4 address No	Auto-assign IPv6 address No
Auto-assign customer-owned IPv4 address No	Customer-owned IPv4 pool -	Outpost ID -	IPv4 CIDR reservations -
IPv6 CIDR reservations -	IPv6-only No	Hostname type IP name	Resource name DNS A record Disabled
Resource name DNS AAAA record Disabled	DNS64 Disabled	Owner 891377354911	

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	Subnet ARN arn:aws:ec2:us-east-1:891377354911:subnet/subnet-0f506e39f264fba8d	State Available	Block Public Access Off
IPv4 CIDR 10.0.144.0/20	Available IPv4 addresses 4091	IPv6 CIDR -	IPv6 CIDR association ID -
Availability Zone us-east-1b	Network border group us-east-1	VPC vpc-0bc7115f56e0762db Hostel-VPC-vpc	Route table rtb-0c854a4dbd281e112 Hostel-VPC-rtb-private2-us-east-1b
Network ACL acl-033c2c44ff43c7c3d	Default subnet No	Auto-assign public IPv4 address No	Auto-assign IPv6 address No
Auto-assign customer-owned IPv4 address No	Customer-owned IPv4 pool -	Outpost ID -	IPv4 CIDR reservations -
IPv6 CIDR reservations -	IPv6-only No	Hostname type IP name	Resource name DNS A record Disabled
Resource name DNS AAAA record Disabled	DNS64 Disabled	Owner 891377354911	

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	Subnet ARN arn:aws:ec2:us-east-1:891377354911:subnet/subnet-0bca727cbace55dd3	State Available	Block Public Access Off
IPv4 CIDR 10.0.0.0/20	Available IPv4 addresses 4088	IPv6 CIDR -	IPv6 CIDR association ID -
Availability Zone us-east-1a	Network border group us-east-1	VPC vpc-0bc7115f56e0762db Hostel-VPC-vpc	Route table rtb-005249f55028dbd10 Hostel-VPC-rtb-public
Network ACL acl-033c2c44ff43c7c3d	Default subnet No	Auto-assign public IPv4 address No	Auto-assign IPv6 address No
Auto-assign customer-owned IPv4 address No	Customer-owned IPv4 pool -	Outpost ID -	IPv4 CIDR reservations -
IPv6 CIDR reservations -	IPv6-only No	Hostname type IP name	Resource name DNS A record Disabled
Resource name DNS AAAA record Disabled	DNS64 Disabled	Owner 891377354911	

subnet-0f70eb0fcd22d4cc6 / Hostel-VPC-subnet-public2-us-east-1b

Details | Flow logs | Route table | Network ACL | CIDR reservations | Sharing | Tags

Details

Subnet ID subnet-0f70eb0fcd22d4cc6	Subnet ARN arn:aws:ec2:us-east-1:891377354911:subnet/subnet-0f70eb0fcd22d4cc6	State Available	Block Public Access Off
IPv4 CIDR 10.0.16.0/20	Available IPv4 addresses 4090	IPv6 CIDR -	IPv6 CIDR association ID -
Availability Zone us-east-1b	Network border group us-east-1	VPC vpc-0bc7115f56e0762db Hostel-VPC-vpc	Route table rtb-005249f55028dbd10 Hostel-VPC-rtb-public
Network ACL acl-033c2c44ff43c7c3d	Default subnet No	Auto-assign public IPv4 address No	Auto-assign IPv6 address No
Auto-assign customer-owned IPv4 address No	Customer-owned IPv4 pool -	Outpost ID -	IPv4 CIDR reservations -
IPv6 CIDR reservations -	IPv6-only No	Hostname type IP name	Resource name DNS A record Disabled
Resource name DNS AAAA record Disabled	DNS64 Disabled	Owner 891377354911	

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
<input type="checkbox"/>	-	sgr-0c5a6deb48bd9242a	IPv4	HTTP	TCP	80	0.0.0.0/0

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...	-

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
d9-Hostel-Builder-c9-f5068dd389354ce4a1dddc6806e99
118

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-033eec528cea06f19	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

General purpose buckets

All AWS Regions

Directory buckets

General purpose buckets (1) Info

Copy ARN

Empty

Delete

Create bucket

Buckets are containers for data stored in S3.

