II.2. Projet Ansible

Ce document est une description du projet Ansible à lire en regard du document de présentation sur l'Infrastructure as Code.

1. Projet Ansible

Ansible est une plate-forme logicielle pour la configuration et la gestion des ordinateurs. Le logiciel combine le déploiement de logiciels multi-noeuds, l'exécution des tâches ad-hoc, et la gestion des configurations.

Il gère les différents noeuds avec un accès à distance natif (tels que les protocoles SSH ou Remote PowerShell ou encore des APIs natives) et ne nécessite l'installation d'aucun logiciel supplémentaire à distance, avec 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. [2]

[2] Page Ansible (software) sur Wikipedia EN)

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 [1].

[1] Gartner, Look Beyond Network Vendors for Network Innovation

3. Version d'Ansible

Ansible est développé et publié avec un cycle de révision de 4 mois. Ce cycle peut être étendu afin de permettre la mise en oeuvre de modifications et de test correctifs. Ansible a une structure de support graduelle qui s'étend à trois versions principales.

https://docs.ansible.com/ansible/latest/reference*appendices/release*and_maintenance.html

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

```
curl -s https://releases.ansible.com/ansible/ansible-latest.tar.gz.sha | \
awk '{ print $2 }' | \
sed 's/\.tar.*//'
```

4. Objectifs de conception de Ansible

Les objectifs de conception de Ansible comprennent[5] :

- **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 jeux 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.

