Microservices de traitements scientifiques sur une infrastructure de type cloud (Openstack, Kubernetes et Docker)

Mohammed Khabzaoui
Laboratoire Paul Painlevé UMR-8524
Université de Lille
Cité scientifique, 59655 Villeneuve d'Ascq.

Yvan Stroppa
Laboratoire Paul Painlevé UMR-8524
Université de Lille
Cité scientifique, 59655 Villeneuve d'Ascq

Résumé

Dans le cadre de nos développements de projets scientifiques et de valorisation de la recherche tels qu'Automathon (temps réel), ExecAndShare (calculs scientifiques) ou encore notre plate-forme des réseaux de chercheurs (big-data) et de la mise à disposition de ressources pour la communauté des Mathématiques et de l'université de Lille, nous avions besoin d'une infrastructure de type cloud qui soit capable d'héberger l'ensemble de ces applicatifs constitués de micro services et de leur offrir des services de haute performance. Parmi les services qu'offre cette infrastructure, on peut citer des services d'intégration et de déploiement continu (CI/CD), des services d'approvisionnement automatique de ressources (conteneurs, espace de travail) et service d'orchestration. Le tout intégré dans un environnement sécurisé qui garantit la continuité de service (HA, réplication et supervision) .

Les services d'orchestration sont bâtis sur Kubernetes s'appuyant sur Docker et hébergés dans un espace cloud de type Openstack. Ils pourront de la sorte s'approvisionner auprès du cloud privé qui servira de réservoir de ressources et leur permettra de s'alimenter via des mécanismes d'autoscaling.

Nos communautés sont dotées de façon très déséquilibrée de moyens informatiques centralisés. Lors de l'élaboration de projets informatiques à caractère scientifique, on se retrouve la plupart du temps avec une solution de mise en oeuvre externe à nos institutions (coûteuse et difficile à maintenir sur le long terme).
C'est pour cela que nous avons mis en place cette infrastructure pour répondre aux besoins qui émergent des projets scientifiques.

Mots-clefs

Cloud, Openstack, Kubernetes, Docker, DEVOPS et Spring boot.

1 Introduction

Nous sommes partis de nos développements de plates-formes et avons essayé de trouver une solution d'infrastructure pour les abriter dans environnement adapté pour une utilisation dédiée et permettre de les maintenir. Sachant que ces applications sont constituées de plusieurs parties et sont bâties sur des solutions de micro-services instanciés dans des conteneurs. Conteneurs qui répondent à des besoins liés à la reproductibilité, performance et continuité de service.

Déployer des applications scientifiques sur un cloud en respectant l'état de l'art n'est pas chose aisée. En effet, trouver les ressources sur lesquelles installer nos applications, développées dans nos structures, être capable de les configurer, de les sécuriser et de les maintenir devient une tâche pleine et complexe qui regroupe un ensemble de compétences complémentaires autant au niveau des technologies de développement que des technologies liées à l'opérationnel. D'autant plus, que ces applications sont constituées de plusieurs parties indépendantes et représentent ainsi un ensemble d'éléments distribués à déployer :un frontal, un intermédiaire et une partie backend. Le backend peut regrouper un ensemble de technologies de stockage de données et de traitements, comme un ensemble de machines pouvant exécuter des codes à la demande. Le tout doit être agrémenté de capacités telles que: scalabilité, performance, reproductibilité et intégrité. Chaque ensemble évolue à sa propre vitesse et doit être capable ,le plus simplement possible et avec le minimum de risque, de se mettre à jour de façon complète ou partielle.

Nous nous proposons à partir de notre expérience acquise lors du déploiement de nos trois applications de définir un modèle et les process nécessaires pour effectuer le déploiement, la mise en service et l'entretien de ce genre de plateforme. La première application nommée Automathon est une application de jeux en ligne, application multi-joueurs et temps réel. La deuxième application PNR est une application big data de traitement de données et de visualisation graphiques et statistiques d'analyse des réseaux de chercheurs. La dernière application est la plateforme ExecAndShare qui est une solution de calculs en ligne bâtie sur le concept de site compagnon.[1]

Nous allons commencer par l'inventaire fonctionnel de ces trois applications, leurs apports scientifiques et leur mode d'utilisation. On finira par la mise en perspective avec une infrastructure de type Openstack qui est la solution retenue pour répondre à ce type de besoin.

