Engineer

YAML

formato de configuración que las personas pueden leer

Qué es

YAML es un formato de texto para datos anidados que indica la estructura con sangría, dos puntos y guiones en lugar de corchetes y comillas. Por eso es una elección habitual para la configuración que las personas escriben y revisan a mano. El nombre es un acrónimo recursivo: «YAML Ain't Markup Language». Los archivos terminan en `.yaml` o `.yml`. RFC 9512, publicado en febrero de 2024, registró el tipo de medio `application/yaml` y señaló `.yaml` como extensión preferida.

Bajo esa puntuación ligera hay un modelo de datos pequeño. Tiene mapas, que asocian claves y valores; secuencias, que son listas ordenadas; y escalares: cadenas, números, booleanos y null. Alrededor de ese modelo, el lenguaje añade lo que necesita quien edita un archivo: comentarios que empiezan con `#`, escalares de bloque que conservan saltos de línea con `|` o los pliegan con `>`, varios documentos en un archivo separados por `---`, anclas y alias (`&name` y `*name`) que permiten reutilizar un valor en otro lugar y etiquetas que nombran un tipo de forma explícita.

El primer framework de YAML se escribió en Perl en 2001. La versión 1.1 de la especificación llegó en 2005 y la 1.2 en 2009, centrada en hacer de YAML un superconjunto estricto de JSON y con muchas de las antiguas reglas de tipado implícito eliminadas. La revisión 1.2.2, publicada en octubre de 2021, corrigió errores y aclaró el texto sin cambiar el lenguaje. Aun así, mucho software sigue leyendo las reglas anteriores: la documentación de Ansible señala que PyYAML y libyaml implementan YAML 1.1.

YAML queda así entre dos vecinos. JSON es su subconjunto estricto, conciso y sin ambigüedades, pensado para que los programas intercambien datos. TOML es un formato de configuración más plano en el que toda cadena va entre comillas, por

lo que el tipo de un valor no se adivina. YAML ofrece más que ambos, y precisamente esa libertad adicional es donde aparecen sus problemas.

Utilice el concepto en una decisión

Aplique «¿Qué es YAML?» 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

La elección real es qué formato dar a un archivo que las personas mantendrán durante años: descripciones de despliegue, flujos de CI o configuración de servidores. Los playbooks de Ansible son YAML, igual que los archivos de configuración de muchas otras herramientas de infraestructura. Para gran parte de ese trabajo, otros ya han elegido el formato y solo queda decidir con qué rigor se escribe.

El argumento a favor de YAML es la legibilidad. Quien revisa puede leer un diff sin contar llaves; un comentario explica por qué un valor es el que es; y un script de varias líneas o un certificado cabe en un escalar de bloque sin escapes. A cambio, los espacios tienen significado: una clave con dos espacios de sangría de más puede acabar en otro mapa aunque el archivo siga siendo válido. Los tabuladores están completamente prohibidos en la sangría. Los valores sin comillas reciben su tipo por cómo se escriben, y esas reglas cambian entre 1.1 y 1.2. Las plantillas añaden una segunda sintaxis: en Ansible, un valor que empieza con `{{` debe ir entre comillas o el analizador lo interpreta como un mapa.

Lo que inclina la decisión es cómo se comprobará el archivo. YAML con un linter y el hábito de entrecomillar cadenas es un formato cómodo para la configuración que un equipo edita en común. Sin ninguna de esas prácticas, un archivo válido puede decir algo distinto de lo que pretendía su autor. Para los datos intercambiados entre programas, JSON es la opción predeterminada más segura, y para un archivo breve de ajustes con pocos niveles de anidamiento, TOML es más sencillo.

Cómo se aplica en la práctica

El repositorio de infraestructura de la empresa abarca servicios desplegados accesibles al público y configuración de otros componentes, incluidas funciones de base de datos, copias de seguridad, transporte de correo y DNS. La implementación registrada comprendía 19 playbooks de Ansible y 34 roles, con 12.065 líneas en total: es un recuento histórico, no el tamaño actual del repositorio. La automatización de copias y correo tiene un paso de activación operativa separado de su configuración. La composición se expresa como datos, en lugar de opciones de línea de comandos. Cada equipo pertenece a un grupo de nivel, y las variables YAML del grupo enumeran sus roles. Por ello, una ejecución de aprovisionamiento no necesita argumentos para llevar cada equipo al estado que declara su grupo. Aquí YAML es el lenguaje del estado declarado, y el resultado se comprueba en lugar de darse por hecho: un equipo ya convergido debe notificar cero cambios. YAML cuesta tiempo; cada cambio tarda más que editar un archivo en el servidor. Como una convergencia aplicada a medias es peor que una que se niega a empezar, ahora se ejecuta antes un play de comprobaciones previas.

