Support pédagogique Ansible - 1

Le programme de la formation

- Mouvement Devops
- Positionnement de Ansible dans le paysage actuel
- Présentation de Ansible
- Les concepts clés
- Quelques éléments de vocabulaire
- Les ressources
- Installation d'Ansible
- Accès SSH sur les serveurs distants

Le programme de la formation

- Les fichiers Inventaire
- Les inventaires dynamiques
- Ansible en mode ad hoc
- La documentation avec ansible-doc
- Cas pratique avec l'installation de Apache
- YAML - Yet Another Markup Language
- Les Playbooks

Le programme de la formation

- Vérification de la syntaxe des playbooks avec ansible-lint
- Les variables et leurs hiérarchies
- Le chiffrement des fichiers
- Ansible et les machines sous Windows
- Le moteur de template Jinja
- Les Templates Jinja
- Exécuter des tâches à plusieurs niveaux

Le programme de la formation

- Délégation des tâches
- Les tags
- Exécution conditionnée
- Les handlers
- Les Rôles
- Cas pratique avec installation de MediaWiki
- Les opérations séquentielles
- Ansible et le cloud

Le programme de la formation

- Mise à jour par roulement et parallèle
- Débogage avec Ansible
- Ansible Galaxy
- Gestion du callback d’affichage
- Optimisation des performances
- AWX / Ansible Tower

Le programme de la formation

- Ajout d’un répartiteur de charge
- Mise à jour et réentrance de script
- (Écrire son propre callback d’affichage)
- (Ecrire des modules)
- (Écrire son propre inventaire dynamique)

Déroulement de la formation

Deux jours les 03 et 04 décembre 2018

--

Matin 09H00/30 - 13H00

On deroule la **Théorie**

Après-midi 14H00 - 17H00/30

On fait de la **Pratique**

Mouvement DevOps

- Le DevOps est un mouvement initié en **2009** visant à rapprocher les équipes de **devs** et les **ops** en charge de l'exploitation. Les initiateurs du DevOps sont des partisans des méthodes **agiles** (Scrum).
- Le terme a été employé dans le cadre des « **DevOpsDays** » de Gand en Belgique en Octobre 2009 https://www.devopsdays.org/
- D'un côté, les **développeurs** (devs)
- D'un autre les **exploitants** ou opérateurs (ops)

Mouvement DevOps

La philosophie DevOps

Culture	• Focus on people • Embrace change and experimentation
Automation	• Continuous delivery • Infrastructure as code
Lean	• Focus on producing value for the end user • Small batch sizes
Measurement	• Measure everything • Show the improvement
Sharing	• Open information sharing • Collaboration and communication

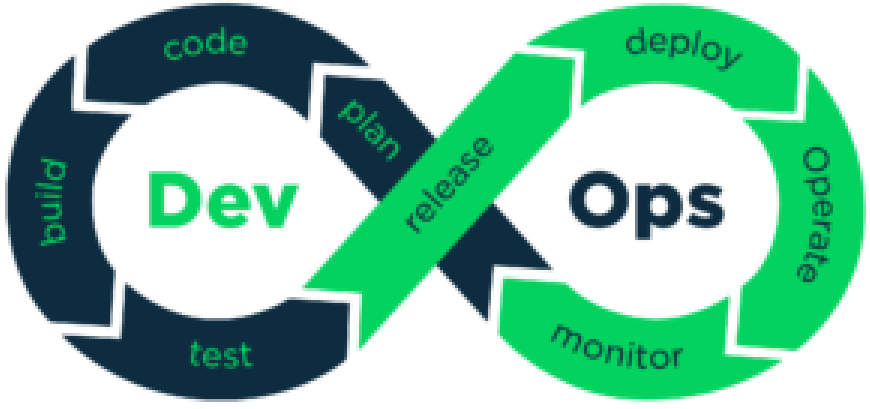
Mouvement DevOps

Les principes DevOps

La philosophie **DevOps** est de livrer du code de qualité, toujours meilleur et plus vite, toujours plus vite.

Cela passe ainsi par une **livraison continue des applications** et des **tests** doivent être effectués à toutes les étapes du processus de développement.

L'**automatisation** doit être sans faille : un code peut être poussé en production que s'il est validé par le **processus de livraison**.



- **Planifier et écrire du code** : organise l'équipe de développement, planifie et écriture du code à l'aide d'un outil de gestion de la configuration de code tel que `Git` et `Trello` pour l'organisation des activités de développement en équipe.
- **Build & Test** : Une fois le code écrit, il faut préparer un environnement d'intégration qui va permettre de faire des tests (`Jenkins` , `GitLab`).
- **Versionning et deploiement** : Une fois les tests validés, des outils de versionning (`Git`) et de gestion de configuration (`Ansible`) peuvent être utilisés pour deployer le code en production
- **Exploitation et Supervision** : Une fois le code est en production, il s'agit de maintenir le système à jour et le superviser (`New Relic` , `Nagios` , `Zabbix` ...).

Mouvement DevOps

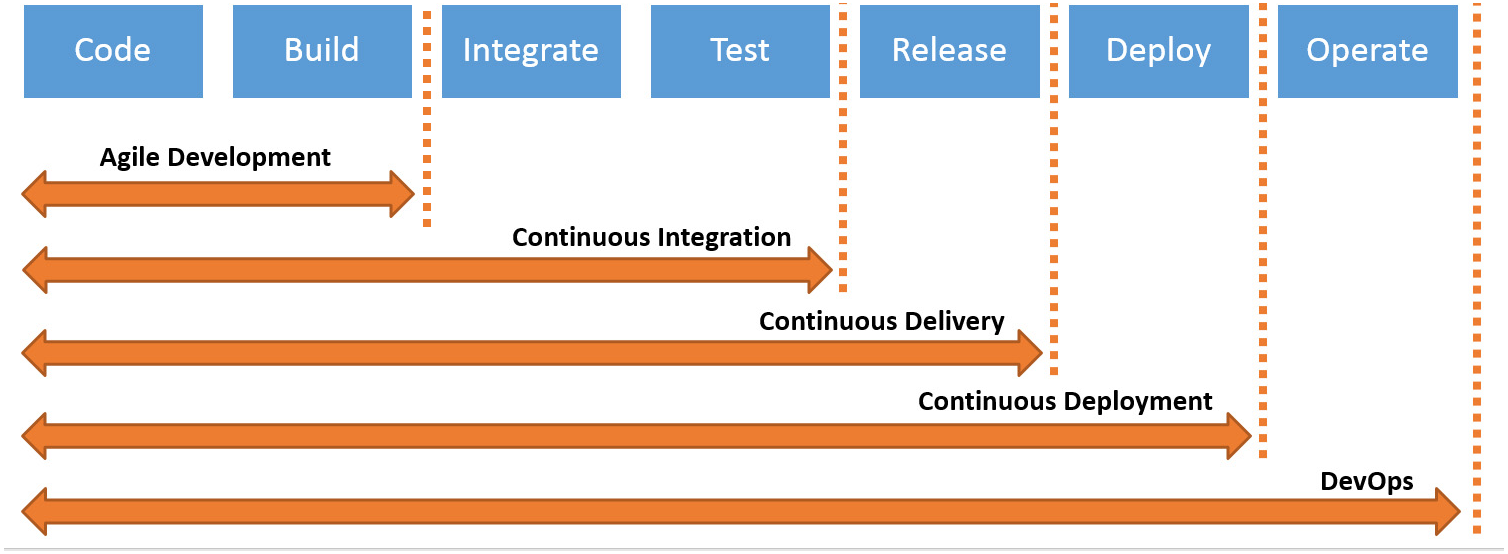
Serverless architecture est l'approche informatique la plus récente dans la création de systèmes dans le **Cloud** (on s'éloigne de la dépendance matérielle).

