26/10/2024

Práctica

Configuración Automática de Infraestructuras con Ansible

Germán Moltó – gmolto@dsic.upv.es - Curso Online de Cloud Computing con Amazon Web Services

Práctica

Configuración Automática de Infraestructuras con Ansible

Contenido

Introducción 3

Resultados de Aprendizaje 4

Conexión y Verificación del Entorno de Trabajo 4

Configuración de Infraestructuras con Ansible 4

Introducción a Ansible 5

Configuración del Entorno para Realizar la Práctica 6

Aprovisionamiento y Configuración de Instancias 6

Sobre el Inventario de Ansible 8

Usando Ansible para la Ejecución Remota de Comandos 9

Fichero de Configuración de Ansible 9

Usando los Módulos de Ansible 12

Ejemplo Práctico: Instalación de Paquetes 13

Ejemplo Práctico: Despliegue Automático de un Servidor Web 15

Creación de Recetas en Ansible: Playbooks 17

Ejemplo de Playbook sencillo: Instalación Remota de Fortune 17

Ejemplo de Playbook: Despliegue Automático de Servidor Web 19

Configuración Automática de Máquina Basada en Roles 20

Aprovisionamiento de Servidor de Base de Datos 23

Playbooks Avanzados: Importación, Templates e Iteración 26

Terminación de Recursos 30

Aprovisionamiento y Configuración de Infraestructuras 31

Aprovisionamiento de Instancias 33

Configuración de Instancias 33

Ejecución del PlayBook....34
Conclusiones....38
Referencias....38
Anexo....41
Anexo A. Configuración de la Máquina Principal y de las Máquinas a Configurar....41

Introducción

Con la aparición de proveedores de Cloud Computing que ofrecen servicio de IaaS (*Infrastructure as a Service*), se facilita en gran medida el aprovisionamiento de infraestructuras virtuales de cómputo, en la forma de máquinas virtuales. Dicha infraestructura debe ser convenientemente configurada para soportar las aplicaciones que se ejecuten sobre ella. Esto implica asignar roles a instancias para desplegar la arquitectura de una aplicación. Por ejemplo, en una aplicación web de venta electrónica puede haber instancias de cómputo dedicadas a servir las páginas web (a través de un servidor web), instancias dedicadas a almacenar de forma redundante la base de datos de productos, instancias que implementan la lógica de negocio (facturación, estadísticas, etc.). Todas ellas forman la arquitectura de una aplicación en la nube.

La configuración de las instancias puede realizarse de forma manual o automática. Una configuración manual implica conectarse a una instancia para crear las cuentas de usuario, instalar y configurar manualmente los paquetes de software, configurar los servicios, etc. Si dicha instancia termina, entonces es necesario volver a realizar la misma configuración en una nueva instancia. Fíjate que, una vez configurada una instancia, es posible guardar una instantánea de la configuración de la misma a modo de imagen maestra o *Golden Image* [1]. El problema de esta aproximación es que se debe ir actualizando tanto la imagen maestra como las instancias con las últimas actualizaciones de seguridad o versiones de las aplicaciones, lo que dificulta la capacidad de mantener de la solución. Esto es especialmente evidente ante un cambio de versión de sistema operativo, que implica partir de una nueva imagen de máquina virtual y realizar nuevamente la configuración e instalación de las aplicaciones.

Por el contrario, la configuración automática de infraestructuras se basa principalmente en la definición de *recetas* que definen el estado que se desea que alcance una determinada instancia (o un grupo de ellas). Estas recetas se traducen a un conjunto de acciones que deben ejecutarse en la instancia para que ésta alcance dicho estado. Un estado puede ser, por ejemplo, la existencia de unas determinadas cuentas de usuario, una configuración específica para el servidor SSH, la instalación y actualización de un determinado paquete, la creación de un determinado fichero, etc. Esto permite por un lado realizar el aprovisionamiento de recursos y por otro la configuración automática de los mismos, por lo que se desacopla el HW y el S.O. de las aplicaciones y la configuración deseada. Esta aproximación facilita el mantenimiento y reproducibilidad de las infraestructuras (que pueden ser virtualizadas o físicas).

Estas herramientas se conocen generalmente en la literatura como herramientas de *DevOps* [2]. Actualmente existen numerosas herramientas de configuración automática de infraestructuras, pero estas son las más relevantes, considerando su adopción por la comunidad (sin ningún orden en particular): Ansible [3], Puppet [4] y Chef [5]. Otras herramientas de esta categoría son: Rex [6], SaltStack [7], Capistrano [8], CFEngine [9] o Juju [10].

En esta práctica trabajarás con Ansible que utiliza una aproximación de tipo *push* donde desde una máquina se ejecutan de forma remota comandos vía SSH en las instancias de la infraestructura para acabar configurando el estado definido por el usuario en una receta. En el caso de Ansible, no es necesario instalar ningún agente de configuración en las instancias remotas. Tan solo es necesario realizar conexiones SSH en la máquina remota, preferiblemente sin tener que especificar la contraseña.

Esta práctica no pretende ser un manual de referencia exhaustivo sobre la herramienta Ansible. Al contrario, pretende introducir los conceptos básicos para que el alumno tenga una idea del

funcionamiento de la herramienta, como un ejemplo de herramienta DevOps y decida si pueden ser de utilidad para sus proyectos presentes o futuros de despliegue de aplicaciones en el Cloud.

Resultados de Aprendizaje

Se espera que, una vez finalizada la práctica, el alumno sea capaz de:

- Entender las ventajas de la configuración automática de infraestructuras virtuales.
- Comprender el funcionamiento de la herramienta Ansible para la ejecución remota de comandos y para la configuración desatendida de infraestructuras.
- Diseñar recetas de Ansible (*playbooks*) sencillas para la configuración de infraestructuras de mediana envergadura.
- Ejecutar recetas para la configuración de infraestructuras aprovisionadas dinámicamente.

1. Tu profesor te habrá asignado un número. Los comandos que aparecen en el boletín hacen referencia al usuario alucloud00. Por favor, **sustituye en los comandos que veas 00 por tu número** (por ejemplo, si tu número es 06, entonces el usuario que has de utilizar es alucloud06). Recuerda esta regla para facilitar tanto tu trabajo como el del resto de compañeros. Para facilitar tu trabajo, en los comandos se utiliza una variable de entorno llamada ID que se debe resolver a tu número.
2. En esta práctica haremos ejecuciones reales sobre un proveedor de Cloud público, que generan un coste económico. Asegúrate de **liberar apropiadamente los recursos** para **no incurrir en costes adicionales**. Si tienes alguna duda al respecto consulta con tu profesor.
3. Recuerda que hay un sistema de recompensas, definido en la Guía de Prácticas, por el que podrás aumentar la duración del curso reportando las discrepancias que encuentres en este boletín frente a posibles cambios introducidos por AWS. Cuento con tu colaboración.

Conexión y Verificación del Entorno de Trabajo

Deberás conectarte vía SSH al entorno de realización de prácticas que te haya indicado tu instructor con la cuenta de usuario especificada. En dicha máquina están instaladas las herramientas necesarias para llevar a cabo la práctica.

Configuración de Infraestructuras con Ansible

Ansible [3] es una herramienta surgida en 2012 que permite la orquestación y administración de sistemas mediante la creación de recetas de configuración que permiten configurar recursos computacionales de forma determinista. Utiliza una aproximación de tipo *push* donde la máquina que ejecuta Ansible realiza conexiones SSH a las máquinas remotas para ejecutar los comandos necesarios para conseguir la configuración especificada por el usuario.

En primer lugar, usaremos Ansible para la ejecución remota de comandos en máquinas de una infraestructura virtualizada basada en máquinas GNU/Linux. Posteriormente, veremos el uso de recetas para configurar de forma automática y desatendida las infraestructuras de cómputo.

La Figura 1 muestra el esquema empleado para la realización de la práctica. Los alumnos se conectan al entorno remoto de realización de prácticas donde existen múltiples cuentas de usuario del estilo alucloudXX. Dicha máquina dispone de Ansible instalado y existen una serie de scripts (disponibles en /opt/cursoaws/ansible) que te permitirán desplegarte un par de máquinas (llamadas *Instancias*

Configurables, en la figura) para luego utilizarlas durante la práctica para las distintas tareas de configuración de las mismas. Dichos scripts utilizan por debajo unas recetas (*playbooks*) de Ansible para realizar el aprovisionamiento de las instancias sobre EC2, que serán etiquetadas (*tags* [22]) con el Tag ansible : alucloudXX para identificarlas y saber que son de tu usuario para realizar la práctica de Ansible. A continuación, los scripts las configuran para permitir la conexión SSH sin contraseña desde el usuario alucloudXX en el entorno de prácticas al usuario ubuntu en las instancias configurables. Una vez finalizada la práctica serás capaz de entender cómo funcionan las recetas de Ansible creadas para realizar estas tareas. Por ahora, puedes utilizarlas como cajas negras.

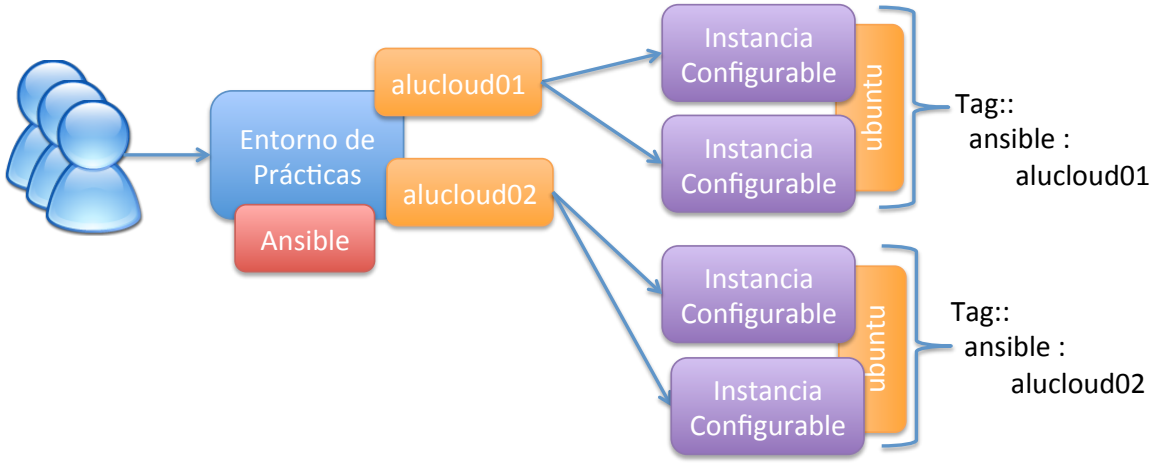


Figura 1. Arquitectura para la práctica de Ansible.

Introducción a Ansible

Ansible permite la ejecución remota de comandos y la configuración desatendida de máquinas remotas. Utiliza un fichero donde se especifican qué máquinas se desea configurar. Este se denomina fichero de *inventario*, usando la terminología de Ansible. Dicho fichero está típicamente en /etc/ansible/hosts aunque es posible modificar la variable de entorno ANSIBLE_HOSTS para utilizar cualquier otro fichero de máquinas.

A continuación, se muestra un ejemplo de fichero de inventario de Ansible.

```
[webservers]
54.205.187.9 ansible_ssh_user=ubuntu
54.205.154.9 ansible_ssh_user=ubuntu

[dbservers]
54.226.194.39 ansible_ssh_user=ubuntu
54.226.164.97 ansible_ssh_user=ubuntu
```

Dicho fichero puede especificar un listado de direcciones IP o nombres de máquina (el caso más sencillo) o puede clasificar las máquinas por categorías (como se muestra en el ejemplo anterior), donde cada categoría implica que dichas máquinas desempeñan un cierto rol en la arquitectura de una aplicación. También es posible especificar *wildcards* para indicar un conjunto de máquinas que cumplen un determinado patrón (por ejemplo www[01:13].acme.com), así como indicar el usuario con el se quieren realizar las conexiones a dicha máquina (mediante *ansible_ssh_user*). Puedes encontrar más información sobre el fichero de inventario de máquinas en [11].

Cada alumno se desplegará un grupo de (2) máquinas para ser configuradas con Ansible. No es necesario permisos de superusuario (*root*) para ejecutar Ansible pero sí es necesario (al menos conveniente) conectarse desde la cuenta de usuario que ejecutes Ansible a una cuenta de la máquina remota sin necesidad de especificar la contraseña. Esto evita tener que introducir la contraseña para cada comando que se quiera ejecutar en la máquina remota. En el Anexo A, al final de esta práctica, tienes explicado el procedimiento necesario para conseguir conectarte por SSH a una máquina remota sin necesidad de especificar la contraseña, aunque no es necesario que realices este procedimiento ahora, puesto que los scripts que tienes a tu disposición lo harán automáticamente.

Configuración del Entorno para Realizar la Práctica

Aprovisionamiento y Configuración de Instancias

Aprovisionamos los recursos necesarios para realizar la práctica. Se desplegarán dos instancias en EC2 y se configurarán para que puedas conectarte mediante SSH a la cuenta de usuario ubuntu de dichas instancias. Para ello, deberás tener (ten en cuenta que, si ya has realizado la práctica de EC2 desde el entorno de prácticas del curso, cumplirás los siguientes requisitos):

- Una clave RSA previamente creada en tu cuenta de usuario (ficheros $HOME/.ssh/id_rsa, para la clave privada y fichero $HOME/.ssh/id_rsa.pub, para la clave pública). Si estas realizando la práctica desde el entorno de prácticas ya debes disponer de dicha clave puesto que ha sido previamente creada por el instructor.
- Un par de claves en la región us-east-1 con la siguiente nomenclatura: alucloud$ID-keypair (donde $ID corresponde con tu identificador de usuario).
- El fichero de clave privada correspondiente a tu par de claves en Amazon EC2, en la siguiente ruta y con la siguiente nomenclatura: $HOME/alucloud$ID-priv.pem. Ten en cuenta que puedes cambiar el nombre del fichero .pem si fuera necesario. El fichero debe tener únicamente permisos de lectura para el propietario (puedes conseguir dichos permisos ejecutando el comando `chmod 0600 $HOME/alucloud$ID-priv.pem`).

