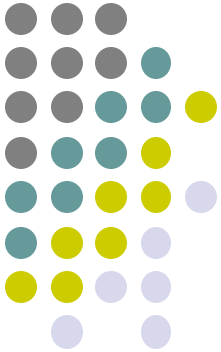


ADMINISTRACIÓN DE INFRAESTRUCTURAS Y SISTEMAS INFORMÁTICOS (AISI)

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://www.ansible.com

Características principales

- Su popularidad y facilidad de uso radica en dos pilares:
  1. La necesidad de instalar Ansible en una **única** máquina (**controlador**) que actúa como punto central desde el cual se gestionan los servidores (*hosts*) de forma **remota** a través de SSH (por defecto)
     - El controlador requiere tener instalado Python 2.X/Python 3.X
       - Ansible soporta Linux/macOS/BSD en el controlador pero NO Windows
     - Los servidores gestionados con Ansible solo requieren disponer de Python 2.X/3.X, sin necesidad de instalar/ejecutar ningún otro agente *software* adicional ni crear/mantener una base de datos centralizada
       - Por eso se dice que su arquitectura es ***agentless*** (sin agentes)
       - No hay ningún consumo de recursos en los *hosts* gestionados con Ansible cuando no se está ejecutando ninguna acción sobre ellos (no existen demonios en ejecución en segundo plano)

Características principales

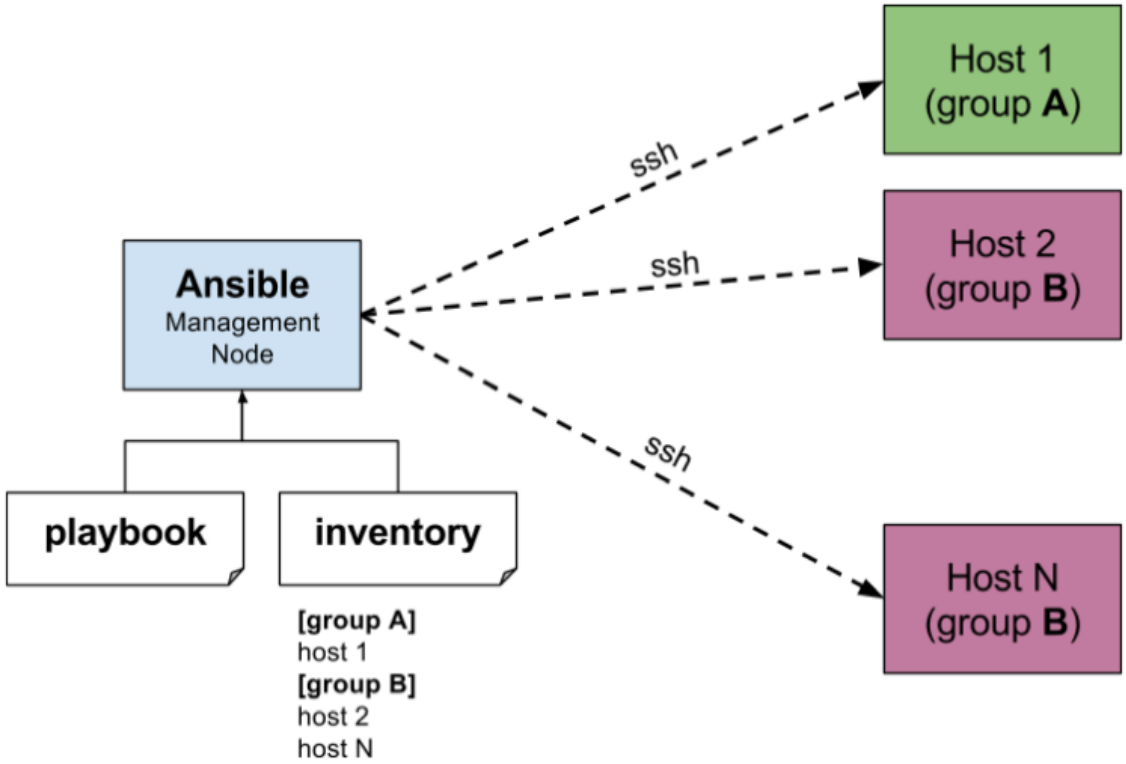
- Su popularidad y facilidad de uso radica en dos pilares:
  1. La necesidad de instalar Ansible en una **única** máquina (**controlador**) que actúa como punto central desde el cual se gestionan los servidores (*hosts*) de forma **remota** a través de SSH (por defecto)
     - El controlador requiere tener instalado Python 2.X/Python 3.X
       - Ansible soporta Linux/macOS/BSD en el controlador pero NO Windows
     - Los servidores gestionados con Ansible solo requieren disponer de Python 2.X/3.X, sin necesidad de instalar/ejecutar ningún otro agente *software* adicional ni crear/mantener una base de datos centralizada
       - Por eso se dice que su arquitectura es ***agentless*** (sin agentes)
       - No hay ningún consumo de recursos en los *hosts* 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 usa un lenguaje simple y legible basado en YAML para describir las acciones y las configuraciones a realizar en los *hosts* gestionados con Ansible
     - Ansible Tower 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 a herramientas IaC de gestión de la configuración similares como Puppet, Chef, CFEngine…



