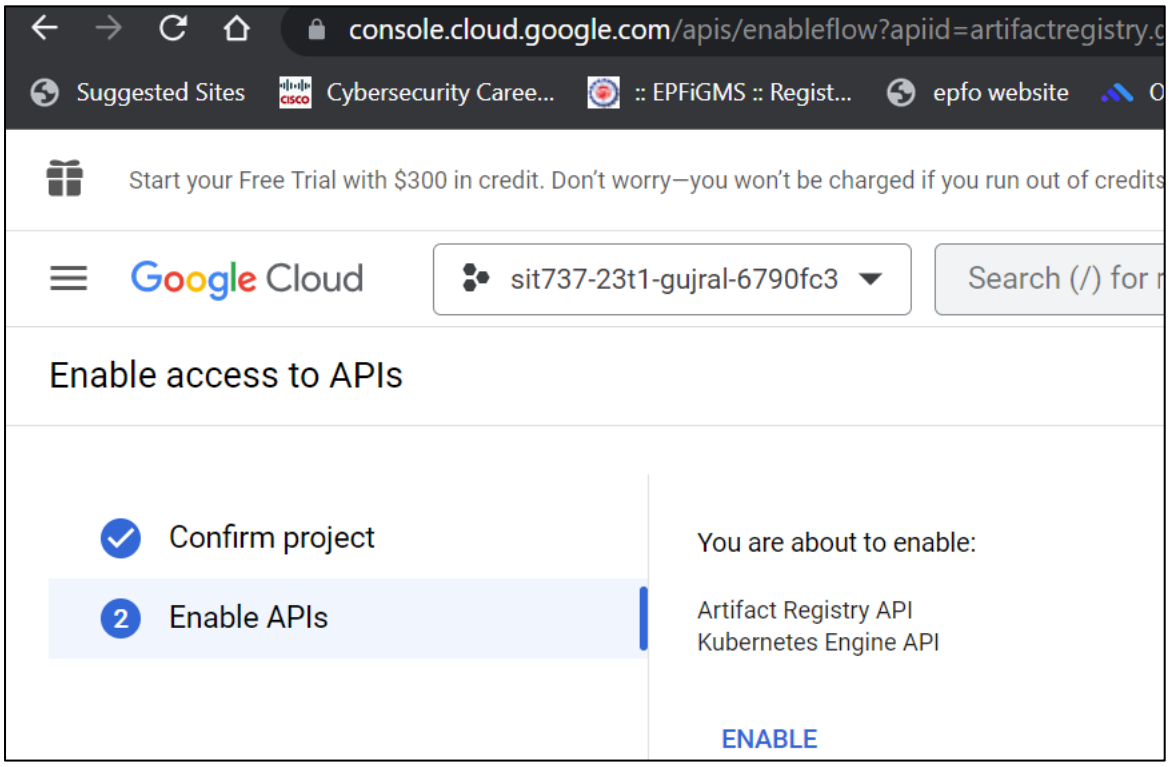
10.1P: MONITORING AND VISIBILITY

Instructions

- Containerize a simple application using Node.js, Docker, and Kubernetes.
- Deploy the containerized application to a GCP Kubernetes cluster.
- Ensure proper monitoring and visibility of your cloud-native application.

Solution

1. Logged in to GCP and navigated under the project sit737-23t1-gujral-6790fc3.
2. Enabled Artifact Registry and Google Kubernetes Engine APIs.



3. Created a GKE cluster

The cluster includes many worker nodes which are VM instances, 3 in this case as seen in this screenshot below. Applications are deployed on clusters, and run on nodes.

