

Desafío 7 - Bootcamp Devops Engineer

Alumno: Ignacio Peretti

Fecha de entrega - 29/07/2024

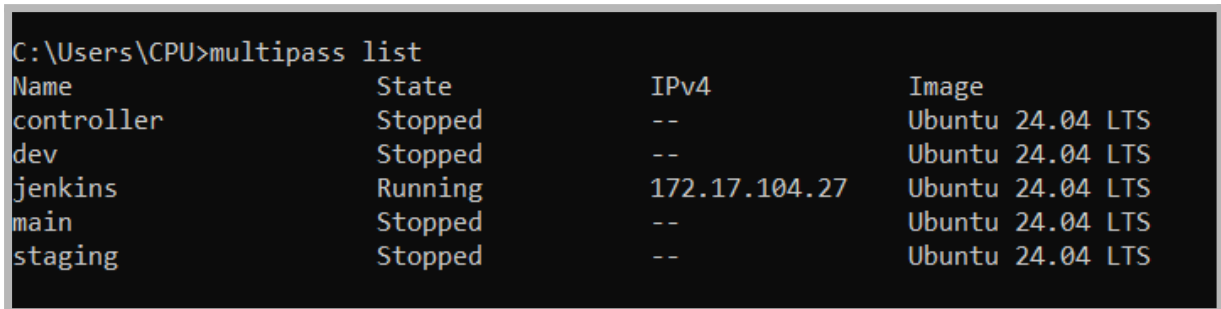
Requisitos:

1. Implementar un pipeline en jenkins para el proyecto visto en el desafío #6.
2. Crear job de jenkins por cada entorno de trabajo y que se ejecute desde su respectiva branch de github [dev,staging,main].
3. Ajustar el inventario para que pueda soportar un conjunto de máquinas para cada entorno.
4. Integrar el webhook de github para que cada vez que se produzca un cambio en el repositorio se ejecute el job del playbook.

Crear instancias

Trabajaremos con 5 instancias.

El controlador, una instancia llamada jenkins en la que estará jenkins corriendo y tendremos una instancia por cada ambiente, dev, staging y main.



Name	State	IPv4	Image
controller	Stopped	--	Ubuntu 24.04 LTS
dev	Stopped	--	Ubuntu 24.04 LTS
jenkins	Running	172.17.104.27	Ubuntu 24.04 LTS
main	Stopped	--	Ubuntu 24.04 LTS
staging	Stopped	--	Ubuntu 24.04 LTS

Vamos a necesitar convertir nuestro controlador en un agente de jenkins por lo que ejecutaremos los siguientes comandos en nuestra instancia controller.

Instalamos Java.

```
sudo apt update
```

```
sudo apt install default-jre
```

Agregar el usuario de Jenkins

```
sudo adduser jenkins
```

Crear directorio de trabajo para Jenkins

```
sudo mkdir /var/lib/jenkins
```

```
sudo chown jenkins:jenkins /var/lib/jenkins
```

Crear ssh key pública y privada

```
sudo su - jenkins
```



```
ssh-keygen
```

Configurar la clave pública en el controlador de Ansible

```
cat ~/.ssh/id_ed25519.pub >> ~/.ssh/authorized_keys
```

Crear una nueva credencial en el controlador de Jenkins, con el usuario Jenkins y la clave privada.

Credentials that should be available irrespective of domain specification to requirements matching.

ID	Name	Kind	Description	
 ansible-controller-key	jenkins (ansible-controller-key)	SSH Username with private key	ansible-controller-key	

Crear un nuevo nodo en Jenkins, con el nombre de nuestro controlador de Ansible.

Es importante a la hora de asignar la etiqueta que pongamos el mismo nombre con el que luego lo llamaremos en nuestro pipeline.

New node

Nombre del nodo

Type

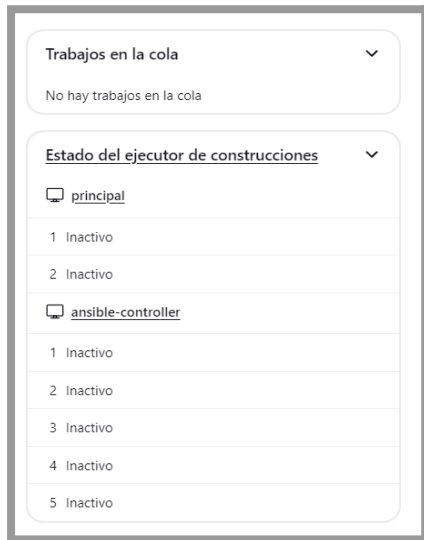
☒ Permanent Agent

Añadir un agente permanente a Jenkins. Es llamado 'permanente' porque Jenkins no provee ningún tipo de integración de alto nivel con estos agentes, como pueda ser aprovisionamiento dinámico. Selecciona este tipo si no hay ningún otro tipo mas adecuado. Por ejemplo cuando se añaden maquinas físicas o virtuales gestionadas desde fuera de Jenkins, etc.

