

UNIVERSITÉ NATIONALE DU VIETNAM À HANOÏ  
INSTITUT FRANCOPHONE INTERNATIONAL

---



Option : Systèmes Intelligents et Multimédia (SIM)

*Promotion* : 22 Année académique 2017-2018

TRAVAUX PRATIQUES ENCADRÉS

THÈME : DevOps(Software Development (dev) and Software Operation  
(ops))

Effectué par :

ZONGO Sylvain

Encadrant :Dr. Ho Tuong Vinh

# Table des matières

<b>Introduction Générale</b>	<b>3</b>
<b>1 Chapitre 1 : Analyse du sujet</b>	<b>3</b>
1.1 Introduction . . . . .	3
1.2 Définition . . . . .	4
1.3 Problématique . . . . .	4
1.4 Objectifs . . . . .	5
1.5 Domaine d'étude . . . . .	5
1.6 Travaux à réaliser . . . . .	6
1.7 Travaux pratiques . . . . .	6
1.8 Résultats attendus . . . . .	6
1.9 Conclusion . . . . .	6
<b>2 Chapitre 2 : Bibliographie</b>	<b>7</b>
2.1 Introduction . . . . .	7
2.2 Histoire . . . . .	7
2.3 Les techniques de DevOps . . . . .	7
2.3.1 Amélioration continue . . . . .	7
2.3.2 La gestion des versions . . . . .	7
2.3.3 Intégration continue . . . . .	8
2.3.4 Déploiement continu . . . . .	8
2.3.5 Tests continus . . . . .	9
2.3.6 Surveillance et retours continus . . . . .	9
2.4 L'état de l'art de DevOps . . . . .	10
2.5 Étude comparative . . . . .	11
2.5.1 Outils de gestion de versions : Git vs SVN . . . . .	12
2.5.2 Outils d'orchestration : Jenkins vs Travis vs GitLab . . . . .	12
2.5.3 Outil de configuration : Chef vs Puppet vs Ansible . . . . .	13
2.5.4 Virtualisation : Machine virtuelle vs Docker . . . . .	14
2.5.5 Plateforme de déploiement :Openshift vs Cloud foundry . . . . .	15
2.6 Conclusion . . . . .	15
<b>3 Chapitre 3 : Notre solution proposée</b>	<b>17</b>
3.1 Introduction . . . . .	17
3.2 Description de la solution proposée . . . . .	17
3.3 Planification des tâches . . . . .	17
3.4 Choix des outils . . . . .	18
3.5 Justification du choix des outils . . . . .	18
3.5.1 Gestion de versions : Git . . . . .	18
3.5.2 Intégration continue :Jenkins . . . . .	19

3.5.3	Déploiement continu :Ansible . . . . .	19
3.5.4	Environnement de virtualisation et de test : Docker . . . . .	20
3.6	Les bonnes pratiques . . . . .	20
3.6.1	La théorie des Contraintes . . . . .	20
3.6.2	Les meilleures pratiques résumées comme «CALMS» . . . . .	21
3.7	Conclusion . . . . .	22
<b>4</b>	<b>Chapitre 4 : phase pratique</b>	<b>23</b>
<b>5</b>	<b>Description du processus de mise en place de DevOps</b>	<b>23</b>
5.1	Les outils . . . . .	23
5.2	Installation de git sur linux . . . . .	23
5.3	Configuration de github . . . . .	23
5.4	Git-Github . . . . .	24
5.5	Installation et configuration de jenkins . . . . .	24
5.5.1	Etape 1 – Add Jenkins PPA . . . . .	24
5.5.2	Etape 2 : Install Jenkins on Ubuntu . . . . .	24
5.5.3	La configuration de plugin Github dans jenkins et Github . . . . .	24
5.6	Automatisation des tests . . . . .	27
5.6.1	Les tests unitaires . . . . .	28
5.7	Heroku . . . . .	28
<b>6</b>	<b>Déploiement automatisé</b>	<b>30</b>
<b>7</b>	<b>Résultats</b>	<b>31</b>
7.1	Page d’accueil : . . . . .	31
7.2	Collaborateurs . . . . .	31
7.3	Jenkins-github . . . . .	32
7.4	Jobs et Builds . . . . .	32
7.5	Modification et statistique . . . . .	33
7.6	Déploiement sur la plate forme heroku . . . . .	34
7.7	Déploiement sur la plate forme heroku . . . . .	35
7.8	Conclusion et perspectives . . . . .	35
	<b>Conclusion Générale</b>	<b>36</b>

# Introduction Générale

DevOps est une méthodologie très en vogue dans tous les niveaux des IT. Il désigne la pratique consistant à rationaliser le processus de développement en améliorant la collaboration, la normalisation et l'automatisation. Le service informatique a un avantage concurrentiel en conciliant les développeurs et les opérateurs. DevOps est arrivé quand la productivité amenée par le cloud, l'agilité demandée par la conquête de l'Internet et le fonctionnement continuellenent amené par la mondialisation de l'Internet, se sont combinés pour créer ce nouveau paradigme. L'évolution des méthodes informatiques ne pouvait plus progresser de façon incrémentale comme elle l'avait fait depuis le mainframe, puis le client serveur. Un point de rupture était atteint. C'est donc une nouvelle façon de faire de l'informatique adaptée à l'attente numéro un des entreprises : leur transformation digitale interne et externe.

Il s'agit d'une approche qui se base sur la synergie entre la partie opérationnelle et la partie production, l'alignement de l'ensemble des équipes du système d'information sur un objectif commun a pour effet de réduire les conflits entre ces différents intervenants, d'éviter les retards dûs à la communication entre eux, et d'améliorer, par conséquent, les délais des livrables (Time-To-Market)[8].

La première étape d'une approche DevOps consiste à reconnaître la façon dont le développement de logiciels, les opérations informatiques et QA sont mutuellement dépendants les uns des autres. Comme mentionné plus haut, DevOps repose sur la collaboration interministérielle et une communication ouverte entre les acteurs clés dans le pipeline de distribution de logiciels afin d'accroître l'efficacité opérationnelle, la prévisibilité et la maintenabilité. L'intégration et l'automatisation de ces éléments au début du processus permet aux équipes de diffuser la prestation de logiciels[3].

## 1 Chapitre 1 : Analyse du sujet

### 1.1 Introduction

La mise en production est un processus faisant intervenir plusieurs équipes aux rôles différents. Faire intervenir autant d'équipes peut mener à des conflits tant le but de chaque équipe est différent de l'autre. Les développeurs souhaitent innover et faire évoluer les applications, l'équipe de production cherche, avant tout, à maintenir la stabilité du système informatique. Chaque équipe se contente des intérêts de son domaine avec des outils qui souvent sont totalement différents avec les outils des autres équipes.

En conséquence, les relations entre ces équipes peuvent être conflictuelles, ces conflits génèrent des retards de livraison, des coûts supplémentaires pour l'entreprise qui n'ont pas été prévus au départ, avec un impact sur le client dont la satisfaction est au centre des préoccupations de l'entreprise.

Il devient donc évident qu'il faut adopter une nouvelle approche qui permet d'unifier le processus

de développement et de production afin d'éviter tous les problèmes précédemment cités. C'est dans cette optique qu'est né DevOp(Software Development (Dev) and Software Opération(Op)) afin de permettre un développement et un déploiement plus efficace et rapide. Ce présent document est une analyse du travail à faire tout au long de nos Travaux Personnels Encadrés(TPE).

## 1.2 Définition

DevOps est composé de Dev qui veut dire Developpment et de Ops qui signifie Opérations . Le terme DevOps désigne une pratique consistant à rationaliser le processus de développement en améliorant la collaboration, la normalisation et l'automatisation. Les applications développées, les infrastructures et les équipes qui les créent ne sont plus séparées, mais au contraire étroitement liées entre elles afin de produire de la valeur ajoutée au sein des entreprises. Il est très important de connaître qu'il existe une multitude de définitions de DevOps mais toutes les définitions ont en commun la collabaration et l'automatisation du déploiement continu.

## 1.3 Problématique

De nos jours de nombreuses équipes interviennent dans le processus de developpement des applications. Cette diversité peut mener à des conflits tant le but de chaque équipe est différent de l'autre. Les développeurs souhaitent, à leur niveau, innover et faire évoluer les applications, l'équipe d'exploitation cherche, avant tout, à maintenir la stabilité du système informatique. Chaque équipe suit ses processus, et travaille avec ses propres outils qui sont rarement communs. En conséquence, les relations entre ces équipes peuvent être conflictuelles, ces conflits génèrent des retards de livraison, des coûts supplémentaires pour l'entreprise qui n'ont pas été prévu au départ, avec un impact sur le client dont la satisfaction est au centre des préoccupations de l'entreprise. Ajouter à cela force est de constater aujourd'hui que les méthodes( Agile, MosCow) de développement ont bien facilité les processus de développement et de livraison de logiciel, cependant il est également à remarquer leur limite majeur car elles seules ne permettent pas de déployer rapidement les nouveaux logiciels qui connaissent une constante évolution d'où la nécessité de DevOps.

Comment briser le mur de la confusion entre les développeurs (au sens large : product owner, concepteur, développeur, testeur...) et des Opérateurs (administrateur système, exploitant, responsable sécurité, ingénieur réseau, DBA – administrateur base données...)?

Comment suivre le rythme de l'évolution rapide des activités?

Le processus de DevOps se présente comme suit : **DevOps = Dev + Ops**[5]

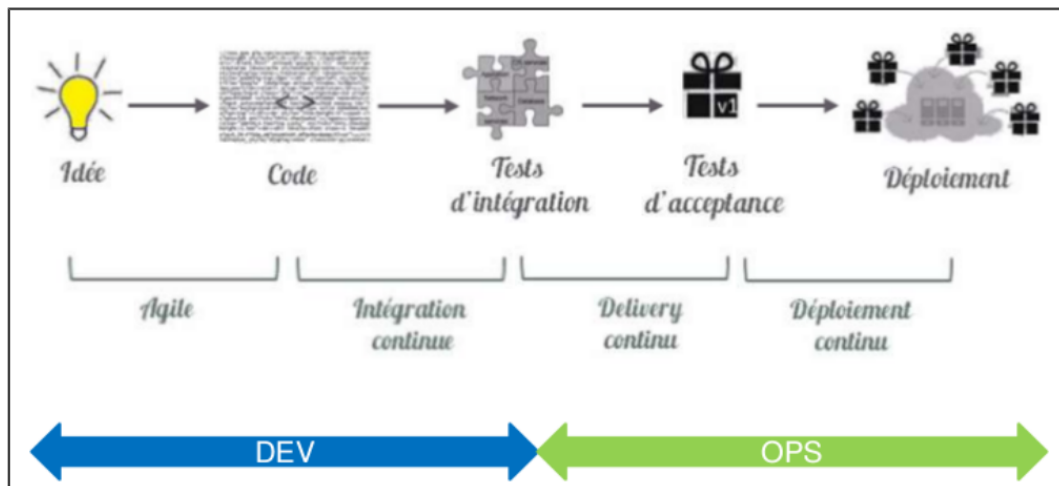


FIGURE 1 – structure DevOp

En resumé les contraintes à résoudre sont :

- Réduire la durée comprise entre la demande de modification d'un IT et sa production ;
- Plus d'arrêt de production ;
- Intolérance aux pannes ;
- Complexité des infrastructures et applications rendant impossibles les mise à jours manuelles ;
- Augmentation de la sollicitation aux systèmes d'informations ;
- Augmentation de la taille des datacenters, des infrastructures et des applications à maintenir.

