Simplifier la configuration et le déploiement de vos serveurs avec Ansible

Capitoul – 17 Octobre 2019
rosalie.viala@ut-capitole.fr

« Un **ansible** est un dispositif théorique permettant de communiquer à une vitesse supraluminique. »

Extrait Wikipédia
Ursula Le Guin
Rocannon's World

Ansible : Sommaire

- **Présentation générale**
- **Ansible en pratique**
  - Pour commencer
  - Les playbooks
  - Les rôles
  - Contrôle des serveurs windows
  - Pour aller plus loin
- **Pourquoi choisir Ansible ?**
- **Ansible à UT1**

Ansible : Présentation générale

Ansible : Outil de déploiement, d'automatisation et de gestion de configuration.

- Logiciel libre créée par Micheal Dhaan (Cobbler)
- Développé en Python
- Première version 20 février 2012
- Red Hat depuis octobre 2015
- Version actuelle 2.8

➔ Ansible Engine : Version professionnelle Red Hat

➔ Ansible Project : Version communautaire

Ansible : Présentation générale

- Agentless
- Langage de configuration YAML
- Mode push

- Installation rapide « Control node », sous CentOS :
  - yum install epel-release
  - yum install ansible
- Python version > 2.6 ou 3.5
- Windows n'est pas supporté

- Gestion des machines clientes « Managed node » par SSH

- Prérequis « Managed Node » :

Python version > 2.6 ou 3.5

Ansible en pratique : pour commencer

- Fichier d'inventaire « **hosts** »

Liste de tous les serveurs connus par Ansible + déclaration des groupes.

- Les **« modules » :**
  - librairies python directement utilisables pour exécuter diverses opérations de configuration sur les systèmes distants
  - nombreux domaines techniques nativement gérables par Ansible
  - développer ses propres modules

- Quelques modules (*extrait documentation ansible*) :
  - systemd – Manage services
  - user – Manage user accounts
  - vmware_host_config_manager – Manage advanced system settings of an ESXi host
  - dnf – Manages packages with the dnf package manager
  - nagios – Perform common tasks in Nagios related to downtime and notifications
  - proxmox_kvm – Management of Qemu(KVM) Virtual Machines in Proxmox VE cluster
  - mysql_db – Add or remove MySQL databases from a remote host
  - shell – Execute shell commands on targets
  - cron – Manage cron.d and crontab entries

Ansible en pratique : pour commencer

- En ligne de commande :

ansible *mon_serveur* -m copy -a "src=/etc/httpd/conf.d/ssl.conf dest=/usr/local/ssl.conf mode=755 owner=root group=root"

```
applissys2.ut-capitole.fr | CHANGED => {
    "changed": true,
    "checksum": "180851eabb81b4150d392210ded7b692b760e03a",
    "dest": "/usr/local/ssl.conf",
    "gid": 0,
    "group": "root",
    "md5sum": "b303e888be6ac2888e816e7aa4668c41",
    "mode": "0755",
    "owner": "root",
    "secontext": "system_u:object_r:usr_t:s0",
    "size": 9443,
    "src": "/root/.ansible/tmp/ansible-tmp-1571121952.65-171443493127329/source",
    "state": "file",
    "uid": 0
}
```

Ansible en pratique : pour commencer

- Pour voir la documentation des modules :

ansible-doc at

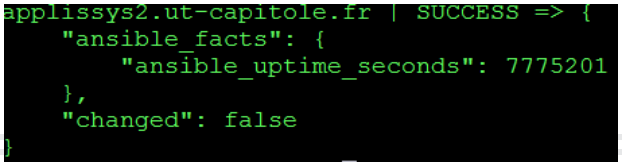
```
[root@puppet ~]# ansible-doc at
  * This module is maintained by The Ansible Core Team
OPTIONS (= is mandatory):

- command
        A command to be executed in the future.
        [Default: (null)]
        type: str

= count
        The count of units in the future to execute the command or script file.

        type: int

- script_file
        An existing script file to be executed in the future.
        [Default: (null)]
        type: str

- state
        The state dictates if the command or script file should be evaluated as present(added) or
        absent(deleted).
        (Choices: absent, present)[Default: present]
        type: str

- unique
        If a matching job is present a new job will not be added.
        [Default: False]
        type: bool

= units
        The type of units in the future to execute the command or script file.
        (Choices: minutes, hours, days, weeks)
        type: str


REQUIREMENTS:  at

AUTHOR: Richard Isaacson (@risaacson)
        METADATA:
          status:
          - preview
          supported_by: core
EXAMPLES:

- name: Schedule a command to execute in 20 minutes as root
  at:
    command: ls -d / >/dev/null
    count: 20
    units: minutes

- name: Match a command to an existing job and delete the job
  at:
    command: ls -d / >/dev/null
    state: absent

- name: Schedule a command to execute in 20 minutes making sure it is unique in the queue
  at:
    command: ls -d / >/dev/null
    count: 20
    units: minutes
    unique: yes
```

