Cloud Computing Tools and Techniques Lab Project

Lab 4

Deploying a Scalable 3-Tier Inventory Management Web Application on AWS

Presented By:

Sachin Mhetre: 22070122119
Mihir Hebalkar: 22070122120
Onkar Mendhapurkar: 22070122135

Under the guidance of: **Dr. Aditi Sharma**

Introduction to Our Cloud-Native Inventory Application

Project Scope

We created a full-stack, cloud-native inventory management system focused on smartphone products to meet modern business needs for scalability and efficiency.

- Frontend built with Next.js hosted on AWS Amplify
- Backend developed using Node.js/Express deployed on EC2 instances
- PostgreSQL database managed via Amazon RDS

Architecture Benefits

This 3-tier architecture supports modular development, allowing independent upgrades and scaling to maintain real-time responsiveness and robustness.

Using cloud services enables automated scaling, monitoring, and efficient CI/CD pipelines for continuous improvement.

Problem Statement: Challenges in Traditional Inventory Systems

Scalability Limitations

Legacy systems often fail under peak load, causing slowdowns and downtime.

Consistency & Control

Lack of centralized data consistency and robust access controls exposes risks.

Monitoring & Updates

Real-time monitoring and modular updates are difficult to implement on traditional platforms.

Our goal is to develop a cloud-native inventory system that overcomes these challenges using AWS's managed services, ensuring reliability and maintainability.

Project Objectives for AWS-Based Inventory System

Scalable Design

Build an app that scales automatically with traffic using AWS auto scaling and load balancing features.

Modern Frontend & Backend

Deploy Next.js frontend on Amplify and Node.js backend on EC2 instances with API Gateway for secure routing.

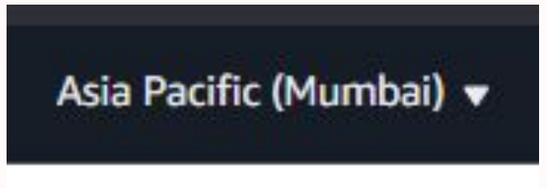
Robust Data Layer

Use PostgreSQL RDS instance configured in private subnet ensuring secure, managed data storage.

Enhanced Monitoring & CI/CD

Implement CloudWatch for real-time monitoring and GitHub integrated deployment for continuous delivery.

Technology Stack Overview



Frontend

Next.js framework hosted via AWS Amplify for fast, responsive UI and automatic deployment from GitHub.

Backend

Node.js with Express server running on EC2 instances behind Application Load Balancer with API Gateway integration.

Database & Infrastructure

Managed PostgreSQL database hosted on RDS in a private subnet, secured and isolated within a VPC.

- Auto Scaling Group manages backend server scalability
- CloudWatch enables logging and performance alarms

Key Features of the Inventory Management System

Inventory Management

Support for adding and editing products, stock monitoring, and price tracking to maintain accurate data.

Sales Dashboard

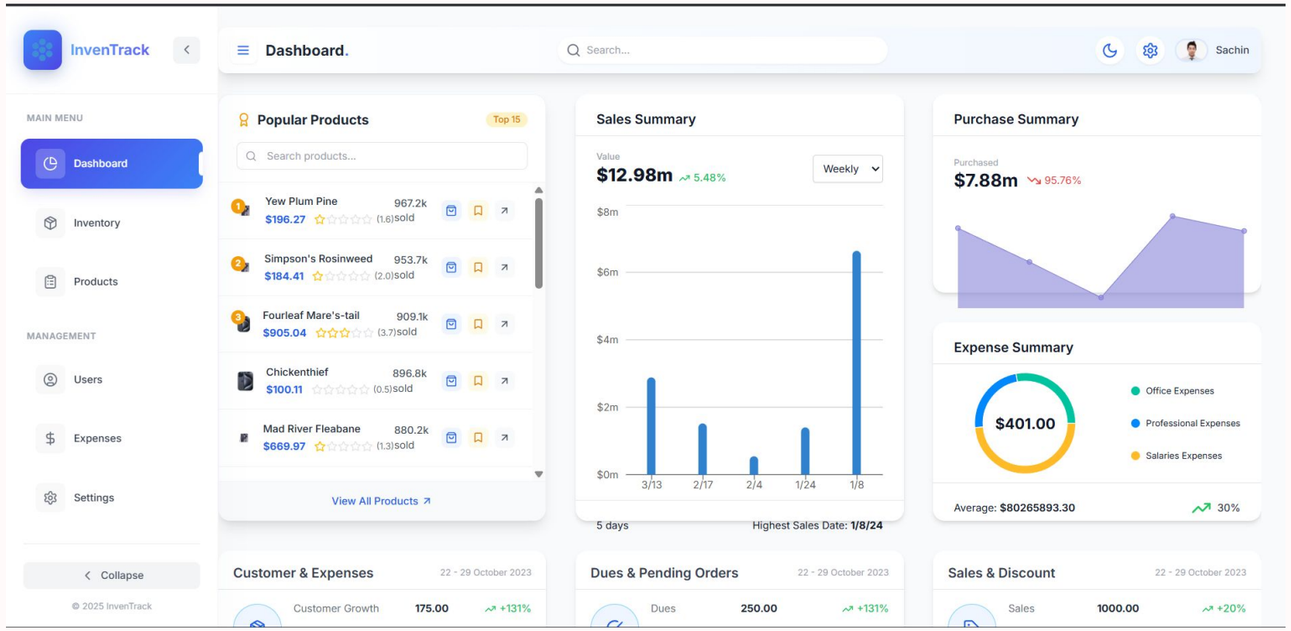
Visualize sales, purchases, and expense trends with actionable data insights for business decision-making.

