

Formation Virtualisation / Cloud

Arnaud Morin



Concernant ces supports de cours

Auteurs initiaux :

- Adrien Cunin et Pierre Freund 
- Arnaud Morin <arnaud.morin@gmail.com>  OVH.com
Innovation is Freedom

Licence

- Licence : Creative Commons BY-SA 4.0 
<https://creativecommons.org/licenses/by-sa/4.0/deed.fr>

Sources

- <https://github.com/Osones/OpenStack-Formations/>
- <https://github.com/arnaudmorin/OpenStack-Formations/>

Objectifs de la formation

- Virtualisation
- Cloud
- Focus sur OpenStack

Objectifs de la formation : Virtualisation

- Comprendre les principes de la virtualisation et son intérêt
- Connaitre le vocabulaire inhérent à la virtualisation
- Avoir une vue d'ensemble sur les solutions existantes de virtualisation

Objectifs de la formation : Cloud

- Comprendre les principes du cloud et son intérêt
- Connaître le vocabulaire inhérent au cloud
- Avoir une vue d'ensemble sur les solutions existantes en cloud public et privé
- Pouvoir déterminer ce qui est compatible avec la philosophie cloud ou pas
- Adapter ses méthodes d'administration système à un environnement cloud (TP)

Objectifs de la formation : OpenStack

- Overview de ce qu'est OpenStack
- Avoir les bases pour utiliser OpenStack
- Déployer une machine virtuelle dans le cloud
- L'administrer et la configurer avec Ansible

Plan

- 1 Virtualisation
- 2 Le Cloud : vue d'ensemble
- 3 OpenStack : projet, logiciel et utilisation

Plan

- 1 Virtualisation
 - Histoire de la virtualisation
 - Principes généraux
 - Comprendre la virtualisation
- 2 Le Cloud : vue d'ensemble
- 3 OpenStack : projet, logiciel et utilisation

Définition

Selon Wikipedia :

« *La virtualisation consiste à faire fonctionner un ou plusieurs systèmes d'exploitation comme un simple logiciel, sur un ou plusieurs ordinateurs (serveurs), au lieu de ne pouvoir en installer qu'un seul par machine.* »

Plan

- 1 Virtualisation
 - Histoire de la virtualisation
 - Principes généraux
 - Comprendre la virtualisation
- 2 Le Cloud : vue d'ensemble
- 3 OpenStack : projet, logiciel et utilisation

Historique - 1

- 1946 Premiers ordinateurs « Turing-complet » (ex : ENIAC)
- 1958 Ordinateurs multitaches (Gamma 60 de Bull) : faire tourner plusieurs programmes en même temps, concept proche de la virtualisation.
- 196x Time-sharing systems, premières notions de Virtual Machine avec le projet IBM M44/44X
- 1972 IBM Mainframe Virtual Machine Facility/370 : **premier système de « full virtualisation » !**
- 1988 IBM sort son Hyperviseur PR/SM de type 1

Historique - 2

- 1990 Emulation de processeurs x86, mac sur Amiga (pionnier du genre)
- 1990 Architecture NUMA (Cloisonnement et partitionnement de la mémoire via des bus)
- 1999 VMWare Workstation : hyperviseur de type 2 intelligent : émulation seulement quand nécessaire
- 2000+ Développements de projet libres (QEMU, KVM, Bochs, Xen, etc.)
- 2004 Intel VT-x : Les VM ont directement accès au CPU. Les hyperviseurs ne font plus d'émulation mais contrôlent qui a accès au CPU

Plan

- 1 Virtualisation
 - Histoire de la virtualisation
 - Principes généraux
 - Comprendre la virtualisation
- 2 Le Cloud : vue d'ensemble
- 3 OpenStack : projet, logiciel et utilisation

Principes généraux de Popek et Goldberg

Popek et Goldberg sont deux chercheurs qui ont introduits des conditions pour qu'un système supporte la virtualisation :

- équivalence** fonctionnement identique dans une VM comme sur une machine physique
- efficacité** une part majoritaire d'instructions doit être exécutée directement sans intervention de l'hyperviseur
- contrôle** l'hyperviseur garde le contrôle des ressources et les partage entre les VM

Ces principes datent de 1974 !

Seulement à partir de 2004 (avec Intel VT) qu'ils sont respectés dans l'architecture x86.

Intérêt de la virtualisation

Trois avantages principaux

- Sécurité
- Coût
- Criticité et performances

- Sécurité
 - ▶ Isolation et cloisonnement
 - ★ Ignorance de la présence d'autres environnements
 - ★ Utilisation des protocoles conventionnels
 - ▶ Etudes de sécurité
 - ★ Contrôle et étude d'environnements infectés
 - ★ Répétition de scénarios

Intérêt de la virtualisation - Coût

- Coût
 - ▶ Mutualisation de ressources physiques
 - ★ Sous utilisation actuelle des serveurs
 - ★ Coût de l'énergie électrique
 - ★ Coût de l'espace en centre de données
 - ★ Coût opérationnel
 - ▶ Meilleure gestion des ressources physiques
 - ★ Allocation exclusive
 - ★ Allocation temporelle

