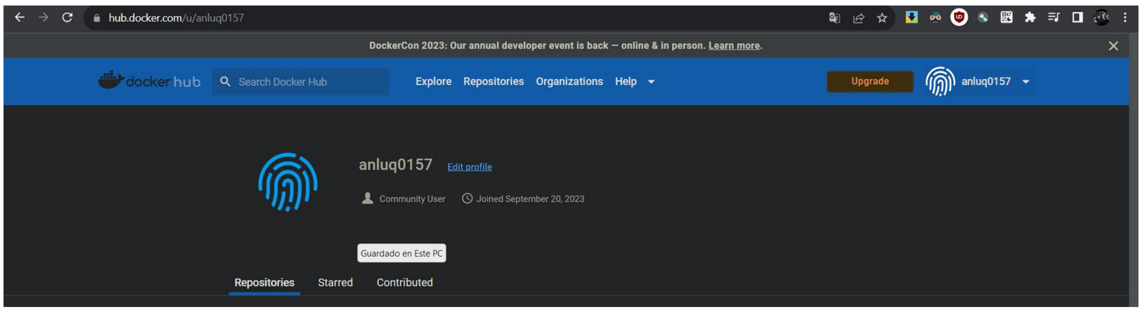
Lab Report 3 (Docker) - Antonio Manuel Luque Molina

- Docker is an open-source containerization platform.
- It allows developers to package their applications into executable components combining application source code with the required OS libraries and dependencies to run the code in any environment.
- Docker is a tool that enables users to run, build, deploy, update and stop containers using commands and automation through a single API.
- A Docker container image is a lightweight, standalone, executable package of software that includes everything needed to run an application: code, runtime, system tools, system libraries and settings.

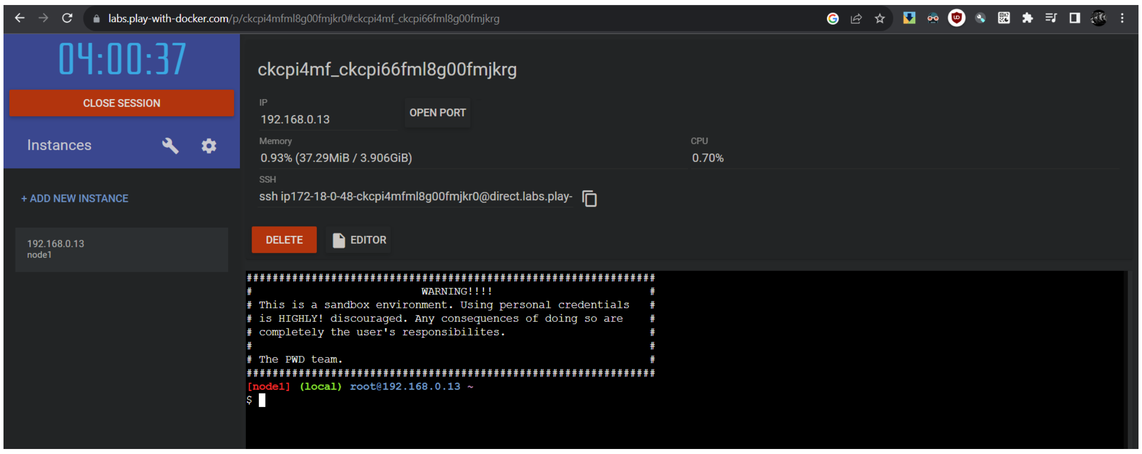
- Images – This is the blueprint of the applications that forms the basis of a container. It acts as a set of instructions to build a docker container. An image is like a snapshot in VM environments.
- Containers - Created from Docker images and runs the actual application.
- Docker daemon – It is the background service running in the host which manages the building and running of docker containers.
- Docker Client - This is a CLI using which user talks to the daemon.
- Docker Hub - It is a registry of docker images. You can access all the publicly available images from here and use them to build a container. You can also publish your own images there.

Creating a Docker account:

We go to https://hub.docker.com/ and We put our UiT email id for example, then We choose the personal Plan that it is free, After the verification of the email We are free of use Docker.

