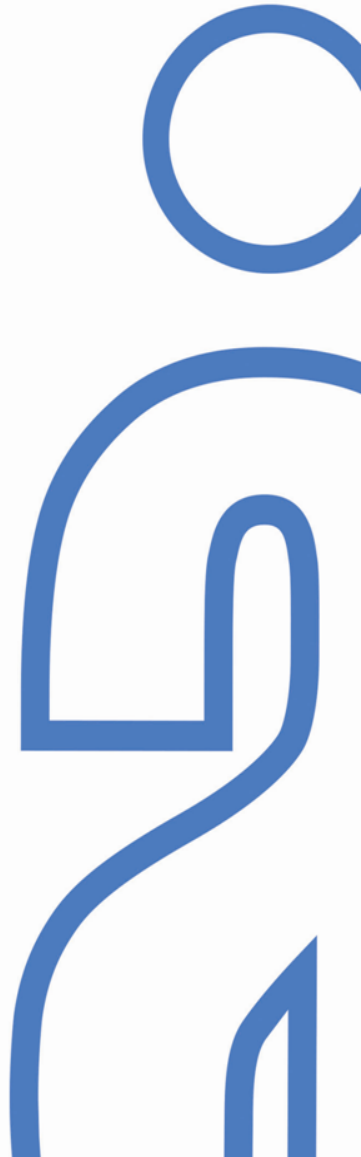
Formation OpenStack

Les bases

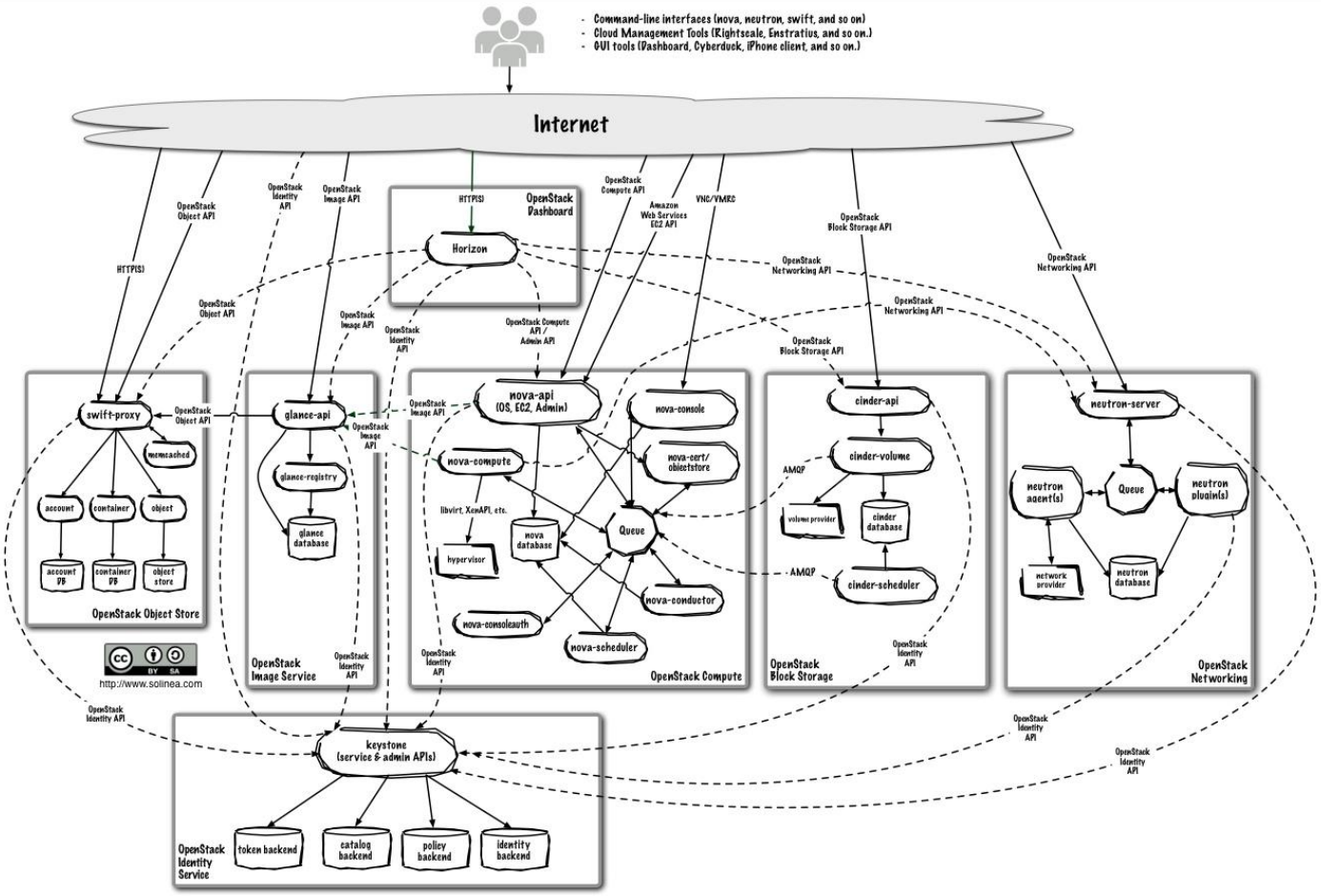


