Introduction

Table des matières I- Introduction.................................................................................................3 A)Présentation de Haute Autorite de Santé- HAS

Mission................................................................................3 B)La région Gfi Ouest.................................................................................4 C)Présentation de la mission..................................................................6

II- Présentation du projet............................................................................7

A)Contexte professionnel du projet.....................................................7

B)Proposition d’un Architecture CI/CD pour HAS es / limites...............................9

III- Démarche suivie ...................................................................................10

A)Processus suivi – méthode................................................................10

B)Organisation du projet.......................................................................13

IV- Réalisation de la mission....................................................................14

A)Justification de solution retenue....................................................14

B)Étapes essentielles................................................................................16

V- Évaluation des réalisations.................................................................21 VI- Conclusion...............................................................................................23

Annexe : Organisation de Gfi Informatique.......................................24

1. Introduction

Ce stage de deuxième année, dans le cadre du BTS SIO, s’effectue au sein du Groupe Gfi (désigné Gfi dans la suite du rapport), pour une durée de six semaines, soit du 04 janvier au 12 février. Il s’agit d’un stage en développement informatique correspondant à mon choix de la branche SLAM (Solutions Logicielles et Applications Métiers) pour ce BTS. A)

Introduction générale L’avènement de l’entreprise numérique implique la révision profonde des modes de création d’application. Il n’est plus possible de patienter 6 mois pour les livrables de développement. Avec ces exigences de temps au marché et l’évolution des projets dans des configurations logicielles et d’infrastructure de plus en plus complexes, générant des risques opérationnels et de planification, les besoins d’industrialiser les tests et de fluidifier les déploiements en production se sont progressivement affirmés. En rapprochant les équipes de développement, de test et d’exploitation, le DevOps répond précisément à ce défi du digital. Les entreprises en ont désormais bien conscience. Parmi les pratiques les plus répandues de DevOps, on trouve la livraison continue et la gestion de configuration. La livraison continue est une stratégie logicielle qui permet aux organisations d’offrir de nouvelles fonctionnalités aux utilisateurs rapidement et efficacement. L’idée de base de livraison continue est de créer un processus reproductible et fiable d’amélioration progressive pour amener le logiciel du concept au client. L’objectif de la livraison continue est de permettre un flux constant de changements vers la production via une ligne de production automatisée de logiciels. Le pipeline de livraison continue est ce qui rend tout cela possible. C’est dans ce cadre s’inscrit le présent projet, qui vise à améliorer un pipeline de livraison continue en intégrant Ansible, un outil de gestion de configuration et d’automatisation. Ce projet s’est déroulé au sein de l’entreprise SQLI, qui vise à régler plusieurs problématiques. Le présent rapport décrit l’essentiel du travail réalisé lors de ce projet, il est organisé en quatre chapitres : Dans le premier chapitre, nous présentons l’entreprise d’accueil SQLI, après nous explorons la démarche DevOps chez SQLI. Puis, nous poursuivons avec une étude de l’existant dont en déduisent les problèmes. Après, nous exposons notre solution et la démarche suivie pour la gestion de projet. Dans le deuxième chapitre, nous présentons une étude comparative des outils DevOps les plus répandus sur le marché, ainsi que l’outil choisi dans chaque catégorie. Dans l’avant-dernier chapitre, analyse et conception, nous exposerons les besoins fonctionnels, les acteurs, les cas d’utilisations et l’architecture physique des environnements constituants du pipeline. Le dernier chapitre est consacré pour présenter les tâches réalisées pour implémenter notre projet. A. Présentation de HAS

Introduction Ce chapitre a pour objectif de situer notre travail par rapport à son cadre général. Nous commençons tout d’abord par la présentation de l’entreprise d’accueil SQLI dans lequel s’est déroulé notre stage, puis nous passons par une étude de l’existant qui permettra de faire une évaluation et critique de ce dernier. Enfin nous présenterons nos solutions, ainsi que la démarche de travail suivie. 1. Entreprise d’accueil Nous donnons dans la première partie de ce chapitre quelques informations sur le groupe SQLI et nous décrivons ses secteurs d’activités et ses différents pôles. Ainsi qu’une vue organisationnelle de SQLI Maroc et une description du projet AMC.

1. Environnement et contexte du stage
2. Environnement
3. La Haute Autorité de Santé

La Haute Autorité de Santé (HAS) a été créée en 2004 par la loi relative à l’assurance maladie. Elle est dirigée par un collège constitué de 8 membres dont le président est nommé par le président de la République. Le Pr Dominique Le Guludec est la présidente actuelle, elle a été nommée le 4 décembre 2017. Chaque membre du collège préside une commission chargée de secteurs / missions différentes.

Les principales missions dont la HAS est chargée sont les suivantes :

* Elle évalue les services rendus à la société par les produits, prestations et technologies de santé, et donne un avis sur leur potentiel remboursement par la sécurité sociale.
* Elle établit des recommandations de bonnes pratiques pour les professionnels du sanitaire, du social et du médico-social avec pour objectif de valoriser et améliorer la pertinence des actes de soins. Elle produit également des recommandations vaccinales et de santé publique.
* Enfin, la HAS réalise la certification des établissements de santé dans le but d’assurer l’égalité de la qualité des soins et de la sécurité des patients dans l’ensemble des hôpitaux, cliniques, mais aussi dans les structures sociales et médico-sociales depuis 2019.

Les travaux de la HAS sont utilisés par les ARS en qualité d’avis et de conseil sur le financement et la gestion des établissements de santé, la prise en charge des produits de santé par la sécurité sociale…

Actuellement, la HAS compte 415 salariés mais ses activités reposent sur plusieurs milliers d’experts et professionnels de santé externes, en particulier dans le processus de certification.

La HAS a défini six priorités stratégiques à l’horizon 2024 :

* Faire de l’innovation un moteur de l’action de la HAS et en favoriser l’accès sécurisé
* Faire de l’engagement des usagers une priorité
* Promouvoir des parcours de santé et de vie efficients
* Mieux intégrer la pertinence et les résultats pour l’usager dans les dispositifs d’évaluation de l’offre de soins et de l’accompagnement
* Renforcer l'efficience de la HAS
* Renforcer l’influence et la présence de la HAS à l’international

La HAS avait en 2020 un budget de 55.79 millions d’euros pour effectuer toutes les missions dont elle est chargée en respectant ses valeurs clés que sont la rigueur scientifique, l’indépendance ainsi que la transparence.

En 2020, la HAS a rendus 491 avis sur des médicaments, 253 sur des dispositifs médicaux ainsi que 32 sur des actes professionnels en vue d’un remboursement. Elle a également rendu quelques avis sur des nouvelles technologies éligible au forfait innovation.

La HAS a publié une centaine de recommandations sur des pratiques cliniques, sur l’accompagnement social et médico-social ainsi que sur des recommandations vaccinales.

En 2020, le système de certification a entièrement été refondé, et plus de 1800 médecins ont été accrédités.

La HAS en bref

ARTICLE HAS - Mis en ligne le 29 oct. 2020

Autorité publique indépendante à caractère scientifique, la Haute Autorité de santé (HAS) vise à développer la qualité dans le champ sanitaire, social et médico-social, au bénéfice des personnes. Elle travaille aux côtés des pouvoirs publics dont elle éclaire la décision, avec les professionnels pour optimiser leurs pratiques et organisations, et au bénéfice des usagers dont elle renforce la capacité à faire leurs choix. Elle a été créée par la loi du 13 août 2004 relative à l’Assurance maladie.

Trois missions Valeurs ÉVALUER les médicaments, dispositifs médicaux et actes professionnels en vue de leur remboursement. RECOMMANDER les bonnes pratiques professionnelles, élaborer des recommandations vaccinales et de santé publique. MESURER ET AMÉLIORER la qualité dans les hôpitaux, cliniques, en médecine de ville, et dans les structures sociales et médicosociales.

**Organisation**

* Un Collège de huit membres dont un président
* Des commissions spécialisées
* Des services repartis en cinq directions opérationnelles
* 434 collaborateurs
* 1 638 experts externes
* 71,87 M€ de budget exécuté en 2022

A la HAS, un équilibre entre 3 activités complémentaires

Une image contenant texte, capture d’écran, conception

Description générée automatiquement

**Six priorités stratégiques à l’horizon 2024**

Pour relever les défis du système de santé, la Haute Autorité de santé (HAS) s'est fixée [six priorités stratégiques pour la période 2019-2024](https://www.has-sante.fr/upload/docs/application/pdf/2018-11/projet_strategique_2019-2024.pdf).

* Faire de l’innovation un moteur de l’action de la HAS et en favoriser l’accès sécurisé
* Faire de l’engagement des usagers une priorité
* Promouvoir des parcours de santé et de vie efficients
* Développer la culture de la pertinence et du résultat dans l'évaluation de l'offre de soins et d'accompagnement
* Renforcer l'efficience de la HAS
* Renforcer l’influence et la présence de la HAS à l’international

*A la suite de la crise sanitaire, de l’actualité liée aux questions climatiques et de la réflexion autour du rôle et de l’impact de l'institution dans son écosystème, la HAS a amendé son projet stratégique. Il a été enrichi pour mettre en avant l’importance des enjeux relatifs à l’environnement, aux questions de santé publique et de prévention et à la responsabilité sociale et environnementale (RSE) de la HAS. Ce ajout constitue une première étape qui devra se poursuivre lors de l’élaboration du prochain projet stratégique.*

 Projet de stage – propositioind de solution devops, architecture ci/cd

1.5. Projet HAS Notre travail de fin d’études sera appliqué sur le projet AMC du client ArcelorMittal dans une première étape. Alors que l’objectif final d’étaler notre travail sur l’ensemble des projets chez SQLI Maroc

1. 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. 2.1. 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 [1]. - 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 confronté 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 [cf. Figure 6] - 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 [cf. Figure 7

2.2. Comment démarrer une transformation DevOps ? Après que nous avons vu les qualités d’avoir une combinaison d’agilité et DevOps. Dans ce qui suit, nous montrons la démarche suivie par SQLI pour mettre en œuvre DevOps. L’entreprise vient de définir un modèle de maturité de livraison continue suivant les 5 thèmes [cf. Figure 9] : - Culture et organisation - Design et architecture - Build et déploiement - Test et vérification - Information et reporting De ce modèle ; l’entreprise déduit une vision globale de la livraison continue de valeur [cf. Figure 8] qui se base sur plusieurs pratiques et concepts clés : - Infrastructure en tant que code (IaC) : C’est traiter la configuration d’infrastructure comme code. - Pipeline en tant que code : C’est définir un pipeline comme code, au lieu de le configurer dans un outil d’intégration continue. Nous n’avons cité que les concepts les plus importants pour notre projet de fin d’études. Figure. 8 : Vision DevOps chez SQLI Pour terminer cette partie, nous citons que ce projet de fin d’études entre dans cette démarche et vise à mettre en œuvre le concept de l’infrastructure en tant que code et le pipeline en tant code chez SQLI. 10 Figure. 9 : Modèle de maturité de livraison continue Chapitre 1 : Cadre générale du projet 11 3. Etude de l’existant Dans Cette partie, nous présentons le pipeline de livraison continue existant chez SQLI pour le projet AMC, ainsi que nos constatations sur l’état actuel afin de donner une solution aux différents problèmes. 3.1. Fonctionnement actuel Pour gérer le développement et la livraison de ses projets, l’entreprise SQLI dispose d’un pipeline d’intégration et livraison continue. Ce pipeline se constitue de [cf. Figure 10] : - Gestionnaire de code source GitLab - Serveur d’intégration continue Jenkins - Gestionnaire de dépôt d’objets binaires Nexus - Serveur de qualimétrie logicielle. Figure. 10 : Interaction entre les composants du pipeline initiale Dans cet état, le serveur Jenkins joue un rôle orchestrateur entre les autres outils à travers des tâches. Une tâche est une manière de compiler, tester, empaqueter, déployer ou d’effectuer des actions sur un projet. Les tâches de Jenkins apparaissent sous plusieurs formes : - Compilation et test unitaire d’un projet - Création des rapports qualimétriques pour code source - Empaquetage d’un projet pour une livraison - Déploiement en environnement de production, test, ou développement. Par exemple, le projet AMC sur lequel nous appliquons notre travail a les tâches suivantes [cf. Figure 11] : - AMC\_Build : Compilation et exécution des tests unitaires sur le projet après récupération du code source depuis GitLab. Chapitre 1: Cadre générale du projet 12 - AMC\_Sonar : Analyse de code source et publication du rapport d’analyse sur SonarQube. - AMC\_Deploy : Déploiement des artéfacts de projet dans les environnements (développement, intégration, Test, préparation, production) après leur récupération depuis Nexus. Figure. 11 : Ensembles des tâches pour le projet AMC De plus des environnements où sont installés les outils du pipeline, chaque projet dispose des environnements de déploiements suivants : - Développement : un environnement où le projet est développé et où les tests unitaires sont performés (ordinateur du développeur). - Intégration : le but de cet environnement est de combiner le travail d’une équipe de développeurs sur un projet pour le tester. - Test : cet environnement permet aux testeurs de lancer les tests fonctionnels et les tests de régressions automatisées ou non sur le projet pour valider la qualité. Et de plus, il est utilisé pour reproduire les bogues. - Préparation (Pre-production) : environnement similaire à l’environnement de production pour exécuter les tests d’acceptantes avant le déploiement en production ou pour exécuter des tests de performance. - Production : l’environnement qui permet aux utilisateurs finaux d’interagir avec le projet. 3.2. Critique de l’existant Suite à la vue globale présentée ci-dessus, nous avons constaté les problématiques suivantes que l’entreprise vise à résoudre : - Les tâches Jenkins sont lancées manuellement par les membres d’équipe de projet au lieu d’être lancé automatiquement à chaque changement du code source. - Tous les environnements sont manuellement configurés (perte du temps), ce qui ne permet pas un provisionnement rapide (allocation et configuration automatique) lors d’un événement critique, ou une demande d’un environnement similaire. - La tâche Jenkins de déploiement se base sur un script Shell qui exécute des commandes sur l’environnement cible afin de déployer les artéfacts. Ce script présente plusieurs désavantages : o Difficile à tester et maintenir. o Ne supporte pas des stratégies de déploiement sans indisponibilité. o Impossibilité de gérer les informations sensitives sans divulgation. Chapitre 1: Cadre générale du projet 13 4. Solution proposée Pour remédier aux problématiques citées dans l’étude de l’existant, nous avons proposé après plusieurs discussions avec l’équipe DevOps de : - Utiliser un gestionnaire de configuration, Ansible, pour gère la configuration et la préparation des environnements, ainsi que pour gérer le déploiement des artéfacts. - Regrouper les différentes tâches Jenkins dans une seule tâche qui s’exécute à chaque modification du code source. Figure. 12 : Interaction des composants du pipeline après intégration Ansible L’amélioration du pipeline initial avec l’intégration de l’outil d’automatisation Ansible [cf. Figure 12] fournit plusieurs avantages : - Améliorer et optimiser le processus de déploiement. - Facilité de reprendre ou de créer une réplication d’un environnement dans le cas d’une panne d’infrastructure. 5. Démarche adoptée Dans cette partie, nous exposons la démarche de travail adopté pour la gestion et la réalisation de notre projet. 5.1. Méthode adoptée Afin d’assurer le bon déroulement de notre projet, nous avons opté pour la méthode agile Scrum, pour pouvoir répondre au mieux au besoin exprimé par l’entreprise en minimisant les risques de dépassement des délais dans le but d’arriver à une satisfaction partagée en préservant une bonne conduite du projet. Chapitre 1: Cadre générale du projet 14 Scrum (« la mêlée ») est une méthode agile qui s’applique à la gestion de projet. Son principe est abordé pour la première fois dans un article de Hirotaka Takeuchi et Ikujiro Nonaka intitulé « The New New Product Development Game » publié en 1986 dans la Havard Business Review. Elle a ensuite été développée dans les années 1990 et formalisée par Ken Schwaber et Jeff Sutherland en 1995 [2]. Le principe de cette méthode est de focaliser l’équipe de manière itérative sur les fonctionnalités à réaliser. Le projet est ainsi découpé en modules fonctionnels qui seront réalisés, testés et livrés par séquences itératives appelées « sprint ». Chaque sprint vise à atteindre un but à partir duquel sont choisies les fonctionnalités à implémenter dans cette phase [cf. Figure 13]. Figure. 13 : Vue globale du processus Scrum Ainsi, nous avons adapté cette méthode à notre cas, en répartissant le projet sur des sprints d’une à trois semaines. À la fin de chacune, une réunion a été tenue pour présenter les tâches réalisées durant le sprint en question tout en fixant les tâches et objectifs à atteindre durant l’itération suivante. Chaque fois qu’une différence est constatée pendant l’inspection, nous étions amenés à adapter le processus en question. Nous avons en fait, planifié des réunions hebdomadaires, en parallèle à ces Sprint-Reviews, pour présenter les actions de la période, les difficultés rencontrées ainsi que les actions de la période suivante et ses perspectives. Ceci a permis de bien fixer les objectifs à chaque incrément et de pouvoir les adapter, mais aussi, de pouvoir à tout moment compléter ou modifier la liste des fonctionnalités à réaliser pour les prochains sprints. 5.2. Planification du projet La planification du projet est une phase inéluctable pour le management de projet. Elle permet de définir les travaux à réaliser, fixer les objectifs, coordonner les actions, maitriser les moyens, diminuer les risques, suivre les actions en cours et rendre compte de l’état d’avancement du projet. Nous avons établi en premier lieu un découpage de projet en tâches, Chapitre 1: Cadre générale du projet 15 puis un diagramme de Gantt qui permettront d’identifier les travaux à compléter en traduisant le projet en une liste de tâches à accomplir. 5.2.1. Découpage de projet en Sprint Le projet s’étale sur 5 sprints, le titre de chaque sprint est une fonctionnalité de notre backlog du projet. Numéro Titre sprint Description 1 Étude comparative des outils DevOps Une étude qui vise à valider le choix, montrer les inconvénients et les avantages des outils déjà utilisés chez SQLI. 2 Mise en place du pipeline de livraison continue Installer et configurer les outils choisis lors de l’étude comparative. 3 Gestion de configuration des environnements de projet AMC Développement des « playbooks » et des « rôles » permettant de configurer et provisionner les machines des environnements AMC. 4 Gestion de configuration des environnements du pipeline Développement des « playbooks » et des « rôles » Ansible permettant de configurer et provisionner les machines du pipeline. 5 Réaliser un pipeline en tant que code Regrouper la totalité des tâches Jenkins dans un seul pipeline exécuté automatiquement. Table. 1 : Liste des sprints avec descriptions 5.2.2. Diagramme de Gant Le diagramme de Gantt permet de visualiser les différentes tâches et leurs durées [cf. Figure 14], à citer qu’avant l’initiation de projet, nous avons effectué une formation d’un mois sur les outils DevOps et une autre sur la plateforme e-commerce Hybris. Conclusion Dans ce chapitre, nous avons présenté le cadre général du projet, en mettant en valeur les solutions des problématiques à résoudre, ainsi que la démarche adoptée pour la gestion de projet. Le chapitre suivant mettra plus de lumière sur les concepts clés de la livraison continue et la gestion de configuration

