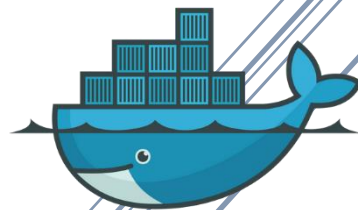


# Despliegue ágil de microservicios



ANSIBLE



docker

aws



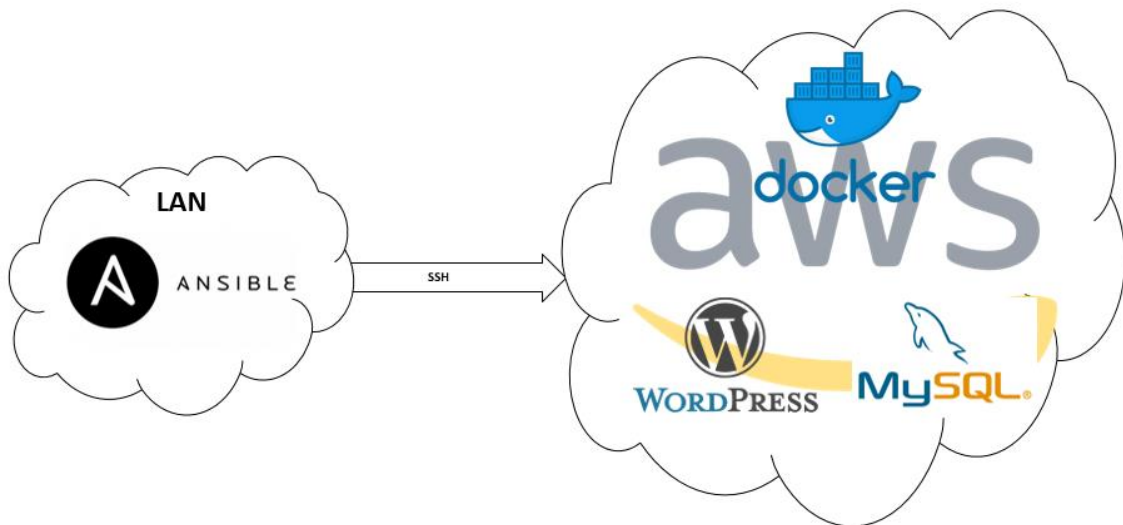
Curso: S21AR

Realizado por: José Antonio Díaz Gómez

## Índice

Introducción .....	3
Crear instancias AWS EC2 con ansible .....	3
Aprovisionar Instancia .....	8
WEBGRAFÍA .....	14

## Introducción



En este esquema se ha montado una instancia en AWS y se ha aprovisionado con ansible. Los servicios aprovisionados son los siguientes.

- Docker.
- Pip.
- Docker-py.

Estos son los servicios aprovisionados en la instancia pero además de eso en docker se ha montado un wordpress que está compuesto de dos contenedores uno con mysql y otro con docker.

## Crear instancias AWS EC2 con ansible

Para poder realizar este esquema deberemos de tener una instancia en **AWS** además de haber copiado la llave pública que nos ofrece para poder conectarnos por ssh a dicha instancia.

**Create key pair**

**Key pair**  
A key pair, consisting of a private key and a public key, is a set of security credentials that you use to prove your identity when connecting to an instance.

Name  
  
The name can be up to 255 characters long. Valid characters include \_, -, a-z, A-Z, and 0-9.

File format  
☒ pem  
For use with OpenSSH  
☐ ppk  
For use with PuTTY

Cancel Create key pair

Por eso crearemos un script yml para que se cree una instancia en aws, para poder crear la instancia necesitaremos los parametros **aws\_access\_key** y **aws\_secret\_key**.