Ainsi dans la suite de notre travail de TPE voici des questions et bien d'autres que nous tenterons de reprendre.

## 1.4 Objectifs

- Faciliter la compréhension et les interactions entre les équipes de developpement et de production ;
- Réduire les retard lors du developpement et surtout de l'exploitation des logiciels ;
- Fourniture rapide de nouvelles fonctionnalités aux utilisateurs ;
- Etre plus réactif et stable (temps de cycle réduits, taux de déploiement plus élevé) ;
- Permettre une meilleure sécurité lors des livraisons applicatives ;
- Avoir une meilleure efficacité organisationnelle (moins d'astreinte, travail consacré aux activités à forte valeur ajoutée) ;
- Baisser les coûts de production ;
- Améliorer la qualité de service.

## 1.5 Domaine d'étude

Devops est un sujet de type apprentissage et maîtrise d'une technologie. Dans son sens large il peut s'appliquer à tout domaine qui a pour vision d'innover car c'est une culture et toute culture

peut être adoptée. Mais au sens strict de la définition il est centré sur le domaine informatique car c'est l'automatisation du développement, des tests et des déploiements des applications. DevOps repose sur les individus, les processus, les technologies, l'application de principes et des méthodes afin de renforcer la collaboration entre les équipes de développement de logiciels et des opérations dans le IT.

## **1.6 Travaux à réaliser**

### **Travaux théoriques**

- Étude et regroupement des approches en fonction des méthodes et techniques, et des outils utilisés ;
- Faire une synthèse des avantages ;
- Faire une synthèse des inconvénients ;
- Définir l'environnement et les outils pour la mise en place de DevOps ;
- Proposer une approche de mise en place.

## **1.7 Travaux pratiques**

- Créer un pipeline de livraison continue et d'automatisation ;
- Établir une liste des bonnes pratiques.

## **1.8 Résultats attendus**

- Meilleure prédiction du déploiement ;
- Augmentation de l'efficacité de développement et déploiement continu ;
- Amélioration de la maintenance opérationnelle, car confortable avec le changement ;
- Mise en place de tâches en libre service pour l'équipe (self-service).

## **1.9 Conclusion**

Dans cette partie nous avons fait l'analyse de notre sujet qui DevOps afin de mieux le comprendre et de délimiter ses frontières. Nous allons à présent aborder la seconde partie qui est la bibliographie dans laquelle nous parlerons de l'état de l'art de DevOps.

## **2 Chapitre 2 : Bibliographie**

### **2.1 Introduction**

Après l'analyse de sujet nous allons dans cette partie faire l'état de l'art, étudier les différentes techniques et approches de DevOps.

### **2.2 Histoire**

Le nom DevOps a été donné par Patrick Debois au mouvement émergent cherchant à améliorer la qualité des services fournis par les solutions informatiques, lors de l'organisation des premiers devopsdays à Gand en Belgique, en octobre 2009. Ses acteurs sont des personnes issues du monde du développement, de l'exploitation mais aussi de la qualité et des tests, des personnes intéressées par l'amélioration de la performance économique ou environnementale des solutions informatiques, ou encore des consultants en organisation ou en méthodes, notamment agile ou lean.

Les personnes se réclamant de DevOps tentent de tirer les leçons de l'efficacité des sociétés acteurs du web, telles que Google, Amazon ou Yahoo afin de pouvoir en faire bénéficier toute société, par le biais de la bonne combinaison de personnes impliquées, de solutions innovantes, d'outils adaptés et de pratiques aux résultats prouvables. Le mouvement devops se veut non dogmatique et souhaite, par le biais d'une réflexion hors des sentiers battus, réinventer l'informatique d'entreprise afin d'utiliser ses ressources informatiques pour augmenter sa compétitivité.

### **2.3 Les techniques de DevOps**

#### **2.3.1 Amélioration continue**

Dans une vision Lean, l'adoption du processus n'est pas une action ponctuelle ; il s'agit d'un processus qui évolue en permanence. Une entreprise doit disposer de processus intégrés qui identifient les domaines d'amélioration au fur et à mesure que l'organisation devient plus mature et qu'elle apprend des processus qu'elle a adoptés. La plupart des entreprises possèdent des équipes dédiées à l'amélioration des processus, qui s'attachent à améliorer les processus en fonction des observations et des leçons apprises. D'autres permettent aux équipes qui adoptent les processus d'évaluer et de déterminer elles-mêmes leurs propres procédures d'amélioration des processus. Quelle que soit la méthode utilisée, l'objectif vise à instaurer l'amélioration continue.[5]

#### **2.3.2 La gestion des versions**

Système logiciel permettant de maintenir et gérer toutes les versions d'un ensemble de fichiers. La gestion de versions permet :

- De revenir aisément à une version précédente ;
- De suivre l'évolution du projet au cours du temps ;
- Le travail en parallèle sur les parties disjointes du projet et gérer les modifications concurrentes ;



- De faciliter la détection et la correction des erreurs.

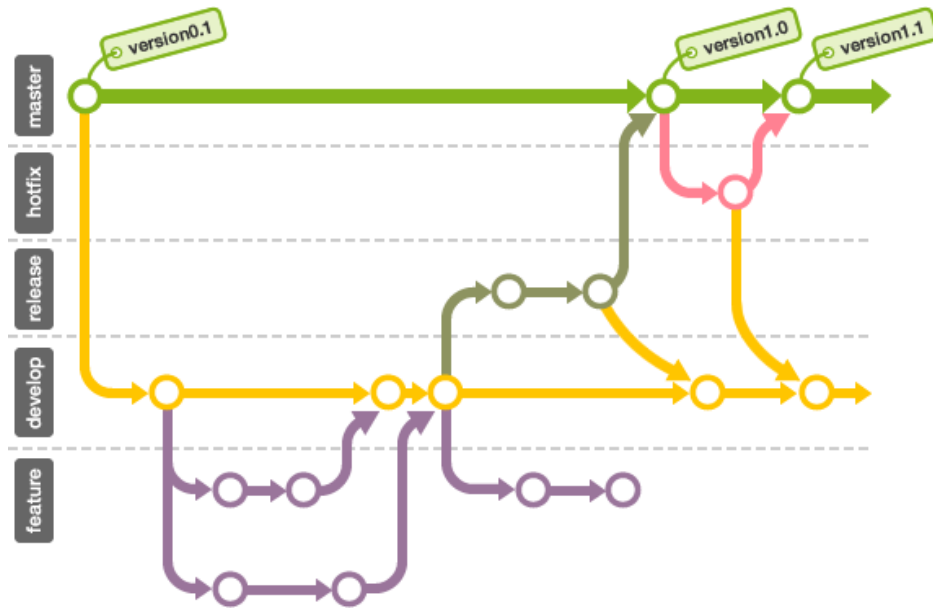


FIGURE 2 – branches pour les versions

### 2.3.3 Intégration continue

L'Intégration Continue, dans sa forme la plus simple, se compose d'un outil qui surveille les modifications de code dans votre gestionnaire de configuration. Dès qu'un changement est détecté, cet outil va automatiquement compiler et tester votre application. Si la moindre erreur arrive alors l'outil va immédiatement avertir les développeurs afin qu'ils puissent tout de suite corriger le problème. Le principal but de cette approche est d'anticiper et d'identifier rapidement les bugs tôt avant la mise en production du logiciel. Cela permet d'avoir une vision plus complète du logiciel, notamment sur les différents points faibles et points forts du code ou de l'équipe. Ceci permet de gagner en réactivité pour faire face aux différents problèmes pouvant être présents dans les diverses phases du projet. Pour bénéficier des avantages qu'apporte le concept d'intégration continue, il est nécessaire de suivre quelques règles de bonnes pratique comme le maintien d'un unique dépôt pour le code source visionné, effectuer plusieurs commit par jour et par développeur, automatiser la compilation du code, et le maintien d'un de compilation court. [7] L'architecture d'un pipeline d'intégration continu, fait intervenir plusieurs éléments[2] :

- Un gestionnaire de code.
- Un serveur d'orchestration ou d'intégration continue.
- Un serveur de gestion de dépôt.

### 2.3.4 Déploiement continu

Le déploiement continu est le processus où toute modification du code, comprenant des tests automatisés et autres vérifications appropriées, est automatiquement déployée en production.

L'objectif est de réduire la durée du cycle ainsi que le temps et l'effort du processus de déploiement. Cela permet également aux équipes de développement de réduire le temps nécessaire pour la livraison de fonctionnalités individuelles ou des corrections de bogue, et par conséquent augmente considérablement la productivité des équipes. Éliminer ou réduire ces périodes d'intenses activités menant à une livraison traditionnelle ou à un déploiement libère également des ressources et du temps pour améliorer les processus[1][8].

### **2.3.5 Tests continus**

L'intégration continue a plusieurs objectifs : permettre les tests et la vérification continus du code ; vérifier que le code produit et intégré à celui des autres développeurs et des autres composants de l'application fonctionne et s'exécute comme prévu ; tester en continu l'application en cours de développement. Les tests continus impliquent de tester au plus tôt et en continu tout au long du cycle de vie, ce qui permet de réduire les coûts, de raccourcir les phases de test et de disposer de retours en continu sur la qualité. Ce processus, appelé également *shift-left testing* vise à intégrer les activités de développement et de test pour que la qualité soit incorporée aussi tôt que possible dans le cycle de vie et non pas repoussée à une phase ultérieure[1]. Cela est facilité par l'adoption de fonctionnalités telles que l'automatisation des tests et la virtualisation des services. La virtualisation des services est une nouvelle fonctionnalité de simulation des environnements de type production qui rend possibles les tests continus.

