DÉVELOPPER DES APPLICATIONS SCALABLES & RÉSILIENTES

avec Google Cloud Platform

Livre blanc réalisé par SKALE 5 et neoxia

“Datacenter is not a collection of computers: Datacenter is a Computer.”

Greg DeMichillie, Director of product management, Cloud Platform at Google

Google Cloud Platform

Introduction

Créer des applications qui sont à la fois résilientes et évolutives (scalables) est un objectif essentiel de toute architecture. Une application bien conçue doit pouvoir évoluer de manière transparente au fur et à mesure que la demande augmente et diminue, et être suffisamment robuste pour résister à la perte d'un ou plusieurs éléments d'infrastructure.

Dans ce document, vous apprendrez comment utiliser Google Cloud Platform pour construire des architectures évolutives et élastiques. Vous verrez comment ces principes sont facilement applicables à vos applications à travers l'exemple d'un déploiement concret d'un projet open source : la solution Opensource de gestion Redmine.

A propos de Redmine

Redmine est une solution développée en Ruby on Rails (RoR). Vous aurez donc l'occasion dans cet article de déployer pas à pas vous même cette application sur la plateforme Google Cloud Platform.

> **Ainsi, le cloud modifie substantiellement de multiples processus, patterns, pratiques, et approches tenus pour acquis.**

De nouvelles approches

Le cloud remet en avant des approches relativement anciennes de la construction d'architectures Internet hautement scalables. Il propose aussi de nouvelles approches qui modifient considérablement la façon de construire et de déployer des applications. C'est pourquoi, vous aurez sans doute le sentiment paradoxal que « tout a changé, mais que rien n'est vraiment différent ».

Ainsi, le cloud modifie substantiellement de multiples processus, patterns, pratiques, et approches tenus pour acquis. Il remet aussi en avant certains principes connus des architectures orientées service, principes qui deviennent plus fondamentaux encore qu'auparavant.

Les applications traditionnelles ont été construites dans un état d'esprit qui reflétait les préoccupations économiques et l'état de l'art en matière d'architecture de l'époque. Le cloud est porteur de nouvelles idées qu'il est nécessaire de s'approprier. Elles sont développées ci-dessous.

SOMMAIRE

LES CONCEPTS DU CLOUD

1. Qu’appelle-t-on Scalabilité et Résilience ?

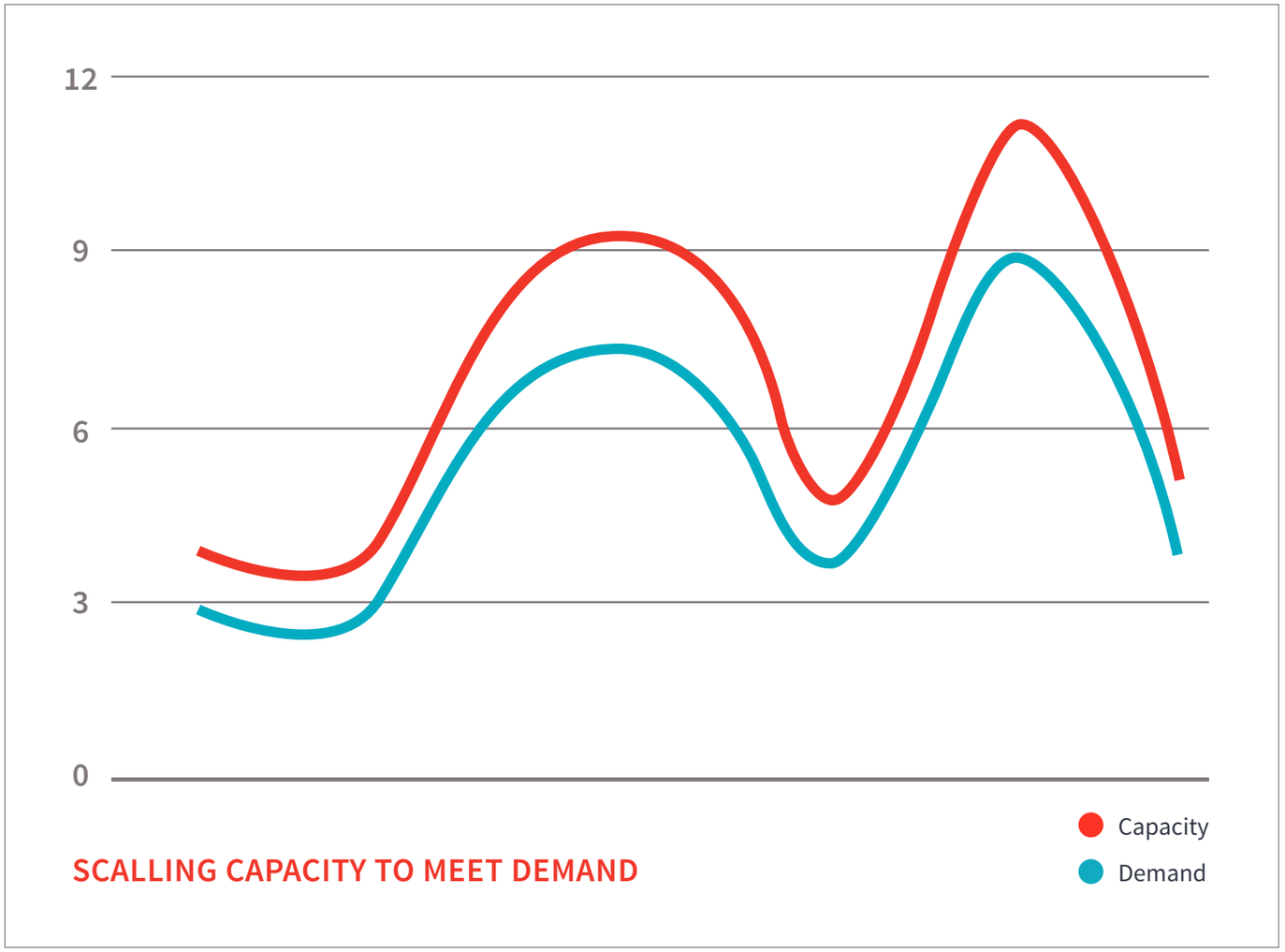
Scalabilité : ajustement de la capacité pour répondre à la demande

La scalabilité et la résilience sont deux caractéristiques essentielles d’une architecture d’un service. Il est donc important de bien comprendre les propriétés qu’elles englobent.

Une application Web dite scalable est une application en capacité de fonctionner de manière optimale avec 1 ou 1 000 000 utilisateur(s). La scalabilité d’un service désigne la capacité de l’application et de l’infrastructure à s’adapter automatiquement pour traiter un niveau de sollicitation variable.

La scalabilité de l’infrastructure s’obtient par ajout et suppression de capacité de traitement (VM, mémoire, cpu, stockage). Le schéma suivant montre comment une application évolutive **répond aux augmentations et aux diminutions de la demande.**

