



Playbook avancé



Dans ce module :

- Les variables
- Les states
- Utilisation des handlers
- Les vaults

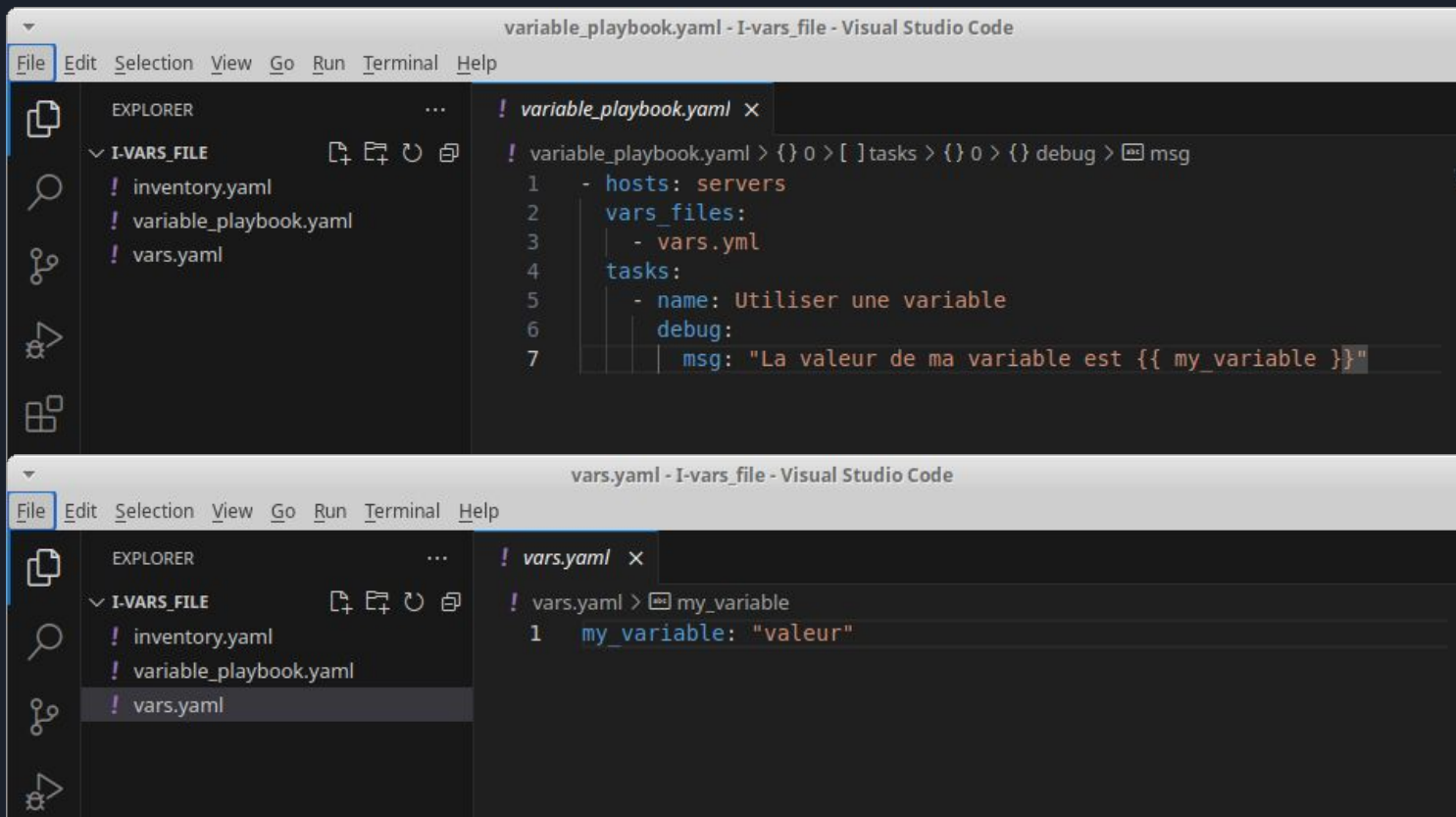
Utilisation de variables



Les variables sont des éléments essentiels dans la création de playbooks Ansible, car elles permettent de **stocker des valeurs** réutilisables.

