bg left

OpenStack

Glenn Y. Rolland <teaching@glenux.net>

Préambule

bg right:33%

Objectifs de la formation

- Comprendre les concepts clés et les bases techniques d'un Cloud privé
- Appréhender OpenStack et ses différentes composantes
- Concevoir un Cloud privé avec OpenStack
- Maîtriser les méthodes et bonnes pratiques de déploiement d'un Cloud privé
- Savoir administrer un Cloud privé

Qui êtes vous ?

Petit tour de présentation... avec 3 questions

- Quel est votre "bagage" ? (expérience, compétences, etc.)
- Pourquoi participez vous à cette formation aujourd'hui ?
- Comment utiliserez-vous ces nouvelles compétences d'ici 2 ou 5 ans ?

Qui suis-je ?

2021 → aujourd'hui

Auteur, conférencier et co-fondateur de CRYPTO-CHEMISTS
Formation et conseil sur l'impact des Blockchain & des technologies P2P.

2018 → aujourd'hui

Directeur technique et co-fondateur de BOLDCODE
Développement et audit logiciel, web et mobile, offshoring éthique au Népal.

2017 → 2022

Directeur technique et co-fondateur de DATA-TRANSITION
Gestion éthique des données, audit des SI, conformité au RGPD.

2010 → 2017

Gérant et co-fondateur de NETCAT (GNUSIDE)
Infrastructures & systèmes en réseau, optimisation de la fiabilité, de la sécurité et de la performance.

2006 → 2010

Ingénieur de recherche chez BEWAN (Pace Group)
Conception de systemes embarqués, et automatisation de la qualité logicielle.

Contact: teaching@glenux.net

Déroulement de la formation

Horaires

- 9h00 - 12h30
- 13h30 - 17h00
- Des pauses le matin et l'après midi

Le cadre

- Liberté de parole dans le respect des autres et des objectifs de la formation
- Bienveillance, nous sommes dans un espace d’apprentissage
- Confidentialité de l’animateur et des participants sur les échanges

La formation chez IB (groupe Cegos) - en quelques mots

width:200px

- 37 ans d'existance (dont 23 en tant que filialle du groupe Cegos)
- 11 centres de formation en France
- 22.000 personnes formées par an
- 600 consultants-formateurs experts

- 1200+ sessions inter-entreprise
- 2100+ sessions intra-entreprise
- 72,2M€ de chiffres d'affaires
- Certification qualité ISO 9001
- Certification Qualiopi

Note: Chiffres de 2019

La formation chez Orsys - en quelques mots

width:300px

- 45 ans d'existance
- 7700 entreprises et administrations clientes
- 85000 personnes formées par an
- 1700 intervenants experts
- 97.5% de clients satisfaits

- 27 centres de formation en France
- 5000+ sessions intra-entreprise
- 7000+ sessions inter-entreprise
- 72,2M€ de chiffres d'affaires

Note: Chiffres de 2019

Introduction

bg right:33%

Cloud computing

Définition du Cloud Computing

- Mise à disposition de ressources informatiques via Internet
- Infrastructures, plateformes et logiciels en tant que services
- Service mesurable (et facturable)
- Paiement à l'utilisation et élasticité des ressources
- Elasticité

Intérêt et avantages du cloud

- Réduction des coûts d'investissement et d'exploitation
- Accès à des ressources évolutives
- Flexibilité et rapidité de déploiement
- Facilité de collaboration et de partage de données
- Mises à jour et maintenance simplifiées

Inconvénients et risques liés au Cloud computing

- Dépendance à l'accès Internet
- Latence et performances réseau
- Risques liés à la sécurité et à la confidentialité des données
- Conformité et réglementations
- Perte de contrôle sur l'infrastructure

Types de Clouds

SaaS (Software as a Service)

- Applications hébergées et gérées par un fournisseur
- Accès via Internet, généralement via un navigateur Web
- Mises à jour et maintenance prises en charge par le fournisseur

PaaS (Platform as a Service)

- Plateformes de développement et d'exécution d'applications
- Services d'infrastructure, de middleware et de développement
- Permet aux développeurs de se concentrer sur le code

IaaS (Infrastructure as a Service)

- Ressources informatiques virtualisées (serveurs, stockage, réseau)
- Fournit des ressources évolutives et à la demande
- Contrôle et gestion de l'infrastructure par l'utilisateur

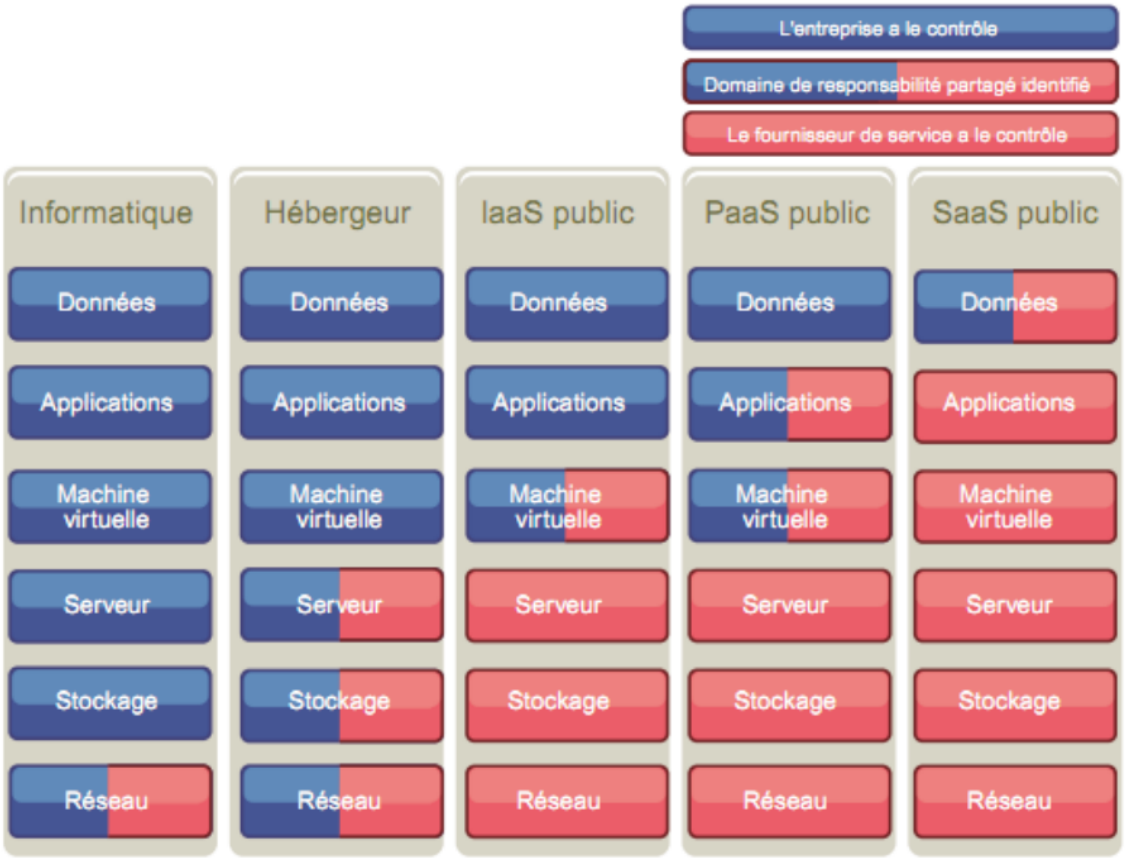
Clouds publics et privés

- Cloud public : ressources partagées et gérées par un fournisseur externe
- Cloud privé : ressources dédiées et gérées en interne ou par un fournisseur
- Contrôle, sécurité et performances différenciés

Clouds hybrides et communautaires

- Cloud hybride : combinaison de clouds publics et privés
- Cloud communautaire : partage de ressources entre organisations similaires
- Flexibilité, partage des coûts et respect des réglementations spécifiques

Responsabilités selon les modèles



Etat du marché

Opérateurs "leaders" actuels du clouds

- Amazon Web Services (AWS) : leader du marché
- Microsoft Azure : offre étendue de services
- Google Cloud Platform (GCP) : puissance de l'IA et du Big Data
- IBM Cloud : solutions hybrides et multicloud
- Oracle Cloud : optimisé pour les bases de données Oracle
- Alibaba Cloud : leader en Chine et Asie

Enjeux et besoin pour un cloud opensource

- Exploitation de ressources matérielle pré-existantes.
- Indépendance vis-à-vis des fournisseurs et réduction de la dépendance
- Personnalisation et adaptation aux besoins spécifiques de l'organisation
- Coûts réduits grâce à l'utilisation de logiciels libres

- Collaboration et partage des connaissances au sein de la communauté
- Amélioration continue grâce aux contributions de la communauté
- Transparence et sécurité renforcées par l'accès au code source
- Interopérabilité et portabilité entre différentes solutions opensource
- Enjeux de souveraineté

Le projet OpenStack

Historique et objectifs du projet

- Créé en 2010 par Rackspace Hosting et la NASA
- Objectif : créer une plateforme de Cloud Computing open source
- Favoriser l'interopérabilité et l'évolutivité
- Offrir une alternative aux solutions propriétaires

Les acteurs clés et contributeurs

- Fondation OpenStack : organisation à but non lucratif qui gère le projet
- Plus de 750 entreprises membres, dont :
  - Red Hat, IBM, HP, Intel, Cisco, Canonical (Ubuntu), Mirantis, SUSE, etc.
- Communauté mondiale de développeurs et d'opérateurs

📚 OpenDev: OpenStack

📚 GitHub: OpenStack (mirror)

Organisation et gouvernance du projet

- Gouvernance ouverte et transparente
- Comités techniques et groupes de travail
  - Assure une répartition équilibrée des responsabilités
  - Encourage la collaboration et l'échange d'idées
- Contributions ouvertes à tous les membres de la communauté
  - Favorise l'innovation et l'amélioration continue du projet OpenStack
  - Processus de prise de décision basé sur le consensus
- Méritocratie : les contributeurs actifs et compétents sont reconnus et récompensés

📚 OpenStack Documentation: OpenStack Governance

Board of Directors:

- Composé de représentants des entreprises membres

- Responsable de la direction stratégique et financière du projet
- Favorise la collaboration entre les membres et l'évolution d'OpenStack

Technical Committee (TC)

- Composé d'experts techniques élus par la communauté
- Garantit la qualité technique et la cohérence entre les projets
- Encourage l'innovation et les meilleures pratiques

Project Teams

- Équipes dédiées à des projets spécifiques au sein d'OpenStack
- Chaque équipe est dirigée par un Project Team Lead (PTL)
- Favorise la répartition des responsabilités et l'expertise spécialisée

Special Interest Groups (SIGs):

- Groupes de travail axés sur des domaines spécifiques ou des problématiques transversales
- Facilitent la collaboration entre les membres ayant des intérêts communs
- Permettent de partager les connaissances et d'améliorer l'efficacité du projet

Les différentes versions d'OpenStack

- Nouvelle version tous les 6 mois
- Chaque version porte un nom de code alphabétique
  - Exemples : Austin (première version), Bexar, Cactus, Diablo, ...
- Dernière version en date :
  - 2023.1 Antelope : publiée
  - 2023.2 Bobcat : en développement
- Chaque version apporte des améliorations, des corrections de bugs et de nouvelles fonctionnalités
- Certaines versions peuvent marquer des étapes importantes ou introduire des changements majeurs

📚 OpenStack Releases

📚 2022-02-10 Release Cadence Adjustment

Comparaison avec les solutions propriétaires

Amazon Web Services (AWS)

- Propriétaire et fermée
  - Exploitation des projets opensource
- Très large gamme de services

Microsoft Azure

- Propriétaire et fermée
- Intégration avec les produits Microsoft

📚 OpenStack vs Azure Stack: 5 Key Differences

Bilan général

- Moins d'options "clés en main" pour les services managés
- Un écosystème de partenaires moins étendu
- Certaines fonctionnalités avancées sont absentes ou moins développées (ex: machine learning, datalakes, outils devops intégrés, IoT et services connectés, CDN mondiaux)
- Intégration avec des outils propriétaires peut être plus complexe
- Support technique généralement moins centralisé et structuré

📚 CompareCloud

Comparaison avec les solutions open source

CloudStack

- Projet créé en 2008 par Cloud.com acheté par Citrix en 2011
- Don en open-source à la fondation Apache, en 2012
- Moins de composants et de fonctionnalités que OpenStack
- Architecture plus simple et plus facile à déployer

📚 https://cloudstack.apache.org/

Eucalyptus

- Projet open source créé en 2008
- Perte de vitesse depuis 2014 (rachat par HP)
- Compatible avec les API Amazon EC2 et S3
- Moins de fonctionnalités et d'évolutivité que OpenStack

📚 https://www.eucalyptus.cloud/

OpenNebula

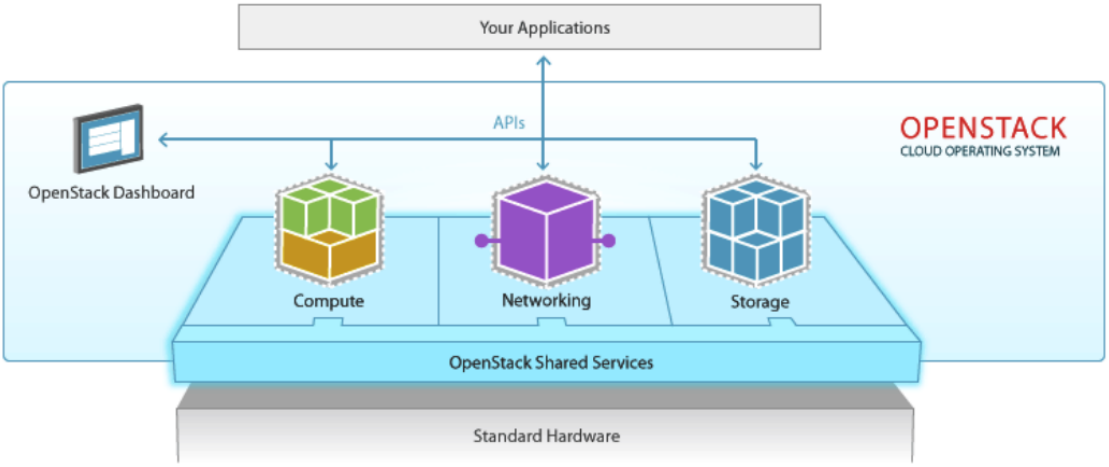
- Projet open source centré sur la gestion des centres de données virtuels
- Moins de fonctionnalités et de composants que OpenStack
- Plus adapté aux petites et moyennes infrastructures
- Utilisé par le StratusLab, Telefonica et Unity Technologies

📚 https://opennebula.io/

Architecture de la plateforme OpenStack

Vue d'ensemble des composants

- OpenStack est modulaire et flexible
- Composants appelés "services" ou "projets"
- Chaque service responsable d'une fonction spécifique