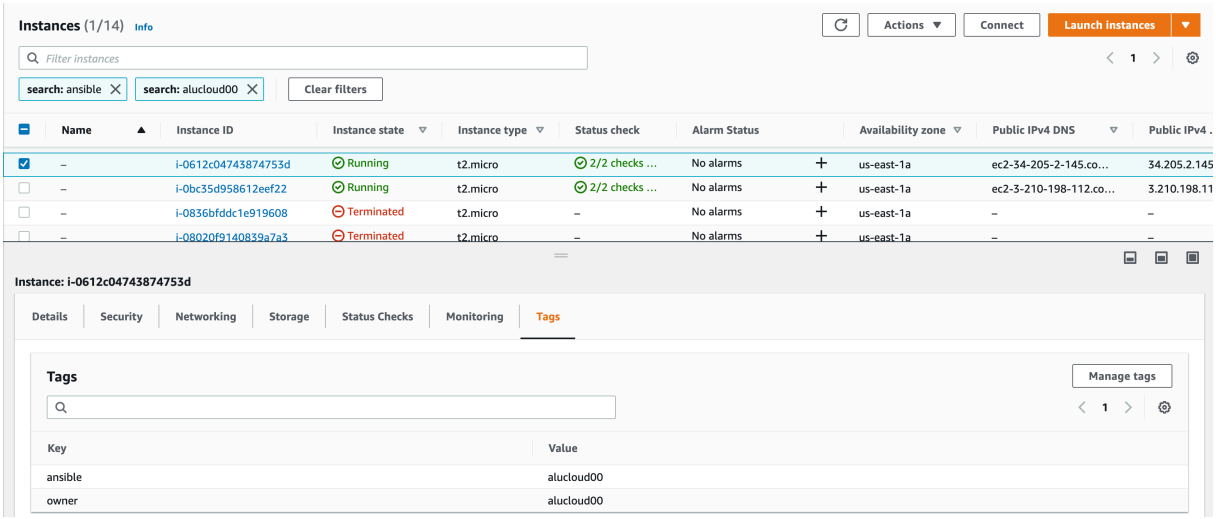
Ten en cuenta que el despliegue y configuración de las instancias puede requerir 1-2 minutos por lo que ten paciencia y no canceles la ejecución de los scripts.

```
:~$ /opt/cursoaws/ansible/self-deploy-ansible-slaves.sh
PLAY [Create VMs] *************************************************************

GATHERING FACTS ***************************************************************
ok: [localhost]

TASK: [Provisioning EC2 instances (This might take a while ... DO NOT INTERRUPT.)]
***
changed: [localhost]

TASK: [Waiting for the new instances to be ready (This might take a while ... DO
NOT INTERRUPT.)] ***
ok: [localhost] => (item={u'kernel'
…
TASK: [Showing your instances information] ************************************
ok: [localhost] => (item={u'kernel': u'aki
…
"msg": "Your instance for Ansible is i-4715536c --> ec2-54-89-87-166.compute-
1.amazonaws.com"
}
…
"msg": "Your instance for Ansible is i-4415536f --> ec2-54-89-97-254.compute-
1.amazonaws.com"
```

```
PLAY RECAP *********************************************************************
localhost                  : ok=4    changed=1    unreachable=0    failed=0


PLAY [Configure VMs] ***********************************************************

GATHERING FACTS ****************************************************************
ok: [ec2-54-89-87-166.compute-1.amazonaws.com]
ok: [ec2-54-89-97-254.compute-1.amazonaws.com]

TASK: [Authorizing user's SSH key for remote user] *****************************
changed: [ec2-54-89-97-254.compute-1.amazonaws.com]
changed: [ec2-54-89-87-166.compute-1.amazonaws.com]



PLAY RECAP *********************************************************************
ec2-54-89-87-166.compute-1.amazonaws.com : ok=2    changed=1    unreachable=0
failed=0
ec2-54-89-97-254.compute-1.amazonaws.com : ok=2    changed=1    unreachable=0
failed=0
```

Si has obtenido algún error durante el despliegue, asegúrate de que cumples los requisitos indicados en el apartado anterior. Ten en cuenta que es posible que obtengas *warnings* relativos a la versión de *urllib* o *chardet*. Puedes ignorarlo. En particular, si tu par de claves no respeta la nomenclatura indicada, puedes avisar al instructor para que proceda a eliminarlo y que puedas crear tú mismo uno nuevo con dicha nomenclatura.

Habrás podido comprobar que el script provoca la creación de dos instancias, cuyos identificadores y nombres DNS se muestran en un mensaje (destacado en color **morado** en el listado anterior). Dichas instancias se etiquetan con la etiqueta ansible : alucloud00.



Puedes comprobar que tienes acceso desde tu cuenta de usuario mediante SSH a la cuenta ubuntu en dichas instancias, sin necesidad de especificar contraseña. Elige cualquiera de las dos máquinas y trata de conectarte:

```
:~$ ssh -i alucloud$ID-priv.pem ubuntu@ec2-54-89-97-254.compute-
1.amazonaws.com
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-1024-aws x86_64)
```

```
…
ubuntu@ip-10-178-170-98:~$
```

Deberás haber podido conectarte a la máquina remota con el usuario ubuntu, indicando tu clave privada, sin que te haya pedido la contraseña. Si te solicita la contraseña revisa los pasos anteriores antes de contactar con el instructor.

Sobre el Inventario de Ansible

Recordemos que el fichero de inventario [24] de Ansible contiene el listado de máquinas, clasificado por categorías, que serán configuradas por Ansible. Dicho fichero, puede ser estático, como se ha explicado anteriormente, pero también se puede utilizar un fichero de inventario dinámico [25]. Por ejemplo, Ansible soporta múltiples plugins de inventario [26] y, entre ellos, está disponible un plugin para EC2 que permite obtener un inventario dinámico con las instancias de EC2 desplegadas con una cuenta de usuario, agrupadas por diferentes criterios. Uno de los criterios es la existencia de una determinada etiqueta (tag) [22] en dichas instancias. Esto permitirá que puedas ejecutar comandos a través de Ansible únicamente a aquellas instancias que tengan una determinada etiqueta, que coincidirá con la etiqueta asignada a las instancias desplegadas en la fase anterior (ansible : alucloudXX).

Prueba a ejecutar el siguiente comando y analiza el resultado que obtienes (en función de las instancias que hayan desplegadas tu resultado puede variar).

```
:~$ ansible-inventory -i /opt/cursoaws/ansible/inventory_aws_ec2.yml --graph
@all:
  |--@aws_ec2:
  |  |--ec2-18-209-237-174.compute-1.amazonaws.com
  |  |--ec2-3-222-215-232.compute-1.amazonaws.com
  |  |--ec2-34-205-126-186.compute-1.amazonaws.com
  |  |--ec2-34-236-138-83.compute-1.amazonaws.com
  |  |--ec2-52-202-41-247.compute-1.amazonaws.com
  |  |--ec2-54-167-165-125.compute-1.amazonaws.com
  |  |--ec2-54-237-163-226.compute-1.amazonaws.com
  |  |--ip-172-31-11-105.ec2.internal
  |  |--ip-172-31-3-77.ec2.internal
  |  |--ip-172-31-55-27.ec2.internal
  |  |--ip-172-31-70-116.ec2.internal
  |  |--ip-172-31-71-13.ec2.internal
  |  |--ip-172-31-8-89.ec2.internal
  |  |--ip-172-31-89-106.ec2.internal
  |--@tag_ansible_alucloud00:
  |  |--ec2-18-209-237-174.compute-1.amazonaws.com
  |  |--ec2-3-222-215-232.compute-1.amazonaws.com
  |--@ungrouped:
```

Verás que se obtiene mucha información sobre las instancias de EC2, pero lo más importante para esta práctica es la sección *tag_ansible_alucloudXX* (en el ejemplo anterior se muestra para el caso del alumno con identificador 00). Puedes obtener los datos en formato JSON [27] usando la opción *list* en lugar de *graph*. Esto sería el equivalente a tener dichas máquinas en un fichero de inventario estático como el siguiente:

```
[tag_ansible_alucloud00]
ec2-54-89-87-166.compute-1.amazonaws.com
```

```
ec2-54-89-97-254.compute-1.amazonaws.com
```

La ventaja de usar un fichero de inventario dinámico frente a uno estático es que puedes aprovisionar nuevos recursos y siempre se verán reflejados en dicho inventario, sin necesidad de editar el fichero de inventario para que siempre esté sincronizado con las instancias desplegadas.

¿Listo para comenzar? Arrancamos con la práctica.

Usando Ansible para la Ejecución Remota de Comandos

En primer lugar, verificaremos que el entorno está configurado apropiadamente. Por comodidad, asumiremos que la variable de entorno ID existe y contiene tu número de identificación de alumno. Esta variable ya está declarada en el entorno de trabajo de prácticas que se te ofrece. Ejecutamos el siguiente comando:

```
:~$ ansible -i /opt/cursoaws/ansible/inventory_aws_ec2.yml --key-file alucloud$ID-
priv.pem  tag_ansible_alucloud$ID -u ubuntu -m ping
ec2-3-210-198-112.compute-1.amazonaws.com | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false,
    "ping": "pong"
}
ec2-34-205-2-145.compute-1.amazonaws.com | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false,
    "ping": "pong"
}
```

Dicho comando está ejecutando el módulo *ping* sobre las instancias de la categoría *tag_ansible_alucloudoo (*en el caso de que ID=00) definidas en el fichero de inventario resultante de la ejecución del script de inventario dinámico (inventory_aws_ec2.yml). La opción `-u` se utiliza para conectarse a la máquina remota vía SSH con el usuario ubuntu ya que, por defecto, se asume el mismo usuario con el que se invoca el comando (alucloud$ID, en el ejemplo). También se especifica el fichero de clave privada necesario para realizar la conexión a las instancias. Es posible indicar *all*, en lugar de *tag_ansible_alucloudoo,* para ejecutar dicho módulo sobre todas las máquinas (no únicamente las de la categoría *tag_ansible_alucloudoo*), pero no es necesario que lo pruebes. Es posible encontrar un listado de los módulos soportados por Ansible en [12], aunque éstos se tratarán más adelante en la práctica.

Por defecto, el fichero de inventario que se elige es el disponible en /etc/ansible/hosts, aunque se puede modificar la ruta al fichero de inventario definiendo la variable de entorno $ANSIBLE_HOSTS.

Fichero de Configuración de Ansible

Hay un fichero de configuración general que habitualmente está en `/etc/ansible/ansible.cfg` (no está en el entorno de prácticas) aunque también se puede tener un fichero de configuración local en `$HOME/.ansible.cfg`. Ten en cuenta que no tienes permisos de escritura en el fichero de configuración general por lo que no trates de introducir ningún cambio en él.

En el entorno de prácticas ya está creado el siguiente fichero de configuración, que será usado automáticamente cada vez que utilices Ansible:

```
:~$ cat /etc/ansible/ansible.cfg
[defaults]
host_key_checking = False
transport = ssh
jinja2_extensions = jinja2.ext.do
remote_user = ubuntu
private_key_file=~/alucloud$ID-priv.pem
interpreter_python = auto
inventory = /opt/cursoaws/ansible/inventory_aws_ec2.yml
force_valid_group_names = ignore

[ssh_connection]
ssh_args = -o ControlMaster=auto -o ControlPersist=900s
pipelining = True
```

Tienes el detalle de para qué sirve cada opción en la documentación [28], pero básicamente le estamos indicando que use inventario dinámico, que se conecte a las máquinas con el usuario *ubuntu* y usando el fichero de clave privada que está disponible en la cuenta de cada alumno.

Por tanto, dado que dicho fichero de configuración existe, realmente puedes volver a ejecutar el comando anterior, pero esta vez sin necesidad de especificar ni el fichero de inventario, ni el usuario remoto, pues están definidos como valores por defecto en el fichero.

```
:~$ ansible tag_ansible_alucloud$ID -m ping
ec2-34-233-71-129.compute-1.amazonaws.com | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
ec2-3-237-199-118.compute-1.amazonaws.com | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
```

Si, por el contrario, obtienes el siguiente mensaje de error:

```
:~$ ansible tag_ansible_alucloud$ID -m ping
ec2-54-147-251-139.compute-1.amazonaws.com | FAILED => SSH Error: data
could not be sent to the remote host. Make sure this host can be reached
over ssh
ec2-54-161-163-209.compute-1.amazonaws.com | FAILED => SSH Error: data
could not be sent to the remote host. Make sure this host can be reached
over ssh
```

Asegúrate de que el fichero de clave privada es el apropiado.

Con Ansible también es posible la ejecución remota de comandos directamente en la máquina destino, al igual que se puede hacer vía SSH directamente, mediante la opción de línea de comandos `-a`:

```
:~$ ansible tag_ansible_alucloud$ID -a "/bin/ls -l /tmp" -f 2
ec2-3-237-199-118.compute-1.amazonaws.com | CHANGED | rc=0 >>
total 20
drwx------ 2 ubuntu ubuntu 4096 Oct  3 05:38
ansible_command_payload_3f4uqgaa
drwx------ 3 root   root   4096 Oct  3 05:31 snap.lxd
drwx------ 3 root   root   4096 Oct  3 05:30 systemd-private-
b044b8e8245a4078aa9d40a8c8343486-systemd-logind.service-ZxTmth
drwx------ 3 root   root   4096 Oct  3 05:30 systemd-private-
b044b8e8245a4078aa9d40a8c8343486-systemd-resolved.service-HZcvmj
drwx------ 3 root   root   4096 Oct  3 05:30 systemd-private-
b044b8e8245a4078aa9d40a8c8343486-systemd-timesyncd.service-sJ73pg
ec2-34-233-71-129.compute-1.amazonaws.com | CHANGED | rc=0 >>
total 20
drwx------ 2 ubuntu ubuntu 4096 Oct  3 05:38
ansible_command_payload_mzh7c1_g
drwx------ 3 root   root   4096 Oct  3 05:31 snap.lxd
drwx------ 3 root   root   4096 Oct  3 05:30 systemd-private-
1dedba08263d4587bea93ff589b889d8-systemd-logind.service-4wdzPi
drwx------ 3 root   root   4096 Oct  3 05:30 systemd-private-
1dedba08263d4587bea93ff589b889d8-systemd-resolved.service-IAg4pi
drwx------ 3 root   root   4096 Oct  3 05:30 systemd-private-
1dedba08263d4587bea93ff589b889d8-systemd-timesyncd.service-xHeDUf
```

Esto ejecuta el comando /bin/ls en todas las máquinas de la categoría tag_ansible_alucloud$ID, conectándote a las máquinas remotas como usuario ubuntu y ejecutando en paralelo (con 2 procesos) para acelerar la ejecución de los comandos en las máquinas. Esto es útil cuando tienes decenas de máquinas para configurar, aunque en realidad el valor por defecto es 5.

