INF8480 – Systèmes répartis et infonuagique
TP1 – Appels de méthodes à distance

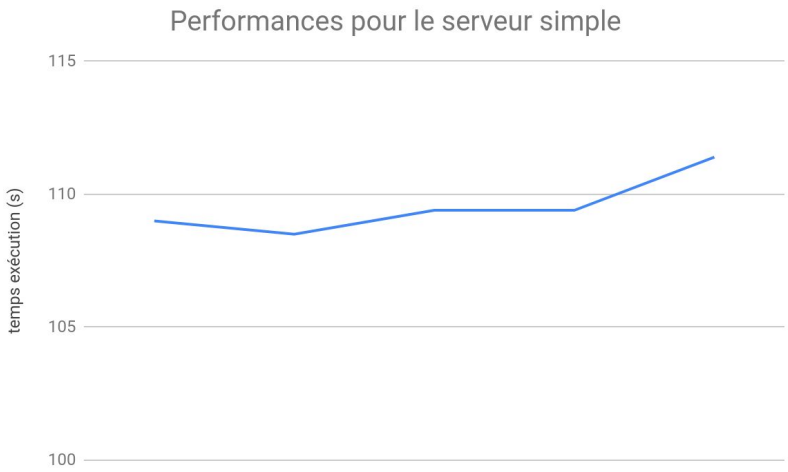
Jérémie LACHKAR - 1918934

Yujia DING - 1801923

Résultats

Sans répartiteur de charge: On écrit un script bash qui lance 50 appels en parallèle au serveur sans répartiteur de charge.

On lance ce script cinq fois pour avoir plusieurs valeurs et on trace le nuage de points associé.



Sans répartiteur de charge, le temps d'exécution des 50 requêtes prend environ 110s.

Avec répartiteur de charge: On écrit un script bash qui lance 50 appels en parallèle au serveur avec répartiteur de charge.

Remarque: il y a un problème avec le LoadBalancer qui n'attribue pas les requêtes. Par manque de temps on ne peut pas le résoudre.

Questions

Question 1:

- **HEAT -** Heat est un service pour orchestrer plusieurs composantes sur le nuage à l'aide de templates.
- **NOVA -** Nova est un service qui permet de déployer une structure de contrôle infonuagique supportant une large variété de technologies comme Hyper-V, VMware, XenServer etc.
- **NEUTRON -** Neutron est un projet OpenStack qui permet de gérer la connectivité réseau entre les interfaces des différents éléments gérés par les autres services OpenStack.
- **CINDER -** Cinder est un service de stockage en bloc pour OpenStack. Il est conçu pour présenter aux utilisateurs des ressources de stockage pouvant être utilisées par OpenStack Compute Project (Nova). Cinder virtualise la gestion des périphériques de stockage en mode bloc et fournit aux utilisateurs une API en libre service leur permettant de demander et de consommer ces ressources sans que l'on sache où leur stockage est réellement déployé ou sur quel type de périphérique.
- **MISTRAL -** Mistral est un service de workflow pour OpenStack. Ce projet a pour but de fournir un mécanisme de définition de tâches et de workflow sans écrire une ligne de code ainsi que la possibilité de les gérer et les exécuter dans sur un environnement cloud.
- **SAHARA -** Le projet Sahara a pour but de mettre à disposition un moyen simple de fournir des frameworks de traitement de données comme Apache Hadoop, Apache Spark ou encore Apache Storm. Ceci est fait en spécifiant divers paramètres de configuration tels que la version du framework, la topologie du cluster, les détails des noeuds du cluster etc.

Question 2:

- **OS::Heat::ResourceGroup -** Cette ressource permet de créer un groupe de ressource similaire, comme on a pu le faire pour créer un groupe de serveurs identiques pour la répartition de charge.
- **OS::Neutron::LBaaS::HealthMonitor -** Cette ressource permet de gérer les essais de connexions à une autre ressource. Si une ressource n'est plus accessible, cela évite d'essayer de la contacter indéfiniment puisque l'on peut limiter le délai d'attente ainsi que le nombre d'essai.
- **OS::Neutron::LBaaS::Pool -** Cette ressource permet de définir le sous-réseau dans lequel se trouve un groupe de noeuds.
- **OS::Neutron::LBaaS::LoadBalancer -** Cette ressource est la ressource qui permet de diriger le trafic vers les différents noeuds du groupe.
- **OS::Nova::Server -** Cette ressource permet de gérer des instances de serveurs Nova qui gère l'exécution de machines virtuelles.

Question 3:

1) La ressource pour remplacer OS::Heat::ResourceGroup et qui permet de modifier dynamiquement le nombre d'instances du serveur est OS::Heat::AutoScalingGroup. Les propriétés obligatoires pour cette ressource sont:
    a) Le nombre d'instance minimal
    b) Le nombre d'instance maximal
    c) La définition de la ressource qui doit être instanciée

2)
    a) La ressource qui permet de lancer une alerte lorsque le taux d'utilisation du CPU des machines atteint un seuil prédéfini est OS::Aodh::GnocchiAggregationByResourcesAlarm. Les propriétés qu'il faut spécifier pour cette ressource sont:
        i) Metric - propriété qui détermine la métrique utilisée par l'alarme. Pour le cpu, on spécifie *cpu_util.*
        ii) Query - la requête pour filtrer les métriques

iii) Resource_type - le type de ressource à modifier. Il s'agit ici de *instance*.
iv) Threshold - définit le seuil critique de la métrique
v) Comparison_operator - spécifie comment la ressource doit comparer la métrique au threshold. Cette propriété peut valoir *lt* ou *gt*.

b) La ressource qui permet d'ajuster automatiquement le nombre de machines virtuelles en fonction de ces alertes est OS::Heat::ScalingPolicy.
   i) Adjustment_type - le type d'ajustement à effectuer qui peut être *change_in_capacity*, *exact_capacity*, *percent_change_in_capacity*.
   ii) Auto_scaling_group_id - l'identifiant du groupe AutoScaling auquel il faut appliquer la politique de scaling.
   iii) Scaling_adjustment - taille de l'ajustement à effectuer. Cette valeur doit être un nombre. Par exemple, spécifier une valeur de 1 doit augmenter de 1 le nombre d'instance dans le groupe.