

Cloud & Architectures

Partie 2
Arnaud RAULET

Partie 2 : Cloud

- Définitions
- Infrastructure as a service
- Datacenter
- Docker / conteneurs
- DevOps
- Infra as Code (IaC)
- Cloud natives applications
- Serverless
- Référentiels, bonnes pratiques et patterns
- Que retenir ?
- Démos
-

Cloud : définitions

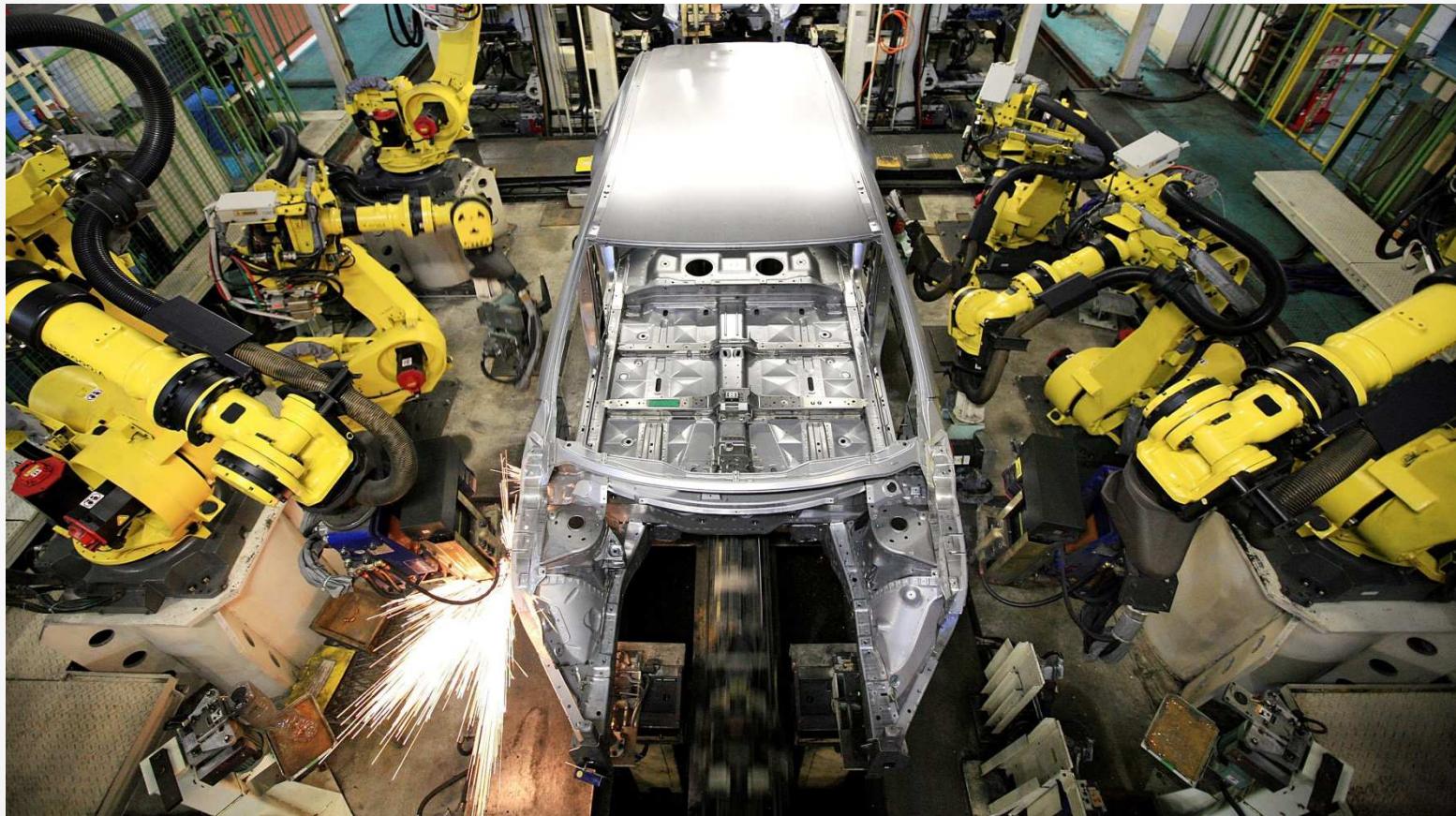
• • •

Classical IT



© PA

IT in the cloud



Magic ?

I WAS HOPING FOR
A SLIGHTLY MORE DETAILED
EXPLANATION OF HOW
CLOUD COMPUTING WORKS
THAN - "IT'S MAGIC"!



© D.Fletcher for CloudTweaks.com

Stupid ?

Richard Stallman, Founder of GNU

*“It's stupidity. It's worse than
stupidity: it's a marketing hype
campaign”*

*- The Guardian Newspaper,
2008*

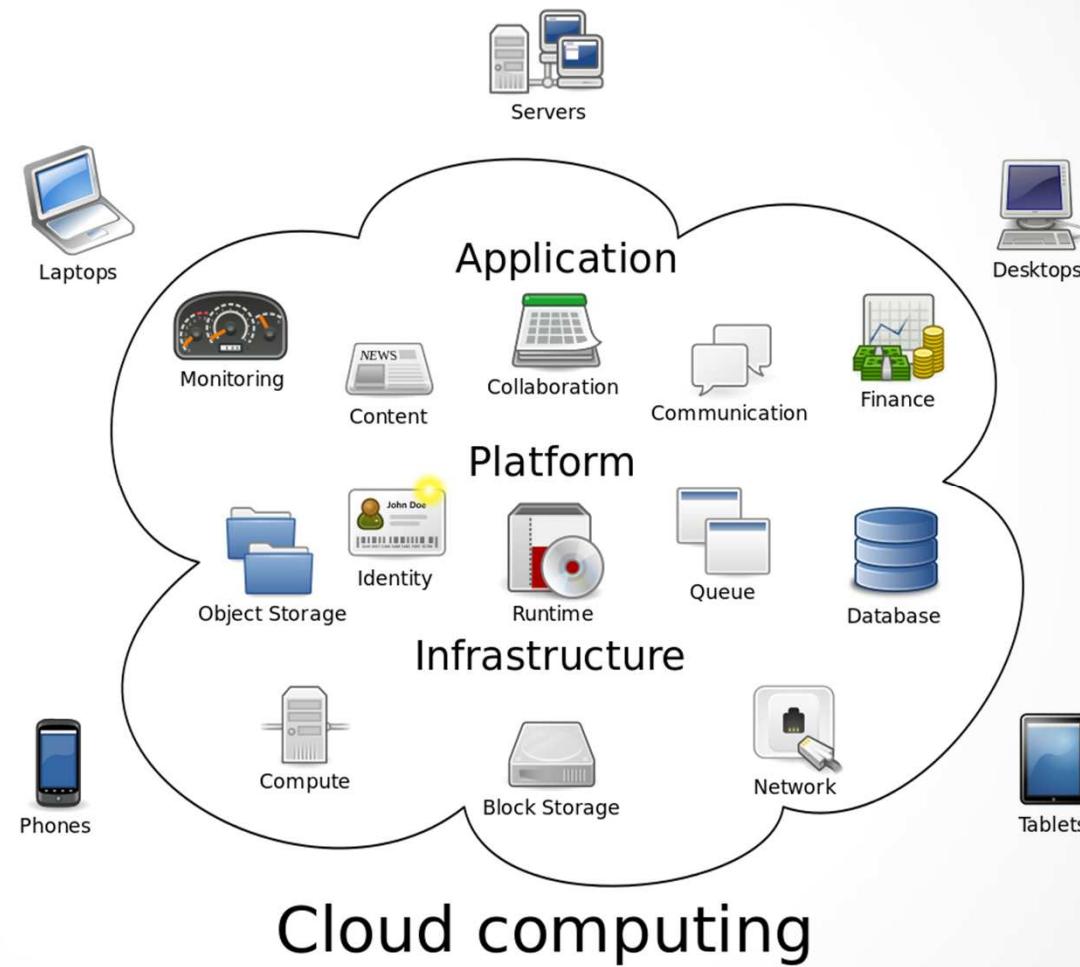


Siri says



“Services that provide common business applications online, which are accessed from a Web browser, while the software and data are stored on the servers; a style of computing in which dynamically scalable and often virtualized resources are provided as a service over the internet”

Wikipedia



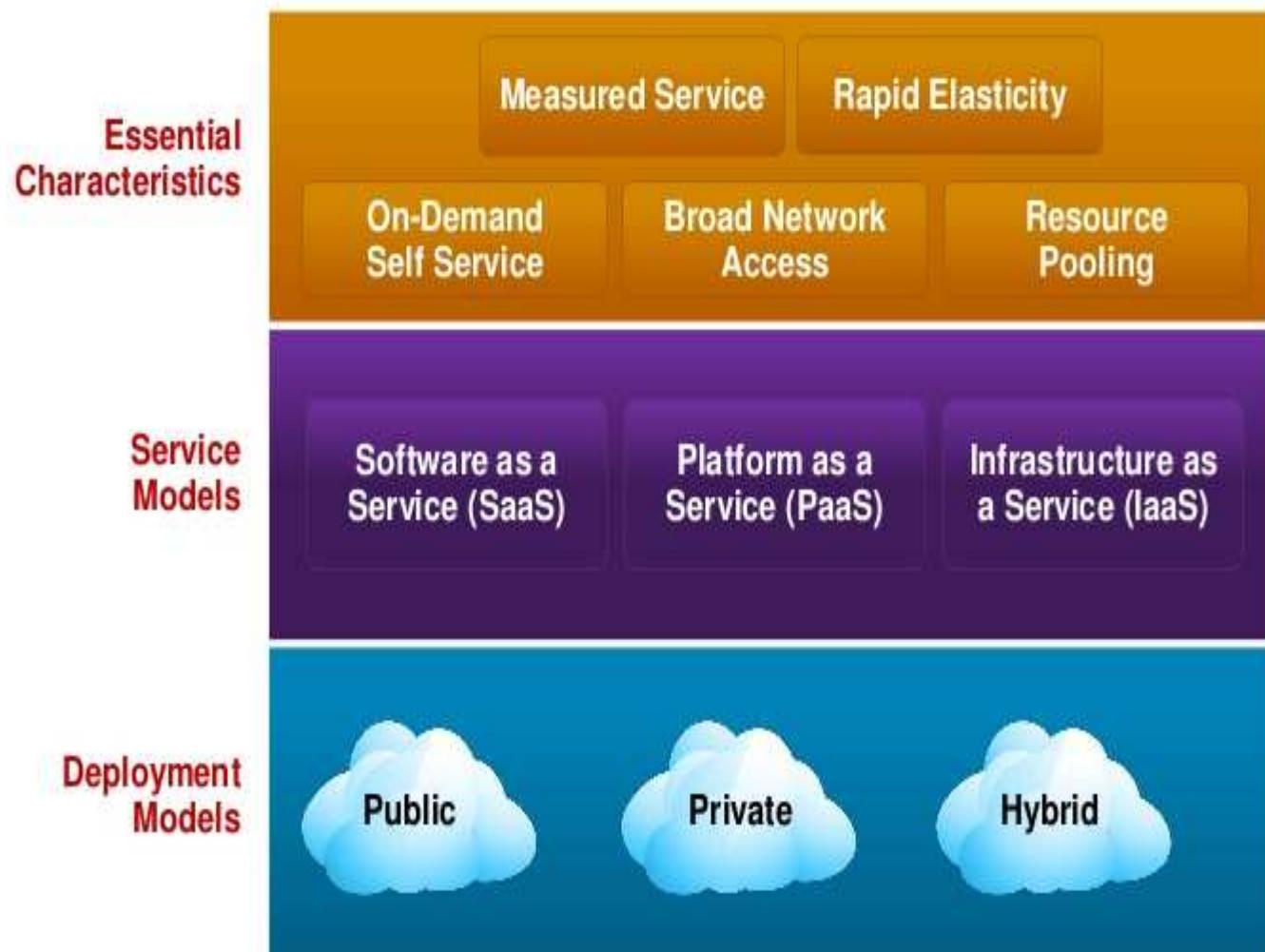
NIST

National Institute of Standards and Technology

- "Le cloud computing est un modèle qui permet un accès omniprésent, pratique et à la demande à un réseau partagé et à un ensemble de ressources informatiques configurables (comme par exemple : des réseaux, des serveurs, du stockage, des applications et des services) qui peuvent être provisionnées et libérées avec un minimum d'administration.
- Ce modèle est composé de **5 caractéristiques essentielles**, de **3 modèles de services** et de **4 modèles de déploiement**."

Cloud Computing – A Recap

NIST Summary



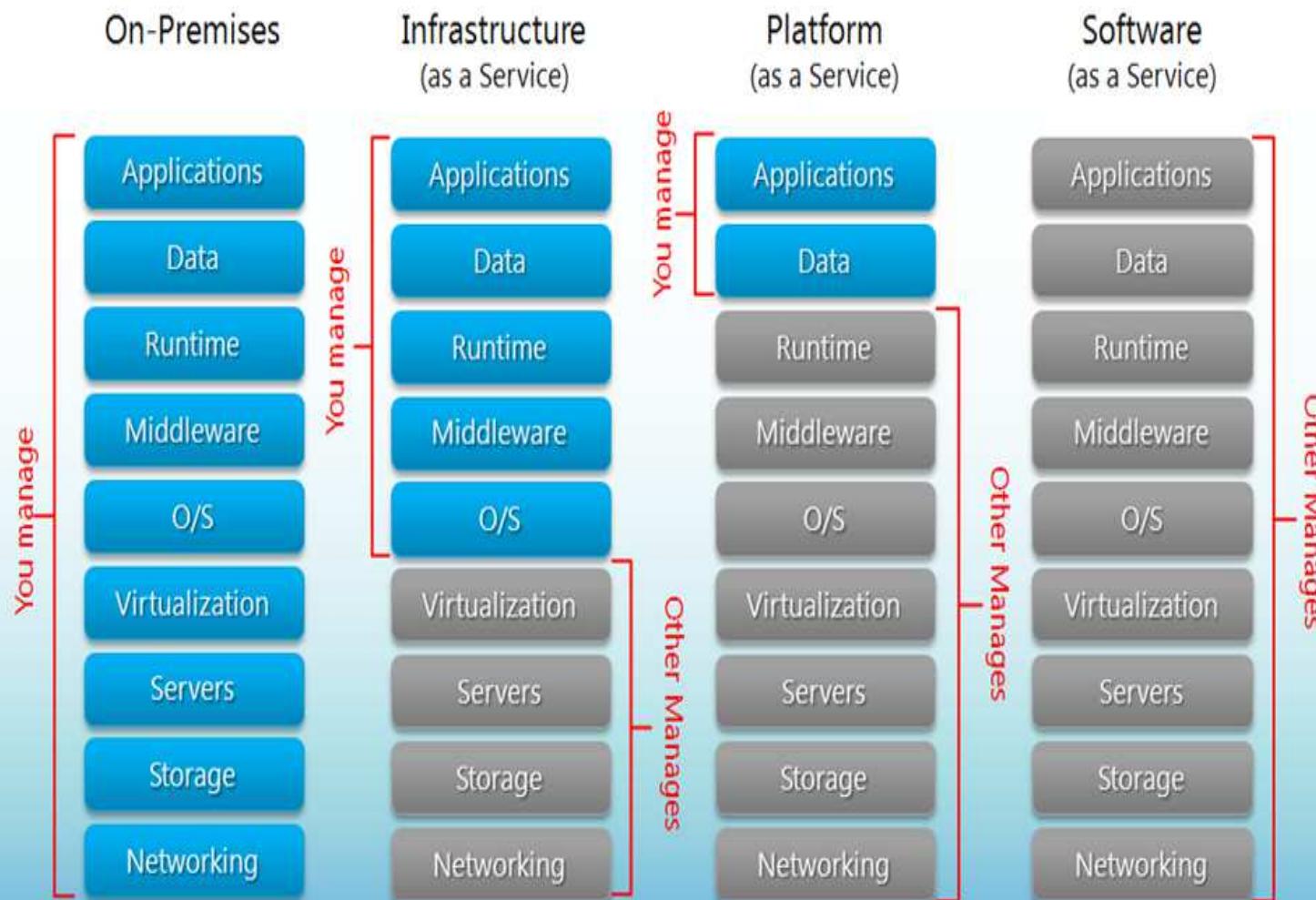
NIST - Les 4 modèles de déploiement

- **Le Cloud privé** est le plus utilisé par les entreprises. Celui-ci peut être hébergé en interne par l'entreprise ou par un tier. Son fonctionnement est dédié à l'entreprise et peut-être accédé par d'autres filiales via un accès sécurisé (VPN).
- **Le Cloud public** est accessible par Internet et géré par un prestataire externe. Il est ouverte au public ou à de grands groupes industriels. Cette infrastructure est possédée par une organisation qui vend des services Cloud, tel que Amazon ou Microsoft, dont nous parlerons un peu plus tard. Pour réutiliser notre exemple de tout à l'heure, une boîte mail accessible via internet est une application hébergée sur un Cloud public.
- **Le Cloud hybride** est, comme son nom l'indique, un mix entre cloud privé et cloud public. Les infrastructures utilisent la même technologies pour permettre la portabilité des applications et des données.
- **Le Cloud communautaire** est un cloud utilisé par plusieurs organisations ayant des besoins communs. Par exemple, une entreprise et son sous-traitant ont besoin d'utiliser en commun une application métier nécessaire au bon déroulement de leur affaire. Les deux entités décident alors de créer en collaboration un Cloud hébergeant cette application.

NIST - Les 3 modèles de services proposés

- **Le SaaS**, pour Software as a Service. Il s'agit de mettre en place un Cloud permettant le partage d'une application métier. Celle-ci est hébergée sur un serveur et accessible par l'utilisateur en réseau via différents terminaux. Ex : [Google Docs/gmail](#), [SalesForce](#).
- **Le PaaS**, pour Platform as a Service. L'objectif du PaaS est de fournir l'environnement informatique adéquat pour que le client puisse y installer et déployer des applications métiers ou autres services. Ainsi, l'objectif est de disposer d'une machine déjà configurée (matériel défini à l'avance, OS et logiciels nécessaire déjà installés) et de travailler dessus en y implantant ses propres logiciels. Ex : [Cloud Foundry](#), [openShift](#), [Azure](#)
- **Le IaaS**, pour Infrastructure as a Service. L'objectif est de louer l'ensemble du matériel à un tier (Serveur, réseau, et autres ressources). L'utilisateur a la possibilité de déployer n'importe quel type de logiciel incluant les systèmes d'exploitation. L'utilisateur ne gère pas ou ne contrôle pas l'infrastructure Cloud sous-jacente mais a le contrôle sur les systèmes d'exploitation, le stockage et les applications. L'objectif de mettre en place le IaaS pour une entreprise est de pouvoir augmenter ou diminuer comme bon lui semble ses ressources informatiques en fonction de ses besoins. Ex : [AWS](#), [Azure](#), [openstack](#), [Rackspace](#)

Separation of Responsibilities



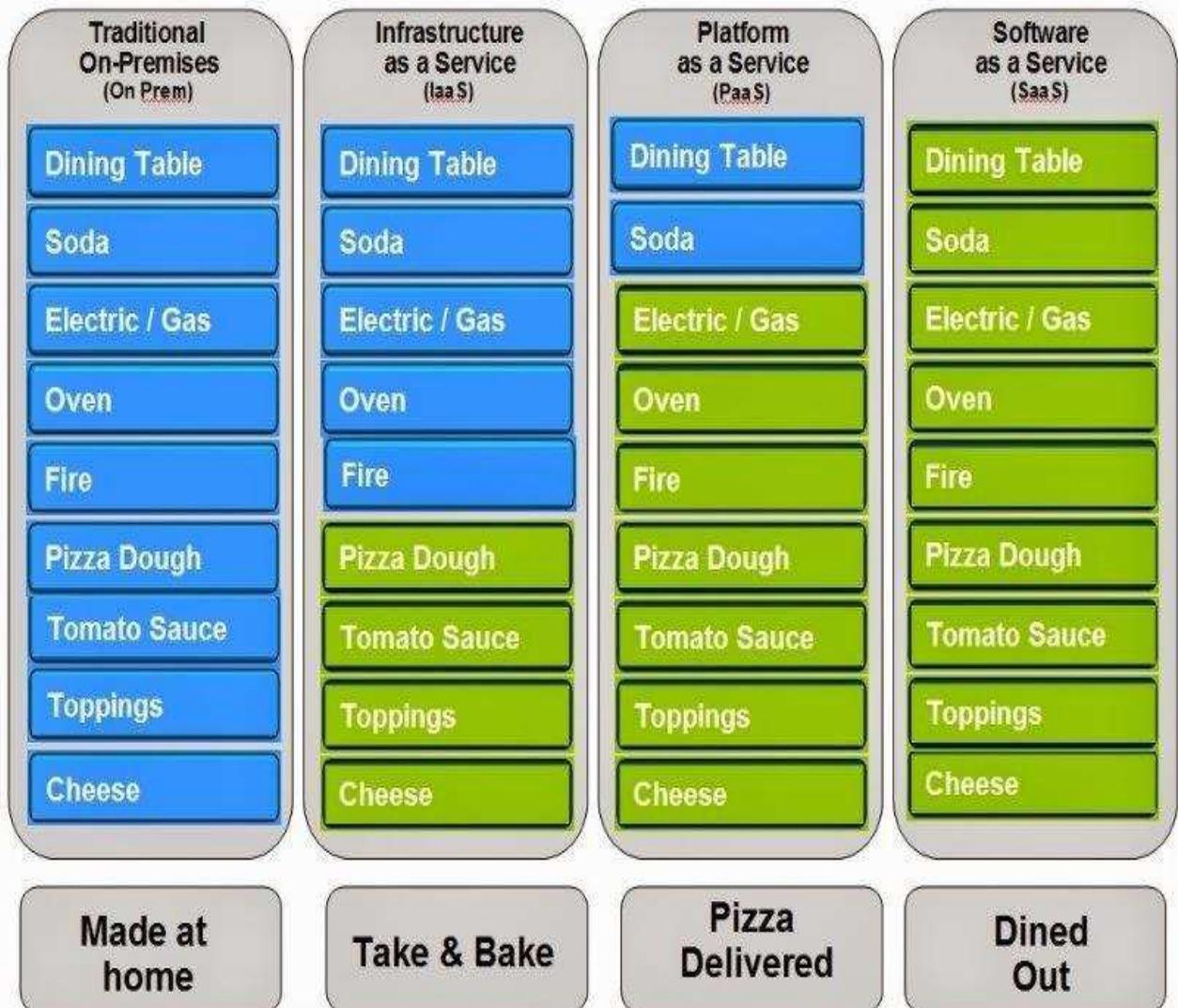
A qui s'adresse chacun des modèles ?



