Administrer votre cloud Openstack sans aspirine avec Kolla

Rémi Cailletaud, Anthony Defize, Nicolas Gibelin, Gabriel Pierre André Moreau

Administrer votre cloud Openstack sans aspirine avec Kolla

Rémi Cailletaud
OSUG – UAR 832
122 rue de la piscine
38 400 Saint-Martin-d'Hères

Anthony Defize
GRICAD – UAR 3758
Bâtiment IMAG
700 avenue centrale
38 400 Saint-Martin-d'Hères

Nicolas Gibelin
GRICAD – UAR 3758
Bâtiment IMAG
700 avenue centrale
38 400 Saint-Martin-d'Hères

Gabriel Moreau
LEGI – UMR 5519
1209-1211 rue de la piscine
38 400 Saint-Martin-d'Hères

Résumé

En 2019, l'équipe en charge du cluster Openstack du campus grenoblois (Nova) présentait la genèse du projet d'une plateforme de cloud computing. Après quelques années d'exploitation, il nous semblait pertinent de nous pencher plus en détail sur la solution que nous avons retenue pour le déploiement et la gestion de cette infrastructure, à l'exception notable de la partie stockage, basée sur Ceph.

Le projet Openstack est devenu la solution libre de cloud computing de référence. Son architecture modulaire et les nombreux services qui la composent permettent de couvrir un très large spectre d'utilisation et représente de ce fait un véritable défi pour les administrateurs, d'autant plus quand les problématiques de haute disponibilité et de passage à l'échelle s'invitent à la table.

Pour Nova, qui propose actuellement 1152 cœurs de calcul, 5 To de RAM, 200 To de stockage distribué et des GPU, nous avons choisi Kolla-Ansible, qui se base sur des services conteneurisés déployés via Ansible. Nous éclairerons les raisons de nos choix, détaillerons le fonctionnement et présenterons les nombreux avantages de cette solution pour le déploiement, l'exploitation quotidienne et les opérations de maintenance importantes telles que les mises à jour, sans oublier d'évoquer rapidement les quelques frayeurs, rares, qui ont parsemé le projet.

Mots-clefs

OpenStack, Cloud Computing, Nuage, IaaS, Ansible, Kolla

1 Introduction

OpenStack est devenue la solution de référence de l'informatique dans les nuages (cloud computing) du monde des logiciels libres. Utilisée sur des millions de cœurs, c'est aussi l'un des projets open source le plus actif actuellement. Son architecture modulaire et les nombreux services qui composent une infrastructure OpenStack permettent de couvrir un très large spectre d'utilisation. C'est l'une des raisons de son déploiement dans de nombreuses structures universitaires, notamment une sur Grenoble.

Détaillons rapidement les atouts de ce type de solution. Plutôt que de réinventer la roue, OpenStack est bâti autour de nombreuses briques logicielles largement éprouvées (MariaDB, Memcached, Open vSwitch…) de l'écosystème GNU+Linux. Il n'y a pas un service, mais des services et ils sont très nombreux. Tous ne sont heureusement pas nécessaires ni utiles dans votre infrastructure. Chaque service, ou composant de base (authentification avec Keystone, virtualisation avec Nova, réseau avec Neutron, stockage avec Cinder), gère une tâche spécialisée. Tous les composants s'échangent des messages au travers d'une API standard utilisant souvent RabbitMQ (ZeroMQ est une autre possibilité, mais elle est moins bien supportée).

Si cette architecture interne entièrement modulaire est une force en termes de développement et d'évolution, il en résulte aussi un système dont le déploiement et l'exploitation peuvent s'avérer complexes. Si on y ajoute les contraintes liées aux environnements de production (haute disponibilité des différents composants, architecture conséquente exposée aux pannes matérielles), la mise en œuvre et la gestion manuelle d'un cluster OpenStack deviennent un sacré challenge ! Heureusement, il existe des solutions permettant l'automatisation de ceux-ci. Ainsi, l'administrateur de la plateforme n'est pas abandonné à son triste sort : nous allons voir comment faciliter son travail initial puis quotidien.

