ASI – Bachelor Développeur FullStack & DevOps

Année 2023-2024

MÉMOIRE D'ACTIVITÉ

L’optimisation des processus de développement avec la conteneurisation et l’intégration continue.

Comment optimiser le processus d'installation et de déploiement d'un projet web au sein de mon entreprise ?

Rédigé et soutenu par : M. Sacha BELIOT

Maître d’apprentissage : M. Loïc MANGOLD

Supervisé par le Référent : M. Nathan LAURY

Établissement / Formation : École IPSSI

Entreprise d’accueil : Studio.gd

CERTIFICAT D'AUTHENTICITE DE MON MEMOIRE

Ce document rempli et signé par l'étudiant doit être inséré dans tous les mémoires après la page de garde

Je, soussigné(e) : NOM et Prénom
BELIOT Sacha

Etudiant(e) de : Formation et année
Bachelor Développeur FullStack & DevOps - Année 2023/2024

Etablissement :
École IPSSI

Certifie que le texte soumis ne comporte aucun passage ou schéma copié sans qu'il soit explicitement fait référence aux sources.

Certifie ne pas avoir dépassé le taux de plagiat autorisé (15% maximum)

Fait à Crécy-la-Chapelle le 03/07/2024

Signature :

École IPSSI : INSTITUT PRIVÉ SUPÉRIEUR DES SYSTÈMES D'INFORMATION
SARL au capital social de 105 000 € - RCS PARIS : 420 793 705– CODE NAF 8559A - n° d'activité 11753362175 - TVA FR 41 420 793 705 – UAI 075724P
Paris : 25, rue Claude Tillier – BP 20024 - 75560 Paris Cedex 12 - Tél : 01.55.43.26.65
Marne-la Vallée : 14, avenue de l'Europe – 77144 Montévrain – Tél. : 01.81.80.77.10
SQY : 8, rue Germain Soufflot 78180 Montigny-le-Bretonneux – Tél : 01.81.80.38.50 - RCS Versailles : 852 544 923 – n° d'activité 11788427678

Remerciements

Je tiens à exprimer ma gratitude à toutes les personnes qui ont contribué à la réalisation de ce mémoire et qui m'ont accompagné tout au long de mon année scolaire et de mon alternance.

Tout d'abord, je remercie chaleureusement l'équipe pédagogique de l'IPSSI ainsi que tous les intervenants pour leurs précieux conseils, leurs connaissances partagées et leur accompagnement durant cette formation. Je suis particulièrement reconnaissant envers M. Nathan Laury, référent pédagogique, pour son soutien et sa réactivité face à mes interrogations, qui ont grandement facilité la rédaction de ce mémoire. Je remercie également Mme Noëmie Da Rocha, coordinatrice pédagogique, pour sa disponibilité et son aide précieuse.

Je souhaite aussi exprimer ma reconnaissance à l'entreprise Studio.gd de m'avoir offert l'opportunité de réaliser mon alternance au sein de leur structure et pour le soutien apporté à mon développement professionnel. Mes sincères remerciements vont à mes collègues de Studio.gd pour leur accueil chaleureux et leur aide constante. L'ambiance de travail agréable et la disponibilité de mes collègues ont été d'une grande aide durant cette année.

Je tiens tout particulièrement à remercier M. Loïc Mangold, mon maître d'apprentissage, pour sa confiance et son soutien tout au long de cette alternance.

Enfin, je souhaite remercier tous mes camarades de classe du groupe MLV BD 24.1 pour leur soutien continu pendant les cours. Leur esprit d'équipe et l'ambiance conviviale qu'ils ont créée ont grandement contribué à mon apprentissage et à mon épanouissement personnel durant cette formation.

Résumé

Ce mémoire explore la mise en place de Docker et des pipelines CI/CD sur un projet web au sein de l'entreprise Studio.gd, spécialisée dans le développement web. Le projet porte sur l'amélioration des processus de développement et de déploiement pour les projets Drupal, en utilisant Docker pour standardiser les environnements de développement et GitLab CI/CD pour automatiser les déploiements.

L'utilisation de Docker permet de créer des environnements de développement reproductibles, cohérents et isolés, facilitant ainsi la gestion des dépendances et la configuration des projets. Docker améliore également la rapidité et l'efficacité du développement en réduisant le temps d'installation du projet sur l'ordinateur de chacun de mes collègues.

Par ailleurs, l'implémentation des pipelines CI/CD avec GitLab assure une intégration continue et un déploiement continu, ce qui réduit les erreurs humaines, accélère les cycles de livraison et renforce la sécurité des déploiements. Ces outils modernes offrent une meilleure gestion des évolutions et des mises à jour, tout en améliorant la qualité et la fiabilité des livrables.

Ce mémoire démontre que l'adoption de Docker et des pipelines CI/CD au sein de Studio.gd optimise considérablement le processus de développement et de déploiement, apportant des avantages significatifs en termes de productivité, de qualité et de réactivité aux besoins des clients.

Abstract

This thesis explores the implementation of Docker and CI/CD pipelines on a web project within the company Studio.gd, which specializes in web development. The project focuses on improving development and deployment processes for Drupal projects by using Docker to standardize development environments and GitLab CI/CD to automate deployments.

The use of Docker allows for the creation of reproducible, consistent, and isolated development environments, thereby facilitating dependency management and project configuration. Docker also enhances the speed and efficiency of development by reducing the time needed to set up the project on each of my colleagues' computers.

Furthermore, the implementation of CI/CD pipelines with GitLab ensures continuous integration and continuous deployment, reducing human errors, speeding up delivery cycles, and strengthening deployment security. These modern tools offer better management of updates and upgrades while improving the quality and reliability of deliverables.

This thesis demonstrates that adopting Docker and CI/CD pipelines within Studio.gd significantly optimizes the development and deployment process, providing substantial benefits in terms of productivity, quality, and responsiveness to client needs.

Sommaire

Liste des abréviations

API : Application Programming Interface

CDI : Contrat à Durée Indéterminée

CD : Continuous Delivery

CI : Continuous Integration

CIM : Configuration Import

CR : Cache Rebuild

CSS : Cascading Style Sheets

HTML : Hypertext Markup Language

HTTP : Hypertext Transfer Protocol

JS : JavaScript

MAMP : Macintosh Apache MySQL PHP/Perl/Python

PDO : PHP Data Objects

PHP : PHP Hypertext Preprocessor

PHP-FPM : PHP FastCGI Process Manager

PME : Petite et Moyenne Entreprise

SARL : Société à Responsabilité Limitée

SQL : Structured Query Language

SSH : Secure Shell

TMA : Tierce Maintenance Applicative

UPDB : Update Database

URL : Uniform Resource Locator

WAMP : Windows Apache MySQL PHP/Perl/Python

XAMPP : Cross-platform Apache MariaDB Perl PHP

YAML : Yet Another Markup Language

Introduction et problématique

Dans un environnement professionnel en constante évolution, l'optimisation des processus de développement revêt une importance cruciale pour assurer la compétitivité et la viabilité des entreprises. Au sein de Studio.gd, une société dédiée au développement de sites web pour une clientèle variée, la capacité de pouvoir installer et déployer rapidement les projets web est une priorité stratégique.

Au cœur des préoccupations de Studio.gd se trouve une problématique fondamentale : **Comment optimiser le processus d'installation et de déploiement d'un projet web au sein de mon entreprise**, afin de garantir une efficacité optimale et une réactivité accrue face aux demandes des clients et des partenaires ? Cette question soulève plusieurs enjeux majeurs au sein de l'entreprise.

Tout d'abord, il est impératif de réduire les délais d'installation des projets web, afin de minimiser les temps morts et d'optimiser la productivité des équipes de développement. Cependant, la diversité des configurations utilisées par chaque développeur, notamment en termes de système d'exploitation, de versions des services nécessaires et d'extensions spécifiques, constitue un défi supplémentaire.

Ensuite, il est essentiel de pouvoir déployer rapidement les projets web en environnement de pré-production, hébergé sur les serveurs internes de l'entreprise, afin de permettre des tests approfondis et une validation rigoureuse avant la mise en production.

Enfin, il est crucial de garantir la fiabilité et la stabilité des déploiements, en minimisant les risques d'erreurs et de dysfonctionnements susceptibles de compromettre la qualité des livrables et la satisfaction des clients.

Dans ce contexte, la décision stratégique d'adopter la conteneurisation, l'intégration et le déploiement continu s'avère être une réponse pertinente et efficace aux défis rencontrés par Studio.gd. En conteneurisant les services nécessaires à chaque

projet web à l'aide de Docker et en les intégrant dans le processus d'intégration et de déploiement continu via GitLab, l'entreprise vise à accélérer significativement le déploiement des projets tout en limitant les risques d'erreurs humaines.

Ainsi, cette problématique met en lumière la nécessité pour l'entreprise Studio.gd d'adopter des approches innovantes et efficaces pour optimiser ses processus d'installation et de déploiement des projets web, dans le but de garantir une satisfaction client maximale et une compétitivité durable sur le marché concurrentiel du développement web.

Partie 1 : Le Sujet

Dans cette première partie, nous plongerons dans une analyse approfondie du sujet de l'optimisation des processus d'installation et de déploiement des projets web au sein de l'entreprise Studio.gd. Nous aborderons les concepts clés et les définitions essentielles pour comprendre pleinement les enjeux.

1.1 Introduction aux processus d'installation et de déploiement

Les processus d'installation et de déploiement des projets web sont des étapes cruciales dans le cycle de vie du développement logiciel. L'installation fait référence à la configuration initiale d'une application ou d'un site web sur un environnement spécifique, tandis que le déploiement implique le transfert de cette application configurée vers un serveur de production ou de pré-production afin de réaliser des tests ou pour une utilisation réelle. Ces deux processus doivent être réalisés de manière efficace pour garantir le bon fonctionnement et la disponibilité continue des applications web.

L'installation d'un projet web comprend généralement plusieurs étapes, telles que la configuration du serveur web, l'installation des dépendances logicielles nécessaires, la mise en place de la base de données et la configuration des paramètres spécifiques à l'application. Ce processus peut être complexe, en particulier dans les environnements où plusieurs technologies et composants logiciels doivent être pris en compte. De plus, les développeurs doivent souvent faire face à des défis liés à la diversité des environnements de développement, chaque membre de l'équipe pouvant utiliser un système d'exploitation différent, des versions logicielles différentes ou des configurations spécifiques.

Après avoir achevé l'installation, le déploiement des évolutions ou des mises à jour effectuées sur le projet web implique le transfert du code développé vers le serveur de pré-production ou de production. Ce processus implique généralement une connexion manuelle au serveur hébergeant l'environnement concerné pour récupérer le code récemment développé. Ensuite, des actions telles que la

réinstallation des dépendances, le vidage du cache du site, l'importation des configurations ou la compilation des styles peuvent être nécessaires. Un déploiement réussi assure que l'application fonctionne correctement et répond aux exigences fonctionnelles et techniques définies.

Les processus d'installation et de déploiement des projets web peuvent être sources de nombreux défis pour les équipes de développement. Les délais prolongés, les erreurs humaines, les conflits de configuration et les incompatibilités entre les environnements sont autant de problèmes potentiels susceptibles de retarder les projets et d'entraver la productivité. Dans ce contexte, l'automatisation et l'optimisation de ces processus deviennent essentielles pour garantir une efficacité maximale et une réactivité accrue face aux demandes des clients et des partenaires.

1.2 Enjeux et défis de l'installation et du déploiement

Les processus d'installation et de déploiement des projets web sont confrontés à divers enjeux et défis qui peuvent entraver leur efficacité et leur fiabilité. Comprendre ces défis est essentiel pour élaborer des solutions efficaces et optimiser ces processus critiques dans le cycle de développement logiciel.

Complexité des environnements

La diversité des environnements de développement, de pré-production et de production peut compliquer le processus d'installation de projets web. Chaque environnement peut avoir sa propre configuration, son système d'exploitation, ses versions logicielles et ses paramètres de sécurité, ce qui rend difficile la création d'un processus unique et homogène.

