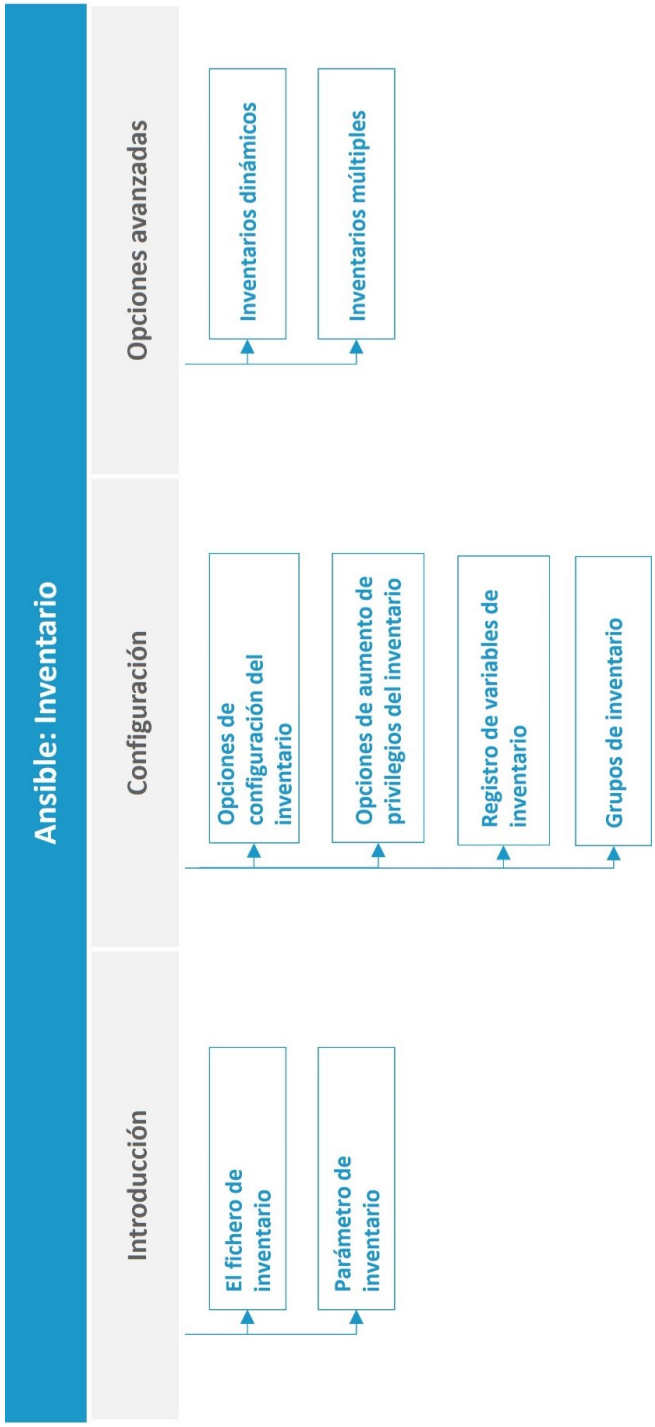
Herramientas de Automatización de Despliegues

Tema 6. Ansible. Inventario

Índice

Esquema



6.1. Introducción y objetivos

En la gestión de la configuración, la herramienta que vayas a utilizar necesita saber qué máquinas debe gestionar, sobre las que aplicará la configuración. Esto se conoce como «inventario». Sin conocer el inventario, podrías llegar a tener un conjunto de *playbooks* que definen tu estado de configuración deseado del entorno, pero no sabrías sobre qué máquinas se debería aplicar.

