6.

ORGANIZACIÓN DE PLAYBOOKS: ROLES E INCLUDES

INTRODUCCIÓN

Aunque es posible escribir un *playbook* en un archivo muy largo, **lo habitual es tratar de reutilizar partes del proyecto Ansible y tener todo organizado**.

En su nivel más básico, incluir archivos con tareas nos permite dividir grandes archivos de configuración en otros más pequeños.

INTRODUCCIÓN

Los roles en Ansible se utilizan con la idea de que incluir archivos con tareas y combinarlos pueden formar un proyecto limpio y con partes reutilizables.

Comenzaremos explicando lo que son los *includes* antes de aprender a controlar los roles.

6.1
INCLUDES

INCLUDES

Ya hemos visto en capítulos anteriores cómo instalar un servidor Apache y asegurarnos de que se esté ejecutando en varios servidores.

¿Y si quisiéramos reutilizar esas instrucciones?

INCLUDES DE TAREAS

Este es el *playbook* original:

```
---
- hosts: all
  remote_user: root
  tasks:
  - name: Ensure Apache is at the latest version
    apt: name=apache2 state=latest
  - name: ensure apache is running
    service: name=apache2 state=started enabled=yes
```

INCLUDES DE TAREAS

Podríamos dejarlo así:

```
---
- hosts: all
  remote_user: root
  tasks:
  - include: apache-playbook.yml
```

INCLUDES DE TAREAS

El fichero **apache-playbook.yml** debería quedar así:

```
---
- name: Ensure Apache is at the latest version
  apt: name=apache2 state=latest
- name: ensure apache is running
  service: name=apache2 state=started enabled=yes
```

Y podríamos **reutilizarlo desde cualquier parte del proyecto**.

INCLUDES DE HANDLERS

Los *handlers* pueden incluirse de igual manera que las tareas, pero en la sección correspondiente.

En capítulos anteriores teníamos este ejemplo:

```
handlers:
    - name: restart php
      service: name=php state=restarted
    - name: restart apache
      service: name=apache state=restarted
```

INCLUDES DE HANDLERS

Para poder reutilizar los handlers, podríamos sacarlos a un fichero externo, como **included-handlers.yml**, cuyo contenido sería:

```
---
- name: restart php
  service: name=php state=restarted
- name: restart apache
  service: name=apache state=restarted
```

INCLUDES DE HANDLERS

Y en la sección de *handlers* tendríamos que hacer el *include*:

```
handlers:
    - include: included-handlers.yml
```

INCLUDES DE PLAYBOOKS

Un *playbook* puede ser incluido en otro *playbook* utilizando la misma sintaxis de include.

Por ejemplo, si tenemos un *playbook* que configura los servidores web y otro que configura los servidores de base de datos, podríamos hacer un include de ambos *playbooks* en uno solo:

```
- include: webserver-playbook.yml
- include: database-playbook.yml
```

INCLUDES DE PLAYBOOKS

Un *playbook* puede ser incluido en otro *playbook* utilizando la misma sintaxis de include.

Por ejemplo, si tenemos un *playbook* que configura los servidores web y otro que configura los servidores de base de datos, podríamos hacer un include de ambos *playbooks* en uno solo:

```
- include: webserver-playbook.yml
- include: database-playbook.yml
```

INCLUDES DINÁMICOS

A partir de Ansible 2.0 los includes pueden ser dinámicos haciendo uso de bucles:

```
---
- hosts: all
  remote_user: root
  tasks:
    - include: foo.yml param={{ item }}
      with_items:
          - 1
          - 2
          - 3
```

INCLUDES DINÁMICOS

foo.yml:

```
---
- name: "Echoing params"
  command: echo "{{ param }}"
```

EJEMPLO COMPLETO CON INCLUDES

```
---
- hosts: all
  pre_tasks:
    - name: Update apt cache if needed.
      apt: update_cache=yes cache_valid_time=3600
      handlers:
        - include: handlers/handlers.yml
  tasks:
    - include: tasks/common.yml
    - include: tasks/apache.yml
    - include: tasks/php.yml
    - include: tasks/mysql.yml
```

EJEMPLO COMPLETO CON INCLUDES

Separar las tareas en diferentes archivos quiere decir que tendremos más archivos que gestionar, pero **hará que nuestro *playbook* sea más mantenible**.

Es mucho más fácil **mantener pequeños grupos de tareas relacionadas entre sí** que un *playbook* excesivamente largo.

El uso de tags también facilita la organización en un proyecto.