Cloud-Based Web Application Deployment on AWS Infrastructure

Salman Ahmad (21i-0469) Syed Haider Naqvi (20i-0816)

May 11, 2025

Contents

1 Introduction

This document details the implementation of a cloud-based web application deployed on AWS infrastructure. The application follows modern cloud-native architecture principles and consists of two separate components:

- A backend service that performs CRUD operations, handles user authentication, manages a PostgreSQL database, and supports secure file and image uploads to S3.
- A frontend client that consumes the backend via REST APIs, deployed on Elastic Beanstalk.

The project demonstrates the use of various AWS services including EC2, RDS, S3, Elastic Beanstalk, VPC, and IAM to create a secure and scalable application architecture.

1.1 Project Overview

- Application Type: Task Management System
- Backend Technologies: Node.js, Express, Sequelize ORM
- Frontend Technologies: React, Bootstrap, Axios
- Database: PostgreSQL on Amazon RDS
- Storage: Amazon S3 for file uploads
- Deployment: Backend on EC2 with Docker, Frontend on Elastic Beanstalk

1.2 Live Demo URLs

- Frontend: `http://taskapp-frontend-prod-env.eba-vupzncxu.ap-southeast-1.elasticbeanstalk.com/`
- Backend API: `http://122.248.202.13:5000/api`

