Guía de Laboratorio Docker

Gerencia de Tecnología.

Documento de Técnico.

Elaborado por:

Julian Rivera-Pineda
Analista de Servicios
jurivera@promerica.com.sv

Banco Promerica de El Salvador

San Salvador 28 de septiembre de 2022

Historial de revisiones

Fecha	Versión	Descripción	Autor	Revisado por
27/09/2022	1	Creación de documento	Julian Rivera-Pineda	

Contenidos

Objetivo General. 2

Listado de Materiales a utilizar. 2

Procedimiento. 3

Parte 1. Configurando el ambiente Play with Docker 3

Parte 2. Ejecutando Microservicio con Payara Micro.... 14

Parte 3. Ejecutando Microservicio con Angular 20

Parte 4. Ejecutando Microservicio con .Net 7 25

Parte 5. Ejercicio de Desafío 1 27

Objetivo General.

Conocer el ambiente de contenedores Docker y sus principales comandos para gestionar cargas de trabajo empleando el laboratorio online Play with Docker.

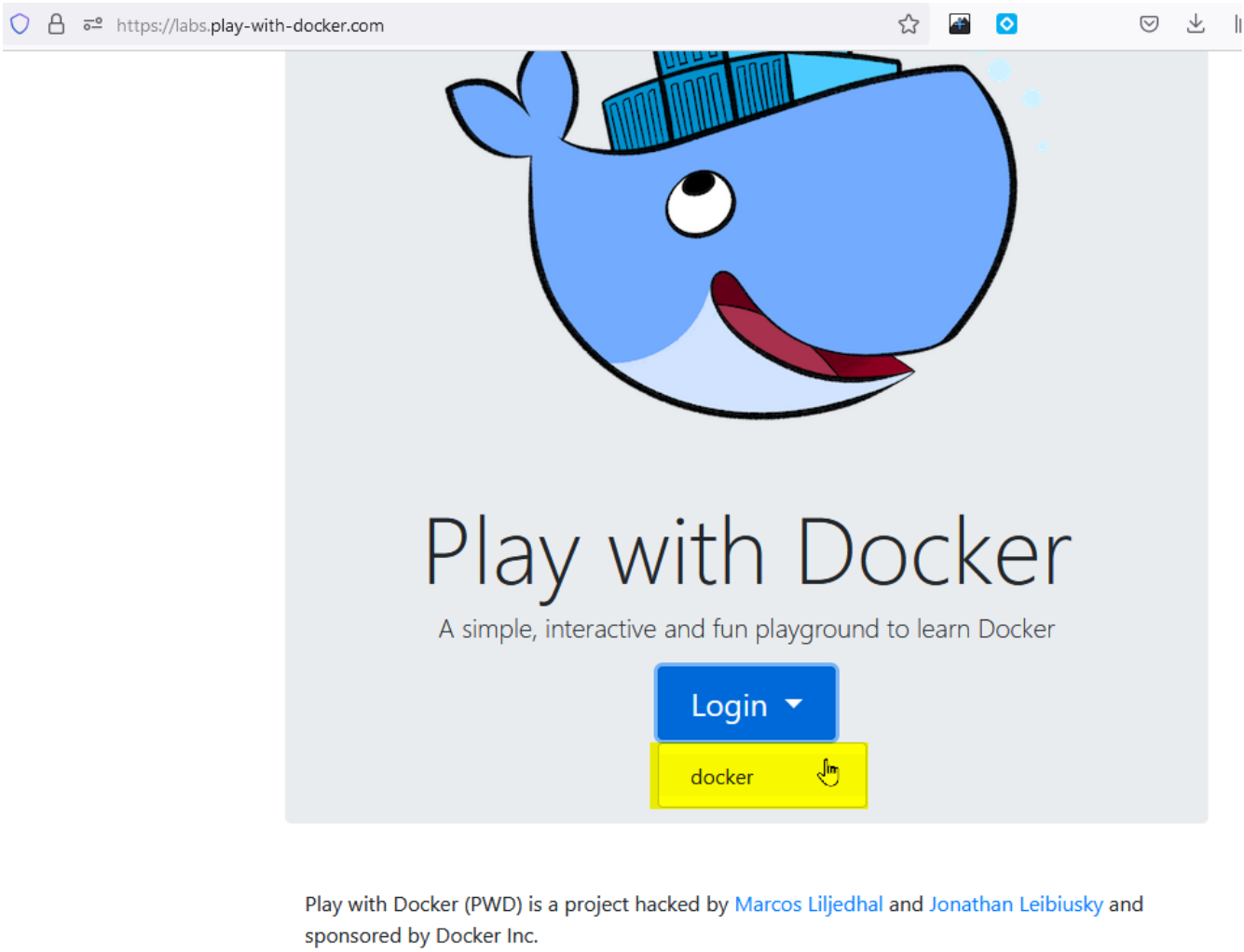
Listado de Materiales a utilizar.

- Navegador de Internet (Firefox, Chrome o Brave)
- Guía de Laboratorio (Esta guía)
- Usuario de Docker hub (Registrarse en el siguiente enlace https://hub.docker.com/signup)

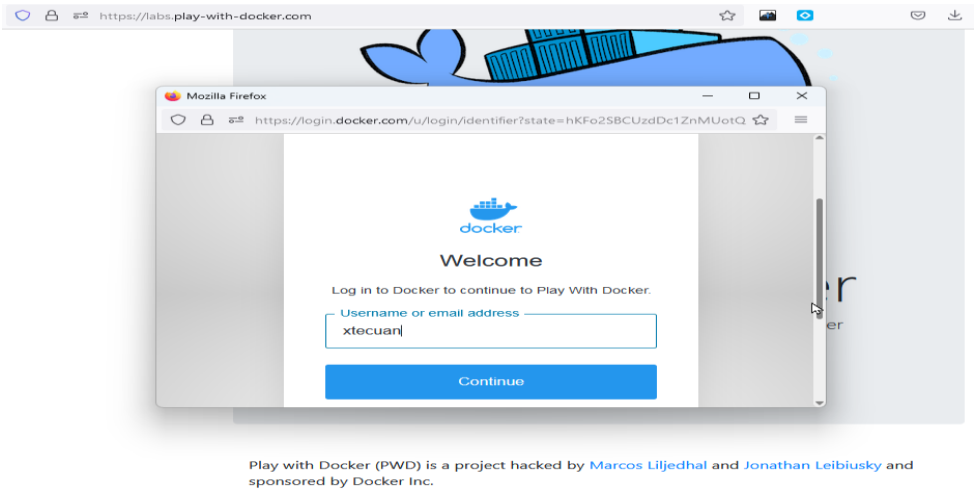
Procedimiento.

Parte 1. Configurando el ambiente Play with Docker

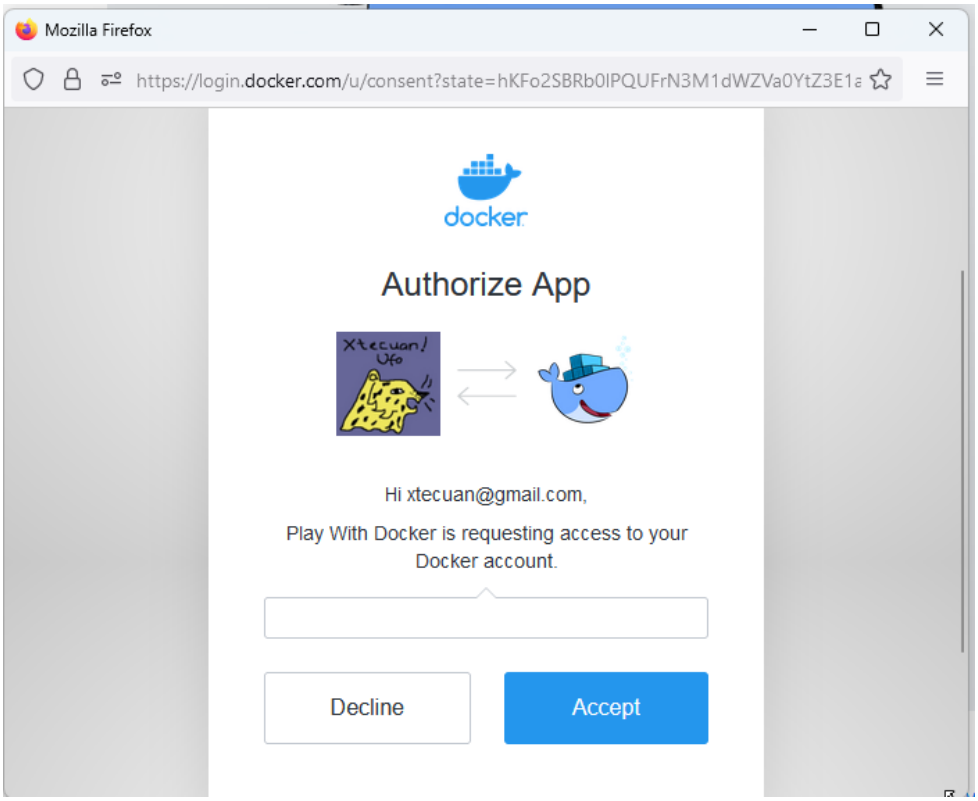
1. Ingresar a la página web de **Play with Docker** https://labs.play-with-docker.com/ :



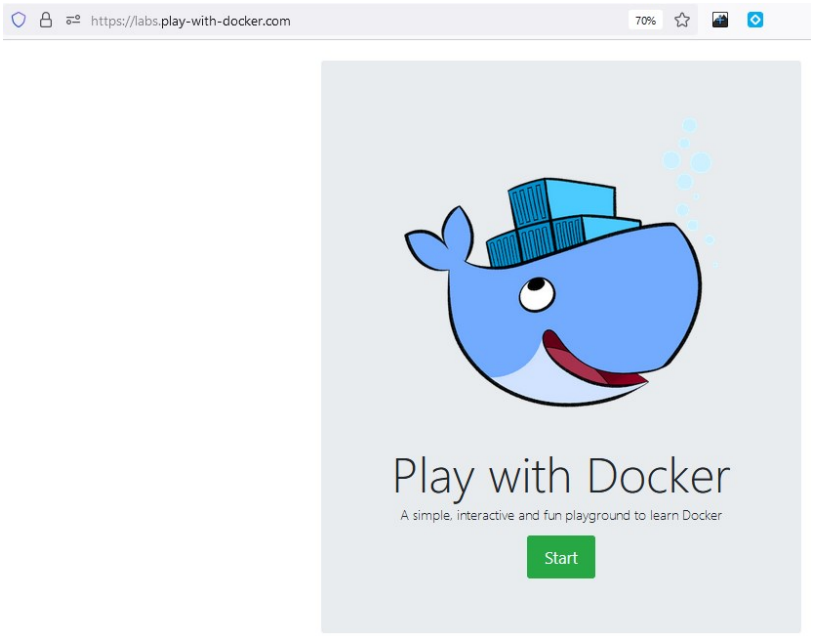
2. Seleccionar Login con docker (Se abrirá una ventana emergente, proceder a ingresar a Docker hub):