Inventario, módulo, tarea y playbook

- **Inventario**
  - Fichero que contiene la lista de FQDNs/IPs de los servidores o *hosts* a gestionar
  - Los nodos pueden organizarse en [**grupos**]
- **Módulo**
  - Unidad básica de código que Ansible puede ejecutar en un *host*
  - El comando ***ansible*** permite ejecutar un módulo una única vez (**comando ad-hoc**)
  - Cada módulo proporciona una funcionalidad particular (e.g. gestión de usuarios, copiar ficheros, instalación de paquetes, gestión de servicios)
    - https://docs.ansible.com/ansible/latest/collections/index_module.html
  - La inmensa mayoría de los módulos son **idempotentes**
- **Tarea**
  - Unidad básica de acción en Ansible (i.e. ejecutar un módulo)
- **Playbook**
  - Fichero YAML que contiene una lista ordenada de "jugadas"
  - Una "jugada" se compone de una o más tareas que ejecutarán módulos Ansible
  - Los playbooks pueden incluir/definir variables además de tareas
  - El comando ***ansible-playbook*** permite ejecutar un playbook

¿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 *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, usualmente mediante SSH (por defecto)
     - La conexión por SSH se realiza con el usuario actual
       - Se puede indicar otro usuario con el parámetro *--user* | *-u*
     - Por defecto, se asume autenticación SSH *passwordless*
       - En caso 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 varios formatos, siendo INI y YAML los más utilizados
- Existen siempre 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:
```

Formato del inventario: variables

- Es posible definir variables de *host* (izquierda) y 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 **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
  - Puede especificar un *host* en concreto, todos los *hosts* (all), un grupo de *hosts*...
    - https://docs.ansible.com/ansible/latest/user_guide/intro_patterns.html#intro-patterns
  - 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 módulos son **idempotentes:** comprueban el estado actual del *host* y no realizan ninguna acción salvo que sea diferente al estado deseado

Comandos ad-hoc: ejemplos

- Módulo ***command***
  - https://docs.ansible.com/ansible/latest/collections/ansible/builtin/command_module.html#command-module
  - Es el módulo por defecto
  - Ejecutamos el comando "*echo Hello World*" especificando diferentes patrones

```
[rober@oceania ~]$ cat /etc/ansible/hosts
[local]
localhost
[rober@oceania ~]$ ansible all -a "echo Hello World"
localhost | CHANGED | rc=0 >>
Hello World

[rober@oceania ~]$ ansible local -a "echo Hello World"
localhost | CHANGED | rc=0 >>
Hello World

[rober@oceania ~]$ ansible localhost -a "echo Hello World"
localhost | CHANGED | rc=0 >>
Hello World

[rober@oceania ~]$ ansible all --limit localhost -a "echo Hello World"
localhost | CHANGED | rc=0 >>
Hello World

[rober@oceania ~]$ ansible all --limit local --list-hosts -a "echo Hello World"
  hosts (1):
    localhost
```

Comandos ad-hoc: ejemplos

- Módulo ***ping***
  - https://docs.ansible.com/ansible/latest/collections/ansible/builtin/ping_module.html

```
[rober@oceania ~]$ cat /etc/ansible/hosts
[local]
localhost
[rober@oceania ~]$ ansible localhost -m ping
localhost | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false,
    "ping": "pong"
}
[rober@oceania ~]$ ansible all -m ping
localhost | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false,
    "ping": "pong"
}
```

Comandos ad-hoc: ejemplos

- Módulos ***yum*/*dnf***
  - https://docs.ansible.com/ansible/latest/collections/ansible/builtin/yum_module.html
  - https://docs.ansible.com/ansible/latest/collections/ansible/builtin/dnf_module.html
  - Comprobamos que el paquete *wget* está actualizado a su última versión disponible en los repositorios de la distribución

```
[rober@oceania ~]$ ansible local -m yum -a "name=wget state=latest"
localhost | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false,
    "changes": {
        "installed": [],
        "updated": []
    },
    "msg": "",
    "rc": 0,
    "results": [
        "All packages providing wget are up to date",
        ""
    ]
}
[rober@oceania ~]$
```

Comandos ad-hoc: ejemplos

- Módulo ***package***
  - https://docs.ansible.com/ansible/latest/collections/ansible/builtin/package_module.html
  - Módulo de gestión de paquetes genérico que soporta diferentes SO Linux
  - Invoca el gestor adecuado (*yum*, *apt*, *dnf*, ...) en función de la distribución

```
[rober@oceania ~]$ ansible local -m package -a "name=wget state=latest"
localhost | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false,
    "changes": {
        "installed": [],
        "updated": []
    },
    "msg": "",
    "rc": 0,
    "results": [
        "All packages providing wget are up to date",
        ""
    ]
}
[rober@oceania ~]$
```

Comandos ad-hoc: ejemplos

- Módulo ***service***
  - https://docs.ansible.com/ansible/latest/collections/ansible/builtin/service_module.html
  - Comprobamos que el servicio de red está en ejecución