Si quieres realizar alguna operación que requiera privilegios de superusuario en la máquina destino es posible hacerlo con la opción --become).

```
:~$ ansible tag_ansible_alucloud$ID -a "cat /etc/shadow"  --become
ec2-54-89-97-254.compute-1.amazonaws.com | success | rc=0 >>
root:!*:15806:0:99999:7:::
daemon:*:15806:0:99999:7:::
```

Para ello, la cuenta ubuntu en la(s) máquina(s) destino debe tener privilegios de *sudo*. Fíjate como estamos listando el fichero `/etc/shadow` de la máquina remota, que para un usuario sin privilegios está prohibido, ya que este fichero almacena los usuarios y las contraseñas [30]. De hecho, si pruebas a ejecutar el comando anterior sin la opción `--become` obtendrás el mensaje de error:

```
:~$ ansible tag_ansible_alucloud$ID -a "cat /etc/shadow"
ec2-34-233-71-129.compute-1.amazonaws.com | FAILED | rc=1 >>
cat: /etc/shadow: Permission deniednon-zero return code
ec2-3-237-199-118.compute-1.amazonaws.com | FAILED | rc=1 >>
cat: /etc/shadow: Permission deniednon-zero return code
```

Es posible copiar un fichero de la máquina principal a las máquinas a configurar. Aunque para ello, hay que utilizar un *módulo*, según la terminología de Ansible, tal y como se ha hecho anteriormente con el módulo *ping*. Veamos que son los módulos con más detalle.

Usando los Módulos de Ansible

Ansible consta de unas librerías de módulos que pueden ser directamente ejecutados en las máquinas remotas (o bien a través de *Playbooks* [16], que se tratarán más adelante). Un módulo no es más que un conjunto de instrucciones u operaciones encargadas de controlar recursos del sistema, como servicios, paquetes, ficheros e incluso para la ejecución de comandos del sistema. Puedes encontrar un listado de los módulos soportados por Ansible en [12]. El usuario también puede definirse sus propios módulos.

Aquí tienes un listado de algunas capacidades que pueden realizarse con Ansible de acuerdo a la funcionalidad que ofrecen sus módulos. Esta lista no es exhaustiva, pero te proporciona una idea general de la versatilidad de Ansible:

Funcionalidad	Módulo(s)
Gestión de ficheros	copy, file, lineinfile, template
Instalación de paquetes	apt, yum
Ejecución de comandos	command, shell, script
Creación de bases de datos y usuarios	mysql_user, mysql_db, postgresql_user, postgresql_db
Creación de usuarios	user
Gestión de servicios	service
Interacción con AWS (y otros)	ec2, s3, ec2_elb, ec2_vol, nova_compute
...	...

Los módulos pueden recibir argumentos de la forma clave=valor, delimitados por espacios, aunque hay módulos que no reciben argumentos y módulos especiales como los usados para la ejecución de comandos que simplemente reciben una cadena con el comando que se desea ejecutar. Los módulos son idempotentes, por lo que tratan de evitar cambios en una máquina salvo que sea necesario para alcanzar el estado indicado por el módulo.

A continuación, se muestra un ejemplo para copiar el fichero /etc/hosts de la máquina en la que se ejecuta el comando *ansible* (el entorno de realización de prácticas) a todas las máquinas de la categoría *tag_ansible_alucloud$ID*. Dicho fichero se copia en destino a la ruta /tmp/hosts. Nótese que si hubiera que copiar el fichero a muchas máquinas se podría especificar el parámetro –f 10 para, por ejemplo, realizar 10 copias en paralelo. Esto permite acelerar la ejecución de la operación. El número de máquinas configuradas es el mismo, solo que se configuran en paralelo tantas como las indicadas en dicho argumento.

```
:~$ ansible tag_ansible_alucloud$ID -m copy -a "src=/etc/hosts
dest=/tmp/hosts"

ec2-3-237-199-118.compute-1.amazonaws.com | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": true,
    "checksum": "a777be51c79c607edd61c8e12cd9b775ddc8a6c6",
    "dest": "/tmp/hosts",
    "gid": 1000,
    "group": "ubuntu",
    "md5sum": "1463508f28edb4d6d5ae349b20e00409",
    "mode": "0664",
```

```
    "owner": "ubuntu",
    "size": 221,
    "src": "/home/ubuntu/.ansible/tmp/ansible-tmp-1601703560.8953671-68911996273451/source",
    "state": "file",
    "uid": 1000
}
ec2-34-233-71-129.compute-1.amazonaws.com | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": true,
    "checksum": "a777be51c79c607edd61c8e12cd9b775ddc8a6c6",
    "dest": "/tmp/hosts",
    "gid": 1000,
    "group": "ubuntu",
    "md5sum": "1463508f28edb4d6d5ae349b20e00409",
    "mode": "0664",
    "owner": "ubuntu",
    "size": 221,
    "src": "/home/ubuntu/.ansible/tmp/ansible-tmp-1601703560.907919-123156050382232/source",
    "state": "file",
    "uid": 1000
}
```

El comando devuelve información sobre el fichero creado en destino. Esta información está en formato JSON (JavaScript Object Notation) [13].

Puedes verificar que el fichero está en la máquina remota listando remotamente su contenido:

```
:~$ ansible tag_ansible_alucloud$ID -a "cat /tmp/hosts"

ec2-34-233-71-129.compute-1.amazonaws.com | CHANGED | rc=0 >>
127.0.0.1 localhost

# The following lines are desirable for IPv6 capable hosts
::1 ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
ff02::3 ip6-allhosts
ec2-3-237-199-118.compute-1.amazonaws.com | CHANGED | rc=0 >>
127.0.0.1 localhost

# The following lines are desirable for IPv6 capable hosts
::1 ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
ff02::3 ip6-allhosts
```

Fíjate que el contenido del fichero puede cambiar dependiendo de la configuración del despliegue de prácticas. A estas alturas, quizá ya te habrás dado cuenta del formato de salida de los comandos de Ansible, que resalta en color verde cuando no es necesario realizar un cambio, con amarillo los cambios correctamente aplicados y en rojo (indicando como *failed*) las ejecuciones que han provocado algún fallo.

Ejemplo Práctico: Instalación de Paquetes

Imagina que deseas utilizar el comando *fortune* [14], de dudosa utilidad práctica más que para ejemplificar, en las máquinas Ubuntu de tu infraestructura. A continuación, se describe una posible secuencia de acciones utilizando Ansible.

En primer lugar, comprobamos que el comando no está instalado en las máquinas a configurar:

```
:~$ ansible tag_ansible_alucloud$ID -a "fortune"
ec2-3-237-199-118.compute-1.amazonaws.com | FAILED | rc=2 >>
[Errno 2] No such file or directory: b'fortune'
ec2-34-233-71-129.compute-1.amazonaws.com | FAILED | rc=2 >>
[Errno 2] No such file or directory: b'fortune'
```

El error obtenido indica que dicho comando no existe en la máquina remota. Instalamos el paquete que contiene dicho comando. Fíjate que Ansible no conoce qué paquete contiene dicho comando, por lo que es responsabilidad del usuario averiguar el paquete que incluye dicha utilidad. En este caso, en GNU/Linux Ubuntu, el paquete se denomina *fortune:*

```
:~$ ansible tag_ansible_alucloud$ID -m apt -a "name=fortune state=present
update_cache=yes"

c2-3-237-199-118.compute-1.amazonaws.com | FAILED! => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "msg": "Failed to lock apt for exclusive operation"
}
ec2-34-233-71-129.compute-1.amazonaws.com | FAILED! => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "msg": "Failed to lock apt for exclusive operation"
}
```

El comando anterior está ejecutando el módulo *apt*, con los argumentos *name=fortune* y *state=present*, sobre todas las máquinas de la categoría *tag_ansible_alucloud$ID,* habiendo hecho previamente una actualización del listado de paquetes. Fíjate que se obtiene un error porque es necesario permisos de superusuario en la máquina remota para instalar un paquete. Para solucionarlo, deberás incluir el argumento --become. Esto permite escalar de privilegios tras la conexión a la máquina remota usando el usuario ubuntu, para permitir la instalación de paquetes.

```
:~$ ansible tag_ansible_alucloud$ID -m apt -a "name=fortune state=present
update_cache=yes" --become

ec2-3-237-199-118.compute-1.amazonaws.com | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1601703893,
    "cache_updated": true,
    "changed": true,
    "stderr": "",
```

```
    "stderr_lines": [],
    "stdout": "Reading package lists...\nBuilding dependency
tree...\nReading state information...\nThe following additional packages
will be installed:\n  fortune-mod fortunes-min librecode0\nSuggested
packages:\n  x11-utils\nThe following NEW packages will be
…
```

Una vez instalado, ya podemos ejecutar de forma remota el comando *fortune*, que únicamente muestra por pantalla una cadena de caracteres con una cita.

```
:~$ ansible tag_ansible_alucloud$ID -a "fortune"
c2-34-233-71-129.compute-1.amazonaws.com | CHANGED | rc=0 >>
On the night before her family moved from Kansas to California, the little
girl knelt by her bed to say her prayers.  "God bless Mommy and Daddy and
Keith and Kim," she said.  As she began to get up, she quickly added, "Oh,
and God, this is goodbye.  We're moving to Hollywood."
ec2-3-237-199-118.compute-1.amazonaws.com | CHANGED | rc=0 >>
Emerson's Law of Contrariness:
      Our chief want in life is somebody who shall make us do what we
      can.  Having found them, we shall then hate them for it.
```

Fíjate como el comando se ejecuta en todas las instancias, lo que permite automatizar la ejecución de comandos remotos sobre un conjunto de máquinas, tal y como hacen otras herramientas [32].

Ejemplo Práctico: Despliegue Automático de un Servidor Web

Imagina que deseas desplegar un servidor web en cada una de las máquinas de tu infraestructura para que muestre un determinado mensaje por defecto. Esto es útil para configurar unos determinados roles en algunas de las máquinas de tu infraestructura. Veamos cómo es posible simplificar esta tarea utilizando Ansible.

En primer lugar, instalamos un servidor (elegimos Apache [15] por comodidad y facilidad de instalación). En Ubuntu, el servidor web Apache se encuentra disponible en el paquete *apache2* (en otras distribuciones de GNU/Linux el paquete se llama *httpd*). Es necesario indicar que se realice un `apt-get update` previamente para que se actualicen las localizaciones de los paquetes, mediante la opción `update_cache=yes` del módulo `apt` de Ansible.

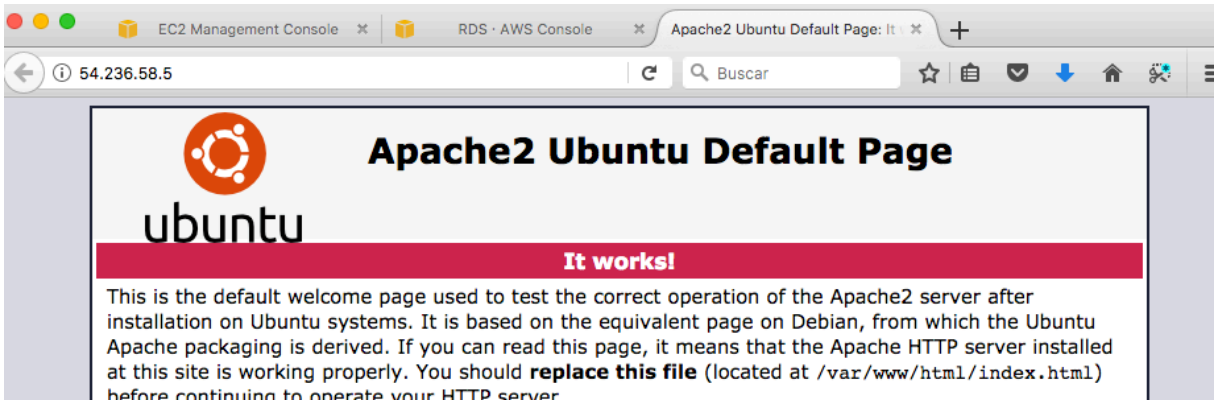
```
:~$ ansible tag_ansible_alucloud$ID -m apt -a "name=apache2 state=present
update_cache=yes" --become

ec2-3-237-199-118.compute-1.amazonaws.com | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1601704045,
    "cache_updated": true,
    "changed": true,
    "stderr": "",
    "stderr_lines": [],
    "stdout": "Reading package lists...\nBuilding dependency
tree...\nReading state information...\nThe following packages were
automatically ins
…
```

A continuación, iniciamos el servidor mediante el módulo *service* [33] de Ansible:

```
:~$ ansible tag_ansible_alucloud$ID -m service -a "name=apache2
state=started" --become

c2-34-233-71-129.compute-1.amazonaws.com | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "name": "apache2",
    "state": "started",
    "status": {
        "ActiveEnterTimestamp": "Sat 2020-10-03 05:47:32 UTC",
        "ActiveEnterTimestampMonotonic": "1019250575",
…
```

En realidad, con la instalación del paquete apache2 se inicia automáticamente el servicio, pero, de esta manera, también sabes cómo gestionar servicios desde Ansible.