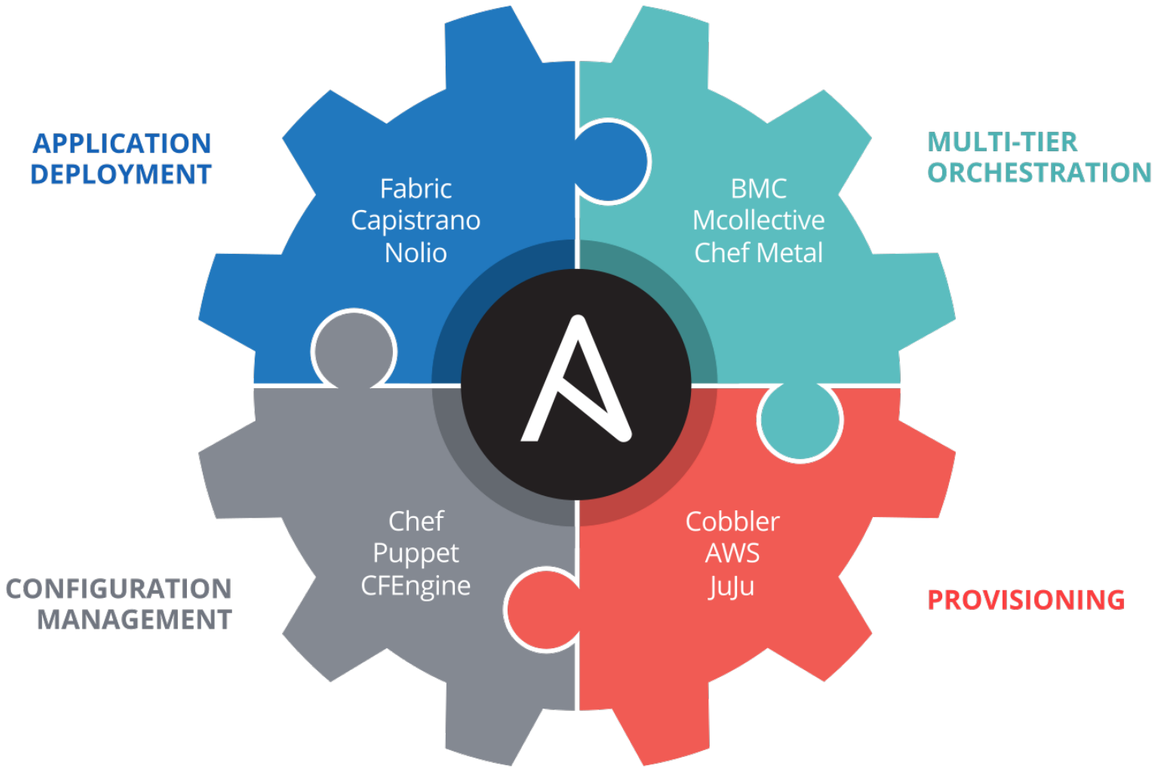
---

[5] Page README du projet Ansible

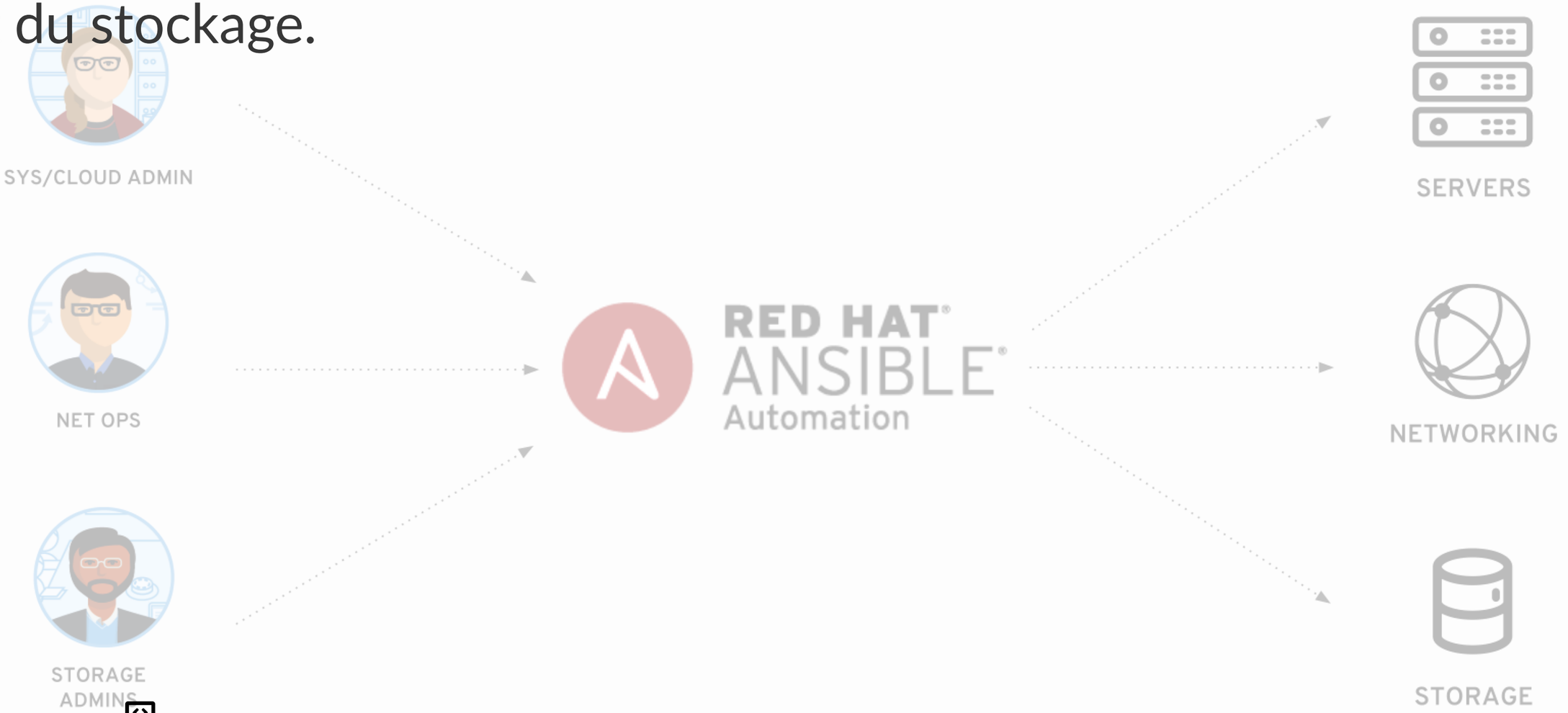
5. Automation d'infrastructures

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

- Le déploiement d'applications (Fabric, Capristano, Nolio)
- L'orchestration multi-tiers (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.



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.

Virtualisation

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

Systèmes d'exploitation

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

Containers

- Ansible Container
- Docker
- Linux Containers (LXC)

Cloud

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

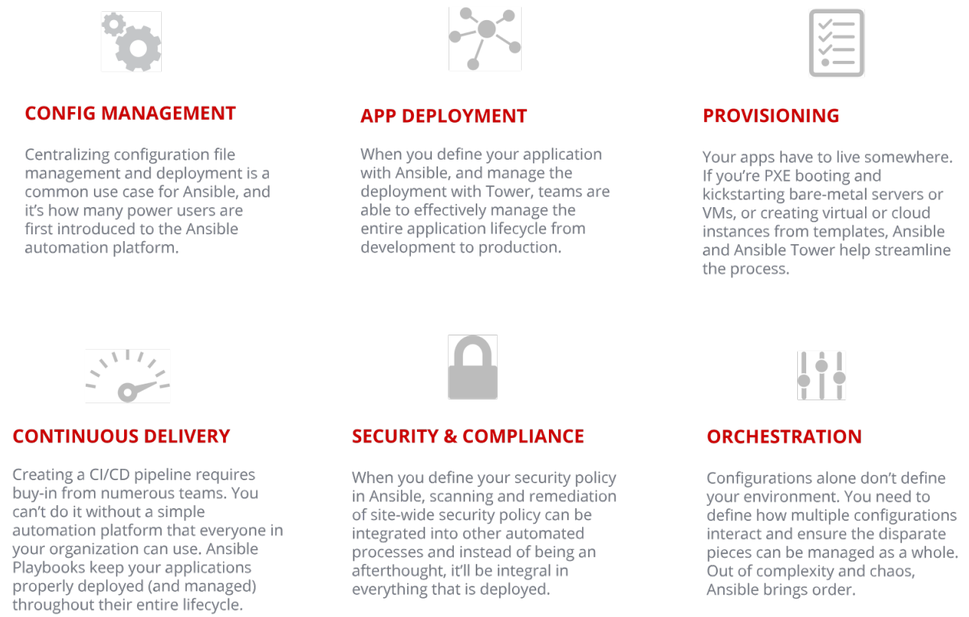
Outils DevOps

- 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
  - ...

6. Cas d'usage

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



7. 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.[1]

[1] Gartner, Look Beyond Network Vendors for Network Innovation

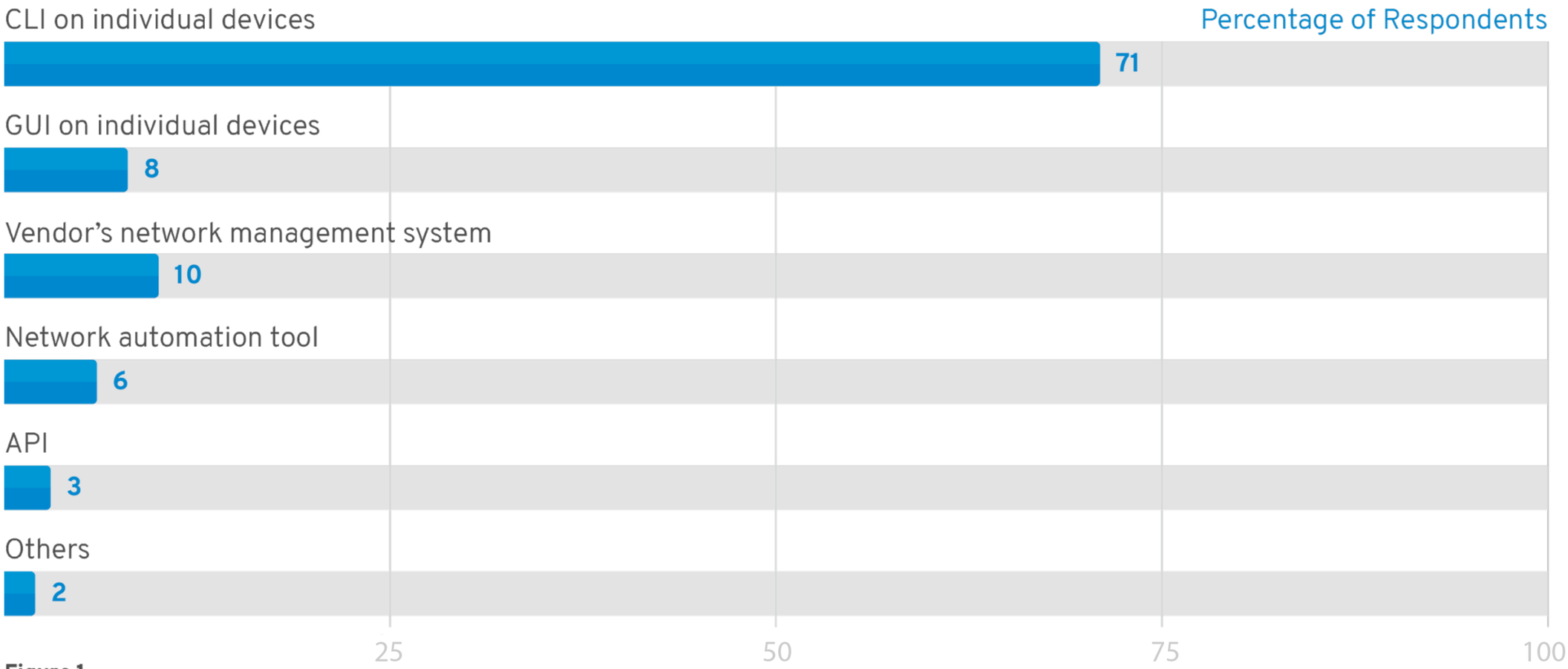
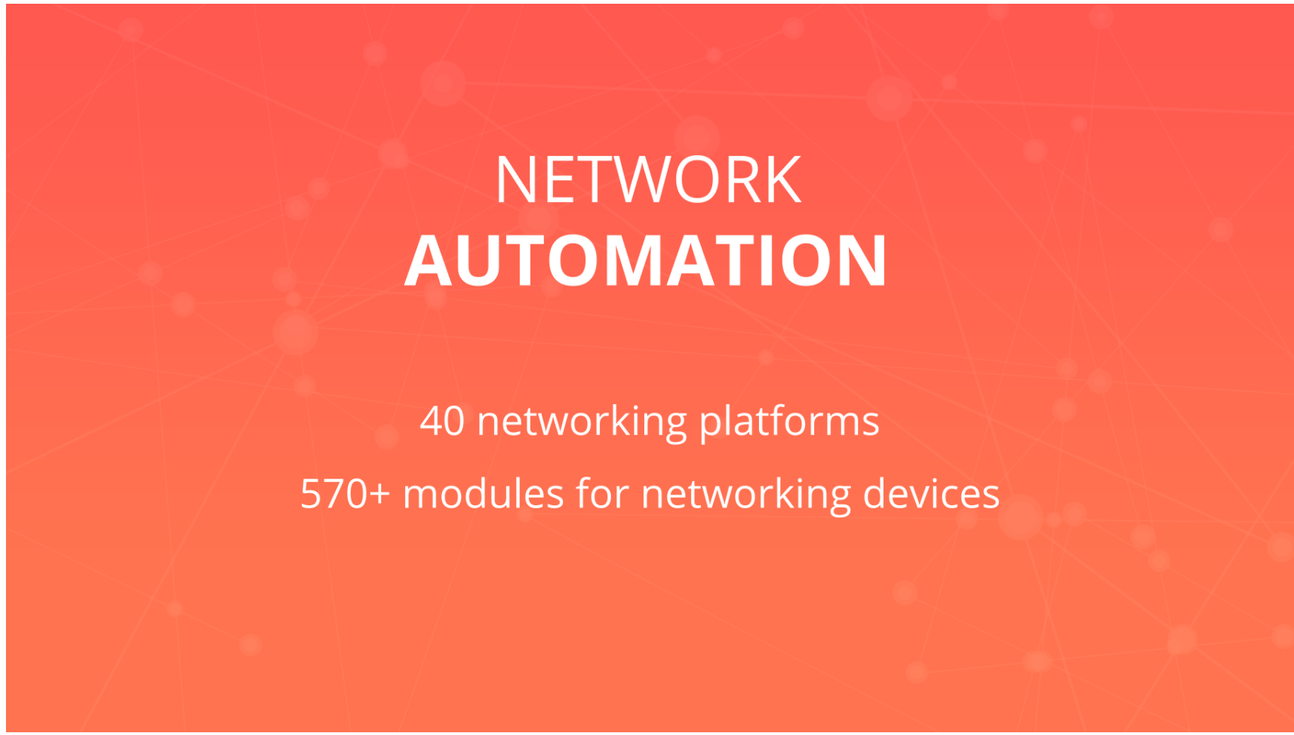