```
[rober@oceania ~]$ ansible local -m service -a "name=network state=started"
localhost | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false,
    "name": "network",
    "state": "started",
    "status": {
        "ActiveEnterTimestamp": "jue 2020-01-09 17:10:40 CET",
        "ActiveEnterTimestampMonotonic": "7260256",
        "ActiveExitTimestampMonotonic": "0",
        "ActiveState": "active",
        "After": "network-pre.target systemd-journald.socket ip6tables.service
ervice",
```

Comandos ad-hoc: ejemplos

- Módulo ***copy***
  - https://docs.ansible.com/ansible/latest/collections/ansible/builtin/copy_module.html
  - Copiamos un fichero de forma remota desde el nodo controlador a un *host* del inventario

```
[rober@oceania ~]$ ansible local -m copy -a "src=/tmp/file dest=/home/rober"
localhost | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": true,
    "checksum": "da39a3ee5e6b4b0d3255bfef95601890afd80709",
    "dest": "/home/rober/file",
    "gid": 1000,
    "group": "rober",
    "md5sum": "d41d8cd98f00b204e9800998ecf8427e",
    "mode": "0664",
    "owner": "rober",
    "size": 0,
    "src": "/home/rober/.ansible/tmp/ansible-tmp-1580459703.02-86207426217556/source",
    "state": "file",
    "uid": 1000
}
```

Comandos ad-hoc: ejemplos

- Módulo ***file***
  - https://docs.ansible.com/ansible/latest/collections/ansible/builtin/file_module.html
  - Creamos un directorio "*folder*" con permisos rw-r--r-- (*mode=644*)

```
[rober@oceania ~]$ ansible local -m file -a "path=/home/rober/folder mode=644 state=directory"
localhost | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": true,
    "gid": 1000,
    "group": "rober",
    "mode": "0644",
    "owner": "rober",
    "path": "/home/rober/folder",
    "size": 4096,
    "state": "directory",
    "uid": 1000
}
```

Escalado de privilegios

- Ansible usa los mecanismos de escalado de privilegios existentes en el SO de los *hosts* para ejecutar tareas usando permisos de *root* y/o los permisos de otros usuarios
- El parámetro ***--become*|*-b*** del comando ansible permite **activar** el escalado de privilegios
  - El parámetro *--become-method* permite especificar el mecanismo de escalado a utilizar (por defecto: *sudo*)
  - Mecanismos de escalado soportados:
    - *[sudo | su | pbrun | pmrun | pfexec | doas | dzdo | ksu | runas | machinectl]*
  - El parámetro *--become-user=USER* permite especificar el usuario con el que ejecutar los comandos (por defecto: *root*)
    - No implica -b
  - El parámetro *--ask-become-pass*|*-K* permite introducir el *password* en caso de ser necesario

Escalado de privilegios: ejemplo

- Intentamos instalar el servidor web nginx usando yum

```
[rober@oceania ~]$ ansible local -m yum -a "name=nginx state=present"
localhost | FAILED! => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false,
    "changes": {
        "installed": [
            "nginx"
        ]
    },
    "msg": "You need to be root to perform this command.\n",
    "rc": 1,
    "results": [
        "Loaded plugins: fastestmirror, langpacks\n"
    ]
}
```

Escalado de privilegios: ejemplo

- Activamos el escalado de privilegios (*-b*) usando *sudo*
  - En este ejemplo, el usuario con el que se conecta el comando ansible (*rober*) está configurado en el fichero */etc/sudoers* del SO para poder ejecutar cualquier comando sin introducir el *password*
  - En caso contrario, el uso de *--ask-become-pass*|*-K* permitiría introducir el *password*

```
[rober@oceania ~]$ ansible local -m yum -b -a "name=nginx state=present"
localhost | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": true,
    "changes": {
        "installed": [
            "nginx"
        ]
    },
    "msg": "",
    "rc": 0,
    "results": [
        "Loaded plugins: fastestmirror, langpacks\nLoading mirror speeds from cached
\n * extras: centos.uvigo.es\n * rpmfusion-free-updates: fr2.rpmfind.net\n * updates:
```

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*
    - Suelen ser específicos de un sistema/*shell*
    - Es (más/muy) complicado crear *scripts* idempotentes (mucha lógica adicional)
    - Sintaxis no 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* donde cada tarea invoca un módulo Ansible
    - 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 (y por tanto son versionables)
  - Un playbook se ejecuta con el comando **ansible-playbook**

Convirtiendo *shell scripts*

- Con Ansible es relativamente sencillo convertir *shell scripts* existentes en playbooks usando el módulo *command*
  - Facilita a los administradores la transición a Ansible
  - Pero no se aprovecha la potencia de usar módulos específicos para determinadas tareas lo que además disminuye la idempotencia

```
# Install Apache.
yum install --quiet -y httpd httpd-devel
# Copy configuration files.
cp httpd.conf /etc/httpd/conf/httpd.conf
cp httpd-vhosts.conf /etc/httpd/conf/httpd-vhosts.conf
# Start Apache and configure it to run at boot.
service httpd start
chkconfig httpd on
```