Para ello nos vamos al apartado IAM Management Console en AWS y creamos un nuevo usuario con permisos AdministratorAccess. Y creamos una nueva **Credencial**, que descargaremos en formato **.csv**



Para aprovisionar una instancia EC2, deberemos descargar con pip los siguientes paquetes: **boto** y **boto3**.

```
root@HLCserver:~# pip install boto boto3
Collecting boto
  Downloading https://files.pythonhosted.org/packages/23/10/c0b78c27298029e4454a472a1919bde20cb182dab1662cec7f2ca1dcc523/boto-2.49.0-py2.py3-n
  100% |#####| 1.4MB 176kB/s
Collecting boto3
  Downloading https://files.pythonhosted.org/packages/18/62/2b271ebf56d10e6a85eb7ce34c231909466ebf67d014d1498b6ef64aa27f/boto3-1.12.15-py2.py3
  100% |#####| 133kB 9.9kB/s
Collecting botocore<1.16.0,>=1.15.15 (from boto3)
  Downloading https://files.pythonhosted.org/packages/ee/83/fee1d0c24cfd38fd683e43e41f91756ba01a648c7a9fe2e3d370223ced94/botocore-1.15.15-py2.
  100% |#####| 5.9MB 82kB/s
Collecting jmespath<1.0.0,>=0.7.1 (from boto3)
  Downloading https://files.pythonhosted.org/packages/a3/43/1e939e1fcd87b827fe192d0c9fc25b48c5b3368902bfb913de7754b0dc03/jmespath-0.9.5-py2.py
  100% |#####| 71kB 470kB/s
Collecting s3transfer<0.4.0,>=0.3.0 (from boto3)
  Downloading https://files.pythonhosted.org/packages/69/79/e6afb3d8b0b4e96cefbdc690f741d7dd24547ff1f94240c997a26fa908d3/s3transfer-0.3.3-py2.
  100% |#####| 71kB 470kB/s
Collecting python-dateutil<3.0.0,>=2.1 (from botocore<1.16.0,>=1.15.15->boto3)
```

Creamos nuestro script de Ansible para desplegar la infraestructura.

Con él, desplegaremos todos los equipos que pondremos en nuestro fichero de hosts y crearemos un grupo de seguridad con permisos de conexión mediante SSH y vía HTTP.

```

ansible+docker > aws-docker > ! ec2.yml
1  ---
2  - name: Aprovisionar instancia EC2
3    hosts: ec2
4    connection: local
5    gather_facts: False
6    tags: provisioning
7    tasks:
8      - include_vars: vars/instancia.yml
9
10     - name: Crear grupo de seguridad
11       local_action:
12         module: ec2_group
13         name: "{{ security_group }}"
14         description: Grupo de seguridad
15         region: "{{ region }}"
16         aws_access_key: "{{ aws_access_key }}"
17         aws_secret_key: "{{ aws_secret_key }}"
18         rules:
19           - proto: tcp
20             from_port: 22
21             to_port: 22
22             cidr_ip: 0.0.0.0/0
23           - proto: tcp
24             from_port: "{{ puerto_externo }}"
25             to_port: "{{ puerto_interno }}"
26             cidr_ip: 0.0.0.0/0
27         rules_egress:|
28           - proto: all
29             cidr_ip: 0.0.0.0/0
30         register: basic_firewall

```

Finalmente lanzamos todas las instancias y nos devuelve la dirección IP pública de todas ellas, para poder conectarnos posteriormente.

```

32     - name: Lanzar instancia EC2
33       ec2:
34         aws_access_key: "{{ aws_access_key }}"
35         aws_secret_key: "{{ aws_secret_key }}"
36         group: "{{ security_group }}"
37         instance_type: "{{ instance_type }}"
38         image: "{{ image }}"
39         wait: true
40         region: "{{ region }}"
41         keypair: "{{ keypair }}"
42         count: "{{ count }}"
43       register: ec2
44
45     - name: Obtener variable EC2
46       debug: var=ec2
47     - name: Obtener dirección Ip
48       debug: var=ec2.instances[0].public_ip

```

Estas son las variables de nuestro despliegue, seleccionando que crearemos instancias micro. En ella deberemos configurar el nombre de nuestra clave que creamos en amazon y también de los códigos de acceso.

