



## Support pédagogique Ansible - 3

- Le chiffrement des fichiers
- Chiffrement d'un champ
- Ansible et les machines sous Windows
- Le moteur de template Jinja
- Les Templates Jinja
- Exécuter des tâches à plusieurs niveaux
- Délégation des tâches
- Installation d'un serveur Apache et configuration avec un fichier Template
- Les tags
- Exécution conditionnée
- Les handlers

# Le chiffrement des fichiers

Certaines informations sensibles sont stockées en clair dans les fichiers de configurations (mot de passe, identifiants...) et ces fichiers sont parfois partagés ou accessibles trop facilement. Il est donc nécessaire de les protéger.

- On chiffre le fichier `intranet-pass.yml` contenant deux variables `user` et `pass`

```
$ ansible-vault encrypt intranet-pass.yml
```

```
New Vault password:  
Confirm New Vault password:  
Encryption successful
```

- Le contenu du fichier après le chiffrement

```
$ cat intranet-pass.yml
```

```
$ANSIBLE_VAULT;1.1;AES256
34386434386464333237353636383738323[....]6362663133386561643538
39663733363830333130643264366338373930343665653630623633613962393438
```

- Le contenu des variables une fois déchiffré par Ansible avec l'option `-ask-vault-pass`

```
$ ansible -m debug -a var=user,pass -e @intranet-pass.yml \
--ask-vault-pass localhost
```

Vault password:

```
localhost | SUCCESS => {
  "user,pass": "(u'alex', u'A!210')"
```

# Utilisation d'un fichier pour stocker le mot de passe de chiffrement

- Le fichier qui va contenir le mot de passe

```
$ echo alexandre > vault.key
```

- Chiffrement d'un fichier avec le mot de passe contenu dans le fichier  
vault.key

```
$ ansible-vault encrypt intranet-pass.yml --vault-password-file vault.key
```

```
Encryption successful
```

- Lecture du fichier chiffré contenant les variables

```
$ ansible -m debug -a var=user,pass -e @intranet-pass.yml \
  --vault-password-file vault.key localhost
```

```
localhost | SUCCESS => {
  "user,pass": "(u'alex', u'A!210')"
```

- La variable `vault_password_file` peut être définie directement dans le fichier de configuration de Ansible

```
$ grep vault_password_file /etc/ansible/ansible.cfg
```

```
#vault_password_file = /path/to/vault_password_file
```

- Création d'un fichier `ansible.cfg` dans le répertoire de travail avec le chemin du fichier password

```
$ cat ./ansible.cfg
```

```
[defaults]
```

```
vault_password_file = /home/alex/.ansible/dawan/vault.key
```

- Les commandes Ansible peuvent être utilisées directement sans option de chiffrement

```
$ ansible -m debug -a var=user,pass -e @intranet-pass.yml localhost
```

```
localhost | SUCCESS => {  
    "user,pass": "(u'alex', u'A!210')"  
}
```

- Il est également possible d'exporter dans l'environnement courant le nom du fichier mot de passe

```
$ export DEFAULT_VAULT_PASSWORD_FILE=vault.key
```

## Modifier le fichier chiffré

```
$ ansible-vault edit intranet-pass.yml
```

## Changement du mot de passe de chiffrement

```
$ ansible-vault rekey intranet-pass.yml
```

```
Vault password:  
New Vault password:  
Confirm New Vault password:  
Rekey successful
```

- En utilisant des fichiers de mot de passe

```
$ ansible-vault rekey \  
--new-vault-password-file ./new-vault.key \  
--vault-password-file ./vault.key \  
intranet-pass.yml
```



# Chiffrement d'un champ

Depuis la version 2.3 de Ansible, il est possible de chiffrer uniquement des champs dans un fichier, ce qui est plutôt pratique pour gérer le contenu des fichiers.

