Support Ansible

Tags	
Auteur	
Liens	
Type	Article
URL	
Usable	☐

Introduction

Qu'est-ce qu'Ansible ?

Ansible est un outil open-source d'automatisation, de déploiement et de gestion de configuration. Il permet de gérer et d'automatiser des tâches sur des serveurs distants ou locaux sans avoir à installer d'agents sur les machines cibles. Conçu pour être simple à utiliser, Ansible utilise le langage de balisage YAML pour décrire les tâches et les configurations, ce qui le rend facile à lire et à écrire pour les administrateurs système et les développeurs.

Comparaison d'Ansible avec Puppet et Chef

Philosophie et approche

Ansible

Ansible est un outil d'automatisation et de gestion de configuration basé sur une approche "push" qui utilise SSH pour se connecter aux hôtes cibles et exécuter des tâches. Ansible est conçu pour être simple, minimaliste et facile à apprendre, avec une syntaxe claire et lisible basée sur YAML.

Puppet

Puppet est un outil de gestion de configuration basé sur une approche "pull" avec une architecture maître/agent. Les agents Puppet s'exécutent sur les hôtes cibles et interrogent le serveur maître pour récupérer les configurations et les ressources. Puppet utilise son propre langage déclaratif, appelé DSL (Domain Specific Language), pour décrire l'état souhaité des ressources et des configurations.

Chef

Chef est également basé sur une approche "pull" avec une architecture maître/agent. Les agents Chef, appelés "chef-client", s'exécutent sur les hôtes cibles et récupèrent les configurations et les recettes du serveur Chef. Chef utilise un langage de script basé sur Ruby, appelé "recipes", pour décrire les étapes nécessaires pour configurer et gérer les ressources.

Architecture

Ansible

- Pas de serveur central ou d'agent requis
- Utilise SSH pour la communication avec les hôtes cibles
- Stocke les playbooks et les configurations localement ou dans un système de contrôle de version

Puppet

- Architecture maître/agent
- Les agents Puppet doivent être installés sur chaque hôte cible
- Les configurations et les modules sont stockés sur le serveur maître Puppet

Chef

- Architecture maître/agent
- Les agents Chef (chef-client) doivent être installés sur chaque hôte cible
- Les recettes et les configurations sont stockées sur le serveur Chef

Langage et syntaxe

Ansible

- Utilise YAML pour décrire les playbooks, les tâches et les configurations
- Facile à lire et à apprendre, même pour les non-programmeurs

Puppet

- Utilise un DSL déclaratif spécifique pour décrire les configurations et les ressources
- Moins lisible et plus complexe que YAML, mais plus flexible et expressif

Chef

- Utilise Ruby et un DSL spécifique pour décrire les recettes et les configurations
- Plus complexe que YAML, mais permet une programmation avancée et une personnalisation étendue

Communauté et écosystème

Ansible

- Large communauté d'utilisateurs et de contributeurs
- Nombreux modules, rôles et intégrations disponibles sur Ansible Galaxy
- Supporté par Red Hat

Puppet

- Large communauté d'utilisateurs et de contributeurs
- Nombreux modules et extensions disponibles sur Puppet Forge
- Supporté par Puppet Labs

Chef

- Large communauté d'utilisateurs et de contributeurs
- Nombreuses recettes et intégrations disponibles sur Chef Supermarket
- Supporté par Chef Software

Ansible Galaxy

Ansible Galaxy est une plateforme et un outil en ligne de commande pour partager et installer des rôles et des collections Ansible. Les fonctionnalités principales comprennent :

- Recherche et installation de rôles et de collections
- Partage de rôles et de collections avec la communauté
- Qualité et notation des rôles et des collections

Ansible Tower (ou AWX)

Ansible Tower est une interface web et une plateforme d'automatisation pour Ansible. Les fonctionnalités principales comprennent :

- Interface utilisateur graphique pour gérer et exécuter des playbooks et des tâches Ansible
- Gestion des inventaires et des accès, y compris le contrôle d'accès basé sur les rôles (RBAC)
- Intégration avec les outils de CI/CD, les services de notification et les systèmes de gestion des logs

Automatisation des tâches d'administration

Ansible facilite l'automatisation des tâches répétitives et chronophages, comme les mises à jour logicielles, la gestion des utilisateurs et l'installation de paquets.

