

University of Maryland - College Park



Cloud Computing Final Project

## **EKS AND MONITORING WITH OPENTELEMETRY**

**Group 11**

[Project Files - GitHub](#)

[Video Demo](#)

Jayesh Pamnani  
Manav Gupta  
Chetan Rajesh  
Sagar Wagh

## Contents

Phase Assignment and Contribution Breakdown.....	3
Infrastructure Automation Using AWS CloudFormation.....	4
VPC and Networking CloudFormation Template:.....	4
Security Group CloudFormation Template:.....	5
EKS Cluster CloudFormation Template: .....	5
EKS Node Group CloudFormation Template:.....	6
EC2 Client Instance CloudFormation Template: .....	7
<b>1. Environment and Initial Application Setup .....</b>	<b>8</b>
1.1 Docker Deployment .....	8
1.1.1 Deployment Environment Setup.....	8
1.1.2 Application Deployment .....	8
1.1.3 Validating Deployment .....	9
1.1.4 Accessing the Application.....	11
1.2 Kubernetes Setup Tasks .....	13
1.2.1 EKS Cluster Setup .....	13
1.2.2 EKS Client Instance Configuration .....	15
1.2.3 Application Deployment.....	16
1.2.4 Application Validation .....	17
1.2.5 Application Accessibility .....	19
<b>2. YAML Splitting and Modular Deployment .....</b>	<b>21</b>
2.1 Original State: Monolithic opentelemetry-demo.yaml .....	21
2.1.1 The Problem with a Single opentelemetry-demo.yaml File .....	21
2.1.2 Why We Need to Split the YAML File .....	21
2.2 First Approach: Dividing by Resource Types .....	22
2.3 Second Approach: Dividing by Services .....	23
2.3.1 Advantages of this Approach .....	23
2.3.2 Namespace Management and Deployment Order .....	23
2.3.3 Applying Resources Individually or Recursively .....	29
2.3.4 Validate Resource Deployment.....	32

## Phase Assignment and Contribution Breakdown

	Group/Individual	Assignee
Phase 1 → 20%	Group	Group Effort
Phase 2 → 20%	Group	Group Effort
Phase 3	Individual	Manav Gupta
Phase 4	Individual	Jayesh Pamnani
Phase 5	Individual	Chetan Rajesh
Phase 6	Individual	Sagar Wagh
Phase 7 → 35%	Group	Group Effort

### Important Links:

[Project Files - GitHub](#)

[Video Demo](#)

# Infrastructure Automation Using AWS CloudFormation

To ensure an efficient, scalable, and reliable setup of our Kubernetes environment in AWS, we leveraged **AWS CloudFormation** to automate the creation and management of critical infrastructure components. CloudFormation allows us to define infrastructure as code (IaC), providing numerous advantages such as automation, consistency, scalability, version control and efficiency.

In this project, we used CloudFormation to provision the following components:

1. VPC and Networking
2. Security Groups
3. EKS Cluster
4. EKS Nodes Group
5. Client EC2 Instance

## VPC and Networking CloudFormation Template:

This CloudFormation template sets up a **multi-AZ VPC environment** tailored for the EKS cluster. It provides a robust and scalable network foundation with high availability and fault tolerance. The main components provisioned are:

1. **VPC:**
  - A virtual private cloud with a CIDR block of 10.0.0.0/16.
  - DNS support and hostnames enabled for seamless resource resolution.
2. **Subnets:**
  - **Two public subnets** for internet-accessible resources such as NAT Gateways.
  - **Two private subnets** for secure communication and resource isolation of EKS worker nodes.
  - Subnets are distributed across multiple Availability Zones (AZs) for high availability.
3. **Internet Gateway:**
  - Enables internet connectivity for resources in the public subnets.
4. **NAT Gateways:**
  - Two **NAT Gateways** in different public subnets to allow outbound internet access for resources in private subnets while maintaining their security.
5. **Route Tables:**
  - Separate route tables for public and private subnets:
    - Public route table with a route to the internet gateway.
    - Private route tables with routes to respective NAT Gateways.

### **Outputs:**

- **VPC ID:** Reference to the created VPC.
- **Public Subnets:** List of public subnet IDs.
- **Private Subnets:** List of private subnet IDs.

This template ensures secure and highly available networking infrastructure that supports the EKS cluster and its workloads.

## Security Group CloudFormation Template:

This template provisions **security groups** that enforce network traffic control for the EKS cluster and its worker nodes, ensuring secure communication within the cluster.

### 1. Cluster Security Group:

- Provides security for the EKS control plane.
- Allows inbound traffic on port **443 (HTTPS)** from any IP (for simplicity, could be restricted in production).

### 2. Node Security Group:

- Facilitates communication between worker nodes and the control plane.
- Inbound rules include:
  - Port **443** and ephemeral ports (**1025-65535**) from the control plane for secure communication.
  - Full internal communication within the VPC CIDR for unrestricted node-to-node communication.

### Outputs:

- **Cluster Security Group ID:** Security Group ID for the EKS control plane.
- **Node Security Group ID:** Security Group ID for the worker nodes.

This template ensures the cluster's security by isolating external access and enabling secure internal communication.

## EKS Cluster CloudFormation Template:

This template sets up the **EKS cluster control plane**, which manages Kubernetes workloads. It ensures the cluster is securely integrated with the VPC and allows API access through defined security rules.

### 1. EKS Cluster Role:

- An IAM role assigned to the EKS cluster for accessing AWS resources.
- Attached with AmazonEKSClusterPolicy.

### 2. EKS Cluster:

- Creates an EKS cluster named **otel-demo**.
- Configured with multiple **subnets** for high availability.
- Integrated with a **security group** for controlled network access.
- Latest Kubernetes version (1.31) is deployed for compatibility and performance.

### Outputs:

- **Cluster Name:** Name of the EKS cluster.
- **Cluster Endpoint:** API endpoint for cluster communication.
- **Cluster Role ARN:** Role ARN associated with the cluster.

This template ensures that the EKS control plane is highly available and securely managed.

## EKS Node Group CloudFormation Template:

This template provisions **managed worker nodes** for the EKS cluster. The nodes are auto-scaled and designed to run containerized workloads efficiently.

### 1. Node IAM Role:

- An IAM role assigned to worker nodes with policies for EKS operations:
  - AmazonEKSWorkerNodePolicy
  - AmazonEC2ContainerRegistryReadOnly
  - AmazonEKS\_CNI\_Policy

### 2. Node Group:

- Managed node group connected to the EKS cluster.
- Configured with **public subnets** for deployment.
- **Instance type:** t3.large, suitable for moderate workloads.
- **Scaling configuration:**
  - Minimum: 2 nodes
  - Maximum: 5 nodes
  - Desired: 2 nodes
- SSH access enabled using a **KeyPair** for administration.
- Tags included for environment tracking (Project: otel-demo, Environment: production).

### Outputs:

- **Node Group Name:** Identifier for the node group.
- **Node Role ARN:** IAM Role ARN assigned to the nodes.

This ensures auto-scaled worker nodes are seamlessly integrated with the cluster, capable of scaling based on workload.

## EC2 Client Instance CloudFormation Template:

This template provisions an **EC2 instance** with pre-installed tools for managing the EKS cluster, serving as a client machine for Kubernetes administration.

### 1. EC2 Instance:

- Instance type: **t3.micro** (or other supported types for flexibility).
- Connected to a **public subnet** for internet access.
- Configured with an **IAM Role** providing access to EKS, ECR, and other necessary AWS services.
- **Security Group** allows inbound traffic on ports **22 (SSH)**, **80 (HTTP)**, and **443 (HTTPS)** for remote access and administration.

### 2. Pre-installed Tools:

- **AWS CLI** for managing AWS services.
- **kubectl** for Kubernetes management.
- **eksctl** for simplified EKS operations.

### 3. User Data Script:

- Installs and configures tools upon instance launch.
- Automatically updates kubeconfig for EKS cluster management.

### Outputs:

- **Client Instance ID:** Identifier for the EC2 instance.
- **Client Public IP:** Public IP for accessing the instance.
- **Client Security Group ID:** Security group associated with the instance.

This setup provides an administrator-friendly environment for seamless management and monitoring of the EKS cluster and workloads

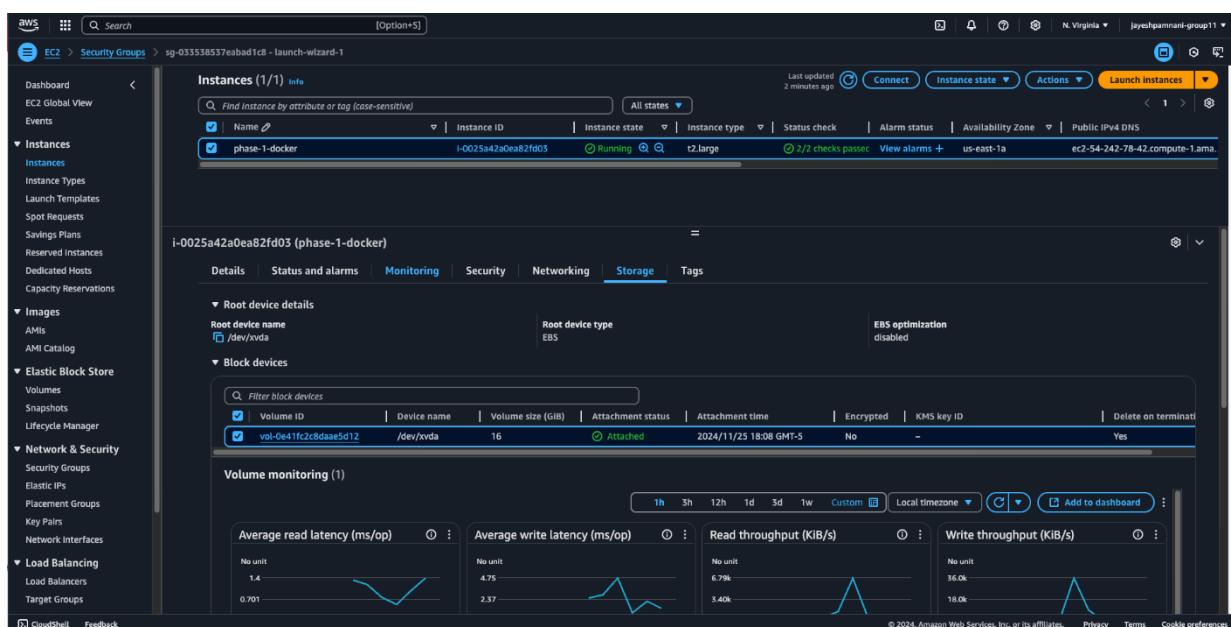
# 1. Environment and Initial Application Setup

## 1.1 Docker Deployment

### 1.1.1 Deployment Environment Setup

Provisioned an **EC2 instance** with the following specifications:

- **Instance Type:** t3.large (or equivalent).
- **Storage:** 16 GB attached to the instance.



### 1.1.2 Application Deployment

Installed and configured **Docker** and **Docker Compose** on the instance using the following steps:

- Install Docker:
  - `sudo yum install docker -y`
- Start and enable Docker:
  - `sudo systemctl start docker`
  - `sudo systemctl enable docker`
- Add ec2-user to the Docker group:
  - `sudo usermod -aG docker ec2-user`
  - `newgrp docker`
- Download and Install Docker Compose
  - `sudo curl -L "https://github.com/docker/compose/releases/download/v2.20.0/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose`

- sudo chmod +x /usr/local/bin/docker-compose
- Verify the installation:
  - docker-compose –version
- Deploy Application Using Docker Compose
  - Clone the repository containing the application:
    - git clone https://github.com/open-telemetry/opentelemetry-demo.git
    - cd opentelemetry-demo
- Run Docker Compose
  - docker-compose up -d

### 1.1.3 Validating Deployment

Verified service using docker ps and captured logs for each service to confirm successful startup using commands:

- docker ps

```

  ec2-user@ip-172-31-29-184:~$ docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS
e19f7244de23 gcr.io/open-telemetry/demo:latest-frontendproxy "/bin/sh -c 'envsubst..." 20 minutes ago Up 19 minutes 0.0.0.0:8080->8080/tcp, :::8080->10000/tcp, :::10000->10000/tcp
d848b7c868dc2 gcr.io/open-telemetry/demo:latest-loadgenerator "loucst -skip-log.s..." 20 minutes ago Up 19 minutes 0.0.0.0:32811->8089/tcp, :::32811->8089/tcp
f3ce652287b7 gcr.io/open-telemetry/demo:latest-frontend "/rm start" 20 minutes ago Up 19 minutes 0.0.0.0:32810->8080/tcp, :::32810->8080/tcp
d264ed084014 gcr.io/open-telemetry/demo:latest-checkoutservice "/checkoutservic..." 20 minutes ago Up 19 minutes 0.0.0.0:32809->5050/tcp, :::32809->5050/tcp
4be144eade1e gcr.io/open-telemetry/demo:latest-recommendationservice "opentelemetry-instr..." 20 minutes ago Up 19 minutes 0.0.0.0:32808->9001/tcp
1ca189ddaa30 gcr.io/open-telemetry/demo:latest-adservice "./build/install/ope..." 20 minutes ago Up 19 minutes 0.0.0.0:32799->9555/tcp, :::32799->9555/tcp
5b38c86e68dc gcr.io/open-telemetry/demo:latest-adservice "./" 20 minutes ago Up 19 minutes 0.0.0.0:32806->7070/tcp, :::32806->7070/tcp
e8544ec90501 gcr.io/open-telemetry/demo:latest-quoteservice "docker-php-entrypoi..." 20 minutes ago Up 19 minutes 0.0.0.0:32801->8090/tcp
e87bb931b8e6 gcr.io/open-telemetry/demo:latest-shippingservice "./instrument.sh dot..." 20 minutes ago Up 19 minutes
b15951a0b8a8 gcr.io/open-telemetry/demo:latest-shippingservice "/app/shippingservice" 20 minutes ago Up 19 minutes 0.0.0.0:32802->50050/tcp, :::32802->50050/tcp
674ea3708e02 gcr.io/open-telemetry/demo:latest-productcatalogservice "./productcatalogser..." 20 minutes ago Up 19 minutes 0.0.0.0:32800->3550/tcp, :::32800->3550/tcp
c5ea30a80901 gcr.io/open-telemetry/demo:latest-frauddetectionservice "java -jar frauddete..." 20 minutes ago Up 19 minutes
a86137ae9338 gcr.io/open-telemetry/demo:latest-emailservice "bundle exec ruby em..." 20 minutes ago Up 19 minutes 0.0.0.0:32807->6060/tcp, :::32807->6060/tcp
6f431d44cc77 gcr.io/open-telemetry/demo:latest-paymentservice "/qm run start" 20 minutes ago Up 19 minutes 0.0.0.0:32803->
c80cd228e0e4 gcr.io/open-telemetry/demo:latest-imagedeprovider "/dockercrash..." 20 minutes ago Up 19 minutes 80/tcp, 0.0.0.0:32805->8081/tcp, :::32805->8081/tcp
d9716d5e846 gcr.io/open-telemetry/demo:latest-flagd "/imageprovider" 20 minutes ago Up 19 minutes 0.0.0.0:32798->4000/tcp, :::32798->4000/tcp
74e3ac38f012 gcr.io/open-telemetry/demo:latest-currencyservice "sh -c '/usr/local/..." 20 minutes ago Up 19 minutes 0.0.0.0:32804->7000/tcp, :::32804->7000/tcp
08cb01638963 otel/opentelemetry-collector:v0.13.0 "/otelcol-contrib ..." 20 minutes ago Up 19 minutes 55678-55679/tcp, 0.0.0.0:32797->4317/tcp, :::32797->4317/tcp, 0.0.0.0:32796->4318/tcp, :::32796->4318/tcp
0263d3880176 opensearchproject/opensearch:2.18.0 "/opensearch-docker" 20 minutes ago Up 20 minutes (healthy) 9300/tcp, 9600/tcp, 9500/tcp, 0.0.0.0:32793->9200/tcp, :::32793->9200/tcp
c3d4fe740130 jaegertracing/all-in-one:62.0 #32794->16686/tcp, :::32794->16686/tcp jaeger
b87735ad67 grafana/grafana:11.3.0 "/run.sh" 20 minutes ago Up 20 minutes 4318/tcp, 5775/udp, 5775/tcp, 9411/tcp, 14256/tcp, 14268/tcp, 6831-6832/udp, 0.0.0.0:32795->4317/tcp, :::32795->4317/tcp, 0.0.0.0:32796->4318/tcp, :::32796->4318/tcp
3518935bd47d gcr.io/open-telemetry/demo:latest-kafka "/_caser=_entrypoint..." 20 minutes ago Up 20 minutes (healthy) 9092/tcp
31e3535082c5 volley/volley:8.0-alpine kafka "docker-entrypoint.s..." 20 minutes ago Up 20 minutes 0.0.0.0:32791->6379/tcp, :::32791->6379/tcp
2f1b531798ec quay.io/prometheus/prometheus:v2.55.1 "/bin/prometheus --w..." 20 minutes ago Up 20 minutes 0.0.0.0:9090->9090/tcp, :::9090->9090/tcp
83497d89e29 gcr.io/open-feature/flag:v11.4 flagd "/flagd-build start ..." 20 minutes ago Up 20 minutes 0.0.0.0:32792->8013/tcp, :::32792->8013/tcp
[ec2-user@ip-172-31-29-184 ~]$
```

