

Rapport Projet Cloud



Software Engineer for the Cloud

Vincent BERNET – Matthieu GEDEON

Table des matières

Badges Ateliers	3
Vincent BERNET	3
Matthieu GEDEON.....	4
Code repository	5
Introduction	5
Kubernetes	5
Architecture & technologies.....	6
Minikube	6
Interesting features.....	6
Problems encountered.....	6
Achievements	7
Possible improvements.....	7
Docker	8
Dockerfile	8
Docker-compose.yaml	8
Conclusion	9

Matthieu GEDEON



Matthieu Gédéon

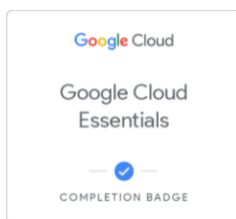
Date d'abonnement : 2021

Votre profil n'est pas public ni accessible.

[Rendre votre profil public](#)

Activités

Badges



Google Cloud Essentials
Earned mar. 24, 2022 EDT

[En savoir plus](#)



Matthieu Gédéon

Date d'abonnement : 2021

Votre profil n'est pas public ni accessible.

[Rendre votre profil public](#)

Activités

Badges

Cours Atelier Quête Quiz Jeu

En cours Terminée

Activité	Type	Date de début	Date de fin	Score	Réussie
Configurer des équilibreurs de charge réseau et HTTP	Atelier	il y a 48 minutes	il y a 12 minutes	100.0/100.0	✓
Infrastructure as Code avec Terraform	Atelier	il y a 18 heures	il y a 17 heures	100.0/100.0	✓
Creating a Streaming Data Pipeline With Apache Kafka	Atelier	il y a 18 heures	il y a 18 heures	100.0/100.0	✓
Kubernetes Engine : Qwik Start	Atelier	10 nov. 2021	10 nov. 2021	100.0/100.0	✓
Getting Started with Cloud Shell et gcloud	Atelier	9 nov. 2021	9 nov. 2021	100.0/100.0	✓
Compute Engine : Qwik Start – Windows	Atelier	9 nov. 2021	9 nov. 2021	100.0/100.0	✓
Créer une machine virtuelle	Atelier	9 nov. 2021	9 nov. 2021	100.0/100.0	✓
A Tour of Google Cloud Hands-on Labs	Atelier	9 nov. 2021	9 nov. 2021	100.0/100.0	✓
Google Cloud Essentials	Quête	9 nov. 2021	il y a 18 minutes		✓

Code repository

Our code is available on Github, through this [Github Repository](https://github.com/gegedev/cloud-project) (<https://github.com/gegedev/cloud-project>). The README is important and hopefully sufficiently detailed in order to test our applications. The project was made in a machine using Windows 10.

Introduction

For this project, we unfortunately did what we could regarding the fact that our team is made of 2 working student in apprenticeship and we had a lot of work to tackle this year.

We thus decided to do a project that will familiarize us with concepts linked to Software Engineering for the Cloud and we established a list of improvements possible for our small application regarding Cloud related technologies.

Our project is made of 2 containers, that is to say pods, inside a Minikube Kubernetes cluster composed of a single node. One contains our small Node JS Express application, made from a local docker image and another contains a MongoDB, created from the Docker Hub remote image.

The small Node Js app just render a profile, the latter can be updated and stored in MongoDB.

We will talk in this report the technologies we used and how we tackled some problems, with also a point on improvements possible for our small application.

Kubernetes

[Kubernetes](#) is a portable, extensible, open-source platform for managing containerized workloads and services, that facilitates both declarative configuration and automation. It has a large, rapidly growing ecosystem. Kubernetes services, support, and tools are widely available.

It permits to orchestrate several services with a load balancer, self-healing capabilities, secret and configuration management and even more.

Its usable using the Kubernetes command tool: kubectl.

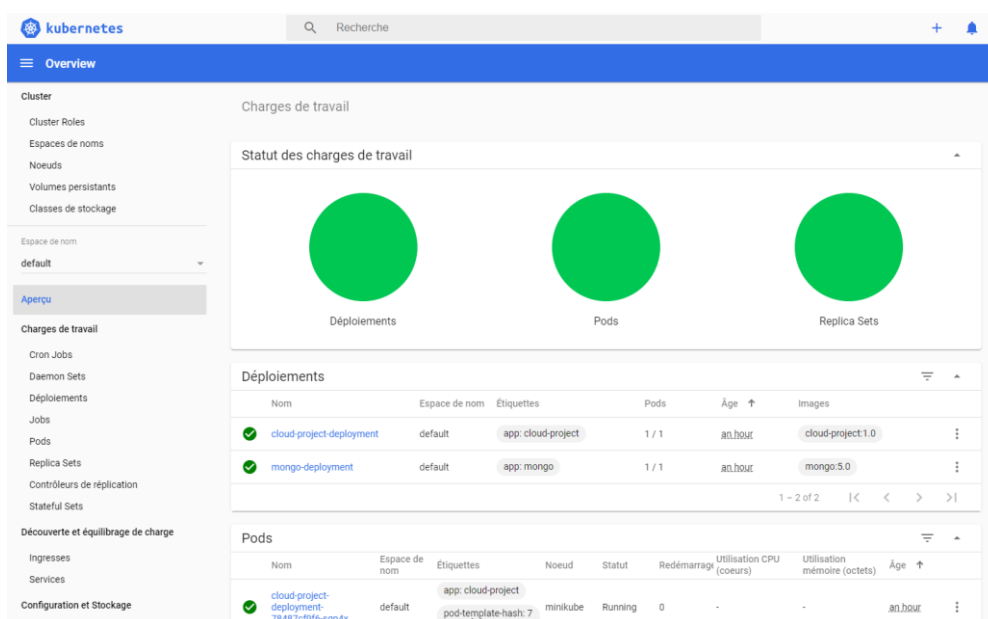
Architecture & technologies

Minikube

[Minikube](#) is a tool that permits to initialize a small Kubernetes Cluster with a single node on a local machine. It is disponible for MacOS, Linux and Windows. We chose the latter for our project.