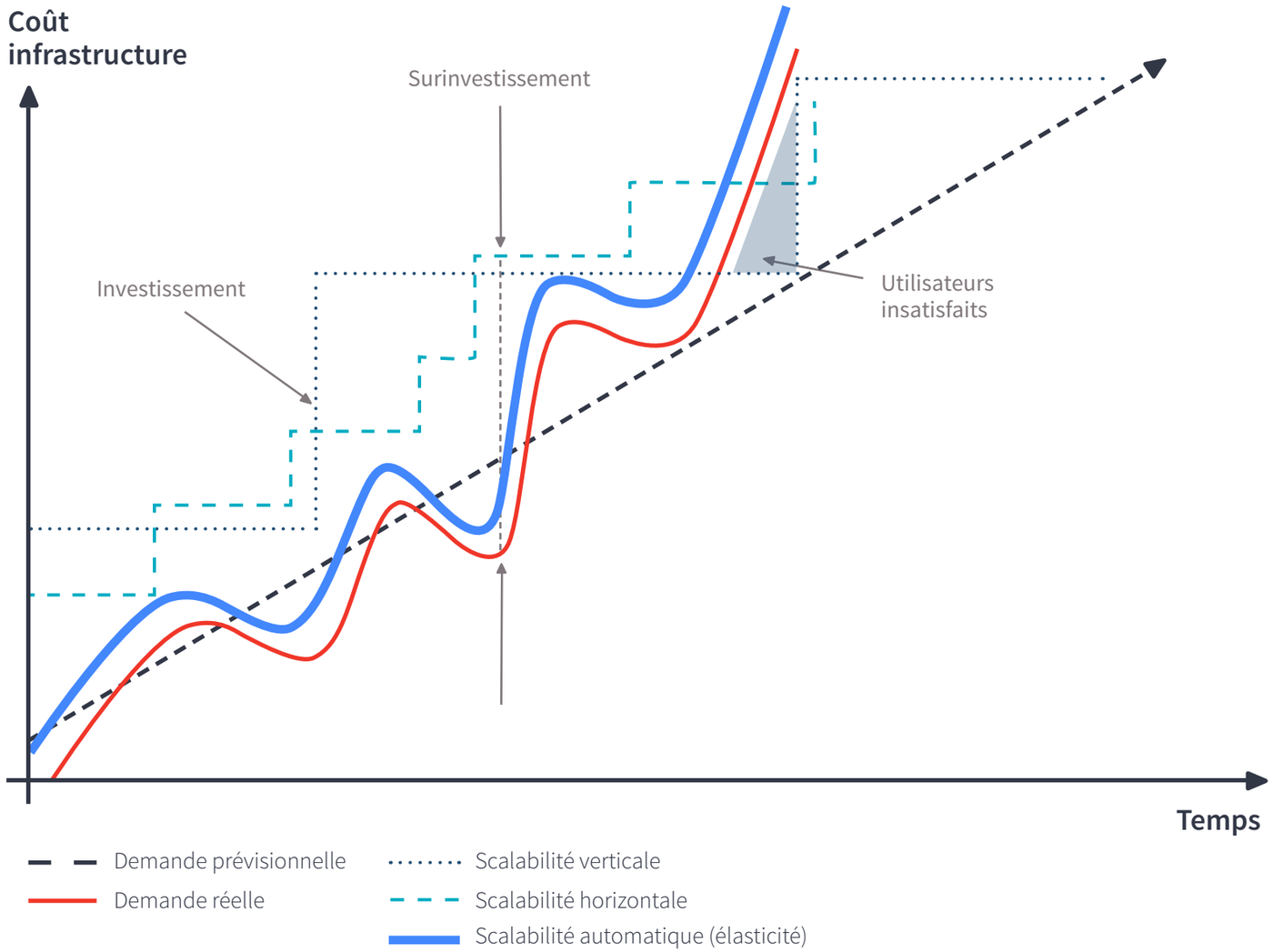
La capacité s’ajuste dynamiquement pour tenir compte des variations de la demande. Cette caractéristique, parfois appelée élasticité, permet de **garantir que vous ne serez facturés uniquement que des ressources que vous consommez.**



2. Qu’est-ce que l’Elasticité ?

SCALABILITÉ ET ÉLASTICITÉ

Le graphique ci-dessous présente les différentes approches qu’un architecte cloud peut mettre en œuvre pour améliorer la scalabilité des applications, et leur permettre de faire face à la demande.



Approche scale-up
(ou scalabilité verticale)

Cette approche consiste à ne pas prêter attention à la scalabilité de l’architecture applicative, et à investir lourdement dans des serveurs plus gros et plus puissants (scalabilité verticale), dans l’objectif de faire face à la demande. Cette approche fonctionne jusqu’à un certain point, mais peut s’avérer extrêmement couteuse et inefficace (ex: gestion des pics de charge, limite de la scalabilité verticale sur des composants techniques tel que les SGBDR, ...)

Approche scale-out
(ou scalabilité horizontale)

Cette approche consiste à élaborer une architecture possédant des propriétés de scalabilité horizontale, et à investir dans l’infrastructure par incréments successifs. La plupart des applications Web adoptent ce modèle, en distribuant leurs composants applicatifs, en répartissant et en fédérant leurs données, et en recourant aux architectures orientées services. **Cette approche est souvent plus efficace que l’approche par scalabilité verticale.** Elle nécessite cependant de prévoir régulièrement la demande, et d’augmenter par à-coups la capacité de l’infrastructure pour faire face à la demande anticipée.

2. Qu’est-ce que l’Elasticité ?

Avec les infrastructures traditionnelles, il est bien souvent nécessaire d’anticiper la capacité de traitement pour une période de plusieurs mois. Si vous sous-estimez la capacité, vos applications n’auront pas à leur disposition la puissance nécessaire pour faire face à une fréquentation inhabituelle, au risque d’entrainer l’insatisfaction de vos clients. Si vous la surestimez, vous gaspillerez probablement une part de vos budgets en ressources inutiles.
L’ajustement à la demande et l’élasticité sont caractéristiques (élasticité automatique) de l’approche cloud et **permettent à l’infrastructure de s’ajuster au plus près de la demande réelle, qu’elle croisse ou décroisse.** Ce faisant, les ressources sont utilisées au mieux, et les coûts sont optimisés.

L’élasticité est l’une des caractéristiques fondamentales du cloud. L’élasticité est la capacité d’ajuster les ressources à la hausse **ou à la baisse automatiquement.** Il est important de remarquer que c’est bien de l’élasticité que résulte à terme l’essentiel des apports du cloud. Un architecte cloud devrait faire sienne la notion d’élasticité, et l’intégrer dans ses architectures applicatives, afin de tirer le meilleur parti du cloud.

Traditionnellement, les applications sont construites pour des infrastructures figées, rigides et acquises à l’avance. En entreprise, l’acquisition et l’installation de serveurs sont relativement exceptionnelles, et ne font pas partie des opérations quotidiennes. C’est pourquoi la majorité des architectures logicielles ne tentent pas d’apporter une réponse aux problématiques de déploiement rapide, ou encore de réduction des besoins en hardware.

Le délai d’acquisition de nouvelles ressources et les investissements en amont sont à ce point élevés, que les architectes logiciels n’ont jamais ressenti le besoin de consacrer du temps et des moyens humains à l’optimisation de l’usage du hardware. Que le hardware sur lequel l’application s’exécute soit sous-utilisé semblait finalement acceptable. Obtenir de nouvelles ressources dans un délai de l’ordre de la minute était clairement impossible, aussi il n’est pas étonnant qu’en matière d’architecture, on ait pu passer à coté de la notion d’élasticité.
Avec l’avènement du cloud, cet état d’esprit a évolué. Ainsi, le cloud computing assouplit considérablement le processus d’obtention des ressources ; il n’est plus nécessaire de passer commande à l’avance, ou de ‘garder en captivité’ le hardware non-utilisé. Au contraire, les architectes cloud **peuvent obtenir les ressources en quelques minutes à peine,** ou même automatiser le processus d’acquisition des ressources, le tout en tirant partie de la scalabilité et de la réactivité naturelle du cloud. Bien entendu, ce schéma reste valable pour la restitution des ressources non-utilisées ou sous-utilisées.

> **“ L’élasticité est la capacité d’ajuster les ressources à la hausse ou à la baisse automatiquement. ”**

3. Résilience : conçu pour résister à la défaillance

On désigne par résilience la capacité d'un service à continuer de fonctionner malgré la défaillance d'un ou plusieurs éléments d'infrastructure (base de données, serveurs Web, accès réseau, ...).
La résilience désigne également la capacité du service à revenir dans un mode nominal de façon automatisée.

Une application pleinement résiliente exige une conception adaptée à la fois au niveau des développements (ex : gestion des états/sessions), ainsi qu'au niveau de l'infrastructure (ex : utilisation de LoadBalancer, bases de données répliquées, ...). Ce document se concentre principalement sur les aspects d'infrastructure sous-jacents à l'application.

Concevoir une architecture scalable et résiliente implique généralement :

- L'utilisation d'équilibreurs de charge (Load Balancer) pour surveiller les serveurs et répartir le trafic vers des serveurs qui peuvent mieux gérer les demandes.
- De distribuer les serveurs d'hébergement (ou VM) dans plusieurs Datacenters.
- De disposer d'une solution de stockage robuste et répliquée.
- De concevoir des applications sans état (dites « stateless ») ou de gérer les états/sessions avec des mécanismes distribués (ex : ferme de serveurs de sessions).

4. Google Cloud Platform: flexible et rentable

La mise en place de la scalabilité et de la résilience avec des architectures traditionnelles exigent souvent des investissements importants (ex : acquisition d'équilibreurs de charge de haute capacité, Data-center multiple, éléments de sécurité de firewall réseau ou applicatif, ...).

De plus, les infrastructures dites classiques (par opposition aux services cloud) impliquent souvent de choisir entre des dépenses excessives sur la capacité de serveur pour gérer les pics d'utilisation, ou un achat calculé en fonction des besoins moyen de puissance. Ce dimensionnement par la moyenne de charge induit un risque important sur les performances des applications et donc sur l'expérience de l'utilisateur lors des pics de trafic.

Google Cloud Platform propose un ensemble de services d'infrastructure permettant de doter rapidement vos applications d'une scalabilité et d'une résilience importante.