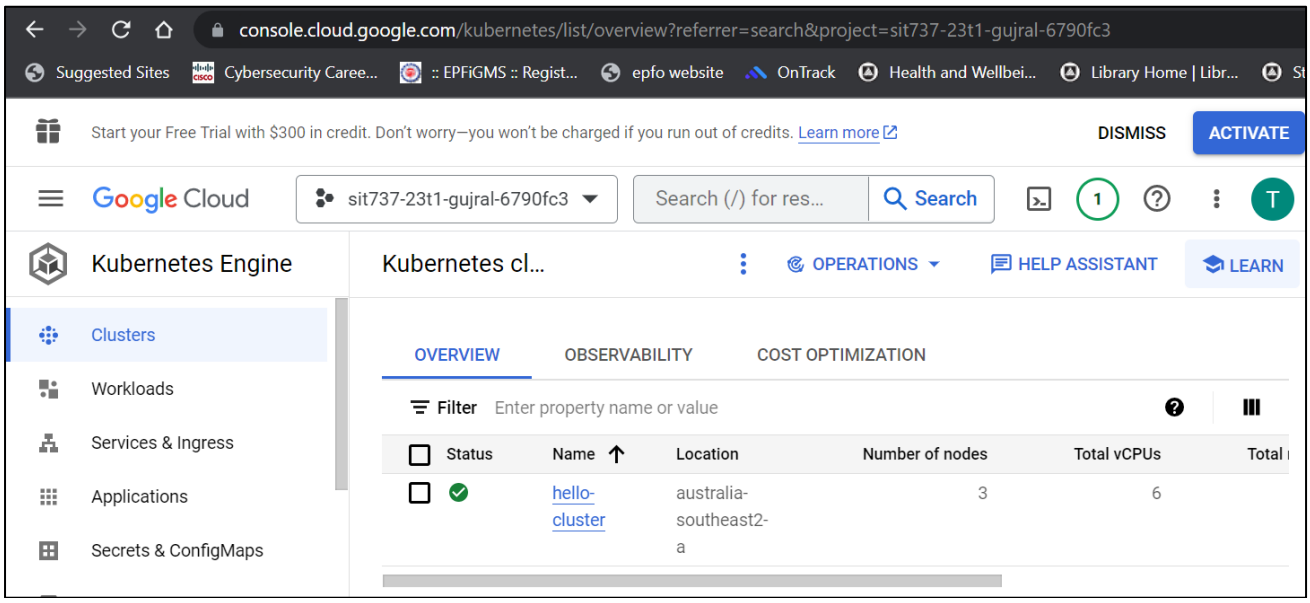
(i) Created a standard cluster named hello-cluster, using command →

```
gcloud container clusters create hello-cluster \
  --region=COMPUTE_REGION
```

here, compute_region for Melbourne could be chosen as australia-southeast2-a, australia-southeast2-b, australia-southeast2-c. The one used below is australia-southeast2-a

```
s222205644@cloudshell:~ (sit737-23t1-gujral-6790fc3)$ gcloud container clusters create hello-cluster     --region=australia-southeast2-a
Default change: VPC-native is the default mode during cluster creation for versions greater than 1.21.0-gke.1500. To create advanced routes based clusters, plea
se pass the `--no-enable-ip-alias` flag
Default change: During creation of nodepools or autoscaling configuration changes for cluster versions greater than 1.24.1-gke.800 a default location policy is
applied. For Spot and PVM it defaults to ANY, and for all other VM kinds a BALANCED policy is used. To change the default values use the `--location-policy` fla
g.
Note: Your Pod address range (`--cluster-ipv4-cidr`) can accommodate at most 1008 node(s).
```

```
g.
Note: Your Pod address range (`--cluster-ipv4-cidr`) can accommodate at most 1008 node(s).
Creating cluster hello-cluster in australia-southeast2-a... Cluster is being deployed...working
Creating cluster hello-cluster in australia-southeast2-a... Cluster is being deployed...working...
Creating cluster hello-cluster in australia-southeast2-a... Cluster is being deployed...working..
Creating cluster hello-cluster in australia-southeast2-a... Cluster is being deployed...working...
Creating cluster hello-cluster in australia-southeast2-a... Cluster is being deployed...working
Creating cluster hello-cluster in australia-southeast2-a... Cluster is being deployed...working.
Creating cluster hello-cluster in australia-southeast2-a... Cluster is being health-checked (master is healthy)...working...
Creating cluster hello-cluster in australia-southeast2-a... Cluster is being health-checked (master is healthy)...done.
Created [https://container.googleapis.com/v1/projects/sit737-23t1-gujral-6790fc3/zones/australia-southeast2-a/clusters/hello-cluster].
To inspect the contents of your cluster, go to: https://console.cloud.google.com/kubernetes/workload_/gcloud/australia-southeast2-a/hello-cluster?project=sit737
-23t1-gujral-6790fc3
kubeconfig entry generated for hello-cluster.
NAME: hello-cluster
LOCATION: australia-southeast2-a
MASTER_VERSION: 1.25.8-gke.500
MASTER_IP: 34.126.202.87
MACHINE_TYPE: e2-medium
NODE_VERSION: 1.25.8-gke.500
NUM_NODES: 3
STATUS: RUNNING
s222205644@cloudshell:~ (sit737-23t1-gujral-6790fc3)$
```



(ii) Got authentication credentials for the cluster

Authentication credentials allow for communication interact with the cluster.