1.1 Application Automathon

L'application Automathon est une application avec des caractéristiques particulières multi-joueurs et temps réel. Ce projet développé au sein du laboratoire de mathématiques Paul Painlevé, consiste à produire des applications de jeux de type "serious games" qui permettent d'utiliser et de mettre en application des approches scientifiques. Dans la première réalisation ChickenPods, nous avons inséré plusieurs aspects mathématiques issus de travaux de recherche, de la géométrie stochastique ***k-***

1. Site compagnon: site web compagnon d'une publication, qui permet de valoriser et diffuser les codes informatiques associés à la publication. Le site permet d'expliquer la partie code produite lors de la recherche et de l'exécuter en ligne soit à partir d'un jeu de données fournis par l'auteur ou sur son propre jeu de données.

DPPs et de l'Intelligence Artificielle (**IA)** sur un concept simple: garantir la survie d'une population de poules. La poursuite de ce projet devrait aboutir à la création d'une startup permettant ainsi la valorisation des réalisations.

Le développement de la solution applicative s'est orienté sur une solution de type NodeJS et des mécanismes de web socket[2].

Plusieurs solutions ont été envisagées avant d'opter pour une solution de type conteneur. Solution qui dans ce cas de figure nous permet d'envisager chaque jeu comme un livrable indépendant et pouvant évoluer à son rythme. La difficulté suivante a été de s'occuper de la mise en service de ces jeux et de les rendre accessibles par les utilisateurs finaux. Pour cela nous nous sommes orientés vers une solution d'orchestration de type **Kubernetes** qui nous a permis d'implémenter les fonctionnalités recherchées.

Le principe de la plateforme est le suivant : à partir d'un frontal les différents utilisateurs en fonction de leur profil (public, inscrit, …) ont accès à des jeux en ligne. Chaque jeu possède des caractéristiques spécifiques (en terme de nombre de joueurs, de conditions d'accès ...). Les joueurs pour participer et jouer peuvent activer un des jeux, envoyer une invitation à un autre joueur ou jouer tout seul. Une fois l'action engagée, c'est le système d'approvisionnement qui doit prendre le relais et faire en sorte de mettre à disposition le jeu. Le système d'approvisionnement peut se comparer à un approvisionnement physique. En effet , on peut imaginer la boite de jeu sur une étagère que l'on peut répliquer à l'infini. Le système prend un exemplaire de cette boite et la met en service dans l'espace de jeu attitré (Kubernetes). Ensuite le ou les joueurs sont orientés vers leur boite respective et peuvent jouer. Une fois la partie terminée, le jeu et son accès sont détruits dans l'espace d'approvisionnement.

Ce système ainsi construit permet à partir d'un dépôt dédié de fournir les différents jeux (applications) que l'on veut mettre à disposition des utilisateurs. Il permet d'assurer l'isolation de chaque jeu et de garantir son exécution.

1.2 Application PNR :

L'objectif de cette application est de construire à partir de données publiques (publications), les réseaux des chercheurs et d'en comprendre la construction et les relations mises en place en fonction des différentes disciplines. D'essayer de différencier les modalités de construction des réseaux en fonction de critères tels que le sexe, la discipline, le ranking(du chercheur) ... . Dans les pratiques, on essaye de mettre en exergue si il existe des stratégies de rapprochement ou d'alliance entre chercheurs, si ces alliances sont durables ou éphémères et ce qu'elles peuvent apporter et produire.

L'architecture de l'application est composée de trois blocs : le premier est le frontal qui est le point d'accès à l'ensemble des services. Il est bâti sur une solution Wordpress afin d'utiliser l'ensemble des plugins pour la gestion du contexte et compte utilisateur. Il s'appuie également sur des bibliothèques Javascript (HighCharts, D3js) pour la construction des représentations graphiques associées aux différents résultats obtenus. L'ensemble des traitements se trouve au niveau du deuxième bloc qui est le web service.

2. Web socket : un canal bidirectionnel permettant : la notification au client d'un changement d'état du serveur et l'envoi de données en mode « pousser » (méthode Push) du serveur vers le client (sans que ce dernier ait à effectuer une requête)

Ce bloc garantit l'acheminement des demandes du frontal vers le backend et la récupération des résultats de ces demandes.

La troisième partie de cette plateforme est la partie traitement de données : plusieurs produits sont implémentées et sont complémentaires pour apporter toutes les fonctionnalités recherchées. Principalement des solutions de type NoSQL tournées autour de Mongodb et d'Arangodb. Pour des raisons de performances et de traitements. Afin de garder une certaine dépendance entre ces parties, nous avons utilisé les mécanismes de traitement interne (fonctions stockées) à ces environnements.

