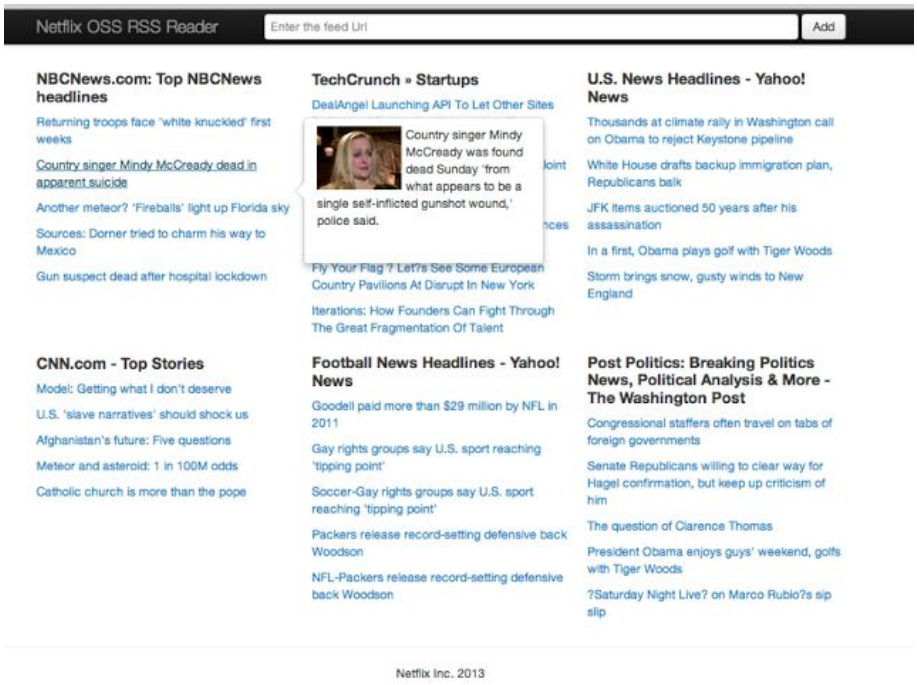
TP ESIR 2 : module Architecture logicielle

Objectif : Réaliser le déploiement de l'application RSS Reader en utilisant les briques architecturales fourni par Netflix OSS.

Introduction

L'application RssReader permet de créer ce que l'on appelle un aggrégateur de news. Un aggrégateur de news est une application cliente qui se connecte à différents fournisseurs de "news" pour les présenter en un seul et même lieu.

Ci-dessous est présenté un snapshot de l'application.



L'application que doit être déployée possède l'architecture décrite dans la figure ci-dessous. A droite se situe l'ensemble des clients qui se connectent à l'application.

L'application est ensuite composée d'un ensemble de 3 micro-services:

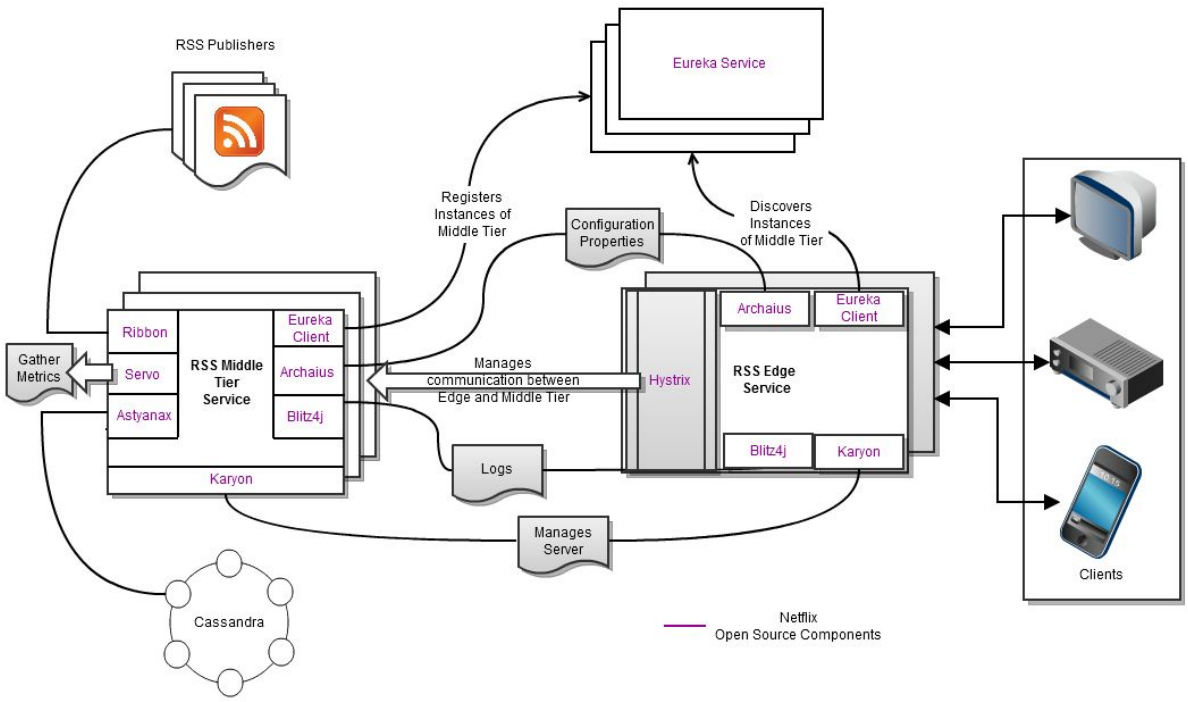
- le micro-service "RSS Edge Service". Ce micro-service a pour but la mise en forme de l'application et de faire l'interface avec les différents clients qui se connectent à l'application.
- le micro-service "Rss Middle Tier Service". Ce micro-service est celui qui agrège les flux de "news" provenant des différentes sources.