```
---
- hosts: all
  become: yes

  tasks:
        - command: yum install --quiet -y httpd httpd-devel
        - command: cp httpd.conf /etc/httpd/conf/httpd.conf
        - command: cp httpd-vhosts.conf /etc/httpd/conf/httpd-vhosts.conf
        - command: service httpd start
        - command: chkconfig httpd on
```

Convirtiendo *shell scripts*

- Playbook equivalente usando módulos de Ansible
  - Su idempotencia depende de los módulos utilizados

```
---
- hosts: all
  become: yes

  tasks:
        - name: Install Apache.
          yum: name={{ item }} state=present
          with_items:
                - httpd
                - httpd-devel
        - name: Copy configuration files.
          copy:
                src: "{{ item.src }}"
                dest: "{{ item.dest }}"
                owner: root
                group: root
                mode: 0644
          with_items:
                - src: "httpd.conf"
                  dest: "/etc/httpd/conf/httpd.conf"
                - src: "httpd-vhosts.conf"
                  dest: "/etc/httpd/conf/httpd-vhosts.conf"
        - name: Make sure Apache is started now and at boot.
          service: name=httpd state=started enabled=yes
```

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

```
---
- hosts: webservers
  remote_user: root

  tasks:
  - name: ensure apache is at the latest version
    yum:
      name: httpd
      state: latest

- hosts: databases
  remote_user: root

  tasks:
  - name: ensure postgresql is at the latest version
    yum:
      name: postgresql
      state: latest
  - name: ensure that postgresql is started
    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:
  - *--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: ejemplo


```
[rober@oceania ~]$ echo "hello world" > testfile
[rober@oceania ~]$ cat hello.yml
---
- hosts: all

  tasks:
    - name: Create a file '/tmp/testfile' with the content 'hello world'.
      copy:
        src: ./testfile
        dest: /tmp/testfile
```

Ejecutando un playbook: ejemplo


```
[rober@oceania ~]$ ansible-playbook hello.yml

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

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

TASK [Create a file '/tmp/testfile' with the content 'hello world'.] ***********************************************
changed: [localhost]

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

[rober@oceania ~]$ cat /tmp/testfile
hello world
[rober@oceania ~]$ ansible-playbook hello.yml

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

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

TASK [Create a file '/tmp/testfile' with the content 'hello world'.] ***********************************************
ok: [localhost]

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

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
  - 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" topics genéricos a los que responder de forma que una tarea puede invocarlo por topic en vez de por su nombre
- Ejemplo típico de uso:
  - Reiniciar un servicio cuando una tarea realiza un cambio en su configuración

```
[rober@oceania ~]$ cat handler.yml
---
- hosts: all

  tasks:
    - name: Enable Apache rewrite module
      apache2_module: name=rewrite state=present
      notify: restart apache

  handlers:
    - name: restart apache
      service: name=apache2 state=restarted
```

Variables

- Hay dos formas de declarar variables para ser usadas en un playbook
  - Por línea de comandos usando el parámetro *--extra-args* | *-e*
    - Formato foo=bar (siempre se interpretan como *strings*)
      - *ansible-playbook example.yml --extra-vars "foo=bar"*
    - 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 **Jinja 2**
  - Forma básica para sustituir una variable: **{{ variable }}**
  - Variable de tipo lista/array: **{{ foo[0] }}**
  - Variable de tipo diccionario/hash: **{{ foo['field1'] }}** y **{{ foo.field1 }}**
- Existen **variables especiales** (p.e., *facts*) cuyos nombres están reservados
  - https://docs.ansible.com/ansible/latest/reference_appendices/special_variables.html#special-variables

Variables

- Ejemplos

```yaml
---
- hosts: example
  vars:
    foo: bar
  tasks:
    # Prints "Variable 'foo' is set to bar".
    - debug: msg="Variable 'foo' is set to {{ foo }}"
```

```yaml
---
# Main playbook file.
- hosts: example
  vars_files:
    - vars.yml
  tasks:
    - debug: msg="Variable 'foo' is set to {{ foo }}"
```

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

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

```yaml
region:
  - northeast
  - southeast
  - midwest
```

```yaml
foo:
  field1: one
  field2: two
```

Facts

- Son variables especiales recopiladas automáticamente por Ansible en los *hosts* remotos durante la ejecución de un playbook
  - Proporcionan información útil sobre los *hosts* (p.e., dirección 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 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
  - Especialmente útil si no se usan y el número de *hosts* remotos es muy elevado
  - Se desactiva incluyendo la sección ***gather_facts: no***
- 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 *hostname*
  - {{ **ansible_facts**['nodename'] }}

Facts

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

```
[rober@oceania ansible]$ ansible localhost -m setup -a "gather_subset=min filter=ansible_distribution*"
localhost | SUCCESS => {
    "ansible_facts": {
        "ansible_distribution": "CentOS",
        "ansible_distribution_file_parsed": true,
        "ansible_distribution_file_path": "/etc/redhat-release",
        "ansible_distribution_file_variety": "RedHat",
        "ansible_distribution_major_version": "7",
        "ansible_distribution_release": "Core",
        "ansible_distribution_version": "7.9",
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false
}
```

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 sección ***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
- 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/user_guide/playbooks_conditionals.html#the-when-statement

```
- hosts: web_servers

  tasks:

     - shell: /usr/bin/foo
       register: foo_result
       ignore_errors: True

     - 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
  - La lista de valores se puede definir directamente en la propia tarea

```
- name: add user testuser1
  user:
    name: "testuser1"
    state: present
    groups: "wheel"

- name: add user testuser2
  user:
    name: "testuser2"
    state: present
    groups: "wheel"
```

```
- name: add several users
  user:
    name: "{{ item }}"
    state: present
    groups: "wheel"
  loop:
     - testuser1
     - testuser2
```

  - Dicha lista también se puede definir en la sección *vars* del 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 la ejecución de una tarea hasta que se cumpla una determinada condición
  - https://docs.ansible.com/ansible/latest/user_guide/playbooks_loops.html#retrying-a-task-until-a-condition-is-met

Loops

- Algunos módulos aceptan una lista de valores como parámetro
  - E.g. módulos para la instalación de paquetes (*package*, *yum*, *apt*, ...)
  - La documentación de un módulo especifica si es posible pasarle una lista como entrada de 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

```
- name: optimal yum
  yum:
    name: "{{  list_of_packages  }}"
    state: present

- name: non-optimal yum, slower and may cause issues with interdependencies
  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/user_guide/playbooks_loops.html#migrating-from-with-x-to-loop

Blocks

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

Bloque que contiene tres tareas

```yaml
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: yes
```

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

Organización de playbooks

- Es posible organizar los playbooks con el objetivo de maximizar su reutilización y mejorar su mantenimiento, además de evitar 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 y roles
  - *vars_files, include_vars*
  - *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 reutiliza las tareas

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

  - Reutilizando playbooks y roles:

Playbook que importa dos playbooks

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

Playbook que reutiliza dos roles (se especifican a nivel de "jugada")

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

Playbook que reutiliza un rol con *include_role* (en este caso se especifican 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 (e.g. tareas) pueden verse afectado por el resultado de la ejecución de tareas previas del playbook
  - Los *imports* se preprocesan cuando se parsea el playbook antes de su ejecución: reutilización **estática**
    - El contenido importado nunca se vería 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
- Más detalles:
  - https://docs.ansible.com/ansible/latest/user_guide/playbooks_reuse.html#comparing-includes-and-imports-dynamic-and-static-re-use

Contenidos

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

Roles

- Un rol permite **empaquetar** tareas, variables, *handlers*, ficheros, etc de forma que se puede reutilizar en múltiples playbooks e incluso compartir con otros usuarios mediante **Ansible Galaxy**
  - Permiten la división lógica de un playbook en múltiples componentes que son fácilmente reutilizables
  - Galaxy es un repositorio público de roles desarrollados por la comunidad
- Para ello, un rol se basa en una jerarquía y nomenclatura determinada de todos los ficheros y directorios que lo componen
  - La estructura de directorios y ficheros se puede crear de forma manual o bien utilizando el siguiente comando
    - *ansible-galaxy role init <role_name>*
- Los roles pueden almacenarse en una carpeta roles en el mismo directorio que un playbook o bien en una ruta global por defecto
  - Ruta global: */etc/ansible/roles*
  - La ruta es configurable en: */etc/ansible/ansible.cfg*

Estructura de un rol

- Ejecutamos: *ansible-galaxy role init myrol*

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

Estructura de un rol

- ***defaults:*** *main.yml*
  - Contiene la definición de las variables por defecto que son globales al rol (e.g. puerto de escucha por defecto de un servidor web)
- ***files***
  - Ficheros no modificables que son necesarios para el funcionamiento del rol, los cuáles serán copiados a los *hosts* remotos (e.g. código fuente de una aplicación web)
- ***templates***
  - Parecido a *files*, pero estos ficheros sí podrían modificarse (son plantillas)
  - Es posible pasar variables de configuración a un *template* para que éste lo aplique a los *hosts*
- ***handlers****: main.yml*
  - Contienes la definición de los manejadores
- ***meta****: main.yml*
  - Contiene metadatos del rol (p.e. autor, versión, licencia) y también permite definir las **dependencias** con otros roles que utiliza así como ***collections***

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

```yaml
# 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'

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

# 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 test se ejecutan en los *hosts* definidos en el fichero *inventory*
- ***vars**: main.yml*
  - Contiene la definición de otras variables utilizadas por el rol
  - Misma función que *defaults* pero con mayor prioridad
    - *defaults*: variables que muy probablemente necesiten ser sobrescritas
    - *vars*: variables cuyo valor por defecto se espera que no sea sobrescrito con frecuencia

Usando roles en un playbook

- La forma clásica de usar un rol en un playbook es con la sección ***roles***
  - Se especifican a nivel de "jugada"

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

- Desde la versión 2.4, es posible usar roles de forma *inline* en la sección *tasks* mediante ***import_role*** o ***include_role***
  - Se especifican a nivel de tarea
  - 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'"
```

Ejecución múltiple

- Aunque un rol se incluya múltiples veces, Ansible lo ejecutará una única vez si los parámetros definidos para el rol no han cambiado