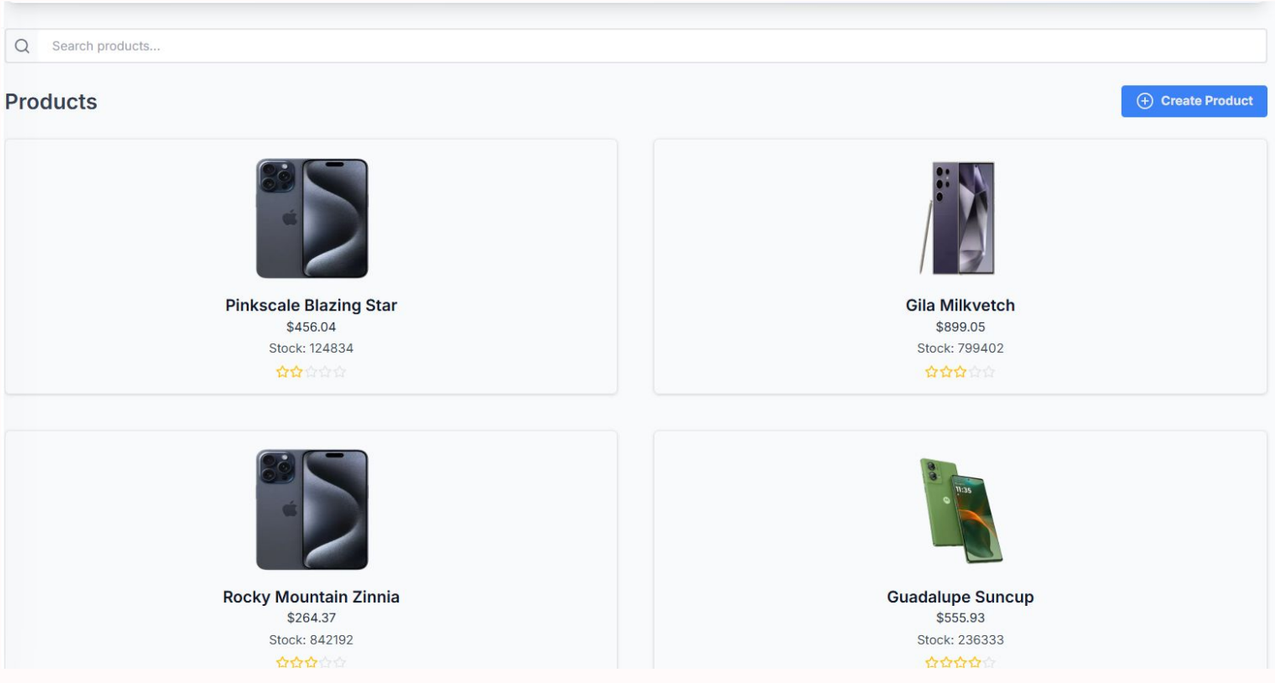
Scalability & Monitoring

Auto-scaling backend instances with load balancer and real-time monitoring via CloudWatch alerts and logs.