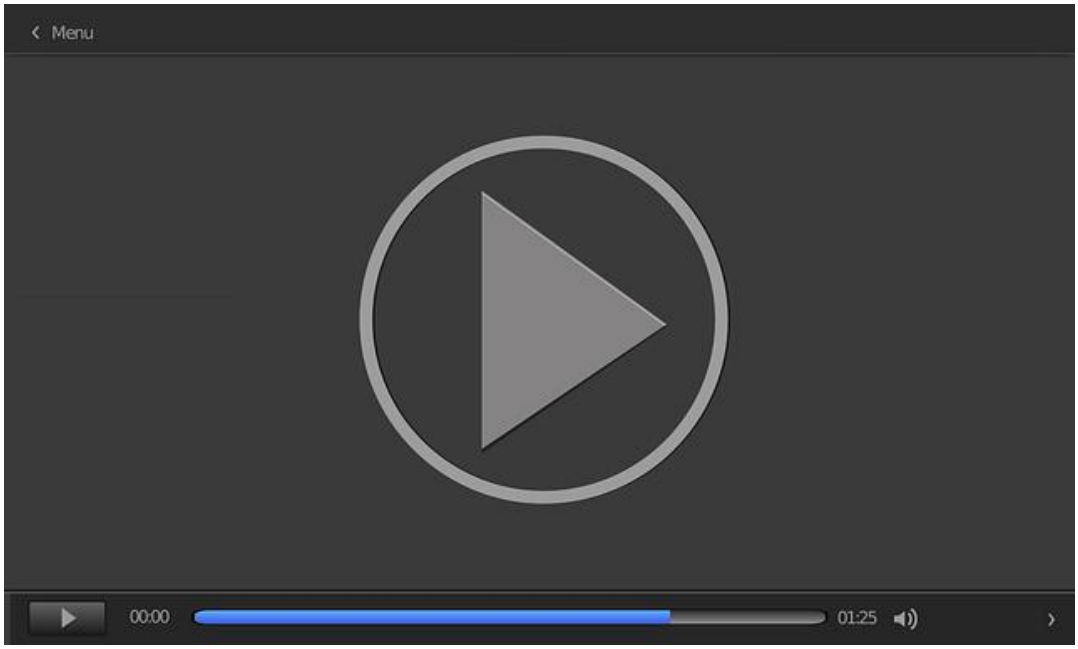
En herramientas como Puppet y Chef, un servidor centralizado se encarga de almacenar esta información. Dado que con Ansible no contamos con un servidor centralizado, necesitamos otro modo de poder proporcionar esta información al código que se ejecuta, para asegurarnos de que se alcanza el estado deseado en las máquinas correspondientes. Aquí es donde entra en juego el fichero de inventario.

Los objetivos que se pretenden conseguir en este tema son los siguientes:

- Definir el inventario en Ansible.
- Conocer las opciones de configuración.
- Conocer las distintas variantes de definición del inventario.

Ideas clave

A continuación, puedes ver el vídeo *Ansible vs. Puppet vs. Chef*:



Accede al vídeo:

https://unir.cloud.panopto.eu/Panopto/Pages/Embed.aspx?id=f0ad7df5-d3fb-487f-9fbc-abb60092fe7b

6.2. El fichero de inventario

Si no se le indica de otro modo, Ansible leerá el fichero de inventario de la ruta . Sin embargo, el definir tu inventario en este fichero no está recomendado. Lo más conveniente es mantener un fichero de inventario distinto por cada proyecto que tengas y pasarlo como parámetro al comando o mediante la opción . A continuación, se presenta un ejemplo de cómo puedes pasar como parámetro un fichero de inventario personalizado al comando ansible para que ejecute sobre el mismo el módulo :

```
ansible all –i /path/to/inventory –m ping
```

El fichero de inventario en Ansible puede implementarse en un fichero INI, o mediante un JSON. Los ejemplos más habituales en la web utilizan un fichero INI, mientras que el fichero JSON se utiliza típicamente para generar el inventario dinámicamente. Usar el formato INI hace que los ficheros de inventario sean normalmente bastante simples. Pueden ser tan sencillos como una simple lista de nombres de *hosts* en los que ejecutar. A continuación, se muestra un fichero de inventario sencillo:

```
host1.example.com
```

```
host2.example.com
```

```
host3.example.com
```

```
192.168.9.29
```

En este ejemplo se define una lista de cuatro hosts sobre los que se quiere ejecutar. Ansible ejecutará sobre cada uno de ellos por turnos, siguiendo el orden establecido, desde hasta . Esta es la forma más sencilla de fichero de inventario que puedes usar,

en la que no hay información adicional para ningún *host* ni agrupaciones, sino sencillamente una lista de *hosts* sobre la que queremos ejecutar Ansible.

Si ejecutas el proceso SSHD en un puerto que no sea el estándar, también puedes especificarlo en tu fichero de inventario. No tienes más que añadir dos puntos seguidos del número de puerto detrás de tu nombre de *host*, como por ejemplo:

```
host1.example.com:50822
```

Si trabajas con un elevado número de servidores que siguen un patrón común para el nombre, puedes utilizar rangos en Ansible para identificarlos, dado que permite definir rangos en los nombres, dentro del fichero de inventario. En lugar de especificar cada *host* uno por uno, puedes usar esta funcionalidad de expansión de rangos para definirlos todos de una vez, tal como, por ejemplo:

```
host[1:3].example.com
```

Esta expresión equivale a:

```
host1.example.com
```

```
host2.example.com
```

```
host3.example.com
```

Para algunos casos de uso, una simple lista de *hosts* a los que conectar no es suficiente. Las máquinas con las que conectas también pueden tener diferentes usuarios de sistema habilitados y pueden no utilizar por defecto tu misma clave SSH. Afortunadamente, Ansible también permite especificar las opciones de configuración SSH que se quieran utilizar a la hora de establecer una conexión. En la siguiente sección veremos cómo hacerlo.