Les variables peuvent être **définies** au niveau du **playbook**, des **inventaires**, ou dans des **fichiers dédiés**.

Utilisation de variable simple



The image displays two Visual Studio Code windows illustrating the use of a simple variable in Ansible.

Top Window: variable_playbook.yaml - I-vars_file - Visual Studio Code

The Explorer sidebar shows the file structure:

- I-VARS_FILE
 - ! inventory.yaml
 - ! variable_playbook.yaml
 - ! vars.yaml

The main editor shows the content of `variable_playbook.yaml`:

```
! variable_playbook.yaml x
! variable_playbook.yaml > {} 0 > [ ]tasks > {} 0 > {} debug > msg
1 - hosts: servers
2   vars_files:
3     - vars.yml
4   tasks:
5     - name: Utiliser une variable
6       debug:
7         msg: "La valeur de ma variable est {{ my_variable }}"
```

Bottom Window: vars.yaml - I-vars_file - Visual Studio Code

The Explorer sidebar shows the file structure:

- I-VARS_FILE
 - ! inventory.yaml
 - ! variable_playbook.yaml
 - ! vars.yaml

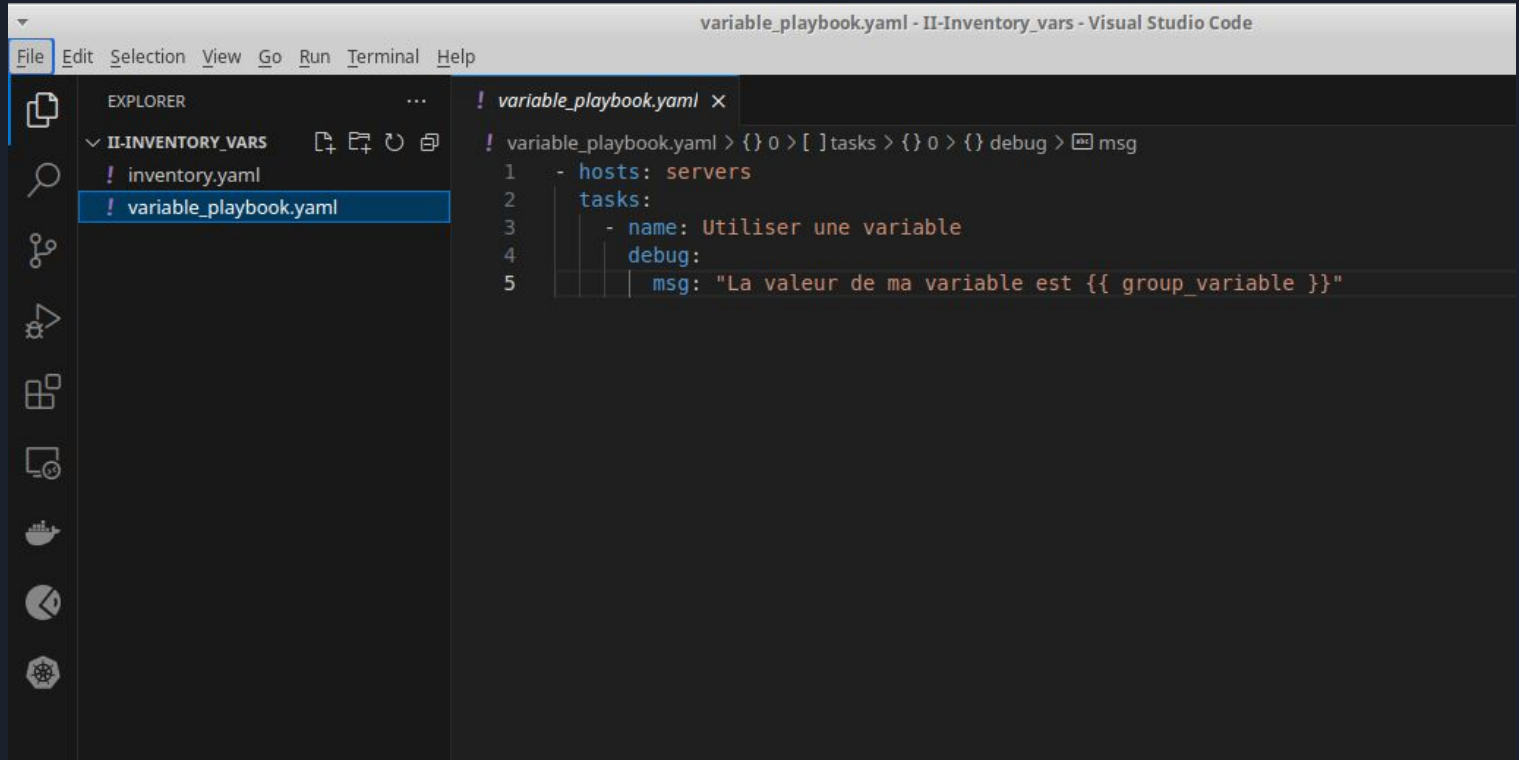
The main editor shows the content of `vars.yaml`:

```
! vars.yaml x
! vars.yaml > my_variable
1 my_variable: "valeur"
```

