**GCP Project**

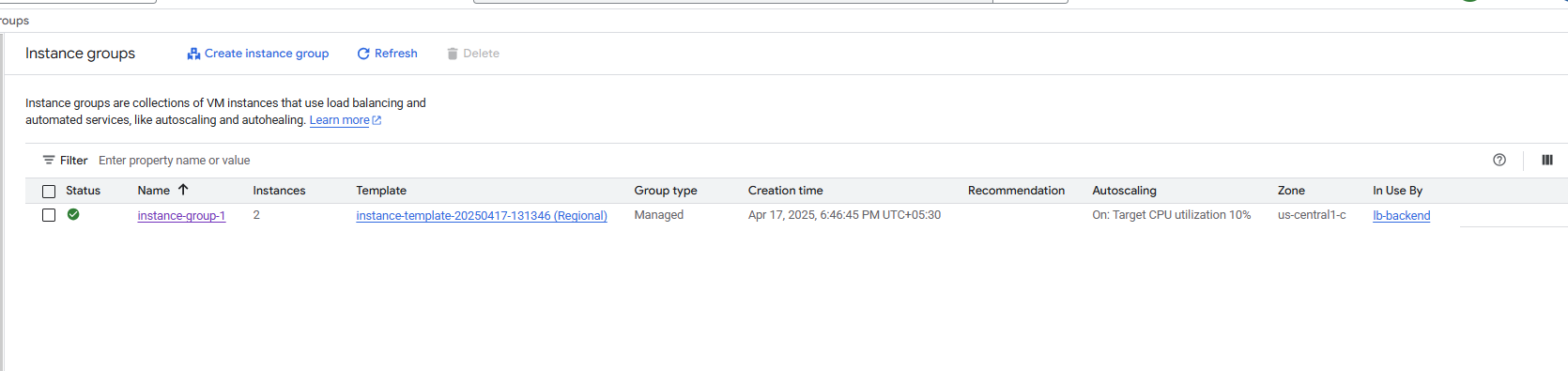
**If you are using ubuntu**

**apt-get update**

**apt-get install apache2 -y**

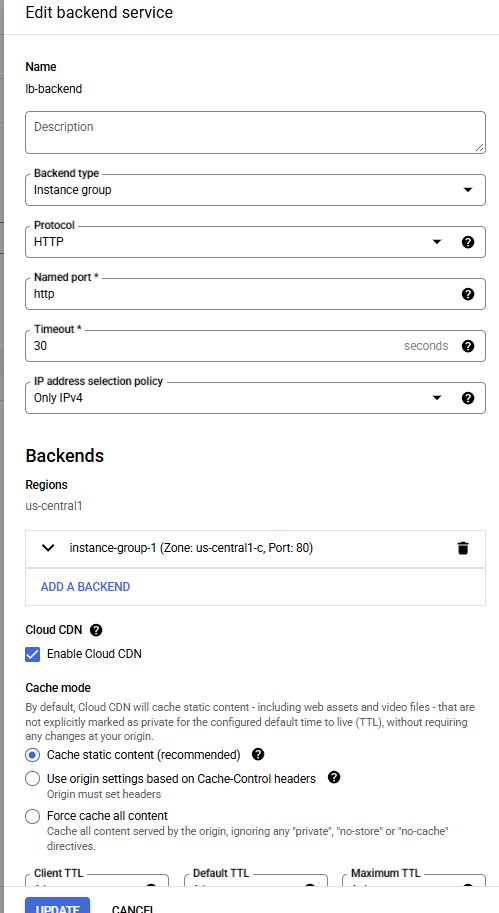
**Task -2 Deploy Compute Engine for Web Servers**

