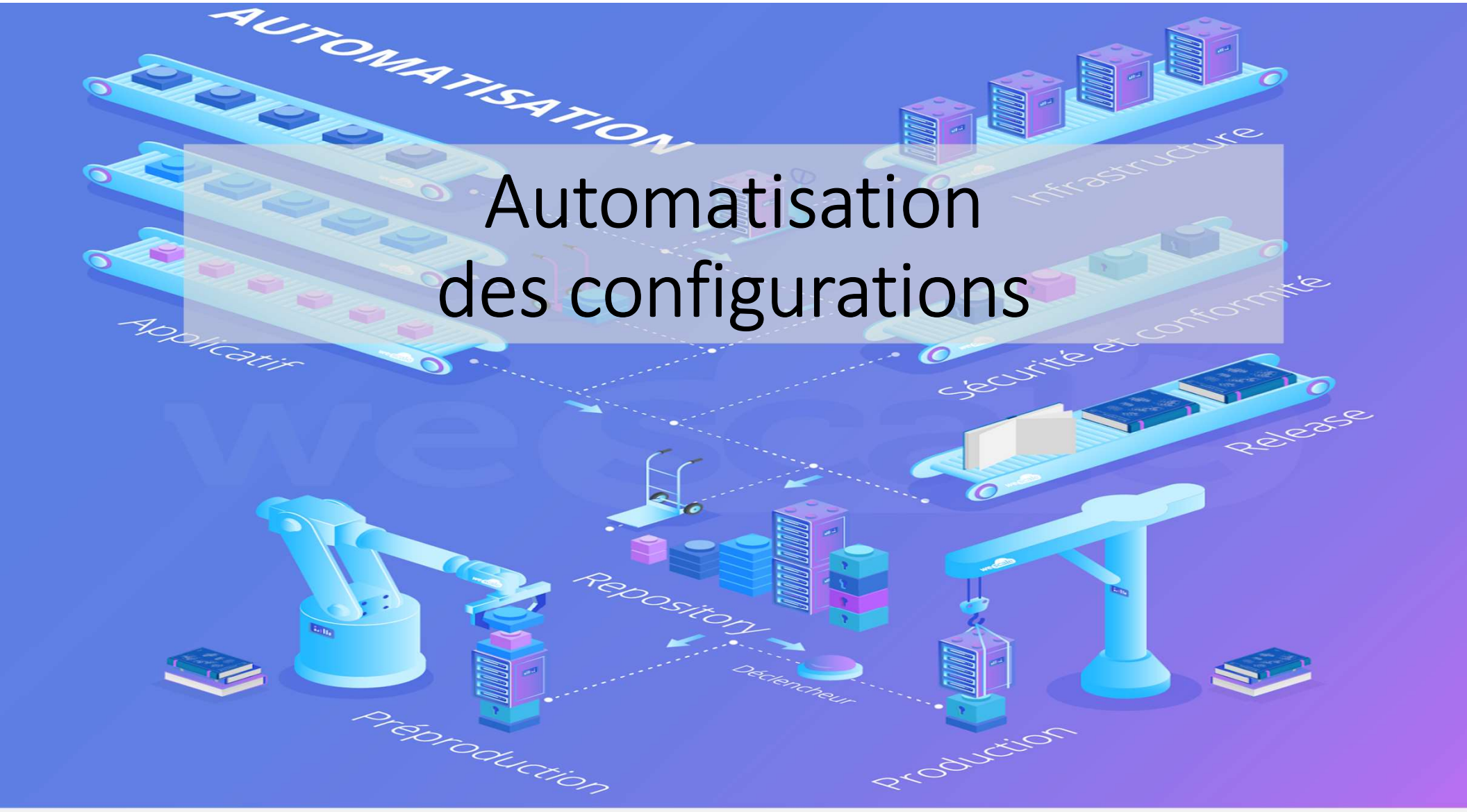


Automatisation des configurations

- Objectif :
  - Effectuer le déploiement de serveurs Intranet et Internet
    - Automatiser le déploiement de serveurs Intranet et Internet

Que font les outils d'automatisation pour nous ?