Docker est un exemple d'architecture sans serveur (Serverless).

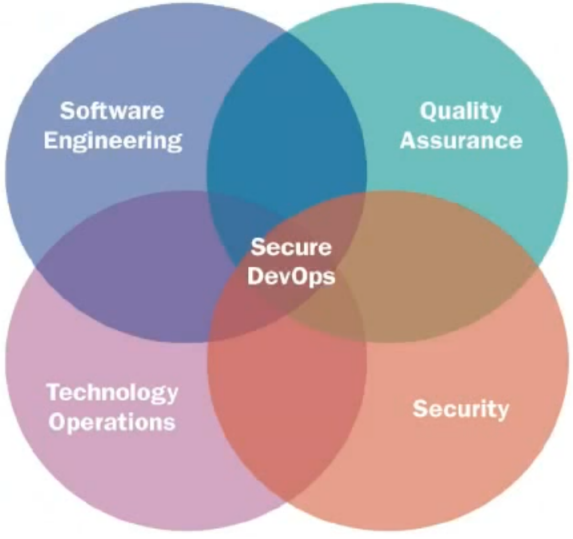
Docker se complète avec des outils de configuration comme **Ansible** pour rendre le code plus portable, pouvant être intégré dans plusieurs environnement, notamment sur des systèmes clouds reparties (le cloud **AWS** est actuellement le plus répandu).

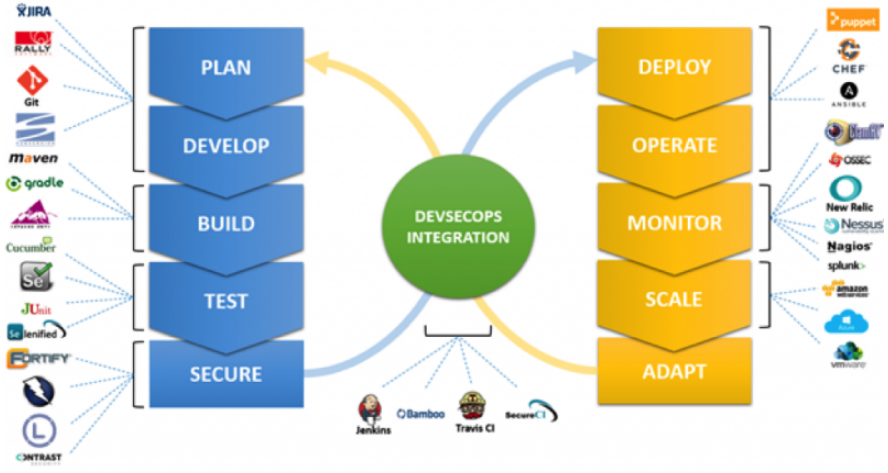
Mouvement DevOps : Automatisation

- Le pipeline **Continuous Integration** (CI) / **Continuous Deployment** (CD), **Continuous Delivery** qui va permettre de mettre en production rapidement le code de manière sécurisée.
- Le **provisionning** des infrastructures qui permet de construire rapidement des environnements reproductibles et maîtrisés.



