

INF8480 Systèmes répartis et infonuagique

TP3 : Initiation aux services de l’infonuagique

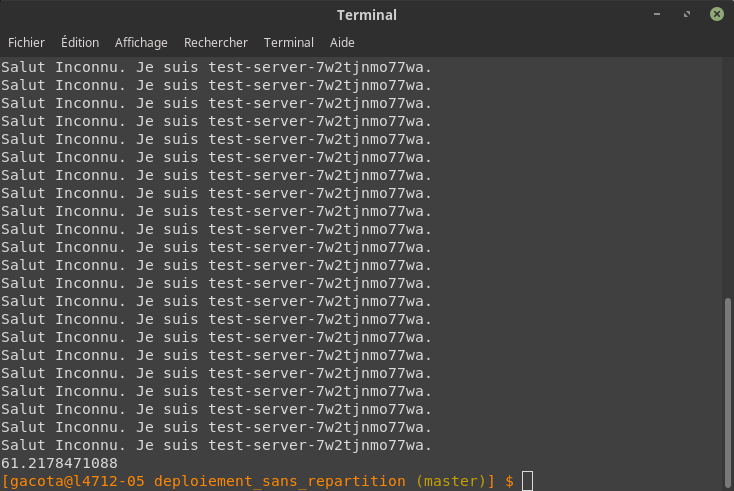
**Gabriel Côté-Jones : 1771119**

**Aladin Riabi 1717705**

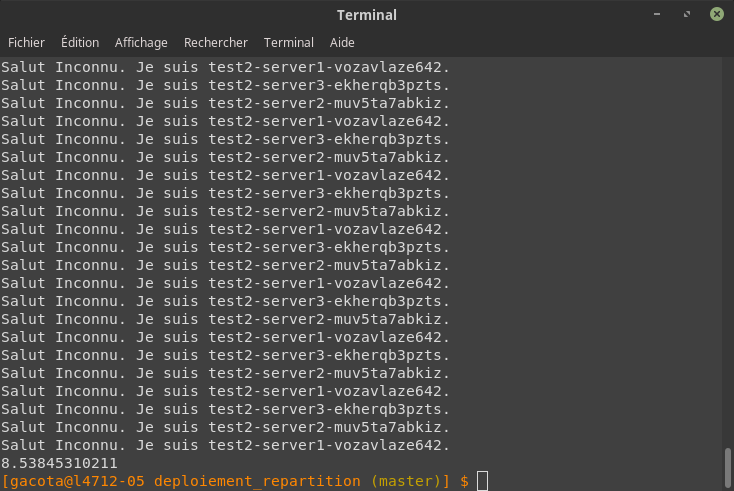
Polytechnique Montréal

15 Avril 2019

**Tests de performance**



*Fig. 1 : Envoi de 50 requêtes http sans répartition.*



*Fig. 2 : Envoi de 50 requêtes http avec répartition*

Les figures 1 et 2 représente l’exécution d’un script Python qui envoi 50 requête http à l’adresse IP flottante d’un service sans et avec répartition de charge. Le tableau suivant montre le temps d’exécution en secondes pour chaque cas.

|  |  |  |
| --- | --- | --- |
|  | **Sans répartition** | **Avec répartition** |
| **Temps de traitement de 50 requêtes http simultanées en secondes** | 61.2 | 8.5 |
| **Temps de traitement de 50 requêtes http séquentielles en secondes** | 25 | 25 |

Bien qu’il y ait trois fois plus de serveurs pour traiter les requêtes dans le cas de répartition de charge, on voit que le temps de traitement est moins que le tier du temps de traitement pour le système sans répartition. Ces résultats peuvent être expliquer par le fait que le serveur sans répartition se retrouve dans une situation de surcharge.

Lors de l’envoi des 50 requêtes simultanés au service sans répartition, on voit clairement qu’il y a une surcharge du serveur. Ce dernier cesse de périodiquement de traiter les requêtes pendant quelques secondes. Lors de certaines exécutions du script, il cesse complètement de traiter toutes requêtes et envoie un message d’erreur 503 pour les requêtes restantes après 109 secondes.

On peut également voir qu’il y a une surcharge du serveur sans répartition lors de l’envoi de 50 requêtes simultanés en envoyant les requêtes séquentiellement. Cette dernière méthode nous donne un temps total de traitement de 25 secondes pour 50 requêtes. Moins de la moitié du temps de traitement des 50 requêtes simultanés.

**Questions**

Question 1 :

* **Nova :** Nova permet l’approvisionnement de serveurs virtuels. Les composants Nova sont faits pour s’exécuter sur des systèmes Linux. Il roule sur ces systèmes en Daemon. C’est-à-dire, ils s’exécutent en arrière-plan et servent à répondre aux requêtes du réseau.
* **Heat :** Heat est le service d’orchestration d’OpenStack. Ce service permet d’exécuter plusieurs composants/applications sur le cloud à l’aide d’un gabarit. Les gabarits Heat décrivent l’infrastructure du système et peuvent être rédiger à la main dansun fichier texte. De plus, Heat est compatible avec plusieurs formats de gabarit.
* **Neutron :** Neutron est la composantes OpenStack qui offre des services de réseautage. Cette composante donne aux utilisateurs d’OpenStack un API leur permettant de créer des topologies de réseautage dans le cloud ainsi que d’y appliquer les politiques de leurs choix.
* **Swift :** Cette composante offre une API pour la gestion de stockage de données sur le cloud. Swift est fait pour stocker des données lorsqu’on est incertains des requis d’espace ou de plan future. Cette composante offre aux utilisateurs d’OpenStack un service de stockage bâtit pour l’optimisation et la mise à l’échelle.
* **Keystone :** Cette composante offre une API pour l’authentification et la gestion d’utilisateurs.
* **Horizon :** Horizon offre aux utilisateurs un interface graphique pour les autres services d’OpenStack.

Question 2 :

* **OS::Heat::ResourceGroup :** Permet l’instantiation/exécution d’un certain nombre de serveur web avec possibilité de mise à l’échelle.
* **OS::Neutron::LBaaS::HealthMonitor :** Permet la surveillance de serveurs web et l’avertissement des serveurs non disponibles au conteneur de serveur.
* **OS::Neutron::LBaaS::Pool :** Le conteneur qui contient les serveurs web.
* **OS::Neutron::LBaaS::LoadBalancer :** Distribut les requêtes entre les different serveurs du conteneur (pool).
* **OS::Nova::Server :** Un serveur virtuel qui s’exécute comme Daemon sur un système Linux.

Question 3 :

1. **OS::Heat::AutoScalingGroup**
2. **OS::Ceilometer::Alarm et OS::Aodh::Alarm**: *alarm\_actions* définit l’action à poser, *threshold* définit le pourcentage d’utilisation du CPU à utiliser comme condition, *comparison\_operator* définit l’opérateur de comparaison avec la valeur *threshold, period* définit le temps d’évaluation.

**Références**

1. Documentation OpenStack tirée de : <https://docs.OpenStack.org>