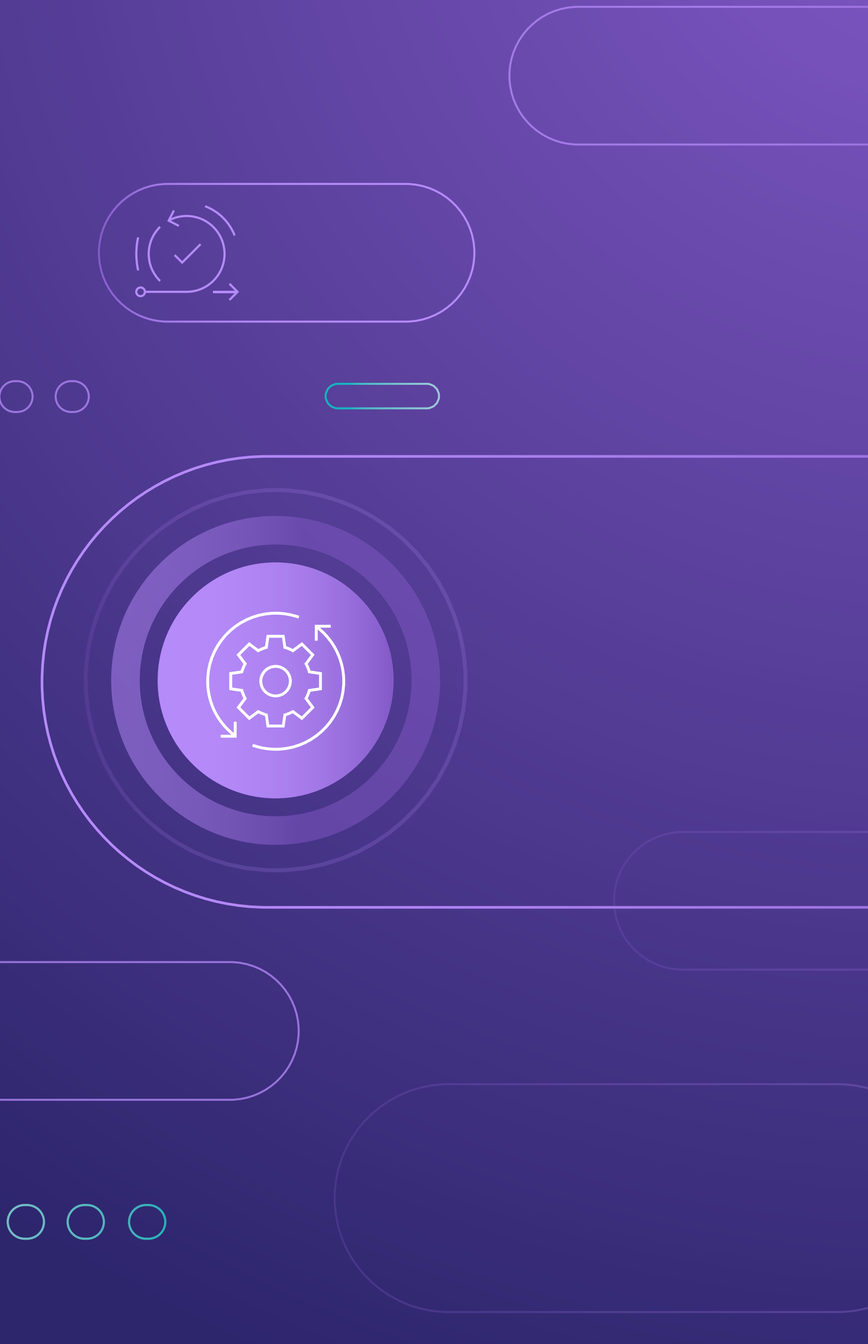
GitLab

Guide complet sur l'approche CI/CD

Des principes fondamentaux à
l'intégration des tests de sécurité et de l'IA



Sommaire

Introduction

L'approche CI/CD (intégration continue/livraison continue) a révolutionné la collaboration entre les équipes logicielles pour convertir le code en applications. L'époque des problèmes d'intégration du code et des processus manuels à répétition est révolue. Avec l'approche CI/CD, le développement logiciel est devenu un processus à la fois agile, fiable et automatisé.

