CLOUD

CONCERNANT CES SUPPORTS DE COURS

SUPPORTS DE COURS RÉALISÉS PAR OSONES

https://osones.com

Logo Osones

- Sources : https://github.com/Osones/formations/
- HTML/PDF : https://osones.com/formations/

Licence Creative Commons BY-SA 4.0

INTRODUCTION

OBJECTIFS DE LA FORMATION : CLOUD

- Comprendre les principes du cloud et son intérêt
- Connaitre le vocabulaire inhérent au cloud
- Avoir une vue d'ensemble sur les solutions existantes en cloud public et privé
- Posséder les clés pour tirer parti au mieux du IaaS
- Pouvoir déterminer ce qui est compatible avec la philosophie cloud ou pas
- Adapter ses méthodes d'administration système et de développement à un environnement cloud

LE CLOUD, VUE D'ENSEMBLE

DÉFINITION FORMELLE

CARACTÉRISTIQUES

Fournir un (des) service(s)...

- Self service
- À travers le réseau
- Mutualisation des ressources
- Élasticité rapide
- Mesurabilité

Inspiré de la définition du NIST
https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublicati 145.pdf

SELF SERVICE

- L'utilisateur accède *directement* au service
- Pas d'intermédiaire humain
- Réponses immédiates
- Catalogue de services permettant leur découverte

À TRAVERS LE RÉSEAU

- L'utilisateur accède au service à travers le réseau
- Le *fournisseur* du service est distant du *consommateur*
- Réseau = internet ou pas
- Utilisation de protocoles réseaux standards (typiquement : HTTP)

MUTUALISATION DES RESSOURCES

- Un cloud propose ses services à de multiples utilisateurs/organisations (*multi-tenant*)
- *Tenant* ou *projet* : isolation logique des ressources
- Les ressources sont disponibles en grandes quantités (considérées illimitées)
- Le taux d'occupation du cloud n'est pas visible
- La localisation précise des ressources n'est pas visible

ÉLASTICITÉ RAPIDE

- Provisionning et suppression des ressources quasi instantané
- Permet le *scaling* (passage à l'échelle)
- Possibilité d'automatiser ces actions de *scaling*
- Virtuellement pas de limite à cette élasticité

MESURABILITÉ

- L'utilisation des ressources cloud est monitorée par le fournisseur
- Le fournisseur peut gérer son *capacity planning* et sa facturation à partir de ces informations
- L'utilisateur est ainsi facturé en fonction de son usage précis des ressources
- L'utilisateur peut tirer parti de ces informations

MODÈLES

On distingue :

- modèles de service : IaaS, PaaS, SaaS
- modèles de déploiement : public, privé, hybride

IAAS

- *Infrastructure as a Service*
- Infrastructure :
- Compute (calcul)
- Storage (stockage)
- Network (réseau)
- Utilisateurs cibles : administrateurs (système, stockage, réseau)

PAAS

- *Platform as a Service*
- Désigne deux concepts :
- Environnement permettant de développer/déployer une application (spécifique à un langage/framework - exemple : Python/Django)
- Ressources plus haut niveau que l'infrastructure, exemple : BDD
- Utilisateurs cibles : développeurs d'application

