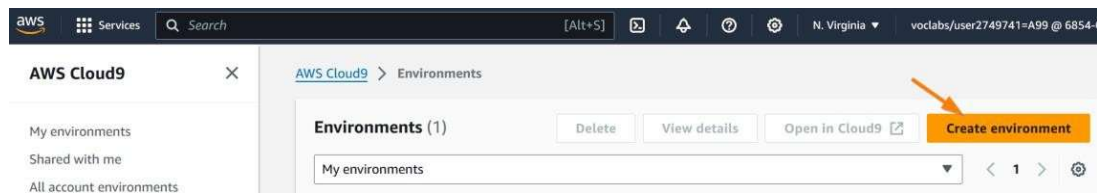
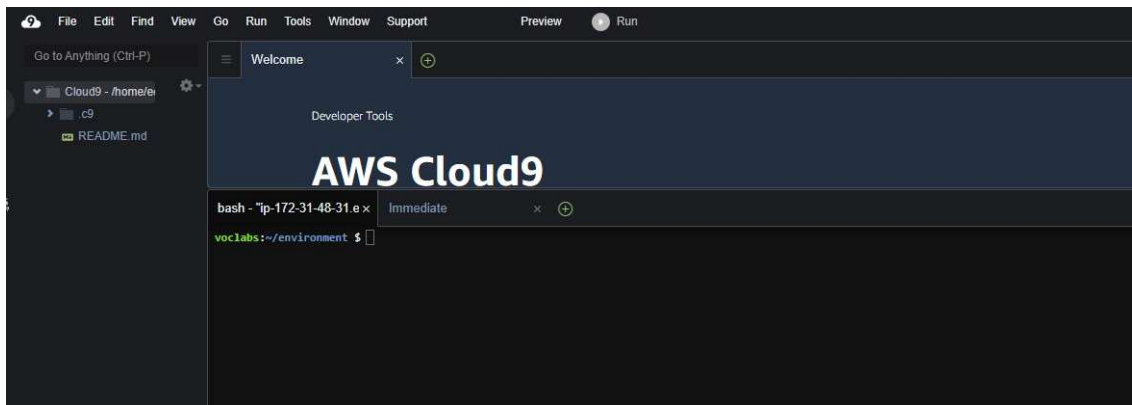


Práctica 7.1: Servidor web sobre contenedor (Despliegue).

A. Preparación...

1. Preparación de entorno Cloud9:

- a) Desde su laboratorio, acceda a Cloud9 y prepare un entorno de desarrollo “environment” o “entorno según el idioma que tengáis AWS.



- b) Utilice la siguiente configuración:

- Nombre: Puede indicar el que desee, como por ejemplo “Cloud9-P7.1”
- Environment type: New EC2 instance
- Instance Type: Con una t2.micro es suficiente.
- Platform: Amazon Linux 2023
- Timeout: 1 hora (puede indicar el que desee)
- Connection (en Network Settings): Secure Shell (SSH), ya que la opción por defecto no está disponible en el learner lab.

☒ t2.micro (1 GiB RAM + 1 vCPU)
Free-tier eligible. Ideal for educational users and exploration.

☐ t3.small (2 GiB RAM + 2 vCPU)
Recommended for small web projects.

☐ m5.large (8 GiB RAM + 2 vCPU)
Recommended for production and most general-purpose development.

☐ Additional instance types
Explore additional instances to fit your need.

Platform [Info](#)
This will be installed on your EC2 instance. We recommend Amazon Linux 2023.

Amazon Linux 2023

Timeout
How long Cloud9 can be inactive (no user input) before auto-hibernating. This helps prevent unnecessary charges.