- docker-compose logs

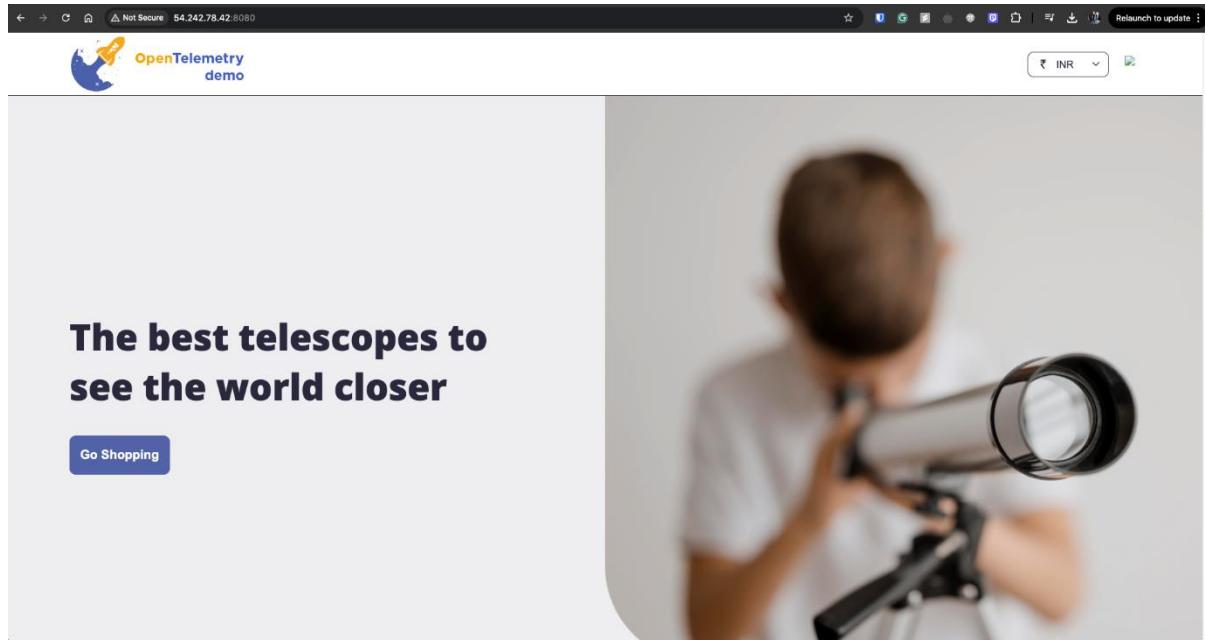
## FrontendProxy Logs:

## Docker application logs for Frontend

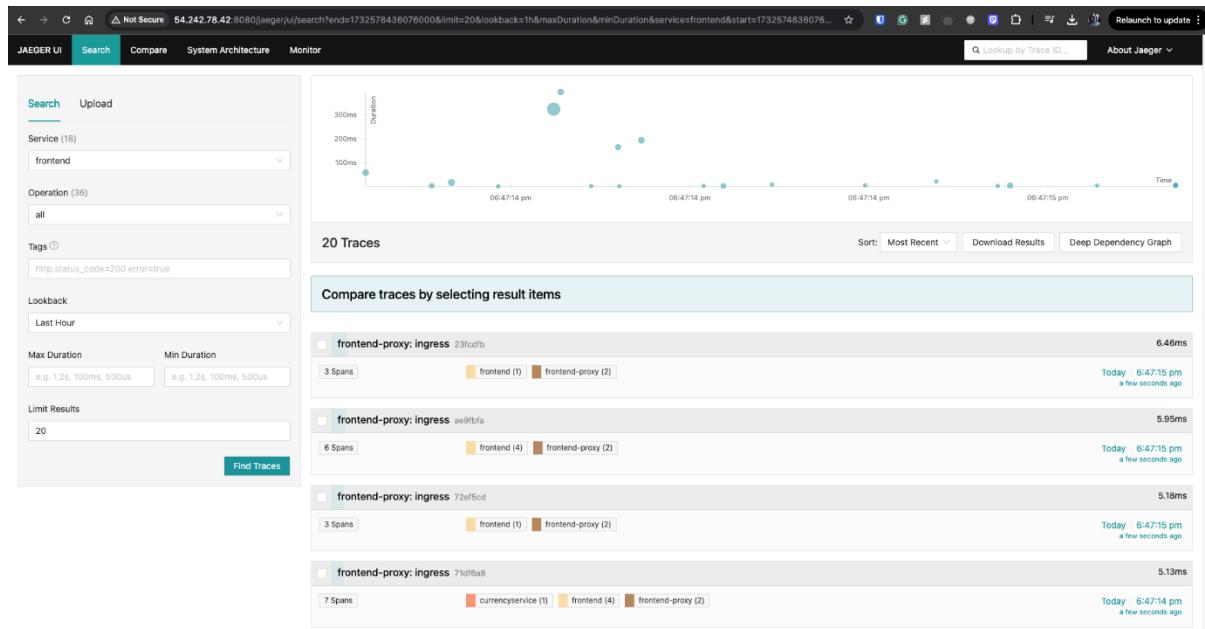
## 1.1.4 Accessing the Application

Accessed the application endpoints exposed by the containers to ensure proper functionality (e.g., frontend, backend, database, or other services) at <http://<public-ip>:8080>.

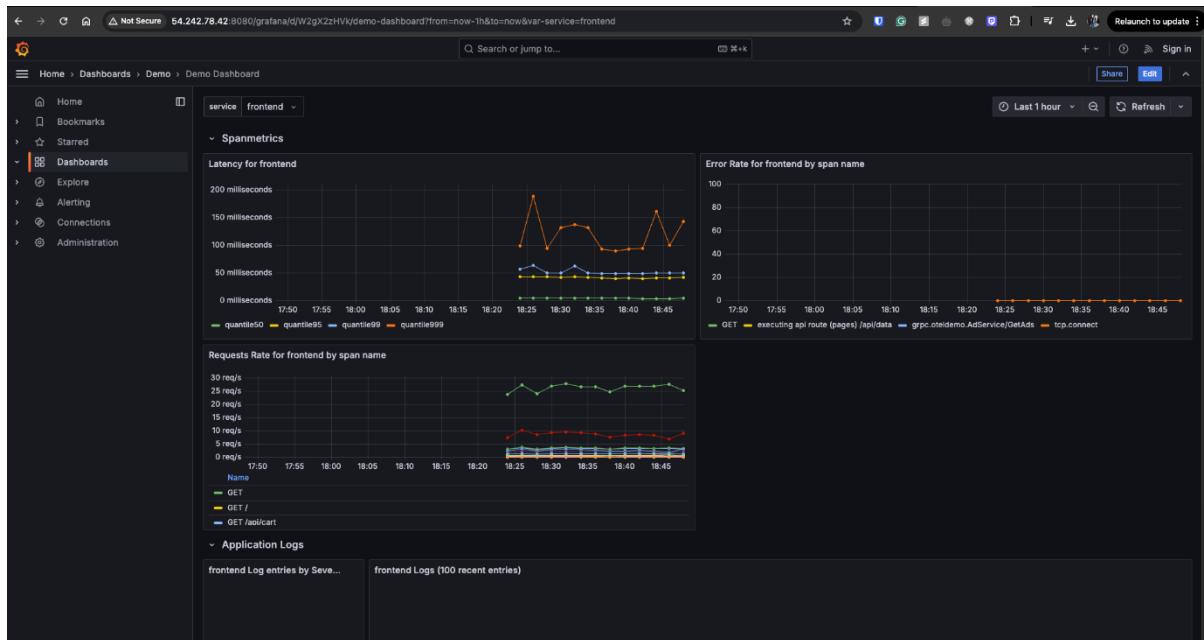
### Application @8080 Port



### Application Jaeger UI



## Application Grafana



## 1.2 Kubernetes Setup Tasks

The setup utilized AWS CloudFormation templates for automation and consistency.

### 1.2.1 EKS Cluster Setup

Created an EKS Cluster using a CloudFormation template to automate resource provisioning:

- Control Plane in a specified VPC and subnets.
- Worker Nodes:
  - At least 2 nodes.
  - Instance Type: t3.large.
- Deployed and configured cluster resources with required IAM roles.

CloudFormation Template Reference: [EKS Cluster CloudFormation Template](#), [EKS Node Groups CloudFormation Template](#)

#### Cluster Details:

- **Cluster Name:** otel-demo
- **Kubernetes Version:** 1.31
- **Cluster Role ARN:** arn:aws:iam::242201285753:role/eks-cluster-stack-EKSClusterRole-JTREuJqNgY (visible in IAM link)
- **Cluster ARN:** arn:aws:eks:us-east-1:242201285753:cluster/otel-demo
- **Security Group:** otel-demo-cluster-sg

#### Node Group Details:

- **Node Group Name:** NodeGroup-YIe2z3FpF4q
- **Instance Type:** t3.large
- **Node Group ARN:** arn:aws:eks:us-east-1:242201285753:nodegroup/otel-demo/NodeGroup-YIe2z3FpF4q/8c8a3e90-cf0b-305d-a0b6-fb8183c8e014
- **Node IAM Role ARN:** arn:aws:iam::242201285735:role/eks-cluster-stack-EKSNodeInstanceRole-JREUqINqYO

**otels-demo**

**Cluster info**

- Status: Active
- Kubernetes version: 1.31
- Support period: Standard support until November 26, 2025
- Provider: EKS

**Cluster health issues:** 0

**Upgrade insights:** 1

**Details**

API server endpoint: https://EC74B93F5E7044A13D01F58DF8AA5FF.sk1.us-east-1.eks.amazonaws.com

Certificate authority: LS0L51CRUJ7tBDRVJ5UZJQFURS0LH50cK1JSURCVENDQWUyZ0FSUJBzQJTNqNtmRbTH0JUVJ8PfZ5ktvWkldmNOQVFTERQXGVEVUTUJPROEXVUUKQ0NS2

OpenID Connect provider URL: https://oidc.eks.us-east-1.amazonaws.com/id/EC74B93F5E7044A13D01F58DF8AA5FF

Cluster IAM role ARN: arn:aws:iam:242201285733:role/eks-cluster-stack-EKSClusterRole-FLDNjlfbyQD

View in IAM

**Kubernetes version settings**

Upgrade policy: Extended

**Overview Resources Compute Networking Add-ons Access Observability Update history Tags**

**otels-demo**

**Cluster info**

- Status: Active
- Kubernetes version: 1.31
- Support period: Standard support until November 26, 2025
- Provider: EKS

**Cluster health issues:** 0

**Upgrade insights:** 1

**Resources**

**Resource types**

- Workloads
- Cluster
  - Nodes

**Cluster: Nodes (2)**

A node is a worker machine in Kubernetes. [Learn more](#)

Node name	Instance type	Node group	Created	Status
ip-10-0-1-54.ec2.internal	t3.large	NodeGroup-YlceZp3FpF4q	Created 44 minutes ago	Ready
ip-10-0-2-162.ec2.internal	t3.large	NodeGroup-YlceZp3FpF4q	Created 44 minutes ago	Ready

**Overview Resources Compute Networking Add-ons Access Observability Update history Tags**

**EKS > Clusters > otels-demo > Node groups > NodeGroup-YlceZp3FpF4q**

**NodeGroup-YlceZp3FpF4q**

**Node group configuration**

- Kubernetes version: 1.31
- AMI type: AL2\_x86\_64
- AMI release version: 1.31.2-20241121
- Instance types: t3.large
- Status: Active
- Disk size: 20 GiB

**Details**

**Node group ARN:** arn:aws:eks:us-east-1:242201285733:nodergroup/otels-demo/NodeGroup-YlceZp3FpF4q/8cc9ba50-cf0b-305d-5d0a-fdb183eca014

**Created:** an hour ago

**Autoscaling group name:** eks-NodeGroup-YlceZp3FpF4q-8cc9ba50-cf0b-305d-5d0a-fdb183eca014

**Node IAM role ARN:** arn:aws:iam:242201285733:role/eks-cluster-stack-NodeInstanceRole-JR1EUqNqYOs

**View in IAM**

**Capacity type:** On-Demand

**Desired size:** 2 nodes

**Minimum size:** 2 nodes

**Maximum size:** 5 nodes

**Subnets:** subnet-0b43872b16079b0fc, subnet-04cf870078d4bda1

**Configure remote access to nodes:** on

**EC2 Key Pair:** endsem

**Allow remote access from:** sg-0c0fade9af6346740

## 1.2.2 EKS Client Instance Configuration

Provisioned a dedicated EC2 instance using a CloudFormation template:

- Instance Type: t3.micro
- Storage: 16 GB attached.

Installed and configured tools:

- kubectl
- AWS CLI
- Eksctl

CloudFormation Template Reference: [EC2 Client Instance CloudFormation Template](#)

The image contains two screenshots of the AWS Management Console, specifically the EC2 Instances page, illustrating the configuration of an EKS client instance.

**Screenshot 1: Details Tab**

This screenshot shows the "Details" tab for the instance `i-01056dd07fc4afdb7 (eks-client-instance)`. The instance is running and has a Public IPv4 address of `34.234.90.100`. It is associated with a VPC ID `vpc-02ca39ce049097b6` and a Subnet ID `subnet-0b43872b16079b0fc`. The instance type is `t2.micro`, and it is part of the `otel-demo-public-subnet-1` Auto Scaling group. The instance ARN is `arn:aws:ec2:us-east-1:124201285735:instance/i-01056dd07fc4afdb7`.