En la región, configuramos en que servidor de AWS se creará, en este caso indica que es en el de Sídney y con una imagen de Ubuntu 18 Server.

```
ansible+docker > aws-docker > vars > ! instancia.yml
1  instance_type: t2.micro
2  security_group: grupo_seguridad
3  image: ami-02a599eb01e3b3c5b
4  keypair: AWS-sidney
5  region: ap-southeast-2
6  hostname: wordpress
7  aws_access_key: [REDACTED]
8  aws_secret_key: [REDACTED]
9  puerto_externo: 80
10 puerto_interno: 8080
11 count: 1
```

Y también tendremos nuestro fichero de instancias, al configurar dos nombres, se creará una instancia.

```
ansible+docker > aws-docker > ≡ instancias
1  [ec2]
2  wordpress
3  |
```

Lanzamos el playbook y se creará el grupo de seguridad y la instancia.

```
root@HLServer:~/ansible+docker/aws-docker# ansible-playbook -i instancias ec2.yml -vvv
ansible-playbook 2.9.4
  config file = /etc/ansible/ansible.cfg
  configured module search path = [u'/root/.ansible/plugins/modules', u'/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python2.7/dist-packages/ansible
  executable location = /usr/bin/ansible-playbook
  python version = 2.7.13 (default, Sep 26 2018, 18:42:22) [GCC 6.3.0 20170516]
Using /etc/ansible/ansible.cfg as config file
host_list declined parsing /root/ansible+docker/aws-docker/instancias as it did not pass its verify_file() method
script declined parsing /root/ansible+docker/aws-docker/instancias as it did not pass its verify_file() method
auto declined parsing /root/ansible+docker/aws-docker/instancias as it did not pass its verify_file() method
Parsed /root/ansible+docker/aws-docker/instancias inventory source with ini plugin

PLAYBOOK: ec2.yml *****
1 plays in ec2.yml

PLAY [Aprovisionar instancia EC2] *****
META: ran handlers
```

Finalmente obtendremos una salida con todos los datos de nuestra instancia en formato de variables.

```

TASK [Obtener variable EC2] *****
task path: /root/ansible+docker/aws-docker/ec2.yml:45
ok: [wordpress] => {
  "ec2": {
    "ansible_facts": {
      "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": true,
    "failed": false,
    "instance_ids": [
      "i-0d4fbab6c64658846"
    ],
    "instances": [
      {
        "ami_launch_index": "0",
        "architecture": "x86_64",
        "block_device_mapping": {
          "/dev/sda1": {
            "delete_on_termination": true,
            "status": "attached",
            "volume_id": "vol-0b93b7f1f1f4e40eb"
          }
        },
        "dns_name": "ec2-13-210-23-147.ap-southeast-2.compute.amazonaws.com",
        "ebs_optimized": false,
        "groups": {
          "sg-089f25ee1e66c0df1": "grupo_seguridad"
        },
        "hypervisor": "xen",
        "id": "i-0d4fbab6c64658846",
        "image_id": "ami-02a599eb01e3b3c5b",
        "instance_type": "t2.micro",
        "kernel": null,
        "key_name": "AWS-sidney",
        "launch_time": "2020-03-06T20:16:31.000Z",
        "placement": "ap-southeast-2a",
        "private_dns_name": "ip-172-31-3-141.ap-southeast-2.compute.internal",
        "private_ip": "172.31.3.141",
        "public_dns_name": "ec2-13-210-23-147.ap-southeast-2.compute.amazonaws.com",
        "public_ip": "13.210.23.147",
        "ramdisk": null,
        "region": "ap-southeast-2",
        "root_device_name": "/dev/sda1",
        "root_device_type": "ebs",
        "state": "running",
        "state_code": 16,
        "tags": {},
        "tenancy": "default",
        "virtualization_type": "hvm"
      }
    ],
    "tagged_instances": [],
  }
}

```

Y con estas variables hemos definidos que al final se nos muestre la IP pública de nuestra instancia.

```

TASK [Obtener dirección Ip] *****
task path: /root/ansible+docker/aws-docker/ec2.yml:47
ok: [wordpress] => {
  "ec2.instances[0].public_ip": "54.252.151.117"
}
META: ran handlers
META: ran handlers

PLAY RECAP *****
wordpress      : ok=5    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

```

Si accedemos a nuestro panel web de AWS, veremos nuestra instancia.

