

Formation OpenStack

Arnaud Morin

Orange Labs



Arnaud Morin (Orange Labs)

Formation OpenStack

1 / 219



Objectifs de la formation

- Virtualisation
- Cloud
- Focus sur OpenStack

Concernant ces supports de cours

Auteurs initiaux :

- Adrien Cunin <adrien.cunin@osones.com> OSONES
- Pierre Freund <pierre.freund@osones.com> OSONES

Modifiés et adaptés par :

- Arnaud Morin <arnaud1.morin@orange.com>

Licence

- Copyright ©2014 Osones
- Creative Commons BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0/deed.fr>

Sources

- <https://github.com/0sones/OpenStack-Formations/>
- <https://github.com/arnaudmorinol/OpenStack-Formations/>

Arnaud Morin (Orange Labs)

Formation OpenStack

2 / 219

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

Arnaud Morin (Orange Labs)

Formation OpenStack

3 / 219



Arnaud Morin (Orange Labs)

Formation OpenStack

4 / 219



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 partie au mieux de l'IaaS
- Pouvoir déterminer ce qui est compatible avec la philosophie cloud ou pas
- Adapter ses méthodes d'administration système à un environnement cloud



Plan

- ① Virtualisation
- ② Le Cloud : vue d'ensemble
- ③ OpenStack : projet, logiciel et utilisation
- ④ Déployer OpenStack
- ⑤ Tirer partie de l'IaaS



Objectifs de la formation : OpenStack

- Connaitre le fonctionnement du projet OpenStack et ses possibilités
- Comprendre le fonctionnement de chacun des composants d'OpenStack
- Pouvoir faire les bons choix de configuration
- Savoir déployer manuellement un cloud OpenStack pour fournir de l'IaaS
- Connaitre les bonnes pratiques de déploiement d'OpenStack
- Être capable de déterminer l'origine d'une erreur dans OpenStack
- Savoir réagir face à un bug



Plan

- ① Virtualisation
 - Histoire de la virtualisation
 - Principes généraux
 - Comprendre la virtualisation
- ② Le Cloud : vue d'ensemble
- ③ OpenStack : projet, logiciel et utilisation
- ④ Déployer OpenStack
- ⑤ Tirer partie de l'IaaS



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

4 Déployer OpenStack

5 Tirer partie de l'IaaS



Historique - 1

- 1946 Premiers ordinateurs « Turing-complet » (ex : ENIAC)
- 1958 Ordinateurs multitâches (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 projets 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

4 Déployer OpenStack

5 Tirer partie de l'IaaS



Intérêt de la virtualisation

Trois avantages principaux

- Sécurité
- Coût
- Criticité et performances



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 - Sécurité

• 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



Plan

- 1 Virtualisation
 - Histoire de la virtualisation
 - Principes généraux
 - Comprendre la virtualisation
- 2 Le Cloud : vue d'ensemble
- 3 OpenStack : projet, logiciel et utilisation
- 4 Déployer OpenStack
- 5 Tirer partie de l'IaaS



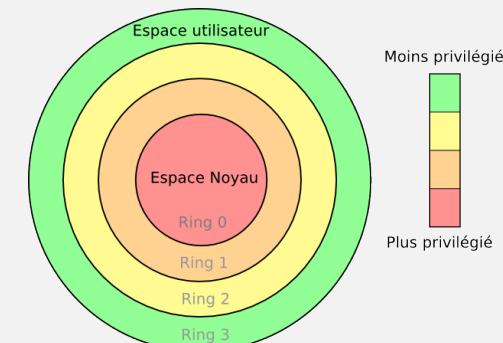
Intérêt de la virtualisation - Criticité et performances

- 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



Les anneaux de protection

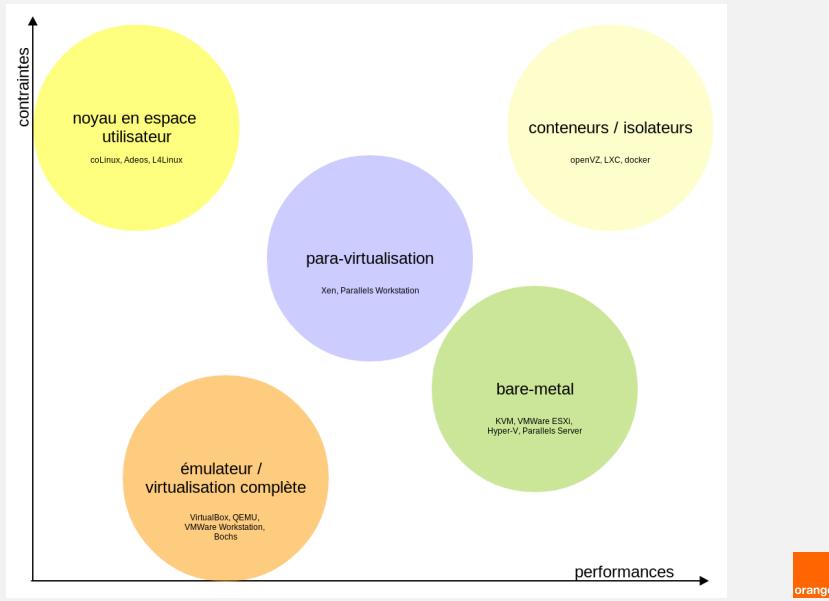
- 4 niveaux de privilège dans l'architecture x86
- Ring 0 : Espace Noyau
- Ring 1 et Ring 2 : généralement pas utilisés
- Ring 3 : Espace Utilisateur



Crédit image : <> Hertzprung <> sur wikipedia, modifié pour le cours



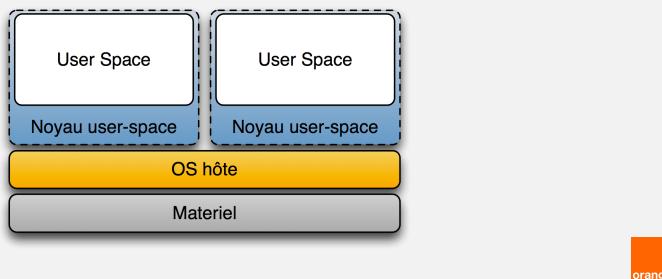
Les modèles de virtualisation en un schéma



Noyau en espace utilisateur

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

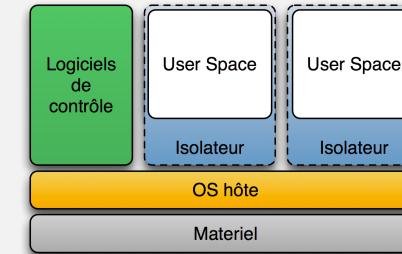
- Pros :
 - ▶ Utile pour tester un noyau linux ou faire du développement de noyau
- Cons :
 - ▶ Pas très performant (empilement de deux noyaux)
 - ▶ Pas très sécurisé (pas d'isolation entre les noyaux)
 - ▶ Nécessite une modification du noyau invité (possible avec linux seulement)



Les conteneurs / isolateurs

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

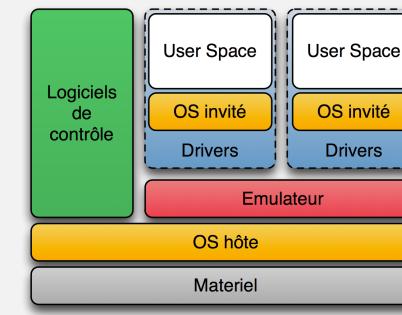
- Pros :
 - ▶ Très performant car peu d'overhead
 - ▶ Permet de faire tourner la même application en mode multi-instance (ex : serveur web)
- Cons :
 - ▶ Même noyau pour toutes les applications
 - ▶ Seulement linux
 - ▶ Pas vraiment de la virtualisation



Émulation / Full virtualisation (Hyperviseur de type 2)

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

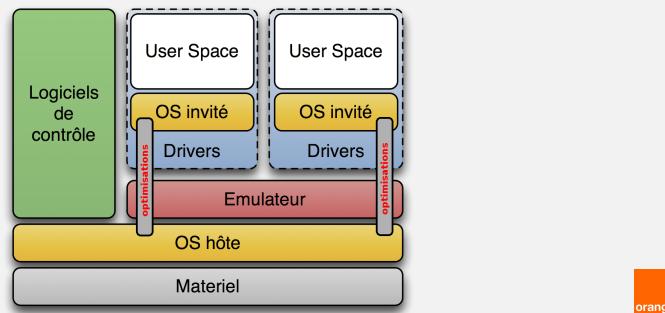
- Pros :
 - ▶ Bonne isolation entre les OS invités
 - ▶ Cohabitation d'architecture CPU et OS hétérogènes
- Cons :
 - ▶ Pas très performant (émulation provoque beaucoup d'overhead)



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)



Assistance matérielle

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

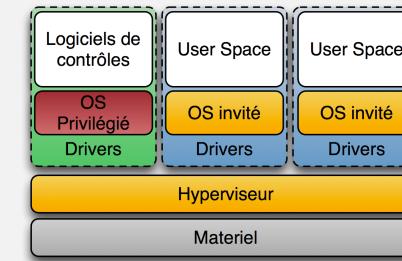
- Intel VT-x, VT-c, VT-d
- AMD-V

« 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. »

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)



Démo

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

4 Déployer OpenStack

5 Tirer partie de l'IaaS



Le cloud, c'est large !

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



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

4 Déployer OpenStack

5 Tirer partie de l'IaaS



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 ?

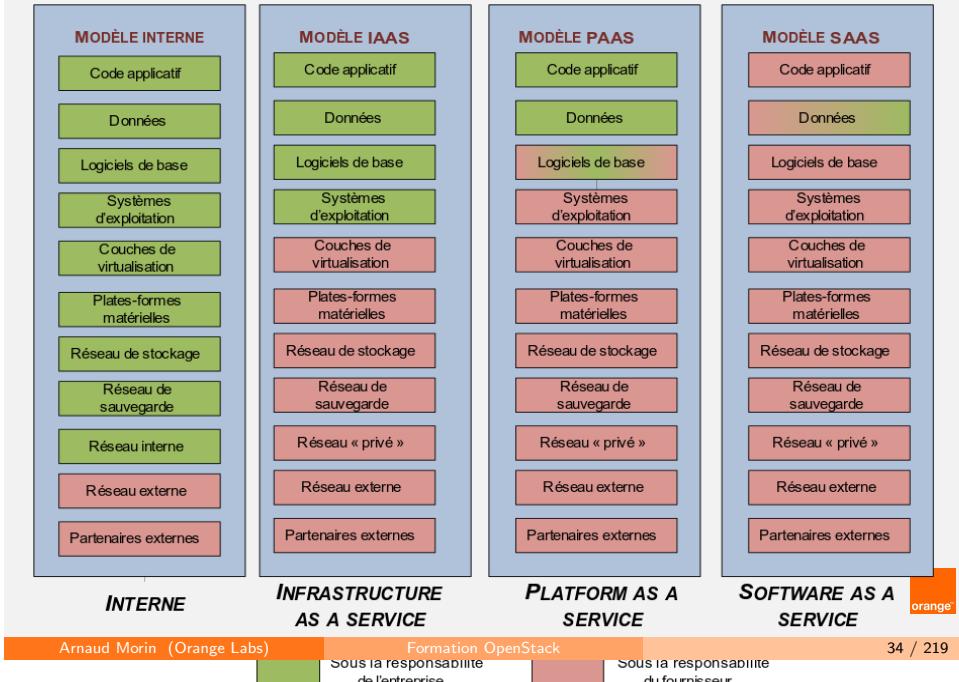
Public fourni par un hébergeur à des clients (AWS, Rackspace Cloud, Azure, etc.)

