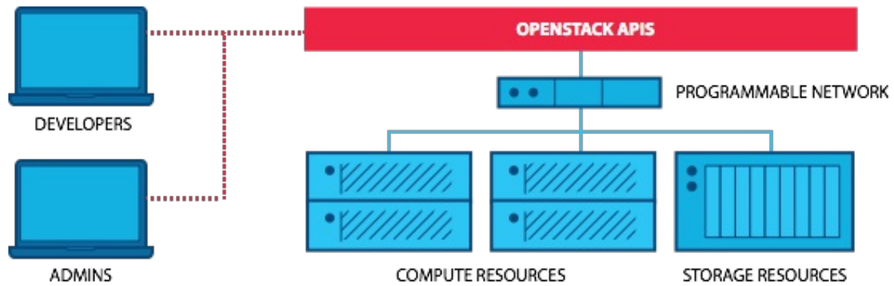
Introduction à OpenStack

Présentation d'OpenStack

Objectifs du projet

> Le projet OpenStack vise à produire la plateforme de Cloud Computing open source de référence qui répondra aux besoins de clouds publics et privés de toute taille, de par sa facilité de déploiement et sa capacité à passer massivement à l'échelle.

- Logiciel open source pour construire un cloud
  - Infrastructure programmable
    - Couche d'abstraction des ressources calcul, stockage et réseau
    - Accès via des APIs
  - Historiquement principalement tourné vers l'IaaS
  - Ajout de fonctionnalités PaaS et SaaS



Présentation d'OpenStack

Historique

- Projet démarré début 2010
- Collaboration entre RackSpace et la NASA

- NASA: Projet Nebula (démarré en 2008)
  - Consolider les serveurs liés aux activités Web
    - Gestion façon cloud
  - Pas d'outil open-source satisfaisant
    - Eucalyptus
      - Passage à l'échelle insuffisant
      - Composants propriétaires
  - Développement de Nova
    - Contrôleur de ressources de calcul
    - API similaire à EC2

- RackSpace
  - Réecriture et mise en open source de sa plateforme interne: Ozone
  - Swift: composant de stockage objet

Présentation d'OpenStack

Un projet ouvert

- Premier "Design Summit" en juillet 2010
  - 25 entreprises partenaires
    - AMD, Autonomic Resources, Citrix, Cloud.com, Cloudkick, Cloudscaling, CloudSwitch, Dell, enStratus, FathomDB, Intel, iomart Group, Limelight, Nicira, NTT DATA, Opscode, PEER 1, Puppet Labs, RightScale, Riptano, Scalr, SoftLayer, Sonian, Spiceworks, Zenoss et Zuora
  - Plus de 160 aujourd'hui
- Les 4 "Open"
  - Source
  - Conception
  - Développement
  - Communauté

Présentation d'OpenStack

La fondation OpenStack

- Créée en septembre 2012
  - Développer, encourager, protéger et proumouvoir openstack
- Trois composantes
  - Conseil d'administration: defini les objectifs de la fondation, contrôle le budget et la marque déposée OpenStack
  - Comité technique: pilote les aspects techniques et gère le projet open source OpenStack
  - Comité utilisateurs: représente les besoins des utilisateurs

AT&T

Ericsson

Huawei

Intel

Rackspace

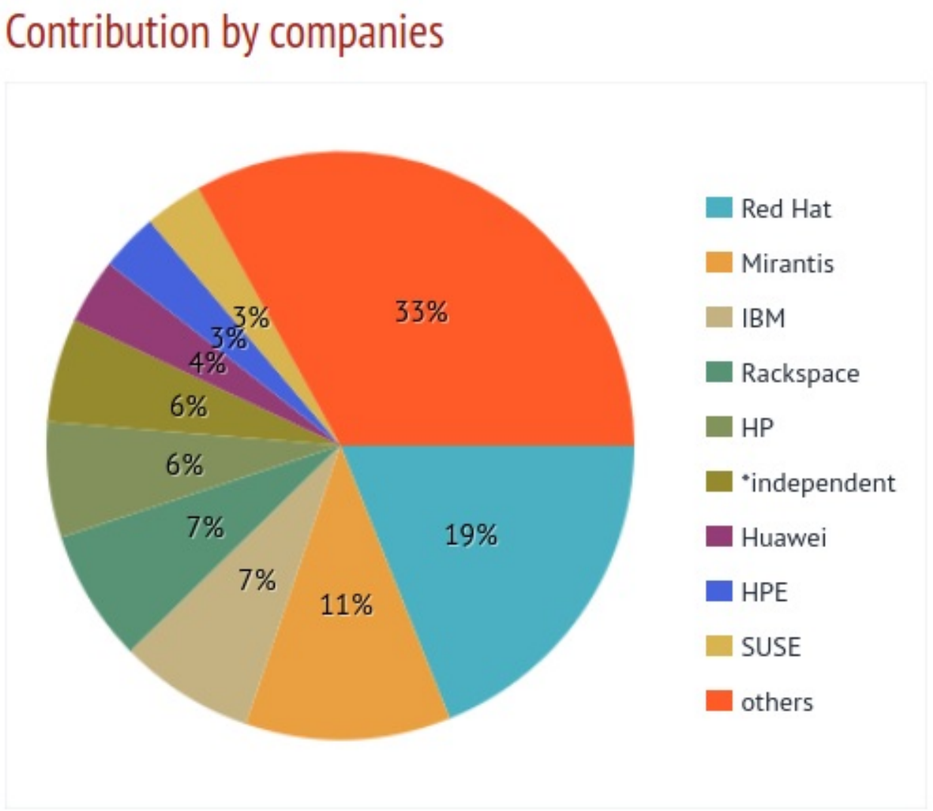
Red Hat, Inc.

SUSE

Tencent Cloud

Présentation d'OpenStack

Principaux contributeurs jusqu'à aujourdhui



Exemples d'utilisateurs

Cloud Public

- Le cloud public d'**OVH** est basé sur OpenStack. OVH héberge plus de 260 000 instances et 150PB de données dans son cloud public.
- De nombreux autres exemples d'opérateurs de taille moyenne référencés sur openstack.org dont **CloudWatt**, anciennement cloud souverain français.

Centres HPC

- Le **CERN** utilise un cloud OpenStack (plus de 8000 noeuds et 22000 instances) pour traiter les données produites par le Large Hadron Collider (LHC).
- De nombreux centres de calcul HPC, (par ex. celui de la NASA) supplémentent leur offre HPC par une offre Cloud basé sur OpenStack pour proposer une offre de calcul plus flexible.

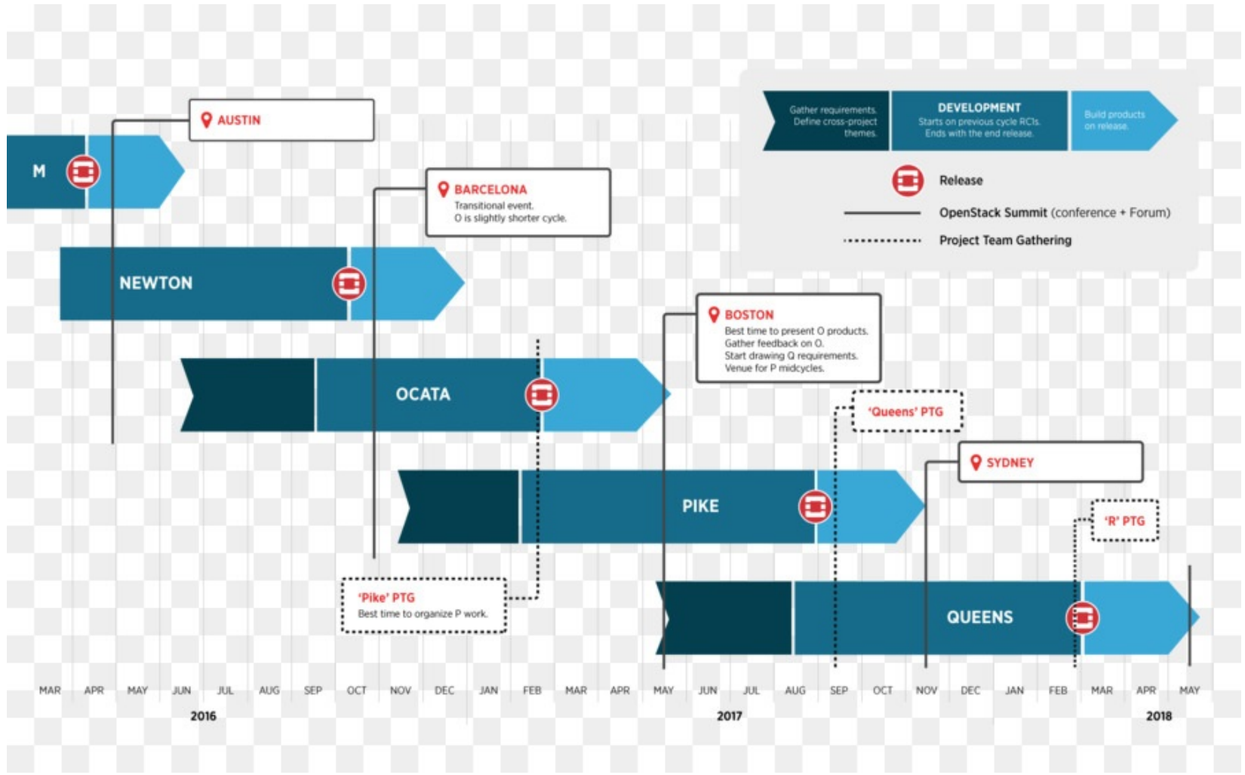
Cloud Privé

- **Walmart** utilise OpenStack pour sa plateform e-commerce mondiale et implémenter ses services destinés aux utilisateurs sur PC, mobile ou borne interactive.

