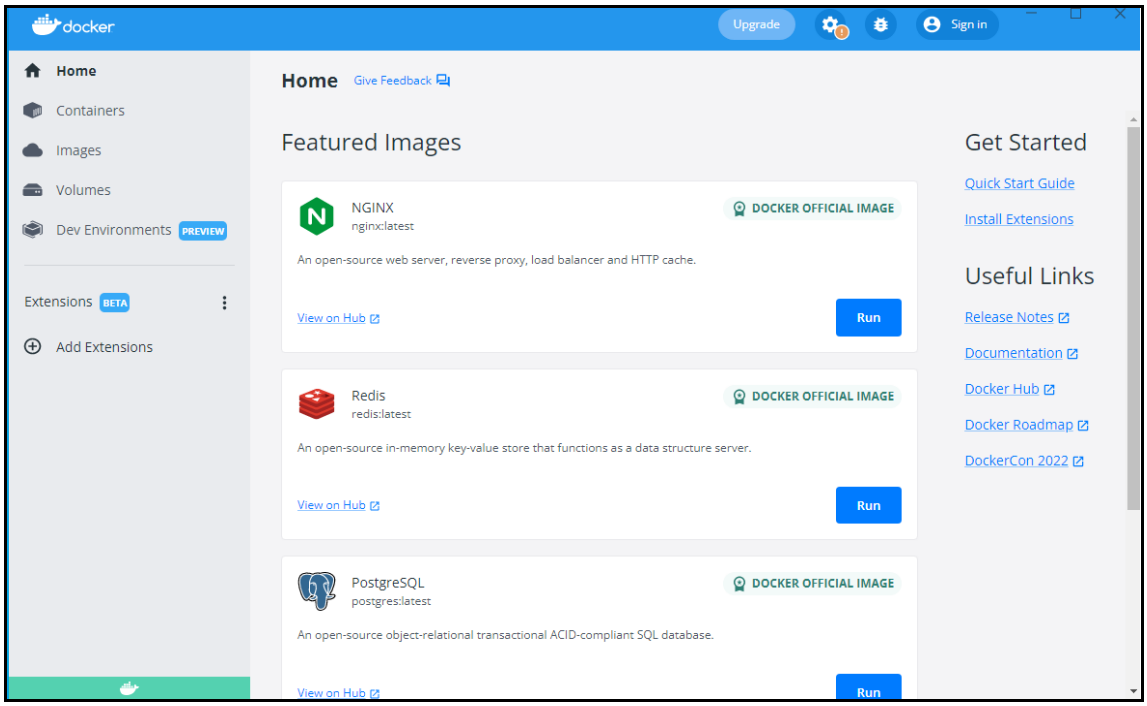
Prácticas Docker

1. Práctica con contenedores Windows

- En primer lugar vamos a trabajar con contenedores de tipo Linux para hacer estas pruebas prácticas sobre docker Desktop
- Arrancamos la herramienta de docker Desktop y nos situamos en Home



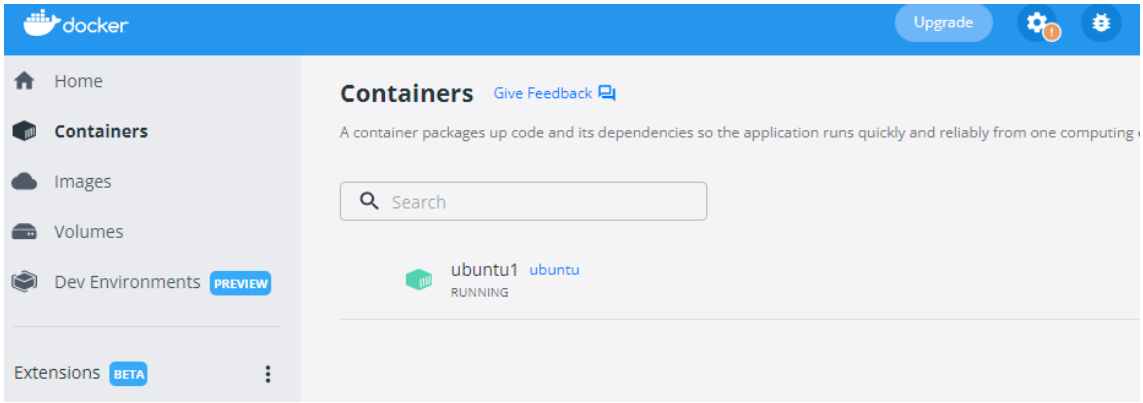
- Abrimos una sesión de comandos y probamos por ejemplo creando un contenedor basado en Ubuntu. Podemos comprobar que se hace exactamente igual a como si lo estuviéramos haciendo dentro de un entorno Linux

```
C:\practicas>docker run --name ubuntu1 -d -it ubuntu
e775eb03552c21332f82bd1e869ddb42bccb07324291ffb4d0a53197229e9404

C:\practicas>docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED          STATUS          PORTS     NAMES
e775eb03552c   ubuntu    "bash"    31 seconds ago   Up 23 seconds             ubuntu1
```

