Présentation du produit Ansible

Objectifs de certification

RHCE EX294 (RHEL8)

Si vous étudiez pour une certification, on répond ici à l'objectif suivant :

- **2. Maîtrise des composants de base d'Ansible**
  - 2.1. Inventaires
  - 2.2. Modules
  - 2.3. Variables
  - 2.4. Facts
  - 2.5. Jeux
  - 2.6. Playbooks
  - 2.7. Fichiers de configuration
  - 2.8. Utiliser la documentation fournie pour trouver des informations spécifiques aux modules et commandes Ansible
- **3. Installation et configuration d'un nœud de contrôle Ansible**
  - 3.1. Installer les paquets requis

Présentation du produit Ansible

En guise d'introduction, on décrira dans ce document le projet Ansible, ses cas d'usage, ses composants, son principe de fonctionnement, son installation et les différents binaires qui l'accompagnent.

Après des généealités, le document décrit la terminologie et les composants Ansible. Il indique les procédures d'installation de Ansible sur une machine de contrôle Red Hat, CentOS, Fedora, Debian, Ubuntu, via PIP ou encore sous Windows WSL.

On tentera ici de comprendre de manière sommaire les composants de base d'Ansible tels que les connexions selon les cibles, les inventaires, les modules, les collections, le mode ad-hoc, les variables, les "Facts", les jeux, les playbooks ou livres de jeux, les fichiers de configuration YAML et INI, les sorties JSON et les modèles Jinja2, les rôles, Ansible Tower, l'idempotence.

Enfin, on ne manquera de décrire l'environnement d'exécution d'Ansible en décrivant les différentes commandes `ansible*` disponibles nativement.

1. Projet Ansible

1.1. Présentation générale

Ansible est un outil open-source de provisionnement de logiciels, de gestion des configurations et de déploiement d'applications qui permet de faire de l'infrastructure un code. Il fonctionne sur de nombreux systèmes de type Unix, et peut configurer aussi bien des systèmes de type Unix que Microsoft Windows. Il comprend son propre langage déclaratif pour décrire la configuration du système. Ansible est sans agent, se connectant temporairement à distance via SSH ou Windows Remote Management (permettant l'exécution à distance de PowerShell) pour effectuer ses tâches.

Ansible gère les différents noeuds avec un accès à distance natif (tels que les protocoles SSH ou Remote PowerShell ou encore des APIs natives). Il ne nécessite l'installation d'aucun logiciel supplémentaire à distance. Il offre des capacités de parallélisation, collecte de métadonnées et gestion des états. Cet aspect de conception "sans agent" installé sur le périphérique est important car il réduit les besoins d'infrastructure pour démarrer une gestion. Les modules fonctionnent grâce à JSON et à la sortie standard et peuvent être écrits dans n'importe quel langage de programmation.Le système utilise notamment YAML pour exprimer des descriptions réutilisables de systèmes, il fournit des sorties en JSON, il traite les variables grâce à des modèles Jinja2.

Le logiciel Ansible a été conçu par un ancien employé Red Hat, Michael DeHaan, également auteur de l'application de serveur de "provisionning" *Cobbler* et co-auteur du framework *Func* pour l'administration à distance. Le code source du logiciel est sous licence GNU General Public v3.0. Red Hat a racheté la société Ansible, Inc. en octobre 2015. [1]

1.2. Le terme Ansible

Une "*ansible*" est un dispositif théorique permettant de réaliser des communications à une vitesse supraluminique (supérieure à la vitesse de la lumière) imaginé en 1966 par Ursula K. Le Guin dans

son roman de science-fiction, *Le Monde de Rocannon*. Elle en détaillera plus tard le concept dans *Les Dépossédés* (1974). L'idée est notamment reprise par d'autres auteurs de livres de science-fiction et des jeux vidéos, la communication étant basée sur l'état d'énergie réciproque de deux particules jumelles. Par ailleurs, Ansible est le titre d'un magazine anglo-saxon consacré à la science-fiction. Enfin, le terme "ansible" peut faire référence à un système de communication hyperspace instantané fictif [2].

1.3. Version d'Ansible

Jusqu'à la version 2.9., le code source Ansible était situé dans un seul projet `ansible/ansible` avec deux mises à jour principales par an.[3]

Pour obtenir le numéro de la dernière version stable d'Ansible :

```
curl -s https://pypi.org/pypi/ansible-base/json | jq -r '.releases | keys | sort'
```

Ansible Classique (jusqu'en version 2.9) se caractérisait par :

- Un dépôt unique ansible/ansible.
- Un paquet unique appelé ansible
- Des sorties importantes deux fois par an

La version 2.10 d'Ansible va fondamentalement changer la portée des plugins inclus dans le dépôt `ansible/ansible`, en déplaçant une grande partie des plugins dans des dépôts de "collection" plus petits et qui seront distribués via https://galaxy.ansible.com/.

Aussi depuis la version 2.10. la version majeure est maintenue pendant un cycle de publication. Lorsque la version suivante sort (par exemple, 2.11), la version plus ancienne (2.10 dans cet exemple) n'est plus maintenue.

Concrètement, le dépôt ansible/ansible (ansible-base) ne contient plus que :

- Les programmes de base Ansible, `ansible-{playbook,galaxy,doc,test,etc}`.
- Quelques documents.
- Un minuscule sous-ensemble de modules et de plugins pour permettre le fonctionnement d'un contrôleur.

Tous ces éléments seront connus sous le nom d'`ansible-base`.