### **2.3.6 Surveillance et retours continus**

Les retours des clients revêtent différentes formes : tickets ouverts par le client, demandes formelles de modification, réclamations informelles ou évaluations sur les magasins d'applications (app stores). Du fait du succès des réseaux sociaux et des magasins d'applications les entreprises ont besoin de processus bien définis pour traiter les retours d'une multitude de sources différentes et pour les intégrer dans les plans de déploiement des logiciels. Ces processus doivent également être suffisamment agiles pour s'adapter au marché et à l'évolution des réglementations. Les retours peuvent aussi provenir de la surveillance des données. Ces données sont issues des serveurs sur lesquels s'exécute l'application (serveurs de développement, de QA ou de production) ou bien d'outils de mesure intégrés à l'application, qui capturent les actions de l'utilisateur. Une surcharge de données peut se produire. Par conséquent, les entreprises doivent utiliser des processus de capture et d'utilisation des données qui améliorent les performances des applications et les environnements où elles sont exécutées. Illustrons l'approche DevOps par un schéma :

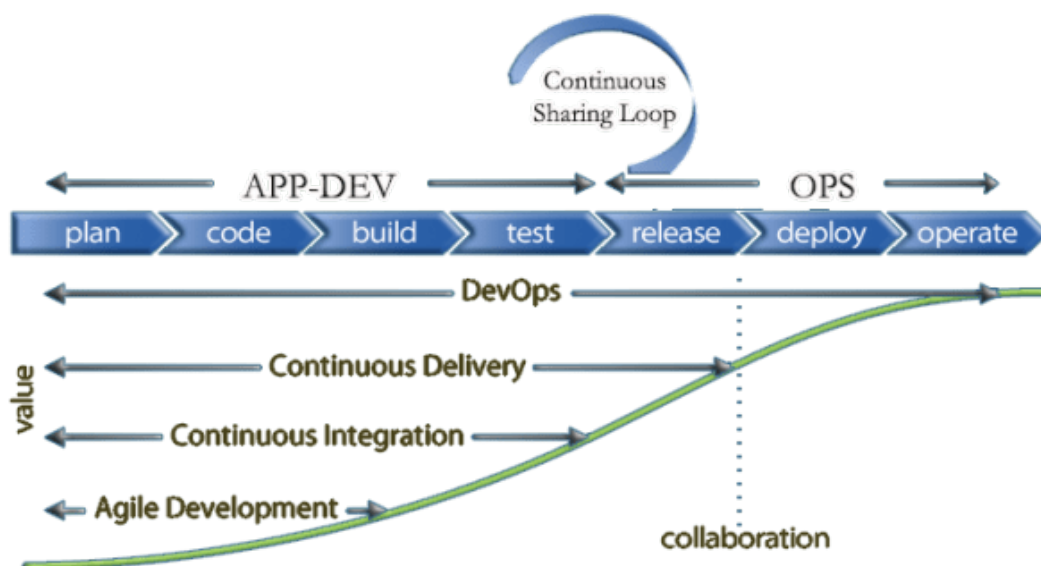


FIGURE 3 – Les composantes de DevOps

## 2.4 L'état de l'art de DevOps

Dans le tableau ci-dessous nous avons quelques approches de DevOps[12].

Différentes approches de DevOps	
Approches	Solutions et outils
Approche IBM	<p>La solution de IBM fournit une plateforme DevOps ouverte et normalisée supportant l'innovation, la rétroaction et l'amélioration de cycle de vie des logiciels tout en permettant à l'entreprise de planifier de suivre et de les gérer. Elle regroupe les capacités en quatre chemins :</p> <ul style="list-style-type: none"> <li>-Plan/Meseasure</li> <li>-Develop/Test</li> <li>-Release/Deploy</li> <li>-Monitor/optimize</li> </ul> <p>Dans la mise en oeuvre cette approche identifie 6 phases</p> <ul style="list-style-type: none"> <li>-Think : conceptualisation ; amélioration et la priorisation des capacités</li> <li>-Code : génération, amélioration, optimisation et test de fonctionnalités</li> <li>-Deliver :production automatisée et livraison d'offres</li> <li>-Run : Services, options et les capacités requises pour l'exécution</li> <li>-Manage : suivi , soutien et récupération d'offres</li> <li>-Learn : Appritissage cotinu basé sur les résultats des expériences</li> </ul> <p>Les outils utilisés sont :Optimizely,Tealeaf,DevOps Insights,Google Analyti,GitHub,Web IDE,Eclipse,Jazz SCM,AppScan,Load Impact,Sauce Labs Speed Curve,Git Repos ,Issue Trackin, Delivery Pipeline,Jenkins,UrbanCode Deploy Cloud Foundry,Docker,OpenStack,Auto Scaling,SoftLayer,Containers,</p>

... suite page suivante...

... suite de la page précédente...

Approches	Solutions et outils
Approche MICROSOFT AZURE	<p>Availability Monitoring,Alert Notification,Runbook Automation, New Relic,Slack,PagerDuty</p> <p>Cette approche adopte un modèle de maturité à 5 étapes qui vise à maximiser à la fois la qualité et la productivité des processus</p> <p>Les étapes sont les suivantes</p> <ul style="list-style-type: none"> <li>-Release</li> <li>-Develop+Test</li> <li>-Plan+Track</li> <li>-Continuous delivery</li> <li>-Monitor+Learn</li> </ul> <p>Elle se base sur un Prototypage de Preuve de Faisabilité(POC-Proof Of Concept)</p> <p>Elle associe le Business (Biz), le Development (Dev) et les Opreations (Ops) ainsi cette approche utilise le terme BizDevOps à la place de DevOps.</p> <p>Elle est construite sur les technologies Microsoft et Google pour les couches architecturales et logicielles suivantes :</p> <p>Couche Interface Utilisateur (UI) : Angular 2 JavaScript Framework réactif, ...</p> <p>Couche d’Affaires : Objets .NET, routines d’Affaires, ...</p> <p>Couche Data et BI : Azure SQL, Power BI...</p> <p>Couche d’Infrastructure (IaaS et PaaS) : MS Azure, AWS, Salesforce, ...</p> <p>Couche de Sécurité : Azure Active Directory, authentification à deux facteurs, accès basé sur les rôles</p> <p>Les outils utilisés par Microsoft pour la mise en place de DevOps sont :</p> <p>Tout type de langage Java, Node.js, .NET, PHP, Python, and others,</p> <p>IDE—IntelliJ,Eclipse, VS Code,Android Studio plugin,IDEA plugin,</p> <p>Build Android apps,Build with Gradle,Build with Maven,Manage Maven packages,Analyti,GitHub,Web IDE,Eclipse, netBeant Jazz SCM,AppScan,</p> <p>Load Impact,Sauce Labs,Speed Curve,Git Repos ,Issue Trackin,</p> <p>Delivery Pipeline,Jenkins,UrbanCode Deploy,Cloud Foundry,Docker</p> <p>OpenStack,Auto Scaling,SoftLayer,Containers,Availability Monitoring</p> <p>Alert Notification,Runbook,Automation</p>

## 2.5 Étude comparative

Pour la mise en place d’une plate forme DevOps, nous avons besoin d’un ensemble d’outils à savoir, un gestionnaire de version, un orchestrateur, un serveur de dépôt maven, un serveur de déploiement continu. De plus, pour faciliter la tâche de la mise en place de certains de ces éléments, nous avons aussi besoin d’un outil de configuration et de gestion d’ordinateur. Dans cette partie de notre travail nous allons faire une étude comparatives des outils open source

exitant.

### 2.5.1 Outils de gestion de versions : Git vs SVN

Git et SVN sont les deux outils de gestion de version les plus répandus sur le marché, voici un tableau comparatif entre ces deux outils qui nous permettra par la suite d'effectuer notre choix technique.<sup>1</sup>

Comparaison : Git vs SVN		
Outils	Avantages	Inconvénients
Git	Gestion des branches Interface console plus évoluée Algorithmes de fusion(merge) Rapidité Surveillance du contenu des fichiers Système de gestion distribué Moins sensibles aux pannes Adapté aux grands projets	Complexité portage sur windows null très grand nombre de commandes Peut devenir très complexe structurellement
SVN	Facilité de gestion Facilité d'utilisation Surveillance des fichiers	Peu adapté aux grands projets interface console moins évoluée Très sensible aux pannes Système de gestion centrée

### 2.5.2 Outils d'orchestration : Jenkins vs Travis vs GitLab

Jenkins, Travis et GitLab sont des outils d'orchestration bien connus sur le marché, chacun d'entre eux présente ses avantages et inconvénients.<sup>2 3</sup>

Comparaison : Jenkins vs Travis vs GitLab		
Outils	Avantages	Inconvénients
Jenkins	Simplicité d'utilisation plusieurs plugings disponibles developpement d'une extension Portables compatible avec tous les outils de versionneing	Configuration complexe Nécessite un serveur dédié, ce qui peut entraîner une dépense supplémentaire

... suite page suivante...

---

1. <http://www.lsv.fr/~hirschi/pdfs/git.pdf> visité le 10/04/2018

2. <https://www.lebigdata.fr/jenkins-definition-avantages> visité le 20/04/2018

3. <https://www.keycdn.com/blog/jenkins-vs-travis/> visité le 18/04/2018

... suite de la page précédente...

Outils	Avantages	Inconvénients
	Adopté à tout type de projets Compatible avec Libvirt, Kubernetes, Docker Gratuit pour télécharger et à utiliser	
Travis	Léger et facile à mettre en place Gratuit pour les projets open source développement d'une extension Portables Outils de versionneing Adopté à tout type de projets Aucun serveur dédié nécessaire Gratuit pour télécharger et à utiliser	compatible uniquement avec Github Options limitées
GitLab ce	Intégré gratuitement Jobs parallèles Scalable Portables Outils de versionneing Adopté aux projets publics et privés gestion de version Gratuit pour télécharger	Nécessite un serveur dédié, ce qui peut entraîner une dépendance supplémentaire son maintien

### 2.5.3 Outil de configuration : Chef vs Puppet vs Ansible

La mise en place des serveurs peut s'avérer une tâche fastidieuse répétitive et sujette aux erreurs de manipulation. Grâce à ces outils il n'est plus nécessaire de configurer chaque poste individuellement, il suffit de lancer un script, et des dizaines, centaines ou voir millier de machines sont configuré toutes en mêmes temps, réduisant ainsi l'effort fourni et les risques d'erreur du à la manipulation. Ici nous dressons un tableau comparatif entre ces trois outils de configuration <sup>4 5</sup>.

Comparaison : Chef vs Puppet vs Ansible		
Outils	Avantages	Inconvénients
Puppet	-Mutable -Client/serveur  -ordre d'exécution séquentielle -Flexibles pour OS et la gestion de middleware	-Configuration initiale compliquée. -Documentation étendue et devient complexe -Ne dispose pas push

... suite page suivante...

4. <https://www.intigua.com/blog/puppet-vs.-chef-vs.-ansible-vs.-saltstack> visité le 09/04/2018

5. <https://blog.takipi.com/deployment-management-tools-chef-vs-puppet-vs-ansible-vs-saltstack-vs-fabric/> visité le 10/04/2018

... suite de la page précédente...

Outils	Avantages	Inconvénients
Ansible	<ul style="list-style-type: none"><li>-Facile d'exécution à distance</li><li>-Faible barrière d'entrée</li><li>-Moteur d'orchestration puissante</li><li>-L'installation et la configuration initiale facile</li><li>-Ordre d'exécution séquentielle</li><li>-Haute sécurité avec SSH</li></ul>	<ul style="list-style-type: none"><li>-Nécessite un accès SSH racine et interpréteur Python communication</li><li>-SSH ralentit fonctions limitées.</li><li>-La plate-forme est nouvelle</li></ul>
Chef	<ul style="list-style-type: none"><li>-Conçu pour les programmeurs</li><li>-Flexibles pour OS et la gestion de middleware</li><li>-Documentation solide</li><li>-Très stable, fiable et mature</li><li>-Ordre d'exécution séquentielle</li><li>-Être centré autour Git</li><li>-Facilite les charges d'installation</li></ul>	<ul style="list-style-type: none"><li>-La documentation est étendu</li><li>-Ne dispose pas push</li><li>-Ce n'est pas un outil simple</li></ul>

#### 2.5.4 Virtualisation : Machine virtuelle vs Docker

Les machines virtuelles reposent sur le principe de la virtualisation du matériel, c'est-à-dire la création d'un environnement virtuel complet simulant un ordinateur avec du « faux matériel » le système d'exploitation invité communique uniquement avec ce « faux matériel » ce qui rend l'environnement virtualisé étanche. Une couche hyperviseur est ajoutée par-dessus du système d'exploitation pour permettre aux différents systèmes d'exploitation de travailler sur une même machine physique. Docker utilise Docker Engine par-dessus du système d'exploitation hôte qui permet la virtualisation du kernel, Docker utilise le même Kernel et le même réseau que sa machine hôte d'où sa description comme étant « léger ». <sup>6</sup>

Comparaison : Docker vs Machine virtuelle		
Outils	Avantages	Inconvénients
Docker	<ul style="list-style-type: none"><li>-Facilite la distribution des applications</li><li>-Encombrement minimal</li><li>-Sauvegarde à l'aide des fichiers file-systems</li><li>-De nombreuse exécution en parallèle</li><li>-Linux et Windows</li><li>-Permet de reconstruire un container</li><li>-Gestion des containers avec peu d'outils,</li><li>-Identique sur toutes les plateformes</li><li>-Des APIs disponibles pour piloter l'ensemble</li></ul>	<ul style="list-style-type: none"><li>-Pas d'interface graphique libre</li><li>-Difficiles à utiliser pour un déploiement à grande échelle sans développer ses propres outils d'administration</li></ul>

... suite page suivante...

