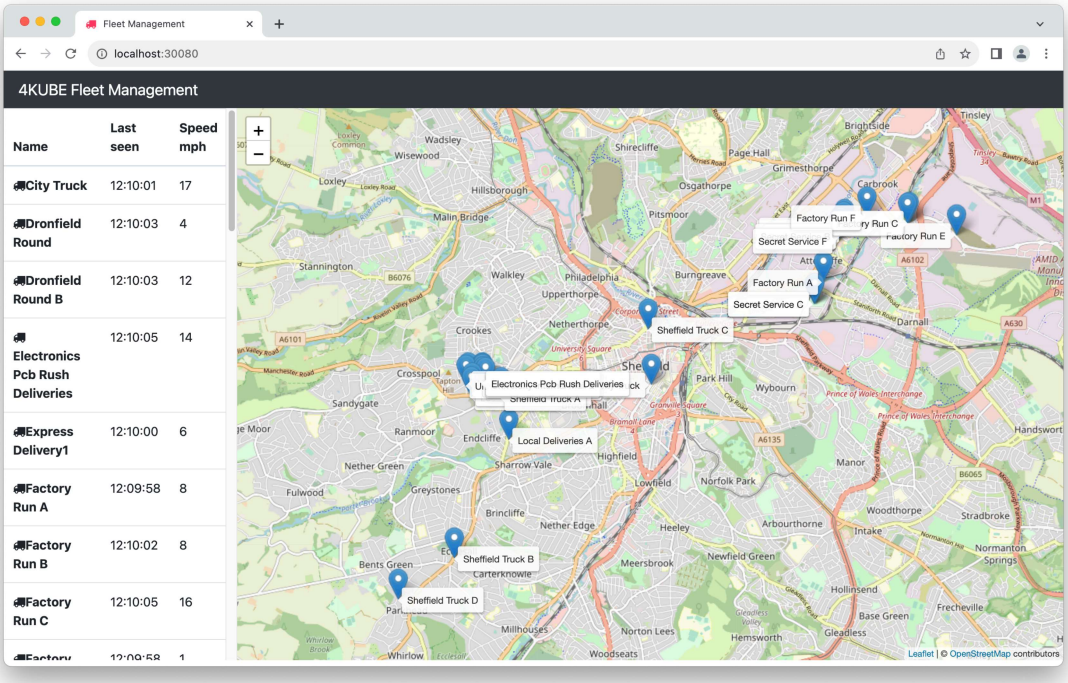
Mini-projet [Sujet]

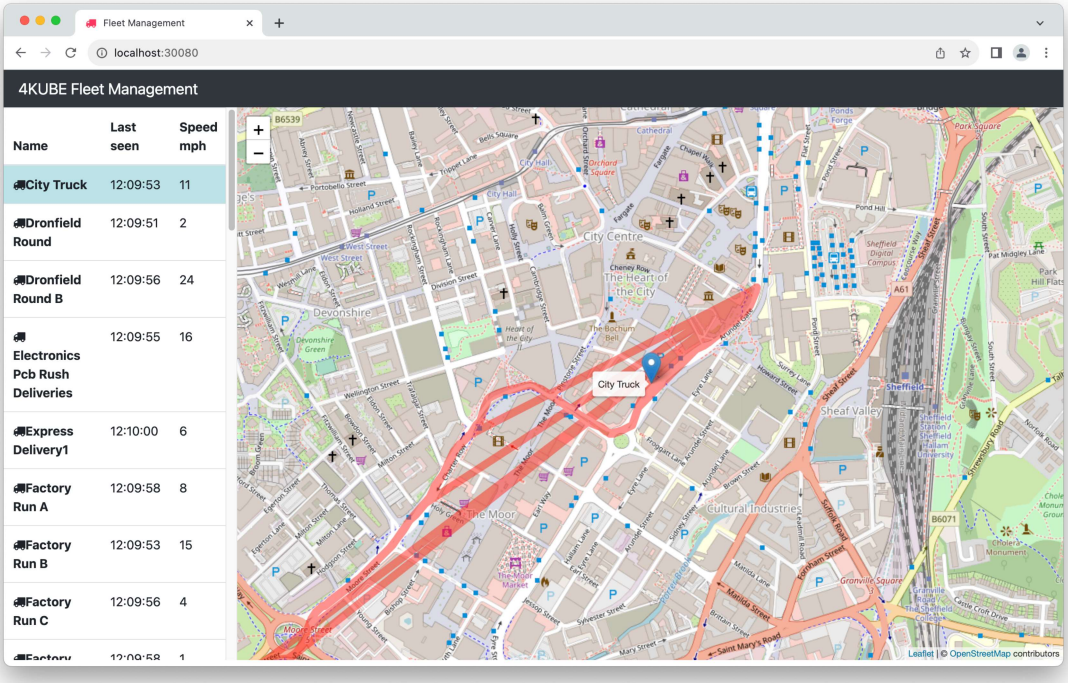
Ouvert le : mercredi 20 novembre 2024, 16:00
À rendre : mercredi 11 décembre 2024, 23:59

Ce mini-projet est à réaliser par groupes de 2 à 3 étudiants en 2 semaines.