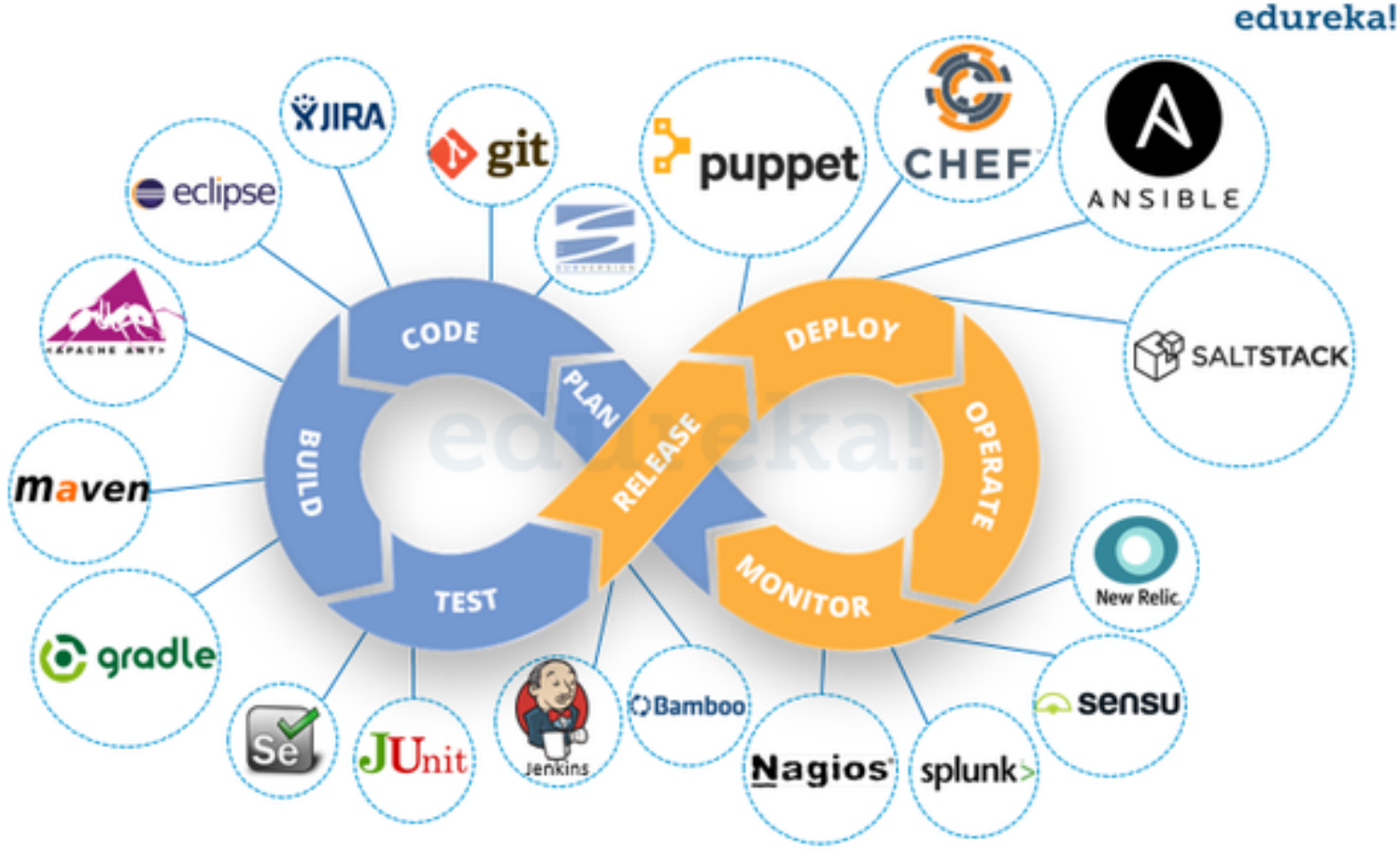
Mouvement DevOps





Mouvement DevOps

Quelques outils DevOps

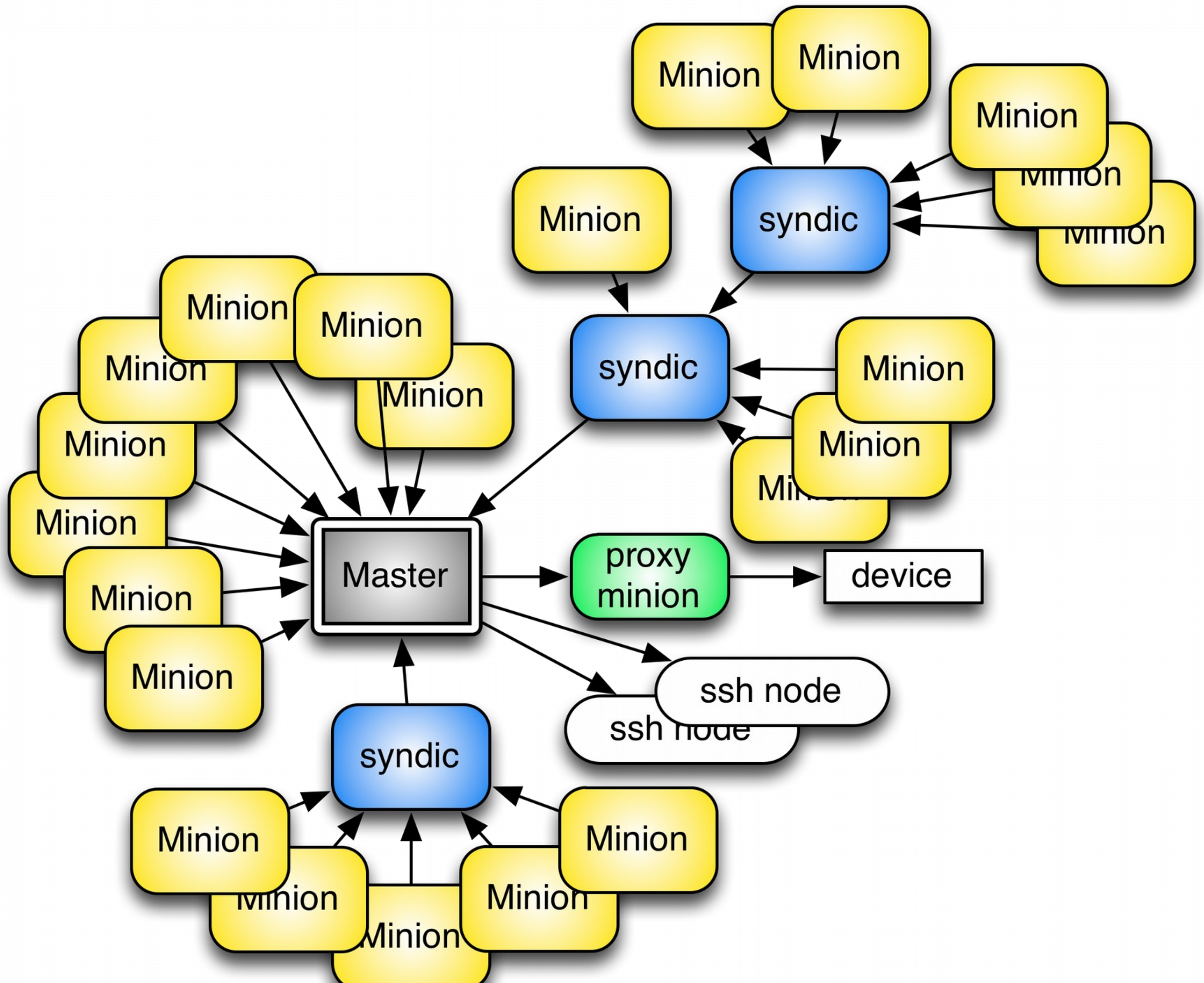


Ansible dans le paysage actuel

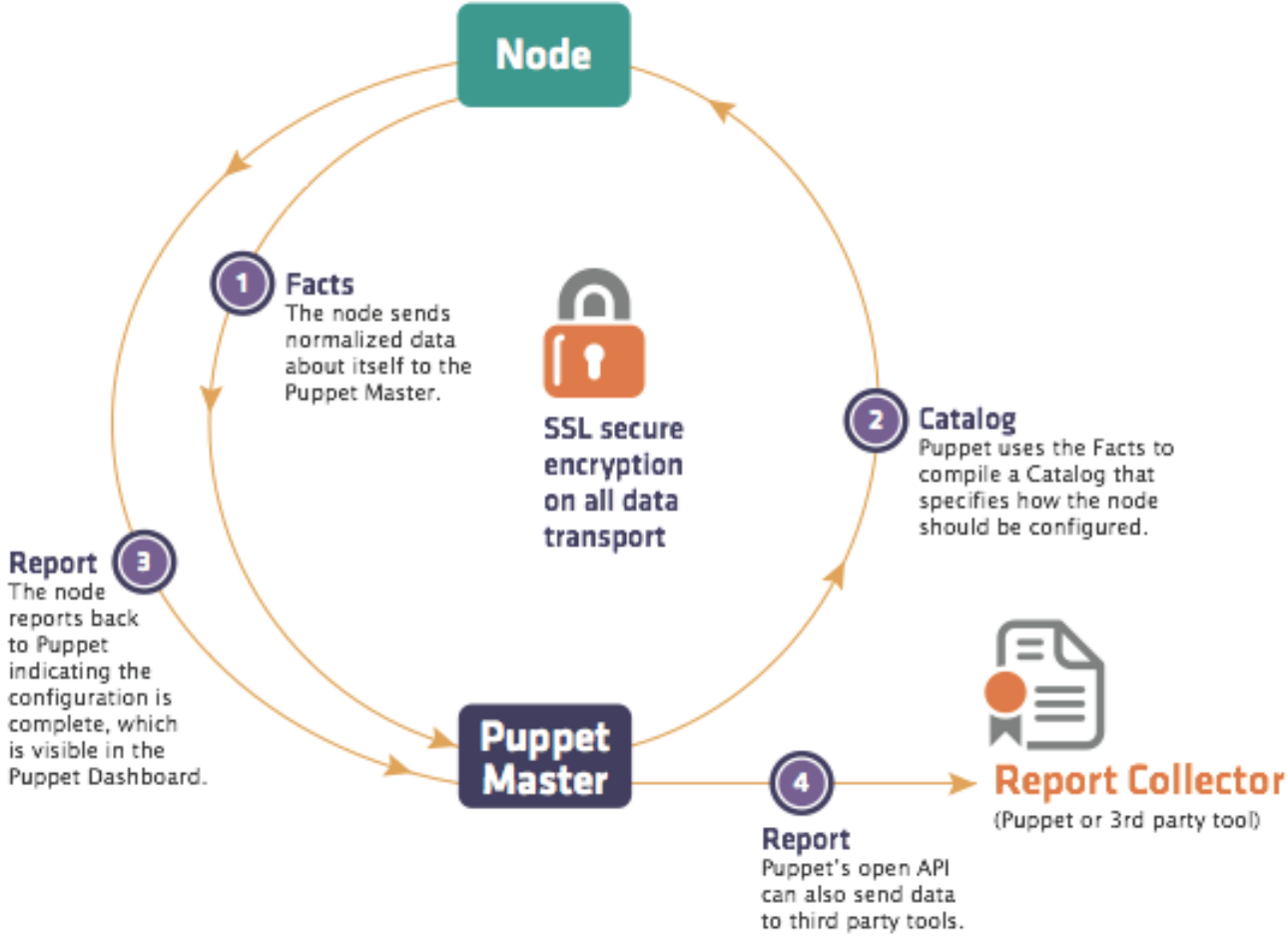
	Editeur	Language	License	Agent-less	Have a GUI	First release	Commits	Contributors
Ansible	**Red Hat**	Python	GPLv3+	Yes	Yes	2012	40 000	3 836
Chef	**Chef**	Ruby, Erlang	Apache 2.0	No	Yes	2009	22 000	563
Puppet	**Puppet**	Ruby	GPL	No	Yes	2005	29 000	502
Salt	**Saltstack**	Python	Apache 2.0	Both	Yes	2011	99 000	2 161

La différence peut se faire sur les performances de Ansible qui sont notamment liées au fait qu'il est agentless (100 nodes par node manager).