Command →

gcloud container clusters get-credentials hello-cluster \
  --region australia-southeast2-a

```
s222205644@cloudshell:~ (sit737-23t1-gujral-6790fc3)$ gcloud container clusters get-credentials hello-cluster \
    --region australia-southeast2-a
Fetching cluster endpoint and auth data.
kubeconfig entry generated for hello-cluster.
s222205644@cloudshell:~ (sit737-23t1-gujral-6790fc3)$
```

4. Deployed application to the cluster

i. Node.js application used is a basic one that prints Welcome to microservices, similar to a Hello World application

Index.js -

```
const { json } = require('express');
const express = require('express');
const app = express();

//Constants
const PORT = 3000;
const HOST = '0.0.0.0';

// Home page request
app.get('/', (req, res) => {
  res.send("Welcome to the microservice");
});

app.listen(PORT, HOST, () => {
  console.log('Running on http://${HOST}:${PORT}');
});
```

Dockerfile –

```
#denotes base image
FROM node:14

#setting working directory
WORKDIR /usr/src/app

COPY package*.json ./

#to install the package listed in package.json file
RUN npm install

COPY index.js index.js

#exposing port outside
EXPOSE 3000
CMD ["node", "index.js"]
```

(iii) Built the image using Docker build command → docker build -t welcome_image .

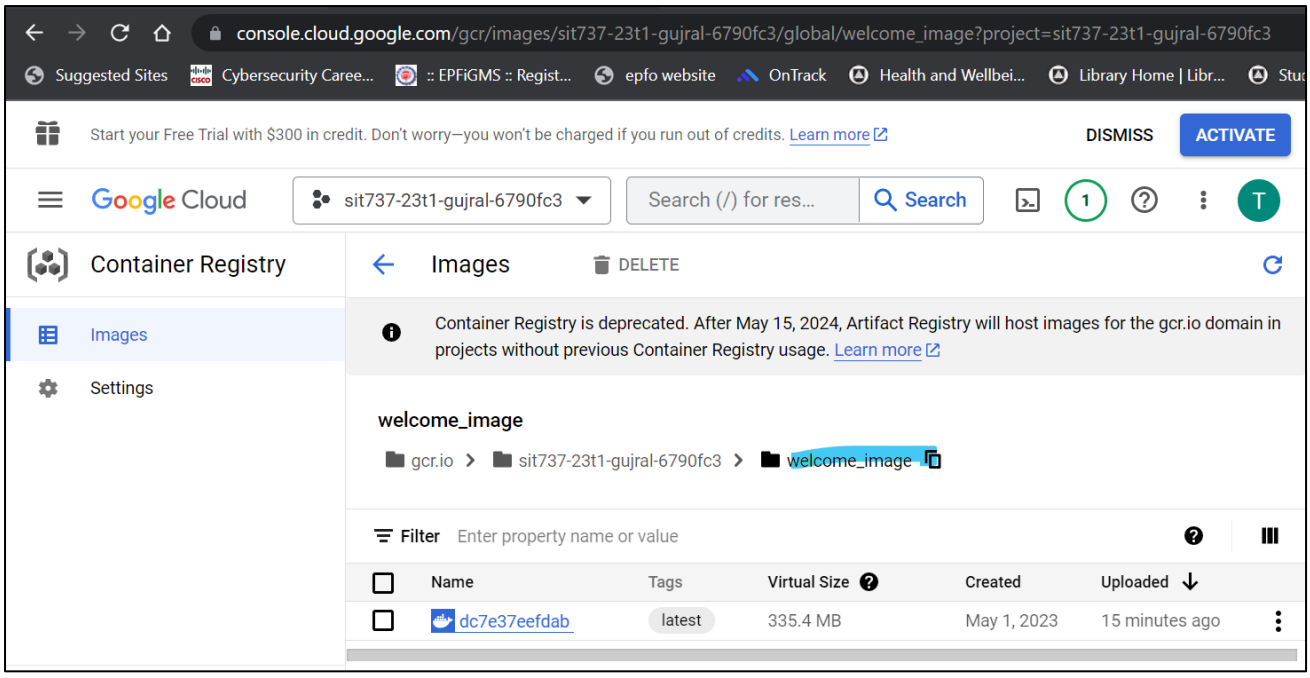
```
PS C:\Tanya\DEAKIN\T1 2023\SIT737 Cloud Native Application Development\tasks\7.1P\7.1Prepo\7.1P> docker build -t welcome_image .
[+] Building 73.2s (11/11) FINISHED
 => [internal] load build definition from Dockerfile                                                    0.1s
```

```
PS C:\Tanya\DEAKIN\T1 2023\SIT737 Cloud Native Application Development\tasks\7.1P\7.1Prepo\7.1P> docker images | select-string  welcome_ima
ge

welcome_image                                     latest
1025b7cb5ca6   5 minutes ago   917MB
```

(iv) Pushed the docker image to GCR

Command used → docker push gcr.io/sit737-23t1-gujral-6790fc3/welcome_image



5. Created the Deployment

Used the command below to create a deployment named hello-server. The pod of the deployment runs the welcome_image container image.

