Mi Libro

Juan Pérez

2025-04-14

Contents

Chapter 1

Introducción

1.1 Ejercicio de Ansible

1.1.1 Enunciado

El ejercicio versa sobre la configuración de servidores empleando Ansible playbooks. Se valorará, la estructuración y claridad del código, el uso de parámetros para facilitar la reutilización de los playbooks, el uso de comentarios, el uso de variables, la organización de los artefactos, el uso de convenciones de nombrado de Ansible y de Yaml y, aunque escapa al objetivo de este ejercicio, el posible uso de recursos para reutilización de artefactos, como los Ansible Roles.

Partiendo de dos servidores, configurados de acuerdo a los requerimientos del apartado 2, debe modificarlos para que sea posible el acceso remoto del usuario root empleando contraseña (el acceso con contraseña está desactivado por defecto en la instalación de Rocky).

A continuación, realizará la siguiente configuración en los dos servidores empleando un playbook:

1. Crear un nuevo usuario llamado “admin” que pueda ejecutar comandos privilegiados sin contraseña.

2. Dar acceso por SSH al usuario “admin” con llave pública.

3. Crear el grupo “wheel” (si no existe) y permitir a sus miembros ejecutar sudo.

4. Añadir una lista variable de usuarios (se proporcionará un ejemplo con al menos dos), añadiéndolos al grupo “wheel” y concediéndoles acceso por SSH con llave pública.

5. Deshabilitar el acceso por contraseña sobre SSH para el usuario root.

Los servidores anteriormente configurados son ahora administrables mediante Ansible empleando el usuario “admin”. Ponga a prueba esta configuración con los siguientes cambios/playbooks:

1. Modifique conveniente el inventario para el uso del nuevo usuario “admin”.

2. Uno de los servidores se empleará para correr Apache Httpd y el otro Nginx. Modifique el inventario de forma conveniente para realizar correctamente su administración.

3. Desarrolle un playbook para implementar los requerimientos del ejercicio 4.1.1 en los dos servidores, instalando en cada caso Apache Httpd o Nginx según la configuración del inventario.

Todos los archivos necesarios para la ejecución de los Playbooks (playbooks, inventario, variables, scripts, ...) deben estar localizado en un directorio (con posibles subdirectorios). Junto a los archivos propios de Ansible, debe proporcionar dos scripts (por ejemplo, iniciarNodosManejados.sh y configurarWebServers.sh) para la ejecución de los playbooks con todos los parámetros necesarios.

1.1.2 Solución

Para la resolución de este ejercicio vamos a distinguir dos partes.

Primera Parte:

Configurar mediante Ansible los siguientes aspectos en ambos servidores:

1. Crear un usuario `admin` con privilegios `sudo` sin contraseña.

2. Dar acceso SSH al usuario `admin` mediante clave pública.

3. Crear el grupo `wheel` (si no existe) y permitir que sus miembros puedan usar `sudo`.

4. Crear una lista variable de usuarios (ejemplo: `juan` y `maria`), añadirlos al grupo `wheel` y permitir acceso SSH con clave pública.

5. Deshabilitar el acceso por contraseña al usuario `root`.

Todo esto realizado después de la ejecución del playbook mediante el comando: `ansible-playbook -i hosts.ini playbook.yml --ask-pass --ask-become-pass`

Como **prerequisitos** debemos de destacar que se deben de clonar dos máquinas virtuales a partir de la primera que hicimos en las prácticas (a la cual se le podía hacer un ping) y debemos de cambiar la ip estática y comprobar con el comando `ip a` que efectivamente ha cambiado y que los pings desde nuestra máquina local funciona, en mi caso las he nombrado `sry1 y srv2`.

Comprobaciones realizadas

1. Verificar existencia del usuario `admin` [] id admin
Salida muestra que el usuario `admin` existe y pertenece a los grupos esperados.

---

2. Verificar que `admin` puede usar `sudo` sin contraseña [] sudo -u admin sudo -n true

Si el comando no solicita contraseña y no da error, significa que `admin` tiene privilegios sudo sin contraseña.

