ONESHOT - CLOUD - SNR2

2021/2022

Contexte

Un vaste centre commercial veut s'installer dans une nouvelle localité et envisage mettre sur pied une infrastructure informatique pour chaque boutique couplé à un tableau de bord qui peut servir de renseignement à tout usager.
Les grandes lignes de l'infrastructure sont les suivantes :

1. Le centre commercial possède un hub central qui centralise toutes les informations sur les boutiques.
2. Chaque boutique a une position géographique principal à l'intérieur du centre commercial.
3. Chaque boutique doit enregistrer son catalogue (ce qu'elle vend et à quel prix) auprès du hub central.
4. Les clients pourront interroger le hub central pour connaître les boutiques qui ont des articles spécifiques accessibles pour leur porte-monnaie.
5. Le hub central peut fournir à chaque client, son historique de visite à chaque boutique à n'importe quel moment.

TAF:

On vous demande de proposer une architecture micro-services en gardant à l'esprit que l'architecture doit pouvoir passer à l'échelle (quel que soit le nombre de joueurs et de boutiques).

1. Faîtes un schéma de votre architecture. Identifier dans votre architecture, les services qui pourraient être des goulots d'étranglement ? Que préconisez-vous pour ses services ?
2. Pour implementer ses services, vous avez le choix entre des containers et des machines virtuelles comme unité d'isolation. Donner deux avantages de chaque unité d'isolation par rapport à l'infrastructure à mettre sur pied.

3. La réglementation voudrait que le centre commerciale soit fermé les Dimanches et en semaine, ouvre entre 8h et 21h30 le reste des jours de la semaine. En supposant que vous déployer vos services dans des machines virtuelles de type `t2.micro` de AWS, quel serait le coût pour maintenir votre architecture pour la première année ? Sachant que les services restent inactif pendant les heures de cloture.

   Les 750h premières heures d'un `t2.micro` sont gratuites, puis facturé à $0.0116/hr\$$ *?*

Déploiement avec des containers

L'entreprise va adopter une solution basée sur les containers et voudrais un prototype afin de visualiser les communications entre les différents modules et fournir les spécifications réseau aux mecs de la sécurité. Pour cela, vous décider d'implémenter l'architecture minimaliste de la Figure 2.

Le centre de renseignement représente le hub central c.à.d qu'il enregistre les boutiques existantes avec leur catalogue. Chaque boutique est charactérisé par sa position dans le centre commercial et sa page d'accueil. La page d'accueil est minimaliste et doit juste contenir de facon non structurée, la position géographique et le catalogue.

Les clients peuvent interroger le centre de renseignement en envoyant sa position dans le magasin, un nombre d'articles souhaité et le budget à disposition. Le centre de renseignement vérifie si dans la liste de boutiques enregistrées, une boutique peut satisfaire la demande et renvoie la page d'accueil de la boutique la plus proche du client si une boutique correspond sinon l'informe de l'incapacité de répondre favorablement à sa requête. On considère que les stocks de chaque boutique sont illimités.

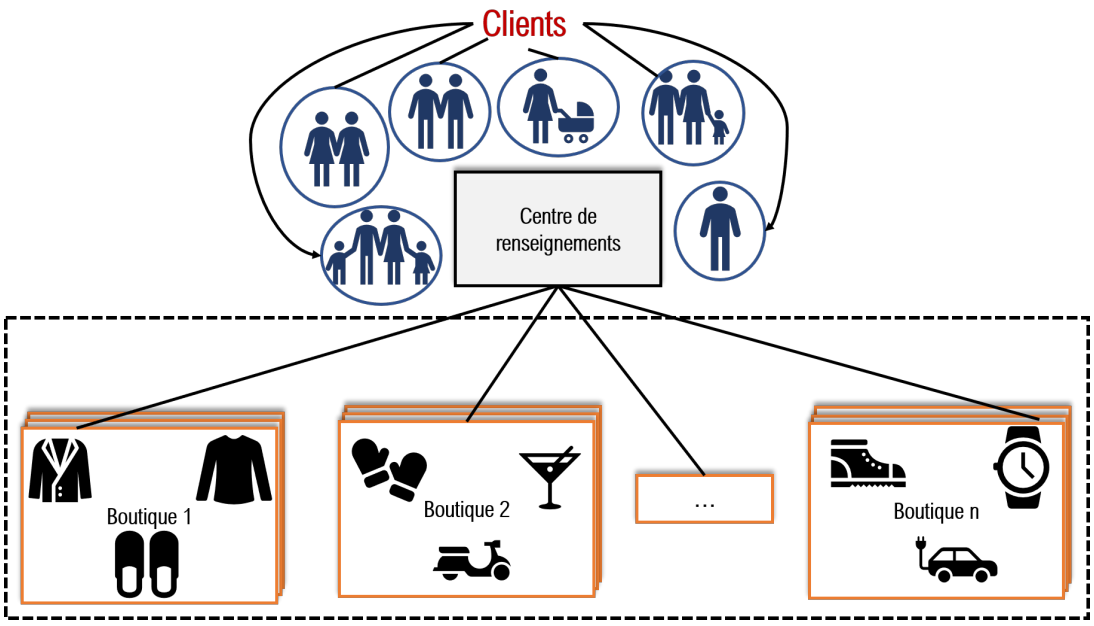

Figure 1: Architecture minimaliste à implémenter.

