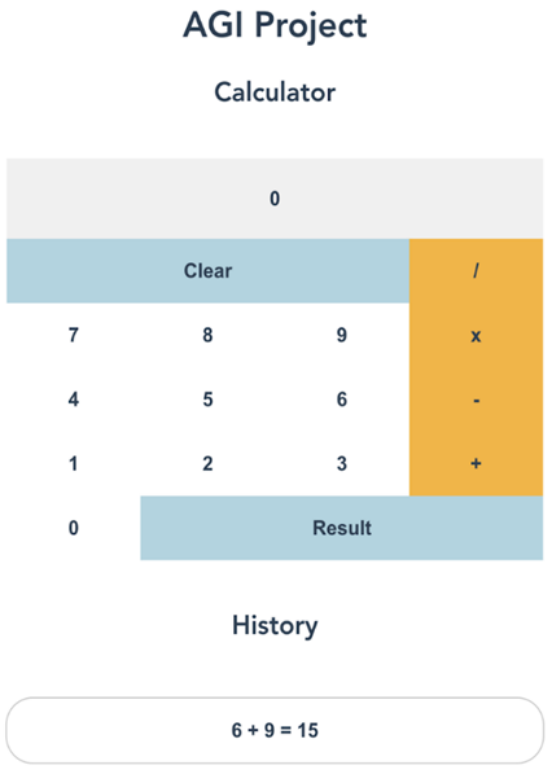
Project Report

AGISIT 23/24 Capstone Project



Group 36

Andrey Bortnikov (108489)

Pedro Dias Rodrigues (99300)

Sylvain David Kévin Ly Migeon (108819)

2023/2024

Introduction

This project outlines the steps involved in deploying and provisioning a tiered Microservices-based containerized Calculator Web Application on a Public Cloud Provider such as Google Cloud Platform (GCP), using container orchestration tools such as Kubernetes as well implementing instrumentation on the applications, services and infrastructure components of the solution, to allow monitoring features by using tools such as Prometheus and Grafana.

Architecture

The foundational architecture of the base solution should encompass the following primary services:
Frontend: Serving as the entry point, it exposes an nginx server for the website. Vue.js.
Backend Services: These services deliver the required functionalities. Java Spring Boot
Object Store/Cache/Database: Responsible for data storage. Redis.
Monitoring Server: Offers insights into the system's performance. Prometheus and Grafana.

The diagram below illustrates a typical application architecture:

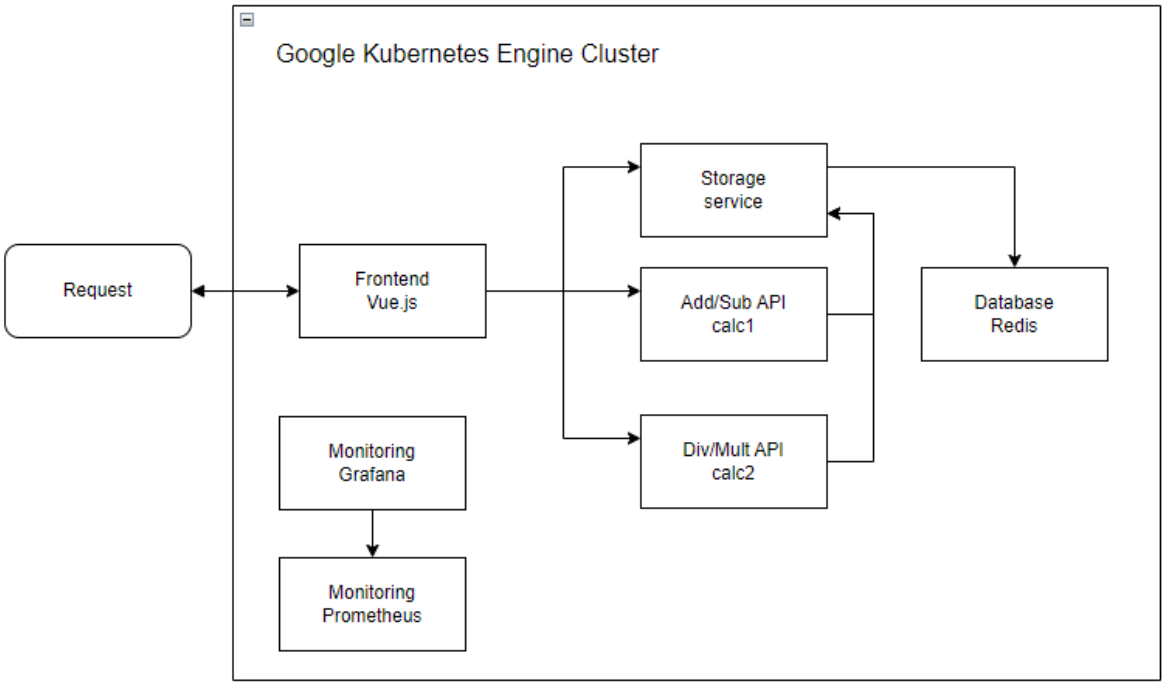


Figure 1 - Global Architecture

Here you will find the link of our project video: https://youtu.be/vWUOFTtiuDM

Methodology

After describing the project architecture, we will delve more deeply into the implementation methodology used, concluding with the enhancements we had the opportunity to add to our solution.