3. Autorizar a Play with Docker desde Docker Hub dando click en Accept:

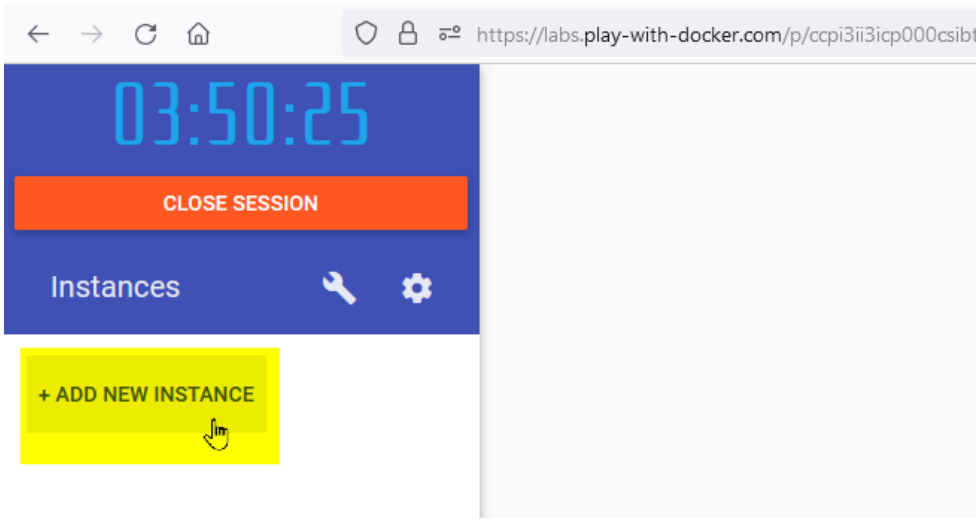


4. Iniciar la sesión de Play with Docker dando click en Start:

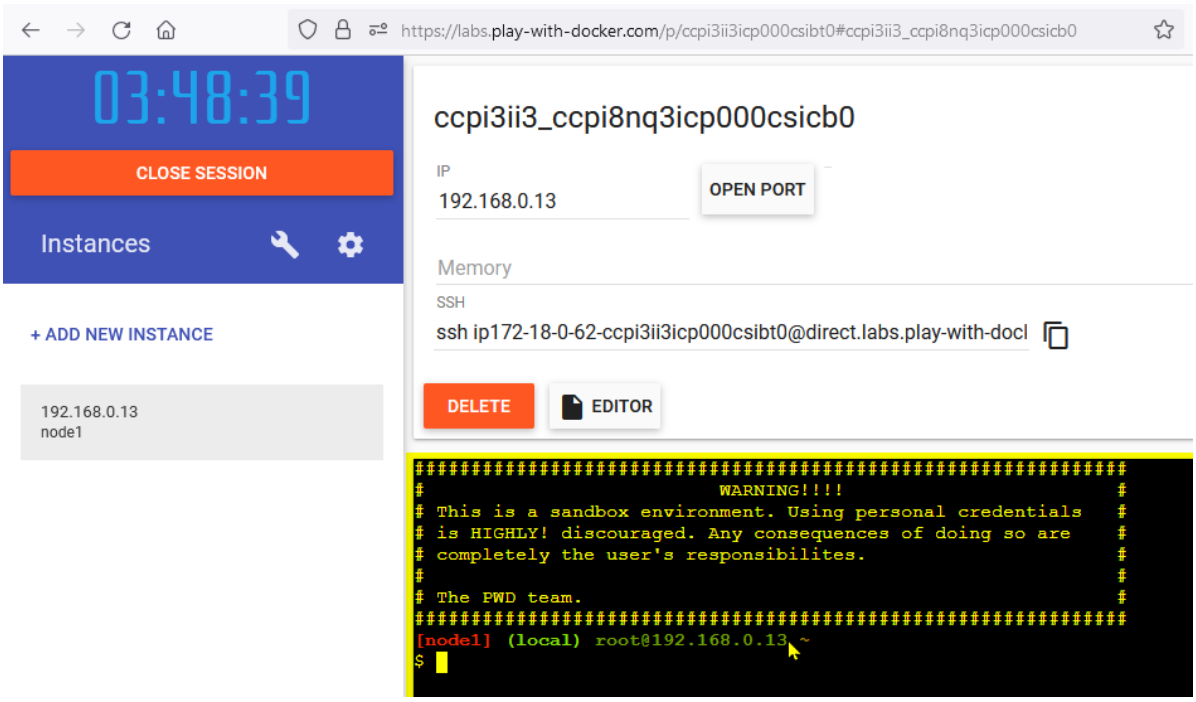


5. Se iniciará la sesión que tiene tiempo límite de 4h:

6. Presionar el enlace que dice ADD NEW INSTANCE:

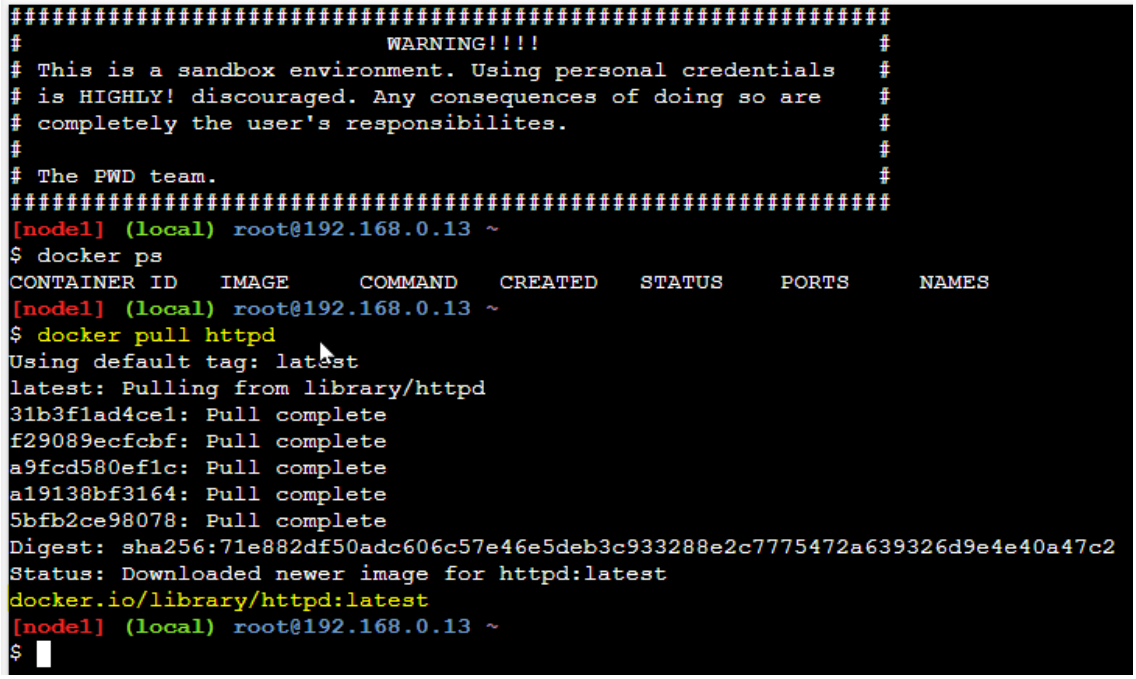


7. Aparecerá una terminal con el runtime de Docker listo para trabajar:

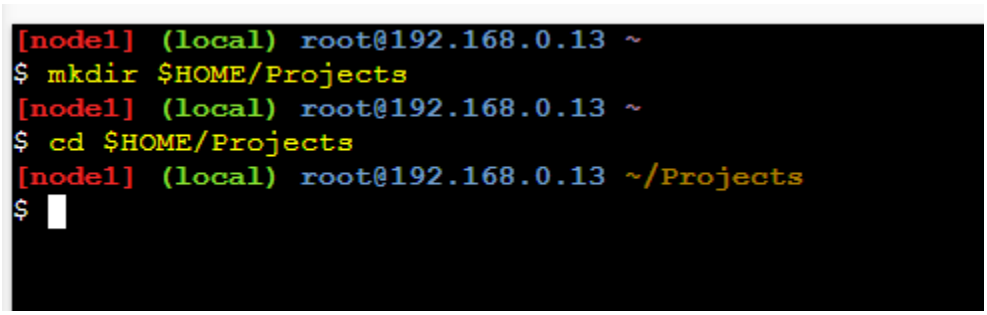


8. Verificar los contenedores que están siendo ejecutados con el comando << ***docker ps*** >>:

9. Como se puede ver al inicio la instancia de docker no muestra ningún contenedor ejecutándose. Hacer pull de una imagen de Apache HTTPD Server con el siguiente comando << ***docker pull httpd*** >>



10. Crear la carpeta de Projects << ***mkdir $HOME/projects*** >> donde crearemos nuestros archivos temporales para trabajar en este ambiente Docker, ingresar a la carpeta usando << ***cd $HOME/Projects*** >>:



11. Clonar repositorio de la guía usando el comando:

<< ***git clone*** ***https://github.com/xtecuan/guia-play-with-docker.git*** >>

```
[node1] (local) root@192.168.0.13 ~/Projects
$ git clone https://github.com/xtecuan/guia-play-with-docker.git
Cloning into 'guia-play-with-docker'...
remote: Enumerating objects: 8, done.
remote: Counting objects: 100% (8/8), done.
remote: Compressing objects: 100% (7/7), done.
remote: Total 8 (delta 0), reused 5 (delta 0), pack-reused 0
Receiving objects: 100% (8/8), done.
[node1] (local) root@192.168.0.13 ~/Projects
$
```

