

Formation OpenStack

Ihab ABADI / UTOPIOS

Cloud Vue d'ensemble (Caractéristiques)

Un cloud fournit un ou plusieurs services...

- En self Service
- À travers le réseau
- En mutualisation des ressources
- Avec une élasticité
- Et mesurabilité

Cloud Vue d'ensemble (Self service)

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

Cloud Vue d'ensemble (À travers le réseau)

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

Cloud Vue d'ensemble (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

Cloud Vue d'ensemble (Élasticité)

- 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

Cloud Vue d'ensemble (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

Cloud Vue d'ensemble– Modèles de service

- Le Cloud permet de fournir plusieurs modèles de service
- IaaS
- PaaS
- SaaS
- ...

Cloud Vue d'ensemble – IaaS

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

Cloud Vue d'ensemble – 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

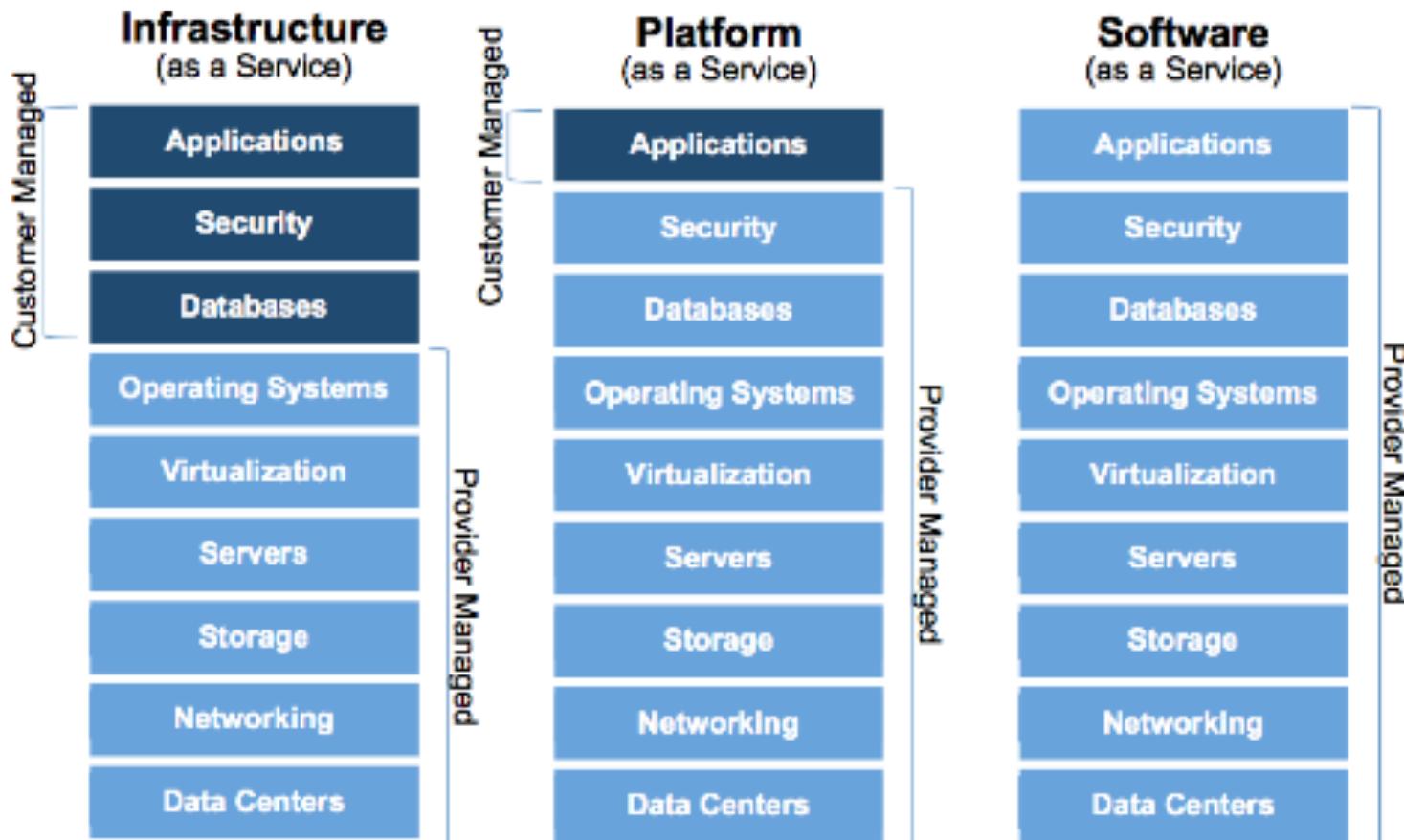
Cloud Vue d'ensemble – SaaS

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

Cloud Vue d'ensemble – Autre

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

Cloud Vue d'ensemble– Modèles de service



Cloud Vue d'ensemble– Modèles de déploiement

- Le Cloud peut être déployer en :
- Public
- Privé
- Hybride

Cloud Vue d'ensemble– Public et privé

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

Cloud Vue d'ensemble – 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)

Cloud Vue d'ensemble – Virtualisation

- 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)

Cloud Vue d'ensemble – API

