

4.4

Aprovisionamiento de Servidores : Ansible

Aprovisionamiento de Servidores : Ansible

- Aprovisionamiento de Servidores : Ansible



Configuration management services

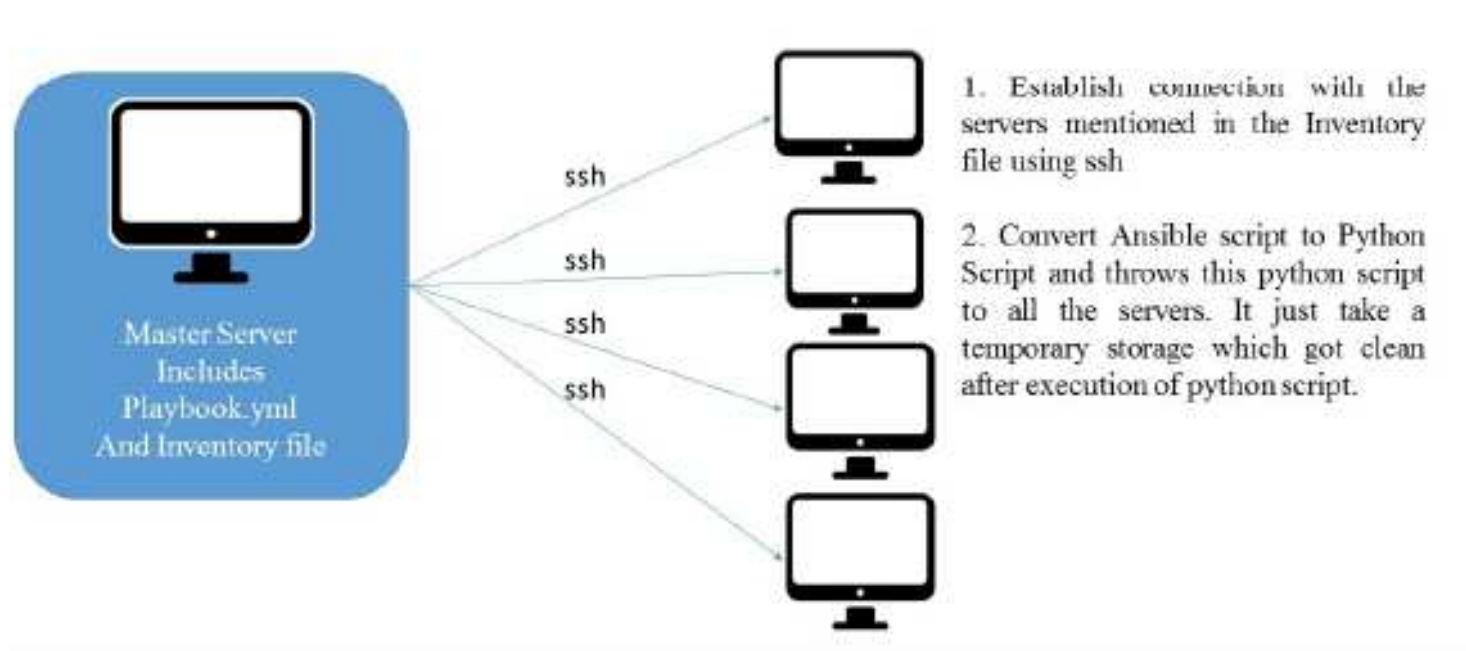
- AWS OpsWorks Stacks
- AWS OpsWorks for Chef Automate
- AWS OpsWorks for Puppet Enterprise



Aprovisionamiento de Servidores : Ansible

- Es una herramienta para gestión de configuración de servidores/recursos
 - Aprovisionamiento
 - Gestión de la configuración
 - Despliegue de aplicaciones
 - Entrega continua
 - Orquestación
- Aprovisionamiento de Servidores : Ansible
 - La configuración del servidor se da después del arranque del servidor
- Utiliza fichero YAML que describe las operaciones a ejecutar
- Idempotente

Aprovisionamiento de Servidores : Ansible



Aprovisionamiento de Servidores : Ansible

- Características principales
 - Agentless: No requiere instalar ningún agente en el objetivo
 - SSH y Python:
 - Ansible se comunica con el servidor destino mediante SSH.
 - Convierte los “playbook” en scripts de Python y los envía al servidor destino y los ejecuta
 - YAML
 - Estructura flexible y multifichero
 - Encriptación
 - Estrategia basada en push
 - <https://docs.ansible.com/>
 - <https://galaxy.ansible.com/>

Aprovisionamiento de Servidores :

Ansible

- Playbooks: Permite describir las tareas a realizar en un servidor
- Inventory: Permite definir la lista de ip sobre las que actuar
- Roles: Permite dividir las operaciones en diferentes carpetas y directorios
 - Reutilización
- Variables/Parametrización

Aprovisionamiento de Servidores : Ansible

Ejercicio 1 : Instalación de 2 Servidores web

- Playbooks, Inventory y variables

Ejercicio 2 : Instalación de Wordpress

- Roles

Ejercicio 3 : Actualización de Servidores AWS EC2

- Inventory Dinámico

Ejercicio 4 : Instalación de Apache en servidores AWS EC2

- Roles, Inventory dinámico y variables

Ejercicio 5 : Creación de arquitectura HA en AWS

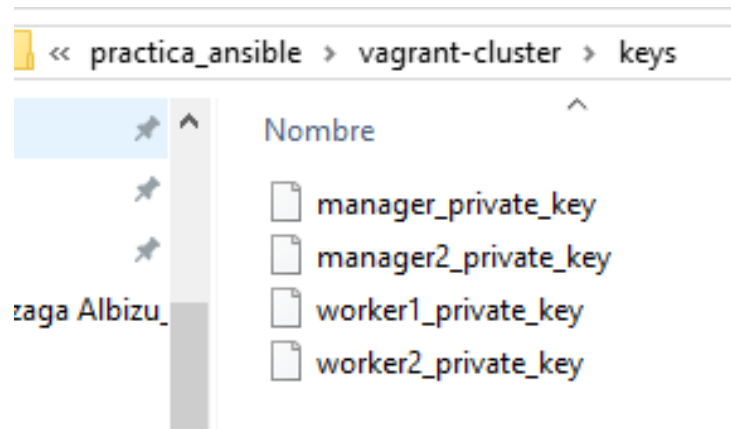
Aprovisionamiento de Servidores :

Ansible

- En los ejercicios de Ansible se utilizará la configuración 'Cluster' de vagrant del repositorio
- Vagrantfile: IP máquinas:
 - Manager: 192.168.100.9
 - Manager2: 192.168.100.10
 - worker1: 192.168.100.11
 - worker2: 192.168.100.12
- El servidor Manager se utiliza para ejecutar Ansible y aprovisionar las tres máquinas que configurarán el cluster
- En algunos ejercicios las máquinas Vagrant se sustituirán por máquinas EC2
- Antes de empezar probar conectividad desde el manager a los otros servidores vía SSH
 - Copiar las claves privadas de acceso a la carpeta de vagrant
- Instalar en manager ansible mediante apt
- Comprobar que los servidores a aprovisionar disponen de Python

Aprovisionamiento de Servidores : Ansible

- Comprobación del entorno de trabajo
 - \$vagrant up
 - \$vagrant ssh manager
 - Copiar las claves privadas al manager al directorio de trabajo
 - vagrant\$ssh -i worker1_private_key vagrant@192.168.100.11
 - Permisos mínimos en las claves
 - chmod 700



```
vagrant@manager:~$ ssh -i .keys/worker1_private_key vagrant@192.168.100.11
Welcome to Ubuntu 16.04.6 LTS (GNU/Linux 4.4.0-146-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

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

69 packages can be updated.
0 updates are security updates.

New release '18.04.2 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

*** System restart required ***
Last login: Fri Oct 11 12:11:20 2019 from 192.168.100.1
vagrant@worker1:~$
```

```
vagrant@manager:~/aws_ansible/ejercicio1$ cp -r /vagrant/keys/* ./
vagrant@manager:~/aws_ansible/ejercicio1$ ls
ansible.cfg          hosts                manager_private_key  worker2_private_key
apache_playbook.yml  manager2_private_key worker1_private_key
```

Aprovisionamiento de Servidores : Ansible

Ejercicio 1 : Instalación de 2 Servidores web

- Playbooks (tasks y handlers), Inventory y variables
 1. Se trabajara desde el servidor MANAGER
 2. Primero configuremos el fichero que indica sobre que servidores queremos actuar (INVENTORY) indicándole el usuario ssh a utilizar por defecto
 3. Definiremos las tareas que queremos ejecutar , en este caso la instalación de Apache y la creación de un fichero de bienvenida
 4. Para este ejercicio se utilizarán los siguientes ficheros

```
└─ ejercicio_1
  └─ ansible.cfg
  └─ ! apache_playbook.yml
  └─ ≡ hosts
```

Aprovisionamiento de Servidores :

Ansible

Ejercicio 1 : Instalación de 2 Servidores web

Definición de los hosts

- Crear el fichero 'hosts'
- En este caso definimos dos servidores dentro del grupo webservers
 - El primero con nombre e ip
 - El segundo solo con IP

```
[webservers]
worker1 ansible_host=192.168.100.11 ansible_ssh_private_key_file=./worker1_private_key
192.168.100.12 ansible_ssh_private_key_file=./worker2_private_key
```

- Se puede configurar la conexión SSH para cada servidor

```
alpha.example.com ansible_user=bob ansible_port=50022
bravo.example.com ansible_user=mary
ansible_ssh_private_key_file=/path/to/mary.key
frontend.example.com ansible_port=50022
yellow.example.com ansible_host=192.168.33.10
```

Aprovisionamiento de Servidores : Ansible

Ejercicio 1 : Instalación de 2 Servidores web

Configuración global

- El fichero de configuración de ansible esta en
 - /etc/ansible/ansible.cfg
- Pero se puede sobrescribir creando un fichero ansible.cfg en el directorio de ejecución
- En nuestro caso definiremos el usuario ssh a utilizar en todas las conexiones por defecto

```
[defaults]  
ansible_user = vagrant
```

Aprovisionamiento de Servidores : Ansible

Ejercicio 1 : Instalación de 2 Servidores web

Definición de las tareas (PLAYBOOK)

- Apache_playbook.yml
- Tarea: Tasks: Instalar apache en los servidores workers

```
---
#PLAYBOOK PARA LA INSTALACION DE APACHE
# Y UNA PAGINA DE BIENVENIDA
- name: Instalacion de apache2
  #Primero indicamos sobre que grupo de
  #servidores vamos a actuar
hosts: webservers
  # Ahora se configuran ciertas variables preestablecidas
  # Permítimos ejecutar como sudo en el servidor remoto
sudo: yes
  #Creamos una variable de parametrización
vars:
    texto: kaixo
  #y ahora las tareas que queremos ejecutar
```

tasks:

```
#Para ejecutar las tareas se utilizan modulos
#En este caso utilizaremos el modulo command
- name : Ejecucion de un simple comando
  command: /bin/echo hello k!
#Tarea para instalar apache mediante apt
- name: install apache2
  apt: name=apache2 update_cache=yes state=latest
```