HÉBERGEZ VOS APPLICATIONS DANS LES DATACENTERS GOOGLE

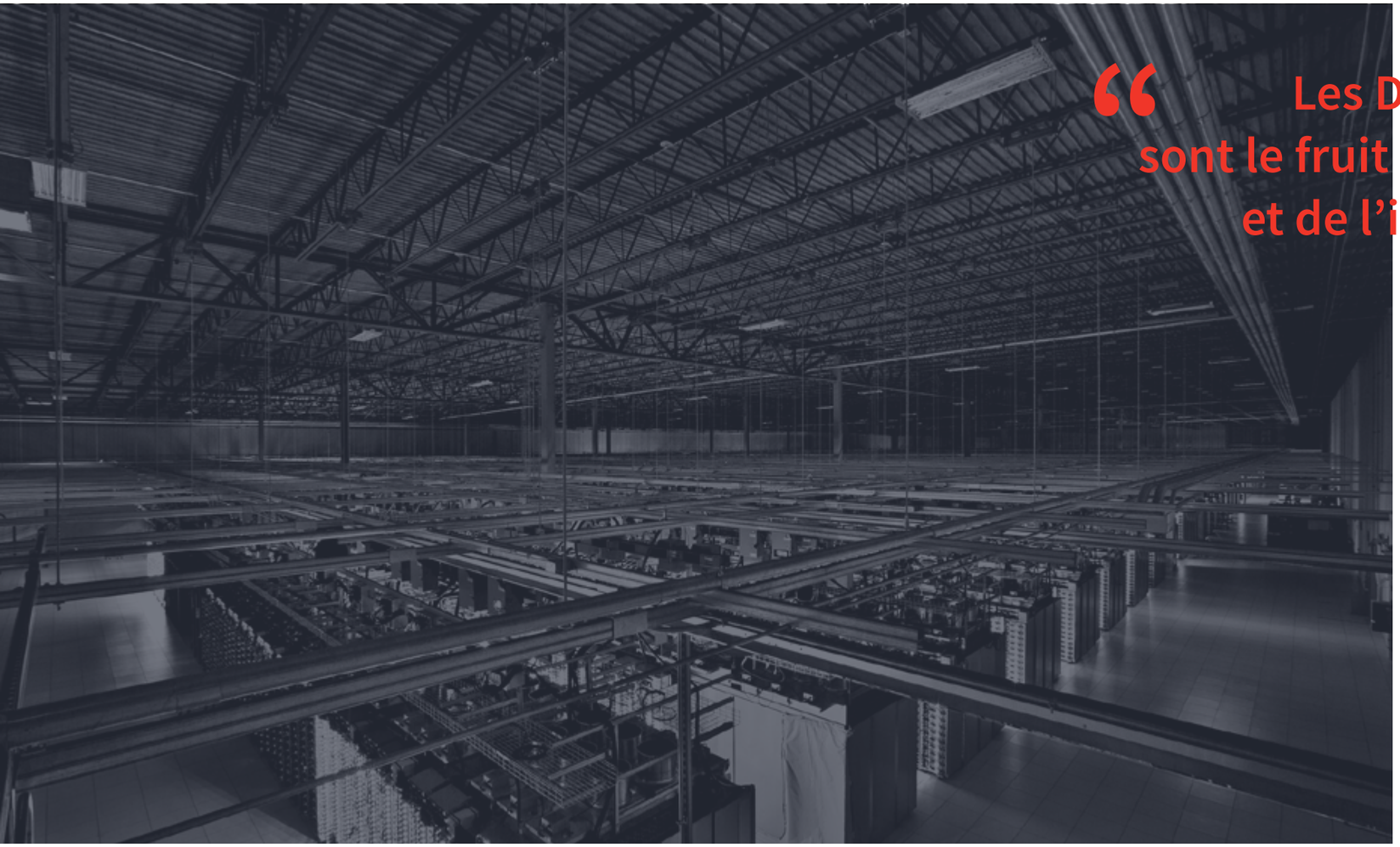
GOOGLE Cloud Platform repose sur les mêmes infrastructures
que les services globaux GOOGLE (Google.com, Gmail, Google Maps, ...).
Ces services qui délivrent plusieurs milliards de résultats de recherches
chaque seconde de manière extrêmement fiable attestent chaque jour
de la puissance colossale des infrastructures GOOGLE.

Avec GOOGLE Cloud Platform (GCP), il vous est désormais possible
d'héberger vos applications sur cette infrastructure et ainsi donner
le meilleur à votre SI ou services en ligne.

1. Le réseau Google

Google possède l'un des réseaux mondial des plus étendu et des plus avancé. Le Back-Bone du réseau de Google est basé sur plusieurs milliers de kilomètres de fibre optique entièrement dédiées à GOOGLE. Il utilise les dernières technologies de SDN (Software Defined Networking) et offre de manière native des services de mise en cache.
Pour exemple, GOOGLE opère directement plusieurs câbles de fibre optique sous les océans Pacifique et Atlantique : **utiliser le réseau WAN de GOOGLE, c'est donc avoir la garantie de bénéficier des meilleures bandes passantes et des latences les plus faibles possibles pour vos applications.**

2. Les Datacenters Google



> “Les Datacenters de Google sont le fruit d’investissements colossaux et de l’innovation permanente.”

Répondant aux normes les plus exigeantes en matière de sécurité, de robustesse et d’environnement, les Datacenter de GOOGLE sont le fruit d’investissements colossaux et de l’innovation permanente de GOOGLE.

C’est pourquoi SKALE-5 a retenu GOOGLE pour héberger les applications de ses clients et ainsi les faire bénéficier des meilleures infrastructures mondiales pour héberger leurs applications et délivrer leur business.
En hébergeant vos applications sur Google Cloud Platform vous bénéficierez instantanément de toute la puissance de l’infrastructure GOOGLE.

3. Carrier Interconnect

Afin de pouvoir bénéficier des bandes passantes, QoS et latences nécessaires aux applications d'un Système d'Information, GOOGLE propose une offre d'interconnexion réseau avec les principaux opérateurs réseau WAN et VPN mondiaux et Européens.

Ce service GOOGLE se nomme CARRIER INTERCONNECT et est directement disponible avec les opérateurs suivants (liste non-exhaustive) :

zayo · verizon · orange Business Services · Level(3) COMMUNICATIONS · EQUINIX · GLOBALCAPACITY · eunetworks · colt · 1cloudstar · telx the interconnection & data center company · interCloud · TATA COMMUNICATIONS · CyrusOne Built for Tomorrow. Ready Today.

4. Sécurité

Parce que la sécurité est fondamentale, l'architecture de sécurité de GOOGLE répond aux certifications les plus exigeantes du marché (ISO27001, SOC2, SSAE 16/ISAE 3402)