Présentation d'OpenStack

Cycle de développement

- Une version tous les 6 mois
  - Nom de ville par ordre alpahabétique
    - Lieu proche du sommet OpenStack associé
  - Compatibilité garantie supérieure à 2 versions



Présentation d'OpenStack

Principales fonctionnalités

Les fonctionnalités coeur d'OpenStack répondent aux principaux besoins d'une plateforme IaaS

- Instances de calcul à la demande
  - VMs, machines physiques et conteneurs
  - Gestion des images
  - Déploiement et configuration
  - Snapshots
  - Migration
- Réseaux virtuels self-service
  - Segmentation L2
  - Sous-réseaux IP
  - Routage
  - Filtrage réseau, firewall
  - Equilibrage de charge

Présentation d'OpenStack

Principales fonctionnalités

Les fonctionnalités coeur d'OpenStack répondent aux principaux besoins d'une plateforme IaaS

- Utilisation multi-tenant
  - Domaines
  - Projets
  - Groupes
  - Utilisateurs
- Stockage
  - Objet (Media, HTML, Images de VM, …)
  - Bloc (Volumes disque persistants)
  - Systèmes de fichiers

Présentation d'OpenStack

Principales fonctionnalités

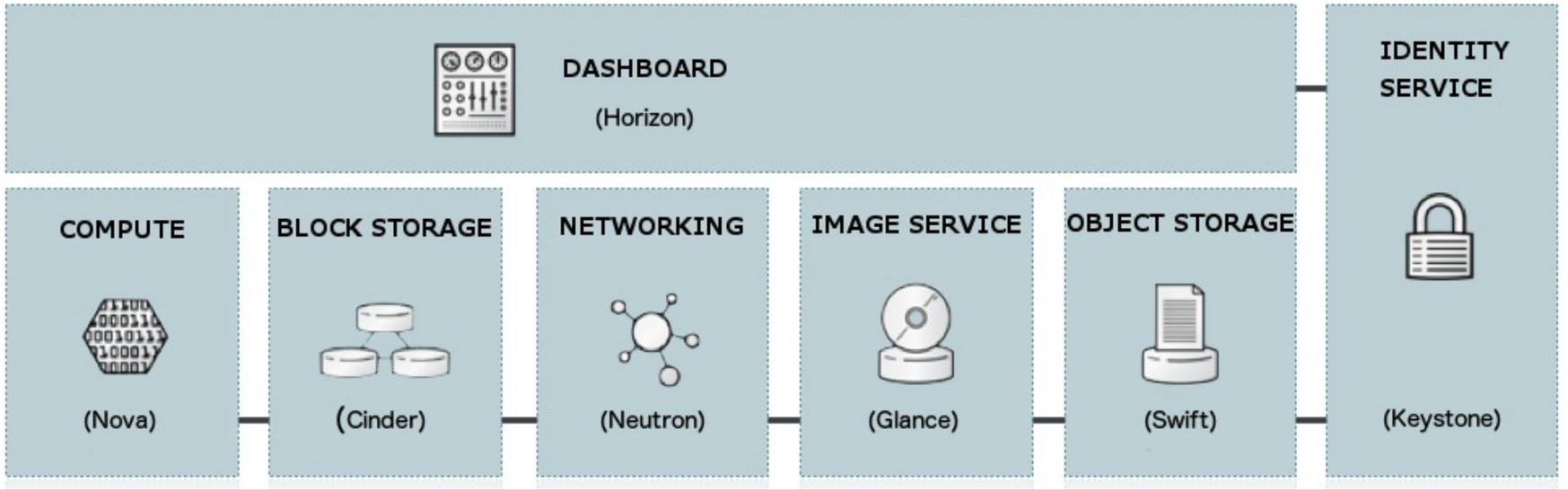
Les fonctionnalités coeur d'OpenStack répondent aux principaux besoins d'une plateforme IaaS

- Interface Web, API et ligne de commande
- Haute disponibilité et montée en charge
  - Gestion de regions multiples
  - Zones de disponibilité
  - Services distribués et redondants

Présentation d'OpenStack

Principaux projets

- Architecture modulaire
  - Projets relativement indépendants sous la direction du comité technique
  - Un leader techique par projet
  - Relative autonomie dans la definition des planning, jalons etc.
  - Communications entre projets à travers des APIs stable



Présentation d'OpenStack

Nombreux projets additionnels

- Plus de 60 composants
  - IaaS, PaaS, SaaS
  - Divers degré de maturité et d'utilisation

Adjutant (Operations Processes automation)
Barbican (Key Manager service)
Blazar (Resource reservation service)
Chef Openstack (Chef cookbooks for deployment)
Cinder (Block Storage service)
Cloudkitty (Rating service)
Congress (Governance service)
Cyborg (Accelerator Life Cycle Management)
Designate (DNS service)
Documentation
Ec2-Api (EC2 API compatibility layer for OpenStack)
Freezer (Backup, Restore, and Disaster Recovery service)
Glance (Image service)
Heat (Orchestration service)
Horizon (Dashboard)
I18n
Infrastructure
Ironic (Bare Metal service)
Karbor (Data Protection Orchestration Service)
Keystone (Identity service)
Kolla
Kuryr
Loci
Magnum (Container Infrastructure Management service)
Manila (Shared File Systems service)
Masakari (Instances High Availability Service)
Mistral (Workflow service)
Monasca (Monitoring)
Murano (Application Catalog service)
Neutron (Networking service)
Nova (Compute service)
Octavia (Load-balancer service)
Openstack Charms (Juju Charms for deployment of OpenStack)
Openstack-Helm (Helm charts for OpenStack services)
Openstackansible (Ansible playbooks and roles for deployment)
Openstackclient (Command-line client)
Openstacksdk (Multi-cloud Python SDK for End Users)
Oslo (Common libraries)
Packaging-Rpm
Powervmstackers
Puppet Openstack (Puppet modules for deployment)
Qinling (Function as a Service)
Quality Assurance
Rally (Benchmark service)
Release Management
Requirements
Sahara (Data Processing service)
Searchlight (Search service)
Senlin (Clustering service)
Solum (Software Development Lifecycle Automation service)
Storlets (Compute inside Object Storage service)
Swift (Object Storage service)
Tacker (NFV Orchestration service)
Telemetry (Telemetry service)

Présentation d'OpenStack

Correspondance avec les services AWS

- Management Console: Horizon
- EC2: Nova
- S3: Swift
- IAM: Keystone
- VPC: Neutron
- RDS: Trove
- Cloudwatch: Ceilometer
- Cloudformation: Heat
- SQS: Zaqar
- Route53: Designate

Demo

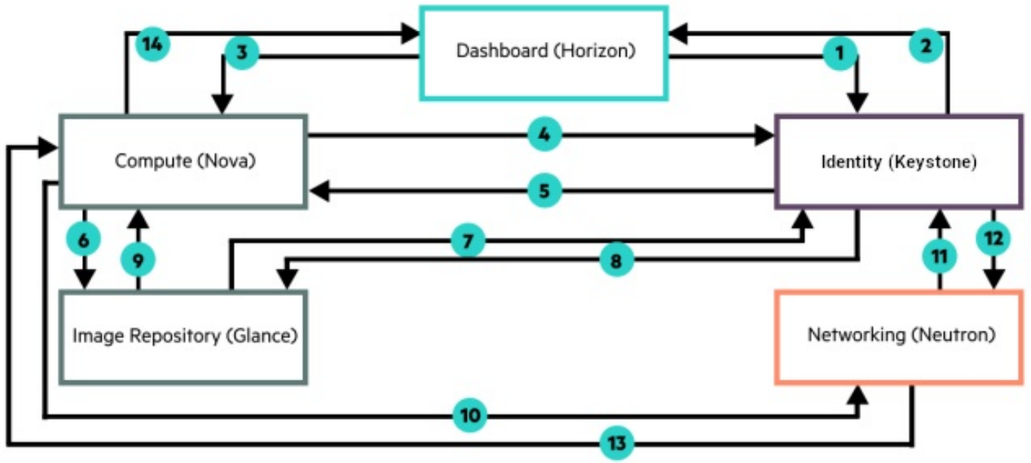
Présentation de Keystone

Service d'identité

- Catalogue de service
  - Chaque composant enregistre les URL de ses API
  - Interne, externe, administration
- Authentification
  - Login/Mdp, LDAP …
  - Fédération avec d'autre fournisseurs d'identité (SAML, OAuth, OpenID etc.)
  - Federation entre plusieurs déploiements OpenStack
- Authorisation
  - Gestion de rôles (RBAC)
  - Mise en relation avec une poliitque propre à chaque service
- Fonctionnement par jetons (token)
  - Récupération d'un jeton en s'authentifiant auprès de Keystone
  - Transfert aux autres services pour s'authentifier auprès d'eux
- Au coeur d'OpenStack
  - Rien ne fonctionne sans ce composant
  - Le premier à installer (et à mettre en haute dipsonibilité)

Présentation de Keystone

Circulation des token



Présentation de Keystone

Concepts gérés par le service

- **Utilisateur**
  - Identité associé à un Login/MDP ou autre backend
- **Groupe**
  - Liste d'utilisateurs
- **Projet**
  - Synonyme de tenant (parfois utilisé dans certaines doc/CLI)
  - Propriétaire des différentes ressources créées (VMs, réseaux, images etc.)
- **Rôle**
  - Identifie les droits associés à un utilisateur
    - Chaque composant OpenStack défini les droits associés à chaque rôle
  - Associations par triplet
    - Un utilisateur (ou groupe) est associé à des rôles dans des projets
    - A l'authentification un utilisateur demande un token pour un projet
  - Rôle admin associé par défaut aux actions sensibles

Présentation de Keystone

Concepts gérés par le service

- **Quota**
  - Limites d’utilisation par projet ou domaine
- **Domaine**
  - Conteneurs pour projets et utilisateurs
  - Décentralisation de la gestion d’un cloud partagé
  - Delégation la gestion d’un domaine à un administrateur de domaine
    - Création des utilisateurs et projets du domaine
    - Répartition des quotas entre les projets

Présentation de Keystone

Exemples de commandes:

- Création d'un projet

```
openstack project create --description "École ENSIIE" ensiie
```

- Création d'un utilisateur

```
openstack user create --password secretpass user1
```

- Création d'un rôle

```
openstack role create teacher
```

- Association d'un utilisateur à un rôle dans un projet

```
openstack role add --project ensiie --user user1 teacher
```

Présentation de Horizon

Service de tableau de bord

- Interface graphique d'OpenStack
- Portail web self-service pour utilisateur et administrateur
- Optionnel car tout peut être fait en ligne de commande
- A l'inverse tous les services OpenStack ne sont pas intégrés à Horizon

Présentation de Glance

Service d'image

- Catalogue d'image
  - Parcours et recherche des images disponible
- Possibilité d'enregistrer dynamiquement de nouvelles images
  - Global ou local à un projet
  - Utilisateur ou administrateur
  - Upload ou création à partir d'une instance en cours
  - Association de métadonnées aux images (clé/valeurs)
- Gestion des formats d'images standard
- Stockage dans divers backend
  - Système de fichiers local
  - Stockage objet local OpenStack ou distant (AWS S3 …)
- Utilisé pour booter une VM
  - Copie de l'image dans un stockage éphémère (souvent sur l'hyperviseur)
  - Création d'un volume persistent

Présentation de Glance

Exemples de commandes

- Uploader une image

```
openstack image create \
            --file /home/user/cirros-0.4-x86_64-disk.img \
            --container-format bare \
            --public \
            cirros-0.4
```

- Positionner des meta-données

```
openstack image set --property architecture=x86_64 \
            --property hypervisor_type=qemu \
            cirros-0.4
```

Présentation de Glance

Exemples de commandes

- Lister les images enregistrés

```
openstack image list
+--------------------------------------+------------+
| ID                                   | Name       |
+--------------------------------------+------------+
| abc5f0c7-7193-464c-b39e-bd04dc4ca7cc | cirros-0.4 |
+--------------------------------------+------------+
```

Présentation de Neutron

Service réseau

- Anciennement nommé Quantum
- Remplace la gestion réseau interne au service **calcul** (Nova)
- Permet la mise en place de réseaux virtuels à la demande
  - Abstraction et simplicité de la configuration
  - Software defined networking
  - Automatisation des déploiements
- Indépendance vis à vis de l'infrastructure physique
  - Topologie, type de matériel, configuration …

