Applications infonuagiques natives et DevOps
(GLO-4008/GLO-7008)
Mohamed Aymen Saied

Travail pratique 2

Énoncé

Consignes de remise

Nous vous demandons de compresser et de déposer votre soumission en une seule archive (**incluant le dossier** `.git`) sur la boite de dépôt. De plus, Nous vous demanderons de remplir le fichier `submission.md` avec tout commentaire ou directive nécessaire au fonctionnement de votre solution.

Votre soumission devrait idéalement être fonctionnelle par défaut, sans nécessiter aucune intervention supplémentaire de la part du correcteur. Dans le cas contraire, des pénalités pourraient être appliquées (voir barème). Les directives additionnelles en liens avec les fonctionnalités avancées n'engendreront pas de pénalité si justifiable tant que celles-ci sont raisonnables et justifiées.

Il vous faudra aussi remplir le fichier `submission.json` avec les informations spécifiées préremplies (nom, NIP, FA, etc.). Référez-vous à la section dédiée pour obtenir les codes des fonctionnalités avancées.

Table de matières

1 Pré-requis

Pour ce travail, vous aurez besoin d'un cluster Kubernetes supportant les Ingress Controllers et éventuellement les Services Mesh. Nous présumons que vous utilisez l'environnement local fourni ou un cluster local Kind.

1.1 Kind

Dans le cas d'utilisation de Kind, assurez-vous d'utiliser la configuration de cluster présente sur le projet initial fourni (fichier `cluster.yaml`) en utilisant la commande lors de l'initialisation de votre cluster:

```
# dans le répertoire du projet de base
$ kind create cluster --config cluster.yaml
```

1.2 Registre d'image

Vous aurez besoin d'héberger les images des différents microservices dans un registre. Ces images doivent être ***publiques*** et accessibles par les correcteurs. Vous pouvez vous créer un compte DockerHub à cet effet. Cette plateforme offre un nombre illimité d'images publiques.

1.3 Format de la Soumission

Toutes les ressources définies doivent se trouver le dossier `./submission`. La configuration de votre système ne devrait pas nécessiter plus que l'exécution d'une commande `kubectl apply -f ./submission`

Vous pouvez valider cela avant de finaliser votre remise grâce à *git*, vous n'êtes pas supposé avoir de changement hors des deux fichiers et du dossier *submission*:

```
$ git status
Your branch is up to date with 'origin/main'.
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   submission.md       <========
        modified:   submission.json     <========
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        submission/                     <========
```

2 Système: *Sentence Analyzer*

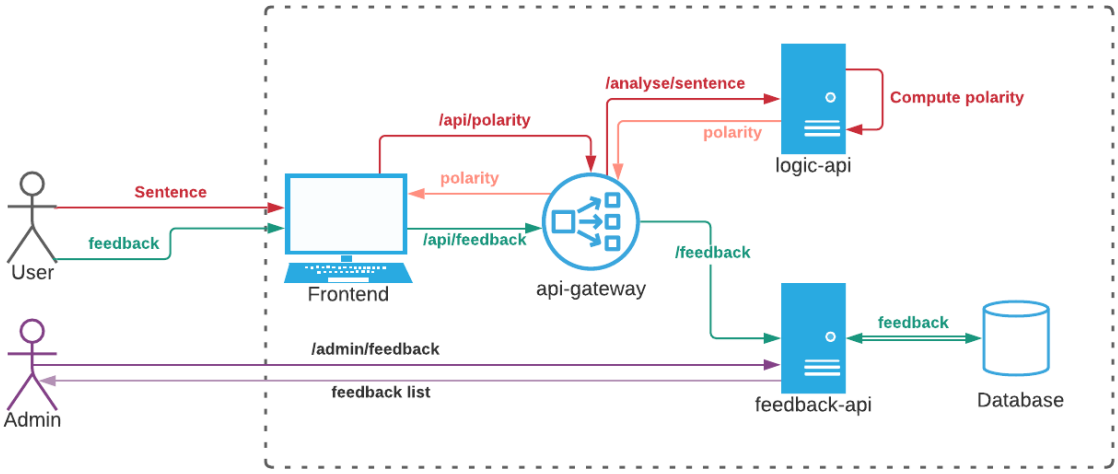
2.1 Description de l'application

Sentence Analyzer est une application très simple exposant une interface web permettant l'entrée d'une phrase (*sentence*) dont la polarité (*Polarity*) est analysée pour déterminer si celle-ci a une tonalité positive ou négative.
La polarité est représentée par un nombre décimal dans **l'intervalle [-1, 1]**.
Par exemple, `I hate it` étant une phrase à tonalité négative, retourne un score de -0.8. À l'inverse, la phrase `I love it` retourne un score de 0.5.

Suite à l'obtention d'un score, il est possible à l'utilisateur de répondre un sondage de rétroaction pour informer le système de la validité du score attribué (Valide/Invalide). Cette rétroaction est ensuite stockée dans une base de donnée pour permettre une amélioration continue du service.

De plus, les administrateurs peuvent obtenir la liste de ces rétroactions en communiquant directement avec le service `feedback-api` exposé à l'extérieur du cluster au travers d'une route spécifique (voir plus bas).

