Engineer

Синьо-зелене розгортання

◐ перемикання трафіку між двома однаковими середовищами

Що це таке

Синьо-зелене розгортання — це спосіб випускати програмне забезпечення, за якого є два продакшн-середовища, зроблені якомога однаковими. Одне з них, назвімо його синім, обслуговує всі запити. Нову версію встановлюють у друге, зелене, і перевіряють там, поки жоден клієнт до нього не потрапляє. Коли версія готова, змінюють те, що стоїть перед обома середовищами, і всі нові запити йдуть у зелене. Синє тим часом працює далі недоторканим, і якщо зелене поводиться погано, та сама зміна у зворотний бік повертає в роботу синє. Англійською це blue-green deployment, і в українських командах часто так і кажуть: blue-green.

Кольори нічого не означають, і в цьому весь сенс. Назви на кшталт «старе» і «нове» чи «основне» і «резервне» міняють значення після кожного релізу, а два кольори просто чергуються: після наступного перемикання живим стає зелене, а вільним — синє. Мартін Фаулер описав цей прийом у своєму блозі-вікі bliki 2010 року й приписує назву Деніелу Терхорсту-Норту та Джезу Гамблу, хоча сам визнає, що пам'ятає її походження туманно. Інструменти пишуть назву кожен по-своєму: Argo Rollouts, контролер Kubernetes, що додає такі стратегії розгортання, у конфігурації називає її `blueGreen`.

Перемикачем може бути будь-яка одна точка, яка вирішує, куди йдуть запити. У Фаулера це маршрутизатор. Так само це може бути ціль балансувальника навантаження, upstream у зворотному проксі або, у Kubernetes, Service, чий селектор міток обирає поди, що отримують трафік. Два середовища не мусять бути двома дата-центрами: Фаулер згадує окреме обладнання, окремі віртуальні машини або дві зони на одній системі з власними IP-адресами як цілком придатні варіанти.

Відмінність найкраще видно поруч із сусідніми стратегіями. Поступове оновлення (rolling update) замінює екземпляри по кілька за раз, тож стара й нова версії відповідають на запити поруч, доки не зникне останній старий екземпляр, а відкат означає ще одне поступове оновлення у зворотному напрямку. Канарковий реліз спершу спрямовує на нову версію невелику частку трафіку й розширює її крок за кроком. Синьо-зелене розгортання переносить усе одразу, але тримає повну робочу копію попередньої версії, тож шлях назад — це один крок, такий самий, як шлях уперед.

Застосуйте поняття до рішення

Застосуйте «Що таке синьо-зелене розгортання?» 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

Реліз стає коротким оборотним кроком замість довгої дороги в один бік. Повільні частини — копіювання файлів, запуск процесів, прогрівання кешів — відбуваються в зеленому середовищі, поки синє ще обслуговує клієнтів, тож ніхто на них не чекає. Саме перемикання швидке, і скасувати його так само швидко: нічого не треба перезбирати чи перевстановлювати, бо попередня версія ніколи не зупинялася. Це змінює те, як команда ставиться до релізу звичайного робочого дня, і те, наскільки швидко вона діє, коли щось виглядає не так.

Зелене середовище можна перевірити в умовах, дуже близьких до продакшну, ще до того, як його побачить клієнт: та сама мережа, та сама конфігурація, та сама база даних. Smoke-тест проти зеленого доводить набагато більше, ніж той самий тест у staging, що відрізняється від продакшну десятком дрібниць. Argo Rollouts оформлює це як окремий preview-Service, який спрямовує трафік на нову версію до просування, щоб її можна було перевірити без продакшн-трафіку.

Вільне середовище має ще одну роботу. Коли реліз устоявся, бік, що щойно затих, стає місцем для підготовки наступного релізу, а доти слугує теплим резервом. Фаулер зауважує, що той самий механізм дає змогу перевіряти

план аварійного відновлення на кожному релізі, адже кожне перемикання — це перемикання на резерв, за яким хтось уважно стежить.

Ціна очевидна. Два продакшн-середовища потребують подвійної потужності, принаймні на час релізу. І хоча прикладний рівень легко продублювати, дані — ні: синє й зелене майже завжди ділять одну базу даних, тож кожен реліз має лишатися читабельним для обох версій. Команда, яка не може цього гарантувати, отримує швидке перемикання вперед і зламане перемикання назад.

Як це виглядає на практиці

У Kubernetes найменша робоча версія — це два Deployment і один Service. Deployment `web-blue` і `web-green` запускають той самий застосунок із мітками `app: web` та `slot: blue` або `slot: green`. Service, до якого звертаються клієнти, обирає один слот:

```
apiVersion: v1
kind: Service
metadata:
  name: web
spec:
  selector:
    app: web
    slot: blue
  ports:
    - port: 80
      targetPort: 8080
```

Реліз кладе новий образ на вільний бік, чекає на нього, перевіряє через окремий Service, що обирає лише `slot: green`, а тоді перемикає робочий:

```
kubectl set image deployment/web-green web=registry.example.com/web:1.5.0
kubectl rollout status deployment/web-green --timeout=5m
# тут smoke-тести проти preview-Service для web-green
kubectl patch service web -p '{"spec":{"selector":{"slot":"green"}}}'
# шлях назад — та сама команда з іншим кольором
kubectl patch service web -p '{"spec":{"selector":{"slot":"blue"}}}'
```

`kubectl patch` за замовчуванням застосовує strategic merge patch, тож у селекторі змінюється лише ключ `slot`, а `app: web` лишається на місці. Контролер сервісу постійно шукає поди, що відповідають селектору, і оновлює кінцеві точки сервісу, коли їхній набір змінюється, тож нові з'єднання потрапляють у зелене, а синє стоїть напоготові з попереднім образом.