Gestion de configuration

Ansible permet de définir et de gérer les configurations des systèmes et des applications de manière centralisée, ce qui assure la cohérence et l'exactitude des configurations sur l'ensemble de l'infrastructure.

Déploiement d'applications

Grâce à Ansible, les développeurs peuvent déployer rapidement et efficacement leurs applications sur diverses plates-formes et environnements, en automatisant le processus de déploiement.

Orchestration de services

Ansible peut orchestrer les services sur plusieurs serveurs, ce qui permet de gérer facilement les dépendances entre les services et d'assurer leur bon fonctionnement.

Provisionnement d'infrastructure

Ansible peut également être utilisé pour provisionner et gérer l'infrastructure, en automatisant la création, la configuration et la suppression de ressources, telles que les machines virtuelles, les réseaux et les stockages.

Comparaison avec d'autres outils d'automatisation

Comparé à d'autres outils d'automatisation, Ansible se distingue par sa simplicité et sa facilité d'utilisation. Voici quelques-unes des caractéristiques qui différencient Ansible des autres solutions :

Simplicité et facilité d'utilisation

Ansible est conçu pour être facile à apprendre et à utiliser, avec une syntaxe YAML claire et des concepts simples.

Pas besoin d'installer d'agent sur les machines cibles

Contrairement à d'autres outils d'automatisation, Ansible n'a pas besoin d'agent installé sur les machines cibles, ce qui simplifie la configuration et la maintenance.

Utilisation du langage YAML pour les playbooks

Ansible utilise le langage de balisage YAML pour décrire les tâches et les configurations, ce qui rend les playbooks faciles à lire et à écrire.

Grande variété de modules disponibles

Ansible dispose d'une large bibliothèque de modules pour gérer diverses tâches et services, ce qui permet aux utilisateurs de trouver facilement le module approprié pour leurs besoins.

Architecture et composants d'Ansible

L'architecture d'Ansible est basée sur une approche agentless et utilise SSH pour les connexions distantes.

Composants d’Ansible

Inventaires

- Les inventaires définissent les hôtes et les groupes d'hôtes sur lesquels les tâches doivent être exécutées.
- Ils peuvent inclure des informations telles que les adresses IP, les noms d'hôtes et les variables spécifiques à chaque hôte ou groupe.

Modules

- Les modules sont des unités de code réutilisables qui permettent d'exécuter des tâches spécifiques sur les hôtes cibles.
- Ansible dispose d'une grande variété de modules pour gérer différents services, applications et systèmes.

Tâches

- Une tâche est une action unique qui doit être exécutée sur les hôtes cibles, comme installer un paquet, créer un fichier ou démarrer un service.
- Les tâches sont définies dans les playbooks et sont exécutées à l'aide des modules appropriés.

Playbooks

- Les playbooks sont des fichiers YAML qui décrivent l'ensemble des tâches à effectuer sur les hôtes cibles.
- Ils sont organisés en "plays", qui sont des ensembles de tâches associées à un groupe d'hôtes.
- Les playbooks permettent d'automatiser et d'orchestrer des processus complexes de manière simple et lisible.

Rôles

- Les rôles sont des structures de répertoire organisées qui permettent de regrouper des tâches, des variables, des fichiers et des templates liés à une fonction spécifique.

- Les rôles facilitent la réutilisation et le partage de configurations entre différents projets et équipes.

Installation et configuration d'Ansible

Prérequis système

Avant d'installer Ansible, assurez-vous que votre système répond aux exigences suivantes :

- Un système d'exploitation basé sur Linux, macOS ou Windows (avec le sous-système Windows pour Linux)
- Python 3.6 ou version ultérieure

Installation d'Ansible

L'installation d'Ansible peut être réalisée de différentes manières, en fonction de votre système d'exploitation et de vos préférences personnelles. Voici quelques méthodes courantes pour installer Ansible :

- Utilisation du gestionnaire de paquets de votre système d'exploitation (par exemple, `apt`, `yum`, `dnf` ou `brew`)
- Installation via `pip`, le gestionnaire de paquets Python
- Installation à partir du code source

Configuration d'Ansible

Ansible peut être configuré de plusieurs manières :

1. Fichier `ansible.cfg`
2. Paramètres de la ligne de commande (CLI)

Configuration d'Ansible

Ansible cherche le fichier `ansible.cfg` à différents endroits, en fonction de l'ordre suivant (du moins prioritaire au plus prioritaire) :

