Engineer

Ворота якості

набір перевірок, які код має пройти, щоб рушити далі

Що це таке

Ворота якості — це набір умов, які зміна має виконати, перш ніж перейти на наступний етап, з одним результатом наприкінці: пройшла або впала. Етапом може бути коміт, злиття в основну гілку, реліз чи розгортання. Умовою може бути все, що машина здатна вирішити сама: чистий прогін лінтера, тести, що проходять, показник покриття, збірка, яка завершується.

Головне в назві — слово «ворота». Перевірка повідомляє, а ворота вирішують, і це рішення виконує те, що керує наступним кроком. Лінтер, який виводить у лог сорок попереджень, — це перевірка. Той самий лінтер, підключений так, що кнопка злиття лишається неактивною, доки його вивід не стане чистим, — це вже частина воріт.

SonarQube Cloud, де цей термін є назвою окремої функції, описує ворота якості як набір умов, за якими код оцінюють під час аналізу. Кожна умова поєднує метрику, оператор порівняння і порогове значення помилки, тож «reliability issues greater than 0» — це умова, і якщо справджується хоч одна з умов, перевірка якості провалюється. Умова стосується або нового коду, або коду загалом.

Платформи для хостингу коду вбудовують ту саму ідею у процес злиття:

- **GitHub** має налаштування захисту гілки «Require status checks before merging». Обов’язкова перевірка повинна мати статус success, skipped або neutral, перш ніж зміни потраплять у захищену гілку.
- **GitLab** має перевірку злиття «Pipelines must succeed». Коли її ввімкнено, merge request неможливо злити, поки його пайплайн падає, а також тоді, коли пайплайна немає зовсім.

- **На власній машині розробника** pre-commit встановлює перевірки як Git-хуки, що спрацьовують під час коміту або push, а виконавець завдань на кшталт Task дає всьому наборові одне ім’я.

Застосуйте поняття до рішення

Застосуйте «Що таке ворота якості?» 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

Стандарт, записаний лише в документі, розмивається поступово, щоразу на один виняток. Стандарт, що валить збірку, виконують усі, зокрема й той, хто його написав, навіть того дня, коли дуже хочеться його пропустити. Ворота роблять узгоджені правила виконуваними, тож їх дотримання більше не залежить від пам’яті й настрою.

Вони також змінюють те, навіщо потрібне рев’ю коду людьми. Коли машина вже спіймала форматування, невикористані імпорти, биті посилання й секрети, що потрапили в код, рецензент може зосередитися на задумі й дизайні, а їх машина оцінити не здатна.

Ідея нового коду дає змогу запровадити ворота на старій кодовій базі. Sonar way, стандартні ворота в SonarQube Cloud, зосереджуються на новому коді, а не на виправленні старого, з умовами на кшталт покриття тестами щонайменше 80,0% і дублювання не більше 3,0% у новому коді. Для pull request використовуються лише умови щодо нового коду. Тож проєкт із тисячами наявних зауважень може ввімкнути перевірку одразу, не розплачуючись спершу за накопичене.

Швидкість вирішує, чи лишать люди ворота ввімкненими. Task запускає залежності завдання паралельно, тож незалежні перевірки працюють одночасно, а не по черзі. Допомагає й поділ за вартістю: дешева половина — на кожен коміт, повний набір — перед тим, як код залишить машину.

У цього сайту є власні ворота якості. Це `task check`, і запускається він на тій самій машині, де пишеться сайт, а не в хмарному сервісі збірки. Кожна діагностика вважається помилкою, генератор сайту працює з `--panicOnWarning`, тож навіть повідомлення про застарілу функцію зупиняє збірку, а локальні вимкнення перевірок у коді на кшталт `# noqa` чи `stylelint-disable` заборонено, тому помилку усувають у її причині, а не приглушують. Хук pre-commit запускає перевірки окремих файлів під час кожного коміту, а хук pre-push — повний `task check`, що триває хвилини, а не секунди. Зелений прогін із чистого дерева ставить на це дерево позначку, і скрипт розгортання відмовляється працювати з деревом без неї. Обійти це можна лише змінною середовища, яка виводить гучне попередження, тож той, хто в надзвичайній ситуації пропускає перевірку, мусить сказати про це вголос.

Як це виглядає на практиці

Сторінка подяк сайту перелічує інструменти всередині цих воріт: vnu для HTML, Stylelint, axe-core на одній сторінці від кожного шаблону, Lighthouse, lychee для посилань, Prettier, markdownlint, ESLint, Ruff, basedpyright, yamllint, ShellCheck, codespell і gitleaks, а Task запускає все це по одній команді на кожне завдання. Сторінка показує й те, що лишається поза цими воротами: з перевірок у браузері, якими керує Puppeteer, це вихід за межі вікна й візуальна регресія, і їх людина запускає на вимогу.