Gestion des dépendances

Les projets web dépendent souvent de nombreuses bibliothèques, frameworks et modules externes afin de fonctionner correctement. Gérer ces dépendances et s'assurer de leur disponibilité et de leur compatibilité lors de l'installation et du

déploiement peut être un défi, en particulier pour les projets de grande envergure avec de nombreuses parties prenantes.

Temps de configuration

La configuration manuelle des serveurs virtuels, des bases de données, des paramètres d'application et d'autres composants lors de l'installation et du déploiement peut être une tâche fastidieuse et sujette aux erreurs. Les délais prolongés associés à cette configuration manuelle peuvent retarder les projets et affecter la productivité de l'équipe de développement.

Gestion des versions et des mises à jour

La gestion des versions logicielles et des mises à jour des composants peut poser des problèmes lors de l'installation et du déploiement des projets web. Assurer la compatibilité entre les différentes versions de logiciels, ainsi que la gestion des correctifs de sécurité et des mises à jour fonctionnelles, est cruciale pour garantir la stabilité et la sécurité des applications déployées.

Réactivité et agilité

Dans un environnement concurrentiel, la capacité à déployer rapidement de nouvelles fonctionnalités et à réagir aux changements des exigences clients est essentielle. Les processus d'installation et de déploiement doivent être suffisamment flexibles et agiles pour s'adapter rapidement aux besoins changeants du marché et des utilisateurs.

En résumé, les enjeux et défis liés à l'installation et au déploiement des projets web nécessitent une approche proactive et innovante pour surmonter ces obstacles et garantir la réussite des projets. L'adoption de solutions telles que la conteneurisation, l'intégration et le déploiement continu peuvent jouer un rôle crucial dans l'optimisation de ces processus critiques.

1.3 Risques et limitations des solutions choisis

Cette section explore les différents aspects des solutions permettant l'optimisation des processus d'installation et de déploiement, tels que la sécurité et la conformité, la gestion de la résistance au changement, la standardisation des environnements, l'automatisation des tâches répétitives, ainsi que la formation et la montée en compétences des équipes. En abordant ces éléments clés, les organisations peuvent non seulement mieux gérer l’adoption de nouvelles solutions, mais aussi encourager une culture de l'innovation.

Dépendance aux outils et technologies

L'introduction de nouvelles technologies peut rendre les organisations dépendantes de ces outils pour le bon fonctionnement de leurs processus d'installation et de déploiement. En cas de panne ou de dysfonctionnement, cela peut entraîner des retards dans les livraisons et affecter la productivité des équipes de développement. De plus, la formation et la montée en compétences des équipes sur ces nouvelles technologies peuvent nécessiter des investissements importants en temps et en ressources.

Sécurité et conformité

L'automatisation des processus de déploiement peut soulever des préoccupations en matière de sécurité et de conformité. Les configurations automatisées sont susceptibles de présenter des vulnérabilités de sécurité si les images ne sont pas régulièrement mises à jour et patchées. De plus, l'automatisation des déploiements peut entraîner des failles de sécurité si les processus ne sont pas correctement sécurisés et contrôlés. Il est donc essentiel de mettre en place des mesures de sécurité et des politiques de conformité strictes pour atténuer ces risques potentiels.

Résistance au changement

La transition vers de nouvelles pratiques peut rencontrer une certaine résistance au sein des équipes. Les membres de l'équipe peuvent craindre de perdre le contrôle sur leurs processus de travail ou de devoir acquérir de nouvelles compétences techniques. De plus, les dirigeants peuvent hésiter à investir dans de nouvelles technologies et pratiques sans garantie de retour sur investissement immédiat. Il est donc important de sensibiliser et d'impliquer toutes les parties prenantes dès le début du processus de transformation pour assurer une adoption réussie et une intégration harmonieuse des nouvelles pratiques.

Standardisation des environnements

Une approche essentielle pour simplifier et rationaliser les processus d'installation et de déploiement est de standardiser les environnements de développement, de test et de production. Cela implique d'adopter les mêmes systèmes d'exploitation sur toutes les machines et d’uniformiser les configurations et les versions logicielles afin qu’elles soient cohérentes à travers tous les environnements, ce qui facilite le transfert des applications entre ces environnements sans nécessiter de modifications majeures. La standardisation des environnements permet également de réduire les erreurs liées aux configurations incompatibles ou manquantes.

Automatisation des tâches répétitives

L'automatisation des tâches répétitives est un pilier fondamental des processus d'installation et de déploiement efficaces. En utilisant des outils d'automatisation, les organisations peuvent réduire les erreurs humaines, accélérer les déploiements et améliorer la cohérence des configurations. L'automatisation permet également d'optimiser les ressources et de libérer du temps pour les équipes de développement, en leur permettant de se concentrer sur des tâches à plus forte valeur ajoutée.

Formation et montée en compétences

Investir dans la formation et le développement des compétences des équipes est crucial pour garantir le succès des processus d'installation et de déploiement. En fournissant une formation continue sur les technologies émergentes, les pratiques DevOps et les bonnes pratiques de développement, les organisations peuvent renforcer les compétences de leurs équipes et favoriser une culture d'apprentissage et d'amélioration continue. De plus, en encourageant la collaboration et le partage des connaissances entre les équipes, les organisations peuvent tirer parti de l'expertise collective pour résoudre les problèmes et innover plus efficacement.

Partie 2 : Cas Réel

2.1 Présentation de l'entreprise Studio.gd

Studio.gd est une entreprise spécialisée dans le design et le développement de sites et d'applications web et mobile pour une clientèle variée. Fondée en 2011 par Antonin Brunon et Sylvain Baronnet. Sa clientèle peut aller des petites entreprises locales tels que des restaurants, des campings ou des associations sportives, aux grandes organisations nationales et internationales comme Quick, Total ou le Crédit Agricole.

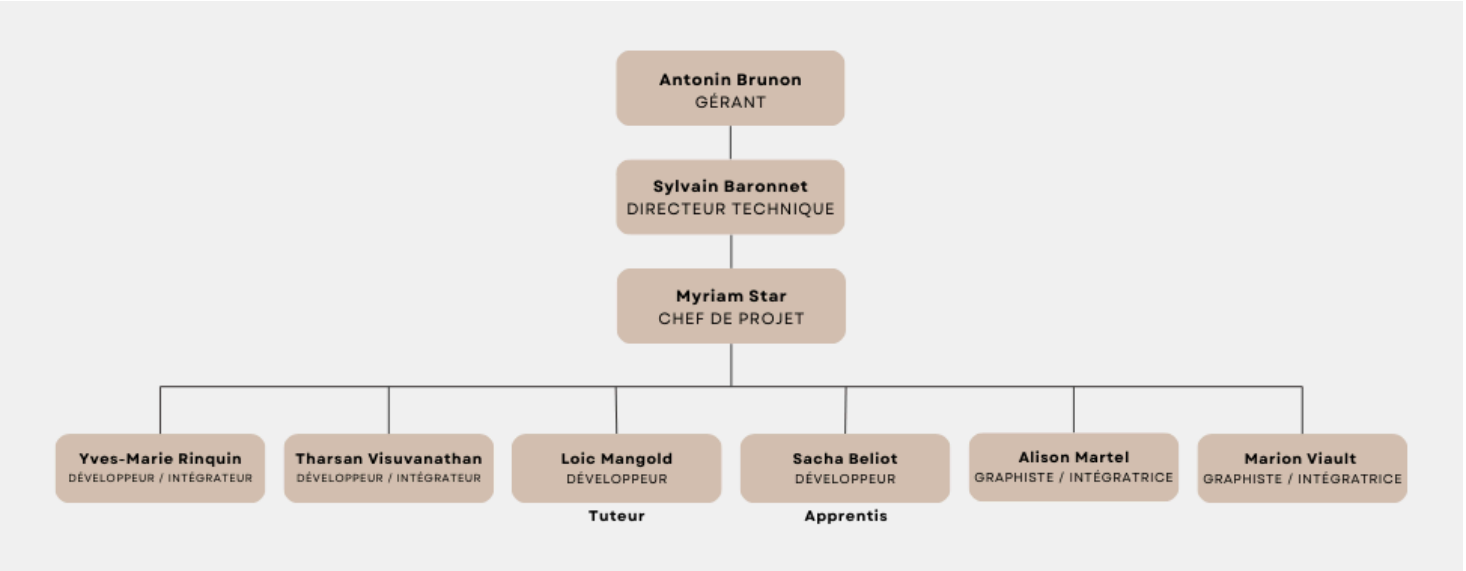
Contexte et secteur d'activité

Situé au cœur de la région d'Île-de-France, Studio.gd opère dans le secteur dynamique et concurrentiel du développement web, offrant ses services à une clientèle allant des petites entreprises locales aux grandes organisations nationales et internationales. L'entreprise se distingue par son expertise dans diverses technologies telles que Drupal, WordPress, Symfony et React Native, permettant ainsi de répondre aux besoins variés de ses clients.

Taille et structure organisationnelle

Studio.gd est une PME à responsabilité limitée (SARL) comptant actuellement 10 employés. L'entreprise se caractérise par une structure organisationnelle agile et collaborative, favorisant la communication et la responsabilisation au sein des équipes. L'organigramme ci-dessous présente la composition de l'équipe, comprenant des développeurs, des intégrateurs, des graphistes et des chefs de projet.

Organigramme de l’entreprise Studio.gd



(Annexe 1 : Organigramme de l’entreprise Studio.gd)

2.2 Mon poste et mes missions chez Studio.gd

En tant que développeur chez Studio.gd, j'ai eu l'opportunité de travailler sur une variété de projets, allant de la maintenance et l'installation de nouvelles fonctionnalités sur des sites existants, en passant par le développement complet de sites web.

J'ai collaboré étroitement avec Loïc Mangold, un développeur expérimenté spécialisé en développement Drupal, sur des projets pour y faire de la maintenance évolutive (TMA) tels que sur la MMJ, mutuelle santé et prévoyance, Total, Axens, et le site de l’armée de l’air “Devenir aviateur”. Ces projets m'ont permis d'acquérir une expertise approfondie dans le développement Drupal et de comprendre les enjeux spécifiques liés à la gestion de projets de grande envergure.

Parallèlement, j'ai également travaillé sur des projets de développement complet de sites web pour des petites entreprises locales, comme le restaurant “Le Flambadou” et l’association sportive “Poissy Volley”, en utilisant la technologie WordPress. Ces projets m'ont donné l'occasion de mettre en pratique mes compétences de

développement front-end et back-end, ainsi que de travailler de manière autonome sur l'ensemble du processus de développement, de la conception à la livraison.

En plus de mes missions de développement, j'ai également contribué à des projets de DevOps, notamment en participant à l'installation et à la configuration des environnements de pré-production sur nos serveurs internes, ainsi qu'à la mise en place de pipelines d'intégration continue pour automatiser le déploiement d'un projet en pré-production et en production.

2.3 Difficultés rencontrés

Dans le contexte de Studio.gd, les membres de l'équipe sont confrontés à des difficultés spécifiques liées aux processus d'installation et de déploiement des projets web, qui ont un impact direct sur l'efficacité opérationnelle et la qualité des livrables. Deux difficultés majeures émergent : la diversité des environnements de développement et les inefficacités liées au déploiement manuel sur les environnements de pré-production et de production.

L'une des principales difficultés rencontrées par les membres de l'équipe est la diversité des environnements de développement. Chaque développeur peut utiliser un système d'exploitation différent, des versions de logiciels variées et des configurations spécifiques, ce qui rend difficile la création d'un environnement de développement homogène. Cette diversité entraîne des incohérences et des incompatibilités entre les environnements, ce qui peut ralentir le processus de développement et augmenter les risques d'erreurs lors du déploiement.

Un autre défi majeur est le temps perdu et les erreurs potentielles associées au déploiement manuel des projets sur les environnements de pré-production et de production. Les déploiements manuels nécessitent une intervention humaine pour transférer le code, mettre à jour les configurations et effectuer d'autres tâches nécessaires, ce qui peut être fastidieux et sujet aux erreurs. Les retards dans les déploiements et les erreurs de configuration peuvent entraîner des interruptions de service, des bugs et une insatisfaction client.