Figure 1
Primary Method for Making Network Changes

Source: Gartner, *Look Beyond Network Vendors for Network Innovation.* January 2018. Gartner ID: G00349636. (n=64)

Ansible peut gérer des périphériques :

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



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.

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

Meilleures pratiques

https://docs.ansible.com/ansible/2.5/network/user*guide*/*network*best*practices*2.5.html

II. Solution Ansible

Introduction

Ce document décrit la terminologie et les composants Ansible. 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, 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.

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

1. Comprendre les composants de base d'Ansible

    - Inventaires
    - Modules
    - Variables
    - "Facts"
    - Jeux
    - Playbooks
    - Fichiers de configuration

Nous allons progressivement apprendre que manipuler Ansible consiste à écrire du texte géré sous forme code informatique. L'ambition est alors de "coder l'infrastructure". Nous sommes à mi-chemin entre une solution de gestion unique et intégrée basée sur un outil facile mais très limité comme Bash et une automation en Python exigeant une longue courbe d'apprentissage pour cetains profils.

Ansible vient avec l'avantage d'être :

- populaire
- flexible
- évolutif
- lisible par tous
- personnalisable
- intégrable
- à courbe d'apprentissage faible

Le seul écueil est de connaître les solution à gérer à travers cet outil 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 qui fonctionne avec état de manière très simple) qui intègre d'autres solutions ou s'intègre à d'autres facilement. Ansible Tower et son fork OpenSource AWX renforce cette capacité de manière crédible.

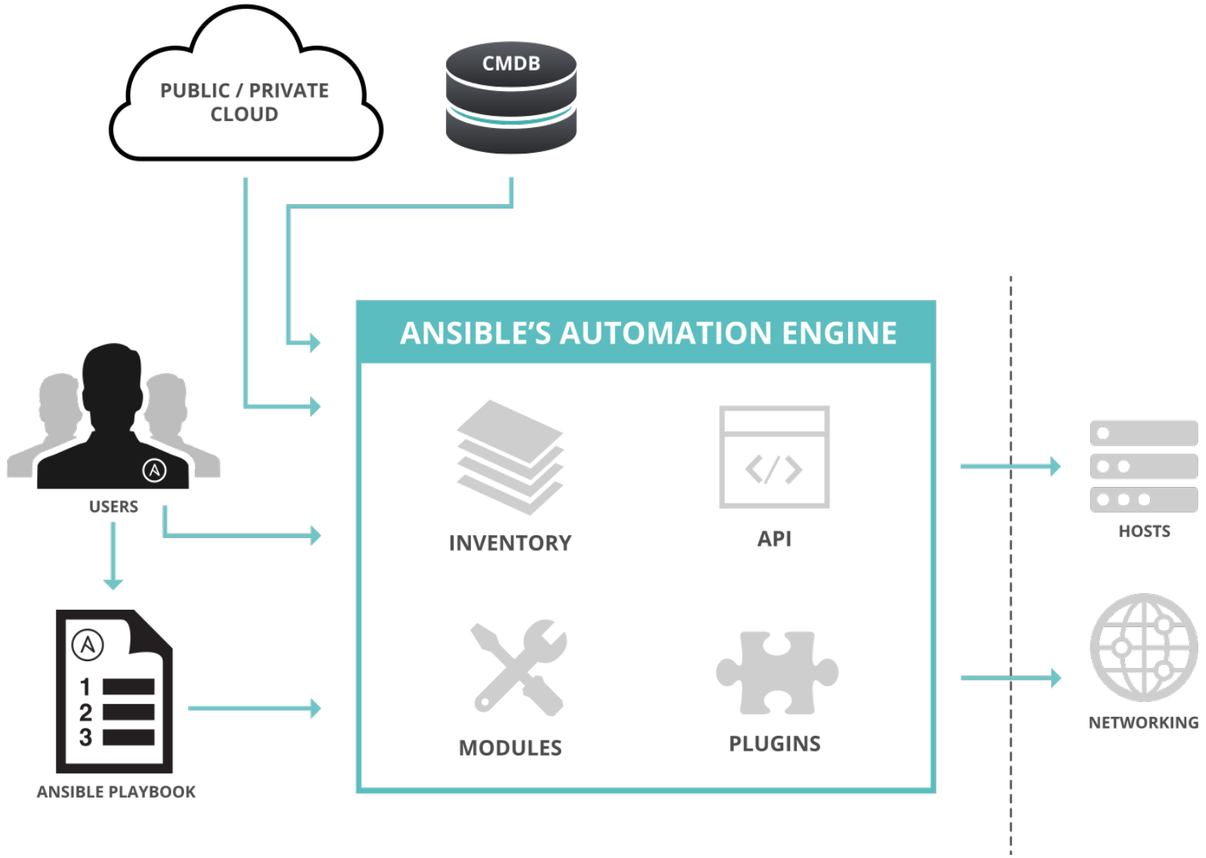
1. Terminologie Ansible

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)

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



2. 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?

