Université Cadi AYYAD

Ecole Nationale des Sciences Appliquées de Safi

Département Génie Informatique, Réseaux et Télécoms

Rapport d'avancement

Orchestration des nœuds Jenkins avec Kubernetes

Réalisé par

Rakhim Chaimae

Encadré par

M. Ed Dafali Ayoub

M. Bouarifi Walid

Classe

5ème Année Génie Informatique

Année Universitaire : 2018 - 2019

Appréciation et signature de l'encadrant

Table des figures

Table des matières

Introduction générale

La transformation digitale est la réponse de chaque entreprise au défi que représente l'essor des technologies numériques. Confrontment cela suppose des évolutions centrées sur l'innovation, l'adaptation et l'agilité, le cloud, le big data, la mobilité, l'automatisation …

Le digital (le numérique) et donc l'informatique, l'IT sont désormais les principaux vecteurs de création de valeur pour une entreprise.

Les organisations se doivent d'être en capacité à mieux collaborer et à mieux s'adapter aux changements et ce dans un cycle d'innovation continue.

Qu'elles soient des entreprises du Net ou non.

Toute entreprise doit être capable de détecter et répondre à un besoin, de changer rapidement et en toute confiance.

Elle doit être capable d'optimiser sa vitesse de livraison d'un produit ou service.

Toutes les étapes de l'identification d'une opportunité à la réalisation des premiers bénéfices doivent être optimisées.

Toute entreprise doit réduire le temps de mise à disposition (lead time) de ses produits ou services.

Chapitre 1 : Contexte générale du projet

Présentation de l'organisme d'accueil

Le groupe SQLI

Créé en 1990, SQLI est le partenaire de référence des entreprises et des marques dans la transformation digitale de leur parcours client et de tous les services internes impactés par cette évolution. Son positionnement unique au confluent du marketing et de la technologie lui permet de répondre de façon globale aux enjeux de développement des ventes et de notoriété (marketing digital & social, expérience client, commerce connecté, data intelligence...) ainsi qu'aux enjeux de productivité et d'efficacité interne (digitalisation des opérations, entreprise collaborative, mobilité et objets connectés, CRM...). Ses 2400 collaborateurs sont répartis en France (Paris, Lyon, Toulouse, Bordeaux, Rouen, Nantes et Lille), en Suisse (Lausanne et Genève), au Luxembourg, en Belgique (Bruxelles et Gand), au Royaume-Uni (Londres), au Maroc (Rabat et Oujda) et en Afrique du Sud (Cape Town).

Figure 1: Les différentes agences de SQLI dans le monde

DevOps chez SQLI

Notre projet de fin d'études s'intègre dans la démarche DevOps chez SQLI. Pour cela, dans cette partie nous allons expliquer l'offre DevOps, pourquoi et comment démarrer une démarche DevOps.

Pourquoi DevOps ?

DevOps est un ensemble de bonnes pratiques pour l'industrialisation du système d'information, plus une stratégie pour réduire le Time To Market. Les méthodes agiles sont des formes efficaces de développement et gestion de projet qui offrent plusieurs avantages :

- Une haute qualité du produit ou service

- La prise en compte des besoins Métier, utilisateurs de manière continue

- Des livraisons régulières des nouvelles fonctionnalités (développements réguliers).

Mais, il reste encore des points d'amélioration afin d'étendre les principes agiles jusqu'aux équipes d'exploitation.

- Les mises en production (donc mises à disposition du produit aux utilisateurs) dépendent de plans de déploiement, de releases planifiées tous les mois, 3 mois, 6 mois, etc.

- Une fois en production, le produit n'est que peu étudié afin de s'assurer qu'il correspond réellement aux attentes des utilisateurs finaux. On note une faible réactivité face aux retours.

Pour ces raisons et d'autres, nous sommes confrontés à un besoin d'étendre les principes agiles sur toute la chaîne de création d'un produit ou service informatique. Ce qui permettra de :

- Réduire le délai de mise en production, de mise en ligne, de mise sur le marché du produit

- Pouvoir livrer et démontrer à tout moment (livraison continue) l'état des produits

- Rapprocher les métiers du développement et des opérations (exploitation) pour une meilleure performance

- Être plus efficient tout au long du cycle de vie du produit

- Gérer efficacement et de façon plus simple les environnements via l'automatisation.

En vertu de ce qui précède, une comparaison des méthodes agile (sans DevOps) et méthode agile avec DevOps se résume dans la possibilité de livrer n'importe quel moment tant que les tests sont valides, au lieu de la fin de chaque sprint .