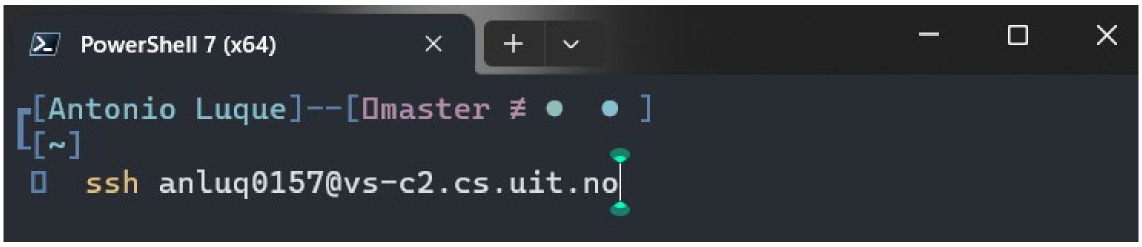


We can also use the web service provided by Docker "Play With Docker" that is a Docker playground wich allows you to run Docker commands:


labs.play-with-docker.com/p/ckcpi4mfml8g00fmjkr0#ckcpi4mf_ckcpi66fml8g00fmjkrg
04:00:37
CLOSE SESSION
Instances
+ ADD NEW INSTANCE
192.168.0.13
node1
ckcpi4mf_ckcpi66fml8g00fmjkrg
IP
192.168.0.13
OPEN PORT
Memory
0.93% (37.29MiB / 3.906GiB)
CPU
0.70%
SSH
ssh ip172-18-0-48-ckcpi4mfml8g00fmjkr0@direct.labs.play-
DELETE
EDITOR

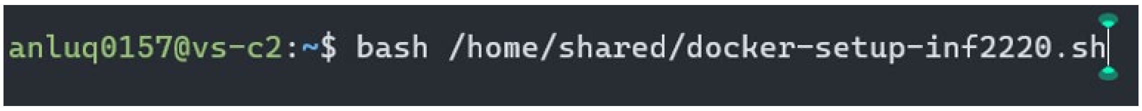
```
##############################################################
#                          WARNING!!!!                       #
# This is a sandbox environment. Using personal credentials  #
# is HIGHLY! discouraged. Any consequences of doing so are   #
# completely the user's responsibilites.                     #
#                                                            #
# The PWD team.                                              #
##############################################################
[node1] (local) root@192.168.0.13 ~
$
```

On the Server:



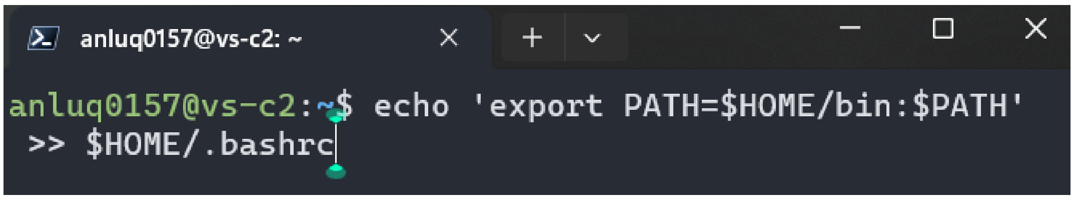
```
┌[Antonio Luque]─[master ≢ ●  ● ]
└[~]
  ssh anluq0157@vs-c2.cs.uit.no
```

This command initiates an SSH (Secure Shell) connection to the server at vs-c2.cs.uit.no.



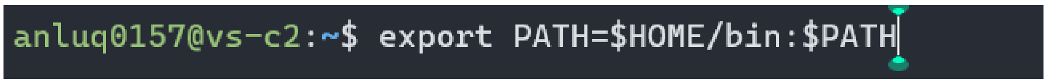
```
anluq0157@vs-c2:~$ bash /home/shared/docker-setup-inf2220.sh
```

This command executes a shell script called docker-setup-inf2220.sh located in the /home/shared/ directory. This script likely contains instructions to install and configure Docker.



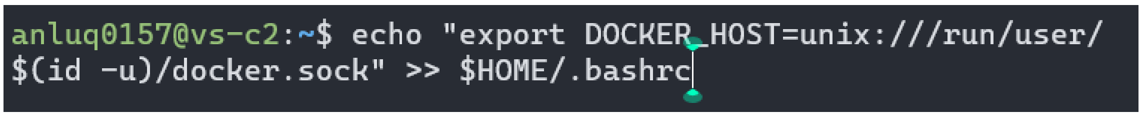
```
anluq0157@vs-c2:~$ echo 'export PATH=$HOME/bin:$PATH'
 >> $HOME/.bashrc
```

This line appends export PATH=$HOME/bin:$PATH to our .bashrc file. It adds the ~/bin directory to our PATH environment variable, which allows us to run executables located in that directory.



