University of Maryland - College Park

Cloud Computing Final Project

EKS AND MONITORING WITH OPENTELEMETRY

Group 11

Individual Assignment – Jayesh Pamnani

Jayesh Pamnani

Table of Contents

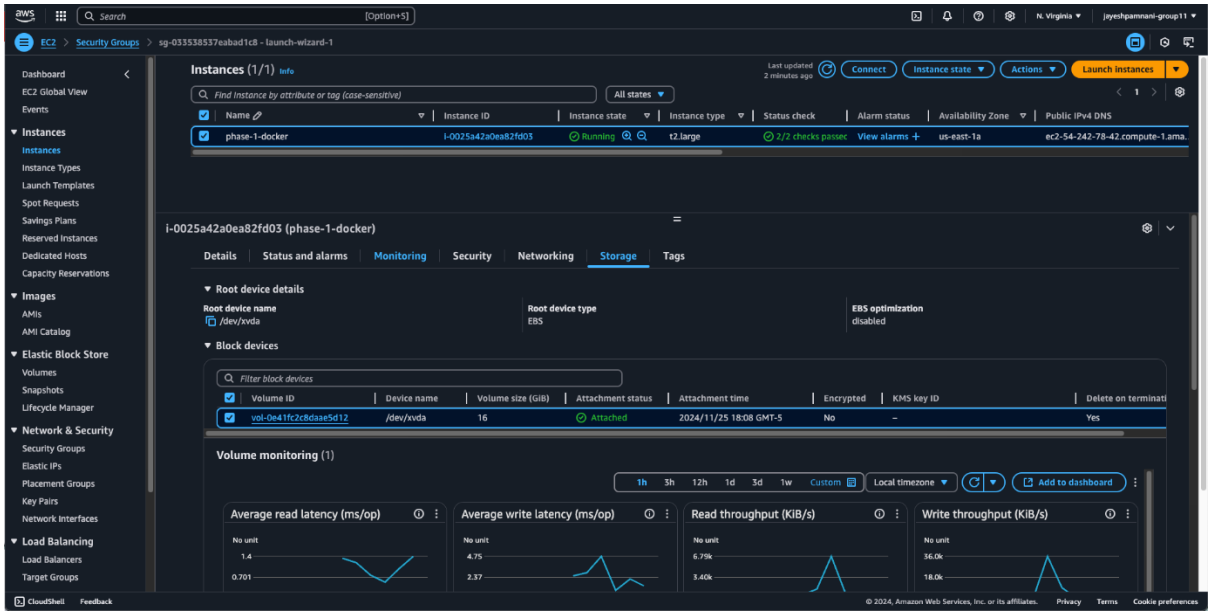
1. Phase 1: Environment and Initial Application Setup

1.1 Docker Deployment

The Docker deployment process was designed to create a scalable environment for hosting and validating the application services.

1.1.1 Instance Details

Type: t2.large
Root Volume: 16GB



1.1.2 Installing Docker and Docker-Compose

Steps Taken:

- Update the instance:
  - sudo yum update -y
- 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

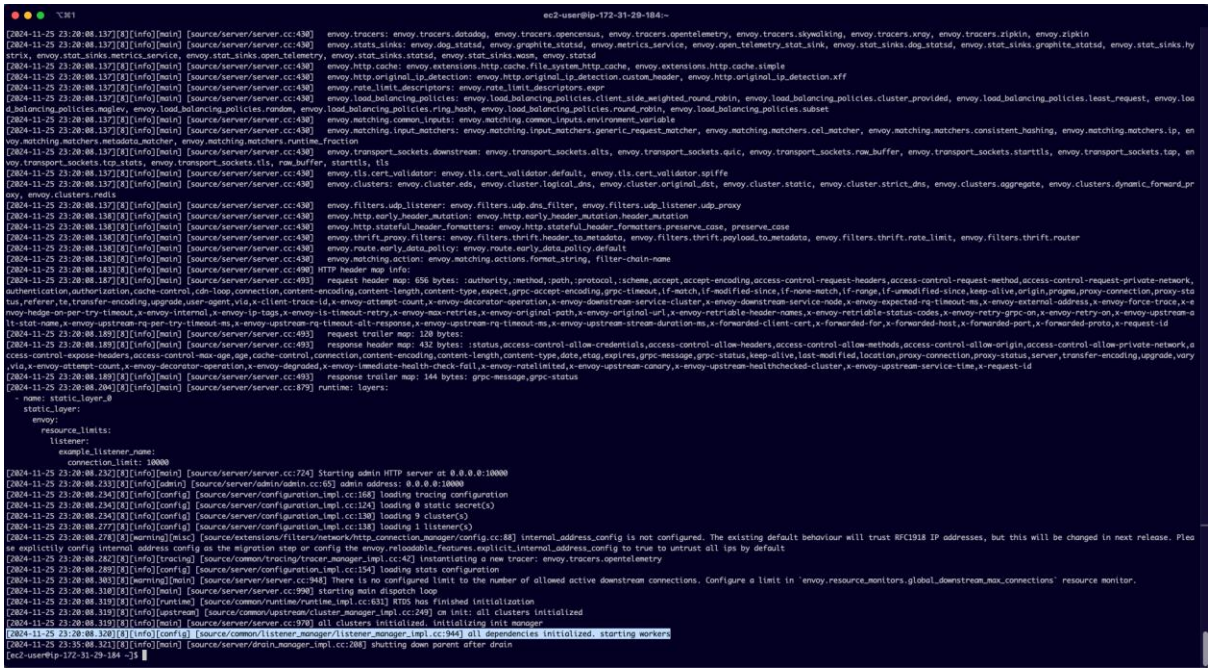
- Download and Install Docker Compose:
    - sudo curl -L "https://github.com/docker/compose/releases/download/v2.20.0/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
    - sudo chmod +x /usr/local/bin/docker-compose
- Verify the installation:
    - docker-compose --version
- Deploy the 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

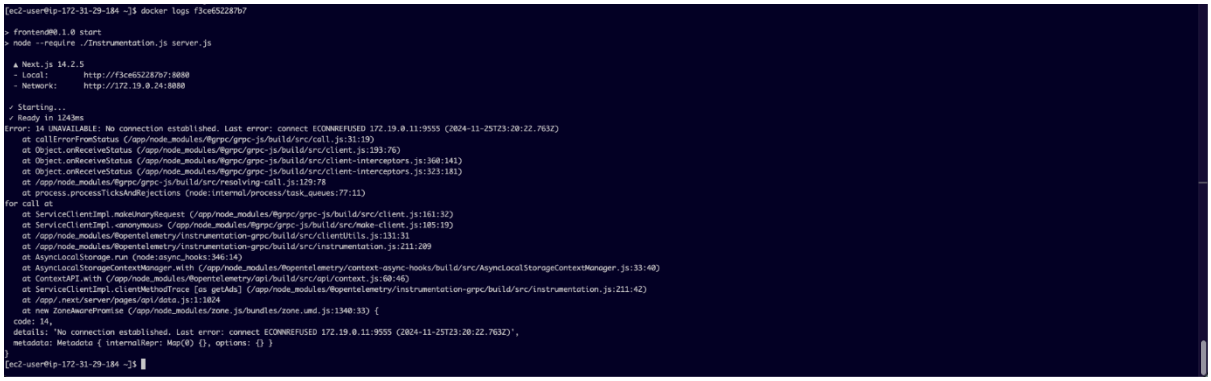
Running `docker ps` to check the running Docker Containers

1.1.4 Container Logs

FrontendProxy Logs

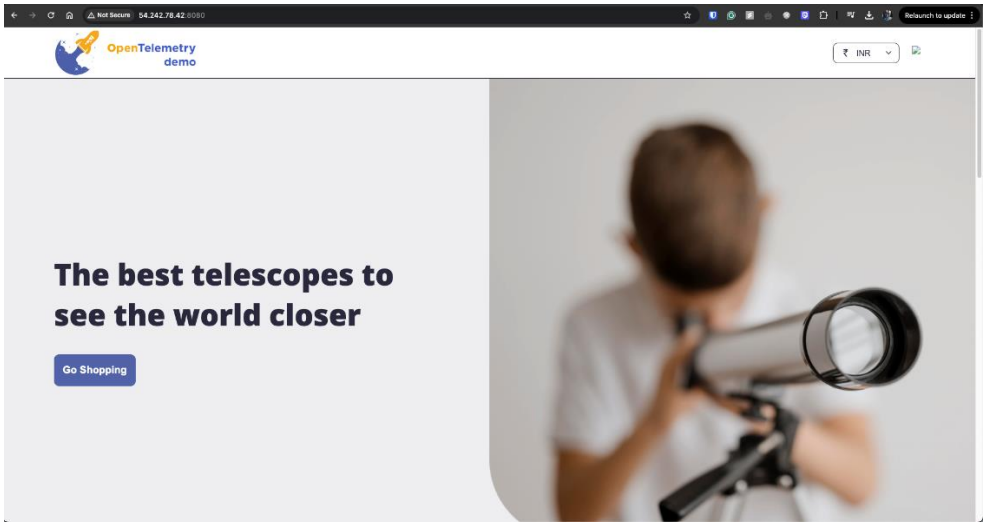


Docker application logs for Frontend

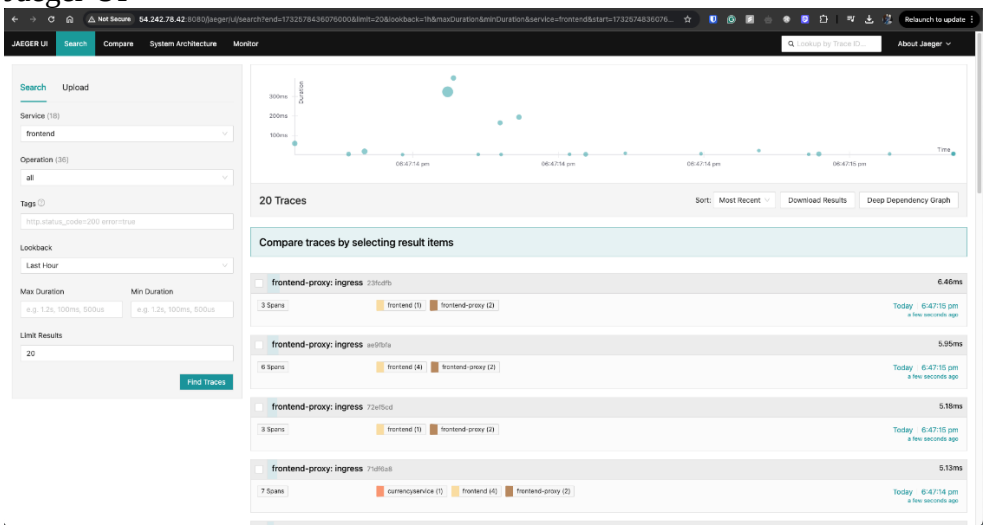


1.1.5 Accessing Application

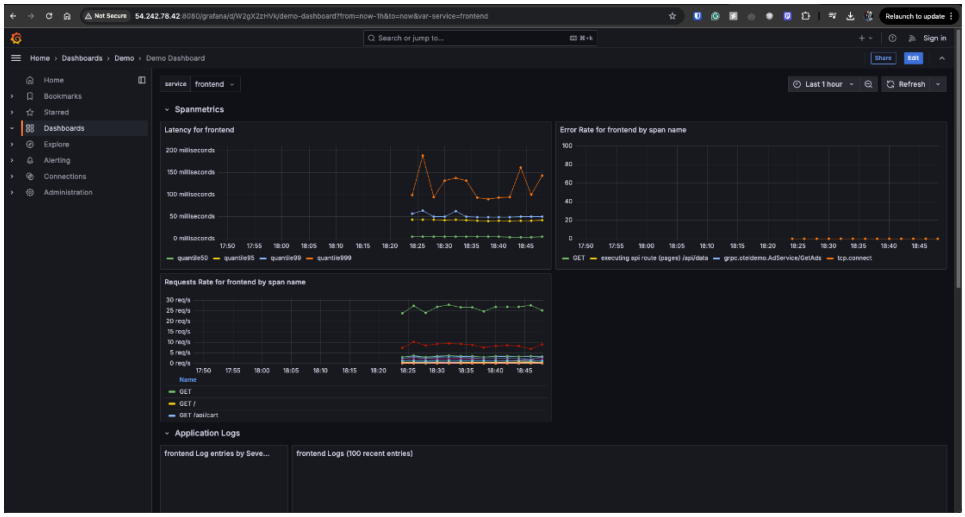
The application was accessed using the Public IP of the EC2 instance at http://<public-ip>:8080



Jaeger UI



Grafana UI



1.2 Kubernetes Setup Tasks

Kubernetes was used to create a robust and scalable cluster for managing the application in a production-like environment.

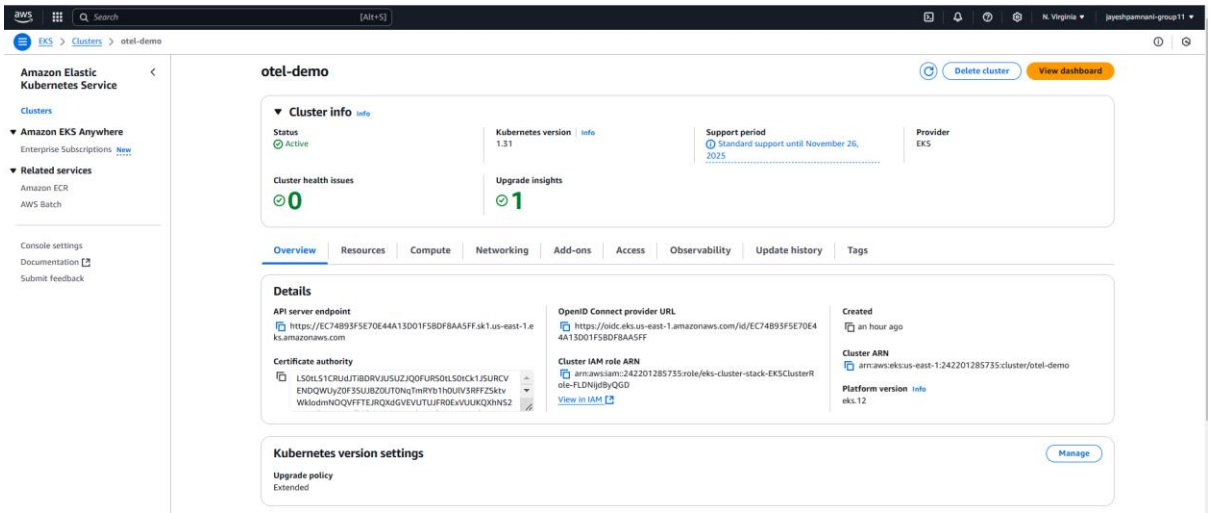
1.2.1 EKS Cluster Configuration Details

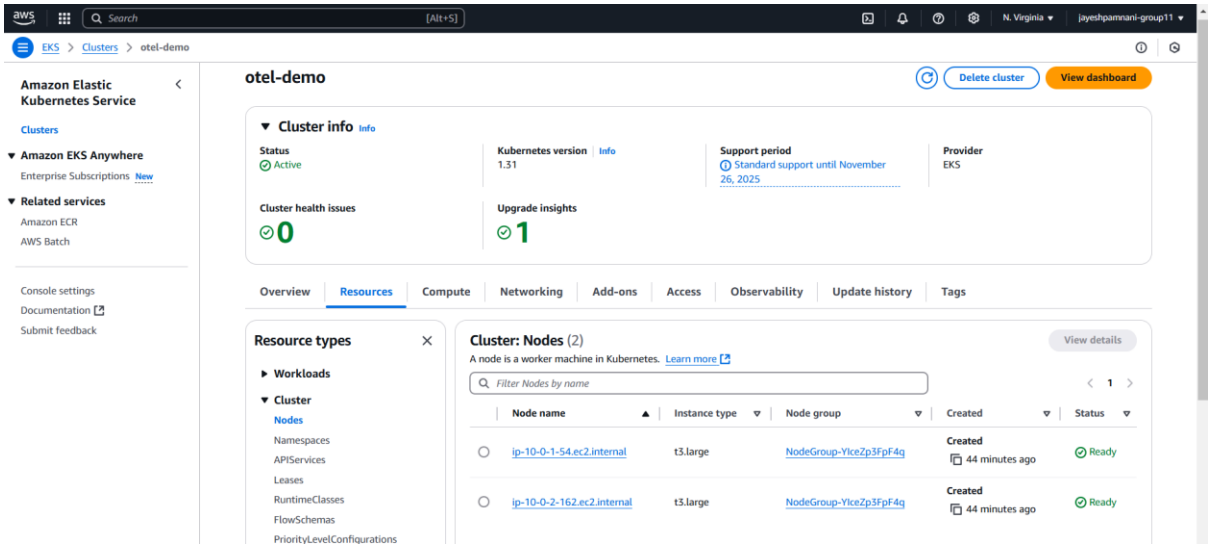
Cluster Details:

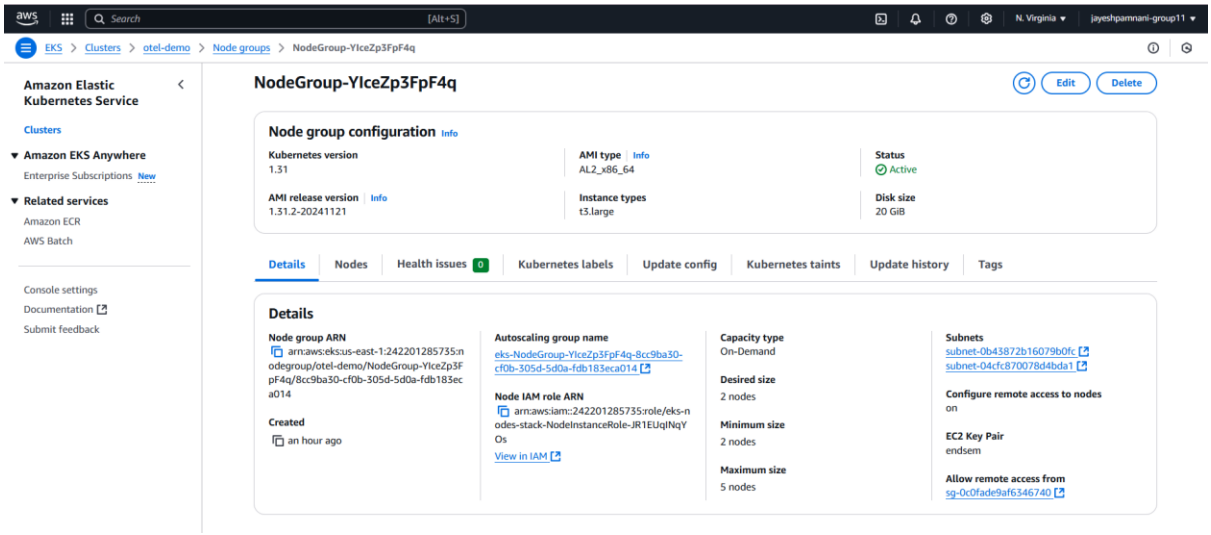
- **Cluster Name**: otel-demo
- **Kubernetes Version**: 1.31
- **Cluster Role ARN**: arn:aws:iam::242201285735:role/eks-cluster-stack-EKSClusterRole-JTREuJqNgY (visible in IAM link)
- **Cluster ARN**: arn:aws:eks:us-east-1:242201285735:cluster/otel-demo
- **Security Group**: otel-demo-cluster-sg