Pro Tip #1

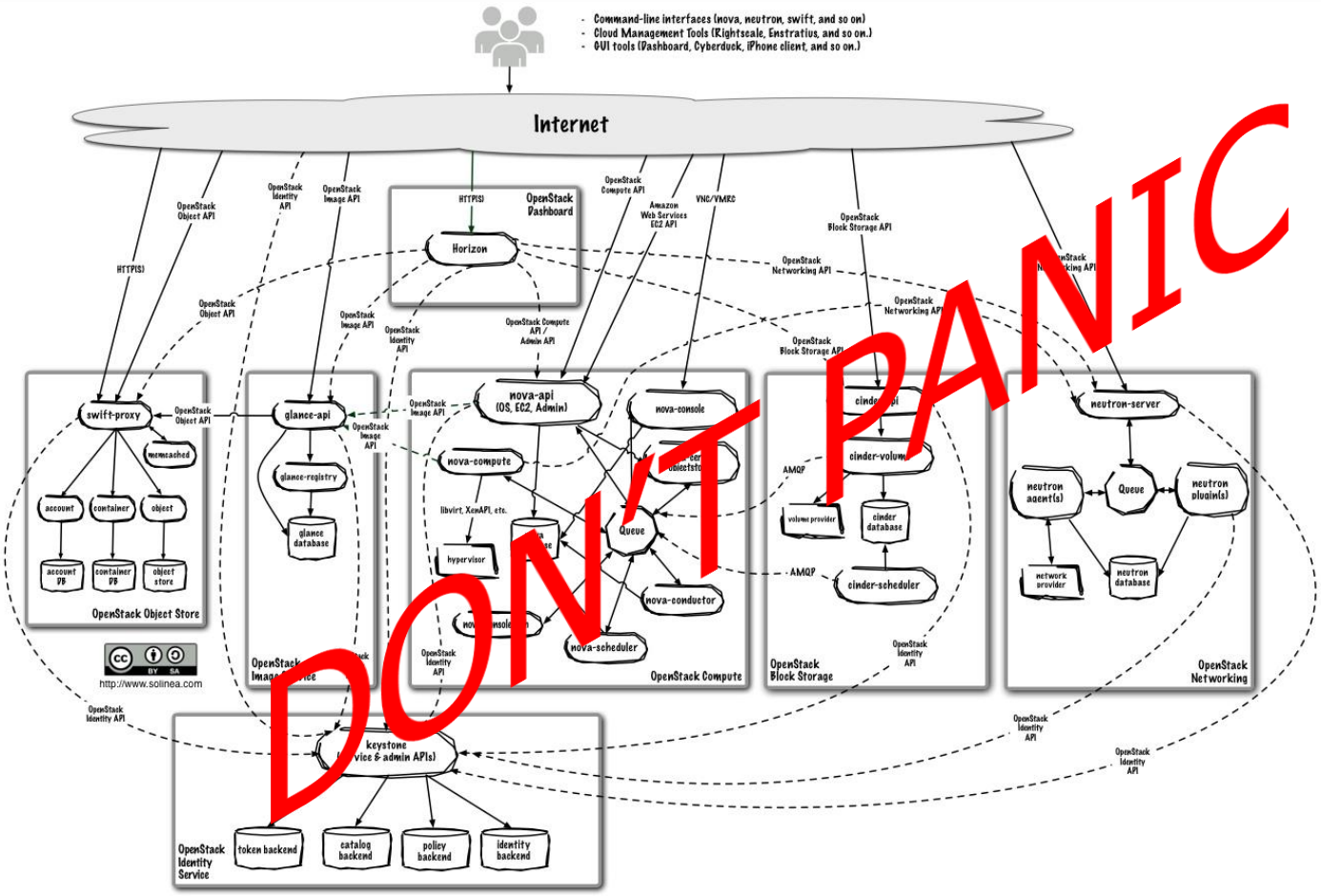
OpenStack

OpenStack, c’est quoi ?