Ces difficultés ont un impact direct sur la productivité et la satisfaction client de Studio.gd. Les retards dans les déploiements et les erreurs de configuration peuvent entraîner des retards dans la livraison des projets, des coûts supplémentaires et une réduction de la qualité des livrables. De plus, la diversité des environnements de développement peut rendre la collaboration et le partage de code entre les membres de l'équipe plus difficiles, ce qui peut ralentir le développement et augmenter les risques d'incohérences et d'erreurs.

Partie 3 : Solution

Dans cette section, nous nous pencherons sur les diverses solutions visant à optimiser les processus d'installation et de déploiement d'un projet Drupal. Pour mieux cerner les défis auxquels nous sommes confrontés, nous examinerons les services nécessaires au fonctionnement d'un site Drupal dans un environnement local, ainsi que les étapes requises pour déployer ce site dans un environnement de pré-production ou de production. Ensuite, nous passerons en revue les différentes options disponibles pour résoudre ces problèmes, en sélectionnant une solution pour chaque, et en expliquant les raisons de ces choix. Enfin, nous décrirons la mise en place de ces solutions et leur fonctionnement.

3.1 L'environnement de développement Drupal

Afin de travailler sur un projet Drupal, il est essentiel de mettre en place un environnement de développement local bien configuré. Cet environnement repose sur plusieurs composants clés qui travaillent ensemble pour offrir une plateforme de travail. Parmi ces composants, Nginx, PHP, et MySQL/MariaDB jouent des rôles cruciaux en assurant respectivement le traitement des requêtes web, l'exécution du code PHP, et la gestion des données. Dans les sections suivantes, nous expliquerons l'importance et la configuration de chaque composant dans le contexte d'un environnement de développement Drupal.

Nginx

Nginx est un serveur web qui joue un rôle essentiel dans l'hébergement des sites web Drupal en environnement de développement local. Contrairement à Apache, Nginx est reconnu pour sa faible consommation de ressources système et sa capacité à gérer de grandes quantités de trafic web de manière efficace. Dans le contexte de Drupal, Nginx assure la réception des requêtes HTTP des navigateurs web et la fourniture des réponses appropriées en exécutant les scripts PHP associés. Il sert également de point d'entrée pour gérer les URL, les paramètres de configuration et les autorisations d'accès.

En intégrant les Virtual Hosts directement à Nginx, nous pouvons créer des environnements isolés pour chaque projet Drupal en cours de développement. Les Virtual Hosts permettent de configurer plusieurs sites web sur une seule machine physique, chacun ayant son propre nom de domaine virtuel et son propre répertoire racine. Cette approche offre une grande flexibilité et permet de simuler efficacement un environnement pour chaque site Drupal en développement.

PHP

PHP, acronyme de "PHP Hypertext Preprocessor", est le langage de script côté serveur qui alimente le cœur de Drupal. Il fournit la logique sous-jacente qui permet à Drupal de fonctionner en traitant les requêtes des utilisateurs et en générant les pages web dynamiquement. En outre, PHP-FPM, acronyme de "PHP Hypertext Preprocessor FastCGI Process Manager" est un gestionnaire de processus PHP qui permet de traiter les scripts PHP de manière rapide et efficace, en les exécutant de manière asynchrone et en gérant des pools de processus PHP distincts pour chaque site web hébergé sur le serveur. Cette isolation assure que les problèmes sur un site n'affectent pas les autres.

Chaque projet web peut être codé avec une version spécifique de PHP-FPM, telles que PHP-FPM 7.4 ou PHP-FPM 8.1, et peut nécessiter différentes extensions pour fonctionner correctement. Il est donc crucial d'avoir plusieurs versions de PHP-FPM installées sur son ordinateur, ainsi que toutes les extensions les plus importantes, afin de pouvoir spécifier la version appropriée pour chaque projet.

Pour simplifier la gestion des différentes versions de PHP-FPM, nous avons configuré les fichiers de configuration spécifique pour chaque version de PHP-FPM. Dans ces fichiers, nous avons défini des ports qui feront fonctionner chaque version de PHP-FPM de façon indépendante et en simultané. Par exemple, dans le fichier de configuration de PHP-FPM 7.4, on définit le port 9074 et dans le fichier de configuration de PHP-FPM 8.1, on définit le port 9081.

Ensuite, lors de la configuration des Virtual Hosts, nous devons spécifier la version de PHP-FPM utilisée pour chaque projet. Cela se fait en indiquant le chemin vers le port correspondant à la version souhaitée. Par exemple, si l'on veut la version de PHP-FPM 8.1, on mettra le port 9081 dans le fichier du Virtual Host. Cette approche nous permet de gérer efficacement les différentes versions de PHP-FPM et d'assurer la compatibilité avec les exigences spécifiques de chaque projet Drupal.

MySQL/MariaDB

Les systèmes de gestion de bases de données, telles que MySQL ou MariaDB, sont des éléments clés dans le stockage et la gestion des données d'un site Drupal. Ils sont utilisés pour créer des bases de données qui stockeront une grande variété d'informations, telles que des informations sur les utilisateurs et leurs permissions, les types et champs de contenu, les taxonomies, la structure des menus, les blocs et leur emplacement, ainsi que les configurations globales du site. Drupal utilise une architecture de base de données relationnelle pour organiser ces données de manière efficace. Les bases de données MySQL/MariaDB permettent aux développeurs d'effectuer des requêtes, de récupérer des informations spécifiques et de maintenir l'intégrité des données tout au long du processus de développement.

En conclusion, ces services fondamentaux - PHP, Nginx, MySQL/MariaDB - sont indispensables pour créer un environnement de développement local pour les projets Drupal.

3.2 Différentes solutions : Optimisation du temps d'installation en local

L'optimisation du temps d'installation d'un projet drupal en local est cruciale pour garantir une productivité maximale des développeurs tout au long du cycle de développement. Dans cette section, nous explorerons différentes solutions afin d'accélérer ce processus, en mettant l'accent sur les outils et les pratiques qui

permettent de réduire le temps nécessaire pour configurer et mettre en place un environnement de développement fonctionnel.

Solution 1 : Installation manuelle

La méthode classique d'installation d'un environnement de développement implique souvent une série d'étapes manuelles fastidieuses. Cela comprend le téléchargement et l'installation de chaque composant qui ne sont pas déjà installés ou installé avec une mauvaise version, ou ne comprenant pas les bonnes extensions, tels que PHP, Apache/Nginx et MySQL/MariaDB, suivis de leur configuration respective, la création et la configuration d'un Virtual Host et la modification du fichier des variables d'environnement afin de l'adapter à son environnement local. Mais il y a aussi l'installation de la base de données, l'installation des dépendances PHP, l'importation des configurations Drupal, la reconstruction des caches, l'installation des dépendances JavaScript et la compilation des fichiers assets du site. Bien que cette approche soit courante, elle peut être chronophage et sujette aux erreurs humaines.

Solution 2 : Utilisation de bundles préconfigurés

Une approche alternative consiste à utiliser des bundles préconfigurés tels que WAMP pour Windows, XAMPP pour plusieurs plates-formes, ou MAMP pour Mac. Ces bundles regroupent généralement des composants tels que le serveur web Apache, le système de gestion de base de données MySQL ou MariaDB, et le langage de script PHP dans un seul package. Ils sont conçus pour simplifier le processus d'installation en fournissant une solution tout-en-un prête à l'emploi. Cependant, ils peuvent parfois inclure des composants superflus ou des configurations par défaut inadaptées aux besoins spécifiques du projet, nécessitant parfois des ajustements ou des désactivations pour correspondre exactement aux exigences du projet Drupal.

Solution 3 : Utilisation de solutions de virtualisation

Dans le cadre de la gestion des environnements de développement, la virtualisation avec des outils comme VirtualBox ou VMware offre une approche flexible. Les développeurs peuvent créer des machines virtuelles dédiées pour chaque projet, permettant ainsi une isolation efficace des environnements et une personnalisation selon les besoins spécifiques. Cependant, cette approche peut présenter des inconvénients, notamment en termes de complexité de configuration, de consommation de ressources système et de maintenance. Bien que la virtualisation soit adaptée aux projets nécessitant des configurations complexes, elle peut nécessiter des connaissances approfondies et des ressources système importantes.

Solution 4 : Utilisation de plateformes de développement cloud

Certaines plateformes de développement cloud, comme Platform.sh ou Pantheon, proposent des environnements de développement Drupal prêts à l'emploi, permettant un déploiement rapide en quelques clics. Ces solutions automatisent la configuration de l'infrastructure sous-jacente, libérant ainsi les développeurs des tâches de gestion des serveurs pour se concentrer pleinement sur le développement. Cependant, l'adoption de ces plateformes peut entraîner des coûts supplémentaires et limiter la flexibilité par rapport aux solutions d'auto-hébergement, notamment en termes de personnalisation des configurations et de contrôle sur l'infrastructure.

Solution 5 : Conteneurisation avec Docker

Une solution de plus en plus populaire pour optimiser le temps d'installation en local est la conteneurisation avec Docker. Docker permet aux développeurs de créer des conteneurs qui encapsulent une application et ses dépendances, garantissant ainsi une uniformité et une reproductibilité de l'environnement de développement. De plus, la gestion des conteneurs Docker est simplifiée grâce à des outils tels que Docker Compose, qui permettent de définir et de gérer des environnements multi-conteneurs avec un seul fichier de configuration.

En conclusion, diverses solutions peuvent être envisagées pour accélérer le processus d'installation en local et augmenter l'efficacité des développeurs Drupal. Chacune présente ses propres avantages et inconvénients, mais la conteneurisation avec Docker se démarque comme une option séduisante en raison de sa simplicité, de sa flexibilité et de sa portabilité ainsi que de son faible coût. De plus, Docker offre une gestion des dépendances simplifiée, une reproductibilité garantie grâce à la portabilité des conteneurs, ainsi qu'une vaste communauté et une large adoption dans ce secteur, fournissant un écosystème riche en outils et en ressources pour les développeurs Drupal.

3.3 Mise en place de la solution Docker

Dans cette partie, nous allons explorer en détail Docker, une technologie qui a radicalement changé la façon dont nous concevons, développons et déployons des applications web. Docker permet de créer des environnements de développement cohérents et reproductibles en encapsulant les applications et leurs dépendances dans des conteneurs légers et isolés. Nous examinerons les concepts clés de Docker, en expliquant comment chacun contribue à la gestion efficace des environnements de développement pour les sites web Drupal. Puis nous détaillerons la mise en place d’un Docker pour un projet Drupal en particulier puis sa mise en fonctionnement.

3.3.1 Comprendre Docker et ses composants

Dans cette section, nous nous pencherons sur les concepts fondamentaux de Docker, notamment les conteneurs, les images, les volumes, les réseaux et Docker Compose. Chacun de ces éléments joue un rôle crucial dans la création d'environnements de développement robustes et reproductibles, permettant une gestion efficace et fluide des applications complexes comme les sites web basés sur Drupal.

Les Conteneurs

Un conteneur est une unité standard de logiciel qui empaquette le code et toutes ses dépendances, permettant à une application de fonctionner rapidement et de manière fiable d'un environnement informatique à un autre. Contrairement aux machines virtuelles, les conteneurs ne contiennent pas de système d'exploitation complet mais partagent le noyau de l'hôte, ce qui les rend plus légers et plus rapides à démarrer. Les conteneurs Docker permettent une isolation des processus et des fichiers systèmes, garantissant que les environnements de développement et de production sont identiques.

Les Images

Les images Docker sont des modèles immuables qui définissent le contenu et les configurations nécessaires pour exécuter une application dans un conteneur. Une image peut être considérée comme un instantané du système de fichiers et des paramètres requis pour exécuter le conteneur. Les images sont construites à partir de fichiers Dockerfile, qui contiennent une série d'instructions décrivant comment construire l'image, telles que l'installation des dépendances, la configuration des services et la copie des fichiers nécessaires. Une fois l'image créée, elle peut être utilisée pour lancer de multiples conteneurs identiques.