2.2 Diagramme des interactions



2.3 Description des services

Service	Port	Endpoints	Requests	Destination
api-gateway	8080	*POST* **/polarity** *POST* **/feedback**	*POST* **/analyse/sentence** *POST* **/feedback**	*logic-api* *feedback-api*
logic-api	5000	*POST* **/analyse/sentence**	NA	NA
feedback-api	9000	*POST* **/feedback** *GET* **/feedback**	*insert* *Fetch all*	Database (SQLite)
frontend	80	*GET /*	POST **/api/polarity** POST **/api/feedback**	*api-gateway*

2.3.1 NOTES IMPORTANTES

1. `feedback-api` expose la route `GET /feedback`. Or, lors de l'interaction avec ce service, l'administrateur passe par la route `GET /admin/feedback`.
2. L'interface web (`frontend`) communique au travers de routes préfixées par `/api` (i.e `/api/polarity`)
3. Il vous est interdit de faire quelconques modifications directement aux services.
4. Il vous faut trouver une implémentation permettant une gestion des communications entre ces différentes routes telles que défini (voir diagramme d'interactions) sous peine de pénalités.

3 Travail demandé

La tâche principale que vous aurez à effectuer est de définir les manifestes de ressources Kubernetes pour ce système en respectant les critères suivants pour chaque composante.

3.1 *api-gateway*

3.1.1 Critères d'acceptation

- 2 Replicas
- LivenessProbe HTTP
- Exposé sous le préfixe `/api` (rappel)

3.2 *logic-api*

3.2.1 Critères d'acceptation

- 2 Replicas
- LivenessProbe HTTP

3.3 *feedback-api*

3.3.1 Critères d'acceptation

- 2 Replicas
- LivenessProbe HTTP
- **Persistance** des résultats de sondages utilisant une base de donnée *SQLite*
- Route administrateur (`/admin`) pour obtenir tous les feedbacks stockés (`/admin/feedback`) (non pas sécurisé par défaut)

Remarque: La configuration de l'Ingress de la route `/admin` ne devrait pas être spécifique au service `feedback-api`. En effet, nous vous demandons de définir un Ingress générique qui pourrait être étendu dans l'éventualité d'ajout de nouvelles routes (i.e `/admin/polarity`).

3.4 Interface web (*frontend*)

3.4.1 Critères d'acceptation

- 1 Replica
- Exposé à l'extérieur du cluster sous la route `/`

3.4.2 Ressources attendues

- *Deployment*
- *Service*
- *Ingress*

3.5 Fonctionnalités avancées

Voir section dédiée.

4 Barème

4.1 Fonctionnalités de base (70%)

- `GET /` retourne l'interface web = **10%**
- Soumission d'une *Sentence* pour analyse fonctionnelle = **10%**
- Retour de la polarité suite à une soumission = **10%**
- Soumission d'un feedback suite à une soumission = **10%**
- Stockage des résultats des sondages dans la persistance SQLite = **10%**
- Obtenir la liste des résultats de sondages grâce à une requête `GET /admin/feedback` = **10%**
- Pénalités pour non-respect de spécificités et/ou des critères de qualité*** (i.e ingress admin non générique, surexposition de services...) = **10%**

Remarque: Nous nous réservons le droit de juger de ce qui se mérite ou non une pénalité et de juger du poids de celle-ci. Utilisez votre bon-sens lors de l'exécution du travail. En cas de doute, n'hésitez pas à poser la question lors d'un laboratoire ou sur le forum.

4.2 Fonctionnalités avancées (30%)

Pour obtenir les derniers **30%**, il vous faudra sélectionner dans la liste suivante des fonctionnalités à implémenter cumulant **un total d'au moins 30 points**.
Si vous décidez d'aller plus loin et d'avoir un total de points plus élevé, **l'excédent ne sera pas comptabilisé**.

- **10% (FA1)** Sécuriser et encrypter les communications au travers de certificats SSL.
- **5% (FA21)** Intégration du *Service Mesh Consul-Connect*
  - **5% (FA22)** Intégration de la fonctionnalité de *Service Discovery de Consul-Connect*
  - **5% (FA23)** Observabilité des services et de leurs états (*healthcheck*) au travers de l'interface de *Consul*
  - **10% (FA24)** Définition d'*Intentions* limitant la communication entre les services au strict minimum
  - **10% (FA25)** Configuration de *Canary Deployment* et/ou *Blue/green (A/B) Deployment* (Avec Consul)
- Observabilité et monitoring **(25%)**
  - **5% (FA31)** Intégration d'un outil de gestion de journaux (*Loki, Fluentd, Logstash*, ...)
  - **5% (FA32)** Intégration de monitoring des ressources physiques (CPU, Mémoire, ...) (*Prometheus*, ...)

  - **5% (FA33)** Intégration de Tracing des communications entre les microservices (*Jaeger*, ...)
  - **10% (FA34)** Visualisation (*Grafana*, *Kibana*, ...)
- **15% (FA41)** Intégration d'un système de *Continuous Delivery* (*ArgoCD*, ...)

Assurez-vous de remplir les documents `submission.md` ***et*** `submission.json` ***avec les codes (FA1, FA21...) associés aux fonctionnalités que vous avez sélectionnées.***

Bon courage et bonne fin de session!