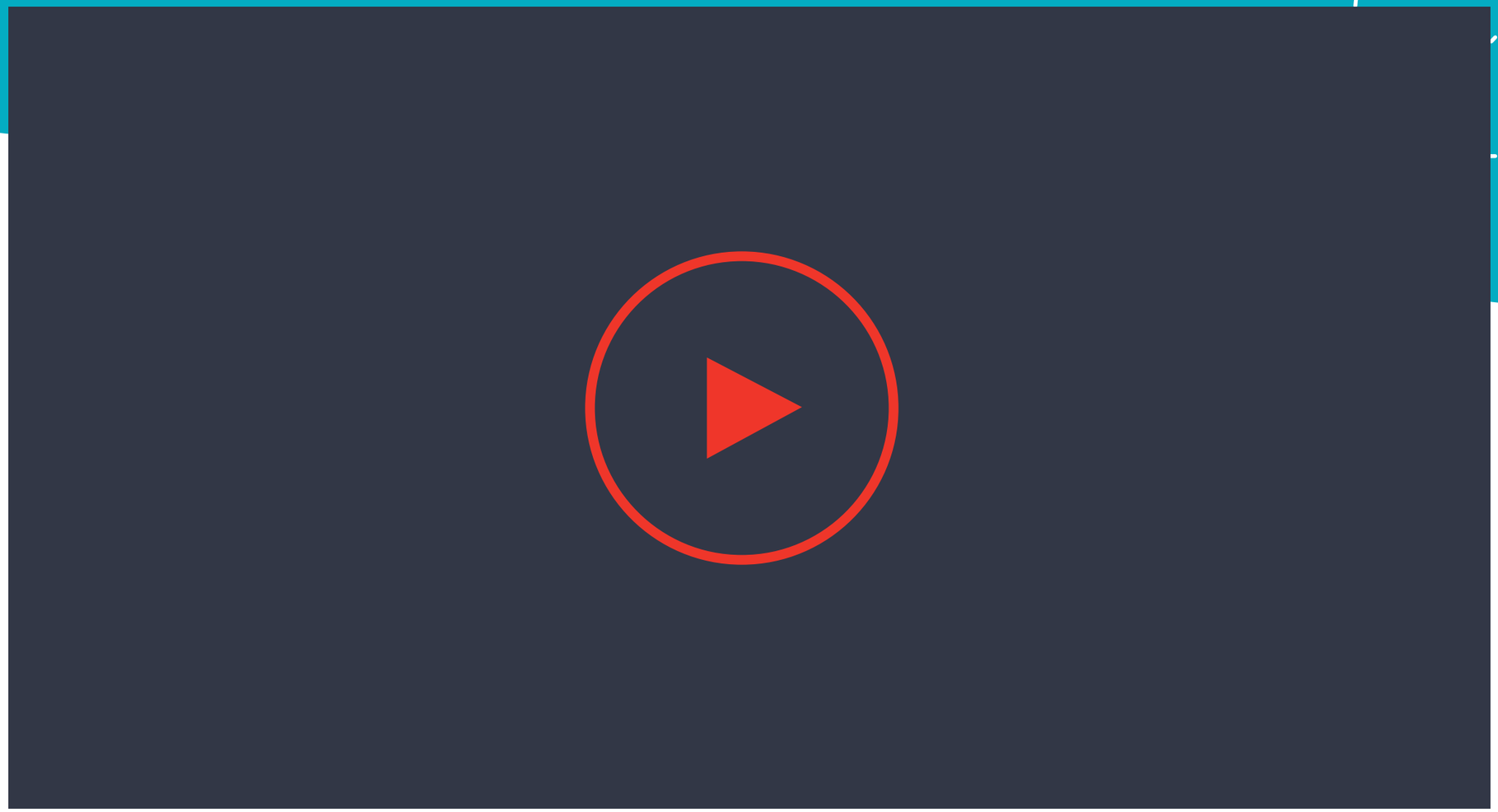
L'ensemble des applications et des données des clients exposées sur internet bénéficient d'une protection native contre les attaques en déni de service (DDOS).

Vos données sont hébergées dans les mêmes Datacenter que les services GOOGLE.COM, GMAIL, GOOGLE MAPS, ... La sécurité de vos données est donc assurée par une équipe de plus de 500 experts uniquement dédiés à la sécurité et d'un niveau d'expertise garantissant une réactivité exemplaire en cas d'alerte de sécurité sur des produits tiers ou des protocoles publiques (ex : faille de sécurité dans le protocole SSL en 2015 corrigé sur GOOGLE CLOUD 2 semaines avant la publication de l'avis de sécurité mondial par les CERT).

La sécurité chez GOOGLE, c'est plus de 500 experts uniquement dédiés à la sécurité des applications.

L'accès à vos instances s'établit par connexion sécurisée SSH (avec certificat) ou RDP pour les instances Windows sur la base de la clé publique fournit au lancement d'une instance et de la clé privée que vous conservez sur votre poste de travail.
De plus l'accès aux API est sécurisé avec vos identifiants, mots de passe et la génération d'un token unique.

5. Datacenter Google : suivez le guide



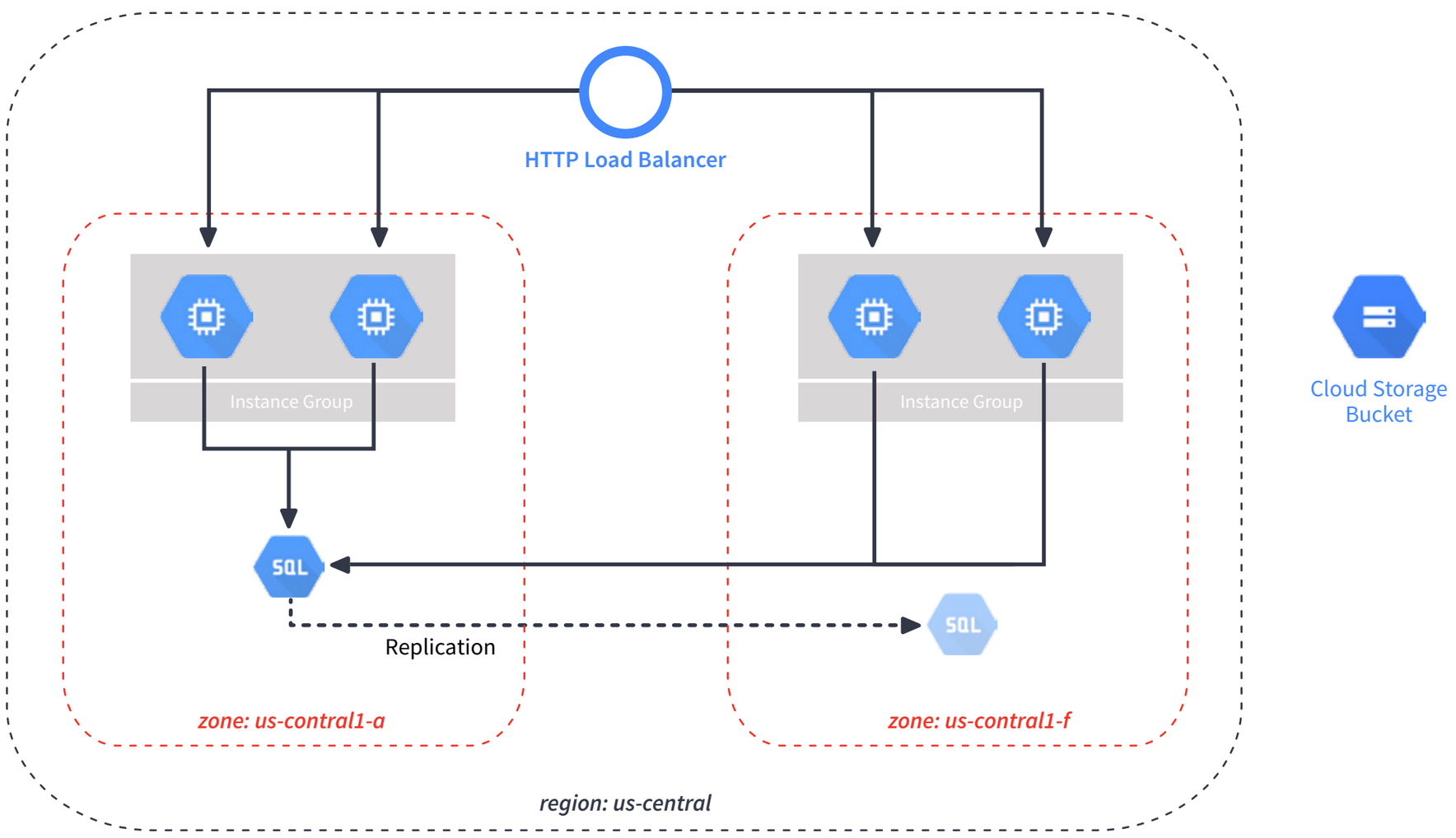
Pour visiter un DATACENTER GOOGLE, nous vous invitons à visionner cette petite vidéo*.

https://www.youtube.com/watch?v=XZmGGAbHqa0

CAS PRATIQUE

Exemple d’un déploiement Redmine

1. Notre architecture



Le schéma ci-dessus montre comment les différents composants Google Cloud Platform travaillent ensemble pour construire une application Web évolutive et résiliente. Le rôle de chaque élément est décrit plus en détail.

1. Notre architecture

EXIGENCES TECHNIQUES	SERVICES GOOGLE CLOUD PLATFORM
L'équilibrage de charge (LB)	https://cloud.google.com/compute/docs/load-balancing/http/
Machine virtuelle (compute)	https://cloud.google.com/compute/docs/zones
Scaling automatique	https://cloud.google.com/compute/docs/autoscaler
Stockage des données	Cloud SQL : https://cloud.google.com/sql/docs Cloud Storage : https://cloud.google.com/storage/

Le tableau suivant indique comment les différents services Google Cloud Platform adressent les exigences pour disposer d'applications scalables et résilientes.

2. Vue d'ensemble des composants

Chaque composant de l'architecture de cet exemple contribue à assurer la scalabilité et la résilience du service.

Équilibreur de charge HTTP

L'équilibreur de charge (Load Balancer) HTTP expose une seule adresse IP publique que les clients utilisent pour accéder à l'application. Cette adresse IP peut être associée à une entrée DNS (enregistrement de type A). Les demandes entrantes sont alors réparties entre les groupes d'instances dans chaque zone en fonction de la capacité de chaque groupe.

Dans une même zone, les demandes sont réparties uniformément sur les instances au sein du groupe. Bien que les loadbalancers HTTP puissent équilibrer le trafic dans plusieurs régions, nous n'utiliserons dans notre exemple qu'une seule région.