2 Contexte

En 2018, l'UAR GRICAD (Unité d'Appui à la Recherche – anciennement UMS), soutenue par l'ensemble de ses tutelles, a décidé de déployer un service d'infrastructure à la demande (IaaS) basé sur la suite logicielle OpenStack. Nom de code : Nova. Cette plateforme, opérée par une équipe de cinq personnes pour un équivalent temps plein d'une personne, est née pour compléter l'offre des plateformes de calcul haute performance (HPC) et d'infrastructures VMware, celles-ci ne pouvant pas répondre simplement aux nouvelles problématiques telles que le Big Data, la formation (TP numérique)… Nous avons aussi rapidement intégré des nœuds GPU pour les besoins en développement dans les domaines de l'intelligence artificielle, mais aussi pour la formation à l'utilisation des suites logicielles de calcul GPU.

Depuis 2019, la plateforme a connu un grand succès auprès des utilisateurs : elle accueille des projets de recherche (maquettage GPU, fouille de données) et aussi des applicatifs de production (une partie des services de la plateforme en ligne du réseau Mathrice[1] en particulier). Elle prend donc régulièrement de l'ampleur aussi bien en capacité de stockage qu'en puissance de calcul. Notre système primaire de stockage Ceph est ainsi passé de 3 à 8 nœuds de stockage, principalement pour améliorer les performances (520 To bruts, à diviser globalement par trois).

1 https://plm-doc.math.cnrs.fr/doc/

Côté OpenStack, nous avons ajouté dix nœuds dont deux nœuds GPU Tesla V100 et V100S, amenant notre capacité totale à 1152 cœurs, 5 To de mémoire vive et 4 GPU, configurés en vGPU.

Dès le début, nous avons cherché une solution capable de gérer tout le cycle de vie : *bootstrap* des machines, déploiement d'OpenStack, mises à jour de celui-ci, extension et réduction de l'infrastructure, ajout ou suppression de fonctionnalités. Nous cherchions une solution libre avec un support communautaire, l'enveloppe budgétaire ne permettant pas de nous offrir un contrat de support.

Parmi les nombreuses solutions de déploiement d'une infrastructure OpenStack, nous avons fait le choix de Kolla. Plus particulièrement, nous avons opté pour la version Kolla-Ansible qui comme son nom l'indique se base sur un ensemble de recettes Ansible. Dans cette solution, tous les services sont conteneurisés. Après un premier article présenté aux JRES 2019 qui détaillait nos choix [1] et trois ans d'utilisation en production, nous avons le recul suffisant pour présenter cette solution plus en détail : son fonctionnement, ses forces et ses faiblesses, ainsi que plusieurs retours d'expérience concernant en particulier les mises à jour majeures de versions.

3 Kolla

3.1 Présentation

Kolla est disponible depuis OpenStack Mikata. Supporté dès sa création par Red Hat, Cisco ou encore Rackspace (qui est à l'origine d'OpenStack avec la NASA), ils ont été rejoints pas un grand nombre de contributeurs dont 99Cloud, un fournisseur de service chinois et StackHPC, qui développe des solutions de calcul hautes performances basées sur OpenStack. La figure 1 présente les principaux contributeurs au projet.