12. Crear un link simbólico para accesar rápidamente a los recursos del repositorio git con el siguiente comando << ***ln -s $HOME/Projects/guía-play-with-docker $HOME/Projects/guia1*** >>

```
[node1] (local) root@192.168.0.13 ~
$ ln -s $HOME/Projects/guia-play-with-docker $HOME/Projects/guia1
[node1] (local) root@192.168.0.13 ~
$ cd $HOME/Projects
[node1] (local) root@192.168.0.13 ~/Projects
$ ls
guia-play-with-docker  guia1                    httpd
[node1] (local) root@192.168.0.13 ~/Projects
$
```

13. Crear la carpeta httpd con el siguiente comando << ***mkdir $HOME/Projects/httpd*** >> y luego << ***cd $HOME/Projects/httpd*** >>:

```
[node1] (local) root@192.168.0.13 ~/Projects
$ mkdir $HOME/Projects/httpd
[node1] (local) root@192.168.0.13 ~/Projects
$ cd $HOME/Projects/httpd
[node1] (local) root@192.168.0.13 ~/Projects/httpd
$
```

14. Copiar el archivo **index.html** a la carpeta del contenedor httpd con el siguiente comando << ***cp $HOME/Projects/guia1/httpd/index.html $HOME/Projects/httpd/*** >> luego moverse a la carpeta httpd con el siguiente comando << ***cd $HOME/Projects/httpd*** >>

```
$ cp $HOME/Projects/guia1/httpd/index.html $HOME/Projects/httpd/
[node1] (local) root@192.168.0.13 ~/Projects
$ cd httpd/
[node1] (local) root@192.168.0.13 ~/Projects/httpd
$ ls
index.html
[node1] (local) root@192.168.0.13 ~/Projects/httpd
$
```

15. Ejecutar el contenedor **httpd** con el siguiente comando:

<<
docker run -dit --name mi_httpd -p 8080:80 -v "$PWD":/usr/local/apache2/htdocs/ httpd:latest
>>

```
[node1] (local) root@192.168.0.13 ~/Projects/httpd
$ docker run -dit --name mi_httpd -p 8080:80 -v "$PWD":/usr/local/apache2/htdocs/ httpd:latest
b650df4d48248727cef364b163ca70dca7a36529b4a1660878edfea127e5ff90
[node1] (local) root@192.168.0.13 ~/Projects/httpd
$
```

16. Hacer << ***docker ps*** >> para ver que el contenedor se está ejecutando:

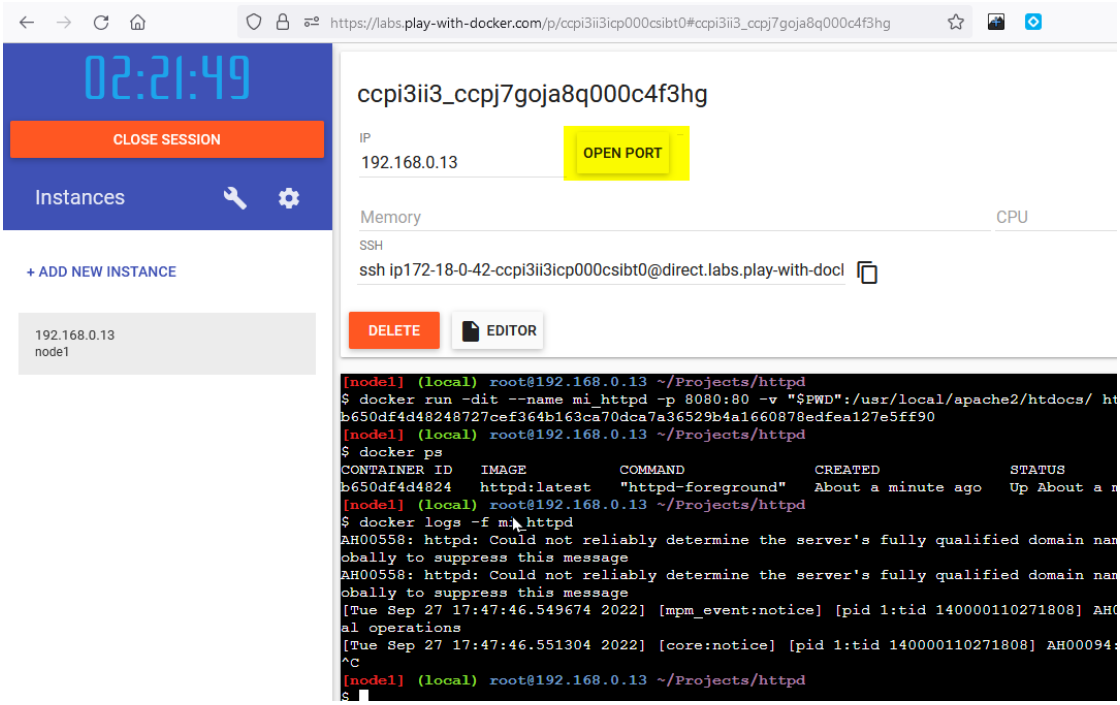
```
$ docker ps
CONTAINER ID   IMAGE          COMMAND              CREATED              STATUS              PORTS                    NAMES
b650df4d4824   httpd:latest   "httpd-foreground"   About a minute ago   Up About a minute   0.0.0.0:8080->80/tcp   mi_httpd
[node1] (local) root@192.168.0.13 ~/Projects/httpd
$
```

17. Revisar el log del contenedor con <<***docker logs -f mi_httpd*** >>

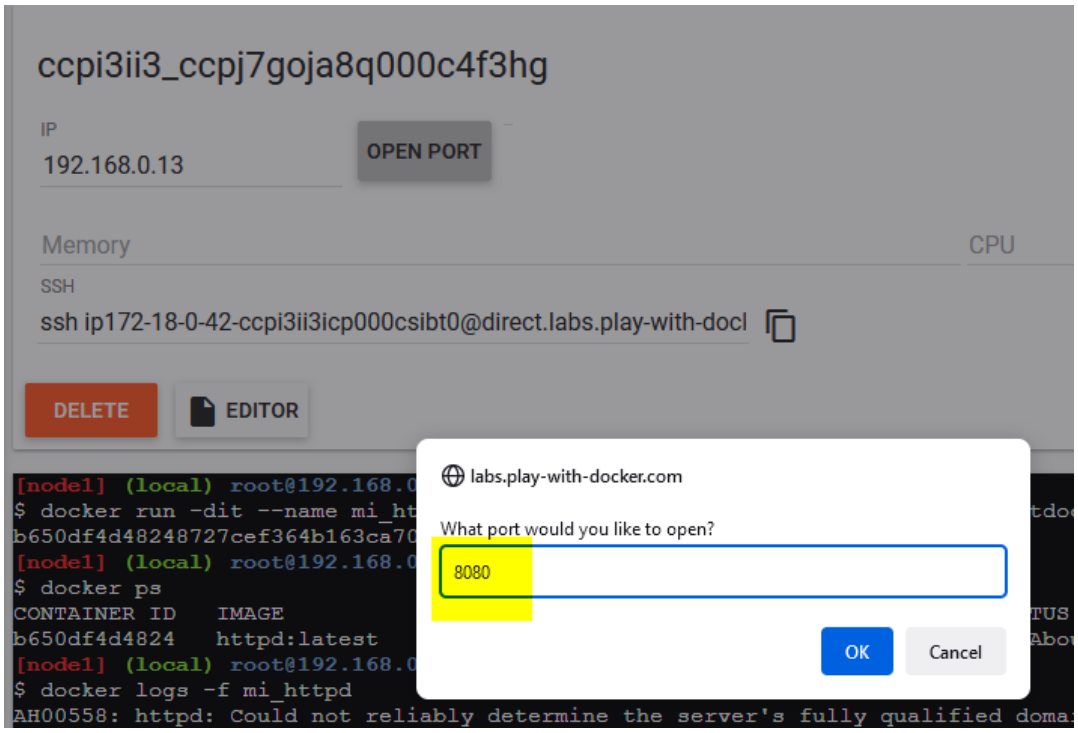
```
$ docker logs -f mi_httpd
AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using 172.17.0.2. Set the 'ServerName' directive gl
obally to suppress this message
AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using 172.17.0.2. Set the 'ServerName' directive gl
obally to suppress this message
[Tue Sep 27 17:47:46.549674 2022] [mpm_event:notice] [pid 1:tid 140000110271808] AH00489: Apache/2.4.54 (Unix) configured -- resuming norm
al operations
[Tue Sep 27 17:47:46.551304 2022] [core:notice] [pid 1:tid 140000110271808] AH00094: Command line: 'httpd -D FOREGROUND'
```

18. Presionar **Ctrl+C** para abandonar la vista de logs de docker

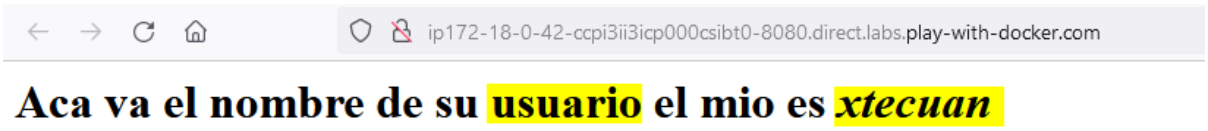
19. Presionar el botón **OPEN PORT** para ver nuestro **httpd** corriendo en Internet:



20. Ingresar el puerto 8080 y presionar **OK**:



21. El resultado será la página web siguiente:



22. Modifique la pagina Web con nano, pero primero instale nado usando el comando
<< ***apk add nano zip unzip*** >> (Se instalarán como extra zip y unzip):

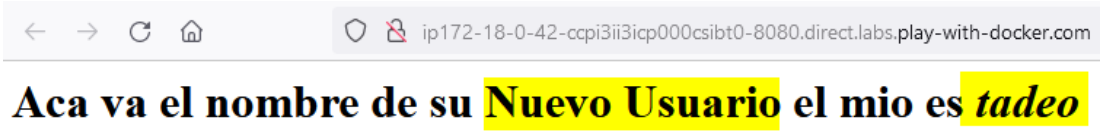
```
[node1] (local) root@192.168.0.13 ~/Projects/httpd
$ apk add nano zip unzip
fetch https://dl-cdn.alpinelinux.org/alpine/v3.16/main/x86_64/APKINDEX.tar.gz
fetch https://dl-cdn.alpinelinux.org/alpine/v3.16/community/x86_64/APKINDEX.tar.gz
(1/3) Installing nano (6.3-r0)
(2/3) Installing unzip (6.0-r9)
(3/3) Installing zip (3.0-r9)
Executing busybox-1.35.0-r13.trigger
OK: 396 MiB in 159 packages
[node1] (local) root@192.168.0.13 ~/Projects/httpd
$
```

23. Hacer << ***nano index.html*** >> cambiar donde dice usuario y xtecuan, escribir lo que quiera:

```
  GNU nano 6.3                                                  index.html
<html>
    <head>
        <title>Pagina de Prueba Guia Play with Docker</title>
    </head>
    <body>
        <h1>Aca va el nombre de su <b>usuario</b> el mio es <i>xtecuan</i></h1>
    </body>
</html>
```

Nota 1: ***Guardar con Ctrl+x Yes y Enter***

24. Recargar la pagina para ver los cambios:



25. ¡Felicidades acaba de crear su primer contenedor Docker!