Launch Instance

Connect

Actions

Filter by tags and attributes or search by keyword

	Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS (IPv4)	IPv4 Public IP
		i-0d4fbab6c64658846	t2.micro	ap-southeast-2a	running	2/2 checks ...	None	ec2-13-210-23-147.ap-...	13.210.23.147

## Aprovisionar Instancia

Para poder aprovisionar esta instancia con ansible deberá de tener **Python** por lo tanto instalaremos Python con el comando ssh.

```
root@HLCserver:~# ssh -i ~/.ssh/AWS.pem ubuntu@18.221.133.81 "apt install -y python"
```

Una vez hecho el paso anterior lo que haremos será descargarnos los roles de pip y docker con ayuda del comando **ansible-galaxy**.

```
root@HLCserver:~/ansible+docker/aws-docker# ansible-galaxy install geerlingguy.pip
```

```
root@HLCserver:~/ansible+docker/aws-docker# ansible-galaxy install geerlingguy.docker
```

Esta es la estructura de carpetas que ha quedado después de realizar este esquema.

```
root@HLCserver:~/ansible+docker/aws-docker# tree
.
├── aws
├── playbook.yml
├── roles
│   └── joseadiaz.docker.py
│       ├── defaults
│       │   └── main.yml
│       ├── files
│       ├── handlers
│       │   └── main.yml
│       ├── meta
│       │   └── main.yml
│       ├── README.md
│       ├── tasks
│       │   └── main.yml
│       └── templates
├── tasks
│   ├── mysql.yml
│   └── wordpress.yml
└── vars
    ├── mysql.yml
    └── wordpress.yml
```

En la carpeta roles lo que tendremos será un rol que se ha creado específicamente para poder instalar Python-setuptools y docker-py.

**Python-setuptools** es una librería que necesitaremos para instalar con pip en ansible.

**Docker-py** es un módulo de ansible que se utiliza para administrar contenedores, entre otras cosas como crear, eliminar, etc.

**./roles/task/main.yml**



```
ansible+docker > aws-docker > roles > joseadiaz.docker.py > tasks > ! main.yml
1  ---
2  - name: Install python-setuptools
3    apt:
4      name: python-setuptools
5
6  - name: Install docker pip
7    pip:
8      name: docker-py
```

En el archivo **playbook.yml** lo que ha cambiado respecto al proyecto de wordpress descrito en otros apartados es que se ha añadido unos roles que son dos de docker, pip y el creado por mi descrito anteriormente.

**./playbook.yml**

```
ansible+docker > aws-docker > ! playbook.yml
1  ---
2  - name: Creando docker instancia AWS
3    hosts: aws
4    become: true
5    roles:
6      - geerlingguy.docker
7      - geerlingguy.pip
8      - role: joseadiaz.docker.py
9    tasks:
10     - include_vars: vars/mysql.yml
11     - include_vars: vars/wordpress.yml
12     - name: Create network
13       docker_network:
14         name: prueba
15         ipam_options:
16           subnet: 192.168.100.0/24
17         driver_options:
18           com.docker.network.bridge.name: prueba
19     - include: tasks/mysql.yml
20     - include: tasks/wordpress.yml
21
```

En este archivo y en el de wordpress poco hay que hablar porque básicamente lo que ha cambiado es que se ha tenido que añadir un nuevo apartado para poder bajar la imagen con la que se creará el contenedor dado que esta imagen no está en la instancia de AWS.

Este apartado tiene una variable que se definirá en los archivos de la carpeta **/vars**.