Présentation de Neutron

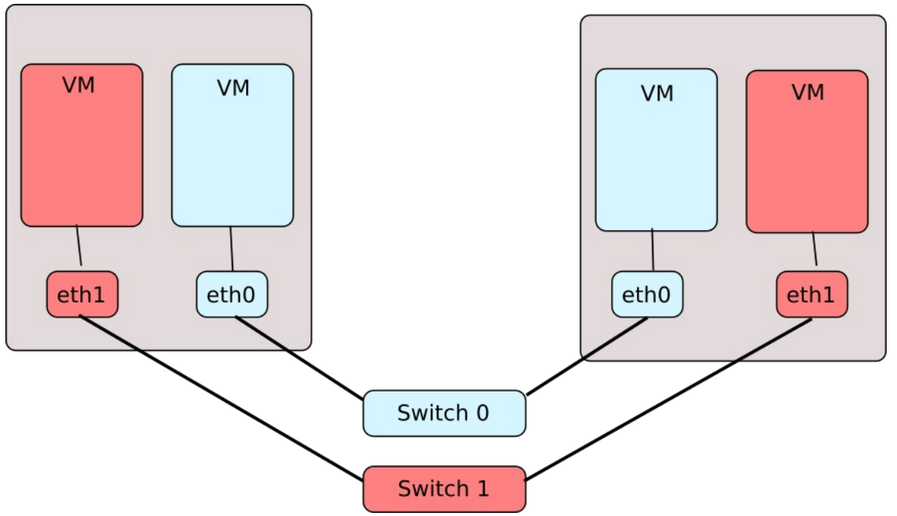
Service réseau

- Un système de plugin modulaire (ML2: modular layer 2)
  - Remplace les plugins monolithiques existant précédemment
  - Différents types de réseaux (pilote de **type**)
    - Réseaux à plat
    - VLANs physiques
    - Réseaux d'overlay (tunnels VXLAN, GRE, …)
  - Sur différents matériels/logiciels (pilote de **mécanisme**)
    - Bridge Linux (switch virtuel simple)
    - OpenVswitch (switch virtuel avancé supportant OpenFlow)
    - Modèles de switchs physiques
    - Contrôleurs SDN externe (OpenDaylight)

Présentation de Neutron

Différents types de réseaux

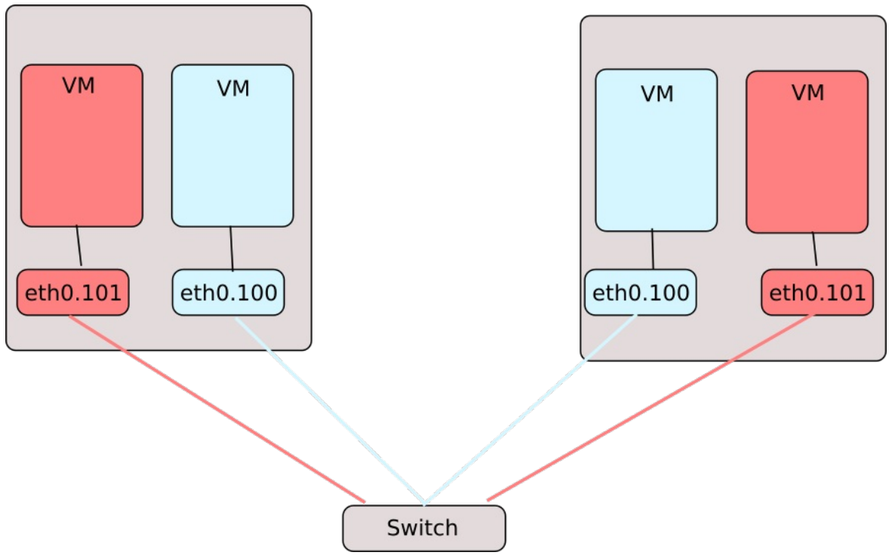
- Réseau à plat
  - Les paquets sont directement envoyés sur interface physique hôte
  - Connexion à un réseau existant
  - Pas d'isolation
    - Généralement pas utilisable directement par un utilisateur
    - Connexion à un réseau existant à travers un routeur
  - Contrainte de placement pour deux VMs dans un même sous-réseau
    - Hyperviseurs dans un même réseau Ethernet L2



Présentation de Neutron

Différents types de réseaux

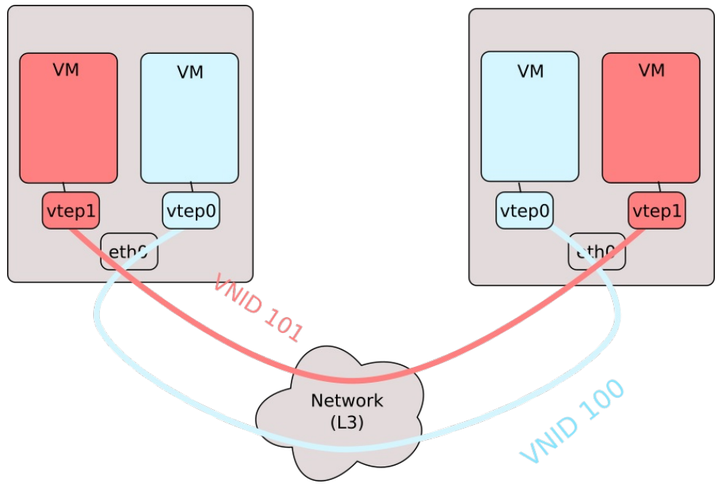
- VLAN
  - Les paquets sont envoyés sur l'interface physique avec un tag VLAN 802.1q
  - Isolation par VLAN
    - Maximum 4094 réseaux indépenants
  - Contrainte de placement pour deux VMs dans un même sous-réseau
    - Hyperviseurs dans un même réseau Ethernet L2



Présentation de Neutron

Différents types de réseaux

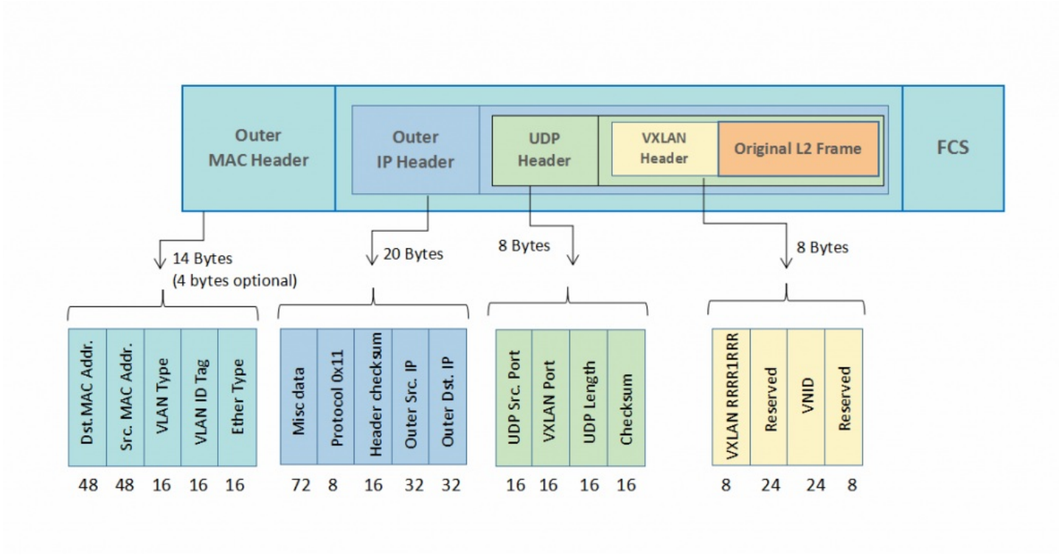
- VXLAN
  - Les paquets sont encapsulés à l'aide du protocole VXLAN
  - Envoi de paquets UDP entre deux hyperviseurs
    - Routage IP entre plusieurs réseaux Ethernets possible
    - Pas de contrainte de placement
  - Jusqu'à 16 milions de réseaux indépendants
  - Surcout en performance, baisse de la MTU disponible
  - Pas de broadcast matériel



Présentation de Neutron

Différents types de réseaux

- VXLAN
  - Les paquets sont encapsulés à l'aide du protocole VXLAN
  - Envoi de paquets UDP entre deux hyperviseurs
    - Routage IP entre plusieurs réseaux Ethernets possible
    - Pas de contrainte de placement
  - Jusqu'à 16 milions de réseaux indépendants
  - Surcout en performance, baisse de la MTU disponible
  - Pas de broadcast matériel



Présentation de Neutron

Concepts gérés par le service

- **Réseau**
  - Équivalent à un segment réseau physique (VLAN sur réseau Layer 2)
  - Partagé ou privé
    - Partagé: Tous les projets peuvent y connecter directement des VMs (créer des ports)
    - Privé: Réservé à un projet
  - Externe ou interne
    - Externe: capable de fournir une connectivité externe
      - Autorisation d'y connecter un routeur faisant passerelle par défaut
      - Routage par SNAT/DNAT, IP flottante…
  - Options *provider*
    - Type de réseau (flat, VXLAN, GRE, VLAN, …)
    - Identifiant de segmentation (numéro de VLAN, …)
    - Réseau physique utilisé
  - Généralement un utilisateur non privilégié peut créer
    - Réseaux internes
    - Privés à un ou plusieurs projet
    - Sans pouvoir spécifier d'options *provider*
    - Définition automatique à partir de la configuration

Présentation de Neutron

Concepts gérés par le service

- **Sous-réseau**
  - Plage IP allouée dans un réseau
    - Unique par réseau
    - Plusieurs réseaux peuvent utiliser les mêmes plages
  - Options courantes
    - Service et IP DHCP
    - IP de passerelle
    - IP de serveur DNS

Présentation de Neutron

Concepts gérés par le service

- **Port**
  - Connexion à un sous réseau
  - Permet d'injecter / recevoir des paquets
  - Intialement un port a juste une existance logique
    - Réservation d'une IP
  - Puis connexion à une VM, un routeur virtuel, un serveur dhcp …
    - Matérialisation du port sur un hyperviseur par exemple
    - Choix du mécanisme adapté parmis ceux disponible sur le noeud
      - OpenVswitch, Linux bridge ….
    - Prévention du *spoofing*
      - MAC, ARP
      - IP

Présentation de Neutron

Concepts gérés par le service

- **Security group**
  - Permet de restreindre le traffic reçu/émis par une instance
  - Un port fait partie d'un ou plusieurs security groups
  - Un security group est défini par un ensemble de règles
  - Chaque règle permet d'ouvrir un flux
    - En émission ou réception
    - Depuis/vers un sous-reseau ou un autre security group
  - Tout traffic ne corresepondant à aucune règle est bloqué
  - Par défaut un port est dans le security group par défaut
    - Autorise tout traffic sortant
    - Autorise tout traffic entrant depuis le même security group
    - Bloque tout traffic entrant de l'extérieur