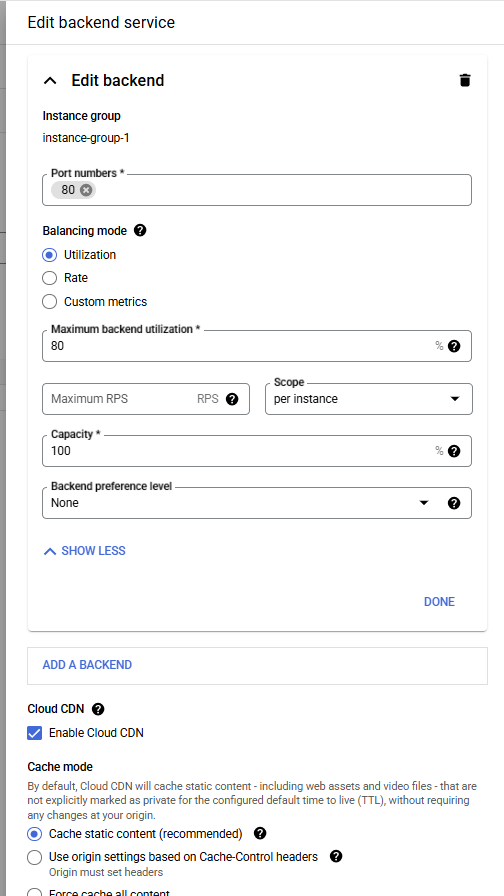
**Set autoscaling 10%**

****

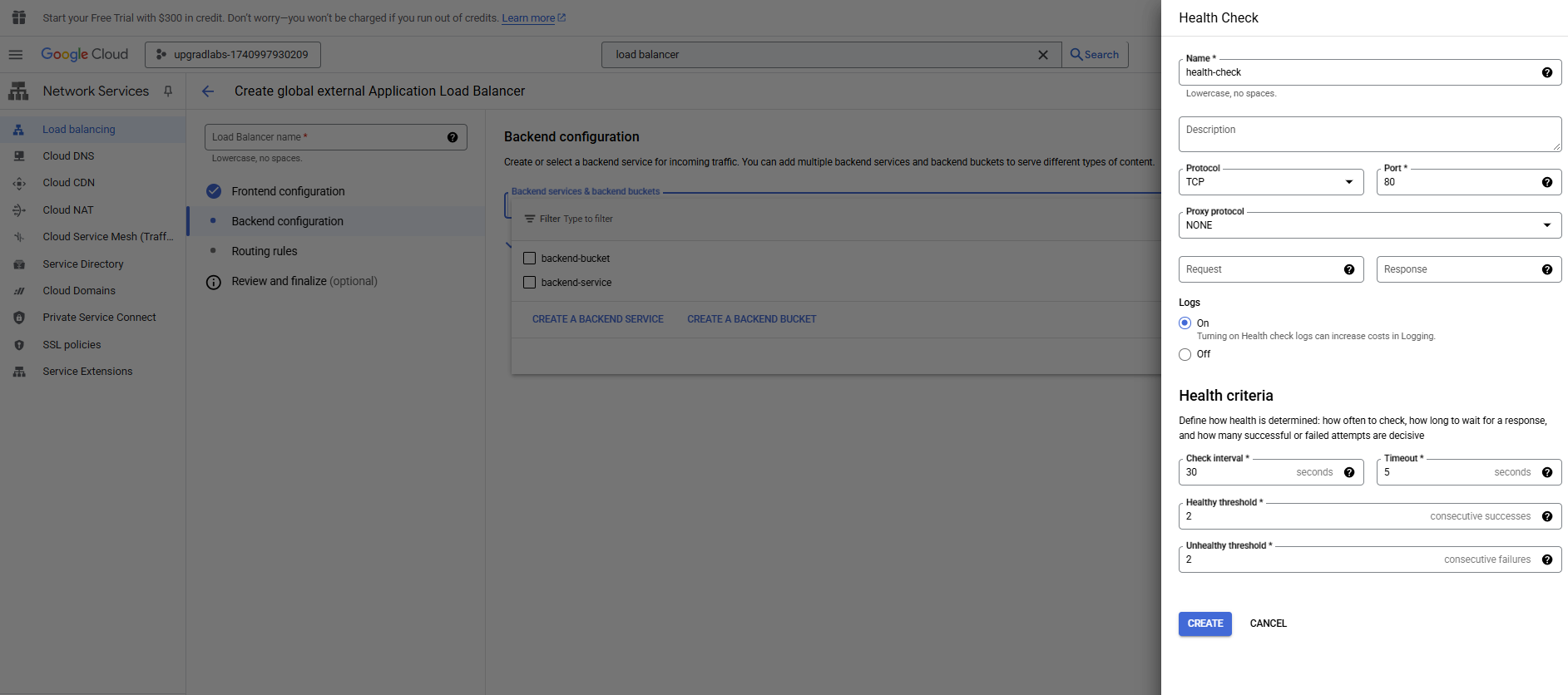
**Vm instance with autoscaling**

**Configure lb backend**

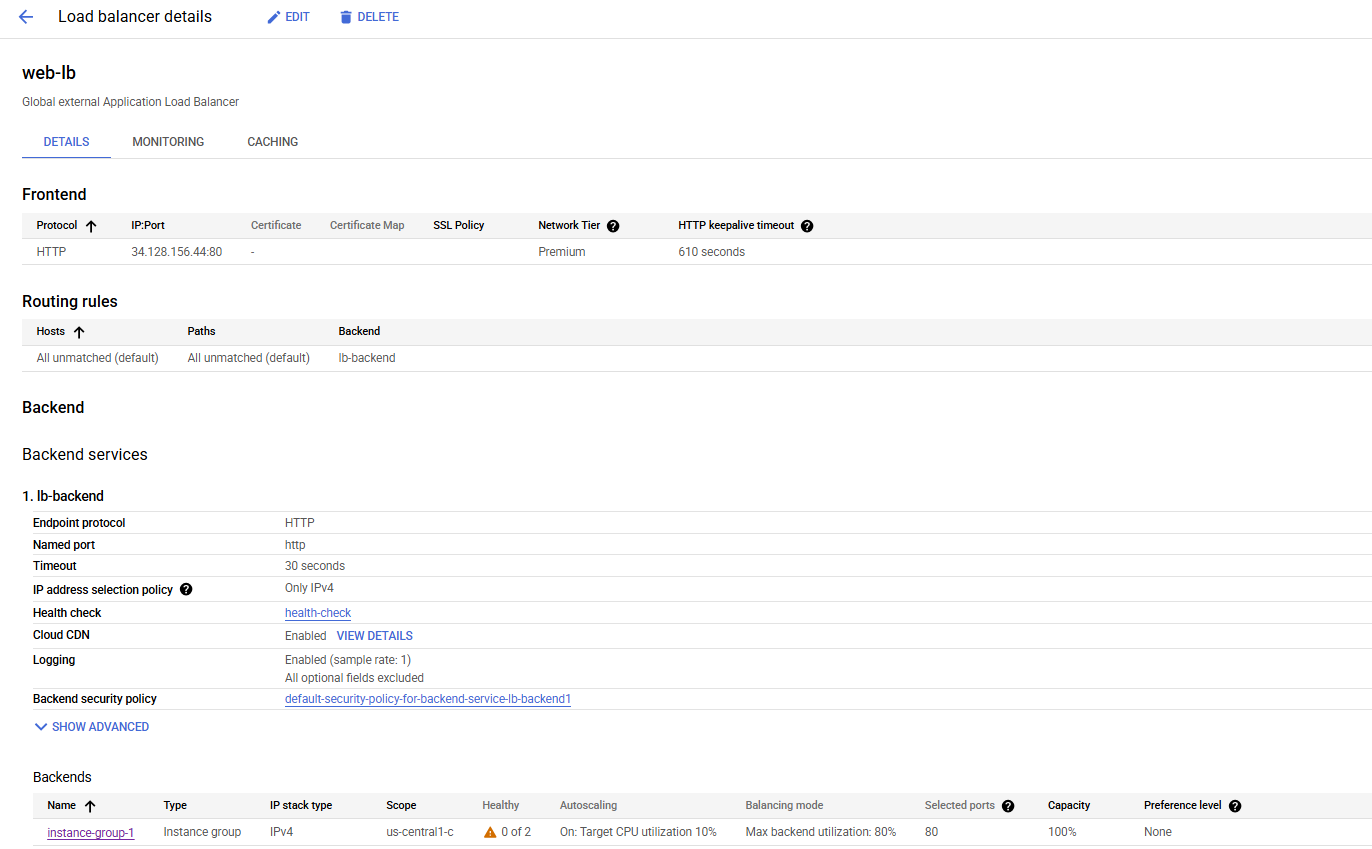




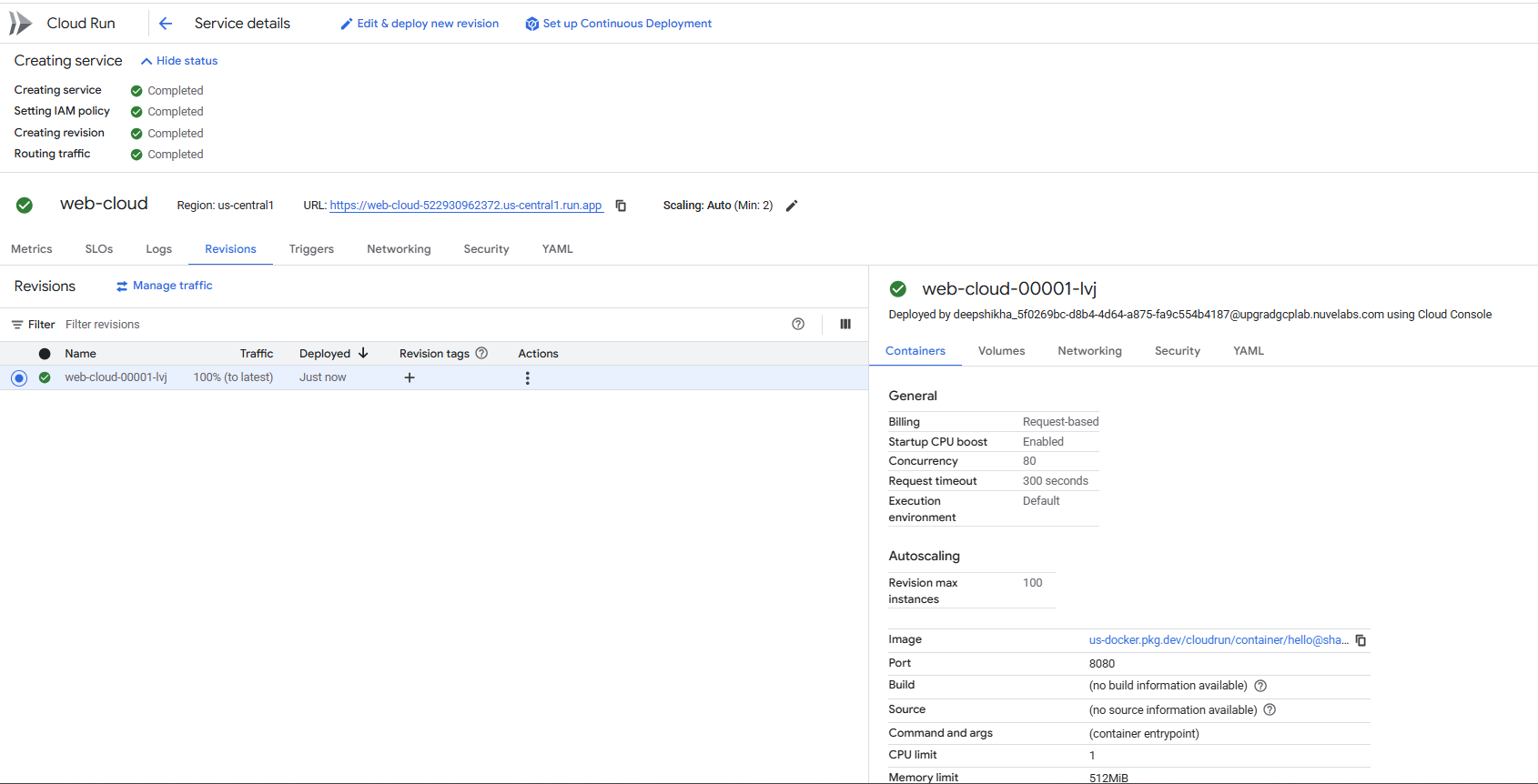
**Creating Health check in load balancer**