Pour ce mini-projet, on vous donne une application distribuée permettant de suivre en temps réel une flotte de véhicules effectuant des livraisons. Voici comment elle se manifeste :



Chaque épingle représente un véhicule. Celles-ci changent de position sans avoir besoin de rafraîchir la page. En cliquant sur l'un des véhicules de la liste de droite, on peut accéder à son trajet :



Cette application distribuée est composée des éléments suivants :

- **fleetman-position-simulator** : une application Spring Boot émettant en continu des positions fictives de véhicules.
- **fleetman-queue** : une queue Apache ActiveMQ qui reçoit puis transmet ces positions.
- **fleetman-position-tracker** : une application Spring Boot qui consomme ces positions reçues pour les stocker dans une base de données MongoDB. Elles sont ensuite disponibles via une API RESTful.
- **fleetman-mongo** : instance de la base de données MongoDB.
- **fleetman-api-gateway** : une *API Gateway* servant de point d'entrée pour l'application web
- **fleetman-web-app** : l'application web présentée précédemment.

Les développeurs de l'application vous fournissent le fichier Docker Compose qu'ils ont utilisé localement :

```
services:
  fleetman-queue:
    image: supinfo4kube/queue:1.0.1
    ports:
      - 8161:8161
      - 61616:61616
    networks:
      - fleetman

  fleetman-position-simulator:
    image: supinfo4kube/position-simulator:1.0.1
    depends_on:
      - fleetman-queue
    environment:
      - SPRING_PROFILES_ACTIVE=local-microservice
    networks:
      - fleetman

  fleetman-position-tracker:
    image: supinfo4kube/position-tracker:1.0.1
    ports:
      - 30010:8080
    environment:
      - SPRING_PROFILES_ACTIVE=local-microservice
    networks:
      - fleetman

  fleetman-api-gateway:
    image: supinfo4kube/api-gateway:1.0.1
    ports:
      - 30020:8080
    environment:
      - SPRING_PROFILES_ACTIVE=local-microservice
    networks:
      - fleetman

  fleetman-webapp:
    image: supinfo4kube/web-app:1.0.0-dockercompose
    ports:
      - 30080:80
    depends_on:
      - fleetman-api-gateway
    networks:
      - fleetman

  fleetman-mongodb:
    image: mongo:3.6.23
    ports:
      - 27017:27017
    volumes:
      - mongo-data:/data/db
    networks:
      - fleetman

networks:
  fleetman:

volumes:
  mongo-data:
```

Notes :

- *V*ous devrez utiliser le profil Spring "**production-microservice**" plutôt que "local-microservice" (à remplacer dans SPRING_PROFILES_ACTIVE) pour que les applications puissent communiquer entre elles une fois dans Kubernetes.
- On changeera le *tag* de l'image supinfo4kube/web-app de 1.0.0-dockercompose à **1.0.0** une fois dans Kubernetes.
- Il se peut que les positions ne s'affichent pas dans l'interface web. Pour corriger le problème, vous pouvez redémarrer fleetman-queue et attendre que les positions s'affichent à nouveau.

Votre rôle est de déployer cette application distribuée sur un cluster Kubernetes. Pour ce faire, vous vous utiliserez les informations présentes dans le fichier Docker Compose et passerez par les étapes suivantes :

1) Créer un déploiement pour chaque conteneur.

2) Créer un service pour chaque déploiement. Vous veillerez à utiliser un service interne ou externe selon ce qui est le plus approprié.

3) Utiliser un volume pour la base de données.

Pour des raisons de simplicité, on se contentera d'accéder directement au service trucks-web-app sans passer par un *ingress controller*.

Enfin, ce mini-projet inclut une partie de recherche. En effet, vous devrez déployer votre application sur un cluster Kubernetes autre que celui de Docker Desktop (ou Minikube). Ce cluster devra comporter 1 nœud master et 2 nœuds worker. Vous pouvez utiliser l'option d'installation de votre choix : kubeadm, kops, kubespray… Pour des raisons de commodité, ce cluster n'est pas à rendre mais vous devrez rédiger un document qui décrit le processus de mise en place de ce cluster.

Livrable

Un fichier .zip comportant (aucun autre format ne pourra être reçu) :

- Vos manifestes Kubernetes.
- Un document texte dans le format de votre choix (.md, .docx, ou .pdf) reprenant toutes les étapes nécessaires pour bâtir votre cluster. Vous pouvez également y ajouter les informations que vous jugerez indispensables pour le correcteur.

Toutes les ressources sont permises, mais on ne trompera en aucun cas le correcteur avec du code qui n'est pas le vôtre.

Barème (40 points)

- Un déploiement et un service existent pour trucks-position-simulator : **2 points**
- Un déploiement et un service existent pour trucks-queue : **2 points**
- Un déploiement et un service existent pour trucks-position-tracker : **2 points**
- Un déploiement et un service existent pour trucks-api-gateway : **2 points**
- Un déploiement et un service existent pour trucks-web-app : **2 points**
- La base de données MongoDB est présente et persistée : **7 points**
- Les composants du projet sont isolés de manière appropriés : **3 points**
- Un document prouve qu'un cluster Kubernetes a été monté : **13 points**
- Les instructions fournies sont claires et concises : **3 points**
- L'application reste disponible si l'un des deux nœuds worker est en échec : **4 points**

Ajouter un travail

Statut de remise

Statut des travaux remis	Aucun devoir n'a encore été remis
Statut de l'évaluation	Non évalué
Temps restant	2 jours 11 heures restants
Dernière modification	-

Commentaires	▸ Commentaires (0)

Contactez-nous

Suivez-nous

Contacter l'assistance du site

Connecté sous le nom « Martiale Mogue-tueche-epse-deumo » (Déconnexion)