For our project, we have chosen to implement and deploy a Calculator with UI providing all the basic operators using microservices architecture.

As you can see in our architecture, we used several components which are all connected. For the frontend, we used Vue.js. In the front page, the user can do some calculations which call the APIs calc1 and calc2 and see a history of operations. Concerning the APIs calc1 and calc2, they are used to calculate respectively the addition and subtraction for calc1 and the multiplication and the division for calc2. These APIs not only do these calculations but store the result in our database. Talking about our database, we used Redis. It is a NoSQL in-memory database which only uses a key:value system. It stores the calculations which are displayed on the front end in the History section. Finally, we have 2 monitoring components which are Grafana and Prometheus. Grafana is used to create dashboards with metric data, it gives a more visual aspect to the data. Prometheus collects and stocks metrics data to be used by Grafana for visualization.

For a better understanding of the workflow of our architecture, the flowchart below explains how a calculation is processed in our solution.

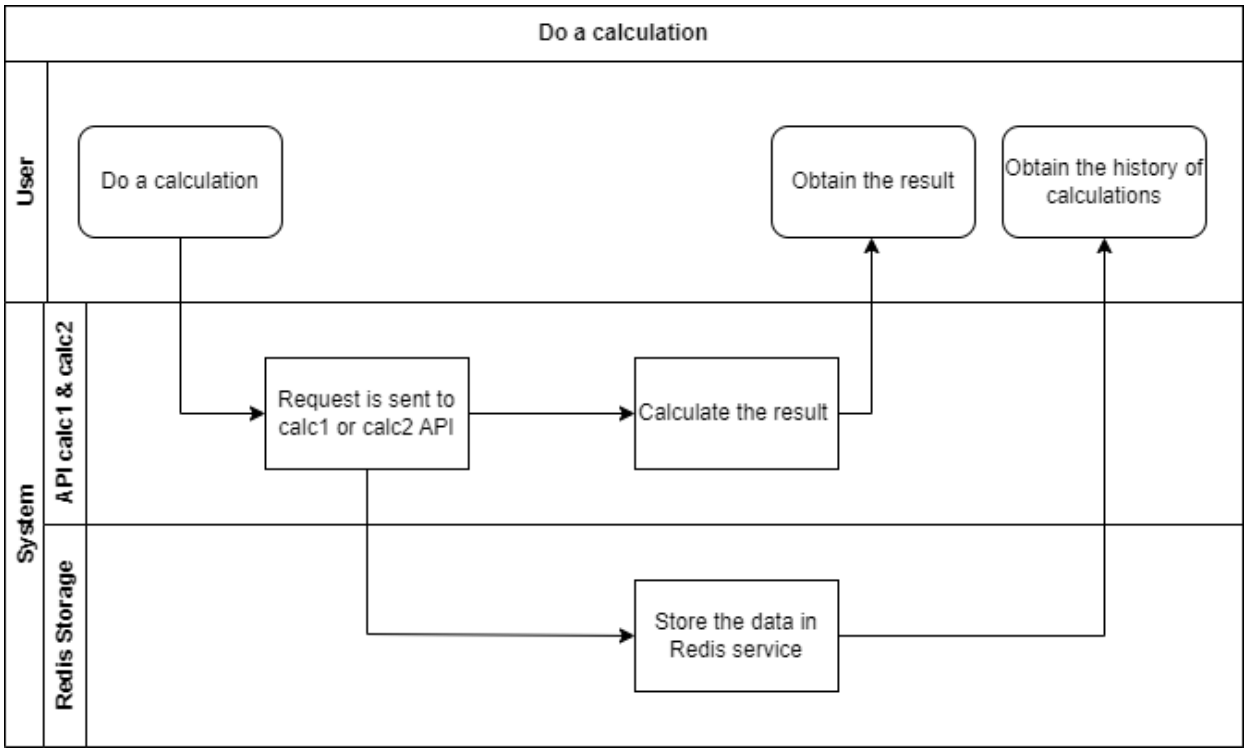


Figure 2 - Flowchart of a calculation

Components

In this subsection, we describe the main components of our solution, which include the K8S cluster, an alternative to Docker Hub, application deployments and services, the Ingress, and monitoring tools.

