Fuente: Bing Chat AI

Bloque 4. Ansible

ANSIBLE

MÁSTER EN INGENIERÍA WEB 2023 – 2024

Administración de Sistemas Operativos v1.2 *("L-62 Princesa de Asturias")*

© José Manuel Redondo López. Universidad de Oviedo

Contenido

Nivel 1 (Cabo): Instalación y uso básico

¿Qué necesito para hacer este bloque de actividades?

MUY IMPORTANTE: A lo largo de este boletín de ejercicios se repetirá mucho la palabra ***endpoint***. Este es un término que designa tanto una **máquina virtual**, como un **contenedor de SO** (LXD) o un **contenedor de aplicaciones** (*Docker*) que se han visto en módulos anteriores. Debes elegir el tipo de *endpoint* que más se adapta a los ejercicios que ya has hecho de otros módulos (*Vagrant*, LXD, *Docker*…), o bien crear una máquina virtual manualmente si no tienes otra opción (se proporciona también una guía para ello). En cualquier caso, lo que uses para las pruebas **debe tener la posibilidad de aceptar conexiones vía SSH**, preferiblemente de forma automática con certificados (ver luego).

Recuerda tener siempre un *endpoint* "base" a mano del tipo que hayas elegido para replicar y/o usar si ocurre algún desastre. **Por último, recuerda también que cualquier ejercicio relativo a poner en marcha *firewall* no puede hacerse dentro de contenedores.**

Para realizar este boletín de ejercicios necesitas por lo menos dos *endpoints*, uno *Linux* que instale *Ansible* y otro, conectado a este primero y también *Linux* preferentemente, que sea el que controles y sobre el que hagas pruebas. **El *endpoint* que instale *Ansible* se recomienda que sea una máquina virtual**, como las que se crearon en el bloque de *Vagrant*, por ejemplo. Como decíamos antes, hay tres opciones para lograr el segundo *endpoint*.

- **Usar otra máquina virtual** conectada con la primera por una red local: En ese caso basta con un servidor *Linux* (recomendado *Ubuntu 18*) sin GUI, que solo necesitaría 2Gb de RAM y 40Gb de disco duro dinámico para realizar las tareas pedidas. Se podría lograr con un *multi-machine environment* de *Vagrant*, si quieres hacer este ejercicio. Si no has hecho los ejercicios de *Vagrant*, puedes crear una manualmente conectada a la primera.
- **Uno o varios contenedores LXD**: Si se reutilizan los contenedores del boletín de ejercicios correspondiente, se podrá hacer todo este bloque de ejercicios en una sola máquina virtual. En caso de usar esta opción, recuerda hacer el proceso de conexión automática SSH con certificados que se mencionó antes.
- **Uno o varios contenedores *Docker***: Se pueden usar de forma análoga a los anteriores, pero recuerda que, si uno muere, tendrás que volver a crearlo y hacer el proceso de acceso automatizado a SSH, al ser efímeros.

De todas estas opciones, la segunda es la que menos recursos consume y la primera es también adecuada. En cualquier caso, la decisión final de la alternativa a usar es tuya.

Creación de conexiones automáticas vía SSH

Esta tarea se incluye porque es necesaria para facilitar el trabajo con *Ansible*. Lo primero que hay que hacer es asegurarse de que la máquina destino tiene instalado el paquete `openssh-server`. Hecho esto, se puede seguir el siguiente procedimiento para conseguir la conexión automática desde la máquina con

Ansible al servidor que va a ser manejado: https://www.linuxbabe.com/linux-server/setup-passwordless-ssh-login. Conviene **comprobar** que la conexión es automática manualmente antes de empezar a trabajar.

```
operario@server1804:~$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/operario/.ssh/id_rsa):
Created directory '/home/operario/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/operario/.ssh/id_rsa.
Your public key has been saved in /home/operario/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:RSmpeU9cqrXcwaLXhh9ioFPJjUIpBf1AFyWHlSdiISU operario@server1804
The key's randomart image is:
+---[RSA 2048]----+
|     .=Eo*B=o    |
|     . =+*++ o   |
|      o B B.*    |
|       + O.B o   |
|        =SO * .  |
|       o o O =   |
|        . o + .  |
|             .   |
|                 |
+----[SHA256]-----+
operario@server1804:~$ ssh-copy-id operario@192.168.1.200
The authenticity of host '192.168.1.200 (192.168.1.200)' can't be established.
ECDSA key fingerprint is SHA256:bCQv2rSfFgVslCqRC76WfohRiS87Y1p3JNYAFETUz/E.
Are you sure you want to continue connecting (yes/no)? 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: INFO: 1 key(s) remain to be installed -- if you are prompt
ed now it is to install the new keys
operario@192.168.1.200's password:
```

Creación de un usuario *sudoer* automático (sin *password*)

Esto es una muy mala práctica de seguridad, pero puede activarse temporalmente mientras los *endpoints* están provisionándose con Ansible porque facilita mucho la tarea. Si se necesita un usuario que se haga con privilegios de `root` con sudo sin solicitar una *password*, es necesario editar el fichero `/etc/sudoers` y añadir una línea como esta, que lo hace sobre el usuario de nombre `ubuntu`:

```
ubuntu ALL=(ALL) NOPASSWD:ALL
```

N1ANSIBLE_INSTALAR: Instalación de *Ansible* *(0,1 puntos)*

Descripción de la actividad

Consiste en tener una **instalación operativa de *Ansible***.

Resultados esperados

Eres capaz de instalar *Ansible* y comprobar que el comando está disponible

Información necesaria para su realización

En esta actividad se trata de **instalar el software *Ansible*** sobre una máquina *Linux* que uses para hacer estas actividades, de manera global (**para todos los usuarios** de la máquina, no para un usuario particular) siguiendo la guía oficial de instalación para el SO que estés usando. Este enlace es para *Ubuntu*:

https://docs.ansible.com/ansible/latest/installation_guide/intro_installation.html#installing-ansible-on-ubuntu, pero también está el proceso documentado para muchos otros sistemas operativos en la misma web. El resumen de pasos a dar en *Ubuntu* aparece en la siguiente imagen

```
operario@server1804:~$ sudo apt-add-repository -y ppa:ansible/ansible
[sudo] password for operario:
Hit:1 http://archive.ubuntu.com/ubuntu bionic InRelease
Hit:2 http://archive.ubuntu.com/ubuntu bionic-updates InRelease
Hit:3 http://archive.ubuntu.com/ubuntu bionic-backports InRelease
Hit:4 http://archive.ubuntu.com/ubuntu bionic-security InRelease
Get:5 http://ppa.launchpad.net/ansible/ansible/ubuntu bionic InRelease [15,9 kB]
Get:6 http://ppa.launchpad.net/ansible/ansible/ubuntu bionic/main amd64 Packages
 [528 B]
Get:7 http://ppa.launchpad.net/ansible/ansible/ubuntu bionic/main Translation-en
 [344 B]
Fetched 16,8 kB in 3s (5.498 B/s)
Reading package lists... Done
operario@server1804:~$ sudo apt-get update
Hit:1 http://archive.ubuntu.com/ubuntu bionic InRelease
Hit:2 http://archive.ubuntu.com/ubuntu bionic-updates InRelease
Hit:3 http://archive.ubuntu.com/ubuntu bionic-backports InRelease
Hit:4 http://archive.ubuntu.com/ubuntu bionic-security InRelease
Hit:5 http://ppa.launchpad.net/ansible/ansible/ubuntu bionic InRelease
Reading package lists... Done
operario@server1804:~$ sudo apt-get install -y ansible
```


N1ANSIBLE_INVENT: Creación de inventarios básicos *(0,2 puntos)*

Descripción de la actividad

Consiste en aprender a **crear un inventario sencillo** de *endpoints* a controlar vía *Ansible*.

Resultados esperados

Eres capaz de crear una inventario de *endpoints* que contenga al mismo tiempo la máquina local y remota, y entiendes **cómo agrupar los *endpoints*** creados en grupos que los contengan (así como hacer grupos de grupos), para hacer una estructura de *endpoints* en las que aplicar acciones adaptada a cualquier escenario. Además, puedes indicar a *Ansible* los medios de conexión adecuados para acceder a los *endpoints* si es necesario.

Información necesaria para su realización

RECUERDA*: *Es muy conveniente que los endpoints que hayas elegido tengan la opción de conectarse automáticamente vía SSH con certificados, y que el usuario en el endpoint remoto tenga la opción de usar sudo para la ejecución de comandos sin preguntar la password.

Creación del inventario de endpoints

Esta actividad pretende que **pruebes la creación de un inventario de *endpoints* para *Ansible* en el fichero donde *Ansible* los guarda por defecto**. Lo haremos en formato INI. Este inventario debe contener los siguientes elementos:

- Bajo la etiqueta `[local]` la propia máquina que tiene instalado *Ansible*
- Bajo la etiqueta `[remoto]` **la IP de otro *endpoint Linux*** que arranques al mismo tiempo que la máquina que tiene instalado *Ansible*, de forma que se use como pruebas para controlar algo remoto. Como decíamos, puede ser un clon de la máquina *Ansible* o cualquier otro *endpoint* creado

en actividades anteriores, lo que te sea más fácil. Debes asegurarte de hacer lo necesario sobre este segundo *endpoint* para que **tenga una IP fija** y así poder hacer los ejercicios sin problema. Si has usado una máquina virtual, **se recomienda consultar la introducción del bloque de *Vagrant*, que te enseña a asignarle una IP fija** para poder poner la misma en el inventario.

