Introducción a las Aplicaciones Web

Ansible

Ansible es una herramienta *open source* para **configurar** y **administrar** una o muchas computadoras.

Gestión de la configuración: La información que describe el software y hardware de una empresa.

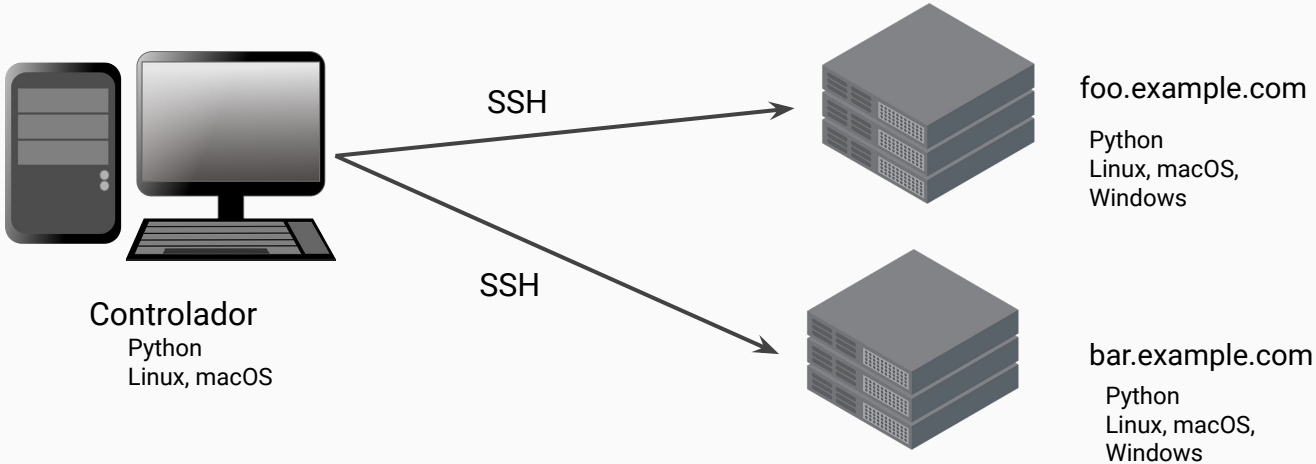
- Control de versiones.
- Actualizaciones a los paquetes de software instalados.
- Direcciones de red de los dispositivos de hardware.
- Instalación y configuración de software y componentes.

Inventario

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

Scripts: Playbook

```
- name: Create database

community.postgresql.postgresql_db:
    name: "{{ app_name }}"
  become: yes
  become_user: postgres
```



Images by OpenClipart-Vectors from Pixabay

Módulos

- Unidades de trabajo autosuficientes en Ansible.
- Son escritos en lenguajes de scripts, como Python, Perl, Ruby, Bash, etc.
- Son idempotentes.

Inventario

- Describe los nodos que pueden ser accedidos por Ansible.
- Los nodos pueden asignarse a grupos.
- Indican la ubicación de las claves utilizadas por la conexión SSH.

