Engineer

Ansible

ein Werkzeug, das Server in beschriebene Zustände bringt

Was es ist

Ansible ist quelloffene Automatisierungssoftware, die sich über SSH mit Rechnern verbindet und sie in den Zustand versetzt, der in YAML-Dateien beschrieben ist: installierte Pakete, vorhandene Benutzer und Dateien sowie laufende Dienste und ihre Einstellungen. Der ausführende Rechner heißt Control Node, die konfigurierten Rechner Managed Nodes. Dort muss kein Agent installiert werden. Ansible meldet sich mit den Zugangsdaten des Betriebssystems an und arbeitet über diese Verbindung. Die Dateien nennen das Ergebnis; auf jedem Host ermittelt Ansible, ob es bereits vorliegt.

Die Bausteine tragen einfache Namen. Ein Inventar listet verwaltete Rechner und ordnet sie Gruppen wie `web` oder `db` zu. Variablen können für eine Gruppe oder einen Host gelten. Ein Playbook enthält ein oder mehrere Plays. Jedes Play verbindet eine Hostgruppe mit einer Aufgabenliste. Jede Aufgabe ruft ein Modul auf, ein kleines Programm für eine bestimmte Art von Gegenstand: Paket, Datei, Dienst, Benutzer oder Firewallregel. Die Projektdokumentation beschreibt Playbooks als wiederholbares, wiederverwendbares und einfaches System für Konfigurationsmanagement und Bereitstellungen auf mehreren Rechnern. Eine Rolle bündelt Aufgaben, Vorlagen, Standardvariablen und Handler unter einem Namen. „Ein PostgreSQL-Server“ oder „ein gehärteter SSH-Daemon“ wird so einmal geschrieben und jedem Host zugewiesen, der ihn braucht.

Ein kurzes Play zeigt die Form:

```
- 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
```

Jede Aufgabe nennt einen Zustand statt einer Handlung. Die meisten Module prüfen zuerst, ob er bereits gilt, und beenden sich dann ohne Änderung. Der zweite Lauf dieses Plays auf einem eingerichteten Host verändert deshalb nichts und meldet dies. Diese Eigenschaft, die Idempotenz, macht es sicher, dasselbe Playbook jedes Mal auf den gesamten Bestand anzuwenden, statt mitzuschreiben, welches Skript welcher Host bereits erhalten hat.

Wer Ansible prüft, vergleicht es häufig mit drei Nachbarn. Gegenüber einem Shellskript liegt der Unterschied darin, dass ein Skript Schritte aufzählt und auch unnötige wiederholt, während ein Modul vor der Handlung prüft. Gegenüber Terraform oder Bicep unterscheidet sich die Schicht: Diese Werkzeuge erzeugen Cloud-Ressourcen wie Netze, Datenbanken und virtuelle Maschinen; Ansible konfiguriert nach deren Entstehung das Innere eines Rechners. Viele Teams verwenden beides. Gegenüber Puppet und Chef unterscheidet sich das Modell: Diese lassen üblicherweise auf jedem Rechner einen Agenten laufen, der seine Konfiguration zentral abholt. Ansible überträgt sie dagegen vom jeweiligen Startort aus und benötigt auf der Gegenseite nur SSH.

Das Konzept für eine Entscheidung nutzen

Nutzen Sie „Was ist Ansible?“ für eine konkrete Entscheidung. Wir können helfen, das Konzept mit dem System in Verbindung zu bringen, das Sie entwickeln oder einkaufen. Die gesamte Einrichtung wird in Dateien festgehalten, damit sie sich exakt wiederherstellen lässt und nicht nur im Kopf einer einzelnen Person existiert.

Unsere Leistungen entdecken Kontakt aufnehmen →

Die Annahmen hinter dem Begriff prüfen

Nutzen Sie die Erklärung und die verlinkten Arbeiten, um Fragen für Ihr Projekt zu formulieren. Wir können die Annahmen gemeinsam mit Ihnen prüfen, bevor ein Werkzeug ausgewählt wird. 1.000 RBAC-Regeln für sichere Linux-Server — 1,000 Zugriffsregeln zum Schließen einer Sicherheitslücke

1.000 RBAC-Regeln für sichere Linux-Server Kontakt aufnehmen →

Warum es wichtig ist

Ein von Hand eingerichteter Server ist nur so reproduzierbar wie das Gedächtnis der einrichtenden Person. Sechs Monate später weiß niemand mehr sicher, warum ein Kernelparameter diesen Wert hat, welches Paket warum festgeschrieben wurde oder was bei einem Neuaufbau zuerst geschehen muss. Playbooks verwandeln diese Erinnerung in Dateien, die gelesen, im Pull Request geprüft und auf einem frischen Rechner mit demselben Ergebnis ausgeführt werden können.