OpenStack, c'est quoi ?



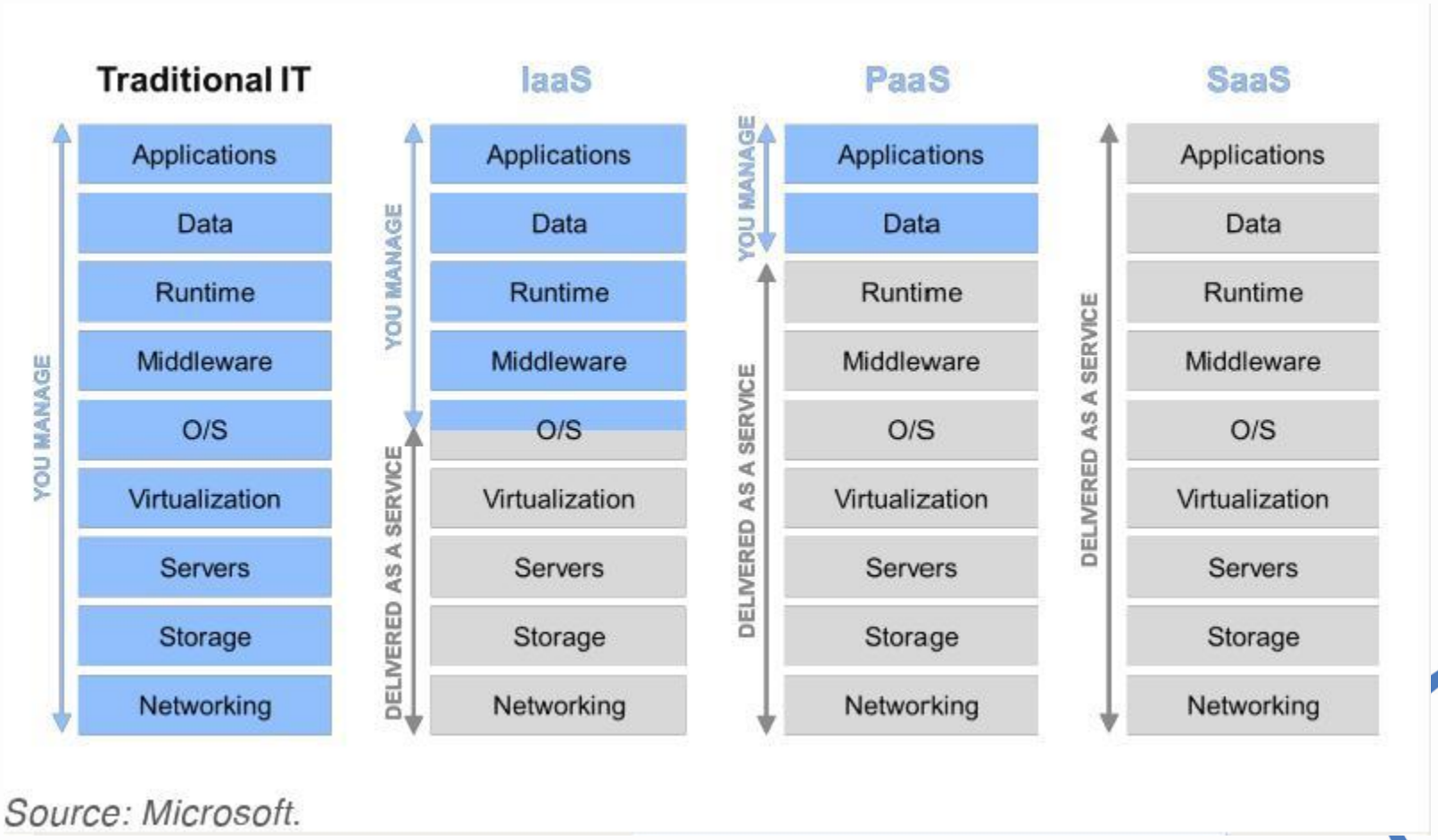
OpenStack, c'est quoi ?



Définition OpenStack

- OpenStack est un ensemble de logiciels open source permettant de déployer des infrastructures de cloud computing (Infrastructure as a service)
  - Wikipedia – 28 Novembre 2014

Rappel Cloud Computing



Source: Microsoft.