Aprovisionamiento de Servidores : Ansible

Ejercicio 1 : Instalación de 2 Servidores web

Definición de las tareas (PLAYBOOK)

- Ejecutar playbook y comprobar la instalación

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

- Comprobar vía curl/navegador: `curl 192.168.100.11`

```
vagrant@manager:~/aws_ansible/ejercicio1$ cp -r /vagrant/keys/* ./
vagrant@manager:~/aws_ansible/ejercicio1$ ls
ansible.cfg      hosts            manager_private_key  worker2_private_key
apache_playbook.yml  manager2_private_key  worker1_private_key

vagrant@manager:~/aws_ansible/ejercicio1$ ansible-playbook -i hosts apache_playbook.yml
[DEPRECATION WARNING]: Instead of sudo/sudo_user, use become/become_user and make sure b
(default). This feature will be removed in version 2.9. Deprecation warnings can be disa
deprecation_warnings=False in ansible.cfg.

PLAY [Instalacion de apache2] *****

TASK [Gathering Facts] *****
ok: [worker1]
ok: [192.168.100.12]

TASK [Ejecucion de un simple comando] *****
changed: [worker1]
changed: [192.168.100.12]

TASK [install apache2] *****
[WARNING]: Could not find aptitude. Using apt-get instead
ok: [192.168.100.12]
ok: [worker1]

PLAY RECAP *****
192.168.100.12      : ok=3    changed=1    unreachable=0    failed=0    skipped=0
worker1            : ok=3    changed=1    unreachable=0    failed=0    skipped=0
```

```
vagrant@manager:~/aws_ansible/ejercicio1$ curl 192.168.100.11 | more
% Total    % Received % Xferd  Average Speed   Time    Time     Time
          Dload  Upload   Total     Spent    Left
0         0     0      0      0      0      0  --:--:-- --:--:-- --:--:--
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://
">
<html xmlns="http://www.w3.org/1999/xhtml">
<!--
  Modified from the Debian original for Ubuntu
  Last updated: 2014-03-19
  See: https://launchpad.net/bugs/1288690
-->
<head>
```



Aprovisionamiento de Servidores : Ansible

Ejercicio 1 : Instalación de 2 Servidores web

Modificación de tareas (PLAYBOOK)

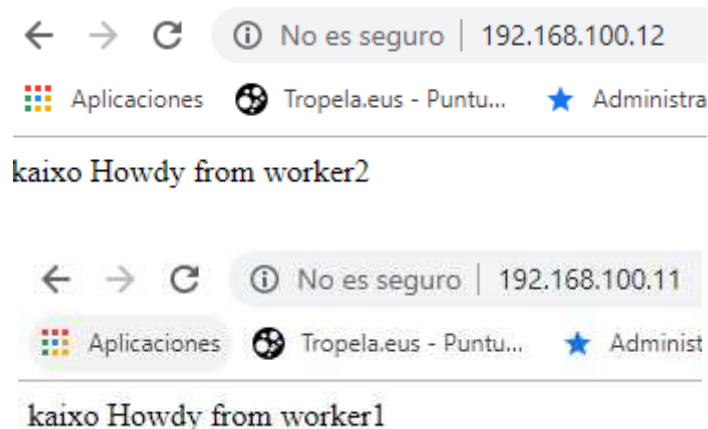
- Creación de una página de inicio
- Reinicio de servidor web
- Y uso de variables en las tareas *{{ variable }}*
 - name: Sortu ongi etorri index.html
 - copy:**
 - dest: /var/www/html/index.html
 - content: "**{{ texto }}** Howdy from **{{ ansible_hostname }}**"
 - name: restart apache2
 - service:** name=apache2 state=restarted

Aprovisionamiento de Servidores : Ansible

Ejercicio 1 : Instalación de 2 Servidores web

Modificación de tareas (PLAYBOOK)

- Comprobación



```
vagrant@manager:~/aws_ansible/ejercicio1$ ansible-playbook -i hosts apache_playbook_1.yml
[DEPRECATION WARNING]: Instead of sudo/sudo_user, use become/become_user and make sure be
(default). This feature will be removed in version 2.9. Deprecation warnings can be disab
deprecation_warnings=False in ansible.cfg.

PLAY [Instalacion de apache2] *****

TASK [Gathering Facts] *****
ok: [192.168.100.12]
ok: [worker1]

TASK [Ejecucion de un simple comando] *****
changed: [192.168.100.12]
changed: [worker1]

TASK [install apache2] *****
[WARNING]: Could not find aptitude. Using apt-get instead
ok: [192.168.100.12]
ok: [worker1]

TASK [Sortu ongi etorri index.html] *****
changed: [worker1]
changed: [192.168.100.12]

TASK [restart apache2] *****
changed: [worker1]
changed: [192.168.100.12]

PLAY RECAP *****
192.168.100.12      : ok=5    changed=3    unreachable=0    failed=0    skipped=0
worker1           : ok=5    changed=3    unreachable=0    failed=0    skipped=0

vagrant@manager:~/aws_ansible/ejercicio1$ curl 192.168.100.11
kaixo Howdy from worker1vagrant@manager:~/aws_ansible/ejercicio1$ curl 192.168.100.12
kaixo Howdy from worker2vagrant@manager:~/aws_ansible/ejercicio1$
```


Aprovisionamiento de Servidores :

Ansible

- **Playbooks**
- Definición de tareas
- Existen diferentes módulos con diferentes tareas

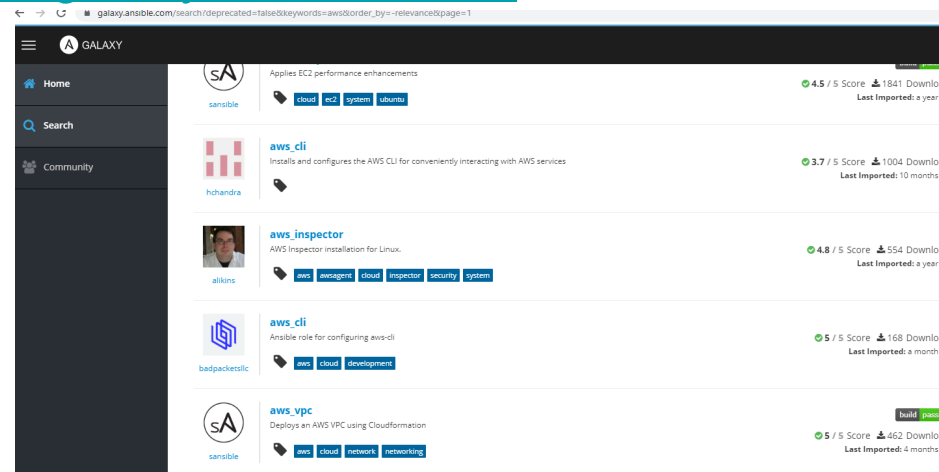
```
ansible-playbook -i <inventory_file> provisioning/playbook.yml
```

```
- hosts: webserver
sudo: yes
remote_user: ec2-user
tasks:
  - name: Updating System
    yum: name=* state=latest
```

```
- hosts: all
tasks:
  - name: Make sure that we can connect to the machine
    ping:
  - name: Install PHP
    apt: name=php5-cli state=present update_cache=yes
```

Aprovisionamiento de Servidores : Ansible

- Playbooks
- Existen infinitad de módulos
- https://docs.ansible.com/ansible/latest/modules/list_of_all_modules.html
- Modulo: apt
https://docs.ansible.com/ansible/latest/modules/apt_module.html#apt-module
- Existen playbooks ya creados
- <https://galaxy.ansible.com/>



Aprovisionamiento de Servidores :

Ansible

- Playbooks

```
---  
- hosts: all  
become: true  
tasks:  
- name: Make sure that we can connect to the machine  
ping:  
- name: Install PHP  
apt: name=php5-cli state=present update_cache=yes  
- name: Install nginx  
apt: name=nginx state=present  
- name: Install mySQL  
apt: name=mysql-server-5.6 state=present
```

```
---  
- hosts: all  
become: true  
tasks:  
- name: Install required packages  
apt: name={{item}} state=present update_cache=yes  
with_items:  
- php5-cli  
- nginx  
- mysql-server-5.6
```

Aprovisionamiento de Servidores :

Ansible

- Inventory:
 - Lista de ips y configuraciones de acceso
 - Configuración de conexión
 - ansible_host
 - ansible_user
 - ansible_port
 - ansible_ssh_private_key_file
 - Ansible_ssh_pass
 - Definición de variables de entorno

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

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

```
bravo.example.com ansible_user=mary
```

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

```
frontend.example.com ansible_port=50022
```

```
yellow.example.com ansible_host=192.168.33.10
```

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

Aprovisionamiento de Servidores :

Ansible

- Inventory:
 - Variables de entorno

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

```
$ cat staging-inventory  
alpha.example.com database_name=staging_db
```

```
$ cat production-inventory  
alpha.example.com database_name=prod
```

Aprovisionamiento de Servidores :

Ansible

- Inventory:
 - Grupos de inventario

```
[web]
```

```
host1.example.com
```

```
host2.example.com
```

```
[database]
```

```
db.example.com
```

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

```
- hosts: web
```

```
tasks:
```

```
- ping:
```

Aprovisionamiento de Servidores :

Ansible

- Inventory:
 - Variables de Grupos de inventario
 - Grupos de grupos

```
[web:vars]  
apache_version=2.4  
engage_flibbit=true
```

```
[web_centos5]  
host1.example.com  
host2.example.com
```

```
[centos5:children]  
web_centos5  
database_centos5
```

```
[centos5:vars]  
apache_version=2.2
```

```
[web_centos6]  
shinynewthing.example.com
```

```
[centos6:children]  
web_centos6  
reporting_centos6
```

```
[centos6:vars]  
apache_version=2.4
```

```
[database_centos5]  
database.example.com
```

```
[reporting_centos6]  
reporting.example.com
```

Aprovisionamiento de Servidores : Ansible

- Inventory:
 - Inventario dinámico : JSON
 - Formato

```
{"my_script": ["dev2","dev"], "_meta": {"hostvars": {"dev2":  
{"ansible_host":"dev2.example.com","ansible_user":"ansible"},"dev":  
{"ansible_host":"dev.example.com","ansible_port":"50022","ansible_user":  
"automation"}}}}
```

- Script de generación

```
machines = fetch_rows("SELECT hostname, user, key, port FROM  
active_machines")  
hostnames = machines.map (m) => return m.hostname  
metadata = {  
  'hostvars'=> {}  
}  
foreach (machines as m) {  
  metadata.hostvars[m.hostname] = {  
    ansible_user => m.user,  
    ansible_port => m.port,  
    ansible_ssh_private_key_file => m.key  
  }  
}  
output_json({  
  'my_script'=> hostnames,  
  '_meta'=> metadata  
})
```


Aprovisionamiento de Servidores :

Ansible

- Inventory:
 - Inventario dinámico :
https://docs.ansible.com/ansible/latest/user_guide/intro_dynamic_inventory.html
 - <https://github.com/ansible/ansible/tree/devel/contrib/inventory>
 - AWS EC2:
 - `ec2.py ec2.ini`

Inventory Script Example: AWS EC2

If you use Amazon Web Services EC2, maintaining an inventory file might not be the best approach, because hosts may come and go over time, be managed by external applications, or you might even be using AWS autoscaling. For this reason, you can use the [EC2 external inventory](#) script.

