
INF8480

École Polytechnique de Montréal

# Travail pratique #3 : Initiation aux services de l'infonuagique

Par

François-Xavier Dueymes (1584742)

Vincent Leduc (1739632)

Remis à

Adel Belkhiri

4 décembre 2018

# Tests de performance

Le script *loop_request.sh* est utilisé avec time pour envoyer simultanément 30 requêtes GET au service web. Pour le service sans répartiteur de charge, l'adresse IP à utiliser est une adresse IP flottante associé manuellement au serveur comme expliqué dans l'annexe de l'énoncé. Pour la version avec le répartisseur de charge, nous avons utilisé la ressource *Neutron::FloatingIP* pour allouer et associer automatiquement une adresse IP flottante au répartiteur de charge. L'adresse allouée est affichée dans la sortie du stack. Il est nécessaire d'utiliser cette adresse IP dans le script de test.

## Service web sans répartiteur de charge

Voici un exemple de sortie :

```
[frdue@l4712-01 tp3 (master)] $ time sh request_loop.sh
Salut Yvon. Je suis tp3-1-server-ailbedk6jhyk.request 1 done
Salut Yvon. Je suis tp3-1-server-ailbedk6jhyk.request 4 done
Salut Yvon. Je suis tp3-1-server-ailbedk6jhyk.request 3 done
Salut Yvon. Je suis tp3-1-server-ailbedk6jhyk.request 5 done
Salut Yvon. Je suis tp3-1-server-ailbedk6jhyk.request 6 done
Salut Yvon. Je suis tp3-1-server-ailbedk6jhyk.request 7 done
Salut Yvon. Je suis tp3-1-server-ailbedk6jhyk.request 25 done
Salut Yvon. Je suis tp3-1-server-ailbedk6jhyk.request 12 done
Salut Yvon. Je suis tp3-1-server-ailbedk6jhyk.request 23 done
Salut Yvon. Je suis tp3-1-server-ailbedk6jhyk.request 2 done
Salut Yvon. Je suis tp3-1-server-ailbedk6jhyk.request 21 done
Salut Yvon. Je suis tp3-1-server-ailbedk6jhyk.request 16 done
Salut Yvon. Je suis tp3-1-server-ailbedk6jhyk.request 17 done
Salut Yvon. Je suis tp3-1-server-ailbedk6jhyk.request 18 done
Salut Yvon. Je suis tp3-1-server-ailbedk6jhyk.request 22 done
Salut Yvon. Je suis tp3-1-server-ailbedk6jhyk.request 20 done
Salut Yvon. Je suis tp3-1-server-ailbedk6jhyk.request 15 done
Salut Yvon. Je suis tp3-1-server-ailbedk6jhyk.request 28 done
Salut Yvon. Je suis tp3-1-server-ailbedk6jhyk.request 19 done
Salut Yvon. Je suis tp3-1-server-ailbedk6jhyk.request 24 done
Salut Yvon. Je suis tp3-1-server-ailbedk6jhyk.request 26 done
Salut Yvon. Je suis tp3-1-server-ailbedk6jhyk.request 14 done
Salut Yvon. Je suis tp3-1-server-ailbedk6jhyk.request 10 done
Salut Yvon. Je suis tp3-1-server-ailbedk6jhyk.request 11 done

real    0m15,528s
user    0m0,160s
sys     0m0,141s
```

Résultats pour 5 tests sur le service web simple :

- 17,571s
- 16,348s
- 15,125s
- 18,046s
- 15,528s

La seule machine virtuelle traite les requêtes qu'elle reçoit l'une après l'autre.

## Service web avec répartiteur de charge

Voici un exemple de sortie

```
[frdue@l4712-15 tp3 (master)] $ time sh request_loop.sh
Salut Yvonne. Je suis my-server-1.Salut Yvonne. Je suis my-server-0.request 2 done
request 5 done
Salut Yvonne. Je suis my-server-1.Salut Yvonne. Je suis my-server-0.request 3 done
request 1 done
Salut Yvonne. Je suis my-server-1.Salut Yvonne. Je suis my-server-0.request 6 done
request 7 done
Salut Yvonne. Je suis my-server-1.Salut Yvonne. Je suis my-server-0.request 13 done
request 4 done
Salut Yvonne. Je suis my-server-1.request 29 done
Salut Yvonne. Je suis my-server-0.request 27 done
Salut Yvonne. Je suis my-server-1.request 18 done
Salut Yvonne. Je suis my-server-0.request 19 done
Salut Yvonne. Je suis my-server-1.request 22 done
Salut Yvonne. Je suis my-server-0.request 25 done
Salut Yvonne. Je suis my-server-1.request 17 done
Salut Yvonne. Je suis my-server-0.request 11 done
Salut Yvonne. Je suis my-server-1.request 20 done
Salut Yvonne. Je suis my-server-0.request 8 done
Salut Yvonne. Je suis my-server-1.request 26 done
Salut Yvonne. Je suis my-server-1.request 24 done
Salut Yvonne. Je suis my-server-0.request 10 done
Salut Yvonne. Je suis my-server-1.request 28 done
Salut Yvonne. Je suis my-server-0.request 14 done
Salut Yvonne. Je suis my-server-1.request 21 done
Salut Yvonne. Je suis my-server-0.request 30 done
Salut Yvonne. Je suis my-server-1.request 15 done
Salut Yvonne. Je suis my-server-0.request 9 done
Salut Yvonne. Je suis my-server-1.request 12 done
Salut Yvonne. Je suis my-server-0.request 23 done
Salut Yvonne. Je suis my-server-1.request 16 done

real    0m8,037s
user    0m0,155s
sys     0m0,109s
```