La section Installation Ansible indique de manière plus détaillée l'installation de Ansible sur la machine de contrôle.

3. Noeuds gérés

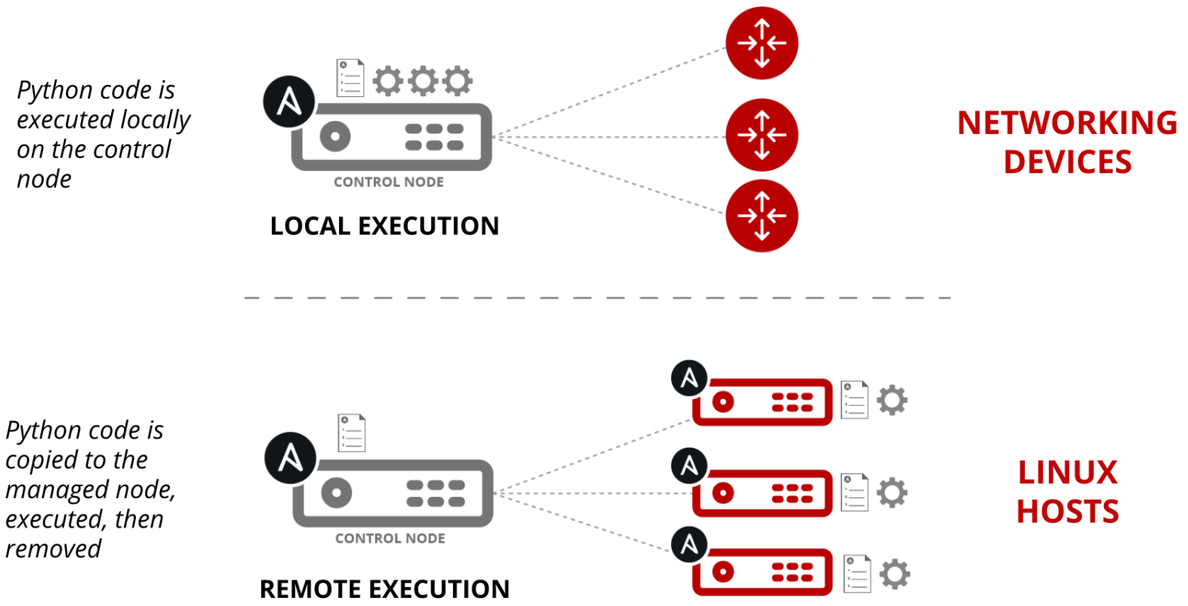
Les noeuds gérés, s'ils sont en Linux/Unix, doivent disposer de Python 2.6 ou version ultérieure, **mais de préférence aujourd'hui le pré-requis est Python 3**. SSH est alors le mode de connexion préféré.

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 API HTTP REST.

On trouvera ici un propos sur les Plugins de connexion.

