

Systemes distribués et architectures cloud

Yves Denneulin

Savoir ce qu'est le cloud

aspects techniques et organisationnelles

connaître le paysage industriel et institutionnel

avoir du recul sur les solutions

Briques de base : containers et orchestration

DevOps et serverless

Sécurité et Souveraineté

Consommation énergétique

Une séance de CM suivi d'une séance de TP

Cloud computing is a delivery model for technology-enabled **services** that provides on-demand access via a network to an elastic pool of shared computing assets (e.g. services, applications, servers, storage, and networks) that can be rapidly provisioned and released with minimal service provider interaction. The entire value can be bi-directionally scaled as needed to enable pay-per-use.

Le cloud c'est juste l'ordinateur de quelqu'un d'autre

Idée aussi vieille que l'informatique moderne (time-sharing dans les 70s)

Essor du cloud commercial étroitement lié au développement d'Internet

Différentes phases

- 90s: Location de places dans un datacenter

- 1999 : Salesforce utilisable dans un navigateur (2004, IPO)

- début des années 2000 : Architecture de grilles (recherche)

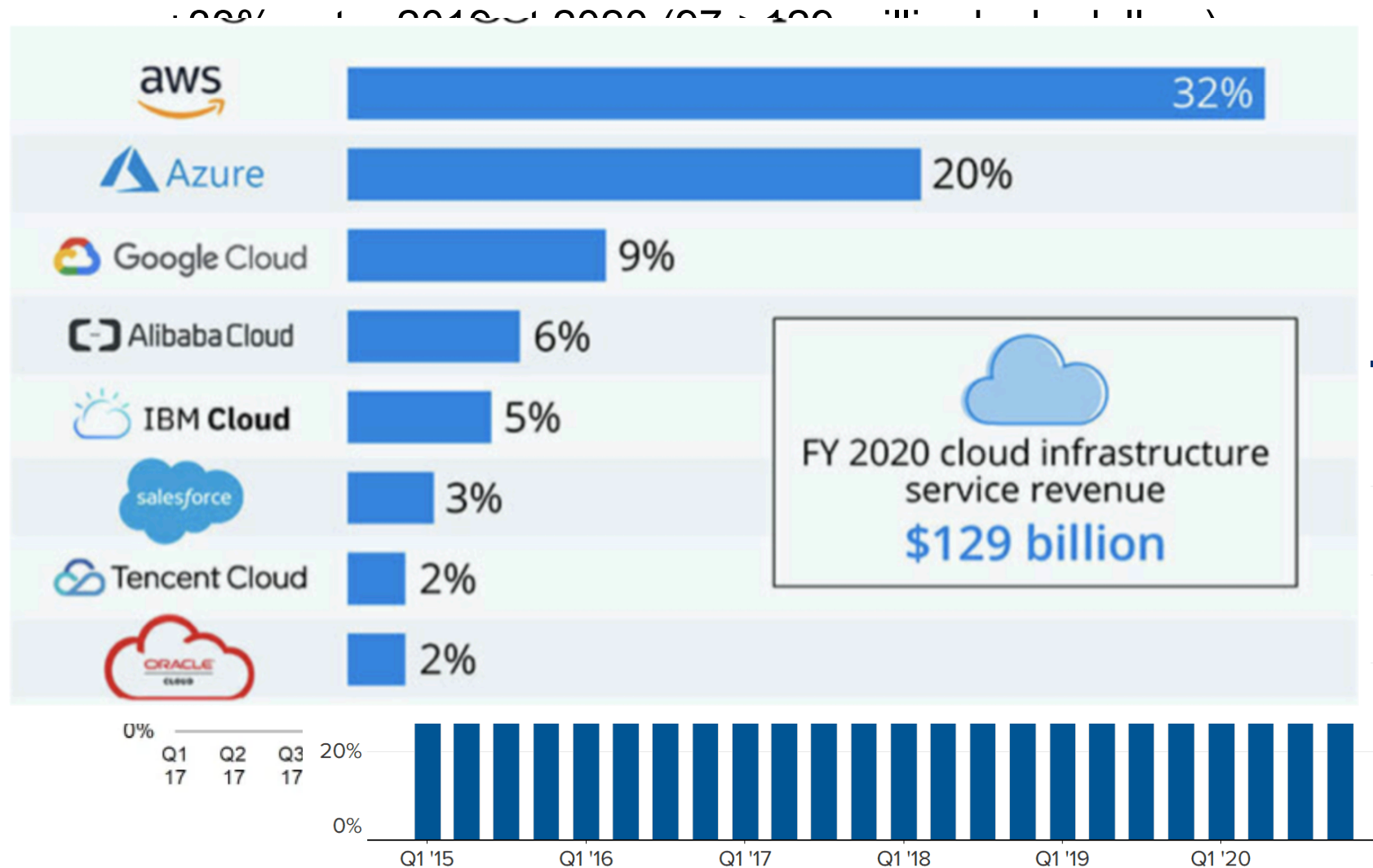
- 2002 : création de AWS

- 2006 : location de machines virtuelles sur AWS

- 2009 : services google aux entreprises par le web (gmail, goffice)

Maintenant solutions disponibles pour toutes les dimensions d'un système informatique

au départ à destination des start-ups, maintenant grand groupe aussi



SOURCE: Company reports





Au delà de la technique...

- Toute activité informatique est devenue un service
 - y compris l'achat d'une machine
 - importance des contrats liant clients et fournisseurs
- Les applications peuvent aussi s'exprimer sous la forme de services
- Les devices ne sont plus qu'une interface
- Le rendement énergétique augmente
 - car utilisation au plus juste
 - vrai pour les infrastructures de base, pas pour les piles!
