

OPENSTACK-ADMIN-USER

CONCERNANT CES SUPPORTS DE COURS

SUPPORTS DE COURS RÉALISÉS PAR ALTER WAY CLOUD CONSULTING

ex Osones - <https://cloud-consulting.alterway.fr>



- Copyright © 2014 - 2019 alter way Cloud Consulting
- Licence : [Creative Commons BY-SA 4.0](#)
- Sources : <https://github.com/Alterway/formations/>
- HTML/PDF : <https://osones.com/formations/>



Licence Creative Commons BY-SA 4.0

INTRODUCTION

OBJECTIFS DE LA FORMATION

- Assimiler les concepts et le vocabulaire liés au cloud
- Manipuler et orchestrer des ressources dans un cloud
OpenStack
- Définir, déployer et maintenir une infrastructure dans le
cloud
- Architecturer une application cloud-ready

PRÉ-REQUIS DE LA FORMATION

- À l'aise dans un environnement Linux (shell)
- Notions de virtualisation
- Optionnel :
 - À l'aise dans un environnement Python

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 du 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

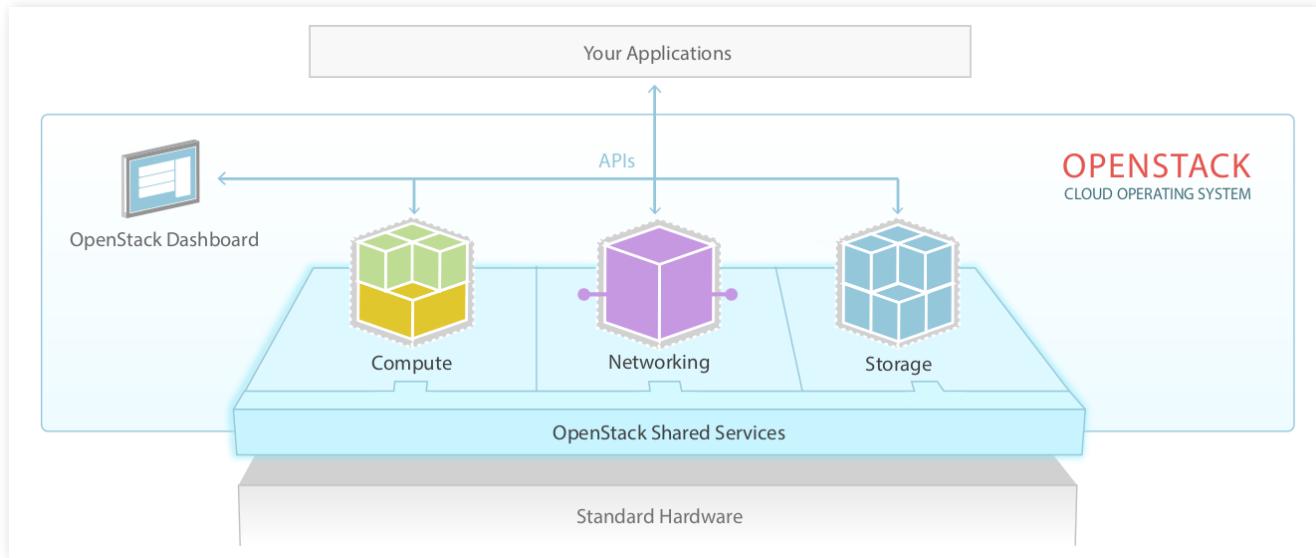
PRÉ-REQUIS DE LA FORMATION

- Compétences d'administration système Linux tel qu'Ubuntu
 - Gestion des paquets
 - Manipulation de fichiers de configuration et de services
 - LVM (Logical Volume Management) et systèmes de fichiers
- Notions :
 - Virtualisation : KVM (Kernel-Based Virtual Machine), libvirt
 - Réseau : iptables, namespaces
 - SQL
- Optionnel :
 - À l'aise dans un environnement Python

OPENSTACK : LE PROJET

TOUR D'HORIZON

VUE HAUT NIVEAU



Version simple

HISTORIQUE

- Démarrage en 2010
- Objectif : le Cloud Operating System libre
- Fusion de deux projets de Rackspace (Storage) et de la NASA (Compute)
- Logiciel libre distribué sous licence Apache 2.0
- Naissance de la Fondation en 2012

MISSION STATEMENT

To produce a ubiquitous Open Source Cloud Computin

LES RELEASES

- 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), Juno (2014.2)
- Kilo (2015.1), Liberty (2015.2)
- Mitaka (2016.1), Newton (2016.2)
- Ocata (2017.1), Pike (2017.2)
- Queens (2018.1), **Rocky** (2018.2)
- Stein (2019.1), Train (2019.2)
- Premier semestre 2020 : Ussuri

QUELQUES SOUTIENS/CONTRIBUTEURS ...

- Editeurs : Red Hat, Suse, Canonical, Vmware, ...
- Constructeurs : IBM, HP, Dell, ...
- Constructeurs/réseau : Juniper, Cisco, ...
- Constructeurs/stockage : NetApp, Hitachi, ...
- En vrac : NASA, Rackspace, Yahoo, OVH, Citrix, SAP, ...
- Google ! (depuis juillet 2015)

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

... ET UTILISATEURS

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

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

LES DIFFÉRENTS SOUS-PROJETS

<https://www.openstack.org/software/project-navigator/>

- 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

LES DIFFÉRENTS SOUS-PROJETS (2)

- Mais aussi :
 - Bare metal (Ironic)
 - Queue service (Zaqar)
 - Database Service (Trove)
 - Data processing (Sahara)
 - DNS service (Designate)
 - Shared File Systems (Manila)
 - Key management (Barbican)
 - Container (Magnum)
- Autres
 - Les clients CLI et bibliothèques
 - Les outils de déploiement d'OpenStack
 - Les bibliothèques utilisées par OpenStack
 - Les outils utilisés pour développer
OpenStack

APIS

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

LES 4 OPENS

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

<https://governance.openstack.org/tc/reference/opens.html>

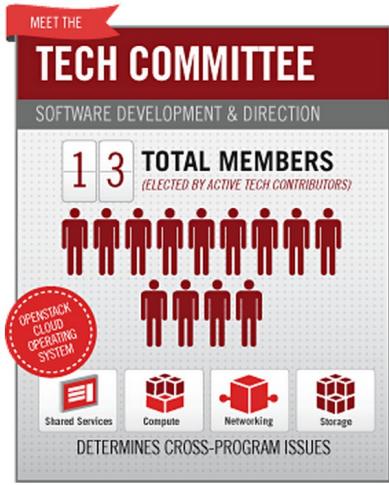
<https://www.openstack.org/four-opens/>

LA FONDATION OPENSTACK

- 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

LA FONDATION OPENSTACK

Technical Committee



Board of Directors



User Committee



Les principales entités de la Fondation

OPEN INFRASTRUCTURE

- Récemment, la Fondation OpenStack s'élargit à l'Open Infrastructure
- Au-delà d'OpenStack, nouveaux projets chapeautés :
 - Kata Containers
 - Zuul
 - Airship
 - StarlingX

RESSOURCES

- Annonces (nouvelles versions, avis de sécurité) : openstack-announce@lists.openstack.org
- Portail documentation : <https://docs.openstack.org/>
- API/SDK : <https://developer.openstack.org/>
- Gouvernance du projet : <https://governance.openstack.org/>
- Versions : <https://releases.openstack.org/>
- Support :
 - <https://ask.openstack.org/>
 - openstack-discuss@lists.openstack.org
 - [#openstack@Freenode](irc://#openstack@Freenode)

RESSOURCES

- Actualités :
 - Blog officiel : <https://www.openstack.org/blog/>
 - Planet : <http://planet.openstack.org/>
 - Superuser : <http://superuser.openstack.org/>
- Ressources commerciales :
<https://www.openstack.org/marketplace/> entre autres
- Job board : <https://www.openstack.org/community/jobs/>

USER SURVEY

- Sondage réalisé régulièrement par la Fondation (tous les 6 mois)
- Auprès des déployeurs et utilisateurs
- Données exploitables : <https://www.openstack.org/analytics>

CERTIFICATION CERTIFIED OPENSTACK ADMINISTRATOR (COA)

- La seule certification :
 - Validée par la Fondation OpenStack
 - Non liée à une entreprise particulière
- Contenu :
 - Essentiellement orientée *utilisateur* de cloud OpenStack
 - <https://www.openstack.org/coa/requirements/>
- Aspects pratiques :
 - Examen pratique, passage à distance, durée : 2,5 heures
 - Coût : \$300 (deux passages possibles)
- Ressources
 - <https://www.openstack.org/coa/>
 - Tips : <https://www.openstack.org/coa/tips/>
 - Handbook : <http://www.openstack.org/coa/handbook>
 - Exercices (non-officiels) :
<https://github.com/A.IONIURI/COA>