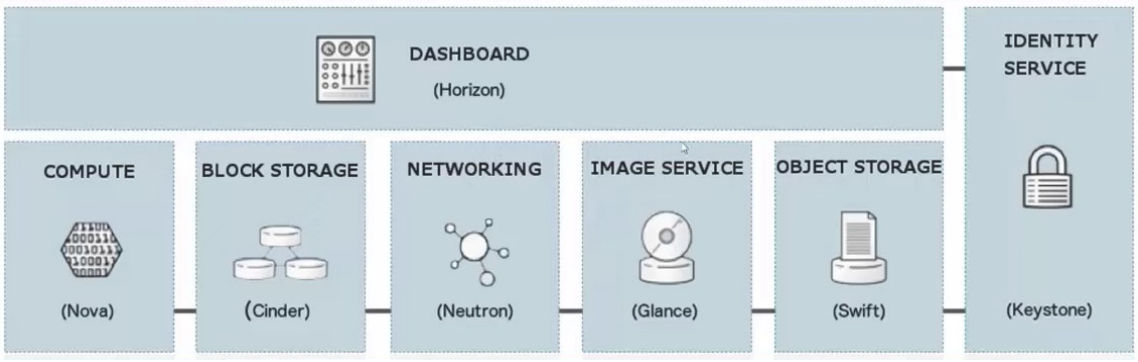


📚 OpenStack Documentation: OpenStack API documentation

Interaction entre les composants

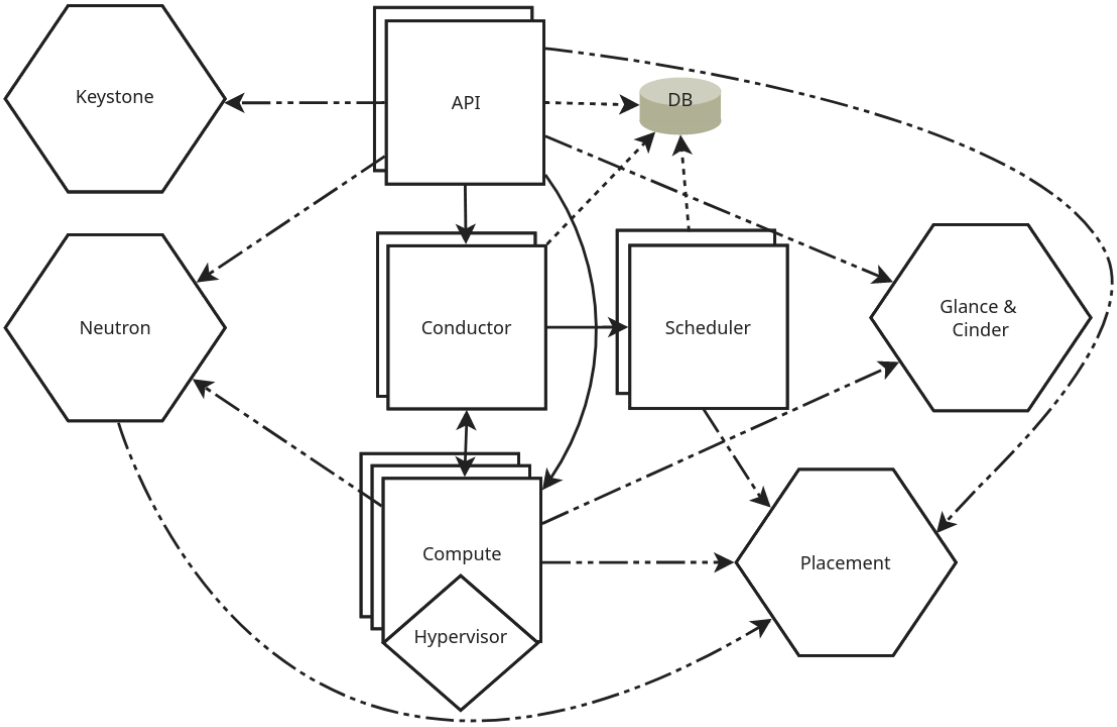
- Services communiquent via API
- Messages transmis par bus de messages (ex : RabbitMQ)
- Standardisation des interactions entre services

Présentation des 6 principales briques



Nova (Compute)

- Gestion des instances de machines virtuelles
- Supporte divers hyperviseurs (KVM, Hyper-V, ESXi)
- Interactions avec d'autres services (Neutron, Cinder, Glance)

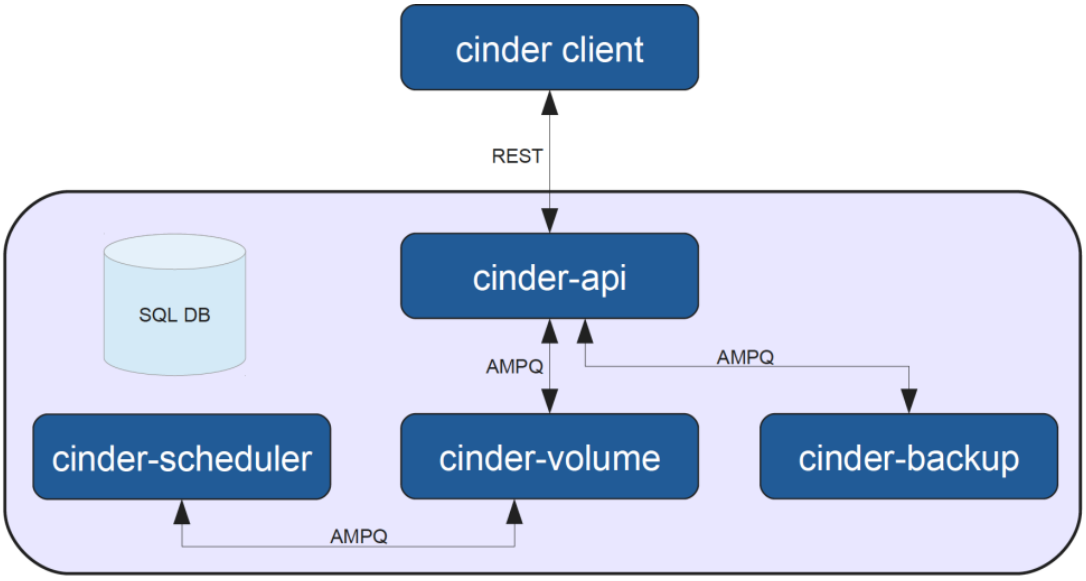


📚 Nova System Architecture

Cinder (Block Storage)

- Gestion des volumes de stockage en mode bloc
- Peut être connecté à des instances Nova

- Divers backends supportés (Ceph, LVM, NetApp, etc.)

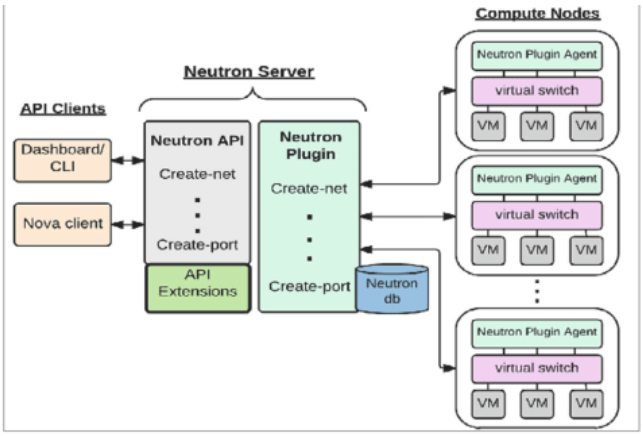


📚 OpenStack Documentation: Cinder, Basic architecture

📚 Laying Cinder Block (Volumes) In OpenStack

Neutron (Networking)

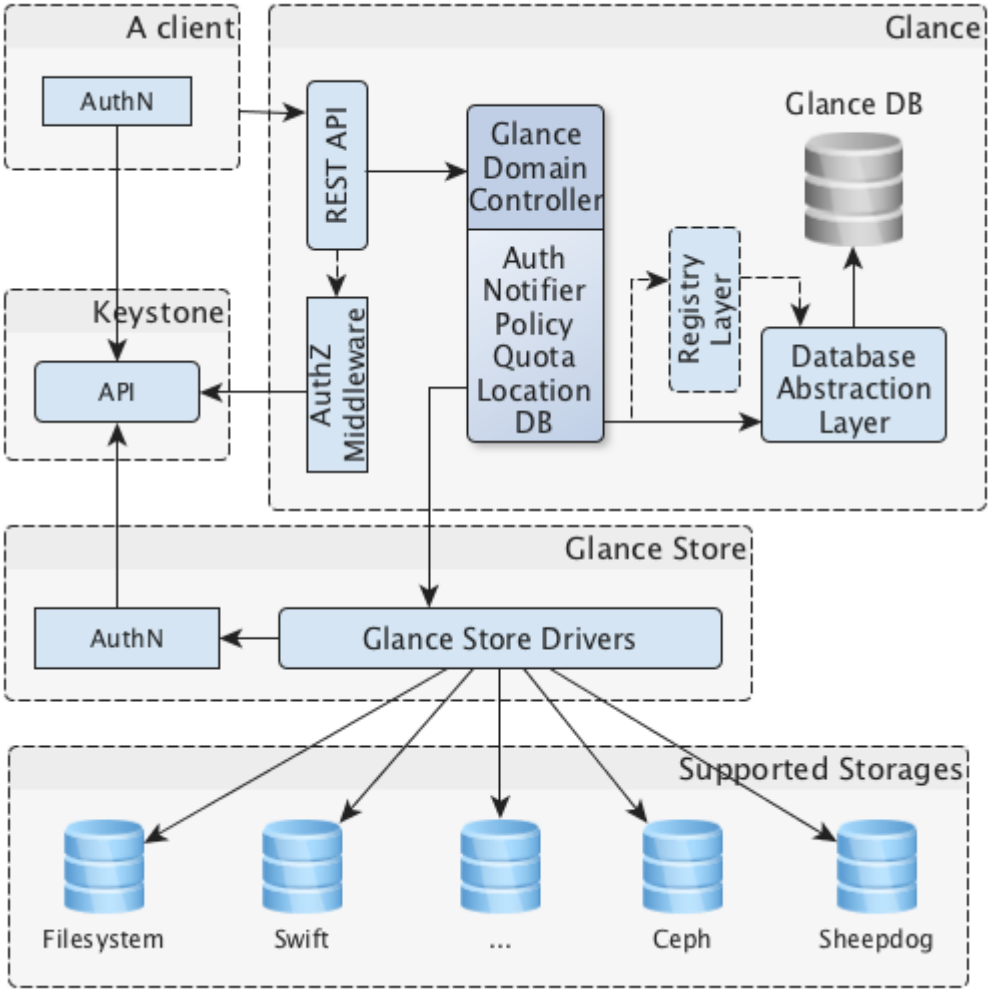
- Gestion des réseaux et des adresses IP
- Supporte divers plugins (Open vSwitch, Linux Bridge)
- Fonctionnalités avancées (LBaaS, VPNaaS, FWaaS)



Glance (Image Service)

- Gestion des images de machines virtuelles
- Stockage et récupération d'images

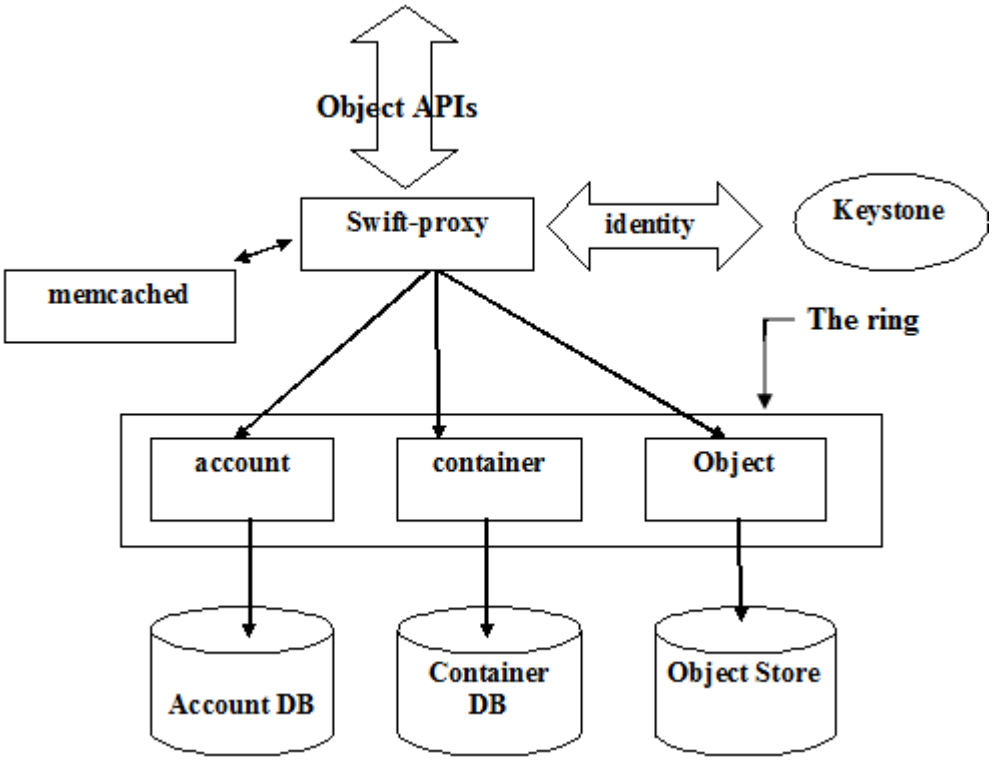
- Supporte divers formats (QCOW2, VMDK, etc.)



📚 OpenStack Documentation: Glance, Basic architecture

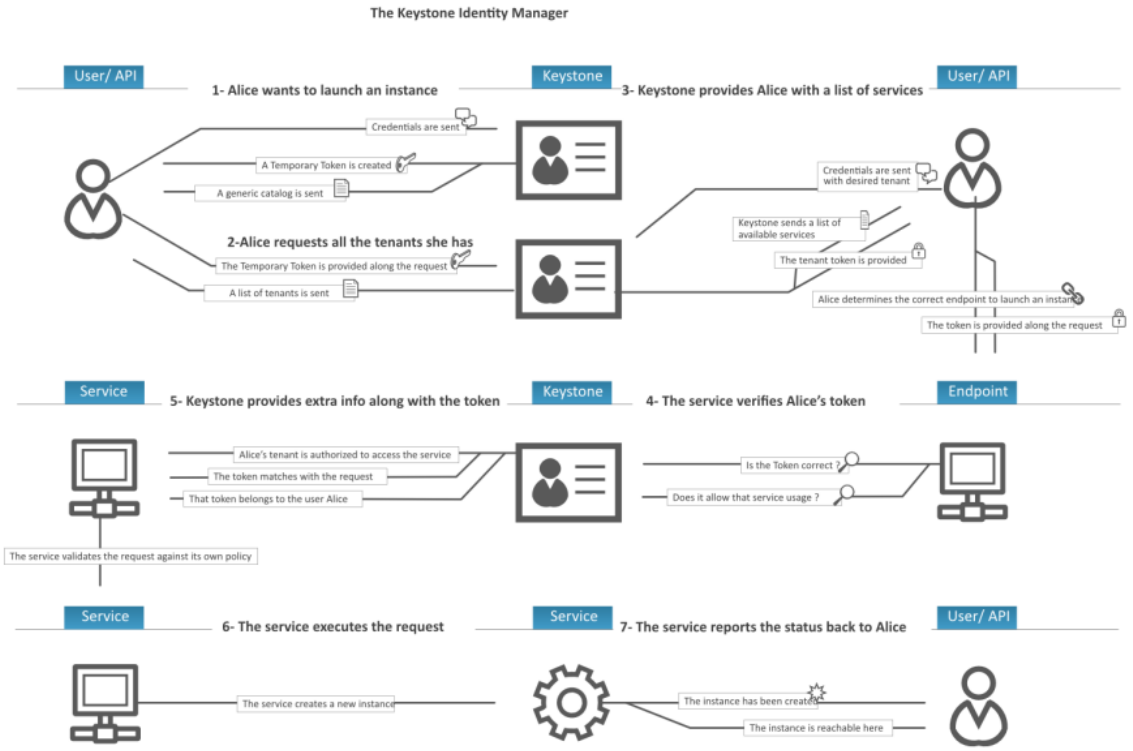
Swift (Object Storage)

- Stockage d'objets distribué et évolutif
- Haute disponibilité et réplication des données
- Supporte API S3 d'Amazon



Keystone (Identity Service)

- Gestion de l'authentification et des autorisations
- Gestion des utilisateurs, rôles et projets
- Supporte divers backends d'authentification (LDAP, OAuth, etc.)