Les Volumes

Les volumes Docker sont utilisés pour stocker les données générées et utilisées par les conteneurs. Ils permettent de persister les données indépendamment du cycle de vie des conteneurs, assurant que les informations importantes ne sont pas perdues lors de la suppression ou de la mise à jour des conteneurs. Les volumes sont particulièrement utiles pour les bases de données, les fichiers de configuration et les logs, car ils permettent de séparer les données de l'application du conteneur lui-même. Les volumes peuvent être partagés entre plusieurs conteneurs, facilitant ainsi la gestion des données dans des environnements multi-conteneurs.

Les Network

Docker Network est un composant essentiel de la plateforme Docker qui permet de gérer la communication entre différents conteneurs ainsi qu'entre les conteneurs et l'hôte. Il fournit une infrastructure pour connecter les conteneurs Docker de manière sécurisée et isolée, facilitant la communication entre les différents services d'une application distribuée. Les réseaux Docker peuvent être configurés de plusieurs manières pour répondre à divers besoins de connectivité et de sécurité.

Lorsque vous créez un réseau Docker, vous pouvez choisir parmi plusieurs pilotes de réseau, chacun ayant des fonctionnalités spécifiques. Les pilotes de réseau les plus couramment utilisés sont le bridge, le host, et l'overlay. Le pilote bridge est le plus utilisé par défaut et permet de créer un réseau privé auquel les conteneurs connectés peuvent communiquer entre eux, mais ils sont isolés du reste du réseau de l'hôte. Le pilote host supprime l'isolation réseau entre le conteneur et l'hôte Docker, permettant au conteneur d'utiliser directement le réseau de l'hôte. Enfin, le pilote overlay permet de créer des réseaux qui s'étendent sur plusieurs hôtes Docker, ce qui est particulièrement utile pour les applications réparties sur plusieurs machines dans un cluster Docker Swarm.

Les réseaux Docker sont configurés à l'aide de commandes Docker CLI ou via des fichiers de configuration Docker Compose. Par exemple, dans un fichier "docker-compose.yml", les réseaux sont définis sous la section "networks". Chaque service peut être connecté à un ou plusieurs réseaux, ce qui facilite la segmentation et la sécurisation de l'application en contrôlant quelles parties de l'application peuvent communiquer entre elles.

Docker Compose

Docker Compose est un outil permettant de définir et de gérer des applications multi-conteneurs. Avec Docker Compose, vous pouvez utiliser un fichier YAML pour configurer les services, les réseaux et les volumes nécessaires pour votre application. Cela simplifie considérablement la gestion des environnements complexes, car toutes les configurations peuvent être centralisées et versionnées

dans un seul fichier. Docker Compose permet de lancer et de gérer tous les conteneurs définis dans le fichier YAML avec une seule commande, facilitant ainsi le développement, le test et le déploiement des applications.

3.3.2 Implémentation de Docker pour le projet Drupal Axens

Axens est une société spécialisée dans la fourniture de technologies avancées et de services pour les secteurs de l'énergie et de la chimie. Nous travaillons actuellement sur le site web de la société Axens, axens.net, en tant que développeurs Drupal. Pour optimiser notre flux de travail et garantir des environnements de développement cohérents, nous avons décidé d'utiliser Docker pour ce projet.

Arborescence du Docker et description des dossiers et fichiers clés (Cf Annexe 2)

L'organisation du projet Docker pour Axens est essentielle pour garantir une configuration claire et fonctionnelle. Voici une description détaillée de l'arborescence du projet Docker ainsi que des dossiers et fichiers clés.

Le fichier "docker-compose.yml", situé à la racine du dossier Docker, contient la configuration nécessaire pour orchestrer les différents services essentiels au bon fonctionnement du site Drupal. Il définit les conteneurs, les réseaux, les volumes et les dépendances nécessaires.

Le dossier "dump/" abrite les sauvegardes de la base de données. Dans ce cas précis, il contient un fichier nommé "axens.sql.gz", qui est une sauvegarde compressée de la base de données MariaDB.

Le fichier ".env", également à la racine du dossier Docker, regroupe les variables d'environnement nécessaires à la configuration des conteneurs. Il inclut des informations cruciales telles que les identifiants de base de données et les clés API.

Le dossier "nginx/" contient les fichiers de configuration pour le serveur web Nginx. Le fichier clé ici est "axens.conf", qui spécifie les directives et paramètres nécessaires pour faire fonctionner le site Drupal. Un sous-dossier, "nginx_logs/", est destiné à stocker les journaux du serveur Nginx, ce qui est utile pour le débogage et la surveillance.

Le dossier "php/" renferme les fichiers de configuration pour le service PHP-FPM. À l'intérieur de ce dossier, on trouve un sous-dossier nommé "fpm/" qui contient des fichiers spécifiques pour PHP-FPM. Parmi eux, le "Dockerfile" définit les instructions nécessaires pour créer une image Docker personnalisée pour PHP-FPM, incluant les extensions et les outils requis. Le fichier "php.ini", également présent, est modifié pour ajuster des paramètres tels que "memory_limit".

Contenu et explication du fichier docker-compose.yml (Cf Annexe 3)

Le fichier "docker-compose.yml" constitue le cœur de la configuration Docker, orchestrant les services Nginx, MariaDB et PHP-FPM nécessaires au fonctionnement du site Drupal. Ce fichier détaille plusieurs aspects essentiels à cette orchestration.

La ligne "version: '3.8'" spécifie la version du format de fichier Docker Compose utilisé. Cela garantit que les fonctionnalités et la syntaxe de la configuration sont compatibles avec cette version.

La section "networks" déclare les réseaux utilisés par les conteneurs pour communiquer entre eux. Dans ce cas, un réseau externe nommé "axens_network" est utilisé pour la communication entre les services, facilitant ainsi l'interconnexion des différents conteneurs.

La section "services" décrit les services (conteneurs) utilisés par le projet, il y en a trois :

nginx : Ce conteneur utilise l'image Nginx "nginx:1.14.2", et le conteneur est nommé "axens_nginx" grâce à "container_name". Ce service est associé au réseau "axens_network" via "networks". Les répertoires de l'hôte sont montés dans le conteneur à l'aide de "volumes", ce qui permet de persister les données tel que la configuration du Virtual Host, les logs Nginx ou le fichier d'environnement. Les ports nécessaires pour accéder au service Nginx sont exposés par "ports". La dépendance du service "php-fpm" est définie par "depends_on", assurant que PHP-FPM est démarré avant Nginx. Enfin, "working_dir" définit le répertoire de travail pour les commandes exécutées dans le conteneur.

mariadb : Ce conteneur utilise l'image MariaDB "mariadb:10.3.39", et le conteneur est nommé "axens_mariadb" grâce à "container_name". Ce conteneur est également associé au réseau "axens_network". Les variables d'environnement nécessaires à la configuration de MariaDB, telles que le mot de passe root, sont définies dans "environment". Les répertoires de l'hôte sont montés dans le conteneur via "volumes" pour persister les données de la base de données ainsi que le répertoire contenant les dumps de base de données. Le répertoire de travail pour les commandes exécutées dans le conteneur est spécifié par "working_dir".

php-fpm : Ce conteneur utilise l'image construite dans le Dockerfile situé dans le dossier "/php/fpm comme spécifié par la ligne "build", à partir de l'image PHP-FPM "php:8.2-fpm". Le conteneur est nommé "axens_php-fpm" grâce à "container_name" et est associé au réseau "axens_network". Les répertoires de l'hôte sont montés dans le conteneur à l'aide de "volumes", permettant de persister le code source du projet web ainsi que le fichier d'environnement et le fichier "php.ini" modifié. Le répertoire de travail pour les commandes exécutées dans le conteneur est spécifié par "working_dir".

La section "volumes" déclare les volumes utilisés pour persister les données indépendamment des conteneurs. Ces volumes garantissent que les informations importantes telles que la base de données ou les logs Nginx ne soient pas perdues lors de la suppression ou de la mise à jour des conteneurs.

Ainsi, le fichier "docker-compose.yml" structure et gère de manière cohérente l'ensemble des services nécessaires au projet Drupal, assurant une configuration et un déploiement efficaces.

Contenu et explication du fichier axens.conf (Cf Annexe 4)

Le fichier "axens.conf" est essentiel pour la configuration du serveur Nginx, qui sert le site Drupal. Ce fichier commence par un bloc "server" qui définit les directives clés pour le serveur web. La directive "listen" spécifie que Nginx écoutera les requêtes sur le port 80, le port HTTP par défaut, ce qui permet au serveur web d'accepter les connexions entrantes. La directive "root" établit le répertoire racine où les fichiers du site web sont stockés, dans ce cas, "/usr/share/nginx/axens/web", permettant à Nginx de localiser les fichiers à servir.

La directive "index" précise les fichiers d'index par défaut, comme index.php et index.html, que Nginx tentera de servir lorsque l'utilisateur accède au répertoire racine du site. La directive "server_name" déclare le nom du serveur virtuel, ici "axens.local.net", ce qui permet à Nginx de gérer plusieurs domaines virtuels sur le même serveur physique.

Un bloc "location /" gère la racine du site. À l'intérieur de ce bloc, la directive "try_files" indique à Nginx d'essayer de servir le fichier demandé. Si ce fichier n'existe pas, Nginx tentera de servir l'URI comme un répertoire et, en dernier recours, passera la requête à index.php avec les arguments de la requête.

Un autre bloc, "location ~ \.php$", gère les requêtes PHP. La directive "try_files" dans ce bloc est similaire, essayant de servir le fichier PHP ou renvoyant une erreur 404 si le fichier n'existe pas. La directive "fastcgi_pass" spécifie l'adresse du service PHP-FPM pour traiter les requêtes PHP, définie ici comme "php-fpm:9000", indiquant que les requêtes PHP doivent être envoyées au service PHP-FPM écoutant sur le port 9000. La directive "fastcgi_param" définit le paramètre "SCRIPT_FILENAME" pour indiquer à PHP-FPM le chemin complet du script PHP à exécuter. Enfin, la directive "include" inclut le fichier "fastcgi_params" qui contient les paramètres par

défaut nécessaires pour FastCGI, simplifiant ainsi la configuration en réutilisant un ensemble standard de paramètres.

En résumé, le fichier "axens.conf" configure Nginx pour qu'il serve efficacement un site Drupal, en gérant les fichiers statiques et en déléguant le traitement des scripts PHP à PHP-FPM, assurant ainsi une performance optimale et une gestion efficace des requêtes.

Contenu et explication du Dockerfile (Cf Annexe 5)

Le Dockerfile utilisé dans ce projet est conçu pour créer une image Docker personnalisée pour le service PHP-FPM, en incluant toutes les extensions et outils nécessaires au bon fonctionnement du site Drupal. La première ligne du Dockerfile, "FROM php:8.2-fpm", spécifie l'image de base à utiliser, ici l'image officielle de PHP 8.2-FPM, fournissant ainsi un environnement PHP-FPM prêt à l'emploi.

Les lignes suivantes commencent par "RUN apt-get update && \", des commandes qui mettent à jour les listes de paquets disponibles et installent les outils et dépendances nécessaires. Par exemple, "apt-get install -y sudo nano iputils-ping mariadb-client" installe sudo (pour l'exécution de commandes avec privilèges élevés), nano (un éditeur de texte), iputils-ping (outil de diagnostic réseau) et le client MariaDB (pour interagir avec la base de données MariaDB).

Une autre commande, "RUN apt-get update && apt-get install -y \", met à jour les listes de paquets et installe des dépendances supplémentaires pour PHP, comme "libfreetype-dev libjpeg62-turbo-dev libpng-dev". Ces bibliothèques de développement sont nécessaires pour la gestion des images par l'extension GD de PHP. La commande "docker-php-ext-configure gd --with-freetype --with-jpeg" configure ensuite l'extension GD de PHP avec le support de FreeType et JPEG. Enfin, la commande "docker-php-ext-install -j$(nproc) gd mysqli pdo_mysql" installe les extensions PHP GD (pour la manipulation des images), MySQLi et PDO MySQL (pour l'interaction avec les bases de données MySQL), utilisant le nombre maximal de processus disponibles pour accélérer la compilation.