RESSOURCES - COMMUNAUTÉ FRANCOPHONE ET ASSOCIATION

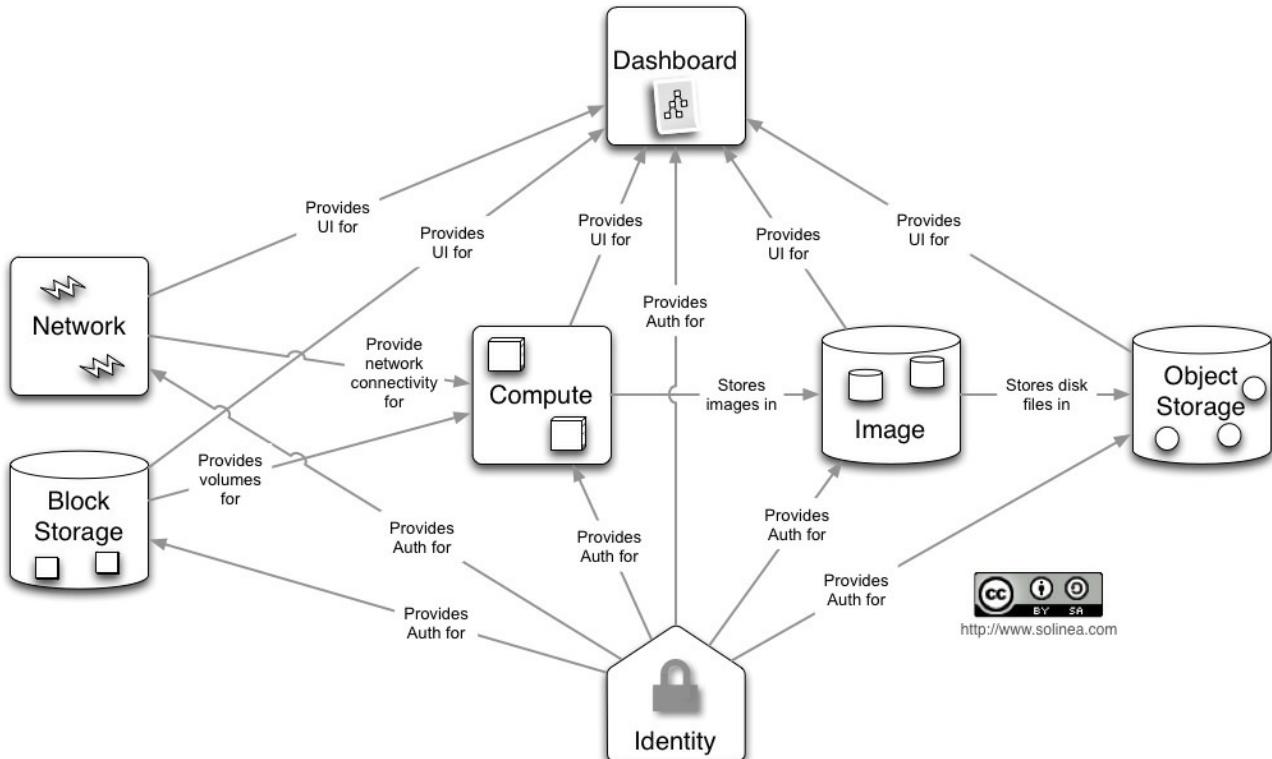


Logo OpenStack-fr

- <https://openstack.fr/> - <https://asso.openstack.fr/>
- Meetups : Paris, Lyon, Toulouse, Montréal, etc.
- OpenStack Days France (Paris) :
<https://openstackdayfrance.fr/>
- Présence à des événements tels que *Paris Open Source Summit*
- Canaux de communication :
 - openstack-fr@lists.openstack.org
 - [#openstack-fr@Freenode](irc://#openstack-fr@Freenode)

FONCTIONNEMENT INTERNE

ARCHITECTURE



Vue détaillée des services

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

IMPLÉMENTATION - DÉPENDANCES

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

MODÈLE DE DÉVELOPPEMENT

STATISTIQUES (2017)

- 2344 développeurs
- 65823 changements
(commits)

[https://www.openstack.org/assets/reports/OpenStack-
AnnualReport2017.pdf](https://www.openstack.org/assets/reports/OpenStack-AnnualReport2017.pdf)

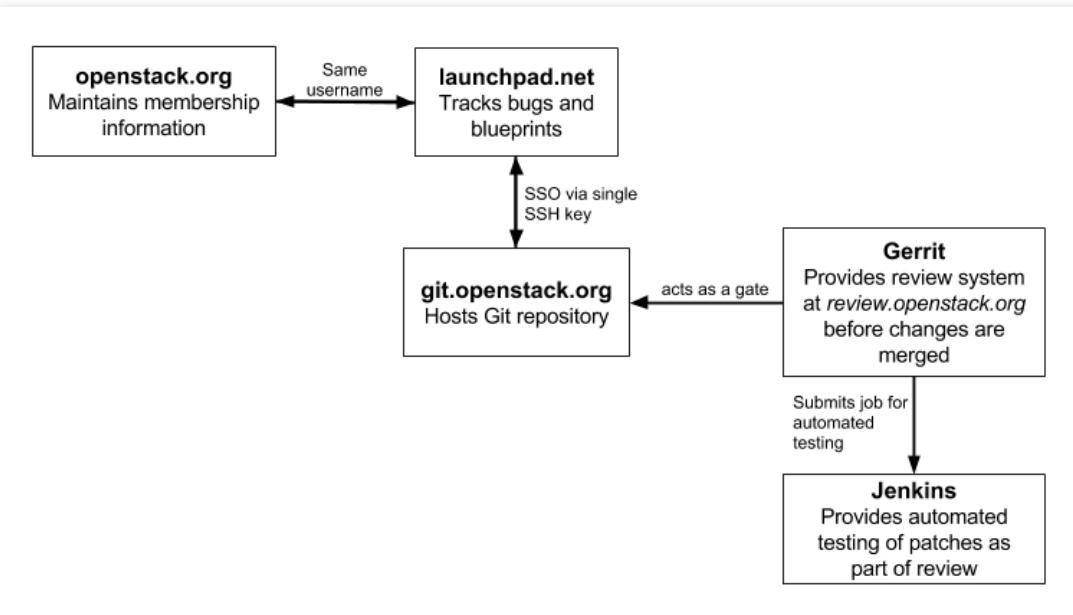
DÉVELOPPEMENT DU PROJET : EN DÉTAILS

- Ouvert à tous (individuels et entreprises)
- Cycle de développement de 6 mois
- Chaque cycle débute par un Project Team Gathering (PTG)
- Pendant chaque cycle a lieu un OpenStack Summit

LES OUTILS ET LA COMMUNICATION

- Code : Git (GitHub est utilisé comme miroir)
- Revue de code (peer review) : Gerrit
- Intégration continue (CI: Continuous Integration) : Zuul
- Blueprints/spécifications et bugs :
- Launchpad
- StoryBoard
- Communication : IRC et mailing-lists
- Traduction : Zanata

DÉVELOPPEMENT DU PROJET : EN DÉTAILS



Workflow de changements dans OpenStack

CYCLE DE DÉVELOPPEMENT : 6 MOIS

- Le planning est publié, exemple :
<https://releases.openstack.org/stein/schedule.html>
- Milestone releases
- Freezes : Feature, Requirements, String
- RC releases
- Stable releases
- Cas particulier de certains composants :
https://releases.openstack.org/reference/release_models.html

PROJETS

- Équipes projet (*Project Teams*) :
<https://governance.openstack.org/reference/projects/index.html>
- Chaque livrable est versionné indépendamment - *Semantic versioning*
- <https://releases.openstack.org/>

QUI CONTRIBUE ?