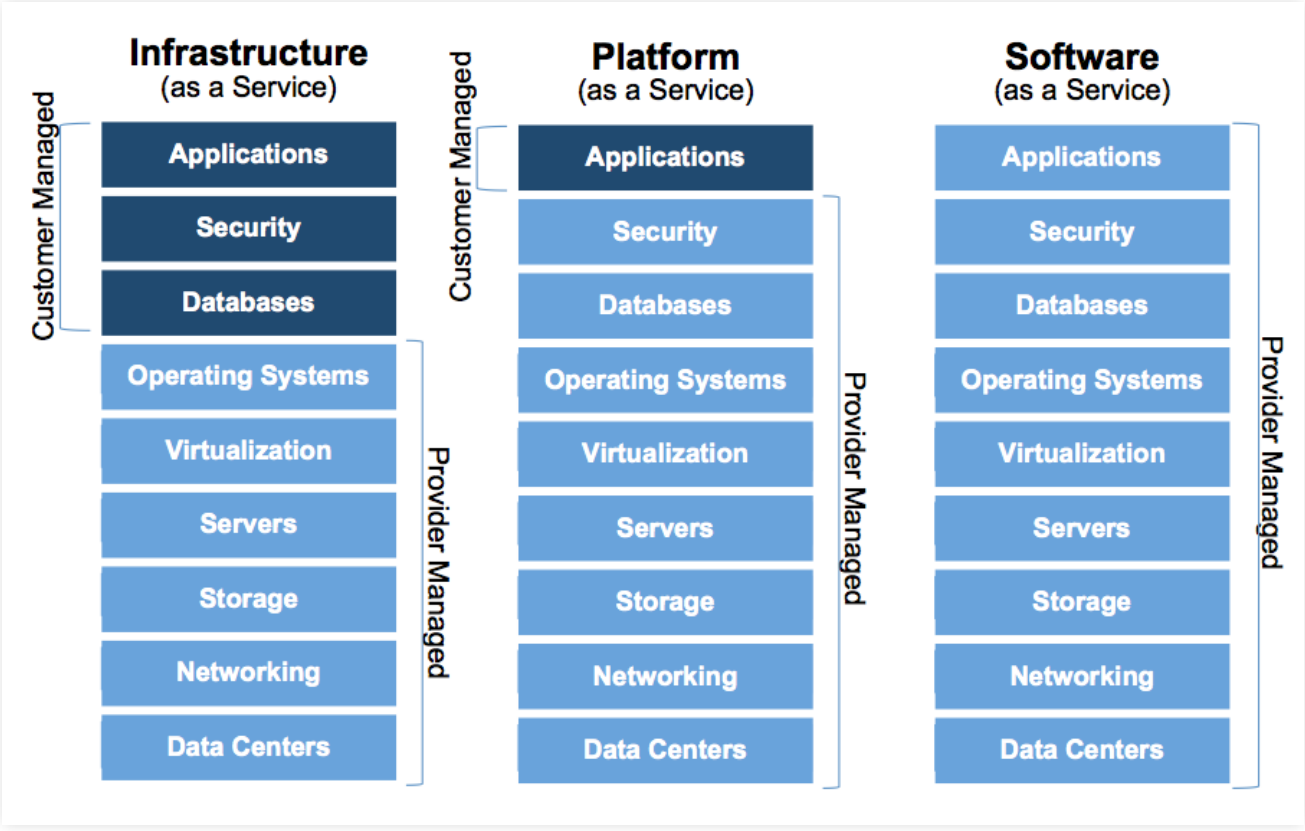
SAAS

- *Software as a Service*
- Utilisateurs cibles : utilisateurs finaux
- Ne pas confondre avec la définition *économique* du SaaS

QUELQUECHOSE AS A SERVICE ?

- Load balancing as a Service (Infra)
- Database as a Service (Platform)
- MonApplication as a Service (Software)
- etc.

LES MODÈLES DE SERVICE EN UN SCHÉMA



IaaS - PaaS - SaaS (source : Wikipedia)

CLOUD PUBLIC OU PRIVÉ ?

À qui s'adresse le cloud ?

- Public : tout le monde, disponible sur internet
- Privé : à une organisation, disponible sur son réseau

CLOUD HYBRIDE

- Utilisation mixte de multiples clouds privés et/ou publics
- Concept séduisant mais mise en œuvre a priori difficile
- Certains cas d'usages s'y prêtent très bien
- Intégration continue (CI)
- Motivations
- Éviter le *lock-in*
- Débordement (*cloud bursting*)

L'INSTANT VIRTUALISATION

Mise au point.