6. <https://www.supinfo.com/articles/single/104-qu-est-que-docker-pourquoi-est-il-different-machines-virtuelles> visité le 10/04/2018

... suite de la page précédente...

Outils de virtualisation	Avantages	Inconvénients
	-Depuis d'autres applications -Leger -Rapide	
Machine virtuelle		-Configuration complexe -Lourde -Moins rapide -Ne peut contenir que peu d'applications

### 2.5.5 Plateforme de déploiement :OpenShift vs Cloud foundry

OpenShift et Cloud foundry sont tous les deux des solutions PAAS pour le déploiement des applications sur des modèle cloud public ou privé et permettent l'hébergement en cloud computing. Ces outils peuvent être intégrés dans un pipeline de déploiement continu d'où notre intérêt. Le tableau comparatif 2.6 sur les deux outils OpenShift Origin et Cloud Foundry se base sur les éléments suivant :<sup>7</sup> Cloud.

- Outils supportés.
- Systèmes d'exploitation

Comparaison : OpenShift vs Cloud Foundry		
Aspect	openShift	Cloud Foundry
Cloud	PAAS	PAAS
Outils supportés	Data dog,Data dog, Jenkins, Eclipse, Jboss studio	GitHub, Eclipse,CloudBees Spring.io
Système d'exploitation	Windows, Linux,Mac	Windows, Linux,Mac

## 2.6 Conclusion

La maturité de DevOps garantit l'émergence de nouveaux outils. Nous avons étudié un ensemble d'outils utiles pour la mise en oeuvre de DevOps. Les outils sont nombreux ainsi le choix judicieux des outils pour le succès d'une mise en oeuvre DevOps se fait en fonctions de nos objectifs mais aussi en fonction de la maturité de notre entreprise pour la mise en place de cette approche. Au cours de notre travail, nous avons décrit de nombreux concepts tels l'intégration continue et le déploiement continu. Nous avons, par la suite, fait une étude comparative sur les

7. <https://comparisons.financesonline.com/openshift-vs-cloud-foundry> visité le 16-04-2018



outils qui peuvent répondre à notre besoin et enfin. A l'issue de cette partie nous allons faire la synthèse de nos choix technologique pour pour la mise en oeuvre de notre solution à proposée[12].

## 3 Chapitre 3 : Notre solution proposée

### 3.1 Introduction

Dans cette partie de notre travail nous allons décrire notre la solution proposée ainsi que les outils pour la mise en place de cette solution. Le choix des outils se basera sur l'étude comparative des outils faite dans la partie recherche bibliographique. Nous donnerons également les bonnes pratiques de DevOps[9].

### 3.2 Description de la solution proposée

L'objectif de notre travail consiste en la mise en place d'une plateforme DevOps comprenant un pipeline d'intégration continue et déploiement continu accompagné d'un tableau de monitoring du système mis en place pour palier à tous ces problèmes que connaît le monde développement et de la livraison des applications. Le pipeline d'intégration continu englobe la mise en place d'un serveur de gestion de version, d'un serveur d'orchestration et d'un serveur de gestionnaire de dépôt.

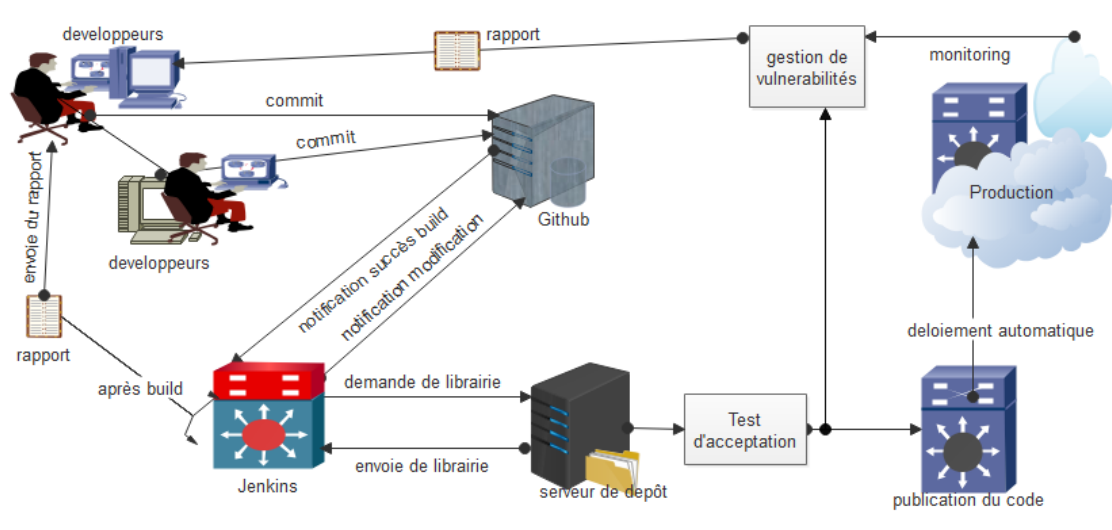


FIGURE 4 – DevOps

Les développeurs pushent leurs codes sur le serveur Git. Cet outil nous permet de créer plusieurs branches pour les différentes modifications et tests de notre code. C'est un élément très important en DevOps pour le retour aux versions antérieures et de corriger les erreurs.

### 3.3 Planification des tâches

Cette partie est l'illustration des différentes tâches et leur durée que nous allons réaliser dans la phase pratique à l'aide de l'outil GanttProject.

taches et durée	
Tache	durée
Création de clusters	21 jours
Configuration du serveur de versioning	5 jours
Mise en place de l'integration continue	21 jours
Tests fonctionnels et d'acceptation	5 jours
Mise en place de deploiement continu	21 jours
Monitoring des clusters et des conteneurs	65 jours

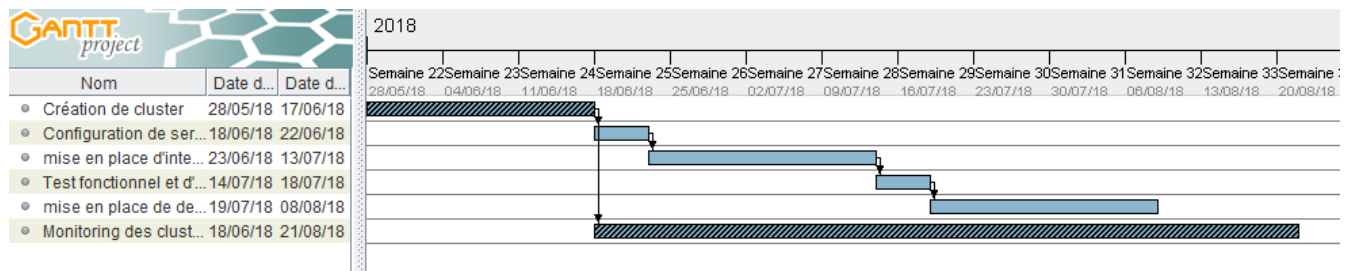


FIGURE 5 – planification des tâches

### 3.4 Choix des outils

A partir de l'étude des outils sur la bibliographie nous allons construire notre plateforme Devops avec les outils suivants :

Fonctionnalités	Outils
Serveur de gestion de version	Git
Serveur d'orchestration	Jenkins
Configuration de cluster	Ansible
Outil de gestion de dépôt maven	Nexus sonatype
Plateforme de deploiement	OpenShift

### 3.5 Justification du choix des outils

#### 3.5.1 Gestion de versions : Git

Nous avons choisi Git pour la gestion des versions pour plusieurs raisons :

- Surveille le contenu des fichiers
- Système de gestion distribuée
- Moins sensibles aux pannes
- Adapté aux grands projets

### 3.5.2 Intégration continue :Jenkins

Dans n'importe quel projet, il est recommandé d'avoir une plateforme d'intégration continue. Ceci permet d'intégrer automatiquement les modifications réalisées sur le code et de notifier les développeurs lorsqu'un test ne passe plus. Dans notre intégration continue nous avons choisi Jenkins puisqu'à la différence des autres outils Jenkins est compatible avec la quasi totalité des environnements, des autres outils grâce à ses nombreuses extensions ainsi que d'autres fonctionnalités comme suit.

- Jenkins détecte un changement sur la branche master. Il lance donc un build
- Jenkins détecte un changement sur la branche master.
- Jenkins notifie l'équipe du problème via mail, slack, etc.
- Jenkins est compatible avec presque tous les environnements.
- Jenkins intègre de nombreux plugings ;
- Possibilité de développer son propre module et de l'intégrer.

### 3.5.3 Deployment continu :Ansible

Ansible automatise l'approvisionnement logiciel, la gestion de la configuration et le déploiement applicatif. À la différence des autres outils Puppet et Chef, avec Ansible nous avons pas besoin d'un administrateur dédié à son administration car sa facilité d'utilisation et de prise de main permet à toute personne de le maîtriser aisément explique ainsi Martin Rusev dans[11]. Au delà cette facilité d'utilisation Ansible est un système sans agent , il gère plus facilement les commutateurs et les tableaux de stockage », explique le directeur de recherche de Gartner[11].

- Simple : l'automatisation est lisible par l'homme ne nécessitant ainsi pas de connaissance ou compétence particulier en codage. Ainsi les équipes sont beaucoup plus productive.
- Le manque de maîtrise élimine les points de défaillance et les problèmes de performance. déploiement sans agent et de la communication est plus rapide que le modèle maître-agent.
- Moteur d'orchestration puissante. accent sur les domaines où d'autres manquent, comme les mises à jour de temps d'arrêt de roulement à des applications multi zéro-niveau à travers le nuage.
- ordre d'exécution séquentielle.
- Haute sécurité avec SSH.
- Installation facile : il n'y a aucun agent à exploiter et utilise OpenSSH et WinRM.
- Facile à prendre en main. Un tableau de bord visuel est intégré avec des éléments de vérification et des notifications améliorée.
- Pour les équipes : gain de temps, plus de productivité, automatiser les tâches redondantes et répétitives, limite les erreurs de déploiement et améliore la qualité de travail en équipe.
- Pour les entreprises : meilleure utilisation des ressources en l'attribuant à l'innovation, faire des économies, faire face à la concurrence avec un déploiement rapide et sécurisé des logiciels.

### 3.5.4 Environnement de virtualisation et de test : Docker

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, bibliothèques, outils et fichiers. Ils sont packagés en un ensemble cohérent et prêt à être déployé sur un serveur et son OS.

