Engineer

# Ansible

una herramienta que lleva servidores al estado declarado

## Qué es

Ansible es software de automatización de código abierto que conecta por SSH con máquinas y lleva cada una a la configuración descrita en archivos YAML: paquetes instalados, usuarios y archivos existentes, servicios activos y su configuración. La máquina desde la que se ejecuta es el nodo de control; las configuradas son nodos gestionados. No necesitan un agente instalado: Ansible inicia sesión con las credenciales del sistema operativo y trabaja sobre esa conexión. Los archivos enumeran el resultado y Ansible averigua en cada host si ya se cumple.

Las piezas tienen nombres sencillos. Un inventario enumera los nodos y los agrupa, por ejemplo en `web` o `db`, con variables para un grupo o un host. Un playbook contiene uno o más plays; cada play asocia un grupo de hosts a una lista de tareas. Cada tarea llama a un módulo, un programa pequeño que gestiona una clase de elemento: paquete, archivo, servicio, usuario o regla de cortafuegos. La documentación describe los playbooks como un sistema sencillo, repetible y reutilizable de gestión de configuración y despliegue en varias máquinas. Un rol agrupa tareas, plantillas, variables predeterminadas y manejadores bajo un nombre: un servidor PostgreSQL o un demonio SSH reforzado se escribe una vez y se aplica a cualquier host que lo necesite.

Un play breve muestra la estructura:

```
- name: Web servers serve the site
  hosts: web
  become: true
  tasks:
    - name: nginx is installed
      ansible.builtin.package:
        name: nginx
```

```
        state: present

    - name: nginx runs now and after a reboot
      ansible.builtin.service:
        name: nginx
        state: started
        enabled: true
```

Cada tarea nombra un estado, no una acción. La mayoría de los módulos comprueba primero si ese estado ya se cumple y termina sin hacer nada cuando es así. Por tanto, ejecutar por segunda vez este play en un host ya configurado no cambia nada y así lo indica. Esa propiedad, la idempotencia, permite ejecutar con seguridad el mismo playbook contra todo el parque de máquinas cada vez, en lugar de llevar la cuenta de qué script se ha aplicado a cada host.

Quienes consideran Ansible suelen compararlo con tres alternativas cercanas. Frente a un script de shell, este enumera pasos y los repite hagan falta o no; un módulo comprueba antes de actuar. Frente a Terraform o Bicep, cambia la capa: estos crean recursos de nube como redes, bases de datos y máquinas virtuales; Ansible configura lo que corre dentro de una máquina existente. Muchos equipos usan uno de cada clase. Frente a Puppet y Chef, cambia el modelo: suelen ejecutar en cada máquina un agente que obtiene la configuración de un servidor central; Ansible la envía desde donde se inicia y solo necesita SSH al otro lado.

Utilice el concepto en una decisión

Aplique «¿Qué es Ansible?» a una decisión real. Podemos ayudarle a relacionar el concepto con el sistema que está construyendo o comprando. Toda la configuración escrita en archivos para poder recrearla con exactitud y evitar que exista solo en la memoria de una persona.

**Explore nuestros servicios** Póngase en contacto →

Compruebe los supuestos detrás del término

Utilice la explicación y los trabajos enlazados para formular preguntas sobre su proyecto. Podemos examinar los supuestos con usted antes de elegir una herramienta. 1.000 reglas RBAC para servidores Linux — 1,000 reglas de acceso escritas para cerrar una brecha de seguridad

**1.000 reglas RBAC para servidores Linux** Póngase en contacto →

## Por qué importa

Un servidor configurado a mano es tan reproducible como la memoria de quien lo preparó. Seis meses después nadie sabe con seguridad por qué un parámetro del kernel vale eso, qué versión de un paquete se fijó y por qué o qué debe hacerse primero al reconstruir la máquina. Escribirlo en playbooks convierte esa memoria en archivos legibles, revisables en una pull request y ejecutables en una máquina nueva con el mismo resultado.

La forma escrita también cambia el significado de desviación. Si el playbook describe el host, ejecutarlo en modo de comprobación sobre una máquina activa informa de cada discrepancia. `--diff` muestra las líneas que cambiarían en cada archivo. Un conjunto reconciliado periódicamente se mantiene cerca de su descripción; una edición manual aparece como cambio en la siguiente ejecución.

El modelo tiene costes. Ansible solo actúa cuando se ejecuta; la desviación entre ejecuciones permanece hasta la siguiente. Las tareas se ejecutan en orden, una a una sobre los hosts coincidentes, y un playbook grande contra muchas máquinas es más lento que un script para una tarea. La lógica YAML y Jinja2 cuesta seguirla al crecer las condiciones de un rol. Una ejecución interrumpida deja hosts parcialmente modificados: no hay una transacción que revertir.

## Cómo se aplica en la práctica

Varios proyectos del sitio muestran un parque de sistemas descrito por escrito a lo largo del tiempo y el trabajo para mantener veraz la descripción.

La infraestructura como código de la empresa abarca servicios públicos desplegados y configuración de otros componentes, incluida la automatización de copias de seguridad y correo cuya activación es un paso separado. La implementación documentada comprende 19 playbooks y 34 roles en 12.065 líneas YAML; las cifras describen aquella etapa, no el tamaño actual del repositorio. Cada host pertenece a un grupo de nivel cuyas variables declaran qué roles ejecuta, por lo que aprovisionar sin argumentos lleva a cada host a su estado declarado. La idempotencia es un contrato: un host ya reconciliado informa de cero cambios, y un play que no pueda afirmarlo no está terminado. Cuatro espacios de nombres de comandos separan las promesas, desde un control que no toca ningún host hasta una verificación que lo modifica a propósito y emite un veredicto. La página declara el coste: cada cambio tarda más que editar un archivo en el servidor, y una reconciliación aplicada a medias es peor que una que se niega a continuar.