Cette méthode repose sur la mise en place d'un pipeline fluide et automatisé, permettant de transférer le code de l'environnement de développement à l'environnement de production, tout en intégrant les retours et suggestions de modification en temps réel. L'intégration continue (CI) aide les équipes à détecter rapidement les problèmes avant qu'ils ne deviennent coûteux. Les modifications de code sont fréquemment fusionnées dans un dépôt partagé, puis testées automatiquement et validées. La livraison continue (CD) vient compléter cette démarche. Elle vise à automatiser les déploiements, rendant les sorties de nouvelles versions prévisibles et harmonieuses.

Grâce à un pipeline CI/CD robuste, les équipes peuvent compiler, tester et déployer leurs logiciels sans dépendre de processus manuels ou de chaînes d'outils complexes. En intégrant l'IA, il est possible d'optimiser encore davantage ce processus. L'IA peut en effet générer automatiquement des pipelines CI/CD, garantissant ainsi la cohérence des contrôles de qualité, de conformité et de sécurité.

Résultat : des logiciels de meilleure qualité, des cycles de livraison plus courts et le déploiement plus fréquent des nouvelles versions pour répondre rapidement aux besoins des utilisateurs et aux demandes du marché.

Explorez dans ce guide tous les aspects des pipelines CI/CD modernes, des principes de base aux bonnes pratiques, en passant par les stratégies avancées. À l'issue de cette lecture, vous saurez comment faire évoluer votre environnement DevSecOps afin de développer et de livrer des logiciels de manière agile, automatisée et efficace.

Grâce à l'intégration continue, les équipes peuvent identifier et corriger les erreurs, ainsi que les failles de sécurité, plus facilement et beaucoup plus tôt dans le processus de développement.

Principes fondamentaux de l'approche CI/CD

Qu'est-ce que l'intégration continue ?

L'intégration continue (CI) est une pratique qui consiste à intégrer régulièrement les modifications de code dans la branche principale d'un dépôt partagé. Elle s'effectue dès que possible, et fréquemment. Après chaque validation ou merge request, les modifications sont automatiquement testées, puis une compilation est déclenchée sans intervention manuelle. Grâce à l'intégration continue, les équipes peuvent identifier et corriger les erreurs, ainsi que les failles de sécurité, plus facilement et beaucoup plus tôt dans le processus de développement.

Qu'est-ce que la livraison continue ?

La livraison continue (CD), parfois appelée déploiement continu, automatise le processus de mise en production des applications. Les développeurs ont ainsi plus de temps à consacrer au suivi des déploiements en cours pour en garantir la réussite. Avec cette méthode, les équipes DevSecOps définissent à l'avance les critères de mise à disposition du code. Lorsque ces critères sont remplis et validés, le code est déployé dans l'environnement de production. Cette automatisation permet aux équipes de gagner en flexibilité et de mettre plus rapidement de nouvelles fonctionnalités à la disposition des utilisateurs, tout en sachant que leurs déploiements ont réussi tous les tests de sécurité.

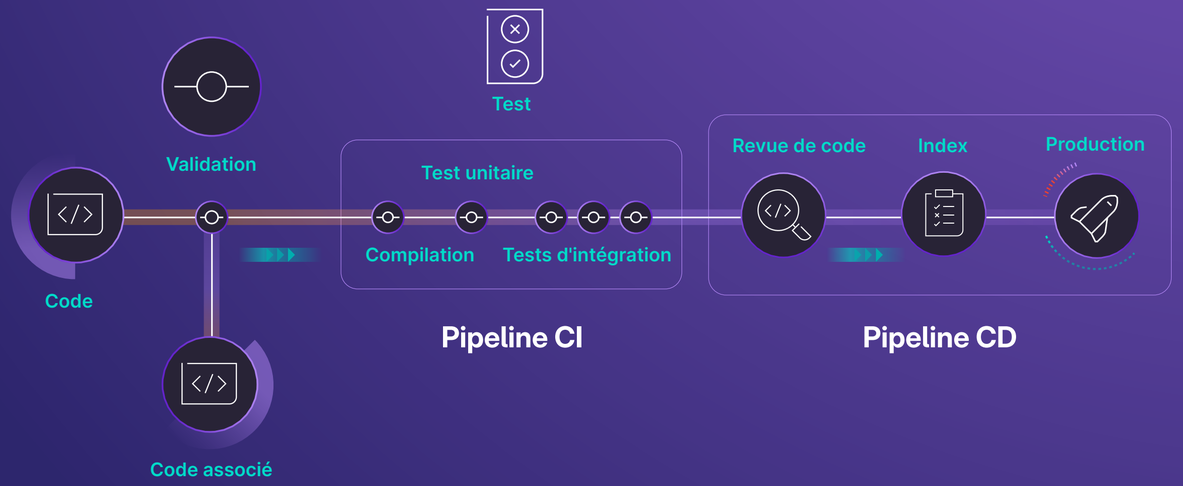
Qu'est-ce qu'un pipeline CI/CD ?