```
---
- hosts: webservers
  roles:
    - foo
    - bar
    - foo
```

- Hay dos formas de ejecutar un rol más de una vez
  - Pasando parámetros diferentes en cada definición del rol (izquierda)
  - Añadiendo la opción ***allow_duplicates: true*** en ***meta/main.yml*** (derecha)

```
---
- hosts: webservers
  roles:
    - { role: foo, vars: { message: "first" } }
    - { role: foo, vars: { message: "second" } }
```

```
# playbook.yml
---
- hosts: webservers
  roles:
    - foo
    - foo

# roles/foo/meta/main.yml
---
allow_duplicates: true
```

Dependencias

- Ansible gestiona automáticamente las dependencias de un rol con otros roles según se definan en ***meta/main.yml***
- Los roles dependientes se ejecutan siempre antes que el rol que los incluye como dependencias
  - Las dependencias siguen las mismas reglas de ejecución múltiple especificadas anteriormente

```
# 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
```

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 (playbooks, roles, módulos, plugins) en un único paquete
  - Todos los módulos usados en los ejemplos previos pertenecen a la *collection* integrada de Ansible: ***ansible.builtin***
    - https://docs.ansible.com/ansible/latest/collections/ansible/builtin/index.html
- Una *collection* se define por su **FQCN: *namespace.collection***
  - FQCN = *Fully Qualified Collection Name*
  - El FQCN se utiliza referenciar contenido dentro la *collection*
    - Módulo ***package*** de la collection ***builtin*: *ansible.builtin.package***
  - Si no se indica *namespace*, se asume la *collection* por defecto (i.e., *builtin*)
- Uno de los principales motivos de su introducción fue el tremendo crecimiento de Ansible en los últimos años, provocando que el número de módulos fuera inmanejable para el equipo de desarrollo
  - Las *collections* permiten un desarrollo y mantenimiento de los módulos de forma independiente al ciclo de lanzamiento del núcleo de Ansible

Estructura de una *collection*

- De forma similar a un rol, una *collection* se basa en una estructura determinada de los ficheros y directorios que la componen
  - Se puede generar una plantilla mediante el siguiente comando
    - *ansible-galaxy collection init <collection_name>*
- El fichero *galaxy.yml* contiene los metadatos necesarios para construir, empaquetar y publicar una *collection*

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

Usando una *collection* en playbooks/roles

- Desde un playbook, se puede referenciar cualquier contenido de una *collection* usando su FQCN
  - Para mayor comodidad, la versión 2.8 introduce una nueva sección en los playbooks: ***collections***
    - Permite simplificar el nombrado de los módulos

```
- hosts: all
  tasks:
    - import_role:
        name: my_namespace.my_collection.role1

    - my_namespace.mycollection.mymodule:
        option1: value
```

```
- hosts: all
  collections:
   - my_namespace.my_collection
  tasks:
    - import_role:
        name: role1

    - mymodule:
        option1: value
```

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

```
# 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 creados 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
  - 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

- Con la introducción de las *collections*, surge Ansible Automation Hub
  - https://www.ansible.com/products/automation-hub
- Se trata de un servicio que ofrece, entre otras cosas, un repositorio donde encontrar *collections* y roles Ansible
  - El contenido Ansible disponible en Automation Hub está **certificado** y **soportado** por Red Hat, Inc.
  - El acceso a 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 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 un rol por autor

```
[rober@oceania ~]$ ansible-galaxy role search --author geerlingguy nginx

Found 12 roles matching your search:

 Name                          Description
 ----                          -----------
 geerlingguy.certbot           Installs and configures Certbot (for Let's Encrypt).
 geerlingguy.collectd-signalfx SignalFx Collectd installation for Linux.
 geerlingguy.drupal            Deploy or install Drupal on your servers.
 geerlingguy.fathom            Fathom web analytics
 geerlingguy.gitlab            GitLab Git web interface
 geerlingguy.htpasswd          htpasswd installation and helper role for Linux servers.
 geerlingguy.munin             Munin monitoring server for RedHat/CentOS or Debian/Ubuntu.
 geerlingguy.nginx             Nginx installation for Linux, FreeBSD and OpenBSD.
 geerlingguy.passenger         Passenger installation for Linux/UNIX.
 geerlingguy.php               PHP for RedHat/CentOS/Fedora/Debian/Ubuntu.
 geerlingguy.pimpmylog         Pimp my Log installation for Linux
 geerlingguy.varnish           Varnish for Linux.
```

Gestión de roles: ejemplo

- Instalación de un rol para Java
  - https://galaxy.ansible.com/geerlingguy/java
  - https://github.com/geerlingguy/ansible-role-java

```
[rober@oceania ~]$ ansible-galaxy role install geerlingguy.java
- downloading role 'java', owned by geerlingguy
- downloading role from https://github.com/geerlingguy/ansible-role-java/archive/1.9.7.tar.gz
- extracting geerlingguy.java to /home/rober/.ansible/roles/geerlingguy.java
- geerlingguy.java (1.9.7) was installed successfully
[rober@oceania ~]$ ls /home/rober/.ansible/roles/geerlingguy.java
defaults  LICENSE  meta  molecule  README.md  tasks  templates  vars
[rober@oceania ~]$ ansible-galaxy role list
# /home/rober/.ansible/roles
- geerlingguy.java, 1.9.7
```