Chapitre 2: Etude comparative des outils DevOps Chapitre 2: Etude comparative des outils DevOps 18 Introduction Pour mettre en place un pipeline de livraison continue, nous avons besoin d’un ensemble d’outils. Mais face à l’abondance des outils dans le marché, choisir la combinaison gagnante reste une tâche laborieuse. Pour cela, une étude comparative est nécessaire pour choisir les meilleurs outils répondant aux besoins techniques de notre projet. L’outil approprié doit comprendre des fonctionnalités adéquates, mais aussi respecte les critères que nous avons étudiés. La démarche suivie et les détails l’étude effectuée sont présentés dans la suite de ce chapitre. 1. Démarche de l’étude comparative Choisir une combinaison des outils DevOps homogène et ayant une communication automatique entre eux est un facteur majeur du succès d’un pipeline de livraison continue. Plusieurs outils existent dans le marché. Le choix, donc, dépend des critères à prendre en considération. Pour bien organiser cette étude, qui constitue une phase préalable à la phase de mise en œuvre du pipeline, nous suivons les étapes suivantes : - Lister les outils les plus connus aux marchés et sélectionner les plus intéressants pour les analyser et étudier. - Élaborer des critères de comparaison pertinents et rigoureux. De plus, nous ne tiendrons pas compte des critères non différenciâtes malgré leur importance. - Comparer les outils selon les critères établis pour sortir avec des tableaux comparatifs, avant d’en faire la synthèse et identifier l’outil approprié. Pour ce qui suit, nous allons présenter, pour chaque catégorie d’outils, l’étude comparative complète qui reflète exactement la démarche que nous venons de décrire. 2. Outils de gestion de code source La gestion du code source (SCM) est la façon dont les changements logiciels sont effectués. Le SCM a un certain nombre d‘objectifs qui portent fondamentalement sur la garantie que les équipes de développement puissent livrer des changements de code de qualité plus rapidement [3]. En améliorant le suivi, la visibilité, la collaboration et le contrôle à travers le cycle de vie des livraisons, les outils de SCM permettent plus de créativité, de liberté et offrent plus d’options aux développeurs travaillant sur des projets complexes. De plus, le SCM peut protéger les fichiers source de toute sorte d’anomalies, et permet à toutes les équipes de savoir qui a effectué quel changement et à quel stade. Cette étude de gestionnaire de code source a pour but de consolider le choix de l’outil GitLab, qui est déjà utilisé en interne, face à ses concurrents BitBucket et GitHub. Chapitre 2: Etude comparative des outils DevOps 19 2.1. Choix des critères Les critères qui vont orienter notre choix d’outil de gestion de code source sont : - Open Source : le code source du logiciel est accessible. - Licence : gratuit ou payant. - Gestionnaire de version : les gestionnaires de version supportés. - Fonctionnalité intégration continue : possibilité de lancer des compilations et teste de code sans serveur d’intégration extérieur. - Suivi de problèmes : Suivre les incidents et les anomalies d’un projet sans recourir à un autre outil. 2.2. Analyse 2.2.1. GitLab GitLab est un gestionnaire web de référentiel git incluant un wiki et des fonctionnalités de suivi de problèmes. GitLab fournit une gestion centralisée des référentiels Git, permettant aux utilisateurs d’avoir le contrôle complet de leurs référentiels ou projets [3]. Écrit en Ruby (avec des compléments Go), le gestionnaire inclut contrôles d’accès granulaires, revues de code, suivi de problèmes, flux d’activités, wikis, et intégration continue. En décembre 2016, il regroupe 1400 contributeurs Open Source et est utilisé par les grandes entreprises comme Sony, IBM, CERN, NASA, etc. 2.2.2. GitHub GitHub est un service de référentiel web hébergé offrant toutes les fonctionnalités de gestion de code source, tout en garantissant un espace aux développeurs pour stocker leurs projets et concevoir des logiciels en parallèle. GitHub fournit des fonctions de collaboration, de contrôle d’accès, des Wikis et des outils simples de gestion de tâches pour projets [3]. Conçu par des développeurs pour les développeurs, GitHub propose une interface graphique et un bureau web ainsi qu’une intégration mobile. Il ne se borne pas au développement logiciel : son aspect ouvert et « réseau social » est fondamental. Il permet de faire une copie du projet public d’une autre personne et de modifier ses fonctionnalités tout en visualisant le travail et les profils de chacun. 2.2.3. BitBucket BitBucket est un service web hébergé pour projets logiciels utilisant le système de contrôle de version Mercurial ou git, et propose des plans commerciaux ou gratuits. Ces derniers incluent un nombre illimité de répertoires privés et jusqu’à 5 utilisateurs. Son élasticité simplifie la collaboration des équipes. L’extraction de requêtes, les permissions de branches et les discussions en ligne sont des fonctionnalités clés. Écrit en Python, en utilisant le Framework web Django, BitBucket permet aux équipes de livrer et de partager du code de meilleure qualité, plus rapidement. Utilisé pour son extrême élasticité, particulièrement dans Chapitre 2: Etude comparative des outils DevOps 20 un environnement commercial, il garantit aux développeurs, vitesse et fiabilité quel que soit l’endroit où le projet [3]. 2.3. Comparaison et synthèse Le tableau synthétise la comparaison des gestionnaires de code source présentés préalablement, selon les critères établis et exposés auparavant. Outil Critère Open Source Oui Non Non Licence Gratuit pour la version communauté Payant Payant Gestionnaire de version Git Git Git, Mercurial Fonctionnalité intégration continue Intégré ou à travers d’autres outils À travers d’autres outils seulement À travers d’autres outils seulement Suivi de problèmes Oui Oui À travers un autre outil Jira Table. 2 : Récapitulatif de comparaison des outils de gestion de code source L’outil GitLab est open source, de plus gratuit pour la version communauté. Cette raison nous a poussées de le choisir sans hésitation. D’une autre, son intégration de fonctionnalité d’intégration continue et suivi de problèmes permettra dans le futur d’abonder l’utilisation des outils à part. 3. Outils d’intégration continue Les logiciels d’intégration continue (CI) permettent aux développeurs de faire un commit de code vers un plus grand référentiel, aussi souvent qu’ils le souhaitent. Les outils construisent et testent le code de façon à ce que toute erreur ou tout bug soit détecté rapidement et transféré au développeur pour résolution [4]. Enfin, le CI facilite le processus de livraison logicielle, raccourcissant les cycles de livraison et laissant aux développeurs plus de liberté pour se concentrer sur l’innovation. Il permet à différents développeurs ou équipes de travailler en parallèle sur différents aspects du même projet. Parmi les logiciels d’intégration continue existants sur le marché, nous avons choisi de comparer Jenkins le plus connu, Travis CI et CircleCI. Chapitre 2: Etude comparative des outils DevOps 21 3.1. Choix des critères Les critères qui vont orienter notre choix d’outil d’intégration continue sont : - Open Source : le code source du logiciel est accessible. - Licence : gratuit ou payant. - Installation : Complexité de mettre en place l’outil pour un environnement de production. - Gestionnaire de code source : supporté par défaut ou avec des plug-ins - Système d’exploitation : supporté pour lancer des compilations et des tests de code. 3.2. Analyse 3.2.1. Jenkins Jenkins est un outil d’intégration continu Open Source écrit en Java. Jenkins est un successeur de Hudson. Il supporte les outils SCM tels que Subversion, git, etc. Jenkins peut également exécuter des scripts Shell et des projets Ant ou Maven [4]. Jenkins dispose en outre de nombreux plug-ins qui le rendent compatible avec tous les langages de programmation et une grande majorité de systèmes de contrôle de version et de référentiels. Jenkins permet aux utilisateurs de concevoir et livrer des applications à grande échelle rapidement et supporte conception, déploiement et automatisation dans la plupart des projets. 3.2.2. TravisCI Travis CI est un service hébergé d’intégration continue (IC) en Open Source pour concevoir et tester des projets hébergés sur GitHub. Les exécutions de builds et de tests sont déclenchées automatiquement toutes les fois qu’un commit est réalisé et poussées vers un référentiel GitHub [4]. Travis CI se configure en plaçant un fichier travis.yml dans le répertoire racine de votre référentiel. Travis CI a été conçu pour exécuter tests et déploiements en laissant les développeurs se concentrer sur le code. Cette automatisation facilite le déploiement simple, rapide et agile pour les équipes logicielles. Travis CI est gratuit pour les projets Open Source, payant pour les projets commerciaux ou privés. 3.2.3. CircleCI CircleCI est une plateforme d’intégration et de déploiement continus qui automatise les processus de build, de test et de déploiement. Il permet aux équipes de développement de déployer des projets logiciels rapidement, tout en facilitant l’élasticité [4]. Chapitre 2: Etude comparative des outils DevOps 22 CircleCI est un serveur Cloud hébergé qui réduit considérablement l’effort de test. Il supporte un grand nombre de technologies, comme Ruby on Rails, Sinatra, Node, Python, PHP, Java et Clojure. Il dispose de quatre fonctionnalités : une configuration rapide, une intégration avec un grand nombre d’outils (conférant de la souplesse dans l’environnement de travail utilisateur), il supporte tous les tests et permet de configurer facilement un enchaînement. L’association de ces fonctionnalités permet à l’utilisateur de livrer des projets mieux testés plus rapidement. 3.3. Comparaison et synthèse Le tableau synthétise la comparaison des serveurs d’intégration continue présentés préalablement, selon les critères établis et exposés auparavant. Outil Critère Open Source Oui Oui Non Licence Gratuit Gratuit, il existe une version payante Payant Installation Facile Difficile Aucune installation, SaaS Gestionnaire de code source GitLab, GitHub, Bitbucket, TFS, CVS, Preforce GitHub, Bitbucket GitHub, Bitbucket Système d’exploitation Linux, MacOs, Windows, Unix Linux, MacOs Linux, MacOs Table. 3 : Récapitulatif de comparaison des outils d’intégration continue D’après le tableau de cette étude et les différentes informations collectées, nous sommes arrivés à la conclusion suivante : Jenkins est l’outil le plus adapté pour notre cas, il est populaire et bénéficie d’une vaste communauté qui délivrent des plug-ins ouvrant un large champ de possibilité, contrairement à Travis CI, sans oublier son aspect open source et licence gratuit qui lui donne un pas d’avance par rapport à CircleCI. 4. Outils de qualimétrie logiciels Les outils de Qualimétrie servent à mesurer la qualité de code, la couverture de tests, etc. Ainsi que publier des rapports sur les différents indicateurs obtenus. Une analyse correcte des données et des rapports publiés par la Qualimétrie permettra de mesurer de manière Chapitre 2: Etude comparative des outils DevOps 23 objective la qualité du code et ainsi appréhender la dette technique qui s’accumule et anticiper les bugs à venir [5]. Nous avons tenu de comparer SonarQube le plus complet à ses nouveaux concurrents Scrutinizer et Codacy. 4.1. Choix des critères Les critères qui vont orienter notre choix d’outil qualimétrie sont : - Open Source : le code source du logiciel est accessible. - Licence : gratuit ou payant. - Nombre de langages supportés : nombre de langage que l’outil peut scanner et analyser. - Communauté de support et documentation. 4.2. Analyse 4.2.1. SonarQube SonarQube (précédemment appelé Sonar) est un logiciel libre développée par SonarSource permettant de mesurer la qualité du code source en continu, en effectuant une analyse statique du code pour détecter les bugs, les odeurs de code et les failles de sécurité sur plus de 20 langages de programmation [6]. SonarQube propose des rapports sur le code dupliqué, les normes de codage, les tests unitaires, la couverture de code, la complexité du code, les commentaires, les bogues et les failles de sécurité. SonarQube peut enregistrer l’historique des métriques et fournir des graphiques d’évolution. Aussi, il fournit une analyse et une intégration entièrement automatisées avec Maven, Ant, Gradle, MSBuild et des outils d’intégration continue (Atlassian Bamboo, Jenkins, Hudson, etc.). 4.2.2. Scrutinizer Scrutinizer est une plate-forme d’inspection continue qui aide à créer de meilleurs logiciels. En permettant de mesurer et suivre en continu la qualité du code avec des indicateurs de code simplifiés et faciles à comprendre. De plus, Scrutinizer offre la possibilité d’avoir des commentaires sur les changements de qualité du code entre les versions et recevoir des rapports hebdomadaires sur la qualité du code [7]. 4.2.3. Codacy Codacy est un outil automatisé d’analyse de qualité de code qui aide les développeurs à expédier de meilleurs logiciels, plus rapidement. Avec Codacy, vous obtenez des modifications d’analyse statique, de complexité cyclomatique, de duplication et de couverture de test des unités de code à chaque modification ou ajout de nouveau code. Chapitre 2: Etude comparative des outils DevOps 24 Codacy peut appliquer des normes de qualité de code ce qui permet de gagner du temps dans les révisions de code et appliquer plus rapidement les meilleures pratiques de sécurité et de codage [8]. 4.3. Comparaison et synthèse Le tableau synthétise la comparaison des outils de Qualimétrie logiciel présentés préalablement, selon les critères établis et exposés auparavant. Critère Outil Open Source Oui Non Non Licence Gratuit Payant Payant Nombre de langage supporté 20 7 11 Communauté de support et Documentation Communauté large et documentation claire Communauté assez pauvre, mais bien documenté Communauté pauvre, mais une documentation riche Table. 4 : Récapitulatif de comparaison des outils de qualimétrie logicielle Dans cette étude SonarQube à un grand pas d’avance par rapport à ses concurrents Codacy et Scrutinizer. D’une part, pour son aspect open source et gratuité de licence. D’autre part, pour le grand nombre de langages qu’il supporte et sa large communauté.