Les autres modules et plugins ont été déplacés dans diverses "collections".

Ainsi pour les "collections" :

- Elles peuvent être publiées indépendamment d'`ansible-base` et d'Ansible, selon un cycle ou une cadence de publication que le responsable de la collection préfère.

- Elles ont leur propre repo (GitHub, GitLab, etc.) avec un backlog dédié, c'est-à-dire plus de backlog partagé de lancement massif et de relations publiques.
- On pourrait continuer à effectuer des tests d'IC et, dans de nombreux cas, on peut les effectuer de manière plus approfondie.

Le paquet publié d'Ansible 2.10 va intégrer `ansible-base` et les diverses collections communautaires qui faisaient auparavant partie d'`ansible/ansible`.

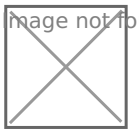
Le paquet `ansible` (contiendra un sous-ensemble de collections) et dépend du nouveau paquet `ansible-base` (Ansible engine).

1.4. Objectifs de conception de Ansible

Les objectifs de conception de Ansible comprennent[4] :

- **Le minimum par nature**. Les systèmes de gestion ne devraient pas imposer des dépendances supplémentaires sur l'environnement.
- **La cohérence**. cf. notion de test unitaire (procédure permettant de vérifier le bon fonctionnement d'une partie précise d'un logiciel ou d'une portion d'un programme).
- **La sécurité**. Ansible ne déploie pas des agents sur les noeuds. Un protocole de transport comme OpenSSH ou HTTPS est seulement nécessaire pour commencer une gestion.
- **La fiabilité**. Lorsqu'il est écrit soigneusement, un livre de jeu Ansible peut être ***idempotent*** afin d'éviter des effets secondaires inattendus sur les systèmes gérés.
- **Une courbe d'apprentissage faible**. Les livres de jeux Ansible utilisent un langage simple et descriptif basé sur YAML et les modèles Jinja2.

2. Automation d'infrastructures



Ansible s'interface avec du matériel, du logiciel ou des solutions d'un grand nombre de fournisseurs du domaine des infrastructures comme :

- Le déploiement d'applications (avec Fabric, Capristano, Nolio, …)
- L'orchestration multi-tiers (avc BMC, Mcollective, Chef Metal, …)
- Le provisionning (Cobbler, AWS, JuJu, …)
- La gestion des configuration (Chef, Puppet, CFEngine, …)

Il répond aux besoins les Admin système / Cloud, aux Net Ops, aux admins de stockage pour une automation des serveurs, du réseau et du stockage.

2.1. Péripériques physiques

- Bare Metal avec Cobbler, Stacki, and Red Hat Satellite
- Réseau avec Cisco, Juniper, Arista, A10, Cumulus Networks, Dell, F5 BigIP, HPE (OpenSwitch), Nokia, Palo Alto Networks etc.
- Stockage avec NetApp, Infinidat, etc.

2.2. Virtualisation

- VMware
- Red Hat Enterprise Virtualization (RHEV)
- Libvirt
- Xenserver
- Vagrant

2.3. Systèmes d’exploitation

- Linux (RHEL, CentOS, Fedora, Ubuntu, et autres)
- Windows et Windows Server
- UNIX

2.5. Containers

- Ansible Container
- Docker
- Linux Containers (LXC)

2.6. Cloud

- Amazon Web Services (AWS)
- Microsoft Azure
- Cloudstack
- OpenStack
- Digital Ocean
- Google Cloud Platform
- Linode

- ProfitBricks
- Rackspace

2.7. Outils DevOps

- Development:
  - Github,
  - Atlassian Bitbucket Pipelines,
  - Gitlabs,
  - Vagrant
  - …
- Integration/Test:
  - Jenkins,
  - Travis CI,
  - Teamcity
  - …
- Deployment:
  - Cloud Providers,
  - Containers,
  - ServiceNow,
  - Systems,
  - Virt Platforms
  - …
- Monitoring/Analytics:
  - Splunk,
  - AppDynamics,
  - Dynatrace,
  - LogicMonitor,
  - InfluxDB
  - …

3. Cas d’usage classiques

- Provisioning
- Configuration Management
- App Deployment
- Continuous Delivery
- Security & Compliance
- Orchestration

Ansible use cases

3.1. Provisioning

3.2. Configuration Management

3.3. App Deployment

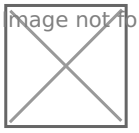
3.4. Continuous Delivery

3.5. Security & Compliance

3.6. Orchestration

4. Automation Réseau

Si vous êtes responsable d’un réseau d’entreprise, vous savez probablement que de nombreuses opérations manuelles sont effectuées via l’interface de ligne de commande (CLI). Il n’est pas surprenant que le principal défi que rencontrent des utilisateurs en matière de réseau consiste à améliorer leur agilité, et cela est resté vrai au cours des deux dernières années.[5]



Ansible peut gérer des périphériques :

- Arista (EOS),
- Cisco (IOS, IOS XR, NX-OS),
- Juniper (JunOS),
- Open vSwitch
- VyOS

4.1. Cas d’usage habituels

- Sauvegarder et restaurer les configurations des périphériques
- Mettre à jour les OS des périphériques réseau
- Vérifier la conformité des configurations (compliance)
- Appliquer des patchs sur base des CVE
- Générer une documentation dynamique

> Fondamentalement, toute opération manuelle peut être automatisée avec Ansible.

