Python (https://cloud.google.com/python/?hl=fr)

Premiers pas avec Django

Vous pouvez facilement commencer à développer des applications Django qui s'exécutent sur GCP. Comme les applications créées sont exécutées sur l'infrastructure sur laquelle sont basés tous les produits Google, vous avez la garantie d'un dimensionnement efficace, à même de desservir l'ensemble de vos utilisateurs, qu'ils soient une poignée ou plusieurs millions.

Plates-formes d'hébergement

Il existe quatre options principales pour déployer Django sur GCP.

Option de déploiement de Django	À privilégier si vous souhaitez bénéficier des avantages suivants	À éviter dans les contextes suivants	Premiers pas
Environnement standard App Engine	• Configuration minimale • Pas de maintenance du serveur • Évolutivité facile	• Bibliothèques système indisponibles dans l'environnement Python standard App Engine	Django dans l'environnement standard App E (https://cloud.google.com/python/django/ap hl=fr)

Option de déploiement de Django	À privilégier si vous souhaitez bénéficier des avantages suivants	À éviter dans les contextes suivants	Premiers pas
Environnement flexible App Engine	• Plupart des avantages d'App Engine • Bibliothèques système et bibliothèques Python qui en dépendent • Environnements d'exécution Docker personnalisés	• Contrôle de toute la machine virtuelle (en dehors du conteneur Docker de l'application)	Django dans l'environnement flexible App Eng (https://cloud.google.com/python/django/m vms?hl=fr)
Google Kubernetes Engine	• Conteneurs Django dans un environnement de microservices • Kit pour concevoir une plate-forme personnalisée basée sur des conteneurs	• Plate-forme en tant que service (PaaS) dotée de fonctionnalités complètes (pour une PaaS basée sur des conteneurs, utilisez l'environnement flexible)	Django sur GKE (https://cloud.google.com/python/django/ku engine?hl=fr)
Compute Engine	• Infrastructure en tant que service (IaaS) familière utilisant des VM • VM Windows	• Environnement sans serveur, ne nécessitant pas de configurer votre propre infrastructure	Django sur Google Cloud Platform Marketpla (https://cloud.google.com/marketplace/solu launchpad/djangostack?q=django&hl=fr)

Bases de données

Le mappeur objet-relationnel (ORM, Object Relational Mapper) Django fonctionne mieux avec une base de données SQL traditionnelle. Si vous démarrez un nouveau projet, Cloud SQL (https://cloud.google.com/sql/?hl=fr) est un choix idéal. En quelques clics, vous pouvez créer une base de données MySQL ou PostgreSQL, dont la gestion et le scaling sont entièrement pris en charge par Google.

Vous pouvez également utiliser d'autres bases de données SQL si vous souhaitez les gérer vous-même sur Compute Engine ou un autre service.

Dans certains cas, l'utilisation d'une base de données NoSQL est nécessaire, selon les besoins d'évolutivité ou de pertinence pour votre modèle de données. L'utilisation de l'ORM Django avec une base de données NoSQL est possible, mais peut être difficile, et impose certaines restrictions. Par exemple, de nombreux types de jointures de base de données peuvent être exprimés dans Django, mais ils ne sont pas compatibles avec Cloud Datastore ni d'autres bases de données NoSQL telles que MongoDB.

Il est possible d'adopter une approche mixte SQL/NoSQL qui utilise différentes bases de données pour divers types de données.

Pour une solution NoSQL gérée et extrêmement évolutive, il est possible d'utiliser Cloud Datastore (https://cloud.google.com/datastore/?hl=fr), une base de données non relationnelle dont le scaling est souvent plus performant que celui d'une solution SQL.

Si vous choisissez d'utiliser une base de données MongoDB, vous pouvez la déployer à l'aide de GCP Marketplace (https://cloud.google.com/marketplace/solution/click-to-deploy-images/mongodb?hl=fr) et la gérer vous-même, ou vous pouvez utiliser le service d'hébergement géré MongoDB fourni par mLab (https://mlab.com/).

Pendant de nombreuses années, l'approche la plus populaire pour faire fonctionner l'ORM Django avec des solutions NoSQL consistait à utiliser la suite Django non-rel (https://github.com/django-nonrel/django-nonrel), mais ce projet est une branche de Django qui n'a pas été mise à jour avec le socle. Un nouveau projet appelé Djangae (https://github.com/potatolondon/djangae) a vu le jour. Il fournit un backend ORM Django pour Cloud Datastore sans diviser Django. Bien que Djangae semble extrêmement prometteur,

cette solution ne prend pas officiellement en charge l'exécution de Django sur App Engine.

Caches

App Engine est livré avec un service intégré Memcached (https://cloud.google.com/appengine/docs/python/memcache/usingmemcache?hl=fr). Pour installer Memcached sur Compute Engine, vous pouvez utiliser GCP Marketplace (https://cloud.google.com/marketplace/solution/bitnami-launchpad/memcached?q=memcache&hl=fr). Pour installer Memcached sur Compute Engine ou GKE, utilisez l'image Docker Memcached (https://hub.docker.com/_/memcached/). De même, vous pouvez installer Redis à l'aide de GCP Marketplace (https://cloud.google.com/marketplace/solution/click-to-deploy-images/redis?hl=fr) ou de l'image Docker Redis (https://hub.docker.com/_/redis/).

Mettre des tâches en file d'attente

App Engine inclut une fonctionnalité de file d'attente de tâches intégrée (https://cloud.google.com/appengine/docs/python/taskqueue/?hl=fr) pour les tâches d'arrière-plan de longue durée. En dehors d'App Engine, utilisez le service Cloud Pub/Sub (https://cloud.google.com/pubsub/?hl=fr), extrêmement évolutif, qui peut être transformé en file d'attente de tâches à l'aide de la file d'attente de tâches Cloud Pub/Sub pour Python (psq) (https://github.com/GoogleCloudPlatform/psq).

RabbitMQ (https://cloud.google.com/marketplace/solution/click-to-deploy-images/rabbitmq?hl=fr) et Kafka (https://cloud.google.com/marketplace/solution/bitnami-launchpad/kafka?q=kafka&hl=fr) sont d'autres options populaires disponibles dans GCP Marketplace pour la mise en file d'attente de tâches. Il existe également des images Docker (https://hub.docker.com/) pour RabbitMQ et Kafka.

(https://developers.google.com/terms/site-policies?hl=fr). *Java is a registered trademark of Oracle and/or its affiliates.*

Dernière mise à jour : avril 10, 2019