Command →

```
kubectl create deployment hello-server \
  --image=gcr.io/sit737-23t1-gujral-6790fc3/welcome_image
```

```
s222205644@cloudshell:~ (sit737-23t1-gujral-6790fc3)$ kubectl create deployment hello-server \
    --image=gcr.io/sit737-23t1-gujral-6790fc3/welcome_image
deployment.apps/hello-server created
```

6. Exposed the Deployment

This is to expose the deployed application outside the cluster, to the internet for any user to be able to access it. It uses a service to expose the hello-server to external traffic.

Command →

kubectl expose deployment hello-server --type LoadBalancer --port 80 --target-port 3000

The --type LoadBalancer flag creates a Compute Engine load balancer for the container. The port 3000 of the application is connected to port 80 i.e., the internet here.

```
s222205644@cloudshell:~ (sit737-23t1-gujral-6790fc3)$ kubectl expose deployment hello-server --type LoadBalancer --port 80 --target-port 3000
service/hello-server exposed
```

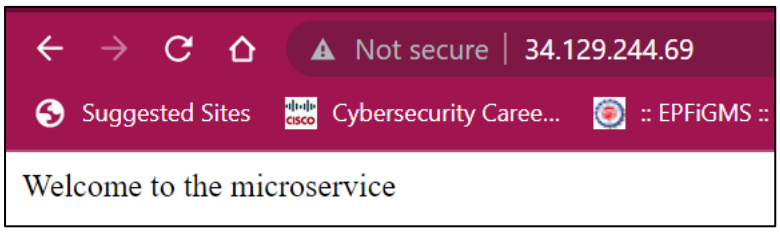
i. Inspecting and viewing the application

kubectl get pods

kubectl get service hello-server

```
s222205644@cloudshell:~ (sit737-23t1-gujral-6790fc3)$ kubectl get pods
NAME                           READY   STATUS    RESTARTS   AGE
hello-server-66b744ddcb-ghdlv  1/1     Running   0          12m
s222205644@cloudshell:~ (sit737-23t1-gujral-6790fc3)$ kubectl get service
NAME           TYPE           CLUSTER-IP     EXTERNAL-IP     PORT(S)        AGE
hello-server   LoadBalancer   10.28.14.122   34.129.244.69   80:32524/TCP   2m22s
kubernetes     ClusterIP      10.28.0.1      <none>          443/TCP        46m
```

ii. Accessing the application from web browser by using the external IP address -



7. Monitoring

Metrics Scope in GCP provides a wide-range of dashboards showcasing the metrics related to the cloud assets and provides with a detailed real-time monitoring.

Observing logs and metrics help sin not only troubleshooting and maintenance but also provides analytics. The most common use of GCP's monitoring dashboards is to monitor the availability, health of APIs, performance, scalability etc.

- **Custom dashboard** "10.1P Dashboard" link - https://console.cloud.google.com/monitoring/dashboards/builder/7e539051-fe0d-45b0-87e7-b73894538a32;duration=PT1H

Created a custom dashboard adding 6 of the metrics below –

(i) Kubernetes container – CPU limit utilization

(ii) VM Instance – CPU utilization

(iii) Kubernetes Node – CPU Usage Time

(iv) Kubernetes Pod – Bytes Received

(v) Kubernetes Container – Memory limit utilization

(vi) Log entries MEAN

The JSON file for the custom dashboard “10.1P Dashboard” is uploaded in the Github repository for the project

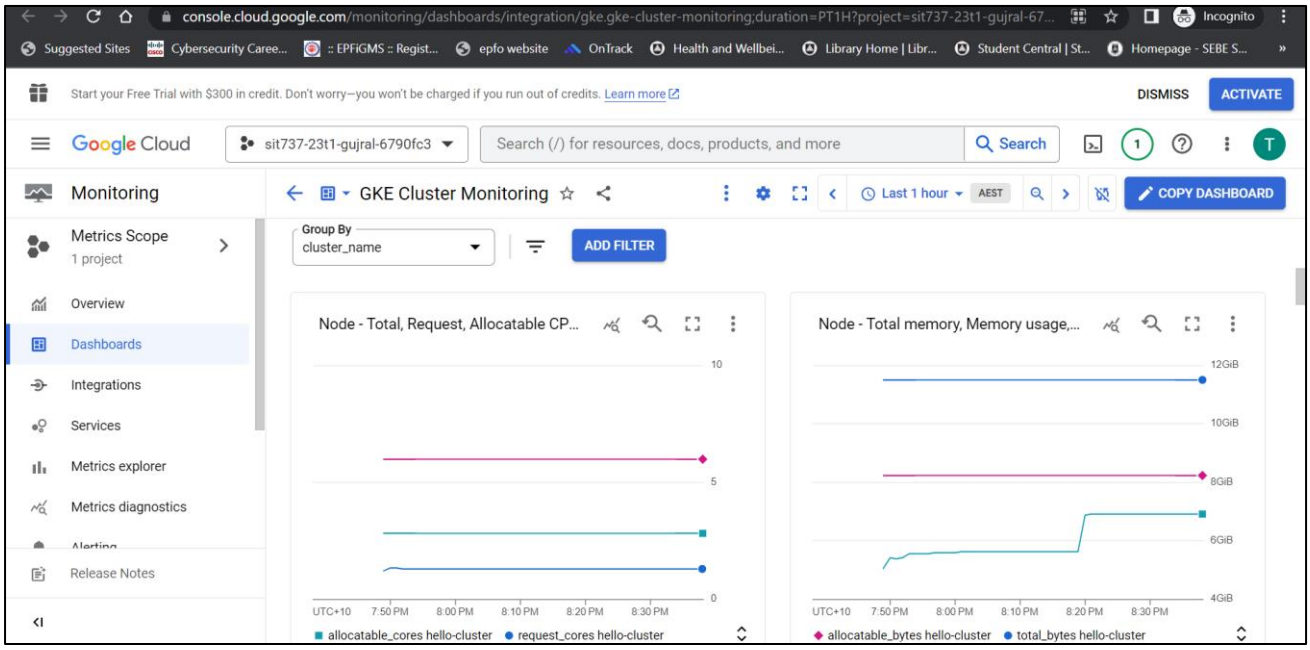
- There are a variety of **default dashboards** covering the metrics exhaustively -

