Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ciencias y Sistemas
Software Avanzado
Laboratorio

# Funcionamiento de Ansible

Feliciano Ernesto Franco Lux
200915532

# Funcionamiento de Ansible

Se utilizarán dos ambientes (local y remoto EC2) para generar el `playbook`, el equipo local ayudará a economizar recursos ya que los recursos locales (Docker container) permitirán hacer pruebas sin consumir la capa gratuita de AWS. A través del archivo local se podrá generar el remoto de EC2.

# Playbooks

Se utiliza el estándar opcional para archivos YAML en el que todos los archivos empiezan por tres guiones "---" y terminan en tres puntos "...".

## Local

Archivo: ansible/nginx_from_local.yml

Previo a generar un archivo `*.yml`, se levantó un contenedor de Docker con Ubuntu 22.04 y ahí se hizo una instalación normal de Nginx, es decir, la ejecución de paquetes con `apt`. Los detalles se pueden visualizar en el archivo Docker.md.

Utilizando el orden y los comando ejecutados en el contenedor de docker se generó el archivo **nginx_from_local.yml**. Se utilizó la documentación oficial del listado de módulos de Ansible:

```
---
- hosts: ubuntu_ssh_i
  tasks:
  - name: Actualizar repositorios -> "apt update"
    apt:
        update-cache: yes
  - name: Instalar Nginx -> "apt install nginx"
    apt: name=nginx state=present
  - name: Copiar sitio web React+Vite
    copy:
        src: /home/feli/Documentos/docker_volumes/dist/
        dest: /var/www/sa_practica1
  - name: Crear archivo de configuración en Nginx para el sitio web copiado
    copy:
        dest: /etc/nginx/sites-enabled/sa_practica1
        content: "server {
                        listen 80;
                        listen [::]:80;

                        #server_name example.ubuntu.com;

                        root /var/www/sa_practica1;
                        index index.html;

                        location / {
                                try_files $uri $uri/ =404;
                        }
                }"
```

```
  - name: Eliminar archivo 'default' en sites-enabled para eliminar el sitio
predeterminado de Nginx
    ansible.builtin.file:
      path: /etc/nginx/sites-enabled/default
      state: absent
  - name: Reiniciar servicio de Nginx
    service: name=nginx state=started
...
```

En donde:

1. **hosts**: `ubuntu_ssh_i` es el alias que le daremos al host en donde se ejecutará el archivo.
2. **Actualizar repositorios -> "apt update"**: Utiliza el módulo `apt` con la propiedad `update-cache: yes` para que actualice los repositorios.
3. **Instalar Nginx -> "apt install nginx"**: Instala Ngix a través del módulo `apt` con las propiedades de nombre del paquete y la versión.
4. **Copiar sitio web React+Vite**: Copiar el código fuente desde el equipo local hacia el remoto a través del módulo copy.
5. **Crear archivo de configuración en Nginx para el sitio web copiado**: Utilizar el módulo copy para crear el nuevo archivo.
6. **Eliminar archivo 'default' en sites-enabled para eliminar el sitio predeterminado de Nginx**: Eliminar el archivvo con el módulo file
7. **Reiniciar servicio de Nginx**: Iniciar servicios con el módulo service

# EC2

Archivo: ansible/nginx_from_local.yml

A diferencia del archivo local, el archivo remoto tuvo ligeras modificaciones. La estructura de instrucciones final quedo así:

```
---
- hosts: nginx_ec2
  become: yes
  tasks:
  - name: Actualizar repositorios -> "apt update"
    apt:
        update-cache: yes
  - name: Instalar Nginx -> "apt install nginx"
    apt: name=nginx state=present
  - name: Copiar sitio web React+Vite
    copy:
        src: /home/ubuntu/
        dest: /var/www/sa_practica1
```

```
        remote_src: true
  - name: Crear archivo de configuración en Nginx para el sitio web copiado
    copy:
        dest: /etc/nginx/sites-enabled/sa_practica1
        content: "server {
                        listen 81;
                        listen [::]:81;

                        #server_name example.ubuntu.com;

                        root /var/www/sa_practica1;
                        index index.html;

                        location / {
                                try_files $uri $uri/ =404;
                        }
                }"
  - name: Reiniciar servicio de Nginx
    service: name=nginx state=reloaded
...
```

Diferencias:

1.**hosts**: `nginx_ec2` es el nuevo alias que se utilizará. Ahora se tiene el atributo `become: yes` que indica que las instrucciones se ejecutarán con permisos de *super usuario*.

2.**Copiar sitio web React+Vite**: Se modifican los paths y ahora se utiliza el atributo `remote_src: true` que indica que las copias serán de remoto a remoto, no local a remoto como en el archivo anterior.

3.**Crear archivo de configuración en Nginx para el sitio web copiado**: Se modifica el número de puerto que se utilizará en EC2.

4.**Eliminar archivo 'default' en sites-enabled para eliminar el sitio predeterminado de Nginx**: Como se modificó el número de puerto, ahora no es necesario eliminar el archivo 'default'.

5.**Reiniciar servicio de Nginx**: Ahora se utiliza el atributo `state=reloaded`.

# Ejecutar playbook

Las pruebas de Ansible se harán en local, los resultados de las pruebas remotas se visualizarán en el terminal que ejecuta Terraform.