En conclusion, ce Dockerfile garantit que l'image Docker personnalisée pour PHP-FPM inclut toutes les extensions et outils nécessaires pour exécuter efficacement le site Drupal. En utilisant ce Dockerfile, les développeurs peuvent s'assurer que leur environnement de développement est cohérent et que toutes les dépendances requises sont présentes.

3.3.3 Déploiement et vérification des conteneurs

Dans cette section, nous allons aborder les étapes nécessaires pour déployer les conteneurs définis dans le fichier "docker-compose.yml", ainsi que les vérifications essentielles pour s'assurer que chaque service fonctionne correctement. Nous détaillerons les commandes à exécuter pour lancer les conteneurs et les techniques pour vérifier leur bon fonctionnement, y compris l'accès au site web via un navigateur.

Commandes pour lancer les conteneurs

Pour lancer les conteneurs définis dans le fichier "docker-compose.yml", nous allons commencer par nous assurer que nous sommes dans le répertoire où se trouve ce fichier. Une fois dans le bon répertoire, nous allons utiliser Docker Compose pour démarrer les services. La commande clé pour lancer les conteneurs est "docker-compose up -d". Cette commande lit les configurations dans le fichier "docker-compose.yml", télécharge les images Docker spécifiées (si elles ne sont pas déjà présentes sur notre machine) et crée les conteneurs décrits dans le fichier. Le flag optionnel "-d" signifie "detached", ce qui permet de lancer les conteneurs en arrière-plan, libérant ainsi notre terminal pour d'autres commandes. Après avoir exécuté cette commande, Docker Compose se chargera de créer et de démarrer les conteneurs pour les services Nginx, MariaDB et PHP-FPM. Pour vérifier que les conteneurs sont en cours d'exécution, nous allons faire la commande "docker ps", qui affiche une liste des conteneurs Docker en cours d'exécution, nous permettant de vérifier que tous les conteneurs nécessaires sont actifs.

Vérification du bon fonctionnement des conteneurs et accès au site via le navigateur

Une fois les conteneurs démarrés, il est important de vérifier que chaque service fonctionne correctement. Nous allons commencer par vérifier les journaux de sortie des conteneurs avec la commande "docker-compose logs". Cette commande affiche les logs des différents services, ce qui peut aider à identifier si un service rencontre des problèmes lors du démarrage. Nous pouvons également vérifier que la base de données MariaDB fonctionne correctement en nous connectant à celle-ci depuis un client MySQL. Pour cela, il faut ouvrir un shell MariaDB à l'intérieur du conteneur avec la commande "docker exec -it axens_mariadb mysql -u root -p" puis, entrer le mot de passe défini dans le fichier ".env" (dans ce cas, "rootdocker"). Nous devrions alors avoir accès à la base de données MariaDB et pouvoir exécuter des requêtes SQL pour vérifier son état.

Pour nous assurer que Nginx et PHP-FPM fonctionnent correctement ensemble, nous pouvons vérifier les logs spécifiques de Nginx et PHP-FPM. Les logs d'accès et d'erreurs de Nginx sont stockés dans le dossier "nginx/nginx_logs". Consultons ces fichiers pour toute erreur ou anomalie. Pour voir les logs de PHP-FPM, nous pouvons exécuter la commande "docker logs axens_php-fpm".

Après avoir vérifié que tous les services sont fonctionnels, ouvrons un navigateur web et accédons à l'URL configurée dans le fichier "axens.conf". Dans cet exemple, l'URL est "http://axens.local.net". Pour que cette URL fonctionne, assurons-nous d'avoir une entrée dans votre fichier "hosts" local qui mappe "axens.local.net" à "127.0.0.1". Ajoutons la ligne "127.0.0.1 axens.local.net" à notre fichier "hosts", situé généralement à "/etc/hosts" sur les systèmes Unix et à "C:\Windows\System32\drivers\etc\hosts" sur Windows. Ensuite, ouvrons notre navigateur et tapons "http://axens.local.net". Nous devrions voir la page d'accueil du site Drupal d’Axens.

En suivant ces étapes, on peut donc lancer les conteneurs Docker et vérifier que le site Drupal fonctionne correctement, accessible via notre navigateur.

3.4 Les processus de déploiement d’un projet Drupal

Dans cette section, nous plongerons dans le processus de déploiement en pré-production et production pour le projet Drupal Axens, en mettant en lumière les différentes étapes clés telles que l'utilisation de Composer, les commandes Drush, et l'automatisation des tâches avec Gulp. Nous explorerons également la connexion via SSH au serveur du site Axens où se trouvent les environnements de pré-production et de production, soulignant l'importance d'un déploiement automatisé pour assurer l’optimisation et la sûreté du processus de déploiement.

Connexion SSH au serveur Axens : Déploiement automatisé

Premièrement, pour déployer les nouvelles fonctionnalités ou corrections de bugs sur les environnements de pré-production et production du site Axens, une connexion SSH sécurisée doit être faite avec le serveur où sont hébergés ces environnements. Ensuite, dans le répertoire correspondant à chaque environnement, que ce soit en pré-production ou en production, les changements sont récupérés via la commande "git pull". Une fois cette étape accomplie, une série de commandes devra être exécutée afin de mettre à jour l'environnement ciblé avec le nouveau code.

Composer : Gestion des dépendances PHP

Composer est un gestionnaire de dépendances pour PHP, largement utilisé dans l'écosystème Drupal pour installer et mettre à jour les bibliothèques et les modules tiers nécessaires au projet. L'étape "composer install" est essentielle lors du déploiement en pré-production et production, car elle garantit que toutes les dépendances requises sont correctement installées et disponibles pour l'application Drupal. Composer analyse le fichier "composer.json" du projet et récupère les packages nécessaires depuis le dépôt Composer, les installant dans le répertoire "vendor".

Drush : Mise à jour de la base de données

Drush est un outil en ligne de commande largement utilisé dans l'écosystème Drupal pour effectuer une variété de tâches, y compris la mise à jour de la base de données. Lors du déploiement en pré-production et production, la commande "drush updb" est utilisée pour appliquer les mises à jour du schéma de base de données, ce qui garantit que la structure de la base de données est synchronisée avec la version actuelle du code Drupal. Cette étape est essentielle pour assurer la cohérence et la compatibilité entre le code et la base de données lors du déploiement de nouvelles fonctionnalités ou corrections de bugs.

Drush : Importation de la configuration

Drupal utilise un système de configuration basé sur des fichiers pour gérer la configuration du site, ce qui permet de versionner et de déployer facilement la configuration entre les environnements. Lors du déploiement en pré-production et production, la commande "drush cim" est utilisée pour importer la configuration du site à partir des fichiers de configuration stockés dans le répertoire "config/sync". Cela garantit que toutes les configurations, y compris les vues, les types de contenu, les blocs, etc., sont synchronisées entre les environnements, assurant ainsi la cohérence et la prévisibilité du site.

Drush : Vidage du cache Drupal

Le cache joue un rôle crucial dans les performances d'un site Drupal, en stockant en mémoire les données fréquemment utilisées pour accélérer le rendu des pages. Lors du déploiement en pré-production et production, la commande "drush cr" est utilisée pour vider et reconstruire le cache Drupal, garantissant que le site utilise les dernières configurations, mises à jour de base de données, et éléments de code. Cela évite les erreurs et les incohérences potentielles causées par l'utilisation de données et de configurations obsolètes dans le cache.

Gulp : Compilation et optimisation des ressources front-end

Gulp est un outil d'automatisation des tâches très utilisé pour le développement web, notamment dans le cadre de la compilation et de l'optimisation des ressources front-end telles que les fichiers CSS et JavaScript. Lors du déploiement en pré-production et production, la tâche "gulp build" est exécutée pour compiler, concaténer, et minimiser les fichiers CSS et JavaScript, optimisant ainsi les performances du site et réduisant les temps de chargement des pages pour les utilisateurs.

En résumé, le déploiement en pré-production et production pour le projet Drupal Axens nécessite plusieurs étapes essentielles : la connexion sécurisée en SSH au serveur d'Axens, l'installation des dépendances avec Composer, la mise à jour de la base de données avec Drush, l'importation de la configuration et le vidage du cache Drupal et la compilation des fichiers CSS et JavaScript.

3.5 Différentes solutions : Optimisation du temps de déploiement

L'optimisation du temps de déploiement en pré-production et production est essentielle pour assurer une mise en ligne rapide et efficace des nouvelles fonctionnalités et des mises à jour sur les sites Drupal. Dans cette section, nous explorerons différentes solutions pour accélérer ce processus, en mettant en lumière les outils et les pratiques qui permettent de réduire les délais entre le développement et la mise en production.

Solution 1 : Déploiement manuel traditionnel

La méthode traditionnelle de déploiement en pré-production et production implique souvent des processus manuels fastidieux. Cela comprend la copie des fichiers de code sur le serveur, l'installation des dépendances, la configuration des paramètres du site, et la mise à jour de la base de données. Bien que cette approche soit courante, elle peut être sujette aux erreurs humaines et prendre beaucoup de temps, en particulier pour les projets complexes.

Solution 2 : Utilisation de services d'hébergement gérés

Les services d'hébergement gérés, tels que Acquia ou Pantheon, offrent des environnements de déploiement Drupal prêts à l'emploi, qui automatisent entièrement le processus de déploiement en pré-production et production. Ces services fournissent des outils conviviaux pour gérer les environnements, effectuer des déploiements, et surveiller les performances du site. Bien que ces solutions puissent être plus coûteuses que l'auto-hébergement, elles offrent une tranquillité d'esprit et une rapidité de déploiement inégalées.

Solution 3 : Utilisation de plates-formes de déploiement automatisé

Certaines plates-formes de déploiement automatisé, telles que Ansible, Puppet ou Chef, offrent des fonctionnalités avancées pour automatiser le déploiement d'applications web sur des serveurs. Ces outils permettent de définir l'infrastructure sous forme de code, ce qui facilite la gestion et la reproductibilité des environnements de déploiement. De plus, ils offrent souvent des fonctionnalités avancées telles que le provisionnement automatique de serveurs, la gestion des configurations, et la surveillance des performances.

Solution 4 : Utilisation de solutions d'orchestration de conteneurs

Les solutions d'orchestration de conteneurs, telles que Kubernetes, offrent une approche moderne pour le déploiement et la gestion des applications conteneurisées en pré-production et production. Kubernetes permet de définir des ensembles de conteneurs, appelés pods, ainsi que des règles de réplication et de mise à l'échelle automatique. Cela permet d'automatiser entièrement le déploiement, la mise à jour et la gestion des applications, réduisant ainsi les délais et les erreurs liées au déploiement manuel.

Solution 5 : Intégration continue et déploiement continu (CI/CD)

L'intégration continue (CI) et le déploiement continu (CD) sont des pratiques de développement logiciel qui visent à automatiser entièrement le processus de construction, de test et de déploiement des applications. En adoptant une approche CI/CD, les développeurs peuvent automatiser la compilation du code, l'exécution des tests, et le déploiement des applications sur les environnements de pré-production et production à chaque nouvelle modification du code source. Cela permet d'accélérer considérablement le cycle de développement et de réduire les risques associés au déploiement manuel.

En résumé, il existe plusieurs solutions pour optimiser le temps de déploiement en pré-production et production des sites Drupal. Ces solutions varient en fonction des besoins spécifiques du projet, de la complexité de l'infrastructure, et des ressources disponibles. Parmi ces solutions, j'ai choisi d'adopter l'intégration continue et le déploiement continu (CI/CD) pour plusieurs raisons. Cette approche permet d'automatiser entièrement le processus de construction, de test et de déploiement, réduisant ainsi les erreurs humaines et accélérant considérablement le cycle de développement. En utilisant des pipelines CI/CD, chaque modification du code est intégrée, testée et déployée de manière cohérente et fiable.

