Fonctionnalités clés de Gitlab

Le processus de versionnement avec Git permet aux équipes de travailler simultanément sur des parties distinctes du code tout en préservant l'intégrité du projet. Les fonctionnalités additionnelles de Gitlab offrent une approche complète pour gérer les aspects techniques et collaboratifs d'un projet.

1. Fonctionnalités avancées de gestion de tâches/tickets

Au-delà du simple suivi des problèmes, une forge logicielle offre des fonctionnalités avancées telles que la possibilité d'attribuer des priorités, d'associer des tâches à des jalons (milestones) et de créer des filtres personnalisés.

2. Tableaux Kanban pour la gestion de projet

Les tableaux Kanban intégrés à une forge logicielle offrent une visualisation claire de l'état d'avancement des tâches. Cela permet aux équipes de suivre les travaux en cours, les tâches à venir, et celles terminées, facilitant ainsi une gestion agile du projet.

3. Intégration d'outils de test

L'intégration d'outils de test au sein d'une forge logicielle permet de garantir la qualité du code et la stabilité du projet. Cette fonctionnalité se manifeste notamment dans le processus des merge requests (demande de fusion de code), où chaque modification du code est soumise à une série de tests automatisés avant d'être fusionnée dans la branche principale. Cette approche, souvent appelée "pipeline de tests", offre une assurance qualité en détectant rapidement les éventuels problèmes introduits par une modification particulière.

L'un des aspects clés de cette intégration est l'utilisation de Quality Gates. Ces portes de qualité définissent des critères spécifiques que le code doit satisfaire pour être accepté. Les Quality Gates peuvent inclure des tests unitaires, des tests d'intégration, des analyses statiques du code et d'autres métriques de qualité. Si le code ne répond pas à ces critères, la merge request peut être automatiquement rejetée, empêchant ainsi l'introduction de défauts dans la branche principale, en attendant des corrections futures.

De plus, l'intégration d'outils de test peut fournir des rapports détaillés sur les performances du code. Cela inclut des informations sur la couverture des tests, les temps d'exécution, et d'autres métriques importantes. Ces données permettent aux développeurs d'identifier rapidement les zones du code qui nécessitent une attention particulière et facilitent la prise de décisions éclairées lors de la validation des modifications.

Cette approche d'intégration d'outils de test au niveau des merge requests offre plusieurs avantages :

- Réduction des erreurs : les tests automatisés permettent de détecter les erreurs potentielles avant qu'elles ne soient fusionnées dans la branche principale, contribuant ainsi à maintenir la stabilité du code.

- Réaction rapide : les développeurs reçoivent rapidement des informations sur la qualité de leur code, favorisant une correction rapide en cas de problème.
- Transparence : les rapports générés par les tests offrent une visibilité complète sur la qualité du code, renforçant la transparence au sein de l'équipe.
- Prise de décisions éclairée : les données de performance du code permettent aux équipes de prendre des décisions éclairées lors de l'acceptation ou du rejet d'une merge request.

En revanche, la mise en place de ces outils peut s'avérer complexe techniquement et nécessite forcément une adoption large au sein de l'équipe de développement concernées. La maturité de l'équipe est un atout pour l'exploitation efficace de ces outils. En somme, l'intégration d'outils de test au sein des merge requests constitue une pratique essentielle pour maintenir la qualité du code et assurer un développement logiciel robuste et fiable.

4. Gestion des collaborateurs et des accès

La fonctionnalité de gestion des collaborateurs et des accès au sein d'une forge logicielle offre un ensemble complet d'outils pour structurer la collaboration, en tenant compte de la hiérarchie entre groupes, sous-groupes et projets. Cette gestion fine des autorisations garantit une sécurité et une confidentialité adéquates tout en facilitant une collaboration efficace.

- Définition des rôles et des permissions au niveau du groupe : au niveau du groupe, Gitlab permet la création de rôles spécifiques avec des permissions prédéfinies (invité, rapporteur, développeur, mainteneur, propriétaire). Ces rôles peuvent être attribués aux utilisateurs en fonction de leurs responsabilités, assurant une granularité dans la gestion des accès.
- Héritage des permissions entre groupes, sous-groupes et projets : une forge logicielle offre la possibilité de définir précisément qui peut voir, contribuer, ou administrer un groupe ou projet particulier. Elle permet également l'héritage des permissions, simplifiant ainsi la gestion en propageant automatiquement les autorisations du groupe parent vers ses sous-groupes et projets. Cela garantit une cohérence dans la définition des rôles et permissions.
- Invitations et gestion des utilisateurs externes : pour les collaborations externes, la forge logicielle propose souvent la possibilité d'inviter des utilisateurs externes avec des permissions spécifiques. Cela assure une collaboration sécurisée et transparente avec des parties extérieures à l'organisation.
- Les collègues du CIRAD ont produit une documentation décrivant plusieurs scénarios de gestion des projets, groupes et les permissions correspondantes : https://gitlab.cirad.fr/cirad/documentation/-/wikis/Les%20sc%C3%A9narios

5. Système de Déploiement Continu (CI/CD)

La fonctionnalité de Système de Déploiement Continu (CI/CD) au sein d'une forge logicielle joue un rôle central dans l'automatisation du processus de développement, de tests, et de déploiement. Cette approche combinant l'Intégration Continue (CI) et le Déploiement Continu (CD), offre plusieurs avantages significatifs pour les équipes de développement.

