INF8480

Systèmes répartis et infonuagique

Groupe laboratoire 03

TP3

Chloé Constantineau – 1720146

Véronique Demers – 1783901

4 décembre 2018

Tests de performance

Une machine

Nb Requête	Moyenne	Médiane	Min	Maximum	Erreur	Débit
30	9051	7143	512	21007	16.67%	1.4 req/sec

Deux machines avec distributeur de charge

Nb Requête	Moyenne	Médiane	Min	Maximum	Erreur	Débit
30	3584	3561	512	6639	0.00%	4.0 req/sec

Nous avons utilisé JMeter pour effectuer nos tests de performance. Il s'agit d'un logiciel libre qui, selon nous, nous donne une meilleure vue d'ensemble de la situation.

Selon les résultats obtenus, nous pouvons remarquer plusieurs choses. La première étant que le taux d'erreur est nul lorsque l'on utilise le répartiteur de charge et de 16.67% lorsqu'il n'y a qu'une machine sans distributeur de charge bien sûr. Ainsi, le répartiteur de charge offre une meilleure disponibilité du service, comme le prouve aussi le débit qui est à 1.4 requêtes/seconde pour le premier tableau contrairement à 4 requêtes/seconde pour le second tableau.

De plus, nous remarquons que la moyenne du temps d'exécution est près de 3 fois inférieur lorsqu'un répartiteur de charge est installé. On remarque aussi que le maximum de temps pour les requêtes est nettement inférieur avec un distributeur de charge, soit pratiquement 7 fois inférieur.

En conclusion, comme attendu, avoir plus d'une machine avec un distributeur de charge est nettement plus efficace et aide à réduire le taux d'erreur ainsi que le temps de réponse.

Questions

Q1)

HEAT

Heat est un système d'orchestration qui est basé sur du templating. En tant que système d'orchestration, il permet d'instancier et déployer des ressources. Il est aussi possible de gérer des mises à jour et de détruire des ressources. C'est ressources peuvent être des serveurs, des adresse IP flottantes, des volumes, des groupes de sécurité ou encore des utilisateurs.

Il s'agit d'un système simple puisque les requis du système sont décrits dans un fichier suivant un certain template. Ce template décrit tous les requis nécessaires. Il est facile à lire, comprendre, écrire et modifier; il peut en soit être traiter comme du code.

NEUTRON

Neutron est un service de connectivité réseau. Il est à utiliser à travers un API soit les Networking Services APIs d'OpenStack. Ces APIs incluent des services de répartition de charge, des services de sécurité, des services de gestion des ressources, des services qui gère la qualité, etc. Ainsi, Neutron est un regroupement de service réseau très variés qui permettent un contrôle facile et optimisé d'un projet infonuagique.

NOVA

Nova est un service de *computing*. Il permet d'instancier et de configurer des machines virtuelles, des serveurs bare-metal et possède aussi des fonctionnalités, bien que limité, peuvent supporter des systèmes de conteneur. Il s'agit donc d'un service qui offre du *computing* à travers de la virtualisation ou de la paravirtualisation.

SWIFT

Swift est un service d'entreposage d'objet (object storing). À ne pas confondre avec Glance qui est un service d'entreposage de block (Block Storage). Il s'agit d'un service qui est scalable. Ainsi, un bon exemple serait l'entreposage des credentials d'utilisateur, soit un objet contenant nom d'utilisateur et mot de passe. Comme il n'y a pas de nombre maximal d'utilisateur, Swift pourra entreposer les credentials et s'étendre si le nombre d'utilisateur est très grand.

KEYSTONE

Keystone est un service d'authentification. Ainsi, il facilite l'authentification de client, la découverte de service et l'autorisation multi-locataire. Un bon exemple de son utilisation serait de faire appel à l'API d'authentification pour vérifier si les *credentials* d'un utilisateur sont valides, suite à quoi il aura accès au système ou non selon la réponse.

Q2)

OS::Heat::ResourceGroup

Permet de créer une ou plusieurs ressources imbriquées configurées de manière identique. Dans notre cas, nous décrivons le type de serveur Nova qui nous concerne.

OS::Neutron::HealthMonitor

Il s'agit d'une ressource qui monitor l'état de nos serveurs, s'ils sont en vie ou non (up or down).

OS::Neutron::Pool

Le pool représente un groupe de nœuds. Dans notre cas, il s'agit de serveurs. Il indique aussi quel type d'algorithme de répartition de charge utilisé (ROUND_ROBIN) et avec quel protocole réseau procéder (HTTP).

OS::Neutron::LoadBalancer

Service qui permet de diriger la charge à travers les serveurs. Il est utilisé conjointement avec le pool décrit plus haut pour connaître les serveurs où répartir et leurs ports.

OS::Nova::Server

Représente une machine virtuelle lancée sur le nuage Openstack avec les paramètres décrits dans le template. Il s'agit de notre serveur web en Python.

Q3)

1) Nous pouvons utiliser soit AWS :: AutoScaling :: AutoScalingGroup qui est associé avec Amazon Web Services ou encore OS :: Heat :: AutoScalingGroup qui est un logiciel libre avec OpenStack.
Avec ce service, les serveurs peuvent être instanciés dynamiquement en fonction de la demande.

2)

a. OS :: Ceilometer :: Alarm

   Ressource qui permet de définir une alarme sur un seuil d'utilisation du CPU.

   Paramètres :
   - statistic (string):
   - evaluation_periods (number): Nombre de période d'évaluation du seuil critique
   - period (number): Temps de la période d'évaluation du seuil critique
   - threshold (number): Seuil critique qui déclenche le début de la période d'évaluation
   - alarm_action : Action à prendre lorsque l'alarme est déclenchée
   - comparison_operator (string): Opérateur de comparaison du seuil critique.

   Ressource qui permet d'ajuster le nombre de machine virtuelle en fonction d'une alerte.

b. OS :: Heat :: ScalingPolicy ou encore AWS :: AutoScaling :: ScalingPolicy

   Il est preferable de rester avec OS :: Heat :: ScalingPolicy si l'on utilise OS :: Heat :: AutoScalingGroup.

   Paramètres :
   - adjustement_type (string) : Le type d'ajustement à apporter.
   - auto_scaling_group_id (string) : Le ID du groupe qui sera affecté par l'ajustement.
   - scaling_adjustement (number) : La valeur voulue pour l'ajustement.