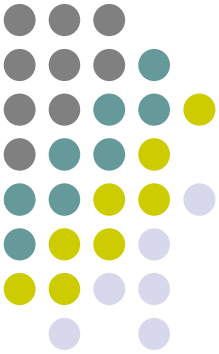


ADMINISTRACIÓN DE INFRAESTRUCTURAS Y SISTEMAS INFORMÁTICOS (AISI)

Grado en Ingeniería Informática

Roberto R. Expósito (roberto.rey.exposito@udc.es)

ANSIBLE

Contenidos

- Introducción
- Conceptos básicos
- Comandos *ad-hoc*
- *Playbooks*
- Roles
- *Collections*
- Galaxy
- Ansible + Vagrant
- Ansible + Packer
- Ansible + Docker

Contenidos

- **Introducción**
- Conceptos básicos
- Comandos *ad-hoc*
- *Playbooks*
- Roles
- *Collections*
- Galaxy
- Ansible + Vagrant
- Ansible + Packer
- Ansible + Docker

¿Qué es Ansible?

- **Ansible** es una herramienta IaC de código abierto desarrollada y soportada comercialmente por Red Hat, Inc. que permite gestionar configuraciones, aprovisionar y administrar recursos de infraestructura, desplegar aplicaciones y orquestar muchas otras tareas IT de una forma sencilla, flexible y ágil

https://github.com/ansible/ansible

Características principales

- Su popularidad y facilidad de uso radica en dos pilares:
  1. Ansible se instala en una **única** máquina (**controlador**) que actúa como punto central desde el cual se gestionan otros servidores de forma **remota** a través de SSH (por defecto)
     - El controlador requiere tener instalado Python 3
       - Ansible soporta Linux/macOS/BSD y Windows con WSL
     - Los servidores gestionados con Ansible solo requieren disponer de Python >=2.7/3 o PowerShell sin necesidad de instalar/ejecutar ningún otro agente *software* adicional ni disponer de una base de datos centralizada
       - Por eso se dice que su arquitectura es ***agentless*** (sin agentes)
       - No hay ningún consumo de recursos en los servidores gestionados con Ansible cuando no se está ejecutando ninguna acción sobre ellos (no existen demonios en ejecución en segundo plano)

Características principales

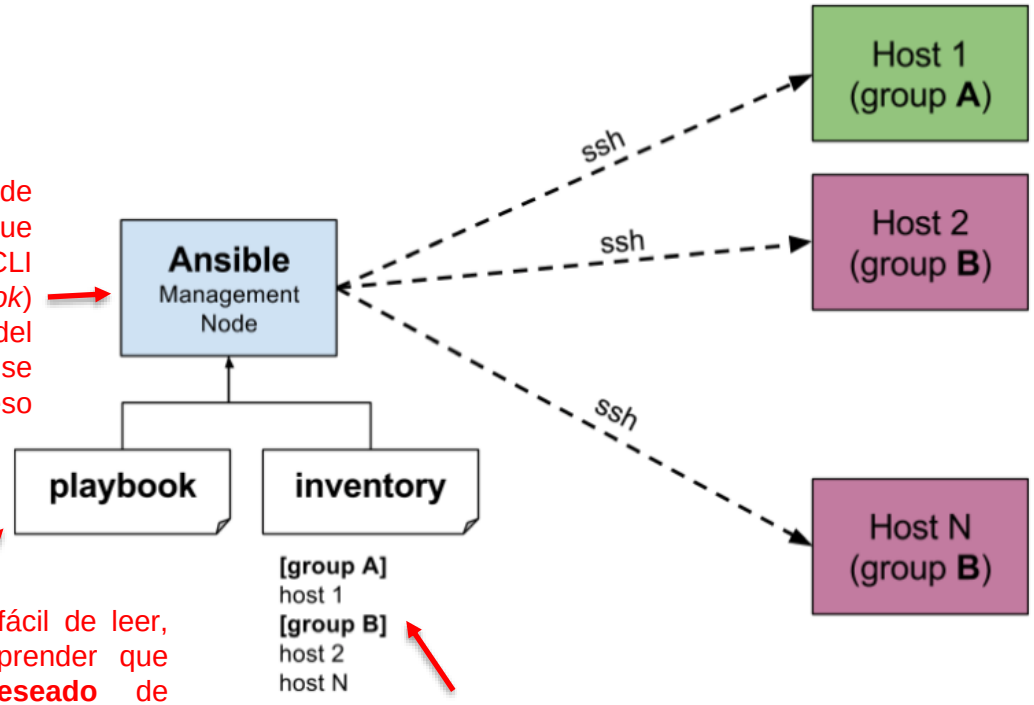
- Su popularidad y facilidad de uso radica en dos pilares:
  1. Ansible se instala en una **única** máquina (**controlador**) que actúa como punto central desde el cual se gestionan otros servidores de forma **remota** a través de SSH (por defecto)
     - El controlador requiere tener instalado Python 3
       - Ansible soporta Linux/macOS/BSD y Windows con WSL
     - Los servidores gestionados con Ansible solo requieren disponer de Python >=2.7/3 o PowerShell sin necesidad de instalar/ejecutar ningún otro agente *software* adicional ni disponer de una base de datos centralizada
       - Por eso se dice que su arquitectura es ***agentless*** (sin agentes)
       - No hay ningún consumo de recursos en los servidores gestionados con Ansible cuando no se está ejecutando ninguna acción sobre ellos (no existen demonios en ejecución en segundo plano)
  2. Su **suave curva de aprendizaje** ya que se basa en un lenguaje simple y legible (YAML) para describir las acciones y las configuraciones a realizar en los servidores gestionados con Ansible
     - Ansible AWX proporciona una interfaz Web y una API REST, entre otras cosas, sobre el CLI de Ansible

Contenidos

- Introducción
- **Conceptos básicos**
- Comandos *ad-hoc*
- *Playbooks*
- Roles
- *Collections*
- Galaxy
- Ansible + Vagrant
- Ansible + Packer
- Ansible + Docker

Arquitectura *masterless/agentless*

- Ansible tiene una arquitectura "solo cliente"
  - Es una de las principales ventajas frente otras herramientas IaC para gestión de la configuración como Puppet, Chef, SaltStack, CFEngine…



Controlador: La máquina donde se instala Ansible y desde la que ejecuta los comandos de CLI (p.e. *ansible*, *ansible-playbook*) para los gestionar servidores del fichero de inventario. No se requiere ejecutar ningún proceso o demonio en *background*

En los nodos a gestionar no es necesario instalar ningún *software* específico (agente) de Ansible

Playbook: Fichero YAML fácil de leer, escribir, compartir y comprender que contiene el **estado deseado** de configuración de los servidores

Inventario: Fichero que contiene la lista de FQDNs/IPs de los servidores o nodos (*hosts*) a gestionar. Pueden organizarse en **[grupos]**

Conceptos: Tarea, *playbook* y módulo

- **Tarea**
  - Unidad básica de "acción" en Ansible que se ejecuta en un nodo gestionado
  - Es posible ejecutar una tarea una única vez con el comando ***ansible*** (**comando *ad-hoc***)
- ***Playbook***
  - Fichero YAML que contiene una lista ordenada de "jugadas"
  - Una "jugada" asigna nodos del inventario a las tareas que se ejecutarán en ellos
    - Puede contener **variables, roles** y listas ordenadas de tareas
    - Se puede ejecutar repetidamente
  - El comando ***ansible-playbook*** permite ejecutar un *playbook*
- **Módulo**
  - Código Python/binarios que Ansible copia y ejecuta en cada nodo gestionado (cuando sea necesario) para realizar la "acción" definida en cada tarea
  - Cada módulo proporciona una funcionalidad particular (p.e. gestión de usuarios, copia de ficheros, instalación de *software*, gestión de servicios)
    - https://docs.ansible.com/ansible/latest/collections/index_module.html
  - Se puede invocar un único módulo con una tarea o invocar varios en un *playbook*
  - Los módulos se agrupan en ***collections*** (colecciones)

Rol y *collection*

- **Rol**
  - Permite cargar automáticamente variables, ficheros, tareas, *handlers* y otros artefactos de Ansible relacionados entre sí en función de una estructura de directorios y ficheros definida
  - Puede reutilizarse fácilmente en múltiples *playbooks* y compartirse con otros usuarios mediante **Ansible Galaxy**
    - Galaxy es un repositorio público de roles y *collections* desarrollados por la comunidad
- ***Collection***
  - Formato de distribución para todo tipo de contenido Ansible (*playbooks*, roles, módulos, *plugins*) en un único paquete
  - Se define por su ***Fully Qualified Collection Name* (FQCN)**: *namespace.collection*
    - El FQCN se utiliza referenciar contenido dentro la *collection*
  - El *core* de Ansible incluye la *collection* integrada (***ansible.builtin***) con más de 150 módulos que proporcionan funcionalidades de todo tipo
    - https://docs.ansible.com/ansible/latest/collections/ansible/builtin/index.html
    - Ejemplo: módulo *package* de la *collection* integrada: *ansible.builtin.package*
    - **Si no se indica el FQCN, por defecto se asume la *collection* integrada**

¿Cómo funciona Ansible?

- Cuando se ejecuta un **comando *ad-hoc*** (mediante *ansible*) o un ***playbook*** (mediante *ansible-playbook*), ocurre lo siguiente:

¿Cómo funciona Ansible?

- Cuando se ejecuta un **comando *ad-hoc*** (mediante *ansible*) o un ***playbook*** (mediante *ansible-playbook*), ocurre lo siguiente:
  1. Se seleccionan los nodos (o *hosts*) del inventario sobre los que se actuará
     - Inventario por defecto: */etc/ansible/hosts*
       - Se puede indicar una ruta diferente con el parámetro *--inventory|-i*

¿Cómo funciona Ansible?

