Ansible est une solution Open-source, développée en langage Python par Michael DeHaan en 2012. Ansible est une plate-forme logicielle librepour la configuration et la gestion des ordinateurs. Elle combine le déploiement de logiciels multi-nœuds, l'exécution des tâches ad-hoc, et la gestion de configuration. Ansible ne nécessite pas de socle « client » pour les machines à déployer : il se base sur le protocole SSH et un interpréteur Python. Pas besoin de serveur imposant pour l'exploiter, carun simple ordinateur Linux peut suffire. Red Hat rachète Ansible Inc. en octobre 2015.

La documentation de Ansible se situe ici : https://docs.ansible.com/

SOMMAIRE

Configuration

Pour cet exemple on va chercher à installer git sur nos serveurs. On commence par écrire unfichier hosts qui va contenir la liste de nos serveurs.

```
[web]
192.168.1.150
```

Ici on précise que l'on a un groupe "web" qui contient un seul serveur. On peut définir plusieurs groupes afin de séparer les serveurs suivant la configuration à y effectuer (si on ades serveurs pour la base de données et des serveurs pour le serveur web par exemple). Ensuite nous allons créer un fichier YAML qui va permettre de décrire notre recette :

```
---
- name: Installation des serveurs web
  hosts: web
  remote_user: root
  tasks:
    - name: Installation de Git
      apt: name=git state=latest
...
```

Le module APT permet d'utiliser la commande apt afin d'installer différents packages sur votre serveur. L'avantage est qu'Ansible n'effectuera pas cette tâche à chaque fois. La seconde fois que vous lancerez le déploiement, il détectera que Git est déjà installé et nefera rien.

Enfin pour lancer le déploiement :

```
ansible-playbook -i hosts playbook.yml
```

Les rôles

Afin de mieux organiser nos taches, on peut les séparer dans des rôles afin de pouvoir lesréutiliser plus simplement.

```
roles
 |-- <nom du role>
        |--- tasks
              |--- main.yml
              |--- installation.yml
              |--- configuration.yml
        |--- handlers
              |--- main.yml
        |--- defaults
              |--- main.yml
```

- Le dossier tasks va contenir les taches, le fichier main.yml sera chargé par défaut et peut inclure d'autres fichiers de taches via un include.
- Le dossier handlers va contenir des taches à effectuer lors de la réussite d'autrestaches (comme le redémarrage d'nginx par exemple).
- Le dossier defaults va contenir les variables par défaut pour ce role. Ces variablespeuvent être modifiées dans le playbook qui chargera ce role.

Les handlers

Les handlers permettent de gérer des taches qui doivent être effectuées en cas de changement d'une tache précédente. Par exemple lors de l'installation d'nginx on souhaite supprimer la configuration par défaut. Suite à cette suppression on doit notifier nginx pour qu'il recharge notre configuration. On va ainsi commencer par créer un handler pour gérer lerechargement d'nginx :

```
# handlers/main.yml
---
- name: nginx reload
  service: name=nginx state=reloaded
```

Ensuite dans notre série de tache nginx :

```
---
- name: Install
  apt: name=nginx state=latest

- name: Start
  service: name=nginx state=started enabled=true

- name: Supprimer default.conf
  file: path=/etc/nginx/sites-enabled/default state=absent
  notify: nginx reload
```

Les variables

Une configuration doit s'adapter suivant les besoins ! Pour cela on peut définir des variablesque l'on pourra utiliser dans nos recettes. Par exemple on va paramétrer la timezone à utiliser pour php. Dans notre fichier defaults/main.yml ou dans la partie vars de notres playbook :

```
php_timezone: Europe/Paris
```

On peut alors injecter cette variable dans notre tache

```
- lineinfile: dest=/etc/php/7.0/fpm/php.ini regexp='date.timezone[\s]?=' line='date.timezone = {{ php_timezone }}'
```

On peut aussi utiliser des listes

```
php_packages:
  - php7.0
  - php7.0-common
  - php7.0-cli
  - php7.0-intl
  - php7.0-curl
  - php7.0-cgi
  - php7.0-fpm
  - php7.0-mcrypt
```

```
  - php7.0-mysql
  - php7.0-gd
# Que l'on utilisera Ensuite
- name: Installation du package {{ item }}
  apt: name={{ item }} state=latest
  with_items: php_packages
```

Les templates

Ces variables peuvent aussi être utilisé dans des templates. Par exemple on souhaite créerune configuration nginx modulable :

```
server{
  server_name www.{{ item.domain }};
  return 301 $scheme://{{ item.domain }}$request_uri;
}

server{
  server_name {{ item.domain }};
  root /var/www/{{ item.domain }};

  {% if item.php %}
  index index.php index.html;

  location ~ \.php$ {
      try_files $uri =404;
      fastcgi_pass unix:/run/php/php7.0-fpm.sock;
      fastcgi_index index.php;
      fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
      include fastcgi_params;
  }

  location / {
      try_files $uri $uri/ /index.php?$query_string;
  }
  {% else %}
  index index.html;
  {% endif %}
}
```

Par défaut, Ansible utilise le moteur de template Jinja que l'on peut appeler depuis le moduleTemplate.

```
- name: Création de la configuration nginx
  template: src=templates/nginx.j2 dest=/etc/nginx/sites-available/{{ item.domain }} force=yes
  notify: nginx reload
```