En este punto ya podemos verificar que el servidor Apache está en ejecución en nuestras instancias. Nos conectamos con un navegador a cualquiera de ellas:

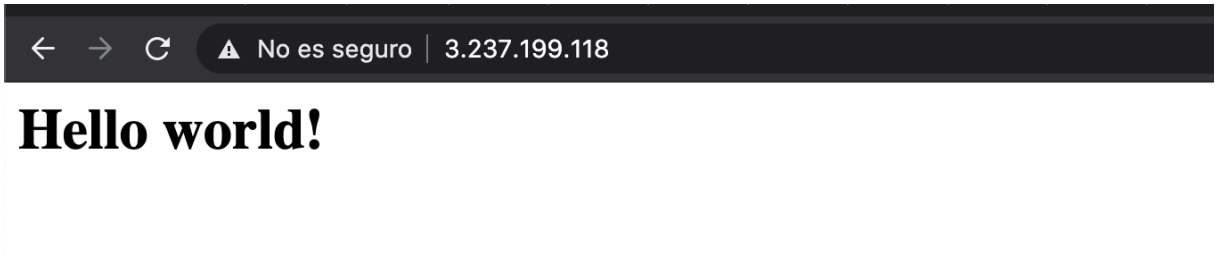


Se muestra la página por defecto del servidor Apache. A continuación, cambiamos el mensaje. Usamos para ello el módulo *copy* especificando la ruta absoluta en la máquina remota que contiene el fichero que por defecto se muestra al acceder al servidor web mediante un navegador web. Fíjate que hace falta permisos apropiados para modificar dicho fichero.

```
:~$ ansible tag_ansible_alucloud$ID -m copy -a
'dest=/var/www/html/index.html content="<h1>Hello world!</h1>"' --become
ec2-54-89-97-254.compute-1.amazonaws.com | success >> {
    "changed": true,
    "dest": "/var/www/index.html",
    "gid": 0,
    "group": "root",
    "md5sum": "fb80156296e1e8f20637d9ef870e0bb6",
    "mode": "0644",
    "owner": "root",
    "size": 21,
    "src": "/home/ubuntu/.ansible/tmp/ansible-tmp-1404728861.04-
22961596373489/source",
    "state": "file",
```

```
    "uid": 0
}
…
```

Refrescamos el navegador web y vemos el contenido actualizado:



Aunque facilitar la ejecución remota y la gestión concurrente de múltiples máquinas es un aspecto importante, la característica fundamental de Ansible (y de otras herramientas de DevOps) es la creación de recetas que permiten englobar la configuración deseada de una máquina. Esto ayuda a la definición y aplicación de un determinado rol a las máquinas de una arquitectura de una aplicación en la nube.

Creación de Recetas en Ansible: Playbooks

Un *playbook* [16] es una receta que permite agregar la configuración que debe ser aplicada en un conjunto de máquinas (paquetes a instalar, configuración de usuarios, configuración de ficheros, etc.). Los playbooks se escriben en lenguaje YAML [17] e incluyen declaración de variables, categoría de máquinas sobre las que se ejecutará, tareas a ejecutar y si es necesario indicación de que son necesarios privilegios de superusuario en la máquina a configurar.

A continuación, veremos algunos ejemplos ilustrativos de uso de playbooks en Ansible.

Ejemplo de Playbook sencillo: Instalación Remota de Fortune

A continuación, se muestra un ejemplo de un playbook sencillo para automatizar el despliegue de la aplicación *fortune* en un conjunto de máquinas:

```
~$ cat /opt/cursoaws/ansible/playbook-fortune.yml
---
- hosts: tag_ansible_alucloudXX
  become: true
  tasks:
    - apt: pkg=fortune state=latest
```

Modifica XX por tu identificador de usuario. Si eres alucloud02 deberás indicar tag_ansible_alucloud02. Puedes copiar desde este boletín y crear un nuevo fichero en la máquina a la que estás conectado con *vim* (o con cualquier otro editor disponible), aunque lo más sencillo es crear dicho fichero usando *sed* de la siguiente manera:

```
~$ sed s/XX/$ID/g /opt/cursoaws/ansible/playbook-fortune.yml >
$HOME/playbook-fortune.yml
```

Esto habrá creado el fichero playbook-fortune.yml en tu carpeta de usuario donde ya se habrá sustituido XX por tu identificador de usuario. Si vas a editar tú mismo el fichero, ten en cuenta que YAML es un lenguaje muy sensible a la indentación [34] y al número de espacios por lo que asegúrate de respetar el formato del lenguaje. En concreto, el playbook de arriba consta de varias secciones:

1. **Hosts:** Indica las máquinas sobre los que se realizarán las tareas indicadas en la sección **Tasks.** Para ello se indica la categoría de las máquinas que debe corresponder a alguna de las indicadas en el fichero de inventario de Ansible (en nuestro caso, el inventario dinámico). Es posible indicar *all* para ejecutar el playbook sobre todas las máquinas de dicho fichero.
2. **Become:** Indica que es necesario escalar privilegios para realizar las tareas. Si es necesario permisos de superusuario para realizar alguna de las tareas, como por ejemplo la instalación de un paquete, es necesario indicarlo. Da lo mismo indicar *true* que *yes*, pues son equivalentes. Tienes más información al respecto en [65].
3. **Tasks:** Indica el listado de tareas o módulos que deben ejecutarse en la máquina destino. Estas tareas se corresponden con los módulos cuyo listado puede encontrarse en [12].

La ejecución de dicho playbook se debe realizar usando el comando *ansible-playbook* y provoca el siguiente resultado:

```
:~$ ansible-playbook playbook-fortune.yml

LAY [tag_ansible_alucloud00]
*********************************************************************
*****

TASK [Gathering Facts]
*********************************************************************
************
ok: [ec2-34-233-71-129.compute-1.amazonaws.com]
ok: [ec2-3-237-199-118.compute-1.amazonaws.com]

TASK [apt]
*********************************************************************
***********************
changed: [ec2-3-237-199-118.compute-1.amazonaws.com]
changed: [ec2-34-233-71-129.compute-1.amazonaws.com]

PLAY RECAP
*********************************************************************
***********************
ec2-3-237-199-118.compute-1.amazonaws.com : ok=2    changed=1
unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
ec2-34-233-71-129.compute-1.amazonaws.com : ok=2    changed=1
unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

La ejecución del comando anterior habrá funcionado correctamente dado que las dos instancias involucradas tienen una cuenta de usuario *ubuntu* sobre la que se puede ejecutar comandos mediante *sudo* sin necesidad de especificar contraseña. En caso de tener que especificar una contraseña para ejecutar comandos mediante sudo es posible invocar el comando ansible-playbook con la opción --ask-sudo-pass (-K). De hecho, si alguna vez ejecutas un playbook y se queda "atascado" es posible que sea debido a que el comando sudo esté esperando la contraseña [21].

Podemos verificar que efectivamente se ha instalado remotamente dicho paquete, mediante la ejecución remota sobre una de las máquinas directamente con SSH. También se podría ejecutar a través de Ansible pero así se muestran diferentes formas:

```
:~$ ssh -i alucloud$ID-priv.pem ubuntu@ec2-54-89-87-166.compute-
1.amazonaws.com /usr/games/fortune

Habit is habit, and not to be flung out of the window by any man, but
coaxed down-stairs a step at a time.
            -- Mark Twain, "Pudd'nhead Wilson's Calendar
```

Además, dado que el comando fortune está accesible desde la variable PATH, ni siquiera es necesario poner la ruta absoluta para invocar el comando. Alternativamente, es posible realizar la ejecución remota de la aplicación sobre todas tus instancias a configurar de la siguiente manera:

```
:~$ ansible tag_ansible_alucloud$ID -a fortune
ec2-54-89-97-254.compute-1.amazonaws.com | success | rc=0 >>
Your manuscript is both good and original, but the part that is good is
not original and the part that is original is not good.
            -- Samuel Johnson

ec2-54-89-87-166.compute-1.amazonaws.com | success | rc=0 >>
Q:    What do you say to a New Yorker with a job?
A:    Big Mac, fries and a Coke, please!
```

Ejemplo de Playbook: Despliegue Automático de Servidor Web

A continuación, se muestra otro ejemplo de Playbook para desplegar de forma automática un servidor web con una página web por defecto. Fíjate como este ejemplo complementa el uso de Ansible para la ejecución remota de comandos, tal y como se hizo al principio de la práctica, para un fin similar. Recuerda indicar correctamente la categoría de máquinas sobre las que se ejecutarán las tareas de este playbook.

```
:~$ cat /opt/cursoaws/ansible/playbook-apache.yml
---
- hosts: tag_ansible_aluclouXX
  become: true
  tasks:
  - name: ensure apache is at the latest version
    apt: pkg=apache2 state=latest update_cache=yes
  - name: ensure apache is running
    service: name=apache2 state=started
  - copy: dest=/var/www/html/index.html content="<h1>Hello world!</h1>"
```

En este caso observa como las dos primeras tareas llevan asociado un nombre, que no es más que una descripción de alto nivel de la tarea (que se mostrará durante la ejecución del playbook), mientras que la última tarea no lleva asociado un nombre. Fíjate como, en el caso de especificar un nombre, el guión va asociado al *name* y no a la tarea en sí (peculiaridades de YAML).

Creamos un fichero en nuestra cuenta de usuario donde hayamos sustituido el XX por tu identificador de usuario:

```
:~$ sed s/XX/$ID/g /opt/cursoaws/ansible/playbook-apache.yml >
$HOME/playbook-apache.yml
```

La ejecución de dicho playbook se realiza a continuación:

```
:~$ ansible-playbook playbook-apache.yml

PLAY [tag_ansible_alucloud00]
************************************************

TASK [setup]
*******************************************************************
ok: [ec2-54-205-242-52.compute-1.amazonaws.com]
ok: [ec2-54-197-168-238.compute-1.amazonaws.com]

TASK [ensure apache is at the latest version]
********************************
ok: [ec2-54-197-168-238.compute-1.amazonaws.com]
ok: [ec2-54-205-242-52.compute-1.amazonaws.com]

TASK [ensure apache is running]
*********************************************
ok: [ec2-54-205-242-52.compute-1.amazonaws.com]
ok: [ec2-54-197-168-238.compute-1.amazonaws.com]

TASK [copy]
*********************************************************************
ok: [ec2-54-197-168-238.compute-1.amazonaws.com]
ok: [ec2-54-205-242-52.compute-1.amazonaws.com]

PLAY RECAP
*********************************************************************
ec2-54-197-168-238.compute-1.amazonaws.com : ok=4    changed=0
unreachable=0    failed=0
ec2-54-205-242-52.compute-1.amazonaws.com : ok=4    changed=0
unreachable=0    failed=0
```

Puedes comprobar fácilmente que el servidor web está funcionando y que se ha copiado el contenido indicado en el fichero /var/www/index.html abriendo un navegador web y dirigiéndolo a la dirección IP de la(s) máquina(s) de la categoría tag_ansible_alucloudXX.

Configuración Automática de Máquina Basada en Roles

Una de las características más interesantes de la configuración automática de máquinas mediante herramientas de tipo *devops* es la configuración basada en roles. Se utiliza un Cloud de tipo IaaS para aprovisionar potencia de cómputo en la forma de máquinas virtuales que posteriormente deben desempeñar un determinado rol (servidor web, servidor de base de datos, servidor de aplicaciones, servidor de mensajería o cualquier otro rol necesario para soportar la arquitectura de una aplicación en la nube). Por ello, la receta (el playbook en Ansible) debe especificar toda la configuración necesaria para configurar una máquina e integrarla dentro de la arquitectura de una aplicación.

Esto permite que, si una instancia falla, entonces es posible aprovisionar una nueva y configurarla automáticamente para que desempeñe el mismo rol que la instancia que ha fallado. Esto facilita la recuperación ante errores de una aplicación Cloud. También ante casos de elasticidad horizontal en el

que se aprovisiona dinámicamente instancias adicionales para desempeñar un cierto rol para gestionar apropiadamente aumentos en la carga de trabajo.

A modo de ejemplo, se desea diseñar una receta en Ansible para configurar automáticamente una máquina (GNU/Linux, basada en Ubuntu) de la siguiente manera:

- Debe haber una cuenta de usuario llamado *anuser* con la contraseña *vaquerito*.
- Debe tener desplegada la versión más reciente de Apache Tomcat [18]
- Debe desplegarse la siguiente aplicación web [19] dentro de Apache Tomcat. Aunque esta aplicación no es más que una aplicación de ejemplo, al final se trata de un fichero WAR y el procedimiento de despliegue sería similar al utilizado para cualquier otra aplicación web basada en Java.

Para ello, se propone el siguiente playbook de Ansible (contenido del fichero playbook-webapp.yml):

