École Polytechnique de Montréal

Département de Génie Informatique et Génie Logiciel

INF8480 : Systèmes répartis et infonuagique

TP #3 – Initiation aux services de l'infonuagique

Réalisé par :

Yacine Benouniche – 1733049

Rendu à :

Adel Belkhiri

Le 04/12/2018

Tests de performance

Performance avec et sans répartiteur de charge

Afin de s'assurer de ne pas considérer des valeurs extrêmes ou aberrantes dans mes résultats, chaque test a été exécuté à trois reprises. Ainsi, pour ce qui est de la performance du service Web simple nous avons comme résultat :

```
('nb reponses recues: 24', 'nb rejets: 6', "temps d'execution : 6584.0 ms")
('nb reponses recues: 24', 'nb rejets: 6', "temps d'execution : 6607.0 ms")
('nb reponses recues: 24', 'nb rejets: 6', "temps d'execution : 6559.0 ms")
```

Concernant la performance du service Web avec répartition de charge :

```
('nb reponses recues: 30', 'nb rejets: 0', "temps d'execution : 4011.0 ms")
('nb reponses recues: 30', 'nb rejets: 0', "temps d'execution : 4012.0 ms")
('nb reponses recues: 30', 'nb rejets: 0', "temps d'execution : 4014.0 ms")
```

Ainsi on peut recapituler ces résultats dans le tableau suivant :

	Service Web simple	Service Web avec répartition de charge
Nombre de tests	30	30
Nombre de succès	24	30
Temps moyen par réponse (en ms)	6583.33	4012.33

Figure 1 : Tableau récapitulatif des résultats des différents tests réalises

On remarque qu'en déployant un service web simple celui-ci est saturé lorsque 30 requêtes lui sont envoyé en simultané et ne peut satisfaire les 30 requêtes, ainsi 6 requêtes sont rejetées et le temps de réponse moyen pour les 24 requêtes réussies est de 6.583 secondes (moyenne pour les 3 test effectués). Cependant si le système est un service Web avec répartition de charge avec 2 instances on remarque directement que les 30 requêtes sont satisfaites et le temps de réponses moyen et nettement inférieur à celui du service web simple avec environ 4 secondes par réponse.

Question 1 :

Heat :

Heat est un composant Openstack qui sert à orchestrer la configuration du système à partir de gabarit, appelés aussi des recettes. Heat permet ainsi de créer un système de nuages, dynamique ou non, avec les différents paramètres et ressources spécifiés dans le fichier de configuration. De plus Heat permet de créer ces ressources si elles n'existent pas. Ainsi si nous voulons par exemple assigner une IP flottante à notre nuage, Heat nous permet de faire cela sans avoir à passer par l'interface Horizon ou encore en ligne de commande. L'adresse IP sera créée et allouer à l'instance spécifiée à la création du système.

Neutron :

Neutron permet quant à lui de définir la connectivité de notre système. Ainsi grâce à ce composant nous pouvons créer, définir et modifier un réseau, les adresses IP utilisées qu'elles soient statiques ou flottantes. Ce composant permet également de détecter une intrusion par le biais de pare-feu. Enfin, on peut définir un LoadBalancer, un équilibreur de charge afin de gérer de façon optimal les ressources de notre système.

Nova :

Nova est le composant responsable de la création, et de la gestion a posteriori, des différentes instances créées. Nova est un ensemble de daemons qui procure un accès aux différents nuages crées antérieurement. Nova reste un petit peu limité par sa compatibilité avec les différents conteneurs pour la virtualisation. En effet Nova ne supporte pas tous les types de conteneurs. De plus ce composant permet de fixer des quotas d'utilisation aux instances et de contrôler le trafic.

Glance :

Glance nous permet de récupérer l'image virtuelle que nous désirons créer. Ainsi lorsque nous spécifions le nom de l'image que l'on va utiliser doit être connu par Glance et nous fixons donc cela comme contraintes dans le gabarit Heat utilisé. Glance nous permet aussi de créer de nouvelles images ou au contraire d'en retirer et également de modifier une image au besoin.

Horizon :

Horizon peut être vue comme la porte d'entrée à Openstack. On peut certes accéder et tout exécuter en lignes de commandes cependant il est beaucoup plus agréable d'avoir à sa disposition une interface web pour gérer ses nuages et c'est ici qu'Horizon intervient. Grace à Horizon nous pouvons monitorer l'état de nos nuages, notre réseau, créer des instances avec ou sans gabarit Heat, associer des adresses IP flottantes et beaucoup d'autres fonctionnalités très intéressantes.

Question 2 :

OS::Heat::ResourceGroup :

Pour ce TP, cette ressource nous a servi à créer 2 serveurs avec son attribut *count* et de définir le type de ressource que l'on voulait justement crée 2 fois avec son attribut *ressource_def.*

OS::Neutron::HealthMonitor :

Le HealthMonitor nous a quant à lui servit à s'assurer de l'activité de notre instance en lui envoyant un message toutes les 15 secondes et à avertir l'utilisateur/développeur de l'inactivité d'une instance en cas d'absence de réponses après 2 essais.

OS::Neutron::Pool :

La ressource Pool nous a permis de définir la façon avec laquelle notre système allait répartir la charge entre les différentes instances.

OS::Neutron::LoadBalancer :

Cette ressource nous sert pour établir la communication entre le monde extérieur et nos 2 instances (comme par exemple avec les requêtes http envoyées par le script de test)

OS::Nova::Server :

C'est l'une des ressources les plus importantes lors de ce TP, elle permet de gérer la machine virtuelle, autrement dit d'instancier la machine virtuelle avec les paramètres souhaités.

Question 3 :

1) Le nom de cette ressource est **OS::Heat::AutoScalingGroup**. Cette ressource a besoin de connaitre le type de ressource qu'elle doit créer, la valeur minimale et maximale du nombre de ressources créées.
2) La ressource OpenStack qui permet de :
    a. Lancer une alerte lorsque le taux d'utilisation du CPU atteint des seuils prédéfinis est : **OS::Ceilometer::Alarm**. Cette ressource doit connaitre le nom de la métrique à utiliser et son threshold, autrement dit quelles valeurs déclenchent l'alarme.
    b. Ajuster automatiquement le nombre de machines virtuelles en fonction de ces alertes : **OS::Heat::ScalingPolicy**. Pour fonctionner cette ressource nécessite le type d'ajustement à réaliser (dans notre cas ça serait 'change_in_capacity'), l'Id du groupe à ajuster et enfin le nouveau nombre de machine virtuelle à utiliser. Cette ressource a pour attribut l'url de l'alarme ainsi le autoscallingGroup sait quelle policy appliquer dépendamment de l'alarme.