- Hecho eso, para **practicar con grupos de servidores**, define la etiqueta `todos` de esta forma para agrupar todos los grupos creados previamente. Esto permite agrupar servidores de cualquier forma posible (**grupos de grupos** con cualquier nivel de anidamiento), dándonos mucha potencia para decidir qué se aplica sobre qué *endpoints*.

```
[todos:children]
local
remoto
```

Especificación de forma de acceso a las endpoints

Con *Ansible* podemos controlar muchos *endpoints*, **pero eso no quiere decir que tengamos la misma forma de acceso a todas**. Aunque activemos el acceso automático vía SSH y certificados a los *endpoint* remotos, podemos usar el inventario para indicar a *Ansible* qué hacer, por ejemplo, en aquellos casos en los que el nombre de usuario que usemos en la máquina que tiene *Ansible* instalado **no sea un nombre de usuario reconocido en algún sistema remoto**. Esto quiere decir que puede haber casos en los que tengamos acceso SSH automático con certificados al *endpoint* usando el usuario A de la máquina que tiene *Ansible*, pero en el *endpoint* remoto el usuario A no existe y tenemos que decirle que usa el B cuando intente conectarse a él.

En ambos casos, esto se puede hace **definiendo una sección complementaria** a alguna de las que hemos definido en el inventario añadiendo `:vars` al nombre de esta. En ella se definirán variables con significado especial que se usarán para el acceso al *endpoint* remoto. Se contemplan dos casos:

- No tenemos acceso automático por certificados y **solo admitimos usuario y clave**. Por motivos de seguridad, esto **solo debe hacerse con la máquina local que tiene *Ansible* instalado**, para que la clave del usuario no viaje por la red en plano. Las variables del sistema que nos permiten el acceso de esta forma son `ansible_user` y `ansible_password`. Ej.:

```
[local]
127.0.0.1
# Para el sistema local se usarán este usuario y clave
[local:vars]
ansible_user=ssiuser
ansible_password=test123...
```

- **Tenemos acceso con certificado**, pero en lugar de usar el nombre de usuario que tenemos en nuestra máquina local debemos usar otro. En este caso podemos usar la variable prefinida `ansible_user` y entonces se hará un `ssh <usuario que se pone en ansible_user>@<endpoint remoto>` al *endpoint* remoto

```
[remoto]
10.181.121.85
10.181.121.14
# Para las dos IPs anteriores se usará el certificado del usuario actual, pero el nombre de
# usuario será “ubuntu” en el momento de hacer la conexión
[remoto:vars]
```

```
ansible_user=ubuntu
```

N1ANSIBLE_CONTROL: Control básico con *Ansible* *(0,1 puntos)*

Descripción de la actividad

Consiste en aprender a **determinar cuando los *endpoints* controlados están activos** y se puede interactuar con ellos correctamente con *Ansible*.

Resultados esperados

Esta actividad finaliza cuando todos los módulos `ping` de *Ansible* probados sobre los objetivos devuelvan una conexión satisfactoria

Información necesaria para su realización

Antes de realizar este ejercicio **debes asegurarte de que has implementado la conexión automática vía SSH** entre la máquina que tiene *Ansible* y el *endpoint* remoto que mencionamos en el ejercicio anterior para facilitar el trabajo. **En caso contrario** debes usar la opción `--ask-pass` al invocar los comandos. Para hacer una prueba de funcionamiento y ver si todo está correctamente instalado, esta actividad te pide lo siguiente:

- Hacer una conexión con **el módulo** `ping` sobre la propia máquina: `ansible --connection=local local -m ping`. Este módulo comprueba que *Ansible* puede contactar con cada *endpoint* (https://docs.ansible.com/ansible/latest/collections/ansible/builtin/ping_module.html).
- Copiar el fichero de `hosts` creado en la actividad anterior a otro en tu cuenta de usuario, por ejemplo, y úsalo así repitiendo la operación anterior: `ansible -i <ruta del fichero de inventario> --connection=local local -m ping`. Esto te permite probar cómo realmente se pueden hacer operaciones **sobre cualquier inventario que tengas**, no sólo el de por defecto.
- Usa ahora el mismo módulo **sobre el *endpoint*** remoto creada anteriormente.
- En caso de que hubiera varios *hosts* en un grupo y por la razón que fuese **quisieramos limitar una operación** cualquiera para que se haga sólo en uno de ellos, podríamos añadir la opción `--limit "<IP o nombre DNS>"`

N1ANSIBLE_SHELLM: Módulo *shell* *(0,1 puntos)*

Descripción de la actividad

Consiste en que **aprendas a usar el módulo** `shell` **de *Ansible*** y entiendas sus ventajas y desventajas.

Resultados esperados

Eres capaz de usar el módulo `shell` sobre cualquier *endpoint* del inventario. Puedes responder a estas preguntas:

- *¿Qué añade el parámetro* `-vvvv` *a la salida del comando?*
- *Si tienes un script que gestiona algunas cosas en una máquina, ¿Cómo podrías usarlo ahora con Ansible?*

Información necesaria para su realización

En esta actividad se trata de seguir las indicaciones de teoría para que, usando el módulo `shell`, procedas a actualizar tanto la máquina que tiene *Ansible* instalado como un *endpoint* remoto. La documentación del módulo `shell` está aquí: https://docs.ansible.com/ansible/latest/collections/ansible/builtin/shell_module.html

Hecho esto, **vuelve a hacer esta operación** añadiendo el parámetro `-vvvv` y viendo qué ocurre entonces. Un ejemplo de uso es: `ansible todos -m shell -a 'sudo apt-get update'`

N1ANSIBLE_INSTALM: Instalación de programas desde línea de comandos: Acciones con privilegios elevados *(0,1 puntos)*

Descripción de la actividad

Consiste en que aprendas a instalar software con *Ansible* sobre *endpoints Debian/Ubuntu* con el módulo de *Ansible* `apt`.

Resultados esperados

Eres capaz de usar el módulo `apt` sobre una *endpoint* remoto para instalar un paquete y también comprobar la idempotencia de dicho módulo

Información necesaria para su realización

Esta actividad consiste en probar lo mencionado en la teoría del curso para **instalar**, a través del módulo `apt` (https://docs.ansible.com/ansible/latest/collections/ansible/builtin/apt_module.html), el paquete `nginx` en el *endpoint* remoto solamente. Comprueba luego que puedes acceder al servidor web recién instalado desde la máquina que tiene *Ansible*. Una vez hecho esto, vuelve a ejecutar la misma orden de nuevo. *¿Se instala algo ahora?*

ATENCIÓN: La instalación de paquetes necesita privilegios de `root`. Esto implica que **la identidad de usuario bajo la que se hace la conexión debe ser *sudoer*** y preferiblemente que no pida la clave para ejecutar `sudo` (ver la introducción de este documento para saber hacerlo). En caso contrario, deben añadirse los siguientes parámetros al comando como se vio en teoría: `--become-user=root -become-method=sudo --ask-become-pass`. Por ejemplo: `ansible todos -b --become-user=root --become-method=sudo -m apt -a 'name=curl state=present'`

NOTA: Este ejercicio cubre la **instalación sobre sistemas *Debian/Ubuntu*** como ejemplo de esta clase de módulos. Pero hay más opciones si tu sistema no es de este tipo:

- Para **sistemas tipo *Red Hat***, se puede usar un módulo equivalente, `yum` (https://docs.ansible.com/ansible/latest/collections/ansible/builtin/yum_module.html) o `dnf`

(https://docs.ansible.com/ansible/latest/collections/ansible/builtin/dnf_module.html#ansible-collections-ansible-builtin-dnf-module).

- Para instalar software de forma que **no haya que preocuparse por el gestor de paquetes** de la distribución de *Linux* del *endpoint*, se puede usar el módulo `package` (https://docs.ansible.com/ansible/latest/collections/ansible/builtin/package_module.html).
- Por otro lado, para **instalar paquetes en sistemas *Windows*** podemos usar el módulo `win_package` (https://docs.ansible.com/ansible/latest/collections/ansible/windows/win_package_module.html).

N1ANSIBLE_PLAYBOOKS1: Uso básico de *playbooks* *(0,2 puntos)*

Descripción de la actividad

Consiste en **aprender a usar un *playbook* de *Ansible*** de forma básica.

Resultados esperados

Eres capaz de crear un *playbook* que actualice, desinstale e instale *software* de un *endpoint* remoto, y además se asegure de que el *software* instalado está operativo, **comprobando primero el *playbook*** antes de ejecutarlo.

Información necesaria para su realización

Esta actividad consiste en **crear un *playbook* sencillo** a partir de los contenidos dados en teoría, que haga lo siguiente sobre un *endpoint* remoto:

- **Actualice los paquetes** del sistema
- **Desinstale** el `nginx` instalado anteriormente
- **Instale** el paquete `apache2`
- **Use un *handler*** para que, una vez instalado `apache2`, se asegure de que está arrancado.

NOTA: Si lo necesitas, recuerda que puedes limitar la aplicación del *playbook* a un solo *host* con `--limit "<IP o nombre DNS>"`

MUY IMPORTANTE: En este ejercicio, **y de aquí en adelante**, debes familiarizarte (y, por supuesto, usar) el ***check mode*** cada vez que vayas a ejecutar un *playbook*: https://docs.ansible.com/ansible/latest/user_guide/playbooks_checkmode.html.