Die Schriftform verändert auch die Bedeutung von Abweichungen. Beschreibt das Playbook den Host, meldet ein Lauf im Check-Modus gegen den echten Rechner jede Stelle, an der beide nicht übereinstimmen. Mit `--diff` zeigt der Bericht die zu ändernden Zeilen jeder Datei. Ein regelmäßig abgeglichener Bestand bleibt seiner Beschreibung nahe. Eine manuelle Bearbeitung erscheint beim nächsten Lauf als Änderung.

Das Modell kostet ebenfalls etwas. Ansible handelt nur bei seiner Ausführung; Abweichungen zwischen Läufen bleiben bis zum nächsten bestehen. Aufgaben laufen nacheinander über alle passenden Hosts. Ein großes Playbook für viele Rechner ist langsamer als ein Skript für eine einzige Sache. Logik in YAML und Jinja2-Vorlagen wird schwer lesbar, sobald eine Rolle viele Bedingungen enthält. Ein vorzeitig abgebrochener Lauf hinterlässt teilweise geänderte Hosts, weil Ansible keine Transaktion zum Zurückrollen besitzt.

Wie es in der Praxis aussieht

Mehrere Projekte dieser Website zeigen, wie ein schriftlich beschriebener Bestand mit der Zeit aussieht und was seine Verlässlichkeit verlangt.

Die Infrastruktur als Code des Unternehmens umfasst bereitgestellte öffentliche Dienste und Konfigurationen weiterer Komponenten, darunter Sicherungs- und E-Mail-Automatisierung, deren Aktivierung ein eigener Schritt ist. Die dokumentierte Implementierung umfasst 19 Playbooks und 34 Rollen in 12.065 YAML-Zeilen. Die Zahlen beschreiben diesen Stand, nicht die aktuelle Größe des Repositorys. Ein Host gehört zu einer Stufengruppe, deren Variablen seine Rollen festlegen. Die Provisionierung ohne Argumente gleicht deshalb jeden Host an seinen deklarierten Zustand an. Idempotenz gilt als Vertrag: Ein abgeglichener Host meldet keine Änderungen, und ein Play, das dies nicht kann, ist nicht fertig. Vier Befehlsnamensräume trennen die Zusagen, von einer Prüfung ohne

Hostzugriff bis zur Verifikation, die einen Host bewusst verändert und ein Urteil liefert. Der Preis wird offen genannt: Jede Änderung dauert länger als die direkte Bearbeitung einer Serverdatei. Ein nur halb durchlaufener Abgleich ist schlimmer als einer, der sich verweigert.

Auch bei Zugriffskontrolle zahlt sich eine schriftliche Beschreibung aus. 1.000 RBAC-Regeln wurden über Ansible konfiguriert, über Entwicklungsumgebungen, Anwendungsinstanzen, PostgreSQL, MS SQL und Linux-Server hinweg. Jede Regel gehört zu einer Rolle und einem System. Das dokumentierte Ergebnis ist ein beschriebener rollenbasierter Regelsatz und eine wiederholbare Art, ihn anzuwenden und zu überprüfen. So weit reicht der Nachweis; eine Verringerung des Sicherheitsrisikos oder eine Ersparnis bei der Einrichtungszeit wird darin nicht beziffert.

Die Wahrhaftigkeit des Probelaufs verlangte eigene Arbeit. Der Check-Modus wurde auf der ganzen Plattform verlässlich gemacht, nachdem ein Probelauf im September 2026 an einer falschen Behauptung über den Host abbrach und ein Flag empfahl, das eine Sicherheits-Sandbox gelockert hätte. Die Untersuchung fand sechs Sonden, die mit Werten entschieden, die der Host nie geliefert hatte. Sonden, die nur einen vom Lauf unberührten Zustand lesen, werden nun auch im Check-Modus ausgeführt. Die anderen werden mit einer Meldung übersprungen, die das Ungeprüfte nennt. Die Regel kam in den Entwicklungsleitfaden.

Wo es schiefgeht

Die meisten Ansible-Probleme entstehen, wenn dem Bericht stärker vertraut wird als dem Mechanismus dahinter.

