



UNIVERSITY OF
MARYLAND

ENPM 818N : CLOUD COMPUTING
GROUP 9 Final Project Report

SPRING 2025

Link to Project Files : [GitHub Link](#)

Bolla Sai Saketh

Fahad Shaker

Shiv Ramolia

Swaraj M Rao

Table of Contents

Project Overview -----	4
Infrastructure Setup-----	6
Overview of the Infrastructure and Implementation -----	6
1. VPC and Networking Structure -----	6
1.1 CIDR Block -----	7
1.2 Multi-AZ VPC Design -----	7
1.3 Subnets -----	8
1.4 Route Tables -----	8
1.5 NAT Gateways and Elastic Ips-----	9
1.6 Internet Gateway -----	9
1.7 Security Groups-----	9
Phase 1 -----	12
Part 1 : Docker Deployment -----	12
1.1 EC2 Instance Setup -----	12
1.2 Application Deployment with Docker Compose -----	13
1.3 Validation and Observability-----	14
1.4 Cleanup-----	18
Part 2: Kubernetes Deployment on Amazon EKS-----	19
2.1 EKS Cluster Provisioning -----	19
2.2 Admin Node Setup-----	21
2.3 Application Deployment to EKS-----	21
Phase 2: Integrating Helm for Deployment -----	28
2.1 Adding and Updating the Helm Repository -----	28
2.2 Helm-Based Deployment into a Separate Namespace-----	29
2.3 Upgrade and Rollback Simulation-----	33
Phase 3: Alerting Service and Notifications -----	37
Phase 4: CI/CD Integration -----	41

4.1 CI/CD pipeline using GitHub Actions -----	41
4.2 Security within workflow & deployment -----	41
4.3 Rollback strategies -----	45
4.4 Testing Stage in CI/CD Pipeline -----	46
Conclusion-----	47

Project Overview

This AWS-based observability platform was designed to demonstrate a scalable, resilient, and production-grade deployment of the OpenTelemetry Demo Application using Amazon EKS. The project aimed to integrate modern DevOps workflows, Kubernetes-native monitoring, and cloud infrastructure best practices to emulate telemetry pipelines.

To begin, a highly available Virtual Private Cloud (VPC) was provisioned using AWS CloudFormation. The VPC architecture included public and private subnets across two Availability Zones, an Internet Gateway, and dual NAT Gateways for fault-tolerant internet access from private resources. Subnets were strategically assigned to support EKS control plane communication, frontend ingress, and internal monitoring services.

Security was established using a layered approach. Dedicated security groups were created for EKS control plane, worker nodes, admin EC2 access, frontend ingress, and monitoring dashboards. These groups restricted network access to essential ports and CIDR ranges, ensuring secure communication paths between services while allowing external access where needed (e.g., HTTP/HTTPS for the frontend).

Docker was used initially on a standalone EC2 instance to validate the application setup using the provided docker-compose.yaml. This step confirmed container orchestration, internal service connectivity, and basic functionality of endpoints like /, /cart, and /products.

The validated setup was then transitioned to Kubernetes. An Amazon EKS cluster was deployed with two t3.xlarge worker nodes, spread across private subnets for high availability. A separate EC2 instance was configured with kubectl, eksctl, and AWS CLI to serve as the cluster administrator node. Application resources were applied using the provided OpenTelemetry manifest file, and successful pod deployments and service accessibility were verified within the otel-demo namespace.

To enhance maintainability and upgradeability, Helm was introduced in the next phase. The OpenTelemetry Helm repository was added, and the application was re-deployed using Helm charts into a separate namespace. This allowed for parameter overrides, simplified upgrades, and rollbacks. Helm's package management capabilities also made it easier to manage complex sets of Kubernetes resources as a unified release.

Monitoring and alerting were implemented using Prometheus and Grafana, deployed within the cluster. Custom Prometheus alert rules were configured to trigger based on pod restart thresholds. Alert notifications were integrated with email using SMTP to notify operators in real time about service-level anomalies.

To close the DevOps loop, a CI/CD pipeline was built using GitHub Actions. The pipeline automated Docker image builds, pushed them to a container registry, and applied changes to the Kubernetes cluster upon successful test completion. Failure handling was addressed using Helm rollback functionality built into the workflow, ensuring the system could safely revert to a stable state if a deployment failed. Secrets were securely managed using GitHub Secrets for tokens, credentials, and kubeconfig files.

Overall, this project effectively demonstrated how a real-world, microservices-based observability platform can be deployed and managed on AWS using EKS, Helm, Prometheus, and CI/CD pipelines. Each phase of the project reflected best practices in infrastructure-as-code, secure cloud deployment, and Kubernetes-native application lifecycle management.

Link to Project Files: [GitHub Link](#)

Infrastructure Setup

This report outlines the planning, deployment, and validation of a production-grade observability platform using OpenTelemetry on AWS. The project focused on leveraging cloud-native technologies to build a scalable, resilient, and secure microservices architecture, emphasizing best practices in infrastructure automation, Kubernetes operations, and performance monitoring.

Key AWS services included Amazon EKS for container orchestration, CloudFormation for infrastructure provisioning, and EC2 for administrative control and testing. Helm was used to streamline Kubernetes resource management, while Prometheus and Grafana enabled robust observability and alerting. The system was further enhanced with a CI/CD pipeline built on GitHub Actions to automate application delivery and rollback. Together, these components provided a reliable, extensible, and cost-conscious framework suitable for real-world telemetry workloads.

Overview of the Infrastructure and Implementation

This architecture is designed to support a secure, scalable, and resilient observability platform using Kubernetes on AWS. It leverages a suite of AWS services to manage networking, compute resources, application orchestration, monitoring, and security. Core components include a highly available VPC, an Amazon EKS cluster with worker nodes distributed across multiple subnets, an EC2-based admin node for cluster control, and secure connectivity managed via custom security groups. Additional layers include Helm-based deployment automation, Prometheus-Grafana for observability, and GitHub Actions CI/CD for continuous delivery—each contributing to the reliability, traceability, and maintainability of the system.

1. VPC and Networking Structure

To ensure high availability, fault tolerance, and secure segmentation of services, the foundational network infrastructure for the OpenTelemetry EKS project was provisioned entirely using AWS CloudFormation. The design includes a multi-AZ VPC, route tables, NAT gateways, and fine-tuned security groups to enforce controlled communication paths between components.

Link to the VPC CloudFormation files : [GitHub Link](#)

The screenshot shows the AWS VPC Resource Map for the 'otel-demo-vpc'. It includes sections for VPC details, Subnets (4), Route tables (4), and Network connections (3). The Subnets section shows two AZs: us-east-1a and us-east-1b, each with two subnets: public and private. The Route tables section lists four route tables: otel-demo-private-rt-2, otel-demo-public-rt, rtb-05bb9b582df902dd2, and otel-demo-private-rt-1. The Network connections section lists three connections: otel-demo-igw, otel-demo-natgw-1, and otel-demo-natgw-2.

1.1 CIDR Block

The entire architecture is encapsulated within a dedicated Virtual Private Cloud (VPC) with the CIDR block:

- **10.0.0.0/16** — This large address range **allows for multiple subnets, scalable node groups, and future network expansion across availability zones**.

The screenshot shows the AWS CIDR Block Management interface. It displays a single IPv4 CIDR entry: 10.0.0.0/16, which is associated with the VPC 'otel-demo-vpc'. There is a button labeled 'Edit CIDRs' at the top right.

1.2 Multi-AZ VPC Design

High availability is achieved by distributing subnets and workloads across two Availability Zones (AZs). The architecture includes:

- **Public Subnet 1:** 10.0.1.0/24 in **AZ1**
- **Public Subnet 2:** 10.0.2.0/24 in **AZ2**
- **Private Subnet 1:** 10.0.3.0/24 in **AZ1**
- **Private Subnet 2:** 10.0.4.0/24 in **AZ2**

This ensures workload distribution for services like EKS worker nodes and public-facing resources such as NAT Gateways or Load Balancers.

<input type="checkbox"/> otel-demo-private-subnet-1	subnet-0c2b5596e115d9b2	<input checked="" type="radio"/> Available	vpc-08aa060272d674c5d otel...	<input type="radio"/> Off	10.0.3.0/24	-	-	251	us-east-1a	use1-az1
<input type="checkbox"/> otel-demo-private-subnet-2	subnet-098b33900eaf9f53	<input checked="" type="radio"/> Available	vpc-08aa060272d674c5d otel...	<input type="radio"/> Off	10.0.4.0/24	-	-	251	us-east-1b	use1-az2
<input type="checkbox"/> otel-demo-public-subnet-1	subnet-08a096745ca0d5b	<input checked="" type="radio"/> Available	vpc-08aa060272d674c5d otel...	<input type="radio"/> Off	10.0.1.0/24	-	-	246	us-east-1a	use1-az1
<input type="checkbox"/> otel-demo-public-subnet-2	subnet-045c22a3535dd49	<input checked="" type="radio"/> Available	vpc-08aa060272d674c5d otel...	<input type="radio"/> Off	10.0.2.0/24	-	-	250	us-east-1b	use1-az2

1.3 Subnets

The subnets were designed as follows:

- **Public Subnets** are configured with **MapPublicIpOnLaunch**: true to allow resources (e.g., NAT Gateways) to be publicly accessible.
- **Private Subnets** host sensitive components like **EKS worker nodes** and monitoring tools, **keeping them isolated from the public internet**.

All subnets are tagged for identification and grouped by functionality.

Actions ▾			
Details			
Subnet ID	subnet-08a6967d76cae0d3b	Subnet ARN	arn:aws:ec2:us-east-1:944362433564:subnet/subnet-08a6967d76cae0d3b
IPv4 CIDR	10.0.1.0/24	State	Available
Availability Zone	us-east-1a	IPv6 CIDR	—
Route table	rtb-02b780c11f49e846 otel-demo-public-rt	Network border group	us-east-1
Auto-assign IPv6 address	No	Default subnet	No
IPv4 CIDR reservations	—	Customer-owned IPv4 pool	—
Resource name DNS A record	Disabled	IPv6-only	No
		DNS64	Disabled
		Hostname type	IP name
		Owner	944362433564

Public Subnet

Actions ▾			
Details			
Subnet ID	subnet-0983a3f900e9af853	Subnet ARN	arn:aws:ec2:us-east-1:944362433564:subnet/subnet-0983a3f900e9af853
IPv4 CIDR	10.0.4.0/24	State	Available
Availability Zone	us-east-1b	IPv6 CIDR	—
Route table	rtb-05a60bc9d15c996c9 otel-demo-private-rt-2	Network border group	us-east-1
Auto-assign IPv6 address	No	Default subnet	No
IPv4 CIDR reservations	—	Customer-owned IPv4 pool	—
Resource name DNS A record	Disabled	IPv6-only	No
		DNS64	Disabled
		Hostname type	IP name
		Owner	944362433564

Private Subnet

1.4 Route Tables

Custom route tables were configured and associated with the subnets to define the traffic flow:

- **Public Route Table** routes **0.0.0.0/0 traffic through an Internet Gateway**, attached to the VPC.
- **Private Route Tables** (1 and 2) route outbound internet traffic **via corresponding NAT Gateways**, allowing private instances to access external services securely without being exposed directly.

Each subnet is associated with its appropriate route table to ensure correct routing.

<input type="checkbox"/>	otel-demo-private-rt-1	rtb-057e0f897dbba922	subnet-0c2b65966e115d...	-	No	vpc-08aa060272d674c5d tel...	944362433564
<input type="checkbox"/>	otel-demo-private-rt-2	rtb-05a60b9d15c996c9	subnet-0983a3f900e9af8...	-	No	vpc-08aa060272d674c5d tel...	944362433564
<input type="checkbox"/>	otel-demo-public-rt	rtb-02b780c111f49e846	2 subnets	-	No	vpc-08aa060272d674c5d tel...	944362433564

1.5 NAT Gateways and Elastic Ips

To support outbound internet access from private subnets:

- **Two NAT Gateways** were provisioned, one in each public subnet.
- Each NAT Gateway uses an **Elastic IP allocated** in advance for stable public addressing.

This setup guarantees continued internet access in case of AZ-specific outages.

nat-0cfdf65b070c54843f / otel-demo-natgw-1				Actions ▾
Details NAT gateway ID: nat-0cfdf65b070c54843f NAT gateway ARN: arn:aws:ec2:us-east-1:944362433564:natgateway/nat-0cfdf65b070c54843f VPC: vpc-08aa060272d674c5d / otel-demo-vpc	Connectivity type : Public Primary public IPv4 address: 54.162.106.204 Subnet: subnet-08a6967d76cae0d3b / otel-demo-public-subnet-1	State : Available Primary private IPv4 address: 10.0.1.191 Created: Wednesday, April 30, 2025 at 19:49:59 EDT	State message : Info Primary network interface ID: eni-0a936e7d0ef73c2dd Deleted: -	

nat-0c4e422eb48b78e7f / otel-demo-natgw-2				Actions ▾
Details NAT gateway ID: nat-0c4e422eb48b78e7f NAT gateway ARN: arn:aws:ec2:us-east-1:944362433564:natgateway/nat-0c4e422eb48b78e7f VPC: vpc-08aa060272d674c5d / otel-demo-vpc	Connectivity type : Public Primary public IPv4 address: 3.219.19.185 Subnet: subnet-0d5c22ca3539ddb49 / otel-demo-public-subnet-2	State : Available Primary private IPv4 address: 10.0.2.243 Created: Wednesday, April 30, 2025 at 19:50:00 EDT	State message : Info Primary network interface ID: eni-01a996cbd9ea90dfd Deleted: -	

1.6 Internet Gateway

An Internet Gateway was attached to the VPC to enable internet connectivity for resources in the public subnets, such as NAT Gateways and load balancer endpoints.

igw-08bdf115ee6d8cac7 / otel-demo-igw				Actions ▾												
Details Info Internet gateway ID: igw-08bdf115ee6d8cac7	State : Attached	VPC ID : vpc-08aa060272d674c5d / otel-demo-vpc	Owner : 944362433564													
Tags <input type="text"/> Search tags <table border="1"> <tr> <td>Key</td> <td>Value</td> </tr> <tr> <td>aws:cloudfor...</td> <td>InternetGateway</td> </tr> <tr> <td>aws:cloudfor...</td> <td>arn:aws:cloudformation:us-east-1:944362433564:stack/otel-demo-vpc/c60d8fd0-261d-11f0-82c1-0ee6593b78bd</td> </tr> <tr> <td>Project</td> <td>OpenTelemetry</td> </tr> <tr> <td>Name</td> <td>otel-demo-igw</td> </tr> <tr> <td>aws:cloudfor...</td> <td>otel-demo-vpc</td> </tr> </table>				Key	Value	aws:cloudfor...	InternetGateway	aws:cloudfor...	arn:aws:cloudformation:us-east-1:944362433564:stack/otel-demo-vpc/c60d8fd0-261d-11f0-82c1-0ee6593b78bd	Project	OpenTelemetry	Name	otel-demo-igw	aws:cloudfor...	otel-demo-vpc	Manage tags < 1 >
Key	Value															
aws:cloudfor...	InternetGateway															
aws:cloudfor...	arn:aws:cloudformation:us-east-1:944362433564:stack/otel-demo-vpc/c60d8fd0-261d-11f0-82c1-0ee6593b78bd															
Project	OpenTelemetry															
Name	otel-demo-igw															
aws:cloudfor...	otel-demo-vpc															

1.7 Security Groups

Security was enforced using purpose-specific Security Groups, provisioned via a separate **CloudFormation template**. These include:

- EKS Cluster Security Group:** Allows secure access to the Kubernetes API server on port 443.

The screenshot shows the AWS CloudFormation console with the security group details for the EKS Cluster. The security group name is "otel-security-groups-ClusterSecurityGroup-BeyBqj1Y03QF". It has a security group ID of "sg-0aee6326e91173ac9", a description of "Allow access to EKS API server (port 443)", and a VPC ID of "vpc-08aa060272d674c5d". The owner is listed as "944362433564". There is one inbound rule entry: "sgr-0b2c3ca5b7f9ec617" (IPv4, HTTPS, TCP, port 443, source 0.0.0.0/0). The outbound rules count is also 1.

- Node Security Group:** Enables inter-node communication and traffic from the cluster control plane.

The screenshot shows the AWS CloudFormation console with the security group details for the Node Security Group. The security group name is "otel-security-groups-NodeSecurityGroup-Z664lilmnaXd". It has a security group ID of "sg-00c5b2ace37db0012", a description of "Allow EKS worker node communication with control plane between nodes", and a VPC ID of "vpc-08aa060272d674c5d". The owner is listed as "944362433564". There are 3 inbound rule entries: "sg-01d4ff40be0be4f" (HTTPS, TCP, port 441, source 0.0.0.0/0), "sg-0bd4774a081225fe" (IPv4, All, All, 10.0.0.0/16), and "sg-0fbfa1a162d72c68" (Custom TCP, TCP, port range 1025 - 65535, source 0.0.0.0/0).

- Admin EC2 Security Group:** Grants SSH (22) and HTTPS (443) access for the EKS administrator instance.

The screenshot shows the AWS CloudFormation console with the security group details for the Admin EC2 Security Group. The security group name is "otel-security-groups-AdministratorSecurityGroup-HKXOJnjBPURQ". It has a security group ID of "sg-0cd9e6abe8f0362f9", a description of "EC2 instance for EKS administration (kubernetes, eksctl)", and a VPC ID of "vpc-08aa060272d674c5d". The owner is listed as "944362433564". There are 4 inbound rule entries: "sg-01d4ff40be0be4f" (HTTPS, TCP, port 441, source 0.0.0.0/0), "sg-0fbfa1a162d72c68" (HTTP, TCP, port 80, source 0.0.0.0/0), "sg-03028949427321" (All TCP, TCP, port range 0 - 65535, source 0.0.0.0/0), and "sg-0bedecfc26f3d8ed6" (SSH, TCP, port 22, source 0.0.0.0/0).

- Application Ingress Security Group:** Allows HTTP (80) and HTTPS (443) traffic to frontend services.

The screenshot shows the AWS CloudFormation console with the security group details for the Application Ingress Security Group. The security group name is "otel-security-groups-AppIngressSecurityGroup-fp0FSN00p1h". It has a security group ID of "sg-0908a8e2585867edb", a description of "Allow public access to frontend apps (HTTP/HTTPS)", and a VPC ID of "vpc-08aa060272d674c5d". The owner is listed as "944362433564". There are 2 inbound rule entries: "sg-0073795779869e4d" (HTTP, TCP, port 80, source 0.0.0.0/0) and "sg-002113a3a9u9780ba2" (HTTPS, TCP, port 443, source 0.0.0.0/0).