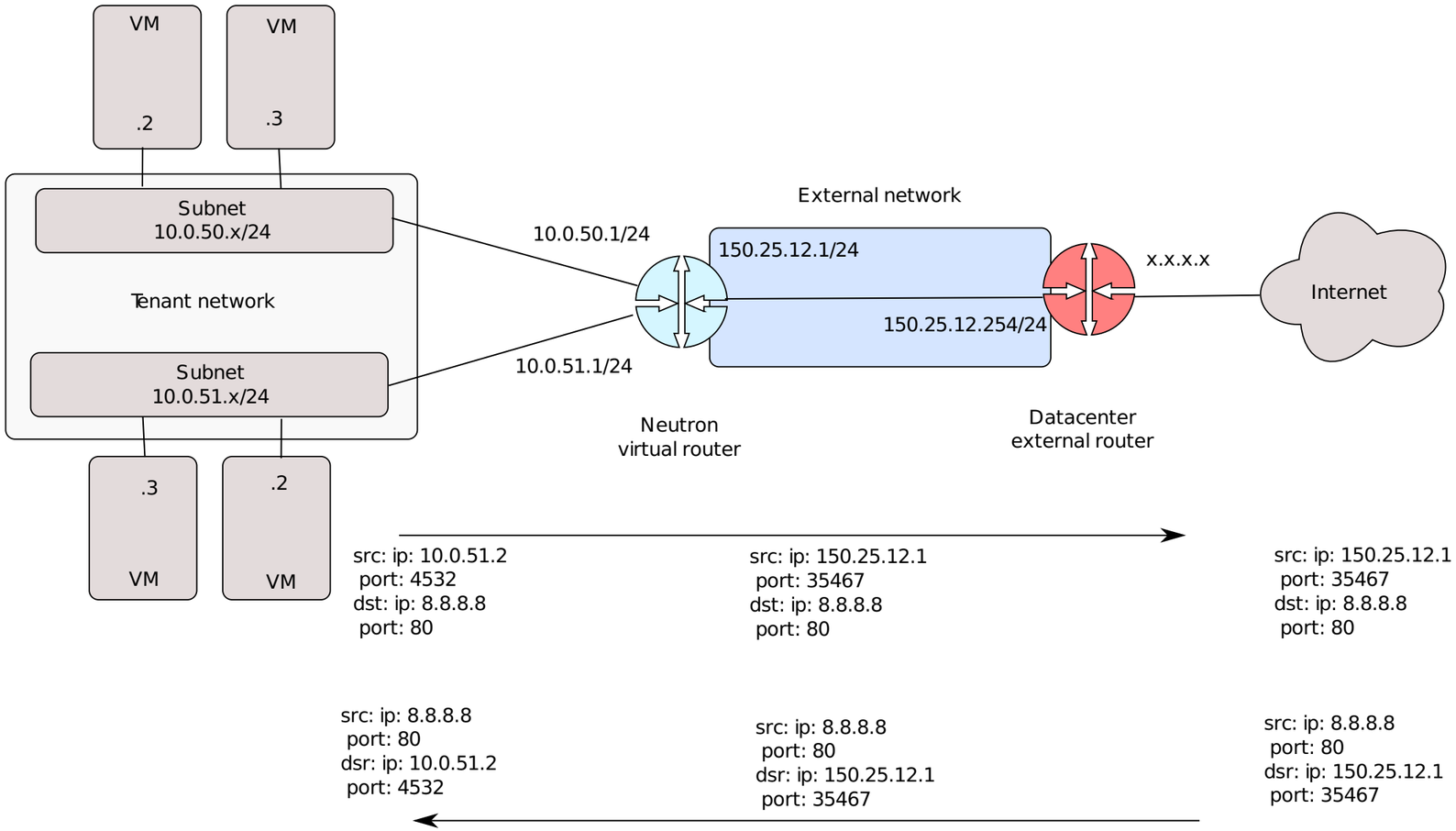
Présentation de Neutron

Concepts gérés par le service

- **Routeur**
  - Permet de router des paquets entre sous-réseaux, généralement:
    - Un ou plusieurs sous-réseaux internes
    - Un unique sous-réseau externe
  - Passerelle par défaut
    - Passerelle du sous-réseau externe
    - SNAT/DNAT, IP flottante …
  - Routage vers le service de metadonnées
    - Fournit des informations pour configurer une VM au démarrage (cloud init)

Présentation de Neutron

Routage NAT



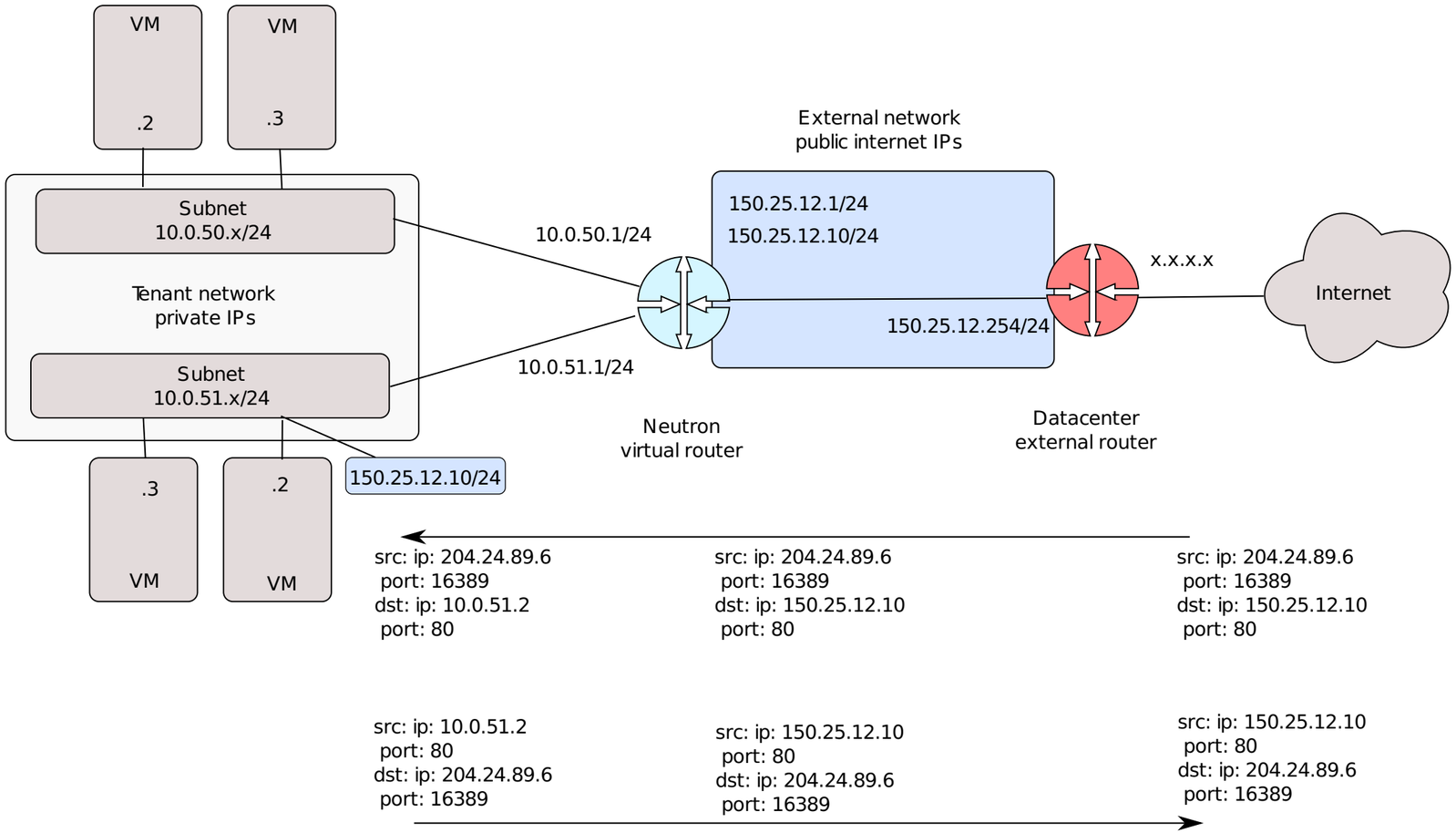
Présentation de Neutron

Concepts gérés par le service

- **IP flottante**
  - Associe une IP externe à une IP interne
  - Permet à un noeud de sortir avec une IP dédiée plutôt que celle du routeur
    - SNAT vers cette IP plutôt que l'IP du routeur
  - Permet à un noeud d'être contacté de l'extérieur par cette IP
    - DNAT de paquet reçu sur l'IP externe vers l'IP interne
  - Ressource souvent limitée par rapport aux IP interne
  - IP publiques disponibles pour l'organisation

Présentation de Neutron

Routage par IP flottante (DNAT)



Présentation de Neutron

Exemple d'utilisation

- Création d'un réseau externe *provider* (administrateur)

```
openstack network create --external --provider-network-type flat \
                         --provider-physical-network physical1 \
                         internet
```

  - le réseau est privé mais externe (on pourra y connecter un routeur)
  - *physical1* fait référence à un identifiant défini dans la configuration du mécanisme utilisé

```
[linux_bridge]
physical_interface_mappings = physical1:eth1
```

Présentation de Neutron

Exemple d'utilisation

- Définition d'un sous-réseau IP dans ce réseau (administrateur)

```
openstack subnet create --network internet --subnet-range 150.25.12.0/24 \
                        --gateway 150.25.12.254  \
                        internet-s1
```

- La passerelle (*gateway*) correspond à l'IP d'un routeur pré-existant
- Utilisée comme passerelle pour les ports définis dans ce sous réseau

Présentation de Neutron

Exemple d'utilisation

- Création d'un réseau interne *tenant* (utilisateur)

```
openstack network create mynetwork
```

  - Chaque réseau contient par défaut:
    - un serveur dhcp pour assigner des IPs dans les différents subnets
    - un serveur de metadonnées pour transférer des configurations type cloud-init
- Création de deux sous-réseau dans ce réseau (utilisateur)

```
openstack subnet create --network mynetwork --subnet-range 10.50.0.1/16 \
                        mysubnet1
openstack subnet create --network mynetwork --subnet-range 10.51.0.1/16 \
                        mysubnet2
```

Présentation de Neutron

Exemple d'utilisation

- Creation d'un routeur (utilisateur)

```
openstack router create myrouter
```

- Ajout de port du routeur dans chaque sous-réseau (utilisateur)

```
openstack router add subnet myrouter mysubnet1
openstack router add subnet myrouter mysubnet2
```

- Ajout d'un port externe pour la passerelle par defaut

```
openstack router set myrouter --external-gateway internet
```

  - Le réseau *provider* internet étant privé on peut pas créer un port directement
  - On peut y attacher un routeur comme il est externe

Présentation de Neutron

Exemple d'utilisation

- Création d'une IP flottante (utilisateur)

```
openstack floating ip create internet --tag myip
```