- **SaaS** : Utilisateurs finaux
- **PaaS** : Développeurs (Dev)
- **IaaS** : Administrateurs (Ops)

Everything As A Service

Pizza as a Service



Magic Quadrant Gartner 2016

Cartographie des différents acteurs du marché, en fonction de leur maturité et leur capacité à innover.



Infrastructure as a Service

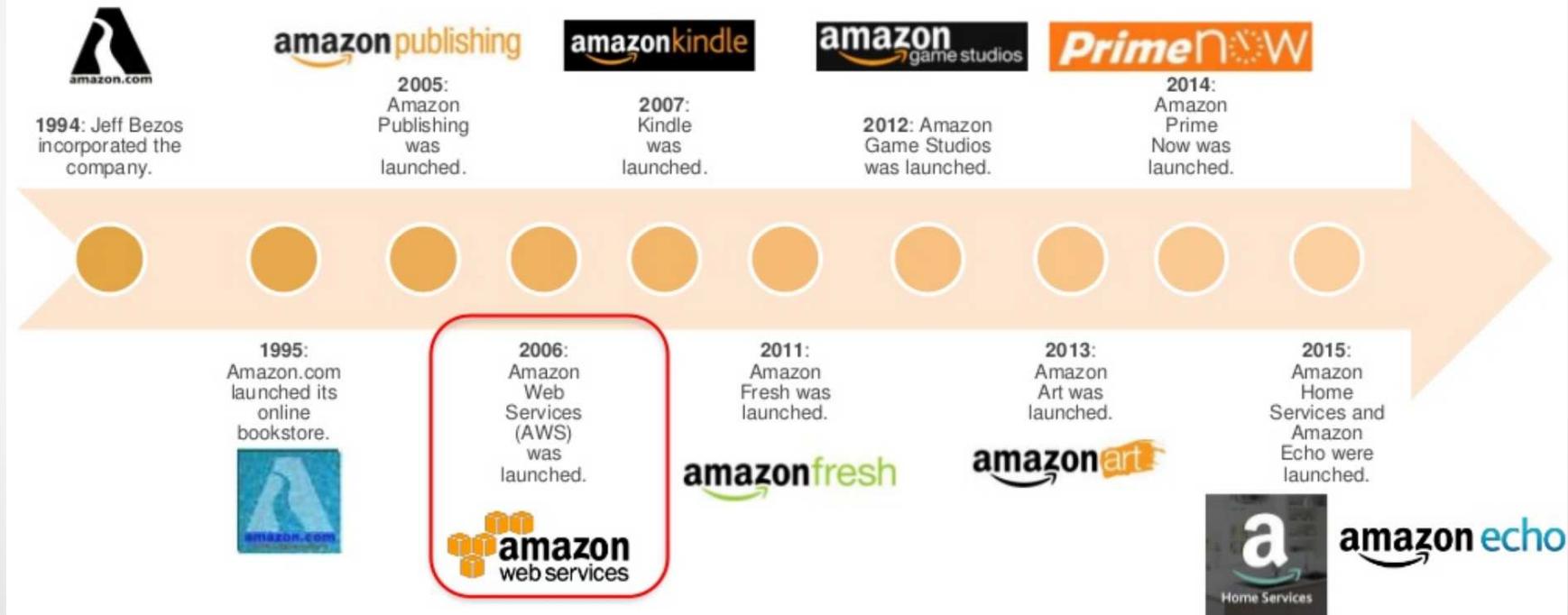
• • •

Exemple d'AWS

Amazon Web Services

Amazon History

Clip slide



AWS : catalogue de services

Amazon Web Services

Compute



Virtual Servers in the Cloud



Run and Manage Docker Containers



Run and Manage Web Apps



Run Code in Response to Events

Storage & Content Delivery



Scalable Storage in the Cloud



Global Content Delivery Network



PREVIEW

Fully Managed File System for EC2



Archive Storage in the Cloud



Large Scale Data Transport



Integrates On-Premises IT Environments with Cloud Storage

Database



Managed Relational Database Service



Predictable and Scalable NoSQL Data Store



In-Memory Cache



Managed Petabyte-Scale Data Warehouse Service

Networking



Isolated Cloud Resources



Dedicated Network Connection to AWS



Scalable DNS and Domain Name Registration

Developer Tools



Store Code in Private Git Repositories



Automate Code Deployments



Release Software using Continuous Delivery

Management Tools



Monitor Resources and Applications



Create and Manage Resources with Templates



Track User Activity and API Usage



Track Resource Inventory and Changes



Automate Operations with Chef



Create and Use Standardized Products



Optimize Performance and Security

Security & Identity



Manage User Access and Encryption Keys



Host and Manage Active Directory



PREVIEW



Analyze Application Security



Filter Malicious Web Traffic

Analytics



Managed Hadoop Framework



Orchestration for Data-Driven Workflows



Run and Scale Elasticsearch Clusters



Work with Real-time Streaming data

Internet of Things



BETA

Connect Devices to the cloud

Mobile Services



BETA

Build, Test, and Monitor Mobile apps



User Identity and App Data Synchronization



Test Android, Fire OS, and iOS apps on real devices in the Cloud



Collect, View and Export App Analytics



Push Notification Service

Application Services



Build, Deploy and Manage APIs



Low Latency Application Streaming



Managed Search Service



Easy-to-use Scalable Media Transcoding



Email Sending Service



Message Queue Service



Workflow Service for Coordinating Application Components

Enterprise Applications



Desktops in the Cloud



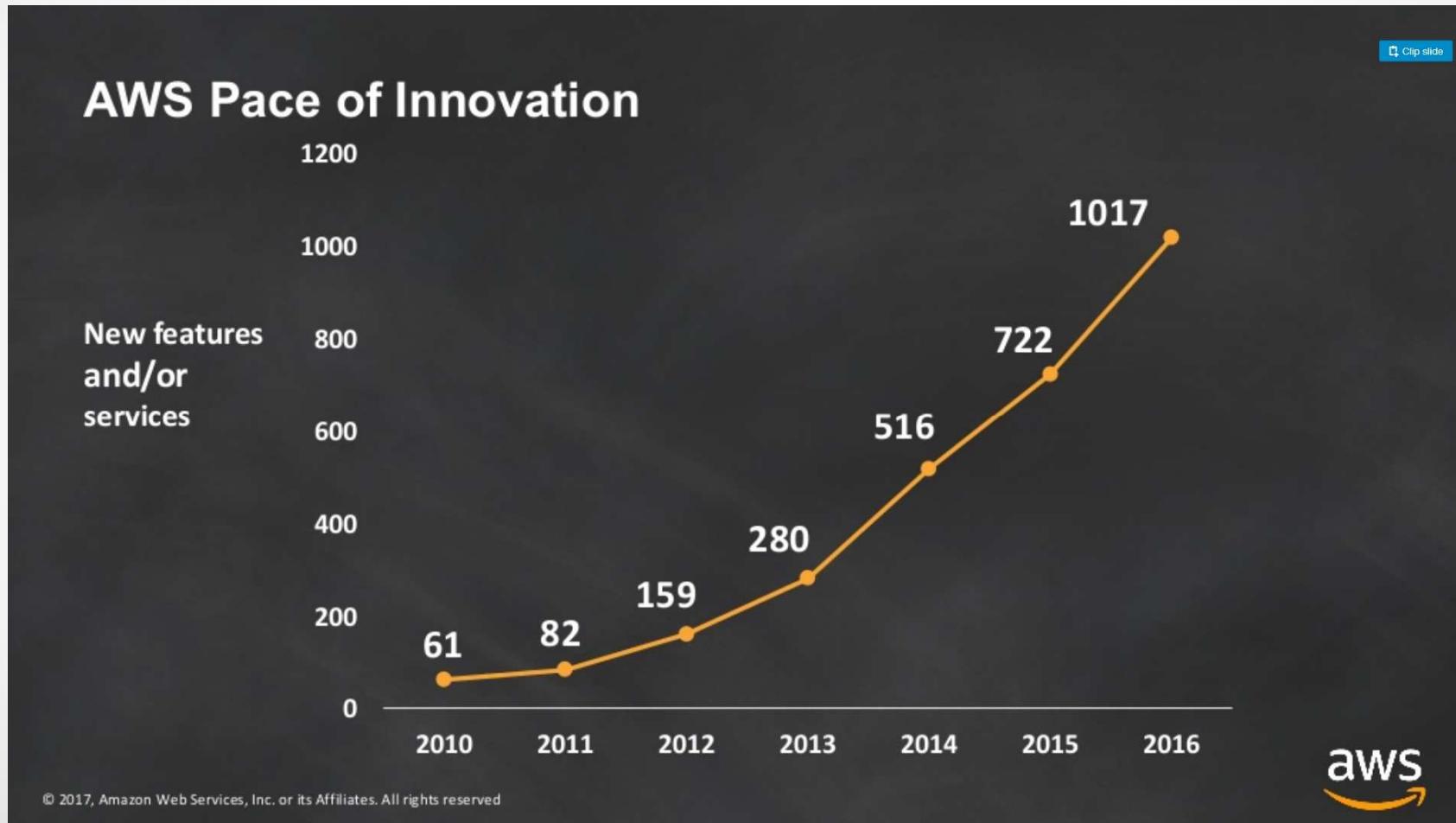
Secure Enterprise Storage and Sharing Service



PREVIEW

Secure Email and Calendaring Service

AWS : catalogue



AWS : marketplace/templates

AWS Marketplace



awsmarketplace

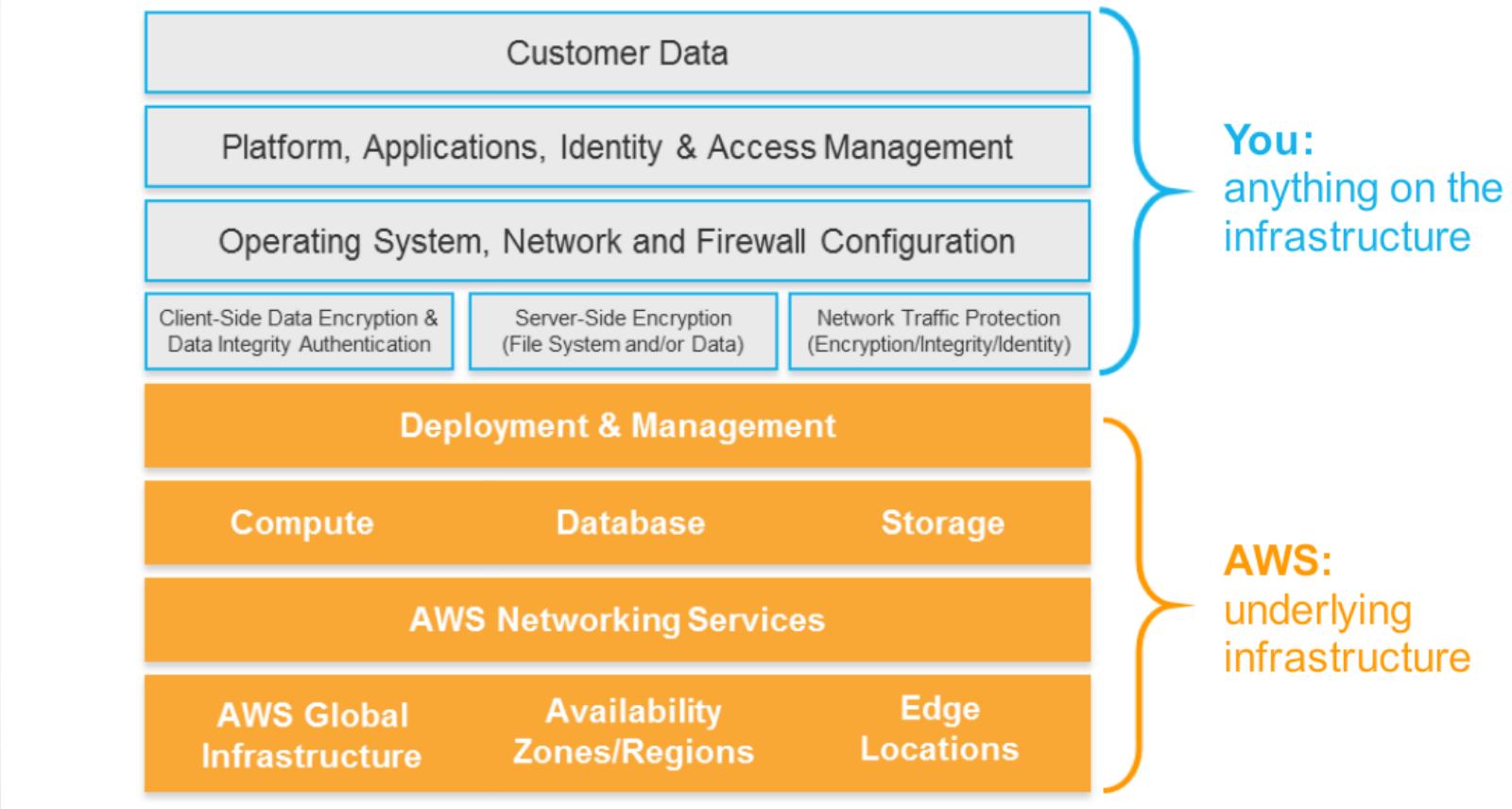
35 Categories

2700 Listings

205,000,000 EC2 Instance Hours

AWS : responsabilités

Shared Security Responsibility Model



AWS : proposition de valeur

AWS Business Professional: Module 1 AWS Value Proposition

Module 1: AWS Value Proposition

Summary

Cost:

- Converting capital expense for variable expense
- Economy of scale allows AWS to pass to lower prices for customers.
- Pay only for what you use
- Save more as you grow with AWS

Elasticity:

- Quickly deploy new applications
- Instantly scale up as the workload grows
- Instantly shut down resources that are no longer required
- When you scale down you don't pay for the infrastructure

Flexibility:

- AWS enables organization to use the programming models, operating systems, databases, and architectures with which they are already familiar.

Security:

- The Shared Responsibility Model provides a global secure infrastructure for AWS and AWS customers.

© 2016 Amazon Web Services, Inc. or its affiliates. all rights reserved.



Training and
Certification

AWS : pay as you go

AWS Business Professional: Module 1 AWS Value Proposition

Module 1: AWS Value Proposition

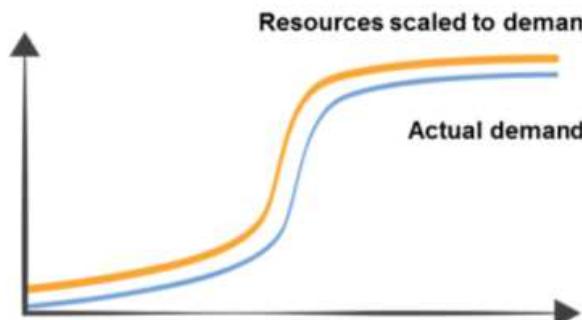
Pay Only for What You Use



Rigid On-Premises Resources



Stop Guessing Capacity



Elastic Cloud-Based Resources



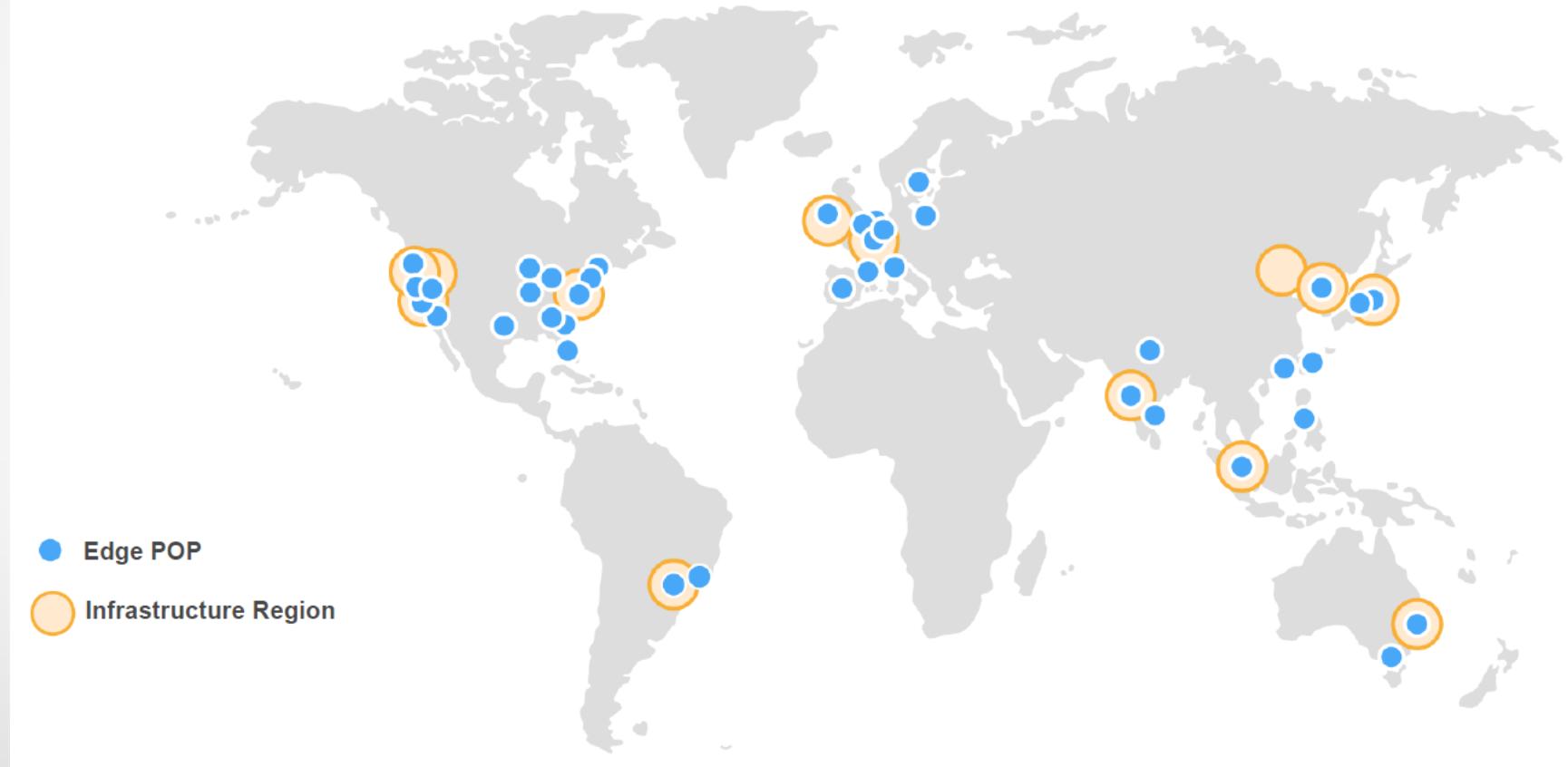
© 2016 Amazon Web Services, Inc. or its affiliates. All rights reserved.