4 hours

Connection and VPC

Connection
Secure Shell (SSH)

Amazon Virtual Private Cloud (VPC)
vpc-0a1a3bcc658bf3eb4

Subnet
subnet-0ab560e2396af1bbc

c) Cuando la instancia esté preparada (puede tardar unos minutos), se puede acceder:

AWS Cloud9 > Environments

Environments (1)

Delete

View details

Open in Cloud9

Create environment

My environments

	Name	Cloud9 IDE	Environment type	Connection	Permission	Owner ARN
<input type="radio"/>	Cloud9	Open	EC2 instance	Secure Shell (SSH)	Owner	arn:aws:sts::730335544032:assumed-role/voclabs/user3027979=consuelo.moya@educa.madrid.org

d) Opcional: Una vez conectado, compruebe la versión de Docker, Git y Python instalado en el entorno (p.e. "docker --versión").

```
bash - "ip-172-31-48-31.e x Immediate
voclabs:~/environment $ docker --version
Docker version 24.0.5, build ced0996
voclabs:~/environment $ git --version
git version 2.40.1
```

```
voclabs:~/environment $ sudo yum -y install python3
Last metadata expiration check: 1:43:40 ago on Mon Mar 18 07:27:38 2024.
Package python3-3.9.16-1.amzn2023.0.6.x86_64 is already installed.
Dependencies resolved.
Nothing to do.
Complete!
```

```
bash - "ip-172-31-29-147. x Immediate x (+)
voclabs:~/environment $ docker --version
Docker version 25.0.13, build 0bab007
voclabs:~/environment $
```

Con Python me daba error lo de versión así que probe como a instalarlo (realmente ya estaba instalado) y ahí se puede ver la version

2. Lanzamiento de contenedor local con sitio web sencillo.

- En Cloud9, cree una carpeta llamada practica (en mi caso). Dentro de ella, descargue el archivo ipcalc2.zip y descomprímalo. Una vez descomprimido, obtendrá una carpeta con el siguiente contenido:

```
AWS Cloud9
bash - "ip-172-31-48-31.e x Immediate x (+)
voclabs:~/environment $ docker --version
Docker version 24.0.5, build ced0996
voclabs:~/environment $ mkdir practica
voclabs:~/environment $ cd practica/
```

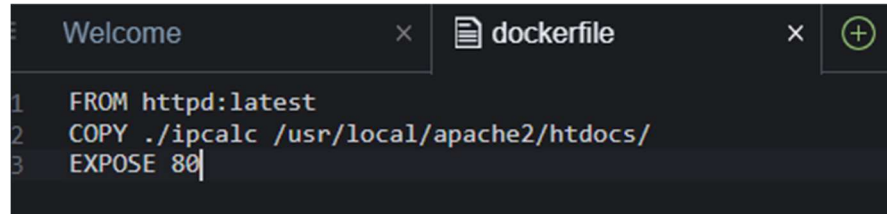
Nota: Si lo desea puede crear la imagen de contenedor con su propia aplicación web, pero en ese caso deberá generar el fichero Dockerfile. Lo veremos mas adelante en la práctica.

La estructura de directorios en cloud9 para nuestra aplicación es la siguiente:

```
▼ practica
  > ipcalc
  dockerfile
  ipcalc2.zip
```

El Dockerfile que creamos tiene esta forma y se guarda en el directorio /practica.

```
FROM httpd:latest
COPY ./ipcalc /usr/local/apache2/htdocs/
EXPOSE 80
```



- b) Acceda a la carpeta (practica/ipcalc) y construya el contenedor “ipcalc” a partir del fichero Dockerfile incluido. ****Cambia la captura de pantalla por la que corresponde a la creación del contenedor ipcalc.****

```
voclabs:~/environment/Practica $ docker build -t 2048 .
[+] Building 4.5s (7/7) FINISHED          docker:default
=> [internal] load build definition from dockerfile      0.0s
=> => transferring dockerfile: 164B                    0.0s
=> [internal] load metadata for docker.io/library/httpd:latest 0.5s
=> [internal] load .dockerignore                        0.0s
=> => transferring context: 2B                          0.0s
=> [internal] load build context                        0.0s
=> => transferring context: 51.33kB                     0.0s
=> [1/2] FROM docker.io/library/httpd:latest@sha256:b913eada26 3.8s
=> => resolve docker.io/library/httpd:latest@sha256:b913eada26 0.0s
=> => sha256:b913eada2685f101f93267e09841099 10.14kB / 10.14kB 0.0s
=> => sha256:34a6173615f337f37833f66be70dce30f 2.09kB / 2.09kB 0.0s
=> => sha256:627c383437d53d53f027b45edd5b6f6113 7.98kB / 7.98kB 0.0s
```

- c) Mediante “Docker images” compruebe cuáles son los contenedores disponibles en su sistema en este momento. ****Cambia la captura de pantalla se debe ver que se ha creado el contenedor ipcalc****

```
voclabs:~/environment/Practica $ docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
2048          latest   17728ac48e78   About a minute ago   117MB
voclabs:~/environment/Practica $
```

- d) Ejecute el contenedor 2048 de forma que el puerto 80 del contenedor se exponga en el puerto 30000 de la instancia Cloud9.