- Créaton d'un port, association d'une IP flottante (utilisateur)

```
openstack port create --network mynetwork --fixed-ip subnet=mysubnet1 myport
openstack floating ip set --port myport myip
```

  - Le port est associé au security group par défaut
- Ajout du port à une instance (utilisateur)

```
openstack server add port <myvm> myport
```

Présentation de Cinder

Service de volumes

- Fournit du stockage persistent pour les instances de calcul
  - Equivalent à AWS EBS
  - Survit à la terminaison d'une instance
- Gère le cycle de vie des périphériques bloc
  - Création, connexion aux VMs, destruction
  - Snapshot, Backup
  - Peuplement à partir d'une image
- Drivers pour divers systèmes de stockage physiques
  - LVM (+iSCSI)
  - NFS
  - Ceph
  - …
- Ordonnancement entre plusieurs backends
  - Critères de placement (filtres)

Présentation de Cinder

Concepts gérés par le service

- **Volume**
  - Espace de stockage bloc (format *raw*) d'une taille choisie
  - Possibilité d'initialiser les données avec:
    - Backup, Snapshot, Image
    - Autre volume
  - Choix parmi les **types** de volume définis par l'administrateur
    - Backend utilisé
    - QoS
    - Chiffrement
- **Snapshot**
  - Copie read-only d'un volume à un instant donné
  - Peut être fait à chaud
- **Backup**
  - Sauvegarde compressée dans un stockage objet
  - Support de backups incrémentaux

Présentation de Cinder

Exemples d'utilisation

- Création d'un volume à partir d'une image

```
openstack volume create --image cirros --size 10 myvolume
```

- Connexion du volume à une VM instance

```
openstack server volume add <myvm> myvolume
```

- Création d'un snapshot

```
openstack volume snapshot create --volume myvolume mysnapshot
```

Présentation de Nova

Service de calcul

- Composant historique d'OpenStack
- Fournit un accès à des ressources de calcul en self-service
  - VM, machines physiques ou conteneurs
- Capable de gérer de nombreux hyperviseurs (exploite libvirt)
  - KVM
  - Xen
  - Hyper-V …
- Coordonne la création de l'instance et des ressources associées
  - Ports réseau
  - Volumes
  - Image

Présentation de Nova

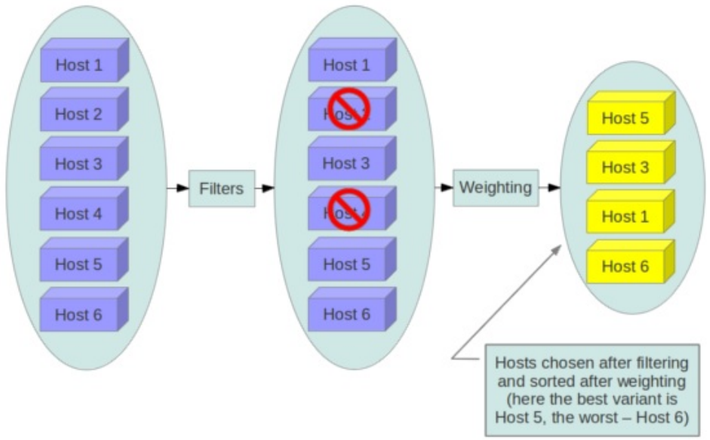
Concepts

- **Serveurs** virtuels (instances)
- **Flavors**
  - Similaire aux types d'instances AWS
  - Définit les caractéristiques des serveurs virtuels
    - Quantité de mémoire et de swap
    - Nombre de cpus virtuels
    - Taille du disque éphémère
    - Métadonnées: paires de clés/valeurs
      - Options prises en compte par l'hyperviseur ex:
        - quota:disk_read_bytes_sec= bande passante maximum en lecture
        - hw:numa_nodes= nombre de noeuds NUMA virtuels
      - Indication sur les caractéristiques de l'hyperviseur
        - capabilities:hypervisor_type= type d'hyperviseur (KVM, Xen …)
      - Clés/valeurs arbitraires
- **Keypair**
  - Clé publique/privée SSH

Présentation de Nova

Concepts

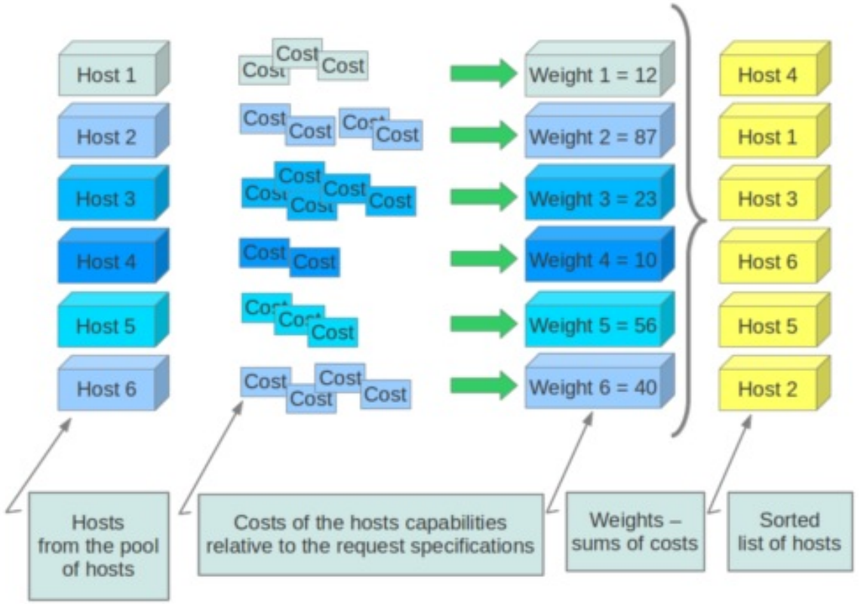
- **Ordonnanceur**
  - Filtres
    - Élimine les noeuds qui n'ont pas les caractéristiques requises
      - Méta-données et état des noeuds
      - Flavor et image choisie
      - **Hints** fournis à l'allocation de l'instance
    - Exemples:
      - ComputeCapabilitiesFilter, ImagePropertiesFilter, DifferentHostFilter, MemoryFilter …



Présentation de Nova

Concepts

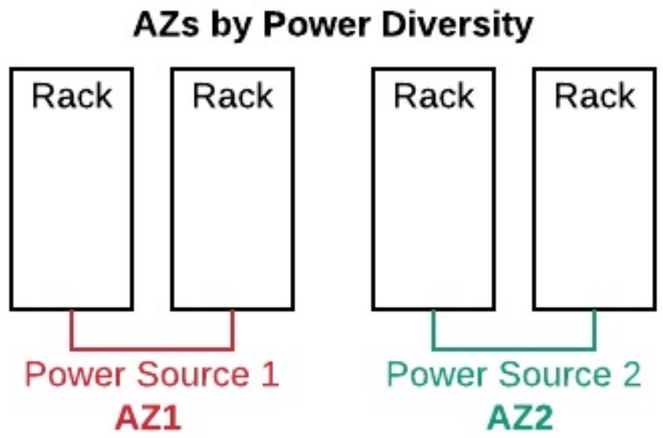
- **Ordonnanceur**
  - Poids
    - Choisi le noeud le plus approprié parmis ceux qui n'ont pas été filtrés
    - Somme de metrique * poids de la metrique
    - Ex: mémoire utilisée, nombre de CPU alloués, taux d'utilisation CPU …



Présentation de Nova

Concepts

- **Host aggregates**
  - Spécification de méta-données pour un groupe d'hyperviseurs
  - Un hyperviseur peut être en plusieurs aggregats
  - Cas particulier: **zone de disponibilité**
    - Attribut spécifique d'un aggregat: *availability_zone=*
    - Un noeud ne peut être que dans un seul aggrégat définissant une AZ



Présentation de Nova

Exemple d'utilisation

- Création d'une flavor (administrateur)

```
openstack flavor create --vcpus 1 --ram 64 --disk 1 m1.nano
```

- Spécification de métadonnées (administrateur)

```
openstack flavor set --property capabilities:hypervisor_type=qemu m1.nano
```

- Lancement d'une instance à disque éphémère

```
openstack server create --flavor m1.nano --image cirros \
                    --network mynetork --key-name mykey \
                    myvm
```

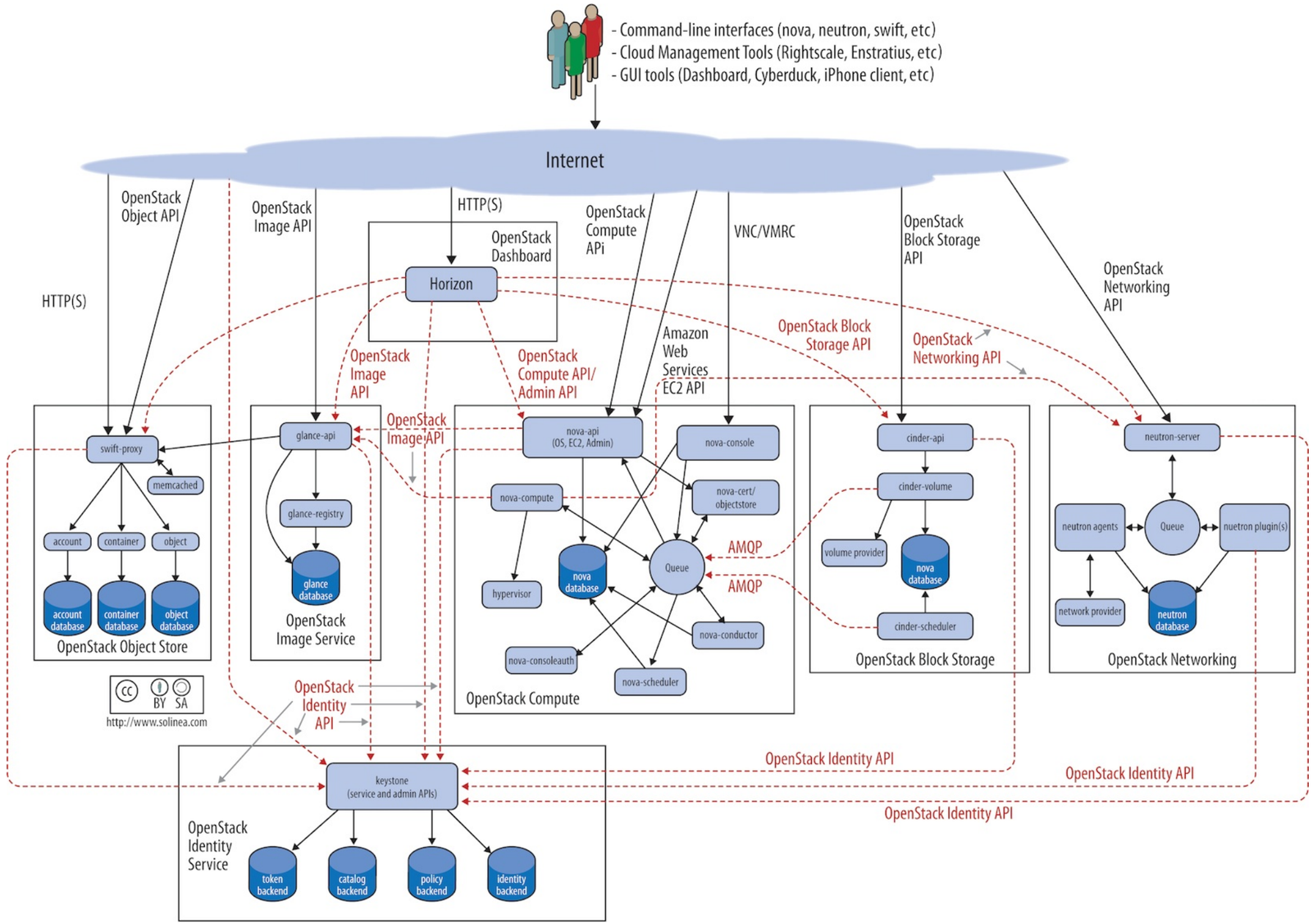
Présentation de Nova

Exemple d'utilisation

- Lancement d'une instance à disque persistent (volume)

```
openstack server create --flavor m1.nano --image cirros \
                     --block-device source=volume,\
           id=myvolume_id,dest=volume,\
                       shutdown=preserve,bootindex=0 \
         --network mynetork --key-name mykey \
                       myvm
```