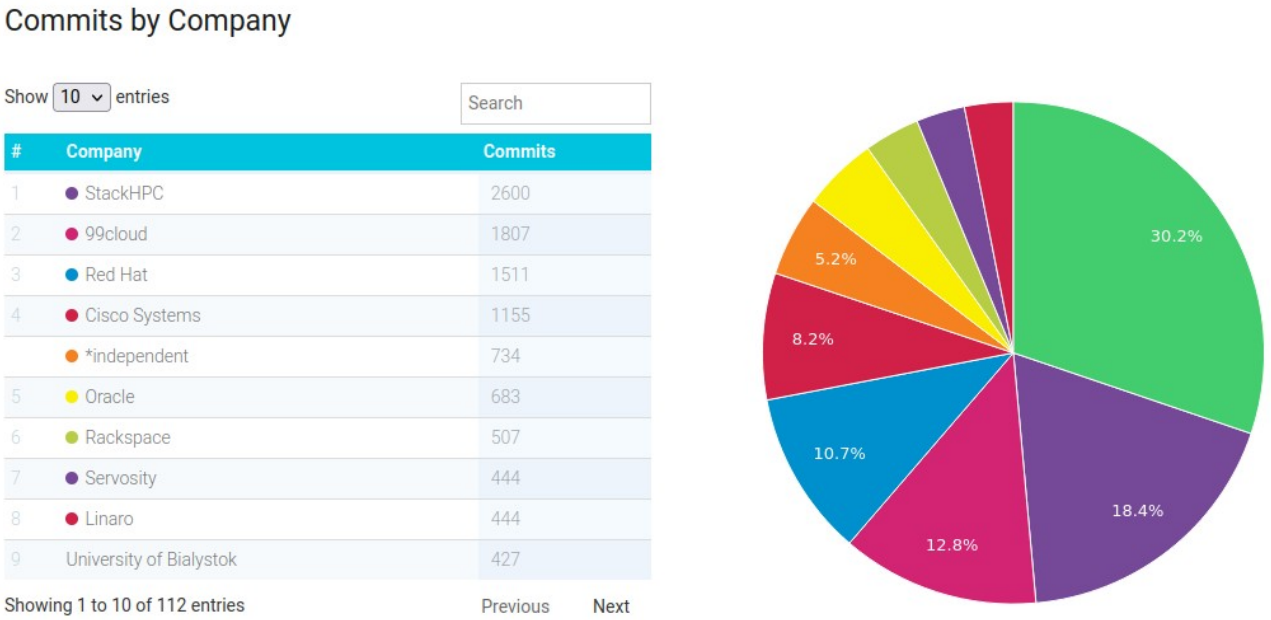

Figure 1: Statistiques de contributions à Kolla

3.2 Objectifs

L'objectif de Kolla est de fournir les conteneurs et les outils de déploiement pour opérer les clouds OpenStack. Initialement, le but était de pouvoir déployer les projets de la *Big Tent*[2], c'est-à-dire les principaux composants d'OpenStack.

Kolla propose une infrastructure *immutable* : pour les mises à jour et reconfiguration, les composants ne sont pas modifiés mais totalement remplacés. Cela facilite le suivi des opérations au jour le jour et permet des mises à jour et des retours en arrière faciles.

Kolla supporte plusieurs distributions Linux : CentOS, Debian, RHEL, Ubuntu. Les dépendances des systèmes hôtes sont très limitées : elles se limitent à Ansible[3], Docker[4] et la bibliothèque docker-py.

Il offre une configuration fonctionnelle par défaut et qui est aussi complètement personnalisable. Elle permet par exemple de configurer très facilement un plan de contrôle hautement disponible.

3.3 Composants principaux

Les principaux composants déployés par Kolla sont les suivants :

- HAProxy[5] et Keepalived[6] pour les services en mode haute disponibilité ;
- MariaDB[7] en mode Galera[8] pour les bases MySQL ;
- Fluentd[9] pour la collecte de logs ;
- Elasticsearch[10] et Kibana[11] pour l'analyse de logs ;
- Chrony[12] pour le NTP ;
- les services classiques et nécessaires à OpenStack : RabbitMQ[13], Memcached[14] et Open vSwitch[15] ;
- et enfin, les composants OpenStack désirés.