1.3 GitHub Repository

- Repository Link: `https://github.com/SoberSalman/TaskManagerAWS`

2 Architecture Diagram

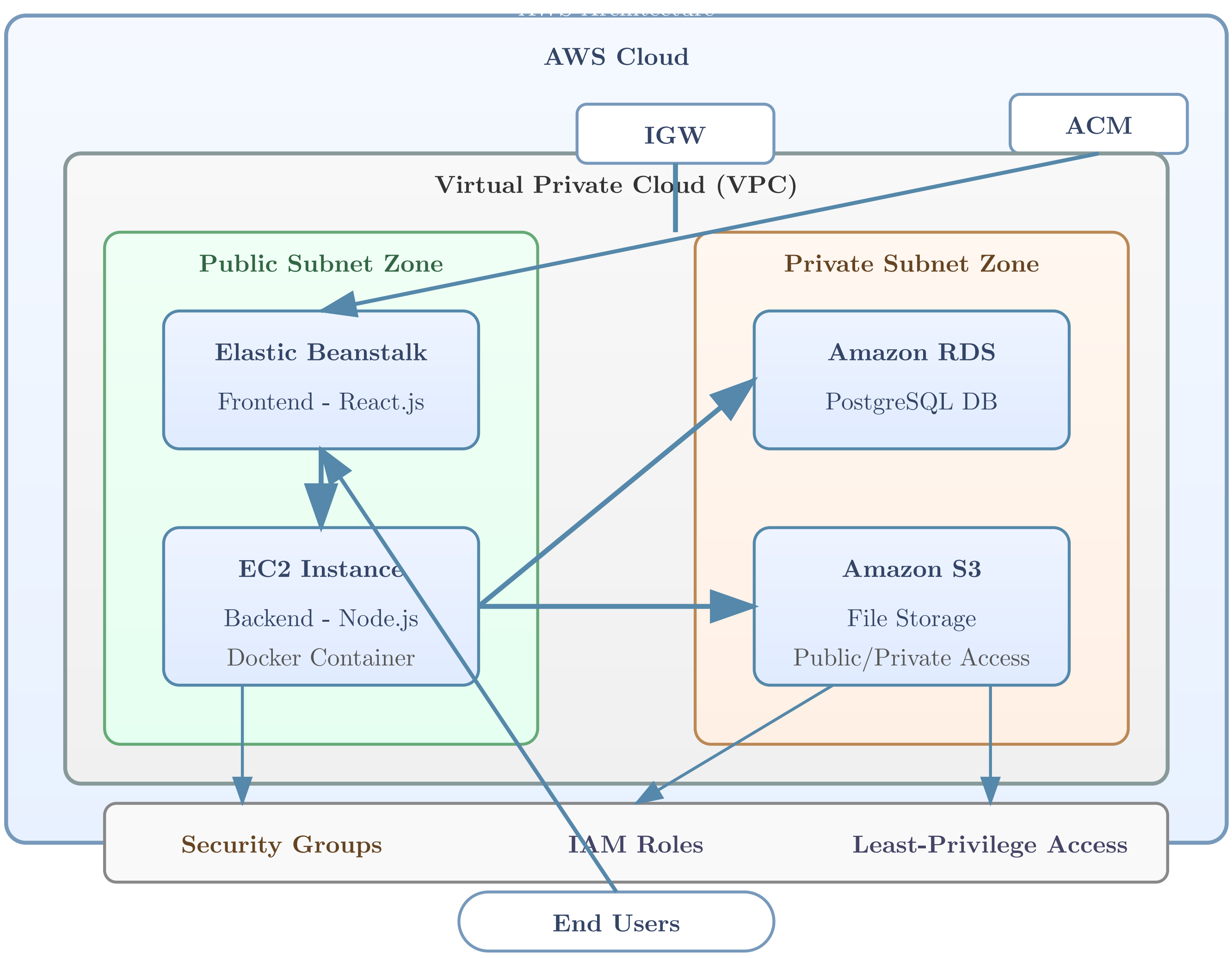


Figure 1: Architecture Diagram

2.1 Architecture Components

- **VPC Configuration:** Custom VPC with public and private subnets across multiple Availability Zones for enhanced security and availability.
- **Frontend:** React application deployed on Elastic Beanstalk, providing a scalable and managed environment.
- **Backend:** Node.js/Express API deployed on an EC2 instance within a public subnet, using Docker for containerization.
- **Database:** PostgreSQL database hosted on Amazon RDS in a private subnet for enhanced security.
- **Storage:** Amazon S3 bucket configured with appropriate permissions for storing user file uploads.
- **Security:** IAM roles and policies implementing the principle of least privilege, security groups restricting traffic, and HTTPS for secure communication.

2.2 Network Flow

1. Users access the frontend application hosted on Elastic Beanstalk.

2. The frontend makes API requests to the backend service running on EC2.
3. The backend authenticates requests, processes data, interacts with the database, and manages file operations with S3.
4. Backend responses are returned to the frontend for display to the user.

3 Implementation Details

3.1 VPC Configuration

- Custom VPC (CIDR: 172.31.0.0/16)
- Public Subnets: 172.31.1.0/24, 172.31.2.0/24
- Internet Gateway for external connectivity
- Route Tables configured for proper traffic routing

3.2 Backend Implementation

- Node.js/Express RESTful API
- JWT-based authentication system
- Sequelize ORM for database interactions
- Integration with AWS S3 for file storage
- Docker containerization for consistent deployment

3.3 Frontend Implementation

- React single-page application
- Bootstrap for responsive design
- Axios for API communication
- React Router for client-side routing
- Context API for state management

3.4 Database Schema

- Users Table: Stores user credentials and profile information
- Tasks Table: Stores task information with foreign key relationship to users

3.5 AWS S3 Integration

- Bucket configured with appropriate CORS settings
- Secure file upload mechanism
- Direct file access URLs for authenticated users

4 Security Measures

4.1 IAM Roles and Policies

- **EC2 Instance Role (TaskApp-EC2-Role):** Provides permissions for the EC2 instance to access S3 and RDS.
- **Elastic Beanstalk Role (TaskApp-EB-Role):** Grants necessary permissions for Elastic Beanstalk to manage resources.
- Principle of least privilege implemented across all roles

4.2 Security Groups

- **EC2 Security Group:**
  - Inbound: Allow HTTP (80), HTTPS (443), and SSH (22) from specific IPs
  - Outbound: Allow all traffic
- **RDS Security Group:**
  - Inbound: Allow PostgreSQL (5432) traffic only from EC2 security group
  - Outbound: Allow all traffic
- **Elastic Beanstalk Security Group:**
  - Inbound: Allow HTTP (80) and HTTPS (443) from anywhere
  - Outbound: Allow all traffic

4.3 Data Protection

- Password hashing using bcrypt
- JWT-based authentication with expiration
- HTTPS for secure data transmission
- Database in private subnet, inaccessible from the internet

5 Deployment Guide

5.1 Prerequisites

- AWS Account with access to required services
- AWS CLI configured with appropriate credentials
- Node.js and npm installed locally
- Docker installed locally (for backend containerization)

5.2 VPC and Network Setup

1. Create a VPC with CIDR block 172.31.0.0/16
2. Create public subnets in different Availability Zones
3. Create an Internet Gateway and attach to VPC
4. Configure route tables to allow internet access

5.3 Backend Deployment

5.3.1 Setting up RDS Database

1. Create a DB subnet group with private subnets
2. Launch PostgreSQL RDS instance with appropriate settings
3. Configure security group to allow access only from backend

