APUNTES ANSIBLE

Manuel Vergara – 2023

Índice

Este documento contiene los apuntes tomados en el curso «Curso de Ansible: Automatización» impartido por «Oforte» en junio de 2023. El curso udemy consta de 10 horas aproximadamente de vídeo-tutoriales. Las prácticas aquí contenidas tuvieron una duración de alrededor de unas 40 horas.

Los apuntes no fueron pensados para compartirlos, por ello pueden tener lagunas de información o contenido adicional respecto al curso, ya que se redactaron para recordar procedimientos y conceptos que el autor creyó relevantes. Teniendo un documento, a mi parecer, tan completo y entendiendo que el conocimiento debe ser libre se decidió compartirlo.

Si te parece útil este documento puedes agradecerlo a través de las vías de contacto de la web https://vergaracarmona.es

Recuerda,

> *"Quien se corta su propia leña se calienta dos veces"*

TEMA 1 - Introducción

Ansible es una herramienta gratuira y de código abierto para la automatización de infraestructuras (Servidores (Linux o Windows), cloud (AWS. GCS, Openstack, Vmware, etc), dispositivos (Switches o routers)) y aplicaciones (Instalación, configuración y administración).

Herramientas similares: chef, puppet o salt

Ventajas:

- No requiere un agente, se conecta mediante ssh desde el servidor ansible.
- Sintaxis simple y fácil de aprender. Yaml
- Seguro y fácil de mantener.
- Rendimiento.
- No requiere saber programar.

Desventajas:

- No es potente como administrado de configuraciones. No tiene un versionado de estados.
- Requiere en grandes entornos configuraciones avanzadas.
- Tardía solución de errores (bugs). Tardan meses en resolver issues

Doc: https://docs.ansible.com/ansible/latest/

1.1. - Instalación

https://docs.ansible.com/ansible/latest/installation_guide/intro_installation.html

Install in ubuntu 22.04: https://www.digitalocean.com/community/tutorials/how-to-install-and-configure-ansible-on-ubuntu-22-04

1.2. - Primeros pasos

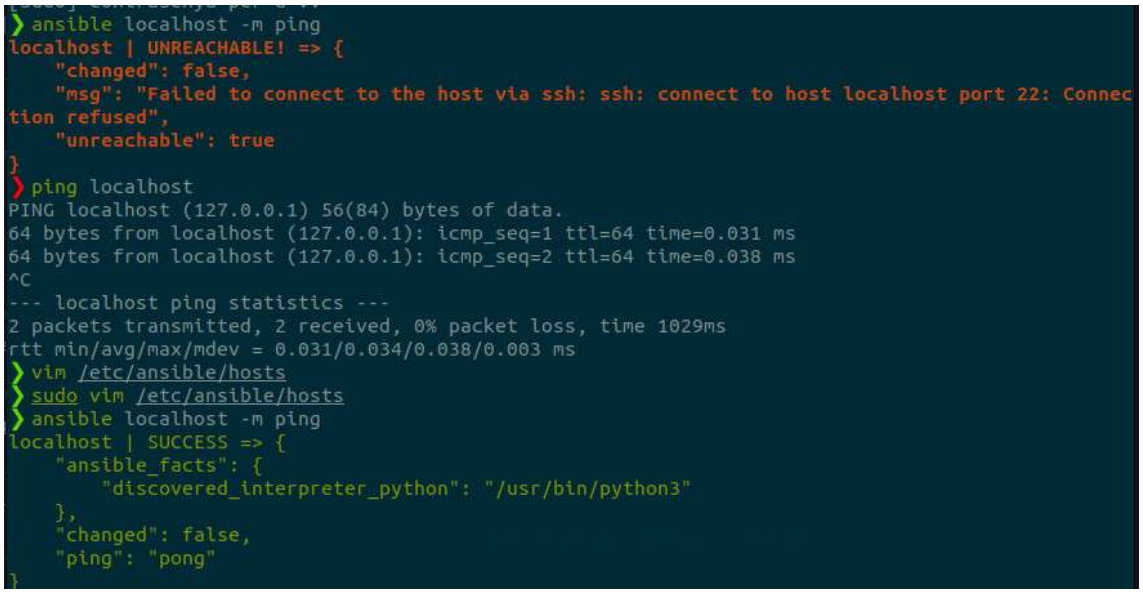
Añadimos en el fichero */etc/*ansible/hosts los servidores. Empezamos con `localhost`

La conexión por defecto es ssh. Para conexión local, detrás de localhost añadimos: ansible_connection=local

Comprobar conexión:

ansible [servidor(es):grupo(s)] -m ping

Primero sin configurar y luego con la configuración



Ejecutar comando:

ansible [servidor(es):grupo(s)] -a "comando"

Por ejemplo:

ansible localhost -a "hostname"

ansible localhost -a "uptime"

Se puede especificar un puerto si el servidor ha cambiado el 22 en el fichero de configuración:

192.168.1.2:2222

Y si necesitamos especificar un usuario lo podemos hacer con la opción -u:

ansible 192.168.1.2 -u usuario1 -m ping

Además, debemos tener configurado el ssh en el servidor. Se crea el par de llaves y se copia la clave pública en el fichero del servidor .ssh/authorized_keys.

Se puede indicar a todos los servidores configurados en el fichero con all:

ansible all -u usuario1 -m ping

Se le puede añadir el usuario directamente en el fichero de configuración:

192.168.1.2:2222 ansible_user=usuario1

De esta manera no necesitaremos especificarlo.

Para conectarse a un usuario que necesite utilizar sudo: -u usuario1 --become

Por ejemplo, para crear un usuario sería así:

ansible 192.168.1.2 -u usuario1 -m user -a "name=oforte state=present"

Pero esto no funcionará, porque necesita permisos root para crear un usuario. Pero con --become, si el usuario tiene permisos sudo podrá hacerlo.

1.3. - Inventario estático

- Formato INI nombre clave=vebr
- Los grupos se especifican entre corchetes. [nombre]
- Un elemento (servidor, dispositivo) puede estar en más de un grupo.
- Un grupo puede tener subgrupos, especificando un grupo padre que contendrá los children: [grupo:children]

[grupo:children]
grupo1
grupo2

- Se pueden definir parámetros (variables) a un grupo: [grupo:vars]

[grupo:vars]
ansible_become=True # También se puede poner detrás del hosts, que tendrá preferencia sobre esto

- Es possible separar las variables en ficheros:

/etc/ansible/group-vars/grupo

/etc/ansible/host-vars/servidor

- Patrones. Es posible utilizar regex como las siguientes sintaxis:

web[1:5].invento.net

web[a:1].invento.net

- Escoger inventario. Con la opción -i podemos especificar la path del fichero.
- Parámetros (variables).
  - ansible-connection:
    - ssh
      - ansible_host
      - ansible_port
      - ansible_user
      - ansible_ssh_private_key_file
    - local
  - ansible-become
    - True
      - ansible_become_method (su | sudo)
      - ansible_become_user (root)
    - False
- Podemos asignar alias a las ips en el fichero de config. Funcionarán como DNS para ansible. Por ejemplo:

debian1 ansible_host=192.168.1.2 ansible_port=2222 ansible_user=usuario1

1.4. - Inventario Dinámico

Es posible utilizar un inventario dinámico a partir de:

- Proveedor Cloud
  - AWS
  - GCP
  - DigitalOcean
- Entornos propios:
  - Openstack
  - Ovist
  - OpenShift
  - Zabbix

https://docs.ansible.com/ansible/latest/plugins/inventory.html#plugin-list

Se supone según el curso que existen unos scripts python de inventory para aplicar con la opción -i y facilitar la conexión con clouds públicos, pero no existe la carpeta en el repositorio oficial de ansible/ansible.

1.5. - Adhoc

El modo adhoc nos permite realizar acciones de forma simple y comprobar la conexión a los elementos del inventario.

La sintaxis:

ansible [opciones] (servidores | grupos | all) [-m módulo] [-a argumentos]

Por defecto, -m usa el **módulo command.**

Por defecto, cuando se pone all o varios servidores y se aplica un comando, lo hará de 5 en 5.

Opciones:

- Se puede especificar filtros con –limit, por ejemplo:

ansible –limit curso01,curso02,curso03 -i ./gce.py all -u alberto -a "cat /etc/debian_version"

- --user para indicar el usuario. También -u
- --become para cambiar a usuario root. También -b
- -f es para indicar un int las tareas simultaneas a realizar.
- --list-hosts para listar los servidores: ansible all --list-host
- -C para realizar un dry-run
- -v para verbose. Mucho más verbose -vvv

El **módulo setup** sirve para obtener información de un servidor o denominados facts (Resume de parámetros)

ansible all -u alberto -m setup | less

El **módulo copy** sirve para copiar al servidores. Necesita un argumento que indique origen y destino de lo que vamos a copiar:

-a "src=/etc/hosts dest==/etc/hosts"

También existen los **módulos yum o apt** para realizar instalaciones. Necesita los argumentos:

-a "name=vim state=present | absent"

present para instalar y absent para desinstalar.

1.6. - Configuración

La configuración global contiene opciones diversas del funcionamiento de Ansible. Contiene:

- Valores por defecto generales
- Configuración de escalar permisos
- Opciones de openSSH
- Opciones SELinux
- Configuraciones para Ansible Galaxy

Orden de cerca de la configuración:

1. Variable de entorno: ANSIBLE.CONFIG
2. Ficheros: ansible.cfg y .ansible.cfg
3. Fichero: /etc/ansible/ansible.cfg # Configuración global

Tienen un ejemplo en el repo:
https://github.com/ansible/ansible/blob/stable-2.9/examples/ansible.cfg

1.7. - Windows

Para administrar servidores Windows se debe configurar el servidor Ansible y el Windows server.

- Instalar la librería python pywinrm
- Definir connection al valor winrm
- Necesitamos powershell 3.0 o superior en el server w
- Habilitar control remoto en el server w. Ansible tiene un script: ConfigureRemotingForAnsible.ps1
- Habilitar el puerto 5186

Más info: https://docs.ansible.com/ansible/latest/os_guide/windows_usage.html

Para comprobar la conexión a Windows se utiliza un módulo de ping concreto:

ansible windowmal -c winrm -u usuario1 -k -m win_ping

Añadimos también la opción para que nos pida el password: -k

Si necesitamos ignorar el certificado utilizaremos la opción:
ansible_winrm_server_cert_validation=ignore

O la opción -e para asignar la variable de entorno:

ansible windowmal -c winrm -e ansible_winrm_server_cert_validation=ignore -u usuario1 -k -m win_ping

1.8. - Combinar inventarios

Es posible utilizar más de un inventario

Podemos combinar inventarios estáticos y dinámicos.