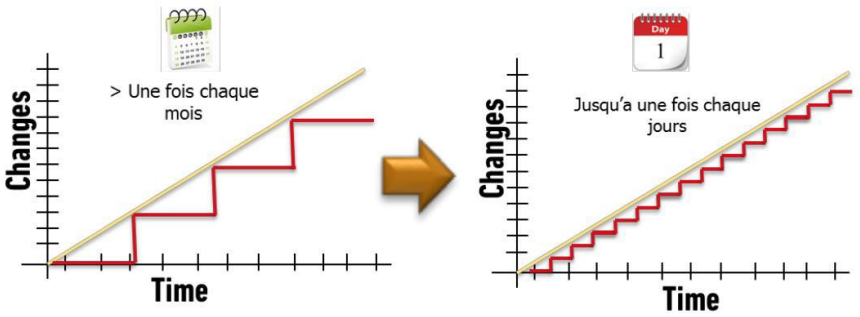


Figure 2: La possibilité de livrer plus fréquemment avec DevOps

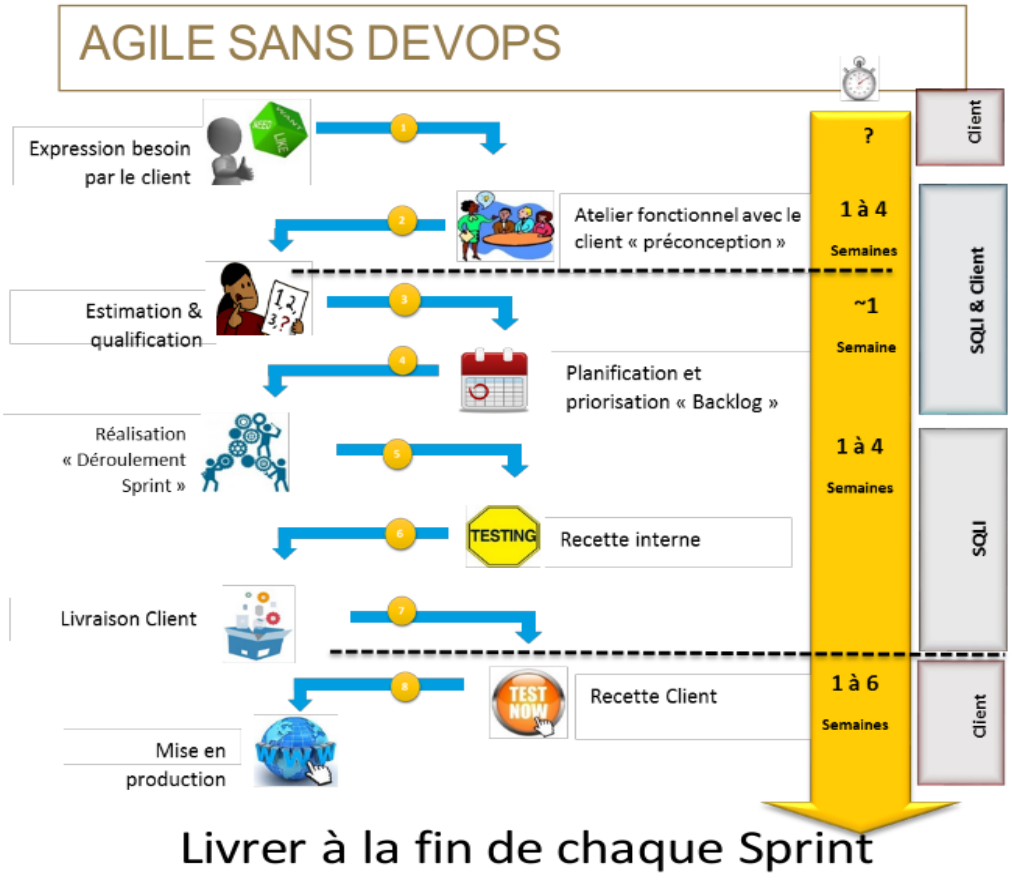


Figure 3: Agile sans Devops

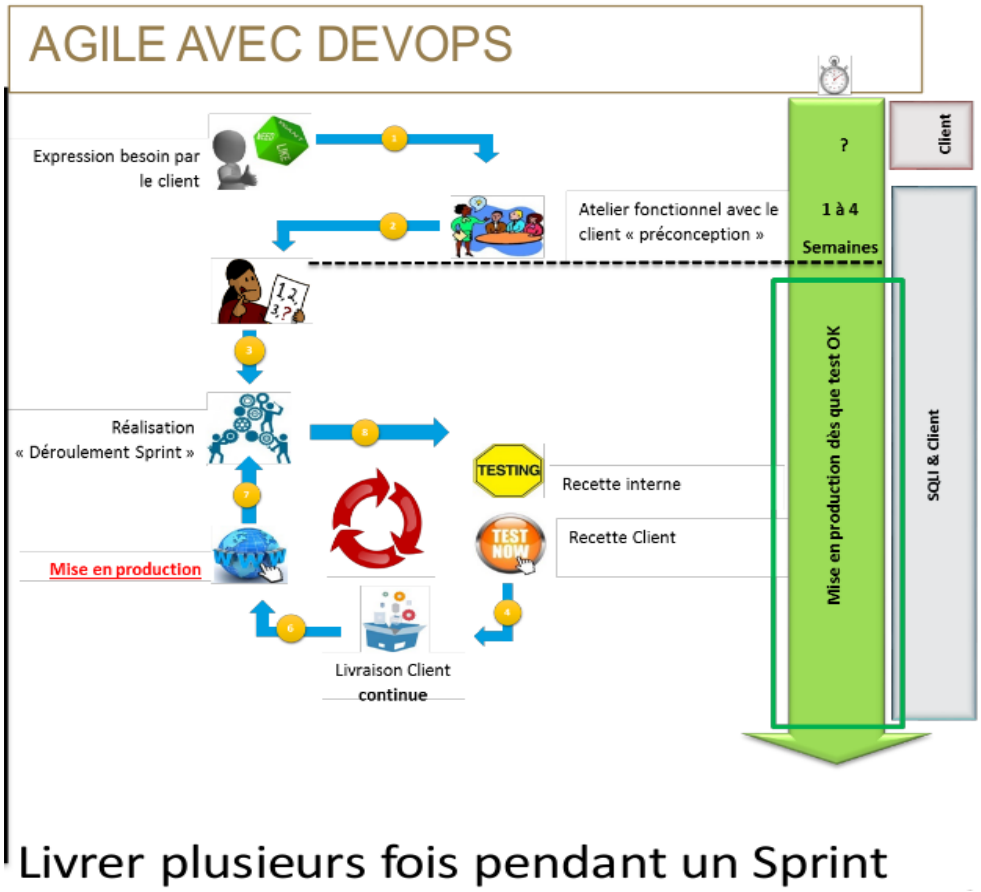


Figure 4: Agile avec Devops

Étude de l'existant et constations :

Dans cette partie, nous présentons tout d'abord la procédure que SQLI ISC MAROC suivait initialement pour le déploiement ses applications, ensuite nous exposons la problématique. Pour gérer le cycle de vie de ses applications avant leur déploiement, SQLI disposait d'une chaîne d'intégration continue basé sur l'outil de gestion de versions "GIT", un serveur d'intégration continue "Jenkins", un gestionnaire de dépôt maven "Nexus sonatype" et enfin pour la mesure de qualité du code source "SonarQube". Une fois les artéfacts de l'application déployée sur Nexus, Ils sont téléchargés pour qu'ils doivent installe dans les serveurs du client. Ou parfois il y a procédures des déploiements continues.