- 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)

Cloud Vue d'ensemble – 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

Cloud Vue d'ensemble – Pourquoi le cloud

- 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

Cloud Vue d'ensemble – Pourquoi le cloud

- 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”

Cloud Vue d'ensemble – Marché Public IaaS

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

Cloud Vue d'ensemble – IaaS privé

- OpenStack
- CloudStack
- Eucalyptus
- OpenNebula

IaaS privé – Concepts

- Toute IaaS Privé doit avoir une infrastructure avec :
 - Compute
 - Storage
 - Network

IaaS privé – Concepts - Compute

- La ressource compute est constituée de :
 - Instance
 - Image
 - Flavor (gabrit)
 - Paire de clé (SSH)

IaaS privé – Concepts – Compute - 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

IaaS privé – Concepts – Compute - Image

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

IaaS privé – Concepts – Compute - FLAVOR

- 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

IaaS privé – Concepts – Compute – 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

IaaS privé – Concepts – réseau

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

IaaS privé – Concepts – Stockage

- En cloud, on distingue deux types de ressources de stockages
 - Block
 - Objet

IaaS privé – Concepts – 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
- 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

IaaS privé – Concepts – Stockage – Block - BOOT

- Démarrer une instance avec un disque racine sur un volume
- Persistance des données du disque racine
- Se rapproche du serveur classique