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
- À grande échelle, 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

4 Déployer OpenStack

5 Tirer partie de l'IaaS



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>



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



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

4 Déployer OpenStack

5 Tirer partie de l'IaaS



Les clouds publics concurrents d'AWS

- Google Compute Engine
- Rackspace
- HP Cloud
- Microsoft Azure
- En France
 - ▶ **Cloudwatt**
 - ▶ **Numergy**
 - ▶ **Outscale**



Amazon Web Services (AWS) et les autres

Service (cloud public) : AWS

- ▶ Pionnier du genre (dès 2002)
- ▶ Elastic Compute Cloud ([EC2](#))
- ▶ Elastic Block Storage ([EBS](#))
- ▶ Simple Storage Service ([S3](#))

Logiciels libres permettant le déploiement d'un cloud privé :

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



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 (flavors)
- Volumes
- Stockage block
- Stockage objet
- IP flottantes/élastiques
- Groupes de sécurité
- Paires de clés
- API REST
- API de metadata et user data
- Cloud-init



À retenir

Virtualisation \neq IaaS



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



AWS

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

4 Déployer OpenStack

5 Tirer partie de l'IaaS

Arnaud Morin (Orange Labs)

Formation OpenStack

49 / 219



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 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

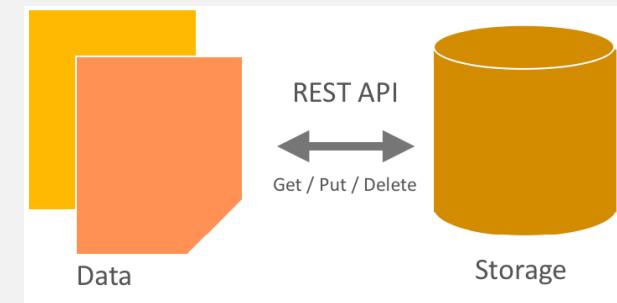
Arnaud Morin (Orange Labs)

Formation OpenStack

50 / 219



Stockage objet : schéma



Arnaud Morin (Orange Labs)

Formation OpenStack

51 / 219



Arnaud Morin (Orange Labs)

Formation OpenStack

52 / 219

SDS : Software Defined Storage

- Utilisation de commodity hardware
- Pas de RAID matériel
- Le logiciel est responsable de garantir les données



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 du SDS
- Théorème CAP : on en choisit deux

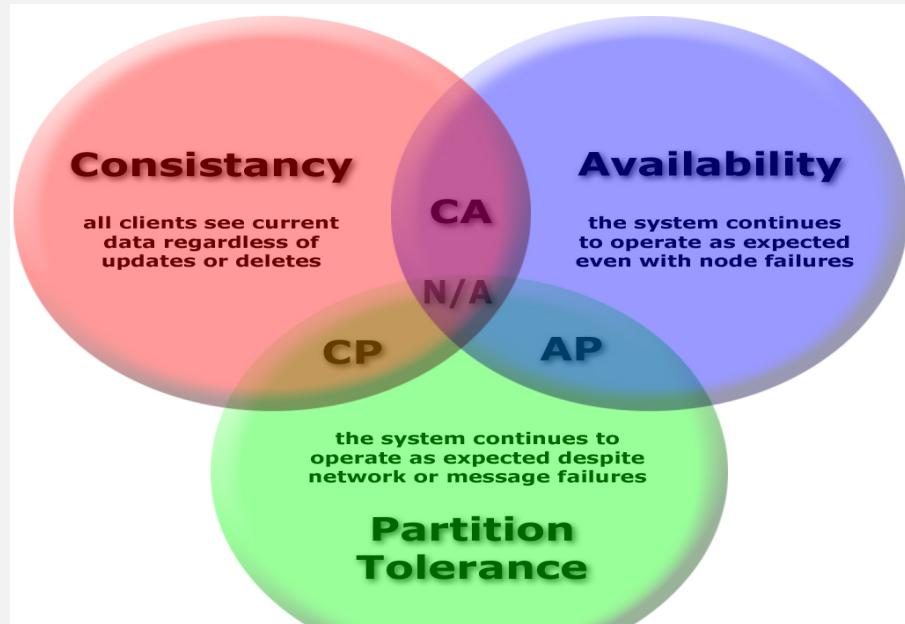
Consistency Tous les noeuds du système voient exactement les mêmes données au même moment

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 projet 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



Pourquoi orchestrer

- Définir tout une infrastructure dans un seul fichier texte
- Être en capacité d'instancier une infrastructure entière en un appel API
- Adapter ses ressources en fonction de ses besoins en temps réel (autoscaling)