Zone

Une zone correspond à un DATACENTER localisé dans une région. Les différentes zones d'une région sont reliées entres-elles par un réseau haut débit à faible latence. SKALE-5 recommande le déploiement d'applications sur plusieurs zones dans une région.

Groupe instances

Un groupe d'instances est une collection d'instances homogènes qui peuvent être ciblées par un équilibreur de charge HTTP. Les instances sont ajoutées et retirées d'un groupe par un gestionnaire de groupe d'instances. Un groupe et un gestionnaire d'instances sont nécessaires dans chaque zone d'exécution.

Scaling automatique

L'autoscaler ajoute ou supprime des instances Google Compute Engine à un groupe d'instances par l'intermédiaire du gestionnaire de groupe. Il se base pour cela sur l'utilisation du processeur (CPU), ou d'autres signaux. Un autoscaler est requis pour chaque groupe d'instances que vous souhaitez voir scaler automatiquement (à la hausse et à la baisse).

Instance

Une instance est une machine virtuelle hébergée sur l'infrastructure de Google.

Stockage des données

Cloud SQL est une base de donnée MySQL hautement disponible et complètement prise en charge par Google : on parle alors d'un service managé de BDD. La réplication, le cryptage, les patches et les sauvegardes sont inclus dans la gestion automatisée. Une instance Cloud SQL est déployée dans une zone unique et les données sont répliquées automatiquement dans d'autres zones.

La solution Redmine, employée dans notre exemple, utilisant MySQL, est automatiquement compatible (et ce de manière transparente) avec Google Cloud SQL.

Cloud Storage est la solution de stockage dit « stockage objet » de Google. Cloud Storage permet aux fichiers d'être stockés, sécurisés et récupérés grâce a une interface simple et évolutive.

Dans notre exemple, Google Cloud Storage est utilisé pour distribuer les clés SSH aux instances Google Compute Engine, et stocker les fichiers téléchargés dans Redmine. Ceci nous permet ainsi de n'avoir aucune information stockée sur les disques « locaux » des instances.

3. Résilience

Pour que notre exemple d'architecture soit résilient, le système doit être en mesure de remplacer automatiquement les instances défaillantes ou indisponibles.

Quand une nouvelle instance est créée, elle doit être en mesure de :

- Connaître et comprendre son rôle dans le système
- Se configurer automatiquement
- Gérer ses dépendances
- Commencer la prise en charge des requêtes automatiquement

Afin de remplacer automatiquement une instance indisponible, vous pouvez utiliser un des composants de Google Compute Engine suivant :

- Les scripts de démarrage
- Les groupes d'instances
- Le gestionnaire de groupe d'instances
- Le modèle d'instance

Le script de démarrage s'exécute lorsque votre instance démarre ou redémarre. Vous pouvez utiliser ces scripts, pour installer des applications et des mises à jours, pour s'assurer qu'un service est bien démarré ou **pour installer un utilitaire de gestion de déploiement comme Chef, Puppet, Salt ou Ansible.**

Notre scénario utilise un script pour installer Chef Solo, qui déploiera les composants de notre application.

Vous trouverez des précisions sur comment utiliser ces scripts Chef et configurer automatiquement les instances à la fin de notre document.

En complément du script de démarrage, nous avons besoin de définir un ensemble de paramètres nécessaires au lancement d'une instance Google Compute Engine. Par exemple, nous devons spécifier le gabarit de serveur, l'image à utiliser, les disques que vous souhaitez attacher à l'instance. Pour ce faire, nous allons utiliser la notion de modèle d'instance que nous propose GCP.

3. Résilience

Le couple *modèle d'instance* + *script de démarrage* définit la manière de lancer une instance Google Compute Engine et de configurer les applications sur cette dernière. Ainsi l'instance se voit assigner un rôle spécifique dans votre architecture (ex : Serveur Web, Base de données).

Startup Script

```
#! /bin/bash
sudo apt-get install apache2=2.2.20-1ubuntu1
scp myuser@108.59.87.185:version_1_2_3/index.php /var/www/
```

Instance Template

```
$ gcloud compute instance-templates create my-template \
         --machine-type n1-standard-4 \
         --image debian-7 \
         --metadata-from-file startup-script="startup.sh"
```

GCE Instance

3. Résilience

Un **gestionnaire de groupe d'instances** permet de sélectionner un modèle d'instance à appliquer sur une nouvelle instance Google Compute Engine. On peut ainsi déterminer le nombre d'instances que l'on souhaite créer. Comme son nom l'indique, le gestionnaire de groupe d'instances, intègre une instance dans un groupe. Ce groupe nous permet de gérer des collections d'instances et ainsi d'éviter une gestion individuelle de chaque instance.

Le diagramme ci-dessous décrit comment ces composants interagissent ensemble :

Startup Script

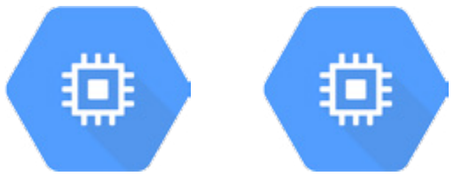
```
#! /bin/bash
sudo apt-get install apache2=2.2.20-1ubuntu1
scp myuser@108.59.87.185:version_1_2_3/index.php /var/www/
```

Instance Template

```
$ gcloud compute instance-templates create my-template \
          --machine-type nl-standard-4 \
          --image debian-7 \
          --metadata-from-file startup-script="startup.sh"
```

zone: us-central1-a

Instance Group



Instance Group Manager

```
$ gcloud preview managed-instance-groups \
          --zone us-central1-a create my-group \
          --base my-group \
          --size 2 \
          --template my-template
```

Notez que le gestionnaire de groupe d'instances et le groupe d'instance sont spécifique a une zone donnée, il réside dans une zone unique. Pour déployer une application sur plusieurs zones, il est alors nécéssaire de créer ces ressources dans chaque zone ciblée.

Avec les scripts de démarrage, les modèles d'instances, le gestionnaire de groupe d'instances et les groupes d'instances, vous disposez dorénavant d'un système qui peut remplacer les instances non fonctionnelles par des nouvelles.

4. Détection de défaillance

Dans cette section, nous allons présenter comment détecter la défaillance d'une instance.

A ce niveau, notre exemple d'application a presque tous les outils nécessaires pour que l'architecture soit résiliente. Il ne lui manque plus qu'à détecter les instances défaillantes **afin qu'il puisse les remplacer automatiquement.**

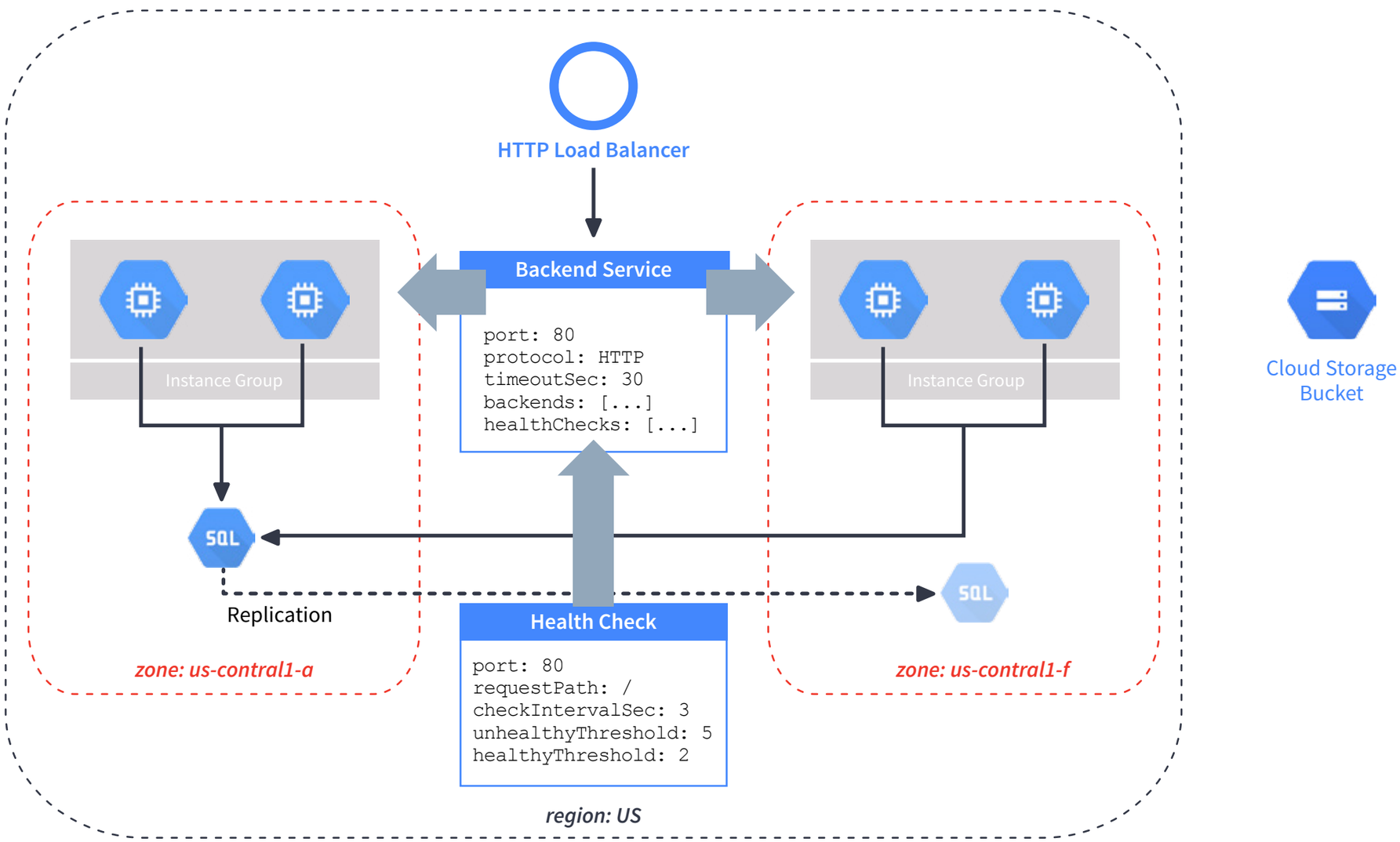
Notre application utilise un Load Balancer afin de distribuer le trafic entrant.

