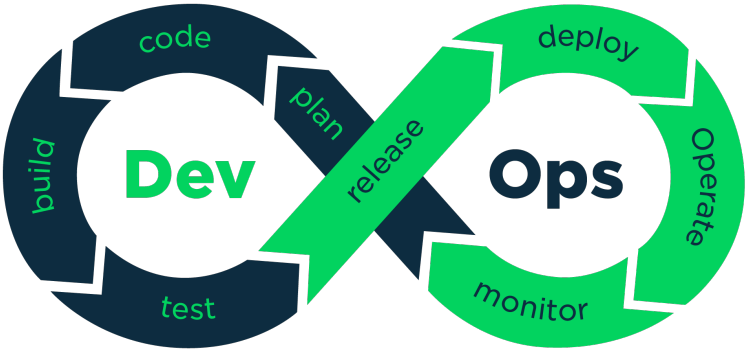
B5 - Advanced DevOps

B-DOP-500

Bernstein

Orchestrate your own symphony of containers



1.2

Bernstein

- The totality of your source files, except all useless files (binary, temp files, obj files,...), must be included in your delivery.

Orchestration is the *arrangement of a piece of music in parts so that it can be played by an orchestra* (from the *Oxford Advanced Learner's Dictionary*).
In other terms, it is the process of organizing the different components of a system in order for them to be (correctly) executed by a generic infrastructure.

Leonard Bernstein was an exceptional composer (his most famous work being the soundtrack of *West Side Story*), pianist, and conductor in his days.

Leonard Bernstein conducting Hector Berlioz's *Symphonie fantastique, Op. 14* with the Orchestre National de France in 1976.

During this project, you are going to become the Leonard Bernstein of containers!

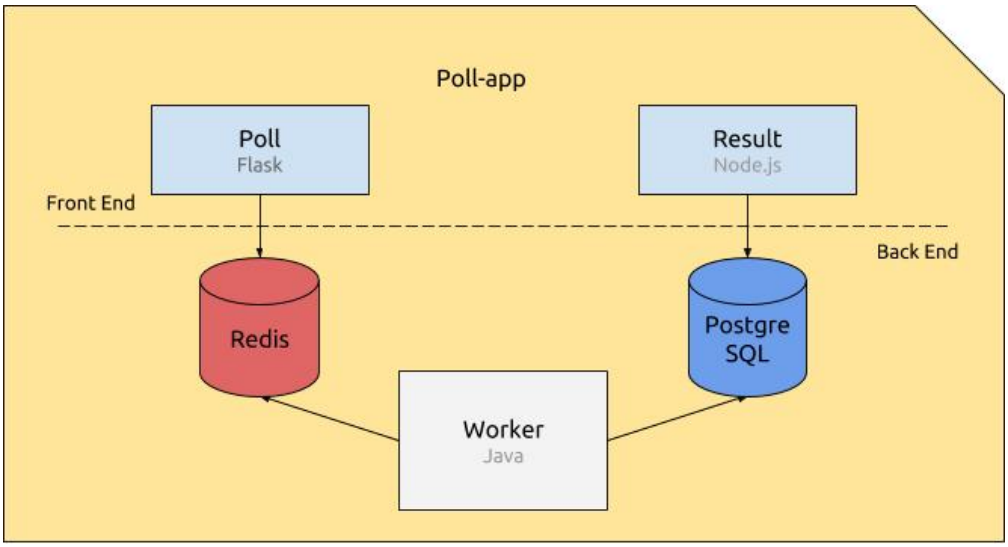
The orchestra you are going to conduct is one of the most popular of its kind: Kubernetes.

Kubernetes is an open-source platform that allows you to manage containerized applications and services, in a scalable and automatic way.

You will deploy an application to a multi-host cluster using Kubernetes, and you will use Traefik as a reverse proxy and load balancer.

The application you will be working on during this project is a simple web poll application.

There are five elements constituting the application:
- ***Poll***, a Flask Python web application that gathers votes and push them into a Redis queue.
- A **Redis queue**, which holds the votes sent by the Poll application, awaiting for them to be consumed by the Worker.
- The ***Worker***, a Java application which consumes the votes being in the Redis queue, and stores them into a PostgreSQL database.
- A **PostgreSQL database**, which (persistently) stores the votes stored by the Worker.
- ***Result***, a Node.js web application that fetches the votes from the database and displays the… well, result. ;)



This looks familiar does not it?

Specifications

You have to define 1 load balancer, 2 databases and 3 services, two of which will be routed using Traefik. You will also have to define a monitoring tool to get information on your containers.

Databases

`redis`:

- Based on `redis:5.0`.
- Namespace: `default`.
- Not replicated.
- Always restarts.
- Exposes port 6379.
- Is not enabled in Traefik.

