Engineer

GitHub

hébergement Microsoft pour les dépôts Git

Ce que c’est

GitHub est un service hébergé appartenant à Microsoft qui conserve les dépôts Git sur ses serveurs et construit autour d’eux un lieu de travail : une vue web du code et de son historique, des demandes d’intégration pour proposer et revoir des changements, des tickets pour suivre le travail et GitHub Actions pour exécuter des tâches automatiquement lorsqu’un événement se produit dans un dépôt.

À la base se trouve Git ordinaire. Un dépôt cloné depuis GitHub est un dépôt Git normal avec tout son historique. Le travail circule entre un ordinateur portable et GitHub par les push et fetch habituels. Ce qu’ajoute GitHub reste de son côté de la connexion : discussions de revue, tickets, règles de branche, paramètres d’automatisation et liste des droits de chacun sont conservés dans les propres systèmes de GitHub, pas dans le dépôt cloné.

Une pull request est le nom que GitHub donne à une proposition de fusion des changements d’une branche dans une autre. Elle montre les différences ligne par ligne, recueille commentaires et approbations et affiche le résultat de chaque contrôle automatique exécuté sur le code proposé. Une demande ouverte comme brouillon est marquée inachevée et ne peut pas être fusionnée avant que quelqu’un la déclare prête pour la revue.

GitHub Actions est l’automatisation intégrée au service. Un workflow est un fichier YAML dans le répertoire `.github/workflows` du dépôt. Il nomme les événements qui le déclenchent, comme un push, une nouvelle demande d’intégration ou une planification horaire, et énumère des tâches composées d’étapes. Chaque étape exécute une commande shell ou une action, un composant d’automatisation fourni sous forme de paquet et partageable entre

dépôts. Les tâches s'exécutent sur des runners : des machines virtuelles sous Ubuntu Linux, Windows ou macOS que GitHub démarre à neuf pour chaque exécution, ou vos propres machines enregistrées comme runners auto-hébergés.

Deux autres services surveillent ce qui entre dans un dépôt. Dependabot lit un fichier `dependabot.yml` et ouvre des demandes d'intégration qui actualisent les dépendances. Séparément, il en ouvre pour les dépendances ayant des vulnérabilités connues. La protection des pushs inspecte les envois entrants à la recherche de secrets comme des jetons d'accès et bloque un push qui en contient avant qu'il n'atteigne le dépôt.

Utilisez le concept pour prendre une décision

Utilisez « Qu'est-ce que GitHub ? » 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

Pour beaucoup d'équipes, GitHub est le lieu où le travail s'effectue, pas seulement où il est stocké. Le code, la discussion de chaque changement, la trace de ses approbateurs et le résultat de chaque contrôle se côtoient. Une seule page permet donc généralement de comprendre pourquoi une ligne existe.

L'automatisation compte autant que l'hébergement. Avec Actions à côté du code, une demande d'intégration peut porter ses propres preuves : les tests ont été exécutés et ont réussi, la compilation a fonctionné, les scanners ont rapporté leurs observations. Les branches protégées font de ces preuves une règle. Une branche peut n'accepter les changements que par une demande d'intégration, après un nombre fixé de revues favorables et une fois réussis des contrôles d'état nommés. La ligne principale n'avance ainsi que par des changements revus et contrôlés.

Comme GitHub se trouve au point d'entrée du code dans le projet, c'est aussi là que résident les contrôles de la chaîne d'approvisionnement : quelles actions tierces un workflow peut exécuter, à quelle version exacte, comment les dépendances sont tenues à jour et si un secret est arrêté lors du push. Ces

contrôles sont le plus souvent activés et réglés par dépôt ou organisation. Une grande partie de la sécurité réelle d'un projet GitHub réside donc dans des paramètres invisibles lors d'une revue de code.

C'est la contrepartie d'un service hébergé. Vous obtenez un environnement complet et entretenu sans exploiter de serveurs ; en échange, les règles qui protègent votre code se configurent dans l'interface d'un tiers, avec des réglages prédéfinis choisis pour le cas général plutôt que pour le vôtre.

À quoi cela ressemble en pratique

Trois pages consacrées au travail de l'entreprise montrent différents usages de GitHub.

Une chaîne de livraison construite sur GitHub Actions sépare le travail en workflows distincts : un pour les contrôles de qualité du code, un pour les tests et un par environnement pour les déploiements. CodeQL, la revue des dépendances et une étape de nomenclature des composants logiciels s'exécutent à leurs côtés. Le parcours va du commit à un service fonctionnant dans Azure, et la promotion fait passer une version construite d'un environnement à l'autre selon un trajet défini, au lieu de le faire à la main. Le frontend est livré sur une image distroless sans shell ni gestionnaire de paquets, l'API Go sur une base Alpine allégée.