Dentro de un directorio añadimos los inventarios

Con la opción -i especificamos el path

Es posible hacer referencia desde un inventario al otro.

También es posible tener los directorios group.vars y host.vars

Recomendado usar --list-hosts

TEMA 2 - Playbooks

Un playbook contiene una lista de "jugadas" (tareas) a realizar en una lista de servidores especificados. Además incluye configuraciones y variables entre otros elementos.

El formato es YAML (Ain't Markup Language), muy simple de escribir y leer.

Ejemplo:

```
---
- name: Mi primer playbook
  hosts: all
  remote_user: usuario1
  become: True
  become_method: sudo/su/pbrun/ksu
  check_mode: True                    #modo para chequear lo que va a aplicar
  tasks:
    - name: Copiar fichero hosts
      copy: src=/etc/hosts dest=/etc/hosts
    - name: Actualizar
      service:
```

La extensión debe ser yaml. Para invocar el fichero:

```
ansible-playbook [-i inventario] [opciones] playbook.yaml
```

La configuración puede ir a nivel de playbook o a nivel de task.

2.1. - Esenciales

- hosts: Lista de servidores a administrar. Es posible especificar grupos o servidores individuales. El separador es : y es posible utilizar los siguientes carácteres: & que cumplan ambos grupos. ! para la negación. Ejemplos:

hosts: servweb

hosts: servweb1 :& servweb2 # Los servidores de los dos grupos.

hosts: servweb :& madrid # Servidores que estén en ambos grupos.

hosts: servweb :! madrid # Servidores que estén en servweb y no estén en madrid.

Antes de aplicar, podemos usar opciones para que nos indique información:

- --list-hosts nos indicará a qué servidores se va a aplicar.
- --list-tags nos indicará las tags
- --list-tasks nos indicará las tasks

También podemos filtrar servidores con –limit:

--limit 'serv1,serv2'

Se puede filtrar por grupos igual que en el hosts del playbook.

--limit servweb :& madrid

2.2. - Ansible playbooks

Más opciones:

- -i inventario → Especifica un fichero, un script o un directorio.
- --syntax_check → Comprobar la sintaxis del fichero
- --step → Para que nos pregunte en cada tarea si queremos realizar
- --start-at-task → Para inicial desde una tarea en concreto.
- --forks | -f → Limitar las tareas en paralelo
- -v (vvv) → Verbose

2.3. - Variables

Una variable contiene un valor modificable para dinamizar las tareas. Algunos ejemplos:

- Definir el puerto de una aplicación
- Indicar un usuario.
- Establecer una ruta de un fichero

Las variables pueden ser definidas en el servidor (Facts), en playbook (O elemntos varios) y en la línea de comandos con la opción -e

Pueden ser usadas en tareas, plantillas (Ficheros estáticos con lógica) y en otros como las condiciones, bucles, etc.

Ejemplo:

En un fichero hosts.j2 especificamos el nombre de las variables:

{{ miip }} {{ ansible_hostname }} {{ ansible_fqdn }}

Y en el playbook variables.yaml le damos la clave:valor.

```
Vars:
  miip:192.168.150.150
```

ansible_hostname y ansible_fqdn serán variables dinámicas que cogerá del sistema

Si buscamos con setup veremos que los dinámicos constan en la configuración:

ansible localhost -m setup | grep -e "miip\|ansible_hostname\|ansible_fqdn"

Podemos sobreescribir en el comando el valor de la variable estática con la opción -e:

ansible-playbook -i hosts -e miip="4.3.2.1" variable.yaml

2.4. - Sintaxis

La sintaxis para YAML simplifica la definición de playbooks. Además ofrece diferentes opciones para mejorar la legibilidad.

Por ejemplo

```
- name: Copiar ficheros
  copy: src=/etc/hosts dest=/etc/hosts owner=root group: root mode: 0644
```

Puede ser también

```
- name: Copiar ficheros
  copy:
    src: /etc/hosts
    dest: /etc/hosts
    owner: root
    group: root
    mode: 0644
```

Las listas también pueden tener ambos formatos:

```
instalar:
  - apache2
  - mariadb
  - php8
instalar: ["apache2", "mariadb", "php8"]
```

Los diccionarios con clave y valor:

```
    midic:
      editor: vim
      visual: tree
      comprimir: zip
      descomprimir: unzip
```

Se puede llamar con milista_larga.editor por ejemplo.

Y si queremos asignar un texto largo usaremos la barra vertical:

```
texto_largo: |
```

Y para una línea larga en un formato más legible mayor que:

```
linea_larga: >
```

El módulo dubg permite mostrar un texto o el valor de una variable en pantalla:

```
  tasks:
    - name: Crear fichero pruebas sintaxis
      template:
        src: 03_1_pruebas_sintax.j2
        dest: /tmp/pruebas_sintax.j2

    - name: Mostrar variable milista
      debug: var=milista
```

```
  - name: Mostrar variable midic
    debug: var=midic

  - name: Mostrar variable valor de midic
    debug: var=midic.comprimir

  - name: Mostrar variable texto_largo
    debug: var=texto_largo

  - name: Mostrar variable linea_larga
    debug: var=linea_larga
```

2.5. - Handlers

Un handler es una tarea que sólo se ejecutará en caso de que otra tarea la llame. El uso más común es reiniciar un servicio cuando se cambie la configuración.

Ejemplo:

```
tasks:
  - name: Configurar sshd.config
    copy:
      src: sshd.config
      dest: /etc/sshd.config
    notify: reiniciar_sshd

handlers:
  - name: reiniciar_sshd
    service:
      name: ssh
      state: restarted
```

Los handlers se ejecutaran al final del playbook. Para evitar que los handlers se adelanten a las tareas.

2.6. - Include y Roles

Es possible dividir un playbook en distintas partes para facilitar la edición y el tratado.

Con include especificamos otro fichero que contiene el playbook o una tarea.

Ejemplo:

```
- name: Primer play
  hosts: serviweb
  tasks:
    - name: tarea en este fichero
      copy: src=fichero dest=fichero
    - include: otra_tarea.yaml  ##### DEPRECADO!!!


- name: tarea en fichero
  service: name=nombre state=started
```

include is deprecated. Ahora se usa estas formas más específicas:

Re-using files and roles🔗

Ansible offers two ways to re-use files and roles in a playbook: dynamic and static.

- For dynamic re-use, add an `include_*` task in the tasks section of a play:
  - include_role
  - include_tasks
  - include_vars
- For static re-use, add an `import_*` task in the tasks section of a play:
  - import_role
  - import_tasks

Con los roles podemos crear una estructura de ficheros y directorios para separar los elementos, permite ser reusados facilmente y es possible descargar roles predefinidos.

Ansible galaxy es un servicio en línea que actúa como un repositorio de roles de Ansible. Un rol de Ansible es una unidad de organización que incluye tareas, variables, archivos y otros componentes necesarios para realizar tareas específicas en un sistema.

Roles/
  nombre/
   files/
   templates/
   tasks/main.yml → include....
   handlers/main.yml
   vars/main.yml
   defaults/main.yml
   meta/main.yml

Para hacer referencia a los roles:

roles:
  - rol1
  - rol2

Y para pasarle variables:

roles:
  - role: rol1, clave: valor

La extensión de archivo `.j2` se asocia comúnmente con plantillas de Jinja2. Jinja2 es un motor de plantillas para Python que permite generar dinámicamente texto o código basado en una plantilla y datos proporcionados. En el contexto de Ansible, las plantillas Jinja2 se utilizan para generar archivos de configuración o scripts dinámicamente durante la ejecución del playbook.

2.7. - Templates

Dentro de una plantilla (template) se pueden especificar instrucciones:

- Expresiones: {{ variable }} → {{ ansible-fqdn }}
- Control: {% ... %} →
  - Condición:
    {% if ansible_distribution == "Debian" %}
    Mostrar sólo en sistemas Debian
    {% endif %}
- Bucle:
  {% for usuario in lista_usuarios %}
  usuario
  {% endfor %}
- Comentarios: {# comentarios varios #} Para comentar nuestras plantillas, no se copiará al servidor.

2.8. - Prioridad variables

El orden (de menor a mayor) de prioridad es el siguiente:

- Directorio «Default» definidas en un rol.
- Variables de grupo (inventario → group.vars/all → group.vars/<grupos>).
- Variables de servidor (inventario → host.vars/<servidor>).
- «Facts» del servidor.
- Variables del play (-→ vars.prompt → vars.files)
- Variables del role (Definidas en /roles/rol/vars/main.yaml)
- Variables de bloque → Variables de tareas
- Parámetros de role → include-params → include-vars
- set-facts /registered-vars
- extra vars (En la línea de comandos)

2.9. - Condiciones

Es possible condicionar la ejecución de una tarea, la inclusión de un fichero o el uso de un rol usando la expresión when.

Ejemplo a nivel de tarea:

```
-name: instalar apache2
 apt:
   name: apache2
   state: latest
 when: ansible_distribution == "Debian"


-name: instalar httpd
 yum:
   name: httpd
   state: latest
 when: ansible_os_family == "RedHat"
```

Ejemplos a nivel de expresión include:

```
- name: Instalar apache2
  include: instalar-apache2.yaml
  when: ansible_distribution == "Debian"
```

```
- name: Instalar httpd
  include: instalar-apache2.yaml
  when: ansible_os_family == "RedHat"
```

También podemos crear distintos roles y aplicarlos cuando es el apropiado:

```
roles:
  - {role: apache2, when: ansible_distribution == "Debian"}
```

2.10. - Bucles

Hemos visto como recorrer una lista dentro de una plantilla, ahora vamos a recorrer una lista dentro de una tarea.

Para las listas y los diccionarios, utilizaremos la expresión with_items

Ejemplo instalando paquetes:

```
- name: Instalar software necesario
  apt:
     name: "{{ item }}"
     state: latest
  with_items:
     - mariadb
     - php5
     - phpmyadmin
```

Ejemplo creando usuarios:

```
- name: Crear usuarios
  user:
    name: "{{ item nombre }}"
    state: present
    groups: "{{ item grupo }}"
  with_items:
     - {nombre: usuario1, grupo: www-data}
     - {nombre: usuario2, grupo: www-data}
```

Es posible especificar una variable en with_items: "{{ variable }}"

2.11. - Register

La expresión register nos permitye guardar en una variable el resultado de la acción realizada por un módulo en una tarea.