- Cuando se ejecuta un **comando *ad-hoc*** (mediante *ansible*) o un ***playbook*** (mediante *ansible-playbook*), ocurre lo siguiente:
  1. Se seleccionan los nodos (o *hosts*) del inventario sobre los que se actuará
     - Inventario por defecto: */etc/ansible/hosts*
       - Se puede indicar una ruta diferente con el parámetro *--inventory|-i*
  2. Se conecta a los *hosts* seleccionados mediante SSH (por defecto)
     - La conexión por SSH se realiza con el usuario actual del controlador
       - Se puede indicar otro usuario con el parámetro *--user|-u*
     - Por defecto, se asume autenticación SSH *passwordless*
       - Si es necesario, el parámetro *--ask-pass|-k* permite introducir un *password*

¿Cómo funciona Ansible?

- Cuando se ejecuta un **comando *ad-hoc*** (mediante *ansible*) o un ***playbook*** (mediante *ansible-playbook*), ocurre lo siguiente:
  1. Se seleccionan los nodos (o *hosts*) del inventario sobre los que se actuará
     - Inventario por defecto: */etc/ansible/hosts*
       - Se puede indicar una ruta diferente con el parámetro *--inventory|-i*
  2. Se conecta a los *hosts* seleccionados mediante SSH (por defecto)
     - La conexión por SSH se realiza con el usuario actual del controlador
       - Se puede indicar otro usuario con el parámetro *--user|-u*
     - Por defecto, se asume autenticación SSH *passwordless*
       - Si es necesario, el parámetro *--ask-pass|-k* permite introducir un *password*
  3. Se copian los módulos necesarios a los *hosts* seleccionados mediante SFTP (por defecto)
     - Es posible configurar Ansible para usar SCP en vez de SFTP

¿Cómo funciona Ansible?

- Cuando se ejecuta un **comando *ad-hoc*** (mediante *ansible*) o un ***playbook*** (mediante *ansible-playbook*), ocurre lo siguiente:
  1. Se seleccionan los nodos (o *hosts*) del inventario sobre los que se actuará
     - Inventario por defecto: */etc/ansible/hosts*
       - Se puede indicar una ruta diferente con el parámetro *--inventory|-i*
  2. Se conecta a los *hosts* seleccionados mediante SSH (por defecto)
     - La conexión por SSH se realiza con el usuario actual del controlador
       - Se puede indicar otro usuario con el parámetro *--user|-u*
     - Por defecto, se asume autenticación SSH *passwordless*
       - Si es necesario, el parámetro *--ask-pass|-k* permite introducir un *password*
  3. Se copian los módulos necesarios a los *hosts* seleccionados mediante SFTP (por defecto)
     - Es posible configurar Ansible para usar SCP en vez de SFTP
  4. Se inicia la ejecución de los módulos Ansible correspondientes en los *hosts* seleccionados

Formato del inventario

- Ansible soporta, entre otros, los formatos INI y YAML para definir el inventario
- Siempre existen dos grupos por defecto (implícitos)
  - **all**: contiene todos los *hosts*
  - **ungrouped**: contiene todos los *hosts* que no pertenecen a ningún otro grupo que no sea el grupo all
- Se pueden usar rangos numéricos y alfabéticos para definir los *hosts*
  - www[01:50].example.com (con stride: www[01:50:2].example.com)
  - db-[a:f].example.com

INI

```
mail.example.com

[webservers]
foo.example.com
bar.example.com

[dbservers]
one.example.com
two.example.com
three.example.com
```

YAML

```
all:
  hosts:
    mail.example.com:
  children:
    webservers:
      hosts:
        foo.example.com:
        bar.example.com:
    dbservers:
      hosts:
        one.example.com:
        two.example.com:
        three.example.com:
```

Inventario: variables y grupos de grupos

- Es posible definir variables de *host* (izquierda) y variables de grupo (derecha)
  - Esas variables pueden usarse luego en los *playbooks*

```
[atlanta]
host1 http_port=80 maxRequestsPerChild=808
host2 http_port=303 maxRequestsPerChild=909
```

```
[atlanta]
host1
host2

[atlanta:vars]
ntp_server=ntp.atlanta.example.com
proxy=proxy.atlanta.example.com
```

```
atlanta:
  host1:
    http_port: 80
    maxRequestsPerChild: 808
  host2:
    http_port: 303
    maxRequestsPerChild: 909
```

```
atlanta:
  hosts:
    host1:
    host2:
  vars:
    ntp_server: ntp.atlanta.example.com
    proxy: proxy.atlanta.example.com
```

- Se pueden crear **metagrupos (grupos de grupos)**
  - INI
    - Usando *[group:children]*
  - YAML
    - Usando la entrada *children:*

```
[atlanta]
host1
host2

[raleigh]
host2
host3

[southeast:children]
atlanta
raleigh
```

Contenidos

- Introducción
- Instalación
- Conceptos básicos
- **Comandos *ad-hoc***
- *Playbooks*
- Roles
- *Collections*
- Galaxy
- Ansible + Vagrant
- Ansible + Packer
- Ansible + Docker

Comandos *ad-hoc*

- Los comandos *ad-hoc* se ejecutan con ***ansible***
  - Permiten ejecutar una única tarea de forma simple y rápida en uno o varios *hosts*, pero no son reusables directamente
- Sintaxis general del comando *ansible*:
  - *ansible [pattern] -m [module] -a "[module options]"*
- El **patrón** selecciona los *hosts* del inventario sobre los cuáles se ejecutará el módulo correspondiente
  - Se puede especificar un *host* en concreto, todos (all), un grupo...
    - https://docs.ansible.com/ansible/latest/inventory_guide/intro_patterns.html
  - Los *hosts* seleccionados se pueden modificar mediante el parámetro *--limit|-l*
  - Se puede obtener la lista de *hosts* seleccionados por el patrón sin ejecutar nada usando el parámetro *--list-hosts*
- Su modelo **declarativo**, como el de los *playbooks*, planifica las acciones a realizar para alcanzar el **estado final deseado**
  - La mayoría de los módulos son **idempotentes:** comprueban el estado actual del *host* y finalizan sin realizar ninguna acción si ya se ha alcanzado el estado deseado

Comandos *ad-hoc*: ejemplos

- Módulo ***command***
  - Ejecución general de comandos (no se procesan a través del *shell*)
    - https://docs.ansible.com/ansible/latest/collections/ansible/builtin/command_module.html
    - Es el **módulo por defecto** si no se especifica el parámetro *-m*
  - El módulo ***shell*** es casi exactamente como ***command*** pero ejecuta el comando a través de un *shell* (*/bin/sh*)
    - https://docs.ansible.com/ansible/latest/collections/ansible/builtin/shell_module.html

```
vagrant@ansible:~$ hostname
ansible
vagrant@ansible:~$ cat /etc/ansible/hosts
[local]
ansible
vagrant@ansible:~$ ansible all -a "echo Hello World"
ansible | CHANGED | rc=0 >>
Hello World
vagrant@ansible:~$ ansible all -m command -a "echo Hello World"
ansible | CHANGED | rc=0 >>
Hello World
vagrant@ansible:~$ ansible all -m ansible.builtin.command -a "echo Hello World"
ansible | CHANGED | rc=0 >>
Hello World
vagrant@ansible:~$ ansible local -a "echo Hello World"
ansible | CHANGED | rc=0 >>
Hello World
vagrant@ansible:~$ ansible ansible -a "echo Hello World"
ansible | CHANGED | rc=0 >>
Hello World
vagrant@ansible:~$
```

En estos ejemplos, el nodo controlador (donde está instalado Ansible) se gestiona a sí mismo (es también un nodo del fichero inventario)

Comandos *ad-hoc*: ejemplos

- Módulo ***ping***
  - Intenta la conexión a un *host*, verifica la existencia de un binario Python utilizable y devuelve *pong* si tiene éxito
    - https://docs.ansible.com/ansible/latest/collections/ansible/builtin/ping_module.html

```
vagrant@ansible:~$ ansible all -m ansible.builtin.ping
ansible | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
vagrant@ansible:~$ ansible local -m ping
ansible | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
vagrant@ansible:~$
```

Por simplicidad, omitiremos el FQCN cuando usemos módulos de la *collection* por defecto (*builtin*)

Comandos *ad-hoc*: ejemplos

- Módulo ***service***
  - Gestión de servicios del sistema, compatible con varios sistemas de inicio: *systemd, BSD init, SysV, Solaris SMF, upstart*
    - https://docs.ansible.com/ansible/latest/collections/ansible/builtin/service_module.html
  - Exiten también módulos para sistemas específicos como systemd
    - https://docs.ansible.com/ansible/latest/collections/ansible/builtin/systemd_module.html
  - Iniciamos el servicio de red (network)

```
vagrant@ansible:~$ ansible local -m service -a "name=network.target state=started"
ansible | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "name": "network.target",
    "state": "started",
    "status": {
        "ActiveEnterTimestamp": "Fri 2024-02-02 08:18:10 UTC",
        "ActiveEnterTimestampMonotonic": "3495874",
        "ActiveExitTimestamp": "n/a",
        "ActiveExitTimestampMonotonic": "0",
        "ActiveState": "active",
```

En este ejemplo, el servicio ya estaba en ejecución, por lo que el módulo no realiza cambios y devuelve SUCCESS (salida en color verde)