Un pipeline CI/CD est une série structurée d'étapes (compilation, test et déploiement) qui automatise et rationalise le processus de livraison de logiciels. Chaque étape agit comme un mur qualité et permet de s'assurer que seul le code validé passe à l'étape suivante. Les premières étapes gèrent les vérifications de base, telles que la compilation et les tests unitaires. Les étapes ultérieures peuvent inclure des tests d'intégration, de performance et de conformité, ainsi que des déploiements de préproduction dans divers environnements.

Le pipeline peut être configuré de manière à nécessiter des approbations manuelles aux points critiques, par exemple avant le déploiement en production, tout en automatisant les tâches de routine. Cela offre aux développeurs un retour rapide sur l'état de leurs modifications. Cette approche structurée assure la cohérence, réduit les erreurs humaines et fournit une piste d'audit claire du transfert des modifications de code de l'environnement de développement à l'environnement de production. Les pipelines modernes sont souvent implémentés sous forme de code. Ils peuvent ainsi être contrôlés, testés et tenus à jour, de la même manière que le code d'application.

Voici d'autres termes associés à l'approche CI/CD qu'il est important de connaître :

- **Validation :** un changement de code.
- **Job :** instructions qu'un runner doit exécuter.
- **Runner :** agent ou serveur qui exécute chaque tâche individuellement et que vous pouvez démarrer ou arrêter selon les besoins.
- **Étapes :** mot-clé qui définit certaines étapes d'un job, telles que « compilation » ou « déploiement ». Les jobs d'une même étape s'exécutent en parallèle. Les pipelines sont configurés à l'aide d'un fichier YAML nommé .gitlab-ci.yml, soumis au contrôle de version et situé à la racine du projet.



Les avantages de l'approche CI/CD dans le développement logiciel moderne

L'approche CI/CD apporte de nombreux avantages qui transforment le développement logiciel moderne en réduisant considérablement les délais et les risques associés à la livraison de nouvelles fonctionnalités et corrections de bogues. Grâce à une boucle de rétroaction continue, les équipes DevSecOps peuvent garantir que leurs modifications sont automatiquement validées par rapport à l'ensemble du code base.

Au-delà des aspects techniques, l'approche CI/CD favorise une culture de collaboration et de transparence au sein des équipes de développement logiciel. Grâce à une visibilité en temps réel du statut des compilations, des tests et des déploiements, il est plus facile d'identifier et de résoudre les goulots d'étranglement au cours processus de livraison. L'automatisation offerte par l'approche CI/CD réduit également la charge cognitive des développeurs. Ils peuvent ainsi se concentrer sur l'écriture de code plutôt que sur la gestion de processus de déploiement manuels. La satisfaction et la productivité des développeurs s'en trouvent améliorées. Les risques généralement associés aux étapes critiques du processus de publication de logiciel s'en trouvent diminués. Les équipes peuvent expérimenter de nouvelles idées sans craindre de compromettre le projet, sachant que des mécanismes de contrôle robustes, comme les revues de code rapides, sont intégrés au processus. Elles peuvent rapidement revenir à une version antérieure si nécessaire. L'approche CI/CD encourage donc une culture d'innovation et d'amélioration continue.

