Engineer

Forge

plateforme d'hébergement et de collaboration sur le code

Ce que c’est

Une forge, aussi appelée forge logicielle, forge de code ou forge Git, est le serveur qu’une équipe choisit comme lieu de référence pour ses dépôts, avec les services qui les entourent. Chaque clone complet d’un dépôt Git contient tout l’historique : en principe, aucune copie ne prévaut sur une autre. En pratique, une équipe a besoin d’un endroit où pousser le code, qui décide des droits de lecture et d’écriture et où se déroulent les discussions sur ce code. Cet endroit est la forge.

À la base, une forge conserve des dépôts nus, c’est-à-dire l’historique sans fichiers extraits dans un répertoire de travail, et utilise un ou plusieurs des protocoles de transfert de Git. Le livre Git en recense quatre : le protocole local pour un répertoire de la même machine, HTTP, SSH et le protocole propre à Git, dont le démon écoute sur le port 9418 sans aucune authentification. Tout le reste s’ajoute à cette base, dans des proportions très variables.

La documentation de Forgejo décrit le côté le plus complet de cette gamme : suivi des tickets, demandes d’intégration, wiki intégré, registre de paquets pour de nombreux écosystèmes de langages, Forgejo Actions pour exécuter des tâches, miroirs de dépôts, migration depuis d’autres services, protection des branches et des étiquettes, et webhooks. Soft Serve se situe à l’autre extrémité. Il se présente comme un « serveur Git auto-hébergeable pour la ligne de commande » : un seul binaire, `soft`, qui sert les dépôts par SSH, HTTP ou le protocole Git, contrôle les accès à l’aide de clés publiques SSH et prend en charge les collaborateurs, les dépôts privés et les webhooks. Ce sont deux forges ; elles diffèrent par la part du travail de l’équipe qu’elles cherchent à réunir.

Les forges ont aussi commencé à communiquer entre elles. ForgeFed est un protocole de fédération des plateformes d'hébergement et de collaboration sur des projets logiciels, conçu comme une extension d'ActivityPub. Il représente les dépôts, les gestionnaires de tickets, les gestionnaires de correctifs, les projets et les équipes comme des acteurs qui s'envoient des activités. Il vise un langage commun aux réseaux de forges et d'outils de gestion de projets : une personne pourrait ouvrir, depuis un serveur, un ticket concernant un dépôt d'un autre serveur, et un dépôt pourrait changer de serveur tandis que son ancienne copie resterait archivée en lecture seule et renverrait vers sa nouvelle adresse.

Utilisez le concept pour prendre une décision

Utilisez « Qu'est-ce qu'une forge ? » 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 forge est le lieu où les règles peuvent être imposées. Une règle sur la machine d'un développeur ne s'applique que tant que cette machine est correctement configurée ; un hook sur le serveur s'exécute à chaque push. Le hook `pre-receive` de Git reçoit sur l'entrée standard chaque référence poussée, sous la forme d'un ancien identifiant, d'un nouvel identifiant et d'un nom. S'il se termine avec un code non nul, aucune référence n'est mise à jour. La protection des branches, les revues obligatoires et les vérifications de signature des commits sur les grandes forges affinent la même idée : la copie partagée refuse ce que l'équipe a convenu de ne pas accepter.

Choisir une forge, c'est aussi choisir qui détient le code. Un service hébergé prend en charge les coûts d'exploitation et la disponibilité. En contrepartie, le code, la liste des accès et toutes les discussions résident sur le matériel d'un tiers et sont soumis à ses conditions. L'auto-hébergement inverse cet échange. Aucun choix n'est gratuit. La différence compte surtout pour une entreprise qui indique à ses clients où se trouvent leurs données.

Une forge détient aussi des données que Git ne conserve pas. Les commits et les étiquettes voyagent avec chaque clone ; les tickets, les fils de revue, les listes d'accès et les hooks côté serveur appartiennent à la seule forge. Cette séparation détermine ce qu'une sauvegarde doit couvrir et ce qu'une migration entre forges doit emporter.

À quoi cela ressemble en pratique

L'entreprise exploite sa propre forge. Trois projets présentés sur ce site montrent ce qu'il a fallu mettre en place.

Son code source était auparavant hébergé par un service tiers. Le projet de déploiement de sa propre forge Git part d'un constat : exploiter une forge implique d'en assurer l'authentification, le contrôle d'accès, le stockage, les sauvegardes et la façade publique. La forge est Soft Serve, un seul binaire supervisé par le système d'exploitation, choisi pour son approche centrée sur SSH et l'absence d'interface web d'administration. Elle est configurée comme une forge privée : les accès anonymes et sans clé sont refusés, et chaque dépôt déclaré est marqué privé. Son service SSH n'écoute que sur l'interface de bouclage, sur le port 23231 ; les accès extérieurs passent par un hôte de rebond. Un multiplexeur permettant de partager un port public entre HTTPS et Git sur SSH a été écrit, puis laissé désactivé : un point d'écoute inutilisé reste une surface d'attaque. Le résultat est énoncé simplement : l'entreprise héberge son propre code sur son propre matériel.

Le dépôt de ce site y réside lui aussi. Ses consignes désignent la forge auto-hébergée, Soft Serve sur SSH, comme dépôt distant canonique, qualifient l'ancienne copie hébergée de miroir historique et précisent qu'il n'y a pas de CI hébergée : les contrôles de qualité s'exécutent localement et dans les hooks Git.