2 https://www.openstack.org/videos/summits/vancouver-2015/the-big-tent-a-look-at-the-new-openstack-projects-governance

3 https://www.ansible.com/

4 https://www.docker.com/

5 https://www.haproxy.org/

6 https://www.keepalived.org/

7 https://mariadb.com/

8 https://mariadb.com/kb/en/library/galera-cluster/

9 https://www.fluentd.org/

10 https://www.elastic.co/de/products/elasticsearch

11 https://www.elastic.co/products/kibana

12 https://chrony.tuxfamily.org/

13 https://www.rabbitmq.com/

14 https://www.memcached.org/

15 https://www.openvswitch.org/

3.4 Des images de conteneurs

La mission première de Kolla est de fournir des images de conteneurs pour les différents composants tiers nécessaires (HAProxy, Memcached, MariaDB…) et les différents composants OpenStack (une quarantaine pour la version OpenStack Xena).

Afin de respecter au mieux la bonne pratique « un processus = un conteneur », les différents projets sont généralement séparés en plusieurs micro services. Par exemple, pour Cinder, nous trouverons trois images de conteneur : cinder-volume, cinder-scheduler, cinder-api.

L'image de base des conteneurs peut-être basée sur différentes distributions (CentOS, Debian, RHEL, Ubuntu), disponible avec une installation des services via le gestionnaire de paquets de la distribution (image *binary*) ou via pip depuis les sources (image *source*). Notons cependant que les versions CentOS et les saveurs *binary* seront bientôt dépréciées.

Les images sont construites via un système de *templates* qui permet de construire des images personnalisées très facilement, comme illustré dans la figure 2 : il s'agit d'un *template* de fichier *Dockerfile* pour construire l'image du conteneur Heat API, le service d'orchestration d'Openstack. Les instructions de types *block* permettent la surcharge globale ou par service (voir figure 3). La partie centrale permet l'installation des paquets nécessaires selon la distribution de base et la saveur choisie. Le fichier *kolla_extended_start* est le point d'entrée du conteneur, permettant d'exécuter les migrations de base si nécessaire.

Figure 2: Image du conteneur Heat

```
FROM {{ namespace }}/{{ image_prefix }}heat-base:{{ tag }}
{% block labels %}
LABEL maintainer="{{ maintainer }}" name="{{ image_name }}" build-
date="{{ build_date }}"
{% endblock %}

{% block heat_api_header %}{% endblock %}

{% import "macros.j2" as macros with context %}

{% if install_type == 'binary' %}
    {% if base_package_type == 'rpm' %}
       {% set heat_api_packages = ['openstack-heat-api'] %}
    {% elif base_package_type == 'deb' %}
        {% set heat_api_packages = ['heat-api'] %}
    {% endif %}
{{ macros.install_packages(heat_api_packages | customizable("packages")) }}
{% endif %}

COPY extend_start.sh /usr/local/bin/kolla_heat_extend_start
RUN chmod 755 /usr/local/bin/kolla_heat_extend_start

{% block heat_api_footer %}{% endblock %}
{% block footer %}{% endblock %}
```

Le cluster OpenStack de GRICAD propose un service qui permet de publier automatiquement les noms DNS des IP flottantes dans la solution IPAM EfficientIP SolidServer de l'UGA. Pour cela, nous avons développé un plugin Neutron[16] qui doit être installé dans l'image du conteneur neutron-server. La figure 3 illustre la facilité avec laquelle on peut surcharger une image.

```
{% extends parent_template %}

{% set neutron_server_packages_append = ['python-pip'] %}

{% block neutron_server_footer %}
RUN pip install externaldns-solidserver==0.1.6
{% endblock %}
```

Figure 3: Surcharge de l'image neutron-server (neutron-overrides.j2)

La construction de l'image est tout aussi facile, illustrée dans la figure 4.

```
$ kolla-build --config-file kolla-build.conf \
   --template-override neutron-overrides.j2 neutron-server
```

