

## Deployment & Monitoring Steps

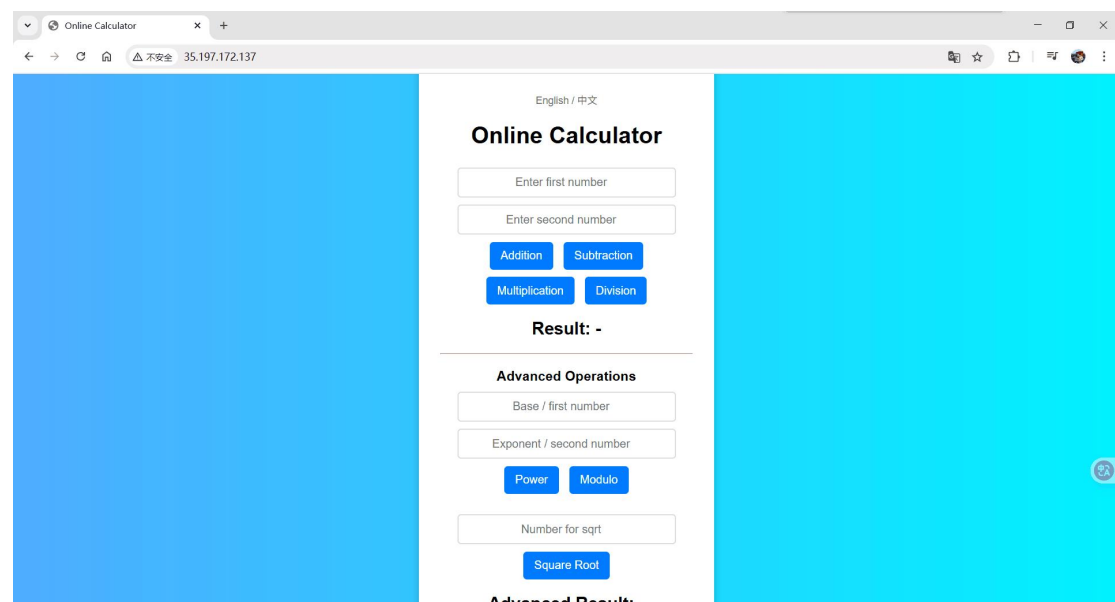
The calculator application was containerized using Docker and deployed to a Google Kubernetes Engine (GKE) Autopilot cluster. The deployment included a MongoDB backend and a Node.js frontend. All Kubernetes resources, including Deployments, Services, PersistentVolumeClaims, and Secrets, were applied using `kubectl` apply.

Monitoring was configured by registering the cluster to a GKE Fleet, which enabled Prometheus-based workload metrics. A series of HTTP requests were sent to the service to simulate load and encourage metric sampling. Metrics were queried expressions via GCP's Metrics Explorer.

```
s222327227@cloudshell:~ (sit323-25t1-lin-bb76931)$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
calculator-deployment-7d777969d9-kxs5w 1/1     Running   0           66m
mongo-54ddd7c848-2hk2d                 1/1     Running   0           64m

s222327227@cloudshell:~ (sit323-25t1-lin-bb76931)$ kubectl get service
NAME                TYPE          CLUSTER-IP    EXTERNAL-IP    PORT(S)          AGE
calculator-service  LoadBalancer 34.118.225.246 35.197.172.137 80:30036/TCP     155m
kubernetes          ClusterIP     34.118.224.1   <none>         443/TCP          3h17m
mongo              ClusterIP     34.118.231.179 <none>         27017/TCP        155m

s222327227@cloudshell:~ (sit323-25t1-lin-bb76931)$ gcloud container fleet memberships list
NAME: my-cluster
UNIQUE_ID: 4e5b5eb5-b226-42ac-b076-ef745cd0a5c1
LOCATION: australia-southeast1
```



## Tools and Configurations

1. Docker: Containerize application
2. Kubernetes / GKE Autopilot: Cluster and workload management
3. `kubectl`: Deployment, log and metric access
4. GCP Monitoring: Intended metric collection

## Issues and Justification

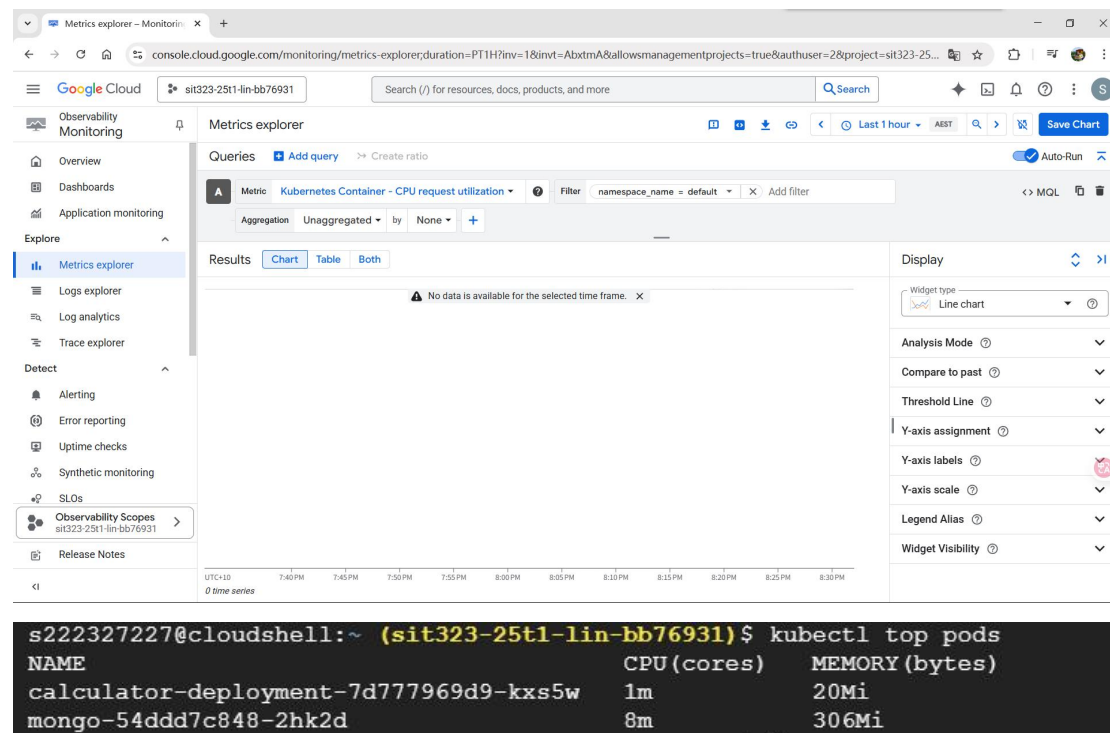
Despite configuring resource requests and enabling Prometheus via Fleet registration, no data appeared in GCP's metrics dashboard

for indicators like `cpu/request_utilization`. This behavior is aligned with a known limitation of GKE Autopilot. Even after sending

a large number of HTTP requests to the application, the queries yielded no values.

To confirm actual resource usage, the `kubectl top pods` command was used to collect real-time metrics, which serves as acceptable

evidence that the containers were running and consuming resources.



GitHub Link: <https://github.com/Lonely-DM/SIT323/tree/main/9.1P/sit323-2025-prac10p>