École CentraleSupélec, campus de Metz

Rapport projet Tweetoscope

Equipe 9

03 Décembre 2021

Anas El Belbali, Maxence Duplech-Gauffre, Gaspard Vingtrinier

CentraleSupélec

Année Universitaire 2021-2022

Is the solution designed to scale ?

Chaque composant de la pipeline du Tweetoscope a été designé séparemment. Ainsi, il est très facile de multiplier les instances de chaque bloc. Par exemple, pour obtenir 3 "learners", il suffit de créeer trois containers du learner grâce au Dockerfile corespondant. Un exemple est présenté dans la vidéo sur Intercell. On peut également multiplier les partitions de chaque topic à l'avance en modifiant les fichiers docker-compose-middleware.yml et kafka-zookeeper.yml pour docker-compose et kubernetes respectivement.

To what extend is your solution fault-tolerant ?

Lors du deploiement sur kubernetes, que ce soit sur minikube ou intercell, lorsqu'un node est détruit, un nouveau vient immédiatemment prendre sa place. Aini, l'application peut continuer à fonctionner après une courte pause.
Pour tester ceci, nous avons essayé de détruire tous les blocs de la pipeline (du tweet generator au predictor) un à un, puis certain par paire. Dans tous les cas, on vérifiait après que Tweetoscope envoyait bien des messages d'alertes au topic corespondant.

Is your solution dockerized ? What is the size of your various images ? Are the images built manually or did you write Dockerfiles ?

Chaque partie de notre Tweetoscope a été dockerisée. Tous les Dockerfile se trouvent dans le dossier docker de notre dépôt.
La taille de l'image du tweet generator a été optimisée en partant d'une image alpine plutôt que Ubuntu.
Pour les autres, nous avons utilisé Ubuntu comme image de base. Cela vient du fait que nous avions des problèmes d'installation des modules Python nécessaires au bon fontionnement de notre application. Nous avons manqué de temps pour investiguer ce problème.
Toutes les images docker ont été postée sur DockerHub avec la convention de nom suivante : *gaspardv/tweetoscope_NomDuBloc* (par exemple gaspardv/tweetoscope_collector).

How do you deploy the various components of the application ? Do you use Docker Compose or Kubernetes ? Do you use Minikube or multiple nodes on Intercell ?

Toutes les approches de déploiement ont été utilisées. Il est par exemple possible d'utiliser docker-compose ou minikube pour deployer l'application en local. On peut également utiliser Kubernetes sur les clusters de l'école. Kubernetes a comme avantage d'être "fault-tolerant".

Continuous integration pipeline

Nous avons une pipeline cicd. Celle-ci va automatiquement mettre à jour les images dockers sur DockerHub à chaque fois qu'une modification a lieu sur le dépôt Git.
Nous avons essayé d'implémenter des tests unitaires automatique en Python avec pytest. Malheureusement, nous avons manqué de temps pour résoudre un problème de "pip : command not found". Ce bloc de la pipeline n'est donc toujours pas opérationnel.

Is the Git clean ?

Le Git dispose d'un .gitignore et d'un .dockerignore pour ne pas envoyer sur git et docker des fichiers indésirables (__pycache__, jupyter notebook etc...). De plus, nous avons également un fichier requirements.txt afin de faciliter l'intallation des dépendences python nécessaires. Bien sûr, cette intallation est optionnelle pour l'utilisateur si celui-ci utilise seulement des versions de notre application avec docker-compose ou kubernetes.
Un README est également présent. Il fourni les intructions d'installation ainsi que des explications sur les différents éléments de notre project.
Enfin, chaque branche est supprimée une fois qu'elle est fusionnée avec la dernière version du Tweetoscope. Cela permet de gagner en visibilité sur l'avancement du projet.

Conclusion

Tous les points clefs du projet ont été réalisé. Nous avons construit un Tweetoscope opérationnel qui peut être déployé sur les clusters de l'école.