Le programme

- Le projet est porté par la fondation OpenStack qui a pour but de promouvoir OpenStack et organiser les développements
  - (http://www.openstack.org/foundation/)

- OpenStack est un programme OpenSource sous licence Apache 2.0

- OpenStack est entièrement développé en Python

Quelques chiffres

- Qelques chiffres :
  - 10 000+ : Nombre de personnes dans la fondation
  - 850+ : Nombre d'entreprises qui supportent la fondation
  - 2 738 : Nombre de développeurs ayant participé à la dernière version
  - 4 500+ : Nombre de personnes au dernier OpenStack Summit

Organisation des développements

- Une Release tout les 6 mois (3 mois à l'origine)
- Chaque Release porte un nom de code en lien avec le lieu de l'OpenStack Summit de la release

- Organisation d'un cycle de développement :
  - Organisation de l'OpenStack Summit (5 jours) ayant pour but de fixer les fonctions à ajouter dans la release
  - Itération #1 et Itération #2 lors du cycle dédiées à l'ajout de fonctionnalité
  - Itération #3 dédiée à fixer les bugs
  - En fin de cycle, une ou plusieurs release candidate avant la release définitive

Historique des versions

- A : Austin (10/2010)
- B : Bextar (02/2011)
- C : Cactus (04/2011)
- D : Diablo (09/2011)
- E : Essex (04/2012)
- F : Folsom (10/2012)
- G : Grizzly (04/2013)
- H : Havanna (10/2013)
- I : Icehouse (04/2014)
- J : Juno (10/2014)
- K : Kilo (04/2015)
- L : ???

Organisation des développements

- La gestion de projet de chaque projet est assuré par un PTL (Project Technical Leader) qui est élu par la communauté.
- Les PTL sont élus pour une release uniquement

- La gestion du programme est assuré par un Release Manager
  - C'est le même depuis le début : Thierry Carrez

Les principes d'architecture

- Chaque projet doit fournir une API complète
- Chaque projet utilise sa propre base de données
- Chaque projet à plusieurs composants afin de faciliter la scalabilité
- Chaque composant communique avec les autre composants à travers des messages AQMP
- Afin de pouvoir assurer la haute disponibilité, les composants doivent privilégier le mode actif/actif

Les composants OpenStack

- Le programme OpenStack est composé de différents projets assurant une fonction particulière
  - Keystone : Gestion d'identité et d'authentification
  - Glance : Gestion des images de VM
  - Nova : Gestion des VM
  - Cinder : Gestion du stockage de type Block
  - Swift : Gestion du stocakge de type Object
  - Neutron : Gestion du réseau
  - Horizon : Interface Web
  - Ceilometer : Gestionnaire de métriques (utilisations)

Les composants OpenStack

- Heat : Service d'orchestration de VM
- Trove : Gestion de base de données
- Sahara : Hadoop As A Service

- Autres projets (incubation) :
  - Ironic : Gestion du Bare Metal
  - Zaqar : Gestion de messages
  - TripeO : Déploiement d'instance OpenStack avec OpenStack
  - Barbican : Stockage de données cryptées
  - Designate : DNS as a Service
  - Manila : Stockage partagé
  - Murano : Gestion d'un catalogue de service

Aperçu des différents projets

Keystone : Gestion d'identité et de droits

- Keystone va fournir :
  - Gestion des droits pour les tenants (ou projets)
  - Gestion des droits pour les utilisateurs (pour chaque tenants)
  - Gestion du catalogue de services (pour chaque projets)
- L'authentification se base sur un token :
  - Un utilisateur va s'authentifier avec un username, un password et l'id de son tenant
  - Keystone va renvoyer un token (avec un ttl)
  - L'utilisateur va passer le token aux autres services et le service va demander à keystone de valider le token

Keystone : Gestion d'identité et de droits

- Keystone va stocker les données soit sur :
  - Une base MySQL
  - Une ou plusieur(s) base LDAP

- Keystone est capable de travailler en actif/actif
  - L'ajout de serveurs keystone suffit pour augmenter la capacité de traitement

Glance : Le gestionnaire d'image

- Glance permet de stocker les images des OS utilisés par les VM

- Glance est composé de 2 composants :
  - Glance-api (actif/actif)
  - Glance-registry (actif/actif)

- Glance est capable de stockage les images dans :
  - Directement sur disque
  - Ceph
  - Swift
  - GridFS (mongoDB)
  - HTTP (read-only)