4.2. Cas d’usage habituels - automatiser des tâches discrètes

- S’assurer de la présence/absence de VLANs
- Activer / Désactiver Netflow sur les interfaces
- Gestion les entrées des access-list pare-feu

4.3. Meilleures pratiques

https://docs.ansible.com/ansible/2.5/network/user_guide/network_best_practices_2.5.html

5. Automatiser et coder l’infrastructure

Nous allons progressivement apprendre que manipuler Ansible consiste à écrire du texte géré sous forme de code informatique. L’ambition est donc de “coder l’infrastructure”. Alors comment coder l’infrastructure ? D’un point de vue historique, pour un parc de machines Linux, on commencera avec du Bash, ensuite probablement du Perl, et sans doute du Python. Or le personnel d’infrastructure s’il a choisi cette voie dans les métiers de l’informatique dipose de peu

d’appétence pour une pratique de “développeur” d’autant plus que l’apprentissage d’un langage de programmation classique exigera une longue courbe d’apprentissage pour cetains profils.

5.1. Automatiser l’infrastructure

Or Ansible permettra au personnel informatique d’automatiser les infrastructures. Il a l’avantage d’être :

- populaire, *diposant d’une communauté large*,
- flexible, *sans limite sur les fonctions ou les cibles à gérer*,
- évolutif, *aisé à adapter (en Python ou autres)*,
- lisible par tous et à courbe d’apprentissage faible, *grâce au format YAML*.

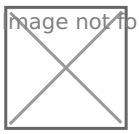
Le seul écueil est de connaître les solutions à gérer à travers cet outil de telle sorte qu’une maîtrise d’une solution Ansible passe aussi par une bonne maîtrise de ce qui est déployé par Ansible.

Il faut alors voir Ansible comme une surcouche sans état (mis en intermédiaire qui fonctionne de manière très simple) qui intègre d’autres solutions ou s’intègre à d’autres facilement. Ansible Tower et son fork OpenSource AWX renforcent cette capacité de manière crédible en ajoutant des fonctions de gestion des utilisateurs, du code et des secrets.

5.2. Coder l’infrastructure

D’ailleurs la gestion du code source (versions, SCM, git) et des secrets (vaults) sont des sujets connexes qui s’intègre à l’usage d’Ansible.

6. Composants Ansible



Fonctionnement de Ansible

En vue de contrôler des noeuds distants, des utilisateurs lancent des “playbooks” (livres de jeu) à partir d’un noeud de contrôle grâce à Ansible Engine.

Ansible Engine utilise différents composants comme :

- des modules
- des plugins
- un API
- un inventaire (inventory)

- des collections

Ansible Engine **se connecte** et **se pratique** de manière différente sur les hôtes terminaux (Linux/Unix et Windows) des périphériques du réseau. Hôtes terminaux et périphériques du réseau sont des cibles dont les missions et la gestion sont bien différentes.

6.1. Machine de contrôle

La machine de contrôle doit être un hôte Linux / Unix (par exemple, Red Hat Enterprise Linux, Debian, CentOS, OS X, BSD, Ubuntu), et Python 2.7 ou Python 3.5+ sont requis. Avec une version Windows 10 Pro, il est possible d'utiliser Ansible avec WSL, voir Can Ansible run on Windows?

Le chapitre Installation Ansible indique de manière plus détaillée l'installation de Ansible sur la machine de contrôle.

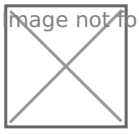
6.2. Noeuds gérés

Les noeuds gérés, s'ils sont en Linux/Unix, doivent disposer de Python 2.6 ou version ultérieure, **mais de préférence aujourd'hui le pré-requis est Python 3**. SSH est alors le mode de connexion préféré.

Depuis sa version 1.7, Ansible peut également gérer les noeuds Windows. Dans ce cas, on utilise PowerShell à distance de manière native au lieu de SSH.

Le matériel d'infrastructure réseau ("constructeurs"), pare-feux, serveurs de stockage, fournisseurs IaaS, solutions de virtualisation sont supportés via SSH, NETCONF/YANG ou encore via des API HTTP REST.

On trouvera ici un propos sur les Plugins de connexion.



Fonctionnement Ansible

Source : How Network Automation is Different

6.3. Clés SSH

La machine de contrôle et les noeuds gérés devraient *a priori* simplement communiquer grâce au service SSH.

Les mots de passe sont pris en charge, mais la gestion des clés SSH avec ssh-agent est l'un des meilleurs moyens d'utiliser Ansible. Il est également possible d'utiliser parmi beaucoup de possibilités des authentifications Kerberos ou autres. Les connexions "root" ne sont pas obligatoires, on peut se connecter en tant qu'utilisateur, et puis utiliser "su" ou "sudo". Le module "authorized_key" d'Ansible est un excellent moyen d'utiliser Ansible pour contrôler les accès SSH.

```
ssh-agent bash
ssh-add ~/.ssh/id_rsa
```

Notez que `ssh-agent` s'utilise avec des clés avec des passphrase.

On trouvera sous les liens un document de formation détaillé sur l'usage du protocole avec OpenSSH (Linux/Windows) et d'autres sur la configuration de Cisco IOS pour supporter des connexions SSH.

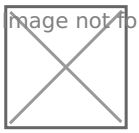
Ansible supporte beaucoup d'autres types de connexions/communications avec ses cîbles.

6.4. Le dossier de projet

- Fichier de configuration
- Inventaire
- Livres de jeu

6.5. L'inventaire

Pour démarrer un gestion Ansible, vous avez besoin d'un inventaire Ansible.