---

3. Verificar acceso SSH con clave pública para `admin` [] ssh -i claves/id_rsa_admin admin@192.168.56.103

Se establece conexión sin necesidad de contraseña, lo que indica que la clave pública funciona correctamente.

---

4. Verificar existencia del grupo `wheel` y sus miembros [] getent group wheel

Salida contiene a `admin`, `juan`, `maria`, etc.

---

5. Verificar claves públicas de usuarios variables [] cat /home/juan/.ssh/authorized_keys cat /home/maria/.ssh/authorized_keys

Salida muestra las claves públicas configuradas correctamente para cada usuario.

---

6. Verificar deshabilitación del login por contraseña para `root` [] sudo grep 'PermitRootLogin' /etc/ssh/sshd_config

Salida:

[] PermitRootLogin prohibit-password

Esto indica que el acceso a `root` solo es posible mediante clave pública (no con contraseña).

Explicación detallada de los pasos que se han seguido para la elaboración del ejercicio La estructura de directiorios que tengo es:

[] Ejercicio: claves group_vars SEGUNDA_VERSION tasks playbook.yml hosts.ini

- Donde en claves lo que hemos hecho ha sido generar las claves para el usuario que se nos pide (admin) y para los extras que en mi caso han sido Juan y María (por ejemplo). El comando para generar la claves ha sido: `ssh-keygen -t rsa -b 4096 -C "juan@rocky" -f ~/.ssh/id_rsa_juan`, donde podemos cambiar la ruta, este ha sido el caso de juan, y en mi caso la ruta ha sido `claves/`.

- Dentro de group vars tenemos el archivo `all.yml`:

[] # Archivo: group_vars/all.yml

Definimos el nombre del usuario administrador que se creará en los servidores. admin_user: admin

Especificamos la clave pública SSH para el usuario administrador. # La clave se obtiene utilizando el módulo 'lookup' para leer el archivo correspondiente. ssh_pub_key_admin: "{{ lookup('file', 'claves/id_rsa_admin.pub') }}"

Definimos una lista de usuarios adicionales que se crearán en los servidores. usuarios_extra: # Primer usuario adicional: juan - nombre: juan # Clave pública SSH para el usuario juan, obtenida de un archivo. llave: "{{ lookup('file', 'claves/id_rsa_juan.pub') }}" # Segundo usuario adicional: maria - nombre: maria # Clave pública SSH para el usuario maria, obtenida de un archivo. llave: "{{ lookup('file', 'claves/id_rsa_maria.pub') }}"

- En el directorio `SEGUNDA_VERSION` se encuentra el otro ejercicio que veremos mas adelante.

- Dentro de task encontramos el fichero `crear_usuario.yml`:

[] # Tarea para crear un usuario con privilegios específicos - name: Crear usuario {{ item.nombre }} # Nombre de la tarea, se usa el nombre del usuario de la lista user: name: "{{ item.nombre }}" # Nombre del usuario a crear, tomado de la lista de usuarios groups: wheel # Añadir el usuario al grupo 'wheel' para privilegios sudo append: yes # Asegurarse de que el usuario se añade al grupo sin eliminar otros grupos shell: /bin/bash # Definir el shell predeterminado para el usuario state: present # Asegurarse de que el usuario está presente en el sistema

Tarea para añadir la clave pública SSH al usuario creado - name: Añadir clave SSH a {{ item.nombre }} # Nombre de la tarea, se usa el nombre del usuario de la lista authorized_key: user: "{{ item.nombre }}" # Especificar el usuario al que se añadirá la clave key: "{{ item.llave }}" # Clave pública SSH que se añadirá, tomada de la lista de usuarios

- A continuación, tenemos el fichero `playbook.yml`:

[] --- # Playbook principal para configurar servidores Rocky con Ansible - name: Configurar servidores Rocky con Ansible hosts: servidores # Define el grupo de hosts en el inventario que se configurarán become: true # Habilita la elevación de privilegios para las tareas

tasks: # Permitir temporalmente el acceso root por contraseña - name: Permitir acceso root por contraseña temporalmente lineinfile: path: /etc/ssh/sshd_config # Archivo de configuración de SSH regexp: '^#?PermitRootLogin' # Busca la línea que comienza con PermitRootLogin line: 'PermitRootLogin yes' # Habilita el acceso root por contraseña notify: Restart SSH # Notifica al manejador para reiniciar el servicio SSH

Crear el usuario admin con privilegios específicos - name: Crear usuario admin user: name: "{{ admin_user }}" # Nombre del usuario, definido en las variables groups: wheel # Añade el usuario al grupo wheel append: yes # Asegura que no se eliminen otros grupos shell: /bin/bash # Define el shell predeterminado state: present # Asegura que el usuario exista

Permitir al usuario admin usar sudo sin contraseña - name: Permitir sudo sin contraseña al usuario admin copy: dest: "/etc/sudoers.d/{{ admin_user }}" # Archivo de configuración sudoers para admin content: "{{ admin_user }} ALL=(ALL) NOPASSWD:ALL" # Configuración sudo sin contraseña mode: '0440' # Permisos seguros para el archivo

Añadir clave pública SSH para el usuario admin - name: Añadir clave pública SSH para admin authorized_key: user: "{{ admin_user }}" # Usuario al que se añadirá la clave key: "{{ ssh_pub_key_admin }}" # Clave pública definida en las variables

Asegurar que el grupo wheel existe - name: Asegurar grupo wheel existe group: name: wheel # Nombre del grupo state: present # Asegura que el grupo exista

Permitir al grupo wheel usar sudo sin contraseña - name: Permitir sudo sin contraseña a wheel copy: dest: "/etc/sudoers.d/wheel" # Archivo de configuración sudoers para el grupo wheel content: "%wheel ALL=(ALL) NOPASSWD:ALL" # Configuración sudo sin contraseña mode: '0440' # Permisos seguros para el archivo

Crear usuarios adicionales con acceso SSH y sudo - name: Crear usuarios adicionales con acceso SSH y sudo include_tasks: tasks/crear_usuario.yml # Incluye las tareas definidas en crear_usuario.yml loop: "{{ usuarios_extra }}" # Itera sobre la lista de usuarios adicionales

Deshabilitar el acceso root por contraseña - name: Deshabilitar acceso root por contraseña lineinfile: path: /etc/ssh/sshd_config # Archivo de configuración de SSH regexp: '^#?PermitRootLogin' # Busca la línea que comienza con PermitRootLogin line: 'PermitRootLogin prohibit-password' # Deshabilita el acceso root por contraseña notify: Restart SSH # Notifica al manejador para reiniciar el servicio SSH

handlers: # Manejador para reiniciar el servicio SSH - name: Restart SSH service: name: sshd # Nombre del servicio SSH state: restarted # Reinicia el servicio

- Por último tenemos el fichero `hosts.ini`:

[] # Archivo de inventario de Ansible que define los servidores y sus variables.

[servidores] 192.168.56.103 # Dirección IP del primer servidor gestionado. 192.168.56.104 # Dirección IP del segundo servidor gestionado.

[servidores:vars] ansible_user=ismMV01 # Usuario predeterminado para conectarse a los servidores. # ansible_ssh_private_key_file=~/.ssh/id_rsa # Ruta opcional a la clave privada SSH para autenticación. ansible_connection=ssh # Método de conexión a los servidores, en este caso SSH.

Nota: Para ejecutar el playbook, utilice el siguiente comando: # ansible-playbook -i hosts.ini playbook.yml --ask-pass --ask-become-pass

Segunda Parte:

Los ficheros que tenemos en el segundo ejercico que nos piden es:

[] SEGUNDA_VERSION: claves inventory vars playbooks configurarWebServers.sh iniciarNodosManejados.sh tasks

Las modificaciones respecto el anterior son:

[] # Archivo de inventario de Ansible que define los servidores y su configuración.

