Engineer

Інфраструктура як код

керування інфраструктурою через версіоновані файли опису

Що це таке

Інфраструктура як код — це підхід, за якого все, на чому працює система (машини, мережі, правила брандмауера, балансувальники навантаження, записи DNS), описують у текстових файлах, зберігають ці файли в системі контролю версій, а справжнє середовище з них будує інструмент. Оригіналом стають файли. Зміна інфраструктури починається як зміна у файлі, проходить те саме рев'ю, що й код застосунку, і потрапляє в робочу систему лише тоді, коли її застосує інструмент. В англомовній документації підхід називають Infrastructure as Code, часто скорочено IaC.

Посібник NIST SP 800-204C, присвячений DevSecOps для програм, побудованих із мікросервісів, описує інфраструктуру як код так: це код, що створює й налаштовує ресурси, на яких розгортають програму, і робить це відтворювано й однаково щоразу. За тим самим посібником, ці описи тримають під контролем версій тими самими інструментами, що й вихідний код самої програми. У відтворюваності вся суть. Середовище, яке можна створити з файлів, можна створити знову: поруч із першим, після аварії або в тестовому обліковому записі у вівторок по обіді.

Описи бувають двох основних видів. Декларативний опис називає кінцевий результат, скажімо, сервер певного розміру в певній мережі з одним відкритим портом, а інструмент сам визначає, що створити, що змінити, а що залишити. Імперативний опис — це перелік кроків: створи це, приєднай те, потім відкрий порт. SP 800-204C зараховує інфраструктуру як код до декларативного коду: команда записує, що їй потрібно, а інструмент перетворює це на робочий екземпляр. HashiCorp так само описує мову Terraform: вона задає мету, а не кроки до неї. Різниця видна під час другого запуску. Перелік кроків фіксує, як щось зробили одного разу, а декларативний опис — що має існувати, і саме це інструментові треба знати, коли частина вже існує.

Єдиної мови немає. SP 800-204C зауважує, що залежно від інструмента код пишуть або звичайною скриптовою мовою, як-от Python, JavaScript чи TypeScript, або власною мовою конфігурації інструмента, наприклад HCL від HashiCorp, яка може бути сумісною зі стандартним форматом на кшталт JSON, а може й ні. Мова Terraform належить до другого виду, і її файли мають розширення `.tf`. Azure приймає шаблони Resource Manager у JSON або в коротшій мові Bicep. Ansible пише свої плейбуки в YAML і дістається до машин через SSH з їхніми наявними обліковими даними, не встановлюючи на них агента.

Корисно розрізняти два завдання, які обидва потрапляють під цю назву. Підготовка ресурсів (provisioning) створює складники: віртуальну машину, диск, мережу, брандмауер. Налаштування формує те, що працює всередині вже наявної машини: пакети, облікові записи, службові файли. Одні інструменти тяжіють до першого завдання, інші до другого, і межа між ними розмита. Інфраструктурою як кодом обидва робить те, що опис лежить у репозиторії, а інструмент приводить справжню систему у відповідність до нього.

Інструменти ще й по-різному пам’ятають, що вони збудували. Terraform веде файл стану, який зіставляє ресурси з конфігурації зі створеними насправді об’єктами, і документація називає цей стан джерелом істини для середовища. Вісер файлу стану не має, бо Azure сам веде облік розгорнутого. Те, де лежить цей облік, визначає, що станеться, коли двоє людей запустять інструмент одночасно або коли облік загубиться.

Застосуйте поняття до рішення

Застосуйте «Що таке інфраструктура як код?» до конкретного рішення. Ми можемо пов'язати поняття із системою, яку ви створюєте або купуєте. Уся конфігурація записана у файлах, тож її можна відтворити точно, і вона не живе лише в чиїйсь голові.

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. Написала 1 000 правил RBAC для Linux-серверів — 1,000 правил доступу написано, щоб усунути вразливість

Написала 1 000 правил RBAC для Linux-серверів Зв'язатися з нами →

Чому це важливо

До інфраструктури як коду кожне середовище налаштовували вручну й потім так само вручну латали. Тестове середовище отримало виправлення, якого не отримав продакшн, порт, відкритий для перевірки, так і не закрили, одна

машина з трьох дістала новішу версію пакета. SP 800-204C називає це дрейфом: чинні налаштування вже не дають того робочого стану, який задумували. Посібник називає й звичайний шлях до нього: зміну, зроблену у вебконсолі чи в командному рядку в обхід коду. Коли реліз працює на тестовому середовищі й падає в продакшні, причина часто саме в такому розходженні.