Este modo permite **simular la ejecución de un *playbook*** detectando un gran nº de posibles errores que podrían ocurrir durante la misma. **No realiza ningún cambio en los *endpoints* de destino** (aunque la salida de a entender lo contrario), pero ejecutar un *playbook* que haya pasado un procedimiento de *check* **da más garantías** de que la ejecución en sí irá como se espera que sin usar este modo. Usarlo es muy sencillo: `ansible-playbook <fichero .yml del playbook> --check`.

Nivel 2 (Sargento): Resolución de problemas sencillos frecuentes

N2ANSIBLE_FACTS: Obtención de *facts* *(0,1 puntos)*

Descripción de la actividad

Consiste en que seas capaz de **obtener todas las *facts* de un *endpoint*** remoto con *Ansible*.

Resultados esperados

Eres capaz de obtener las *facts* de cualquier host controlado por Ansible. Puedes contestar a estas preguntas:

- *¿Crees que obtener esta información es peligroso desde el punto de vista de la seguridad de los endpoints? ¿Por qué?*
- *¿Entiendes que toda esta información ahora puede usarse para construir scripts de Ansible que controlen endpoints? ¿Qué opinas del control que te dan sobre ellas a la vista de los resultados?*

Información necesaria para su realización

En esta actividad se trata de poner en práctica la obtención de *facts* tanto con el *host* local como con el remoto que estamos manejando en estas actividades de la forma vista en teoría. El resultado sería algo como el que aparece en esta imagen, aunque **un listado completo de todas las *facts*** que puedes extraer de un *endpoint* cualquiera lo puedes ver aquí: https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_vars_facts.html .

```
operario@server1804:~/ansible-example$ ansible -m setup --connection=local loca
lhost
[DEPRECATION WARNING]: Distribution Ubuntu 18.04 on host 127.0.0.1 should use
/usr/bin/python3, but is using /usr/bin/python for backward compatibility with
prior Ansible releases. A future Ansible release will default to using the
discovered platform python for this host. See https://docs.ansible.com/ansible/
2.8/reference_appendices/interpreter_discovery.html for more information. This
feature will be removed in version 2.12. Deprecation warnings can be disabled
by setting deprecation_warnings=False in ansible.cfg.
127.0.0.1 | SUCCESS => {
    "ansible_facts": {
        "ansible_all_ipv4_addresses": [
            "10.0.2.15",
            "192.168.2.8"
        ],
        "ansible_all_ipv6_addresses": [
            "fe80::a00:27ff:fe88:c8fb",
            "fe80::a00:27ff:fecc:c3eb"
        ],
        "ansible_apparmor": {
            "status": "enabled"
        },
        "ansible_architecture": "x86_64",
        "ansible_bios_date": "12/01/2006",
        "ansible_bios_version": "VirtualBox",
        "ansible_cmdline": {
            "BOOT_IMAGE": "/boot/vmlinuz-4.15.0-54-generic",
            "maybe-ubiquity": true,
            "ro": true,
            "root": "UUID=cfd296e1-063e-484e-98cc-eff44a4c2b11"
        },
        "ansible_date_time": {
            "date": "2019-07-12",
            "day": "12",
```

N2ANSIBLE_MANAGE: Operaciones de gestión en línea de comandos *(0,1 puntos)*

Descripción de la actividad

Consiste en que aprendas una serie de **opciones de uso de *Ansible*** que te permitirán más flexibilidad y control en la ejecución de sus tareas.

Resultados esperados

Eres capaz de usar opciones de gestión variadas de *Ansible* que permiten optimizar su ejecución, comprobar datos de los sistemas administrados y comprobar lo que hacen las operaciones antes de hacerlas

Información necesaria para su realización

Esta actividad consiste en probar estas operaciones de **gestión con la línea de comandos de *Ansible***. No tienen que ver con la modificación de *endpoint* remotos, porque esto lo haremos a través de *playbooks*.

- **Aceleración de conexiones `ssh`**: El uso de *endpoints* remotos en *Ansible* necesita un uso intensivo de conexiones `ssh`. Por tanto, la velocidad a la que estas se hagan influye mucho en el rendimiento de las operaciones. Hay una manera de acelerar estas conexiones activando una característica especial de la configuración de *Ansible*. Para ello, vamos al fichero de configuración de *Ansible* (`ansible.cfg` en el directorio `/etc`) y buscamos la sección `[ssh_connection]`. Añadimos entonces la línea `pipelining=True` para intentar conseguir esa optimización.
- **Lanzar operaciones en segundo plano o asíncronas**: *Ansible* permite lanzar operaciones en segundo plano (asíncronas) para aquellos casos en los que estas tarden mucho en completarse y queramos ir haciendo otras cosas en paralelo. Para probarlo, usar el siguiente comando para actualizar los paquetes de todas nuestros *endpoints* remotos: `ansible all -b -B 3600 -P 0 -a "apt-get -y update"`. El significado de los parámetros es el siguiente:
  - `-B`: Tiempo máximo (en segundos) que dejaremos funcionar a la tarea
  - `-P`: Intervalo de tiempo (en segundos) que esperaremos para preguntar a los servidores sobre cómo va el trabajo lanzado
- **Comprobación de ficheros de *log***: Otra forma rápida de comprobar si alguno de los servidores manejados ha tenido una incidencia es ver rápidamente sus últimas entradas de *log*. Probar estos comandos para ver el efecto: `ansible all -b -a “tail /var/log/messages”`, `ansible all -b -a "tail /var/log/auth.log"`
- **Ver información de ficheros en *endpoints* remotos**: `stat` es un módulo que se puede utilizar para obtener información variada sobre cualquier fichero de los *endpoint* remotos manejados (https://docs.ansible.com/ansible/latest/collections/ansible/builtin/stat_module.html). Esta actividad también trata de probar a obtener información variada de cualquier fichero con un comando como este: `ansible all -m stat -a "path=/etc/environment"`

N2ANSIBLE_PLAYBOOKS2: Operaciones típicas sobre *endpoints* remotos con módulos de *Ansible* básicos y *playbooks* *(0,4 puntos)*

Descripción de la actividad

Consiste en usar **opciones más avanzadas de los *playbooks* de *Ansible*** para poder instalar el *software* y configurar los *endpoints* de forma que se adapta a más casos de uso.

Resultados esperados

Te has familiarizado suficientemente con el uso de *playbooks* para manejar *endpoint* remotos mediante el uso de módulos de *Ansible* que hacen operaciones comunes, consultando su documentación oficial para entender su funcionamiento

Información necesaria para su realización

Hemos visto que se pueden invocar módulos directamente en línea de comandos con *Ansible*, pero que el uso de *playbooks* es más flexible y potente al permitir guardar en un fichero de texto operaciones reproducibles sobre muchos *endpoints*. **El objetivo de esta tarea es lograr un mayor entendimiento de estos *playbooks*.**

Para ello se pide **crear un *playbook* nuevo** donde probar los siguientes comandos *Ansible* que hacen operaciones comunes. Es preferible hacer esto sobre un *endpoint* remoto *Linux* recién instalado para evitar problemas.

- **Pre y post *tasks***: Añadir al *playbook* una tarea que actualice la caché de `apt` antes de la ejecución de cualquier otra tarea, y otra tarea que elimine automáticamente los paquetes no usados en ese momento (`autoremove` y `autoclean`). Usar para ello el módulo `apt` visto en el nivel anterior: https://docs.ansible.com/ansible/latest/collections/ansible/builtin/apt_module.html

- **Gestión de usuarios y grupos**: Convertir estos comandos, que se dan sólo como ayuda, **en algo usable vía *playbook***, creando un usuario que se llame como vuestro UO y que pertenezca a un grupo nuevo llamado `custom_users` y también al grupo `sudoers`:

```
ansible <endpoint remoto> -b -m group -a "name=magos state=present"
ansible <endpoint remoto> -b -m user -a "name=rararasputin group=magos createhome=yes"
```

  Para hacer estas operaciones es necesario usar los módulos `group` (https://docs.ansible.com/ansible/latest/collections/ansible/builtin/group_module.html) y `user` (https://docs.ansible.com/ansible/latest/collections/ansible/builtin/user_module.html). Para la **generación de las claves de usuario** de manera segura se pueden seguir estas indicaciones: https://docs.ansible.com/ansible/latest/reference_appendices/faq.html#how-do-i-generate-encrypted-passwords-for-the-user-module

- **Gestión de paquetes multiplataforma**: Como mencionamos en el nivel anterior, para sistemas *Debian* el módulo `apt` es el encargado de hacer la gestión de paquetes, pero para sistemas *Red Hat* se debe usar el módulo `yum` o el `dnf`. Como esto hace que la gestión de *software* en un *endpoint*

remoto se pueda volver innecesariamente complicada si tenemos *endpoints* heterogéneos, una forma sencilla de gestionar paquetes sin preocuparse del tipo de distribución de *Linux* que tenemos es usar el módulo `package` que hemos mencionado (https://docs.ansible.com/ansible/latest/collections/ansible/builtin/package_module.html). Esta actividad es una oportunidad muy buena para probarlo, y por tanto se pide usar este módulo para instalar el paquete con el *playbook* `apache2`, adaptando este comando: `ansible app -b -m package -a "name=apache2 state=present"`

- **Gestión de ficheros y directorios**: Una vez instalado un servidor web, vamos a usar la web que proporcionamos como ejemplo para dotarlo de contenido. Para ello se pide usar estos comandos como ejemplo para hacer las siguientes tareas, que se añadirán al *playbook* creado:

```
ansible multi -m copy -a "src=/etc/hosts dest=/tmp/hosts"
ansible multi -b -m fetch -a "src=/etc/hosts dest=/tmp"
ansible multi -m file -a "dest=/tmp/test mode=644 state=directory"
```

  - Crear una carpeta `sample_web` dentro de `/var/www/html` con el módulo `file` que debe tener como propietario y grupo al usuario `www-data` (https://docs.ansible.com/ansible/latest/collections/ansible/builtin/file_module.html).
  - Copiar la carpeta con la web de ejemplo desde el host al directorio creado anteriormente, usando para ello el módulo `copy` (https://docs.ansible.com/ansible/latest/collections/ansible/builtin/copy_module.html).
  - Copiar del *endpoint* remoto su fichero de configuración de *Apache 2* (`/etc/apache2/apache2.conf`) a la máquina que tiene *Ansible* instalado para revisarlo posteriormente. Usar para ello el módulo `fetch` (https://docs.ansible.com/ansible/latest/collections/ansible/builtin/fetch_module.html)

- **Gestión de tareas programadas con *cron***: Usar la documentación del módulo `cron` (https://docs.ansible.com/ansible/latest/collections/ansible/builtin/cron_module.html) para **lanzar una tarea que haga una actualización de paquetes y del *software*** instalado en cada reinicio del *endpoint*.