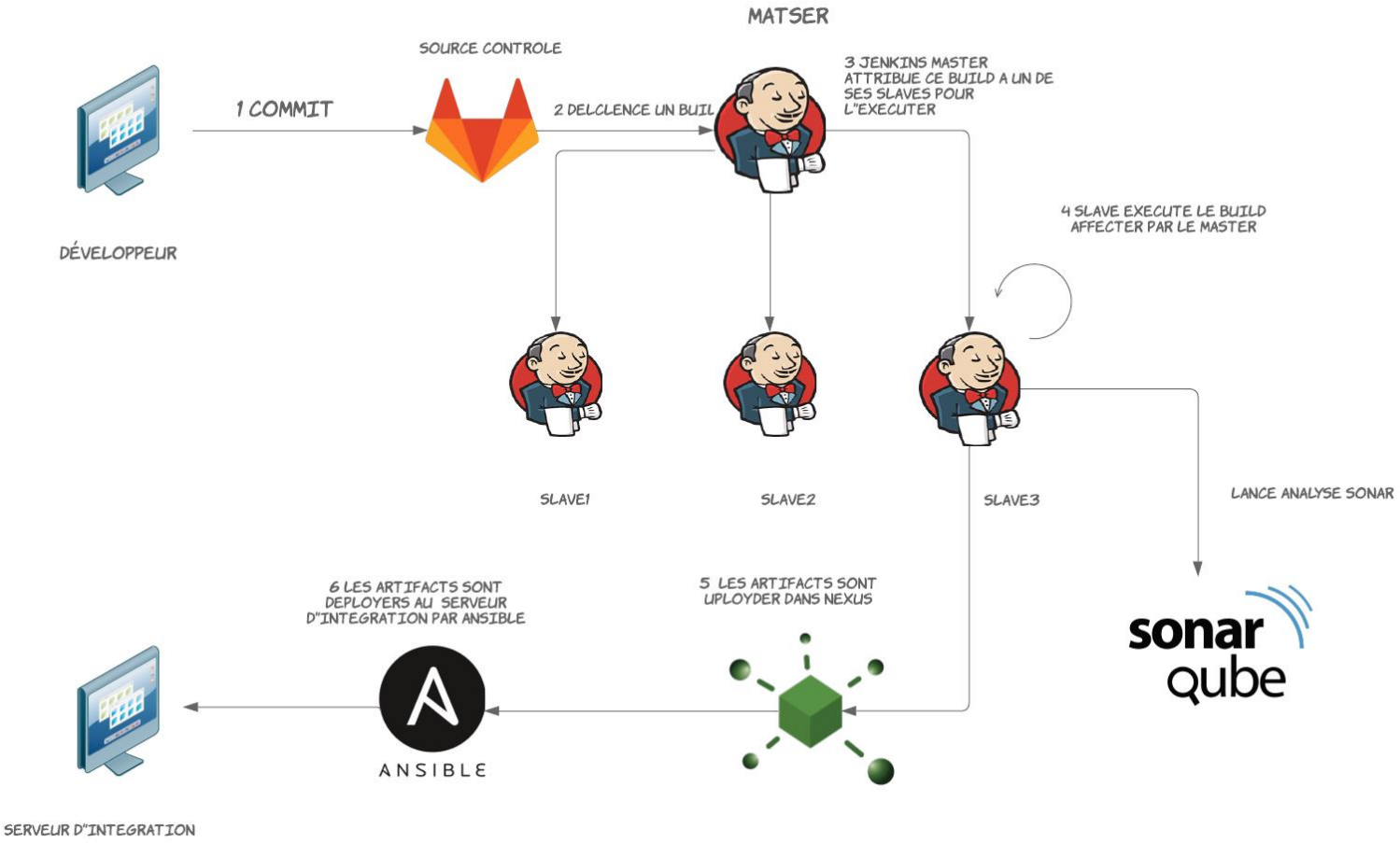


Figure 5: la chaîne d'intégration continue chez SQLI ISC MAROC

Cette chaine est construite de façon automatise dans des fichiers de configuration dans Jenkins

Jenkins master attribuer les Jobs sur ses slaves selon le cas chaque Job est tiqueter pour qu'elle soit exécuter dans un slave spécifique, ces slave sont des machines virtuelles.

Pour les projets lourde comme le cas SAP hybris la machine nécessite au mois 8 GB de RAM et 4 ou 6 GPU . Ce que présente un énorme problème au niveau des allocations des ressources chez SQLI MAROC.

La solution ?

Pour remédier à ce problème on a pensé à utiliser la containerisation pour optimiser les ressources. Les jobs doivent s'exécuter dans des conteneurs et non pas des machines virtuelles.

Diagramme de Gantt :

Le travailler effectuer dans ce stage est deviser en plusieurs stage de formation et intégration plus le monte de compétence en tant qu'un ingénieur Devops.

Name	Begin date	End date
Semaine d'acceuil et presentation d;entreprise	04/02/19	08/02/19
Formation initiale	11/02/19	25/02/19
Introduction Devops	11/02/19	11/02/19
Git	12/02/19	12/02/19
Jira/GitlabCI	13/02/19	13/02/19
Maven	14/02/19	14/02/19
Jenkins	15/02/19	15/02/19
Sonar	18/02/19	18/02/19
Shell Scripting	18/02/19	18/02/19
Ansible	19/02/19	19/02/19
Nexus	19/02/19	19/02/19
Monitoring	20/02/19	20/02/19
Ngnix/Apache2	20/02/19	20/02/19
Hybris	21/02/19	25/02/19
Reunion des sujets des PFE	04/03/19	04/03/19
Affectation des sujets de PFE	06/03/19	06/03/19
Documentation sur Kubernates	07/03/19	27/03/19
Creation et configuration de Kuberntes Cluster	28/03/19	10/04/19
Integration de K8s avec Jenkins	11/04/19	19/04/19
Formation de monte de competence	15/04/19	19/04/19
Creation et integration du pipeline dans Jenkins Interne SQLI pour le projet	18/04/19	25/04/19