Variable globale et par groupe

```
! inventory.yml x
! inventory.yml > {} servers > {} children > {} group2 > {} vars
Ansible Inventory - Ansible inventory files (inventory.json)
1 servers:
2   hosts:
3     server1:
4       ansible_host: developer@10.10.10.102
5     server2:
6       ansible_host: developer@10.10.10.226
7   vars:
8     common_variable: "valeur commune pour tous les serveurs"
9   children:
10     group1:
11       hosts:
12         server1:
13         vars:
14           group_variable: "valeur spécifique au groupe 1"
15     group2:
16       hosts:
17         server2:
18         vars:
19           group_variable: "valeur spécifique au groupe 2"
20
```

Utilisation





A vous !

Why ?

Cette étape permet de rendre votre playbook réutilisable

Objectif ?

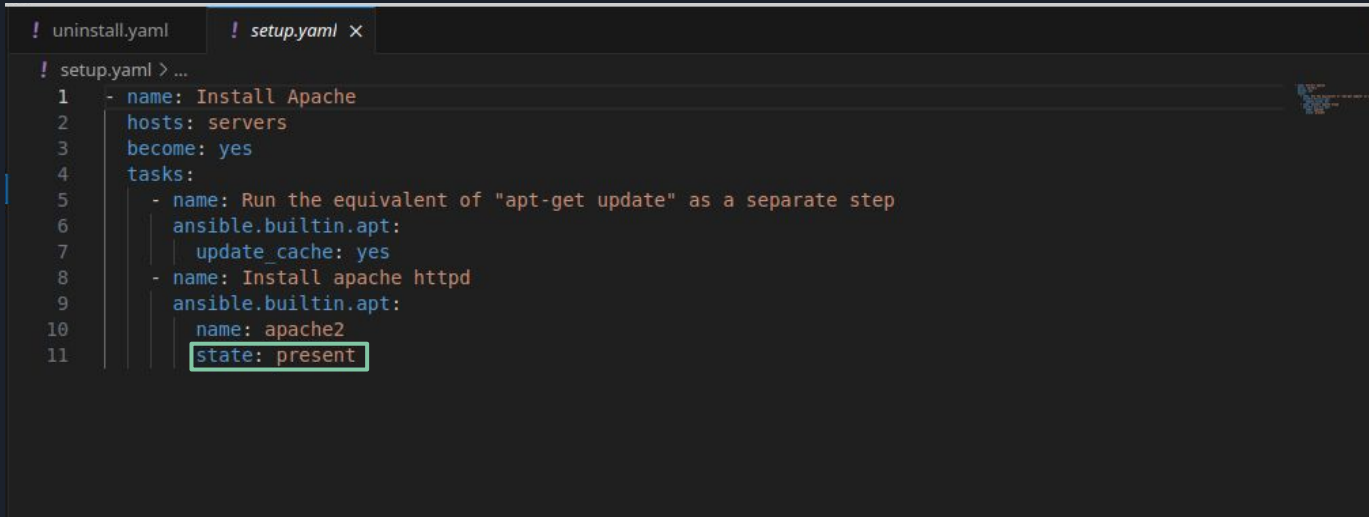
Ajouter une variable au module présenté auparavant

Outils ?