1. `/etc/ansible/ansible.cfg`
2. `~/.ansible/ansible.cfg`
3. `ansible.cfg` dans le répertoire courant où se trouve votre playbook

4. Variable d'environnement `ANSIBLE_CONFIG` si elle est définie

Configuration d'Ansible

Voici un exemple de fichier `ansible.cfg` :

Le fichier de configuration `ansible.cfg` contient plusieurs options de configuration globale pour Ansible. Voici ce que chaque option fait :

inventory

Chemin d'accès à l'inventaire d'Ansible. Par défaut, Ansible utilise `/etc/ansible/hosts`.

forks

Nombre maximum de processus enfants à exécuter en parallèle. Par défaut, Ansible utilise 5.

sudo_user

Nom d'utilisateur pour exécuter les commandes avec `sudo`. Par défaut, Ansible utilise `root`.

ask_sudo_pass

Demander à l'utilisateur d'entrer un mot de passe pour `sudo` lorsqu'il est nécessaire. Par défaut, Ansible utilise `False`.

ask_pass

Demander à l'utilisateur d'entrer un mot de passe pour SSH lorsqu'il est nécessaire. Par défaut, Ansible utilise `False`.

gathering

Mode de collecte des faits (facts) sur les hôtes cibles. Par défaut, Ansible utilise `implicit`. Cette option contrôle comment Ansible récupère les informations sur les hôtes cibles, telles que les interfaces réseau, les partitions de disque et les versions du système d'exploitation. Les valeurs possibles sont `smart`, `explicit` et `implicit`.

gather_subset

Sous-ensemble de faits à collecter. Par défaut, Ansible utilise `all`.

roles_path

Chemin d'accès aux rôles d'Ansible. Par défaut, Ansible utilise `/etc/ansible/roles`.

log_path

Chemin d'accès au fichier journal d'Ansible. Par défaut, Ansible utilise `/var/log/ansible.log`.

vault_password_file

Chemin d'accès au fichier contenant le mot de passe pour déchiffrer les données chiffrées avec `ansible-vault`. Par défaut, Ansible utilise `None`.

fact_caching_connection

Chemin d'accès au cache des faits. Par défaut, Ansible utilise `/tmp`.

pipelining

Activer ou désactiver le pipelining. Par défaut, Ansible utilise `False`. Le pipelining est une fonctionnalité qui permet d'optimiser l'exécution des tâches en envoyant plusieurs commandes à la fois sur les hôtes cibles plutôt qu'une à la fois. Cela réduit le temps d'exécution global des playbooks, mais peut poser des problèmes de compatibilité avec certains modules et certains systèmes.

Pour plus d'informations sur la configuration d'Ansible, consultez la documentation officielle.

Utilisation de la commande `ansible-config`

La commande `ansible-config` offre plusieurs options pour afficher et gérer les paramètres de configuration d'Ansible :

- `ansible-config view` : affiche le fichier `ansible.cfg` pris en compte
- `ansible-config list` : liste toutes les variables et leurs valeurs
- `ansible-config dump` : affiche toutes les variables de configuration
- `ansible-config dump --only-changed` : affiche uniquement les valeurs modifiées par rapport aux valeurs par défaut

Consultez la documentation officielle pour plus d'informations sur la commande `ansible-config`.

Générer une clé SSH et la copier sur les nœuds en utilisant ssh-copy-id

Dans cette section, nous expliquerons comment générer une clé SSH sur votre nœud de contrôle (par exemple, votre ordinateur local ou un serveur centralisé) et copier la clé publique sur les nœuds cibles en utilisant l'utilitaire `ssh-copy-id`. Cela permettra à Ansible de se connecter et d'exécuter des commandes sur les nœuds cibles sans avoir besoin de saisir un mot de passe à chaque fois.

Générer une clé SSH

1. Ouvrez un terminal sur votre nœud de contrôle (ordinateur local ou serveur centralisé).
2. Exécutez la commande suivante pour générer une nouvelle paire de clés SSH (clé publique et clé privée) :

```
ssh-keygen -t rsa -b 4096 -C "your_email@example.com"
```

Remplacez `"your_email@example.com"` par votre adresse e-mail ou un commentaire pour identifier la clé.