Comandos *ad-hoc*: ejemplos

- Módulos ***copy*** y ***template***
  - Copia de ficheros o estructuras de directorios desde la máquina local (controlador) o remota a una ubicación en la máquina remota
    - https://docs.ansible.com/ansible/latest/collections/ansible/builtin/copy_module.html
  - Si se necesita interpolación de variables en los ficheros copiados, se debe usar el módulo ***template***
    - https://docs.ansible.com/ansible/latest/collections/ansible/builtin/template_module.html

```
vagrant@ansible:~$ cat file
Hello World
vagrant@ansible:~$ ansible local -m copy -a "src=./file dest=/tmp/file"
ansible | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": true,
    "checksum": "648a6a6ffffdaa0badb23b8baf90b6168dd16b3a",
    "dest": "/tmp/file",
    "gid": 1000,
    "group": "vagrant",
    "md5sum": "e59ff97941044f85df5297e1c302d260",
    "mode": "0664",
    "owner": "vagrant",
    "size": 12,
    "src": "/home/vagrant/.ansible/tmp/ansible-tmp-1706870354.6268404-4
    "state": "file",
    "uid": 1000
}
vagrant@ansible:~$ cat /tmp/file
Hello World
vagrant@ansible:~$
```

En este ejemplo, el fichero no existía en el destino por lo que se copi. El módulo realiza cambios en el estado del nodo y devuelve CHANGED (salida en color marrón)

Comandos *ad-hoc*: ejemplos

- Módulo ***file***
  - Modificación de atributos de ficheros, directorios o enlaces simbólicos y sus destinos
  - Alternativamente, eliminación de ficheros, enlaces simbólicos o directorios
    - https://docs.ansible.com/ansible/latest/collections/ansible/builtin/file_module.html
  - Creamos un directorio "*folder*" con permisos rw-r--r-- (*mode=644*)

```
vagrant@ansible:~$ ls -l
total 4
-rw-rw-r-- 1 vagrant vagrant 12 Feb  2 10:38 file
vagrant@ansible:~$
vagrant@ansible:~$ ansible local -m file -a "path=./folder mode=644 state=directory"
ansible | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": true,
    "gid": 1000,
    "group": "vagrant",
    "mode": "0644",
    "owner": "vagrant",
    "path": "./folder",
    "size": 4096,
    "state": "directory",
    "uid": 1000
}
vagrant@ansible:~$ ls -l
total 8
-rw-rw-r-- 1 vagrant vagrant   12 Feb  2 10:38 file
drw-r--r-- 2 vagrant vagrant 4096 Feb  2 10:47 folder
vagrant@ansible:~$
```

Comandos *ad-hoc*: ejemplos

- Módulos ***yum*/*dnf*/*apt*/*package***
  - Instalación, actualización, eliminación y listado de paquetes y grupos
    - https://docs.ansible.com/ansible/latest/collections/ansible/builtin/yum_module.html
    - https://docs.ansible.com/ansible/latest/collections/ansible/builtin/dnf_module.html
    - https://docs.ansible.com/ansible/latest/collections/ansible/builtin/apt_module.html
    - https://docs.ansible.com/ansible/latest/collections/ansible/builtin/package_module.html
  - *Package* es un módulo genérico que soporta diferentes SO Linux
    - Invoca al gestor adecuado (*yum*, *apt*, *dnf*, ...) en función de la distribución

```
vagrant@ansible:~$ ansible local -m package -a "name=apache2 state=latest"
ansible | FAILED! => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1706723687,
    "cache_updated": false,
    "changed": false,
    "msg": "'/usr/bin/apt-get -y -o \"Dpkg::Options::=--force-confdef\" -o \"Dpkg::Options::=--force-confold\"
'apache2=2.4.52-1ubuntu4.7'' failed: E: Could not open lock file /var/lib/dpkg/lock-frontend - open (13: Permission
E: Unable to acquire the dpkg frontend lock (/var/lib/dpkg/lock-frontend), are you root?\n",
    "rc": 100,
    "stderr": "E: Could not open lock file /var/lib/dpkg/lock-frontend - open (13: Permission denied)\nE: Unable to
he dpkg frontend lock (/var/lib/dpkg/lock-frontend), are you root?\n",
    "stderr_lines": [
        "E: Could not open lock file /var/lib/dpkg/lock-frontend - open (13: Permission denied)",
        "E: Unable to acquire the dpkg frontend lock (/var/lib/dpkg/lock-frontend), are you root?"
    ],
    "stdout": "",
    "stdout_lines": []
}
vagrant@ansible:~$
```

Escalado de privilegios

- Ansible usa los mecanismos de escalado de privilegios del SO para ejecutar tareas usando permisos de *root* y/o los permisos de otros usuarios
  - https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_privilege_escalation.html
- El parámetro ***--become*|*-b*** del comando *ansible* **activa** el escalado de privilegios
  - El parámetro *--become-method* permite especificar el mecanismo de escalado a utilizar (el mecanismo **por defecto** es ***sudo***)
  - Mecanismos de escalado soportados:
    - *[sudo | su | pbrun | pfexec | doas | dzdo | ksu | runas | machinectl]*
  - El parámetro *--become-user* permite especificar el usuario en el que te "conviertes" (el usuario **por defecto** es ***root***)
    - **No implica *–b***
    - No configura el usuario con el que te conectas por SSH
  - El parámetro *--ask-become-pass*|*-K* permite introducir un *password* si es necesario

```
webserver ansible_become=true
```

Existen variables para gestionar el escalado de privilegios en el fichero de inventario (a nivel de *host* o a nivel de grupo): *ansible_become, ansible_become_method, ansible_become_user...*

Escalado de privilegios

- Activamos el escalado de privilegios (*-b*) usando *sudo* y el usuario *root*
  - En este ejemplo, el usuario con el que se conecta *ansible* (*vagrant*) está configurado en */etc/sudoers* para poder ejecutar comandos como *root* sin introducir el *password*

```
vagrant@ansible:~$ ansible local -b -m package -a "name=apache2 state=latest"
ansible | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1706723687,
    "cache_updated": false,
    "changed": true,
    "stderr": "",
    "stderr_lines": [],
    "stdout": "Reading package lists...\nBuilding dependency tree...\nReading state
ache2-doc apache2-suexec-pristine | apache2-suexec-custom www-browser\nThe following
e2\n0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.\nNeed to get 0 B/
n, 546 kB of additional disk space will be used.\nSelecting previously unselected pa
r(Reading database ... 5%\r(Reading database ... 10%\r(Reading database ... 15%\r(Re
 ... 25%\r(Reading database ... 30%\r(Reading database ... 35%\r(Reading database ..
g database ... 50%\r(Reading database ... 55%\r(Reading database ... 60%\r(Reading d
%\r(Reading database ... 75%\r(Reading database ... 80%\r(Reading database ... 85%\r
ase ... 95%\r(Reading database ... 100%\r(Reading database ... 86831 files and direc
 to unpack .../apache2_2.4.52-1ubuntu4.7_amd64.deb ...\r\nUnpacking apache2 (2.4.52-
4.52-1ubuntu4.7) ...\r\napache-htcacheclean.service is a disabled or a static unit n
g triggers for man-db (2.10.2-1) ...\r\nProcessing triggers for ufw (0.36.1-4ubuntu0
ART-KCUR: 5.15.0-92-generic\nNEEDRESTART-KEXP: 5.15.0-92-generic\nNEEDRESTART-KSTA:
    "stdout_lines": [
        "Reading package lists...",
        "Building dependency tree...",
        "Reading state information...",
        "Suggested packages:",
        "  apache2-doc apache2-suexec-pristine | apache2-suexec-custom www-browser",
        "The following NEW packages will be installed:",
        "  apache2",
        "0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.",
        "Need to get 0 B/97.8 kB of archives.",
        "After this operation, 546 kB of additional disk space will be used.",
```

```
vagrant@ansible:~$ ansible local -b -m apt -a "name=apache2 state=latest"
ansible | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1706723687,
    "cache_updated": false,
    "changed": false
}
vagrant@ansible:~$
```

La ejecución del comando múltiples veces es **idempotente**

Contenidos

- Introducción
- Instalación
- Conceptos básicos
- Comandos *ad-hoc*
- ***Playbooks***
- Roles
- *Collections*
- Galaxy
- Ansible + Vagrant
- Ansible + Packer
- Ansible + Docker

Introducción

- La ejecución de comandos *ad-hoc* mediante *ansible* es una herramienta útil y sencilla para la administración remota de múltiples servidores
  - Especialmente en comparación al uso de *shell scripts* ya que:
    - Suelen ser específicos de un SO/*shell*
    - Es complicado crear *scripts* idempotentes (requieren mucha lógica adicional)
    - Su sintaxis no es muy legible para los humanos (en comparación a YAML)
- Sin embargo, la verdadera potencia de Ansible reside en los ***playbooks***
  - Ficheros YAML **reutilizables** que especifican la lista de "jugadas" a ejecutar en los *hosts* para establecer en ellos un determinado **estado** de configuración
    - Cada "jugada" se mapea a un grupo de *hosts* del inventario y se encarga de una parte del objetivo global del *playbook* mediante la ejecución de una o más tareas
    - Una "jugada" es una **lista ordenada de tareas** a ejecutar en los *hosts*
    - Un *playbook* se ejecuta en orden de "jugada" (de arriba a abajo según se definen en el fichero), y en orden de tarea dentro de cada "jugada"
  - Siguen el paradigma IaC
  - Un *playbook* se ejecuta con el comando ***ansible-playbook***