- La virtualisation est une technologie permettant d'implémenter la fonction *compute*
- Un cloud fournissant du compute *peut* utiliser la virtualisation
- Mais peut également utiliser :
- Du bare-metal
- Des containers (système)

LES APIS, LA CLÉ DU CLOUD

- Rappel : API pour *Application Programming Interface*
- Au sens logiciel : Interface permettant à un logiciel d'utiliser une bibliothèque
- Au sens cloud : Interface permettant à un logiciel d'utiliser un service (XaaS)
- Interface de programmation (via le réseau, souvent HTTP)
- Frontière explicite entre le fournisseur (provider) et l'utilisateur (user)
- Définit la manière dont l'utilisateur communique avec le cloud pour gérer ses ressources
- Gérer : CRUD (Create, Read, Update, Delete)

API REST

- Une ressource == une URI (*Uniform Resource Identifier*)
- Utilisation des verbes HTTP pour caractériser les opérations (CRUD)
- GET
- POST
- PUT
- DELETE
- Utilisation des codes de retour HTTP
- Représentation des ressources dans le corps des réponses HTTP

REST - EXEMPLES

```
GET http://endpoint/volumes/
GET http://endpoint/volumes/?size=10
POST http://endpoint/volumes/
DELETE http://endpoint/volumes/xyz
```

EXEMPLE CONCRET

GET /v2.0/networks/d32019d3-bc6e-4319-9c1d-6722fc136a22


```
{
 "network":{
  "status":"ACTIVE",
  "subnets":[ "54d6f61d-db07-451c-9ab3-b9609b6b6f0b" ],
  "name":"private-network",
  "provider:physical_network":null,
  "admin_state_up":true,
  "tenant_id":"4fd44f30292945e481c7b8a0c8908869",
  "provider:network_type":"local",
  "router:external":true,
  "shared":true,
  "id":"d32019d3-bc6e-4319-9c1d-6722fc136a22",
  "provider:segmentation_id":null
 }
}
```

POURQUOI LE CLOUD ? CÔTÉ ÉCONOMIQUE

- Appréhender les ressources IT comme des services “fournisseur”
- Faire glisser le budget “investissement” (Capex) vers le budget “fonctionnement” (Opex)
- Réduire les coûts en mutualisant les ressources, et éventuellement avec des économies d'échelle
- Réduire les délais
- Aligner les coûts sur la consommation réelle des ressources

POURQUOI LE CLOUD ? CÔTÉ TECHNIQUE

- Abstraire les couches basses (serveur, réseau, OS, stockage)
- S'affranchir de l'administration technique des ressources et services (BDD, pare-feux, load-balancing, etc.)
- Concevoir des infrastructures scalables à la volée
- Agir sur les ressources via des lignes de code et gérer les infrastructures "comme du code"

LE MARCHÉ

AMAZON WEB SERVICES (AWS), LE LEADER

AWS logo

- Lancement en 2006
- À l'origine : services web "e-commerce" pour développeurs
- Puis : d'autres services pour développeurs
- Et enfin : services d'infrastructure
- Récemment, SaaS

ALTERNATIVES IAAS PUBLICS À AWS

- **Google Cloud Platform**
- **Microsoft Azure**
- Rackspace
- DreamHost
- DigitalOcean
- En France :
- Cloudwatt (Orange Business Services)
- Numergy (SFR)
- OVH
- Ikoula
- Scaleway
- Outscale

FAIRE DU IAAS PRIVÉ

- **OpenStack**
- CloudStack
- Eucalyptus
- OpenNebula

OPENSTACK EN QUELQUES MOTS

OpenStack logo

- Naissance en 2010
- Fondation OpenStack depuis 2012
- Écrit en Python et distribué sous licence Apache 2.0
- Soutien très large de l'industrie et contributions variées

EXEMPLES DE PAAS PUBLIC

- Amazon Elastic Beanstalk (https://aws.amazon.com/fr/elasticbeanstalk)
- Google App Engine (https://cloud.google.com/appengine)
- Heroku (https://www.heroku.com)

SOLUTIONS DE PAAS PRIVÉ

- Cloud Foundry, Fondation (https://www.cloudfoundry.org)
- OpenShift, Red Hat (https://www.openshift.org)
- Solum, OpenStack (https://wiki.openstack.org/wiki/Solum)

LES CONCEPTS INFRASTRUCTURE AS A SERVICE

LA BASE

- Infrastructure :
- Compute
- Storage
- Network

RESSOURCES *COMPUTE*

- Instance
- Image
- Flavor (gabarit)
- Paire de clé (SSH)

INSTANCE

- Dédiée au compute
- Durée de vie typiquement courte, à considérer comme éphémère
- Ne doit pas stocker de données persistantes
- Disque racine non persistant
- Basée sur une image