N2ANSIBLE_INFRAWEB: Aprovisionamiento de una infraestructura de web y base de datos *(0,3 puntos)*

Descripción de la actividad

Consiste en usar los conocimientos adquiridos de *Ansible* para **provisionar dos *endpoints* que representen una estructura típica de una web sencilla**, con un *endpoint* alojando la web y otro su SGBD.

Resultados esperados

Eres capaz de poner en marcha una infraestructura típica para servir una web de manera automatizada usando *Ansible*, poniendo en marcha tanto servidor web como servidor de base de datos

Información necesaria para su realización

En esta tarea se trata de ampliar la anterior para crear una infraestructura más compleja de *endpoints* que represente a una estructura web sencilla típica, donde uno tendrá el servidor web y otro una base de datos. Para ello **se trata de hacer otro *playbook*** que contenga una configuración para una base de datos *MySQL* o *MariaDB*. Esto nos servirá para trabajar con módulos más complejos y específicos para manejar SGBD:

- `mysql_db` (https://docs.ansible.com/ansible/latest/collections/community/mysql/mysql_db_module.html)
- `mysql_user` (https://docs.ansible.com/ansible/latest/collections/community/mysql/mysql_user_module.html)

Por tanto en esta actividad se necesitan dos *endpoints* comunicados entre ellos de alguna forma. Podrían ser, por ejemplo, dos máquinas virtuales independientes clonadas de alguna forma de una máquina base para ahorrar tiempo. Por ejemplo, **el nivel 5 de *Vagrant*** tiene una sección para practicar con *linked clones* y **entornos multi-máquina** que se adapta bien a este ejercicio.

Acciones sobre el primer *endpoint*:

- Tendrá la **configuración del servidor web** hecha a partir de la actividad anterior.
- Instalar el paquete `php-mysql` para soportar MySQL desde PHP
- Copiar este script PHP para servirlo desde el *Apache* (permisos `0664` y cambia `localhost` por la IP del *endpoint*) y reiniciar *Apache* luego.

```
<?php

$connection = new PDO('mysql:host=localhost;dbname=umbrella', 'klaus', 'klaus');
$statement  = $connection->query('SELECT academy FROM umbrella');

echo $statement->fetchColumn(); ?>
```

Acciones sobre el segundo *endpoint*:

- Instalar el paquete `mariadb-server` y asegurarse de que su servicio asociado (`mariadb`) está arrancado
- Instalar el paquete `python-mysqldb` necesario para hacer tareas con la base de datos
- Usando el módulo `mysql_db` de *Ansible*, crear una nueva base de datos de nombre (`name`) `umbrella` y con `collation` que sea `utf8_general_ci`
- Crear un nuevo usuario para la base de datos usando el módulo `mysql_user` llamado `klaus` (*password* `klaus`), con privilegios (`priv`) `*.*:ALL` y con el host inicializado a la IP del host que tiene la base de datos.
- Proporcionar datos de ejemplo para la nueva BD al *endpoint*, copiando este fichero en él

```
CREATE TABLE IF NOT EXISTS academy (
  message varchar(255) NOT NULL
) ENGINE=MyISAM DEFAULT CHARSET=utf8;

INSERT INTO demo (message) VALUES('Hardgreeves!');
```

- Pasar los datos de ejemplo a la base de datos ejecutando con el módulo `shell` esto: `cat <ruta donde has copiado el fichero .sql anterior> | mysql -u klaus -pklaus umbrella`

Nivel 3 (Teniente de navío): Resolución de situaciones más avanzadas

N3ANSIBLE_ASYNC: Ordenamiento, asincronía y sincronización de *tasks*: *register*, *async*, *when* y *handlers* *(0,2 puntos)*

Descripción de la actividad

Consiste en **aprender a lanzar tareas dentro de un *playbook* de *Ansible* de manera asíncrona o condicional** para optimizar los tiempos de ejecución de los *scripts* de este.

Resultados esperados

Eres capaz de ordenar tareas para que se ejecuten secuencialmente una detrás de otra cuando alguna ha terminado satisfactoriamente, y además también eres capaz de lanzar tareas en paralelo y esperar a que terminen. Puedes responder a estas preguntas:

- *¿Qué uso crees que puede tener lanzar tareas asíncronas?*
- *¿Puedes pensar en un ejemplo en el que tener tareas síncronas y asíncronas incremente el rendimiento de un playbook?*

Información necesaria para su realización

Pruebas de register, when y notify

Estas tres palabras reservadas de *Ansible* **son la base para construir condicionales** en ellos *playbooks* (https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_conditionals.html), es decir, el equivalente a un `if` en un lenguaje tradicional. Por ello, es importante familiarizarse con su uso.

Para ello vamos a hacer **un nuevo *playbook*** que modele la instalación de `nginx` que usa `register-when` y `notify` vista en teoría (transparencia "***Uso de Playbooks***") sobre un *endpoint* remoto que añadiremos al inventario al que llamaremos `nginx_server`, y que puede estar basado en uno de los anteriores. Tras la instalación de `nginx` para probar esto, y comprobar que se ejecutan los `handlers` definidos, es necesario hacer dos tareas adicionales:

- **Lanzar el *playbook* anterior dos veces** y comprobar que efectivamente los `handlers` no se ejecutan la segunda vez debido a la idempotencia de las tareas de *Ansible*
- **Modificar el *playbook* de *Apache 2* anterior** para que cuando se cree el nuevo directorio para la web de *Apache 2* se ejecute un `handler` que recargue Apache

Asincronía de tareas

Dado que el resto de las actividades las vamos a hacer con ficheros de texto, conviene también probar este tipo de tareas introducidas en los mismo y no solo en línea de comandos como antes. Esto requiere el uso de `register` para temas de sincronía. La actividad consiste en lo siguiente, usando para completarla los ejemplos de esta web: https://docs.ansible.com/ansible/latest/user_guide/playbooks_async.html

- **Lanza una tarea en paralelo** con la de `nginx` anterior invocando con el módulo `shell` a la orden `sleep` 10 segundos. Usar la opción `async` para esperar a que se complete `20` segundos y poner `0` en `poll` para que se lancen en paralelo. Convertir la tarea de `nginx` en asíncrona también. Usar para todo ello los ejemplos del enlace anterior.
- **Esperar a que ambas tareas acaben** usando el ejemplo de sincronización del enlace anterior de forma que no se pueda continuar con el *playbook* hasta que las dos tareas asíncronas informen de que han terminado.

`NOTA`: Usar `async` **no es compatible** con el *check mode*.

N3ANSIBLE_VARFACTS: Uso de variables y *facts* *(0,4 puntos)*

Descripción de la actividad

Consiste en **aprender a usar las variables de los *scripts* de *Ansible*** en distintos puntos y además a **usar *facts* como valores utilizables** en cualquier playbook que diseñes

Resultados esperados

Entiendes la definición de variables desde distintos puntos del *playbook* y además puedes usar la definición de variables para crear tareas que implementen bucles. Por otro lado, sabes también **cómo acceder a las *facts* de los *endpoint*** que manejas.

Información necesaria para su realización

En esta actividad vamos a practicar con la definición y uso de variables en *playbooks*. Para ello vamos a estudiar la definición de variables en distintos ámbitos según necesidades.

Uso de facts como variables

Puedes usar *facts* como valores de alguna variable en alguna de las pruebas siguientes si quieres, usando las indicaciones de https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_vars_facts.html. Como se dijo en el módulo anterior, las *facts* de *Ansible* son **datos de los sistemas remotos que manejamos** (SO, IPs, sistemas de ficheros...). Puedes acceder a estos datos usando la variable predefinida `ansible_facts` (algunas variables se pueden referenciar directamente con el prefijo `ansible_`, pero yo prefiero usar un esquema de acceso único para todas).

Este ejemplo usa el modelo del **primer disco de un sistema remoto** en un *playbook*: `{{ ansible_facts['devices']['xvda']['model'] }}`, mientras que este usa el **nombre del host**: `{{ ansible_facts['nodename'] }}`

Se pueden **usar *facts* en los condicionales** que vimos en el ejercicio anterior (https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_conditionals.html) y también en **plantillas** (niveles siguientes). También se pueden crear **grupos de *hosts* dinámicos** que encajen con un criterio concreto establecido por las *facts*. Esto es un tema avanzado que no vamos a tocar en el curso, pero si tienes interés puedes consultar el módulo `group_by` (https://docs.ansible.com/ansible/7/collections/ansible/builtin/group_by_module.html).