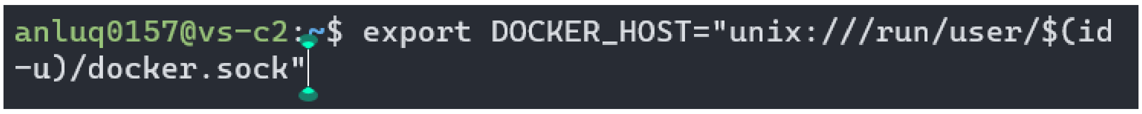
```
anluq0157@vs-c2:~$ export PATH=$HOME/bin:$PATH
```

This line updates the PATH environment variable for our current shell session to include the ~/bin directory.



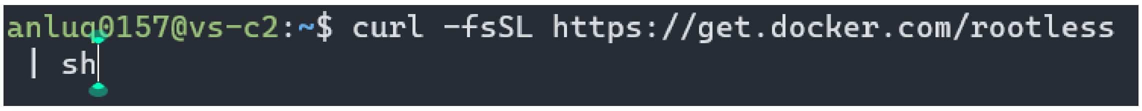
```
anluq0157@vs-c2:~$ echo "export DOCKER_HOST=unix:///run/user/
$(id -u)/docker.sock" >> $HOME/.bashrc
```

This line appends export DOCKER_HOST=unix:///run/user/$(id -u)/docker.sock to our .bashrc file. It sets the DOCKER_HOST environment variable to connect Docker to a Unix socket.



```
anluq0157@vs-c2:~$ export DOCKER_HOST="unix:///run/user/$(id
-u)/docker.sock"
```

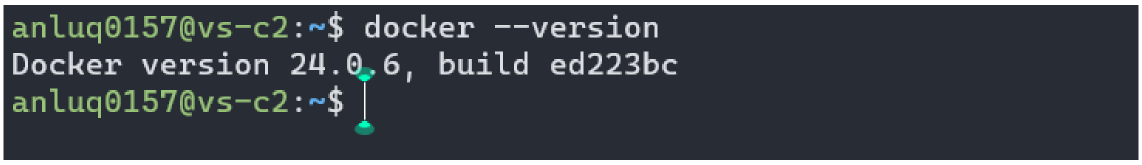
This line updates the DOCKER_HOST environment variable for your current shell session.



```
anluq0157@vs-c2:~$ curl -fsSL https://get.docker.com/rootless
 | sh
```

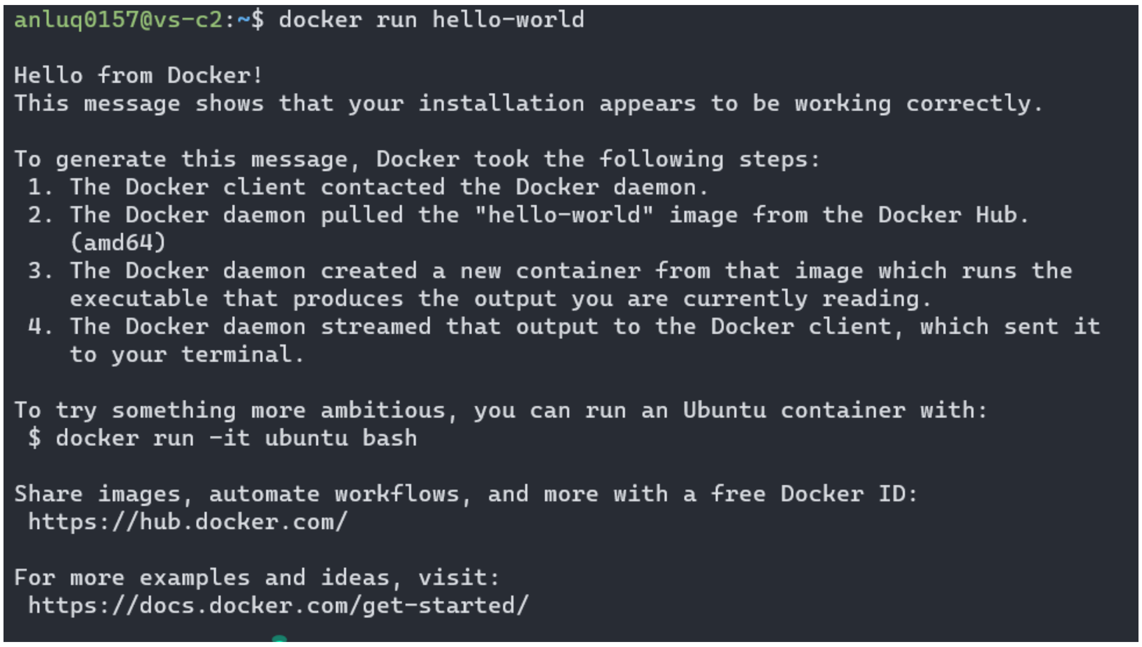
This command uses curl to download and execute a script from https://get.docker.com/rootless. This script is responsible for installing Docker in a rootless mode, which is a more secure way to run Docker without requiring root privileges.