IaaS privé – Concepts – 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

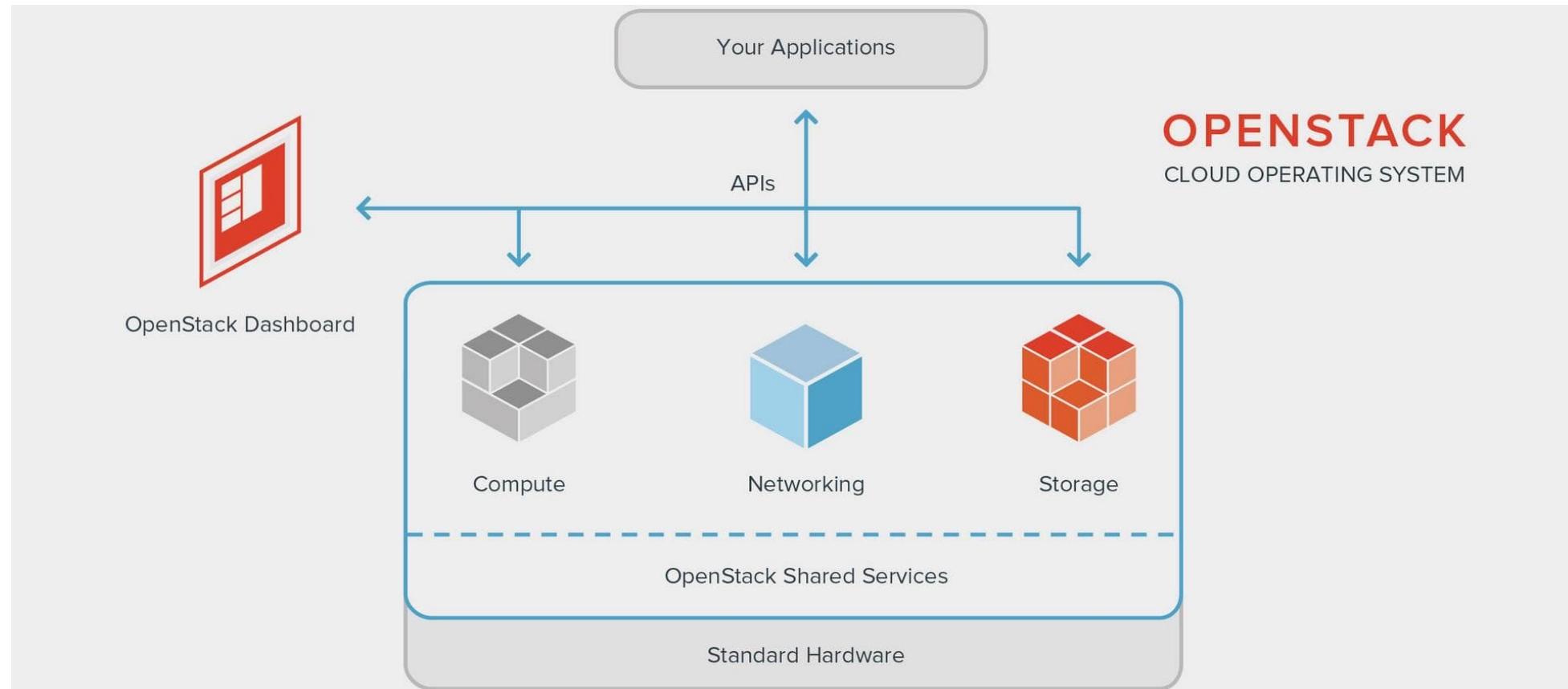
OpenStack – Introduction

- 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

OpenStack – Introduction – Fondation

- Entité de gouvernance principale et représentation juridique 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)
- Ressources humaines : marketing, événementiel, release management, quelques développeurs (principalement sur l'infrastructure)
- 600 organisations à travers le monde 80000 membres individuels dans 170 pays