Ejemplo

- *Playbook* que instala y configura el servidor Apache
  - Contiene una única "jugada" que define tres tareas

```
---
- name: Update web servers
  hosts: web
tasks:
- name: Install Apache in ubuntu
  ansible.builtin.apt:
    name: "apache2"
    state: latest
- name: Copy index.html
  ansible.builtin.copy:
    src: index.html
    dest: /var/www/html/index.html
- name: Starting a Apache Server
  ansible.builtin.service:
    name: "apache2"
    state: started
```

Las tareas de esta jugada se ejecutarán en los *hosts* del inventario que pertenecen al grupo web

Lista de tareas

Primera tarea que usa el módulo *apt* para instalar el paquete *apache2*

Playbook con dos jugadas

- Como mínimo, **cada jugada** de un *playbook* debe definir:
  - Los *hosts* del inventario en los que ejecutarse (usando un patrón)
  - Al menos una tarea a ejecutar en los *hosts* seleccionados

Primera jugada →

Lista de tareas de la primera jugada →

Segunda jugada →

Lista de tareas de la segunda jugada →

```
---
- name: Update web servers
  hosts: webservers
  remote_user: root

  tasks:
  - name: Ensure apache is at the latest version
    ansible.builtin.yum:
      name: httpd
      state: latest

  - name: Write the apache config file
    ansible.builtin.template:
      src: /srv/httpd.j2
      dest: /etc/httpd.conf

- name: Update db servers
  hosts: databases
  remote_user: root

  tasks:
  - name: Ensure postgresql is at the latest version
    ansible.builtin.yum:
      name: postgresql
      state: latest

  - name: Ensure that postgresql is started
    ansible.builtin.service:
      name: postgresql
      state: started
```

Ejecutando un *playbook*

- Sintaxis del comando *ansible-playbook*:
  - *ansible-playbook playbook.yml*
- Algunos parámetros disponibles para *ansible-playbook* son los mismos que para el comando *ansible*
  - *--inventory|-i, --user|-u, --limit|-l, --list-hosts*
  - Parámetros relacionados con el escalado de privilegios (p.e. *--become|-b*)
- Otros parámetros interesantes de *ansible-playbook* son:
  - *--check*: el *playbook* se ejecuta, pero en lugar de realizar modificaciones proporciona un informe sobre los cambios que se habrían realizado
  - *--list-tasks*: lista todas las tareas que serían ejecutadas por el *playbook* pero sin llegar a ejecutarlas
  - *--start-at-task TASK*: permite iniciar la ejecución el *playbook* en la tarea indicada como parámetro
  - *--step*: ejecuta cada tarea paso a paso solicitando confirmación previa
  - *--syntax-check*: realiza una comprobación de la sintaxis del *playbook* sin ejecutar ninguna tarea

Ejecutando un *playbook*

Playbook con **una** tarea

```
vagrant@ansible:~$ echo "Hello World" > file
vagrant@ansible:~$ cat file
Hello World
vagrant@ansible:~$ cat hello.yml
---
- hosts: all

  tasks:
    - name: Copy a file
      ansible.builtin.copy:
        src: ./file
        dest: /tmp/testfile
vagrant@ansible:~$
```

Se ejecutan **dos** tareas

```
vagrant@ansible:~$ ansible-playbook hello.yml

PLAY [all] ********************************************************************************************************

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

TASK [Copy a file] ************************************************************************************************
changed: [ansible]

PLAY RECAP ********************************************************************************************************
ansible                    : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

vagrant@ansible:~$ cat /tmp/testfile
Hello World
vagrant@ansible:~$
```

Ejecutando un *playbook*


```
vagrant@ansible:~$ cat /tmp/testfile
Hello World
vagrant@ansible:~$ ls -l /tmp/testfile
-rw-rw-r-- 1 vagrant vagrant 12 Feb  2 12:36 /tmp/testfile
vagrant@ansible:~$
vagrant@ansible:~$ ansible-playbook hello.yml

PLAY [all] *********************************************************************************************

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

TASK [Copy a file] *************************************************************************************
ok: [ansible]

PLAY RECAP *********************************************************************************************
ansible                    : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

vagrant@ansible:~$ ls -l /tmp/testfile
-rw-rw-r-- 1 vagrant vagrant 12 Feb  2 12:36 /tmp/testfile
vagrant@ansible:~$
```

Handlers

- Tareas especiales que se ejecutan **al final** de un grupo de tareas de una jugada **solo si** alguna de las tareas ha realizado algún cambio del estado
  - Se usa la opción ***notify*** indicando el nombre del manejador (*handler*) a **invocar**
  - Se ejecuta una única vez, aunque sea notificado desde múltiples tareas
  - No se ejecuta si la jugada falla antes de ser notificado
    - Se puede forzar su ejecución (parámetro *--force-handlers*)
  - Un manejador puede invocar a otro manejador
  - Un manejador puede "escuchar" (opción *listen*) *topics* genéricos a los que responder de forma que una tarea puede notificar un *topic* que agrupe múltiples *handlers*
- Ejemplo típico de uso de un *handler*:
  - Reiniciar un servicio cuando una tarea realiza un cambio en su configuración

```
vagrant@ansible:~$ cat handler.yml
---
- hosts: all

  tasks:
    - name: Enable Apache rewrite module
      community.general.apache2_module:
        state: present
        name: rewrite
      notify: restart apache

  handlers:
    - name: restart apache
      ansible.builtin.service:
        state: restarted
        name: apache2
```

Este módulo de Ansible pertenece a una *collection* que no es la integrada. Se activa un módulo de servidor Apache. Para que tenga efecto el cambio, debe reiniciarse el servicio de Apache, por lo que se invoca el *handler* que lo reinicia

Handlers


```
vagrant@ansible:~$ ansible-playbook -b handler.yml

PLAY [all] *********************************************************************

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

TASK [Enable Apache rewrite module] ********************************************
changed: [ansible]

RUNNING HANDLER [restart apache] ***********************************************
changed: [ansible]

PLAY RECAP *********************************************************************
ansible                    : ok=3    changed=2    unreachable=0    failed=0    skipped=0
```

```
vagrant@ansible:~$ ansible-playbook -b handler.yml

PLAY [all] *********************************************************************

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

TASK [Enable Apache rewrite module] ********************************************
ok: [ansible]

PLAY RECAP *********************************************************************
ansible                    : ok=2    changed=0    unreachable=0    failed=0    skipped=0
```

Variables

- Hay dos formas de declarar variables para ser usadas en un *playbook*
  - Por línea de comandos usando el parámetro *--extra-args|-e*
    - En formato *foo=bar*
      - *ansible-playbook example.yml --extra-vars "foo=bar"*
    - En formato JSON (permite definir variables numéricas, listas…)
      - *ansible-playbook example.yml --extra-vars '{"version":"1.23.45"}'*
    - Especificando un fichero JSON o YAML con las variables
      - *ansible-playbook example.yml --extra-vars "@some_file.json"*
  - En el propio *playbook*
    - Declarándolas en una sección ***vars*** en formato *foo:bar*
    - Declarándolas en un fichero YAML en formato *foo:bar* y referenciando el fichero en una sección ***vars_files*** o ***include_vars***
- Para acceder al valor de una variable se usa el motor de plantillas **Jinja2**
  - Forma básica para sustituir una variable: **{{ *variable* }}**
  - Variable de tipo lista/array: **{{ *foo[0]* }}**
  - Variable de tipo diccionario/hash: **{{ *foo['field1']* }}** y **{{ *foo.field1* }}**

Variables

- Ejemplos

```
1  ---
2  - hosts: example
3    vars:
4      foo: bar
5    tasks:
6      # Prints "Variable 'foo' is set to bar".
7      - debug: msg="Variable 'foo' is set to {{ foo }}"
```

```
1  ---
2  # Main playbook file.
3  - hosts: example
4    vars_files:
5      - vars.yml
6    tasks:
7      - debug: msg="Variable 'foo' is set to {{ foo }}"
```

```
1  ---
2  # Variables file 'vars.yml' in the same folder as the playbook.
3  foo: bar
```

Definiendo variables de tipo lista (arriba) y de tipo diccionario (abajo)

```
region:
  - northeast
  - southeast
  - midwest
foo:
  field1: one
  field2: two
```

Accediendo a una variable tipo lista

```
"{{ region[0] }}"
```

Facts

```
TASK [Gathering Facts]
ok: [ansible]
```


- Los *Facts* son variables especiales recopiladas automáticamente por Ansible en los *hosts* gestionados durante la ejecución de un *playbook*
  - Proporcionan información útil sobre los *hosts* (p.e., direcciones IP, tipo/versión de SO, espacio en disco, número de cores, ...)
- Obtener los *facts* disponibles
  - Usando el módulo ***setup***
    - Ejemplo de comando *ad-hoc*: *ansible <hostname> -m ansible.builtin.setup*
  - Uso del módulo ***debug*** en un *playbook*
- Se puede desactivar la recopilación de *facts* con el objetivo de acelerar la ejecución de un *playbook*
  - Útil si no se usan y el número de *hosts* remotos es muy elevado
  - Se desactiva incluyendo en el *playbook* la sección ***gather_facts: false***

```
- hosts: whatever
  gather_facts: false
```

- Ejemplos de acceso al *fact* "eth0" para obtener la dirección IP
  - {{ *ansible_facts["eth0"]["ipv4"]["address"]* }} | {{ *ansible_facts.eth0.ipv4.address* }}
- Ejemplo de acceso al *fact* para obtener el *hostname*
  - {{ *ansible_facts['hostname']* }} | {{ *ansible_facts.hostname* }}