6. Outils de gestion de configuration et automatisation Les outils de gestion de configuration sont utilisés pour contrôler les changements, les mises à jour et les modifications sur l’infrastructure d’une entreprise. Les outils gèrent l’équilibre souhaité au sein de grands environnements composés de serveurs, effectuent une supervision en continu, et assurent que l’infrastructure est configurée selon les bonnes spécifications, éliminant les risques et réduisant le temps de résolution des incidents. Ces fonctionnalités facilitent la collaboration à travers l’entreprise en éliminant les problèmes de contrôle de version et en garantissant l’uniformité des modifications. Enfin, les outils de gestion de configuration sont indispensables pour éviter les trop grandes variations entre serveurs et infrastructure, économisant de facto temps et coûts requis pour réparer les erreurs [10]. Dans la catégorie des outils de gestion de configuration trois outils sont les plus connus et utilisés par les entreprises : Ansible, Puppet et Chef. Cette étude vient pour montrer les avantages et les inconvénients de l’outil Ansible qui est proposé pour ce projet face à Puppet et Chef. Chapitre 2: Etude comparative des outils DevOps 27 6.1. Choix des critères Les critères qui vont orienter notre choix d’outil de gestion de configuration et automatisation sont : - Sans Agent : Pouvoir contrôler des machines sans configuration apriori ou installation d’agent. - Langage - Dépendance client : les logiciels pré requis pour configurer une machine. - Mécanisme : de partage de configuration, push ou pull. - Approche de description d’état de machine, déclaratif ou procédural. - Installation : Complexité de mettre en place l’outil pour un environnement de production. - Contributeur : nombre de contributeur dans le code source de l’outil. 6.2. Analyse 6.2.1. Ansible Ansible est un moteur d’automatisation Open Source qui automatise le provisionnement logiciel, la gestion de configuration et le déploiement applicatif [10]. Ansible livre une automatisation IT simple, qui met fin aux tâches répétitives et libère les équipes DevOps pour qu’elles se concentrent sur des tâches plus stratégiques. Ansible facilite le déploiement applicatif, la gestion de configuration et l’orchestration, le tout depuis un système unique. Pour y parvenir, Ansible a été conçu comme un outil simple, fiable, et convivial, contenant un minimum de dépendances. De plus, son architecture sans agents garantit sa sécurité et réduit les frais de réseau. 6.2.2. Chef Chef est un outil de gestion de configuration et une plateforme d’automatisation conçu pour rationaliser les tâches de provisionnement, de configuration et de maintenance des serveurs d’une entreprise. Il transforme l’infrastructure en code, la rendant souple, versionable, lisible et testable, quelle que soit la plateforme où il s’exécute ou la taille du réseau [10]. Écrit en Ruby et Erlang, il peut s’intégrer avec une grande variété de plateformes cloud, y compris Rackspace, Internap, Amazon EC2, Google Cloud Platform, OpenStack, SoftLayer, et Microsoft Azure. Comme son nom l’indique, Chef requiert l’écriture d’une série de recettes décrivant une série de ressources qui doivent se trouver dans un état particulier : les packages devant être installés, les services devant être exécutés, ou les fichiers devant être écrits. Chapitre 2: Etude comparative des outils DevOps 28 6.2.3. Puppet Puppet est un outil de gestion de configuration logicielle Open Source qui assure un moyen standard de livrer et d’exploiter les logiciels, quel que soit l’endroit où ils s’exécutent. L’utilisateur doit déclarer le statut final des applications déployées et de l’infrastructure provisionnée au moyen d’un langage facile d’accès [10]. Compatible avec Unix et Windows, Puppet est livré avec son propre langage déclaratif pour décrire la configuration du système. Il existe deux éditions de l’outil : Open Source et professionnelle. Cette dernière inclut un GUI, des API et des outils en ligne de commande pour la gestion des nœuds. L’objectif de Puppet est de permettre de partager, de tester et d’appliquer des changements à travers un Datacenter, tout en garantissant visibilité et reporting pour la prise de décision et la conformité. Son véritable objectif est de mettre en œuvre une manière élastique et standard d’automatiser le déploiement et la mise en production des applications. 6.3. Comparaison et synthèse Le tableau synthétise la comparaison des gestionnaires de configuration présentés préalablement, selon les critères établis et exposés auparavant. Outil Critère Sans Agent Oui Non Non Langage Python Ruby Ruby Dépendance client Python, sshd, bash Ruby, sshd, bash Ruby Mécanisme Push Pull Pull Approche Procédural Procédural Déclarative Installation Facile Peu facile Difficile Contributeur 3432 522 492 Table. 6 : Récapitulatif de comparaison des outils de gestion de configuration et automatisation L’avantage d’Ansible par rapport aux autres outils de gestion de configuration est son aspect sans agent, c’est-à-dire, il n’a pas besoin d’une configuration a priori pour contrôler une machine. De même, sa communauté est plus grande, et chaque mois de nouvelles options sont ajoutées. Donc, Ansible reste le plus adapté et facile à mettre en œuvre en comparaison avec Chef et Puppet. Chapitre 2: Etude comparative des outils DevOps 29 7. Synthèse et choix À l’issu de chaque étude comparative, nous avons défini notre choix technologique parmi les outils énoncés, mais nous allons récapituler. Comme gestionnaire de code source nous avons opté pour GitLab, et pour l’outil de qualimétrie, note choix c’est porté sur SonarQube. Pour la gestion de configuration, nous avons utilisé Ansible. Et comme serveur d’intégration continue Jenkins. Et pour le dépôt des artéfacts, nous avons gardé Nexus. Le tableau récapitule nos choix. Fonctionnalité Outil Gestionnaire de code source GitLab Serveur d’intégration continue Jenkins Outil de qualimétrie logiciel SonarQube Dépôt d’objet binaire Nexus Gestionnaire de configuration Ansible Table. 7 : Tableau récapitulatif des outils choisis Conclusion Au cours de ce chapitre, nous avons fait une étude comparative des outils à mettre en place dans notre pipeline de livraison continue. Dans un premier temps, nous avons défini la démarchée à suivre, ensuite, nous avons comparé chaque catégorie d’outils pour sortir avec notre choix. Enfin, nous avons fait une synthèse des outils choisis.

Chapitre 3: Analyse et Conception Chapitre 3: Analyse et Conception 31 Introduction À travers ce chapitre, nous entamons l’analyse et la conception qui présente une étape fondamentale qui précède la réalisation. Nous commençons par une description des objectifs du projet et des besoins. Nous procédons ensuite à la définition des acteurs et un cas d’utilisation global du pipeline. Et nous terminons par représenter l’infrastructure du système et l’interaction entre ces composants. 1. Objectifs du projet Les objectifs de ce projet sont : - Mettre en place un pipeline de livraison continue qui supporte la gestion de configuration. - Mettre en œuvre la pratique du pipeline en tant que code, en anglais « Pipeline as Code ». - Mettre en œuvre la pratique de l’infrastructure en tant que code, en anglais « Infrastructure as Code ». Les deux derniers objectifs seront appliqués sur le projet AMC, dans une vision d’étaler ces pratiques sur d’autres projets. 2. Analyse des besoins Dans cette partie, nous commençons par analyser les besoins fondamentaux de notre système à travers la partie besoins fonctionnels. Puis, nous finissions par les besoins non fonctionnels à considérer pour ce système. 2.1. Besoins fonctionnels Un recensement a permis de dévoiler les besoins suivants : - Lancement automatique des tâches Jenkins suite à une requête de fusion GitLab. - Notification d’utilisateur en cas d’échoue d’une tâche Jenkins. - Supporter la compilation et la construction de projet avec des systèmes d’exploitation différents. - Authentification centralisée des utilisateurs à travers LDAP. - Automatisation de déploiement avec Ansible. - Configuration des environnements avec Ansible. - Possibilité de livrer n’importe quand. 2.2. Besoins non fonctionnels De plus des besoins fonctionnels déjà cités, notre système de livraison continue doit respecter un nombre de critères donnant une meilleure qualité de la solution obtenu : - Performance : le temps de réponse du système (les différents outils) doit être minimal. Chapitre 3: Analyse et Conception 32 - Extensibilité : le système doit pouvoir supporter l’évolution et l’extensibilité de ses composants, possibilité d’ajouter des nœuds en cas de montée en charge. - Convivialité : les interfaces des outils doivent être intuitives et faciles à comprendre, et permettre aux utilisateurs de réaliser leurs objectifs sans ambigüité. 3. Identification des acteurs Pour une plateforme de livraison continue, nous identifions quatre acteurs : - Développeur : c’est utilisateur qui peut modifier le code source pour un objectif donné (ajout d’une fonctionnalité, correction d’erreur, etc.) et lancer après des tâches Jenkins pour assurer l’intégration continue et la livraison continue du projet. - Testeur : c’est un utilisateur qui lance des tâches Jenkins (normalement de déploiement dans les environnements de test) afin d’assurer la régression et la conformité des changements aux besoins ou pour tester les fonctionnalités du projet. - Chef de projet : c’est un utilisateur qui hérite du développeur, mais il fait aussi le suivi des correctifs et la réalisation des fonctionnalités. - Configurateur : c’est un utilisateur qui peut configurer les environnements d’un projet d’une manière manuelle ou automatique avec Ansible. De plus, il est l’administrateur de la plateforme de livraison continue. 4. Cas d’utilisation global du pipeline Nous voulons donner une version globale du comportement fonctionnel de notre plateforme de livraison continue. Le diagramme de cas d’utilisation ci-dessous liste les cas d’utilisations généraux, ainsi que les acteurs interagissant avec le système. Figure. 15 : Diagramme de cas d’utilisation général Chapitre 3: Analyse et Conception 33 Pour lever l’ambigüité et rendre clairs les cas d’utilisation, nous donnons une description de chaque cas dans le tableau suivant : Cas d’utilisation Description Gérer code source sur GitLab Le développeur peut récupère le code source, et le pousser sur GitLab après modification. Lancer une tâche dans Jenkins Le développeur et le testeur peuvent lancer depuis Jenkins plusieurs tâches : - Compilation et exécution des tests unitaire - Analyse de qualimétrie et publication du rapport - Publication des Artéfacts - Déploiement dans un environnement - Annuler une tâche en cours d’exécution - Ce cas inclut la notification de l’acteur en cas d’erreur. Télécharger les artefacts depuis Nexus Le téléchargement des artéfacts publié récemment pour les tester par les développeurs ou les testeurs. Consulter rapport de qualimétrie sur SonarQube La consultation du rapport de qualimétrie permet au développeur et au chef de projet d’examiner les métriques de qualité logicielle. Administre plateforme livraison continue L’administration des outils du pipeline de livraison continue, contient plusieurs actions : - Gérer les utilisateurs et leurs droits d’accès - Configurer les outils (Authentification LDAP, Notification par mail, etc.) - Gérer les dépôts code source GitLab (Création, Configuration, Suppression) - Gérer les tâches Jenkins (Création, Configuration, Suppression) - Gérer les dépôts d’objet binaire Nexus - Gérer les bornes de qualité (Quality Gate) et les profiles de qualité sur SonarQube. Configurer environnements avec Ansible La configuration des environnements est une étape essentielle avant le déploiement des artéfacts. Avec Ansible, nous créons des « playbooks », leur exécution permet de mettre en places toutes les exigences d’un artéfact. Table. 8 : Description des cas d’utilisations Après cette description, nous rappelons et il est clair d’ailleurs que nous avons essayé de donner une vue générale de l’interaction des acteurs avec notre système sans entrer dans les détails et essayer de recenser tous les cas d’utilisation possibles. Chapitre 3: Analyse et Conception 34 5. Architecture physique globale du pipeline Avant de mettre en place et implémenter le pipeline de livraison continue, nous devons estimer les ressources informatiques nécessaires à allouer. Le diagramme de déploiement cidessous donne une vue globale des nœuds physiques du système et leurs relations. Figure. 16 : Diagramme déploiement pipeline livraison continue Dans le diagramme, nous remarquons que le serveur Jenkins a plusieurs esclaves. Ces esclaves sont dédiés pour lancer des tâches de compilation et de construction des projets. De cette manière, nous pouvons construire des projets sur des systèmes d’exploitation différents (Linux, Windows, MacOs). Et nous pouvons étendre notre système en cas d’une monté en charge sur les esclaves par une simple réplication d’un nœud esclave. Nous ajoutons que le nœud serveur de messagerie et serveur d’annuaire ont une communication avec tous les autres nœuds. Le premier pour assurer la notification par messagerie électronique lors des événements déclenchants (échec, erreur, succès, espace insuffisant, etc.). Le deuxième permet de fournir une authentification centralisée des utilisateurs aux différents outils. Les détails des ressources (CPU, RAM, Disc) de chaque nœud sont déduits depuis la documentation officielle de chaque outil en tenant compte de quelques critères (Nombre d’utilisateurs estimé, etc.). Chapitre 3: Analyse et Conception 35 6. Architecture physique des environnements du projet AMC Le projet AMC dont nous avons déjà cité [cf. Paragraphe 1.1.5] se base sur la plateforme ecommerce Hybris. Afin de livrer en continu et avec qualité, le projet à plusieurs environnements (intégration, test, préparation, production). Après que nous donnons un aperçu de l’architecture générale typique de l’environnement, SAP Hybris commerce. Nous donnerons l’architecture physique spécifique de chaque environnement du projet AMC. 6.1. Architecture Déploiement Hybris Le diagramme suivant illustre une architecture de déploiement qui mappe les composants clés sur une infrastructure commune [11]. Ce n’est pas un plan exact qui peut être utilisé pour n’importe quel projet, mais qui donne une idée juste des composants matériels typiques utilisés dans l’architecture de déploiement. Figure. 17 : Architecture de déploiement commun Chapitre 3: Analyse et Conception 36 Plus de détails sur chaque composant sont donnés dans le tableau ci-dessous. Composant Rôle Équilibreur de charge L’utilisation principale d’un équilibreur de charge consiste à équilibrer le trafic sur plusieurs serveurs afin que la charge puisse être partagée entre plusieurs nœuds. Cela améliorera également la fiabilité du système, dans le cas d’une panne d’un nœud, l’équilibreur de charge peut acheminer les demandes entrantes vers d’autres nœuds. Serveur web Les serveurs Web agissent comme un proxy inverse pour transférer les demandes entrantes. Les serveurs Web aideront à sécuriser l’application, à accélérer le trafic Web et à réduire la charge sur les serveurs d’applications. Cluster d’application Hybris Le cluster Hybris est constitué de plusieurs serveurs d’applications Hybris Commerce, souvent appelées nœuds. Un cluster doit séparer le trafic client des traitements en ajoutant des nœuds de Frontend et de back-office spécifique. Serveurs de recherche Pour des raisons fonctionnelles et de performance, il est fortement recommandé d’utiliser une technologie de recherche dédiée pour la recherche et l’indexation. Base de données Hybris commerce prend en charge une variété de systèmes de bases de données. La réplication ou le clustering de base de données est recommandé pour le basculement et la haute disponibilité. Serveur de fichiers Hybris commerce utilise le système de fichiers pour stocker les images et autres éléments multimédias. Un NFS ou Storage Area Network (SAN) peut être utilisé pour stocker les ressources, évitant ainsi de devoir répliquer les fichiers sur tous les serveurs. Si les ressources statiques sont servies directement à partir des serveurs Web (par exemple, Apache), le système de fichiers partagé doit également être accessible aux serveurs Web. Table. 9 : Description du rôle de chaque composant La distribution de ces composants doit être pris sur l’évaluation de plusieurs critères (l’évolutivité, la haute disponibilité, la performance, la sécurité) et d’autres qualités du système pour garantir la configuration et la taille correctes. À noter que les environnements utilisés pour le développement et les tests d’acceptation des utilisateurs ne nécessitent pas une grande taille. Ces environnements sont principalement utilisés pour les tests internes et ne doivent pas être chargés avec un grand nombre de demandes et de processus. Cela étant dit, ils ne devraient pas être sous-dimensionnés. 6.2. Environnement d’intégration L’environnement d’intégration est utilisé par les développeurs afin de vérifier leurs changements du code source. Le déploiement à ses environnements s’effectue d’une manière automatique par Jenkins. Chapitre 3: Analyse et Conception 37 Figure. 18 : Digramme déploiement environnement d’intégration 6.3. Environnement de test Cet environnement est utilisé par les testeurs pour vérifier les fonctionnalités et les nouveaux changements par des contrôles manuels ou automatisés. Les testeurs peuvent lancer des tâches de déploiement envers ces environnements. L’architecture représentée dans le diagramme est proche de la production, mais pas similaire. Si les tests passent dans ces environnements, l’application alors peut être dépoilée dans l’environnement de préparation. Figure. 19 : Digramme déploiement environnement de test Chapitre 3: Analyse et Conception 38 6.4. Environnement de préparation et production Dans ce projet, les environnements production et préparation « pré production » sont similaires. L’utilisation principale de l’environnement préparation est de tester tous les scripts et procédures d’installation, configuration et migration, avant qu’ils ne soient appliqués à l’environnement de production. Cela garantit que toutes les mises à niveau majeures et mineures de l’environnement de production seront effectuées de manière fiable et sans erreur, en un minimum de temps. Alors que l’environnement de production est ou les serveurs dont va interagir le client et/ou seront les informations réelles des utilisateurs. Figure. 20 : Digramme déploiement environnement de préparation et production En fin, nous voyons la différence entre les environnements. Dans un premier lieu dans taille et nombre de nœuds dans le cluster Hybris. Dans un autre l’utilisation de réplication pour la base donnée MySQL. De plus nous ajoutons que le cluster Hybris dans la production contient deux « fronts end » ce qui permet de réaliser un déploiement sans indisponibilité. Chapitre 3: Analyse et Conception 39 Conclusion Ce chapitre nous a permis dans un premier lieu d’identifier les acteurs de notre système, ensuite nous avons spécifié les besoins fonctionnels et non fonctionnels de notre système avant de clôturer avec les architectures de déploiements des environnements du projet AMC et la plateforme de livraison continue.