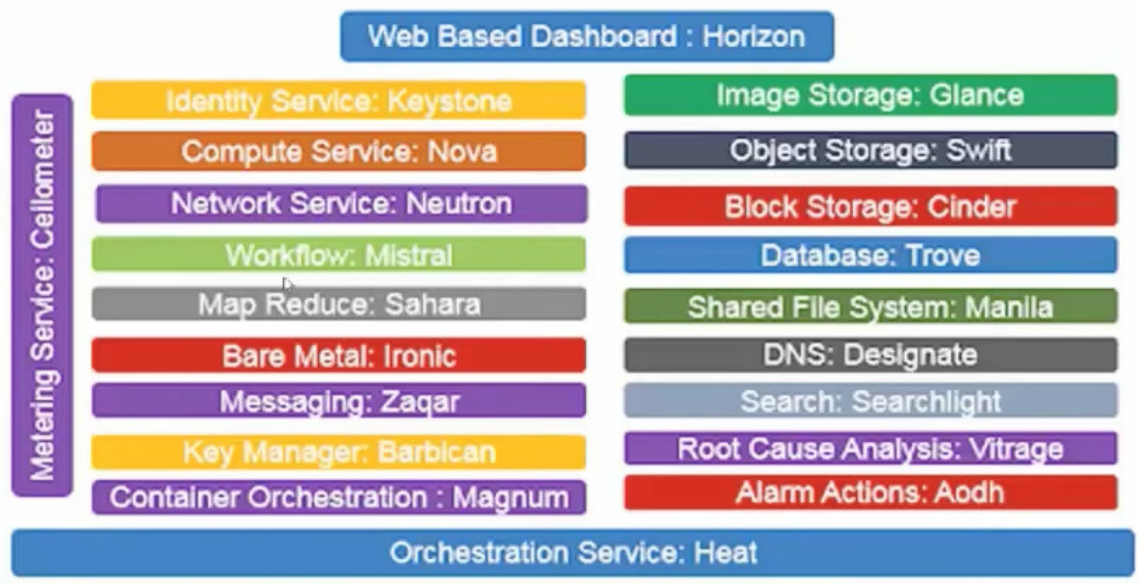


📚 OpenStack Keystone Workflow & Token Scoping

Horizon (Dashboard)

- Interface web de gestion d'OpenStack
- Permet de gérer et surveiller les ressources
- Extensible via plugins

Composants optionnels et extensions



Compute projets

- **Nova**
- **Glance**
- **Ironic** (Bare Metal) : service de gestion des machines physiques, permettant de gérer des machines physiques comme des ressources de calcul dans un environnement OpenStack.
- **Magnum** : provisionning et management de moteurs d'orchestration
- **Zun** (Container service) : service de déploiement et de gestion de conteneurs, permettant de créer et de gérer des conteneurs dans un environnement OpenStack.
- **Storlet** (FaaS) : implémente les Fonctions-as-a-service dans le contexte d'Object Storage.

Securité, Identité et Conformité

- **KeyStone**
- **Barbican** : provisioning dsécurisé, management des secrets (password, certificats X.509, clés de chiffrement, etc.)
- **Congress** - service de gouvernance
- **Mistral** (Workflow service) : service de gestion des flux de travail, permettant de créer et de gérer des flux de travail pour automatiser les tâches dans un environnement OpenStack.

Stockage, Sauvegarde et Recovery

- **Cinder**
- **Swift**
- **Manila** (Shared File System) : service de partage de fichiers, permettant de créer et de gérer des systèmes de fichiers partagés dans un environnement OpenStack.
- **Kardor** : backup et restauration des data et meta-data d'applications
- **Freezer** : backup et restauration de fichiers et de filesystemes

Réseau

- **Neutron**
- **Designate** (DNSaaS) : service de gestion des noms de domaine, permettant de créer et de gérer des enregistrements DNS dans un environnement OpenStack.
- **Dragonflow** : controller de SDN distribué
- **Kuryr** : plugin réseau Docker
- **Octavia** (Load Balancer) : service de répartition de charge, permettant de créer et de gérer des équilibreurs de charge dans un environnement OpenStack.
- **Tacker** (NFV Orchestration) : service de gestion des fonctions réseau virtuelles (NFV), permettant de créer et de gérer des fonctions réseau telles que des routeurs, des pare-feu et des passerelles VPN.
- **Tricile** : extension des abstration réseau à travers plusieurs instances openstack

Data et Analyse

- **Trove** : DBaaS
- **Sahara** (Data processing) : service de traitement de données, permettant de créer et de gérer des clusters de traitement de données dans un environnement OpenStack.

Projects applicatifs

- **Heat** (Orchestration) : Gestion de l'infrastructure en tant que code (IaaC), automatisation du déploiement de ressources. Supporte des modèles de déploiement (HOT, AWS CloudFormation)
- **Murano** (Application Catalog) : catalogue d'applications permettant de déployer et de gérer des applications dans un environnement OpenStack.

Et aussi ...

- **Cyborg** (Accelerator management) : service de gestion des accélérateurs matériels tels que les GPU, permettant de les utiliser pour accélérer les charges de travail dans un environnement OpenStack.
- **Masakari** (High Availability) : service de haute disponibilité, permettant de surveiller les noeuds de l'infrastructure OpenStack et de les récupérer en cas de panne.
- **Monasca** (Monitoring) : service de surveillance des performances de l'infrastructure OpenStack, permettant de collecter et d'analyser des données de performances pour optimiser les ressources et résoudre les problèmes.
- **Rally** (Benchmarking) : outil de benchmarking pour tester les performances d'OpenStack, permettant de simuler des charges de travail et de mesurer les performances de l'infrastructure.
- **Senlin** (Cluster management) : service de gestion de clusters, permettant de créer et de gérer des clusters de machines virtuelles ou physiques dans un environnement OpenStack.
- **Watcher** (Infrastructure optimization) : service d'optimisation de l'infrastructure, permettant d'optimiser les ressources de l'infrastructure OpenStack en fonction des charges de travail et des objectifs métier.

Méthodes d'installation

Installation manuelle

- Procédure étape par étape
- Grande flexibilité et contrôle
- Grande quantité de composants - haute complexité
- Temps d'installation très long
- Risque d'erreurs plus élevé
- Idéal pour apprendre et maîtriser OpenStack

Utilisation de scripts et d'outils d'automatisation

- Gains de temps et de fiabilité
- Réduction des erreurs humaines
- Installation plus rapide
- Nécessite une connaissance préalable des outils

Packstack

- Outil basé sur Puppet
- Installation rapide d'OpenStack
- Idéal pour les environnements de test et de développement

📚 OpenStack Wiki: Packstack

DevStack

- Outil de développement et de test
- Configuration rapide d'un environnement OpenStack
- Facile à mettre à jour
- Ne convient pas pour les déploiements en production

Ansible

- Outil d'automatisation et de gestion de configuration

- Playbooks pour déployer OpenStack
- Flexible et extensible
- Convient pour les déploiements en production

Kolla Ansible

- Déploiement d'OpenStack avec Docker
- Simplifie la gestion et la mise à jour
- Réduction de la complexité de maintenance
- Solution résiliente et évolutive

Distributions OpenStack

Points communs

- Versions pré-configurées d'OpenStack
- Support commercial et maintenance
- Mises à jour et correctifs réguliers
- Coût plus élevé que les installations manuelles

Red Hat OpenStack Platform

- Basée sur la distribution CentOS
- Support de Red Hat
- Intégration avec d'autres produits Red Hat
- Formation et certification disponibles

Mirantis Cloud Platform

- Support Kubernetes et Docker
- Gestion des mises à jour en continu
- Formation et services professionnels

SUSE OpenStack Cloud

- Basée sur la distribution SUSE Linux
- Intégration avec d'autres produits SUSE

- Support et services professionnels

Ubuntu OpenStack

- Basée sur la distribution Ubuntu
- Intégration avec d'autres produits Canonical
- Support et services professionnels

Critères de sélection d'une méthode d'installation

- Complexité de l'infrastructure
- Budget et ressources disponibles
- Niveau d'expertise technique
- Besoins en termes de support et de maintenance
- Objectifs à long terme et évolutivité

Commandes de base

FIXME: Basic Openstack commands and usage

Understanding the Openstack command-line client

- Overview of the `openstack` command-line client and its usage
- Understanding the structure and syntax of openstack commands
- Exploring the various openstack commands and options available

Basic usage of the openstack client to interact with Openstack services

- Using the openstack client to authenticate and connect to an Openstack cloud
- Performing basic operations on Openstack services such as creating, listing, and deleting resources
- Understanding the openstack output format and how to interpret it

Virtual machines and instances

Using the openstack client to create, start, stop, and delete virtual machines and instances

```
openstack server create \
    --image Ubuntu-20.04 \
    --flavor m1.medium \
    --network my_network \
    --security-group my_security_group my_vm
```

```
openstack server start my_vm
```

```
openstack server stop my_vm
```

```
openstack server reboot --hard my_vm
```

```
openstack server delete my_vm
```

Managing virtual machine metadata and user data

```
openstack server set \
   --property user_data="#!/bin/bash\necho 'Hello World'\n" my_vm
```

Attaching and detaching volumes to instances

```
openstack server add volume my_vm my_volume
openstack server remove volume my_vm my_volume
```

Using the openstack client to connect to instances using ssh

Use the openstack server list command to find the IP address of an instance:

```
openstack server list
```

Use the ssh command to connect to the instance

Flavors and images

Configuring flavors

- To list all existing flavors, use the command openstack flavor list
- To create a new flavor, use the command

  ```
  openstack flavor create --ram <ram_size> --vcpus <vcpu_count> --disk <disk_size>
  <flavor_name>
  ```

- To show details for a specific flavor, use the command

  ```
  openstack flavor show <flavor_name>
  ```

- To delete a flavor, use the command

  ```
  openstack flavor delete <flavor_name>
  ```

Managing images

- To list all existing images, use the command

  ```
  openstack image list
  ```

- To upload a new image, use the command

```
openstack image create \
   --container-format <container_format> \
   --disk-format <disk_format> \
   --file <image_file> <image_name>
```

- To show details for a specific image, use the command

```
openstack image show <image_name>
```

- To delete an image, use the command

```
openstack image delete <image_name>
```

Note:

- In order to create an image using "openstack server image create" command, the instance should be in shutdown state
- Make sure you are providing the right container-format and disk-format when uploading an image

Creating a new image from an existing instance

- To create an image from an existing instance, use the command

```
openstack server image create --name <image_name> <instance_name> (--wait)
```

- When you create a new image from an existing instance, you are essentially taking a snapshot of the instance's current state and creating a new image from it.
- When you create an image from an existing instance, a copy of the root file system of the instance will be captured and stored as an image. This image will include all data, configurations and settings from the moment the image is created. This image can be used to create new instances later on, with identical or similar configuration as the original instance.

Understanding the concept of public and private images

Public images

- are accessible to all users and projects within an OpenStack cloud.

- They are typically used for common operating systems and application images that are intended to be shared and reused across an organization.

Private images

- Only accessible to the user or project that created or owns the image.
- They are typically used for custom images or images that contain sensitive data and are intended to be used by specific users or projects only.
- By default, images are set to private

To make an image public, use the command

```
openstack image set --public <image_name>
```

To make an image private, use the command

```
openstack image set --private <image_name>
```

Lab session:

- Authentication to OpenStack using openstack command
- Creating virtual machines, instances and managing them
- Managing flavor and images and related operations
- Adding metadata and user data
- Attaching and Detaching Volumes to instances
- Connecting instances using ssh

Setting up an Openstack development environment

Installing and configuring an Openstack development environment using DevStack

- Prerequisites for installing DevStack (system requirements, software dependencies)
- Step-by-step instructions for installing DevStack on a Linux-based system
- Configuring DevStack for different deployment scenarios (single-node, multi-node)
- Verifying the DevStack installation by accessing the Openstack dashboard (Horizon)

Understanding the basic DevStack configuration files and options

- Understanding the DevStack local.conf file, which is used to configure the DevStack installation
- Configuring options such as the release version of Openstack, the host IP address, and the enabled services
- Understanding the DevStack stackrc file, which contains environment variables used by DevStack

Troubleshooting common DevStack installation issues

- Identifying and resolving issues related to system requirements and software dependencies
- Resolving issues related to network configuration, hostnames, and firewall rules
- Troubleshooting issues related to the Openstack services and their configuration files
- Tips and best practices for debugging and resolving DevStack installation issues

Lab Session:

- Installing DevStack
- Configuring DevStack local.conf file
- Verifying the DevStack installation by accessing the Openstack dashboard
- Troubleshooting common DevStack installation issues if they occur

Lab session

Setting up the development environment

- Installing virtualbox or VMware on the local machine
  - Download the appropriate version of virtualbox or VMware for your system from the official website
  - Install virtualbox or VMware following the instructions provided by the installer.

- Installing Ubuntu 18.04 as the host operating system
  - Once virtualbox or VMware is installed, create a new virtual machine and select Ubuntu 18.04 as the operating system
  - Follow the prompts to configure the virtual machine as desired (e.g. assigning RAM, disk space)
  - Start the virtual machine and complete the installation of Ubuntu 18.04
  - (If you are using VMware, you should follow their guide for creating and configuring a new Virtual machine)

- Installing DevStack
  - Once Ubuntu is installed, open a terminal and run the following commands:
    - sudo apt-get update
    - sudo apt-get upgrade
    - sudo apt-get install git
    - git clone https://github.com/openstack-dev/devstack.git
    - cd devstack
  - Copy the local.conf file to the devstack folder run ./stack.sh to start the installation, it could take a while

- Configuring DevStack
  - Once the installation of DevStack is completed, you can log in to the OpenStack dashboard by navigating to http:///dashboard in your browser
  - run the command source openrc to setup the environment

    - run the command openstack token issue to get the token, you can use this token to interact with openstack using API

Getting familiar with the OpenStack CLI

- Familiarizing with the basic openstack commands such as:
    - openstack --help : to understand the different option of openstack command
    - openstack user list : To list all the user in the openstack deployment
    - openstack flavor list : To list all the flavor available in the deployment
- Creating and managing virtual machines, networks, and storage using the openstack client
    - openstack server create : Creating a virtual machine
    - openstack server list : Listing all the virtual machines
    - openstack network list : Listing all the networks
    - openstack subnet list : Listing all the subnets
    - openstack router list : Listing all the routers
- Understanding the Openstack API and how to interact with it
    - sending basic rest requests to interact with openstack
    - How to retrieve token

Basic operations on users, roles and Projects

- Create a user
    - openstack user create --password
- List all users
    - openstack user list
- Create a role
    - openstack role create
- Add a role to a user
    - openstack role add --user --project
- Create a project
    - openstack project create --domain

- Add a user to a project
    - openstack role add --user --project
- Assign a role to a user in a project
    - openstack role add --user --project

Hands-on exercise and case study:

- Create a virtual machine
- Create and attach a volume to the virtual machine
- Create a network, subnet, and router
- Allocate a floating IP
- Create security groups and rules
- Create a new user, assign role and add to the project
- demonstrate the change of access level with the role.

Travaux pratiques 5.1. Comparaison des différentes méthodes d'installation 5.2. Sélection d'une méthode d'installation adaptée au contexte 5.3. Installation et configuration d'une plateforme OpenStack de base 5.4. Validation de l'installation et tests de fonctionnement

Machines virtuelles

bg right:33%

Introduction à Nova

Qu'est-ce que Nova ?

- Nova est le service de calcul d'OpenStack.
- Nova gère les instances de machines virtuelles (VM) et les ressources de calcul.
- Nova est conçu pour être évolutif et distribué.
- Nova interagit avec d'autres composants d'OpenStack (par exemple, Keystone, Glance, Neutron).

Fonctionnalités et composants de Nova

- Gestion des instances de VM (création, modification, suppression).
- Planification des instances sur les hôtes.
- Gestion des ressources de calcul (CPU, RAM, stockage).
- API RESTful pour l'interaction avec d'autres services et outils.
- Support de plusieurs hyperviseurs (KVM, Xen, VMware, Hyper-V, etc.).
- Composants principaux :
  - nova-api : reçoit et traite les requêtes des utilisateurs.
  - nova-scheduler : sélectionne l'hôte approprié pour les instances.
  - nova-conductor : interagit avec la base de données et les autres services.
  - nova-compute : gère les instances et les ressources de calcul sur les hôtes.
  - nova-db : stocke les informations sur les instances, les hôtes, les réseaux et les volumes. Permet de conserver l'état des ressources et des configurations.