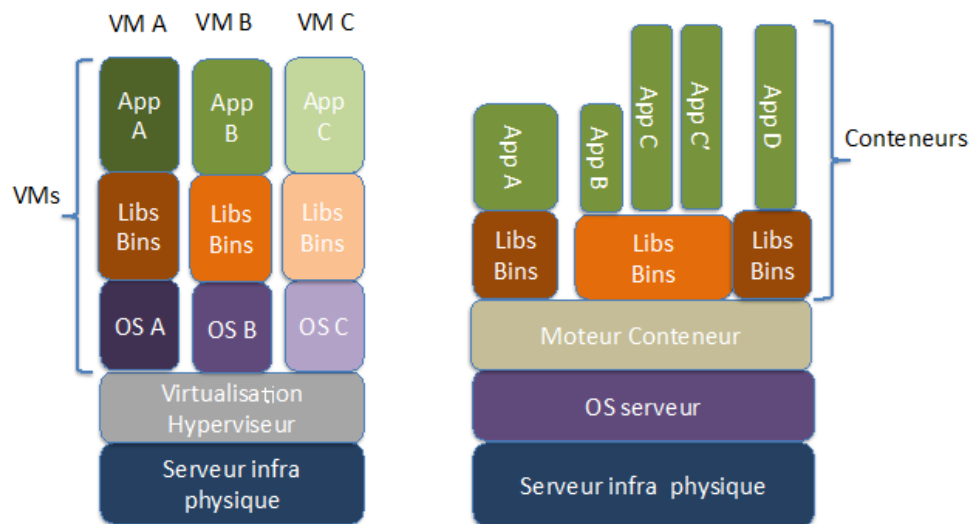


FIGURE 6 – *Caption*

le conteneur est plus rapide, plus léger, plus efficace et utilise moins de ressources que les machines virtuelles ce qui donne l'avantage à son utilisation que celles-ci[4].

## 3.6 Les bonnes pratiques

C'est l'introduction d'un certain nombre de meilleures pratiques et de transformations qui apportent chacune un petit avantage vers l'objectif plus large de versions plus rapides et de meilleure qualité. Chacune de ces transformations a un effort et un coût. L'importance dépend de la situation actuelle de l'organisation, du type d'application et du bénéfice souhaité de la mise en œuvre de DevOps.

### 3.6.1 La théorie des Contraintes

La théorie des Contraintes est l'une des meilleures pratiques en DevOps et se focalise sur cinq étapes de mise au point :

- Identifier la contrainte : Localisez le maillon le plus faible dans le système, ce qui peut être à la fois physique ou une règle de politique.
- Exploiter la contrainte : Apporter des améliorations rapides à la productivité des contraintes avec les ressources existantes.

- Subordonnent tout autre : Faites de la contrainte votre priorité pour obtenir un maximum d'efficacité. Tout le reste doit prendre un siège arrière.
- Élevez la contrainte : prendre toutes les mesures nécessaires pour éliminer la contrainte. Si tout le reste a échoué, apporter des changements majeurs au système existant à ce point.
- Répétez le processus : itération continue aidera à prévenir la complaisance.

Les étapes ci-dessus sont grands pour concentrer les efforts sur l'identification et l'élimination des goulots d'étranglement au sein d'un processus. Juste en suivant ces pratiques DevOps / Lean avec la théorie de la méthodologie des contraintes, les équipes informatiques seront en mesure de voir une augmentation spectaculaire de la productivité, la qualité de la production, ainsi que la satisfaction interne et externe[10].

### 3.6.2 Les meilleures pratiques résumées comme «CALMS»

[6]

- La culture

DevOps est une approche extrêmement centrée sur les personnes. La mise en place de DevOps se fait avec des personnes qui sont prêtes à faire face aux changements. Si des outils d'automatisation et des processus ont été adoptés par l'entreprise, sans des personnes capables d'exploiter de façon efficiente ces derniers, ils ne seront d'aucune utilité. C'est pourquoi il est primordial avant toute mise en place de DevOps d'accompagner au changement. Cela permettra de briser l'organisation héritée des modèles traditionnels (Waterfall, Cycle en V, etc.). Les équipes doivent également être sensibilisées sur l'intérêt des tests (unitaires, intégrations, performances, etc.), l'automatisation, etc.

- L'automatisation

L'automatisation est l'un des aspects clés de la mise en place de DevOps. Des outils doivent être disponibles pour mettre en place une chaîne de livraison, du développement jusqu'à la production, passant par des environnements de tests. De ce fait, il est indispensable d'avoir une infrastructure d'intégration et de la livraison continue. Toute intervention manuelle doit être proscrite.

- Lean

Comme je l'ai dit plus haut, DevOps s'inspire en partie de la culture Lean. Cette dernière est centrée sur l'objectif à atteindre, tout en évitant les pertes de temps. DevOps se focalise donc sur la livraison de la valeur ajoutée aux utilisateurs, en éliminant tout ce qui n'est pas utile pour ceux-ci. Les livraisons doivent être fréquentes et Just in Time.

- Mesure

Dans un projet DevOps, les parties prenantes doivent s'améliorer de façon continue. Il est quasi impossible d'apporter des améliorations sans être en mesure d'évaluer de façon fiable ce qui a été fait. C'est pourquoi il est recommandé de disposer des indicateurs permettant de mesurer les performances de l'application, d'évaluer la qualité des livrables, etc. Même lorsque l'application est en production, des indicateurs de performances et de qualité doivent permettre de mesurer la qualité de l'application.

- Partage

DevOps vise une meilleure cohésion entre les équipes autour du même but. La circulation des informations entre celles-ci est donc primordiale pour l'atteinte de l'objectif commun. Les équipes doivent communiquer et être soudées. Un problème ne doit pas être imputé à une personne. Mais, l'ensemble des parties prenantes doit collaborer pour offrir une solution rapide et fiable.

— Approche Agile

Avec l'approche Agile, rien n'est figé. L'équipe projet doit être capable de se remettre sans cesse en cause et de chercher continuellement à évoluer.

### **3.7 Conclusion**

Au cours de cette étape, nous avons défini de nombreux concepts tels l'intégration continue et le déploiement continu. Nous avons dans cette étape proposée notre solution suivie d'une justification des outils que nous avons choisis pour la mise en place de cette solution. A l'issue de cette partie nous allons faire la synthèse de nos outils et pourquoi ces outils pour notre solution. [9]

## 4 Chapitre 4 : phase pratique

## 5 Description du processus de mise en place de DevOps

### 5.1 Les outils

- git
- github
- Jenkins
- Heroku

### 5.2 Installation de git sur linux

- `sudo apt-get update`
- `sudo apt-get install git`
- `sudo git -version` : pour la vérification de la version installée

### 5.3 Configuration de github

- Créer d'abord un compte github
- Créer un repository

The screenshot shows the GitHub 'Join GitHub' page. At the top, it says 'Join GitHub' and 'The best way to design, build, and ship software.' Below this is a three-step process bar: Step 1: Create personal account (active), Step 2: Choose your plan, and Step 3: Tailor your experience. The main section is 'Create your personal account'. It has three input fields: 'Username' with the value 'ZONGOSYLVAIN', 'Email address' with 'onezongos@gmail.com', and 'Password' with masked characters. Each field has a green checkmark icon to its right. Below the password field is a note: 'Use at least one lowercase letter, one numeral, and seven characters.' To the right of the form is a box titled 'You'll love GitHub' containing three items: 'Unlimited collaborators', 'Unlimited public repositories', and three green checkmarks next to 'Great communication', 'Frictionless development', and 'Open source community'. At the bottom of the form is a 'Verify account' link.

FIGURE 7 – *github*



## 5.4 Git-Github

Une fois le git installé et le compte github créé suivez les instructions suivantes pour envoyer votre projet sur github.

1. Ouvrir le terminal
2. Se rendre dans le repertoire contenant le projet et taper les commandes suivantes
3. `git init`
4. `git add -all`
5. `git commit -m "version du projet"`
6. `git remote add origin https ://github.com/NomUtilisateur/NomRepository.git`
7. `git push -u origin master`
  - Ajouter le nom d'utilisateur
  - Ajouter le mot de passe

## 5.5 Installation et configuration de jenkins

### 5.5.1 Etape 1 – Add Jenkins PPA

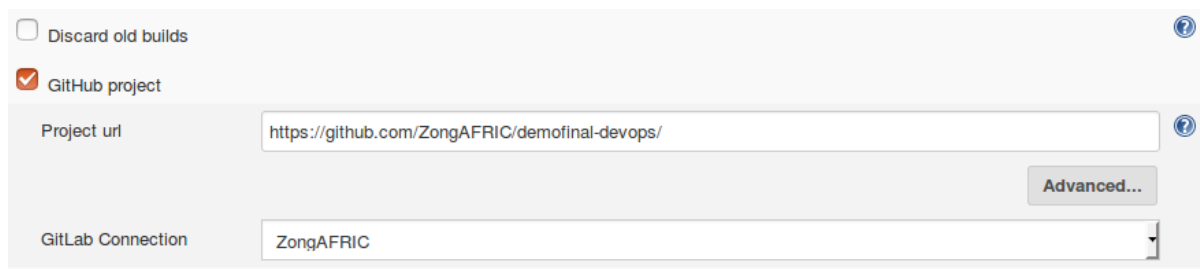
1. `wget -q -O - https ://pkg.jenkins.io/debian/jenkins.io.key | sudo apt-key add -`
2. `sudo sh -c 'echo deb http ://pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources.list.d/jenkins.list'`

### 5.5.2 Etape 2 : Install Jenkins on Ubuntu

1. `sudo apt-get update`
2. `sudo apt-get install jenkins`
3. <https://www.digitalocean.com/community/tutorials/how-to-install-jenkins-on-ubuntu-16-04> Guide d'installation

### 5.5.3 La configuration de plugin Github dans jenkins et Github

#### Côté Jenkins



☐ Discard old builds

☒ GitHub project

Project url

GitLab Connection

Advanced...

FIGURE 8 – *github*

## Source Code Management

☐ None

☒ Git

Repositories

Repository URL

Credentials

Branches to build

Branch Specifier (blank for 'any')

Repository browser

FIGURE 9 – github

## Coté github :configuration du webhook

<> Code ⓘ Issues 0 🔄 Pull requests 0 📁 Projects 0 📖 Wiki 📊 Insights ⚙ Settings

Options

Collaborators

Branches

**Webhooks**

Integrations & services

Deploy keys

Moderation

Interaction limits

Webhooks / **Add webhook**

We'll send a POST request to the URL below with details of any subscribed events. You can also specify which data format you'd like to receive (JSON, x-www-form-urlencoded, etc). More information can be found in [our developer documentation](#).

**Payload URL \***

**Content type**

**Secret**

**Which events would you like to trigger this webhook?**

☒ Just the push event.

☐ Send me **everything**.

☐ Let me select individual events.

☒ **Active**

We will deliver event details when this hook is triggered.