Commands:



```
anluq0157@vs-c2:~$ docker --version
Docker version 24.0.6, build ed223bc
anluq0157@vs-c2:~$
```

This command checks our version of docker.



```
anluq0157@vs-c2:~$ docker run hello-world

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/
```

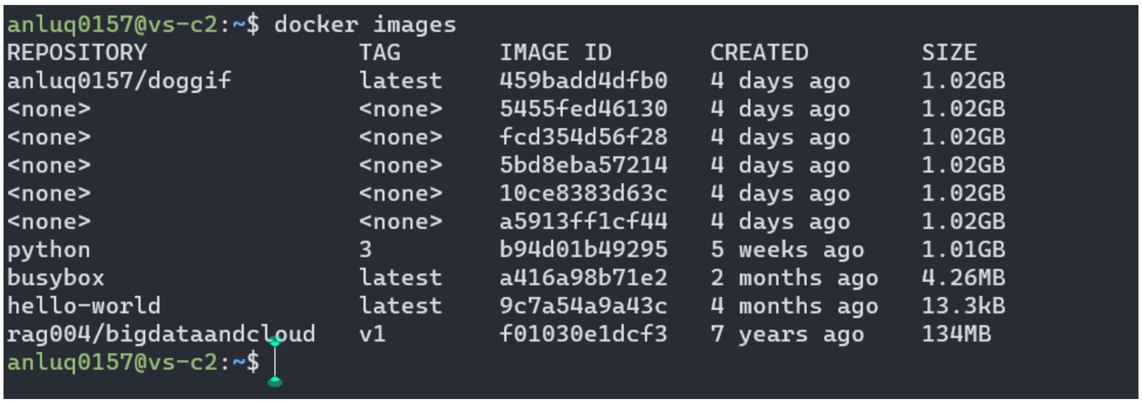
Docker checks if the "hello-world" image is available locally on our system. If it's not available locally, Docker will attempt to download it from the Docker Hub, which is a central repository for Docker images.

Once the image is available (either locally or after downloading), Docker runs a container based on that image.

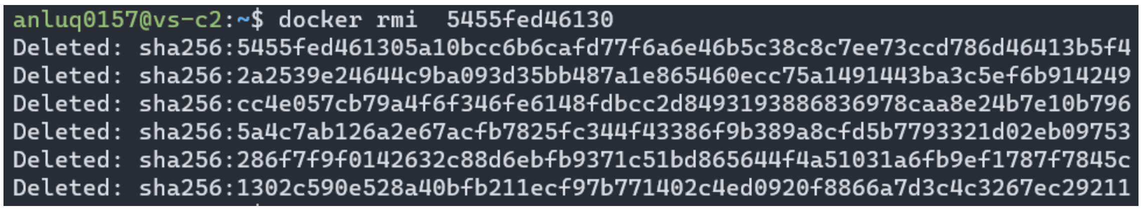
The "hello-world" image, in this case, is a very simple image that contains a minimal program. When you run it, the program does the following:

It prints a welcome message to the terminal, indicating that Docker is installed and functioning correctly.

After displaying this information, the container exits. It was created solely for the purpose of providing a simple demonstration and verifying that Docker can run containers successfully.