- Le micro-service “Eureka Service”. Ce micro-service est un composant open source développé par la société Netflix qui permet de faire de la découverte de service. C’est donc ce micro-service qui permettra au “Rss Edge Service” de trouver l’emplacement du micro-service “Rss Middle Tier Service”

Il est également possible de connecter très facilement ce système à une base de donnée Cassandra pour le stockage des données. La configuration par défaut de l’application utilise un stockage en mémoire sur le “Rss Middle Tier Service”. Dans un premier temps, il est conseillé pour des raison de simplicité de mise en place de conserver cette configuration par défaut. Par la suite, vous pourrez si vous le souhaitez remplacer le stockage en mémoire par le stockage sur Cassandra. Tous les composants des différents micro-services décrits en violet sur l’architecture, sont des composants open source développé par la société Netflix.

La liste des composants utilisés dans cette application est la suivante :

- Archaius: Un client pour la configuration dynamique de propriété.
- Astyanax: Un client pour la connexion à Cassandra.
- Blitz4j: Utilistaire de logging non-bloquant.
- Eureka: Service de découverte et d’enregistrement des services.
- Governator: Injection de dépendances.
- Hystrix: Gestion de la tolérance des fautes des dépendances.
- Karyon: La base du serveur pour gérer les requêtes entrantes et le cycle de vie du microserver ainsi que le démarage des autres composants qui sont contenus dans le micro-service.
- Ribbon: Un client REST pour les requêtes sortante.
- Servo: Permet de récupérer des métriques sur l’exécution.



Le poste de blog concernant cette application se trouve ici (http://techblog.netflix.com/2013/03/introducing-first-netflixoss-recipe-rss.html).

Travail demandé

L'objectif globale de ce mini-projet est de déployer cette application afin de comprendre son fonctionnement. Par la suite, il est demandé de réaliser certaines améliorations de l'architecture afin de permettre à l'application de passer à l'échelle d'un grand nombre de clients et d'être tolérante aux fautes (la panne de certains serveurs contenant le micro-service "Rss Middle Tier Service" principalement). Tout peut être installé sur une seule machine ou plusieurs machine différentes.

Voici les 5 grandes étapes du projet :

1. **Installer l'application** - vérifier son fonctionnement et tester manuellement à l'aide d'un navigateur que l'application est accessible et que l'ajout et la création de votre propre aggrégateur de news est possible.
    a. Vous trouverez le code de l'application ici. (https://github.com/Netflix/recipes-rss)
    b. Vous trouverez les instructions sur la façon de la construire, puis de la démarrer ici. (https://github.com/Netflix/recipes-rss/wiki)