- **Monitoring Security Group:** Internal-only access to Prometheus (9090) and Grafana (3000) dashboards from within the VPC.

sg-0fc0d945161475551 - otel-security-groups-MonitoringSecurityGroup-5w6DxKC0KyN7

[Actions ▾](#)

Details	Security group ID	Description	VPC ID
Security group name otel-security-groups-MonitoringSecurityGroup-5w6DxKC0KyN7	sg-0fc0d945161475551	Internal-only Prometheus/Grafana to access K8s services	vpc-08aa060272d674c5d
Owner 944362433564	Inbound rules count 2 Permission entries	Outbound rules count 1 Permission entry	

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

Inbound rules (2)

Name	Security group rule ID	IP version	Type	Protocol	Port range	Source
-	sgr-0cae6b886d49ce2bf	IPv4	Custom TCP	TCP	9090	10.0.0.0/16
-	sgr-0de2d83a7c008f16a	IPv4	Custom TCP	TCP	3000	10.0.0.0/16

[Manage tags](#) | [Edit inbound rules](#)

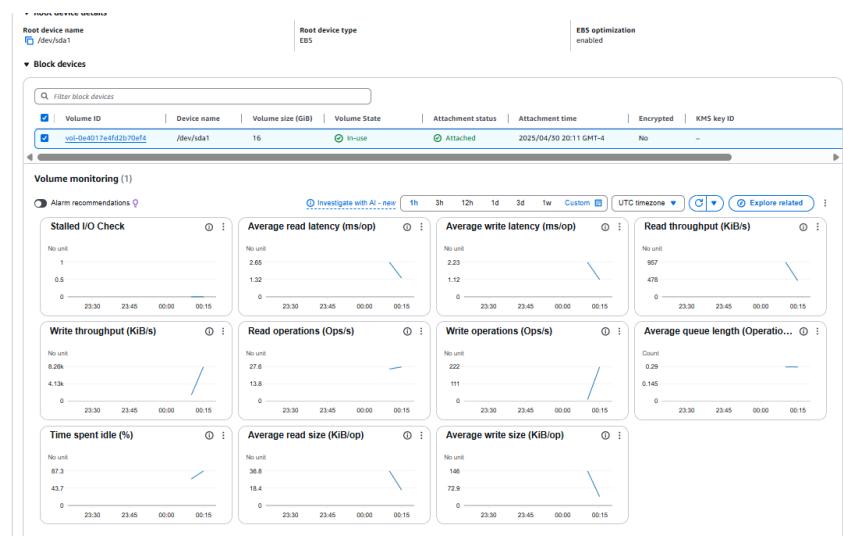
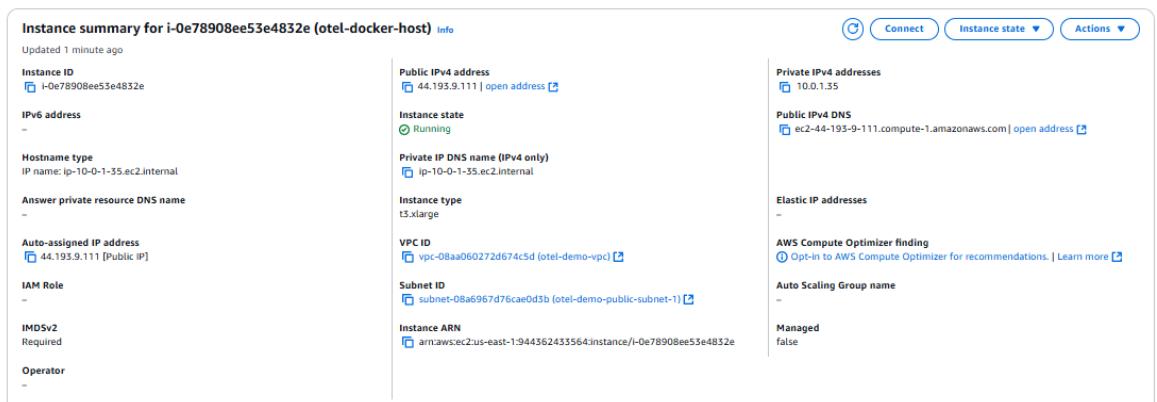
Phase 1

Part 1 : Docker Deployment

As a first step in validating the OpenTelemetry Demo Application, a standalone EC2 instance was provisioned to deploy the application using Docker. This approach enabled isolated testing of all services defined in the official docker-compose.yaml prior to Kubernetes integration.

1.1 EC2 Instance Setup

To begin the project, a dedicated EC2 instance was provisioned on AWS to host the OpenTelemetry demo application using Docker. The instance used was of type **t3.large**, meeting the requirement with **4 vCPU and 16.0 GiB's of memory**, with an **EBS volume attached and configured with at least 16GB of storage**. For operating system Ubuntu was selected(**Ubuntu Server 24.04 LTS**), and Docker along with Docker Compose were installed. The current user was added to the docker group, allowing Docker commands to be executed without elevated privileges.



1.2 Application Deployment with Docker Compose

The OpenTelemetry demo application was deployed using the docker-compose.yml file from the official OpenTelemetry GitHub repository.

Deployment steps included:

- Cloning the repository using:

```
git clone https://github.com/open-telemetry/opentelemetry-demo.git
```

- Navigating to the directory and deploying the application stack using :

```
cd opentelemetry-demo
```

```
docker compose up -d
```

- This brought up all essential services including:
 - frontend
 - loadgen
 - checkoutservice
 - productcatalogservice
 - otel-collector
 - jaeger
 - grafana
 - several backend microservices

Container health and status were verified using: **docker ps**

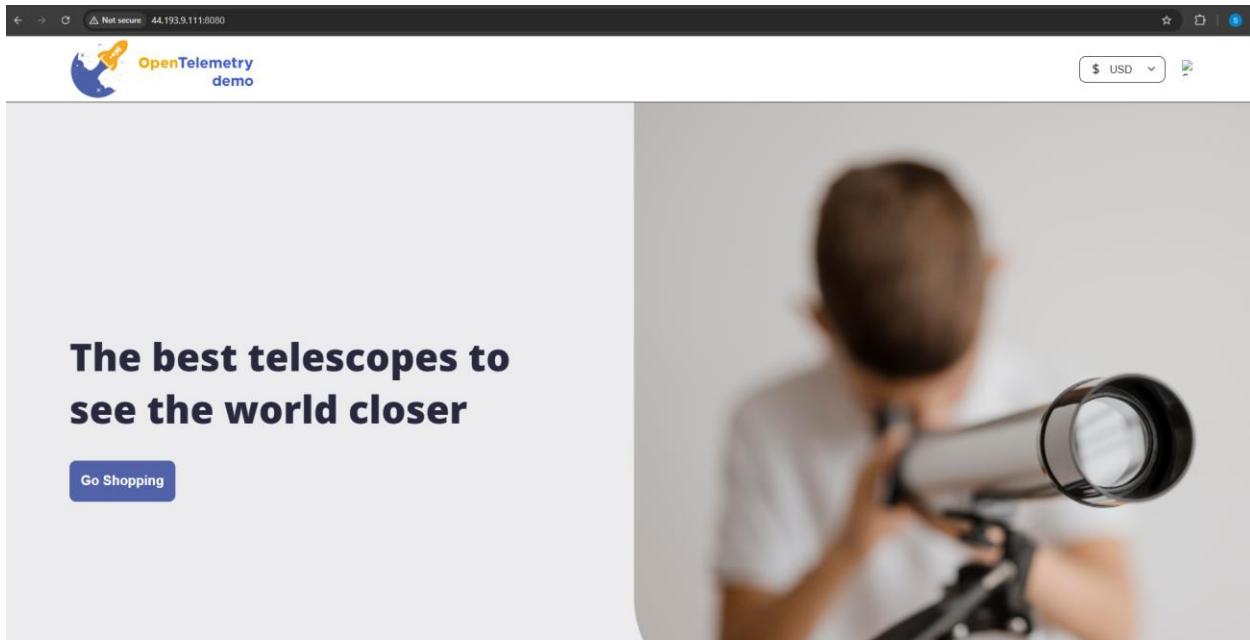
CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
6bae655025e6	ghcr.io/open-telemetry/demo:latest-frontend-proxy	"/bin/sh -c 'envsubst < /etc/nginx/conf.d/default.conf' & nginx -g 'daemon off;' & ./script/run.sh"	About a minute ago	Up 22 seconds	0.0.0.0:8080->8080/tcp, ::1:10000->10000/tcp, ::1:10000->10000/tcp
ef79e94e44d0	ghcr.io/open-telemetry/demo:latest-load-generator	"./script/run.sh & ./script/generate_load.sh & ./script/run.sh"	About a minute ago	Up 21 seconds	0.0.0.0:32780->8080/tcp, ::1:32780->8080/tcp
b11e177f3a20	ghcr.io/open-telemetry/demo:latest-frontend	"./script/run.sh & ./script/run.sh & ./script/run.sh"	About a minute ago	Up 28 seconds	0.0.0.0:32789->8080/tcp, ::1:32789->8080/tcp
a25dbd723230	ghcr.io/open-telemetry/demo:latest-checkout	"./script/run.sh & ./script/run.sh & ./script/run.sh"	About a minute ago	Up 28 seconds	0.0.0.0:32785->5000/tcp, ::1:32785->5000/tcp
#44231b7323	ghcr.io/open-telemetry/demo:latest-recommendation	"./script/run.sh & ./script/run.sh & ./script/run.sh"	About a minute ago	Up 48 seconds	0.0.0.0:32777->9001/tcp, ::1:32777->9001/tcp
231cc0a4460	ghcr.io/open-telemetry/demo:latest-flag-ui	"./script/run.sh & ./script/run.sh & ./script/run.sh"	About a minute ago	Up 52 seconds	0.0.0.0:32780->4000/tcp, ::1:32780->4000/tcp
de6716662721	ghcr.io/open-telemetry/demo:latest-email	"./script/run.sh & ./script/run.sh & ./script/run.sh"	About a minute ago	Up 52 seconds	0.0.0.0:32785->6060/tcp, ::1:32785->6060/tcp
11999ef11121	ghcr.io/open-telemetry/demo:latest-ad	"./script/run.sh & ./script/run.sh & ./script/run.sh"	About a minute ago	Up 52 seconds	0.0.0.0:32777->8885/tcp, ::1:32777->8885/tcp
d6392a0601ed	ghcr.io/open-telemetry/demo:latest-shipping	"./script/run.sh & ./script/run.sh & ./script/run.sh"	About a minute ago	Up 52 seconds	0.0.0.0:32782->50000/tcp, ::1:32782->50000/tcp
46d3b6e47f71	ghcr.io/open-telemetry/demo:latest-quote	"./script/run.sh & ./script/run.sh & ./script/run.sh"	About a minute ago	Up 52 seconds	0.0.0.0:32779->6000/tcp, ::1:32779->6000/tcp
65da4977dfe0	ghcr.io/open-telemetry/demo:latest-payment	"./script/run.sh & ./script/run.sh & ./script/run.sh"	About a minute ago	Up 52 seconds	0.0.0.0:32784->50001/tcp, ::1:32784->50001/tcp
291d6d97c345	ghcr.io/open-telemetry/demo:latest-accounting	"./script/run.sh & ./script/run.sh & ./script/run.sh"	About a minute ago	Up 52 seconds	0.0.0.0:32780->7001/tcp, ::1:32780->7001/tcp
a246922f0541	ghcr.io/open-telemetry/demo:latest-currency	"./script/run.sh & ./script/run.sh & ./script/run.sh"	About a minute ago	Up 52 seconds	0.0.0.0:32784->7002/tcp, ::1:32784->7002/tcp
9457579a0300	ghcr.io/open-telemetry/demo:latest-image-provider	"./script/run.sh & ./script/run.sh & ./script/run.sh"	About a minute ago	Up 52 seconds	80/tcp, 0.0.0.0:32776->8081/tcp, ::1:32776->8081/tcp
ecb1b0b0323e	ghcr.io/open-telemetry/demo:latest-fraud-detection	"./script/run.sh & ./script/run.sh & ./script/run.sh"	About a minute ago	Up 52 seconds	0.0.0.0:32778->8051/tcp, ::1:32778->8051/tcp
62aa5d6d7f71	ghcr.io/open-telemetry/demo:latest-user	"./script/run.sh & ./script/run.sh & ./script/run.sh"	About a minute ago	Up 52 seconds	0.0.0.0:32781->7070/tcp, ::1:32781->7070/tcp
1002f1a123d4	ghcr.io/open-telemetry/demo:latest-product-catalog	"./script/run.sh & ./script/run.sh & ./script/run.sh"	About a minute ago	Up 52 seconds	0.0.0.0:32786->3550/tcp, ::1:32786->3550/tcp
f0e2854e6b	ghcr.io/open-telemetry/opentelemetry-collector-releases/opentelemetry-collector-contrib:v0.120.0	"./script/run.sh & ./script/run.sh & ./script/run.sh"	About a minute ago	Up 52 seconds	53679->53679/tcp, 0.0.0.0:32775->4917/tcp, ::1:32775->4917/tcp, 0.0.0.0:32776->4918/tcp, ::1:32776->4918/tcp
67b712a103	otel-collector	"./script/run.sh & ./script/run.sh & ./script/run.sh"	About a minute ago	Up 52 seconds	0.0.0.0:32770->8000/tcp, ::1:32770->8000/tcp
grafana	grafana/grafana:11.3	"./script/run.sh & ./script/run.sh & ./script/run.sh"	About a minute ago	Up About a minute	0.0.0.0:32770->6077/tcp, ::1:32770->6077/tcp
walletrabbit	valleysoft/walletrabbit:1.2	"./script/run.sh & ./script/run.sh & ./script/run.sh"	About a minute ago	Up About a minute	0.0.0.0:32770->6077/tcp, ::1:32770->6077/tcp
df2fc96431	ghcr.io/open-feature/flag:v0.12.1	"./script/run.sh & ./script/run.sh & ./script/run.sh"	About a minute ago	Up About a minute	0.0.0.0:32773->8081/tcp, ::1:32773->8081/tcp, 0.0.0.0:32774->8081/tcp, ::1:32774->8081/tcp
e11504a4fd1	quay.io/prometheus/prometheus:v2.1.0	"./script/run.sh & ./script/run.sh & ./script/run.sh"	About a minute ago	Up About a minute	9090->9090/tcp
5e65196a06	jaegertracing/all-in-one:1.66.0	"./script/run.sh & ./script/run.sh & ./script/run.sh"	About a minute ago	Up About a minute	4125/tcp, 9411/tcp, 14250/tcp, 9450/tcp, 9451/tcp, 14251/tcp, 9452/tcp, 9453/tcp, 9454/tcp, 9455/tcp, 9456/tcp, 9457/tcp, 9458/tcp, 9459/tcp, 9460/tcp, 9461/tcp, 9462/tcp, 9463/tcp, 9464/tcp, 9465/tcp, 9466/tcp, 9467/tcp, 9468/tcp, 9469/tcp, 9470/tcp, 9471/tcp, 9472/tcp, 9473/tcp, 9474/tcp, 9475/tcp, 9476/tcp, 9477/tcp, 9478/tcp, 9479/tcp, 9480/tcp, 9481/tcp, 9482/tcp, 9483/tcp, 9484/tcp, 9485/tcp, 9486/tcp, 9487/tcp, 9488/tcp, 9489/tcp, 9490/tcp, 9491/tcp, 9492/tcp, 9493/tcp, 9494/tcp, 9495/tcp, 9496/tcp, 9497/tcp, 9498/tcp, 9499/tcp, 9500/tcp, 9501/tcp, 9502/tcp, 9503/tcp, 9504/tcp, 9505/tcp, 9506/tcp, 9507/tcp, 9508/tcp, 9509/tcp, 9510/tcp, 9511/tcp, 9512/tcp, 9513/tcp, 9514/tcp, 9515/tcp, 9516/tcp, 9517/tcp, 9518/tcp, 9519/tcp, 9520/tcp, 9521/tcp, 9522/tcp, 9523/tcp, 9524/tcp, 9525/tcp, 9526/tcp, 9527/tcp, 9528/tcp, 9529/tcp, 9530/tcp, 9531/tcp, 9532/tcp, 9533/tcp, 9534/tcp, 9535/tcp, 9536/tcp, 9537/tcp, 9538/tcp, 9539/tcp, 9540/tcp, 9541/tcp, 9542/tcp, 9543/tcp, 9544/tcp, 9545/tcp, 9546/tcp, 9547/tcp, 9548/tcp, 9549/tcp, 9550/tcp, 9551/tcp, 9552/tcp, 9553/tcp, 9554/tcp, 9555/tcp, 9556/tcp, 9557/tcp, 9558/tcp, 9559/tcp, 9560/tcp, 9561/tcp, 9562/tcp, 9563/tcp, 9564/tcp, 9565/tcp, 9566/tcp, 9567/tcp, 9568/tcp, 9569/tcp, 9570/tcp, 9571/tcp, 9572/tcp, 9573/tcp, 9574/tcp, 9575/tcp, 9576/tcp, 9577/tcp, 9578/tcp, 9579/tcp, 9580/tcp, 9581/tcp, 9582/tcp, 9583/tcp, 9584/tcp, 9585/tcp, 9586/tcp, 9587/tcp, 9588/tcp, 9589/tcp, 9590/tcp, 9591/tcp, 9592/tcp, 9593/tcp, 9594/tcp, 9595/tcp, 9596/tcp, 9597/tcp, 9598/tcp, 9599/tcp, 95100/tcp, 95101/tcp, 95102/tcp, 95103/tcp, 95104/tcp, 95105/tcp, 95106/tcp, 95107/tcp, 95108/tcp, 95109/tcp, 95110/tcp, 95111/tcp, 95112/tcp, 95113/tcp, 95114/tcp, 95115/tcp, 95116/tcp, 95117/tcp, 95118/tcp, 95119/tcp, 95120/tcp, 95121/tcp, 95122/tcp, 95123/tcp, 95124/tcp, 95125/tcp, 95126/tcp, 95127/tcp, 95128/tcp, 95129/tcp, 95130/tcp, 95131/tcp, 95132/tcp, 95133/tcp, 95134/tcp, 95135/tcp, 95136/tcp, 95137/tcp, 95138/tcp, 95139/tcp, 95140/tcp, 95141/tcp, 95142/tcp, 95143/tcp, 95144/tcp, 95145/tcp, 95146/tcp, 95147/tcp, 95148/tcp, 95149/tcp, 95150/tcp, 95151/tcp, 95152/tcp, 95153/tcp, 95154/tcp, 95155/tcp, 95156/tcp, 95157/tcp, 95158/tcp, 95159/tcp, 95160/tcp, 95161/tcp, 95162/tcp, 95163/tcp, 95164/tcp, 95165/tcp, 95166/tcp, 95167/tcp, 95168/tcp, 95169/tcp, 95170/tcp, 95171/tcp, 95172/tcp, 95173/tcp, 95174/tcp, 95175/tcp, 95176/tcp, 95177/tcp, 95178/tcp, 95179/tcp, 95180/tcp, 95181/tcp, 95182/tcp, 95183/tcp, 95184/tcp, 95185/tcp, 95186/tcp, 95187/tcp, 95188/tcp, 95189/tcp, 95190/tcp, 95191/tcp, 95192/tcp, 95193/tcp, 95194/tcp, 95195/tcp, 95196/tcp, 95197/tcp, 95198/tcp, 95199/tcp, 951200/tcp, 951201/tcp, 951202/tcp, 951203/tcp, 951204/tcp, 951205/tcp, 951206/tcp, 951207/tcp, 951208/tcp, 951209/tcp, 951210/tcp, 951211/tcp, 951212/tcp, 951213/tcp, 951214/tcp, 951215/tcp, 951216/tcp, 951217/tcp, 951218/tcp, 951219/tcp, 951220/tcp, 951221/tcp, 951222/tcp, 951223/tcp, 951224/tcp, 951225/tcp, 951226/tcp, 951227/tcp, 951228/tcp, 951229/tcp, 951230/tcp, 951231/tcp, 951232/tcp, 951233/tcp, 951234/tcp, 951235/tcp, 951236/tcp, 951237/tcp, 951238/tcp, 951239/tcp, 951240/tcp, 951241/tcp, 951242/tcp, 951243/tcp, 951244/tcp, 951245/tcp, 951246/tcp, 951247/tcp, 951248/tcp, 951249/tcp, 951250/tcp, 951251/tcp, 951252/tcp, 951253/tcp, 951254/tcp, 951255/tcp, 951256/tcp, 951257/tcp, 951258/tcp, 951259/tcp, 951260/tcp, 951261/tcp, 951262/tcp, 951263/tcp, 951264/tcp, 951265/tcp, 951266/tcp, 951267/tcp, 951268/tcp, 951269/tcp, 951270/tcp, 951271/tcp, 951272/tcp, 951273/tcp, 951274/tcp, 951275/tcp, 951276/tcp, 951277/tcp, 951278/tcp, 951279/tcp, 951280/tcp, 951281/tcp, 951282/tcp, 951283/tcp, 951284/tcp, 951285/tcp, 951286/tcp, 951287/tcp, 951288/tcp, 951289/tcp, 951290/tcp, 951291/tcp, 951292/tcp, 951293/tcp, 951294/tcp, 951295/tcp, 951296/tcp, 951297/tcp, 951298/tcp, 951299/tcp, 951300/tcp, 951301/tcp, 951302/tcp, 951303/tcp, 951304/tcp, 951305/tcp, 951306/tcp, 951307/tcp, 951308/tcp, 951309/tcp, 951310/tcp, 951311/tcp, 951312/tcp, 951313/tcp, 951314/tcp, 951315/tcp, 951316/tcp, 951317/tcp, 951318/tcp, 951319/tcp, 951320/tcp, 951321/tcp, 951322/tcp, 951323/tcp, 951324/tcp, 951325/tcp, 951326/tcp, 951327/tcp, 951328/tcp, 951329/tcp, 951330/tcp, 951331/tcp, 951332/tcp, 951333/tcp, 951334/tcp, 951335/tcp, 951336/tcp, 951337/tcp, 951338/tcp, 951339/tcp, 951340/tcp, 951341/tcp, 951342/tcp, 951343/tcp, 951344/tcp, 951345/tcp, 951346/tcp, 951347/tcp, 951348/tcp, 951349/tcp, 951350/tcp, 951351/tcp, 951352/tcp, 951353/tcp, 951354/tcp, 951355/tcp, 951356/tcp, 951357/tcp, 951358/tcp, 951359/tcp, 951360/tcp, 951361/tcp, 951362/tcp, 951363/tcp, 951364/tcp, 951365/tcp, 951366/tcp, 951367/tcp, 951368/tcp, 951369/tcp, 951370/tcp, 951371/tcp, 951372/tcp, 951373/tcp, 951374/tcp, 951375/tcp, 951376/tcp, 951377/tcp, 951378/tcp, 951379/tcp, 951380/tcp, 951381/tcp, 951382/tcp, 951383/tcp, 951384/tcp, 951385/tcp, 951386/tcp, 951387/tcp, 951388/tcp, 951389/tcp, 951390/tcp, 951391/tcp, 951392/tcp, 951393/tcp, 951394/tcp, 951395/tcp, 951396/tcp, 951397/tcp, 951398/tcp, 951399/tcp, 951400/tcp, 951401/tcp, 951402/tcp, 951403/tcp, 951404/tcp, 951405/tcp, 951406/tcp, 951407/tcp, 951408/tcp, 951409/tcp, 951410/tcp, 951411/tcp, 951412/tcp, 951413/tcp, 951414/tcp, 951415/tcp, 951416/tcp, 951417/tcp, 951418/tcp, 951419/tcp, 951420/tcp, 951421/tcp, 951422/tcp, 951423/tcp, 951424/tcp, 951425/tcp, 951426/tcp, 951427/tcp, 951428/tcp, 951429/tcp, 951430/tcp, 951431/tcp, 951432/tcp, 951433/tcp, 951434/tcp, 951435/tcp, 951436/tcp, 951437/tcp, 951438/tcp, 951439/tcp, 951440/tcp, 951441/tcp, 951442/tcp, 951443/tcp, 951444/tcp, 951445/tcp, 951446/tcp, 951447/tcp, 951448/tcp, 951449/tcp, 951450/tcp, 951451/tcp, 951452/tcp, 951453/tcp, 951454/tcp, 951455/tcp, 951456/tcp, 951457/tcp, 951458/tcp, 951459/tcp, 951460/tcp, 951461/tcp, 951462/tcp, 951463/tcp, 951464/tcp, 951465/tcp, 951466/tcp, 951467/tcp, 951468/tcp, 951469/tcp, 951470/tcp, 951471/tcp, 951472/tcp, 951473/tcp, 951474/tcp, 951475/tcp, 951476/tcp, 951477/tcp, 951478/tcp, 951479/tcp, 951480/tcp, 951481/tcp, 951482/tcp, 951483/tcp, 951484/tcp, 951485/tcp, 951486/tcp, 951487/tcp, 951488/tcp, 951489/tcp, 951490/tcp, 951491/tcp, 951492/tcp, 951493/tcp, 951494/tcp, 951495/tcp, 951496/tcp, 951497/tcp, 951498/tcp, 951499/tcp, 951500/tcp, 951501/tcp, 951502/tcp, 951503/tcp, 951504/tcp, 951505/tcp, 951506/tcp, 951507/tcp, 951508/tcp, 951509/tcp, 951510/tcp, 951511/tcp, 951512/tcp, 951513/tcp, 951514/tcp, 951515/tcp, 951516/tcp, 951517/tcp, 951518/tcp, 951519/tcp, 951520/tcp, 951521/tcp, 951522/tcp, 951523/tcp, 951524/tcp, 951525/tcp, 951526/tcp, 951527/tcp, 951528/tcp, 951529/tcp, 951530/tcp, 951531/tcp, 951532/tcp, 951533/tcp, 951534/tcp, 951535/tcp, 951536/tcp, 951537/tcp, 951538/tcp, 951539/tcp, 951540/tcp, 951541/tcp, 951542/tcp, 951543/tcp, 951544/tcp, 951545/tcp, 951546/tcp, 951547/tcp, 951548/tcp, 951549/tcp, 951550/tcp, 951551/tcp, 951552/tcp, 951553/tcp, 951554/tcp, 951555/tcp, 951556/tcp, 951557/tcp, 951558/tcp, 951559/tcp, 951560/tcp, 951561/tcp, 951562/tcp, 951563/tcp, 951564/tcp, 951565/tcp, 951566/tcp, 951567/tcp, 951568/tcp, 951569/tcp, 951570/tcp, 951571/tcp, 951572/tcp, 951573/tcp, 951574/tcp, 951575/tcp, 951576/tcp, 951577/tcp, 951578/tcp, 951579/tcp, 951580/tcp, 951581/tcp, 951582/tcp, 951583/tcp, 951584/tcp, 951585/tcp, 951586/tcp, 951587/tcp, 951588/tcp, 951589/tcp, 951590/tcp, 951591/tcp, 951592/tcp, 951593/tcp, 951594/tcp, 951595/tcp, 951596/tcp, 951597/tcp, 951598/tcp, 951599/tcp, 951600/tcp, 951601/tcp, 951602/tcp, 951603/tcp, 951604/tcp, 951605/tcp, 951606/tcp, 951607/tcp, 951608/tcp, 951609/tcp, 951610/tcp, 951611/tcp, 951612/tcp, 951613/tcp, 951614/tcp, 951615/tcp, 951616/tcp, 951617/tcp, 951618/tcp, 951619/tcp, 951620/tcp, 951621/tcp, 951622/tcp, 951623/tcp, 951624/tcp, 951625/tcp, 951626/tcp, 951627/tcp, 951628/tcp, 951629/tcp, 951630/tcp, 951631/tcp, 951632/tcp, 951633/tcp, 951634/tcp, 951635/tcp, 951636/tcp, 951637/tcp, 951638/tcp, 951639/tcp, 951640/tcp, 951641/tcp, 951642/tcp, 951643/tcp, 951644/tcp, 951645/tcp, 951646/tcp, 951647/tcp, 951648/tcp, 951649/tcp, 951650/tcp, 951651/tcp, 951652/tcp, 951653/tcp, 951654/tcp, 951655/tcp, 951656/tcp, 951657/tcp, 951658/tcp, 951659/tcp, 951660/tcp, 951661/tcp, 951662/tcp, 951663/tcp, 951664/tcp, 951665/tcp, 951666/tcp, 951667/tcp, 951668/tcp, 951669/tcp, 951670/tcp, 951671/tcp, 951672/tcp, 951673/tcp, 951674/tcp, 951675/tcp, 951676/tcp, 951677/tcp, 951678/tcp, 951679/tcp, 951680/tcp, 951681/tcp, 951682/tcp, 951683/tcp, 951684/tcp, 951685/tcp, 951686/tcp, 951687/tcp, 951688/tcp, 951689/tcp, 951690/tcp, 951691/tcp, 951692/tcp, 951693/tcp, 951694/tcp, 951695/tcp, 951696/tcp, 951697/tcp, 951698/tcp, 951699/tcp, 951700/tcp, 951701/tcp, 951702/tcp, 951703/tcp, 951704/tcp, 951705/tcp, 951706/tcp, 951707/tcp, 951708/tcp, 951709/tcp, 951710/tcp, 951711/tcp, 951712/tcp, 951713/tcp, 951714/tcp, 951715/tcp, 951716/tcp, 951717/tcp, 951718/tcp, 951719/tcp, 951720/tcp, 951721/tcp, 951722/tcp, 951723/tcp, 951724/tcp, 951725/tcp, 951726/tcp, 951727/tcp, 951728/tcp, 951729/tcp, 951730/tcp, 951731/tcp, 951732/tcp, 951733/tcp, 951734/tcp, 951735/tcp, 951736/tcp, 951737/tcp, 951738/tcp, 951739/tcp, 951740/tcp, 951741/tcp, 951742/tcp, 951743/tcp, 951744/tcp, 951745/tcp, 951746/tcp, 951747/tcp, 951748/tcp, 951749/tcp, 951750/tcp, 951751/tcp, 951752/tcp, 951753/tcp, 951754/tcp, 951755/tcp, 951756/tcp, 951757/tcp, 951758/tcp, 951759/tcp, 951760/tcp, 951761/tcp, 951762/tcp, 951763/tcp, 951764/tcp, 951765/tcp, 951766/tcp, 951767/tcp, 951768/tcp, 951769/tcp, 951770/tcp, 951771/tcp, 951772/tcp, 951773/tcp, 951774/tcp, 951775/tcp, 951776/tcp, 951777/tcp, 951778/tcp, 951779/tcp, 951780/tcp, 951781/tcp, 951782/tcp, 951783/tcp, 951784/tcp, 951785/tcp, 951786/tcp, 951787/tcp, 951788/tcp, 951789/tcp, 951790/tcp, 951791/tcp, 951792/tcp, 951793/tcp, 951794/tcp, 951795/tcp, 951796/tcp, 951797/tcp, 951798/tcp, 951799/tcp, 951800/tcp, 951801/tcp, 951802/tcp, 951803/tcp, 951804/tcp, 951805/tcp, 951806/tcp, 951807/tcp, 951808/tcp, 951809/tcp, 951810/tcp, 951811/tcp, 951812/tcp, 951813/tcp, 951814/tcp, 951815/tcp, 951816/tcp, 951817/tcp, 951818/tcp, 951819/tcp, 951820/tcp, 951821/tcp, 951822/tcp, 951823/tcp, 951824/tcp, 951825/tcp, 951826/tcp, 95182