5.3.2 Setting up S3 Bucket

1. Create an S3 bucket for file storage
2. Configure CORS settings to allow cross-origin requests
3. Set up bucket policy for appropriate access control

5.3.3 Deploying Backend to EC2

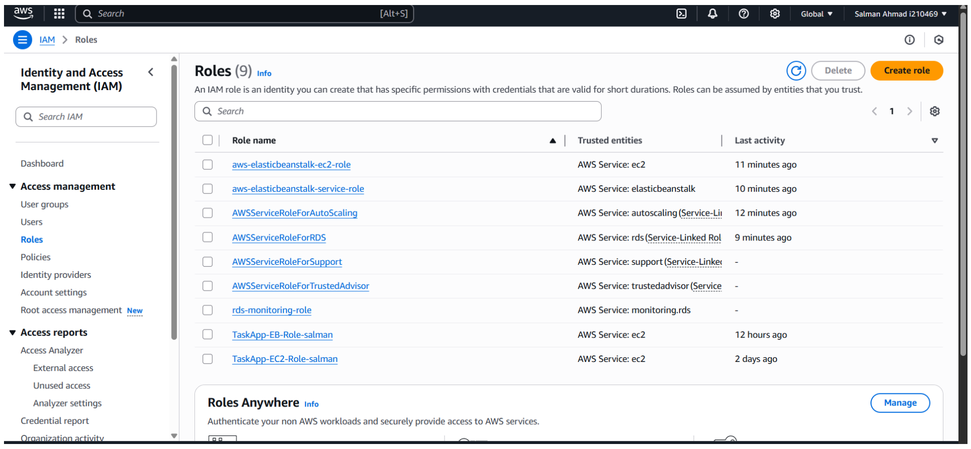
1. Launch EC2 instance in public subnet
2. Attach IAM role with necessary permissions
3. Connect to instance using SSH
4. Install Docker and other dependencies
5. Clone repository and navigate to backend directory
6. Configure environment variables for database and S3 access
7. Build and run Docker container

5.4 Frontend Deployment

1. Navigate to frontend directory locally
2. Configure API endpoint in environment variables
3. Build React application
4. Create necessary configuration files (.ebextensions, Procfile)
5. Create deployment package (ZIP file)
6. Create Elastic Beanstalk application and environment
7. Upload and deploy application code
8. Configure environment variables if needed

6 Screenshots and Evidence

6.1 IAM Role Creation and Attachment



Roles (9) Info

An IAM role is an identity you can create that has specific permissions with credentials that are valid for short durations. Roles can be assumed by entities that you trust.

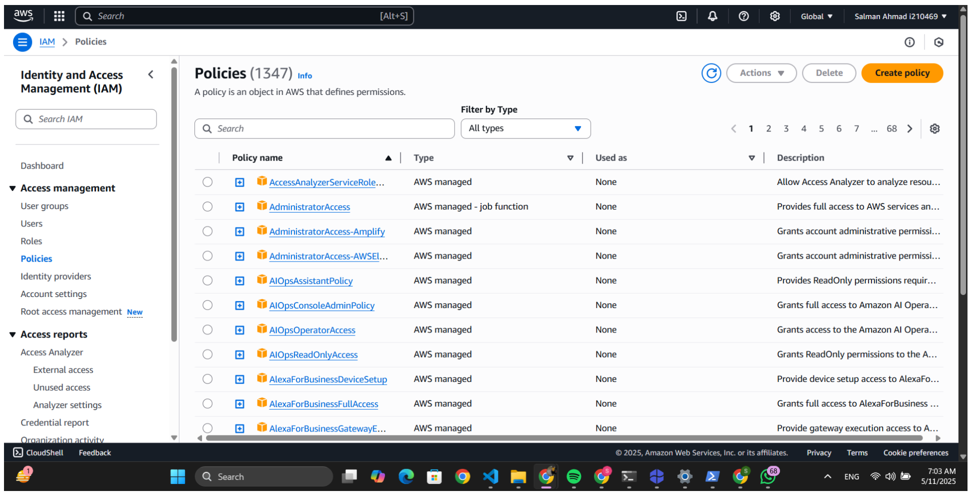
Role name	Trusted entities	Last activity
aws-elasticbeanstalk-ec2-role	AWS Service: ec2	11 minutes ago
aws-elasticbeanstalk-service-role	AWS Service: elasticbeanstalk	10 minutes ago
AWSServiceRoleForAutoScaling	AWS Service: autoscaling (Service-Li	12 minutes ago
AWSServiceRoleForRDS	AWS Service: rds (Service-Linked Rol	9 minutes ago
AWSServiceRoleForSupport	AWS Service: support (Service-Linke	-
AWSServiceRoleForTrustedAdvisor	AWS Service: trustedadvisor (Service	-
rds-monitoring-role	AWS Service: monitoring.rds	-
TaskApp-EB-Role-salman	AWS Service: ec2	12 hours ago
TaskApp-EC2-Role-salman	AWS Service: ec2	2 days ago

Roles Anywhere Info

Authenticate your non AWS workloads and securely provide access to AWS services.

