Observability

In the first part, we learned about observability and explored the concepts of metrics, logging, and tracing. We also discussed Prometheus and reviewed its architecture.

In today's session, we will cover how to install Prometheus and what PromQL (Prometheus Query Language) is.

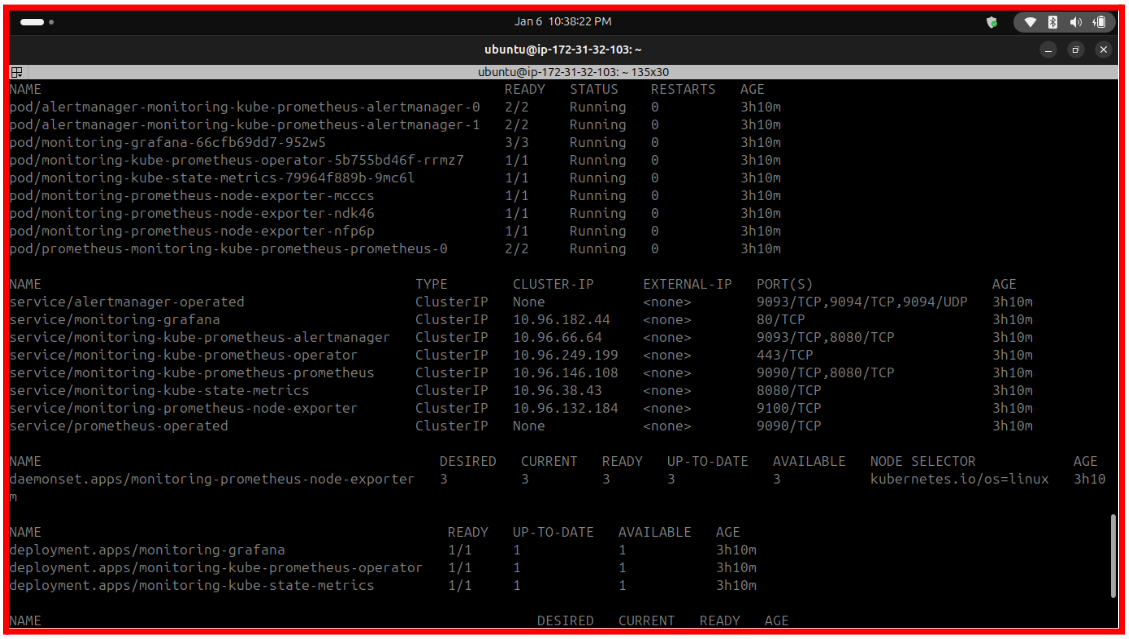
Let's start for installation

Step 1: Clone this repository: Observability Repository

Step 2: Run the following bash script to install Helm, Prometheus, and Grafana:

bash Prometheus-installation.shPrometheus and Grafana provide detailed information about your cluster, so make sure you have a running cluster (it can be any type, like Minikube or EKS, etc.).

Step 3 : Verify Installation by command Kubectl get all -n monitoring



```
NAME                                                 READY   STATUS    RESTARTS   AGE
pod/alertmanager-monitoring-kube-prometheus-alertmanager-0   2/2     Running   0          3h10m
pod/alertmanager-monitoring-kube-prometheus-alertmanager-1   2/2     Running   0          3h10m
pod/monitoring-grafana-66cfb69dd7-952w5                      3/3     Running   0          3h10m
pod/monitoring-kube-prometheus-operator-5b755bd46f-rrmz7     1/1     Running   0          3h10m
pod/monitoring-kube-state-metrics-79964f889b-9mc6l           1/1     Running   0          3h10m
pod/monitoring-prometheus-node-exporter-mcccs                1/1     Running   0          3h10m
pod/monitoring-prometheus-node-exporter-ndk46                1/1     Running   0          3h10m
pod/monitoring-prometheus-node-exporter-nfp6p                1/1     Running   0          3h10m
pod/prometheus-monitoring-kube-prometheus-prometheus-0       2/2     Running   0          3h10m

NAME                                             TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)                       AGE
service/alertmanager-operated                    ClusterIP   None            <none>        9093/TCP,9094/TCP,9094/UDP    3h10m
service/monitoring-grafana                       ClusterIP   10.96.182.44    <none>        80/TCP                        3h10m
service/monitoring-kube-prometheus-alertmanager  ClusterIP   10.96.66.64     <none>        9093/TCP,8080/TCP             3h10m
service/monitoring-kube-prometheus-operator      ClusterIP   10.96.249.199   <none>        443/TCP                       3h10m
service/monitoring-kube-prometheus-prometheus    ClusterIP   10.96.146.108   <none>        9090/TCP,8080/TCP             3h10m
service/monitoring-kube-state-metrics            ClusterIP   10.96.38.43     <none>        8080/TCP                      3h10m
service/monitoring-prometheus-node-exporter      ClusterIP   10.96.132.184   <none>        9100/TCP                      3h10m
service/prometheus-operated                      ClusterIP   None            <none>        9090/TCP                      3h10m

NAME                                                 DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR            AGE
daemonset.apps/monitoring-prometheus-node-exporter   3         3         3       3            3           kubernetes.io/os=linux   3h10
m

NAME                                                  READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/monitoring-grafana                    1/1     1            1           3h10m
deployment.apps/monitoring-kube-prometheus-operator   1/1     1            1           3h10m
deployment.apps/monitoring-kube-state-metrics         1/1     1            1           3h10m

NAME                                                             DESIRED   CURRENT   READY   AGE
```

