TP3 - Initiation aux services de l'infonuagique

INF4410 - Systèmes répartis et infonuagique

Nom de l'étudiant : Félix La Rocque Carrier
Matricule : 1621348

Nom de l'étudiant : Mathieu Gamache
Matricule : 1626377

Polytechnique Montréal
5 décembre 2016

Tests de performance

Performance avec et sans répartiteur de charge

En utilisant l'outil JMeter, nous avons monté un Plan de Test lançant 50 requêtes simultanées et compilant les diverses réponses dans un rapport. Celui-ci nous montre les données suivantes.

50 Samples avec une instance:

Sample	Moyenne	median	90%	95%	99%	Min	Max	Error	Throughput
50	15637	21003	21007	21008	21017	760	21017	58%	2.4
50	15646	21002	21005	21007	21011	761	21011	58%	2.4
50	15649	21002	21005	21005	21010	764	21010	58%	2.4

50 Samples avec Load Balancer:

Sample	Moyenne	median	90%	95%	99%	Min	Max	Error	Throughput
50	9915	9714	18717	19429	20947	762	20947	0%	2.4
50	9712	9708	17165	17916	18703	763	18703	0%	2.7
50	9892	9706	18719	19425	20186	764	20186	0%	2.5

Comme nous pouvons le voir, en utilisant qu'une seule instance, nous avons premièrement un grand taux d'erreur (HTTP 502). Celle-ci est dû au fait que nous avons un petit timeout sur le serveur, lançant ainsi une erreur (java.net.ConnectException: Connection timed out: connect) lorsque le délais est dépassé.

Avec le Load Balancer, les choses sont beaucoup mieux. Aucune erreur et de meilleures temps de réponse en moyenne. Il en demeure q'une moyenne de 9 à 10 seconde pour une réponse de requête web est immense, surtout pour ne retourner qu'une string.

Questions

Question 1: Description composantes OpenStack

Heat : OpenStack Orchestration

Permet de définir des template qu'on peut ré-utiliser. Ces template contiennent l'infrastructure de l'application, les configurations des différentes ressources et comment ils sont reliés entre eux.

Il est aussi utilisé pour l'orchestration: être en mesure de déployer (créer et configurer les instances) les aussi faciliter les mises à jour (en modifiant le template, Heat va savoir les modification à effectuer) et aussi la destructions des ressources (va tout nettoyer).

Neutron: OpenStrack Networking

Permet de définir le réseau comme un service, il vise donner plus de contrôle sur comment la résautique dans le projet cloud ainsi qu'à le rendre plus facile d'utilisation. Plus spécifiquement il permet de définir la topologie (comment les ressources sont connecté entre-eux) ajouter des services réseaux (Firewall, LoadBalancer,...), contrôler le trafic (groupe de sécurité,...)

Nova: OpenStack Compute

Nova est responsable d'instancier et configurer les machines virtuelles (peut être des containers, utilisés différente technologie de virtualisation et paravirtualisation). Facilite la croissance horizontal: ajouter plus de serveurs est trivial!

Swift

C'est un système de sauvegarde d'objet (binaire) qui garantit une grande disponibilité et redondance. Les objets sont disponibles par projet.(équivalent à Amazon S3)

Ceilometer

Service de télémétrie: permet de recueillir plusieurs métrique: nombre d'instance lancé, taux d'utilisation (CPU, mémoire,...),... et permet de créer des alarmes liés aux métriques ce qui permet de réagir(envoyer email, scale up/down,...).

Question 2: Description des ressources

Heat::ResourceGroup:

- Permet de créer plusieurs fois une ressource identique en écrivant seulement 1 fois la configuration.

Neutron::HealthMonitor:

- Permet de surveiller le statut (en vie ou non) d'une ressource, dans notre projet on surveille les serveurs du pool.

Neutron::Pool:

- Représente un group de noeuds, dans notre cas nos serveurs et il indique l'algorithme de balancement pour déterminer quel resource utiliser.

Neutron::LoadBalancer:

- Permet le trafic d'être redirigé vers différents serveurs, est utilisé avec le Pool pour savoir comment répartir.

Nova::Server:

- Représente une machine virtuelle existant dans le nuage OpenStack, dans notre TP il lance un serveur python.

Question 3: Optimisation

1) Ressource permettant de modifier dynamiquement
   OS::Heat::AutoScalingGroup
   Cette ressource permet de créer dynamiquement la quantité voulue d'instance

- max_size et min_size : Pour spécifier l'intervalle d'instance que l'on veut pour scaler.
- resource : Définition d'une ressource (Instance)

2) Ressource permettant de lancer alerte et ajuster nombre machines

    - OS::Ceilometer::Alarm
      Permet de définir des alarmes du type :
        - Scale-up if the average CPU > 50% for 1 minute
        - Scale-down if the average CPU < 15% for 10 minutes

      Pour réagir à des changements dans les machines déployer pour en modifier la quantité et la grandeur

Paramètres:

- statistic: Type de formule pour calculer la limite (Minimum, Maximum,Moyenne)
- evaluation_periods: Nombre de périodes à évaluer
- period : Temps d'évaluation pour une limite
- threshold : Seuil d'évaluation
- alarm_actions : Actions à exécuter si l'alarme est déclenché
- comparaison_operator : Opérateur de comparaison pour le suils (plus grand, plus petit, égal, ...)

- OS::Heat::ScalingPolicy
  Ressource qui permet de scale les AutoScalingGroup

  Paramètres:
  - adjustment_type : Type d'ajustement à apporter au AutoScaling group (exemple: changement dans le nombre d'instances)
  - auto_scaling_group_id : Sur quel Auto Scaling Group la modification est à apporter.
  - scaling_ajustment : Taille de l'ajustement à apporter.