- Création d'une chaîne chiffrée avec `encrypt_string`

```
$ ansible-vault encrypt_string 'sdq44sd0SKz!' \  
--vault-password-file ./vault.key
```

```
!vault |  
      $ANSIBLE_VAULT;1.1;AES256  
      34336633656163653436333432613031323336643737[...]  
Encryption successful
```

- Le fichier `partial-vault.yml` contenant le chiffrement du pass

```
user: "alex"  
pass : !vault |  
        $ANSIBLE_VAULT;1.1;AES256  
        34336633656163653436333432613031323336643737[...]
```

- Affichage des variables user et pass

```
$ ansible -m debug -a var=user,pass localhost \  
        -e @partial-vault.yml --vault-password-file ./vault.key
```

```
localhost | SUCCESS => {  
    "user,pass": "(u'alex', u'sdq44sd0SKz!')"  
}
```

# Ansible et les machines sous Windows

Depuis la version version 1.7, Ansible prend en charge Windows. Ansible utilise le langage PowerShell (versions 3 et supérieure) pour gérer les machines Windows

- Il faut installer la dépendance `pywinrm`

```
$ pip install pywinrm
```

Télécharger une ISO W2016SRV pour les tests

Il est nécessaire de préparer le serveur `Windows` avec le script `ConfigureRemotingForAnsible.ps1` à l'emplacement suivant :  
<https://github.com/ansible/ansible/blob/devel/examples/scripts/ConfigureRemotingForAnsible.ps1>

- Sur le serveur `Windows`

Lancez une `console` PowerShell.

- Puis lancer le script (en vérifiant que l'utilisateur a les droits suffisants pour le faire)

```
.\ConfigureRemotingForAnsible.ps1
```

- Préparation du fichier `inventaire` pour les serveurs `Windows`

```
[win]
windows1

[win:vars]
ansible_user=Administrator
ansible_password= variable à chiffrer ou mettre dans le fichier pass
ansible_connection=winrm
ansible_winrm_server_cert_validation=ignore
```

- Test de ping sur le serveur `Windows`

```
$ ansible -i win.inv -m win_ping all
```

```
windows1 | SUCCESS => {
  "changed": false,
  "ping": "pong"
}
```

# Installation d'un package

- L'installation du paquet `procexp` va se faire via le gestionnaire `chocolatey` sur le serveur `Windows`

```
- name: "Install paquet procexp"
  hosts: win
  gather_facts: no
  tasks:
    - name: "Install procexp"
      win_chocolatey:
        name: "procexp"
        state: present
```

```
$ ansible-playbook -i win.inv procexp.yml
```

```
PLAY [Install paquet procexp] *****

TASK [Install procexp] *****
changed: [windows1]

PLAY RECAP *****
windows1 : ok=1 changed=1 unreachable=0 failed=0
```

# Le moteur de template Jinja

Jinja2 est un langage de template moderne et convivial pour Python.

<http://jinja.pocoo.org/docs/dev/templates/>

Il est utilisé comme modèle de fichier contenant des variables exécutées en fonction du contexte.

- Un exemple de fichier avec la variable `inventory_hostname`

```
# jinja.html.j2
<html>
  <head>
    <title>Machine {{inventory_hostname}}</title>
  </head>
  <body>
    <p>Cette machine s'appelle {{inventory_hostname}}</p>
  </body>
</html>
```

- Le fichier inventaire avec deux serveurs

```
cat inventaire-jinja  
www1  
www2
```

- Le fichier de configuration utilisant le module `template` pour générer deux fichiers `HTML` contenant le nom des serveurs

```
$ cat jinja.yml  
  
---  
  
- name: "Generate html file for each host"  
  hosts: all  
  connection: local  
  tasks:  
    - name: "html file generation"  
      template:  
        src: "jinja.html.j2"  
        dest: "{{playbook_dir}}/{{inventory_hostname}}.html"
```



- Le fichier de configuration est joué sur l'inventaire créé précédemment

```
$ ansible-playbook -i inventaire-jinja jinja.yml
```

```
PLAY [Generate html file for each host] *****
```

```
TASK [Gathering Facts] *****
```

```
ok: [www1]
```

```
ok: [www2]
```

```
TASK [html file generation] *****
```

```
changed: [www2]
```

```
changed: [www1]
```

```
PLAY RECAP *****
```

```
www1                : ok=2    changed=1    unreachable=0    failed=0
```

```
www2                : ok=2    changed=1    unreachable=0    failed=0
```

