TD3
Docker multi-stacks : centralisation des logs et publication des services

Motivation

Lors des TPs précédents, nous utilisé docker-compose pour architecturer **plusieurs** services d'**une** même application (aussi appelée stack).

Dans la vraie vie, il n'est pas rare de faire co-exister **plusieurs** stacks sur **une** même serveur. Ces différentes stacks peuvent correspondre à :
— des applications différentes
— deux instances de la même application pour des clients différents
— une version de production, une version de test/pré-prod
— ...
Dans ce contexte il est particulièrement interessant de pouvoir centraliser les logs

1) Centraliser les logs

2) Publier les services grâce à un edge-router afin d'identifier une stack de façon plus claire que par un numéro de port arbitraire sur localhost.

Documentation utiles

`https://docs.docker.com/build/building/multi-stage/`
`https://docs.docker.com/develop/dev-best-practices/`
`https://docs.docker.com/develop/develop-images/dockerfile_best-practices/`
`https://docs.docker.com/develop/security-best-practices/`

Docker en ligne

Créez vous un compte sur `https://hub.docker.com/signup`
A l'aide de ce compte, connectez vous à la plate-forme `https://labs.play-with-docker.com/`. Vous disposerez de sessions de 4h pour faire vos essais.

1 Centralisation des logs

1.1 Installation de la stack Loki/Grafana

Il existe une solution de centralisation de logs, qui est elle même une application stack, décrite dans le docker-compose ci-après :

```
version: "3"

networks:
  loki:

services:
#loki est un gestionnaire centralisé de logs
  loki:
    image: grafana/loki:2.9.0
    ports:
      - "3100:3100"
    command: -config.file=/etc/loki/local-config.yaml
    networks:
      - loki

  #promtail permet de faire remonter les logs de la machine locale dans le loki
```

```
  promtail:
    image: grafana/promtail:2.9.0
    volumes:
      - /var/log:/var/log
    command: -config.file=/etc/promtail/config.yml
    networks:
      - loki

#grafana est l'interface graphique qui permet d'interroger l'API de loki
  grafana:
    environment:
      - GF_PATHS_PROVISIONING=/etc/grafana/provisioning
      - GF_AUTH_ANONYMOUS_ENABLED=true
      - GF_AUTH_ANONYMOUS_ORG_ROLE=Admin
    entrypoint:
      - sh
      - -euc
      - |
        mkdir -p /etc/grafana/provisioning/datasources
        cat <<EOF > /etc/grafana/provisioning/datasources/ds.yaml
        apiVersion: 1
        datasources:
        - name: Loki
          type: loki
          access: proxy
          orgId: 1
          url: http://loki:3100
          basicAuth: false
          isDefault: true
          version: 1
          editable: false
        EOF
        /run.sh
    image: grafana/grafana:latest
    ports:
      - "3000:3000"
    networks:
      - loki
```

a. Lancez ce compose

b. En lisant le docker-compose, trouver comment accéder a Grafana :

1.2 Centralisation des logs d'une autre stack

Déployer loki/grafana juste pour visualiser les logs de sa machine locale serait quelque peu overkill. Le but du jeu est d'y centraliser les logs de vos autres stacks.

c. Pour chacun des containers de chacune de des stacks de votre serveur (pour le moment uniquement ceux du TP1) ajouter une un champs *logging* pour remonter les logs vers la stack $loki/grafana$.

```
    logging:
      driver: loki
      options:
        loki-url: http://127.0.0.1:3100/loki/api/v1/push
        loki-pipeline-stages: |
          - regex:
              expression: '(level|lvl|severity)=(?P<level>\w+)'
          - labels:
              level:
```

Il est possible que docker vous demande d'installer un plugin

```
$docker plugin install grafana/loki-docker-driver:latest --alias loki
--grant-all-permissions
```

d. Testez
   — Butinez sur le server *app*.
   — Provoquez une erreur 404.
   — Retrouvez cette 404 à l'aide de l'interface Grafana
   — Quelle serait la suite de commandes pour retrouver le log cette erreur 404 sans la stack $loki/grafana$?

2 Mise en place d'un Edge-Router pour publier les services

2.1 Installation de la stack Taefik

e. Créer un nouveau docker-compose.yaml pour Traefik :

```
version: '3'
services:
  traefik:
    image: traefik:v2.10
    command:
      - "--api.insecure=true"
      - "--providers.docker=true"
      - "--providers.docker.exposedbydefault=false"
      - "--entrypoints.web.address=:80"
    ports:
      - "80:80"
      - "8080:8080"
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
    networks:
      - external

networks:
  external:
    external: true
```

f. Créer le réseau external avec la commande *docker* (RTFM !)

g. Si l'une de vos stacks expose les ports 80 et/ou 8080, libérez le(s) (en changeant pour un autre port dans votre compose).

h. le cas échéant, relancez

```
$  docker compose up -d
```

i. Une fois ces modifications réalisées, lancez le docker-compose de traefik et vérifiez que traefik répond bien sur les ports 80 et 8080.

2.2 Configuration d'un service / stack

Treafik utilise les labels docker pour découvrir les services automatiquement,

j. Ajouter ces label dans les composes de vos stacks. Seuls les image qui exposent un port sont concernés :

```
    labels:
      - "traefik.enable=true"
      - "traefik.http.routers.{mon_service}-web.rule=Host(`{URL.td.anthonymoll.fr}`)"
      - "traefik.http.routers.{mon_service}-web.entrypoints=web"
      - "traefik.http.services.{mon_service}.loadbalancer.server.port={port_du_service}"
```

où :

mon_service le nom du service (différent pour chaque service)

URL l'adresse qui sera accessible via `http://URL.td.anthonymoll.fr`

port_du_service est le port exposé par le docker

Ici, vous utilisez un DNS externe (celui du chargé de TD) . Sur vous avez accès à votre propre DNS ou plus simplement au fichier $/etc/hosts$, vous pouvez y ajouter les lignes :
127.0.0.1 URL 127.0.0.1 URL2

k. Pensez à ajouter le réseau "external" dans vos compose de stacks pour permettre au treafik de communiquer avec elles. Inspirez vous de la syntaxe du compose de traefik.

3 Mise en place d'une version prod et d'une version dev d'une même application

Reprenez l'application du TP1

l. Faites co-exister :
— la version php7.2 + mariadb
— la version php8.2 + postgresql.

m. La version 7.2 sera accessible via l'url `http://prod` et la 8.2 via l'url `http://latest`

n. L'intégralité des logs seront remontés dans loki/grafana

o. Grafana sera rendu accessible à l'url `http://grafana`