Inventory (inventaire)

Par défaut, Ansible représente les machines qu'il gère à l'aide d'un fichier INI très simple qui place toutes les machines gérées dans des groupes de votre choix. On peut le représenter dans d'autres formats comme YAML.

Pour ajouter de nouvelles machines, aucun serveur de signature supplémentaire n'est nécessaire, alors que NTP ou DNS sont critiques au moment des authentifications.

S'il existe une autre source de confiance dans votre infrastructure, Ansible peut également y ajouter *dynamiquement* des éléments tels que des informations d'inventaire, de groupe et variables à partir de sources telles que EC2, Rackspace, OpenStack, vSphere, etc.

CMDB

Voici à quoi ressemble un fichier d'inventaire en format INI :

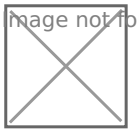
```
[webservers]
www1.example.com
www2.example.com

[dbservers]
db0.example.com
db1.example.com
```

Une fois que les hôtes d'inventaire sont répertoriés, des variables "d'inventaire", c'est-à-dire portant sur les cibles de gestion, peuvent leur être attribuées dans des fichiers texte simples (dans un sous-répertoire appelé '`group_vars/`' ou '`host_vars/`') ou directement dans le fichier d'inventaire.

Les Fichiers d'inventaire sont développés plus loin dans le support de formation. Le détail concernant l'inventaire ne sont pas indispensable pour commencer. Toutefois, veuillez retenir que les commandes `ansible`, `ansible-playbook` et `ansible-inventory`, qui permettent d'exploiter Ansible, **exigent toujours un inventaire défini**.

6.6. Modules



Modules et Tasks

Les modules sont "les outils dans la boîte-à-outils".

Ansible fonctionne en se connectant aux noeuds à gérer et en leur envoyant des petits programmes, appelés "modules Ansible". Ces programmes sont écrits pour être des modèles de ressources de l'état souhaité du système. Ansible exécute ensuite ces modules (via SSH par défaut) grâce au protocole JSON sur la sortie standard et les supprime lorsque l'action est terminée.

La bibliothèque de modules peut résider sur n'importe quelle machine et sans aucun serveur central, démon ou base de données. En général, l'administrateur travaille avec son programme de terminal favori, un éditeur de texte et probablement un système de gestion de version (SCM) come Git pour suivre les modifications apportées à son contenu.

Rien n'interdit d'écrire son propre module. Ces modules peuvent contrôler les ressources comme des services, des paquets ou des fichiers (n'importe quoi en réalité), ou encore exécuter des commandes système. [1]

Le document de formation prévoit une initiation à l'usage des modules en mode ad-hoc et dans les livres de jeu.

A partir de la version 2.10, les modules Ansible font partie d'espace nommé comme des collections et se référence selon l'espace nommé qui leur sont réservés.

6.7. Plugins

Les Plugins apportent des fonctionnalités complémentaires à Ansible.

Plugins

On peut connaître tout type de plugins (qui sont des sortes de "modules" spécialisés) :

- `connexions` : ssh, winrm, local, ...
- `inventory` : ini, yaml, scripts, liste, ...
- `become` : su, sudo, enable, runas
- `cache` : Les plugins de cache permettent à Ansible de stocker les faits rassemblés ou d'inventorier les données sources sans avoir à les récupérer à la source.
- Les plugins de `callback` permettent d'ajouter de nouveaux comportements à Ansible lors de la réponse aux événements : horodatage, say, ...
- `strategy` : free, linear, ...
- ...

6.8. Les collections

Les "collections" sont un format de distribution pour du contenu Ansible qui peut inclure des livres de jeu, des rôles, des modules et des plugins. Les "collections" sont distribuées par Ansible Galaxy via `ansible-galaxy collection install <namespace.collection>`. Depuis la version 2.10., les "collections" comprennent les modules tiers au projet `ansible-base`.[2]

7. Exécution des tâches

Une tâche est l'appel à un module Ansible. Le module Ansible contient localement tout le code utile à l'exécution. Il est donc important de disposer du code à jour des modules.

7.1. Le mode Ad Hoc

Le mode Ad-Hoc permet l'exécution de tâches "ad-hoc" parallèles.

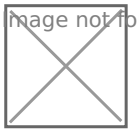
Dès qu'une instance est disponible, on peut lui parler immédiatement, sans aucune configuration supplémentaire, ici sur une instance Red Hat :

```
ansible all -m ping
ansible foo.example.com -m yum -a "name=httpd state=installed"
ansible foo.example.com -a "/usr/sbin/reboot"
```

Un accès à des modules de ressources basés sur des états, ainsi qu'à des commandes "raw" (brutes) est disponible. Ces modules sont assez faciles à écrire. Par ailleurs, Ansible est livré avec énormément de modules déjà développés de telle sorte qu'un bonne partie du travail est déjà réalisée.

Le document reprend de nombreux exemples de l'usage des tâches ad-hoc.

7.2. Playbooks (Livres de jeu)



Playbooks

Les livres de jeu (playbooks) sont écrits selon un langage d'automatisation simple et puissant. Les livres de jeu peuvent orchestrer avec précision plusieurs parties d'une topologie d'infrastructure, avec un contrôle très détaillé du nombre de machines à traiter à la fois.

L'approche d'Ansible en matière d'orchestration est une approche simple et précise : le code d'automatisation devrait être pérenne et il devrait y avoir très peu de choses à retenir sur la syntaxe ou des fonctionnalités spéciales.