Gestion des ressources et interactions avec les autres services OpenStack

- Nova est le gestionnaire principal des ressources de calcul dans OpenStack.
- Nova orchestre les cycles de vie des instances, en contrôlant leur création, leur mise à l'échelle, leur suspension, leur redémarrage, leur migration et leur suppression.

- Nova interagit avec les autres services d'OpenStack pour fournir des fonctionnalités intégrées :
  - Keystone pour l'authentification des utilisateurs et la définition des autorisations.
  - Glance pour la récupération des images de VM à partir desquelles les instances sont créées.
  - Neutron pour la configuration des réseaux virtuels et l'attribution des adresses IP aux instances.
  - Cinder pour la fourniture de volumes de stockage en mode bloc aux instances.
  - Swift pour le stockage d'objets, y compris les instantanés d'instance et les images de disque.
  - etc.

Fonctionnalités avancées et extensibilité de Nova

- La gestion des quotas pour les projets et les utilisateurs.
- La planification des instances en fonction des politiques et des ressources disponibles.
- L'équilibrage de charge entre les hôtes de calcul.
- La haute disponibilité et la récupération après sinistre grâce à la migration en direct et aux instantanés d'instance.
- Nova permet l'intégration avec des outils d'automatisation et d'orchestration, tels que Heat, Terraform et Ansible, via son API RESTful.
- Nova offre également une extensibilité grâce à des plugins et des pilotes pour divers hyperviseurs, matériels et services tiers.

Gestion des images

Utilisation des images avec Nova

- Importer des images dans le catalogue d'images Glance
- Choisir une image pour créer une instance
- Lister les images disponibles
  - CLI: `openstack image list`
  - Horizon: Compute > Images
- Utiliser des formats d'image pris en charge : qcow2, raw, vmdk, vhd, etc.

Gestion des gabarits

- Understanding the concept of flavors and how they are used in Nova
- Creating and managing flavors
- Using the Nova command-line client to manage flavors

Gestion du réseau virtuel

Comprendre les réseaux virtuels dans OpenStack

- Réseaux virtuels : abstraction des réseaux physiques pour les instances
- Réseau Flat : pas de VLAN, toutes les instances partagent le même réseau
- Réseau VLAN : chaque projet a son propre VLAN, isolation des réseaux
- Réseau VXLAN/GRE : encapsulation des paquets, évolutivité accrue
- Utilisation de Neutron en combinaison avec Nova pour la gestion du réseau

Configuration du réseau virtuel avec Nova

- Configuration de Nova pour utiliser Neutron (`neutron.conf`)
- Définition du type de réseau (Flat, VLAN, VXLAN/GRE) dans nova.conf
- Configuration du pilote de réseau virtuel (Open vSwitch, Linux Bridge) dans `nova.conf`
- Configuration de l'adresse du serveur Neutron (`neutron.url`)
- Création et configuration de réseaux virtuels dans Horizon ou via la CLI `openstack network create`
- Association des réseaux virtuels aux instances lors de leur création

Gestion des adresses IP et des sous-réseaux

- Utilisation des pools d'adresses IP (Floating IP) pour les instances
- Création et configuration des sous-réseaux dans Horizon ou via la CLI `openstack subnet create`
- Attribution des adresses IP aux instances lors de leur création
- Modification des adresses IP et des sous-réseaux associés aux instances en cours d'exécution
- Gestion des adresses IP publiques et privées pour les instances
- Utilisation de la CLI `nova floating-ip-associate` pour associer des adresses IP flottantes aux instances

Gestion des instances

Création d'instances

- Sélectionner une image, un type de machine virtuelle (flavor) et d'autres paramètres (réseau, clé SSH, etc.)
- Créer une instance avec openstack server create ou dans Horizon sous "Compute" > "Instances" > "Launch Instance"
- Spécifier des métadonnées pour personnaliser l'instance
- Attacher des volumes à l'instance pour étendre le stockage
- Configurer la sécurité et les groupes de sécurité pour contrôler l'accès à l'instance

Manipulation des instances

- Arrêter une instance
  - CLI: `openstack server stop <instance_id>`
  - Horizon: Compute > Instances > Instance Actions" > Stop
- Redémarrer une instance avec
  - CLI: `openstack server reboot <instance_id>`
  - Horizon: Compute > Instances > Instance Actions > Reboot
- Suspendre et reprendre une instance
  - CLI: `openstack server suspend <instance_id>` et `openstack server resume <instance_id>`
  - Horizon: Compute > Instances > Instance Actions > Suspend / Resume
- Redimensionner une instance (changer le flavor) avec
  - CLI: `openstack server resize <instance_id> <new_flavor>`
  - Horizon: Compute > Instances > Instance Actions > Resize
- Supprimer une instance avec
  - CLI: `openstack server delete <instance_id>`
  - Horizon: Compute > Instances > Instance Actions > Delete
- Créer un instantané de l'instance avec
  - `openstack server image create <instance_id>`

- Horizon: Compute > Instances > Instance Actions > Create Snapshot

Mise en œuvre et configuration de Nova

Installation de Nova

- Installer les paquets requis :
  - openstack-nova-api
  - openstack-nova-conductor
  - openstack-nova-consoleauth
  - openstack-nova-novncproxy
  - openstack-nova-scheduler
  - openstack-nova-placement-api
- Activer et démarrer les services :

```
# systemctl enable --now nova-api.service
# systemctl enable --now nova-scheduler.service
# systemctl enable --now nova-conductor.service
# systemctl enable --now nova-consoleauth.service
# systemctl enable --now nova-novncproxy.service
# systemctl enable --now nova-placement-api.service
```

Configuration initiale de Nova

- Modifier le fichier de configuration /etc/nova/nova.conf :
  - Spécifier les informations de connexion à la base de données (section [database])
  - Définir les informations d'authentification avec Keystone (section [keystone_authtoken])
  - Configurer l'API placement (section [placement])
  - Configurer l'accès aux images avec Glance (section [glance])
- Appliquer la configuration et redémarrer les services :

```
# systemctl restart nova-api.service
# systemctl restart nova-scheduler.service
# systemctl restart nova-conductor.service
# systemctl restart nova-consoleauth.service
```

```
# systemctl restart nova-novncproxy.service
# systemctl restart nova-placement-api.service
```

- Synchroniser la base de données :

```
# nova-manage db sync
```

Personnalisation et optimisation de la configuration

- Configurer l'hyperviseur (section [libvirt]) :
  - Choisir l'hyperviseur (KVM, QEMU, ESXi, etc.)
  - Configurer les paramètres spécifiques à l'hyperviseur
- Optimiser les performances (section [DEFAULT]) :
  - Configurer la taille des pools de travailleurs
  - Personnaliser les paramètres de mise en cache
- Configurer la gestion des réseaux (section [neutron]) :
  - Spécifier les informations d'authentification avec Neutron
  - Configurer les paramètres du réseau
- Gérer les quotas (section [quota]) :
  - Configurer les quotas par défaut pour les instances, les cœurs, la RAM, etc.
- Trouver les informations dans Horizon :
  - Accéder à l'onglet "Compute" puis "Instances" pour gérer les instances
  - Accéder à l'onglet "Admin" puis "Hypervisors" pour gérer les hyperviseurs
- Utiliser la CLI nova ou la CLI openstack :
  - Utiliser `nova list` ou `openstack server list` pour lister les instances
  - Utiliser `nova hypervisor-list` ou `openstack hypervisor list` pour lister les hyperviseurs

Gestion d'hyperviseurs multiples

Introduction aux hyperviseurs supportés

Configuration d'ESXi avec Nova

- Installer et configurer VMware vSphere et ESXi
- Configurer Nova pour utiliser ESXi comme hyperviseur
  - Modifier le fichier /etc/nova/nova.conf
  - Changer l'option "compute_driver" à "vmwareapi"
  - Renseigner les informations d'authentification et de connexion pour vSphere
- Redémarrer les services Nova après la configuration
- Vérifier la configuration via la CLI nova ou la CLI openstack

Configuration de KVM avec Nova

- Installer et configurer KVM et libvirt sur les nœuds de calcul
- Configurer Nova pour utiliser KVM comme hyperviseur
  - Modifier le fichier /etc/nova/nova.conf
  - Changer l'option "compute_driver" à "libvirt"
  - Renseigner le type d'hyperviseur à "kvm"
- Redémarrer les services Nova après la configuration
- Vérifier la configuration via la CLI nova ou la CLI openstack

Gestion des ressources entre différents hyperviseurs

- Comprendre l'allocation des ressources dans OpenStack
- Utiliser les filtres d'hôte pour contrôler la répartition des instances
  - Modifier le fichier /etc/nova/nova.conf
  - Activer et configurer les filtres d'hôte appropriés Utiliser les agrégats d'hôtes pour regrouper les hôtes par hyperviseur
  - Créer des agrégats via Horizon, la CLI nova ou la CLI openstack

    - Ajouter des hôtes aux agrégats et définir des métadonnées
- Surveiller l'utilisation des ressources par hyperviseur via Horizon, la CLI nova ou la CLI openstack

Introduction to Openstack Compute

Introduction to Openstack Compute (Nova)

Explanation of Nova and its role in the Openstack ecosystem

FIXME

Nova Architecture and Components

Nova consists of the following components:

- Nova API: The Nova API receives and responds to incoming API requests for VM management, such as create, delete, and start/stop operations.
- Nova Scheduler: The Nova scheduler is responsible for placing new instances on hosts according to the available resources and configured policies.
- Nova Conductor: The Nova conductor is a service that acts as a intermediary between the Nova API and the other Nova services.
- Nova Compute: The Nova compute service is responsible for creating, starting, and terminating instances on the hypervisors.

How Nova is responsible for managing and provisionning virtual machines

FIXME

Use Cases and Scenarios for Nova

Nova can be used in many scenarios to provide infrastructure as a service (IaaS) and platform as a service (PaaS)

- In private clouds to provide a self-service infrastructure to internal users
- In public clouds to provide a multi-tenant infrastructure to external users
- As a stand-alone virtualization solution, supplementing existing virtualization environments

Infrastructure as a Service (IaaS)

- Nova can be used to provide IaaS (Infrastructure as a Service)

- IaaS allows customers to rent virtualized computing resources (e.g. CPU, memory, storage, network) on-demand, over the internet
- This provides customers with the ability to run their own operating systems and applications on the rented resources, without the need to invest in and maintain physical infrastructure

Platform as a Service (PaaS)

- Nova can be used to provide PaaS (Platform as a Service)
- PaaS allows customers to rent a platform to run their own applications on, without the need to manage the underlying infrastructure
- This provides customers with the ability to focus on their application development, while the provider handles the underlying infrastructure

Private and Public Clouds

- Nova can be used to build both private and public clouds
- A private cloud is a cloud infrastructure operated solely for a single organization, typically by that organization or by a third party, and only members of that organization are allowed to use it
- A public cloud is a cloud infrastructure operated for the benefit of the general public, by a commercial organization, such as Amazon Web Services or Microsoft Azure

Augmenting Existing Virtualization Solutions

- Nova can be used to augment existing virtualization solutions
- For example, organizations that already have VMware or Hyper-V can use Nova to provision and manage virtual machines on top of their existing virtualization infrastructure

Setting up Nova in OpenStack

Installing Nova

To install Nova, you need to have a working OpenStack environment. The installation process for Nova is typically included as part of the OpenStack installation process

Configuring Nova

To configure Nova, you need to modify the /etc/nova/nova.conf file. The following are common options that you will need to configure:

- compute_driver: The driver that Nova will use to interact with the hypervisors
- auth_strategy: The authentication strategy to use (e.g. keystone)
- my_ip: The IP address of the Nova API server
- network_api_class: The networking API class to use (e.g. nova.network.neutronv2.api.API)
- glance_api_servers: The Glance API servers to use
- transport_url: The messaging transport URL

Integrating Nova with other OpenStack services

Nova requires integration with other OpenStack services to function properly: - Keystone for authentication - Neutron for networking - Cinder and Glance for storage

You can check the detail of the configurations and the instruction of integration in the OpenStack documentation

Using the Nova CLI and Dashboard

Nova Command-line client (CLI)

The Nova command-line client (CLI) is a powerful tool for managing your OpenStack cloud. The nova command-line client is installed as part of the python-novaclient package.

Common Nova commands: - nova list: List all instances on the cloud - nova show : Show details of a specific instance - nova delete : Delete a specific instance - nova boot --flavor --image : Create and boot a new instance

Nova Dashboard

The Nova Dashboard, also known as Horizon, provides a web-based user interface for OpenStack cloud administrators and users.

You can use the Nova Dashboard to perform the same actions that you can with the Nova CLI, such as: - Creating and managing instances - Creating and managing flavors

Use Cases and Scenarios for Nova

- Explanation of typical use cases for Nova such as infrastructure as a service (IaaS) and platform as a service (PaaS)
- Discussion of common scenarios where Nova can be used, such as in private and public clouds, as well as how it can be used to augment existing virtualization solutions

Setting up Nova in OpenStack

- Overview of the process for installing and configuring Nova
- Explanation of how Nova integrates with other Openstack components like Keystone for authentication, Neutron for networking, and Cinder and Glance for storage

Using the Nova CLI and Dashboard

- Overview of the Nova command-line client (CLI) and its usage
- Demonstration of how to use the Nova CLI to manage virtual machines
- Overview of the Nova dashboard and its usage

Lab: Setting up a test OpenStack environment with Nova

- Students will work on a lab to set up a test OpenStack environment with Nova
- Students will practice using the Nova CLI and dashboard to manage virtual machines

Quiz

- A quiz will be held to evaluate student's understanding of the module content and their ability to apply the knowledge to real-world scenarios

Live Migration and Evacuation

- Understanding the concept of live migration and evacuation
- Configuring live migration and evacuation
- Performing live migrations and evacuations

Scheduling and Placement

- Understanding the concept of scheduling and placement in Nova
- Configuring and managing scheduling and placement policies
- Troubleshooting scheduling and placement issues

Stockage bloc

bg right:33%

Présentation de Cinder

Introduction à Cinder

- Cinder : service de stockage en mode bloc d'OpenStack
- Fournit des volumes persistants aux instances de machines virtuelles
- Fonctionne avec divers backends de stockage
- Peut être utilisé avec des hyperviseurs compatibles, tels que KVM, ESXi et Hyper-V

Architecture et composants

- Cinder-API : expose les API pour les opérations de stockage en mode bloc
- Cinder-Scheduler : sélectionne le backend de stockage approprié en fonction des critères de filtrage et de pondération
- Cinder-Volume : gère les opérations de volume, telles que la création, la suppression et l'attachement
- Les drivers de volume : assurent la communication avec les backends de stockage spécifiques
- Les bases de données : stockent les informations sur les volumes et les backends de stockage

Utilisation et cas d'usage

- Création de volumes persistants pour les instances de machines virtuelles
- Sauvegarde et restauration de volumes
- Snapshots de volumes pour créer des instantanés de données
- Migration de volumes entre différents backends de stockage
- Gestion des quotas de stockage pour les projets et les utilisateurs

Manipulation de Cinder

Pour les opérations avec Cinder, on peut utiliser :

- Horizon : l'interface web d'OpenStack, qui permet de gérer les volumes dans la section "Volumes" sous "Computing"

- CLI nova (obsolete) : l'interface en ligne de commande d'OpenStack, qui permet de gérer les volumes avec des commandes comme :
  - `nova volume-create` ,
  - `nova volume-delete`
  - `nova volume-attach`
- CLI openstack :
  - `openstack volume create`
  - `openstack volume delete`
  - `openstack volume set`

Mise en œuvre du stockage en mode bloc avec Cinder

Installation de Cinder

- Installer les paquets nécessaires : cinder-api, cinder-scheduler, cinder-volume
- Configurer la base de données pour Cinder
- Synchroniser la base de données avec la commande : cinder-manage db sync