The link to the default dashboards is - https://console.cloud.google.com/monitoring/dashboards?project=sit737-23t1-gujral-6790fc3&pageState=(%22dashboards%22:(%22t%22:%22All%22))

A few of the dashboards are captured and showcased below –

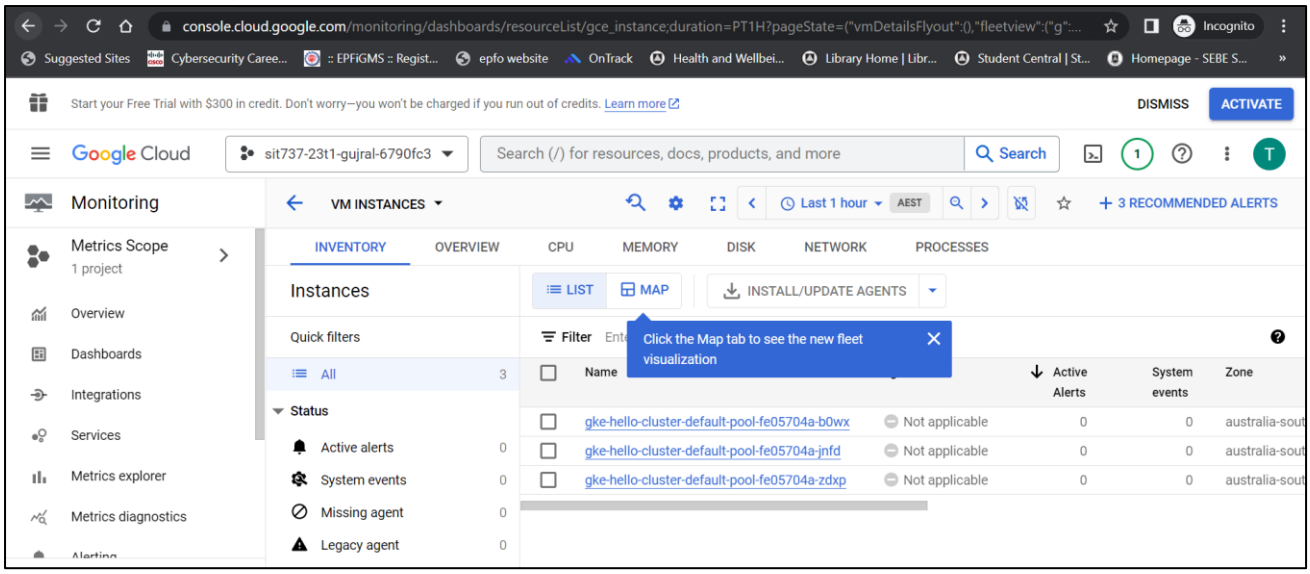
(i) GKE cluster monitoring

https://console.cloud.google.com/monitoring/dashboards/integration/gke.gke-cluster-monitoring?project=sit737-23t1-gujral-6790fc3