Facts

- Ejemplo de uso del módulo ***setup***
  - https://docs.ansible.com/ansible/latest/collections/ansible/builtin/setup_module.html

```
vagrant@ansible:~$ ansible all -m setup -a "gather_subset=min filter=ansible_distribution*"
ansible | SUCCESS => {
    "ansible_facts": {
        "ansible_distribution": "Ubuntu",
        "ansible_distribution_file_parsed": true,
        "ansible_distribution_file_path": "/etc/os-release",
        "ansible_distribution_file_variety": "Debian",
        "ansible_distribution_major_version": "22",
        "ansible_distribution_release": "jammy",
        "ansible_distribution_version": "22.04",
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false
}
```

- Ejemplo de tarea que imprime todos los *facts* usando el módulo ***debug***
  - https://docs.ansible.com/ansible/latest/collections/ansible/builtin/debug_module.html

```
- name: Print all available facts
  ansible.builtin.debug:
    var: ansible_facts
```

Variables registradas

- Es posible definir una variable cuyo valor sea el resultado de la ejecución de una tarea y así poder ser utilizada en tareas posteriores
  - Se definen (i.e. "registran") en un *playbook* usando la palabra clave ***register***
  - Las variables registradas de esta forma solo son válidas durante la ejecución del *playbook* actual
  - El resultado producido por una tarea dependerá del módulo utilizado
    - Todos los módulos indican en su documentación los posibles valores de retorno en la sección ***Return Values***
- Ejemplo de uso: ejecutar una tarea dependiendo del resultado de otra tarea previa haciendo uso de una **sentencia condicional *when***
  - https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_conditionals.html#basic-conditionals-with-when

Variable *foo_result* definida con el resultado de la ejecución del comando *foo*

Se ejecutará esta tarea si el resultado de la tarea previa fue 5

```
- hosts: web_servers

  tasks:

     - name: Run a shell command and register its output as a variable
       ansible.builtin.shell: /usr/bin/foo
       register: foo_result
       ignore_errors: true

     - name: Run a shell command using output of the previous task
       ansible.builtin.shell: /usr/bin/bar
       when: foo_result.rc == 5
```

Loops

- Permiten repetir la ejecución de tareas un número determinado de veces usando una lista de valores como entrada

```
- name: Add user testuser1
  ansible.builtin.user:
    name: "testuser1"
    state: present
    groups: "wheel"

- name: Add user testuser2
  ansible.builtin.user:
    name: "testuser2"
    state: present
    groups: "wheel"
```

```
- name: Add several users
  ansible.builtin.user:
    name: "{{ item }}"
    state: present
    groups: "wheel"
  loop:
     - testuser1
     - testuser2
```

En este ejemplo, la lista se define en la propia tarea

```
- name: Add several users
  ansible.builtin.user:
    name: "{{ item.name }}"
    state: present
    groups: "{{ item.groups }}"
  loop:
    - { name: 'testuser1', groups: 'wheel' }
    - { name: 'testuser2', groups: 'root' }
```

Iterando sobre una lista de *hashes*

  - La lista también se puede definir como una variable en el *playbook* o en un fichero de variables y acceder mediante su nombre (*loop: "{{ somelist }}"*)
- El uso combinado con ***when*** permite ejecutar la tarea en determinados *ítems*

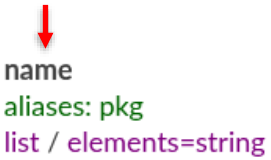
```
tasks:
    - name: Run with items greater than 5
      ansible.builtin.command: echo {{ item }}
      loop: [ 0, 2, 4, 6, 8, 10 ]
      when: item > 5
```

- Con ***until*** podemos repetir una tarea hasta cumplir una determinada condición
  - https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_loops.html#retrying-a-task-until-a-condition-is-met

Loops

- Algunos módulos aceptan una lista de valores de entrada como parámetro
  - P.e.: módulos para la instalación de paquetes (*package*, *yum*, *apt*, ...)
  - La documentación de un módulo especifica si es posible pasarle una lista como entrada en alguno de sus parámetros
  - En estos casos es preferible pasar la lista como parámetro al módulo en vez de usar un *loop* para repetir la tarea sobre los valores de la lista (ver ejemplo)

Documentación del parámetro *name* del módulo *yum*




```
- name: Optimal yum
  ansible.builtin.yum:
    name: "{{ list_of_packages }}"
    state: present

- name: Non-optimal yum, slower and may cause issues with interdependencies
  ansible.builtin.yum:
    name: "{{ item }}"
    state: present
  loop: "{{ list_of_packages }}"
```

- También es posible definir *loops* con una sintaxis alternativa: ***with_**
  - *with_list*, *with_items*, *with_dict*, ...
  - Es posible reemplazar cualquier versión *with_** con un *loop*:
    - https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_loops.html#migrating-from-with-x-to-loop

Blocks

- Permiten crear bloques que **agrupan tareas** de forma lógica
  - Las tareas del bloque heredan cualquier configuración aplicada a nivel de bloque

Bloque que contiene tres tareas

```
tasks:
  - name: Install, configure, and start Apache
    block:
      - name: Install httpd and memcached
        ansible.builtin.yum:
          name:
          - httpd
          - memcached
          state: present

      - name: Apply the foo config template
        ansible.builtin.template:
          src: templates/src.j2
          dest: /etc/foo.conf

      - name: Start service bar and enable it
        ansible.builtin.service:
          name: bar
          state: started
          enabled: True
    when: ansible_facts['distribution'] == 'CentOS'
    become: true
    become_user: root
    ignore_errors: true
```

La sentencia *when* se evalúa antes de ejecutar las tareas del bloque. Además, las tres tareas heredan el escalado de privilegios

```
tasks:
  - name: Attempt and graceful roll back demo
    block:
      - name: Print a message
        ansible.builtin.debug:
          msg: 'I execute normally'

      - name: Force a failure
        ansible.builtin.command: /bin/false

      - name: Never print this
        ansible.builtin.debug:
          msg: 'I never execute, due to the above task failing, :-('
    rescue:
      - name: Print when errors
        ansible.builtin.debug:
          msg: 'I caught an error'

      - name: Force a failure in middle of recovery! >:-)
        ansible.builtin.command: /bin/false

      - name: Never print this
        ansible.builtin.debug:
          msg: 'I also never execute :-('
    always:
      - name: Always do this
        ansible.builtin.debug:
          msg: "This always executes"
```

Gestión de errores en bloques, similar a la gestión de excepciones de un lenguaje de programación

Organización de *playbooks*

- Es posible organizar los *playbooks* con el objetivo de maximizar la reutilización y mejorar su mantenimiento, además de evitar así crear *playbooks* excesivamente largos
- Ansible proporciona **cuatro artefactos reutilizables:**
  - **Ficheros** que contengan únicamente la definición de **variables**
  - **Ficheros** que contengan únicamente la definición de **tareas**
  - ***Playbooks*** que contengan al menos una "jugada"
    - También pueden definir variables, tareas y otros
  - **Roles** que contengan un conjunto de tareas, variables, *handlers* relacionadas entre sí de acuerdo a una determinada estructura de ficheros y directorios
- Reutilización de ficheros de variables/tareas y roles
  - *include_vars, vars_files*
  - *include_tasks, import_tasks*
  - *include_role, import_role, roles*
- Reutilización de playbooks:
  - *import_playbook*

Includes/Imports

- Ejemplos
  - Reutilizando tareas:

Fichero de tareas →

```
# common_tasks.yml
- name: placeholder foo
  command: /bin/foo
- name: placeholder bar
  command: /bin/bar
```

Playbook que utiliza el fichero de tareas →

```
tasks:
- import_tasks: common_tasks.yml
# or
- include_tasks: common_tasks.yml
```

  - Reutilizando *playbooks* y roles:

Playbook que importa otros dos *playbooks* →

```
- import_playbook: webservers.yml
- import_playbook: databases.yml
```

Playbook que utiliza dos roles a "nivel de jugada" →

```
---
- hosts: webservers
  roles:
    - common
    - webservers
```

Playbook que incluye un rol a "nivel de tarea" →

```
---
- hosts: webservers
  tasks:
    - name: Print a message
      ansible.builtin.debug:
        msg: "this task runs before the example role"

    - name: Include the example role
      include_role:
        name: example

    - name: Print a message
      ansible.builtin.debug:
        msg: "this task runs after the example role"
```

Includes vs Imports

- Siendo similares, presentan algunas diferencias en como Ansible los trata y procesa internamente
  - Los *includes* se procesan a medida que se ejecuta el *playbook*: reutilización **dinámica**
    - El contenido incluido (p.e. tareas) pueden verse afectado por el resultado de la ejecución de tareas previas del *playbook*
  - Los *imports* se pre-procesan cuando se parsea el *playbook* antes de su ejecución: reutilización **estática**
    - El contenido importado nunca se verá afectado por la ejecución de otras tareas previas del *playbook*
- Además, los *includes* soportan el uso de bucles mientras que con los *imports* no es posible
  - Los *imports* también se ejecutan más rápido porque son pre-procesados
- Más detalles:
  - https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_reuse.html#comparing-includes-and-imports-dynamic-and-static-reuse

Contenidos

- Introducción
- Instalación
- Conceptos básicos
- Comandos *ad-hoc*
- *Playbooks*
- **Roles**
- *Collections*
- Galaxy
- Ansible + Vagrant
- Ansible + Packer
- Ansible + Docker

Roles

- Un rol **integra** tareas, variables, *handlers*, ficheros, módulos, etc de forma que se pueden reutilizar en múltiples *playbooks* y compartir con otros usuarios mediante el repositorio Ansible Galaxy
  - Permiten la división lógica de un *playbook* en múltiples componentes que son fácilmente reutilizables y mantenibles
- Se basan en una jerarquía y nomenclatura determinada de todos los ficheros y directorios que componen el rol
  - Dicha estructura se puede crear manualmente o bien utilizando el comando:
    - *ansible-galaxy role init <role_name>*