Create

Testear la conexión controlador Jenkins con el Agente.

Para esto construimos una tarea llamada `ansible-test` y comprobamos la conexión.



S	W	Nombre ↓	Último Éxito	Último Fallo	Última Duración
✓	☀	ansible-test	2 Min 14 Seg #1	N/D	16 Seg

Una vez configurado nuestro agente, conectaremos las instancias mediante SSH.

Para ello compartiremos la clave pública creada anteriormente con el comando `ssh-keygen`, para ver esta clave podemos ejecutar el comando **`cat .ssh/id_ed25519.pub`**.

Copiamos el contenido de nuestra llave pública y la pegamos en el archivo **`authorized_keys`** de las instancias dev, staging y main.

Para editar el archivo `authorized_keys` podemos ejecutar el comando **`vi .ssh/authorized_keys`**

Compruebo que las llaves se añadieron correctamente.

```
ubuntu@controller:~$ ssh-copy-id ubuntu@172.17.104.33
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/ubuntu/.ssh/id_ed25519.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: WARNING: All keys were skipped because they already exist on the remote system.
(if you think this is a mistake, you may want to use -f option)

ubuntu@controller:~$ ssh-copy-id ubuntu@172.17.100.59
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/ubuntu/.ssh/id_ed25519.pub"
The authenticity of host '172.17.100.59 (172.17.100.59)' can't be established.
ED25519 key fingerprint is SHA256:9QgGoXyGyIJrF2F0MqX6Ys93s2Ln7RW0PiaKVWmhOV4.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: WARNING: All keys were skipped because they already exist on the remote system.
(if you think this is a mistake, you may want to use -f option)

ubuntu@controller:~$ ssh-copy-id ubuntu@172.17.102.22
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/ubuntu/.ssh/id_ed25519.pub"
The authenticity of host '172.17.102.22 (172.17.102.22)' can't be established.
ED25519 key fingerprint is SHA256:jNezx93d15a03M1M2i1LiKFZq3i/mBp8AHupCcZFrPg.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: WARNING: All keys were skipped because they already exist on the remote system.
(if you think this is a mistake, you may want to use -f option)
```

Completado esto tendremos conexión mediante SSH con las otras instancias.

Creación de un pipeline multibranch

Accederemos a jenkins y procederemos a crear una nueva tarea.

Elegiremos la opción Multibranch Pipeline, en mi caso lo voy a llamar **ansible-jenkins**.

Configuramos nuestro pipeline

En la opción Branch Source seleccionamos la opción GIT, agregamos la dirección de nuestro repositorio y en el caso de ser privado seleccionamos la credencial correspondiente.

En Build Configuration colocamos la dirección del nuestro archivo Jenkinsfile.

El resto de la configuración podemos dejarla como esta.

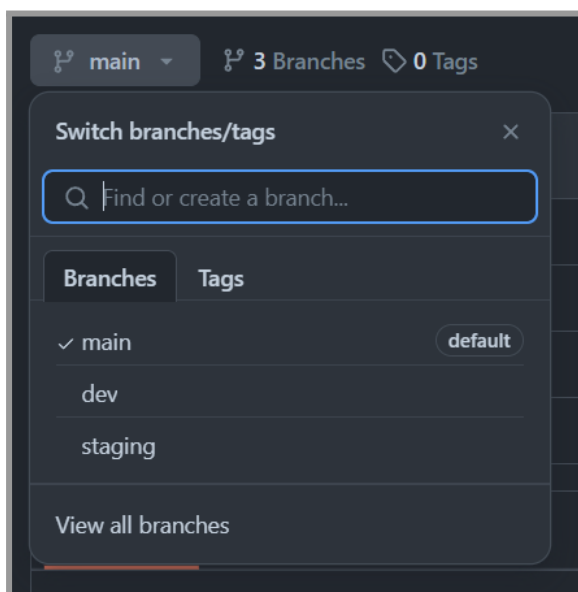
Creación de un repositorio en GitHub

A continuación tendremos nuestras instancias corriendo, nuestro pipeline multi-branch y nuestro controlador listo.