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

4 Déployer OpenStack

5 Tirer partie de l'IaaS



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 }
```



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



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

4 Déployer OpenStack

5 Tirer partie de l'IaaS



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

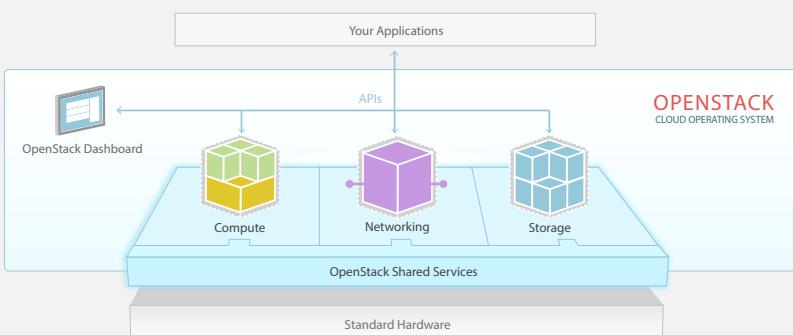
- Présentation du projet et du logiciel
- DevStack : faire tourner rapidement un OpenStack
- Utiliser OpenStack

4 Déployer OpenStack

5 Tirer partie de l'IaaS



Vue haut niveau



Plan

1 Virtualisation

2 Le Cloud : vue d'ensemble

3 OpenStack : projet, logiciel et utilisation

- Présentation du projet et du logiciel
- DevStack : faire tourner rapidement un OpenStack
- Utiliser OpenStack

4 Déployer OpenStack

5 Tirer partie de l'IaaS



Historique

- Démarrage en 2010
- Objectif : 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 releases jusqu'à aujourd'hui :
 - ▶ Austin (2010.1)
 - ▶ Bexar (2011.1)
 - ▶ Cactus (2011.2)
 - ▶ Diablo (2011.3)
 - ▶ Essex (2012.1)
 - ▶ Folsom (2012.2)
 - ▶ Grizzly (2013.1)
 - ▶ Havana (2013.2)
 - ▶ **Icehouse (2014.1)**
 - ▶ Novembre 2014 : Juno



Quelques soutiens/contributeurs ...

- Rackspace et la NASA
- Canonical
- Red Hat
- Suse
- HP
- IBM
- Dell, Intel
- Cisco, Juniper
- NetApp, VMWare
- Yahoo, Bull
- Mais aussi : Mirantis, StackOps, eNovance

<http://www.openstack.org/foundation/companies/>



Les différents sous-projets

- OpenStack Compute - Nova
- OpenStack (Object) Storage - Swift
- OpenStack Block Storage - Cinder
- OpenStack Networking - Neutron
- OpenStack Image Service - Glance
- OpenStack Identity Service - Keystone
- OpenStack Dashboard - Horizon
- OpenStack Metering - Ceilometer
- OpenStack Orchestration - Heat
- OpenStack Database Service - Trove



... et utilisateurs

- Tous les contributeurs précédemment cités
- En France : **CloudWatt** et **Numergy**
- Wikimedia
- CERN
- Paypal
- Comcast
- Etc. Sans compter les implémentations confidentielles

<http://www.openstack.org/user-stories/>

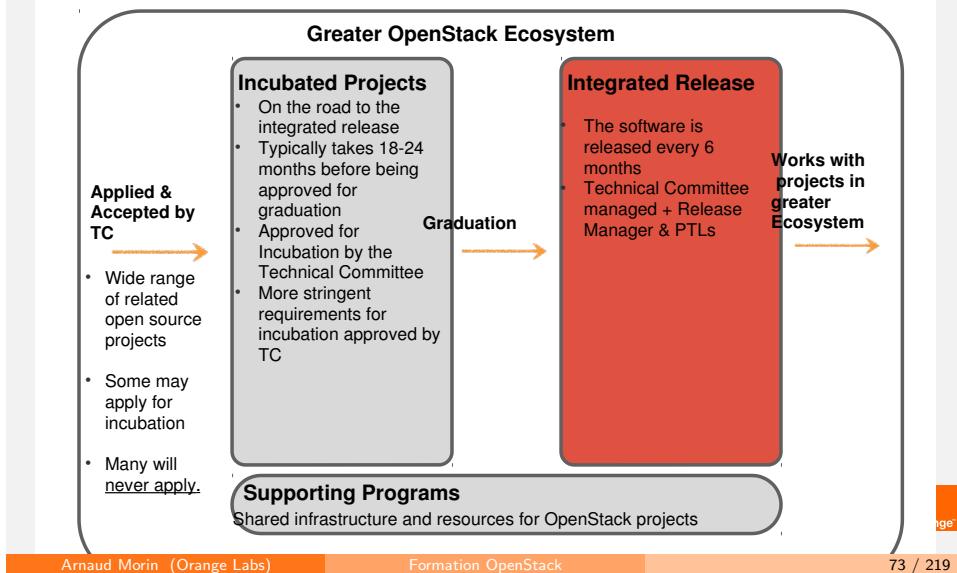


Les différents sous-projets (2)

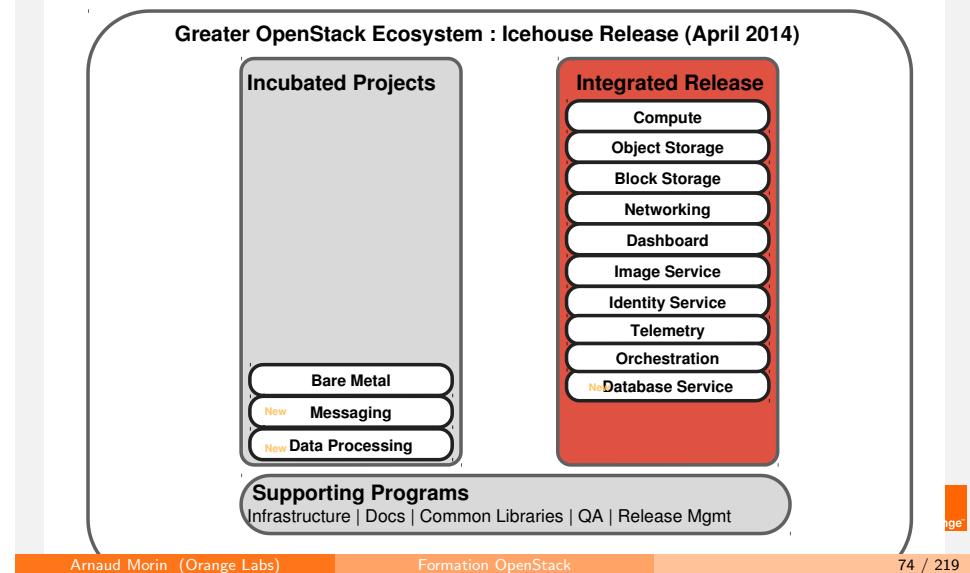
- Incubating (à jour Icehouse)
 - ▶ Bare metal (Ironic)
 - ▶ Queue service (Marconi)
 - ▶ Data processing (Sahara)
- Intéressants
 - ▶ DNS service (Designate)
- Autres
 - ▶ Oslo
 - ▶ rootwrap : wrapper pour les commandes root utilisé par les projets
 - ▶ TripleO
 - ▶ Tempest, Grenade
 - ▶ Les clients (python-*client)



Cycle de vie des projets au sein d'OpenStack



Les projets dans Icehouse



Nouveautés de Icehouse



- Nova : rolling upgrades
- Nouvelles fonctionnalités disponibles dans Horizon
- Nouveaux drivers dans Neutron
- Nouveaux drivers dans Cinder
- Le nouveau format HOT est recommandé pour les templates Heat
- etc.
- En détails :
<https://wiki.openstack.org/wiki/ReleaseNotes/Icehouse>

Traduction

- La question de la traduction est dorénavant prise en compte
- Seules certaines parties sont traduites, comme Horizon
- La traduction française est aujourd'hui une des plus avancées
- Utilisation de la plateforme Transifex

Développement du projet

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



La fondation OpenStack

- Entité de gouvernance principale du projet
- Les membres du board sont issus des entreprises sponsors et élus par les membres individuels
- Tout le monde peut devenir membre individuel (gratuitement)
- La fondation supporte le projet par différents moyens :
 - ▶ Événements : organisation (Summits) ou participation (OSCON, etc.)
 - ▶ Infrastructure de développement (serveurs)
 - ▶ Ressources humaines : marketing, release manager, quelques développeurs (principalement sur l'infrastructure)
- 850 organisations à travers le monde
- 9500 membres individuels dans 100 pays

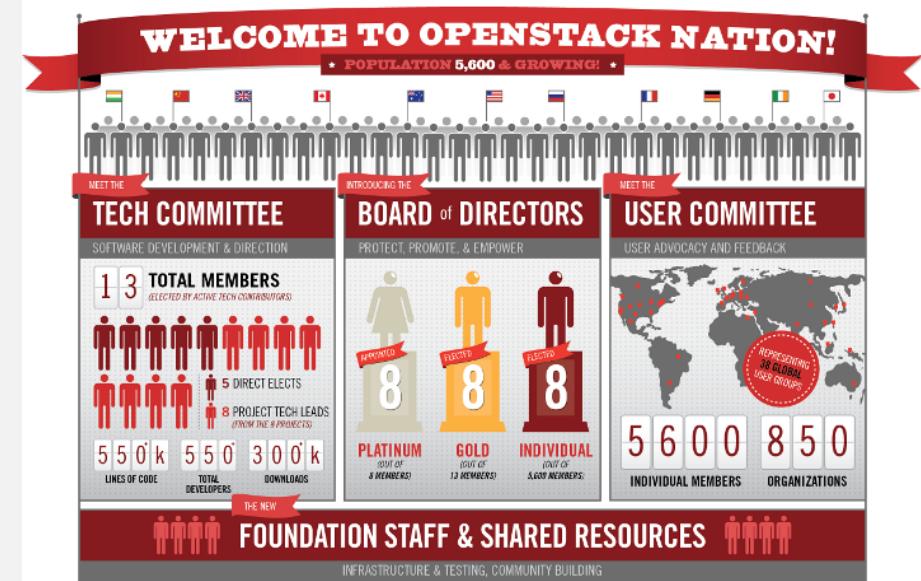


Développement du projet : en détails

- Ouvert à tous (individuels et entreprises)
- Cycle de développement de 6 mois débuté par un (design) summit
- Outils : Launchpad (blueprints, bugs) + Git + GitHub (code)
- Sur chaque commit : peer review (Gerrit) + intégration continue (exécution de différents tests par Jenkins)
- Plateforme de référence et modèle de développement : Ubuntu
- Développement hyper actif : 17000 commits dans Icehouse (+25%)
- Fin 2012, création d'une entité indépendante de gouvernance : la fondation OpenStack



La fondation OpenStack



Cycle de développement : 6 mois

- Le planning est publié, exemple : https://wiki.openstack.org/wiki/Icehouse_Release_Schedule
- Milestone releases
- Freezes : FeatureProposal, Feature, String
- RC releases
- Stable releases
- Ce modèle de cycle de développement a évolué depuis le début du projet
- Cas particulier de Swift



Stackforge

- Forge pour les nouveaux projets en lien avec OpenStack
