Résumé CLD - TE2

CLD

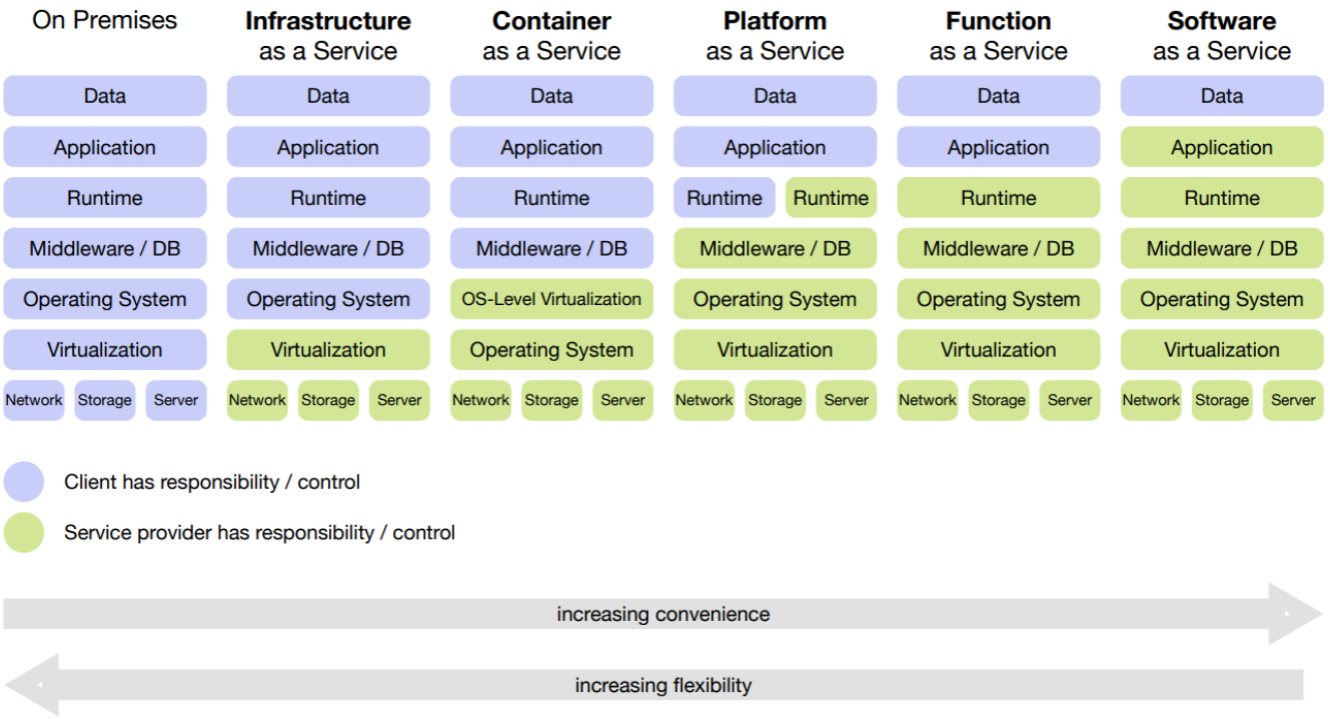
PaaS, Storage as a Service, NoSQL, Container cluster management, IaaS and labs

Résumé du document

Table des matières

1. PaaS

Platform-as-a-Service (PaaS) permet aux développeurs de déployer des applications scalables dans le cloud, en se concentrant sur le code de l'application et en déléguant le reste au fournisseur de services cloud.



Dans un PaaS nous avons donc la main uniquement sur l'application et les données, le reste est géré par le fournisseur de service cloud. Le PaaS existe aussi pour l'**IoT** ainsi que l'**IA**.

