Project

This project aims to teach you how to deploy a full application in a Kubernetes cluster.

Application

The application you're going to deploy is a simple ecommerce admin where a visitor can manage catalog and search products by keywords. It has been built in a microservice architecture:

- 4 applications:
    - A frontend application in Angular
    - An API in Laravel (PHP)
    - An indexer in Node.JS
    - A reporting job in Go
- 3 databases:
    - MySQL
    - RabbitMQ
    - Elasticsearch

The web application can create or delete products by requesting the API. On product change, the API publishes updates into a RabbitMQ exchange. The indexer consumes a RabbitMQ queue and indexes products into Elasticsearch. The frontend application sends search queries to Elasticsearch, through the API.

Once a day, at midnight, the reporting job will send the number of products available in the database to Microsoft Teams, using a webhook.

See architecture diagram in appendix, at the end of the document.

Requirements

- Your infrastructure must be resilient. Services should be replicated when it's possible.
- Auto-scaling rules must allow your platform to accept a growing number of requests.
- A node failure must not cause any downtime. A delay in indexation is permitted.
- Your application resource consumption must be limited.
- Docker images must be stored on a private Docker registry (such as Github).
- Databases must be deployed with Helm (charts built by the community are good).
- Databases must be secured with a password. Some custom configuration may be set, for better performance.
- Only the load balancer must be exposed to the Internet.
- Propose a zero-downtime deployment strategy.
- Passwords must be stored in Kubernetes secrets.
- Kubernetes resources must have coherent labels, in order to filter resources easily.
- Application must be deployed using a dedicated Helm chart.
- Build your Helm charts for being customizable (example: resource consumption) and reusable (example: for other PHP applications, a React frontend...).

Choose between 2 of the following advanced requirements:

- A monitoring stack based on Prometheus and Grafana (example: a dashboard for monitoring PostgreSQL)
- Log aggregation (such as ELK)
- GitOps: deployment pipeline using Argo
- Run serverless functions into Kubernetes

Using Helm in advanced requirements is appreciated.

<u>Bonuses:</u>

- Let's Encrypt
- Internal Certificate Authority for securing network communication between services
- cAdvisor
- Build k8s cluster with Terraform
- A staging configuration with Kustomize
- Different ACLs for developer and system administrators
- Deploy Istio on top of networking layer
- Prometheus alerting into Teams, Slack or SMS (many channels depending on criticity?)
- Move secrets into Hashicorp Vault
- Build something cool with Kubernetes operators

Delivery

Please start before the review, a fresh new Kubernetes cluster in the cloud, with many nodes.

During your project presentation, you will deploy the services into the orchestrator using nothing more than “kubectl apply -f ... -f ... -f ...” and “helm …”.