Voici quelques-uns des principaux avantages de l'approche CI/CD :

Des déploiements plus fréquents. Les équipes peuvent déployer plus régulièrement de petites modifications en toute sécurité, ce qui réduit les risques et la complexité associés aux lancements de nouvelles fonctionnalités à la fois volumineuses et peu fréquentes. Cela permet des cycles de retours plus rapides et un développement plus réactif.

Délai de mise sur le marché plus court. Les processus de compilation et de déploiement automatisés permettent de fournir rapidement de nouvelles fonctionnalités et de corriger les bogues, ce qui réduit considérablement le temps entre l'écriture du code et sa mise à disposition dans l'environnement de production.

Tests plus rapides. En exécutant automatiquement des tests sur les modifications de code et en exécutant plusieurs suites de tests simultanément dans différents environnements, l'approche CI/CD réduit considérablement le temps de test par rapport aux approches séquentielles ou manuelles.

Amélioration de la qualité du code. Grâce aux tests automatisés effectués tout au long du processus de développement, les bogues sont immédiatement détectés et éliminés, sans jamais se retrouver dans la branche principale. Cette sécurité garantit une meilleure qualité globale du code et permet de s'assurer que chaque nouvelle version fonctionne exactement comme prévu.

Récupération plus rapide. L'approche CI/CD facilite la résolution des problèmes et la récupération après incident, réduisant ainsi le temps moyen de résolution (MTTR). Les pratiques de déploiement continu consistent en des petites mises à jour logicielles fréquentes. Ainsi, lorsque des bogues apparaissent, il est plus facile de les identifier. Les développeurs ont la possibilité de corriger rapidement les bogues ou de revenir à une version antérieure afin que le client puisse rapidement reprendre son activité.

Conformité et audits plus simples. Les tâches de conformité peuvent être intégrées dans le cycle de développement, ce qui réduit le risque de publier des applications non conformes. Les systèmes CI/CD conservent des enregistrements détaillés de l'ensemble des compilations, tests et déploiements, ce qui vous donne une piste d'audit complète qui facilite la résolution des problèmes, le maintien des exigences de conformité et la réalisation des audits.

Moins de changement de contexte. Le fait de recevoir des retours en temps réel sur leur code permet aux développeurs de travailler plus facilement sur une seule chose à la fois et de réduire au maximum leur charge cognitive. En manipulant de petites portions de code qui sont testées automatiquement, les développeurs peuvent déboguer leur code rapidement, tout en ayant encore l'esprit plongé dans la programmation. Cela facilite la détection des erreurs, car il y a moins de code à examiner.

Processus cohérents. Les pipelines CI/CD créent un processus standardisé d'intégration et de déploiement du code. Les équipes peuvent ainsi plus facilement collaborer, intégrer de nouveaux développeurs, réduire les risques, respecter les dates de déploiement importantes et, en fin de compte, livrer des logiciels de meilleure qualité plus rapidement.

Développeurs plus heureux (et plus productifs). En renforçant l'automatisation du processus de déploiement et en réduisant le changement de contexte, l'équipe peut consacrer davantage de temps à des projets plus enrichissants pour les développeurs et l'entreprise.

Utilisateurs et clients plus satisfaits. Les bogues en production étant moins nombreux et les nouvelles fonctionnalités et corrections étant mises à disposition plus rapidement et plus souvent, il devient beaucoup plus facile d'améliorer la satisfaction et la fidélisation des clients, et de remporter des contrats avec de nouveaux clients.