Security

Restrict API access and protect data via private subnets, enforcing secure communication and permissions.





AWS Network & Compute Setup

1

VPC & Subnet Creation

Established a Virtual Private Cloud with segregated public subnet for EC2 and private subnet for RDS database.

2

Internet Connectivity

Configured Internet Gateway and route tables to enable secure external access to public subnet resources.

3

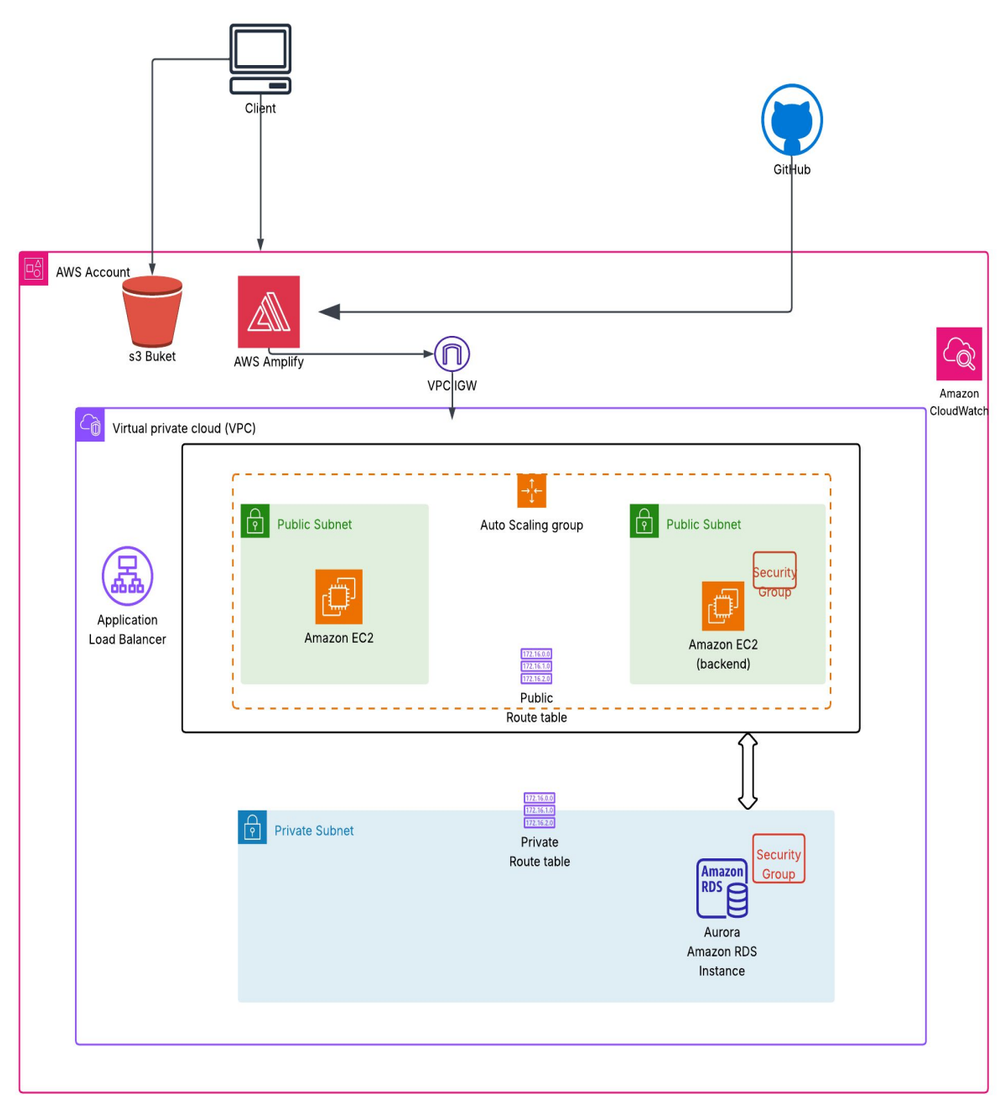
EC2 Backend Deployment

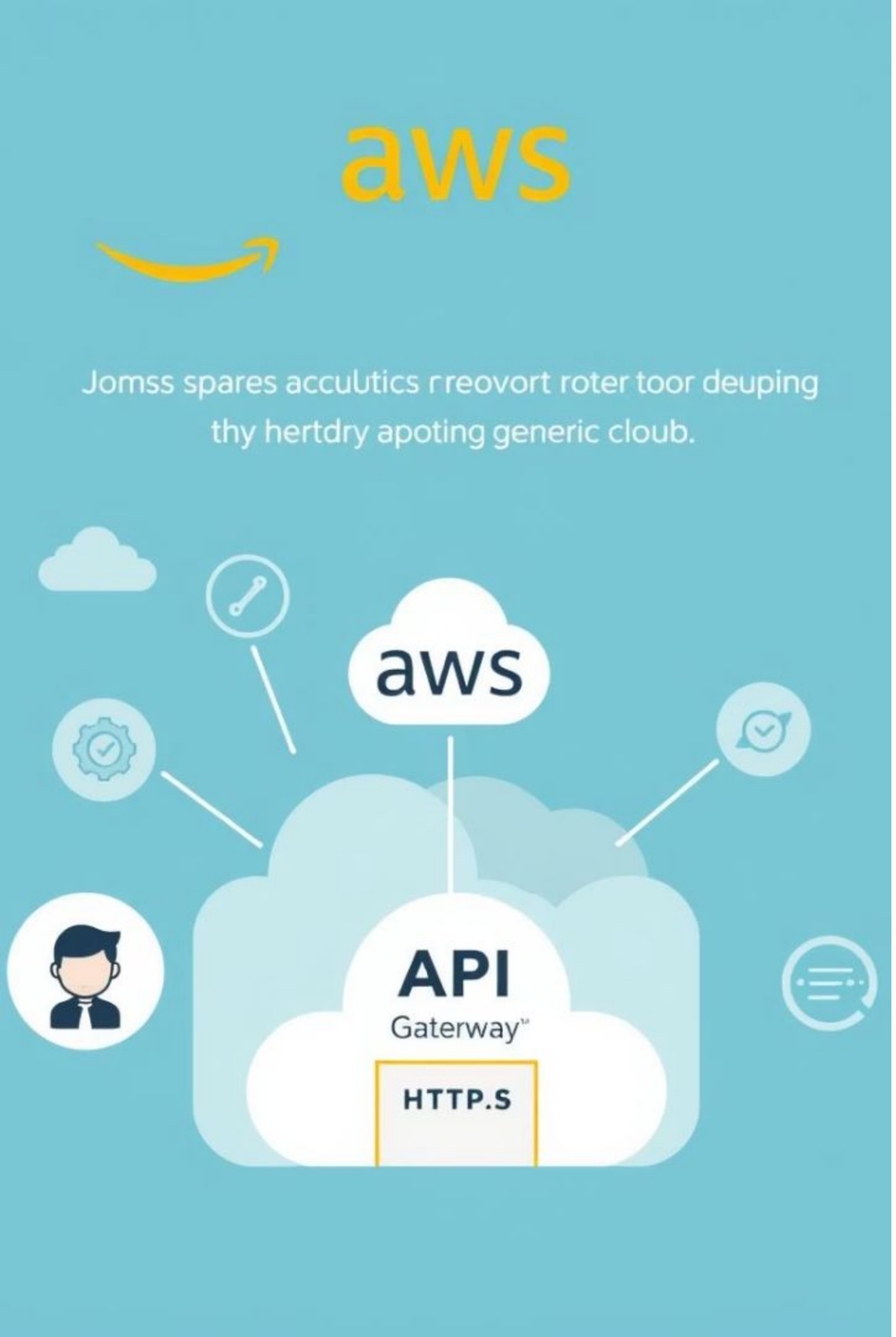
Launched and configured EC2 instance, deployed backend application using PM2 for process management.

4

Access & Testing

Connected using EC2 Instance Connect and validated backend availability on assigned public IP address.





Frontend Deployment & API Gateway Configuration

1

AWS Amplify Frontend

Hosted Next.js frontend application with continuous deployment connected to GitHub repository for auto-builds.

2

API Gateway Setup

Configured API Gateway as HTTPS proxy between frontend requests and backend HTTP services securely.

3

Environment Variables

Configured Amplify environment for sensitive backend URLs and credentials ensuring secure configuration management.