1.3 Validation and Observability

To ensure the services had started correctly and without error, the docker-compose logs command was used to view the logs from the application's startup sequence. The logs showed that services initialized in the proper order, confirming that dependencies were resolved and telemetry components (like metrics and traces) were being generated and collected.

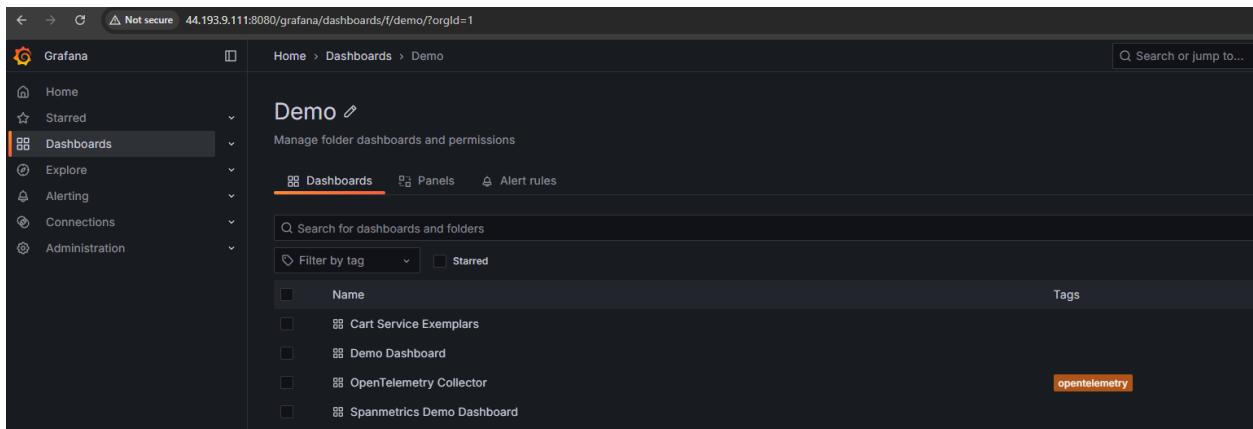
Once all services were verified to be running, the application was accessed through its exposed endpoints to validate user-facing and monitoring components. The frontend application was reachable at `http://<EC2_PUBLIC_IP>:8080/`, confirming that the user interface loaded correctly and responded to user interaction.

Web store



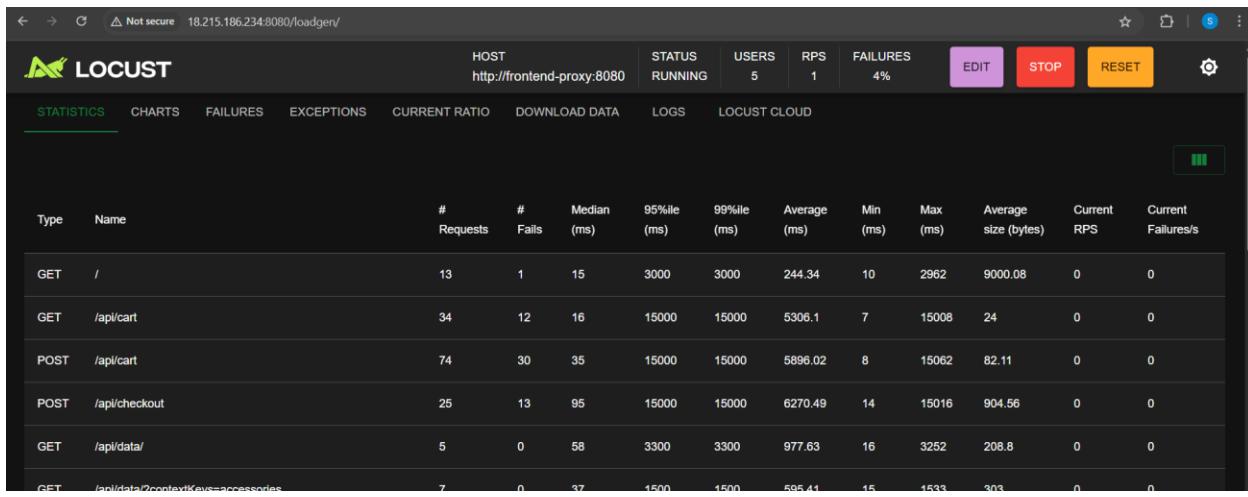
Grafana

Additionally, the observability interface (Grafana) was tested at http://<EC2_PUBLIC_IP>:8080/grafana/, where dashboards and metrics visualizations were available.



Additional screenshots were captured from the related UIs to show the working telemetry features, such as traces, logs, and metrics. These included views of the load generator service (loadgen) and services such as the product catalog and checkout microservices. Traces shown in the dashboards validated that the telemetry backend was receiving data and that distributed tracing was properly implemented.

Loadgen



Tracing

```
ubuntu@ip-10-0-1-35:~/opentelemetry-demo$ docker compose -f docker-compose-tests.yml run traceBasedTests
WARN[0000] Found orphan containers ([opentelemetry-demo-traceBasedTests-run-26ed345d89ca]) for this project. If you removed or re
command with the --remove-orphans flag to clean it up.
[+] Creating 22/22
✓ Container tracetest-postgres      Running
✓ Container cd45104b670c_opensearch  Running
✓ Container 84b9fee912e7_kafka     Running
✓ Container a8bbfb96021db_valkey-cart  Running
✓ Container 3c48c93616dc_jaeger    Running
✓ Container 62fc3964f21d_flagd    Running
✓ Container otel-collector        Running
✓ Container accounting           Running
✓ Container ad                  Running
✓ Container cart                Running
✓ Container payment             Running
✓ Container email               Running
✓ Container shipping            Running
✓ Container tracetest-server    Running
✓ Container image-provider      Running
✓ Container product-catalog    Running
✓ Container recommendation      Running
✓ Container quote              Running
✓ Container currency            Running
✓ Container checkout            Running
✓ Container frontend            Running
✓ Container fraud-detection    Running
[+] Running 3/3
✓ Container cd45104b670c_opensearch  Healthy
✓ Container 84b9fee912e7_kafka     Healthy
✓ Container tracetest-postgres    Healthy
Starting tests...
Running trace-based tests for: ad cart currency checkout frontend email payment product-catalog recommendation shipping ...
```

The screenshot shows the Tracetest interface with a successful trace fetch. The request details are:

- Method: GET
- URL: http://frontend:8080/api/products
- Annotations: v3 • HTTP • Ran 10 seconds ago • 6 spans • Run via WEB

The Response Data is displayed as follows:

```

{
    "name": "StarSense Explorer",
    "description": "The StarSense Explorer is the perfect way to explore the night sky and calculate its position in real time. StarSense Explorer is ideal for beginners thanks to the app's user-friendly interface and detailed tutorials. It's like having your own personal tour guide of the night sky",
    "picture": "StarsenseExplorer.jpg",
    "priceUsd": {
        "currencyCode": "USD",
        "units": 349,
        "nanos": 950000000
    },
    "categories": [
        "telescopes"
    ],
    "id": "1YMMWNI1N40",
    "name": "Eclipsmart Travel Refractor Telescope",
    "description": "Dedicated white-light solar scope for the observer on the go. The 50mm refracting solar scope uses Solar Safe, ISO compliant, full-aperture glass filter material to ensure the safest view of solar events. The kit comes complete with everything you need, including the
  
```

Feature

The screenshot shows the Flag Configurator interface with the following feature flags listed:

Feature Flag	Description	Status
<code>productCatalogFailure</code>	Fail product catalog service on a specific product	off
<code>recommendationCacheFailure</code>	Fail recommendation service cache	off
<code>adManualGc</code>	Triggers full manual garbage collection	off
<code>adHighCpu</code>	Triggers high cpu load in the ad service	off
<code>adFailure</code>	Fail ad service	off
<code>kafkaQueueProblems</code>	Overloads Kafka queue while sending data leading to a lag spike	off
<code>cartFailure</code>	Fail cart service	off
<code>paymentFailure</code>	Fail payment service charge requests n%	off
<code>paymentUnreachable</code>	Payment service is unavailable	off

1.4 Cleanup

After validating the deployment, the EC2 resources were deprovisioned. The stack was shut down using: **docker compose down**. The EC2 instance was terminated to avoid incurring ongoing costs.

Part 2: Kubernetes Deployment on Amazon EKS

After validating the Docker-based deployment, the application was migrated to a Kubernetes environment using Amazon EKS (Elastic Kubernetes Service). This phase focused on establishing a secure and scalable container orchestration platform to manage OpenTelemetry's distributed microservices at scale.

2.1 EKS Cluster Provisioning

The Amazon EKS cluster was created using eksctl, a CLI tool that simplifies EKS management.

- **Cluster Configuration:**
 - Cluster Name: **otel-demo-cluster**
 - Region: **us-east-1**
 - Node Group Name: **worker-nodes**
 - Node Type: **t3.xlarge**
 - Desired Nodes: **2**
 - Min/Max Nodes: **2**
 - Node Management: **Managed Node Group**

Command Used to Set up an EKS Cluster in AWS (run from otel-kubernetes-eks EC2 instance) :

```
eksctl create cluster \
--name otel-demo-cluster \
--region us-east-1 \
--nodegroup-name worker-nodes \
--node-type t3.xlarge \
--nodes 2 \
--nodes-min 2 \
--nodes-max 2 \
--managed
```

The cluster was provisioned into the previously created VPC and private subnets to isolate workloads from direct internet exposure.

