STACK LABS

Pascal Gillet <pascal.gillet@stack-labs.com>

[hippi-spark-k8s] Feedback - Suivi de mission

Pascal Gillet <pascal.gillet@stack-labs.com> 16 novembre 2020 à 15:19
À : Florence BRARD <FBRARD@sii.fr>, David Desviel <david.desviel@stack-labs.com>, Xavier Villalba <xavier.villalba@stack-labs.com>, Pascal ROUANET <PROUANET@sii.fr>, "QUINCHARD, Eric" <eric.quinchard@airbus.com>, "DELBENDE, Marc" <marc.delbende@airbus.com>, "ROUGET, Matthieu" <matthieu.rouget@airbus.com>

Bonjour à Tous,

Voici les travaux réalisés la semaine dernière:

État d'avancement: **Nominal**

- **Submission d'un job Spark en Python avec le Spark Operator:**
Le client Python est fondé sur les ressources standard de Kubernetes (`Pod`, `Service`, `ConfigMap`, etc.), et offre des méthodes pour créer, lire, mettre à jour et supprimer de telles ressources.
On peut par exemple créer de façon purement programmative un `Pod`, un `Service`, etc. avec des classes d'objets dédiées.
Le Spark Operator, quant à lui, gère la ressource *custom* `SparkApplication`. Ce type (`kind`) de ressource n'est donc pas supporté directement dans le client Python. Il offre néanmoins des méthodes d'utilité pour gérer des ressources custom.

Exemple pour créer un job Spark avec le Spark Operator:

```
api = client.CustomObjectsApi()
# create the resource
api.create_namespaced_custom_object(...)
```

Voir l'exemple complet ici:
https://gitlab.com/stack-labs/client/airbus-defence-and-space/hippi-spark-k8s/hippi-spark-k8s/-/blob/master/k8s_python_client_examples/spark_operator_job_create.py

La submission d'un job est donc différente selon qu'on utilise le Spark Operator ou le "spark-submit" en mode client, et il faudra donc prévoir 2 chemins de code.

`SparkApplication` est une ressource "de façade": une fois lancée, elle crée *in fine* des `Pods` driver et executors.
Donc les autres opérations nécessaires à la plate-forme HiPPI (obtenir le statut, lecture des logs, kill) sont communes quelle que soit la méthode de submission choisie.
Concernant le kill, il faudra juste s'assurer de supprimer la `SparkApplication` en plus.

- **Obtenir le statut du pod driver:**
Voir un exemple ici:
https://gitlab.com/stack-labs/client/airbus-defence-and-space/hippi-spark-k8s/hippi-spark-k8s/-/blob/master/k8s_python_client_examples/pod_namespace_status.py

Petite subtilité ici: l'opération ne peut pas être "watched", c'est-à-dire répétée à intervalles réguliers pendant une période de temps donnée, avec le mécanisme de `Watch` offert par la librairie Python. Mais on peut tout à fait boucler nous-mêmes sur cet appel de fonction dans un thread séparé.

- **Obtenir les logs du pod driver**
Voir un exemple ici:
https://gitlab.com/stack-labs/client/airbus-defence-and-space/hippi-spark-k8s/hippi-spark-k8s/-/blob/master/k8s_python_client_examples/pod_namespace_log.py

Ici, l'opération peut être "watched". On peut récupérer les logs en cours d'exécution en streaming jusqu'à la complétion du job, ou *a posteriori* (tant que le driver n'est pas "garbage collecté").
Cette opération a été testée dans un thread séparé, pour ne pas retarder le fil d'exécution principal (ex: exécution de plusieurs jobs en parallèle).

Cette semaine marque un jalon important du projet: tous les voyants sont au **vert** pour commencer à intégrer la librairie Python pour K8s dans le spark-client.

Je vais donc m'atteler le temps restant à refactorer/configurer le spark-client pour le support de k8s.

TODO pour Éric:
Je fais suite au point sur le script `main` en Python pour Mesos.
Pour les applications Python, on passe simplement un fichier `.py` en 1er argument du `spark-submit` (au lieu d'un JAR pour les applications Java) et on ignore l'option `--class`. On peut aussi ajouter des fichiers Python .zip, .egg ou .py au chemin de recherche avec --py-files.
À la lecture du code, je ne vois rien relatif à la contrainte que tu m'as évoquée ce matin. Est-ce que tu peux itérer avec Marc pour me préciser le problème?

Merci,
--

STACK LABS
Pascal GILLET | Big Data & Cloud Architect
stack-labs.com
21 Boulevard de la Marquette, 31000 Toulouse
France · Canada