Les outils d'automatisation offrent des fonctionnalités puissantes par rapport aux stratégies d'automatisation ad hoc utilisant BASH, Python ou d'autres langages de programmation. Ces outils permettent aux développeurs de :

- Simplifier et uniformiser
- Accélérer le développement grâce à des fonctionnalités prêtes à utiliser
- Faciliter la réutilisation, séparer les responsabilités et promouvoir la sécurité
- Effectuer la découverte et la gestion d'inventaire
- Gère la croissance (Handle scale)
- Mobiliser la communauté

Concepts critique

Idempotence : un aperçu

- Un logiciel Idempotent produit le même résultat souhaitable chaque fois qu'il est exécuté.
- Dans un logiciel de déploiement, Idempotence permet la convergence et la composabilité et permet de :
  - Rassembler plus facilement des composants dans des collections qui créent de nouveaux types d'infrastructure et effectuent de nouvelles tâches opérationnelles.
  - Exécuter des collections complètes de développement/déploiement pour réparer en toute sécurité les petits problèmes d'infrastructure, effectuer des mises à niveau progressives, modifier la configuration ou gérer la mise à l'échelle.

Procédure vs déclarative

- Le code procédural peut atteindre l'idempotence, mais de nombreux outils de gestion de l'infrastructure, de déploiement et d'orchestration ont adopté une autre méthode, qui consiste à créer un déclaratif.
- Un déclaratif est un modèle statique utilisé par le middleware qui intègre des détails spécifiques au déploiement, examine les circonstances actuelles et met l'infrastructure réelle en alignement avec le modèle, et généralement le chemin le moins long.

Concepts critique

Provisionnement vs. configuration vs. déploiement vs. orchestration

Provisionnement	Configuration	Déploiement	Orchestration
Il s'agit d'obtenir une infrastructure de calcul, de stockage et de réseau (réelle ou virtuelle), d'activer les communications, de la mettre en service et de la rendre prête à l'emploi par les opérateurs et les développeurs.	Cela signifie installer des applications et des services de base et effectuer les opérations, les tâches et les tests nécessaires pour préparer une plate-forme de bas niveau pour déployer des applications ou une plate-forme de niveau supérieur.	Cela implique la création, l'organisation, l'intégration et la préparation d'applications multi-composants ou de plates-formes de niveau supérieur, souvent sur plusieurs nœuds.	Cela peut se référer à plusieurs choses : • Automatisation construite par l'utilisateur ou inhérente à la plate-forme visant à gérer les cycles de vie des charges de travail et à réagir dynamiquement aux conditions changeantes. • Des processus ou des workflows qui relient les tâches d'automatisation pour offrir des avantages commerciaux, tels que le libre-service (self-service).

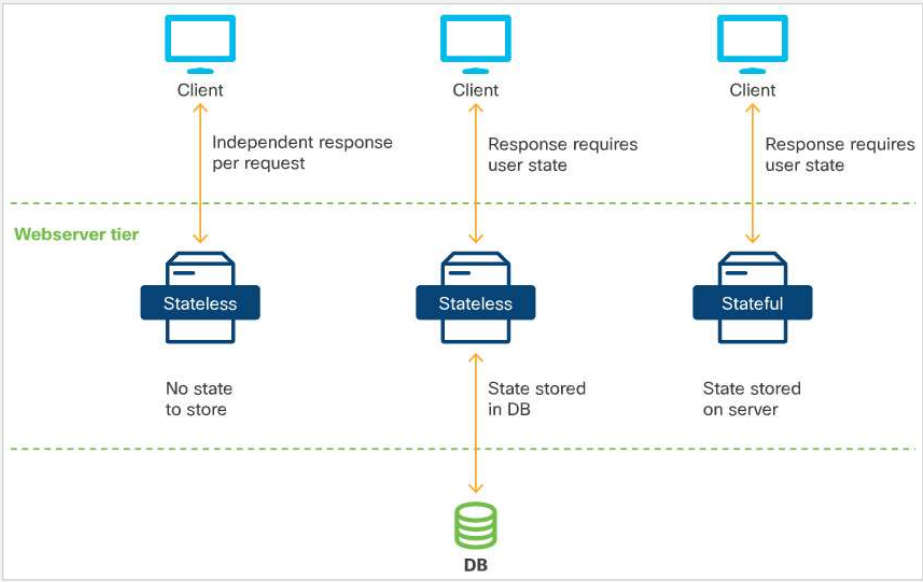
Concepts critique

Apatridie (Statelessness)

L'automatisation fonctionne mieux lorsque les applications peuvent être rendues sans état. Cela signifie que leur redéploiement en place ne détruit ni ne perd aucune trace des données dont les utilisateurs ou les opérateurs ont besoin.