**Screenshot 2: Storage Tab**

This screenshot shows the "Storage" tab for the same instance. It displays the root device details, which is an EBS volume named `/dev/xvda` with a size of 8 GB. The volume monitoring section shows metrics for Average read latency (ms/op), Average write latency (ms/op), Read throughput (KiB/s), and Write throughput (KiB/s) over the last hour. The average read latency is 0.66 ms, average write latency is 1.01 ms, read throughput is 1.98 KiB/s, and write throughput is 5.29 KiB/s.

### 1.2.3 Application Deployment

Used the **opentelemetry-demo.yaml** Kubernetes manifest for deploying the OpenTelemetry Demo application.

- git clone <https://github.com/open-telemetry/opentelemetry-demo.git>
- cd opentelemetry-demo/Kubernetes
- kubectl apply --namespace otel-demo -f opentelemetry-demo.yaml

Validated the deployment:

- kubectl get all -n otel-demo

Validation:

Ensured all resources in the otel-demo namespace were running as expected. Accessed application endpoints via port-forwarding or LoadBalancer.

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

NAME	READY	STATUS	RESTARTS	AGE
pod/opentelemetry-demo-accountingservice	1/1	Running	0	16m
pod/opentelemetry-demo-adbservice	1/1	Running	0	16m
pod/opentelemetry-demo-cartservice	1/1	Running	0	16m
pod/opentelemetry-demo-checkoutservice	1/1	Running	0	16m
pod/opentelemetry-demo-currencyservice	1/1	Running	0	16m
pod/opentelemetry-demo-emailservice	1/1	Running	0	16m
pod/opentelemetry-demo-flagd	1/1	Running	0	16m
pod/opentelemetry-demo-frontendproxy	1/1	Running	0	16m
pod/opentelemetry-demo-grafana	1/1	Running	0	16m
pod/opentelemetry-demo-imageprovider	1/1	Running	0	16m
pod/opentelemetry-demo-imagedatastore	1/1	Running	0	16m
pod/opentelemetry-demo-jaeger-agent	1/1	Running	0	16m
pod/opentelemetry-demo-loadgenerator	1/1	Running	0	16m
pod/opentelemetry-demo-otel-col	1/1	Running	0	16m
pod/opentelemetry-demo-paymentservice	1/1	Running	0	16m
pod/opentelemetry-demo-productcatalogservice	1/1	Running	0	16m
pod/opentelemetry-demo-quotequoterservice	1/1	Running	0	16m
pod/opentelemetry-demo-recommendationservice	1/1	Running	0	16m
pod/opentelemetry-demo-shippingservice	1/1	Running	0	16m
pod/opentelemetry-demo-valkey	1/1	Running	0	16m
pod/otel-demo-opensearch	1/1	Running	0	16m

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	AGE
service/opentelemetry-demo-adbservice	ClusterIP	172.20.153.217	<none>	16m
service/opentelemetry-demo-cartservice	ClusterIP	172.20.239.47	<none>	16m
service/opentelemetry-demo-checkoutservice	ClusterIP	172.20.155.19	<none>	16m
service/opentelemetry-demo-currencyservice	ClusterIP	172.20.13.91	<none>	16m
service/opentelemetry-demo-emailservice	ClusterIP	172.20.34.39	<none>	16m
service/opentelemetry-demo-flagd	ClusterIP	172.20.203.23	<none>	16m
service/opentelemetry-demo-frontend	ClusterIP	172.20.65.238	<none>	16m
service/opentelemetry-demo-frontendproxy	LoadBalancer	172.20.221.99	a2bbc6da0afc4ad89eff764083a51d7-1448227332.us-east-1.elb.amazonaws.com	8080:30994/TCP
service/opentelemetry-demo-grafana	ClusterIP	172.20.100.211	<none>	16m
service/opentelemetry-demo-imageprovider	ClusterIP	172.20.53.111	<none>	16m

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
deployment.apps/opentelemetry-demo-accountingservice	1/1	1	1	16m
deployment.apps/opentelemetry-demo-adbservice	1/1	1	1	16m
deployment.apps/opentelemetry-demo-cartservice	1/1	1	1	16m
deployment.apps/opentelemetry-demo-checkoutservice	1/1	1	1	16m
deployment.apps/opentelemetry-demo-currencyservice	1/1	1	1	16m
deployment.apps/opentelemetry-demo-emailservice	1/1	1	1	16m
deployment.apps/opentelemetry-demo-imagedatastore	1/1	1	1	16m
deployment.apps/opentelemetry-demo-jaeger-agent	1/1	1	1	16m
deployment.apps/opentelemetry-demo-loadgenerator	1/1	1	1	16m
deployment.apps/opentelemetry-demo-otel-col	1/1	1	1	16m
deployment.apps/opentelemetry-demo-paymentservice	1/1	1	1	16m
deployment.apps/opentelemetry-demo-productcatalogservice	1/1	1	1	16m
deployment.apps/opentelemetry-demo-prometheus-server	1/1	1	1	16m
deployment.apps/opentelemetry-demo-quoterservice	1/1	1	1	16m
deployment.apps/opentelemetry-demo-recommendationservice	1/1	1	1	16m
deployment.apps/opentelemetry-demo-shippingservice	1/1	1	1	16m
deployment.apps/opentelemetry-demo-valkey	1/1	1	1	16m
service/otel-demo-opensearch	ClusterIP	172.20.70.137	<none>	16m
service/otel-demo-opensearch-headless	ClusterIP	172.20.72.185	<none>	16m
deployment.apps/opentelemetry-demo-opensearch	None	<none>	16m	

NAME	READY	AGE
replicaset.apps/opentelemetry-demo-accountingservice-8555cb59fb	1	1m
replicaset.apps/opentelemetry-demo-adservice-5d4fb56d6	1	1m
replicaset.apps/opentelemetry-demo-cartservice-694df79ff7	1	1m
replicaset.apps/opentelemetry-demo-checkoutservice-744b6ccdd6	1	1m
replicaset.apps/opentelemetry-demo-dynamycurrency-service-6b8fcfd9bd6	1	1m
replicaset.apps/opentelemetry-demo-emailservice-66d8476fd6	1	1m
replicaset.apps/opentelemetry-demo-flagd-679866d67	1	1m
replicaset.apps/opentelemetry-demo-frauddetectionservice-85d48f855f	1	1m
replicaset.apps/opentelemetry-demo-frontend-59bccd8fdb	1	1m
replicaset.apps/opentelemetry-demo-frontendproxy-74f988cfb4	1	1m
replicaset.apps/opentelemetry-demo-grafana-69b6bd5d4	1	1m
replicaset.apps/opentelemetry-demo-imageprovider-7466d894fb	1	1m
replicaset.apps/opentelemetry-demo-jaegeer-7785549bb	1	1m
replicaset.apps/opentelemetry-demo-kafka-76d4d9f48b	1	1m
replicaset.apps/opentelemetry-demo-loadgenerator-54db796687	1	1m
replicaset.apps/opentelemetry-demo-otelcol-5c757fcf	1	1m
replicaset.apps/opentelemetry-demo-paymentservice-857974bcdb	1	1m
replicaset.apps/opentelemetry-demo-productcatalogservice-6bc98644f9	1	1m
replicaset.apps/opentelemetry-demo-prometheus-server-57cd8f9d46	1	1m
replicaset.apps/opentelemetry-demo-quoteservice-5658c58fd7	1	1m
replicaset.apps/opentelemetry-demo-recommendationservice-6f576fd574	1	1m
replicaset.apps/opentelemetry-demo-shippingservice-fcfc7765	1	1m
replicaset.apps/opentelemetry-demo-valkey-68b4cb4498	1	1m
NAME	READY	AGE
statefulset.apps/otel-demo-opensearch	1/1	16m

## 1.2.4 Application Validation

Accessed application to validate functionality:

- Used port-forwarding or LoadBalancer to expose services.
- Confirmed endpoints were operational.

Logs from Key Application Pods to confirm successful deployment:

### Frontend

```
[ec2-user@ip-10-0-1-11 ~]$ kubectl logs pod/opentelemetry-demo-frontend-59bccd8fdb-n27wm -n otel-demo
> frontend@0.1.0 start
> node --require ./Instrumentation.js server.js
  ▲ Next.js 14.2.5
  - Local:          http://opentelemetry-demo-frontend-59bccd8fdb-n27wm:8080
  - Network:        http://10.0.1.185:8080

  ✓ Starting...
  ✓ Ready in 597ms
```

### Grafana

```
[ec2-user@ip-10-0-1-11 ~]$ kubectl logs pod/opentelemetry-demo-grafana-69b6bd5dd4-p9qgs -n otel-demo
✓Downloaded and extracted grafana-opensearch-data source v2.22.0 zip successfully to /var/lib/grafana/plugins/grafana-opensearch-data source

Please restart Grafana after installing or removing plugins. Refer to Grafana documentation for instructions if necessary.

logger=settings t=2024-11-28T20:23:08.287133715Z level=info msg="Starting Grafana" version=11.3.0 commit=d9455ff7db73be9db7d412e49a68bec767f2b5a branch=HEAD compiled=2024-11-28T20:23:08Z
logger=settings t=2024-11-28T20:23:08.287627272Z level=info msg="Config loaded from" file=/usr/share/grafana/conf/default.ini
logger=settings t=2024-11-28T20:23:08.287643126Z level=info msg="Config loaded from" file=/etc/grafana/grafana.ini
logger=settings t=2024-11-28T20:23:08.287650811Z level=info msg="Config overridden from command line" arg="default.paths.data=/var/lib/grafana"
logger=settings t=2024-11-28T20:23:08.287652911Z level=info msg="Config overridden from command line" arg="default.paths.logs=/var/log/grafana"
logger=settings t=2024-11-28T20:23:08.287669396Z level=info msg="Config overridden from command line" arg="default.paths.provisioning=/etc/grafana/provisioning"
logger=settings t=2024-11-28T20:23:08.287676866Z level=info msg="Config overridden from command line" arg="default.log.mode=console"
logger=settings t=2024-11-28T20:23:08.287683528Z level=info msg="Config overridden from Environment variable" var="GF_PATHS_DATA=/var/lib/grafana/"
logger=settings t=2024-11-28T20:23:08.287689793Z level=info msg="Config overridden from Environment variable" var="GF_PATHS_LOGS=/var/log/grafana"
logger=settings t=2024-11-28T20:23:08.28769597Z level=info msg="Config overridden from Environment variable" var="GF_PATHS_PLUGINS=/var/lib/grafana/plugins"
logger=settings t=2024-11-28T20:23:08.287701882Z level=info msg="Config overridden from Environment variable" var="GF_PATHS_PROVISIONING=/etc/grafana/provisioning"
logger=settings t=2024-11-28T20:23:08.287708055Z level=info msg="Config overridden from Environment variable" var="GF_SECURITY_ADMIN_USER=admin"
logger=settings t=2024-11-28T20:23:08.287720824Z level=info msg="Target target=[all]"
logger=settings t=2024-11-28T20:23:08.287739109Z level=info msg="Path Home" path=/usr/share/grafana
logger=settings t=2024-11-28T20:23:08.287744491Z level=info msg="Path Data" path=/var/lib/grafana/
logger=settings t=2024-11-28T20:23:08.287744591Z level=info msg="Path Logs" path=/var/log/grafana
logger=settings t=2024-11-28T20:23:08.287750084Z level=info msg="Path Plugins" path=/var/lib/grafana/plugins
logger=settings t=2024-11-28T20:23:08.287755787Z level=info msg="Path Provisioning" path=/etc/grafana/provisioning
logger=settings t=2024-11-28T20:23:08.287761492Z level=info msg="App mode production"
logger=settings t=2024-11-28T20:23:08.288353654Z level=info msg="Feature toggles" cloudWatchCrossAccountQuerying=true addFieldFromCalculatedStatFunctions=true cloudWatchMetricsStreaming=true dashboardSceneColor=true ssosSetting=true accessControl=true roleBased=true managedPluginsInstall=true groupScopedTableTransformations=true transformationsRedesign=true dashboardSceneColor=true dashboardSceneColor=true autoMigrateXChartPanel=true promqlScope=true lokalQuerySplitting=true PrometheusMetricEncyclopedia=true formatString=true useLokiInfiniteScrolling=true correlations=true alertingInsights=true autoMigrateXChartPanel=true publicDashboards=true dataplaneFrontend=true noFallback=true alertingSimplifiedRouting=true dashboardScene=true alertingNoDataErrorExecution=true lokiMetricDataPlane=true publicDashboardsScene=true dashgpt=true nest=true annotationPermissionUpdate=true cloudWatchRoundUpEndTime=true lokalQueryHints=true PrometheusAzureOverride=true audience=true topnav=true awsAsyncQueryCaching=true e_lokiStructuredMetadata=true pinNavItem=true influxdbBackendMigration=true tlsManagement=true explore=true angularRepresentationUI=true logSc
```