You can use this script in one of two ways. The easiest is to use Ansible's `-i` command line option and specify the path to the script after marking it executable:

```
ansible -i ec2.py -u ubuntu us-east-1d -m ping
```

The second option is to copy the script to `/etc/ansible/hosts` and `chmod +x` it. You will also need to copy the `ec2.ini` file to `/etc/ansible/ec2.ini`. Then you can run ansible as you would normally.

To successfully make an API call to AWS, you will need to configure Boto (the Python interface to AWS). There are a [variety of methods](#) available, but the simplest is just to export two environment variables:

```
export AWS_ACCESS_KEY_ID='AK123'  
export AWS_SECRET_ACCESS_KEY='abc123'
```

You can test the script by itself to make sure your config is correct:

```
cd contrib/inventory  
./ec2.py --list
```

Aprovisionamiento de Servidores :

Ansible

- Roles:
 - Permite dividir los playbooks en secciones lógicas y así refactorizar
 - Dividir en ficheros
 - Simplificar los playbooks
 - Reutilizar playbooks
 - Podemos tener
 - un fichero para variables
 - Tareas
 - Manejadores
 - Para ejecutar un role hay que incluirlo en un playbook
 - ANSIBLE GALAXY
 - Repositorio de roles
 - CLI
 - 10.000 roles

Aprovisionamiento de Servidores : Ansible

Ejercicio 2 : Instalación de Wordpress

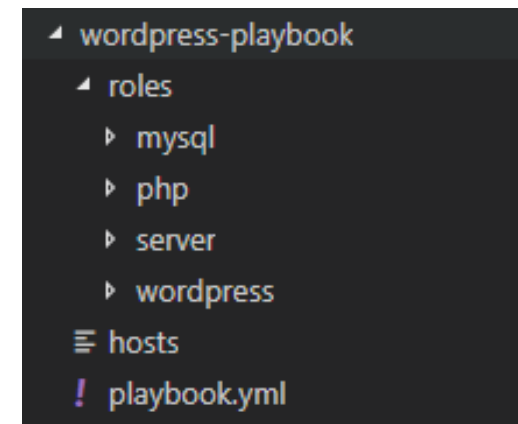
- En este ejercicio se dividirá el playbook en diferentes Roles
- Roles:
 - Servidor Apache
 - Role de PHP
 - Servidor Mysql
 - Wordpress
 - Uso de “**defaults.yml**” para definir variables de un Role/Playbook
- En los playbooks de los roles se utilizará el operador “**with_items**”
 - Este operador permite ejecutar una tarea sobre una lista **{{ ítem }}**
- En uno de los playbooks se utilizarán **HANDLERS**
 - Los handlers permiten realizar operaciones/tareas ante eventos
- Para crear las plantillas (carpetas y ficheros) de los roles se utilizará el comando *ansible-galaxy*
- Se realizará el aprovisionamiento en dos servidores
 - Se utilizará el cluster de vagrant
 - El manager será el servidor que realizará el aprovisionamiento
 - Los workers serán los servidores

Aprovisionamiento de Servidores : Ansible

Ejercicio 2 : Instalación de Wordpress

Creación de roles

- <https://dotlayer.com/how-to-use-an-ansible-playbook-to-install-wordpress/>
- Utilizar la configuración Vagrant para un cluster
 - Manager 192.168.100.8 -> Instalador
 - Manager2 192.168.100.9
 - Worker1 192.168.100.11 -> Objetivo
 - Worker1 192.168.100.12 -> Objetivo
- Crear en una carpeta los ficheros
 - hosts
 - playbook.yml
- Crear los roles mediante galaxy: php , server, mysql, wordpress
 - \$ansible-galaxy init php
 - \$ansible-galaxy init server
 - ...



Aprovisionamiento de Servidores : Ansible

Ejercicio 2 : Instalación de Wordpress

Fichero inventario: hosts

- Con variable de configuración

[wordpress]

192.168.100.11 ansible_ssh_private_key_file=./keys/worker1_private_key

192.168.100.12 ansible_ssh_private_key_file=./keys/worker2_private_key

Aprovisionamiento de Servidores : Ansible

Ejercicio 2 : Instalación de Wordpress : **Playbook.yml**

- Tareas y roles : Se definirá que roles se utilizarán/ejecutarán además de tareas suplementarias
 - La tarea mira si Python esta instalado y sino lo instalar mediante comandos raw (sin utilizar python)

Playbook para instalar wordpress en un LAMP

- **hosts:** all

gather_facts: false

tasks:

- name: install python 2

raw: test -e /usr/bin/python || (apt -y update && apt install -y python-minimal)

- hosts: wordpress

roles:

- server

- php

- mysql

- wordpress

Aprovisionamiento de Servidores : Ansible

Ejercicio 2 : Instalación de Wordpress

Playbook -> Role Server -> server/task/main.yml

```
---
# tasks file for server

- name: Update apt cache
  apt: update_cache=yes cache_valid_time=3600
  become: yes

- name: Install required software
  apt: name={{item}} state=present
  become: yes
  with_items:
    - apache2
    - mysql-server
    - php-mysql
    - php
    - libapache2-mod-php
    - python-mysqldb
```

Aprovisionamiento de Servidores : Ansible

Ejercicio 2 : Instalación de Wordpress

Playbook -> Role -> PHP/task/main.yml

```
---
# tasks file for php

- name: Install php extensions
  apt: name={{item}} state=present
  become: yes
  with_items:
    - php-gd
    - php-ssh2
```


Aprovisionamiento de Servidores : Ansible

Ejercicio 2 : Instalación de Wordpress

Playbook -> Role -> MYSQL

- Fichero de variable *mysql/defaults/main.yml*
- Tareas *mysql/tasks/main.yml*

```
---
# tasks file for mysql

- name: Create mysql database
  mysql_db: name={{wp_mysql_db}}
  become: yes

- name: Create mysql user
  mysql_user:
    name={{wp_mysql_user}}
    password={{wp_mysql_password}}
    priv=*.*:ALL
  become: yes
```

```
---
# defaults file for mysql
wp_mysql_db: wordpress
wp_mysql_user: wordpress
wp_mysql_password: password
```

Aprovisionamiento de Servidores :

Ansible

Ejercicio 2 : Instalación de Wordpress

Playbook -> Role -> WORDPRESS

- Obtención del código de wordpress, descomprimirlo y modificar la configuración del document root del servidor apache y pedir rearrancar el servidor

tasks file for wordpress

- name: Dowload Wordpress

get_url:

url=https://wordpress.org/latest.tar.gz

dest=/tmp/wordpress.tar.gz

validate_certs=no

- name: Extract wordpress

unarchive: src=/tmp/wordpress.tar.gz dest=/var/www copy=no

become: yes

- name: Update default Apache site

become: yes

lineinfile:

dest=/etc/apache2/sites-enabled/000-default.conf

regexp="(.)+DocumentRoot /var/www/html"

line="DocumentRoot /var/www/wordpress"

notify:

- restart apache2

Aprovisionamiento de Servidores : Ansible

Ejercicio 2 : Instalación de Wordpress

Playbook -> Role -> WORDPRESS

- Copiar el código de wordpress a la carpeta “documentRoot”
- Modificar las líneas del wp-config.php para indicar la información de conexión a BBDD

- name: Copy sample config file

command: mv /var/www/wordpress/wp-config-sample.php /var/www/wordpress/wp-config.php
creates=/var/www/wordpress/wp-config.php
become: yes

- name: WordPress config file

lineinfile:

dest=/var/www/wordpress/wp-config.php
regexp="{{ item.regexp }}"
line="{{ item.line }}"

with_items:

- {'regexp': "define\\('DB_NAME', '(.)+' \\);", 'line': "define('DB_NAME', '{{wp_mysql_db}}');"}
- {'regexp': "define\\('DB_USER', '(.)+' \\);", 'line': "define('DB_USER', '{{wp_mysql_user}}');"}
- {'regexp': "define\\('DB_PASSWORD', '(.)+' \\);", 'line': "define('DB_PASSWORD', '{{wp_mysql_password}}');"}
become: yes

Aprovisionamiento de Servidores : Ansible

Ejercicio 2 : Instalación de Wordpress

Playbook -> Role -> WORDPRESS

- Task notify y handler
 - Wordpress/handlers/main.yml

```
---  
# handlers file for wordpress  
- name: restart apache2  
  service: name=apache2 state=restarted  
  become: yes
```

Aprovisionamiento de Servidores :