- *Active Technical Contributor* (ATC)
- Personne ayant au moins une contribution récente dans un projet OpenStack reconnu
- Droit de vote (TC et PTL)
- *Core reviewer*: ATC ayant les droits pour valider les patchs dans un projet
- *Project Team Lead* (PTL) : élu par les ATCs de chaque projet
- Stackalytics fournit des statistiques sur les contributions
<http://stackalytics.com/>

OÙ TROUVER DES INFORMATIONS SUR LE DÉVELOPPEMENT D'OPENSTACK

- Comment contribuer
- <https://docs.openstack.org/project-team-guide/>
- <https://docs.openstack.org/infra/manual/>
- Informations diverses, sur le wiki
- <https://wiki.openstack.org/>
- Les blueprints et bugs sur Launchpad/StoryBoard
- <https://launchpad.net/openstack/>
- <https://storyboard.openstack.org/>
- <https://specs.openstack.org/>

OÙ TROUVER DES INFORMATIONS SUR LE DÉVELOPPEMENT D'OPENSTACK

- Les patchs proposés et leurs reviews sont sur Gerrit
- <https://review.openstack.org/>
- L'état de la CI (entre autres)
- <http://status.openstack.org/>
- Le code (Git) et les tarballs sont disponibles
- <https://git.openstack.org/>
- <https://tarballs.openstack.org/>
- IRC
- Réseau Freenode
- Logs discussions et infos réunions :
<http://eavesdrop.openstack.org/>
- Mailing-lists
- <http://lists.openstack.org/>

UPSTREAM TRAINING

- Deux jours de formation
- Apprendre à devenir contributeur à OpenStack
- Les outils
- Les processus
- Travailler et collaborer de manière ouverte

OPENSTACK INFRA

- Équipe projet en charge de l'infrastructure de développement d'OpenStack
- Travaille comme les équipes de dev d'OpenStack et utilise les mêmes outils
- Résultat : Infrastructure as code open source
<https://opensourceinfra.org/>
- Utilise du cloud (hybride)
- Développe certains outils
- Zuul
- yaml2ical

OPENSTACK SUMMIT

- Tous les 6 mois en milieu de cycle de développement
- Aux USA jusqu'en 2013, aujourd'hui alternance Amérique de Nord et Asie/Europe
- Quelques dizaines au début à des milliers de participants aujourd'hui
- En parallèle : conférence (utilisateurs, décideurs) et Forum (développeurs/opérateurs, remplace une partie du précédent Design Summit)
- Détermine le nom de la prochaine release : lieu/ville à proximité du Summit

EXEMPLE DU SUMMIT D'AVRIL 2013 À PORTLAND

openstack summit

users
devs

PORLAND//2013



EXEMPLE DU SUMMIT D'OCTOBRE 2015 À TOKYO



Photo : Elizabeth K. Joseph, CC BY 2.0, Flickr/pleia2

EXEMPLE DU SUMMIT D'OCTOBRE 2015 À TOKYO



Photo : Elizabeth K. Joseph, CC BY 2.0, Flickr/pleia2

EXEMPLE DU SUMMIT D'OCTOBRE 2015 À TOKYO

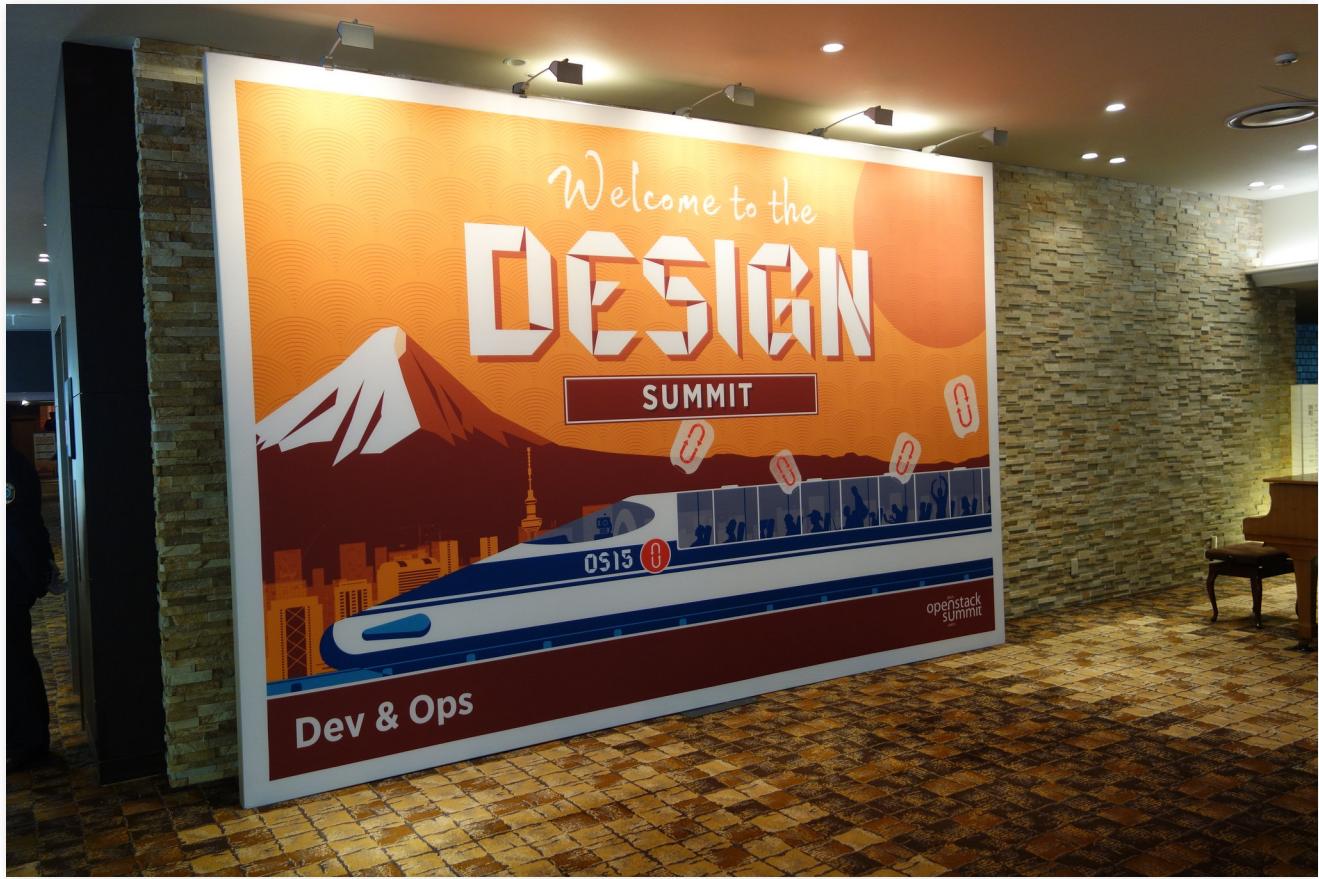
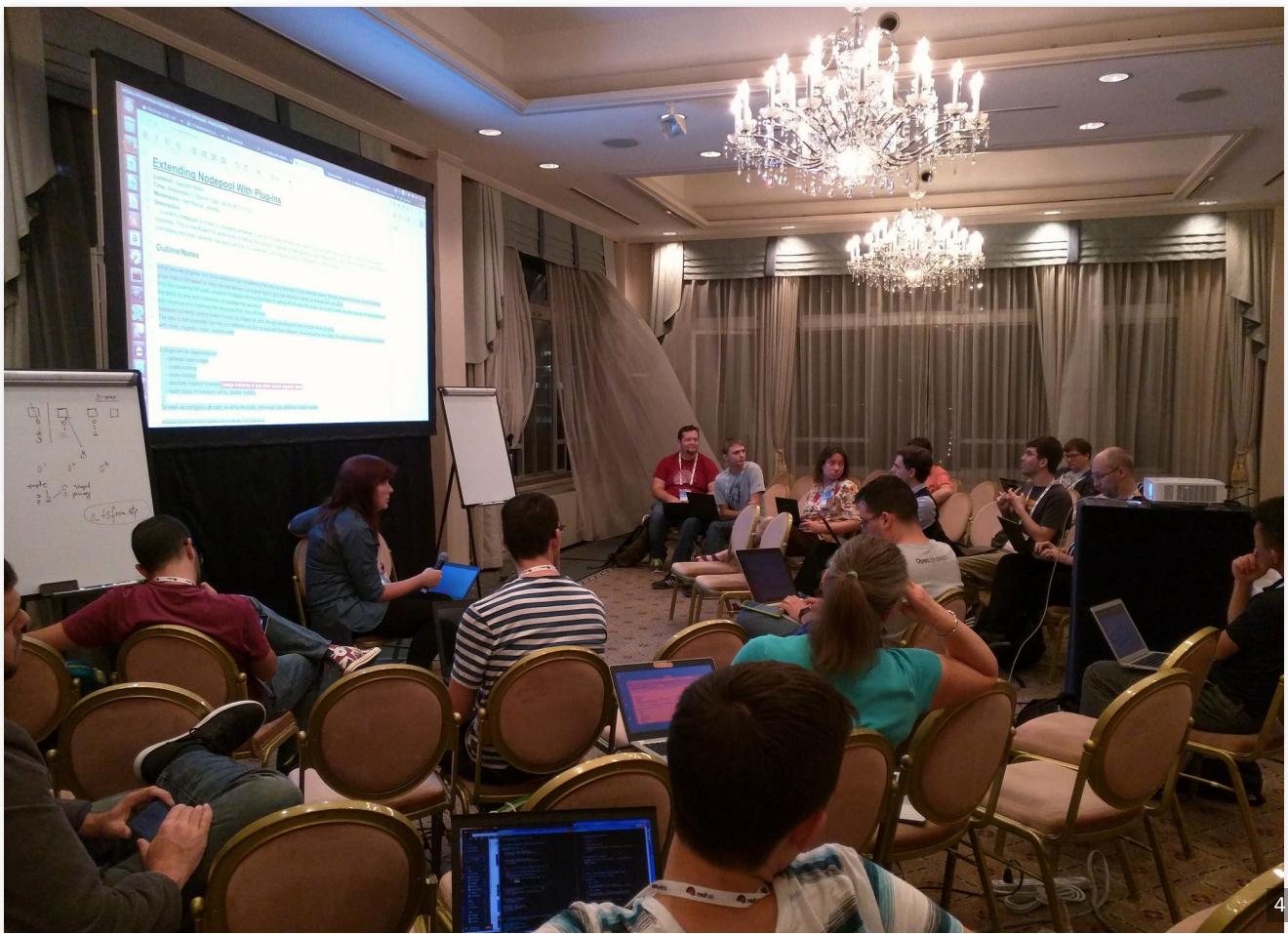


