Local Kube

Projet de java

Mohamed YAHYAOUI / Ruben SERO

I) Introduction

LocalKube est une application permettant de contrôler des applications Java à l'intérieur de conteneurs docker. Les applications seront contrôlé par SpringBoot à l'aide de requête JSON. Les logs seront stockés dans une base de donnée SqliteJDBC et accessible également à l'aide de requêtes JSON. Dans les parties suivantes, nous vous expliqueront les différentes fonctionnalités en illustrant nos exemples en utilisant Postman comme le client effectuant les requêtes REST.

L'application peut être lancé à l'aide de la commande :

java -jar –enable-preview local-kube.jar

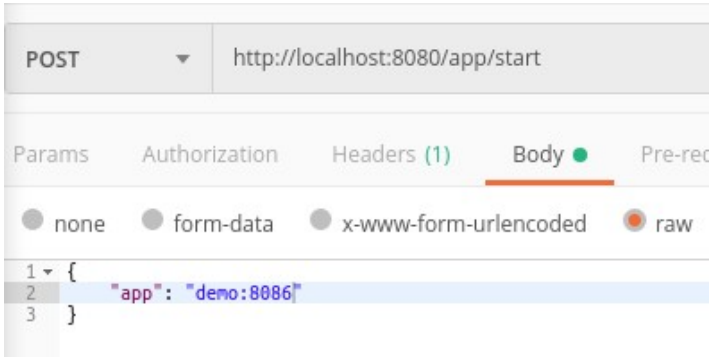
Note : la partie auto-scale n'est pas présente sur cette application.

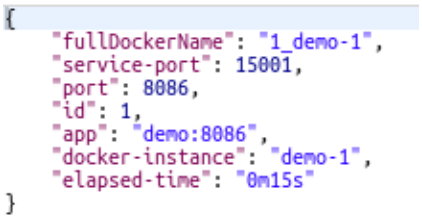
II) Application : Démarrer, stopper et lister

Note: Toutes les requêtes JSON pour une action sur l'application commence par app/

A) Démarrer une application (app/start) :

Après avoir lancé le programme, il est possible de lancer une ou plusieurs application. Pour cela, il faut lancer une requête JSON de type POST car LocalKube doit connaître le nom et le port de l'application à démarrer.

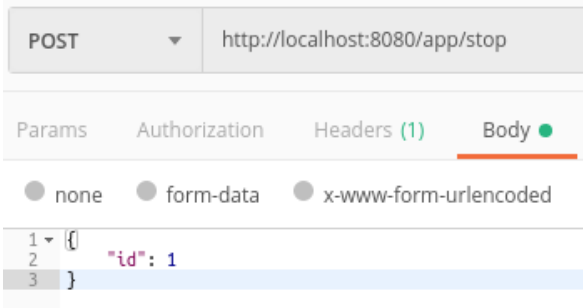


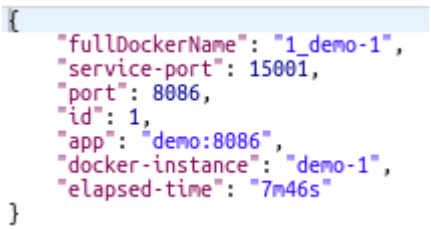


Exemple de requête et de reponse pour le lancement de l'application demo sur le port 8086.

B) Stopper une application (app/stop) :

Pour stopper une application, on appelle la requête POST JSON app/stop. Il faudra préciser l'id de l'application à fermer.

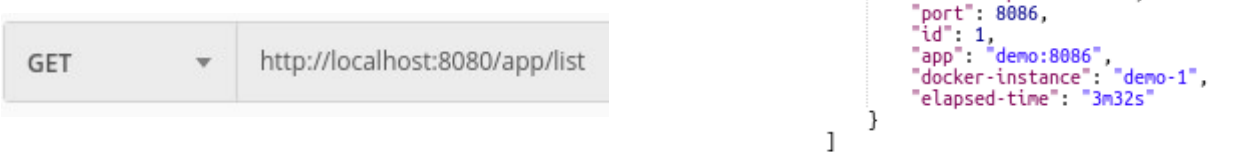




Exemple de requête stoppant l'application ayant pour id 1.