Ejemplo:

```
- name: Ejecutar comando
  command: uptime
  register: salida

- name: Mostrar salida
  debug:
    var: salida
```

Para ver solo el argumento stdout se le indica así: salida.stdout

Además, tenemos los valores:


```
ok: [ubuntu2310] => {
    "resultado_uptime": {
        "changed": true,
        "cmd": [
            "uptime",
            "-p"
        ],
        "delta": "0:00:00.004454",
        "end": "2023-11-20 19:46:04.071926",
        "failed": false,
        "msg": "",
        "rc": 0,
        "start": "2023-11-20 19:46:04.067472",
        "stderr": "",
        "stderr_lines": [],
        "stdout": "up 1 hour, 14 minutes",
        "stdout_lines": [
            "up 1 hour, 14 minutes"
        ]
    }
}
```


stdout_lines separa el resultado por líneas si el output tiene saltos

2.12. - Ignore errors

La expresión ignore_errors permite ignorar una tarea marcada como error y continuará con el resto de tareas.

Ejemplo:

```
- name: comprobar si fichero existe
  command: ls /noexiste.conf
  register: existe
  ignore_errors: true
```

La expresión ignore_errors se puede especificar a nivel de playbook pero ¡Cuidado! Esto hará que todas las tareas darán ok, no se podrá debuguear los errores.

Es posible utilizar la condición:
not existed.failed
para realizar una tarea independiente.

También se puede utilizar así:
existed.failed !=false

2.13. - Failed When

Las expresiones failed_when y changed_when permiten especificar las condiciones para marcar una tarea como fallida o cambiada.

En el caso de un comando será marcado como error si el return code (rc) es distinto a 0.

$?

Ejemplo cambio a fallida:

```
- name: Ejecutar comando
  command: ip a
  register: salida
  failed_when: "'eth2' not in salida.stdout_lines"              # Marca como error si no está
```

Ejemplo cambio a cambiada:

```
- name: Ejecutar comando
  command: uptime
  changed_when: False                 # Marca como cambiada
```

TEMA 3 - Módulos

Ansible está compuesto de una multitud de módulos. Estos pueden administrar sistemas, dispositivos, usuarios y mucho más. Cada tarea en un playbook está asociada a un módulo. Los argumentos pueden ser obligatorios u opcionales y además suelen tener un valor por defecto. Se separan en las siguientes categorías:

- Cloud
- Clustering
- Commands
- Crypto
- Database
- Files
- Identity
- Inventory (servidores, proveedores..)
- messaging
- monitoring
- network
- Notification
- Packaging
- Remote management
- Source control
- Storage
- System
- Utilities
- Web infrasestructure
- Windows

Podemos utilizar el comando ansible-doc para listar (-l) todos los módulos disponibles en el equipo. Podemos ver ejemplos añadiendo el módulo:
ansible-doc template

Con -t podemos buscar el tipo:
ansible-doc -t {become,cache,callback,cliconf,connection,httpapi,inventory,lookup,netconf,shell,vars,module,strategy,test,filter,role,keyword}

3.1. - Módulo Ficheros Open SSL

Los modulos de tratado de fichero permiten trabajar con ficheros, plantillas y directorios.

Listado:

- acl – Establece y obtiene información de listas de control de acceso ACL
- archive – Crea un fichero comprimido a partir de una lista de fichero o estructura de directorios
- assemble - Crea un fichero de configuración desde distintos fragmentos.
- blockinfile – Inserta/actualiza/elimina un bloque de texto de un fichero.
- copy – Copia ficheros a ubicaciones remotas
- fetch – Obtiene ficheros de un nodo remoto a local
- file – Establece atributos a ficheros como permisos, propietarios, grupo, etc
- find – Devuelve una lista de ficheros a partir de un patrón
- inifile – Gestionar ficheros de tipo ini
- iso.extract – Extrae ficheros de una imagen ISO
- lineinfile – Asegura que una línia está en un fichero, a reemplaza una línia usando regex
- patch – Aplica parches utilizando GNU patch
- replace – Reemplaza las coincidencias de un texto especificado por otro indicado.
- stat – Obtiene información de ficheros o sistemas de ficheros
- synchronize – Módulo para sincronizar utilizando rsync
- tempfile – Crear ficheros o directorios temporales
- template – Para copiar y procesar una plantilla a un nodo remoto.
- unarchive – Extrae fichero (opcionalmente después de copiarlo)
- xattr – Establece/obtiene atributos extendidos.

Los módulos para openSSL son los siguientes:

- openssl_privatekey – Generar claves privadas de OpenSSL
- openssl_publickey – Generar claves públicas de OpenSSL

3.1.1. - copy

Opciones requeridas:

- dest=/directorio/fichero

Opciones opcionales:

- backup=yes/no
- content="contenido"
- force=yes/no
- owner=usuario
- group=grupo
- mode=modo
- src=/directorio/fichero

Ejemplos:

```
- name: Copiar configuración
  copy:
    src: apache2.conf
    dest: /etc/apache2/apache2.conf
    owner: www-data
    group: www-data


   - name: Crear fichero con contenido especificado
    copy:
      content: "Hola mundo"
      dest: /tmp/hola.txt
```

3.1.2. - template

Opciones requeridas:

- dest=/directorio/fichero
- src=/directorio/fichero

Opciones opcionales:

- backup=yes/no
- force=yes/no
- owner=usuario
- group=grupo
- mode=modo

Ejemplo:

```
- name: Copiar configuración
  template:
    src: apache2.conf
    dest: /etc/apache2/apache2.conf
    owner: www-data
    group: www-data
```

3.1.3. - file

Opciones requeridas:

- path=/directorio/fichero

Opciones opcionales:

- backup=yes/no
- force=yes/no
- owner=usuario
- group=grupo
- mode=modo
- state= {file, link, directory, hard, touch, absent}

Ejemplo:

```
  - name: Propiedades del fichero
    file:
      path: /tmp/hola.txt
      mode: 777
```

Asegurar que el directorio existe

```
- name: Asegurar que el directorio existe
    file:
      path: /var/log/journal
      state: directory
      owner: root
      group: systemd-journal
      mode: 2775
```

assegurar que el directorio no existe

```
- name: Asegurar que el directorio no existe
    file:
      path: /tmp/hola.txt
      state: absent
```

3.1.4. - stat

Opciones requeridas:

- path=/directorio/fichero

Opciones opcionales:

- get_attributes=true/false
- get_checksum=true/false
- get_md5=true/false
- get_mime=true/false

Ejemplos:

```
- name: Obtener información y guardarla
  stat:
    path: /etc/services
    register: datos

- name: Mostrar información
  debug:
    var: datos

- name: Usarlo en condición
  debug:
      msg: "Es directorio"
  when: datos.stat.isdir
```

3.1.5. - fetch

Opciones requeridas:

- src=/directorio/fichero # nodo
- dest=/directorio/fichero # ansible

Opciones opcionales:

- fail_on_missing=no/yes
- flat=no/yes # Copiar tal cual el fichero en el directorio indicando el fichero en el dest

Ejemplos:

```
- name: Guardar fichero de config red
  fetch:
    src: /etc/network/interfaces
    dest: /tmp/backup/
```

3.1.6. - unarchive

Opciones requeridas:

- src=/directorio/fichero
- dest=/directorio/fichero

Opciones opcionales:

- owner=usuario
- group=grupo
- mode=modo
- remote_src=no/yes
- list_files=no/yes

Ejemplos:

```
- name: Copia y extrae aplicación
  unarchive:
    src: oracle.tyz
    dest: /opt/oracle/


- name: Extrae fichero
  unarchive:
    src: /tmp/oracle.tyz
    dest: /opt/oracle/
    remote_src: true
```

3.1.7. - lineinfile

Opciones requeridas:

- line="texto"
- dest=/directorio/fichero

Opciones opcionales:

- owner=usuario
- group=grupo
- mode=modo
- backup=no/yes

- insertafter=expresión
- insertbefore=expresión
- regexp=expresión
- state=present/absent

Ejemplos:

```
- name: deshabilitar SELinux
  lineinfile:
    patch: /etc/selinux/config
    regexp: '^SELINUX='
    line: 'SELINUX=disabled'


- name: Borrar línea de un fichero
  lineinfile:
    patch: /etc/sudoers
    state: absent
    regexp: '^%wheel'


- name: Añadir antes de una línea de un fichero
  lineinfile:
    patch: /etc/apache2/ports.conf
    regexp: '^Listen'
    insertafter: "listen 80"
    line: "Listen 8080"
```

3.1.8. - blockinfile

Opciones requeridas:

- block="texto"
- dest=/directorio/fichero

Opciones opcionales:

- owner=usuario
- group=grupo
- mode=modo
- backup=no/yes

- insertafter=expresión
- insertbefore=expresión
- marker=marcador
- state=present/absent

Ejemplos:

```
- name: Configurar sshd.config
  blockinfile:
    patch: /etc/ssh/sshd.config
    block: |
       match user monitor
            PasswordAuthentication no
```

3.1.9. - openssl_privatekey

Opciones requeridas:

- path=/directorio/fichero

Opciones opcionales:

- force=false/true
- size=4096
- state=present/absent
- type=RSA/DSA

Ejemplos:

```
- name: Instala módulo de Python requerido
  apt:
    name: python.openssl
    state: latest

- name: Generar clave privada
  openssl_privatekey:
    path: /etc/ssl/private/oforte.net.pem
```

3.2. - Módulos Gestor Paquetes

Gestor de paquetes para lenguajes de programación:

- bower – Administra paquetes para desarrollo web
- bundler – Administra dependencias Ruby Gem.
- composer – Administra librerías PHP.
- cpanm – Gestiona módulos Perl (CPAN)
- easy_install – Gestiona módulos/librerías Python.
- gem – Gestiona Ruby Gems.
- maven_artifact – Descarga Artifacts desde un repositorio maven.
- npm – Gestionar paqueste node.js con npm.
- pear – Gestiona paquetes pear/pcl
- pip – Gestiona librerías/módulos Python.

Gestor de paquetes para Sistemas Operativos:

- apk – Gestiona paquetes Android
- apt / apt_key / apt_repository - Gestiona APT (Debian, Ubuntu,...)
- dnf – Gestiona paquetes Fedora
- macports – Gestion paquetes MacPorts
- openbsd_pkg – Gestiona paquetes OpenBSD
- Opkg – Gestiona paquetes OpenWRT
- package – Gestor de paquetes genérico (wrapper) Sirve para llamar otros.
- Pacman – Gestor de paquetes Arch Linux
- pkg5 – Gestor de paquetes Solaris 11
- pkgin – Gestiona paquetes SmartOS, NetBsd y otros
- pkgng – Gestiona paquetes FreeBSD >= 9.0
- portage – Gestiona paquetes Gentoo
- redhat_subscription – Administra registro y subscripciones de redhat
- slackpkg – Gestiona paquetes slackwhere >=12.2
- swdepot – Gestiona paquetes HP.UX
- yum / yum_repository - Gestiona paquetes YUM (Centoos, Red Hat, Rocky...)