Les livres de jeu sont écrits en langage **YAML**, Ain't Markup Language. YAML expose un minimum de syntaxe et propose un modèle de configuration ou de processus plutôt qu'un langage de script ou de programmation.

Chaque livres de jeu est composé de une ou plusieurs "**séances de jeu (plays)**" exposés sous forme d'une liste. Un "livre de jeu" organise des tâches en jeux. Mais il de bonne pratique de l'organiser en "**roles**". Un "role" ajoute un niveau d'abstraction dans l'exécution des tâches d'un livre de jeu.

Un jeu est une analogie sportive qui définit un état ou un modèle et qui se “rejoue” de différentes manières à d’autres moments.

Voici trois exemples de livres de jeu. Le premier exemple montre un livre de jeu avec un seul jeu nommé “DEPLOY VLANS” qui s’applique sur les hôtes “access” d’un inventaire. Celui-ci est constitué d’une seule tâche “ENSURE VLANS EXIST”

```
---
- name: DEPLOY VLANS
  hosts: access
  connection: network_cli
  gather_facts: no
  tasks:
    - name: ENSURE VLANS EXIST
      nxos_vlan:
        vlan_id: 100
        admin_state: up
        name: WEB
```

Le second exemple est un livre de jeu qui lance un seul jeu constitué de cinq tâches.

```
---
- name: Configure webserver with nginx
  hosts: webservers
  become: True
  become_method: sudo
  tasks:
    - name: install nginx
      apt:
        name: nginx
        update_cache: yes
    - name: copy nginx config file
      copy:
        src: files/nginx.conf
        dest: /etc/nginx/sites-available/default
    - name: enable configuration
      file:
        dest: /etc/nginx/sites-enabled/default
        src: /etc/nginx/sites-available/default
```

```
      state: link
  - name: copy index.html
    template:
      src: templates/index.html.j2
      dest: /usr/share/nginx/html/index.html
      mode: 0644
  - name: restart nginx
    service:
      name: nginx
      state: restarted
```

Enfin, voici un livre de jeu qui lance un seul jeu constiué d'un nombre indéterminé de tâches appelées via des rôles : apache, mysql, php, ...

```
---
- name: DEPLOY DRUPAL
  hosts: webserver
  vars_files:
    - vars/main.yml
  roles:
    - apache
    - mysql
    - php
    - php-mysql
    - composer
    - drush
    - drupal
```

7.3. Les rôles

Roles

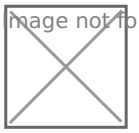
Les rôles (*roles*) permettent de charger automatiquement des *vars_files*, des *tasks*, des *handlers* sur base d'une structure de fichier définie. Les rôles sont intéressants pour leur aspect "ré-utilisable".

Par contre, peu d'éléments permettent d'évaluer la qualité de ces propositions de code "ré-utilisable".

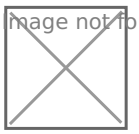
7.4. Ansible Tower

Ansible Tower est un API, un service Web et une console Web conçue pour rendre Ansible utilisable pour les équipes informatiques avec différents profils. Ansible Tower offre une solution complète de gestion des contrôles d'acccès basée sur des rôles. C'est une console centrale de gestion des tâches d'automatisation.

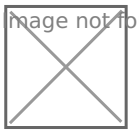
Ansible Tower est un produit commercial pris en charge par Red Hat, Inc. Red Hat a annoncé à l'AnsibleFest 2016 que Ansible Tower passait en Open Source. En 2017, Tower est publié en opensouce sous le nom de AWX. Semaphore est une alternative open source à Ansible Tower, écrit en Go.[3]



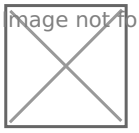
agentless



Aperçu Ansible Tower



Ansible Tower



Roles

Voir AWX.

> Sources des images : License MIT github.com/network-automation/linklight

8. Installation Ansible

8.1. Introduction

On se fonde ici sur la page officielle Ansible Installation Guide. Pour commencer une automation et une gestion avec Ansible, vous avez principalement besoin :

- d'une station Linux et de ses outils
- Python 3
- git
- un éditeur de texte
- un client openssh

Pour la suite des exercices, vous aurez aussi besoin de cibles à gérer. Le document Monter un lab Linux pour Ansible peut vous aider à configurer un lab Ansible avec des machines virtuelles.

Pour connaitre les dernières versions d'Ansible :

```
curl -s https://pypi.org/pypi/ansible-base/json | jq -r '.releases | keys | sort'
```

8.2. Paquets Fedora, RHEL et CentOS

Sur RHEL8, CentOS8 et Fedora :

```
sudo dnf install ansible
```

Sur RHEL7 et CentOS7 :

```
sudo yum install ansible
```

Les fichiers RPMs pour RHEL 7 et RHEL 8 sont disponibles dans le repository Ansible Engine.

Pour l'activer sur RHEL 8 :

```
sudo subscription-manager repos --enable=ansible-2-for-rhel-8-x86_64-rpms
```

Pour l'activer sur RHEL 7 :

```
sudo subscription-manager repos --enable rhel-7-server-ansible-2.9-rpms
```

Pour construire le RPM à partir des sources et l'installer :

```
git clone https://github.com/ansible/ansible.git
cd ./ansible
make rpm
sudo rpm -Uvh ./rpm-build/ansible-*.noarch.rpm
```

8.3. Ubuntu : Dernières versions via Apt

Les programmes Ansible pour sont disponibles en PPA ici.

