Engineer

GitLab

plateforme Git que l'on peut aussi exploiter soi-même

Ce que c’est

GitLab est une application unique bâtie autour des dépôts Git : elle héberge le code et y ajoute des demandes de fusion pour la revue, des tickets pour la planification et un système CI/CD intégré qui compile, teste et déploie depuis le même endroit. Elle se distingue par le choix de son lieu d’exécution. Le même produit existe en GitLab.com, le service exploité par GitLab ; en GitLab Self-Managed, à installer sur ses propres serveurs ; et en GitLab Dedicated, une instance réservée à un seul client, hébergée et entretenue par GitLab dans la région AWS que ce client préfère.

En dessous se trouve Git ordinaire. Chaque projet est un dépôt Git que l’on peut cloner, vers lequel on peut pousser des commits et depuis lequel on peut en récupérer avec les commandes habituelles, et l’historique d’un clone est complet. Tout ce que GitLab ajoute, des fils de revue et tickets aux résultats des pipelines et aux autorisations, est conservé par GitLab lui-même, hors du dépôt : un clone n’en emporte rien.

Une merge request est le nom donné par GitLab à un changement proposé en attente de fusion : la même idée que d’autres plateformes appellent pull request. La documentation la décrit comme le lieu central de la revue de code, de la discussion et du suivi de la modification. Une page montre la description, les changements de code avec des commentaires de revue dans les lignes concernées, l’état des pipelines CI/CD, les rapports indiquant si la fusion est possible et la liste des commits. Une demande peut être liée à un ticket et le fermer automatiquement lors de sa fusion. Sans modification des réglages initiaux, la branche de référence du projet, généralement main, est protégée : seules les personnes ayant le rôle Maintainer ou supérieur peuvent y fusionner des changements.

Le système CI/CD est configuré dans un fichier nommé `.gitlab-ci.yml` à la racine du projet. Il déclare les étapes, tâches et scripts qui forment ensemble un pipeline : les étapes définissent l'ordre, par exemple compiler, puis tester, puis déployer ; les tâches sont les travaux effectués dans chaque étape. Elles sont exécutées par GitLab Runner, un programme distinct écrit en Go et livré en un seul binaire. Un runner se connecte à une instance GitLab, attend des tâches, exécute chacune sur la machine elle-même, dans un conteneur Docker ou sur un serveur distant par SSH, puis renvoie le résultat.

Utilisez le concept pour prendre une décision

Utilisez « Qu'est-ce que GitLab ? » pour éclairer une décision réelle. Nous pouvons vous aider à relier ce concept au système que vous construisez ou achetez. Des changements testés, publiés et réversibles, pour livrer aux clients un mardi après-midi plutôt qu'un vendredi soir tard.

Découvrir nos prestations Contactez-nous →

Vérifiez les hypothèses derrière le terme

Appuyez-vous sur l'explication et les réalisations liées pour formuler les questions de votre projet. Nous pouvons examiner les hypothèses avec vous avant le choix d'un outil. Déploiement SaaS GIS automatisé : GitHub Actions — déploiement, données et reporting automatisés

Déploiement SaaS GIS automatisé : GitHub Actions Contactez-nous →

Pourquoi c'est important

La plupart des activités d'une équipe logicielle entre l'écriture et la livraison du code peuvent se dérouler dans une instance GitLab : le code, la revue, le pipeline qui le teste, le ticket qui l'a demandé et la trace de qui a approuvé quoi. Une application signifie un ensemble de comptes et de droits, un endroit où chercher et des liens durables entre un ticket, la demande de fusion qui y a répondu et le pipeline qui l'a vérifiée.

La possibilité de l'exploiter soi-même est la raison même pour laquelle beaucoup d'organisations choisissent GitLab. Lorsque le code ne peut pas quitter l'infrastructure de l'entreprise, à cause d'un contrat, d'une autorité de contrôle ou d'une simple règle, un service hébergé est exclu. GitLab peut alors être installé dans le réseau qui doit le garder. La documentation en fait un usage de premier plan : elle publie des architectures de référence pour des installations de 1 000 à 50 000 utilisateurs, avec le paquet Linux et un chart Helm pour Kubernetes.

Ce contrôle a un prix, payé en exploitation et pas seulement en licences. Exploiter GitLab soi-même impose de dimensionner un serveur, de maintenir en bon état sa base PostgreSQL et son stockage Redis ou Valkey, de fournir des

machines aux runners, de prendre des sauvegardes réellement restaurables et de suivre les mises à niveau dans l'ordre exigé par le projet. Les offres hébergées confient ce travail à GitLab ; l'offre autogérée vous le laisse.

À quoi cela ressemble en pratique

Le travail de l'entreprise éclaire ces deux aspects de GitLab.

Au cours de plusieurs années d'ingénierie des données, des pratiques CI/CD fondées sur GitHub et GitLab, avec Bash et Python pour assurer l'automatisation sous-jacente, ont remplacé les étapes manuelles dans l'analyse des données comme dans le développement logiciel. Le passage du développement à la livraison a été standardisé pour être identique à chaque fois, au lieu d'être réinventé pour chaque projet. L'analyse, longtemps traitée comme un monde distinct et plus manuel, a été intégrée au même pipeline que le développement. La page consigne environ 4 000 heures économisées et des livraisons plus régulières, parce que le travail n'était plus réalisé un peu différemment à chaque fois.

