

Déploiement d'applications avec **Ansible et AWX**

Juin 2024

Karim Ayari
Université Claude Bernard Lyon 1

Ansible est une plateforme pour la configuration et la gestion de serveurs. Il s'appuie sur un inventaire structuré ainsi que sur l'exécution de tâches au travers du protocole SSH.

Il ne nécessite donc aucun client !

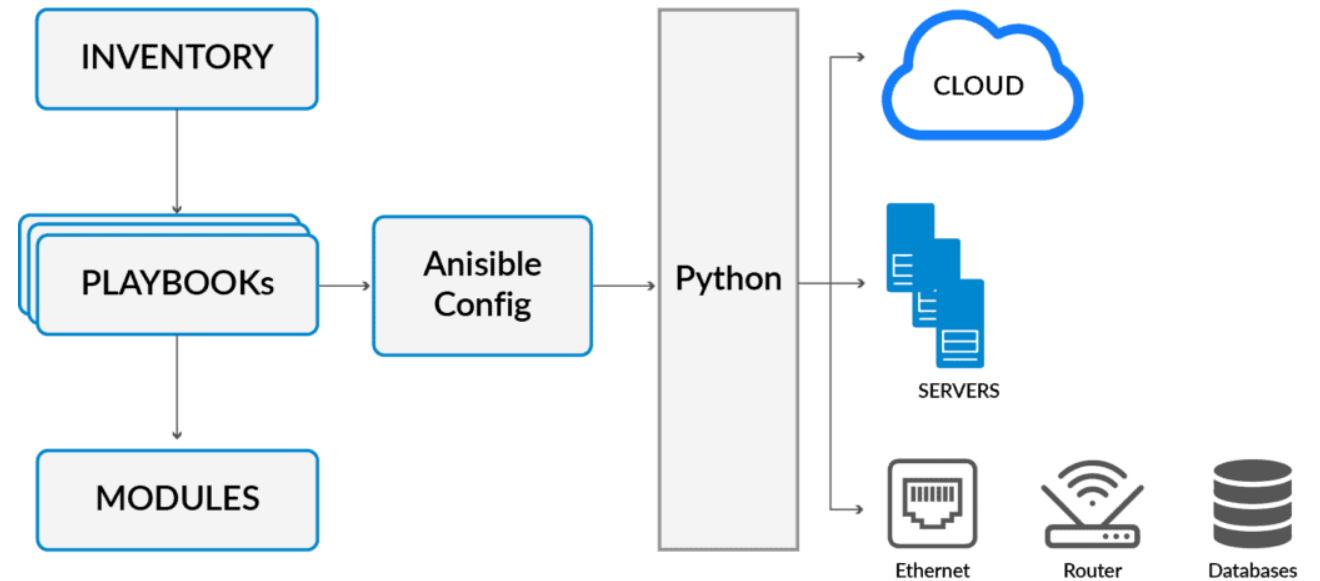


Assurer une homogénéité dans la configuration de vos serveurs ou tout autre équipement pourvu d'un accès ssh ou d'une API.

Déployer des machines virtuelles, des applications, des fichiers de configuration, installation de paquets, création de comptes utilisateurs...



- Par défaut Ansible fonctionne en mode push : c'est le contrôleur qui exécutera vos tâches sur les hôtes de votre inventaire à l'aide de modules Python.
- Il exécutera chaque tâche dans l'ordre, une à la fois, sur toutes les machines correspondant à votre inventaire.
- Il est également capable d'exécuter le code localement sur votre contrôleur selon votre besoin et le type d'équipement ciblé (ex: api vmware)



Tâche: une tâche est l'action qui sera exécutée sur les hôtes. Elles sont exécutées séquentiellement et dans l'ordre établi dans votre playbook.

Playbook: il s'agit de l'ensemble de votre projet, il contient vos tâches, variables, conditions et permettra d'automatiser votre infrastructure.

Inventaire: Il permet de sélectionner les hôtes qui subiront vos tâches. Il peut être manuel (2 formats : **ini et yaml**) ou dynamique.

Modules: programmes sur lesquels s'appuie Ansible pour interagir avec vos systèmes. Ils sont définis dans vos tâches. Par exemple: apt, yum, file, template etc...

