Ansible

Automatisez votre infrastructure

Killian & **pi**

Et consultez nos formations sur formations.minet.net

Les problèmes

Je veux modifier un paramètre dans mes configs ssh.

```
7 PasswordAuthentication yes
8 PermitRootLogin yes
9 PrintMotd no
```

Les problèmes

Je veux modifier un paramètre dans mes configs ssh.

```
7 PasswordAuthentication yes
8 PermitRootLogin no
9 PrintMotd no
```

Les problèmes

Je veux modifier un paramètre dans mes configs ssh.

```
7 PasswordAuthentication yes
8 PermitRootLogin no
9 PrintMotd no
```

bash — me@bastion

```
$ vim /etc/ssh/sshd_config/config
$ ssh me@server1
$ ssh me@server2
... (imaginez avec une centaine de serveurs !)
```

Les problèmes

Je veux re-déployer la même chose que sur un autre serveur.

- Lire la doc
- Regarder ce qu'on avait fait
- Copier / coller des choses ?

Et si je veux avoir de légères différences ?

Ce qu'il faut

- Un moyen de toucher plusieurs serveurs en même temps
- Un moyen de reproduire sans erreur des procédures
- Un outil simple et modulaire

Les solutions !

-  Ansible
-  NixOS
-  pyinfra
-  terraform
- puppet, chef...

Ansible

- Mature
- Agentless
- Utilise SSH (**entre autres !**)
- Riche (communautaire)
- Open source !

Présentation rapide

Installation

bash — localhost

```
$ python3 -m venv venv
$ source venv/bin/activate
$ pip install ansible
...
Successfully installed MarkupSafe-3.0.3 PyYAML-6.0.3 ansible-1
3.3.0 ansible-core-2.20.2 cffi-2.0.0 cryptography-46.0.5 jinja
2-3.1.6 packaging-26.0 pycparser-3.0 resolvelib-1.2.1
```

Exemple

bash — localhost

```
$ ansible -m ansible.builtin.ping all
ansible2 | SUCCESS => {
  "changed": false,
  "ping": "pong"
}
ansible1 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
```

Exemple

bash — localhost

```
$ ansible-playbook playbooks/ping.yaml
PLAY [all] *********************************************************

TASK [Ping to test connectivity] ***********************************
ok: [ansible1]
ok: [ansible2]

PLAY RECAP *********************************************************
ansible1 : ok=1    changed=0    unreachable=0    failed=0    ski
pped=0    rescued=0    ignored=0
ansible2 : ok=1    changed=0    unreachable=0    failed=0    ski
pped=0    rescued=0    ignored=0
```

Ok, mais il s'est passé quoi, là ?

Exemple

Regardons le fichier `playbooks/ping.yaml`

On peut constater que c'est du yaml

```yaml
- hosts: all
  tasks:
    - name: Ping to test connectivity
      ansible.builtin.ping:
```

Pour vous aider à comprendre le yaml, voici ce que donnerait exactement la même configuration en JSON

```json
[{
    "hosts": "all",
    "tasks": [
      {"name": "Ping to test connectivity",
       "ansible.builtin.ping": null}
    ]
}]
```

En détail

```yaml
- hosts: all
  tasks:
    - name: Ping to test connectivity
      ansible.builtin.ping:
```

bash — localhost

```
$ ansible-playbook playbooks/ping.yaml
PLAY [all] ***************************************

TASK [Ping to test connectivity] *********************
ok: [ansible1]
ok: [ansible2]

PLAY RECAP ***************************************
ansible1 : ok=1    changed=0    unreachable=0    failed
=0    skipped=0    rescued=0    ignored=0
ansible2 : ok=1    changed=0    unreachable=0    failed
=0    skipped=0    rescued=0    ignored=0
```

Les différentes parties de Ansible

- Playbooks
- Modules
- Inventaire
- Jinja2
- Autres (roles, plugins, collections...)

Playbooks

C'est juste un fichier `yaml` avec une liste de choses à exécuter !

Mais quoi exécuter ?

Tâches

```yaml
- name: Ping to test connectivity
  ansible.builtin.ping:
- name: Install SSSD
  register: result
  ansible.builtin.apt:
    name: sssd
    state: present
- name: Copy config
  if: "some condition..."
  ansible.builtin.copy:
    src: ./sssd.conf
    dest: /etc/sssd/sssd.conf
    owner: root
    group: root
    mode: "0600"
```

- Mots-clefs relatif aux tâches : ici
- Modules disponibles : ici

Modules

```yaml
- name: Copy config
  if: "some condition..."
  ansible.builtin.copy:
    src: ./sssd.conf
    dest: /etc/sssd/sssd.conf
    owner: root
    group: root
    mode: "0600"
```

This is the **latest** (stable) Ansible community documentation. For Red Hat Ansible Automation Platform subscriptions, see Life Cycle (https://access.redhat.com/support/policy/updates/ansible-automation-platform) for version details.

Important: The ansible-core 2.19/Ansible 12 release has made **significant templating changes that might require you to update playbooks and roles**. The templating changes enable reporting of numerous problematic behaviors that went undetected in previous releases, with wide-ranging positive effects on security, performance, and user experience. You should validate your content to ensure compatibility with these templating changes before upgrading to ansible-core 2.19 or Ansible 12. See the porting guide (https://docs.ansible.com/projects/ansible/devel/porting_guides/porting_guide_12.html) to understand where

Modules

```yaml
- name: Copy config
  if: "some condition..."
  ansible.builtin.copy:
    src: ./sssd.conf
    dest: /etc/sssd/sssd.conf
    owner: root
    group: root
    mode: "0600"
```

Ça peut être équivalent à faire ça:

bash — localhost

```
$ scp ./sssd.conf user@remote:/etc/sssd/sssd.conf
$ chown root:root /etc/sssd/sssd.conf
$ chmod "0600" /etc/sssd/sssd.conf
```

C'est ansible qui execute les actions, nous on ne dit que ce que l'on **souhaite** !