Lo que haremos ahora es crear un repositorio en github con las branches que definimos en nuestro pipeline. (main - dev - staging)

Un ejemplo del como quedo mi repositorio (<https://github.com/IgnacioPeretti/Desafio-7.git>)

Las 3 branches creadas.

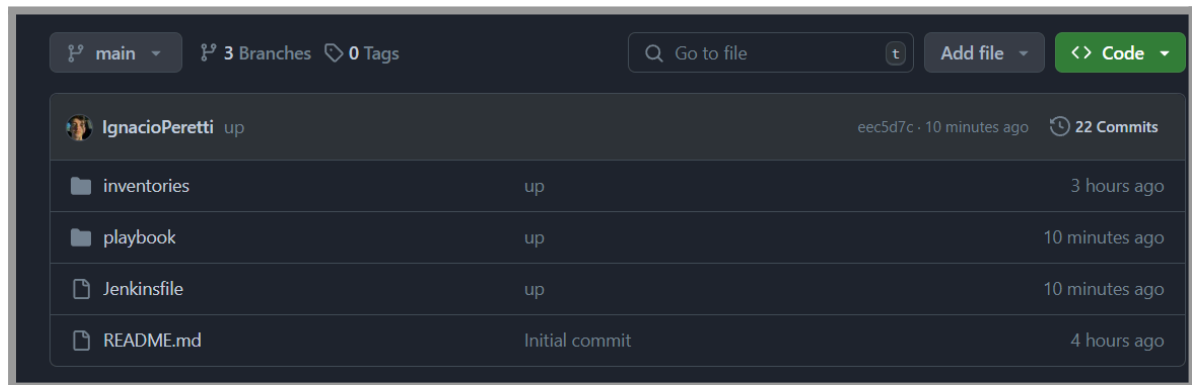


Lo que contiene mi repositorio es :

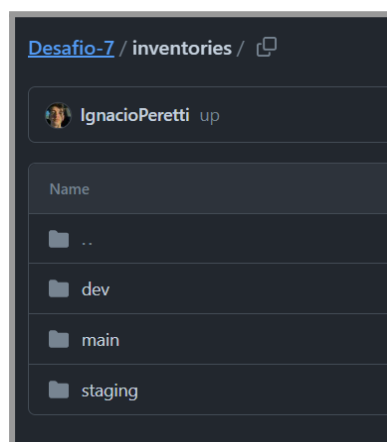
Una carpeta inventories para guardar los inventarios de cada ambiente.

Una carpeta playbook donde guardaremos el playbook de ansible.

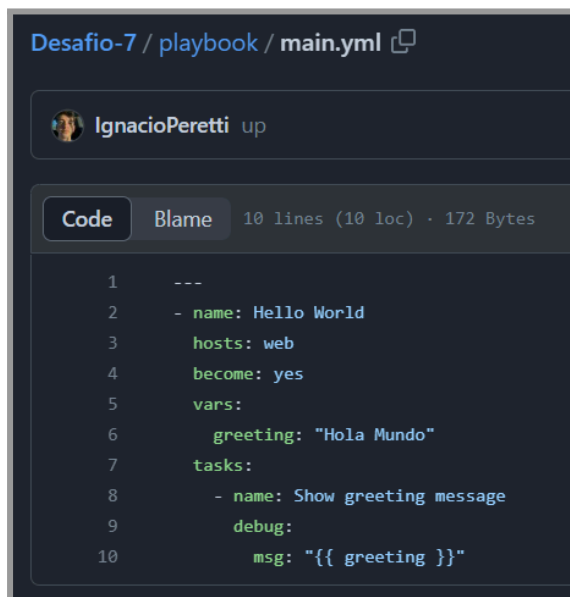
El archivo principal Jenkinsfile.



Las carpetas dentro de la carpeta **inventories**, cada una contiene un archivo inventory.init perteneciente a cada instancia con sus ips y usuarios.



El contenido de mi archivo dentro de la carpeta playbook. (main.yml)