FIGURE 10 – webhook

### Build Triggers

- ☒ Build whenever a SNAPSHOT dependency is built
- ☒ Schedule build when some upstream has no successful builds
- ☐ Trigger builds remotely (e.g., from scripts)
- ☐ Build after other projects are built
- ☐ Build periodically
- ☐ Build when a change is pushed to GitLab. GitLab webhook URL: http://localhost:8080/project/devops-demo
- ☐ GitHub Branches
- ☐ GitHub Pull Requests
- ☒ GitHub hook trigger for GITScm polling
- ☐ Gitlab Merge Requests Builder
- ☐ Poll SCM

FIGURE 11 – *webhook*

## Configuration de message jenkins

Jenkins > configuration

### E-mail Notification

SMTP server: smtp.gmail.com

Default user e-mail suffix: @gmail.com

☒ Use SMTP Authentication

User Name: onezongos@gmail.com

Password: .....

☒ Use SSL

SMTP Port: 465

Reply-To Address:

Charset: UTF-8

☐ Test configuration by sending test e-mail

### GitHub Pull Requests

Published Jenkins URL:

Actualise local repo on factory creation: ☐

**Save** **Apply**

FIGURE 12 – *message*

☒ E-mail Notification

Recipients:

☒ Send e-mail for every unstable build

☐ Send separate e-mails to individuals who broke the build

☒ Send e-mail for each failed module

FIGURE 13 – message

## 5.6 Automatisation des tests

Jenkins est excellent dans l'analyse des résultats des tests. Cependant, c'est à nous d'écrire les tests appropriés et de configurer notre script de build pour qu'il les exécute automatiquement.

Il y a de nombreux outils de tests unitaires, avec la famille xUnit occupant une place prépondérante. Dans le monde Java, JUnit est le standard de facto, bien que TestNG soit aussi un framework de test unitaire populaire avec un certain nombre de fonctionnalités innovantes. Pour les applications C, le framework NUnit propose des fonctionnalités similaires à celles fournies par JUnit, comme le fait Test : :Unit pour Ruby. Pour C/C++, il y a CppUnit, et les développeurs PHP peuvent utiliser PHPUnit.

Ces outils peuvent aussi être utilisés pour les tests d'intégration, les tests fonctionnels et les tests web. De nombreux outils de tests web, comme Selenium, WebDriver, et Watir, génèrent des rapports compatibles xUnit. Les outils Behaviour-Driven Development et de tests d'acceptation automatisés comme easyb, Fitnessse, Concordion sont aussi orientés xUnit. Il serait important de faire la distinction dans les tâches de build afin d'obtenir un compte-rendu plus rapide, vos tests devront être groupés dans des catégories bien définies, en commençant par les rapides tests unitaires, ensuite viendront les tests d'intégration, pour finir par exécuter les tests fonctionnels et web, plus longs.

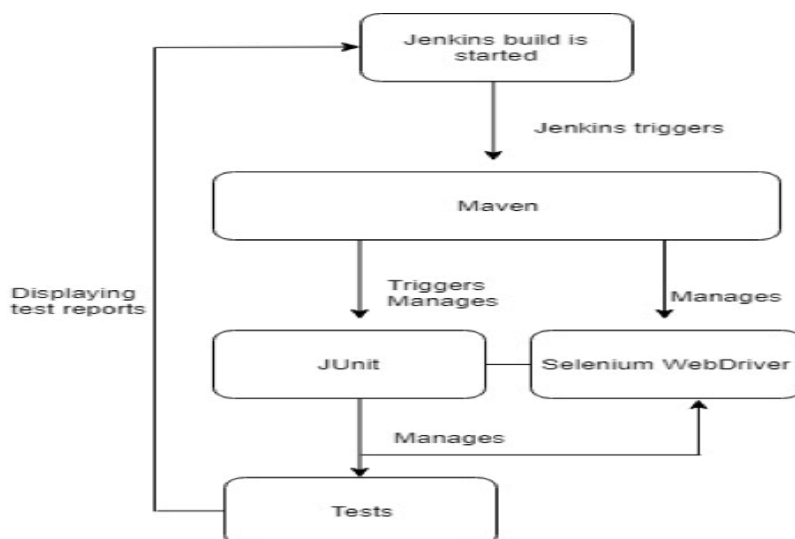


FIGURE 14 – schéma des Tests

### 5.6.1 Les tests unitaires

L'automatisation des tests unitaires permet d'effectuer plusieurs vérifications rapidement, mais aussi de répéter cette opération à l'infini. Cela, tout en garantissant le bon fonctionnement des différentes briques logicielles de manière unitaire.

La valeur ajoutée des tests unitaires automatisés réside dans le choix des tests implémentés que l'on choisit d'implanter. En effet, il est intéressant de pouvoir tester du code complexe rapidement et de manière automatisée.

Il sera aussi possible de tester de manière automatique les bugs connus d'une application afin de ne pas reproduire ces derniers. Cela permet en effet de diminuer les bugs d'une application.

- Aller sur le tableau de bord de jenkins
- Choisir le job sur lequel on veut l'appliquer
- Aller dans configurer
- Option Build
- Add Build step
- Invoke Ant
- Ajout de location du fichier build.xml

## 5.7 Heroku

- Créer un compte heroku
- Créer une application de la plateforme

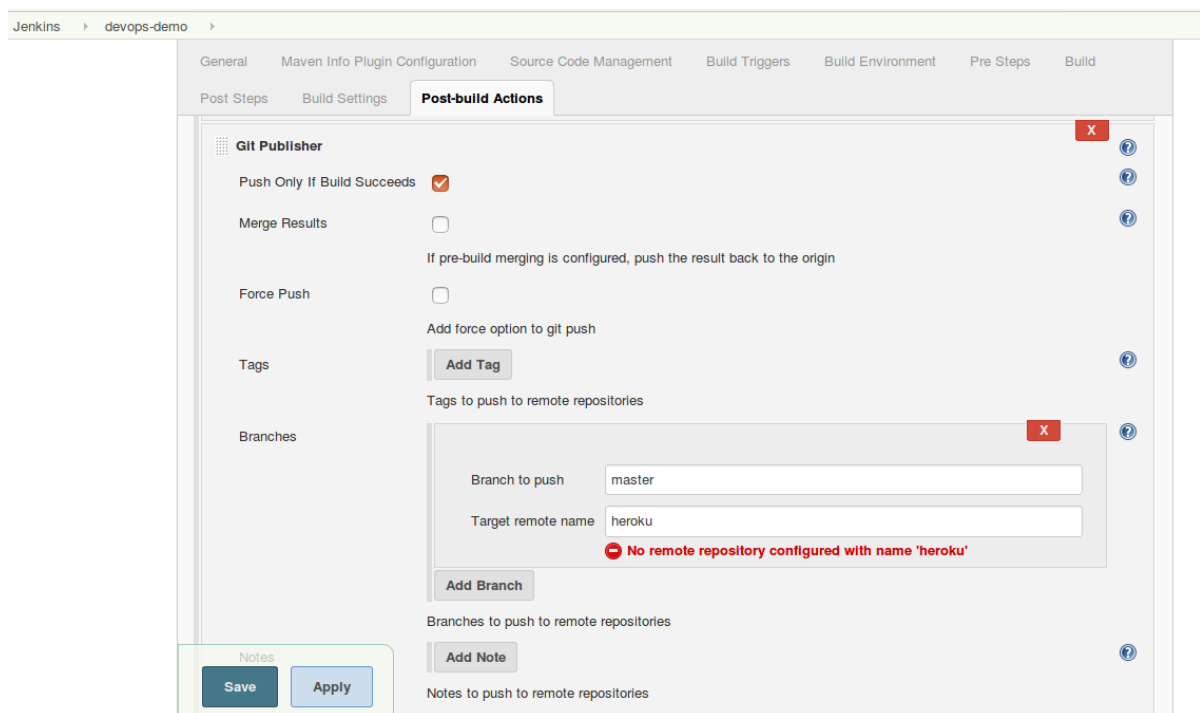


FIGURE 15 – heroku

## La structure du programme

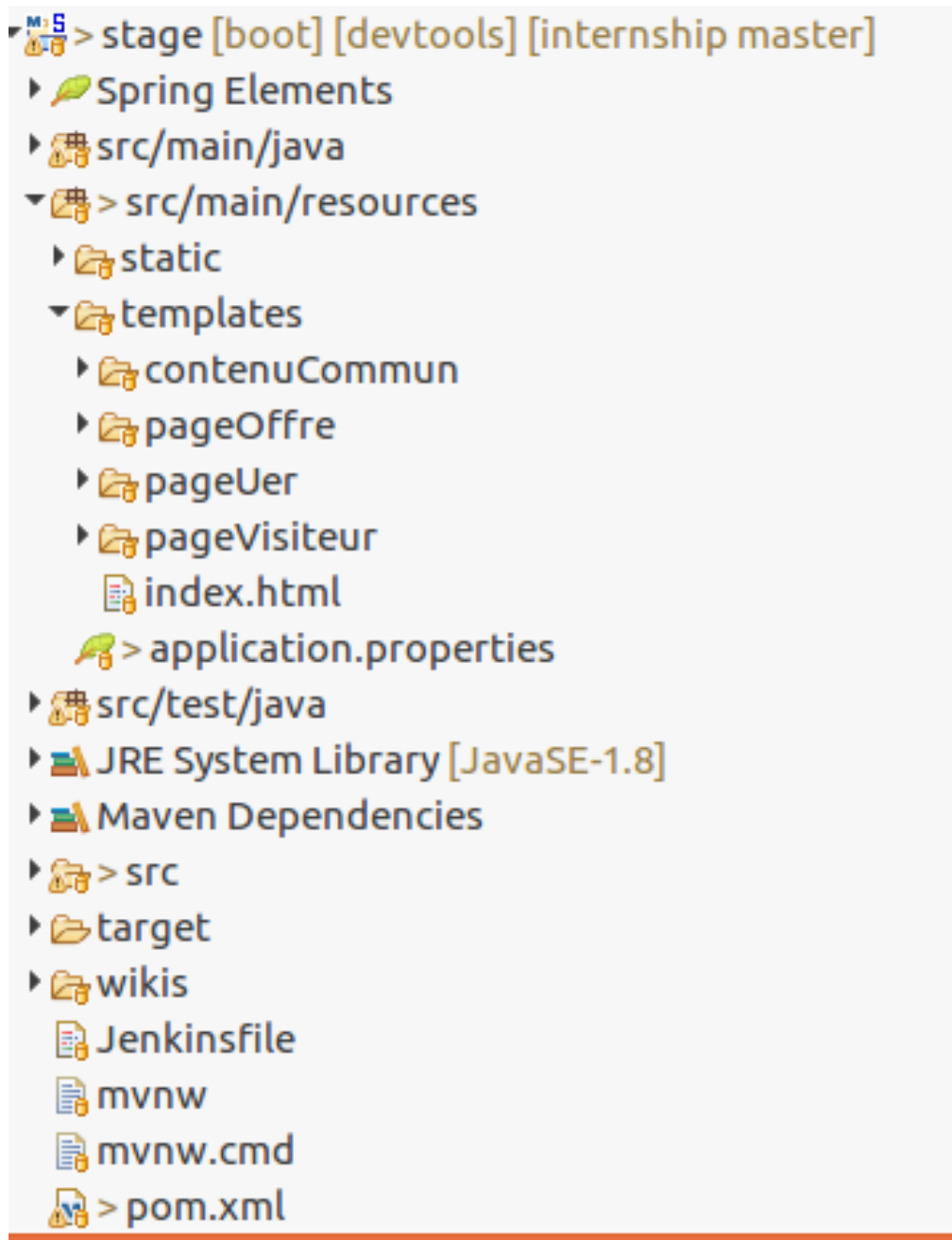


FIGURE 16 – heroku

### Créer un fichier Jenkinsfile à la racine pour le pipeline

Dans ce fichier nous avons de haut en l'ordre d'exécution des tâches définies la la pipeline :