OpenStack – Introduction



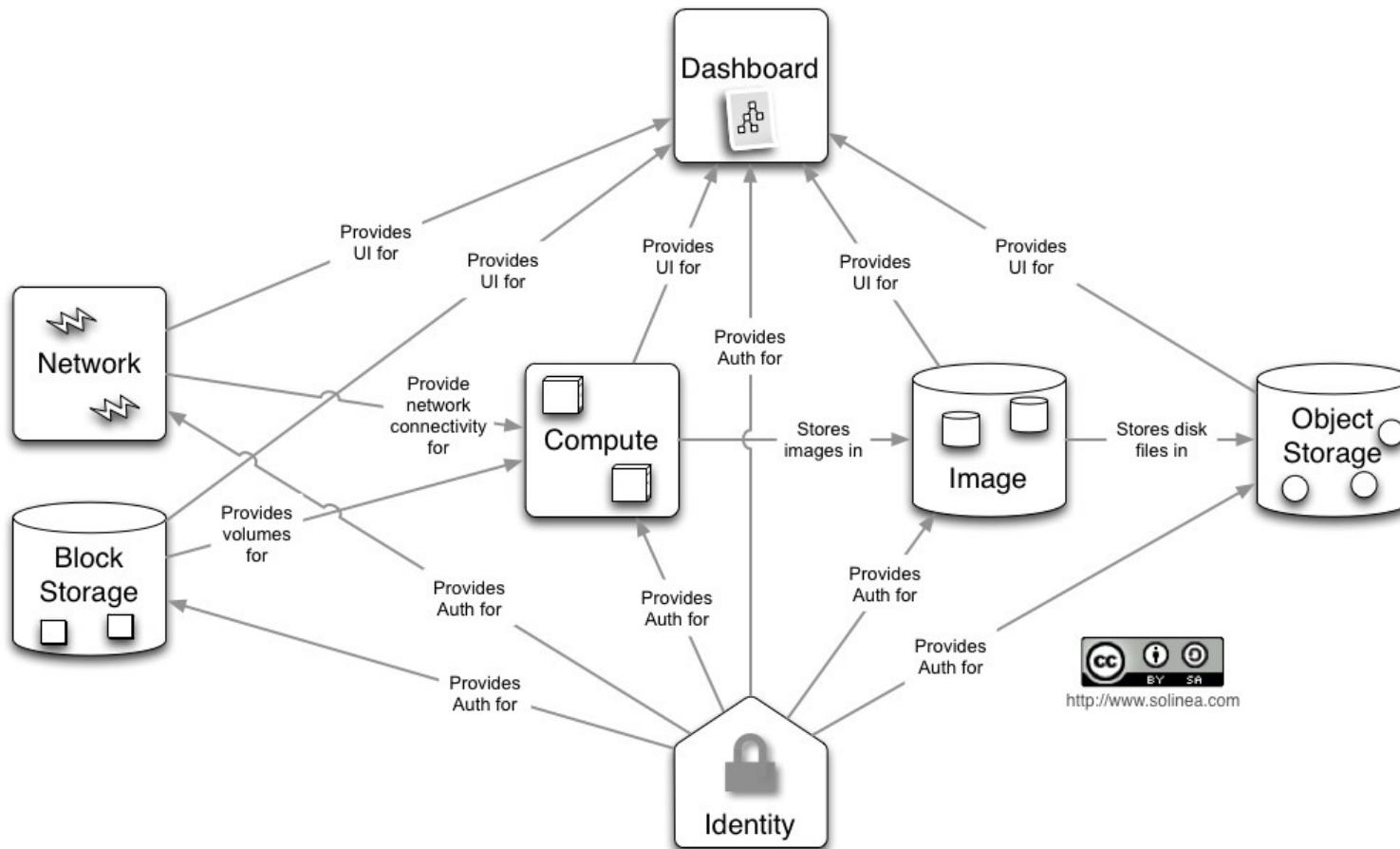
OpenStack – Introduction

- OpenStack est composé de plusieurs sous projets totalement indépendant.
- 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 Telemetry - Ceilometer
- OpenStack Orchestration - Heat

OpenStack – Introduction

- Chaque projet supporte son API OpenStack
- Certains projets supportent l'API AWS équivalente (Nova/EC2, Swift/S3)

OpenStack – Architecture



OpenStack – Implémentation

- Tout est développé en Python (Django pour la partie web)
- Chaque projet est découpé en plusieurs services (exemple : API, scheduler, etc.)
- Réutilisation de composants existants et de bibliothèques existantes
- Utilisation des bibliothèques oslo.* (développées par et pour OpenStack) : logs, config, etc.
- Utilisation de rootwrap pour appeler des programmes sous-jacents en root

OpenStack – Implémentation - Dépendances

- Base de données : relationnelle SQL (MySQL/MariaDB)
- Communication entre les services : AMQP (RabbitMQ)
- Mise en cache : Memcached
- Stockage distribué de configuration : etcd

OpenStack – 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 de la production

OpenStack – Fonctionnement devStack

- Support d'Ubuntu 16.04/17.04, Fedora 24/25, CentOS/RHEL 7, Debian, OpenSUSE
- 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

OpenStack – Configuration devstack

```
[[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
```

OpenStack – Conseils d'utilisation devstack

- 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

OpenStack – Utilisation

- Toutes les fonctionnalités sont accessibles par l'API Les clients (y compris Horizon) utilisent l'API
- Des crédentials sont nécessaires, avec l'API OpenStack :
- utilisateur mot de passe projet (tenant) domaine

OpenStack – Les APIs Openstack

- Une API par service OpenStack
- Versionnée, la rétro-compatibilité est assurée
- Le corps des requêtes et réponses est formatté avec JSON
- Architecture REST
- Les ressources gérées sont spécifiques à un projet
- <https://developer.openstack.org/#api>

OpenStack – Les APIs Openstack

- Direct, en HTTP, via des outils comme curl Avec une bibliothèque
- Les implémentations officielles en Python OpenStackSDK
- D'autres implémentations, y compris pour d'autres langages (exemple : jclouds)
- Shade (bibliothèque Python incluant la business logic)
- Avec les outils officiels en ligne de commande
- Avec Horizon
- Au travers d'outils tiers, plus haut niveau (exemple : Terraform)