Principales différences entre l'approche CI/CD et le développement traditionnel

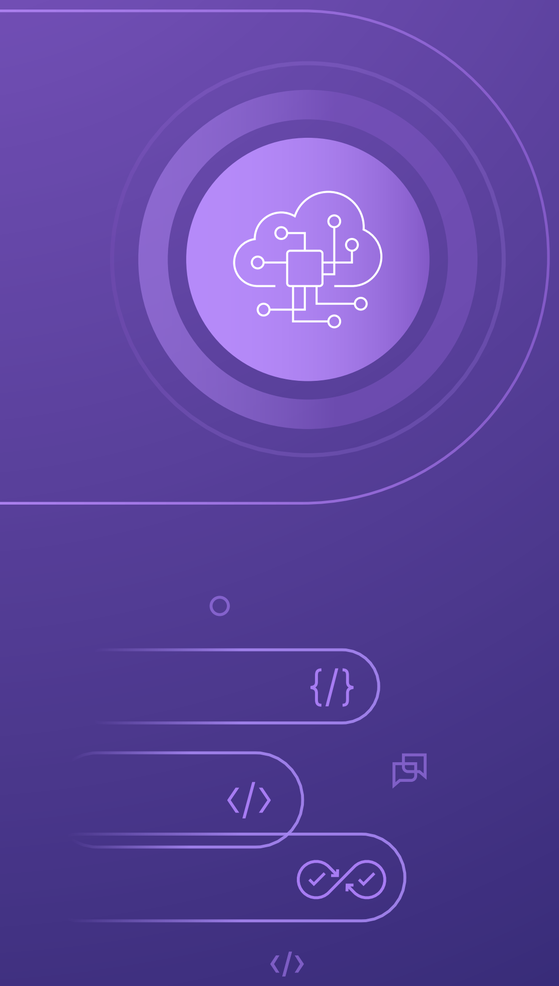
L'approche CI/CD diffère du développement logiciel traditionnel à bien des égards, notamment :

Validations de code fréquentes
Dans le cadre du développement logiciel traditionnel, les développeurs travaillent souvent de manière isolée et intègrent rarement leurs modifications dans le code base principal. Cela entraîne des conflits de merge et d'autres problèmes chronophages. Avec l'approche CI/CD, les développeurs effectuent régulièrement le push des validations, parfois plusieurs fois par jour. Les conflits de merge sont détectés rapidement et le code base est maintenu à jour.

Réduction des risques
Les méthodes de développement logiciel traditionnelles reposent sur des cycles de test à rallonge et une planification rigoureuse avant la sortie de chaque nouvelle version. Bien que ce type de développement ait pour objectif de réduire au maximum les risques, il entrave souvent la capacité à trouver et à résoudre les problèmes. À l'inverse, l'approche CI/CD permet de gérer les risques en appliquant de petites modifications incrémentielles. Ces changements, surveillés de près, peuvent être facilement annulés en cas de problème.

Tests automatisés et continus
Dans le cadre du développement logiciel traditionnel, les tests sont exécutés à la fin du processus de développement, ce qui peut entraîner des retards de livraison et des corrections de bogues coûteuses. L'approche CI/CD, en revanche, intègre des tests automatisés, exécutés en continu tout au long du processus de développement logiciel, déclenchés à chaque validation de code. Elle permet aux développeurs de recevoir des retours immédiats et d'implémenter rapidement les correctifs nécessaires.

Déploiements automatisés, reproductibles et fréquents
L'automatisation des déploiements dans l'approche CI/CD réduit le stress et les efforts habituellement associés aux déploiements massifs de logiciels. Ce processus automatisé est reproductible dans tous les environnements, garantissant ainsi un gain de temps, une réduction des risques d'erreurs et une cohérence accrue à chaque déploiement.



Bonnes pratiques pour la mise en œuvre et la gestion de l'approche CI/CD

Votre maîtrise de l'approche CI/CD dépend grandement des bonnes pratiques que vous mettez en œuvre.

- Validez tôt, validez souvent.
- Optimisez les étapes du pipeline.
- Simplifiez et accélérez les compilations.
- Utilisez les échecs pour améliorer les processus.
- Assurez-vous que l'environnement de test reflète l'environnement de production.
- Pensez à intégrer votre pipeline CI/CD à votre système de gestion du code source (SCM).
- Lancez-vous avec les outils et infrastructures disponibles, puis itérez pour améliorer vos processus.
- Utilisez le moins d'outils possibles pour optimiser la livraison continue.
- Évitez l'accumulation de tickets ou de merge requests en surveillant les progrès en continu.
- Simplifiez les tests d'acceptation par l'utilisateur et le déploiement vers l'environnement de préproduction grâce à l'automatisation.
- Automatisez la gestion du pipeline de sortie des nouvelles versions.
- Mettez en œuvre la surveillance du pipeline pour gagner en visibilité et en productivité.
- Intégrez les fonctionnalités d'IA pour rendre votre processus plus efficace et plus sûr.