Inventaire (format ini)

**Membres du groupe1
et variables d'hôte**

```
[groupe1]
hote1 worker=1
hote2 worker=2
hote3 worker=3 php_upload_max_filesize='4096M' php_post_max_size='4096M'
```

**Membres du groupe2
et variables d'hôte**

```
[groupe2]
hote4 worker=4
```

**Groupe "all" ayant pour sous-groupes
les groupes 1 et 2**

```
[all:children]
groupe1
groupe2
```

Variables spécifiques au groupe1

```
[groupe1:vars]
php_upload_max_filesize='1024M'
php_post_max_size='1024M'
```

Variables spécifiques au groupe2

```
[groupe2:vars]
php_upload_max_filesize='2048M'
php_post_max_size='2048M'
```

Playbook

Hôtes
Récupération des facts

Section tâches

Nom de la 1ere tâche
Module qui sera utilisé
Paramètres du module

2e tâche

3e tâche

```
- hosts: 'all'
gather_facts: no
tasks:
  - name: 'Génération du fichier php.ini'
    template:
      src: cli-php.ini.j2
      dest: /tmp/php.ini
      backup: true
  - name: 'Cat fichier'
    shell: |
      cat /tmp/php.ini
    register: result
  - name: 'Affiche le contenu du fichier'
    debug:
      var: result.stdout_lines
```

Exécution de notre playbook

ansible-playbook playbook.yml -i inventory

En retour vous recevrez le résultat de l'exécution de votre playbook avec un code couleur bien précis.

```
PLAY [all] *****
TASK [Génération du fichier php.ini] *****
changed: [192.168.45.79]
changed: [192.168.45.81]
changed: [192.168.45.80]

TASK [Cat fichier] *****
changed: [192.168.45.81]
changed: [192.168.45.79]
changed: [192.168.45.80]

TASK [Affiche le contenu du fichier] *****
ok: [192.168.45.80] => {
  "result.stdout_lines": [
    "### Fichier géré par Ansible ###",
    "# Template : cli-php.ini.j2",
    "#####",
    "",
    "[PHP]",
    "post_max_size = 1024M",
    "upload_max_filesize = 1024M"
  ]
}
ok: [192.168.45.81] => {
  "result.stdout_lines": [
    "### Fichier géré par Ansible ###",
    "# Template : cli-php.ini.j2",
    "#####",
    "",
    "[PHP]",
    "post_max_size = 1024M",
    "upload_max_filesize = 1024M"
  ]
}
ok: [192.168.45.79] => {
  "result.stdout_lines": [
    "### Fichier géré par Ansible ###",
    "# Template : cli-php.ini.j2",
    "#####",
    "",
    "[PHP]",
    "post_max_size = 4096M",
    "upload_max_filesize = 4096M"
  ]
}

PLAY RECAP *****
192.168.45.79      : ok=3    changed=2    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
192.168.45.80      : ok=3    changed=2    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
192.168.45.81      : ok=3    changed=2    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

Les variables

Les variables vous permettent d'écrire vos tâches, rôles et playbook qui seront réutilisables.

Elles vous aideront à spécifier des différences de configuration et peuvent être définies à différents endroits!

C'est pourquoi il est important de suivre ces principes :

- misez sur la simplicité, même si elles peuvent être définies à différents endroits essayez de limiter le nombre de méthodes et le nombre d'emplacements que vous utilisez pour vos variables.
- dans le cas d'une configuration commune privilégiez l'utilisation des variables de groupe au lieu de définir les variables hôte par hôte.
- dans le cas de gros projets, n'hésitez pas à organiser vos variables dans différents fichiers afin d'en faciliter la recherche, la lecture.

Attention : Ansible applique la priorité des variables

https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_variables.html#understanding-variable-precedence

Le rôle

Le rôle s'appuie sur une structure de fichiers organisée, il vous permet de charger automatiquement les variables, fichiers, tâches, gestionnaires (handlers) Ansible associés en fonction.

Après avoir regroupé votre contenu en rôle, vous pouvez facilement les réutiliser et les partager avec d'autres utilisateurs.

Vous pouvez recréer l'arborescence à la main mais le plus simple est d'utiliser la commande dédiée :