26. Explore en la ayuda del comando **docker run para ver** que flags/opciones han sido empleadas para levantar nuestro primer contenedor con el siguiente comando << ***docker run --help*** >>

```
[node1] (local) root@192.168.0.13 ~
$ docker run --help

Usage:  docker run [OPTIONS] IMAGE [COMMAND] [ARG...]

Run a command in a new container

Options:
      --add-host list                Add a custom host-to-IP mapping (host:ip)
  -a, --attach list                  Attach to STDIN, STDOUT or STDERR
      --blkio-weight uint16          Block IO (relative weight), between 10 and 1000, or 0 to disable (default 0)
      --blkio-weight-device list     Block IO weight (relative device weight) (default [])
      --cap-add list                 Add Linux capabilities
      --cap-drop list                Drop Linux capabilities
      --cgroup-parent string         Optional parent cgroup for the container
      --cgroupns string              Cgroup namespace to use (host|private)
                                     'host':    Run the container in the Docker host's cgroup namespace
                                     'private': Run the container in its own private cgroup namespace
                                     '':        Use the cgroup namespace as configured by the
                                                default-cgroupns-mode option on the daemon (default)
      --cidfile string               Write the container ID to the file
      --cpu-period int               Limit CPU CFS (Completely Fair Scheduler) period
      --cpu-quota int                Limit CPU CFS (Completely Fair Scheduler) quota
      --cpu-rt-period int            Limit CPU real-time period in microseconds
      --cpu-rt-runtime int           Limit CPU real-time runtime in microseconds
```


Usage: docker run [OPTIONS] IMAGE [COMMAND] [ARG...]

Run a command in a new container

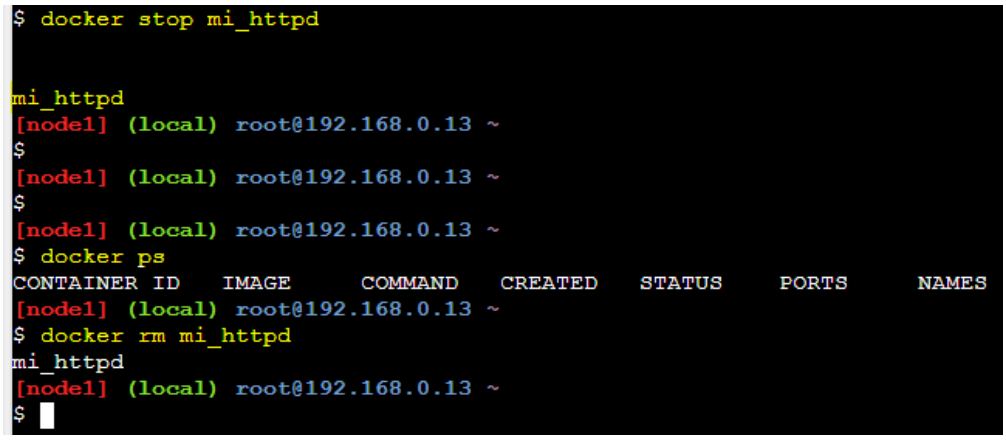
Options:
--add-host list Add a custom host-to-IP mapping (host:ip)
-a, --attach list Attach to STDIN, STDOUT or STDERR
--blkio-weight uint16 Block IO (relative weight), between 10 and 1000, or 0 to disable (default 0)
--blkio-weight-device list Block IO weight (relative device weight) (default [])
--cap-add list Add Linux capabilities
--cap-drop list Drop Linux capabilities
--cgroup-parent string Optional parent cgroup for the container
--cgroupns string Cgroup namespace to use (host|private)
'host': Run the container in the Docker host's cgroup namespace
'private': Run the container in its own private cgroup namespace
'': Use the cgroup namespace as configured by the
default-cgroupns-mode option on the daemon (default)
--cidfile string Write the container ID to the file
--cpu-period int Limit CPU CFS (Completely Fair Scheduler) period
--cpu-quota int Limit CPU CFS (Completely Fair Scheduler) quota
--cpu-rt-period int Limit CPU real-time period in microseconds
--cpu-rt-runtime int Limit CPU real-time runtime in microseconds
-c, --cpu-shares int CPU shares (relative weight)
--cpus decimal Number of CPUs
--cpuset-cpus string CPUs in which to allow execution (0-3, 0,1)
--cpuset-mems string MEMs in which to allow execution (0-3, 0,1)
-d, --detach Run container in background and print container ID
--detach-keys string Override the key sequence for detaching a container

```
      --device list                Add a host device to the container
      --device-cgroup-rule list    Add a rule to the cgroup allowed devices list
      --device-read-bps list       Limit read rate (bytes per second) from a device (default [])
      --device-read-iops list      Limit read rate (IO per second) from a device (default [])
      --device-write-bps list      Limit write rate (bytes per second) to a device (default [])
      --device-write-iops list     Limit write rate (IO per second) to a device (default [])
      --disable-content-trust      Skip image verification (default true)
      --dns list                   Set custom DNS servers
      --dns-option list            Set DNS options
      --dns-search list            Set custom DNS search domains
      --domainname string          Container NIS domain name
      --entrypoint string          Overwrite the default ENTRYPOINT of the image
  -e, --env list                   Set environment variables
      --env-file list              Read in a file of environment variables
      --expose list                Expose a port or a range of ports
      --gpus gpu-request           GPU devices to add to the container ('all' to pass all GPUs)
      --group-add list             Add additional groups to join
      --health-cmd string          Command to run to check health
      --health-interval duration   Time between running the check (ms|s|m|h) (default 0s)
      --health-retries int         Consecutive failures needed to report unhealthy
      --health-start-period duration   Start period for the container to initialize before starting
health-retries countdown (ms|s|m|h) (default 0s)
      --health-timeout duration    Maximum time to allow one check to run (ms|s|m|h)
(default 0s)
      --help                       Print usage
  -h, --hostname string            Container host name
      --init                       Run an init inside the container that forwards signals and reaps
processes
  -i, --interactive                Keep STDIN open even if not attached
      --ip string                  IPv4 address (e.g., 172.30.100.104)
      --ip6 string                 IPv6 address (e.g., 2001:db8::33)
      --ipc string                 IPC mode to use
      --isolation string           Container isolation technology
      --kernel-memory bytes        Kernel memory limit
  -l, --label list                 Set meta data on a container
      --label-file list            Read in a line delimited file of labels
      --link list                  Add link to another container
      --link-local-ip list         Container IPv4/IPv6 link-local addresses
      --log-driver string          Logging driver for the container
      --log-opt list               Log driver options
      --mac-address string         Container MAC address (e.g., 92:d0:c6:0a:29:33)
  -m, --memory bytes               Memory limit
      --memory-reservation bytes   Memory soft limit
      --memory-swap bytes          Swap limit equal to memory plus swap: '-1' to enable
unlimited swap
      --memory-swappiness int      Tune container memory swappiness (0 to 100) (default -1)
      --mount mount                Attach a filesystem mount to the container
      --name string                Assign a name to the container
      --network network            Connect a container to a network
      --network-alias list         Add network-scoped alias for the container
```

```
      --no-healthcheck           Disable any container-specified HEALTHCHECK
      --oom-kill-disable         Disable OOM Killer
      --oom-score-adj int        Tune host's OOM preferences (-1000 to 1000)
      --pid string               PID namespace to use
      --pids-limit int           Tune container pids limit (set -1 for unlimited)
      --platform string          Set platform if server is multi-platform capable
      --privileged               Give extended privileges to this container
  -p, --publish list             Publish a container's port(s) to the host
  -P, --publish-all              Publish all exposed ports to random ports
      --pull string              Pull image before running ("always"|"missing"|"never") (default
"missing")
      --read-only                Mount the container's root filesystem as read only
      --restart string           Restart policy to apply when a container exits (default "no")
      --rm                       Automatically remove the container when it exits
      --runtime string           Runtime to use for this container
      --security-opt list        Security Options
      --shm-size bytes           Size of /dev/shm
      --sig-proxy                Proxy received signals to the process (default true)
      --stop-signal string       Signal to stop a container (default "SIGTERM")
      --stop-timeout int         Timeout (in seconds) to stop a container
      --storage-opt list         Storage driver options for the container
      --sysctl map               Sysctl options (default map[])
      --tmpfs list               Mount a tmpfs directory
  -t, --tty                      Allocate a pseudo-TTY
      --ulimit ulimit            Ulimit options (default [])
  -u, --user string              Username or UID (format: <name|uid>[:<group|gid>])
      --userns string            User namespace to use
      --uts string               UTS namespace to use
  -v, --volume list              Bind mount a volume
      --volume-driver string     Optional volume driver for the container
      --volumes-from list        Mount volumes from the specified container(s)
-w, --workdir string             Working directory inside the container
```