Figure 6: La liste des taches effectue jusqu'à présent

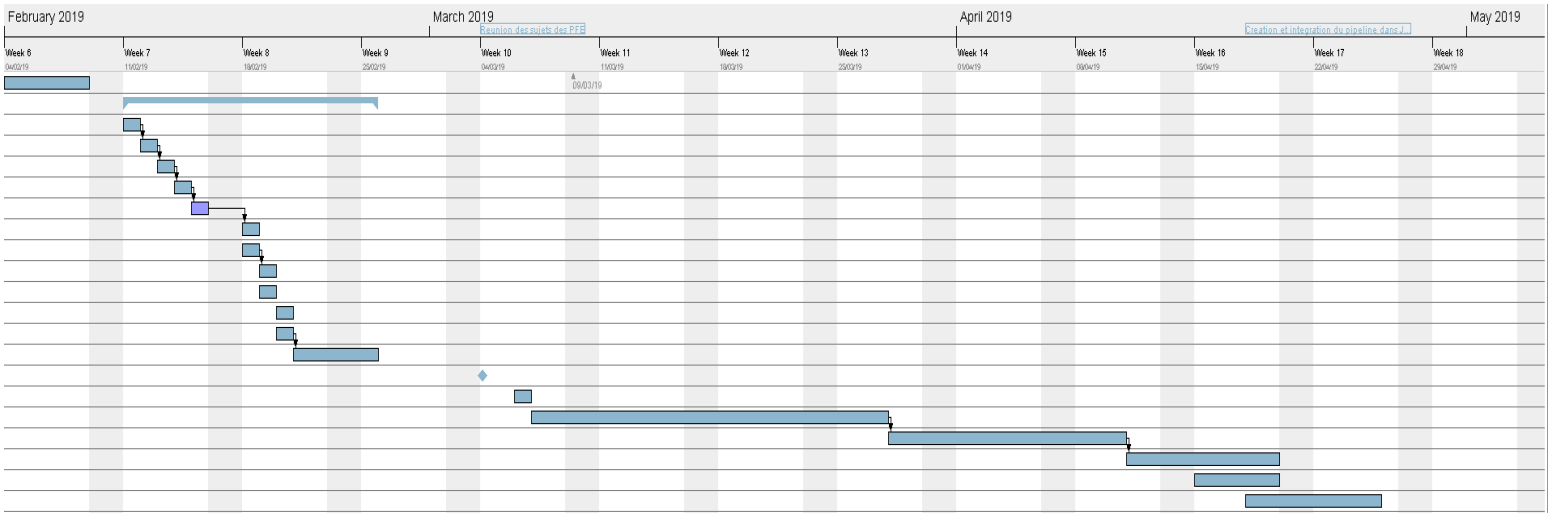


Figure 7: Diagramme de gantt

Chapitre 2 : Etude préalable

DevOps est un ensemble de pratiques qui automatisent les processus entre les équipes de développement et IT afin de leur permettre de développer, tester et livrer des logiciels plus rapidement et avec plus de fiabilité. Le concept de DevOps repose sur la mise en place d'une culture de la collaboration entre les équipes qui étaient, historiquement, cloisonnées. Parmi les avantages assurés, le gain de confiance, l'accélération des livraisons, la capacité à résoudre les tickets plus rapidement ou encore la gestion plus efficace des tâches non planifiées.

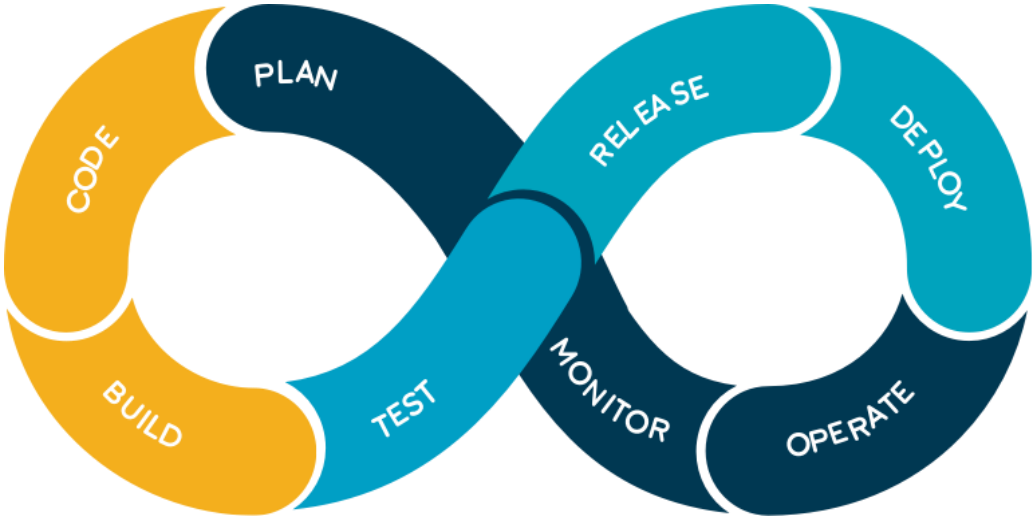


Figure 8: les différentes Processus Devops

Les rôles dans Devops :

Dans cette partie on va expliquer les rôles de chaque participant dans devops :

Le rôle des Devs : Par Devs, on désigne toute personne impliquée dans la conception, le développement et les tests d'un logiciel avant qu'il n'entre entre production : les développeurs, les gestionnaires de produits, les testeurs, les **Product Owners** et les **QAs.**

Rôle opérationnel : Il s'agit de toute personne intervenant dans l'exploitation et la maintenance de la production : les ingénieurs systèmes, les DBAs, les ingénieurs réseaux, le personnel de sécurité, etc.

Les Développeurs (Dev) sont donc chargés de produire de l'innovation et délivrer les nouvelles fonctionnalités aux utilisateurs dans les meilleurs délais. Les Ingénieurs d'opérations (Ops) sont chargés de s'assurer que les utilisateurs ont accès à un système stable, rapide et réactif.

Bien que le but ultime de Dev et Ops soit de rendre l'utilisateur satisfait des systèmes qu'ils fournissent, leurs visions sur la façon d'y arriver restaient diamétralement opposées.

Notion de base :

Les termes Continuous Integration (CI), Continuous Delivery (CD), Continuous Deploiment (CD) et respectivement en français Intégration continue, Livraison continue, Déploiement continu et sont très présents depuis quelques années. Dans la méthodologie AGILE, ces trois concepts s'intègrent de manière quasi naturelle et bien souvent les équipes construisent leurs logiciels en suivant ces principes sans en avoir pleinement conscience. Le but est de permettre l'évolution de projets de manière fluide entre la production de code, et le déploiement final. La majeure partie des projets est généralement soumise à la vue de différentes équipes :

- les développeurs, techniciens intégrateurs de solutions
- les équipes fonctionnelles qui délivrent des spécifications d'intégration et qui, souvent, sont aussi testeurs de nouvelles fonctionnalités
- les opérateurs, administrateurs système, qui vont mettre en production les versions finales

Ces 3 équipes sont disséminées dans l'entreprise, parfois géographiquement, et il n'est pas rare que la communication entre elles ne soit ni claire ni efficace. L'une des principales raisons de cette fracture est l'absence d'outils pour leur permettre de communiquer efficacement.

Intégration Continue

L'Intégration Continue (continuous integration) consiste à intégrer les changements apportés au code informatique d'un projet logiciel de façon continuelle, afin de détecter et de corriger immédiatement les éventuelles erreurs. Découvrez la définition précise et les avantages de cette pratique.

