Engineer

Ansible

un outil qui place un serveur dans un état décrit par écrit

Ce que c’est

Ansible est un logiciel d’automatisation à code source ouvert qui se connecte aux machines par SSH et les place dans la configuration décrite par des fichiers YAML : paquets installés, utilisateurs et fichiers présents, services actifs et paramètres. La machine qui l’exécute est le nœud de contrôle ; celles qu’il configure sont les nœuds gérés. Aucun agent n’a besoin d’y être installé : Ansible se connecte avec les identifiants du système et travaille par cette connexion. Les fichiers décrivent le résultat et Ansible détermine sur chaque hôte s’il est déjà atteint.

Les composants ont des noms simples. Un inventaire liste les nœuds gérés et les classe en groupes comme `web` ou `db`, avec des variables pour un groupe ou un hôte. Un playbook contient un ou plusieurs plays, chacun associant un groupe d’hôtes à une liste de tâches. Chaque tâche appelle un module, petit programme sachant gérer un type d’élément : paquet, fichier, service, utilisateur, règle de pare-feu. La documentation décrit les playbooks comme un système simple, répétable et réutilisable de gestion de configuration et de déploiement sur plusieurs machines. Un rôle regroupe tâches, modèles, variables aux valeurs prédéfinies et gestionnaires sous un nom : « serveur PostgreSQL » ou « démon SSH durci » est écrit une fois puis appliqué à tout hôte concerné.

Un court play montre la structure :

```
- 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
```

Chaque tâche nomme un état, pas une action. La plupart des modules vérifient d'abord si cet état est déjà atteint et ne font rien dans ce cas. La seconde exécution sur un hôte configuré ne change donc rien et le signale. Cette propriété, l'idempotence, permet de relancer sans risque le même playbook sur tout le parc plutôt que de suivre les scripts déjà appliqués à chaque hôte.

Les personnes qui envisagent Ansible le comparent souvent à trois voisins. Face à un script shell, la différence est qu'un script énumère des étapes et les rejoue, nécessaires ou non, alors qu'un module vérifie avant d'agir. Face à Terraform ou Bicep, c'est la couche : ces outils créent des ressources cloud — réseaux, bases, machines virtuelles — tandis qu'Ansible configure l'intérieur d'une machine existante. Beaucoup d'équipes utilisent les deux. Face à Puppet et Chef, c'est le modèle : ceux-ci exécutent généralement un agent sur chaque machine qui récupère sa configuration d'un serveur central, tandis qu'Ansible la pousse depuis son point de lancement et ne nécessite que SSH à destination.

Utilisez le concept pour prendre une décision

Utilisez « Qu'est-ce qu'Ansible ? » pour éclairer une décision réelle. Nous pouvons vous aider à relier ce concept au système que vous construisez ou achetez. Toute la configuration décrite dans des fichiers, pour la recréer à l'identique sans qu'elle ne dépende de la mémoire d'une seule personne.

Découvrir nos prestations Contactez-nous →

Vérifiez les hypothèses derrière le terme

Appuyez-vous sur l'explication et les réalisations liées pour formuler les questions de votre projet. Nous pouvons examiner les hypothèses avec vous avant le choix d'un outil. 1 000 règles RBAC pour sécuriser Linux — 1,000 règles d'accès écrites pour combler une faille de sécurité

1 000 règles RBAC pour sécuriser Linux Contactez-nous →

Pourquoi c'est important

Un serveur configuré à la main n'est reproductible que dans la mesure où la personne qui l'a configuré s'en souvient. Six mois après, personne ne sait avec certitude pourquoi un paramètre du noyau a cette valeur, quel paquet a été maintenu à une version précise et pourquoi, ni l'ordre nécessaire à une

reconstruction. Écrire la configuration en playbooks transforme cette mémoire en fichiers lisibles, examinables dans une demande de fusion et réexécutables sur une machine neuve avec le même résultat.

Cette forme écrite change aussi le sens de la dérive. Lorsque le playbook décrit l'hôte, l'exécuter en mode vérification sur une machine en service signale tous leurs écarts. Avec `--diff`, le rapport montre les lignes de fichiers qui changeraient. Un parc régulièrement ramené à cet état reste proche de sa description, et une modification manuelle apparaît comme changement à la prochaine exécution.

Le modèle a un coût. Ansible n'agit que lorsqu'il est exécuté : la dérive intermédiaire reste jusqu'au passage suivant. Les tâches s'exécutent dans l'ordre, une à la fois sur tous les hôtes correspondants ; un grand playbook sur de nombreuses machines est plus lent qu'un script spécialisé. La logique YAML et Jinja2 devient difficile à suivre lorsqu'un rôle multiplie les conditions. Une exécution interrompue laisse aussi les hôtes partiellement modifiés, car Ansible n'a aucune transaction à annuler.

À quoi cela ressemble en pratique

Plusieurs projets de ce site montrent un parc décrit par écrit au fil du temps et le travail nécessaire pour garder cette description fidèle.