****

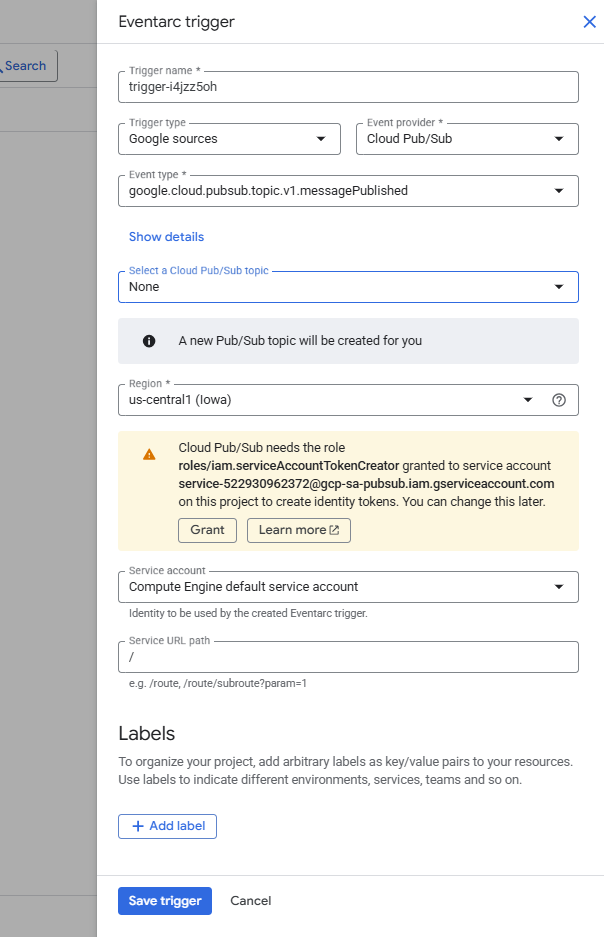
Load balancer

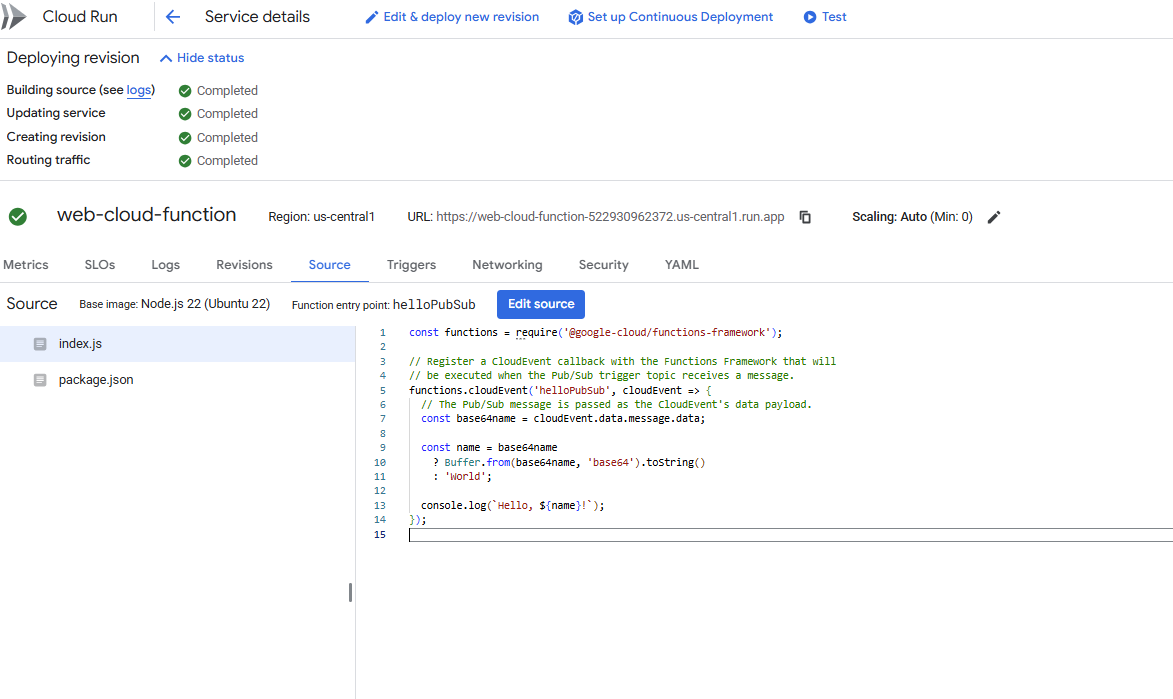


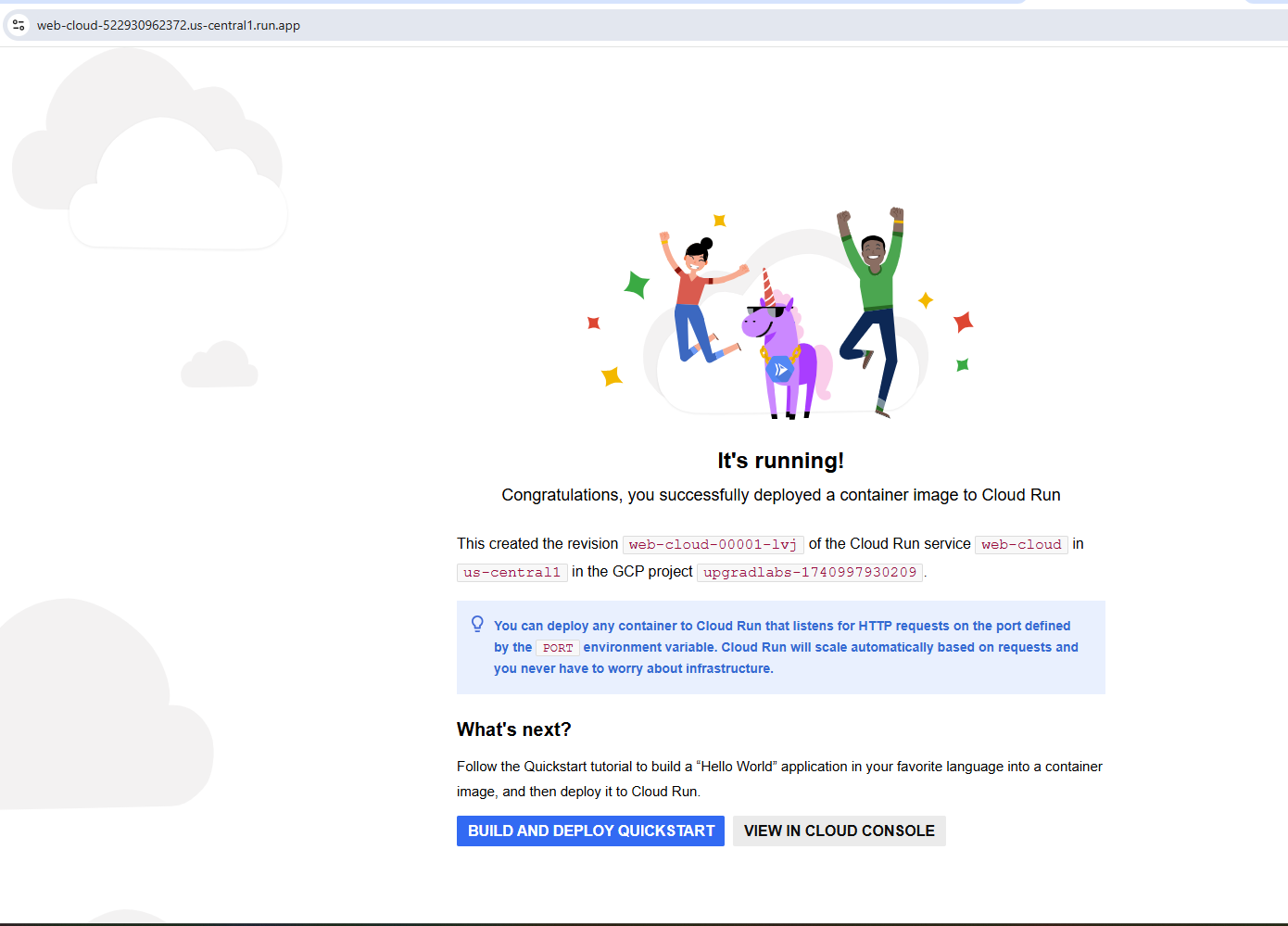
**Deploy Serverless Functions (Cloud Run & Cloud Functions)**



