Engineer

DevOps

∞ практики, що поєднують розробку програм і їхню експлуатацію

Що це таке

DevOps — спосіб організувати роботу над програмним забезпеченням, за якого ті, хто пише систему, відповідають і за те, як її збирають, випускають і тримають у робочому стані, а кроки між написанням коду та його запуском автоматизовано, а не передано від однієї команди до іншої. Назва складена з development та operations, тобто розробки й експлуатації, двох груп, які вона мала об'єднати.

NIST, американський National Institute of Standards and Technology, визначає DevOps як практики автоматизації процесів між командами розробки та експлуатації, щоб вони могли збирати, тестувати й випускати програми швидше й надійніше. Корисність цього визначення в тому, що воно називає головне: процеси посередині. Компанія може мати чудових розробників і чудових адміністраторів і все одно втрачати дні там, де одні передають реліз іншим.

На практиці DevOps проявляється як кілька звичок, що підсилюють одна одну:

- **Спільна відповідальність за продакшн.** Той, хто змінює код, бачить, що робить ця зміна після запуску, через моніторинг, логи й сповіщення, і відповідає за неї.
- **Автоматизований шлях до релізу.** Збірка, тести й розгортання працюють як пайплайн, щоразу однаково, зазвичай поверх CI.
- **Інфраструктура, записана у файлах.** Сервери, мережі й бази даних описано в шаблонах або плейбуках, що лежать поруч із кодом, тож середовище можна відбудувати, а не відтворювати з пам'яті.
- **Одна збірка, багато середовищ.** Артефакт, що пройшов тести на staging, — той самий, що потрапляє в production; змінюються лише його налаштування.

- **Малі й часті релізи.** Кожен містить небагато змін, тож помилку легко знайти й легко відкотити.

DevOps — не продукт і лише умовно назва посади. Постачальники продають інструменти під цією назвою, а компанії наймають «DevOps-інженерів», але ні інструмент, ні посада не є самою практикою. DevOps-інженер зазвичай будує й підтримує пайплайни та інфраструктуру, які дають іншим розробникам випускати зміни самостійно, і це роль усередині практики, а не її заміна.

Застосуйте поняття до рішення

Застосуйте «Що таке DevOps?» 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

Кожна передача між командами щось губить. Розробник знає, навіщо зроблено зміну і від чого вона залежить; адміністратор знає, як виглядають сервери і що зламалося минулого місяця. Коли реліз переходить від одних до інших як тікет і пакет, обидві половини цього знання доходять неповними, а прогалини виявляються під час розгортання, коли їх знаходити найдорожче.

The Twelve-Factor App, методологія для створення програм, що надаються як сервіс (software-as-a-service), точно називає три такі прогалини. Прогалина в часі: код, написаний сьогодні, потрапляє в production за кілька тижнів. Прогалина в людях: одна група пише код, а інша його розгортає. Прогалина в інструментах: на ноутбуці працює одна база даних, а в production — інша. DevOps значною мірою є спробою закрити всі три: розгортати через години після написання, дати авторам розгортати самим і тримати всі середовища якомога схожішими.

Розмір релізу — другий аргумент. Книга Google про site reliability engineering каже прямо: часті релізи означають менше змін між версіями. Реліз із трьома змінами легко розібрати, коли щось іде не так. Реліз, у якому їх триста, — це вже пошуки.

Третій аргумент — те, що та сама книга називає toil, тобто рутиною: робота, пов'язана з експлуатацією сервісу, яка є ручною, повторюваною, придатною до автоматизації, не має тривалої цінності й зростає разом із сервісом. Команда, яка щовівторка вручну перезапускає процес, оновлює сертифікати за нагадуванням у календарі й розширює диски, коли спрацьовує сповіщення, витрачає час на роботу, яку скрипт виконав би надійніше. Автоматизація такої роботи — спосіб, у який невелика команда встигає за системою, що росте.

Настанови з безпеки дійшли того самого. Настанова NIST щодо DevSecOps для cloud-native застосунків виходить із пайплайна з етапами збирання, тестування, пакування, розгортання та експлуатації і зараховує інфраструктуру, політики часу виконання й моніторинг до видів коду, які цей пайплайн описує.

Як це виглядає на практиці

Сторінки цього сайту показують DevOps не стільки як посаду, скільки як обсяг відповідальності: одна людина проєктує систему, виводить її в продакшн і утримує там.