- zypper / zypper_repository – Gestiona paquetes/repositorios SUSE/OpenSUSE

3.2.1. - Gestor de paquetes para lenguajes de programación

3.2.1.1. - cpanm

Opciones:

- from_path=ruta
- name=nombre
- locallib=ruta
- mirror=mirror
- mirror_only=no/yes
- notest=no/yes
- version=versión
- system_lib=directorio

Ejemplo:

```
- name: Instala primero gcc
  yum:
    name: gcc
    state: latest

- name: Instala primero cpanm
  yum:
    name: perl_App_cpanminus
    state: latest

- name: Instalar módulo DBI
  cpanm:
    name: DBI

 - name: Instalar versión específica
   cpanm:
     name: DBI
     version: "1.360"
```

3.2.1.2. - easy_install

Opciones requeridas:

- name=nombre

Opciones opcionales:

- state=present/latest
- virtualenv= no/yes
- virtualenv_command=comando
- virtualenv_site_packages=no/yes
- executable=ruta_ejecutable

Ejemplos:

```
- name: Instala pip
  easy_install:
    name: pip
    state: latest
```

3.2.1.3. - pip

Opciones:

- name=nombre
- state=present/latest/absent/forcereinstall
- virtualenv= no/yes
- virtualenv_command=comando
- virtualenv_site_packages=no/yes
- executable=ruta_ejecutable
- requirements=fichero.txt
- version=versión
- chdir=ruta

Ejemplos:

```
- name: Instala primero pip
  easy_install:
    name: pip
    state: latest
```

```
- name: Instala requests
  pip:
    name: requests
    state: latest

- name: Instalar requisitos
  pip:
    requirements: /miapp/requirements.txt
```

3.2.2. - Gestor de paquetes para Sistemas Operativos

3.2.2.1. - apt

Opciones:

- name=nombre [:versión]
- state=present/latest/absent/build-dep
- upgrade= no/yes/dist/full/safe
- force= no/yes
- update_cache=no/yes
- purge= no/yes
- dest=ruta_ejecutable
- autoremove=no/yes
- default_release=no/yes

Ejemplos:

```
- name: Instala nginx
  apt:
    name: nginx
    state: latest

- name: actualizar lista de paquetes
  apt:
    update_cache: yes

- name: Actualizar todos los paquetes
  apt:
    upgrade: dist
```

3.2.2.2. - apt_key

Opciones:

- data=Cloud
- file= fichero
- id=str
- state=present/absent
- url=str
- validate_certs=no/yes

Ejemplos:

```
  - name: Añadir clave usando servidor
    apt_key:
      keyserver: keyserver.ubuntu.com
      id: 36A107869245C8950F
```

```
- name: Añadir utilizando un fichero adjunto
 apt_key:
 url: "https://ftp.master.debian.org/keys/ansible.etc"
 state: present
```

3.2.2.3. - apt_repository

Opciones requeridas:

- repo=origen

Opciones opcionales:

- state=present/absent
- filename= nombre
- update_cache=no/yes
- validate_certs= no/yes
- mode=modo (420)

Ejemplos:

```
- name: Añadir repositorio google-chrome
  apt_repository:
    repo: deb http://dl.google.com/linux/chrome/dev/ stable main
    state: present
    filename: "google-chrome"

- name: Añadir repository Ubuntu
  apt_repository:
    repo: 'ppa:nginx/stable'
```

3.2.2.4. - package

Opciones requeridas:

- name=origen
- state=present/absent/latest

Opciones opcionales:

- use=auto/yum/apt

Ejemplos:

```
- name: Instalar última versión de ntpdate
  package:
    name: nptdate
    state: latest
```

3.2.2.5. - redhat_subscription

Opciones:

- state=present/absent
- activationkey=clave
- auto_attach=no/yes
- org_id=organ_id
- pool=nombre
- username=usuario
- password=clave
- server_hostname=servidor
- force_register=no/yes

Para la conexión podemos utilizar la activationkey o username/password.

Ejemplos:

```
- name: Registrar sistema
  redhat_subscription:
    state: present
    username:  usuario@mail.es
    password: clave
    auto_attach: yes

- name: Registrar sistema
  redhat_subscription:
    state: present
    activationkey: clave_rhel
    org_id: 2435
    pool: '^Red Hat Enterprise Server'
```

3.2.2.6. - yum

Opciones requeridas:

- name=nombre/ruta

Opciones opcionales:

- state=present/absent/latest
- conf_file= ruta
- disable_gpg_check=no/yes
- disablerepo= nombre
- enablerepo=nombre
- update_cache= no/yes

Ejemplos:

```
- name: Añadir última versión de apache
  yum:
    name: httpd
    state: latest

- name: Actualizar todos los paquetes
  yum:
    name: '*'
    state: latest
```

```
- name: Instalar grupo
  yum:
    name: '@Development tools'
    state: present
```

3.2.2.7. - yum_repository

Opciones requeridas:

- name=nombre/ruta

Opciones opcionales:

- state=present/absent
- description= descripción
- base_url=dirección
- file= nombre_fichero
- mirrorlist=dirección
- enabled= no/yes
- gpgcheck= no/yes

Ejemplos:

```
- name: Activar EPEL
  yum_repository:
    name: epel
    state: present
    description: 'EPEL YUM Repo'
    baseurl: https://download.fedoraproject.org/pub/epel/$releaserver/$basearch/
```

3.3. - Módulos Comandos

La siguiente lista permite ejecutar comandos en el nodo remoto:

- command
- expect – Ejecuta un comando y responde a la introducción de datos.
- raw – Envía comandos sin filtrar por ssh
- script – Transfiere y ejectura un script indicado
- shell – Utiliza /bin/sh para ejecutar los comandos. Permite pipelines con && o >>

3.3.1. - command

Opciones:

- chdir=/directorio
- creates=/fichero/comprobar
- executable=/ruta/binario
- removes=/fichero/comprobar

Ejemplos:

```
- name: Obtener uname
  command: uname -a
  register: salida_uname

- name: Crear base de datos si no existe
  command: /sbin/createdb sh
  args:
     chdir: /var/lib/mysql
     creates: /basededatos/existe
```

3.3.2. - expect

Opciones requeridas:

- command=comando
- responses=respuestas

Opciones Opcionales:

- chdir=/directorio
- creates=/fichero/comprobar
- echo=no/yes
- timeout=30

Ejemplos:

```
- name: Instalar pexpect
  yum:
    name: pexpect
    state: latest

- name: Cambiar contraseña usuario
  expect:
    command: password usuario
    responses:
      (?i)password: "supersecreta$"
```

3.3.3. - raw

Opciones:

- executable=/ruta/ejecutable

Ejemplos:

```
- name: Actualizar paquetes e instalar uno
  raw: apt update && apt install vim
```

3.3.4. - script

Opciones:

- creates=/fichero/comprobar
- removes=/fichero/comprobar
- decrypt=no/yes

Ejemplos:

```
- name: Copia y ejecuta el script
  script: /mi/fichero/local.sh argumentos
```

3.3.5. - shell

Opciones:

- chdir=/directorio
- creates=/fichero/comprobar
- executable=/ruta/binario /bin/sh

- removes=/fichero/comprobar

Ejemplos:

```
- name: Obtener uname
  command: uname -a | tee fichero.log
  register: salida_uname

- name: Obtener uname con argumentos
  command: uname -a | tee fichero.log
  register: salida_uname
  args:
     chdir: /tmp
     executable: /bin/bash
```

3.4. - Módulos Utilidades

Listado de módulos para diversas utilidades:

- assert – Asegura que se cumplan ciertas condiciones
- debug – Mostrar texto personalizado o el valor de una variable
- fail – Fallar con un mensaje expecífico
- include – Incluye un playbooks/tareas ##### DEPRECATED
- include_vars - utiliza para incluir variables desde archivos externos
- include_tasks - incluir otro archivo de tareas en tu playbook
- include_role - incluir roles de Ansible en tu playbook
- pause – Pausar la ejecución de un playbook x tiempo
- set_fact – Establece un fact
- wait_for – Espera a que se cumpla una condición para continuar

3.4.1. - assert

Opciones requeridas:

- that=condición/es

Opciones opcionales:

- msg="mensaje a mostrar"

Ejemplos:

```
- name: Se ejecuta si la distro es de RedHat
  assert:
    that: "ansible_os_family!="RedHat"

- name: Ejecutar si la variable número este entre el 0 y el 100
  assert:
    that:
      - "numero<=100"
      - "numero>=0"
    msg: "El número está entre el 0 y el 100"
```

3.4.2. - debug

Opciones:

- msg="mensaje a mostrar"
- var=variable
- verbosity=1 ### Cuantas v -vvv

Ejemplos:

```
- name: Mostrar texto con variables de ansible
  debug:
    msg: "Nombre {{ ansible_hostname }} {{ ansible_fqdn }}"

- name: ejecutar un comando y guardar resultado para más adelante
  shell: uptime
  register: resultado

- name: Recoger resultado anterior
  debug:
    var: resultado
```

3.4.3. - fail

Opciones:

- msg= "Mensaje a mostrar"

Ejemplos:

```
- name: Si el valor no es esperado mostrar mensaje
  fail:
    msg: "Dato incorrecto"
  when: valor not in ["y", "Y"]
```

3.4.4. - pause

Opciones:

- minutos= minutos
- prompt= "Texto a mostrar"
- seconds= segundos

Ejemplos:

```
- name: esperar 2 minutos
  pause:
    minutes: 2
- name: Indicar texto
  pause:
    prompt: "Comprobar acceso aplicación"
```

3.4.5. - includes

3.4.5.1. - include_vars

- depth - Especifica la profundidad máxima para buscar archivos
- dir -Define el directorio base para buscar el archivo de variables
- extensions -Lista de extensiones de archivo que se considerarán al buscar
- file - Especifica el archivo desde el cual cargar las variables.
- files_matching - Patrón de coincidencia de archivos para buscar archivos de variables
- free-form - Puede usarse para pasar variables adicionales al módulo
- hash_behaviour - Controla el comportamiento del manejo de errores
- ignore_files - Lista de nombres de archivo que deben ignorarse
- ignore_unknown_extensions - Si se establece en `true`, ignorará las extensiones de archivo desconocidas
- name - Define el nombre de la variable que contendrá los datos cargados

Ejemplo:

```
- name: Incluir variables desde el archivo
  include_vars:
    file: variables.yml
```

```
 - name: Mostrar las variables cargadas
    debug:
       var: user
```