```
# ansible-galaxy role init role_demo
```

Vault

Ansible Vault est une fonctionnalité d'ansible qui vous permet de conserver des données sensibles telles que des mots de passe ou des clés dans des fichiers cryptés dans des playbooks ou des rôles.

ansible-vault encrypt fichier.yml

ansible-vault edit fichier.yml

ansible-vault view fichier.yml

ansible-vault decrypt fichier.yml

```
karim@systemos:~/Public/venv/ansible/demo$ cat info.yml
password: monmotdepasseunpeunul
karim@systemos:~/Public/venv/ansible/demo$ ansible-vault encrypt info.yml
New Vault password:
Confirm New Vault password:
Encryption successful
karim@systemos:~/Public/venv/ansible/demo$ cat info.yml
$ANSIBLE_VAULT;1.1;AES256
33363237316230373566373930323437396532613565373462363130343939306462366361666231
3437633232336631626563303433303131376432353732370a333737643039643061373032333765
32366638643966626537663538396330663635643066643034373063323361633762343066623766
3537663533623261390a346133633035386564386236363961316464653565326336653561623835
366264666665343762623266653431646662626234643763323730366139383133336663030663635
3533313235373463656365633761653933633162326432313961
karim@systemos:~/Public/venv/ansible/demo$
```