Intérêt de la virtualisation - Criticité et performances

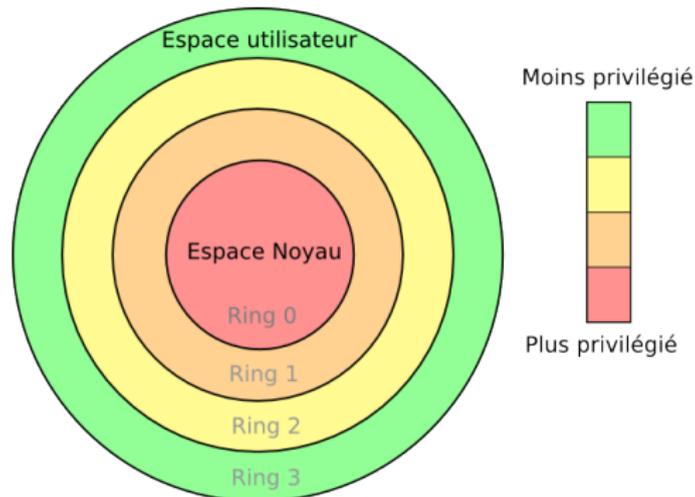
- Criticité et performances
 - ▶ Possibilité de mettre en pause et de copier un environnement logiciel complet
 - ★ Sauvegarde
 - ★ Clonage
 - ▶ Migration d'environnements logiciels
 - ★ Transfert d'un environnement logiciel vers une autre machine physique
 - ▶ Allocation dynamique de ressources
 - ★ Flexibilité de l'offre
 - ★ Adaptabilité en cas de montée en charge

Plan

- 1 Virtualisation
 - Histoire de la virtualisation
 - Principes généraux
 - Comprendre la virtualisation
- 2 Le Cloud : vue d'ensemble
- 3 OpenStack : projet, logiciel et utilisation

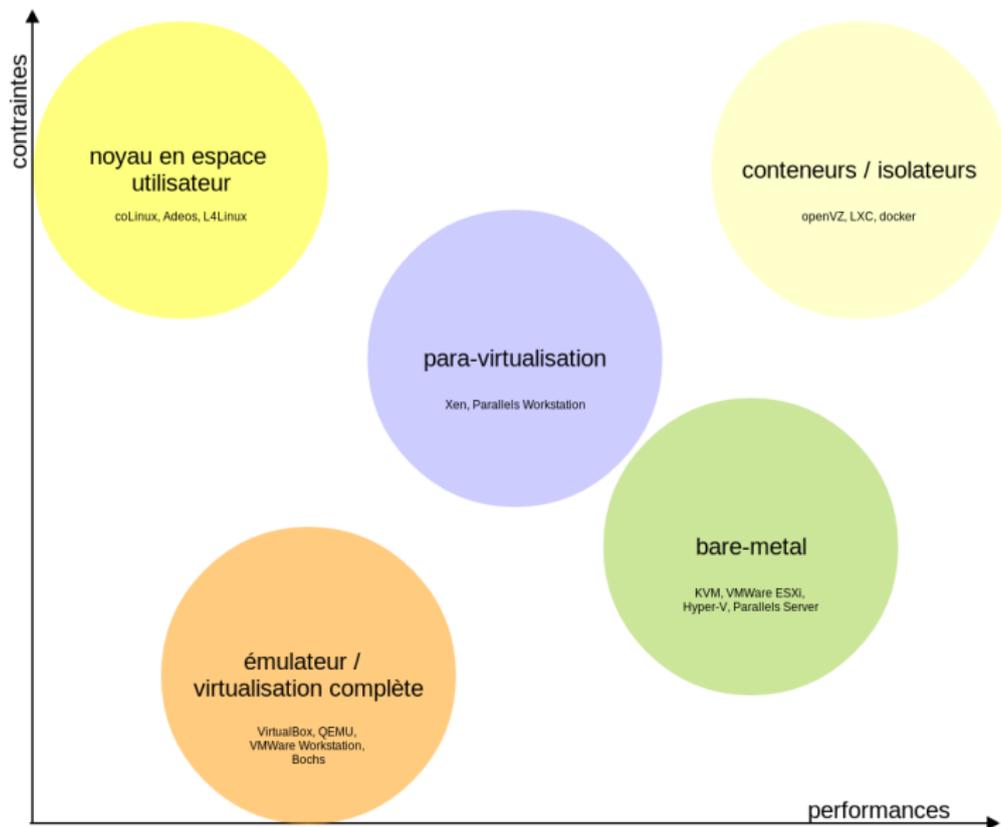
Les anneaux de protection

- 4 niveaux de privilèges dans l'architecture x86
- Ring 0 : Espace Noyau
- Ring 1 et Ring 2 : généralement pas utilisés
- Ring 3 : Espace Utilisateur



Crédit image : « Hertzprung » sur wikipedia, modifié pour le cours

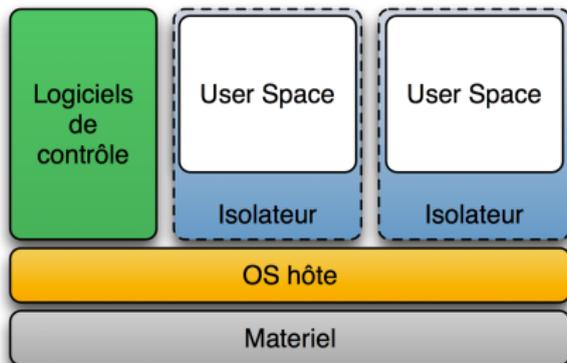
Les modèles de virtualisation en un schéma



Les conteneurs / isolateurs

Principe : isoler l'exécution d'applications dans un contexte (aussi appelé zone d'exécution)