El control de acceso es otro ámbito donde compensa una descripción escrita. Se configuraron 1.000 reglas RBAC mediante Ansible en entornos de desarrollo, instancias de aplicaciones, PostgreSQL, MS SQL y servidores Linux, cada una vinculada a un rol y un sistema. El resultado documentado es un conjunto de reglas basado en roles y un método repetible para aplicarlo y revisarlo. Eso es lo que respalda el registro; no cuantifica la reducción del riesgo de seguridad ni el ahorro de tiempo de configuración.

Mantener veraz la simulación requirió trabajo propio. Se hizo que el modo de comprobación dijera la verdad en toda la plataforma tras una simulación de septiembre de 2026 detenida en una aserción errónea sobre el host que recomendaba un indicador capaz de relajar un aislamiento de seguridad. La revisión encontró seis sondas que decidían con valores nunca proporcionados por el host. Las que leen un estado no tocado por la ejecución se ejecutan ahora también en modo de comprobación; las demás se omiten con un mensaje que nombra lo no verificado. La regla se incorporó a la guía de desarrollo.

## Dónde surgen los problemas

La mayoría de problemas de Ansible nace de confiar más en el informe que en el mecanismo que lo produce.

**La simulación informa de éxito para comandos que nunca ejecutó.** El modo de comprobación simula. Los módulos compatibles predicen cambios; los no compatibles, según la documentación, no informan ni hacen nada. `command` y `shell` solo lo admiten parcialmente: salvo que se indique `creates` o `removes`, omiten el comando y registran una salida vacía. Un `when:` posterior que lea ese resultado decide sobre un valor no medido. Una sonda de solo lectura puede forzarse a ejecutar; si no puede hacerlo, debe decirlo:

```
- name: Read the firewall state, even in check mode
  ansible.builtin.command: ufw status
  register: fw
  check_mode: false
  changed_when: false

- name: Say what this dry run did not verify
  ansible.builtin.debug:
    msg: "check mode: the service restart was not exercised"
  when: ansible_check_mode
```

**Una ejecución se detiene en su novena tarea con las ocho primeras ya aplicadas.** Cuando una tarea falla en un host, Ansible lo retira del resto de la ejecución y deja intactos los cambios que ya haya hecho. Las comprobaciones que puedan ejecutarse antes de que algo toque un host deben ir delante de la reconciliación. Eso ocurrió después de un cambio de secretos en la infraestructura de este sitio: un play previo valida ahora primero el vault en local, ejecutando los mismos archivos de tareas que la reconciliación real en aproximadamente un segundo. Así, un secreto que no se resuelve cuesta una

negativa de dos segundos en un portátil, en vez de un cambio parcialmente aplicado a un servidor activo. La misma página señala una trampa: los facts definidos durante un play sobreviven a él, por lo que encadenar una comprobación y una reconciliación en una sola invocación puede compartir un valor que la comprobación debería haber vuelto a leer.

**Un host intacto informa de cambios en cada ejecución.** `command` y `shell` siempre notifican cambios porque no saben qué hizo el comando. Un playbook lleno de ellos nunca converge a cero y los cambios reales se pierden entre el ruido. Prefiera un módulo que gestione directamente el recurso, añada `creates` o `removes` o indique con `changed_when` cuándo la salida significa un cambio.

**Una contraseña aparece en un registro o repositorio.** Las variables son texto plano salvo cifrado. Ansible Vault cifra archivos de variables o valores sueltos para guardarlos en el repositorio; `no_log: true` evita mostrar los argumentos y resultados de una tarea. Ejecute una vez con detalle elevado en pruebas y lea la salida antes de confiar en cualquiera de los dos mecanismos.

## Lecturas adicionales

- Documentación de la comunidad Ansible — el punto de entrada del propio proyecto a la instalación, los conceptos y todas las guías.
- Playbooks de Ansible — plays, tareas, módulos, orden de ejecución y qué significa la idempotencia en la práctica.
- Validación de tareas: modos check y diff — qué puede y qué no puede ver una simulación y los controles de cada tarea.
- Cómo construir el inventario — hosts, grupos y las variables asociadas a ellos.
- Proteger datos sensibles con Ansible Vault — cifrar archivos y valores individuales que residen junto a los playbooks.
- ansible.builtin.command — la referencia del módulo, incluido su soporte parcial del modo de comprobación.



Glosario

Elija según sus restricciones reales

Describa la ruta de despliegue, los riesgos operativos y quiénes se encargan de las guardias. Podemos definir el alcance de un cambio con comprobaciones, una vía de recuperación y responsabilidades claras.

**Vea cuánto cuesta** Póngase en contacto →

## Haga práctica la terminología

Pregunte cómo se aplica «¿Qué es Ansible?» a su organización. Basta un ejemplo concreto de su sistema actual para iniciar la conversación.

**Explore nuestros servicios**

welcome@engineer.company +45 50 10 61 84 Copenhague, Dinamarca

Llévese esta página

Tarjeta de contacto

Creado con esmero y cariño por D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Derechos de autor © 2025 – actualidad · Engineer ApS · Todos los derechos reservados.

Impreso desde https://engineer.company/es/glossary/ansible/