Microservices Architectures - Practices with JHipster

Henrique Figueiredo Conte, Teodor Neagu, Tetiana Yakovenko

January 2023

1 Introduction

The goal of the project is to practice microservices architecture using JHipster, Google Cloud Platform (GCP), Prometheus, Grafana and Gatling. The project was made by the following group:

- Henrique Figueiredo Conte - Github - henrique.figueiredo-conte@grenoble-inp.org
- Tetiana Yakovenko - Github - tetiana.yakovenko@grenoble-inp.org
- Teodor Neagu - Github - teodor.neagu@grenoble-inp.org

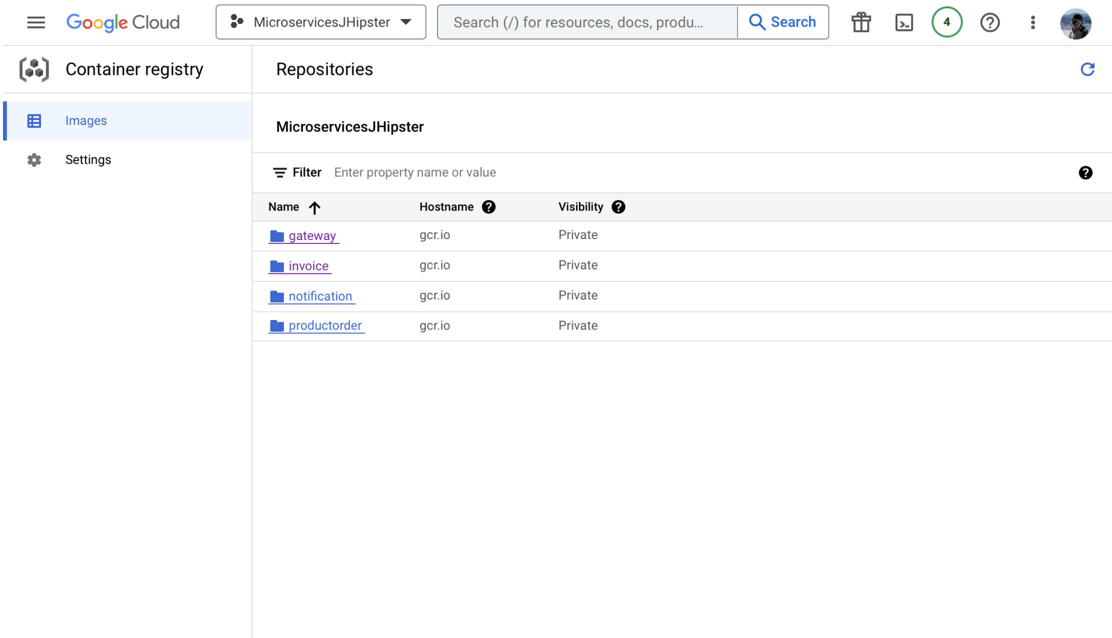
The source code is available on a Github repository.

2 Tasks

2.1 Deployment of Microservices with JHipster on GCP

The first task was to deploy microservices with JHipster on GCP. In order to do that, we followed the tutorial given by the teacher, and after following all the steps from the *Generate the microservices architecture*, we deployed the services to GCP:

Figure 1: Container registry

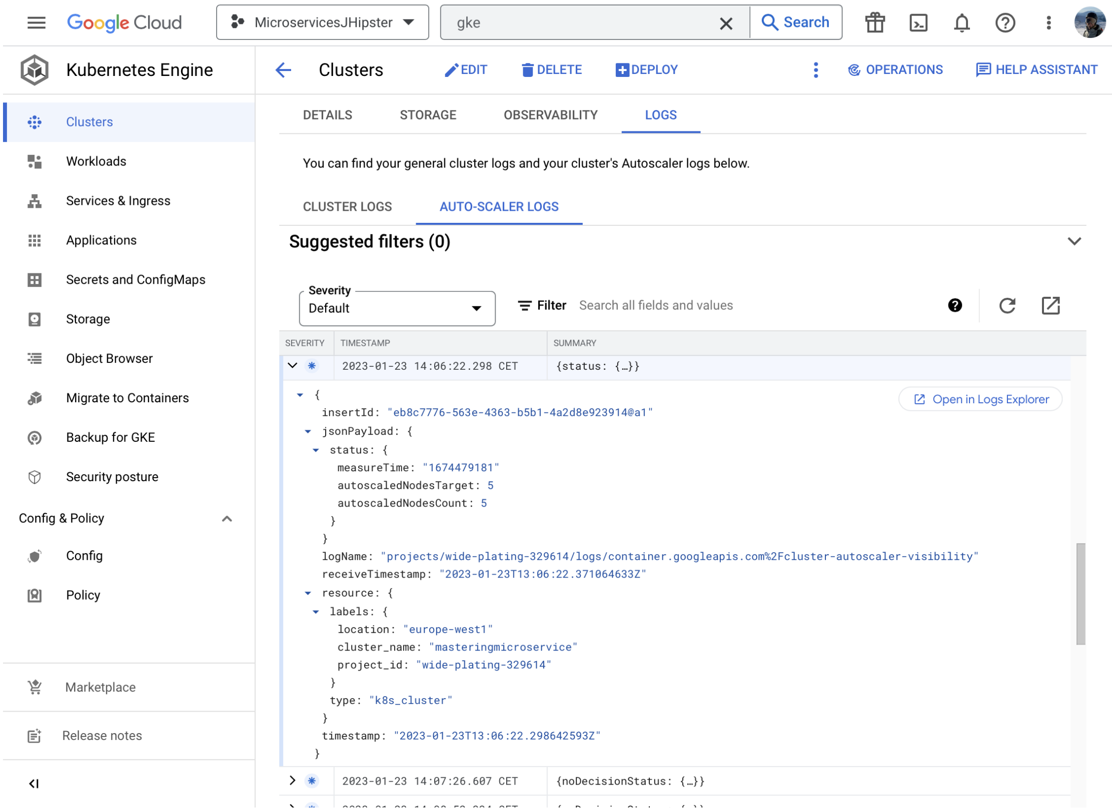


This is our container registry, that manages Docker images. There, we deployed the images from the *gateway*, *invoice*, *notification* and *productorder*.

2.2 Enabling scalability on GCP for one microservice

In order to enable scalability for at least one microservice, we followed the tutorial from Google, Scaling an application. The tutorial taught us how to enable auto scaling for a Kubernetes cluster.

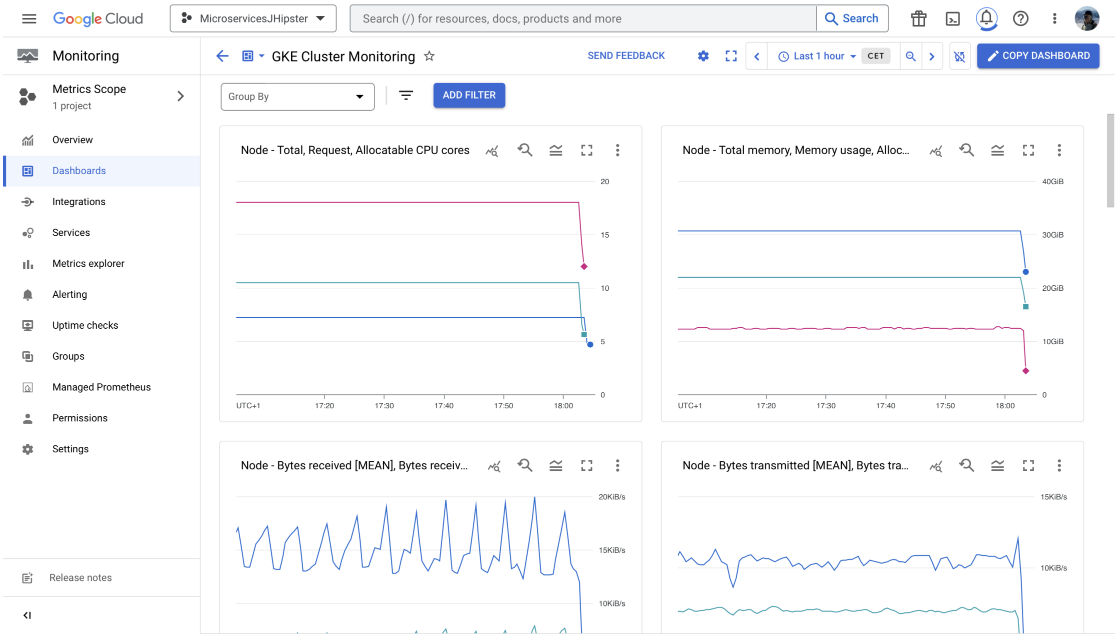
Figure 2: Kubernetes cluster auto scaling



2.3 Monitoring dashboard

The Kubernetes Engine from Google Cloud already provides a monitoring dashboard, showing metrics such as allocatable CPU cores and total memory. Here is a screenshot of our monitoring dashboard:

Figure 3: GKE monitoring dashboard



2.4 Load injection with Gatling for demonstrating scalability

Unfortunately, we didn't understand how Gatling works in order to demonstrate scalability. According to the settings we defined, we believe our cluster will autoscale, but we couldn't see it happening in practice.