Este ejemplo muestra una `fact` en un condicional when como el del ejercicio anterior:

```
tasks:
  - name: Apagar sistemas tipo Debian
    ansible.builtin.command: /sbin/shutdown -t now
    when: ansible_facts['os_family'] == "Debian"
```

Variables de inventario (globales)

Lo primero que vamos a hacer **es definir variables a nivel de inventario**, aplicadas a grupos. Por ejemplo podemos definir una variable que indique un usuario por defecto que vamos a usar en todos los servidores definidos en los ejercicios anteriores agregando un nuevo grupo que se llame como uno de los creados (en este caso `todos`) y definiendo la etiqueta `var` asociado al mismo. Por ejemplo:

```
[todos:vars]
usuario_default=<Tu UO>
```

Las variables de inventario **pueden tener mucha más complejidad**, como se puede observar aquí: https://docs.ansible.com/ansible/latest/user_guide/intro_inventory.html

Variables de playbook

En esta tarea vamos a **crear y usar una variable que contenga el nombre de la subcarpeta** a crear a la hora de desplegar la web de ejemplo de dos formas. Primero **usando el elemento** `vars` de *Ansible* **y luego definiéndola en un fichero** `.yml` **aparte** referenciado con `vars_files`. Debe comprobarse que en ambos casos la ejecución y acceso a los mismos es correcta. Se puede usar esta documentación como referencia: https://docs.ansible.com/ansible/latest/user_guide/playbooks_variables.html

Variables de task y bucles

El lenguaje *Ansible* admite bucles con la expresión `with_items` (y variantes) o `loop`: https://docs.ansible.com/ansible/latest/user_guide/playbooks_loops.html, lo que permite realizar una misma tarea sobre varios elementos sin necesidad de repetirla varias veces. Si bien los bucles pueden ser muy complejos, practicaremos sobre ellos con un bucle simple con `with_items` para copiar varios ficheros a distintos destinos.

Se trata por tanto de completar el siguiente esquema para copiar en un *endpoint* remoto una serie de ficheros de vuestra elección en la ruta que queráis

```
- name: Copiar ficheros a rutas
  copy:
    src: "{{ item.source }}"
    dest: "{{ item.destination }}"
    owner: <variable definida con vars o vars_files como antes>
    group: <variable definida con vars o vars_files como antes>
```

```
  with_items:
       - source: <<fichero 1 >>
         destination: <<ruta para el fichero 1>>
       - source: <<fichero 2 >>
         destination: <<ruta para el fichero 2>>
… (Más ficheros)
```

Precauciones con el uso de variables

Si un valor en el *playbook* empieza con el nombre de una variable (Ej.: `{{valor_calculado}}`) debes poner toda la expresión entre `“”` para crear una sintaxis YAML válida: https://docs.ansible.com/ansible/latest/user_guide/playbooks_variables.html . Por ejemplo:

```
    - name: Copiar ficheros a rutas
      copy:
        src: "{{ item.source }}"
        dest: "{{ item.destination }}"
        owner: ubuntu
        group: ubuntu
      with_items:
      - source: /home/ssiuser/config2.yaml
        destination: "{{dest_path}}"
      - source: /home/ssiuser/config3.yaml
        destination: "{{dest_path}}"
```

N3ANSIBLE_ADVMODULOS: Uso de módulos más avanzados *(0,4 puntos)*

Descripción de la actividad

Consiste en aprender a usar **módulos más avanzados para el manejo de *endpoints* remotos**, que te den aún más flexibilidad a la hora de elegir las operaciones que haces sobre los mismos.

Resultados esperados

Eres capaz de usar módulos avanzados para manipular el contenido de los *endpoints*: **comprobar contenidos** de ficheros y la **existencia** de estos (y directorios), **reemplazar líneas concretas** de ficheros, **descomprimir** ficheros y obtenerlos de lugares externos al host.

Con este ejercicio se pretende que **logres la autonomía completa** para introducir en un *endpoint* cualquier cosa que te haga falta para hacer una operación.

Información necesaria para su realización

Esta actividad se usará para que te familiarices con módulos de *Ansible* más avanzados que pueden serte de utilidad en escenarios típicos

- **Usar el módulo** `command` para ejecutar un escaneo de la seguridad de un *endpoint* con la herramienta `lynis`, asegurándose de que esta herramienta está instalada antes con las técnicas

ya vistas. https://docs.ansible.com/ansible/latest/collections/ansible/builtin/command_module.html

- **Usar el módulo `lineinfile`** para asegurarse de que en el fichero de configuración de `ssh` del *endpoint* no se permite hacer *login* interactivo a `root` (buscar qué línea lo hace y comprobar que tiene el valor correcto). También debe usarse para **modificar una línea de algún fichero** usando expresiones regulares para encontrarla, y para añadir una línea a otro fichero de vuestra elección, usando para ello los ejemplos que se proporcionan en la documentación oficial. https://docs.ansible.com/ansible/latest/collections/ansible/builtin/lineinfile_module.html
- Finalmente, vamos a **simular un procedimiento de instalación típico de un programa desde un fichero `.tar`** mediante el módulo `get_url`. Para ello:
  - **Descárgate un fichero `.tar`** que encuentres o crees tú al efecto de un servidor y que dentro tenga un *script* `install.sh` que instale o simule instalar algo (https://docs.ansible.com/ansible/latest/collections/ansible/builtin/get_url_module.html)
  - **Luego descomprímelo a un destino conocido** con el módulo `unarchive` (https://docs.ansible.com/ansible/latest/collections/ansible/builtin/unarchive_module.html)..
  - **Usa posteriormente el módulo `stat`** ya mencionado en un nivel anterior para comprobar si existe un fichero `install.sh` en la carpeta de destino
  - Hecho esto, **usa el módulo `command`** para ejecutarlo y asegurarse de que ha creado la carpeta `/tmp/sucess`.

N3ANSIBLE_VAULT: *Ansible Vault* *(0,2 puntos)*

Descripción de la actividad

Consiste en **aprender a usar *Ansible Vault*** para guardar adecuadamente secretos de tu infraestructura que necesites para desplegar algo o hacer alguna operación.

Resultados esperados

Eres capaz de entender cómo usar variables en un *playbook* cuyo valor esté cifrado con *Ansible Vault*.

Información necesaria para su realización

Como se dijo en la teoría, el uso de *Ansible Vault* es necesario para **prevenir el acceso público de datos secretos**, y por tanto es de capital importancia para la seguridad. En esta actividad se va a probar el uso a partir de un *playbook* sencillo para familiarizarse con él. El dato secreto a manejar son claves de usuario. Las claves de usuario se pueden guardar en texto plano (**algo a evitar a toda cosa en producción**) o hasheadas. En este último caso, y ante la posibilidad de poder averiguar su contenido mediante ataques de fuerza bruta, lo mejor es cifrar esas hash con las opciones del *Vault*.

Para usarlas, lo primero es crear dichas hashes siguiendo este procedimiento:

- Instalamos el paquete `whois` para poder usar el comando `mkpasswd`: `sudo apt install whois`

- Creamos una *hash* de la *password* para cada usuario que vamos a crear, pidiéndose esa clave por consola: `mkpasswd --method=SHA-512`
- El resultado se puede copiar y pegar para componer un archivo `.yml` como este. Recuerda que las claves hasheadas eran por ejemplo un requisito para el módulo `user` visto anteriormente:

```
admin_password:
$6$mMxygbmHl$Ko83RdLr2hLJnliOMKwrzsFk7TXqhReGjwEtUi2HjUTBAm8aE9qDesOViaDOEK57wcjPq6eDkkdXE
JrCMp/MC1
deploy_password:
$6$6CuOAbhtAu$FEiI7CPkk8d1KRn1xJ/Kb0B6deb6IoeosRPAggKc9U5NJ6rlY.e3Uq8bOU88JDIPHJHSFuk1Grdb
zA5v2yPJW0
```

- Una vez el archivo está terminado se puede cifrar con: `ansible-vault encrypt <ruta al fichero .yml anterior>`. Nos solicitará una clave de cifrado que debemos guardar en lugar seguro. El fichero ahora resulta ilegible:

```
operario@server1804:~/ansible-example/roles/users/vars$ cat main.yml
$ANSIBLE_VAULT;1.1;AES256
31643462336239663430363132323032646535376432303162333339666432656266363030343531
63666264376231656336383763663534363662333566313 70a393361356531666638356635303962
32306163666666613139313063326565376263636361626537353733623336653935383763393732
30633135366135396 50a3661663933613137313965363031373337376436623162653163333633437
33346339366337316364663761653438333936363834323631323530316362623330626130663266
30383332643338396533643531373731613633331316364663761336634346631333264383239313 4
30663062376566376531383334636631626535396465616131363665656165663762323463313763
62343963373262663031316337303931663764656234666463383235333835313237333032613461
32353732353964366162312324383131613030613635313637363766633036346132366166326263
62663735336166643964623137323430616631343430373038623930636564366461613037383463
66393565343333323433633262363031623536356331613831633132306366316134396664656139
34356639633161643861313730393334663564666531653230356463353366393665363735303464
64643733663064306435393830663364373730333066393436616333643864316661353430363539
38376633613931316432333430326261343230346538633036376238663137376338393532313838
65613764343131363733663534633832373730363762346139633238616530666235313531626636
34653161313038333335346332613030343238623065386136343833633266373134363462343833
34326661333935396632643039626136356437383265386237313864366533653130
operario@server1804:~/ansible-example/roles/users/vars$
```

Usar ahora esta información para hacer una nueva versión del *playbook* que creó usuarios en un nivel anterior, usando `vars_files` para enlazarlo desde el *playbook* y crear un conjunto de usuarios en un bucle que usen todos la misma *password* que está cifrada por el *Ansible Vault* en un fichero de variables.