./task/mysql.yml

```
---
- name: pull imagen
  docker_image:
    name: "{{ image_mysql }}"
    source: pull

- name: Desplegar contenedor base de datos
  docker_container:
    name: "{{ container_name_mysql }}"
    image: "{{ container_mysql_image }}"
    volumes:
      - "{{ container_mysql_volume }}:/var/lib/mysql"
    ports:
      - "{{ container_port_publish_mysql }}:{{ container_port_private_mysql }}"
    env:
      MYSQL_ROOT_PASSWORD: "{{ password_root_mysql }}"
      MYSQL_DATABASE: "{{ database_mysql }}"
      MYSQL_USER: "{{ user_mysql }}"
      MYSQL_PASSWORD: "{{ password_user_mysql }}"
    networks:
      - name: prueba
        ipv4_address: 192.168.100.2
    state: started
    restart: yes
```

En el archivo wordpress.yml hemos creado el mismo apartado que el mysql.conf.

./task/wordpress.yml

```
---
- name: pull imagen
  docker_image:
    name: "{{ image_wordpress }}"
    source: pull

- name: Desplegar contenedor WordPress
  docker_container:
    name: "{{ container_name_wordpress }}"
    image: "{{ container_wordpress_image }}"
    volumes:
      - "{{ container_mysql_volume }}:/var/www/html"
    ports:
      - "{{ container_http_port_publish_wordpress }}:{{ container_http_port_private_wordpress }}"
    env:
      WORDPRESS_DB_HOST: "{{ db_host }}"
      WORDPRESS_DB_USER: "{{ user_mysql }}"
      WORDPRESS_DB_PASSWORD: "{{ password_user_mysql }}"
      WORDPRESS_DB_NAME: "{{ database_mysql }}"
    networks:
      - name: prueba
        ipv4_address: 192.168.100.3
    state: started
    restart: yes
```

En los dos archivos que hay en la carpeta **/vars** solamente se ha añadido una cosa nueva y es la variable **image\_\*** para poder poner el nombre de la imagen que se descargará de dockerhub.

**./vars/mysql.yml**

```
ansible+docker > aws-docker > vars > ! mysql.yml
1  image_mysql: "mysql"
2  container_name_mysql: "mysql"
3  container_mysql_image: "mysql:5.7"
4  container_mysql_volume: "/mysql"
5  password_root_mysql: "bolson"
6  database_mysql: "wordpress"
7  user_mysql: "wordpress"
8  password_user_mysql: "wordpress"
9  container_port_publish_mysql: "3306"
10 container_port_private_mysql: "3306"
11
```

**./vars/wordpress.yml**

```
ansible+docker > aws-docker > vars > ! wordpress.yml
1  image_wordpress: "wordpress"
2  container_name_wordpress: "WordPress"
3  container_wordpress_image: "wordpress:latest"
4  container_wordpress_volume: "/wordpress"
5  container_http_port_publish_wordpress: "8080"
6  container_http_port_private_wordpress: "80"
7  db_host: "mysql"
8
9
```

Además se ha creado un inventario donde se ha definido un grupo **aws**, este grupo contendrá la IP y el usuario de la instancia para poder aprovisionarla, en este caso el usuario es **ubuntu** porque nuestra instancia tiene como SO Ubuntu.

**./aws**

```
ansible+docker > aws-docker > vars > ! aws
1  [aws]
2  aws ansible_host=18.188.12.163 ansible_user=ubuntu
3
```

Una vez creados todos los archivos y descargados los roles solo quedará ejecutar el comando **ansible-playbook** para poder aprovisionar nuestra instancia, en este comando se le pasará el **playbook.yml** además del **inventario** con la opción **-i** y la **llave pública** de ssh para poder conectarnos a la instancia.

```
root@HLCserver:~/ansible+docker/aws-docker# ansible-playbook playbook.yml -i aws --key-file=~/.ssh/AWS.pem
[WARNING]: Found both group and host with same name: aws

PLAY [Creando docker instancia AWS] *****
```

Se comprueba que el resultado ha sido exitoso.

```
changed: [aws]
TASK [pull imagen] *****
changed: [aws]
TASK [Desplegar contenedor WordPress] *****
changed: [aws]
PLAY RECAP *****
aws : ok=23  changed=3  unreachable=0  failed=0  skipped=7  rescued=0  ignored=0
```