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-JREUqINqYOs
- **Number of Nodes**: 2







1.2.2 EC2 Instance Details Used as the EKS Client

Client Instance Details:

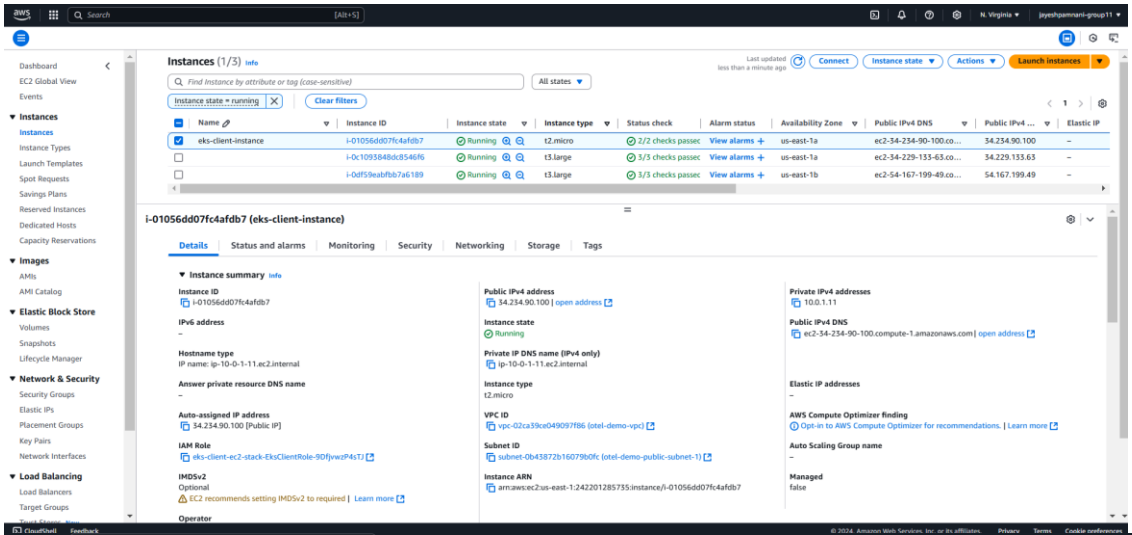
- **Instance Name:** eks-client
- **Instance Type:** t2.micro
- **Key Pair:** Used for SSH Access
- **Storage Volume:** 8 GiB

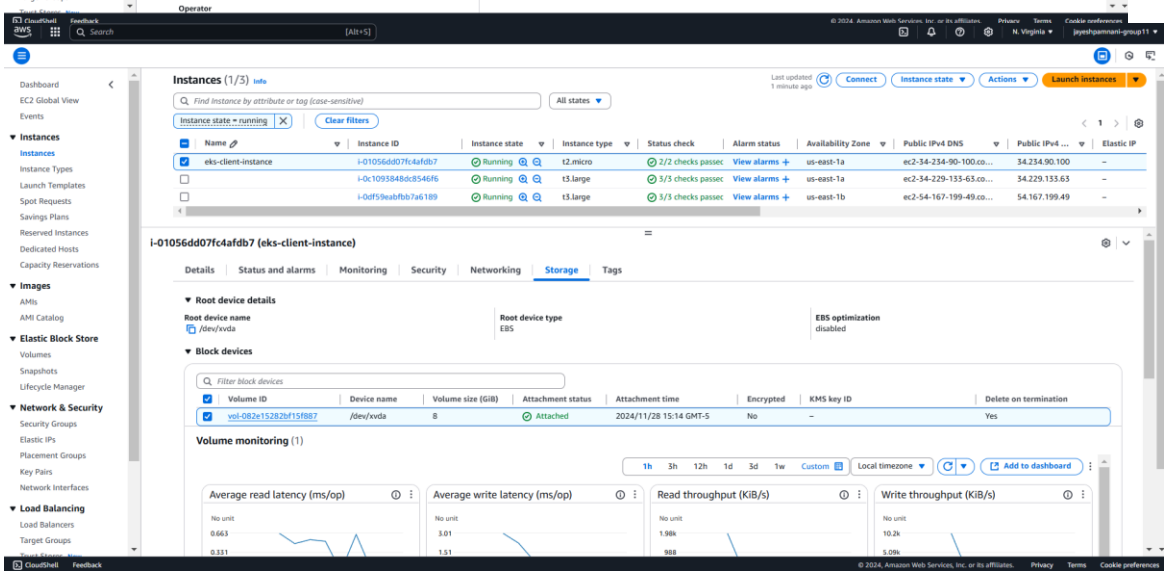
Installed Tools on EKS Client:

1. **AWS CLI:** Configured using the root access key and secret.
2. **kubectl:** Configured using the EKS cluster configuration.
3. **eksctl:** Used for managing the EKS cluster and node groups.

Steps Taken:

- Launched EC2 instance manually with the appropriate security group allowing SSH, HTTPS, and HTTP access.
- Installed tools manually using SSH commands:
    - curl -o kubectl https://s3.amazonaws.com/amazon-eks/1.31.0/2023-08-22/bin/linux/amd64/kubectl
    - chmod +x ./kubectl
    - sudo mv ./kubectl /usr/local/bin
    - curl -Lo eksctl_Linux_amd64.tar.gz https://github.com/weaveworks/eksctl/releases/latest/download/eksctl_Linux_amd64.tar.gz
    - tar -xzf eksctl_Linux_amd64.tar.gz
    - chmod +x eksctl
    - sudo mv eksctl /usr/local/bin
    - rm eksctl_Linux_amd64.tar.gz
- Configured kubectl with:aws eks update-kubeconfig --region us-east-1 --name otel-demo





1.2.3 Status of Pods, Services, Deployment (kubectl get all-n otel-demo)

No modifications was made to the original manifest.

- Cloned the opentelemetry-demo repository:
  - git clone https://github.com/open-telemetry/opentelemetry-demo.git
  - cd opentelemetry-demo/kubernetes
- Applied the deployment manifest:
  - kubectl apply --namespace otel-demo -f opentelemetry-demo.yaml
- Validated the deployment:
  - kubectl get all -n otel-demo

```
[ec2-user@ip-10-0-1-11 ~]$ kubectl get all -n otel-demo
NAME                                                          READY   STATUS    RESTARTS   AGE
pod/opentelemetry-demo-accountingservice-8555cb59fb-cslr8     1/1     Running   0          16m
pod/opentelemetry-demo-adservice-5df4bf56d6-8glvz             1/1     Running   0          16m
pod/opentelemetry-demo-cartservice-694df79ff7-6p7hm           1/1     Running   0          16m
pod/opentelemetry-demo-checkoutservice-744b6cdd6-tvv9w        1/1     Running   0          16m
pod/opentelemetry-demo-currencyservice-6b8fcd9bd6-ptzgj       1/1     Running   0          16m
pod/opentelemetry-demo-emailservice-66d8476fdc-ffqwt          1/1     Running   0          16m
pod/opentelemetry-demo-flagd-679866d767-zt5x6                 2/2     Running   0          16m
pod/opentelemetry-demo-frauddetectionservice-85d48f855f-wv2mh 1/1     Running   0          16m
pod/opentelemetry-demo-frontend-59bccd8fdb-n27wm              1/1     Running   0          16m
pod/opentelemetry-demo-frontendproxy-74f988cfb4-smd2v         1/1     Running   0          16m
pod/opentelemetry-demo-grafana-69b6bd5dd4-g9gqs               1/1     Running   0          16m
pod/opentelemetry-demo-imageprovider-7466d894fb-2bv2j         1/1     Running   0          16m
pod/opentelemetry-demo-jaeger-7785549bb-rwh25                 1/1     Running   0          16m
pod/opentelemetry-demo-kafka-76d4d9f48b-4vwfk                 1/1     Running   0          16m
pod/opentelemetry-demo-loadgenerator-54db796687-l6vjb         1/1     Running   0          16m
pod/opentelemetry-demo-otelcol-5c757cfcf-sfn5f                1/1     Running   0          16m
pod/opentelemetry-demo-paymentservice-857974bcdb-d2fdh        1/1     Running   0          16m
pod/opentelemetry-demo-productcatalogservice-6bc98644f9-7nvqd 1/1     Running   0          16m
pod/opentelemetry-demo-prometheus-server-57cd8f9d46-sl79w     1/1     Running   0          16m
pod/opentelemetry-demo-quoteservice-5658c58fd7-tgjff          1/1     Running   0          16m
pod/opentelemetry-demo-recommendationservice-6f576fd574-r9vxv 1/1     Running   0          16m
pod/opentelemetry-demo-shippingservice-fcfc7765-98h9z         1/1     Running   0          16m
pod/opentelemetry-demo-valkey-68b4cb4498-2vwt9                1/1     Running   0          16m
pod/otel-demo-opensearch-0                                    1/1     Running   0          16m

NAME                                              TYPE           CLUSTER-IP       EXTERNAL-IP
 PORT(S)                                                                   AGE
service/opentelemetry-demo-adservice              ClusterIP      172.20.153.217   <none>
 8080/TCP                                                                  16m
service/opentelemetry-demo-cartservice            ClusterIP      172.20.239.47    <none>
 8080/TCP                                                                  16m
service/opentelemetry-demo-checkoutservice        ClusterIP      172.20.155.19    <none>
 8080/TCP                                                                  16m
service/opentelemetry-demo-currencyservice        ClusterIP      172.20.13.91     <none>
 8080/TCP                                                                  16m
service/opentelemetry-demo-emailservice           ClusterIP      172.20.34.39     <none>
 8080/TCP                                                                  16m
service/opentelemetry-demo-flagd                  ClusterIP      172.20.203.23    <none>
 8013/TCP,4000/TCP                                                         16m
service/opentelemetry-demo-frontend               ClusterIP      172.20.65.238    <none>
 8080/TCP                                                                  16m
service/opentelemetry-demo-frontendproxy          LoadBalancer   172.20.221.99    a24bc6da0afcf4ad89eef764083a51d7-1448227332.us-east-1.elb.amazonaws.com   8080:30994/TCP
                                              16m
service/opentelemetry-demo-grafana                ClusterIP      172.20.100.211   <none>
 80/TCP                                                                    16m
service/opentelemetry-demo-imageprovider          ClusterIP      172.20.53.111    <none>
 8081/TCP                                                                  16m
```

```
service/opentelemetry-demo-imageprovider          ClusterIP      172.20.53.111    <none>
 8081/TCP                                                                  16m
service/opentelemetry-demo-jaeger-agent           ClusterIP      None             <none>
 5775/UDP,5778/TCP,6831/UDP,6832/UDP                                       16m
service/opentelemetry-demo-jaeger-collector       ClusterIP      None             <none>
 9411/TCP,14250/TCP,14267/TCP,14268/TCP,4317/TCP,4318/TCP                  16m
service/opentelemetry-demo-jaeger-query           ClusterIP      None             <none>
 16686/TCP,16685/TCP                                                       16m
service/opentelemetry-demo-kafka                  ClusterIP      172.20.163.133   <none>
 9092/TCP,9093/TCP                                                         16m
service/opentelemetry-demo-loadgenerator          ClusterIP      172.20.182.82    <none>
 8089/TCP                                                                  16m
service/opentelemetry-demo-otelcol                ClusterIP      172.20.179.54    <none>
 6831/UDP,14250/TCP,14268/TCP,8888/TCP,4317/TCP,4318/TCP,9464/TCP,9411/TCP   16m
service/opentelemetry-demo-paymentservice         ClusterIP      172.20.64.251    <none>
 8080/TCP                                                                  16m
service/opentelemetry-demo-productcatalogservice  ClusterIP      172.20.119.137   <none>
 8080/TCP                                                                  16m
service/opentelemetry-demo-prometheus-server      ClusterIP      172.20.245.180   <none>
 9090/TCP                                                                  16m
service/opentelemetry-demo-quoteservice           ClusterIP      172.20.224.18    <none>
 8080/TCP                                                                  16m
service/opentelemetry-demo-recommendationservice  ClusterIP      172.20.203.219   <none>
 8080/TCP                                                                  16m
service/opentelemetry-demo-shippingservice        ClusterIP      172.20.70.137    <none>
 8080/TCP                                                                  16m
service/opentelemetry-demo-valkey                 ClusterIP      172.20.70.77     <none>
 6379/TCP                                                                  16m
service/otel-demo-opensearch                      ClusterIP      172.20.72.185    <none>
 9200/TCP,9300/TCP,9600/TCP                                                16m
service/otel-demo-opensearch-headless             ClusterIP      None             <none>
 9200/TCP,9300/TCP,9600/TCP                                                16m

NAME                                                      READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/opentelemetry-demo-accountingservice      1/1     1            1           16m
deployment.apps/opentelemetry-demo-adservice              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-flagd                  1/1     1            1           16m
deployment.apps/opentelemetry-demo-frauddetectionservice  1/1     1            1           16m
deployment.apps/opentelemetry-demo-frontend               1/1     1            1           16m
deployment.apps/opentelemetry-demo-frontendproxy          1/1     1            1           16m
deployment.apps/opentelemetry-demo-grafana                1/1     1            1           16m
deployment.apps/opentelemetry-demo-imageprovider          1/1     1            1           16m
deployment.apps/opentelemetry-demo-jaeger                 1/1     1            1           16m
deployment.apps/opentelemetry-demo-kafka                  1/1     1            1           16m
deployment.apps/opentelemetry-demo-loadgenerator          1/1     1            1           16m
```

```
deployment.apps/opentelemetry-demo-otelcol                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-quoteservice           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

NAME                                                      DESIRED   CURRENT   READY   AGE
replicaset.apps/opentelemetry-demo-accountingservice-8555cb59fb       1         1       1       16m
replicaset.apps/opentelemetry-demo-adservice-5df4bf56d6               1         1       1       16m
replicaset.apps/opentelemetry-demo-cartservice-694df79ff7             1         1       1       16m
replicaset.apps/opentelemetry-demo-checkoutservice-744b6cdd6          1         1       1       16m
replicaset.apps/opentelemetry-demo-currencyservice-6b8fcd9bd6         1         1       1       16m
replicaset.apps/opentelemetry-demo-emailservice-66d8476fdc            1         1       1       16m
replicaset.apps/opentelemetry-demo-flagd-679866d767                   1         1       1       16m
replicaset.apps/opentelemetry-demo-frauddetectionservice-85d48f855f   1         1       1       16m
replicaset.apps/opentelemetry-demo-frontend-59bccd8fdb                1         1       1       16m
replicaset.apps/opentelemetry-demo-frontendproxy-74f988cfb4           1         1       1       16m
replicaset.apps/opentelemetry-demo-grafana-69b6bd5dd4                 1         1       1       16m
replicaset.apps/opentelemetry-demo-imageprovider-7466d894fb           1         1       1       16m
replicaset.apps/opentelemetry-demo-jaeger-7785549bb                   1         1       1       16m
replicaset.apps/opentelemetry-demo-kafka-76d4d9f48b                   1         1       1       16m
replicaset.apps/opentelemetry-demo-loadgenerator-54db796687           1         1       1       16m
replicaset.apps/opentelemetry-demo-otelcol-5c757cfcf                  1         1       1       16m
replicaset.apps/opentelemetry-demo-paymentservice-857974bcdb          1         1       1       16m
replicaset.apps/opentelemetry-demo-productcatalogservice-6bc98644f9   1         1       1       16m
replicaset.apps/opentelemetry-demo-prometheus-server-57cd8f9d46       1         1       1       16m
replicaset.apps/opentelemetry-demo-quoteservice-5658c58fd7            1         1       1       16m
replicaset.apps/opentelemetry-demo-recommendationservice-6f576fd574   1         1       1       16m
replicaset.apps/opentelemetry-demo-shippingservice-fcfc7765           1         1       1       16m
replicaset.apps/opentelemetry-demo-valkey-68b4cb4498                  1         1       1       16m

NAME                                     READY   AGE
statefulset.apps/otel-demo-opensearch    1/1     16m
```

