Engineer

Розгортання

перенесення змін туди, де ними справді користуються

Що це таке

Розгортання — це послідовність кроків, яка забирає перевірену версію застосунку із системи збирання й вводить її в роботу: зібраний образ або файли потрапляють на машини чи платформу, що їх виконуватиме, версія отримує конфігурацію саме цього середовища, запускається, і на неї переводять справжні запити. Англійською це deployment, а в робочих розмовах часто кажуть просто «деплой». Слово позначає і дію, і її результат: можна сказати, що розгортання зупинилося на кроці з базою даних, або спитати, що сталося з учорашнім розгортанням. Розгортати можна вебсайт, API, заплановане завдання, зміну в базі даних чи бекенд мобільного застосунку.

Розгортання й випуск часто розводять. Розгорнути — значить покласти новий код туди, де він може працювати; випустити — вирішити, хто його побачить. Функцію, сховану за прапорцем, можна розгорнути в понеділок, а увімкнути для клієнтів у четвер, не торкаючись серверів удруге. Такий поділ дає команді постачати код малими й частими кроками, а рішення про терміни лишаються за тими, хто відповідає за продукт.

Багато розгортань сьогодні постачають контейнерний образ. Open Container Initiative, створена в червні 2015 року під егідою Linux Foundation, публікує навколо нього три специфікації: як влаштований образ, як середовище виконання запускає його після розпакування і як реєстри зберігають і віддають образи. Образ ідентифікують за дайджестом — хешем від вмісту, тож той самий дайджест завжди означає ті самі байти. Тег на кшталт `1.4.2` — це зручна для читання вказівка на образ, і якщо знову надіслати образ під тим самим тегом, тег вказуватиме вже на щось інше.

Методологія The Twelve-Factor App, на яку часто посилаються, коли йдеться про вебсервіси, ділить шлях від коду до запущеного сервісу на три етапи, суворо відокремлені один від одного. Етап збирання перетворює коміт на виконуваний пакет. Етап випуску поєднує цей пакет із конфігурацією одного середовища. Етап виконання запускає процеси з обраного випуску. Кожен випуск має унікальний ідентифікатор і після створення ніколи не змінюється; будь-яка зміна дає новий випуск, тож відкат зводиться до вибору попереднього.

Саме перемикання відбувається по-різному, залежно від того, що витримує сервіс. Стратегія recreate зупиняє стару версію, а потім запускає нову, і між ними лишається пауза. Поступове розгортання замінює екземпляри по кілька за раз, тож якийсь час трафік обслуговують обидві версії; саме так за замовчуванням працює Deployment у Kubernetes. За схеми blue-green нова версія працює поруч зі старою, увесь трафік перемикають одним кроком, а стара лишається напоготові, щоб забрати його назад. Канарковий реліз спершу спрямовує на нову версію невелику частку трафіку й розширює її, доки частота помилок і затримки лишаються в нормі.

Застосуйте поняття до рішення

Застосуйте «Що таке 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

Саме під час розгортання зміна починає приносити користь, і саме тоді помилка починає діставатися до користувачів. Команди, які розгортають рідко, зазвичай накопичують у кожному релізі багато змін, тож кожне розгортання несе більший ризик, а коли щось ламається, підозрюваних більше. Малі й часті розгортання тримають кожен крок зрозумілим і легким для скасування.

Розгортання вручну послідовне рівно настільки, наскільки послідовна людина, яка його виконує. Кроки пропускають, міняють місцями або виконують трохи інакше, коли підміняють колегу у відпустці. Конвеєр щоразу

виконує ті самі кроки й лишає слід: який коміт, який артефакт, яке середовище і коли. З цього сліду починається розбір інциденту, і часто це найшвидший спосіб з’ясувати, що саме змінилося.

Найцінніша звичка — зібрати один раз і просувати результат. Артефакт, перевірений у staging, має бути байт у байт тим самим, що потрапляє в production, а відрізнятися може лише конфігурація, яку середовище надає артефактові, щойно той прибуває. Якщо записати дайджест під час збирання й розгортати за дайджестом, цю обіцянку можна перевірити:

```
podman build -t registry.example.com/web:1.4.2 .
podman push --digestfile web.digest registry.example.com/web:1.4.2
kubectl -n staging set image deployment/web \
  web=registry.example.com/web@$(cat web.digest)
# той самий дайджест іде в production, коли staging перевірено
kubectl -n production set image deployment/web \
  web=registry.example.com/web@$(cat web.digest)
```

Відкат тримається на тій самій дисципліні. Коли випуски незмінні й мають ідентифікатори, повернутися назад означає вказати на попередній, і це займає секунди. Коли кожне середовище збирає все з вихідного коду, повернення означає наново зібрати старий коміт і сподіватися, що залежності розв’яжуться так само, як тоді.