Photo : Elizabeth K. Joseph, CC BY 2.0, Flickr/pleia2

EXEMPLE DU SUMMIT D'OCTOBRE 2015 À TOKYO



PROJECT TEAM GATHERING (PTG)

- Depuis 2017
- Au début de chaque cycle
- Remplace une partie du précédent Design Summit
- Dédié aux développeurs

TRADUCTION

- Équipe officielle *i18n*
- Seules certaines parties sont traduites, comme Horizon
- La traduction française est aujourd'hui une des plus avancées
- Utilisation d'une plateforme web basée Zanata :
<https://translate.openstack.org/>

DEVESTACK : FAIRE TOURNER RAPIDEMENT UN OPENSTACK

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 de la production

FONCTIONNEMENT DE 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

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
```

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

UTILISER OPENSTACK

LE PRINCIPE

- 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

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>

ACCÈS AUX APIS

- 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)

CLIENTS OFFICIELS

- 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 credentials par paramètres, variables d'environnement ou fichier de configuration
- L'option `--debug` affiche la communication HTTP

OPENSTACK CLIENT

- 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`

<https://docs.openstack.org/python-openstackclient/latest/configuration/index.html#configuration-files>

KEYSTONE : AUTHENTIFICATION, AUTORISATION ET CATALOGUE DE SERVICES

PRINCIPES

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

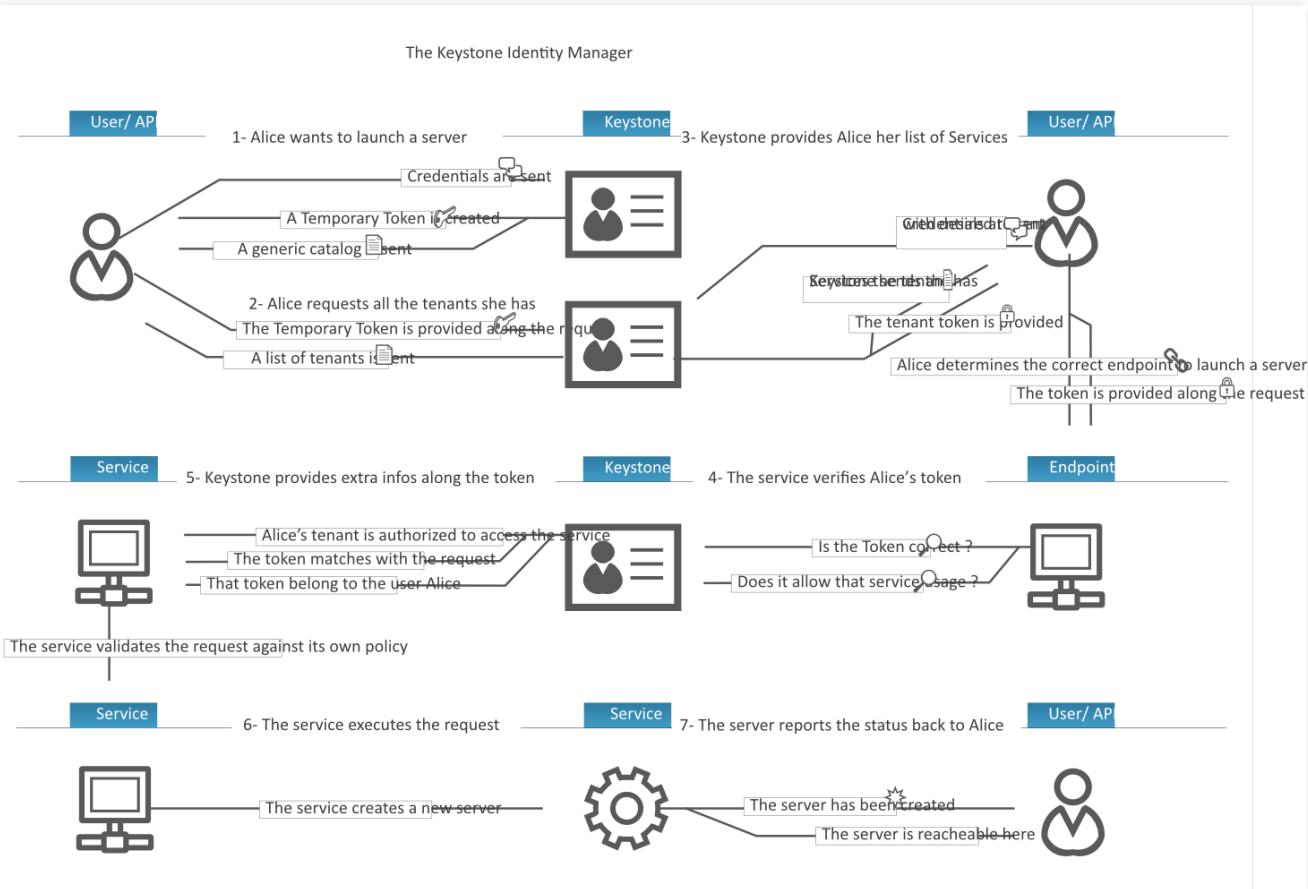
API

- 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)

CATALOGUE DE SERVICES

- Pour chaque service, plusieurs endpoints sont possibles en fonction de :
 - la région
 - le type d'interface (public, internal, admin)

SCÉNARIO D'UTILISATION TYPIQUE



Interactions avec Keystone

NOVA : COMPUTE

PRINCIPES

- 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

PROPRIÉTÉS D'UNE INSTANCE

- 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

API

Ressources gérées :

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

ACTIONS SUR LES INSTANCES

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

GLANCE : REGISTRE D'IMAGES

PRINCIPES

- Registre d'images et de snapshots
- Propriétés sur les images

API

- API v2 : version courante, gère images et snapshots
- API artifacts : version future, plus généraliste

TYPES D'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

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

PARTAGE DES IMAGES

- 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)

TÉLÉCHARGER DES 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/>

NEUTRON : RÉSEAU

API

L'API permet notamment de manipuler ces 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

LES 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

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

FONCTIONNALITÉS SUPPLÉMENTAIRES

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

- Load Balancing
- Firewall : diffère des groupes de sécurité
- VPN : permet d'accéder à un réseau privé sans IP flottantes
- QoS

CINDER : STOCKAGE BLOCK

PRINCIPES

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

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)

HEAT : ORCHESTRATION

GÉNÉRALITÉS

- 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

UN TEMPLATE HEAT ORCHESTRATION TEMPLATE (HOT)

parameters - resources - outputs