Approfondissons quelques-unes de ces bonnes pratiques.

Intégration de votre pipeline CI/CD à votre système de SCM

La gestion du code source (SCM) et les fonctionnalités CI/CD constituent la base des pratiques modernes de développement logiciel. Les systèmes SCM, comme Git, offrent une solution centralisée pour suivre les modifications, gérer les versions de code et faciliter la collaboration entre les membres de l'équipe, tout en maintenant une branche principale stable qui contient un code prêt à être déployé dans l'environnement de production.

L'approche CI/CD automatise les étapes de compilation, de test et de validation du code géré par le système SCM. Si le système SCM et l'outil CI/CD sont étroitement intégrés (ou qu'ils font idéalement partie de la même plateforme DevSecOps), ce processus est automatisé. Lorsqu'un développeur soumet ses modifications de code, le système CI/CD récupère automatiquement le code le plus récent, le combine avec le code base existant, puis exécute une série de vérifications automatisées. Celles-ci comprennent généralement la compilation du code, l'exécution de tests unitaires, l'analyse statique du code et la vérification de la couverture de code. En cas d'échec d'une de ces étapes, l'équipe en est immédiatement informée, ce qui lui permet de résoudre les problèmes avant qu'ils n'affectent d'autres développeurs ou qu'ils ne soient reflétés dans l'environnement de production.

Cette intégration étroite entre le contrôle de version et l'intégration continue crée une boucle de rétroaction constante. Elle garantit la qualité du code, facilite la correction des failles de sécurité et prévient l'accumulation de problèmes d'intégration. Elle améliore également la productivité et accélère le processus de développement.

En revanche, en utilisant un système SCM et des outils CI/CD disparates ou qui ne sont pas étroitement intégrés, les développeurs doivent constamment basculer de l'un à l'autre, en déclenchant manuellement les compilations après les validations et en recopiant les informations de compilation dans les merge requests. Ce changement de contexte augmente les risques d'erreur humaine et impacte négativement la traçabilité entre les modifications de code et les déploiements. Le manque d'intégration rend également plus difficile la mise en œuvre de stratégies automatisées couvrant l'ensemble du cycle de développement logiciel, ce qui oblige les équipes à créer des scripts personnalisés ou des webhooks pour combler l'écart entre les systèmes. Lors de la résolution de problèmes, les développeurs doivent rassembler des informations provenant de divers outils qui ne sont pas connectés entre eux. Il leur faut donc plus de temps pour comprendre l'origine du problème et l'étape du pipeline où l'erreur s'est produite.

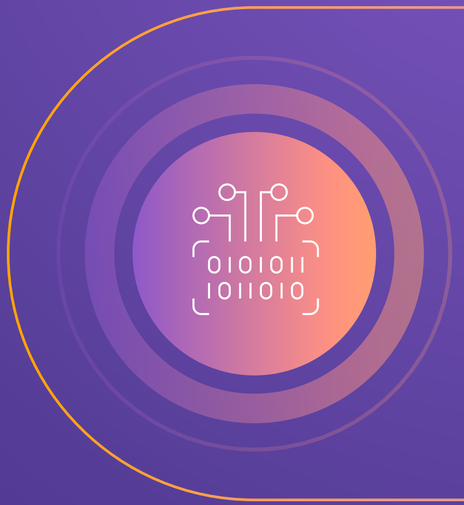
Optimisation de l'approche CI/CD avec l'IA

L'IA contribue à améliorer les processus CI/CD de plusieurs façons tout au long du cycle de développement.

De nombreuses entreprises optent pour l’IA afin d’identifier et de résoudre les problèmes qui surviennent dans leurs pipelines CI/CD. Au lieu d'explorer les job logs, les messages d'erreur et les traces d'exécution pour tenter de déterminer pourquoi un job CI/CD a échoué, les développeurs peuvent utiliser l'IA pour déterminer la cause profonde et suggérer une solution.