- Ils bénéficient de l'infrastructure du projet OpenStack, mais la séparation reste claire
- Les projets démarrent dans Stackforge et peuvent ensuite rejoindre le projet OpenStack



Où trouver des informations sur le développement d'OpenStack

- Principalement sur le wiki : <https://wiki.openstack.org>
- Les blueprints et bugs sur Launchpad : <https://launchpad.net/openstack>
- Les patchs proposés et leurs reviews sont sur Gerrit : <https://review.openstack.org>
- Le code est disponible dans Git : <https://git.openstack.org>
- Les sources (tarballs) sont disponibles : <http://tarballs.openstack.org/>



OpenStack Summit

- Aux USA jusqu'en 2013
- Aujourd'hui : alternance USA et Asie/Europe
- Quelques centaines au début à 4500 de participants aujourd'hui
- En parallèle : conférence (utilisateurs, décideurs) et Design Summit (développeurs)
- Détermine le nom de la release : lieu/ville à proximité du Summit



Exemple du Summit d'avril 2013 à Portland

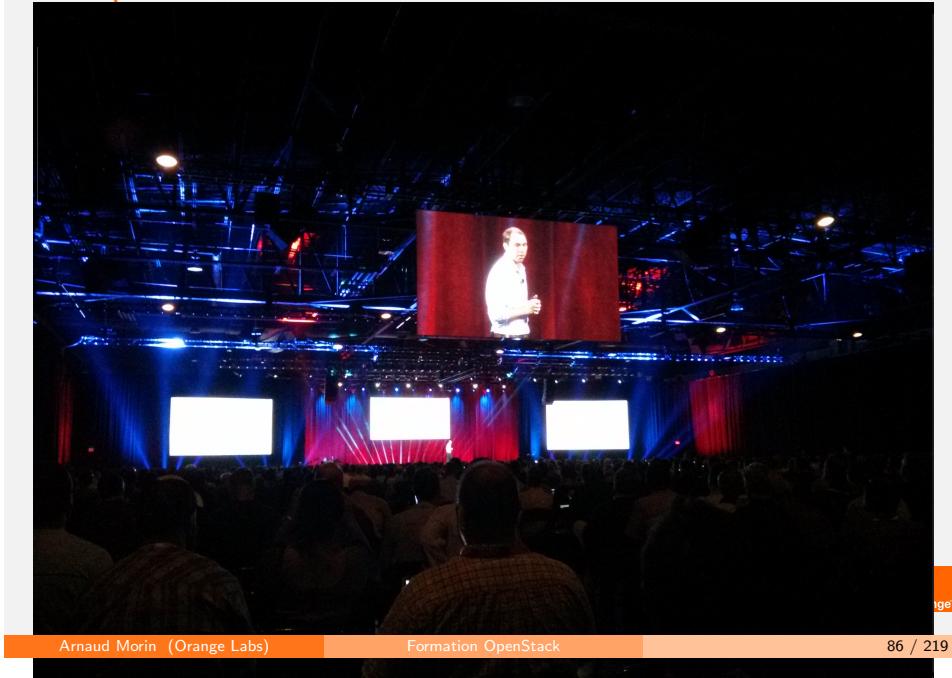


Arnaud Morin (Orange Labs)

Formation OpenStack

85 / 219

Exemple du Summit de mai 2014 à Atlanta

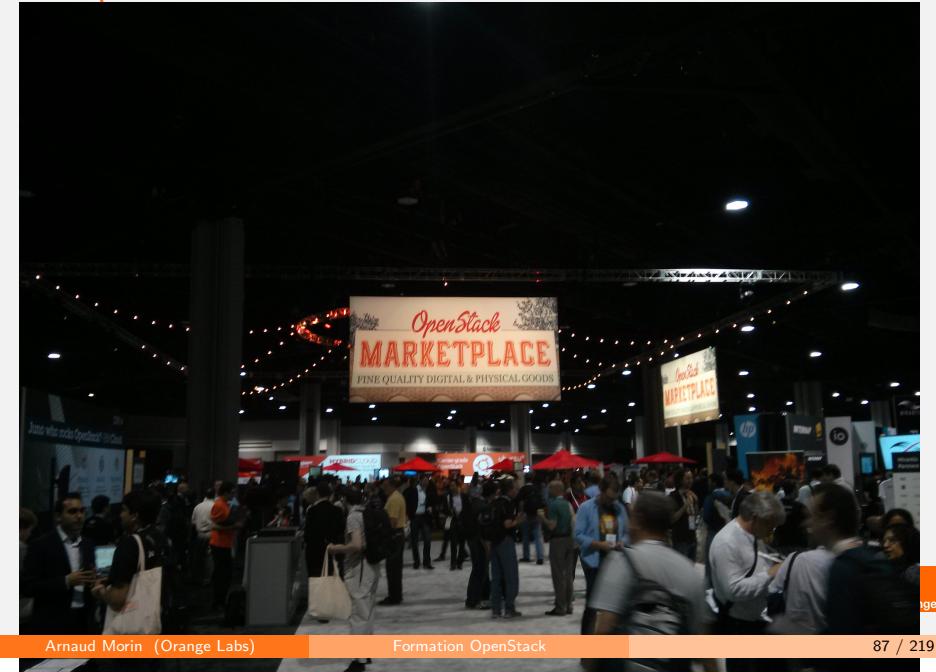


Arnaud Morin (Orange Labs)

Formation OpenStack

86 / 219

Exemple du Summit de mai 2014 à Atlanta



Arnaud Morin (Orange Labs)

Formation OpenStack

87 / 219

Principes de conceptions

<https://wiki.openstack.org/wiki/BasicDesignTenets>

- Scalability and elasticity are our main goals
- Any feature that limits our main goals must be optional
- Everything should be asynchronous. If you can't do something asynchronously, see #2
- All required components must be horizontally scalable
- Always use shared nothing architecture (SN) or sharding. If you can't share nothing/shard, see #2
- Distribute everything. Especially logic. Move logic to where state naturally exists.
- Accept eventual consistency and use it where it is appropriate.
- Test everything. We require tests with submitted code. (We will help you if you need it)



Arnaud Morin (Orange Labs)

Formation OpenStack

88 / 219

Implémentation

- Chaque sous-projet est découpé en plusieurs services
- Communication entre les services : AMQP (RabbitMQ)
- Base de données : MySQL
- Réutilisation de nombreux composants existants
- OpenVSwitch
- Tout est développé en Python (Django pour Horizon)
- APIs supportées : OpenStack et équivalent Amazon
- Multi tenancy

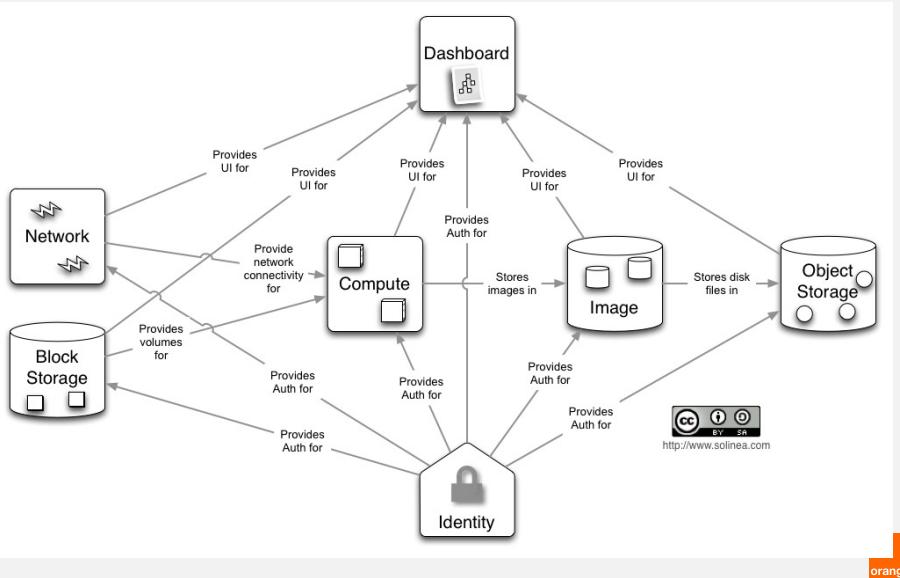


Multi-tenant

- Notion générale : un déploiement du logiciel permet de multiples utilisations
- Un cloud OpenStack permet aux utilisateurs de travailler dans des environnements isolés
- Les instances, réseaux, images, etc. sont associés à un tenant
- Certaines ressources peuvent être partagées entre tenants (exemple : image publique)
- On peut aussi parler de "projet"



Architecture



Interface web / Dashboard : Horizon

Screenshot of the Ubuntu OpenStack Dashboard. The top bar shows "ubuntu® openStack Dashboard" and "Logged in as admin". A success message says "Success: Launched instance named 'ubuntu'." The main area is titled "Instances" and shows a table of running instances:

Instance Name	IP Address	Size	Keypair	Status	Task	Power State	Actions
ubuntu	10.92.193.4	m1.small 2GB RAM 1 VCPU 10GB Disk	-	Active	None	Running	[Launch Instance] [Terminate Instances]
server	10.92.193.6	m1.tiny 512MB RAM 1 VCPU 0 Disk	macle	Active	None	Running	[Create Snapshot]
server1	10.92.193.3	m1.tiny 512MB RAM 1 VCPU 0 Disk	macle	Active	None	Running	[Create Snapshot]

The sidebar includes links for "Project", "Admin", "Manage Compute", "Overview", "Instances", "Volumes", "Images & Snapshots", "Access & Security", and "Networks". A footer note says "Caution: You are acting as an admin user in the project dashboard." and "Learn More".



Ressources

- <http://docs.openstack.org/>
- <https://ask.openstack.org>
- openstack@lists.openstack.org
- [#openstack@Freenode](#)
- <http://api.openstack.org/>
- Communauté française :
 - ▶ <http://openstack.fr/>
 - ▶ openstack-fr@lists.openstack.org
 - ▶ [#openstack-fr@Freenode](#)
 - ▶ Association



Utilité de DevStack

- Déployer rapidement un OpenStack
- Utilisé par les développeurs pour tester leurs changements
- Utilisé pour faire des démonstrations
- Utilisé pour tester les APIs sans se préoccuper du déploiement
- Ne doit PAS être utilisé pour la production



Plan

- ➊ Virtualisation
- ➋ Le Cloud : vue d'ensemble
- ➌ OpenStack : projet, logiciel et utilisation
 - Présentation du projet et du logiciel
 - DevStack : faire tourner rapidement un OpenStack
 - Utiliser OpenStack
- ➍ Déployer OpenStack
- ➎ Tirer partie de l'IaaS



Fonctionnement de DevStack

- Un script shell qui fait tout le travail : stack.sh
- Un fichier de configuration : local.conf
- Installe toutes les dépendances nécessaires (paquets)
- Clone les dépôts git (branche master par défaut)
- Lance tous les composants dans un screen



Configuration : local.conf

Exemple