Directorio y ficheros que componen un rol

```
vagrant@ansible:~$ ansible-galaxy role init myrol
- Role myrol was created successfully
vagrant@ansible:~$ tree myrol/
myrol/
├── README.md
├── defaults
│   └── main.yml
├── files
├── handlers
│   └── main.yml
├── meta
│   └── main.yml
├── tasks
│   └── main.yml
├── templates
├── tests
│   ├── inventory
│   └── test.yml
└── vars
    └── main.yml

8 directories, 8 files
vagrant@ansible:~$
```

Estructura de un rol

- ***defaults:*** *main.yml*
  - Contiene la definición por defecto de las variables que usa el rol (p.e. puerto de escucha por defecto de un servidor web) y que puede ser necesario modificar por parte del usuario (variables "dinámicas")
  - Estas variables tienen **la menor prioridad** por lo que son fáciles de sobrescribir
- ***vars****: main.yml*
  - Contiene la definición de otras variables utilizadas por el rol cuyo valor no se espera que sea necesario ser sobrescrito con frecuencia (variables "estáticas")
  - Misma función que *defaults* pero **con mayor prioridad**
- ***files***
  - Ficheros estáticos que son necesarios para el funcionamiento del rol, los cuáles suele ser necesario copiar a los *hosts* remotos mediante el módulo *copy* (p.e. código fuente de una aplicación, fichero de configuración estático)
- ***templates***
  - Parecido a *files*, pero estos ficheros funcionan a modo de plantillas siendo necesario modificarlas mediante el módulo *template*
- ***handlers****: main.yml*
  - Contienes la definición de los manejadores

Estructura de un rol

- ***tasks**: main.yml*
  - Contiene las tareas a ejecutar por el rol
  - Pueden estar definidas en el fichero *main.yml* o en cualquier otro fichero YAML siempre y cuando se haga referencia al mismo desde *main.yml*
    - Útil para definir tareas específicas del SO en ficheros separados

Fichero *main.yml* que importa dos ficheros de tareas usando sentencias condicionales que comprueban la familia Linux usando *Facts*

```
# roles/example/tasks/main.yml
- name: Install the correct web server for RHEL
  import_tasks: redhat.yml
  when: ansible_facts['os_family']|lower == 'redhat'

- name: Install the correct web server for Debian
  import_tasks: debian.yml
  when: ansible_facts['os_family']|lower == 'debian'
```

Fichero que instala Apache en sistemas RHEL y derivados (p.e. Rocky Linux)

```
# roles/example/tasks/redhat.yml
- name: Install web server
  ansible.builtin.yum:
    name: "httpd"
    state: present
```

Fichero que instala Apache en sistemas Debian y derivados (p.e. Ubuntu)

```
# roles/example/tasks/debian.yml
- name: Install web server
  ansible.builtin.apt:
    name: "apache2"
    state: present
```

Estructura de un rol

- ***tests**: test.yml inventory*
  - Tests del rol (sintaxis de los ficheros YAML, su idempotencia, etc)
  - Los tests se ejecutan en los *hosts* definidos en el fichero *inventory*
    - Ansible Molecule está diseñado para ayudar en el desarrollo y *testing* de roles de Ansible con soporte para tests con múltiples instancias, SO y escenarios de prueba
- ***meta**: main.yml*
  - Contiene metadatos del rol (p.e. autor, versión, licencia)
  - También se definen en este fichero las **dependencias** con otros roles así como con ***collections***

```
# roles/myapp/meta/main.yml
---
dependencies:
  - role: common
    vars:
      some_parameter: 3
  - role: apache
    vars:
      apache_port: 80
  - role: postgres
    vars:
      dbname: blarg
      other_parameter: 12
```

Usando roles en un *playbook*

- Mediante la sección *roles* a nivel de "jugada"

```
---
- hosts: webservers
  roles:
    - common
    - webservers
```

- En la sección *tasks* a nivel de tarea mediante ***import_role*** o ***include_role***
  - También es posible hacerlo de forma condicional usando ***when***

```
---
- hosts: webservers
  tasks:
    - debug:
        msg: "before we run our role"
    - import_role:
        name: example
    - include_role:
        name: example
    - debug:
        msg: "after we ran our role"
```

```
---
- hosts: webservers
  tasks:
    - include_role:
        name: some_role
      when: "ansible_facts['os_family'] == 'RedHat'"
```

Contenidos

- Introducción
- Instalación
- Conceptos básicos
- Comandos *ad-hoc*
- *Playbooks*
- Roles
- ***Collections***
- Galaxy
- Ansible + Vagrant
- Ansible + Packer
- Ansible + Docker

Collections

- Formato introducido en la versión 2.8 que permite distribuir todo tipo de contenido Ansible (*playbooks*, roles, módulos, *plugins*) en un único paquete
  - El *core* de Ansible incluye la *collection* por defecto (FQCN: ***ansible.builtin***)
    - https://docs.ansible.com/ansible/latest/collections/ansible/builtin/index.html
  - En Galaxy también disponemos de *collections* desarrolladas por la comunidad
- De forma similar a un rol, una *collection* sigue una estructura determinada de ficheros y directorios que se puede generar mediante el siguiente comando
  - *ansible-galaxy collection init <namespace.collection_name>*

El fichero *galaxy.yml* contiene la información necesaria para construir y empaquetar una *collection*

```
collection/
├── docs/
├── galaxy.yml
├── meta/
│   └── runtime.yml
├── plugins/
│   ├── modules/
│   │   └── module1.py
│   ├── inventory/
│   └── .../
├── README.md
├── roles/
│   ├── role1/
│   ├── role2/
│   └── .../
├── playbooks/
│   ├── files/
│   ├── vars/
│   ├── templates/
│   └── tasks/
└── tests/
```

El fichero *runtime.yml* contiene metadatos adicionales como por ejemplo requerir una versión mínima de Ansible para la *collection*

Usando *collections* en *playbooks*/roles

- **Desde un *playbook*,** se puede referenciar cualquier contenido de una *collection* **especificando su FQCN** (figura izquierda)

```yaml
- name: Reference collections contents using their FQCNs
  hosts: all
  tasks:

    - name: Import a role
      ansible.builtin.import_role:
        name: my_namespace.my_collection.role1

    - name: Call a module
      my_namespace.mycollection.my_module:
        option1: value
```

```yaml
- name: Run a play using the collections keyword
  hosts: all
  collections:
    - my_namespace.my_collection

  tasks:

    - name: Import a role
      ansible.builtin.import_role:
        name: role1

    - name: Run a module not specifying FQCN
      my_module:
        option1: value
```

Desde la versión 2.8, la **sección *collections*** definida a nivel de "jugada" permite listar colecciones en las que el *playbook* debe buscar los roles y módulos sin necesidad de usar su FQCN

- **Desde un rol,** las *collections* se definen en el fichero *meta/main.yml*

```yaml
# myrole/meta/main.yml
collections:
  - my_namespace.first_collection
  - my_namespace.second_collection
  - other_namespace.other_collection
```

Contenidos

- Introducción
- Instalación
- Conceptos básicos
- Comandos *ad-hoc*
- *Playbooks*
- Roles
- *Collections*
- **Galaxy**
- Ansible + Vagrant
- Ansible + Packer
- Ansible + Docker

¿Qué es Ansible Galaxy?

- Galaxy es un repositorio público y gratuito que contiene roles y *collections* de Ansible desarrollados y soportados por la comunidad
  - https://galaxy.ansible.com
- Cualquier usuario puede usar los roles/*collections* disponibles en Galaxy mediante el comando ***ansible-galaxy***
  - https://docs.ansible.com/ansible/latest/cli/ansible-galaxy.html
  - Mediante GitHub los usuarios también pueden contribuir con su propio contenido a la comunidad Ansible Galaxy
- Galaxy es un proyecto de código abierto, con lo que también es posible desplegar un repositorio Galaxy privado
  - https://github.com/ansible/galaxy
  - El comando *ansible-galaxy* usa por defecto el repositorio público
  - Dicho repositorio por defecto es configurable en */etc/ansible/ansible.cfg*
  - La opción *--server|-s* del comando *ansible-galaxy* permite especificar un repositorio diferente

Ansible Galaxy vs Ansible Automation Hub

- Ansible Automation Hub es un servicio ofrecido por Red Hat, inc que ofrece, entre otras cosas, un repositorio Galaxy
  - https://www.ansible.com/products/automation-hub
- Proporciona *collections* cuyo contenido está **certificado y soportado** comercialmente por Red Hat, Inc
  - El acceso al Automation Hub forma parte de la subscripción de pago al producto Red Hat Ansible Automation Platform
    - https://www.ansible.com/products/automation-platform
  - Es compatible con Galaxy y en esencia son muy similares entre sí
    - Se accede al Automation Hub mediante el mismo comando *ansible-galaxy* previamente configurado para usar un determinado *token* de acceso

Gestión de roles

- Inicializar un nuevo rol
  - *ansible-galaxy role init <role_name>*
- Buscar un rol
  - *ansible-galaxy role search <searchterm>*
- Obtener información sobre un rol
  - *ansible-galaxy role info <username.role_name>*
- Instalar un rol (desde fichero, desde una URL o desde Galaxy)
  - *ansible-galaxy role install <namespace.role_name>*
- Instalar una versión específica de un rol
  - *ansible-galaxy role install <namespace.role_name>,v1.0.0*
- Listar roles instalados y sus versiones
  - *ansible-galaxy role list*
- Eliminar un rol instalado previamente
  - *ansible-galaxy role remove <namespace.role_name>*