Un hook côté serveur doit être testé dans l'environnement qui reçoit un véritable push Git. Il a donc été testé avec un vrai dépôt nu et un vrai push avant d'être considéré comme fiable. Le test crée un dépôt nu, installe le hook, clone, crée un commit, pousse et vérifie les deux chemins : acceptation et refus. Il a révélé deux défauts, invisibles à la lecture du fichier. Le second est instructif : lorsqu'il exécute un hook, Git exporte une variable désignant le répertoire du dépôt. La suite de tests lancée par le hook en héritait ; dix-neuf tests échouaient sur du code correct, jusqu'à ce que l'environnement soit nettoyé à la frontière entre les deux contextes. La page en précise aussi la limite : le test ne construit pas de push comportant des mises à jour forcées ou des suppressions de branches.

Tout cela partage un hôte à un seul cœur et 512 Mo de mémoire avec tous les autres services de l'entreprise accessibles au public : sept domaines TLS, un service onion, des serveurs Gemini et Gopher et le blocage des intrusions. L'automatisation des sauvegardes est implémentée séparément ; cela ne signifie pas que les sauvegardes sont actives. Une forge dotée d'un panneau web aurait ajouté une application résidente à ce budget, d'où l'exclusion de ces plateformes dès la conception. Sur cet hôte, la mémoire disponible est descendue jusqu'à environ 310 Mo sur 464 au cours d'une journée de mesures enregistrées.

Où les problèmes surviennent

Sur certaines forges, les visiteurs anonymes ont un accès en lecture par défaut. La documentation de Soft Serve indique qu'à l'installation, les connexions anonymes disposent de droits de lecture seule, contrôlés par `anon-access` et `allow-keyless`. Une forge privée règle le premier sur `no-access` et le second sur false. Il est utile de vérifier le résultat depuis une machine sans clé chargée. Le démon propre à Git est encore plus radical : le livre Git précise qu'un dépôt servi par ce démon est soit clonable par tout le monde, soit non servi.

Un hook qui appelle Git dans un autre répertoire agit sur le mauvais dépôt. Pendant un push, Git exécute les hooks de réception dans le dépôt nu et exporte `GIT_DIR` et des variables apparentées pour que les commandes Git du hook le retrouvent. Tous les processus qu'il lance en héritent, y compris des tests dans une copie de travail située ailleurs. Le manuel githooks demande de supprimer ces variables avant de travailler sur un autre dépôt. Un hook `pre-receive` minimal qui teste chaque commit poussé dans un répertoire temporaire le fait dans un sous-shell :

```
while read -r _ newrev _; do
  git cat-file -e "$newrev^{commit}" 2>/dev/null || continue
  tree=$(mktemp -d)
  git archive "$newrev" | tar -x -C "$tree"
  (
    unset $(git rev-parse --local-env-vars)
    cd "$tree" && ./run-tests
  ) || exit 1
done
```

Une forge restaurée retrouve son code, mais pas ses règles. N'importe quel clone peut repeupler les dépôts d'un serveur perdu, mais le livre Git rappelle que les hooks côté serveur et les paramètres similaires restent sur le serveur. Les tickets,

revues, clés et listes d'accès n'ont jamais fait partie des clones. Sauvegarder une forge implique de sauvegarder son répertoire de données et sa configuration, de restaurer les deux quelque part et de vérifier qu'un push que le hook doit refuser est effectivement refusé.

Chaque point d'écoute ouvert par une forge est un élément à défendre. L'exemple de configuration de Soft Serve place SSH sur le port 23231, HTTP sur 23232, le démon Git sur 9418 et un serveur de statistiques sur 23233. Une équipe qui pousse uniquement par SSH peut désactiver les autres ou les lier à l'interface de bouclage : les ports ouverts correspondent alors aux seuls usages réels.

Changer de forge emporte le code et laisse les discussions derrière. Un push miroir copie toutes les références sous `refs/`, branches et étiquettes comprises, et rien d'autre. Les tickets, fils de revue, labels et listes d'accès nécessitent l'outil de migration de la nouvelle forge, ou un protocole comme ForgeFed : celui-ci copie les commits, les étiquettes, les tickets, les demandes de fusion et les commentaires, puis laisse l'ancien dépôt archivé avec un lien vers le nouveau.

Pour aller plus loin

- Spécification ForgeFed — le protocole de fédération des forges, ses acteurs, ses activités et le processus de migration qui archive l'ancien dépôt.
- Soft Serve — la documentation du projet : un serveur Git en un seul binaire, centré sur SSH, avec notamment les paramètres `anon-access` et `allow-keyless`.
- Documentation Forgejo — les guides d'administration et d'utilisation d'une forge auto-hébergée complète, de l'installation aux sauvegardes, aux Actions et aux miroirs.
- githooks — tous les hooks exécutés par Git, dont `pre-receive`, et l'environnement dont hérite un hook de push.
- Les protocoles — les quatre moyens utilisés par Git pour transférer des données entre machines, et leurs coûts de configuration et de sécurité.
- ActivityPub — la recommandation du W3C étendue par ForgeFed.



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 qu'une forge ? » 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/forge/