Parámetro de inventario al ejecutar Ansible

Hasta este punto, hemos utilizado siempre para aprovisionar con Ansible. Esto es sencillamente un envoltorio que ejecutará el comando que corresponda dependiendo del aprovisionador que utilices. En este caso, se utilizará el comando para aprovisionar mediante Ansible el *playbook* que se indique a la instancia Vagrant con la que se trabaja. Para otras herramientas como Puppet o Chef, Vagrant se encargará de ejecutar el comando correspondiente para aprovisionar con ellas.

No obstante, no debes depender de Vagrant siempre a la hora de ejecutar Ansible porque estés trabajando en un entorno donde únicamente haya una máquina virtual ejecutando SSH. Para ejecutar en otros entornos, necesitas ser capaz de ejecutar Ansible en todas las máquinas que contengan sin requerir el uso de Vagrant.

Para poder probar la ejecución manual de Ansible, necesitas habilitar en tu la gestión de red privada para poder así acceder por SSH a la máquina virtual. Esto se hace editando el fichero y descomentando o añadiendo la siguiente línea:

```
config.vm.network "private_network", ip: "192.168.33.10"
```

Una vez guardados los cambios en el fichero, ejecuta para reiniciar tu máquina y habilitar esta red. Esto proporcionará a la máquina virtual creada una IP que podrás utilizar para conectar con ella. Una vez que tengas dirección IP, puedes manejarla como si no fuera una máquina gestionada por Vagrant, sino otra máquina accesible en la red en la que necesitas ejecutar Ansible. Esto podría ser una máquina virtual que se encuentra alojada en tu mismo equipo, o en algún lugar de la nube. Para Ansible, es sencillamente una máquina a la que puede conectarse.

El comando permite proporcionar múltiples parámetros, pero solo son necesarios unos pocos para poder ejecutar Ansible. El siguiente ejemplo utiliza el parámetro “ ” con el fichero de inventario, para indicar los *hosts* con los que trabajar, y el nombre del *playbook* a ejecutar:

```
ansible-playbook –i <inventory_file> provisioning/playbook.yml
```

El fichero de inventario que se incluye a continuación es un ejemplo que puedes probar para a ejecutar Ansible manualmente. Contiene la dirección IP de la máquina virtual a la que debe conectarse, el usuario de acceso y la ruta al archivo de clave privada a utilizar para la conexión mediante SSH. La clave privada es equivalente a una contraseña, y proporciona una firma criptográfica privada que se utiliza para verificar tu identidad, o la de la máquina *host* que se conecta, sin necesidad de introducir ningún valor de forma interactiva:

```
192.168.33.10 ansible_user=vagrant
```

```
ansible_ssh_private_key_file=.vagrant/machines/default/virtualbox/private_key
```

Todo su contenido se encuentra en una única línea. Guardaremos el fichero con el nombre y, a continuación, ejecutaremos el siguiente comando, que nos deberá devolver la misma salida que obtuviste al ejecutar , dado que equivale al fichero de inventario que genera y utiliza Vagrant al ejecutar Ansible sobre la máquina virtual.

```
ansible-playbook -i inventory provisioning/playbook.yml
```

6.3. Opciones de configuración del inventario

Al ejecutar Ansible directamente contra tu máquina virtual Vagrant, has necesitado configurar y , para así utilizar las credenciales correctas que había generado Vagrant. La siguiente tabla muestra las opciones de configuración más comunes.

El primer conjunto de opciones que se listan en la tabla están relacionadas con la conexión SSH que Ansible utiliza para ejecutar los comandos en los servidores remotos. Para la mayor parte de los servidores solo será necesario configurar estas dos primeras propiedades.

Opciones de configuración del inventario	
Opción de configuración	**Explicación**
`ansible_host`	Permite utilizar un nombre diferente a su *hostname* real para un *host* en el fichero de inventario y en los *playbooks*. Esto es útil cuando quieres referirte a una máquina cuya dirección IP es dinámica y puede cambiar. Por ejemplo, en el fichero de inventario: `alpha ansible_host=192.168.33.10` Te permite referirte a la máquina «alpha» en cualquier parte, y Ansible se conectará a la dirección IP 192.168.33.10 cuando necesite acceder a ella.