4

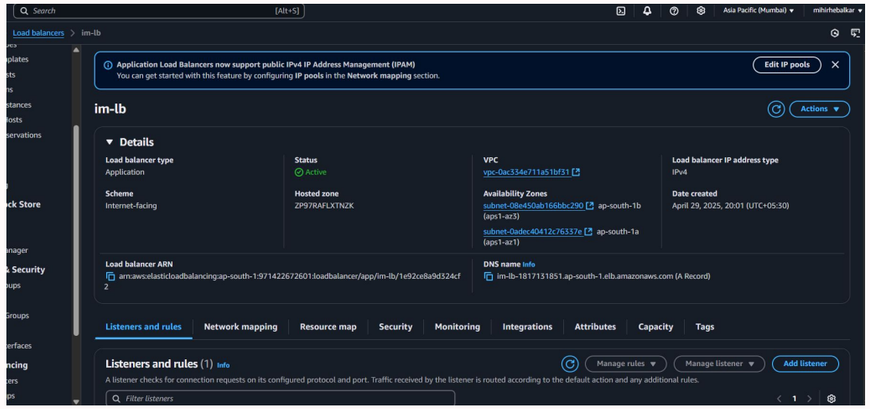
Production Endpoint

The final deployed API is accessible via a secure public URL optimized for scalable frontend-backend communication.

Load Balancer, Auto Scaling & S3 Integration

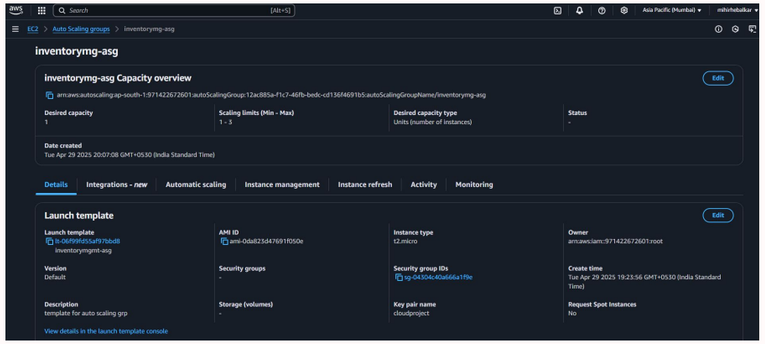
Application Load Balancer

Configured ALB with multi-AZ target groups to distribute inbound traffic evenly for high availability.



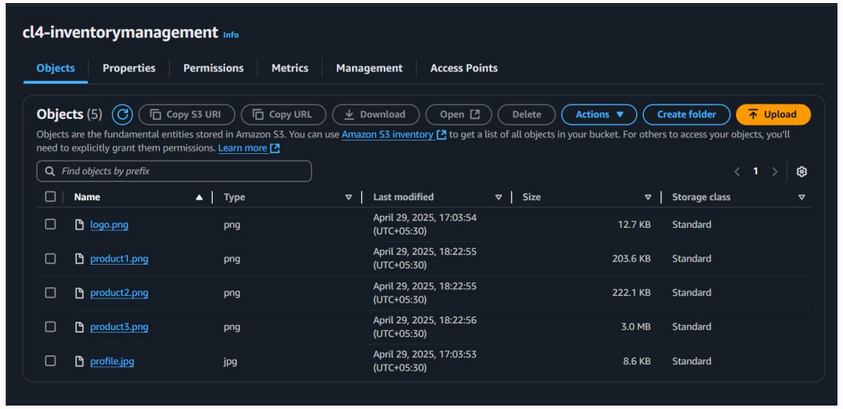
Auto Scaling Group

Set up auto scaling policy for backend EC2 instances to dynamically adjust capacity based on load.



Asset Storage in S3

Created an S3 bucket to store static assets such as logos and files with public access policy for delivery. Frontend configuration updated to enable image rendering from the S3 bucket seamlessly.



Conclusion and Impact

Scalable & Highly Available

Elastic infrastructure ensures consistent performance regardless of user demand or growth.

Modular & CI/CD Ready

Continuous integration and delivery pipelines enable rapid, reliable updates and modular improvements.

Secure Data Management

Private subnet database and controlled API access maintain strong security and compliance posture.

Comprehensive Monitoring

CloudWatch integration allows real-time insight and alerting for operational health and issues.

This AWS deployed 3-tier inventory system offers a robust platform for growing businesses requiring efficient, secure, and scalable inventory management.