```
[ [local|localrc] ]
ADMIN_PASSWORD=secrete
DATABASE_PASSWORD=$ADMIN_PASSWORD
RABBIT_PASSWORD=$ADMIN_PASSWORD
SERVICE_PASSWORD=$ADMIN_PASSWORD
SERVICE_TOKEN=a682f596-76f3-11e3-b3b2-e716f9080d50
#FIXED_RANGE=172.31.1.0/24
#FLOATING_RANGE=192.168.20.0/25
#HOST_IP=10.3.4.5
```



Plan

1 Virtualisation

2 Le Cloud : vue d'ensemble

3 OpenStack : projet, logiciel et utilisation

- Présentation du projet et du logiciel
- DevStack : faire tourner rapidement un OpenStack
- Utiliser OpenStack

4 Déployer OpenStack

5 Tirer partie de l'IaaS



Conseils d'utilisation

- DevStack installe beaucoup de choses sur la machine
- Il est recommandé de travailler dans une VM
- Pour tester tous les composants OpenStack dans de bonnes conditions, plusieurs Go de RAM sont nécessaires
- L'utilisation de Vagrant est conseillée



Le principe

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



Accès aux APIs

- Direct, en HTTP, via des outils comme curl
- Avec une bibliothèque
 - ▶ Les implémentations officielles en Python
 - ▶ D'autres implémentations pour d'autres langages (exemple : jclouds)
- Avec les outils officiels en ligne de commande
- Avec Horizon



Plan

- ➊ Virtualisation
- ➋ Le Cloud : vue d'ensemble
- ➌ OpenStack : projet, logiciel et utilisation

- ➍ Déployer OpenStack
 - Les briques nécessaires
 - Keystone : Authentification, autorisation et catalogue de services
 - Nova : Compute
 - Glance : Registre d'images
 - Neutron : Réseau en tant que service
 - Cinder : Stockage block
 - Horizon : Dashboard web
 - Swift : Stockage objet
 - Ceilometer : Collecte de métriques
 - Heat : Orchestration des ressources
 - Trove : Database as a Service
 - Designate : DNS as a Service
 - Quelques autres composants intéressants
 - Bonnes pratiques pour un déploiement en production
 - Faire face aux problèmes



Clients officiels

- Le projet fournit des clients officiels : python-PROJETclient
- 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

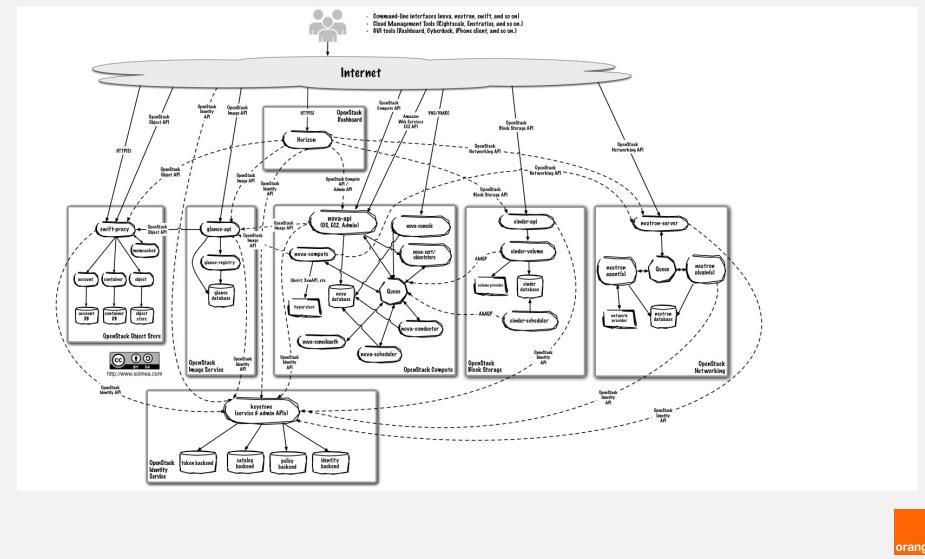


Ce qu'on va voir

- Installer OpenStack à la main <http://docs.openstack.org/icehouse/install-guide/install/apt/content/>
- Donc comprendre son fonctionnement
- Tour d'horizon des solutions de déploiement



Architecture détaillée



Quelques éléments de configuration généraux

- Tous les composants doivent être configurés pour communiquer avec Keystone
- La plupart doivent être configurés pour communiquer avec MySQL et RabbitMQ
- Les composants découpés en plusieurs services ont souvent un fichier de configuration par service
- api-paste.ini contient des paramètres concernant le service API

Plan

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

- Déployer OpenStack
 - Les briques nécessaires
 - Keystone : Authentification, autorisation et catalogue de services
 - Nova : Compute
 - Glance : Registry d'images
 - Neutron : Réseau en tant que service
 - Cinder : Stockage block
 - Horizon : Dashboard web
 - Swift : Stockage objet
 - Ceilometer : Collecte de métriques
 - Heat : Orchestration des ressources
 - Trove : Database as a Service
 - Designate : DNS as a Service
 - Quelques autres composants intéressants
 - Bonnes pratiques pour un déploiement en production
 - Faire face aux problèmes

- Tirer partie de l'IaaS

Système d'exploitation

- OS Linux avec Python
- Historiquement : Ubuntu
- Red Hat s'est largement rattrapé
- SUSE, etc.

Python



- OpenStack est aujourd'hui compatible Python 2.7
- Afin de ne pas réinventer la roue, beaucoup de dépendances sont nécessaires
- Un travail de portage vers Python 3 est en cours



Pourquoi l'utilisation de SQLAlchemy

SQLAlchemy

- Support de multiples BDD
- Gestion des migrations (changement de structure des données)



Base de données

- Permet de stocker la majorité des données gérées par OpenStack
- Chaque composant a sa propre base
- OpenStack utilise l'ORM Python SQLAlchemy
- Support théorique équivalent à celui de SQLAlchemy
- MySQL est l'implémentation la plus largement testée et utilisée
- SQLite est principalement utilisé dans le cadre de tests et démo
- Certains déploiements fonctionnent avec PostgreSQL



Passage de messages

- AMQP : Advanced Message Queuing Protocol
- Messages, file d'attente, routage
- Les processus OpenStack communiquent via AMQP
- Plusieurs implémentations possibles : Qpid, 0MQ, etc.
- RabbitMQ par défaut



RabbitMQ

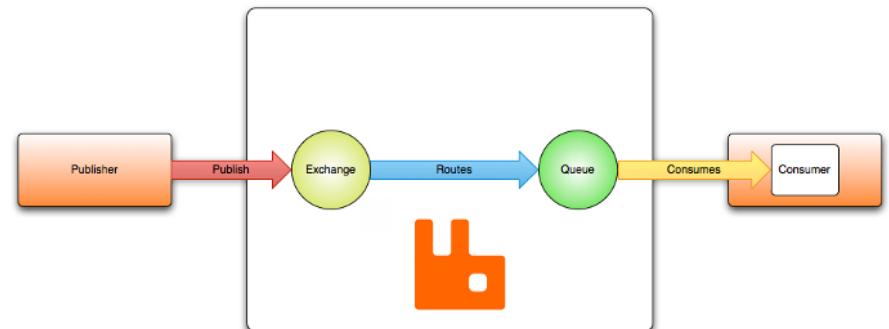


- RabbitMQ est implémenté en Erlang
- Une machine virtuelle Erlang est donc nécessaire



"Hello world" RabbitMQ

"Hello, world" example routing



Plan

- ➊ Virtualisation
- ➋ Le Cloud : vue d'ensemble
- ➌ OpenStack : projet, logiciel et utilisation
- ➍ Déployer OpenStack
 - Les briques nécessaires
 - Keystone : Authentification, autorisation et catalogue de services
 - Nova : Compute
 - Glance : Registre d'images
 - Neutron : Réseau en tant que service
 - Cinder : Stockage block
 - Horizon : Dashboard web
 - Swift : Stockage objet
 - Ceilometer : Collecte de métriques
 - Heat : Orchestration des ressources
 - Trove : Database as a Service
 - Designate : DNS as a Service
 - Quelques autres composants intéressants
 - Bonnes pratiques pour un déploiement en production
 - Faire face aux problèmes
- ➎ Tirer partie de l'IaaS



Principes

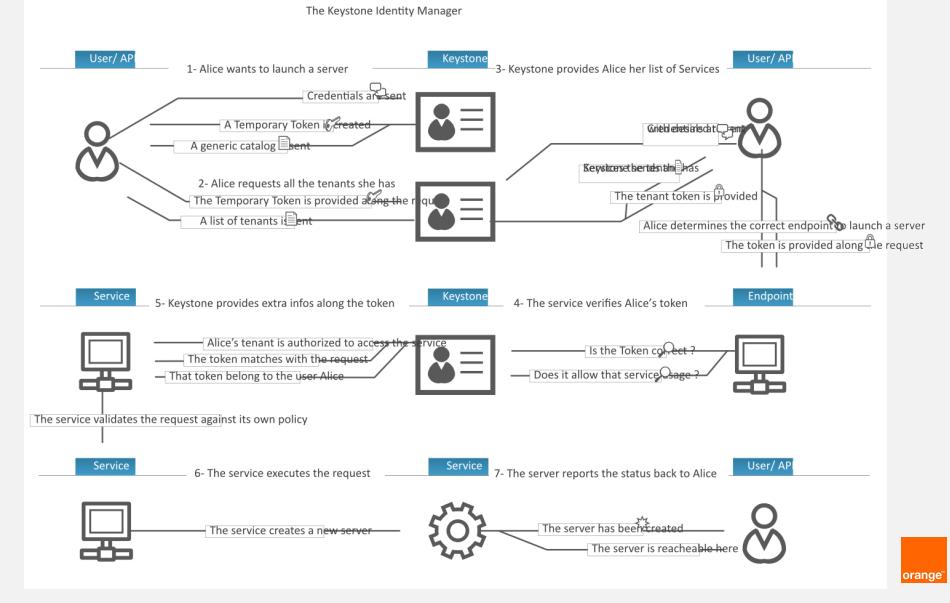
- Annuaire des utilisateurs et des groupes
- Catalogue de services
- Gère l'authentification et l'autorisation
- Support des domaines dans l'API v3
- Fournit un token à l'utilisateur



- API admin : port 35357
- API utilisateur : port 5000
- Deux versions : v2 (actuelle) et v3 (future)
- Gère *utilisateurs, groupes, domaines* (APIv3)
- Les utilisateurs ont des *rôles* sur des *tenants* (projets)



Scénario d'utilisation typique



Installation et configuration

- Paquet : keystone
- Configuration d'un token ADMIN pour la configuration initiale
- Backends : choix de SQL ou LDAP (ou AD)
- Configurer les différents services
- Policy.json
- Services et endpoints
- Utilisateurs, groupes, domaines



Enregistrer un service et son endpoint

Il faut renseigner l'existence des différents services (catalogue) dans Keystone :