IMAGE CLOUD

- Image disque contenant un OS déjà installé
- Instanciable à l'infini
- Sachant parler à l'API de metadata

API ... DE METADATA

- http://169.254.169.254
- Accessible depuis l'instance
- Fournit des informations relatives à l'instance
- Expose les *userdata*
- L'outil cloud-init permet d'exploiter cette API

FLAVOR (GABARIT)

- *Instance type* chez AWS
- Définit un modèle d'instance en termes de CPU, RAM, disque (racine), disque éphémère
- Le disque éphémère a, comme le disque racine, l'avantage d'être souvent local donc rapide

PAIRE DE CLÉ

- Clé publique + clé privée SSH
- Le cloud manipule et stocke la clé publique
- Cette clé publique est utilisée pour donner un accès SSH aux instances

RESSOURCES RÉSEAU 1/2

- Réseau L2
- Port réseau
- Réseau L3
- Routeur
- IP flottante
- Groupe de sécurité

RESSOURCES RÉSEAU 2/2

- Load Balancing as a Service
- VPN as a Service
- Firewall as a Service

RESSOURCES STOCKAGE

Le cloud fournit deux types de stockage

- Block
- Objet

STOCKAGE BLOCK

- **Volumes** attachables à une instance
- Accès à des raw devices type */dev/vdb*
- Possibilité d'utiliser n'importe quel système de fichiers
- Possibilité d'utiliser du LVM, du chiffrement, etc.
- Compatible avec toutes les applications existantes
- Nécessite de *provisionner* l'espace en définissant la taille du volume

DU STOCKAGE PARTAGÉ ?

- Le stockage block n'est **pas** une solution de stockage partagé comme NFS
- NFS se situe à une couche plus haute : système de fichiers
- Un volume est *a priori* connecté à une seule machine

"BOOT FROM VOLUME"

Démarrer une instance avec un disque racine sur un **volume**

- Persistance des données du disque racine
- Se rapproche du serveur classique

STOCKAGE OBJET

- API : faire du CRUD sur les données
- Pousser et retirer des **objets** dans un **container**/bucket
- Pas de hiérarchie, pas de répertoires, pas de système de fichiers
- Accès lecture/écriture uniquement par les APIs
- Pas de *provisioning* nécessaire
- L'application doit être conçue pour tirer parti du stockage objet

ORCHESTRATION

- Orchestrer la création et la gestion des ressources dans le cloud
- Définition de l'architecture dans un **template**
- Les ressources créées à partir du **template** forment la **stack**
- Il existe également des *outils* d'orchestration (plutôt que des *services*)

BONNES PRATIQUES D'UTILISATION

POURQUOI DES BONNES PRATIQUES ?

Deux approches :

- Ne pas évoluer
- Risquer de ne pas répondre aux attentes
- Se contenter d'un cas d'usage *test & dev*
- Adapter ses pratiques au cloud pour en tirer parti pleinement

HAUTE DISPONIBILITÉ (HA)

- Le control plane (les APIs) du cloud est HA
- Les ressources provisionnées ne le sont pas forcément

PET VS CATTLE

Comment considérer ses instances ?

- Pet
- Cattle

INFRASTRUCTURE AS CODE

Avec du code

- Provisionner les ressources d'infrastructure
- Configurer les dites ressources, notamment les instances

Le métier évolue : Infrastructure Developer

SCALING, PASSAGE À L'ÉCHELLE

- Scale out plutôt que Scale up
- Scale out : passage à l'échelle horizontal
- Scale up : passage à l'échelle vertical
- Auto-scaling
- Géré par le cloud
- Géré par un composant extérieur

APPLICATIONS CLOUD READY

- Stockent leurs données au bon endroit
- Sont architecturées pour tolérer les pannes
- Etc.

