gradiant
Connectivity · Intelligence · Security

Automatizando Aplicaciones con Ansible

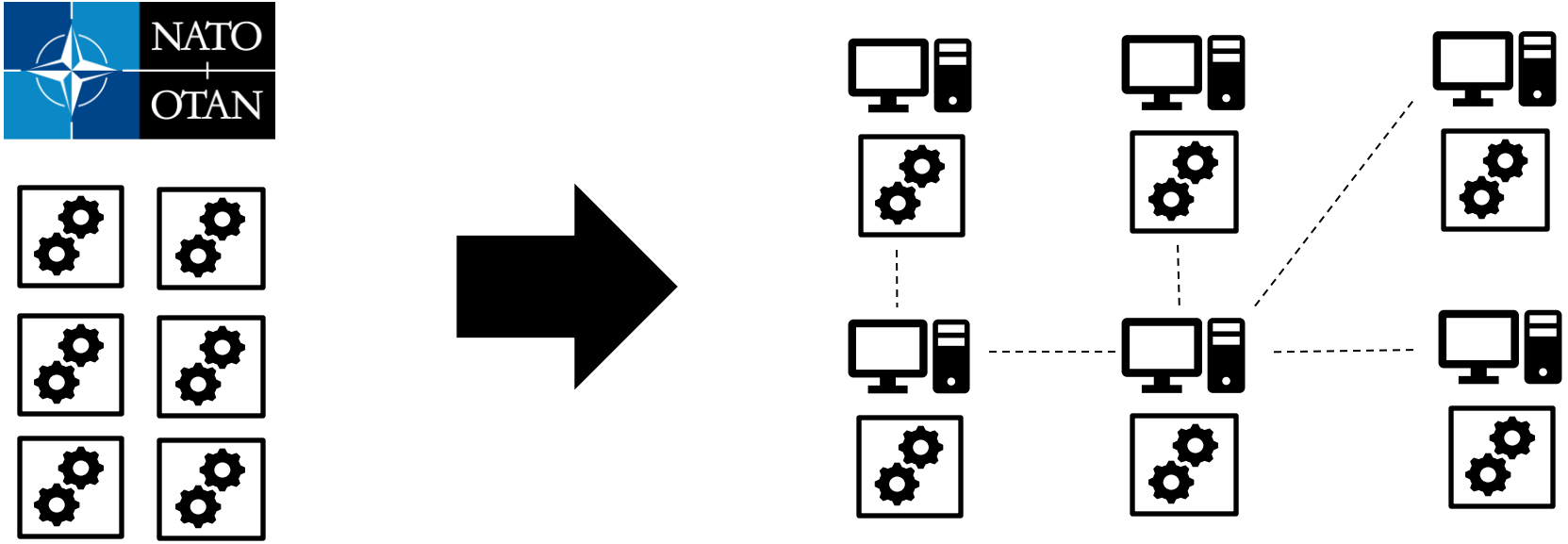
Sobre mi

Manuel Landín Gómez

Ingeniero – Investigador
Línea Cloud Native, **Gradiant**

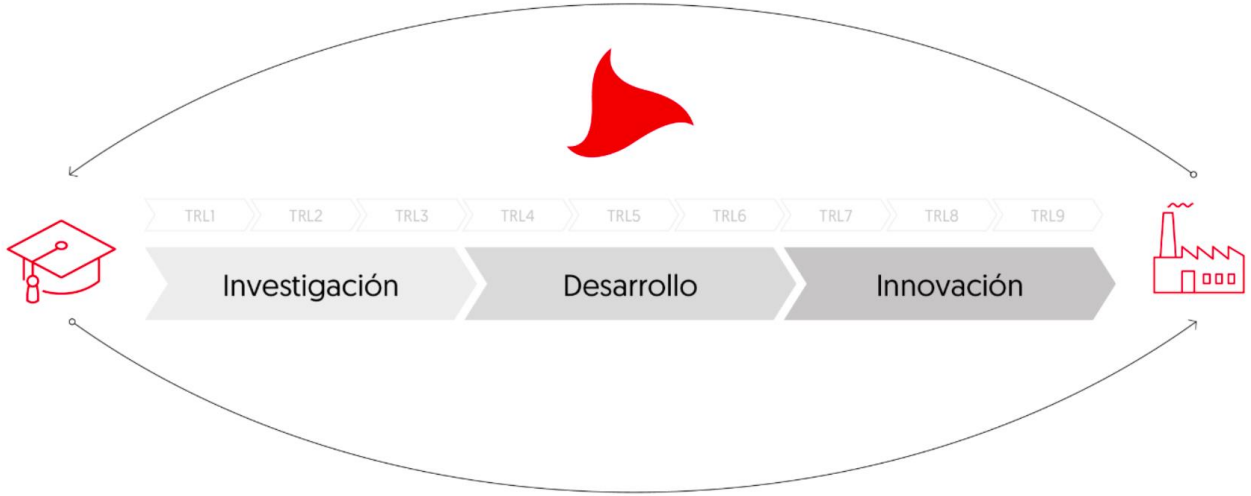
docker kubernetes YAML HashiCorp Terraform ANSIBLE GO git GitHub