Source : How Network Automation is Different



Clés SSH

Machine de contrôle et 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 quelles machines peuvent accéder à quels hôtes.

```
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 suppoter des connexions SSH.

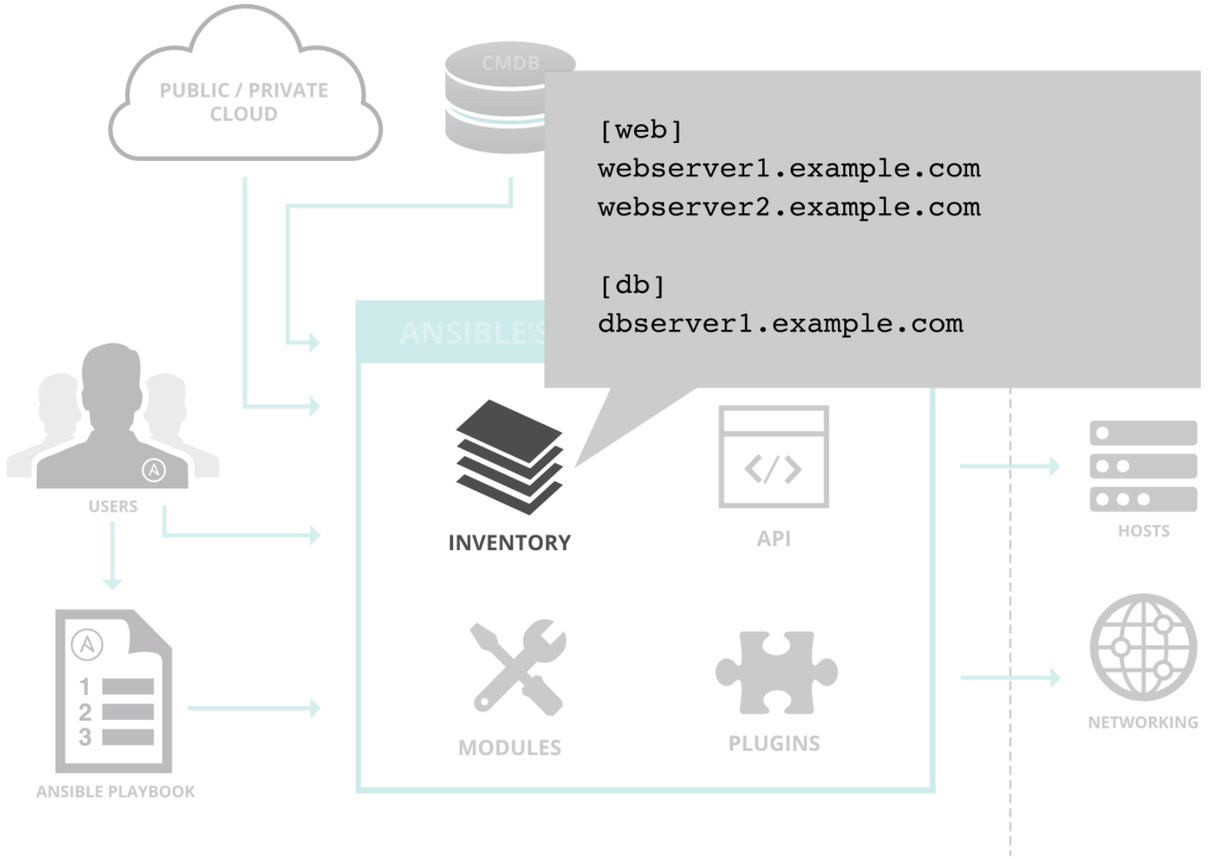
Ansible supporte beaucoup d'autres types de connexions/ communications avec ses cîbles.

4. L'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.

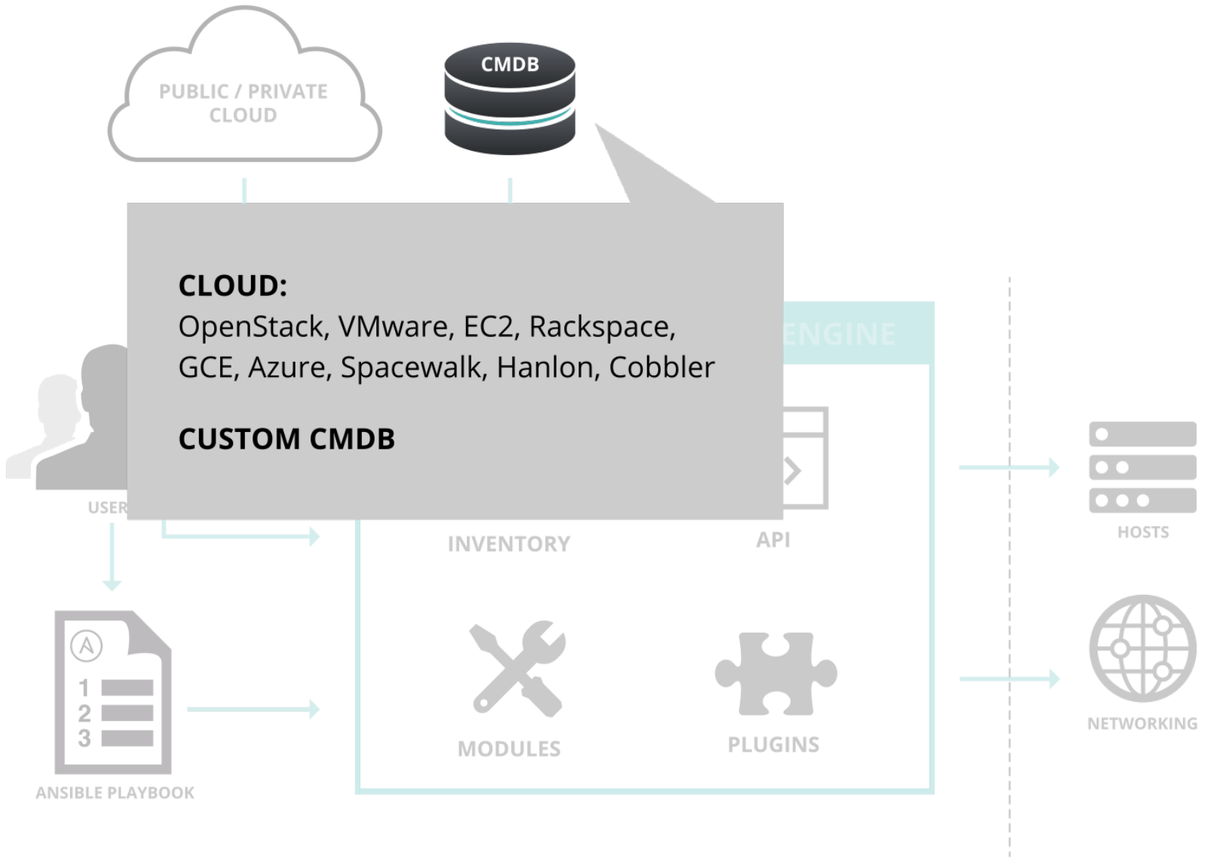
S'il existe une autre source de confiance dans votre infrastructure, Ansible peut également y ajouter des éléments tels que des informations d'inventaire, de groupe et variables à partir de sources telles que EC2, Rackspace, OpenStack, etc.



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 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.

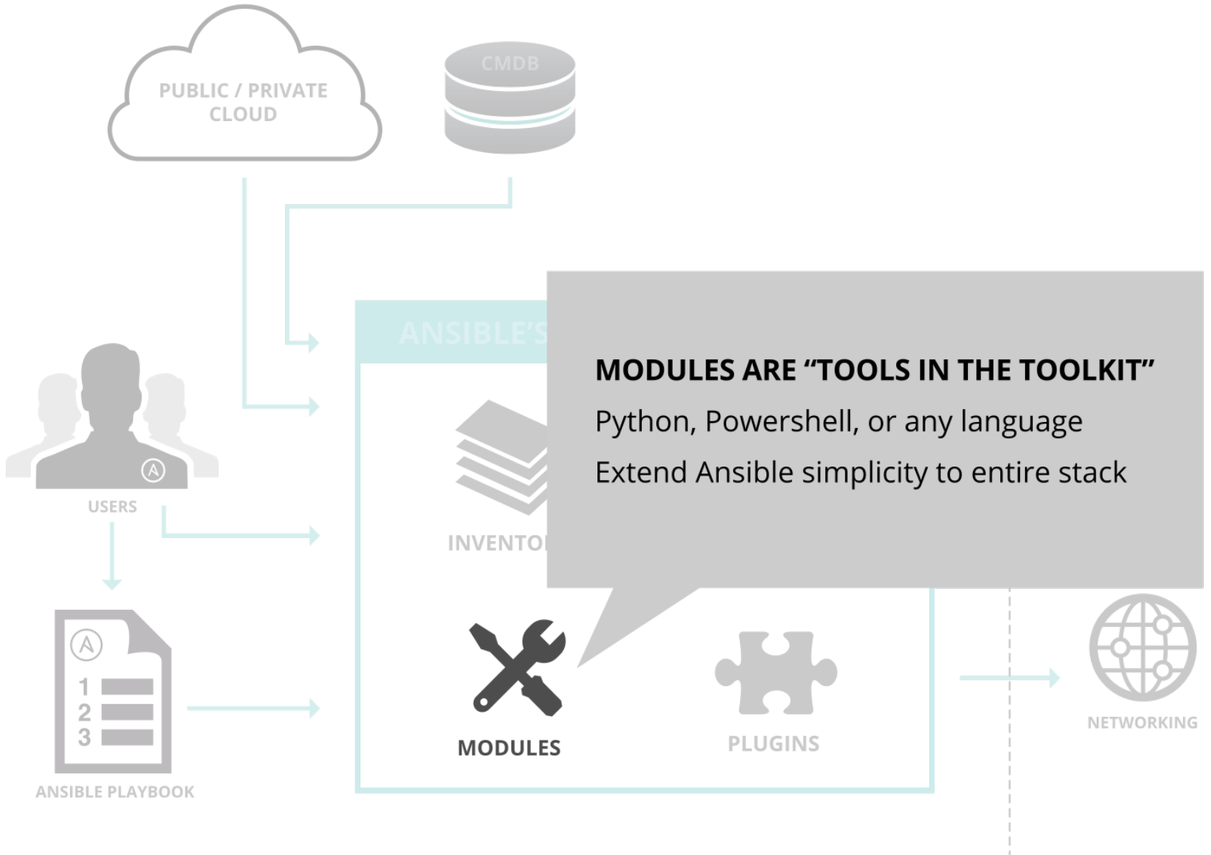
Aussi, on peut utiliser un inventaire dynamique pour extraire un inventaire de sources de données telles que EC2, Rackspace ou OpenStack.

Les Fichiers d'inventaire sont développés plus loin dans le support de formation.

5. Modules

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 contrôle 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. [6]

!!! Découverte des modules

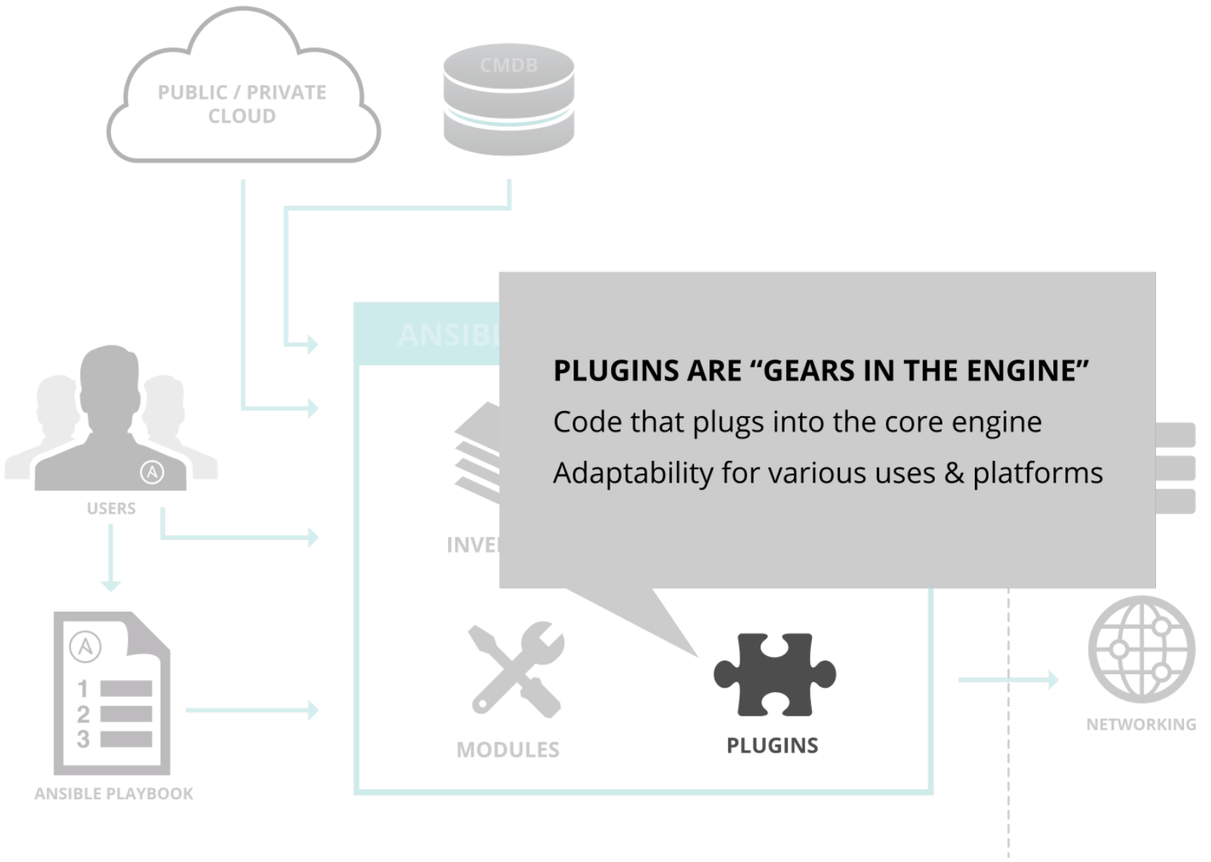
---

[6] Working With Modules

6. Plugins

Les Plugins apportent des fonctionnalités complémentaires à Ansible.