Cf. https://12factor.net/

DERRIÈRE LE CLOUD

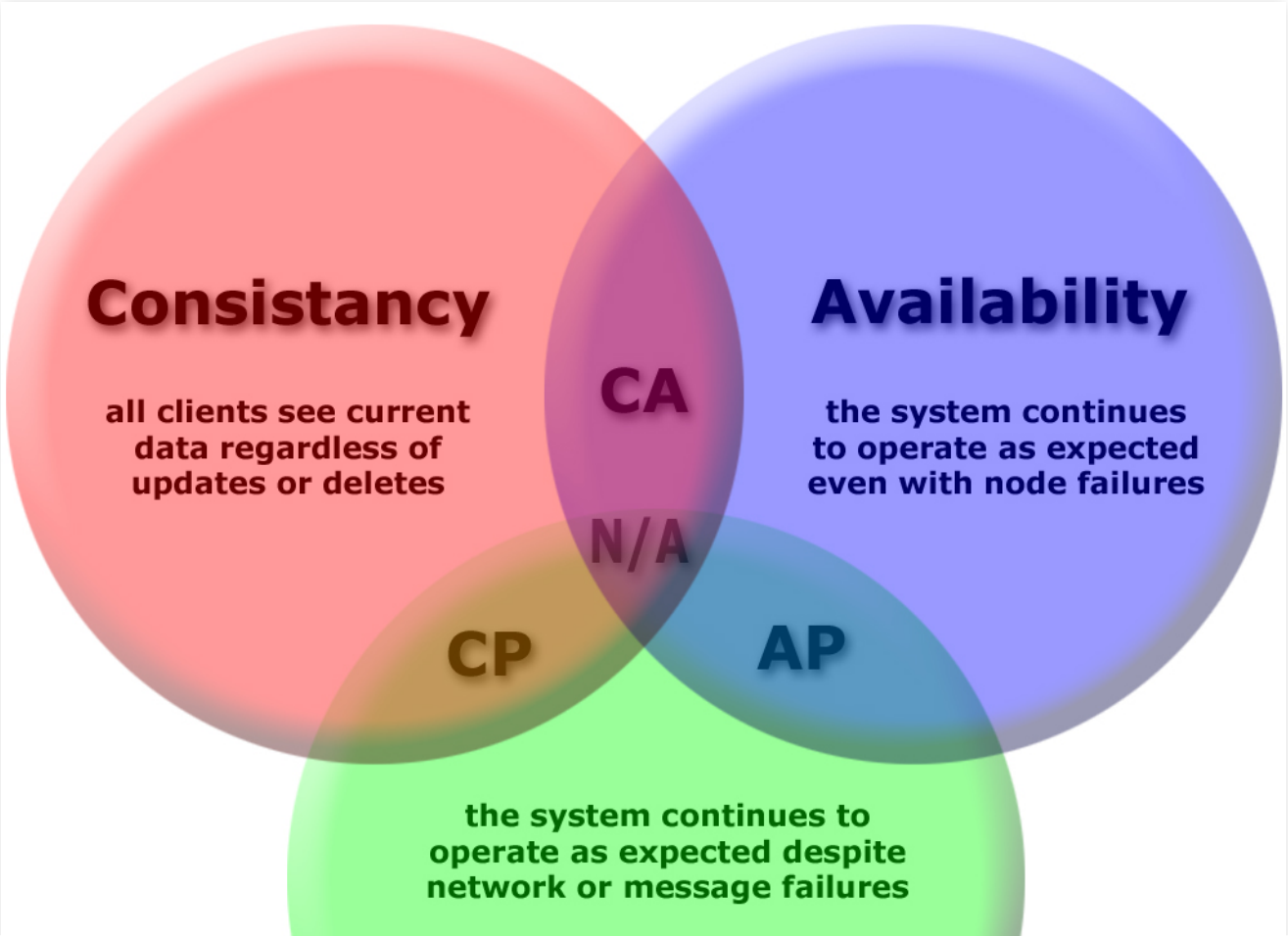
COMMENT IMPLÉMENTER UN SERVICE DE COMPUTE

- Virtualisation
- Containers (système)
- Bare metal

IMPLÉMENTATION DU STOCKAGE : (SOFTWARE DEFINED STORAGE) SDS

- **Attention** : ne pas confondre avec le sujet block vs objet
- Utilisation de commodity hardware
- Pas de RAID matériel
- Le logiciel est responsable de garantir les données
- Les pannes matérielles sont prises en compte et gérées
- Le projet **Ceph** et le composant **OpenStack Swift** implémentent du SDS
- Voir aussi **Scality**

SDS - THÉORÈME CAP



Consistency - Availability - Partition tolerance