Inventario

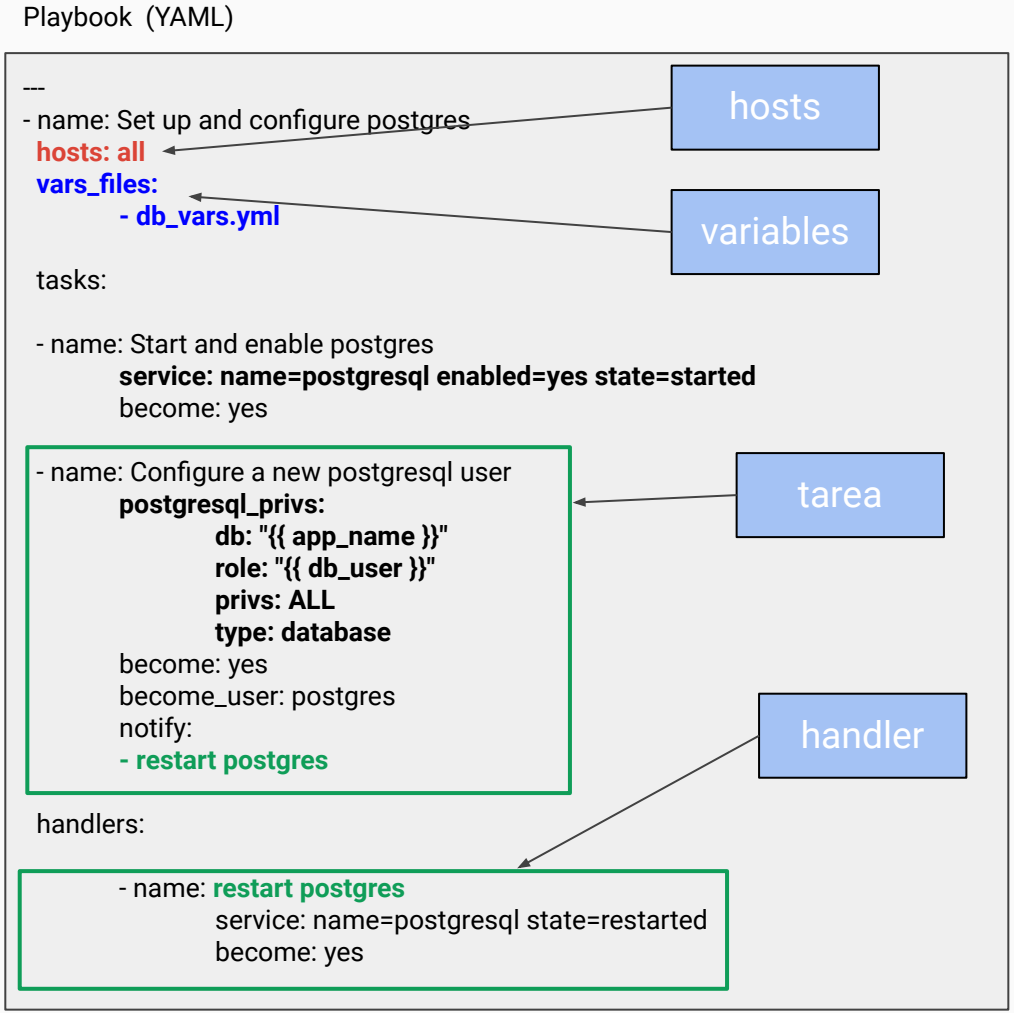
```
46.231.22.122

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

Playbook (YAML)

```
---
- name: Set up and configure postgres
  hosts: all
  vars_files:
        - db_vars.yml

  tasks:

  - name: Start and enable postgres
        service: name=postgresql enabled=yes state=started
        become: yes

  - name: Configure a new postgresql user
        postgresql_privs:
              db: "{{ app_name }}"
              role: "{{ db_user }}"
              privs: ALL
              type: database
        become: yes
        become_user: postgres
        notify:
        - restart postgres

  handlers:
        - name: restart postgres
              service: name=postgresql state=restarted
              become: yes
```

Playbooks

- Cada libro de jugadas (playbook) contiene una lista de tareas.
- Las tareas son ejecutadas en orden, contra cada máquina que encaja con el patrón del host, para luego seguir con la próxima tarea.
- El objetivo de un Playbook el mapear un grupo de host a una lista de tareas.
- Los hosts donde fallen las tareas son sacados de la rotación de las jugadas restantes.
- El objetivo de cada tarea es ejecutar un módulo, con parámetros muy específicos.
- Se pueden usar variables utilizando el concepto de plantillas.
- Cada tarea debe tener un nombre, este se muestra mientras el Playbook se ejecuta.



Roles

- Los roles organizan en una estructura de directorios, los archivos necesarios para ejecutar uno o más playbooks asociados al rol.
- Los nombres son estándar.
- Se debe incluir por lo menos un directorio.
- Se pueden utilizar a nivel de tareas, pero lo clásico es a nivel del playbook:

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

```
roles/
  common/               # Esta jerarquía representa un "rol"
    tasks/              #
      main.yml          # playbook principal, puede incluir otros archivos.
    handlers/           #
      main.yml          # archivo con los handlers utilizados en el rol.
    templates/          #
      ntp.conf.j2       #  plantillas en formato Jinja 2
    files/              #
      bar.txt           #  archivos que se pueden enviar a los nodos
      foo.sh            #  scripts utilizados por el recurso
    vars/               #
      main.yml          #  variables asociadas al rol
    defaults/           #
      main.yml          # variables por defecto
    meta/               #
              main.yml  # dependencias del rol
    library/            # se pueden incluir módulos
    module_utils/       # utilerías
    lookup_plugins/     # plugins

  webtier/              # otro rol
  monitoring/           # un rol más
  fooapp/               # último rol
```

Estructura de un despliegue

- El playbook inicial se encuentra en la raíz del folder del despliegue.
- El inventario se encuentra en el directorio **`hosts`**.
- El directorio **`group_vars`** incluye las variables que se utilizan por todos los grupos.
- Se incluye el directorio **`roles`** con la estructura estándar.
- En un folder externo almacenamos las claves secretas.