Modules

Les modules essayent d'être **idempotents** !

C'est à dire que demander 2 fois la même action ne l'exécutera qu'une fois.

bash — localhost

```
$ ansible-playbook playbooks/copy_task.yaml
...
TASK [Copy config] *********************************************
changed: [ansible1]
changed: [ansible2]
...

$ ansible-playbook playbooks/copy_task.yaml
...
TASK [Copy config] *********************************************
ok: [ansible1]
ok: [ansible2]
...
```

Quelques modules utiles

ansible.builtin.apt

Installez des packets en utilisant APT

```yaml
- name: Install apache httpd (state=present is optional)
  ansible.builtin.apt:
    name: apache2
    state: present
- name: Update repositories cache and install "foo" package
  ansible.builtin.apt:
    name: foo
    update_cache: yes

- name: Remove "foo" package
  ansible.builtin.apt:
    name: foo
    state: absent
```

Lien vers la documentation : ici

Quelques modules utiles

ansible.builtin.copy

Copiez des fichiers

```yaml
- name: Copy file with owner and permissions
  ansible.builtin.copy:
    src: /srv/myfiles/foo.conf
    dest: /etc/foo.conf
    owner: foo
    group: foo
    mode: '0644'
- name: Copy using inline content
  ansible.builtin.copy:
    content: '# This file was moved to /etc/other.conf'
    dest: /etc/mine.conf
- name: Copy a new "sudoers" file into place, after passing validation with visudo
  ansible.builtin.copy:
    src: /mine/sudoers
    dest: /etc/sudoers
    validate: /usr/sbin/visudo -csf %s
```

Lien vers la documentation : ici

Quelques modules utiles

ansible.builtin.file

Gérez vos fichiers

```yaml
- name: Change file ownership, group and permissions
  ansible.builtin.file:
    path: /etc/foo.conf
    owner: foo
    group: foo
    mode: '0644'
- name: Create a directory if it does not exist
  ansible.builtin.file:
    path: /etc/some_directory
    state: directory
    mode: '0755'
- name: Remove file (delete file)
  ansible.builtin.file:
    path: /etc/foo.txt
    state: absent
```

Lien vers la documentation : ici

Quelques modules utiles

ansible.builtin.shell

Executez des commandes shell

```
1 - name: Execute the command in remote shell
2   ansible.builtin.shell: somescript.sh >> somelog.txt
```

Lien vers la documentation : ici

Quelques modules utiles

ansible.builtin.systemd_service

Gérez vos services systemd

```yaml
- name: Make sure a service unit is running
  ansible.builtin.systemd_service:
    state: started
    name: httpd
- name: Stop service cron, if running
  ansible.builtin.systemd_service:
    name: cron
    state: stopped
- name: Reload service httpd, in all cases
  ansible.builtin.systemd_service:
    name: httpd.service
    state: reloaded
- name: Restart service cron, also issue daemon-reload to pick up config changes
  ansible.builtin.systemd_service:
    state: restarted
    daemon_reload: true
    name: cron
```

Lien vers la documentation : ici

Quelques modules utiles

Et bien bien d'autres encore...

N'hésitez pas à chercher sur google, ou demander à ChatGPT le nom d'un module pour faire quelque chose.

Une fois le module trouvé, lisez la documentation, il y a toujours des exemples !

Lien vers la documentation : ici

L'inventaire

A quoi correspond `hosts: all` ?

L'inventaire

Pour ça, il suffit de regarder un **fichier d'inventaire**.

Par exemple, `inventory.yaml`:

```yaml
production:
  hosts:
    gitlabint.minet.lan:
  children:
    dns_servers:
      hosts:
        ns1.minet.lan:
        ns2.minet.lan:
```

L'inventaire

Pour ça, il suffit de regarder un **fichier d'inventaire**.

Par exemple, `inventory.yaml`:

```yaml
production:
  vars:
    ansible_user: airopi
  hosts:
    gitlabint.minet.lan:
  children:
    dns_servers:
      hosts:
        ns1.minet.lan:
        ns2.minet.lan:
          ansible_user: root
```

Pour plus d'infos, référez-vous à la documentation.

Jinja2

```
production:
  hosts:
    ansible1:
      ansible_host: 172.31.0.22
      my_variable: "foo"
    ansible2:
      ansible_host: 172.31.0.23
      my_variable: "bar"
```

Jinja2

```yaml
production:
  hosts:
    ansible1:
      ansible_host: 172.31.0.22
      my_variable: "foo"
    ansible2:
      ansible_host: 172.31.0.23
      my_variable: "bar"
```

On créer un playbook simple:

```yaml
- hosts: all
  tasks:
    - name: "Create file with content"
      content: "{{ my_variable }}"
      dest: "/root/my_file.txt"
```

bash — localhost

```
$ ssh root@172.31.0.22 cat /root/my_file.txt
foo
$ ssh root@172.31.0.23 cat /root/my_file.txt
bar
```

Et tant d'autres choses à découvrir...

Lien vers le tp

https://wiki.minet.net/fr/tps/tp-formations/ansible