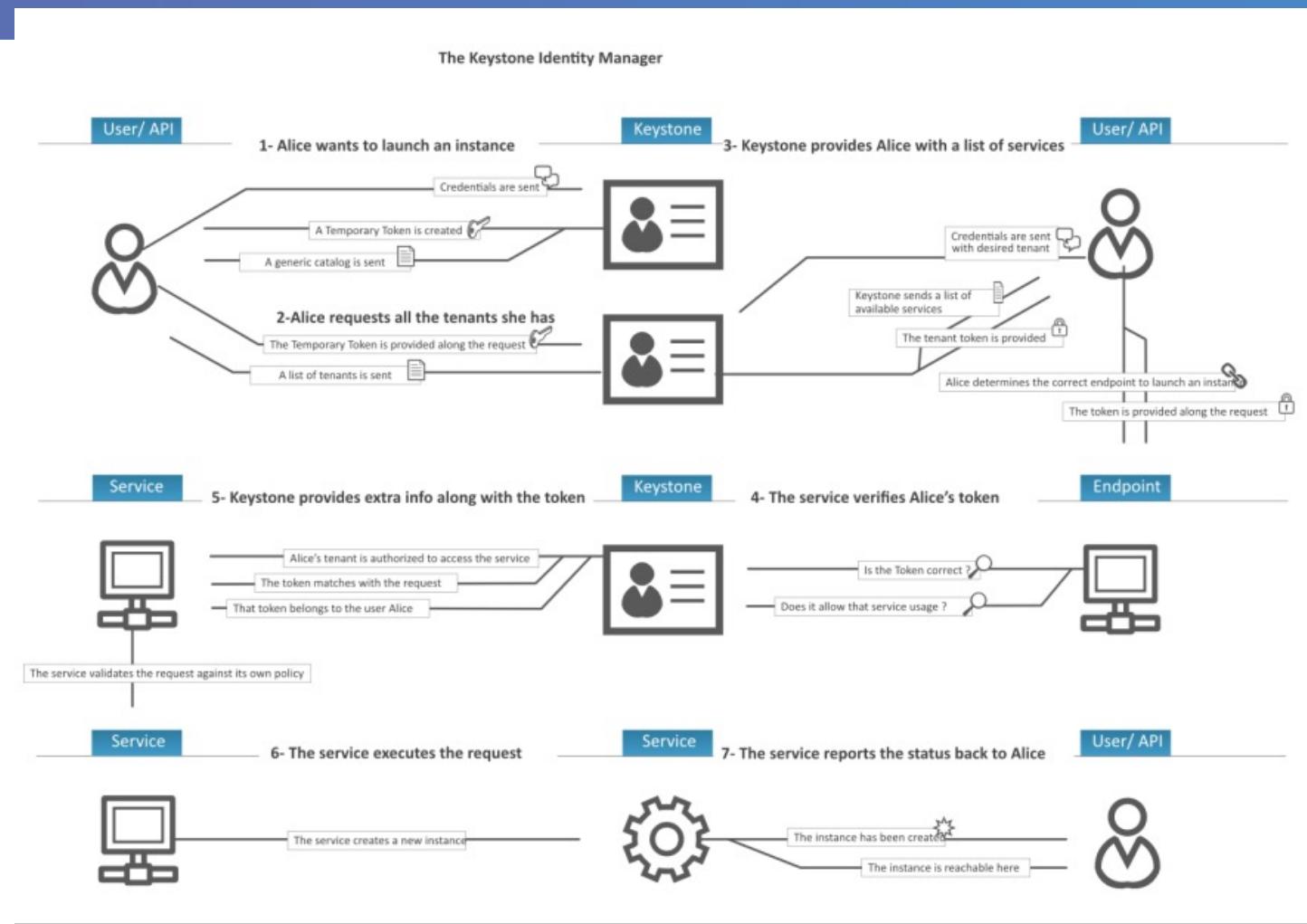
OpenStack – Clients

- OpenStack fournit des clients officiels
- Historiquement : python-PROJETclient (bibliothèque Python et CLI)
- Aujourd'hui : openstackclient (CLI)
- Outils CLI
- L'authentification se fait en passant les crédentials par paramètres, variables d'environnement ou fichier de configuration
- L'option --debug affiche la communication HTTP
- Client CLI unifié
- Commandes du type
- openstack <ressource> <action> (shell interactif disponible)
- Vise à remplacer les clients CLI spécifiques
- Permet une expérience utilisateur plus homogène Fichier de configuration clouds.yaml