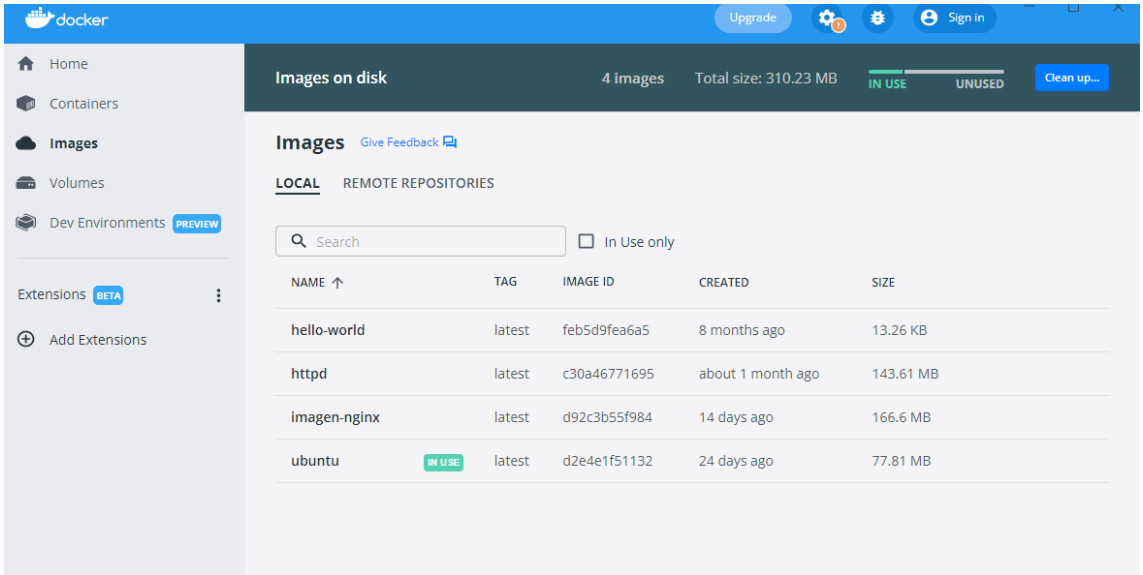
- Comprobamos las imágenes

```
C:\practicas>docker images
REPOSITORY      TAG       IMAGE ID       CREATED          SIZE
imagen-nginx    latest    d92c3b55f984   13 days ago      167MB

httpd           latest    c30a46771695   4 weeks ago      144MB
hello-world     latest    feb5d9fea6a5   8 months ago     13.3kB
```

- Comprobamos las propiedades de la imagen desde el entorno gráfico:
- Contenedores