L'orientation de la solution vers une architecture distribuée pour plusieurs raisons, indépendance des couches (blocs) , évolution des couches, scalabilité et performance.

1.3 Application ExecAndShare

Cette troisième application développée au sein de LEO (Laboratoire D'Economie d'Orléans) depuis 2012 permet de mettre en ligne la notion de site compagnon associé à une publication. Elle permet au chercheur de mettre en ligne une production scientifique (un code de calcul) pour permettre de reproduire un résultat de calcul ou d'utiliser un code sur ses propres jeux de données. Pour cela, le chercheur dans une première étape doit suivre une procédure de dépôt qui l'accompagne dans la fourniture des différentes informations (méta-données) associées à son code. Le code est ensuite pris en compte par le support qui après vérification le met en ligne et le rend disponible. Pour cela, la plateforme est constituée de trois parties distinctes asynchrones. Un frontal développé en Java qui permet d'approvisionner et d'utiliser les sites compagnons, une partie intermédiaire qui permet de prendre en compte les demandes et les paramètres d'exécution (enregistrement de la demande de job) et un backend qui s'occupe de l'exécution et de la restitution des résultats. Dans sa nouvelle version une simplification de l'architecture est effectuée ainsi qu'une meilleure indépendance et isolation des traitements. Pour permettre l'isolation des traitements nous nous appuyons sur les conteneurs et un orchestrateur. Ceci nous permet d'ajouter un ensemble de fonctionnalités : de gestion de version, facilité de déploiement, appropriation.

Dépôt de brevet : https://patents.justia.com/patent/20130124686

System and a Method for Sharing Computing Resources Associated to Scientific Publications.

2 Problématique et analyse :

Les besoins générés par ces trois exemples sont :

- hébergement de structure multi-tiers,
- accès à des ressources de calculs et de traitements,
- accès à du stockage,
- scalabilité en fonction des sollicitations,
- reproductibilité,

- maintenance
- versionning,
- sécurité et de confidentialité,
- sauvegarde.

Situations précédentes :

- Pour Automathon: l'ensemble de la plateforme était hébergé au laboratoire Paul Painlevé installé sur une solution ProxMox avec un ensemble de 6 Vms.
- Pour ExecAndShare, la solution était de deux VM hébergées chez Huma-Num et d'un accès à la grille de calcul de l'In2P3 via une VM spécifique.
- Pour PNR : la solution était également hébergée sous un environnement virtualisé ProxMox de trois VMs.

Les solutions de supervision et de monitoring étaient spécifiques à chaque structure. Les possibilités d'évolution des applications étaient également complexes et spécifiques. Les solutions n'étaient pas industrialisées.

Lorsqu'on analyse les différentes plates-formes ainsi présentées, on distingue des similitudes : multi-composantes, indépendance entre les parties, des responsables différents, un cycle de vie différent, des besoins de contrôle. Donc l'objectif de l'infrastructure est d'industrialiser les déploiements, permettre l'utilisation des services et leur évolution.

3 Solution proposée :

Après la mise en place au sein de l'université de Lille de la solution Openstack dans le cadre de l'enseignement depuis 2012 (à disposition des étudiants pour créer des VM) et d'un perfectionnement au niveau du laboratoire de Paul Painlevé UMR8524 (à l'aide d'un apprenti pour la mise en place d'une solution HA autour d'Openstack), il nous est apparu intéressant de construire autour de cette solution une plateforme nommée MKYS permettant d'abriter nos applications complexes.

La structure retenue pour MKYS peut se comparer à une un serveur à lames, c'est à dire un gestionnaire de lames où chaque lame est contient l'ensemble des ressources nécessaires au fonctionnement de la plateforme applicative. Les ressources se composent de serveurs, d'éléments réseaux de niveau L2 et L3, de ressources de stockage et autres éléments de sécurité qui permettent la bonne exécution des différentes parties qui la composent.

La plateforme MKYS offre les services suivants:

- Définition et construction d'une lame
- Approvisionnement de la lame sur l'infrastructure
- Utilisation des ressources mises à disposition au sein des lames
- Maintenance de la lame et de ces éléments.

Pour définir une lame, il faut être autorisé et avoir les droits spécifiques. Pour cela, nous nous appuyons sur la notion de projet d'Openstack pour la définition des utilisateurs, rôles et la gestion des quotas associés. Cette notion de projet regroupe tous les éléments d'autorisation et de gestion des droits sur les éléments (VM, réseaux ...) appartenant à un ou plusieurs projets.

