T10 - Cloud

T-CLO-902

Kubi

take the helm

0.4.1

{EPITECH.}

Kubi

delivery method: Github
repository name: $CourseCode-$GroupName.git

- The totality of your source files, except all useless files (binary, temp files, obj files,...), must be included in your delivery.
- All the bonus files (including a potential specific Makefile) should be in a directory named *bonus*.

You have to deploy a simple e-commerce administration app in a Kubernetes cluster.

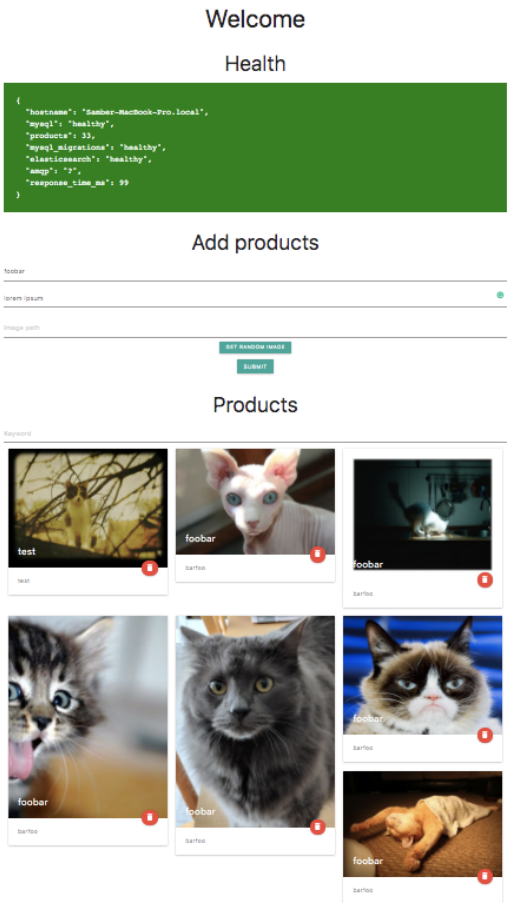
Through this app, a visitor can manage a catalog and search products by keywords.
It has been built in a microservice architecture, composed of:

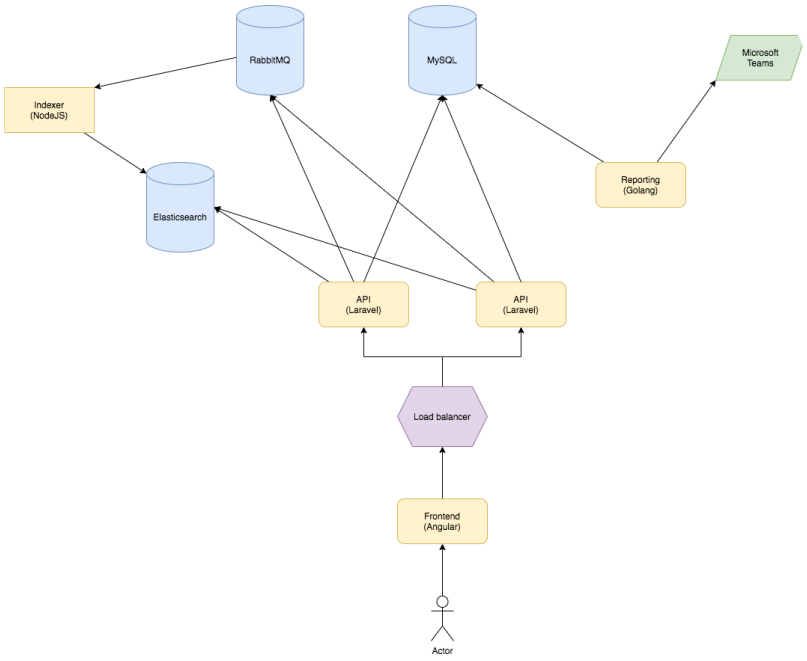
- a frontend application in Angular,
- an API in Laravel (PHP),
- an indexer in Node.JS,
- a reporting job in Go,
- 3 databases (MySQL, RabbitMQ, Elasticsearch).

You may need some additional services for your infrastructure.

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.





You are asked to create or deploy the following services and tools:

- your own **Helm chart for each microservice**; charts must be reusable and customizable in case we need it in the future (different Docker images, environment variables…),
- an internal **load balancer** setup to be the single entrypoint of your infrastructure,
- a **monitoring stack** based on Prometheus and Grafana,
- a **log aggregation**,
- databases, internal load balancer, log aggregation, monitoring must be deployed with Helm public packages,
- docker images, stored on a **private Docker registry** (such as Github).

High-quality and secured services are required:

- the **resource consumption** must be limited,
- databases must be **secured with password**; some custom settings may be set, for better performance,
- passwords must be stored in **Kubernetes secrets**,
- kubernetes resources must have **coherent labels**, in order to filter resources easily,
- 2 **roles** (sysadmin and developer) must be configured with suitable permissions (at least one user per role).

The infra must also respect a high-availability standard:

- your infrastructure must be **resilient**: services should be replicated when possible,
- **auto-scaling rules** must allow your platform to accept a growing number of requests,
- a node failure must **not make any downtime**; a delay in indexation is permitted,
- propose a **zero-downtime deployment** strategy; a demonstration must be prepared.

Delivery

Push your configurations into a repo and invite your teacher.
Documentation is appreciated.
Concerning the final defense of your project:

1. before the presentation, start a fresh new Kubernetes cluster in the cloud, with many nodes,
2. during the presentation, deploy the services into the orchestrator using nothing more than `kubectl apply -f ... -f ... -f ...` and `helm ...`,
3. in order to demonstrate some features, such as auto-scaling, execute in live scripts or commands you wrote beforehand to send lots of requests to applications.
4. demonstrate a full deployment process and a broken deployment with automatic rollback.

Feel free to enrich the applications code with some memory leaks, loops consuming CPU, or anything that could lead to errors and container failure.

Bonuses

You may also want to add extra points, such as:

- an encryption with Let's Encrypt,
- a staging environment, built with Kustomize,
- deploy cAdvisor for better debugging,
- deploy Istio on top of networking layer,
- a continuous integration flow.