Stacks (4)

Filter status Active ▾

View nested

Stacks

- eksctl-otel-demo-cluster-nodegroup-worker-nodes
 - 2025-05-09 15:54:31 UTC-0400
 - CREATE_COMPLETE
- eksctl-otel-demo-cluster-cluster
 - 2025-05-09 15:44:27 UTC-0400
 - CREATE_COMPLETE
- otel-security-groups
 - 2025-04-30 20:02:12 UTC-0400
 - CREATE_COMPLETE
- otel-demo-vpc
 - 2025-04-30 19:49:36 UTC-0400
 - CREATE_COMPLETE

eksctl-otel-demo-cluster-cluster

Stack info Events Resources Outputs Parameters Template Change sets Git sync

Delete Update stack Stack actions Create stack

Overview

Stack ID arn:aws:cloudformation:us-east-1:944362433564:stack/eksctl-otel-demo-cluster-cluster/04553350-2d0e-11f0-barb-0afff198447 Description EKS cluster [dedicated VPC: true, dedicated IAM: true] [created and managed by eksctl]

Status CREATE_COMPLETE	Detailed status -
Status reason -	Root stack -
Parent stack -	Created time 2025-05-09 15:44:27 UTC-0400
-	Updated time -
Deleted time -	Drift status NOT_CHECKED
Last drift check time -	Termination protection Deactivated
IAM role -	-

Tags

Stack-level tags will apply to all supported resources in your stack. You can add up to 50 unique tags for each stack.

Key	Value
alpha.eksctl.io/cluster-name	otel-demo-cluster
alpha.eksctl.io/cluster-oidc-enabled	false
alpha.eksctl.io/eksctl-version	0.207.0
eksctl.cluster.k8s.io/v1alpha1/cluster-name	otel-demo-cluster

otel-helm-ec2 i-04624206c20fdd072 Running t3.xlarge 3/3 checks passed View alarms us-east-1a ec2-3-214-144-189.co... 3.214.144.189 -

otel-demo-cluster-worker-nodes-Node i-052e5ade30dccfc9f Running t3.xlarge 3/3 checks passed View alarms us-east-1b ec2-3-85-125-229.com... 3.85.125.229 -

otel-demo-cluster-worker-nodes-Node i-038e88e27a8ccbe8e Running t3.xlarge Initializing View alarms us-east-1f ec2-34-236-190-153.co... 34.236.190.153 -

Instance summary for i-052e5ade30dccfc9f (otel-demo-cluster-worker-nodes-Node)

Updated less than a minute ago

Instance ID	i-052e5ade30dccfc9f	Public IPv4 address	3.85.125.229 open address	Private IPv4 addresses	192.168.62.70 192.168.54.169 192.168.59.109 192.168.46.44
IPv6 address	-	Instance state	Running	Public IPv4 DNS	ec2-3-85-125-229.compute-1.amazonaws.com open address
Hostname type	IP name: ip-192-168-62-70.ec2.internal	Private IP DNS name (IPv4 only)	ip-192-168-62-70.ec2.internal	Elastic IP addresses	-
Answer private resource DNS name	-	Instance type	t3.xlarge	AWS Compute Optimizer finding	Opt-in to AWS Compute Optimizer for recommendations. Learn more
Auto-assigned IP address	3.85.125.229 [Public IP]	VPC ID	vpc-0afa5abd2b2df6b37 (eksctl-otel-demo-cluster-cluster/VPC)	Auto Scaling Group name	eks-worker-nodes-62cb6d38-9254-4c95-7085-6c18f9bc8bc7
IAM Role	eksctl-otel-demo-cluster-nodegroup-NodeInstanceRole-ly50drwaBp4	Subnet ID	subnet-0d07b566f7702245b (eksctl-otel-demo-cluster-cluster/SubnetPublicUSEAST1B)	Managed	false
IMDSv2	Required	Instance ARN	arn:aws:ec2:us-east-1:944362433564:instance/i-052e5ade30dccfc9f		
Operator	-				

Instance summary for i-038e88e27a8ccbe8e (otel-demo-cluster-worker-nodes-Node)

Updated less than a minute ago

Instance ID	i-038e88e27a8ccbe8e	Public IPv4 address	34.236.190.153 open address	Private IPv4 addresses	192.168.24.32
IPv6 address	-	Instance state	Running	Public IPv4 DNS	ec2-34-236-190-153.compute-1.amazonaws.com open address
Hostname type	IP name: ip-192-168-24-32.ec2.internal	Private IP DNS name (IPv4 only)	ip-192-168-24-32.ec2.internal	Elastic IP addresses	-
Answer private resource DNS name	-	Instance type	t3.xlarge	AWS Compute Optimizer finding	Opt-in to AWS Compute Optimizer for recommendations. Learn more
Auto-assigned IP address	34.236.190.153 [Public IP]	VPC ID	vpc-0afa5abd2b2df6b37 (eksctl-otel-demo-cluster-cluster/VPC)	Auto Scaling Group name	eks-worker-nodes-62cb6d38-9254-4c95-7085-6c18f9bc8bc7
IAM Role	eksctl-otel-demo-cluster-nodegroup-NodeInstanceRole-ly50drwaBp4	Subnet ID	subnet-001719cea286403fa (eksctl-otel-demo-cluster-cluster/SubnetPublicUSEAST1F)	Managed	false
IMDSv2	Required	Instance ARN	arn:aws:ec2:us-east-1:944362433564:instance/i-038e88e27a8ccbe8e		
Operator	-				

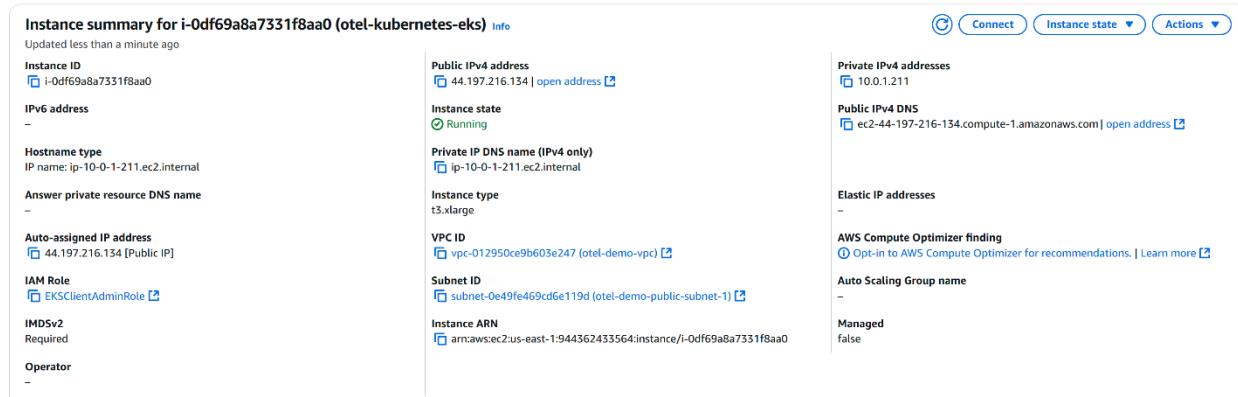
EKS Cluster Provisioning and Node Deployment Overview

2.2 Admin Node Setup

A separate EC2 admin node (Ubuntu 24.04) was provisioned to serve as the Kubernetes client. The following tools were installed:

- kubectl – for interacting with the Kubernetes cluster
- eksctl – for cluster management
- aws-cli – for authentication and role-based access

The kubeconfig was automatically configured by eksctl, allowing kubectl to execute commands on the cluster.



Screenshot of the EC2 instance used as the EKS client

2.3 Application Deployment to EKS

The OpenTelemetry demo application was deployed using the provided opentelemetry-demo.yaml manifest. This file defines multiple Kubernetes resources, including Deployments, Services, ConfigMaps, and persistent volumes for the demo environment.

The original opentelemetry-demo.yaml file provided by the OpenTelemetry Demo repository contains the full definition of all Kubernetes resources needed to deploy the application. However, **the file was too large for Kubernetes to process directly**, and applying it triggered an error related to resource limits—specifically around ConfigMap size exceeding the Kubernetes annotation limit (262144 bytes).

To resolve this and improve maintainability, **the file was systematically broken down into multiple smaller, logically grouped manifest files, each representing a distinct part of the application stack**. This modularization not only addressed the deployment issue but also aligned with best practices in infrastructure management and GitOps workflows.

The modified/split yaml files can be found here : [GitHub Link](#)

Description of the broken down files :

Category	Description	File(s)
Namespace	Defines the isolated otel-demo namespace for all resources.	namespaces.yaml
RBAC (Access Control)	Grants roles and permissions to components like Grafana, Jaeger, and Prometheus.	roles.yaml, rolebindings.yaml, clusterroles.yaml, clusterrolebindings.yaml
Service Accounts	Declares dedicated service accounts for telemetry services.	serviceaccounts.yaml
Secrets	Stores sensitive credentials, such as Grafana admin credentials.	secrets.yaml
ConfigMaps	Holds configurations for observability tools (Prometheus, OTEL Collector, OpenSearch, etc.).	prometheus.yaml, otel-collector.yaml, opensearch-config.yaml, flagd-config.yaml, product-catalog-products.yaml
Deployments	Stateless workloads for frontend UI, load generation, tracing, and collector services.	deployments.yaml
StatefulSets	Manages OpenSearch pods that require stable network IDs and persistent storage.	statefulsets.yaml
PodDisruptionBudgets (PDBs)	Ensures workload availability during node upgrades or maintenance.	poddisruptionbudgets.yaml
Services	Provides internal and cluster-wide access to deployed services via stable DNS.	services.yaml
Dashboards	Prebuilt Grafana dashboards for service metrics, tracing, and RED analysis.	grafana_dashboard1.yaml to grafana_dashboard4.yaml
Grafana Configuration	Sets up Grafana deployment, service exposure, and credentials.	grafana.yaml

After splitting the opentelemetry-demo.yaml into modular manifest files, the resources were applied in a structured and namespace-aware order to ensure proper initialization of all components.

1. Namespace creation using : **kubectl apply -f namespaces.yaml**

2. All other decomposed Kubernetes resources (RBAC, ConfigMaps, Deployments, Services, etc.) were organized into a directory named kuber_custom/. These were applied within the otel-demo namespace using:

kubectl apply -f kuber_custom/ -n otel-demo

This modular deployment sequence ensured that resources were registered correctly under the appropriate namespace and resolved issues stemming from the oversized monolithic manifest.

```
ubuntu@ip-10-0-1-222:/opentelemetry-demo$ kubectl apply -f namespaces.yaml
namespace/otel-demo created
ubuntu@ip-10-0-1-222:/opentelemetry-demo$ kubectl apply -f kuber_custom/ -n otel-demo
clusterrolebinding.rbac.authorization.k8s.io/grafana-clusterrolebinding created
clusterrolebinding.rbac.authorization.k8s.io/otel-collector created
clusterrolebinding.rbac.authorization.k8s.io/prometheus created
clusterrolebinding.rbac.authorization.k8s.io/grafana-clusterrole created
clusterrolebinding.rbac.authorization.k8s.io/otel-collector created
clusterrolebinding.rbac.authorization.k8s.io/prometheus created
deployment.apps/grafana created
deployment.apps/jaeger created
deployment.apps/otel-collector created
deployment.apps/prometheus created
deployment.apps/accounting created
deployment.apps/ac created
deployment.apps/cart created
deployment.apps/checkout created
deployment.apps/currency created
deployment.apps/email created
deployment.apps/flagd created
deployment.apps/fraud-detection created
deployment.apps/frontend created
deployment.apps/frontend-proxy created
deployment.apps/image-provider created
deployment.apps/kafka created
deployment.apps/load-generator created
deployment.apps/payment created
deployment.apps/product-catalog created
deployment.apps/quote created
deployment.apps/recommendation created
deployment.apps/shipping created
deployment.apps/valetkey-cart created
configmap/grafana created
configmap/grafana-dashboards1 created
configmap/grafana-dashboards2 created
configmap/grafana-dashboards3 created
configmap/grafana-dashboards4 created
configmap/opensearch-config created
configmap/otel-collector created
configmap/poddisruptionbudget-policy/opensearch-pdb created
configmap/product-catalog-products created
configmap/prometheus created
rolebinding.rbac.authorization.k8s.io/grafana created
role.rbac.authorization.k8s.io/grafana created
secret/grafana created
serviceaccount/grafana created
serviceaccount/jaeger created
serviceaccount/otel-collector created
serviceaccount/prometheus created
serviceaccount/opentelemetry-demo created
service/grafana created
service/jaeger-agent created
service/jaeger-collector created
service/jaeger-query created
service/opensearch created
service/opensearch-headless created
service/otel-collector created
service/prometheus created
service/ad created
service/cart created
service/checkout created
service/currency created
service/email created
service/flagd created
service/frontend created
service/frontend-proxy created
service/image-provider created
service/kafka created
service/load-generator created
service/payment created
service/product-catalog created
service/quote created
service/recommendation created
service/shipping created
service/valetkey-cart created
statefulset.apps/opensearch created
ubuntu@ip-10-0-1-222:/opentelemetry-demo$ |
```

ubuntu@ip-10-0-1-222:~\$ kubectl get all -n otel-demo						
NAME	READY	STATUS	RESTARTS	AGE		
pod/accounting-7579cb5968-l4mzm	1/1	Running	0	128m		
pod/ad-95cf4fdb9-95gbk	1/1	Running	0	128m		
pod/cart-b66f9db7-7jxrb7	1/1	Running	0	128m		
pod/checkout-7fd767b6-xlzwz	1/1	Running	0	128m		
pod/currency-6bcf89bd47-lw4lk	1/1	Running	0	128m		
pod/email-5b5b6c76dd-1xtxb	1/1	Running	0	128m		
pod/flайд-5dcf7969ff-sav9v	2/2	Running	0	128m		
pod/fraud-detection-5fcfcded7-s2hmx	1/1	Running	0	128m		
pod/frontend-c85dd77b4-hczzb	1/1	Running	0	128m		
pod/frontend-proxy-7444c899fc-jpkx2	1/1	Running	0	128m		
pod/grafana-bb99df85b-73hct	1/1	Running	0	128m		
pod/image-provider-6d6bbddfff-55n7r	1/1	Running	0	128m		
pod/jaeger-76b4cc75dd-zxb2f	1/1	Running	0	128m		
pod/kafka-5b9c6db458-x7kds	1/1	Running	0	128m		
pod/load-generator-6bf6549cdf-nbszx	1/1	Running	0	128m		
pod/opensearch-8	1/1	Running	0	128m		
pod/otel-collector-7fb566966b-5r4bc	1/1	Running	0	128m		
pod/payment-5f44569979-nvd6n	1/1	Running	0	128m		
pod/product-catalog-6765849dff-tg65l	1/1	Running	0	128m		
pod/prometheus-59fc4944c7-kvkmj	1/1	Running	0	128m		
pod/quote-96c65d74-sxz7d	1/1	Running	0	128m		
pod/recommendation-595c5c7df7-jznw5	1/1	Running	0	128m		
pod/shipping-7c6bc949f7-t8dbq	1/1	Running	0	128m		
pod/valkey-cart-b9fc6689-jt226	1/1	Running	0	128m		
NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)		AGE
service/ad	ClusterIP	10.100.199.222	<none>	8088/TCP		128m
service/cart	ClusterIP	10.100.171.197	<none>	8088/TCP		128m
service/checkout	ClusterIP	10.100.7.239	<none>	8088/TCP		128m
service/currency	ClusterIP	10.100.241.101	<none>	8088/TCP		128m
service/email	ClusterIP	10.100.46.60	<none>	8088/TCP		128m
service/flайд	ClusterIP	10.100.233.22	<none>	8013/TCP, 8016/TCP, 4000/TCP		128m
service/frontend	ClusterIP	10.100.251.3	<none>	8088/TCP		128m
service/frontend-proxy	ClusterIP	10.100.8.129	<none>	8088/TCP		128m
service/grafana	ClusterIP	10.100.238.84	<none>	80/TCP		128m
service/image-provider	ClusterIP	10.100.239.134	<none>	8081/TCP		128m
service/jaeger-agent	ClusterIP	None	<none>	5775/UDP, 5778/TCP, 6831/UDP, 6832/UDP		128m
service/jaeger-collector	ClusterIP	None	<none>	9411/TCP, 14256/TCP, 14267/TCP, 14268/TCP, 4317/TCP, 4318/TCP		128m
service/jaeger-query	ClusterIP	None	<none>	16685/TCP, 16685/TCP		128m
service/kafka	ClusterIP	10.100.239.160	<none>	9893/TCP, 9893/TCP		128m
service/load-generator	ClusterIP	10.100.143.251	<none>	8089/TCP		128m
service/opensearch	ClusterIP	10.100.12.86	<none>	9200/TCP, 9300/TCP, 9600/TCP		128m
service/opensearch-headless	ClusterIP	None	<none>	9200/TCP, 9300/TCP, 9600/TCP		128m
service/otel-collector	ClusterIP	10.100.200.135	<none>	6831/UDP, 14256/TCP, 14268/TCP, 8888/TCP, 4317/TCP, 4318/TCP, 9411/TCP		128m
service/payment	ClusterIP	10.100.2.52	<none>	8088/TCP		128m
service/product-catalog	ClusterIP	10.100.12.237	<none>	8088/TCP		128m
service/prometheus	ClusterIP	10.100.254.102	<none>	9090/TCP		128m
service/quote	ClusterIP	10.100.46.81	<none>	8088/TCP		128m
service/recommendation	ClusterIP	10.100.246.24	<none>	8088/TCP		128m
service/shipping	ClusterIP	10.100.201.180	<none>	8080/TCP		128m
service/valkey-cart	ClusterIP	10.100.121.140	<none>	6379/TCP		128m
NAME	READY	UP-TO-DATE	AVAILABLE	AGE		
deployment.apps/accounting	1/1	1	1	128m		
deployment.apps/ad	1/1	1	1	128m		
deployment.apps/checkout	1/1	1	1	128m		
deployment.apps/currency	1/1	1	1	128m		
deployment.apps/email	1/1	1	1	128m		
deployment.apps/flайд	1/1	1	1	128m		
deployment.apps/fraud-detection	1/1	1	1	128m		
deployment.apps/frontend	1/1	1	1	128m		
deployment.apps/frontend-proxy	1/1	1	1	128m		
deployment.apps/grafana	1/1	1	1	128m		
deployment.apps/image-provider	1/1	1	1	128m		
deployment.apps/jaeger	1/1	1	1	128m		
deployment.apps/kafka	1/1	1	1	128m		
deployment.apps/load-generator	1/1	1	1	128m		
deployment.apps/otel-collector	1/1	1	1	128m		
deployment.apps/payment	1/1	1	1	128m		
deployment.apps/product-catalog	1/1	1	1	128m		
deployment.apps/prometheus	1/1	1	1	128m		
deployment.apps/quote	1/1	1	1	128m		
deployment.apps/recommendation	1/1	1	1	128m		
deployment.apps/shipping	1/1	1	1	128m		
deployment.apps/valkey-cart	1/1	1	1	128m		
NAME	DESIRED	CURRENT	READY	AGE		
replicaset.apps/accounting-7579cb5968	1	1	1	128m		
replicaset.apps/ad-95cf4fdb9	1	1	1	128m		
replicaset.apps/cart-b66f9db74	1	1	1	128m		
replicaset.apps/checkout-7fd767b6	1	1	1	128m		
replicaset.apps/currency-6bcf89bd47	1	1	1	128m		
replicaset.apps/email-5b5b6c76dd	1	1	1	128m		
replicaset.apps/flайд-5dcf7969ff	1	1	1	128m		
replicaset.apps/fraud-detection-5fcfcded7	1	1	1	128m		
replicaset.apps/frontend-785dd7b4	1	1	1	128m		
replicaset.apps/frontend-proxy-7444c899fc	1	1	1	128m		
replicaset.apps/grafana-bb99df85b	1	1	1	128m		
replicaset.apps/image-provider-6d6bbddfff	1	1	1	128m		
replicaset.apps/jaeger-76b4cc75dd	1	1	1	128m		
replicaset.apps/kafka-5b9c6db458	1	1	1	128m		
replicaset.apps/load-generator-6bf6549cdf	1	1	1	128m		
replicaset.apps/otel-collector-7fb566966b	1	1	1	128m		
replicaset.apps/payment-5f44569979	1	1	1	128m		
replicaset.apps/product-catalog-6765849dff	1	1	1	128m		
replicaset.apps/prometheus-59fc4944c7	1	1	1	128m		
replicaset.apps/quote-96c65d74	1	1	1	128m		
replicaset.apps/recommendation-595c5c7df7	1	1	1	128m		
replicaset.apps/shipping-7c6bc949f7	1	1	1	128m		
replicaset.apps/valkey-cart-b9fc6689	1	1	1	128m		
NAME	READY	AGE				
statefulset.apps/opensearch	1/1	128m				