Chapitre 4 : Réalisation Chapitre 4: Réalisation 41 Introduction Dans ce chapitre, nous mettons en place le pipeline de livraison continue amélioré. Ainsi que l’application des pratiques Pipeline as Code et Infrastructure as Code. 1. Mise en place du pipeline La livraison continue permet aux organisations de fournir des logiciels à moindre risque. Le chemin vers la livraison continue commence par la modélisation du pipeline de livraison de logiciels utilisés au sein de l’organisation et se concentre ensuite sur l’automatisation de tout cela. La rétroaction directe précoce, activée par l’automatisation des pipelines, permet une livraison plus rapide des logiciels par rapport aux méthodes de livraison traditionnelles. Jenkins est le couteau suisse dans la chaîne d’outils de livraison de logiciels. Le personnel des développeurs et des opérations (DevOps) ont des mentalités différentes et utilisent des outils différents pour accomplir leurs tâches respectives. Depuis Jenkins s’intègre à une grande variété d’outils, il sert de point d’intersection entre les équipes de développement et d’opérations. Figure. 21 : Processus de livraison continue La livraison continue est un processus — plutôt que des outils — et nécessite un état d’esprit et une culture qui doivent percoler du haut vers le bas au sein d’une organisation. Une fois que l’organisation a adopté la philosophie, la partie suivante et la plus difficile consiste à cartographier le flux des logiciels au fur et à mesure qu’ils passent du développement à la production [14]. Dans notre cas, nous avons réalisé une étude comparative des outils DevOps pour choisir les meilleurs et les plus adaptés aux besoins de SQLI. Ce qui nous a donné une architecture globale. Dans les parties qui suivent, nous précédons à la mise en place d’outils. Chapitre 4: Réalisation 42 Figure. 22 : Architecture globale des outils de livraison continue [15] La figure illustre un pipeline de livraison continue typique pour une application basée sur Java. La première étape du processus de CD est l’intégration continue. Le code créé sur l’ordinateur portable d’un développeur est continuellement intégré dans un référentiel de contrôle de version partagé et commence automatiquement son parcours dans tout le pipeline. L’objectif principal du pipeline de construction est de prouver que les changements sont prêts pour la production. Un code de modification de la configuration peut échouer à n’importe quel stade du pipeline, et cette modification sera par conséquent rejetée et non marquée comme prête pour le déploiement en production. Initialement, l’application logicielle à laquelle un changement de code est appliqué est construite et testée isolément, et une certaine forme d’analyse de la qualité du code peut (devrait) également être appliquée, peut-être en utilisant un outil tel que SonarQube. Le code qui passe avec succès les tests unitaires et composants initiaux et les métriques de qualité de code se déplacent vers la droite dans le pipeline et s’exerce dans un contexte intégré plus Chapitre 4: Réalisation 43 large. Finalement, le code qui a été entièrement validé émerge du pipeline et est marqué comme prêt pour le déploiement en production. Certaines organisations déploient automatiquement des applications qui ont réussi à naviguer dans le pipeline de génération et à passer tous les contrôles de qualité, ce qui est appelé « livraison continue ». 1.1. GitLab Après installation de GitLab, nous configurons LDAP pour avoir l’authentification centralisée et STMP pour recevoir les notifications depuis la page d’administration. Figure. 23 : Page d’Accueil GitLab Ainsi pour crée un dépôt de code source pour le projet AMC. Figure. 24 : Création de repository AMC Chapitre 4: Réalisation 44 1.2. Jenkins Après installation de Jenkins, nous configurons LDAP et STMP, ainsi que les outils des constructions comme Maven et Ant. En fin, nous ajoutons les esclaves Jenkins. Afin de recevoir les notifications en courrier électronique depuis Jenkins, nous devons configurer les paramétrées de STMP, dans la vue gestion Jenkins sous la section notification e-mail. Figure. 25 : Configuration de STMP Nous configurons l’outil de construction Maven dans la vue globale de configuration des outils. Figure. 26 : Configuration Outil construction Maven Nous terminons par l’ajout des Slaves que nous avons déjà cité dans la conception, comme suite. Figure. 27 : Configuration esclave Jenkins.