Argo Rollouts автоматизує той самий танець одним об'єктом `Rollout`: `activeService`, який він під час просування спрямовує на нові поди, і необов'язковий `previewService`, що бачить їх заздалегідь. З `autoPromotionEnabled: false` розгортання зупиняється перед перемиканням, доки хтось не виконає `kubectl argo rollouts promote web`.

Послуга DevOps та автоматизації CI/CD обіцяє зміни, які перевірено, випущено і які можна відкотити, а синьо-зелене розгортання — один зі способів зробити останнє дешевим. Воно спирається на дві звички, які розгортання платформи в Azure показують у дії, хоч і без другого продакшн-слота: шаблони Bicep тримають цільову інфраструктуру однаковою в усіх середовищах, а той самий зібраний артефакт просувають від development через staging до production, і він підхоплює потрібні налаштування там, куди потрапляє. Такий маршрут веде реліз крізь середовища одне за одним. Синьо-зелена схема ставить у кінці цього шляху близнюка, і саме ці дві звички — однакова інфраструктура та один незмінний артефакт — дають команді підстави вірити, що близнюк справді однаковий.

Де виникають проблеми

Зміна схеми перетворює шлях назад на глухий кут. Зелене стартує, запускає міграцію, що перейменовує стовпець, і без проблем обслуговує трафік. Через десять хвилин збій повертає трафік на синє, яке досі звертається до старої назви стовпця й падає на кожному запиті. Відкат спланували для застосунку, але не для даних. Порада Фаулера з 2015 року — розвести ці дві речі: спершу випустити зміну бази даних, з якою працюють обидві версії, потім застосунок, а те, що потрібне лише старій версії, прибрати, коли нова вже устоялася.

Стан, записаний у зеленому, губиться після перемикання назад. Усе, що зелене тримає лише в себе, наприклад файли, завантажені на локальний диск, сесії в пам'яті чи власний кеш відкладених завдань, відсутнє в синьому, коли трафік повертається туди. Користувачі опиняються розлогіненими або

не знаходять щойно збереженого. Тримайте спільний стан у сховищах, якими користуються обидва кольори. Якщо зеленому все ж потрібні власні дані, Фаулер пропонує подавати транзакції в обидва середовища або запускати зелене лише на читання й вмикати запис, коли воно себе виправдало.

Шлях назад згортають раніше, ніж проявляється збій. Argo Rollouts зменшує старий ReplicaSet після перемикання, а `scaleDownDelaySeconds` за замовчуванням дорівнює 30. Збій, що з'являється за чверть години, нічне завдання чи повільний витік пам'яті не знаходить синього, на яке можна повернутися. Встановіть затримку, що покриває час, за який проблеми зазвичай виходять назовні, і прийміть, що платою за це є потужності:

```
strategy:
  blueGreen:
    activeService: web
    previewService: web-preview
    autoPromotionEnabled: false
    scaleDownDelaySeconds: 3600
```

Просування йде за готовністю, а не за коректністю. `autoPromotionEnabled` за замовчуванням дорівнює true, тож Argo Rollouts перемикає, щойно нові поди готові. Readiness probe каже, що процес відповідає, а не що оформлення замовлення працює. Робіть паузу перед просуванням, як у блоці вище, і перевіряйте через preview-Service, або додайте `prePromotionAnalysis`, що перериває розгортання в разі невдачі, і `postPromotionAnalysis`, що повертає попередню версію, якщо аналіз провалюється вже після перемикання.

Вільний бік поступово розходиться з робочим. Між релізами хтось вносить термінову зміну конфігурації лише в живе середовище. Наступний реліз потрапляє на вільний бік, де досі старе налаштування, і перемикання, яке мало змінити одну річ, змінює дві. Будуйте обидва слоти з тих самих шаблонів і того самого джерела конфігурації, щоб зміна одного була зміною обох, і використовуйте вільний бік як staging для наступного релізу, як описує Фаулер.

Що почитати далі

- Blue Green Deployment — опис Мартіна Фаулера 2010 року з його доповненням 2015 року про зміни бази даних.
- Argo Rollouts: BlueGreen — активний і preview-Service, автоматичне просування, аналіз до й після перемикання та затримка перед згортанням старої версії.
- Kubernetes Service — як селектор обирає поди, що отримують трафік сервісу.
- kubectl patch — strategic merge, merge і JSON-патчі, тобто команда, що перемикає селектор у прикладі вище.

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 синьо-зелене розгортання?» 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/blue-green-deployment/