L'infrastructure sous forme de code de l'entreprise couvre les services publics déployés et la configuration d'autres composants, notamment l'automatisation des sauvegardes et de la messagerie, dont l'activation est une étape distincte. La mise en œuvre consignée comprend 19 playbooks et 34 rôles sur 12 065 lignes de YAML ; ces chiffres décrivent cette étape, pas la taille actuelle du dépôt. Un hôte appartient à un groupe de niveau dont les variables déclarent ses rôles ; le provisionnement sans argument ramène donc chaque hôte à son état déclaré. L'idempotence est un contrat : un hôte conforme annonce zéro changement, et un play incapable de le faire n'est pas terminé. Quatre espaces de noms de commandes séparent les engagements, du contrôle ne touchant aucun hôte à la vérification qui le modifie volontairement et rend un verdict. La page expose le coût : chaque changement est plus lent qu'une édition directe sur le serveur, et une convergence partielle est pire qu'un refus.

Le contrôle d'accès bénéficie aussi d'une description écrite. 1 000 règles RBAC ont été configurées par Ansible dans les environnements de développement, instances applicatives, PostgreSQL, MS SQL et serveurs Linux, chacune liée à un rôle et un système. Le résultat consigné est un ensemble documenté de règles fondées sur les rôles et une méthode reproductible pour les appliquer et les examiner. C'est ce que le dossier étaye ; il ne chiffre ni réduction du risque de sécurité ni gain de temps de configuration.

Rendre la simulation fidèle a nécessité un travail distinct. Le mode vérification a été rendu honnête sur toute la plateforme après qu'une simulation, en septembre 2026, a échoué sur une assertion fausse concernant l'hôte et conseillé un indicateur qui aurait assoupli un bac à sable de sécurité. L'examen a trouvé six sondes prenant des décisions à partir de valeurs jamais fournies par l'hôte. Celles qui lisent un état non touché par l'exécution tournent désormais même en mode vérification ; les autres sont ignorées avec un message nommant ce qui n'a pas été vérifié, et la règle figure dans le guide de développement.

Où les problèmes surviennent

La plupart des problèmes Ansible viennent d'une confiance accordée au rapport plutôt qu'au mécanisme qui le produit.

Une simulation déclare conformes des commandes jamais exécutées. Le mode vérification est une simulation. Les modules compatibles prédisent leurs changements ; selon la documentation, les autres ne font ni ne rapportent rien. `command` et `shell` ne le prennent en charge que partiellement : sans `creates` ou `removes`, ils ignorent la commande et enregistrent un résultat dont la sortie est vide. Tout `when:` ultérieur lisant ce résultat prend une décision à partir d'une valeur non mesurée. Une sonde en lecture seule peut être forcée à s'exécuter ; celle qui ne peut pas le faire doit le dire :

```
- 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
```

Une exécution s'arrête à la neuvième tâche après application des huit premières. Lorsqu'une tâche échoue sur un hôte, Ansible l'exclut de la suite et conserve les changements déjà effectués. Les vérifications possibles avant tout accès à l'hôte doivent précéder la convergence. Après une modification de secrets sur le parc de ce site, un play préalable valide désormais d'abord le coffre localement. Il exécute les mêmes fichiers de tâches que la convergence réelle en environ une seconde : un secret non résolu coûte donc un refus de deux secondes sur un ordinateur portable plutôt qu'une modification partielle du serveur en service. La page note aussi un piège : les faits définis dans un play lui survivent. Un contrôle et une convergence enchaînés dans un appel peuvent donc partager une valeur que le contrôle aurait dû relire.

Un hôte que personne n'a modifié annonce des changements à chaque passage. `command` et `shell` signalent toujours un changement, sans savoir ce que la commande a fait. Un playbook qui en est rempli ne converge jamais vers zéro et cache les vrais changements dans le bruit. Préférez un module gérant directement l'élément, ajoutez `creates` ou `removes`, ou utilisez `changed_when` pour indiquer quand la sortie signifie un changement.

Un mot de passe apparaît dans un journal ou dans le dépôt. Les variables sont en clair si rien ne les chiffre. Ansible Vault chiffre les fichiers de variables ou les valeurs individuelles pour les conserver dans le dépôt, et `no_log: true` empêche l'affichage des arguments et résultats d'une tâche. Exécutez une fois le playbook avec une forte verbosité dans un environnement de test et lisez sa sortie avant de faire confiance à ces protections.

Pour aller plus loin

- Documentation communautaire d'Ansible — point d'entrée du projet pour l'installation, les concepts et tous les guides.
- Playbooks Ansible — plays, tâches, modules, ordre d'exécution et idempotence en pratique.
- Validation des tâches : modes vérification et différences — possibilités et limites de la simulation, réglages par tâche.
- Construire un inventaire — hôtes, groupes et variables associées.
- Protéger les données sensibles avec Ansible Vault — chiffrement des fichiers et valeurs conservés à côté des playbooks.
- ansible.builtin.command — référence du module, dont sa prise en charge partielle du mode vérification.

Glossaire

Choisissez selon vos contraintes réelles

Présentez le parcours de déploiement, les risques opérationnels et les personnes d'astreinte. Nous pouvons définir un changement avec des vérifications, une procédure de reprise et une responsabilité claire.

Consulter les tarifs Contactez-nous →

Rendez la terminologie concrète

Demandez comment « Qu'est-ce qu'Ansible ? » s'applique à votre organisation. Un exemple concret tiré de votre système actuel suffit pour engager la discussion.

Découvrir nos prestations

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

Emportez cette page avec vous

Fiche de contact

Créé avec soin et amour par D.

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

Droit d'auteur © 2025 – aujourd'hui · Engineer ApS · Tous droits réservés.

Imprimé depuis https://engineer.company/fr/glossary/ansible/