!!! Types de plugins



7. Exécution des tâches

Concept de tâche. --> module local. Tout le code utile à l'exécution est local --> importance des màj.

- Ad-Hoc
- Livre de jeux
- Roles
- Ansible Tower

8. Le mode Ad Hoc

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

Dès qu'une instance est disponible, on peut lui parler immédiatement, sans aucune configuration supplémentaire, ici sur une instance Red Ha t:

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

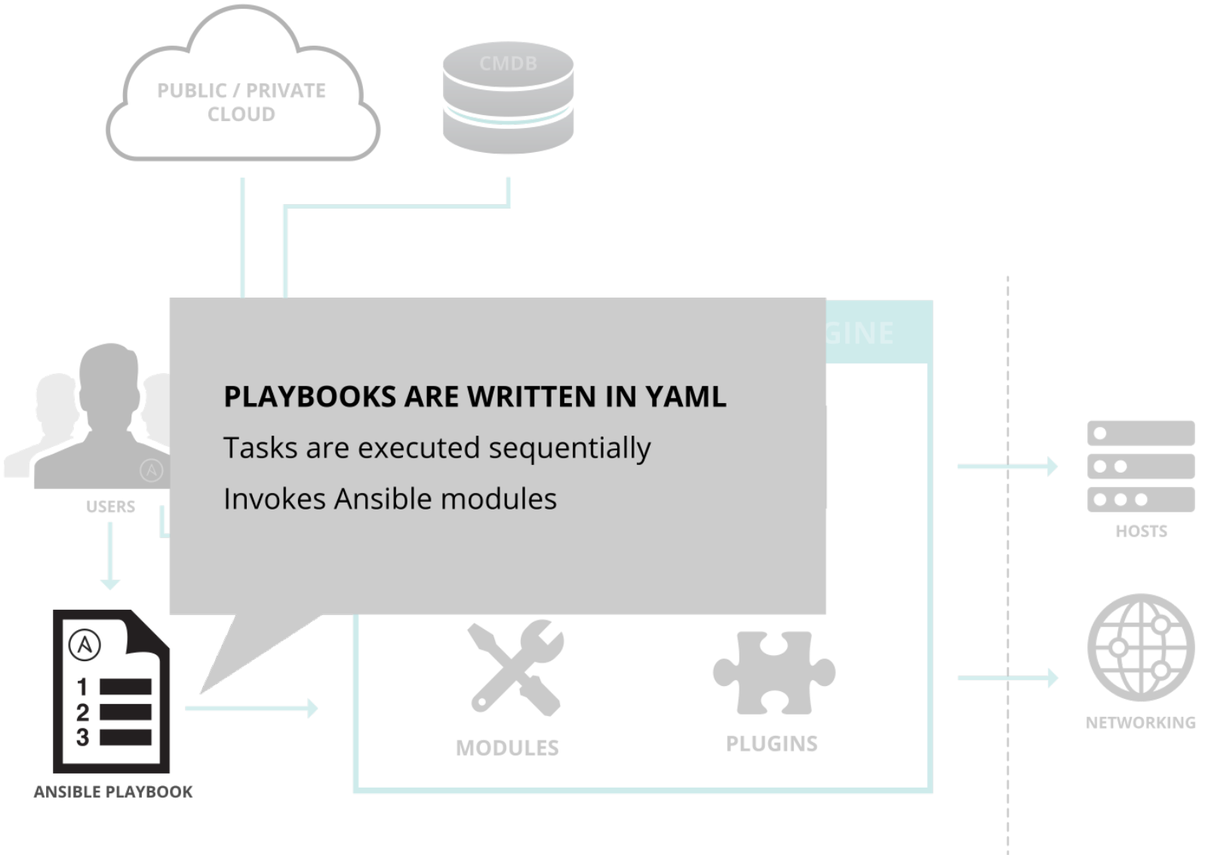
Un accès à des modules de ressources basés sur des états, ainsi qu'à des commandes "raw" est disponible. Ces modules sont extrêmement faciles à écrire. Par ailleurs, Ansible est livré avec énormément de modules déjà développés (plus de 750) de telle sorte qu'un bonne partie du travail est déjà réalisée.

!!! Mode Ad-Hoc

9. Playbooks (Livres de jeu)

Les livres de jeu (playbooks) sont écrits selon un langage d'automatisation simple et puissant. Les Playbooks 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 (playbooks) 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 un ou plusieurs "**séances de jeu (plays)**" dans une liste. Un "livre de jeu" organise des tâches. Mais il de bonne pratique de l'organiser en "**roles**". Un "role" ajoute un niveau d'abstraction dans l'exécution 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 indérminé 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
```