1.2.4 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
```

Payment service

```
[ec2-user@ip-10-0-1-11 ~]$ kubectl logs pod/opentelemetry-demo-paymentservice-857974bcdb-d2fdh -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":1732825387034,"pid":17,"hostname":"opentelemetry-demo-paymentservice-857974bcdb-d2fdh","msg":"PaymentService gRPC server started on port 8080"}
{"level":30,"time":1732825463131,"pid":17,"hostname":"opentelemetry-demo-paymentservice-857974bcdb-d2fdh","trace_id":"d108b341b28aef16bdef34abf62c8adf","span_id":"e054971
490f538cf","trace_flags":"01","request":{"amount":{"currencyCode":"USD","units":{"low":5984,"high":0,"unsigned":false},"nanos":789999983},"creditCard":{"creditCardNumber"
:"4485-4803-8707-3547","creditCardCvv":682,"creditCardExpirationYear":2039,"creditCardExpirationMonth":9}},"msg":"Charge request received."}
{"level":30,"time":1732825463156,"pid":17,"hostname":"opentelemetry-demo-paymentservice-857974bcdb-d2fdh","trace_id":"30517ce2a9a295d98e29690bfafe3a51","span_id":"fdf3f72
59ca81136","trace_flags":"01","request":{"amount":{"currencyCode":"USD","units":{"low":1830,"high":0,"unsigned":false},"nanos":99999993},"creditCard":{"creditCardNumber":
"4916-0816-6217-7968","creditCardCvv":397,"creditCardExpirationYear":2039,"creditCardExpirationMonth":5}},"msg":"Charge request received."}
{"level":30,"time":1732825463165,"pid":17,"hostname":"opentelemetry-demo-paymentservice-857974bcdb-d2fdh","trace_id":"30517ce2a9a295d98e29690bfafe3a51","span_id":"fdf3f72
59ca81136","trace_flags":"01","transactionId":"4359aac7-1dd5-498c-842b-442ccf73f077","cardType":"visa","lastFourDigits":"7968","amount":{"units":{"low":1830,"high":0,"uns
igned":false},"nanos":99999993,"currencyCode":"USD"},"msg":"Transaction complete."}
{"level":30,"time":1732825463237,"pid":17,"hostname":"opentelemetry-demo-paymentservice-857974bcdb-d2fdh","trace_id":"d108b341b28aef16bdef34abf62c8adf","span_id":"e054971
490f538cf","trace_flags":"01","transactionId":"8a2f03df-691c-43d3-8dee-0a2af9f7b6aa","cardType":"visa","lastFourDigits":"3547","amount":{"units":{"low":5984,"high":0,"uns
igned":false},"nanos":789999983,"currencyCode":"USD"},"msg":"Transaction complete."}
{"level":30,"time":1732825466116,"pid":17,"hostname":"opentelemetry-demo-paymentservice-857974bcdb-d2fdh","trace_id":"0463430a5e314a1512f9bc0fccce4ba6","span_id":"97beb12
92fbc34af","trace_flags":"01","request":{"amount":{"currencyCode":"USD","units":{"low":512,"high":0,"unsigned":false},"nanos":799999996},"creditCard":{"creditCardNumber":
"4929-5431-0337-5647","creditCardCvv":793,"creditCardExpirationYear":2039,"creditCardExpirationMonth":6}},"msg":"Charge request received."}
{"level":30,"time":1732825466118,"pid":17,"hostname":"opentelemetry-demo-paymentservice-857974bcdb-d2fdh","trace_id":"0463430a5e314a1512f9bc0fccce4ba6","span_id":"97beb12
92fbc34af","trace_flags":"01","transactionId":"7cf999a4-9aee-4e53-9105-2de3c9d7c860","cardType":"visa","lastFourDigits":"5647","amount":{"units":{"low":512,"high":0,"unsi
gned":false},"nanos":799999996,"currencyCode":"USD"},"msg":"Transaction complete."}
{"level":30,"time":1732825487845,"pid":17,"hostname":"opentelemetry-demo-paymentservice-857974bcdb-d2fdh","trace_id":"f535ef1b95924c8a36f545c315655ed6","span_id":"a5d8606
9efc03916","trace_flags":"01","request":{"amount":{"currencyCode":"USD","units":{"low":292,"high":0,"unsigned":false},"nanos":909999999},"creditCard":{"creditCardNumber":
"4532-4211-7434-1278","creditCardCvv":118,"creditCardExpirationYear":2039,"creditCardExpirationMonth":2}},"msg":"Charge request received."}
{"level":30,"time":1732825487846,"pid":17,"hostname":"opentelemetry-demo-paymentservice-857974bcdb-d2fdh","trace_id":"f535ef1b95924c8a36f545c315655ed6","span_id":"a5d8606
9efc03916","trace_flags":"01","transactionId":"0eb3c096-f702-4a58-bf91-4b7d08e9cf1c","cardType":"visa","lastFourDigits":"1278","amount":{"units":{"low":292,"high":0,"unsi
gned":false},"nanos":909999999,"currencyCode":"USD"},"msg":"Transaction complete."}
{"level":30,"time":1732825496502,"pid":17,"hostname":"opentelemetry-demo-paymentservice-857974bcdb-d2fdh","trace_id":"7ce94583f0acc4d24b828d2834200602","span_id":"cbd9001
3be0bb6b1","trace_flags":"01","request":{"amount":{"currencyCode":"USD","units":{"low":1499,"high":0,"unsigned":false},"nanos":799999997},"creditCard":{"creditCardNumber"
```

```
7947fdc6d","trace_flags":"01","transactionId":"65d3cf8f-3155-4c00-a9de-247d1f27c54e","cardType":"visa","lastFourDigits":"3657","amount":{"units":{"low":644,"high":0,"unsi
gned":false},"nanos":849999997,"currencyCode":"USD"},"msg":"Transaction complete."}
{"level":30,"time":1732827860804,"pid":17,"hostname":"opentelemetry-demo-paymentservice-857974bcdb-d2fdh","trace_id":"f79f7db6caf8644400980f2b5379ac66","span_id":"7644999
1d55df24e","trace_flags":"01","request":{"amount":{"currencyCode":"USD","units":{"low":3233,"high":0,"unsigned":false},"nanos":599999980},"creditCard":{"creditCardNumber"
:"4916-0816-6217-7968","creditCardCvv":397,"creditCardExpirationYear":2039,"creditCardExpirationMonth":5}},"msg":"Charge request received."}
{"level":30,"time":1732827860804,"pid":17,"hostname":"opentelemetry-demo-paymentservice-857974bcdb-d2fdh","trace_id":"f79f7db6caf8644400980f2b5379ac66","span_id":"7644999
1d55df24e","trace_flags":"01","transactionId":"c3562f3d-a330-4aa6-8384-edc30e1a2231","cardType":"visa","lastFourDigits":"7968","amount":{"units":{"low":3233,"high":0,"uns
igned":false},"nanos":599999980,"currencyCode":"USD"},"msg":"Transaction complete."}
{"level":30,"time":1732827869399,"pid":17,"hostname":"opentelemetry-demo-paymentservice-857974bcdb-d2fdh","trace_id":"190279781da098aa63666e21d18e01b","span_id":"279e08a
250f6f4f7","trace_flags":"01","request":{"amount":{"currencyCode":"USD","units":{"low":501,"high":0,"unsigned":false},"nanos":999999996},"creditCard":{"creditCardNumber":
"4485-4803-8707-3547","creditCardCvv":682,"creditCardExpirationYear":2039,"creditCardExpirationMonth":9}},"msg":"Charge request received."}
{"level":30,"time":1732827869402,"pid":17,"hostname":"opentelemetry-demo-paymentservice-857974bcdb-d2fdh","trace_id":"1902797081da098aa63666e21d18e01b","span_id":"279e08a
250f6f4f7","trace_flags":"01","transactionId":"ec863886-e7a6-4909-8d1f-393961f8d962","cardType":"visa","lastFourDigits":"3547","amount":{"units":{"low":501,"high":0,"unsi
gned":false},"nanos":999999996,"currencyCode":"USD"},"msg":"Transaction complete."}
{"level":30,"time":1732827872870,"pid":17,"hostname":"opentelemetry-demo-paymentservice-857974bcdb-d2fdh","trace_id":"83ab37ebbf85dccc60a41fc4e3d93ba0","span_id":"7a9af3c
34abee83f","trace_flags":"01","request":{"amount":{"currencyCode":"USD","units":{"low":119,"high":0,"unsigned":false},"nanos":399999999},"creditCard":{"creditCardNumber":
"4539-1103-5661-7083","creditCardCvv":784,"creditCardExpirationYear":2039,"creditCardExpirationMonth":4}},"msg":"Charge request received."}
{"level":30,"time":1732827872870,"pid":17,"hostname":"opentelemetry-demo-paymentservice-857974bcdb-d2fdh","trace_id":"83ab37ebbf85dccc60a41fc4e3d93ba0","span_id":"7a9af3c
34abee83f","trace_flags":"01","transactionId":"a583aef4-2e32-40dc-85df-70c85fd34bfe","cardType":"visa","lastFourDigits":"7083","amount":{"units":{"low":119,"high":0,"unsi
gned":false},"nanos":399999999,"currencyCode":"USD"},"msg":"Transaction complete."}
{"level":30,"time":1732827884150,"pid":17,"hostname":"opentelemetry-demo-paymentservice-857974bcdb-d2fdh","trace_id":"784c5dc6f574595bb59ee957bab705c3","span_id":"b10bfd0
08d34f315","trace_flags":"01","request":{"amount":{"currencyCode":"USD","units":{"low":4157,"high":0,"unsigned":false},"nanos":599999999},"creditCard":{"creditCardNumber":
"4432-8015-6152-0454","creditCardCvv":672,"creditCardExpirationYear":2039,"creditCardExpirationMonth":1}},"msg":"Charge request received."}
{"level":30,"time":1732827884151,"pid":17,"hostname":"opentelemetry-demo-paymentservice-857974bcdb-d2fdh","trace_id":"784c5dc6f574595bb59ee957bab705c3","span_id":"b10bfd0
08d34f315","trace_flags":"01","transactionId":"c709a3b4-3f44-4637-8e99-43b5dd62ffa8","cardType":"visa","lastFourDigits":"0454","amount":{"units":{"low":4157,"high":0,"uns
igned":false},"nanos":59999999,"currencyCode":"USD"},"msg":"Transaction complete."}
{"level":30,"time":1732827892856,"pid":17,"hostname":"opentelemetry-demo-paymentservice-857974bcdb-d2fdh","trace_id":"baba40089420e03c4ed502cde73f9f7c","span_id":"4904fa3
cd2df16ff","trace_flags":"01","request":{"amount":{"currencyCode":"USD","units":{"low":1862,"high":0,"unsigned":false},"nanos":849999989},"creditCard":{"creditCardNumber"
:"4929-5431-0337-5647","creditCardCvv":793,"creditCardExpirationYear":2039,"creditCardExpirationMonth":6}},"msg":"Charge request received."}
{"level":30,"time":1732827892856,"pid":17,"hostname":"opentelemetry-demo-paymentservice-857974bcdb-d2fdh","trace_id":"baba40089420e03c4ed502cde73f9f7c","span_id":"4904fa3
cd2df16ff","trace_flags":"01","transactionId":"8c8e6dd7-7d21-4bb6-b7d9-98f874f9398c","cardType":"visa","lastFourDigits":"5647","amount":{"units":{"low":1862,"high":0,"uns
igned":false},"nanos":849999989,"currencyCode":"USD"},"msg":"Transaction complete."}
{"level":30,"time":1732827907030,"pid":17,"hostname":"opentelemetry-demo-paymentservice-857974bcdb-d2fdh","trace_id":"e25d1e47cd9e57916705e9869b7bf521","span_id":"0809c76
d205975a1","trace_flags":"01","request":{"amount":{"currencyCode":"USD","units":{"low":26233,"high":0,"unsigned":false},"nanos":339999996},"creditCard":{"creditCardNumber
":"4539-1103-5661-7083","creditCardCvv":784,"creditCardExpirationYear":2039,"creditCardExpirationMonth":4}},"msg":"Charge request received."}
{"level":30,"time":1732827907033,"pid":17,"hostname":"opentelemetry-demo-paymentservice-857974bcdb-d2fdh","trace_id":"e25d1e47cd9e57916705e9869b7bf521","span_id":"0809c76
d205975a1","trace_flags":"01","transactionId":"40ff21d4-0d84-4c71-8e05-e71cfd4cbac8","cardType":"visa","lastFourDigits":"7083","amount":{"units":{"low":26233,"high":0,"un
signed":false},"nanos":339999996,"currencyCode":"USD"},"msg":"Transaction complete."}
{"level":30,"time":1732827926995,"pid":17,"hostname":"opentelemetry-demo-paymentservice-857974bcdb-d2fdh","trace_id":"e70cd4269346e3315db33f53b34120e7","span_id":"a3952d0
4207bc953","trace_flags":"01","request":{"amount":{"currencyCode":"USD","units":{"low":5714,"high":0,"unsigned":false},"nanos":749999998},"creditCard":{"creditCardNumber
":"4485-4803-8707-3547","creditCardCvv":682,"creditCardExpirationYear":2039,"creditCardExpirationMonth":9}},"msg":"Charge request received."}
{"level":30,"time":1732827926995,"pid":17,"hostname":"opentelemetry-demo-paymentservice-857974bcdb-d2fdh","trace_id":"e70cd4269346e3315db33f53b34120e7","span_id":"a3952d0
4207bc953","trace_flags":"01","transactionId":"5ec4c469-6f9d-43d0-b244-7f24755550210","cardType":"visa","lastFourDigits":"3547","amount":{"units":{"low":5714,"high":0,"uns
igned":false},"nanos":749999998,"currencyCode":"USD"},"msg":"Transaction complete."}
```


Grafana


```
[ec2-user@ip-10-0-1-11 ~]$ kubectl logs pod/opentelemetry-demo-grafana-69b6bd5dd4-g9gqs -n otel-demo
✔Downloaded and extracted grafana-opensearch-datasource v2.22.0 zip successfully to /var/lib/grafana/plugins/grafana-opensearch-datasource

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

logger=settings t=2024-11-28T20:23:08.28713371Z level=info msg="Starting Grafana" version=11.3.0 commit=d9455ff7db73b694db7d412e49a68bec767f2b5a branch=HEAD compiled=202
4-11-28T20:23:08Z
logger=settings t=2024-11-28T20:23:08.28762727Z level=info msg="Config loaded from" file=/usr/share/grafana/conf/defaults.ini
logger=settings t=2024-11-28T20:23:08.28764312Z level=info msg="Config loaded from" file=/etc/grafana/grafana.ini
logger=settings t=2024-11-28T20:23:08.28765021Z level=info msg="Config overridden from command line" arg="default.paths.data=/var/lib/grafana/"
logger=settings t=2024-11-28T20:23:08.28765681Z level=info msg="Config overridden from command line" arg="default.paths.logs=/var/log/grafana"
logger=settings t=2024-11-28T20:23:08.28766294Z level=info msg="Config overridden from command line" arg="default.paths.plugins=/var/lib/grafana/plugins"
logger=settings t=2024-11-28T20:23:08.28766939Z level=info msg="Config overridden from command line" arg="default.paths.provisioning=/etc/grafana/provisioning"
logger=settings t=2024-11-28T20:23:08.28767606Z level=info msg="Config overridden from command line" arg="default.log.mode=console"
logger=settings t=2024-11-28T20:23:08.28768352Z level=info msg="Config overridden from Environment variable" var="GF_PATHS_DATA=/var/lib/grafana/"
logger=settings t=2024-11-28T20:23:08.28768979Z 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=2024-11-28T20:23:08.28770188Z level=info msg="Config overridden from Environment variable" var="GF_PATHS_PROVISIONING=/etc/grafana/provisioning"
logger=settings t=2024-11-28T20:23:08.28770805Z level=info msg="Config overridden from Environment variable" var="GF_SECURITY_ADMIN_USER=admin"
logger=settings t=2024-11-28T20:23:08.28771423Z level=info msg="Config overridden from Environment variable" var="GF_SECURITY_ADMIN_PASSWORD=*********"
logger=settings t=2024-11-28T20:23:08.28772082Z level=info msg=Target target=[all]
logger=settings t=2024-11-28T20:23:08.28773358Z level=info msg="Path Home" path=/usr/share/grafana
logger=settings t=2024-11-28T20:23:08.28773910Z level=info msg="Path Data" path=/var/lib/grafana/
logger=settings t=2024-11-28T20:23:08.28774449Z level=info msg="Path Logs" path=/var/log/grafana
logger=settings t=2024-11-28T20:23:08.28775008Z level=info msg="Path Plugins" path=/var/lib/grafana/plugins
logger=settings t=2024-11-28T20:23:08.28775587Z level=info msg="Path Provisioning" path=/etc/grafana/provisioning
logger=settings t=2024-11-28T20:23:08.28776149Z level=info msg="App mode production"
logger=featuremgmt t=2024-11-28T20:23:08.28835307Z level=info msg=FeatureToggles cloudWatchCrossAccountQuerying=true addFieldFromCalculationStatFunctions=true cloudWatch
NewLabelParsing=true dashboardSceneForViewers=true transformationsVariableSupport=true transformationsRedesign=true dashboardSceneSolo=true ssoSettingsApi=true accessCont
rolOnCall=true managedPluginsInstall=true groupToNestedTableTransformation=true promQLScope=true lokiQuerySplitting=true prometheusMetricEncyclopedia=true formatString=tr
ue logsInfiniteScrolling=true correlations=true alertingInsights=true autoMigrateXYChartPanel=true prometheusConfigOverhaulAuth=true publicDashboards=true dataplaneFronte
ndFallback=true alertingSimplifiedRouting=true dashboardScene=true alertingNoDataErrorExecution=true lokiMetricDataplane=true publicDashboardsScene=true dashgpt=true nest
edFolders=true annotationPermissionUpdate=true cloudWatchRoundUpEndTime=true lokiQueryHints=true prometheusAzureOverrideAudience=true topnav=true amsAsyncQueryCaching=tru
e lokiStructuredMetadata=true pinNavItems=true influxdbBackendMigration=true tlsMemcached=true notificationBanner=true exploreMetrics=true angularDeprecationUI=true logsC
```


Cartservice


```
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=9SIQT8TOJO, 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 ~]$ kubectl logs pod/opentelemetry-demo-shippingservice-fcfc7765-98h9z -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: 81221474-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: "380"
20:24:56 [INFO] Sending Quote: 380.0
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
20:25:55 [INFO] Received quote: "203.1"
20:25:55 [INFO] Sending Quote: 203.10
20:25:55 [INFO] Tracking ID Created: 4e50d228-4392-42ce-9162-652bbad4683d
20:26:47 [INFO] Received quote: "1580"
20:26:47 [INFO] Sending Quote: 1580.0
20:26:47 [INFO] Tracking ID Created: 6fcb7c64-8075-401b-87ee-18adddbd6e3c
20:26:49 [INFO] Received quote: "441"
20:26:49 [INFO] Sending Quote: 441.0
20:26:49 [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 Accessing Application

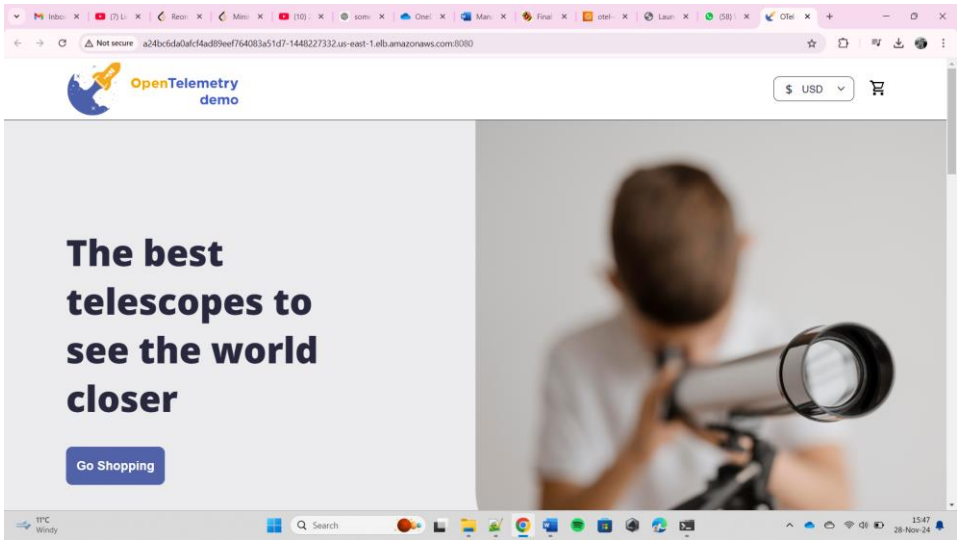
Port Forwarding Test:
Verified local access via port-forwarding for the frontendproxy service:
`kubectl port-forward svc/opentelemetry-demo-frontendproxy -n otel-demo 8080:8080`
Identified that we're unable to access the application through a browser, so we configured the frontendpoxy service to use a LoadBalancer.