3.6 Mise en place de la solution d’automatisation de déploiement avec CI/CD

Dans cette section, nous décrirons en détail le processus de mise en place d'une solution d'intégration continue (CI) et de déploiement continu (CD) pour le projet Drupal Axens. Nous commencerons par une compréhension approfondie des concepts fondamentaux de l'automatisation de déploiement avec CI/CD, y compris le fonctionnement des pipelines GitLab, des jobs et des runners, ainsi que la communication entre le serveur CI interne et le serveur Axens distant. Ensuite, nous aborderons l'implémentation pratique de cette solution, couvrant l'installation du runner GitLab, la configuration du fichier `.gitlab-ci.yml` et la définition des étapes de build et de déploiement. Enfin, nous expliquerons comment vérifier le bon

fonctionnement du pipeline CI/CD et résoudre les éventuels problèmes pour assurer une livraison de code fiable et continue.

3.6.1 Comprendre l’automatisation du déploiement avec CI/CD

Fonctionnement du runner du serveur CI interne

Le runner du serveur CI interne est responsable de l'exécution des pipelines GitLab, qui sont des séries de tâches automatisées (ou jobs) qui définissent les étapes du processus CI/CD. Le runner récupère les instructions du fichier de configuration ".gitlab-ci.yml" du référentiel GitLab, puis exécute les jobs définis dans ce fichier. Il peut s'agir de tâches telles que la compilation du code, l'exécution des tests, la construction des artefacts, ou le déploiement sur des environnements de pré-production ou de production.

Communication avec le serveur Axens distant

Une fois les tests réussis sur le serveur CI interne, la communication avec le serveur Axens distant est établie pour déployer les modifications sur les environnements de pré-production et de production. Cette communication se fait généralement via SSH, un protocole de communication sécurisé, qui permet d'établir une connexion sécurisée entre les serveurs. Des scripts et des commandes automatisés sont utilisés pour transférer les fichiers de code, les dépendances, et les configurations vers les environnements cibles, assurant ainsi un déploiement fiable et sans erreur.

Pipelines GitLab : Automatisation du processus CI/CD

Les pipelines GitLab sont des ensembles de jobs organisés en séquences logiques, qui définissent les étapes du processus CI/CD. Chaque pipeline peut contenir plusieurs stages, et chaque stage peut contenir plusieurs jobs. Les jobs sont des tâches automatisées qui peuvent être exécutées en parallèle ou en séquence, en fonction des besoins du processus de développement. Ils peuvent inclure des

étapes telles que la compilation du code, les tests unitaires, les tests fonctionnels, et le déploiement sur les environnements de pré-production et de production.

En résumé, l'automatisation des tests et du déploiement avec CI/CD pour le projet Drupal Axens implique la mise en place de pipelines GitLab, l'exécution de jobs sur le runner du serveur CI interne, et la communication avec le serveur Axens distant pour le déploiement sur les environnements de pré-production et de production. Ces processus automatisés garantissent la qualité, la fiabilité et la rapidité du cycle de développement, en permettant une intégration continue des modifications de code et un déploiement rapide et sûr des nouvelles fonctionnalités sur le site Axens.

3.6.2 Implémentation d’une CI/CD pour le projet Drupal Axens

Dans cette section, nous allons détailler la mise en place d'une intégration continue (CI) et d'un déploiement continu (CD) pour le projet Drupal Axens en utilisant GitLab CI/CD. Nous expliquerons chaque étape, depuis la mise en place d'un runner GitLab sur le serveur Studio.gd, jusqu'à la création et la configuration du fichier ".gitlab-ci.yml".

Pour commencer, il est nécessaire d'installer GitLab Runner sur le serveur de Studio.gd. Ce processus débute par le téléchargement et l'installation de GitLab Runner, suivie de son enregistrement auprès de GitLab. Lors de l'enregistrement, des informations comme l'URL du serveur GitLab, le jeton d'enregistrement, une description du runner, des étiquettes, et le type d'exécuteur (dans ce cas, "shell") sont fournies. Une fois enregistré, le runner est démarré pour qu'il soit prêt à exécuter des tâches.

Pour sécuriser les connexions SSH, une variable d'environnement "SSH_PRIVATE_KEY" est créée dans les paramètres du projet GitLab. Cette variable contient la clé SSH privée, ce qui permet au runner de se connecter au serveur distant de manière sécurisée. En configurant cette variable comme "Environment Variable" et en activant l'option "Protected", l'accès à cette clé est restreint aux pipelines des branches protégées uniquement.

Ensuite, nous créons le fichier ".gitlab-ci.yml" à la racine du projet Axens (Cf Annexe 6). Ce fichier définit les étapes et les scripts pour le pipeline CI/CD. La première section, "workflow", spécifie les règles de déclenchement des pipelines, établissant que les pipelines ne s'exécutent que pour les branches "staging" et "main". Le fichier utilise l'image Docker "debian:bullseye" comme base pour les jobs.

Le pipeline est divisé en deux étapes principales : "build" et "deploy". La première étape, "build-theme", s'exécute pour les branches "staging" et "main". Pendant cette étape, plusieurs commandes sont exécutées pour installer Node.js et Yarn, naviguer dans le répertoire du thème, installer les modules Node nécessaires, installer Gulp globalement, et construire les fichiers CSS et JavaScript. Les fichiers résultants sont conservés en tant qu'artefacts, avec une durée de conservation d'une semaine. Les artefacts incluent les fichiers "style.css", "script.js" et "mobile.js".

Avant chaque job, un script de préparation "before_script" est exécuté pour configurer la connexion SSH. Ce script installe "rsync" et "ssh-agent" si nécessaire, démarre "ssh-agent", ajoute la clé SSH privée, stockée dans la variable d'environnement "SSH_PRIVATE_KEY" à l'agent, crée le répertoire SSH avec les permissions appropriées, et désactive la vérification stricte des clés hôtes pour éviter les interruptions lors des connexions SSH.

La deuxième étape, "deploy-OVH-staging", déploie le site sur les environnements de pré-production et de production en fonction de la branche. Pour la branche "main", le déploiement doit être validé manuellement, alors que pour la branche "staging", il est automatique. Le job de déploiement dépend de la construction du thème et utilise SSH pour se connecter au serveur distant, naviguer dans le répertoire approprié, récupérer les dernières modifications via "git pull", installer les dépendances avec Composer, et exécuter plusieurs commandes Drush pour vider le cache, mettre à jour la base de données et importer la configuration.

Ainsi, en suivant ces étapes, un pipeline CI/CD complet et automatisé est mis en place pour le projet Drupal Axens, permettant une intégration et un déploiement continus, fiables et sécurisés des nouvelles fonctionnalités et mises à jour du site.

3.6.3 Vérification et fonctionnement de la CI/CD

Une fois que nous avons configuré notre pipeline CI/CD dans GitLab, il est crucial de vérifier que tout fonctionne correctement. Cette section explique comment déclencher le pipeline, vérifier son bon fonctionnement et résoudre les problèmes éventuels.

Pour déclencher le pipeline CI/CD, nous commençons par effectuer un commit et pousser les modifications vers l'une des branches spécifiées dans les règles du pipeline ("staging" ou "main"). Nous apportons des modifications à un fichier dans notre projet, par exemple en ajoutant une nouvelle fonctionnalité ou en corrigeant un bug dans le code. Ensuite, nous enregistrons nos modifications avec un message de commit descriptif et poussons les modifications vers la branche appropriée, "staging" ou "main". Ces actions déclenchent automatiquement le pipeline CI/CD défini dans le fichier ".gitlab-ci.yml".

Une fois le pipeline déclenché, nous pouvons suivre son avancement dans l'interface de GitLab. Nous allons dans la section "CI/CD" de notre projet et cliquons sur "Pipelines" (Cf Annexe 7). Nous voyons alors une liste des pipelines exécutés avec leur statut actuel. En cliquant sur le pipeline en cours, nous pouvons voir les détails de chaque job et vérifier leur progression.

Nous commençons par vérifier les logs des jobs pour nous assurer que chaque étape s'exécute correctement. Pour l'étape "build-theme", nous nous assurons que Node.js et Yarn sont installés sans erreurs, que les modules Node sont correctement installés et que les fichiers CSS et JavaScript sont construits et enregistrés comme artefacts. Les artefacts peuvent être consultés et téléchargés depuis l'interface de GitLab pour vérification (Cf Annexe 8).

Pour l'étape "deploy-OVH-staging", nous vérifions que la connexion SSH est établie correctement avec le serveur distant, que les fichiers de code sont correctement récupérés avec "git pull", que les dépendances sont installées sans problèmes via Composer, et que les commandes Drush sont exécutées sans erreurs pour vider le cache, mettre à jour la base de données et importer la configuration. Les logs de ces actions nous fournissent des informations détaillées sur chaque commande exécutée.

Si nous rencontrons des erreurs ou des échecs dans l'une des étapes du pipeline, nous examinons attentivement les messages d'erreur dans les logs pour identifier la cause du problème. Les erreurs courantes peuvent inclure des problèmes de permissions, des échecs de téléchargement de dépendances, ou des erreurs de connexion SSH. Une fois les problèmes identifiés, nous apportons les corrections nécessaires et redéclenchons le pipeline pour vérifier que tout fonctionne comme prévu.

En suivant ces étapes, nous pouvons nous assurer que notre pipeline CI/CD fonctionne correctement et que les déploiements sur les environnements de pré-production et de production se déroulent sans accroc. Cette vérification régulière nous permet de maintenir la qualité et la fiabilité de notre processus de déploiement automatisé pour le projet Drupal Axens.

Dans cette section, nous avons approfondi l'optimisation des processus d'installation et de déploiement pour les projets Drupal. Nous avons débuté par une analyse des composants clés de l'environnement de travail Drupal, tels que Nginx, PHP-FPM et MariaDB. Ensuite, nous avons examiné les étapes nécessaires pour déployer un projet Drupal, mettant en lumière l'utilisation des commandes Composer, Drush et Gulp pour automatiser les processus.

Nous avons ensuite évalué différentes solutions pour optimiser ces deux aspects critiques. Pour l'installation de projets Drupal, nous avons choisi Docker, tandis que pour le déploiement, nous avons opté pour l'intégration CI/CD avec GitLab.

Nous avons détaillé la mise en œuvre de ces solutions, expliquant leur fonctionnement et leur configuration spécifique pour le projet Axens. Enfin, nous avons testé et validé leur fonctionnement pour assurer leur intégrité et leur performance.

Partie 4 : Bilan et perspectives

Bilan du projet

Au cours de ce projet, j'ai rencontré et surmonté plusieurs difficultés. L'une des principales a été de devoir me former de manière autonome aux technologies Docker et à la CI/CD de Gitlab, car à l'époque où je devais développer cette solution en entreprise, nous n'avions pas encore abordé ces sujets à l'école IPSSI. Une autre difficulté a été de demander au client du projet Axens de whitelister le serveur de Studio.gd afin que le runner de la CI situé sur notre serveur puisse effectuer les déploiements nécessaires.

J'ai également dû faire face à une difficulté technique spécifique, le site ne fonctionnait pas correctement via Docker car le “memory_limit” de l'image PHP-FPM de base n'était pas suffisamment élevé. Pour résoudre ce problème, j'ai dû trouver une solution en envoyant un nouveau fichier “php.ini” via les volumes du container PHP-FPM avec le bon “memory_limit”, pour qu’il écrase l’ancien “php.ini” et que le site puisse fonctionner correctement.

Axes d'amélioration futures

Pour améliorer notre solution, je vois plusieurs axes de développement futurs. Tout d'abord, l'adoption généralisée sur tous les projets de Docker pour homogénéiser les environnements de développement ce qui pourrait grandement réduire les incohérences et les incompatibilités. En encapsulant toutes les dépendances et configurations nécessaires dans des conteneurs, nous pourrions garantir un environnement de développement cohérent pour tous les membres de l'équipe sur tous les projets de l’entreprise et non uniquement sur le projet Axens.

Ensuite, la mise en place des déploiements via des pipelines CI/CD sur tous les projets de l’entreprise pourrait aussi réduire le temps perdu et les erreurs associées aux déploiements manuels. Il faudrait aussi intégrer des tests automatisés lors des déploiements afin de garantir des déploiements plus fiables et limiter les risques.