```
$ keystone service-create --name=cinderv2 --type=volumev2 \
--description="Cinder Volume Service V2"
$ keystone endpoint-create \
--region=myRegion
--service-id=...
--publicurl=http://controller:8776/v2/%\(\tenant_id\)\s \
--internalurl=http://controller:8776/v2/%\(\tenant_id\)\s \
--adminurl=http://controller:8776/v2/%\(\tenant_id\)\s
```



Tester

```
$ keystone service-list  
...  
$ keystone user-list  
...  
$ keystone token-get  
...
```



Principes

- Gère les instances
- IP flottantes, groupes de sécurité
- Les instances sont créées à partir des images fournies par Glance
- Les interfaces réseaux des instances sont associées à des ports Neutron

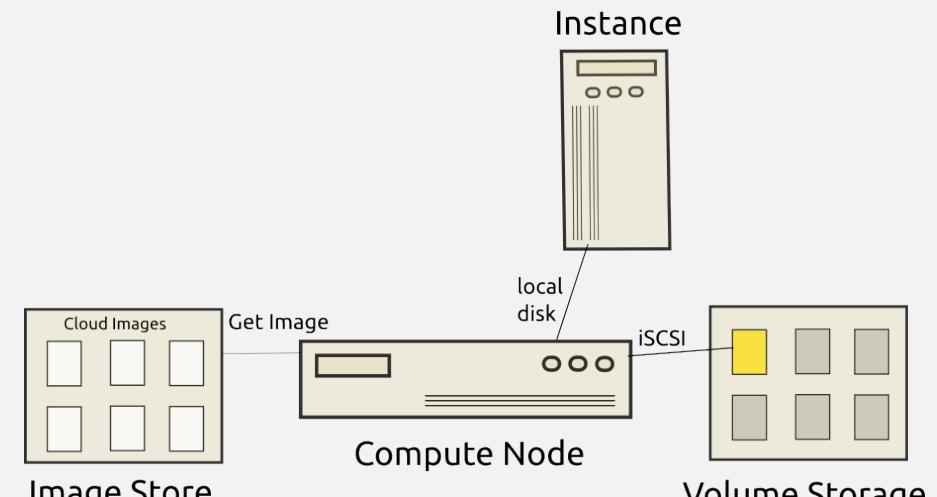


Plan

- ➊ Virtualisation
- ➋ Le Cloud : vue d'ensemble
- ➌ OpenStack : projet, logiciel et utilisation
- ➍ Déployer OpenStack
 - ➌ Les briques nécessaires
 - ➌ Keystone : Authentification, autorisation et catalogue de services
 - ➌ Nova : Compute
 - ➌ Glance : Registre d'images
 - ➌ Neutron : Réseau en tant que service
 - ➌ Cinder : Stockage block
 - ➌ Horizon : Dashboard web
 - ➌ Swift : Stockage objet
 - ➌ Ceilometer : Collecte de métriques
 - ➌ Heat : Orchestration des ressources
 - ➌ Trove : Database as a Service
 - ➌ Designate : DNS as a Service
 - ➌ Quelques autres composants intéressants
 - ➌ Bonnes pratiques pour un déploiement en production
 - ➌ Faire face aux problèmes
- ➎ Tirer partie de l'IaaS



Interactions avec les autres composants



Gère :

- Instances
- Flavors (types d'instance)
- Security Groups (groupes de sécurité)
- Floating IPs (IPs flottantes)

Les instances sont redimensionnables et migrables d'un hôte physique à un autre.



Les groupes de sécurité

- Équivalent à un firewall devant chaque instance
- Une instance peut être associée à un ou plusieurs groupes de sécurité
- Gestion des accès en entrée et sortie
- Règles par protocole (TCP/UDP/ICMP) et par port



Rôle des flavors

- Équivalent des "instance types" d'AWS
- Définit un modèle d'instance en termes de CPU, RAM, disque



Nova api

- Double rôle
- API de manipulation des instances par l'utilisateur
- API à destination des instances : API de metadata
- L'API de metadata doit être accessible à l'adresse `http://169.254.169.254/`
- L'API de metadata fournit des informations de configuration personnalisées à chacune des instances



Nova compute

- Exécute les machines virtuelles
- Tire partie de libvirt ou d'autres APIs comme XenAPI
- Drivers : libvirt, XenAPI, VMWare ESX, Docker
- Permet de récupérer les logs de la console et une console VNC

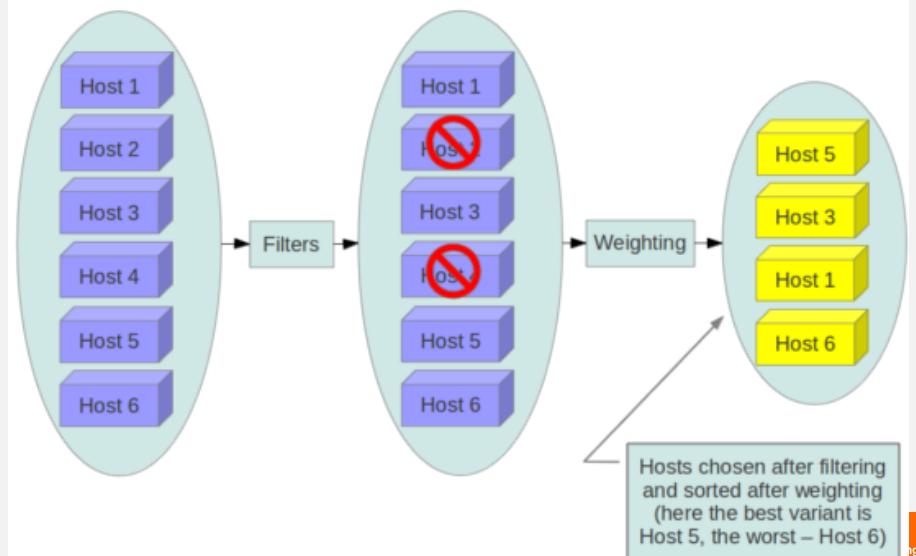


Nova scheduler

- Service qui distribue les demandes d'instances sur les noeuds compute
- Filter, Chance, Multi Scheduler
- Filtres, par défaut : AvailabilityZoneFilter, RamFilter, ComputeFilter
- Tri par poids, par défaut : RawWeigher



Le scheduler Nova en action



Nova conductor

- Service facultatif qui améliore la sécurité
- Fait office de proxy entre les noeuds compute et la BDD
- Les noeuds compute, vulnérables, n'ont donc plus d'accès à la BDD



Tester