Find buckets by name

< 1 >

Name	AWS Region	Creation date
<div></div> hostel-images-51-75	US East (N. Virginia) us-east-1	December 14, 2025, 11:10:33 (UTC+05:00)

RDS

hostel-database

[Modify](#)

Summary

DB identifier
hostel-database

CPU

3.27%

Status
 Available

Class
db.t3.micro

Role
Instance

Current activity

1 Connections

Engine
MySQL Community

Region & AZ
us-east-1a

Recommendations

2 Informational

Connectivity & security

Monitoring

Logs & events

Configuration

Zero-ETL integrations

Maintenance & backups

Data migrations

Tags

Recommendations

Connectivity & security

Endpoint & port

Endpoint
 [hostel-database.cik66ac6omul.us-east-1.rds.amazonaws.com](#)

Port
3306

Networking

Availability Zone
us-east-1a

VPC
[Hostel-VPC-vpc-0bc7115f56e0762db](#)

Subnet group
[default-vpc-0bc7115f56e0762db](#)

Subnets
[subnet-0f506e19f264fba8d](#)
[subnet-0bca727cbace55dd3](#)
[subnet-0f70eb0fcd22d4cc6](#)
[subnet-00a7ca485b6ee6a07](#)

Network type
IPv4

Security

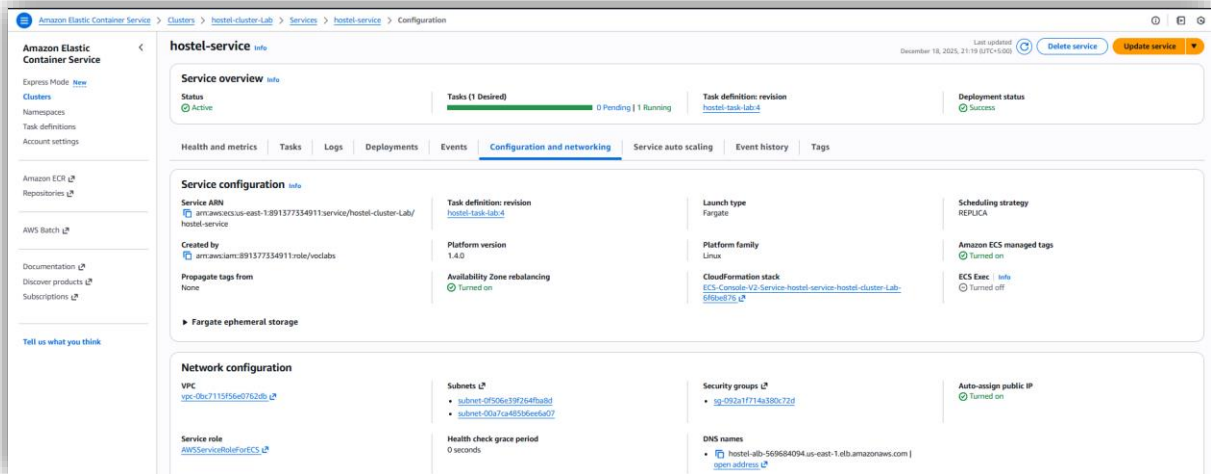
VPC security groups
[DB-SG \(sg-0aceaf7ad48b3855b\)](#)
 Active