Configuration des services Cinder

- Modifier le fichier de configuration /etc/cinder/cinder.conf
  - Configurer les informations d'authentification et les points d'accès pour les autres services d'OpenStack
  - Configurer les informations du backend de stockage (ex: LVM, Ceph, NFS)
  - Configurer les options spécifiques aux pilotes de stockage (si nécessaire)
- Redémarrer les services de Cinder après la configuration
  - systemctl restart cinder-api
  - systemctl restart cinder-scheduler
  - systemctl restart cinder-volume

Backend supportés par Cinder

Les backends de stockage

- Types de backends de stockage pris en charge :
  - Stockage en mode bloc (SAN)
  - Stockage en mode fichier (NAS)
  - Stockage objet
- Exemples de backends pris en charge :
  - Ceph RBD, LVM, NFS, GlusterFS,
  - Dell EMC, HPE 3PAR, IBM, NetApp, Pure Storage, VMware VMDK

Configuration des backends

- Configuration des backends dans le fichier cinder.conf :
  - Définir plusieurs sections de backends
  - Spécifier les paramètres spécifiques à chaque backend

📚 OpenStack Documentation: Cinder Configuration

- Exemple de configuration pour le backend Ceph RBD :
  - `rbd_user`, `rbd_pool`, `rbd_secret_uuid`, `rbd_ceph_conf`
- Exemple de configuration pour le backend LVM :
  - `volume_driver`, `volume_group`, `lvm_type`
- Exemple de configuration pour le backend NFS :
  - `nfs_shares_config`, `nfs_mount_options`
- Utilisation de la CLI:
  - `openstack volume service list`

Intégration avec les systèmes de stockage existants

- Adapter la configuration de Cinder pour utiliser un système de stockage existant :
  - Utiliser les paramètres spécifiques au backend
  - Adapter les paramètres de connexion (authentification, adresses IP, etc.)
- Migration de volumes entre backends :
  - Utiliser la commande `openstack volume migrate` pour déplacer un volume d'un backend à un autre
  - Migrer les volumes sans interruption de service
- Vérifier l'intégration dans Horizon :
  - Créer un volume avec le nouveau backend sélectionné
  - Attacher le volume à une instance
  - Vérifier le bon fonctionnement du volume avec le nouveau backend

Stockage objet

bg right:33%

Présentation de Swift

Introduction à Swift

- Swift est un système de stockage d'objets distribué et hautement disponible
- Fait partie intégrante du projet OpenStack
- Conçu pour stocker et récupérer de grandes quantités de données non structurées
- Scaling horizontal pour supporter des petaoctets de données
- Réplication automatique des données pour assurer la durabilité et la disponibilité

Utilisation et cas d'usage

- Stockage d'objets volumineux comme des images, vidéos, sauvegardes, archives, etc.
- Idéal pour les applications nécessitant une grande capacité de stockage et une disponibilité élevée
- Utilisé en conjonction avec d'autres services OpenStack comme Glance pour stocker les images de machines virtuelles
    - Gestion des conteneurs et objets dans le panneau Project > Object Store > Containers
- Intégration avec la CLI OpenStack pour la gestion des conteneurs et des objets
    - Création et suppression des conteneurs :
        - `openstack container create <container_name>`
        - `openstack container delete <container_name>`
    - Listing des conteneurs et des objets avec
        - `openstack container list`
        - `openstack object list <container_name>`
    - Téléchargement et téléversement des objets avec
        - `openstack object save <container_name> <object_name>`
        - `openstack object create <container_name> <object_name>`

- Visualisation et gestion des conteneurs et objets via l'interface Web Horizon

Architecture et composants

- Composants principaux : serveurs Proxy, serveurs de stockage et anneau (Ring)
- Serveurs Proxy : gèrent les requêtes des clients et interagissent avec les serveurs de stockage
- Serveurs de stockage : responsables du stockage et de la récupération des objets
- Anneau (Ring) : structure de données logique qui répartit les objets sur les serveurs de stockage
- Réplication et rééquilibrage automatiques pour assurer la cohérence des données

Fonctionnement du Ring Builder dans Swift

- Création des rings
  - Trois types de rings : account, container, object
  - Chaque ring gère une partie spécifique du stockage
- Partitionnement
  - Division de l'espace de stockage en partitions égales
  - Les partitions sont des unités de base pour la distribution des données
- Répartition des partitions
  - Attribution des partitions aux nœuds de stockage selon capacité et poids
  - Équilibrage des réplicas sur différents nœuds et zones de stockage
- Génération du fichier de configuration du ring
  - Fichier de configuration pour chaque type de ring (ring.gz)
  - Distribution des fichiers aux nœuds du cluster Swift
  - Utilisation pour déterminer l'emplacement des objets et réplicas
- Mise à jour des rings
  - Utilisation du ring builder pour mettre à jour les rings en cas de changements

    - Redistribution et rééquilibrage des modifications sur le cluster

- Commandes swift-ring-builder
    - Utilisées pour créer, gérer et mettre à jour les rings
    - Planification et configuration importantes pour optimiser performance et disponibilité

Mise en œuvre et configuration de Swift

Installation de Swift

- Préparer l'environnement système
  - Installer les paquets requis
  - Configurer les dépôts logiciels appropriés
- Installer les services Swift
  - Proxy Server
  - Account Server
  - Container Server
  - Object Server
- Configuration des services Swift
  - Configurer le fichier /etc/swift/proxy-server.conf
  - Configurer le fichier /etc/swift/account-server.conf
  - Configurer le fichier /etc/swift/container-server.conf
  - Configurer le fichier /etc/swift/object-server.conf

Configuration des services Swift

- Configurer le Proxy Server
  - Définir les options d'authentification
  - Spécifier le pipeline WSGI
  - Configurer les paramètres du réseau
- Configurer les services Account, Container et Object
  - Définir les paramètres du réseau
  - Configurer les chemins de stockage
  - Spécifier les politiques de réplication et les paramètres de temps de réconciliation
- Configurer les services complémentaires
  - Activer le service de dispersion (pour vérifier la répartition des données)
  - Configurer les quotas de conteneurs et d'objets

Paramétrage des politiques de stockage

- Créer des politiques de stockage
  - Éditer le fichier /etc/swift/swift.conf
  - Définir les politiques de stockage avec les noms et les indices uniques
- Appliquer les politiques de stockage
  - Ajouter les politiques aux fichiers de configuration des services (proxy-server.conf, account-server.conf, container-server.conf, object-server.conf)
  - Redémarrer les services Swift pour appliquer les modifications
    - `systemctl restart swift-proxy`
    - `swift-init restart all`
- Gérer les politiques de stockage
  - Dans la configuration, au niveau des serveurs swift
  - Pas dans Horizon
- Utiliser les politiques de stockage via la CLI
  - Utiliser les commandes openstack pour créer, lister et supprimer des conteneurs avec différentes politiques de stockage
    - `openstack container create --storage-policy <policy_name> <container_name>`
    - `openstack container list --long`
  - Utiliser les commandes openstack pour afficher les détails et les statistiques des politiques de stockage
    - `swift stat -v`

Gestion des pools de stockage avec Swift

Création et gestion des pools

- Création de pools de stockage avec la CLI Swift
  - Utiliser swift-ring-builder pour créer un anneau (ring)
  - Ajouter des devices avec swift-ring-builder
- Configuration des zones de stockage
  - Définir les zones pour une répartition optimale des données
- Modification et suppression des pools
  - Utiliser swift-ring-builder pour modifier ou supprimer un device

Répartition des données

- Fonctionnement de la répartition des données dans Swift
  - Utilisation d'algorithmes de répartition (consistent hashing)
  - Réplication des données pour garantir la durabilité
- Contrôle de la répartition des données
  - Configurer le nombre de réplicas
  - Choisir le niveau de durabilité souhaité
- Optimisation de la répartition des données
  - Ajuster les poids des devices dans l'anneau (ring)

Supervision et maintenance des pools

- Surveillance des pools de stockage
  - Utiliser les outils de monitoring (par exemple: Swift Healthcheck)
  - Vérifier l'état des services Swift avec systemctl ou service
- Maintenance des pools de stockage
  - Remplacement d'un device défaillant
  - Mise à jour de la configuration de l'anneau (ring) avec swift-ring-builder

📚 OpenStack Documentation: Swift Ring builder

Understanding Cinder and Swift Architecture

- Introduce the OpenStack Storage service, explain what Cinder and Swift do
- Explain the architecture of Cinder and Swift, and their components
- Introduce the concepts of block and object storage, and how they are implemented in Cinder and Swift
- Explain the Cinder and Swift APIs, and how they interact with other OpenStack components

Managing Block Storage with Cinder

- Explain how to create and manage Cinder volumes
- Explain how to attach and detach Cinder volumes to instances
- Explain how to create and manage Cinder snapshots
- Explain how to configure Cinder storage backends and understand their different features.

Managing Object Storage with Swift

- Explain how to create and manage Swift containers
- Explain how to upload and download objects to/from Swift
- Explain how to configure and understand Swift's replication, ring, and policies
- Explain how to use the Swift CLI and API

Differences between Cinder and Swift

- Explain the main differences between Cinder and Swift and when to use each one
- Discuss the benefits and drawbacks of using Cinder or Swift
- Give examples of use cases for Cinder and Swift

Eval

Lab

- Create and manage a Cinder volume and attach it to an instance
- Create and manage a Cinder snapshot
- Create and manage a Swift container and upload/download objects to/from it.

Assignments

- Research and find real-world use cases for Cinder and Swift
- Compare and contrast different storage backends that can be used with Cinder

Exam

- Questions will be designed to evaluate student's understanding of Cinder and Swift architecture and components, ability to manage and create block and object storage, and differences between the two services and their use cases.

Gestion du stockage avec Cinder
6.1. Création et gestion des volumes
6.2. Attachement et détachement des volumes aux instances
6.3. Sauvegarde et restauration des volumes

Gestion du stockage avec Swift
7.1. Création et gestion des conteneurs
7.2. Téléchargement et téléversement des objets
7.3. Gestion des métadonnées et des ACLs

Gestion du réseau

bg right:33%

Introduction à Neutron

Historique et évolution

- Projet Quantum lancé en 2011 pour gérer le réseau dans OpenStack
- Renommé en Neutron en 2013 pour éviter les conflits de marque
- Evolution continue pour répondre aux besoins changeants du cloud computing
- Intégration avec d'autres technologies de virtualisation réseau (SDN, NFV)

Fonctionnalités principales

- Gestion des réseaux virtuels
- Gestion des sous-réseaux et des adresses IP
- Gestion des routeurs virtuels et de la connectivité L3
- Support pour les réseaux VLAN, VXLAN et GRE
- Intégration avec Open vSwitch pour les switchs virtuels
- Extension de fonctionnalités via des plugins et des drivers
- API RESTful pour l'automatisation et l'intégration avec d'autres outils

Composants de Neutron

- neutron-server : service central pour gérer les requêtes API
- neutron-plugin : plugin pour le backend spécifique (ex: Open vSwitch, Linux Bridge)
- neutron-dhcp-agent : agent pour gérer les services DHCP
- neutron-l3-agent : agent pour gérer les routeurs virtuels et la connectivité L3
- neutron-metadata-agent : agent pour fournir les métadonnées aux instances

Utilisation

Informations dans Horizon :

- Onglet "Réseaux" pour gérer les réseaux, sous-réseaux, et routeurs
- Onglet "Instances" pour connecter les instances aux réseaux

Commandes CLI :

- Utiliser `openstack network` et `openstack subnet` pour gérer les réseaux et sous-réseaux
- Utiliser `openstack router` pour gérer les routeurs virtuels
- Utiliser `nova boot` avec l'option `--nic` pour connecter une instance à un réseau spécifique

Switchs virtuels avec Open vSwitch

Présentation d'Open vSwitch

- Open vSwitch (OVS) : switch virtuel multilayer open source
- Fonctionne sur Linux et d'autres systèmes d'exploitation
- Conçu pour supporter les réseaux Cloud
- Compatible avec les normes de gestion de réseaux SDN (Software-Defined Networking)
- Gestion avancée du trafic réseau entre instances virtuelles et le réseau physique

Installation et configuration

- Installer Open vSwitch sur les noeuds réseau :
  - apt-get install openvswitch-switch (Debian/Ubuntu)
  - yum install openvswitch (CentOS/RHEL)
- Créer un bridge OVS :
  - ovs-vsctl add-br br-int
- Ajouter des interfaces au bridge :
  - ovs-vsctl add-port br-int eth1
- Configurer les VLANs si nécessaire :
  - ovs-vsctl set port eth1 tag=VLAN_ID

Intégration avec Neutron

- Configurer Neutron pour utiliser OVS comme mécanisme de gestion des réseaux virtuels
  - Modifier le fichier /etc/neutron/plugins/ml2/ml2_conf.ini
  - Changer le type de mécanisme à mechanism_drivers = openvswitch
- Redémarrer les services Neutron pour prendre en compte les modifications
  - systemctl restart neutron-server
  - systemctl restart neutron-openvswitch-agent

- Vérifier l'intégration d'OVS avec Neutron
  - Utiliser Horizon : aller dans le tableau de bord administrateur, puis Réseaux > Agents
  - Utiliser la CLI openstack : openstack network agent list
  - Rechercher l'agent Open vSwitch dans les résultats

Topologies de réseau Cloud

Réseau plat (Flat Network)

- Pas de segmentation, toutes les machines virtuelles sur le même réseau physique
- Simplicité de mise en œuvre et de gestion
- Limite la scalabilité et augmente les risques de sécurité
- Configuration dans Neutron : choix du mécanisme de type "flat"
- Visualisation dans Horizon : onglet "Réseaux", section "Réseaux"

Réseau VLAN (Virtual LAN)

- Segmentation du réseau physique en plusieurs réseaux logiques
- Améliore la sécurité et l'isolation entre les machines virtuelles
- Utilise des tags 802.1Q pour identifier les réseaux logiques
- Configuration dans Neutron : choix du mécanisme de type "vlan"
- Visualisation dans Horizon : onglet "Réseaux", section "Réseaux"
- CLI : `openstack network create --provider-network-type vlan`

Réseau VXLAN (Virtual Extensible LAN)

- Encapsulation de trames Ethernet dans des paquets IP
- Permet de créer des réseaux overlay indépendants de l'infrastructure physique
- Améliore la scalabilité par rapport au VLAN, jusqu'à 16 millions d'identifiants de réseau
- Configuration dans Neutron : choix du mécanisme de type "vxlan"
- Visualisation dans Horizon : onglet "Réseaux", section "Réseaux"
- CLI : `openstack network create --provider-network-type vxlan`

Réseau GRE (Generic Routing Encapsulation)

- Encapsulation de paquets de divers protocoles dans des paquets IP

- Permet de créer des réseaux overlay indépendants de l'infrastructure physique
- Moins performant que VXLAN en raison de l'absence d'optimisation matérielle
- Configuration dans Neutron : choix du mécanisme de type "gre"
- Visualisation dans Horizon : onglet "Réseaux", section "Réseaux"
- CLI : `openstack network create --provider-network-type gre`

Daemon de routage (L3)

Principe du routage L3

- Routage L3 : interconnexion de réseaux IP à différents niveaux hiérarchiques
- Fonctions principales :
  - Transmission de paquets entre sous-réseaux
  - Gestion de la table de routage
  - Implémentation de politiques de routage
- Utilisation de routeurs virtuels pour le routage dans OpenStack
- Gestion des adresses IP publiques et privées
- Support de NAT (Network Address Translation) pour les instances

Mise en œuvre avec Neutron

- Neutron fournit le service L3 pour le routage et la gestion des routeurs virtuels
- Création de routeurs virtuels dans OpenStack via Neutron
- Association des routeurs virtuels avec des réseaux externes et internes
- Configuration des interfaces de routeur pour les sous-réseaux
- Utilisation des commandes CLI neutron et openstack pour gérer les routeurs
- Accès aux routeurs et gestion du routage via l'interface Horizon