Ansible

Ejercicio 2 : Instalación de Wordpress

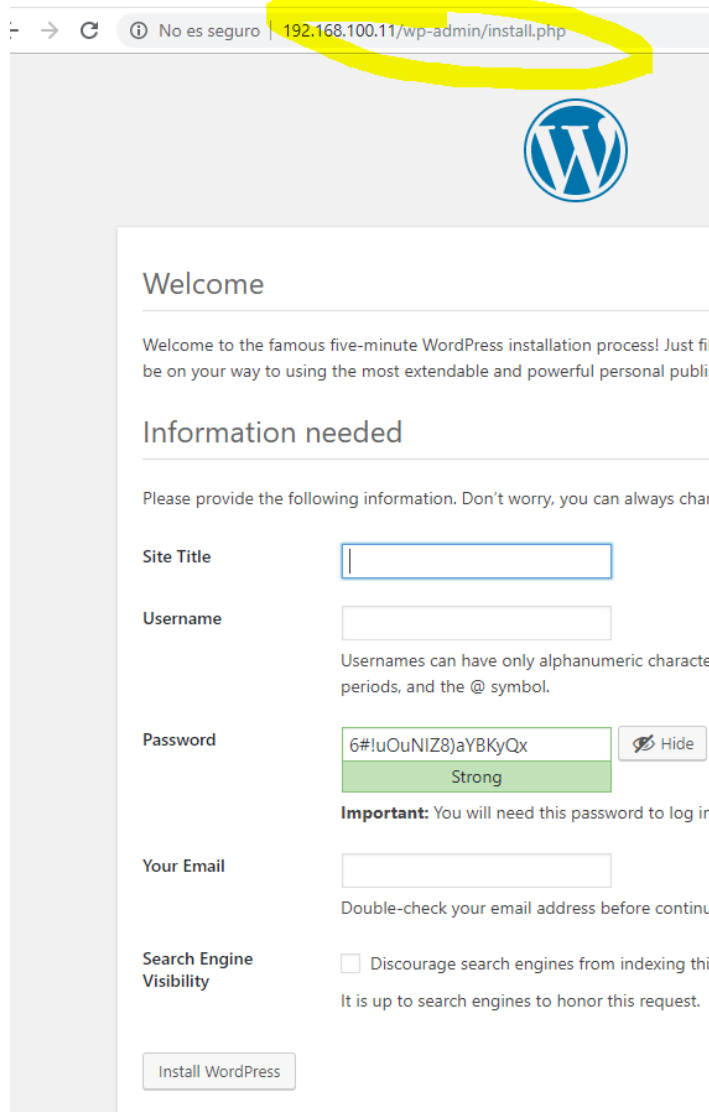
Ejecutar playbook

- `chmod 700 ./keys/*`
- `$ansible-playbook playbook.yml -i hosts -u vagrant`

```
vagrant@manager:~/wordpress-playbook$ ansible-playbook playbook.yml -i hosts -u vagrant
PLAY [all] *****
TASK [install python 2] *****
The authenticity of host '192.168.100.12 (192.168.100.12)' can't be established.
ECDSA key fingerprint is SHA256:L65P1QP/qP51rT51MVRQCY5oxK6H1MH1W1HBUjDy3HA.
Are you sure you want to continue connecting (yes/no)? changed: [192.168.100.11]
yes
changed: [192.168.100.12]
PLAY [wordpress] *****
TASK [Gathering Facts] *****
ok: [192.168.100.11]
ok: [192.168.100.12]
TASK [server : UPdate apt cache] *****
[WARNING]: Could not find aptitude. Using apt-get instead
ok: [192.168.100.11]
changed: [192.168.100.12]
TASK [server : Install required software] *****
[DEPRECATION WARNING]: Invoking "apt" only once while using a loop via squash_actions is deprecated.
Instead of using a loop to supply multiple items and specifying 'name: "{{item}}", please use 'name:
['apache2', 'mysql-server', 'php7.0-mysql', 'php7.0', 'libapache2-mod-php7.0', 'python-mysqldb']' and
remove the loop. This feature will be removed in version 2.11. Deprecation warnings can be disabled by
setting deprecation_warnings=False in ansible.cfg.
[DEPRECATION WARNING]: Invoking "apt" only once while using a loop via squash_actions is deprecated.
Instead of using a loop to supply multiple items and specifying 'name: "{{item}}", please use 'name:
['apache2', 'mysql-server', 'php7.0-mysql', 'php7.0', 'libapache2-mod-php7.0', 'python-mysqldb']' and
remove the loop. This feature will be removed in version 2.11. Deprecation warnings can be disabled by
setting deprecation_warnings=False in ansible.cfg.
ok: [192.168.100.11] => (item=[u'apache2', u'mysql-server', u'php7.0-mysql', u'php7.0', u'libapache2-mod-
php7.0', u'python-mysqldb'])
changed: [192.168.100.12] => (item={u'regexp': u"define\\( 'DB_PASSWORD', '(.)+ '\\\\);", u'line': u"defin
e('DB_PASSWORD', 'password');"})
RUNNING HANDLER [wordpress : restart apache2] *****
changed: [192.168.100.11]
changed: [192.168.100.12]
PLAY RECAP *****
192.168.100.11 : ok=13 changed=5 unreachable=0 failed=0 skipped=0 rescued=0
192.168.100.12 : ok=13 changed=4 unreachable=0 failed=0 skipped=0 rescued=0
vagrant@manager:~/wordpress-playbook$
```

Aprovisionamiento de Servidores : Ansible

Ejercicio 2 : Instalación de Wordpress



The screenshot shows the WordPress installation page in a web browser. The address bar shows the URL `192.168.100.11/wp-admin/install.php`, which is highlighted with a yellow circle. The page features the WordPress logo at the top. Below the logo, there is a 'Welcome' section followed by a paragraph: 'Welcome to the famous five-minute WordPress installation process! Just fill in the information below and you'll be on your way to using the most extendable and powerful personal publishing platform.' The 'Information needed' section follows, with a sub-header 'Please provide the following information. Don't worry, you can always change these settings later.' The form includes fields for 'Site Title', 'Username', 'Password', and 'Your Email'. The 'Password' field is filled with '6#!uOuNIZ8)aYBKyQx' and has a 'Hide' button. Below the password field, it says 'Strong'. There is also a checkbox for 'Search Engine Visibility' with the label 'Discourage search engines from indexing this site'. At the bottom, there is an 'Install WordPress' button.

Welcome

Welcome to the famous five-minute WordPress installation process! Just fill in the information below and you'll be on your way to using the most extendable and powerful personal publishing platform.

Information needed

Please provide the following information. Don't worry, you can always change these settings later.

Site Title

Username

Usernames can have only alphanumeric characters, spaces, periods, and the @ symbol.

Password

6#!uOuNIZ8)aYBKyQx

Hide

Strong

Important: You will need this password to log in.

Your Email

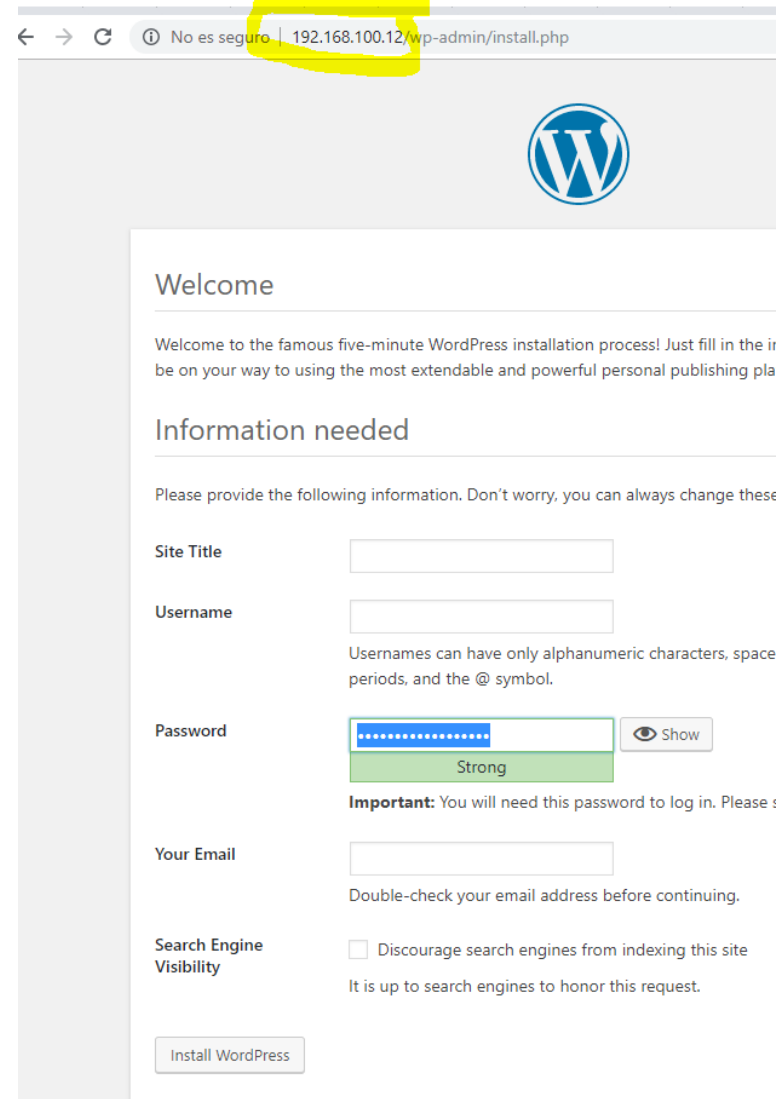
Double-check your email address before continuing.

Search Engine Visibility

☐ Discourage search engines from indexing this site.

It is up to search engines to honor this request.

Install WordPress



The screenshot shows the WordPress installation page in a web browser. The address bar shows the URL `192.168.100.12/wp-admin/install.php`, which is highlighted with a yellow circle. The page features the WordPress logo at the top. Below the logo, there is a 'Welcome' section followed by a paragraph: 'Welcome to the famous five-minute WordPress installation process! Just fill in the information below and you'll be on your way to using the most extendable and powerful personal publishing platform.' The 'Information needed' section follows, with a sub-header 'Please provide the following information. Don't worry, you can always change these settings later.' The form includes fields for 'Site Title', 'Username', 'Password', and 'Your Email'. The 'Password' field is filled with a masked password and has a 'Show' button. Below the password field, it says 'Strong'. There is also a checkbox for 'Search Engine Visibility' with the label 'Discourage search engines from indexing this site'. At the bottom, there is an 'Install WordPress' button.

Welcome

Welcome to the famous five-minute WordPress installation process! Just fill in the information below and you'll be on your way to using the most extendable and powerful personal publishing platform.

Information needed

Please provide the following information. Don't worry, you can always change these settings later.

Site Title

Username

Usernames can have only alphanumeric characters, spaces, periods, and the @ symbol.

Password

.....

Show

Strong

Important: You will need this password to log in. Please save it.

Your Email

Double-check your email address before continuing.

Search Engine Visibility

☐ Discourage search engines from indexing this site.

It is up to search engines to honor this request.