- La complexité des piles logicielles est cachée (+/-)
 - baisse des compétences nécessaires pour s'en servir
- Le passage à l'échelle est techniquement très facile
 - nivelle le paysage
 - gros facteur de croissance des start-ups
 - méthode de développement et d'intégration continue

- Restructuration des organisations
 - cloud d'entreprise ou d'administration
 - réflexions au niveau des gouvernements
 - quel degré de mutualisation ?
 - quels facteurs de sécurité et de disponibilité ?
- Enjeu de la souveraineté
 - europe/USA/Chine
- Attention au Vendor lock-in
- Évolution très rapide, coût de mise à niveau à ne pas négliger
 - savoir chercher et exploiter les ressources en ligne
 - Linux règne en maître

- Agilité : passage à l'échelle transparent et rapide
- Protection des données
 - externalisation de la responsabilité
- Réduction des coûts (matériels, logiciels, humains, énergie)
- Recentrage sur le coeur de métier de l'entreprise
- Rapidité de déploiement
 - cloud accompagne le développement agile : devops
 - évolution vers le Serverless
- Pour les informaticiens, là où beaucoup de choses se passent (construction et utilisation)

Exemples de fonctionnalités fournies par un cloud

- Services transparents aux utilisateurs
 - Siri, Alexa, Google Home
- Utilisation de logiciels
 - Office365
- Utilisation de services logiciels
 - bibliothèque de Machine Learning, chatGPT
- Espace de stockage
 - DropBox, onecloud, gitlab
- Hébergement de ressources physiques explicites
 - Machine virtuelle, machine physique

rôle
central de
la notion
de service

Machine Learning

- Amazon SageMaker
- Amazon Augmented AI
- ★ Amazon CodeGuru
- Amazon Comprehend Recommandation et l'exécution d'
- Amazon Forecast
- Amazon Fraud Detector
- Amazon Kendra
- Amazon Lex
- Amazon Personalize
- Amazon Polly
- Amazon Rekognition
- Amazon Textract
- Amazon Transcribe
- Amazon Translate
- AWS DeepComposer
- AWS DeepLens
- AWS DeepRacer

Quelle pourcentage
d'activités faites vous
en local vs cloud ?