3.4.5.2. - include_tasks

Opciones:

- apply - es útil para aplicar una estructura específica de tareas
- file - especificar un archivo diferente al que se incluirán las tareas.
- Free-form - pasar variables adicionales al módulo o personalizar su comportamiento

Ejemplo:

```
- name: Incluir tareas desde el archivo
  include_tasks: tareas.yml
```

3.4.5.3. - include_role

Opciones requeridas:

- name – Define el nombre a incluir

Opciones opcionales:

- allow_duplicates - Permite o prohíbe la inclusión de un rol más de una vez
- apply - aplicar una estructura específica de roles
- defaults_from - especificar otro rol desde el cual se deben cargar las variables predeterminadas
- handlers_from - especificar otro rol desde el cual se deben cargar los manejadores.
- rolespec_validate - Valida la especificación del rol antes de ejecutarlo.
- tasks_from - especificar otro rol desde el cual se deben cargar las tareas.
- vars_from - especificar otro rol desde el cual se deben cargar las variables.
- Public - Si se establece en true, las variables estarán disponibles para las tareas que siguen a la tarea

Ejemplo:

```
- name: Incluir rol en el playbook
  include_role:
     name: mi_rol
```

3.4.6. - set_falt

Opciones:

- set_fact: clave=valor

Ejemplo:

```
- name: Añadir fact
  set_fact:
     nombre: {{ ansible_hostname | upper }}

- name: Muestra facts
   debug:
      var: nombre
```

3.4.7. - wait_for

Opciones:

- state=present/absent/started/stopped
- port= puerto
- delay=segundos
- host=servidor
- exclude_hosts=servidores
- connection_timeout=segundos
- path= ruta
- search_regex=patrón
- timeout=segundos

Ejemplos:

```
    - name: Comprobar si puerto escucha
      wait_for:
        ports: 8080
        delay: 10

    - name: Esperar hasta que el fichero exista
      wait_for:
        path: /tmp/exista
    - name: Esperar hasta que el fichero NO exista
      wait_for:
        path: /tmp/exista
        state: absent
```

```
- name: Esperar hasta que un servidor este arrancado
  wait_for:
    ports: 22
    host: '{{ ansible_hostname }}'
    search_regex: OpenSSH
    delay: 10
```

3.5. - Módulos Notificaciones

Estos módulos permiten notificar a través de distintas vías: (chat, sms, mail...)

- cisco_spark – Envía un mensaje a un canal o usuario en Cisco Spark
- flowdock -
- hipchat
- irc
- jabber
- mattermost
- mqt – Mensajería IoT
- nexmo – Envía SMS
- pushbullet – A dispositivos
- pushover – A móviles
- rocketchat
- sendgrid – A través de la API de la plataforma
- slack
- sns – AWS SNS
- Telegram
- twilio

3.5.1. - hipchat

Opciones requeridas:

- token
- msq
- room

Opciones opcionales:

- api
- color
- from
- msq_format
- notify = yes/no
- validate_certs

Ejemplo:

```
- name: enviar mensaje a sala
  hipchat:
    token: codigo
    room: nombre_sala
    msg: "Ola ke ase"
```

3.5.2. - mail

Opciones requeridas:

- subject

Opciones opcionales:

- host
- port
- username
- password
- to
- body
- cc
- bcc

- secure

Ejemplo:

```
- name: Enviar mail
  mail:
    host: correo@invent.es
    port: 25
    subject: ansible-informe
    to: correo@elotro.es
    from:  esoes@invent.es
    body: "Ola, ke ase {{ ansible_fdqn }}"
```

3.5.3. - pushbullet

Opciones requeridas:

- api_key
- title

Opciones opcionales:

- body
- channel
- device
- push_type

Ejemplo:

```
- name: Instalar pushbullet.py
  pip:
    name: pushbullet.py
    state: present

- name: Enviar notificación
  pushbullet:
    api_key: clave
    device: dispositivo
    title: "Notificación ansible"
```

3.5.4. - pushover

Opciones requeridas:

- app_token
- user_key
- msg

Opciones opcionales:

- pri

Ejemplo:

```
- name: Enviar notificación
  pushover:
     app_token: clave
     user_key: clave
     msg: "Notificación desde ansible"
```

3.5.5. - rocketchat

Opciones requeridas:

- token
- domain

Opciones opcionales:

- msg
- channel
- username
- color
- protocol
- validate_certs

Ejemplo:

```
- name: enviar mensaje
  rockerchat:
     token: clave
     domain: claroquesi.guapi
     msg: "Ola ke ase"
     channel: #saluda
```

3.5.6. - telegram

Opciones requeridas:

- token

Opciones opcionales:

- api_args
  - chat_id: 000000
  - parse_mode: "markdown"
  - text: "Your precious application has been deployed: https://example.com"
  - disable_web_page_preview: true
  - disable_notification: true
- api_method

Ejemplos:

```
- name: Send notify to Telegram
  telegram:
    token: '9999999:XXXXXXXXXXXXXXXXXXXXXXX'
    api_args:
      chat_id: 000000
      parse_mode: "markdown"
      text: "Your precious application has been deployed: https://example.com"
      disable_web_page_preview: true
      disable_notification: true

- name: Forward message to someone
  telegram:
    token: '9999999:XXXXXXXXXXXXXXXXXXXXXXX'
    api_method: forwardMessage
    api_args:
      chat_id: 000000
      from_chat_id: 111111
      disable_notification: true
      message_id: '{{ saved_msg_id }}'
```

3.5.7. - slack

Opciones requeridas:

- token

Opciones opcionales:

- msg
- channel
- username
- color
- validate_certs

Ejemplo:

```
- name: enviar notificación slack
  slack:
    token: clave
    msg: "Ola ke ase {{ inventory_hostname }}"
    channel: #saluda
  delegate_to: localhost
  when: notificar == "slack"
```

3.6. - Módulos Bases de Datos

3.6.1. - mysql

- mysql_db - Add or remove MySQL databa...
- mysql_info - Gather informati...
- mysql_query - ...
- mysql_replication - Ma...
- mysql_role - Adds, removes, o...
- mysql_user - Adds or removes a user...
- mysql_variables - Manage ...
- proxysql_mysql_users - Adds or removes mysql users from pr...

3.6.1.1. - mysql_db

Opciones requeridas:

- name

Opciones opcionales:

- state: present/absent/dump/import
- login_host
- login_password
- login_port
- login_user
- encoding
- collation
- target

Ejemplo:

```
- name: Instalar librería python
  pip:
    name: python.mysql
    state: latest

- name: Crear si no existe la base de datos
  mysql_db:
    name: libros_bd
    state: present

- name: Copia de seguridad de todas las bbdd
  mysql_db:
    state: dump
    name: all
    target: /tmp/{{ ansible_hostname }}.sql
```

3.6.1.2. - mysql_user

Opciones requeridas:

- name

Opciones opcionales:

- state: present/absent
- password
- encrypted
- login_host
- login_password
- login_port
- login_user
- login_unix_socket
- priv
- append_privs

Ejemplo:

```
- name: Instalar librería python
  pip:
    name: python.mysql
    state: latest

- name: Crear usuario y darle permisos
  mysql_user:
    name: user_libros_bd
    password: clave
    state: present
    priv: "user_libros_bd.*:ALL
```

3.6.2. - Postgres

- postgresql_copy - Copy data between a file/program ...
- postgresql_db - Add or remove PostgreSQL databa...
- postgresql_ext - Add or remove PostgreSQL exte...
- postgresql_idx - Create or drop indexes from...
- postgresql_info - Gather information ab...
- postgresql_lang - Adds, removes or changes procedural languages with...
- postgresql_membership - Add or remove Postgr...
- postgresql_owner - Change an owner of Post...
- postgresql_pg_hba - Add, remove or modify a...
- postgresql.postgresql_ping - Check remote PostgreS...
- postgresql_privs - Grant or revoke privileges on Postg...
- postgresql_publication - Add, update, or remove ...
- postgresql_query - ...
- postgresql_schema - Add or re...
- postgresql_script - Run PostgreSQL ...
- postgresql_sequence - Create, drop, or alter...
- postgresql_set - Change a PostgreSQL server c...
- postgresql_slot - Add or remove replication slots from...
- postgresql_subscription - Add, update, or remove P...
- postgresql_table - Create, drop, or mod...
- postgresql_tablespace - Add or remove PostgreSQL tablesp...
- postgresql_user - Create, alter, or remove a user (role) from a Post...
- postgresql_user_obj_stat_info - Gather statistics about P...

3.6.2.1. - postgresql_db

Opciones requeridas:

- name

Opciones opcionales:

- state: present/absent
- login_host
- login_password
- port=3306
- login_user
- login_unix_socket
- encoding
- lc_collation
- template

Ejemplo:

```
- name: Instalar librería python
  pip:
    name: psycopg2
    state: latest

- name: Crear si no existe la base de datos
  postgresql_db:
    name: libros_bd
    state: present
    encoding: UTF-8
```

3.6.2.2. - postgresql_user DEPRECATED » postgresql_privs

Opciones requeridas:

- database

Opciones opcionales:

- state: present/absent
- login_host
- login_password
- port=3306
- login_user
- login_unix_socket
- password
- encrypted
- privs

Ejemplo:

```
- name: Instalar librería python
  pip:
    name: psycopg2
    state: latest

- name: Crear si no existe la base de datos
  postgresql_privs:
    database: library
    state: present
    privs: SELECT,INSERT,UPDATE
    type: table
    objs: books,authors
    schema: public
    roles: librarian,reader
    grant_option: true
```

3.6.3. - Mongodb

- mongodb_balancer - Manages the MongoDB Sh...
- mongodb_index - Creates or drops indexes ...
- mongodb_info - Gather information ...
- mongodb_maintenance -Enables or disables maintenance mode ...
- mongodb_oplog - Res...
- mongodb_parameter - Change an administrative paramet...
- mongodb_replicaset - Initialise...
- mongodb_role - Adds or removes a role f...
- mongodb_schema - Manages MongoDB Docu...
- mongodb_shard - Add or remove shards ...
- mongodb_shard_tag - ...
- mongodb_shard_zone - ...
- mongodb_shell - Run commands...
- mongodb_shutdown - Cleans up all database resources and then terminates the...
- mongodb_status - Validates the st...
- mongodb_stepdown - Step down the MongoDB nod...
- mongodb_user - Adds or removes a user f...