- Les deux fichiers `HTML` sont bien présents et contiennent respectivement la valeur des variables demandées (`inventory_hostname`)

```
$grep title www*.html
```

```
www1.html:      <title>Machine www1</title>  
www2.html:      <title>Machine www2</title>
```

# Les Templates Jinja

La notion de template va être illustrée en interrogeant les informations remontées avec le module `setup` (champ `gather_facts` )

Les `facts` fournissent un nombre impressionnant d'information sur les serveurs interrogés.

```
$ ansible -m setup localhost
```

le module `setup` retourne plus de 800 lignes

```
localhost | SUCCESS => {  
  "ansible_facts": {  
    "ansible_all_ipv4_addresses": [  
      "192.168.0.13",  
      "192.168.122.1"  
    ],  
    "ansible_all_ipv6_addresses": [  
      "fe80::3b02:2621:6bd0:8d4b"  
    ],  
    "ansible_apparmor": {  
      "status": "enabled"  
    },  
    "ansible_architecture": "x86_64",  
    "ansible_bios_date": "02/26/2018",  
    "ansible_bios_version": "2.6.2",  
    "ansible_cmdline": {  
      "BOOT_IMAGE": "/boot/vmlinuz-4.15.0-23-generic",  
      "quiet": true,  
      "ro": true,  
      "root": "UUID=86715eb6-c478-42b6-b0da-92e067525242",  
      "splash": true,  
      "vt.handoff": "1"  
    },  
    "ansible_date_time": {  
      "date": "2018-06-24",  
      "day": "24",
```

- Récupération des variables `ansible_all_ipv4_addresses` pour les afficher dans un template `Jinja`

```
# template-net.html.j2
---

<html>
  <head>
    <title>Machine {{inventory_hostname}}</title>
  </head>
  <body>
    <p>Cette machine s'appelle {{inventory_hostname}}</p>
    <p>Ci-dessous la liste de ces adresses :</p>
    <ul>
{% for ip in ansible_all_ipv4_addresses %}
      <li>{{ip}}</li>
{% endfor %}
    </ul>
  </body>
</html>
```

La notion de boucle est ajoutée pour afficher toutes les adresses IP du serveur.

- Le fichier `jinja2.yml` qui fait appelle au fichier source `template-net.html.j2`

```
- name: "Generate html file for each host"
  hosts: all
  connection: local
  tasks:
    - name: "html file generation"
      template:
        src: "template-net.html.j2"
        dest: "{{playbook_dir}}/{{inventory_hostname}}.html"
```

- Le fichier de configuration est rejoué

```
$ ansible-playbook -i inventaire-jinja jinja2.yml
```

- Les deux fichiers `HTML` ont été modifiés pour intégrer les adresses `IP` dans le corps du fichier.

```
$ grep li *.html
```

```
www1.html:    <p>Ci-dessous la liste de ces adresses :</p>
www1.html:    <li>192.168.0.13</li>
www1.html:    <li>192.168.122.1</li>
www2.html:    <p>Ci-dessous la liste de ces adresses :</p>
www2.html:    <li>192.168.0.13</li>
www2.html:    <li>192.168.122.1</li>
```

On remarque ici que les adresses `IP` sont celles de `localhost`

# Exécuter des tâches à plusieurs niveaux

## Comment interroger un serveur distant et faire une action locale ?

Un exemple pour illustrer ce concept : récupérer l'adresse IP d'un serveur et modifier le template Jinja en local.

- Le fichier de configuration modifié avec `connection: local`, qui a été déplacé dans la tâche pour qu'elle soit exécutée en local

```
- name: "Generate html file for each host"
  hosts: all
  tasks:
    - name: "html file generation"
      template:
        src: "template-net.html.j2"
        dest: "{{playbook_dir}}/{{inventory_hostname}}.html"
      connection: local
```



- Lancement du `playbook`

```
$ ansible-playbook -i inventaire jinja3.yml
```

```
PLAY [Generate html file for each host] *****
```

```
TASK [Gathering Facts] *****
```

```
ok: [centos]
```

```
TASK [html file generation] *****
```

```
changed: [centos]
```

```
PLAY RECAP *****
```

```
centos                : ok=2    changed=1    unreachable=0    failed=0
```

- Le fichier local a été mis à jour avec les informations des serveurs distants

```
grep li *.html
```

```
centos.html:      <p>Ci-dessous la liste de ces adresses :</p>
```

```
centos.html:      <li>192.168.122.123</li>
```

## Délégation des tâches

Il est parfois utile de faire des actions différentes sur certains des serveurs de l'inventaire, `delegate_to` peut être utilisé pour ça.