```
karim@systemos:~/Public/venv/ansible/demo$ ansible-vault view info.yml
Vault password:
password: monmotdepasseunpeunul
karim@systemos:~/Public/venv/ansible/demo$
```

Pré-requis

- Installer une version d'Ansible sur votre contrôleur (apt, yum, pip...)
- Configurer l'accès ssh avec un compte dédié sur vos hôtes et lui attribuer les droits sudo
- Autorisation de pare-feu vers les hôtes.

Comment bien démarrer ?

- Identifier ce que vous souhaitez automatiser,
- Créer votre inventaire,
- Créer et garder vos playbook aussi simples que possible,
- Utiliser Ansible conjointement à un logiciel de gestion de version comme Git.

Démo

- [Installation avec pip](#)
- [Démo](#)
- [Création d'un rôle](#)

Pour conclure

Ansible est essentiellement un moteur d'état déclaratif. À travers des tâches, les playbooks décrivent l'état dans lequel le système doit être configuré.

On dit qu'il est "idempotent" : l'idempotence désigne toute opération pouvant être exécutée plusieurs fois **sans changer le résultat final** après la première itération.

Le résultat est que vous pouvez exécuter vos playbooks plusieurs fois et le résultat final devrait rester le même: les serveurs ciblés seront dans leur état souhaité.

AWX fournit une interface web, une API REST et un moteur de tâches construits sur Ansible.

Il va simplifier votre utilisation d'Ansible et deviendra la plaque tournante de toutes vos tâches d'automatisation.

C'est la version communautaire et open-source (Apache 2.0) de Red Hat Ansible Automation (ex Ansible Tower).



Son objectif : vous faciliter la vie avec vos projets Ansible

Il embarque nativement Ansible-core qui est une version minimale d'Ansible avec des plugins d'inventaire qui vous permettront de les créer de manière dynamique.

Possibilité d'utiliser des environnements d'exécution différents selon vos projets.

Vous pourrez programmer l'exécution de vos modèles de job et être notifié.

Vous pourrez créer des template de job ou des flux de travail (workflow).

Mais aussi...

Il embarque une gestion de permissions par organisation, par équipe, par utilisateur,

Il va également stocker tous vos credentials de manière sécurisée (clé privée ssh, ansible vault, différents mots de passe...),

Un accès à son API.

Pré-requis

Il s'installe sur un cluster Kubernetes (k8s, k3s, openshift...) à l'aide de l'opérateur awx-operator.

2 déploiements : awx-web pour l'interface et awx-task qui va gérer l'exécution des job

L'installation inclut une base Postgres ainsi qu'un environnement d'exécution par défaut

Il doit avoir un accès à vos serveurs cibles, à vos sources d'inventaire (vmware, openstack...), à votre serveur de gestion de version.