1. mvn clean
2. mvn install
3. mvn test (test unitaire avec jUnit)
4. mvn package (création des artefacts)

5. pushToHeroku cloudSpace : 'test-space', credentialsId : 'zongo', organization : 'test-org', target : 'heroku'

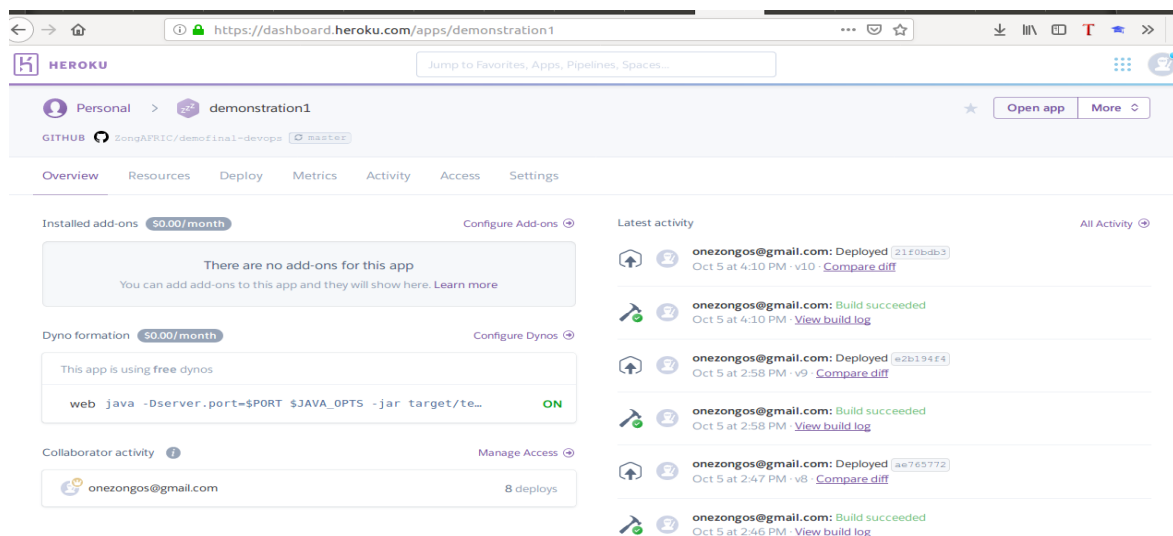
```
pipeline{
  agent any
  tools {
    maven 'M3'
  }
  stages{
    stage('-----clean-----'){
      steps{
        sh "mvn clean"
      }
    }
    stage('-----install-----'){
      steps{
        sh "mvn install"
      }
    }
    stage('-----test-----'){
      steps{
        post {
          always {
            junit 'target/surefire-reports/*.xml'
          }
        }
      }
    }
    stage('-----package-----'){
      steps{
        sh "mvn package"
      }
    }
    stage('build and push'){
      steps{
        sh 'mvn clean install'
        /* generer à partir de pilipe syntax pour le deployment sur Heroku*/
        pushToHeroku cloudSpace: 'test-space', credentialsId: 'zongo', organization: 'test-org', target: 'heroku'
      }
    }
  }
}
```

FIGURE 17 – contenu de jenkinsfile

## 6 Déploiement automatisé

L'un des principes fondamentaux du déploiement automatisé est de réutiliser vos fichiers binaires déployables. Il est inefficace et potentiellement dangereux de réaliser un nouveau build pendant le processus de déploiement. En effet, imaginer que vous exécutiez une série de tests unitaire et d'intégration sur une version donnée de votre application, avant de la déployer dans un environnement de test. Si vous reconstruisez le binaire avant de le déployer, le code source peut avoir changé depuis la version testée, et donc vous ne savez pas exactement ce que vous déployez.

Un processus plus efficace est de réutiliser des binaires générés par un build précédent. Par exemple, vous pourriez configurer une tâche de build de façon à exécuter les tests unitaires et d'intégration avant de déployer un fichier binaire (généralement un fichier WAR ou EAR). Vous pouvez faire cela très efficacement en utilisant le plugin Copy Artifact. Ce plugin vous permet de copier un artefact d'un autre espace de travail dans l'espace de travail de la tâche de build courante. Cela, lorsque combiné avec un trigger de build normal ou avec le plugin Build Promotion, vous permet de déployer précisément le fichier binaire que vous avez construit et testé dans la phase précédente.

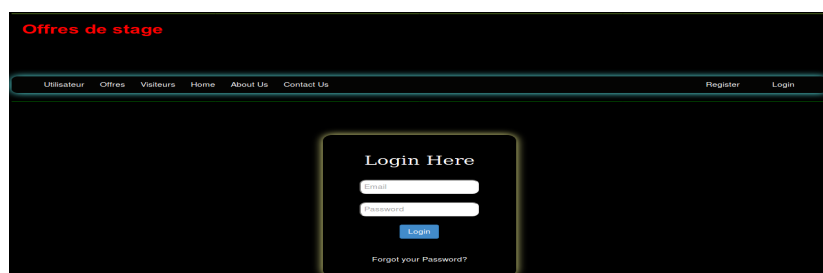


plateforme heroku

## 7 Résultats

Dans cette section du travail nous illustrons les résultats de nos travaux sur Devops, nous avons développé une application en suivant le processus d'intégration continue et de déploiement continu.

### 7.1 Page d'accueil :



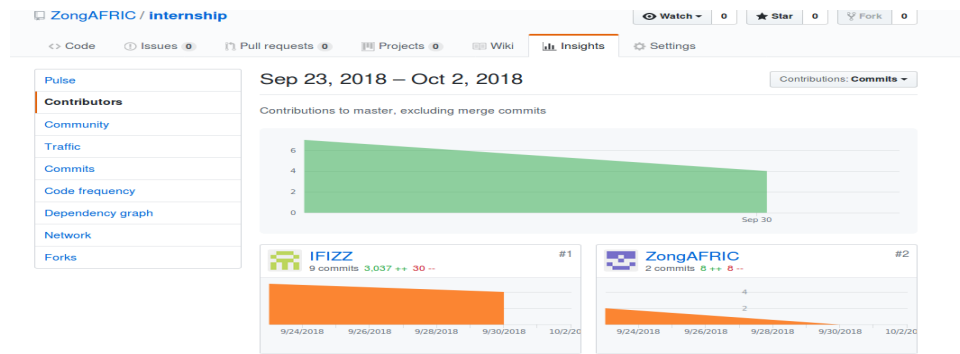
[application]

Cette application nous a permise de tester notre plate forme devops mise en place.

### 7.2 Collaborateurs

Dans la mise en place de notre plate forme DevOps nous avons développé une application java "offre de stage". Nous avons créé deux comptes github avec lesquels nous avons travaillé comme deux membres différents sur le projet afin de mieux comprendre la valeur et le rôle de chaque utilisateur dans la réalisation du projet. Ainsi ce graphe nous permet de voir la participation des membres de l'équipe sur le projet.





collaborateurs

## 7.3 Jenkins-github

Ces deux figures illustrent la récupération automatique de notre application sur le serveur de gestion de version github par le seveur d'intégration jenkins, et l'envoi de message au developpeur en cas d'erreur.

### récupération automatique du code sur github par jenkins

### Envoi de mail par jenkins

## 7.4 Jobs et Builds

Illustration des divers jobs créés et des différents builds construits sur ces jobs. La dernière figure illustre le succès des différentes étapes d'intégration continue , de test continu et de dé-

ploiement continu.

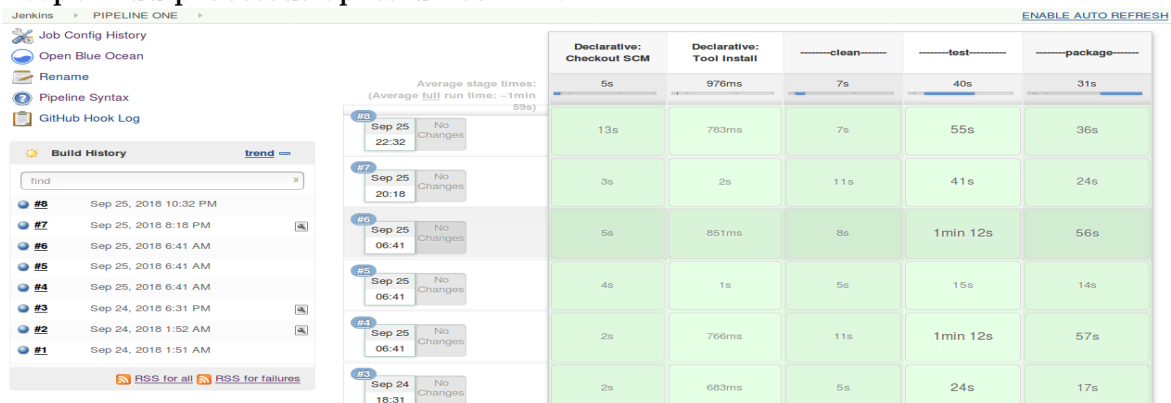
## Étape 1 du processus après un commit

<h1>Demonstration pour DevOps</h1>

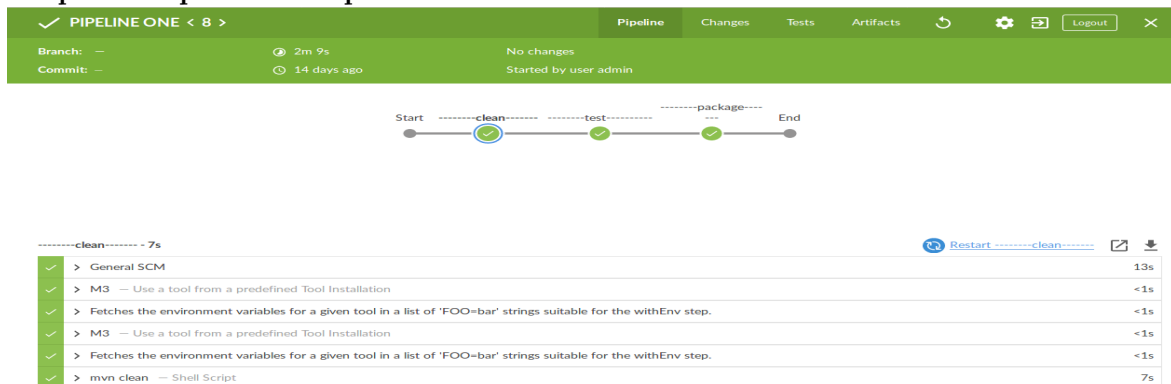
[add description](#)