N3ANSIBLE_ROLES1: Uso básico de roles *(0,3 puntos)*

Descripción de la actividad

Consiste en aprender a **hacer un uso básico de los roles de *Ansible***.

Resultados esperados

Eres capaz de convertir *playbooks* en roles y que se ejecuten para llegar al mismo resultado que antes

Información necesaria para su realización

Esta tarea consiste en comprobar que **tienes un conocimiento adecuado de los roles** y puedes usarlos de forma efectiva. Para completarla debes usar los dos *playbooks* que venimos usando hasta el momento (el de *Apache* y el de *MySQL*) y **separar sus elementos** como se ha explicado en la teoría para **convertirlos en dos roles independientes**, de forma que se lancen ambos roles finalmente para alcanzar el mismo resultado anterior.

Ten en cuenta que deben encapsularse en el rol todos los elementos que hemos añadido durante los ejercicios: variables, uso del *vault*, *handlers*, etc.

N3ANSIBLE_PROVWEB3CAPAS: Provisionamiento de una infraestructura de *front-back-SGBD* *(0,5 puntos)*

Descripción de la actividad

Consiste en poder **desplegar vía *Ansible* todo el *software* necesario para poder hostear una web en una estructura clásica de 3 capas**, como la vista en teoría.

Resultados esperados

Eres capaz de desplegar una infraestructura en tres capas como al SNI del tema 1 pero simplificada, donde uno de los *endpoints* está protegido por un *firewall* para solo admitir conexiones de una red concreta

Información necesaria para su realización

Esta última tarea de este bloque consiste en usar todos los conocimientos adquiridos hasta ahora para **desplegar en forma de roles que se ejecuten de forma conjunta** una configuración de una infraestructura en 3 *endpoints* del tipo que decidáis, de manera que haya una infraestructura en 3 capas similar a la vista en el tema 1 de teoría.

Además de esto, nos servirá para trabajar con **un módulo más complejo que maneja un *firewall***, `firewalld` (https://docs.ansible.com/ansible/latest/collections/ansible/posix/firewalld_module.html). No obstante, para ahorrar recursos, no se trata de imitar la infraestructura tal cual sino **instalar dicho *firewall* en una de los *endpoints* a modo de ejemplo**. Para desplegar esta infraestructura hay que seguir las siguientes indicaciones:

- **Crear un *endpoint*** que sea el `frontend`, con un `nginx` sirviendo la página web de ejemplo.
- **Otro *endpoint*** que sea un `backend`, con una instalación de la última versión de *Tomcat* (no hace falta introducir más código, salvo que lo tengáis de una aplicación que ya hayáis hecho u os hagáis descargado y no os haga perder tiempo).
- **Un *endpoint* final** con una instalación de *MySQL* del tipo de la vista antes, aunque no hace falta introducir información en la base de datos en esta ocasión. Además, en este mismo *endpoint* hay que seguir estos pasos para **configurarle un *firewall***, en este caso `firewalld`.
    - Instalar el módulo de *Ansible* `firewalld` y asegurarse de que su servicio asociado (`firewalld`) está arrancado

- Usar el módulo `firewalld` (ver ejemplos de su documentación) para conseguir lo siguiente
  - Crear una zona (`zone`) llamada `db` que sea permanente
  - Crear un origen (`source`) que **esté asignado a la red en la que está el servidor de** `backend`. Asignar dicho origen a la zona anterior y habilitarlo de forma **permanente.**
  - **Habilitar el puerto** (`port`) `3306/tcp` en la zona `db` de forma permanente.

Todos estos *endpoints* deben estar **correctamente actualizados** y **tener un usuario diferente** para entrar en sesión a los mismos.

Nivel 4 (Capitán de navío): Usos avanzados de Roles

N4ANSIBLE_ROLES2: Uso avanzado de roles y encadenamiento *(0,3 puntos)*

Descripción de la actividad

Consiste en **aprender a usar roles de *Ansible* existentes de manera combinada** para lograr operaciones más complejas con ellos.

Resultados esperados

Eres capaz de poner en marcha un servidor web encadenando roles existentes.

Información necesaria para su realización

Esta actividad consiste en poner en marcha el rol de `nginx` visto en teoría completo (con la parte de usuarios que leen sus claves del *Vault*, la configuración HBP5 adaptada y el soporte `ssl`), pero con las siguientes modificaciones:

- Hay una `pre_task` que actualiza todo el sistema de paquetes
- La variable `domain` se inicializa al valor de tu UO y debe ser accesible para todos los roles
- Se debe copiar la web entregada por defecto como web a servir

N4ANSIBLE_REFACTORROLES: Encapsulamiento de operaciones en roles *(0,4 puntos)*

Descripción de la actividad

Consiste en **entender la técnica de *refactoring*** clásica de programación, pero aplicada al código de los roles de *Ansible* para hacer roles pequeños de un propósito claro.

Resultados esperados

Eres capaz de refactorizar un rol existente para convertirlo en una secuencia de roles más pequeños que hagan solo una tarea concreta. Puedes responder a estas preguntas

- *¿Cómo crees que esto te puede ayudar en el futuro tener roles más pequeños que sólo tengan un función determinada?*

- *¿Encuentras algún paralelismo con la programación con lenguajes generalistas que has visto hasta el momento?*

Información necesaria para su realización

Esta tarea consiste en **refactorizar** la anterior de manera que se creen **roles mucho más granulares** que permitan reutilizarse más fácilmente en otras operaciones. Para ello se deberán **crear roles independientes que contengan sólo las operaciones indicadas**:

- **Incorporar el repositorio oficial** de `nginx` e instalarlo
- **Configurar** `nginx` con las indicaciones de H5BP simplificada tal y como vimos en la teoría y se hizo en el ejercicio anterior
- **Borrar la configuración del sitio** por defecto de *Nginx*
- **Crear y habilitar la configuración de un sitio** especificado por una variable `domain`, que debe estar disponible para todos los roles que creemos
- **Copiar un sitio dado** al sitio web indicado por la variable `domain`
- **Modificar los permisos de la raíz** de una web a un valor y usuario correctos para poder servirlos
- **Crear los dos usuarios** vistos que permitan administrar correctamente el sistema
- Como `post_task`, recargar *Nginx*

Enlazar todos estos roles más pequeños para que se consiga el mismo efecto que antes. Si ves que lo que has hecho repite código en alguno de los ficheros, **puedes considerar usar una plantilla de *Ansible***.

N4ANSIBLE_PRERREQROL: Prerrequisitos en roles *(0,2 puntos)*

Descripción de la actividad

Consiste en que entiendas cómo se pueden **establecer prerrequisitos antes de proceder a instalar un rol** de *Ansible* dado, creando **dependencias** entre roles.

Resultados esperados

Eres capaz de automatizar la instalación de los roles de los que depende uno dado a través de **un fichero de requisitos** que contenga todas esas dependencias. Puedes contestar a estas preguntas:

- *¿Puedes explicar qué similitud encuentras entre los roles de Ansible y los módulos o librerías de un lenguaje de propósito general cualquiera?*
- *¿Esta operación ejecuta el contenido de los roles, o solo los instala para que estén disponibles para usarlos posteriormente?*

Información necesaria para su realización

Una de las mayores ventajas de usar los roles ya disponibles en *Ansible Galaxy* es poder ver su código para aprender cómo funcionan, instalar todo o parte de estos si nos hace falta, o usarlos para reutilizar el código de todo lo que la comunidad es capaz de proporcionarnos. Si nos interesa un rol, podemos

instalarlo mediante `ansible-galaxy role install <nombre del rol que hemos visto en la Ansible Galaxy>`.

No obstante, si hay muchos roles a reutilizar la instalación de estos puede llegar a ser muy tediosa. Para evitar eso, podemos hacer uso de los **prerrequisitos de los roles**. Consiste en usar un fichero (normalmente llamado `requirements.yml`) donde, en forma de lista, se enumeren **todos los roles necesarios para hacer una operación**, de manera que si se llama a `ansible-galaxy install -r requirements.yml`, se instalarán todos los roles que contiene en una sola operación, librándonos de hacerlo de uno en uno. Probar a hacer esto con una serie de roles reales del *Ansible Galaxy*, que son los que se usarían en el futuro para construir una infraestructura LAMP, creando un `requirements.yml` con este contenido y probando que la instalación funciona correctamente:

```
---
roles:
  - name: geerlingguy.mysql
  - name: geerlingguy.apache
  - name: geerlingguy.php
  - name: geerlingguy.php-mysql
  - name: geerlingguy.java
```

⚒️ N4ANSIBLE_ROLINCLUDE: Inclusión de roles *(0,1 puntos)*

🕵️ Descripción de la actividad

Consiste en **aprender cómo funciona la inclusión de roles** en *Ansible*.