Figure 2: IAM Role Creation

6.2 IAM Policies



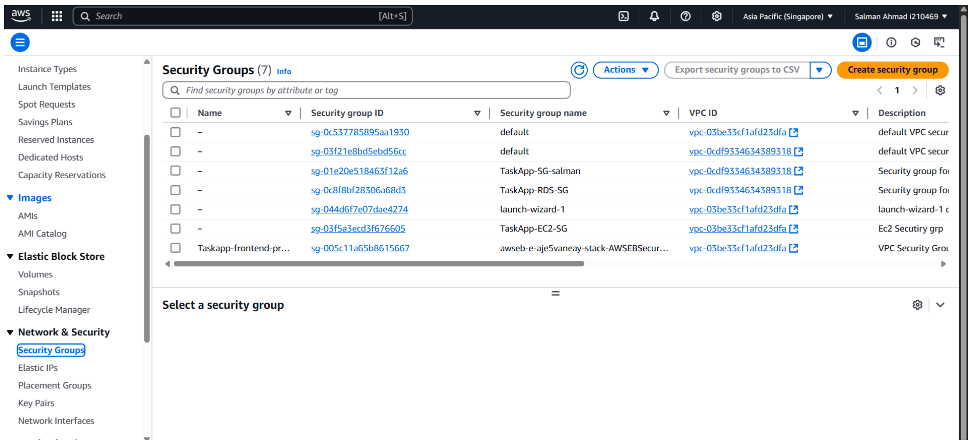
Policies (1347) Info

A policy is an object in AWS that defines permissions.

Policy name	Type	Used as	Description
AccessAnalyzerServiceRole…	AWS managed	None	Allow Access Analyzer to analyze resou…
AdministratorAccess	AWS managed - job function	None	Provides full access to AWS services an…
AdministratorAccess-Amplify	AWS managed	None	Grants account administrative permissi…
AdministratorAccess-AWSEl…	AWS managed	None	Grants account administrative permissi…
AIOpsAssistantPolicy	AWS managed	None	Provides ReadOnly permissions requir…
AIOpsConsoleAdminPolicy	AWS managed	None	Grants full access to Amazon AI Opera…
AIOpsOperatorAccess	AWS managed	None	Grants access to the Amazon AI Opera…
AIOpsReadOnlyAccess	AWS managed	None	Grants ReadOnly permissions to the A…
AlexaForBusinessDeviceSetup	AWS managed	None	Provide device setup access to AlexaFo…
AlexaForBusinessFullAccess	AWS managed	None	Grants full access to AlexaForBusiness …
AlexaForBusinessGatewayE…	AWS managed	None	Provide gateway execution access to A…

Figure 3: Configured IAM Policies

6.3 Security Groups Configuration



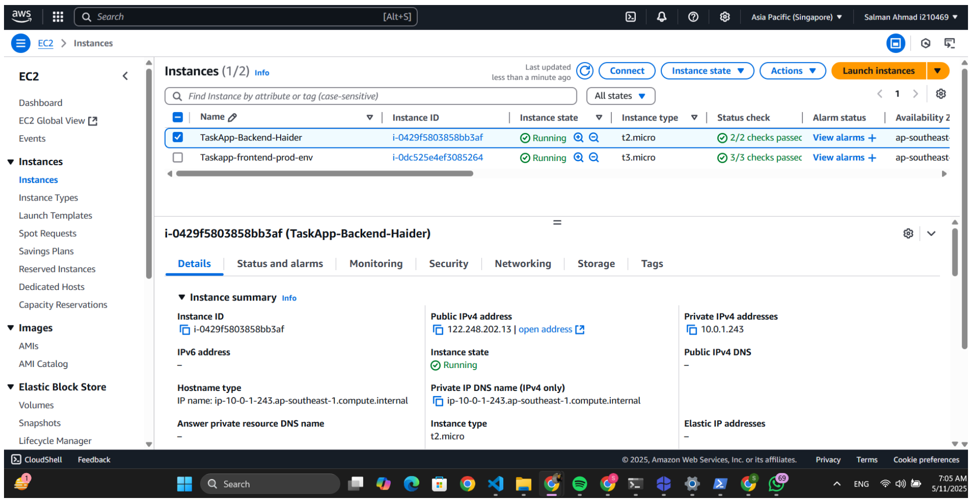
Security Groups (7)