Install WordPress

Aprovisionamiento de Servidores :

Ansible

- Roles: CLI

```
ansible-galaxy install geerlingguy.git -p roles
```

```
vagrant@manager:~$ ansible-galaxy install geerlingguy.git -p roles
- downloading role 'git', owned by geerlingguy
- downloading role from https://github.com/geerlingguy/ansible-role-git/archive/2.0
.5.tar.gz
- extracting geerlingguy.git to /home/vagrant/.ansible/roles/geerlingguy.git
- geerlingguy.git (2.0.5) was installed successfully
- downloading role '%E2%80%93', owned by
[WARNING]: - -p was NOT installed successfully: Content has no field named
'owner'

ERROR! - you can use --ignore-errors to skip failed roles and finish processing the
list.
vagrant@manager:~$ |
```

```
vagrant@manager:~$ ls .ansible/roles/
geerlingguy.git
vagrant@manager:~$ |
```

- hosts: all

roles:

- geerlingguy.git

Aprovisionamiento de Servidores :

Ansible

- Roles: CLI `ansible-galaxy install geerlingguy.git -p roles`
- Estructura de roles
 - <identificador>.<nombreRol>
- Para crear un rol : init

```
mkdir -p provisioning/roles
cd provisioning/roles
ansible-galaxy init proyey.php
```

```
vagrant@manager:~$ ls .ansible/roles/
geerlingguy.git
vagrant@manager:~$ |
```

```
.
├── README.md
├── defaults
│   └── main.yml
├── files
├── handlers
│   └── main.yml
├── meta
│   └── main.yml
├── tasks
│   └── main.yml
├── templates
├── tests
│   ├── inventory
│   └── test.yml
└── vars
    └── main.yml
```


Aprovisionamiento de Servidores :

Ansible

- Roles: CLI `ansible-galaxy install geerlingguy.git -p roles`
- Incluir varios ficheros de tareas en main.yml

```
└─ tasks
|   └─ extensions.yml
|   └─ main.yml
|   └─ php.yml
```

```
---
- include:'php.yml'
- include:'extensions.yml'
```

Aprovisionamiento de Servidores :

Ansible

Ejercicio 3 : Instalación de Apache en servidores AWS EC2

- Roles, Inventory dinámico, variables, gather-facts y ansible-vault
- Se crearan dos maquinas EC2 en una subred pública de la VPC por defecto
- Y se utilizará unas tareas sobre localhost/maquina a recibir las tareas de creación EC2
- Como se utilizarán IP dinámicas y antes de la creación se desconocen sus IP se utilizara un Inventory Dinámico mediante la tarea/modulo
 - **addHost**
- Una vez agregadas las nuevas máquinas EC2 al inventory se realizará el aprovisionamiento de Apache en estas máquinas
- Ansible no tiene un rollback automático como Cloudformation al borrar un stack
 - Esto implica la eliminación explícita de los recursos en un nuevo PLAYBOOK
- Ansible mediante facts ofrece automáticamente información de las diferentes máquinas
 - hostname, IP address, filesystems, OS releases, Users, Network parameters, CPU, memory, ...
 - Los gather_facts se pueden utilizar como variables {{ }} una vez obtenidos
 - El modulo que trabaja los facts de los equipos remotos se llama setup , permite obtener información
 - En un playbook gather_facts: true llama al modulo setup

Aprovisionamiento de Servidores : Ansible

Ejercicio 3 : Instalación de Apache en servidores AWS EC2

Primero un poco de teoria sobre Facts y Variables

- Lectura de facts de localhost
 - Mediante comando
 - ssh-add localhost_private_key
 - ansible local -m setup -i hosts
 - En un playbook se ejecuta con
 - **gather_facts: true**
 - Obtencion del tipo de variable

{{ <the variable name> | type_debug }}

```
---
- hosts: local
  connection: local
  gather_facts: true
  vars:
    - instance_type: t2.micro
    - security_group: webserver
    - image: ami-0c379e7083fbed
    - keypair: master_practica
    - region: eu-west-1
    - count: 2
  vars_files:
```

```
vagrant@manager:~/aws_ansible/ejercicio3$ ssh-agent bash
vagrant@manager:~/aws_ansible/ejercicio3$ ssh-add ../ejercicio1/manager_private_key
Identity added: ../ejercicio1/manager_private_key (../ejercicio1/manager_private_key)
vagrant@manager:~/aws_ansible/ejercicio3$ cat hosts
[local]
localhost ansible_host=127.0.0.1vagrant@manager:~/aws_ansible/ejercicio3$
vagrant@manager:~/aws_ansible/ejercicio3$ ansible local -m setup -i hosts
[DEPRECATION WARNING]: Distribution Ubuntu 16.04 on host localhost should use /usr/bin/python2
to run the command. See https://docs.ansible.com/ansible/2.8/reference_appendices/interpreter_discovery.html
for more details.
localhost | SUCCESS => {
  "ansible_facts": {
    "ansible_all_ipv4_addresses": [
      "192.168.100.10",
      "10.0.2.15",
      "172.17.0.1"
    ],
    "ansible_all_ipv6_addresses": [
      "fe80::a00:27ff:feab:db30",
      "fe80::26:7fff:fe6e:ca81"
    ],
    "ansible_apparmor": {
      "status": "enabled"
    },
    "ansible_architecture": "x86_64",
    "ansible_bios_date": "12/01/2006",
    "ansible_bios_version": "VirtualBox",
    "ansible_cmdline": {
      "BOOT_IMAGE": "/boot/vmlinuz-4.4.0-165-generic",
      "console": "ttyS0",
      "ro": true,
      "root": "UUID=70296e03-0ac0-4419-a43b-c2589dbe6ef0"
    }
  }
```

Aprovisionamiento de Servidores :

Ansible

Ejercicio 3 : Instalación de Apache en servidores AWS EC2

- Utilizando gather_facts en un playbook

```
---
- name: Ansible Variable Example Playbook
  hosts: app
  tasks:

    # display the variable data type
    - debug:
        msg:
          - " Data type of 'ansible_architecture' is {{ ansible_architecture | type_debug }}"
          - " Data type of 'ansible_apparmor' is {{ ansible_apparmor | type_debug }}"
          - " Data type of 'ansible_all_ipv4_addresses' is {{ ansible_all_ipv4_addresses | type_debug }}"

    # Simply printing the value of fact which is Ansible UnSafe Text type
    - debug:
        msg: "{{ ansible_architecture }}"

    # Accessing an element of dictionary
    - debug:
        msg: "{{ansible_apparmor.status}}"

    # Accessing the list
    - debug:
        msg: "{{ansible_all_ipv4_addresses}}"

    # Accessing the Second Element of the list
    - debug:
        msg: "{{ansible_all_ipv4_addresses[1]}}"
```

Aprovisionamiento de Servidores : Ansible

Ejercicio 3 : Instalación de Apache en servidores AWS EC2

- Utilizando gather facts en un playbook

```
aksarav@middlewareinventory:~/VirtualBox VMs/vagrantVM$ ansible-playbook ansible-variable.yml -i ansible_hosts

PLAY [Ansible Variable Example Playbook] *****

TASK [Gathering Facts] *****
ok: [mwiapp01]

TASK [debug] *****
ok: [mwiapp01] => {
  "msg": [
    "Data type of 'ansible_architecture' is AnsibleUnsafeText ",
    "Data type of 'ansible_apparmor' is dict ",
    "Data type of 'ansible_all_ipv4_addresses' is list "
  ]
}

TASK [debug] *****
ok: [mwiapp01] => {
  "msg": "x86_64"
}

TASK [debug] *****
ok: [mwiapp01] => {
  "msg": "disabled"
}

TASK [debug] *****
ok: [mwiapp01] => {
  "msg": [
    "10.0.2.15",
    "192.168.60.4"
  ]
}

TASK [debug] *****
ok: [mwiapp01] => {
  "msg": "192.168.60.4"
}

PLAY RECAP *****
mwiapp01                : ok=6   changed=0    unreachable=0    failed=0

aksarav@middlewareinventory:~/VirtualBox VMs/vagrantVM$
```

Aprovisionamiento de Servidores :

Ansible

Ejercicio 3 : Instalación de Apache en servidores AWS EC2

- Utilizando listas de variables

```
---
- name: Ansible Variable Example Playbook
  hosts: app
  tasks:

    # Print the Dictionary
    - debug:
        msg: "{{ansible_mounts}}"

    # Parsing through Variable Dictionary
    - debug:
        msg: "Mount Point {{item.mount}} is at {{item.block_used/item.block_total*100}} percent "
        loop: "{{ansible_mounts}}"

    # Execute Host based task using variable
    - name: Execute the command only mwiapp01 server
      become: yes
      become_user: root
      shell: "uname -a"
      when: "{{ ansible_hostname == 'mwiapp01' }}"
```

Aprovisionamiento de Servidores :

Ansible

Ejercicio 3 : Instalación de Apache en servidores AWS EC2

Manos a la obra con el ejercicio

- Objetivo:
 - Primer playbook
 - Crear 2 máquinas con un tag
 - Crear un grupo de seguridad
 - Instalar en las maquinas creadas Apache2 y arrancar
 - Segundo playbook
 - Terminar las instancias EC2 creadas con un tag
- Requisitos previos en la maquina destinada a crear las maquinas Ec2
 - \$pip install boto
 - \$pip install awscli
 - ansible –versión
 - 2.8.0
 - Cuenta AWS
 - Access key para acceso CLI -> ansible-vault
 - Par de claves para acceso SSH a recursos EC2

Aprovisionamiento de Servidores :

Ansible

Ejercicio 3 : Instalación de Apache en servidores AWS EC2

Configuración cuenta AWS

- Podemos utilizar *aws configure* (~/.aws/credentials) y de esta manera ansible utilizará dicha configuración
 - Si tenemos que utilizar un token de acceso obligatorio utilizar ~/.aws/credentials
- Si utilizamos *aws configure* (~/.aws/credentials) el id y secreto de acceso se guardarán en abierto
- En lugar de utilizar ~/.aws/credentials el modulo de EC2 de ansible permite especificar en cada tarea el `aws_access_key` y el `aws_secret_key` (no permite especificar un `aws_access_token`)
 - En este caso podemos utilizar variables para especificar dicha configuración
 - Las variables se pueden especificar en un fichero .yml
 - Mediante ansible-vault podemos crear ficheros .yml encriptados de variables

Aprovisionamiento de Servidores :

Ansible

Ejercicio 3 : Instalación de Apache en servidores AWS EC2

Configurar los playbook con el AWS Access-key

- Gestión de secretos : ansible-vault

```
$ansible-vault create aws_keys.yml
```

```
aws_access_key: BOITAGAGAFAT1111
```

```
aws_secret_key: iJ2JJ292828282IsZD43RVuSKFnUt
```

```
aws_session_token: FQoGZXIvYXd
```

```
$ cat aws_keys.yml
```

```
$ANSIBLE_VAULT;1.1;AES256
```

```
63333038396266346466383037653433613336643164316566353030663162303434323339316330  
3661333432313564646432333563343935323463346163630a656233303535333534346262616465  
34373063393132336165313562613830306262646538656334643532303861366539336234363462  
6438376165396638390a326334303263303530643965373539323239623931383839383539616631  
38343534643061373361373239313264633562323936663130626537333164666262633636306464  
39396531616335313563323339633237396131363938616262663536303664333065636334616163  
35383631616231626661346532346666386338346336666535636263663334343364326237303366  
61313764363331356330386639323666373433323733383636373635656335313234643364333832  
3066
```

```
$ansible-playbook -i hosts --ask-vault-pass ec2.yml
```

Aprovisionamiento de Servidores :

Ansible

Ejercicio 3 : Instalación de Apache en servidores AWS EC2 Y para utilizar los vaults en un playbook

```
- hosts: local
  connection: local
  gather_facts: true
  vars:
    instance_type: t2.micro
    security_group: webserver_sg
    image: ami-04670c8e8d4c718ac #ami-04b9e92b5572fa0d1
    keypair: educate_1
    region: us-east-1
    count: 2
  vars_files:
    - aws_keys.yml

- name: Launch the new EC2 Instance
  ec2:
    aws_access_key: "{{ aws_access_key }}"
    aws_secret_key: "{{ aws_secret_key }}"
    group: "{{ security_group }}"
    instance_type: "{{ instance_type }}"
    image: "{{ image }}"
    wait: true
    region: "{{ region }}"
    keypair: "{{ keypair }}"
    count: "{{ count }}"
    register: ec2
```

\$ansible-playbook -i hosts --ask-vault-pass ec2.yml

Aprovisionamiento de Servidores :

Ansible

Ejercicio 3 : Instalación de Apache en servidores AWS EC2

Las credenciales para AWS CLI del classroom requieren token!!!