`ansible_user`	El usuario SSH con el que acceder a la máquina remota:
	`ansible_user=Michael`
	
	Sería equivalente a:
	`ssh Michael@host1.example.com`

`ansible_port`	El puerto en el que tu servidor SSH está a la escucha:
	`ansible_port=50822`
	
	Sería equivalente a:
	`ssh host1.example.com -p 50822`

`ansible_ssh_private_key_file`	El fichero de clave SSH utilizado para acceder:
	`ansible_ssh_private_key_file=/path/to/id_rsa`
	
	Sería equivalente a:
	`ssh -i /path/to/id_rsa`

`ansible_ssh_pass`	Si el usuario con el que conectas a la máquina requiere una contraseña, se especifica en el fichero de inventario mediante esta propiedad. **Nota:** esto es muy inseguro, y deberías utilizar «autenticación por clave SSH» o utilizar la opción `--ask-pass` en la línea de comandos para proporcionar la contraseña en tiempo de ejecución.

Tabla 1. Opciones de configuración del inventario. Fuente: elaboración propia.

ansible_ssh_common_args	Parámetros adicionales que proporcionar a la llamada a cualquiera de los comandos SSH, SFTP o SCP. Por ejemplo:
	`ansible_ssh_common_args='-o ForwardAgent=yes'`
	Es equivalente a:
	`ssh -o ForwardAgent=yes host1.example.com`

ansible_ssh_extra_args	Igual que `ansible_ssh_common_args`, pero los argumentos que especifiques solo se usarán cuando Ansible ejecute SSH.
ansible_sftp_extra_args	Igual que `ansible_ssh_common_args`, pero los argumentos que especifiques solo se usarán cuando Ansible ejecute SFTP.
ansible_scp_extra_args	Igual que `ansible_ssh_common_args`, pero los argumentos que especifiques solo se usarán cuando Ansible ejecute SCP.

Tabla 1. Opciones de configuración del inventario. Fuente: elaboración propia.

El ejemplo de fichero de inventario a continuación utiliza alguna de estas opciones:

```
alpha.example.com ansible_user=bob ansible_port=50022

bravo.example.com ansible_user=mary

ansible_ssh_private_key_file=/path/to/mary.key

frontend.example.com ansible_port=50022

yellow.example.com ansible_host=192.168.33.10
```

Con esto se establece un puerto alternativo para las máquinas y , usuarios específicos con los que accede a y , proporciona el fichero de clave privada para y por último define que el nombre es realmente la dirección IP 192.168.33.10. Esta información adicional es excesiva para un fichero de inventario tan pequeño, pero sirva como ejemplo ilustrativo.

Esto no acaba aquí y hay incluso más opciones disponibles. La siguiente tabla muestra opciones de aumento de privilegios que puedes utilizar en tus ficheros de

inventario.

Opciones de aumento de privilegios del fichero de inventario	
Opción de configuración	**Explicación**
`ansible_become_method`	Indica el método que se utilizará para obtener privilegios de superusuario. Por defecto se usa `sudo`, pero puede ser cualquiera de los siguientes métodos: `sudo`, `su`, `pbrun`, `pfexec` o `doas`. Algunas como `pbrun` son herramientas comerciales de seguridad, que no se usarán salvo en casos muy concretos. `sudo` es la opción más adecuada para la mayoría de los casos.
`ansible_become_user`	Por defecto, `become` te elevará a nivel root. Si dispones de otro usuario con los permisos adecuados para realizar la tarea a ejecutar, puedes especificarlo con esta opción de configuración para usarlo en lugar de `root`, Esto es equivalente a ejecutar `sudo` pasándole un usuario como parámetro: `sudo -u myuser`.

Tabla 2. Opciones de aumento de privilegios del fichero de inventario. Fuente: elaboración propia.

Estas opciones relacionadas con el aumento de privilegios pueden definirse en el fichero de inventario, pero no se aplicarán salvo que establezcas en tus *playbooks*.

Dado el siguiente fichero de inventario, y , ambos usarán el usuario cuando se establezca en el *playbook*. usará el usuario y usará , que es el usuario por defecto:

```
alpha.example.com ansible_become_user=automation
```

```
bravo.example.com ansible_become_user=automation
```

```
frontend.example.com ansible_become_user=ansible
```

```
yellow.example.com
```

Cabe destacar que este usuario no es el que se utiliza para acceder a la máquina (se

debe definir el argumento para eso). Este usuario es con el que se ejecutará al usar en tu *playbook* o tarea. Será tu responsabilidad asegurarte de que el usuario al que cambies tenga los permisos suficientes para ejecutar la tarea que corresponda.

6.4. Registro de variables de inventario

Aparte de poder establecer las variables especiales de Ansible en el inventario, tal como o , puedes también definir cualquier variable que quieras usar a continuación en un *playbook* o plantilla. Sin embargo, definir variables en el fichero de inventario no es la solución más adecuada generalmente. Hay muchos otros lugares donde definir variables en un *playbook* y la mayoría serán más adecuados que este.