Résultats pour 5 tests sur le service web avec répartiteur de charge :

- 7,539s
- 7,553s
- 7,545s
- 7,542s
- 8,037s

Nous avons observé que pour certaines requêtes, les deux serveurs répondaient en même temps.

Le service web met deux fois moins de temps à répondre aux 30 requêtes que le service déployé sur une seule machine virtuelle. Le répartiteur de charge reçoit les requêtes et les distribue entre les deux serveurs selon la méthode *Roune Robin*. Les résultats deux fois plus petits que ceux du service sans répartiteur de charge sont cohérents avec la répartition des requêtes sur deux serveurs.

# Question 1

**Nova**

C'est le composant de calcul d'OpenStack, considéré aussi comme le plus complexe. Il permet de créer de multiples instances de machines virtuelles sur de larges réseaux d'hôtes exécutant Nova et de les gérer. Nova se veut d'être utilisable indépendamment du matériel de l'hôte et du type d'hyperviseur choisi. C'est le composant central d'OpenStack et qui a besoin d'intéragir en permanence avec d'autres composants, comme Keystone pour l'authentification et Glance pour la gestion des images.

**Neutron**

Ce composant prend en charge les réseaux et l'adressage IP dans OpenStack. C'est Neutron qui gère les adresses IP flottantes qu'il est possible d'assigner aux différentes instances crées afin de les relier à Internet. Neutron permet aussi aux utilisateurs de créer des réseaux et de leur définir des groupes de sécurité, et de déployer des réseaux selon une l'infrastructure définie.

**Heat**

C'est le composant d'OpenStack qui sert à l'orchestration. Dans ce TP, c'est avec Heat que nous avons construit des gabarits pour le déploiement de nos machines virtuelles. Ces gabarits permettent d'automatiser certains processus de déploiement. Par exemple, dans ce TP, cela nous a permis d'automatiser le téléchargement et le lancement du serveur.

**Keystone**

C'est le composant d'authentification d'Openstack. Il permet aux utilisateurs de s'authentifier sur la plate-forme et de se connecter aux autres utilisateurs. Par exemple, on peut utiliser Keystone pour contrôler l'accès à certaines ressources (volumes, machines virtuelles, IP flottantes) dans une organisation en possédant plusieurs (un directeur n'aura probablement pas les mêmes accès qu'un stagiaire). Aussi, Keystone serait utile dans le cas ou une compagnie décide d'installer un nouveau nuage et désirerait en permettre l'accès avec le nom d'utilisateur et le mot de passe déjà assignés à ses employés.

**Glance**

C'est le service d'image d'OpenStack. Il gère l'ensemble des fonctionnalités liées aux images distribuées aux instances, ainsi que les métadonnées de celles-ci (ex : format, conteneur). Il permet ainsi de contrôler l'accès aux propriétés des images. Il peut aussi être utilisé pour déterminer le type d'hyperviseur sur lequel une image spécifique sera instanciée.

## Question 2

**OS::Heat::ResourceGroup** : Elle permet de générer le nombre spécifié de machines virtuelles ayant des caractéristiques identiques (ici 2 machines virtuelles).

**OS::Neutron::HealthMonitor** : Elle permet de vérifier si le *pool* est utilisable, c'est-à-dire si elle peut traiter les requêtes qui lui sont envoyées.

**OS::Neutron::Pool** : Elle regroupe les machines virtuelles et divise les requêtes selon la méthode de répartition de charge choisie (Round Robin dans notre cas).

**OS::Neutron::LoadBalancer :** Elle permet de lier le *pool* de ressources à un port particulier (8080 dans notre cas).

**OS::Nova::Server** : Elle permet de définir le réseau, l'image, la saveur ainsi que le script à exécuter sur chacune des machines virtuelles.

## Question 3

1) OS::Heat::AutoScalingGroup

2) OS::Ceilometer::Alarm : Permet de définir des alarmes. Il faut lui donner la métrique à utiliser ainsi que la limite à tester. On peut également lui spécifier l'opérateur de comparaison. La métrique utilisé dans ce cas-ci serait *cpu.*

OS::Heat::ScalingPolicy : Permet de modifier le nombre d'instance avec la propriété *scaling_adjustment* associée à un nombre. Si on veut ajouter une instance, ce nombre sera 1 et, à l'inverse, si on souhaite en retirer une, ce nombre sera –1. Il faut toutefois liée la politique d'ajustement dans la propriété *alarm_actions* de *Alarm* pour que cette action soit prise.