**Create Function, choose a trigger (HTTP, Pub/Sub, etc.), and deploy**

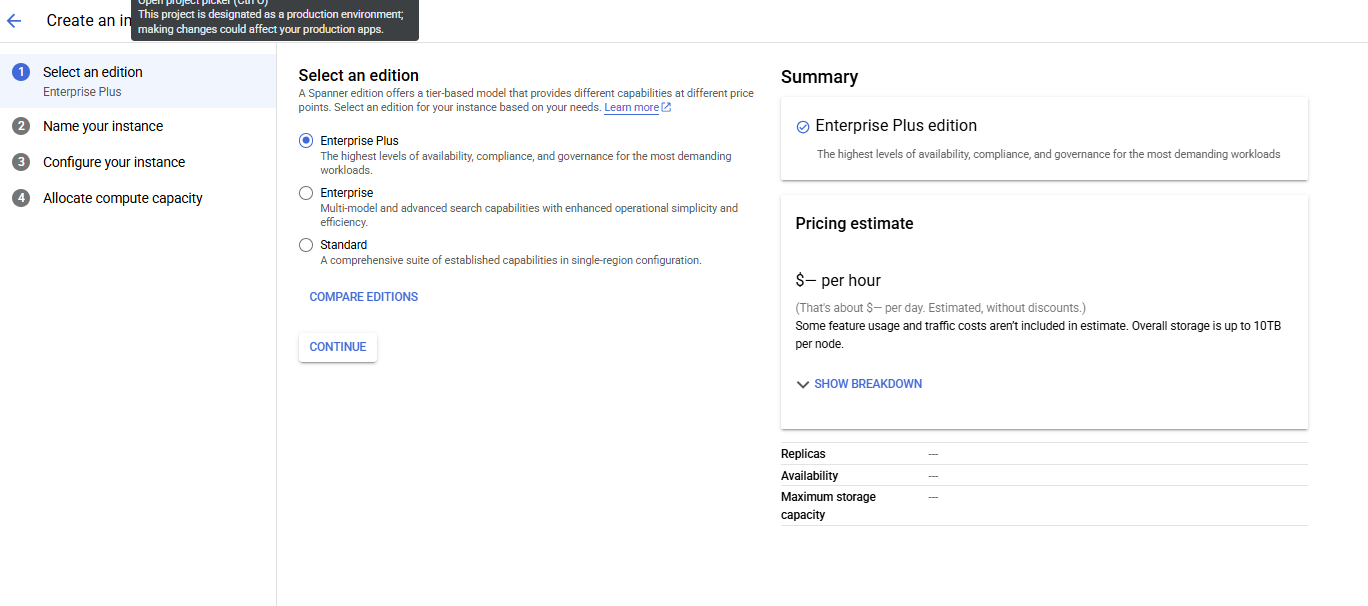


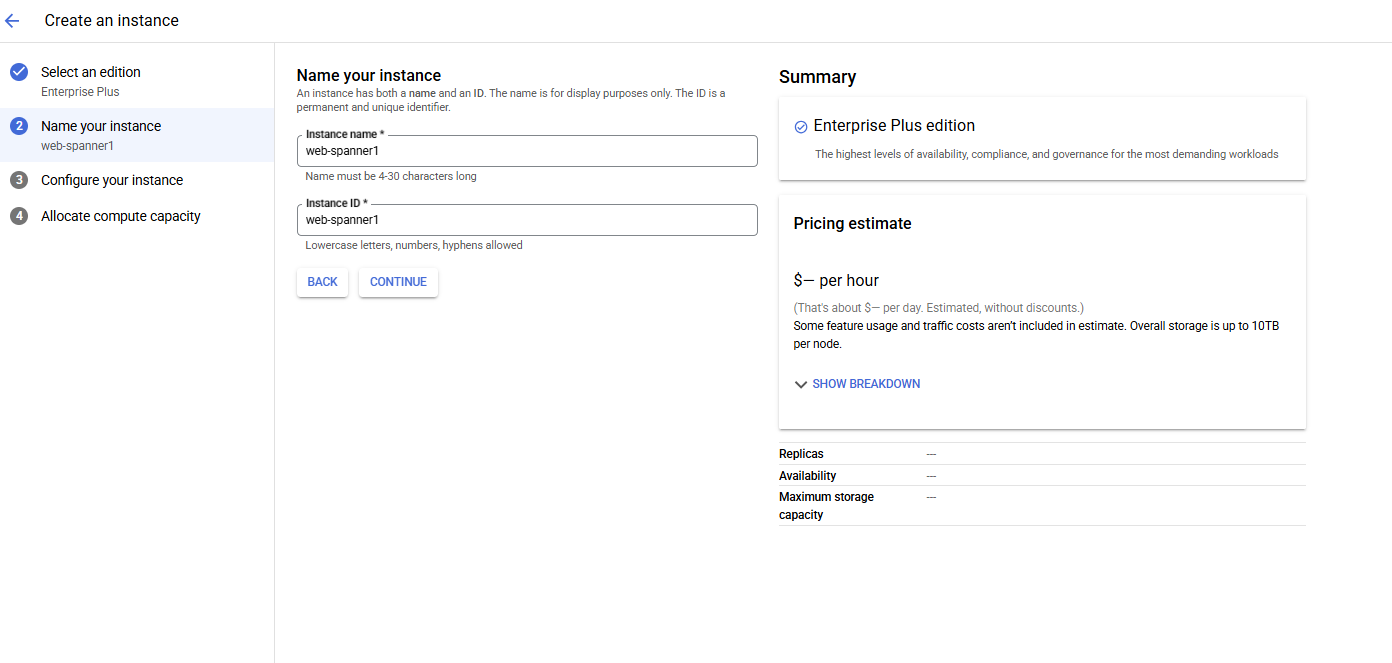
cloud run url run in browser

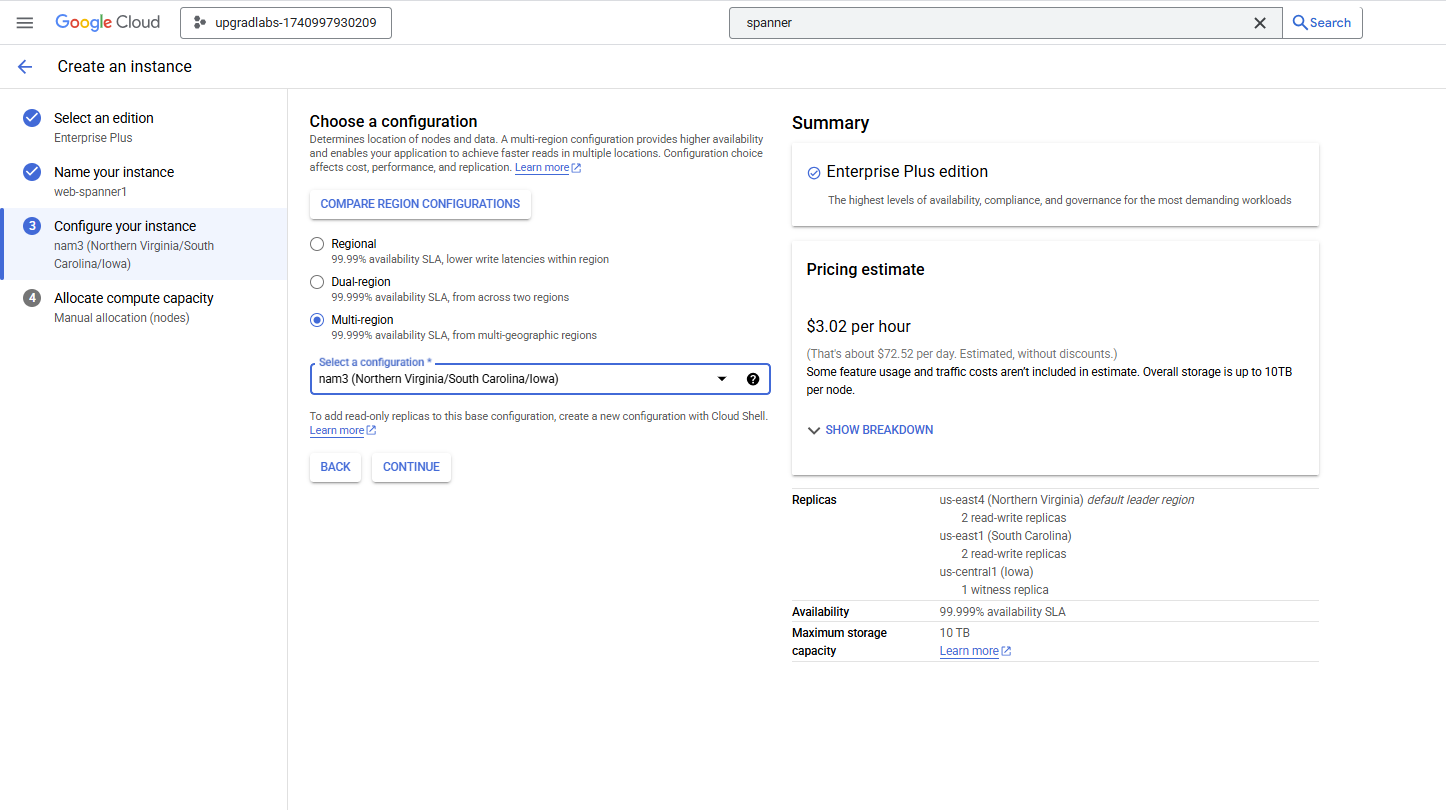


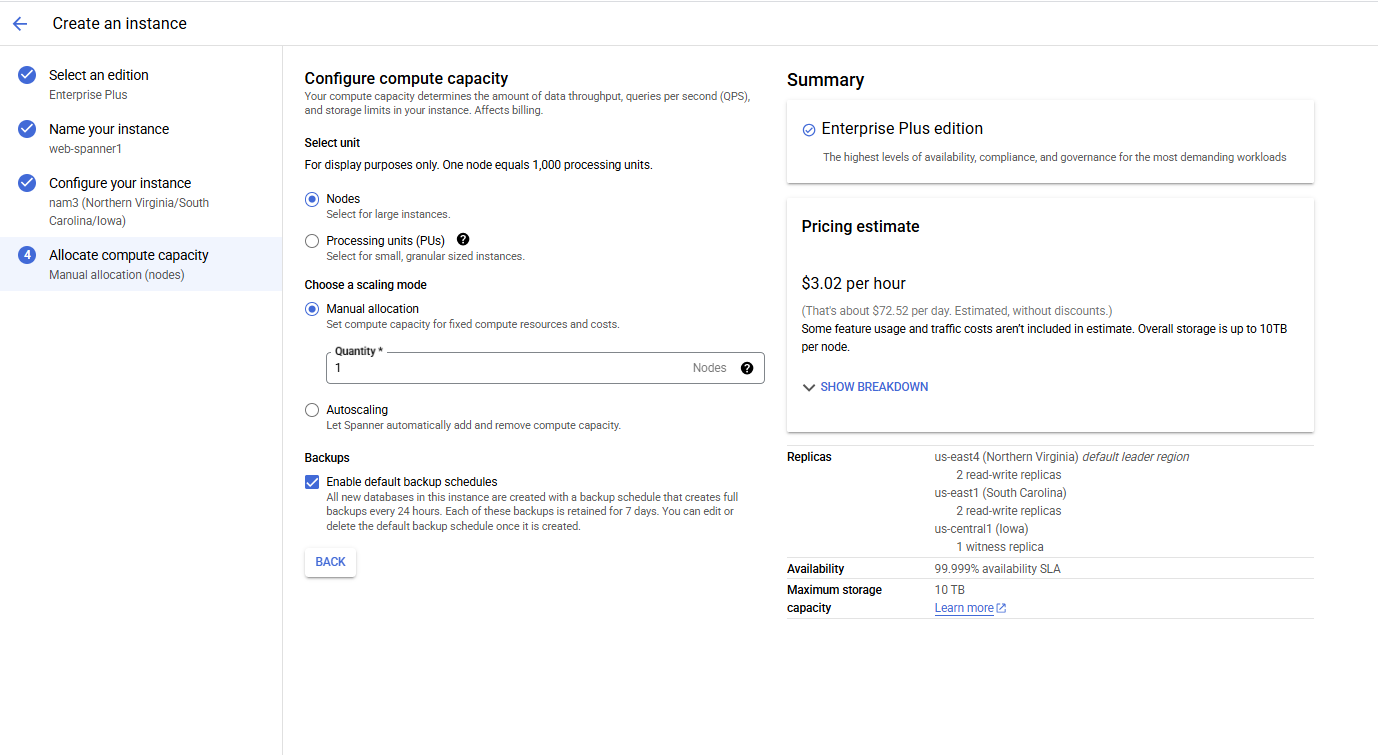
**Task 3 -** **Use Cloud Spanner for global-scale, high-traffic apps**