Si estás pensando añadir una variable a un fichero de inventario, analiza si este dato debería estar en un inventario realmente. ¿Se trata de un valor por defecto? ¿Es algo relacionado con un tipo específico de máquinas o con una aplicación específica? Más adelante veremos otros sitios más recomendables donde puedes definir estas variables.

No obstante, si acabas decidiendo que el fichero de inventario es el sitio adecuado donde definir tu variable, es muy sencillo definirla. Por ejemplo, si quieres una variable que se llame accesible desde tu *playbook*, puedes definir un *host* como se muestra a continuación:

```
host1.example.com vhost=staging.example.com
```

Esto será útil en ciertas ocasiones, como por ejemplo cuando ejecutas tus sitios web de *staging* y producción desde la misma máquina y necesitas mediante el fichero de inventario diferenciar con cuál de los entornos estás trabajando. Si haces cambios de base de datos en tu *playbook*, pero no quieres impactar el despliegue en producción cuando pruebas en *staging*, puedes usar los siguientes ficheros de inventario para especificar la base de datos que quieres utilizar en cada caso:

```
$ cat staging-inventory
```

```
alpha.example.com database_name=staging_db
```

```
$ cat production-inventory
```

```
alpha.example.com database_name=prod
```

Como podemos ver, la variable estará disponible desde tu *playbook* con el valor correspondiente en cada entorno, de manera que puedas ejecutar lo que necesites en la base de datos oportuna. Lo único que debes hacer es asegurarte de proporcionar el fichero de inventario adecuado al ejecutar Ansible.

Por ejemplo, para aplicar el *playbook* en el entorno de producción, usaremos:

```
ansible-playbook –i production-inventory playbook.yml
```

6.5. Grupos de inventario

Hasta aquí hemos estado utilizando una lista simple de máquinas sobre las que ejecutar Ansible. No obstante, esto no se corresponde con la realidad, donde solemos tener balanceadores de carga, servidores web, servidores de aplicaciones, bases de datos, etc. Se hace necesario poder agrupar estos conjuntos de servidores por tipo y poder referenciarlos como a un único grupo. Ansible permite definir grupos de inventario para dar soporte a este caso de uso.

Para definir un grupo de servidores en el fichero de inventario se utilizan las cabeceras de sección INI, incluyendo su nombre entre corchetes, tal como especifica el formato de archivos INI:

```
[web]

host1.example.com

host2.example.com

[database]

db.example.com
```

En este fragmento de inventario, definimos dos *hosts* agrupados como servidores web, y otro en el grupo . En el formato INI los corchetes sirven como marcadores de sección, y lo que sería el nombre de la sección (lo que se incluya dentro de los corchetes) será el nombre del grupo.

Cuando ejecutamos Ansible podemos especificar en qué grupos de *hosts* se deben ejecutar nuestros comandos. Hasta ahora, hemos utilizado el grupo especial all para indicar que se ejecute en todos los *hosts* listados. Ahora podríamos especificar web o para que Ansible ejecute solo en el grupo de servidores indicado. El siguiente

comando ejecutará el módulo únicamente en el grupo web:

```
ansible web –i /path/to/inventory –m ping
```

También puedes establecerlo en un *playbook* cambiando simplemente el valor de *hosts*: al principio de tu *playbook*, tal como:

```
- hosts: web

tasks:

- ping:
```

De la misma manera que puedes establecer variables para máquinas específicas, también puedes establecer variables para los grupos. Para ello, utiliza una cabecera especial con el nombre de grupo y el sufijo en tu fichero de inventario:

```
[web:vars]

apache_version=2.4

engage_flibbit=true
```

Las variables así definidas estarán disponibles en la ejecución Ansible para cualquier máquina del grupo web. ¡También puedes incluso crear grupos de grupos!

Imaginemos que tienes un grupo de máquinas en producción que son una mezcla de servidores CentOS 6 y CentOS 7. El fichero de inventario podría parecerse al siguiente:

```
[web_centos6]

host1.example.com
```

```
host2.example.com

[web_centos7]

shinynewthing.example.com

[database_centos]

database.example.com

[reporting_centos7]

reporting.example.com
```

Si necesitas ejecutar algo únicamente en las máquinas CentOS 6, típicamente tendrías que hacerlo sobre ambos grupos, y .