Training and
Certification

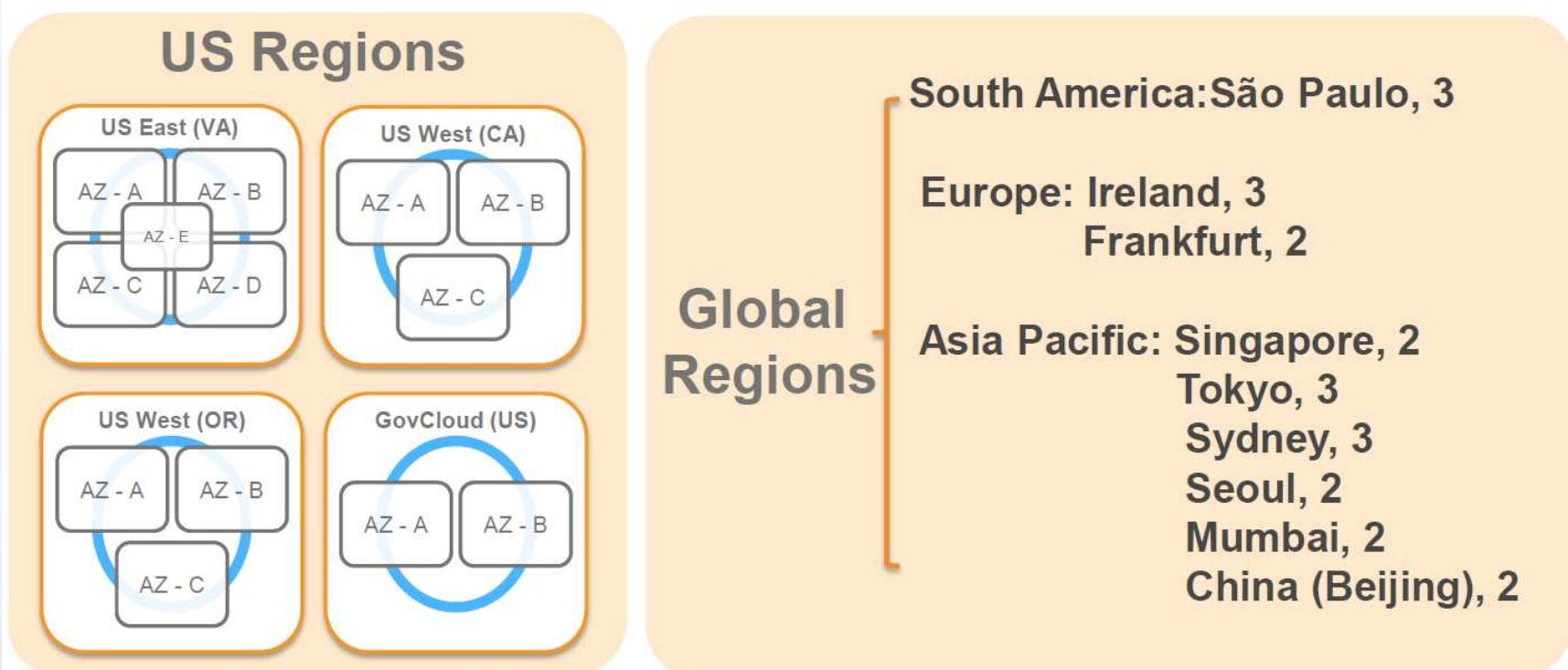
AWS : carte des infrastructures

Global Infrastructure



AWS : regions & AZ

Regions, Availability Zones



Note: Conceptual drawing only. The number of Availability Zones (AZ) may vary.

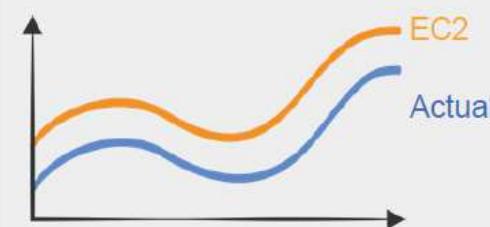
AWS vs virtualisation

AWS Compute Services



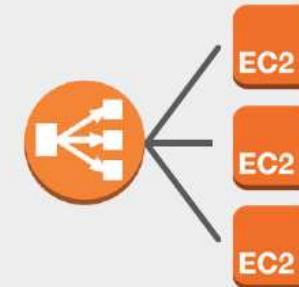
Amazon EC2

Web service
providing **resizable**
compute capacity



Auto Scaling

Automatically **scale**
Amazon EC2
capacity **up or down**



Elastic Load Balancing

Automatically **distribute**
traffic across multiple
Amazon EC2 instances

AWS vs virtualisation

Storage



Images
Videos
Files
Binaries
Snapshots

Amazon S3

A durable, scalable
object store



Amazon EBS

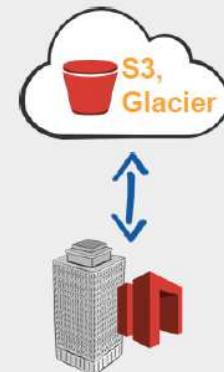
Block storage
for use with Amazon
EC2



Images
Videos
Files
Binaries
Snapshots

Amazon Glacier

Low cost storage
for **archiving** and
backup



AWS Storage Gateway

Integrates on-
premises IT and
AWS storage

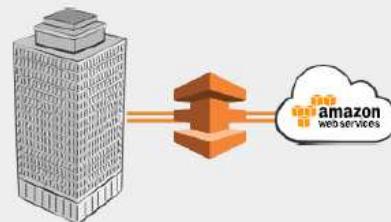
AWS vs virtualisation

Networking Services



Amazon VPC

Private, isolated
section of the AWS
cloud



AWS Direct Connect

Private connectivity
between AWS and your
data center



Amazon Route 53

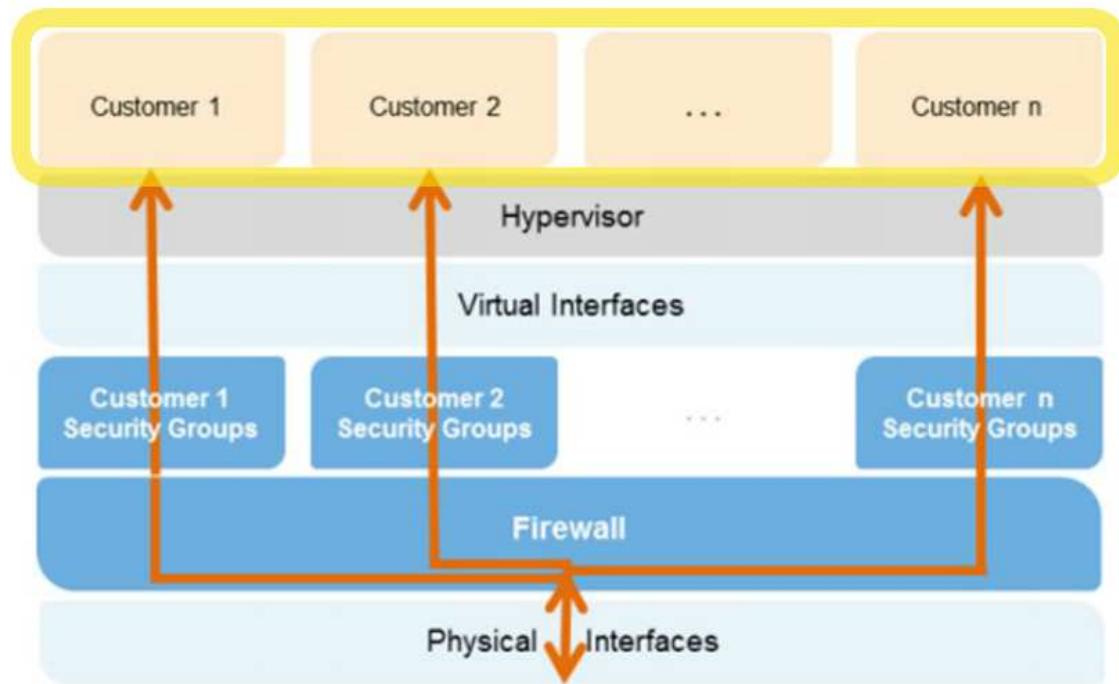
**Domain Name
System (DNS)** web
service

AWS : security groups

Compute Security

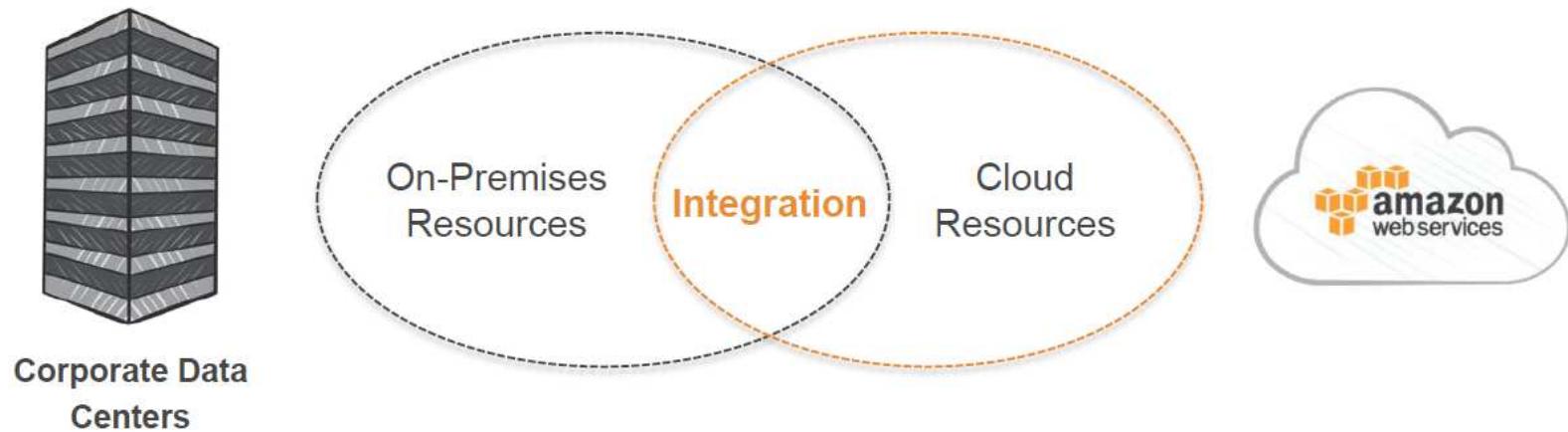
Amazon EC2

- Instances are isolated by the Xen hypervisor
- AWS firewall - within the hypervisor layer
- RAM is separated



AWS : Hybrid

The Cloud Isn't An “All or Nothing” Choice



AWS : Hybrid

AWS Support for Hybrid IT Architectures



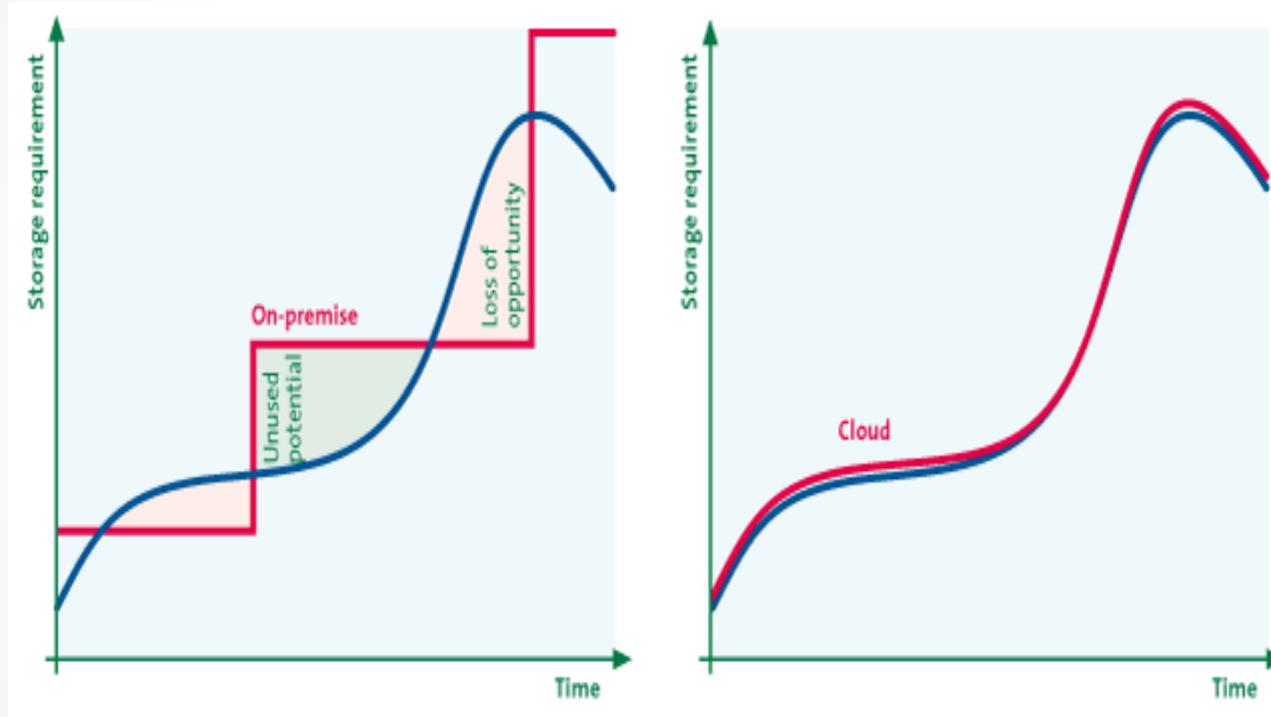
AWS : certifications de sécurité

Audits & Attestations

- **Maintain** alignment with thousands of global requirements and best practices.
- **Validate** a ubiquitous security control environment.
- **Enable** customers to **assess** their organization's compliance with industry and government requirements.



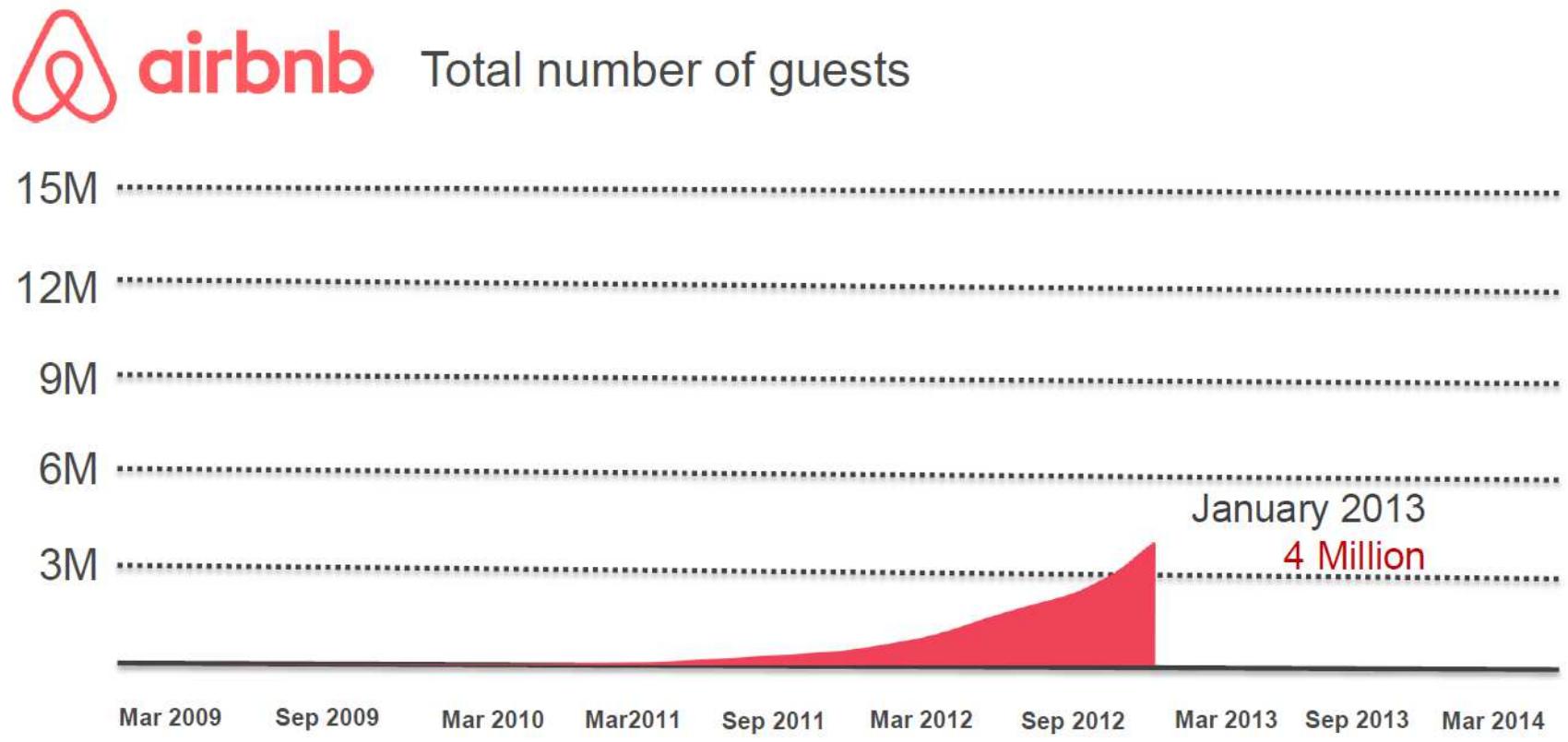
L'élasticité du Cloud au profit du business



Adaptation des ressources du cloud en fonction des besoins, pour éviter les sous-charges et les surcharges !