```
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
```

CONSTRUIRE UN TEMPLATE À PARTIR D'EXISTANT

Multiples projets en cours de développement

- Flame (Cloudwatt)
- HOT builder
- Merlin

HORIZON : DASHBOARD WEB

PRINCIPES

- 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

UTILISATION

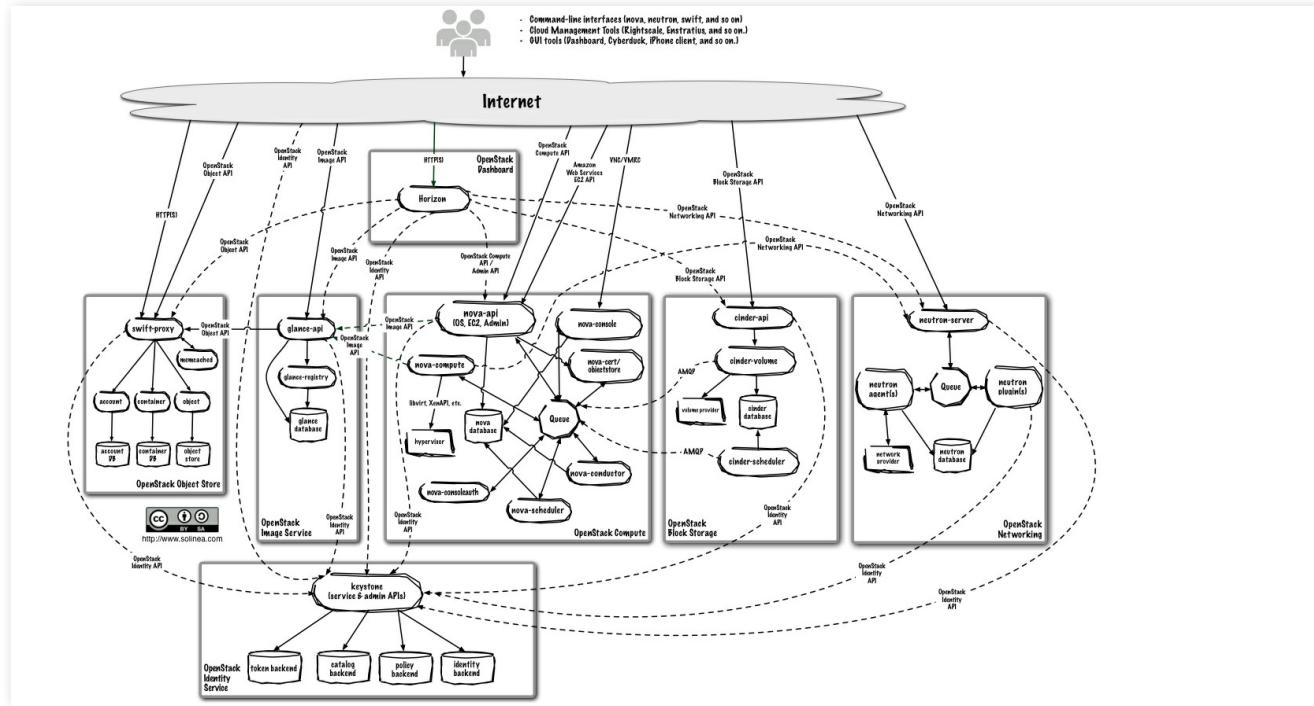
- Une interface par projet (possibilité de switcher)
- Catalogue de services accessible
- Téléchargement d'un fichier de configuration
clouds.yaml
- Une zone “admin” restreinte

DÉPLOYER OPENSTACK

CE QU'ON VA VOIR

- Installer OpenStack à la main
<https://docs.openstack.org/install-guide/>
- Donc comprendre son fonctionnement
- Passer en revue chaque composant plus en détails
- Tour d'horizon des solutions de déploiement

ARCHITECTURE DÉTAILLÉE

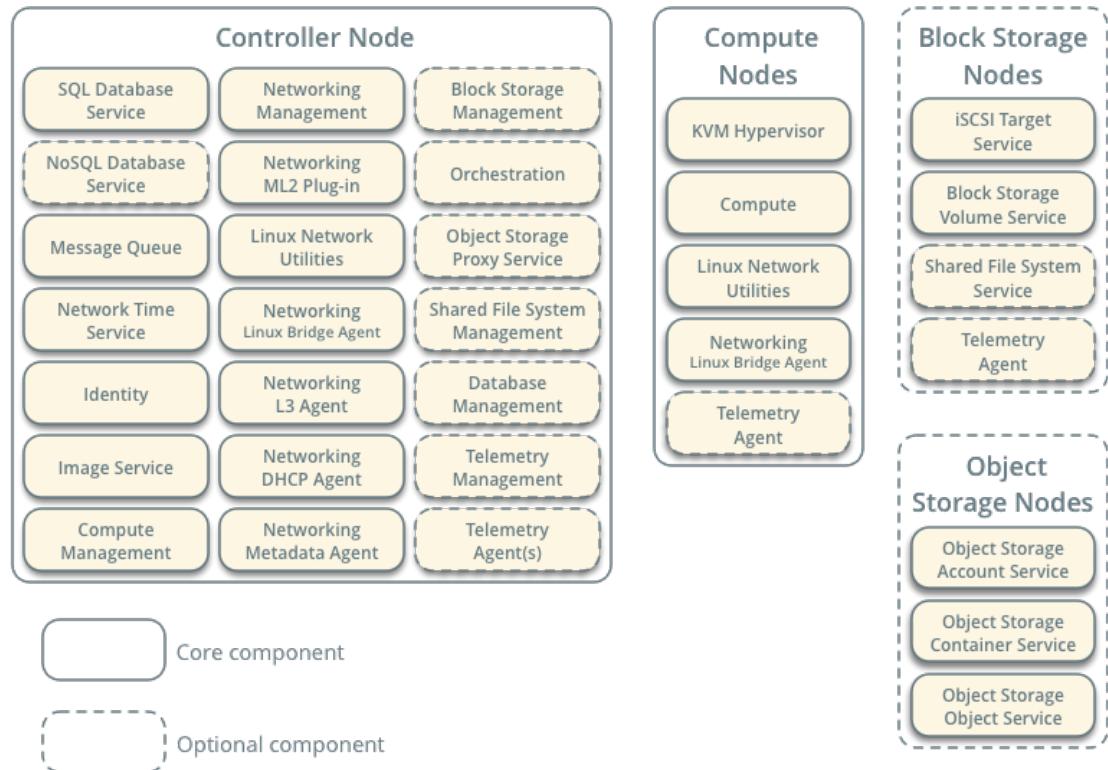


Vue détaillée des services

ARCHITECTURE SERVICES

Networking Option 2: Self-Service Networks

Service Layout



QUELQUES ÉLÉMENTS DE CONFIGURATION GÉNÉRAUX

- 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

SYSTÈME D'EXPLOITATION

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

PYTHON



Logo Python

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

BASE DE DONNÉES MYSQL/MARIADB

- 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



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



Logo RabbitMQ

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

"HELLO WORLD" RABBITMQ

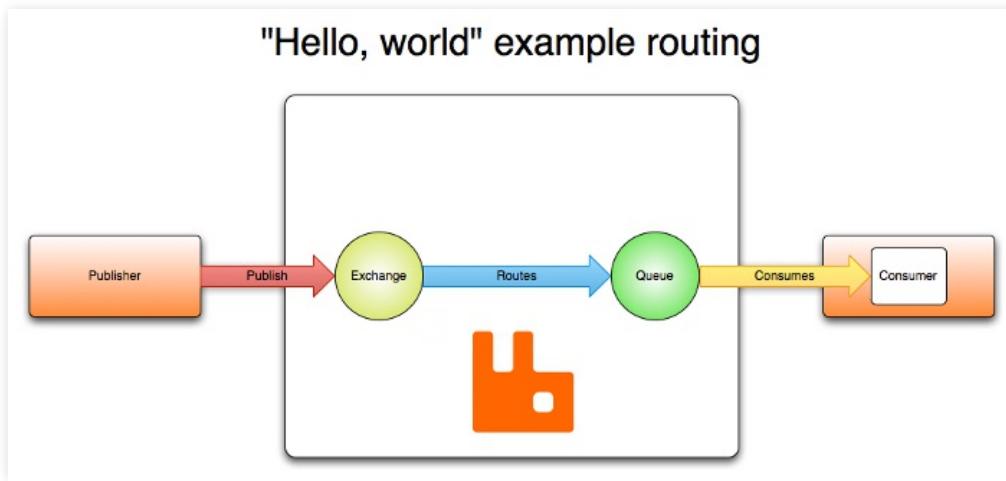


Illustration simple du fonctionnement de RabbitMQ

CACHE MEMCACHED

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

KEYSTONE : AUTHENTIFICATION, AUTORISATION ET CATALOGUE DE SERVICES

INSTALLATION ET CONFIGURATION

- Paquet APT : keystone
- Intégration serveur web WSGI (Apache par défaut)
- Fichier de configuration: /etc/keystone/keystone.conf
- Backends utilisateurs/groupes : SQL, LDAP (ou Active Directory)
- Backends projets/rôles/services/endpoints : SQL
- Backends tokens : SQL, Memcache, aucun (suivant le type de tokens)

DRIVERS POUR TOKENS

- Uuid
- PKI
- Fernet

BOOTSTRAP

- Création des services et endpoints (à commencer par Keystone)
- Création d'utilisateurs, groupes, domaines
- Fonctionnalité de bootstrap

NOVA : COMPUTE

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

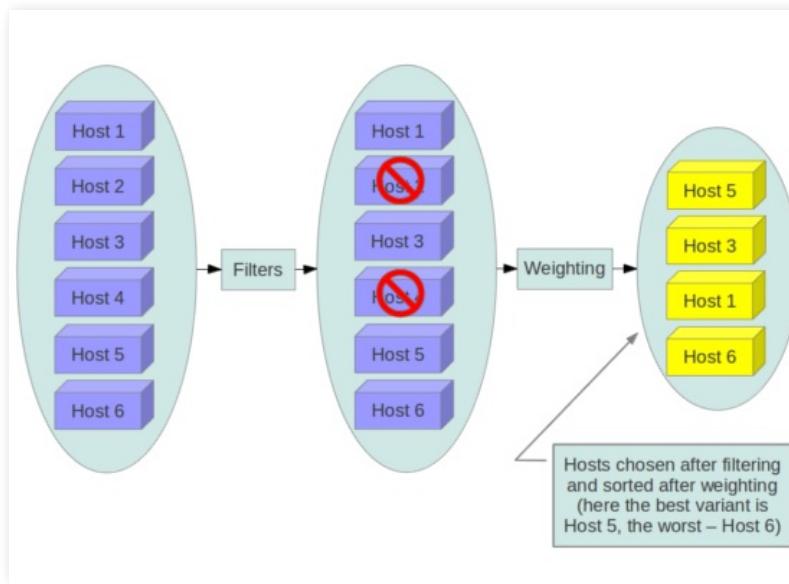
NOVA COMPUTE

- Pilote les instances (machines virtuelles ou physiques)
- Tire partie de libvirt ou d'autres APIs comme XenAPI
- Drivers : libvirt (KVM, LXC, etc.), XenAPI, VMWare vSphere, Ironic
- Permet de récupérer les logs de la console et une console VNC

NOVA SCHEDULER

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

LE SCHEDULER NOVA EN ACTION



Fonctionnement de nova-scheduler

NOVA CONDUCTOR

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

GLANCE : REGISTRE D'IMAGES

BACKENDS

- Swift ou S3
- Ceph
- HTTP
- Répertoire local

INSTALLATION

- Paquet APT : glance-api

NEUTRON : RÉSEAU EN TANT QUE SERVICE

PRINCIPES

- *Software Defined Networking* (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 : LinuxBridge par défaut, d'autres implémentations libres/propriétaires, logicielles/matérielles existent

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

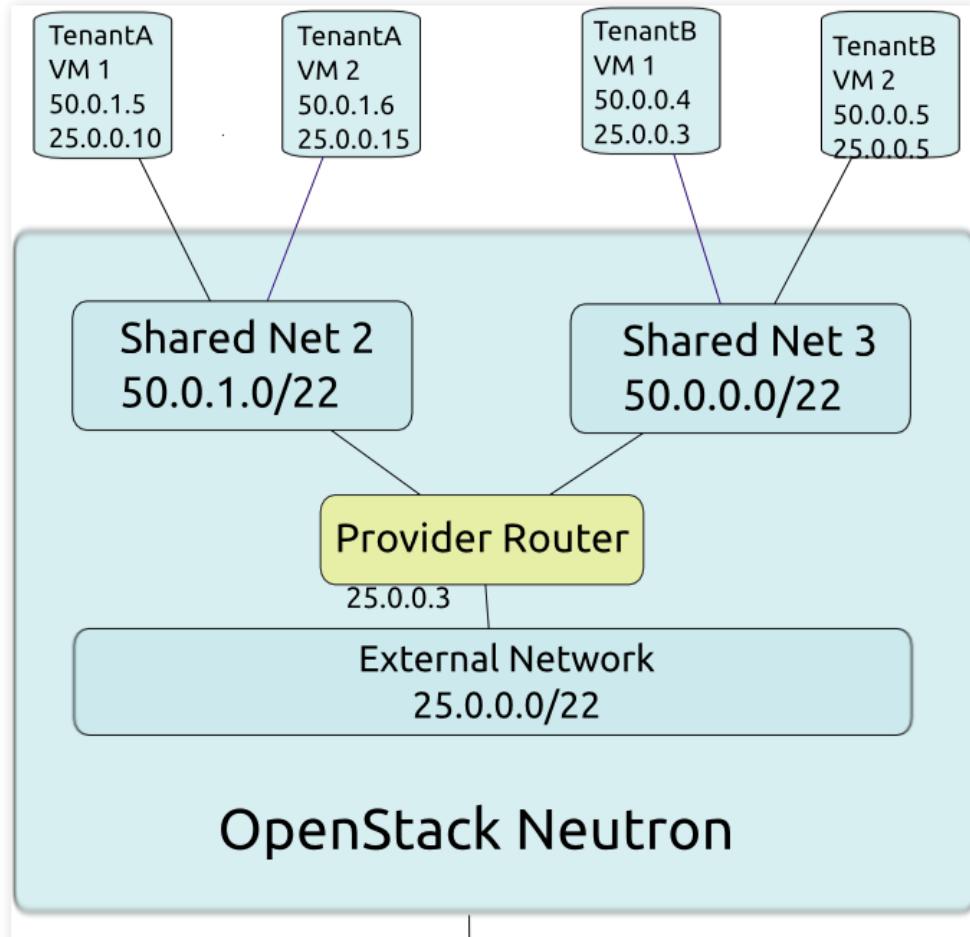
PLUGINS ML2

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

IMPLEMENTATION

- Chaque réseau est un *bridge*
- Les bridges sont étendus entre les machines via des tunnels (type VXLAN) si nécessaires
- 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

SCHÉMA



25.0.0.1



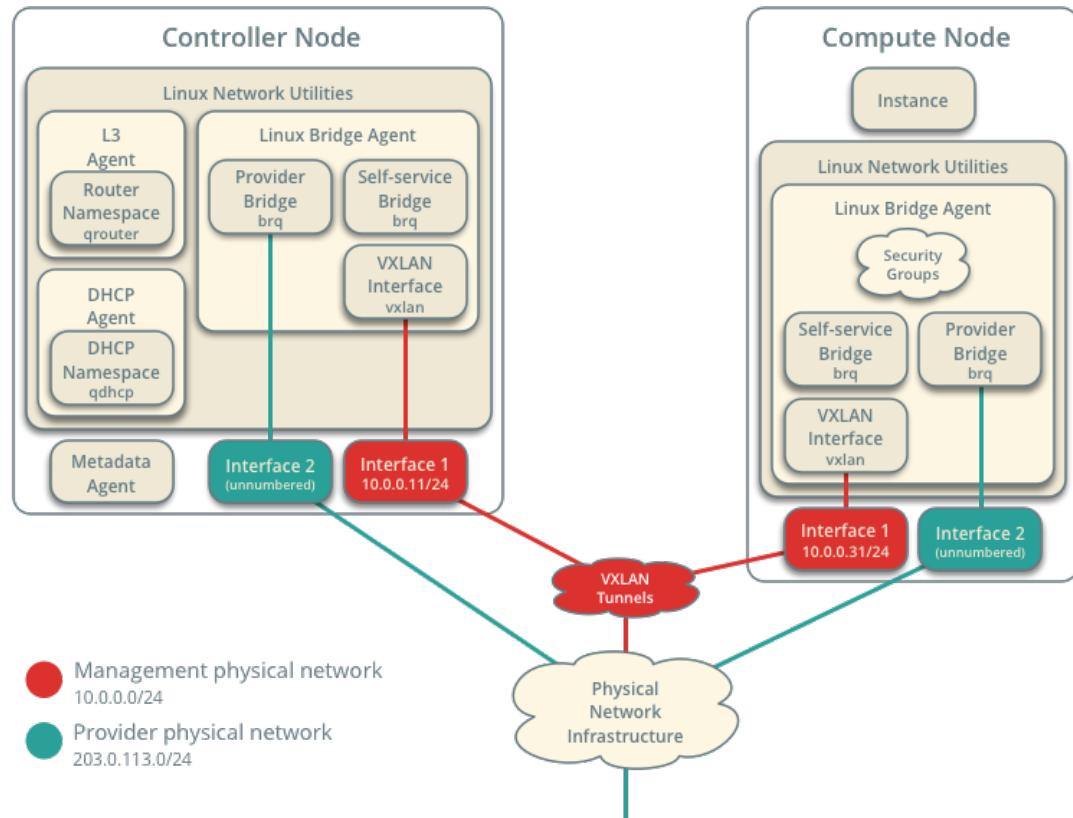
Physical Router

Vue utilisateur du réseau

SCHÉMA

Networking Option 2: Self-Service Networks

Overview



CINDER : STOCKAGE BLOCK

PRINCIPES

- Auparavant nova-volume
- Fournit des volumes
- Attachement des volumes via iSCSI par défaut

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 (optionnel) : backup vers un object store

BACKENDS

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

HORIZON : DASHBOARD WEB

PRINCIPES

- Horizon est un module Django
- OpenStack Dashboard est l'implémentation officielle de ce module



Logo du framework web Python Django

CONFIGURATION

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

SWIFT : STOCKAGE OBJET

PRINCIPES

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