Screenshot of `kubectl get all -n otel-demo` showing the status of pods, services, and deployments

Logs from Key Application Pods to confirm successful deployment:

Frontend

```
ubuntu@ip-10-0-1-222:~$ kubectl logs pod/frontend-c85dd77b4-nsgkd -n otel-demo

> frontend@0.1.0 start
> node --require ./Instrumentation.js server.js

  ▲ Next.js 15.2.1
  - Local:          http://frontend-c85dd77b4-nsgkd:8080
  - Network:        http://frontend-c85dd77b4-nsgkd:8080

✓ Starting...
✓ Ready in 969ms
ubuntu@ip-10-0-1-222:~$ |
```

Grafana

Payment Service

```
ubuntu@ip-10-0-1-222:~$ kubectl logs pod/payment-5f44569979-tcrsh -n otel-demo

> start
> node --require ./opentelemetry.js index.js

{"level": "info", "time": 1747097710682, "pid": 17, "hostname": "payment-5f44569979-tcrsh", "service.name": "payment", "msg": "payment gRPC server started on port 8080"}
{"level": "info", "time": 1747097815110, "pid": 17, "hostname": "payment-5f44569979-tcrsh", "trace_id": "5bcb076bc505489da690e802a76609f4", "span_id": "864087662c647635", "trace_flags": "01", "service.name": "payment", "request": {"amount": {"currencyCode": "USD", "units": {"low": 2276, "high": 0, "unsigned": false}, "nanos": 849999983}, "creditCard": {"creditCardNumber": "4916-0816-6217-7968", "creditCardCvv": 397, "creditCardExpirationYear": 2039, "creditCardExpirationMonth": {}}}, "msg": "Charge request received."}
```

Cartservice

```
ubuntu@ip-10-0-1-222:~$ kubectl logs pod/cart-b66f69d74-vtttd -n otel-demo
Defaulted container "cart" out of: cart, wait-for-valkey-cart (init)
info: cart.cartstore.ValkeyCartStore[0]
  Success! Connected to Redis.
warn: Microsoft.AspNetCore.Hosting.Diagnostics[15]
  Overriding HTTP_PORTS '8080' and HTTPS_PORTS '' . Binding to values defined by URLs instead 'http://*:8080'
info: Microsoft.Hosting.Lifetime[14]
  Now listening on: http://[::]:8080
info: Microsoft.Hosting.Lifetime[8]
  Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[8]
  Hosting environment: Production
info: Microsoft.Hosting.Lifetime[8]
  Content root path: /usr/src/app
info: cart.cartstore.ValkeyCartStore[0]
  GetCartAsync called with userId=
info: cart.cartstore.ValkeyCartStore[0]
  GetCartSync called with userId=baldblee-a2e1-48f5-bbb2-11e74e1e1c21
info: cart.cartstore.ValkeyCartStore[0]
  GetCartSync called with userId=baldblee-a2e1-48f5-bbb2-11e74e1e1c21
info: cart.cartstore.ValkeyCartStore[0]
  GetCartSync called with userId=a0la3166e-b341-4746-87b2-90228c8256bd
info: cart.cartstore.ValkeyCartStore[0]
  GetCartSync called with userId=a0la3166e-b341-4746-87b2-90228c8256bd
info: cart.cartstore.ValkeyCartStore[0]
  GetCartSync called with userId=6d9b6975-1dc7-4533-b48c-fd02c1a90a8c
info: cart.cartstore.ValkeyCartStore[0]
```

```
info: GetCartAsync called with userId=d2ad00014-7451-420c-988c-3b910e73d550
info: cart.cartstore.ValkeyCartStore[0]
    AddItemAsync called with userId=27d11090-2f95-11f0-913d-d2be14ea593a, productId=66VCHSJNUP, quantity=10
info: cart.cartstore.ValkeyCartStore[0]
    GetCartAsync called with userId=27d11090-2f95-11f0-913d-d2be14ea593a
info: cart.cartstore.ValkeyCartStore[0]
    GetCartAsync called with userId=260d0de5-f080-489a-a463-1f16ced627f9
info: cart.cartstore.ValkeyCartStore[0]
    GetCartAsync called with userId=b2ad60f4-7451-426c-968c-3b916e75d556
info: cart.cartstore.ValkeyCartStore[0]
    GetCartAsync called with userId=260d0de5-f080-489a-a463-1f16ced627f9
info: cart.cartstore.ValkeyCartStore[0]
    GetCartAsync called with userId=29d9143e-1a00-4541-9d24-6ade6db0e561
info: cart.cartstore.ValkeyCartStore[0]
    GetCartAsync called with userId=29d9143e-1a00-4541-9d24-6ade6db0e561
info: cart.cartstore.ValkeyCartStore[0]
    GetCartAsync called with userId=260d0de5-f080-489a-a463-1f16ced627f9
info: cart.cartstore.ValkeyCartStore[0]
    AddItemAsync called with userId=29886eec-2f95-11f0-913d-d2be14ea593a, productId=OLJCESPC7Z, quantity=2
info: cart.cartstore.ValkeyCartStore[0]
```

Shipping Service

```
ubuntu@ip-10-0-1-222:~$ kubectl logs pod/shipping-7c6bc949f7-zwl7t -n otel-collector
00:54:59 [INFO] OTel pipeline created
00:54:59 [INFO] listening on 0.0.0.0:8080
00:56:55 [INFO] Received quote: "1417.6"
00:56:55 [INFO] Sending Quote: 1417.60
00:56:55 [INFO] Tracking ID Created: 1235ea52-6f80-4855-a0a3-b46ff20994b2
00:57:01 [INFO] Received quote: "1824.1"
00:57:01 [INFO] Sending Quote: 1824.10
00:57:01 [INFO] Tracking ID Created: 6f837571-ffea-4472-951f-4e2c7f45bd5e
00:57:54 [INFO] Received quote: "553.8"
00:57:54 [INFO] Sending Quote: 553.79
00:57:54 [INFO] Tracking ID Created: 96735d6d-7389-41a6-b686-41dcf04c8e09
00:58:06 [INFO] Received quote: "276.6"
00:58:06 [INFO] Sending Quote: 276.60
00:58:06 [INFO] Tracking ID Created: 67faa92b-0e2e-4d93-b3bb-f1efe7fedd21
00:58:07 [INFO] Received quote: "1411.8"
00:58:07 [INFO] Sending Quote: 1411.80
00:58:07 [INFO] Tracking ID Created: ae848a34-1c3a-4b95-9d6e-5e665b51d06e
00:58:10 [INFO] Received quote: "283.8"
00:58:10 [INFO] Sending Quote: 283.80
00:58:10 [INFO] Tracking ID Created: 7e068d5a-b278-4a7d-8ac3-0d76f92009c0
00:58:13 [INFO] Received quote: "511"
00:58:13 [INFO] Sending Quote: 511.0
00:58:13 [INFO] Tracking ID Created: 85dcc260-604d-47ad-b8c2-cdec05954d56
00:58:26 [INFO] Received quote: "66"
00:58:26 [INFO] Sending Quote: 66.0
```

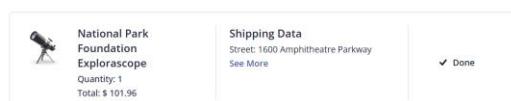
Screenshot of accessible application

Port forwarding from the local machine to the frontend-proxy service on port 8080 was successfully established using kubectl. This allows local browser access to the OpenTelemetry demo UI for validation and testing purposes

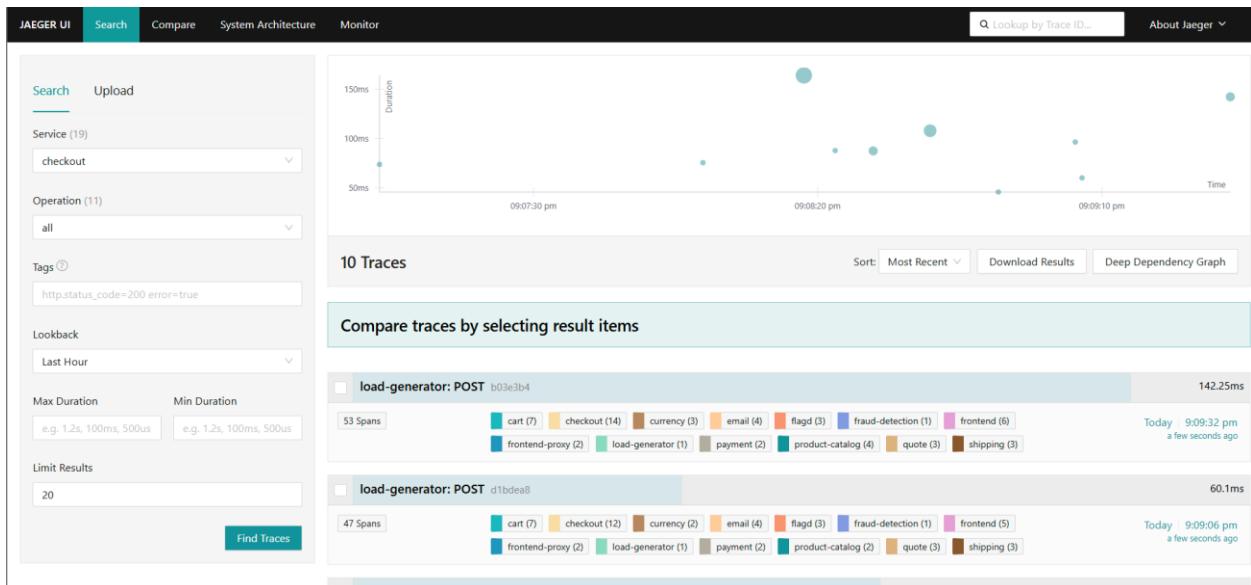
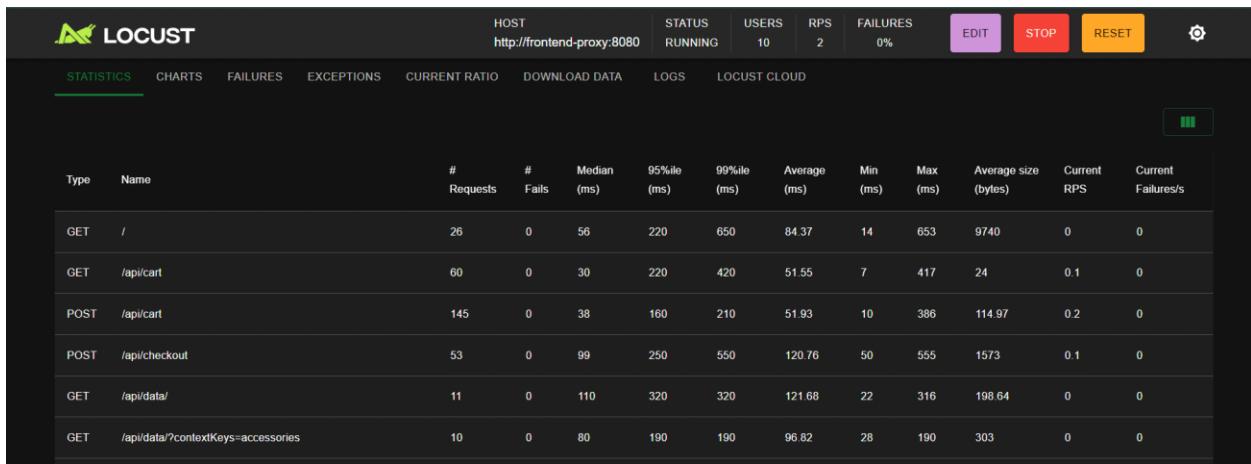
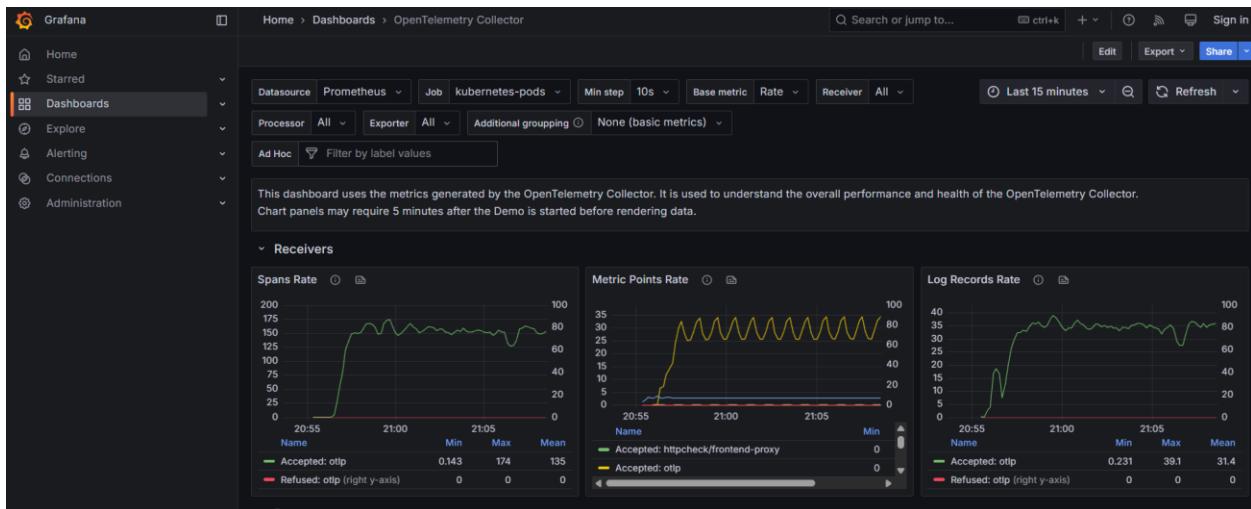


Your order is complete!

We've sent you a confirmation email.



[Continue Shopping](#)



Phase 2: Integrating Helm for Deployment

To streamline the management of Kubernetes resources and adopt a more maintainable and scalable deployment strategy, Helm was integrated in this phase. Helm enables templated configuration, versioned releases, and simplified rollbacks, which are critical in managing complex microservice environments like the OpenTelemetry Demo.

2.1 Adding and Updating the Helm Repository

The official OpenTelemetry Helm charts were sourced from the OpenTelemetry Helm Chart Repository. The Helm client was first installed on the EKS admin EC2 node, followed by repository configuration.

The following commands were run:

```
helm repo add open-telemetry https://open-telemetry.github.io/opentelemetry-helm-charts
```

```
helm repo update
```

```
ubuntu@ip-10-0-1-222:~$ helm repo add open-telemetry https://open-telemetry.github.io/opentelemetry-helm-charts
"open-telemetry" already exists with the same configuration, skipping
```

```
ubuntu@ip-10-0-1-222:~$ helm search repo open-telemetry
NAME          CHART VERSION   APP VERSION   DESCRIPTION
open-telemetry/opentelemetry-collector  0.122.5      0.123.1       OpenTelemetry Collector Helm chart for Kubernetes
open-telemetry/opentelemetry-demo        0.37.1       2.0.2        opentelemetry demo helm chart
open-telemetry/opentelemetry-ebpf       0.1.6        v0.10.2      OpenTelemetry eBPF Helm chart for Kubernetes
open-telemetry/opentelemetry-kube-stack  0.5.2        0.120.0      OpenTelemetry Quickstart chart for Kubernetes. ...
open-telemetry/opentelemetry-operator    0.88.6       0.124.0      OpenTelemetry Operator Helm chart for Kubernetes
ubuntu@ip-10-0-1-222:~$
```

```
ubuntu@ip-10-0-1-222:~$ helm repo update
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "open-telemetry" chart repository
Update Complete. ⚡Happy Helming!⚡
ubuntu@ip-10-0-1-222:~$ |
```

2.2 Helm-Based Deployment into a Separate Namespace

To isolate the Helm-managed resources from the manually applied manifests in Phase 1, a new namespace `otel-demo-helm` was created using this command

```
kubectl create namespace otel-demo-helm
```

```
ubuntu@ip-10-0-1-222:~$ kubectl create namespace otel-demo-helm
namespace/otel-demo-helm created
ubuntu@ip-10-0-1-222:~$ |
```

```
ubuntu@ip-10-0-1-222:~$ kubectl get namespaces
NAME          STATUS   AGE
default       Active   11m
kube-node-lease Active   11m
kube-public    Active   11m
kube-system    Active   11m
otel-demo-helm Active   21s
ubuntu@ip-10-0-1-222:~$ |
```

The Helm chart was deployed using the following command:

```
helm install otel-demo open-telemetry/opentelemetry-demo -n otel-demo-helm
```

```
ubuntu@ip-10-0-1-222:~$ helm install otel-demo open-telemetry/opentelemetry-demo -n otel-demo-helm
NAME: otel-demo
LAST DEPLOYED: Tue May 13 02:20:29 2025
NAMESPACE: otel-demo-helm
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
=====
OTEL DEMO
=====

- All services are available via the Frontend proxy: http://localhost:8080
  by running these commands:
    kubectl --namespace otel-demo-helm port-forward svc/frontend-proxy 8080:8080

  The following services are available at these paths after the frontend-proxy service is exposed with port forwarding:
  Webstore           http://localhost:8080/
  Jaeger UI          http://localhost:8080/jaeger/ui/
  Grafana            http://localhost:8080/grafana/
  Load Generator UI http://localhost:8080/loadgen/
  Feature Flags UI  http://localhost:8080/feature/
ubuntu@ip-10-0-1-222:~$ |
```

This deployed all required components — including the frontend, backend microservices, OTEL Collector, Prometheus, Grafana, and Jaeger — as a unified Helm release. Custom values can be overridden using a `values.yaml` file, if needed.