## Payment Service

```
[ec2-user@ip-18-0-1-1 ~]$ kubectl logs pod/opentelemetry-demo-paymentservice-857974bcd-b2fdh -n otel-demo

> paymentservice@1.4.0 start
> node --require ./opentelemetry.js index.js

(node:17) [DEP0040] DeprecationWarning: The 'punycode' module is deprecated. Please use a userland alternative instead.
(Use 'node --trace-deprecation ...' to show where the warning was created)
(node:17) DeprecationWarning: Calling start() is no longer necessary. It can be safely omitted.

[{"level": "30", "time": "1732825870834", "pid": "17", "hostname": "opentelemetry-demo-paymentservice-857974bcd-b2fdh", "msg": "PaymentService gRPC server started on port 8080"}, {"level": "30", "time": "1732825631317", "pid": "17", "hostname": "opentelemetry-demo-paymentservice-857974bcd-b2fdh", "trace_id": "d108b341b28ae1f6bde34abf6c28ad", "span_id": "e054971490f538cf", "trace_flags": "01", "request": {"amount": {"currencyCode": "USD", "units": "low": 5984, "high": 0, "unsigned": false}, "nanos": "789999983", "creditCard": {"creditCardNumber": "+4485-4803-8707-3547", "cardType": "Visa", "lastFourDigits": "3547", "cardType": "Visa", "lastFourDigits": "3547", "cardExpirationYear": "2039", "creditCardExpirationMonth": "91"}, "msg": "Charge request received."}, {"level": "30", "time": "17328256313165", "pid": "17", "hostname": "opentelemetry-demo-paymentservice-857974bcd-b2fdh", "trace_id": "d108b341b28ae1f6bde34abf6c28ad", "span_id": "e054971490f538cf", "trace_flags": "01", "request": {"amount": {"currencyCode": "USD", "units": "low": 5984, "high": 0, "unsigned": false}, "nanos": "799999983", "creditCard": {"creditCardNumber": "+4485-0816-2241-7989", "cardType": "CreditCard", "lastFourDigits": "7989", "cardType": "CreditCard", "lastFourDigits": "7989", "cardExpirationYear": "2039", "creditCardExpirationMonth": "51"}, "msg": "Charge request received."}, {"level": "30", "time": "17328256313165", "pid": "17", "hostname": "opentelemetry-demo-paymentservice-857974bcd-b2fdh", "trace_id": "30517ce2a9295d98e29690bfaf3a51", "span_id": "e4d3f7259ca8136", "trace_flags": "01", "transactionId": "3559acfc-0d55-408c-941b-402fc7f2977", "cardType": "Visa", "lastFourDigits": "79687", "amount": {"units": {"low": 1830, "high": 0, "unsigned": false}, "nanos": "999999993", "currencyCode": "USD"}, "msg": "Transaction complete."}, {"level": "30", "time": "1732825643237", "pid": "17", "hostname": "opentelemetry-demo-paymentservice-857974bcd-b2fdh", "trace_id": "d108b341b28ae1f6bde34abf6c28ad", "span_id": "e054971490f538cf", "trace_flags": "01", "transId": "a2fa2f3d-691c-4d35-8de3-02a2fafb6baa", "cardType": "Visa", "lastFourDigits": "3547", "amount": {"units": {"low": 5984, "high": 0, "unsigned": false}, "nanos": "789999983", "currencyCode": "USD"}, "msg": "Transaction complete."}, {"level": "30", "time": "1732825646116", "pid": "17", "hostname": "opentelemetry-demo-paymentservice-857974bcd-b2fdh", "trace_id": "d108b341b28ae1f6bde34abf6c28ad", "span_id": "97beb1292fcba48", "trace_flags": "01", "request": {"amount": {"currencyCode": "USD", "units": "low": 512, "high": 0, "unsigned": false}, "nanos": "799999996", "creditCard": {"creditCardNumber": "+9239-5431-0337-5647", "cardType": "CreditCard", "lastFourDigits": "5647", "cardType": "CreditCard", "lastFourDigits": "5647", "cardExpirationYear": "2039", "creditCardExpirationMonth": "61"}, "msg": "Charge request received."}, {"level": "30", "time": "1732825646116", "pid": "17", "hostname": "opentelemetry-demo-paymentservice-857974bcd-b2fdh", "trace_id": "d108b341b28ae1f6bde34abf6c28ad", "span_id": "97beb1292fcba48", "trace_flags": "01", "transId": "7cfc99ad-03ee-4e53-9105-2de3c92c868", "cardType": "Visa", "lastFourDigits": "5647", "amount": {"units": {"low": 512, "high": 0, "unsigned": false}, "nanos": "799999996", "currencyCode": "USD"}, "msg": "Transaction complete."}, {"level": "30", "time": "1732825874845", "pid": "17", "hostname": "opentelemetry-demo-paymentservice-857974bcd-b2fdh", "trace_id": "f535eb95924c836f48c315655ed6", "span_id": "a5d86669efc03916", "trace_flags": "01", "request": {"amount": {"currencyCode": "USD", "units": "low": 292, "high": 0, "unsigned": false}, "nanos": "999999999", "creditCard": {"creditCardNumber": "+4532-4211-7439-1278", "cardType": "CreditCard", "lastFourDigits": "1278", "cardType": "CreditCard", "lastFourDigits": "1278", "cardExpirationYear": "2039", "creditCardExpirationMonth": "21"}, "msg": "Charge request received."}, {"level": "30", "time": "1732825874845", "pid": "17", "hostname": "opentelemetry-demo-paymentservice-857974bcd-b2fdh", "trace_id": "f535eb95924c836f48c315655ed6", "span_id": "a5d86669efc03916", "trace_flags": "01", "transId": "9eb3c096-7f02-4585-bf91-4b7d88eacf", "cardType": "Visa", "lastFourDigits": "1278", "amount": {"units": {"low": 292, "high": 0, "unsigned": false}, "nanos": "999999999", "currencyCode": "USD"}, "msg": "Transaction complete."}, {"level": "30", "time": "1732825466832", "pid": "17", "hostname": "opentelemetry-demo-paymentservice-857974bcd-b2fdh", "trace_id": "7ce94583f68ac4dd4b82828342868082", "span_id": "cb90813bebbdb60", "trace_flags": "01", "request": {"amount": {"currencyCode": "USD", "units": "low": 1449, "high": 0, "unsigned": false}, "nanos": "799999997", "creditCard": {"creditCardNumber": "+4485-0816-2241-7989", "cardType": "CreditCard", "lastFourDigits": "7989", "cardType": "CreditCard", "lastFourDigits": "7989", "cardExpirationYear": "2039", "creditCardExpirationMonth": "51"}, "msg": "Charge request received."}
```

"79974fcdf", "trace\_flags": "0x1", "transactionId": "65dc3f8f-3105-4c09-9a9e-247d127bc4e", "cardType": "visa", "lastFourDigits": "3657", "amount": {"units": "low": 644, "high": 0, "unsigned": false}, "nanos": "849999999", "currencyCode": "USD", "usd": "644", "msg": "Transaction complete."}, {"level": "30", "time": "1732872868084", "pid": "17", "hostname": "opentelemetry-demo-paymentservice-857974bcd-b2df4", "trace\_id": "f797f7bdcfa864408986fb2b5379ac66", "span\_id": "7644999", "creditCard": {"creditCardNumber": "+4916-0816-6217-7968", "creditCardExpirationYear": "2039", "creditCardExpirationMonth": "5"}, "msg": "Charge request received."}, {"level": "30", "time": "1732872868084", "pid": "17", "hostname": "opentelemetry-demo-paymentservice-857974bcd-b2df4", "trace\_id": "f797f7bdcfa864408986fb2b5379ac66", "span\_id": "7644999", "creditCard": {"creditCardNumber": "+3552f34-0330-4ae4-8384-ed3ce1a2231", "cardType": "visa", "lastFourDigits": "7968", "amount": {"units": "low": 3233, "high": 0, "unsigned": false}, "nanos": "599999998", "currencyCode": "USD", "usd": "3233", "msg": "Transaction complete."}, {"level": "30", "time": "1732872869399", "pid": "17", "hostname": "opentelemetry-demo-paymentservice-857974bcd-b2df4", "trace\_id": "1902797881da098aa363662e1d18e1b", "span\_id": "279e08a250ff64f47", "trace\_flags": "0x1", "request": {"amount": {"currencyCode": "USD", "units": "low": 501, "high": 0, "unsigned": false}, "nanos": "999999996", "creditCard": {"creditCardNumber": "+4845-4803-8707-3547", "creditCardExpirationYear": "2039", "creditCardExpirationMonth": "9"}, "msg": "Charge request received."}, {"level": "30", "time": "1732872869402", "pid": "17", "hostname": "opentelemetry-demo-paymentservice-857974bcd-b2df4", "trace\_id": "1902797881da098aa363662e1d18e1b", "span\_id": "279e08a250ff64f47", "trace\_flags": "0x1", "request": {"amount": {"currencyCode": "USD", "units": "low": 501, "high": 0, "unsigned": false}, "nanos": "999999996", "creditCard": {"creditCardNumber": "+e8c3886-4476-4999-bdf1-393961bf9d92", "cardType": "visa", "lastFourDigits": "3547", "amount": {"units": "low": 501, "high": 0, "unsigned": false}, "nanos": "399999999", "currencyCode": "USD", "usd": "399999999", "msg": "Transaction complete."}, {"level": "30", "time": "1732872870776", "pid": "17", "hostname": "opentelemetry-demo-paymentservice-857974bcd-b2df4", "trace\_id": "8ab37ebbf85dccc60a1ufc4e3d93ba0", "span\_id": "7a9af3c543ab8e83f", "trace\_flags": "0x1", "request": {"amount": {"currencyCode": "USD", "units": "low": 119, "high": 0, "unsigned": false}, "nanos": "399999999", "creditCard": {"creditCardNumber": "+5329-1102-5661-7883", "creditCardExpirationYear": "2039", "creditCardExpirationMonth": "11"}, "msg": "Charge request received."}, {"level": "30", "time": "173287287287870", "pid": "17", "hostname": "opentelemetry-demo-paymentservice-857974bcd-b2df4", "trace\_id": "83ab37ebbf85dccc60a1ufc4e3d93ba0", "span\_id": "7a9af3c543ab8e83f", "trace\_flags": "0x1", "request": {"amount": {"currencyCode": "USD", "units": "low": 119, "high": 0, "unsigned": false}, "nanos": "399999999", "creditCard": {"creditCardNumber": "+5329-1102-5661-7883", "creditCardExpirationYear": "2039", "creditCardExpirationMonth": "11"}, "msg": "Charge request received."}, {"level": "30", "time": "1732872881151", "pid": "17", "hostname": "opentelemetry-demo-paymentservice-857974bcd-b2df4", "trace\_id": "7845cd56f754595bb98e957bab5c3", "span\_id": "b10bfdf08d34f315", "trace\_flags": "0x1", "request": {"amount": {"currencyCode": "USD", "units": "low": 4157, "high": 0, "unsigned": false}, "nanos": "599999999", "creditCard": {"creditCardNumber": "+4432-8015-6152-0454", "creditCardExpirationYear": "2039", "creditCardExpirationMonth": "11"}, "msg": "Charge request received."}, {"level": "30", "time": "1732872881151", "pid": "17", "hostname": "opentelemetry-demo-paymentservice-857974bcd-b2df4", "trace\_id": "7845cd56f754595bb98e957bab5c3", "span\_id": "b10bfdf08d34f315", "trace\_flags": "0x1", "request": {"amount": {"currencyCode": "USD", "units": "low": 4157, "high": 0, "unsigned": false}, "nanos": "599999999", "creditCard": {"creditCardNumber": "+4432-8015-6152-0454", "creditCardExpirationYear": "2039", "creditCardExpirationMonth": "11"}, "msg": "Charge request received."}, {"level": "30", "time": "1732872882856", "pid": "17", "hostname": "opentelemetry-demo-paymentservice-857974bcd-b2df4", "trace\_id": "bab400889428e034ed502cde73f9f7c", "span\_id": "4904fa3cd2df1f6f", "trace\_flags": "0x1", "request": {"amount": {"currencyCode": "USD", "units": "low": 1862, "high": 0, "unsigned": false}, "nanos": "849999989", "creditCard": {"creditCardNumber": "+4929-5431-0337-5647", "creditCardExpirationYear": "2039", "creditCardExpirationMonth": "6"}, "msg": "Charge request received."}, {"level": "30", "time": "1732872892856", "pid": "17", "hostname": "opentelemetry-demo-paymentservice-857974bcd-b2df4", "trace\_id": "bab400889428e034ed502cde73f9f7c", "span\_id": "4904fa3cd2df1f6f", "trace\_flags": "0x1", "request": {"amount": {"currencyCode": "USD", "units": "low": 1862, "high": 0, "unsigned": false}, "nanos": "849999989", "creditCard": {"creditCardNumber": "+4929-5431-0337-5647", "creditCardExpirationYear": "2039", "creditCardExpirationMonth": "6"}, "msg": "Charge request received."}, {"level": "30", "time": "173287289707030", "pid": "17", "hostname": "opentelemetry-demo-paymentservice-857974bcd-b2df4", "trace\_id": "e25164f7c9e5975167865869875b521", "span\_id": "a8890c76d02957513", "trace\_flags": "0x1", "request": {"amount": {"currencyCode": "USD", "units": "low": 26233, "high": 0, "unsigned": false}, "nanos": "339999995", "creditCard": {"creditCardNumber": "+4485-1103-5661-7883", "creditCardExpirationYear": "2039", "creditCardExpirationMonth": "4"}, "msg": "Charge request received."}, {"level": "30", "time": "1732872907030", "pid": "17", "hostname": "opentelemetry-demo-paymentservice-857974bcd-b2df4", "trace\_id": "e25164f7c9e5975167865869875b521", "span\_id": "a8890c76d02957513", "trace\_flags": "0x1", "request": {"amount": {"currencyCode": "USD", "units": "low": 26233, "high": 0, "unsigned": false}, "nanos": "339999995", "creditCard": {"creditCardNumber": "+4485-1103-5661-7883", "creditCardExpirationYear": "2039", "creditCardExpirationMonth": "4"}, "msg": "Charge request received."}, {"level": "30", "time": "1732872926959", "pid": "17", "hostname": "opentelemetry-demo-paymentservice-857974bcd-b2df4", "trace\_id": "e70cd269346e331bd5b3f53b34128e7", "span\_id": "a3952d042079c53", "trace\_flags": "0x1", "request": {"amount": {"currencyCode": "USD", "units": "low": 5714, "high": 0, "unsigned": false}, "nanos": "749999998", "creditCard": {"creditCardNumber": "+4485-4803-8707-3547", "creditCardExpirationYear": "2039", "creditCardExpirationMonth": "9"}, "msg": "Charge request received."}, {"level": "30", "time": "1732872926959", "pid": "17", "hostname": "opentelemetry-demo-paymentservice-857974bcd-b2df4", "trace\_id": "e70cd269346e331bd5b3f53b34128e7", "span\_id": "a3952d042079c53", "trace\_flags": "0x1", "request": {"amount": {"currencyCode": "USD", "units": "low": 5714, "high": 0, "unsigned": false}, "nanos": "749999998", "creditCard": {"creditCardNumber": "+4485-4803-8707-3547", "creditCardExpirationYear": "2039", "creditCardExpirationMonth": "9"}, "msg": "Charge request received."}, {"level": "30", "time": "1732872926959", "pid": "17", "hostname": "opentelemetry-demo-paymentservice-857974bcd-b2df4", "trace\_id": "5ec1469f9d3-42d4-b24e-7f2475586210", "span\_id": "a3952d042079c53", "trace\_flags": "0x1", "request": {"amount": {"currencyCode": "USD", "units": "low": 5714, "high": 0, "unsigned": false}, "nanos": "749999998", "creditCard": {"creditCardNumber": "+5ec1469f9d3-42d4-b24e-7f2475586210", "cardType": "visa", "lastFourDigits": "3547", "amount": {"units": "low": 5714, "high": 0, "unsigned": false}, "nanos": "749999998", "currencyCode": "USD", "usd": "5714", "msg": "Transaction complete."}, {"level": "30", "time": "1732872926959", "pid": "17", "hostname": "opentelemetry-demo-paymentservice-857974bcd-b2df4", "trace\_id": "5ec1469f9d3-42d4-b24e-7f2475586210", "span\_id": "a3952d042079c53", "trace\_flags": "0x1", "request": {"amount": {"currencyCode": "USD", "units": "low": 5714, "high": 0, "unsigned": false}, "nanos": "749999998", "creditCard": {"creditCardNumber": "+5ec1469f9d3-42d4-b24e-7f2475586210", "cardType": "visa", "lastFourDigits": "3547", "amount": {"units": "low": 5714, "high": 0, "unsigned": false}, "nanos": "749999998", "currencyCode": "USD", "usd": "5714", "msg": "Transaction complete."}

Cartservice

```
CartService
    GetCartSync called with userId=8c69f7d5-a554-4dc7-aad0-1363d5993305
info: cartservice.cartstore.ValkeyCartStore[0]
    GetCartAsync called with userId=e3003d1f-e609-45f3-9127-fcaabedead71
info: cartservice.cartstore.ValkeyCartStore[0]
    GetCartAsync called with userId=8c69f7d5-a554-4dc7-aad0-1363d5993305
info: cartservice.cartstore.ValkeyCartStore[0]
    GetCartAsync called with userId=8c69f7d5-a554-4dc7-aad0-1363d5993305
info: cartservice.cartstore.ValkeyCartStore[0]
    GetCartAsync called with userId=e3003d1f-e609-45f3-9127-fcaabedead71
info: cartservice.cartstore.ValkeyCartStore[0]
    GetCartAsync called with userId=a775728a-a331-4b91-9918-1622bf698d36
info: cartservice.cartstore.ValkeyCartStore[0]
    GetCartAsync called with userId=a775728a-a331-4b91-9918-1622bf698d36
info: cartservice.cartstore.ValkeyCartStore[0]
    GetCartAsync called with userId=b366d762-b437-495f-a995-ae51618728a1
info: cartservice.cartstore.ValkeyCartStore[0]
    GetCartAsync called with userId=b366d762-b437-495f-a995-ae51618728a1
info: cartservice.cartstore.ValkeyCartStore[0]
    GetCartAsync called with userId=6ce8e540-8492-45ca-b36f-38c2d437f2de
info: cartservice.cartstore.ValkeyCartStore[0]
    GetCartAsync called with userId=6ce8e540-8492-45ca-b36f-38c2d437f2de
info: cartservice.cartstore.ValkeyCartStore[0]
    GetCartAsync called with userId=28fa1ead-dcce-462e-8939-18105610c688
info: cartservice.cartstore.ValkeyCartStore[0]
    GetCartAsync called with userId=28fa1ead-dcce-462e-8939-18105610c688
info: cartservice.cartstore.ValkeyCartStore[0]
    AddItemAsync called with userId=e9968f6e-adc9-11ef-b32a-f67e1a10e378, productId=66VCHSJNUP, quantity=5
info: cartservice.cartstore.ValkeyCartStore[0]
    GetCartAsync called with userId=e9968f6e-adc9-11ef-b32a-f67e1a10e378
info: cartservice.cartstore.ValkeyCartStore[0]
    AddItemAsync called with userId=e998f4a2-adc9-11ef-b32a-f67e1a10e378, productId=9SIQT8T0J0, quantity=2
info: cartservice.cartstore.ValkeyCartStore[0]
    GetCartAsync called with userId=e998f4a2-adc9-11ef-b32a-f67e1a10e378
```