Les deux états d'une demande sont les suivants :

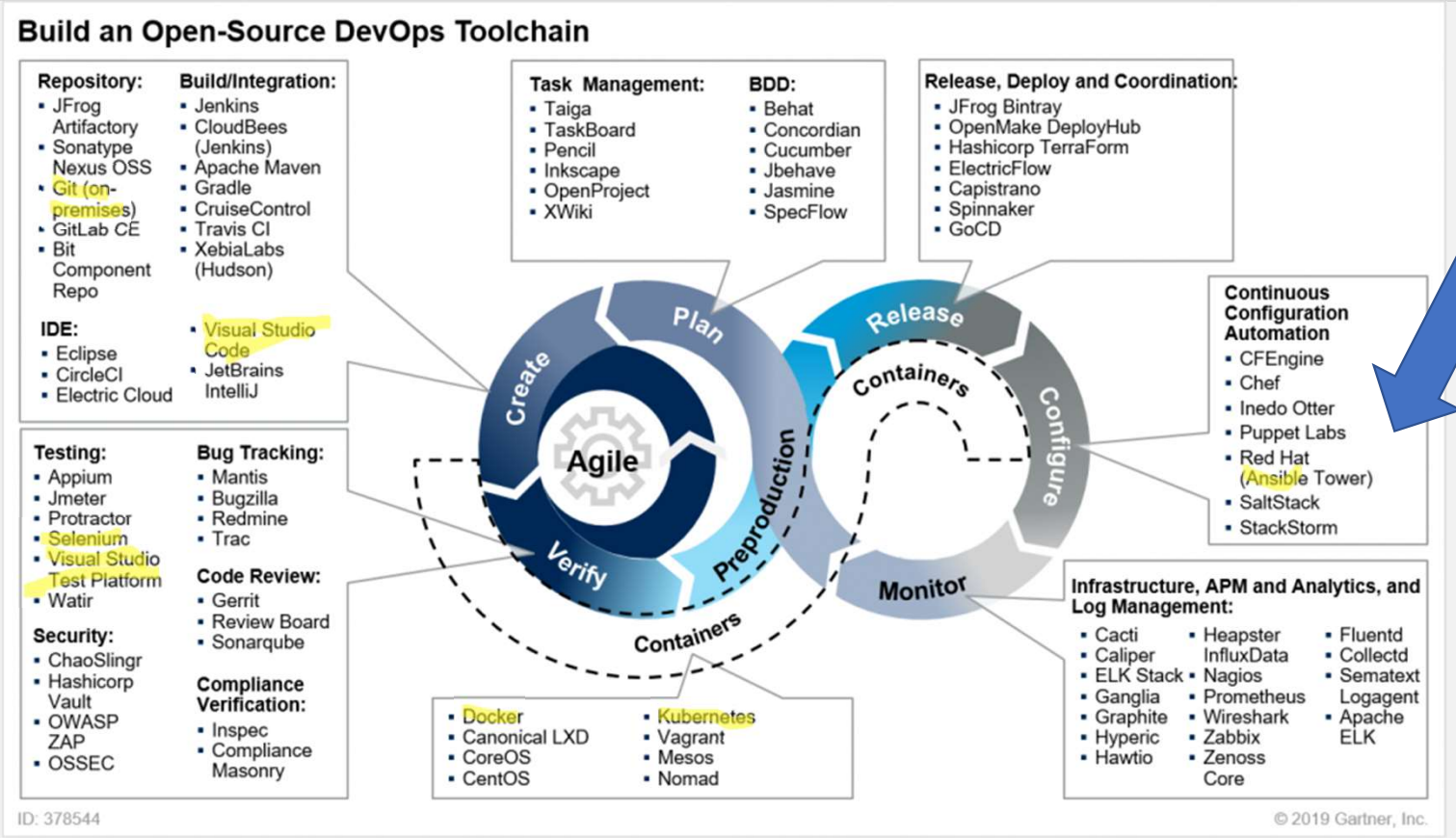
- **Avec état (Stateful)** : application qui enregistre des informations importantes dans des fichiers ou dans une base de données sur le fichier local.
- **Sans état (Stateless)** : application qui conserve son état dans une base de données distincte ou qui fournit un service qui ne nécessite aucune mémoire d'état entre les invocations.