- Imágenes



- Accedemos al contenedor

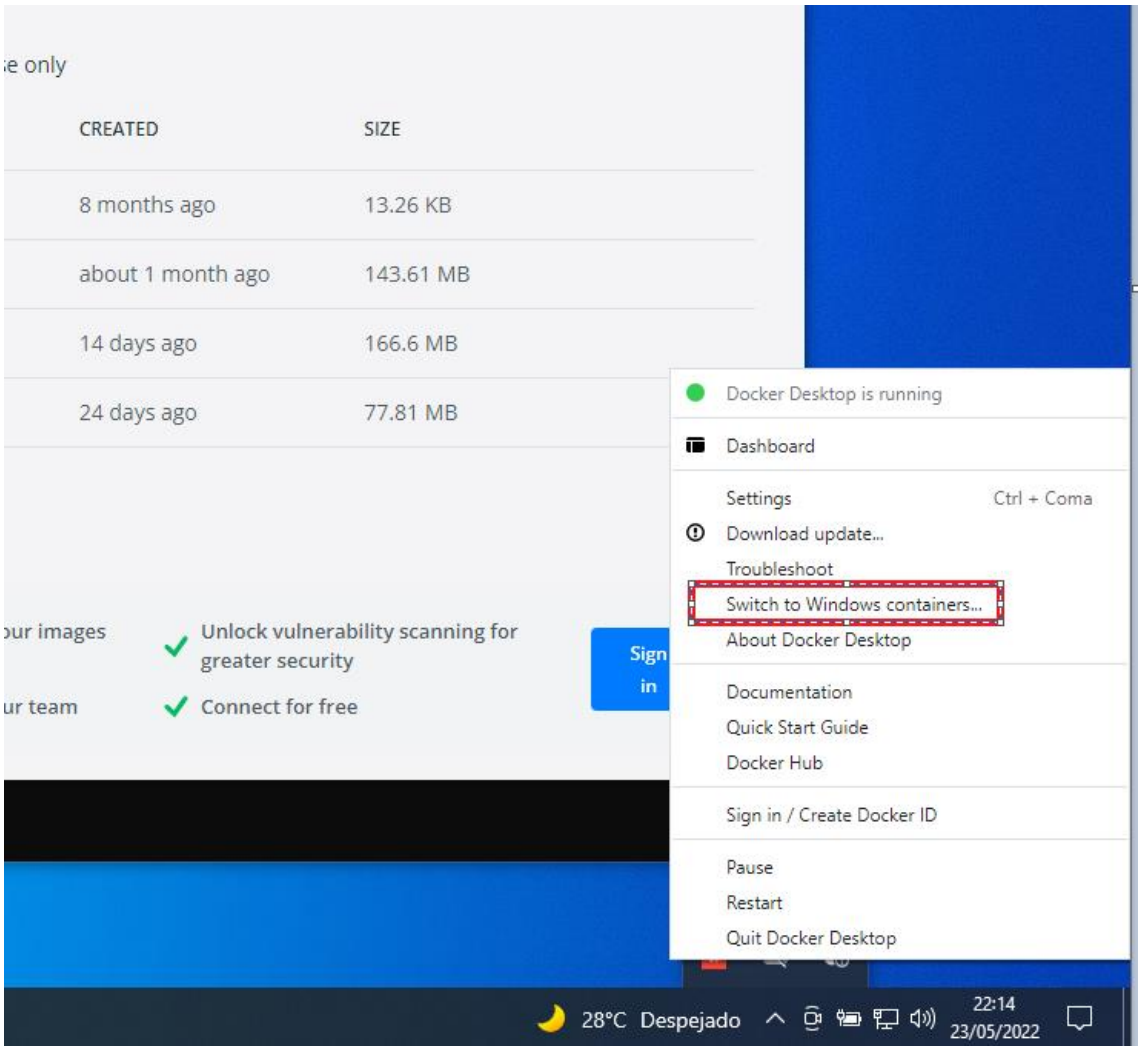
```
C:\practicas>docker exec -it ubuntu1 bash
root@e775eb03552c:/# ls
```

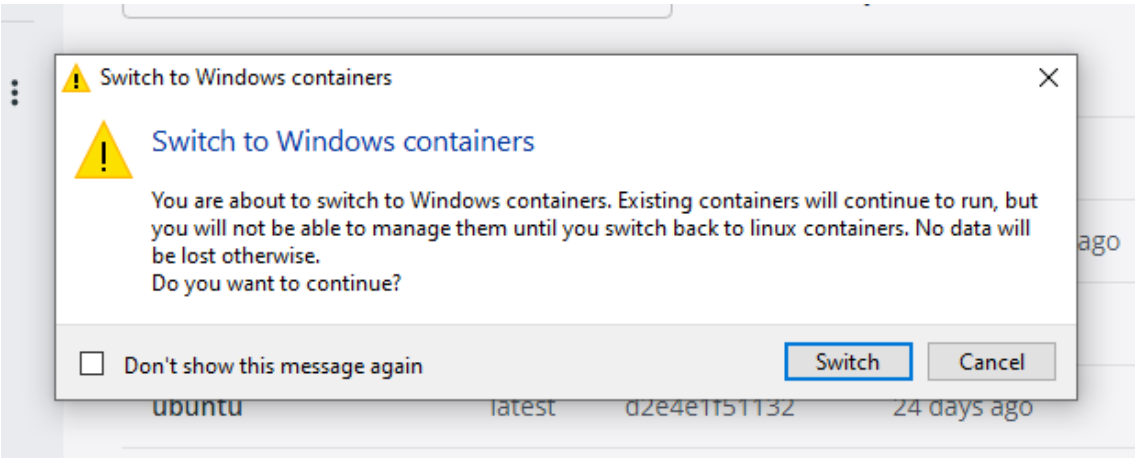
```
bin  boot  dev  etc  home  lib  lib32  lib64  libx32  media  mnt
opt  proc  root  run  sbin  srv  sys  tmp  usr  var
root@e775eb03552c:/#
```