```
$ nova list  
...  
$ nova create  
...
```



Principes

- Registre d'images (et des snapshots)
- Propriétés sur les images
- Est utilisé par Nova pour démarrer des instances
- Peut utiliser Swift comme back-end de stockage



Plan

- ➊ Virtualisation
- ➋ Le Cloud : vue d'ensemble
- ➌ OpenStack : projet, logiciel et utilisation
- ➍ Déployer OpenStack
 - Les briques nécessaires
 - Keystone : Authentification, autorisation et catalogue de services
 - Nova : Compute
 - Glance : Registre d'images
 - Neutron : Réseau en tant que service
 - Cinder : Stockage block
 - Horizon : Dashboard web
 - Swift : Stockage objet
 - Ceilometer : Collecte de métriques
 - Heat : Orchestration des ressources
 - Trove : Database as a Service
 - Designate : DNS as a Service
 - Quelques autres composants intéressants
 - Bonnes pratiques pour un déploiement en production
 - Faire face aux problèmes
- ➎ Tirer partie de l'IaaS



Propriétés des images dans Glance

L'utilisateur peut définir un certain nombre de propriétés dont certaines seront utilisées lors de l'instanciation

- Type d'image
- Architecture
- Distribution
- Version de la distribution
- Espace disque minimum
- RAM minimum
- Publique ou non



Types d'images

Glance supporte un large éventail de types d'images, limité par le support de l'hyperviseur sous-jacent à Nova

- raw
- qcow2
- ami
- vmdk
- iso



Installation

- Paquet glance-api : fournit l'API
- Paquet glance-registry : démon du registre d'images en tant que tel



Backends

- Swift ou S3
- Ceph
- HTTP
- Répertoire local



Tester

```
$ glance image-list  
...  
$ glance image-create  
...
```



Plan

- ① Virtualisation
- ② Le Cloud : vue d'ensemble
- ③ OpenStack : projet, logiciel et utilisation
- ④ Déployer OpenStack
 - Les briques nécessaires
 - Keystone : Authentification, autorisation et catalogue de services
 - Nova : Compute
 - Glance : Registre d'images
 - Neutron : Réseau en tant que service
 - Cinder : Stockage block
 - Horizon : Dashboard web
 - Swift : Stockage objet
 - Ceilometer : Collecte de métriques
 - Heat : Orchestration des ressources
 - Trove : Database as a Service
 - Designate : DNS as a Service
 - Quelques autres composants intéressants
 - Bonnes pratiques pour un déploiement en production
 - Faire face aux problèmes
- ⑤ Tirer partie de l'IaaS



Arnaud Morin (Orange Labs)

Formation OpenStack

141 / 219

Fonctionnalités supplémentaires

Outre les fonctions réseau de base niveaux 2 et 3, Neutron peut fournir d'autres services :

- Load Balancing (HAProxy, ...)
- Firewall (vArmour, ...) : diffère des groupes de sécurité
- VPN (Openswan, ...) : permet d'accéder à un réseau privé sans IP flottantes

Ces fonctionnalités se basent également sur des plugins



Arnaud Morin (Orange Labs)

Formation OpenStack

143 / 219

Principes

- SDN
- Auparavant Quantum et nova-network
- neutron-server : fournit l'API
- Agent DHCP : fournit le service de DHCP pour les instances
- Agent L3 : gère la couche 3 du réseau, le routage
- Plugin : OpenVSwitch par défaut, d'autres implémentations libres/propriétaires, logicielles/matérielles existent



Arnaud Morin (Orange Labs)

Formation OpenStack

142 / 219

API

L'API permet notamment de manipuler ces ressources

- Réseau : niveau 2
- Subnet : niveau 3
- Port

Les ports peuvent correspondre à des interfaces d'instance



Arnaud Morin (Orange Labs)

Formation OpenStack

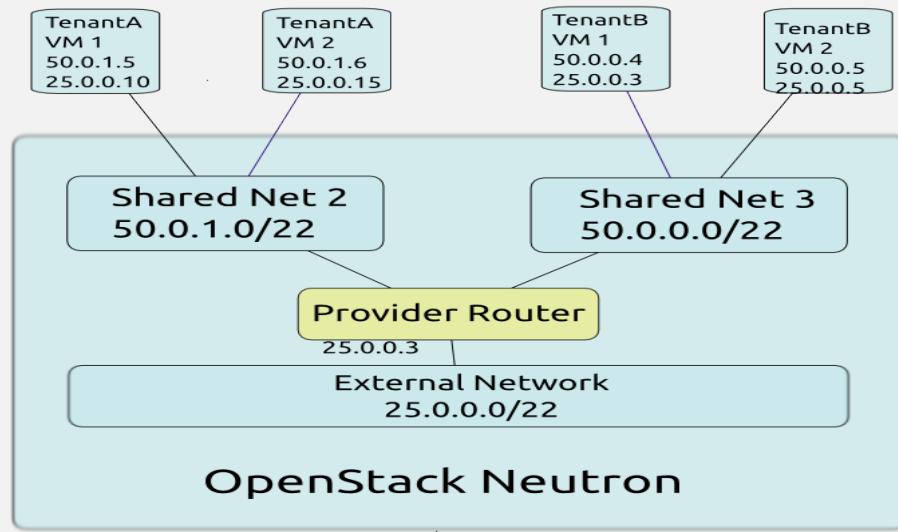
144 / 219

Plugins ML2

- Modular Layer 2
- OpenVSwitch
- OpenDaylight
- Contrail, OpenContrail
- Nuage Networks
- cf. OpenFlow



Schéma



Implémentation

- Neutron tire partie des namespaces réseaux du noyau Linux pour permettre l'IP overlapping
- Le proxy de metadata est un composant qui permet aux instances isolées dans leur réseau de joindre l'API de metadata fournie par Nova



Plan

- ➊ Virtualisation
- ➋ Le Cloud : vue d'ensemble
- ➌ OpenStack : projet, logiciel et utilisation
- ➍ Déployer OpenStack
 - Les briques nécessaires
 - Keystone : Authentification, autorisation et catalogue de services
 - Nova : Compute
 - Glance : Registre d'images
 - Neutron : Réseau en tant que service
 - Cinder : Stockage block
 - Horizon : Dashboard web
 - Swift : Stockage objet
 - Ceilometer : Collecte de métriques
 - Heat : Orchestration des ressources
 - Trove : Database as a Service
 - Designate : DNS as a Service
 - Quelques autres composants intéressants
 - Bonnes pratiques pour un déploiement en production
 - Faire face aux problèmes
- ➎ Tirer partie de l'IaaS



Principes

- Auparavant nova-volume
- Fournit des volumes (stockage block) attachables aux instances
- Gère différents types de volume
- Gère snapshots et backups de volumes
- Attachement via iSCSI par défaut

Du stockage partagé ?

- Cinder n'est **pas** une solution de stockage partagé comme NFS
- OpenStack (tout comme AWS) ne fournit pas de solution *NFS as a Service*
- Cf. le projet Manila



Utilisation

- Volume supplémentaire (et stockage persistant) sur une instance
- Boot from volume : l'OS est sur le volume
- Fonctionnalité de backup vers un object store (Swift ou Ceph)

Installation

- Paquet cinder-api : fournit l'API
- Paquet cinder-volume : création et gestion des volumes
- Paquet cinder-scheduler : distribue les demandes de création de volume
- Paquet cinder-backup : backup vers un object store



Backends

- Utilisation de plusieurs backends en parallèle possible
- LVM (par défaut)
- GlusterFS
- Ceph
- Systèmes de stockage propriétaires type NetApp



Principes

- Utilise les APIs existantes pour fournir une interface
- Horizon est un module Django
- OpenStack Dashboard est l'implémentation officielle de ce module



Plan

- ➊ Virtualisation
- ➋ Le Cloud : vue d'ensemble
- ➌ OpenStack : projet, logiciel et utilisation
- ➍ Déployer OpenStack
 - ➌ Les briques nécessaires
 - ➌ Keystone : Authentification, autorisation et catalogue de services
 - ➌ Nova : Compute
 - ➌ Glance : Registre d'images
 - ➌ Neutron : Réseau en tant que service
 - ➌ Cinder : Stockage block
 - ➌ Horizon : Dashboard web
 - ➌ Swift : Stockage objet
 - ➌ Ceilometer : Collecte de métriques
 - ➌ Heat : Orchestration des ressources
 - ➌ Trove : Database as a Service
 - ➌ Designate : DNS as a Service
 - ➌ Quelques autres composants intéressants
 - ➌ Bonnes pratiques pour un déploiement en production
 - ➌ Faire face aux problèmes
- ➎ Tirer partie de l'IaaS



Configuration

- local_settings.py
- Les services apparaissent dans Horizon s'ils sont répertoriés dans le catalogue de services de Keystone



Utilisation

- Une zone "admin" restreinte
- Une interface par tenant



Principes

- SDS : Software Defined Storage
- Utilisation de commodity hardware
- Théorème CAP : on en choisit deux
- Accès par les APIs
- Architecture totalement acentrée
- Pas de base de données centrale



Plan

- ➊ Virtualisation
- ➋ Le Cloud : vue d'ensemble
- ➌ OpenStack : projet, logiciel et utilisation
- ➍ Déployer OpenStack
 - Les briques nécessaires
 - Keystone : Authentification, autorisation et catalogue de services
 - Nova : Compute
 - Glance : Registre d'images
 - Neutron : Réseau en tant que service
 - Cinder : Stockage block
 - Horizon : Dashboard web
 - Swift : Stockage objet
 - Ceilometer : Collecte de métriques
 - Heat : Orchestration des ressources
 - Trove : Database as a Service
 - Designate : DNS as a Service
 - Quelques autres composants intéressants
 - Bonnes pratiques pour un déploiement en production
 - Faire face aux problèmes
- ➎ Tirer partie de l'IaaS



Implémentation

- Proxy : serveur API par lequel passent toutes les requêtes
- Object server : serveur de stockage
- Container server : maintient des listes d'objects dans des containers
- Account server : maintient des listes de containers dans des accounts
- Chaque objet est répliqué n fois (3 par défaut)

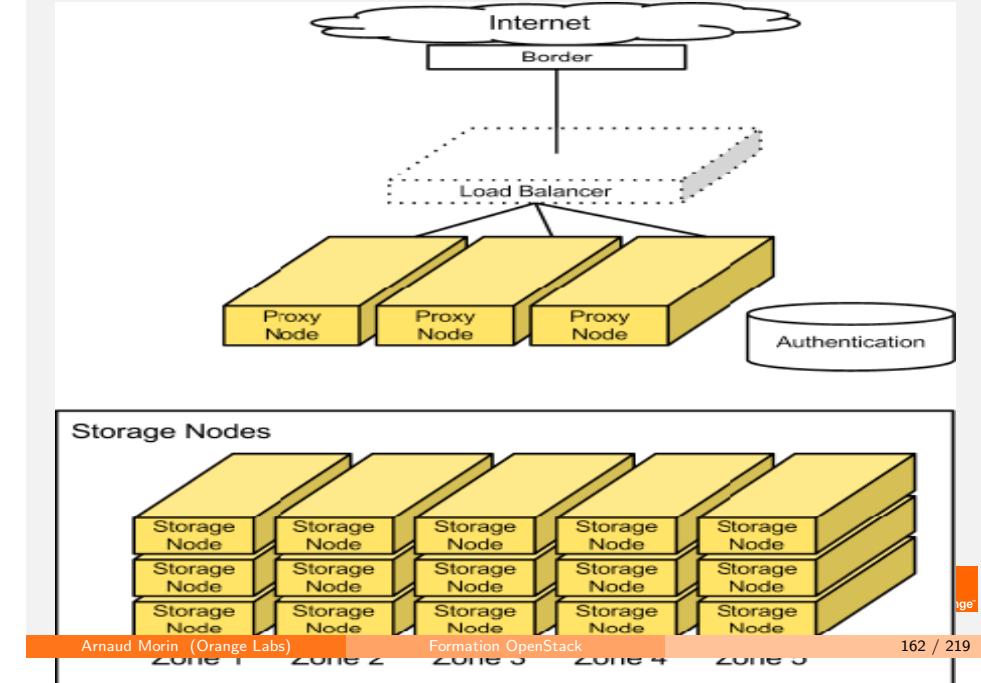


Le ring

- Problème : comment décider quelle donnée va sur quel object server
- Le ring est découpé en partitions
- On situe chaque donnée dans le ring afin de déterminer sa partition
- Une partition est associée à plusieurs serveurs



Schéma



Plan

- ➊ Virtualisation
- ➋ Le Cloud : vue d'ensemble
- ➌ OpenStack : projet, logiciel et utilisation
- ➍ Déployer OpenStack
 - Les briques nécessaires
 - Keystone : Authentification, autorisation et catalogue de services
 - Nova : Compute
 - Glance : Registre d'images
 - Neutron : Réseau en tant que service
 - Cinder : Stockage block
 - Horizon : Dashboard web
 - Swift : Stockage objet
- ➎ Ceilometer : Collecte de métriques
 - Heat : Orchestration des ressources
 - Trove : Database as a Service
 - Designate : DNS as a Service
 - Quelques autres composants intéressants
 - Bonnes pratiques pour un déploiement en production
 - Faire face aux problèmes
- ➏ Tirer partie de l'IaaS



Surveiller l'utilisation de son infrastructure avec Ceilometer

- Indexe différentes métriques concernant l'utilisation des différents services du cloud
- Fournit des APIs permettant de récupérer ces données
- Base pour construire des outils de facturation
- Utilise MongoDB (par défaut) pour le stockage



Plan

- ① Virtualisation
- ② Le Cloud : vue d'ensemble
- ③ OpenStack : projet, logiciel et utilisation
- ④ Déployer OpenStack
 - Les briques nécessaires
 - Keystone : Authentification, autorisation et catalogue de services
 - Nova : Compute
 - Glance : Registre d'images
 - Neutron : Réseau en tant que service
 - Cinder : Stockage block
 - Horizon : Dashboard web
 - Swift : Stockage objet
 - Ceilometer : Collecte de métriques
 - Heat : Orchestration des ressources
 - Trove : Database as a Service
 - Designate : DNS as a Service
 - Quelques autres composants intéressants
 - Bonnes pratiques pour un déploiement en production
 - Faire face aux problèmes
- ⑤ Tirer partie de l'IaaS



Arnaud Morin (Orange Labs)

Formation OpenStack

165 / 219

Autoscaling avec Heat

Heat implémente la fonctionnalité d'autoscaling

- Se déclenche en fonction des alarmes produites par Ceilometer
- Entraîne la création de nouvelles instances



Arnaud Morin (Orange Labs)

Formation OpenStack

167 / 219

Orchestrer son infrastructure avec Heat

- Équivalent d'Amazon Cloud Formation
- Orchestrer les ressources compute, storage, network, etc.
- Doit se coupler avec cloud-init
- Description de son infrastructure dans un fichier template, format JSON (CFN) ou YAML (HOT)



Arnaud Morin (Orange Labs)

Formation OpenStack

166 / 219

Un template HOT