OpenStack – Keystone

- Keystone est responsable de l'authentification, l'autorisation et le catalogue de services.
- L'utilisateur standard s'authentifie auprès de Keystone L'administrateur interagit régulièrement avec Keystone
- API v3 : port 5000
- Gère :
- Utilisateurs, groupes
- Projets (tenants)
- Rôles (lien entre utilisateur et projet) Domaines
- Services et endpoints (catalogue de services) Fournit :
- Tokens (jetons d'authentification)
- Pour chaque service, plusieurs endpoints sont possibles en fonction de :
- la région
- le type d'interface (public, internal, admin)

OpenStack – Keystone



OpenStack – Nova

- Gère principalement les instances
- Les instances sont créées à partir des images fournies par Glance
- Les interfaces réseaux des instances sont associées à des ports Neutron
- Du stockage block peut être fourni aux instances par Cinder
- Les instances sont :
 - Éphémère, a priori non hautement disponible Définie par une flavor
 - Construite à partir d'une image
 - Optionnel : attachement de volumes Optionnel : boot depuis un volume
 - Optionnel : une clé SSH publique Optionnel : des ports réseaux

OpenStack – Nova - API

- Ressources gérées :
 - Instances
 - Flavors (types d'instance)
 - Keypairs : ressource propre à l'utilisateur (et non propre au projet)

OpenStack – Nova – Action sur les instances

- Reboot / shutdown
- Snapshot
- Lecture des logs
- Accès VNC
- Redimensionnement
- Migration (admin)

OpenStack – Glance

- Glance est un registre d'images et de snapshots
- Glance permet de gérer les propriétés sur les images
- Glance supporte un large éventail de types d'images, limité par le support de la technologie sous-jacente à Nova
- raw qcow2 ami vmdk iso

OpenStack – Glance – propriétés des images

- 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
- Image publique : accessible à tous les projets
- Par défaut, seul l'administrateur peut rendre une image publique
- Image partagée : accessible à un ou plusieurs autre(s) projet(s)

OpenStack – Glance – Télécharger les images

- La plupart des OS fournissent des images régulièrement mises à jour :
- Ubuntu : <https://cloud-images.ubuntu.com/>
- Debian : <https://cdimage.debian.org/cdimage/openstack/> CentOS : <https://cloud.centos.org/centos/>

OpenStack – Neutron

- À l'aide de l'api, neutron permet de manipuler les ressources :
- Réseau (network) : niveau 2
- Sous-réseau (subnet) : niveau 3
- Port : attachable à une interface sur une instance, un load- balancer, etc.
- Routeur
- IP flottante, groupe de sécurité

OpenStack – Neutron – IP Flottantes

- En plus des fixed IPs portées par les instances
- Allocation (réservation pour le projet) d'une IP depuis un pool
- Association d'une IP allouée à un port (d'une instance, par exemple)
- Non portées directement par les instances

OpenStack – Neutron – 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
- Cible une adresse IP, un réseau ou un autre groupe de sécurité

OpenStack – Cinder

- Fournit des volumes (stockage block) attachables aux instances
- Gère différents types de volume
- Gère snapshots et backups de volumes
- 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)

OpenStack – Heat

- Heat est la solution native OpenStack, service d'orchestration
- Heat fournit une API de manipulation de stacks à partir de templates
- Un template Heat suit le format HOT (Heat Orchestration Template), basé sur YAML