En lugar de ello, puedes crear un grupo de grupos utilizando el sufijo en tu nombre de grupo:

```
[centos6:children]

web_centos6

database_centos6

[centos7:children]

web_centos7

reporting_centos7
```

Con este grupo de grupos ya solo tendrás que apuntar a los *hosts* centos6 si solo necesitas ejecutar en los servidores CentOS 6. Es más, puedes también definir variables para este nuevo grupo, como harías con cualquier otro grupo:

```
[centos6:vars]

apache_version=2.2

[centos7:vars]

apache_version=2.4
```

Los grupos son una herramienta muy potente y pueden ser bastante útiles en el futuro cuando definamos variables fuera del fichero de inventario.

6.6. Un ejemplo de inventario

A continuación, se muestra un ejemplo de un despliegue ficticio que contiene un servidor web, una base de datos y un balanceador de carga. El entorno se ha ido creando a lo largo de un tiempo, por lo que existen varias versiones de sistemas operativos disponibles, con diferentes usuarios y métodos de acceso a los *hosts*:

```
[web_centos6]

fe1.example.com ansible_user=michael

ansible_ssh_private_key_file=michael.key

fe2.example.com ansible_user=michael

ansible_ssh_private_key_file=michael.key

[web_centos7]

web[1:3].example.com ansible_user=automation ansible_port=50022

ansible_ssh_private_key_file=/path/to/auto.key

[database_centos7]

db.example.com ansible_user=michael

ansible_ssh_private_key_file=/path/to/db.key

[loadbalancer_centos7]

lb.example.com ansible_user=automation ansible_port=50022
```

```
ansible_ssh_private_key_file=/path/to/lb.key

[web:children]

web_centos6

web_centos7

[database:children]

database_centos7

[loadbalancer:children]

loadbalancer_centos7
```

Las máquinas más antiguas funcionan con CentOS 6 y usan la cuenta personal de Michael como inicio de sesión de Ansible. Las máquinas más modernas tienen usuarios de automatización propios, con claves privadas definidas. El *host* de base de datos utiliza una cuenta personal y tiene una clave SSH diferente. Por último, el balanceador de carga dispone de un usuario de automatización, pero en vez de utilizar su clave de automatización, tiene su clave propia del balanceador de carga.

Finalmente, se definen grupos de grupos para hacer posible apuntar a todos los servidores web o a todos los servidores de base de datos como a un grupo, independientemente de la versión de CentOS. Aunque el grupo solo tiene un grupo hijo por ahora, puedes querer agregar *hosts* con CentOS 7 en un futuro. Desde luego, tener un grupo preparado para usarse cuando sea necesario puede ahorrarnos mucho tiempo.

Observando este fichero de inventario, puede verse que hay siete *hosts* en este despliegue (fe1, fe2, web1, web2, web3, db y lb), qué usuario utilizará Ansible para acceder y qué clave usará para ello. Incluso se puede saber el puerto en el que está

ejecutando el demonio SSH.

Un fichero de inventario bien escrito no debe ser simplemente algo que Ansible debe usar, sino que debe servir también como documentación propia del entorno, para cuando necesites referirte a ella, o mostrárselo a alguien.

6.7. Opciones avanzadas de inventario

Inventarios dinámicos

Cuando únicamente tienes uno o dos servidores que administrar, es fácil mantener un fichero de inventario manualmente y no supone demasiado trabajo. Pero a medida que el número de servidores aumenta, este trabajo se complica cada vez más, aparte de que mantener una lista numerosa de servidores manualmente puede convertirse en una tarea propensa a errores.

Cuando se gestiona un número elevado de servidores, es poco probable que el control del inventario dependa de un archivo de texto estático; es probable que se opte en su lugar por una hoja de cálculo o una base de datos. ¿No sería fantástico poder usar estas mismas fuentes en Ansible como la fuente de datos de inventario?

Ansible dispone del concepto de **inventario dinámico,** que consiste en un fichero JSON que proporciona todos los datos necesarios sobre sus *hosts*. El formato de archivo JSON no es tan legible a simple vista como el formato INI, ya que está diseñado principalmente para ser leído por máquinas. Ansible hace su propio procesamiento de estos formatos y realiza algunas comprobaciones al acceder al fichero de inventario que se proporciona.

Cuando se ejecuta Ansible, este comprobará si el fichero pasado como parámetro del inventario es un archivo ejecutable. En caso de serlo, lo ejecutará y Ansible utilizará su analizador JSON para leer los datos resultantes. Si no es ejecutable, Ansible lo leerá suponiendo que está en formato de fichero INI y fallará en el análisis si es un archivo JSON estático.

El uso de un archivo ejecutable permite leer datos de inventario sobre sus *hosts* desde cualquier fuente de datos accesible local o remotamente, tales como un API remoto, una base de datos local, un grupo de ficheros que se analizan y comparan…

a fin de cuentas, lo que sea que se necesite para obtener la lista de servidores sobre la que ejecutar.