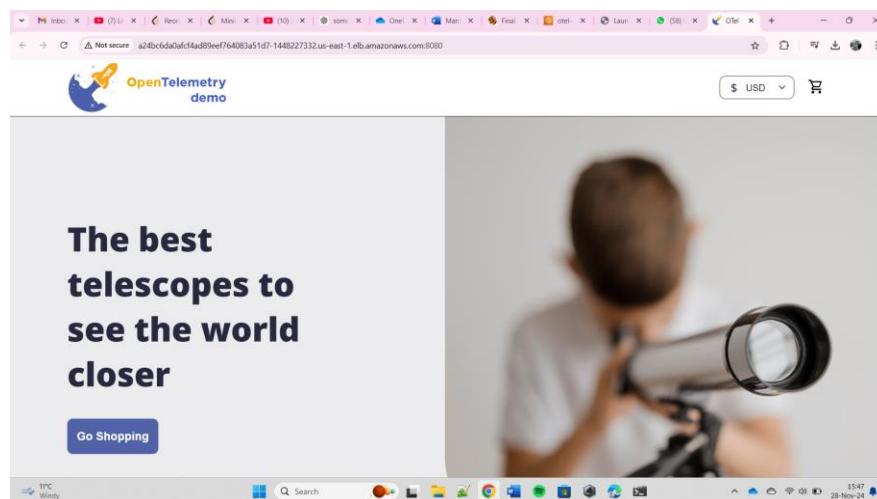
## Shipping Service

```
[ec2-user@ip-10-0-1-11 kubernetes]$ kubectl logs pod/opentelemetry-demo-shippingservice-fcfc7765-98hz -n otel-demo
20:22:56 [INFO] Otel pipeline created
20:22:56 [INFO] listening on 0.0.0.0:8080
20:24:23 [INFO] Received quote: "1225.6"
20:24:23 [INFO] Sending Quote: 1225.59
20:24:23 [INFO] Received quote: "996.6"
20:24:23 [INFO] Sending Quote: 996.60
20:24:23 [INFO] Tracking ID Created: 81221d74-b112-4f6d-956d-a83f45770553
20:24:23 [INFO] Tracking ID Created: f0c388dd-6f60-46cc-920f-9e1d04934d1c
20:24:26 [INFO] Received quote: "281.6"
20:24:26 [INFO] Sending Quote: 281.60
20:24:26 [INFO] Tracking ID Created: 7bc9d672-851f-4c0f-9f50-62d6442f23fd
20:24:47 [INFO] Received quote: "267"
20:24:47 [INFO] Sending Quote: 267.0
20:24:47 [INFO] Tracking ID Created: 45c2a75a-b4c5-4233-9097-3112f8d65c14
20:24:56 [INFO] Received quote: "389"
20:24:56 [INFO] Sending Quote: 389.0
20:24:56 [INFO] Tracking ID Created: 7a6f1368-8a1f-403c-aaf3-19abf2a39b36
20:25:17 [INFO] Received quote: "339.6"
20:25:17 [INFO] Sending Quote: 339.60
20:25:17 [INFO] Tracking ID Created: 008c5a5c-30b9-4200-b1a1-da07edf11610
20:25:44 [INFO] Received quote: "128.4"
20:25:44 [INFO] Sending Quote: 128.40
20:25:44 [INFO] Tracking ID Created: 21eb88df-23d1-4908-963f-3e0198b21c0b
20:25:59 [INFO] Received quote: "283.1"
20:25:59 [INFO] Sending Quote: 283.10
20:26:58 [INFO] Tracking ID Created: 4e50d228-4392-42ce-9162-652bbad4683d
20:26:47 [INFO] Received quote: "1588"
20:26:47 [INFO] Sending Quote: 1588.0
20:26:49 [INFO] Tracking ID Created: 6fcf7c64-8075-401b-87ee-18addbd6e3c
20:26:49 [INFO] Received quote: "441.0"
20:26:49 [INFO] Sending Quote: 441.0
20:26:51 [INFO] Tracking ID Created: 77d9ca7a-f4a8-423b-b526-9c40522091b3
20:26:51 [INFO] Received quote: "324.4"
20:26:51 [INFO] Sending Quote: 324.39
20:26:51 [INFO] Tracking ID Created: c83b515d-900d-4a10-80f5-709c59b4f471
20:27:19 [INFO] Received quote: "455"
20:27:19 [INFO] Sending Quote: 455.0
20:27:19 [INFO] Tracking ID Created: 264892d3-d88e-424c-9e51-558ce0e2e3ba
```

## 1.2.5 Application Accessibility

```
[ec2-user@ip-10-0-1-11 kubernetes]$ kubectl get svc opentelemetry-demo-frontendproxy -n otel-demo
[ec2-user@ip-10-0-1-11 kubernetes]$ 
AGE
opentelemetry-demo-frontendproxy LoadBalancer 172.20.221.99 a24bc6da0afc4ad89eef764083a51d7-1448227332.us-east-1.elb.amazonaws.com 8080:30994/TCP 36m
[ec2-user@ip-10-0-1-11 kubernetes]$
```

Accessing the application via the LoadBalancer's EXTERNAL-IP:8080



The first telescope that uses your smartphone to analyze 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

**\$ 349.95**

Quantity  
1

Add To Cart

Your order is complete!

We've sent you a confirmation email.

Starsense Explorer Refractor Telescope	<b>Shipping Data</b> Street: 1600 Amphitheatre Parkway See More	✓ Done
Quantity: 1	Total: \$ 349.95	

Continue Shopping

## **2. YAML Splitting and Modular Deployment**

In this phase, we try to modularize the application deployment by splitting the monolithic opentelemetry-demo.yaml file into smaller, logical components. By dividing the configuration into distinct, service-specific YAML files, we can ensure better deployment flexibility, ease of debugging, and more manageable updates. This also allows for deploying resources independently based on their requirements, reducing the risk of affecting other services.

### **2.1 Original State: Monolithic opentelemetry-demo.yaml**

Initially, the deployment process used a single, monolithic opentelemetry-demo.yaml file containing all the Kubernetes resources for the entire application.

#### **2.1.1 The Problem with a Single opentelemetry-demo.yaml File**

While convenient for a quick, one-step deployment, it introduced several issues:

- **Hard to Debug:** When one resource failed (e.g., a pod didn't start or a service wasn't accessible), identifying and resolving the issue was challenging since the entire file had to be reapplied.
- **Resource Dependency Problems:** Some resources (like ConfigMaps and Secrets) are prerequisites for Deployments, so a misconfiguration or race condition could cause failures.
- **High Risk of Unintended Changes:** Reapplying the entire file to fix a single issue would potentially affect resources that were already functioning correctly, causing unnecessary disruptions.

#### **2.1.2 Why We Need to Split the YAML File**

The need for splitting the original YAML file arises from several challenges:

- **Simplified Debugging:** By splitting the YAML file, each resource can be deployed or redeployed individually, allowing for easier identification of issues.
- **Targeted Deployment:** In complex environments with multiple interdependent services, resources can be applied selectively. If a failure occurs, only the affected resources need to be redeployed, instead of restarting everything.
- **Clearer Resource Management:** With the separation of resources, each component can be updated and versioned independently, enhancing maintainability and tracking.

## 2.2 First Approach: Dividing by Resource Types

The initial approach involved dividing the YAML files by resource type, such as:

- 01-namespaces/
- 02-services/
- 03-deployments/
- 04-configmaps/
- 05-secrets/

While this strategy worked in some cases, it had its shortcomings:

- **Scattered Context:** Resources related to a single service (e.g., Grafana) were placed in multiple folders (e.g., ServiceAccount in services/, ConfigMap in configmaps/, etc.). This made it difficult to track and manage all the components related to a single service.
- **Deployment Complexity:** To deploy all resources for a service, it required navigating across multiple folders. This added complexity and risked missing dependencies.

## 2.3 Second Approach: Dividing by Services

A more logical and efficient approach was adopted by grouping resources based on the service they pertain to. For example:

- 03-opentelemetry-collector/
  - Contains YAML files for deploying the OpenTelemetry collector, such as service account, config map, cluster roles, and deployment.
- 04-jaeger/
  - Contains files for the Jaeger service, including service account, service, and deployment.
- 06-grafana/
  - All Grafana-related files, including service account, secrets, config map, role bindings, deployment, etc.

### 2.3.1 Advantages of this Approach

This approach allowed the following benefits:

- **Service-Specific Grouping:** All resources related to a specific service are now located in a single folder, making it easier to manage, debug, and understand.
- **Simpler Deployment:** By grouping resources for each service together, it is easier to deploy them as a cohesive unit, reducing the need to jump between multiple folders.

### 2.3.2 Namespace Management and Deployment Order

A critical aspect of deploying resources in Kubernetes is ensuring the correct order of operations, especially when dependencies exist between resources. Kubernetes validates and applies resources in a specific order:

1. **Namespace and Service Accounts:** These are the first resources to be deployed, as they establish the namespace and required access control.
2. **Core Services:** Resources that provide foundational services (e.g., OpenTelemetry Collector, Jaeger) are deployed next.
3. **Visualization Tools:** Tools like Grafana and Prometheus are deployed after core services, as they rely on the services being operational.

We adopted a numbered pattern and naming convention to organize the YAML files and ensure a clear, logical order for resource deployment. This system helps ensure that Kubernetes resources are applied in the correct order, respecting dependencies between different components.

Below is a more detailed explanation of the naming convention and numbering pattern:

### A) Numbered Folder Structure

Each folder in the otel-demo-yaml directory has been numbered sequentially, starting from 01-08, to represent the order in which they should be deployed. This numbering system is not only intuitive but also prevents mistakes when applying the YAML files in a large, interconnected system.

```
otel-demo-yaml/
├── 01-namespaces          (Namespace setup comes first)
├── 02-core                 (Core resources like service accounts)
├── 03-opentelemetry-collector (OpenTelemetry Collector resources)
├── 04-jaeger               (Jaeger deployment)
├── 05-prometheus           (Prometheus setup)
├── 06-grafana              (Grafana resources)
├── 07-opensearch            (OpenSearch deployment)
└── 08-application          (Application-related services and deployments)
```

### B) Numbered Files Within Folders

Inside each folder, the YAML files are also numbered sequentially (e.g., 01, 02, 03, etc.). This ensures that resources within each folder are applied in the correct order as well.

#### *Why Number the Files?*

- **Handling Dependencies Between Resources:** Some Kubernetes resources depend on others. For example, a service account (e.g., 01-serviceaccount.yaml) must exist before a deployment (e.g., 02-deployment.yaml) that uses that service account can be applied. By numbering the files, we can ensure that Kubernetes applies them in the correct sequence.
- **Logical Flow:** The order of resource application is logical and helps avoid conflicts or race conditions in the cluster. Each resource that is listed first is assumed to be foundational or required by the next resource.

Example in the 03-opentelemetry-collector/ folder:

```
03-opentelemetry-collector/
├── 01-otelcol-serviceaccount.yaml    (Service account is created first)
├── 02-otelcol-configmap.yaml         (ConfigMap for the collector)
├── 03-otelcol-clusterrole.yaml      (ClusterRole for permissions)
├── 04-otelcol-clusterrolebinding.yaml (Binding the role to the service account)
├── 05-otelcol-service.yaml          (OpenTelemetry Collector Service)
└── 06-otelcol-deployment.yaml       (Finally, the deployment of OpenTelemetry Collector)
```

### C) Deployment Flow Based on Numbering

When deploying the resources, the numbering pattern ensures that Kubernetes checks the folder and file sequence, and applies them in a structured way:

1. **Namespaces** are created first, followed by any foundational resources like service accounts or roles.
2. Then, **core services** such as OpenTelemetry Collector, Jaeger, and Prometheus are deployed to provide essential telemetry and monitoring.
3. **Visualization and storage services**, such as Grafana and OpenSearch, are deployed after the core monitoring components are set up.
4. Finally, **application-related services** (e.g., frontend, cartservice, paymentservice, etc.) are deployed. These are dependent on the monitoring and telemetry services, so they are deployed after those foundational resources are established.

### D) Dependencies Between Resources

The numbering system also helps manage dependencies in Kubernetes resources.

For instance:

- A service account must exist before a deployment using that service account.
- A ConfigMap must be created before it can be referenced by a deployment or pod.
- A role and rolebinding must be applied before any pods can make use of them.

```
03-opentelemetry-collector/
├── 01-otelcol-serviceaccount.yaml    --> Service account needed before any pods use it
├── 02-otelcol-configmap.yaml        --> ConfigMap for application configuration
├── 03-otelcol-clusterrole.yaml     --> Role permissions
├── 04-otelcol-clusterrolebinding.yaml --> Binding the role to the service account
├── 05-otelcol-service.yaml         --> Service definition for communication
└── 06-otelcol-deployment.yaml      --> Deployment referencing above resources
```

This numbering pattern creates a natural progression for the resources, ensuring that all dependencies are met before the next step in the deployment process.

The split approach resulted in:

- **28 Directories**
- **67 Files** in total, each representing a discrete Kubernetes resource.

The final structure grouped the resources in a logical, hierarchical way. For example, all files related to OpenTelemetry Collector were grouped under 03-opentelemetry-collector, and all services for individual application components (e.g., frontend, backend, payment service) were grouped under 08-application, each with their own subfolder.

## **E) Explanation of the file structure and explaining the purpose of each resource:**

The original single YAML file opentelemetry-demo.yaml was broken down into several smaller files, each representing a distinct Kubernetes resource. These files were then organized into folders based on the type of resource they define.

### **Folder Structure and Total Files**

- **Root Directory:** otel-demo-yaml/
  - **Total Number of Files:** 67
  - **Total Number of Directories:** 28

### **Folder Breakdown:**

1. **01-namespaces**
  - **Files:**
    - 01-otel-demo-namespace.yaml
  - **Purpose:** Defines the otel-demo namespace.
2. **02-core**
  - **Files:**
    - serviceaccount.yaml
  - **Purpose:** Defines core resources like service accounts and base permissions.
3. **03-opentelemetry-collector**
  - **Files:**
    - 01-otelcol-serviceaccount.yaml
    - 02-otelcol-configmap.yaml
    - 03-otelcol-clusterrole.yaml
    - 04-otelcol-clusterrolebinding.yaml
    - 05-otelcol-service.yaml
    - 06-otelcol-deployment.yaml
  - **Purpose:** Resources related to the OpenTelemetry Collector service.
4. **04-jaeger**
  - **Files:**
    - 01-jaeger-serviceaccount.yaml
    - 02-jaeger-service.yaml
    - 03-jaeger-deployment.yaml
  - **Purpose:** Resources related to Jaeger for distributed tracing.
5. **05-prometheus**
  - **Files:**
    - 01-prometheus-serviceaccount.yaml
    - 02-prometheus-configmap.yaml
    - 03-prometheus-clusterrole.yaml
    - 04-prometheus-clusterrolebinding.yaml
    - 05-prometheus-service.yaml
    - 06-prometheus-deployment.yaml
  - **Purpose:** Resources related to Prometheus for monitoring.