Apprentissages et perspectives personnelles

Sur le plan personnel, ce projet m'a permis d'acquérir une expertise approfondie dans le développement et le déploiement de projets Drupal, ainsi que dans l'utilisation de Docker et des pipelines CI/CD. Ces technologies sont particulièrement prisées dans les entreprises de développement web.

Pour l'avenir, je prévois de terminer mes études et d'obtenir mon BAC+3, avec l'objectif de me faire embaucher en CDI chez Studio.gd. Je souhaite me spécialiser davantage dans les technologies de conteneurisation et d'automatisation des déploiements, afin de développer ces compétences au sein de l'entreprise. En parallèle, je compte continuer à approfondir mes connaissances sur WordPress, Drupal et Symfony, afin de pouvoir gérer des projets plus complexes.

Conclusion

Nous avons exploré en détail les processus d'installation et de déploiement des projets web au sein de l'entreprise Studio.gd, mettant en lumière leur importance cruciale dans le cycle de vie du développement d'un projet web. Les défis tels que la complexité des environnements, la gestion des dépendances, le temps de configuration, et la gestion des versions et des mises à jour sont des obstacles courants. Ces défis soulignent la nécessité d'une approche proactive et innovante pour optimiser ces processus.

Puis Nous avons exploré la structure et les activités de Studio.gd, une entreprise spécialisée dans le développement web et mobile. Avec une équipe de 11 employés, l'entreprise combine expertise et agilité pour répondre aux besoins variés de ses clients. En tant que membre de cette équipe, j'ai travaillé sur une large gamme de projets, de la maintenance évolutive à la création de nouveaux sites web, développant des compétences approfondies en développement Drupal et WordPress et participant à des initiatives DevOps.

Ensuite, nous avons exploré les défis et solutions liés à l'installation et au déploiement de projets Drupal, ainsi que la mise en place de Docker pour le projet Drupal Axens. En détaillant l'environnement de développement local, nous avons mis en lumière l'importance de Nginx, PHP et MySQL/MariaDB pour créer un environnement fonctionnel. Nous avons aussi examiné des méthodes pour optimiser le temps d'installation d'un projet Drupal en local, avec Docker se distinguant par sa capacité à offrir une configuration rapide et une reproductibilité garantie. La mise en place de Docker pour le projet Drupal Axens a permis de créer des environnements de développement reproductibles et cohérents, réduisant les risques de disparités entre les environnements de développement et de production.

Nous avons également détaillé le processus de déploiement en pré-production et production, et exploré diverses solutions pour optimiser ce processus, en soulignant l'importance de l'intégration continue et du déploiement continu (CI/CD) pour automatiser et sécuriser le cycle de développement. La mise en place de GitLab

CI/CD pour le projet Drupal Axens a permis une intégration et un déploiement continus, sécurisés et fiables, améliorant l'efficacité et la fiabilité des processus de déploiement.

Ce projet a été riche en défis et en apprentissages. La nécessité de se former de manière autonome à Docker et à la CI/CD de GitLab a été particulièrement marquante. Les difficultés techniques ont été surmontées grâce à des solutions adaptées.

Pour améliorer la solution, plusieurs axes de développement futurs ont été identifiés, notamment l'adoption généralisée de Docker et la mise en place des déploiements CI/CD sur tous les projets. Sur le plan personnel, ce projet m'a permis d'acquérir une expertise approfondie dans le développement et le déploiement de projets Drupal, ainsi que dans l'utilisation de Docker et des pipelines CI/CD. Pour l'avenir, je prévois de terminer mes études, obtenir mon BAC+3, et me faire embaucher en CDI chez Studio.gd, tout en continuant à approfondir mes connaissances sur WordPress, Drupal et Symfony.

En conclusion, ce projet a non seulement permis d'améliorer significativement le processus de développement et de déploiement pour le projet Axens, mais a également ouvert des perspectives de développement personnel et professionnel en me permettant d’obtenir des compétences nécessaires pour évoluer dans le domaine du développement web.

Bibliographie / Webographie

Docker

1. https://docs.docker.com/guides/
3. https://www.youtube.com/watch?v=fqMOX6JJhGo
4. https://docker.com/blog/
5. https://www.udemy.com/course/docker-mastery/

CI/CD GitLab

1. https://docs.gitlab.com/ee/ci/
2. https://www.youtube.com/watch?v=qP8kir2GUgo
3. https://cours.brosseau.ovh/tp/ci/gitlab/runner.html

Nginx

1. https://nginx.org/en/docs/
2. https://www.nginx.com/blog/
3. https://www.linode.com/docs/web-servers/nginx/
4. https://makina-corpus.com/drupal/bien-debuter-avec-nginx

PHP

1. https://www.php.net/docs.php
2. https://www.w3schools.com/php/
3. https://serverhub.com/kb/complete-guide-on-how-to-install-php-extensions-on-ubuntu/

MariaDB

1. https://mariadb.com/kb/en/documentation/

Drush

1. https://www.drush.org/latest/
2. https://drupal.stackexchange.com/questions/254407/order-of-drush-commands-for-automated-deployment

Gulp

1. https://gulpjs.com/docs/en/getting-started/quick-start

Composer

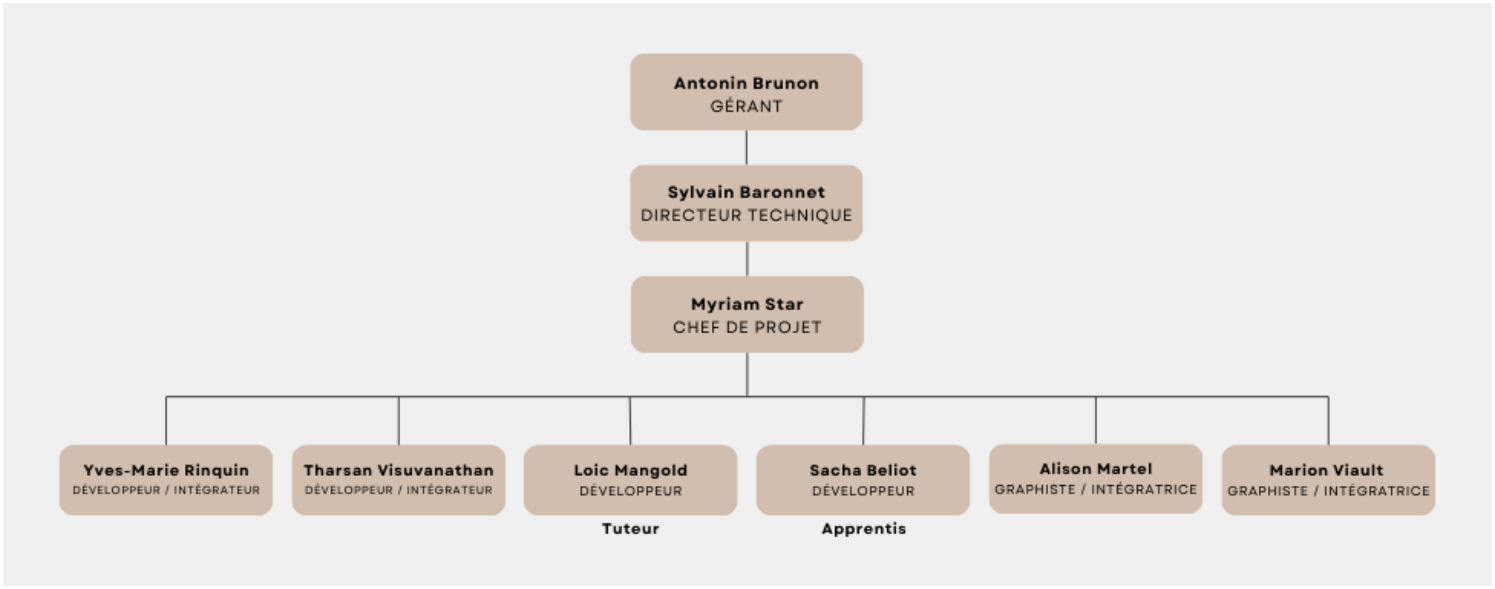
1. https://getcomposer.org/doc/
2. https://www.hostinger.fr/tutoriels/comment-installer-et-utiliser-composer

SSH

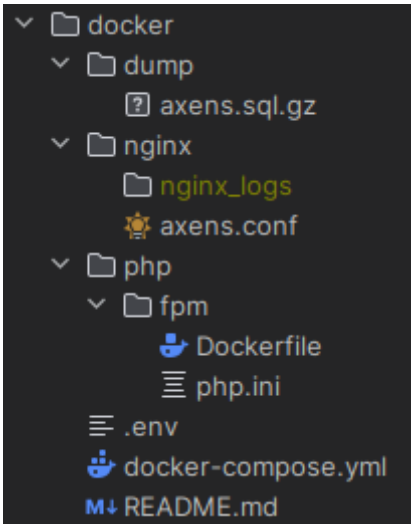
1. https://www.ssh.com/academy/ssh
2. https://www.infomaniak.com/fr/support/faq/1941/se-connecter-en-ssh-et-utiliser-des-commandes-en-ligne

Annexes

Annexe 1 : Organigramme de l'entreprise Studio.gd



Annexe 2 : Arborescence du Docker



Annexe 3 : Fichier docker-compose.yml

```yaml
version: '3.8'

networks:
  axens_network:
    external: true

services:
  nginx:
    image: nginx:1.14.2
    container_name: axens_nginx
    networks:
      - axens_network
    volumes:
      - ../:/usr/share/nginx/axens
      - ./nginx/axens.conf:/etc/nginx/conf.d/axens.conf
      - ./nginx/nginx_logs:/var/log/nginx
      - ./.env:/usr/share/nginx/axens/.env
    ports:
      - "80:80"
    depends_on:
      - php-fpm
    working_dir: /var/log/nginx

  mariadb:
    image: mariadb:10.3.39
    container_name: axens_mariadb
    networks:
      - axens_network
    environment:
      MYSQL_ROOT_PASSWORD: rootdocker
    volumes:
      - ./dump:/usr/share/nginx/axens/dump
      - mariadb_data:/var/lib/mysql
    working_dir: /usr/share/nginx/axens/dump


  php-fpm:
    build: ./php/fpm
    image: php:8.2-fpm
    container_name: axens_php-fpm
    networks:
      - axens_network
    volumes:
      - ../:/usr/share/nginx/axens
      - ./php/fpm/php.ini:/usr/local/etc/php/php.ini
      - ./.env:/usr/share/nginx/axens/.env
    working_dir: /usr/share/nginx/axens

volumes:
  mariadb_data:
  nginx_logs:
```

Annexe 4 : Fichier axens.conf

```
server {
        listen 80;
        listen [::]:80;
        root /usr/share/nginx/axens/web;
        index index.php index.html;
        server_name axens.local.net;
        location / {
                try_files $uri $uri/ /index.php?$args;
        }
        location ~ \.php$ {
                try_files $uri =404;
                fastcgi_pass php-fpm:9000;
                fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
                include fastcgi_params;
        }
}
```