Sobre mi trabajo con Ansible en Gradiant



Sobre Gradiant

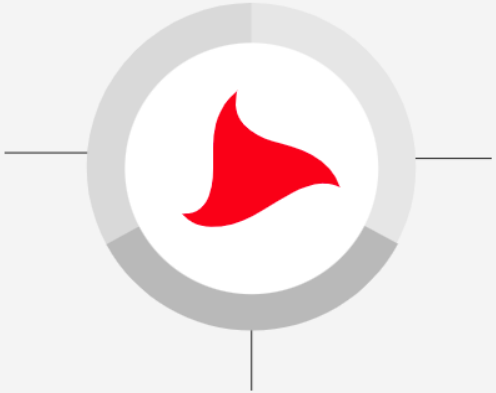
Gradiant es una fundación privada, sin ánimo de lucro establecida en diciembre de 2007 y nacida con el objetivo de alinear la I+D universitaria con las demandas empresariales, desempeñando un papel fundamental en la generación y transferencia de conocimiento en tecnologías de la información en las comunicaciones (TIC) hacia las empresas.



Sobre los ejes tecnológicos de Gradiant

Conectividad

Sistemas de comunicaciones en internet de las cosas para la transmisión de datos



Inteligencia

Infraestructura y algoritmo para la extracción de valor de los datos convirtiéndolos en información útil y accionable

Seguridad

Protección de los datos y los sistemas de información y protección de la privacidad

gradiant
Connectivity · Intelligence · Security

Automatizando Aplicaciones con Ansible, ahora sí

¿Qué es Ansible?

Ansible es una plataforma de automatización de código abierto.

Simple. Fácil de leer, escribir y comunicar para un humano.

Potente. Despliega aplicaciones, gestiona configuraciones, automatiza flujos de trabajo...

Sin agentes. No necesita instalar software en los hosts para funcionar.

¿Cómo interactúo con Ansible?

Ansible instala una serie de binarios en nuestro sistema. Estos binarios conforman la interfaz que usamos para comunicarnos con Ansible:

ansible-playbook Ejecuta el playbook especificado.

```
ansible-playbook <playbook.yaml>
```

ansible-galaxy Accede al repositorio Galaxy de Ansible.

```
ansible-galaxy [collection | role] <action>
```