To get started with live status or metrics, follow these steps:

Step 1: Clone this repository: Kubernetes-Manifest Repository

Step 2: Go to the "Nginx" folder and apply each manifest to start working with live status and metrics.

Step 3 : To access Prometheus and Grafana when working with any virtual machine, don't forget to add `--address 0.0.0.0` after the port-forward command.

Use Port forward command as for Prometheus UI

```
kubectl port-forward service/prometheus-operated -n monitoring 9090:9090
```

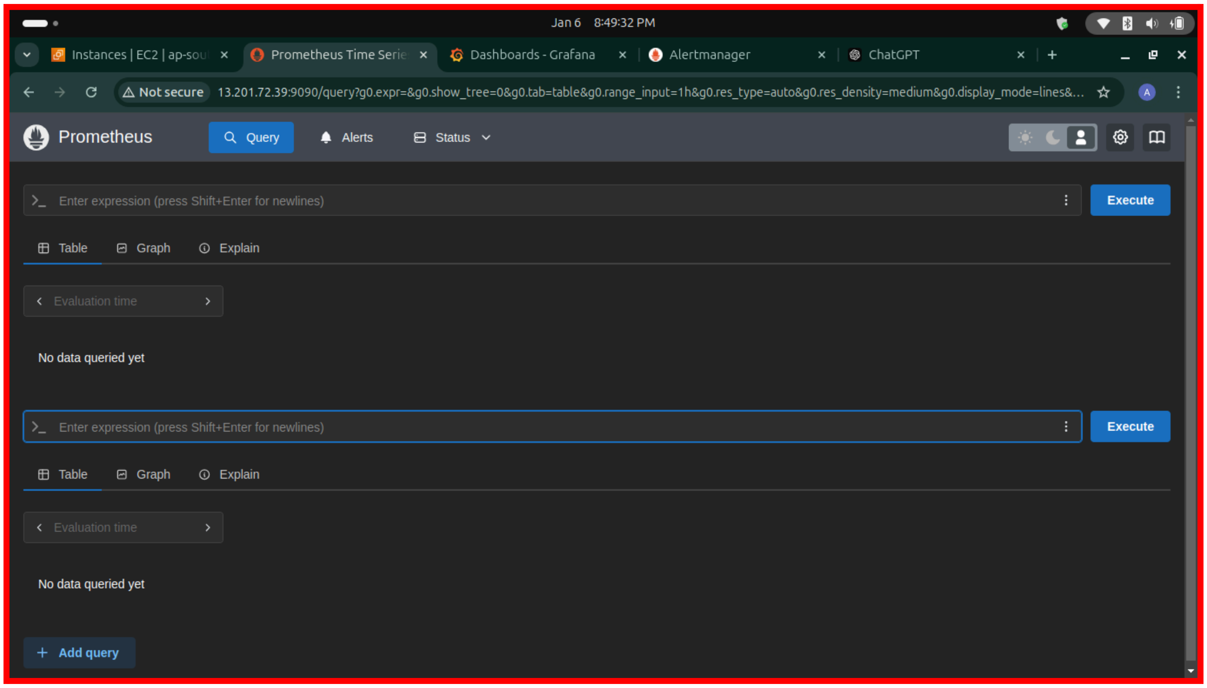
Use Port forward command for Grafana dashboard

```
kubectl port-forward service/monitoring-grafana -n monitoring 8080:80
```