Y el modulo EC2 de ansible no ofrece la configuración de tokens, por lo tanto para hacer este ejercicio no se podrá utilizar Vault para encriptar, se utilizará *~/.aws/credentials* en abierto!!!!!!

Además dependiendo de la versión de Ansible es preferible establecer las credenciales en variables de entorno para que todo funcione perfectamente. Por lo tanto, ejecutar mejor en un fichero.sh):

```
export AWS_ACCESS_KEY_ID=$(sed -n -e 's/aws_access_key_id=//p' < ~/.aws/credentials)
export AWS_SECRET_ACCESS_KEY=$(sed -n -e 's/aws_secret_access_key=//p' < ~/.aws/credentials)
export AWS_SESSION_TOKEN=$(sed -n -e 's/aws_session_token=//p' < ~/.aws/credentials)
```

Aprovisionamiento de Servidores :

Ansible

Ejercicio 3 : Instalación de Apache en servidores AWS EC2

- Crear una carpeta para el playbook *ec2.yml*
- Guardar las claves de acceso de AWS en el vault
 - `ansible-vault create aws_keys.yml`
`aws_access_key: HYUCDSWYUCYFNMCMBOITV64`
`aws_secret_key: VDVKORIRihucs6s34iMcMw4T`
 - En el caso de utilizar Classroom utilizar en abierto `~/.aws/credentials`

```
[default]
aws_access_key_id=ASIATPHGRILVDDDSR
aws_secret_access_key=ydQqfX6thdW675Aa8kmq6eGEZT0k0PUcyZBo
aws_session_token=FQoGZXIvYXdzEDYaDPxfHeXh2rV7l2ZKiBVvcjPouC53rvGPI/YsyjLtMbtBQ=78=
```

- Configurar el inventario *hosts*
 - En el inventario solo se listara el ordenador que ejecutara AWS CLI para la creación de recursos
 - Una vez creados los servidores EC2 se lanzaran tareas de aprovisionamiento software donde se leerán las IP de los recursos EC2 de forma dinámica

```
[local]
localhost ansible_host=127.0.0.1 ansible_ssh_private_key_file=./manager_private_key
```

Aprovisionamiento de Servidores : Ansible

Ejercicio 3 : Instalación de Apache en servidores AWS EC2

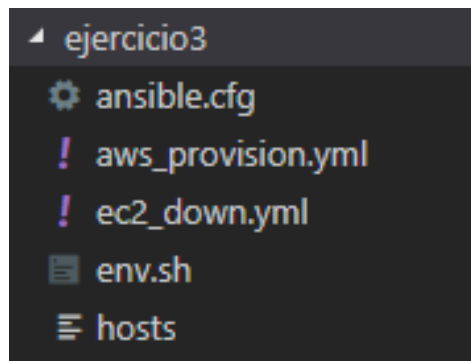
- En ansible.cfg se configurará la configuración global de SSH para las demás máquinas

[defaults]

host_key_checking = False

private_key_file = /home/vagrant/aws_ansible/ejercicio3/educate_1.pem

- Ficheros para el ejercicio



Aprovisionamiento de Servidores : Ansible

Ejercicio 3 : Instalación de Apache en servidores AWS EC2

Playbook *aws_provision.yml*

- Utiliza variables para especificar
 - El tipo de instancias
 - El nombre del grupo de seguridad
 - El tipo de ami
 - El nombre del par de claves de acceso SSH
 - La región
 - Numero de instancias
- Fichero de variables encriptado (vault) para claves de acceso a AWS (en el caso del classroom no es posible)

Aprovisionamiento de Servidores : Ansible

Ejercicio 3 : Instalación de Apache en servidores AWS EC2

- Playbook para las acciones a realizar por el localhost que realizara la creación de los recursos EC2

```
- hosts: local
  connection: local
  gather_facts: false
  vars:
    instance_type: t2.micro
    security_group: webservers_sg
    image: ami-04670c8e8d4c718ac
    keypair: educate_1
    region: us-east-1
    count: 2
  vars_files:
    - aws_keys.yml
```

Aprovisionamiento de Servidores : Ansible

Ejercicio 3 : Instalación de Apache en servidores AWS EC2

- Al no utilizar Vault comentar las líneas de ***vars_files***:

```
- hosts: local
  connection: local
  gather_facts: false
  vars:
    instance_type: t2.micro
    security_group: webservers_sg
    image: ami-04670c8e8d4c718ac
    keypair: educate_1
    region: us-east-1
    count: 2
  #vars_files:
  # - aws_keys.yml
```


Aprovisionamiento de Servidores :

Ansible

Ejercicio 3 : Instalación de Apache en servidores AWS EC2

Playbook-> aws_provision.yml

- Especificación de clave privada PEM
 - a) En el inventory

```
[local]
localhost
[servers]
server1 ansible_ssh_private_key_file="{{ playbook_dir }}/master_practicas_irlanda.pem"
```

b) ansible.cfg

```
[defaults]
host_key_checking = False
private_key_file = /home/vagrant/aws_ansible/prueba2/master_practicas_irlanda.pem
```

Aprovisionamiento de Servidores : Ansible

Ejercicio 3 : aws_provision.yml : LAS TAREAS ::Creando un grupo de seguridad

tasks:

- name: Create a security group

ec2_group:

name: "{{ security_group }}"

description: The webserver security group

region: "{{ region }}"

aws_access_key: "{{ aws_access_key }}"

Comentar esta línea!!!!!!!!!!!!!!!!!!!!!!!!!!!!

aws_secret_key: "{{ aws_secret_key }}"

Comentar esta línea!!!!!!!!!!!!!!!!!!!!!!!!!!!!

rules:

- proto: tcp

from_port: 22

to_port: 22

cidr_ip: 0.0.0.0/0

- proto: tcp

from_port: 80

to_port: 80

cidr_ip: 0.0.0.0/0

- proto: tcp

from_port: 443

to_port: 443

cidr_ip: 0.0.0.0/0

rules_egress:

- proto: all

cidr_ip: 0.0.0.0/0

Aprovisionamiento de Servidores : Ansible

Ejercicio 3 : aws_provision.yml : LAS TAREAS :: Creación y ejecución de las instancias

- Se registra la tarea para poder acceder a la información de las instancias creadas a posteriori
 - Las IP públicas por ejemplo
- La tarea espera la finalización de la creación de los servidores
 - name: Launch the new EC2 Instance

ec2:

aws_access_key: "{{ aws_access_key }}"

Comentar esta línea!!!!!!!!!!!!!!!!!!!!!!!!!!!!

aws_secret_key: "{{ aws_secret_key }}"

Comentar esta línea!!!!!!!!!!!!!!!!!!!!!!!!!!!!

group: "{{ security_group }}"

instance_type: "{{ instance_type }}"

image: "{{ image }}"

wait: true

region: "{{ region }}"

keypair: "{{ keypair }}"

count: "{{ count }}"

register: ec2

Aprovisionamiento de Servidores :

Ansible

Ejercicio 3 : aws_provision.yml : : LAS TAREAS :: Agregando las nuevas ips al inventario de forma dinámica

- Se agregan las IP de los servidores creados al grupo **webservers**
- Se utiliza la variable registrada previamente **ec2**

- name: Add the newly created host so that we can further contact it

add_host:

name: "{{ item.public_ip }}"

groups: webservers

with_items: "{{ ec2.instances }}"

Aprovisionamiento de Servidores : Ansible

Ejercicio 3 : aws_provision.yml : LAS TAREAS :: Tagueando las instancias creadas

- Type = webserver
- Estos tags nos permitirán eliminar desde otro playbook los recursos creados en una operación
- Se utiliza la variable registrada previamente **ec**

- name: Add tag to Instance(s)

ec2_tag:

aws_access_key: "{{ aws_access_key }}"

aws_secret_key: "{{ aws_secret_key }}"

resource: "{{ item.id }}"

region: "{{ region }}"

state: "present"

with_items: "{{ ec2.instances }}"

args:

tags:

Type: webserver

Aprovisionamiento de Servidores :

Ansible

Ejercicio 3 : aws_provision.yml : : LAS TAREAS :: **Comprobar la conectividad con los servidores creados**

- Se utiliza la variable registrada previamente **ec**

- name: Wait for SSH to come up

wait_for:

host: "{{ item.public_ip }}"

port: 22

state: started

with_items: "{{ ec2.instances }}"

Aprovisionamiento de Servidores : Ansible

Ejercicio 3 : aws_provision.yml : : LAS TAREAS :: **Aprovisionando con apache las máquinas creadas**

- Se ejecuta sobre el grupo webservers creado dinámicamente del inventario
- Se ejecutan dos tareas
 - Una para instalar apache y la segunda para arrancar el servicio

- **hosts:** webservers

remote_user: ubuntu

become: yes

gather_facts: no

pre_tasks:

- name: 'install python'

- raw: 'sudo apt-get -y install python'

tasks:

- name: Install Apache

- apt:**

- name: apache2

- state: present

- **service:**

- name: apache2

- state: started

- enabled: yes

Aprovisionamiento de Servidores : Ansible

Ejercicio 3 : aws_provision.yml : Ejecución del playbook

ansible-playbook -i hosts --ask-vault-pass aws_provision.yml

ansible-playbook -i hosts -aws_provision.yml **(sin vaults!!!!)**

	Nam	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks
		i-0695300e173a6d6c5	t2.micro	eu-west-1a	 pending	 Initializing
		i-0d27b00f8ec1b29db	t2.micro	eu-west-1a	 pending	 Initializing

```

PLAY [webservers] *****

TASK [install python] *****
changed: [54.194.39.199]
changed: [54.194.39.199]
changed: [34.244.103.225]

TASK [Install Apache] *****
[WARNING]: Updating cache and auto-installing missing dependency: python-apt

[WARNING]: Updating cache and auto-installing missing dependency: python-apt

[WARNING]: Could not find aptitude. Using apt-get instead

[DEPRECATION WARNING]: Distribution Ubuntu 18.04 on host 54.194.39.199 should use /usr/bin/python3, but is using /u
Ansible releases. A future Ansible release will default to using the discovered platform python for this host. See
https://docs.ansible.com/ansible/2.8/reference_appendices/interpreter_discovery.html for more information. This fea
warnings can be disabled by setting deprecation_warnings=False in ansible.cfg.
changed: [54.194.39.199]
[DEPRECATION WARNING]: Distribution Ubuntu 18.04 on host 34.244.103.225 should use /usr/bin/python3, but is using /
Ansible releases. A future Ansible release will default to using the discovered platform python for this host. See
https://docs.ansible.com/ansible/2.8/reference_appendices/interpreter_discovery.html for more information. This fea
warnings can be disabled by setting deprecation_warnings=False in ansible.cfg.
changed: [34.244.103.225]

TASK [service] *****
ok: [34.244.103.225]
ok: [54.194.39.199]

PLAY RECAP *****
34.244.103.225      : ok=3    changed=2    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
54.194.39.199      : ok=3    changed=2    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
localhost          : ok=5    changed=3    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
  