Now, let's dig into our implementation methodology and detail how we implemented and linked our components. Front-end is written in Vue.js. It is communicating with back-end services with Axios(a promise-based HTTP Client for Node.js and the browser). Back-end is written in modern Java 21, consists of 3 services: calc1, calc2 and storage. Mathematical operations are distributed between calc1 and calc2 just to add additional service and demonstrate microservices architecture. Storage is connected to the independent Redis service. Grafana is connected to Prometheus to collect data and serve a convenient UI with metrics. Both Grafana and Prometheus are separate services.

Kubernetes (K8S) Cluster

K8S is an open-source container orchestration system that automates the deployment, scaling, and management of computer applications. To set up a cluster, yaml configuration files have been created. Configuration can be found in our GitLab repository.

Application Deployments and Services

Our application consists of a series of deployments, pods, and services. Deployments guide K8S in creating or modifying instances of pods that host containerized applications. A pod comprises one or more containers that share storage and network resources, following specific run instructions.

Services offer a logical abstraction for groups of pods in a cluster. Since pods are ephemeral, services provide a consistent name and unique IP address to a group of pods that serve the same purpose. Even when pods are replaced, the service ensures accurate referencing. The application relies on the following services:

- **Vuecalc:** Facilitates interactions with calculator functionalities.
- **Storage:** Retrieves data and operations performed by the calculator.
- **Redis:** Functions as the data storage solution for the Storage service.
- **Calc1**: Performs Add and Subtract functions, saves operations to storage.
- **Calc2**: Performs Divide and Multiply functions, saves operation to storage.

Monitoring

To monitor the cluster, we use a prepared and automatically configured application from Google Marketplace called Prometheus. This solution simplifies end-to-end monitoring of the K8S cluster using Prometheus components. It automatically constructs the necessary Prometheus configurations and finds all cluster endpoints to receive data .This setup enables the storage of real-time metrics by Prometheus, which then can be visualized in Grafana.

Description of REST endpoints:

Storage service
Endpoints:

1. GET /healthz - health check
2. GET /operations - list all saved operations
3. POST /create - create a new operation
4. GET /delete - delete an operation (used for scheduled job to delete old operations)

Calc1 service
Endpoints:

1. GET /healthz - health check
2. GET /add - add operation for 2 numbers
3. GET /sub - subtract operation for 2 numbers

Calc2 service
Endpoints:

1. GET /healthz - health check
2. GET /multiply - multiply operation for 2 numbers
3. GET /divide - divide operation for 2 numbers

Deployment Methods and Advanced Components

We successfully deployed a Cloud-Native application using Google Kubernetes Engine (GKE) on Google Cloud, creating a high-availability and scalable infrastructure.

We leveraged the Google Cloud SDK, kubectl, Kubernetes providers, and Google Cloud Console throughout the project. To set up the project, we created a new project in Google Cloud, enabled APIs, and generated credentials. The infrastructure deployment was executed by running ‘kubectl apply’. After it, we verified and checked the infrastructure using the ´kubectl get’ command to ensure its functionality.

Additionally, as shown in the video, we added a Prometheus application from the Google Marketplace to our project to enable the monitoring and instrumentation.

Through the development of this project, we gained hands-on experience in deploying Cloud-Native applications in a GCP Kubernetes cluster, highlighting the power of Kubernetes for managing containerized applications at scale.

Conclusion

In conclusion, our Capstone Project achieved its primary objective by successfully deploying a tiered Microservices-based containerized Web Application on Google Cloud Platform (GCP). We meticulously followed an Infrastructure as Code (IaC) approach, leveraging Kubernetes for resource provisioning and ensuring that our solution can be easily reproduced in the future. This accomplishment not only demonstrates our practical skills in cloud architecture and deployment but also underscores the significance of ensuring consistency and scalability. Moreover, we implemented a robust monitoring and instrumentation system using Prometheus and Grafana, which grants us valuable insights into the performance and health of our application, serving as a foundation for proactive issue resolution and future optimization.

Throughout this project, we learned the essential value of structured collaboration. The foundation we've built opens doors to future possibilities for advanced features, such as a Continuous Integration/Continuous Deployment (CI/CD) pipeline and more complicated Kubernetes configurations. These enhancements could further elevate the functionality, resilience, and scalability of our application, showing our readiness to embrace emerging technologies and their potential impact on cloud-native solutions. This project signifies the culmination of our educational journey, arming us with practical insights and skills applicable in real-world scenarios, both for enhancing existing systems and embarking on new cloud-native endeavors.