L'Intégration Continue est en quelque sorte le socle du Déploiement Continu. Pour résumer le principe, chaque modification poussée par un développeur dans le service de gestion de sources va être automatiquement testés pour s'intégrer dans le tronc de développement 1 - ce qui induit qu'à tout moment l'application est considérée comme étant potentiellement livrable.

Livraison Continue

Continuous Delivery est une orientation du développement logiciel consistant à livrer régulièrement et de manière automatisée. Il est possible de choisir de livrer les changements de façon quotidienne, hebdomadaire, ou autre fréquence en fonction des besoins propres à l'entreprise et à sa clientèle.

Déploiement Continue

Le déploiement continu va encore plus loin, et consiste à livrer chaque changement apporté au logiciel à la clientèle. Dans ce cas de figure, il n'y a pas d'intervention humaine. Les seuls changements qui ne sont pas déployés sont ceux qui échouent à un test. Cette pratique permet

d'accélérer la boucle de feedback, et permet aussi aux développeurs de mieux se focaliser sur le développement du logiciel puisqu'il n'y a plus de « date de relaxe » à anticiper.

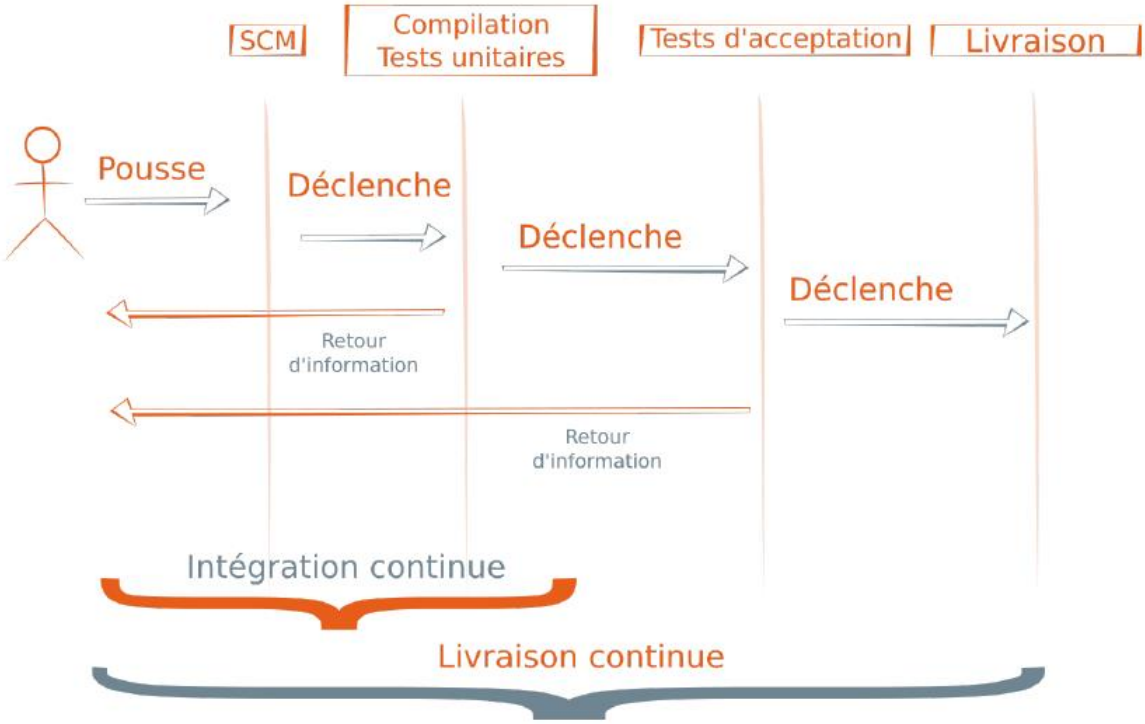


Figure 9: Séquence de livraison continue

Intégration continue avec Jenkins :

Jenkins s'appelait à l'origine Hudson. C'est un outil d'Intégration Continue open source basée sur la technologie Java. Avant tout, il est gratuit et multiplateforme d'où on peut l'utiliser sur n'importe quelle plateforme sans avoir de doute sur le dysfonctionnement de quelques parties ou quelques fonctionnalités...

Le deuxième grand avantage est sa facilité d'utilisation. C'est alors tout à fait logique s'il bénéficie de la majorité du marché. Plus de 1000 plugins existant déjà pour faciliter l'intégration avec quasiment tous les outils de développement connus. Jenkins offre donc une forte interconnexion avec plusieurs outils et environnements.

Figure 10: Logo de Jenkins

Le fonctionnement du Jenkins :

Lorsqu'on insère un bout de code le répertoire du code source du projet Jenkins capte automatiquement tous les changements sur le répertoire lors de sa vérification. Et de cette manière il détecte tout événement de modification. Ainsi, on est notifié en cas de changement. En suite Jenkins prépare pour lancer un nouveau build du projet. Pour assurer l'automatisation de ce processus il faut obligatoirement télécharger un fichier de type .war et le lancer.

Il existe différentes façons de déploiement Jenkins.

- La première s'effectue à travers un serveur d'application web. Cette option est vraiment très rapide à mettre en place, par exemple avec Tomcat ou autre type de serveur.
- La deuxième forme c'est le Standalone. Il s'agit d'une ligne de commande simple qui permet d'exécuter le fichier war de Jenkins. Dans ce cas on n'est pas obligé d'utiliser le serveur d'application web.
- La troisième forme Slave, une stratégie Stratégies Maître/Esclave. Elle permet d'effectuer la distribution. Par conséquent une réduction en utilisation du serveur, C'est faisable qu'à partir d'un Jenkins installé en mode master.

Stratégie master/slave dans Jenkins :

Le rôle du maître

La machine maîtresse est typiquement la machine sur laquelle est installée Jenkins mais dans le cas d'une architecture distribuée elle servira principalement à définir la typologie de la grappe d'esclaves à contrôler. C'est aussi à ce niveau que le workflow de la campagne de test sera défini notamment par le biais de tâches plus ou moins complexes appelées jobs.

Enfin cette machine jouera le rôle d'agrégateur en regroupant les résultats des différents tests ainsi que les rapports d'utilisation (charge) de la grappe d'esclaves.

Le rôle des esclaves

Les esclaves ont pour rôle principal la réalisation des jobs dispatchés par la machine maîtresse. Faire d'une machine un esclave pour Jenkins requiert uniquement de configurer un jar client (le slave agent) au niveau de la machine et de s'assurer que la communication est possible avec la machine désignée comme maître.