Et vos proches ?

Quelle évolution en 10
ans ?

Paiement à l'usage ou au forfait

Organisationnel : *commoditisation* des ressources matérielles et logicielles

- flexibilité accrue des configurations et des solutions

- prototypage plus rapide

- évolution des compétences nécessaires pour développer et faire tourner un système complet

Financier : passage du modèle de l'investissement amortissable au modèle du service

- côté clients : lissage, montée en puissance automatique

- côté vendeurs : pour chaque client, prévisionnel plus facile à faire

Privé : tout est géré par l'entité

déploiement et services fournis au reste de l'entreprise

Communautaire : application métier privée partagée entre plusieurs tiers

Public : ressources fournies par un tiers (*hyper-scalers*)

dans un environnement partagé

éventuellement avec des ressources dédiées

avec une politique de sécurité propre, des données propres

Hybride : combinaison des deux

exemple : une infrastructure en propre faisant appel à des services externes

De facto, tout devient hybride maintenant

Différentes catégories de services

IaaS : infrastructure (matériel : machines + stockage + réseau)

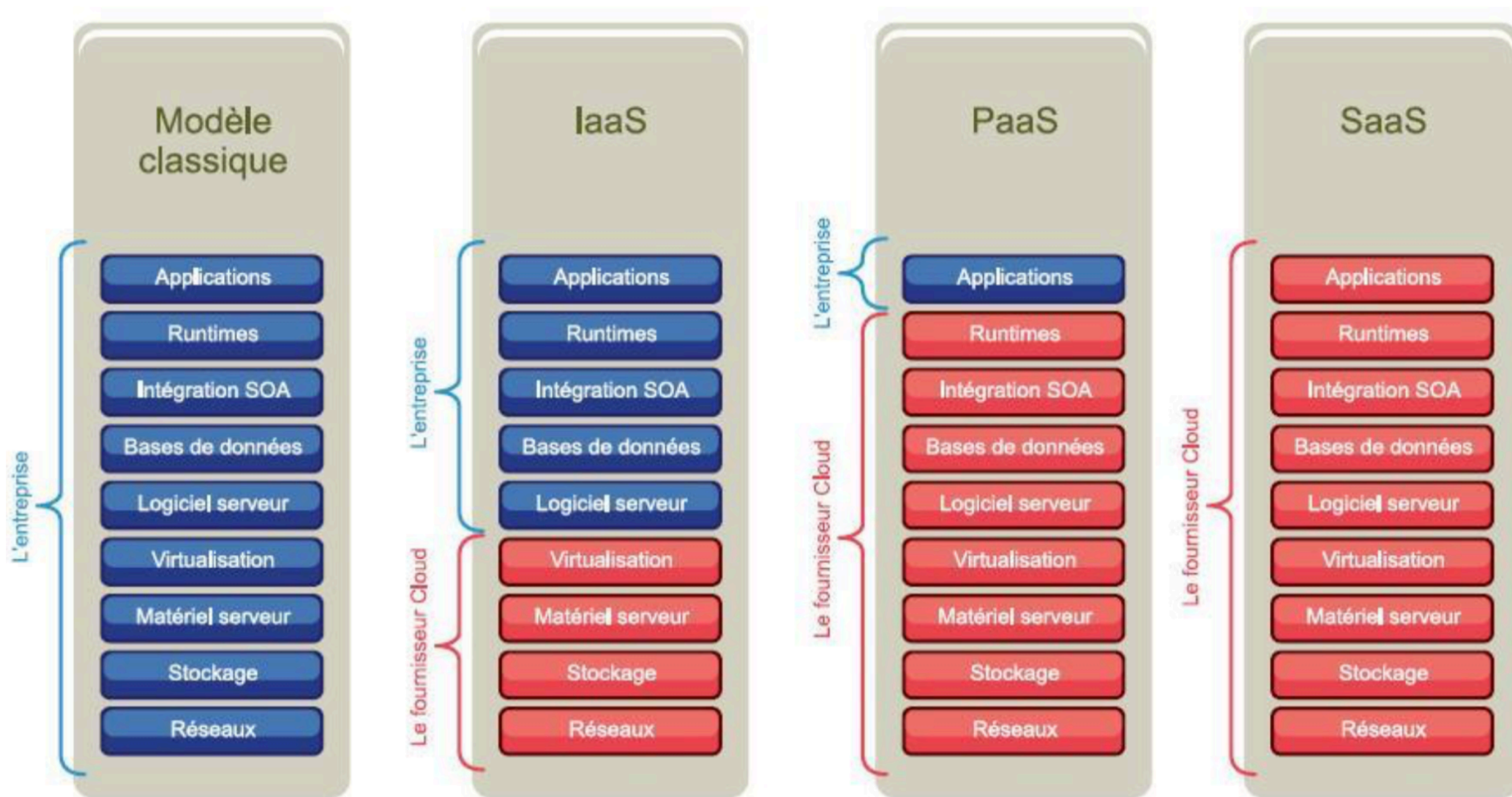
CaaS : Container as a Service (~machine virtuelle)