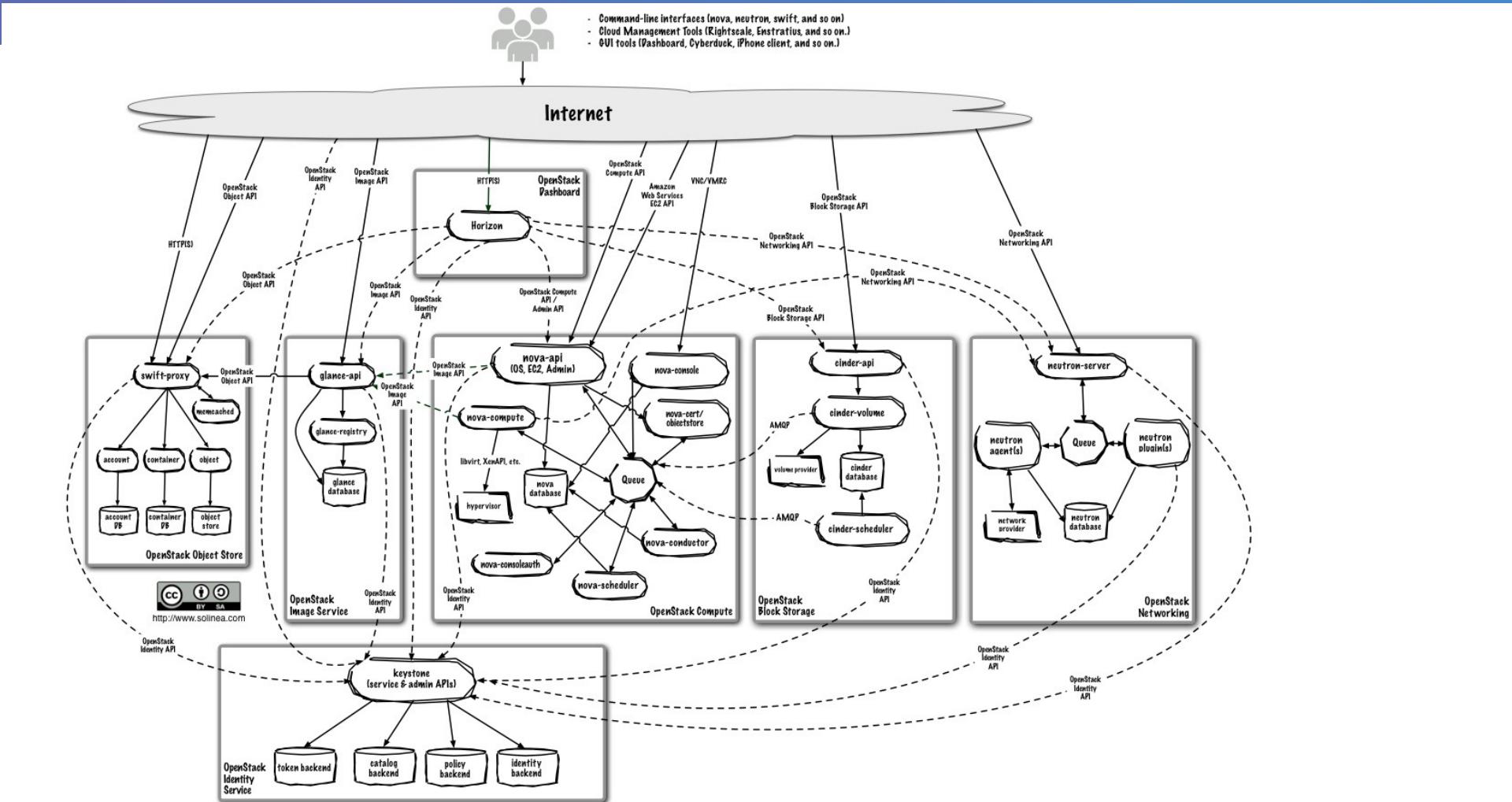
OpenStack – Horizon

- Fournit une interface web
- Utilise les APIs existantes pour fournir une interface utilisateur
- Log in possible sans préciser un projet : Horizon détermine la liste des projets accessible
- Une interface par projet (possibilité de switcher)
- Catalogue de services accessible
- Téléchargement d'un fichier de configuration clouds.yaml
- Une zone “admin” restreinte

OpenStack – Horizon

- Fournit une interface web
- Utilise les APIs existantes pour fournir une interface utilisateur
- Log in possible sans préciser un projet : Horizon détermine la liste des projets accessible
- Une interface par projet (possibilité de switcher)
- Catalogue de services accessible
- Téléchargement d'un fichier de configuration clouds.yaml
- Une zone “admin” restreinte