- Lorsque vous êtes invité à saisir le fichier dans lequel enregistrer la clé, appuyez sur Entrée pour accepter le fichier par défaut (`~/.ssh/id_rsa`) ou spécifiez un autre chemin si vous préférez.
- Vous serez ensuite invité à saisir une passphrase pour protéger votre clé privée. Vous pouvez laisser ce champ vide pour ne pas utiliser de passphrase, ou saisir une passphrase sécurisée pour renforcer la sécurité de votre clé.

La commande `ssh-keygen` générera alors une paire de clés SSH (clé publique et clé privée) et les enregistrera dans les fichiers spécifiés (par défaut : `~/.ssh/id_rsa` pour la clé privée et `~/.ssh/id_rsa.pub` pour la clé publique).

Copier la clé publique sur les nœuds cibles

- Pour copier la clé publique sur un nœud cible, utilisez la commande `ssh-copy-id` suivie de l'adresse du nœud cible et du nom d'utilisateur :

```
ssh-copy-id user@target_node_ip
```

Remplacez `user` par le nom d'utilisateur du nœud cible (par exemple, `root` ou `ubuntu`) et `target_node_ip` par l'adresse IP ou le nom d'hôte du nœud cible.

Copier la clé publique sur les nœuds cibles

- Vous serez invité à saisir le mot de passe du nœud cible pour confirmer l'opération. Une fois que vous avez saisi le mot de passe, la clé publique sera copiée dans le fichier `~/.ssh/authorized_keys` du nœud cible.
- Répétez l'étape précédente pour chaque nœud cible sur lequel vous souhaitez copier votre clé publique.

Inventaires

Définition et format des inventaires

Les inventaires sont des fichiers texte ou des scripts qui décrivent les hôtes et les groupes d'hôtes sur lesquels les tâches Ansible doivent être exécutées. Un fichier d'inventaire peut inclure des informations telles que :

- Adresses IP
- Noms d'hôtes
- Variables spécifiques à chaque hôte ou groupe

Un exemple simple de fichier d'inventaire :

```
[group1]
host1.example.com
host2.example.com

[group2]
host3.example.com ansible_ssh_user=customuser
host4.example.com ansible_ssh_user=customuser

[group1:vars]
ansible_ssh_user=admin
```

Groupes et hôtes

Dans les fichiers d'inventaire, les hôtes peuvent être organisés en groupes pour faciliter la gestion des configurations et des tâches. Un hôte peut appartenir à plusieurs groupes, et les groupes peuvent être imbriqués pour créer des hiérarchies de groupes.

Les groupes sont définis en utilisant la syntaxe `[group_name]`, suivie d'une liste d'hôtes, un par ligne. Les variables spécifiques à un groupe peuvent être définies en utilisant la syntaxe `[group_name:vars]`.

Variables d'inventaire

Les variables d'inventaire sont des données spécifiques à un hôte ou à un groupe qui peuvent être utilisées dans les playbooks et les templates pour personnaliser les configurations en fonction des besoins de chaque hôte. Les variables d'inventaire peuvent être définies dans les fichiers d'inventaire, dans des fichiers de variables séparés, ou dans des fichiers de données externes.

Inventaires dynamiques

Les inventaires dynamiques sont des scripts ou des services qui génèrent des données d'inventaire à la demande, en fonction des ressources disponibles dans votre infrastructure. Les inventaires dynamiques permettent d'automatiser la découverte et la gestion des hôtes, en particulier dans les environnements cloud et les infrastructures en constante évolution.

Modules

Qu'est-ce qu'un module Ansible ?

Un module Ansible est une unité de code réutilisable qui permet d'exécuter une tâche spécifique sur les hôtes cibles. Les modules sont généralement écrits en Python, mais peuvent également être écrits dans d'autres langages, tant qu'ils sont compatibles avec l'interface de module Ansible.

Modules courants

Ansible dispose d'une grande variété de modules pour gérer différents services, applications et systèmes. Voici quelques exemples de modules couramment utilisés :

- `apt`, `yum`, `dnf`, `pacman` : gestion des paquets sur différentes distributions Linux
- `file` : gestion des fichiers et des répertoires
- `template` : création de fichiers de configuration à partir de templates Jinja2
- `copy` : copie de fichiers locaux vers des hôtes distants
- `fetch` : récupération de fichiers depuis des hôtes distants
- `service` : gestion des services système (démarrage, arrêt, redémarrage, etc.)
- `user` : gestion des utilisateurs et des groupes
- `command` et `shell` : exécution de commandes arbitraires sur les hôtes cibles