Le postulat de base pour les machines esclaves est que ces dernières ne sont pas fiables dans le temps et qu'il faut être capable de substituer une machine par une autre à tout moment. Pour permettre cette substitution, Kohsuke commença donc sa présentation avec un plugin de base de Jenkins à savoir l'outil de déploiement et d'installation automatique.

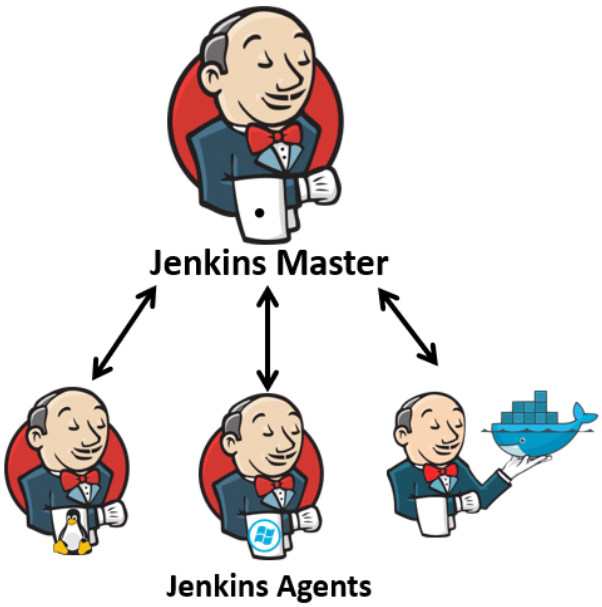


Figure 11: Architecture Master/Slaves Jenkins

Les conteneurs et les machines virtuelles

Qu'est-ce qu'un conteneur ?

Un conteneur est une enveloppe virtuelle qui permet de packager une application avec tous les éléments dont elle a besoin pour fonctionner : fichiers source, runtime, librairies, outils et fichiers. Ils sont packagés en un ensemble cohérent et prêt à être déployé sur un serveur et son OS.

Contrairement à la virtualisation de serveurs et à une machine virtuelle, le conteneur n'intègre pas d'OS, il s'appuie directement sur le système d'exploitation du serveur sur lequel il est déployé.

Qu'est-ce que docker ?

Docker est un projet Open Source proposant une surcouche qui automatise et simplifie le déploiement d'applications dans des conteneurs virtuels.

A l'origine basée sur le format de Conteneurs Linux LXC, la société Docker a élargi le projet en proposant notamment une API qui permet d'exécuter des conteurs standards et portables d'un serveur Linux à l'autre.

La virtualisation de serveurs permet, au travers d'une couche logicielle appelée Hyperviseur, de créer sur un serveur physique des machines virtuelles qui utiliseront les ressources physiques du serveur avec leur propre OS. L'Hyperviseur assure l'allocation des ressources physiques entre les VMs, et chacune d'elle se comporte comme un serveur autonome intégrant son propre OS sur lequel les applications seront déployées et exécutées.

Contrairement à la virtualisation, un conteneur n'embarque pas d'OS, il s'appuie directement sur celui du serveur sur lequel il est déployé. Un conteneur est donc beaucoup plus léger qu'une

VM qui intégrera un OS. Une VM se mesure en Go quand le volume d'un conteneur se chiffrera en Mo.

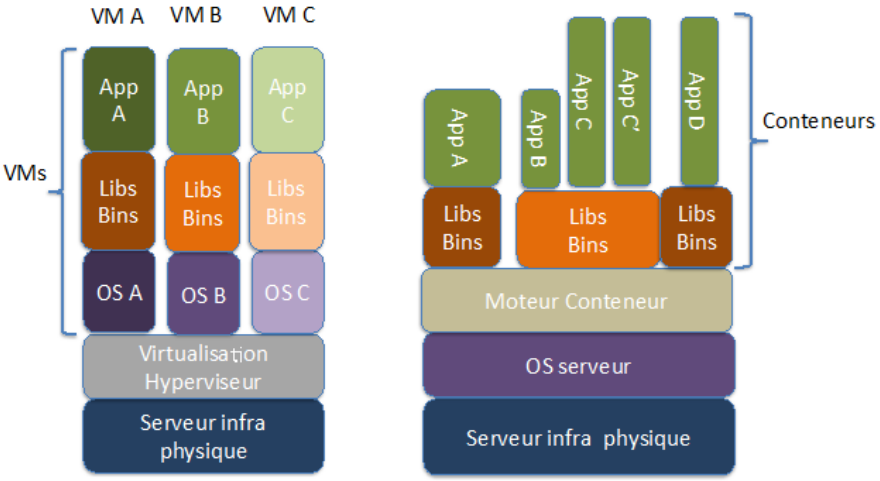


Figure 12: les différences entre conteneurs et virtualisation

Le fonctionnement de docker

Cgroups

Sous Linux il existe deux systèmes de gestion des conteneurs : OpenVz et cgroups (LXC). OpenVz est développé par une entreprise et n'est pas directement inclus dans les distributions Linux. Le mode cgroups/LXC est lié maintenant à Linux et incorporé dans le noyau, il est proposé dans toutes les distributions récentes. Même si openVz possède des fonctionnalités non présentes dans les cgroups (déplacement d'un conteneur à chaud entre deux serveurs), c'est sur ce LXC qu'est basé Docker.

Grâce aux cgroups, Docker possède la couche de virtualisation de serveurs nécessaire au fonctionnement des environnements virtuels, il va apporter la couche de gestion, d'outils et des méthodes qui vont faciliter et même révolutionner l'usage de cette technologie de virtualisation.

Orchestration de conteneurs

Les applications informatiques modernes ne sont plus monolithiques mais composées de nombreuses briques logicielles interconnectés et organisées de manière à isoler les différents composants d'une application, faciliter leur déploiement et la répartition de la charge des services sur plusieurs serveurs. Cette approche permet la mise en place d'infrastructures complexes et exigeantes nécessitant une très haute disponibilité de service et une séparation claire entre les composants applicatifs (Dev) et les composants opérationnels (Ops).

Ces architectures de type "**microservices**" doivent être supervisée par un orchestrateur qui sera notamment en charge de déployer les différents composants, d'associer dynamiquement les volumes de données aux différents services ou encore de gérer les aspects liés au réseau entre

ces différents composants. Plusieurs solutions d'orchestration open source existent mais la plus avancée est clairement **Kubernetes** qui est intégrée dans différentes solutions aux fonctionnalités plus étendues comme **OpenShift** ou **Rancher**. Nos spécialistes maîtrisent ces différentes technologies, du point de vue du développement mais également sur l'ensemble des aspects liés à la mise en place et la maintenance de ces services d'orchestration.