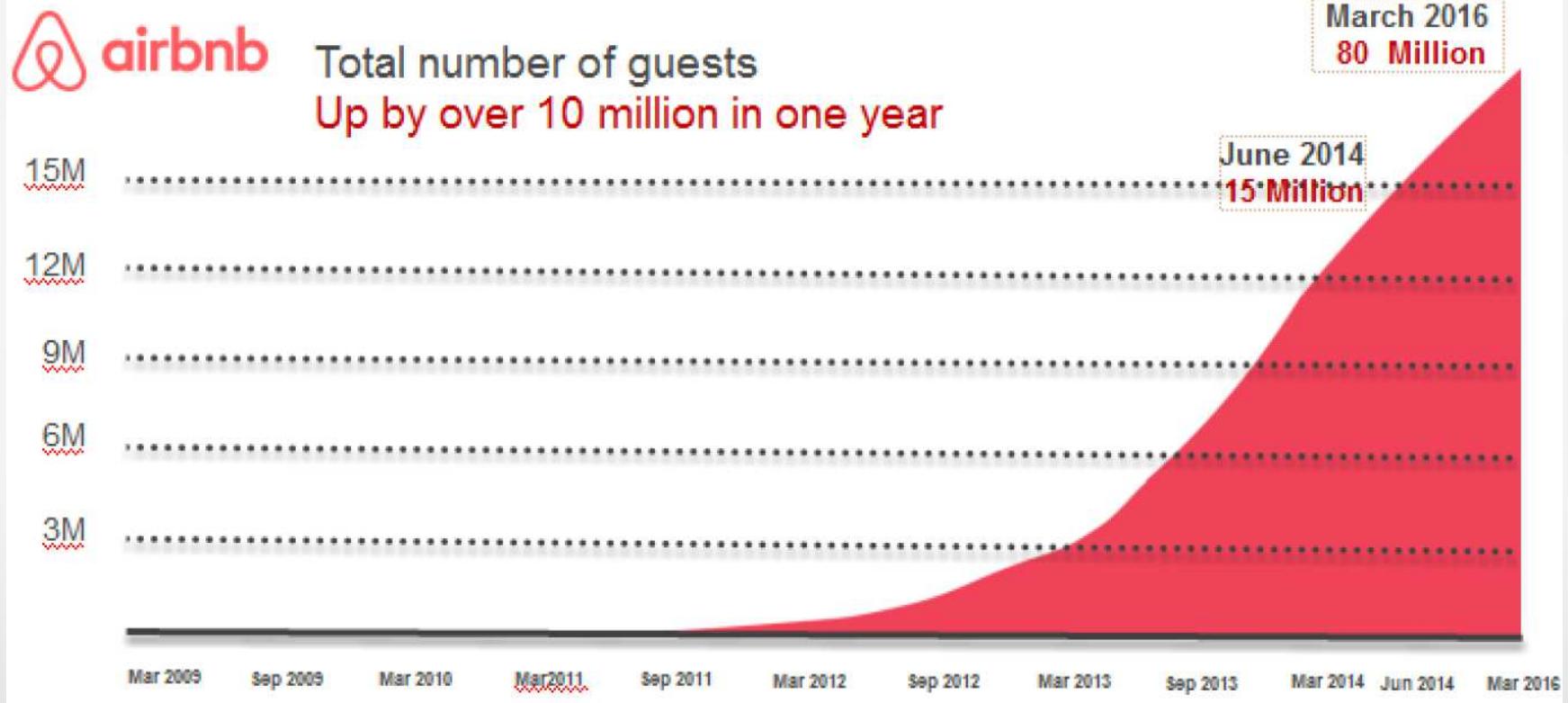
AWS : airbnb

Case Study: Airbnb



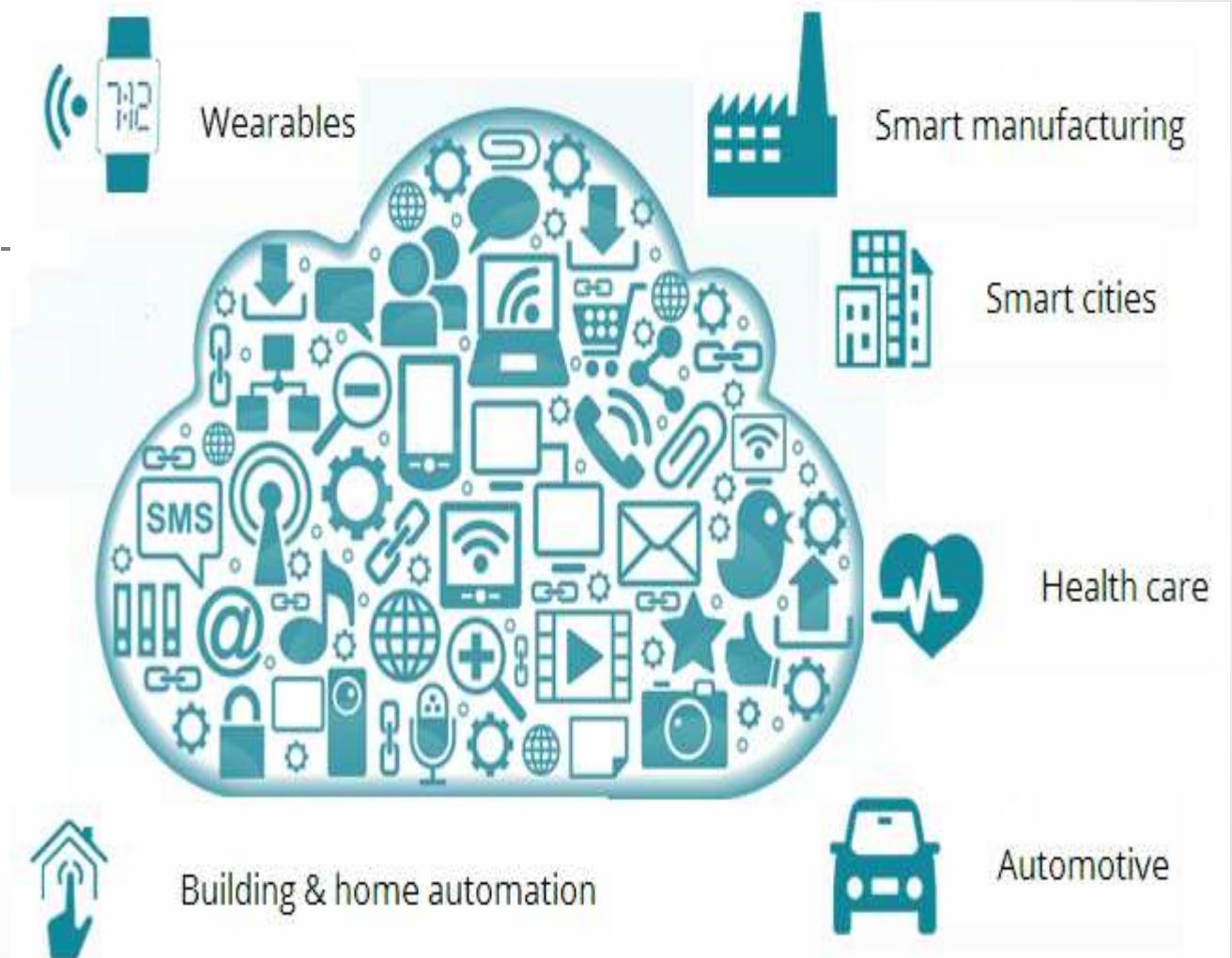
AWS : airbnb

Case Study: Airbnb



Cloud & IoT (Internet of Things)

- MQTT
 - M2M
(Machine-to-Machine)
 - Sigfox
 - LoRa
 - NB-IoT



Cloud & Big-Data & BI (Business Intelligence)

- Hadoop
 - Splunk
 - QlikView



Cloud, les grands acteurs du numérique

- **GAFA(M)** : Google, Apple, Facebook, Amazon (Microsoft)
Big Four ou Big Five du Web
Géants du numérique des années 2000. Microsoft a rejoint le cercle.
- **NATU** : Netflix, Airbnb, Tesla, Uber
Nouveaux géants du numérique qui reposent sur des modèles économiques différents
- **BATX** : **Baidu, Alibaba, Tencent, Xiaomi**
Les géants du Web chinois

Datacenter :
Parce que les serveurs et les données sont
bien quelque part
...



Le Normandie : un datacenter écologique
L'économie d'énergie correspond à la consommation
électrique annuelle totale d'une ville de 15 000 habitants par an



- Une approche automatique du point de vue de l'énergie perdue
 - ✓ Confinement corridor
 - ✓ Vitesse de ventilation différente
 - ✓ Choix d'équipements économiques
- Optimisation du système de refroidissement
 - ✓ Le système de refroidissement représente 65% de l'énergie perdu sur un DC

Orange divise par 3 Datacenter la perte d'énergie et la production de CO₂ associée

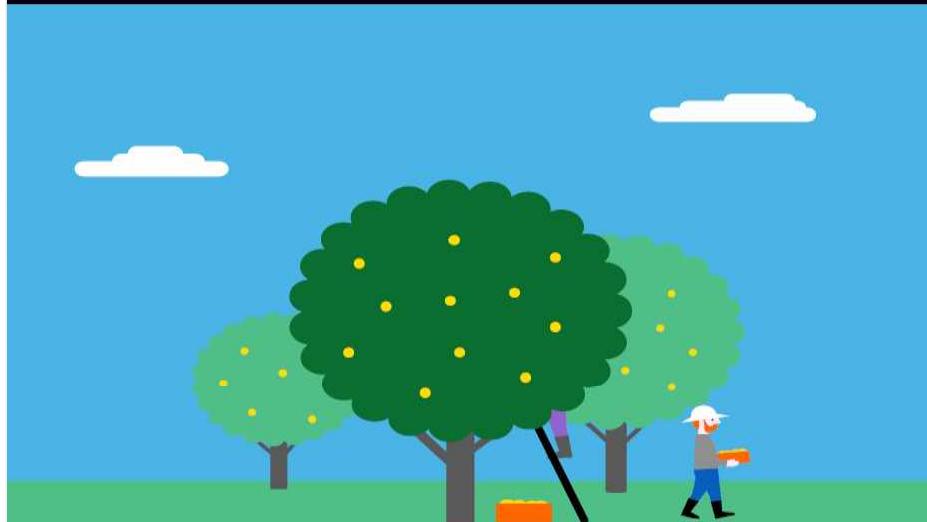
PUE = 1.3

Power Usage Effectiveness

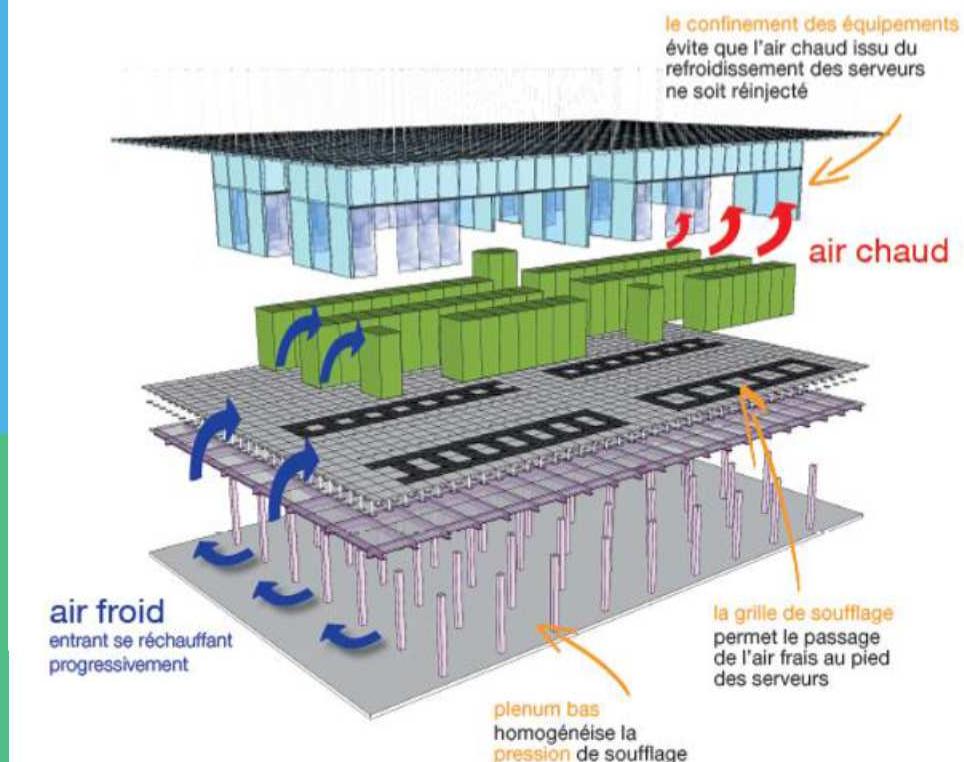
Orange à la pointe de l'innovation

Free cooling total utilisé pendant 85% du temps sur une année et free cooling partiel utilisé 10% du temps supplémentaire

Le free cooling, une nouvelle méthode plus performante



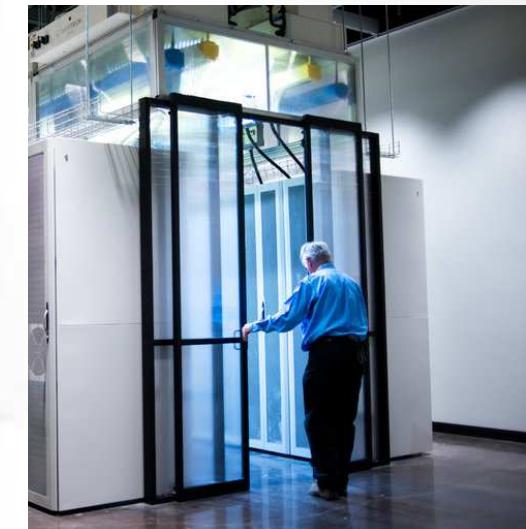
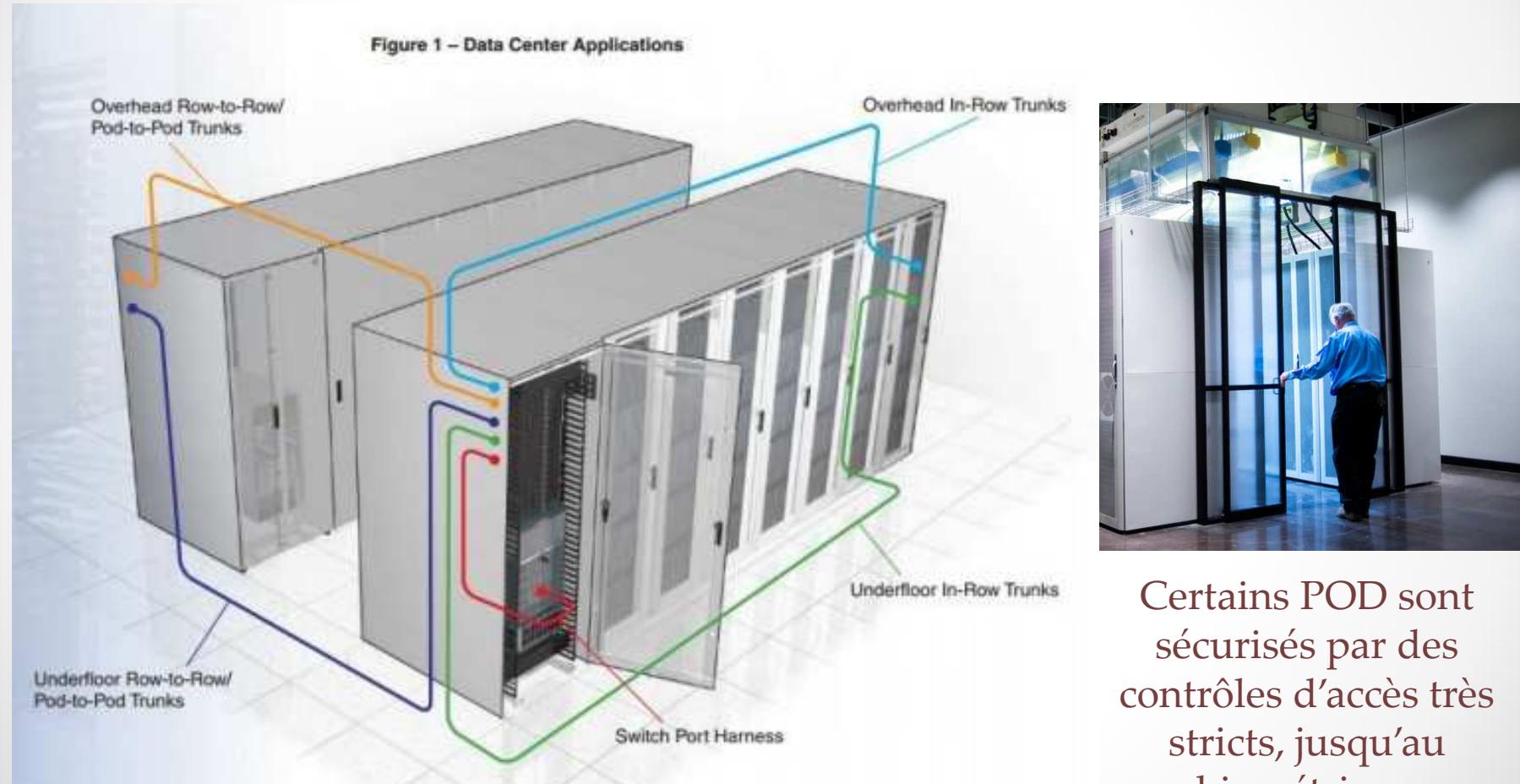
- Le free cooling est un système d'air conditionné économique
 - L'air extérieur est filtré et utilisé pour refroidir les systèmes informatiques.
 - Un système de traitement d'air classique pour la production d'eau glacée est couplé au free cooling pour assurer la meilleure disponibilité.
- Un confinement d'air permet un rendement calorique optimum dans la salle IT



Le plus grand datacenter de France fonctionnant en free cooling

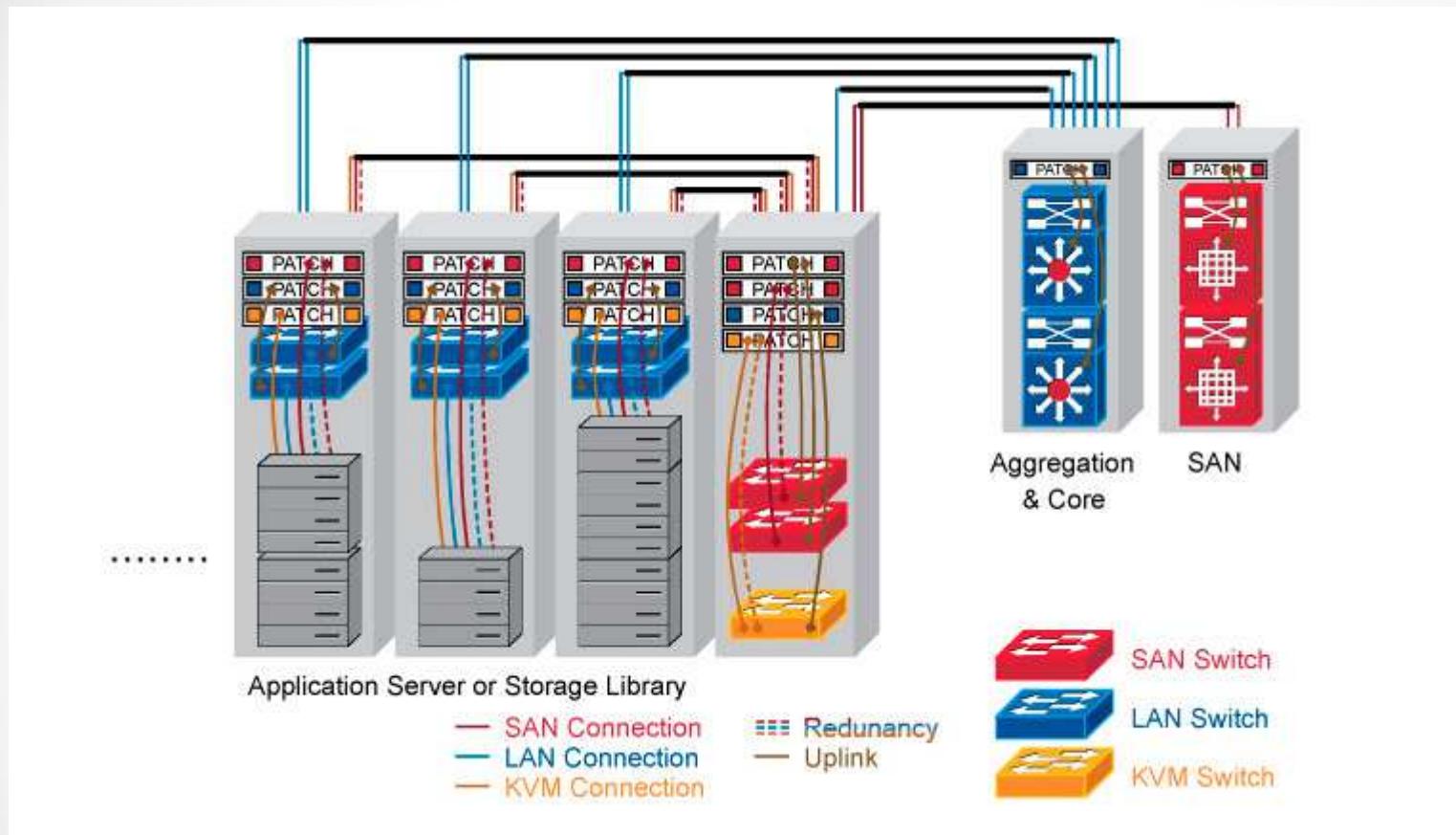
DataCenter : POD

Pod : ensemble de baies uniformes, réparti en deux rangées identiques



Certains POD sont sécurisés par des contrôles d'accès très stricts, jusqu'au biométrique

DataCenter : Top-of-Rack, End-of-Row



Dans un demi POD, chaque baie contient des switchs pour ses équipements (Top-of-Rack), et les éléments mutualisés ou d'agrégation sont situés dans les baies en fin de rangée (End-of-Row)

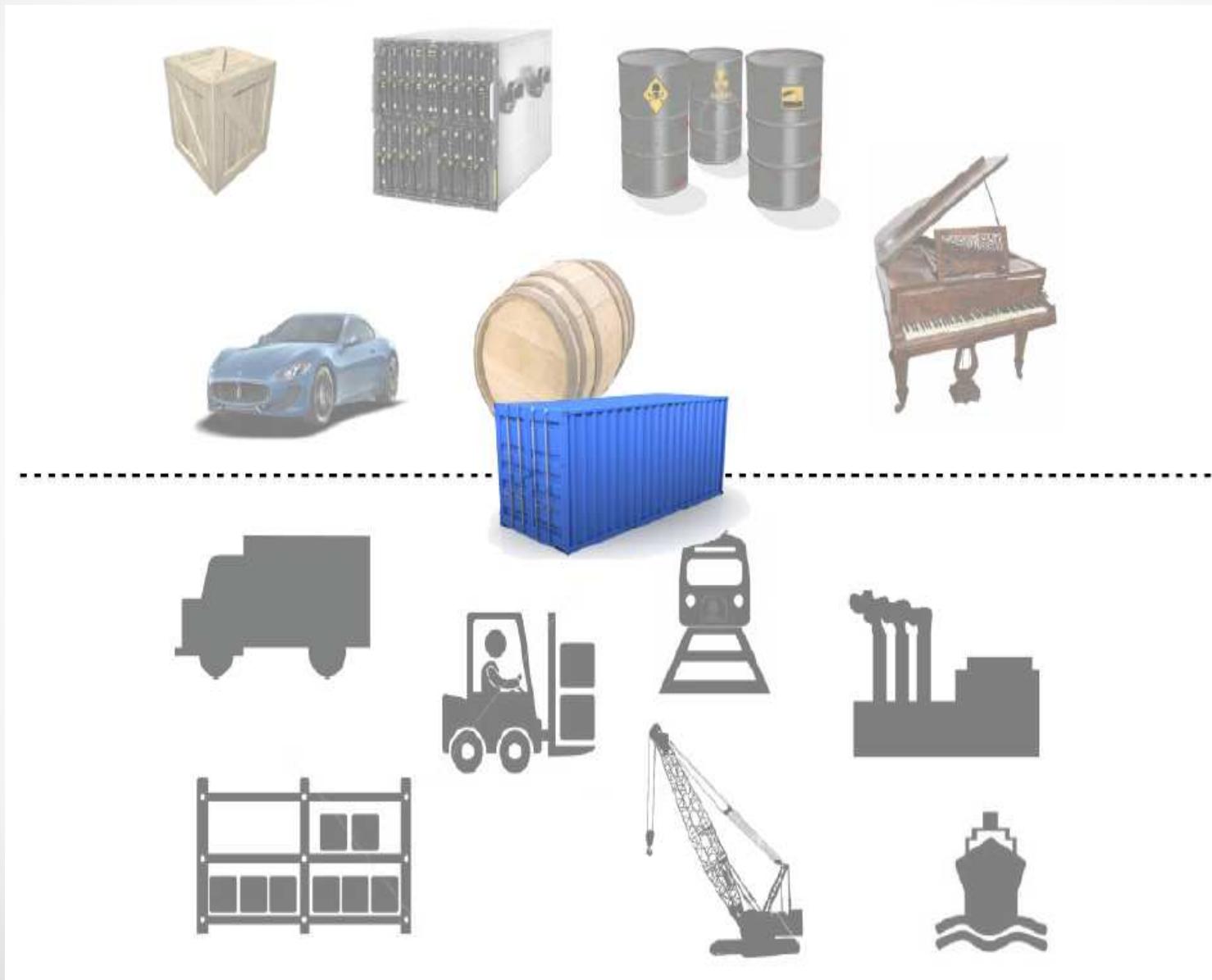
-

Docker :