Ensuite pour la construction de la lame, chaque administrateur de la lame définit les caractéristiques de celle-ci dans une limite de ressources attribuées par le système de quota d'Openstack. Cette définition de lame consiste à caractériser chaque partie en terme de ressources et à définir leur configuration.

Cette définition se fera à partir d'une interface Web simplifiée de type Magnum qui permet aux utilisateurs de définir et de caractériser ses besoins. Une fois l'opération effectuée, le système génére un template en yaml qui contiendra l'ensemble des éléments ainsi définis. L'utilisateur peut choisir parmi un ensemble de modèles celui qui lui convient le mieux et l'adapter à ses besoins. Une bibliothèque de templates est accessible et versionnée dans un dépôt dédié.

Pour garantir et offrir la possibilité aux utilisateurs autorisés de gérer le cycle de vie de leur lame, une lame nommée Devops a également été installée sur MKYS. Elle est transversale à toutes les autres lames et contient l'ensemble des services attendus pour des opérations de maintenance. On peut citer : les services de gestion de versions, des services CI/CD et des services de monitoring.

Chaque lame ainsi installée, sera connectée (interconnexion de réseaux) avec la lame Devops qui peut se comparer à une carte de fond de panier pour continuer dans l'analogie avec le blade. Et permettra les remontées de logs et les actions de mise à jour au sein de chaque lame.

Pour ce faire, les scripts d'installation construits à partir de la description yaml prennent en compte l'installation de tous les dispositifs de surveillance (sondes) à la construction des éléments d'interconnexion nécessaires entre les éléments réseaux pour permettre les échanges.

Pour permettre cette mise en place, nous avons installé sous Openstack les services de base tels que Keystone, Glance, Neutron, Nova, Cinder et les services complémentaires tels que Heat, Gnocchi, Ceilometer, Magnum, Designate pour la partie déploiement.

Chaque brique de base garantit un ensemble fonctionnel permettant de remplir les objectifs de la plateforme MKYS.

L'accès à la plateforme et aux services exposés, se fera selon trois modes différents :

- Mode utilisateur : Les utilisateurs des services offerts par les lames, accèdent à partir d'un portail dédié à la plateforme MKYS et sont redirigés vers le service recherché.
- Mode devops : Cet accès permet aux utilisateurs autorisés d'effectuer les mises à jour de leur lame, son déploiement et l'accès au monitoring pour avoir une vision de l'utilisation de leur lame.

- Mode administrateur : Une entrée Administrateurs via Horizon pour l'accès aux différents éléments qui composent les lames.