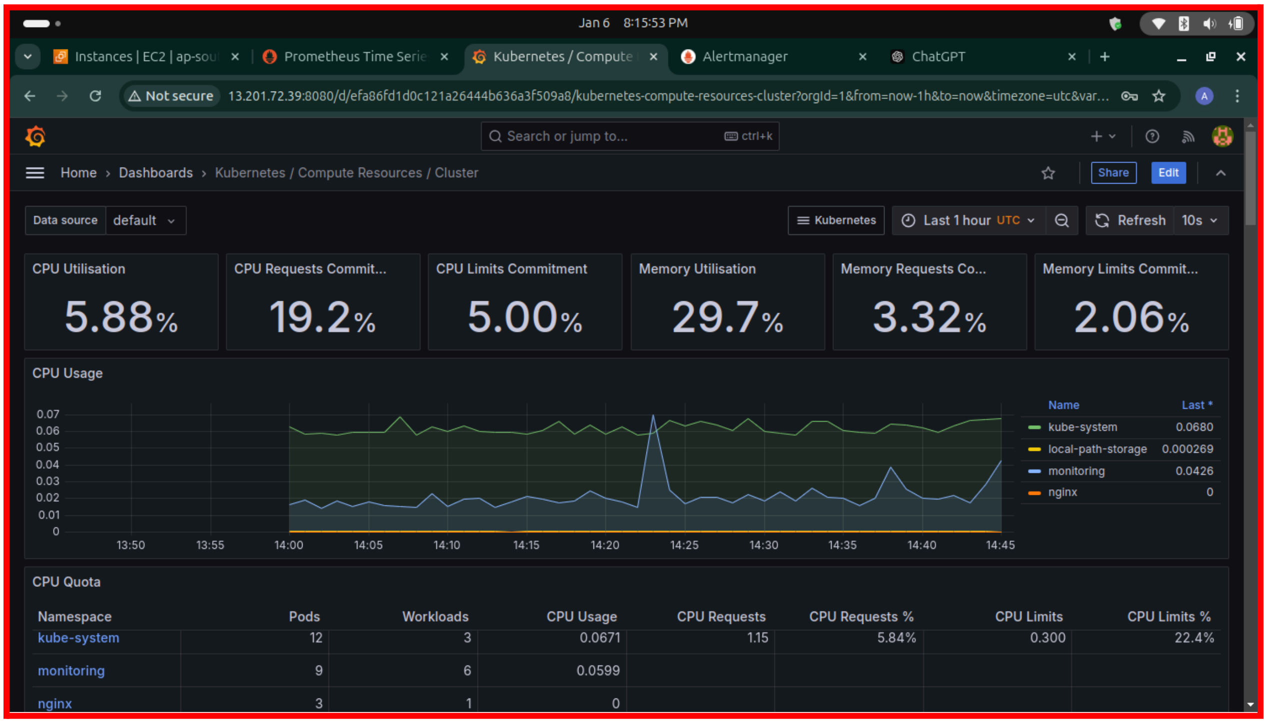
For Alert manager UI use

```
kubectl port-forward service/alertmanager-operated -n monitoring 9093:9093
```

Prometheus UI



Grafana Dashboard



13.201.72.39:8080/d/efa86fd1d0c121a26444b636a3f509a8/kubernetes-compute-resources-cluster?orgId=1&from=now-1h&to=now&timezone=utc&var...