L'IA leur permet également de mieux comprendre, hiérarchiser et corriger les problèmes de sécurité et les vulnérabilités détectés dans leur code. Par exemple, lorsqu’un scan de sécurité exécuté au sein du pipeline CI/CD détecte une vulnérabilité, l’IA peut la résumer en quelques phrases précises, fournir des exemples de la façon dont elle pourrait être exploitée et suggérer le code nécessaire pour la corriger.

À elle seule, l'approche CI/CD est un moyen puissant de livrer des logiciels de meilleure qualité plus rapidement. Avec l'IA, votre processus de livraison de logiciels est encore plus efficace et plus sécurisé.



Premiers pas avec l'approche CI/CD

Pour commencer à utiliser les méthodes CI/CD, identifiez un projet simple mais représentatif qui servira de projet pilote. Sélectionnez une application simple avec des exigences de test de base. Vous pourrez ainsi vous concentrer sur l'apprentissage du fonctionnement des pipelines plutôt que sur des scénarios de déploiement complexes. Assurez-vous que votre code est soumis au contrôle de version et qu'il fait l'objet de tests automatisés de base, ne serait-ce que quelques tests unitaires. L'objectif est de créer un pipeline basique que vous pourrez améliorer progressivement à mesure que vos compétences progressent.

Dans le cas de GitLab, le processus commence par la création d'un fichier .gitlab-ci.yml dans le répertoire racine de votre projet. Ce fichier YAML définit les étapes (de base comme la compilation, les tests et le déploiement) et les jobs de votre pipeline. Voici un exemple de pipeline simple : l'étape « Compilation » compile votre code et génère des artefacts, l'étape « Test » exécute les tests unitaires et l'étape « Déploiement » effectue le push de votre application vers un environnement de préproduction. GitLab détecte automatiquement ce fichier et commence à exécuter votre pipeline chaque fois que des modifications sont transmises via un push à votre dépôt. La plateforme fournit des runners intégrés pour exécuter les jobs de votre pipeline, mais vous pouvez également configurer vos propres runners si vous souhaitez davantage de contrôle.

Une fois que vous maîtrisez les principes de base, enrichissez votre pipeline progressivement avec des fonctionnalités plus avancées. Par exemple, ajoutez des contrôles pour vérifier la qualité du code, exécutez un scan de sécurité ou automatisez le déploiement du nouveau code dans l'environnement de production. La plateforme DevSecOps de GitLab inclut des fonctionnalités telles que la gestion de la conformité, les variables de déploiement et les portes d'approbation manuelle que vous pouvez intégrer à mesure que votre pipeline évolue. Soyez attentif à la durée d'exécution du pipeline et exécutez dans la mesure du possible des jobs en parallèle. Pensez à ajouter des notifications et un traitement approprié des erreurs afin que les membres de l'équipe soient rapidement informés en cas de pipeline défectueux. Pensez à documenter les problèmes que vous rencontrez le plus souvent en indiquant les solutions adoptées. Ces données deviendront inestimables quand votre équipe s'agrandira.

Essayez GitLab CI/CD.

Inscrivez-vous à un essai de GitLab Ultimate et essayez gratuitement pendant 60 jours la plateforme DevSecOps alimentée par l'IA.

En savoir plus

À propos de GitLab

GitLab est la plateforme DevSecOps alimentée par l'IA la plus complète pour l'innovation logicielle. Elle offre une interface centralisée avec un magasin de données unifié, un modèle d'autorisation unique, une chaîne de valeur cohérente, et des rapports centralisés, permettant de sécuriser, déployer et collaborer autour de votre code en un seul endroit. Il s'agit de la seule véritable plateforme DevSecOps complète et compatible avec tous les clouds qui regroupe et centralise toutes les fonctionnalités DevSecOps.

Avec GitLab, les entreprises peuvent créer, livrer et gérer du code rapidement et en continu, et ainsi donner vie à leurs ambitions. GitLab permet aux clients et aux utilisateurs d'innover plus rapidement, de s'adapter plus facilement, de servir et fidéliser les clients plus efficacement. GitLab est une plateforme open source qui s'appuie sur sa communauté grandissante, constituée de milliers de développeurs et de millions d'utilisateurs, pour proposer en permanence des innovations.

GitLab