Architecture générale



Architecture générale

Projets OpenStack

- Un projet fait généralement intervenir les sous-composants suivant:
  - Un serveur d'API REST
  - Une base de données
  - Des files de messages
  - Un ordonnanceur
  - Des agents
- Les agents peuvent être centralisés ou distribués
  - *Centralisés*: chaque agent peut traiter l'ensemble des requêtes du type qui le concerne (à travers une file de messages partagée)
    - Ex: nova-conductor traite les requêtes venant de tout hyperviseur
  - *Distribués*: chaque agent est configuré pour traiter une sous partie des ressources
    - Ex: neutron-linuxbridge-agent: traite les ports et bridges Linux sur un noeud donné

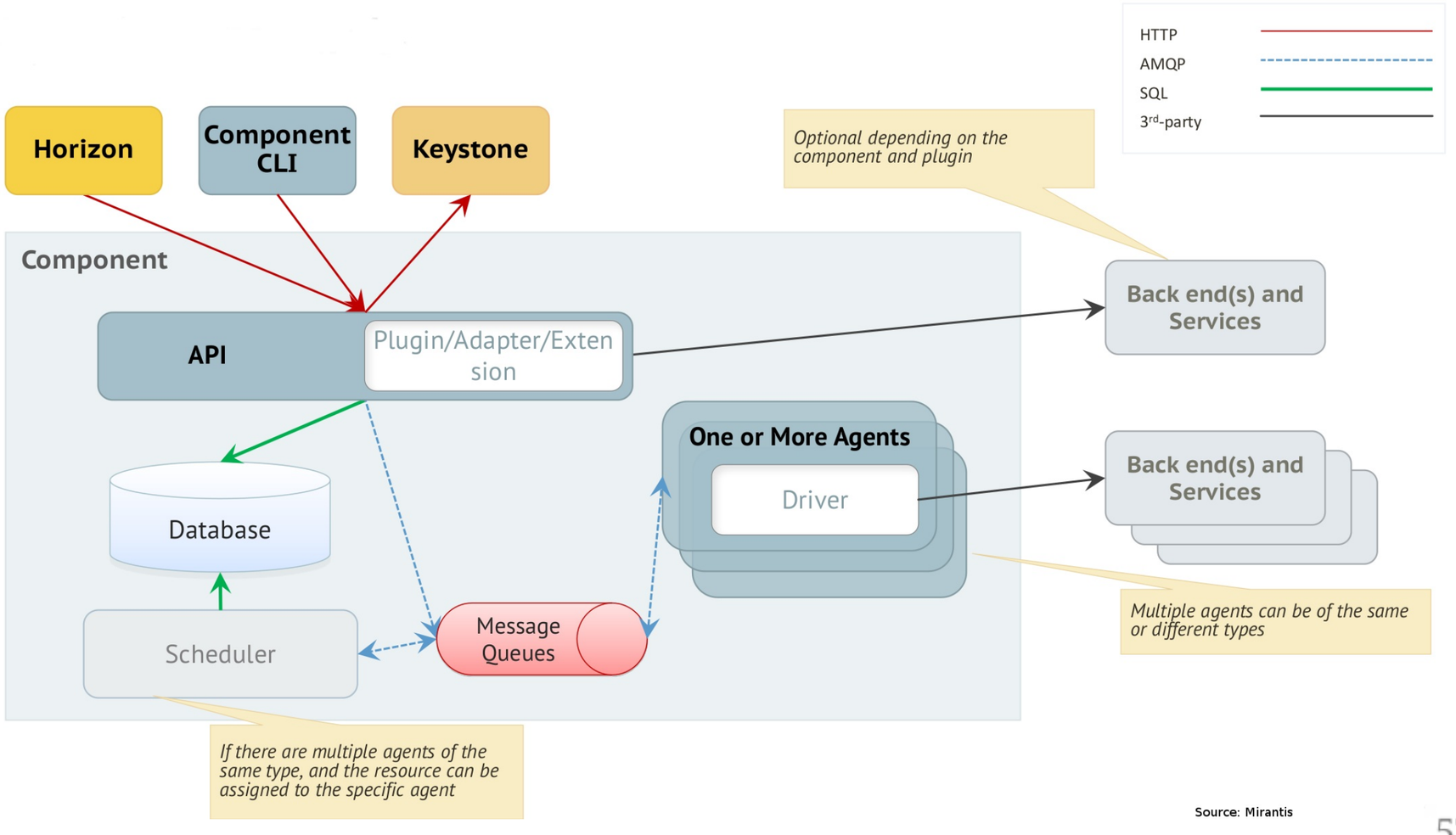
Architecture générale

API réseaux

- API REST
  - Outils CLI
  - Clients externes
  - Communications entre projets OpenStack
- Files de messages
  - Communication entre composants d'un porjet
  - Protocole AMQP (Advanced Message Queuing Protocol)
    - RabbitMQ
  - Bibliothèque oslo.messaging
    - RPC à l'aide de file de messages

Architecture d'un projet

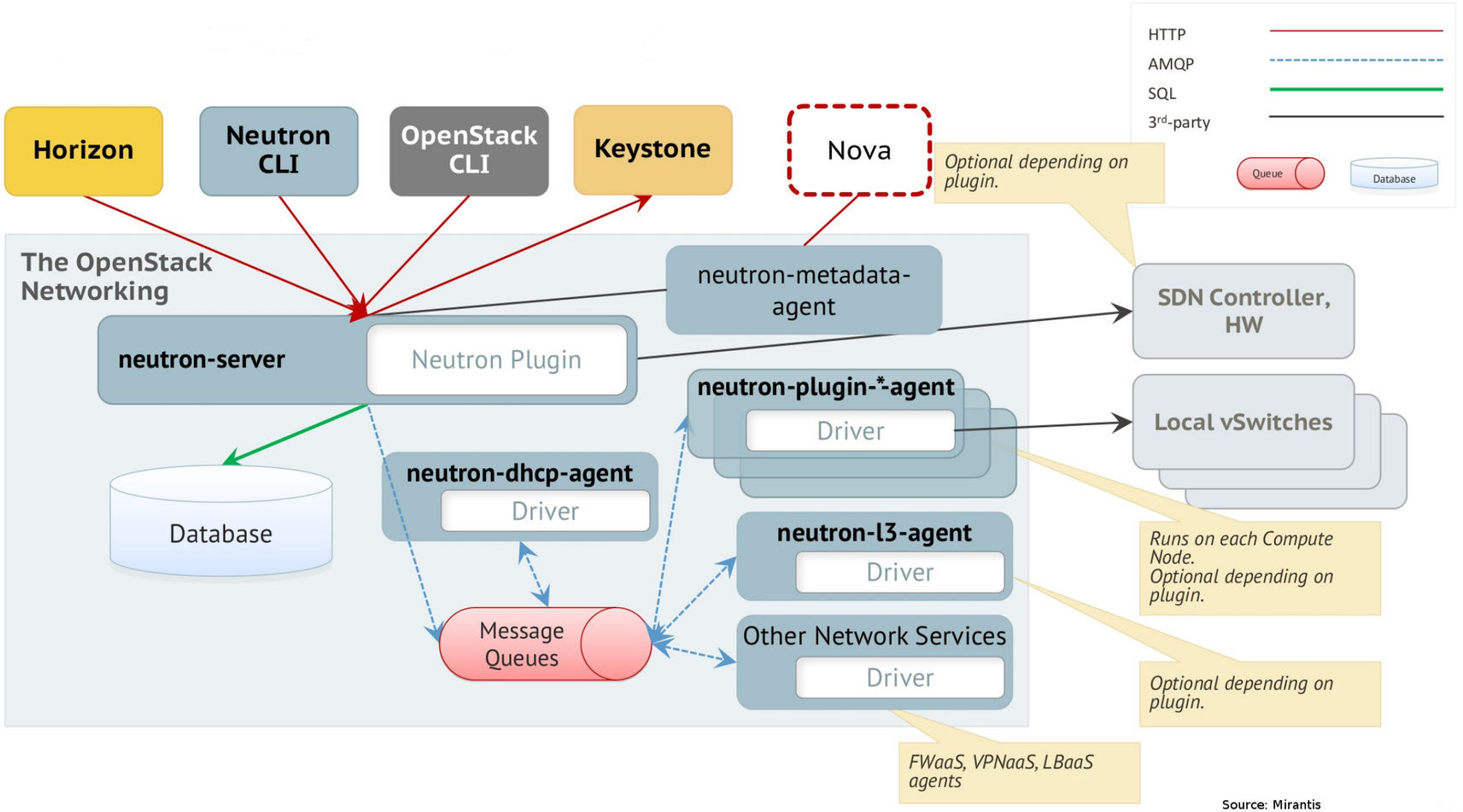
Composants typiques



Source: Mirantis

Architecture d'un projet

Ex: Neutron



Source: Mirantis

Architecture d'un projet

Etapes d'installation d'un service

- Installer un serveur MySQL et RabbitMQ
  - Généralement commun à tous les projets OpenStack déployés
  - Créer un login/mdp MySQL et RabbitMQ pour le service
  - Intialiser la DB du service (script de migration fourni)

Architecture d'un projet

Etapes d'installation d'un service

- Déclarer le service dans keystone

```
openstack user create --domain default --password <password> glance
openstack role add --project service --user glance admin
openstack service create --name glance \
            --description "OpenStack Image service" \
            image
openstack endpoint create --region RegionOne image internal <url>
openstack endpoint create --region RegionOne image external <url>
openstack endpoint create --region RegionOne image admin <url>
```

Architecture d'un projet

Etapes d'installation d'un service

- Configurer les sous-composants
  - Mots de passe d'accès à la DB, RabbitMQ et Keystone
  - Configurations propres à chaque sous-composant et chaque noeud
- Configurer la politque d'accès aux API
  - policy.json