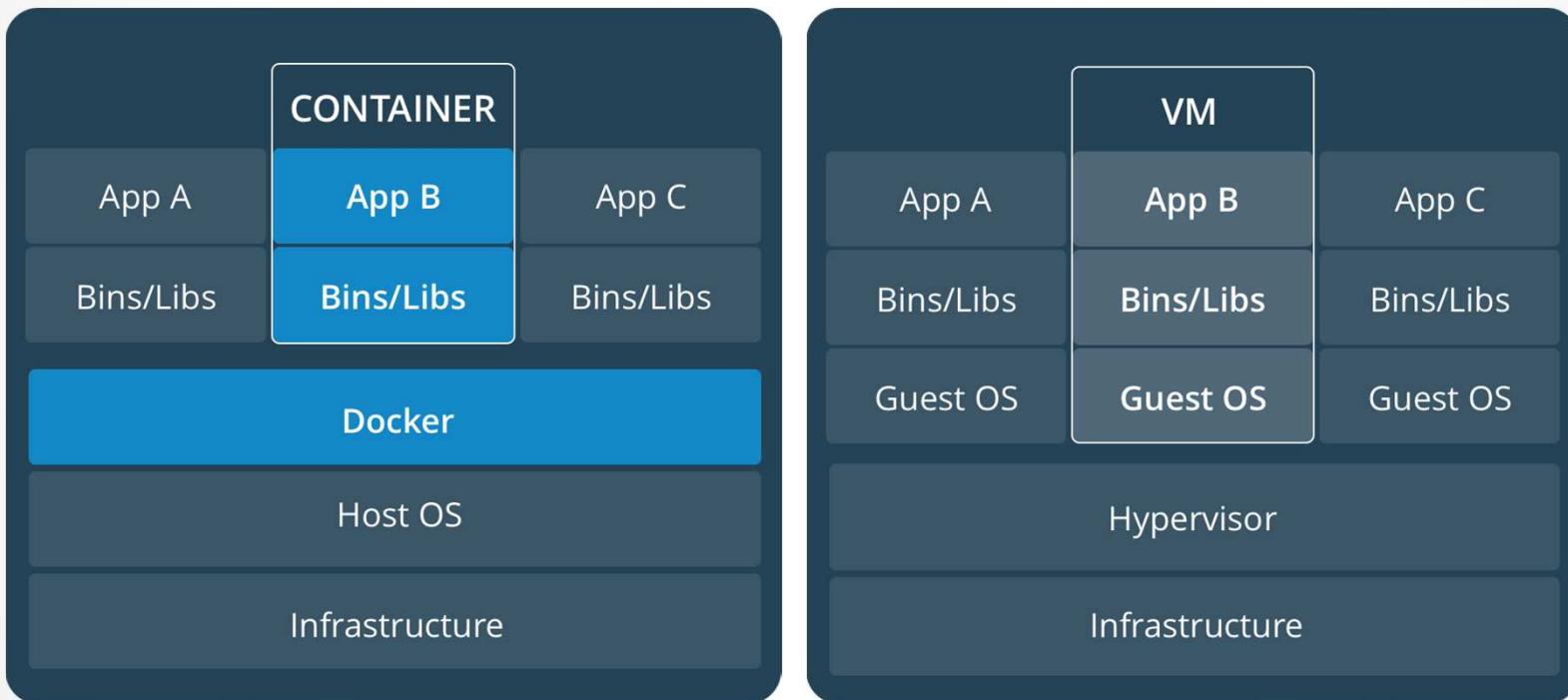
La révolution du container

• • •

Containeurs : la solution universelle

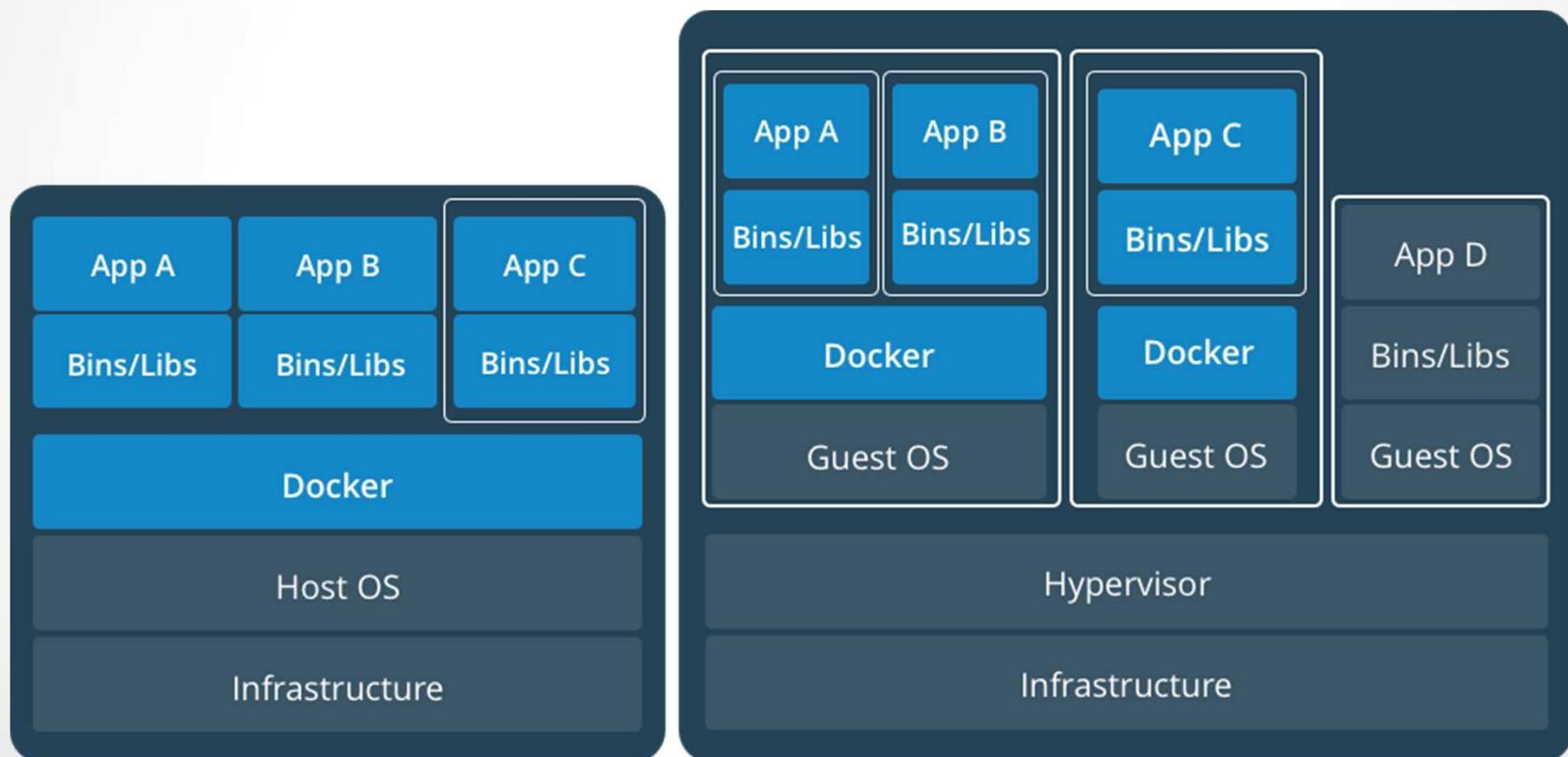


Docker vs VM



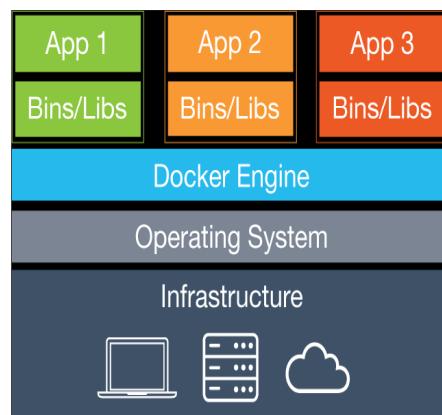
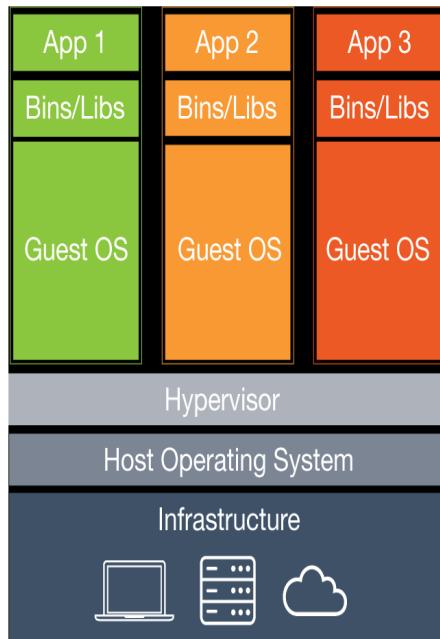
Les containers sont plus légers puisqu'ils partagent le noyau de l'hôte

Over the metal Over the IaaS



Les deux sont possibles

VM vs Conteneur



Conteneur :

- Isolation des processus
- Partage du noyau
- Utilise les fonctionnalités du noyau Linux (cgroups, namespaces, SELinux, ...)
- Pas de système de fichier propre. Utilise un répertoire de l'OS hôte
- Seules les interfaces réseau réelles peuvent être utilisées
- Les ressources sont gérées au niveau de l'OS hôte

- Vers un modèle d'architecture
- Cloud natif



Déploiements immuables

Garantie d'un déploiement identique
Réduit le risque de problèmes au déploiement causés par les différences entre les environnements : le container est autoporteur
Fournit des environnements de test identiques à ceux de production

Rapidité et facilité du déploiement

Puisqu'il s'agit du même container qu'on achemine d'un environnement à l'autre, des étapes se retirent du processus, ce qui en améliore l'efficacité
Aussi, plus les déploiements sont rapides, plus l'environnement affecté est stable.

Facilité du Rollback

Avec le système de container, il est facile de retirer un container livré et en réactiver un précédent (versionning des images).

Passage à l'échelle

Dans le cas où il y a un besoin de créer de nouvelles instances, elles sont plus faciles à mettre en place puisqu'on ne les crée pas de rien

• Les Conteneurs



Modèle d'isolation permettant d'exécuter une application de manière étanche d'un container par rapport à un autre

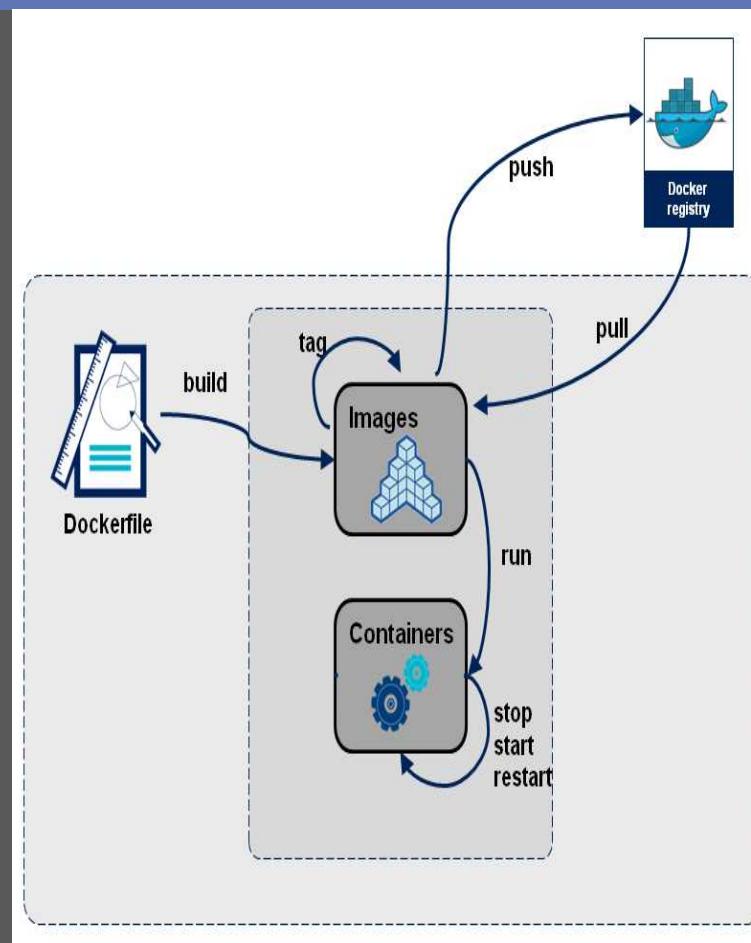
Agnostique sur le contenu

Agnostique sur le transporteur

Isolation

Légèreté

Immuabilité



Le Dockerfile : le 'makefile' du container, les commandes qui permettent de construire l'image

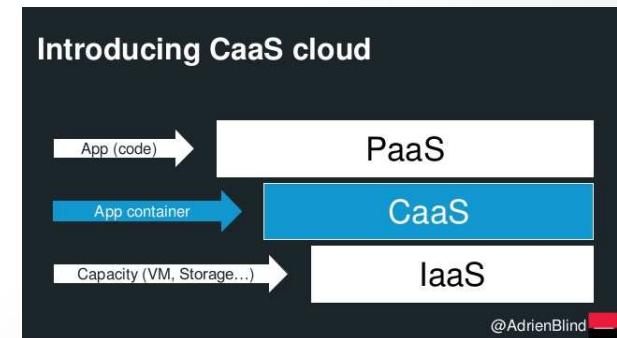
L'image : la version statique du container (son snapshot), elle est faite de layers

Le container : L'instanciation de l'image créée avec le docker file

La Registry : la bibliothèque d'image, stocker et partager des images

Container as a Service

- Nouveau modèle de service : CaaS
- Positionné entre le IaaS et le PaaS
- Techniquement basé sur l'isolation
- Les CSP proposent des offres de CaaS
 - Amazon EC2 Container Service (ECS)
 - Container Engine de Google
 - Pareil côté AZURE
- Orchestrateurs :
 - Kubernetes
 - Docker Swarm
 - Apache Mesos



Les orchestrateurs



L'orchestrateur permet
d'opérer des conteneurs dans un cluster

Service discovery

Permet de localiser les services en fournissant leur adresse et port

High availability Scalability

Maintient d'un nombre minimum d'instances d'un service
+
Variation du nombre d'instances en fonction de la charge

Resource management

Utilisation optimisée des ressources du cluster

Port management

Affecte les adresses internes et externes du conteneur ainsi que le port d'écoute

Vitesses de déploiement

CHANGEMENT DE VITESSE

Datacenter



Déploiement
dans le mois

Pendant des
années

Développement
en cascade

Virtualisation



Déploiement
dans la minute

Pendant des mois

Agile

Docker



Déploiement dans la
seconde

Pendant quelques
heures/minutes

DevOps

DevOps :

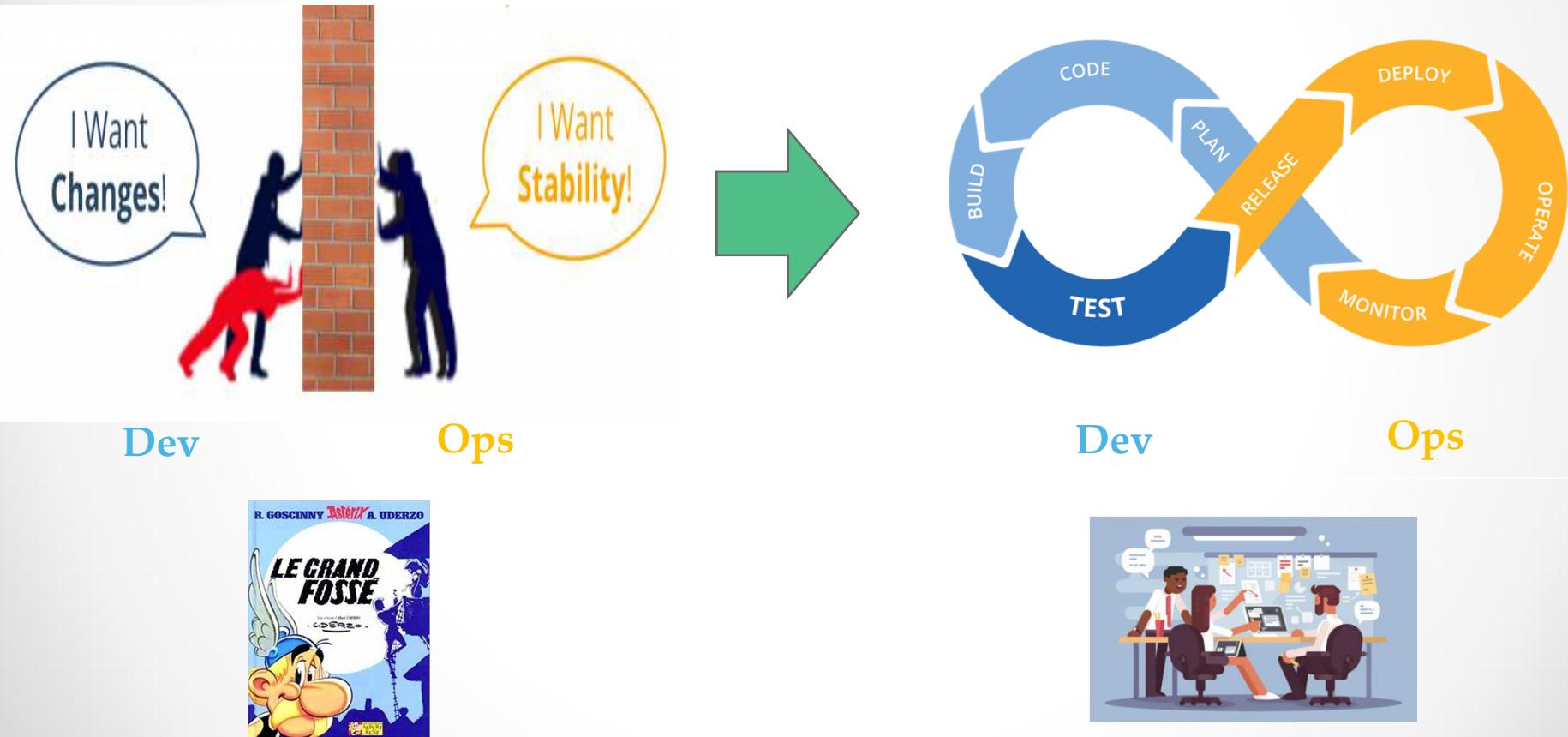
Changement d'organisation

• • •

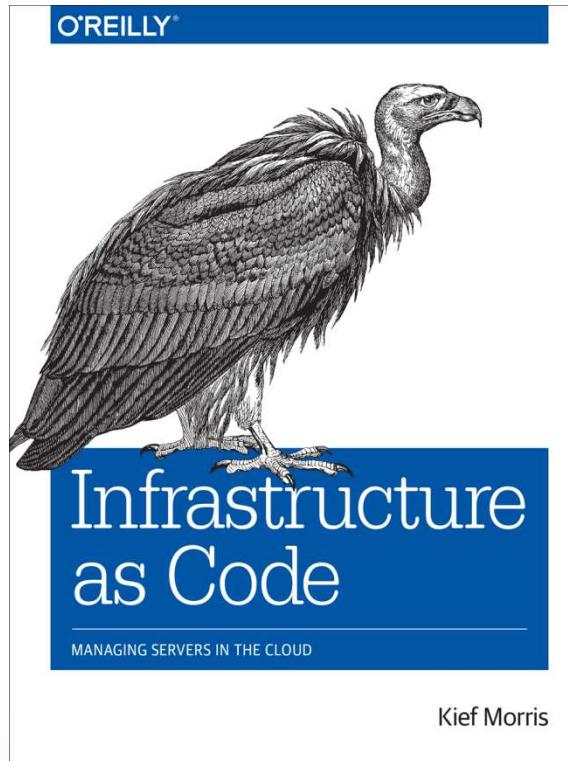
DevOps

- « You build it, you run it »
- http://www.youtube.com/watch?v=_I94-tJlovg
- Equipe mixte “Dev” et “Ops” pour construire un service, le déployer, le maintenir, redéployer, gérer en configuration

de Dev versus Ops à Dev + Ops



Infra as Code (IaC) : le monde du software defined



IaC pour quoi faire ?