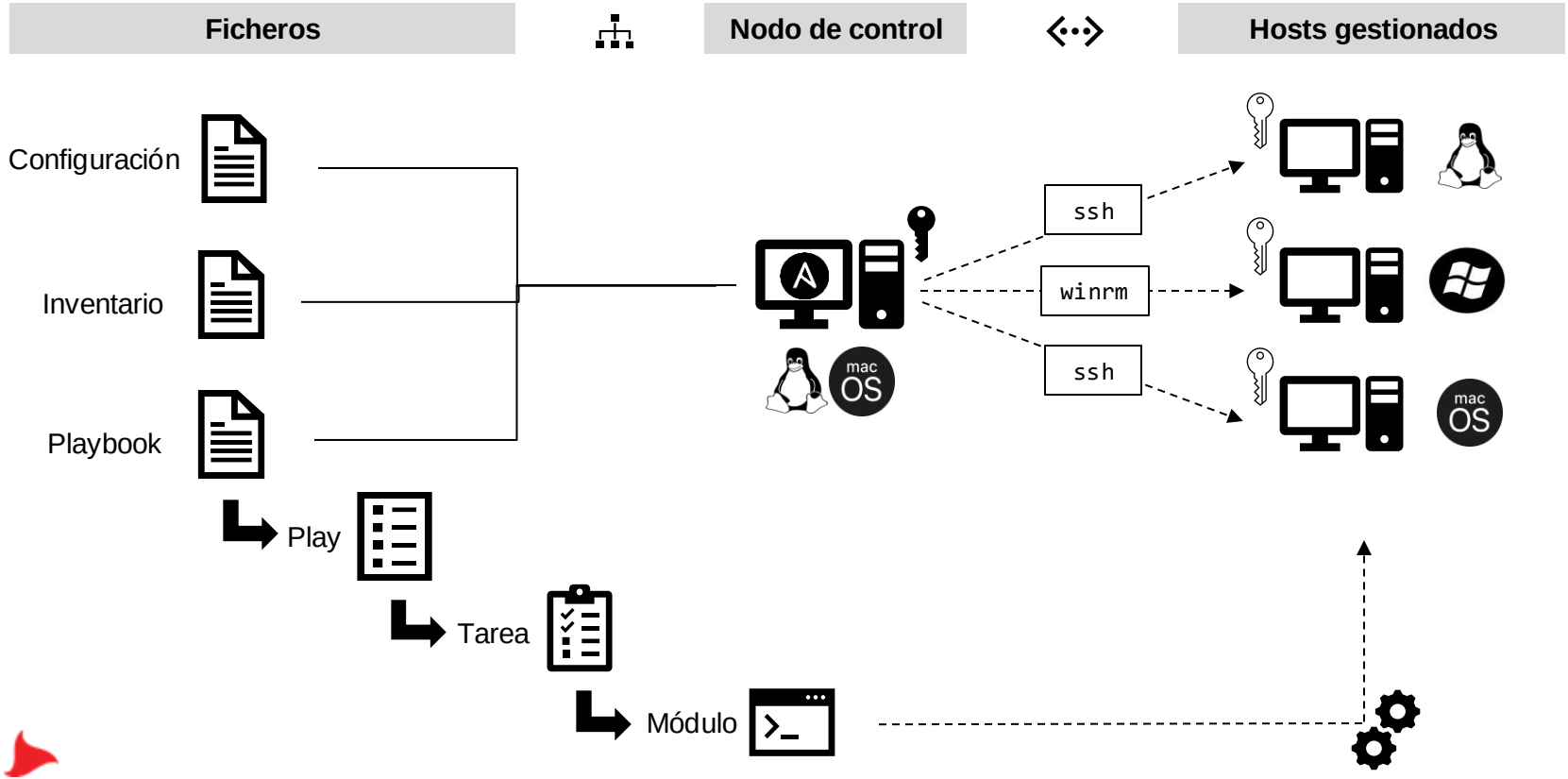
ansible-inventory Lista información relacionada con el inventario.

```
ansible-inventory [--list | --graph ] [--yaml]
```

ansible-doc Muestra la documentación asociada al módulo especificado.

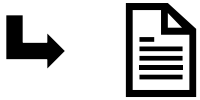
```
ansible-doc <module>
```

Arquitectura de Ansible



Configuración

Nodo de control



ansible.cfg

NOTA. Para usar tareas de Ansible que requieren privilegios elevados, se necesita que el `remote_user` tenga los permisos adecuados en el archivo `/etc/sudoers` o en algún archivo adicional de `/etc/sudoers.d/*`

Un **archivo de configuración** recoge en secciones la configuración por defecto que se aplicará al usar Ansible. Se ubica en puntos específicos de nuestro sistema de ficheros, donde Ansible los encontrará automáticamente.

```
[defaults]
inventory = inventory/hosts
remote_user = vagrant
host_key_checking = False
private_key_file = ~/.vagrant.d/insecure_private_key
roles_path = roles
timeout = 30

[privilege_escalation]
become = True
become_method = sudo
become_user = root
become_ask_pass = False
```

(mayor prioridad)

`ANSIBLE_CONFIG=path`

`./ansible.cfg`

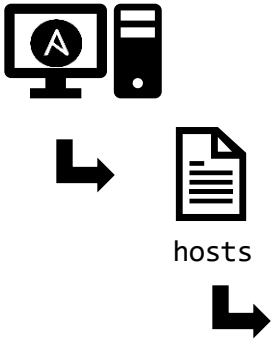
`~/.ansible.cfg`

`/etc/ansible/ansible.cfg`

(menor prioridad)