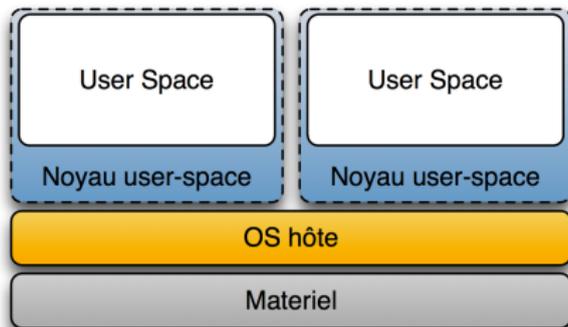
- Pros :
 - ▶ Très performant car peu d'overhead
 - ▶ Permet de faire tourner la même application en mode multi-instance (ex : serveur web)
- Cons :
 - ▶ Même noyau pour toutes les applications
 - ▶ Seulement linux
 - ▶ Pas vraiment de la virtualisation



Noyau en espace utilisateur

Principe : faire tourner un noyau Linux dans l'espace utilisateur

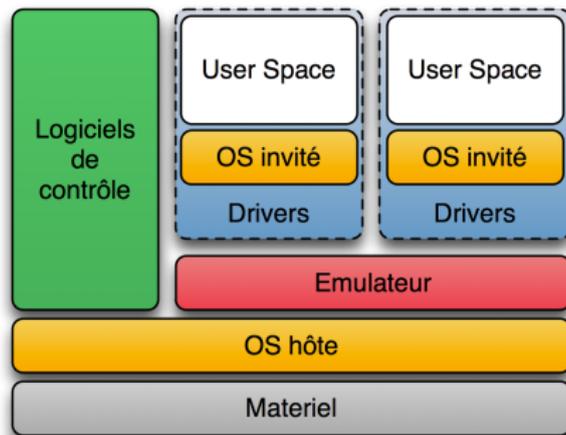
- Pros :
 - ▶ Utile pour tester un noyau linux ou faire du développement de noyau
- Cons :
 - ▶ Pas très performant (empilement de deux noyaux)
 - ▶ Pas très sécurisé (pas d'isolation entre les noyaux)
 - ▶ Nécessite une modification du noyau invité (possible avec linux seulement)



Émulation / Full virtualisation (Hyperviseur de type 2)

Principe : la machine hôte émule le matériel pour la machine invité

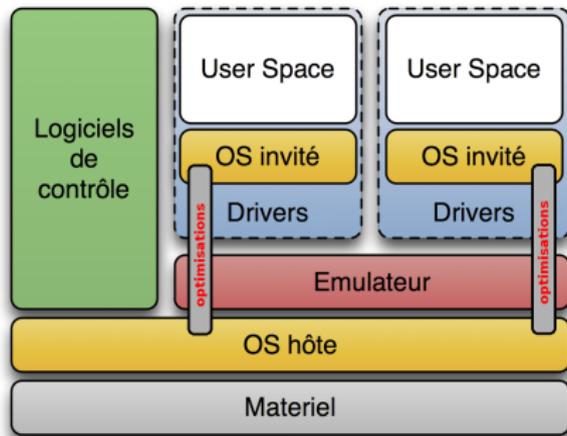
- Pros :
 - ▶ Bonne isolation entre les OS invités
 - ▶ Cohabitation d'architecture CPU et OS hétérogènes
- Cons :
 - ▶ Pas très performant (émulation provoque beaucoup d'overhead)



Para-virtualisation

Principe : évolution de l'émulation par modification des OS invités

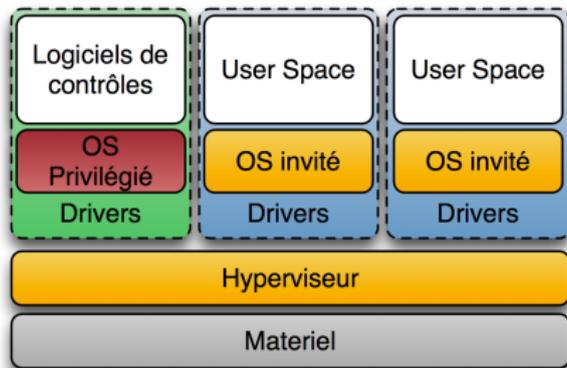
- Pros :
 - ▶ Bonne isolation entre les OS invités
 - ▶ Cohabitation d'architecture CPU et OS hétérogènes
 - ▶ Performance correctes
- Cons :
 - ▶ Nécessite la modification du noyau de l'OS invité (possible sur Linux seulement)



Bare metal (Hyperviseur de type 1)

Principe : noyau système léger qui partage l'accès aux ressources matérielles avec les OS invités

- Pros :
 - ▶ Bonne isolation entre les OS invités
 - ▶ Cohabitation d'architecture CPU et OS hétérogènes
 - ▶ Très performant (couche d'abstraction minimale)
- Cons :
 - ▶ Nécessite la virtualisation matérielle (Intel VT-x ou AMD-V)



Assistance matérielle

À partir de 2004, Intel et AMD ont ajouté à leurs processeurs des instructions CPU supplémentaires pour aider à la virtualisation :

- Intel VT-x, VT-c, VT-d
- AMD-V

« To assist virtualization, VT and Pacifica insert a new privilege level beneath Ring 0. Both add nine new machine code instructions that only work at "Ring -1," intended to be used by the hypervisor. »

Exemples de création de machines virtuelles

Plan

1 Virtualisation

2 Le Cloud : vue d'ensemble

- Le Cloud : les concepts
- PaaS : Platform as a Service
- IaaS : Infrastructure as a Service
- Stockage : block, objet, SDS
- Orchestrer les ressources de son IaaS
- APIs : quel rôle ?

