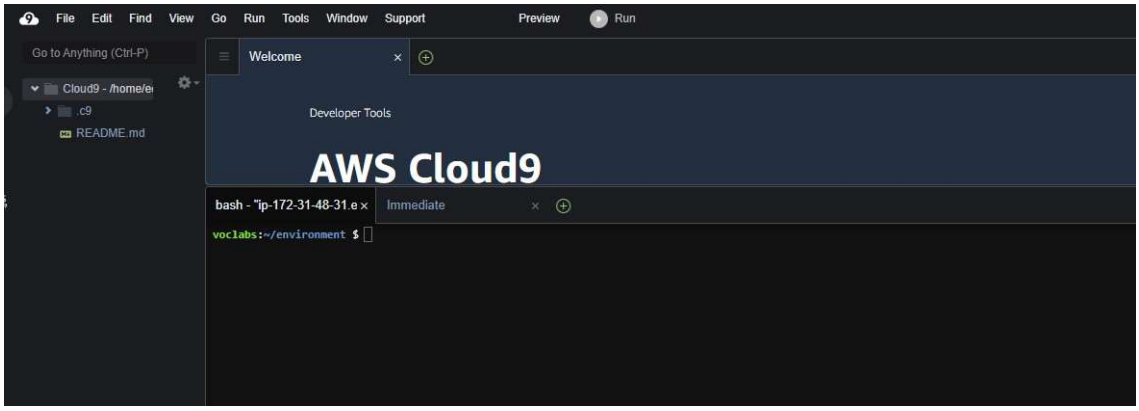
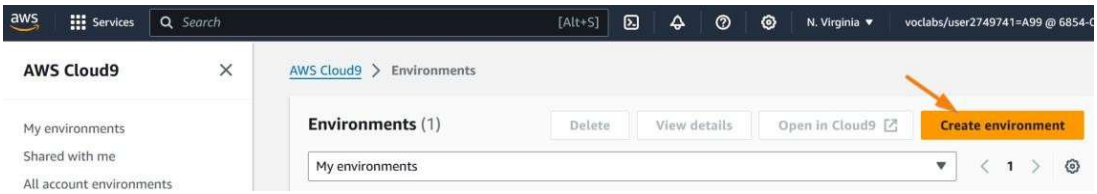
# Práctica 7.1: Servidor web sobre contenedor (Despliegue).

## A. Preparación…

### 1. Preparación de entorno Cloud9:

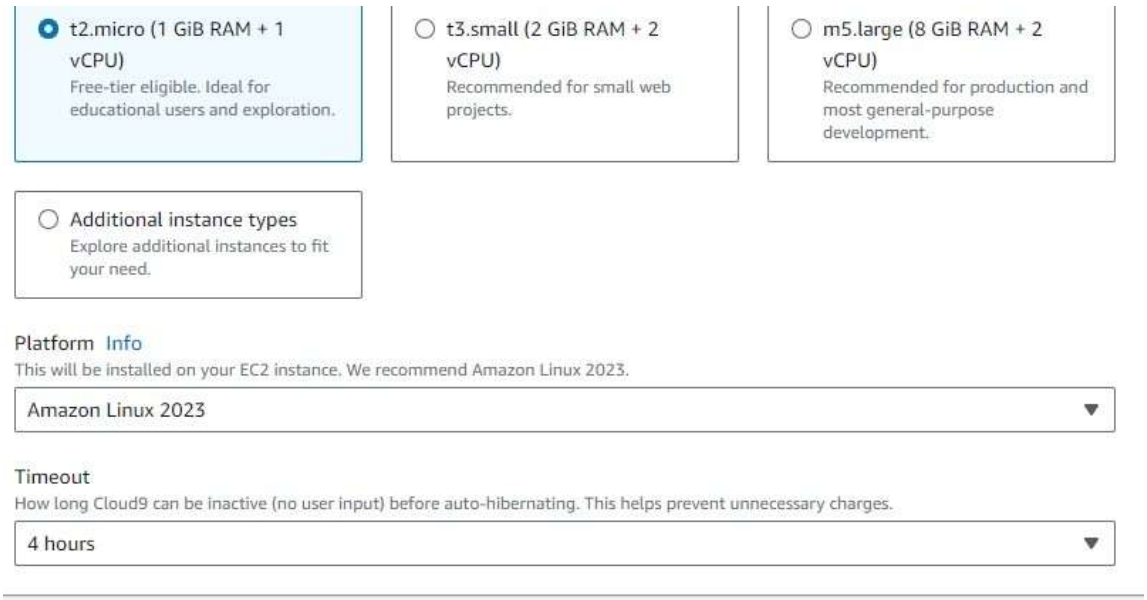
a) Desde su laboratorio, acceda a Cloud9 y prepare un entorno de desarrollo "environment" o "entorno según el idioma que tengáis AWS.