Inventario

Nodo de control



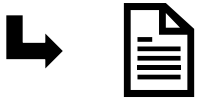
Un **inventario** define una colección de hosts que Ansible gestiona. Estos hosts pueden asignarse a grupos, los cuales pueden gestionarse colectivamente. Los grupos pueden contener subgrupos, y los hosts pueden ser miembros de múltiples grupos.

```
[db]
database ansible_host=192.168.56.101

[app]
app_instance_1 ansible_host=192.168.56.102
app_instance_2 ansible_host=192.168.56.103

[web]
load_balancer ansible_host=192.168.56.104

[back:children]
app
web
```

Hosts gestionados

Inventario

Nodo de control



hosts

```
[linux]
linux_host ansible_host=192.168.1.101

[windows]
windows_host ansible_host=192.168.1.102

[mac]
mac_host ansible_host=192.168.1.103

[all:vars]
ansible_user=admin_user
ansible_ssh_private_key_file=~/.ssh/id_rsa
ansible_python_interpreter=/usr/bin/python3

[linux:vars]
ansible_connection=ssh
ansible_become=true
ansible_become_method=sudo
ansible_become_password=linux_user_password

[windows:vars]
ansible_connection=winrm
ansible_user=Administrator
ansible_password=windows_password
ansible_winrm_transport=ntlm
ansible_port=5985

[mac:vars]
ansible_connection=ssh
ansible_become=true
ansible_become_method=su
ansible_become_password=mac_user_password
```

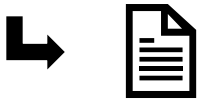
Hosts gestionados

Es posible definir variables de inventario asociadas a hosts o grupos de hosts con la partícula `:vars`.

Las variables aquí definidas tienen mayor prioridad que aquellas definidas en el `ansible.cfg`.

Playbooks

Nodo de control



`playbook.yml`

Un **playbook** es un archivo de texto que contiene una lista de uno o más plays que se ejecutan en un orden específico.

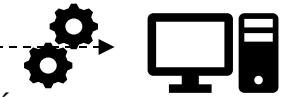
```
---
- name: Esto es un Play
  hosts: db
  become: true

  tasks:

    - name: Esto es una tarea
      ansible.builtin.copy:
        src: files/app
        dest: /opt/goapp/app
        owner: root
        group: root
        mode: '0755'
```

Un **play** es una secuencia de tareas que se aplican, en orden, a uno o más hosts seleccionados de tu inventario.

Una **tarea** es la aplicación de un módulo para realizar una unidad de trabajo específica.

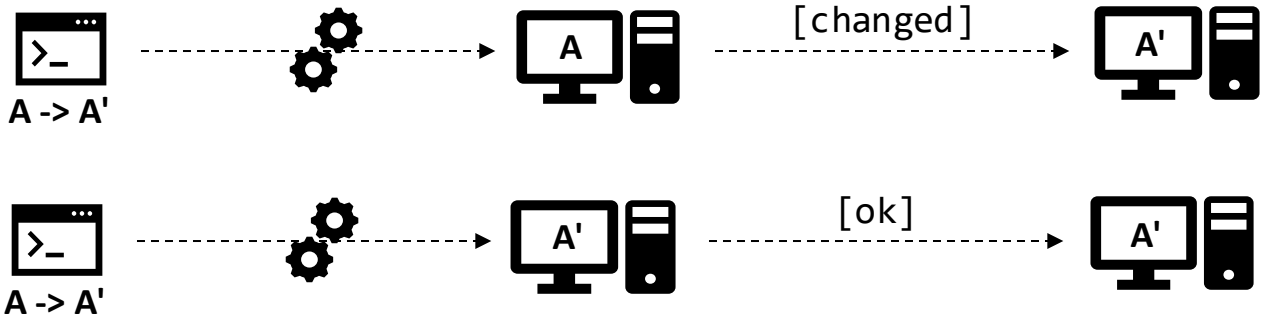
Un **módulo** es un ejecutable escrito en algún lenguaje y que opera sobre el sistema.