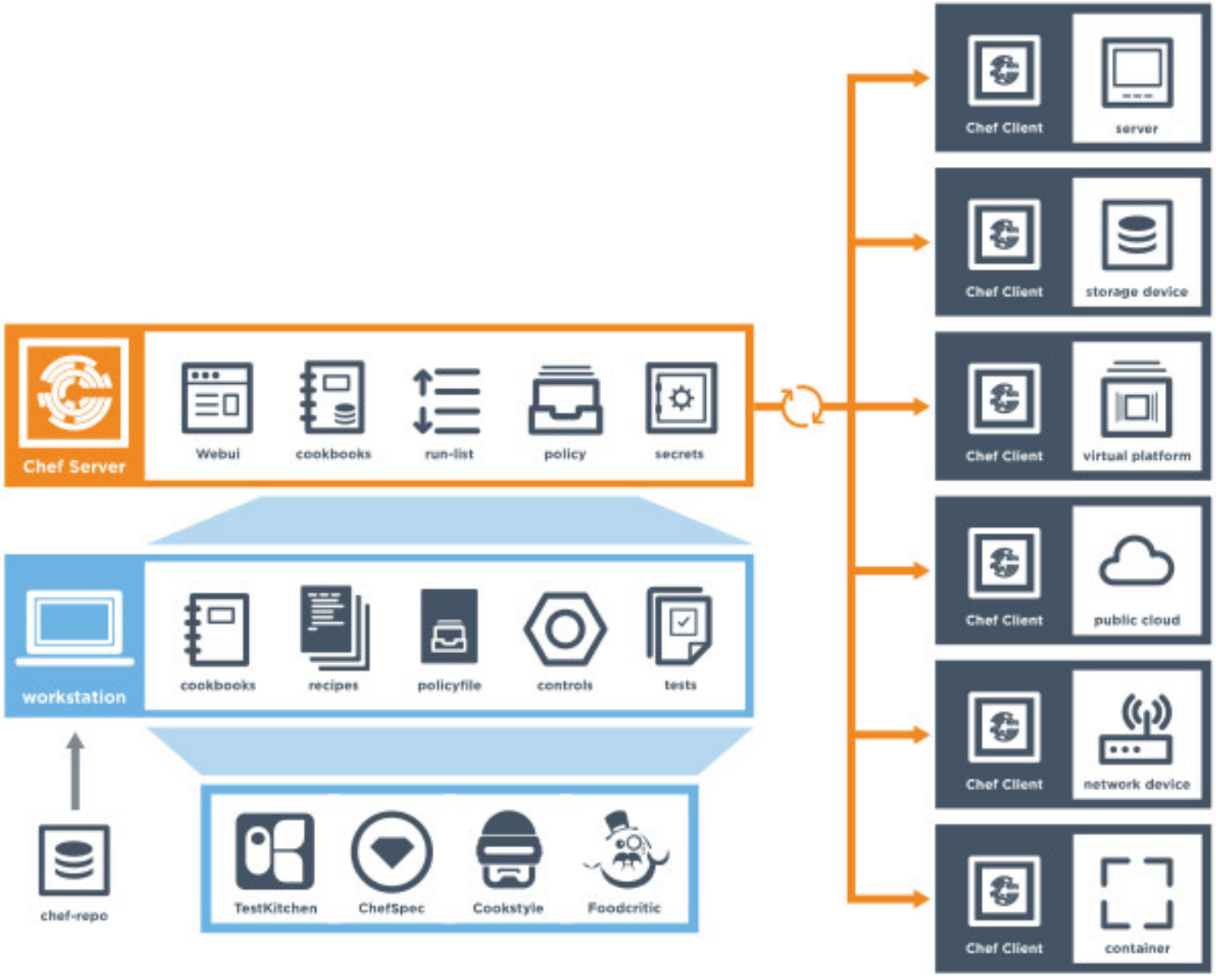
Salt logique



Puppet logique

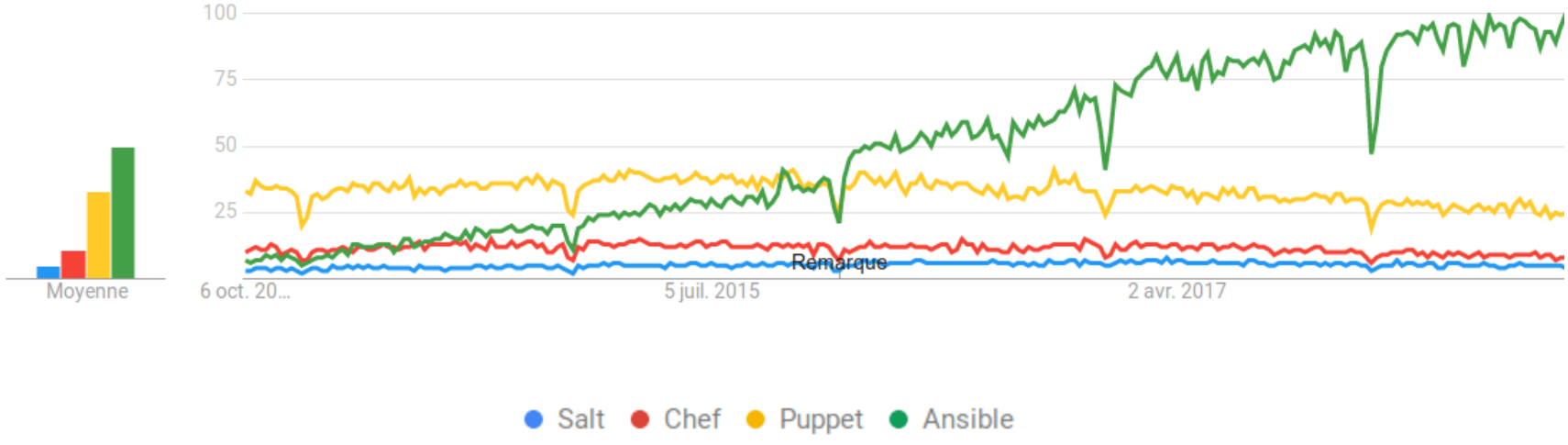


Chef logique

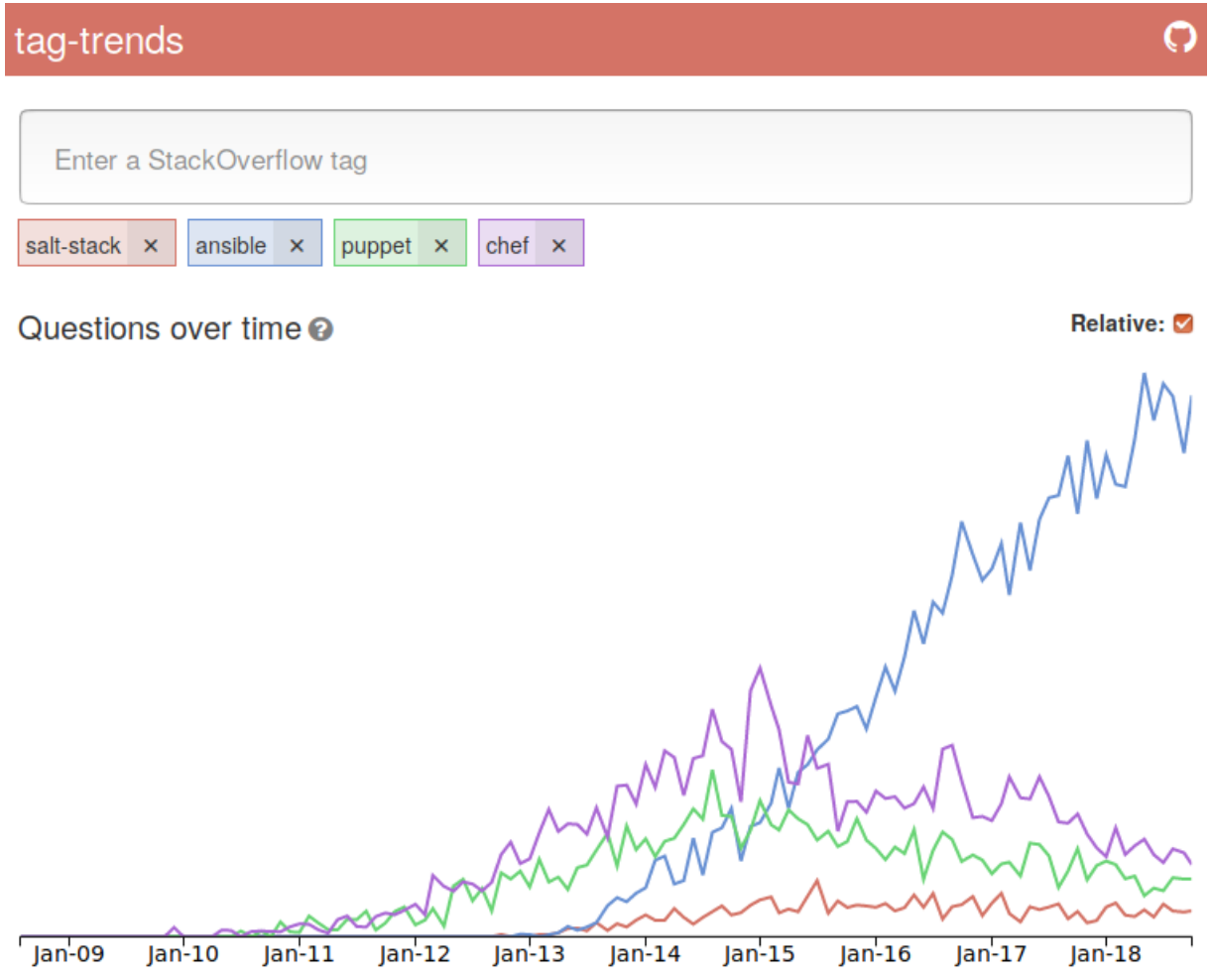


Ansible dans le paysage actuel

Popularité sur Google Trend



Ansible dans le paysage actuel



Source sur StackOwerFlow

Présentation de Ansible

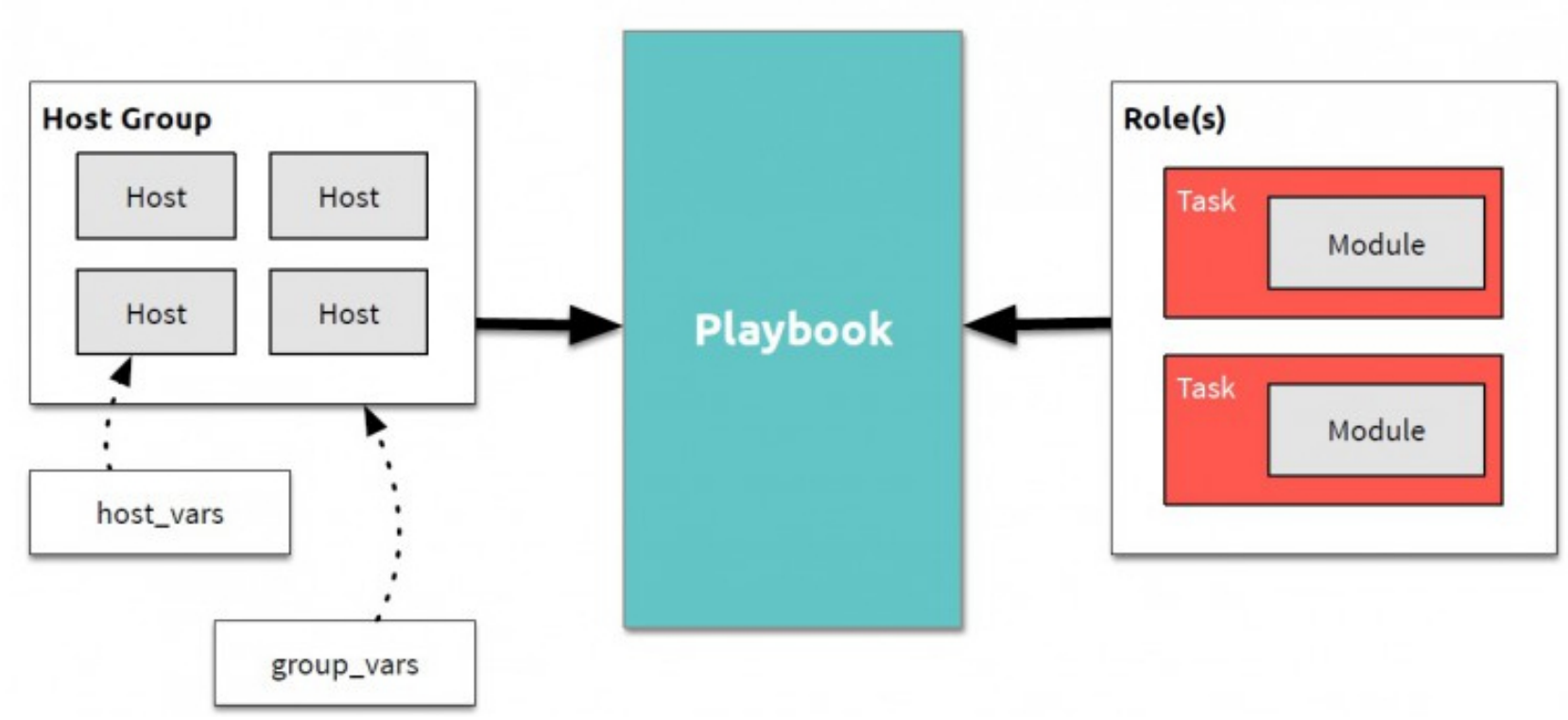
Michael DeHaan

- Créateur de Ansible en **2012**
- Achat par RedHat en **2015**
- **3,500+** Community Contributors
- **1,600+** Ansible Modules
- Version **2.7** (Octobre 2018)

- Une version majeure approximativement tous les deux mois
- **Agentless** (OpenSSH)
- **Idempotence** : une opération a le même effet qu'on l'applique une ou plusieurs fois
- Format de données en `Json`
- Langage de programmation objet en `Python`
- Format de représentation de données en `YAML`
- Intègre des modules pour des composants matériels (notamment pour piloter du matériel réseau)

Ansible est un outil d'**automatisation** informatique. Il peut configurer des systèmes, déployer des logiciels et orchestrer des tâches informatiques avancées telles que des déploiements continus.