## 6. 06-grafana

- **Files:**

- 01-grafana-serviceaccount.yaml
- 02-grafana-secret.yaml
- 03-grafana-configmap.yaml
- 04-grafana-clusterrole.yaml\
- 05-grafana-clusterrolebinding.yaml
- 06-grafana-role.yaml
- 07-grafana-rolebinding.yaml
- 08-grafana-service.yaml
- 09-grafana-deployment.yaml

- **Purpose:** Resources related to Grafana for visualization and monitoring.

## 7. 07-opensearch

- **Files:**

- 01-opensearch-poddisruptionbudget.yaml
- 02-opensearch-configmap.yaml
- 03-opensearch-service.yaml
- 04-opensearch-statefulset.yaml

- **Purpose:** Resources related to OpenSearch for logs and data storage.

## 8. 08-application

- **Files:** A collection of files for various application services such as accountingservice, adservice, cartservice, etc. These include service and deployment files for each microservice.
- **Purpose:** Each subfolder contains the deployment and service definitions for a specific application service.

## Screenshots of the created folder structure containing the split YAML –

```
manavgupta@Manavs-MacBook-Pro:~/Documents/UMD/OneDrive - University of Maryland/UMD Courses/Sem 3/ENPM 818N Cloud Computing/endsem/cloud818n-group11-endsem/kubernetes
$ tree otel-demo-yml
+-- subfolders git:(main) * tree otel-demo-yml
|-- 01-namespaces
|   |-- 01-otel-namespace.yaml
|   |-- 02-otel-serviceaccount.yaml
|   |-- 03-otel-metrics-collector
|       |-- 01-otel-col-serviceaccount.yaml
|       |-- 02-otel-col-serviceaccount-role.yaml
|       |-- 03-otel-col-clusterrole.yaml
|       |-- 04-otel-col-clusterrolebinding.yaml
|       |-- 05-otel-col-service.yaml
|       |-- 06-otel-col-deployment.yaml
|   |-- 04-jaeger
|       |-- 01-jaeger-serviceaccount.yaml
|       |-- 02-jaeger-service.yaml
|       |-- 03-jaeger-deployment.yaml
|   |-- 05-prometheus
|       |-- 01-prometheus-serviceaccount.yaml
|       |-- 02-prometheus-configmap.yaml
|       |-- 03-prometheus-clusterrole.yaml
|       |-- 04-prometheus-clusterrolebinding.yaml
|       |-- 05-prometheus-service.yaml
|       |-- 06-prometheus-deployment.yaml
|   |-- 06-grafana
|       |-- 01-grafana-serviceaccount.yaml
|       |-- 02-grafana-secret.yaml
|       |-- 03-grafana-configmap.yaml
|       |-- 04-grafana-clusterrole.yaml
|       |-- 05-grafana-clusterrolebinding.yaml
|       |-- 06-grafana-role.yaml
|       |-- 07-grafana-rolebinding.yaml
|       |-- 08-grafana-service.yaml
|       |-- 09-grafana-deployment.yaml
|   |-- 07-opensearch
|       |-- 01-opensearch-poddisruptionbudget.yaml
|       |-- 02-opensearch-configmap.yaml
|       |-- 03-opensearch-service.yaml
|       |-- 04-opensearch-statefulset.yaml
|   |-- 08-application
|       |-- accountingservice
|           |-- 01-accountingservice-deployment.yaml
|       |-- orderservice
|           |-- 01-orderservice-service.yaml
|           |-- 02-orderservice-deployment.yaml
|       |-- cartservice
|           |-- 01-cartservice-service.yaml
|           |-- 02-cartservice-deployment.yaml
|       |-- checkoutservice
|           |-- 01-checkoutservice-service.yaml
|           |-- 02-checkoutservice-deployment.yaml
|       |-- currencyservice
|           |-- 01-currencyservice-service.yaml
|           |-- 02-currencyservice-deployment.yaml
|       |-- emilservice
|           |-- 01-emilservice-service.yaml
|           |-- 02-emilservice-deployment.yaml
|       |-- flag
|           |-- 01-flag-configmap.yaml
|
|-- 09-kafka
    |-- 01-kafka-service.yaml
    |-- 02-kafka-deployment.yaml
|-- loggenerator
    |-- 01-loggenerator-service.yaml
    |-- 02-loggenerator-deployment.yaml
|-- paymentservice
    |-- 01-paymentservice-service.yaml
    |-- 02-paymentservice-deployment.yaml
|-- productcatalogservice
    |-- 01-productcatalogservice-service.yaml
    |-- 02-productcatalogservice-deployment.yaml
|-- quoteservice
    |-- 01-quoteservice-service.yaml
    |-- 02-quoteservice-deployment.yaml
|-- recommendationservice
    |-- 01-recommendationservice-service.yaml
    |-- 02-recommendationservice-deployment.yaml
|-- shippingservice
    |-- 01-shippingservice-service.yaml
    |-- 02-shippingservice-deployment.yaml
|-- volkey
    |-- 01-volkey-service.yaml
    |-- 02-volkey-deployment.yaml
```

```
manavgupta@Manavs-MacBook-Pro:~/Documents/UMD/OneDrive - University of Maryland/UMD Courses/Sem 3/ENPM 818N Cloud Computing/endsem/cloud818n-group11-endsem/kubernetes
$ tree
+-- accountingservice
|   |-- 01-accountingservice-deployment.yaml
+-- orderservice
|   |-- 01-orderservice-service.yaml
|   |-- 02-orderservice-deployment.yaml
+-- cartservice
|   |-- 01-cartservice-service.yaml
|   |-- 02-cartservice-deployment.yaml
+-- checkoutservice
|   |-- 01-checkoutservice-service.yaml
|   |-- 02-checkoutservice-deployment.yaml
+-- currencyservice
|   |-- 01-currencyservice-service.yaml
|   |-- 02-currencyservice-deployment.yaml
+-- emilservice
|   |-- 01-emilservice-service.yaml
|   |-- 02-emilservice-deployment.yaml
+-- flag
|   |-- 01-flag-configmap.yaml
|   |-- 02-flag-deployment.yaml
+-- fraudetectionservice
|   |-- 01-fraudetectionservice-deployment.yaml
+-- frontendservice
|   |-- 01-frontendservice-service.yaml
|   |-- 02-frontendservice-deployment.yaml
+-- frontendproxy
|   |-- 01-frontendproxy-service.yaml
|   |-- 02-frontendproxy-deployment.yaml
+-- insigntypeservice
|   |-- 01-insigntypeservice-service.yaml
|   |-- 02-insigntypeservice-deployment.yaml
+-- kafkastorage
|   |-- 01-kafka-service.yaml
|   |-- 02-kafka-deployment.yaml
+-- loggenerator
|   |-- 01-loggenerator-service.yaml
|   |-- 02-loggenerator-deployment.yaml
+-- paymentservice
|   |-- 01-paymentservice-service.yaml
|   |-- 02-paymentservice-deployment.yaml
+-- productcatalogservice
|   |-- 01-productcatalogservice-service.yaml
|   |-- 02-productcatalogservice-deployment.yaml
+-- quotestypeservice
|   |-- 01-quotestypeservice-service.yaml
|   |-- 02-quotestypeservice-deployment.yaml
+-- recommendationservice
|   |-- 01-recommendationservice-service.yaml
|   |-- 02-recommendationservice-deployment.yaml
+-- shippingservice
|   |-- 01-shippingservice-service.yaml
|   |-- 02-shippingservice-deployment.yaml
+-- volkey
|   |-- 01-volkey-service.yaml
|   |-- 02-volkey-deployment.yaml
```

28 directories, 67 files

### 2.3.3 Applying Resources Individually or Recursively

The goal here is to apply the split YAML files, either individually or recursively, to deploy the resources in a controlled manner, ensuring that each resource is deployed correctly before moving on to the next. This method also ensures that if a service encounters an issue, it can be troubleshooted and redeployed without affecting other parts of the system.

### 2.3.3.1 Individually Applying Resources

We began the deployment process by applying the namespace resources using --namespace flag to ensure that all subsequent resources were applied to the correct namespace (otel-demo).

*Command Used:*

```
kubectl apply --namespace otel-demo -f 01-namespaces/otel-demo-namespace.yaml
```

## Namespace Deployment:

```
[ec2-user@ip-10-0-1-11 otel-demo-yml]$ ll
total 4
drwxrwx--x 2 ec2-user ec2-user 38 Nov 29 01:35 otel-namespaces
drwxrwx--x 2 ec2-user ec2-user 33 Nov 29 01:35 otel-core
drwxrwx--x 2 ec2-user ec2-user 100 Nov 29 01:35 otel-metrics-collector
drwxrwx--x 2 ec2-user ec2-user 100 Nov 29 01:35 otel-prometheus
drwxrwx--x 2 ec2-user ec2-user 237 Nov 29 01:35 otel-seaeger
drwxrwx--x 2 ec2-user ec2-user 100 Nov 29 01:35 otel-groffend
drwxrwx--x 2 ec2-user ec2-user 200 Nov 29 01:35 otel-research
drwxrwx--x 21 ec2-user ec2-user 4096 Nov 29 01:35 otel-application
[ec2-user@ip-10-0-1-11 otel-demo-yml]$ kubectl apply -n namespaces otel-demo -f otel-namespaces.yaml
namespaces/otel-namespaces created
[ec2-user@ip-10-0-1-11 otel-demo-yml]$ kubectl get all -n otel-demo
No resources found in otel-demo namespace.
[ec2-user@ip-10-0-1-11 otel-demo-yml]$ [ec2-user@ip-10-0-1-11 otel-demo-yml]$
```

After ensuring that the namespace was set up, we moved on to apply core resources which has service accounts that are foundational and must be applied before any application-specific components like services and deployments.

*Command Used:*

```
kubectl apply --namespace otel-demo -f 02-core/ -R
```

## Core Deployment:

```
[ec2-user@ip-10-0-1-11 otel-demo-yaml]$ kubectl apply --namespace otel-demo -f otel-core.yaml  
serviceaccount/otel-metrics-demo created  
[ec2-user@ip-10-0-1-11 otel-demo-yaml]$ kubectl get all -n otel-demo  
No resources found in otel-demo namespace.  
[ec2-user@ip-10-0-1-11 otel-demo-yaml]$
```

We then proceeded to apply resources for each service. Each service (e.g., OpenTelemetry Collector, Prometheus, Grafana, etc.) was organized into its own folder. These folders contained all relevant resources such as service accounts, secrets, config maps, roles, services, and deployments. Finally, we applied resources for the individual application services (e.g., frontend, paymentservice, adservice, etc.). Each of these services had its own folder and contained all related Kubernetes resources (services, deployments, etc.).

## Otecol Deployment:

```
[ec2-user@ip-10-0-1-11 ~]$ kubectl apply --namespace otel-demo -f 03-opentelemetry-collector.yaml
serviceaccount/opentelemetry-demo-otelcol created
clusterrolebinding/rbac.authorization.k8s.io/opentelemetry-demo-otelcol created
clusterrolebinding/rbac.authorization.k8s.io/opentelemetry-demo-otelcol created
service/opentelemetry-demo-otelcol created
deployment.apps/opentelemetry-demo-otelcol created
deployment.apps/opentelemetry-demo-otelcol[1] $ kubectl get all -n otel-demo
NAME                                         READY   STATUS    RESTARTS   AGE
pod/opentelemetry-demo-otelcol-5c757ccfcf-6szr7l 1/1     Running   0          8s
NAME           TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
Service/opentelemetry-demo-otelcol  ClusterIP   172.26.129.226   <none>        6831/TCP,14250/TCP,14268/TCP,8888/TCP,4317/TCP,4318/TCP,9464/TCP,9411/TCP   8s
NAME           READY   UP-TO-DATE   AVAILABLE   AGE
Deployment.apps/opentelemetry-demo-otelcol  1/1     1           1           9s
NAME           DESIRED   CURRENT   READY   AGE
Replicaset.apps/opentelemetry-demo-otelcol-5c757ccfcf  1         1         1         9s
[ec2-user@ip-10-0-1-11 ~]$ kubectl get pods -n otel-demo

```

## Jaeger Deployment:

```
[ec2-user@ip-10-0-1-11 ~]$ kubectl apply -n命名空间 otel-demo -f 04-jaeger.yaml
serviceaccount/otellemetry-demo-jaeger created
service/otellemetry-demo-jaeger created
service/otellemetry-demo-jaeger-agent created
service/otellemetry-demo-jaeger-collector created
service/otellemetry-demo-jaeger-query created
deployment.apps/otellemetry-demo-jaeger created
[ec2-user@ip-10-0-1-11 ~]$ kubectl get all -n otel-demo
NAME                                     READY   STATUS    RESTARTS   AGE
pod/otellemetry-demo-jaeger-7785549b-n2bp   1/1    Running   0          7s
pod/otellemetry-demo-otelcol-5c757cf-cf652f1  1/1    Running   0          63s

NAME                           TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)          AGE
service/otellemetry-demo-jaeger-agent   ClusterIP  None        <none>        5775/UDP,5778/TCP,6831/UDP,6832/UDP   7s
service/otellemetry-demo-jaeger-collector ClusterIP  None        <none>        9411/TCP,14258/TCP,14267/TCP,14268/TCP,4317/TCP,4318/TCP   7s
service/otellemetry-demo-jaeger-query   ClusterIP  None        <none>        16686/TCP,16685/TCP   7s
service/otellemetry-demo-otelcol       ClusterIP  None        <none>        6831/UDP,14258/TCP,14268/TCP,8888/TCP,4317/TCP,4318/TCP,9464/TCP,9411/TCP   63s

NAME          READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/otellemetry-demo-jaeger   1/1    1           1           7s
deployment.apps/otellemetry-demo-otelcol  1/1    1           1           63s

NAME          DESIRED   CURRENT   READY   AGE
replicaset.apps/otellemetry-demo-jaeger-7785549b   1           1           1           7s
replicaset.apps/otellemetry-demo-otelcol-5c757cf-cf   1           1           1           63s
[ec2-user@ip-10-0-1-11 ~]$ kubectl get all -n otel-demo-yaml
```