PaaS : plateforme (piles logicielles complexes)

SaaS : services (aux utilisateurs ou briques logicielles)

Point clé : standardisation des protocoles de communication et des piles logicielles (open source)

Source : SYNTEC



Traditionnelle

application bien connue, stable dans son comportement
tourne pendant des années sans modification
dimensionnement effectué lors du déploiement
stable dans le temps, charge homogène à supporter

Conçue pour le cloud

Distribuée

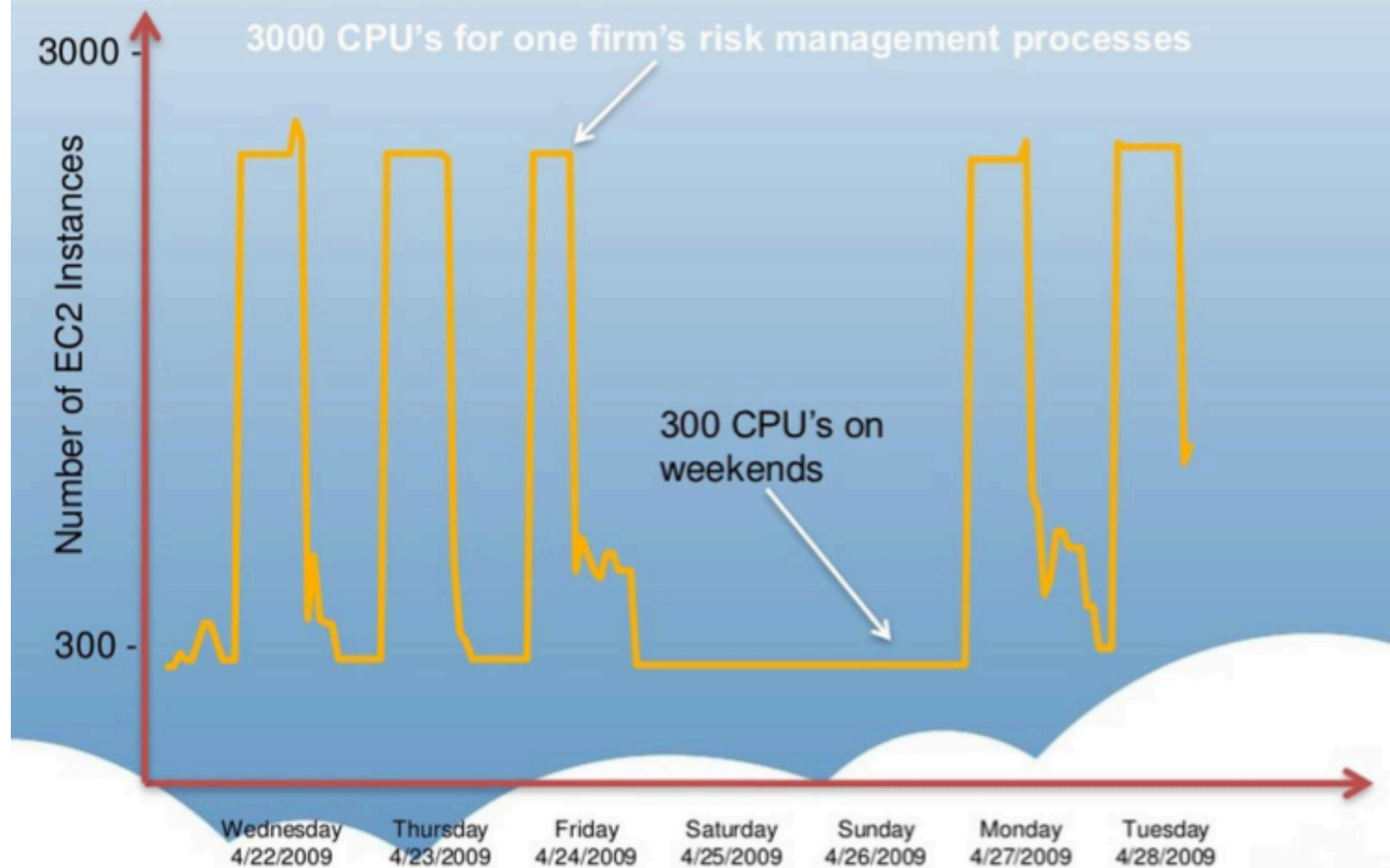
Hautement modulaire : invocation de fonctions ou de domaine fonctionnel