Gestión de roles: ejemplo

- Búsqueda de roles por autor y nombre

```
vagrant@ansible:~$ ansible-galaxy role search --author geerlingguy nginx

Found 1 roles matching your search:

 Name              Description
 ----              -----------
 geerlingguy.nginx Nginx installation for Linux, FreeBSD and OpenBSD.
vagrant@ansible:~$
vagrant@ansible:~$ ansible-galaxy role search --author geerlingguy

Found 101 roles matching your search:

 Name                              Description
 ----                              -----------
 geerlingguy.adminer               Installs Adminer for Database management.
 geerlingguy.ansible               Ansible for RedHat/CentOS/Debian/Ubuntu.
 geerlingguy.apache                Apache 2.x for Linux.
 geerlingguy.apache-php-fpm        Apache 2.4+ PHP-FPM support for Linux.
 geerlingguy.aws-inspector         AWS Inspector installation for Linux.
 geerlingguy.awx                   Installs and configures AWX (Ansible Tower's
 geerlingguy.awx-container         Ansible AWX container for Docker.
 geerlingguy.backup                Backup for Simple Servers.
```

Gestión de roles: ejemplo

- Instalación del rol para Nginx

```
vagrant@ansible:~$ ansible-galaxy role install geerlingguy.nginx
Starting galaxy role install process
- downloading role 'nginx', owned by geerlingguy
- downloading role from https://github.com/geerlingguy/ansible-role-nginx/archive/3.1.4.tar.gz
- extracting geerlingguy.nginx to /home/vagrant/.ansible/roles/geerlingguy.nginx
- geerlingguy.nginx (3.1.4) was installed successfully
vagrant@ansible:~$
vagrant@ansible:~$ ls .ansible/roles/geerlingguy.nginx/
LICENSE  README.md  defaults  handlers  meta  molecule  tasks  templates  vars
vagrant@ansible:~$
vagrant@ansible:~$ ansible-galaxy role list
# /home/vagrant/.ansible/roles
- geerlingguy.nginx, 3.1.4
# /etc/ansible/roles
[WARNING]: - the configured path /usr/share/ansible/roles does not exist.
vagrant@ansible:~$
```

- La opción *-p* permite indicar el directorio destino de un rol

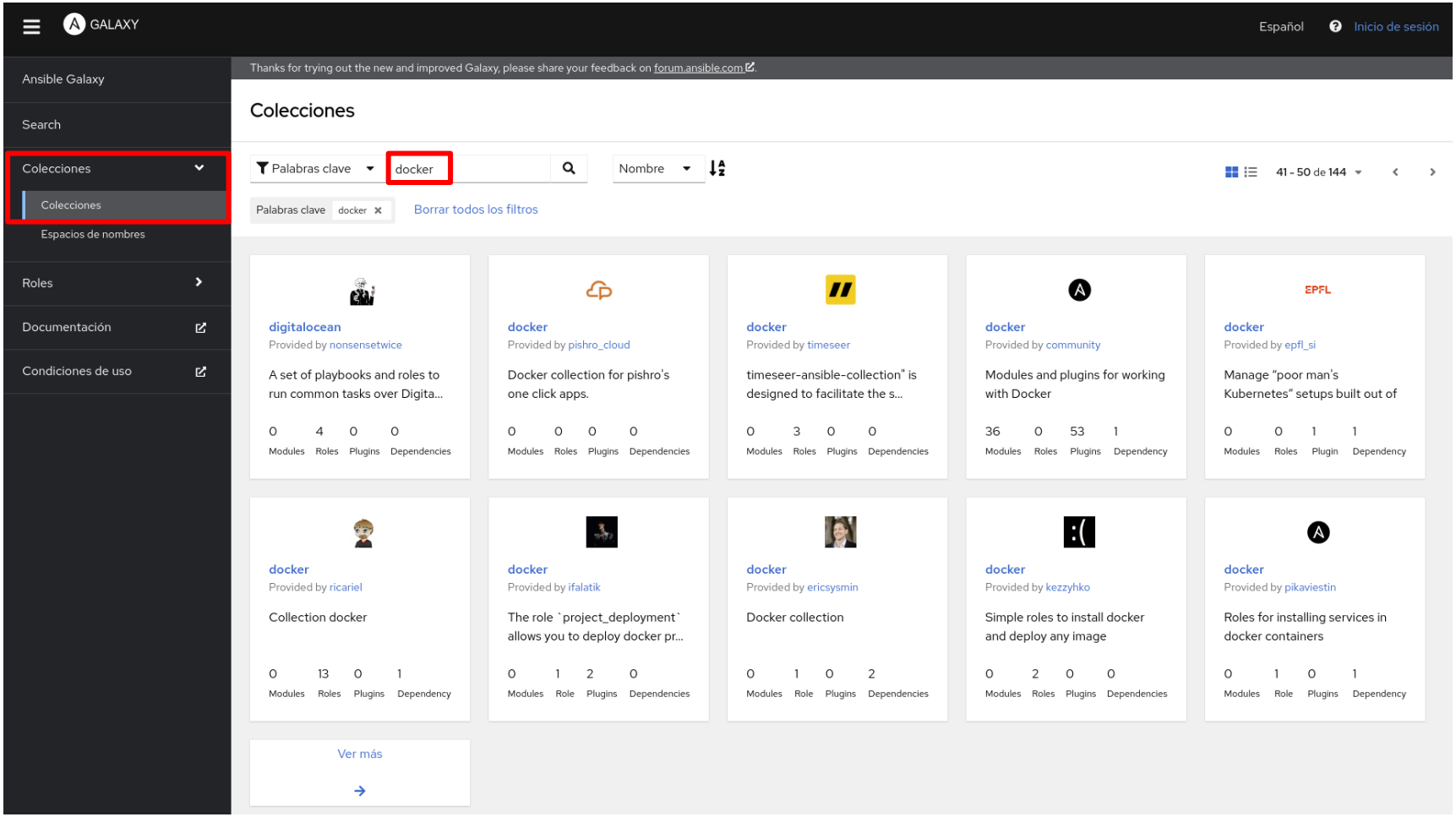
```
vagrant@ansible:~$ ansible-galaxy role install geerlingguy.java -p ./roles
Starting galaxy role install process
- downloading role 'java', owned by geerlingguy
- downloading role from https://github.com/geerlingguy/ansible-role-java/archive/2.3.3.tar.gz
- extracting geerlingguy.java to /home/vagrant/roles/geerlingguy.java
- geerlingguy.java (2.3.3) was installed successfully
vagrant@ansible:~$
vagrant@ansible:~$ ls roles/geerlingguy.java/
LICENSE  README.md  defaults  meta  molecule  tasks  templates  vars
vagrant@ansible:~$
vagrant@ansible:~$ ansible-galaxy role list -p roles
# /home/vagrant/roles
- geerlingguy.java, 2.3.3
# /home/vagrant/.ansible/roles
- geerlingguy.nginx, 3.1.4
# /etc/ansible/roles
[WARNING]: - the configured path /usr/share/ansible/roles does not exist.
vagrant@ansible:~$
```

Gestión de *collections*

- Buscar una *collection*
  - Solo es posible desde el sitio web: https://galaxy.ansible.com/ui/collections/
- Inicializar una nueva *collection*
  - *ansible-galaxy collection init <collection_name>*
- Instalar una *collection* (desde fichero, desde una URL o desde Galaxy)
  - *ansible-galaxy collection install <namespace.collection>*
- Instalar una versión específica de una *collection*
  - *ansible-galaxy collection install <namespace.collection>:1.0.0*
- Construir una *collection* para ser publicada en Galaxy
  - *ansible-galaxy collection build <collection_dir>*
- Publicar una *collection* en Galaxy
  - *ansible-galaxy collection publish <collection_path>*
- Listar *collections* instaladas y sus versiones (solo Ansible >= 2.10)
  - *ansible-galaxy collection list*
- Descargar una *collection* (solo Ansible >= 2.10)
  - *ansible-galaxy collection download <collection_path>*

Gestión de *collections*: ejemplo

- Búsqueda de una *collection*



Gestión de *collections*: ejemplo

- Instalación de una *collection* para Prometheus
  - https://galaxy.ansible.com/ui/repo/published/cloin/prometheus/

```
vagrant@ansible:~$ ansible-galaxy collection install cloin.prometheus
Starting galaxy collection install process
Process install dependency map
Starting collection install process
Downloading https://galaxy.ansible.com/api/v3/plugin/ansible/content/published/collections/artifacts/cloin-prome
grant/.ansible/tmp/ansible-local-2206rm_ysfpb/tmpkkvzlh6s/cloin-prometheus-0.1.0-9jwnuafq
Installing 'cloin.prometheus:0.1.0' to '/home/vagrant/.ansible/collections/ansible_collections/cloin/prometheus'
cloin.prometheus:0.1.0 was installed successfully
vagrant@ansible:~$
vagrant@ansible:~$ ls .ansible/collections/ansible_collections/cloin/prometheus/
FILES.json  LICENSE  MANIFEST.json  README.md  docs  meta  plugins  requirements.txt  roles
vagrant@ansible:~$
```