Screenshot of all pods and services running under **otel-demo-helm** namespace

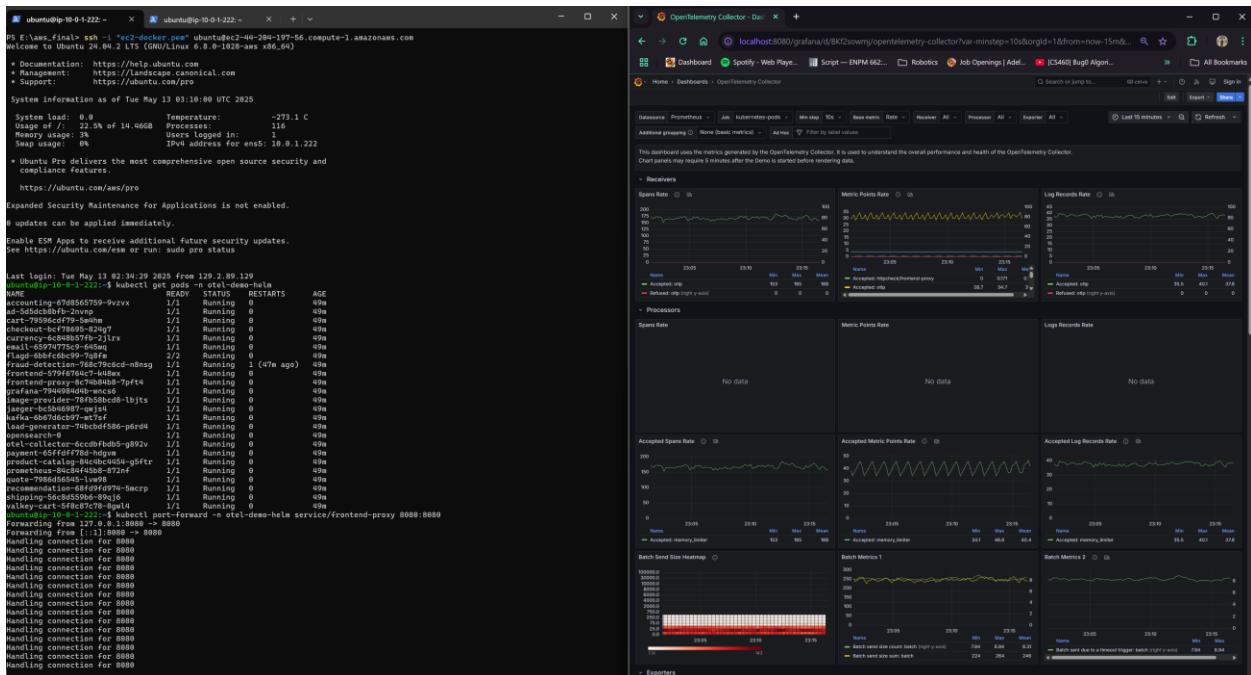
NAME	READY	STATUS	RESTARTS	AGE	
pod/accounting-67d8565759-9vzx	1/1	Running	0	2m6s	
pod/ad-5d5dcbb8fb-2nvnp	1/1	Running	0	2m8s	
pod/cart-79596cdf79-5m4hm	1/1	Running	0	2m8s	
pod/checkout-bcf78695-824q7	1/1	Running	0	2m6s	
pod/currency-6c848b57fb-2jlx	1/1	Running	0	2m7s	
pod/email-6597f775c9-645wq	1/1	Running	0	2m7s	
pod/flagn-6bbfc6bc99-7q8fm	2/2	Running	0	2m7s	
pod/fraud-detection-768c79c6cd-n8msg	1/1	Running	1 (17s ago)	2m8s	
pod/frontend-579f6f764c7-kd8wx	1/1	Running	0	2m8s	
pod/frontend-proxy-8c74b84b8-7pf4	1/1	Running	0	2m8s	
pod/grafana-7944984d4ub-wncs6	1/1	Running	0	2m8s	
pod/image-provider-84c4b45b8-872nf	1/1	Running	0	2m8s	
pod/jaeger-be5b46987-qwj5l	1/1	Running	0	2m7s	
pod/kafka-b6b7d6cb97-mttsf	1/1	Running	0	2m5s	
pod/load-generator-74bcdf586-p6rd4	1/1	Running	0	2m8s	
pod/opensearch-0	1/1	Running	0	2m8s	
pod/otel-collector-6ccdbfbdb5-g892v	1/1	Running	0	2m8s	
pod/payment-65ffdff78d-hdgvn	1/1	Running	0	2m8s	
pod/product-catalog-84c4b45b4-g5ftx	1/1	Running	0	2m8s	
pod/prometheus-84c4b45b8-872nf	1/1	Running	0	2m8s	
pod/quote-7986d6545-lvw98	1/1	Running	0	2m8s	
pod/recommendation-68fd9fd974-5mcrp	1/1	Running	0	2m6s	
pod/shipping-56c8d559b6-89qj6	1/1	Running	0	2m6s	
pod/valkey-cart-5f8c87c78-8gwU4	1/1	Running	0	2m8s	
NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
service/ad	ClusterIP	10.100.165.128	<none>	8086/TCP	2m9s
service/cart	ClusterIP	10.100.213.144	<none>	8086/TCP	2m9s
service/checkout	ClusterIP	10.100.238.48	<none>	8086/TCP	2m9s
service/currency	ClusterIP	10.100.182.218	<none>	8086/TCP	2m9s
service/email	ClusterIP	10.100.2.127	<none>	8086/TCP	2m9s
service/flagn	ClusterIP	10.100.140.66	<none>	8013/TCP, 8016/TCP, 4000/TCP	2m9s
service/frontend	ClusterIP	10.100.176.173	<none>	8086/TCP	2m9s
service/frontend-proxy	ClusterIP	10.100.167.230	<none>	8086/TCP	2m9s
service/grafana	ClusterIP	10.100.224.240	<none>	80/TCP	2m9s
service/image-provider	ClusterIP	10.100.96.183	<none>	8081/TCP	2m9s
service/jaeger-agent	ClusterIP	None	<none>	5775/UDP, 5778/TCP, 6831/UDP, 6832/UDP	2m9s
service/jaeger-collector	ClusterIP	None	<none>	9411/TCP, 14256/TCP, 14267/TCP, 14268/TCP, 4317/TCP, 4318/TCP	2m9s
service/jaeger-query	ClusterIP	None	<none>	16686/TCP, 16685/TCP	2m9s
service/kafka	ClusterIP	10.100.119.68	<none>	9692/TCP, 9993/TCP	2m9s
service/load-generator	ClusterIP	10.100.149.159	<none>	8089/TCP	2m9s
service/opensearch	ClusterIP	10.100.183.166	<none>	9206/TCP, 9300/TCP, 9600/TCP	2m9s
service/opensearch-headless	ClusterIP	None	<none>	9209/TCP, 9300/TCP, 9600/TCP	2m9s
service/opensearch-headless	ClusterIP	None	<none>	9209/TCP, 9300/TCP, 9600/TCP	2m9s
service/otel-collector	ClusterIP	10.100.214.104	<none>	6831/UDP, 14256/TCP, 14267/TCP, 8888/TCP, 4317/TCP, 4318/TCP, 9411/TCP	2m9s
service/payment	ClusterIP	10.100.77.199	<none>	8088/TCP	2m9s
service/product-catalog	ClusterIP	10.100.13.233	<none>	8088/TCP	2m9s
service/prometheus	ClusterIP	10.100.83.241	<none>	9099/TCP	2m9s
service/quote	ClusterIP	10.100.40.47	<none>	8080/TCP	2m9s
service/recommendation	ClusterIP	10.100.183.176	<none>	8088/TCP	2m9s
service/shipping	ClusterIP	10.100.127.135	<none>	8086/TCP	2m9s
service/valkey-cart	ClusterIP	10.100.201.225	<none>	6379/TCP	2m9s
NAME	READY	UP-TO-DATE	AVAILABLE	AGE	
deployment.apps/accounting	1/1	1	1	2m8s	
deployment.apps/ad	1/1	1	1	2m8s	
deployment.apps/cart	1/1	1	1	2m8s	
deployment.apps/checkout	1/1	1	1	2m8s	
deployment.apps/currency	1/1	1	1	2m8s	
deployment.apps/email	1/1	1	1	2m8s	
deployment.apps/flagn	1/1	1	1	2m8s	
deployment.apps/fraud-detection	1/1	1	1	2m8s	
deployment.apps/frontend	1/1	1	1	2m8s	
deployment.apps/frontend-proxy	1/1	1	1	2m8s	
deployment.apps/grafana	1/1	1	1	2m8s	
deployment.apps/image-provider	1/1	1	1	2m8s	
deployment.apps/jaeger	1/1	1	1	2m8s	
deployment.apps/kafka	1/1	1	1	2m8s	
deployment.apps/load-generator	1/1	1	1	2m8s	
deployment.apps/otel-collector	1/1	1	1	2m8s	
deployment.apps/payment	1/1	1	1	2m8s	
deployment.apps/product-catalog	1/1	1	1	2m8s	
deployment.apps/prometheus	1/1	1	1	2m8s	
deployment.apps/quote	1/1	1	1	2m8s	
deployment.apps/recommendation	1/1	1	1	2m8s	
deployment.apps/shipping	1/1	1	1	2m8s	
deployment.apps/valkey-cart	1/1	1	1	2m8s	
NAME	DESIRED	CURRENT	READY	AGE	
replicaset.apps/accounting-67d8565759	1	1	1	2m7s	
replicaset.apps/ad-5d5dcbb8fb	1	1	1	2m8s	
replicaset.apps/cart-79596cd7f9	1	1	1	2m8s	
replicaset.apps/checkout-bcf78695	1	1	1	2m6s	
replicaset.apps/currency-6c848b57fb	1	1	1	2m7s	
replicaset.apps/email-6597f775c9	1	1	1	2m7s	
replicaset.apps/flagn-6bbfc6bc99	1	1	1	2m7s	
replicaset.apps/fraud-detection-768c79c6cd	1	1	1	2m8s	
replicaset.apps/frontend-579f6f764c7	1	1	1	2m8s	
replicaset.apps/frontend-proxy-8c74b84b8	1	1	1	2m8s	
replicaset.apps/grafana-7944984d4ub	1	1	1	2m8s	
replicaset.apps/image-provider-78fb58bcd8	1	1	1	2m6s	
replicaset.apps/jaeger-be5b46987	1	1	1	2m7s	
replicaset.apps/kafka-b6b7d6cb97	1	1	1	2m6s	
replicaset.apps/load-generator-74bcdf586	1	1	1	2m8s	
replicaset.apps/otel-collector-6ccdbfbdb5	1	1	1	2m8s	
replicaset.apps/payment-65ffdff78d	1	1	1	2m8s	
replicaset.apps/product-catalog-84c4b45b4	1	1	1	2m8s	
replicaset.apps/prometheus-84c4b45b8	1	1	1	2m8s	
replicaset.apps/quote-7986d6545	1	1	1	2m8s	
replicaset.apps/recommendation-68fd9fd974	1	1	1	2m6s	
replicaset.apps/shipping-56c8d559b6	1	1	1	2m6s	
replicaset.apps/valkey-cart-5f8c87c78	1	1	1	2m8s	
NAME	READY	AGE			
statefulset.apps/opensearch	1/1	2m8s			

Application deployed successfully using Helm:

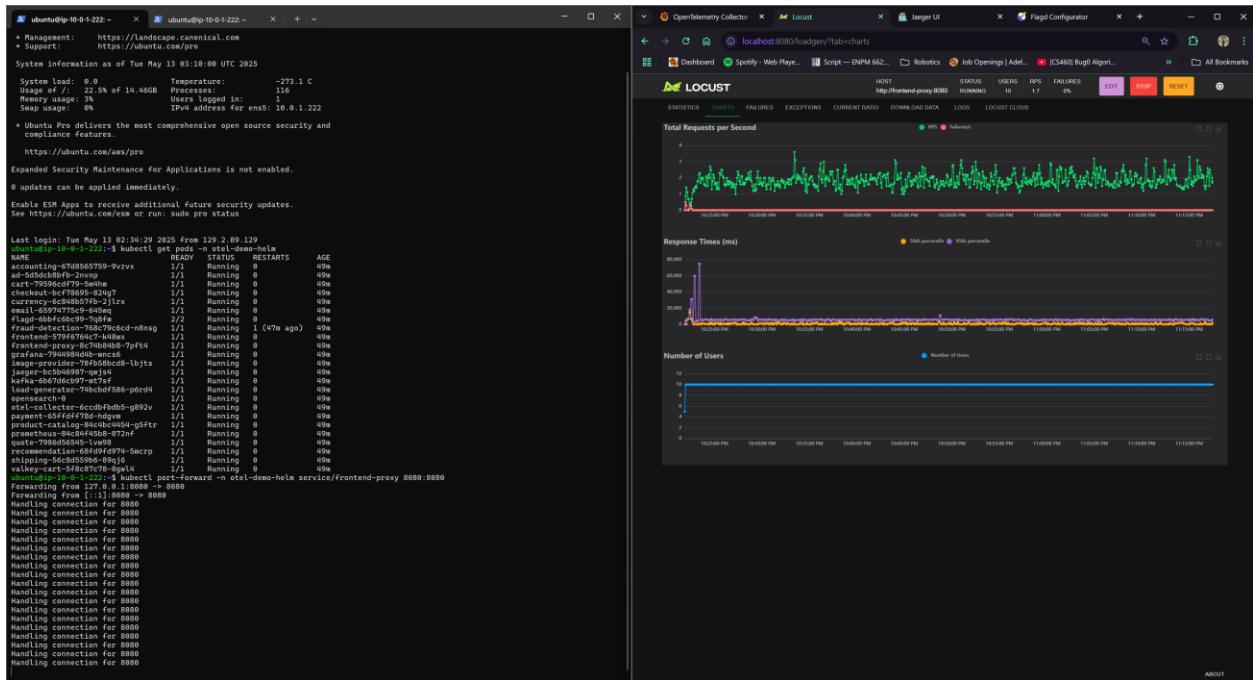
Web store:

The screenshot shows a browser window with two tabs. The left tab displays terminal output from an Ubuntu host, showing system information, service status, and logs related to Kubernetes and OpenTelemetry. The right tab shows a web storefront for 'otel-demo'. It features a header with the store logo and navigation links. Below the header is a message: 'Your order is complete!'. Underneath this, there's a summary box for a 'Solar Filter' item, showing a quantity of 1 at \$69.95, with a 'Done' button. Further down, a section titled 'You May Also Like' displays four telescope-related products with their names and prices: 'Ecliptica Travel Refractor Telescope' (\$129.95), 'Optical Tube Assembly' (\$359.00), 'Lens Cleaning Kit' (\$21.95), and 'Roof Binoculars' (\$209.95). A note at the bottom indicates a 20% discount for the telescope.

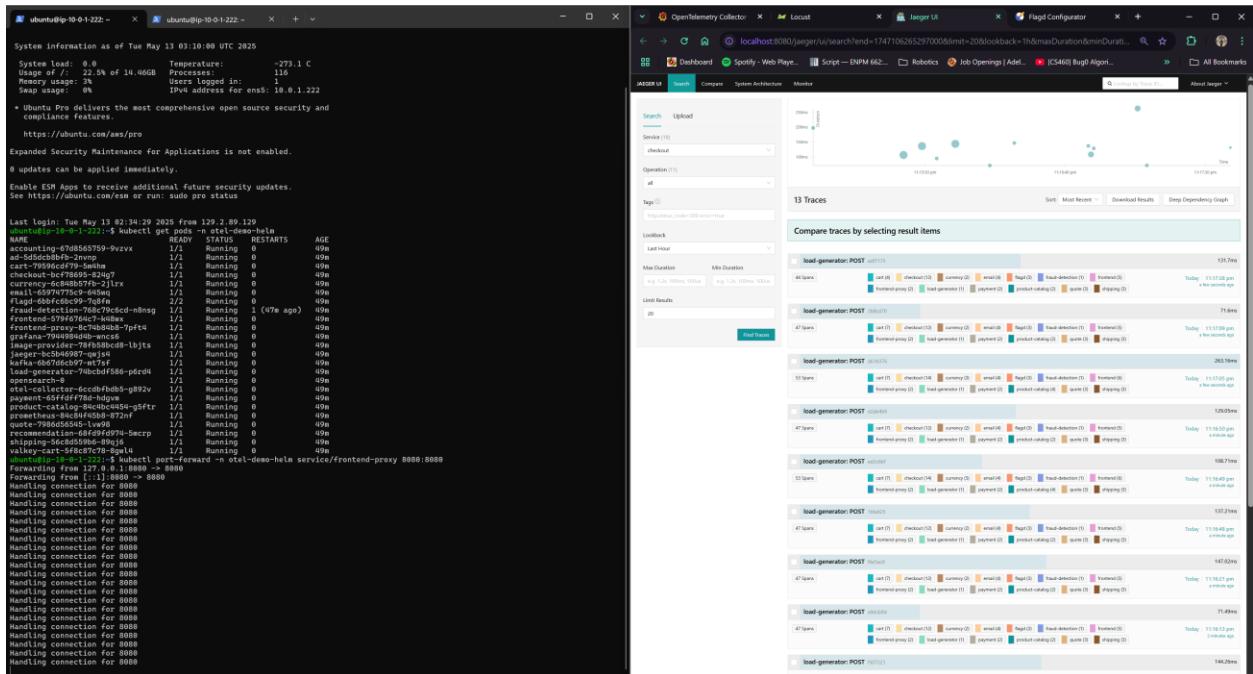
Grafana:



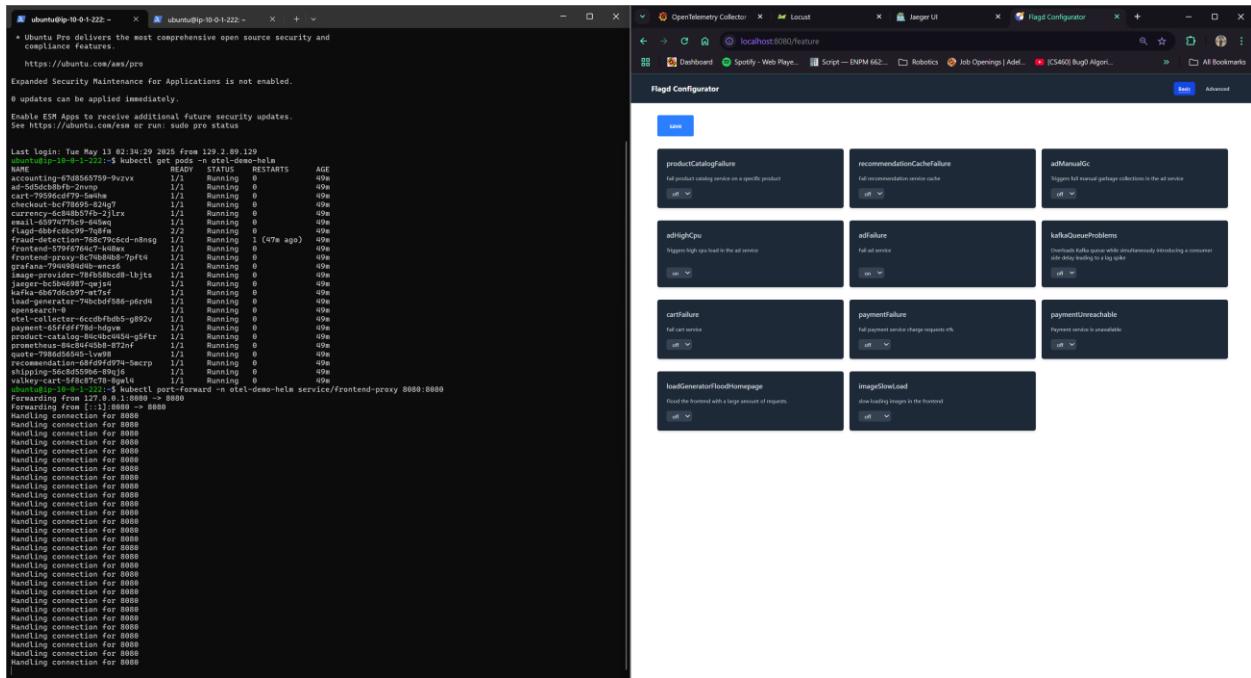
Load Generator UI:



Jaeger UI:



Flagd configurator UI:



2.3 Upgrade and Rollback Simulation

To test Helm's upgrade and rollback features, we simulated a configuration change by increasing the number of Jaeger replicas from 1 to 2. This type of upgrade is useful in real-world scenarios for scaling observability components based on load or redundancy requirements.