🎖️ Resultados esperados

Eres capaz de usar roles de *Ansible Galaxy* para instalar un sistema LAMP completo sobre el *endpoint* que quieras, maximizando así la reutilización de roles y entendiendo su utilidad. Puedes contestar a esta pregunta: *¿Qué harías ahora para mejorar el código de tus aprovisionadores Ansible de forma que sea más mantenible y reutilizable?*

📖 Información necesaria para su realización

El importar roles ya creados, y también el esfuerzo que hemos hecho anteriormente en refactorizarlos, nos lleva a poder hacer operaciones que simplemente **sean una secuencia determinada de llamadas a roles ya creados**, sin que haya código aparte del contenido en dicho roles. Puedes verlo como una forma de modularidad extrema: todo se construye a base de llamar de manera coordinada "ladrillos" que ya tienes.

Para probar esto, vamos a usar los roles instalados como prerrequisito anteriormente para **instalar un sistema LAMP completo** en un *endpoint* que creemos a propósito para esto y añadamos a nuestro inventario para este fin. Como ayuda, usar este fichero:

```
---
- hosts: all
  become: yes
```

```
  roles:
    - geerlingguy.mysql
    - geerlingguy.apache
    - geerlingguy.php
    - geerlingguy.php-mysql
```

N4ANSIBLE_CONDINCLUDE: Inclusión de elementos en función de la plataforma *(0,2 puntos)*

Descripción de la actividad

Consiste en aprender una técnica para **cargar diferentes valores en función de las características del *endpoint*** que se está administrando.

Resultados esperados

Sabes cómo cargar ficheros con variables o tareas en función del valor que tengan ciertas *facts*, por ejemplo para hacer *playbooks* o roles que se comporten de manera distinta en función del sistema operativo de cada *endpoint*

Información necesaria para su realización

Esta actividad consiste en aprovechar la información que nos dan las *facts* para poder hacer **inclusión de variables o tareas que dependan de la plataforma subyacente de los *endpoints*** que estamos administrando, en lugar de una inclusión directa de una tarea o valores fijos como se hace en casos más sencillos. Como hay tareas específicas para sistemas *Linux* y sistemas *Windows*, esto nos permite hacer roles que realmente sean multiplataforma con el mínimo esfuerzo, ya que podemos hacer preguntas primero sobre el endpoint que se va a manejar y luego, en función de los resultados de las facts del mismo, ejecutar una u otra cosa.

Para practicarlo, se pide que a partir de este fragmento de código:

```
- name: Incluir variables según el tipo de SO
  include_vars: "{{ansible_os_family}}.yml"
- name: Incluir tareas según el tipo de SO
  include_taks: tasks-"{{ansible_os_family}}.yml"
```

Se desarrolle un *playbook* o rol que **cargue un fichero de variables u otro en función de la familia del sistema operativo del *endpoint* administrado**, dentro del cual habrá una variable (mismo nombre en ambos ficheros) cuyo valor sea "*Estamos en un sistema tipo Debian*" o "*Estamos en un sistema tipo Red Hat*". Posteriormente, cargar un fichero de tareas u otro según el mismo criterio que haga lo siguiente:

- **Actualice la lista de paquetes** según el sistema de destino (`apt` para *Debian*, `yum` o `dnf` para *Red Hat*)
- **Muestre, usando el módulo** `debug`, la variable cargada anteriormente. https://docs.ansible.com/ansible/latest/collections/ansible/builtin/debug_module.html

Hecho esto, se pide **desarrollar una versión alternativa con tareas condicionales**, usando elementos como `when: ansible_facts['os_family'] == "Debian"`

Nivel 5 (Almirante): Operaciones avanzadas con Ansible

N5ANSIBLE_PROVADV: Provisionamiento de infraestructuras avanzadas *(0,5 puntos)*

Descripción de la actividad

Consiste en provisionar distintos *endpoints* con *Ansible*, **instalándoles el software que hace falta para otras asignaturas del máster.**

Resultados esperados

Esta actividad se completará cuando puedas crear satisfactoriamente las máquinas virtuales enumeradas (u otro tipo de *endpoints* que quieras, en función de los ejercicios que hayas hecho antes), que tendrán instalado automáticamente *software* que se usa en otras asignaturas del máster.

Información necesaria para su realización

En esta actividad se usarán todos los conocimientos anteriores para preparar máquinas virtuales (u otro tipo de *endpoints*, como contenedores LXD) **que tengan el siguiente *software* instalado con *Ansible***. Estos *endpoints* corresponderán a entornos de desarrollo que se van a usar en otras asignaturas del máster, por lo que se recomienda que **se instalen con soporte de GUI y con las extensiones de virtualización** del proveedor de esta que estamos utilizando, si es aplicable.

Si ya has realizado estas operaciones en el módulo de *Vagrant*, puedes hacer este ejercicio convirtiendo los *provisioners* a *Ansible* usando módulos siempre que sea posible (**no simplemente abusando del módulo `shell`**).

- **Arquitectura y Diseño de Sitios Web**: Un *Ubuntu 18* o superior con *Eclipse* y *Spring Tools for Eclipse*
- **Diseño y Programación de Interfaces de Usuario**: Un *Ubuntu 18* con *Visual Studio Code* y *NodeJS*
- **Gestores de Contenidos Web**: Un *Ubuntu 18* o superior con XAMPP
- **Programación Orientada a Objetos:** Un *Ubuntu 18* con *Pycharm*, *Python 3* y `virtualenv`
- **Sistemas de Persistencia de Objetos**: Un *Ubuntu 18* con *Hadoop*

N5ANSIBLE_DYNINV: Inventarios dinámicos *(0,3 puntos)*

Descripción de la actividad

Consiste en entender el **concepto de inventario dinámico de *Ansible*** y aprender a crear y usar uno.

Resultados esperados

Eres capaz de crear un inventario dinámico y usarlo, entendiendo su filosofía de uso y utilidad

Información necesaria para su realización

Hasta ahora hemos visto ejemplos donde se define un **inventario puramente estático**, donde tenemos los *endpoints* en los que vamos a trabajar en un fichero (bien el global o bien uno que pasamos en el momento de ejecutar el *playbook* o *rol*). No obstante, **esto no es viable si la infraestructura cambia a lo largo del tiempo**, con *hosts* que se arrancan y paran en función de la demanda de los clientes, como por ejemplo en aplicaciones de nube o bien en clústeres *Kubernetes* como los que vamos a ver en el próximo módulo. También es posible encontrarse con esta situación si nuestra arquitectura sirve *hosts* de **diferentes proveedores de nube**.

En estos casos, lo más oportuno es **crear un inventario de hosts dinámico**, es decir, que el inventario se genere a partir de algún elemento activo al que preguntemos por el contenido actual del inventario en un momento dado, y nos devuelva un inventario como el que hemos visto en teoría, bien en formato INI o formato YAML, **con los *endpoints* que deben funcionar en un momento dado**. *Ansible* tiene dos formas de hacer esto: A través de *plugins* específicos de *Ansible* que implementen esta función o **a través de *scripts***.

Los *scripts* deben ser programas en algún interprete instalado en el sistema cuyo resultado de su ejecución sea el inventario a crear escrito en la salida estándar, como si estuviera escrito en un fichero de inventario como los que hemos visto. ***Ansible* leerá esa salida estándar** y la usará como inventario a la hora de ejecutar una operación de provisionamiento.

De esta manera podemos construir nuestro inventario de *endpoints* mediante cálculos complejos hechos vía código, adaptando el inventario a cualquier necesidad o escenario que necesitemos. Además, para evitar problemas de rendimiento, **podemos definir todas las variables asociadas al *host*** en un elemento de nivel global llamado `_meta`, que permitirá devolver las variables asociadas a un *host* en una sola ejecución del *script*.

Para probar esta técnica de construir inventarios dinámicos, en esta actividad se pide modificar este *script* de *Python* para que devuelva en la salida impresa por el programa un inventario equivalente al que definimos en el nivel 1 de este boletín de actividades, y ejecutarlo luego con `ansible -i <nombre del fichero Python>.py all -m ansible.builtin.ping` para comprobar que todos los *endpoints* definidos están accesibles, es decir, obteniendo el mismo resultado que en dicha actividad.

```
#!/usr/bin/env python3

'''
Ejemplo de inventario dinámico en forma de script Python para Ansible
'''
import argparse
import json

class InventarioDePrueba(object):

    def __init__(self):
        """
        Aqui se puede hacer de todo: leer de entrada estándar, ficheros,
        Llamar a servicios externos...
```

```
        Aunque simplemente escribamos un inventario en forma de string podemos hacer un
        programa todo lo complejo que queramos para construir dicho inventario.
        """
        print(json.dumps(self.inventario_ejemplo()));

    # Example inventory for testing.
    def inventario_ejemplo(self):
        return {
            'group': {
                'hosts': ['192.168.28.71', '192.168.28.72'],
                'vars': {
                    'ansible_user': 'asouser',
                    'ansible_ssh_private_key_file':
                        '~/.vagrant.d/insecure_private_key',
                    'ansible_python_interpreter':
                        '/usr/bin/python3',
                }
            },
            '_meta': {
                'hostvars': {
                    '192.168.28.71': {
                        'host_specific_var': 'host1'
                    },
                    '192.168.28.72': {
                        'host_specific_var': 'host2'
                    }
                }
            }
        }

# Mostrar el inventario.
InventarioDePrueba()
```

Este es un tema muy avanzado al que **se le puede sacar más partido**. Si quieres sabes más acerca del mismo se recomienda consultar estos enlaces

- https://docs.ansible.com/ansible/latest/user_guide/intro_dynamic_inventory.html
- https://docs.ansible.com/ansible/latest/dev_guide/developing_inventory.html

⚒️ N5ANSIBLE_ANSVAGRANT: *Vagrant* y *Ansible* *(0,2 puntos)*

🕵️ Descripción de la actividad

Consiste en **usar *Ansible* como *provisioner*** de una máquina *Vagrant*.