- Définir une infra pour déployer des applications.
- Provisionner, modifier, reconstruire un environnement.
- La « configuration » d'une plateforme est du « code » et se versionne (SVN, GIT)
- Un déploiement d'environnement se teste et peut être rejoué.
- Des bouts de code sont réutilisables (capitalisation)

•

Outils/service d'IaC

Solutions proposées par les Cloud Service Provider

- AWS CloudFormation: <https://aws.amazon.com/cloudformation/>
- Azure Resource Manager Templates :
<https://docs.microsoft.com/en-us/azure/azure-resource-manager/resource-group-overview>
- Google Cloud Deployment Manager Templates :
<https://cloud.google.com/deployment-manager/>

Service core d'Openstack

- OpenStack Heat : <https://wiki.openstack.org/wiki/Heat>

Outils multi-plateformes

- Terraform : <https://www.terraform.io/>

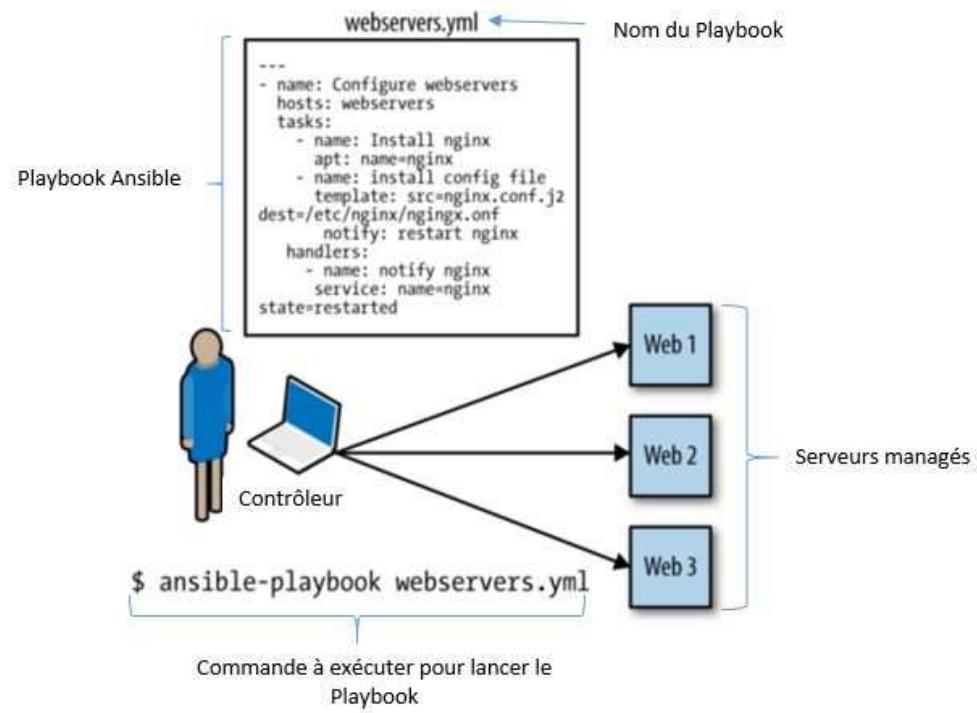
Gestion de configuration

- Déployer des « éléments de configuration »
 - des logiciels (packages)
 - des fichiers de configurations
 - des données applicatives
 - parfois des éléments d'infrastructure
- Architecture
 - Un inventaire (liste de serveurs)
 - Une recette / playbook
 - Avec ou sans agents
- Différents outils :
 - Chef : <https://www.chef.io/>
 - Puppet : <https://puppet.com/>
 - Ansible : <https://www.ansible.com/>
 - SaltStack : <https://saltstack.com/>

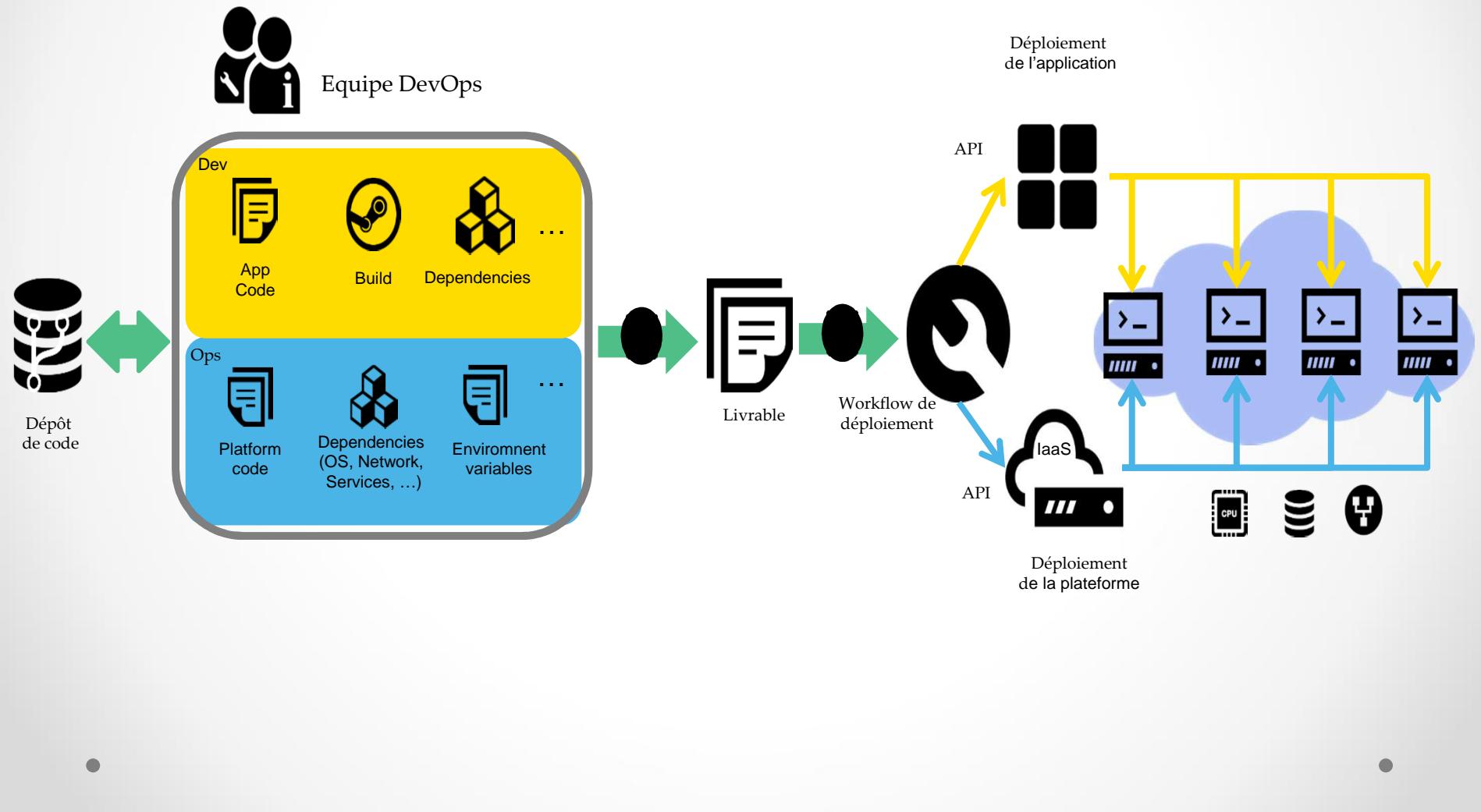
Exemple : Ansible

```
- name: Configure webserver with nginx
hosts: webservers
sudo: True

tasks:
- name: install nginx
apt: name=nginx update_cache=yes
- name: copy nginx config file
copy: src=files/nginx.conf dest=/etc/nginx/sites-available/default
- name: enable configuration
file:
  dest=/etc/nginx/sites-enabled/default
  src=/etc/nginx/sites-available/default
  state=link
- name: copy index.html
template: src=templates/index.html.j2 dest=/usr/share/nginx/html/index.html
mode=0644
- name: restart nginx
service: name=nginx state=restarted
```



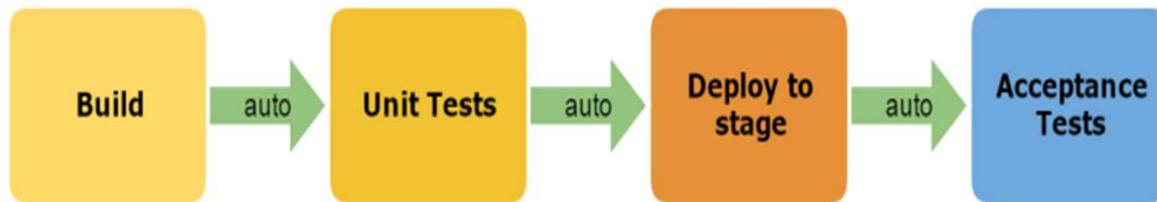
DevOps, infra as code, continuous delivery



Continuous...

CI/CD

Continuous Integration



Continuous Delivery



Continuous Deployment



Concrètement ?

- Pour déployer un service, on peut utiliser
 - Des templates publiques ou privés
 - Des outils orientés « Infra as Code »
 - Des outils de Gestion de Configuration.
 - Il est possible d'utiliser plusieurs méthodes de manière complémentaires.
 - DevOps : rapprocher les compétences Dev et Ops.
 - « CI / CD » : Continuous Integration, Continuous Delivery. Automatisation des tests et des déploiements.
 - Pour profiter pleinement du cloud
-

Cloud Native Applications :

Les applications s'adaptent au cloud

• • •

Cloud Native Application ?



Des applications conçues
pour bénéficier des avantages du cloud

Pour gagner en
vitesse



- ◆ Selfservice
- ◆ Automatisation de la création d'environnements
- ➔ La prise de risque et l'innovation sont possibles

Pour passer à l'échelle



Avec l'augmentation, planifiée ou non, du volume ou du trafic du service, l'architecture Cloud permet de répondre au besoin de passage à l'échelle.

Pour une meilleure disponibilité

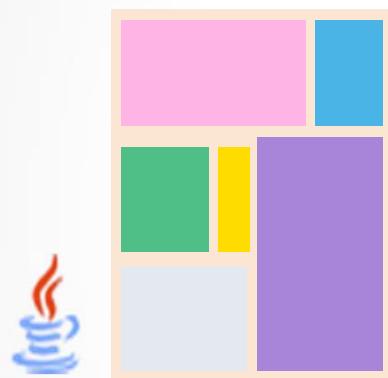


- ◆ Résistance et isolation aux pannes.
- ◆ Redémarrage automatique
- ◆ Monitoring sur des métriques business,
- ◆ Triggers pour lancer des actions automatiquement

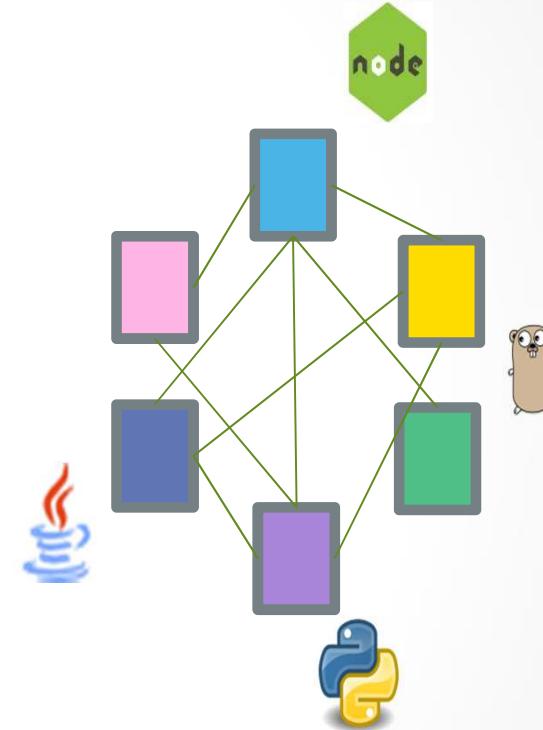
- **Prêt pour des applications Cloud natives**
- Niveau de maturité d'une application Cloud (ODCA)

Maturity Level	Description
Level 3: Adaptive	<ul style="list-style-type: none"> • Application can dynamically migrate across infrastructure providers without interruption of service. • Application can elastically scale out/in appropriately based on stimuli.
Level 2: Abstracted	<ul style="list-style-type: none"> • Services are stateless. • Application is unaware and unaffected by failure of dependent services. • Application is infrastructure agnostic and can run anywhere.
Level 1: Loosely Coupled	<ul style="list-style-type: none"> • Application is composed of loosely coupled services. • Application services are discoverable by name. • Application compute and storage are separated. • Application consumes one or more cloud services: compute, storage, network.
Level 0: Virtualized	<ul style="list-style-type: none"> • Application runs on virtualized infrastructure. • Application can be instantiated from an image or script.

- Monolithe vs Microservices



Application monolithique



Application microservices

- Vers un modèle d'architecture Cloud nat
- Open Data Center Alliance(ODCA)



Formée en Octobre 2010, l'Open Data Center Alliance (ODCA) est une organisation de leaders de l'IT qui ont décidé de porter d'une voix unifiée les besoins et attentes en termes de Data Center et de Cloud.

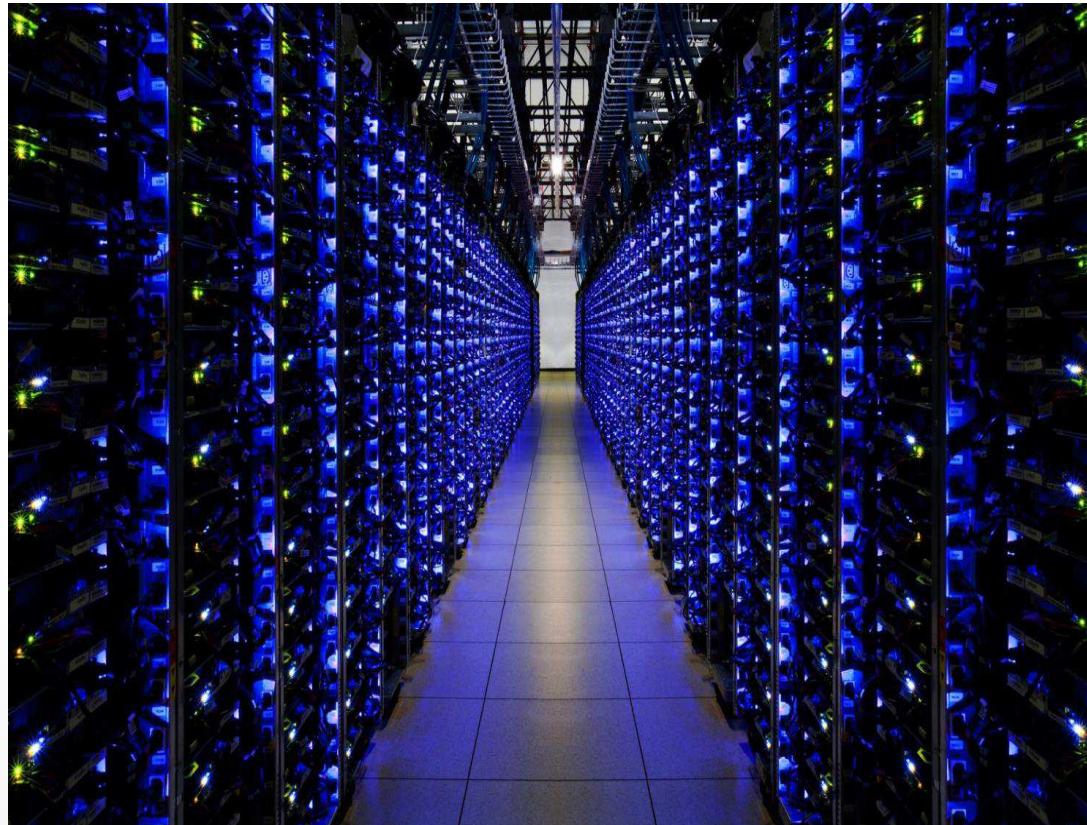
La mission de l'ODCA est d'accélérer la transformation vers le Cloud, notamment en partageant les bonnes pratiques de construction et d'implémentation de services dans le Cloud

Principes structurels proposés par l'ODCA

No.	Principle	Attribute	Priority
1	Resilient to failure	<ul style="list-style-type: none"> Resiliency is designed into the application, rather than wrapped around it after the fact. Failures in cloud infrastructure are handled fluidly without interruption of service. 	High
2	Resilient to latency	<ul style="list-style-type: none"> Applications adapt gracefully to latency rather than timing out/failing. 	High
3	Secure	<ul style="list-style-type: none"> Applications are based on secure lifecycle standards and include built-in security. Data at rest and in transit is encrypted. APIs are protected by authentication and authorization. 	High
4	Location independent	<ul style="list-style-type: none"> Applications discover services dynamically rather than relying on hard-coded dependencies. 	High
5	Elastically scalable	<ul style="list-style-type: none"> Applications respond to demand levels, growing and shrinking as required, in and among clouds. 	High
6	SOA/Compose-ability	<ul style="list-style-type: none"> Applications consume and expose web services with APIs discoverable at runtime. The structure incorporates small, stateless components designed to scale out. 	High
7	Designed for manageability	<ul style="list-style-type: none"> Applications are instrumented and expose metrics and management interfaces. 	Medium
8	Infrastructure independent	<ul style="list-style-type: none"> Applications make no assumptions about the underlying infrastructure, using abstractions in relation to the operating system, file system, database, and so on. 	Medium
9	Defined tenancy	<ul style="list-style-type: none"> Each application should have a deliberate, defined single tenancy or multitenancy model. 	Medium
10	Available end-user self-service	<ul style="list-style-type: none"> Users should be able to register themselves to use the app through a self-service registration interface, without entering an IT service request. 	Medium
11	Bandwidth aware	<ul style="list-style-type: none"> APIs and application protocols are designed to minimize bandwidth consumption. 	Low
12	Cost and resource consumption aware	<ul style="list-style-type: none"> Application architecture is designed to minimize costs due to bandwidth, CPU, storage consumption, and I/O requests. 	Low

Serverless

En route vers le no-ops





A serverless world...