Publicly accessible
No

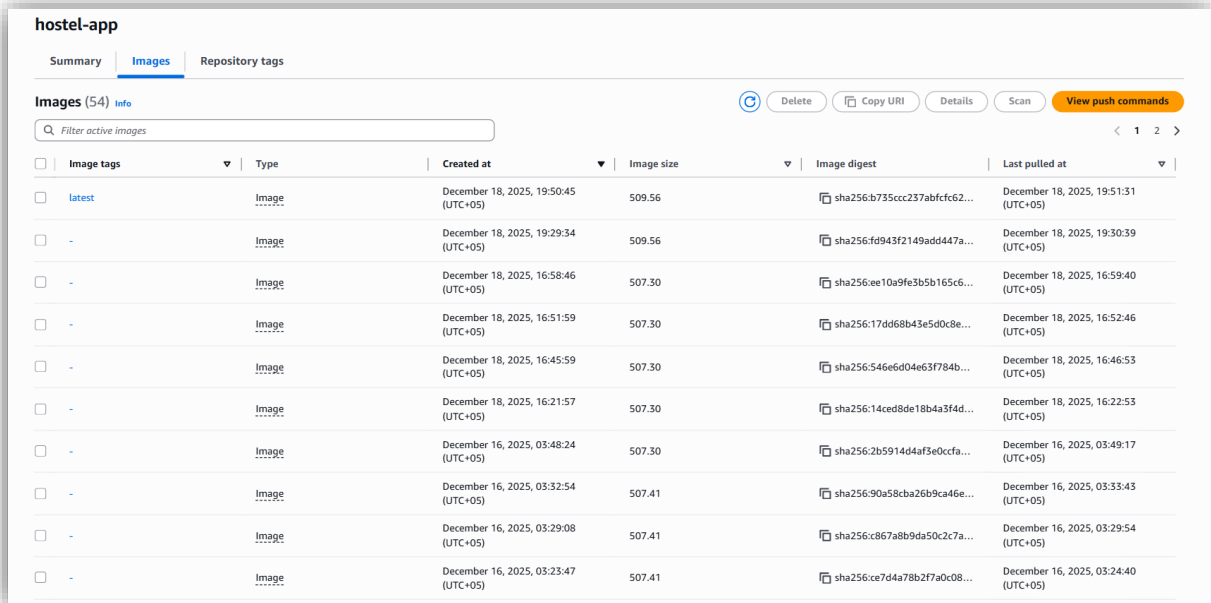
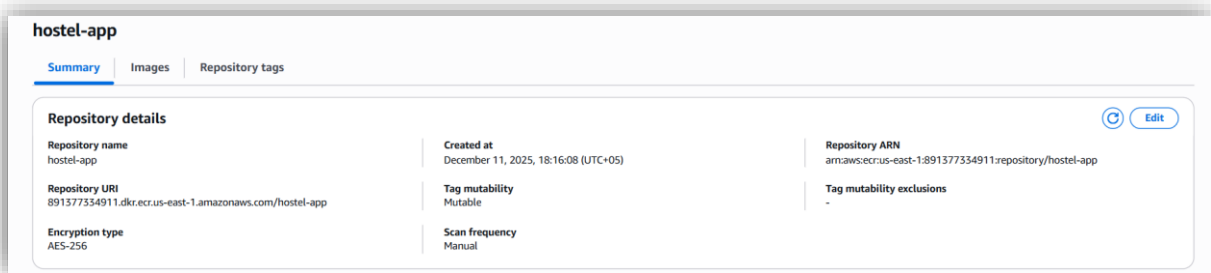
Certificate authority
[Info](#)
[rds-ca-rsa2048-g1](#)

Certificate authority date
May 26, 2061, 04:34 (UTC+05:00)

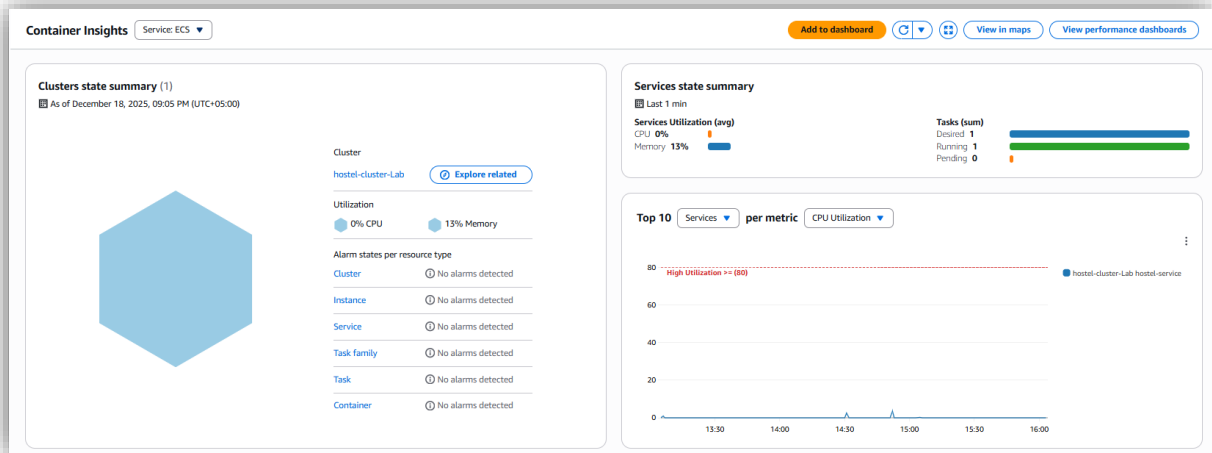
DB instance certificate expiration date
December 11, 2026, 17:54 (UTC+05:00)