## Local

Se crea un archivo de hosts con el alias y las variables. El nombre del archivo es `hosts`, pero puede ser otro, y el contenido es el siguiente:

```
ubuntu_ssh_i ansible_host=172.17.0.2 ansible_connection=ssh ansible_user=root
ansible_password=pass
```

En el comando que se ejecutará se indica el archivo de `hosts` que se utilizará, en él se encuentra la IP de nuestro contenedor.

```
$ ansible-playbook nginx_from_local.yml -i hosts
```

En esta instrucción no se incluyó la `private key` porque solo era un contenedor local. Para hacer una conexión SSH hacia un contenedor se deben seguir las instrucciones de la sección **Docker SSH** que se encuentran al final de este archivo.

## EC2

En el archivo terraform/main.tf se encuentra escrito el comando a ejecutar en EC2, dentro del `resource "aws_instance" "ansible" {}` se encuentra un `provisioner "remote-exec" {}` con el comando:

```
ansible-playbook --private-key key_pair nginx_from_ec2.yml -i hosts
```

En donde –`private-key` es el parámetro que solicita la llave privada.

# Ansible en línea de comandos

### Inventario

Ver el inventario global

```
$ cat /etc/ansible/hosts
```

Ejecutamos el `módulo (-m)` ping para verificar. *Comando ad-hoc*.

```
$ ansible localhost -m ping
```

### Hosts locales

Creamos un archivo `hosts`

```
$ touch hosts
```

Agregamos el contenido al archivo en formato `INI`

```
localhost
192.168.0.9
```

Utilizamos el módulo ping con el argumento `-m para módulos`. *Comando ad-hoc*.

```
$ ansible 192.168.0.9 -m ping -i hosts
```

Ejecutar el comando literal, es decir, ad-hoc, en el servidor. Cuando no se coloca el argumento **-m** en la instrucción, ansible ejecuta el módulo predeterminado Shell.

```
$ ansible localhost -a 'echo hola'

$ ansible localhost -a 'ls /'

$ ansible localhost -a 'uname'
```

Si se quisiera instalar **vim** en una distribución que utilice paquetería **apt** entonces la siguiente instrucción se podría utilizar en ansible

```
$ ansible localhost -m apt -a 'name= vim state=present' -b -K
```

el parámetro `-b` indica que se utilizará el usuario **root* y `-K` es para que solicite la contraseña en la terminal

# Docker SSH

Creamos el nuevo contenedor con dos puertos abiertos y corriendo en background.

```
$ docker run -it --name ubuntu_ssh -p 8080:80 -p 2200:22 -d ubuntu
```

Accedemos al nuevo contenedor

```
$ docker exec -it ubuntu_ssh bash
```

Actualizamos los repositorios

```
# apt update
```

Instalamos el servidor OpenSSH

```
# apt install openssh-server
```

Verificamos el estado de los servicios, especialmente el del SSH.

```
# service --status-all
```

Notamos que el servicio se encuentra apagado inicialmente, antes de encenderlo se debe cambiar la contraseña del `root` del contenedor para que se pueda acceder desde SSH. Para este ejemplo utilizaremos el valor *"pass"* como contraseña:

```
# passwd root
```

También hay que editar el documento de acceso de SSH para que se pueda acceder con `root`. Se puede instalar *nano* en el contenedor para editar el documento, se asumirá que ya está instalado:

```
# nano /etc/ssh/sshd_config
```

En el archivo buscar la sección `Authentication`, en esa sección buscar el parámetro `PermitRootLogin`, colocar el valor `Yes` para que se pueda iniciar sesión SSH con el usuario root. Se recomienda agregar una nueva línea debajo de la que está comentada en caso que se desee regresar al valor predeterminado:

```
# Authentication:

#PermitRootLogin prohibit-password
PermitRootLogin Yes
```

Ahora sí encender el servicio SSH:

```
# service ssh start
```

Verificamos nuevamente el estado de los servicios y notamos que SSH se encuentra encendido.

```
# service --status-all
```

Desde otra consola probar conexión:

```
ssh root@localhost -p 2200
```

**Error en cambio de huella**. Es posible que al ejecutar el anterior comando en la terminal aparezca un *warning* con el siguiente mensaje: `WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!`. Esto ocurre porque el host(contenedor) que se está usando es distinto al que tiene registrado SSH. Se corrige con el siguiente comando que elimina las entradas de ese [host:puerto] del archivo .ssh/known_hosts:

```
$ ssh-keygen -R [localhost]:2200
o
$ ssh-keygen -R 172.17.0.2
```

Aceptamos el fingerprint y listo.

En este punto nuestro contenedor puede ser accedido desde un cliente SSH, eso quiere decir que podríamos ejecutar un comando Ansible y obtendríamos una respuesta. Antes debemos agregar la IP de nuestro contenedor al archivo de hosts global o local.

Para obtener la IP de nuestro contenedor se puede utilizar el siguiente comando, si queremos toda la información se debe retirar el argument `-f` y el formato entre comillas.

```
$ docker inspect -f "{{ .NetworkSettings.IPAddress }}" [container-name-or-id]
```

Archivo de hosts global o inventario global

```
nano /etc/ansible/hosts
```

Utilizaremos un inventario(hosts) local para ejecutar nuestras instrucciones Ansible, para eso creamos el archivo `hosts` y le agregamos la IP de nuestro contenedor en formato `INI`

```
ubuntu_ssh_i ansible_host=172.17.0.2 ansible_connection=ssh ansible_user=root
ansible_password=pass
```

Donde `ubuntu_ssh_i` (alias de inventario) y las demás variables servirán para la autenticación SSH.

Ejecutar un comando ad-hoc de ansible

```
$ ansible ubuntu_ssh_i -m ping -i hosts
```

El parámetro `-i` hace referencia a *Inventory* (hosts). El comando ad-hoc de Ansible fue ejecutado correctamente