  Lancement des sous-composants chaque noeud
  - Serveurs HTTP avec interface WSGI
  - Serveurs indépendant

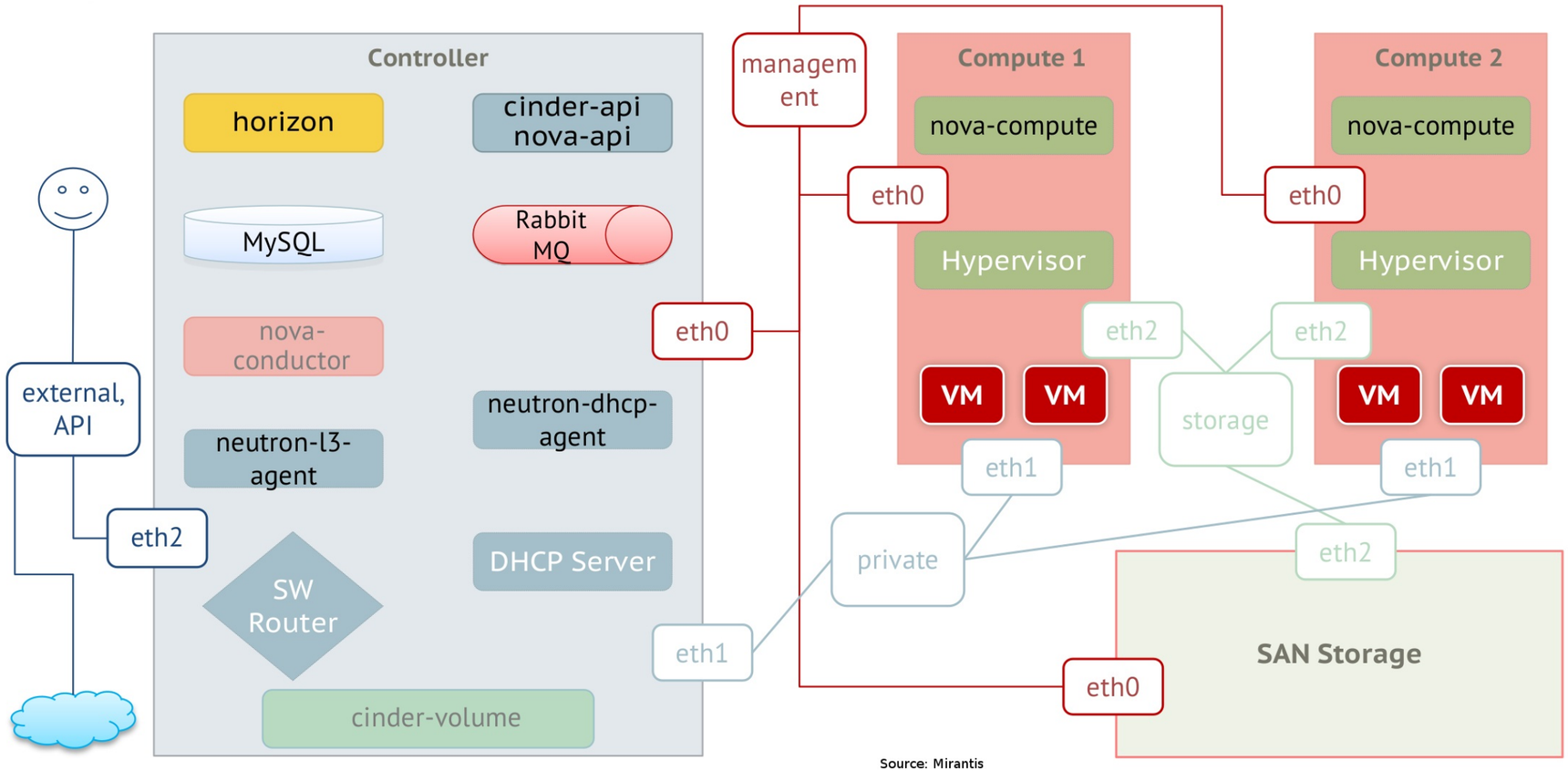
Déploiement sur un cluster

Réseaux logiques

- Réseau de gestion
  - Communications entre les composants OpenStack
- Réseau de stockage
  - Communications depuis les instances/hyperviseurs vers les stockages bloc et objet
- Réseaux privés
  - Communications privées entre les instances
- Réseaux externes
  - Communications entre les instances et l'extérieur
- Réseau API
  - Accès des utilisateurs aux API OpenStack depuis l'extérieur

Déploiement sur un cluster

Architecture d'un petit cluster OpenStack



Source: Mirantis

Mise en haute disponibilité

Composants distribués

- Ne gère que les ressources d'un noeud physique
  - Ex: ressources de calcul d'un hyperviseur
- En cas de panne le noeud physique n'est plus disponible
- Pas de HA nécessaire
  - Vérifier que le service s'exécute
  - Relance automatique si le noeud est encore fonctionnel
- Ex: nova-compute, neutron-plugin-linuxbridge-agent, …

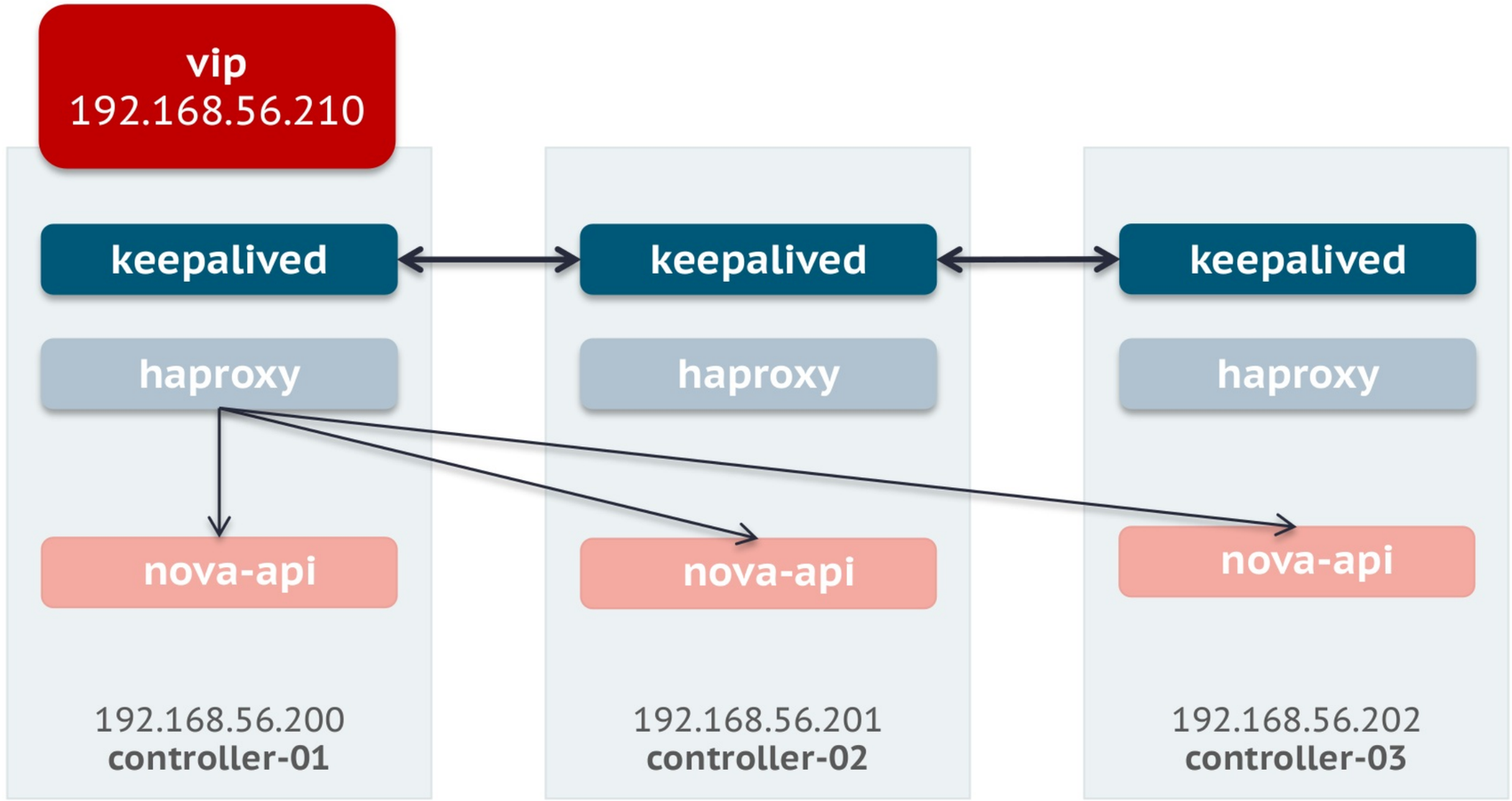
Mise en haute disponibilité

Composants centralisés

- Composants généralement *stateless*
  - Stocke ses information dans
    - File de message
    - Base de données SQL
- Serveurs d'API
  - Déployer plusieurs derrière un équilibreur de charge HTTP
  - Ex: keystone, glance-api, nova-api, neutron-server