El contenido de mi archivo Jenkinsfile

```
Code Blame 50 lines (48 loc) · 1.8 KB Code 55% faster with GitHub Copilot

1 pipeline {
2     agent { label 'ansible-controller' }
3     stages {
4         stage('Preparation') {
5             steps {
6                 script {
7
8                     if (env.BRANCH_NAME == 'dev') {
9                         env.INVENTORY = 'inventories/dev/inventory.init'
10                    } else if (env.BRANCH_NAME == 'staging') {
11                        env.INVENTORY = 'inventories/staging/inventory.init'
12                    } else if (env.BRANCH_NAME == 'main') {
13                        env.INVENTORY = 'inventories/main/inventory.init'
14                    }
15                }
16                echo "Using inventory: ${env.INVENTORY}"
17            }
18        }
19        stage('Checkout') {
20            steps {
21                checkout scm
22            }
23        }
24        stage('Copy Files') {
25            steps {
26                script {
27                    // Creamos los directorios necesarios en el agente
28                    sh """
29                        mkdir -p ~/inventories/dev ~/inventories/staging ~/inventories/main ~/playbook
30                    """
31
32                    // Copiamos los archivos necesarios al agente remoto
33                    sh """
34                        cp ${WORKSPACE}/inventories/dev/inventory.init ~/inventories/dev/
35                        cp ${WORKSPACE}/inventories/staging/inventory.init ~/inventories/staging/
36                        cp ${WORKSPACE}/inventories/main/inventory.init ~/inventories/main/
37                        cp ${WORKSPACE}/playbook/main.yml ~/playbook/
38                    """
39                }
40            }
41        }
42        stage('Run Ansible Playbook from Jenkins') {
43            steps {
44                sh """
45                    ansible-playbook -i ~/${env.INVENTORY} ~/playbook/main.yml --ssh-extra-args='-o StrictHostKeyChecking=no'
46                """
47            }
48        }
49    }
50 }
```

Función de este código

El pipeline se ejecutará en un nodo que tenga la etiqueta '`ansible-controller`'.

Realiza varias etapas relacionadas con la gestión de configuraciones y la ejecución de un playbook de Ansible.

Stage 'Preparation'

Esta etapa configura una variable de entorno `INVENTORY` basada en el nombre de la rama (`BRANCH_NAME`). Dependiendo de si la rama es '`dev`', '`staging`' o '`main`', se asigna un archivo de inventario específico. Luego, se imprime en la consola qué archivo de inventario se está utilizando.

Stage 'Checkout':

Esta etapa realiza un `checkout` del código fuente del repositorio desde el sistema de control de versiones (SCM). Esto asegura que el código fuente esté disponible en el entorno de construcción.

Stage 'Copy Files':

En esta etapa, se crean directorios necesarios en el agente de Jenkins y se copian archivos de inventario y un playbook de Ansible desde el espacio de trabajo (`WORKSPACE`) a las ubicaciones apropiadas en el agente.

Stage 'Run Ansible Playbook from Jenkins':

Finalmente, se ejecuta un playbook de Ansible utilizando el archivo de inventario que se configuró anteriormente. El comando `ansible-playbook` usa el archivo de inventario especificado y el playbook `main.yml` para realizar la configuración.

Las opciones `--ssh-extra-args='-o StrictHostKeyChecking=no'` se usan para evitar advertencias sobre claves SSH.

A continuación ejecutamos desde Jenkins y si todos los ambientes tienen una correcta salida debería verse así, con esto tendremos configurado un ambiente multi-branch.

Branches (3)						
S	W	Name ↓	Último Éxito	Último Fallo	Última Duración	
✓	☀	dev	36 Seg #6	N/D	14 Seg	▶
✓	☀	main	35 Seg #11	N/D	14 Seg	▶
✓	☀	staging	35 Seg #6	N/D	12 Seg	▶

Si verificamos el Console Output de cada ambiente.

Dev

```
+ ansible-playbook -i /home/jenkins/inventories/dev/inventory.init /home/jenkins/playbook/main.yml --ssh-extra-args=-o StrictHostKeyChecking=no

PLAY [Hello World] *****

TASK [Gathering Facts] *****
ok: [172.17.104.33]

TASK [Show greeting message] *****
ok: [172.17.104.33] => {
  "msg": "Hola Mundo"
}

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

[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

Staging

```
+ ansible-playbook -i /home/jenkins/inventories/staging/inventory.init /home/jenkins/playbook/main.yml --ssh-extra-args=-o StrictHostKeyChecking=no

PLAY [Hello World] *****

TASK [Gathering Facts] *****
ok: [172.17.100.59]

TASK [Show greeting message] *****
ok: [172.17.100.59] => {
  "msg": "Hola Mundo"
}

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

[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

Main

```
+ ansible-playbook -i /home/jenkins/inventories/main/inventory.init /home/jenkins/playbook/main.yml --ssh-extra-args=-o StrictHostKeyChecking=no

PLAY [Hello World] *****

TASK [Gathering Facts] *****
ok: [172.17.102.22]

TASK [Show greeting message] *****
ok: [172.17.102.22] => {
  "msg": "Hola Mundo"
}

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

[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

Diagrama del ejercicio.