1.5. Ansible Afin de mettre en place, Ansible nous l’installons sur la machine de Jenkins. Ou l’avoir installé sur machine esclave de Jenkins. Nous avons choisi la première méthode pour sa facilité. 1.5.1. Installation et configuration Pour installer Ansible, nous trouvons plusieurs méthodes. La méthode la plus flexible c’est d’utiliser le système de gestion de package PIP, au lieu d’utiliser le gestionnaire de package du système d’exploitation. Pour installer une version donnée dans la machine Jenkins, nous lançons la commande : user@jenkins# sudo pip2 install ansible==2.1 après nous ajoutons le plug-in d’Ansible dans Jenkins, pour se faire, depuis le menu gestion de Jenkins, nous choisissons l’option gestion de plug-ins. Dans l’onglet disponible, nous cherchons Ansible. Ensuite, nous avons coché la case adjacente au plug-in et cliqué sur Installer sans redémarrer. Figure. 34 : Installation Jenkins Ansible plug-in Après l’installation du plug-in, pour le configurer nous allons à la gestion des outils dans l’interface web (gérer Jenkins> Vue Configuration globale de l’outil). Dans la section Ansible, nous ajoutons notre installation Ansible. Nous définissons le nom et le chemin d’accès au répertoire exécutable Ansible. Figure. 35 : Configuration outil Ansible dans Jenkins 1.5.2. Configuration des nœuds Pour exécuter les Playbooks Ansible à partir de Jenkins, nous devons configurer un accès SSH sans mot de passe entre le serveur Jenkins et les machines nœuds des environnements. Chapitre 4: Réalisation 48 Parce que nous avons besoin d’une forme de connexion automatique du serveur Jenkins aux machines nœuds. Nous devons également configurer le lancement de la commande « sudo » sans mot de passe pour l’utilisateur distant. - SSH login sans mot de passe : Pour chaque nœud, nous utilisons ssh-copy-id qui ajoute la clé publique à l’utilisateur distant ~/.ssh /authorized\_keys. jenkins@server:~ $ ssh-copy-id — i ~/. ssh/id\_rsa.pub user@hostname - La commande Sudo sans mot de passe : Pour activer sudo sans mot de passe, nous devons éditer le fichier/etc/sudoers, sur chaque nœud. En ajoutant la ligne suivante : $ echo « user ALL=(ALL : ALL) NOPASSWD:ALL » | sudo tee -a /etc/sudoers 1.5.3. Configuration tâche Ansible dans Jenkins Cette configuration d’une tâche normale qui exécute Ansible peut être utilisée dans les cas simples sans utiliser la ligne de commande. La configuration de la tâche est comme suit. Dans la vue de configuration de la tâche Jenkins, nous ajoutons le référentiel git de Playbook Ansible. Figure. 36 : Ajout du dépôt git tâche Ansible Et dans l’étape de construction du travail Jenkins, nous sélectionnons « Invoke Ansible Playbook » et nous spécifions dans les entrées du nom d’installation Ansible, le chemin du Playbook, le chemin du fichier d’inventaire. Figure. 37 : Configuration exécution Ansible depuis tâche Ansible Chapitre 4: Réalisation 49 Après l’exécution d’une tâche, nous avons un journal d’exaction. Ce journal est la sortie de lacement du Playbook sur les machines d’inventaire. Figure. 38 : Journal d’exécution d’une tâche Ansible Nous expliquons par plus de détails les concepts de base d’Ansible dans la partie gestion de configuration dans ce chapitre. 1.5.4. Ansible Vault pour les informations sensitives Le « Vault » est une fonctionnalité d’Ansible qui nous permet de conserver des données sensibles telles que les mots de passe ou les clés dans des fichiers cryptés, plutôt que de les laisser en clair dans les Playbooks ou les rôles. Ces fichiers cryptés peuvent ensuite être distribués ou placés dans le contrôle de source. Pour exécuter un Playbook Ansible avec des fichiers de données cryptés par Vault, nous devons fournir le mot de passe Vault. Pour notre cas, ce mot de passe est stocké dans vault\_pass.txt. Alors pour chiffrer ou déchiffrer un fichier de Vault : # ansible-vault {encrypt|decrypt} --vault-password-file={vault pass file} {vault file} Pour utiliser cette fonctionnalité avec Jenkins Ansible Plugin, nous devons fournir de même le fichier vault\_pass.txt. Figure. 39 : Ajout d’un fichier secret dans Jenkins Chapitre 4: Réalisation 50 2. Tâche pipeline dans Jenkins Le modèle d’interaction par défaut avec Jenkins, historiquement, a été très axé sur l’interface utilisateur Web, obligeant les utilisateurs à créer manuellement des tâches, puis à remplir manuellement les détails via un navigateur Web. Cela nécessite des efforts supplémentaires pour créer et gérer des tâches afin de tester et de générer plusieurs projets. Il conserve également la configuration d’une tâche à construire, tester et déployer séparément du code en cours de construction, test et déploiement. Cela empêche les utilisateurs d’appliquer leurs meilleures pratiques de livraison continue existantes aux configurations de travail ellesmêmes. 2.1. Pipeline as Code Pipeline as Code décrit un ensemble de fonctionnalités qui permettent aux utilisateurs de Jenkins de définir des processus de travail en pipeline avec code, stocké et versionné dans un référentiel source. Ces fonctionnalités permettent à Jenkins de découvrir, gérer et exécuter des tâches pour plusieurs référentiels et branches sources, éliminant ainsi la nécessité de créer et de gérer manuellement les travaux [12]. Pour utiliser Pipeline comme code, les projets doivent contenir un fichier nommé Jenkinsfile dans la racine du référentiel, qui contient un « script de pipeline ». 2.1. Jenkinsfile La présence du fichier Jenkinsfile à la racine d’un référentiel permet à Jenkins de gérer et d’exécuter automatiquement les tâches en fonction des branches du référentiel. Le fichier Jenkins doit contenir un « script de pipeline », en spécifiant les étapes d’exécution du travail. Le script a toute la puissance de Pipeline disponible, de quelque chose d’aussi simple que l’appel d’un constructeur Maven, à une série d’étapes interdépendantes, qui ont coordonné une exécution parallèle avec des phases de déploiement et de validation. 2.2. Jenkinsfile Projet AMC Notre Jenkinsfile du projet AMC contiendra plusieurs étapes chaque étape effectue des tâches spécifiques. Ces tâches sont : - Compilation et construction - Test unitaire - Analyse qualimétrie et publication rapport - Empaquetage des artéfacts - Publication des artéfacts sur Nexus - Déploiement sur environnement Chapitre 4: Réalisation 51 Figure. 40 : Jenkinsfile projet AMC Ce fichier Jenkinsfile est exécuté à chaque fois qu’il y a un changement au code source. À noter que l’exemple ci-dessus ne contient pas tous les détails. Dans la figure ci-dessous, on trouve un exemple d’exécution. Figure. 41 : Tâche pipeline projet AMC Après le succès des étapes constructions, test unitaire, empaquetage et publication des artéfacts sur Nexus. Nous passons à l’étape de déploiement des artéfacts. Ces artéfacts sont de la plateforme e-commerce Hybris, projet AMC. Pour les déployer, nous utilisons un « playbook » Ansible, dont nous allons donner le détail dans la prochaine partie. Dans la figure suivante, nous voyons la publication des artéfacts dans le dépôt d’objet binaire de type Maven, nommé « amc-releases ». Chapitre 4: Réalisation 52 Figure. 42 : page d’accueil Nexus et liste d’artéfacts De même cette figure ci-dessous, nous montrons les critères de qualité de code pour le projet AMC, ce rapport est actualisé à chaque analyse. À noter que si le projet dépasse la borne de qualité « Quality Gate » la tâche pipeline échouera. Figure. 43 : Rapport d’analyse qualimétrie D’après ce que nous venons de montrer, ce pipeline de livraison n’est pas encore mature en terme de tests. Pas de tests d’intégration et de régression après les tests unitaires. Aussi, les tests fonctionnels ne sont pas automatisés et se font manuellement par les testeurs. De plus les tests de performances et d’acceptation ne sont pas encore implémentés. Dans l’état actuel (Début du projet) peut être acceptable, mais les tests doivent être mise en place dans le futur. Autre point à signaler, à part le déploiement en continu dans les environnements d’intégration, le déploiement dans les autres environnements est manuelle est exécuté par les Chapitre 4: Réalisation 53 utilisateurs (Testeur, Chef projet, Développeur). Alors que dans les meilleures pratiques de livraison continue le déploiement doit être automatique jusqu’au les environnements de préparation « pré production » à chaque changement. 3. Gestion de configuration et Automatisation déploiement avec Ansible Dans les prochains paragraphes, nous allons découvrir l’architecture d’Ansible et son fonctionnement. Puis nous montons la gestion de configuration des environnements et le déploiement du projet AMC avec Ansible. 3.1. Architecture Ansible Ansible est un moteur d’automatisation simple qui automatise le Cloud, le provisionnement, la gestion de la configuration, le déploiement d’applications, l’orchestration de services ainsi que beaucoup d’autres fonctionnalités [14]. Dans cette section, nous allons vous donner un aperçu de la façon dont Ansible fonctionne. Figure. 44 : Architecture Ansible 3.1.1 Module Ansible fonctionne en se connectant aux nœuds et en poussant de petits programmes, appelés « Ansible Modules ». Ces programmes sont écrits pour être des modèles de ressources de l’état souhaité du système. Ansible exécute ensuite ces modules (sur SSH par défaut) et les supprime une fois terminé. La bibliothèque de modules peut résider sur n’importe quel ordinateur et aucun serveur, démon ou base de données n’est requis. 3.1.2 Inventaire Chapitre 4: Réalisation 54 L’inventaire fournit la liste des hôtes (serveurs managés) sur lesquels Ansible exécutera les tâches. Il peut également être utilisé pour regrouper les hôtes et configurer les variables pour les hôtes et les groupes. Figure. 45 : Exemple fichier Ansible Inventory 3.1.3. Playbooks Le Playbook est le langage de configuration d’Ansible utilisé pour effectuer l’ensemble des tâches et d’étapes de configuration ou pour faire appliquer une politique sur les nœuds distants. Les modules Ansible sont utilisés dans le Playbook pour exécuter une opération. Le Playbook est écrit en YAML qui est un langage simple, lisible par l’homme et développé dans un anglais de base comme la langue du texte. Les Playbooks sont plus susceptibles d’être conservés dans le système de contrôle de version et utilisés pour assurer les configurations des systèmes distants. Les Playbooks sont utilisés de la simple configuration et la gestion des nœuds distants jusqu’au déploiement avancé impliquant des mises à jour, le suivi des serveurs, la répartition de charge, etc. Figure. 46 : Exemple Playbook pour installation Apache 3.2. Automatisation Déploiement SAP Hybris e-commerce plateforme Pour déployer les artéfacts d’Hybris e-commerce plateforme, nous suivons plusieurs étapes : - Arrêter le serveur d’application Hybris Chapitre 4: Réalisation 55 - Crée un backup de la version installée - Téléchargement des artéfacts HybrisServer— \*. zip de la nouvelle version depuis Nexus - Décompression des artéfacts dans le répertoire/opt/hybris/(doit être vide) - Lancer le serveur d’application Hybris Nous avons donné les étapes à suivre dans le cas d’un déploiement dans un seul serveur. Dans le cas d’un cluster Hybris, la procédure se lance sur un nœud à la fois. Pour réaliser cette procédure avec Ansible, nous avons créé un Playbook qui s’exécute sur un groupe serveur web dans un inventaire. 3.2.1. Inventaire Nous donnons un exemple de l’inventaire de l’environnement de test. Cet environnement contient deux serveurs Hybris Frontend et Backoffice. Alors ces deux derniers qui seront affectés par le déploiement et sont regroupés par Web servers. Figure. 47 : Inventaire environnement test 3.2.2. Playbook déploiement Hybris Dans ce qui suit nous donnons une version du Playbook de déploiement Hybris qui appliquer les étapes déjà citées sur les nœuds web servers. Chapitre 4: Réalisation 56 Figure. 48 : Playbook déploiement Hybris Le Playbook fait appelle un rôle Ansible, un rôle est un ensemble des tâches et opérations organisé selon une structure standardisée pour offrir plus de modularité et réutilisabilité. Ce rôle nommé hybris-deploy contient plusieurs tâches [cf Figure. 49]. L’option serial est égale à un, ce qui impose à Ansible de lancer les opérations sur un nœud à la fois. 3.3.2. Exécution Pour exécuter ou lancer le Playbook sur un serveur ou ensemble de serveur Hybris, nous devons renseigner les paramètres suivants : - SERVER : nom d’hôte serveur Nexus - REPOSITORYID : l’identifiant de dépôt Nexus - GROUPID : Le nome de projet - ARTIFACTS : Une liste des artéfacts - VERSION : La version des artéfacts Cela rend ce Playbook applicable à tous les projets qui se base sur la plateforme Hybris, et non pas seulement le projet AMC. La commande suivante est un exemple d’exécution du Playbook : ansible-playbook -l front -i hosts plays/deploy.yml '{"SERVER": "'${SERVER}'","REPOSITORYID":"amcreleases", "GROUPID":"com.arcelormittal", "ARTIFACTS": ["hybrisServer-Platform","hybrisServer-AllExtensions","hybrisServer-Config"], "VERSION": "'${HYBRIS\_VERSION}'", "EXTENSION": "zip"}' --vault-password-file=../vault\_pass.txt Chapitre 4: Réalisation 57 Figure. 49 : Tâches du rôle Ansible hybris-deploy Chapitre 4: Réalisation 58 3.3. Gestion de configuration des environnements L’automatisation joue un rôle essentiel pour nous permettre de diffuser le logiciel de manière répétée et fiable. Un objectif clé consiste à prendre les processus manuels répétitifs, tels que la génération, le déploiement, les tests de régression et le provisionnement d’infrastructure, et à les automatiser. Pour ce faire, nous devons contrôler tous les éléments requis pour exécuter ces processus, notamment le code source, les scripts de test et de déploiement, l’infrastructure et les informations de configuration des applications, ainsi que les nombreuses bibliothèques et packages dont nous dépendons. Nous voulons également simplifier la recherche de l’état actuel et historique de nos environnements. Dans notre cas nous utilisons Ansible pour gérer la configuration des environnements du projet AMC. Chaque machine dans un environnement à des besoins et des dépendances spécifiques que nous devons gère d’une manière ou d’autres. Avec les outils de gestion de configuration, nous décrivons l’état d’une ressource et l’outil s’empare de garder la machine dans cet état. Nous prenons l’exemple du serveur Hybris de l’environnement de Test, à fin que Hybris peut se déployer sur ce serveur, le serveur doit respecter plusieurs points : - Application Oracle Java installé - Utilisateur hybris existe - Application ImageMagick installé - Les répertoires initiaux de la plateforme Hybris existent Pour répondre à ce besoin de pré-configuration de machine, nous avons créé un Playbook qui s’exécute sur le groupe web servers de l’inventaire environnement de Test [cf. Partie 3.2.1]. 3.3.1. Playbook configuration serveur Hybris De même que le Playbook de déploiement, ce Playbook fait appelle à un rôle nommé Hybrissite. Figure. 50 : Playbook configuration serveur Hybris Chapitre 4: Réalisation 59 Nous voyons que ces tâches utilisent des modules afin de rendre le serveur dans l’état désiré. Figure. 51 : Tâche du rôle Ansible hybris-site 3.3.2. Exécution Pour lancer ce Playbook, nous utilisons tous simplement la commande suivante : ansible-playbook -i hosts plays/site.yml --vault-password-file=.. /vault\_pass.txt Nous avons vu que la configuration du serveur Hybris, alors que toutes les autres machines sont configurées (MySQL, Solr, etc.) avec Ansible en utilisant des rôles déjà développés. Dans ce cadre, nous citons le site Ansible Galaxy qui est une bibliothèque des rôles qui peut être utilisé ou modifier dans quelque cas pour correspondre notre propre exigence. Ces Playbooks sont versionnés et ajoutés au code source, et décrivant l’état de notre infrastructure physique. Ce qui entre dans la pratique d’infrastructure en tant que code. Chapitre 4: Réalisation 60 Conclusion Ce dernier chapitre porte sur la réalisation des objectifs de notre travail, ainsi après la mise en place du pipeline. Nous montrons l’agrégation des tâches Jenkins dans une seule tâche de type Pipeline. Et nous terminons par la gestion de configuration des environnements avec Ansible.

Conclusion –

Générale Dans le cadre de ce projet de fin d’études, nous étions amenés à améliorer un pipeline de livraison continue en intégrant l’outil de gestion de configuration et d’automatisation Ansible. Après une étude comparative des outils DevOps, nous avons choisi les outils les plus adaptés aux besoins de SQLI. Et pour mettre en place ces outils, nous avons effectué une analyse et conception. L’analyse nous a permis de dégager les besoins fonctionnels et non fonctionnels, ainsi que les cas d’utilisation globaux de la plateforme. Alors que dans la conception, nous avons exposé les architectures physiques des environnements et de plateforme de livraison continue. Et enfin, nous avons terminé par une mise en place des différents outils. Dans le cadre du projet AMC, nous avons appliqué l’infrastructure en tant que code avec Ansible pour gérer la configuration et le déploiement dans les environnements, qui a permis de réduire le temps de préparation des machines. De plus, nous avons regroupé les tâches Jenkins dans un pipeline en tant que code qui s’exécute à chaque changement de code source et déploie automatiquement dans les environnements d’intégration après la passation des tests. Lors de ce travail, nous avons rencontré des problèmes liés à l’approvisionnement des machines qui ne peuvent pas être automatisées à cause de plateforme de virtualisation. Comme perspective à ce travail, nous proposons l’intégration de l’automatisation des processus de monitoring applicatif de performances et des machine.

Mise en place d’Ansible dans un pipeline de livraison continue Résumé Afin de réduire le délai de commercialisation et produire des logiciels de qualité, l’entreprise SQLI opte à intégrer les pratiques DevOps dans les projets de ses clients. Dans ce cadre vient ce projet pour améliorer et optimiser un pipeline de livraison continue, qui gère les changements depuis le code source jusqu’à la livraison en production, en intégrant l’outil Ansible pour la gestion de configuration et l’automatisation de déploiement. Après une étude des principales valeurs et pratiques de DevOps notamment la livraison continue et la gestion de configuration, nous avons effectué une étude comparative des outils d’automatisation de ces pratiques. Suite à l’étude comparative, une analyse des besoins et une conception de l’architecture du projet se sont apparues nécessaires, avant la mise en place du pipeline de livraison continue optimisé. Ce projet a permis de minimiser le temps de déploiement dans les différents environnements et d’avoir une résilience contre les pannes d’infrastructures. Mots clés : Agile, DevOps, Livraison Continue, Gestion de configuration, Ansible.

References : [DevOps : Construire un pipeline CI/CD avec Jenkins [Guide ultime] (data-transitionnumerique.com)](https://www.data-transitionnumerique.com/devops-jenkins/)

**DevOps : Construire un pipeline CI/CD avec Jenkins**

[Tweetez](https://twitter.com/intent/tweet?text=DevOps+%3A+Construire+un+pipeline+CI%2FCD+avec+Jenkins&url=https%3A%2F%2Fwww.data-transitionnumerique.com%2Fdevops-jenkins%2F%3Futm_source%3Dtwitter%26utm_medium%3Dsocial%26utm_campaign%3DSocialWarfare&via=Juvenal_JVC)

[Partagez](https://www.linkedin.com/cws/share?url=https%3A%2F%2Fwww.data-transitionnumerique.com%2Fdevops-jenkins%2F%3Futm_source%3Dlinkedin%26utm_medium%3Dsocial%26utm_campaign%3DSocialWarfare)

Enregistrer

[Print](https://www.data-transitionnumerique.com/devops-jenkins/)

[Partagez](https://www.facebook.com/share.php?u=https%3A%2F%2Fwww.data-transitionnumerique.com%2Fdevops-jenkins%2F%3Futm_source%3Dfacebook%26utm_medium%3Dsocial%26utm_campaign%3DSocialWarfare)

[WhatsApp](https://api.whatsapp.com/send?text=https%3A%2F%2Fwww.data-transitionnumerique.com%2Fdevops-jenkins%2F%3Futm_source%3Dwhatsapp%26utm_medium%3Dsocial%26utm_campaign%3DSocialWarfare)

**0PARTAGES**

Les méthodes DevOps telles que le CI/CD, notamment avec Jenkins sont de plus en plus appliquées au sein des entreprises de nos jours. En effet, ces concepts ne cessent de prendre de la place dans le monde de la création de solutions informatiques, surtout depuis l’avènement du Big Data. Pour preuve, il figure de nos jours parmi les [métiers porteurs du Big Data](https://www.data-transitionnumerique.com/les-7-metiers-porteurs-du-big-data-en-2019-2020/), puisque les ingénieurs DevOps/Cloud sont de plus en plus recherchés.

Cependant, pour que le processus de production d’une solution puisse se dérouler parfaitement, il faut connaître toutes les manières de procéder afin de livrer une application performante dans un délai minimisé.

Dans ce tutoriel exhaustif, nous allons vous apprendre à maîtriser ces étapes afin que vous puissiez construire un pipeline CI/CD avec Jenkins. Mais tout d’abord, définissons le contexte et voyons quelques notions essentielles avant de débuter la pratique.

DevOps : Définition

Mis en contexte

Il faut savoir que la réalisation d’un projet nécessite l’intervention de deux équipes informatiques : l’équipe de développement et l’équipe opérationnelle. La première s’occupe de la création de la structure, du développement et des tests de la solution. La seconde équipe, quant à elle, se charge du déploiement, de la mise en production et de la maintenance de l’application.

Avant, ces deux entités travaillaient indépendamment, mais cette manière de procéder présente quelques inconvénients. On constate par exemple des problèmes de compatibilité entre l’environnement de développement et de production. En outre, le délai entre ces deux étapes est rallongé et la qualité du produit amoindrie.

Afin de pallier ces soucis, certains modes de travail doivent être modifiés. L’adoption de la méthode agile est la première solution suggérée, mais les besoins nécessitent d’autres approches, notamment le DevOps.

Qu’est-ce que le DevOps ?

Le DevOps est apparu en 2007 et a intéressé tout de suite les métiers concernés par le domaine de l’informatique. Ce mot est la fusion de deux mots et des deux principales phases de conception d’une solution : la partie développement et la partie opérationnelle. Comme son appellation le suggère, il repose sur le fait que ces deux équipes doivent collaborer ensemble afin de proposer une application plus performante dans un délai plus petit.

Avant tout, le DevOps est une culture à adopter par les intervenants, une culture de partage entre chaque membre de l’équipe. Il s’agit également de changer la vision en ce qui concerne le cycle de développement d’une solution, c’est-à-dire apporter et adopter une nouvelle manière de travailler.

Le DevOps consiste aussi à automatiser le maximum de tâches chronophages pour les travailleurs afin de minimiser le délai de mise en production. Plusieurs outils sont utilisés pour cela dans le but d’améliorer la qualité perçue par le client et de mesurer efficacement l’impact de la solution à travers des KPIs.