Як це виглядає на практиці

За розгортання однієї платформи в Azure відповідали від початку до кінця, і кожна зміна проходила тим самим шляхом через development, staging і production, а напряму в робоче середовище ніхто нічого не викладав. Конвеєр CI/CD збирає образи й доставляє їх, а шаблони Bicep тримають цільову інфраструктуру однаковою в усіх середовищах, тож кожне просування потрапляє на ідентичну інфраструктуру. Налаштування, що різняться між середовищами, зберігають окремо від секретів, і тому одна збірка проходить усі етапи, підхоплюючи налаштування кожного середовища на місці, без перезбирання дорогою. Сторінка повідомляє, що зміни потрапляють у кожне середовище цим визначеним шляхом без імпровізованих ручних розгортань, а випуск став контрольованим кроком, який щоразу відбувається однаково.

Вагу такої послідовності в годинах показала раніша робота: автоматизація конвеєрів збирання, тестування й розгортання заощадила приблизно 4 000 годин, бо шлях від зміни до production став стандартним, а ручні кроки, що залежали від того, хто їх виконував, зникли. Сторінка повідомляє, що випуски пришвидшилися й стали надійнішими.

На власній платформі компанії розгортання відокремлене від підготовки хоста. Контейнерний шар на Podman і Quadlet дає root один раз підготувати шлях розгортання. Після цього непривілейований користувач випускає нову версію, запускаючи один фіксований скрипт через вузько окреслене правило, яке дозволяє лише цю одну команду, і то без аргументів. Скрипт збирає образ і перезапускає службу, тож звичайний реліз не потребує оболонки з правами root і не дає виконати довільну команду.

Де виникають проблеми

Більшість проблем із розгортанням виникає в розриві між тим, що перевірили, і тим, що насправді змінилося.

У production працює те, чого staging не бачив. Коли кожне середовище збирає власну копію, діапазони залежностей розв’язуються в новіші версії, базовий образ змінюється між двома збираннями або прапорець збирання стоїть інакше, і артефакт, що пройшов тести, — уже не той, який обслуговує клієнтів. Збирайте один раз, записуйте дайджест, просувайте саме його, а все, що залежить від середовища, тримайте в конфігурації, а не в артефакті.

Тег зсувається, і відкат потрапляє не на ту версію. Розгортання, яке посилається на `latest` або повторно використовує тег версії після перезбирання, залежить від того, на що тег вказує в день запуску. Документація Kubernetes радить не використовувати `:latest` у production, бо тоді важче зрозуміти, яка версія працює, і важче правильно відкотитися. Беріть теги версій, які ніколи не використовуються повторно, і фіксуйте дайджест там, де платформа це дозволяє.

Конвеєр звітує про успіх раніше, ніж нова версія запрацювала. Багато команд, які запускають розгортання, завершуються, щойно платформа прийняла запит, а нові екземпляри ще стартують і можуть упасти за хвилину. Крок, який чекає, доки нова версія повідомить про готовність, а тоді надсилає кілька справжніх запитів і перевіряє відповіді, перетворює тихий збій на червону збірку, поки попередній випуск ще під рукою.

Новий код і стара схема зустрічаються посеред розгортання. Під час поступового чи канаркового розгортання старі й нові версії обслуговують трафік пліч-о-пліч з однією базою даних. Міграція, яка перейменовує чи видаляє стовпець, ламає старі екземпляри, що ще працюють, і робить відкат небезпечним, бо старий код очікує того, що вже прибрали. Розносьте такі зміни на кілька розгортань: спершу додайте нову структуру, потім випустіть код, що працює з обома, і прибирайте стару структуру лише тоді, коли вона не потрібна жодній із запущених версій.

Що почитати далі

- Build, release, run — аргументи Twelve-Factor App на користь трьох окремих етапів і випусків, які після створення не змінюють.
- Open Container Initiative — проєкт під егідою Linux Foundation, що стоїть за специфікаціями образів, середовищ виконання й розповсюдження, на які спирається більшість контейнерних розгортань.
- OCI Image Format Specification — маніфести, шари, конфігурація й дайджести вмісту, що ідентифікують образ.
- Kubernetes Deployments — поступові оновлення, стратегія recreate, граничний термін просування й відкати.
- Kubernetes images — теги, дайджести, політики завантаження образів і чому `:latest` ускладнює відкат.

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 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/deployment/