Interesting features

Minikube provides a lot of cool features. Most importantly is the Kubernetes dashboard accessible using the command `minikube dashboard`. It shows the user informations about his services and many more. Those service can also be accessed quickly using the command `minikube service [service – name]`.



Kubernetes Dashboard

```
PS C:\Users\matth\Documents\Efrei\M2\Cloud\cloud-project\configurations> minikube service cloud-project-service
* Starting tunnel for service cloud-project-service.
+-----+-----+-----+-----+
| NAMESPACE | NAME | TARGET PORT | URL |
+-----+-----+-----+-----+
| default | cloud-project-service | | http://127.0.0.1:53049 |
+-----+-----+-----+-----+
* Opening service default/cloud-project-service in default browser...
! Because you are using a Docker driver on windows, the terminal needs to be open to run it.
```

Minikube service access

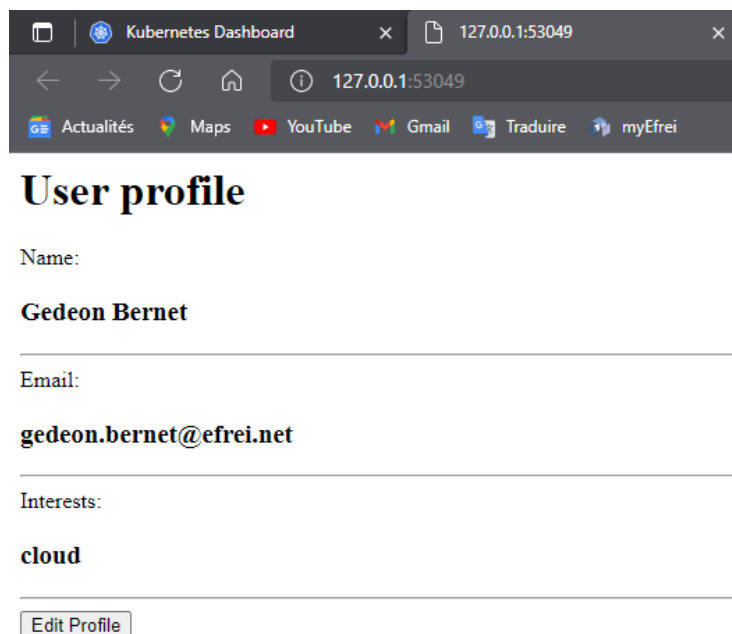
Problems encountered

Minikube is started using docker as a driver (commands and setup in repository's README) to make it run as a container. The only problem is that the Minikube container has its own version of Docker inside it which raised some problems while trying to build containers from images that were not referenced in the registry. We fixed that problem following instructions made on this [Medium blog post](#).

We also had a problem regarding the Minikube version, which made our deployed services unreachable. We fixed it through advices found on this [StackOverflow issue](#).

Achievements

We successfully deployed 2 services using 2 yaml files as build descriptions. These files also consume a ConfigMap and a Secret yaml file in order to store useful and shared information.



User profile

Name: **Gedeon Bernet**

Email: **gedeon.bernet@efrei.net**

Interests: **cloud**

[Edit Profile](#)

User profile

Name:

Email:

Interests:

[Update Profile](#)

Deployed app

Possible improvements

Minikube is useful in order to test small applications and get the gist of Kubernetes. But to deploy a more realistic and practical application architecture, we could use [K3s](#) or kubeadm in order to have multiple nodes and a more robust environment (at the cost of an increased complexity).

Docker

Docker takes away repetitive, mundane configuration tasks and is used throughout the development lifecycle for fast, easy and portable application development – desktop and cloud.

It makes easy to build, share and run applications around different machines using different OS, as it is inspired on the concept of virtualization but in a lighter way.

We used it in order to create an image of our small application, but also tested container interoperability using a docker-compose file to launch 2 containers.

Dockerfile

```
FROM node:12.18.1
ENV NODE_ENV production
ENV MONGO_DB_USERNAME=admin \
    MONGO_DB_PWD=password
WORKDIR /usr/src/app
COPY ["package.json", "package-lock.json", "./"]
RUN npm install --production --silent && mv node_modules ../
COPY . .
EXPOSE 3000
CMD npm start
```

Docker-compose.yml

```
version: '3.4'

services:
  mongodb:
    image: mongo
    ports:
      - 27017:27017
    environment:
      - MONGO_INITDB_ROOT_USERNAME=admin
      - MONGO_INITDB_ROOT_PASSWORD=password
    volumes:
      - mongo-data:/data/db
  cloud-project:
    image: cloud-project
    build: .
    environment:
      NODE_ENV: production
    ports:
      - 3000:3000
    volumes:
      - ../usr/src/app
    depends_on:
      - mongodb
volumes:
  mongo-data:
    driver: local
```


Conclusion

We know that our project might be a bit lighter compared to a real cluster using subscriptions with Kafka.

But we understood the grasp of Cloud development basis and put in practice using a small example.