Hosts gestionados



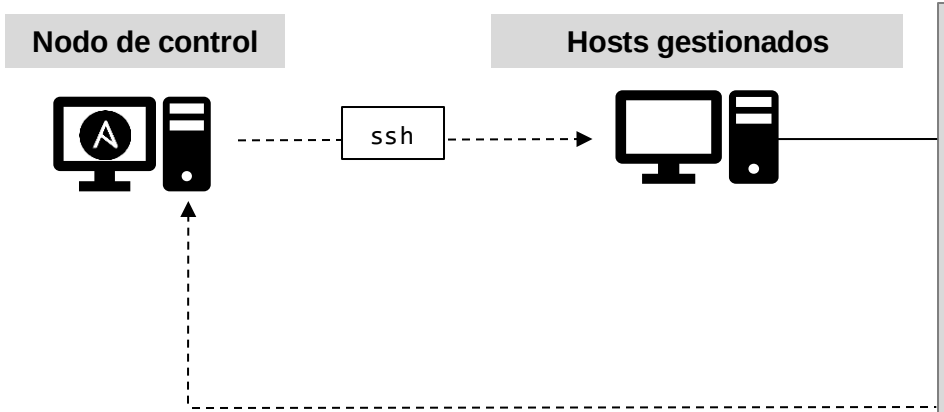
Playbooks > Idempotencia

La **idempotencia** es la propiedad por la cual la ejecución sucesiva de un recurso no altera el resultado más de una vez. En el contexto de Ansible, entendemos la idempotencia como la obtención del mismo estado final de un sistema independientemente del número de veces que ejecutemos un playbook sobre dicho sistema. La idempotencia depende del comportamiento de los módulos ejecutados en los playbooks y es una propiedad deseable en un proyecto de Ansible.



Playbooks > **Facts**

Los **facts** en Ansible son datos o información sobre los hosts gestionados que Ansible recopila automáticamente al ejecutar un playbook o una tarea. Esta información se utiliza para tomar decisiones dentro de las tareas y playbooks, adaptando su comportamiento según las características específicas de cada sistema. Los facts recopilados por Ansible se almacenan como variables en el diccionario especial `ansible_facts`. Estas variables están disponibles durante la ejecución del playbook y puedes usarlas directamente en las tareas.

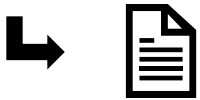



```
{
  "ansible_facts": {
    "ansible_architecture": "x86_64",
    "ansible_distribution": "Ubuntu",
    "ansible_distribution_version": "22.04",
    "ansible_distribution_release": "jammy",
    "ansible_memtotal_mb": 4096,
    "ansible_processor": [
      "Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz"
    ],
    "ansible_default_ipv4": {
      "address": "192.168.1.100",
      "netmask": "255.255.255.0",
      "network": "192.168.1.0",
      "gateway": "192.168.1.1"
    },
    …
}
```

Roles

Nodo de control

Un **rol** en Ansible es una forma de organizar y reutilizar el código en un proyecto de automatización. Los roles dividen las tareas relacionadas en archivos y directorios bien estructurados, facilitando la gestión, el mantenimiento y la reutilización de las automatizaciones en diferentes proyectos.



playbook.yml

```
▶ ansible-galaxy init <nombre-rol>
```

Play

db

```
- name: Este play usa un rol
  hosts: db
  become: true

  roles:
    - db
```

```
db/
    tasks/
        main.yml
    handlers/
        main.yml
    vars/
        main.yml
    defaults/
    files/
    templates/
    meta/
        main.yml
    tests/
  README.md
```

AGENDA 2030

Variables

Las **variables** en Ansible pueden definirse hasta en 22 lugares diferentes y ser utilizadas en cualquier cadena de caracteres con la sintaxis `{{ variable }}`. Algunos lugares de definición son:

- prioridad

Definidas en rutas por defecto

Definidas dentro de un playbook

Mapeadas desde ficheros

Definidas en tiempo de ejecución

Definidas por línea de comandos

+ prioridad

AGENDA 2030

Variables

Las **variables** en Ansible pueden definirse hasta en 22 lugares diferentes y ser utilizadas en cualquier cadena de caracteres con la sintaxis `{{ variable }}`. Algunos lugares de definición son:

- prioridad

Definidas en rutas por defecto

Definidas dentro de un playbook

Mapeadas desde ficheros

Definidas en tiempo de ejecución

Definidas por línea de comandos

+ prioridad

```
project/
  config.cfg
  inventory/
    hosts
  group_vars/
    nombre_del_grupo.yml
  host_vars/
    nombre_del_host.yml
  roles/
    rol/
      vars/
        main.yml
      defaults/
        main.yml
  README.md
```

AGENDA 2030

Variables

Las **variables** en Ansible pueden definirse hasta en 22 lugares diferentes y ser utilizadas en cualquier cadena de caracteres con la sintaxis `{{ variable }}`. Algunos lugares de definición son:

- prioridad

Definidas en rutas por defecto

Definidas dentro de un playbook

Mapeadas desde ficheros

Definidas en tiempo de ejecución

Definidas por línea de comandos

+ prioridad

```
---
- name: Play con variables definidas directamente
  hosts: localhost
  vars:
    app_name: "miapp"
    app_port: 8080
    db_user: "admin"
    db_password: "secreto123"

  tasks:
    - name: Mostrar el nombre de la aplicación
      ansible.builtin.debug:
        msg: "Mi app: {{ app_name }}"
```

Variables

Las **variables** en Ansible pueden definirse hasta en 22 lugares diferentes y ser utilizadas en cualquier cadena de caracteres con la sintaxis `{{ variable }}`. Algunos lugares de definición son:

- prioridad

Definidas en rutas por defecto

Definidas dentro de un playbook

Mapeadas desde ficheros

Definidas en tiempo de ejecución

Definidas por línea de comandos

+ prioridad

```
---
- name: Usar archivo externo
  hosts: all
  vars_files:
    - vars/variables.yml

  tasks:

    - name: Mostrar una variable
      ansible.builtin.debug:
        msg: "El nombre de la app es {{ app_name }}"
```

AGENDA 2030

Variables

Las **variables** en Ansible pueden definirse hasta en 22 lugares diferentes y ser utilizadas en cualquier cadena de caracteres con la sintaxis `{{ variable }}`. Algunos lugares de definición son:

- prioridad

Definidas en rutas por defecto

Definidas dentro de un playbook

Mapeadas desde ficheros

Definidas en tiempo de ejecución

Definidas por línea de comandos

+ prioridad

```
---
- name: Registrar variables
  hosts: localhost

  tasks:
    - name: Ejecutar un comando y registrar la salida
      ansible.builtin.command: uname -r
      register: kernel_version

    - name: Mostrar el resultado registrado
      ansible.builtin.debug:
        msg: "Kernel version: {{ kernel_version.stdout }}"
```

AGENDA 2030

Variables

Las **variables** en Ansible pueden definirse hasta en 22 lugares diferentes y ser utilizadas en cualquier cadena de caracteres con la sintaxis `{{ variable }}`. Algunos lugares de definición son:

- prioridad

Definidas en rutas por defecto

Definidas dentro de un playbook

Mapeadas desde ficheros

Definidas en tiempo de ejecución

Definidas por línea de comandos

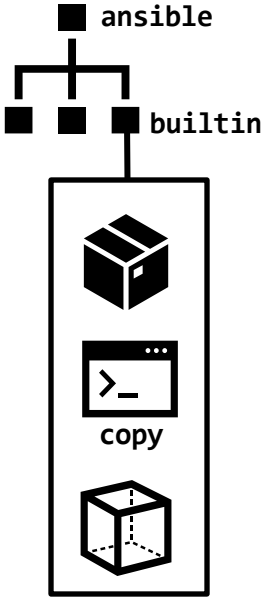
+ prioridad

```
ansible-playbook deploy.yml --extra-vars "app_name=miapp app_port=8080"
```

AGENDA 2030

Colecciones

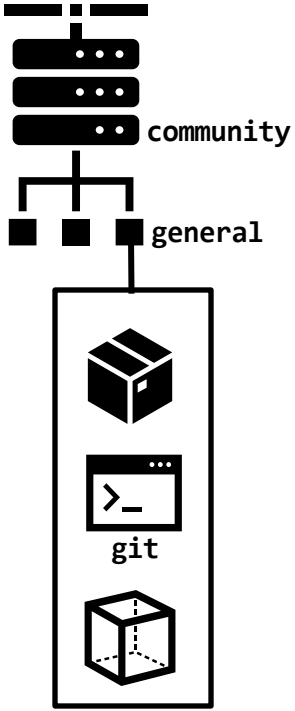
Una **colección** de contenido de Ansible proporciona un conjunto de módulos, roles y otros complementos relacionados que pueden usarse en los playbooks.




```
---
- name: Esto es un Play
  hosts: db
  become: true

  tasks:

    - name: Esto es una tarea
      ansible.builtin.copy:
        src: files/app
        dest: /opt/goapp/app
        owner: root
        group: root
        mode: '0755'
```