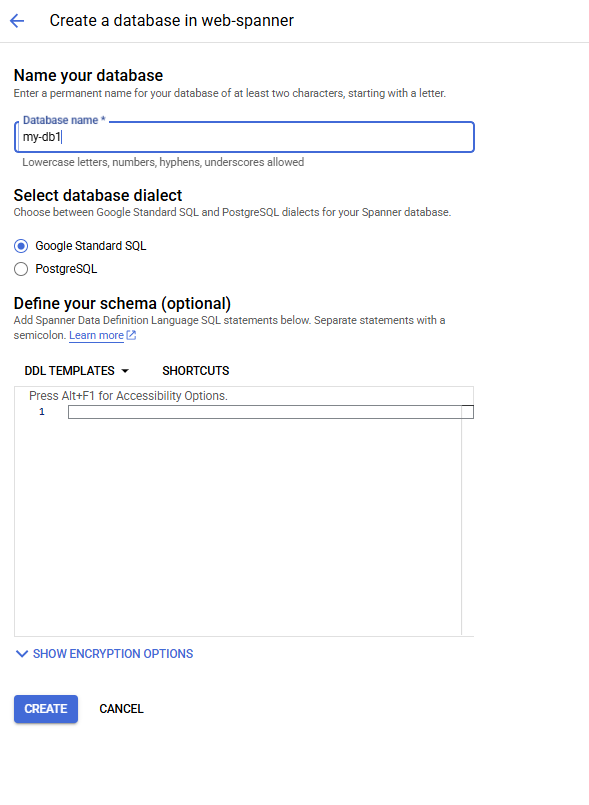
Create spanner instance



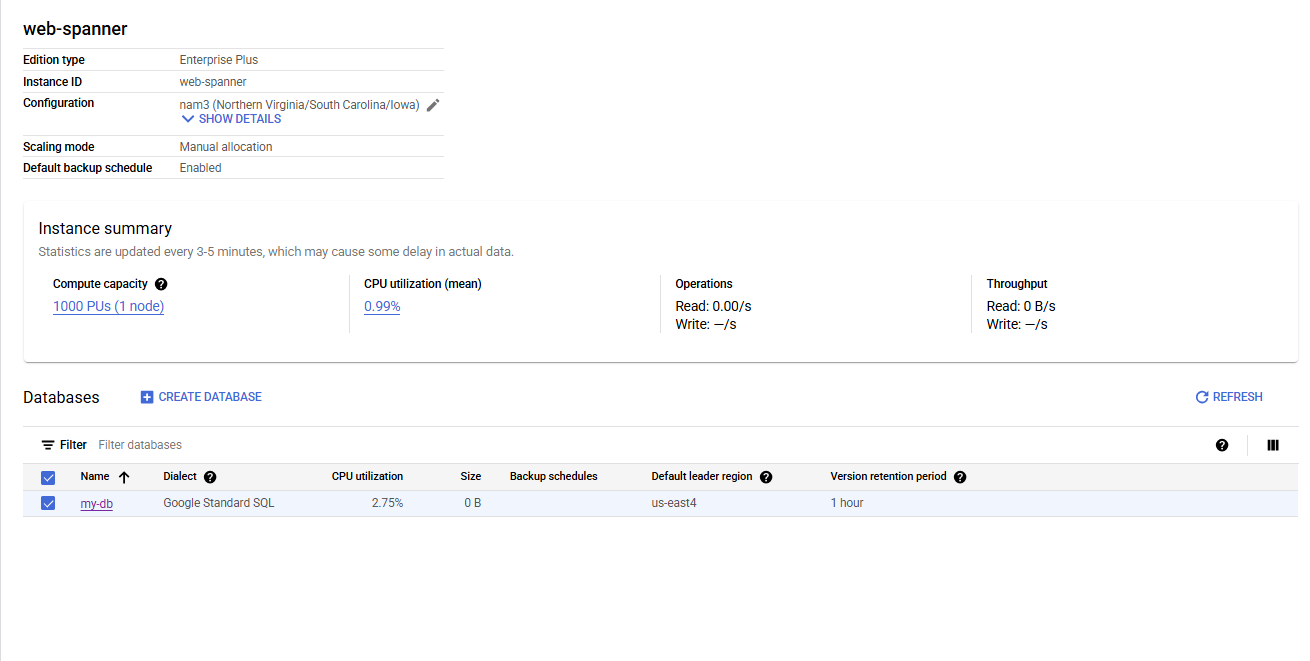
Configure instance

Automate compute capacity

Create Database



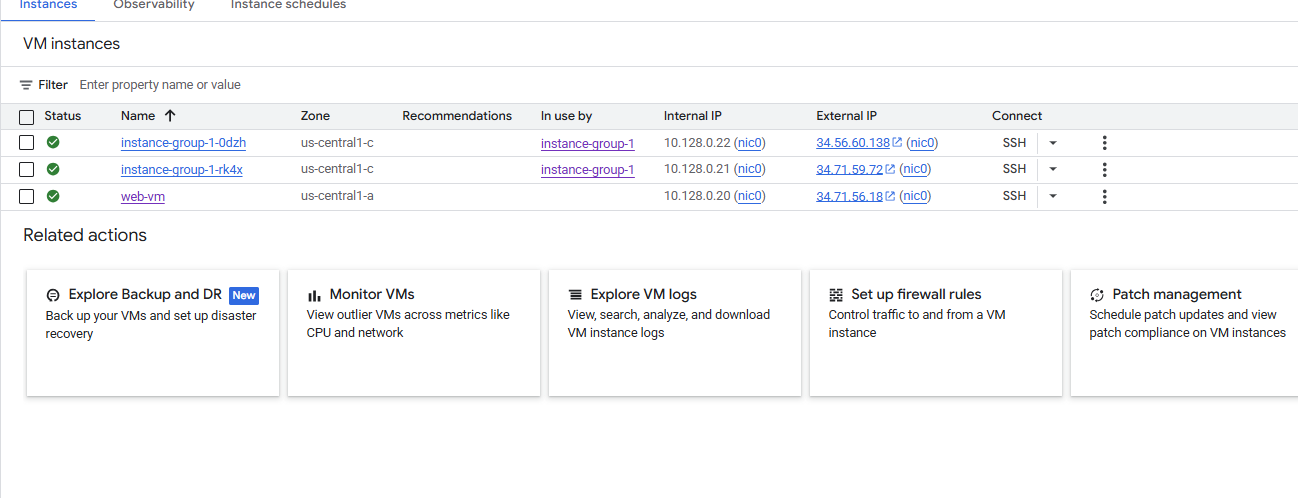
Web Spanner

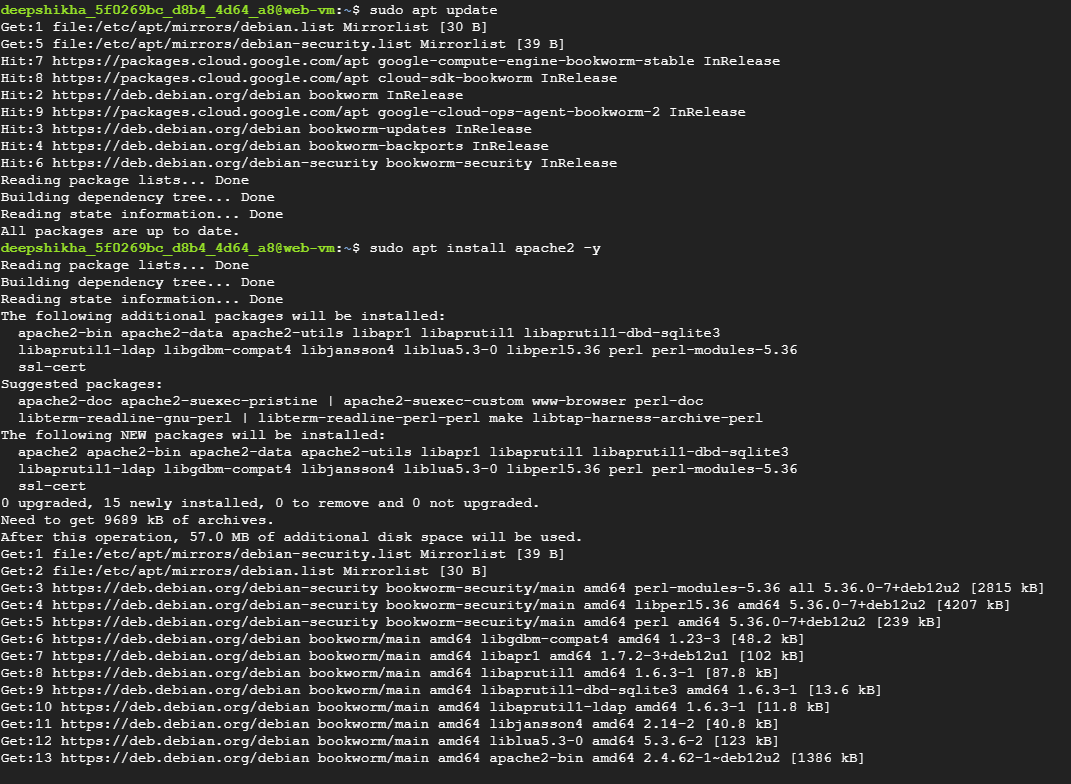


**Task 4 -** **Use Cloud CDN for low-latency delivery**

**Install a Web Server**

* + SSH into the VM → Run:

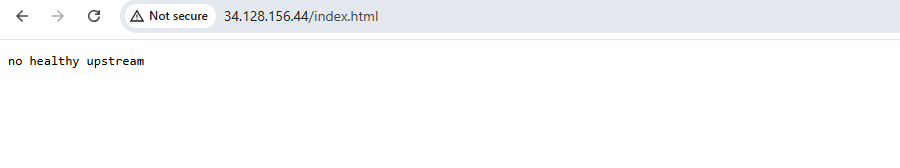




Output

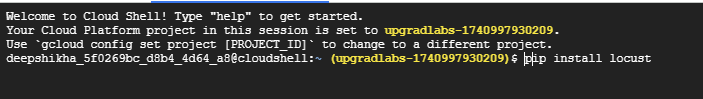


Test the CDN

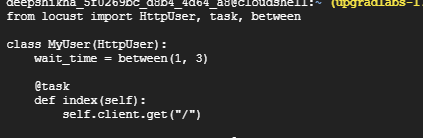


**Task 5 - Simulate extreme loads using Google Cloud Performance Testing Tools**

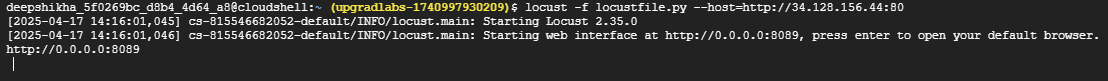
Option 1: Using Cloud Shell + Locust



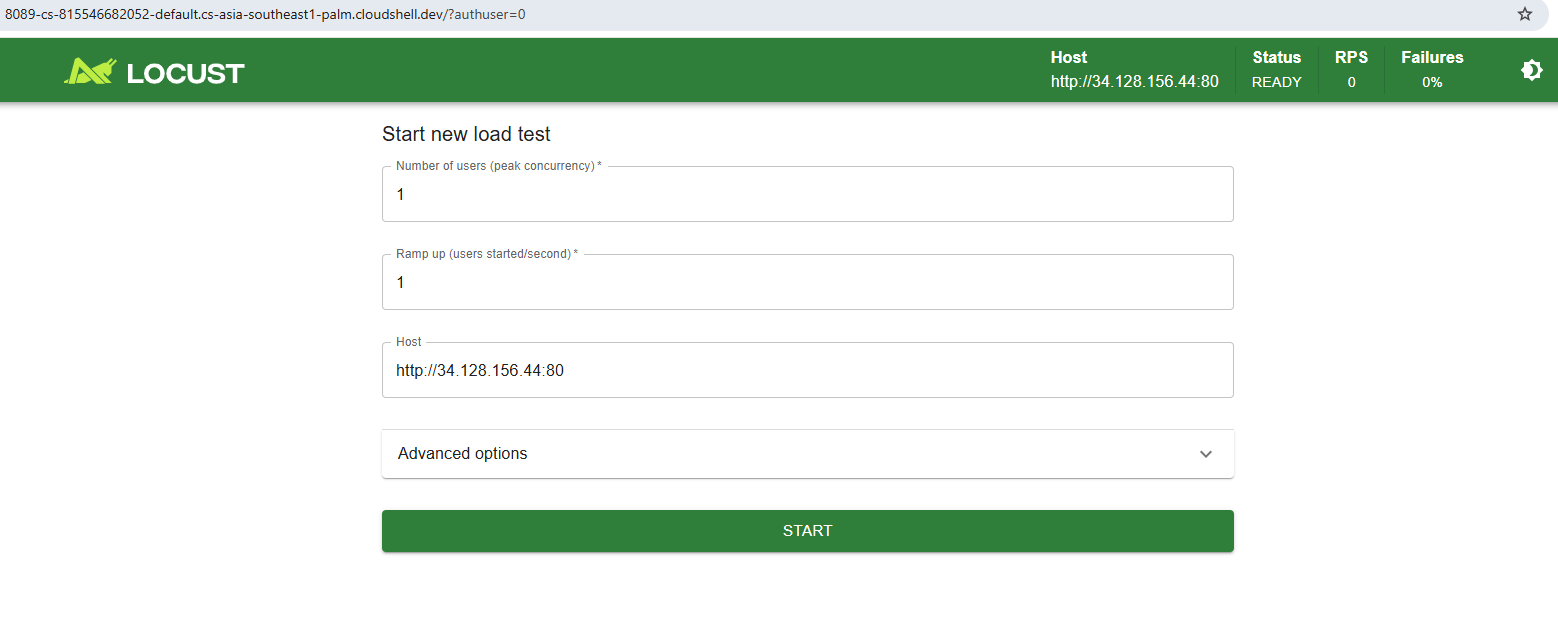
Create locustfile.py



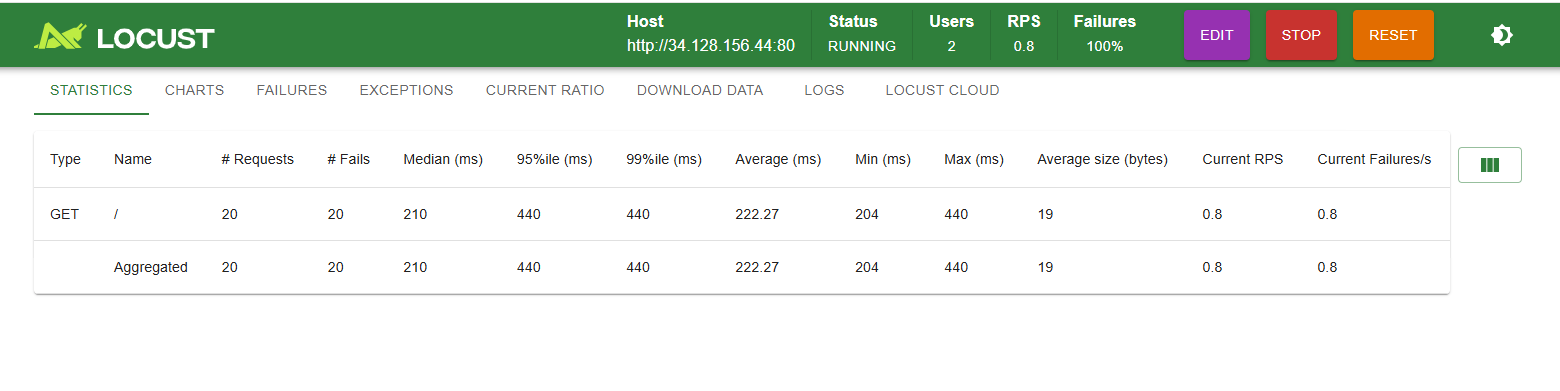
locust -f locustfile.py --host=http:// 34.128.156.44:80



Open URL given in Cloud Shell (usually port 8089)