Initially, only a single pod for Jaeger was deployed as shown below (Output shows one Jaeger pod running.)

NAME	READY	STATUS	RESTARTS	AGE
accounting-67d8565759-9vzvx	1/1	Running	0	14m
ad-5d5dc8bfb-2nvnp	1/1	Running	0	14m
cart-79596cdf79-5m4hm	1/1	Running	0	14m
checkout-bcf78695-824g7	1/1	Running	0	14m
currency-6c848b57fb-2jlrx	1/1	Running	0	14m
email-65974775c9-645wq	1/1	Running	0	14m
flagd-6bbfc6bc99-7q8fm	2/2	Running	0	14m
fraud-detection-768c79c6cd-n8nsg	1/1	Running	1 (12m ago)	14m
frontend-579f6764c7-k48wx	1/1	Running	0	14m
frontend-proxy-8c74b84b8-7pft4	1/1	Running	0	14m
grafana-7944984d4b-wncs6	1/1	Running	0	14m
image-provider-78fb58bcd8-lbjts	1/1	Running	0	14m
jaeger-bc5b46987-qwjs4	1/1	Running	0	14m
kafka-6b67d6cb97-mt7sf	1/1	Running	0	14m
load-generator-74bcbdf586-p6rd4	1/1	Running	0	14m
opensearch-0	1/1	Running	0	14m
otel-collector-6ccdbfbdb5-g892v	1/1	Running	0	14m
payment-65ffdff78d-hdgvm	1/1	Running	0	14m
product-catalog-84c4bc4454-g5ftr	1/1	Running	0	14m
prometheus-84c84f45b8-872nf	1/1	Running	0	14m
quote-7986d56545-lvw98	1/1	Running	0	14m
recommendation-68fd9fd974-5mcrp	1/1	Running	0	14m
shipping-56c8d559b6-89qj6	1/1	Running	0	14m
valkey-cart-5f8c87c78-8gw14	1/1	Running	0	14m
ubuntu@ip-10-0-1-222:~\$				

Performing the Upgrade:

Before making any changes to the Helm release, we extracted the currently deployed configuration using:

```
helm get values -n otel-demo-helm otel-demo --all > values.yaml
```

This command retrieves all user-supplied values currently applied to the otel-demo release, including default and custom configurations. The output is redirected into a values.yaml file which serves two key purposes:

- **Baseline Reference:** It allows us to understand and compare the existing deployment configuration before applying any updates.
- **Safe Upgrade Practice:** Having a snapshot of the current values ensures we can safely roll back or re-apply the original settings if the upgrade fails or causes unexpected issues.

This is considered a best practice when performing upgrades with Helm, especially in production-like environments, as it preserves state and enables reproducibility.

```
ubuntu@ip-10-0-1-222:~$ helm upgrade otel-demo open-telemetry/opentelemetry-demo -n otel-demo-helm -f values.yaml
Release "otel-demo" has been upgraded. Happy Helm-ing!
NAME: otel-demo
LAST DEPLOYED: Tue May 13 02:56:43 2025
NAMESPACE: otel-demo-helm
STATUS: deployed
REVISION: 2
TEST SUITE: None
NOTES:
=====
OTEL DEMO
=====
- All services are available via the Frontend proxy: http://localhost:8080
by running these commands:
  kubectl --namespace otel-demo-helm port-forward svc/frontend-proxy 8080:8080

The following services are available at these paths after the frontend-proxy service is exposed with port forwarding:
Webstore          http://localhost:8080/
Jaeger UI         http://localhost:8080/jaeger/ui/
Grafana           http://localhost:8080/grafana/
Load Generator UI http://localhost:8080/loadgen/
Feature Flags UI  http://localhost:8080/feature/
ubuntu@ip-10-0-1-222:~$ |
```

After performing the Helm upgrade using the modified values.yaml file, we validated that the configuration was applied successfully.

NAME	READY	STATUS	RESTARTS	AGE
accounting-67d8565759-9vzvx	1/1	Running	0	37m
ad-5d5dc8bbfb-2nvnp	1/1	Running	0	37m
cart-79596cdf79-5m4hm	1/1	Running	0	37m
checkout-bcf78695-824g7	1/1	Running	0	37m
currency-6c848b57fb-2jlrx	1/1	Running	0	37m
email-65974775c9-645wq	1/1	Running	0	37m
flagd-6bbfc6bc99-7q8fm	2/2	Running	0	37m
fraud-detection-768c79c6cd-n8nsg	1/1	Running	1 (35m ago)	37m
frontend-579f6764c7-k48wx	1/1	Running	0	37m
frontend-proxy-8c74b84b8-7pf4	1/1	Running	0	37m
grafana-7944984d4b-wncs6	1/1	Running	0	37m
image-provider-78fb58bcd8-lbjts	1/1	Running	0	37m
jaeger-bc5b46987-qwjs4	1/1	Running	0	37m
jaeger-bc5b46987-swv42	1/1	Running	0	78s
kafka-6b67d6cb97-mt7st	1/1	Running	0	37m
load-generator-74bcfdf586-p6rd4	1/1	Running	0	37m
opensearch-0	1/1	Running	0	37m
otel-collector-6ccdbfbdb5-g892v	1/1	Running	0	37m
payment-65ffdff78d-hdpvm	1/1	Running	0	37m
product-catalog-84c4bc4454-g5ftr	1/1	Running	0	37m
prometheus-84c84f45b8-872nf	1/1	Running	0	37m
quote-7986d56545-lvw98	1/1	Running	0	37m
recommendation-68fd9fd974-5mcrp	1/1	Running	0	37m
shipping-56c8d559b6-89qj6	1/1	Running	0	37m
valkey-cart-5f8c87c78-8gw14	1/1	Running	0	37m

As shown in the screenshot above, the output confirms that two Jaeger pods (jaeger-bc5b46987-qwjs4 and jaeger-bc5b46987-swv42) are now running in the otel-demo-helm namespace. This change demonstrates that Helm correctly interpreted the updated replicas: 2 configuration in the values file and triggered a rolling update for the Jaeger Deployment.

Each Jaeger pod is in the Running state with 1/1 containers ready, indicating a healthy rollout. The second pod (-swv42) is newer (AGE: 78s) compared to the other (AGE: 37m), clearly showing that it was created as a result of the upgrade operation.

This outcome validates that:

- Helm accurately applies changes from the values.yaml file.

- Kubernetes dynamically scales deployments based on the updated Helm release.
- Stateful components like Jaeger remain fully functional and horizontally scalable.

This successful upgrade sets the stage for demonstrating Helm's rollback capability by simulating a failure next. After confirming the upgrade was successful, we used Helm's built-in version control to roll back the deployment to its previous state.

Checking Helm History:

```
ubuntu@ip-10-0-1-222:~$ helm history -n otel-demo-helm otel-demo
REVISION      UPDATED             STATUS          CHART           APP VERSION   DESCRIPTION
1            Tue May 13 02:20:29 2025  superseded    opentelemetry-demo-0.37.1  2.0.2        Install complete
2            Tue May 13 02:56:43 2025  deployed       opentelemetry-demo-0.37.1  2.0.2        Upgrade complete
ubuntu@ip-10-0-1-222:~$ |
```

As seen in the output, two revisions existed:

- Revision 1: The initial install (Install complete)
- Revision 2: The updated release with Jaeger replicas = 2 (Upgrade complete)

Performing the Rollback

To revert the changes and restore the original configuration (with a single Jaeger replica), we ran:

```
ubuntu@ip-10-0-1-222:~$ helm rollback otel-demo -n otel-demo-helm
Rollback was a success! Happy Helming!
```

Post-Rollback Verification:

```
ubuntu@ip-10-0-1-222:~$ kubectl get pods -n otel-demo-helm
NAME                           READY   STATUS    RESTARTS   AGE
accounting-67d8565759-9vzvx   1/1    Running   0          49m
ad-5d5dc8bfb-2nvnp           1/1    Running   0          49m
cart-79596cdf79-5m4hm        1/1    Running   0          49m
checkout-bcf78695-824g7       1/1    Running   0          49m
currency-6c848b57fb-2jlrx     1/1    Running   0          49m
email-65974775c9-645wq        1/1    Running   0          49m
flagd-6bbfc6bc99-7q8fm        2/2    Running   0          49m
fraud-detection-768c79c6cd-n8nsg 1/1    Running   1 (47m ago) 49m
frontend-579f6764c7-k48wx      1/1    Running   0          49m
frontend-proxy-8c74b84b8-7pft4 1/1    Running   0          49m
grafana-7944984d4b-wncs6      1/1    Running   0          49m
image-provider-78fb58bcd8-lbjts 1/1    Running   0          49m
jaeger-bc5b46987-qwjs4        1/1    Running   0          49m
kafka-6b67d6cb97-mt7st        1/1    Running   0          49m
load-generator-74bcfdf586-p6rd4 1/1    Running   0          49m
opensearch-0                   1/1    Running   0          49m
otel-collector-6ccdbfbdb5-g892v 1/1    Running   0          49m
payment-65ffdff78d-hdgvm      1/1    Running   0          49m
product-catalog-84c4bc4454-g5ftr 1/1    Running   0          49m
prometheus-84c84f45b8-872nf    1/1    Running   0          49m
quote-7986d56545-lvw98        1/1    Running   0          49m
recommendation-68fd9fd974-5mcrp 1/1    Running   0          49m
shipping-56c8d559b6-89qj6       1/1    Running   0          49m
valkey-cart-5f8c87c78-8gw14     1/1    Running   0          49m
ubuntu@ip-10-0-1-222:~$ |
```

Only **one Jaeger pod** (jaeger-bc5b46987-qwjs4) **remained**, indicating that the system successfully returned to the original state.

Phase 3: Alerting Service and Notifications

Prometheus was installed as the required monitoring solution using Helm with the following command:

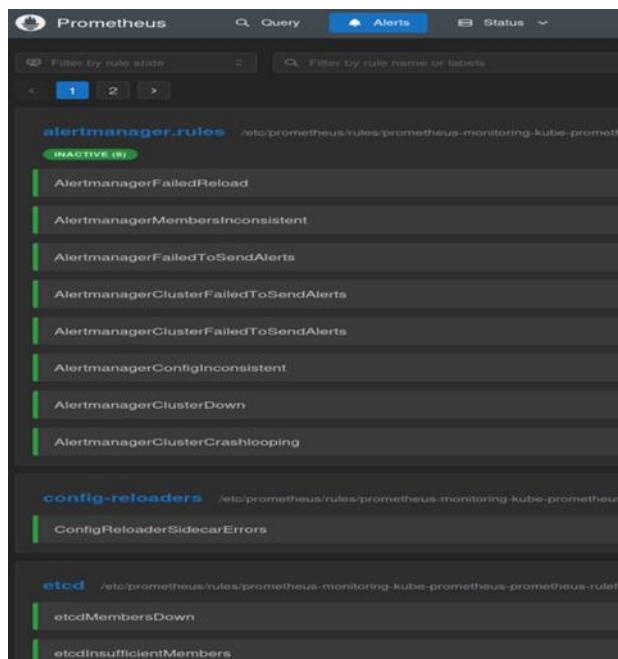
- **Helm install monitoring prometheus-community/kube-prometheus-stack --namespace monitoring**

NAME	READY	STATUS	RESTARTS	AGE
alertmanager-monitoring-kube-prometheus-alertmanager-0	2/2	Running	0	13m
crashloop-pod	0/1	CrashLoopBackOff	16 (4m37s ago)	61m
monitoring-grafana-5c548d845b-njmkr	3/3	Running	0	133m
monitoring-kube-prometheus-operator-d756c74bd-qwl7j	1/1	Running	0	133m
monitoring-kube-state-metrics-7c4bdd9c57-6jv75	1/1	Running	0	133m
monitoring-prometheus-node-exporter-pz6tf	1/1	Running	0	133m
monitoring-prometheus-node-exporter-r42kc	1/1	Running	0	133m
prometheus-monitoring-kube-prometheus-prometheus-0	2/2	Running	0	133m

We confirmed that Prometheus, Alertmanager, Grafana, and supporting components such as kube-state-metrics and node-exporter were all running successfully. To further verify the installation, we used port-forwarding to access the Prometheus UI from localhost:

- **kubectl port-forward svc/monitoring-kube-prometheus-prometheus -n monitoring 9090:9090**

Navigating to <http://localhost:9095> allowed us to view the Prometheus dashboard.



The screenshot shows the Prometheus dashboard with the 'Alerts' tab selected. In the 'alertmanager.rules' section, there are 8 inactive rules. One rule, 'AlertmanagerFailedReload', is highlighted. Other rules listed include 'AlertmanagerMembersInconsistent', 'AlertmanagerFailedToSendAlerts', 'AlertmanagerClusterFailedToSendAlerts', 'AlertmanagerConfigInconsistent', 'AlertmanagerClusterDown', and 'AlertmanagerClusterCrashlooping'. Below this, there are sections for 'config-reloaders' and 'etcd' with their respective rules.

To enable automatic detection of abnormal behavior in the cluster, we configured Prometheus alerting rules. A custom alert was defined to monitor pod restart frequency. This alert triggers when any container within a pod restarts more than twice in a five-minute period — a common sign of crash loops or unstable workloads.

The custom alert rule was written using Prometheus' expression language and applied using the following file:

```
apiVersion: monitoring.coreos.com/v1
kind: PrometheusRule
metadata:
  name: pod-restart-rules
  namespace: monitoring
spec:
  groups:
    - name: pod-restart-alerts
      rules:
        - alert: PodHighRestartCount
          expr: increase(kube_pod_container_status_restarts_total[5m]) > 2
          for: 1m
          labels:
            severity: warning
          annotations:
            summary: "Pod is restarting frequently"
            description: "Pod {{ $labels.pod }} in namespace {{ $labels.namespace }} restarted more than 2 times in 5 minutes."
```

This file was applied to the cluster with:

- **kubectl apply -f pod-restart-alert.yaml**

To test the alert, we created a pod that continuously fails using the following manifest:

```
apiVersion: v1
kind: Pod
metadata:
  name: crashloop-pod
  namespace: monitoring
spec:
  restartPolicy: Always
  containers:
    - name: crashloop
      image: busybox
      command: ["/bin/sh", "-c", "exit 1"]
```

Email Notifications

To receive alerts externally, we configured Alertmanager to send email notifications via Gmail's SMTP service. This required enabling 2-Step Verification on the Gmail account and generating a 16-character App Password, which was used in place of a regular Gmail password.

```
global:
  smtp_smarthost: 'smtp.gmail.com:587'
  smtp_from: 'f.shaker02@gmail.com'
  smtp_auth_username: 'f.shaker02@gmail.com'
  smtp_auth_password: 'rvuohtjbtdxwpzru'

route:
  receiver: gmail-alerts
  group_by: ['alertname']
  group_wait: 10s
  group_interval: 5m
  repeat_interval: 1h

receivers:
  - name: gmail-alerts
    email_configs:
      - to: 'shaker.55@outlook.com'
        send_resolved: true
```

This file was base64-encoded and inserted into the running Kubernetes environment via the Alertmanager secret:

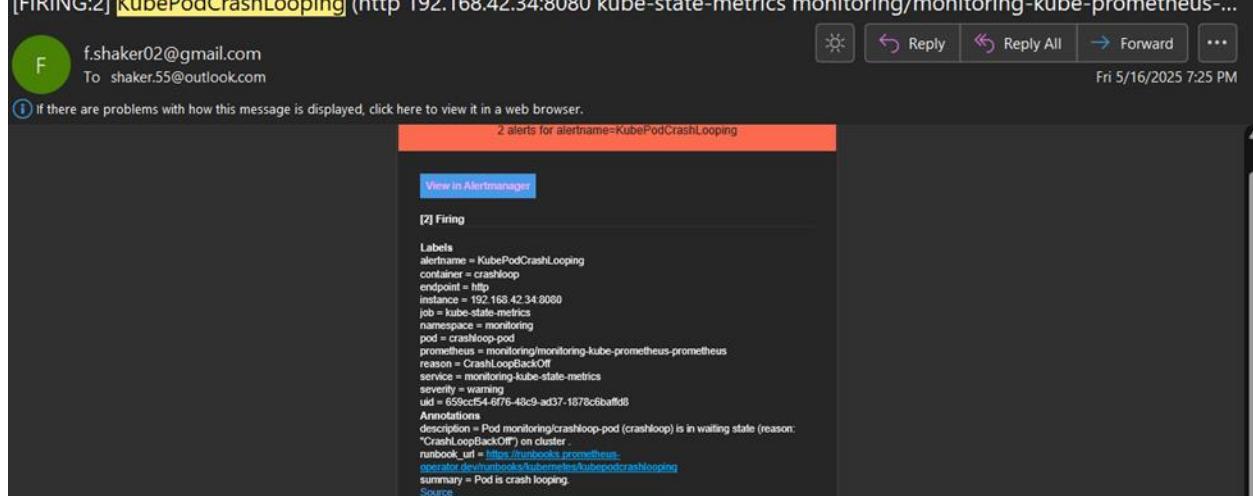
- **kubectl edit secret alertmanager-monitoring-kube-prometheus-alertmanager -n monitoring**

After saving the changes, the Alertmanager pod was restarted to apply the new configuration:

- **kubectl delete pod -l app.kubernetes.io/name=alertmanager -n monitoring**

The following screenshot confirms that the email alert was successfully triggered and delivered to the configured Gmail account when the crashloop-pod began restarting continuously:

[FIRING:2] **KubePodCrashLooping** (<http://192.168.42.34:8080/kube-state-metrics/monitoring/monitoring-kube-prometheus-...>)



The email subject is [FIRING:2] KubePodCrashLooping. It contains a link to a monitoring dashboard showing two alerts for the KubePodCrashLooping rule.

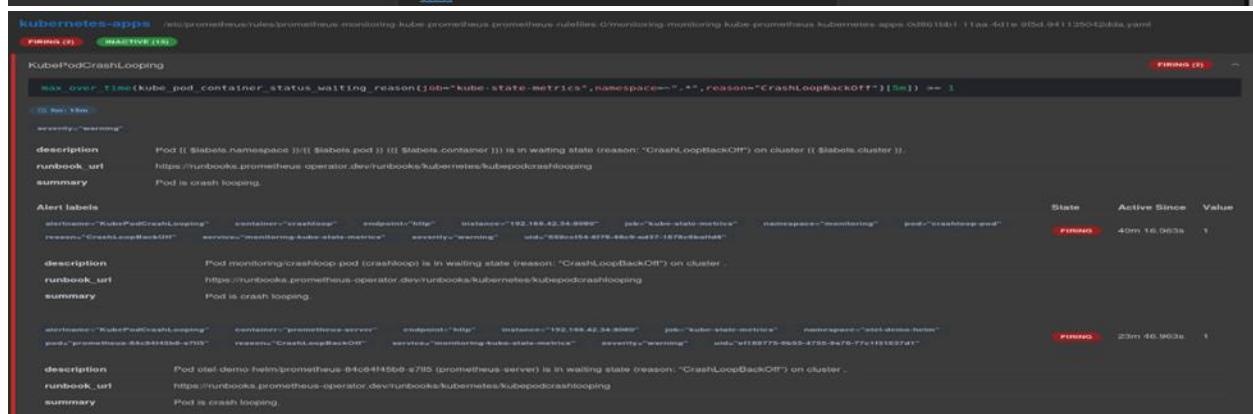
Labels:

- alername = KubePodCrashLooping
- container = crashloop
- endpoint = http
- instance = 192.168.42.34:8080
- job = kube-state-metrics
- namespace = monitoring
- pod = crashloop-pod
- prometheus = monitoring/monitoring-kube-prometheus-prometheus
- reason = CrashLoopBackOff
- service = monitoring-kube-state-metrics
- severity = warning
- uid = 659cc54-6f76-48c9-ad37-1878c6ba8fd8

Annotations:

- description = Pod monitoring/crashloop-pod (crashloop) is in waiting state (reason: "CrashLoopBackOff") on cluster.
- runbook_url = <https://runbooks.prometheus-operator.dev/runbooks/kubernetes/kubepodcrashlooping>
- operator_deployment = monitoring-kube-state-metrics
- summary = Pod is crash looping.
- source =

kubernetes-apps (<https://runbooks.prometheus-operator.dev/runbooks/kubernetes/prometheus/prometheus/rules:kube-monitoring-monitoring-kube-prometheus-kubernetes-apps-0d803dd1-11aa-4d1e-9143-11250-0200a.yaml>)



The screenshot shows the configuration for the KubePodCrashLooping rule in the Kubernetes Monitoring rules. It includes the alerting query, labels, annotations, and alert labels.