Voir le schéma global : Figure1

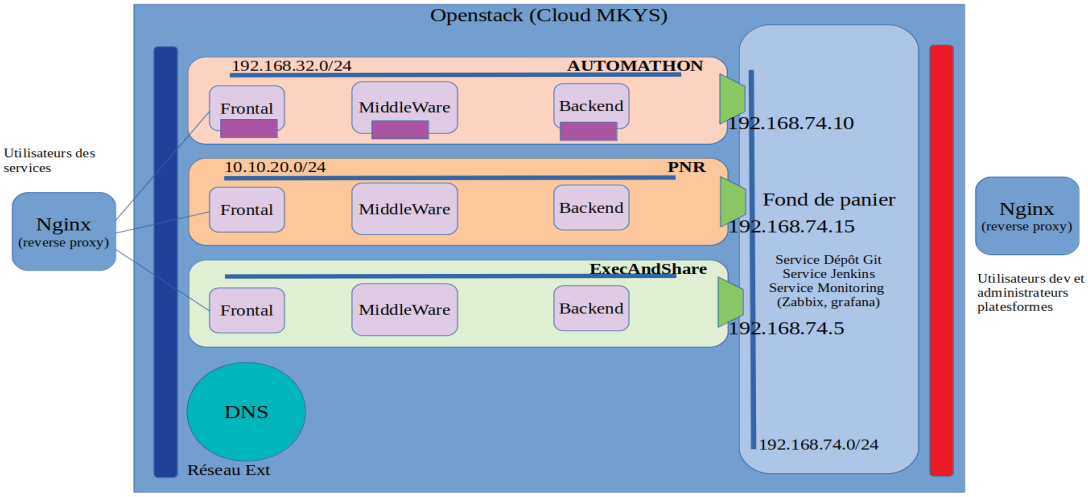

Figure 1 - OpenStack

4 Détails de la structure de MKYS

4.1 L'accès à la plateforme :

Un portail unique permet d'accéder aux différentes applications et fonctionnalités offertes par la plateforme via deux reverses proxies dont un avec une configuration "dynamique". La configuration dynamique nécessaire surtout pour Automathon permet d'adapter (créer ou détruire) en fonction des parties à jouer les routes vers les jeux déployés. Les modifications sont effectuées à l'aide d'un web service installé sur le reverse proxy et qui reçoit de l'application les notifications d'aménagement des règles de routage (ajout ou suppression de règle et rechargement de la configuration). La solution NGINX permet de répondre parfaitement à cette situation c'est à dire un "routage dynamique" sans altération des connexions existantes.

Sous Openstack, deux réseaux externes (au sens Openstack) sont utilisés pour permettre de différencier les accès des utilisateurs (Users ou Devops). Chaque réseau est configuré avec un pool d'ip flottantes permettant l'accès aux frontaux de chaque plateforme du côté utilisateurs comme du côté Devops.

La lame Devops contient l'ensemble des services de mise à jour, de déploiement, de monitoring et de supervision adossée aux contrôles d'accès de Keystone. Cette lame permet au groupe Devops d'accéder à l'ensemble de ces services et d'intervenir sur leur lame.

La lame composée de différentes parties subit lors de l'intégration une configuration particulière pour permettre son interconnexion avec la lame devops.

4.2 Comment déployer les plates-formes sous Openstack:

Le déploiement de chaque plateforme se fait de manière manuel en mode administrateur ou à l'aide de Heat via Magnum. Le tout repose sur une base d'images/snapshots adaptée et évolutive en fonction des besoins issue de la brique fonctionnelle d'Openstack qui est Glance.

Des templates Heat permettent soit en ligne de commandes soit à partir de l'interface de construire et de déployer directement des ensembles cohérents en s'appuyant sur les primitives d'Openstack en terme de ressources. Ces éléments s'appuient sur des fichiers de type yaml.

Le déploiement de l'ensemble s'accompagne d'un déploiement complémentaire pour permettre les interconnexions avec la lame de fond de panier. De plus des éléments de supervision sont installés en même temps et configurés pour être en mesure de remonter les informations à la console de supervision dans la partie Devops. Le tout pourra reposer sur les services de Gnocchi et de Ceilometer pour permettre la scalabilité des solutions mises en place en fonction de la charge.

4.2.1 Comment y accéder:

Dans les paramètres d'installation, il faut préciser quels sont les éléments accessibles et les ports ouverts pour récupérer une ou plusieurs adresses flottantes et pour positionner les règles de sécurité d'accès dans Openstack. Ensuite une demande est émise auprès des administrateurs de la plateforme MKYS qui autoriseront ou pas cette demande. Le fait d'autoriser permettra de définir sur les frontaux d'accès les régles adéquates et de compléter le portail de la plateforme. Quand toutes ces opérations sont effectuées, les utilisateurs finaux accédent aux services.

Le retrait d'un service se fait de la même façon, on retire la redirection à l'aide d'une demande auprès des administrateurs de MKYS. La route pourra être retirée de façon partielle en fonction de la demande. Par exemple on pourra conserver les accès à la partie Devops et bloquer les accès utilisateurs.

4.2.2 Comment les entretenir et les surveiller:

Cette partie se fera à l'aide de la lame devops de la plateforme, lame qui est entretenue par les administrateurs de MKYS et permet aux devops d'accéder aux fonctionnalités qui leur incombent. Plusieurs services sont disponibles dans cette partie : un dépôt Git, un service Jenkins, un système de monitoring de type Zabbix, plus des accès dans le cadre de Kubernetes aux dashboards installés. On peut accéder à l'aide de Grafana aux métriques collectées par les sondes Prometheus installées dans les orchestrateurs.

Chaque infrastructure ainsi hébergée doit pouvoir se mettre à jour et on doit pouvoir garantir un retour si besoin à la version N-1. A partir de cette partie on doit être en mesure d'agir sur tout ou partie de la lame de façon réversible si possible.

4.3 Détail des lames:

4.3.1 Détail de la configuration de la lame Automathon

Projet : automathon , groupe d'utilisateurs: automathon-admin ; rôles: admin

La plateforme Automathon est composée de trois parties, une partie frontale à laquelle ont accès les utilisateurs. cette partie frontale est composée de deux sous-parties : un serveur web avec Wordpress et une base de données Mysql.