- vars.yaml

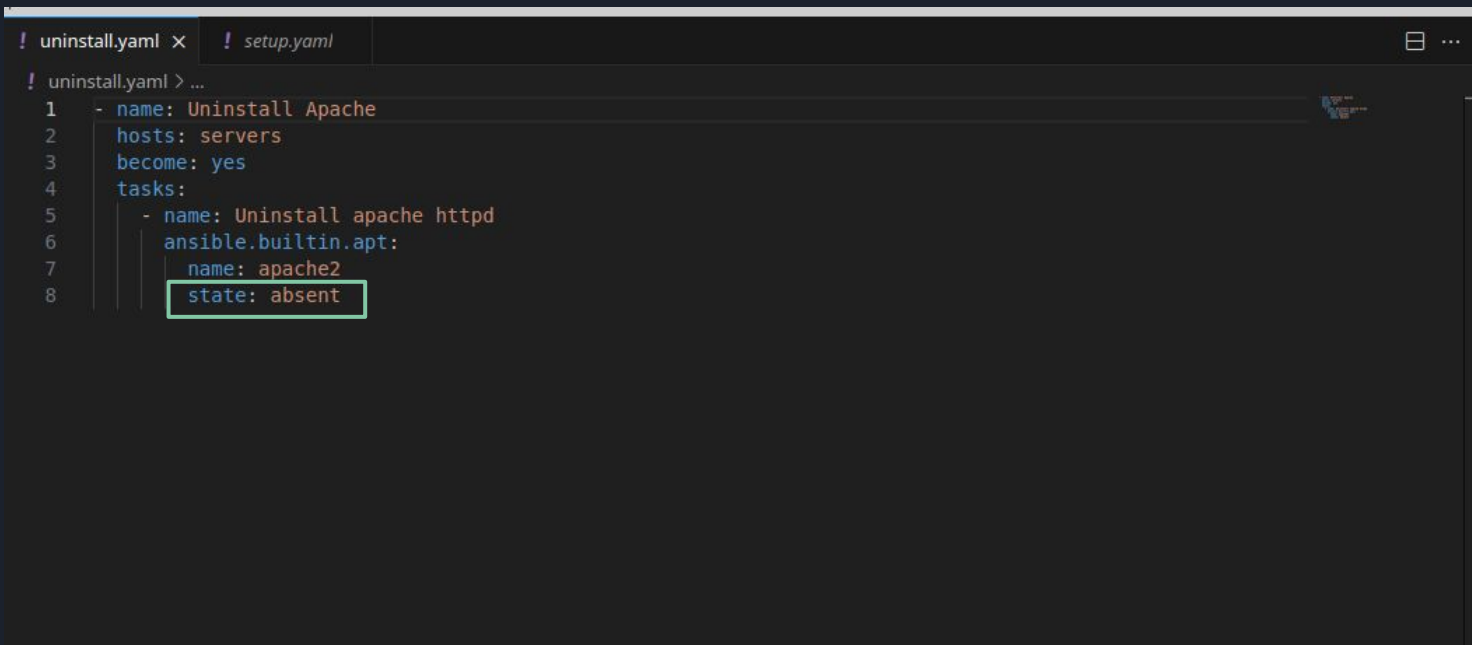
Apt & state

Etat désiré : Doit être présent



```
! uninstall.yaml  ! setup.yaml x
! setup.yaml > ...
1  - name: Install Apache
2    hosts: servers
3    become: yes
4    tasks:
5      - name: Run the equivalent of "apt-get update" as a separate step
6        ansible.builtin.apt:
7          update_cache: yes
8      - name: Install apache httpd
9        ansible.builtin.apt:
10         name: apache2
11         state: present
```