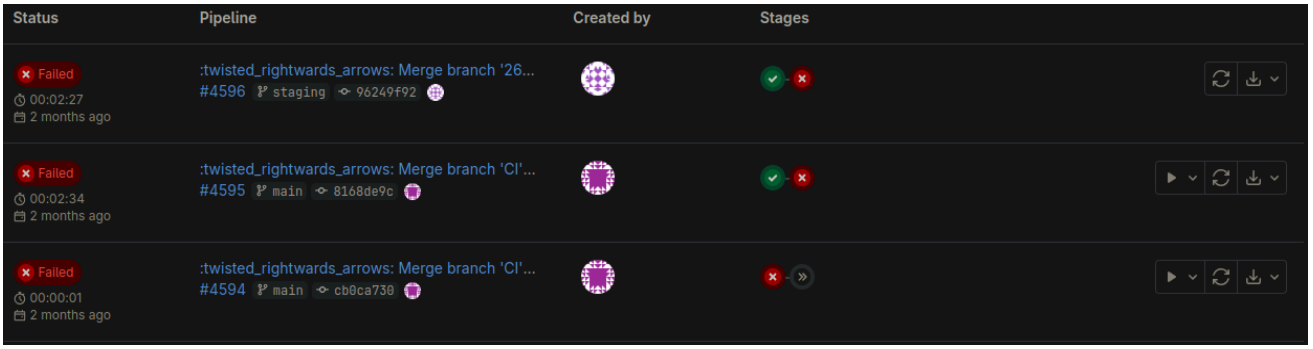
Annexe 5 : Fichier Dockerfile

```
FROM php:8.2-fpm

RUN apt-get update && \
    apt-get install -y sudo \
                       nano \
                       iputils-ping \
                       mariadb-client

RUN apt-get update && apt-get install -y \
    libfreetype-dev \
    libjpeg62-turbo-dev \
    libpng-dev \
  && docker-php-ext-configure gd --with-freetype --with-jpeg \
  && docker-php-ext-install -j$(nproc) gd mysqli pdo_mysql
```

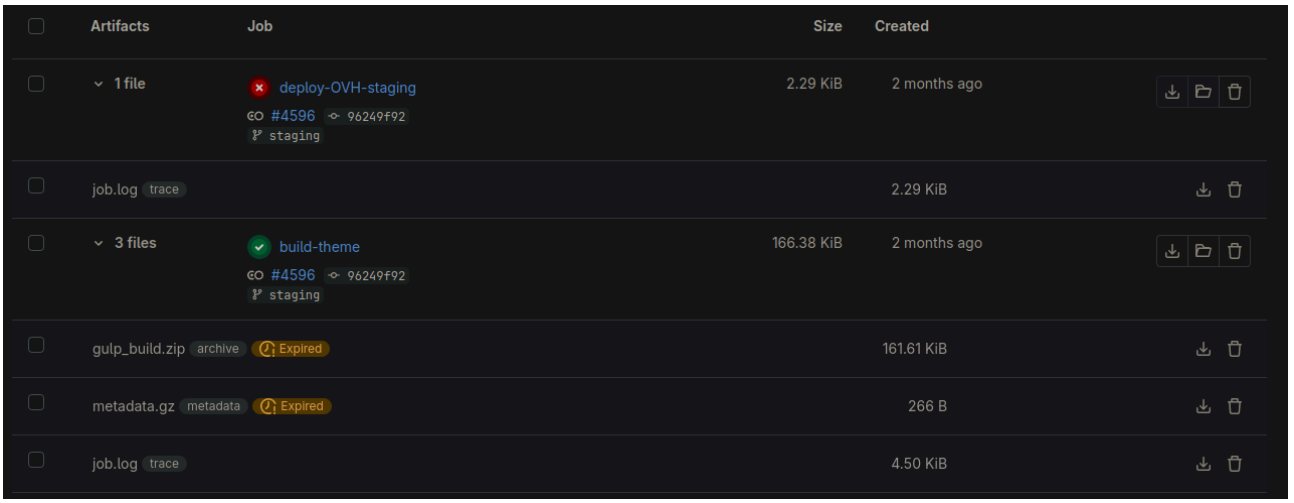
Annexe 6 : Fichier .gitlab-ci.yml

```yaml
workflow:
  rules:
    - if: $CI_COMMIT_BRANCH == "staging" || $CI_COMMIT_BRANCH == "main"

# Docker image based on Debian
image: debian:bullseye

# Stage's list
stages:
  - build
  - deploy

# Build job
build-theme:
  rules:
    - if: $CI_COMMIT_BRANCH == "staging" || $CI_COMMIT_BRANCH == "main"
  stage: build
  script:
    # Install NODEJS & YARN
    - apt-get update && apt-get install -y nodejs yarn
    - cd web/themes/axens
    # Install NODE_MODULES.
    - yarn install
    # Install GULP
    - yarn global add gulp
    # Build CSS and JS
    - gulp build
  artifacts:
    name: 'gulp build'
    expire_in: 1 week
    paths:
      - web/themes/axens/assets/style.css
      - web/themes/axens/assets/script.js
      - web/themes/axens/assets/mobile.js

# Script to allow SSH connection
before_script:
  # Install rsync && ssh-agent if not already installed, it is required by Docker.
  - 'command -v ssh-agent >/dev/null || ( apt-get update && apt-get install -y rsync openssh-client)'
  # Run ssh-agent (inside the build environment)
  - eval $(ssh-agent -s)
  # Add the SSH key stored in SSH_PRIVATE_KEY variable to the agent store
  - echo "${SSH_PRIVATE_KEY}" | tr -d '\r' | ssh-add -
  # Create the SSH directory and give it temporary the right permissions
  - mkdir -p ~/.ssh
  # Disable host key checking (susceptible to man-in-the-middle attacks).
  - echo -e "Host *\n\tStrictHostKeyChecking no\n\n" > ~/.ssh/config

# Deploy job
deploy-OVH-staging:
  rules:
    - if: $CI_COMMIT_BRANCH == "main"
      when: manual
      variables:
        FOLDER: "www-axens"
        REMOTE_BRANCH: "main"
    - if: $CI_COMMIT_BRANCH == "staging"
      variables:
        FOLDER: "www-axens-rct"
        REMOTE_BRANCH: "staging"
  stage: deploy
  dependencies:
    - build-theme
  script:
    - ssh -A octaveoctave@91.188.64.20 "cd /var/www/$FOLDER/axens/ && git pull origin $REMOTE_BRANCH && ls -la"
    - ssh -A octaveoctave@91.188.64.20 "cd /var/www/$FOLDER/axens/ &&
      composer install --prefer-dist --no-interaction --optimize-autoloader"
    - ssh -A octaveoctave@91.188.64.20 "cd /var/www/$FOLDER/axens/ && ./vendor/bin/drush cr &&
      ./vendor/bin/drush updb -y && ./vendor/bin/drush cim -y && ./vendor/bin/drush cr"
```

Annexe 7 : État des pipelines CI/CD dans GitLab



Annexe 8 : Artifacts et état des jobs dans le pipeline CI/CD de GitLab



Glossaire

- **Composer :** Gestionnaire de dépendances pour PHP, utilisé pour installer et mettre à jour les bibliothèques et modules nécessaires dans un projet Drupal.

- **Composer.json :** Fichier de configuration de Composer, listant les dépendances et les métadonnées du projet.

- **Drush :** Outil en ligne de commande pour gérer les tâches administratives de Drupal, telles que la mise à jour de la base de données et l'importation de configurations.

- **Commandes Drush (updb, cim, cr) :**
  - drush updb : Met à jour le schéma de base de données.
  - drush cim : Importe la configuration du site à partir des fichiers de configuration.
  - drush cr : Vide et reconstruit le cache de Drupal.

- **Cache :** Mécanisme de stockage temporaire pour accélérer l'accès aux données fréquemment utilisées.

- **Gulp :** Outil d'automatisation des tâches pour le développement web, utilisé pour compiler et optimiser les fichiers front-end comme CSS et JavaScript.

- **SSH (Secure Shell) :** Protocole de communication sécurisé permettant de se connecter à des serveurs distants et d'exécuter des commandes.

- **Git :** Système de contrôle de version distribué permettant de suivre les modifications dans le code source et de collaborer avec d'autres développeurs.

- **GitLab :** Plate-forme de gestion de dépôts Git, offrant des outils pour l'intégration et le déploiement continus (CI/CD).

- **CI/CD (Continuous Integration/Continuous Deployment) :** Pratiques de développement visant à automatiser les processus de construction, de test et de déploiement des applications.

- **.gitlab-ci.yml :** Fichier de configuration pour définir les pipelines CI/CD dans GitLab.

- **Pipeline :** Ensemble de tâches automatisées définies pour les processus CI/CD, telles que la compilation du code et le déploiement sur des serveurs.

- **Runner :** Service exécutant les tâches définies dans les pipelines CI/CD, comme les tests ou les déploiements.

- **Artefacts :** Fichiers générés par un processus de build, tels que les fichiers compilés ou les résultats de tests, stockés pour une utilisation ultérieure.

- **Environnements de pré-production et production :** Configurations de serveurs utilisées pour tester (pré-production) et déployer (production) des applications avant et après leur mise en ligne.

- **Docker :** Plate-forme permettant de créer, déployer et exécuter des applications dans des conteneurs, offrant une isolation des environnements.

- **Conteneurisation :** Méthode de virtualisation permettant d'exécuter des applications et leurs dépendances dans des environnements isolés et légers appelés conteneurs.

- **Image Docker :** Modèle immuable contenant tout ce qu'il faut pour exécuter une application, utilisé pour créer des conteneurs Docker.

- **Conteneurs Docker :** Des unités standard de logiciel qui empaquettent une application et ses dépendances, permettant une exécution rapide dans différents environnements.

- **Volumes Docker :** Des mécanismes pour stocker les données générées et utilisées par les conteneurs de manière persistante.

- **Docker Compose :** Un outil permettant de définir et de gérer des applications multi-conteneurs via des fichiers de configuration YAML.

- **Dockerfile :** Un fichier contenant une série d'instructions utilisées pour construire une image Docker.

- **Node.js :** Environnement d'exécution JavaScript côté serveur, souvent utilisé pour le développement web.

- **Yarn :** Gestionnaire de packages JavaScript, utilisé pour installer et gérer les dépendances des projets Node.js.

- **Nginx :** Serveur web et proxy inverse, utilisé pour servir les applications web et gérer les connexions HTTP.

- **PHP :** Un langage de script côté serveur utilisé pour le développement web, alimentant le cœur de Drupal pour traiter les requêtes et générer des pages web dynamiques.

- **PHP-FPM (FastCGI Process Manager) :** Gestionnaire de processus pour PHP, améliorant les performances des applications web en gérant les connexions de manière efficace.

- **MariaDB :** Système de gestion de base de données relationnelle, souvent utilisé comme alternative à MySQL.

- **Ansible, Puppet, Chef :** Outils d'automatisation de la gestion des configurations et du déploiement des applications sur des serveurs.

- **Kubernetes :** Plate-forme d'orchestration de conteneurs, permettant de gérer le déploiement, la mise à l'échelle et les opérations des applications conteneurisées.

- **Drupal :** Un système de gestion de contenu (CMS) open-source utilisé pour créer et gérer des sites web.

- **Environnement de développement :** Un cadre comprenant les outils et les logiciels nécessaires pour développer et tester des applications localement.

- **MySQL/MariaDB :** Systèmes de gestion de bases de données relationnelles utilisés pour stocker et gérer les données d'un site Drupal.

- **Virtual Hosts :** Une technique permettant de configurer plusieurs sites web sur une seule machine physique, chacun ayant son propre domaine et répertoire racine.

- **Machine virtuelle (VM) :** Une simulation logicielle d'un ordinateur physique, permettant d'exécuter plusieurs systèmes d'exploitation sur une seule machine physique.

- **Cloud :** Un ensemble de services informatiques disponibles à la demande via internet, souvent utilisés pour le stockage et l'exécution d'applications.

- **Noyau de l'hôte :** La partie centrale du système d'exploitation qui gère les opérations fondamentales et les interactions avec le matériel.

- **YAML :** Un format de fichier lisible par l'humain, utilisé pour configurer les services dans Docker Compose.

- **HTTP :** Hypertext Transfer Protocol, le protocole utilisé pour transférer des documents web sur le réseau internet.

- **Extension PHP :** Des modules ajoutant des fonctionnalités supplémentaires à PHP, nécessaires pour exécuter certaines applications.

- **Sauvegarde compressée (dump) :** Une copie des données de la base de données compressée pour économiser de l'espace de stockage.

- **Port :** Un point d'accès numérique utilisé pour la communication entre les systèmes informatiques via les réseaux.

- **FastCGI :** Une interface qui permet d'améliorer la performance des scripts PHP en les exécutant de manière persistante et efficace.

- **Whitelister :** Le fait d’inscrire un ensemble d'entités (personnes, comptes, machines…) auxquels on attribue un niveau de liberté ou de confiance maximum dans un système particulier.