Outils d'automatisation populaires

- Le premier outil d'automatisation moderne a été Puppet qui a été introduit en 2005 comme open source, puis commercialisé sous le nom de Puppet Enterprise par Puppet Labs en 2011.
- Les outils d'automatisation les plus populaires sont Ansible, Puppet, Chef. Ils partagent les caractéristiques suivantes :
  - Relativement facile à apprendre
  - Disponible en version open source
  - Les plugins et les adaptateurs leur permettent de contrôler directement ou indirectement de nombreux types de ressources
- De nombreuses autres solutions existent également. Les fournisseurs de cloud privés et publics approuvent souvent leurs propres outils pour une utilisation sur leurs plateformes, tels que le projet HEAT d'OpenStack, AWS CloudFormation, SaltStack et Terraform.

Aperçu d'outils open-source pour DevOps.



Source : [Gartner Blog Network]

Les premiers produits d'automatisation

- On peut citer notamment des produits comme
  - Puppet, Chef.
  - Malgré leur qualité ces produits avaient tout de même quelques défauts :
    - Ils demandaient un savoir-faire très spécifique.
    - Il fallait absolument installer des agents sur les serveurs à gérer.
    - Il n'est pas possible de gérer l'installation initiale de l'agent.
    - Enfin, les méthodes d'administration classiques à base de SSH étaient complètement ignorées.

Ansible

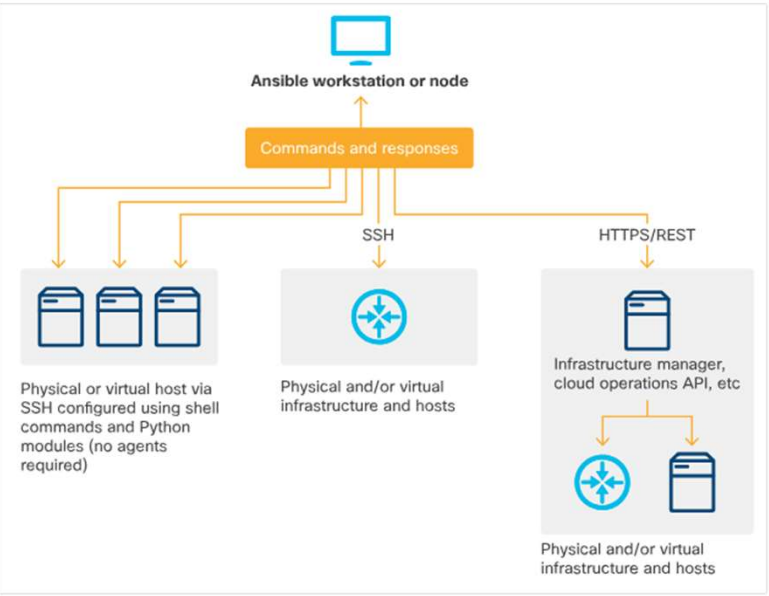
- Ansible est arrivé un peu après
  - Ansible, vous pourrez vous appuyer sur vos méthodes d'exploitation existantes (SSH, WinRM, Docker, API, etc.) :
  - Pas d'agent à installer sur les machines distantes, il faut juste un interpréteur Python.
  - Pas d'infrastructure à installer pour gérer les machines, vous pouvez tout faire depuis un poste quelconque avec les accès ad hoc.

L’objectif d’Ansible

- Ansible est un outil d'automatisation informatique.
  - Il permet de **configurer** des systèmes,
  - de **déployer** des logiciels et
  - **d'orchestrer** des tâches informatiques plus avancées, telles que des déploiements continus ou des mises à jour permanentes sans temps d'arrêt.
- Et ce pour :
  - la gestion de tous les environnements, des petites configurations avec une poignée d'instances aux environnements d'entreprise avec plusieurs milliers d'instances.
  - Si nécessaire, Ansible peut facilement se connecter à Kerberos, LDAP et d'autres systèmes centralisés de gestion de l'authentification.

Architecture Ansible :

- Le nœud de contrôle s'exécute sur n'importe quelle machine Linux exécutant Python 2 ou 3. Toutes les mises à jour du système sont effectuées sur le nœud de contrôle.
- Le nœud de contrôle se connecte aux ressources gérées via SSH et permet à Ansible de :
  - Exécutez des commandes shell sur un serveur distant, ou effectuez des transactions avec un routeur distant, ou une autre entité réseau, via son interface REST.
  - Injectez des scripts Python dans les cibles et les supprimez après leur exécution.
  - Installez Python sur les machines cibles si nécessaire.
- Les plugins permettent à Ansible de collecter des faits et d'effectuer des opérations sur une infrastructure qui ne peut pas exécuter Python localement.