All						
S	W	Name ↓	Last Success	Last Failure	Last Duration	Fav
		<a href="#">Demo-Build</a>	8 hr 9 min - <a href="#">#1735</a>	4 days 6 hr - <a href="#">#1595</a>	8.3 sec	
		<a href="#">Demo-message</a>	28 min - <a href="#">#2353</a>	4 days 4 hr - <a href="#">#2326</a>	2.2 sec	
		<a href="#">Demo-run</a>	8 hr 8 min - <a href="#">#2817</a>	4 days 4 hr - <a href="#">#2790</a>	3.3 sec	
		<a href="#">Demo-tpe</a>	N/A	10 days - <a href="#">#19</a>	1 min 15 sec	
		<a href="#">devops-demo</a>	N/A	8 hr 13 min - <a href="#">#8</a>	6.9 sec	
		<a href="#">internship</a>	N/A	8 hr 10 min - <a href="#">#2</a>	20 ms	
		<a href="#">JavaJob</a>	8 hr 10 min - <a href="#">#1</a>	N/A	19 ms	
		<a href="#">PIPELINE ONE</a>	6 days 18 hr - <a href="#">#8</a>	N/A	2 min 8 sec	

## Étape 2 du processus après un commit



## Étape 3 du processus après un commit

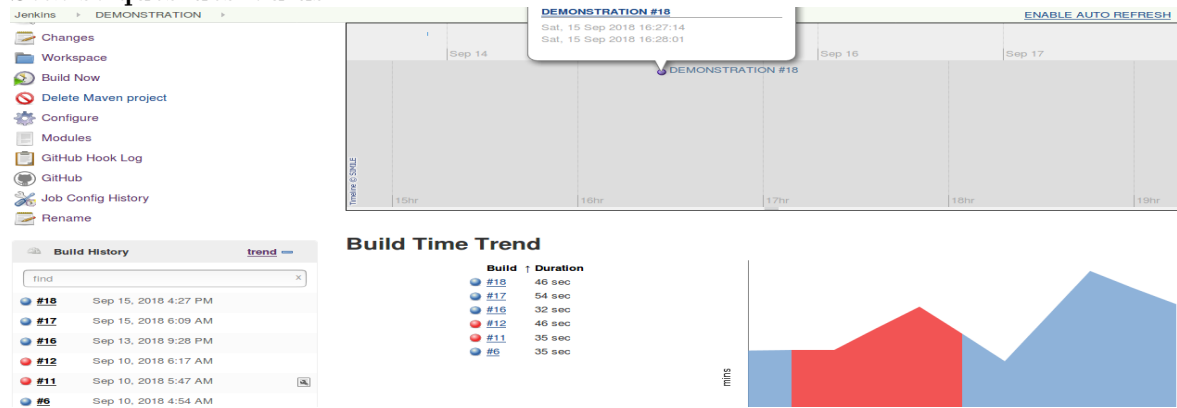


## 7.5 Modification et statistique

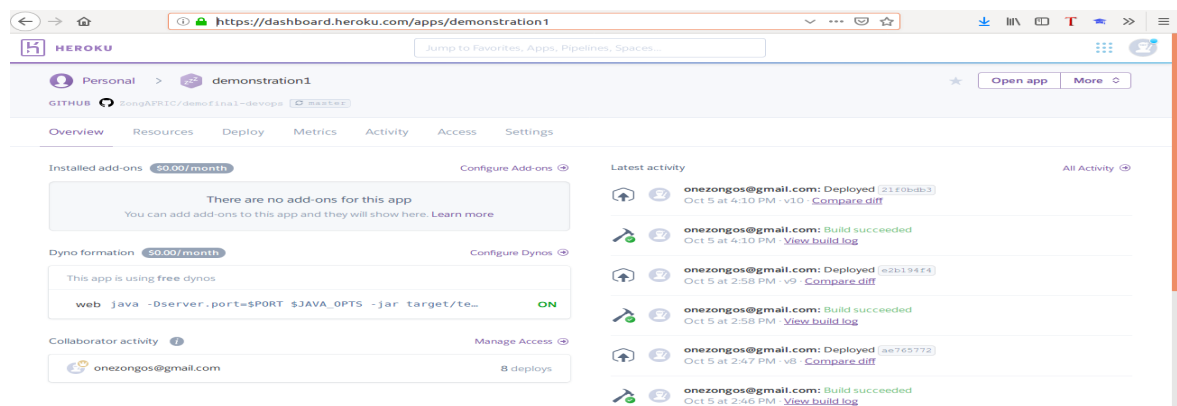
Cette première figure illustre la différence entre une ancienne configuration et une nouvelle et permet de faire la restauratuion en cas de problème dans la nouvelle configuration. Il est rendu possible à l'aide du plugin job configuration. La deuxième image montre les statistiques des divers builds effectués et nous donne une proportion graphique sur le builds pour voir la santé de notre code. Les graphes de couleur rouge représentent les builds echoués et ceux de couleur bleue les builds effectués avec succès.

Jenkins > PIPELINE ONE > Job Config History		ENABLE AUTO REFRESH	
Operation: Changed User: admin		Operation: Changed User: SYSTEM	
Restore this configuration		Restore this configuration	
<pre> 13 &lt;displayName&gt;PIPELINE ONE&lt;/displayName&gt; 14 &lt;keepDependencies&gt;false&lt;/keepDependencies&gt; 15 &lt;properties&gt; 16   &lt;com.coravy.hudson.plugins.github.GithubProjectProperty 17     plugin="github@1.29.2"&gt; 18     &lt;projectUrl&gt;https://github.com/IFIZZ&lt;/projectUrl&gt; 19   &lt;/com.coravy.hudson.plugins.github.GithubProjectProperty&gt; 20   &lt;com.dabsquared.gitlabjenkins.connection.GitLabConnectionProperty 21     plugin="gitlab-plugin@1.5.9"&gt; 22     &lt;gitLabConnection&gt;ZongAFRIC&lt;/gitLabConnection&gt; 23   &lt;/com.dabsquared.gitlabjenkins.connection.GitLabConnectionProperty&gt; 24 &lt;/org.jenkinsci.plugins.workflow.job.properties.PipelineTriggersJobProperty&gt; 25 &lt;triggers&gt; 26   &lt;com.cloudbees.jenkins.GithubPushTrigger plugin="github@1.29.2"&gt; 27   &lt;/com.cloudbees.jenkins.GithubPushTrigger&gt; 28 &lt;/triggers&gt; 29 &lt;/org.jenkinsci.plugins.workflow.job.properties.PipelineTriggersJobProperty&gt; </pre>		<pre> 13 &lt;displayName&gt;PIPELINE ONE&lt;/displayName&gt; 14 &lt;keepDependencies&gt;false&lt;/keepDependencies&gt; 15 &lt;properties&gt; 16   &lt;com.coravy.hudson.plugins.github.GithubProjectProperty 17     plugin="github@1.29.2"&gt; 18     &lt;projectUrl&gt;https://github.com/IFIZZ&lt;/projectUrl&gt; 19   &lt;/com.coravy.hudson.plugins.github.GithubProjectProperty&gt; 20   &lt;com.dabsquared.gitlabjenkins.connection.GitLabConnectionProperty 21     plugin="gitlab-plugin@1.5.9"&gt; 22     &lt;gitLabConnection&gt;ZongAFRIC&lt;/gitLabConnection&gt; 23   &lt;/com.dabsquared.gitlabjenkins.connection.GitLabConnectionProperty&gt; 24 &lt;/org.jenkinsci.plugins.workflow.job.properties.PipelineTriggersJobProperty&gt; 25 &lt;triggers&gt; 26   &lt;com.cloudbees.jenkins.GithubPushTrigger plugin="github@1.29.2"&gt; 27   &lt;/com.cloudbees.jenkins.GithubPushTrigger&gt; 28 &lt;/triggers&gt; 29 &lt;/org.jenkinsci.plugins.workflow.job.properties.PipelineTriggersJobProperty&gt; </pre>	

## Statistiques des bulds



## 7.6 Déploiement sur la plate forme heroku



## 7.7 Déploiement sur la plate forme heroku



## 7.8 Conclusion et perspectives

Ce projet nous a conduit à la mise en place d'un processus DevOps à travers l'intégration de plusieurs outils pour pour l'automatisation du processus complet de developpement que :

- L'integration continue des codes
- Les tests automatiques
- Le deployment automatique des artefacts

Cependant cette solution peut toujours être améliorée en ajoutant certains concepts tels que la mise en place de slaves pour ses multiple avantages.

## Conclusion Générale

Tout au long de ce travail pratique encadré nous avons fait nous avons d'abord analysé et l'état de l'art du sujet, ensuite nous avons fait une étude bibliographique et enfin notre solution proposée pour la résolution du problème. Grâce aux pratiques Agile et DevOps c,les clients sont informés de l'existence d'une mise à niveau automatisée disponible pour toutes les personnes associées à un projet. Nous avons mis en place notre solution proposée à l'aide des outils que sont github, jenkins et heroku.Nous sommes parvenus mettre notre solution en oeuvre, cependant des points restent à améliorer tels : **l'utilisation de docker pour la configurations des slaves avec jenkins pour une meilleure utilisation comme :**

- L'excétion de multiple job en parallèle
- la mise en place du load balancing

Nous avons constacté la difficulté de mettre en place une telle plateforme, de part l'installation et la configuration des certains outils. Nous retenons que les pratiques DevOps sont intégrées dans de nombreuses entreprises de grandes renommées telles microsoft, Amazone, Google, Heraka et bien d'autres pour leur valeur apportée à l'amélioration de la qualité du produit délivré mais aussi pour la réduction du delai de livraison.

## Références

- [1] <https://dzone.com/articles/what-is-devops-the-beginners-guide-from-logzio>, consultée le 20/04/2018.
- [2] Michael Azoff. Ovum decision matrix : Selecting a devops release management solution, 2016–17. *Ovum Software Solutions*, mar, 2016.
- [3] CodeChef. Devops and microsoft. <https://jenkins-le-guide-complet.github.io/continuous-integration-with-hudson.pdf>, 2018-05-01, consultée le 01/05/2018.
- [4] CodeChef. Devops and microsoft. <https://www.visualstudio.com/learn/what-is-devops/>, 2018-05-01, consultée le 01/05/2018.
- [5] e Sanjeev Sharma et Bernie Coyne. Devops pour les nuls, 2016–17. *2é édition limitée, IBM*, 2016.
- [6] Jérémy GROSS. Le devops en pratique. <https://blog.squad.fr/digital-factory/le-devops-en-pratique-2.html>, publié le 27 Juin 2017, consultée le 15/05/2018.
- [7] Michael Httermann. *DevOps for developers*. Apress, 2012.
- [8] Michael Hüttermann. Introducing devops. In *DevOps for Developers*, pages 15–31. Springer, 2012.
- [9] Lucy Ellen Lwakatare, Pasi Kuvaja, and Markku Oivo. Dimensions of devops. In *International Conference on Agile Software Development*, pages 212–217. Springer, 2015.
- [10] par Stefan Thorpe. Aborder la théorie des contraintes avec devops. <https://dzone.com/articles/addressing-the-theory-of-constraints-with-devops>, 15 mars 2018 · DevOps Zone · Didacticiel, consultée le 05/05/2018.
- [11] adaptation Jean Elyan Paul Rubens CIO/IDG. Pourquoi ansible est devenu le favori des devops. <https://www.lemondeinformatique.fr/actualites/lire-pourquoi-ansible-est-devenu-le-favori-des-devops-68328.html>, publié le 26 Mai 2017, consultée le 08/05/2018.
- [12] Mark Paulk. Capability maturity model for software. *Encyclopedia of Software Engineering*, 1993.