- La opción *-p* permite indicar el directorio destino

```
vagrant@ansible:~$ ansible-galaxy collection install geerlingguy.php_roles -p collections
Starting galaxy collection install process
[WARNING]: The specified collections path '/home/vagrant/collections' is not part of the configured Ansible collec
'/home/vagrant/.ansible/collections:/usr/share/ansible/collections'. The installed collection will not be picked u
within a playbook-adjacent collections directory.
Process install dependency map
Starting collection install process
Downloading https://galaxy.ansible.com/api/v3/plugin/ansible/content/published/collections/artifacts/geerlingguy-p
me/vagrant/.ansible/tmp/ansible-local-2236g0lzlviv/tmpnefx7w8s/geerlingguy-php_roles-3.0.0-o6lknfr_
Installing 'geerlingguy.php_roles:3.0.0' to '/home/vagrant/collections/ansible_collections/geerlingguy/php_roles'
geerlingguy.php_roles:3.0.0 was installed successfully
'community.general:7.5.2' is already installed, skipping.
vagrant@ansible:~$
vagrant@ansible:~$ ls collections/ansible_collections/geerlingguy/php_roles/
FILES.json  LICENSE  MANIFEST.json  README.md  galaxy-deploy.yml  meta  roles
vagrant@ansible:~$
```

Instalar roles/*collections* desde fichero

- Es posible instalar múltiples roles y *collections* desde un fichero YAML (***requirements.yml***) donde deben ser definidos
  - Instalar roles y *collections*:
    - *ansible-galaxy install -r requirements.yml*
  - Es posible instalarlos por separado:
    - Instalar solo roles: *ansible-galaxy role install -r requirements.yml*
    - Instalar solo *collections*: *ansible-galaxy collection install -r requirements.yml*
- Ejemplo de fichero *requirements.yml*

```yaml
---
roles:
  # Install a role from Ansible Galaxy.
  - name: geerlingguy.java
    version: "1.9.6" # note that ranges are not supported for roles

collections:
  # Install a collection from Ansible Galaxy.
  - name: community.general
    version: ">=7.0.0"
    source: https://galaxy.ansible.com
```

Contenidos

- Introducción
- Instalación
- Conceptos básicos
- Comandos *ad-hoc*
- *Playbooks*
- Roles
- *Collections*
- Galaxy
- **Ansible + Vagrant**
- Ansible + Packer
- Ansible + Docker

Aprovisionando *guests* Vagrant con Ansible

- Vagrant proporciona dos *provisioners* distintos que pueden ser usados para aprovisionar las VM usando *playbooks* de Ansible
  - ***ansible***
    - El comando *ansible-playbook* se ejecuta en el equipo anfitrión (***host***)
    - https://developer.hashicorp.com/vagrant/docs/provisioning/ansible
  - ***ansible_local***
    - El comando *ansible-playbook* se ejecuta en la VM (***guest***)
    - https://developer.hashicorp.com/vagrant/docs/provisioning/ansible_local
- El fichero *inventory* puede ser generado automáticamente por Vagrant
  - https://developer.hashicorp.com/vagrant/docs/provisioning/ansible_intro#auto-generated-inventory
  - También es posible usar un fichero específico con la opción *inventory_path*
    - https://developer.hashicorp.com/vagrant/docs/provisioning/ansible_common

Provisioner *ansible*

- Es necesario instalar Ansible en el *host* que ejecuta Vagrant

```
Vagrant.configure("2") do |config|
  config.vm.provision "ansible" do |ansible|
    ansible.playbook = "provisioning/playbook.yml"
  end
end
```

```
.
|-- Vagrantfile
|-- provisioning
|   |-- group_vars
|           |-- all
|   |-- roles
|           |-- bar
|           |-- foo
|   |-- playbook.yml
```

Provisioner *ansible_local*

- Es necesario instalar Ansible en la VM
  - Por defecto, **Vagrant intenta instalar Ansible en la VM automáticamente**
  - https://developer.hashicorp.com/vagrant/docs/provisioning/ansible_local#install

```
Vagrant.configure("2") do |config|
  # Run Ansible from the Vagrant VM
  config.vm.provision "ansible_local" do |ansible|
    ansible.playbook = "playbook.yml"
  end
end
```

Requirements:

- The `playbook.yml` file is stored in your Vagrant's project home directory.
- The default shared directory is enabled (`.` → `/vagrant`).

Contenidos

- Introducción
- Instalación
- Conceptos básicos
- Comandos *ad-hoc*
- *Playbooks*
- Roles
- *Collections*
- Galaxy
- Ansible + Vagrant
- **Ansible + Packer**
- Ansible + Docker

Aprovisionando imágenes Packer con Ansible

- De forma similar a Vagrant, Packer dispone de un *plugin* que proporciona dos *provisioners* de Ansible para aprovisionar las imágenes
  - ***ansible***
    - El comando *ansible-playbook* se ejecuta en el ***host***
    - https://developer.hashicorp.com/packer/integrations/hashicorp/ansible/latest/components/provisioner/ansible
    - Es necesario instalar Ansible en el ***host*** que ejecuta Vagrant
  - ***ansible_local***
    - El comando *ansible-playbook* se ejecuta en la VM (***guest***)
    - https://developer.hashicorp.com/vagrant/docs/provisioning/ansible_local#install
    - En este caso, Packer no instala automáticamente Ansible en la VM así que debe instalarse previamente usando, por ejemplo, un *provisioner shell*

```
build {
    sources = [
        "source.digitalocean.example"
    ]

    provisioner "ansible" {
      playbook_file = "./playbook.yml"
    }
```

Contenidos

- Introducción
- Instalación
- Conceptos básicos
- Comandos *ad-hoc*
- *Playbooks*
- Roles
- *Collections*
- Galaxy
- Ansible + Vagrant
- Ansible + Packer
- **Ansible + Docker**

Gestión de contenedores Docker con Ansible

- La *collection* ***community.docker*** proporciona múltiples módulos que permiten gestionar contenedores Docker mediante *playbooks*
  - https://docs.ansible.com/ansible/latest/collections/community/docker/index.html
  - https://galaxy.ansible.com/ui/repo/published/community/docker/
- Módulos relevantes de la *collection* de Docker:
  - *docker_container*: gestión del ciclo de vida de contenedores
  - *docker_container_info*: inspección de contenedores
  - *docker_image*: gestión de imágenes
  - *docker_image_info*: inspección de imágenes
  - *docker_network*: gestión de redes
  - *docker_compose_v2*: gestión de contenedores mediante ficheros *compose*
  - *docker_swarm_service*: gestión de servicios Swarm
  - *docker_swarm*: gestión de un clúster Swarm

Al instalar Ansible, por defecto se instalan múltiples *collections* incluyendo la de Docker

```
vagrant@ansible:~$ ansible-galaxy collection list | grep docker
community.docker                3.4.11
vagrant@ansible:~$
```

Ejemplo

- Ejecutaremos un *playbook* que se encargará de:
  - Crear una imagen Docker usando el módulo ***docker_image***
    - La imagen se basa en Debian e instala el servidor web Nginx modificando su página de inicio
    - Contenido del *Dockerfile*

```
vagrant@ansible:~$ cat nginx/Dockerfile
FROM debian:stable-slim
RUN apt-get update
RUN apt-get install -y nginx
RUN echo "Nginx is running in Docker" > /var/www/html/index.html
EXPOSE 80
ENTRYPOINT ["nginx", "-g", "daemon off;"]
```

  - Iniciar un contenedor Docker usando la imagen previamente creada mediante el módulo ***docker_container***
  - Comprobar los contenedores en ejecución

Ejemplo

- Contenido del *playbook*

```
vagrant@ansible:~$ cat docker.yml
---
- hosts: all

  tasks:
    - name: Create Docker image
      community.docker.docker_image:
        name: mynginx
        source: build
        build:
          path: /home/vagrant/nginx
          pull: yes
        state: present

    - name: Run Docker container
      community.docker.docker_container:
        image: mynginx
        name: web
        state: started

    - name: Check if the container is running
      ansible.builtin.shell: docker ps
      register: containers

    - name: Print variable
      ansible.builtin.debug:
        var: containers.stdout
```

El fichero *Dockerfile* está en esta ruta

Esta tarea imprime la salida estándar (*stdout*) resultado de la ejecución de la tarea previa, la cual registro la variable *containers*

Ejemplo

- Ejecución del *playbook* y comprobaciones posteriores

```
vagrant@ansible:~$ ansible-playbook docker.yml

PLAY [all] *********************************************************************************************************

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

TASK [Create Docker image] *****************************************************************************************
changed: [ansible]

TASK [Run Docker container] ****************************************************************************************
changed: [ansible]

TASK [Check if the container is running] ***************************************************************************
changed: [ansible]

TASK [Print variable] **********************************************************************************************
ok: [ansible] => {
    "containers.stdout": "CONTAINER ID   IMAGE     COMMAND                  CREATED                  STATUS
bc7acd3ceaee   mynginx   \"nginx -g 'daemon of…\"   Less than a second ago   Up Less than a second   80/tcp    web"
}

PLAY RECAP *********************************************************************************************************
ansible                    : ok=5    changed=3    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

```
vagrant@ansible:~$ docker image ls
REPOSITORY   TAG           IMAGE ID       CREATED          SIZE
mynginx      latest        259d544c0f72   14 seconds ago   110MB
debian       stable-slim   0cffa97cb601   5 days ago       74.8MB
vagrant@ansible:~$
vagrant@ansible:~$ docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS          PORTS     NAMES
1a3876c399f0   mynginx   "nginx -g 'daemon of…"   17 seconds ago   Up 16 seconds   80/tcp    web
vagrant@ansible:~$
vagrant@ansible:~$ docker inspect --format "{{.NetworkSettings.IPAddress}}" web
172.17.0.2
vagrant@ansible:~$ curl http://172.17.0.2
Nginx is running in Docker
vagrant@ansible:~$
```