Le code de l'entreprise illustre une autre manière d'aborder l'auto-hébergement. Lorsqu'il a quitté un hébergeur tiers, il est passé sur une forge exploitée par l'entreprise, en un seul binaire, centrée sur SSH et sans interface web d'administration, choisie selon le principe que moins il y a d'interfaces, moins il y a à défendre. Cette forge partage un hôte de 512 Mo de mémoire pour les services accessibles au public, dont environ 464 Mo utilisables, avec un serveur web pour sept domaines, un service onion Tor, un serveur Gemini, un serveur Gopher et le blocage des intrusions. L'automatisation des sauvegardes est implémentée séparément ; cela ne signifie pas que les sauvegardes sont actives.

Aucune des deux pages ne mentionne GitLab : la comparaison qui suit est le calcul propre à cet article. GitLab documente 8 Go de mémoire comme le minimum pour faire fonctionner une installation sur un seul nœud ; cet hôte entier en possède un seizième. Pouvoir exploiter GitLab soi-même suppose d'avoir un serveur dimensionné pour GitLab. Une petite forge dédiée à un seul usage est l'autre moyen de garder le code sur ses propres machines.

Un fichier de pipeline minimal montre la forme de la configuration CI/CD, avec deux étapes exécutées dans l'ordre et une tâche dans chacune :

```
stages:
    - build
    - test

build-site:
    stage: build
    script:
        - make build

run-tests:
    stage: test
    script:
        - make test
```

Où les problèmes surviennent

- **Un petit serveur ralentit toute l'instance.** La documentation de GitLab donne 16 Go de mémoire comme base pour une installation sur un seul nœud et 8 Go comme minimum dans un environnement à mémoire limitée. Elle avertit aussi que, sous charge, le swap peut fortement dégrader les performances et demande de le désactiver si possible, ou de prévoir assez de mémoire pour que GitLab ne l'utilise jamais.
- **Une sauvegarde est restaurée dans une instance incapable de lire ses propres données.** La commande de sauvegarde omet le répertoire de configuration `/etc/gitlab` et, avec lui, `gitlab-secrets.json`, les clés qui déchiffrent les valeurs chiffrées de la base. Sans ce fichier, les valeurs restent verrouillées dans la base restaurée. La documentation demande de sauvegarder la configuration séparément, au minimum `gitlab-secrets.json` et `gitlab.rb`.
- **Une mise à niveau saute un arrêt obligatoire.** Les parcours de mise à niveau comportent des versions à installer avant toute version ultérieure ; depuis GitLab 17.5, elles se situent en x.2, x.5, x.8 et x.11. Les migrations en arrière-plan de chaque arrêt doivent être terminées avant la mise à niveau suivante : une instance en retard de plusieurs versions rattrape donc son retard un arrêt à la fois.
- **Un stockage géré n'est finalement pas pris en charge.** PostgreSQL est la seule base de données prise en charge. Redis ou Valkey conserve les sessions et les tâches en arrière-plan et doit être une instance autonome : Redis Cluster et les variantes sans serveur de Redis et Valkey ne sont pas pris en charge. Un service cloud géré est autorisé, à condition de relever de l'une des formes autonomes.

- **Les pipelines attendent des machines que personne n'avait prévues.** GitLab.com fournit des runners prêts à l'emploi, mais sur une instance autogérée, l'administrateur fournit et gère l'infrastructure d'exécution des tâches CI/CD. Il doit notamment veiller à ce que les runners aient assez de capacité pour la charge de l'organisation. Une équipe très occupée avec un seul petit runner fait la queue derrière lui.

Pour aller plus loin

- Documentation GitLab — la documentation complète pour utiliser, installer et administrer GitLab.
- Prérequis d'installation — mémoire, processeur, stockage et versions de PostgreSQL et Redis prises en charge.
- Sauvegarder GitLab — ce qu'une sauvegarde contient, ce qu'elle omet et comment conserver les secrets à côté.
- Parcours de mise à niveau — les arrêts obligatoires et la raison pour laquelle les migrations en arrière-plan doivent se terminer entre eux.
- Premiers pas avec GitLab CI/CD — le fichier `.gitlab-ci.yml`, les étapes, tâches et pipelines.
- GitLab Runner — le programme qui exécute les tâches et ses modes d'exécution.

Glossaire

Choisissez selon vos contraintes réelles

Présentez le parcours de déploiement, les risques opérationnels et les personnes d'astreinte. Nous pouvons définir un changement avec des vérifications, une procédure de reprise et une responsabilité claire.

Consulter les tarifs Contactez-nous →

Rendez la terminologie concrète

Demandez comment « Qu'est-ce que GitLab ? » s'applique à votre organisation. Un exemple concret tiré de votre système actuel suffit pour engager la discussion.

Découvrir nos prestations

welcome@engineer.company +45 50 10 61 84 Copenhague, Danemark

Emportez cette page avec vous

Fiche de contact

Créé avec soin et amour par D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Droit d'auteur © 2025 – aujourd'hui · Engineer ApS · Tous droits réservés.

Imprimé depuis https://engineer.company/fr/glossary/gitlab/