У проєкті, який веде одна людина, рецензента немає взагалі, і саме з цієї ситуації починається одна сторінка портфоліо. Стандарти мали стати виконуваними, щоб порушення блокувало коміт, а не чекало, поки його хтось помітить. На описаному в документації етапі впровадження в серпні 2026 року відповіддю став набір перевірок із 22 лінтерів без рівня попереджень, а також перевірки, специфічні для проєкту, щодо яких готові інструменти не мають думки: наприклад, що розгорнуте дерево не містить приватних шляхів чи приватних імен хостів і що текст дотримувався правил тону всіма трьома мовами. Збій, який існував і раніше, виправляють у тому самому проході. Ціну сторінка називає прямо: комітити повільно, а ці перевірки — єдиний рецензент у проєкті.

Перевірки можуть розходитися й між собою. Інша сторінка описує проєкт із хуком коміту й повним гейтом якості, документом, де стверджувалося, що хук запускає гейт, і двома переліками перевірок, які вели вручну у двох файлах. Тест, що визначає, які перевірки обидва переліки насправді викликають, показав п’ять перевірок, що існували лише в гейті, і це були

саме ті, що читають текст. Кожна перевірка, що захищала код, запускалася автоматично; кожна, що захищала текст, — лише тоді, коли хтось про неї згадував. П'ять перевірок перенесли в хук, дві з них пришвидшили, щоб вони вкладалися в його ліміт часу, а тест порівняння залишили, щоб виявляти розбіжності між переліками.

Де виникають проблеми

Більшість воріт якості дають збій так, що пропускають зайве й далі світяться зеленим.

Попередження накопичуються за зеленим результатом. Ворота, що падають лише на помилках, дають попередженням накопичуватися, доки лог не стане задовгим для читання, і тоді справжня проблема приходить як попередження номер двісті один. Приберіть рівень попереджень: правило або варте того, щоб на ньому падати, або вимкнене.

Пропущена перевірка зараховується як пройдена. На GitHub статус `skipped` задовольняє обов'язкову перевірку, а завдання, пропущене через власну умову `if:`, повідомляє статус `success`. Завдання, що залежить від невдалого завдання, також пропускається й може не заблокувати злиття. GitHub радить використовувати `always()` разом із `needs`: зробіть обов'язковим одне завдання, яке запускається завжди й проходить лише тоді, коли всі його залежності завершилися успішно.

```
# .github/workflows/check.yml (уривок)
jobs:
    gate:
        if: always()
        needs: [lint, test, build]
        runs-on: ubuntu-latest
        steps:
            - run: |
                  for result in ${{ join(needs.*.result, ' ') }}; do
                      test "$result" = success || exit 1
                  done
```

Буває й протилежна пастка: обов'язковий робочий процес, пропущений через фільтр шляхів або гілок, лишає свої перевірки в стані `pending` і блокує кожне злиття. Не робіть обов'язковими робочі процеси, які можна пропустити.

Два переліки розходяться. Хук запускає один набір, сервер — інший, а документ запевняє, що вони однакові. Дайте перевіркам одне визначення і викликайте його з усіх місць або тримайте тест, який порівнює переліки й падає, коли вони різняться.

Ворота стають повільними, і їх починають обходити. Прапорець, що пропускає хуки, завжди під рукою, а на GitHub захист гілки за замовчуванням не поширюється на адміністраторів репозиторію; це змінює налаштування «Do not allow bypassing the above settings». Виміряйте, куди йде час, тримайте частину для коміту невеликою, а справді повільну перевірку винесіть у задокументований ручний крок.

Перевірка проходить, хоча нічого не перевірила. Лінтер, націлений на glob, під який не потрапляє жоден файл, або шаблон ignore, що поглинає цілу теку, завершується чисто, а чисте завершення виглядає так само, як чистий код. Перевіряйте самі ворота: дайте кожній перевірці файл, про який відомо, що він хибний, і чекайте, що вона впаде.

Що почитати далі

- Introduction to quality gates — умови, стандартні ворота Sonar way і зосередженість на новому коді.
- About protected branches — обов’язкові перевірки статусу і те, хто може їх обійти.
- Troubleshooting required status checks — які статуси вважаються успішними і як поводяться пропущені робочі процеси та завдання.
- Auto-merge — перевірка GitLab «Pipelines must succeed» і те, як вона трактує пропущені пайплайни.
- pre-commit — перевірки, оголошені один раз і встановлені як Git-хуки для етапів коміту й push.
- Task dependencies — як Task запускає залежності завдання паралельно, а команди — по черзі.

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 ворота якості?» 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/quality-gate/