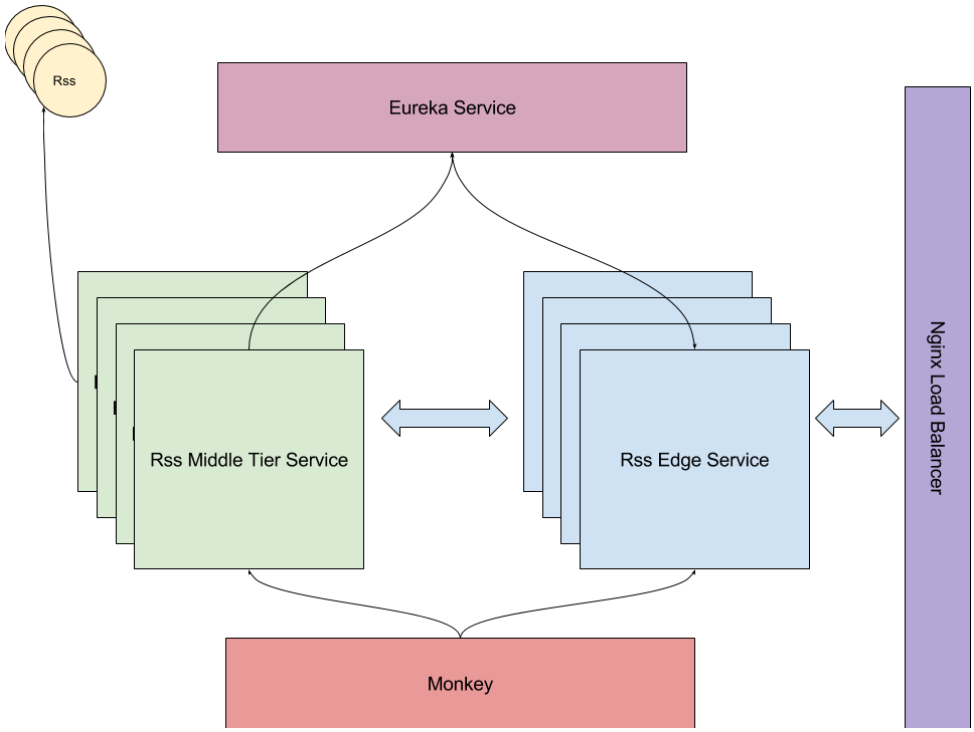
2. Mettre dans des conteneurs docker les 3 micro services et relancer l'application ainsi réalisée. Vous testerez de nouveau votre application.

3. Lancer plusieurs instances (au moins 2 de chaque) des 2 micro-services applicatifs, afin de permettre une redondance pour la tolérance aux fautes ainsi que pour le passage à l'échelle. Vous ferez en sorte que cela fonctionne.
   La suite est considéré comme du bonus pour votre projet.

4. Ajouter devant vos micro-services "RSS Edge Service" un load balancer Nginx afin de répartir automatiquement la charge (et éventuellement de gérer la disponibilité ou non des micro-service "RSS Edge Service".

5. Coder et ajouter un monkey qui va venir aléatoirement éteindre certains docker (principalement les docker contenant les micro-services "Rss Middle Tier Service", mais il peut également s'occuper des micro-service "RSS Edge Service"). Ce monkey peut également remettre en marche des micro-services.

L'application finale à l'étape 5 doit avoir une architecture proche de celle présenté dans la figure suivante :



Quelques commentaires sur l'installation de l'application

Pour la construction de l'application, vous devrez utiliser git (pour cloner le repository) et gradle (un gestionnaire de build, dans la même ligné que maven). Ces outils doivent être installés préalablement. Il faudra également que vous installiez une version de tomcat pour faire marcher le service Eureka. La procédure de build vous générera un fichier jar pour les micro-services "RSS Edge Service" et "Rss Middle Tier Service". Vous aurez également le jar de tomcat à lancer en copiant le war de Eureka dans le répertoire webapp de tomcat (tout cela vous est expliqué en détail sur le wiki dont le lien vous est donné au dessus).

Afin de vous simplifier la tache (la compilation ne marche pas forcément du premier coup):

- Nous vous conseillons de commenter dans l'application la partie du build qui génère la javadoc ainsi que la partie utilisant findbugs. Vous trouverez cela dans les fichiers "gradle/convention.gradle" et "gradle/check.gradle".

Rendu

Il vous sera demandé de faire une démonstration de votre application finale. Cette démonstration doit être scénarisée afin de mettre en valeur les choix que vous avez fait et les qualités de votre architecture. Vous devrez également rendre un document de 5 pages maximum expliquant l'architecture de votre projet (des schémas sont les bienvenus) et fournissant une analyse des forces et des faiblesses de votre architecture.