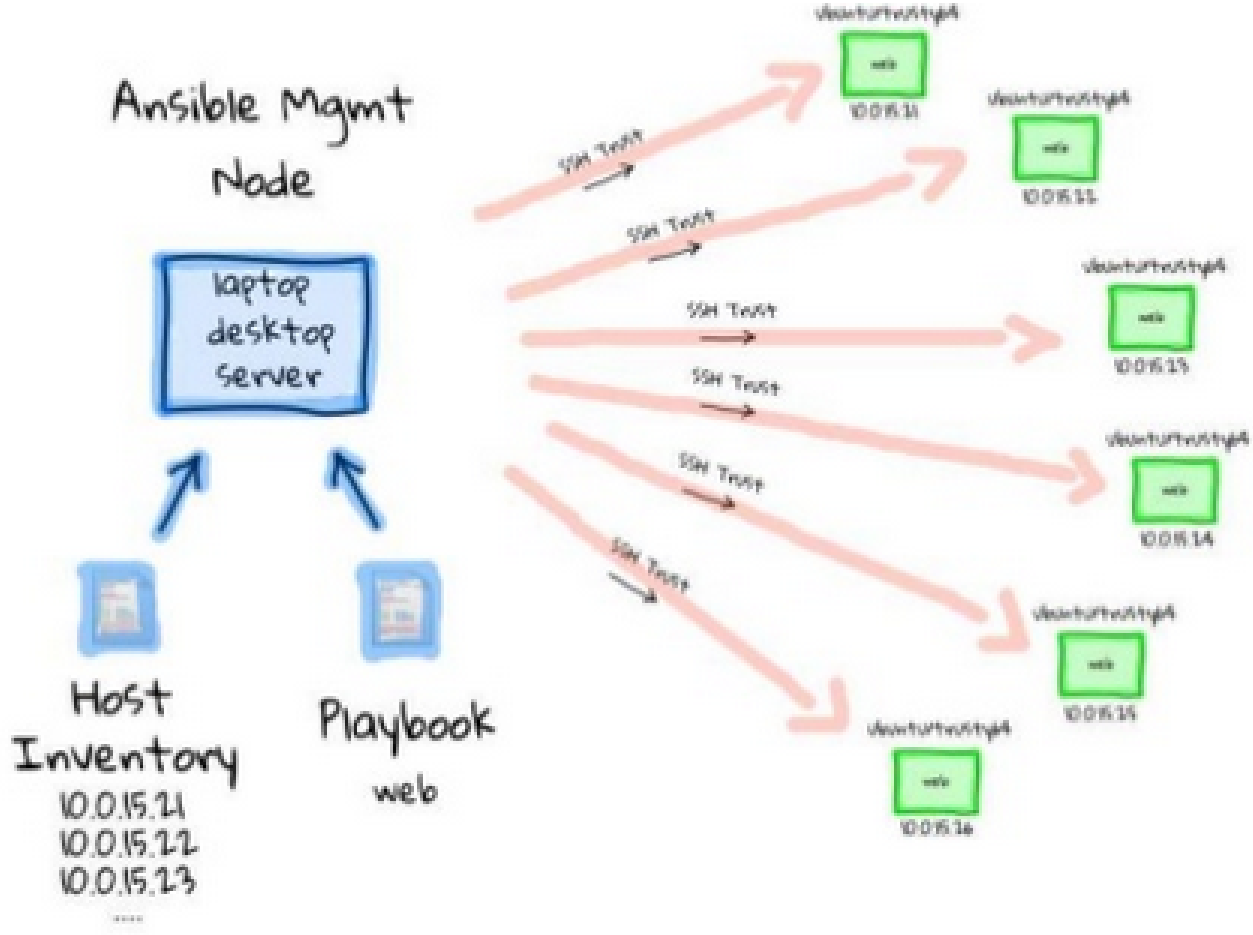
Les concepts clés



Quelques éléments de vocabulaire

- **Inventaire** : l'inventaire des serveurs (des hôtes)
- **Fact** : les informations récupérées automatiquement par Ansible sur les hôtes (variable d'environnement, version d'OS, …)
- **Template** : un modèle de fichier au format `Jinja2`
- **Module** : ce qui sera exécuté sur un serveur
- **Task** : un ensemble de module
- **Rôle** : un ensemble de tâche, de variables, de template regroupés fonctionnellement
- **Playbook** : c'est un fichier de configuration au format `YAML` qui assemble des actions

Comment Ansible fonctionne



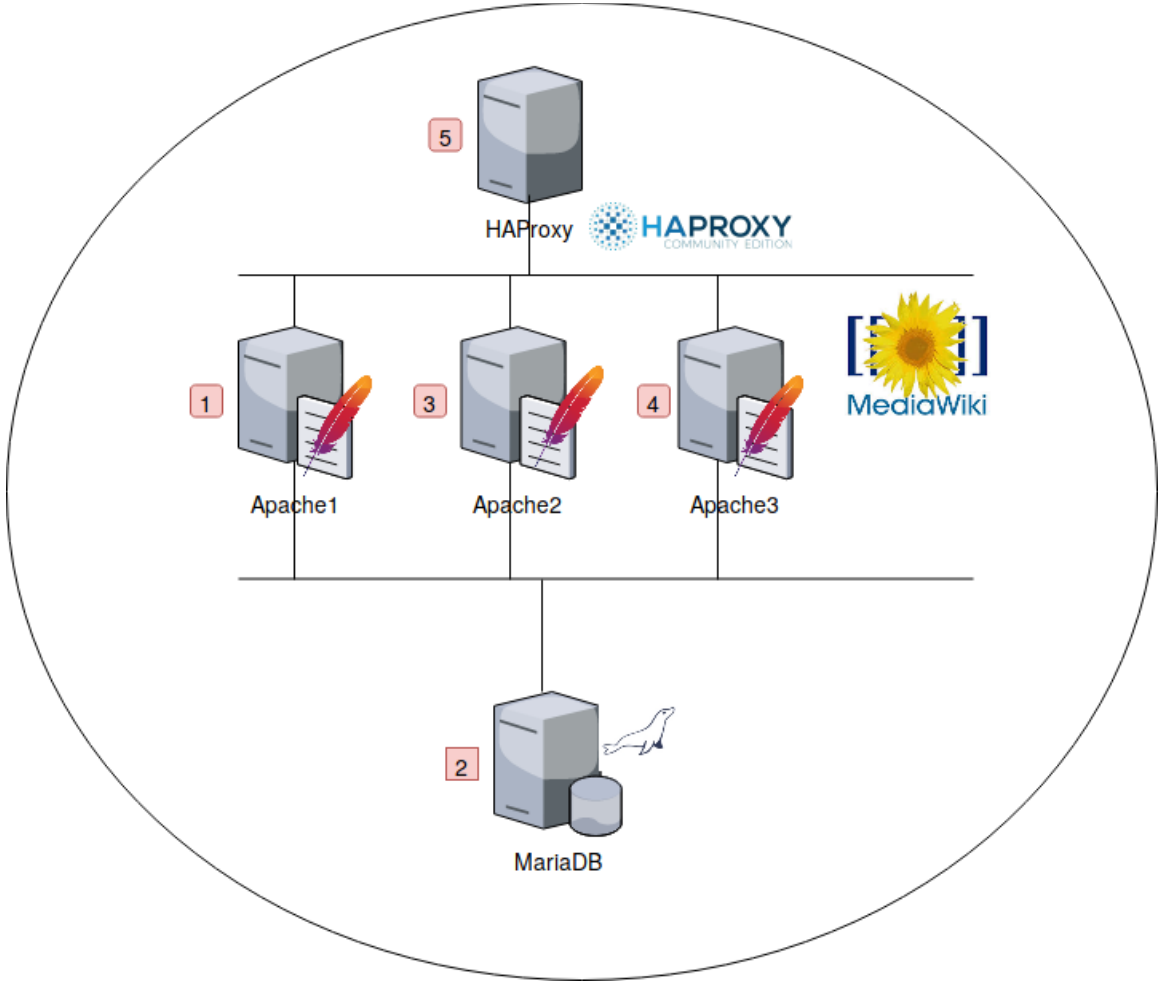
Les ressources

- **Ansible Documentation** : couvre toutes les options Ansible en profondeur.
  - https://docs.ansible.com/ansible/
- **Ansible Glossary** : si il y a un terme que vous ne semblez pas comprendre, consultez le glossaire.
  - https://docs.ansible.com/ansible/glossary.html
- **Ansible Mailing List -** groupe de discussion Google
  - https://groups.google.com/forum/#!forum/ansible-project