Doce mil líneas de YAML necesitan que las lea una máquina antes que una persona. En la etapa documentada de implementación de agosto de 2026, YAML era uno de los lenguajes que comprobaba un control de commits con 22 linters, junto con HTML, CSS, JavaScript, Python, Markdown y shell. El control no tiene nivel de advertencia: todo diagnóstico detiene el commit, y las supresiones dentro del código están prohibidas por completo. Un comentario para ignorar una regla o una directiva que la desactive en el archivo no ofrece una vía para eludir el control. Para YAML esto importa porque la mayoría de las trampas siguientes son sintaxis perfectamente válida; solo una regla que las rechaza las detecta antes de que lleguen a un equipo.

Dónde surgen los problemas

Las trampas siguientes tienen algo en común: el archivo se analiza sin errores y significa otra cosa.

Un código de país se convierte en false. Bajo las reglas de YAML 1.1, `yes`, `no`, `on`, `off`, `y` y `n` sin comillas son booleanos en varias combinaciones de mayúsculas y minúsculas. El esquema básico de YAML 1.2 solo conserva `true` y `false`, con tres combinaciones de mayúsculas y minúsculas. RFC 9512

advierte que un analizador que siga la sintaxis antigua puede leer `{insecure: n}` como false en lugar de como la letra n. Una lista de países nórdicos muestra el problema:

```
countries:
    - DK
    - SE
    - NO # YAML 1.1 reads this as false
```

Ponga entre comillas cualquier cadena que pueda interpretarse como otra cosa, escriba los booleanos solo como `true` y `false` y deje que un linter haga cumplir ambas reglas.

Un número de versión pierde un dígito. Los números sin comillas se convierten, de modo que `version: 1.10` llega como el número de coma flotante 1,1. Bajo las reglas de 1.1, un cero inicial indica un número octal, así que un código escrito como `010` se convierte en 8. La documentación de Ansible pide entrecomillar una versión como `"1.0"` precisamente por este motivo. Ponga entre comillas identificadores, códigos y versiones: son cadenas que contienen dígitos.

Una clave acaba bajo el elemento padre equivocado. La sangría determina a qué mapa pertenece una clave; una línea desplazada dos espacios mueve un ajuste a otro lugar sin ningún error. La especificación exige además que cada clave de un mapa sea única. Una clave repetida mucho más abajo en un bloque largo es fácil de pasar por alto durante la revisión, por lo que yamllint incluye una regla para cada uno de esos dos errores. Ejecútelo antes de utilizar el archivo.

Cargar un archivo ejecuta código. Las etiquetas pueden nombrar tipos específicos de un lenguaje. Un cargador que las respete construirá el objeto que pida el archivo. Según la documentación de PyYAML, usar `yaml.load` con un cargador no seguro es comparable a usar `pickle.load` : puede construir objetos e invocar funciones de Python. La elección del cargador importa; las versiones actuales exigen indicarlo de forma explícita. `yaml.safe_load` solo reconoce etiquetas básicas de YAML. RFC 9512 pide desactivar de forma predeterminada la ejecución de código en los deserializadores. Use el cargador seguro para los archivos de configuración corrientes.

Un archivo corto llena la memoria. Las anclas y los alias convierten un documento YAML en un grafo en lugar de un árbol. Si cada ancla repite dos veces la anterior, cada línea añadida duplica el tamaño: treinta líneas cortas describen más de mil millones de cadenas cuando un programa recorre el resultado como un árbol ordinario. RFC 9512 lo llama expansión exponencial de datos, el ataque billion laughs, y propone procesadores que limitan la profundidad de recursión de las anclas. Acepte YAML externo solo mediante uno de esos procesadores y pruebe el límite en el analizador que realmente utiliza.

Lecturas adicionales

- YAML 1.2.2 specification — el propio lenguaje, incluido el esquema básico que determina el tipo de los valores sin comillas.
- RFC 9512: YAML Media Type — registro de `application/yaml`, con notas de seguridad sobre booleanos, ejecución de código y expansión de alias.
- YAML specification changes — cambios de cada revisión, de 1.2.1 a 1.2.2.
- Ansible: YAML Syntax — subconjunto que utiliza Ansible y sus problemas con las comillas.
- yamllint rules — comprobaciones de valores interpretados como verdaderos, claves duplicadas, números octales y sangría.
- PyYAML documentation — diferencia entre `load` y `safe_load`, explicada por la propia biblioteca.

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 YAML?» 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/yaml/