Un programme continu d'analyse de sécurité utilise les contrôles qui entourent le code sur GitHub. Chaque GitHub Action tierce est épinglée à un hachage de commit plutôt qu'à une étiquette : personne ne peut déplacer une étiquette et changer ce qu'exécute un workflow. Dependabot surveille 6 écosystèmes, Gitleaks analyse l'historique à la recherche de secrets d'authentification et chaque compilation produit une nomenclature des composants logiciels avec Anchore et Syft. La page en nomme aussi le vrai coût : le volume des signalements. Un scanner qui signale tout apprend aux gens à l'ignorer ; garder un signal exploitable demande donc un tri continu des signalements.

Le déploiement d'une application SIG en logiciel-service a déplacé son cycle de test, compilation et déploiement dans des workflows GitHub Actions : une livraison ne dépendait plus du fait que quelqu'un se souvienne des étapes à suivre. Le traitement récurrent des données et les rapports sont passés dans des tâches Python, Bash et SQL planifiées, et la configuration et les secrets ont été standardisés pour que chaque environnement se comporte de la même façon.

Voici l'épinglage dans un workflow. La longue chaîne hexadécimale est l'identifiant complet du commit de la version qui a été revue, et le commentaire conserve à côté la version lisible par une personne :

```
steps:
    - uses: actions/checkout@3d3c42e5aac5ba805825da76410c181273ba90b1 # v7.0.1
```

Où les problèmes surviennent

- **Une action épinglée à une étiquette change sans prévenir.** Une étiquette comme `v4` peut être déplacée ou supprimée par qui contrôle le dépôt de l'action : le code exécuté demain par un workflow peut donc différer du code revu aujourd'hui. GitHub recommande d'épingler l'action à un SHA de commit complet, qu'un attaquant ne pourrait imiter qu'en produisant « une collision SHA-1 pour le contenu d'un objet Git valide ».
- **Un jeton de workflow possède plus de droits que sa tâche n'en exige.** Chaque exécution reçoit un `GITHUB_TOKEN`, dont les autorisations déterminent ce qu'une étape compromise pourrait modifier. GitHub recommande d'adopter la lecture des contenus du dépôt comme autorisation de base et de ne donner l'écriture qu'aux tâches qui en ont besoin.
- **Du texte non fiable s'exécute comme une commande.** Un workflow qui insère directement le titre d'une demande d'intégration dans un script intégré au workflow laisse la personne qui a rédigé le titre choisir une partie du script. Placer la valeur dans une variable d'environnement intermédiaire la conserve comme donnée. Les déclencheurs `pull_request_target` et `workflow_run` lancent des workflows privilégiés, et GitHub avertit qu'y extraire le code d'une demande d'intégration non fiable expose le dépôt à une compromission.
- **Les administrateurs passent outre les règles de branche.** Sans réglage contraire, une règle de protection de branche exempte de ses restrictions les personnes ayant des droits d'administration sur le dépôt. Activer « Do not allow bypassing the above settings », l'option interdisant de contourner les paramètres ci-dessus, les soumet aussi aux règles.
- **Un secret passe à travers la protection des pushs.** La forme préactivée protège les pushs que chaque utilisateur effectue lui-même vers les dépôts publics. Celle qui protège un dépôt contre tous les contributeurs exige GitHub Secret Protection et reste désactivée jusqu'à son activation explicite. Même alors, toute personne ayant le droit d'écriture peut contourner un blocage en donnant un motif, ce qui déclenche une alerte et s'inscrit dans le journal d'audit. Une fois poussé, un secret d'authentification vit dans chaque clone : renouvelez-le au lieu de compter sur une réécriture de l'historique.

Pour aller plus loin

- Documentation GitHub — la documentation complète des dépôts, demandes d'intégration, Actions et fonctions de sécurité.
- Comprendre GitHub Actions — workflows, événements, tâches, étapes et runners, et leurs relations.
- Référence d'utilisation sécurisée — épinglage des actions, droits des jetons et injection de script, expliqués par GitHub lui-même.
- À propos des branches protégées — revues et contrôles d'état obligatoires, et personnes autorisées à les contourner.
- Mises à jour de version Dependabot — comment un fichier `dependabot.yml` pilote les demandes de mise à jour.
- Protection des pushs — les deux formes de protection, ce que chacune bloque et la consignation des contournements.

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 GitHub ? » 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/github/