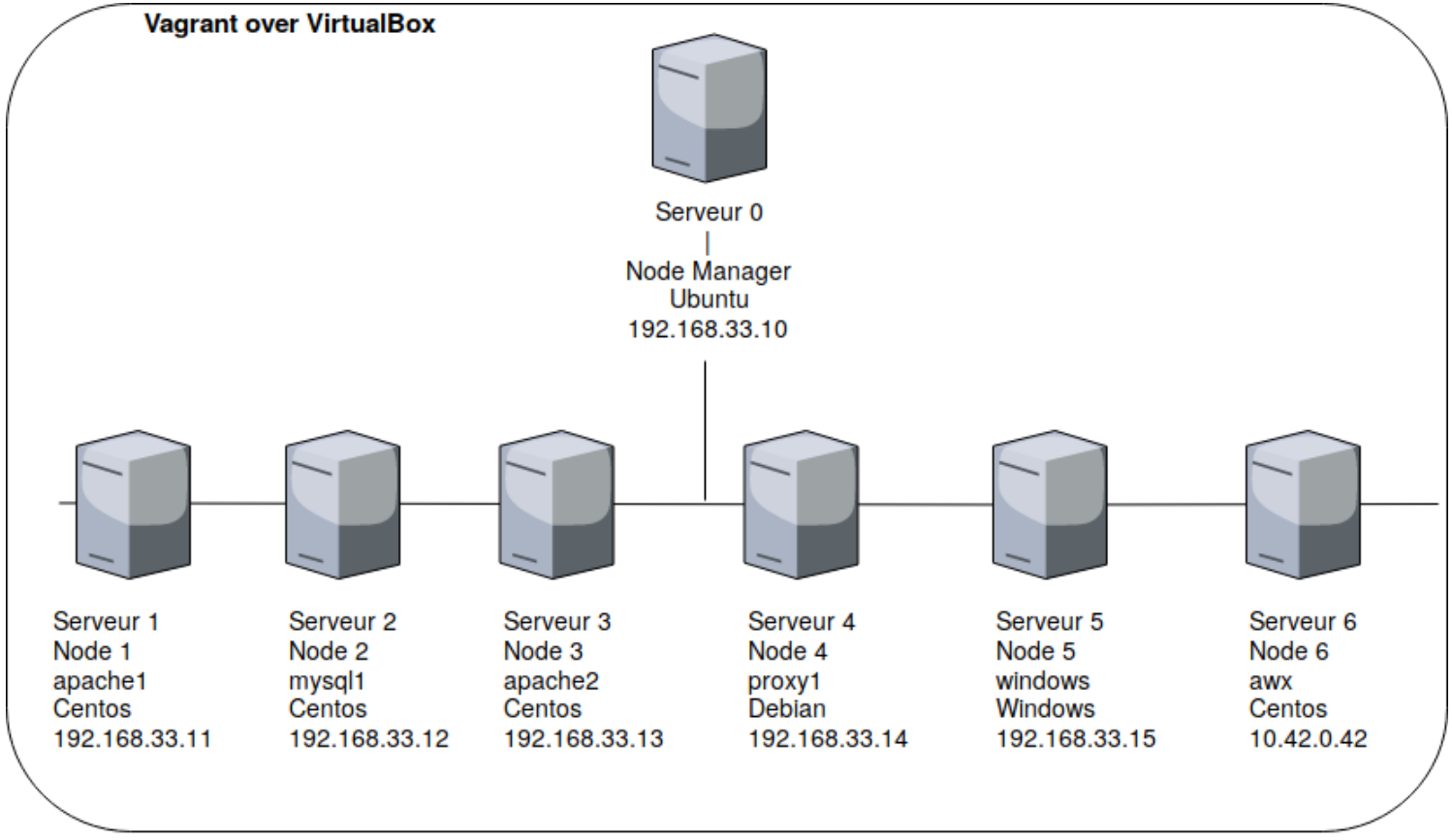
- **Ansible on GitHub** : le référentiel de code Ansible officiel.
  - https://github.com/ansible/ansible
- **Ansible Example Playbooks on GitHub** : de nombreux exemples de configurations de serveur communes.
  - https://github.com/ansible/ansible-examples
- **Getting Started with Ansible** : un guide simple sur la communauté et les ressources d'Ansible
  - https://www.ansible.com/get-started
- **Ansible Blog** : le Blog Ansible
  - https://www.ansible.com/blog

Schéma de la mise en œuvre

Automatisation de l'installation de MediaWiki (gestion de la scalabilité et de la mise à jour des versions)



Pour les travaux pratiques



Les éléments pratiques pour le LAB de TEST

- Utilisation de Vagrant sur VirtBox pour provisionner l'environnement de test
- Création d'un répertoire par serveur
- `vagrant init ubuntu/bionic64`
- `vagrant init centos/7`
- `vagrant init ansible/tower`
- `vagrant up`
- `vagrant ssh`
- Sur Centos : `setenforce 0`

- Pour Centos `ifup`
- Pour **Ubuntu** : Modification du fichier `Vagrantfile` pour configurer le réseau
  - `config.vm.network :private_network, ip: "192.168.33.x"`
- Pour **Windows** : Modification du fichier `Vagrantfile` pour configurer le réseau
  - `config.vm.network "private_network", ip: "192.168.33.x"`
- `vagrant reload`
- Test ping entre le node manager et le node1
- S'assurer que les serveurs sortent sur le Net
- Utiliser l'utilisateur vagrant (pass: vagrant)

Installation d’Ansible

- Installation via les packages systèmes
- Installation via `pip` de Python (`virtualenv`)
- Installation via les sources (Archives ou Git)

Installation sur CentOS

Deux étapes sont nécessaires :

- Activation des sources Extra Package Enterprise Linux :

```
$ sudo yum install epel-release
```

- Installation d'Ansible :

```
$ sudo yum install ansible
```

- La version Ansible CentOS

```
$ sudo yum info ansible
```

```
Paquets disponibles
Nom                 : ansible
Architecture        : noarch
Version             : 2.5.5
```

Installation via pip

- Installer de `pip`

```
$ yum install python2-pip
```

- Installation de Ansible

```
$ pip install ansible
```

- Version installée

```
$ ansible --version
```

```
ansible 2.5.5
  config file = None
  ~
  python version = 2.7.5 (default, Apr 11 2018, 07:36:10)
  [GCC 4.8.5 20150623 (Red Hat 4.8.5-28)]
```

Utilisation de virtualenv

- Il permet de faire cohabiter plusieurs versions d'Ansible
- Il est associé au mécanisme de packaging `pip`
- Pas besoin des droits Administrateur pour faire l'installation

```
$ yum install python-virtualenv
```

```
$ virtualenv ansible2.4
```

```
$ source ansible2.4/bin/activate
```

```
$ pip install ansible==2.4
```

```
(ansible2.4) [alex@localhost ~]$ ansible --version
```

```
ansible 2.4.0.0
```

virtualenv : Installation d'une deuxième version de Ansible sur le même serveur pour le même utilisateur

```
$ virtualenv ansible2.5
```

```
$ source ansible2.5/bin/activate
```

```
$ pip install ansible==2.5
```

```
(ansible2.5) [alex@localhost ~]$ ansible --version
```

```
ansible 2.5.0
 ~
  python version = 2.7.5 (default, Apr 11 2018, 07:36:10)
  [GCC 4.8.5 20150623 (Red Hat 4.8.5-28)]
```

```
$ deactivate
```

virtualenv : Installation Ansible 2.5 avec un environnement Python 3

- Installation de Python 3

```
$ yum -y install https://centos7.iuscommunity.org/ius-release.rpm
```

```
$ yum -y install python36u
```

```
$ yum -y install python36u-pip
```