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

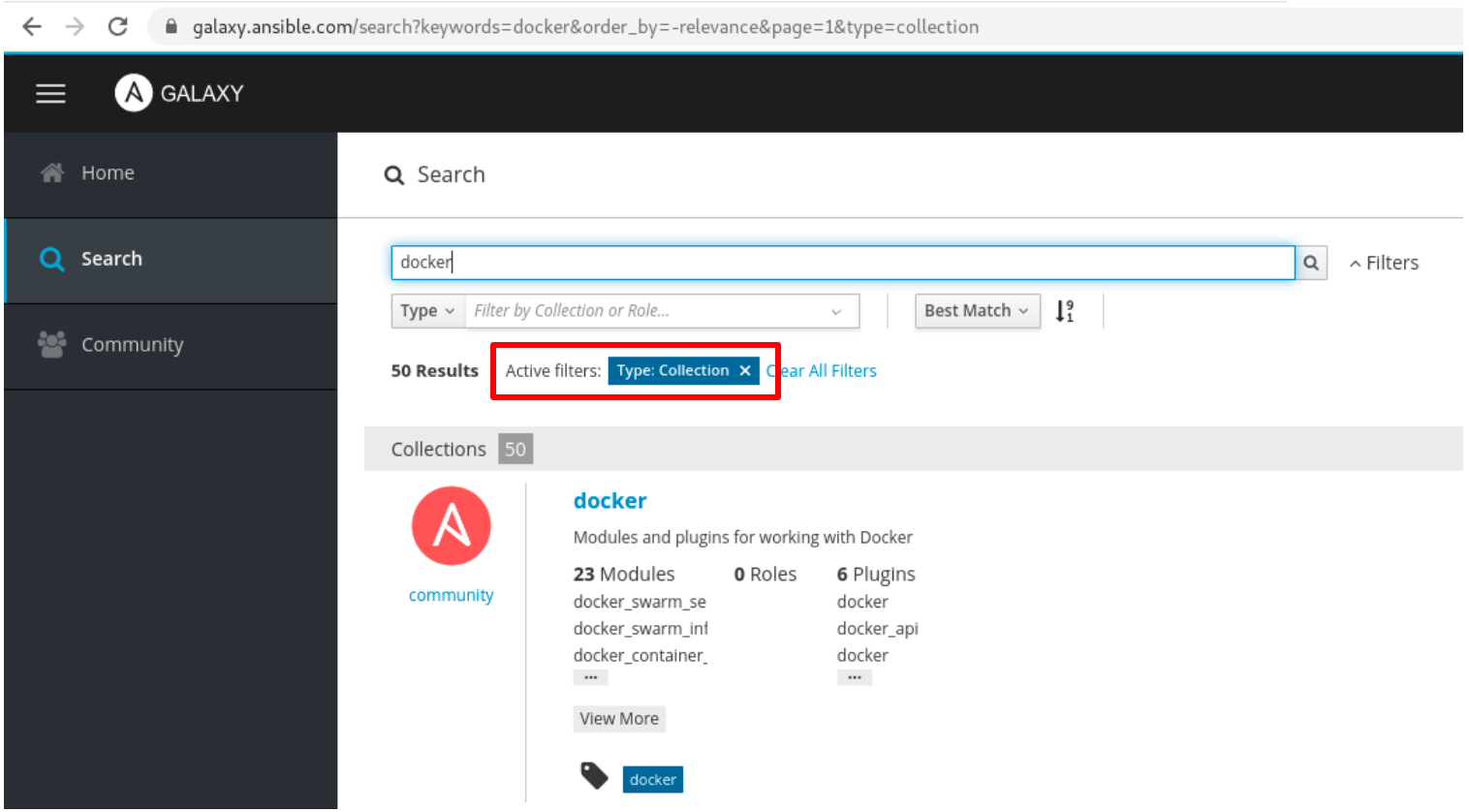
```
[rober@oceania ~]$ ansible-galaxy role install geerlingguy.java -p ./roles
- downloading role 'java', owned by geerlingguy
- downloading role from https://github.com/geerlingguy/ansible-role-java/archive/1.9.7.tar.gz
- extracting geerlingguy.java to /home/rober/roles/geerlingguy.java
- geerlingguy.java (1.9.7) was installed successfully
[rober@oceania ~]$ ls roles/geerlingguy.java/
defaults  LICENSE  meta  molecule  README.md  tasks  templates  vars
[rober@oceania ~]$ ansible-galaxy role list -p roles/
# /home/rober/roles
- geerlingguy.java, 1.9.7
# /home/rober/.ansible/roles
- geerlingguy.java, 1.9.7
```

Gestión de *collections*

- Buscar una *collection*
  - Solo desde el sitio web de Galaxy filtrando por *collection*
- 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 PHP
  - https://galaxy.ansible.com/geerlingguy/php_roles
  - https://github.com/geerlingguy/ansible-collection-php_roles

```
[rober@oceania ~]$ ansible-galaxy collection install geerlingguy.php_roles
Process install dependency map
Starting collection install process
Installing 'geerlingguy.php_roles:0.9.5' to '/home/rober/.ansible/collections/ansible_collections/geerlingguy/php_roles'
[rober@oceania ~]$ ls /home/rober/.ansible/collections/ansible_collections/geerlingguy/php_roles
FILES.json  MANIFEST.json  README.md  roles  tests
```