Name	Security group ID	Security group name	VPC ID	Description
–	sg-0c537785895aa1930	default	vpc-03be33cf1afd23dfa	default VPC secur
–	sg-03f21e8bd5ebd56cc	default	vpc-0cdf9334634389318	default VPC secur
–	sg-01e20e518463f12a6	TaskApp-SG-salman	vpc-0cdf9334634389318	Security group fo
–	sg-0c8f8bf28306a68d3	TaskApp-RDS-SG	vpc-0cdf9334634389318	Security group fo
–	sg-044d6f7e07dae4274	launch-wizard-1	vpc-03be33cf1afd23dfa	launch-wizard-1 c
–	sg-03f5a3ecd3f676605	TaskApp-EC2-SG	vpc-03be33cf1afd23dfa	Ec2 Secutiry grp
Taskapp-frontend-pr…	sg-005c11a65b8615667	awseb-e-aje5vaneay-stack-AWSEBSecur…	vpc-03be33cf1afd23dfa	VPC Security Grou

Select a security group

Figure 4: Security Groups Configuration

6.4 EC2 Instance Deployment



Instances (1/2)

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Z
TaskApp-Backend-Haider	i-0429f5803858bb3af	Running	t2.micro	2/2 checks passec	View alarms +	ap-southeast
Taskapp-frontend-prod-env	i-0dc525e4ef3085264	Running	t3.micro	3/3 checks passec	View alarms +	ap-southeast

i-0429f5803858bb3af (TaskApp-Backend-Haider)

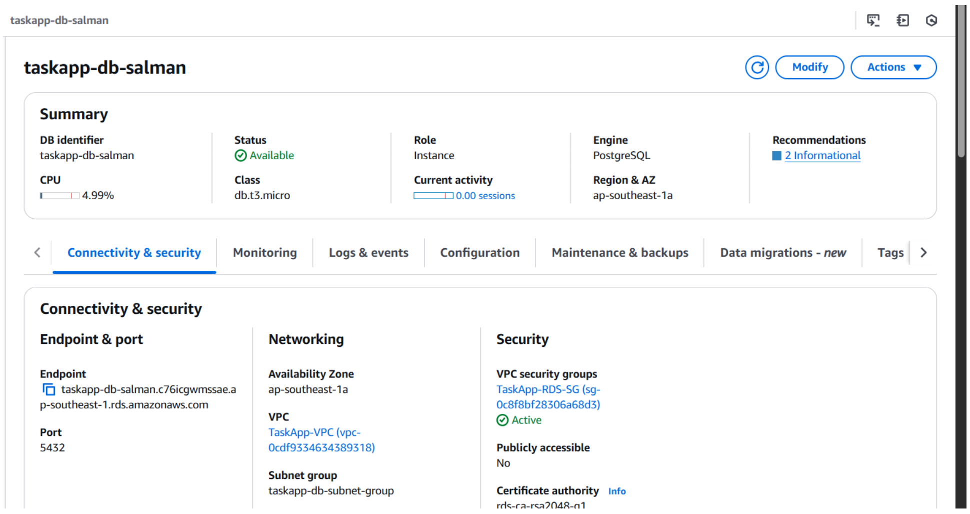
Instance summary

Instance ID i-0429f5803858bb3af	**Public IPv4 address** 122.248.202.13 | open address	**Private IPv4 addresses** 10.0.1.243
IPv6 address –	**Instance state** Running	**Public IPv4 DNS** –
Hostname type IP name: ip-10-0-1-243.ap-southeast-1.compute.internal	**Private IP DNS name (IPv4 only)** ip-10-0-1-243.ap-southeast-1.compute.internal	
Answer private resource DNS name –	**Instance type** t2.micro	**Elastic IP addresses** –

© 2025, Amazon Web Services, Inc. or its affiliates.