Ansible espera que el ejecutable devuelva un formato JSON específico que se utilizará como fuente del inventario de máquinas. Veamos un ejemplo a continuación:

```
{"my_script": ["dev2","dev"],"_meta": {"<em>host</em>vars":
{"dev2":
```

```
{"ansible_<em>host</em>":"dev2.example.com","ansible_user":"ansible"},"dev":
```

```
{"ansible_<em>host</em>":"dev.example.com","ansible_port":"50022",
"ansible_user":"automation"}}}}
```

Como podemos observar, se trata de un conjunto de pares clave-valor, aunque los conjuntos pueden a su vez ser subconjuntos. La primera entrada tiene como clave el nombre del *script*, y como valor, la lista de nombres de *host* que se usará como inventario. A continuación, hay una sección de metadatos en la que para cada nombre de *host* hay algunas entradas que contienen el nombre de *host* a utilizar, el usuario SSH con el que acceder y el puerto SSH.

Si por ejemplo tuviéramos una base de datos que contiene todas las máquinas y quisiéramos utilizar esa base de datos como fuente del inventario, el ejecutable podría implementar algo parecido al siguiente pseudocódigo:

```
machines = fetch_rows("SELECT <em>host</em>name, user, key, port
FROM
```

```
active_machines")
```

```
<em>host</em>names = machines.map (m) => return m.
<em>host</em>name metadata = {
```

```
'<em>host</em>vars'=> {}
```

```
}

foreach (machines as m) {

 metadata.<em>host</em>vars[m.<em>host</em>name] = {

 ansible_user => m.user,

 ansible_port => m.port,

 ansible_ssh_private_key_file => m.key

 }

}

output_json({ 'my_script'=> <em>host</em>names, '_meta'=>
metadata})
```

La función se encarga de leer la información de la base de datos, mientras que el resto del script formatea la información tal como Ansible la espera. Aunque este ejemplo solo está leyendo los datos desde un único lugar, podría tener que leer desde múltiples fuentes si fuera necesario para combinar todos los datos en el ejecutable con el que estás trabajando y devolverlos en el formato que espera Ansible.

Existen un gran número de soluciones de inventario dinámico ya construidas disponibles para Ansible.

Para *hosts* ubicados en Amazon AWS puedes optar por utilizar un inventario dinámico, debido a las capacidades de autoescalado de la nube. Aunque un inventario dinámico puede ser de naturaleza estática (por ejemplo, si lo mantienes en una base de datos de forma manual), es muy útil cuando estás utilizando entornos realmente dinámicos, como por ejemplo Amazon EC2, para crear máquinas bajo

demanda. En este caso, debes asegurarte al crear el *host* de que tiene las etiquetas adecuadas para que, cuando ejecutes Ansible la próxima vez, se aprovisione junto con todas las demás máquinas de tu entorno.

Si no utilizas OpenStack en lugar de AWS, también existe ya implementado un script de inventario dinámico para OpenStack. Antes de implementar tu propio script, busca primero uno que te pueda servir y aprovéchalo. Si no encuentras ninguno que sea adecuado a tu caso, entonces puedes escribir el tuyo propio, aunque siempre puedes aprovechar y reutilizar partes de algunos donde se resuelva alguna situación semejante a la tuya.

Inventarios múltiples

Por último, ¿qué ocurre cuando se tiene una combinación de máquinas físicas y servidores en la nube en tu entorno? No es posible obtener la fuente desde una API basada en la nube exclusivamente, ya que no sabe acerca de sus máquinas físicas, y, posiblemente, no se pueda realizar un seguimiento de todos los servidores en la nube manualmente. Afortunadamente, para estos casos Ansible también tiene una solución.

Si la ruta del fichero inventario que pasas como parámetro a Ansible es un directorio, Ansible leerá todos y cada uno de los archivos en ese directorio como un inventario y los fusionará. Esto te permite tener un fichero de inventario de gestión manual, así como otra parte en , por ejemplo, que genera dinámicamente el inventario de Amazon EC2. Si el inventario es grande, incluso podríamos tener múltiples archivos INI desglosados por centro de datos o por rol (o cualquier división arbitraria que se pueda tener), así como varios ejecutables que se ejecutan para hablar con varios proveedores de nube. Cuando los datos de todos estos ficheros sean recogidos por Ansible, se tratarán como si fueran un único inventario extenso.

6.8. Referencias bibliográficas

Heap, M. (2016). *Ansible: from Beginner to Pro*. Apress.

Hochstein, L. y Moser R. (2014). *Ansible: Up and Running: Automating Configuration Management and Deployment the Easy Way*. O'Reilly Media.