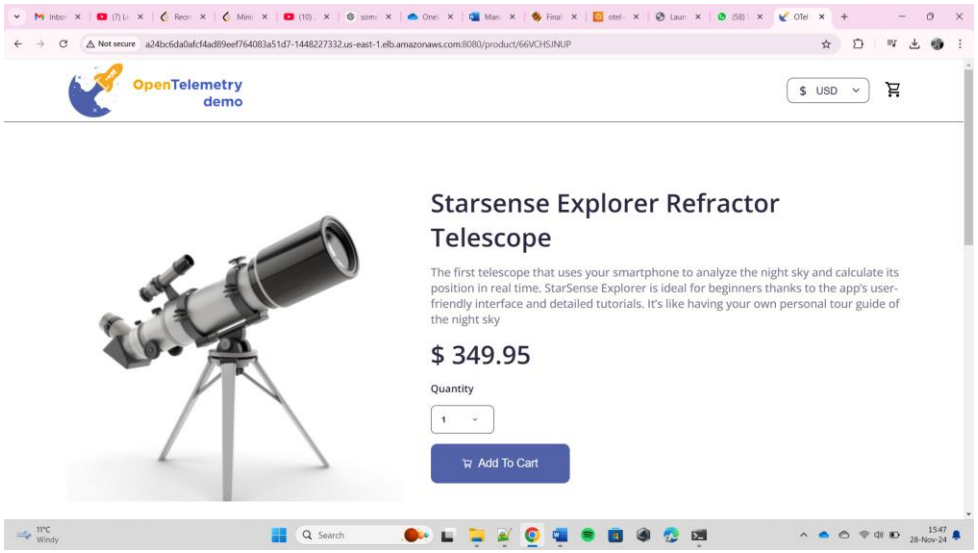
LoadBalancer Configuration:

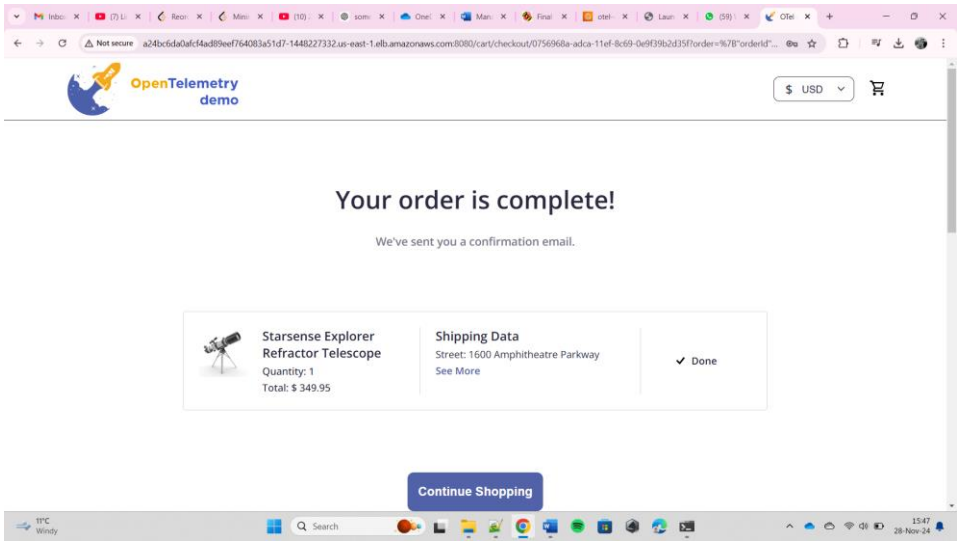
- Edited the frontendproxy service to include LoadBalancer type:
  - kubectl edit svc opentelemetry-demo-frontendproxy -n otel-demo
- Changed the type field:
  - type: LoadBalancer
- Verified the external IP:
  - kubectl get svc opentelemetry-demo-frontendproxy -n otel-demo

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

```
[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   a24bc6da0afcf4ad89eef764083a51d7-1448227332.us-east-1.elb.amazonaws.com   8080:30994/TCP   36m
[ec2-user@ip-10-0-1-11 kubernetes]$
```


OpenTelemetry demo
$ USD
The best telescopes to see the world closer
Go Shopping


OpenTelemetry demo
$ USD
Starsense Explorer Refractor Telescope
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

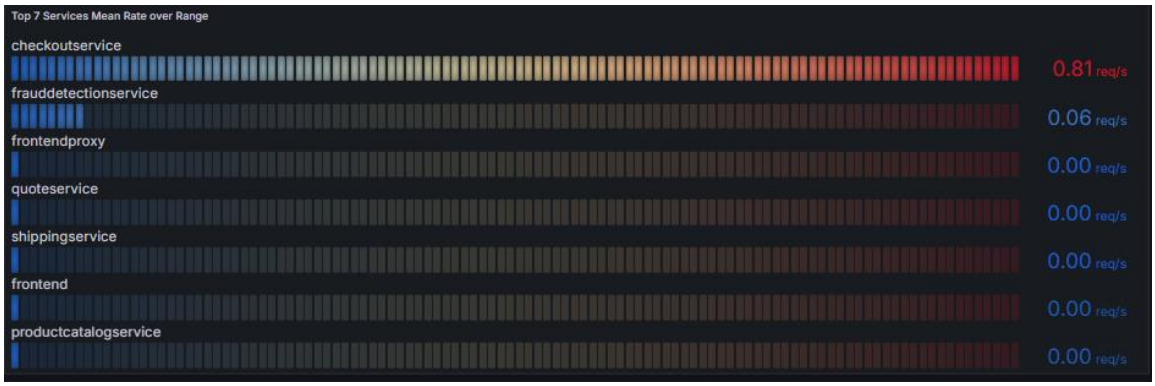

OpenTelemetry demo
$ USD
Your order is complete!
We've sent you a confirmation email.
Starsense Explorer Refractor Telescope
Quantity: 1
Total: $ 349.95
Shipping Data
Street: 1600 Amphitheatre Parkway
See More
Done
Continue Shopping

2. Phase-4: Customizing Grafana Dashboards for EKS Monitoring

In this phase, we aimed to monitor Kubernetes (EKS) metrics in Grafana, such as node status, pod-level metrics, and cluster health. While the default OpenTelemetry deployment provided application-level metrics and traces, it lacked Kubernetes-specific data like node and pod resource usage. To achieve this, we extended the setup by enabling Prometheus scrape targets for EKS metrics and configuring the OpenTelemetry Collector (otel-collector) to push these metrics to Prometheus.

2.1 Leveraging Pre-Built OpenTelemetry Dashboards

1. Verified that Prometheus and Grafana were deployed as part of the OpenTelemetry Helm release:
   - kubectl get pods-n otel-demo-helm
   - kubectl get svc-n otel-demo-helm
2. Checked the Grafana LoadBalancer endpoint to access the Grafana UI:
   - kubectl get svc my-otel-demo-grafana-n otel-demo-helm
3. Logged into Grafana using the default admin credentials:
   - Username: admin
   - Password: Retrieved from the Helm release's secret.
4. Explored the default dashboards in Grafana:
   - Found several pre-configured dashboards having application-level metrics (traces, latency, errors).
   - Kubernetes metrics were missing.



2.2 Customize Dashboards for EKS Metrics

2.2.1 Configuring Prometheus to scrape Kubernetes Metrics

Steps taken to ensure that EKS metrics are visible in Grafana (Phase 4 Config File) –

1. Enabled kube-state-metrics, node-exporter, and pushgateway in Prometheus Helm values:

```yaml
prometheus:
  configmapReload:
    prometheus:
      enabled: true
  prometheus-pushgateway:
    enabled: true

  kube-state-metrics:
    enabled: true

  prometheus-node-exporter:
    enabled: true
```

Also enabled configmapReload for Prometheus so that upon detecting a change, it triggers Prometheus to reload the configuration files and apply the new settings.

2. Added scrape targets for these components in prometheus.yaml:

```yaml
serverFiles:
  prometheus.yml:
    scrape_configs:
      - job_name: 'otel-collector'
        honor_labels: true
        kubernetes_sd_configs:
          - role: pod
            namespaces:
              own_namespace: true
        relabel_configs:
          - source_labels: [__meta_kubernetes_pod_annotation_opentelemetry_community_demo]
            action: keep
            regex: true
      - job_name: 'kube-state-metrics'
        static_configs:
          - targets: ['my-otel-demo-kube-state-metrics.otel-demo-helm:8080']
      - job_name: 'node-exporter'
        static_configs:
          - targets: ['my-otel-demo-prometheus-node-exporter.otel-demo-helm:9100']
      - job_name: 'pushgateway'
        static_configs:
          - targets: ['my-otel-demo-prometheus-pushgateway.otel-demo-helm:9091']
```

This scrape configuration allows Prometheus to scrape metrics from OpenTelemetry Collector pods. Specifically, it uses Kubernetes service discovery (kubernetes_sd_configs) to identify pods that are part of the same namespace as Prometheus. This is important for integrating Prometheus with OpenTelemetry to monitor metrics collected by the OpenTelemetry Collector. The relabel_configs ensures that only the pods annotated with opentelemetry_community_demo are scraped by Prometheus.

The targets ensure that Prometheus collects metrics from all relevant components in the Kubernetes environment.

Now, while Prometheus collected metrics from kube-state-metrics and node-exporter, we needed otel-collector to act as an intermediary to process and export metrics to Prometheus and other backends like OpenSearch.

3. Adding otel-collector configuration

```yaml
opentelemetry-collector:
  config:
    receivers:
      otlp:
        protocols:
          http:
            # Since this collector needs to receive data from the web, enable cors for all origins
            # `allowed_origins` can be refined for your deployment domain
            cors:
              allowed_origins:
                - "http://*"
                - "https://*"
      httpcheck/frontendproxy:
        targets:
          - endpoint: 'http://{{ include "otel-demo.name" . }}-frontendproxy:8080'
      redis:
        endpoint: "valkey-cart:6379"
        collection_interval: 10s
      prometheus:
        config:
          scrape_configs:
            - job_name: 'kube-state-metrics'
              static_configs:
                - targets: ['kube-state-metrics.kube-system:8080']
            - job_name: 'node-exporter'
              static_configs:
                - targets: ['node-exporter.kube-system:9100']
```

The OTLP (OpenTelemetry Protocol) receiver is configured to allow the collection of traces and metrics via HTTP. The CORS settings allow data to be received from various sources (potentially from different domains), which is necessary for collecting telemetry data from web applications. The allowed_origins wildcard ensures that the OTLP receiver can accept data from any origin.

The otel-collector is also set up to scrape Kubernetes metrics (using Prometheus scraping) from the kube-state-metrics and node-exporter services running in the Kubernetes cluster. This allows OpenTelemetry Collector to act as an intermediary for processing, transforming, and exporting these metrics to other systems, such as Prometheus or Jaeger.

4. Adding exporters configuration

```
exporters:
  ## Create an exporter to Jaeger using the standard `otlp` export format
  otlp:
    endpoint: '{{ include "otel-demo.name" . }}-jaeger-collector:4317'
    tls:
      insecure: true
  # Create an exporter to Prometheus (metrics)
  otlphttp/prometheus:
    endpoint: 'http://{{ include "otel-demo.name" . }}-prometheus-server:9090/api/v1/otlp'
    tls:
      insecure: true
  opensearch:
    logs_index: otel
    http:
      endpoint: "http://otel-demo-opensearch:9200"
      tls:
        insecure: true
```

The exporters define where the metrics and traces collected by the OpenTelemetry Collector should be sent:

- Jaeger: Metrics and traces can be exported to Jaeger using the OTLP protocol for tracing.
- Prometheus: Metrics are also exported to Prometheus via an OTLP endpoint, which is particularly useful for pushing processed metrics back to Prometheus for monitoring and visualization.
- OpenSearch: Logs can be sent to OpenSearch for logging and storage.

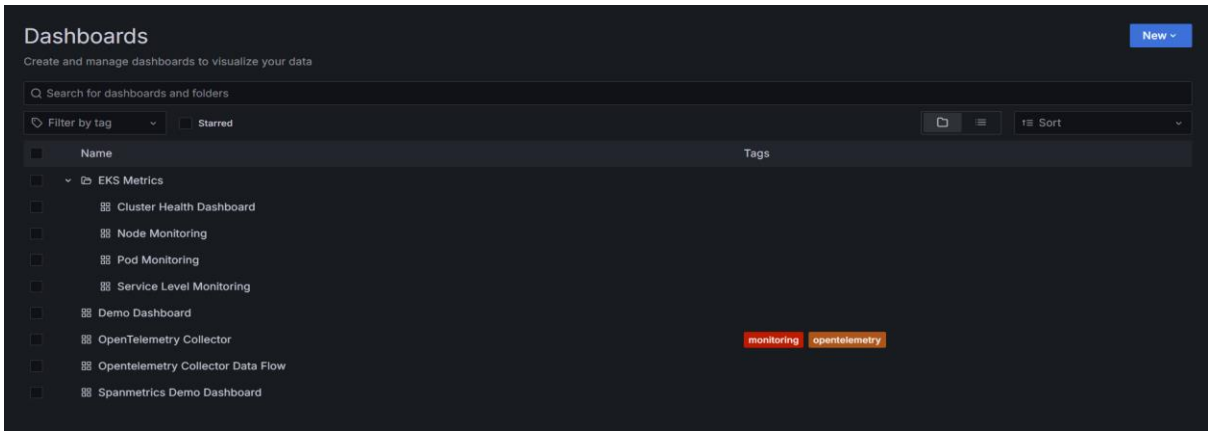
These exporters ensure that telemetry data is pushed to appropriate backends for further processing and analysis.

5. Finally, we applied the updated configuration via Helm upgrade and validated the otel-collector logs:
    - helm upgrade my-otel-demo open-telemetry/opentelemetry-demo \--namespace otel-demo-helm-f updated-values.yaml
    - kubectl logs <otel-collector-pod-name>-n otel-demo-helm

2.2.2 Creating Dashboards

Once we were able to see the kube_ and node_ metrics in Grafana, we proceeded with creating separate dashboards for metrics of:

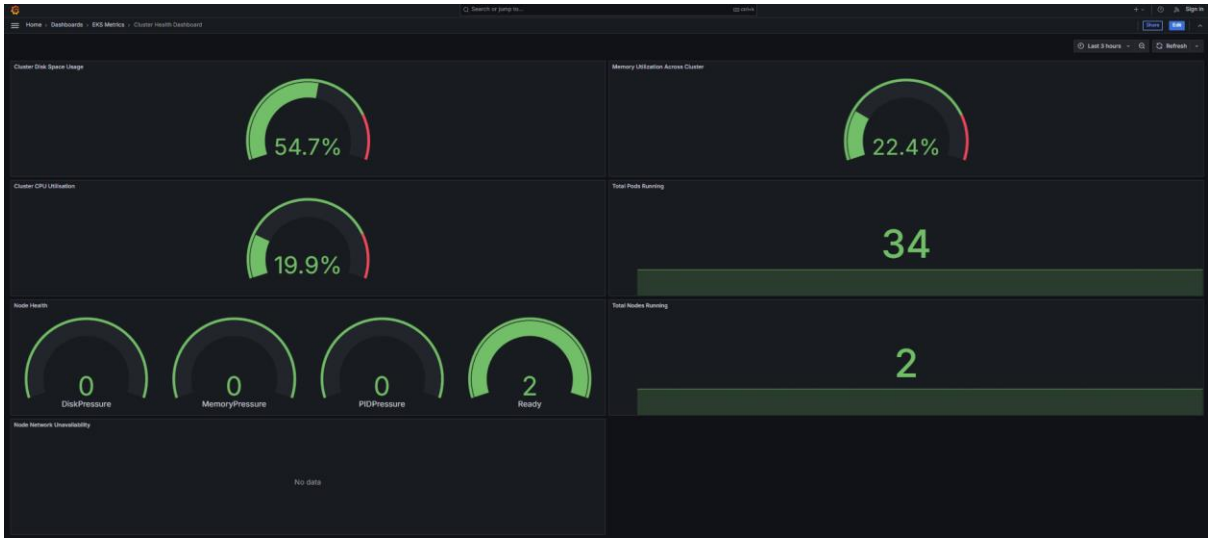
- Cluster-Level
- Node-Level
- Pod-Level
- Service-Level



2.2.2.1 Cluster-Level

The cluster health dashboard provides an at-a-glance view of the current state of the EKS cluster. Visualizations include:

- **Cluster Disk Space Usage**: Displays the current disk space usage across the cluster. The current usage is 54.7% which indicates that a little over half of the total disk space in the cluster is in use.
- **Memory Utilization Across Cluster**: The memory utilization displays the memory usage by the cluster using the formula (1 – AvailableMemory/TotalMemory). Currently, the cluster is currently at 22.4%, which indicates there is still ample available memory capacity for the workloads running on the cluster.
- **CPU Utilization Across Cluster**: Represents the percentage of CPU resources in use across the cluster. The current utilization is 19.9%, suggesting the compute resources are not being heavily utilized and there is room for additional workloads if needed.
- **Total Pods Running**: Displays the total number of pods currently running in the cluster. The total number of pods currently running in the cluster is 34, which appears to be a normal and expected level of activity.
- **Total Nodes Running**: Shows the number of nodes currently up and operational within the cluster. Currently, 2 nodes are running and functioning.
- **Node Health**: Indicates the health status of the nodes in terms of DiskPressure, MemoryPressure, PIDPressure and Readily available nodes. All pressure metrics (DiskPressure, MemoryPressure, PIDPressure) are at 0, indicating no issues. 2 nodes are marked as ready.
- **Node Network Unavailability**: Monitors the network availability of nodes in the cluster. Since there are no node failures, there is no data present in this visualization.