- Salimos y lo borramos

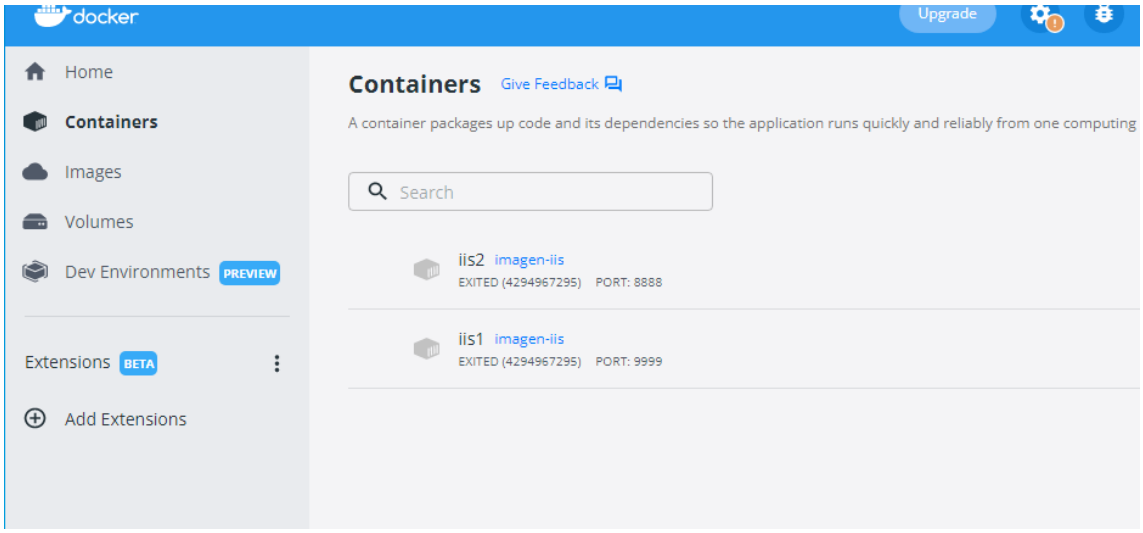
```
root@e775eb03552c:/# exit
exit

C:\practicas>docker rm -f ubuntu1
ubuntu1
```

