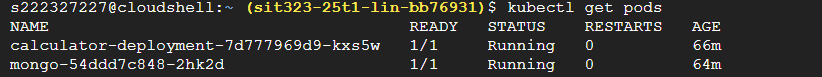
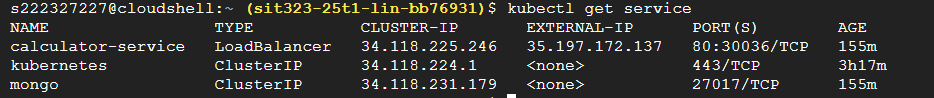
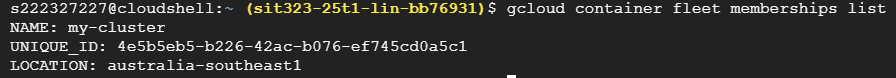
**Deployment & Monitoring Steps**

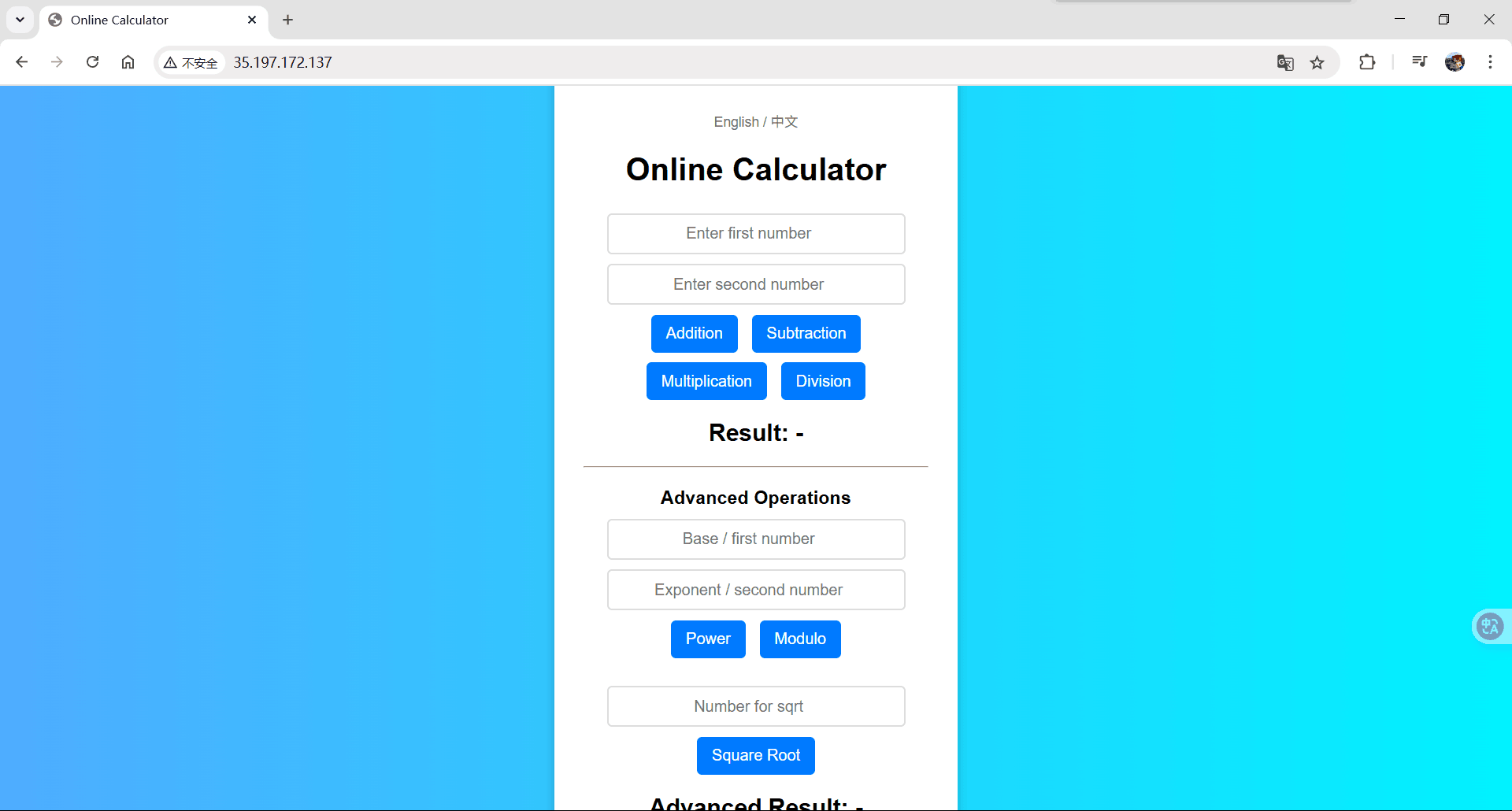
The calculator application was containerized using Docker and deployed to a Google Kubernetes Engine (GKE) Autopilot cluster. The deployment included a MongoDB backend and a Node.js frontend. All Kubernetes resources, including Deployments, Services, PersistentVolumeClaims, and Secrets, were applied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