- Creation de l'environnement Python 3

```
$ virtualenv -p /usr/bin/python3.6 python3
```

```
Running virtualenv with interpreter /usr/bin/python3.6
Using base prefix '/usr'
New python executable in /home/alex/python3/bin/python3.6
Also creating executable in /home/alex/python3/bin/python
Installing setuptools, pip, wheel...done.
```

```
$ source  python3/bin/activate
```

```
(python3) [alex@localhost ~]$ python --version && pip --version
```

```
Python 3.6.5
pip 9.0.1 from /home/alex/python3/lib/python3.6/site-packages (python 3.6)
```

- Installation de Ansible dans l'environement Python 3

```
$ pip install ansible
```

```
(python3) [alex@localhost ~]$ ansible --version
```

```
ansible 2.5.5
 ~
  python version = 3.6.5 (default, Apr 10 2018, 17:08:37)
  [GCC 4.8.5 20150623 (Red Hat 4.8.5-16)]
```

Accès SSH sur les serveurs distants

> SSH est utilisé par défaut pour se connecter aux serveurs distants
> Néanmoins, Ansible gère d'autres types de connexion (`local`, `docker`, `chroot/jail`, `winrm`)

- Créer une clés SSH

```
$ ssh-keygen -t ecdsa -f .ssh/id_ecdsa_ansible
```

- Copier la clés publique sur le serveur

```
$ ssh-copy-id root@serveur
```

> `pipelining = False` : réduit le nombre d'opérations SSH requises pour exécuter un module sur le serveur distant
> Pour les connexions `sudo`, désactiver `requiretty` in `/etc/sudoers`

> Si la copie n'est pas possible car la connexion par mot de passe est désactivée

- Récupérez le contenu du fichier `~/.ssh/id_ecdsa_ansible.pub` sur votre machine Ansible.
- Se connecter sur la machine distante.
- Créez un répertoire `.ssh` avec des droits restreints à l'utilisateur :

```
$ mkdir -p ~/.ssh
```

```
$ chmod 700 ~/.ssh
```

- Créez un fichier `authorized_keys` avec des droits restreints :

```
$ touch ~/.ssh/authorized_keys
```

```
$ chmod 600 ~/.ssh/authorized_keys
```

- Déposez la clé dans le fichier `authorized_keys`
- Pour éviter de ressaisir à chaque fois la *passphrase*

```
$ ssh-add ~/.ssh/id_ecdsa_ansible
```

```
$ eval $(ssh-agent)
```

Les fichiers Inventaire

- Déclarer un groupe de serveurs : `[]`

```
[apache]
www1
www2

[mysql]
db1
db2
```

> Le groupe `ALL` contient tous les serveurs de l'inventaire

Regroupement de serveurs

> Plusieurs combinaisons peuvent être imaginées pour regrouper des serveurs et composer des ensembles sans les répéter.

```
# Pour toutes les serveurs, la connexion se fait en local
[all:vars]
ansible_connection=local

# Groupe apache, contenant un serveur
[apache]
rec-apache-1 apache_url=rec.wiki.localdomain

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

# Un groupe avec deux serveurs, identifiés par une séquence entre []
[active-directory]
active-directory-[1:2]

[microservices]
container-1 ansible_connection=docker

# Le groupe linux est composé des groupes apache et mysql
[linux:children]
apache
mysql
```

```
[windows:children]
active-directory

[container:children]
microservices

# Pour le groupe windows, les connexions se font en winrm
[windows:vars]
ansible_connection=winrm

[container:vars]
ansible_connection=localhost
```

Les inventaires dynamiques

> Il est possible de récupérer des listes de serveurs depuis des outils déjà présent existants grâce à des plugins.

- Outils de supervision (Nagios, Zabbix...)
- Infrastructure virtualisation (VMware, Proxmox...)
- Cloud (AWS, Azure ou autre)
- Conteneurs de type Docker (Kubernetes ou Docker Swarm)

> Les scripts d'inventaire dynamique sont disponibles sur : https://github.com/ansible/ansible/tree/devel/contrib/inventory

Ansible en mode ad hoc

- Créer un fichier `inventaire`

```
centos ansible_user=alex ansible_host=192.168.122.123
```

- On lance la première commande Ansible

```
$ ansible -i inventaire -m ping all
centos | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
```

- `-i` : **i**nventaire
- `-m` : **m**odule

Le format Json

- **RFC 8259**: The JavaScript Object Notation (`JSON`) Data Interchange Format
- C'est un format **textuel**, qui permet de représenter des **structures de données hiérarchiques**.
- `{...}` : les accolades définissent un **objet**.
- `"language":"Java"` : Les guillemets (double-quotes) et les double-points définissent un **couple clé/valeur** (on parle de membre).
- `[...]` : Les crochets définissent un **tableau** (ou array en anglais).
- `{"id":1, "language":"json", "author":"Douglas Crockford"}` : Les virgules permettent de séparer les **membres** d'un tableau ou, comme ici, d'un objet .

> A noter : pas de virgule pour le dernier membre d'un objet

Le format Json

```
{
  "fruits": [
    { "kiwis": 3,
      "mangues": 4,
      "pommes": null
    },
    { "panier": true }
  ],
  "legumes": {
      "patates": "amandine",
      "poireaux": false
    },
    "viandes": ["poisson","poulet","boeuf"]
 }
```

- Il y a 3 **membres** : `fruits` , `legumes` et `viandes`
- `fruits` est constitué d'un **tableau** qui contient 2 **objets** : le premier objet contient 3 membres et le second un membre
- `legumes` est défini par un objet constitué par 2 membres
- `viandes` est défini par un tableau de 3 éléments

Ansible en mode ad hoc

> Utilisation du module `setup` pour interroger le nom de l'utilisateur connecté via Ansible sur le serveur `centos`

```
$ ansible -i inventaire -m setup -a 'filter=ansible_user_id' all
```

```
centos | SUCCESS => {
    "ansible_facts": {
        "ansible_user_id": "alex"
    },
    "changed": false
}
```

> Le code du module `setup` est accessible dans le package des librairie Python `/usr/lib/python2.7/dist-packages/ansible/modules/system/setup.py` (sur un Ubuntu)

> Utiliser `sudo` pour ne pas utiliser le `root`
> La commande pour ajouter une ligne dans le fichier `/etc/sudoers`

```
$ ansible -i inventaire \
  -m lineinfile \
  -a "path=/etc/sudoers \
  line='alex ALL=(ALL) ALL'" \
  --become-method=su --become --ask-become-pass all
```

- `-a` : argument
- `--become-method=su` : passe en tant que root avec la méthode `su`
- `--become` : escalade des privilèges en `root`

> La ligne `alex ALL=(ALL) ALL` a été ajoutée

```
$ grep alex /etc/sudoers
```

```
alex ALL=(ALL) ALL
```

> Connexion avec l'utilisateur `alex` qui a maintenant le privilège `SUDO`

```
[alex@localhost ~]$ sudo -l
```

```
~
L'utilisateur alex peut utiliser les commandes suivantes sur localhost :
    (ALL) ALL
```

Résumé avant d'aller plus loin

- Le mouvement DevOps
- Présentation générale de Ansible
- Les aspects et le concept technique de Ansible
- Les différentes installations de Ansible (paquet système, virtualenv avec les notions de versions différentes de Python et Ansible sur un même environnement)
- La configuration de SSH pour se connecter à un serveur distant
- Les inventaires des serveurs et leurs compositions
- Le format Json
- Ansible en mode ad doc avec la notion d'`inventaire` et l'utilisation de quelques modules pour interroger des valeurs d'environnement, puis en modifiant le fichier `sudoers` pour ajouter des privilèges à un utilisateur