Cette architecture nous permet donc d'utiliser les 2 services suivants pour identifier les instances fonctionnelles :

- **Health checks :**
  C'est un détecteur de défaillance HTTP. Il spécifie le port et l'adresse à interroger pour chaque instance. Le détecteur de défaillance attend une réponse 200 OK depuis une instance fonctionnelle.

- **Backend services :**
  Ce dernier définit un ou plusieurs groupes d'instances qui peuvent recevoir du traffic depuis l'équilibreur de charge. Il spécifie le port et le protocole exposé dans les instances (exp : HTTP port 80).

4. Détection de défaillance



Le diagramme ci-dessus décrit l'architecture de l'application et comment le « backend services » et le « HTTP Health check » sont liés à l'équilibreur de charge et au groupe d'instances.

5. Résilience des données avec Google Cloud SQL

Les trois grands composants d'une architecture applicative sont : le réseau, le traitement et le stockage. A ce stade, nous avons couvert le réseau et le traitement. Nous allons maintenant nous attacher à rendre résilient notre stockage.
Dans notre exemple, nous utilisons Google Cloud SQL. Ce service nous permet de disposer d'une base de données MySQL 5.5 hautement disponible et entièrement prise en charge par Google. Avec Cloud SQL, la gestion de la réplication, du cryptage, des patchs et des sauvegardes est entièrement assurée de manière transparente par les services managés de Google Cloud Platform.

Une base de données Google Cloud SQL est localisée sur une région étendue, ce qui signifie que les données sont automatiquement répliquées dans plusieurs Datacenter Google d'une même région (ex : plusieurs Datacenter dans la zone Europe). Ainsi chaque transaction sera exécutée sur l'ensemble des réplicats de votre base de données. Cette réplication, entièrement transparente pour votre application, rend votre service tolérant à la défaillance d'un, voire de plusieurs serveurs de base de données.

Cloud SQL vous permet de choisir entre deux types de réplication :

- **Réplication synchrone**

Dans une réplication synchrone, les transactions (ex : écritures) sont exécutées sur l'ensemble des serveurs avant de « rendre la main » au client (votre application). Une réplication synchrone permet de garantir une parfaite consistance des données en cas d'incident. En revanche, cette stratégie de réplication induit un temps d'écriture plus long et peut avoir des conséquences importantes sur les performances applicatives.

- **Réplication asynchrone**

Comme son nom l'indique, une réplication asynchrone permet d'asynchroniser les mises-à-jour sur les réplicas. C'est-à-dire que le système rend la main rapidement à l'application sans attendre que la transaction ait été exécutées sur l'ensemble des réplicas. Ainsi la réplication asynchrone assure une écriture plus rapide dans la base de données puisque vous n'avez pas à attendre la fin de réplication. La contrepartie de cette vitesse de réplication est le risque de perdre la dernière mise-à-jour (dernier WRITE ou UPDATE) lors d'un incident majeur. Néanmoins, les SGBD savent contrôler et assurer la consistance de leurs données et ce, même dans des systèmes de réplications asynchrones.

Dans notre exemple, l'application Redmine utilise une réplication synchrone. Ce choix a été justifié dans notre cas par la faible charge en écriture portant sur notre service (qui délivre du contenu, plus qu'il n'en reçoit).

Vous devez choisir entre une réplication synchrone ou asynchrone en fonction des performances d'écriture de votre système et les spécifications de durabilités des données.

6. Scalabilité

Vous disposez d'une application résiliente. Voyons maintenant comment rendre votre service scalable. Nous avons défini la scalabilité plus haut dans notre document, c'est pourquoi je vous propose de la résumer par la propriété suivante :
Notre application doit pouvoir fonctionner aussi bien avec un seul utilisateur qu'avec 1 000 000 utilisateurs et les ressources devront augmenter ou diminuer en fonction de ces utilisateurs.

Pour que les ressources de l'application puissent augmenter ou diminuer, il est nécessaire que notre système dispose :

- **D'un moyen par lequel il peut ajouter ou supprimer des instances depuis un service.**

Il est aussi nécessaire de décider quand une instance doit être ajoutée et supprimée : c'est tout le rôle du composant Autoscaler de Google Cloud Platform.

- **D'un espace de stockage de fichiers partagés.**

En effet, comme nos instances sont considérées comme éphémères (c.-à-d. qu'elles peuvent naitre et disparaître en fonction de la charge), il devient alors nécessaire de ne pas stocker les fichiers sur leurs disques locaux. L'architecture de notre application résout ce problème pour les données relationnelles en les sauvegardant dans une instance Cloud SQL. Nos fichiers sont quant à eux stockés sur Google Cloud Storage pour répondre à ce besoin.

6. Scalabilité

Scaler votre service avec l'Autoscaler

Google Compute Engine Autoscaler nous permet d'ajuster dynamiquement les ressources nécessaires à une application.

Quand le trafic ou la charge augmente, l'autoscaler ajoute des ressources afin d'absorber l'excèdent de charge et supprime les ressources quand le trafic ou la charge baisse afin d'aider à diminuer les coûts d'infrastructure. Autoscaler éxécute ces actions automatiquement en se basant sur des règles de scaling que vous pouvez définir vous même.