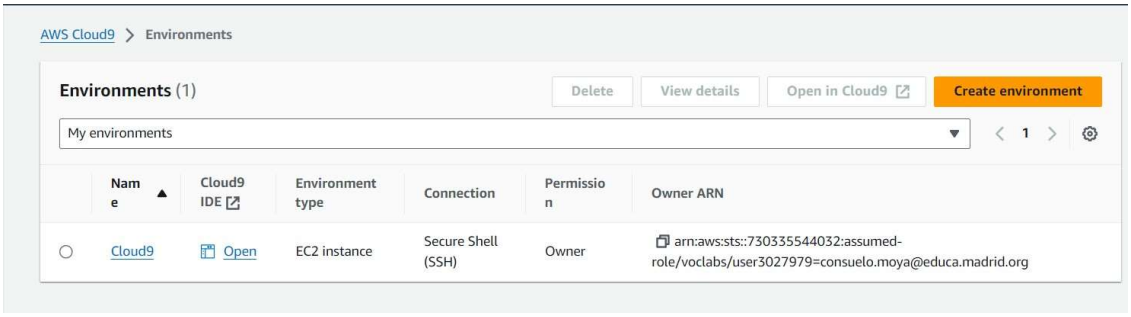

b) Utilice la siguiente configuración:

- 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.



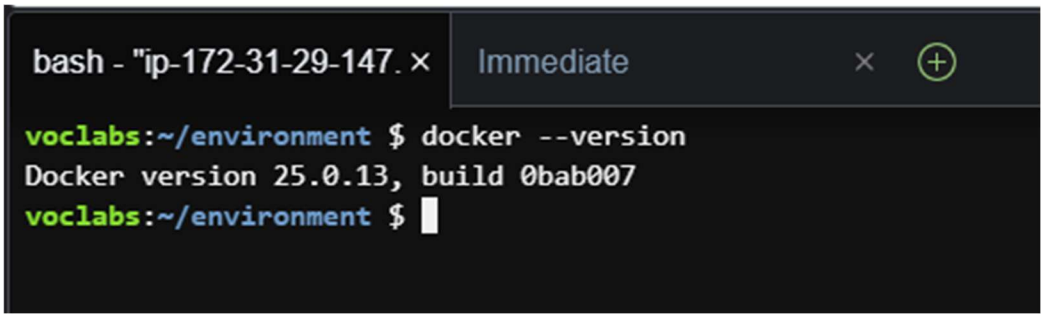


c) Cuando la instancia esté preparada (puede tardar unos minutos), se puede acceder:




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

```
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!
```



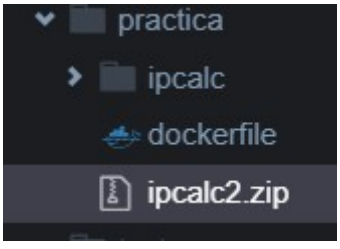

**Con Python me daba error lo de versión asi que probe como a instalarlo (realmente ya estaba instalado) y ahí se puede ver la version**

**2. Lanzamiento de contenedor local con sitio web sencillo.**

a) 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:

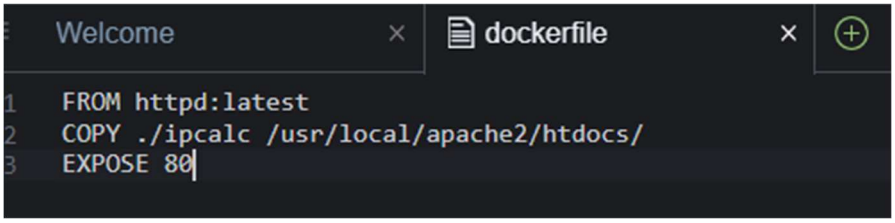

Nota: Si lo desea puede crear la imagen de contenedor con su propia aplicación web, pero en ese caso deberá generar el fichero Dockerfile. Lo veremos mas adelante en la práctica.

