Engineer

GitHub

сервіс Microsoft для сховищ git

Що це таке

GitHub — це хостинговий сервіс, що належить Microsoft: він зберігає git-репозиторії на власних серверах і будує навколо них місце для роботи. Тут є вебперегляд коду та його історії, pull request для пропонування й рецензування змін, issues для обліку задач і GitHub Actions, що автоматично запускає завдання, щойно в репозиторії щось відбувається.

Під усім цим лежить звичайний git. Репозиторій, клонований із GitHub, — це звичайний git-репозиторій з усією історією, і робота переходить між ноутбуком і GitHub звичними push і fetch. Те, що додає GitHub, живе на його боці з'єднання: обговорення під час рев'ю, issues, правила для гілок, налаштування автоматизації та список того, кому що дозволено, зберігаються в системах GitHub, а не в репозиторії, який ви клонували.

Pull request — це назва, яку GitHub дає пропозиції злити зміни з однієї гілки в іншу. Він показує різницю рядок за рядком, збирає коментарі й схвалення та відображає результат кожної автоматичної перевірки, запущеної для запропонованого коду. Pull request, відкритий як чернетка, позначено незавершеним, і злити його можна лише після того, як хтось позначить його готовим до рев'ю.

GitHub Actions — це автоматизація, вбудована в сервіс. Робочий процес, workflow, — це YAML-файл у каталозі `.github/workflows` репозиторію. Він називає події, які його запускають, як-от push, новий pull request або розклад, і перелічує завдання, складені з кроків. Кожен крок виконує команду оболонки або action — запаковану частину автоматизації, яку можна використовувати в різних репозиторіях. Завдання виконуються на runner-

ах: це віртуальні машини, які GitHub щоразу запускає наново з Ubuntu Linux, Windows або macOS, або власні машини, зареєстровані як self-hosted runners.

Ще два сервіси стежать за тим, що потрапляє в репозиторій. Dependabot читає файл `dependabot.yml` і відкриває pull request, які оновлюють залежності, а окремо відкриває pull request для залежностей з відомими вразливостями. Push protection перевіряє вхідні push на секрети, як-от токени доступу, і блокує push, що містить такий секрет, ще до того, як він дійде до репозиторію.

Застосуйте поняття до рішення

Застосуйте «Що таке GitHub?» 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

Для багатьох команд GitHub — це місце, де відбувається робота, а не лише місце, де вона зберігається. Код, обговорення кожної зміни, запис про те, хто її схвалив, і результат кожної перевірки лежать поруч, тож на питання, навіщо існує певний рядок, зазвичай можна відповісти з однієї сторінки.

Автоматизація важить не менше за хостинг. Коли Actions стоїть поруч із кодом, pull request може нести власні докази: тести запустилися й пройшли, збирання вдалося, сканери повідомили, що побачили. Захищені гілки перетворюють ці докази на правило. Гілку можна налаштувати так, щоб вона приймала зміни лише через pull request, лише після певної кількості схвальних рев'ю і лише після того, як пройдуть названі перевірки статусу. Тоді головна лінія рухається тільки через переглянуті й перевірені зміни.

Оскільки GitHub стоїть там, де код входить у проєкт, саме тут живуть засоби контролю ланцюжка постачання: які сторонні actions може запускати workflow і в якій точній версії, як залежності лишаються свіжими, чи зупиняється секрет на етапі push. Ці засоби здебільшого вмикають і

налаштовують окремо для кожного репозиторію чи організації. Тому значна частина реальної безпеки проєкту на GitHub живе в налаштуваннях, яких під час рев'ю коду ніхто не бачить.

У цьому й полягає компроміс хостингового сервісу. Ви отримуєте повне, доглянуте середовище без власних серверів, а натомість правила, що захищають ваш код, налаштовуються в чужому інтерфейсі, з усталеними значеннями, обраними для загального випадку, а не для вашого.

Як це виглядає на практиці

Три сторінки з роботи компанії показують GitHub у різних ролях.