(NOTA: Es necesario haber terminado un Nv.1 de Vagrant para terminar con garantías este ejercicio)

🎖️ Resultados esperados

Eres capaz de combinar el uso de *Vagrant* y *Ansible* para provisionar una máquina *Vagrant*

Información necesaria para su realización

Este ejercicio consiste en transformar el *provisioner* básico que usamos en el Nv.1 de *Vagrant* vía *script* de *bash* **en un equivalente exacto hecho con un *playbook* o rol de *Ansible***, priorizando el uso de módulos de *Ansible* frente a la invocación de comandos directa a través del módulo `shell`.

Para ello puede usarse esta documentación: https://www.vagrantup.com/docs/provisioning/ansible_intro y basarse en esta `Vagrantfile` de ejemplo.

```
# -*- mode: ruby -*-
# vi: set ft=ruby :
Vagrant.configure("2") do |config|
  config.vm.box = "generic/ubuntu1804"

  config.vm.provider :virtualbox do |v|
    v.memory = 1024
  end

  # provisioner Ansible
  config.vm.provision :ansible do |ansible|
    ansible.playbook = "playbook.yml"
    ansible.become = true
  end
end
```

Ten en cuenta que *Ansible* solo está disponible en entornos *Linux* de manera nativa, por lo que para hacer este ejercicio **necesitarás un *Linux***. Esto se puede hacer creando una máquina virtual *Linux* pequeña en la máquina *Linux* que os dan en nube (**si tiene las extensiones de virtualización por *hardware* que *VirtualBox* necesita, permitiendo virtualización anidada**), en una local (de nuevo, si la MV tiene esas extensiones *hardware* de virtualización anidada activas, cosa que debe soportar tu CPU), o en un host *Linux* que tengáis en un PC propio vuestro o al que podáis acceder.

N5ANSIBLE_ANSIDOCK: Docker y *Ansible* *(0,2 puntos)*

Descripción de la actividad

Consiste en aprender a **instalar *Docker* gracias a las funciones de aprovisionamiento que nos da *Ansible***. Además de eso, también debes aprender a **hacer un uso básico de *Docker*** mediante *Ansible*.

(NOTA: Es necesario haber terminado un Nv.2 de Docker para terminar con garantías este ejercicio)

Resultados esperados

Eres capaz de usar *Ansible* para instalar *Docker* en un *endpoint*, arrancar un contenedor a partir de una imagen y también para provisionar el contenedor arrancado. Puedes contestar a esta pregunta: *¿Distingue Ansible entre máquinas virtuales y contenedores Docker a la hora de trabajar con endpoints?*

Información necesaria para su realización

Esta tarea consiste en hacer lo siguientes pasos, bien usando los módulos ya vistos o algunos de los **módulos especiales que *Ansible* tiene para interactuar con *Docker***: https://docs.ansible.com/ansible/latest/collections/community/docker/index.html :

- **Instalar *Docker*** en un *endpoint* siguiendo las instrucciones de teoría para ello
- **Instalar una imagen que hayamos creado en el módulo de *Docker*** que tenga una conexión de SSH operativa (**NOTA**: por defecto no es posible hacer login remoto como `root` en SSH, **debes crear un usuario *sudoer* sin *password*** en el contenedor destino)
- **Automatizar la conexión vía SSH** a ese contenedor *Docker*, o bien generar la imagen ya con ese soporte incluido
- **Provisionar un contenedor** de la imagen creado con *Ansible* instalando algún paquete que tú quieras. El contenedor debe tener una IP conocida para poder colocarlo en un inventario.

N5ANSIBLE_ANSILXD: LXD y *Ansible* *(0,2 puntos)*

Descripción de la actividad

Consiste en aprender a **instalar *LXD* gracias a las funciones de aprovisionamiento que nos da *Ansible***. Además de eso, también debes aprender a **hacer un uso básico de *LXD*** mediante *Ansible*.

(NOTA: Es necesario haber terminado un Nv.2 de LXD para terminar con garantías este ejercicio)

Resultados esperados

Eres capaz de usar *Ansible* para arrancar un contenedor *LXD* en un *endpoint* a partir de una imagen y también para provisionar el contenedor arrancado. Puedes contestar a esta pregunta: *¿Distingue Ansible entre máquinas virtuales y contenedores LXD a la hora de trabajar con endpoints?*

Información necesaria para su realización

Esta tarea consiste en hacer lo siguientes pasos, bien usando los módulos ya vistos o algunos de los que *Ansible* tiene especiales para interactuar con *LXD*: https://docs.ansible.com/ansible/latest/collections/community/general/lxd_profile_module.html

- **Instalar *LXD*** en un *endpoint* siguiendo las instrucciones de teoría para ello
- **Instalar una imagen** que hayamos creado en el módulo de *LXD* que tenga una conexión de SSH operativa (**NOTA**: por defecto no es posible hacer login remoto como `root` en SSH, debes crear un usuario sudoer sin password en el contenedor destino)
- **Automatizar la conexión vía SSH** a ese contenedor *LXD*, o bien generar la imagen ya con ese soporte incluido
- **Provisionar un contenedor** de la imagen creado con *Ansible* instalando algún paquete que tú quieras. El contenedor debe tener una IP conocida para poder colocarlo en un inventario.

Insignias y Autoevaluación

Esta asignatura hace uso de insignias. Esto quiere decir que, cuando vayas desbloqueando niveles de maestría en la tecnología, podrás obtener **una insignia pública** que luego podrás usar en tu currículum gracias al soporte de estas del campus virtual. Este enlace te enseña a usar esta característica para poder unirlas a tu currículum: https://docs.moodle.org/all/es/Insignias. Cada insignia pública **lleva adjuntos los textos de las habilidades que desbloquea** que puedes consultar en la siguiente tabla, para que cualquiera los vea si decides usarlas y sepa lo que has conseguido hacer, validado por mí. De esta manera, mi intención es ayudarte a aumentar tus posibilidades de contratación en un futuro, **exponiendo claramente lo que has podido hacer en esta asignatura** referente a cada tecnología sobre la que has decidido evaluarte.

Rango de Maestría en la Tecnología y Habilidades Demostradas		Insignia Pública Obtenida
Nivel 1: Cabo de *Ansible*		
	Puede instalar correctamente *Ansible*	
	Puede crear un inventario de *endpoints* remotos, uniendo también la local, y distinguiendo ambas, comprobando luego que la conexión con la misma es posible ***(2 ejercicios)***	
	Puede ejecutar comandos a voluntad sobre *endpoints* vía *Ansible* con el módulo *shell*	
	Puede instalar programas con *Ansible* y el módulo [redacted]	
	Puede crear *playbooks* sencillos que gestionen el software instalado en un *endpoint*	
Nivel 2: Sargento de *Ansible*		
	Puede obtener *facts* que describen cualquier *endpoint* remoto	
	Puede optimizar las conexiones SSH durante la ejecución de *Ansible* y consultar sus logs e información de ficheros desde línea de comandos	
	Controla cómo interaccionar con *endpoints* remotos con módulos de *Ansible* que modelen operaciones típicas, como actualizar *endpoints*, gestionar usuarios y grupos, ficheros / directorios y tareas programadas	
	Puede provisionar con *Ansible* una infraestructura de dos *endpoints* web-SGBD	
Nivel 3: Teniente de Navío de *Ansible*		
	Puede controlar el orden de ejecución de tareas para que unas vayan a continuación de otras y también lanzar tareas asíncronas y esperar a que terminen	
	Entiende como se pueden definir y usar variables en diferentes ámbitos, del más global al más local, y usarlas en bucles	

	Controla como usar módulos que permitan interaccionar con los *endpoints* de forma más avanzada, como por ejemplo comprobar y modificar contenidos de ficheros, descomprimir ficheros, obtener ficheros de sistemas remotos y otras operaciones que permitan tener más libertad para interaccionar con los *endpoints*	
	Puede cifrar secretos con *Ansible Vault* adecuadamente	
	Puede transformar un *playbook* en un rol equivalente	
	Es capaz de provisionar una infraestructura *front-back-sgbd* automáticamente con *Ansible*, además de activar un *firewall* por seguridad	
Nivel 4: Capitán de Navío *Ansible*		
	Puede encadenar la ejecución de varios roles	
	Puede refactorizar los roles en roles más pequeños que permitan reutilizarlos más fácilmente	
	Puede instalar prerrequisitos de roles	
	Puede usar módulos de *Ansible Galaxy* de forma combinada para lograr hacer operaciones más complejas	
	Sabe implementar una vía sencilla de desarrollar roles y *playbooks* multiplataforma	
Nivel 5: Almirante *Ansible*		
	Puede provisionar con *Ansible* infraestructuras usadas en contextos típicos	
	Entiende y puede generar inventarios dinámicos con *scripts*	
	Puede combinar *Vagrant* y *Ansible* para crear máquinas virtuales provisionadas según tus necesidades	
	Puede combinar *Docker/LXD* y *Ansible* para instalar el *software* de contenedores donde sea necesario e interaccionar con él para lanzar y provisionar contenedores ***(2 ejercicios)***	
	**Cruz con el distintivo amarillo en *Ansible***: Ha completado con éxito **4,8 puntos** (aproximadamente el 80%) de las tareas correspondientes de los 5 niveles de la parte de *Ansible*, demostrando así su maestría en esta tecnología y estar preparado para usarla en tu futuro trabajo de forma efectiva.