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

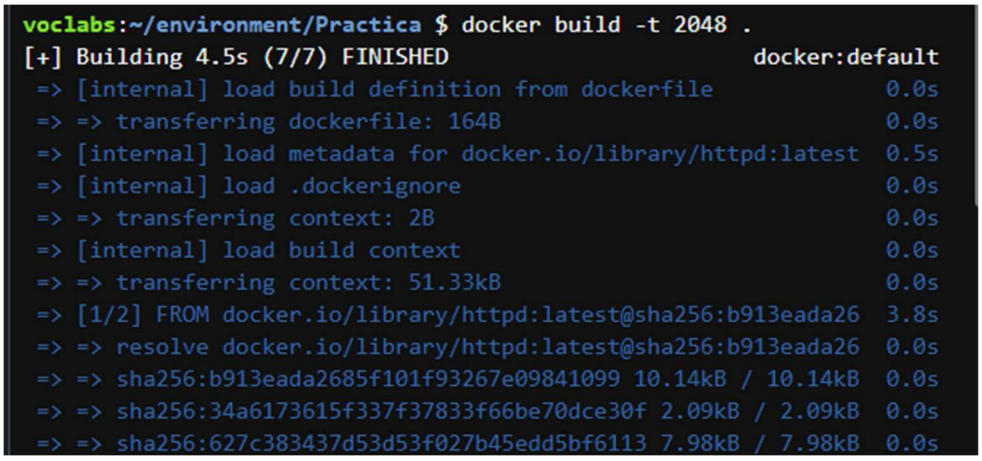


El Dockerfile que creamos tiene esta forma y se guarda en el directorio /practica.

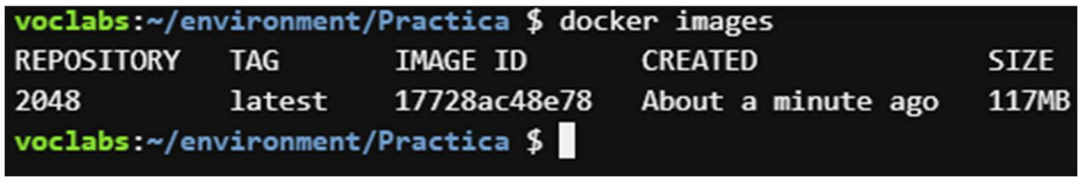
```
FROM httpd:latest
COPY ./ipcalc /usr/local/apache2/htdocs/
EXPOSE 80
```




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.*****

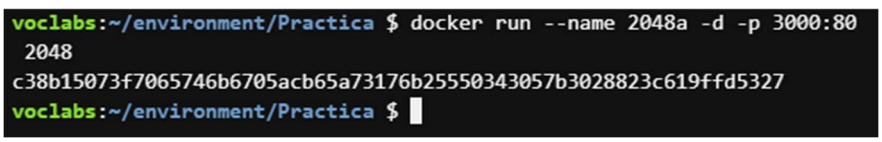



c) Mediante "Docker images" compruebe cuáles son los contenedores disponibles en su sistema en este momento. ****Cambia la captura de pantalla se debe ver que se ha creado el contenedor ipcalc*****




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

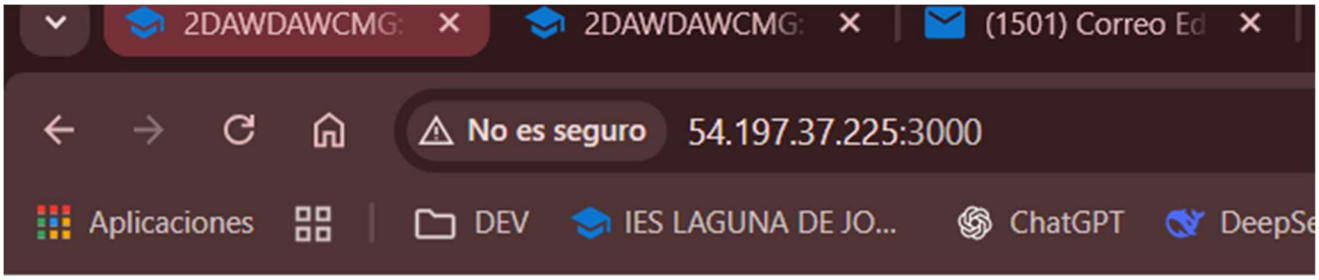
`docker run -d -p 30000:80 ipcalc` ****Cambia por tu captura****



