Exemples INFO910

Docker

Application Vue

```
# On choisit une version antérieur de node en raison de problème de compatibilité
avec certains packages de vue.js
FROM node:16-alpine

# installe un simple serveur http pour servir un contenu statique
RUN npm install -g http-server

# définit le dossier 'app' comme dossier de travail
WORKDIR /app

# copie 'package.json' et 'package-lock.json' (si disponible)
COPY package*.json ./

# installe les dépendances du projet
RUN npm install

# copie les fichiers et dossiers du projet dans le dossier de travail (par exemple
: le dossier 'app')
COPY . .

# construit l'app pour la production en la minifiant
RUN npm run build

EXPOSE 8080
CMD [ "http-server", "dist" ]
```

Application Node

```
# On travaille avec la dernière version de node
FROM node:lts-alpine


# définit le dossier 'app' comme dossier de travail
WORKDIR /app

# copie 'package.json' et 'package-lock.json' (si disponible)
COPY package*.json ./

# installe les dépendances du projet
```

```
RUN npm install

# copie les fichiers et dossiers du projet dans le dossier de travail (par exemple
: le dossier 'app')
COPY . .

# On expose le port 3000 pour le serveur web et le port 9000 pour le serveur de
socket
EXPOSE 3000
EXPOSE 9000

# On lance le serveur web et le serveur de socket
CMD [ "node", "app.js" ]
```

Docker compose du tp

```
version: "3"
services:
  # Définition du conteneur servant le front (application Vue.js)
  front:
    build:
      context: ./Client/app
      dockerfile: Dockerfile
    image: appli/vue
    container_name: front
    ports:
      - "8080:8080"
    # On attend que le conteneur back soit prêt avant de démarrer le front
    depends_on:
      - back
  # Définition du conteneur servant le back (application Node.js pour l'API REST
et le serveur WebSocket)
  back:
    build:
      context: ./Server
      dockerfile: Dockerfile
    image: appli/back
    container_name: back
    ports:
      - "3000:3000"
      - "9000:9000"
    # On attend que le conteneur MongoDB soit prêt avant de démarrer le back
    depends_on:
      - mongo
    # Permet l'accès à la bdd depuis le code sans avoir à spécifier l'adresse de
la bdd
    external_links:
      - mongo

  # Définition du conteneur MongoDB, utilisant une image déjà prête de monogdb
  mongo:
```

```
    image: mongo
    container_name: mongo
    # La base de données se relancera en cas de crash
    restart: always
    ports:
      - "27017:27017"
    volumes:
      - ./data:/data/db
```

Application php

```
FROM php:7.1-apache-buster

RUN cp "$PHP_INI_DIR/php.ini-development" "$PHP_INI_DIR/php.ini"

# install mysql pdo extension
RUN docker-php-ext-install pdo_mysql

COPY chat_php  /var/www/html
```

Docker compose exemple chat php

```
version: "3.9"

services:
  mysql:
    image: "mysql:8.0.11"
    environment:
    #  MYSQL_ROOT_PASSWORD: "password"
      MYSQL_ROOT_PASSWORD_FILE: "/run/secrets/mysql/password"
    secrets:
      - source: mysql
        target: mysql/password
    command: --default-authentication-plugin=mysql_native_password
    volumes:
      - "./mysql_data:/var/lib/mysql"
    configs:
      - source: chat_init
        target: /docker-entrypoint-initdb.d/chat.sql
    networks:
      - demonet

  chat:
    image: "registry.gitlab.com/docker-exemple/chat_docker/chat:master"
    build:
      context: php
      dockerfile: Dockerfile
    secrets:
      - source: mysql
```

```
        target: mysql/password
    ports:
      - 9000:80
    networks:
      - demonet
    depends_on:
      - mysql

networks:
  demonet:

secrets:
  mysql:
    file: ./passwd

configs:
  chat_init:
    file: ./chat.mysql
```

Kubernetes

Exemple Ingress

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: chat
  labels:
    name: chatdemo
spec:
  rules:
  - host: "chat.k8s.learninglab.eu"
    http:
      paths:
      - pathType: Prefix
        path: "/"
        backend:
          service:
            name: chat
            port:
              number: 80
```

Exemple configmap