TAF: En prenant en compte les spécifications ci-dessus:

1. Implémenter et tester le service correspondant aux boutiques, clients, et centre de renseignement. Restez minimaliste: à l'initialisation d'une boutique, generér aléatoirement le catalogue, et position géographique, puis l'enregistrer auprès du centre de renseignement. Vos clients doivent intérroger le centre commercial

chaque 100ms avec de nouveaux paramètres et chaque 10s, devra afficher son historique de demandes et préciser pour chaque demande s'il y avait une enseigne disponible ou pas.

2. Écrire les Dockerfile de vos services pour les encapsuler dans des containers.
3. Tester votre implémentation en lançant un centre commercial avec 10 boutiques et 5 clients.
4. En moyenne sur 5mnin, quel est le taux de réussite des demandes des clients ?

Passage à l'échelle

L'entreprise voudrait identifier les composants critique votre déploiement. Pour cela elle vous demande de procéder à une surveillance spéciale. En effet, elle vous demande d'implémenter un composant qui interrogera chaque 10ms le centre de renseignement et calculer le temps de traitement de la requête. Le composant devra afficher (logger) les temps de réponse dans sa console (ou à un autre endroit où il pourra être visualisé).

TAF:

1. Implémenter ce composant et décrire son Dockerfile pour pouvoir l'exécuter dans un container. (Vous pouvez simplement afficher les temps de réponse et les récupérer avec `docker logs`)

2. Démarrer le composant et comparer les temps de réponse pour un centre commerciale avec 10 et 20 boutiques pour 25 clients. Commenter vos résultats.

3. Les observations montrent qu'un container pour le centre de renseignement peut rapidement devenir un goulot d'étranglement. Si possible, rendez votre centre de renseignement scalable c.-à-d., créer plusieurs instances et implémenter un load balancer pour répartir les requêtes des clients entre ces différentes instances (comme sur la Figure **??**).

4. Re-itérez le test du temps de réponse pour des parties de 10, 20, 30 boutiques pour 25 clients. Qu'observez-vous ?

5. Démarrer une dernière grosse simulation en lancant un cluster simultané de 3 centres commerciaux (qui possède 15 boutiques et 25 clients chacun) de telle sorte que chaque cluster évolue dans son propre réseau docker. Qu'observez-vous avec votre composant de monitoring ?

6. Si votre budget d'erreur est de 50000$, et que l'entreprise se considère sans faille si elle arrive à répondre à 99.999% des requêtes des clients. Un problème P rend votre cluster incapable de servir 0.0005% des requêtes. Quel pourcentage du budget d'erreur sera utilisé pour regler P ? Combien va t'on dépenser ?

Votre rendu sera composé des réponses aux questions théoriques, codes des services, les Dockerfile, le .yaml, et les commandes utilisées par docker-compose si vous l'avez utilisé (+ les commandes que vous avez exploitées pour créer les réseaux).

COURAGE !!!

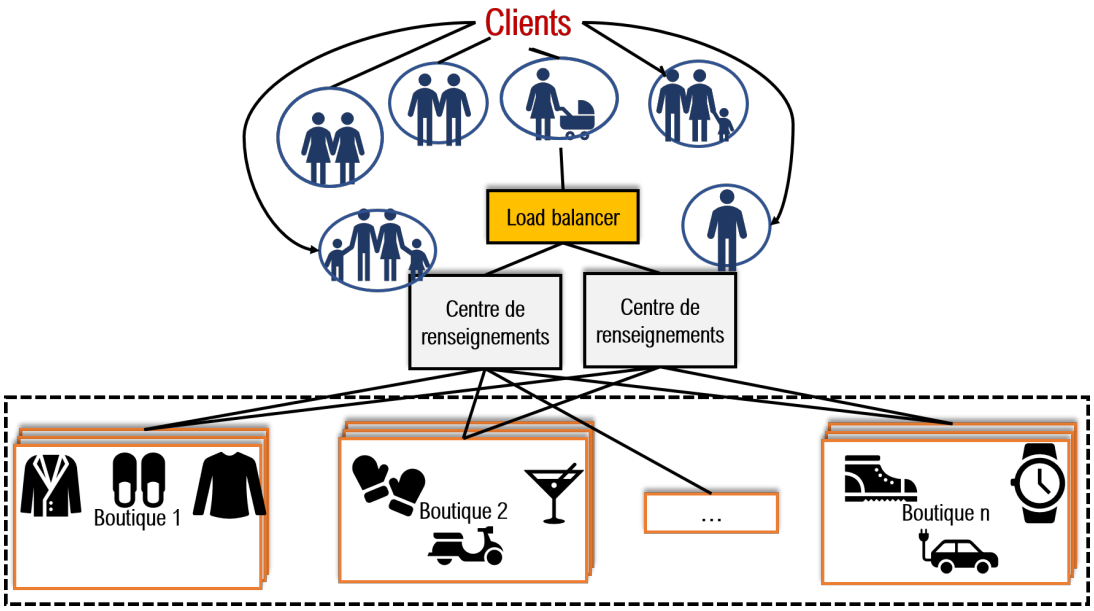

Figure 2: Architecture minimaliste à implémenter (avec loadbalancer).