C) Lister les applications lancées (app/list) :

Pour lister toutes les applications qui tournent, on appelle la requête GET JSON app/list. Le JSON renvoyé comportera une liste des informations de chaque applications qui tournent.




```
[
    {
        "fullDockerName": "1_demo-1",
        "service-port": 15001,
        "port": 8086,
        "id": 1,
        "app": "demo:8086",
        "docker-instance": "demo-1",
        "elapsed-time": "3m32s"
    }
]
```


Exemple de requête list .

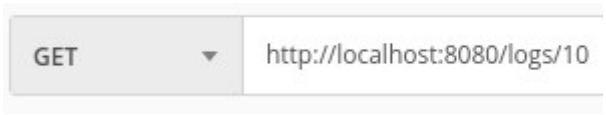
III) Log : Filtrer suivant des conditions

Note: Toutes les requêtes JSON pour une action sur l'application commence par app/
Note 2: Tout les résultats JSON seront sous la même forme que l'exemple A.

Tous les logs renvoyés par chaque application sont enregistrés dans des fichiers de logs. Pour accéder à ces logs, il est possible d'utiliser différentes requêtes pour avoir différent filtres.

A) Filtrer les logs suivant le temps (logs/ :time) :

Tout d'abord, si l'on souhaite filtrer les logs des n dernières minutes, on peut le faire en utilisant la requête GET /logs/n. (n étant un entier sinon une erreur sera renvoyée)

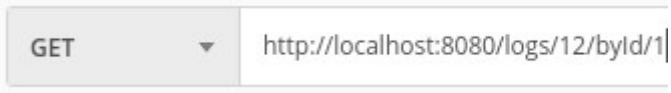



```
[
    {
        "port": 8086,
        "service-port": 15001,
        "docker-instance": "demo-1",
        "message": "",
        "timestamp": "2020-12-30 12:11:04.542",
        "id": 1,
        "app": "demo:8086"
    },
    {
        "port": 8086,
        "service-port": 15001,
        "docker-instance": "demo-1",
        "message": "   .   ___                          _            __ _ _",
        "timestamp": "2020-12-30 12:11:04.544",
        "id": 1,
        "app": "demo:8086"
    }
]
```


Exemple de requête renvoyant les logs des dix dernières minutes et son résultat .

B) Filtrer les logs suivant le temps et l'ID (logs/ :time/byId/:id) :

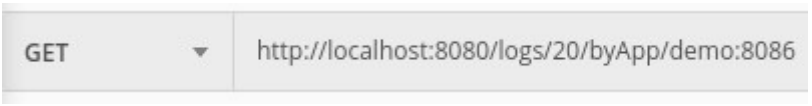
Pour filtrer les logs plus précisément, on peut préciser l'ID. Dans ce cas, on lance la requête GET /logs/n/byId/id tel que n est correspond au n dernières minutes et id correspond à l'id que doivent comporter les applications retournées.



Exemple de requête renvoyant les logs des douze dernières minutes des applications ayant l'id 1.

C) Filtrer les logs suivant le temps et le nom de l'application (logs/ :time/byApp/ :app) :

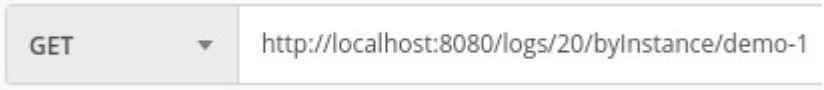
Pour filtrer filtrer suivant le temps et le nom de l'application, on lance la requête GET /logs/n/byApp/app tel que n correspond au n dernières minutes et app correspond au nom de l'application que doivent comporter les applications retournées.



Exemple de requête renvoyant les logs des vingts dernières minutes des applications ayant pour nom demo:8086.

D) Filtrer les logs suivant le temps et l'instance docker (logs/ :time/byInstance/;instance) :

Pour filtrer filtrer suivant le temps et l'instance docker, on lance la requête GET /logs/n/byInstance/instance tel que n correspond au n dernières minutes et instance correspond l'instance docker que doivent comporter les applications retournées.



Exemple de requête renvoyant les logs des vingts dernières minutes des applications ayant pour instance docker demo-1.

E) Filtrer les logs suivant le temps et un des trois critères précédent (logs/ :time) :

Pour filtrer les logs suivant un des trois critères précédent sans le préciser, on lance la requête GET /logs/n/value tel que n correspond au n dernières minutes et value correspond soit à l'id, soit au nom de l'application, soit à l'instance docker que doivent comporter les applications retournées.