Ansible

Installation d'Ansible

- L'application de nœud de contrôle Ansible est installée sur une machine Linux à partir de son dépôt de paquets public.

Structure de code ansible

- Dans la structure de code Ansible, le travail est séparé en fichiers YAML (**.yml**) qui contiennent une séquence de tâches, exécutées dans l'ordre de haut en bas. Ansible a des centaines de modules Python pré-construits qui enveloppent des fonctions au niveau du système d'exploitation et des méta-fonctions.

Playbooks et rôles

- Un playbook Ansible peut être écrit comme un document monolithique avec une série de tâches nommées et modulaires.
- Les développeurs construisent un modèle d'une tâche DevOps complexe à partir de séquences de tâches de playbook de bas niveau appelées rôles, puis référencer dans des playbooks de niveau supérieur, ajoutant parfois des tâches supplémentaires au niveau du playbook.
- Cette séparation des préoccupations assure la clarté, la réutilisation et la partageabilité des rôles.

Ansible

Organisation du projet Ansible

Les projets ansibles sont organisés dans une structure de répertoire imbriquée. La hiérarchie est facilement placée sous contrôle de version et utilisée pour l'infrastructure de style GitOps en tant que code.

Les éléments de hiérarchie de dossiers ansibles incluent les fichiers d'inventaire, les fichiers variables, les fichiers de bibliothèque et d'utilitaires, les fichiers de playbook principaux.

Ansible à l'échelle

- Il existe des défis de taille pour les grandes entreprises, tels que la gestion et le contrôle de l'accès à de nombreux nœuds Ansible de manière flexible et sécurisée. Cela inclut également la mise en place de contrôleurs à distance en toute transparence et en toute sécurité sous le contrôle de l'automatisation centralisée de l'entreprise.
- Pour cela, il existe deux solutions de plan de contrôle : Red Hat Ansible Tower et AWX projet.
- Les implémentations d'Ansible à plus grande échelle bénéficient également d'Ansible Vault, une fonctionnalité intégrée qui permet le chiffrement des mots de passe et d'autres informations sensibles.

Ansible : inventaire

- L'inventaire sous Ansible est un fichier au format INI. La déclaration d'un groupe se fait en utilisant le nom du groupe entre crochets.
- Les machines rattachées à ce groupe sont simplement ajoutées à la suite de la déclaration du groupe (une machine par ligne).
- Les noms utilisés doivent être présents dans le serveur DNS ou dans le fichier hosts de la machine de contrôle.
- Par défaut, vous accédez aux machines avec SSH. Néanmoins, Ansible gère d'autres types de connexion. Vous pouvez retenir les modes suivants :
    - ssh : connexion vers tout type d'Unix (Linux, *BSD, AIX, Solaris, etc.) ;
    - local : cas de la machine localhost ;
    - docker : connexion à des conteneurs ;
    - chroot/jail : travail dans un environnement isolé ;
    - winrm : connexion aux machines Windows.

```
#Commentaire
Au début les machines sans groupe (très rare)
pc-devops-1
#Suivi des groupes :
[apache]
srv-apache-1
srv-apache-2
[mysql]
srv-mysql-1
[active-directory]
active-directory-1 ansible_connection=winrm
srv-win-1 ansible_connection=winrm
[container]
container-1 ansible_connection=docker
```

Regroupement de machines

- Il est possible de regrouper des machines à l'aide de sections se terminant par ":children".
- Vous pouvez également ajouter des variables
- Ces variables peuvent être positionnées au niveau de la déclaration des machines ou au niveau des groupes.
- Les variables déclarées au niveau des machines prennent le pas sur celles au niveau du groupe et les fichiers **host_vars** et **group_vars** sont prioritaires sur les variables se trouvant dans le fichier d'inventaire.