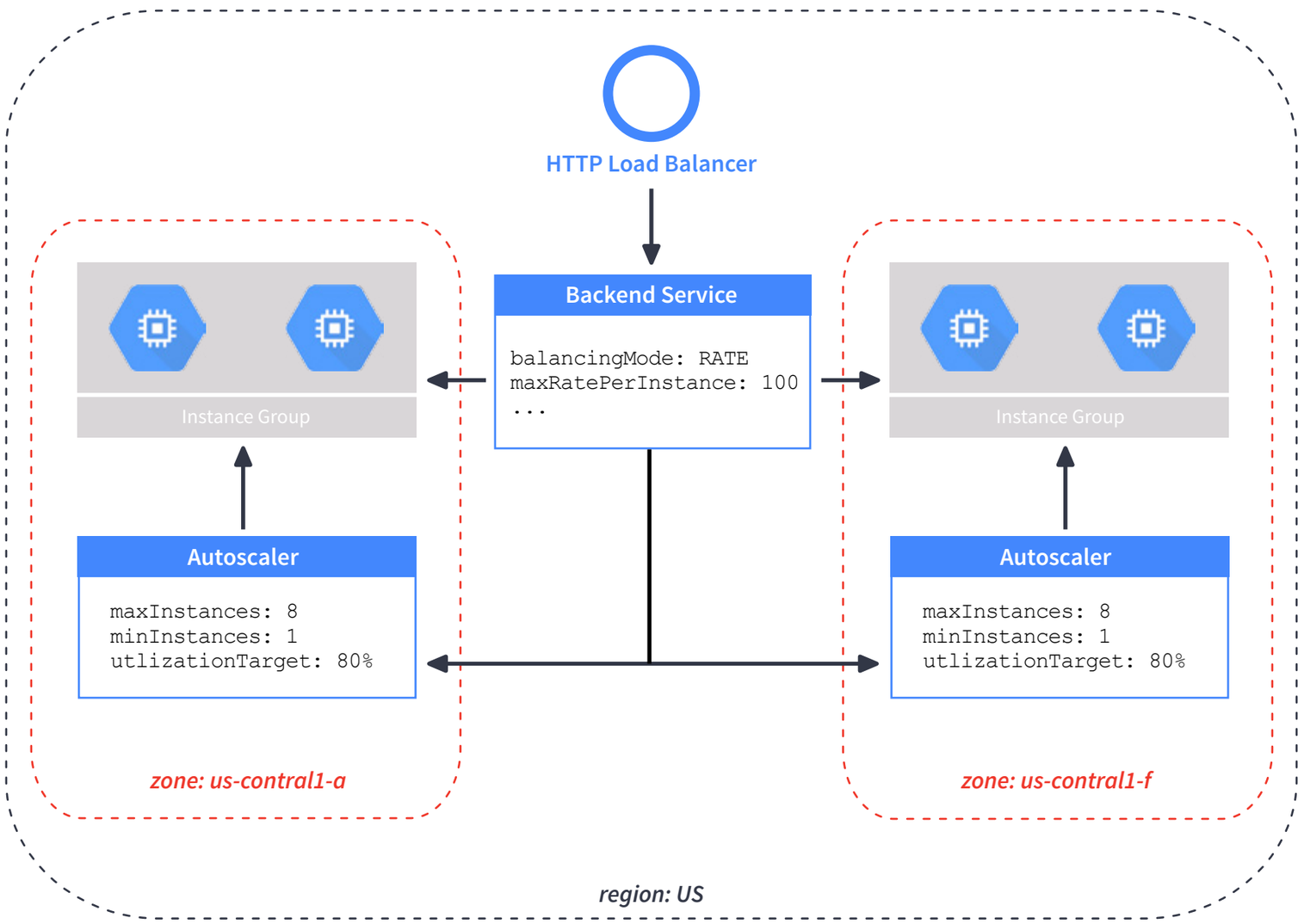
L'intérêt d'autoscaler est double :

1. **Vos utilisateurs disposent de performances toujours optimales, et ce, quelque soit le niveau de charge portant sur votre application.**

2. **Vous disposez dorénavant d'un formidable levier d'optimisation de vos coûts d'infrastructure.**

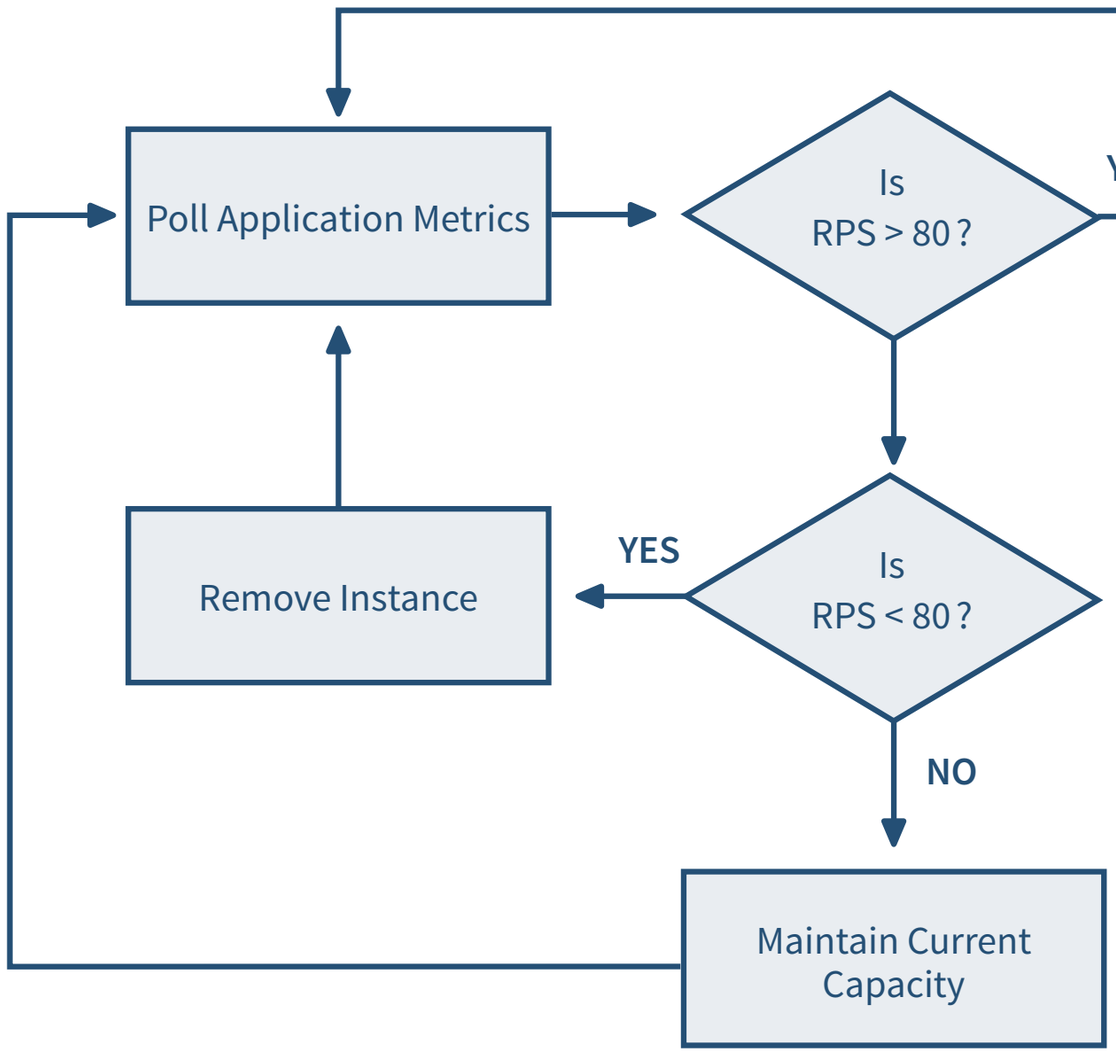
Autoscaler peut faire varier (« scaler ») le nombre de machine virtuelle en se basant sur l'utilisation du CPU/ Mémoires /... , la serving capacity ou sur les Cloud Monitoring Metric (c.-à-d. des métriques nombreux et personnalisables). Notre exemple REDIS utilise les serving capacity metric afin d'ajouter ou supprimer des instances Google Compute. Notre métrique se base sur le nombre de requêtes par seconde (RPS) que chaque instance peut recevoir depuis l'équilibreur de charge.

6. Scalabilité



Dans notre exemple nous avons défini un balancing Mode = RATE. Cette propriété informe notre LoadBalancer d'équilibrer en fonction du RPS défini dans la propriété maxRatePerInstance (fixé à 100 pour notre exemple).

6. Scalabilité



La section « Cas pratique » est sous licence Creative Commons 3.0 et librement adaptée de l'article « Building scalable resilient Web application with GCP ».

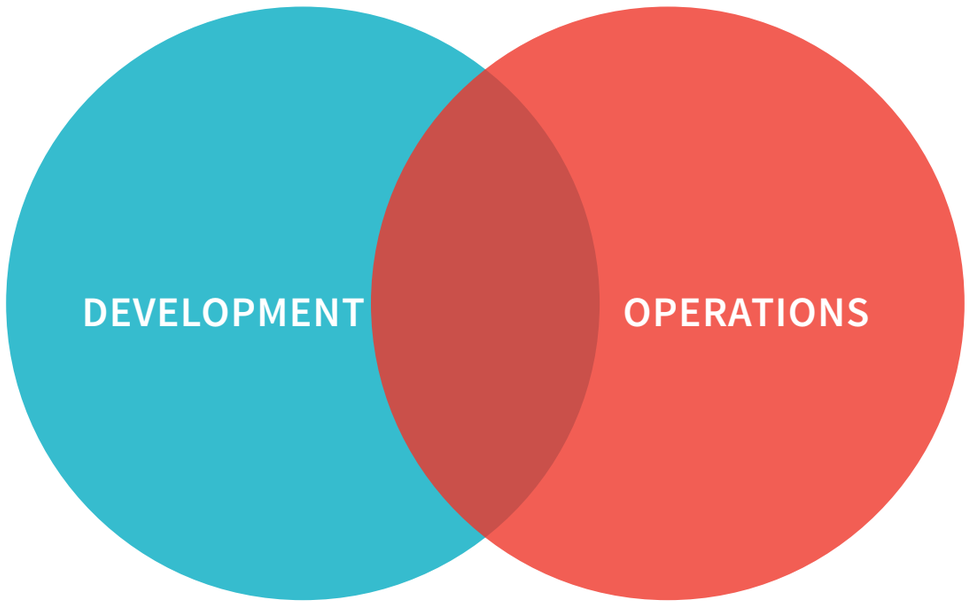
Chaque Autoscaler définit la propriété « utilizationTarget » qui définit en pourcentage le seuil de charge maximum.