3.6.3.1. - mongodb_user

Opciones requeridas:

- database
- name

Opciones opcionales:

- state: present/absent
- login_host
- login_password
- login_port=3306
- login_user

- roles = 'read', 'readWrite', 'dbAdmin', 'userAdmin', 'clusterAdmin', 'readAnyDatabase', 'readWriteAnyDatabase', 'userAdminAnyDatabase', 'dbAdminAnyDatabase'

Ejemplo:

```
- name: Create 'burgers' database user with name 'bob' and password '12345'.
  community.mongodb.mongodb_user:
    database: burgers
    name: bob
    password: 12345
    state: present

- name: Create a database user via SSL (MongoDB must be compiled with the SSL option and configured properly)
  community.mongodb.mongodb_user:
    database: burgers
    name: bob
    password: 12345
    state: present
    ssl: True

- name: Delete 'burgers' database user with name 'bob'.
  community.mongodb.mongodb_user:
    database: burgers
    name: bob
    state: absent

- name: Define more users with various specific roles (if not defined, no roles is assigned, and the user will be added via pre mongo 2.2>
  community.mongodb.mongodb_user:
    database: burgers
    name: ben
    password: 12345
    roles: read
    state: present

- name: Define roles

  community.mongodb.mongodb_user:
    database: burgers
    name: jim
    password: 12345
    roles: readWrite,dbAdmin,userAdmin
    state: present
```

3.6.4. - influxdb

- influxdb_database - Man...
- influxdb_query - Query dat...
- influxdb_retention_policy - Manage Influ...
- influxdb_user - ...
- influxdb_write - Write dat...

3.6.4.1. - influxdb_database

Opciones requeridas:

- database_name
- hostname

Opciones opcionales:

- state: present/absent
- password
- port=8086
- username

Ejemplos:

```
- name: Create database
  influxdb_database:
     hostname: "{{influxdb_ip_address}}"
     database_name: "{{influxdb_database_name}}"

- name: Destroy database
  influxdb_database:
     hostname: "{{influxdb_ip_address}}"
     database_name: "{{influxdb_database_name}}"
     state: absent

- name: Create database using custom credentials
  influxdb_database:
     hostname: "{{influxdb_ip_address}}"
     username: "{{influxdb_username}}"
     password: "{{influxdb_password}}"
     database_name: "{{influxdb_database_name}}"
     ssl: true
     validate_certs: true
```

3.6.5. - Vertica

- vertica_configuration - Updates Vertica co...
- vertica_info - Gathers ...
- vertica_role - Adds or removes Vertica database roles and...
- vertica_schema - Adds or removes Vertica dat...
- vertica_user - Adds or removes Vertica database u...

3.6.6. - Misc

- elasticsearch_plugin
- kibana_plugin
- redis
- riats

3.7. - Otros módulos de interés

Revisa la documentación con ansible-doc

- Módulos Gestionar Sistema
  - alternatives – Gestionar alternativas para comandos
  - at – Programar ejecución
  - authorized.keys – Gestiona el fichero de SSH para las claves públicas autorizadas.
  - cron
  - crypttab – Cifrar dispositivos
  - filesystem
  - fireworld
  - gluster.volume
  - group
  - hostname
  - iptables
  - know_hosts - Añadir o eliminar registros

    - lvg – Grupos de volumenes LVM
    - lvol – Volumenes lógicos LVM
    - mount
    - open_iscsi
    - openwrt_init
    - pam_limits
    - pamd
    - ping
    - SELinux
        - seboolean
        - selcontext
        - selique
        - selique_permissive
        - seport
    - service
    - setup
    - sysctl
    - systemd
    - timezone
    - user
- Módulos Windows
    - win_acl
    - win_chocolatey
    - win_command
    - win_copy
    - win_environment
    - win_feature
    - win_file
    - win_get_url

    - win_group
    - win_line_file
    - win_msi
    - win_package
    - win_ping
    - win_reboot
    - win_regedit
    - win_schedule_task
    - win_service
    - win_share
    - win_shell
    - win_stat
    - win_template
    - win_timezone
    - win_unzip
    - win_updates
    - win_uri
    - win_user
- Módulos Control Versiones
    - bzr
    - git
    - git_config
    - github_deploy_key
    - github_issue
    - github_key
    - github_release
    - github_repo
    - github_webhook
    - github_webhook_info

    - gitlab_branch
    - gitlab_deploy_key
    - gitlab_group
    - gitlab_group_members
    - gitlab_group_variable
    - gitlab_hook
    - gitlab_instance_variable
    - gitlab_merge_request
    - gitlab_project
    - gitlab_project_badge
    - gitlab_project_members
    - gitlab_project_variable
    - gitlab_protected_branch
    - gitlab_runner
    - gitlab_user
    - hg
    - subversion
- Módulos Infra Web
    - apache2_mod_proxy
    - apache2_module
    - deploy_helper
    - djanfo_manage
    - ejabberd_user
    - htpasswd
    - jboss
    - jenkins_job
    - jenkins_plugin
    - jira
    - letsencrypt

    - supervisorctl
    - taiga_issue
    - rabbitmq_binding
    - rabbitmq_exchange
    - rabbitmq_feature_flag
    - rabbitmq_global_parameter
    - rabbitmq_parameter
    - rabbitmq_plugin
    - rabbitmq_policy
    - rabbitmq_publish
    - rabbitmq_queue
    - rabbitmq_upgrade
    - rabbitmq_user
    - rabbitmq_user_limits
    - rabbitmq_vhost
    - rabbitmq_vhost_limits
    - win_rabbitmq_plugin
- Módulos Cloud Cluster - https://docs.ansible.com/ansible/2.9/modules/list_of_cloud_modules.html
    - AWS
        - cloudformation
        - cloudtrail
        - dynamodb
        - ec2
        - elasticache
        - iam
        - route53
        - s3
        - sns/sqs/srs
    - Atomic

        - host
        - image
    - Azure
        - network
            - interfaces
            - public IP
            - subnets
            - virtual networks
        - resource groups
        - security groups
        - storage accounts
        - virtual machines
    - Centurylink
        - alerts
        - firewall
        - load balancer
        - servers
        - public IP
    - CloudStack
        - Account
        - firewall
        - host
        - network
        - roles
        - routers
    - Digital Ocean
        - block storage
        - domain
        - ssh key

        - droplets
        - tags
    - docker
        - contenedor
        - imagenes
        - network
        - servicios
    - gcloud
        - instancias
        - storage
        - dns
        - load balancer
        - redes/firewall
        - tags
        - backend service
    - openstack
    - avirt
    - preofitbricks
    - rockspace
    - vmware
    - ....
- Módulos Monitorización
    - datadog_downtime
    - datadog_event
    - datadog_monitor
    - grafana_dashboard
    - grafana_datasource
    - grafana_folder
    - grafana_notification_channel

- grafana_organization
- grafana_organization_user
- grafana_plugin
- grafana_team
- grafana_user
- logic_monitor
- logic_monitor_facts
- monit
- nagios
- newrelic_deployment
- pagerduty
- pagerduty_alert
- sensu_check
- sensu_subscription
- zabbix_action
- zabbix_authentication
- zabbix_autoregister
- zabbix_discovery_rule
- zabbix_globalmacro
- zabbix_group
- zabbix_group_info
- zabbix_host
- zabbix_host_events_info
- zabbix_host_info
- zabbix_hostmacro
- zabbix_housekeeping
- zabbix_maintenance
- zabbix_map
- zabbix_mediatype

- zabbix_proxy
- zabbix_proxy_info
- zabbix_regexp
- zabbix_script
- zabbix_service
- zabbix_settings
- zabbix_template
- zabbix_template_info
- zabbix_token
- zabbix_user
- zabbix_user_directory
- zabbix_user_info
- zabbix_user_role
- zabbix_usergroup
- zabbix_valuemap

TEMA 4 - Galaxy

Ansible Galaxy es un repositorio en linea, gratuito, donde se alojan roles a ser utilizados por nuestros playbooks.

https://galaxy.ansible.com/ui/

Es posible compartir los roles propios, autenticandose con una cuenta github.

A través del comando ansible-galaxy es posible:

- Descargar roles desde Galaxy o SCM (Sistema de Control de Versiones)
- Crear roles
- Eliminar roles
- Realizar tareas en la web de Galaxy (subir/editar/eliminar roles)

La sintaxis es ansible-galaxy [acción] [opciones] [argumentos]

Acciones:

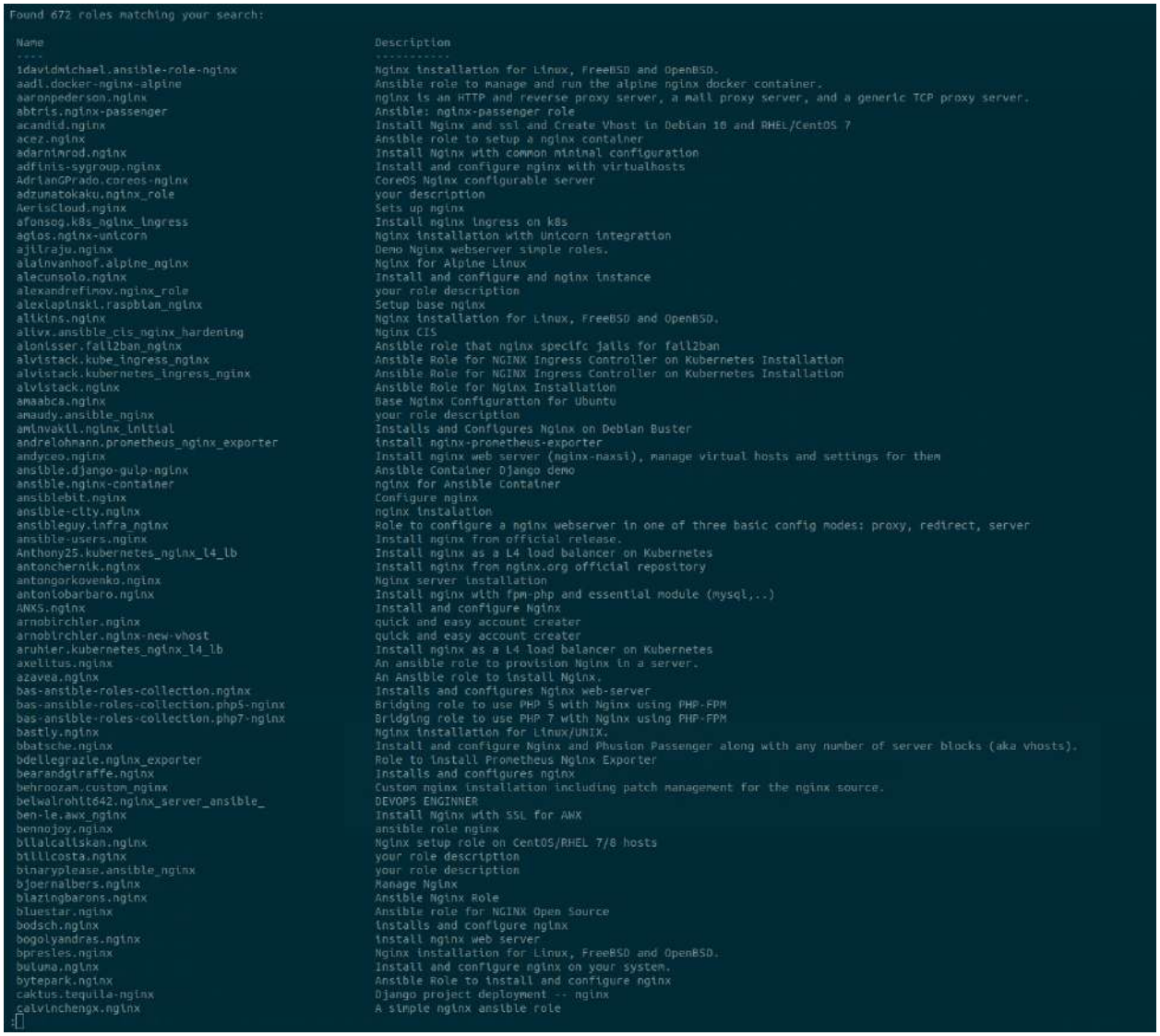
- delete – Elimar un rol de Galaxy
- import – Desde Github a Galaxy
- info
- init – Inicializa una estructura de directorios.
- install – Descarga rol (Por defecto en /etc/ansible/roles/, pero se puede personalizar en ansible.cfg)
- list
- login
- remove – Eliminar un rol del servidor pero no de la web de Galaxy
- search
- setup – Crear integración con travis CI