Figure 4: Construction d'une image personnalisée

3.5 Des scripts Ansible

À l'époque de notre choix, il existait deux manières d'orchestrer les conteneurs : Kolla-Ansible et Kolla-Kubernetes. Comme leurs noms le laissent imaginer, le premier se base sur des scripts Ansible et le second sur des manifestes Kubernetes. Afin de ne pas ajouter une couche supplémentaire de complexité à une époque où Kubernetes n'était pas encore une technologie maîtrisée par tous, nous avons opté pour Kolla-Ansible. Ce choix s'est avéré judicieux puisque depuis, Kolla-Kubernetes a été abandonné au profit d'Openstack-Helm, qui est toujours en développement et ne se base pas sur les images Kolla. Notons que l'utilisation de Kubernetes pour le déploiement est une solution qui se popularise[17] et que nous expérimenterons probablement dans le futur.

Le déploiement via Ansible se déroule comme suit : dans un premier temps, les configurations personnalisées sont fusionnées avec les valeurs par défaut et les fichiers de configuration résultants sont copiés sur les hôtes concernés ; dans un deuxième temps, Ansible lance le *bootstrap* du service, qui crée si nécessaire la base de données et l'utilisateur de la base, et effectue les migrations ; dans un dernier temps, il démarre les services et vérifie leur bonne santé.

16 https://gricad-gitlab.univ-grenoble-alpes.fr/gricad/openstack/externaldns-solidserver

17 https://docs.mirantis.com/mos/latest/ref-arch/openstack/openstack-k8s-arch.html

Outre les configurations des services OpenStack, seuls deux fichiers permettent la configuration du cluster : le fichier Ansible classique d'inventaire (*inventory.yml)* qui liste les nœuds ainsi que leurs rôles respectifs et le fichier *globals.yml* qui définit par exemple la saveur des conteneurs et les images personnalisées, la version d'OpenStack et les services à déployer.

Comme pour la construction d'image, il est très facile de personnaliser la configuration, qui sera fusionnée avec celle par défaut. Dans la figure 5, nous présentons la configuration du service *nova-compute* qui provisionne les instances des machines virtuelles. Nous remarquons deux choses : sa concision (le fichier *nova.conf* effectivement déployé fait une centaine de lignes) et la possibilité de *templating* : le cluster OpenStack de GRICAD propose des vGPU sur certains nœuds et un seul fichier de configuration permet de configurer les différents nœuds en fonction des variables de l'inventaire.

```
[DEFAULT]
vnc_keymap=fr

[libvirt]
cpu_mode = custom
cpu_model = Broadwell-noTSX-IBRS
images_rbd_pool=ephemeral-vms
images_type=rbd
images_rbd_ceph_conf=/etc/ceph/ceph.conf
rbd_user=nova
disk_cachemodes='"network=writeback"'

{% if gpu is defined %}
[devices]
enabled_vgpu_types = {{ gpu }}
{% endif %}
```

Figure 5: Configuration du service nova-compute

4 Retours d'expérience

Dans cette partie, nous allons nous concentrer sur plusieurs retours d'expériences :

- mise à jour ;
- crash brutal de l'infrastructure due à une coupure électrique de la salle provoquée par une mauvaise manipulation lors d'un contrôle électrique ;
- ajout de fonctionnalités.

Ce sera l'occasion de découvrir les possibilités offertes par Kolla dans l'exploitation quotidienne.

4.1 Mise à jour Queen vers Rocky

Ce changement de version majeure Openstack était pour nous le premier depuis l'installation et la mise en production de Nova. Il s'agissait donc d'une épreuve du feu qui pouvait confirmer la

pertinence du choix de Kolla. Notons qu'une montée en version avait eu lieu sur une infrastructure de test minimale.

Le processus de mise à jour se déroule comme suit : tout d'abord, fusion de la nouvelle version du fichier *globals.yml* avec le fichier existant. Pendant ce temps, il est possible de télécharger les images avec la commande `kolla-ansible – i inventory.yml pull`.