3 OpenStack : projet, logiciel et utilisation

Plan

- 1 Virtualisation
- 2 Le Cloud : vue d'ensemble
 - Le Cloud : les concepts
 - PaaS : Platform as a Service
 - IaaS : Infrastructure as a Service
 - Stockage : block, objet, SDS
 - Orchestrer les ressources de son IaaS
 - APIs : quel rôle ?
- 3 OpenStack : projet, logiciel et utilisation

Le cloud, c'est large !

- Stockage
- Calcul
- Virtualisation++
- Abstraction du matériel
- Service et facturation à la demande
- Accès normalisé par des APIs
- Flexibilité, élasticité

WaaS : Whatever as a Service

- Principalement
 - IaaS Infrastructure as a Service
 - PaaS Platform as a Service
 - SaaS Software as a Service
- Mais aussi :
 - ▶ Database as a Service
 - ▶ Network as a Service
 - ▶ Firewall as a Service
 - ▶ Load balancing as a Service
 - ▶ Whatever as a Service

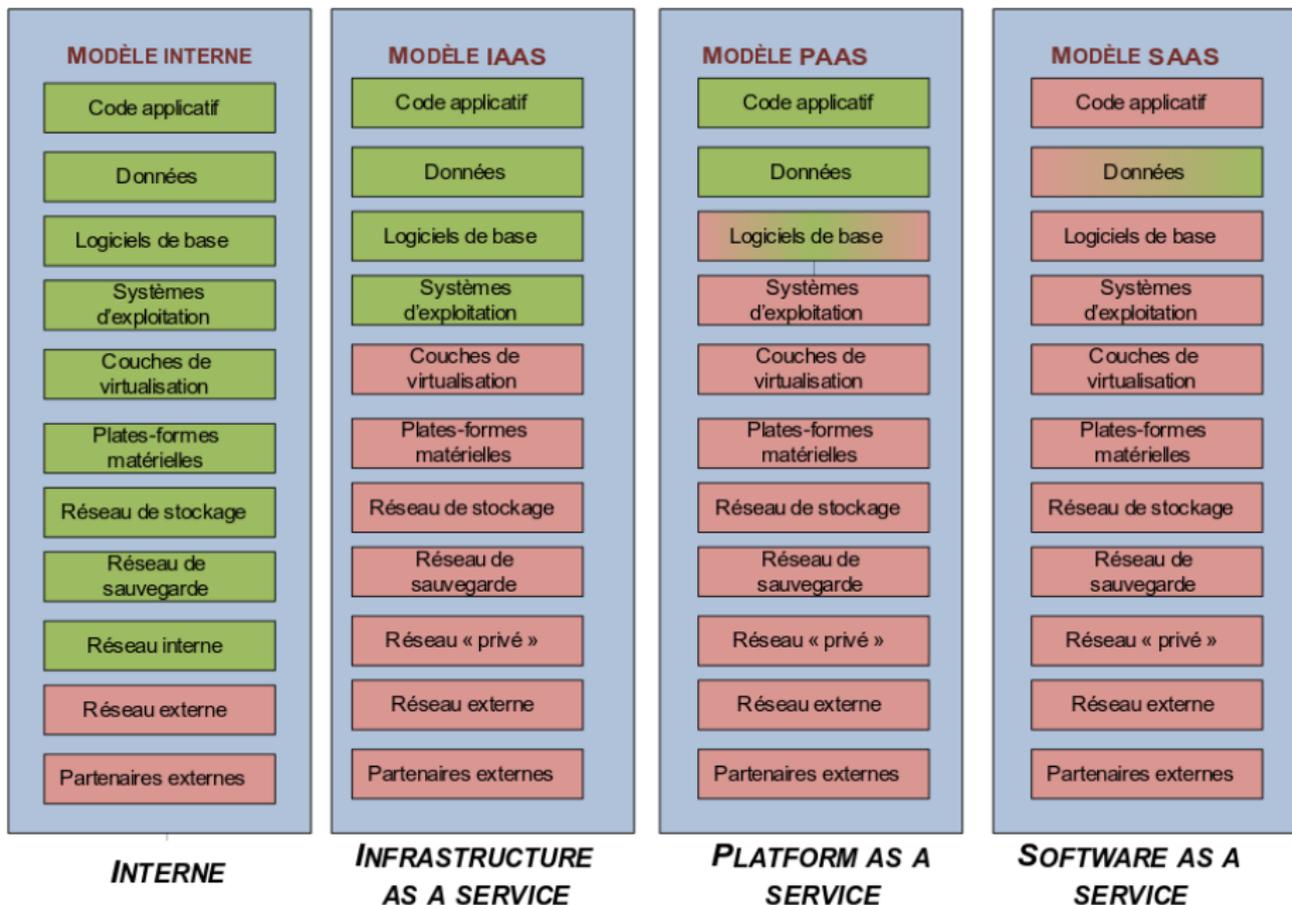
Cloud public ou cloud privé ?

Public fourni par un hébergeur à des clients (OVH, AWS, Rackspace Cloud, Azure, etc.)