## Prometheus Deployment:

```
[ec2-user@ip-10-0-1-11 ~]$ kubectl apply -n命名空间 otel-demo -f 05-prometheus.yaml
serviceaccount/otellemetry-demo-prometheus-server created
configmap/otellemetry-demo-prometheus-server created
clusterrole.rbac.authorization.k8s.io/otellemetry-demo-prometheus-server created
clusterrolebinding.rbac.authorization.k8s.io/otellemetry-demo-prometheus-server created
service/otellemetry-demo-prometheus-server created
deployment.apps/otellemetry-demo-prometheus-server created
[ec2-user@ip-10-0-1-11 ~]$ kubectl get all -n otel-demo
NAME                                     READY   STATUS    RESTARTS   AGE
pod/otellemetry-demo-jaeger-7785549b-n2bp   1/1    Running   0          10m
pod/otellemetry-demo-otelcol-5c757cf-cf652f1  1/1    Running   0          11m
pod/otellemetry-demo-prometheus-server-57cd8ff9d46-pfwvn  1/1    Running   0          7m48s

NAME                           TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)          AGE
service/otellemetry-demo-jaeger-agent   ClusterIP  None        <none>        5775/UDP,5778/TCP,6831/UDP,6832/UDP   10m
service/otellemetry-demo-jaeger-collector ClusterIP  None        <none>        9411/TCP,14258/TCP,14267/TCP,14268/TCP,4317/TCP,4318/TCP   10m
service/otellemetry-demo-jaeger-query   ClusterIP  None        <none>        16686/TCP,16685/TCP   10m
service/otellemetry-demo-otelcol       ClusterIP  None        <none>        6831/UDP,14258/TCP,14268/TCP,8888/TCP,4317/TCP,4318/TCP,9464/TCP,9411/TCP   10m
service/otellemetry-demo-prometheus-server       ClusterIP  None        <none>        9890/TCP   7m48s

NAME          READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/otellemetry-demo-jaeger   1/1    1           1           10m
deployment.apps/otellemetry-demo-otelcol  1/1    1           1           11m
deployment.apps/otellemetry-demo-prometheus-server  1/1    1           1           7m48s

NAME          DESIRED   CURRENT   READY   AGE
replicaset.apps/otellemetry-demo-jaeger-7785549b   1           1           1           10m
replicaset.apps/otellemetry-demo-otelcol-5c757cf-cf   1           1           1           11m
replicaset.apps/otellemetry-demo-prometheus-server-57cd8ff9d46  1           1           1           7m48s
[ec2-user@ip-10-0-1-11 ~]$ kubectl get all -n otel-demo-yaml
```

## Grafana Deployment:

```
[ec2-user@ip-10-0-1-11 ~]$ kubectl apply -n命名空间 otel-demo -f 06-grafana.yaml
serviceaccount/otellemetry-demo-grafana created
secret/otellemetry-demo-grafana created
configmap/otellemetry-demo-grafana created
clusterrole.rbac.authorization.k8s.io/otellemetry-demo-grafana-clusterrole created
clusterrolebinding.rbac.authorization.k8s.io/otellemetry-demo-grafana-clusterrolebinding created
role.rbac.authorization.k8s.io/otellemetry-demo-grafana created
rolebinding.rbac.authorization.k8s.io/otellemetry-demo-grafana created
service/otellemetry-demo-grafana created
deployment.apps/otellemetry-demo-grafana created
[ec2-user@ip-10-0-1-11 ~]$ kubectl get all -n otel-demo
NAME                                     READY   STATUS    RESTARTS   AGE
pod/otellemetry-demo-grafana-9fb6bd5dd4-tcavx  1/1    Running   0          72s
pod/otellemetry-demo-jaeger-7785549b-n2bp      1/1    Running   0          13s
pod/otellemetry-demo-otelcol-5c757cf-cf652f1   1/1    Running   0          13s
pod/otellemetry-demo-prometheus-server-57cd8ff9d46-pfwvn  1/1    Running   0          9m55s

NAME                           TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)          AGE
service/otellemetry-demo-grafana   ClusterIP  None        <none>        80/TCP   73s
service/otellemetry-demo-jaeger-agent   ClusterIP  None        <none>        5775/UDP,5778/TCP,6831/UDP,6832/UDP   13m
service/otellemetry-demo-jaeger-collector ClusterIP  None        <none>        9411/TCP,14258/TCP,14267/TCP,14268/TCP,4317/TCP,4318/TCP   13m
service/otellemetry-demo-jaeger-query   ClusterIP  None        <none>        16686/TCP,16685/TCP   13m
service/otellemetry-demo-otelcol       ClusterIP  None        <none>        6831/UDP,14258/TCP,14268/TCP,8888/TCP,4317/TCP,4318/TCP,9464/TCP,9411/TCP   13m
service/otellemetry-demo-prometheus-server       ClusterIP  None        <none>        9890/TCP   9m55s

NAME          READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/otellemetry-demo-grafana   1/1    1           1           72s
deployment.apps/otellemetry-demo-jaeger   1/1    1           1           13s
deployment.apps/otellemetry-demo-otelcol  1/1    1           1           13s
deployment.apps/otellemetry-demo-prometheus-server  1/1    1           1           9m55s

NAME          DESIRED   CURRENT   READY   AGE
replicaset.apps/otellemetry-demo-grafana-69b6bd5dd4   1           1           1           72s
replicaset.apps/otellemetry-demo-jaeger-7785549b   1           1           1           13s
replicaset.apps/otellemetry-demo-otelcol-5c757cf-cf   1           1           1           13s
replicaset.apps/otellemetry-demo-prometheus-server-57cd8ff9d46  1           1           1           9m55s
[ec2-user@ip-10-0-1-11 ~]$ kubectl get all -n otel-demo-yaml
```

## OpenSearch Deployment:

```
[ec2-user@ip-10-0-1-11 ~]$ kubectl apply -n命名空间 otel-demo -f 07-opensearch.yaml
poddisruptionbudget.policy/otel-demo-opensearch-pdb created
configmap/otel-demo-opensearch-only created
secret/otel-demo-opensearch created
service/otel-demo-opensearch-headless created
statefulset.apps/otel-demo-opensearch created
[ec2-user@ip-10-0-1-11 ~]$ kubectl get all -n otel-demo
NAME                                     READY   STATUS    RESTARTS   AGE
pod/otel-demo-opensearch-69b6bd5dd4-tcavx  1/1    Running   0          3m28s
pod/otellemetry-demo-jaeger-7785549b-n2bp   1/1    Running   0          15m
pod/otellemetry-demo-otelcol-5c757cf-cf652f1  1/1    Running   0          16m
pod/otellemetry-demo-prometheus-server-57cd8ff9d46-pfwvn  1/1    Running   0          12m
pod/otel-demo-opensearch   1/1    Running   0          86s

NAME                           TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)          AGE
service/otel-demo-opensearch   ClusterIP  None        <none>        9200/TCP,9300/TCP,9600/TCP   3m29s
service/otellemetry-demo-jaeger-agent   ClusterIP  None        <none>        5775/UDP,5778/TCP,6831/UDP,6832/UDP   1s
service/otellemetry-demo-jaeger-collector ClusterIP  None        <none>        9411/TCP,14258/TCP,14267/TCP,14268/TCP,4317/TCP,4318/TCP   15m
service/otellemetry-demo-jaeger-query   ClusterIP  None        <none>        16686/TCP,16685/TCP   15m
service/otellemetry-demo-otelcol       ClusterIP  None        <none>        6831/UDP,14258/TCP,14268/TCP,8888/TCP,4317/TCP,4318/TCP,9464/TCP,9411/TCP   15m
service/otel-demo-opensearch-server     ClusterIP  None        <none>        9200/TCP,9300/TCP,9600/TCP   86s
service/otel-demo-opensearch-headless ClusterIP  None        <none>        9200/TCP,9300/TCP,9600/TCP   86s

NAME          READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/otel-demo-opensearch   1/1    1           1           3m29s
deployment.apps/otellemetry-demo-jaeger   1/1    1           1           15m
deployment.apps/otellemetry-demo-otelcol  1/1    1           1           16m
deployment.apps/otellemetry-demo-prometheus-server  1/1    1           1           12m
deployment.apps/otel-demo-opensearch-headless  1/1    1           1           86s

NAME          DESIRED   CURRENT   READY   AGE
replicaset.apps/otel-demo-opensearch-69b6bd5dd4   1           1           1           3m29s
replicaset.apps/otellemetry-demo-jaeger-7785549b   1           1           1           15m
replicaset.apps/otellemetry-demo-otelcol-5c757cf-cf   1           1           1           16m
replicaset.apps/otellemetry-demo-prometheus-server-57cd8ff9d46  1           1           1           12m
[ec2-user@ip-10-0-1-11 ~]$ kubectl get all -n otel-demo-yaml
```

## Application Deployment:

### 2.3.3.2 Recursive Application of the Entire Structure

After confirming that all services had been applied successfully, we applied the entire project recursively, starting from the root folder of the project to deploy all the services in the correct order.

#### *Command Used:*

```
kubectl apply --namespace otel-demo -f otel-demo-yaml/ -R
```

## 2.3.4 Validate Resource Deployment

### 2.3.4.1 Checking Pod and Service Status

After applying the resources, the next step was to validate the deployment by checking the status of the pods, services, and other resources. This ensures that everything was deployed correctly and is functioning as expected.

We used the kubectl get all-n otel-demo command to check the status of all resources within the otel-demo namespace. This command lists all the pods, services, deployments, and other resources, making it easy to confirm that they were all running correctly.

```
[ec2-user@ip-10-0-1-11 kubernetes]$ kubectl get all -n otel-demo
NAME                                     READY   STATUS    RESTARTS   AGE
pod/opentelemetry-demo-accountingservice-8555c59fb-dc4ks   1/1    Running   0          59s
pod/opentelemetry-demo-accountingservice-5d4f43688-5ppj   1/1    Running   0          59s
pod/opentelemetry-demo-cartservice-694df79f7-wwms7   1/1    Running   0          59s
pod/opentelemetry-demo-checkoutservice-744bc6dd0-6tsll   1/1    Running   0          59s
pod/opentelemetry-demo-currencyservice-688fcfd9d6-582gn   1/1    Running   0          58s
pod/opentelemetry-demo-emailservice-668476f6dc-h2wkg   1/1    Running   0          58s
pod/opentelemetry-demo-flapd-67986d767-4spq   2/2    Running   0          58s
pod/opentelemetry-demo-fraudtectionservice-85d48f859f-6gypv 1/1    Running   0          58s
pod/opentelemetry-demo-frontends-598cc0fda-6d8pr   1/1    Running   0          58s
pod/opentelemetry-demo-grafana-699e6544d-5xvc   1/1    Running   0          58s
pod/opentelemetry-demo-grafana-699e6544d-bfnz2   1/1    Running   0          59s
pod/opentelemetry-demo-imagerender-7466d89fb-jktnj   1/1    Running   0          57s
pod/opentelemetry-demo-jagger-7785f96b-nnh9   1/1    Running   0          59s
pod/opentelemetry-demo-loadgenerator-6949f480-lsgaq   1/1    Running   0          57s
pod/opentelemetry-demo-logsparser-67986d767-p6z2x   1/1    Running   0          57s
pod/opentelemetry-demo-otello-5c77cf-ch1vnz   1/1    Running   0          60s
pod/opentelemetry-demo-paymentservice-857974ec0d-9ng24  1/1    Running   0          57s
pod/opentelemetry-demo-productcatalog-736bf9466-f674d   1/1    Running   0          57s
pod/opentelemetry-demo-quarantine-service-568c58fd7-c7wm  1/1    Running   0          56s
pod/opentelemetry-demo-recommendationservice-6f576fd574-fdtl 1/1    Running   0          56s
pod/opentelemetry-demo-shippingservice-fccf765-8vnzn   1/1    Running   0          56s
pod/opentelemetry-demo-valley-68b4c949b-9fdcv   1/1    Running   0          56s
pod/opentelemetry-demo-search-9   1/1    Running   0          59s

NAME                TYPE        CLUSTER-IP      EXTERNAL-IP      PORT(S)           AGE
service/opentelemetry-demo-odbservice   ClusterIP  172.20.154.173   <none>        8080/TCP          59s
service/opentelemetry-demo-cartservice  ClusterIP  172.20.75.125   <none>        8080/TCP          59s
service/opentelemetry-demo-checkoutservice ClusterIP  172.20.191.266  <none>        8080/TCP          59s
service/opentelemetry-demo-currencyservice ClusterIP  172.20.52.243   <none>        8080/TCP          59s
service/opentelemetry-demo-emailservice  ClusterIP  172.20.142.203  <none>        8080/TCP          59s
service/opentelemetry-demo-flapd   ClusterIP  172.20.117.118   <none>        8013/TCP,4000/TCP 59s
service/opentelemetry-demo-frontendsproxy ClusterIP  172.20.137.247  <none>        8080/TCP          58s
service/opentelemetry-demo-grafana   ClusterIP  172.20.8.253   <none>        8080/TCP          59s
service/opentelemetry-demo-kafka-provider ClusterIP  172.20.243.211  <none>        9301/TCP          58s
service/opentelemetry-demo-jagger-agent ClusterIP  None         <none>        5775/UDP,5778/TCP,6831/UDP,6832/UDP 60s
service/opentelemetry-demo-jagger-collector ClusterIP  None         <none>        9411/TCP,14268/TCP,14269/TCP,4311/TCP,4318/TCP 60s
service/opentelemetry-demo-jagger-gateway ClusterIP  None         <none>        10411/TCP,10685/TCP 60s
service/opentelemetry-demo-jagger-provider ClusterIP  172.20.61.171   <none>        9082/TCP,3093/TCP 58s
service/opentelemetry-demo-leasegenerator ClusterIP  172.20.249.256   <none>        8889/TCP          58s
service/opentelemetry-demo-otel   ClusterIP  172.20.249.53   <none>        6831/UDP,14248/TCP,14268/TCP,8888/TCP,4317/TCP,4318/TCP,9464/TCP,9411/TCP 60s
service/opentelemetry-demo-paymentcatalogservice ClusterIP  172.20.10.23   <none>        8080/TCP          58s
service/opentelemetry-demo-prometheus-server ClusterIP  172.20.58.78   <none>        9090/TCP          59s
service/opentelemetry-demo-quarantineservice ClusterIP  172.20.141.28   <none>        8088/TCP          58s
service/opentelemetry-demo-recommendationservice ClusterIP  172.20.20.235   <none>        8080/TCP          58s
service/opentelemetry-demo-productcatalogservice ClusterIP  172.20.10.44   <none>        8080/TCP          58s
service/opentelemetry-demo-valley   ClusterIP  172.20.19.118   <none>        6379/TCP          58s
service/opentelemetry-demo-search   ClusterIP  172.20.6.42    <none>        9200/TCP,9300/TCP,9600/TCP          59s
service/opentelemetry-demo-search-headless ClusterIP  None         <none>        9200/TCP,9300/TCP,9600/TCP          59s

NAME          READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/opentelemetry-demo-accountingservice  1/1    1          1          59s
deployment.apps/opentelemetry-demo-odbservice       1/1    1          1          59s
deployment.apps/opentelemetry-demo-cartservice      1/1    1          1          59s
deployment.apps/opentelemetry-demo-checkoutservice  1/1    1          1          59s
deployment.apps/opentelemetry-demo-emailservice     1/1    1          1          59s

NAME          READY   DESIRED   CURRENT   READY   AGE
replicaset.apps/opentelemetry-demo-accountingservice 1/1    1          1          1          59s
replicaset.apps/opentelemetry-demo-odbservice       1/1    1          1          1          59s
replicaset.apps/opentelemetry-demo-cartservice      1/1    1          1          1          59s
replicaset.apps/opentelemetry-demo-checkoutservice  1/1    1          1          1          59s
replicaset.apps/opentelemetry-demo-currencyservice  1/1    1          1          1          59s
replicaset.apps/opentelemetry-demo-emailservice     1/1    1          1          1          59s
replicaset.apps/opentelemetry-demo-flapd            1/1    1          1          1          58s
replicaset.apps/opentelemetry-demo-frontendservice  1/1    1          1          1          58s
replicaset.apps/opentelemetry-demo-frontend         1/1    1          1          1          58s
replicaset.apps/opentelemetry-demo-frontendsproxy  1/1    1          1          1          58s
replicaset.apps/opentelemetry-demo-grafana          1/1    1          1          1          58s
replicaset.apps/opentelemetry-demo-grafana-provider 1/1    1          1          1          58s
replicaset.apps/opentelemetry-demo-jagger           1/1    1          1          1          60s
replicaset.apps/opentelemetry-demo-kafka             1/1    1          1          1          58s
replicaset.apps/opentelemetry-demo-leasegenerator   1/1    1          1          1          58s
replicaset.apps/opentelemetry-demo-logsparser       1/1    1          1          1          60s
replicaset.apps/opentelemetry-demo-paymentcatalog  1/1    1          1          1          58s
replicaset.apps/opentelemetry-demo-productcatalogserver 1/1    1          1          1          59s
replicaset.apps/opentelemetry-demo-prometheus-server 1/1    1          1          1          59s
replicaset.apps/opentelemetry-demo-quarantine       1/1    1          1          1          58s
replicaset.apps/opentelemetry-demo-recommendationservice 1/1    1          1          1          58s
replicaset.apps/opentelemetry-demo-shippingservice  1/1    1          1          1          58s
replicaset.apps/opentelemetry-demo-valley           1/1    1          1          1          58s

NAME          READY   AGE
statefulset.apps/opentelemetry-demo-search          1/1    59s
[ec2-user@ip-10-0-1-11 kubernetes]$
```

```
[ec2-user@ip-10-0-1-11 kubernetes]$ kubectl get all -n otel-demo
NAME                                     READY   STATUS    RESTARTS   AGE
pod/opentelemetry-demo-recommendationservice-772c-227-295   1/1    Running   0          59s
pod/opentelemetry-demo-shippingservice-5d4f43688-5ppj       1/1    Running   0          59s
pod/opentelemetry-demo-otel-9   1/1    Running   0          59s
pod/opentelemetry-demo-search-9   1/1    Running   0          59s
pod/opentelemetry-demo-leasegenerator-7466d89fb-776d574   1/1    Running   0          59s
replicaset.apps/opentelemetry-demo-currencyservice-744bc6dd0 1/1    1          1          59s
replicaset.apps/opentelemetry-demo-emailservice-668476f6dc   1/1    1          1          58s
replicaset.apps/opentelemetry-demo-flapd-67986d767-4spq   1/1    1          1          58s
replicaset.apps/opentelemetry-demo-frontendsproxy-7498c6f64 1/1    1          1          58s
replicaset.apps/opentelemetry-demo-grafana-699e6544d         1/1    1          1          59s
replicaset.apps/opentelemetry-demo-leasegenerator-668476f6dc 1/1    1          1          58s
replicaset.apps/opentelemetry-demo-jagger-7785f96b-nnh9   1/1    1          1          60s
replicaset.apps/opentelemetry-demo-kafka-76d49f480          1/1    1          1          57s
replicaset.apps/opentelemetry-demo-logsparser-54d976667   1/1    1          1          57s
replicaset.apps/opentelemetry-demo-leasegenerator-668476f6dc 1/1    1          1          58s
replicaset.apps/opentelemetry-demo-paymentcatalog-8555c59fb 1/1    1          1          57s
replicaset.apps/opentelemetry-demo-productcatalogserver-8c38644f9 1/1    1          1          57s
replicaset.apps/opentelemetry-demo-prometheus-server-57cd8f946 1/1    1          1          59s
replicaset.apps/opentelemetry-demo-quarantine-6f576fd574   1/1    1          1          56s
replicaset.apps/opentelemetry-demo-recommendationservice-6f576fd574 1/1    1          1          56s
replicaset.apps/opentelemetry-demo-shippingservice-fccf765   1/1    1          1          56s
replicaset.apps/opentelemetry-demo-valley-68b4c949b-9fdcv   1/1    1          1          56s

NAME          READY   AGE
statefulset.apps/opentelemetry-demo-search          1/1    59s
[ec2-user@ip-10-0-1-11 kubernetes]$
```