Dans un second temps, la commande `kolla-ansible – i inventory.yml upgrade` lance la mise à jour. Les scripts kolla-ansible mettent alors à jour des services les uns après les autres en vérifiant qu'un service fonctionne toujours après l'opération. La mise à jour n'entraîne pas d'interruption de service et les plus attentifs ne remarqueront que quelques pertes de paquets lors des reconfigurations de Neutron.

Cette opération s'était apparemment déroulée sans accroc, mais nous avons rapidement constaté des dysfonctionnements de l'authentification. Une analyse des journaux du service OpenStack chargé de l'authentification (Keystone) nous confirme que la nouvelle gestion des jetons[18] apparue dans OpensStack Rocky n'est pas correctement gérée par Kolla-Ansible. Il nous a fallu nous connecter au conteneur concerné via la commande `docker exec – it keystone_fernet bash` et lancer les scripts de création de jetons manuellement.

Si cet accroc n'a pas eu d'impact sur la connectivité des VM, le portail web et les différents services ont été inaccessibles pendant quelques dizaines de minutes. Il a aussi mis en lumière la documentation parfois lacunaire du projet Kolla-Ansible (même si la situation est en voie d'amélioration) et le besoin d'avoir des connaissances de base de Docker pour corriger les éventuels bugs.

4.2 Mise à jour Rocky vers Train

Au cours de l'exploitation de la plateforme, nous avons pu constater que malgré l'utilisation de Docker, il existe encore des dépendances entre le système hôte et celui des conteneurs : pour certains services, des répertoires applicatifs du système hôte sont partagés, entraînant une dépendance, par exemple pour les versions de Python. Jusqu'ici, nous utilisions Ubuntu pour le système hôte, car il était le mieux connu des différents intervenants, et CentOS pour les conteneurs, car c'était l'implémentation de référence du projet Kolla.

Cette montée en version a été l'occasion de passer hôte et conteneurs en Ubuntu 18.04, qui devenait l'implémentation de référence du projet Kolla suite à l'annonce de CentOS Stream[19].

On notera au passage que Kolla-Ansible supporte les sauts de version, puisque nous sommes alors passés à la version N+2 d'Openstack.

Cette opération délicate s'est déroulée sans accroc et encore une fois sans perte de connectivité pour les machines virtuelles, cette fois grâce au mécanisme de migration à chaud qui nous a permis de redémarrer les hôtes.

18 https://specs.openstack.org/openstack/charm-specs/specs/rocky/implemented/keystone-fernet-tokens.html

19 https://wiki.centos.org/Manuals/ReleaseNotes/CentOSStream

4.3 Perte d'alimentation électrique

En 2020, une fausse manipulation d'un technicien chargé du contrôle électrique annuel de la salle du datacentre grenoblois qui héberge la plateforme provoqua l'arrêt soudain et complet des installations, onduleur compris.

Au redémarrage de la plateforme, nous constatons alors que la base de données, organe central d'OpenStack, est corrompue. Son déploiement en mode Galera complique les opérations manuelles, mais par chance, Kolla-Ansible propose une fonctionnalité de sauvegarde et restauration de base grâce à laquelle nous avons pu retrouver une plateforme fonctionnelle en quelques minutes.

5 Conclusion

L'environnement Kolla est aisé à prendre en main. Il permet de déployer un cluster OpenStack fonctionnel sans rentrer dans tous les détails techniques. Son architecture aide à la centralisation et l'historisation de l'ensemble des configurations et nous donne de bonnes bases pour entrer dans les pratiques d'*Infrastructure As Code*. Kolla a très peu de dépendances vis-à-vis du système hôte. Il est bâti autour d'outils simples et de technologies éprouvées depuis plusieurs années. Notons qu'une connaissance des briques de base Docker et Ansible est cependant indispensable pour être à l'aise.