s'appuie sur des *frameworks* (piles logicielles) évolutifs
cycle de vie à la semaine ou au mois

peut avoir à supporter des workloads hétérogènes

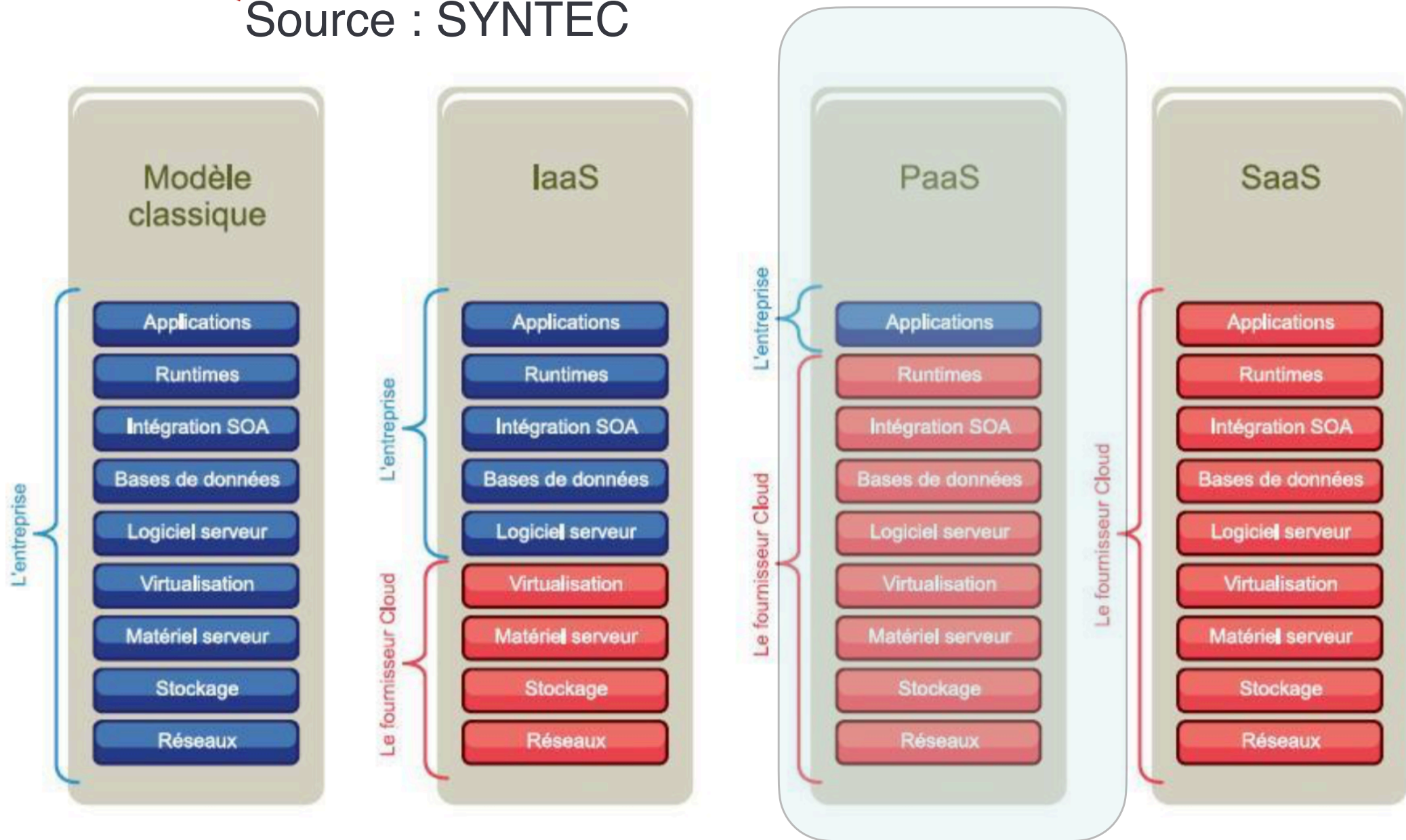
Big Data & IA : passage à l'échelle rapide implique cloud