Etat désiré : Doit être absent



```
! uninstall.yaml × ! setup.yaml
! uninstall.yaml > ...
1 - name: Uninstall Apache
2   hosts: servers
3   become: yes
4   tasks:
5     - name: Uninstall apache httpd
6       ansible.builtin.apt:
7         name: apache2
8         state: absent
```

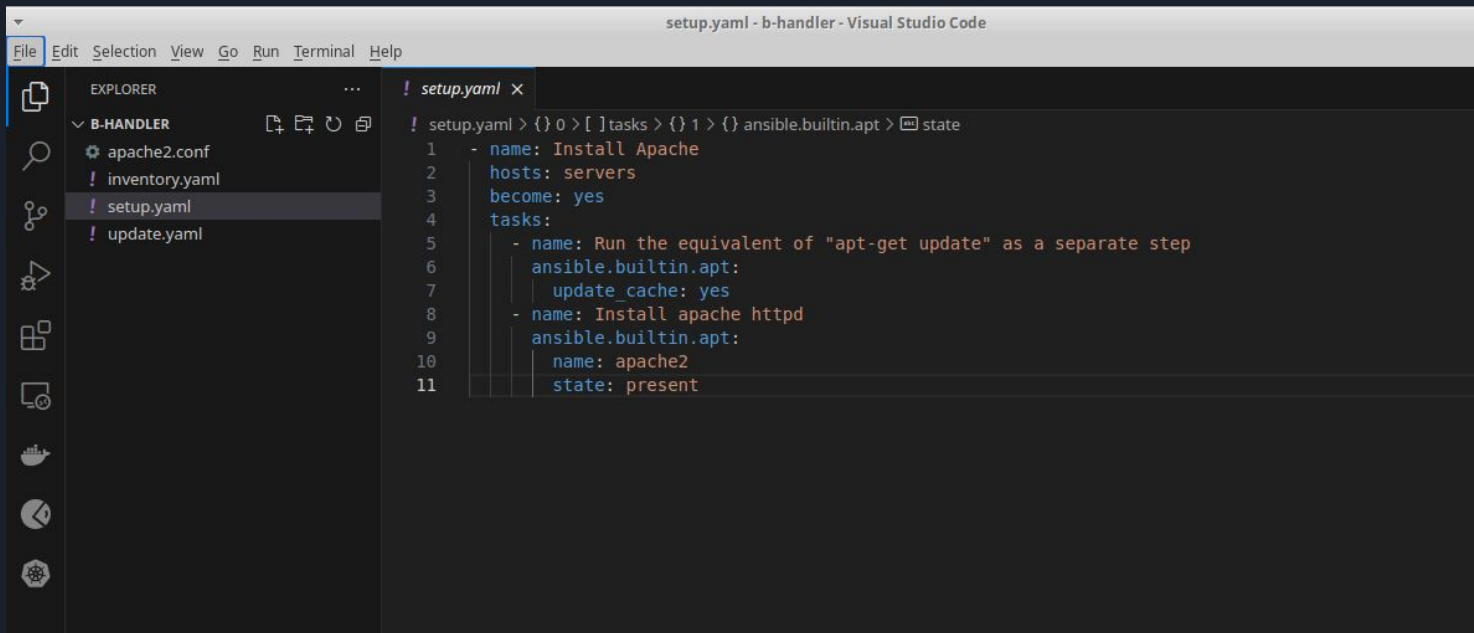
Utilisation des handlers

Les handlers sont des tâches déclenchées uniquement si une autre tâche annonce des changements dans l'état des hôtes cibles. Les handlers permettent d'exécuter des actions de manière conditionnelle pour gérer les événements qui nécessitent une réaction spécifique.

Les handlers sont souvent utilisés pour redémarrer des services, effectuer des actions de nettoyage, ou envoyer des notifications en cas de modifications dans l'infrastructure.

Pour déclencher un handler, il est nécessaire de notifier explicitement son exécution dans les tâches précédentes du playbook en utilisant le module `'notify'`. Les handlers ne sont activés que si des changements ont été détectés lors de l'exécution des tâches antérieures.