Since Grafana alerts can only be set on time series charts. Gauge and other visualization types (like pie charts or heatmaps) do not currently support alerts directly as alerts in Grafana rely on evaluating a time series over a defined period. Since gauge visualizations display a single-point value rather than a time-series trend, they can't directly leverage the alerting mechanism.

To rectify this and to be able to set up alerts for critical metrics, I changed a few visualizations to a Time-series chart.

By closely monitoring the cluster health and acting on any emerging issues, we can help ensure the EKS environment continues to perform reliably and efficiently to support the critical workloads.

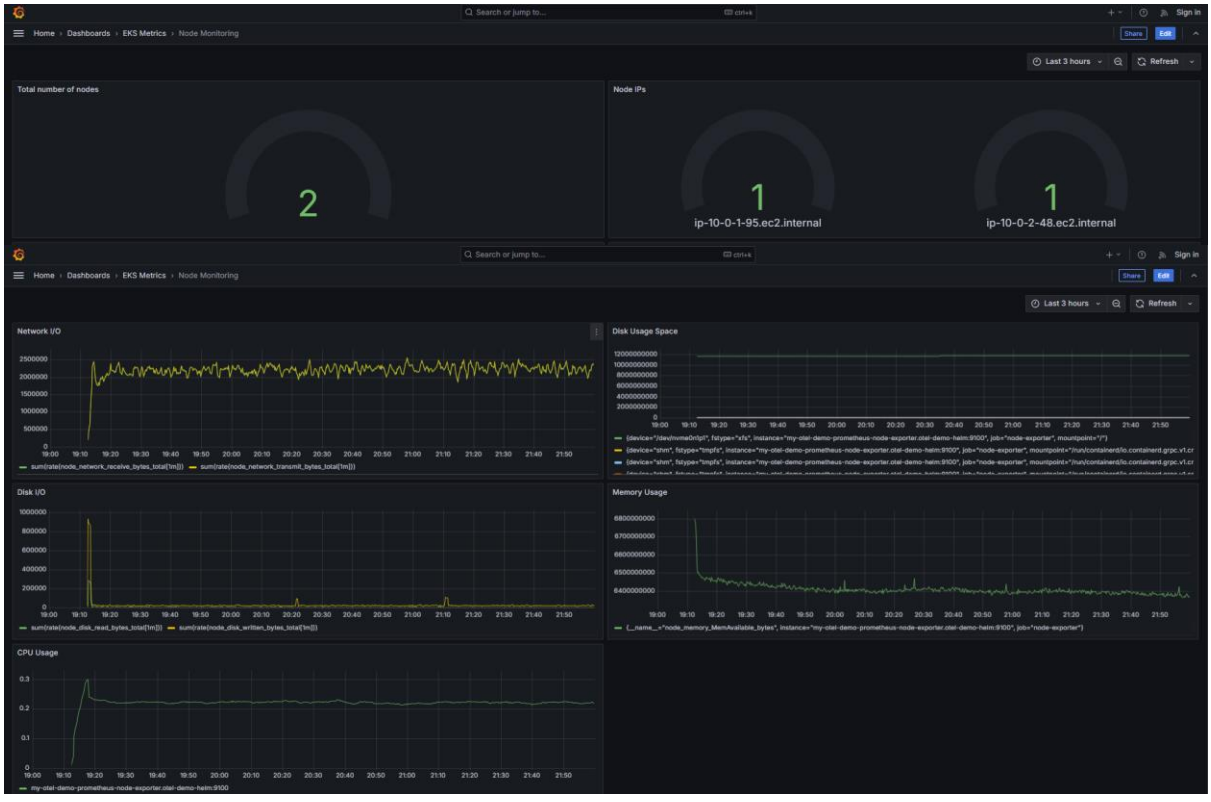
JSON File of Dashboard

2.2.2.2 Node-Level

The node monitoring dashboard provides visibility into the key metrics and status of the nodes within the EKS cluster. Visualizations include:

- **Total Number of Nodes**: Displays the number of nodes running in the cluster at a time. The cluster currently has 2 nodes running, which appears to be the expected configuration.
- **Node IPs**: Indicates the IPs of the nodes. The dashboard shows the 2 nodes with the following IP addresses:
    - ip-10-0-1-95.ec2.internal
    - ip-10-0-2-48.ec2.internal
- **Node Resource Utilization**: The CPU, memory, and disk usage charts show the resource consumption patterns across the 2 nodes over the last 3 hours:
    - **CPU Usage**: Measures the CPU utilization of the node over time. It has been relatively low, fluctuating between 5-20% on average, which suggests the nodes have ample compute capacity available.
    - **Memory Usage**: Monitors available memory on the node. Memory utilization has been stable, hovering around 65-70% of total capacity. This indicates the nodes have enough memory to support the current workloads.
    - **Disk Usage**: Displays the disk space usage across various devices and filesystems mounted on the node. Disk usage appears stable with minimal changes over time, indicating consistent disk space utilization.
- **Network I/O**: Tracks the rate of network data received and transmitted by the node.
- **Disk I/O**: Tracks the rate of disk read and write operations. Initial spike in both read and write operations, followed by a stable, low-level usage pattern.

JSON File of Dashboard

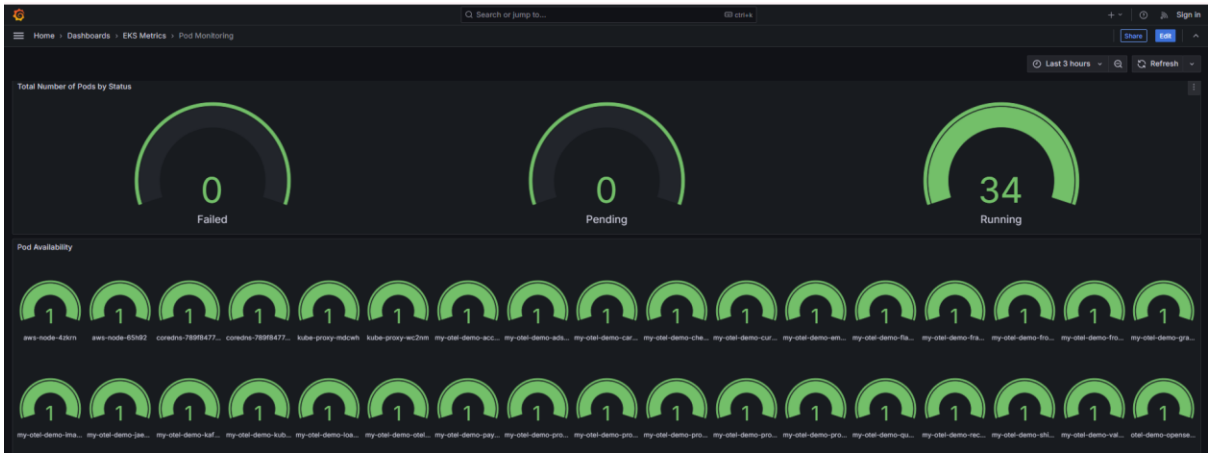


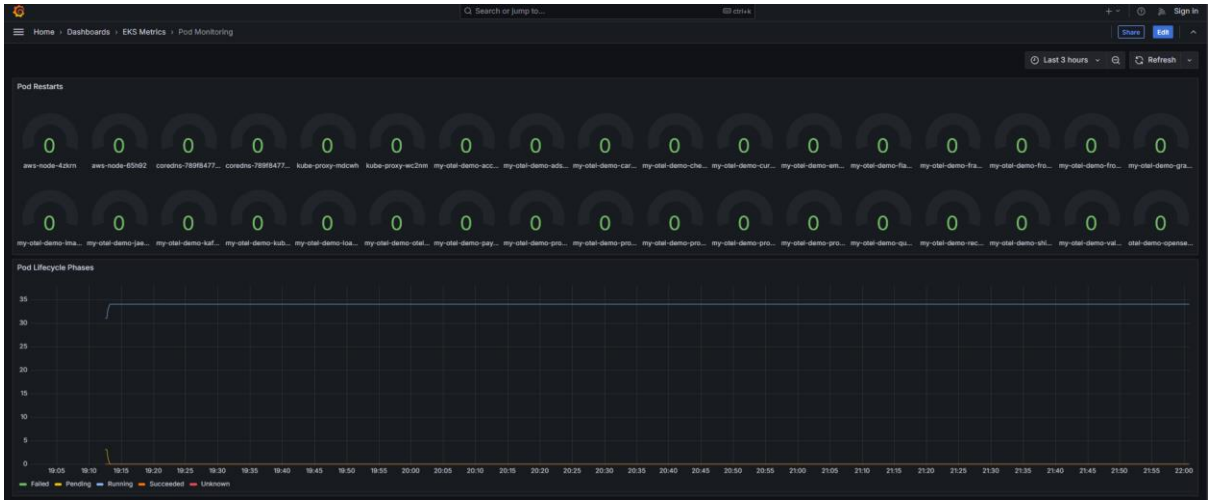
2.2.2.3 Pod-Level

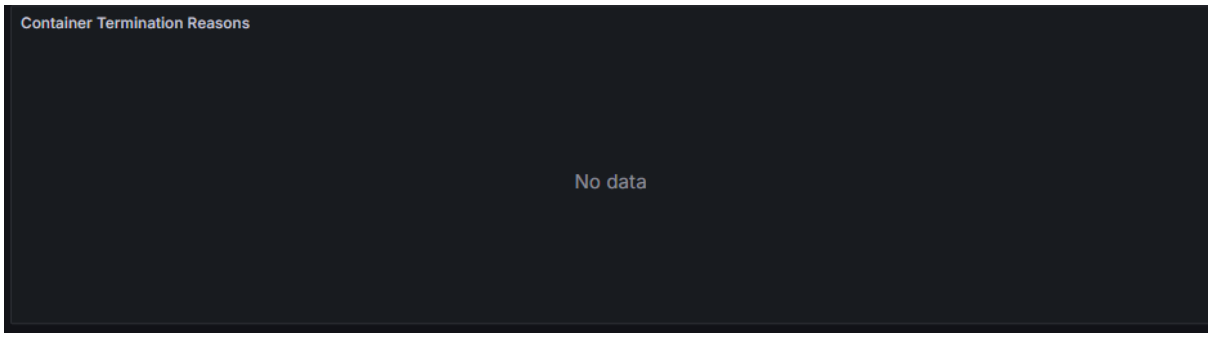
The pod monitoring dashboard provides visibility into the key metrics and status of the 34 pods within the EKS cluster. Visualizations include:

- **Total Number of Pods by Status**: Displays the number of failed, pending and running pods. Currently, there are 0 Failed pods, 0 Pending pods, and 34 Running pods.
- **Pod Availability**: Displays the status of each of the 34 pods so that it becomes easier to figure out which pod is down. Currently, 34 pods available out of 34, indicating 100% availability.
- **Memory Requests by Each Pod**: Graphs indicate the memory usage per pod.
- **CPU Requests by Each Pod**: Displays CPU utilization in terms of requested CPU shares. Pods like otel-demo-opensearch are actively consuming higher CPU (0.100 CPU shares), while most others maintain low usage (~0.050 CPU shares).
- **Total Pod Restarts**: Displays a count of the pod restarts over time.
- **Pod Restarts**: Breaks down the Total Pod Restarts visualization into 34 circular graphs, corresponding to different pods within the cluster, indicating the restart count made by each pod.
- **Pod Lifecycle Phases**: A time-series graph tracking the number of pods in various states (e.g., Running, Pending, Failed, etc.) at a given time.
- **Container Termination Reasons**: Tracks the reasons for container terminations across all pods in the EKS cluster. Currently, no pods have recently experienced terminations, suggesting stable cluster operations.

JSON File of Dashboard




Home › Dashboards › EKS Metrics › Pod Monitoring
Share
Edit
Last 3 hours
Refresh
Pod Restarts
Pod Lifecycle Phases
Failed
Pending
Running
Succeeded
Unknown

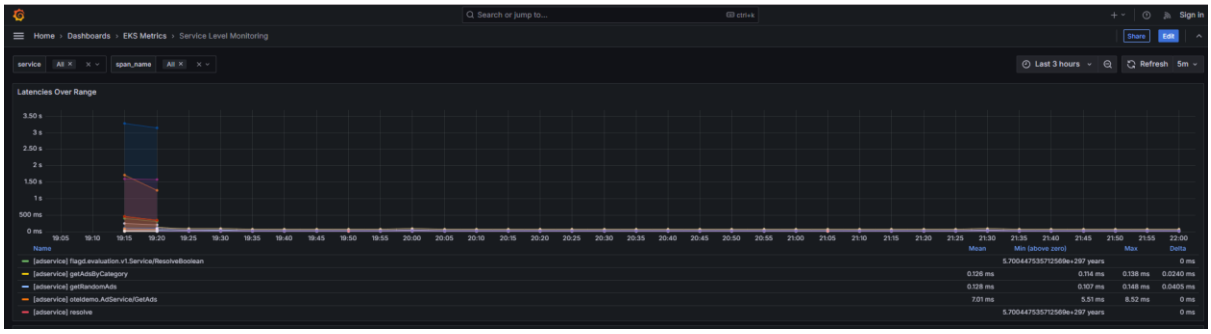

Container Termination Reasons
No data

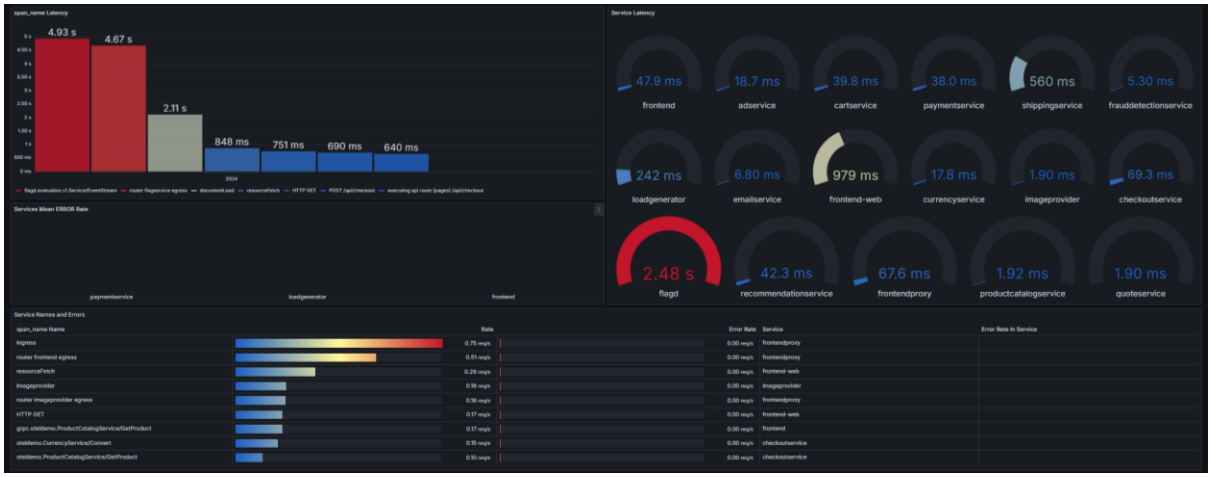
2.2.2.4 Service-Level

The dashboard represents service performances in the EKS Cluster. This dashboard is designed to track and analyse latency, call rates, and error rates for various services. Visualizations include:

- **Latencies Over Range**: Tracks and compares the latency trends for specific spans within services over time, aiding in identifying performance bottlenecks.
- **Services Mean Rate**: Provides a snapshot of the average request rate for each service, helping assess traffic load and service usage.
- **Span Name Latency**: Uses a 99.9th percentile quantile to highlight outlier latencies to pinpoint spans causing potential slowdowns.
- **Service Latency**: Like Span Name Latency, but identifies latency outliers at the service level, enabling a high-level performance overview.
- **Services Mean ERROR Rate**: Tracks the error rate for services.
- **Service Names and Errors**: Breaks down error counts by service and span name.