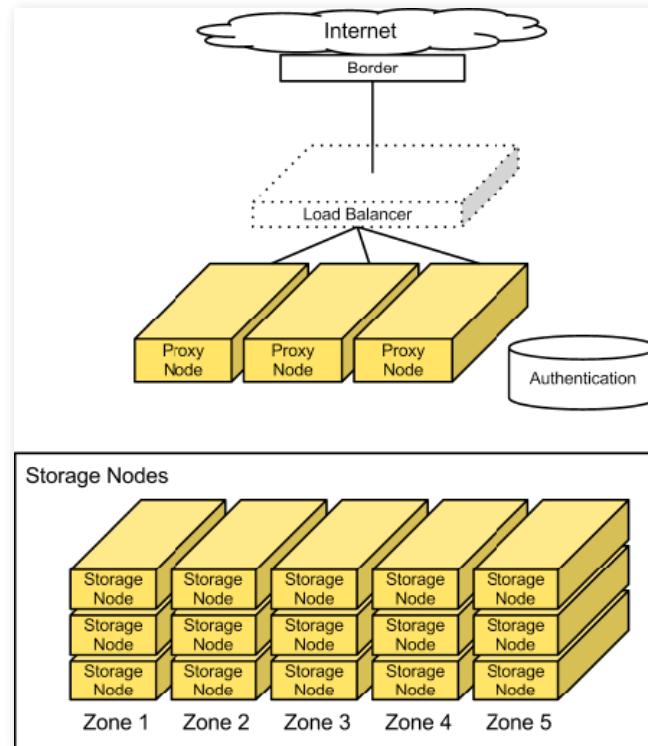
IMPLÉMENTATION

- 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



Architecture Swift

CEILOMETER : COLLECTE DE MÉTRIQUES

SURVEILLER L'UTILISATION DE SON INFRASTRUCTURE AVEC CEILOMETER

- Indexer et stocker différentes métriques concernant l'utilisation des différents services du cloud
- Fournir des APIs permettant de récupérer ces données
- Base pour construire des outils de facturation (exemple : CloudKitty)

CEILOMETER

- Récupère les données et les stocke
- Historiquement : stockage MongoDB
- Aujourd'hui : stockage Gnocchi

GNOCCHI : TIME-SERIES DATABASE

- Pourquoi Gnocchi ? Palier aux problème de scalabilité de Ceilometer + MongoDB
- Initié par des développeurs de Ceilometer et intégré à l'équipe projet Ceilometer
- Fournit une API pour lire et écrire les données
- Se base sur une BDD relationnelle et un système de stockage objet

HEAT : ORCHESTRATION DES RESSOURCES

ARCHITECTURE

- heat-api
- heat-engine

QUELQUES AUTRES COMPOSANTS INTÉRESSANTS

TROVE : DATABASE AS A SERVICE

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

DESIGNATE : DNS AS A SERVICE

- Gère différents backends : BIND, PowerDNS, etc.

BARBICAN : KEY MANAGEMENT AS A SERVICE

- Backends possibles
 - Fichiers chiffrés
 - PKCS#11
 - KMIP
 - Dogtag

MAGNUM : CONTAINER INFRASTRUCTURE AS A SERVICE

- Backends: Swarm,
Kubernetes

OPENSTACK EN PRODUCTION ET OPÉRATIONS

BONNES PRATIQUES POUR UN DÉPLOIEMENT EN PRODUCTION

QUELS COMPOSANTS DOIS-JE INSTALLER ?

- Keystone est indispensable
- L'utilisation de Nova va de paire avec Glance et Neutron
- Cinder et Swift s'apprécient en fonction des besoins de stockage
- Swift peut être utilisé indépendamment des autres composants
- Heat coûte peu
- Les services plus haut niveau s'évaluent au cas par cas

<https://docs.openstack.org/arch-design/>

PENSER DÈS LE DÉBUT AUX CHOIX STRUCTURANTS

- Distribution et méthode de déploiement
- Politique de mise à jour
- Drivers/backends : hyperviseur, stockage block, etc.
- Réseau : quelle architecture et quels drivers

LES DIFFÉRENTES MÉTHODES D'INSTALLATION

- DevStack est à oublier pour la production
- Le déploiement à la main comme vu précédemment n'est pas recommandé car peu maintenable
- Distributions OpenStack packagées et prêtes à l'emploi
- Distributions classiques et gestion de configuration
- Déploiement continu

MISES À JOUR ENTRE VERSIONS MAJEURES

- OpenStack supporte les mises à jour $N \rightarrow N+1$
- Swift : très bonne gestion en mode *rolling upgrade*
- Autres composants : tester préalablement avec vos données
- Lire les release notes
- Cf. articles de blog du CERN

<https://techblog.web.cern.ch/techblog/>

MISES À JOUR DANS UNE VERSION STABLE

- Fourniture de correctifs de bugs majeurs et de sécurité
- OpenStack intègre ces correctifs sous forme de patchs dans la branche stable
- Publication de *point releases* intégrant ces correctifs issus de la branche stable
- Durée variable du support des versions stables, dépendant de l'intérêt des intégrateurs

ASSIGNER DES RÔLES AUX MACHINES

Beaucoup de documentations font référence à ces rôles :

- Controller node : APIs, BDD, AMQP
- Network node : DHCP, routeur, IPs flottantes
- Compute node : Hyperviseur/pilotage des instances

Ce modèle simplifié n'est pas HA.

HAUTE DISPONIBILITÉ

Haute disponibilité du IaaS

- MySQL/MariaDB, 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 : <https://docs.openstack.org/ha-guide/>

Conférences par Florian Haas, Hastexo :

<https://www.openstack.org/community/speakers/profile/398/florian-haas>

HAUTE DISPONIBILITÉ DE L'AGENT L3 DE NEUTRON

- *Distributed Virtual Router* (DVR)
- L3 agent HA (VRRP)

CONSIDÉRATIONS APIs

- Des URLs uniformes pour toutes les APIs :
 - Utiliser un reverse proxy
 - Mettre à jour le catalogue de services
- Apache/mod_wsgi pour servir les APIs lorsque cela est possible (Keystone, etc.)

Guide Operations : <https://docs.openstack.org/openstack-ops/content/>

DÉCOUPAGE RÉSEAU

- Management network : réseau d'administration
- Data/instances network : réseau pour la communication inter instances
- External network : réseau externe, dans l'infrastructure réseau existante
- Storage network : réseau pour le stockage Cinder/Swift
- 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/MariaDB et RabbitMQ