Documentación de referencia de Ansible

Red Hat, Inc. (2020). *Ansible Documentation.* https://docs.ansible.com/ansible/latest/index.html

En el sitio oficial de documentación Ansible es donde podrás encontrar la documentación de referencia más completa y actualizada de la herramienta, así como la versión de documentación correspondiente con la propia versión de la herramienta que estés utilizando. Es el primer recurso al que acceder en busca que cualquier concepto Ansible, para asegurarnos de su veracidad y actualidad.

Documentación de referencia del inventario de Ansible

Red Hat, Inc. (2020). *How to build your inventory*. https://docs.ansible.com/ansible/latest/user_guide/intro_inventory.html

Como decíamos en el primer recurso, en el sitio oficial de documentación Ansible es donde podrás encontrar la documentación de referencia más completa y actualizada de la herramienta, y este enlace te proporciona acceso directo a la sección donde se explica el inventario en Ansible.

Ansible Tips and Tricks – Inventory

DePorter, M. (2016). *Host Inventories*. Ansible Tips and Tricks. https://ansible-tips-and-tricks.readthedocs.io/en/latest/ansible/inventory/

Esta wiki contiene información complementaria a la documentación oficial, incluyendo ejemplos adicionales y buenas prácticas. Esta sección de la wiki es la relativa al inventario.

Test

1. ¿Es recomendable utilizar el fichero de inventario por defecto de Ansible?

   A. Sí, porque así no te tienes que preocupar de definir otro fichero.

   B. No, porque es recomendable mantener un fichero de inventario por cada proyecto.

   C. Sí, porque Ansible detectará así automáticamente los *hosts* a gestionar en la red.

   D. No, porque los *hackers* pueden así conocer fácilmente tu inventario.

2. ¿Cuál es el formato habitual de un fichero de inventario?

   A. XML.

   B. JSON.

   C. INI para los estáticos, XML para los dinámicos.

   D. INI para los estáticos, JSON para los dinámicos.

3. ¿Se pueden usar rangos en los ficheros de inventario INI?

   A. Sí, para especificar una serie de servidores con un patrón de nombre común.

   B. No, solo nombres individuales, alias y grupos.

   C. Sí, para especificar un valor para un conjunto de propiedades con un patrón de nombre común.

   D. Ninguna de las anteriores.

4. ¿Para qué se utiliza el parámetro de inventario ?

   A. Para especificar el usuario SSH con el que Ansible conecta al *host*.

   B. Para especificar el usuario con el que ejecuta Ansible en local.

   C. Para indicar el usuario con permisos de administrador.

   D. Para crear un usuario en el *host*.

5. ¿Qué significa la siguiente línea en un fichero de inventario: ?

A. El servidor que tiene como alias

B. El alias que se asocia al *host*

C. El servidor y el servidor virtual

D. El servidor al que se define la variable

6. ¿Para qué sirve un grupo de inventario?

A. Para poder establecer variables a un conjunto de servidores a la vez.

B. Para poder agrupar un conjunto homogéneo de servidores y manejarlos como una única entidad.

C. Para agrupar grupos de servidores en una agrupación superior.

D. Todas las anteriores.

7. ¿Cómo se define un grupo de grupos de inventario?

A. Simplemente definiendo un grupo, y añadiendo dentro nombres de otros grupos, en lugar de *hosts*.

B. Utilizando el sufijo en tu nombre de grupo.

C. Utilizando el sufijo en tu nombre de grupo.

D. No se pueden anidar grupos de inventario.

8. ¿Para cuándo es más adecuado utilizar un fichero de inventario dinámico?

A. Para definir más de 10 *hosts* en el inventario.

B. Cuando mantenemos nuestra lista de servidores en un fichero INI.

C. Cuando existe un número elevado de servidores y necesitamos utilizar una fuente de datos externa que nos proporcione el inventario.

D. Ninguna de las anteriores.

9. ¿Cómo podemos utilizar un fichero de inventario dinámico?

A. Indicando un fichero XML en el parámetro al ejecutar Ansible.

B. Indicando un fichero ejecutable en el parámetro al ejecutar Ansible.

C. Indicando un fichero JSON en el parámetro al ejecutar Ansible.

D. Indicando un fichero sin extensión INI en el parámetro al ejecutar Ansible.

10. ¿Cómo especificar un inventario mixto, parte estático y parte dinámico?

A. Especificando un directorio como fichero de inventario, donde reside un fichero INI para lo estático, y uno o varios ejecutables para la parte dinámica.

B. Incluyendo 2 parámetros al ejecutar Ansible.

C. No se puede, el inventario o es estático o es dinámico.

D. Haciendo un *script* de inventario dinámico que lea el estático y lo transforme a dinámico.