Privé plateforme et ressources internes

Hybride utilisation de ressources publiques au sein d'un cloud privé

Les modèles cloud en un schéma



Pourquoi du cloud ? Côté business

- Baisse des coûts par la mutualisation des ressources
- Utilisation uniquement des ressources qui sont nécessaires
- Va de pair avec l'automatisation du SI, l'agilité
- Donc réduction des délais et reproductibilité
- À grande échelle et/ou en cloud hybride, garantie de service

Pourquoi du cloud ? Côté technique

- Abstraction des couches plus basses
- On peut tout programmer à son gré (tout est API!)
- Permet la mise en place d'architectures scalables (en théorie)

Plan

- 1 Virtualisation
- 2 Le Cloud : vue d'ensemble
 - Le Cloud : les concepts
 - PaaS : Platform as a Service
 - IaaS : Infrastructure as a Service
 - Stockage : block, objet, SDS
 - Orchestrer les ressources de son IaaS
 - APIs : quel rôle ?
- 3 OpenStack : projet, logiciel et utilisation

PaaS : les principes

- Fourniture d'une plateforme de développement
- Fourniture d'une plateforme de déploiement
- Pour un langage / un framework
- Principalement utilisé par des développeurs

Exemples de PaaS publics

- Amazon OpsWork / Elastic Beanstalk
- Google App Engine
- Heroku
- OVH (Lamp Stack et Ruby Stack pour développeurs)
<http://www.ovh.com/fr/vps/systeme-exploitation.xml>

Solutions de PaaS privé

- Cloud Foundry
- OpenShift (Red Hat)
- Solum

Plan

- 1 Virtualisation
- 2 Le Cloud : vue d'ensemble
 - Le Cloud : les concepts
 - PaaS : Platform as a Service
 - **IaaS : Infrastructure as a Service**
 - Stockage : block, objet, SDS
 - Orchestrer les ressources de son IaaS
 - APIs : quel rôle ?
- 3 OpenStack : projet, logiciel et utilisation

Amazon Web Services (AWS) et les autres

- Service (cloud public) : [AWS](#)
 - ▶ Pionnier du genre (dès 2006)
 - ▶ Elastic Compute Cloud ([EC2](#))
 - ▶ Elastic Block Storage ([EBS](#))
 - ▶ Simple Storage Service ([S3](#))
- Logiciels libres permettant le déploiement d'un cloud privé :
 - ▶ Eucalyptus
 - ▶ CloudStack
 - ▶ Archipel
 - ▶ OpenNebula
 - ▶ **OpenStack**

Les clouds publics concurrents d'AWS

- Google Compute Platform
- Rackspace
- HP Cloud
- Microsoft Azure
- En France
 - ▶ **Cloudwatt**
 - ▶ **Outscale**
 - ▶ **OVH**
 - ▶ **Ikoula**
 - ▶ **Scaleway (Iliad)**

Virtualisation dans le cloud

- Le cloud IaaS repose souvent sur la virtualisation
- Ressources compute ← virtualisation
- Virtualisation complète : KVM, Xen
- Virtualisation containers : OpenVZ, LXC, Docker

Notions et vocabulaire IaaS

- Images
- Instances
- Types d'instance / gabarits (flavors)
- Volumes
- Stockage block
- Stockage objet
- IP flottantes/élastiques
- Groupes de sécurité
- Paires de clés
- Templates, stacks
- API REST
- API de metadata et user data
- cloud-init, cloud-config

Notions et vocabulaire IaaS

- L'instance est par définition éphémère
- Elle doit être utilisée comme ressource de calcul
- Une image se personnalise lors de son instantiation grâce à l'API de metadata
- Séparer les données des instances
- Choix du type de stockage : éphémère, volume, objet

À retenir

Virtualisation \neq IaaS

Regardons l'interface web d'Amazon

Plan

1 Virtualisation

2 Le Cloud : vue d'ensemble

- Le Cloud : les concepts
- PaaS : Platform as a Service
- IaaS : Infrastructure as a Service
- **Stockage : block, objet, SDS**
- Orchestrer les ressources de son IaaS
- APIs : quel rôle ?

3 OpenStack : projet, logiciel et utilisation

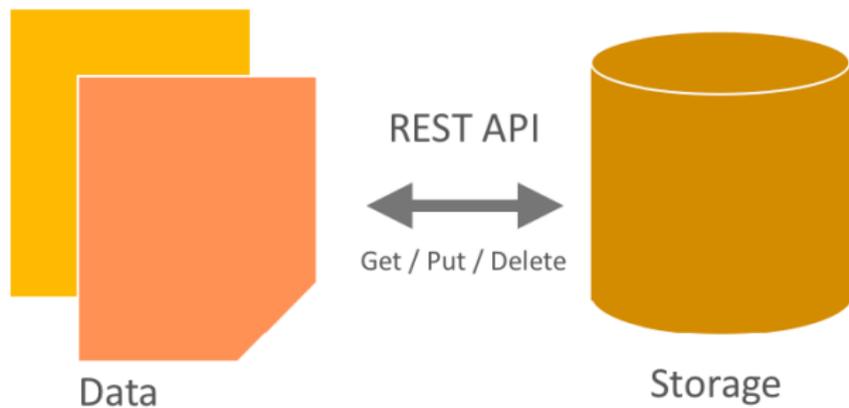
Stockage block

- Accès à des raw devices type `/dev/vdb`
- Possibilité d'utiliser n'importe quel système de fichiers
- Compatible avec toutes les applications legacy
- Technos : iSCSI, Fiber Channel, FCoE

Stockage objet

- Pousser et retirer des objets dans un container/bucket
- Pas de hiérarchie des données, pas de système de fichiers
- Accès par les APIs
- L'application doit être conçue pour tirer partie du stockage objet