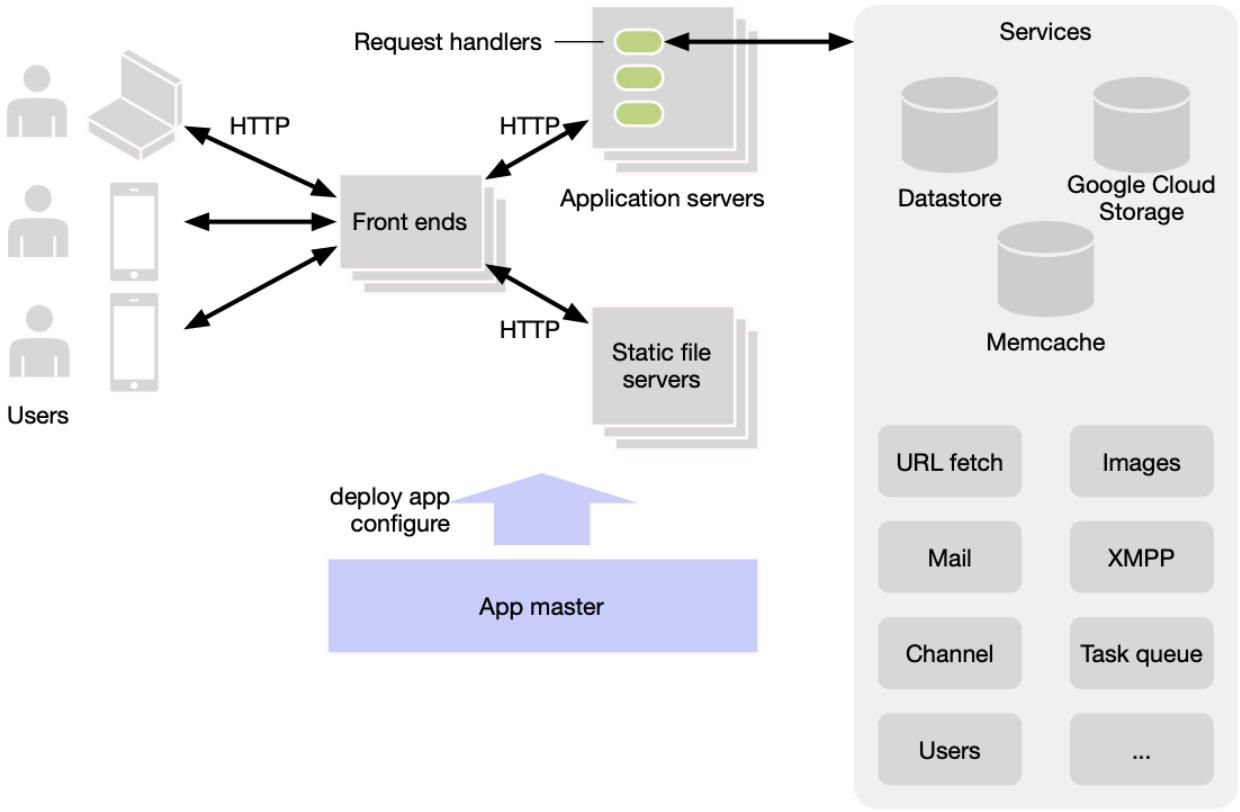
Le cloud provider gère donc tout ce qui concerne :
- provisionnement automatique des machines virtuelles (VM)
- maintenance du système d'exploitation
- installation et maintenance du serveur d'applications web
- installation de l'application et déploiement des mises à jour de l'application
- provisionnement des load-balancers, vérification de l'état des VM
- mise à l'échelle automatique avec ajustement adaptatif
- bases de données et data stores gérés

1.1. Risque de verrouillage des fournisseurs

Dans de nombreux cas les solutions de PaaS sont propriétaires (architectures et API) et ne sont pas compatibles entre elles. Il est donc difficile de migrer d'un fournisseur à un autre. Il est donc important de bien choisir son fournisseur de PaaS ou d'opter pour une solution open-source.