- Ahora nos ponemos en modo Windows Containers
-

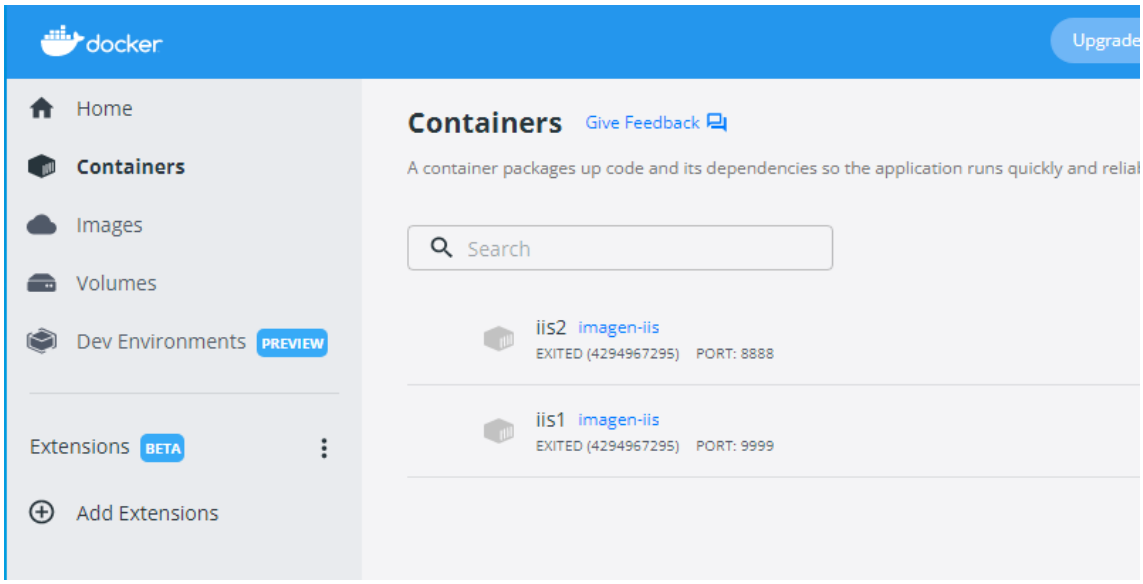




- Comprobamos que ha cambiado el entorno gráfico y se presentan las imágenes o contenedores que tengamos en Windows. En mi caso
- Contenedores



- Imágenes



- Abrimos otra sesión de “Símbolo de sistema” y comprobamos las posibles imágenes

```
C:\>docker images
REPOSITORY                                  TAG       IMAGE ID
CREATED       SIZE
imagen-iis                                  latest    f19cb5a865d7
13 days ago   5.73GB
mcr.microsoft.com/windows/servercore        20H2      8b5604092845
7 weeks ago   5.62GB
mcr.microsoft.com/windows/nanoserver        20H2      4396611cfaeb
7 weeks ago   263MB
```

- Vamos a crear ahora un contenedor basado en una imagen que está montada con una aplicación core net

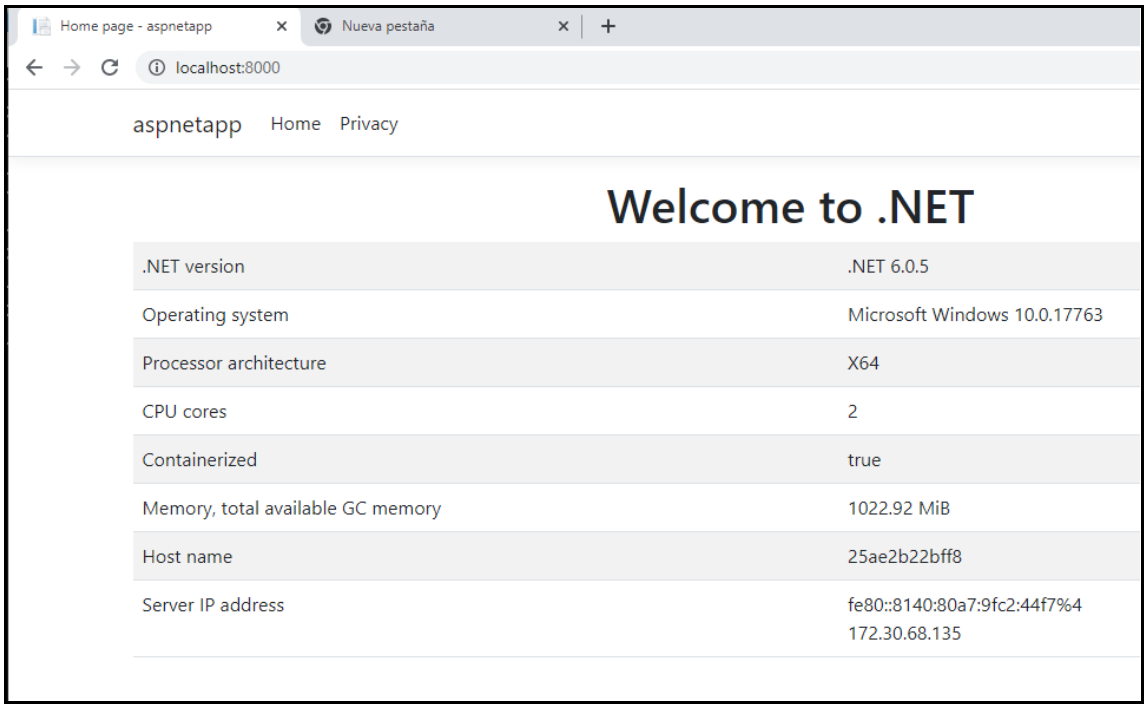
```
docker run --rm -it -p 8000:8080
mcr.microsoft.com/dotnet/samples:aspnetapp
```

