Engineer

Форджа

спільний сервер для репозиторіїв коду та співпраці

Що це таке

Форджа, англійською forge, яку ще називають програмною форджею, форджею коду або git-форджею, — це сервер, який команда домовляється вважати домівкою своїх репозиторіїв, разом із сервісами, що працюють довкола них. Кожен повний клон репозиторію Git несе всю історію, тож у принципі жодна копія не стоїть вище за іншу. На практиці ж команді потрібне одне місце, куди пушити, одне місце, яке вирішує, хто може читати й писати, і одне місце, де точиться розмова про код. Це місце і є форджа.

Під капотом форджа зберігає голі (bare) репозиторії, тобто форму, що містить лише історію без робочої копії файлів, і розмовляє одним або кількома протоколами передавання Git. Книга Git називає чотири: локальний протокол для каталогу на тій самій машині, HTTP, SSH і власний протокол Git, тобто демон на порту 9418 без будь-якої автентифікації. Усе інше будується згори, і обсяг цієї надбудови дуже різниться.

Документація Forgejo описує повний край шкали: трекер задач, пул-реквести, вбудовану вікі, реєстр пакунків для багатьох мовних екосистем, Forgejo Actions для запуску завдань, дзеркала репозиторіїв, імпорт з інших сервісів, захист гілок і тегів, а також вебхуки. Soft Serve стоїть на протилежному краю. Проєкт описує себе як Git-сервер для командного рядка, який можна розгорнути самостійно: один бінарний файл `soft`, що віддає репозиторії через SSH, HTTP або протокол Git, керує доступом за публічними SSH-ключами й підтримує співавторів, приватні репозиторії та вебхуки. Обидва є форджами; різниця в тому, яку частину роботи команди кожен намагається вмістити.

Форджі також почали спілкуватися між собою. ForgeFed, протокол федерації для платформ хостингу програмних проєктів і спільної роботи над ними, побудований як розширення ActivityPub, моделює репозиторії, трекери задач, трекери патчів, проєкти й команди як акторів, що надсилають одне одному активності. Його мета — спільна мова для мереж форджів та інструментів керування проєктами, щоб людина на одному сервері могла відкрити задачу щодо репозиторію на іншому, а репозиторій міг переїхати між серверами, тоді як стара копія архівується в режимі лише для читання і вказує на нову адресу.

Застосуйте поняття до рішення

Застосуйте «Що таке форджа?» до конкретного рішення. Ми можемо пов'язати поняття із системою, яку ви створюєте або купуєте. Перевірено, випущено і з можливістю відкату, тож зміни доходять до клієнтів у вівторок удень, а не пізно в п'ятницю.

Переглянути наші послуги Зв'язатися з нами →

Перевірте припущення за терміном

Використайте пояснення та пов'язані роботи, щоб сформулювати запитання для проєкту. Ми можемо перевірити припущення разом до вибору інструмента. Автоматизувала розгортання GIS SaaS через CI/CD — розгортання, дані та звітність — усе без ручної роботи

Автоматизувала розгортання GIS SaaS через CI/CD Зв'язатися з нами →

Чому це важливо

Саме у форджі правила стають обов’язковими. Правило, що живе на машині розробника, діє лише доти, доки ту машину правильно налаштовано, а хук на сервері запускається під час кожного пушу. Хук Git `pre-receive` зчитує кожне надіслане посилання зі стандартного входу як старий ідентифікатор, новий ідентифікатор та ім’я, і якщо він завершується з ненульовим статусом, Git не оновлює жодного з посилань. Захист гілок, обов’язкові рецензії та вимога підписаних комітів у більших форджах уточнюють ту саму ідею: спільна копія відхиляє те, чого команда домовилася не приймати.

Вибір форджі — це також вибір того, хто тримає код. Хостинговий сервіс бере на себе витрати на роботу й доступність, а натомість код, список доступу та кожне обговорення лежать на чужому обладнанні за чужими правилами. Самостійний хостинг перевертає цей обмін. Ціну має кожен варіант, і різниця найважливіша для бізнесу, який розповідає клієнтам, де зберігаються їхні дані.

Форджа також зберігає дані, яких Git не має. Коміти й теги переходять із кожним клоном, а задачі, обговорення в рецензіях, списки доступу та серверні хуки належать лише форджі. Цей поділ визначає, що має

охоплювати резервна копія і що треба перенести під час переїзду між форджами.

Як це виглядає на практиці

Компанія тримає власну форджу, і три проєкти на цьому сайті показують, чого це вимагало.

Вихідний код компанії раніше зберігався в сторонньому хостинг-сервісі, а проєкт, у якому компанія розгорнула власну git-форджу, починається з думки, що мати форджу означає справді її тримати: автентифікація, контроль доступу, сховище, резервні копії й публічне обличчя. Форджа — це Soft Serve, один бінарний файл під наглядом операційної системи, обраний за те, що він працює насамперед через SSH і не має адміністративного вебінтерфейсу. Вона приватна за замовчуванням: анонімний доступ заборонено, доступ без ключа заборонено, а кожен задекларований репозиторій позначено як приватний. Її SSH-сервер слухає лише loopback-інтерфейс на порту 23231, а ззовні до нього дістаються через проміжний хост. Мультиплексор, який розводив би HTTPS і git через SSH на одному публічному порту, написано й залишено вимкненим, бо порт, яким ніхто не користується, теж є поверхнею атаки. Підсумок сформульовано просто: компанія тримає власний код на власному обладнанні.