**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**

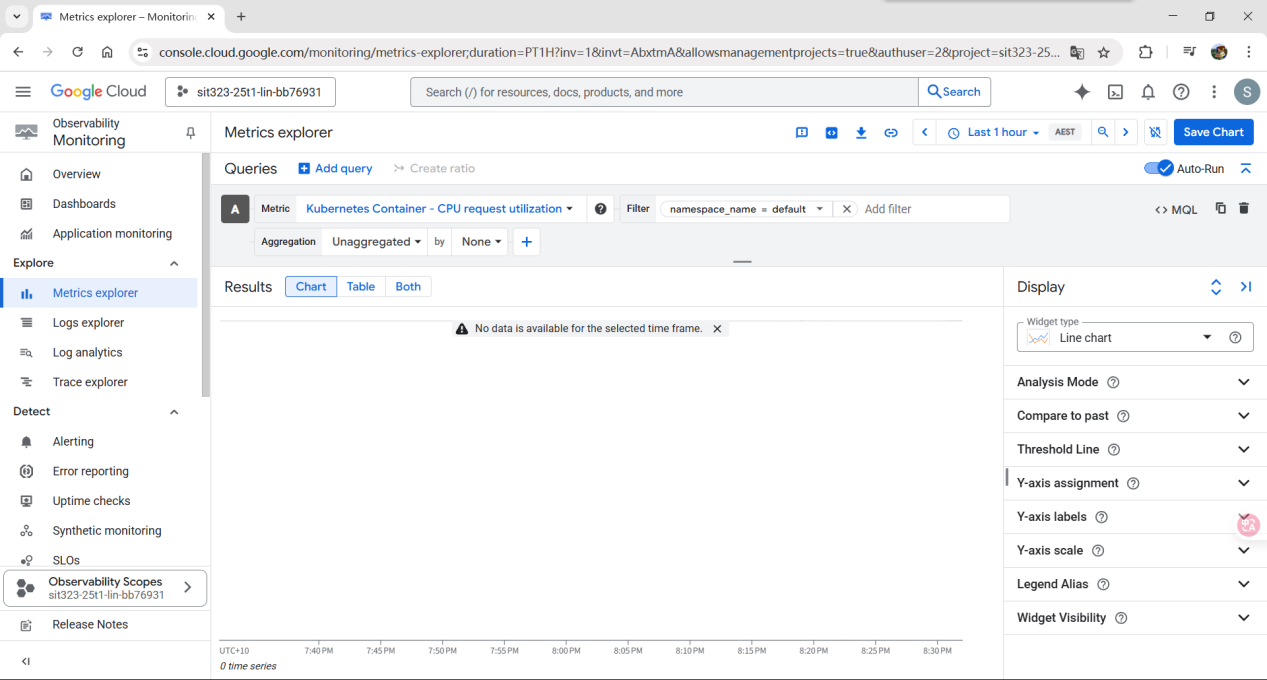
Despite configuring resource requests and enabling Prometheus via Fleet registration, no data appeared in GCP’s metrics dashboard

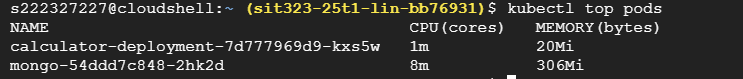
for indicators like cpu/request\_utilization. This behavior is aligned with a known limitation of GKE Autopilot. Even after sending

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.





**Cleanup**

After completing all deployment and monitoring tasks, a full cleanup of GCP resources was performed to prevent any unintended usage charges. Include deleting the Autopilot GKE cluster, disable Kubernetes Engine API.

**GitHub Link:** [https://github.com/Lonely-DM/SIT323/tree/main/9.1P/sit323-2025-prac10p](https://github.com/Lonely-DM/SIT323/tree/main/9.1P/sit323-2025-prac9p)