```
apiVersion: v1
data:
  init-db.sql: "-- MySQL dump 10.13  Distrib 8.0.0-dmr, for Linux (x86_64)\r\n--
\r\n--
    Host: localhost    Database: chat\r\n-- ------------------------------------
----------------\r\n--
    Server version\t8.0.0-dmr\r\n\r\n/*!40101 SET
@OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT
    */;\r\n/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS
*/;\r\n/*!40101
    SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;\r\n/*!40101 SET NAMES
    utf8 */;\r\n/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;\r\n/*!40103 SET
TIME_ZONE='+00:00'
    */;\r\n/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0
*/;\r\n/*!40014
    SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0
*/;\r\n/*!40101
    SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;\r\n/*!40111
    SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;\r\n\r\n--\r\n-- Current
Database:
    `chat`\r\n--\r\n\r\nCREATE DATABASE /*!32312 IF NOT EXISTS*/ `chat` /*!40100
DEFAULT
    CHARACTER SET latin1 */;\r\n\r\nUSE `chat`;\r\n\r\n--\r\n-- Table structure
for
    table `chat`\r\n--\r\n\r\nDROP TABLE IF EXISTS `chat`;\r\n/*!40101 SET
@saved_cs_client
    \    = @@character_set_client */;\r\n/*!40101 SET character_set_client = utf8
    */;\r\nCREATE TABLE `chat` (\r\n  `name` varchar(50) COLLATE
utf8mb4_0900_ai_ci
    DEFAULT NULL,\r\n  `description` text COLLATE utf8mb4_0900_ai_ci,\r\n  `id`
bigint(20)
    NOT NULL AUTO_INCREMENT,\r\n  PRIMARY KEY (`id`)\r\n) ENGINE=InnoDB
AUTO_INCREMENT=2
    DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;\r\n/*!40101 SET
character_set_client
    = @saved_cs_client */;\r\n\r\n--\r\n-- Dumping data for table `chat`\r\n--
\r\n\r\nLOCK
    TABLES `chat` WRITE;\r\n/*!40000 ALTER TABLE `chat` DISABLE KEYS */;\r\nINSERT
    INTO `chat` VALUES ('sortie de ski',NULL,1);\r\n/*!40000 ALTER TABLE `chat`
ENABLE
    KEYS */;\r\nUNLOCK TABLES;\r\n\r\n--\r\n-- Table structure for table
`message`\r\n--\r\n\r\nDROP
    TABLE IF EXISTS `message`;\r\n/*!40101 SET @saved_cs_client     =
@@character_set_client
    */;\r\n/*!40101 SET character_set_client = utf8 */;\r\nCREATE TABLE `message`
    (\r\n  `id` bigint(20) NOT NULL AUTO_INCREMENT,\r\n  `pseudo` varchar(30)
COLLATE
    utf8mb4_0900_ai_ci DEFAULT NULL,\r\n  `date_mesg` datetime DEFAULT
CURRENT_TIMESTAMP,\r\n
    \ `content` text COLLATE utf8mb4_0900_ai_ci,\r\n  `chat` bigint(20) NOT
NULL,\r\n
    \ PRIMARY KEY (`id`)\r\n) ENGINE=InnoDB AUTO_INCREMENT=10 DEFAULT
CHARSET=utf8mb4
```