```


Aprovisionamiento de Servidores : Ansible

Ejercicio 3 : aws_provision.yml : Ejecución del playbook
 ansible-playbook -i hosts --ask-vault-pass ec2.yml

```

PLAY RECAP *****
34.244.103.225      : ok=3
54.194.39.199      : ok=3
localhost          : ok=5
  
```

Filter by tags and attributes or search by keyword

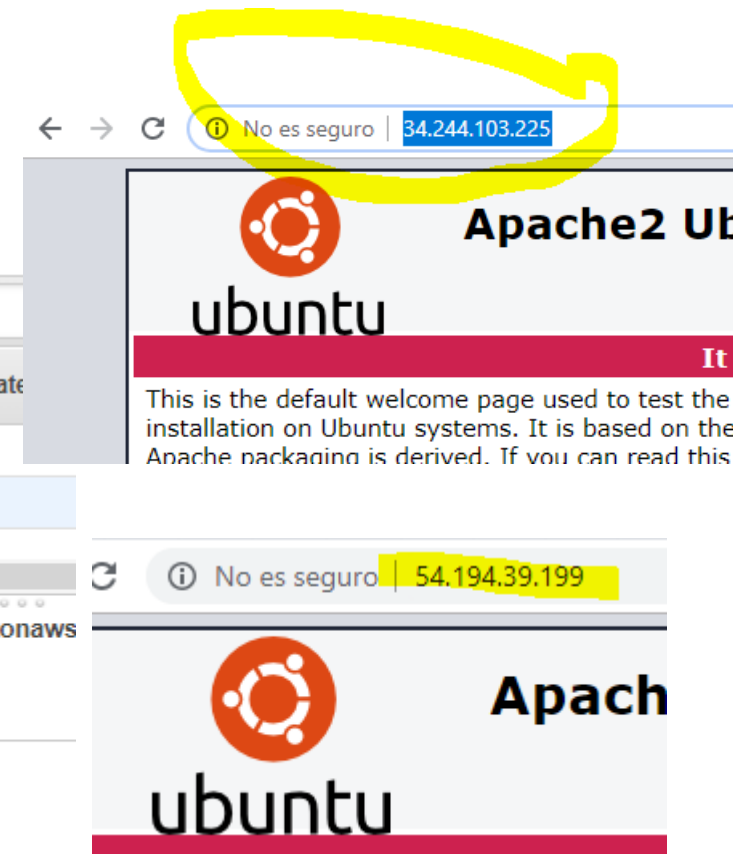
	Name	Instance ID	Instance Type	Availability Zone	Instance State
<input type="checkbox"/>		i-0695300e173a6d6c5	t2.micro	eu-west-1a	running
<input checked="" type="checkbox"/>		i-0d27b00f8ec1b29db	t2.micro	eu-west-1a	running

Instance: **i-0d27b00f8ec1b29db** Public DNS: ec2-54-194-39-199.eu-west-1.compute.amazonaws.com

Description | Status Checks | Monitoring | **Tags**

Add/Edit Tags

Key	Value
Type	webserver



Aprovisionamiento de Servidores :

Ansible

Ejercicio 3 : EC2_down.yml : playbook para eliminar los recursos creados

```

agrant@manager:~/aws_ansible/prueba2$ ansible-playbook -i hosts --ask-vault-pass ec2_down.yml
Vault password:

LAY [local] *****

ASK [Gathering Facts] *****
k: [localhost]

ASK [Gather EC2 facts] *****
k: [localhost]



ASK [debug] *****
k: [localhost] => {
  "ec2": {
    "changed": false,
    "failed": false,
    "instances": [
      {

```

```

PLAY RECAP *****
localhost : ok=4  changed=1  unreachable=0  failed=0  skipped=0  rescued=0  ignored=0

```

<input type="checkbox"/>	Nam▼	Instance ID▼	Instance Type▲	Availability Zone▼	Instance State▼
<input type="checkbox"/>		i-0695300e173a6d6c5	t2.micro	eu-west-1a	 terminated
<input checked="" type="checkbox"/>		i-0d27b00f8ec1b29db	t2.micro	eu-west-1a	 terminated

Aprovisionamiento de Servidores :

Ansible

Ejercicio 3 : EC2_down.yml : playbook para eliminar los recursos creados

```
- hosts: local
  connection: local
  vars:
    region: us-east-1
  vars_files:
    - aws_keys.yml          Comentar esta línea!!!!!!
  tasks:
    - name: Gather EC2 facts
      ec2_instance_facts:
        region: "{{ region }}"
        filters:
          "tag:Type": "webserver"
        aws_access_key: "{{ aws_access_key }}"      Comentar esta línea!!!!!!
        aws_secret_key: "{{ aws_secret_key }}"      Comentar esta línea!!!!!!
        register: ec2
    - debug: var=ec2

    - name: Terminate EC2 Instance(s)
      ec2:
        instance_ids: '{{ item.instance_id }}'
        state: absent
        region: "{{ region }}"
        aws_access_key: "{{ aws_access_key }}"      Comentar esta línea!!!!!!
        aws_secret_key: "{{ aws_secret_key }}"      Comentar esta línea!!!!!!
        with_items: "{{ ec2.instances }}"
```

Aprovisionamiento de Servidores : Ansible

Ejercicio 4 : Actualización de Servidores AWS EC2

- El objetivo de esta práctica es actualizar algunos servidores EC2 una vez creados y sin conocer sus IP
- Se utilizar el Inventory Dinámico pero en este caso mediante
 - Utilizar scripts ec2.py y ec2.ini
- <https://aws.amazon.com/es/blogs/apn/getting-started-with-ansible-and-dynamic-amazon-ec2-inventory-management/>

Aprovisionamiento de Servidores : Ansible

Ejercicio 4 : Actualización de Servidores AWS EC2 Credenciales de acceso CLI AWS

- ~/.aws/credentials
- sh env.sh

```
export AWS_ACCESS_KEY_ID=$(sed -n -e 's/aws_access_key_id=//p' < ~/.aws/credentials)
export AWS_SECRET_ACCESS_KEY=$(sed -n -e 's/aws_secret_access_key=//p' < ~/.aws/credentials)
export AWS_SESSION_TOKEN=$(sed -n -e 's/aws_session_token=//p' < ~/.aws/credentials)
```

Aprovisionamiento de Servidores : Ansible

Ejercicio 4 : Actualización de Servidores AWS EC2

Asignación del Inventory al script Ec2.py

- Descarga de los script ec2.py y ec2.ini

```
wget https://raw.githubusercontent.com/ansible/ansible/devel/contrib/inventory/ec2.py
```

```
wget https://raw.githubusercontent.com/ansible/ansible/devel/contrib/inventory/ec2.ini
```

```
vagrant@manager:~/aws_ansible/ejercicio4$ ls  
ec2.ini  ec2.py  
vagrant@manager:~/aws_ansible/ejercicio4$ |
```

- Copiar a la carpeta de Ansible /etc/ansible/ec2.py
- Indicar a Ansible ec2.py como Inventory de Hosts

```
export ANSIBLE_HOSTS=/etc/ansible/ec2.py
```

```
export EC2_INI_PATH=/etc/ansible/ec2.ini
```

- Agregar a SSH las claves de acceso a las maquinas Ec2

```
ssh-agent bash
```

```
ssh-add ~/.ssh/educate_1.pem
```

Aprovisionamiento de Servidores : Ansible

Ejercicio 4 : Actualización de Servidores AWS EC2

Obtener la lista de maquinas

- Verificar la configuración ec2.ini
 - Regiones validas
 - rds=false
 - Elasticcache=false
- `ec.py --list`

```
vagrant@manager:~/aws_ansible/ejercicio4$ /etc/ansible/ec2.py --list| more
{
  "_meta": {
    "hostvars": {
      "3.81.120.138": {
        "ansible_host": "3.81.120.138",
        "ec2_in_monitoring_element": false,
        "ec2_account_id": "238854161131",
        "ec2_ami_launch_index": "0",
        "ec2_architecture": "x86_64",
        "ec2_block_devices": {
          "sda1": "vol-02ed2b3392205f05f"
        },
        "ec2_client_token": "",
        "ec2_dns_name": "ec2-3-81-120-138.compute-1.amazonaws.com",
        "ec2_ebs_optimized": false,
```

Aprovisionamiento de Servidores :

Ansible

Ejercicio 4 : Actualización de Servidores AWS EC2

Ejecutar tareas sobre algunas maquinas EC2 de AWS

- Comprobación conectividad SSH

```
ansible -m ping -i /etc/ansible/ec2.py tag_* -u ubuntu
```

```
vagrant@manager:~/aws_ansible/ejercicio4$ ansible -m ping -i /etc/ansible/ec2.py tag_* -u ubuntu
```

```
54.164.149.251 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python"
  },
  "changed": false,
  "ping": "pong"
}
```










```
3.95.138.220 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python"
  },
  "changed": false,
  "ping": "pong"
}
```


Aprovisionamiento de Servidores : Ansible

Ejercicio 4 : Actualización de Servidores AWS EC2

Ejecutar tareas sobre algunas maquinas EC2 de AWS

- Ahora lanzamos una nueva instancia y volvemos a ejecutar
- Y ansible mediante ec2.bv detectara la nueva instancia

	i-0c637b998d87166e3	t2.micro	us-east-1a	 running	 2/2 checks passed
	i-0ca6e59fa22e18f20	t2.micro	us-east-1c	 running	 2/2 checks passed
	i-0ce19ba472bf35921	t2.micro	us-east-1a	 pending	 Initializing

```
vagrant@manager:~/aws_ansible/ejercicio4$ ansible -m ping -i /etc/ansible/ec2.py tag_* -u ubuntu
```

```

54.164.149.251 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "changed": false,
  "ping": "pong"
}

3.95.138.220 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python"
  },
  "changed": false,
  "ping": "pong"
}

54.162.159.208 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "changed": false,
  "ping": "pong"
}