Ansible en pratique : pour commencer

- ansible *mon_serveur* -m setup

Récupère les « facts » = propriétés système lues par Ansible :

- ansible mon_serveur -m setup -a "filter=*uptime*"



Ansible en pratique : les playbooks

- Les **« playbooks »** : scénario de « **tasks**»
- Une **tâche** fait appel à des modules, éventuellement à des handlers, des variables et des templates

installation.yml :

```
- hosts: applissys2.ut-capitole.fr
  tasks:
  - name: Copie de fichier
    copy:
      src: /etc/httpd/conf.d/ssl.conf
      dest: /usr/local/ssl.conf
      mode: 0755
      owner : root
      group : root

  - name: Installation de paquets
    yum:
       name: "{{packages}}"
       state: present
    vars:
       packages:
        - sysstat
        - vim
        - rsync
```

Ansible en pratique : les playbooks

⇒ Exécution : commande **ansible-playbook**

ansible-playbook installation.yml

1er lancement :

```
PLAY [applissys2.ut-capitole.fr] ***********************************************************************************

TASK [Gathering Facts] *********************************************************************************************
ok: [applissys2.ut-capitole.fr]

TASK [Copie de fichier] ********************************************************************************************
changed: [applissys2.ut-capitole.fr]

TASK [Installation de paquets] *************************************************************************************
ok: [applissys2.ut-capitole.fr]

PLAY RECAP *********************************************************************************************************
applissys2.ut-capitole.fr  : ok=3    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

2ème lancement :

```
PLAY [applissys2.ut-capitole.fr] ***********************************************************************************

TASK [Copie de fichier] ********************************************************************************************
ok: [applissys2.ut-capitole.fr]

TASK [Installation de paquets] *************************************************************************************
ok: [applissys2.ut-capitole.fr]

PLAY RECAP *********************************************************************************************************
applissys2.ut-capitole.fr  : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

Ansible en pratique : les playbooks

- **Handlers** : actions déclenchées lors d'un changement d'état
- **Variables** :
  - Facts
  - à la demande à l'intérieur d'un playbook
  - fichier de variables dans les rôles
  - fichier d'inventaire
- **Template** : fichier modèle, personnalisable à partir du langage Jinja2

Ansible en pratique : les playbooks

- Applicables sur un hôte, un groupe, « all »
- Traitement en série des tâches sur chaque hôte
- Une erreur entraine l'arrêt de l'exécution du playbook sur le serveur concerné

- Quelques options utiles avec la commande **ansible-playbook** :

--check → mode dry-run

--syntax-check → vérification de la syntaxe à l'intérieur du playbook

--extra-vars → fixer des valeurs à des variables

--limit → filtre d'application du playbook sur un host/group

--diff → voir les différences avant/après application de la tâche

Ansible en pratique : les playbooks

- **Exemple d'utilisation des facts avec debug :**

```
- hosts: applissys2.ut-capitole.fr
  tasks:
  - name: Afficher IP
    debug:
      msg: "Ma distrib : {{ansible_distribution}}"
```

```
TASK [Afficher IP] ********************
ok: [applissys2.ut-capitole.fr] => {
    "msg": "Ma distrib : CentOS"
```

- **Condition :** application des tâches après validation de certaines conditions **when**

```
- name: Installation de paquets
  yum:
     name: "{{packages}}"
     state: present
  vars:
     packages:
      - sysstat
      - vim
      - rsync
  when: ansible_distribution == 'CentOS'
```

- **Boucle :** répétition de tâche

```
- name: Ajouter les utilisateurs de test
  user:
    name: "{{item}}"
    state: present
    groups: "test"
  loop:
    - user1
    - user2
```

Ansible en pratique : les rôles

- Ansible utilise les « **roles** » :

  un rôle regroupe un ensemble organisé de fichiers de configuration YAML et de fichiers références facilement réutilisables