- "Build and run applications
- without thinking about servers
- ...pay per request not for idle"



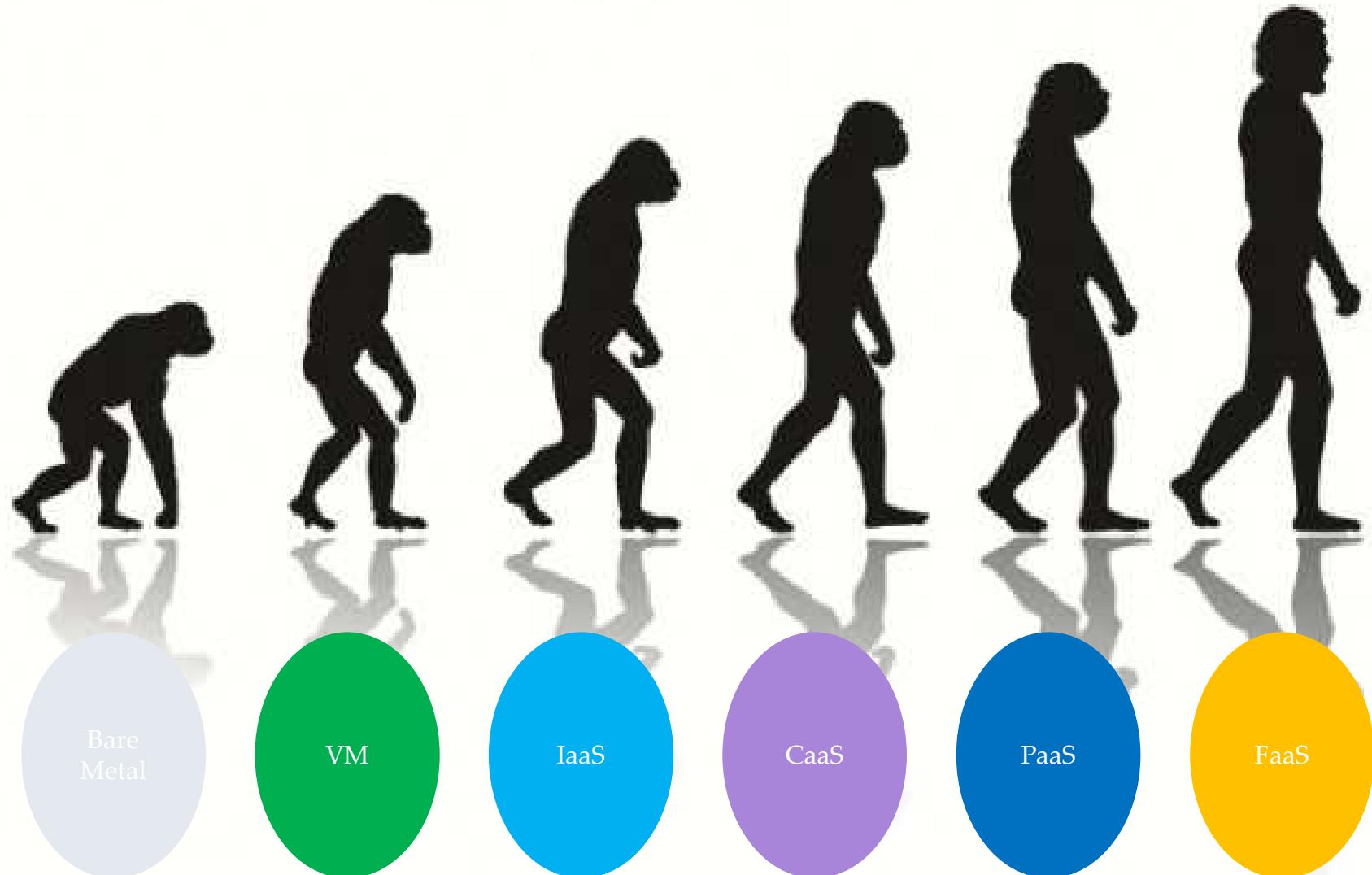
Serverless : Définition



- Une architecture de plateforme *Cloud Native*
- pour des *traitements courts et sans états*
- déclenchés par des événements
- qui passe à l'échelle de façon *transparente*
- et *facturée à l'usage réel* (à la milliseconde)

•

Serverless : Une évolution naturelle



Serverless



- **FaaS :**

- Function as a Service

- **BaaS :**

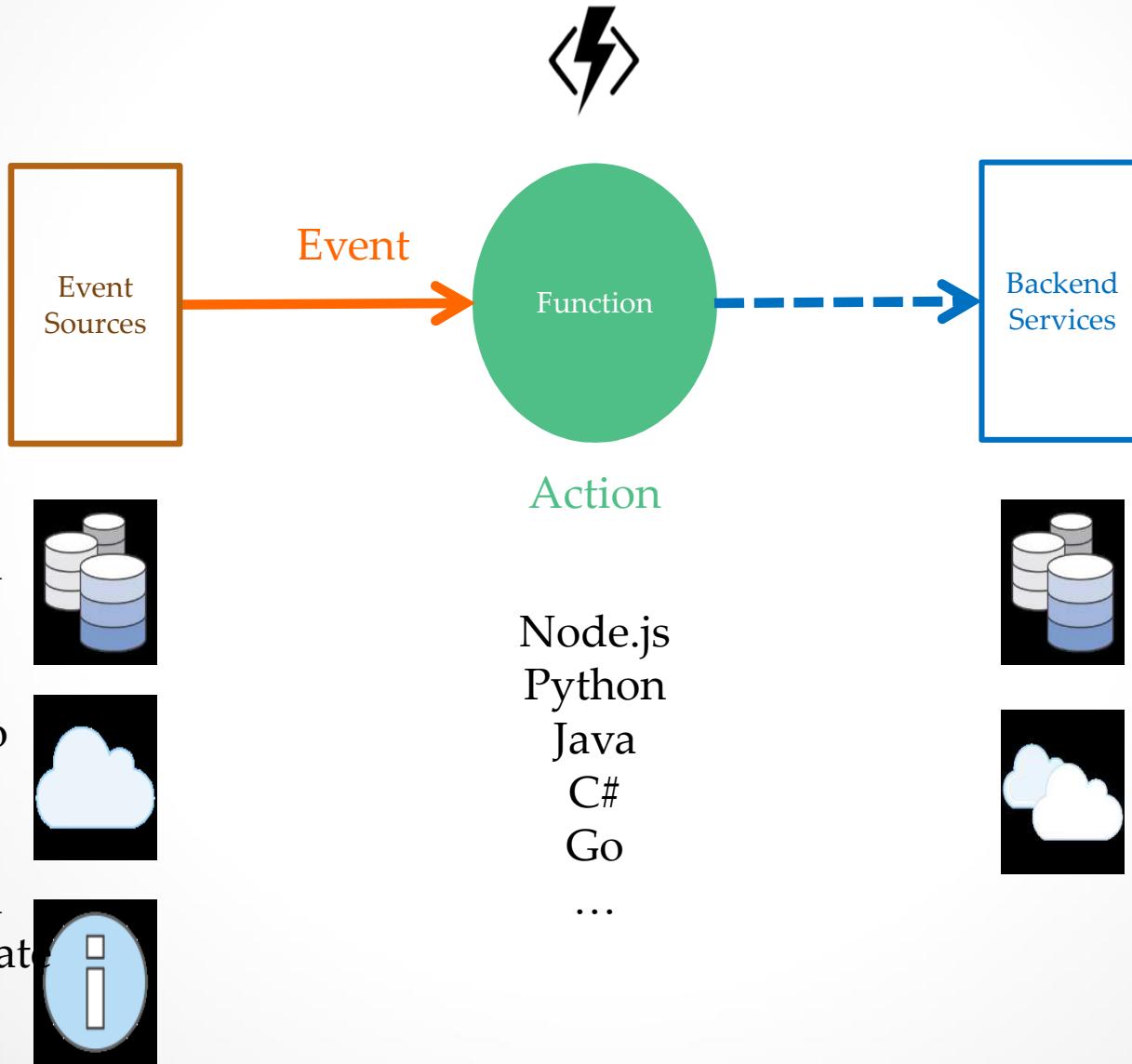
- Backend as a Service

- Les principales implémentations :

- AWS Lambda
 - Azure Functions
 - Google Functions
 - IBM OpenWhisk
 - Oracle fn



Principle





Serverless - Avantages

- Pas de gestion de l'infrastructure
- Elasticité
- Haute disponibilité
- Facturation au consommé réel (pas d'usage = pas de coût)
- Time to Market réduit
- Gestion des ressources optimisée (pour le fournisseur)

Serverless - Inconvénients



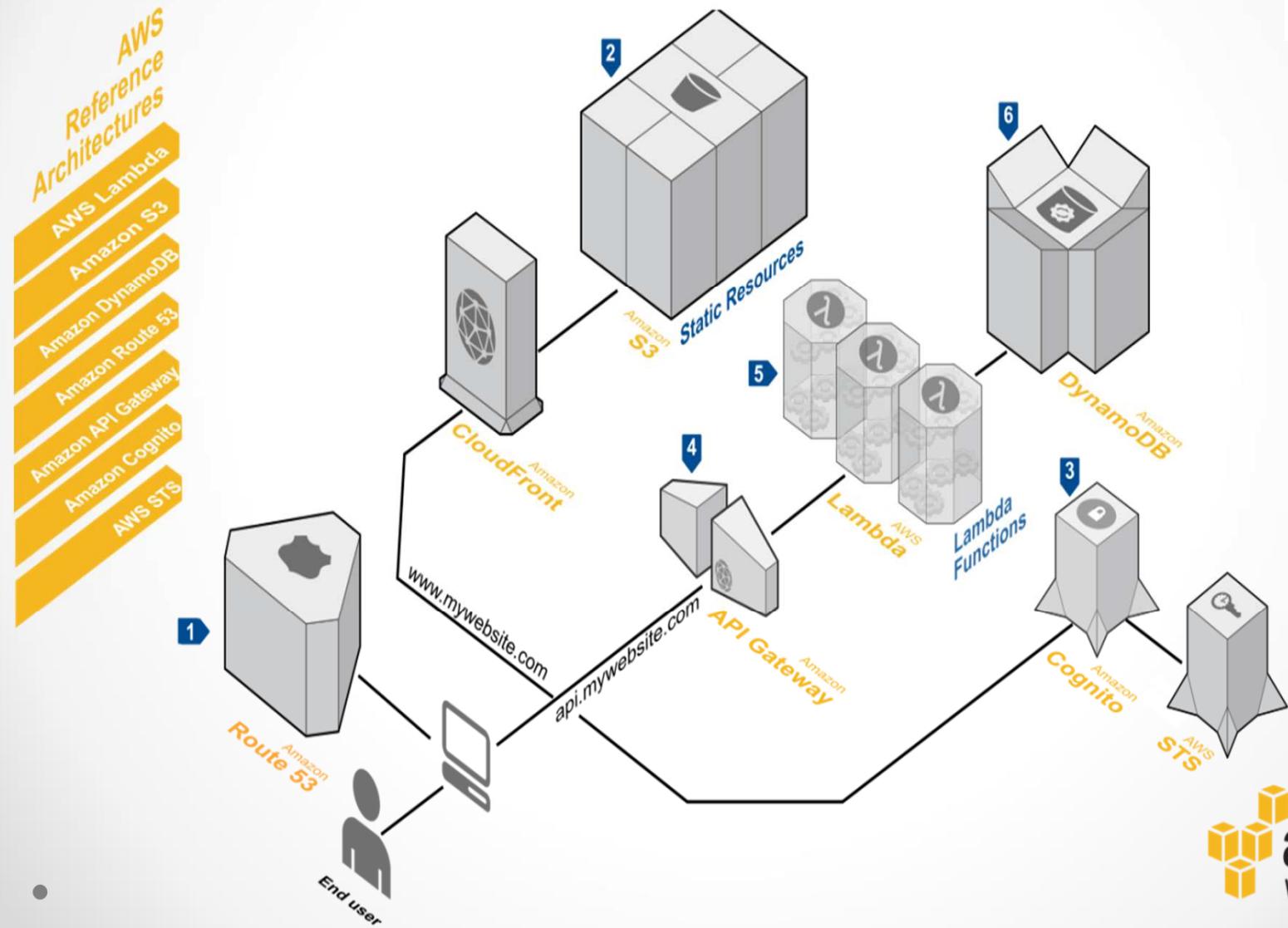
- Fonctions sans état
- Lock-in des clouds providers
- Sécurité : infra mutualisée et exposition des fonctions
- Durée d'exécution limitée (qq minutes)
- Latence au démarrage (qq ms)
- Tests (en particulier tests d'intégration)
- Supervision et debugging
- Maîtrise des coûts

Serverless – Use cases



- Microservices
- IoT
- Batch processing
- devOps
- Gestion d'événements
- ...

Application Web



 **amazon**
web services²

A serverless world...

“ Build and run applications without thinking about servers

... pay per request not for idle



Scales with usage



High availability built-in



Never pay for idle

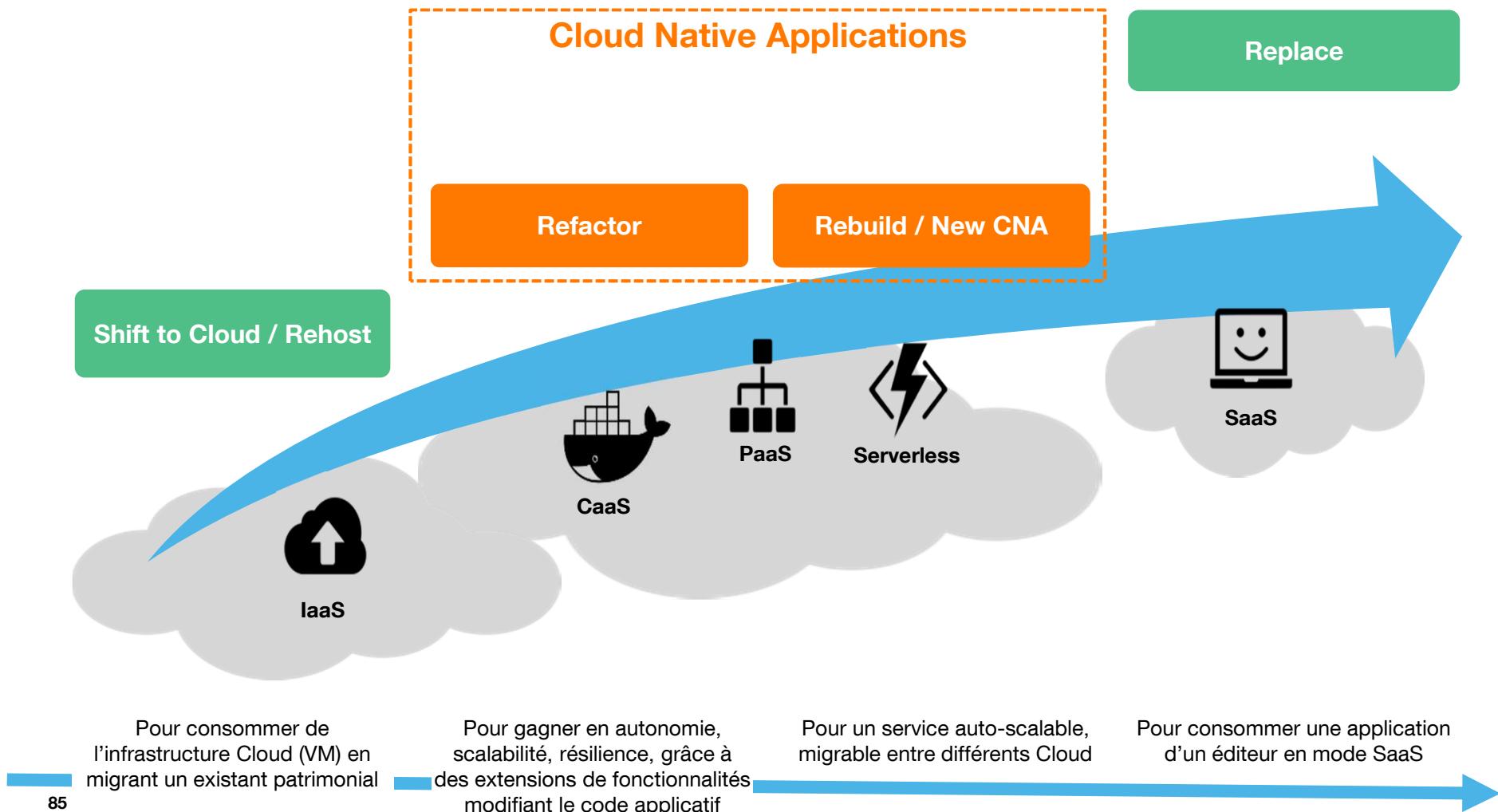


No servers to provision or manage

Référentiels, bonnes pratiques et patterns

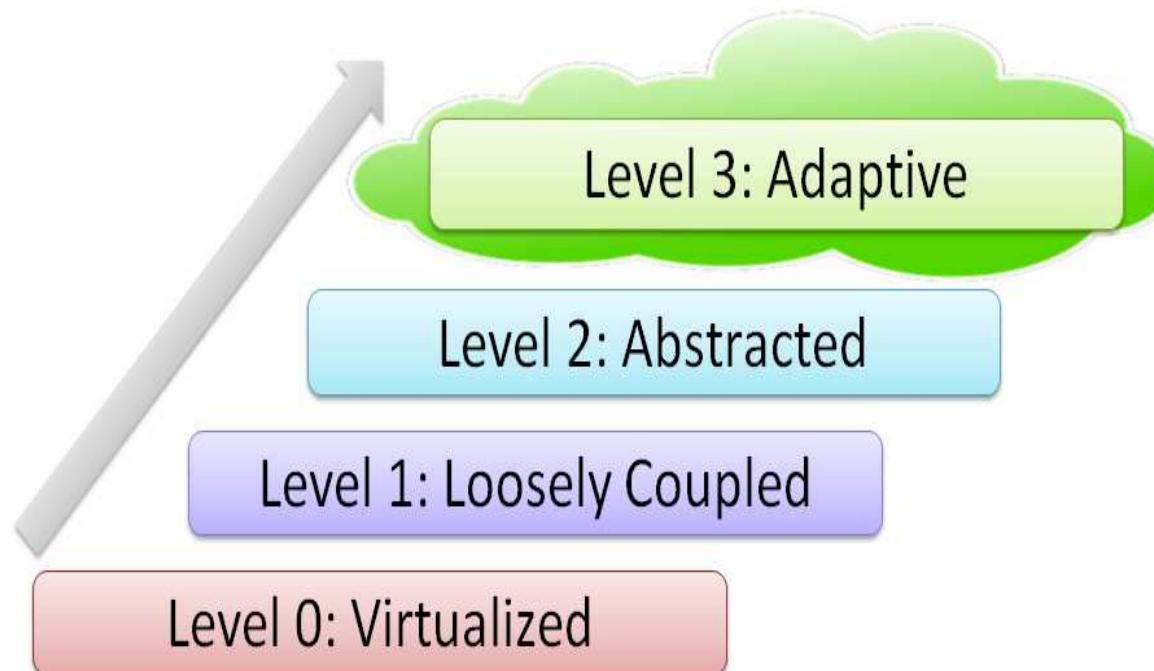
...

Différentes stratégies de migration d'application vers le Cloud



Niveau de maturité d'une application Cloud

ODCA - Open Data Center Alliance



12 factors



- Une méthodologie pour construire des applications cloud native
- Développée par des ingénieurs d'Heroku
- Orientée applications sans état pour des déploiements sur des plateformes cloud

<https://12factor.net/>



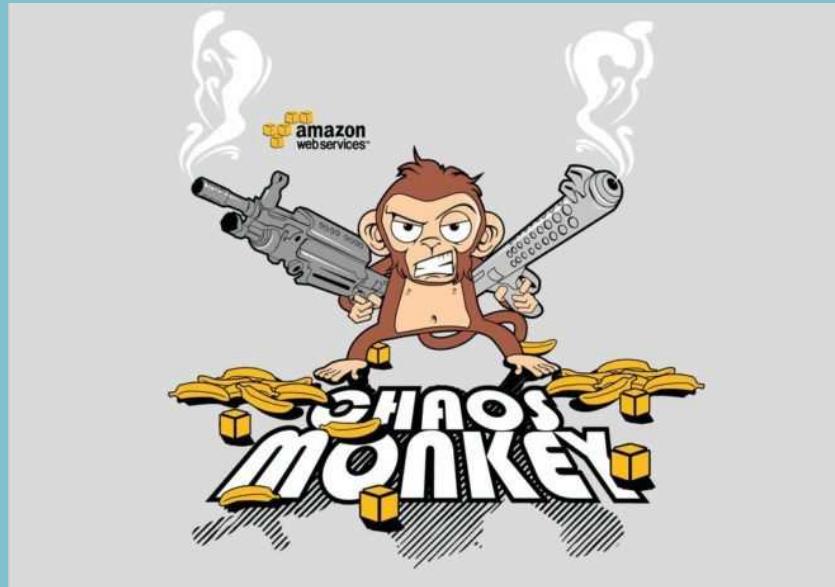
Les patterns cloud

Quelques exemples...

Les patterns cloud (1/4)

- Design for failure

Anticiper les défaillances (matériel, réseau, relocalisation des instances...)



Les patterns cloud (2/4)

- **Circuit breaker**

S'intègre entre l'application et le service distant. Renvoi immédiatement une réponse lorsque le service est indisponible sans attendre de timeout et sans envoyer la requête au service.

Ex : Hystrix

- **Load balancer**

Distribue les requêtes vers les différentes instances du service. Il est soit centralisé, soit intégré au client.

Ex : HA Proxy (centralisé), Ribbon (client side)

- **Service discovery**

Il permet de mettre à disposition des clients des informations dynamiques sur les services : localisation (IP, port), configuration, état

Ex : Etcd, Consul, Zookeeper, Netflix Eureka

Les patterns cloud (3/4)

- **API Gateway**

Fournit au clients un seul point d'entrée aux services backend. Cela permet de masquer la complexité des nombreux micro-services et des différents protocols utilisés en interne. C'est une implémentation du pattern Façade.

- **Health Manager**

Monitor les process et le statut des services (health) pour assurer que les ressources déployées sont celles demandées et qu'elles fonctionnent correctement : “desired state” vs “actual state”

Ex : Fleet (Docker), Kubernetes, CloudFoundry HM9000

- **Health Endpoint Monitoring**

Implémente des tests fonctionnels des services en plus des tests techniques (process). Typiquement, vérifie le code retour HTTP, les temps de réponses, ...