Elle s'appuie sur un micro-service (web-service) de type API-REST qui sert de lien avec la partie backend. C'est cette partie qui détient la connaissance de l'ensemble du backend et qui effectue les différentes demandes d'approvisionnement.

Un dépôt de conteneurs lui est associé, et permet de stocker et de mettre à disposition les conteneurs de jeux.

Ensuite le tout repose sur un orchestrateur de type Kubernetes qui garantit l'approvisionnement de conteneurs et des mécanismes de persistance et de routage associé.

4.3.2 Détail de la configuration de la lame PNR

Projet : PNR, groupe d'utilisateurs: pnr-admin, rôles: admin

Pour le frontal : application sous Wordpress avec les plugins adaptés aux fonctionnalités recherchées. Gestion de compte et librairie de visualisation de graphes sous HighCharts.js et d3.js.

Le web service est une partie développée sous Spring Boot pour disposer d'un framework de haut niveau dans l'exploitation de cette brique. Autant au niveau performance que de sécurité.

Pour la persistance on a opté pour deux bases de données complémentaires Mongodb et Arangodb qui sont installées dans un espace Kubernetes pour offrir les performances attendues.

4.3.3 Détail de la configuration de la lame ExecAndShare

La plateforme ExecAndShare est bâtie sur le même modèle que la première application Automathon car le besoin est similaire. Pour des raisons de temps et d'historique le frontal est conservé sous Java et communique avec un web service intermédiaire qui exécutera l'approvisionnement des conteneurs de calculs qui pourront dans le meilleur cas être appariés à de la GPU s'ils le nécessitent. Cette solution sur ce type d'architecture va permettre d'apporter toutes les conditions d'exécution optimales nécessaires pour cette application. A la fin de chaque calcul, les conteneurs seront détruits ce qui évitera des effets de bords éventuels.

4.3.4 Lame Devops

Détail de la lame Devops:

- Dépôt GIT : Un serveur Git permettant aux différents responsables de Lame d'entretenir les versions de leurs différentes parties de plateforme.
- CI/CD: Jenkins ou Gitlab. Lors de la mise à jour des différents éléments et de leur validation, permet de déposer sur les lames les mises à jour soit directement dans les espaces dédiés soit en reconstruisant l'image Docker correspondante et la déposer dans le dépôt attitré.
- Supervision/Monitoring: Outil de collecte des métriques définis pas l'utilisateur lors du paramétrage de sa lame. Ces métriques sont remontés par les agents déployés sur les lames et concentrés dans un ou plusieurs dashboards permettant de visualiser les activités des différents éléments installés que ce soit serveur, base de données ou conteneurs.

A la différence des autres lames, celle-ci est gérée par les administrateurs de MKYS et utilisée par les administrateurs des lames pour l'accès aux services fournis.

5 Conclusion

L'intégration de ces différentes plates-formes dans le cloud d'Openstack nous a permis de nous confronter à un ensemble de difficultés et de définir des pistes de solutions. Nous pensions naïvement que les solutions internes d'Openstack de type Magnum allaient résoudre les difficultés de déploiement d'ensemble complexe sur tous types de systèmes. Ceci s'est avéré limité à des solutions sous des systèmes de type Fedora-atomic ou coreos. De plus, l'utilisation de ces systèmes dans cette infrastructure n'est pas chose aisée, car il faut appréhender l'ensemble des briques et des concepts informatiques les plus larges. L'inconvénient de ce type de solution est sa complexité qui est masquée par les procédures d'installations fournies par Openstack et sa communauté. En effet, les procédures d'installation sont limpides , et on peut bénéficier également de package d'installation Ansible ou Puppets. Effectuer l'installation en suivant un guide détaillé n'est pas chose complexe par contre lors d'une anomalie ou d'un dysfonctionnement on se retrouve bien seul et face à une immensité technique adossée à une multitude de forums divers et variés, qui finissent pas vous entraîner dans des chemins ou il devient difficile de ressortir et d'avoir sa réponse, étant donné la multitude des configurations et versions existantes. Ce qui est valide également en mode production où la nécessité de comprendre l'ensemble du Cloud est essentiel pour surmonter les difficultés et permettre une utilisation optimale.

Malgré tout, à force de volonté et avec une assise technique de haut niveau que l'on se doit d'acquérir pour mener à bien ce type de projet on arrive à construire des systèmes stables qui profitent pleinement des fonctionnalités offertes par Openstack.