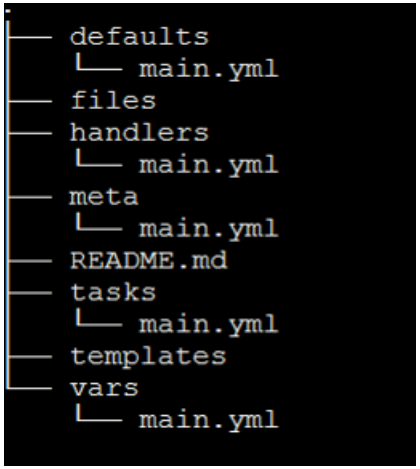
- Ansible Galaxy :

  site web de partage de rôles développés par la communauté

- Gestion des rôles commande **ansible-galaxy**

  - Pour créer un nouveau rôle vide :

    ansible-galaxy init *mon_role*



Ansible en pratique : les rôles

- Récupérer un rôle depuis ansible-galaxy :
  ansible-galaxy install *auteur.role_name*
- Recherche de rôle sur ansible-galaxy :
  ansible-galaxy search apache --auteur *nom_auteur*
- Lister les rôles installés et leurs versions :
  ansible-galaxy list

- Utilisation dans un playbook :

```
- hosts:
      all
  roles:
    - apache
    - mysql
    - client-nagios
```

Ansible en pratique : contrôle des serveurs Windows

- Le serveur Ansible communique avec les serveurs Windows grâce à :
  - WinRM → Powershell
  - Win32-OpenSSH depuis Ansible 2.8 (expérimental)
- Prérequis « Control Node » :
  - Library python pywinrm
- Prérequis « Managed Node » :
  - PowerShell version 3.0
  - .NET Framework 4.0
  - Listener WinRM actif
- Eléments problématiques :
  - Authentification : NTLM, Kerberos, Certificats
  - Elévation de privilèges : become_user, become_method
- Modules dédiés Windows (*extrait documentation ansible*) :
  - win_shell – Execute shell commands on target hosts
  - win_user – Manages local Windows user accounts
  - win_package – Installs/uninstalls an installable package
  - win_copy – Copies files to remote locations on windows hosts

Ansible en pratique : pour aller plus loin

- **tags** : dans un *playbook* permet de n'exécuter qu'une partie de celui-ci
- **block** : dans un *playbook* autorise le traitement groupés de plusieurs tâches
- **include_tasks :** dans un playbook exécution des tâches d'un fichier .yml intermédiaire
- **register** : enregister le résultat d'une tâche
- **vault** : sécurises les données sensibles
- **ansible-lint** : pour vérifier la « qualité » de votre playbook
- Gestion des erreurs : comportement par défaut adaptable par des fonctionnalités avancées (serial, ignore_errors, failed_when…)

- API Ansible
- **Ansible Tower** (offre commerciale): propose une interface graphique de management pour Ansible : état en temps réel des déploiements, historique, scheduler…
- Recueil des Best Pratice : https://docs.ansible.com/ansible/latest/user_guide/playbooks_best_practices.html

Pourquoi choisir Ansible ?

- Facilité de mise en place de la solution : pas d'agent
- Large éventail de modules et rôles(Ansible Galaxy)
- Flexibilité → playbooks
- Simplicité du langage YAML
- Communauté étendue
- Polyvalence → interaction à différents niveaux : systèmes d'exploitations, applications, équipements réseaux

Un outil presque parfait pour…

- La destruction massive
- Développer moins de scripts
- Boire plus de café
- Améliorer son organisation de travail
- Ne plus rédiger de documentation technique

Ansible à UT1

- Passage de Puppet à Ansible en 2018 (version communautaire)
- Plus de 200 serveurs sous Ansible
- 35 sous windows
- 50 roles
- 3 roles Ansible Galaxy

Questions ?

Liens utiles

- https://docs.ansible.com/ansible/latest/index.html
- https://docs.ansible.com/ansible/latest/user_guide/modules.html
- https://docs.ansible.com/ansible/devel/modules/list_of_windows_modules.html

- https://www.tartarefr.eu/ansible-par-la-pratique-premiere-partie-les-bases/
- https://www.tartarefr.eu/ansible-par-la-pratique-deuxieme-partie-premiers-playbooks-avec-les-roles/
- https://www.tartarefr.eu/ansible-par-la-pratique-troisieme-partie-utilisation-avancee-de-nos-playbooks/