ECR- REPOSITORY



Testing workflow

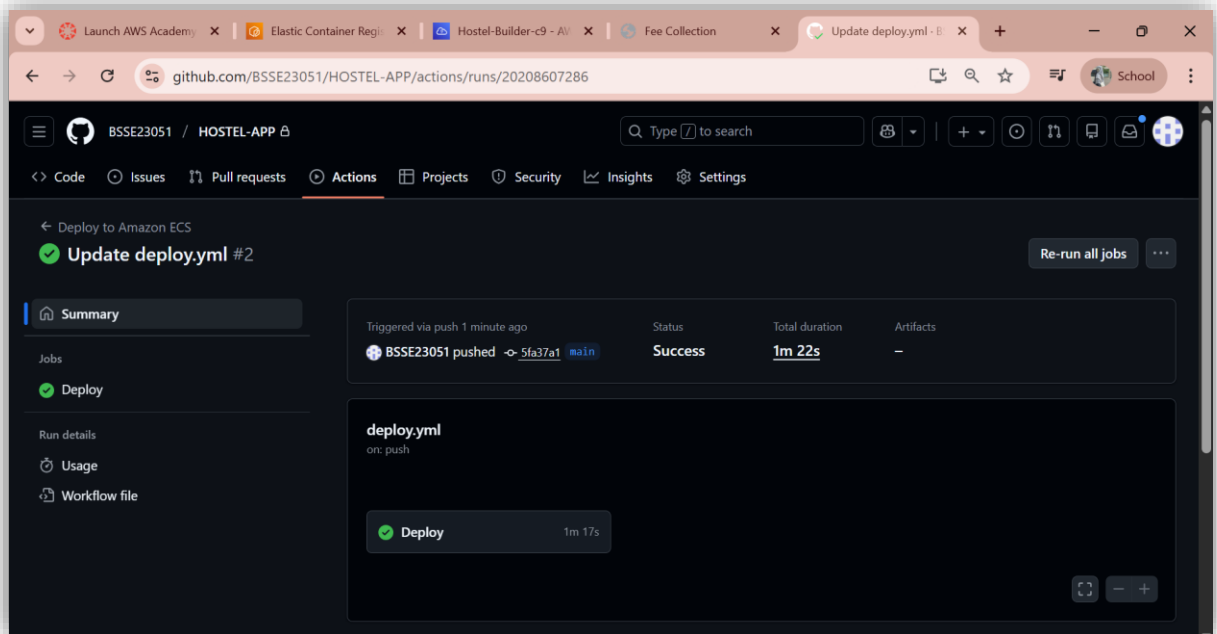


The screenshot shows a Windows terminal window with the command prompt running the Docker build process for a Python application. The output shows the build steps, including downloading the Dockerfile, pulling the Docker image, and building the container image.

```
PS C:\Users\hh\Desktop\HMS> docker build -t hostel-app .
[+] Building 204.1s (12/12) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 800B
=> [internal] load metadata for docker.io/library/python:3.12-slim
=> [auth] library/python:pull token for registry-1.docker.io
=> [internal] load .dockerignore
=> => transferring context: 140B
=> [1/6] FROM docker.io/library/python:3.12-slim@sha256:590cad70271b6c1795c6a11fb5c1
=> => resolve docker.io/library/python:3.12-slim@sha256:590cad70271b6c1795c6a11fb5c1
=> sha256:1f384a3df5003cc3a739008d6e3c2b2afc752887e9ce09757747c0bbb6e 250B / 250B
=> sha256:89933f78055059f29cf8b5a9c0b6df0fe9d96c388b99215881bf6 12.11MB / 12.11MB
=> sha256:dff024aded812f05863f68d31b04030038e01017329961ea2d5f37 1.29MB / 1.29MB
=> extracting sha256:dff024aded812f05863f68d31b04030038e01017329961ea2d5f37e6a1c 0.8s
=> extracting sha256:89933f78055059f29cf8b5a9c0b6df0fe9d96c388b99215881bf653ed6f1 1.3s
=> extracting sha256:1f384a3df5003cc3a739008d6e3c2b2afc752887e9ce09757747c0bbb6e6 0.0s
=> [internal] load build context
=> => transferring 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 . 0.2s
=> [5/6] RUN pip install --no-cache-dir -r requirements.txt 57.2s
=> [6/6] COPY . . 0.5s
=> exporting to image 64.0s
```