10. Les rôles

Roles

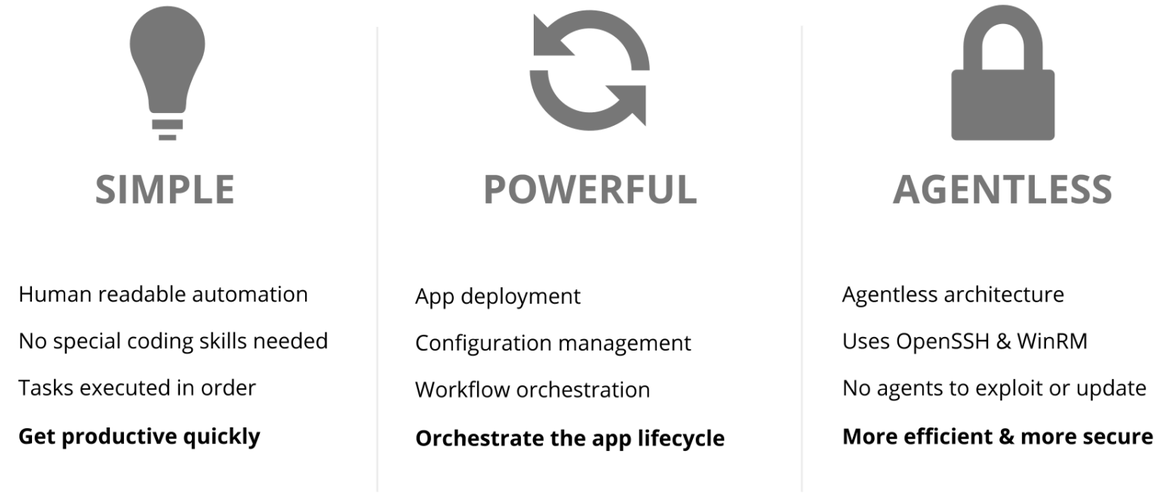
Les rôles (*roles*) permettent de charger automatiquement des *vars*files, *des _tasks*, des *handlers* sur base d'une structure de fichier définie. Les rôles sont intéressants pour leur aspect "ré-utilisable".