- Un accès MySQL/MariaDB par base et par service
- Un utilisateur Keystone par service
- Limiter l'accès en lecture des fichiers de configuration (mots de passe, token)
- Veille sur les failles de sécurité : OSSA (*OpenStack Security Advisory*), OSSN (... *Notes*)

Guide sécurité : <https://docs.openstack.org/security-guide/>

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

https://docs.openstack.org/openstack-ops/content/scaling.html#segregate_cloud

HOST AGGREGATES / AGRÉGATS D'HÔTES

- Spécifique Nova
- L'administrateur définit des agrégats d'hôtes via l'API
- L'administrateur associe flavors et agrégats via des couples clé/valeur communs
- 1 agrégat \equiv 1 point commun, ex : GPU
- L'utilisateur choisit un agrégat à travers son choix de flavor à la création d'instance

AVAILABILITY ZONES / ZONES DE DISPONIBILITÉ

- Spécifique Nova et Cinder
- Groupes d'hôtes
- Découpage en termes de disponibilité : Rack, Datacenter, etc.
- L'utilisateur choisit une zone de disponibilité à la création d'instance
- L'utilisateur peut demander à ce que des instances soient démarrées dans une même zone, ou au contraire dans des zones différentes

RÉGIONS

- Générique OpenStack
- 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)

CELLS / CELLULES

- Spécifique Nova
- 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)

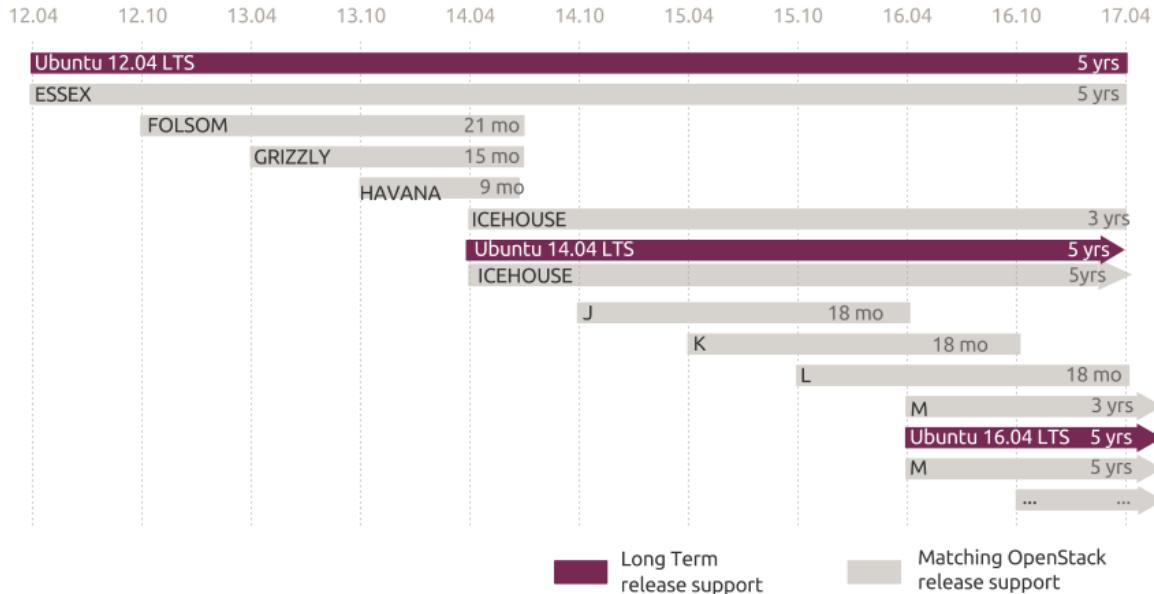
PACKAGING D'OPENSTACK - UBUNTU

- 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

UBUNTU CLOUD ARCHIVE (UCA)

Openstack Support Model

LTS support 5 years, non LTS support 18 months.



Support d'OpenStack dans Ubuntu via l'UCA

PACKAGING D'OPENSTACK DANS LES AUTRES DISTRIBUTIONS

- OpenStack est intégré dans les dépôts officiels de Debian
- Red Hat propose RHOS/RDO (déploiement basé sur TripleO)
- Comme Ubuntu, le cycle de release de Fedora est synchronisé avec celui d'OpenStack

LES DISTRIBUTIONS OPENSTACK

- StackOps : historique
- Mirantis : Fuel
- HP Helion : Ansible custom
- etc.

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 nœuds compute lorsque cela est nécessaire
- Fonctionne conjointement avec Ironic pour le déploiement bare-metal

DÉPLOIEMENT BARE-METAL

- Le déploiement des hôtes physiques OpenStack peut se faire à l'aide d'outils dédiés
- MaaS (Metal as a Service), par Ubuntu/Canonical : se couple avec Juju
- Crowbar / OpenCrowbar (initialement Dell) : utilise Chef
- eDeploy (eNovance) : déploiement par des images
- Ironic via TripleO

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, PLAYBOOKS ANSIBLE

- *Puppet OpenStack* et *OpenStack Ansible*: modules Puppet et playbooks Ansible
- Développés au sein du projet OpenStack
- <https://wiki.openstack.org/wiki/Puppet>
- <https://docs.openstack.org/developer/openstack-ansible/install-guide/>

DÉPLOIEMENT CONTINU

- OpenStack maintient un master (trunk) toujours stable
- Possibilité de déployer au jour le jour le master (CD : *Continuous Delivery*)
- Nécessite la mise en place d'une infrastructure importante
- Facilite les mises à jour entre versions majeures

TEST ET VALIDATION : 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
- Cf. aussi Rally

GÉRER LES PROBLÈMES

LES RÉFLEXES EN CAS D'ERREUR OU DE COMPORTEMENT ERRONÉ

- Travaille-t-on sur le bon projet ?
- 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 ou une API renvoie 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

OPÉRATIONS

GESTION DES LOGS

- Centraliser les logs
- Logs d'API
- Logs autres composants
OpenStack
- Logs BDD, AMQP, etc.

BACKUP

- Bases de données
- Mécanisme de déploiement, plutôt que les fichiers de configuration

MONITORING

- Réponse des APIs
- Vérification des services OpenStack et dépendances

UTILISATION DES QUOTAS

- Limiter le nombre de ressources allouables
- Par utilisateur ou par projet
- Support dans Nova
- Support dans Cinder
- Support dans Neutron

ARCHITECTURES CLOUD-READY

CONCEVOIR UNE APPLICATION POUR LE CLOUD

12-FACTOR

“The Twelve-Factor App” <https://12factor.net/>

- Écrit par Heroku
- Suivre (tout) le code dans un VCS
- Configuration

ADAPTER OU PENSER SES APPLICATIONS “CLOUD READY” 1/3

Cf. les design tenets du projet OpenStack et Twelve-Factor

<https://12factor.net/>

- Architecture distribuée plutôt que monolithique
 - Facilite le passage à l'échelle
