T6 - DevOps

T-DOP-600

DevOps

Kubernetes Clustering and Traefik Proxifying

1.2

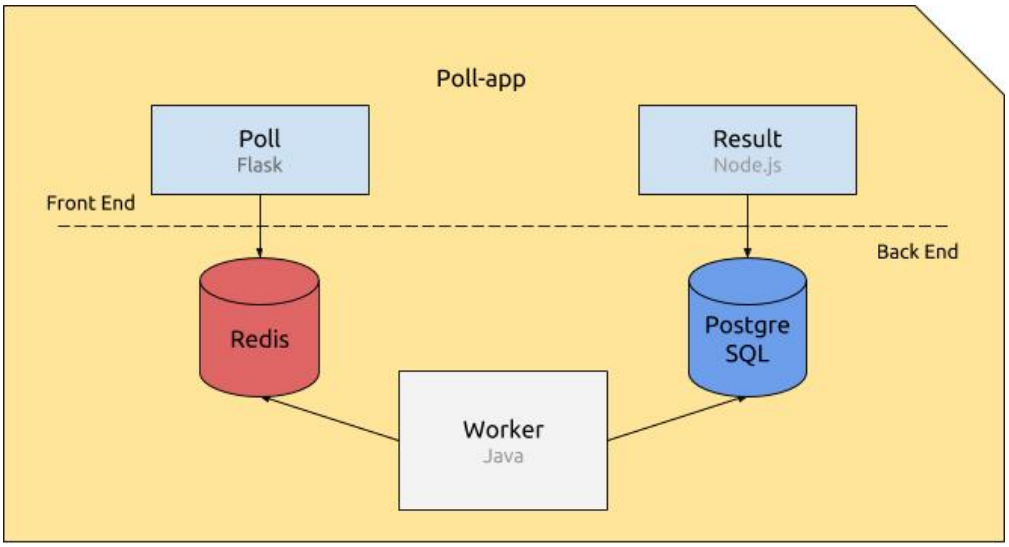
DevOps

repository name: DOP_clusterization_*$ACADEMICYEAR*
repository rights: ramassage-tek

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

This project aims to teach you how to deploy in a cluster using Kubernetes along with how to use Traefik as a reverse proxy and load balancer.

The application you are working on during this project is a simple poll web application. Poll is a Python Flask web application that gathers the votes to push them into a Redis queue. The Java Worker consumes the votes stored in the Redis queue, then pushes it into a PostgreSQL database. Finally, the Node.js Result web application fetches the votes from the DB and displays the result.



If you never used any container orchestrator (such as Kubernetes, Swarm (RIP), Mesos, Nomad, Rancher…): please read the kick-off and work on the bootstrap first.

You are to define 1 load balancer, 2 databases and 3 services, two of which will be routed using Traefik.

redis:

- Based on `redis:5.0`.
- Namespace: default.
- Not replicated.
- Always restarts.
- Exposes port 6379.
- Isn't enabled on Traefik.

postgres:

- Based on `postgres:12`.
- Namespace: default.
- Not replicated.
- Always restarts.
- Exposes port 5432.
- Isn't enabled on Traefik.
- Has a persistant volume: /var/lib/postgresql/data.

- Environment variables:
  - POSTGRES_HOST
  - POSTGRES_PORT
  - POSTGRES_DB
  - POSTGRES_USER
  - POSTGRES_PASSWORD

poll:

- Based on `epitechcontent/t-dop-600-poll:k8s`.
- Namespace: default.
- Replicated: once (== 2 instances).
- Always restarts.
- No more than 128M of memory
- 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.
- No more than 256M of memory
- Always restarts.
- Isn't enabled on 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).
- No more than 128M of memory
- 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

traefik:

- Based on `traefik:1.7`.
- Namespace: kube-public.
- Replicated: once (== 2 instances).
- Always restarts.
- Traefik needs authorization to access Kubernetes internal API.
- Exposes port 80 (http proxy) and 8080 (admin dashboard) into k8s cluster.
- Exposes port 30021 (http proxy) and 30042 (admin dashboard) on host.

cadvisor:

- Based on `google/cadvisor:latest`.

- Namespace: kube-system.
- Scheduled on all nodes.
- Always restarts.
- Exposes port 8080.

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

Common environment variables must be stored in k8s ConfigMap.

POSTGRES_USER and POSTGRES_PASSWORD must be stored in k8s 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`

Your project will be tested with:

```
kubectl apply -f cadvisor.daemonset.yaml

kubectl apply -f postgres.secret.yaml \
              -f postgres.configmap.yaml \
              -f postgres.volume.yaml \
              -f postgres.deployment.yaml \
              -f postgres.service.yaml

kubectl apply -f redis.configmap.yaml \
              -f redis.deployment.yaml \
              -f redis.service.yaml

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

kubectl apply -f traefik.rbac.yaml \
              -f traefik.deployment.yaml \
              -f traefik.service.yaml

# Create database manually after first deploy
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>

# Adds 2 fake DNS to /etc/hosts
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 recommanded to use a "Kubernetes as a Service" platform: EKS, GKE, Digital Ocean…

Cloud platforms: `https://education.github.com/pack`.

Installing a full k8s cluster locally is complex. Minikube is also not built for multi-node clusters. Take a look on k3s.

+ DELIVERY

Your git repository should have at least this following files:

```
.
./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

0 directories, 19 files
```

Read this list of files carefully! Just saying…

+ BONUS

- Build an auto-scaling policy for `result` and `poll` services: from 2 to 10 instances, based on CPU > 70% usage.
- `result` and `poll` containers should die if services are unhealthy (database unaccessible ?).
- Configure a public DNS with LetsEncrypt certificate.
- Explore the “RBAC” user access control.
- Deploy a full Kubernetes cluster on virtual machines, with a quorum of masters
- Deploy a monitoring stack based on Prometheus.
- Deploy Istio on top of k8s networking layer.
- …