Avec l'explosion de l'utilisation des conteneurs logiciels, principalement popularisés par Docker, arrivent de nouvelles problématiques. Docker Compose répond en partie à certaines de ces problématiques, en regroupant un ensemble de conteneurs sur un même host. Mais comment automatiser et industrialiser le déploiement, la montée en charge et la gestion des a projet en fonctionnant dans des conteneurs ? C'est à cette question que Kubernetes se propose de répondre.

Kubernetes

Kubernetes est un projet Open Source créé par Google en 2015. Il permet d'automatiser le déploiement et la gestion d'applications multi-container à l'échelle. Il s'agit d'un système permettant d'exécuter et de coordonner des applications containerisées sur un cluster de machines. C'est une plateforme conçue pour gérer entièrement le cycle de vie des applications et services containerisés en utilisant des méthodes de prédictibilité, de scalabilité et de haute disponibilité.

Principalement compatible avec Docker, Kubernetes peut fonctionner avec n'importe quel système de container conforme au standard Open Container Initiative en termes de formats d'images et d'environnements d'exécution. Par son caractère open source, Kubernetes peut aussi être utilisé librement par n'importe qui, n'importe où.

Figure 13: Logo de Kubernetes

Les éléments de kubernetes :

À l'instar de toute technologie, Kubernetes dispose d'une terminologie propre. Pour vous aider à mieux appréhender Kubernetes, voyons ensemble les termes les plus courants.

Matser (Maître) : serveur qui contrôle les nœuds Kubernetes, sur lequel toutes les tâches sont assignées.

Node (Nœuds) : machines qui exécutent les tâches qui leur sont assignées. Ces nœuds sont contrôlés par le serveur maître Kubernetes.

Pod : groupe d'un ou de plusieurs conteneurs déployés sur un seul nœud. Tous les conteneurs d'un pod partagent une même adresse IP, un même IPC, un même nom d'hôte et d'autres ressources. Les pods séparent le réseau et le stockage du conteneur sous-jacent. Ainsi, vous pouvez déplacer vos conteneurs au sein du cluster très simplement.

Replicaset Controller(Contrôleur de réplication) : composant qui vérifie le nombre de copies identiques d'un pod qui doivent s'exécuter quelque part dans le cluster.

Service : élément qui dissocie les définitions de tâche des pods. Les proxies de service de Kubernetes transfèrent automatiquement les requêtes de service vers le pod pertinent, même si celui-ci a été précédemment déplacé ou remplacé.

Kubelet : service exécuté sur des nœuds qui lit les manifestes du conteneur pour s'assurer que les conteneurs définis ont démarré et fonctionnent.

kubectl : outil de configuration en ligne de commande de Kubernetes.

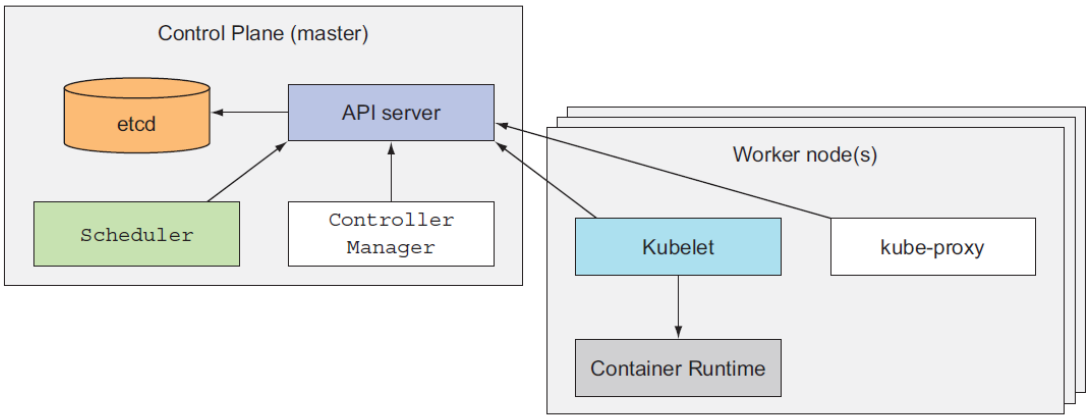


Figure 14: Les composants responsable de mise en place de kubernetes cluster

Fonctionnement de Kubernetes :

Les architectures Kubernetes reposent sur plusieurs concepts et abstractions : certaines existaient déjà auparavant, d'autres lui sont spécifiques. La principale abstraction sur laquelle repose Kubernetes est le cluster, à savoir le groupe de machines exécutant Kubernetes et les containers qu'il gère.

Un cluster Kubernetes doit avoir un master : le système qui commande et contrôle toutes les autres machines du cluster. Un cluster Kubernetes hautement disponible réplique les fonctions

du master sur les différentes machines, mais seul un master à la fois exécute le ***controller-manager*** et le ***scheduler***.

Chaque ***cluster*** contient des noeuds Kubernetes. Il peut s'agir de machines physiques ou virtuelles. Les noeuds quant à eux exécutent des pods : les objets Kubernetes les plus basiques pouvant être créés ou gérés. Chaque pod représente une seule instance d'une application ou d'un processus en cours d'exécution sur Kubernetes, et se constitue d'un ou plusieurs containers. Tous les containers sont lancés et répliqués en groupe dans le pod. Ainsi, l'utilisateur peut se concentrer sur l'application plutôt que sur les containers.

Le **controller** est une autre abstraction permettant de gérer la façon dont les pods sont déployés, créés ou détruits. En fonction des différentes applications à gérer, il existe différents pods. Une autre abstraction est le service, qui permet d'assurer la persistance des applications même si les pods sont détruits. Le service décrit la façon dont un groupe de pods peut être accédé via le réseau.

On dénombre d'autres composants clés de Kubernetes. Le **scheduler** répartit les workloads entre les noeuds pour assurer l'équilibre entre les ressources, et garantir que les déploiements correspondent aux besoins des applications. Le controller manager quant à lui assure que l'état du système (applications, workloads…) correspond à l'état désiré défini dans les paramètres de configuration Etcd.

Chapitre 3 : Réalisation

Dans ce chapitre on va présenter les différentes étapes de mise en place de **kubernetes** cluster et le déploiement des pods Jenkins dans **ce dernies ainsi** .