Version : INV

```
[all:vars]
ansible_connection=local

[apache]
rec-apache-1 apache_url=rec.wiki.localdomain

[mysql]
rec-mysql-1 mysql_user_password=MyPassWord!

[active-directory]
active-directory-1

[microservices]
container-1 ansible_connection=docker

[linux:children]
apache
mysql

[windows:children]
active-directory

[container:children]
microservices

[windows:vars]
ansible_connection=winrm

[container:vars]
ansible_connection=localhost
```

Version : YAML

```
all :
  vars:
    ansible_connection: local

linux:
  children:
    apache:
      hosts:
        rec-apache-1:
          apache_url: "rec.wiki.localdomain"

    mysql:
      hosts:
        rec-mysql-1:
          mysql_user_password: "MyPassWord!"

windows:
  children:
    active-directory:
      hosts:
        active-directory-1: {}
  vars:
    ansible_connection: "winrm"

container:
  children :
    microservices :
      hosts :
        container-1 :
          ansible_connection :  "docker"
      vars :
        ansible_connection :  "localhost"
```

Ansible : variable d’inventaire

- Ces variables peuvent être positionnées au niveau de la déclaration des machines ou au niveau des groupes. Ci-dessous un exemple de déclaration de la variable `apache_url` pour la machine `rec-apache-1`

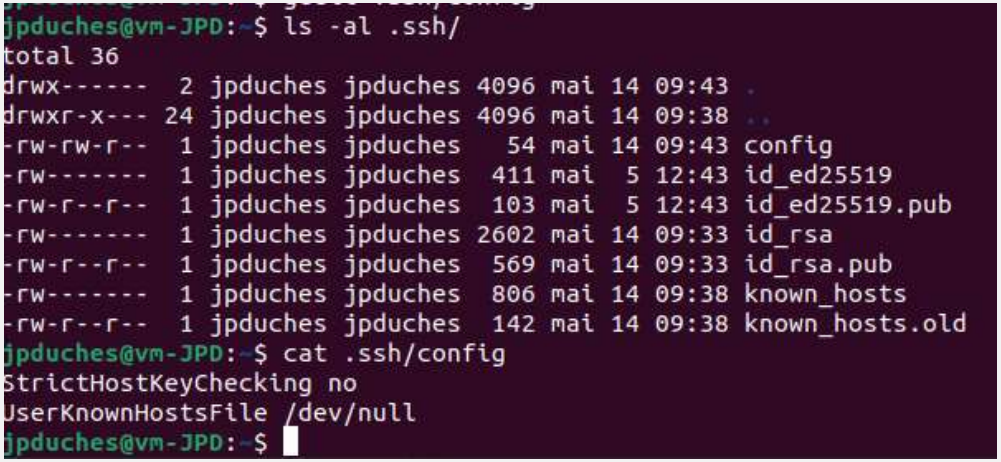
Ansible : playbook

- Un playbook peut contenir énormément d'informations, mais la plupart du temps vous pouvez vous contenter d'un sous-ensemble relativement restreint.
- Exemple de champs couramment utilisés :
  - Le nom de playbook : name;
  - La liste des machines cibles : hosts;
  - Les options de gestion de la sécurité
  - La gestion du lancement de la collecte des informations : setup
  - Liste d'instructions à dérouler : tasks.
- Exemple dans le cas d'un déploiement d'Apache httpd :
  - Un appel au module apt ou yam pour l'installation du package;
  - Un appel au module service;
  - Un appel au module firewall;
  - Un appel à copy;
- Pour chaque module :
  - Le nom du module à exécuter;
  - Les options nécessaires au fonctionnement du module dans des sous-champs
  - Un champs name avec une descriptions de l'opération. Pour clarté.

```
- name: "Apache Installation"
  hosts: all
  tasks:
    - name: "Install apache package"
      yum:
        name: "httpd"
        state: "present"
    - name: "Start apache service"
      service:
        name: "httpd"
        state: "started"
        enabled: yes
    - name: "Allow http connections"
      firewalld:
        service: "http"
        permanent: yes
        state: "enabled"
    - name: "Copy test.html"
      copy:
        src: "test.html"
        dest: "/var/www/html"
       owner: "apache"
       group: "apache"
```

L’importance de SSH

- Les clés publiques et privées
- Know_hosts
  (les hôtes connut)
- Config



Ansible-doc

- Pour trouver l ’aide :
  - Directement dans le terminal :

  $ansible-doc –l |grep apt

  $ansible-doc apt
- Le site : http://docs.ansible.com/

- Moteur de recherche :

  ansible playbook keywords

Référence sur Ansible

- GitHub du projet : https://github.com/ansible/ansible
- Groupe de discussion : https://groups.google.com/forum/#!forum/ansible-project