Opciones:

- -f / --force – Sobreescribe el rol
- -i / --ignore-errors – Ignora errores
- -n / --no-deps - No instala dependencias
- -p / --role-path – Especifica un directorio distinto
- -r / --roles-file – Indicar fichero con lista de roles a instalar

4.1. - Ejemplo de comandos

Ejemplo de "ansible-galaxy search nginx":



```
Found 672 roles matching your search:

 Name                                          Description
 ----                                          -----------
 1davidmichael.ansible-role-nginx              Nginx installation for Linux, FreeBSD and OpenBSD.
 aadl.docker-nginx-alpine                      Ansible role to manage and run the alpine nginx docker container.
 aaronpederson.nginx                           nginx is an HTTP and reverse proxy server, a mail proxy server, and a generic TCP proxy server.
 abtris.nginx-passenger                        Ansible: nginx-passenger role
 acandid.nginx                                 Install Nginx and ssl and Create Vhost in Debian 10 and RHEL/CentOS 7
 acez.nginx                                    Ansible role to setup a nginx container
 adarnimrod.nginx                              Install Nginx with common minimal configuration
 adfinis-sygroup.nginx                         Install and configure nginx with virtualhosts
 AdrianGPrado.coreos-nginx                     CoreOS Nginx configurable server
 adzunatokaku.nginx_role                       your description
 AerisCloud.nginx                              Sets up nginx
 afonsog.k8s_nginx_ingress                     Install nginx ingress on k8s
 agios.nginx-unicorn                           Nginx installation with Unicorn integration
 ajilraju.nginx                                Demo Nginx webserver simple roles.
 alainvanhoof.alpine_nginx                     Nginx for Alpine Linux
 alecunsolo.nginx                              Install and configure and nginx instance
 alexandrefimov.nginx_role                     your role description
 alexlapinskl.raspbian_nginx                   Setup base nginx
 alikins.nginx                                 Nginx installation for Linux, FreeBSD and OpenBSD.
 alivx.ansible_cis_nginx_hardening             Nginx CIS
 alonisser.fail2ban_nginx                      Ansible role that nginx specifc jails for fail2ban
 alvistack.kube_ingress_nginx                  Ansible Role for NGINX Ingress Controller on Kubernetes Installation
 alvistack.kubernetes_ingress_nginx            Ansible Role for NGINX Ingress Controller on Kubernetes Installation
 alvistack.nginx                               Ansible Role for Nginx Installation
 amaabca.nginx                                 Base Nginx Configuration for Ubuntu
 amaudy.ansible_nginx                          your role description
 aminvakil.nginx_initial                       Installs and Configures Nginx on Debian Buster
 andrelohmann.prometheus_nginx_exporter        install nginx-prometheus-exporter
 andyceo.nginx                                 Install nginx web server (nginx-naxsi), manage virtual hosts and settings for them
 ansible.django-gulp-nginx                     Ansible Container Django demo
 ansible.nginx-container                       nginx for Ansible Container
 ansiblebit.nginx                              Configure nginx
 ansible-city.nginx                            nginx instalation
 ansibleguy.infra_nginx                        Role to configure a nginx webserver in one of three basic config modes: proxy, redirect, server
 ansible-users.nginx                           Install nginx from official release.
 Anthony25.kubernetes_nginx_l4_lb              Install nginx as a L4 load balancer on Kubernetes
 antonchernik.nginx                            Install nginx from nginx.org official repository
 antongorkovenko.nginx                         Nginx server installation
 antonlobarbaro.nginx                          Install nginx with fpm-php and essential module (mysql,..)
 ANXS.nginx                                    Install and configure Nginx
 arnobirchler.nginx                            quick and easy account creater
 arnoblrchler.nginx-new-vhost                  quick and easy account creater
 aruhier.kubernetes_nginx_l4_lb                Install nginx as a L4 load balancer on Kubernetes
 axelitus.nginx                                An ansible role to provision Nginx in a server.
 azavea.nginx                                  An Ansible role to install Nginx.
 bas-ansible-roles-collection.nginx            Installs and configures Nginx web-server
 bas-ansible-roles-collection.php5-nginx       Bridging role to use PHP 5 with Nginx using PHP-FPM
 bas-ansible-roles-collection.php7-nginx       Bridging role to use PHP 7 with Nginx using PHP-FPM
 bastly.nginx                                  Nginx installation for Linux/UNIX.
 bbatsche.nginx                                Install and configure Nginx and Phusion Passenger along with any number of server blocks (aka vhosts).
 bdellegrazie.nginx_exporter                   Role to install Prometheus Nginx Exporter
 bearandgiraffe.nginx                          Installs and configures nginx
 behroozam.custom_nginx                        Custom nginx installation including patch management for the nginx source.
 belwalrohit642.nginx_server_ansible_          DEVOPS ENGINNER
 ben-le.awx_nginx                              Install Nginx with SSL for AWX
 bennojoy.nginx                                ansible role nginx
 bilalcallskan.nginx                           Nginx setup role on CentOS/RHEL 7/8 hosts
 billlcosta.nginx                              your role description
 binaryplease.ansible_nginx                    your role description
 bjoernalbers.nginx                            Manage Nginx
 blazingbarons.nginx                           Ansible Nginx Role
 bluestar.nginx                                Ansible role for NGINX Open Source
 bodsch.nginx                                  installs and configure nginx
 bogolyandras.nginx                            install nginx web server
 bpresles.nginx                                Nginx installation for Linux, FreeBSD and OpenBSD.
 buluma.nginx                                  Install and configure nginx on your system.
 bytepark.nginx                                Ansible Role to install and configure nginx
 caktus.tequila-nginx                          Django project deployment -- nginx
 calvinchengx.nginx                            A simple nginx ansible role
:
```

Podemos obtener información con "ansible-galaxy info bennojoy.nginx"

```
❯ ansible-galaxy info  bennojoy.nginx

Role: bennojoy.nginx
        description: ansible role nginx
        commit:
        commit_message:
        created: 2023-05-08T20:29:48.593423Z
        download_count: 3429
        github_branch: master
        github_repo: nginx
        github_user: bennojoy
        id: 3924
        imported: None
        modified: 2023-10-29T18:44:46.892920Z
        path: ('/home/v/.ansible/roles', '/usr/share/ansible/roles', '/etc/ansible/roles')
        upstream_id: 2
        username: bennojoy
```

Se instala con "ansible-galaxy install bennojoy.nginx" y después podríamos verlo en el directorio por defecto "ls /etc/ansible/roles"

Después podemos ver los roles disponibles con "ansible-galaxy list"

Si queremos crear una estructura de rol, queramos importarla a Galaxy o no, podemos crearla con "ansible-galaxy init prueba_estructura_rol".

```
❯ ansible-galaxy init 01_estructura_rol
- Role 01_estructura_rol was created successfully
❯ tree
.
└── 01_estructura_rol
    ├── defaults
    │   └── main.yml
    ├── files
    ├── handlers
    │   └── main.yml
    ├── meta
    │   └── main.yml
    ├── README.md
    ├── tasks
    │   └── main.yml
    ├── templates
    ├── tests
    │   ├── inventory
    │   └── test.yml
    └── vars
        └── main.yml

9 directories, 8 files
```

Para compartir un rol:

1. Deberemos crear un repo en github.
2. Inicializar el rol con ansible-galaxy (Recomendado)
3. Editar meta/main.yml para especificar autor, descripción, plataformas y etiquetas (Estas dos últimas son las que utilizarán para encontrar el rol mediante la acción search del comando. "Platforms" es obligatorio.

4. Publicar el código (commit/push) en el repo github.
5. Iniciar sesión "ansible-galaxy login"
6. Y ahora ya podemos importar nuestro trabajo de Github a Galaxy: "ansible-galaxy import [github-usuario] [github-repositorio]"

Para borrar un rol de Galaxy usamos el comando "ansible-galaxy delete [github-usuario] [github-repositorio]"

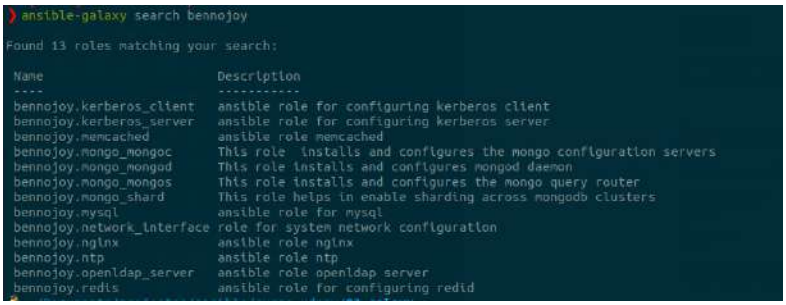
4.2. - Ficheros roles

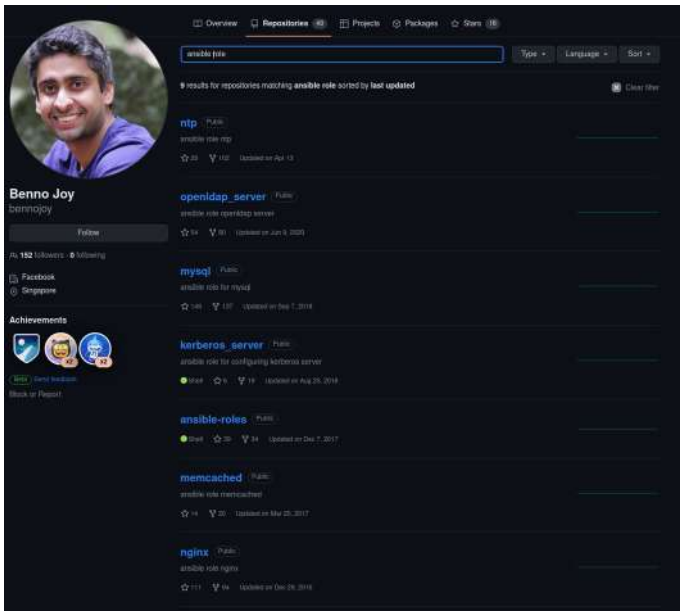
Es posible crear un fichero con el listado de roles a instalar.