Ein Probelauf meldet Erfolg für nie ausgeführte Befehle. Der Check-Modus simuliert. Unterstützende Module sagen Änderungen voraus; nicht unterstützende melden laut Dokumentation nichts und tun nichts. `command` und `shell` unterstützen ihn nur teilweise. Ohne `creates` oder `removes` überspringen sie den Befehl und registrieren ein Ergebnis mit leerer Ausgabe. Ein späteres `when:`, das dieses Ergebnis liest, entscheidet damit anhand eines nie gemessenen Werts. Eine rein lesende Sonde darf trotzdem ausgeführt werden. Was nicht laufen kann, sollte das mitteilen:

```
- 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
```

Der neunte Schritt stoppt, die ersten acht sind bereits angewendet. Scheitert eine Aufgabe auf einem Host, nimmt Ansible ihn aus dem restlichen Lauf und lässt seine Änderungen stehen. Prüfungen, die vor jedem Hostzugriff möglich sind, gehören vor den Abgleich. So geschah es nach einer Änderung an den Secrets im eigenen Bestand: Ein Vorab-Play prüft nun den Vault zuerst lokal. Es führt dieselben Aufgabendateien wie der echte Abgleich in ungefähr einer Sekunde aus. Ein nicht auflösbares Secret führt so zu einer zweisekündigen Ablehnung auf dem Laptop statt zu einer teilweise angewendeten Änderung am laufenden Server. Die Seite nennt auch eine Falle: In einem Play gesetzte Facts überleben es. Eine Prüfung und ein Abgleich im selben Aufruf können deshalb einen Wert teilen, den die Prüfung neu hätte lesen müssen.

Ein unberührter Host meldet bei jedem Lauf Änderungen. `command` und `shell` melden immer eine Änderung, weil sie nicht wissen können, was der Befehl getan hat. Ein Playbook voller solcher Aufgaben erreicht nie null Änderungen. Echte Änderungen verschwinden im Rauschen. Bevorzugen Sie ein direkt verwaltendes Modul, ergänzen Sie `creates` oder `removes` oder beschreiben Sie mit `changed_when`, wann eine Ausgabe eine Änderung bedeutet.

Ein Passwort erscheint im Protokoll oder Repository. Variablen sind ohne Verschlüsselung Klartext. Ansible Vault verschlüsselt Variablendateien oder Einzelwerte für das Repository. `no_log: true` verhindert bei einer Aufgabe die Ausgabe ihrer Argumente und Ergebnisse. Führen Sie das Playbook einmal mit hoher Ausführlichkeit in einer Testumgebung aus und lesen Sie die Ausgabe, bevor Sie sich auf beides verlassen.

Weiterführende Lektüre

- Ansible Community Documentation — der offizielle Einstieg zu Installation, Konzepten und sämtlichen Leitfäden.
- Ansible-Playbooks — Plays, Aufgaben, Module, Ausführungsreihenfolge und praktische Idempotenz.

- Aufgaben validieren: Check- und Diff-Modus — was Probeläufe sehen können und was nicht, einschließlich der Schalter pro Aufgabe.
- Ein Inventar aufbauen — Hosts, Gruppen und ihre Variablen.
- Vertrauliche Daten mit Ansible Vault schützen — Verschlüsselung von Dateien und Einzelwerten neben den Playbooks.
- ansible.builtin.command — die Modulreferenz mit ihrer teilweisen Check-Modus-Unterstützung.

Glossar

Für Ihre tatsächlichen Rahmenbedingungen auswählen

Beschreiben Sie den Bereitstellungsweg, die Betriebsrisiken und die Bereitschaftsverantwortlichen. Wir können eine Änderung mit Prüfungen, einem Wiederherstellungsweg und klaren Zuständigkeiten abgrenzen.

Preise ansehen Kontakt aufnehmen →

Die Begriffe praktisch nutzbar machen

Fragen Sie, wie „Was ist Ansible?“ auf Ihre Organisation zutrifft. Ein konkretes Beispiel aus Ihrem bestehenden System reicht aus, um das Gespräch zu beginnen.

Unsere Leistungen entdecken

welcome@engineer.company +45 50 10 61 84 Kopenhagen, Dänemark

Nehmen Sie diese Seite mit

Kontaktkarte

Mit Sorgfalt und Liebe gestaltet von D.

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

Copyright © 2025 – heute · Engineer ApS · Alle Rechte vorbehalten.

Gedruckt von https://engineer.company/de/glossary/ansible/