Інфраструктура як код виносить оригінал за межі робочої системи. Команда змінює опис, а робоча система йде за ним. Ніхто не заходить на сервер, щоб поправити правило брандмауера: правлять файл, що описує брандмауер, а зміну виконує інструмент. З цього випливають три речі.

Перша — рев'ю та історія. Зміна мережі стає диффом у pull request, який читає друга людина, перш ніж щось станеться. Через кілька місяців на питання, хто й навіщо відкрив порт, відповідає історія комітів, а не чиясь пам'ять. Скасувати зміну означає зробити revert і ще раз запустити інструмент.

Друга — повторюваність. Добре написаний опис ідемпотентний: застосований до середовища, яке вже йому відповідає, він залишає все як є, а застосований до середовища, що відхилилося, повертає його назад. Шляхів до цього два: або підлаштувати те, що вже є, або відкинути його й збудувати нове з опису. Другий шлях, узятий за правило, — це незмінна інфраструктура.

Третя — дешеві середовища. SP 800-204C описує ці інструменти як автоматизацію всього життя інфраструктури, від створення ресурсів до їх вивільнення, і зауважує, що частина її живе так недовго, що підтримувати її немає сенсу. Команда може створювати кілька тестових середовищ на вимогу й потім прибирати їх, а сам код можна перевірити, перш ніж він щось зачепить: посібник вимагає сканувати кожен його фрагмент на вразливості до злиття. Гілка може отримати власну короткочасну копію продакшн-схеми, а рецензент — побачити попередній перегляд зміни до того, як її схвалити.

Ціну теж варто назвати прямо. Змінити одне правило брандмауера через pull request довше, ніж клацнути у вебконсолі, і під час збою це відчувається дорого. Кожен інструмент має власну мову й власну модель стану, які треба вивчити. А файл, що вміє збудувати середовище, вміє його й видалити: важіль діє однаково далеко в обидва боки, тому попередній перегляд важить стільки ж, скільки й запуск після нього.

Як це виглядає на практиці

Ось одна зміна на шляху від правки до робочої системи, у налаштуванні, де CI-завдання запускає інструмент і публікує пробний прогін у pull request:

```
День 1   09:10  Правка опису мережі: дозволити TCP 443 з офісу
         09:12  Відкрито pull request; CI робить пробний прогін (план)
         09:14  План у pull request: одне правило додано, жодне не прибрано
         10:30  Рецензент читає і план, і дифф, і схвалює
         10:31  Злиття; CI застосовує саме той план, що пройшов рев'ю
         10:33  Повторний запуск того самого опису: змін немає
День 40         Хтось питає, чому відкритий 443: історія комітів знає хто
```

План — це та частина, що виправдовує весь процес. Дифф показує, що автор хотів змінити, а план — що інструмент справді збирається зробити, зокрема те, чого автор не чекав, наприклад ресурс, який інструмент має намір знищити. Terraform уміє зберегти план у файл через `-out`, і `terraform apply` з цим файлом виконує саме записані в ньому операції, тож о 10:31 працює те, що рецензент схвалив о 10:30. Тихий повторний запуск о 10:33 підтверджує, що опис і система тепер узгоджені.

На цьому сайті сторінка послуги Infrastructure as Code описує відтворюване розгортання серверів на основі конфігурації під контролем версій, а не з опорою на чиюсь пам’ять.

Безпечний шлях до цього — здебільшого питання порядку, і запис у портфоліо за вересень 2026 року показує чому. Ідеться про ще один сервер, розміщений в іншого провайдера. Попередню машину налаштовували вручну й захистили пізніше, тож якийсь час щойно створений хост, що мав лише стандартну політику паролів, був відкритий для всього інтернету. Цього разу сервер створили з репозиторію, і захист з’явився раніше за саму машину. Плей брандмауера запускають до плею сервера, а в другого провайдера теги призначають до хмарного брандмауера, адже правилам, націленим на теги, ні до чого прив’язатися, доки тегів немає. Повторний запуск нічого не зіпсує: якщо машина вже є, роль просто бере її до роботи замість створювати ще одну. Лише ця роль коштує грошей, тому в її власній документації записано регіон, тип машини з характеристиками та місячну ціну в євро. Запис також чесно каже, чого не бачить пробний прогін. Обидва брандмауери налаштовуються викликами REST API провайдерів з універсального HTTP-завдання, а це завдання не підтримує режиму

перевірки (check mode). Через це пробний прогін його оминає. Справжня репетиція — окремий запуск плею брандмауера: він нічого не коштує, і його можна відкотити.