Nota 2: para detener el contenedor ejecutar << ***docker stop mi_httpd*** >>

Nota 3: para eliminar el contenedor ejecutar << ***docker rm mi_httpd*** >>



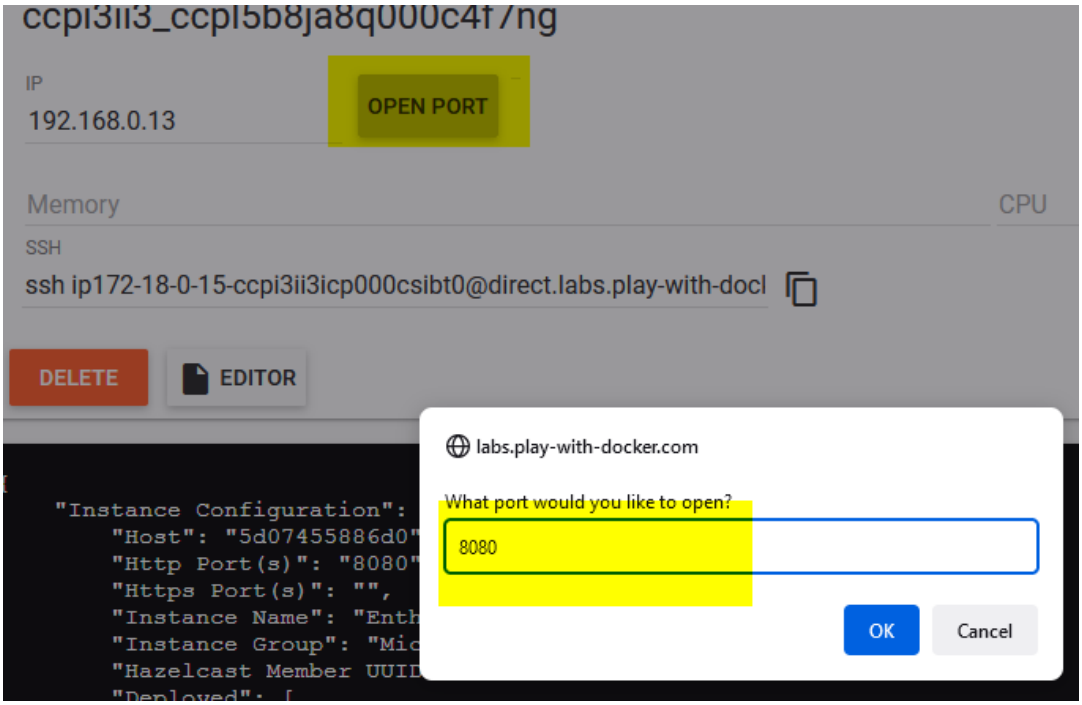
Parte 2. Ejecutando Microservicio con Payara Micro

1. Para ejecutar un contenedor con Payara Micro sin ninguna aplicación publicada hacemos

<< ***docker run -p 8080:8080 payara/micro*** >>

```
[node1] (local) root@192.168.0.13 ~
$ docker run -p 8080:8080 payara/micro
Unable to find image 'payara/micro:latest' locally
latest: Pulling from payara/micro
9621f1afde84: Pull complete
53138e828aac: Pull complete
53d3d5878603: Pull complete
494169db3af8: Pull complete
9bdb5392ad35: Pull complete
Digest: sha256:14dc425a52ca316505915e2a3bc7ff6d1fcaea0dd13a05d0b5769c171c5f9ea2
Status: Downloaded newer image for payara/micro:latest
[2022-09-27T19:47:54.753+0000] [] [WARNING] [] [PayaraMicro] [tid: _ThreadID=1 _ThreadName=main] [timeMillis: 1664308074753] [levelValue:
900] Payara Micro Runtime directory is located in a temporary file location which can be cleaned by system processes.

[2022-09-27T19:47:54.791+0000] [] [INFO] [] [PayaraMicro] [tid: _ThreadID=1 _ThreadName=main] [timeMillis: 1664308074791] [levelValue: 800
] Payara Micro Runtime directory is located at /tmp/payaramicro-rt5240215424352983604tmp

[2022-09-27T19:47:54.805+0000] [] [INFO] [] [fish.payara.micro.boot.runtime.PayaraMicroRuntimeBuilder] [tid: _ThreadID=1 _ThreadName=main]
 [timeMillis: 1664308074805] [levelValue: 800] Built Payara Micro Runtime
```

2. Si hacemos OPEN PORT en el puerto 8080:



3. El resultado sería el siguiente:

4. Presionamos **Ctrl + C** para interrumpir la ejecución del contenedor
5. Compilar código fuente de ejemplo de microservicio en Eclipse Microprofile usando Maven en Docker:

<< ***cd $HOME/Projects/guia1/book-store*** >>

<< ***docker run -it --rm --name my-maven-project -v "$(pwd)":/tmp/src -w /tmp/src maven:3.8.6-openjdk-11 mvn clean install*** >>

```
[node1] (local) root@192.168.0.13 ~/Projects/guia1/book-store
$ docker run -it --rm --name book-store-compiler -v "$(pwd)":/tmp/src -w /tmp/src maven:3.8.6-openjdk-11 mvn clean install
[INFO] Scanning for projects...
[INFO]
[INFO] ----------------------< sv.org.devfest:book-store >----------------------
[INFO] Building Taller MicroProfile :: book-store 1.0-SNAPSHOT
[INFO] --------------------------------[ war ]---------------------------------
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-clean-plugin/2.5/maven-clean-plugin-2.5.pom
```

6. El resulta será el siguiente:

```
[INFO] Installing /tmp/src/target/restapi.war to /root/.m2/repository/sv/org/devfest/book-store/1.0-SNAPSHOT/book-store-1.0-SNAPSHOT.war
[INFO] Installing /tmp/src/pom.xml to /root/.m2/repository/sv/org/devfest/book-store/1.0-SNAPSHOT/book-store-1.0-SNAPSHOT.pom
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  11.986 s
[INFO] Finished at: 2022-09-27T20:14:20Z
[INFO] ------------------------------------------------------------------------
[node1] (local) root@192.168.0.13 ~/Projects/guia1/book-store
$
```

7. Crear una imagen personalizada de Payara Micro para publicar el ejemplo ***book-store:***

<< ***cd $HOME/Projects/guia1/book-store*** >>

```
  GNU nano 6.3
FROM payara/micro
COPY target/restapi.war $DEPLOY_DIR
```

8. Hacer build de la imagen personalizada (Se requiere haber concluido con éxito los numerales del 5 y 6):

<< ***docker build -t mipayara .*** >>

```
[node1] (local) root@192.168.0.8 ~/Projects/guia1/book-store
$ docker build -t mipayara .
Sending build context to Docker daemon  21.84MB
Step 1/2 : FROM payara/micro
 ---> 2b2266e1373c
Step 2/2 : COPY target/restapi.war $DEPLOY_DIR
 ---> Using cache
 ---> fb01ce934d82
Successfully built fb01ce934d82
Successfully tagged mipayara:latest
[node1] (local) root@192.168.0.8 ~/Projects/guia1/book-store
$
```

9. Ejecutar la imagen personalizada de Payara Micro:

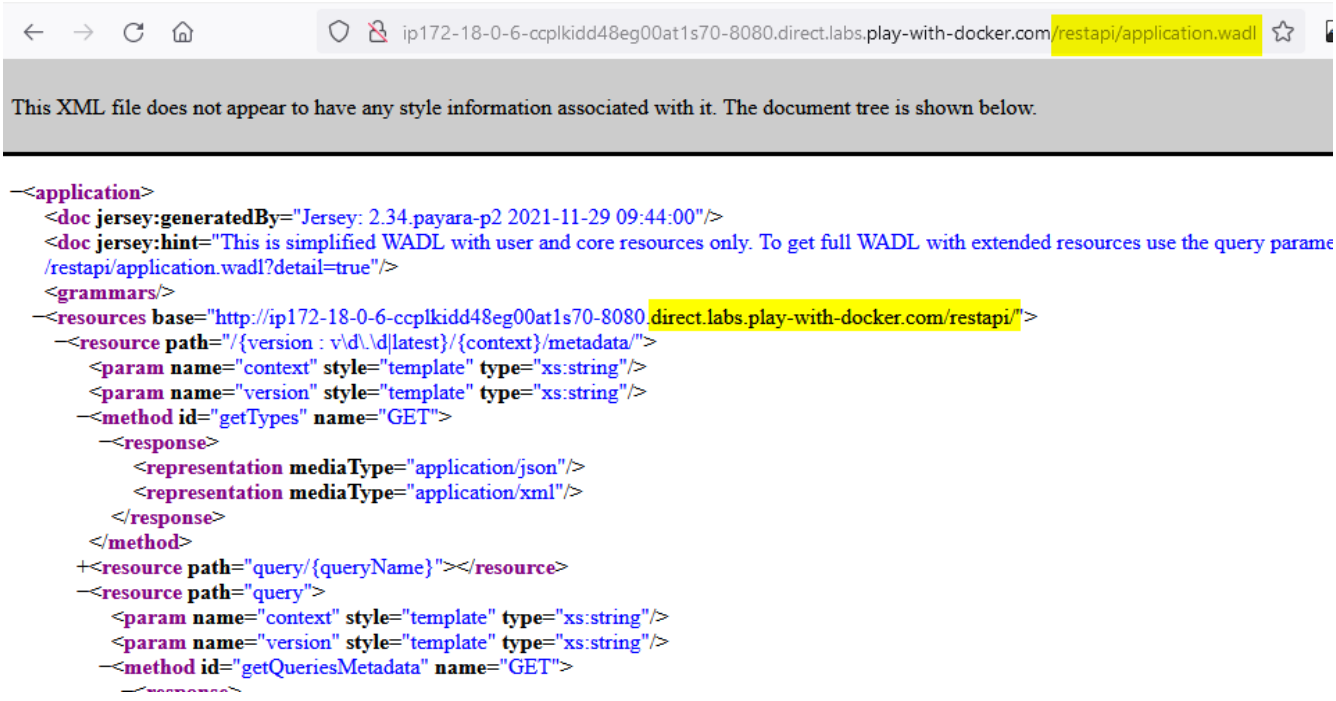
<< ***docker run -dit --name mipayara_container -p 8080:8080 mipayara:latest*** >>

```
[node1] (local) root@192.168.0.8 ~/Projects/guia1/book-store
$ docker run -dit --name mipayara_container -p 8080:8080 mipayara:latest
7deced0008ff523a09ec24ba0dd5bd6178e4c29a3d9f6109a0f8c233906a8916
[node1] (local) root@192.168.0.8 ~/Projects/guia1/book-store
$ docker ps
CONTAINER ID   IMAGE            COMMAND                  CREATED         STATUS         PORTS                              NAMES
7deced0008ff   mipayara:latest   "/bin/sh entrypoint.…"   6 seconds ago   Up 5 seconds   6900/tcp, 0.0.0.0:8080->8080/tcp   mipayara_conta
iner
[node1] (local) root@192.168.0.8 ~/Projects/guia1/book-store
$ docker logs -f mipayara_container
[2022-09-27T20:47:09.184+0000] [] [WARNING] [] [PayaraMicro] [tid: _ThreadID=1 _ThreadName=main] [timeMillis: 1664311629184] [levelValue:
900] Payara Micro Runtime directory is located in a temporary file location which can be cleaned by system processes.

[2022-09-27T20:47:09.220+0000] [] [INFO] [] [PayaraMicro] [tid: _ThreadID=1 _ThreadName=main] [timeMillis: 1664311629220] [levelValue: 800
] Payara Micro Runtime directory is located at /tmp/payaramicro-rt3269497805604447907tmp

[2022-09-27T20:47:09.236+0000] [] [INFO] [] [fish.payara.micro.boot.runtime.PayaraMicroRuntimeBuilder] [tid: _ThreadID=1 _ThreadName=main]
 [timeMillis: 1664311629236] [levelValue: 800] Built Payara Micro Runtime

[2022-09-27T20:47:12.180+0000] [] [INFO] [] [fish.payara.boot.runtime.BootCommand] [tid: _ThreadID=1 _ThreadName=main] [timeMillis: 166431
```

```
[2022-09-27T20:47:41.935+0000] [] [INFO] [] [PayaraMicro] [tid: _Thread
] [[

Payara Micro URLs:
http://7deced0008ff:8080/restapi

'restapi' REST Endpoints:
GET     /restapi/
POST    /restapi/
GET     /restapi/application.wadl
GET     /restapi/books
POST    /restapi/books
DELETE  /restapi/books/{id}
GET     /restapi/books/{id}
PUT     /restapi/books/{id}
GET     /restapi/hello
POST    /restapi/{context}/entity/{type}
PUT     /restapi/{context}/entity/{type}
DELETE  /restapi/{context}/entity/{type}/{id}
GET     /restapi/{context}/entity/{type}/{id}
DELETE  /restapi/{context}/entity/{type}/{id}/{attribute}
GET     /restapi/{context}/entity/{type}/{id}/{attribute}
POST    /restapi/{context}/entity/{type}/{id}/{attribute}
GET     /restapi/{context}/metadata
```