Illustration avec un exemple : le fichier `template-net.html.j2` est en local, le `playbook` est lancé en local, les `facts` sont récupérés des deux serveurs (`centos` et `centos-clone`), les actions (création de répertoire et exécution du template) sont réalisées sur le serveur `centos`.

- L'inventaire avec 2 serveurs

```
$ cat inventaire-jinja
```

```
centos  
centos-clone
```

- Le fichier de configuration à jouer

```
- name: "Generate html file for each host"
  hosts: all
  gather_facts: yes
  vars:
    host_inventory: "centos"
    inventory_dir:  "/var/www/html/inventory"
  tasks:
    - name: "Create template directory"
      file:
        path: "{{inventory_dir}}"
        owner: "apache"
        group: "apache"
        mode: "0755"
        state: "directory"
      delegate_to: "{{host_inventory}}"
    - name: "html file generation"
      template:
        src: "template-net.html.j2"
        dest: "{{inventory_dir}}/{{inventory_hostname}}.html"
      delegate_to: "{{host_inventory}}"
```

- Exécution du `playbook`

```
$ ansible-playbook -i inventaire-jinja jinja4.yml -b -K
```

- Résultat des actions

```
PLAY [Generate html file for each host] *****

TASK [Gathering Facts] *****
ok: [centos]
ok: [centos-clone]

TASK [Create template directory] *****
ok: [centos-clone -> centos]
ok: [centos -> centos]

TASK [html file generation] *****
changed: [centos -> centos]
changed: [centos-clone -> centos]

PLAY RECAP *****
centos                : ok=3    changed=1    unreachable=0    failed=0
centos-clone          : ok=3    changed=1    unreachable=0    failed=0
```

- Sur le serveur Inventaire : centos

```
grep li\> /var/www/html/inventory/*.html
/var/www/html/inventory/centos-clone.html: <li>192.168.122.124</li>
/var/www/html/inventory/centos.html: <li>192.168.122.123</li>
```

- Sur le serveur centos-clone , le répertoire n'a pas été créé

```
ls /var/www/html/
www.html
```

Afin d'éviter de multiplier les exécutions de tâches redondantes, il est possible d'utiliser `run_once: yes` pour forcer l'exécution d'une tâche une seule fois.

- Pour l'exemple précédent, la création du répertoire template est exécutée deux fois (autant de serveur que de tentatives)

```
TASK [Create template directory] *****
ok: [centos-clone -> centos]
ok: [centos -> centos]
```

- En ajouter `run_once: yes`

```
- name: "Generate html file for each host"
  hosts: all
  gather_facts: yes
  vars:
    host_inventory: "centos"
    inventory_dir:  "/var/www/html/inventory"
  tasks:
    - name: "Create template directory"
      file:
        path: "{{inventory_dir}}"
        owner: "apache"
        group: "apache"
        mode: "0755"
        state: "directory"
        delegate_to: "{{host_inventory}}"
# Execution une seule fois
  run_once: yes
    - name: "html file generation"
      template:
        src: "template-net.html.j2"
        dest: "{{inventory_dir}}/{{inventory_hostname}}.html"
        delegate_to: "{{host_inventory}}"
```

- Rejouer le playbook

```
$ ansible-playbook -i inventaire-jinja jinja4.yml -b -K
```

- La tâche de création du répertoire est exécutée une seule fois.

```
PLAY [Generate html file for each host] *****

TASK [Gathering Facts] *****
ok: [centos-clone]
ok: [centos]

TASK [Create template directory] *****
ok: [centos -> centos]

TASK [html file generation] *****
ok: [centos -> centos]
ok: [centos-clone -> centos]

PLAY RECAP *****
centos                : ok=3    changed=0    unreachable=0    failed=0
centos-clone          : ok=2    changed=0    unreachable=0    failed=0
```