Los atributos que se pueden especificar son los siguientes:

- src – El origen del rol
  - usuario nombre – Instalar desde Galaxy
  - Dirección SCM - http://..., https://..., gitthttp..., git@...
- scm – Especifica el tipo de SCM: git o hg
- version – Especifica la versión a instalar
- name – Especifica un nombre distinto al original

Ejemplo:





- src: bennojoy.mysql

- src: https://github.com/bennojoy/redis

- src: https://github.com/bennojoy/ntp
  version: master
  name: nginx_master

Para instalar tan solo tenemos que usar el comando: "ansible-galaxy install -r requirements.yml"

TEMA 5 - Tower

Ansible tower es una plataforma de automatización y gestión de configuración desarrollada por Red Hat. proporciona una interfaz web y otras características adicionales para facilitar la administración y la implementación de la automatización Ansible.

Algunas de las características clave de Ansible Tower incluyen:

1. **Interfaz web:** Proporciona una interfaz gráfica de usuario para administrar y monitorear entornos de automatización basados en Ansible.
2. **Control de acceso:** Permite gestionar los permisos y accesos de los usuarios a través de roles y permisos predefinidos.
3. **Programación de trabajos:** Permite programar y ejecutar trabajos de automatización en momentos específicos.
4. **Registro y auditoría:** Ofrece capacidades de registro y auditoría para realizar un seguimiento de las actividades y cambios realizados por los usuarios.
5. **Dashboards y paneles:** Proporciona paneles visuales para mostrar el estado de los trabajos y la salud general del entorno.
6. **Escalabilidad:** Facilita la gestión de entornos de automatización a gran escala y distribuidos.

Los componentes son los siguientes:

- **Proyectos** (Repositorio de playbooks) Local o SCM (git, hg o svn)
- **Inventarios** (Estáticos o dinámicos (cloud, satellite))
- **Plantillas de trabajo** (Definición de playbooks, inventarios y diversas opciones como sudo/su, credenciales, registros, etc)
- **Trabajos** (Ejecución de las plantillas) Resultado y salida.
- **Configuraciones** (Credenciales, equipos, organizaciones, etc)

ansible-tower de RedHat tiene tres ediciones:

- **self-support** – Sin soporte ni características especiales (LDAP, system tracking, cuestionarios, etc)
- **standard** – Con soporte 8x5 però sin características especiales
- **premium** – Soporte 24x7 y características especiales.

Es gratuita la edición self-support para 10 nodos o menos.

La edicción premium tiene un trial de un mes:
https://www.redhat.com/en/technologies/management/ansible/trial

5.1. - Instalación ansible tower

Docs https://docs.ansible.com/ansible-tower/latest/html/userguide/overview.html

Requisitos:

- Servidor específico para tower (No compartir con otras aplicaciones) Servidor web + servidor bbdd que puede crear conflictos
- Distribuciones soportadas: RHEL 7, Centos 7, Ubuntu 14.04
- Navegador chrome o firefo. Otros pueden funcionar pero no soportados
- Al menos 2 GB de RAM (Recomendado 4 GB)
- Al menos 20 GB de disco, de los cuales 10 GB deben ser dedicados a /var.
- En el caso de RHEL /Centos es necesario habilitar repositorio EPEL.

Escenarios:

1. Un servidor con la aplicación y el servidor
2. Un servidor con la aplicación y una bbdd postgress a parte
3. Alta disponibilidad. Distintos servidores con la aplicación apuntando a la misma bbdd

Lo recomendable es que la bbdd tenga un respaldo.

Pasos:

- RHEL/Centos - Instalar repositorio EPEL: https://docs.fedoraproject.org/es/epel/ En algunos casos podemos necesitar el repositorio de extras.
- Ubuntu – apt install sofwarr-poperties-common && apt-add-repository ppa:ansible/ansible
- Dos métodos de instalación:
  - Instalador – https://releases.ansible.com/ansible-tower/setup
  - Bundle: https://releases.ansible.com/ansible-tower/setup-bundle
- Descomprimir instalador: tar xvfz ansible-tower-setup-[bundle]-latest.tar.gz
- Editar fichero inventory con los servidores y contraseñas

```
[tower]
localhost ansible_connection=local

[database]

[all:vars]
admin_password='Start123'

pg_host=''
pg_port=''

pg_database='awx'
pg_username='awx'
pg_password='Start123'

rabbitmq_port=5672
rabbitmq_vhost=tower
rabbitmq_username=tower
rabbitmq_password=''
rabbitmq_cookie=cookiemonster
```

- Ejecutar ./setup.sh
- Se puede seguir el log de la instalación en /var/log/tower/setup-.....log
- Si todo ha ido bien ya podremos acceder a la web.
- Configuramos la licencia.
- Se debe configurar los puertos https 443 y el 8080 para los eventos en tiempo real.

5.2. - Configuración ansible tower

Un vez instalado es possible configurar los siguientes elementos:

- **Organizaciones** – Agrupar contenido para administrar permisos de los defierentes departamentos.
- **Usuarios** – Permisos
- **Equipos** – Dividir una organización para asociar contenido y sus permisos
- **Credenciales** – Añadir contraseñas, claves ssh, etc. Para acceder a servidores, inventarios o proyectos.
- **Trabajos de gestión** – Limpiar historial, flujo de actividad, etc
- **scripts de inventario** – En el caso de no tener integración nativa al inventario, es possible definit un script personalizado para obtener los servidores.
- **Notificaciones**
- **Licencia**
- **Configuraciones tower**

TEMA 6 - Avanzado

6.1. - Modulo Debug

El módulo debug nos ayuda a visualizar el contenido de variables o expresiones de las plantillas.

Sintaxis:

```
- debug:
    msg: "Cadena de texto a interpretar"
```

Ejemplo:

```
- name: Ejemplo debug
  hosts: localhost
  tasks:
    - debug:
        var: ansible_host
    - debug:
        msg: "Mi IP es {{ ansible_host }}"
```

El parámetro verbosity (numérico. Indica el nivel de "verbose" que muestra

6.2. - tags

Las etiquetas nos permiten especificar selectivamente qué tareas se ejecutarán o se omitirán.

Sintaxis:

```
- modulo: acciones
  tags: etiqueta1

- modulo: acciones
  tags: [etiqueta1, etiqueta2]
```

En la línea de comando se puede expecificar con la opción --tags.

Existe una tag especial llamada always que provoca que siempre se ejecutará.

6.3. - lookup

La directiva lookup nos permite obtener datos desde el servidor que está ejecutando el playbook.

Ejemplo: "{{ lookup('file', '/etc/motd') }}"

```
- name: Ejemplo lookup 1
  hosts: www
  tasks:
     - debug:
          msg: "{{ lookup('file', '/etc/motd') }}"

- name: Ejemplo lookup 2 en variable
  hosts: www
  tasks:
     - set_fact:
          etc_motd: "{{ lookup('file', '/etc/motd') }}"
     - copy:
          dest: "/tmp/ejemplo"
          content: "{{ etc_motd }}"
```

Otros plugins:

- password – Genera y almacena una clave. "{{ lookup('password', '/tmp/pass.txt') }}"
- csvfile – Lee ficheros csv. "{{ lookup('csvfile', 'fila2 file=test.csv delimiter=, col=2') }}"
- ini – Lee ficheros ini. "{{ lookup('ini', 'engine section=PHP file=/etc/php5/cli.php.ini') }}"
- dig – Resolver DNS. "{{ lookup('dig', 'vergaracarmona.es', wantlist=True') }}"
- env – Variable de entorno. "{{ lookup('env', 'DISPLAY') }}"
- template "{{ lookup('template', 'prueba.j2') }}"

6.4. - ansible Vault

El comando ansible-vault permite cifrar ficheros con una contrasña para proteger datos sensibles.

Sintaxis:

ansible-vault create fichero.yaml

ansible-vault edit fichero.yaml

ansible-vault encrypt fichero.yaml

ansible-vault decrypt fichero.yaml

ansible-vault view fichero.yaml

El comando ansible-playbook incluye dos opciones para indicar la clave: --ask-vault-pass * y --valut-password-file=file

6.5. - Tareas asincronas

Cuando realizamos una tarea en un servidor a través de un módulo, la conexión permanece abierta esperando su finalización. Ansible nos da la opción de realizarla en segundo plano y consultar el estado periodicamente.

Ejemplo de una espera que puede ser innecesaria:

```
- name:  Ejemplo async
  hosts: www
  tasks:
    - command: sleep 10
```

Existen varios métodos.

Método 1 – Con comprobación cada x tiempo

```
- modulo: Argumentos
   async: tiempo_maximo
   poll: tiempo_consulta
```

Ejemplo:

```
- name:  Ejemplo async
  hosts: www
  tasks:
    - command: sleep 10
       async: 30
       poll: 5
```

Método 2 – Dejando que sea otra tarea quien lo compruebe mediante variable

```
- modulo: Argumentos
   async: tiempo_maximo
   poll: 0
   register: tarea_async
```

Ejemplo:

```
- name:  Ejemplo async
  hosts: www
  tasks:
     - command: sleep 10
       async: 30
       poll: 0
       register: estado_sleep
     - degub:
          msg: "Otra tarea"
     - degub:
          msg: "Otra tarea más"
     -  async_status:
           jid: "{{ estado_sleep.ansible_job_id }}"
       register: estado
       until: estado.finished
       retries: 30
```