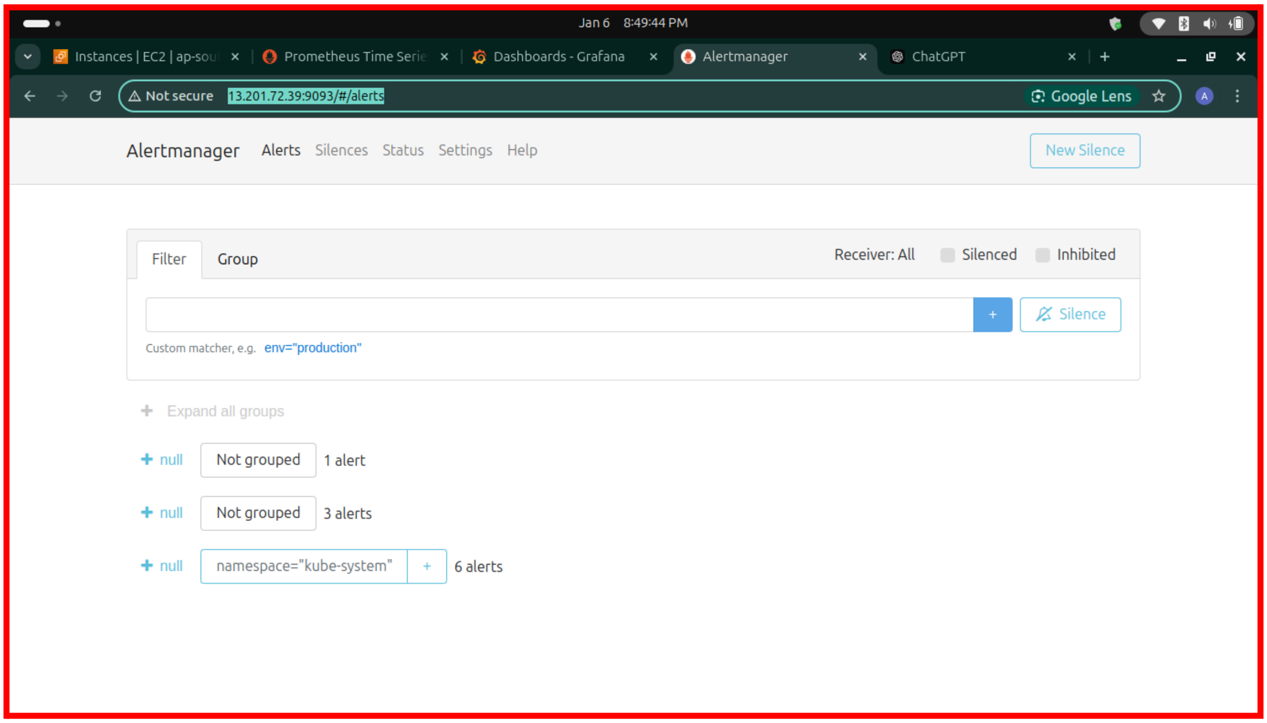
CPU Utilisation	CPU Requests Commit...	CPU Limits Commitment	Memory Utilisation	Memory Requests Co...	Memory Limits Commit...
5.88%	19.2%	5.00%	29.7%	3.32%	2.06%

Name	Last *
kube-system	0.0680
local-path-storage	0.000269
monitoring	0.0426
nginx	0

CPU Quota

Namespace	Pods	Workloads	CPU Usage	CPU Requests	CPU Requests %	CPU Limits	CPU Limits %
kube-system	12	3	0.0671	1.15	5.84%	0.300	22.4%
monitoring	9	6	0.0599				
nginx	3	1	0				

AlertManger UI



13.201.72.39:9093/#/alerts

Custom matcher, e.g. env="production"

+ null Not grouped 1 alert

+ null Not grouped 3 alerts

+ null namespace="kube-system" + 6 alerts

Author - Amit Ashok

Mobile - +9028490639

What is Exporter

Next, we need to understand how Prometheus collects data across the cluster and how this data is displayed in Grafana's dashboard format.

In Kubernetes (K8s), an exporter is a tool that gathers data (metrics) from a service, system, or application and converts it into a format that Prometheus can understand. It acts as a bridge, transforming raw data into structured metrics for Prometheus to monitor effectively.

While Prometheus has a basic UI for visualizing metrics, Grafana enhances this by providing interactive and customizable dashboards. Grafana integrates with Prometheus, fetching the collected metrics and displaying them in various dashboards in a user-friendly format.

Exporter Overview:

- Node Exporter: Provides node-related metrics, such as CPU, memory, disk usage, and other hardware-level metrics.
- Kube-state-metrics: Exposes Kubernetes cluster state metrics by interacting with the Kubernetes API server, providing insights into the state of resources like Deployments, Pods, and Nodes.
- Custom Exporter: Designed to gather custom metrics not provided by default. Developers set up a custom metric server and expose details through the /metrics endpoint, such as login events, sign-ups, HTTP requests, or application-specific data.

In basic terms the exporter collects data from various sources of cluster and provides them to prometheus.

Thank you for the day! We covered how to install Prometheus and Grafana on a Kubernetes cluster and also understood how Prometheus scrapes metrics from the cluster.