```

Aprovisionamiento de Servidores : Ansible

Ejercicio 4 : Actualización de Servidores AWS EC2

Ejecutar tareas sobre algunas maquinas EC2 de AWS

- Y ahora vamos a actualizar aquellas maquinas EC2 con el tag Type=webserver porque es necesario aplicarles un update en sus aplicaciones

```
ansible -m apt -a "name=* state=latest" tag_* -i /etc/ansible/ec2.py -u ubuntu -b --become-method=sudo --become-user=root
```

```
vagrant@manager:~/aws_aws/ejercicio4$ ansible -m apt -a "name=* state=latest" tag_* -i /etc/ansible/ec2.py -u ubuntu -b --become-method=sudo --become-user=root
[DEPRECATION WARNING]: The TRANSFORM_INVALID_GROUP_CHARS settings is set to allow bad characters in group names by default, this will change, but still be user configurable on deprecation. This feature will be removed in version 2.10. Deprecation warnings can be disabled by setting deprecation_warnings=False in ansible.cfg.
[WARNING]: Invalid characters were found in group names but not replaced, use -vvvv to see details

[WARNING]: Could not find aptitude. Using apt-get instead

54.162.159.208 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "changed": false,
  "msg": "Reading package lists...\nBuilding dependency tree...\nReading state information...\nCalculating upgrade...\n0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.\n",
  "stderr": "",
  "stderr_lines": [],
  "stdout": "Reading package lists...\nBuilding dependency tree...\nReading state information...\nCalculating upgrade...\n0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.\n",
  "stdout_lines": [
    "Reading package lists...",
    "Building dependency tree...",
    "Reading state information...",
    "Calculating upgrade...",
    "0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded."
  ]
}
```

Aprovisionamiento de Servidores :

Ansible

Ejercicio 4 : Actualización de Servidores AWS EC2

Ejecutar tareas sobre algunas maquinas EC2 de AWS

- Otros ejemplos

```
ansible -m command -a "apt update" tag_* -i /etc/ansible/ec2.py -u ubuntu -b --  
become-method=sudo --become-user=root
```

```
ansible -m command -a "apt install python3" tag_* -i /etc/ansible/ec2.py -u ubuntu -b  
--become-method=sudo --become-user=root
```

Aprovisionamiento de Servidores : Ansible

Ejercicio 4 : Actualización de Servidores AWS EC2

Ejecutar tareas sobre algunas maquinas EC2 de AWS

- Utilizando ec2.py con un playbook

```
- hosts: tag_Type_webserver
  user: ubuntu
  sudo: yes
  tasks:
    - name: Update all packages to latest
      apt: name=* state=latest

    - name: Install apache2
      apt: name='apache2' state=present
```

ansible-playbook playbook.yml -i /etc/ansible/ec2.py

```
PLAY [tag_Type_webserver] *****

TASK [Gathering Facts] *****
ok: [54.162.159.208]
ok: [3.95.138.220]

TASK [Update all packages to latest]
[WARNING]: Could not find aptitude.

ok: [3.95.138.220]
ok: [54.162.159.208]

TASK [Install specific nginx package]
ok: [3.95.138.220]
ok: [54.162.159.208]

PLAY RECAP *****
3.95.138.220           : ok=3
54.162.159.208        : ok=3
```

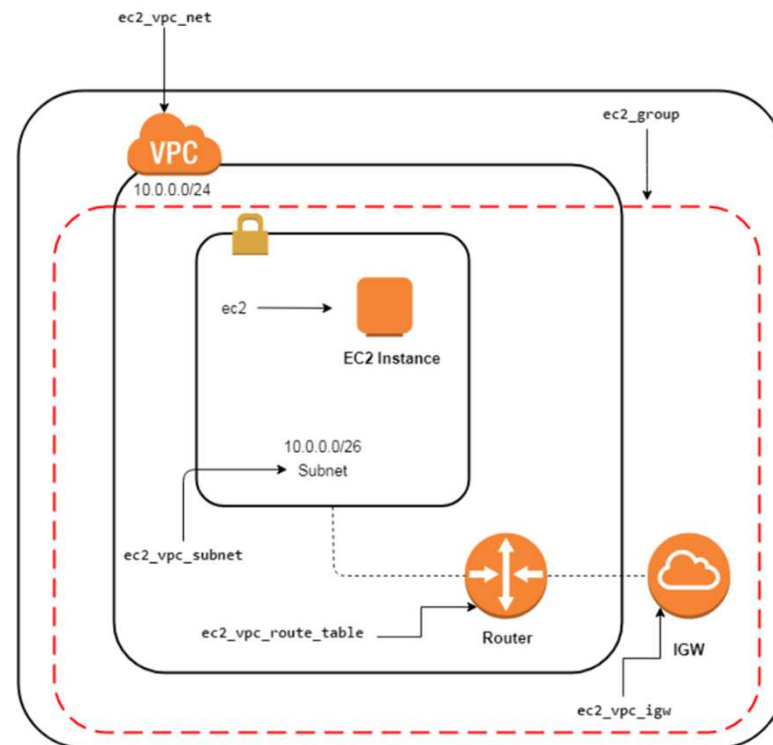
Aprovisionamiento de Servidores :

Ansible

Ejercicio 5 : Creación de arquitectura HA en AWS

En este ejercicio se plantea crear una arquitectura HA para wordpress con Ansible

- Crear un VPC
- Crear subredes
 - 1 pública
 - 2 privadas
- 1 IGW
- Tablas de rutado
- Grupos de seguridad
- Crear un bastion SSH
- Crear un ELB
- Crear dos servidores WEB



Aprovisionamiento de Servidores :

Ansible

Ejercicio 5 : Creación de arquitectura HA en AWS

- Se utilizara un role para crear el VPC y todo los elementos de red
- Se utilizará un role para crear una maquina bastion
- Se utilizará un role para crear el balanceo de carga ELB
- Se utilizará un playbook para destruir todos los recursos creados
- Se utilizará un playbook con roles y las un par de tareas para aprovisionar las maquinas creadas

Aprovisionamiento de Servidores :

Ansible

Ejercicio 5 : Creación de arquitectura HA en AWS

- En este ejercicio se plantean una serie de iteraciones
 1. Todos las maquinas en una subred pública donde los servidores presentan Ips públicas (En esta iteración la agregación de instancias al elb se ejecuta con *'ignore_errors'* debido a que en el momento de agregar al elb las instancias estas no tienen un servidor web que ofrece una página para el HEALTH_CHECK)
 2. Esta iteración es equivalente al anterior pero para evitar el "ignore_errors" la tarea de agregación de maquinas al ELB se realiza tras el aprovisionamiento de cada una de las maquinas EC2 (esto se realiza en el playbook ec2.yml)
 3. Ahora los servidores se crean con Ips privadas lo cual no permite aprovisionar desde vuestro PC a las maquinas EC2 que realizarán la tarea de servidores web. Por lo tanto solo podréis acceder a las maquinas via ssh desde el bastion

Aprovisionamiento de Servidores :

Ansible

Ejercicio 5 : Creación de arquitectura HA en AWS

- En este ejercicio se plantean una serie de iteraciones
 4. En esta iteración se crea se utiliza el playbook ec2.down para poder finalizar todos los recursos creados
 5. Se divide el aprovisionamiento en dos playbook. El primero se ejecutara en localhost. En este ejemplo todas las IP vuelven a ser públicas
 - a) Para la creación de la red y el bastion
 - b) Para el aprovisionamiento de los servdiores y la creación del balanceador de carga
 6. En esta ultima iteración los servidores web van en las subredes privadas. En este caso el playbook que crea la red y el bastion se ejecuta en localhost. Mientras que el segundo playbook se ejecuta en el BASTION, ya que los servidores web tienen lps privadas

Aprovisionamiento de Servidores :

Ansible

Ejercicio 5 : Creación de arquitectura HA en AWS

- En estas transparencias solo se muestra la primera iteración.
- El código de las demás iteraciones la tenéis en Mudle y en gitlab
- Todo el código ofrecido esta configurado para ejecutarse en
 - Región: eu-west-1
 - Availability zones: us-east-1a, us-east-1b , us-east-1c
 - Con Access Id/Key
- Por lo tanto tendréis que modificare el entorno y las variables para la ejecución en vuestro Classroom/Starter Account

Aprovisionamiento de Servidores : Ansible

Ejercicio 5 : Creación de arquitectura HA en AWS

Iteración 1: HA con ips públicas

- Copiar el código
- Modificar variables
- Ejecutar playbook

```
ansible-playbook -i hosts ec2.yml
```

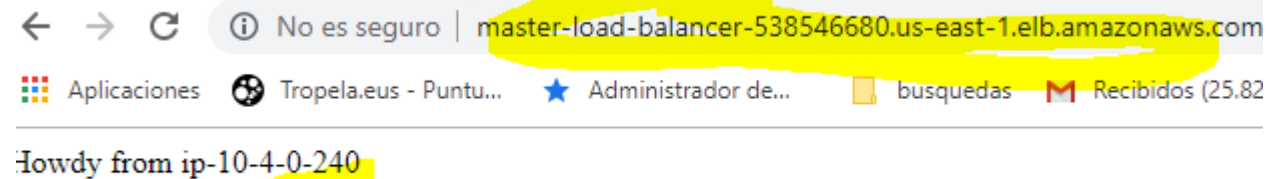
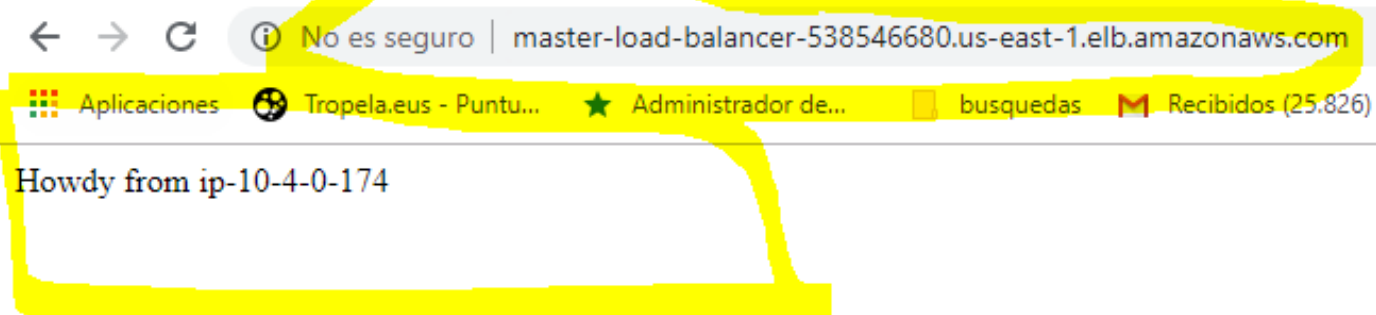
```
en_index : u'0', u'dns_name': u'ec2-54-162-204-15',  
: False, u'launch_time': u'2019-10-25T13:59:12',  
ture': u'x86_64', u'hypervisor': u'xen', u'vir  
  
PLAY RECAP *****  
54.162.204.15      : ok=5    changed=4  
localhost         : ok=40   changed=3  
  
vagrant@manager:~/aws_ansible/ejercicio5/prueb
```

Aprovisionamiento de Servidores : Ansible

Ejercicio 5 : Creación de arquitectura HA en AWS

Iteración 1: HA con ips públicas

ansible-playbook -i hosts ec2.yml



Aprovisionamiento de Servidores : Ansible

Ejercicio 5 : Creación de arquitectura HA en AWS

Iteración 2,3,4,5,6: A realizar utilizando como base la primera iteración