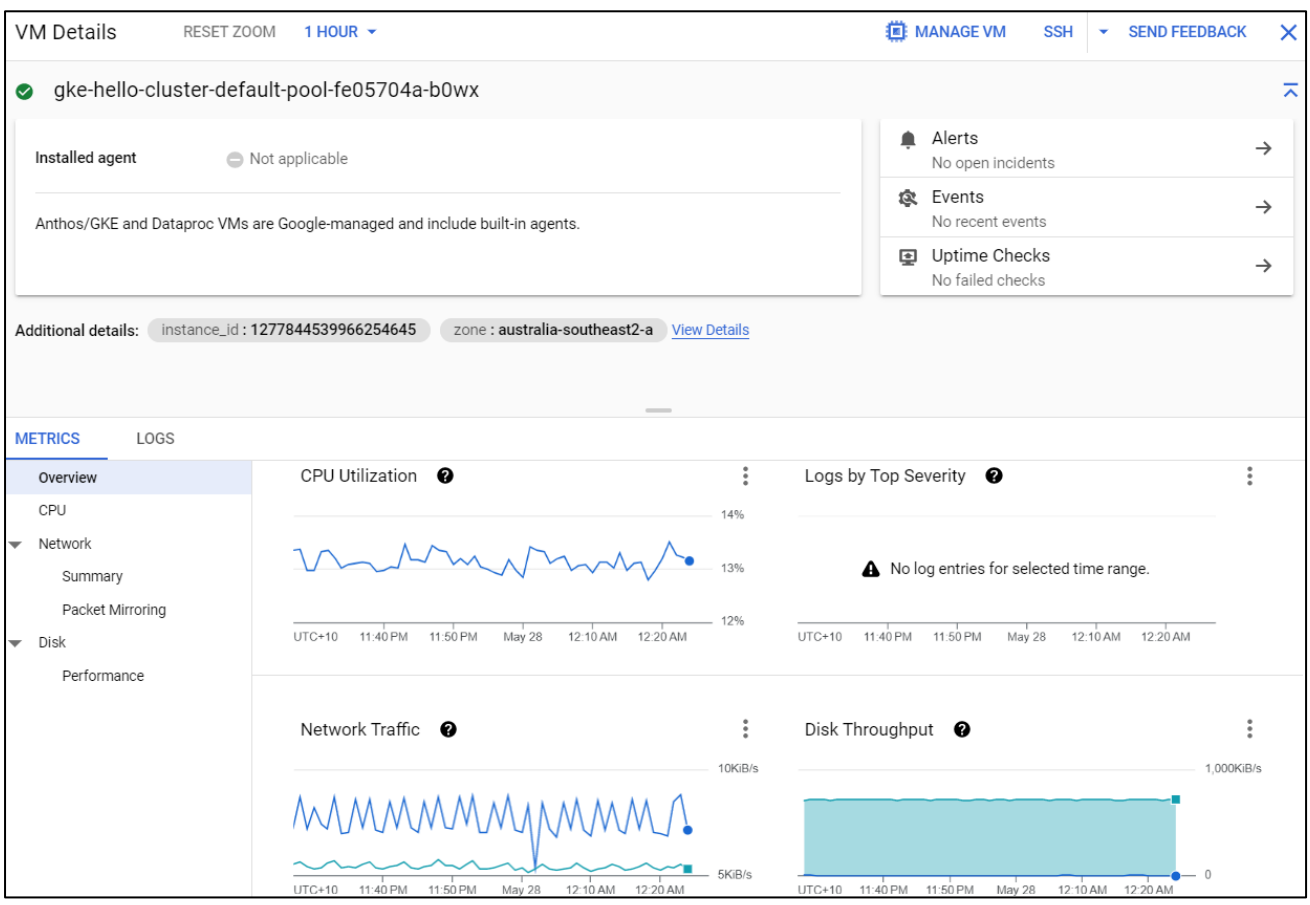


(ii) VM instances

https://console.cloud.google.com/monitoring/dashboards/resourceList/gce_instance;duration=PT1H?pageState=(%22vmDetailsFlyout%22:(),%22fleetview%22:(%22g%22:%5B%22resource.labels.zone%22,%22metadata.systemLabels.name%22%5D,%22d%22:%5B%5D,%22s%22:%22COUNT%22,%22m%22:%22compute.googleapis.com%2Finstance%2Fcpu%2Futilization%22))&project=sit737-23t1-gujral-6790fc3