Installation :

Il y a 3 packages nécessaire pour utiliser kubernetes sont : kubelet, kubeadm, kubectl.

Plus de details sur l'installation et configuration dans l'annexe

Création d'un cluster dans Kubernetes :

Configurez le master

Le master, c'est la machine qui va gérer le cluster. Nous n'allons utiliser qu'un seul master. Pas la peine de s'ennuyer à aller plus loin pour le moment. Sachez tout de même que c'est une mauvaise pratique : s'il se casse la figure, vous êtes dans la mouise.

Initialisation

Pour initialiser le master (k8smaster), pour faire comprendre à la VM que c'est elle la cheffe, tapez la commande suivante :

$$kubeadm\ init\ --pod-network-cidr = 10.244.0.0/16$$

Cette commande va initialiser le cluster et lui assigner une plage d'IP privée

```
[init] Using Kubernetes version: v1.14.0
[preflight] Pulling images required for setting up a Kubernetes cluster
[preflight] This might take a minute or two, depending on the speed of your internet connection
[preflight] You can also perform this action in beforehand using 'kubeadm config images pull'
[certs] Using certificateDir folder "/etc/kubernetes/pki"
[...]
Your Kubernetes control-plane has initialized successfully!
To start using your cluster, you need to run the following as a regular user:
  mkdir -p $HOME/.kube
  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
  sudo chown $(id -u):$(id -g) $HOME/.kube/config

You should now deploy a pod network to the cluster.
Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:
  https://kubernetes.io/docs/concepts/cluster-administration/addons/
```

```
Then you can join any number of worker nodes by running the following on each as root:
kubeadm join 10.242.2.246:6443 --token 16nv1q.13wpz38pdbyptl46 \
  --discovery-token-ca-cert
-hash sha256:017b694e05f70dfcbe78b6eb6c0f1e3282e39a94588662f3f331ad282fe55e04
```

Le message affirmant que le master est bien initialisé parle de lui-même

Configurez l'utilisateur qui aura le droit de jouer avec k8s

Jongler entre k8s et docker peut être épuisant : certaines commandes se jouent en root et d'autres en simple utilisateur. Pensez à changer d'utilisateur si rien ne se passe ou si le terminal vous insulte.

```
mkdir -p $HOME/.kube

sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

Configurer le réseau du cluster

La suite du message d'initialisation :

```
You should now deploy a pod network to the cluster.
Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:
 https://kubernetes.io/docs/concepts/cluster-administration/addons/
```

k8s se sert d'un pod pour gérer son réseau. Il en existe plusieurs et celui que j'ai réussi à faire marcher du premier coup s'appelle **Flannel**.

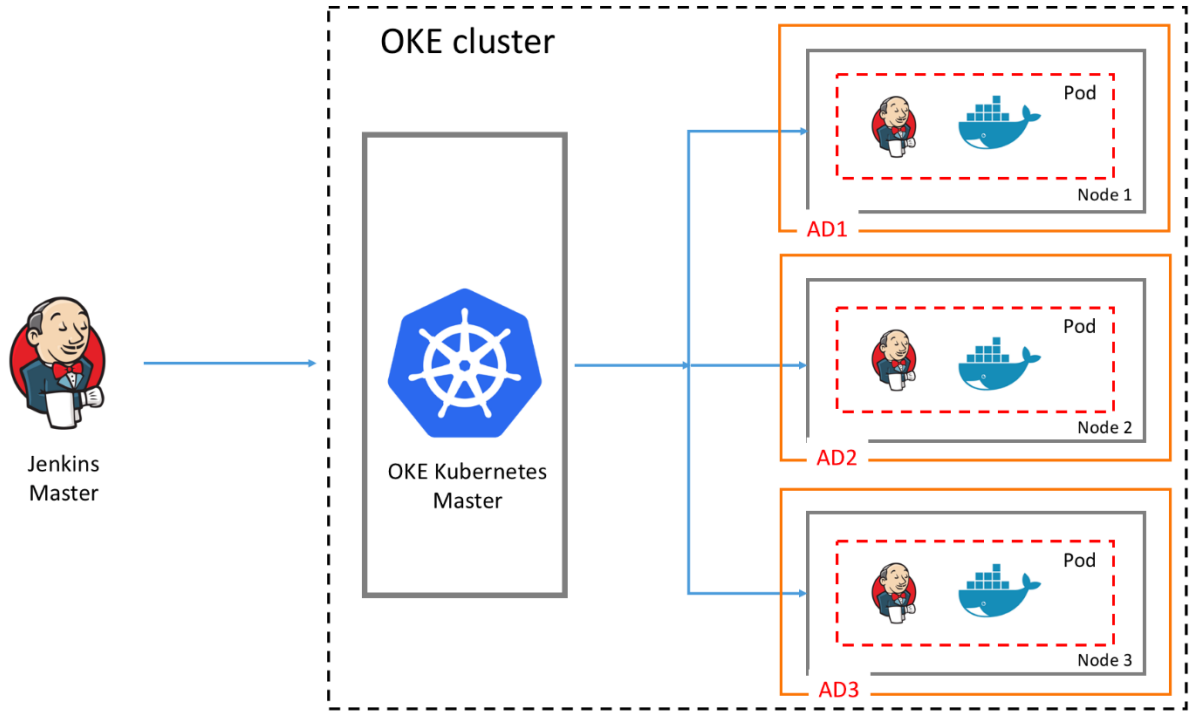
```
clusterrole.rbac.authorization.k8s.io/flannel
created clusterrolebinding.rbac.authorization.k8s.io/flannel
created serviceaccount/flannel created configmap/kube-flannel-cfg create
daemonset.extensions/kube-flannel-ds-amd64 created
daemonset.extensions/kube-flannel-ds-arm64 created
daemonset.extensions/kube-flannel-ds-arm created
daemonset.extensions/kube-flannel-ds-ppc64le created
daemonset.extensions/kube-flannel-ds-s390x created
```

$$kubectl\ get\ pods\ --all-namespaces\ -o\ wide$$

Cette commande permet d'afficher les différentes pods dans les différents namespaces dans k8s cluster.

Intégration entre Jenkins et kubernetes :

L'idée générale de cette intégration est que lors d'un build Jenkins Kuberntes Cree un pod qui contient les conteneurs que l'applications nécessite pour le cycle d'intégration, Livraison et le déploiement à la fin de pipeline les pods et les conteneurs sont supprimés automatiquement pour libérer de l'espace disque et mémoire.



Cette intégration est archive a l'aide d'un plugins dans jenkins .