Assignment 1

Building a Kubernetes Deployable Microservice Based Application

PA2577 - Tillämpad Cloud Computing och Big Data

Johannes Botéus (jobx18@student.bth.se)

December 01, 2023

Blekinge Institute of Technology
371 79 Karlskrona
Sweden

Contents

1 Software Description

The presented software is a prototype of a simple website where one can up- and down vote a collection of famous quotes where anyone, who has access to the website, can participate. The current state of the most up votes represents the most popular quote choice.

2 Software Architecture Design

The software aims to present a simple demonstration of a microservice-based Kubernetes deployment of a frontend- and backend service which communicate with each other via RESTful API calls. The backend, in turn, communicates with a database. The frontend and backend are implemented with Flask and the database provider is MongoDB. The whole infrastructure is layed out as three Kubernetes configuration files in their respective roles where the frontend and backend are made possible to be scaled either by reapplying with a new replica count or possibly changing the configuration to be performance based, hence changing the pod count depending on resource/performance demand. The database is not prepared to be horizontally scaled but the choice of using a NoSQL database should aid in possible efforts to horizontally scale, if one would like to make that choice which could prove more difficult with a classic relational database.

All Kubernetes deployments are reachable inside the cluster but also from outside by using LoadBalancer Services making it possible both for external browser API requests to reach the backend as well as making it possible to balance traffic evenly between all services and let cloud providers give every service a external reachable address. The database is also configured to have some persistent storage to be able to pause and restart the k8s deployment.

3 Architecture Challenges and Security

A benefit of having everything separate with the API's is the fact that one could have the different services split up on different cloud or cloud/local (hybrid) infrastructures. If one would want to stay compliant with certain privacy laws and regulations by hosting the database on-premise, this architecture would make this possible. The main traffic would be balanced in the cloud with sufficient resources to uphold in times of heavy usage while the database could be on-prem and only gets hit possible once to load some precached data to the client browser while every site refresh would be handled in the cloud.

A important point for this kind of deployment is the security check and validation of RESTful API calls between the frontend and backend as well as

the database. This deployment does not focus on these aspects but production deployments should make use of K8s possibility to use secrets in their configuration files and the actual loosely connected microservice should also use authenticated tokes when communicating with each other. The database is also not locked down and uses the default out-of-the-box settings. Setting correct environment variables and securing from not getting leaked in possible version control software is an important part.

Other risks in this deployment is also the possibility of data loss by having persistent storage being part of a certain worker node and not the complete deployment. Using cloud native storage solution and/or network storage, which is backed up for critical data, is also an important consideration.