Le schéma ci-dessus montre le cycle de vie du mouvement DevOps. Il démontre la continuité de tout le processus de réalisation d’un projet informatique, c’est-à-dire que les équipes sont en contact permanent.

Cette image confirme également que de nombreuses tâches sont automatisées afin d’appliquer l’intégration continue (CI), le déploiement continu (CD) et le monitoring continu, nous allons détailler ces concepts juste après. Ces derniers sont la base de la méthode DevOps.

Pipeline CI/CD : Définition

Avant de définir ce qu’est un pipeline CI/CD, il est important de connaître individuellement chaque partie de ce terme.

Intégration continue

L’intégration continue est un processus mis en place par les développeurs. Il consiste à fusionner progressivement les codes d’un programme dans un emplacement partagé. Autrement dit, les développeurs effectuent des modifications sur le programme, les testent individuellement, puis envoient les codes sur un référentiel partagé afin qu’ils puissent justement être intégrés au reste du programme.

Ce processus offre de nombreux avantages, notamment la réduction des tâches lourdes à effectuer, car on procède étape par étape dans la réalisation de chaque fonctionnalité, ce qui réduit les risques de conflits. Il permet également à ce que plusieurs développeurs puissent travailler en même temps sur un seul et même projet.

En adoptant cette pratique, on identifie vite les problèmes et les différents bugs. Et dans le cas où l’on ne détecterait pas ces derniers au moment des tests unitaires, ils seront tout de même vite résolus grâce à ce processus.

Déploiement continu / Distribution continue

Comme vous pouvez le constater par le titre de cette section, CD peut être interprété de deux façons différentes. Cependant, dans tous les cas, il repose sur le travail effectuer en intégration continue, c’est-à-dire que le référentiel partagé est mis en place.

La distribution continue consiste à ce que l’équipe opérationnelle accède au référentiel partagé afin que l’application puisse être mise en production rapidement. L’objectif ici est donc de réduire l’écart entre les deux principales équipes DevOps afin d’accélérer la production de l’application et, par conséquent, le délai de disponibilité de cette dernière auprès des utilisateurs.

Le déploiement continu, quant à lui, fait référence à l’automatisation des étapes de mise en production d’une solution. En effet, en utilisant ce processus, l’intégration continue effectuée par les développeurs est directement transférée à l’environnement de production. Les modifications peuvent donc être visibles au niveau des clients plus rapidement.

Cependant, l’équipe opérationnelle gère cette mise à jour comme elle le souhaite, c’est-à-dire qu’elle peut décider du moment et de la circonstance de déploiement selon les critères qu’elle impose, malgré le fait que ce soit automatisé par la suite.

Ce processus permet aux opérationnels de se focaliser sur les tâches non automatisées et donc de réduire considérablement le délai entre le moment de la création de l’application et celui où les clients l’utilisent.

Qu’est-ce qu’un pipeline CI/CD ?

L’utilisation de ces deux processus ci-dessus aboutit à la création de ce que l’on appelle un pipeline CI/CD, mais qu’est-ce que c’est réellement ?

Un pipeline CI/CD est un ensemble d’étapes permettant de distribuer une nouvelle version d’une application. Ce pipeline repose sur l’automatisation afin que cette distribution puisse s’effectuer de manière rapide et efficace. Toute l’approche DevOps est donc réunie et utilisée en un seul pipeline. Les processus que nous venons de citer dans les sections précédentes sont intégrés dans un job et s’effectuent automatiquement au fur et à mesure que l’application évolue.

L’adoption de cette méthode a révolutionné le monde du développement et de la production de solutions, car chaque pôle se concentre plus sur l’aspect pratique de la création, mais plus sur les soucis d’intégration ou autres tâches chronophages.

Les étapes de la construction d’un pipeline CI/CD

La construction d’un pipeline CI/CD se constitue d’une suite d’étapes et d’instructions à suivre, comme on a pu le voir dans sa définition. On peut regrouper ces étapes dans plusieurs phases à savoir :

* Le développement de l’application : cela consiste à coder l’application suivant les environnements que les développeurs ont choisis ;
* Les différents tests qui permettent de s’assurer de la qualité et de la cohérence de chaque fonctionnalité ;
* L’intégration continue, c’est-à-dire le transfert du programme au référentiel partagé. Lors de cette étape, le livrable peut prendre différentes formes et peut impliquer plusieurs outils ;
* Le déploiement continu qui consiste à mettre en production l’application. L’environnement de déploiement peut également varier selon le besoin de l’entreprise ou de l’équipe opérationnelle. Et pareillement, cette étape peut impliquer plusieurs outils ;
* La validation et conformité qui peut contenir plusieurs autres étapes selon le besoin de l’entreprise. Par exemple, c’est lors de cette étape que l’on examine les images ou que l’on vérifie si toutes les mentions sont bien présentes.

Jenkins : l’outil ultime pour construire un pipeline CI/CD

Plusieurs outils permettent de créer un pipeline CI/CD tels que GitLab, GoCD, Travis CI, Screwdriver ou encore Spinnaker. Cependant, Jenkins reste le plus complet. Il est le meilleur dans ce domaine et par conséquent le plus utilisé.

Jenkins est, comme vous vous en doutez, un outil d’automatisation des processus d’intégration continue et de déploiement continu. C’est un outil open source écrit en [Java](https://www.data-transitionnumerique.com/apprenez-programmation-java/) qui est basé sur un projet nommé Hudson créé par Oracle et repris par la fondation Eclipse pour être ensuite abandonné.

Sa principale utilisation réside dans la création de pipelines CI/CD. Ainsi, il permet aux utilisateurs de se connecter sur des systèmes de gestion de codes sources tels que GitHub. On peut également utiliser des outils de conteneurisation comme Docker et Kubernetes avec Jenkins. D’autres opérations sont possibles et plusieurs plug-ins sont disponibles afin qu’il puisse prendre en charge de nombreux autres outils.

La création d’un pipeline Jenkins repose sur la création d’un fichier nommé **Jenkinsfile**. Il s’agit d’un fichier texte contenant toutes les étapes à suivre pour créer le pipeline. Généralement, il contient une partie déclarative et une partie script qui, elles, contiennent plusieurs étapes afin de rendre le pipeline CI/CD fonctionnel.

Les outils associés à Jenkins pour la création d’un pipeline CI/CD

On peut utiliser Jenkins avec plusieurs outils comme en témoignent les nombreux plug-ins que l’on peut y installer comme extensions. Mais en général, pour construire un pipeline CI/CD fonctionnel, les différents types d’outils suivants sont nécessaires.

Les gestionnaires de codes sources

Les gestionnaires de codes sources permettent à Jenkins et aux autres membres de l’équipe de développement d’accéder aux codes sources de l’application. Il existe plusieurs gestionnaires, mais **Git**reste le meilleur et le plus utilisé d’entre eux.

C’est un gestionnaire de codes sources open source créé par Linus Torvalds et dont le principal contributeur est Junio C Hamano. Il permet de stocker les programmes sur un serveur dédié sur lequel tous les contributeurs peuvent accéder. Il sert également à gérer l’évolution de l’arborescence du projet en intégrant les modifications apportées par chaque développeur.

Les outils d’automatisation de compilation

Les outils d’automatisation de compilation permettent de déployer l’application qui vient d’être codée. Ces outils ne sont pas utiles pour les applications écrites avec certains langages tels que JavaScript. Cependant, pour les langages compilés comme Java, ils sont vraiment utiles.

En intégration continue, le plus utilisé d’entre eux est **Maven**. Il s’agit d’un outil géré par Apache Software Foundation permettant de créer un fichier jar pour distribuer un ensemble de classes Java.

Les serveurs d’applications

Les serveurs vont servir d’emplacement de stockage pour l’application à gérer. Il peut s’agir d’un serveur web, d’une machine locale ou d’une machine virtuelle.

Les outils tels que **Docker**et **Kubernetes**peuvent également être considérés comme des serveurs d’applications. En effet, ces derniers permettent de stocker, de gérer et de lancer les applications sous forme de conteneurs.

Les outils de tests

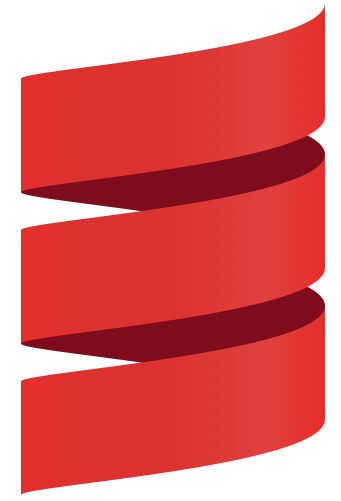
On peut également effectuer les tests dans un pipeline Jenkins. Cette étape, comme on l’a vu un peu plus haut, permet de valider chaque fonctionnalité et de s’assurer que l’application livrée est bien fonctionnelle.

Certains langages disposent d’outils compatibles avec Jenkins afin d’effectuer ces tests tels que **JUnit** et **PowerMok** pour Java ou encore **Pytest** et **Tox**pour Python.

Pourquoi utiliser Jenkins ?

Comme nous l’avons mentionné un peu plus haut, il existe plusieurs outils permettant d’implémenter des pipelines CI/CD. Mais pourquoi Jenkins sort du lot, pourquoi le choisir ? En effet, Jenkins offre plusieurs avantages intéressants à savoir :

* Jenkins est open source et gratuit. On peut donc en disposer pour tout type de projet, que ce soit dans le monde professionnel ou en apprentissage ;
* Sa communauté est très active. Vous pouvez donc compter sur elle en cas de problèmes ;
* On peut l’installer de plusieurs façons et, quelle que soit la manière que l’on choisit, cela se fait très simplement ;
* Il est également multiplateforme, c’est-à-dire qu’il fonctionne autant sur Windows, Linux ou macOS ;
* Jenkins dispose d’une interface sur laquelle on peut facilement gérer chaque fonctionnalité d’un pipeline ;
* Il permet d’effectuer l’intégration et le déploiement continus de manière automatique ;
* En cas de problème que niveau de chaque étape, Jenkins va vous prévenir à travers les logs que l’on peut consulter au niveau de l’interface ;
* Il est compatible avec la plupart des plateformes de développement, de test et de déploiement. En plus, une multitude de plug-ins sont disponibles sur Jenkins afin de faciliter la création du pipeline CI/CD. Et si vous ne trouvez pas le plug-in dont vous avez besoin, vous pouvez le créer vous-même.

[](https://www.data-transitionnumerique.com/wp-content/uploads/2021/02/scala-logo.jpg)

**FORMATION PRATIQUE  - SCALA POUR LE BIG DATA**

**Vous souhaitez monter en compétence sur la programmation Scala ? Nous vous offrons cette vidéo-tutoriel d'une heure à l'intérieur de laquelle vous apprendrez:**

1. 1

à manipuler les collections (mutables et immutables) de scala, ainsi que ses structures de données pour développer des applications data (listes, tableaux, tuples)

1. 2

à utiliser les structures conditionnelles de scala (For, ForEach, While) pour développer des applications efficaces

1. 3

à utiliser Scala  pour résoudre les problèmes rencontrés lors de la programmation des tâches Big Data

**Cette formation va faire de vous un bon développeur en scala. Téléchargez-là simplement en cliquant sur le bouton ci-après.**

[Télécharger la formation](https://www.data-transitionnumerique.com/devops-jenkins/)

Tutoriel : construire un pipeline CI/CD avec Jenkins de A à Z

Dans cette partie, nous allons voir toutes les étapes à suivre afin de construire un pipeline CI/CD fonctionnel avec Jenkins.

Installer Docker et Kubernetes

Docker et Kubernetes font partie des prérequis pour ce tutoriel. Il faut donc les installer sur votre machine. Pour ce faire, nous vous invitons à consulter leurs tutoriels respectifs sur notre blog. Cliquez [ici :Maitrisez la conteneurisation d’application avec Docker](https://www.data-transitionnumerique.com/docker-tuto-complet/) pour installer Docker et [ici : Maîtrisez le déploiement et l’orchestration d’applications avec Kubernetes](https://www.data-transitionnumerique.com/kubernetes-guide-complet/) pour Kubernetes, ou plus précisément *Minikube*que nous allons utiliser dans ce tutoriel vu que nous sommes en local.

Une fois que vous terminez l’installation de ces deux outils, vous pouvez passer à l’étape suivante.

Installer Jenkins

Il existe plusieurs moyens d’installer Jenkins, nous vous invitons à tous les consulter sur leur [site officiel](https://www.jenkins.io/doc/book/installing/). Dans ce tutoriel, nous allons l’installer en le déployant dans un cluster Kubernetes. Il existe également plusieurs manières d’effectuer ce déploiement, mais ici, nous allons utiliser un fichier *YAML*pour cela.

Configurer Jenkins

Avant de nous lancer dans la création du pipeline, il faut que l’on effectue quelques configurations au niveau de Jenkins.

Ajouter des plug-ins

Quelques plug-ins relatifs aux configurations que l’on vient d’effectuer sont également nécessaires. Pour les installer, il faut aller dans *Administrer Jenkins* puis dans *Gestion des plug-ins*. Situons-nous au niveau des plug-ins disponibles et tapons le nom du plug-in que l’on souhaite installer dans la barre de recherche.

Pour ce tutoriel, nous avons besoin des plug-ins **Kubernetes CLI**, **Maven Integration plug-in** et **Docker**. Pour commencer l’installation, nous devons cliquer sur l’un des deux boutons situés en bas à gauche.

1. Le service de la transformation numérique et logistique

Le service STNL se divise en deux unités et un pôle :

* Le pôle logistique, a la charge de la gestion du courrier, des salles de réunions, des locaux de la HAS, du mobilier et des véhicules et des fournitures.
* L’unité infrastructure et support poste de travail a la charge de l’administration et de la maintenance de l’infrastructure serveurs, réseau, téléphonie et des postes de travail de la HAS ainsi que du support informatique aux utilisateurs.
* L’unité applications dans laquelle j’ai été intégré pour mon stage, qui est composée essentiellement de chefs de projets, a la charge du pilotage, de la conception, et du suivi de la maintenance des applications déployées par la HAS.

Son importance est capitale pour la HAS. En effet les systèmes d’informations déployés et gérés par le service ne sont pas utilisés uniquement par les salariés de la HAS, mais également par des utilisateurs externes (professionnels de santé notamment). On peut estimer le nombre d’utilisateurs des services d’information de la HAS à plus de 10 000.

Pour accompagner son projet stratégique, la HAS s'appuie sur un plan de transformation numérique 2020-2024. Il a pour objectif la modernisation d'un certain nombre de systèmes d'information ainsi que le développement de nouveaux services numériques pour couvrir tous les champs d'activité de la HAS. Il repose sur 4 piliers : la simplicité d'usage, la traçabilité des travaux, la satisfaction des utilisateurs et l'accompagnement des utilisateurs

Scrum- reference if like to read : [(PDF) Mémoire de Fin d'Etudes Pour l'obtention du diplôme D'Ingénieur d'Etat Génie Informatique Option : Système D'information Membres de jury | joairia lafhal - Academia.edu](https://www.academia.edu/36811214/M%C3%A9moire_de_Fin_dEtudes_Pour_lobtention_du_dipl%C3%B4me_DIng%C3%A9nieur_dEtat_G%C3%A9nie_Informatique_Option_Syst%C3%A8me_Dinformation_Membres_de_jury)

1. La plateforme de développement PEGA
2. Contexte

1. Problématique

Aujourd’hui, la HAS se repose encore beaucoup sur des formats papiers pour la gestion des actes internes. Cela a pour conséquence de ralentir les démarches, il y a un manque de suivi et beaucoup de pertes d'informations qui ne sont pas retrouvées par la suite.