Environnement d'exécution (ee) ?

Il s'agit d'une image de conteneur qui va vous permettre d'incorporer les dépendances utiles à votre projets (collections Ansible, package pip, fichiers).

Chaque ee vous permet d'avoir une image personnalisée pour exécuter les tâches, et chacun d'entre eux contient uniquement ce dont vous avez besoin pour exécuter la tâche, rien de plus.

C'est ansible-buider qui générera votre ee à l'aide d'un fichier de définition (yaml) que vous aurez créé au préalable.

Ensuite c'est ansible-runner qui exécutera vos playbook au sein de cet environnement.

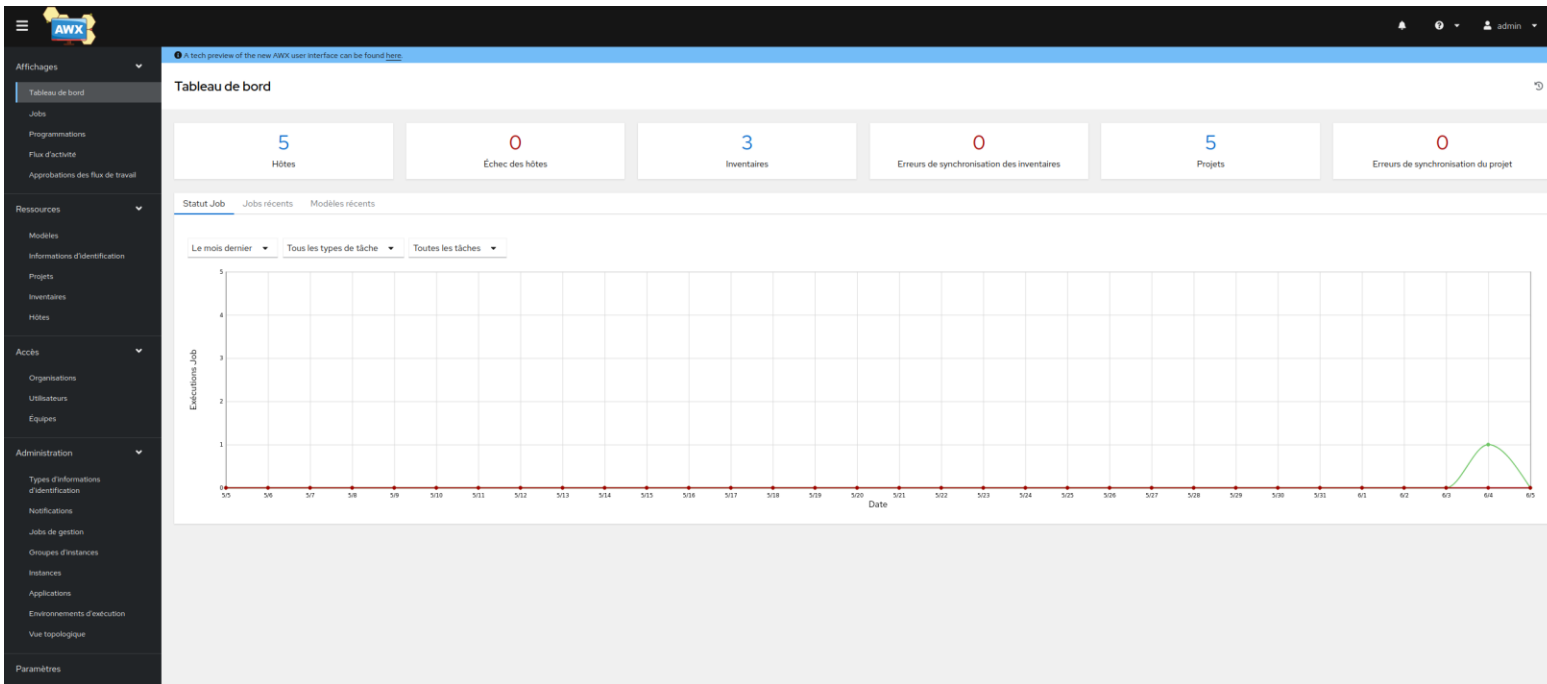


Fonctionnement

- Lors de l'exécution d'un job ou d'une mise à jour d'inventaire, un nouveau conteneur sera créé à partir de l'image de votre environnement d'exécution,
- Ansible-runner exécutera vos playbooks à l'intérieur de ce conteneur,
- Le conteneur est supprimé une fois la tâche terminée,
- Vous aurez le même résultat que si vous aviez exécuté la commande Ansible vous-même.

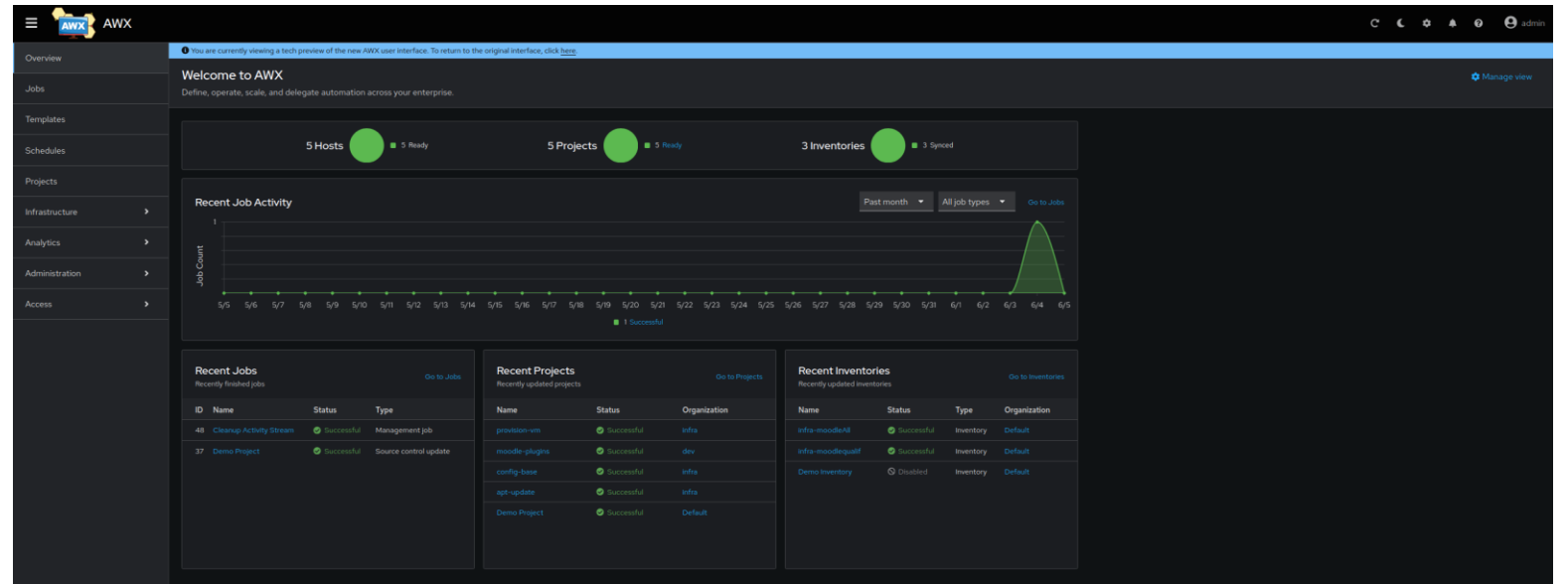
C'est installé, que fait-on maintenant ?

- **Intégrer vos différentes information d'idenfitication:** clé privée, mot de passe ansible-vault etc...
- **Créer votre inventaire:** il peut être manuel (créé directement depuis AWX) ou bien vous allez avoir la possibilité de définir une ou plusieurs sources: un projet Git, VMware VCenter, OpenStack...
- **Importer vos projets:** il s'agit de vos playbook. Ils peuvent être manuels et stockés sur un volume (hostpath, nfs...) ou bien synchronisés avec une source telle que Git (il sera tout de même synchronisé localement).
- **Créer votre modèle de job:** c'est un ensemble de paramètres permettant d'exécuter une tâche Ansible. Il doit être lié à un projet et à un inventaire. Il pourra intégrer un questionnaire ainsi qu'une programmation.



Interface actuelle

Nouvelle interface
(encore en développement)



- ▼ moodle
 - > moodle-data
 - > moodle-infra
 - > moodle-instances
 - > moodle-prod
 - ▼ moodle-qualif
 - ▼ Qualif
 - 📁 prac-mdldbq1
 - 📁 prac-mdlq1**
 - 📁 prac-mdlq2
 - 📁 prac-mdltaskq

LAUNCH WEB CONSOLE

Notes

moodleQualif

- ▼ moodle
 - > moodle-data
 - > moodle-infra
 - > moodle-instances
 - > moodle-prod
 - ▼ moodle-qualif
 - ▼ Qualif
 - 📁 prac-mdldbq1
 - 📁 prac-mdlq1
 - 📁 prac-mdlq2
 - 📁 prac-mdltaskq**

LAUNCH WEB CONSOLE

Notes

moodleTaskQualif

Démonstration (AWX 23.5.1)

- [AWX](#) v23.5.1



icap.ansible 16:07

Job #80 Configuration Linux running: <https://towerhost/#/jobs/playbook/80>



icap.ansible 16:08

Job #80 Configuration Linux successful: <https://towerhost/#/jobs/playbook/80>