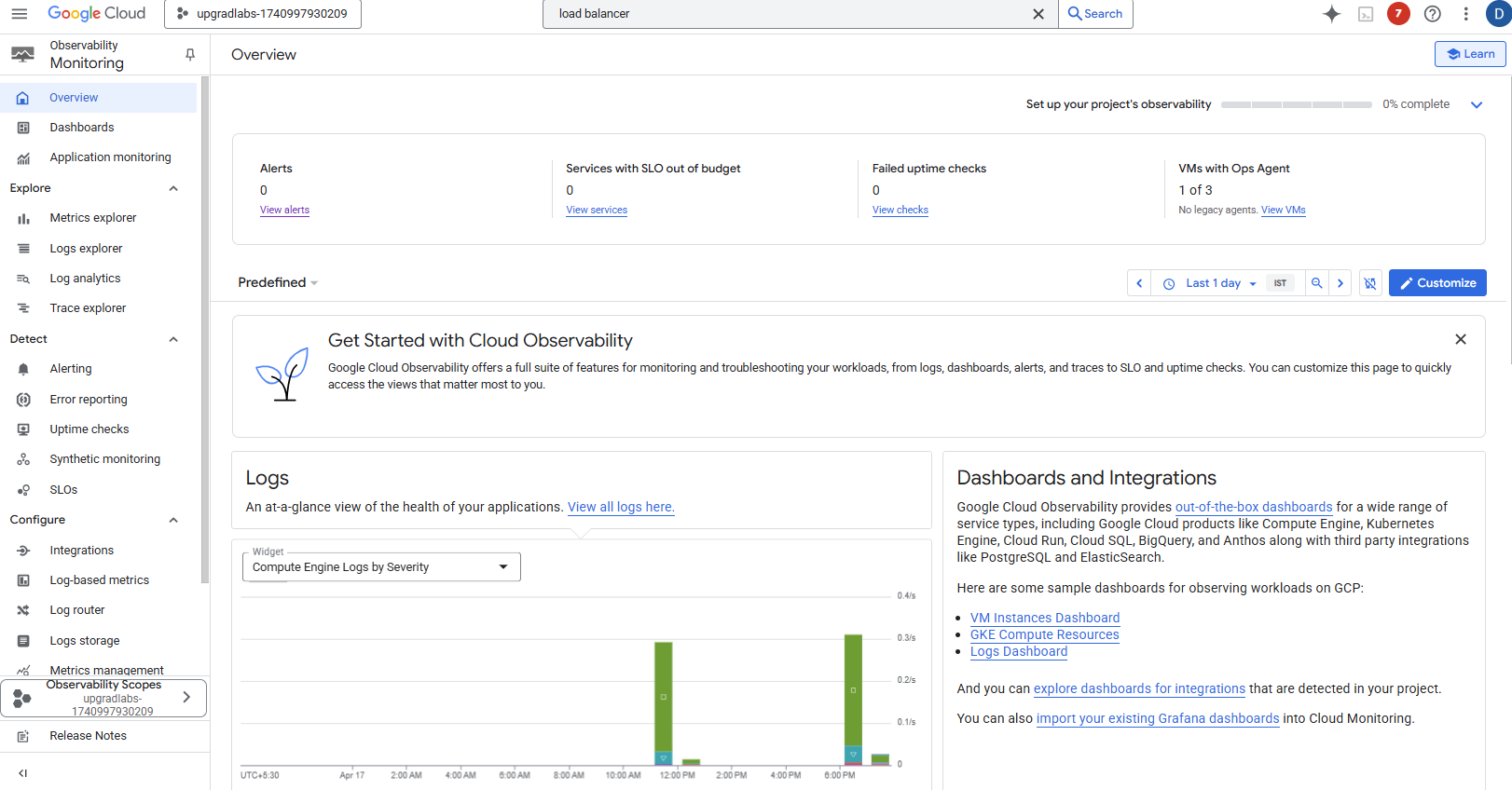


Start test with 10–100 users and monitor results

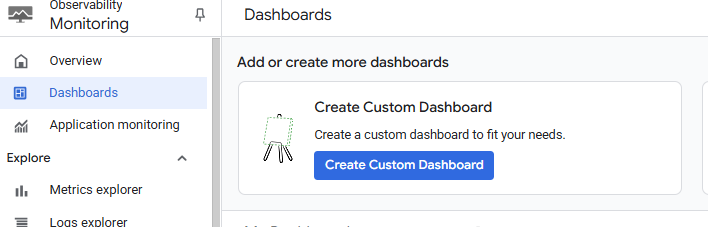


**Task 6 - Evaluate performance metrics and adjust scaling parameters**

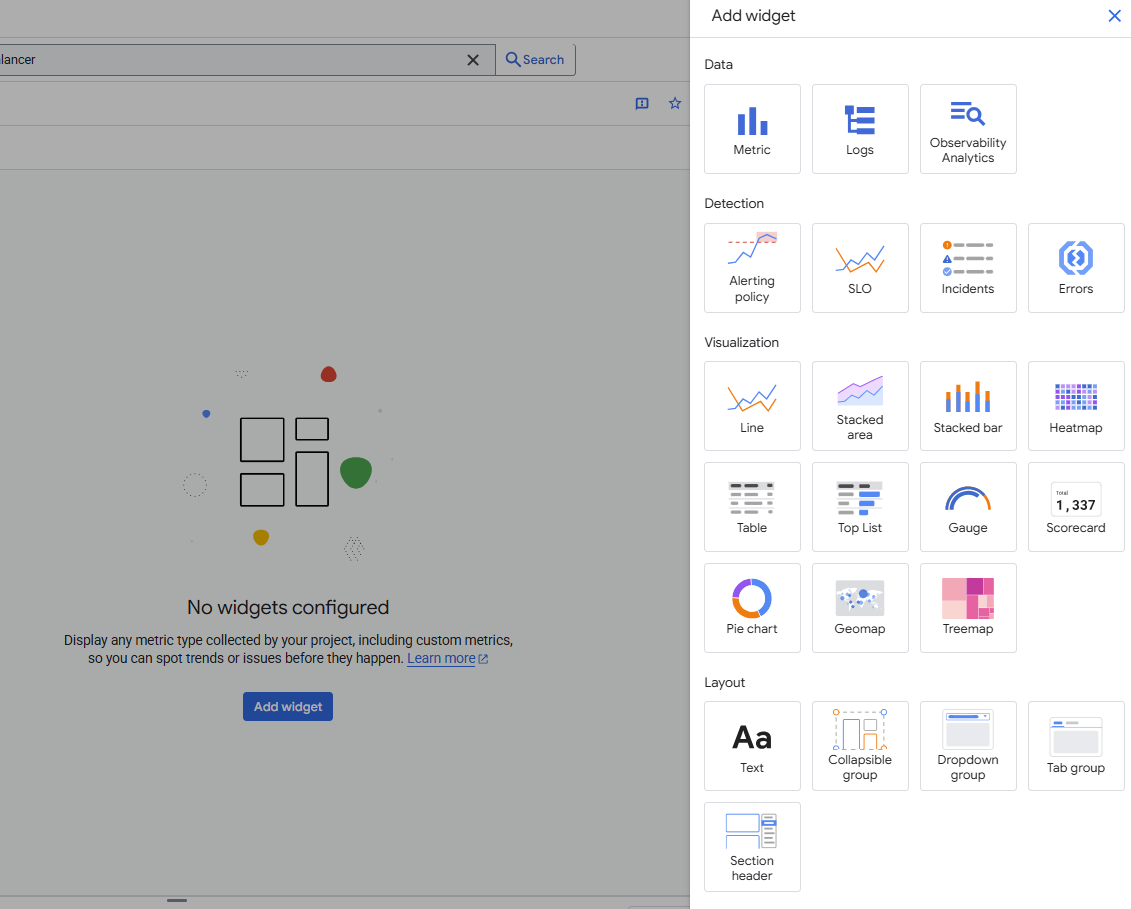