```
    COLLATE=utf8mb4_0900_ai_ci;\r\n/*!40101 SET character_set_client =
@saved_cs_client
    */;\r\n\r\n--\r\n-- Dumping data for table `message`\r\n--\r\n\r\nLOCK TABLES
    `message` WRITE;\r\n/*!40000 ALTER TABLE `message` DISABLE KEYS */;\r\nINSERT
    INTO `message` VALUES (1,'Albert','2020-11-18 10:32:45','Salut',1),
(2,'Joe','2020-11-18
    10:33:05','Hi',1),(3,'Albert','2020-11-18 10:33:31','C\\'est toujours bon pour
    samedi ?',1),(4,'Pierre','2020-11-18 10:34:26','Oui, il parait qu\\'il y a de
    la neige...',1),(5,'Pauline','2020-11-18 10:08:11','ils annoncent 1m80 en haut
    des pistes... et pres d\\'1m en bas',1),(6,'Martine','2020-11-18
10:39:14','Super
    !',1),(7,'Joe','2020-11-18 10:53:28','mio, je ne peux pas : je suis de corvee
    chez ma belle-mere',1),(8,'Pierre','2020-11-18 10:54:09','ca c\\'est vraiment
    la poisse',1),(9,'Albert','2020-11-18 11:12:47','bye',1);\r\n/*!40000 ALTER
TABLE
    `message` ENABLE KEYS */;\r\nUNLOCK TABLES;\r\n/*!40103 SET
TIME_ZONE=@OLD_TIME_ZONE
    */;\r\n\r\n/*!40101 SET SQL_MODE=@OLD_SQL_MODE */;\r\n/*!40014 SET
FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS
    */;\r\n/*!40014 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS */;\r\n/*!40101 SET
CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT
    */;\r\n/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS
*/;\r\n/*!40101
    SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;\r\n/*!40111 SET
SQL_NOTES=@OLD_SQL_NOTES
    */;\r\n\r\n-- Dump completed on 2020-11-18 23:15:54\r\n"
kind: ConfigMap
metadata:
  name: init-db
  labels:
    app: chatdemo
    role: bd
```

Attribution d'une adresse Ip externe au service

Si un *load balancer* est accessible à partir du cluster (le *load balancer* est généralement fournit par l'hébergeur du cluster kubernetes) et qu'il reste des adresses IP disponibles, il est possible d'attribuer une adresse IP externes aux services (en utilisant un service de type `LoadBalancer`).
Un exemple est donné dans la dernière version de l'application pour le service du chat : k8s/chat-svc.yaml.
Sur le cluster utilisé pour le déploiement, cela rend le service de chat php accessible sur l'adresse IP *163.172.192.95* (http://163.172.192.95).

Utilisation d'un `Ingress`

Dans le cas de services de type web, autre solution pour exposer un service vers l'extérieur est de l'exposer au moyen d'une ressource de type `Ingress`, ce qui suppos qu'un `Ingress Controler` ait été préalablement déployé sur le cluster.

Si c'est le cas les ressources de type `Ingress` permettent de mettre en place des règles de routages pour les requêtes entrantes de l'IP associés à l'`ingress controler`.

Ces règles permettent de rediriger les requêtes entrantes, en fonction de nom de l'hôte utilisé dans les requêtes entrantes ou des préfixes des requêtes http, vers un service ou un autre.

Un exemple d'`ingress` est donné dans la dernière version de l'application pour le service du chat : k8s/ingress.yml. Il redirige toutes les requêtes envoyées sur l'hôte http://chat.k8s.learninglab.eu vers le service correspondant au chat php.

Gestion de la montée en charge de l'application

Dans la première version de l'application (branche feature/v1), à cause de l'utilisation des sessions php, il n'était pas possible d'augmenter le nombre de pods de la partie web/php pour répondre à une augmentation de la charge de l'application.

Dans cette seconde version, un serveur Redis est utilisé pour gérer les sessions php.
Cela permet, puisque la partie web/php est maintenant sans état, de gérer les variations de la charge de travail au niveau de la partie web/php en augmentant ou en diminuant le nombre de pods.

Si l'addon metrics-server (minikube / microk8s) ou l'application metrics-server est activée sur le cluster, alors on pourra également utiliser la fonctionnalité de mise à l'échelle horizontale pour adapter le nombre de pods (de la partie php) à la charge travail.

La mise à l'échelle horizontale peut être mise en place soit en utilisant une commande `kubectl` :

```
kubectl autoscale --namespace chatdemo deployment chat --max=10 --min=1 --cpu-
percent=80
```

soit en utilisant une ressource k8s de type `HorizontalPodAutoscaler`

Dans la configuration utilisée pour l'exemple, autoscale.yaml, le nombre de pods variera entre 1 et 10 en essayant de cibler, pour les pods de la partie php, une utilisation moyenne de 80% de le demande (*requests.cpu*) CPU définie dans le déploiement de la partie web (k8s/chat-dep.yaml).

Pour voir fonctionner la mise à l'échelle, on peut charger artificiellement la partie web/php en utilisant par exemple l'utilitaire *apache bench* (https://httpd.apache.org/docs/2.4/fr/programs/ab.html).

La commande suivante :

```
kubectl run test --rm -i -n chatdemo --image=stalb/ab -- ab -c 40 -n 100000
http://chat/chat.php?pseudo=Joe\&chat=1
```

permettra de d'exécuter 100 000 requêtes (avec 40 requêtes en parallèles), ce qui devrait suffisamment charger le serveur pour provoquer, au bout de quelques minutes, une augmentation du nombre de pods de la partie web.

Une fois toutes les requêtes effectuées, la diminution de la charge de travail devrait également entraîner à terme une diminution du nombre de pods (à noter que la descente est plus lente que la montée).