Dans notre exemple, chaque Autoscaler fixe le paramètre « utilization-Target » à 80%. Ainsi notre autoscaler créera une nouvelle instance / VM lorsque la charge dépassera 80 RPS (soit 80 % de 100 RPS). A contrario, l'autoscaler détruira des instances lorsque ce RPS sera inférieur à ce seuil.

Fort heureusement chaque autoscaler définit un nombre maximal et un nombre minimal d'instances qu'il ne pourra pas dépasser.

CLOUD & DEVOPS

1. Qu'appelle-t-on DevOps ?



« DevOps » est un terme issu de la contraction des mots anglais « development » (développement) et « operations » (exploitation).

La construction par juxtaposition de ces 2 termes met l'accent sur le rapprochement entre les équipes, pratiques et outils dits de BUILD (dev) et de RUN (Ops). Ce rapprochement prôné par le mouvement DevOps permet de fluidifier les mises en services des fonctionnalités et des applications .

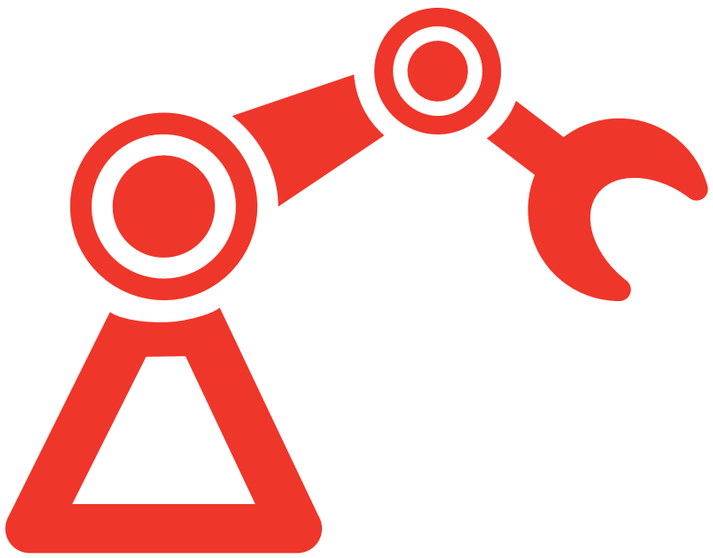
Bien que l'outillage facilite l'automatisation, le mouvement **DevOps constitue plus une culture et un référentiel de pratiques qu'une suite de solutions logicielles.**
La culture DevOps se caractérise principalement par :

- Un déploiement régulier des applications, la seule répétition contribuant à fiabiliser le processus
- La standardisation et l'automatisation de tout déploiement ou paramétrage
- Une intégration continue incluant des « tests continus »
- Une boucle d'amélioration courte (i.e. un feed-back rapide des utilisateurs)
- Une surveillance étroite de l'exploitation et de la qualité de production factualisée par **des métriques et indicateurs « clé ».**

La place de l'automatisation

Les équipes DevOps ont à cœur de standardiser (ex : utilisation de conventions) et d'automatiser chacune des actions à réaliser sur les environnements (y compris la création des environnements eux-mêmes). On parle alors d'automatisation.

Il est évident que l'automatisation des déploiements est antérieure au mouvement DevOps, néanmoins l'automatisation apportée par ce mouvement repose sur les paradigmes du développement agile (ex : intégration continue, pilotage par les tests, conventions plutôt que configuration, utilisation de gestionnaires de configuration pour tout, …).



La convergence des pratiques du développement et du RUN a fait émerger ces dernières années les avantages suivants :

- Etablissement d'une culture et d'un langage partagé : celle du test, de l'itératif, de l'intégration continue, de la gestion de la dette technique et des API
- Accélération et fluidification des mises en service (et donc diminution des coûts de RUN)
- Apparition de nouveaux outils et framework de RUN (CHEF, PUPPET, ANSIBLE, TERRAFORM, …)
- Pilotage des infrastructures par API des Cloud Providers ?

Du point de vue de l'outillage, DevOps permet d'industrialiser, standardiser et simplifier le développement (le terme ici n'est pas neutre) des scripts de déploiement. Que les scripts soient développés avec un système avec agent (CHEF, PUPPET) ou sans agent (ANSIBLE), leur mise au point et leur degré de réutilisation est très largement facilité par ces outils. Ces solutions (Opensource pour la plupart) peuvent être comparées aux frameworks légers dans le monde du développement.

2. Cloud et DevOps : Datacenter as code

Nous avons vu que les pratiques DevOps privilégient l'automatisation et le développement des actions à réaliser sur les infrastructures. C'est donc tout naturellement que les API Cloud constituent un levier extraordinaire pour piloter/programmer son infrastructure.

Soyons bien clair, nul besoin d'être hébergé chez un Cloud Provider pour pouvoir utiliser les outils et les pratiques DevOps. En effet, chacun peut développer des scripts ANSIBLE pour automatiser ses déploiements systèmes et applicatifs quelque soit son infrastructure. En revanche, la création/tests/modifications de l'infrastructure en elle-même est grandement facilitée par l'utilisation des API.
Prenons l'exemple d'un environnement simple :

- 3 x Loadbalancer
- 6 x Serveurs WEB
- 2 x serveurs de Cache
- 2 x BDD en read-replica
- 5 x entrées DNS
- 40 x règles de Firewall

Une équipe DevOps dont le système d'information est hébergé sur un IasS tel que GOOGLE Cloud Platform, utilisera toute la puissance et la facilité des API de la plateforme pour entièrement (100%) développer et donc automatiser/tester le déploiement des éléments d'infrastructure.

De plus ces appels API seront réalisés à partir des outils DevOps (ex : CHEF) du marché dont la richesse de l'écosystème fournit une importante bibliothèque de « plugins » pour chacun des cloud providers leaders.

API + DEVOPS
= SUCCESS

On dit souvent que « l'informatique se voit partout sauf dans les chiffres de la productivité » (Paradoxe de Solow, Prix Nobel d'économie en 1987).

Néanmoins, si il y a bien un couple qui permet de gagner en productivité et fiabilité, il s'agit bien de celui-ci :

1. Equipe DevOps

2. Cloud exposant ses services sous forme d'API (reconnues et/ou standardisées)

Si par ailleurs vous disposez d'une équipe de BUILD Agile efficace en capacité d'alimenter vos équipes de RUN, alors votre DSI constituera la rampe de lancement tant attendue par les équipes métiers (Rapidité, Time to Market, Pivot, compétitivité sur les coûts informatiques, capacité d'innovation).

INTÉGRATION, INDUSTRIALISATION
ET INFOGÉRENCE CLOUD

Nos experts architecturent, déploient et opèrent vos applications
24x7x365 sur les meilleurs cloud du marché.

Restez concentrés sur le BUILD : Nous gérons le RUN de vos applications

Migrer

La transition vers le Cloud nécessite une expertise afin d'en maximiser les bénéfices. **Concentrez vous sur votre business** : nous migrons pour vous vos applications dans les règles de l'art (Automatisation, Sécurité, Exploitation et performances).

Concevoir

Les services IaaS évoluent en permanence permettant ainsi d'aller toujours plus vite, d'être plus évolutif et moins cher. Notre équipe d'architectes Cloud **conçoit pour vous les meilleures architectures Cloud pour votre business.**

Sécuriser

Design, Gestion des clés SSH, Authentification à double facteurs, serveur de bastion : sécuriser votre infrastructure même sur les meilleurs Cloud peut s'avérer complexe. Nos outils, **modèles et expertises vous permettront de faire bien, vite et donc sécurisé.**

Opérer

Intervenir à deux heures du matin... Appliquer un patch urgent... Restaurer une base de données : des taches parfois fastidieuses mais nécessaires au succès de votre business. Nous associons outillage, industrialisation DevOps et **mobilisation de nos équipes pour veiller 24x7x365 sur vos applications et données.**

INTÉGRATION, INDUSTRIALISATION
ET INFOGÉRENCE CLOUD

www.skale-5.com

info@skale-5.com

+33 1 58 36 40 60

33 Avenue des Champs Elysées
75008 PARIS

Twitter : twitter.com/skalefive

Github : github.com/orgs/skale-5

Linkedin : www.linkedin.com/company/skale-5

Décembre 2015