```

config-base
├── Project
├── config-base ~/git-repo/config-base
│   ├── roles
│   │   ├── config
│   │   │   ├── defaults
│   │   │   ├── files
│   │   │   ├── handlers
│   │   │   ├── meta
│   │   │   └── tasks
│   │   │       ├── cloud-init.yml
│   │   │       ├── install.yml
│   │   │       ├── main.yml
│   │   │       ├── netplan.yml
│   │   │       └── nftables.yml
│   │   ├── templates
│   │   │   ├── environment.j2
│   │   │   └── wgetrc.j2
│   │   ├── tests
│   │   ├── vars
│   │   └── README.md
│   │   └── users
│   │       ├── defaults
│   │       ├── files
│   │       ├── handlers
│   │       ├── meta
│   │       └── tasks
│   │           ├── dev.yml
│   │           ├── infra.yml
│   │           ├── main.yml
│   │           └── templates
│   │               └── usertpl.yml.j2
│   │           └── vars
│   │               └── README.md
│   │           ├── .gitignore
│   │           ├── base.yml
│   │           └── README.md
│   └── External Libraries
│   └── Scratches and Consoles
└── base.yml
    1  - name: Configuration des utilisateurs pour les serveurs Linux
    2  hosts: '{{ group }}'
    3  gather_facts: 'yes'
    4
    5  roles:
    6  - role: 'users'
    7    tags: 'users'
    8
    9  - role: 'config'
   10  tags: 'config'
  
```

```

config-base > roles > config > tasks > main.yml
├── Project
├── config-base ~/git-repo/config-base
│   ├── roles
│   │   ├── config
│   │   │   ├── defaults
│   │   │   ├── files
│   │   │   ├── handlers
│   │   │   ├── meta
│   │   │   └── tasks
│   │   │       ├── cloud-init.yml
│   │   │       ├── install.yml
│   │   │       ├── main.yml
│   │   │       ├── netplan.yml
│   │   │       └── nftables.yml
│   │   ├── templates
│   │   │   ├── environment.j2
│   │   │   └── wgetrc.j2
│   │   ├── tests
│   │   ├── vars
│   │   └── README.md
│   │   └── users
│   │       ├── defaults
│   │       ├── files
│   │       ├── handlers
│   │       ├── meta
│   │       └── tasks
│   │           ├── dev.yml
│   │           ├── infra.yml
│   │           ├── main.yml
│   │           └── templates
│   │               └── usertpl.yml.j2
│   │           └── vars
│   │               └── README.md
│   │           ├── .gitignore
│   │           ├── base.yml
│   │           └── README.md
│   └── External Libraries
│   └── Scratches and Consoles
└── base.yml
    1  ---
    2  # tasks file for install
    3  - name: Install files
    4    include_tasks: install.yml
    5
    6  - name: Netplan configuration
    7    include_tasks: netplan.yml
    8
    9  - name: Install nftables
   10    include_tasks: nftables.yml
   11
   12  - name: Disable cloud-init
   13    include_tasks: cloud-init.yml
  
```

Merci de votre attention !

<https://www.ansible.com/>

<https://docs.ansible.com/>

<https://forum.ansible.com/>

<https://galaxy.ansible.com/>

ansible-doc *nom_du_module*

Doc AWX

<https://ansible.readthedocs.io/projects/awx/en/latest/userguide/index.html>

Template Kustomize pour installer AWX :

<https://github.com/kurokobo/awx-on-k3s>