```
├── developer
│   ├── deploy.yml
│   ├── group_vars
│   │   └── all
│   ├── hosts
│   └── roles
│       └── developer
│           ├── vars
│           ├── tasks
│           └── templates
└── ssh-keys
    ├── mi_llave.pem
    └── mi_llave2.pem
```

```
# deploy.yml

- name: Configuración para desarrollo en django
  hosts: all
  user: ubuntu
  roles:
    - developer
```

Ejemplo

Seguiremos los siguientes pasos:

- Arranca una instancia en AWS, Ubuntu 22.04, instancia small, crea un **juego de claves (key-pair)** en este ejemplo lo nombraremos como: **mi-llave**.
- Descarga el archivo **.pem** asignada en el lanzamiento.
- Recuerda que la clave es secreta.

Vamos a configurar la instancia recién creada utilizando Ansible

- En caso de estar en una computadora con Windows no podemos ejecutar Ansible localmente.
  - En este ejemplo asumimos que estamos en Windows
  - Como alternativa vamos a utilizar CodeSpaces de GitHub.
  - Crea un nuevo repo a partir de la plantilla: https://github.com/mariosky/django-playbook.
  - Lanza un CodeSpace en tu copia del repositorio.
- Sube tu archivo **.pem** al directorio ssh-keys.

Prueba conectarte a tu instancia:

```
/workspaces/django-playbook (main) $ chmod 400 ssh-keys/mi-llave.pem
/workspaces/django-playbook (main) $ ssh -i ssh-keys/mi-llave.pem ubuntu@<ip de tu instancia>
```

Ejemplo

Si te pudiste conectar…

Desconéctate, **vamos a instalar Ansible en el controlador** (en este caso el CodeSpace)

```
curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py
python3 get-pip.py --user
python3 -m pip install --user ansible

En caso de este warning:
WARNING: The scripts ansible, ansible-config, ansible-connection, ansible-console,
ansible-doc, ansible-galaxy, ansible-inventory, ansible-playbook, ansible-pull and
ansible-vault are installed in '/home/ubuntu/.local/bin' which is not on PATH.
Exportamos la variable y revisamos que ya se hizo
export PATH="/home/ubuntu/.local/bin:$PATH"
env | grep PATH

ansible --version
```

Ejemplo

Vamos a probar Ansible para enviar un comando a la instancia:

Debemos agregar un archivo llamado hosts en la raíz del directorio developer:

```
web ansible_host=<ip> ansible_user=ubuntu ansible_port=22 ansible_ssh_private_key_file=../ssh-keys/mi-llave.pem
```

```
@mariosky ➜ /workspaces/django-playbook/developer $ ansible -i hosts all -m ping
web | SUCCESS => {
        "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
        },
        "changed": false,
        "ping": "pong"
}
```

Ejemplo

Vamos a probar Ansible para enviar un comando a la instancia:

Debemos agregar un archivo llamado hosts en la raíz del directorio developer:

```
[web]
host1  ansible_host=<ip> ansible_user=ubuntu ansible_port=22 ansible_ssh_private_key_file=../ssh-keys/mi-llave.pem
```

```
@mariosky ➜ /workspaces/django-playbook/developer $ ansible -i hosts all -m ping
web | SUCCESS => {
        "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
        },
        "changed": false,
        "ping": "pong"
}
```

Cambia all por:

web

webs

host1

¿Qué pasa?
Recuerda Grupos, patrones.

Ejemplo

Podemos utilizar YAML para el inventario
Agregar un archivo llamado hosts.yml en la raíz del directorio developer:

```yaml
web:
  hosts:
    host1:
      ansible_host: 54.236.46.151
      ansible_user: ubuntu
      ansible_port: 22
      ansible_ssh_private_key_file: /workspaces/django-playbook/ssh-keys/mi-llave.pem
```

Ejemplo

Ejecutemos un playbook para actualizar el server

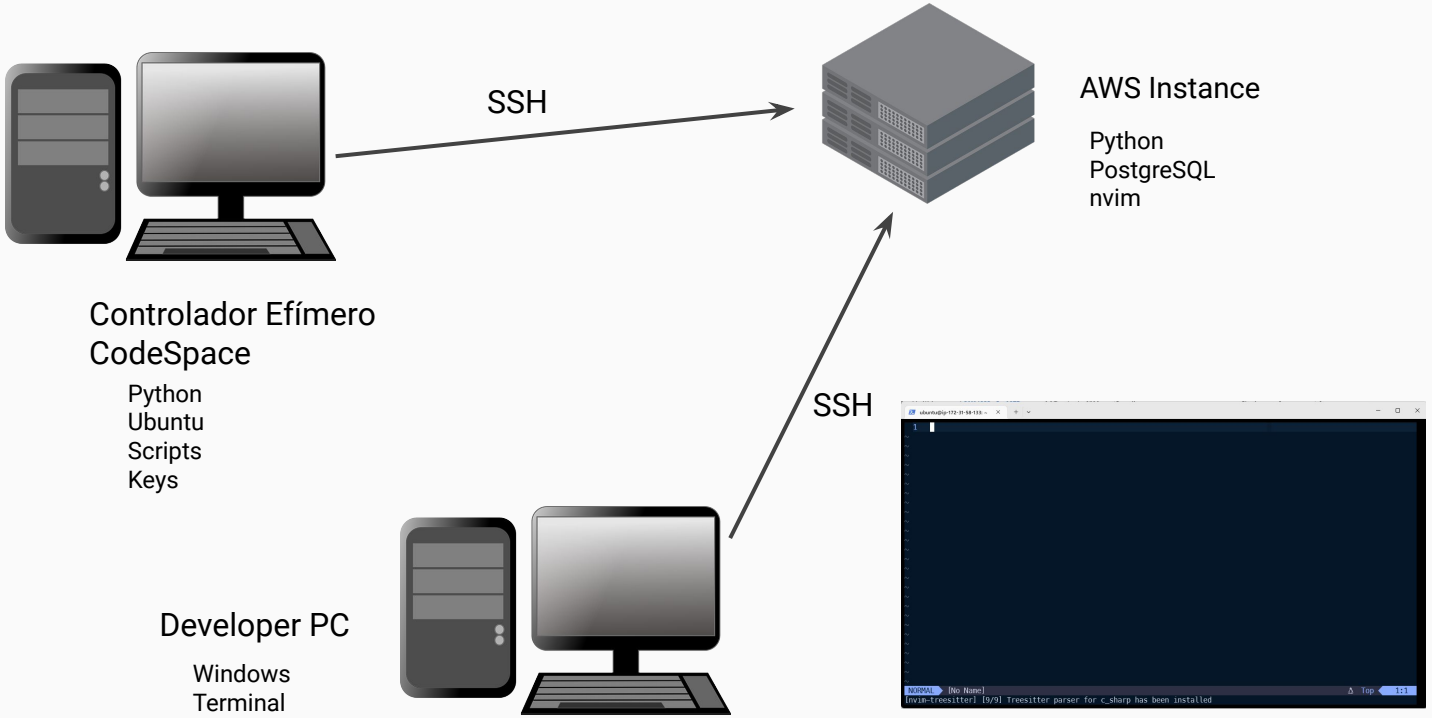
```
@mariosky ➜ /workspaces/django-playbook/developer/play (main) $ ansible-playbook developer.yml -i ../hosts

PLAY [Update Server and Install Dependencies]
**********************************************************************************************
***

TASK [Gathering Facts]
**********************************************************************************************
**************
ok: [host1]

TASK [Add the user developer]
**********************************************************************************************
ok: [host1]

TASK [Upgrade]
**********************************************************************************************
ok: [host1]

PLAY RECAP **************************
host1                      : ok=8 changed=1    unreachable=0      failed=1     skipped=0    rescued=0    ignored=0
```

Estructura hasta el momento

```
@mariosky ➜ /workspaces/django-playbook/developer (main) $ tree
.
├── deploy.yml
├── group_vars
│   └── all
├── hosts
├── hosts.yml
├── play
│   ├── db_vars.yml
│   ├── developer.yml
│   ├── neovim_setup.yml
│   ├── postgres.yml
│   └── server.yml
└── roles
    └── common
        ├── handlers
        ├── tasks
        │   └── main.yml
        └── vars
            └── main.yml
```

```
ansible-playbook deploy.yml -i hosts
```

Estructura hasta el momento



Otras fuentes (inglés)

Módulos de Ansible
https://docs.ansible.com/ansible/latest/collections/index.html

Ejemplo de un despliegue a producción de una app de Django con Ansible y Fabric.
Automating Django Deployments with Fabric and Ansible – Real Python

Tutorial para principiantes.
Ansible Tutorial for Beginners: Playbook & Examples

Wikipedia en español.
https://es.wikipedia.org/wiki/Ansible_(software)