Engineer

Ansible

інструмент, що приводить сервер до описаного стану

Що це таке

Ansible — це програмне забезпечення з відкритим кодом для автоматизації, яке під'єднується до машин через SSH і налаштовує кожну так, як описано в наборі YAML-файлів: які пакети встановлено, які користувачі й файли існують, які служби працюють і як вони налаштовані. Машину, з якої запускають Ansible, називають керівним вузлом; машини, які він налаштовує, — керованими вузлами, і жодного агента на них ставити не треба, бо Ansible входить із власними обліковими даними операційної системи й робить усю роботу через це з'єднання. Файли задають результат, а Ansible сам з'ясовує на кожному хості, чи цей результат уже досягнуто.

Складники мають прості назви. Інвентар (inventory) перелічує керовані вузли й розкладає їх на групи на кшталт `web` чи `db`, зі змінними, що діють на всю групу або на один хост. Плейбук містить один чи кілька плеїв (plays), і кожен плей зіставляє групу хостів зі списком завдань. Кожне завдання викликає модуль — невелику програму, яка знає, як керувати однією річчю: пакетом, файлом, службою, користувачем, правилом брандмауера. Власна документація проєкту описує плейбуки як повторювану, придатну до повторного використання й просту систему керування конфігурацією та розгортання на багатьох машинах. Роль збирає завдання, шаблони, типові змінні й обробники під одним ім'ям, тож «сервер PostgreSQL» чи «захищений демон SSH» пишуть один раз і застосовують до будь-якого хоста, якому це потрібно.

Короткий плей показує структуру:

```
- name: Вебсервери обслуговують сайт
  hosts: web
  become: true
  tasks:
```

```
    - name: nginx установлено
      ansible.builtin.package:
        name: nginx
        state: present

    - name: nginx працює зараз і запускається після перезавантаження
      ansible.builtin.service:
        name: nginx
        state: started
        enabled: true
```

Кожне завдання називає стан, а не дію. Більшість модулів спершу перевіряє, чи цей стан уже є, і завершується без жодних дій, якщо так, тож другий прогін цього плея на налаштованому хості нічого не змінює й прямо про це повідомляє. Ця властивість, ідемпотентність, і робить безпечним запуск того самого плейбука проти всього парку щоразу, замість того щоб вести облік, який хост отримав який скрипт.

Хто зважує Ansible, зазвичай порівнює його з трьома сусідами. Порівняно зі shell-скриптом різниця в тому, що скрипт перелічує кроки й виконує їх знову, потрібні вони чи ні, а модуль спершу перевіряє, а тоді діє. Порівняно з Terraform чи Bicep різниця в рівні: ці інструменти створюють хмарні ресурси — мережі, бази даних, віртуальні машини, — а Ansible налаштовує те, що працює всередині машини, коли вона вже існує. Багато команд користуються і тим, і тим. Порівняно з Puppet і Chef різниця в моделі: там зазвичай на кожній машині працює агент, що забирає конфігурацію з центрального сервера, а Ansible надсилає зміни звідти, де його запустили, і з іншого боку потребує лише SSH.

Застосуйте поняття до рішення

Застосуйте «Що таке Ansible?» 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

Сервер, налаштований вручну, відтворюваний рівно настільки, наскільки добре пам’ятає людина, яка його налаштовувала. За пів року вже ніхто напевно не скаже, чому параметр ядра має саме таке значення, яку версію пакета зафіксували й навіщо або що треба зробити першим, коли машину відбудовують. Коли налаштування записане у вигляді плейбуків, ця пам’ять стає файлами, які можна прочитати, переглянути в pull request і знову запустити на новій машині з тим самим результатом.

Записана форма змінює й те, що означає дрейф. Коли плейбук є описом хоста, прогін у режимі перевірки проти живої машини показує кожне місце, де вони розходяться. З прапорцем `--diff` звіт показує рядки в кожному файлі, які було б змінено. Парк, який регулярно зводять до описаного стану, тримається близько до свого опису, а хост, який редагували вручну, виринає як зміна під час наступного прогону плейбука.

Модель має свою ціну. Ansible діє лише тоді, коли його запускають, тож дрейф між двома прогонами лишається до наступного. Завдання виконуються по черзі, по одному на всіх відповідних хостах, і великий плейбук проти багатьох машин повільніший за скрипт, який робить одну річ. Логіку, записану в YAML і шаблонах Jinja2, стає важко відстежувати, коли роль обростає умовами. А прогін, що зупинився посередині, залишає хости частково зміненими, бо в Ansible немає транзакції, яку можна відкотити.

Як це виглядає на практиці

Кілька проєктів на цьому сайті показують, як записана інфраструктура виглядає з часом і чого коштує тримати її чесною.

Інфраструктура компанії як код охоплює розгорнуті публічні сервіси та конфігурацію інших компонентів, зокрема автоматизацію резервного копіювання й пошти, активація яких є окремим кроком. Зафіксована реалізація складається з 19 плейбуків і 34 ролей та 12 065 рядків YAML; ці цифри описують той етап, а не нинішній розмір репозиторію. Хост належить до групи рівня, змінні якої визначають його ролі, тому підготовка без аргументів приводить кожен хост до оголошеного стану. Ідемпотентність вважається контрактом: хост, приведений до потрібного стану, повідомляє про нуль змін, а плей, який не може цього сказати, ще не завершений. Чотири простори імен команд розділяють обіцянки — від контролю якості,

що не торкається жодного хоста, до перевірки, яка навмисно змінює хост і видає висновок. Сторінка чесно називає ціну: кожна зміна займає більше часу, ніж пряме редагування файла на сервері, а застосування, що зупинилося на півдорозі, гірше за те, яке відмовилося починати.