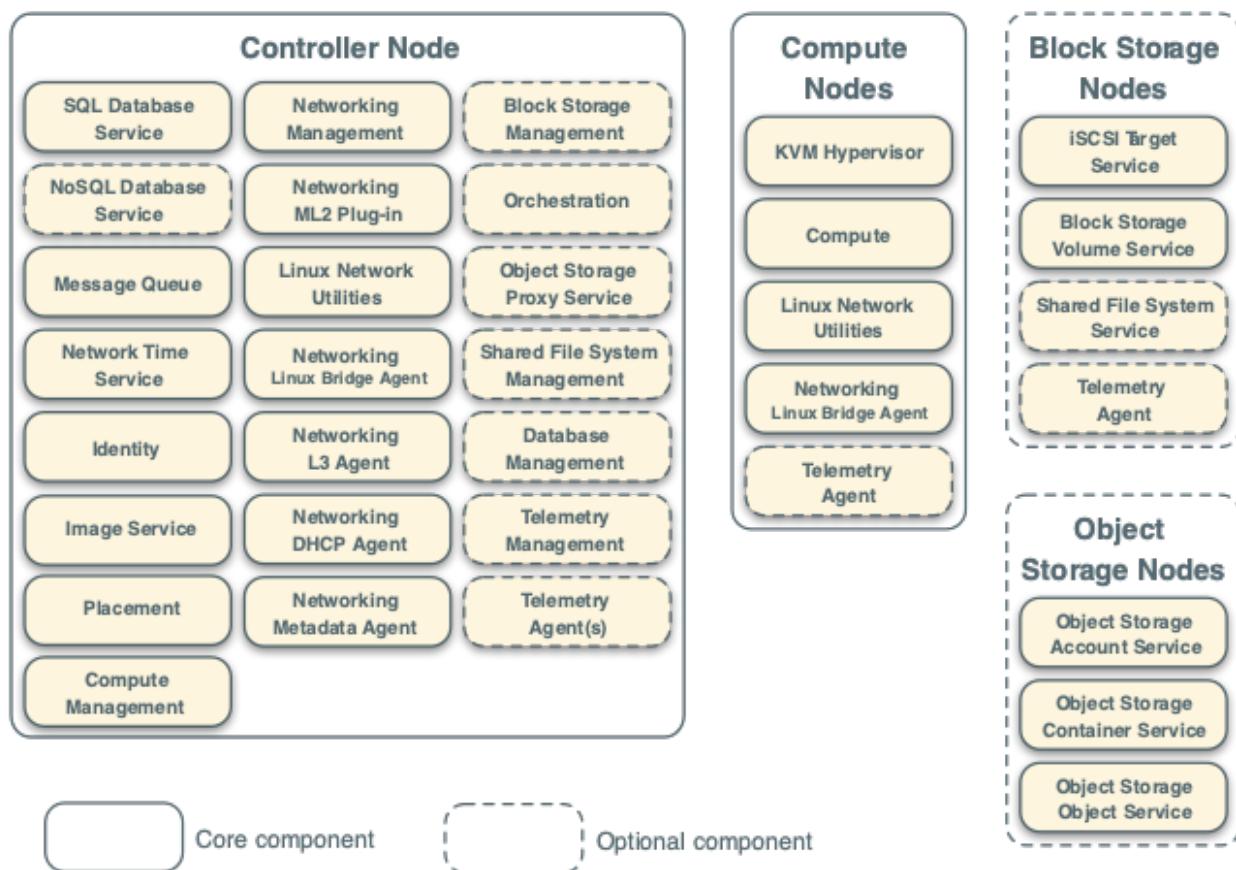
OpenStack – Déployer



OpenStack – Déployer

Networking Option 2: Self-Service Networks

Service Layout



OpenStack – Déployer

- Tous les composants doivent être configurés pour communiquer avec Keystone
- La plupart doivent être configurés pour communiquer avec MySQL/MariaDB et RabbitMQ
- Les composants découpés en plusieurs services ont parfois un fichier de configuration par service
- Le fichier de configuration policy.json précise les droits nécessaires pour chaque appel API

OpenStack – Déployer - OS

- OS Linux avec Python Ubuntu
- Red Hat
- SUSE
- Debian, Fedora, CentOS, etc.

OpenStack – Déployer - Python

- OpenStack est compatible Python 2.7
- Comptabilité Python 3 presque complète
- Afin de ne pas réinventer la roue, beaucoup de dépendances sont nécessaires

OpenStack – Déployer – 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 (et support des migrations)
- MySQL/MariaDB 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

OpenStack – Déployer – 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

OpenStack – Déployer – Cache

- Plusieurs services tirent parti d'un mécanisme de cache
- Memcached est l'implémentation par défaut

OpenStack – Déployer – Installation des services

- Documentation sur <https://docs.openstack.org/install-guide/openstack-services.html>