1. Enjeux et objectifs

Lors de mon arrivée à la HAS, mon stage était censé se focaliser sur deux projets :

* Le premier était le projet D2T, qui avait pour objectif la gestion des demandes de forfait de télétravail. L’application étant déjà en production à mon arrivée, mon rôle était la correction de tous les bugs éventuels et surtout l’apport d’améliorations à cet outil. En effet, les utilisateurs et plus particulièrement le service RH avait émis plusieurs demandes d’améliorations.
* Le second était le projet GEC, qui avait pour objectif la gestion de la réception à l’envoi d’une réponse de tous les courriels et courriers qui sont adressé à la HAS. Ce projet est développé par des prestataires externes à la HAS, et le rôle de la HAS est uniquement d’effectuer la gestion de projet, j’ai donc été associé à ce projet via les chefs de projets.

Après plus d'un mois de stage, toutes les améliorations possibles avaient été apportées sur D2T. Sur le projet GEC, mon rôle d'observateur, riche en enseignements sur la gestion de projet, ne pouvait pas constituer l'élément principal de mon travail. J’ai donc pris l’initiative de proposer un nouveau projet à mon maitre de stage.

La crise de la Covid-19 a fortement impacté la HAS, comme toute autre entreprise, sur la question du télétravail. En 2021, l’entreprise a donc décidé que ce mode de fonctionnement pouvait être adopté définitivement sous conditions que certaines règles soient respectées. La création d’une charte de télétravail a donc suivi cette décision.

Jusqu’alors, le suivi du respect de cette charte passait par un fichier Excel que chaque salarié devait remplir. Afin de faciliter la gestion des jours de télétravail, j’ai proposé l’idée de créer une extension de l’application D2T que je maitrisais bien.

Ce projet après avoir été validé par mon maitre de stage et ses supérieurs, a pris le nom de « Suivi des jours de télétravail ». L’outil attendu par les responsables de la HAS doit répondre à plusieurs besoins :

* Un agent doit pouvoir remplir un calendrier avec ses jours prévisionnels de télétravail et cela sur plusieurs mois voire années à venir. Les jours doivent pouvoir être différenciés entre forfait et occasionnel, ainsi qu’entre matin, après-midi et journée entière. Le prévisionnel du télétravail d’un agent doit respecter la charte de télétravail ainsi que le forfait de l’agent.
* Un manager doit pouvoir en plus de la déclaration de ses propres jours de télétravail, consulter sur un calendrier adapté et de façon claire, le télétravail de tous ses agents. Il doit pouvoir également annuler un jour prévisionnel d’un agent.
* Le manager doit envoyer chaque mois un tableau Excel rapportant le télétravail réalisé par chaque agent de son équipe. Cette tache doit pouvoir être réalisée en un seul clic.
* Le service RH doit pouvoir consulter en fonction de la direction et du service le télétravail réalisé par tous les agents de la HAS.

1. Présentation du projet A) Contexte professionnel du projet Il s’agit d’enrichir le portail de fonctionnalités pour qu’il puisse, à terme, supporter l’ensemble des processus métier des agents de la CNAMTS, en répondant à des enjeux majeurs tel que : • Améliorer l’efficience de l’Assurance Maladie par la mise en œuvre de processus simplifiés et de workflows. • Élargir l’offre de services en ligne en apportant une solution automatisée et sécurisée pour le traitement de ces demandes. • Maîtriser les risques par une automatisation croissante de certaines tâches. Ce portail est basé sur une infrastructure Oracle WebLogic Server, un serveur d’application JAVA JEE constitué entre autre d’un portail (WebLogic Portal). Pour des raisons de performance, la CNAMTS a souhaité développer son propre portail (appelé Portail Agent) et l’a intégré à l’infrastructure WebLogic pour remplacer WebLogic Portal. L’objectif du projet pour Gfi est de développer des plugins et de les intégrer au portail de la CNAMTS, chaque plugin ajoutant des fonctionnalités métiers. Le projet, pour la CNAMTS est un chantier qui s’intègre dans un vaste programme, le Programme 6 (programme d’évolution de la CNAMTS). Pour cela, le portail doit répondre à certaines exigences : • Mettre à niveau fonctionnellement le portail. • Intégrer les processus métiers des agents de la CNAMTS dans un portail WEB unique, intégré, sécurisé et adapté aux différentes populations d’agents et aux différents usages. • Renforcer et diversifier une offre de services de qualités pour usagers (personnes assurés à la CNAMTS). • Améliorer l’ergonomie du poste de travail pour les salariés de la CNAMTS. • Créer une nouvelle interface homme-machine (IHM) qui respecte le Référentiel Général d’Accessibilité pour les Administrations (RGAA).
2. Mes activités propres et détaillées pendant mon stage

J’ai été intégré à l’équipe de développeurs composées de Benoit COUDERC et Anthony COUE dès le début de mon stage, et ils m’ont suivi sur toute la durée de celui-ci.

Ils ont travaillé avec moi sur la quasi-totalité de mes activités détaillées ci-dessous quand ils en avaient le temps. En effet, leur double casquette développeur/chef de projet ne leur laissait pas autant de temps que moi pour avancer sur les sujets de mon stage.

Nous nous sommes donc organisés de la manière suivante : Un point à trois nous permettait de définir les sujets sur lesquels je pouvais avancer tout seul, je travaillais ensuite en autonomie, puis nous abordions les sujets bloquants ensemble.

1. Scientifiques et techniques

La partie scientifique et technique de mon stage est exclusivement réalisée sur la plateforme PEGA, une plateforme de développement *Low-code*. PEGA permet à l’utilisateur de développer son application de manière très visuelle, suivant un principe de *drag & drop*. PEGA génère automatiquement le code en java.

1. Prise en main de la plateforme

N’ayant jamais travaillé sur une plateforme de Low-code, je ne pouvais pas commencer la réalisation de mes objectifs de stage sans m’approprier des connaissances sur ce type de développement et plus particulièrement sur la plateforme PEGA.

1. PEGA academy

Durant la première partie de mon stage, j’ai suivi des cours en ligne dispensés par PEGA, via la *PEGA academy.* Etant donné la courte durée de mon stage, j’ai uniquement suivi le module système architecte des cours dispensés expliquant comment réaliser le développement d’un projet basique sur la plateforme.

Les cours en ligne sont organisés de manière très captivante, en effet chaque cours théorique de 15-20 minutes s’accompagne d’un exercice de développement sur une plateforme « bac à sable ».

J’ai passé une semaine complète à travailler sur ce module, avant de commencer à m’intéresser aux projets sur lesquels j’allais contribuer tout en continuant ma formation à côté. Le temps que j’ai passé sur le module débutant a été d’environ 50 heures.

Ce site m’a également été très utile lors de la réalisation de mes différentes tâches, car il détaille toutes les fonctionnalités proposées par la plateforme. J’ai donc pris le réflexe d’aller chercher sur *PEGA academy* dès que je me retrouvais bloqué.

PEGA propose également une plateforme*PEGA Community* où les utilisateurs peuvent partager leurs questions/problèmes et où des experts PEGA prennent du temps pour essayer d’apporter des réponses/solutions. Cette plateforme m’a également énormément aidé tout au long de mon stage.

1. Découverte des applications D2T et DNF

Après avoir pris en main la plateforme et en avoir compris le fonctionnement, j’ai eu également l’opportunité de découvrir les applications développées en interne par l’équipe de la HAS. Cette étape, toujours théorique, m’a permis d’avoir un autre point de vue sur la plateforme. Là où la *PEGA academy* m’a apporté des connaissances sur tout ce qu’il était possible de faire grâce à la plateforme, A. Coué et B. Couderc m’ont donné un aperçu des difficultés que l’on peut rencontrer lors du développement d’un projet.

1. Environnements PEGA

Comme pour tout développement, il faut savoir distinguer les environnements de développement, test production et autres afin de pouvoir travailler sur l’un sans impacter les autres. PEGA prend en charge cet aspect du développement en mettant à disposition plusieurs en**vironnements :**

1. Ateliers de conception

Ces ateliers se sont découpés en trois présentations, d’abord à l’aide d’un diagramme fonctionnel ainsi qu’une maquette créée sur Miro, un outil de travail collaboratif visuel. Cette première étape a servi à cerner les besoins auxquels répondait l’application :

1. Rapport personnel
2. Des compétences techniques et organisationnels

Ce stage m’a apporté énormément de compétences techniques très spécifiques. En effet, j’ai passé l’essentiel de mon temps à développer sur la plateforme PEGA. Ces compétences pourront m’être très utiles à l’avenir si je suis amené à développer à nouveau sur cette plateforme. Une expérience, des contacts et la connaissance des certifications de PEGA pourraient effectivement m’apporter énormément dans le futur.

De plus, les compétences acquises sur cette plateforme ne seront pas perdues si je suis amené à ne plus travailler dessus. En effet, en dehors du fonctionnement spécifique de PEGA, j’ai développé des compétences de modélisations objets, qui sont très présente sur la plateforme. J’ai également à présent un aperçu du fonctionnement des plateformes Low-code.

Ce stage m’a permis de découvrir et d’assimiler beaucoup de connaissances sur les méthodologies agiles et plus particulièrement la méthode SAFe. Au cours de toutes les étapes du projet GEC que j’ai suivies, j’ai appris les rôles de tous les différents acteurs ainsi que les nombreuses étapes nécessaires au bon déroulement d’un projet de cette envergure.

1. Une expérience en entreprise

Ce stage n’a pas vraiment été ma première expérience en entreprise, en effet j’avais déjà travaillé à la blanchisserie du CHU d’Angers pour des emplois étudiants. Il s’agit cependant de ma première expérience dans le secteur de mes études, et donc potentiellement dans lequel je vais être amené à travailler dans les années à venir. J’ai découvert dans le service STNL, un mode de fonctionnement qui me correspond tout à fait. Malgré une hiérarchie très claire mise en place, je ne me suis pas senti tout en bas de la pyramide à devoir obéir à tous les ordres que l’on pouvait me donner. Pendant ces 14 semaines de stage, j’ai travaillé main dans la main avec tous les collaborateurs de la HAS. Toutes les personnes avec qui j’ai eu le plaisir d’échanger ont toujours été prêt à m’aider, à discuter d’égal à égal avec moi.

 Une des peurs que j’avais avant de commencer mon stage était liée aux stéréotypes du stagiaire. Je ne voulais surtout pas que mon stage se résume à réaliser les tâches que mes supérieurs n’auraient pas envie de faire. Au lieu de ça, mon maitre de stage m’a accompagné dans la réalisation de mon projet, et a tout mis en œuvre pour que mon stage soit le plus intéressant et enrichissant pour moi.

Même si j’ai trouvé dommage que le mode de fonctionnement du télétravail soit partiellement adopté par la HAS, ce qui réduit grandement le contact humain, j’ai trouvé que cela était bien réalisé, avec un certain nombre de règles à respecter notamment celle qui impose un minimum de jours en présentiel.

J’ai donc appris le fonctionnement d’un service dans une entreprise, avec des réunions bimensuelles, des entretiens d’évaluation annuels etc…

1. Mon projet professionnel

Avant même de commencer mon stage, j’avais comme projet professionnel de devenir chef de projet SI. Cependant, je ne savais pas exactement ce à quoi correspond ce métier, et j’avais énormément de questions à ce propos.

L’unité application de la HAS dans laquelle s’est déroulé mon stage a l’énorme avantage pour moi d’être composé en grande partie de chefs de projets. Au cours de ces 14 semaines passées avec eux, j’ai eu le loisir d’observer leur quotidien et de leur poser toutes les questions qui me venaient. Le seul point qui m’a alors rebuté est le fait que ces chefs de projets ne touchent pas du tout à la partie technique/développement des projets. En effet, comme le service STNL fait appel à un prestataire pour le développement de presque tous ses projets, le rôle des chefs de projets se limite à du pilotage. B. COUDERC et A. COUE avec qui j’ai travaillé sur mon projet, quant à eux se concentrent beaucoup plus sur le développement, et moins sur la partie gestion de projet qui est de mon point de vue tout autant intéressante. J’ai donc posé la question à plusieurs membres du service, qui avaient déjà eu d’autres expériences professionnelles, et qui ont su me répondre. Il est effectivement possible d’avoir des responsabilités de pilotage de projet et de développement sur le même projet. J’ai donc en tête à présent de chercher un poste qui répondrait à cette description pour mon stage de fin d’étude ainsi que pour mes premiers emplois après mes études.

J’ai également pu aborder à plusieurs reprises la question des prestataires. En effet, beaucoup d’entreprises préfèrent faire appel à des prestataires externes pour la totalité de la gestion de leur projets S.I. Dans le cas de la HAS, le rôle de ces partenaires se limite au développement des projets, ceux-ci étant piloté en interne. Après avoir récolté plusieurs avis, j’en suis venu à la conclusion que pour un début de carrière, le statut de prestataire avait beaucoup d’avantages. D’abord, il permet de créer beaucoup de relations qui peuvent être cruciales aujourd’hui pour avancer dans sa carrière. Ensuite, être prestataire peut me permettre de vérifier si le métier de chef de projet est vraiment ce qui me plait. Une réorientation dans ce milieu est ensuite faisable sans trop de difficultés.

Ce stage, en dehors des compétences techniques qu’il m’a apporté, a donc également été un stage d’observation pour moi et a permis d’affiner mon projet professionnel.

1. Conclusion du rapport

J’ai trouvé ce stage de 14 semaines à la HAS très enrichissant. J’ai pu découvrir non pas un mais plusieurs métiers qui m’intéressais comme projet professionnel.

J’ai été très heureux de me voir confier des responsabilités en participant au développement d’un outil utilisé par tous les agents de la HAS, mon seul regret étant de n’avoir pas eu assez de temps et de n’avoir présenté qu’un prototype de mon projet sans pouvoir l’améliorer avec des retours utilisateurs.

Travailler avec B. Couderc et A. Coué fut un plaisir, ceux-ci s’étant grandement investis dans mon projet lorsque leurs emplois du temps le permettaient.

Les taches que l’on m’a confié au début du stage concernaient des corrections de bugs et des petites améliorations de l’application D2T. Cette période m’a permis de prendre en main la plateforme PEGA et d’apprendre à développer dessus. Après plus d’un mois sur ces sujets, plusieurs réunions avec les développeurs et mon maitre de stage ont permis de créer un nouveau projet qui serait le principal objectif de mon stage : le projet suivi des jours de télétravail. Ce projet a pour objectif de permettre aux agents de déclarer leurs jours de télétravail, et aux managers de suivre tout leur service. J’ai donc travaillé sur ce sujet la totalité de mon temps à partir de son lancement. Grace à l’aide des développeurs du STNL, j’ai réussi à finir dans les temps le projet et à en proposer une V0.

J’ai également eu la chance de suivre la construction du projet GEC, piloté par la HAS mais développé par des prestataires. Cela m’a permis d’apprendre beaucoup sur la méthodologie agile et sur les occupations d’un chef de projet.

Je remercie encore une fois tous les membres de l’unité application à laquelle j’ai été intégré, pour leur accueil. Dès que j’ai eu besoin d’aide, ils se sont rendus disponible, ce qui m’a permis de ne pas rester bloqué en attendant des réponses à mes questions. Leurs commentaires sur mon projet ont toujours été très constructifs et positifs ce qui m’a beaucoup motivé. Enfin, mon intégration dans l’équipe a été rapide et complète, j’ai donc eu un cadre de travail très agréable durant me 14 semaines à la HAS.

En conclusion, je suis très heureux d’avoir passé tous ce temps à la HAS, car cela m’a permis de confirmer que je souhaite travailler dans le développement de logiciels, applications dans le futur.