Colecciones

Si el recurso no está incluido en Ansible por defecto, debemos descargarlo de la Ansible Galaxy:




```
ansible-galaxy collection install community.general
```

```
---
- name: Esto es un Play
  hosts: db
  become: true

  tasks:

    - name: Esto es una tarea
      community.general.git:
        repo: https://.../ansible-examples.git
        dest: /home/
        version: main
    become: true
```

Demo

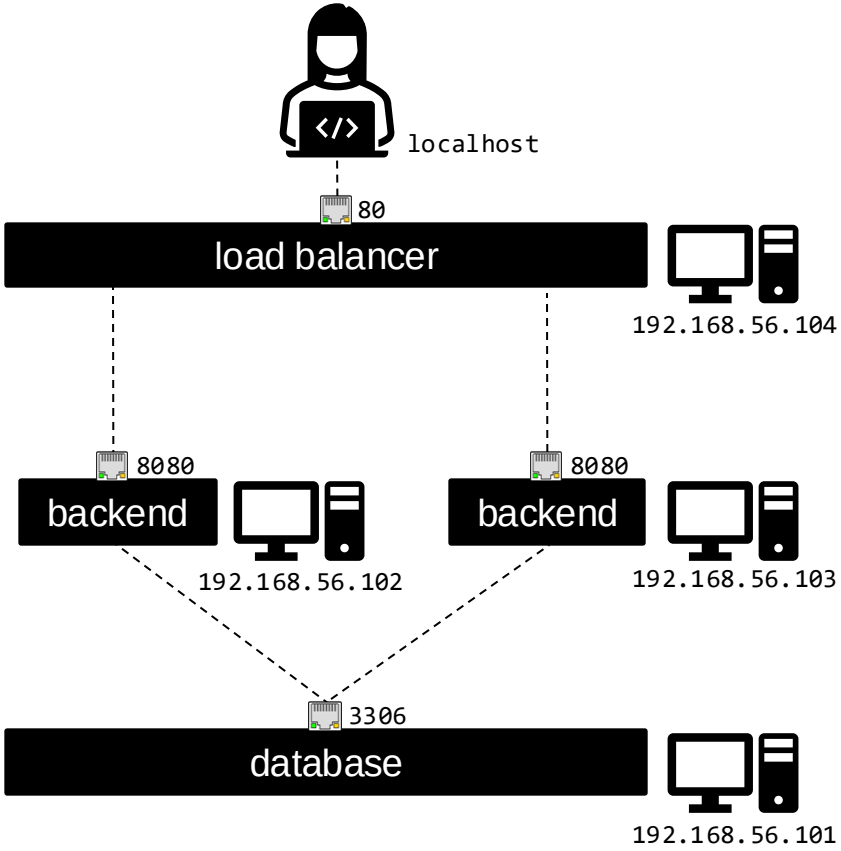
La siguiente demo tiene como objetivo desplegar una aplicación sencilla en Golang sobre la arquitectura de la imagen de la derecha.

La aplicación ya está compilada y consiste en un servicio que expone dos *endpoints*:

- Un `GET` a `/` nos permite consultar la lista de datos almacenados en una tabla de mariadb.
- Un `POST` a `/save` nos permite crear una nueva entrada en la base de datos con el *timestamp* de la petición.

El despliegue estará automatizado con Ansible. La creación de las máquinas virtuales estará automatizada con Vagrant.

https://github.com/Gradiant/SI-CLOUD-ansible-workshop.git



Documentación

Ansible - https://docs.ansible.com/

Vagrant - https://developer.hashicorp.com/vagrant/docs

Virtual Box - https://www.virtualbox.org/manual/

Curl - https://curl.se/

OpenSSH - https://www.openssh.com/manual.html

Gracias por la atención