1.2. Google App Engine

Google App Engine est un PaaS qui permet de déployer des applications web et mobiles. Il est possible de déployer des applications web et mobiles sans se soucier de la gestion des serveurs.

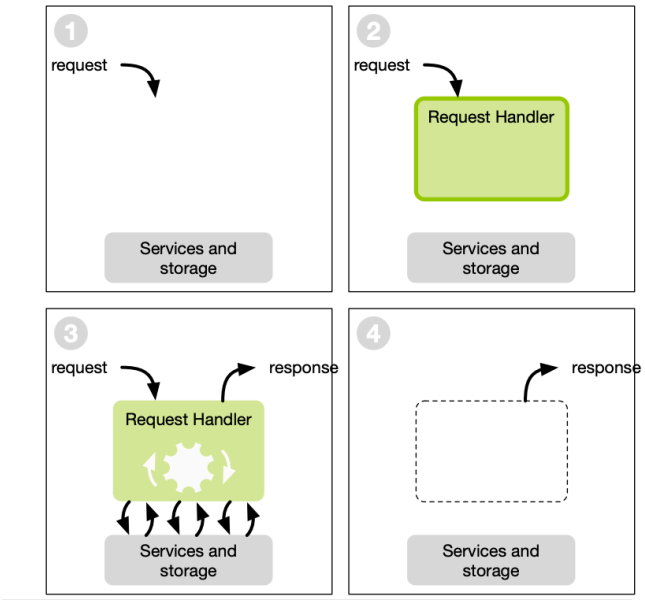


1.2.1. Architecture

- Multi-tenant : La plateforme de Google déploie automatiquement le code de votre application et ajuste l'échelle pour s'adapter à la charge.
- Pas de contrôle des serveurs, équilibreurs de charge ou groupes de mise à l'échelle automatique.
- La plateforme contrôle et partage les ressources entre tous les locataires.
- Frontend : Reçoit les requêtes HTTP, identifie le locataire et l'application, et route la requête.
- Effectue des vérifications de l'état des instances d'application et assure l'équilibrage de la charge.
- Serveurs d'application : Exécutent le code de l'application dans des instances contenaires ou machines virtuelles.
- Fournit le système d'exploitation et l'environnement d'exécution (Java, Python, etc.).
- App master : Gère les instances d'application, déploie le code, remplace les instances défaillantes et ajuste automatiquement le nombre d'instances.