The screenshot shows a Visual Studio Code editor window with the file explorer on the left displaying a project named 'HMS'. The file explorer lists files like 'templates', 'update_students_exc...', 'view_all_fee_records...', '.dockerignore', 'gitattributes', 'gitignore', 'app.py', 'forms.py', 'LICENSE', 'models.py', 'Pipfile', 'Pipfile.lock', 'Procfile', 'Readme.md', 'requirements.txt', 'runtime.txt', 'Screenshot from 2025...', and 'SECURITY.md'. The main editor area shows a file named 'app.py' with a function 'def create_tables():'. Below the editor, the 'TERMINAL' tab is active, displaying the output of a Docker build and push command. The output shows the export of the Docker image 'hostel-app:latest' to the Amazon ECR registry and its subsequent push. The build details show that the image is based on 'python:3.12.10-slim' and contains the application code. The push command is successful, and the image is now available in the repository.

```
View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/jhjiivr71lucvxtdegnfwqbro
PS C:\Users\hh\Desktop\HMS> docker tag hostel-app:latest 891377334911.dkr.ecr.us-east-1.amazonaws.com/hostel
-app
PS C:\Users\hh\Desktop\HMS> docker push 891377334911.dkr.ecr.us-east-1.amazonaws.com/hostel-app
Using default tag: latest
The push refers to repository [891377334911.dkr.ecr.us-east-1.amazonaws.com/hostel-app]
bfc218516263: Layer already exists
89933f780550: Layer already exists
1f384a3df500: Layer already exists
3465ff059709: Layer already exists
46133d9ae53b: Pushed
1733a4cd5954: Layer already exists
dfff024aded8: Layer already exists
a995e770a6c7: Layer already exists
fbbaa986c62: Layer already exists
latest: digest: sha256:8fe00aadb02ec8ad82783f8ddc352fac2d0c0df92cb589a0df22f8f3f1baa09a size: 856
PS C:\Users\hh\Desktop\HMS>
```



Monitoring / logging (CloudWatch)

Rules on default event bus (9)