 - Limite les domaines de *failure*
- Couplage faible entre les composants

ADAPTER OU PENSER SES APPLICATIONS “CLOUD READY” 2/3

- Bus de messages pour les communications inter-composants
- Stateless : permet de multiplier les routes d'accès à l'application
- Dynamicité : l'application doit s'adapter à son environnement et se reconfigurer lorsque nécessaire
- Permettre le déploiement et l'exploitation par des outils d'automatisation

ADAPTER OU PENSER SES APPLICATIONS “CLOUD READY” 3/3

- Limiter autant que possible les dépendances à du matériel ou du logiciel spécifique qui pourrait ne pas fonctionner dans un cloud
- Tolérance aux pannes (*fault tolerance*) intégrée
- Ne pas stocker les données en local, mais plutôt :
 - Base de données
 - Stockage objet
- Utiliser des outils standards de journalisation

MODULAIRE

- Multiples composants de taille raisonnable
- Philosophie Unix
- Couplage faible et interface documentée

PASSAGE À L'ÉCHELLE

- Vertical vs Horizontal
- Scale up vs Scale out
- Plusieurs petites instances plutôt qu'une grosse instance

STATEFUL VS STATELESS

- Beaucoup de stateful dans les applications legacy
- Nécessite de partager l'information d'état lorsque plusieurs workers
- Le stateless élimine cette contrainte

TOLÉRANCE AUX PANNEES

- L'infrastructure n'est pas hautement disponible
- L'API d'infrastructure est hautement disponible
- L'application doit anticiper et réagir aux pannes

STOCKAGE DES DONNÉES

- Base de données relationnelles
- Base de données NoSQL
- Stockage bloc
- Stockage objet
- Stockage éphémère
- Cache, temporaire

DESIGN TENETS D'OPENSTACK (EXEMPLE) 1/2

1. Scalability and elasticity are our main goals
2. Any feature that limits our main goals must be optional
3. Everything should be asynchronous. If you can't do something asynchronously, see #2
4. All required components must be horizontally scalable

DESIGN TENETS D'OPENSTACK (EXEMPLE) 2/2

5. Always use shared nothing architecture (SN) or sharding. If you can't Share nothing/shard, see #2
6. Distribute everything. Especially logic. Move logic to where state naturally exists.
7. Accept eventual consistency and use it where it is appropriate.
8. Test everything. We require tests with submitted code. (We will help you if you need it)

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

CONCEVOIR UNE INFRASTRUCTURE POUR LE CLOUD

AUTOMATISATION

- Automatiser la gestion de l'infrastructure : indispensable
- Création des ressources
- Configuration des ressources

INFRASTRUCTURE AS CODE

- Travailler comme un développeur
- Décrire son infrastructure sous forme de code
(Heat/Terraform, Ansible)
- Suivre les changements dans un VCS (git)
- Mettre en place de la revue de code
- Utiliser des mécanismes de tests
- Exploiter des systèmes d'intégration et déploiement continu

BESOIN D'ORCHESTRATION

- Manager tous les types de ressources par un point d'entrée
- Description de l'infrastructure dans un fichier (*template*)
- Heat (intégré à OpenStack), Terraform

TESTS ET INTÉGRATION CONTINUE

- Style de code
- Validation de la syntaxe
- Tests unitaires
- Tests d'intégration
- Tests de déploiement complet

TOLÉRANCE AUX PANNEES

- Tirer parti des capacités de l'application
- Ne pas tenter de rendre l'infrastructure compute HA

AUTOSCALING GROUP

- Groupe d'instances similaires
- Nombre variable d'instances
- Scaling automatique en fonction de métriques
- Permet le passage à l'échelle *horizontal*

MONITORING

- Prendre en compte le cycle de vie des instances : DOWN != ALERT
- Monitorer le service plus que le serveur

BACKUP

- Être capable de recréer ses instances (et le reste de son infrastructure)
- Données (applicatives, logs) : block, objet

COMMENT GÉRER SES IMAGES ?

- Utilisation d'images génériques et personnalisation à l'instanciation
- Création d'images plus ou moins personnalisées :
 - Modification à froid : libguestfs, virt-builder, virt-sysprep
 - Modification au travers d'une instance : automatisation possible avec Packer
 - Construction *from scratch* : diskimage-builder (TripleO)
 - Construction *from scratch* avec des outils spécifiques aux distributions (openstack-debian-images pour Debian)

CONCLUSION

POUR CONCLURE

- 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
- Les métiers d'architecture logicielle et infra évoluent