JSON File of Dashboard



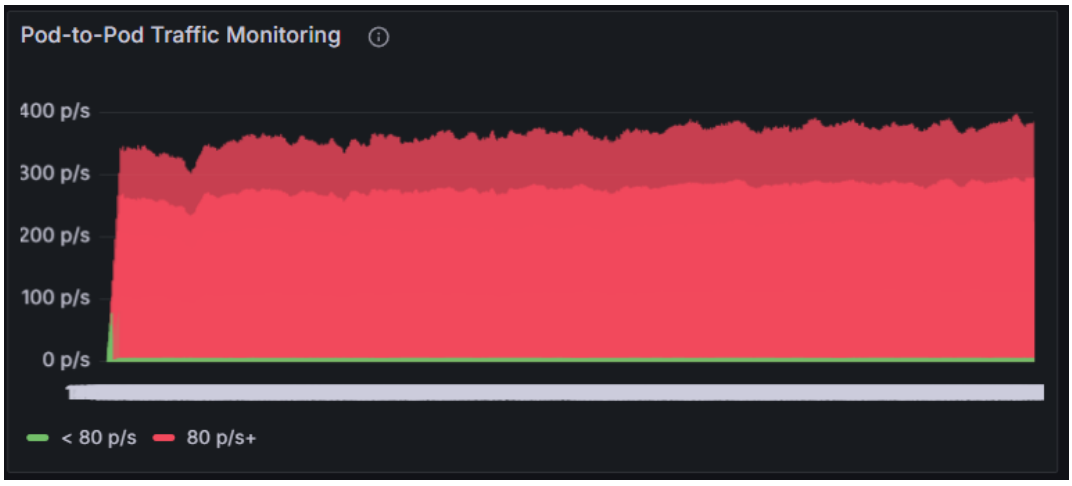


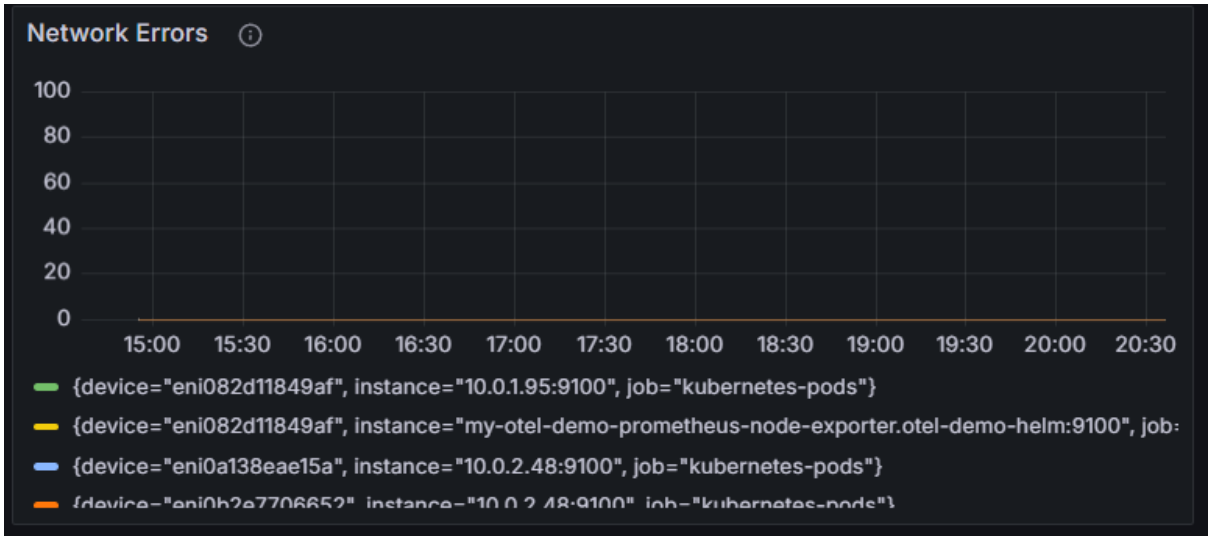
2.3 Extending Dashboards to Include Network and Storage Monitoring

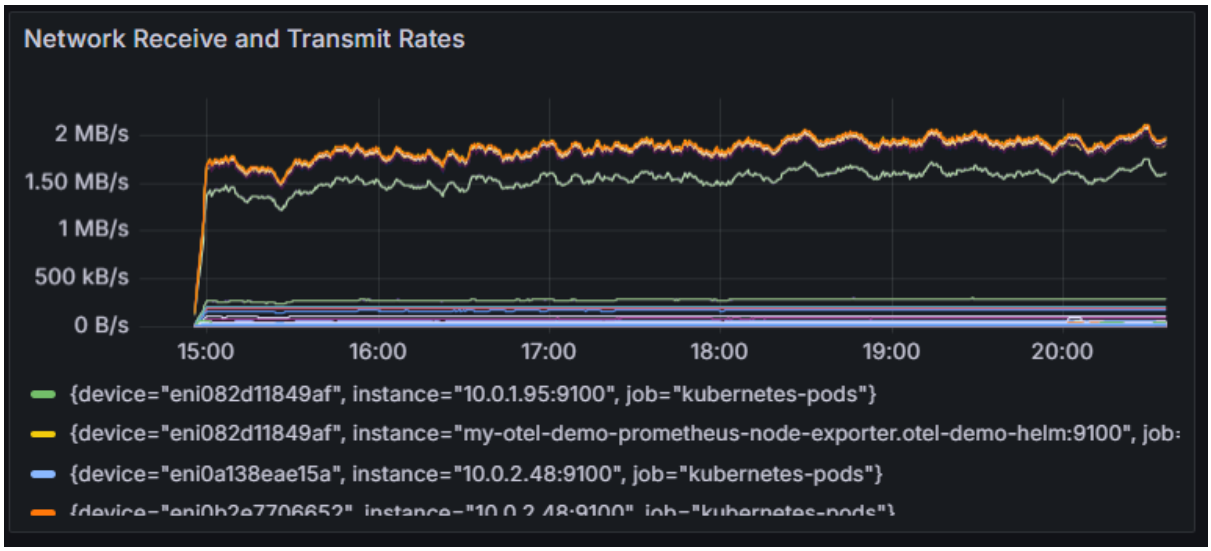
2.3.1 Network Monitoring

Track the flow of network traffic between pods, identify issues like packet drops or latency, and ensure seamless communication within the cluster.
Visualizations built for network monitoring are:

- **Network Latency**: Monitors the latency (in milliseconds) for HTTP requests sent to the frontend proxy.
- **Pod-to-Pod Traffic Monitoring**: Measures the packet traffic received across pods in the cluster over time.
- **Network Errors**: Counts network errors during data reception or transmission over a 5-minute window.
- **Network Receive and Transmit Rates**: Measures the rate of network data received and transmitted, excluding loopback traffic, over a period.



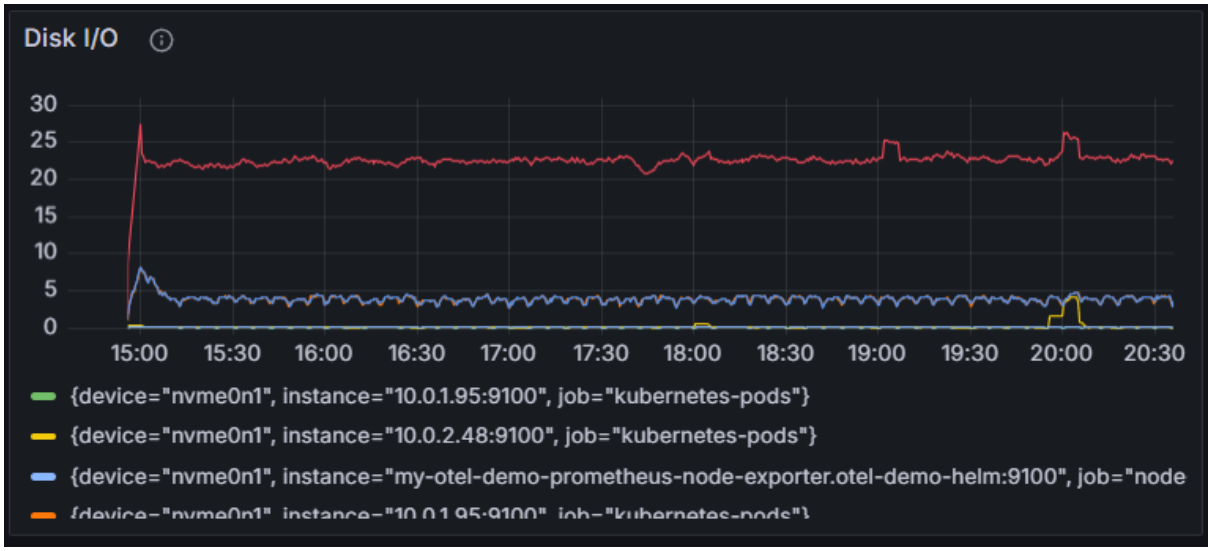

Network Errors
100
80
60
40
20
0
15:00
15:30
16:00
16:30
17:00
17:30
18:00
18:30
19:00
19:30
20:00
20:30
{device="eni082d11849af", instance="10.0.1.95:9100", job="kubernetes-pods"}
{device="eni082d11849af", instance="my-otel-demo-prometheus-node-exporter.otel-demo-helm:9100", job:
{device="eni0a138eae15a", instance="10.0.2.48:9100", job="kubernetes-pods"}

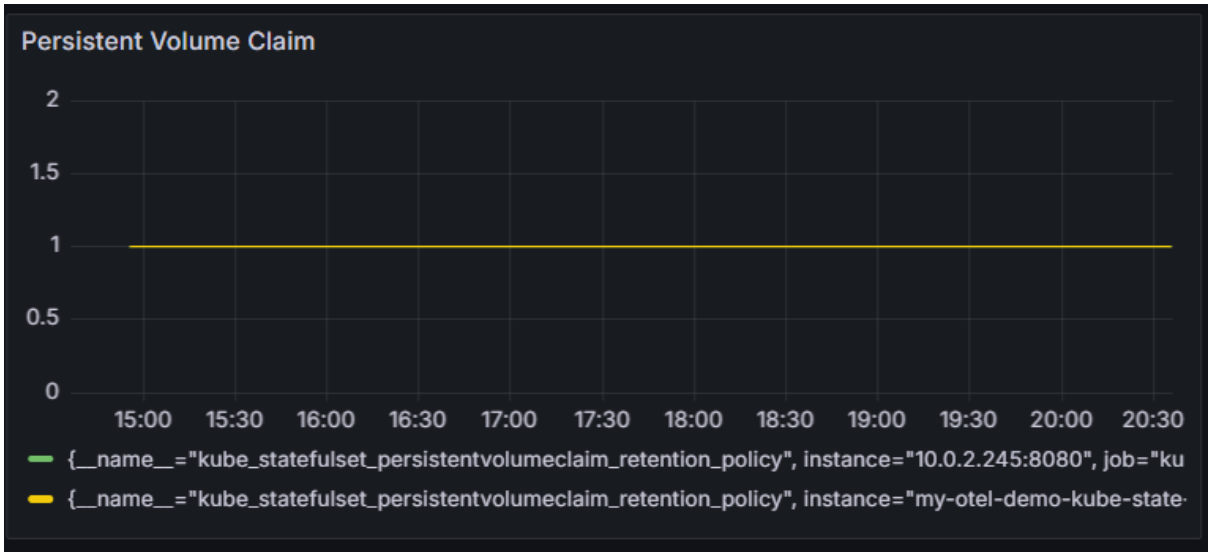

Network Receive and Transmit Rates
2 MB/s
1.50 MB/s
1 MB/s
500 kB/s
0 B/s
15:00
16:00
17:00
18:00
19:00
20:00
{device="eni082d11849af", instance="10.0.1.95:9100", job="kubernetes-pods"}
{device="eni082d11849af", instance="my-otel-demo-prometheus-node-exporter.otel-demo-helm:9100", job:
{device="eni0a138eae15a", instance="10.0.2.48:9100", job="kubernetes-pods"}

2.3.2 Storage Monitoring

Monitor disk I/O operations and storage utilization to prevent resource contention and maintain system reliability. Visualizations built for network monitoring are:

- **Disk I/O**: Measures the rate of disk read and write operations per second over time. Helps in detecting I/O bottlenecks or resource contention impacting storage performance.
- **Persistent Volume Claim**: Monitors retention policies for persistent volume claims in Kubernetes. Ensures proper configuration and utilization of storage resources for stateful applications.





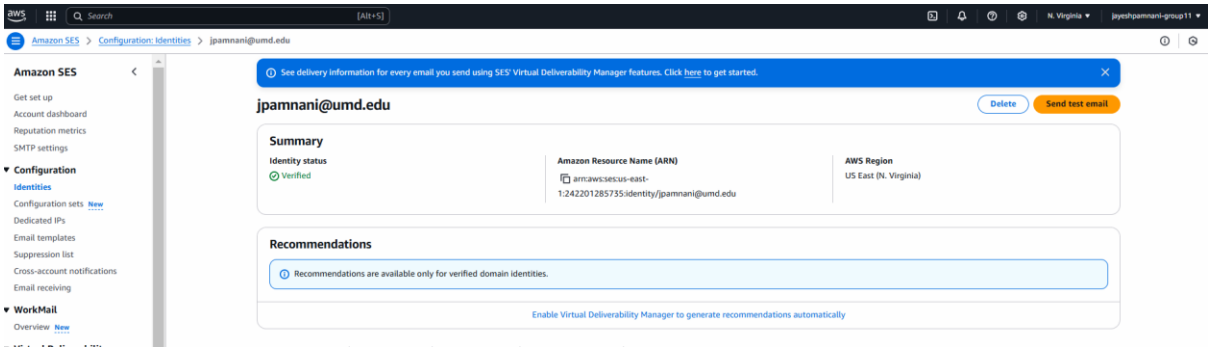
2.4 Setting Up Aerts in Grafana and Configuring Notifications

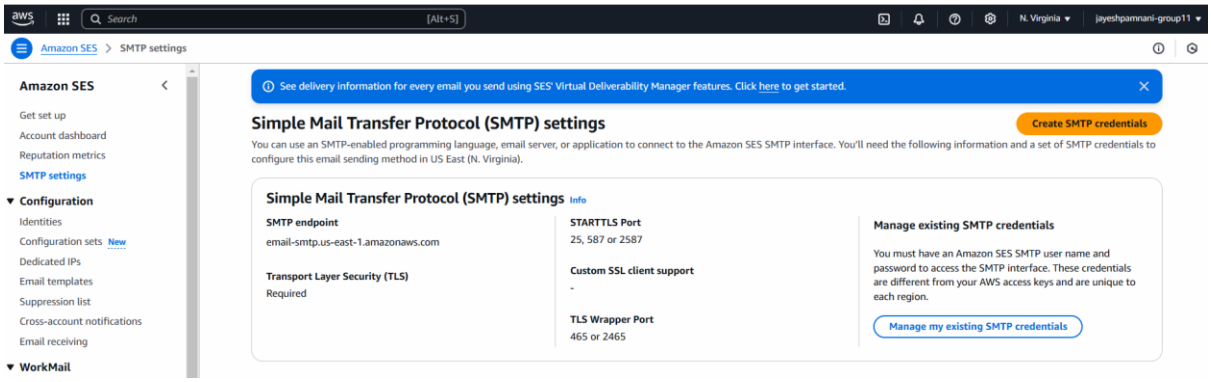
2.4.1 Email Setup

To send email alerts from Grafana, we used Amazon SES as the SMTP server.

Steps taken to configure email:

1. **Amazon SES Configuration**
    - Verified Email Address: Verified the email address in Amazon SES to enable sending emails from this address.
    - Obtained SMTP Credentials: Retrieved the SMTP username and password from the Amazon SES console.





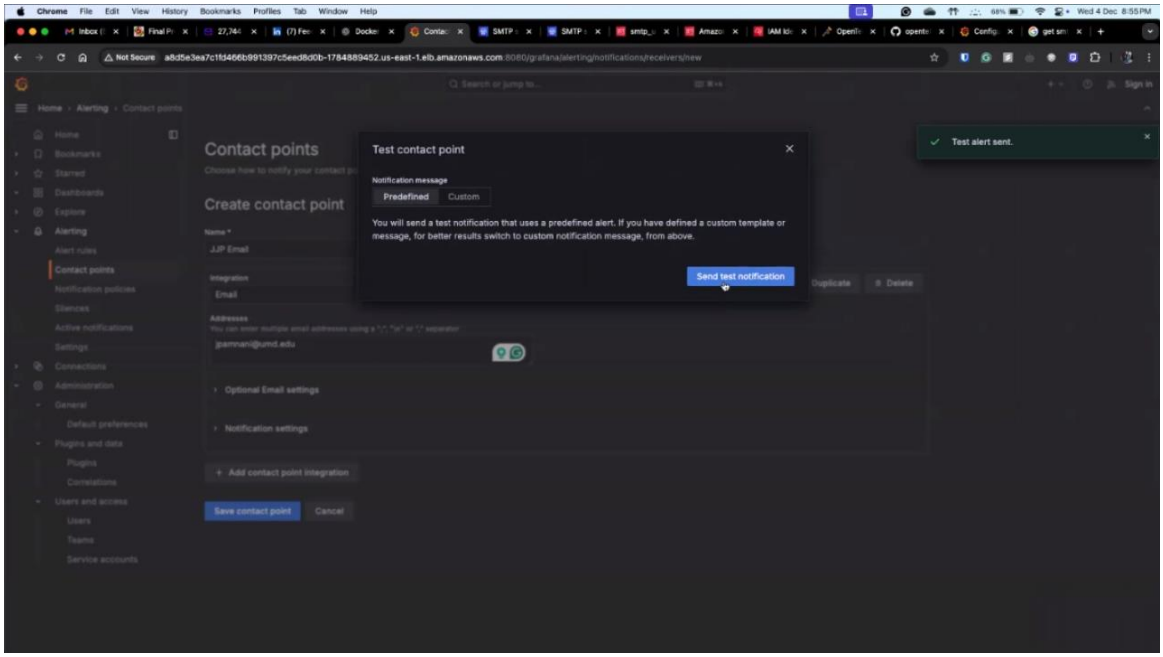
2. **Updating Grafana Configuration**
    - Added SMTP configurations to the Grafana values file for Helm.

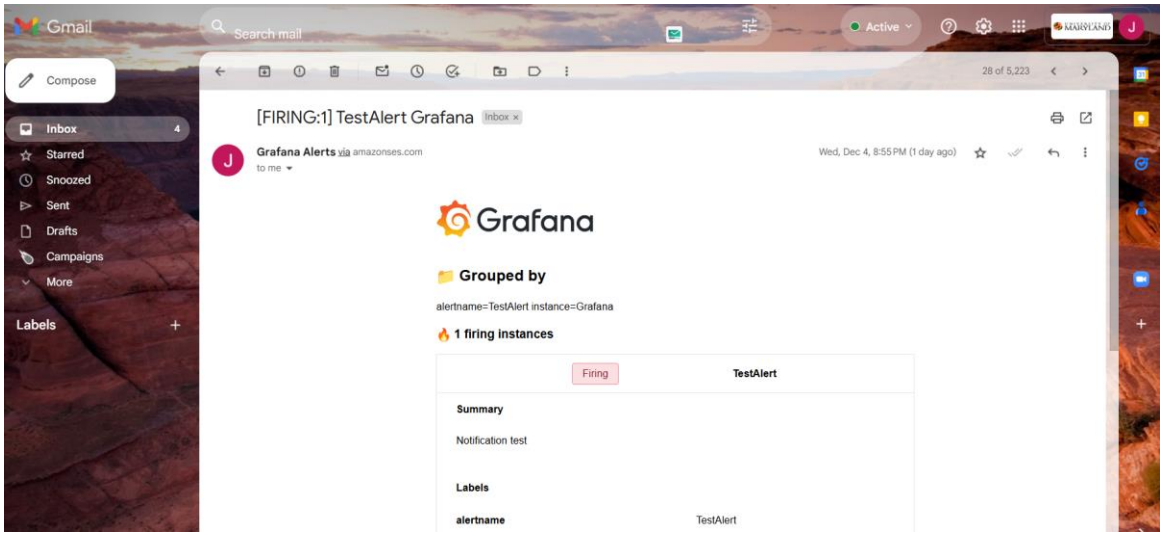
```yaml
grafana:
  enabled: true
  assertNoLeakedSecrets: false
  grafana.ini:
    smtp:
      enabled: true
      host: "email-smtp.us-east-1.amazonaws.com:587"  # Replace with the AWS SES SMTP endpoint
      user: "[redacted]"   # Replace with your AWS SES SMTP username
      password: "[redacted]"  # Replace with your AWS SES SMTP password
      from_address: "jpamnani@umd.edu"  # Replace with the email address used in AWS SES
      from_name: "Grafana Alerts"  # Sender name
      startTLS_policy: "OpportunisticStartTLS"
    unified_alerting:
      enabled: true
      email_notifications_enabled: true
```

- Temporarily disabled `assertNoLeakedSecrets` to bypass validation for testing purposes.