Stockage objet : schéma



SDS : Software Defined Storage

- 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

Deux solutions : OpenStack Swift et Ceph

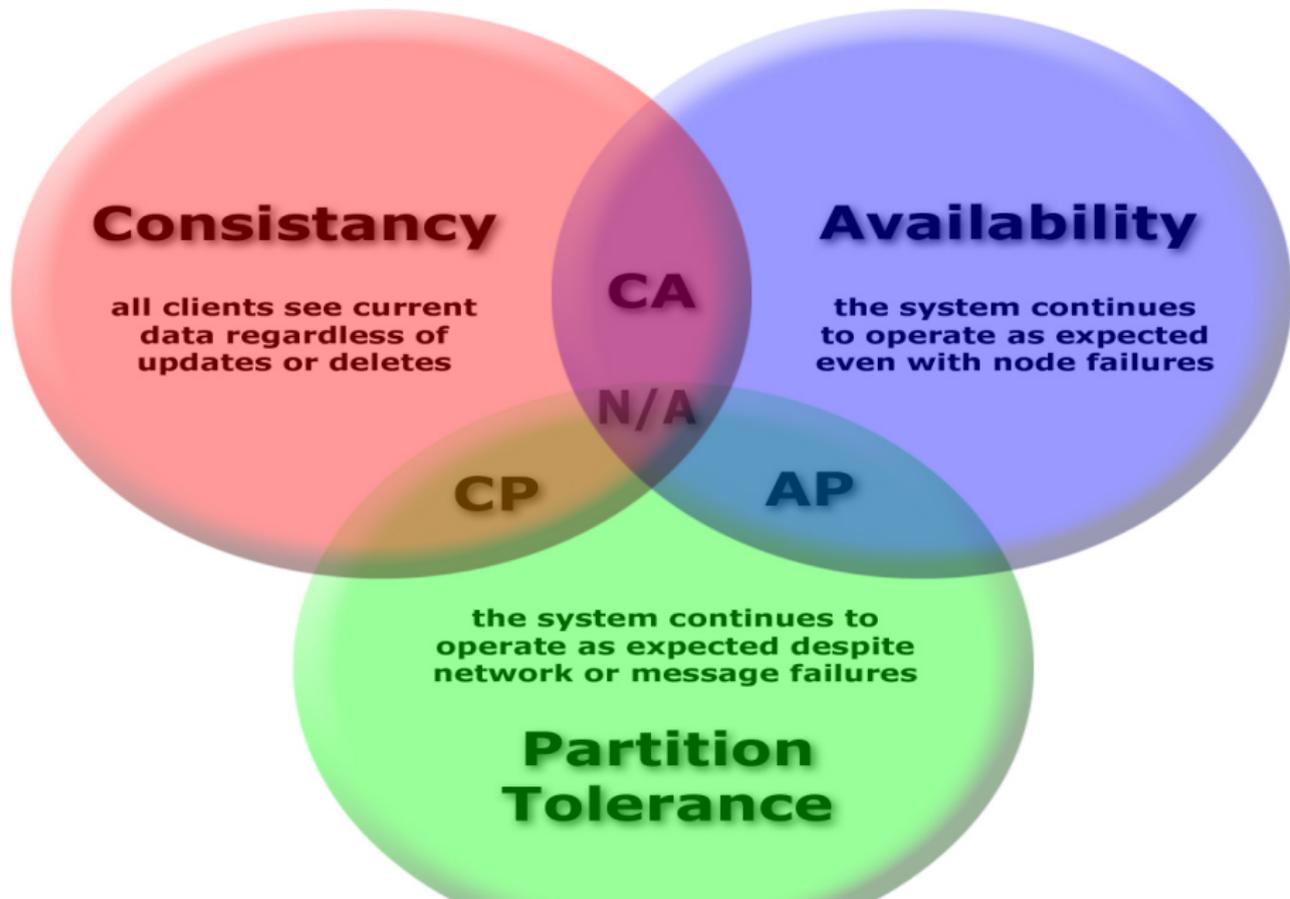
- Swift fait partie du projet OpenStack et fournit du stockage objet
- Ceph fournit du stockage objet, block et FS
- Les deux sont implémentés en SDS
- Théorème CAP : on en choisit deux

Consistency Tous les noeuds du système voient exactement les mêmes données au même moment

Availability Toutes les requêtes doivent recevoir une réponse

Partition Tolerance En cas de morcellement en sous-réseaux, chacun doit pouvoir fonctionner de manière autonome

Théorème CAP



Swift

- Swift est un module d'OpenStack
- Le projet est né chez Rackspace avant la création d'OpenStack
- Swift est en production chez Rackspace depuis lors
- C'est le composant le plus mature d'OpenStack
- Théorème CAP : AP

Ceph

- Projet totalement parallèle à OpenStack
- Supporté par une entreprise (Inktank) récemment rachetée par Red Hat
- Fournit d'abord du stockage objet
- L'accès aux données se fait via RADOS :
 - ▶ Accès direct depuis une application avec librados
 - ▶ Accès via une API REST grâce à radosgw
- La couche RBD permet d'accéder aux données en mode block (volumes)
- CephFS permet un accès par un système de fichiers POSIX
- Théorème CAP : CP

Plan

1 Virtualisation

2 Le Cloud : vue d'ensemble

- Le Cloud : les concepts
- PaaS : Platform as a Service
- IaaS : Infrastructure as a Service
- Stockage : block, objet, SDS
- Orchestrer les ressources de son IaaS
- APIs : quel rôle ?