VI- Conclusion J’ai pu effectuer ce stage dans d’excellentes conditions de travail, j’ai eu la chance : • de pouvoir travailler dans une ESN importante, • d’étudier leurs méthodes de travail sur un projet de taille non-négligeable. C’est une expérience qui n’est pas facilement accessible pour un étudiant et qui m’a permis d’apprendre beaucoup et de gagner énormément en ce qui concerne l’expérience professionnelle, un avantage qui je suis sûr aura de l’importance pour mon avenir. J’ai eu également la chance d’être au contact de personnes compétentes aussi bien d’un point de vue technique que pédagogique. C’est une chose différente de mes expériences professionnelles passées où je devais travailler de manière autonome, sans encadrement. Cela m’a permis de découvrir de manière concrète comme fonctionne une entreprise informatique et m’a donné un aperçu du processus de développement des projets d’envergures. J’ai eu au début un travail d’adaptation et de formation qui s’est avéré très intéressant, ce qui m’a fait progresser d’un point de vue technique : • j’ai maintenant une idée précise de ce qu’est l’intégration continue et le mouvement DevOps, • j’ai progressé techniquement grâce aux nouvelles technologies assimilées. J’ai appris l’importance du travail à faire, en amont, avant de développer, mais aussi l’importance du Test Driven Development et de l’Agilité. Il est vrai que techniquement, je n'ai pas apporté énormément aux autres développeurs, en revanche, j'ai réussi à prendre du recul par rapport aux problèmes que l'équipe rencontrait et à apporter d'autres idées grâce à mon point de vue d'étudiant, divergent du point de vue des développeurs présents. C'était pour moi vraiment enrichissant de pouvoir débattre des solutions qu'il fallait choisir et de la démarche suivie. Je pense pouvoir dire que je suis vraiment satisfait de ce stage, il fut différent de celui effectué en premier année, c'était aussi une de mes attentes. D'un point de vue personnel j'ai énormément progressé, que ce soit techniquement, professionnellement ou encore humainement. Pour l'entreprise, j'ai pu aider aux développements d'un projet, j’ai pu apporter une manière de voir les choses différentes, peut-être parfois « naïve » mais qui m'a semblé importante de prendre en compte. Enfin, ce stage a permis de voir les deux parties (l'entreprise et le stagiaire) travailler ensemble avant une éventuelle future alternance.

[Rapport de projet\_de\_fin\_d\_\_tudes\_\_pfe\_\_safwen (8) | PDF (slideshare.net)](https://fr.slideshare.net/safwenbenfredj/rapport-de-projetdefindtudespfesafwen-8)

1. Références bibliographiques

**Méthodologie agile :**

<https://www.scaledagileframework.com/>

<https://has-sante.atlassian.net/jira/>

Supports utilisé lors de PI planning.

**HAS :**

<https://www.has-sante.fr/>

<https://fr.wikipedia.org/wiki/Haute_Autorit%C3%A9_de_sant%C3%A9>

Fichiers mis à disposition sur Teams par la HAS (organigramme, présentation du service, charte du télétravail, présentation du PTN)

**PEGA :**

Ensemble des comptes-rendus des réunions par suite du *sourcing* auprès d’éditeur de solution BPM.

<https://academy.pega.com/>

<https://community.pega.com/>

<https://collaborate.pega.com/>

Environnement de développement de PEGA.

1. Glossaire des acronymes

**HAS : Haute Autorité de Santé.**

**STNL : Service de la Transformation Numérique et Logistique.**

**D2T : Demandes de télétravail.** Application développée sur PEGA en interne utilisée par la HAS pour que les agents puissent faire des demandes de forfait de télétravail.

**DNF : Dématérialisation des Notes de Frais.** Application développée sur PEGA en interne utilisée par la HAS pour permettre à ses agents ainsi qu’à ses intervenants externes de communiquer leurs notes de frais de manière entièrement dématérialisé.

**GEC : Gestion Electronique des Courriers.** Application développée sur PEGA par la HAS et KE utilisée par la HAS pour assurer une gestion et un suivi des courriers papier ainsi que des courriels.

**KE : Knowledge Expert.** Entreprise de prestataires externes à la HAS. La HAS fait appel à eux sur de nombreux projets de transformation numérique. Ils sont spécialisés dans le développement sur PEGA, étant un de leur principaux partenaire en Europe.

**PEGA : Pegasystems.** Plateforme de développement Low-Code utilisé par le service STNL de la HAS dans la plupart de ces développements.

**RH : Ressources Humaines.** Services gérant le recrutement, les contrats, la paie, les carrières, la formation des agents de la HAS.

**SI : Systèmes d’informations.**

**ARS : Agence régionale de Santé.**

**HTML : HyperText Markup Language.** Langage informatique décrivant la présentation d’une page web pour l’utilisateur.

**CSS : Cascading Style Sheets.** Langage informatique plus communément connus sous l’appellation de feuille de style, sert à décrire la présentation, le style des documents HTML.

**PI Planing : Programm Increment Planing.** Il s’agit d’une étape de la méthode agile SAFe au cours de laquelle les acteurs se réunissent pour faire un point sur ce qui a été et ce qui va être développé.

**BPM : Business Process Management.** Offre un nouvel angle de perception du rôle des technologies au service du management et de la performance.

**Live U.I. : Live User Interface.** Outil propose sur la plateforme PEGA, servant à debugger.

1. Table des figures et images utilisées

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Swimming Pool Services</title>

</head>

<body>

<header>

<img src="yourlogo.png" alt="Swimming Pool Logo">

<h1>Sotalia Pool Services</h1>

</header>

<nav>

<a href="#home">Home</a>

<a href="#services">Services</a>

<a href="#products">Products</a>

<a href="#water-treatment">Water Treatment</a>

<a href="#renovation">Renovation</a>

<a href="#shop-online">Shop Online</a>

<a href="#login">Login</a>

</nav>

<div class="banner">

<div class="search-container">

<input type="text" class="search-bar" placeholder="Enter your search">

<button class="search-button">Search</button>

</div>

<p class="welcome-paragraph">Welcome to our webpage</p>

</div>

<iv class="gallery">

<div class="gallery-item">

<img src="https://ik.imagekit.io/ikmedia/backlit.jpg" alt="Photo 1">

<div class="item-content">

<div class="item-text">Beautiful Landscape</div>

<button class="button" onclick="handleButtonClick('Button 1')">View Details</button>

</div>

</div>

<div class="gallery-item">

<img src="https://ik.imagekit.io/ikmedia/backlit.jpg" alt="Photo 2">

<div class="item-content">

<div class="item-text">Cute Animals</div>

<button class="button" onclick="handleButtonClick('Button 2')">View Details</button>

</div>

</div>

<div class="gallery-item">

<img src="https://ik.imagekit.io/ikmedia/backlit.jpg" alt="Photo 4">

<div class="item-content">

<div class="item-text">Cityscape at Night</div>

<button class="button" onclick="handleButtonClick('Button 3')">View Details</button>

</div>

</div>

<div class="gallery-item">

<img src="https://ik.imagekit.io/ikmedia/backlit.jpg" alt="Photo 3">

<div class="item-content">

<div class="item-text">Cityscape at Night</div>

<button class="button" onclick="handleButtonClick('Button 4')">View Details</button>

</div>

</div>

<div class="container">

<div class="product">

<div class="product-title">Service 1</div>

<div class="product-description">Description of Service 1. Lorem ipsum dolor sit amet, consectetur adipiscing elit.</div>

<button class="button" onclick="handleButtonClick('Service 1')">Add to Cart</button>

</div>

<div class="product">

<div class="product-title">Service 1</div>

<div class="product-description">Description of Service 2. Lorem ipsum dolor sit amet, consectetur adipiscing elit.</div>

<button class="button" onclick="handleButtonClick('Service 2')">Book Now</button>

</div>

<div class="product">

<div class="product-title">Service 3</div>

<div class="product-description">Description of Service 3. Lorem ipsum dolor sit amet, consectetur adipiscing elit.</div>

<button class="button" onclick="handleButtonClick('Service 3')">Book Now</button>

</div>

</div>

<div class="article">

<div class="overlay"></div>

<h1>Welcome to Our Article</h1>

<p>This is a sample article with a paragraph and a photo as a background. Replace this text with your own content.</p>

<img src="https://ik.imagekit.io/ikmedia/backlit.jpg" alt="Photo 4">

<div class="container">

<div class="column">

<div class="content">

<img src="https://ik.imagekit.io/ikmedia/backlit.jpg" alt="Photo 1">

<div class="label">Label 1</div>

<button class="button" onclick="handleButtonClick('Button 1')">Click Me</button>

</div>

</div>

<div class="column">

<div class="content">

<img src="https://ik.imagekit.io/ikmedia/backlit.jpg" alt="Photo 1">

<div class="label">Label 2</div>

<button class="button" onclick="handleButtonClick('Button 2')">Click Me</button>

</div>

</div>

<div class="column">

<div class="content">

<img src="https://ik.imagekit.io/ikmedia/backlit.jpg" alt="Photo 1">

<div class="label">Label 3</div>

<button class="button" onclick="handleButtonClick('Button 3')">Click Me</button>

</div>

</div>

</div>

<footer>

<div class="container">

<div class="column">

<button class="login-button">Login</button>

<button class="password-button">Password</button>

</div>

<div class="column">

<p class="contact-info">Contact us at: example@example.com</p>

</div>

</div>

<div class="social-buttons">

<a href="https://facebook.com" target="\_blank" class="social-button">Facebook</a>

<a href="https://twitter.com" target="\_blank" class="social-button">Twitter</a>

<a href="https://instagram.com" target="\_blank" class="social-button">Instagram</a>

<!-- Add more social buttons as needed -->

</div>

</footer>

</body>

</html>

Css

body {

font-family: Arial, sans-serif;

margin: 0;

padding: 0;

box-sizing: border-box;

font-family: Arial, sans-serif;

background-color: #ccffff;

background-image: url('path/to/your/background-image.jpg');

background-size: cover; /\* Ensures the image covers the entire body \*/

background-position: center; /\* Centers the background image \*/

background-repeat: no-repeat; /\* Prevents the image from repeating \*/

}

header {

background-color: #3498db;

color: white;

padding: 15px;

text-align: center;

display: flex;

align-items: center;

}

header img {

max-width: 100px;

margin-right: 10px;

}

nav {

background-color: #2980b9;

overflow: hidden;

}

nav a {

float: left;

display: block;

color: white;

text-align: center;

padding: 14px 16px;

text-decoration: none;

}

nav a:hover {

background-color: #3498db;

color: black;

}

.search-container {

margin-top: 20px;

}

.search-bar {

padding: 10px;

width: 300px;

border: none;

border-radius: 5px;

font-size: 16px;

}

.search-container {

text-align: center;

margin: 20px;

float: right;

}

.search-bar {

padding: 10px;

width: 175px;

border: none;

border-radius: 5px;

}

.search-button {

background-color: #4CAF50;

color: white;

padding: 10px 20px;

font-size: 16px;

border: none;

border-radius: 5px;

cursor: pointer;

transition: background-color 0.3s;

margin-top: 10px;

}

.search-button:hover {

background-color: #45a049;

}

.button {

display: inline-block;

padding: 10px 20px;

font-size: 16px;

text-align: center;

text-decoration: none;

border-radius: 5px;

margin: 10px;

cursor: pointer;

background-color: #2ecc71;

color: #fff;

border: none;

}

.contact-container {

margin: 20px;

padding: 20px;

background-color: white;

}

.container {

max-width: 1200px;

margin: 20px auto;

display: flex;

}

.column {

flex: 1;

box-sizing: border-box;

background-color: #ffffff;

border-radius: 8px;

box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

margin: 10px;

}

.content {

padding: 20px;

text-align: center;

}

img {

max-width: 100%;

height: auto;

border-radius: 8px;

margin-bottom: 10px;

}

.label {

font-size: 18px;

font-weight: bold;

margin-bottom: 10px;

color: #333333;

}

.button {

background-color: #4CAF50;

color: white;

padding: 10px 20px;

font-size: 16px;

border: none;

border-radius: 5px;

cursor: pointer;

transition: background-color 0.3s;

}

.button:hover {

background-color: #45a049;

}

.banner {

position: relative;

width: 100%;

height: 150px; /\* Adjust the height as needed \*/

background-image: url('https://ik.imagekit.io/ikmedia/backlit.jpg'); /\* Replace with your image URL \*/

background-size: cover;

background-position: center;

color: #0000b3;

text-align: center;

display: flex;

flex-direction: column;

justify-content: center;

align-items: center;

}

.banner-text {

font-size: 24px;

font-weight: bold;

margin-bottom: 10px;

}

.sub-text {

font-size: 16px;

}

.gallery {

max-width: 1200px;

margin: 20px auto;

display: flex;

flex-wrap: wrap;

justify-content: space-around;

}

.gallery-item {

width: 400px;

margin: 20px;

box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

overflow: hidden;

border-radius: 8px;

}

img {

max-width: 100%;

height: auto;

border-radius: 8px 8px 0 0;

}

.item-content {

padding: 15px;

text-align: center;

}

.item-text {

font-size: 16px;

margin-bottom: 10px;

color: #333333;

}

.button {

background-color: #4CAF50;

color: white;

padding: 10px 20px;

font-size: 16px;

border: none;

border-radius: 5px;

cursor: pointer;

transition: background-color 0.3s;

}

.button:hover {

background-color: #45a049;

}

.product {

margin-bottom: 20px;

border-bottom: 1px solid #dddddd;

padding-bottom: 15px;

}

.product-title {

font-size: 18px;

font-weight: bold;

color: #333333;

}

.product-description {

font-size: 14px;

color: #666666;

margin-bottom: 10px;

}

.button {

background-color: #4CAF50;

color: white;

padding: 10px 20px;

font-size: 16px;

border: none;

border-radius: 5px;

cursor: pointer;

transition: background-color 0.3s;

}

.button:hover {

background-color: #45a049;

}

/\*body box-sizing: border-box; \*/

/\*main {

padding: 20px;

}\*/

/\*article \*/

.article {

background-image: url(img src='https://ik.imagekit.io/ikmedia/backlit.jpg');

/\* Replace with your image URL \*/

background-size: cover;

background-position: center;

color: blue;

text-align: center;

padding: 40px;

/\*height: 100vh;\*/

display: flex;

flex-direction: column;

justify-content: center;

align-items: center;

width: 100%;

max-width: 100%;

height: auto;

}

.article p {

font-size: 18px;

margin-bottom: 20px;

}

/\* Optional: Add an overlay to improve readability of text \*/

/\*.overlay {

position: absolute;

top: 0;

left: 0;

width: 90%;

height: 70%;

background: rgba(0, 0, 0, 0.5);

}\*/

.social-buttons {

text-align: center;

margin-top: 20px;

}

.social-buttons a {

display: inline-block;

margin: 0 10px;

text-decoration: none;

}

.social-button {

background-color: #4CAF50;

color: white;

padding: 10px 20px;

font-size: 16px;

border: none;

border-radius: 5px;

cursor: pointer;

transition: background-color 0.3s;

}

.social-button:hover {

background-color: #45a049;

}

/\*footer/\*/

.container {

display: flex;

justify-content: center;

align-items: center;

height: 100vh;

}

.column {

text-align: center;

width: 300px;

padding: 20px;

box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

}

.login-button, .password-button {

background-color: #4CAF50;

color: white;

padding: 10px 20px;

font-size: 16px;

border: none;

border-radius: 5px;

cursor: pointer;

transition: background-color 0.3s;

margin-bottom: 20px;

}

.login-button:hover, .password-button:hover {

background-color: #45a049;

}

.contact-info {

font-size: 16px;

}