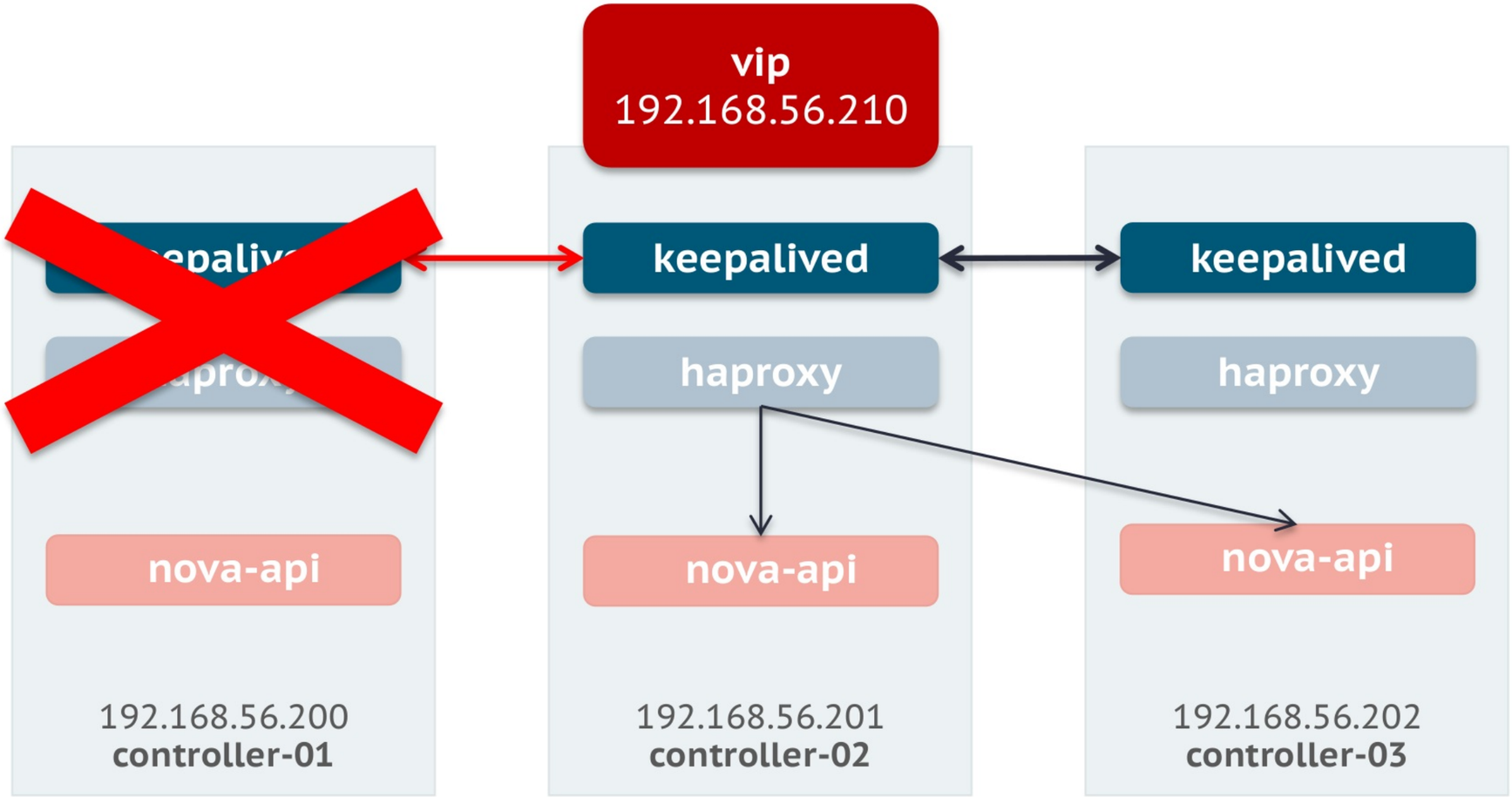
Mise en haute disponibilité

Exemple de mise en HA d'un service d'API



Mise en haute disponibilité

Exemple de mise en HA d'un service d'API



Mise en haute disponibilité

Composants centralisés

- Autres services
  - Supporte généralement une mode multi-actif
    - Permet HA et passage à l'échelle
  - Répartition des messages AMQP
  - Ex: neutron-*-agent, nova-conductor, nova/cinder-scheduler

MySQL

- Réplication synchrone
- Galera cluster

RabbitMQ

- Files mirroirs

Détails d'implémentation de Neutron

Composants réseaux Linux mis en jeu

- Interface TAP
  - Permet de recevoir/émettre des paquets depuis un programme en *userspace*
  - Interface Ethernet point à point
  - Le réseau physique est remplacé par un processus utilisateur
  - Qemu dans le cas d'une VM
    - injecte dans la VM les paquets lus sur le TAP
    - injecte dans le TAP les paquets envoyés par la VM
  - Utilisation par neutron:
    - Lorsqu'un port est attaché à une VM
    - Création d'un TAP sur l'hyperviseur

```
ip tuntap add tap[port_id] mode tap
```

Détails d'implémentation de Neutron

Composants réseaux Linux mis en jeu

- Interface Bridge
  - Switch virtuel
  - Apprend les ports de destination pour chaque MAC
    - En fonction des paquets transmis
  - Broadcast des paquets si le port correspondant à la MAC n'est pas connu
  - Utilisation par Neutron (mode Linuxbridge):
    - Un bridge par réseau sur chaque noeud concerné
      - Hyperviseur hébergeant une VM
      - Contrôleur hébergeant routeur ou agent dhcp
    - l2population: pré-peuplement des tables MAC avec les MAC des VM

```
brctl addbr brq[networkid]
brctl addif brq[networkid] tap[portid]
brctl addif brq[networkid] eth1
bridge fdb add [mac] dev [port]
```

Détails d'implémentation de Neutron

Composants réseaux Linux mis en jeu

- Interface VXLAN
  - Tunnels entre noeuds physiques
    - Taggés avec un identifiant de segmentation
    - Un identifiant associé à chaque réseau virtuel
  - Utilisation par neutron:
    - Un *endpoint* VXLAN par réseau sur chaque noeud concerné
      - Hyperviseur hébergeant une VM
      - Contrôleur hébergeant routeur ou agent dhcp

```
ip link add vxlan-[id] type vxlan vni [id]
brctl addif  brq[networkid] vxlan-[id]
bridge fdb add fa:16:3e:3c:3e:06 dev vxlan-[id] \
       dst [hostip] self permanent
```

Détails d'implémentation de Neutron

Composants réseaux Linux mis en jeu

- Namespace réseau
  - Pile réseau indépendante
    - Interfaces, IPs assignées
    - Tables de routages
    - Règles IPTables
  - Utilisation par neutron
    - Processus hôtes dans les mêmes réseaux virtuel que les VMs
    - Namespaces DHCP: qdhcp-[network_id]
      - dnsmasq (écoute sur une IP d'un réseau virtuel)
      - metadata-agent (écoute sur l'IP metadata EC2: 169.254.169.254)
    - Namespaces router: qrouter-[router_id]
      - routage entre les sous-réseaux des réseaux virtuels
      - règles IPtables

Détails d'implémentation de Neutron

Composants réseaux Linux mis en jeu

- Namespace réseau: commandes utiles
  - Lister les namespaces existants

```
ip netns list
```

  - Exécuter une commande dans un namespace

```
ip netns exec [nsname] ip addr list
```

  - Trouver le namespace dans lequel s'exécute un processus

```
ip netns identify [pid]
```

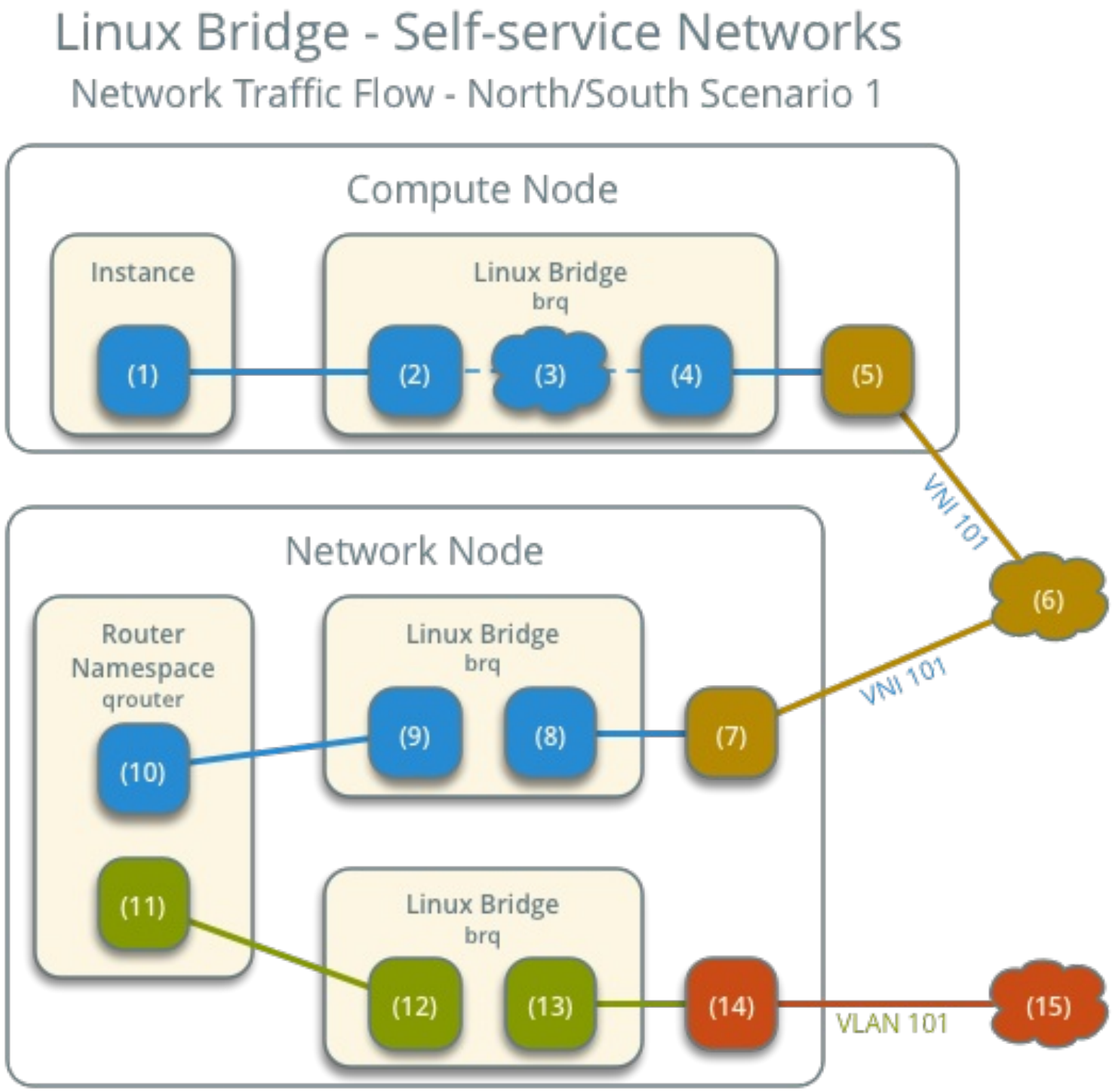
Détails d'implémentation de Neutron

Composants réseaux Linux mis en jeu

- Interfaces veth
  - Deux interfaces reliées par un lien ethernet virtuel
  - Permet de relier deux namespaces
  - Exemple:
    - Bridge associé à un réseau virtuel dans le namespace princpal
    - Ports associés au dhcp et routers dans leurs namespaces respectifs

```
ip link add dev tap[portid] type veth peer name qr[portid]
brctl addif brq[networkid] tap[portid]
sudo ip link set qr[portid] netns qrouter-[routerid]
```

Détails d'implémentation de Neutron



Détails d'implémentation de Neutron

Composants réseaux Linux mis en jeu

- IPTables
  - Filtre et transforme les paquets traversant les interfaces
  - Plusieurs chaînes et tables contenant des règles
  - Chaine: étape à laquelle la règle est appliquée
  - Table: type de règle
    - Filter: filtrage
    - NAT: traduction d'adresse
    - Mangle: modifications spécialisées (ex: QoS)
    - Raw: configuration des connexions exemptées de tracking

Détails d'implémentation de Neutron

- Chaque règle IPTables définit
  - Des critères d'application de la règle
    - protocole/ip/port source/destination
      - interace source/destination
  - Une cible
    - ACCEPT/DROP
    - [chain]
    - RETURN
    - DNAT/SNAT
      - –to-destination/–to-source
      - …
- Exemple: SNAT dans le namespace d'un routeur

```
iptables -A neutron-l3-agent-float-snat -s 172.16.1.3/32 \
     -j SNAT --to-source 10.201.0.116
```

Merci de votre attention !

Des questions ?