Pour configurer PPA et installer ansible :

```
sudo apt-get update
sudo apt-get -y install software-properties-common
sudo apt-add-repository -y ppa:ansible/ansible
sudo apt-get update
sudo apt-get -y install ansible
```

Les paquets Debian/Ubuntu packages peut être construits par les sources :

```
make deb
```

8.4. Debian : Dernières versions via Apt

La procédure Debian est indentique à celle Ubuntu PPA.

Ajouter cette ligne au fichier `/etc/apt/sources.list` :

```
deb http://ppa.launchpad.net/ansible/ansible/ubuntu trusty main
```

Exécuter les commandes :

```
sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys 93C4A3FD7BB9C367
sudo apt-get update
sudo apt-get install ansible
```

Note : Cette méthode a été vérifiée avec des sources Trusty sources en Debian Jessie et Stretch mais pourrait ne pas être supportée dans les versions plus récentes.

8.5. Centos : Installation via PIP

```
sudo yum -y remove ansible
sudo yum install -y python3-pip
sudo alternatives --set python /usr/bin/python3
sudo pip3 install ansible --user
```

8.6. Debian/Ubuntu : Installation via PIP

```
sudo apt -y remove ansible
sudo apt update
sudo apt -y install python3-pip
sudo pip3 install --upgrade pip
sudo pip3 install ansible --user
sudo apt -y install sshpass
```

8.7. Windows 10 Pro WSL

Avec une version Windows 10 Pro, il est possible d'utiliser de commander Ansible avec WSL, voir Can Ansible run on Windows? :

Il n'est pas possible de faire fonctionner Ansible sur un hôte Windows alors qu'il peut le gérer. Mais Ansible peut être exécuté sous Windows Subsystem for Linux (WSL). Notons que WSL n'est pas supporté par Microsoft ou Ansible; il ne devrait pas être utilisé sur des systèmes en production.

Pour installer Ansible sur WSL, il est nécessaire d'exécuter ces commandes dans un terminal bash :

```
sudo apt-get update
sudo apt-get install python-pip git libffi-dev libssl-dev -y
pip install ansible pywinrm
```

Pour exécuter Ansible à partir des sources plutôt qu'une version pour WSL, il faut désinstaller la version présente de pip et puis cloner le repo git.

```
pip uninstall ansible -y
git clone https://github.com/ansible/ansible.git
source ansible/hacking/env-setup

# to enable Ansible on login, run the following
echo ". ~/ansible/hacking/env-setup -q' >> ~/.bashrc
```

8.8. Complétion bash pour Ansible

Complétion Bash pour Ansible

La procédure consiste à installer `python-argcomplete` et à le configurer :

```
cat << EOF > ./setup-python-argcomplete.sh
#!/bin/bash
if [[ ! $UID == 0 ]] ; then echo "you must be root" ; exit ; fi
if [[ -f /etc/redhat-release ]] ; then
yum -y install epel-release && yum -y install python-argcomplete
fi
if [[ -f /etc/lsb-release ]] ; then
apt-get update && \
apt-get upgrade --yes --force-yes -o Dpkg::Options::="--force-confdef" -o Dpkg::Options::="--force-confold" && \
apt-get -y install python3-pip && \
pip3 install --upgrade pip && \
pip3 install argcomplete
fi
activate-global-python-argcomplete
EOF
bash -x ./setup-python-argcomplete.sh
```

8.9. Script d’installation automatique

Script d’installation automatique de Ansible pour Centos ou Ubuntu.

```
cat << EOF > ./setup-ansible.sh
#!/bin/bash
```

```
if [[ ! $UID == 0 ]] ; then echo "you must be root" ; exit ; fi
if [[ -f /etc/redhat-release ]] ; then
yum -y remove ansible
yum install -y python3-pip
alternatives --set python /usr/bin/python3
pip3 install ansible --user
pip3 install argcomplete
fi
if [[ -f /etc/lsb-release ]] ; then
apt-get -y remove ansible
apt-get update
apt-get upgrade --yes --force-yes -o Dpkg::Options::="--force-confdef" -o Dpkg::Options::="--force-confold"
apt-get -y install python3-pip
pip3 install --upgrade pip
pip3 install ansible --user
pip3 install argcomplete
apt-get -y install sshpass
fi
echo "PATH=$HOME/.local/bin:$PATH" >> $HOME/.bash_profile
export PATH="$HOME/.local/bin:$PATH"
activate-global-python-argcomplete
ansible --version
EOF
bash -x ./setup-ansible.sh
```

8.10. Image Docker

On propose ici d'utiliser une image de conteneur Docker comme environnement d'exécution d'Ansible. Cette méthode permet de garder l'environnement original indépendant et propre tout en bénéficiant d'un environnement d'exécution stable. On peut même utiliser d'ancienne version d'Ansible au cas où on devrait gérer des noeuds qui ne supporte pas de versions supérieures. On est libre de fabriquer ses images localement et des les adapter.