Une infrastructure OpenStack s'agrandit et diminue au gré des besoins et des pannes (ou de la fin des contrats de garantie). Le logiciel OpenStack lui-même évolue chaque année vers une nouvelle version offrant plus de fonctionnalité et corrigeant des bogues. Cette gestion dans le temps du code et du cluster s'avère aussi importante que l'installation initiale. Avec Kolla, nous avons vu que les ajouts ou suppressions de nœuds de calcul ou des services, les mises à jour, les montées en version (le cluster est actuellement dans la version Ussuri et très bientôt Victoria), la configuration de services complexes et leur redondance sontgrandement facilitées.

Un de nos besoins que Kolla ne couvre pas pour le moment est le provisionnement de nœud sur du matériel dédié *(baremetal)*. Pour l'instant, nous utilisons des scripts maisons comme solution de contournement. Heureusement, dans cet écosystème riche en innovation, il existe le projet Kayobe[20] qui couvre ce besoin tout en utilisant Kolla en sous-main. Nous suivons son évolution et peut être que dans deux ans, qui sait, nous aurons une suite grenobloise à cette belle histoire…

20 https://docs.openstack.org/kayobe/latest/

Annexe

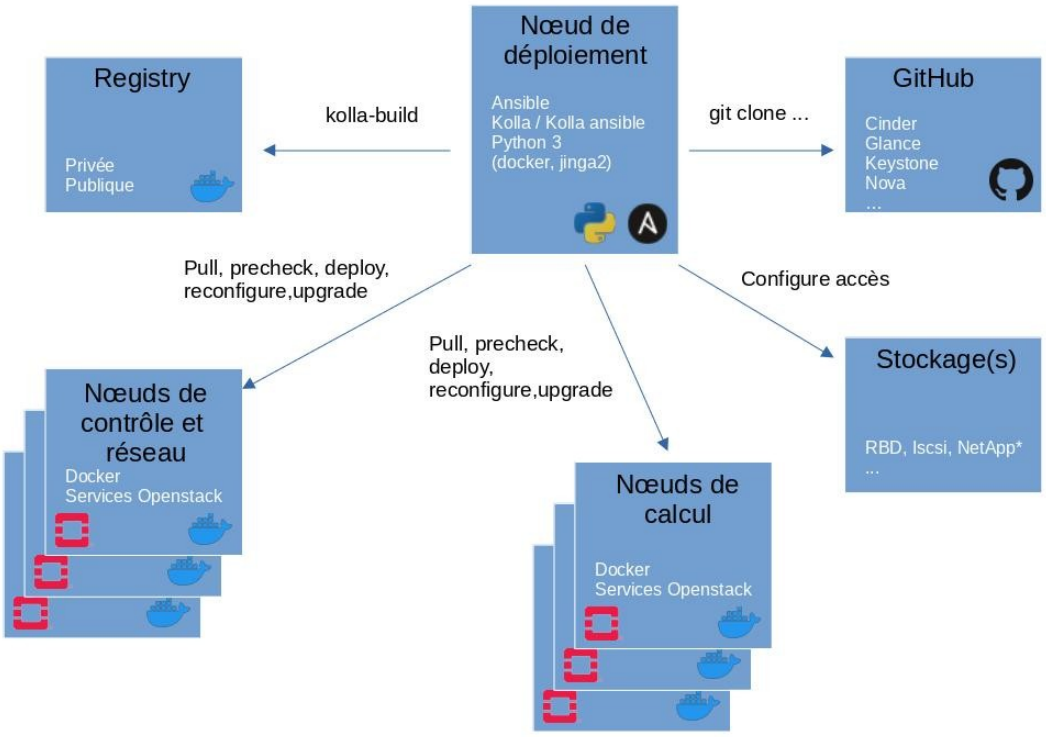

Figure 6: Architecture générale de Kolla et Kolla-Ansible

Bibliographie

[1] Nicolas Gibelin, Rémi Cailletaud, Gabriel Moreau, Jean-François Scariot, Gabrielle Feltin, Anthony Defize. Nova – Un nuage arc-en-ciel au-dessus des Alpes, JRES2019, Dijon, décembre 2019.