Glance : Le gestionnaire d'image

- Glance prend en charge les images de type :
  - ISO
  - QCOW2 (format KVM)
  - AKI/AMI/ARI (format AWS)
  - RAW
  - VDI (format Virtualbox)
  - VHD (format Windows)
  - VMDK (format Vmware)

Nova : Le gestionnaire de VM

- Nova permet de gérer les différents hyperviseurs d'une instance OpenStack

- Nova est composé de :
  - Nova-api (API)
  - Nova-scheduler
  - Nova-compute
  - Nova-conductor (abstraction layer for DB)
  - Nova-cert (only for EC2 API)
  - Nova-consoleauth (authentication for console)
  - Nova-novncproxy (export console)

Nova : Le gestionnaire de VM

- Nova est capable de gérer les hyperviseurs :
  - KVM
  - Xen
  - Hyper-V
  - VMWare
  - LXC
  - Docker
  - BareMetal (under development see Ironic project)

Nova : Le gestionnaire de VM

- Lors de la création d'une VM, il faut sélectionner un gabarit CPU/RAM/Disque (flavor)
- Le disque va contenir l'OS.
  - Si besoin d'un autre disque, il faut se tourner vers cinder (cf après)

- La création d'un VM passe par :
  - Demande via une requête API vers nova-api
  - Message envoyé à un nova-scheduler qui va choisir l'HV
  - Le nova-compute de l'hyperviseur va exécuter les actions pour créer la VM

Cinder : Le gestionnaire de stockage Block

- Cinder va gérer le stockage de type block
- Cinder est composé de :
  - Cinder-api
  - Cinder-scheduler
  - Cinder-volume
- Après la création d'un disque via cinder, il est possible d'attacher le disque à une VM

Cinder : Le gestionnaire de stockage Block

- Cinder est capable d'adresser différents matériels de stockage
  - Disques via LVM
  - Ceph
  - NFS
  - NetApp
  - EMC
  - HP 3Par
  - …
- Cinder ne fait pas l'intermédiaire pour les IO et la VM va discuter directement le système de stockage
  - Via iSCSI pour le stockage LVM
  - Via NFS pour le stockage NFS
  - Via RBD pour le stockage Ceph
  - …

Swift : Le gestionnaire de stockage Objet

- Swift est un clone d'Amazon S3 qui permet de stocker des fichiers en mode objet
- Le protocole de transport est HTTP(S)

- Swift est un projet un peu à part dans le domaine d'OpenStack et il n'est pas intégré dans le cycle de développement

Neutron : Le gestionnaire de réseau

- Neutron va gérer les aspects réseaux (L2,L3 et NFV)

- Neutron est composé de :
  - Neutron-server (API)
  - Neutron-plugin* (gestion du L2, voir slide suivant)
  - Neutron-l3-agent (gestion du l3)
  - Neutron-metadata-agent
  - Neutron-dhcp-agent

- Neutron va sûrement être coupé en 2 dans les prochaines release afin d'avoir :
  - Gestionnaire de la base L2/L3
  - Gestion NFV (Firewall, VPN, LB, etc…)

Neutron : Le gestionnaire de réseau

- Neutron est capable d'adresser des solutions software mais aussi des solutions hardwares (type Alcatel Nuage, Cisco Nexus, etc…)

- Au niveau L2, Neutron est capable de gérer des réseaux virtualisés ou non :
  - VLAN
  - VXLAN
  - Tunnel GRE

- En mode software, Openvswitch est le plus utilisé

Neutron : Le gestionnaire de réseau

- Au niveau L3, le neutron-l3-agent va faire aussi de gateway pour les réseaux et nécessite la mise en place d'un cluster type pacemaker pour faire de la haute disponibilité (actif/passif)

- Depuis Juno, le neutron-l3-agent lève cette limitation et il est maintenant possible d'avoir une configuration HA avec un système DVR (Distributed Virutal Router)

Neutron : Le gestionnaire de réseau

- En plus de la gestion du L2/L3, Neutron est capable de proposer des services de type NFV dont :
  - Firewall as-a-service
  - Load-balancer as-a-service (basé sur Haproxy ou solutions hardware type F5)
  - VPN as-a-service
  - …
- L'utilisation de ce type de services n'est pas encore compatible avec DVR

Horizon : Interface Web

- Horizon est une interface Web permettant de piloter une instance OpenStack
- Horizon est développé avec le framework Python Django
- Nécessite l'utilisation d'un serveur Web type Apache/nginx ou autres
- L'authentification de l'interface s'appuie sur Keystone
- Pour chaque action dans l'interface, Horizon va juste générer la requête aux APIs