```
:~$ cat /opt/cursoaws/ansible/playbook-webapp.yml
---
- hosts: tag_ansible_alucloudXX
  become: yes
  tasks:
  - user: name=anuser
password=$6$mAQYj/gR6wYvd$l0AfD30LdDv0j6IkBtal0VHZxIHjTLWM00iyei2h6AZ97vvfm
.OqpEyUfNJwIApU87ATR7nxYuJ2sNGl6Gbuj/ shell=/bin/bash
  - name: ensure Apache Tomcat is at the latest version
    apt: pkg=tomcat7 state=latest update_cache=yes
  - name: ensure Apache Tomcat is running
    service: name=tomcat7 state=started
  - name: Retrieve and deploy web app
    get_url: url=http://tomcat.apache.org/tomcat-7.0-doc/appdev/sample/{{
item }} dest=/var/lib/tomcat7/webapps/{{ item }}
    with_items:
     - sample.war
```

El playbook incluye las acciones que se ejecutarán sobre las máquinas de la categoría *tag_ansible_alucloudXX*. Los comandos se ejecutarán en la máquina remota como el usuario *ubuntu* con privilegios de superusuario (mediante *sudo*). Se realizan las siguientes tareas:

- Se crea el usuario *anuser* utilizando el módulo *user*. La contraseña hay que especificarla de forma cifrada, tal y como aparecería en el fichero */etc/shadow* de cualquier máquina Linux. Para averiguar la forma cifrada de una determinada contraseña puedes utilizar el comando *mkpasswd*[1]

[1] Con el comando *mkpasswd –m sha-512* e indicando la contraseña.

- Se instala la última versión del paquete tomcat9 (que contiene Apache Tomcat 9). Si ya estaba instalado, entonces se actualiza a la última versión disponible en los repositorios de paquetes de la distribución Linux utilizada (en este caso Ubuntu 20.04 LTS).
- Se inicia el servicio Apache Tomcat, que estará escuchando por defecto en el puerto 8080 [19].
- Se descarga el fichero *sample.war* desde la siguiente URL (https://tomcat.apache.org/tomcat-9.0-doc/appdev/sample/sample.war). Para ello se utiliza el módulo *get_url* con una sintaxis (el uso de *with_items*) que se utiliza principalmente para descargar múltiples ficheros. Se utiliza para que la conozcas, puesto que en este caso tan solo se necesita descargar un solo fichero. El fichero sample.war se descarga en el directorio */var/lib/tomcat9/webapps*. Esto permite que Apache Tomcat despliegue automáticamente la aplicación y esté disponible en la siguiente URL: **¡Error! Referencia de hipervínculo no válida.**

Adaptamos el fichero para que indique nuestro código de usuario:

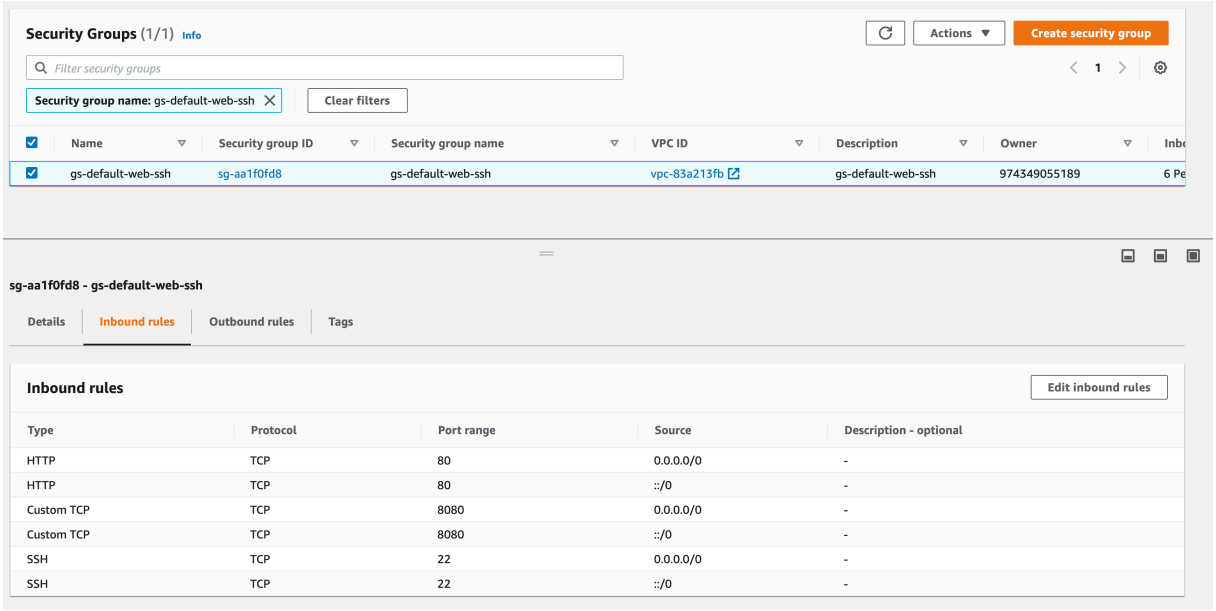
```
:~$ sed s/XX/$ID/g /opt/cursoaws/ansible/playbook-webapp.yml >
$HOME/playbook-webapp.yml
```

Finalmente, ejecutamos el playbook:

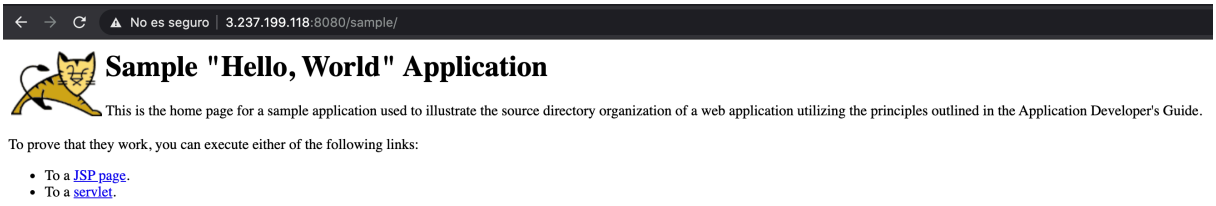
```
$ ansible-playbook playbook-webapp.yml

PLAY [tag_ansible_alucloud00]
*********************************************

GATHERING FACTS
*************************************************************
ok: [ec2-54-89-87-166.compute-1.amazonaws.com]
ok: [ec2-54-89-97-254.compute-1.amazonaws.com]

TASK: [user name=anuser
password=$6$mAQYj/gR6wYvd$l0AfD30LdDv0j6IkBtal0VHZxIHjTLWM00iyei2h6AZ97vvfm
.OqpEyUfNJwIApU87ATR7nxYuJ2sNGl6Gbuj/ shell=/bin/bash] ***
changed: [ec2-54-89-97-254.compute-1.amazonaws.com]
changed: [ec2-54-89-87-166.compute-1.amazonaws.com]

TASK: [ensure Apache Tomcat is at the latest version]
************************
changed: [ec2-54-89-97-254.compute-1.amazonaws.com]
changed: [ec2-54-89-87-166.compute-1.amazonaws.com]

TASK: [ensure Apache Tomcat is running]
***********************************
ok: [ec2-54-89-97-254.compute-1.amazonaws.com]
ok: [ec2-54-89-87-166.compute-1.amazonaws.com]

TASK: [Retrieve and deploy web app]
*****************************************

changed: [ec2-54-89-97-254.compute-1.amazonaws.com] => (item=sample.war)
changed: [ec2-54-89-87-166.compute-1.amazonaws.com] => (item=sample.war)

PLAY RECAP
*****************************************************************
```

```
ec2-54-89-87-166.compute-1.amazonaws.com : ok=5    changed=3
unreachable=0    failed=0
ec2-54-89-97-254.compute-1.amazonaws.com : ok=5    changed=3
unreachable=0    failed=0
```

Para conectar al puerto 8080 de cualquiera de las dos instancias, el grupo de seguridad con el que han sido desplegadas las instancias debe permitir el tráfico de entrada a dicho puerto. En esta práctica, el grupo de seguridad utilizado para las instancias ya permite el tráfico TCP entrante a dicho puerto, por lo que no hay que añadir ninguna nueva regla:



Si conectamos a la URL para acceder a la aplicación web (en cualquiera de las dos instancias) podemos comprobar que la aplicación se ha desplegado correctamente: http:// 54.205.187.9:8080/sample



Si obtienes un error asegúrate de que estás indicando el protocolo http (por defecto los navegadores usan automáticamente el protocolo https y este redirige las peticiones al puerto 443). Además, asegúrate de estar indicando el puerto 8080 como destino de la conexión.

Efectivamente, con la creación del playbook has podido configurar de forma automatizada las diferentes instancias para que soporten el rol de servidores de aplicaciones. Obviamente, la configuración específica depende del tipo de aplicación a desplegar.

Aprovisionamiento de Servidor de Base de Datos

Es habitual en las arquitecturas de aplicaciones en la nube la existencia de servidores de bases de datos. Una base de datos relacional se puede externalizar a un servicio en la nube como es el caso de Amazon RDS [35]. Sin embargo, a menudo puede ser interesante desplegar un servidor de base de datos propio, por diferentes razones, como reducir costes, disponer de mayor control sobre el Sistema Gestor de Base de Datos, etc. En este ejemplo realizaremos la configuración automática de un servidor (o varios servidores) de base de datos MySQL mediante un playbook de Ansible realizando la carga de datos a partir de un fichero almacenado en un bucket de S3. Esto puede permitir una forma sencilla de restaurar una base de datos para ofrecer alta disponibilidad en caso de fallo (si bien el esquema Multi-AZ en RDS [36] o los propios servicios de replicación de MySQL [37] pueden ser una alternativa conveniente para este propósito). Se diseñará una receta para garantizar una determinada configuración software en tu grupo de instancias ya aprovisionadas, etiquetadas con la etiqueta tag_ansible_alucloudXX.

Se desea que dichas máquinas tengan:

- MySQL Server instalado y actualizado a la última versión. El servicio MySQL debe quedar iniciado tras la instalación. El usuario por defecto para la conexión al servidor MySQL es *root* y sin contraseña)
- Una base de datos llamada *world* que será rellenada en base al fichero almacenado en `s3://cursocloudaws-public/world_innodb_small.sql`

Para ello, se creará un playbook de Ansible usando los módulos que ofrece [12] para realizar la instalación de MySQL server (módulo *apt* [31]), la descarga del fichero de Amazon S3 (módulo *s3* [38]), la creación de la base de datos (módulo mysql_db [39]) y la importación de los datos a la base de datos, que se puede hacer con el propio módulo anterior.

```
:~$ cat /opt/cursoaws/ansible/playbook-recipe-mysql.yml
---
- hosts: tag_ansible_alucloudXX
  become: true
  tasks:
  - name: Install MySQL (This might take a while. DO NOT INTERRUPT)
    apt: pkg=mysql-server state=latest update_cache=yes
  - name: Install python-mysqldb (to configure MySQL from Ansible)
    apt: pkg=python-mysqldb state=latest
  - name: ensure MySQLd is running
    service: name=mysql state=started
  - name: Retrieve DB from Amazon S3
    get_url:
url=https://s3.amazonaws.com/cursocloudaws/world_innodb_small.sql
dest=/tmp/world_innodb_small.sql
  - name: Create DB
    mysql_db: name=world state=present login_user=root login_password=""
  - name: Import DB
    mysql_db: name=world state=import login_user=root login_password=""
target="/tmp/world_innodb_small.sql"
```

Fíjate como en el playbook se da un nombre a todas las acciones para identificar las diferentes acciones posteriormente en el log de ejecución. En primer lugar, se usa el módulo `apt` para la instalación del paquete `mysql-server`, que es el nombre del paquete de instalación de MySQL Server en las distribuciones Ubuntu. Fíjate en el uso de `update_cache` para actualizar las ubicaciones de los paquetes de los repositorios. A continuación, se asegura el inicio del servicio `mysql` a usando el módulo `service`.

Para descargar el fichero `world_innodb_small.sql` que contiene los comandos SQL para reconstruir la base de datos se descarga del fichero `world_innodb_small.sql` disponible en la carpeta `ansible` del bucket `cursocloudaws`, para ser almacenado en el directorio `/tmp` de las máquinas remotas sobre las que se ejecuta este playbook. Finalmente, se crea la base de datos usando el módulo mysql_db en el que se especifica el nombre de la base de datos que se pretende crear (que esté presente) indicando los parámetros de conexión al servidor de base de datos (usuario root y password vacío). Luego, se importan los datos a la base de datos desde el fichero previamente descargado.

Modificamos XX por tu identificador de usuario y a ejecutar el playbook:

```
:~$ sed s/XX/$ID/g /opt/cursoaws/ansible/playbook-recipe-mysql.yml >
$HOME/playbook-recipe-mysql.yml
:~$ ansible-playbook playbook-recipe-mysql.yml
PLAY [tag_ansible_alucloud00]
***********************************************

GATHERING FACTS
*************************************************************
ok: [ec2-54-89-81-208.compute-1.amazonaws.com]

TASK: [Install MySQL (This might take a while. DO NOT INTERRUPT)]
*************
changed: [ec2-54-89-81-208.compute-1.amazonaws.com]

TASK: [Install python3-mysqldb (to configure MySQL from Ansible)]
**************
changed: [ec2-54-89-81-208.compute-1.amazonaws.com]

TASK: [ensure MySQLd is running]
******************************************
ok: [ec2-54-89-81-208.compute-1.amazonaws.com]

TASK: [Copy BOTO's credential file to remote location]
************************
changed: [ec2-54-89-81-208.compute-1.amazonaws.com]

TASK: [Retrieve DB from Amazon S3]
****************************************
changed: [ec2-54-89-81-208.compute-1.amazonaws.com]

TASK: [Create DB]
*********************************************************
changed: [ec2-54-89-81-208.compute-1.amazonaws.com]

PLAY RECAP
********************************************************************
ec2-54-89-81-208.compute-1.amazonaws.com : ok=7    changed=5
unreachable=0    failed=0
```

Puedes verificar que se ha procedido a la configuración adecuada de la instancia conectándote a ella y verificando la existencia de la base de datos `world`. Deberás conectarte como usuario root y sin contraseña.

```
:~$ ssh -i $HOME/alucloud$ID-priv.pem ubuntu@ec2-54-89-81-208.compute-
1.amazonaws.com
ubuntu@ip-10-141-186-149:~$ sudo mysql -u root -p
Enter password: ← Pulsar Enter
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| world              |
+--------------------+
4 rows in set (0.00 sec)
mysql> use world;
Database changed
mysql> show tables;
+-----------------+
| Tables_in_world |
+-----------------+
| City            |
| Country         |
| CountryLanguage |
+-----------------+
3 rows in set (0.01 sec)
mysql> select * from City;
+-----+----------------------------+-------------+----------------------+------------+
| ID  | Name                       | CountryCode | District             | Population |
+-----+----------------------------+-------------+----------------------+------------+
|   1 | Kabul                      | AFG         | Kabol                |    1780000 |
|   2 | Qandahar                   | AFG         | Qandahar             |     237500 |
…
| 249 | Campos dos Goytacazes      | BRA         | Rio de Janeiro       |     398418 |
| 250 | Mauá                       | BRA         | São Paulo            |     375055 |
+-----+----------------------------+-------------+----------------------+------------+
250 rows in set (0.00 sec)
```

Playbooks Avanzados: Importación, Templates e Iteración

En esta sección se cubren algunos aspectos adicionales de Ansible, como la importación de playbooks, el uso de templates [41] de Jinja2 [42] y la iteración o ejecución de módulos con diferentes variables. Al igual que en las secciones anteriores se ejemplificará el uso de Ansible a través de un ejemplo práctico.

Imagina que necesitas configurar unas máquinas con unas determinadas cuentas de usuario de manera que dispongan de un determinado fichero de configuración con unas credenciales instaladas, donde cada usuario tiene unas credenciales diferentes. Por ejemplo, asumiremos que queremos instalar Boto [40] en dichas máquinas para que los usuarios puedan acceder a AWS usando sus credenciales (que inyectaremos automáticamente en sus cuentas de usuario por medio del playbook).

Además, se desea separar el playbook en dos diferentes, uno para definir las variables y otro para indicar las tareas a ejecutar, haciendo uso de las características de importación de módulos [44] de Ansible. Además, se desea que los usuarios reciban un cierto mensaje cuando se conecten (vía SSH) a la(s) máquina(s) configurada(s).

Para ello definimos en primer lugar un playbook con la declaración de variables necesarias. Tienes disponible dicho fichero ya creado en `/opt/cursoaws/ansible` en el entorno de prácticas:

```
:~$ cat /opt/cursoaws/ansible/vars-advplaybook.yml
    accounts:
        - { name: "user00", pw: "QPeR2gsw4MZEk", ak: "AKIAJAIQMN42O7AGSC6A", sk:
"ft0ftS7FD0H5L5Tu3m/Dg2D
JoIb/GZ6niIWZbFeW" }
        - { name: "user01", pw: "V22YKgEokQv6o", ak: "AKIAIQZICAOA3T7AXXQP", sk:
"ssPvGujteBv0V/KRsJiY98R
ijNVo/Mg221VlLSBn" }
        - { name: "user02", pw: "z0iVK2EJ1JOCZ", ak: "AKIAI25CX6A55PKILAPQ", sk:
"wOPwD1UeEBgaKvBf3wn5x9k
h/6RTVB7amKGkm3ZQ" }
```

Se declara una variable llamada accounts que contiene una lista de cuentas de usuario donde para cada uno se especifica el nombre (atributo `name`), el password a utilizar (o mejor dicho la forma cifrada de la contraseña, con el atributo `pw`) y el Access Key ID (atributo `ak`) y el Secret Access Key (atributo `sk`). El nombre de los atributos es completamente arbitrario. Las claves que aparecen en el listado son inventadas, para el ejemplo.

A continuación, definimos un playbook llamado `tasks-advplaybook.yml` para ejecutar las tareas. Tienes disponible dicho fichero ya creado en `/opt/cursoaws/ansible` en el entorno de prácticas:

```
:~$ cat /opt/cursoaws/ansible/tasks-advplaybook.yml

    ####################################
    #  --- User's Accounts --------  #
    ####################################

    - name: Create user accounts
      user: name={{ item.name }} password={{ item.pw }} shell=/bin/bash
generate_ssh_key=yes
      with_items: "{{accounts}}"

    - name: Create BOTO credentials file
      template: src=/opt/cursoaws/ansible/boto.j2 dest=/home/{{ item.name }}/.boto
owner={{ item.name }} group={{ item.name }} mode=0600
      with_items: "{{accounts}}"

    ####################################################
    #  --------- Software  -------------------------  #
    ####################################################

    - name: Install PIP to bootstrap Boto (python-pip)
      apt: pkg=python3-pip state=latest

    - name: Install boto
      pip: name=boto state=latest


    ####################################################
    #  --------- Configuration ----------------------  #
    ####################################################

    - name: Copy motd.tail
      copy: src=/opt/cursoaws/ansible/motd.tail dest=/etc/motd force=yes owner=root
group=root mode=
```

```
0644
```

. El contenido de dicha plantilla de Jinja2 llamada boto.j2 es el siguiente:

```
:~$ cat /opt/cursoaws/ansible/boto.j2
[Credentials]
aws_access_key_id = {{ item.ak }}
aws_secret_access_key = {{ item.sk }}
```

Fíjate cómo el contenido del fichero referencia a los atributos `ak` y `sk` de un `item` que se instanciará desde la correspondiente tarea que usa el módulo `user`, para cada uno de los valores de la lista definida en la variable `accounts`. Esto permite la invocación del módulo `user` para la creación de los diferentes usuarios en las máquinas a configurar.

A continuación se utiliza el módulo `template` [41] que permite copiar el fichero boto.j2 al fichero .boto del directorio home de cada usuario, modificando su contenido de manera que, por ejemplo, para el usuario `user01`, el fichero `/home/user01/.boto` pasa a tener el siguiente contenido:

```
[Credentials]
aws_access_key_id = AKIAIQZICAOA3T7AXXQP
aws_secret_access_key = ssPvGujteBv0V/KRsJiY98R
ijNVo/Mg221VlLSBn
```

Esto permite parametrizar un fichero en función del valor de ciertas variables, lo que facilita la configuración para determinadas aplicaciones.

A continuación, se procede a la instalación de Boto (previamente debe instalarse Pip) y, finalmente, se copia el fichero `/opt/cursoaws/ansible/motd.tail` a `/etc/motd` en la máquina remota. Esto permitirá mostrar el contenido de ese fichero cuando un usuario haga login en el sistema a través de un shell interactivo. Fíjate que se podría haber descargado ese fichero de una URL o de un bucket de S3 pero, así ves diferentes formas de copiar ficheros en las máquinas a configurar.

A continuación, creamos un playbook que referencie los dos playbooks anteriores, el de definición de variables (`vars-advplaybook.yml`) y el de definición de tareas a ejecutar (`tasks-advplaybook.yml`). Tienes disponible dicho fichero ya creado en `/opt/cursoaws/ansible` en el entorno de prácticas.

```
:~$ cat /opt/cursoaws/ansible/advplaybook00.yml
---
- hosts: tag_ansible_alucloudXX
  become: yes
  tasks:
    - include_vars: vars-advplaybook.yml
    - include: tasks-advplaybook.yml
```

Fíjate como dicho playbook hace uso de los módulos `include_vars` [43] e `include` para incluir los playbooks previamente definidos. Esto permite reutilizar definiciones de variables y tareas.

Procedemos a modificarlo para especificar tu indicador de alumno (a estas alturas ya sabrás que si eres el alumno alucloud09 deberás indicar como hosts: tag_ansible_alucloud09). Copiaremos en tu cuenta

de usuario los ficheros YAML de variables y tareas, por si quisieras modificarlos para jugar con Ansible. Ten en cuenta que el siguiente comando abarca dos líneas.

```
:~$ cp /opt/cursoaws/ansible/*-adv* $HOME && sed s/00/$ID/g
/opt/cursoaws/ansible/advplaybook00.yml > $HOME/advplaybook.yml
```

Verificamos la ejecución del playbook anterior:

```
:~$ ansible-playbook advplaybook.yml
PLAY [tag_ansible_alucloud00] ************************************************

GATHERING FACTS ***************************************************************
ok: [ec2-54-89-81-208.compute-1.amazonaws.com]

TASK: [include_vars vars-advplaybook.yml] *************************************
ok: [ec2-54-89-81-208.compute-1.amazonaws.com]

TASK: [Copy BOTO's credential file to remote location] ************************
ok: [ec2-54-89-81-208.compute-1.amazonaws.com]

TASK: [Retrieving Jinja2 Templates from Amazon S3] ****************************
changed: [ec2-54-89-81-208.compute-1.amazonaws.com] => (item=boto.j2)

TASK: [Create user accounts] **************************************************
ok: [ec2-54-89-81-208.compute-1.amazonaws.com] => (item={'sk':
'ft0ftS7FD0H5L5Tu3m/Dg2DJoIb/GZ6niIWZbFeW', 'ak': 'AKIAJAIQMN42O7AGSC6A', 'name':
'user00', 'pw': 'QPeR2gsw4MZEk'})
ok: [ec2-54-89-81-208.compute-1.amazonaws.com] => (item={'sk':
'ssPvGujteBv0V/KRsJiY98RijNVo/Mg221VlLSBn', 'ak': 'AKIAIQZICAOA3T7AXXQP', 'name':
'user01', 'pw': 'V22YKgEokQv6o'})
ok: [ec2-54-89-81-208.compute-1.amazonaws.com] => (item={'sk':
'wOPwD1UeEBgaKvBf3wn5x9kh/6RTVB7amKGkm3ZQ', 'ak': 'AKIAI25CX6A55PKILAPQ', 'name':
'user02', 'pw': 'z0iVK2EJ1JOCZ'})

TASK: [Create BOTO credentials file] ******************************************
ok: [ec2-54-89-81-208.compute-1.amazonaws.com] => (item={'sk':
'ft0ftS7FD0H5L5Tu3m/Dg2DJoIb/GZ6niIWZbFeW', 'ak': 'AKIAJAIQMN42O7AGSC6A', 'name':
'user00', 'pw': 'QPeR2gsw4MZEk'})
ok: [ec2-54-89-81-208.compute-1.amazonaws.com] => (item={'sk':
'ssPvGujteBv0V/KRsJiY98RijNVo/Mg221VlLSBn', 'ak': 'AKIAIQZICAOA3T7AXXQP', 'name':
'user01', 'pw': 'V22YKgEokQv6o'})
ok: [ec2-54-89-81-208.compute-1.amazonaws.com] => (item={'sk':
'wOPwD1UeEBgaKvBf3wn5x9kh/6RTVB7amKGkm3ZQ', 'ak': 'AKIAI25CX6A55PKILAPQ', 'name':
'user02', 'pw': 'z0iVK2EJ1JOCZ'})

TASK: [Install PIP to bootstrap Boto (python-pip)] ****************************
ok: [ec2-54-89-81-208.compute-1.amazonaws.com]

TASK: [Install boto] **********************************************************
changed: [ec2-54-89-81-208.compute-1.amazonaws.com]

TASK: [Copy motd.tail] ********************************************************
ok: [ec2-54-89-81-208.compute-1.amazonaws.com]

PLAY RECAP ********************************************************************
ec2-54-89-81-208.compute-1.amazonaws.com : ok=9    changed=2    unreachable=0
failed=0
```

Es posible que obtengas algún warning indicando "*The input password appears not to have been hashed. The 'password' argument must be encrypted for this module to work properly*", pero puedes ignorarlo. Realizamos la conexión a la máquina recién configurada y verificamos que el playbook se ha ejecutado correctamente:

```
:~$ ssh -i $HOME/alucloud$ID-priv.pem ubuntu@ec2-54-89-81-208.compute-1.amazonaws.com
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-1024-aws x86_64)

 * Documentation:  https://help.ubuntu.com/

  System information as of Tue Jul 15 16:48:00 UTC 2014

  System load:  0.05              Processes:           63
  Usage of /:   13.4% of 7.87GB   Users logged in:     1
  Memory usage: 49%               IP address for eth0: 10.141.186.149
  Swap usage:   0%

  Graph this data and manage this system at
https://landscape.canonical.com/

  Get cloud support with Ubuntu Advantage Cloud Guest:
    http://www.ubuntu.com/business/services/cloud

  Use Juju to deploy your cloud instances and workloads:
    https://juju.ubuntu.com/#cloud-precise

*** /dev/xvda1 will be checked for errors at next reboot ***


     _ _      _
     /\              (_)  |    | |
    /  \    _ __  ___ _| |__  | | ___
   / /\ \ | '_ \/ __| | '_ \| |/ _ \
  / ____ \| | | \__ \ | |_) | |  __/
 /_/    \_\_| |_|___/_|_.__/|_|\___|

(c) ================================ (c)
Last login: Tue Jul 15 16:46:33 2014 from ip-10-152-140-20.ec2.internal
```

¡Enhorabuena! Has realizado la configuración automática de tus recursos de cómputo de forma automatizada mediante recetas de alto nivel. En este punto, te recomiendo que explores la documentación de Ansible por si quieres conocer más detalles de esta herramienta.

Terminación de Recursos

Una vez finalizadas las actividades de la práctica, es importante que liberes todos los recursos aprovisionados. Para ello, tienes a tu disposición un script llamado self-terminate-ansible-slaves.sh para que tú mismo elimines las instancias que has estado utilizando para configurar. Ejecuta dicho script. Deberás confirmar la eliminación pulsando la tecla `y`:

```
:~$ /opt/cursoaws/ansible/self-terminate-ansible-slaves.sh
Will terminate instances i-4715536c i-4415536f
Are you sure? [yY]y
{
```

```
    "TerminatingInstances": [
        {
            "InstanceId": "i-4715536c",
            "CurrentState": {
                "Code": 32,
                "Name": "shutting-down"
            },
            "PreviousState": {
                "Code": 16,
                "Name": "running"
            }
        },
        {
            "InstanceId": "i-4415536f",
            "CurrentState": {
                "Code": 32,
                "Name": "shutting-down"
            },
            "PreviousState": {
                "Code": 16,
                "Name": "running"
            }
        }
    ]
}
Instances succesfully terminated
```

Esto provocará la terminación de todas tus instancias. Por favor, asegúrate de haber eliminado las instancias antes de proseguir con la práctica. Evitemos gastos innecesarios.

Aprovisionamiento y Configuración de Infraestructuras

En las secciones anteriores has realizado la configuración automatizada de infraestructuras usando Ansible a partir de una infraestructura de (2) máquinas (virtuales) que previamente habías aprovisionado mediante un script, llamado `self-deploy-ansible-slaves.sh`, desde el entorno de prácticas. En realidad, dicho script utiliza también Ansible, pero esta vez como herramienta de despliegue o aprovisionamiento de infraestructura, no únicamente como herramienta de configuración de infraestructuras existentes.

El objetivo de esta sección es conocer las capacidades básicas de aprovisionamiento de máquinas virtuales ofrecidas por Ansible. Ansible permite realizar el despliegue de máquinas virtuales en diferentes proveedores Cloud públicos como es el caso de Microsoft Azure [45] (mediante el módulo azure [46]), AWS (mediante el módulo ec2 [46]) o Google Compute Engine (mediante el módulo gce [47]), así como en plataformas Cloud on-premise creadas mediante OpenStack (mediante el módulo nova_compute [49] o VMware vSphere (mediante el módulo vsphere_guest [49]). También puede desplegar máquinas virtuales directamente usando libvirt [50] (mediante el módulo virt [51]).

A continuación, se muestra una receta en YAML (también llamado playbook), apta para ser ejecutada desde Ansible, que permite realizar el aprovisionamiento y configuración de instancias de EC2 para desplegar de forma automatizada la aplicación Node Cellar [52] pero la versión sin backend de base de datos, cuyo código fuente está disponible para su descarga en [53]. El playbook está disponible en el entorno de realización de prácticas:

```
:~$ cat /opt/cursoaws/ansible/ansible-provision-ec2.yml
```

```
---
- name: Provision VMs
  hosts: all
  connection: local
  tags:
    - provision

  vars:
      instance_type: t2.micro
      image: ami-7dce6507 #Ubuntu 14.04 HVM
      subnet_id: subnet-2bfb6c4f #subnet-default-1a-public

  tasks:
    - name: Provisioning EC2 instance(s) (This might take a while ... DO NOT
INTERRUPT.)
      local_action: ec2
           keypair="alucloud{{ id }}-keypair"
           group="gs-aws-{{ id }}"
           instance_type={{ instance_type }}
           image={{ image }}
           wait=true
           count=1
           vpc_subnet_id={{subnet_id}}
           region="us-east-1"
           instance_tags='ansible=alucloud{{ id }}'
      register: ec2vms

    - name: Add new instance to host group
      local_action: add_host hostname={{ item.public_ip }} groupname=ec2hosts
      with_items: "{{ ec2vms.instances }}"