`postgres`:

- Based on `postgres:12`.
- Namespace: `default`.
- Not replicated.
- Always restarts.
- Exposes port 5432.
- Is not enabled in Traefik.
- Has a persistent volume: `/var/lib/postgresql/data`.
- Environment variables:
    - `POSTGRES_HOST`
    - `POSTGRES_PORT`
    - `POSTGRES_DB`
    - `POSTGRES_USER`
    - `POSTGRES_PASSWORD`

Services

`poll`:

- Based on `epitechcontent/t-dop-600-poll:k8s`.
- Namespace: `default`.
- Replicated: once (== 2 instances).
- Always restarts.
- Memory limit of 128M.
- Exposes port 80.
- Has a Traefik rule matching `poll.dop.io` host and proxying to `poll` service.
- Environment variables:
    - `REDIS_HOST`

`worker`:

- Based on `epitechcontent/t-dop-600-worker:k8s`.
- Namespace: `default`.
- Not replicated.
- Memory limit of 256M.
- Always restarts.
- Is not enabled in Traefik.
- Environment variables:
    - `REDIS_HOST`
    - `POSTGRES_HOST`
    - `POSTGRES_PORT`
    - `POSTGRES_DB`
    - `POSTGRES_USER`
    - `POSTGRES_PASSWORD`

`result`:

- Based on `epitechcontent/t-dop-600-result:k8s`.
- Namespace: `default`.
- Replicated: once (== 2 instances).
- Memory limit of 128M.
- Always restarts.
- Exposes port 80.
- Has a Traefik rule matching `result.dop.io` host and proxying to `result` service.
- Environment variables:
    - `POSTGRES_HOST`
    - `POSTGRES_PORT`
    - `POSTGRES_DB`
    - `POSTGRES_USER`
    - `POSTGRES_PASSWORD`

LOAD BALANCER

`traefik`:

- Based on `traefik:2.7`.
- Namespace: `kube-public`.
- Replicated: once (== 2 instances).
- Always restarts.
- Needs authorization to access Kubernetes internal API.
- Exposes port 80 (HTTP proxy) and 8080 (admin dashboard) into Kubernetes cluster.
- Exposes port 30021 (HTTP proxy) and 30042 (admin dashboard) on host.

MONITORING TOOL

`cadvisor`:

- Based on `gcr.io/cadvisor/cadvisor:latest`.
- Namespace: `kube-system`.
- Scheduled on all nodes.
- Always restarts.
- Exposes port 8080.

In order to improve high availability, replicated services must run on different nodes.

All environment variables must be stored in Kubernetes' `ConfigMap`.

`POSTGRES_USER` and `POSTGRES_PASSWORD` must be stored in Kubernetes' Secrets.

At the end of the project, you should be able to open the `poll` application into your browser:

- Result -> `result.dop.io:30021`
- Poll -> `poll.dop.io:30021`
- Traefik dashboard -> `localhost:30042`

The expected deployment process is the following:

Terminal

```
~/B-DOP-500> kubectl apply -f cadvisor.daemonset.yaml
~/B-DOP-500> kubectl apply -f postgres.secret.yaml \
  -f postgres.configmap.yaml \
  -f postgres.volume.yaml \
  -f postgres.deployment.yaml \
  -f postgres.service.yaml
~/B-DOP-500> kubectl apply -f redis.configmap.yaml \
  -f redis.deployment.yaml \
  -f redis.service.yaml
~/B-DOP-500> kubectl apply -f poll.deployment.yaml \
  -f worker.deployment.yaml \
  -f result.deployment.yaml \
  -f poll.service.yaml \
  -f result.service.yaml \
  -f poll.ingress.yaml \
  -f result.ingress.yaml
~/B-DOP-500> kubectl apply -f traefik.rbac.yaml \
  -f traefik.deployment.yaml \
  -f traefik.service.yaml
~/B-DOP-500> # Create database manually after first deploy
~/B-DOP-500> echo "CREATE TABLE votes \
(id text PRIMARY KEY, vote text NOT NULL);" \
  | kubectl exec -i <postgres-deployment-id> -c <postgres-container-id> \
    - psql -U <username>
~/B-DOP-500> # Adds 2 fake DNS to /etc/hosts
~/B-DOP-500> echo "$(kubectl get nodes -o \
  jsonpath='{ $.items[*].status.addresses[?(@.type=="ExternalIP")].address }') \
  poll.dop.io result.dop.io" \
  | sudo tee -a /etc/hosts
```

Environment

You will need at least 1 Kubernetes master and 2 nodes (workers).
You can run it locally, but it is highly recommended to use a "Kubernetes as a Service" platform.
Examples of such platforms include (but are not limited to) *Amazon Elastic Kubernetes Service*, *Google Kubernetes Engine*, and *Digital Ocean*.

See also the cloud platforms there: `https://education.github.com/pack`.

Installing a full Kubernetes cluster locally is complex. Minikube is also not built for multi-node clusters (**which you *need* for your project**). Take a look at K3s.

TECHNICAL FORMALITIES

Your project will be entirely evaluated with Automated Tests, by analyzing your configuration files.

In order to be correctly evaluated, your repository should at least contain the following files:

Terminal

```
~/B-DOP-500> find
./cadvisor.daemonset.yaml
./poll.deployment.yaml
./poll.ingress.yaml
./poll.service.yaml
./postgres.configmap.yaml
./postgres.deployment.yaml
./postgres.secret.yaml
./postgres.service.yaml
./postgres.volume.yaml
./redis.configmap.yaml
./redis.deployment.yaml
./redis.service.yaml
./result.deployment.yaml
./result.ingress.yaml
./result.service.yaml
./traefik.deployment.yaml
./traefik.rbac.yaml
./traefik.service.yaml
./worker.deployment.yaml
~/B-DOP-500>
```

Read this list carefully and ask yourselves what need to be in each file.