Utilisation de modules dans les tâches

Les modules sont utilisés dans les tâches définies dans les playbooks. Chaque tâche doit spécifier le module à utiliser et les arguments à passer au module. Par exemple, pour installer le paquet `nginx` en utilisant le module `apt`, on écrirait une tâche comme suit :

```
- name: Installer nginx
  ansible.builtin.apt:
    name: nginx
    state: present
  become: yes
```

Recherche et documentation des modules

Pour trouver et consulter la documentation des modules Ansible, vous pouvez utiliser les commandes suivantes :

- `ansible-doc -l` : liste tous les modules disponibles
- `ansible-doc <module_name>` : affiche la documentation pour le module spécifié

Vous pouvez également consulter la documentation en ligne sur le site Web d'Ansible : https://docs.ansible.com/ansible/latest/collections/ansible/builtin/index.html

Tâches, Playbooks et exécution

Création d'une tâche

Une tâche est une action unique à effectuer sur les hôtes cibles. Pour créer une tâche, spécifiez le module à utiliser et les arguments à passer au module. Par exemple :

```
- name: Installer nginx
  ansible.builtin.apt:
    name: nginx
    state: present
```

Création d'un playbook

Un playbook est un fichier YAML qui décrit l'ensemble des tâches à effectuer sur les hôtes cibles. Un playbook est organisé en "plays", chacun associé à un groupe d'hôtes. Voici un exemple simple de playbook :

```
---
- name: Configurer les serveurs web
  hosts: webservers
  tasks:
    - name: Installer nginx
      ansible.builtin.apt:
        name: nginx
        state: present
```

Exécution d'un playbook

Pour exécuter un playbook, utilisez la commande `ansible-playbook` suivie du nom du fichier de playbook et de l'inventaire à utiliser :

```
ansible-playbook -i inventory.ini myplaybook.yml
```

Vous pouvez également ajouter des options supplémentaires, telles que `-v` pour augmenter la verbosité de la sortie, ou `--limit` pour limiter l'exécution à certains hôtes.

Rôles

Qu'est-ce qu'un rôle Ansible ?

Un rôle Ansible est une structure de répertoire organisée qui permet de regrouper des tâches, des variables, des fichiers et des templates liés à une fonction

spécifique. Les rôles facilitent la réutilisation et le partage de configurations entre différents projets et équipes.

Structure d'un rôle

La structure typique d'un rôle Ansible inclut les répertoires suivants :

```
my_role/
├── defaults/
│   └── main.yml
├── files/
├── handlers/
│   └── main.yml
├── meta/
│   └── main.yml
├── tasks/
│   └── main.yml
├── templates/
└── vars/
    └── main.yml
```

- `defaults` : contient les variables par défaut du rôle, qui peuvent être remplacées par des variables d'autres sources (inventaires, playbooks, etc.)
- `files` : contient les fichiers statiques à copier sur les hôtes cibles
- `handlers` : contient les handlers, qui sont des tâches spéciales déclenchées par d'autres tâches (par exemple, redémarrer un service après une modification de configuration)
- `meta` : contient des métadonnées sur le rôle, telles que les dépendances, les auteurs et les licences
- `tasks` : contient les tâches principales du rôle, définies dans un fichier `main.yml`
- `templates` : contient les templates Jinja2 à utiliser pour générer des fichiers de configuration dynamiques
- `vars` : contient les variables spécifiques au rôle, qui ne devraient généralement pas être modifiées par les utilisateurs

Utilisation des rôles dans les playbooks

Pour utiliser un rôle dans un playbook, ajoutez une section `roles` dans le play et spécifiez le nom du rôle. Par exemple :

```
---
- name: Configurer les serveurs web
  hosts: webservers
  roles:
    - nginx
```

Vous pouvez également passer des variables spécifiques au rôle en utilisant la syntaxe `role_name: { variable_name: value }`. Par exemple :

```
---
- name: Configurer les serveurs web
  hosts: webservers
  roles:
    - role: nginx
      nginx_listen_port: 8080
```

Création et partage de rôles

Pour créer un rôle, utilisez la commande `ansible-galaxy` avec l'option `init` et spécifiez le nom du rôle :

```
ansible-galaxy init my_role
```

Cela générera la structure de répertoire de base pour le rôle. Vous pouvez ensuite ajouter vos tâches, variables, fichiers et templates comme décrit précédemment.