    - name: Waiting for the new instance(s) to be ready (This might take a while
... DO NOT INTERRUPT.)
      local_action: wait_for
         host={{item.public_dns_name}} port=22 delay=30 timeout=450 state=started
      with_items: "{{ ec2vms.instances }}"

    - name: Showing your instance(s) information
      debug: msg="Provisioned instance for Ansible {{ item.id }} --> {{
item.public_dns_name }}"
      with_items: "{{ ec2vms.instances }}"

####################
#
#  CONFIGURE VMs
#
####################
- name: Configure VM(s)
  hosts: ec2hosts
 # hosts: tag_ansible_alucloud00
  become: yes
  tags:
    - configuration
  pre_tasks:
    - name: Wait for cloud-init to finish
      wait_for: >
        path=/var/log/cloud-init.log
        timeout=15
        search_regex="final-message"

  tasks:
    - name: Install Apache Web Services with PHP support
      apt: name=apache2 update_cache=yes state=present
    - apt: name=php5 state=present
```

```
    - name: Retrieve Web Application from S3 bucket
      get_url: url=https://s3.amazonaws.com/cursoscloudupv/cloudformation/cellar-
webapp-mem.tgz dest=/tmp validate_certs=no owner=ubuntu group=ubuntu
    - name: Unpacking Web Application
      unarchive: src="/tmp/cellar-webapp-mem.tgz" dest="/var/www/html" copy=no
```

Esta receta (o playbook) tiene dos partes diferenciadas. En primer lugar, el aprovisionamiento de las máquinas virtuales (sección `-name: Provision VMs`) y, posteriormente, la configuración de las mismas (sección `-name: Configure VMs`). Analicemos con detalle cada sección de este documento YAML.

Aprovisionamiento de Instancias

El despliegue de instancias de EC2 se realiza mediante el módulo *ec2* [46], indicando el par de claves, el grupo de seguridad, el tipo de instancia, el identificador de la AMI, el número de instancias, la región y las etiquetas que se desea asignar a dichas instancias. Fíjate que los identificadores que están entre llaves (como por ejemplo `{{ id }}`) hacen referencias a variables, que pueden estar definidas tanto en la sección *vars*, como es el caso del identificador de la AMI (*image*), como pasarse como parámetros en la ejecución del playbook, como es el caso de `{{ id }}`.

Mediante `wait = true` se garantiza que el módulo ec2 no retorna el control hasta que las instancias no están en estado *running*. Usando *register* se almacena el resultado del comando en una variable registrada (*registered variable* [55]) de Ansible, que le hemos llamado *ec2vms*, y que contendrá referencias a las instancias desplegadas. Esta información será necesaria posteriormente para acceder a configurarlas. Mediante el módulo *add_host* [56] se consigue añadir las nuevas máquinas aprovisionadas a un fichero de inventario en memoria que mantiene Ansible, para luego referenciarlas durante la configuración. A continuación, se espera a que las instancias estén accesibles por SSH usando el módulo *wait_for* [57]. Recuerda que cuando una instancia pasa a estado *running* todavía es posible que el SO esté arrancando. Finalmente, se muestra un mensaje por pantalla indicando el nombre DNS de la instancia (o instancias) aprovisionadas, usando el módulo *debug* [57].

Fíjate que esta sección de YAML (llamada *Provision VMs*) se ejecuta sobre todos los hosts (*hosts: all*) definidos en el fichero de inventario. No obstante, veremos más adelante que ejecutaremos este playbook con un fichero de inventario que únicamente contendrá un host (*localhost*, es decir, 127.0.0.1). Ten en cuenta que se puede etiquetar (comando *tags*) una sección de un playbook para ejecutar únicamente parte de dicho playbok.

Veamos la sección del playbook destinada a configuración de las instancias

Configuración de Instancias

La configuración de instancias se lleva a cabo usando la sección *Configure VMs* del playbook. En ella se definen los comandos que se ejecutarán sobre todos los hosts incluidos en la variable *ec2hosts*. Recuerda que dicha variable fue rellenada con las instancias aprovisionadas anteriormente. Se realizará la conexión a las instancias como usuario Ubuntu con privilegios de superusuario (*become: yes*) usando la clave privada que se indicará como argumento en el momento de ejecutar el playbook.

En primer lugar, aparece una sección *pre_tasks* [61] que permite definir las tareas que se ejecutarán antes de las que se relacionan en la sección *tasks*. En concreto, se espera a que en el fichero de log de cloud-init [63] aparezca la palabra clave “final-message”. Esto indica que las tareas de configuración de la instancia han sido finalizadas. Es necesario este paso dado que se ha detectado [64] que instalar

paquetes mediante *apt* justo después de aprovisionar una instancia puede provocar un error que impida la instalación de los mismos.

Estas son las tareas que se realizarán:

1. Instalar el servidor web Apache con soporte para PHP
    a. Recuerda que es necesario incluir *update_cache = yes* en el módulo *apt* [59] para que se actualicen previamente las ubicaciones de los paquetes a actualizar de los repositorios de Ubuntu. De lo contrario, puedes tener problemas a la hora de localizar paquetes durante la instalación de los mismos.
2. Descargar la aplicación web desde un bucket de S3.
    a. Para ello se usa el módulo *get_url* [58].
3. Descomprimir la aplicación en la ruta destino del servidor web
    a. Para ello se usa el módulo *unarchive* [61], donde es necesario especificar (*copy = no*) dado que el fichero ya se encuentra en las instancias (de lo contrario trataría de copiarlo desde la máquina local, el entorno de prácticas, a las instancias de EC2).

Ejecución del PlayBook

Ejecutamos la receta de Ansible. Para ello, es necesario indicar el fichero de inventario que contiene las máquinas sobre las que se ejecutarán los comandos, el fichero de clave privada usado para conectarse a las instancias, así como los valores para las variables referenciadas en el playbook (en este caso, el identificador de alumnos, referenciando en el playbook mediante la variable *id* y cuyo valor está en la variable de entorno *$ID*).

Con respecto al fichero de inventario, dado que los comandos de aprovisionamiento se ejecutan en local y que los nombres de hosts a configurar se obtendrán dinámicamente al desplegar las instancias, únicamente es necesario indicar un fichero de inventario "*dummy*" que incluya la propia máquina local. En realidad, ni siquiera sería necesario indicar un fichero de inventario, pero el comando ansible-playbook espera recibir uno, por lo que nos vemos obligados a prepararle uno. Este es el contenido del fichero de inventario:

```
:~$ cat /opt/cursoaws/ansible/local-inventory
[localhost]
127.0.0.1
```

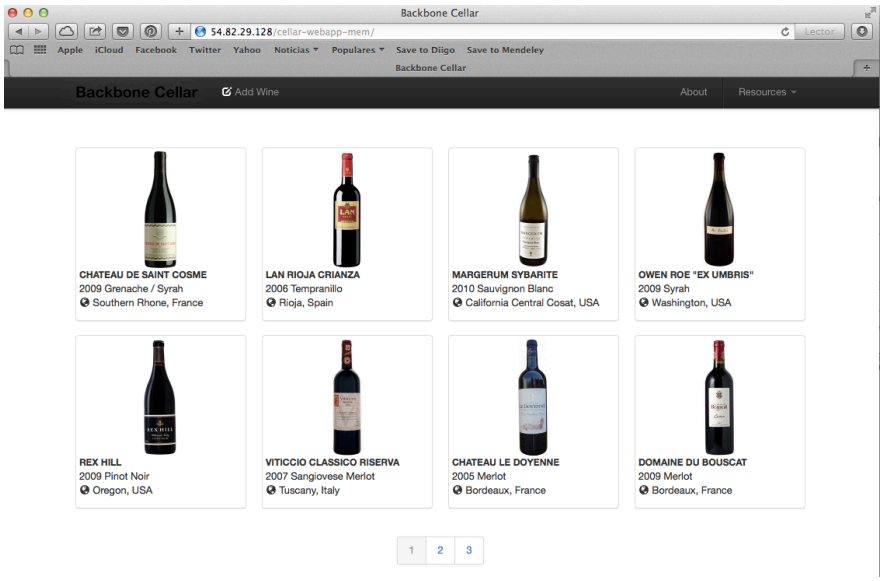
Ejecutamos el playbook usando el siguiente comando.

```
:~$ ansible-playbook -i /opt/cursoaws/ansible/local-inventory
--private-key $HOME/alucloud$ID-priv.pem --extra-vars "id=$ID"
/opt/cursoaws/ansible/ansible-provision-ec2.yml

PLAY [Provision VMs] ***********************************************************

GATHERING FACTS ****************************************************************
ok: [127.0.0.1]

TASK: [Provisioning EC2 instance(s) (This might take a while ... DO NOT
INTERRUPT.)] ***
changed: [127.0.0.1]

TASK: [Add new instance to host group] *****************************************
ok: [127.0.0.1] => (item={u'kernel': u'aki-919dcaf8', u'root_device_type': u'ebs',
u'private_dns_name': u'ip-10-230-19-28.ec2.internal', u'public_ip':
u'54.82.29.128', u'private_ip': u'10.230.19.28', u'id': u'i-ba348354', u'state':
```

```
u'running', u'virtualization_type': u'paravirtual', u'architecture': u'x86_64',
u'ramdisk': None, u'key_name': u'alucloud00-keypair', u'image_id': u'ami-3251905a',
u'public_dns_name': u'ec2-54-82-29-128.compute-1.amazonaws.com', u'state_code': 16,
u'placement': u'us-east-1e', u'ami_launch_index': u'0', u'dns_name': u'ec2-54-82-
29-128.compute-1.amazonaws.com', u'region': u'us-east-1', u'launch_time': u'2014-
09-23T13:04:49.000Z', u'instance_type': u't1.micro', u'root_device_name':
u'/dev/sda1', u'hypervisor': u'xen'})

TASK: [Waiting for the new instance(s) to be ready (This might take a while ... DO
NOT INTERRUPT.)] ***
ok: [127.0.0.1] => (item={u'kernel': u'aki-919dcaf8', u'root_device_type': u'ebs',
u'private_dns_name': u'ip-10-230-19-28.ec2.internal', u'public_ip':
u'54.82.29.128', u'private_ip': u'10.230.19.28', u'id': u'i-ba348354', u'state':
u'running', u'virtualization_type': u'paravirtual', u'architecture': u'x86_64',
u'ramdisk': None, u'key_name': u'alucloud00-keypair', u'image_id': u'ami-3251905a',
u'public_dns_name': u'ec2-54-82-29-128.compute-1.amazonaws.com', u'state_code': 16,
u'placement': u'us-east-1e', u'ami_launch_index': u'0', u'dns_name': u'ec2-54-82-
29-128.compute-1.amazonaws.com', u'region': u'us-east-1', u'launch_time': u'2014-
09-23T13:04:49.000Z', u'instance_type': u't1.micro', u'root_device_name':
u'/dev/sda1', u'hypervisor': u'xen'})

TASK: [Showing your instance(s) information] ********************************
ok: [127.0.0.1] => (item={u'kernel': u'aki-919dcaf8', u'root_device_type': u'ebs',
u'private_dns_name': u'ip-10-230-19-28.ec2.internal', u'public_ip':
u'54.82.29.128', u'private_ip': u'10.230.19.28', u'id': u'i-ba348354', u'state':
u'running', u'virtualization_type': u'paravirtual', u'architecture': u'x86_64',
u'ramdisk': None, u'key_name': u'alucloud00-keypair', u'image_id': u'ami-3251905a',
u'public_dns_name': u'ec2-54-82-29-128.compute-1.amazonaws.com', u'state_code': 16,
u'placement': u'us-east-1e', u'ami_launch_index': u'0', u'dns_name': u'ec2-54-82-
29-128.compute-1.amazonaws.com', u'region': u'us-east-1', u'launch_time': u'2014-
09-23T13:04:49.000Z', u'instance_type': u't1.micro', u'root_device_name':
u'/dev/sda1', u'hypervisor': u'xen'}) => {
    "item": {
        "ami_launch_index": "0",
        "architecture": "x86_64",
        "dns_name": "ec2-54-82-29-128.compute-1.amazonaws.com",
        "hypervisor": "xen",
        "id": "i-ba348354",
        "image_id": "ami-3251905a",
        "instance_type": "t1.micro",
        "kernel": "aki-919dcaf8",
        "key_name": "alucloud00-keypair",
        "launch_time": "2014-09-23T13:04:49.000Z",
        "placement": "us-east-1e",
        "private_dns_name": "ip-10-230-19-28.ec2.internal",
        "private_ip": "10.230.19.28",
        "public_dns_name": "ec2-54-82-29-128.compute-1.amazonaws.com",
        "public_ip": "54.82.29.128",
        "ramdisk": null,
        "region": "us-east-1",
        "root_device_name": "/dev/sda1",
        "root_device_type": "ebs",
        "state": "running",
        "state_code": 16,
        "virtualization_type": "paravirtual"
    },
    "msg": "Provisioned instance for Ansible i-ba348354 --> ec2-54-82-29-
128.compute-1.amazonaws.com"
}

PLAY [Configure VM(s)] ******************************************************

GATHERING FACTS *************************************************************
ok: [54.82.29.128]
```

```
TASK: [Install Apache Web Services with PHP support] *************************
changed: [54.82.29.128]

TASK: [apt name=php5 state=present] ******************************************
changed: [54.82.29.128]

TASK: [Retrieve Web Application from S3 bucket] ******************************
changed: [54.82.29.128]

TASK: [Unpacking Web Application] ********************************************
changed: [54.82.29.128]

PLAY RECAP ********************************************************************
127.0.0.1                  : ok=5    changed=1    unreachable=0    failed=0
54.82.29.128               : ok=5    changed=4    unreachable=0    failed=0
```

Una vez desplegada y configurada la instancia de forma automática, tan solo queda conectarse mediante un navegador web para verificar que la aplicación funciona correctamente. Tendrás que conectarte a la dirección (recuerda sustituir la IP por la correspondiente a tu instancia): http://54.82.29.128/cellar-webapp-mem/



¡Enhorabuena! Has utilizado Ansible como herramienta para el aprovisionamiento y la configuración automática de infraestructuras virtuales sobre un Cloud como el de AWS.

Ten en cuenta que dado que se han etiquetado las instancias al desplegarlas mediante Ansible (en este caso con la etiqueta ansible / alucloud00), es posible usar el inventario dinámico (a través del siguiente comando, como ya viste al principio de la práctica) para obtener un listado de dichas instancias:

```
:~$ ansible-inventory -i /opt/cursoaws/ansible/inventory_aws_ec2.yml --graph
…
|--@tag_ansible_alucloud00:
  |  |--ec2-18-213-111-144.compute-1.amazonaws.com
  |  |--ec2-3-210-205-102.compute-1.amazonaws.com
…
```

De hecho, es perfectamente posible ejecutar únicamente una parte del PlayBook (por ejemplo, la parte de configuración, etiquetada con la etiqueta (tag) *configuration*) sobre el conjunto de hosts etiquetados como ansible_alucloud$ID modificando el apartado *hosts* de dicho apartado para que, en lugar de referenciar a los hosts ec2hosts se referencie a tag_ansible_alucloudOO (sustituye OO por tu identificador de usuario). Tendrás que copiar el playbook a tu cuenta de usuario y realizar dicha modificación (se puede hacer fácilmente con *sed* como se muestra a continuación). De esta manera, el playbook quedaría:

```
:~$ sed s/ec2hosts/tag_ansible_alucloud$ID/g /opt/cursoaws/ansible/ansible-
provision-ec2.yml > my-ansible-provision-ec2.yml

:~$ ansible-playbook -i /opt/cursoaws/ansible/inventory_aws_ec2.yml --private-key
$HOME/alucloud$ID-priv.pem --extra-vars "id=$ID" --tag configuration  my-ansible-
provision-ec2.yml
PLAY [Configure VM(s)] ***************************************************

GATHERING FACTS ***************************************************************
ok: [ec2-54-82-29-128.compute-1.amazonaws.com]

TASK: [Install Apache Web Services with PHP support] *************************
ok: [ec2-54-82-29-128.compute-1.amazonaws.com]

TASK: [apt name=php5 state=present] ******************************************
ok: [ec2-54-82-29-128.compute-1.amazonaws.com]

TASK: [Retrieve Web Application from S3 bucket] ******************************
ok: [ec2-54-82-29-128.compute-1.amazonaws.com]

TASK: [Unpacking Web Application] ********************************************
ok: [ec2-54-82-29-128.compute-1.amazonaws.com]

PLAY RECAP ********************************************************************
ec2-54-82-29-128.compute-1.amazonaws.com : ok=5    changed=0    unreachable=0
failed=0
```

En este caso, al ejecutar la fase de configuración sobre un grupo de instancias que ya ha sido configurada, mantienen exactamente la misma configuración (dado que el playbook de Ansible es idempotente). Sin embargo, es importante conocer las ventajas de tener un inventario dinámico, de manera que puedas aprovisionar un conjunto de recursos (con Ansible y otras herramientas) y luego aplicar configuraciones a grupos de instancias en función de las etiquetas asignadas. Esto permite tener instancias etiquetadas en función del rol que pueden desempeñar dentro de una arquitectura y utilizar Ansible para inyectar dicha configuración y hacer que una instancia adopte un determinado rol.

Una vez hayas terminado de jugar con Ansible, no te olvides de terminar todos los recursos. Recuerda que, como tus instancias están etiquetadas, puedes hacerlo fácilmente de la siguiente manera:

```
:~$ self-terminate-ansible-slaves.sh
Will terminate instances i-ba348354
Are you sure? [yY] y
{
    "TerminatingInstances": [
        {
            "InstanceId": "i-ba348354",
```

```
            "CurrentState": {
                "Code": 32,
                "Name": "shutting-down"
            },
            "PreviousState": {
                "Code": 16,
                "Name": "running"
            }
        }
    ]
}
Instances succesfully terminated
```

Conclusiones

Ansible permite simplificar la ejecución remota de comandos sobre máquinas remotas, así como la definición de recetas que facilitan la configuración desatendida de máquinas. En un entorno de tipo Cloud, esto posibilita la creación de recetas de configuración (*playbooks*) que permiten definir roles para las máquinas que componen la arquitectura de una aplicación Cloud.

Esto permite que cuando una instancia de máquinas virtual sea aprovisionada sobre una infraestructura Cloud, esta sea automáticamente configurada para desempeñar el rol para el que fue creada. Por ello, las herramientas de *DevOps* como Ansible ayudan en la configuración automática de aplicaciones Cloud.

Referencias

[1] Golden Image. https://searchservervirtualization.techtarget.com/definition/golden-image
[2] DevOps. https://en.wikipedia.org/wiki/DevOps
[3] Ansible Works. https://www.ansible.com/
[4] Puppet. https://puppetlabs.com
[5] Chef. https://www.chef.io
[6] Rex. https://rexify.org
[7] SaltStack. https://saltstack.com/
[8] Capistrano. https://github.com/capistrano/capistrano
[9] CFEngine. https://cfengine.com
[10] Juju. https://juju.ubuntu.com
[11] Inventory. https://docs.ansible.com/ansible/latest/inventory_guide/intro_inventory.html
[12] Ansible Modules. https://docs.ansible.com/ansible/latest/module_plugin_guide/index.html
[13] JavaScript Object Notation (JSON). https://www.json.org
[14] Fortune. https://en.wikipedia.org/wiki/Fortune_(Unix)
[15] Apache. https://httpd.apache.org
[16] PlayBooks. https://docs.ansible.com/ansible/latest/playbook_guide/playbooks.html
[17] YAML. https://www.yaml.org
[18] Apache Tomcat. https://tomcat.apache.org
[19] Sample Application for Apache Tomcat. https://tomcat.apache.org/tomcat-7.0-doc/appdev/sample/
[20] Apache Tomcat documentation. https://tomcat.apache.org/tomcat-7.0-doc/
[21] Ansible Playbooks. https://docs.ansible.com/ansible/latest/playbook_guide/playbooks.html

[22] Tagging Your Amazon EC2 Resources. https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/Using_Tags.html
[23] Infrastructure Manager (IM): https://www.grycap.upv.es/im/
[24] Inventory. https://docs.ansible.com/ansible/latest/inventory_guide/intro_inventory.html
[25] Dynamic Inventory. https://docs.ansible.com/ansible/latest/inventory_guide/intro_dynamic_inventory.html
[26] Inventory Plugins. https://docs.ansible.com/ansible/latest/plugins/inventory.html#inventory-plugins
[27] JSON. https://es.wikipedia.org/wiki/JSON
[28] The Ansible Configuration File. https://docs.ansible.com/ansible/latest/installation_guide/intro_configuration.html
[29] remote_user. https://docs.ansible.com/ansible/latest/inventory_guide/connection_details.html#setting-a-remote-user
[30] passwd. https://es.wikipedia.org/wiki/Passwd
[31] apt- Manage apt-packages. https://docs.ansible.com/ansible/latest/collections/ansible/builtin/apt_module.html
[32] C3. https://www.csm.ornl.gov/srt/c3/
[33] Service – Manage services. https://docs.ansible.com/ansible/latest/collections/ansible/builtin/service_module.html
[34] Indentación. https://es.wikipedia.org/wiki/Indentación
[35] Amazon RDS (Relational Database Service). https://aws.amazon.com/rds
[36] Implementaciones en zonas de disponibilidad múltiples (Multi-AZ) de Amazon RDS. https://aws.amazon.com/es/rds/multi-az/
[37] Chapter 17. Replication. https://dev.mysql.com/doc/refman/5.6/en/replication.html
[38] S3 module. https://docs.ansible.com/ansible/latest/collections/amazon/aws/aws_s3_module.html
[39] Mysql_db module. https://docs.ansible.com/ansible/latest/collections/community/mysql/mysql_db_module.html
[40] Boto. https://boto3.amazonaws.com/v1/documentation/api/latest/index.html
[41] template – Templates a file out to a remote server. https://docs.ansible.com/ansible/latest/collections/ansible/builtin/template_module.html
[42] Jinja2. https://jinja.palletsprojects.com/en/3.1.x/
[43] include_vars. https://docs.ansible.com/ansible/latest/collections/ansible/builtin/include_vars_module.html
[44] Task Include Files and Encouraging Reuse. https://docs.ansible.com/ansible/latest/collections/ansible/builtin/include_tasks_module.html
[45] Microsoft Azure. https://azure.microsoft.com/es-es/
[46] azure module. https://docs.ansible.com/ansible/latest/scenario_guides/guide_azure.html
[47] ec2 module. https://docs.ansible.com/ansible/latest/collections/amazon/aws/ec2_instance_module.html
[48] gce module. https://docs.ansible.com/ansible/latest/scenario_guides/guide_gce.html
[49] nova_compute module. https://docs.ansible.com/ansible/latest/collections/openstack/cloud/compute_service_info_module.html
[50] vsphere_guest module. https://docs.ansible.com/ansible/latest/collections/community/vmware/vmware_guest_info_module.html
[51] libvirt. https://libvirt.org

[52] virt module. https://docs.ansible.com/ansible/latest/collections/community/libvirt/virt_module.html
[53] Node Cellar. https://github.com/ccoenraets
[54] Cellar Web App. https://s3.amazonaws.com/cursoscloudupv/cloudformation/cellar-webapp-mem.tgz
[55] Registered Variables. https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_variables.html#registering-variables
[56] add_host module. https://docs.ansible.com/ansible/latest/collections/ansible/builtin/add_host_module.html
[57] wait_for module. https://docs.ansible.com/ansible/latest/collections/ansible/builtin/wait_for_module.html
[58] debug module. https://docs.ansible.com/ansible/latest/collections/ansible/builtin/debug_module.html
[59] get_url module. https://docs.ansible.com/ansible/latest/collections/ansible/builtin/get_url_module.html
[60] apt module. https://docs.ansible.com/ansible/latest/collections/ansible/builtin/apt_module.html
[61] unarchive. https://docs.ansible.com/ansible/latest/collections/ansible/builtin/unarchive_module.html
[62] Playbook Roles and Include Statements. https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_reuse_roles.html
[63] CloudInit. https://help.ubuntu.com/community/CloudInit
[64] Running apt commands immediately after EC2 provisioning in AWS can be problematic. https://groups.google.com/g/ansible-project/c/u4tW6pqU-VM
[65] Become. https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_privilege_escalation.html

Anexo

Anexo A. Configuración de la Máquina Principal y de las Máquinas a Configurar

A continuación se describe el proceso que deberías seguir para configurar tanto la máquina principal (MP) como las máquinas que pretendes configurar (MC). Esta configuración ya ha sido aplicada en el entorno para la realización de las prácticas por lo que **NO DEBES EJECUTAR ESTOS COMANDOS EN EL ENTORNO DE PRÁCTICAS**. Si quieres realizar la práctica salta directamente a la siguiente sección. Esta información tan solo se te entrega para que entiendas el funcionamiento de Ansible y por si quisieras utilizar la herramienta en tus propios proyectos.

1. Creación de par de claves RSA

Desde la MP, conectado con tu usuario, deberás verificar si ya tienes un par de claves RSA creado. Si existen los ficheros $HOME/.ssh/id_rsa (clave privada) y $HOME/.ssh/id_rsa.pub (clave pública), entonces puedes utilizar este par de claves (pasa al paso siguiente). De lo contrario, deberás crear un par de claves RSA con el comando `ssh-keygen`. Pulsa Enter en las preguntas que te plantea, para elegir las opciones por defecto (fichero en el que se guarda la clave privada y contraseña vacía).

```
alucloud00@precise32:~$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/alucloud00/.ssh/id_rsa):
Created directory '/home/alucloud00/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/alucloud00/.ssh/id_rsa.
Your public key has been saved in /home/alucloud00/.ssh/id_rsa.pub.
The key fingerprint is:
f3:99:c9:41:d1:0e:a6:56:87:b8:96:78:9d:d4:ea:51 alucloud00@precise32
The key's randomart image is:
+--[ RSA 2048]----+
|          ..+    |
|         . *.E   |
|        . O.B    |
|       . B.= .   |
|        +S...    |
|          +.=    |
|           *     |
|                 |
|                 |
+-----------------+
```

Esto habrá creado los ficheros $HOME/id_rsa y $HOME/id_rsa.pub.

2. Autorizar conexión por SSH sin contraseña en las máquinas a configurar

Para conectarnos mediante SSH, sin tener que indicar la contraseña, desde una cuenta de usuario de una máquina (A) a una cuenta de usuario de otra máquina (B) procedemos de la siguiente manera. Asumimos que el usuario en (A) es *alucloudoo* y que el usuario de la máquina (B), cuya IP es es 54.226.194.39, es *ubuntu*.

a) Copiar la clave pública de (A) a la máquina (B)

```
alucloud00@precise32:~/.ssh$ scp $HOME/.ssh/id_rsa.pub
ubuntu@54.226.194.39:.
The authenticity of host '54.226.194.39 (54.226.194.39)' can't be
established.
ECDSA key fingerprint is
c5:6c:7d:d2:40:c6:26:b8:01:4c:4a:dc:29:2b:71:ac.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '54.226.194.39' (ECDSA) to the list of
known hosts.
ubuntu@54.226.194.39's password:
id_rsa.pub                                            100%  402
0.4KB/s   00:00
```

Deberás introducir la contraseña del usuario ubuntu en (B) para realizar la copia.

b) Introducir el contenido de la clave pública en el fichero $HOME/.ssh/authorized_keys de (B)

```
alucloud00@precise32:~/.ssh$ ssh ubuntu@54.226.194.39 "cat
~/id_rsa.pub >> ~/.ssh/authorized_keys"
```

Fíjate que se está ejecutando un comando remoto vía SSH para introducir el contenido del fichero de clave pública en el fichero $HOME/.ssh/authorized_keys de la máquina (B). Si vas a copiar este comando en un terminal, ten en cuenta que la línea está partida en dos. Deberás introducir la contraseña para ejecutar este comando.

c) Verifica que puedes conectarte via SSH sin contraseña desde (A) a (B)

```
alucloud00@precise32:~/.ssh$ ssh ubuntu@54.226.194.39 "/bin/ls"
examples.desktop
id_rsa.pub
```

Fíjate que se está ejecutando el comando /bin/ls en la máquina (B) a petición de la máquina (A). Esto ya no debería pedirte la contraseña. A partir de este punto, ya puedes conectarte a la máquina (B) y ejecutar comandos remotos mediante SSH sin necesidad de especificar la contraseña. Si por el contrario te sigue pidiendo la contraseña repasa los pasos anteriores.