Контроль доступу — ще одна сфера, де письмовий опис окупається. 1 000 правил RBAC налаштовано за допомогою Ansible у середовищах розробників, примірниках застосунків, PostgreSQL, MS SQL і на серверах Linux, причому кожне правило пов'язане з конкретною роллю та системою. Зафіксований результат — документований набір рольових правил і відтворюваний спосіб їх застосовувати та перевіряти. Саме це підтверджує запис; він не дає кількісної оцінки зниження ризику безпеки чи економії часу на налаштування.

Щоб пробний прогін казав правду, знадобилася окрема робота. Check mode змусили казати правду на всій платформі після того, як у вересні 2026 року пробний прогін зупинився на твердженні, що хибно описувало хост, і порадив встановити прапорець, який послабив би пісочницю безпеки. Обхід виявив шість зондів, що ухвалювали рішення за значеннями, яких хост ніколи не давав. Зонди, які читають стан, не зачеплений прогоном, тепер виконуються навіть у режимі перевірки; решту пропускають із повідомленням, що саме лишилося неперевіреним, а правило потрапило до посібника з розробки.

Де виникають проблеми

Більшість клопоту з Ansible походить від того, що звіту довіряють більше, ніж механізму за ним.

Пробний прогін показує чистий результат для команд, яких він не виконував. Режим перевірки — це симуляція. Модулі, що його підтримують, передбачають свою зміну; документація каже, що модулі без такої підтримки нічого не звітують і нічого не роблять. Модулі `command` і `shell` підтримують його лише частково: якщо не вказано `creates` чи `removes`, вони пропускають команду й реєструють результат із порожнім виводом. Будь-яка подальша умова `when:`, що читає цей результат, ухвалює рішення за значенням, якого ніхто не вимірював. Зонду, який лише читає хост, можна наказати виконуватися попри режим перевірки, а той, що виконатися не може, має про це сказати:

```
- name: Прочитати стан брандмауера навіть у режимі перевірки
  ansible.builtin.command: ufw status
  register: fw
  check_mode: false
  changed_when: false

- name: Повідомити, що пробний прогін не перевірив
  ansible.builtin.debug:
    msg: "режим перевірки: перезапуск служби не перевірено"
  when: ansible_check_mode
```

Прогін зупиняється на дев'ятому завданні, а перші вісім уже застосовано. Коли завдання на хості падає, Ansible виключає цей хост із решти прогону й залишає все, що вже змінив. Перевірки, які можна виконати до того, як щось торкнеться хоста, мають стояти перед застосуванням конфігурації. Саме так сталося після зміни секретів у власному господарстві компанії: попередній перевірочний плей тепер спершу перевіряє сховище локально і виконує ті самі файли завдань, що й справжнє застосування конфігурації, приблизно за секунду, тож секрет, значення якого не вдається визначити, коштує двосекундної відмови на ноутбуці замість частково застосованої зміни на живому сервері. Та сама сторінка описує пастку: факти, встановлені під час одного плея, діють і після нього, тож перевірка й застосування конфігурації, зчеплені в один виклик, можуть поділити значення, яке перевірка мала б прочитати наново.

Хост, якого ніхто не чіпав, звітує про зміни на кожному прогоні. Модулі `command` і `shell` повідомляють про зміну щоразу, коли виконуються, бо не можуть знати, що зробила команда. Плейбук, повний таких завдань, ніколи не сходиться до нуля, і справжні зміни тонуть у шумі. Віддавайте перевагу модулю, який керує річчю напряму, додайте `creates` чи `removes` або скажіть Ansible через `changed_when`, коли вивід означає зміну.

Пароль з'являється в журналі чи в репозиторії. Змінні зберігаються відкритим текстом, якщо їх ніщо не шифрує. Ansible Vault шифрує файли змінних або окремі значення, щоб вони могли лежати в репозиторії, а `no_log: true` на завданні тримає його аргументи й результати поза виводом. Запустіть плейбук один раз із високою детальністю в тестовому середовищі й прочитайте, що він друкує, перш ніж покладатися на будь-що з цього.

Що почитати далі

- Ansible Community Documentation — власна точка входу проєкту до встановлення, понять і всіх посібників.
- Ansible playbooks — плеї, завдання, модулі, порядок виконання й що ідемпотентність означає на практиці.
- Validating tasks: check mode and diff mode — що пробний прогін бачить і чого не бачить, і перемикачі для окремих завдань.
- How to build your inventory — хости, групи й прив’язані до них змінні.
- Protecting sensitive data with Ansible Vault — шифрування файлів і окремих значень, що лежать поруч із плейбуками.
- ansible.builtin.command — довідка модуля, зокрема про його часткову підтримку режиму перевірки.

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 Ansible?» 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/ansible/