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

```
[rober@oceania ~]$ ansible-galaxy collection install geerlingguy.php_roles -p ./collections
[WARNING]: The specified collections path '/home/rober/collections' is not part of the configured Ansible collections paths
'/home/rober/.ansible/collections:/usr/share/ansible/collections'. The installed collection won't be picked up in an Ansible run.

Process install dependency map
Starting collection install process
Installing 'geerlingguy.php_roles:0.9.5' to '/home/rober/collections/ansible_collections/geerlingguy/php_roles'
```

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*
  - En versiones Ansible <= 2.9, deben instalarse 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.
  - src: geerlingguy.java
    version: 1.9.6

collections:
  # Install a collection from Ansible Galaxy.
  - name: geerlingguy.php_roles
    version: 0.9.3
    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 una VM Vagrant usando Ansible Playbooks
  - ***ansible***
    - El comando *ansible-playbook* se ejecuta en el equipo anfitrión (***host***)
    - https://www.vagrantup.com/docs/provisioning/ansible.html
  - ***ansible_local***
    - El comando *ansible-playbook* se ejecuta en la VM (***guest***)
    - https://www.vagrantup.com/docs/provisioning/ansible_local.html
- El fichero *inventory* es generado automáticamente por Vagrant
  - Es posible usar un fichero específico mediante la opción *inventory_path*
    - https://www.vagrantup.com/docs/provisioning/ansible_common.html

Provisioner *ansible*

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

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

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

Provisioner *ansible_local*

- Es necesario instalar Ansible en la VM
  - Por defecto, Vagrant intenta instalar Ansible automáticamente
  - https://www.vagrantup.com/docs/provisioning/ansible_local.html#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 proporciona dos *provisioners* distintos que pueden usar para aprovisionar las imágenes máquina
  - ***ansible***
    - El comando *ansible-playbook* se ejecuta en el ***host***
    - https://www.packer.io/plugins/provisioners/ansible/ansible
    - Es necesario instalar Ansible en el **host** que ejecuta Vagrant
  - ***ansible_local***
    - El comando *ansible-playbook* se ejecuta en la VM (***guest***)
    - https://www.packer.io/plugins/provisioners/ansible/ansible-local
    - En este caso, Packer no instala automáticamente Ansible en la VM así que debe instalarse previamente usando, por ejemplo, un *provisioner shell*

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 y *plugins* que permiten gestionar contenedores Docker mediante playbooks
  - https://docs.ansible.com/ansible/latest/collections/community/docker/docsite/scenario_guide.html
  - https://galaxy.ansible.com/community/docker
- Módulos más relevantes
  - *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*: orquestación de contenedores mediante ficheros compose
  - *docker_swarm_service*: gestión de servicios Swarm
- Además, existen otras herramientas como ansible-bender, que permite generar imágenes Docker a partir de un playbook
  - https://github.com/ansible-community/ansible-bender

Ejemplo

- Ejecutaremos un playbook que se encarga 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

```
[rober@oceania docker]$ cat Dockerfile
FROM debian
RUN apt-get update
RUN apt-get install -y nginx
RUN echo 'Nginx is running in your container' > /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***

Ejemplo

- Contenido del playbook: main.yml

```
[rober@oceania docker]$ cat main.yml
---
- hosts: localhost

  tasks:
    - name: Ensure Docker image is built from the Dockerfile.
      docker_image:
        name: web
        source: build
        build:
          path: /home/rober/ansible/docker
          pull: yes
        state: present

    - name: Ensure the container is running.
      docker_container:
        image: web:latest
        name: docker
        state: started

    - name: Check if container is running.
      shell: docker ps
      register: ps

    - debug: var=ps.stdout
```

Ejemplo

- Ejecución del playbook

```
[rober@oceania docker]$ ansible-playbook main.yml

PLAY [localhost] ***************************************************************************************************************

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

TASK [Ensure Docker image is built from the Dockerfile.] ***********************************************************************
changed: [localhost]

TASK [Ensure the container is running.] ****************************************************************************************
changed: [localhost]

TASK [Check if container is running.] ******************************************************************************************
changed: [localhost]

TASK [debug] *******************************************************************************************************************
ok: [localhost] => {
    "ps.stdout": "CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS                  PORTS
 web:latest          \"nginx -g 'daemon of…\"   1 second ago        Up Less than a second   80/tcp              docker"
}
```

Ejemplo

- Accedemos al servidor web

```
[rober@oceania docker]$ docker image ls web
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
web                 latest              5997477afcde        6 minutes ago       196MB
[rober@oceania docker]$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS               NAMES
a0ff82cdce40        web:latest          "nginx -g 'daemon of…"   6 minutes ago       Up 6 minutes        80/tcp              docker
[rober@oceania docker]$ docker inspect --format "{{.NetworkSettings.IPAddress}}" docker
172.17.0.2
[rober@oceania docker]$ curl 172.17.0.2
Nginx is running in your container
```