Figure 5: EC2 Instance for Backend

6.5 RDS Database Configuration


taskapp-db-salman

taskapp-db-salman

Summary

DB identifier	Status	Role	Engine	Recommendations
taskapp-db-salman	Available	Instance	PostgreSQL	2 Informational
CPU	**Class**	**Current activity**	**Region & AZ**	
4.99%	db.t3.micro	0.00 sessions	ap-southeast-1a	

Connectivity & security

Endpoint & port

Endpoint
taskapp-db-salman.c76icgwmssae.ap-southeast-1.rds.amazonaws.com

Port
5432

Networking

Availability Zone
ap-southeast-1a

VPC
TaskApp-VPC (vpc-0cdf9334634389318)

Subnet group
taskapp-db-subnet-group

Security

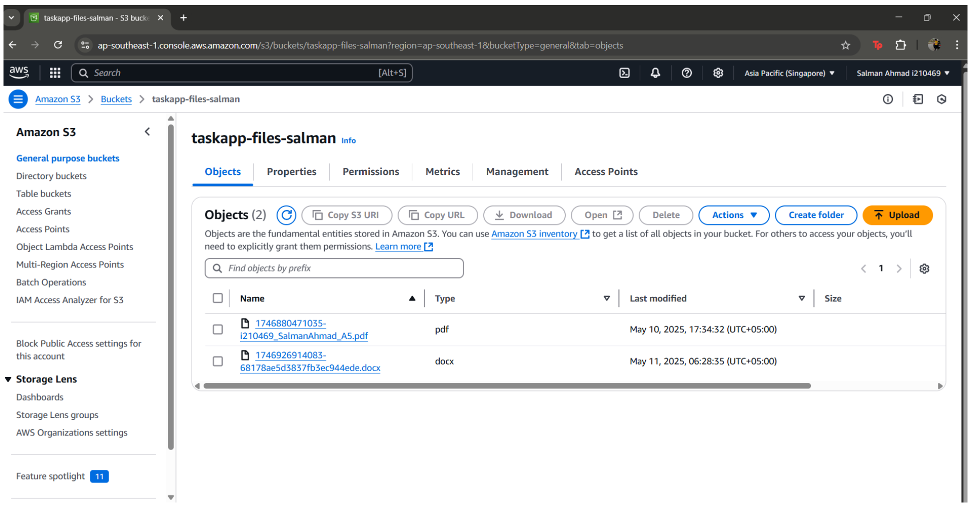
VPC security groups
TaskApp-RDS-SG (sg-0c8f8bf28306a68d3)
Active

Publicly accessible
No

Certificate authority Info
rds-ca-rsa2048-g1

Figure 6: RDS Database Configuration

6.6 S3 Bucket Configuration


taskapp-files-salman - S3 bucke

ap-southeast-1.console.aws.amazon.com/s3/buckets/taskapp-files-salman?region=ap-southeast-1&bucketType=general&tab=objects

taskapp-files-salman Info

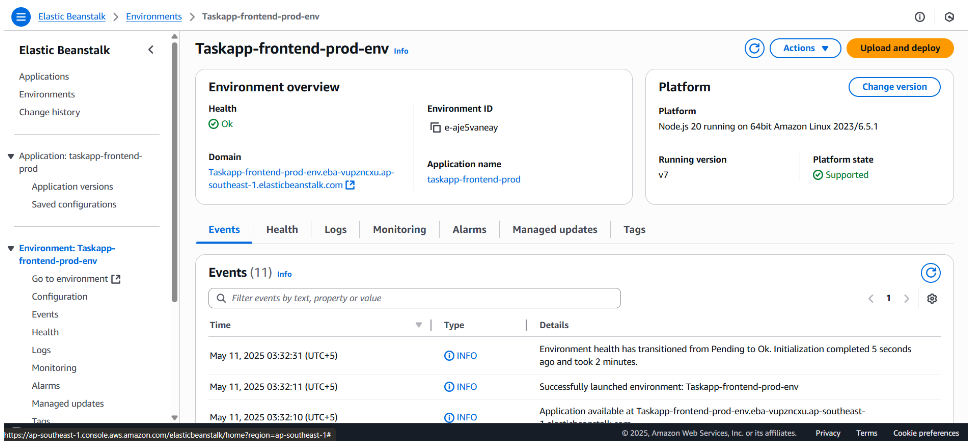
Objects (2)

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Name	Type	Last modified	Size
1746880471035-i210469_SalmanAhmad_A5.pdf	pdf	May 10, 2025, 17:34:32 (UTC+05:00)	
1746926914083-68178ae5d3837fb3ec944ede.docx	docx	May 11, 2025, 06:28:35 (UTC+05:00)	