Example: Wall Street App on Amazon EC2

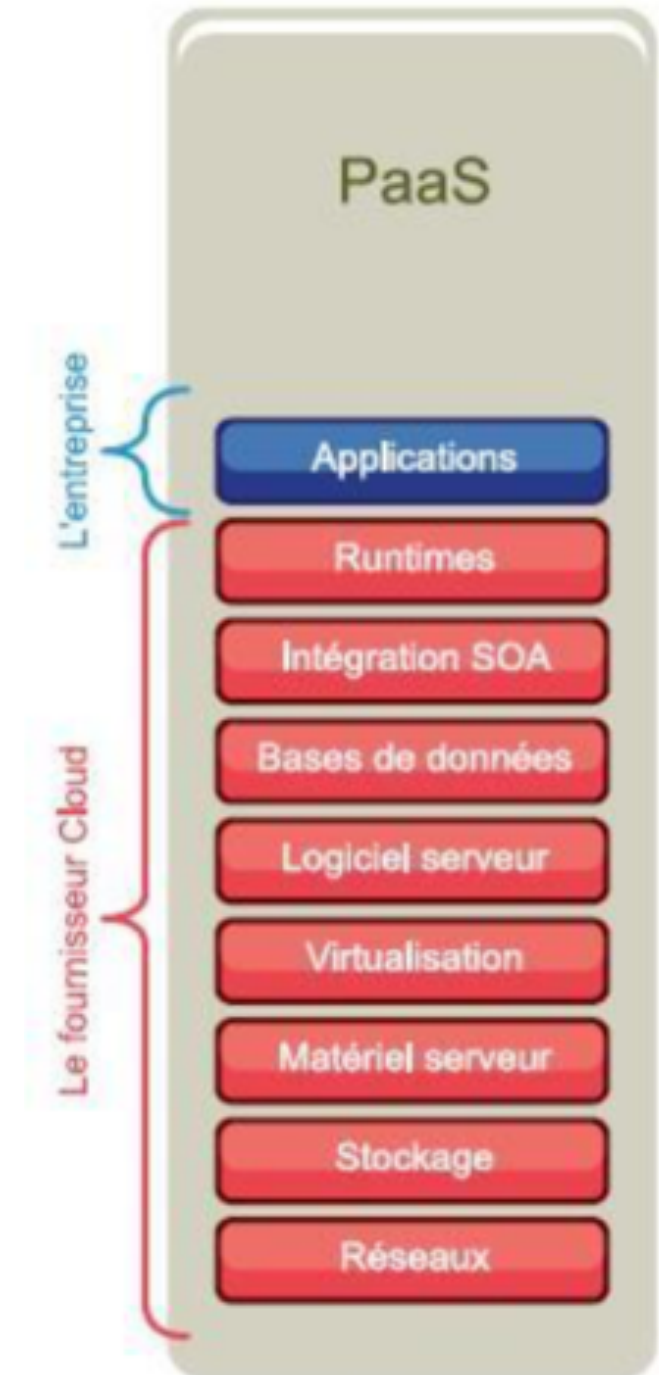


Vision schématique

Source : SYNTEC



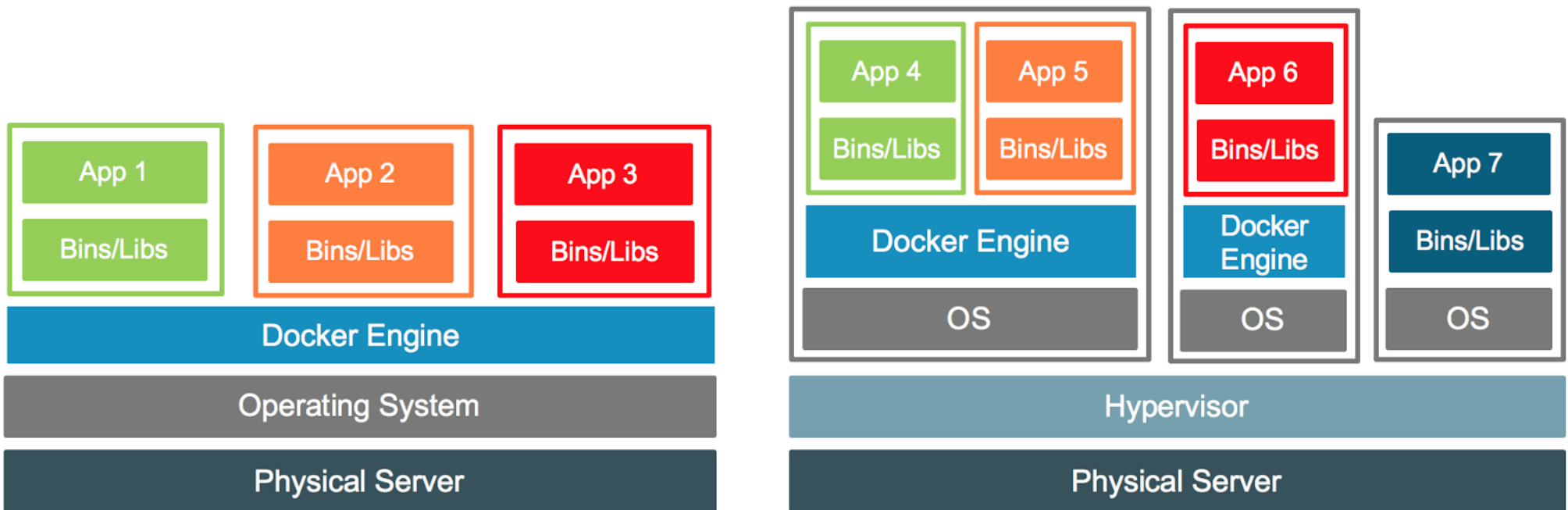
- Platform as a Service
 - le niveau le plus utilisé
- Fournit une pile complète
 - Préinstallée
 - configurée
- Services : déploiement et exécution d'une application
 - sous quelle forme ?