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

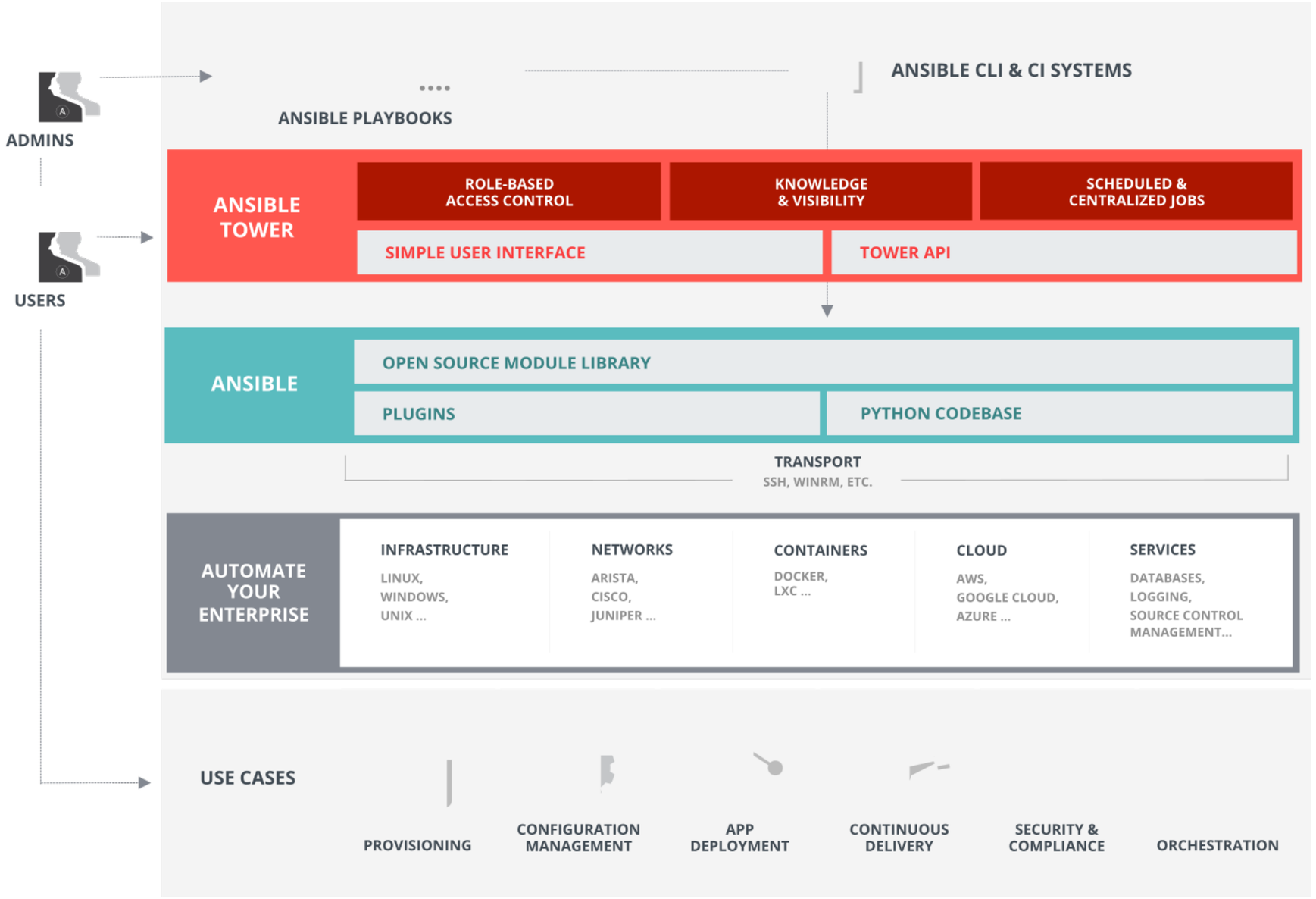
11. Ansible Tower

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 dispose de toute une gestion des contrôle d'acccès basé sur des rôles. C'est une console centrale de gestion des tâches d'automatisation. Tower est un produit commercial pris en charge par Red Hat, Inc. Red Hat a annoncé à l'AnsibleFest 2016 que Tower passait en Open Source. En 2017, Tower est publié en opensoucé sous le nom de AWX. Semaphore est une alternative open source à Ansible Tower, écrit en Go.[3]

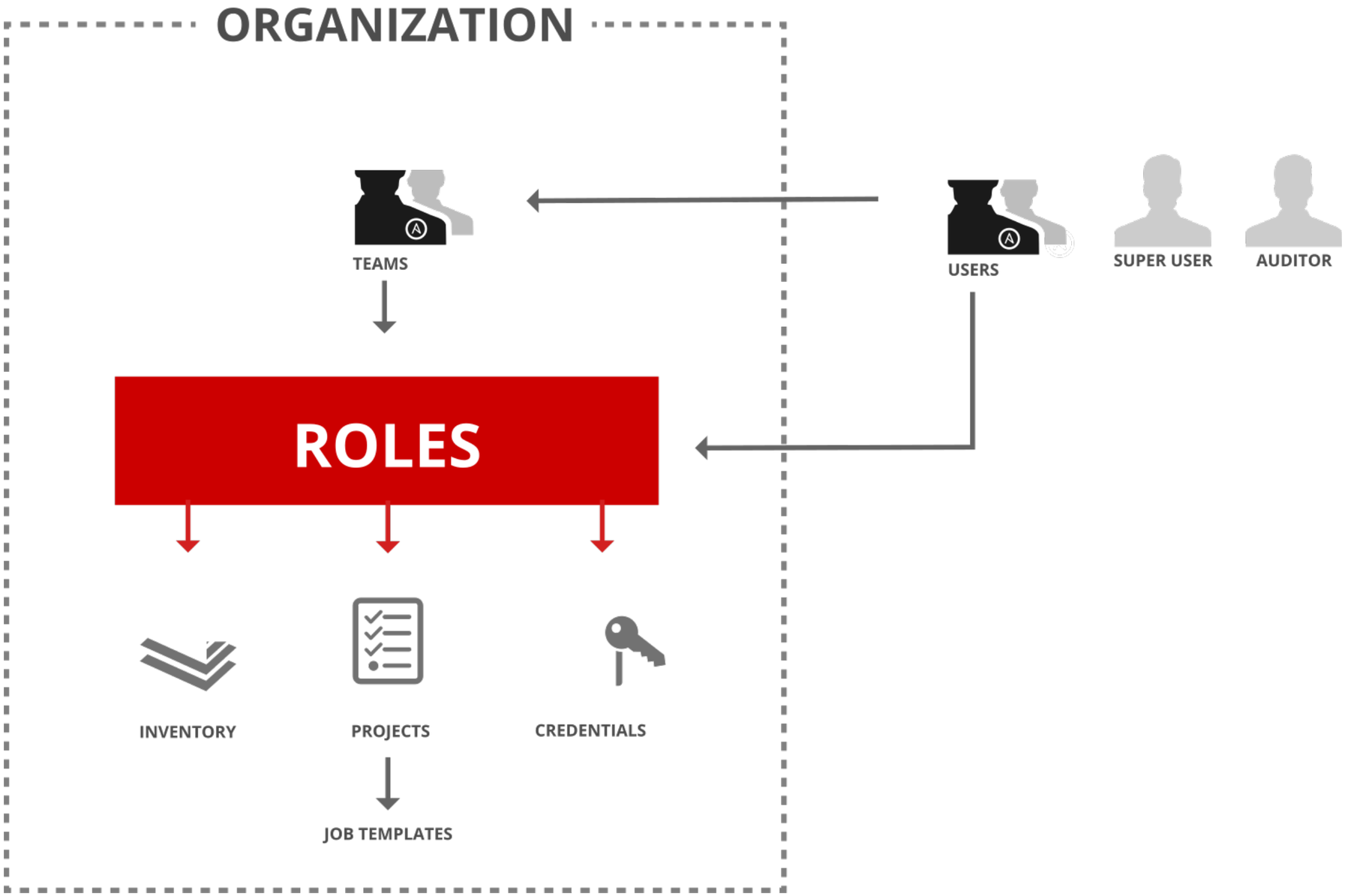
Voir AWX.



[3] https://fr.wikipedia.org/wiki/Ansible_(logiciel)).


ADMINS
USERS
ANSIBLE PLAYBOOKS
ANSIBLE CLI & CI SYSTEMS
ANSIBLE TOWER
ROLE-BASED ACCESS CONTROL
KNOWLEDGE & VISIBILITY
SCHEDULED & CENTRALIZED JOBS
SIMPLE USER INTERFACE
TOWER API
ANSIBLE
OPEN SOURCE MODULE LIBRARY
PLUGINS
PYTHON CODEBASE
TRANSPORT
SSH, WINRM, ETC.
AUTOMATE YOUR ENTERPRISE
INFRASTRUCTURE
LINUX, WINDOWS, UNIX ...
NETWORKS
ARISTA, CISCO, JUNIPER ...
CONTAINERS
DOCKER, LXC ...
CLOUD
AWS, GOOGLE CLOUD, AZURE ...
SERVICES
DATABASES, LOGGING, SOURCE CONTROL MANAGEMENT...
USE CASES
PROVISIONING
CONFIGURATION MANAGEMENT
APP DEPLOYMENT
CONTINUOUS DELIVERY
SECURITY & COMPLIANCE
ORCHESTRATION


RED HAT®
ANSIBLE®
Automation
RED HAT ANSIBLE TOWER
Scale + operationalize your automation
CONTROL
KNOWLEDGE
DELEGATION
RED HAT ANSIBLE ENGINE
Support for your Ansible automation
SIMPLE
POWERFUL
AGENTLESS
FUELED BY AN INNOVATIVE OPEN SOURCE COMMUNITY


ORGANIZATION
TEAMS
USERS
SUPER USER
AUDITOR
ROLES
INVENTORY
PROJECTS
CREDENTIALS
JOB TEMPLATES

https://iac.goffinet.org/solution-ansible/