3. **Helm Upgrade**
   - Applied the updated configuration with the following command:
     helm upgrade my-otel-demo opentelemetry-demo-n otel-demo-helm-f updated-values.yaml

4. **Testing the Email Configuration**
   - Created a Contact Point in Grafana.
   - Added all details including the email which was verified on Amazon SES.
   - Ran a test email alert in Grafana and verified that email notifications were received successfully.





The configuration was tested successfully, verifying that emails are received when alerts fire, ensuring that critical alert notifications are sent to the configured email addresses promptly.

2.4.2 Alerting on Grafana

Grafana provides robust alerting capabilities for monitoring critical system metrics and notifying users when specific thresholds are breached.

Visualization Type Limitation: Grafana's alerting mechanism works exclusively with time-series visualizations. Visualization types like gauge, pie charts, or heatmaps, which show single-point values, are not directly compatible with alerting since they lack the required time-series trend.

Visualization Adjustment:
To enable alerting for critical metrics:

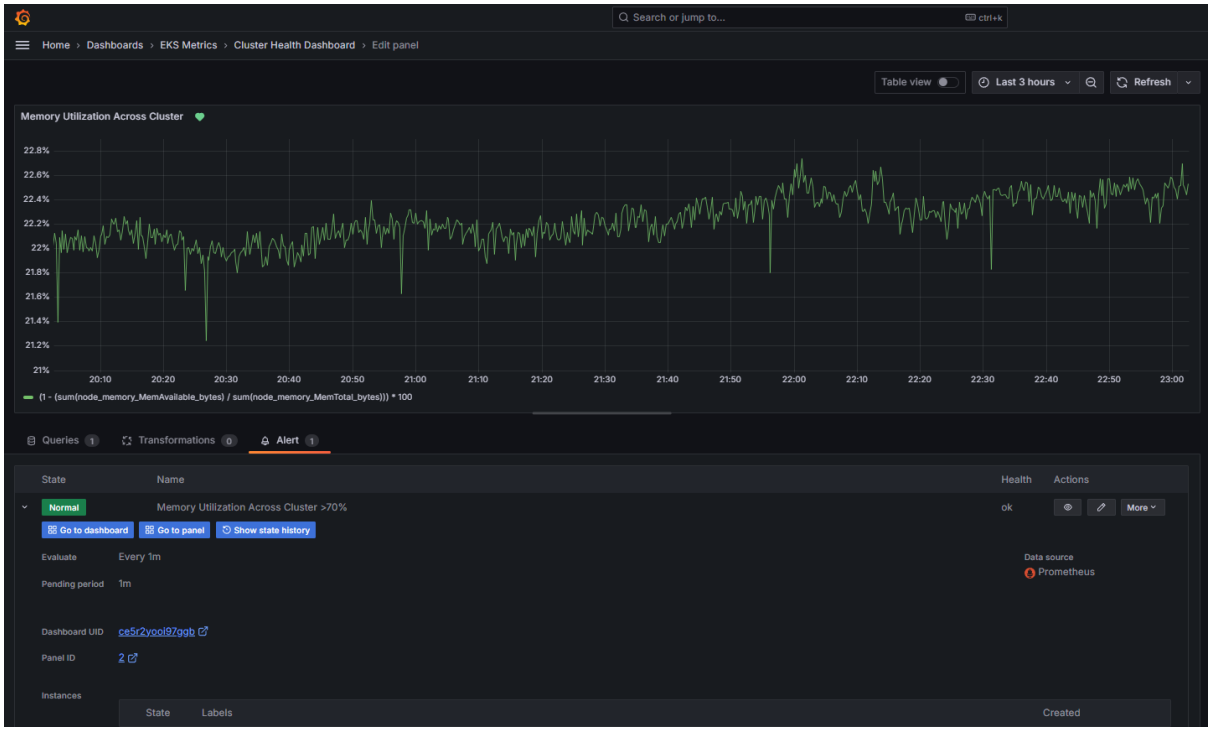
- Certain visualizations were ***converted*** to **time-series charts**.
- This change allowed us to define and evaluate alert conditions based on trends over time, ensuring reliable and actionable alerts.

Alerts set up on Grafana:

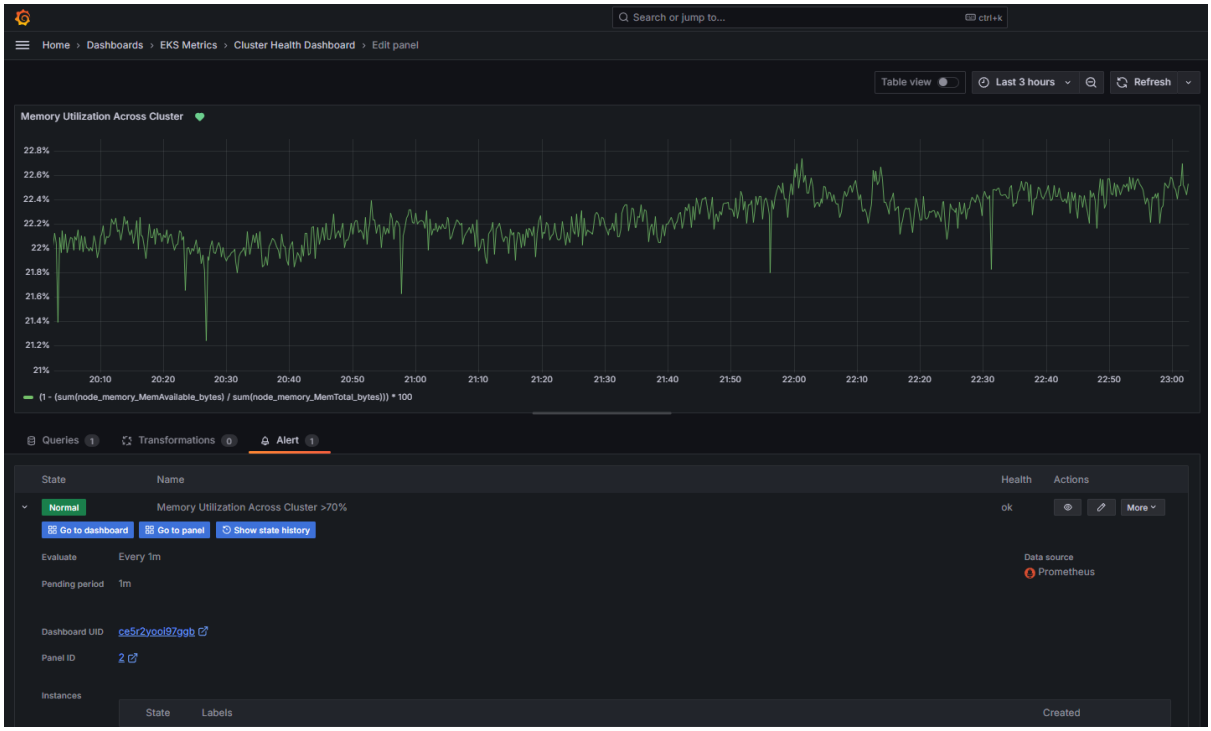
2.4.2.1 Cluster-Level Monitoring

Cluster-wide alerts were designed to monitor aggregate resource usage and workload distribution across the entire cluster.

1. **Cluster CPU Usage Alert (>65%)**: Ensures the CPU usage at the cluster level remains below 65%. Helps prevent over-provisioning and ensures enough capacity for additional workloads.



2. **Cluster Memory Usage Alert (>70%)**: Monitors the percentage of memory utilized by the cluster. Ensures resource limits are not exceeded, preventing system crashes or slowdowns.

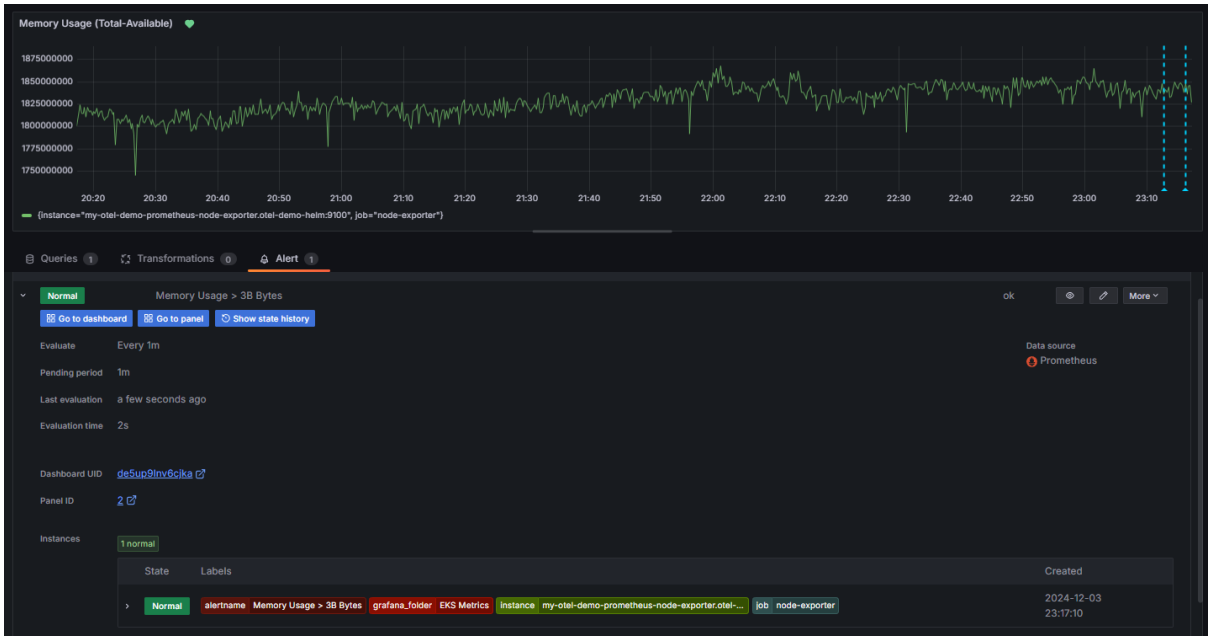


2.4.2.2 Node-Level Monitoring

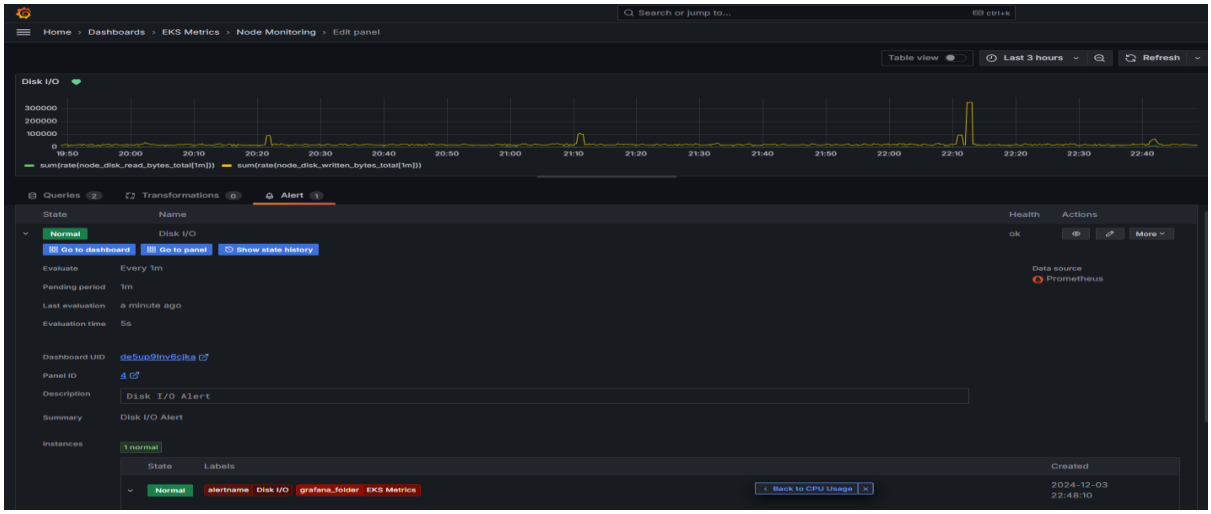
Node-level alerts were configured to track individual system resource metrics, ensuring optimal resource utilization and availability.

1. **CPU Usage Alert (>70%)**: Tracks CPU usage across nodes and triggers if usage exceeds 70%.

2. **Memory Usage Alert (>3B Bytes)**: Monitors memory consumption per node. Alerts if a node's memory exceeds 3GB, indicating potential memory pressure.



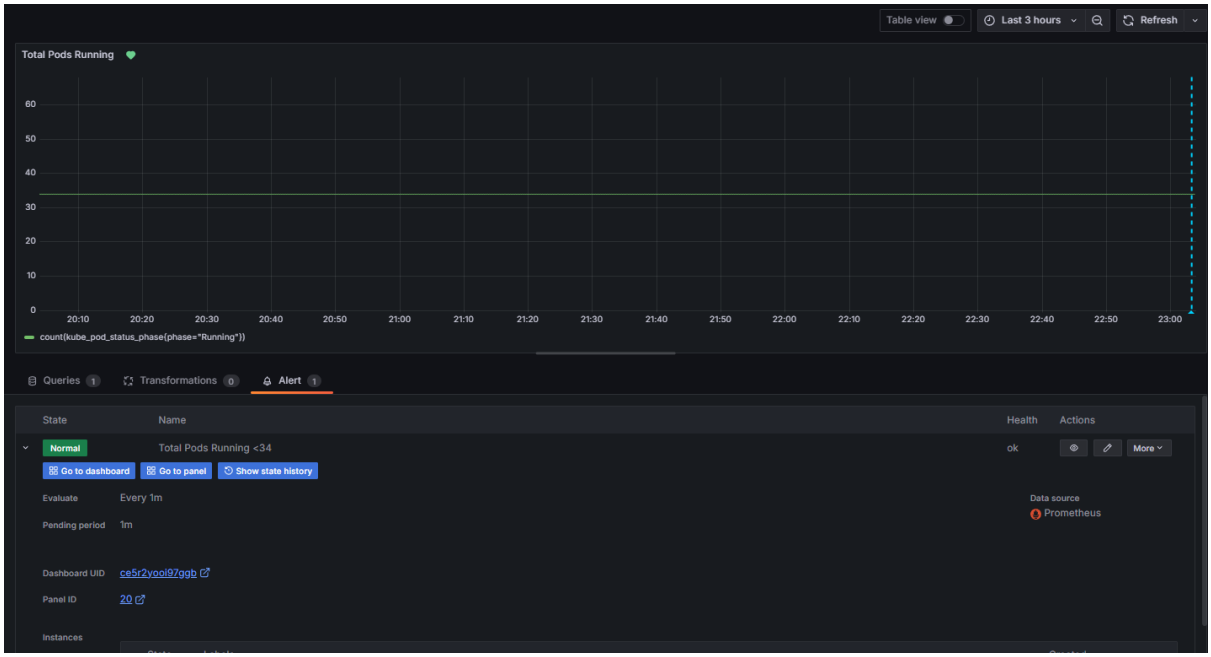
3. **Disk I/O Alert (>500,000,000 Bytes)**: Monitors disk read/write rates using 2 alerts. Alerts if read/write bytes exceed 500MB, signifying excessive disk activity.



4. **Total Nodes Alert (<2)**: Tracks the total number of nodes in the cluster. Triggers if the count falls below 2, indicating node unavailability or failure.

2.4.2.3 Pod-Level Monitoring

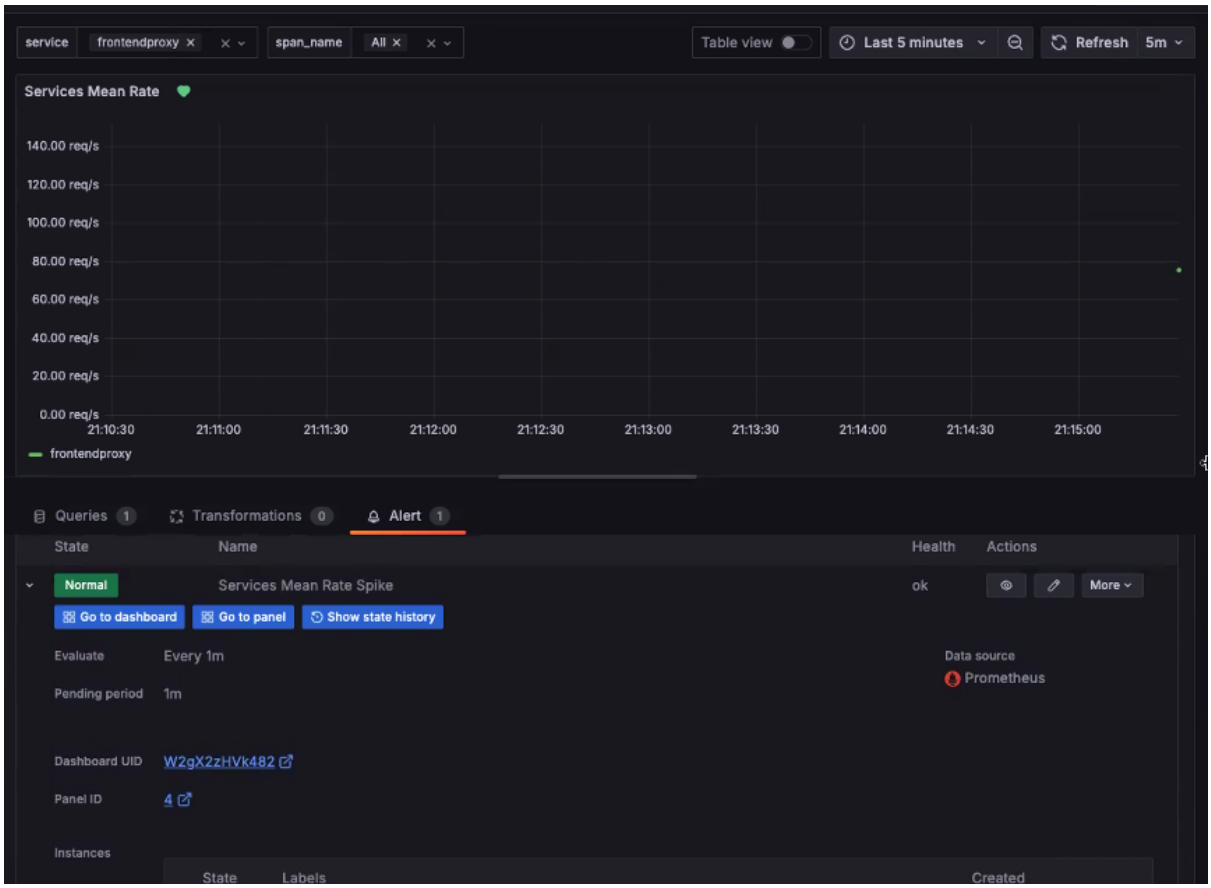
Pods Count Alert (<34 Pods): Alerts if the total number of pods in the cluster drops below 34.