Figure 7: S3 Bucket for File Storage

6.7 Elastic Beanstalk Deployment


Elastic Beanstalk > Environments > Taskapp-frontend-prod-env

Elastic Beanstalk
Applications
Environments
Change history
Application: taskapp-frontend-prod
Application versions
Saved configurations
Environment: Taskapp-frontend-prod-env
Go to environment
Configuration
Events
Health
Logs
Monitoring
Alarms
Managed updates
Tags

Taskapp-frontend-prod-env Info
Actions
Upload and deploy

Environment overview
Health: Ok
Environment ID: e-aje5vaneay
Domain: Taskapp-frontend-prod-env.eba-vupzncxu.ap-southeast-1.elasticbeanstalk.com
Application name: taskapp-frontend-prod

Platform
Change version
Platform: Node.js 20 running on 64bit Amazon Linux 2023/6.5.1
Running version: v7
Platform state: Supported

Events | Health | Logs | Monitoring | Alarms | Managed updates | Tags

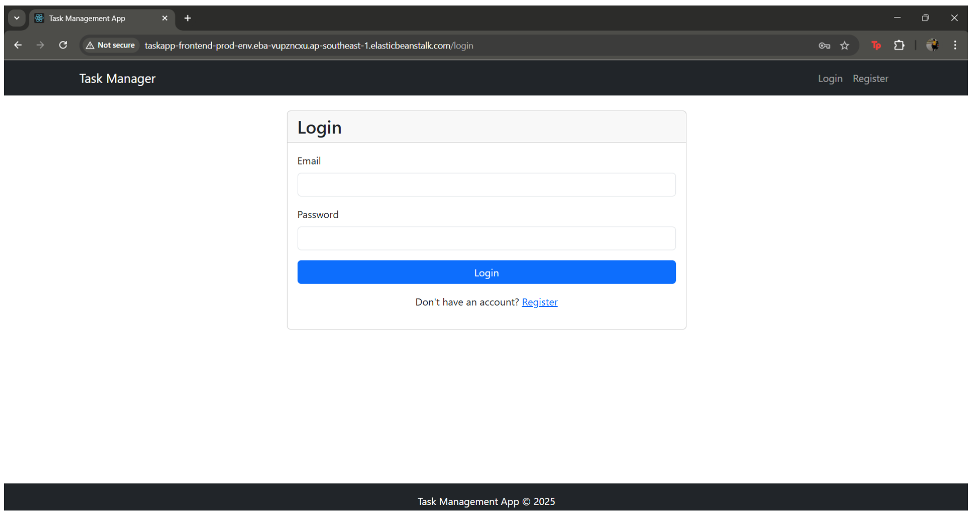
Events (11) Info
Filter events by text, property or value

Time	Type	Details
May 11, 2025 03:32:31 (UTC+5)	INFO	Environment health has transitioned from Pending to Ok. Initialization completed 5 seconds ago and took 2 minutes.
May 11, 2025 03:32:11 (UTC+5)	INFO	Successfully launched environment: Taskapp-frontend-prod-env
May 11, 2025 03:32:10 (UTC+5)	INFO	Application available at Taskapp-frontend-prod-env.eba-vupzncxu.ap-southeast-

https://ap-southeast-1.console.aws.amazon.com/elasticbeanstalk/home?region=ap-southeast-1#
© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Figure 8: Elastic Beanstalk Deployment