1.2.2. Request handler

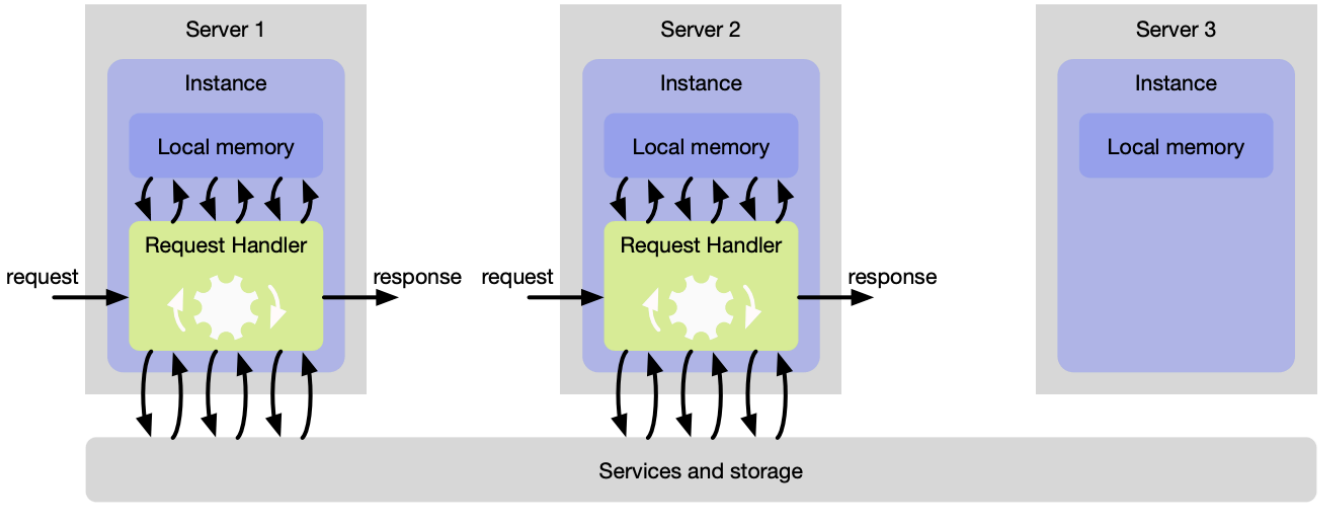
- Request Handler : Code responsable de créer une réponse HTTP à une requête HTTP.
- Exemple : Java servlet.
- Cycle de vie dans App Engine :
  1. Une requête arrive.
  2. Le Request Handler est créé et reçoit la requête.
  3. Le Request Handler crée la réponse, en utilisant éventuellement d'autres services cloud et stockage cloud.
  4. Une fois la réponse envoyée, App Engine peut supprimer le Request Handler de la mémoire.
- Le Request Handler doit être sans état pour que ce cycle de vie fonctionne (sauf en cas de mise à l'échelle manuelle).



- Lorsque le nombre de requêtes augmente, App Engine alloue des Request Handlers supplémentaires.
- Tous les Request Handlers traitent les requêtes en parallèle (mise à l'échelle automatique).

1.2.3. App instance

- App Instance : Conteneur ou machine virtuelle qui exécute le code de l'application.



- Le cycle de vie du Request Handler peut être satisfaisant, mais il n'est pas pratique d'instancier et de détruire un programme pour chaque requête.
  - L'initialisation d'un programme est coûteuse en temps, surtout pour la mémoire locale.
- Les instances sont des conteneurs dans lesquels vivent les Request Handlers.
  - Une instance conserve la mémoire locale initialisée pour les requêtes suivantes.
- Une application a un certain nombre d'instances allouées pour traiter les requêtes.
  - Le front-end distribue les requêtes parmi les instances disponibles.
  - Si nécessaire, App Engine alloue de nouvelles instances.
  - Si une instance n'est pas utilisée pendant un certain temps, App Engine la libère.

On peut voir les app instances dans le tableau de bord de Google Cloud Platform.

Prix: vous êtes facturé en fonction du temps d'allocation d'une instance (parmi d'autres frais).

1.2.4. Type de scaling

Lors du téléchargement d'une application, vous spécifiez le type de mise à l'échelle dans un fichier de configuration. Le type de mise à l'échelle contrôle comment la plateforme crée les instances.

- **Mise à l'échelle manuelle** : Vous spécifiez manuellement le nombre d'instances à instancier. Les instances fonctionnent en continu, permettant à l'application d'effectuer une initialisation complexe et de s'appuyer sur l'état de sa mémoire au fil du temps (elles peuvent **stateful**).
- **Mise à l'échelle basique** : La plateforme commence avec zéro instance et crée une instance lorsqu'une requête est reçue. L'instance est arrêtée lorsque l'application devient inactive. L'application doit être **stateless**. Le développeur contrôle directement deux paramètres : le **nombre maximum d'instances** et le **délai d'inactivité** pour supprimer les instances.
- **Mise à l'échelle automatique** : La plateforme décide quand créer et supprimer des instances en utilisant des algorithmes prédictifs basés sur le taux de requêtes, les latences de réponse, et d'autres métriques de l'application. Le développeur n'a qu'un contrôle indirect en ajustant certains paramètres. L'application doit être stateless. Ce type de mise à l'échelle est celui par défaut.