3 OpenStack : projet, logiciel et utilisation

Pourquoi orchestrer

- Définir toute une infrastructure dans un seul fichier texte
- Être en capacité d'instancier une infrastructure entière en un appel API
- Autoscaling
 - ▶ Adapter ses ressources en fonction de ses besoins en temps réel
 - ▶ Fonctionnalité incluse dans le composant d'orchestration d'OpenStack

Exemples d'orchestration

- Amazon CloudFormation
- OpenStack Heat
- Microsoft Azure Automation

Exemples d'orchestration

```
resources:
  my_instance:
    type: AWS::EC2::Instance
    properties:
      KeyName: { get_param: KeyName }
      ImageId: { get_param: ImageId }
      InstanceType: { get_param: InstanceType }
```

Plan

- 1 Virtualisation
- 2 Le Cloud : vue d'ensemble
 - Le Cloud : les concepts
 - PaaS : Platform as a Service
 - IaaS : Infrastructure as a Service
 - Stockage : block, objet, SDS
 - Orchestrer les ressources de son IaaS
 - APIs : quel rôle ?
- 3 OpenStack : projet, logiciel et utilisation

API ?

- *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)
- Il s'agit le plus souvent d'API HTTP REST

Exemple concret

```
GET /v2.0/networks/<network_id>
{
  "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
  }
}
```

Plan

- 1 Virtualisation
- 2 Le Cloud : vue d'ensemble
- 3 **OpenStack : projet, logiciel et utilisation**
 - Présentation du projet et du logiciel
 - Utiliser OpenStack

Plan

- 1 Virtualisation
- 2 Le Cloud : vue d'ensemble
- 3 OpenStack : projet, logiciel et utilisation
 - Présentation du projet et du logiciel
 - Utiliser OpenStack

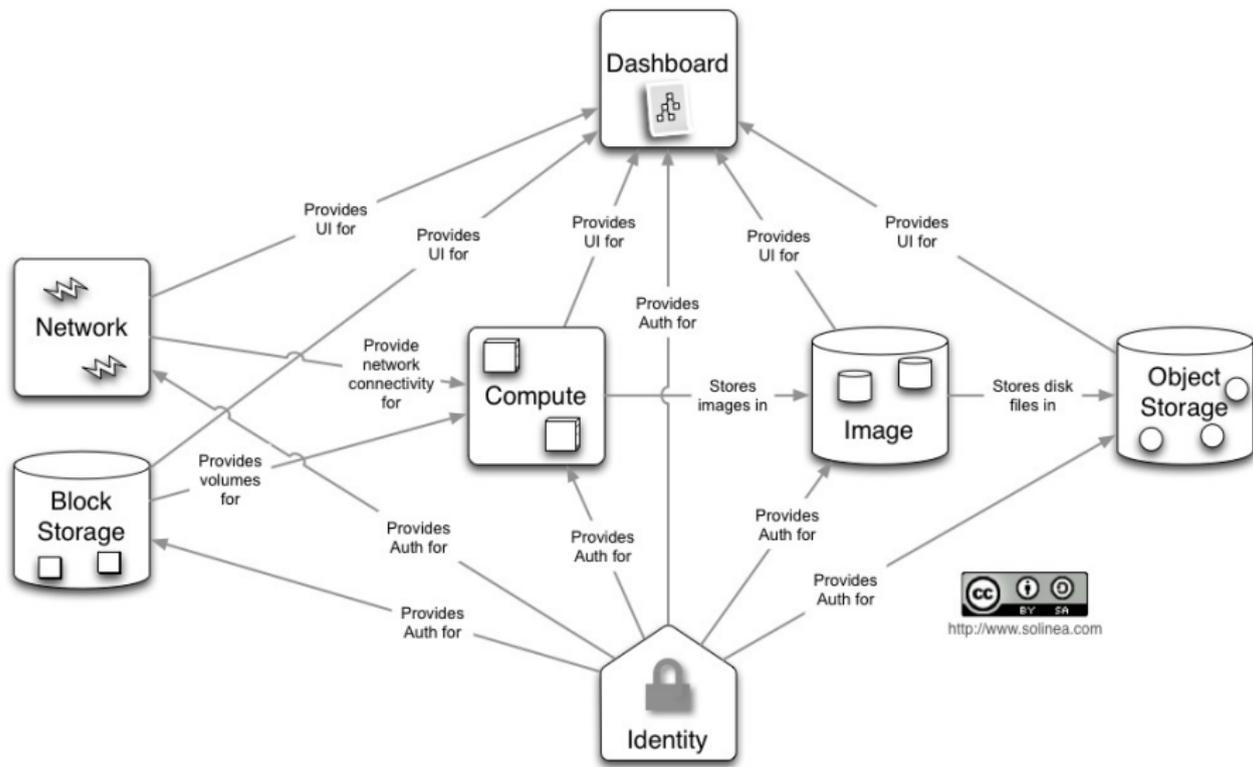
Historique

- Démarrage en 2010
- Objectif : construire le Cloud Operating System libre
- Fusion de deux projets de Rackspace (Storage) et de la NASA (Compute)
- Développé en Python et distribué sous licence Apache 2.0
- Les dernières releases jusqu'à aujourd'hui :
 - ▶ Mitaka (2016.1)
 - ▶ Newton (2016.2)
 - ▶ Ocata (2017.1)
 - ▶ Pike (2017.2)
 - ▶ Queens (2018.1) – prévu pour avril
 - ▶ Rocky (2018.2) – prévu pour octobre

Les 4 Opens

- Open Source
- Open Design
- Open Development
- Open Community

Vue haut niveau



Les différents sous-projets principaux

- OpenStack est composé d'un ensemble de briques :
 - ▶ OpenStack Compute - Nova
 - ▶ OpenStack Block Storage - Cinder
 - ▶ OpenStack Networking - Neutron
 - ▶ OpenStack Image Service - Glance
 - ▶ OpenStack Identity Service - Keystone
 - ▶ OpenStack Telemetry - Ceilometer
 - ▶ OpenStack Orchestration - Heat
 - ▶ OpenStack Dashboard - Horizon
 - ▶ OpenStack (Object) Storage - Swift
 - ▶ et beaucoup d'autres...

