Engineer

GitLab

платформа сховищ, яку можна запустити самому

Що це таке

GitLab — це єдиний застосунок, побудований довкола git-репозиторіїв: він зберігає код і додає до нього merge request для рев'ю, issues для планування та вбудовану систему CI/CD, яка збирає, тестує й розгортає з того самого місця. Від інших його відрізняє вибір того, де він працює. Той самий продукт доступний як GitLab.com, сервіс, який GitLab веде сам; як GitLab Self-Managed, який встановлюють на власні сервери; і як GitLab Dedicated, окремий екземпляр для одного клієнта, який GitLab розміщує й обслуговує в регіоні AWS, обраному цим клієнтом.

Під усім цим лежить звичайний git. Кожен проєкт — це git-репозиторій, який можна клонувати, надсилати в нього зміни й отримувати їх звичними командами, а історія в клоні повна. Усе, що GitLab додає зверху, від обговорень під час рев'ю та issues до результатів конвеєрів і прав доступу, зберігає сам GitLab поза репозиторієм, і клон нічого з цього не несе.

Merge request — це назва, яку GitLab дає запропонованій зміні, що чекає на злиття; інші платформи називають ту саму ідею pull request. Документація описує його як центральне місце, де переглядають код, ведуть обговорення і відстежують зміну. На одній сторінці видно опис, зміни в коді з коментарями до окремих рядків, стан конвеєрів CI/CD, звіти про те, чи можна зміну злити, і список комітів. Merge request можна пов'язати з issue, і тоді він автоматично закриє це issue під час злиття. Усталено основну гілку проєкту, зазвичай main, захищено, і зливати в неї можуть лише люди з роллю Maintainer або вищою.

Систему CI/CD налаштовують у файлі `.gitlab-ci.yml` у корені проєкту. Він визначає етапи, завдання та скрипти, з яких складається конвеєр: етапи задають порядок, наприклад збирання, потім тести, потім розгортання, а завдання — це роботи, які виконуються на кожному етапі. Завдання виконує

GitLab Runner, окрема програма, написана мовою Go й поширювана як один бінарний файл. Runner під’єднується до екземпляра GitLab, чекає на завдання, виконує кожне просто на машині, у контейнері Docker або на віддаленому сервері через SSH і повідомляє результат назад.

Застосуйте поняття до рішення

Застосуйте «Що таке GitLab?» 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

Більшість того, що команда розробників робить між написанням коду та його випуском, може відбуватися в одному екземплярі GitLab: код, рев’ю, конвеєр, що його тестує, issue, у якому про нього попросили, і запис про те, хто що схвалив. Один застосунок означає один набір облікових записів і прав, одне місце для пошуку і зв’язки, які тримаються між issue, merge request, що його закрив, і конвеєром, що зміну перевірив.

Можливість вести GitLab самостійно — причина, через яку багато організацій узагалі його обирають. Коли код має лишатися на власній інфраструктурі компанії через договір, регулятора чи внутрішні правила, хостинговий сервіс відпадає, а GitLab можна встановити всередині тієї мережі, яка має цей код берегти. Документація вважає це повноцінним сценарієм: вона публікує еталонні архітектури для встановлень на 1 000–50 000 користувачів, поруч із пакетом для Linux і Helm-чартом для Kubernetes.

Цей контроль має ціну, і платять її роботою з експлуатації, а не лише ліцензіями. Вести GitLab самостійно означає підібрати для нього сервер, підтримувати в доброму стані базу PostgreSQL і сховище Redis або Valkey, виділити машини для runner-ів, робити резервні копії, з яких справді можна відновитися, і проходити оновлення в тому порядку, якого вимагає проєкт. Хостингові варіанти передають цю роботу GitLab; самостійний лишає її вам.

Як це виглядає на практиці

Робота компанії стикалася з GitLab з обох боків.

Протягом кількох років роботи з даними практики CI/CD на основі GitHub і GitLab, де саму автоматизацію забезпечували Bash і Python, замінили ручні кроки і в аналізі даних, і в розробці. Шлях роботи від розробки до постачання стандартизували, щоб він щоразу був однаковим, а не вигадувався заново для кожного проєкту, а аналітичний напрям, який довго вважали окремим і більш ручним світом, увійшов у той самий конвеєр, що й розробка. Сторінка називає приблизно 4 000 заощаджених годин і більш узгоджені результати постачання, бо роботу більше не виконували щоразу трохи по-різному.