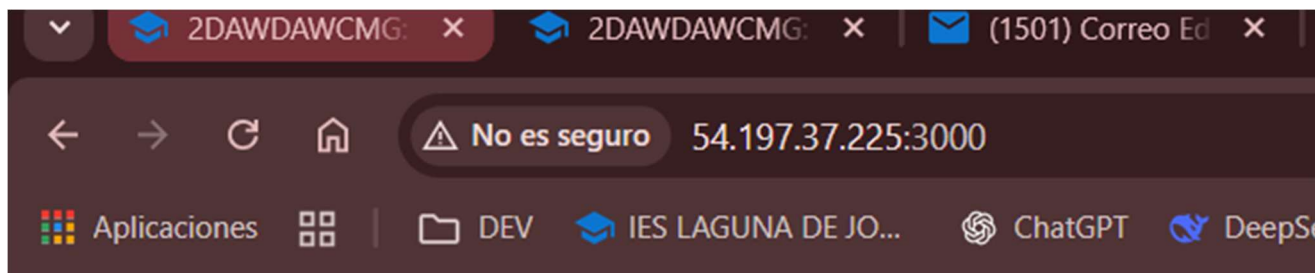
`docker run -d -p 30000:80 ipcalc` ****Cambia por tu captura****

```
voclabs:~/environment/Practica $ docker run --name 2048a -d -p 3000:80
2048
c38b15073f7065746b6705acb65a73176b25550343057b3028823c619ffd5327
voclabs:~/environment/Practica $
```

e) Mediante netstat compruebe que su instancia EC2 expone el puerto 30.000:

```
voclabs:~/environment/Practica $ sudo netstat -ltnp
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      2130/sshd: /usr/sbi
tcp        0      0 127.0.0.1:40609         0.0.0.0:*               LISTEN      1979/containerd
tcp        0      0 0.0.0.0:3000            0.0.0.0:*               LISTEN      4285/docker-proxy
tcp6       0      0 :::22                   :::*                     LISTEN      2130/sshd: /usr/sbi
tcp6       0      0 :::3000                  :::*                     LISTEN      4294/docker-proxy
voclabs:~/environment/Practica $
```

f) Abra el puerto 30.000 (**En clase, lo hicimos con el puerto 3000) en el Grupo de Seguridad asociado a la instancia Cloud9. Utilizando la IP pública de la instancia, compruebe en un navegador que el sitio web es accesible en Internet. ****Cambia por tu captura****



It works!

Inbound rules (3)							
Q Search							
<input type="checkbox"/>	Name	Security group rule ID	IP version	Type	Protocol	Port range	
<input type="checkbox"/>	-	sgr-08e040269b896a4a5	IPv4	SSH	TCP	22	
<input type="checkbox"/>	-	sgr-0b9da5bba73e25d1a	IPv4	SSH	TCP	22	
<input type="checkbox"/>	-	sgr-04ad9a9f5fa3ca857	IPv4	Custom TCP	TCP	3000	

- g) A partir de este momento ya no se va a utilizar el contenedor que se está ejecutando en la instancia cloud9, por lo que puede detenerlo. ****Para detener un contenedor el comando stop seguido del ID del container**** Cambia la captura por la correspondiente**

```
voclabs:~/environment/Practica $ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS
PORTS         NAMES
c38b15073f70   2048          "httpd-foreground"      20 minutes ago Up 20 m
0.0.0.0:3000->80/tcp, :::3000->80/tcp 2048a
voclabs:~/environment/Practica $ docker stop c38b15073f70
c38b15073f70
voclabs:~/environment/Practica $
```

Para la **aplicación uxos** todo lo que hemos hecho antes para la aplicación del juego se haría del siguiente modo

```
voclabs:~/environment/practica/2048 $ cd ..
voclabs:~/environment/practica $ cd ..
voclabs:~/environment $ mkdir uxos
voclabs:~/environment $ cd uxos
```