Implémentation

- Chaque brique est découpée en plusieurs services
- Ex. avec nova :
 - ▶ nova-api
 - ▶ nova-compute
 - ▶ nova-scheduler
 - ▶ nova-conductor
 - ▶ etc.
- Communication entre les services : AMQP (souvent RabbitMQ)
- Base de données : relationnelle SQL (souvent MySQL ou MariaDB)
- Neutron utilise pour le réseau : OpenVSwitch et LinuxBridge
- En général : réutilisation de composants existants
- Tout est développé en Python (Django pour la partie web)
- APIs supportées : OpenStack et équivalent Amazon
- Multi tenancy

Plan

- 1 Virtualisation
- 2 Le Cloud : vue d'ensemble
- 3 OpenStack : projet, logiciel et utilisation
 - Présentation du projet et du logiciel
 - Utiliser OpenStack

Le principe

- Les APIs sont le principal atout d'OpenStack
- Toutes les fonctionnalités sont accessibles par l'API
- Les clients (y compris Horizon) utilisent l'API
- Des credentials sont nécessaires
 - ▶ API OpenStack : utilisateur + mot de passe + tenant
 - ▶ API AWS : access key ID + secret access key

Les APIs OpenStack

- Chaque service OpenStack a sa propre API
- Chaque API est versionnée, la rétro-compatibilité est assurée
- Toutes les APIs sont HTTP REST
- Certains services sont aussi accessibles via une API différente compatible AWS
- <http://developer.openstack.org/api-ref.html>

Authentification et catalogue de service

- Avant tout appel a un des services par l'API, il faut s'identifier
- Récupération d'un jeton (token)
- Récupération du catalogue de services
- Pour chaque service, récupération d'un ou plusieurs endpoints HTTP (API)

Accès aux APIs

- Direct, en HTTP, via des outils comme curl
- Avec une bibliothèque
 - ▶ OpenStack SDK (python)
 - ▶ Shade (python)
 - ▶ Les librairies python-client de chaque projet
 - ▶ D'autres implémentations pour d'autres langages (exemple : jclouds)
- Avec les clients officiels en ligne de commande
- Avec Horizon
- <http://docs.openstack.org/user-guide/sdk-overview.html>

Clients officiels

- Le projet OpenStack fournit des clients officiels : python-PROJETclient
- Dedans on trouve :
 - ▶ Bibliothèques Python
 - ▶ Outils CLI
 - ★ L'authentification se fait en passant les credentials par paramètres ou variables d'environnement
 - ★ L'option `-debug` affiche la communication HTTP

OpenStack Client

- Un nouveau client qui les unifie tous
- Commandes du type *openstack service action*
- Vise à remplacer à terme les clients spécifiques
- Permet une expérience utilisateur plus homogène
- Possibilité d'utiliser un fichier de configuration clouds.yaml

Ressources

- Applications : <http://apps.openstack.org/>
- Actualités :
 - ▶ Blog officiel : <http://www.openstack.org/blog/>
 - ▶ Planet : <http://planet.openstack.org>
 - ▶ Superuser : <http://superuser.openstack.org/>
 - ▶ OpenStack Community Weekly Newsletter
- Ressources commerciales : <http://www.openstack.org/marketplace/>
entre autres

Démo et TP

The screenshot shows the OpenStack dashboard interface. At the top left is the OpenStack logo and the text 'openstack'. To the right of the logo is a dropdown menu showing 'demo'. Further right is a user profile icon labeled 'admin' and a 'Se déconnecter' button. The main content area is titled 'Instances' and contains a sub-section also titled 'Instances'. Below this sub-section is a table with the following columns: 'Nom de l'instance', 'Nom de l'image', 'Adresse IP', 'Taille', 'Key Pair', 'Etat', 'Zone de Disponibilité', 'Tâche', 'Etat de l'alimentation', 'Durée de Fonctionnement', and 'Actions'. The table contains one row for an instance named 'myinstance' with the following details: image 'cirros-0.3.2-x86_64-uec', IP '10.0.0.7', size 'm1.tiny', key pair '-', state 'Active', availability zone 'nova', task 'None', power state 'Running', and uptime '0 minute'. Below the table, it says 'Affichage de 1 élément'. To the right of the table are buttons for 'Lancer une instance', 'Soft Reboot instances', and 'Terminate instances'. A sidebar on the left contains navigation links for 'Projet', 'Compute', 'Vue d'Ensemble', 'Instances', 'Volumes', 'Images', 'Accès et Sécurité', 'Réseau', 'Orchestration', 'Admin', and 'Identity'. The 'Instances' link is currently selected.

<input type="checkbox"/>	Nom de l'instance	Nom de l'image	Adresse IP	Taille	Key Pair	Etat	Zone de Disponibilité	Tâche	Etat de l'alimentation	Durée de Fonctionnement	Actions
<input type="checkbox"/>	myinstance	cirros-0.3.2-x86_64-uec	10.0.0.7	m1.tiny	-	Active	nova	None	Running	0 minute	Créer un instantané