Власний репозиторій цього сайту живе саме там. Його інструкції називають канонічним віддаленим репозиторієм самостійно розгорнуту форджу, Soft Serve через SSH, старішу хостингову копію називають застарілим дзеркалом і прямо кажуть, що хостингового CI немає: перевірка якості працює локально та в git-хуках.

Серверний хук треба перевіряти в середовищі, яке приймає справжній Git push, тож хук відрепетирували на справжньому bare-репозиторії та справжньому пуші, перш ніж йому довіряти. Репетиція створює bare-репозиторій, встановлює хук, клонує, робить коміт, пушить і перевіряє обидва сценарії: прийняття й відхилення. Вона виявила два дефекти, обидва невидимі у файлі. Повчальний другий: запускаючи хук, Git експортує змінну з каталогом репозиторію, набір тестів, який запустив хук, її успадкував, і дев’ятнадцять тестів ставали червоними на правильному коді, доки середовище не очистили на межі. Сторінка також називає своє обмеження: примусові оновлення та видалення гілок — це пуші, яких репетиція не відтворює.

Усе це ділить один одноядерний хост із 512 МБ пам’яті з усім іншим, що компанія обслуговує публічно: від семи доменів із TLS до onion-сервісу, сервера Gemini, сервера Gopher і блокування вторгнень. Автоматизацію резервного копіювання реалізовано окремо; це не те саме, що активні резервні копії. Форджа з вебпанеллю була б ще однією постійно запущеною програмою за цього бюджету, тому платформи з вебпанеллю відкинули ще на етапі проєктування; у вибірках, зафіксованих протягом доби на цьому хості, найнижче значення доступної пам’яті становило близько 310 МБ із 464.

Де виникають проблеми

Анонімні відвідувачі отримують доступ на читання за замовчуванням у деяких форджах. Документація Soft Serve каже, що з коробки він дає анонімним з’єднанням доступ лише на читання, а керують цим два параметри, `anon-access` і `allow-keyless`. Приватна форджа встановлює перший у `no-access`, а другий у false, і результат варто перевірити з машини без завантаженого ключа. Власний демон Git ще прямолінійніший: книга Git зазначає, що репозиторій, який він віддає, або доступний для клонування всім, або не віддається взагалі.

Хук, що викликає Git в іншому каталозі, влучає не в той репозиторій. Під час пушу Git запускає хуки отримання всередині bare-репозиторію та експортує `GIT_DIR` і пов’язані змінні, щоб команди Git у хуку його знаходили. Усе, що запускає хук, їх успадковує, зокрема й тести в робочій копії деінде. Посібник githooks радить очищати ці змінні, перш ніж працювати в чужому репозиторії. Мінімальний хук `pre-receive`, що перевіряє кожен надісланий коміт у тимчасовому каталозі, робить це в підоболонці:

```
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
```

Відновлена форджа повертається з кодом, але без своїх правил. Будь-який клон може знову наповнити репозиторії втраченого сервера, проте книга Git попереджає, що серверні хуки й подібні налаштування лишаються позаду, а

задач, рецензій, ключів і списків доступу в клонах ніколи й не було. Резервна копія форджі — це копія її каталогу даних і конфігурації, а потім відновлення обох деінде й перевірка, що пуш, який хук має відхилити, справді відхиляється.

Кожен порт, який відкриває форджа, доводиться захищати. Зразкова конфігурація Soft Serve має SSH на порту 23231, HTTP на 23232, демон Git на 9418 і сервер статистики на 23233. Команда, яка пушить лише через SSH, може вимкнути решту або прив'язати їх до loopback, і відкритих портів залишиться рівно стільки, скільки використовується.

Переїзд на іншу форджу переносить код і залишає розмову позаду. Дзеркальний пуш копіює кожне посилання в `refs/`, зокрема гілки й теги, і більше нічого. Задачам, обговоренням у рецензіях, міткам і спискам доступу потрібен інструмент імпорту нової форджі або протокол на кшталт ForgeFed, який копіює коміти, теги, задачі, мерж-реквести й коментарі та залишає старий репозиторій в архіві з посиланням на новий.

Що почитати далі

- Специфікація ForgeFed — протокол федерації для форджів з акторами, активностями та процедурою переїзду, яка архівує старий репозиторій.
- Soft Serve — власна документація проєкту про Git-сервер в одному бінарному файлі, що працює насамперед через SSH, зокрема про параметри `anon-access` і `allow-keyless`.
- Документація Forgejo — посібники для адміністраторів і користувачів повнофункціональної форджі для самостійного хостингу, від встановлення й резервних копій до Actions і дзеркал.
- githooks — усі хуки, які запускає Git, зокрема `pre-receive`, і середовище, яке успадковує хук пушу.
- The Protocols — чотири способи, якими Git переносить дані між машинами, і чого кожен вартує в налаштуванні та безпеці.
- ActivityPub — рекомендація W3C, яку розширює ForgeFed.

Словник

Обирайте за фактичними умовами

Опишіть шлях розгортання, експлуатаційні ризики та чергову команду. Ми можемо визначити зміну з перевірками, шляхом відновлення та чіткою відповідальністю.

Дивитися, скільки це коштує Зв'язатися з нами →

Зробіть термінологію практичною

Запитайте, як «Що таке форджа?» стосується вашої організації. Конкретного прикладу з поточної системи достатньо для початку розмови.

Переглянути наші послуги

welcome@engineer.company +45 50 10 61 84 Копенгаген, Данія

Візьміть цю сторінку із собою

Візитівка

Дбайливо зроблено з любов'ю — D.

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

Авторське право © 2025 – дотепер · Engineer ApS · Усі права захищено.

Надруковано з https://engineer.company/uk/glossary/forge/