Find rules

Any status

Delete

Enable

Edit

CloudFormation Template

Create rule

<input type="checkbox"/>	Name	Status	Type	Event bus	ARN	Description
<input type="checkbox"/>	MonitoringRule	Enabled	Standard	default	arn:aws:events:us-east-1:8913773349:rule/MonitoringRule	MonitoringRule
<input type="checkbox"/>	resourceFunctionRule	Enabled	Standard	default	arn:aws:events:us-east-1:8913773349:rule/resourceFunctionRule	-
<input type="checkbox"/>	voc-bedrock-cw-rule	Enabled	Standard	default	arn:aws:events:us-east-1:8913773349:rule/voc-bedrock-cw-rule	bedrock job state change events
<input type="checkbox"/>	voc-bedrockapi-cw-rule	Enabled	Standard	default	arn:aws:events:us-east-1:8913773349:rule/voc-bedrockapi-cw-rule	bedrock api events
<input type="checkbox"/>	voc-codebuild-cw-rule	Enabled	Standard	default	arn:aws:events:us-east-1:8913773349:rule/voc-codebuild-cw-rule	codebuild build state change events
<input type="checkbox"/>	voc-ec2-cw-rule	Enabled	Standard	default	arn:aws:events:us-east-1:8913773349:rule/voc-ec2-cw-rule	ec2 state change events
<input type="checkbox"/>	voc-redshift-cw-rule	Enabled	Standard	default	arn:aws:events:us-east-1:8913773349:rule/voc-redshift-cw-rule	redshift events
<input type="checkbox"/>	voc-redshiftapi-cw-rule	Enabled	Standard	default	arn:aws:events:us-east-1:8913773349:rule/voc-redshiftapi-cw-rule	redshift api events
<input type="checkbox"/>	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)

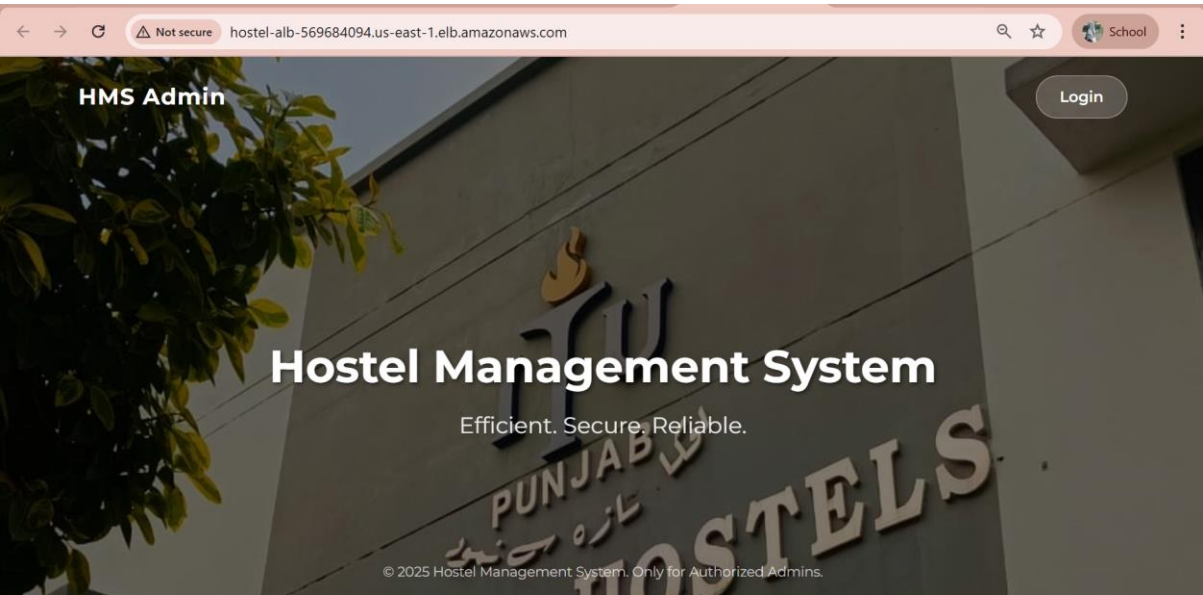
By default, we only load up to 10,000 log groups.

Filter log groups or try pattern search

Exact match

<input type="checkbox"/>	Log group	Log class	Anomaly d...	Deletion protection	Data protection	Sensitive data count	Retention
<input type="checkbox"/>	/aws/ec2/containerinsights/hostel-cluster-Lab/perfor...	Standard	Configure	Off	-	-	1 day
<input type="checkbox"/>	/aws/lambda/RedshiftEventSubscription	Standard	Configure	Off	-	-	Never expire
<input type="checkbox"/>	/aws/lambda/RedshiftOverwatch	Standard	Configure	Off	-	-	Never expire
<input type="checkbox"/>	/aws/lambda/RoleCreationFunction	Standard	Configure	Off	-	-	Never expire
<input type="checkbox"/>	/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>