Installation de docker avec des droits privilégiés :

```
docker_user="ubuntu"
sudo yum -y install curl python3-pip || sudo apt update && sudo apt -y install curl python3-pip
curl -fsSL https://get.docker.com -o get-docker.sh ; sudo sh get-docker.sh
sudo pip3 install docker-compose
```

```
sudo usermod -aG docker ${docker_user}
```

Le projet des images Ansible Cytopia offre des configuration prêtes à l'emploi. Deux exemples simples :

```
sudo docker run --rm cytopia/ansible:latest-tools ansible -m ping localhost
```

Si on veut utiliser la version `2.3` par exemple :

```
sudo docker run --rm cytopia/ansible:2.3-tools ansible --version
```

9. Exécutables Ansible

Ansible vient avec plusieurs programmes. Cela peut sembler absurde mais le fait de prendre connaissance d'emblée de l'environnement complet peut favoriser l'apprentissage. Nous poursuivons toujours un objectif d'un examen de certification sur Ansible : Maîtrise des composants de base d'Ansible.

Références : Working with Command Line Tools.

Programme	Description
`ansible`	programme initial pour l'exécution de commandes ad-hoc
`ansible-config`	Vérifie la configuration courante d'Ansible
`ansible-inventory`	Liste les informations de l'inventaire en format JSON ou YAML
`ansible-doc`	Permet de consulter la documentation hors-ligne
`ansible-playbook`	Permet d'exécuter des livres de jeu
`ansible-vault`	Permet de chiffrer les fichiers qui contiennent des données sensibles
`ansible-galaxy`	Permet de gérer des rôles sur Ansible galaxy
`ansible-console`	Offre une console interactive REPL pour l'exécution de tâches Ad-Hoc
`ansible-pull`	ansible-pull est un petit script qui prend ses informations de configuration d'un repo git et qui exécute un livre de jeu Ansible sur ce contenu
`ansible-test`	Utilitaire de test

ansible est le programme initial pour l'exécution de commandes ad-hoc, ansible-config Vérifie la configuration courante d'Ansible ; ansible-inventory Liste les informations de l'inventaire en format JSON ou YAML ; ansible-doc permet de consulter la documentation hors-ligne ; ansible-playbook permet d'exécuter des livres de jeu ; ansible-vault permet de chiffrer les fichiers qui contiennent des données sensibles ; ansible-galaxy permet de gérer des rôles et des collections sur Ansible galaxy ; ansible-console offre une console interactive REPL pour l'exécution de tâches Ad-Hoc ; ansible-pull ansible-pull est un petit script qui prend ses informations de configuration d'un repo git et qui exécute un livre de jeu Ansible sur ce contenu ; ansible-test est un tilitaire de test.

On ne manquera pas de citer l'outil ansible-lint qui valide des livres de jeu Ansible sur le plan syntaxique.

9.1. Programme ansible

Le programme `ansible` permet d'exécuter des tâches *ad hoc*.

```
ansible --version
```

La commande `ansible` attend toujours un hôte ou un groupe d'hôtes d'inventaire comme argument. Ici une option `-m` désigne l'usage du module `ping`.

```
ansible -m ping localhost
```

Ici l'usage implicite du module `raw` avec un argument :

```
ansible localhost -a "cat /etc/resolv.conf"
```

Le binaire `ansible` comprend de nombreuses options qui seront abordées dans le chapitre sur les exécutions ad hoc :

```
ansible --help
usage: ansible [-h] [--version] [-v] [-b] [--become-method BECOME_METHOD]
               [--become-user BECOME_USER] [-K] [-i INVENTORY] [--list-hosts]
               [-l SUBSET] [-P POLL_INTERVAL] [-B SECONDS] [-o] [-t TREE] [-k]
               [--private-key PRIVATE_KEY_FILE] [-u REMOTE_USER]
               [-c CONNECTION] [-T TIMEOUT]
               [--ssh-common-args SSH_COMMON_ARGS]
               [--sftp-extra-args SFTP_EXTRA_ARGS]
               [--scp-extra-args SCP_EXTRA_ARGS]
```

```
        [--ssh-extra-args SSH_EXTRA_ARGS] [-C] [--syntax-check] [-D]
        [-e EXTRA_VARS] [--vault-id VAULT_IDS]
        [--ask-vault-password | --vault-password-file VAULT_PASSWORD_FILES]
        [-f FORKS] [-M MODULE_PATH] [--playbook-dir BASEDIR]
        [-a MODULE_ARGS] [-m MODULE_NAME]
        pattern

Define and run a single task 'playbook' against a set of hosts

positional arguments:
  pattern               host pattern

optional arguments:
  --ask-vault-password, --ask-vault-pass
                        ask for vault password
  --list-hosts          outputs a list of matching hosts; does not execute
                        anything else
  --playbook-dir BASEDIR
                        Since this tool does not use playbooks, use this as a
                        substitute playbook directory.This sets the relative
                        path for many features including roles/ group_vars/
                        etc.
  --syntax-check        perform a syntax check on the playbook, but do not
                        execute it
  --vault-id VAULT_IDS  the vault identity to use
  --vault-password-file VAULT_PASSWORD_FILES, --vault-pass-file VAULT_PASSWORD_FILES
                        vault password file
  --version             show program's version number, config file location,
                        configured module search path, module location,
                        executable location and exit
  -B SECONDS, --background SECONDS
                        run asynchronously, failing after X seconds
                        (default=N/A)
  -C, --check           don't make any changes; instead, try to predict some
                        of the changes that may occur
  -D, --diff            when changing (small) files and templates, show the
                        differences in those files; works great with --check
  -M MODULE_PATH, --module-path MODULE_PATH
                        prepend colon-separated path(s) to module library (def
```

```
                ault=~/.ansible/plugins/modules:/usr/share/ansible/plu
                gins/modules)
  -P POLL_INTERVAL, --poll POLL_INTERVAL
                set the poll interval if using -B (default=15)
  -a MODULE_ARGS, --args MODULE_ARGS
                module arguments
  -e EXTRA_VARS, --extra-vars EXTRA_VARS
                set additional variables as key=value or YAML/JSON, if
                filename prepend with @
  -f FORKS, --forks FORKS
                specify number of parallel processes to use
                (default=5)
  -h, --help            show this help message and exit
  -i INVENTORY, --inventory INVENTORY, --inventory-file INVENTORY
                specify inventory host path or comma separated host
                list. --inventory-file is deprecated
  -l SUBSET, --limit SUBSET
                further limit selected hosts to an additional pattern
  -m MODULE_NAME, --module-name MODULE_NAME
                module name to execute (default=command)
  -o, --one-line        condense output
  -t TREE, --tree TREE  log output to this directory
  -v, --verbose         verbose mode (-vvv for more, -vvvv to enable
                connection debugging)

Privilege Escalation Options:
  control how and which user you become as on target hosts

  --become-method BECOME_METHOD
                privilege escalation method to use (default=sudo), use
                `ansible-doc -t become -l` to list valid choices.
  --become-user BECOME_USER
                run operations as this user (default=root)
  -K, --ask-become-pass
                ask for privilege escalation password
  -b, --become          run operations with become (does not imply password
                prompting)

Connection Options:
```

```
  control as whom and how to connect to hosts

--private-key PRIVATE_KEY_FILE, --key-file PRIVATE_KEY_FILE
                      use this file to authenticate the connection
--scp-extra-args SCP_EXTRA_ARGS
                      specify extra arguments to pass to scp only (e.g. -l)
--sftp-extra-args SFTP_EXTRA_ARGS
                      specify extra arguments to pass to sftp only (e.g. -f,
                      -l)
--ssh-common-args SSH_COMMON_ARGS
                      specify common arguments to pass to sftp/scp/ssh (e.g.
                      ProxyCommand)
--ssh-extra-args SSH_EXTRA_ARGS
                      specify extra arguments to pass to ssh only (e.g. -R)
-T TIMEOUT, --timeout TIMEOUT
                      override the connection timeout in seconds
                      (default=10)
-c CONNECTION, --connection CONNECTION
                      connection type to use (default=smart)
-k, --ask-pass        ask for connection password
-u REMOTE_USER, --user REMOTE_USER
                      connect as this user (default=None)

Some modules do not make sense in Ad-Hoc (include, meta, etc)
```