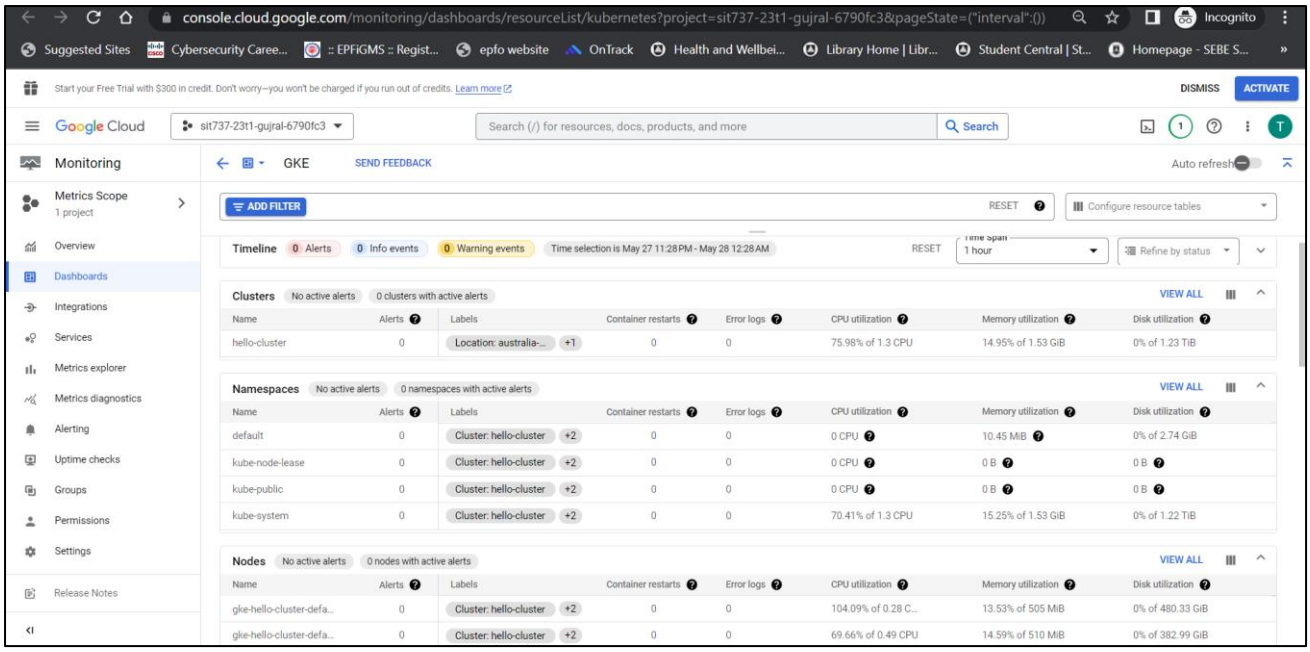


Further, for each VM metrics can be seen as –



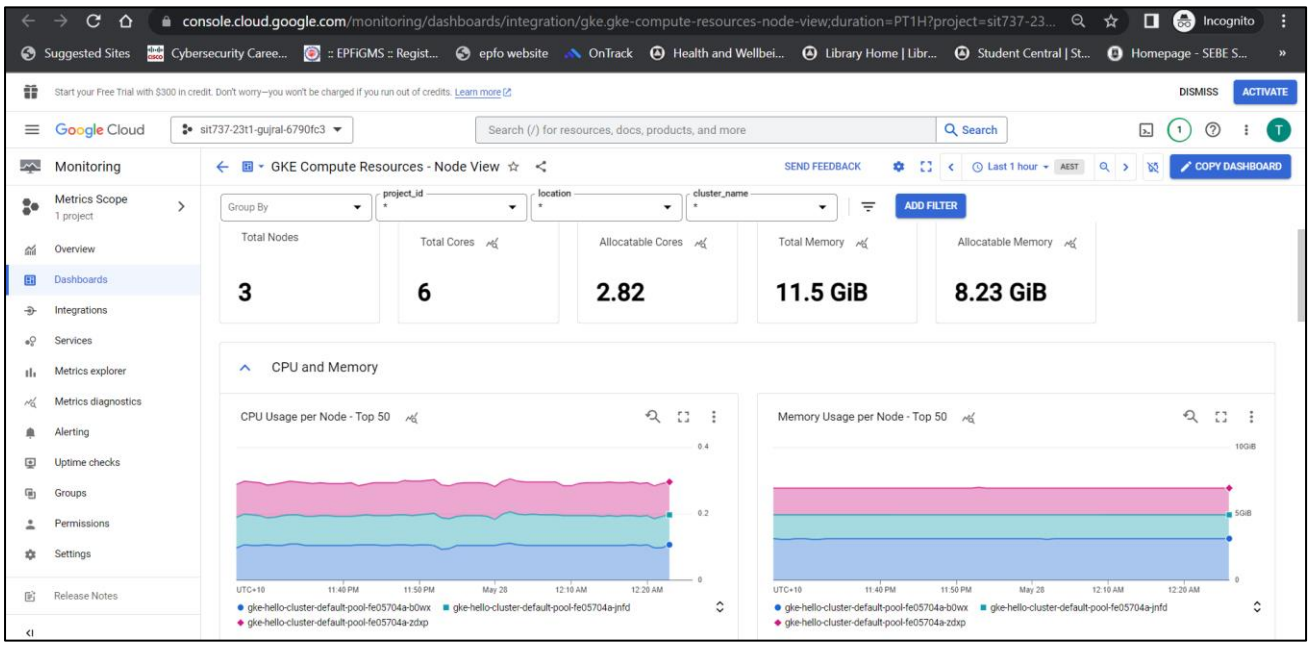
(iii) GKE metrics –

https://console.cloud.google.com/monitoring/dashboards/resourceList/kubernetes?project=sit737-23t1-gujral-6790fc3&pageState=(%22interval%22:())