#### 2.3.4.2 Reviewing Pod Logs

If any pods were not running or showed errors, we reviewed the logs for those specific pods using the command `kubectl logs <pod-name>-n otel-demo`. We checked the logs to identify any issues related to service misconfigurations, missing dependencies, or Kubernetes errors. We also checked the running application and took some screenshots.

## Frontend Logs:

```
[ec2-user@ip-10-0-1-11 kubernetes]$ kubectl logs pod/opentelemetry-demo-frontend-598ccdf8fd-4bgr -n opentelemetry
```

```
> frontend:1.0.0 start
  node --require ./Instrumentation.js server.js
```

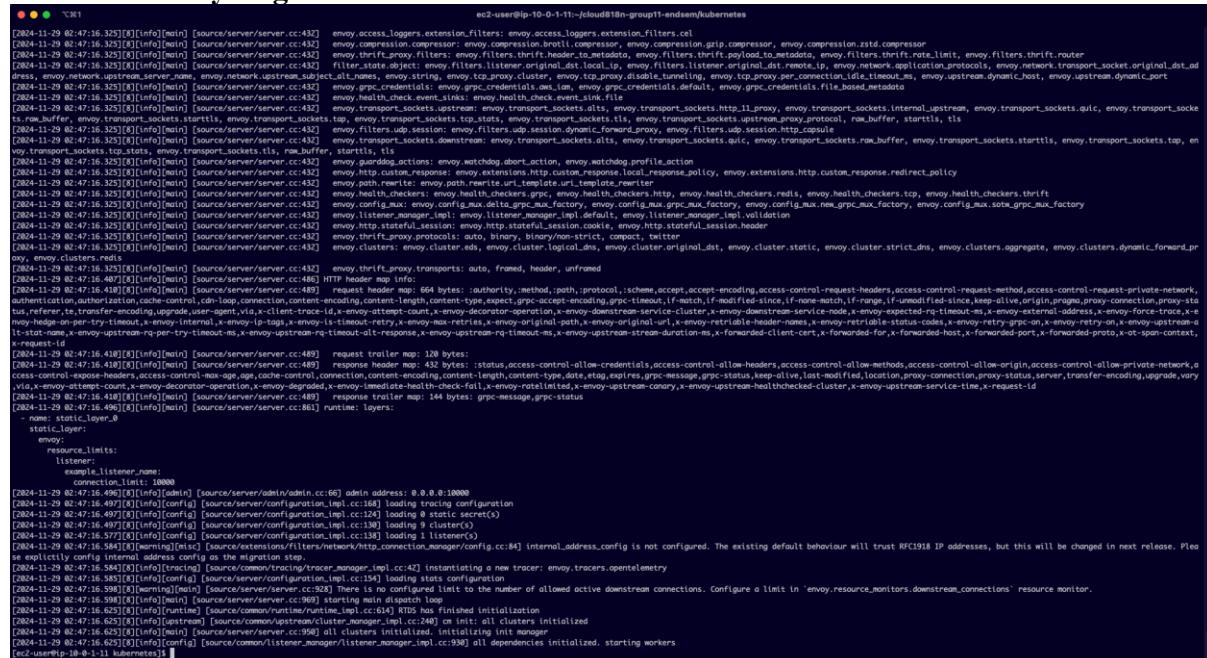
```
  ▲ Next.js 14.2.5
  - Local:   http://opentelemetry-demo-frontend-598ccdf8fd-4bgr:8080
  - Network: http://10.0.1.10:8080
```

```
  ✓ Starting...
  ✓ Ready in 190ms
```

```
[ec2-user@ip-10-0-1-11 kubernetes]$
```



## Frontend Proxy Logs:



## **Jaeger Logs:**

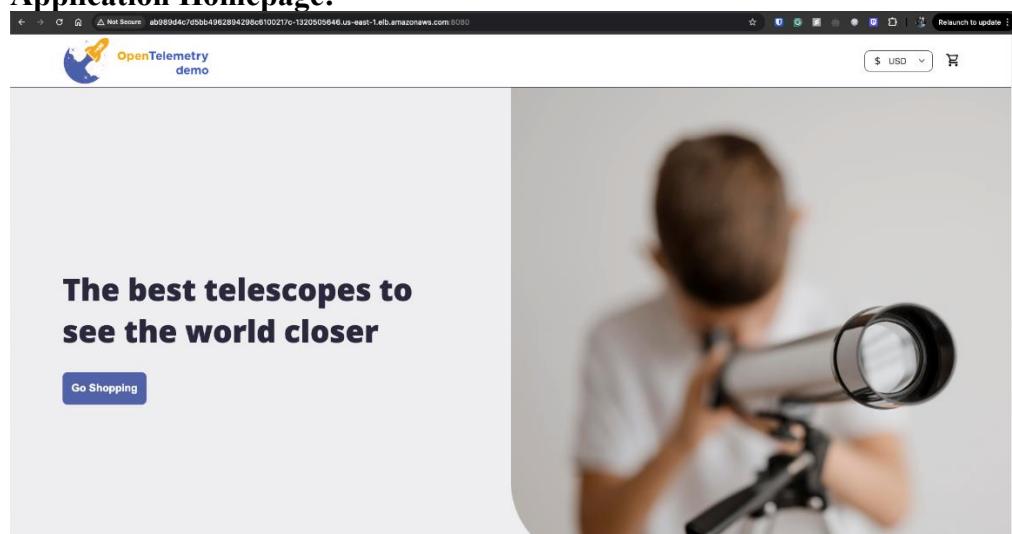
## Prometheus Logs:

## Grafana Logs:

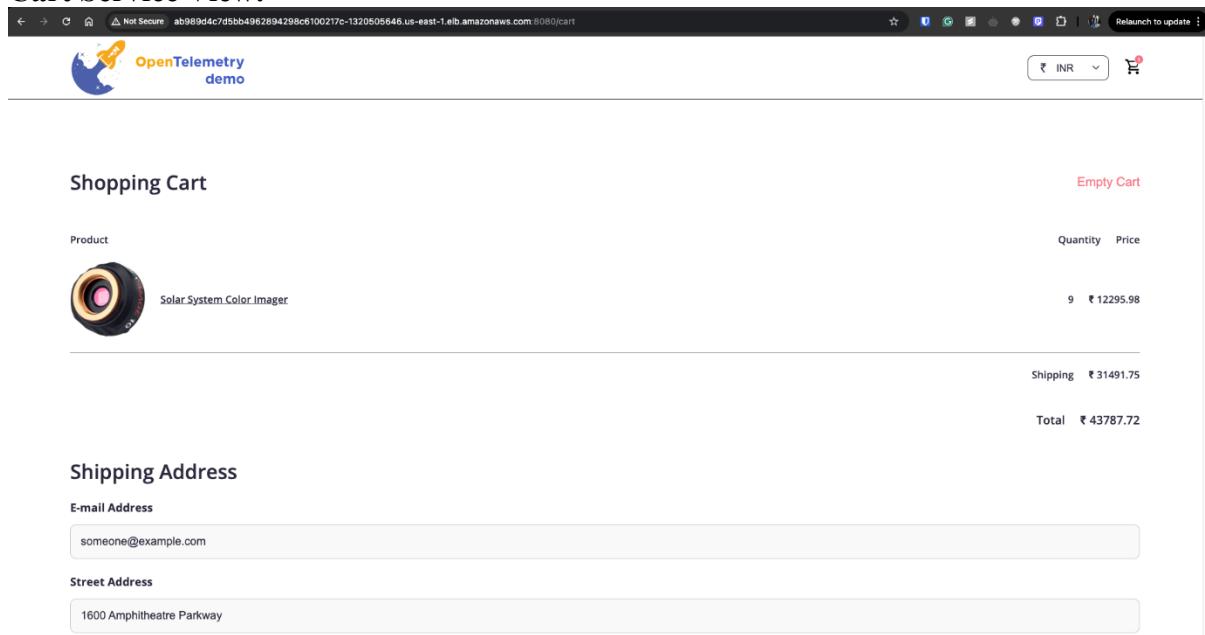
## ProductCatalogService Logs:

```
[ec2-user@ip-10-0-11-kubernetes]$ kubectl logs deployment.apps/open telemetry-demo-productcatalogservice -n otsl-demo
time="2024-11-29T08:47:16Z" level=info msg="Loaded 10 products"
time="2024-11-29T08:47:16Z" level=info msg="ProductCatalogService gRPC server started on port: 8080"
```

### **Application Homepage:**



## Cart Service View:



Not Secure ab989d4c7d5bb4962894298c6100217c-1320505646.us-east-1.elb.amazonaws.com:8080/cart

Relaunch to update

OpenTelemetry demo

INR

Empty Cart

Product

	Quantity	Price
Solar System Color Imager	9	₹ 12295.98

Shipping ₹ 31491.75

Total ₹ 43787.72

Shopping Cart

Shipping Address

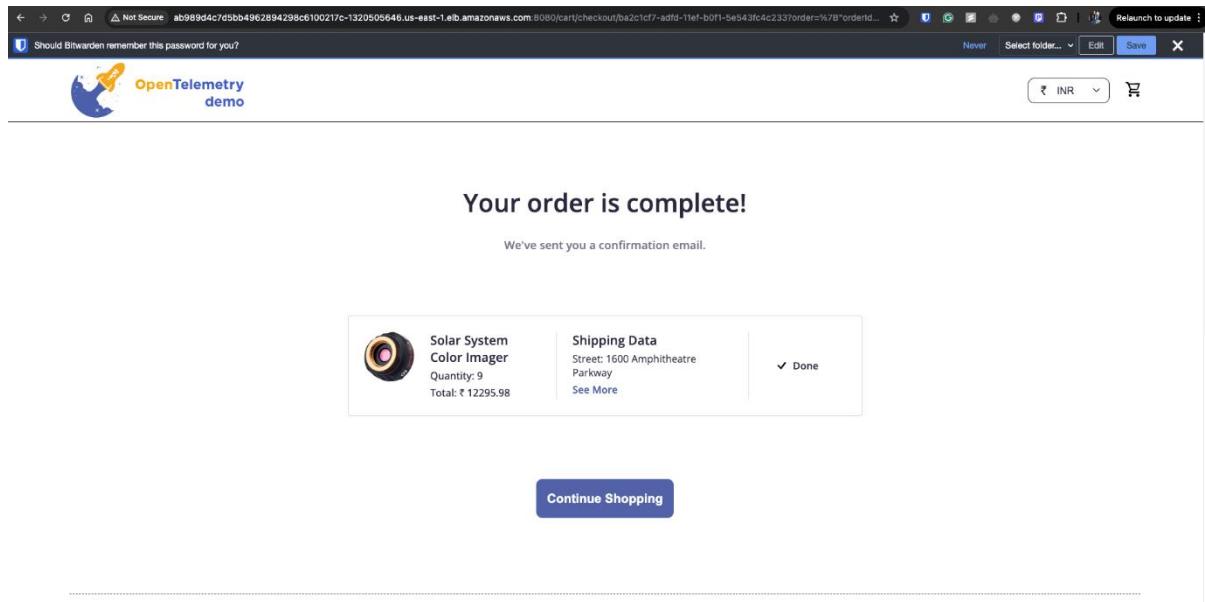
E-mail Address

someone@example.com

Street Address

1600 Amphitheatre Parkway

## Checkout Service:



Not Secure ab989d4c7d5bb4962894298c6100217c-1320505646.us-east-1.elb.amazonaws.com:8080/cart/checkout/ba2c1cf7-adfd-11ef-b0f1-5e543fc4c233?order=%7B"orderId... Never Select folder... Edit Save X

Should Bitwarden remember this password for you?

OpenTelemetry demo

INR

Your order is complete!

We've sent you a confirmation email.

 Solar System Color Imager	Shipping Data
Quantity: 9	Street: 1600 Amphitheatre Parkway
Total: ₹ 12295.98	See More
✓ Done	

Continue Shopping