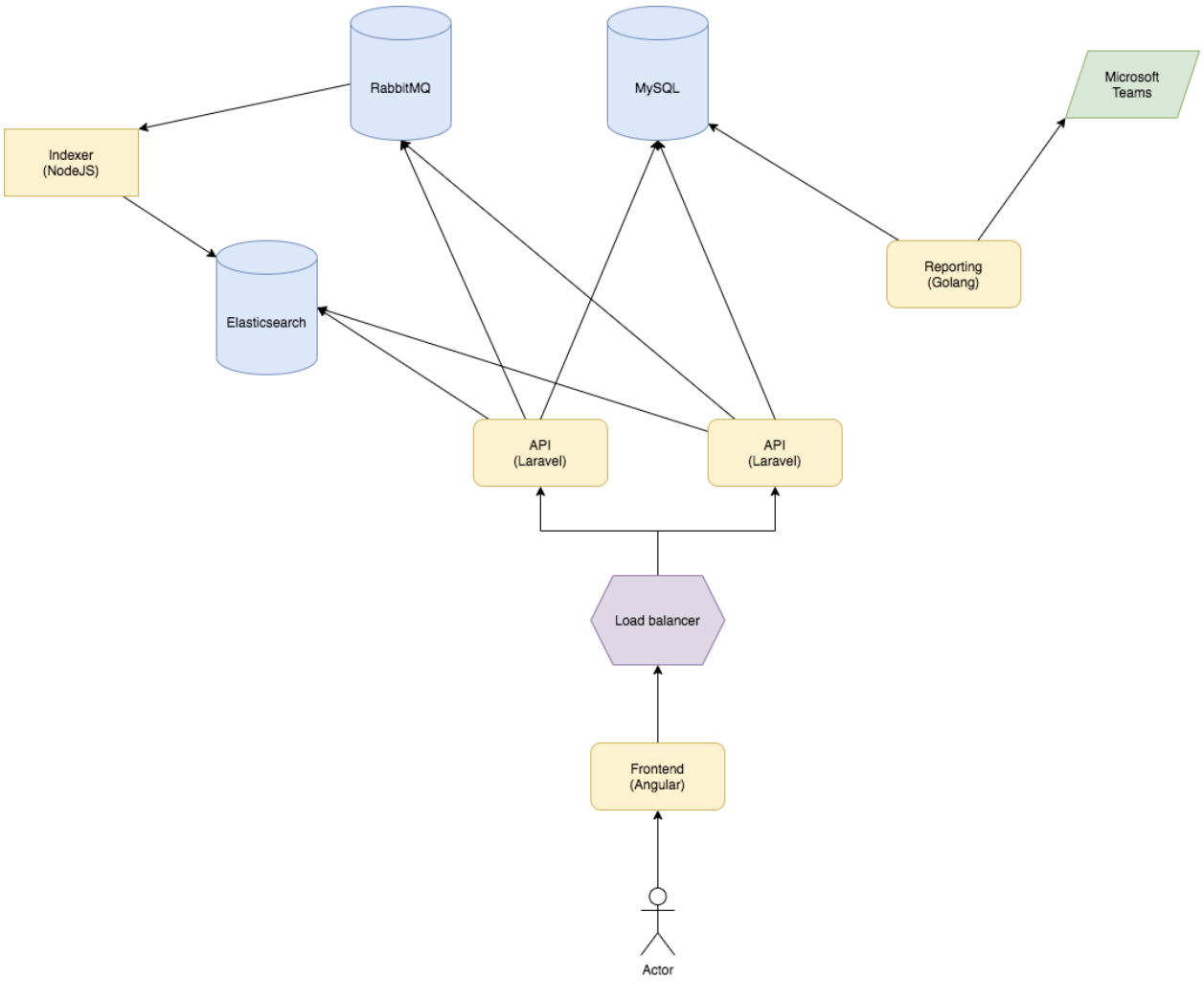
In order to demonstrate some features, such as auto-scaling, prepare some scripts or commands that send a lot of requests to applications.

Feel free to add in the applications code, some memory leaks, loops consuming CPU, or anything that could cause errors and container failure.

Push your configurations into a repo and invite your teacher. Documentation is appreciated.

Appendix

Application architecture



Application screenshot

Welcome

Health


```
{
  "hostname": "Samber-MacBook-Pro.local",
  "mysql": "healthy",
  "products": 33,
  "mysql_migrations": "healthy",
  "elasticsearch": "healthy",
  "amqp": "?",
  "response_time_ms": 99
}
```


Add products

foobar

lorem ipsum

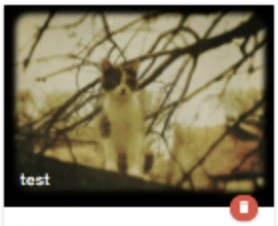
Image path

GET RANDOM IMAGE

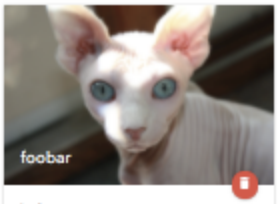
SUBMIT

Products

Keyword

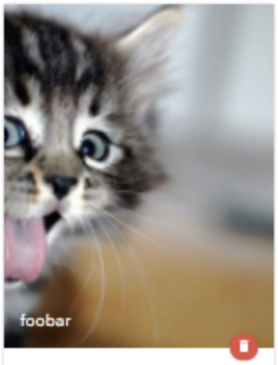


test



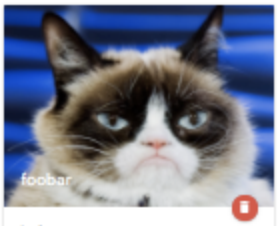
barfoo

barfoo

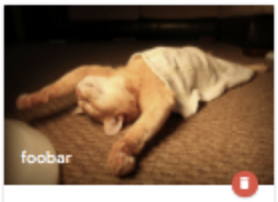


barfoo

barfoo



barfoo



barfoo