Alert labels:

- alername = "KubePodCrashLooping"
- container = "crashloop"
- endpoint = "http"
- instance = "192.168.42.34:8080"
- job = "kube-state-metrics"
- namespace = "monitoring"
- pod = "crashloop-pod"

Annotations:

- description = Pod monitoring/crashloop-pod (crashloop) is in waiting state (reason: "CrashLoopBackOff") on cluster.
- runbook_url = <https://runbooks.prometheus-operator.dev/runbooks/kubernetes/kubepodcrashlooping>
- summary = Pod is crash looping.

Alerting:

- alername = "KubePodCrashLooping"
- container = "prometheus-server"
- endpoint = "http"
- instance = "192.168.42.34:8080"
- job = "kube-state-metrics"
- namespace = "otel-demo-helm"
- pod = "prometheus-8fc84ff5bb-8785"

Annotations:

- description = Pod otel-demo-helm/prometheus-8fc84ff5bb-8785 (prometheus-server) is in waiting state (reason: "CrashLoopBackOff") on cluster.
- runbook_url = <https://runbooks.prometheus-operator.dev/runbooks/kubernetes/kubepodcrashlooping>
- summary = Pod is crash looping.

Phase 4: CI/CD Integration

4.1 CI/CD pipeline using GitHub Actions

To *build, deploy, test, rollback* the Open Telemetry demo app, we have utilized GitHub actions – GitHub's ubuntu env, GitHub Container registry, GitHub secrets, aws auth (for EKS) and Custom Helm Charts.

This is our [repository](#) that holds all the Workflow, manifest, docker compose files.

The screenshot shows the GitHub repository page for 'OpenTelemetry-Application-Deployment-in-EKS'. The repository is public and was forked from 'SwaraJ Mundrappa/Rao/OpenTelemetry-Application-Deployment-in-EKS'. The commit history shows 28 commits from the main branch, with the most recent being a merge by 'sakethbolla' 2 days ago. The repository has 0 stars, 0 forks, and 0 releases. The 'Languages' section indicates that TypeScript is the primary language at 53.4%, followed by Python at 10.6%.

Pipeline is created in the path [.github/workflows/ci-cd-frontend.yml](#)

This CI/CD pipeline uses Docker Compose to build images for all services and pushes them to GitHub Container Registry (GHCR). These images are then used to create pods in EKS using Helm, which helps keep the deployment simple and consistent.

4.2 Security within workflow & deployment

For authentication, we used a **GitHub Personal Access Token (GPAT)**.

For security, we created **GitHub secrets** and used them in the workflow code as shown below:

```
aws-access-key-id: ${{ secrets.AWS_ACCESS_KEY_ID }}
```

```
aws-secret-access-key: ${{ secrets.AWS_SECRET_ACCESS_KEY }}
```

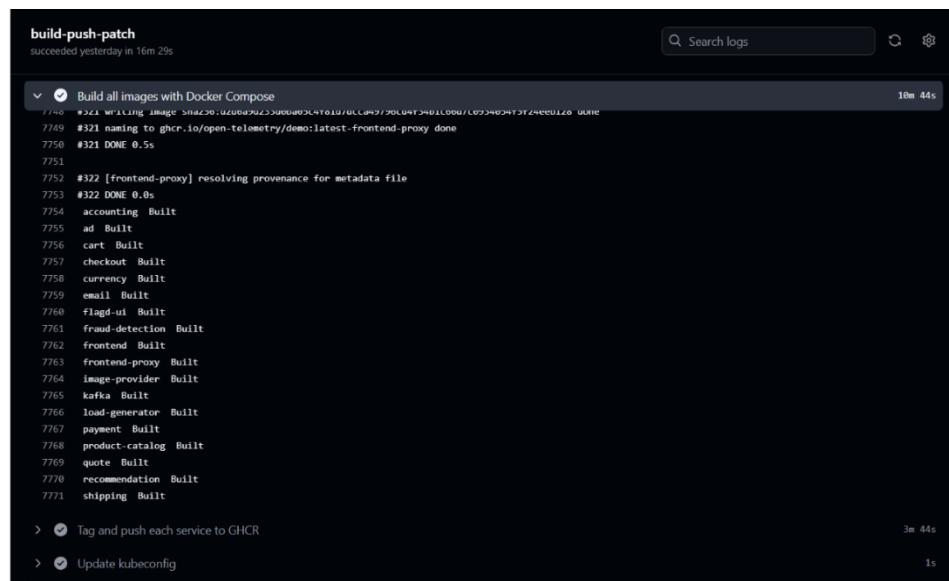
These secrets were referenced directly in the GitHub Actions workflow to keep credentials secure.

```

30 - name: Log in to GHCR
31   run: echo "${{ secrets.GHCR_PAT }}" | docker login ghcr.io -u ${{ github.actor }} --password-stdin
32
33 - name: Build all images with Docker Compose
34   run: docker compose -f $COMPOSE_FILE build
35
36 - name: Tag and push each service to GHCR
37   run: |
38     SERVICES=$(docker compose -f $COMPOSE_FILE config --services)
39     for SERVICE in $SERVICES; do
40       LOCAL_IMAGE=$(docker compose -f $COMPOSE_FILE config | awk "/^services:/,/^networks:/ { print \$1 }" | awk "/$SERVICE/{flag=1;next} [^ ]/{flag=0}flag && /image:/ {print \$2}")
41
42       if [ -z "$LOCAL_IMAGE" ]; then
43         echo "✖ Skipping $SERVICE - image not found"
44         continue
45       fi
46
47       GHCR_IMAGE="$REPO_BASE/$SERVICE:latest"
48       echo "👉 Tagging $LOCAL_IMAGE + $GHCR_IMAGE"
49       docker tag "$LOCAL_IMAGE" "$GHCR_IMAGE" || { echo "✖ Failed to tag $LOCAL_IMAGE"; continue; }
50
51       echo "🚀 Pushing $GHCR_IMAGE"
52       docker push "$GHCR_IMAGE" || echo "⚠ Push failed - continuing"
53     done
54

```

The pipeline logs below indicate building images from the [docker-compose.yml](#) file.



The next step in the workflow ensures that the docker images are pushed to **personal GHCR**.

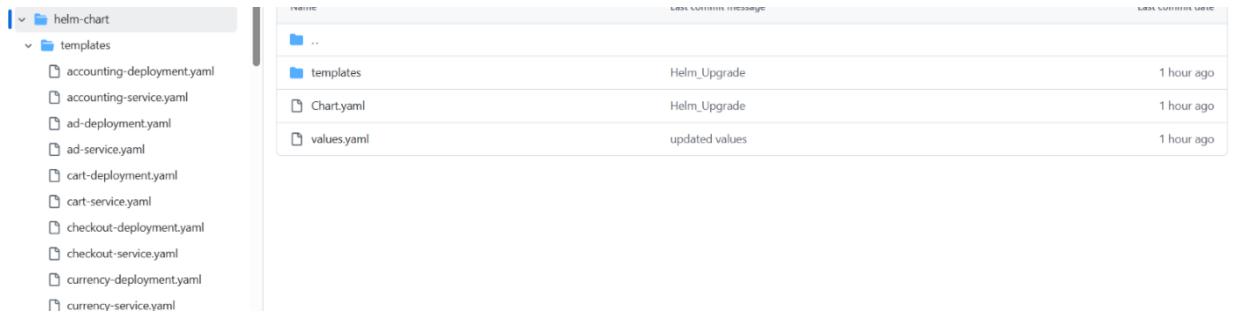
The screenshot shows a CI/CD pipeline interface with the following details:

- Summary:** A link to the overall project summary.
- Jobs:** A list of jobs including "build-push-patch" (selected), "Run details", "Usage", and "Workflow file".
- build-push-patch:** A detailed log of the build and push process to GHCR, showing numerous Docker image layers being pushed sequentially.
- Log View:** A search bar for logs and a timestamp indicating the log was updated 3m 44s ago.

The screenshot shows a package manager interface with the following details:

- Search Bar:** "Type [] to search" with a magnifying glass icon.
- Toolbar:** Includes icons for search, refresh, and other actions.
- Statistics:** "Stars 1" and a "Type: All" dropdown.
- Filter Options:** "Visibility: All" and "Sort by: Most downloads".
- List of Packages:**
 - otel-demo/email** (Private) - Published 2 days ago by Food_Crave69
 - otel-demo/quote** (Private) - Published 2 days ago by Food_Crave69
 - otel-demo/cart** (Private) - Published 2 days ago by Food_Crave69
 - otel-demo/flagd-ui** (Private) - Published 2 days ago by Food_Crave69
 - otel-demo/frontend-proxy** (Private) - Published 2 days ago by Food_Crave69
 - otel-demo/accounting** (Private) - Published 2 days ago by Food_Crave69

Since these are new docker images in personal GHCR, we had to create a new helm chart. we created a custom helm-structure with the chart.yaml , value.yaml , templates -- deployment, service.yaml



After pushing the images, the workflow connects to Kubernetes EKS cluster.

The screenshot shows a CI/CD pipeline interface. On the left, there's a sidebar with 'Summary', 'Jobs' (selected), 'Run details', 'Usage', and 'Workflow file'. The main area displays a 'build-deploy' job that has succeeded 1 hour ago. It contains two main sections: 'Update kubeconfig' and 'Test access to cluster'. The 'Update kubeconfig' section shows a command to run 'aws eks update-kubeconfig' and add a new context for an EKS cluster. The 'Test access to cluster' section shows a command to run 'kubectl get nodes' and lists one node: 'ip-192-168-45-62.ec2.internal' with status 'Ready', role '<none>', age '6h46m', and version 'v1.32.3-eks-473151a'. Below these sections, a list of steps shows the workflow: 'Create namespace if not exists', 'Deploy using Helm', 'Verify deployment', 'Post Configure AWS credentials', 'Post Checkout code', and 'Complete job', all of which have been completed successfully.

Then, its deployed using helm upgrade command
helm upgrade --install otel-ghcr ./helm-chart \ --namespace \$NAMESPACE \ --create-namespace

Summary

Jobs

build-deploy

Run details

Usage

Workflow file

build-deploy
succeeded 9 hours ago in 27s

Deploy using Helm

```

1 ► run helm upgrade --install otel-ghcr ./helm-chart \
17 Release "otel-ghcr" has been upgraded. Happy Helm-ing!
18 NAME: otel-ghcr
19 LAST DEPLOYED: Sun May 18 05:26:42 2025
20 NAMESPACE: otel-demo-cl-cd
21 STATUS: deployed
22 REVISION: 6
23 TEST SUITE: None

```

Verify deployment

```

1 ► Run kubectl get pods -n $NAMESPACE
15 NAME READY STATUS RESTARTS AGE
16 accounting-559597c99-p2s28 1/1 Running 0 5s
17 accounting-5c564c5b-qvjq9j 1/1 Terminating 0 41s
18 ad-79cb09c964-fvzzs 1/1 Running 0 41s
19 ad-c657ab899-4cpzb 0/1 ContainerCreating 0 5s
20 cart-97c5c599b-1kucn 0/1 ContainerCreating 0 5s
21 cart-77d4fb89-19cad 1/1 Running 1 (17s ago) 41s
22 checkout-55968f8587-9flrm 0/1 CrashLoopBackOff 1 (14s ago) 41s
23 checkout-5fdf548d54-2z1g2 0/1 ContainerCreating 0 4s
24 currency-5756b75565-szfb2 0/1 Pending 0 4s
25 currency-687c8687f4-rxzs 0/1 CrashLoopBackOff 2 (19s ago) 41s
26 email-74688459540-w7qng 0/1 Pending 0 4s
27 email-7980ccf99-w5282 0/1 ContainerCreating 0 41s
28 flagd-5965dc5cffc-866vn 0/1 Pending 0 4s
29 flagd-5965dc5cffc-82ttx 1/1 Running 0 41s
30 fraud-detection-74766998c8-m9xgs 0/1 Pending 0 3s
31 fraud-detection-7cf4c69fc9-cbt4t 1/1 Running 0 41s

```

4.3 Rollback strategies

The reason many pods are giving CrashLoopBackOff and error is, Services like checkout, payment, product-catalog require DB connection strings or REDIS_HOST, KAFKA_BROKER, etc. Which weren't present due to custom Helm upgrade, and custom docker images.

Summary

Jobs

build-deploy

Rollback on failure

Run details

Usage

Workflow file

build-deploy
succeeded 1 hour ago in 14s

Deploy using Helm

```

19 LAST DEPLOYED: Sun May 18 05:00:29 2025
20 NAMESPACE: otel-demo-cl-cd
21 STATUS: deployed
22 REVISION: 12
23 TEST SUITE: None

```

Wait for pods to be ready

```

1 ► Run ATTEMPTS=0
25 Pods are ready

```

Verify deployment

```

1 ► Run kubectl get pods -n $NAMESPACE
15 NAME READY STATUS RESTARTS AGE
16 accounting-559597c99-9szar 1/1 Running 0 9h
17 ad-c657ab899-qrxwp 1/1 Running 16 (5m50s ago) 9h
18 cart-97c5c599b-52zxs 0/1 Error 19 (5m36s ago) 9h
19 checkout-5fdf548d54-1ccm8 0/1 CrashLoopBackOff 19 (4m55s ago) 9h
20 currency-5756b75565-mzwf7 0/1 CrashLoopBackOff 19 (4m49s ago) 9h
21 email-74688459540-w6fzg 0/1 CrashLoopBackOff 19 (3m36s ago) 9h
22 flagd-5965dc5cffc-bhdhw 1/1 Running 0 9h
23 fraud-detection-74766998c8-2vft4 1/1 Running 0 9h
24 frontend-5c5b5649b6-98j4 1/1 Running 0 9h
25 frontend-proxy-56d77d595-brb66 0/1 CrashLoopBackOff 19 (3m57s ago) 9h
26 grafana-79449844db-949nu 0/1 ContainerCreating 0 9h
27 image-provider-6457fb447-5cxyj 0/1 CrashLoopBackOff 19 (4m34s ago) 9h
28 kafka-6f56f915fb-wshpd 0/1 CrashLoopBackOff 15 (2m58s ago) 9h
29 load-generator-5c7cc7d74-vwjkw 1/1 Running 0 9h

```

This should not affect the application, so we have integrated a simple rollback command in our pipeline to revert our application back whenever we face any issue.

To test out rollback, I performed.

Failed build

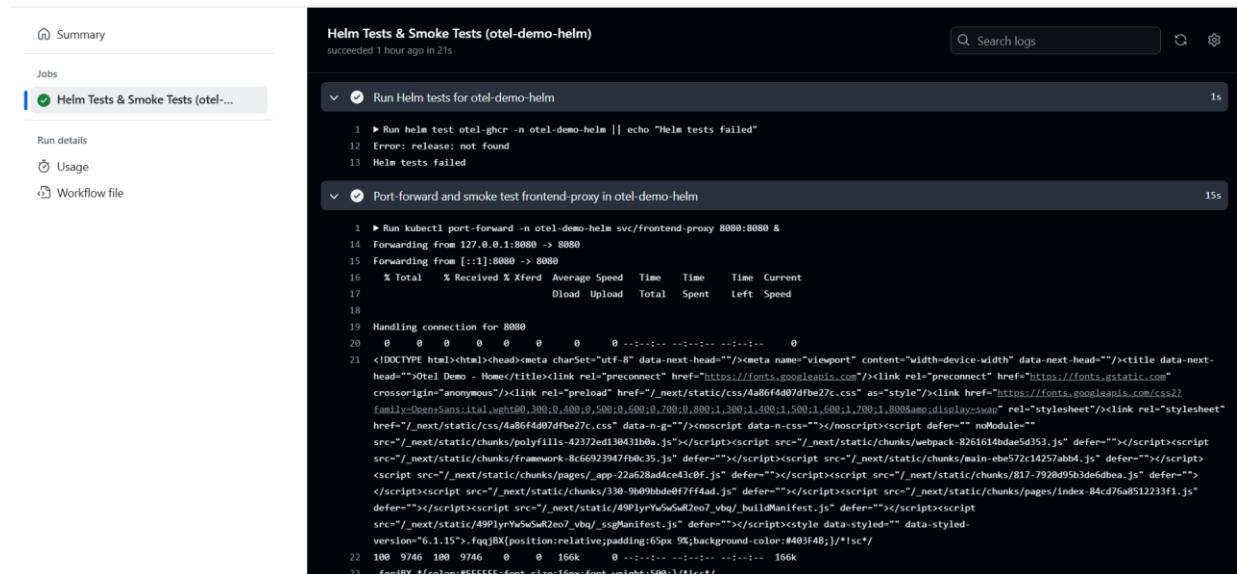
4.4 Testing Stage in CI/CD Pipeline

The testing stage in our CI/CD pipeline ensures the application components are functionally verified after deployment but before they are promoted as stable. This acts as a gatekeeper to catch deployment issues like service crashes, Helm misconfigurations, or failing pods before the change reaches production.

What We Test (Testing Pipeline)

1. Helm Tests: We run built-in Helm release tests to validate Kubernetes objects are healthy and responsive.
 2. Smoke Tests: These validate basic application functionality post-deployment, such as service reachability and response codes.
 3. Pod Health & Readiness: Kubernetes native kubectl get pods and kubectl describe pod help ensure all containers are in Running or Completed state with zero restarts or error statuses.
 4. Manual Port-Forward Verification: We expose services like frontend-proxy locally.

Successful build



Conclusion

Through the successful deployment of the OpenTelemetry Demo Application on AWS, this project validated a range of advanced cloud-native technologies and DevOps practices in a real-world context. The resulting architecture is not only scalable and observable but also demonstrates key operational principles required for modern distributed systems.

A major outcome of this project was the deep integration of automated infrastructure, security, monitoring, and CI/CD. Fine-grained AWS networking, including multi-AZ VPC layouts and strict security group policies, established a secure operational baseline. The use of modular Kubernetes resource definitions and Helm chart templating enabled rapid iteration and reproducibility across environments.

The monitoring stack—leveraging Prometheus and Grafana—provided actionable observability, with custom alerting and real-time notifications closing the feedback loop for operational awareness. The automation of build, test, deployment, and rollback processes via GitHub Actions ensured rapid, reliable delivery, while also illustrating how to integrate Helm and Kubernetes best practices into a CI/CD workflow.

Key technical lessons learned include:

- Modularization and Automation: Breaking down large manifests and using Helm charts streamlined complex deployments, simplified upgrades, and reduced operational risk.
- Observability by Design: Embedding telemetry, alerting, and dashboards at every layer provided immediate feedback and made troubleshooting more efficient.
- Resilience and Recovery: Simulating failures and rollbacks tested the platform's ability to self-heal and maintain service continuity under adverse conditions.
- Security and Compliance: Automating credential management and restricting network access minimized exposure and aligned with least-privilege principles.

This platform is now positioned for further extension—whether by scaling workloads, integrating additional telemetry sources, or adapting the CI/CD pipeline to new application requirements. The project's architecture and automation patterns serve as a blueprint for future microservices-based observability solutions on AWS or similar cloud environments.