- El resultado debe ser similar al siguiente:

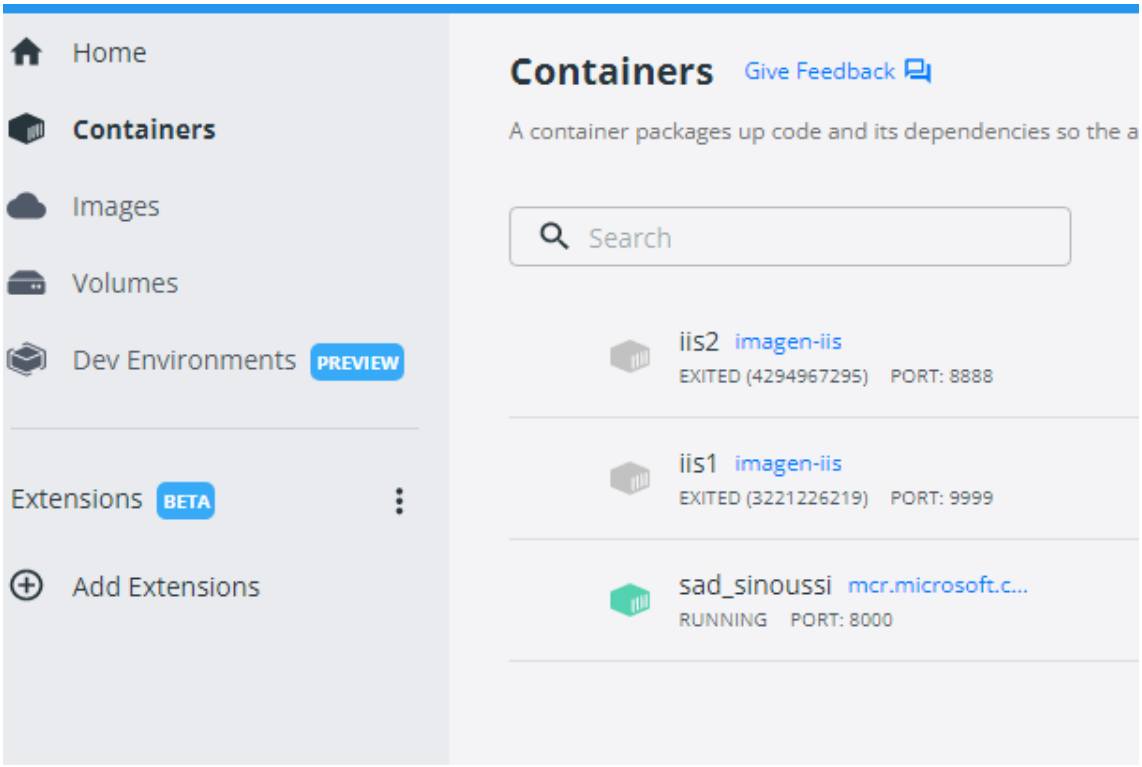
```
warn:
Microsoft.AspNetCore.DataProtection.Repositories.FileSystemXmlRe
pository[60]
      Storing keys in a directory
'C:\Users\ContainerUser\AppData\Local\ASP.NET\DataProtection-
Keys' that may not be per
sisted outside of the container. Protected data will be
unavailable when container is destroyed.
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: http://[::]:8080
```

```
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Production
info: Microsoft.Hosting.Lifetime[0]
      Content root path: C:\app\
```

- Ahora podemos abrir un navegador y entrar por el puerto 8000



- Podemos comprobar en el Docker Desktop que aparece el contenedor
-


Home
Containers
Images
Volumes
Dev Environments
PREVIEW
Extensions
BETA
Add Extensions
Containers
Give Feedback
A container packages up code and its dependencies so the a
Search
iis2 imagen-iis
EXITED (4294967295) PORT: 8888
iis1 imagen-iis
EXITED (3221226219) PORT: 9999
sad_sinoussi mcr.microsoft.c...
RUNNING PORT: 8000