Ceilometer : le gestionnaire de métrique

- Ceilometer va gérer les différentes métriques de l'instance OpenStack

- Ceilometer est composé de :
  - Ceilometer-api
  - Ceilometer-collector
  - Ceilometer-agent
  - Ceilometer-alarm

- Il y a un très grand nombres de métriques disponibles (nombre d'instances lancées, nombre de Mbits par tenant, etc…)

Ceilometer : le gestionnaire de métrique

- 2 mode de fonctionnement :
  - Ceilometer va récupérer à intervalle régulier des infos sur les différents composants d'OpenStack
  - Ceilometer va recevoir des composants OpenStack des infos (rôle du collector)

- Du fait du grand nombres de métriques, les données sont stockées dans une base NoSQL : MongoDB

- L'objectif de Ceilometer est de permettre de faciliter le travailler de monitoring, de capacity planning et de billing

Architecture OpenStack

- Une architecture typique d'OpenStack est :
  - 1 ou N nœud dit « controler »
  - 1 ou N nœud dit « compute »
  - 1 ou N noeud dit « network » (pre-juno)
  - 1 ou N nœud dit « storage »

Architecture Controller

- Le rôle des serveurs « controller » est principalement de faire tourner les différentes API ainsi que les services transverses
- Pour les petits déploiements, il y a généralement 2 serveurs « controller »
- Les services portés sont :
  - Keystone
  - Glance
  - Cinder
  - Nova-api, nova-scheduler
  - Neutron-server
  - Horizon

Architecture « compute »

- Le rôle des serveurs « compute » est de faire tourner les VM
- Le nombre de nœuds va dépendre du nombre de VM max du Cloud
- Les services portés sont :
  - Nova-compute, nova-conductor, nova-cert et nova-novnc
  - Cinder-volume si le stockage est fait directement sur les HV
  - Neutron-plugin (L2) et l3 agent (dans le cas du DVR)

Architecture « network »

- Le rôle des serveurs « network » est de gérer le réseau des VM
- NB: Avec Juno, possibilité d'enlever ces nœuds et utiliser le DVR sauf si utilisation des services NFV
- Typiquement 2 serveurs en Actif/Passif (il est possible de faire tourner ça sur les nœuds « controller »)

- Les services portés sont :
  - Neutron-l3-agent
  - Neutron-dhcp-agent
  - Neutron-metadata-agent

Architecture « storage »

- Le rôle des seveurs « storage » est d'assurer le stockage des VM et des volumes
- De plus en plus d'installations d'OpenStack se tournent vers le système Ceph

- Le nombre de serveurs pour un cluster Ceph est totalement dépendant de la volumétrie souhaitée pour le Cloud mais aussi du nombre d'IO

Ceph

- Ceph est un système de stockage distribué sans SPOF

- L'accès à un cluster Ceph est possible via :
  - RadosGW : Gateway HTTP (identique à S3 ou Swift)
  - RBD : Protocole assez similaire à du iSCSI mais l'accès se fait par l'ensemble des nœuds du cluster ceph
  - CephFS : Filesystem posix (PAS POUR LA PRODUCTION pour le moment)

- La donnée est répliquée 3 fois

Ceph

- Du fait que Ceph gère lui-même la réplication, il n'y a pas d'intérêt d'avoir du RAID sur les disques

- Une configuration matérielle typique est :
  - Bi Proc
  - 128 Go de RAM
  - 2 disques systèmes (avec RAID)
  - 2 disques SSD pour l'écriture des journaux Ceph
  - X disques SATA/SAS en JBOD

Ceph

- Depuis Icehouse, il est possible d'utiliser RBD pour :
  - Stocker les VM (avec nova) permettant de pouvoir faire du live migration
  - Stocker les volumes (avec cinder)

- Cela permet de gérer un seul système de stockage et une grande facilité d'administration

Gestion de la haute disponibilité OpenStack

- OpenStack n'intègre pas de mécanisme de gestion de la haute disponibilité
  - Nécessite de mettre en place du load-balancing pour les services permettant de travailler en actif/actif (ex: les apis)
  - Nécessite de mettre en place des systèmes de clustering type pacemaker pour les services en actif/passif
  - Au niveau des bases MySQL, l'utilisation de Galera Cluster est assez habituelle ou via pacemaker+drbd
  - Au niveau RabbitMQ, il intègre du clustering de base

AGARIK, HEBERGEUR D’UN MONDE INTELLIGENT