Configuration du routage

- Création d'un routeur virtuel :
  - Horizon : Réseau > Routeurs > Créer un routeur
  - CLI : `openstack router create ROUTER_NAME`
- Association d'un routeur à un réseau externe :
  - Horizon : Réseau > Routeurs > Définir le réseau externe
  - CLI : `neutron router-gateway-set ROUTER_ID EXTERNAL_NETWORK_ID`
- Ajout d'une interface de routeur pour un sous-réseau :
  - Horizon : Réseau > Routeurs > Ajouter une interface

    - CLI : `openstack router add subnet ROUTER_ID SUBNET_ID`
- Configuration des règles de routage et NAT :
    - Horizon : Réseau > Routeurs > Gérer les règles
    - CLI : Utiliser `neutron security-group-rule-create` pour créer des règles

Vérification du routage

- Vérification de la configuration du routage :
    - Horizon : Network > Topology
    - CLI : `openstack router show ROUTER_ID` et `openstack router list`

Mise en œuvre et configuration de Neutron

Installation des paquets Neutron

- Installer les paquets nécessaires:
  - apt-get install neutron-server neutron-plugin-ml2 Installer les paquets pour les agents L3 et DHCP:
  - apt-get install neutron-l3-agent neutron-dhcp-agent
- Installer les paquets pour Open vSwitch:
  - apt-get install neutron-plugin-openvswitch-agent
- Installer les paquets pour le contrôleur de réseau:
  - apt-get install openvswitch-controller

Configuration des fichiers de Neutron

- Configurer le fichier `/etc/neutron/neutron.conf` :
  - Paramétrer la connexion à la base de données
  - Configurer l'authentification avec Keystone
- Configurer le fichier `/etc/neutron/plugins/ml2/ml2_conf.ini` :
  - Paramétrer les mécanismes de pilotes (ex: openvswitch, linuxbridge)
  - Configurer les types de réseau (ex: flat, vlan, vxlan)
  - Configurer les gammes d'adresses pour les réseaux VXLAN
- Configurer le fichier `/etc/neutron/l3_agent.ini` :
  - Paramétrer l'interface externe pour le routage L3
- Configurer le fichier `/etc/neutron/dhcp_agent.ini` :
  - Activer l'agent DHCP
  - Configurer l'interface pour l'agent DHCP
- Configurer le fichier `/etc/neutron/plugin.ini` :
  - Spécifier le fichier de configuration ML2

Démarrage des services Neutron

* Redémarrer les services après la configuration: * `systemctl restart neutron-server` * `systemctl restart neutron-plugin-openvswitch-agent` * `systemctl restart neutron-l3-agent` * `systemctl restart neutron-dhcp-agent` * Vérifier l'état des services Neutron: * `systemctl status neutron-server` * `systemctl status neutron-plugin-openvswitch-agent` * `systemctl status neutron-l3-agent` * `systemctl status neutron-dhcp-agent`

Trouver les informations sur le réseau

- Trouver les informations dans Horizon:
  - Aller dans le tableau de bord "Réseau"
  - Vérifier les configurations de réseau et les agents
- Utiliser la CLI nova:
  - nova network-list
  - nova floating-ip-list
- Utiliser la CLI openstack:
  - openstack network list
  - openstack subnet list
  - openstack router list

Introduction

- Overview of Neutron and its role in Openstack
- Understanding the Neutron architecture and components

Managing Virtual Networks and Subnets

- Creating and managing virtual networks in Neutron
- Creating and managing subnets in Neutron
- Using DHCP and DNS in Neutron networks

Router and Floating IPs

- Understanding the concepts of routers and floating IPs in Neutron
- Creating and managing routers in Neutron
- Allocating and releasing floating IPs in Neutron
- Using routers to connect virtual networks
- Configuring and managing SNAT and DNAT

Evaluation

Labs

- Hands-on lab sessions will be provided during the week where students will practice creating virtual networks, subnets, security groups, firewall rules, routers, and floating IPs using the Neutron command-line interface and the Openstack dashboard
- Students will also get an opportunity to practice troubleshoot common Neutron networking issues

Quiz and Review

- A quiz will be given at the end of the week to evaluate student's understanding of the course content
- Students will have the opportunity to ask questions and review the material covered during the week

Travaux pratiques

Créer et configurer un réseau virtuel
6.1. Création d'un réseau virtuel avec Neutron
6.2. Configuration des sous-réseaux et passerelles
6.3. Création de routeurs virtuels
6.4. Connexion des instances aux réseaux virtuels
6.5. Vérification et test de la connectivité réseau

Authentification et autorisations

bg right:33%

Présentation de la brique Keystone

Introduction à Keystone

- Keystone : service d'authentification et d'autorisation d'OpenStack
- Gère les identités et les accès aux ressources d'OpenStack
- Fournit un point central pour la gestion des utilisateurs, projets et services
- Utilise des jetons pour authentifier les requêtes

Fonctionnalités principales de Keystone

- Authentification : vérifie les identifiants des utilisateurs
  - Utilisation de la CLI OpenStack ou de l'interface Horizon pour s'authentifier
- Autorisation : détermine les permissions d'accès aux ressources
  - Basé sur les rôles attribués aux utilisateurs
  - Possibilité de définir des politiques de sécurité personnalisées
- Gestion des services : enregistre et découvre les services d'OpenStack
  - Catalogue des services disponibles dans l'infrastructure
  - Accessible via Horizon ou la CLI OpenStack
- Gestion des jetons : émet et valide les jetons d'authentification
  - Jetons temporaires utilisés pour authentifier les requêtes API

Composants de Keystone

- API : interface pour interagir avec le service Keystone
  - Supporte les requêtes HTTP et les réponses au format JSON
  - Utilisation de la CLI OpenStack ou de l'interface Horizon pour accéder aux fonctionnalités
- Endpoint : point d'accès aux services d'OpenStack
  - URL spécifique pour chaque service et région
  - Permet d'interagir avec les autres services d'OpenStack

- Backend : stockage des données de Keystone
    - Utilisation de bases de données relationnelles comme MySQL ou PostgreSQL
    - Peut également utiliser des solutions de stockage LDAP pour la gestion des utilisateurs et des groupes
- Middleware : composant logiciel intermédiaire pour valider les jetons
    - Intégré aux autres services d'OpenStack pour sécuriser l'accès aux API
    - Utilise les jetons pour vérifier les permissions et authentifier les requêtes

Création des utilisateurs, projets et rôles

Utilisateurs dans Keystone

- Objectif des utilisateurs : représenter des individus ou des systèmes dans OpenStack
- Authentification : identifiants (login/mot de passe) ou autres méthodes (tokens)
- Association aux domaines et projets pour déterminer les droits d'accès

Utilisation

- Horizon : Identity > Users
- Commandes CLI :
  - `openstack user create`
  - `openstack user list`
  - `openstack user delete`

Projets et domaines

- Projets : unités organisationnelles pour regrouper et isoler les ressources
- Domaines : conteneurs pour gérer plusieurs projets et utilisateurs
- Multi-tenancy : isolation des ressources et des politiques entre différents projets
- Horizon : Identity > Projects et Domains
- Commandes CLI :
  - `openstack project create`
  - `openstack domain create`

Rôles et assignations de rôles

- Rôles : ensemble de droits et permissions pour les utilisateurs
- Assignation de rôles : lien entre utilisateurs, projets/domaines et rôles

- Rôles courants : admin, member, reader
- Commandes CLI :
  - `openstack role create`
  - `openstack role assignment create`
- Interface Horizon : Identity > Roles et Role Assignments

Processus de création d'utilisateurs, projets et rôles

- Étape 1 : créer un domaine (optionnel, si multi-domaines requis)
- Étape 2 : créer un projet
- Étape 3 : créer un utilisateur et l'associer à un projet
- Étape 4 : créer un rôle
- Étape 5 : assigner un rôle à un utilisateur pour un projet/domaine spécifique
- Utilisation des commandes CLI et de l'interface Horizon pour chaque étape

Configuration des utilisateurs, projets et rôles

Gestion des utilisateurs

- Création d'utilisateurs avec la CLI openstack: `openstack user create`
- Modification d'utilisateurs: `openstack user set`
- Suppression d'utilisateurs: `openstack user delete`
- Listage des utilisateurs: `openstack user list`
- Assignation d'un rôle à un utilisateur: `openstack role add`
- Horizon: Identity > Users

Gestion des projets et domaines

- Création de projets: `openstack project create`
- Modification de projets: `openstack project set`
- Suppression de projets: `openstack project delete`
- Listage des projets: `openstack project list`
- Création de domaines: `openstack domain create`
- Modification de domaines: `openstack domain set`
- Suppression de domaines: `openstack domain delete`
- Horizon: Identity > Projects et Domains

Gestion des rôles et assignations

- Création de rôles: `openstack role create`
- Modification de rôles: `openstack role set`
- Suppression de rôles: `openstack role delete`
- Listage des rôles: `openstack role list`
- Assignation de rôles aux utilisateurs: `openstack role add --user --project`
- Retrait de rôles aux utilisateurs: `openstack role remove --user --project`
- Horizon: Identity > Roles

Utilisation des politiques de sécurité

- Définition des politiques de sécurité dans le fichier `policy.json`
- Contrôle d'accès basé sur les rôles (RBAC)
- Configuration de politiques pour les services OpenStack
- Horizon: Identity > Policies

```
{
   "identity:create_user": "role:admin"
}
```

📚 OpenStack Documentation: The policy.json file

Mise en œuvre et configuration

Installation de Keystone

- Installer les paquets nécessaires
  - apt-get install keystone
- Vérifier les dépendances (ex: Python)
- Consulter la documentation officielle pour les autres systèmes d'exploitation

Configuration des fichiers de Keystone

- Modifier le fichier de configuration principal
  - `/etc/keystone/keystone.conf`
- Configurer la connexion à la base de données
  - `[database]`
  - `connection = mysql+pymysql://keystone:KEYSTONE_DBPASS@controller/keystone`
- Configurer le token Fernet
  - `[token]`
  - `provider = fernet`

📚 OpenStack Documentation: KeyStone - Install and configure (OBS) 📚 OpenStack Documentation: KeyStone - Install and configure (Ubuntu)

Initialisation de la base de données Keystone

- Créer la base de données MySQL
  - `CREATE DATABASE keystone;`
- Accorder les privilèges nécessaires
  - `GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'localhost' IDENTIFIED BY 'KEYSTONE_DBPASS';`
  - `GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'%' IDENTIFIED BY 'KEYSTONE_DBPASS';`
- Initialiser la base de données
  - `keystone-manage db_sync`

Démarrage du service Keystone

- Démarrer et activer le service Keystone
  - `systemctl enable keystone`
  - `systemctl start keystone`
- Vérifier le statut du service
  - `systemctl status keystone`
- Configurer les variables d'environnement
  - `export OS_USERNAME=admin`
  - `export OS_PASSWORD=ADMIN_PASS`
  - `export OS_PROJECT_NAME=admin`
  - `export OS_USER_DOMAIN_NAME=Default`
  - `export OS_PROJECT_DOMAIN_NAME=Default`
  - `export OS_AUTH_URL=http://controller:5000/v3`
  - `export OS_IDENTITY_API_VERSION=3`
- Utiliser la CLI OpenStack pour vérifier la configuration
  - `openstack token issue`

Overview of Keystone architecture and components:

- Keystone is the Identity Service of Openstack, responsible for providing authentication and authorization for all the OpenStack services.
- Keystone has a simple and extensible architecture, with a modular design that allows new backends to be added easily.
- Keystone's main components are the identity service, the assignment service, and the catalog service.

Managing Users and Projects:

- Creating and managing users and tenants (or projects) in Keystone.
- Understanding roles and role assignments in Keystone
- Creating and managing custom roles

Configuring Authentication and Authorization:

- Configuring authentication methods (e.g. password, token, SAML, OAuth)
- Configuring user-facing authentication and authorization policies (e.g. password complexity, lockout policies)
- Understanding how authentication tokens work in Keystone

Evaluation

Hands-on exercises

- Creating and managing users, tenants and roles
- Configuring authentication methods and policies
- Using the Keystone command-line client and API

Quiz

- A quiz will be given to evaluate the student's understanding of the Keystone's architecture, components, and functionalities.

Travaux pratiques

5.1. Installation et configuration de Keystone 5.2. Création et gestion d'utilisateurs, projets et rôles 5.3. Assignation de rôles aux utilisateurs 5.4. Exploration de la gestion des services avec Keystone

Introduction au Dashboard Horizon

Rôle et objectifs du Dashboard Horizon

Composants et architecture

Navigation et utilisation du Dashboard Horizon

Authentification et connexion

Navigation et organisation des panneaux

Création et gestion des projets

Gestion des instances et des images

Gestion des réseaux et de la connectivité

Gestion du stockage (Swift et Cinder)

Gestion des utilisateurs, rôles et autorisations (Keystone)

Intégration avec d'autres services OpenStack

Gestion des ressources avancées (Heat, Ceilometer, etc.)

Utilisation d'extensions et de plugins

Understanding Horizon architecture and components

- Horizon is a web-based user interface for Openstack that allows users to interact with and manage the resources of their OpenStack cloud. It is built using Django and is tightly integrated with other OpenStack services.
- The main components of Horizon include the dashboard, panels, and views. The dashboard is the top-level page that organizes all the panels, and each panel represents a specific functionality of OpenStack. Views are the individual pages within a panel that display specific information or allow users to perform specific actions.

Navigating and using the Openstack dashboard

- Students will learn how to log in to the Horizon dashboard and navigate through the different panels and views. They will also learn how to perform common tasks such as creating and managing instances, managing networks and storage, and managing users and projects.

Customizing and extending the dashboard

- Students will learn how to customize the layout and appearance of the Horizon dashboard, such as adding custom panels, changing colors and fonts, and adding custom logo. Also they will learn how to extend the functionality of the dashboard by writing custom Django views and hooks, and integrating with other services.

Evaluation

Hands-on Labs

- Students will work on labs to have practical experience on all the covered topics of this week.

Quiz and Homework

- A Quiz will be held to evaluate students' understanding of the week's content.
- Homework will be assigned to students to practice and reinforce the concepts learned during the week.

Dashboard

bg right:33%

Gestion des orchestration et Infrastructure As Code (Heat)

bg right:33%

Infrastructure as Code, Heat and Terraform

Définition de l'Infrastructure as Code (IaC)

- Infrastructure as Code (IaC) : approche pour gérer et provisionner les ressources informatiques
- Utilisation de fichiers de configuration textuels pour décrire l'état souhaité de l'infrastructure
- Mise en place de l'infrastructure automatisée avec des outils et des scripts

Avantages de l'IaC

- Rapidité de déploiement
  - Provisionnement automatisé des ressources
  - Réduction du temps consacré à la configuration manuelle
  - Accélération du processus de déploiement des infrastructures
- Répétabilité
  - Uniformisation des environnements de développement, de test et de production
  - Réduction des erreurs humaines lors du provisionnement
  - Facilité de mise à l'échelle
- Traçabilité
  - Suivi des modifications de l'infrastructure grâce au versionnement du code
  - Meilleure collaboration entre les équipes
  - Audit facilité des changements d'infrastructure
- Simplification de la gestion des environnements
  - Centralisation des configurations et des paramètres
  - Définition claire des responsabilités
  - Réutilisation des configurations pour différents projets

Les principes de base de l'IaC

- Codification de l'infrastructure
  - Description de l'infrastructure avec un langage déclaratif
  - Utilisation de templates pour décrire les ressources et leurs relations
- Versionnement du code
  - Utilisation de systèmes de gestion de version (ex. Git)
  - Suivi des modifications et collaboration entre les équipes
- Tests et validation
  - Vérification de la syntaxe et des erreurs dans les fichiers de configuration
  - Tests automatisés pour valider le comportement de l'infrastructure
  - Utilisation de l'intégration et du déploiement continu (CI/CD)
- Automatisation du déploiement et de la gestion
  - Utilisation d'outils de déploiement (ex. Heat, Terraform)
  - Gestion des modifications d'infrastructure avec des pipelines automatisés
  - Contrôle des accès et des autorisations via des outils comme Keystone et Horizon

Présentation de Heat et Terraform pour OpenStack

Introduction à Heat

- Origines et objectifs
  - Projet OpenStack dédié à l'orchestration
  - Automatiser et gérer le déploiement d'infrastructures complexes
- Composants principaux
  - heat-api: interface RESTful pour interagir avec Heat
  - heat-engine: cœur de l'orchestrateur, traite les templates et exécute les actions
  - heat-dashboard: plugin Horizon pour gérer Heat depuis l'interface web
- Fonctionnement de Heat
  - Utilisation de templates YAML (Heat Orchestration Template, HOT)
  - Description des ressources, paramètres et dépendances
  - Déploiement, mise à jour et suppression des stacks

Introduction à Terraform

- Origines et objectifs
  - Outil IaC multi-cloud développé par HashiCorp
  - Provisionner, modifier et détruire des infrastructures de manière déclarative
- Composants principaux
  - Terraform Core: interprète les fichiers de configuration et gère les ressources
  - Providers: extensions pour interagir avec différentes plateformes (OpenStack, AWS, etc.)