9.2. Programme ansible-config

Le programme `ansible-config` vérifie la configuration courante d'Ansible avec une des trois options `list`, `dump` et `view`.

```
ansible-config dump
```

Un chapitre du document examine les options de configuration les plus intéressantes.

9.3. Programme ansible-doc

Le binaire `ansible-doc` permet de consulter la documentation hors-ligne, identique à la documentation en ligne.

Par défaut ansible-doc effectue sa recherche dans les plugins de type “module”. Ici, pour obtenir la liste des modules documentés :

```
ansible-doc -l | grep 'yum'
```

Ici pour lister par exemple les plugins de connexion :

```
ansible-doc -t connection -l
```

Pour afficher l’aide du module “yum” :

```
ansible-doc yum
```

Pour afficher uniqument les arguments du module :

```
ansible-doc yum -s
```

9.4. Programme ansible-playbook

Le programme `ansible-playbook` permet d’exécuter des livres de jeu.

```
ansible-playbook playbook.yml --list-hosts
```

```
ansible-playbook --list-tasks playbook.yml
```

Les options du binaire `ansible-playbook` sont semblables à celles du binaire `ansible`.

9.5. Programme ansible-galaxy

Le programme `ansible-galaxy` permet de gérer des rôles et des “collections”, notamment de modules tiers, avec Ansible galaxy.

```
ansible-galaxy init testrole
```

```
ls testrole/
defaults  files  handlers  meta  README.md  tasks  templates  tests  vars
```

9.6. Programme ansible-inventory

Le programme `ansible-inventory` liste les informations de l'inventaire en format JSON ou YAML.

Avec l'action –list, l'inventaire sort en format JSON :

```
ansible-inventory -i inventory --list
```

Mais il pourrait sortir en format YAML :

```
ansible-inventory -i inventory --list -y
```

Ou sous forme de graphe :

```
ansible-inventory -i inventory --graph
```

9.7. Programme ansible-vault

Le programme `ansible-vault` permet de chiffrer les fichiers qui contiennent des données sensibles.

9.8. Programme ansible-pull

`ansible-pull` est un petit script qui prend ses informations de configuration d'un repo git et qui exécute un livre de jeu Ansible sur ce contenu.

9.9. Programme ansible-console

Le programme `ansible-console` offre une console interactive REPL pour l'exécution de tâches Ad-Hoc.

9.10. Programme ansible-lint

ansible-lint est un utilitaire qui permet de tester la qualité des livres de jeu en proposant des suggestions de bonne pratique.

On installe `ansible-lint` avec `pip` :

```
pip3 install ansible-lint
```

1. Working With Modules ↩ ↩[2]
2. https://github.com/ansible-collections/overview ↩ ↩[2]
3. https://fr.wikipedia.org/wiki/Ansible_(logiciel). ↩ ↩[2]
4. Page README du projet Ansible ↩
5. Gartner, Look Beyond Network Vendors for Network Innovation ↩

Revision #1
Created 3 October 2021 21:03:53 by garfi
Updated 3 October 2021 21:05:29 by garfi