```
heat_template_version: 2013-05-23

description: Simple template to deploy a single compute instance

resources:
  my_instance:
    type: OS::Nova::Server
    properties:
      key_name: my_key
      image: F18-x86_64-cfntools
      flavor: m1.small
```



Arnaud Morin (Orange Labs)

Formation OpenStack

168 / 219

Fonctionnalités avancées de Heat

- Nested stacks
- Environments
- Providers



Principe

- Fournit des bases de données relationnelles, à la AWS RDS
- A vocation à supporter des bases NoSQL aussi
- Gère notamment MySQL comme back-end
- Se repose sur Nova pour les instances dans lesquelles se fait l'installation d'une BDD



Plan

- ➊ Virtualisation
- ➋ Le Cloud : vue d'ensemble
- ➌ OpenStack : projet, logiciel et utilisation

➍ Déployer OpenStack

- Les briques nécessaires
- Keystone : Authentification, autorisation et catalogue de services
- Nova : Compute
- Glance : Registre d'images
- Neutron : Réseau en tant que service
- Cinder : Stockage block
- Horizon : Dashboard web
- Swift : Stockage objet
- Ceilometer : Collecte de métriques
- Heat : Orchestration des ressources
- Trove : Database as a Service
- Designate : DNS as a Service
- Quelques autres composants intéressants
- Bonnes pratiques pour un déploiement en production
- Faire face aux problèmes

➎ Tirer partie de l'IaaS

Services

- trove-api : API
- trove-taskmanager : gère les instances BDD
- trove-guestagent : agent interne à l'instance



Plan

- ➊ Virtualisation
- ➋ Le Cloud : vue d'ensemble
- ➌ OpenStack : projet, logiciel et utilisation

- ➍ Déployer OpenStack
 - Les briques nécessaires
 - Keystone : Authentification, autorisation et catalogue de services
 - Nova : Compute
 - Glance : Registre d'images
 - Neutron : Réseau en tant que service
 - Cinder : Stockage block
 - Horizon : Dashboard web
 - Swift : Stockage objet
 - Ceilometer : Collecte de métriques
 - Heat : Orchestration des ressources
 - Trove : Database as a Service
 - Designate : DNS as a Service
 - Quelques autres composants intéressants
 - Bonnes pratiques pour un déploiement en production
 - Faire face aux problèmes



- ➎ Tirer partie de l'IaaS

Arnaud Morin (Orange Labs)

Formation OpenStack

173 / 219



Principe

- Équivalent d'AWS Route 53
- Gère différents backends : BIND, etc.

Plan

- ➊ Virtualisation
- ➋ Le Cloud : vue d'ensemble
- ➌ OpenStack : projet, logiciel et utilisation

- ➍ Déployer OpenStack
 - Les briques nécessaires
 - Keystone : Authentification, autorisation et catalogue de services
 - Nova : Compute
 - Glance : Registre d'images
 - Neutron : Réseau en tant que service
 - Cinder : Stockage block
 - Horizon : Dashboard web
 - Swift : Stockage objet
 - Ceilometer : Collecte de métriques
 - Heat : Orchestration des ressources
 - Trove : Database as a Service
 - Designate : DNS as a Service
 - Quelques autres composants intéressants
 - Bonnes pratiques pour un déploiement en production
 - Faire face aux problèmes



- ➎ Tirer partie de l'IaaS

Arnaud Morin (Orange Labs)

Formation OpenStack

175 / 219



Ironic

- Anciennement Nova bare-metal
- Permet le déploiement d'instance sur des machines physiques (plutôt que VMs)
- Repose sur des technologies telles que PXE, TFTP



Arnaud Morin (Orange Labs)

Formation OpenStack

176 / 219

Oslo, ou OpenStack common

- Oslo contient le code commun à plusieurs composants d'OpenStack
- Son utilisation est transparente pour le déployeur

rootwrap

- Wrapper pour l'utilisation de commandes en root
- Configuration au niveau de chaque composant qui l'utilise
- Permet d'écrire des filtres sur les commandes



TripleO

- OpenStack On OpenStack
- Objectif : pouvoir déployer un cloud OpenStack (overcloud) à partir d'un cloud OpenStack (undercloud)
- Autoscaling du cloud lui-même : déploiement de nouveaux noeuds compute lorsque cela est nécessaire
- Fonctionne conjointement avec Ironic pour le déploiement bare-metal

Tempest

- Suite de tests d'un cloud OpenStack
- Effectue des appels à l'API et vérifie le résultat
- Est très utilisé par les développeurs via l'intégration continue
- Le déployeur peut utiliser Tempest pour vérifier la bonne conformité de son cloud



Plan

- ① Virtualisation
- ② Le Cloud : vue d'ensemble
- ③ OpenStack : projet, logiciel et utilisation
- ④ Déployer OpenStack
 - Les briques nécessaires
 - Keystone : Authentification, autorisation et catalogue de services
 - Nova : Compute
 - Glance : Registre d'images
 - Neutron : Réseau en tant que service
 - Cinder : Stockage block
 - Horizon : Dashboard web
 - Swift : Stockage objet
 - Ceilometer : Collecte de métriques
 - Heat : Orchestration des ressources
 - Trove : Database as a Service
 - Designate : DNS as a Service
 - Quelques autres composants intéressants
- ⑤ Bonnes pratiques pour un déploiement en production
 - Faire face aux problèmes
- ⑥ Tirer partie de l'IaaS



Arnaud Morin (Orange Labs)

Formation OpenStack

181 / 219

Penser dès le début aux choix structurants

- Distribution et méthode de déploiement
- Hyperviseur
- Réseau : quelle architecture et quels drivers
- Politique de mise à jour



Arnaud Morin (Orange Labs)

Formation OpenStack

183 / 219

Quels composants dois-je installer ?

- Keystone est indispensable
- L'utilisation de Nova va de paire avec Glance et Neutron
- Cinder s'avérera souvent utile
- Ceilometer et Heat vont souvent ensemble
- Swift est indépendant des autres composants



Arnaud Morin (Orange Labs)

Formation OpenStack

182 / 219

Les différentes méthodes d'installation

- DevStack est à oublier pour la production
- TripleO est encore trop jeune
- Le déploiement à la main comme vu précédemment n'est pas recommandé car peu maintenable
- Distributions OpenStack packagées et prêtées à l'emploi
- Distributions classiques et gestion de configuration



Arnaud Morin (Orange Labs)

Formation OpenStack

184 / 219

Assigner des rôles aux machines

Beaucoup de documentations font référence à ces rôles :

- Controller node : APIs, BDD, AMQP
- Network node : Routeur
- Compute node : Hyperviseur

Ce modèle simplifié n'est pas HA.



Haute disponibilité de l'agent L3 de Neutron

- Plusieurs solutions possibles



Haute disponibilité

Haut disponibilité de l'IaaS

- MySQL, RabbitMQ : HA classique (Galera, Clustering)
- Les services APIs sont stateless et HTTP : scale out et load balancers
- La plupart des autres services OpenStack sont capables de scale out également

Guide HA :

<http://docs.openstack.org/high-availability-guide/content/>



Considérations pour une environnement de production

- Des URLs uniformes pour toutes les APIs : utiliser un reverse proxy
- Utilisation des quotas
- Prévoir les bonnes volumétries, notamment pour les données Ceilometer
- Monitoring

Guide Operations :

<http://docs.openstack.org/trunk/openstack-ops/content/>



Découpage réseau

- Management network : réseau d'administration
- Data network : réseau pour la communication inter instances
- External network : réseau externe, dans l'infrastructure réseau existante
- API network : réseau contenant les endpoints API



Considérations liées à la sécurité

- Indispensable : HTTPS sur l'accès des APIs à l'extérieur
- Sécurisation des communications MySQL et RabbitMQ
- Un accès MySQL par base et par service
- Un utilisateur Keystone par service
- Limiter l'accès en lecture des fichiers de configuration (mots de passe, token)



Segmenter son cloud

- Host aggregates : machines physiques avec des caractéristiques similaires
- Availability zones : machines dépendantes d'une même source électrique, d'un même switch, d'un même DC, etc.
- Regions : chaque région a son API
- Cells : permet de regrouper plusieurs cloud différents sous une même API



Host aggregates / agrégats d'hôtes

- L'administrateur définit des agrégats via l'API
- 1 agrégat ≡ 1 point commun, ex : GPU
- L'utilisateur peut choisir un agrégat à la création d'instance



Availability zones / zones de disponibilité

- Groupe d'hôtes
- Découpage en termes de disponibilité : Rack, Datacenter, etc.
- L'utiliser peut choisir une zone de disponibilité
- L'utilisateur peut demander à ce que des instances soient démarrées dans une même zone, ou au contraire dans des zones différentes

Régions

- Équivalent des régions d'AWS
- Un service peut avoir différents endpoints dans différentes régions
- Chaque région est autonome
- Cas d'usage : cloud de grande ampleur (comme certains clouds publics)

Cellules / Cells

- Fonctionnalité de Nova uniquement
- Un seul nova-api devant plusieurs cellules
- Chaque cellule a sa propre BDD et file de messages
- Ajoute un niveau de scheduling (choix de la cellule)

Packaging d'OpenStack - Ubuntu

- Le packaging est fait dans de multiples distributions, RPM, DEB et autres
- Ubuntu est historiquement la plateforme de référence pour le développement d'OpenStack
- Le packaging dans Ubuntu suit de près le développement d'OpenStack, et des tests automatisés sont réalisés
- Canonical fournit la Ubuntu Cloud Archive, qui met à disposition la dernière version d'OpenStack pour la dernière Ubuntu LTS

Openstack

LTS support 5 years

12.04 12.10 13.04 13.10 14.04

Ubuntu 12.04 LTS

ESSEX

FOLSOM

GRIZZLY

HAVANA

- OpenStack est intégré dans les dépôts officiels de Debian
- Red Hat propose RHOS
- Comme Ubuntu, le cycle de release de Fedora est synchronisé avec celui d'OpenStack

Arnaud Morin (Orange Labs)

Formation OpenStack

197 / 219

Arnaud Morin (Orange Labs)

Formation OpenStack

198 / 219

Les distributions OpenStack

- StackOps
- Mirantis
- HP
- etc.

Déploiement bare metal

- Le déploiement des hôtes physiques OpenStack peut se faire à l'aide d'outils dédiés
- Canonical/Ubuntu propose MaaS : Metal as a Service
- Dell propose Crowbar
- eDeploy (eNovance)

Gestion de configuration

- Puppet, Chef, CFEngine, Saltstack, Ansible, etc.
- Ces outils peuvent aider à déployer le cloud OpenStack
- ... mais aussi à gérer les instances (section suivante)



Modules Puppet

- PuppetLabs maintient (avec d'autres) des modules pour déployer OpenStack
- <https://forge.puppetlabs.com/puppetlabs/openstack>



Plan

- ➊ Virtualisation
- ➋ Le Cloud : vue d'ensemble
- ➌ OpenStack : projet, logiciel et utilisation
- ➍ Déployer OpenStack
 - Les briques nécessaires
 - Keystone : Authentification, autorisation et catalogue de services
 - Nova : Compute
 - Glance : Registre d'images
 - Neutron : Réseau en tant que service
 - Cinder : Stockage block
 - Horizon : Dashboard web
 - Swift : Stockage objet
 - Ceilometer : Collecte de métriques
 - Heat : Orchestration des ressources
 - Trove : Database as a Service
 - Designate : DNS as a Service
 - Quelques autres composants intéressants
 - Bonnes pratiques pour un déploiement en production
 - Faire face aux problèmes
- ➎ Tirer partie de l'IaaS



Les réflexes en cas d'erreur ou de comportement erroné

- Travaille-t-on sur le bon tenant ?
- Est-ce que l'API renvoie une erreur ? (le dashboard peut cacher certaines informations)
- Si nécessaire d'aller plus loin :
 - ▶ Regarder les logs sur le cloud controller (`/var/log/<composant>/*.log`)
 - ▶ Regarder les logs sur le compute node et le network node si le problème est spécifique réseau/instance
 - ▶ Éventuellement modifier la verbosité des logs dans la configuration



Est-ce un bug ?

- Si le client CLI crash, c'est un bug
- Si le dashboard web affiche une erreur 500, c'est peut-être un bug
- Si les logs montrent une stacktrace Python, c'est un bug
- Sinon, à vous d'en juger



Deux visions

Une fois un cloud IaaS en place, deux optiques possibles :

- Garder les mêmes pratiques tout en profitant du self service et de l'agilité de la solution
- Faire évoluer ses pratiques, tant côté applicatif que système

"Pets vs Cattle"



Plan

- ➊ Virtualisation
- ➋ Le Cloud : vue d'ensemble
- ➌ OpenStack : projet, logiciel et utilisation
- ➍ Déployer OpenStack
- ➎ Tirer partie de l'IaaS
 - Côté applications
 - Côté système



Plan

- ➊ Virtualisation
- ➋ Le Cloud : vue d'ensemble
- ➌ OpenStack : projet, logiciel et utilisation
- ➍ Déployer OpenStack
- ➎ Tirer partie de l'IaaS
 - Côté applications
 - Côté système



Adapter ou faire ses applications "cloud ready"

- Stateless : permet de multiplier les routes d'accès à l'application
- Ne pas stocker les données en local, mais plutôt :
 - ▶ Base de données
 - ▶ Stockage objet
- Utiliser des outils standards de journalisation



Plan

- ➊ Virtualisation
- ➋ Le Cloud : vue d'ensemble
- ➌ OpenStack : projet, logiciel et utilisation
- ➍ Déployer OpenStack
- ➎ Tirer partie de l'IaaS
 - Côté applications
 - Côté système



Adopter une philosophie DevOps

- Infrastructure as Code
- Scale out au lieu de scale up
- HA niveau application plutôt qu'infrastructure



Utiliser des images cloud

Une image cloud c'est :

- Une image disque contenant un OS déjà installé
- Une image qui peut être instanciée en n machines sans erreur
- Un OS sachant parler à l'API de metadata du cloud (cloud-init)

La plupart des distributions fournissent aujourd'hui des images cloud.



Cirros

- Cirros est l'image cloud par excellence
- OS minimaliste
- Contient cloud-init
- <https://launchpad.net/cirros>



Faire ou modifier une image cloud

- Utilisation de libguestfs



Cloud-init

- Cloud-init est un moyen de tirer partie de l'API de metadata, et notamment des user data
- L'outil est intégré par défaut dans la plupart des images cloud
- À partir des user data, cloud-init effectue les opérations de personnalisation de l'instance
- cloud-config est un format possible de user data



Exemple cloud-config

```
#cloud-config
mounts:
  - [ xvdc, /var/www ]
packages:
  - apache2
  - htop
```



Configurer et orchestrer ses instances

- Outils de gestion de configuration (les mêmes qui permettent de déployer OpenStack)
- Juju

Conclusion

- Le cloud révolutionne l'IT
- OpenStack est le projet libre phare sur la partie IaaS
- Déployer OpenStack n'est pas une mince affaire
- L'utilisation d'un cloud IaaS implique des changements de pratique



Références

- http://www.antoinebenkemoun.fr/data/PPT_AC.pdf

