École Polytechnique Montréal
Département de Génie Informatique et Génie Logiciel

INF8480
Systèmes Répartis et Infonuagique

Travail pratique 3
Initiation aux services de l'infonuagique

Soumis par
Ibrahima Séga Sangaré (1788085)
Mohamed Djemai (1911666)

Le 4 décembre 2018

Tests de performance :

- ### Sans répartiteur de charge :

Dans la première partie, un seul serveur s'est chargé de communiquer les réponses au 30 requêtes générées par le script se trouvant dans le fichier *test_performance.py*. Nous avons effectué une dizaine de fois le script pour observer les possibles variations entre les valeurs de temps moyen pris par le serveur pour ces réponses. La moyenne de temps que nous avons observée s'élève à **5.25 secondes**. Le fichier de configuration correspondant est celui nommé *simple_server.yaml* Si vous voulez reproduire les tests, il faudra faire attention à l'adresse IP flottante utilisée dans le fichier *test_performance.py* et refaire l'association d'adresse IP flottante. Il est possible qu'elle ait changée.

- ### Avec répartiteur de charge :

  Ensuite, nous avons répété les tests de la première partie avec un répartiteur de charge distribuant les tâches sur 2 instances de serveurs. La moyenne de temps que nous avons observée s'élève à **4.00 secondes**. Le fichier de configuration correspondant est celui nommé *load_balancing.yaml*.

Comme il est possible de le constater, le répartiteur de charge améliore grandement les performances du serveur en diminuant le temps moyen pris par le service pour fournir des réponses. Bien que la charge soit répartie entre 2 serveurs, la différence n'est pas très grande même s'il s'agit d'une diminution de **24%**, ce qui n'est pas anodin. Étant donné que les requêtes ne sont pas très volumineuses, on aurait pu s'attendre à de tels résultats. Cependant, dans un scénario avec des requêtes faisant des demandes plus exigeantes, on pourrait retrouver des résultats plus différents en faveur des performances réalisées avec le répartiteur de charge.

Question 1:

Heat:
Heat est le cœur de fonctionnement du programme d'orchestration de OpenStack. Heat permet de lancer plusieurs composants d'infonuagique à travers juste un fichier texte. Il utilise une des instructions sous forme de texte comme un lange de programmation compréhensible par l'être humain et par la machine. Avec Heat on peut lancer des instances comme un serveur, créer des adresses IPs flottantes, etc..

Heat permet aussi de fournir un service d'autoscaling à travers la template. La template peut aussi spécifier les relations entre les différents composants constituant le système. Aussi il permet de gérer tout de cycle de vie de l'application (depuis le lancement jusqu'à la fermeture) [1].

Neutron:
Il permet l'implémentation de services et de bibliothèques associées pour fournir une abstraction de réseau, évolutive et indépendante de la technologie à la demande. Il fournit un API riche pur développer tous les types de réseaux possibles. Il permet aussi d'utiliser des plugins qui utilise les autres technologies. Ces plugins peuvent être inclus ou séparés [2].

Nova :
Nova est un composant de OpenStack responsable pour gérer les ressources de calcul. Il est compatible avec beaucoup de technologies de virtualisation (KVM, VMware, Xen). L'architecture de Nova est conçue pour évoluer horizontalement en rajoutant du matériel [3].

Cinder :
Cinder offre le service de stockage en mode bloc pour OpenStack. Cinder virtualise la gestion des périphériques de stockage en mode bloc et fournit aux utilisateurs finaux une API d'utiliser ces ressources sans savoir exactement où leur stockage est réellement déployé ou sur quel type de périphérique.

Horizon :
Horizon est le tableau de bord de OpenStack. Il s'agit d'une application web qui permet aux utilisateurs et aux administrateurs de gérer leurs clouds à travers une interface graphique [3].

Question 2 :

OS::Heat::ResourceGroup :
Crée une ou plusieurs ressources configurées de manière identique. Par exemple 3 serveurs identiques.

OS::Neutron::HealthMonitor :
Cette ressource crée un moniteur qui vérifie les noeuds fonctionnels. Un moniteur qui interroge les serveurs à intervalle régulier pour détecter ceux qui ne sont pas fonctionnels.

OS::Neutron::LoadBalancer :
Une ressource permet de créer et gérer des répartiteurs de charge. Les requêtes sont réparties de manière à partager équitablement la charge et minimiser le temps de réponse.

OS::Neutron::Pool :
Pour répartir la charge, il faut un groupe de machines, cette ressource crée un bassin de machines virtuelles identiques pour servir des requêtes.

OS::Nova::Server :
Une ressource serveur qui gèrent une instance de machine virtuelle.

Question 3 :

1)

Une ressource de Openstack permettant de modifier dynamiquement le nombre d'instances du serveur est la ressource OS::Heat::AutoScalingGroup. Il faut obligatoirement mentionner les paramètres suivants pour pouvoir définir une telle ressource : **max_size, min_size,** et **resource**. Le paramètre **max_size** permet de mentionner le maximum d'instances de ressources permis en même temps. Naturellement, le paramètre **min_size** indique le minimum d'instances permis. Ces deux premiers paramètres sont des entiers supérieurs ou égaux à 0. Le paramètre ressource permet d'indiquer le type de ressource que l'on souhaite avoir dans le groupe de ressource. On s'attend à une *map* définissant les propriétés de la ressource. Par exemple, on peut définir une ressource de type OS::Nova::Server.

2)

- Pour lancer une alerte lorsque le taux d'utilisation du CPU de vos machines atteint des seuils prédéfinis, il faut utiliser la ressource OS::Ceilometer::Alarm. Les paramètres nécessaires pour cette ressource sont les suivants : **meter_name** et **threshold**. Le paramètre **meter_name** est une chaine de caractères qui permet d'identifier l'alarme que l'on veut déclencher. Le paramètre **threshold** est le seuil contre lequel on compare le taux d'utilisation actuel atteint pour évaluer s'il faut déclencher ou non une alerte.

- Pour ajuster automatiquement le nombre de machines virtuelles en fonction de ces alertes, il faut utiliser la ressource OS::Heat::ScalingPolicy. Les paramètres obligatoires attendus sont les suivants : **adjusment_type, auto_scaling_group_id, scaling_adjustment.** Le paramètre **adjustment_type** est une chaine de caractères précisant le type de changements que l'on voudrait apporter (nombre de machines virtuelles dans notre cas) de manière absolue ou relative par rapport au paramètre qui sera ajustée. Il y a trois valeurs possibles pour ce champ : change_in_capacity, exact_capacity, percent_change_in_capacity. Le paramètre **auto_scaling_group_id** permet d'indiquer l'identifiant du groupe de resource (AutoScalingGroup), la politique s'applique. Le dernier paramètre, **scaling_adjustment**, est un nombre entier qui précise la magnitude de l'ajustement. Dans ce cas, il s'agirait d'un nombre entier pour augmenter ou diminuer le nombre de machines virtuelles.

Références

[1] https://wiki.openstack.org/wiki/Heat
[2] https://wiki.openstack.org/wiki/Neutron
[3] https://en.wikipedia.org/wiki/OpenStack
[4] https://docs.openstack.org/heat/pike/template_guide/openstack.html