2.4.2.4 Service-Level Monitoring

To monitor application performance during specific scenarios like load testing, service-level alerts were configured:

1. Request Rate (>120 requests/sec): Tracks the rate of incoming requests per service. Alerts if any service exceeds 120 requests per second at any given time.



2.4.3 Significance of These Alerts

- Proactive Monitoring: Alerts are designed to proactively detect performance issues and resource constraints before they escalate into critical problems.
- Resource Optimization: By keeping track of CPU, memory, and disk I/O, the alerts ensure balanced resource utilization, preventing node failures or unresponsiveness.
- Scalability Assurance: Service-level alerts validate that the system can handle increased traffic during load tests, ensuring reliability under peak conditions.
- Cluster Health: Monitoring total nodes, pod counts, and cluster-wide CPU/memory usage ensures the cluster remains healthy and functional.

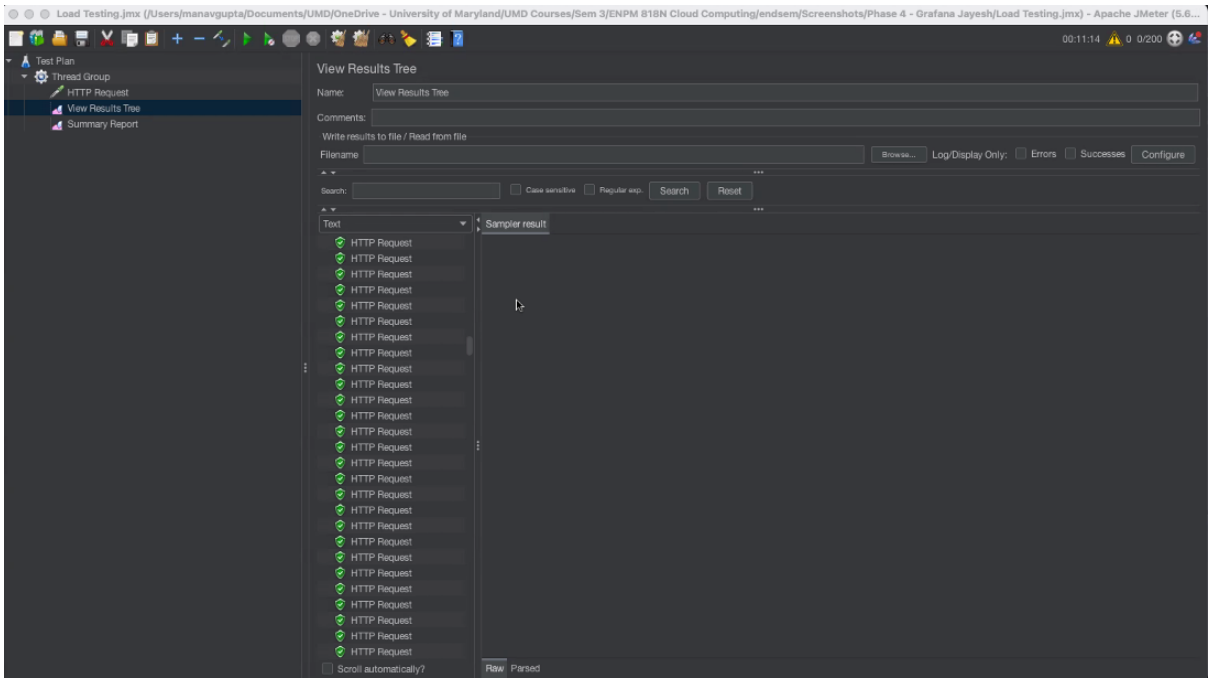
2.5 Load Testing and Dashboard Validation

To ensure the dashboards' effectiveness and accuracy, a load test was conducted using Apache JMeter. The testing aimed to simulate real-world traffic and validate that the dashboards provide actionable, real-time insights into system performance under load.

2.5.1 Testing Approach

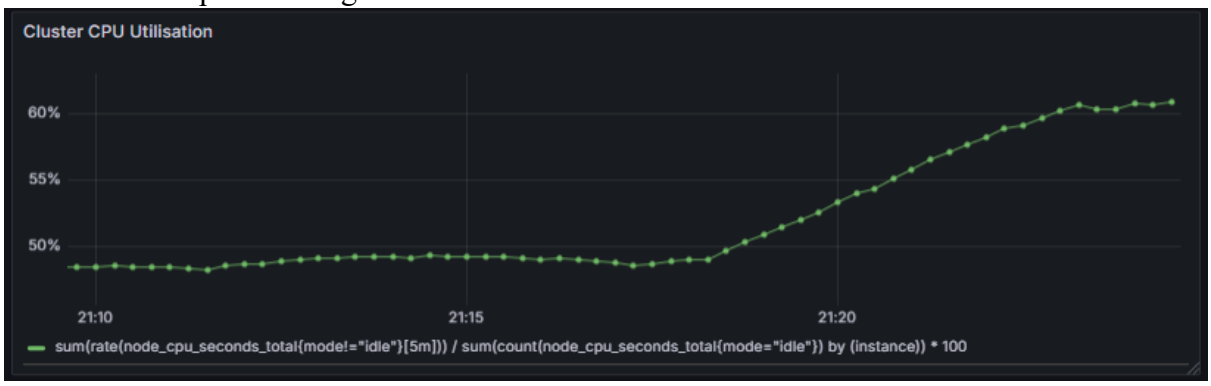
- **Tool Used**: JMeter was configured to simulate 200 concurrent users generating traffic indefinitely.
- **Metrics Observed**: The metrics that were monitored in real-time using Grafana are:
  - CPU utilization
  - Network traffic
  - Service latency
- **Objective**:
  - Validate that the dashboards accurately display real-time data.
  - Identify bottlenecks or areas of concern under load.

JMeter Requests: All requests passed successfully without errors. This validated that the dashboards accurately reflect real-time data and assist in identifying bottlenecks during high-load scenarios.

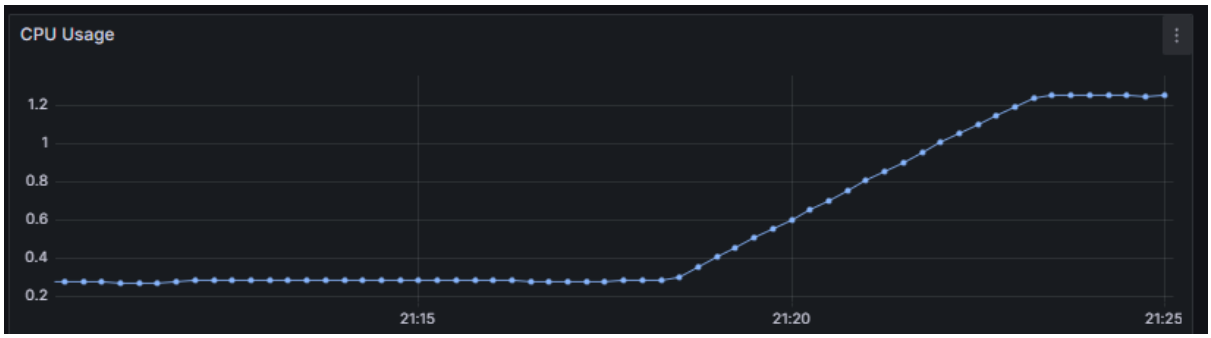


2.5.2 Results and Observations

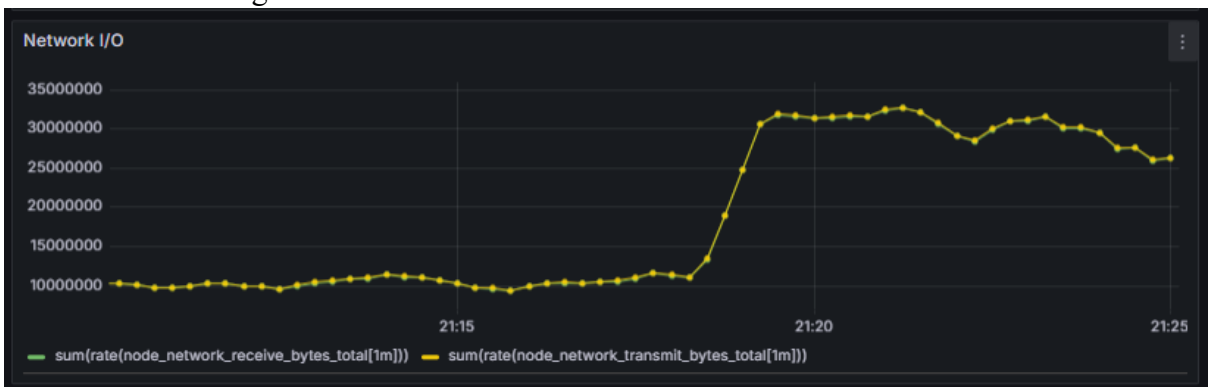
- **Node CPU Utilization**: Increased from 50% to 60%, indicating additional resource consumption during the test.



- **Pod CPU Utilization**: Increased from ~0.2 CPU shares/seconds to 1.2 CPU shares/seconds.

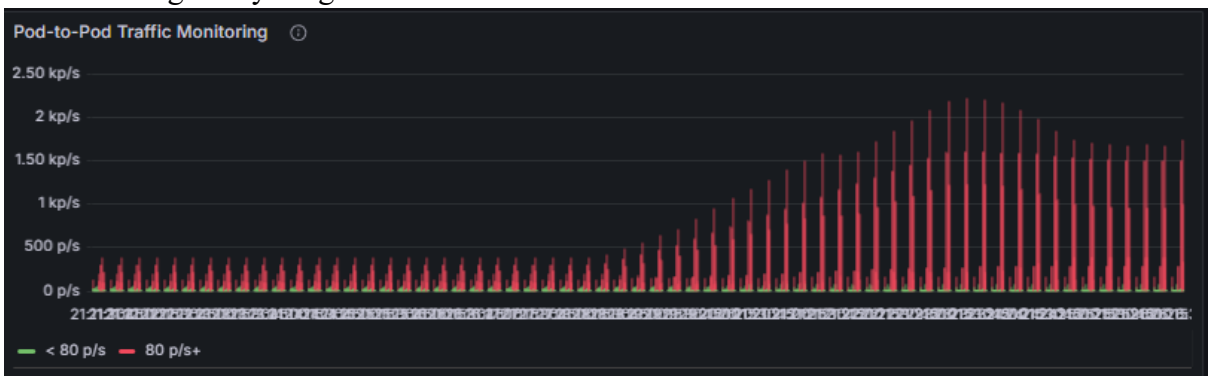


- **Network Traffic**: Network transmit rate saw a significant increase, reflecting higher data exchange due to the load test.

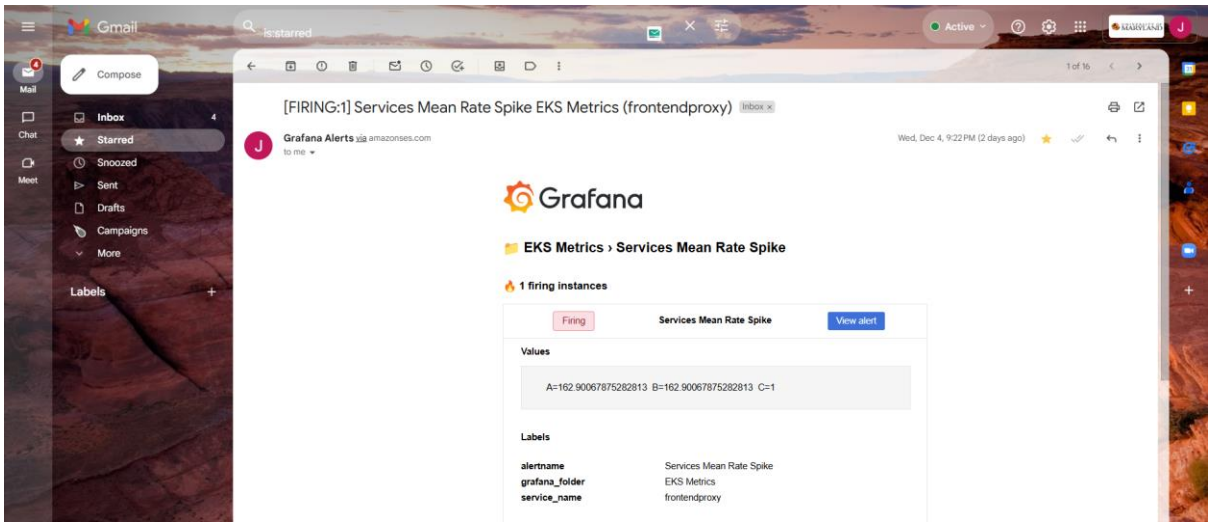


- **Service Latencies**: Latency was observed across various services, providing insight into how components respond to increased traffic:
    - Loadgen Service: Increased from 242ms to 4.31s.
    - Frontendweb Service: Increased from 979ms to 1.44s.
    - Frontendproxy Service: Increased from 67.6ms to 4.01s.
    - Frontend Service: Increased from 47.9ms to 477ms.

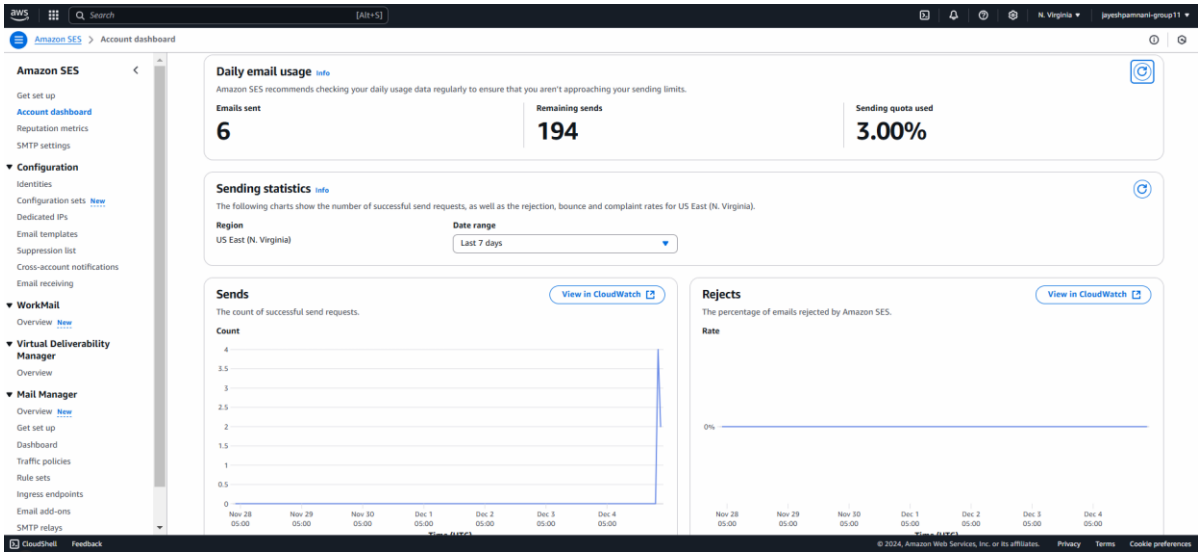
- **Pod-to-Pod Traffic Monitoring**: Observed a noticeable rise in pod-to-pod traffic, correlating with the increased load. This highlights inter-service communication surges during heavy usage scenarios.



The Request Rate Alert (>120 requests/sec), configured in Grafana, was triggered during the load test. This alert helped confirm that the system exceeded the defined threshold, highlighting the effectiveness of the alerting mechanism for monitoring scalability.



Email notifications for triggered alerts were successfully sent via Amazon SES, with the configuration tested and validated.



Challenges and Solutions

- **Initial errors in the scrape targets (no such host)**: Corrected service names to align with Kubernetes DNS conventions.
- **Misconfigured Prometheus exporter endpoint (too many colons error)**: Fixed the endpoint syntax by removing unnecessary prefixes.
- **Learning curve with PromQL to craft custom queries**: Used Grafana's query builder and referred to Prometheus documentation to simplify the process.
- **Grafana inaccessibility due to lack of a LoadBalancer**: Configured a LoadBalancer to enable external access to Grafana.
- **kube-state-metrics and node-exporter target failures**: Resolved by correcting target service names in the Prometheus configuration.
- **Incorrect Prometheus exporter endpoint causing otel-collector crash**: Fixed the endpoint syntax and validated logs to ensure proper configuration.
- **Validation error (assertNoLeakedSecrets) during deployment**: Temporarily disabled this check for testing while ensuring secure handling of sensitive information later.
- **SMTP connection issues with Amazon SES**: Troubleshot and validated SES credentials and email verification for reliable email delivery.
- **Missing network and storage metrics in dashboards**: Enhanced dashboards by creating custom queries for persistent storage and network traffic monitoring.
- **Email configuration errors with SMTP setup**: Resolved by carefully testing SES credentials and successfully verifying email alerts sent during load testing.

This phase achieved:

- A robust alerting system using Amazon SES to notify critical events.
- Comprehensive dashboards for cluster, node, pod, and service-level monitoring, with additional insights into network and storage performance.
- Validated dashboards under real-world load using JMeter, ensuring reliability and responsiveness during high traffic scenarios.