    - Modules: réutilisation de configurations pour faciliter la gestion de l'infrastructure

- Fonctionnement de Terraform
    - Utilisation de fichiers de configuration HCL (HashiCorp Configuration Language)
    - Description des ressources, variables et dépendances
    - Planification et exécution des changements d'infrastructure

Comparaison entre Heat et Terraform

- Points communs et différences
    - Les deux permettent l'automatisation et la gestion d'infrastructures
    - Heat est spécifique à OpenStack, Terraform est multi-cloud

- Avantages et inconvénients de chaque solution
    - Heat: intégration native avec OpenStack, moins polyvalent que Terraform
    - Terraform: prise en charge de plusieurs plateformes, courbe d'apprentissage plus longue

- Cas d'utilisation typiques
    - Heat: déploiements OpenStack purs, gestion des ressources OpenStack spécifiques
    - Terraform: déploiements multi-cloud, infrastructure hybride

Intégration de Heat et Terraform avec OpenStack

- Interaction avec les briques OpenStack
    - Heat: intégration native avec Nova, Swift, Glance, Neutron, etc.
    - Terraform: utilisation du provider OpenStack pour interagir avec les briques

- Configuration et authentification
    - Heat: configuration via openrc.sh, authentification avec Keystone

    - Terraform: configuration du provider OpenStack, authentification avec les variables d'environnement

- Exemples d'utilisation pour OpenStack
    - Heat: déploiement de VMs, création de réseaux, ajout de stockage, etc.
    - Terraform: provisionnement de VMs, allocation d'adresses IP, configuration de réseaux, etc.

Présentation de la brique Heat

Objectifs et fonctionnalités de Heat

- Automatiser le déploiement et la gestion des ressources OpenStack.
- Gérer l'évolutivité des ressources en fonction des besoins.
- Faciliter l'infrastructure As Code avec des templates.

Composants clés de Heat

- Heat API : service RESTful permettant d'interagir avec Heat.
- Heat Engine : interprète et exécute les templates HOT.
- Heat CLI : interface en ligne de commande pour interagir avec Heat.
- Dashboard Heat (intégré à Horizon) : interface graphique pour gérer les templates et les stacks.

Intégration avec les autres briques d'OpenStack

- Interaction avec Keystone pour l'authentification et la gestion des autorisations.
- Gestion des instances via Nova.
- Configuration et gestion des réseaux avec Neutron.
- Gestion du stockage des images avec Glance
- Idem pour les autres ressources (swift, etc.)
- Heat s'intègre dans Horizon pour faciliter la gestion des stacks et des templates.

Le langage de modélisation HOT (Heat Orchestration Template)

Introduction au langage HOT

- Heat Orchestration Template (HOT) : format YAML pour décrire les ressources et les configurations
- HOT facilite la gestion et l'automatisation des déploiements d'infrastructure
- Compatible avec les autres services OpenStack
- Utilisation de la CLI Heat pour déployer et gérer les templates

Structure d'un template HOT

- En-tête avec la version du template et description (optionnelle)
- Section "resources" pour décrire les ressources
- Section "outputs" pour définir les sorties (optionnelle)
- Section "parameters" pour définir les paramètres d'entrée (optionnelle)

En-tête

- La première ligne spécifie la version du langage à utiliser dans le template.
- La description est optionnelle et fournit des informations sur le but du template.

```
heat_template_version: 2018-03-02
description: Deploy a simple web server
```

Section "resources"

- Décrit les ressources
- Liste des ressources à déployer, avec leurs types, noms et propriétés.

```
resources:
  my_instance:
    type: OS::Nova::Server
    properties:
      image: "Ubuntu 18.04"
      flavor: "m1.small"
```

Section "outputs"

- Définit les sorties (optionnelle)
- Les sorties fournissent des informations sur les ressources déployées (ex: adresses IP, URL).

```
outputs:
  instance_ip:
    description: The IP address of the deployed instance
    value: { get_attr: [my_instance, first_address] }
```

Section "parameters"

- Définit les paramètres d'entrée (optionnelle) :
- Les paramètres permettent de personnaliser le déploiement (ex: choix de l'image, de la taille de l'instance).

```
parameters:
  image_name:
    type: string
    default: "Ubuntu 18.04"
    description: Name of the image to use for the instance
```

Exemple complet

```
heat_template_version: 2018-03-02
description: Deploy a simple web server

parameters:
  image_name:
    type: string
    default: "Ubuntu 18.04"
    description: Name of the image to use for the instance

resources:
  my_instance:
    type: OS::Nova::Server
    properties:
      image: { get_param: image_name }
      flavor: "m1.small"

outputs:
  instance_ip:
    description: The IP address of the deployed instance
    value: { get_attr: [my_instance, first_address] }
```

Ressources, propriétés et attributs

- Ressources : éléments d'infrastructure déployés par Heat (ex: instances, réseaux)
- Propriétés : spécifications et configurations des ressources
- Attributs : informations dynamiques sur les ressources, accessibles après déploiement
- Liste complète des ressources et propriétés dans la documentation OpenStack

Fonctions intrinsèques

- Fonctions pour manipuler et traiter les données du template
- Elles facilitent la gestion des ressources, des attributs et des paramètres dans un template HOT.
- Exemples : `get_attr`, `get_param`, `get_resource`, ref
- Permettent d'éviter la duplication de code et de simplifier la maintenance
- Documentation complète des fonctions intrinsèques dans la documentation OpenStack

📚 OpenStack Documentation: Fonctions intrinsèques

get_attr

- Utilisée pour récupérer la valeur d'un attribut d'une ressource
- Syntaxe : `{ get_attr: [ resource_name, attribute_name ] }`
- Exemple :

```
resources:
  my_instance:
    type: OS::Nova::Server
    properties:
      ...

outputs:
  instance_ip:
    description: The IP address of the instance
    value: { get_attr: [ my_instance, first_address ] }
```

get_param

- Utilisée pour récupérer la valeur d'un paramètre d'entrée du template

- Syntaxe : `{ get_param: parameter_name }`
- Exemple :

```
parameters:
  image_id:
    type: string
    description: ID of the image to use for the instance

resources:
  my_instance:
    type: OS::Nova::Server
    properties:
      image: { get_param: image_id }
      ...
```

get_resource

- Utilisée pour récupérer le nom d'une ressource
- Syntaxe : `{ get_resource: resource_name }`
- Exemple :

```
resources:
  my_instance:
    type: OS::Nova::Server
    properties:
      ...

  my_volume:
    type: OS::Cinder::Volume
    properties:
      ...

  my_volume_attachment:
    type: OS::Cinder::VolumeAttachment
    properties:
      instance_uuid: { get_resource: my_instance }
      volume_id: { get_resource: my_volume }
```

ref

- Utilisée pour référencer une ressource, généralement pour récupérer son ID
- Syntaxe : `{ ref: resource_name }`
- Exemple :

```
resources:
  my_instance:
    type: OS::Nova::Server
    properties:
```

```
    ...

my_security_group:
  type: OS::Neutron::SecurityGroup
  properties:
    ...

my_instance_security_group:
  type: OS::Nova::ServerSecurityGroup
  properties:
    server: { ref: my_instance }
    security_group: { ref: my_security_group }
```

Commandes de gestion de stack

- Déploiement d'une stack

```
openstack stack create \
  -t my_stack.yaml my_stack
```

- Voir la stack crée

```
openstack stack show my_stack
```

- Lister les stack

```
openstack stack list
```

- Lister les evenements d'une stack

```
openstack stack event list my_stack
```

- Lister les ressources d'une stack

```
openstack stack resource list my_stack
```

- Supprimer une stack

```
openstack stack delete my_stack
```

Gestion des instances dans Heat

Création d'instances via Heat

- Utilisation de la ressource OS::Nova::Server pour définir une instance
- Spécification des propriétés essentielles : `image`, `flavor`, `key_name`

```
heat_template_version: 2018-03-02

resources:
  my_instance:
    type: OS::Nova::Server
    properties:
      image: "cirros-0.3.5-x86_64-disk"
      flavor: "m1.tiny"
      key_name: "my_key"
```

📚 OpenStack Documentation: Software configuration

Configuration et personnalisation des instances

- Utilisation de `user_data` pour exécuter des scripts lors du lancement de l'instance
- Utilisation des métadonnées pour définir des valeurs personnalisées
- Passage de paramètres via Heat pour personnaliser l'instance lors de la création

User data

```
resources:
  my_instance:
    type: OS::Nova::Server
    properties:
      image: "Ubuntu 20.04"
      flavor: "m1.small"
      key_name: "my_keypair"
      user_data: |
        #!/bin/bash
        apt-get update
        apt-get install -y apache2
```

Cloud init

```
cloud-config
package_upgrade: true
packages:
 - apache2
write_files:
 - path: /var/www/html/index.html
   content: |
     <html>
       <head>
         <title>Welcome to my website!</title>
       </head>
       <body>
         <h1>Hello, World!</h1>
       </body>
     </html>
runcmd:
 - systemctl enable apache2
 - systemctl start apache2
```

Meta données

```
resources:
  my_instance:
    type: OS::Nova::Server
    properties:
      image: "Ubuntu 20.04"
      flavor: "m1.small"
      key_name: "my_keypair"
      metadata:
        environment: "production"
        app_name: "my_app"
```

Paramètres

```
parameters:
  instance_name:
    type: string
    description: Name of the instance

resources:
  my_instance:
    type: OS::Nova::Server
    properties:
      name: {get_param: instance_name}
      image: "Ubuntu 20.04"
      flavor: "m1.small"
      key_name: "my_keypair"
```

```
openstack stack create -t my_template.yaml --parameter
"instance_name=my_custom_instance" my_stack
```

Gestion des réseaux

Création et configuration de réseaux virtuels via Heat

- Utilisation de la ressource `OS::Neutron::Net` pour créer un réseau virtuel
- Spécification des propriétés : `name`, `shared`, `tenant_id`

```
resources:
  my_network:
    type: OS::Neutron::Net
    properties:
      name: "my_network"
      shared: false
```

```
openstack stack create -t network_template.yaml my_network_stack
```

Gestion des sous-réseaux et des routeurs

- Utilisation de la ressource `OS::Neutron::Subnet` pour créer un sous-réseau
- Spécification des propriétés : `network_id`, `cidr`, `gateway_ip`, `allocation_pools`
- Utilisation de la ressource `OS::Neutron::Router` pour créer un routeur
- Spécification des propriétés : `external_gateway_info`, `name`

```
  resources:
    my_net:
      type: OS::Neutron::Net
      properties:
        name: my_net

    my_subnet:
      type: OS::Neutron::Subnet
      properties:
        network_id: { get_resource: my_net }
        cidr: 192.168.1.0/24

    my_router:
      type: OS::Neutron::Router
      properties:
        name: my_router
        external_gateway_info:
          network: { get_param: external_network_id }

    my_router_interface:
      type: OS::Neutron::RouterInterface
      properties:
        router_id: { get_resource: my_router }
        subnet_id: { get_resource: my_subnet }


---
```

Gestion des volumes

Création de volumes via Heat

- Utilisation de la ressource `OS::Cinder::Volume` pour créer un volume
- Spécification des propriétés : `size`, `image`, `name`

```
resources:
  my_volume:
    type: OS::Cinder::Volume
    properties:
      size: 10
      name: MyVolume
      image: <image_id>
```

Attachement de volumes

- Utilisation de la ressource `OS::Cinder::VolumeAttachment` pour attacher un volume à une instance
- Spécification des propriétés : `instance_uuid`, `volume_id`, `mountpoint`

```
resources:
  my_volume_attachment:
    type: OS::Cinder::VolumeAttachment
    properties:
      instance_uuid: { get_resource: my_instance }
      volume_id: { get_resource: my_volume }
      mountpoint: /dev/vdb
```

Configuration avancée des volumes

- Utilisation de la ressource `OS::Cinder::Volume` pour configurer un volume
- Spécification des propriétés : `availability_zone`, `metadata`, `volume_type`

```
my_volume:
  type: OS::Cinder::Volume
  properties:
    size: 10
    availability_zone: nova
    metadata:
      role: database
    volume_type: ssd
```

Utilisation des snapshots

- Utilisation de la ressource `OS::Cinder::VolumeSnapshot` pour créer un snapshot
- Spécification des propriétés : `volume_id`, `name`, `description`
- Exemple de création et configuration d'un snapshot dans un template HOT

```
my_volume_snapshot:
  type: OS::Cinder::VolumeSnapshot
  properties:
    volume_id: { get_resource: my_volume }
    name: "Database Volume Snapshot"
    description: "Snapshot of the database volume"
```

l'API

Pour chaque service, openstack propose une api avec un endpoint

(cf Manage Compute > Access & security)

Le job d'openstack c'est de fédérer les différents projets de les faire fonctionner ensemble et d'uniformiser la communication/le paramétrage

La CLI

```
pip install python-openstackclient
```

Le fichier openrc.sh Les variables du fchier openrc.sh Sinon utiliser les parametres

```
--os-username
--os-password
--os-project-id
--os-project-name
--os-auth-url
```

Le SDK

Destiné à d'autres langages Destiné aux développeurs Plus facile en python (car la lib existe déjà) Les autres langages utilisent l'API

Heat

Enjeux: re-déployer, templatiser, prendre en compte l'infra existante

```
User
Heat CLI tools
Heat API
Cloutwatch API
MQ
Heat Engine
DB
```

Ensemble des ressources = 1 stack

Exemple de fichier heat (yaml)

```
---
heat_template_version:
parameters:
  key_name:
    type: string
    description:
    default:
  network_name:
    type: string
    description:
    default:
resources:
  toto1:
    type: OS::Nova::Server
  properties:
    key_name: {get_param: key_name}
    flavor: t1.devops
    image: rhel6.2.1
    networks: [{ network: {get_param: network_name} }]
```

Terraform

Multi cloud (pas seulement OS)

Merci pour votre attention !

bg left

Advanced topics

bg right:33%

Deploying and Managing a Production OpenStack Environment:

- Best practices for deploying a production-ready OpenStack environment
- Planning for high availability and scalability
- Best practices for managing and maintaining an OpenStack cloud
- Creating custom images, flavors, and scripts for automating OpenStack deployment

Managing and Scaling an OpenStack Cloud:

- Strategies for scaling an OpenStack cloud to meet changing demands
- Managing and optimizing the performance of an OpenStack cloud
- Strategies for monitoring and troubleshooting an OpenStack cloud
- Managing and maintaining an OpenStack cloud in a multi-tenant environment

Troubleshooting and Monitoring OpenStack:

- Understanding OpenStack log files and troubleshooting tools
- Understanding OpenStack service logs and log rotation
- Tools and techniques for monitoring and troubleshooting an OpenStack cloud
- Best practices for maintaining an OpenStack cloud in a production environment

OpenStack in a Multi-node Environment:

- Configuring a multi-node OpenStack environment
- Deploying and configuring HA for OpenStack components
- Multi-node deployment strategies
- Best practices for managing and maintaining a multi-node OpenStack environment

Final Project

- Students will work on a group project to deploy and manage a multi-node OpenStack cloud environment, simulating real-world scenarios and providing an opportunity to apply the knowledge and skills gained throughout the course.
- Project evaluations and presentations

Chat

Chat 1: https://chat.openai.com/chat/028d02ee-dfbf-42a2-8422-5b2447132526

Prompt 00 crystalization

[TECHY is a technology expert with more than 20 years of experience. TECHY are specialized in systems, networks and infrastructures. They understand technology strength, opportunities, weaknesses and threats). TECHY focuses only on the technical and technological aspects of things. TECHY is methodical and analytical in his answers. TECHY admits his incompetence on other topics. TECHY behaves like a blend of Linus Torvalds, Marc Andreessen, Paul Graham, Bruce Schneier, Bram Cohen, Philip Zimmermann, Jon Callas, Derek Atkins, Adam Back, Theo de Raadt, Nick Szabo, Runa Sandvik, David Chaum, Richard Matthew Stallman, Alan Turing, Ada Lovelace. I all messages, you start your reply by “TECHY:” and then you will answer to the questions or to the prompt like if TECHY was answering himself.]