Exemple handler partie 1 - Install



The screenshot shows the Visual Studio Code interface with a file explorer on the left and a code editor on the right. The file explorer shows a project named 'B-HANDLER' with files 'apache2.conf', 'inventory.yaml', 'setup.yaml', and 'update.yaml'. The 'setup.yaml' file is selected. The code editor shows the content of 'setup.yaml', which is an Ansible playbook. The playbook has a play for 'servers' with a task 'Install Apache' that includes a handler 'Run the equivalent of "apt-get update" as a separate step'. The handler is defined in the 'handlers' section of the playbook.

```
! setup.yaml x
! setup.yaml > {} 0 > [ ] tasks > {} 1 > {} ansible.builtin.apt > state
1 - name: Install Apache
2   hosts: servers
3   become: yes
4   tasks:
5     - name: Run the equivalent of "apt-get update" as a separate step
6       ansible.builtin.apt:
7         update_cache: yes
8     - name: Install apache httpd
9       ansible.builtin.apt:
10        name: apache2
11        state: present
```


Exemple handler partie 2 - vérification

! update.yaml ×

! update.yaml > {} 0 > [] handlers > {} 0 > {} service > state

```
1 - name: Update Apache
2   hosts: servers
3   become: yes
4   tasks:
5     - name: Ensure Apache config is updated
6       template:
7         src: apache2.conf
8         dest: /etc/apache2/apache2.conf
9       notify: restart apache # Nom du handler qui sera appelé
10  handlers:
11    - name: restart apache
12      service:
13        name: apache2
14        state: restarted
```

Vault

Crypter un fichier


```
ansible-vault encrypt secret.yaml  
ansible-vault edit secret.yaml  
ansible-vault decrypt secret.yaml
```

! secret.yaml X

d-vault > ! secret.yaml

```
1  $ANSIBLE_VAULT;1.1;AES256
2  39303565393734323537316139653733393733363436653662313633363436313464613537313565
3  6661646137653665373535366166646264336666376161340a326262633933333764336236313439
4  31386564346261343536353436343866383234306264643034616238633565663831633865653166
5  3830373761666630370a313466383365666432646232336364336431396438323132636264303466
6  38666632383536616561383664633038616665313066383932613961663030303834
7
```

! showSecret.yaml X

d-vault > ! showSecret.yaml > {} 0 > []tasks > {} 0 > {} debug >  msg

```
1  - hosts: servers
2    vars_files:
3      - secret.yaml
4    tasks:
5      - name: Utiliser un fichier encrypté
6        debug:
7          msg: "Your password is {{password}}"
```

ansible-playbook showSecret.yaml -i inventory.yaml --ask-vault-pass

**Crypter une chaîne de
caractère**

```
! inventory.yml 1 x
d-vault > encryptString > ! inventory.yml > {} servers > {} hosts > {} server2 > # ansible_port
Ansible Inventory - Ansible inventory files (inventory.json)
1  servers:
2    vars:
3      user: !vault |
4          $ANSIBLE_VAULT;1.1;AES256
5          64383435376332633531373164616563616261653639383039386637313661323465336265383837
6          3665356338386432643563643966663039666636633462330a613064323333343133376633633639
7          65363636636432363033626565316662386230346536626665396338303932636361326330613961
8          3763363163666132330a613938306337643532366433386130383465303365343931633633383337
9          6662
10   hosts:
11     server1:
12       ansible_user: "{{user}}"
13       ansible_host: 10.10.10.22
14       ansible_port: 22
15     server2:
16       ansible_user: "{{user}}"
17       ansible_host: 10.10.10.190
18       ansible_port: 22

! main.yml x
d-vault > encryptString > ! main.yml > {} 0 > [ ] tasks > {} 0 > [ ] ping
1  - hosts: servers
2    tasks:
3      - name: Send a ping to host with encrypted string
4        ping:
```

ansible-vault encrypt_string developer --ask-vault-pass



A vous !

Why ?

Réutiliser toutes les notions vu

Objectif ?

Configurer un server avec apache2 avec des variables cryptés dans l'inventaire et des handler lors de la modification du fichier de configuration apache grâce au module template et service

Déposer son projet dans Github

Outils ?

- Vault, handler, service et template