Де виникають проблеми

Інфраструктурний код зазвичай ламається так, що це скидається на успіх. Інструмент відпрацьовує без помилок, CI-запуск зелений, а файли й справжня система тихо розходяться.

Швидку ручну правку під час інциденту скасовує наступний запуск. Хтось о другій ночі відкриває порт у консолі, збій минає, а за тиждень плановий запуск знову закриває порт, бо файли про нього так і не дізналися. Буває й навпаки: інструмент більше ніхто не запускає, і файли перестають описувати реальність. Перенесіть аварійну зміну в код того ж дня і заведіть плановий запуск, який порівнює файли з тим, що працює. SP 800-204C вимагає такої звірки одразу після кожного розгортання і далі періодично під час роботи системи. У Terraform детальний код виходу команди plan робить із цього коротке нічне завдання:

```
terraform plan -detailed-exitcode
# exit 0: змін немає, exit 1: помилка, exit 2: є зміни
```

На гілці, яку вже застосовано, код виходу 2 означає, що справжня система відійшла від коду, і комусь варто з’ясувати чому.

Видалення ресурсу з файлу не завжди видаляє сам ресурс. Інструменти по-різному тлумачать ресурс, якого немає в описі. Для шаблонів Azure Resource Manager документація Microsoft про what-if пише, що ресурс, який існує, але відсутній у шаблоні, видаляється лише в режимі complete. В інших режимах він працює далі й далі коштує грошей. Вивчіть правило видалення свого інструмента, перш ніж на нього покладатися, і шукайте в попередньому перегляді як очікувані видалення, так і ті, яких боїтеся.

Пробний прогін бачить не все. Попередній перегляд — це модель зміни, а не гарантія. Документація Microsoft про what-if наводить властивості, які позначаються як видалені, хоча Azure лише підставляє значення за замовчуванням під час розгортання, і називає такий результат шумом; будь-яка властивість, задана через функцію `reference`, показується як змінена при

кожному запуску. Запис у портфоліо вище показує іншу прогалину: завдання без підтримки режиму перевірки пробний прогін пропускає, тож попередній перегляд про нього мовчить. Знайте відомий шум свого інструмента й вважайте неперевіреним усе, чого попередній перегляд не бачить, доки це не покриє репетиція на чомусь дешевому й оборотному.

Секрети потрапляють у репозиторій або в стан. Пароль, вписаний у файл опису, лишається в історії кожної копії репозиторію. Менш очевидне: документація Terraform попереджає, що стан може містити секрети, а за замовчуванням цей стан — локальний файл `terraform.tfstate`. Тримайте секрети в окремому сховищі секретів і посилайтеся на них за назвою, зберігайте стан віддалено з блокуванням і контролем доступу, як радить HashiCorp, і ніколи не правте файл стану вручну: для потрібних змін існує команда `terraform state`.

Скрипт видають за опис. Shell-скрипт у репозиторії, що створює сервер, перебуває під контролем версій, але він не ідемпотентний: запустіть його двічі, і серверів стане два або посередині з'явиться помилка. Користуйтеся декларативними ресурсами або кроками, які спершу перевіряють, а потім діють, як роль вище, що підхоплює наявну машину. Документація Ansible формулює планку прямо: коли система вже в стані, описаному плейбуком, повторний запуск плейбука її не змінює.

Що почитати далі

- Implementation of DevSecOps for a Microservices-based Application with Service Mesh (NIST SP 800-204C) — декларативний код інфраструктури під контролем версій, дрейф і сканування.
- Terraform Language Documentation — декларативна мова, її ресурси та блоки.
- Command: plan — пробний прогін, збережені плани й детальні коди виходу.
- State — що записує Terraform, де це зберігається і чому це чутливі дані.
- Introduction to Ansible — автоматизація без агентів через SSH, плейбуки й бажаний стан.
- Template deployment what-if — попередній перегляд розгортання в Azure, типи змін і відомий шум.

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 інфраструктура як код?» 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/infrastructure-as-code/