Les patterns cloud (4/4)

- **Blue-Green deployment**

Déploiement progressif sur l'ensemble des instances pour éviter une interruption de service.

- **Centralized monitoring**

Recueil, agrégation, indexation, et analyse des événements dans un service de monitoring centralisé.

- **Idempotence**

Assure que des duplications de messages ou des données erronées n'affectent pas le fonctionnement de l'application

- **Feature Flipping**

Activation/désactivation de fonctionnalités en production à chaud pour tout ou partie des utilisateurs.

Chaos Engineering

Appliquer le principe d'**anti-fragilité** qui consiste à vérifier la **robustesse et la résilience** de la plateforme en introduisant des **pannes et des erreurs**

- Chaos Monkey
- Chaos Gorilla
- Chaos Kong
- Janitor Monkey
- Doctor Monkey
- Compliance Monkey
- Latency Monkey
- Security Monkey



Sécurité dans le cloud



- **Cloud Security Alliance (CSA)**
- <https://cloudsecurityalliance.org/>

Organisation à but non lucratif ayant pour mission de « promouvoir l'utilisation de bonnes pratiques afin d'assurer la sécurité au sein des environnements de cloud computing et de fournir des informations sur les utilisations du cloud computing, dans le but de contribuer à la sécurité de l'informatique sous toutes ses formes ».

-

Que retenir ?

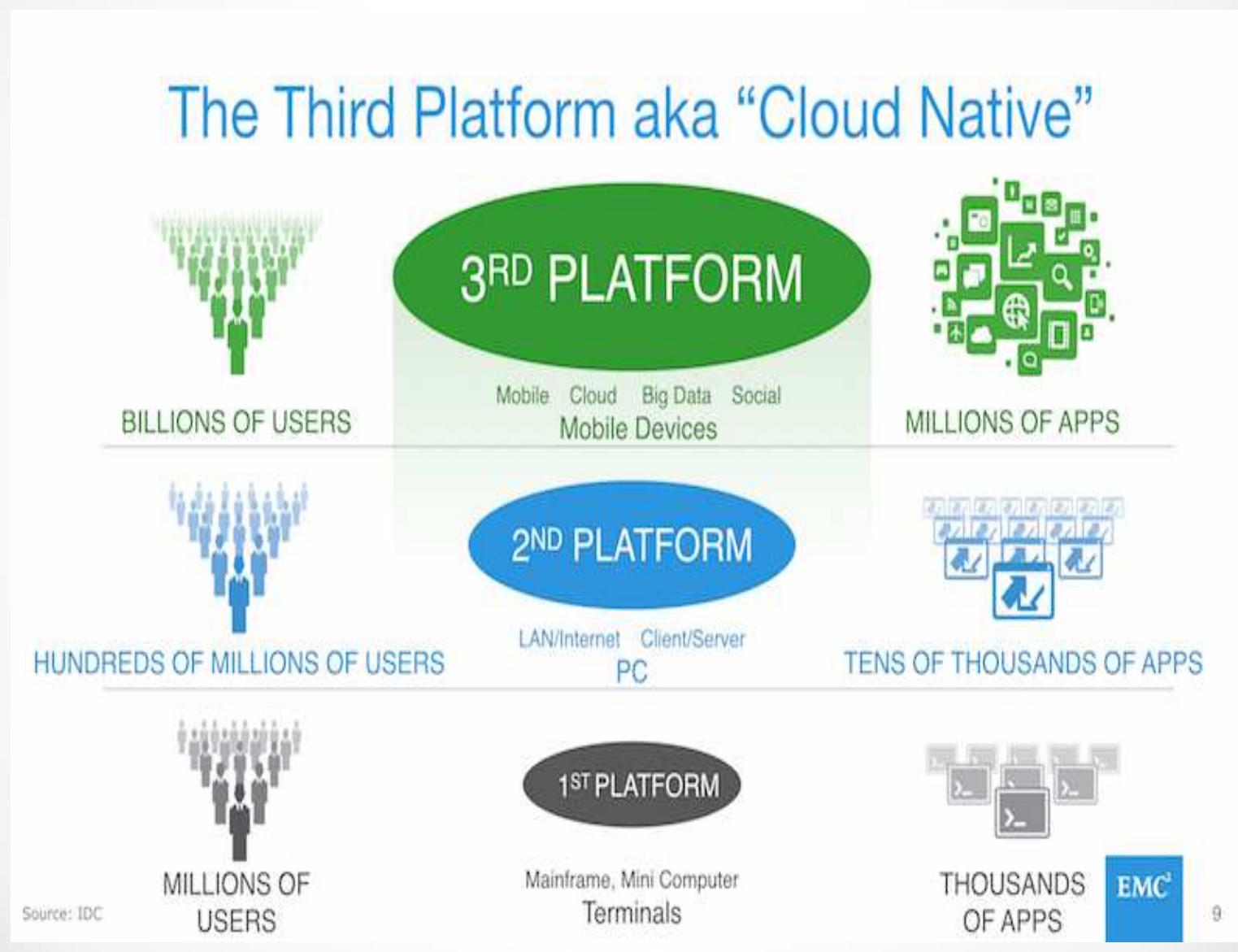
Evolution ou révolution

• • •

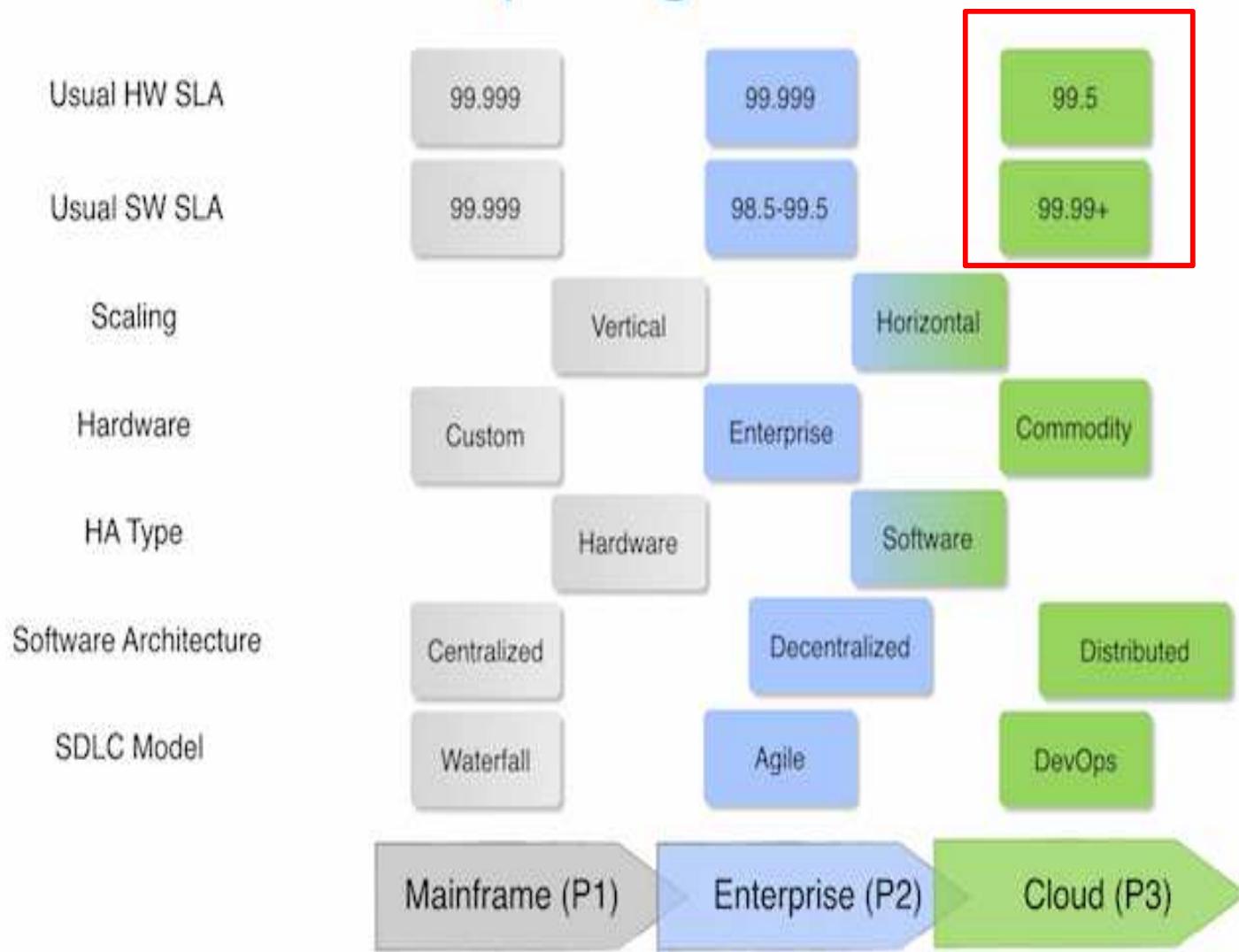
Fifth Generation of Computing



Les générations de plateformes selon Dell/EMC



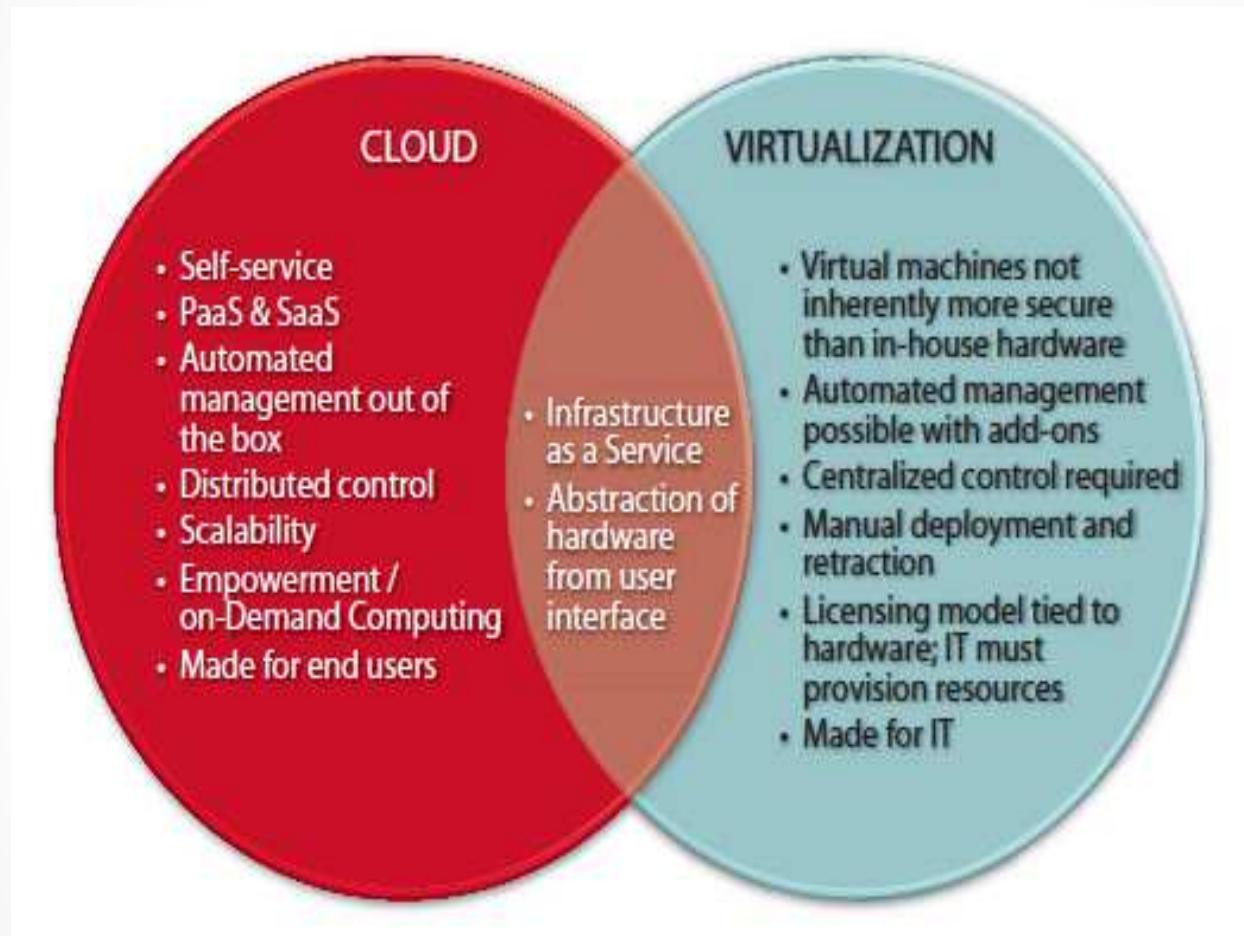
Cloud Computing Patterns Are Different



Techniquement

- Utilisation de la virtualisation
 - Catalogue de service conséquent
 - Infra pilotables « as code »
 - Container = isolation, donc techno ancienne, mais usages nouveaux.
 - Les applications conçues pour le cloud, pour profiter pleinement des « apports » du Cloud
-

Cloud vs Virtualisation



Pets versus Cattle

Scalabilité verticale ou horizontale



Scale Up

- Ils ont un « vrai » nom
- Ils sont uniques
- On en prend soin individuellement
- En cas de problème, on les soigne
- En cas de besoin plus important, on les booste!



Scale Out

- Ils ont un identifiant
- Ils sont identiques aux autres
- On gère le troupeau
- En cas de problème, on le change
- En cas de besoin plus important, on en rajoute

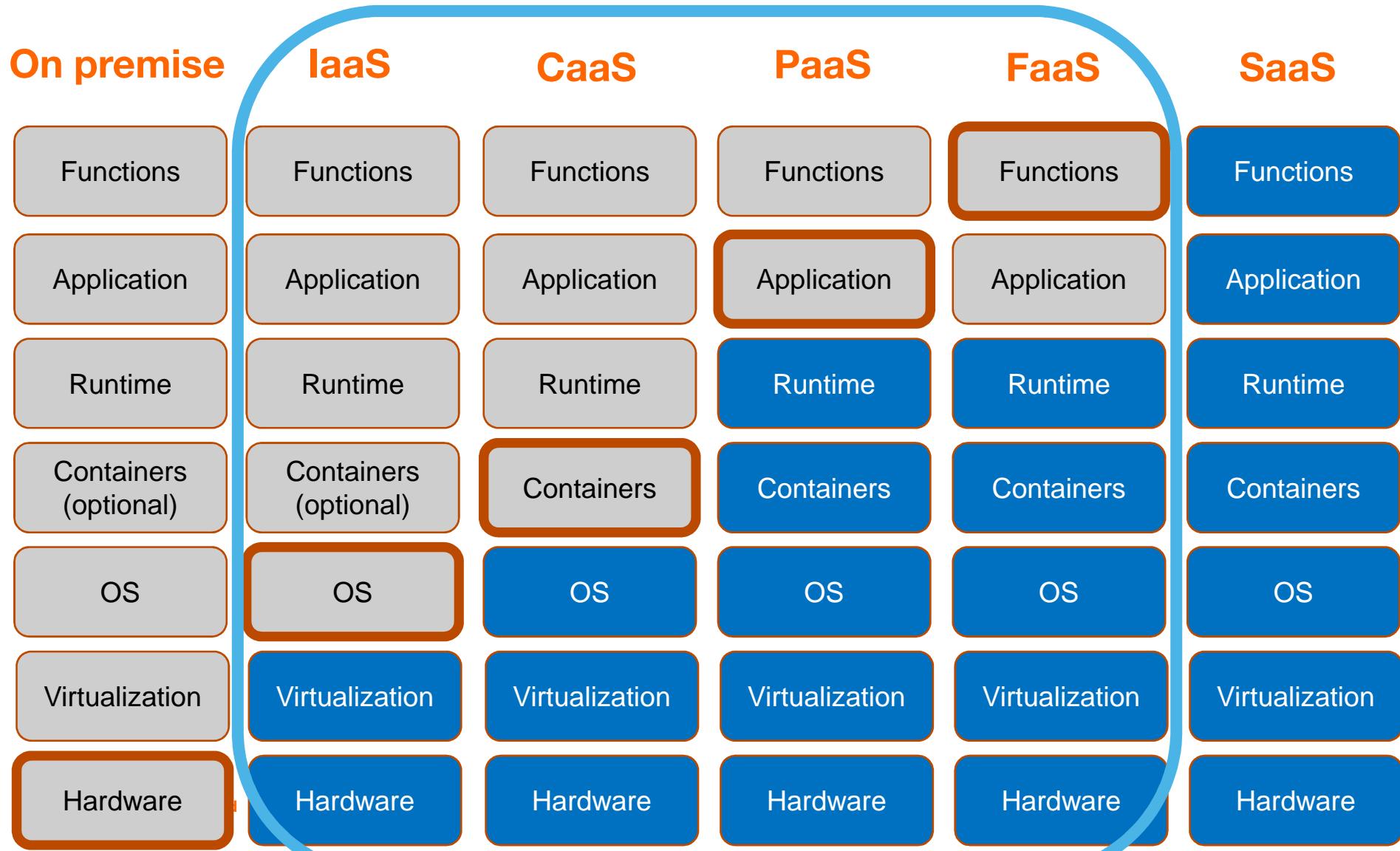
Modèle cible : Cattle



Séparation des responsabilités

Géré par
l'utilisateur

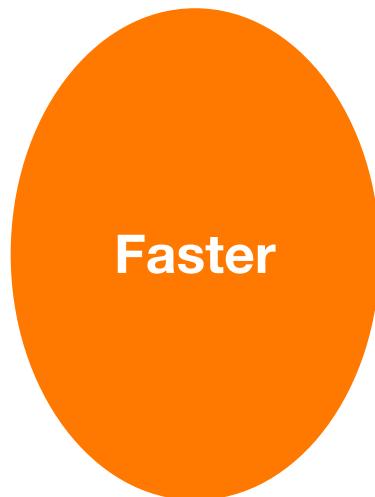
Géré par le
fournisseur



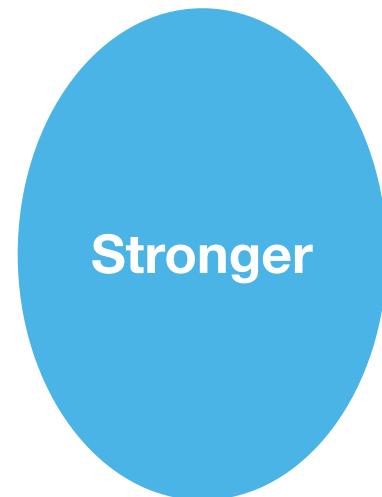
Les motivations pour aller vers le cloud



Développer plus vite
avec **plus de**
fonctionnalités pour
moins cher



Réduire le **temps**
d'atteinte du marché
pour rester dans la
compétition



Proposer une haute
qualité de service face
au marché



Aller vers la **culture de**
la mesure pour
s'améliorer en continu

Usages

- Consommation à l'usage, donc utiliser les ressources quand c'est nécessaire, et les économiser.
- Modèle OPEX : location des ressources et des services. Pas d'achats, d'investissements lourds, de gestion de capacité, et de décommissionnement.
- XaaS : Everything as a Service. Il faut utiliser les services proposés par le cloud (Load-Balancer, DBaaS, VPNaas, etc ...)
-

Organisation

- DevOps : équipe mixtes qui travaillent ensemble
- CI / CD : automatiser les tests d'intégration le plus possible, potentiellement jusqu'à la production.

•

Les enjeux

- **Vendor locking** : ne pas être trop fortement lié à un CSP
- **Shadow IT** : maîtriser ses inventaires (assets) sur les différents tenant d'une entreprise dans les clouds publiques. Ce n'est pas parce que des environnements ne sont pas hébergés sur les infras de l'entreprise qu'il ne faut pas les maîtriser.
- La sécurité ...

Cloud, les craintes, les risques

- Sécurité des données
- Pérennité du fournisseur
- Changement des processus
- Perte de contrôle des DSI
- Localisation des données
- Et problématique juridique
- Le Patriot Act
- Le Vendor Locking



- C'est avant tout, une relation de **confiance** avec son
 - **Cloud Service Provider**

Complément d'infos pratiques

- **Calculette AWS :**
http://calculator.s3.amazonaws.com/index.html?Ing=fr_FR#
- **Quick starts « bastion linux » :**
<https://aws.amazon.com/fr/quickstart/architecture/linux-bastion/>
- **Well architected platforms :**
<https://aws.amazon.com/fr/architecture/well-architected/>
- **Offre « Gratuite » AWS (pendant un an) :**
<https://aws.amazon.com/fr/free/>
- **TCO : Total Cost of Ownership.**
Couts de possession d'une infra, d'un service. (achat + maintenance + administration + ... tout ce qui permet de rendre le service)



Démos

• • •