6.8 Application Screenshots

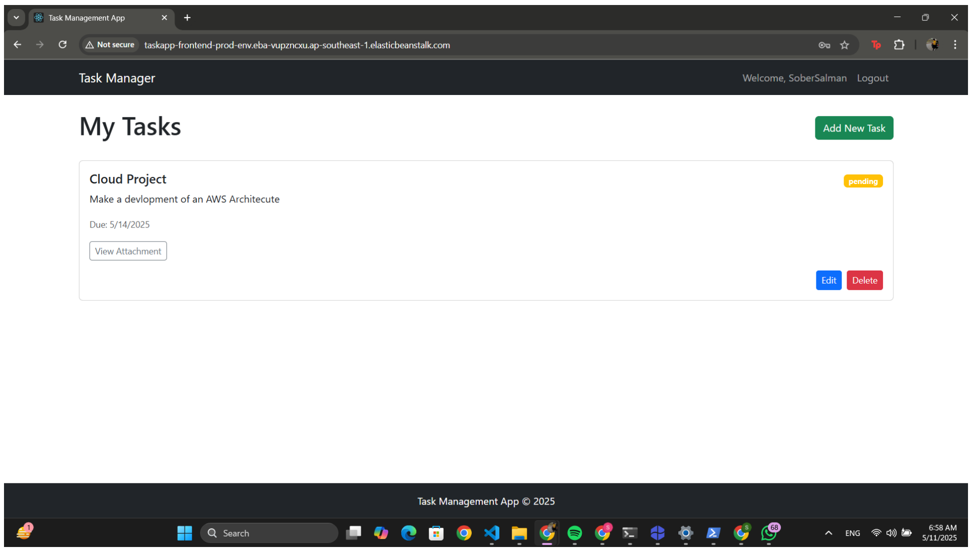

Task Management App
Not secure taskapp-frontend-prod-env.eba-vupzncxu.ap-southeast-1.elasticbeanstalk.com/login

Task Manager
Login Register

Login
Email
Password
Login
Don't have an account? Register

Task Management App © 2025

Figure 9: Application Login Screen



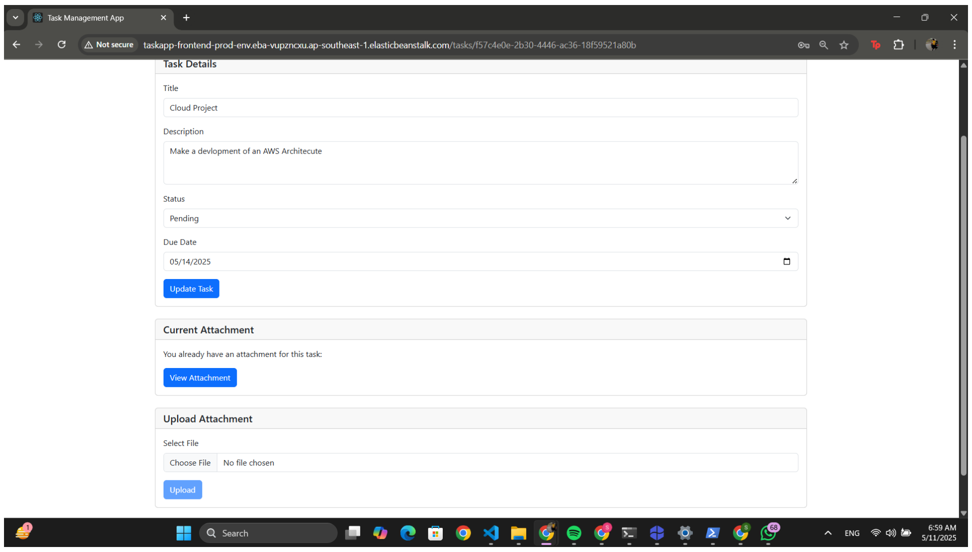
My Tasks

Cloud Project
pending
Make a devlopment of an AWS Architecute
Due: 5/14/2025

Task Management App © 2025

6:58 AM 5/11/2025

Figure 10: Application Dashboard


Task Management App
Not secure
taskapp-frontend-prod-env.eba-vupzncxu.ap-southeast-1.elasticbeanstalk.com/tasks/f57c4e0e-2b30-4446-ac36-18f59521a80b
Update Task
View Attachment
Choose File
Upload
Search
ENG

Task Details

Title: Cloud Project

Description: Make a devlopment of an AWS Architecute

Status: Pending

Due Date: 05/14/2025

Current Attachment

You already have an attachment for this task:

Upload Attachment

Select File: No file chosen

6:59 AM 5/11/2025

Figure 11: Task Detail with File Upload

7 Conclusion

This project successfully implements a cloud-based web application on AWS infrastructure following modern cloud-native architecture principles. The application includes all required components:

- User authentication system
- CRUD operations for task management
- Database integration using Amazon RDS
- File upload capabilities with Amazon S3
- Frontend and backend deployed as separate components
- Secure infrastructure with proper IAM roles and security groups

The deployment process demonstrates practical knowledge of AWS services and cloud architectural patterns. The application is secure, scalable, and follows best practices for cloud application development and deployment.

7.1 Future Improvements

- Implement HTTPS with AWS Certificate Manager
- Set up CI/CD pipeline for automated deployments
- Add monitoring and alerting with CloudWatch
- Implement auto-scaling for the application tier
- Enhance security with AWS WAF and Shield