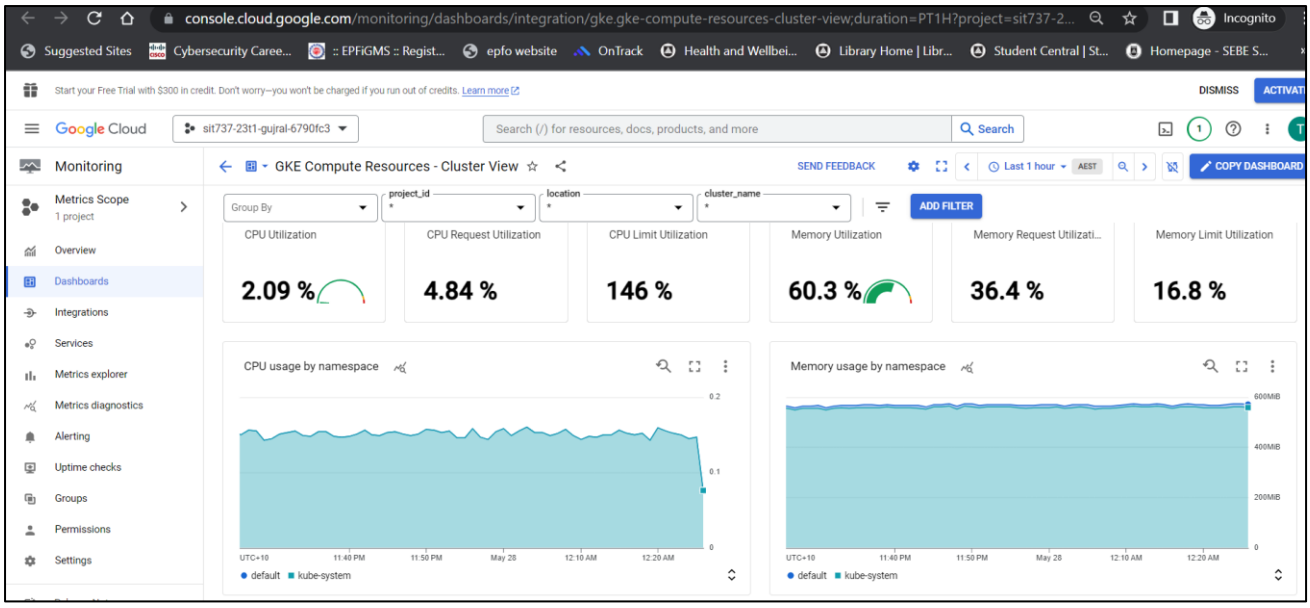
(iv) GKE Compute Resources (Node View) –

https://console.cloud.google.com/monitoring/dashboards/integration/gke.gke-compute-resources-node-view?project=sit737-23t1-gujral-6790fc3



(v) GKE Compute Resources (Cluster View) –

https://console.cloud.google.com/monitoring/dashboards/integration/gke.gke-compute-resources-cluster-view;duration=PT1H?project=sit737-23t1-gujral-6790fc3


console.cloud.google.com/monitoring/dashboards/integration/gke.gke-compute-resources-cluster-view;duration=PT1H?project=sit737-2...
Incognito
Suggested Sites
Cybersecurity Caree...
:: EPFiGMS :: Regist...
epfo website
OnTrack
Health and Wellbei...
Library Home | Libr...
Student Central | St...
Homepage - SEBE S...
Start your Free Trial with $300 in credit. Don't worry—you won't be charged if you run out of credits. Learn more
DISMISS
ACTIVAT
Google Cloud
sit737-23t1-gujral-6790fc3
Search (/) for resources, docs, products, and more
Search
Monitoring
GKE Compute Resources - Cluster View
SEND FEEDBACK
Last 1 hour
AEST
COPY DASHBOARD
Metrics Scope
1 project
Group By
project_id
location
cluster_name
ADD FILTER
Overview
Dashboards
Integrations
Services
Metrics explorer
Metrics diagnostics
Alerting
Uptime checks
Groups
Permissions
Settings
CPU Utilization
2.09 %
CPU Request Utilization
4.84 %
CPU Limit Utilization
146 %
Memory Utilization
60.3 %
Memory Request Utilizati...
36.4 %
Memory Limit Utilization
16.8 %
CPU usage by namespace
0.2
0.1
0
UTC+10
11:40 PM
11:50 PM
May 28
12:10 AM
12:20 AM
default
kube-system
Memory usage by namespace
600MiB
400MiB
200MiB
0
UTC+10
11:40 PM
11:50 PM
May 28
12:10 AM
12:20 AM
default
kube-system