# Installation d'un serveur Apache et configuration avec un fichier template

- Installation de `Apache` en reprenant les exemples précédents
- Configuration d'un alias avec un fichier `template`
- Démarre `Apache`

- Création du fichier template `inventory.conf.j2` , qui reprend les configurations à pousser sur le serveur.

```
# inventory.conf.j2
Alias /inventaire /var/www/html/inventory

# Donne des droits d'accès à tout le monde
<Directory /var/www/html/inventory/>
    Order Allow,Deny
    Allow from All
</Directory>
```

- L'exécution est limitée au serveur centos ( `hosts: centos` ) et la copie de la configuration est effectuée via le module `template`

```
# apache-inventaire.conf.yml

- name: "Apache installation"
  hosts: centos
  tasks:
    - name: "Apache package installation"
      yum:
        name: "httpd"
        state: "present"
    - name: "Apache configuration"
      template:
        src: "inventory.conf.j2"
        dest: "/etc/httpd/conf.d/inventory.conf"
        owner: "apache"
        group: "apache"
    - name: "Start apache service"
      service:
        name: "httpd"
        state: "started"
        enabled: yes
```

- Le playbook installe le serveur Apache sur le serveur `centos` et le configure un alias `inventory` sur le serveur `centos`

```
$ ansible-playbook -i inventaire-jinja apache-inventaire.conf.yml -K -b
```

- Le contenu du répertoire `inventory` est bien visible sur le serveur



	<a href="#">Name</a>	<a href="#">Last modified</a>	<a href="#">Size</a>	<a href="#">Description</a>
	<a href="#">Parent Directory</a>		-	
	<a href="#">centos-clone.html</a>	2018-06-25 12:22	247	
	<a href="#">centos.html</a>	2018-06-25 12:22	235	

# Les tags

Les `tags` vont permettre de découper l'exécution d'un `playbook` en plusieurs section, chaque section sera repérée par des `Tags` .

- Ajout d'un champs `tags` dans le `playbook`

```
- name: "Apache installation"
hosts: centos
tasks:
  - name: "Apache package installation"
    yum:
      name: "httpd"
      state: "present"
  - name: "Apache configuration"
    template:
      src: "inventory.conf.j2"
      dest: "/etc/httpd/conf.d/inventory.conf"
      owner: "apache"
      group: "apache"
  - name: "Start apache service"
# Ajoute un tag ici pour la tâche restart
  tags: restart
  service:
    name: "httpd"
    state: "started"
    enabled: yes
```

- Pour lister tous les tags d'un playbook

```
$ ansible-playbook apache-inventaire.conf.yml --list-tags
```

```
playbook: apache-inventaire.conf.yml
```

```
play #1 (centos): Apache installation TAGS: []  
TASK TAGS: [restart]
```

- Pour inclure ou exclure un tags de l'exécution du playbook
  - `--tags LIST_TAGS` (la liste des tags est séparée par une ',')
  - `--skip LIST_TAGS`

```
$ ansible-playbook -i inventaire-jinja apache-inventaire.conf.yml \  
-K -b --tags restart
```

# Exécution conditionnée

Le principe est d'exécuter une tâche si une condition est vérifiée

- On va utiliser le module `register` et `debug` pour respectivement stocker la sortie de la tâche et l'afficher

```
- name: "Apache installation"
  hosts: centos
  tasks:
    - name: "Apache configuration"
      template:
        src: "inventory.conf.j2"
        dest: "/etc/httpd/conf.d/inventory.conf"
        owner: "apache"
        group: "apache"
# register execution result for template creation
  register: apache_conf
# instruction permettant de scruter le contenu de la variable
  - debug: var=apache_conf
```



- Lancement du `playbook`

```
ansible-playbook -i inventaire-jinja apache-when.yml -K -b
```