Idée : distribuer une application avec son environnement

Passage de l'IaaS au PaaS

Your Datacenter or VPC



Solution de Container la plus répandue

Composée de

- un constructeur d'image de containers

- un gestionnaire d'exécution des containers

- une interface REST

 - permet une automatisation de la gestion des containers

- un repository de containers

 - officiel mais possibilité d'en créer des locaux

```
$ cat Dockerfile
#This is a sample Image
FROM ubuntu
MAINTAINER poftut@gmail.com

RUN apt-get update
RUN apt-get install -y apache2
```

<https://www.poftut.com/how-to-use-dockerfile-to-build-images-for-docker-containers/>

Permet de normaliser et figer un environnement de développement et d'exécution

on ne conserve un container que quand toutes les versions des composants qu'il contient fonctionnent ensemble

Efficacité de gestion

déploiement en moins d'une seconde

à partir d'une image (souvent en cache) qui peut venir d'un repository

S'appuie sur le noyau (et les librairies systèmes) de l'hôte et sur le support de container qui doit y être installé



Création et exécution d'un container

Dockerfile :

```
FROM debian  
RUN apt-get update
```

```
CMD [ « bash », «-c », ...]
```

exécuté au moment de la
création de l'image

exécuté au moment du
démarrage du container

dans le répertoire où se trouve ce fichier Dockerfile

```
docker build . -t mon_container
```

crée une image appelée mon_container

puis

```
docker run mon_container
```

l'exécute avec des options éventuelles (—port ...)

Une image de container peut être stocké sous forme de source -> une liste d'éléments à récupérer et assembler et des instructions à exécuter

léger, permet de figer une configuration qui fonctionne binaire (image) -> une archive déployable immédiatement

Une image docker est un instantané d'un système de fichiers avec quelques indications supplémentaires

Un container en cours d'exécution est
un file system (écriture=copy-on-write)
une pile IP (avec une adresse propre)
un process group
qui exécute une image de container
par défaut non interactif

Les containers partagent beaucoup de ressources (dont le noyau)

- + : faible usage de la mémoire

- : plus de contention potentielle (passage à l'échelle)

Containers plus utilisés pour faire de gros volumes d'I/O et de communication que de calcul

le recouvrement calcul/communication limite l'impact de la contention

Partage de mémoire possible entre containers pour des ressources partagées (sécurité ?)

Les containers peuvent rapidement prendre beaucoup d'espace disque
système de couches empilées

Les containers sont des boites noires qui ont vocation à rendre des services
-> ils doivent pouvoir communiquer avec l'extérieur

Un container Docker au démarrage a :
une interface avec adresse IP routable (si l'hôte est connecté)
un routeur
un DNS

Il est sur un réseau local à la machine et partagé entre les containers

Il peut se connecter à l'extérieur mais...

pas moyen de s'y connecter depuis l'extérieur du réseau des containers

Export de port vers la machine hôte possible

Plus d'infos sur : <https://docs.docker.com/engine/network/>



Open Container Interface

Organisation visant à la standardisation des containers (interopérabilité!)

« an open governance structure for the express purpose of creating open industry standards around container formats and runtimes »

Définit trois spécifications

environnement d'exécution : “Specifies the configuration, execution environment, and lifecycle of a container.” In particular, “defines how to properly run a container "filesystem bundle" which fully adheres to the OCI Image Format Specification.”

format des images : Defines the requirements for an OCI Image (container image), which consists of a manifest, an optional image index, a set of filesystem layers, and a configuration

forme de distribution : Defines an API protocol to facilitate and standardize the distribution of content

Repository github avec l'ensemble des éléments de spécification et des implémentations

<https://opencontainers.org/>

Exécutable en IAAS

il existe des instances avec Docker

Soumission directe de Containers

Amazon Elastic Container Service

envois de containers à exécuter

facturation à la seconde + CPU et mémoire

idéal pour : application longue et ponctuelle, build

applications self contained

Azure Container Instances

« serverless containers »

MS Flow pour construire une application en utilisant des containers
comme blocs