Go to Monitoring Dashboard

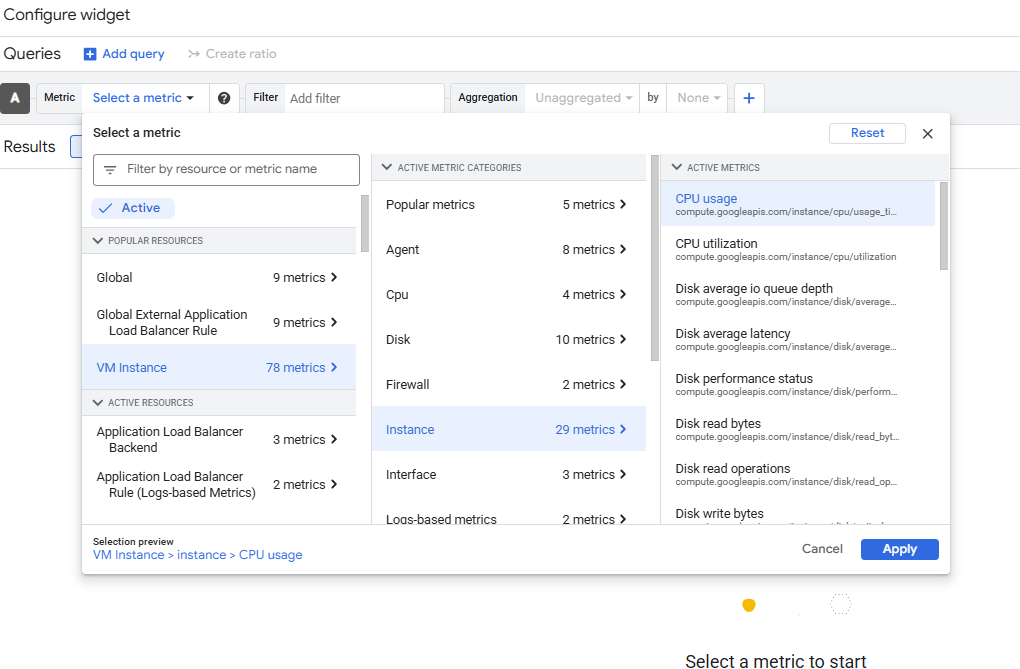
****

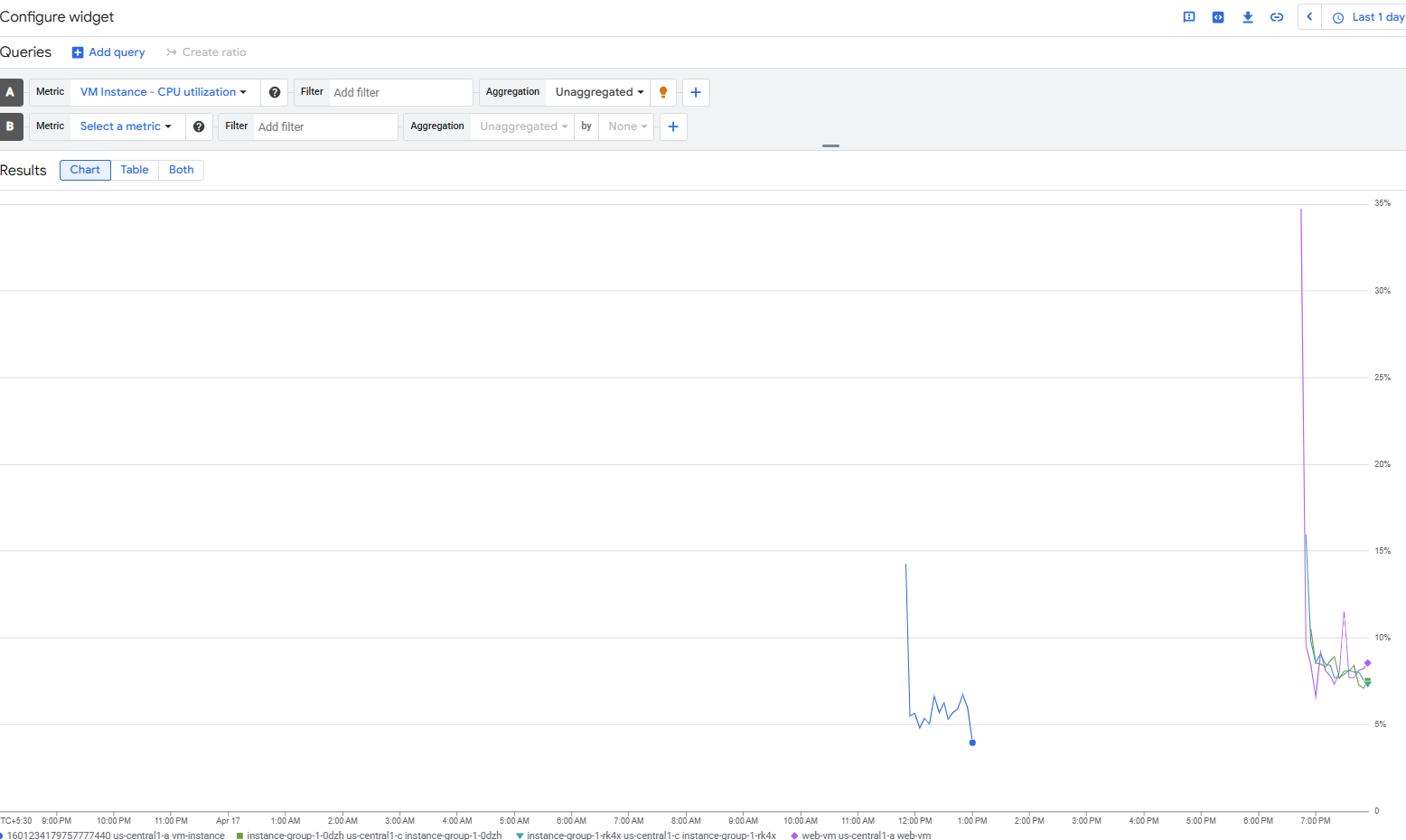
Create a Custom Dashboard



Select line option







Create alert policy