У роботі над картографічним продуктом діяльність засновниці два роки охоплювала інфраструктуру, обробку даних і картографічний застосунок для клієнтів. Робота включала архітектуру, реалізацію та підтримку всіх цих рівнів: проєктування, побудову й обслуговування робочого середовища, процесів обробки даних і картографічного застосунку. Це поєднує розробку й експлуатацію в межах одного проєкту для замовника. Опис підтверджує обсяг і тривалість цієї інженерної роботи та обслуговування, а не вимірювання тривалості безперебійної роботи чи часу відповіді.

На парку з 10 інстансів PostgreSQL і MS SQL Server на VPS з Ubuntu Linux експлуатаційну частину автоматизовано крок за кроком: резервні копії через `pg_dump` і SQL Server Agent, моніторинг у Prometheus і Grafana, регулярне патчування, фаєрволи UFW, рольове керування доступом і задокументоване аварійне відновлення. Сторінка повідомляє про 99,9% безвідмовної роботи на всіх десяти серверах і скорочення ручного обслуговування на 50% завдяки автоматизації, а це аргумент про рутину, виміряний на одному парку серверів.

На платформі в Azure із середовищами development, staging і production розгортання було під повним контролем від початку до кінця, фіксованим маршрутом. Пайплайн CI/CD збирає образи, шаблони Вісер тримають інфраструктуру ідентичною від одного середовища до іншого, а конфігурацію, що відрізняється між середовищами, зберігають окремо від секретів. Завдяки цьому той самий зібраний артефакт просувається через середовища, а не перезбирається для кожного, і сторінка повідомляє про релізи без ситуативних ручних розгортань.

Де виникають проблеми

DevOps ламається тихо: зазвичай словник лишається, а звички зникають.

З'являється посада, а стіна лишається. Команду експлуатації перейменовують або наймають «DevOps-команду», яка сидить між розробниками й продакшном. Релізи так само проходять через передачу, просто з іншою назвою на тому боці. Перевірка проста: запитайте, хто може випустити зміну і кого розбудять, коли вона зламається. Якщо це різні люди з різною інформацією, практика лишилася на папері.

Розгортання досі — людина з чек-листом. Пайплайн збирає й тестує, а потім хтось заходить на сервер і виконує останні п'ять кроків з пам'яті, щоразу трохи інакше. Перенесіть і останні кроки в пайплайн, дайте релізам один маршрут через середовища, а ручний push у production розглядайте як інцидент, який треба пояснити.

Кожне середовище отримує власну збірку. Якщо перезбирати для staging і ще раз для production, протестоване й випущене — різні речі. Залежність, що на другій збірці розв'язалася інакше, проявляється як помилка, яку ніхто не може відтворити. Збирайте один раз, позначайте артефакт комітом, з якого його зібрано, і просувайте саме його, а налаштування беріть із середовища:

```
tag="$(git rev-parse --short HEAD)"
docker build -t "registry.example.com/app:$tag" .
docker push "registry.example.com/app:$tag"
# staging і production запускають той самий тег;
# різняться лише налаштування, з якими його стартують
docker run --env-file staging.env "registry.example.com/app:$tag"
```

Staging відпливає від production. Налаштування, змінене вручну на одному сервері, база даних, оновлена спершу в production, версія бібліотеки, закріплена в одному місці й плаваюча в іншому: кожна така дрібниця робить staging трохи менш корисним як генеральну репетицію. Описуйте інфраструктуру в шаблонах, застосовуйте ті самі шаблони всюди й запускайте в розробці ті самі допоміжні сервіси, що й у production.

Рутина росте разом із системою. Ручні завдання, що займали десять хвилин на тиждень для одного сервера, займають години для двадцяти. Без свідомого бюджету на їхню автоматизацію команда врешті витрачає дні лише на те, щоб усе трималося на плаву. Порахуйте повторювану ручну роботу й обмежте частку часу, яку вона може забирати.

Що почитати далі

- DevOps — визначення NIST разом із публікаціями, з яких його взято.
- Build, release, run — чому збірку роблять один раз і поєднують із конфігурацією, щоб отримати реліз.
- Dev/prod parity — прогалини в часі, людях та інструментах між розробкою і production і як їх закрити.
- Config — усе, що відрізняється між розгортаннями, винесене з коду в середовище.
- Eliminating Toil — визначення toil від Google і аргументи на користь того, щоб його обмежувати.
- Release Engineering — герметичні збірки, релізи за принципом самообслуговування і політики, які забезпечують інструменти випуску.

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 DevOps?» 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/devops/