[apache] # Grupo de servidores que ejecutarán Apache HTTP Server. 192.168.56.103 ansible_user=admin ansible_ssh_private_key_file=../claves/id_rsa_admin # Dirección

IP del servidor Apache. # ansible_user=admin: Usuario con el que se conectará Ansible. # ansible_ssh_private_key_file=../claves/id_rsa_admin: Ruta a la clave privada SSH para autenticación.

[nginx] # Grupo de servidores que ejecutarán Nginx. 192.168.56.104 ansible_user=admin ansible_ssh_private_key_file=../claves/id_rsa_admin # Dirección IP del servidor Nginx. # ansible_user=admin: Usuario con el que se conectará Ansible. # ansible_ssh_private_key_file=../clave Ruta a la clave privada SSH para autenticación.

[webservers:children] # Grupo compuesto por los servidores Apache y Nginx. apache # Incluye el grupo de servidores Apache. nginx # Incluye el grupo de servidores Nginx.

- Dentro de `vars` tenemos los ficheros `apache.yml` y `nginx.yml`:

- Dentro de `vars` tenemos los ficheros `apache.yml` y `nginx.yml`:

[] # Archivo: apache.yml web_package: httpd # Nombre del paquete que se instalará para configurar Apache HTTP Server. web_service: httpd # Nombre del servicio que se gestionará (iniciar, detener, reiniciar). web_port: 80 # Puerto en el que Apache escuchará las solicitudes HTTP.

El fichero `nginx.yml` es similar, solo cambia `httpd` por `nginx`.

- Dentro del fichero playbooks encontramos le siguiente playbook:

[] --- # Playbook para configurar servidores web - name: Configurar servidores web # Nombre del playbook hosts: webservers # Aplica a los hosts definidos en el grupo 'webservers' become: true # Eleva privilegios para ejecutar las tareas como superusuario vars_files: - "../vars/{{ group_names[0] }}.yml" # Carga variables específicas según el grupo del host

tasks: - name: Instalar paquete web # Instala el paquete necesario para el servidor web package: name: "{{ web_package }}" # Nombre del paquete definido en las variables state: present # Asegura que el paquete esté instalado

- name: Asegurar que el servicio está iniciado y habilitado # Inicia y habilita el servicio web service: name: "{{ web_service }}" # Nombre del servicio definido en las variables state: started # Asegura que el servicio esté en ejecución enabled: true # Asegura que el servicio se inicie al arrancar el sistema

- name: Asegurar que el puerto está abierto (firewalld) # Configura el firewall para permitir tráfico en el puerto del servicio firewalld: port: "{{ web_port }}/tcp" # Puerto definido en las variables permanent: yes # Asegura que la regla sea permanente state: enabled # Habilita la regla del firewall immediate: yes # Aplica la regla inmediatamente when: ansible_facts['os_family'] == "RedHat" # Solo aplica si el sistema operativo pertenece a la familia RedHat

- name: Crear una página de prueba # Crea un archivo HTML de prueba en el servidor web copy: dest: "/var/www/html/index.html" # Ruta del archivo HTML content: "Servidor configurado con {{ web_service }} mediante Ansible" # Contenido del archivo HTML when: web_package != "nginx" or ansible_facts['distribution'] != "Rocky" # Condición para evitar conflictos en Rocky con Nginx

- Por último, creamos dos scripts en bash que automaticen todo el proceso:

  – Dentro del fichero `configurarWebServers.sh`:

  [] #!/bin/bash echo "Ejecutando playbook para configurar servidores web..." ansible-playbook -i inventory/hosts.ini playbooks/configurar_web.yml

  – Dentro del fichero `iniciarNodosManejados.sh`:

  [] #!/bin/bash echo "Iniciando máquinas virtuales..." VBoxManage startvm srv1 --type headless VBoxManage startvm srv2 --type headless echo "Esperando 10 segundos para asegurar arranque..." sleep 10 echo "Comprobando conectividad con Ansible..." ansible all -i inventory/hosts.ini -m ping

[]@l@ Con esto concluimos con el ejercicio de Ansible viendo que todo cumple con lo que se nos pide.