Конвеєр постачання на GitHub Actions ділить роботу на окремі workflow: один для брами якості коду, один для тестів і по одному на кожне середовище для розгортання, а поруч із ними працюють CodeQL, перевірка залежностей і крок, що створює перелік складників програмного забезпечення. Шлях веде від коміту до сервісу, що працює в Azure, а просування переносить збірку між середовищами визначеним маршрутом, а не вручну. Фронтенд постачається на distroless-образі без оболонки й без менеджера пакетів, API на Go — на тонкій базі Alpine.

Безперервна програма безпеки користується засобами контролю, що оточують код на GitHub. Кожну сторонню GitHub Action закріплено за хешем коміту, а не за тегом, тож ніхто не може пересунути тег і змінити те, що запускає workflow. Dependabot стежить за 6 екосистемами, Gitleaks сканує історію на облікові дані, а кожна збірка створює перелік складників програмного забезпечення за допомогою Anchore і Syft. Сторінка називає й справжню ціну: обсяг знахідок. Сканер, який повідомляє про все, привчає людей його ігнорувати, тож щоб сигнал лишався корисним, потрібне постійне сортування.

Розгортання GIS-застосунку, що постачається як SaaS перенесло цикл тестування, збирання й розгортання в робочі процеси GitHub Actions, тож випуск перестав залежати від того, чи хтось пам'ятає всі кроки. Регулярна обробка даних і звіти перейшли в заплановані завдання на Python, Bash і SQL, а конфігурацію та секрети стандартизували, щоб кожне середовище поводилося однаково.

Ось як закріплення виглядає у workflow. Довгий шістнадцятковий рядок — повний ідентифікатор коміту того випуску, який переглянули, а коментар тримає поруч зрозумілу людині версію:

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

Де виникають проблеми

- **Action, закріплена за тегом, змінюється без попередження.** Тег на кшталт `v4` може пересунути або видалити той, хто керує репозиторієм action, тож код, який workflow запустить завтра, може відрізнятися від коду, переглянутого сьогодні. GitHub радить закріплювати повний SHA коміту: підробити його нападник зможе, лише створивши колізію SHA-1 для дійсного об'єкта git.
- **Токен workflow має більше влади, ніж потрібно завданню.** Кожен запуск отримує `GITHUB_TOKEN`, і його дозволи визначають, що зможе змінити скомпрометований крок. GitHub радить зробити усталеним доступ лише на читання вмісту репозиторію і давати доступ на запис тільки тим завданням, яким він справді потрібен.
- **Неперевірений текст виконується як команда.** Workflow, що вставляє назву pull request просто в інлайн-скрипт, дає тому, хто цю назву написав, змогу визначити частину скрипту. Якщо передати значення через проміжну змінну середовища, воно лишається даними. Тригери `pull_request_target` і `workflow_run` запускають привілейовані workflow, і GitHub попереджає: checkout неперевіреного pull request у такому workflow відкриває репозиторій для компрометації.
- **Адміністратори проходять повз правила гілок.** Усталено обмеження правила захисту гілки не діють для людей з правами адміністратора репозиторію. Увімкнений параметр «Do not allow bypassing the above settings» поширює правила й на них.
- **Секрет прослизає крізь push protection.** Форма, увімкнена усталено, захищає власні push кожного користувача до публічних репозиторіїв. Форма, що захищає репозиторій від усіх учасників, потребує GitHub Secret Protection і лишається вимкненою, доки хтось її не ввімкне. Навіть тоді будь-хто з доступом на запис може обійти блокування, вказавши причину; це створює сповіщення і запис у журналі аудиту. Щойно облікові дані надіслано, вони є в кожному клоні, тож їх треба замінити, а не покладатися на переписування історії.

Що почитати далі

- GitHub Docs — повна документація про репозиторії, pull request, Actions і засоби безпеки.
- Understanding GitHub Actions — workflow, події, завдання, кроки й runner-и та те, як вони пов’язані.
- Secure use reference — закріплення actions, дозволи токена й ін’єкції в скрипти, як їх описує сам GitHub.
- About protected branches — обов’язкові рев’ю, обов’язкові перевірки статусу і хто може їх обійти.
- Dependabot version updates — як файл `dependabot.yml` керує pull request з оновленнями.
- Push protection — дві форми push protection, що блокує кожна з них і як фіксується обхід.

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 GitHub?» 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/github/