Власний код компанії показує інший бік самостійного хостингу. Після відмови від стороннього хостингового сервісу його перенесли на форджу, яку компанія обслуговує сама: це один бінарний файл, орієнтований насамперед на SSH, без адміністративного вебінтерфейсу, обрану за принципом: що менше інтерфейсу, то менше треба захищати. Ця форджа ділить хост публічних сервісів із 512 МБ пам’яті, з яких близько 464 МБ придатні до використання, з вебсервером для семи доменів, onion-сервісом Tor, сервером Gemini, сервером Gopher і блокуванням спроб вторгнення. Автоматизацію резервного копіювання реалізовано окремо; реалізація автоматизації не означає наявності активних резервних копій.

Жодна з цих двох сторінок не згадує GitLab, і подальше порівняння — власна арифметика цієї статті. GitLab називає 8 ГБ пам’яті мінімумом, з яким може працювати встановлення на одному вузлі; увесь той сервер має шістнадцяту частину цього обсягу. Вести GitLab самостійно означає мати сервер, розрахований на GitLab, а невелика форджа з однією метою — інший шлях тримати код на власних машинах.

Мінімальний файл конвеєра показує форму налаштувань CI/CD: два етапи, що виконуються по черзі, і по одному завданню в кожному:

```
stages:
    - build
    - test

build-site:
```

```
    stage: build
    script:
        - make build

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

Де виникають проблеми

- **Малий сервер сповільнює весь екземпляр.** Документація GitLab називає 16 ГБ пам'яті базовим рівнем для встановлення на одному вузлі, а 8 ГБ — найменшим обсягом, з яким воно може працювати в середовищі з обмеженою пам'яттю. Вона також попереджає, що swap може суттєво знизити продуктивність під навантаженням, і просить вимикати swap, де це можливо, або дати стільки пам'яті, щоб GitLab ніколи ним не користувався.
- **Резервна копія відновлюється в екземпляр, що не може прочитати власні дані.** Команда резервного копіювання не включає каталог конфігурації `/etc/gitlab`, а з ним і `gitlab-secrets.json` — ключі, що розшифровують зашифровані значення в базі даних. Без цього файлу ці значення у відновленій базі лишаються замкненими. Документація просить робити резервну копію конфігурації окремо, щонайменше `gitlab-secrets.json` і `gitlab.rb`.
- **Оновлення перескакує через обов'язкову зупинку.** Шляхи оновлення містять обов'язкові зупинки, тобто версії, які треба встановити перед будь-якою пізнішою, і починаючи з GitLab 17.5 вони припадають на x.2, x.5, x.8 і x.11. Фонові міграції кожної зупинки мають завершитися до початку наступного оновлення, тож екземпляр, що відстав на кілька версій, наздоганяє їх по одній зупинці за раз.
- **Кероване сховище даних виявляється непідтримуваним.** PostgreSQL — єдина підтримувана база даних. Redis або Valkey зберігає сесії та фонові задачі й має бути окремим самостійним екземпляром: Redis Cluster і serverless-варіанти Redis і Valkey не підтримуються. Керований хмарний сервіс дозволено, якщо він саме самостійного типу.
- **Конвеєри чекають на машини, які ніхто не запланував.** GitLab.com надає готові runner-и, а на самостійному екземплярі адміністратор сам забезпечує й обслуговує інфраструктуру, де виконуються завдання CI/CD, зокрема стежить, щоб runner-и мали достатньо потужності для навантаження організації. Завантажена команда з одним малим runner-ом стоїть за ним у черзі.

Що почитати далі

- GitLab Docs — повна документація про використання, встановлення та адміністрування GitLab.
- Installation requirements — пам’ять, процесор, сховище та підтримувані версії PostgreSQL і Redis.
- Back up GitLab — що містить резервна копія, чого в ній немає і як зберігати секрети поруч із нею.
- Upgrade paths — обов’язкові зупинки оновлення і чому фонові міграції мають завершитися між ними.
- Get started with GitLab CI/CD — файл `.gitlab-ci.yml`, етапи, завдання й конвеєри.
- GitLab Runner — програма, що виконує завдання, і способи, якими вона може їх виконувати.

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 GitLab?» 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/gitlab/