Copiaremos el código de la aplicación en /uxos, y descomprimiremos, en cloud9 hay una forma fácil de llevar el fichero comprimido simplemente arrastrando y soltando.

```
voclabs:~/environment/uxos $ docker build -t uxosweb .
[+] Building 0.4s (7/7) FINISHED          docker:default
=> [internal] load build definition from Dockerfile 0.0s
=> => transferring dockerfile: 162B 0.0s
=> [internal] load metadata for docker.io/library/httpd:latest 0.1s
=> [internal] load .dockerignore 0.0s
=> => transferring context: 2B 0.0s
=> [internal] load build context 0.1s
=> => transferring context: 1.19MB 0.0s
=> CACHED [1/2] FROM docker.io/library/httpd:latest@sha256:b91 0.0s
=> [2/2] COPY ./uxos /usr/local/apache2/htdocs/ 0.1s
=> exporting to image 0.0s
=> => exporting layers 0.0s
=> => writing image sha256:54d8c66eea4e2f999fef0573fd270a64512 0.0s
```

```
voclabs:~/environment/uxos $ docker run --name uxosweb -d -p 3001:80 u
xosweb
20cb257e8f7a57af358a51681b5e5f7c08680c18bb1aff4486d1e4b51d055169
voclabs:~/environment/uxos $
```

La aplicación uxos está localizada en apache2 y se crea la imagen de contenedor (copia de la aplicación web que está en apache) y se pone el puerto de escucha 80.

En el grupo de seguridad de la instancia hemos creado una regla de entrada del puerto 3001 para poder desplegar la aplicación uxos. Ahora tenemos 2 imágenes: la de uxos y la del juego 2048.

Inbound rules (4) Manage tags Edit inbound

Search

<input type="checkbox"/>	Name	Security group rule ID	IP version	Type	Protocol	Port range
<input type="checkbox"/>	-	sgr-08e040269b896a4a5	IPv4	SSH	TCP	22
<input type="checkbox"/>	-	sgr-0b9da5bba73e25d1a	IPv4	SSH	TCP	22
<input type="checkbox"/>	-	sgr-0d765bc773b3f9b9f	IPv4	Custom TCP	TCP	3001
<input type="checkbox"/>	-	sgr-04ad9a9f5fa3ca857	IPv4	Custom TCP	TCP	3000

```
bash - "ip-172-31-48-31.e x" Immediate x +
=> => extracting sha256:e9304da947c5e9370b81b929a1f90af2a6d0ca3b668618547ea7a2d75c896610
=> => extracting sha256:b60d4b66b268ff32a89991c974dbe2c9d8cbb9deed52983682e69bae9d02b85
=> [2/2] COPY ./uxos /usr/local/apache2/htdocs/
=> exporting to image
=> => exporting layers
=> => writing image sha256:8f5ee9a04e65023c53b7960924bdb5466a9674b3bb112b83dd591c2942f6f7f
=> => naming to docker.io/library/uxos
voclabs~/environment/uxos $ docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
uxos latest 8f5ee9a04e65 9 seconds ago 169MB
2048 latest f3ba103e0368 46 minutes ago 187MB
voclabs~/environment/uxos $ docker run --name uxosweb -d -p 3001:80 uxos
d843ac32470e08341d4a1649d6d808ca6f21c5093a0283ff05df4d360e447251
voclabs~/environment/uxos $
```

UXOS

BEST DESIGN

MOBILE APPLICATIONS

READ MORE

UXOS

Full Name

Phone Number

Email

Password

Submit

Already have an Account? [login here](#)

```
voclabs:~/environment/uxos $ docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS        PORTS                               NAMES
d843ac32470e   uxos      "httpd -foreground"      3 minutes ago  Up 3 minutes  0.0.0.0:3001->80/tcp, :::3001->80/tcp  uxosweb
0d4fe1f74bd9   2048      "/docker-entrypoint..." 46 minutes ago Up 46 minutes  0.0.0.0:3000->80/tcp, :::3000->80/tcp  2048a
voclabs:~/environment/uxos $
```