Contexte

Une formation

Public concerné

- Architectes,
- Chef de projets,
- Administrateurs,
- Ingénieurs système et réseau,
- et toute personne souhaitant installer une infrastructure de Cloud avec OpenStack

Prérequis

- Avoir une connaissance générale des systèmes d'informations, systèmes et réseaux IP.
- Avoir de bonnes connaissance Linux

Programme de la formation

Conception d'un Cloud OpenStack

- Apports et spécificités d'un Cloud. Type de Clouds : SaaS, PaaS, IaaS, publics, privés.
- Le projet OpenStack. Organisation et structure.
- Architecture de la plateforme. Vue d'ensemble des différentes briques.
- Méthodes d'installation.

Travaux pratiques

- Comparaison et sélection d'une méthode d'installation et installation.

Gestion des machines virtuelles (Nova)

- Présentation de la brique Nova. Mise en œuvre et configuration.
- Gestion des images et des instances. Gestion du réseau virtuel.
- Gestion d'hyperviseurs multiples (Hyper-V, ESXi, KVM).

Travaux pratiques

- Création d'une machine virtuelle depuis Nova.

Gestion du stockage (Swift)

- Vue d'ensemble de Swift.
- Mise en œuvre et configuration.
- Gestion des pools de stockage.
- Mise en œuvre du stockage en mode bloc avec Cinder.
- Backend supportés par Cinder.

Travaux pratiques

- Gestion du stockage avec Cinder.

Gestion des images (Glance)

- Qu'est-ce qu'une image ?
- La brique de gestion des images Glance.
- Création de la base de données. Mise en œuvre et configuration.

- Gestion du stockage des images. La gestion des images EC2 (AMI).

Travaux pratiques

- Créer et configurer des images.

Gestion du réseau

- Vue d'ensemble de la brique Neutron (anciennement Quantum).
- Switchs virtuels avec Open vSwitch.
- Topologies de réseau Cloud.
- Daemon de routage (L3).
- Mise en œuvre et configuration.

Travaux pratiques

- Créer et configurer un réseau virtuel.

Authentification et autorisations

- Présentation de la brique Keystone.
- Création des utilisateurs, projets et rôles.
- Mise en œuvre et configuration.
- Configuration des utilisateurs, projets et rôles.

Travaux pratiques

- Gestion des utilisateurs et des services.

Administration du Cloud

- Vue d'ensemble du client Web Horizon.
- Automatisation avec l'API REST.
- Présentation des API Amazon EC2et S3.
- Automatisation du Cloud avec un outil d'automatisation et d'orchestration comme Cloud-init.

Travaux pratiques

- Administration d'OpenStack depuis Horizon.
- Utilisation de Cloud-init ou d'un autre outil d'automatisation et d'orchestration.

Déploiement

- Présentation de Heat et des outils d'Infrastructure As Code

Prompt 01 chapters

Merci.

Suit la structure du PROGRAMME DE FORMATION.

Concentre toi sur le chapitre « FIXME ».

Focalise toi plus spécifiquement sur les sujets suivants «

FIXME

» de ce chapitre sur lequel on se concentre.

Rédige la table des matiere détaillée de ces différents sujets.

Prompt 02 sub chapter

Super.

Suit la structure du PROGRAMME DE FORMATION.

Concentre toi sur le chapitre « [[FIXME1]] »

Dans ce chapitre, concentre toi sur le sous-chapitre « [[FIXME3]] ».

Focalise toi plus spécifiquement sur les sous-sujets suivants «

[[FIXME3]]

» de ce sous-chapitre, sur lequel on se concentre.

Rédige le contenu détaillée de ces différents sous-sujets, sous forme phrases courtes et de listes à puces (bullet-points) pour remplir le contenu d'une présentation PowerPoint.

Donne des informations plus précises et plus techniques.

Précise les mots "gérer" ou "permet" quand tu les utilises.

Indique où trouver les informations dans Horizon, dans la CLI nova ou la CLI openstack.

Prompt 03 fix content

@@ TEXTE A CORRIGER

[[FIXME: text]]

@@ REQUETE

Dans le TEXTE A CORRIGER, indique où sont les erreurs et les approximations, et propose des corrections pour améliorer le contenu (pour une formation sur OpenStack).

Prompt 04 example code

Super. Suit la structure du PROGRAMME DE FORMATION. Concentre toi sur le chapitre « {{this.parent.title}} » Focalise toi plus spécifiquement sur les sous-sujets suivants « {{this.title}} » Explique la partie suivante : «

{{this.children}}

»

Donne également des exemples d'utilisation avec des exemples en ligne de commande ou des extrait de code, ou de configuration.

Syllabus

Programme de la formation

Please use the following syllabus for the 'beging to advanced Openstack administrator course'

Week 1: Introduction to Openstack

- Overview of the Openstack platform and its components
- Setting up an Openstack development environment
- Basic Openstack commands and usage

Week 2: Openstack Compute (Nova)

- Understanding Nova architecture and components
- Managing virtual machines and instances
- Configuring and managing flavors and images

Week 3: Openstack Networking (Neutron)

- Understanding Neutron architecture and components
- Managing virtual networks and subnets
- Configuring security groups and firewall rules

Week 4: Openstack Storage (Cinder and Swift)

- Understanding Cinder and Swift architecture and components
- Managing block and object storage
- Creating and managing volumes and snapshots

Week 5: Openstack Identity (Keystone)

- Understanding Keystone architecture and components
- Managing users and projects
- Configuring authentication and authorization

Week 6: Openstack Dashboard (Horizon)

- Understanding Horizon architecture and components
- Navigating and using the Openstack dashboard

- Customizing and extending the dashboard

Week 7: Advanced Openstack topics

- Deploying and managing a production Openstack environment
- Managing and scaling an Openstack cloud
- Troubleshooting and monitoring Openstack
- Openstack in a multi-node environment

Week 8: Project and Exam

- Students will be given a project to deploy a multi-node OpenStack cloud, and they will need to complete it
- Written Exam will be held to evaluate the student's understanding of the course content and their ability to apply the knowledge to real-world scenarios

ME

Please follow the syllabus structure and write the detailed content for "week 1 : introduction to openstack"

ME

Follow the syllabus and the content structure above, and write the detailed course content for "Basic Openstack commands and usage"

ME

Please provide all explanations and commands for the "Configuring and managing flavors and images" part, as the content of multiple slides in markdown, separated by "----":

Notes1

- Différence entre les services Core et optionels

Pourquoi Openstack

Introduction au Cloud

Modeles de cloud

- 4 modeles (classique, IaaS, PaaS, SaaS)
- (schema des responsabilités)
- les publics qui correspondent (devops, devs, utilisateurs)

Avantages

- efficacité, fiabilité, flexibilité
- applications comme utilités sur internet
- configuration et manipulation des apps en ligne
- aucun logiciel requis
- outils de déploiement et developmeent en ligne
- resources disponibles en ligne
- libre service sur demande
- rentable et économique

Acteurs internationnaux du cloud

- Amazon (AWS)
- VMWare
- Google (GCP)
- Apache (CloudStack)
- Rackspace
- Microsoft (Azure)
- Heroku

- Openstack

et aussi

- terremark ?
- softlayer ?
- savvis ?

Architecture d'Openstack

Presentation du projet

- Middleware Cloud ouvert / libre
- Solution de cloud privé
- Naissance en 2010
    - Rackspace (stockage) + NASA (calcul)
- Fondation Openstack depuis 2012
    - Juridique
    - RH (dev, marketing, release manager)
    - Infrastructure
    - Organisation des OpenStack summit (4,5K participants)
    - 500 organisations
    - 23K membres individuels
- Ecrit en python et distribué sous licence Apache 2
- Nombreux contributeurs
- Versions
    - Dernieres releases: ...
    - 6 mois d'intervale entre les versions
    - Notion de LTS (FIXME: screenshot)
- Alternative:
    - Cloudstack (Apache) https://cloudstack.apache.org/
    - Eucalyptus (https://www.eucalyptus.cloud/)
    - OpenNebula (https://opennebula.io/)

utilisateurs d'openstack

Contributeurs

- ~1500 contributeurs
- 70 organisations
- • de 7K bogues corrigés
- 800K chaînes traduites
- RedHat, IBM, Dell, Intel, Cisco, Juniper, NetApp, HP, VMWare, Google, Yahoo,

Développement

- Ouvert à tous
- Cycles courts
- Git, Github, Gerrit, Launchpad, Jenkins
- Très actifs (17k commits, progression annuelle +25%)

Utilisateurs

- Public: CERN, Harvard, INFN, MIT, Wikimedia, DGFIP,
- Privé: OVH, Paypal, Seagate, Orange, Disney, Sony, BMW, etc.
- Cloud souverain FR: Cloudwatt (Orange, Thales), Numergy (SFR, Bull)

Architecture

Version simple

- du hardware standard
- OS organisé autour de "services"
- un dashboard (transverse)
- une API

Composants Core (vue détaillée)

- Dashboard
- Compute
- Network
- Image
- Object Storage

- Block Strorage
- Identity

FIXME: schéma des communication entre macro-services

Composants Core (vue micro)

FIXME: schéma des communication entre micro-services

TP 0 : installation

Prerequis

- Connaissances en virtualisation et bases réseau
- Virtualbox
- 8Go de RAM
- 4 VCPU
- 50 Go d'espace disque

=> on se focalise d'abord sur l'utilisation avant d'apprendre à le déployer

Configurer les réseaux

- Management network
- Internal network
- External network
- External network (again)

Forcer "Allow all" sur le promiscuous mode

Ref. https://docs.openstack.org/kolla-ansible/latest/admin/production-architecture-guide.html

Ref. https://docs.openstack.org/devstack/latest/networking.html

TP 0: recovery

Relancer la BDD

```
$ kolla-ansible -i all-in-one -e mariadb_recovery
```

Relancer Nova

```
$ kolla-ansible -i all-in-one --tags nova
```

Password

```
$ cat /etc/kolla/admin-openrc.sh
```

TP 0 : Se connecter à une VM

Présentation des services core

Nova

Glance

...

Vocabulaire

Identité et acces

- Tenant/Projet
- Utilisateur
- Quota
- Catalogue Endpoint

Calcul/Serveur:

- image
  - OS que l'on souhaite déployer
  - installation déjà réalisé & packagée
- instance
  - vm en cours d'exécution dans Openstack
- type d'instance (ou flavor)
  - configuration de la machine (= sa "taille")

- user data
  - customiser le déploiement de la machine
  - ex: commandes à exécuter à la finalisation de la configuration de la vm
- metadata
  - rendre disponible des informations à l'ensemble des VM des info les concernant
  - permet l'introspection (ex: récupérer l'id, etc.)
- cloud-init
  - systeme de config des VM
  - choix de services operationels, etc.

Stockage

- Volume
  - Volumes Cinder
- Conteneur
  - Objets Cinder / Swift

Réseau et sécurité

- Groupe de sécurité
  - Définition d'une politique de sécurité
  - pour les VMs
- Paire de clefs
  - clefs à utiliser pour l'acces en SSH aux VMs
- IP flottante
  - permet d'accéder aux VMs depuis l'extérieur

Orchestration

- Stack
  - Ensemble de resources déjà deployées
- Template

TP-2 : creer votre projet

Voir exercices/02-project-setup/ENONCE.md

TP-3 : Déployer votre premiere instance

Voir exercices/03-first-instance/ENONCE.md

Services complémentaires

Cf. screenshot services complémentaires

FIXME: TP: ajouter des services complémentaires (ex: Trove, ou Manila, ou Ceilometer, ou Heat)

Q: Log Visualizer ? (quelle intégration?)

Q: Minitoring (telegraf, influxdb, grafana)

OpenStack Kolla Ansible

Notion de micro-services

- Monolith vs Microservices
- Notion d'API
- Docker, etC.

Présentation de Ansible

-

Présentation de Kolla

- Projet géré par OpenStack
- 3 réseaux

Mini-projet

GYR: Questions

equivalent CLI de chaque opération ?

c'est quoi un vCPU ? (comment c'est défini?)

* https://wiki.openstack.org/wiki/VirtDriverGuestCPUTopology * https://docs.openstack.org/nova/pike/admin/cpu-topologies.html

fonctionnement des IP flottantes ?

Floating IP addresses in OpenStack serve as a way to provide externally accessible, static IP addresses to instances within a private cloud. They are essential for enabling communication between instances and external networks, as well as for providing stable, reachable endpoints for applications or services running on those instances. Floating IPs are particularly useful in situations where you need to maintain consistent access to an instance or service, even when the underlying virtual machine might change or be replaced.

Here's how floating IPs work in OpenStack:

> Allocation: In OpenStack, floating IPs are allocated from a pool of available public IP addresses. These addresses are managed by the OpenStack Networking service (Neutron). An administrator or user with appropriate permissions can allocate a floating IP from the pool.
>
> Association: Once a floating IP is allocated, it can be associated with an instance. This association is essentially a mapping between the floating IP and the private IP address of the instance. This allows external traffic to be directed to the correct instance within the private network.
>
> NAT (Network Address Translation): OpenStack uses NAT to enable communication between the floating IP address and the private IP address of the instance. When external traffic reaches the floating IP, it is translated to the private IP address of the instance by a Neutron router or other network devices. This translation is performed in both directions, allowing bidirectional communication between the instance and external networks.
>
> Disassociation and Release: If needed, a floating IP can be disassociated from an instance and either associated with another instance or released back into the pool of available floating IPs. This flexibility allows users to easily reassign public IP addresses to different instances as needed.

References

Intro

https://www.openstack.org/use-cases/

Awesome

https://github.com/ntk148v/awesome-openstack https://github.com/ramitsurana/awesome-openstack

Videos

https://www.openstack.org/videos/

Project

- Students will work in groups to deploy a multi-node OpenStack cloud using the technologies and concepts covered in the previous weeks of the course.
- The project should include at least the following components: Nova, Neutron, Cinder, Swift, Keystone, and Horizon.
- Students will need to document their design decisions and provide a step-by-step guide for how to recreate their deployment.
- Additionally, students are required to provide a summary of the limitations, scalability concerns, and any issues encountered during the project.

Exam

- The exam will be a written test, which will be taken individually.
- The exam will cover all topics and concepts covered in the course, including the ones covered in the project work.
- It will contain multiple-choice and short-answer questions.
- The exam is designed to evaluate the students' understanding of the course content and their ability to apply the knowledge to real-world scenarios
- Students are expected to have a good understanding of the following topic: OpenStack architecture, deployment, and management, OpenStack storage, network and compute services and OpenStack security.
- The exam will be graded out of 100 points, and the passing grade will be determined by the instructor.