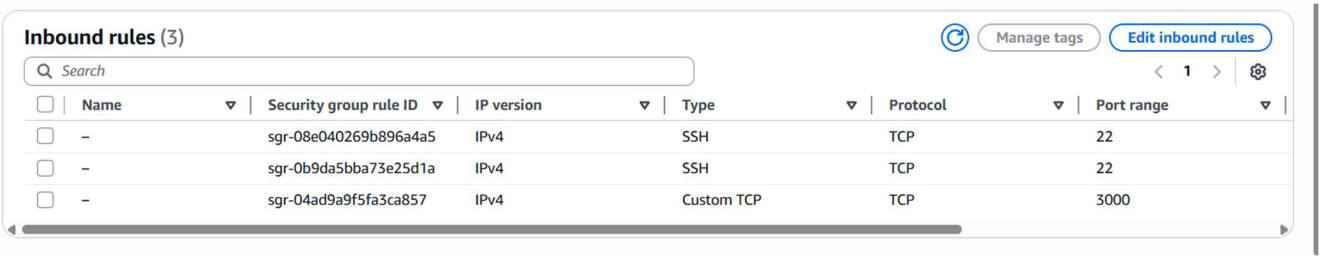
e) Mediante netstat compruebe que su instancia EC2 expone el puerto 30.000:

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



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
inutes   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
```

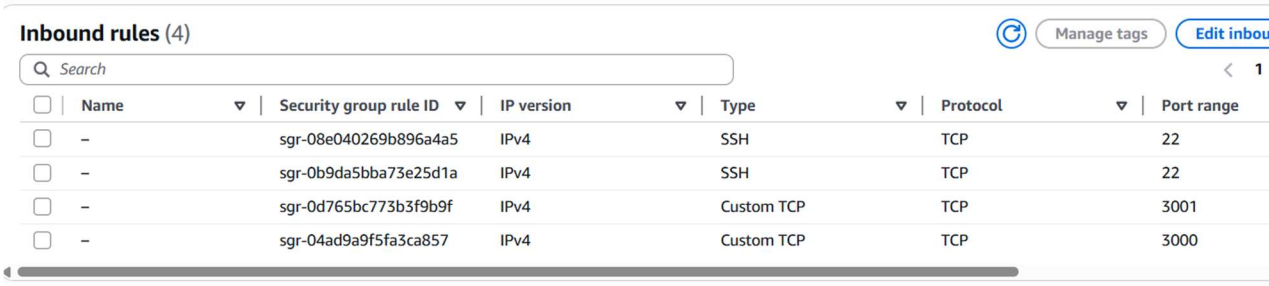
Copiaremos el código de la aplicación en /uxos, y descomprimiremos, en cloud9 hay una forma fácil de llevar el fichero comprimido simplemente arrastrando y soltando.

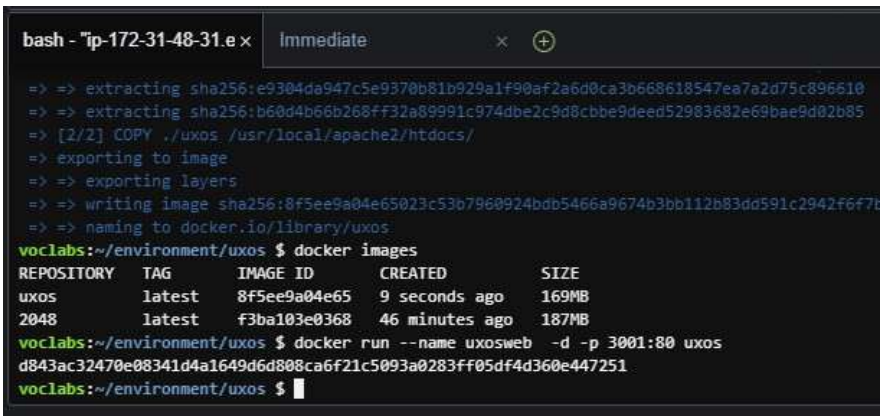
```
voclabs:~/environment/uxos $ docker build -t uxosweb .
[+] Building 0.4s (7/7) FINISHED                                docker:default
 => [internal] load build definition from Dockerfile                      0.0s
 => => transferring dockerfile: 162B                                      0.0s
 => [internal] 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
```

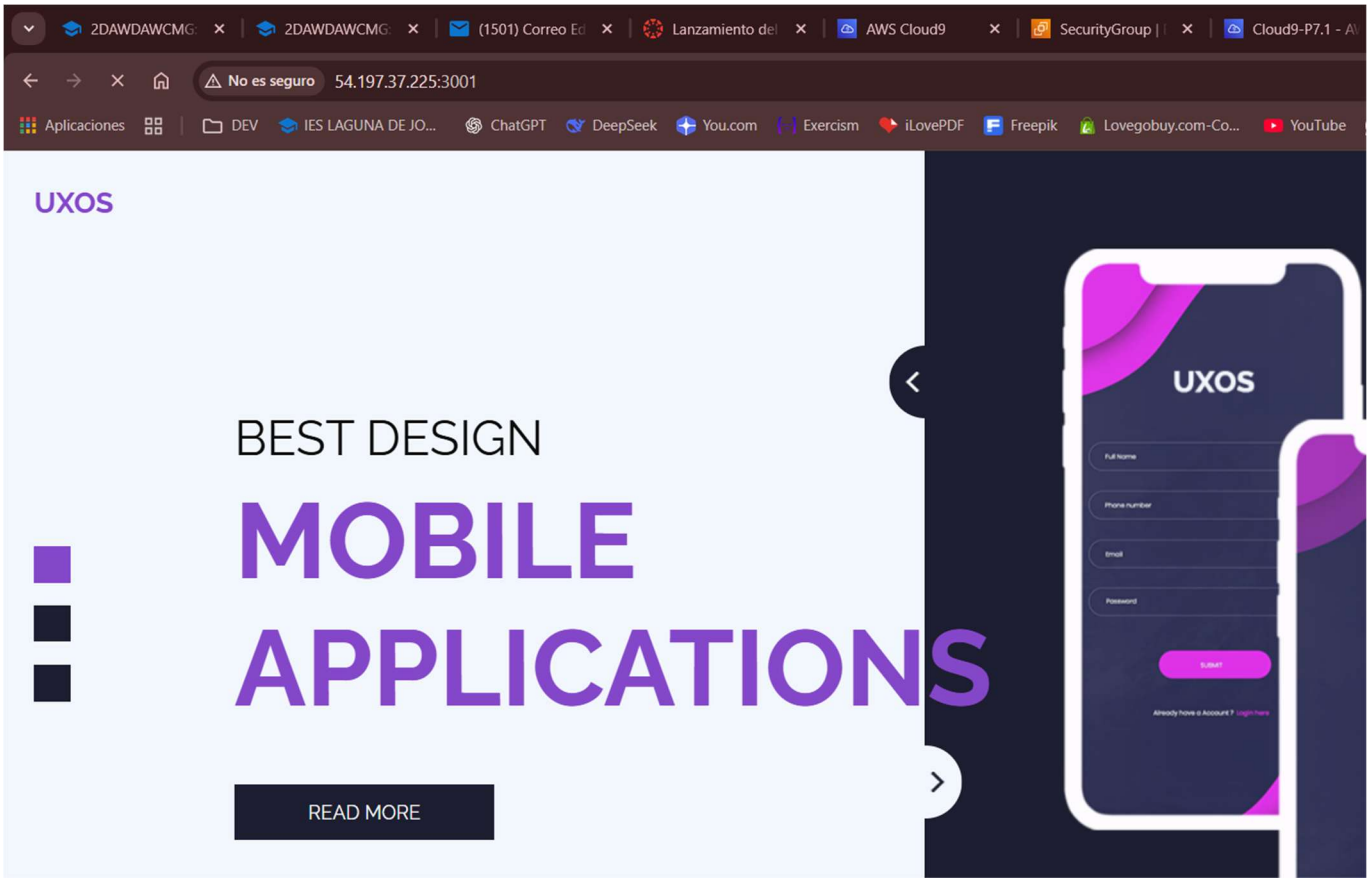
```
voclabs:~/environment/uxos $ docker run --name uxosweb -d -p 3001:80 u
xosweb
20cb257e8f7a57af358a51681b5e5f7c08680c18bb1aff4486d1e4b51d055169
voclabs:~/environment/uxos $
```

La aplicación uxos está localizada en apache2 y se crear 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.






```
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
0dafe1f74bd9   2048      "/docker-entrypoint.…"   46 minutes ago   Up 46 minutes   0.0.0.0:3000->80/tcp, :::3000->80/tcp      2048a
voclabs:~/environment/uxos $
```