Además entraremos a la instancia por ssh y comprobaremos que están corriendo los dos contenedores que se le ha pedido que se creen.

```
ubuntu@DevOps:~$ sudo docker ps
CONTAINER ID   IMAGE                COMMAND                  CREATED        STATUS        PORTS                               NAMES
3e032b4f0a7c   wordpress:latest    "docker-entrypoint.s..." 36 minutes ago Up 25 seconds 0.0.0.0:8080->80/tcp               WordPress
6399eec59f6e   mysql:5.7           "docker-entrypoint.s..." 38 minutes ago Up 33 seconds  0.0.0.0:3306->3306/tcp, 33060/tcp   mysql
```

Una vez comprobado esto solo nos quedará comprobar que nuestro wordpress está funcionando para ello deberemos de ingresar en un navegador y poner la IP de nuestra instancia y el puerto 8080 que es el que le hemos dicho que se exponga al exterior.

Pero esto no funcionará sin antes definir una regla de entrada en el grupo de seguridad de nuestra instancia.

Para ello nos iremos a **Security Groups**.

EC2 > Security Groups

Security Groups (1/2) Info

Filter security groups

	Security group ID	Security group name	VPC ID	Description	Owner	Inbound rules count	Outbound rules count
<input type="checkbox"/>	sg-003b3277f5c906716	default	vpc-0887410e8dbb20b71 ...	default VPC security gr...	915508415579	1 Permission entry	1 Permission entry
<input checked="" type="checkbox"/>	sg-0fee9c5aacd5021d0	launch-wizard-1	vpc-0887410e8dbb20b71 ...	launch-wizard-1 create...	915508415579	2 Permission entries	1 Permission entry

Aquí solo tendremos que definir que permita entrar a todas las direcciones IP que entren por el puerto **8080**.


Inbound rules

Edit inbound rules

Type	Protocol	Port range	Source	Description - optional
Custom TCP	TCP	8080	0.0.0.0/0	-
SSH	TCP	22	0.0.0.0/0	-

Una vez hecho esto ya podremos ingresar nuestra IP y el puerto 8080 en el navegador, comprobaremos que nuestro wordpress está funcionando sin problema alguno.

No es seguro | 18.221.133.81:8080/wp-admin/install.php?step=1



## Hola

¡Bienvenido al famoso proceso de instalación de WordPress en cinco minutos! Simplemente completa la información siguiente y estarás a punto de usar la más enriquecedora y potente plataforma de publicación personal del mundo.

### Información necesaria

Por favor, debes facilitarnos los siguientes datos. No te preocupes, siempre podrás cambiar estos ajustes más tarde.

**Título del sitio**

**Nombre de usuario**

Los nombres de usuario pueden tener únicamente caracteres alfanuméricos, espacios, guiones bajos, guiones medios, puntos y el símbolo @.

**Contraseña**  [Ocultar](#)

**Fuerte**

**Importante:** Necesitas esta contraseña para acceder. Por favor, guárdala en un lugar seguro.

**Tu correo electrónico**

Comprueba bien tu dirección de correo electrónico antes de continuar.

**Visibilidad en los motores de búsqueda** ☐ Disuade a los motores de búsqueda de indexar este sitio

Depende de los motores de búsqueda atender esta petición o no.

[Instalar WordPress](#)

## WEBGRAFÍA

<https://riptutorial.com/es/ansible/example/11343/como-iniciar-la-instancia-de-ec2-desde-las-ami-oficiales-de-amazon--modificarla-y-almacenarla-como-nueva-ami>

[https://docs.ansible.com/ansible/latest/modules/ec2\\_module.html](https://docs.ansible.com/ansible/latest/modules/ec2_module.html)

<http://icewinddale.blogspot.com/2018/03/desplegar-maquinas-en-amazon-ec2-con.html>

<https://galaxy.ansible.com/geerlingguy/docker>

<https://galaxy.ansible.com/geerlingguy/pip>

[https://docs.ansible.com/ansible/latest/modules/docker\\_image\\_module.html](https://docs.ansible.com/ansible/latest/modules/docker_image_module.html)