- Intégration Continue (CI) : la CI automatise le processus d'intégration du code produit par un ou plusieurs développeurs. À chaque modification de code, un processus d'intégration est déclenché, vérifiant la compatibilité et la cohérence du code nouvellement ajouté avec le reste du projet. Cela détecte rapidement les conflits potentiels et les erreurs d'intégration, assurant une qualité continue du code.

- Déploiement Continu (CD) : la CD étend la CI en automatisant également le processus de déploiement du code intégré vers un environnement de production ou de pré-production. Cela permet une mise en production régulière et fiable, réduisant ainsi le risque d'erreurs liées au déploiement manuel.

- Pipelines d'intégration et de déploiement : les pipelines CI/CD définissent les étapes spécifiques du processus, de la compilation du code à son déploiement. Chaque étape est automatisée et peut être configurée pour inclure des tests unitaires, des tests d'intégration, des analyses statiques, et d'autres processus critiques pour la qualité du code. Une validation manuelle peut toutefois être conservée dans le processus afin de maintenir un contrôle manuel

- Avantages de l'automatisation : L'automatisation du CI/CD offre une augmentation significative de la productivité en réduisant le temps nécessaire pour tester et déployer le code. Cela permet aux développeurs de se concentrer davantage sur la création de fonctionnalités plutôt que sur les tâches répétitives et manuelles.

- Feedback rapide : Grâce aux pipelines CI/CD, les développeurs reçoivent un retour très rapide sur la qualité de leur code. Cela favorise une approche itérative du développement où les corrections peuvent être apportées rapidement, améliorant ainsi la qualité du logiciel.

- Utilisation de Runners locaux : l'utilisation de Runners au sein de votre plateforme peut renforcer l'efficacité du processus CI/CD. Ces Runners locaux permettent d'optimiser les performances en effectuant les tâches de compilation et de test directement sur vos ressources informatiques de manière exclusive (par rapport à des Runners partagés fournis par la forge logicielle). Cela concourt à réduire ainsi le temps d'exécution global du pipeline. De même, l'exécution des pipelines sur votre infrastructure peut grandement simplifier le déploiement de votre code du fait que votre Runner peut accéder au réseau des serveurs sous votre responsabilité.

- Déploiements sécurisés et reproductibles : La CD assure des déploiements sécurisés et plus simplement reproductibles. Chaque version déployée est associée à une version spécifique

du code source, permettant une traçabilité précise et la possibilité de revenir à des versions antérieures en cas de besoin.

- Gestion des Environnements : Les outils de CI/CD facilitent la gestion des différents environnements, tels que les environnements de développement, de test et de production. Cela permet une transition fluide du code à travers ces environnements avec une assurance qualité constante.

L'intégration continue et le déploiement continu, avec l'utilisation de Runners locaux, offrent un moyen efficace de garantir la qualité du code, d'accélérer le processus de déploiement, et de fournir un feedback rapide aux développeurs. En automatisant ces processus, les équipes peuvent garantir des déploiements réguliers et fiables, contribuant ainsi à la robustesse et à la stabilité du logiciel.

6. Système de commentaires collaboratifs

Les systèmes de commentaires intégrés facilitent la communication entre les membres de l'équipe. Que ce soit au niveau du code, des problèmes, ou des merge requests, ces commentaires encouragent une collaboration étroite et permettent une compréhension partagée du travail en cours. En particulier, le principe d'un merge request vise à permettre la relecture d'un ensemble de modifications du code source. Il regroupe l'ensemble des commits concourant à développer une nouvelle fonctionnalité ou à corriger un bug. La relecture assure une revue par les pairs (similaire à la revue d'une publication scientifique, mais sur un périmètre restreint à l'équipe de développement) qui peuvent ainsi commenter des portions de code et proposer éventuellement des corrections ou améliorations, avant de valider la demande d'intégration du merge request.

7. Sites web statiques

GitLab Pages est une fonctionnalité qui permet aux utilisateurs de déployer des sites web statiques directement à partir de leurs dépôts Git. Il est ainsi possible de publier facilement des contenus comme des portfolios, des documentations, ou des blogs, en utilisant des générateurs de sites statiques tels que Jekyll, Hugo, ou GitBook. Les utilisateurs peuvent configurer des pipelines CI/CD pour automatiser le processus de déploiement, et GitLab Pages fournit une URL personnalisée pour accéder au site. De plus, GitLab Pages prend en charge des fonctionnalités comme le HTTPS gratuit, permettant un accès sécurisé aux sites web.

8. Registres

Les registres GitLab sont des fonctionnalités intégrées qui permettent aux utilisateurs de stocker, gérer et distribuer des conteneurs Docker, des packages de code, des artefacts, et des modèles directement depuis leurs projets GitLab.

- GitLab Container Registry permet aux développeurs de stocker et de gérer des images Docker directement dans leur dépôt GitLab, facilitant ainsi l'intégration des images dans les pipelines CI/CD.

- GitLab Package Registry offre un espace pour stocker et partager des packages de divers formats (comme NPM, Maven, Composer, etc.), ce qui permet de centraliser la gestion des dépendances et des bibliothèques utilisées dans les projets.

- GitLab Model Registry conçu pour gérer et partager des modèles de machine learning, facilitant leur réutilisation et collaboration au sein des équipes.

Ces registres permettent d'améliorer le flux de travail des développeurs en fournissant des outils pour la gestion des artefacts et des dépendances, tout en garantissant une intégration fluide avec les pipelines de déploiement et les processus CI/CD.

Pour citer cette fiche :
INRAE DipSO (2024). Fonctionnalités clés de Gitlab. Formation OSCAR Ouvrir la science, connaissance à acquérir : module Gestion et partage des algorithmes, codes et logiciels. 5 p.
oscar.inrae.fr