10. Automáticamente se abrirá el Puerto **8080** en el nodo de **Play with Docker** pudiendo ingresar a la pagina web del descriptor del servicio **RESTful** en el path /**restapi/application.wadl**


ccplkidd_ccplkjld48eg00at1s7g
IP
192.168.0.8
OPEN PORT
8080
http://ip172-18-0-6-ccplkidd48
Memory
47.92% (1.872GiB / 3.906GiB)
SSH
ssh ip172-18-0-6-ccplkidd48eg00at1s70@direct.labs.play-with-doc
DELETE
EDITOR
}]]
[2022-09-27T20:47:41.935+0000] [] [INFO] [] [PayaraMicro]
] [[
Payara Micro URLs:
http://7deced0008ff:8080/restapi


ip172-18-0-6-ccplkidd48eg00at1s70-8080.direct.labs.play-with-docker.com/restapi/application.wadl
This XML file does not appear to have any style information associated with it. The document tree is shown below.
<application>
<doc jersey:generatedBy="Jersey: 2.34.payara-p2 2021-11-29 09:44:00"/>
<doc jersey:hint="This is simplified WADL with user and core resources only. To get full WADL with extended resources use the query parame
/restapi/application.wadl?detail=true"/>
<grammars/>
<resources base="http://ip172-18-0-6-ccplkidd48eg00at1s70-8080.direct.labs.play-with-docker.com/restapi/">
<resource path="/{version : v\d\.\d|latest}/{context}/metadata/">
<param name="context" style="template" type="xs:string"/>
<param name="version" style="template" type="xs:string"/>
<method id="getTypes" name="GET">
<response>
<representation mediaType="application/json"/>
<representation mediaType="application/xml"/>
</response>
</method>
<resource path="query/{queryName}"></resource>
<resource path="query">
<param name="context" style="template" type="xs:string"/>
<param name="version" style="template" type="xs:string"/>
<method id="getQueriesMetadata" name="GET">

11. Para probar la aplicación backend puede usar desde su máquina la colección de **POSTman** que esta alojada en el respositorio git de la guía en la URL:

https://github.com/xtecuan/guia-play-with-docker/blob/main/postman/book-store.postman_collection.json

12. Emplee la URL del backend generada por el nodo que tiene el formato siguiente:

http://nombreautogenerado-puerto.**direct.labs.play-with-docker.com/restapi/application.wadl**

13. Otra forma de Probar la API de **book-store** es usando **curl** o empleando los scripts que están en el repositorio:

<< **cd $HOME/Projects/guia1/scripts** >>

<< ***chmod a+x *.sh*** >>

```
[node1] (local) root@192.168.0.8 ~
$ cd $HOME/Projects/guia1/scripts
[node1] (local) root@192.168.0.8 ~/Projects/guia1/scripts
$ ls
callDeleteRestapi.sh    callPostRestapi.sh      callPutRestapi.sh       env.sh
callGetRestapi.sh       callPostRestapi_all.sh  callPutRestapi_all.sh
[node1] (local) root@192.168.0.8 ~/Projects/guia1/scripts
$ chmod a+x *.sh
[node1] (local) root@192.168.0.8 ~/Projects/guia1/scripts
$
```

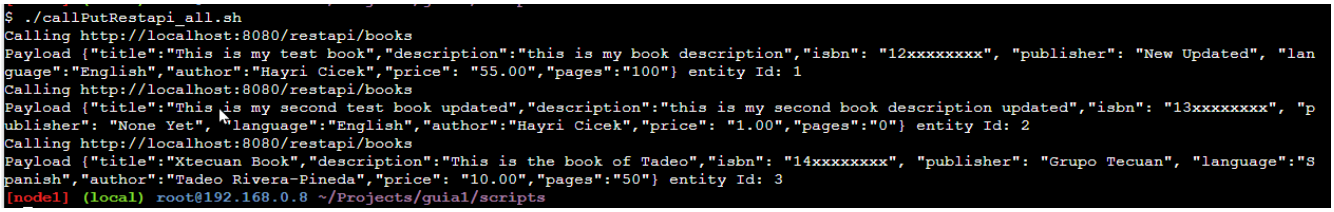
14. Crear Libros:

```
$ ./callPostRestapi_all.sh
Calling http://localhost:8080/restapi/books
Payload {"title":"This is my test book","description":"this is my book description","isbn": "12xxxxxxxx", "publisher": "None Yet", "langua
ge":"English","author":"Hayri Cicek","price": "0.00","pages":"0"}
HTTP/1.1 200 OK
Server: Payara Micro #badassfish
Content-Length: 0
X-Frame-Options: SAMEORIGIN

Calling http://localhost:8080/restapi/books
Payload {"title":"This is my second test book","description":"this is my second book description","isbn": "13xxxxxxxx", "publisher": "None
 Yet", "language":"English","author":"Hayri Cicek","price": "0.00","pages":"0"}
HTTP/1.1 200 OK
Server: Payara Micro #badassfish
Content-Length: 0
X-Frame-Options: SAMEORIGIN
```

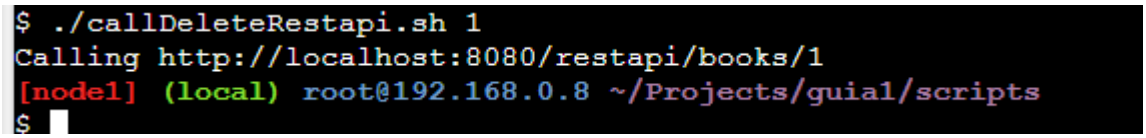
15. Obtener Libros:

```
$ ./callGetRestapi.sh
Calling http://localhost:8080/restapi/books
HTTP/1.1 200 OK
Server: Payara Micro #badassfish
Content-Type: application/json
Content-Length: 592
X-Frame-Options: SAMEORIGIN

[{"author":"Hayri Cicek","description":"this is my book description","id":1,"isbn":"12xxxxxxxx","language":"English","pages":0,"price":0.0
,"publisher":"None Yet","title":"This is my test book"},{"author":"Hayri Cicek","description":"this is my second book description","id":2,
"isbn":"13xxxxxxxx","language":"English","pages":0,"price":0.0,"publisher":"None Yet","title":"This is my second test book"},{"author":"Ta
deo Rivera-Pineda","description":"This is the book of Tadeo","id":3,"isbn":"14xxxxxxxx","language":"Spanish","pages":0,"price":0.0,"publis
her":"None Yet","title":"Xtecuan Book"}][node1] (local) root@192.168.0.8 ~/Projects/guia1/scripts
$
```

16. Modificar Libros:



17. Borrar Libros:



18. ¡Felicidades ha completado la parte 2 de la guía empleando contenedores para la compilación y ejecución de cargas de trabajo!
19. Explore en la ayuda del comando **docker build** para ver que flags/opciones han sido empleadas para construir nuestra primera imagen con el siguiente comando << ***docker build --help*** >>

```
Usage:  docker build [OPTIONS] PATH | URL | -

Build an image from a Dockerfile

Options:
    --add-host list          Add a custom host-to-IP mapping (host:ip)
    --build-arg list         Set build-time variables
    --cache-from strings     Images to consider as cache sources
    --cgroup-parent string   Optional parent cgroup for the container
    --compress               Compress the build context using gzip
    --cpu-period int         Limit the CPU CFS (Completely Fair Scheduler) period
    --cpu-quota int          Limit the CPU CFS (Completely Fair Scheduler) quota
 -c, --cpu-shares int        CPU shares (relative weight)
    --cpuset-cpus string     CPUs in which to allow execution (0-3, 0,1)
    --cpuset-mems string     MEMs in which to allow execution (0-3, 0,1)
    --disable-content-trust  Skip image verification (default true)
 -f, --file string           Name of the Dockerfile (Default is 'PATH/Dockerfile')
    --force-rm               Always remove intermediate containers
    --iidfile string         Write the image ID to the file
    --isolation string       Container isolation technology
    --label list             Set metadata for an image
 -m, --memory bytes          Memory limit
    --memory-swap bytes      Swap limit equal to memory plus swap: '-1' to enable unlimited swap
    --network string         Set the networking mode for the RUN instructions during build (default
"default")
    --no-cache               Do not use cache when building the image
    --pull                   Always attempt to pull a newer version of the image
 -q, --quiet                 Suppress the build output and print image ID on success
    --rm                     Remove intermediate containers after a successful build (default true)
    --security-opt strings   Security options
```