- Le fichier inventaire est déjà présent, il n'y a donc pas de changement d'etat.

```
[...]
TASK [debug] *****
ok: [centos] => {
  "apache_conf": {
    "changed": false,
    "checksum": "97d154e3c0263daa75cc2f99406cebddd356cf7d6",
    "dest": "/etc/httpd/conf.d/inventory.conf",
    "diff": {
      "after": {
        "path": "/etc/httpd/conf.d/inventory.conf"
      },
      "before": {
        "path": "/etc/httpd/conf.d/inventory.conf"
      }
    }
  }
}
[...]
```

- La condition `when` est ajoutée sur la variable `apache_conf.changed`, le redémarrage du service se fera uniquement si la configuration de Apache a changé.

```
- name: "Restart apache service"
  service:
    name: "httpd"
    state: "restarted"
# restart if necessary
  when: apache_conf.changed
- name: "Start apache service"
  service:
    name: "httpd"
    state: "started"
    enabled: yes
```

- Le fichier dans son ensemble

```
- name: "Apache installation"
  hosts: centos
  tasks:
    - name: "Apache configuration"
      template:
        src: "inventory.conf.j2"
        dest: "/etc/httpd/conf.d/inventory.conf"
        owner: "apache"
        group: "apache"
    # register execution result for template creation
      register: apache_conf
    # Instruction permettant de scruter le contenu de la variable
    - debug: var=apache_conf

    - name: "Restart apache service"
      service:
        name: "httpd"
        state: "restarted"
    # restart if necessary
      when: apache_conf.changed

    - name: "Start apache service"
      service:
        name: "httpd"
        state: "started"
        enabled: yes
```

- Relance le `playbook`

```
$ ansible-playbook -i inventaire-jinja apache-when.yml -K -b
```

- Pas de changement constaté, donc le redémarrage du service n'est pas effectué

```
TASK [Restart apache service] *****
skipping: [centos]

TASK [Start apache service] *****
ok: [centos]

PLAY RECAP *****
centos                : ok=4    changed=0    unreachable=0    failed=0
```

# Action Conditionnée au contexte des serveurs

Parfois, en fonction du type de système ou de version présents sur le serveur, les actions à effectuer peuvent être différentes.

- Un exemple de configuration qui condition l'action en fonction de la version du système

```
- hosts: all
  tasks:
    - name: Upgrade packages Wheezy
      apt: upgrade=safe
      when: ansible_distribution_major_version == "7"

    - name: Upgrade packages Jessie
      shell: "apt-get update && apt-get upgrade -y"
      when: ansible_distribution_major_version == "8"
```

# Les handlers

Ce sont des tâches qui sont appelées uniquement en cas de besoin

- En reprenant l'exemple précédent mais en simplifiant le fichier de configuration

```
- name: "Apache installation"
  hosts: centos
# Ajout de la section handlers
  handlers:
    - name: "Restart apache service"
      service:
        name: "httpd"
        state: "restarted"
  tasks:
    - name: "Apache package installation"
      yum:
        name: "httpd"
        state: "present"
    - name: "Apache configuration"
      template:
        src: "inventory.conf.j2"
        dest: "/etc/httpd/conf.d/inventory.conf"
        owner: "apache"
        group: "apache"
# Notify Apache restart handler
    notify: [ "Restart apache service" ]
    - name: "Start apache service"
      service:
        name: "httpd"
        state: "started"
        enabled: yes
```

- Le `handlers` sera exécuté uniquement si il y a un changement.
- Première exécution, pas de changement le `handlers` n'est pas exécuté

```
ansible-playbook -i inventaire-jinja handlers.yml -K -b
```

```
PLAY [Apache installation] *****

TASK [Gathering Facts] *****
ok: [centos]

TASK [Apache package installation] *****
ok: [centos]

TASK [Apache configuration] *****
ok: [centos]

TASK [Start apache service] *****
ok: [centos]

PLAY RECAP *****
centos                : ok=4    changed=0    unreachable=0    failed=0
```



- Deuxième exécution avec changement dans le template

```
TASK [Gathering Facts] *****
ok: [centos]

TASK [Apache package installation] *****
ok: [centos]

TASK [Apache configuration] *****
changed: [centos]

TASK [Start apache service] *****
ok: [centos]

RUNNING HANDLER [Restart apache service] *****
changed: [centos]

PLAY RECAP *****
centos                : ok=5    changed=2    unreachable=0    failed=0
```