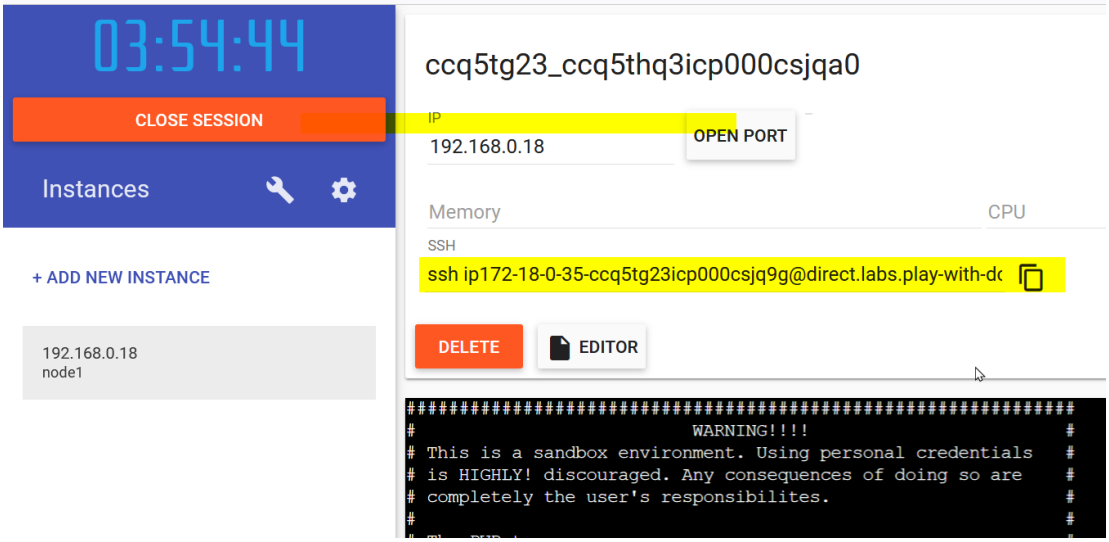
--shm-size bytes	Size of /dev/shm
-t, --tag list	Name and optionally a tag in the 'name:tag' format
--target string	Set the target build stage to build.
--ulimit ulimit	Ulimit options (default [])

Nota 4: para ver el listado de imágenes presentes en el runtime de docker hacer << ***docker images*** >>

```
[node1] (local) root@192.168.0.8 ~/Projects/guia1/scripts
$ docker images
REPOSITORY      TAG                 IMAGE ID       CREATED          SIZE
mipayara        latest              fb01ce934d82   58 minutes ago   255MB
payara/micro    latest              2b2266e1373c   11 days ago      245MB
maven           3.8.6-openjdk-11    6c3ab1faec76   7 weeks ago      664MB
[node1] (local) root@192.168.0.8 ~/Projects/guia1/scripts
$
```

Parte 3. Ejecutando Microservicio con Angular

1. Para esta parte emplearemos una conexión SSH hacia el nodo de Play with Docker, podemos utilizar una terminal ***cmd*** o ***powershell*** de Windows, en la pantalla de la sesión se obtiene el enlace de conexión:



2. Ejecutamos el comando:

<< ***ssh ip172-18-0-35-ccq5tg23icp000csjq9g@direct.labs.play-with-docker.com*** >> desde una interfaz ***cmd o powershell***

```
PowerShell 7.3.0-preview.8
PS C:\Users\xtecuan> ssh ip172-18-0-35-ccq5tg23icp000csjq9g@direct.labs.play-with-docker.com
Connecting to 52.188.144.191:8022
#################################################################
#                          WARNING!!!!                          #
# This is a sandbox environment. Using personal credentials    #
# is HIGHLY! discouraged. Any consequences of doing so are     #
# completely the user's responsibilites.                       #
#                                                              #
# The PWD team.                                                #
#################################################################
[node1] (local) root@192.168.0.18 ~
$
```

3. Ingresamos a la carpeta de la aplicación Angular con el siguiente comando
   << ***cd $HOME/Projects/guia1/my-store*** >>

```
[node1] (local) root@192.168.0.18 ~
$ cd $HOME/Projects/guia1/my-store
[node1] (local) root@192.168.0.18 ~/Projects/guia1/my-store
$
```

4. Se construye la imagen del proyecto Angular ingresando el comando:

   << docker build -t my-store-img . >>

```
[node1] (local) root@192.168.0.18 ~/Projects/guia1/my-store
$ docker build -t my-store-img .
Sending build context to Docker daemon  485.4kB
Step 1/14 : FROM node:16.10.0 as build-stage
16.10.0: Pulling from library/node
5e7b6b7bd506: Downloading [==================================>                ]  32.32MB/50.44MB
fd67d668d691: Download complete
1ae016bc2687: Download complete
0b0af05a4d86: Downloading [=================>                                 ]   16.8MB/51.84MB
ca4689f0588c: Downloading [====>                                              ]  17.19MB/192.4MB
8c33de21d690: Waiting
f113b2c481db: Waiting
0f84649efc4d: Waiting
5990cbd9430a: Waiting
```

5. La construcción de la imagen Docker tomará un tiempo considerable debido a que se usa una construcción de varias etapas como se muestra en el archivo Dockerfile:

```
  GNU nano 6.3                                                Dockerfile
# Stage 0, "build-stage", based on Node.js, to build and compile the frontend
FROM node:16.10.0 as build-stage
WORKDIR /app
COPY package*.json /app/
RUN npm install
COPY ./ /app/
ARG configuration=production
RUN npm run build -- --output-path=./dist/out --configuration $configuration

# Stage 1, based on Nginx, to have only the compiled app, ready for production with Nginx
FROM nginx:1.15
#Copy ci-dashboard-dist
ARG project=my-store
RUN mkdir -p /usr/share/nginx/html/$project
COPY --from=build-stage /app/dist/out/ /usr/share/nginx/html/$project
#Copy default nginx configuration
COPY ./conf/nginx-custom.conf /etc/nginx/conf.d/default.conf
COPY ./conf/nginx.crt /etc/ssl/nginx.crt
COPY ./conf/nginx.key /etc/ssl/nginx.key
```

Nota 5: El primer stage del proceso de build se hace en un contenedor con node version 16.10.0 luego los objetos generados por este proceso se copian al contenedor con nginx 1.15 para ser servidos en el path

/my-store

6. Si el proceso de construcción termina exitosamente se tiene la siguiente vista:

```
 ---> 2a2d0d206d41
Step 8/14 : FROM nginx:1.15
1.15: Pulling from library/nginx
743f2d6c1f65: Pull complete
6bfc4ec4420a: Pull complete
688a776db95f: Pull complete
Digest: sha256:23b4dcdf0d34d4a129755fc6f52e1c6e23bb34ea011b315d87e193033bcd1b68
Status: Downloaded newer image for nginx:1.15
 ---> 53f3fd8007f7
Step 9/14 : ARG project=my-store
 ---> Running in 80776673d86c
Removing intermediate container 80776673d86c
 ---> 4e11f3d24f54
Step 10/14 : RUN mkdir -p /usr/share/nginx/html/$project
 ---> Running in f49a33314dbf
Removing intermediate container f49a33314dbf
 ---> e29ddb246cee
Step 11/14 : COPY --from=build-stage /app/dist/out/ /usr/share/nginx/html/$project
 ---> 82a0e8bd2575
Step 12/14 : COPY ./conf/nginx-custom.conf /etc/nginx/conf.d/default.conf
 ---> ee9249b6d3d0
Step 13/14 : COPY ./conf/nginx.crt /etc/ssl/nginx.crt
 ---> 0668d1f8f180
Step 14/14 : COPY ./conf/nginx.key /etc/ssl/nginx.key
 ---> 6fca3fca8d3a
Successfully built 6fca3fca8d3a
Successfully tagged my-store-img:latest
[node1] (local) root@192.168.0.18 ~/Projects/guia1/my-store
$
```

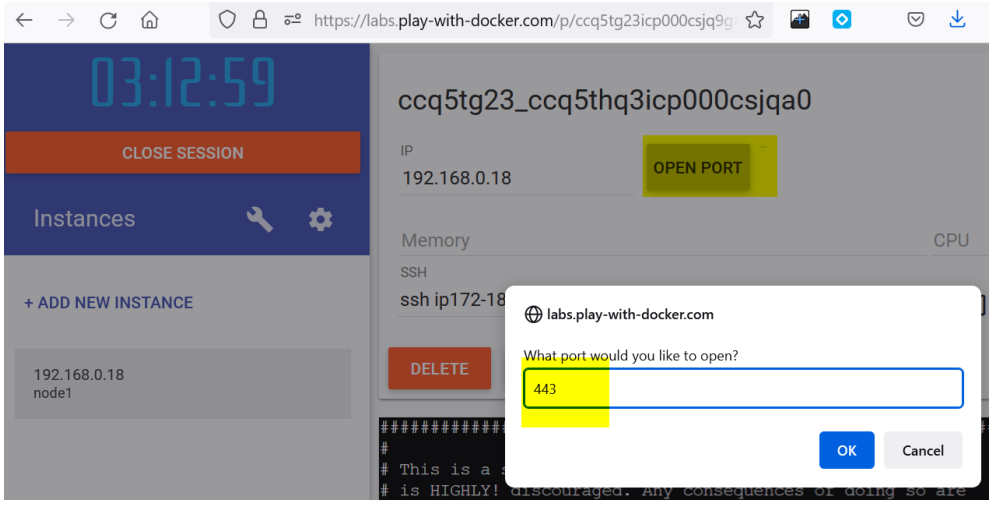
7. Se procede a ejecutar el contenedor con el siguiente comando:

   << docker run -dit --name my-store -p 443:4200 my-store-img:latest >>

```
[node1] (local) root@192.168.0.18 ~/Projects/guia1/my-store
$ docker run -dit --name my-store -p 443:4200 my-store-img:latest
fa1f1b35a2b5e310f34fc8b565991c590b26a989c1301be9680ac8bb74fd8288
[node1] (local) root@192.168.0.18 ~/Projects/guia1/my-store
$
```

```
[node1] (local) root@192.168.0.18 ~/Projects/guia1/my-store
$ docker ps
CONTAINER ID   IMAGE                  COMMAND                  CREATED              STATUS              PORTS
                  NAMES
fa1f1b35a2b5   my-store-img:latest    "nginx -g 'daemon of…"   About a minute ago   Up About a minute   80/tcp, 0.0.0
.0:443->4200/tcp   my-store
```

8. Se tiene que abrir el puerto 443 desde la pagina web de Play with Docker:

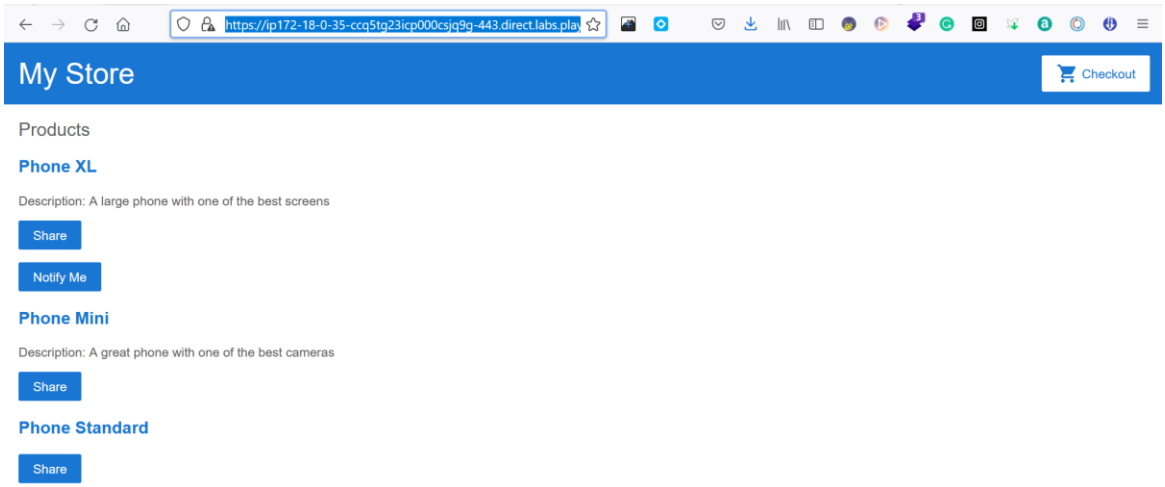


9. Automáticamente la página abrirá una url base pero con el protocolo http, hay que cambiar a https y agregar el path ***/my-store/,*** a manera de ejemplo la url usa el siguiente formato:

   https://ip172-18-0-35-ccq5tg23icp000csjq9g-443.direct.labs.play-with-docker.com/my-store/

   Se tiene que aceptar el certificado auto firmado para lograr ver la página.

10. La vista de la aplicación Angular es la siguiente:



11. ¡Felicidades ha logrado construir y ejecutar un contenedor con Angular!

Parte 4. Ejecutando Microservicio con .Net 7

1. Ingresamos al path ***aspnetapp*** dentro del repositorio con el siguiente comando:

<< ***cd $HOME/Projects/guia1/aspnetapp*** >>

```
[node1] (local) root@192.168.0.18 ~
$ cd $HOME/Projects/guia1/aspnetapp
[node1] (local) root@192.168.0.18 ~/Projects/guia1/aspnetapp
$ ls
Dockerfile      README.md       aspnetapp       aspnetapp.sln
[node1] (local) root@192.168.0.18 ~/Projects/guia1/aspnetapp
$
```

2. Construimos la aplicación con el siguiente comando:

<< ***docker build -t aspnetapp-img .*** >>

```
$ docker build -t aspnetapp-img .
Sending build context to Docker daemon  8.202MB
Step 1/10 : FROM mcr.microsoft.com/dotnet/sdk:7.0 AS build
 ---> 185d6e706adc
Step 2/10 : WORKDIR /source
 ---> Using cache
 ---> 6cb46e2049b8
Step 3/10 : COPY aspnetapp/*.csproj .
 ---> Using cache
 ---> cd6fa02ac7a1
Step 4/10 : RUN dotnet restore --use-current-runtime
 ---> Using cache
 ---> fd5fb5bcd3bb
Step 5/10 : COPY aspnetapp/. .
 ---> Using cache
 ---> 394ba472e530
Step 6/10 : RUN dotnet publish -c Release -o /app --use-current-runtime --self-contained false --no-restore
 ---> Using cache
 ---> a2610d92b45b
Step 7/10 : FROM mcr.microsoft.com/dotnet/aspnet:7.0
 ---> 8555b6b5252f
Step 8/10 : WORKDIR /app
 ---> Running in fac92376cf8d
Removing intermediate container fac92376cf8d
 ---> 32c7efe86eaa
Step 9/10 : COPY --from=build /app .
 ---> 4cd3c09675dc
Step 10/10 : ENTRYPOINT ["dotnet", "aspnetapp.dll"]
 ---> Running in ece0872a4b3e
```

3. Ejecutamos la aplicación con el siguiente comando:

<< ***docker run -dit --name aspnetapp-container -p 8081:80 aspnetapp-img:latest*** >>

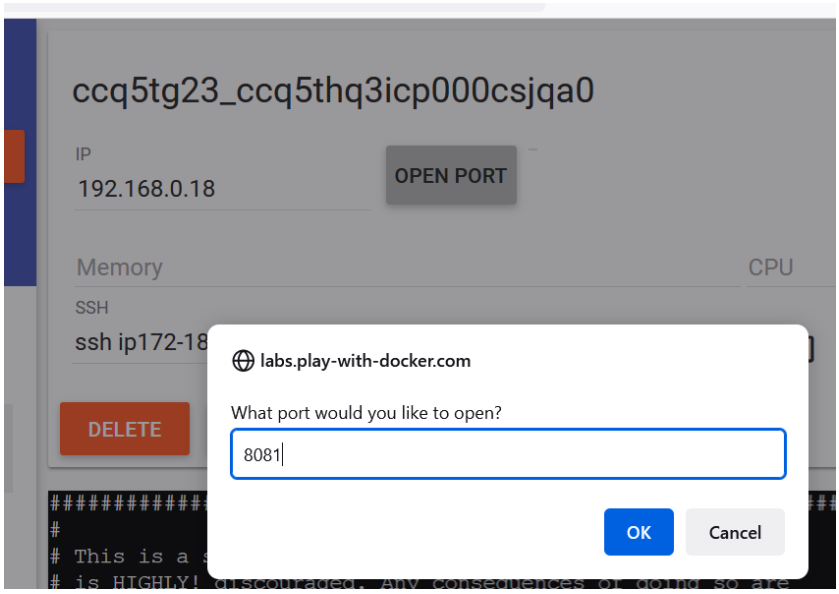
```
[node1] (local) root@192.168.0.18 ~/Projects/guia1/aspnetapp
$ docker run -dit --name aspnetapp-container -p 8081:80 aspnetapp-img:latest
0171ef17d1efbb4079827498577cdf99945a7f499cd5b2bc5fb12c38ba70e3f2
```

4. Revisamos el log de la aplicación con el comando:

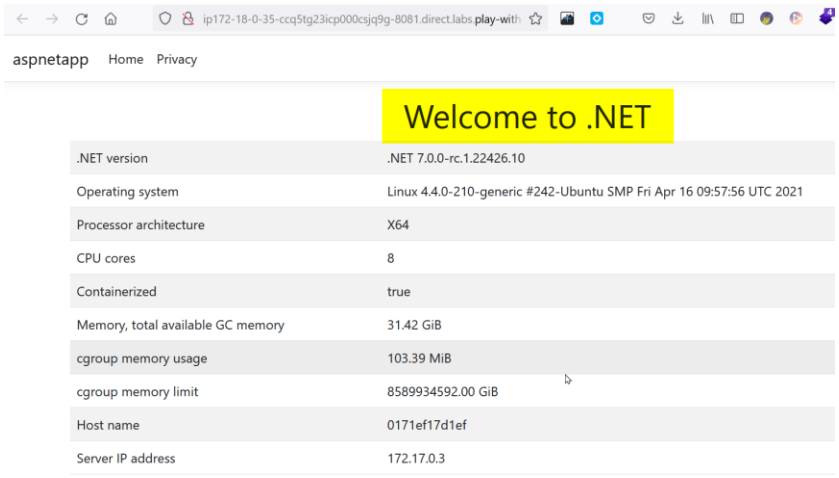
<< ***docker logs -f aspnetapp-container*** >>

```
$ docker logs -f aspnetapp-container
warn: Microsoft.AspNetCore.DataProtection.Repositories.FileSystemXmlRepository[60]
      Storing keys in a directory '/root/.aspnet/DataProtection-Keys' that may not be persisted outside of the conta
iner. Protected data will be unavailable when container is destroyed.
warn: Microsoft.AspNetCore.DataProtection.KeyManagement.XmlKeyManager[35]
      No XML encryptor configured. Key {733f9382-d792-405f-898f-7761d80327bd} may be persisted to storage in unencry
pted form.
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: http://[::]:80
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: /app
```

5. Desde la página Web de Play with Docker abrimos el puerto 8081:



6. El Resultado sería el siguiente:



7. ¡Felicidades ha logrado publicar la aplicación en .Net!

Parte 5. Ejercicio de Desafío 1

1. Instalar un contenedor con mariadb o postgresql empleando Volumenes.
2. Crear una base de datos de ejemplo en cualquiera de las dos bases.
3. Ingresar datos de ejemplo en las tablas de la base de datos.
4. Crear una API RESTful que consuma los datos de ejemplo de la base de datos en cualquier tecnología de las que se han empleado en esta guía de laboratorio.