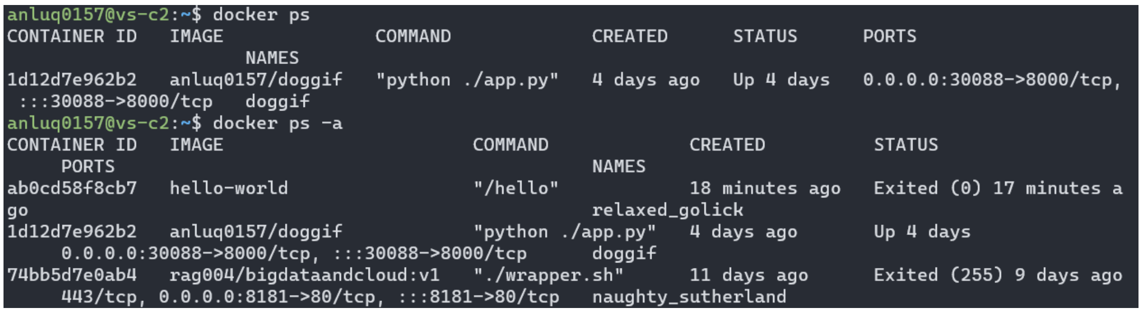


```
anluq0157@vs-c2:~$ docker images
REPOSITORY               TAG       IMAGE ID       CREATED         SIZE
anluq0157/doggif         latest    459badd4dfb0   4 days ago      1.02GB
<none>                   <none>    5455fed46130   4 days ago      1.02GB
<none>                   <none>    fcd354d56f28   4 days ago      1.02GB
<none>                   <none>    5bd8eba57214   4 days ago      1.02GB
<none>                   <none>    10ce8383d63c   4 days ago      1.02GB
<none>                   <none>    a5913ff1cf44   4 days ago      1.02GB
python                   3         b94d01b49295   5 weeks ago     1.01GB
busybox                  latest    a416a98b71e2   2 months ago    4.26MB
hello-world              latest    9c7a54a9a43c   4 months ago    13.3kB
rag004/bigdataandcloud   v1        f01030e1dcf3   7 years ago     134MB
anluq0157@vs-c2:~$
```



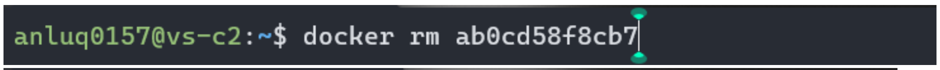
```
anluq0157@vs-c2:~$ docker rmi  5455fed46130
Deleted: sha256:5455fed461305a10bcc6b6cafd77f6a6e46b5c38c8c7ee73ccd786d46413b5f4
Deleted: sha256:2a2539e24644c9ba093d35bb487a1e865460ecc75a1491443ba3c5ef6b914249
Deleted: sha256:cc4e057cb79a4f6f346fe6148fdbcc2d8493193886836978caa8e24b7e10b796
Deleted: sha256:5a4c7ab126a2e67acfb7825fc344f43386f9b389a8cfd5b7793321d02eb09753
Deleted: sha256:286f7f9f0142632c88d6ebfb9371c51bd865644f4a51031a6fb9ef1787f7845c
Deleted: sha256:1302c590e528a40bfb211ecf97b771402c4ed0920f8866a7d3c4c3267ec29211
```

With this 2 commands We can see all the images that We have in our local system and We can delete the images that We don't need.

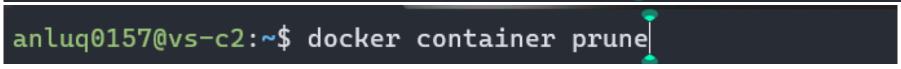


```
anluq0157@vs-c2:~$ docker ps
CONTAINER ID   IMAGE              COMMAND             CREATED      STATUS      PORTS
                     NAMES
1d12d7e962b2   anluq0157/doggif   "python ./app.py"   4 days ago   Up 4 days   0.0.0.0:30088->8000/tcp,
 :::30088->8000/tcp   doggif
anluq0157@vs-c2:~$ docker ps -a
CONTAINER ID   IMAGE                       COMMAND             CREATED          STATUS
     PORTS                                         NAMES
ab0cd58f8cb7   hello-world                 "/hello"            18 minutes ago   Exited (0) 17 minutes a
go                                                 relaxed_golick
1d12d7e962b2   anluq0157/doggif            "python ./app.py"   4 days ago       Up 4 days
     0.0.0.0:30088->8000/tcp, :::30088->8000/tcp   doggif
74bb5d7e0ab4   rag004/bigdataandcloud:v1   "./wrapper.sh"      11 days ago      Exited (255) 9 days ago
     443/tcp, 0.0.0.0:8181->80/tcp, :::8181->80/tcp   naughty_sutherland
```

With this 2 commands We can see a list of all the container that We have in our local system.

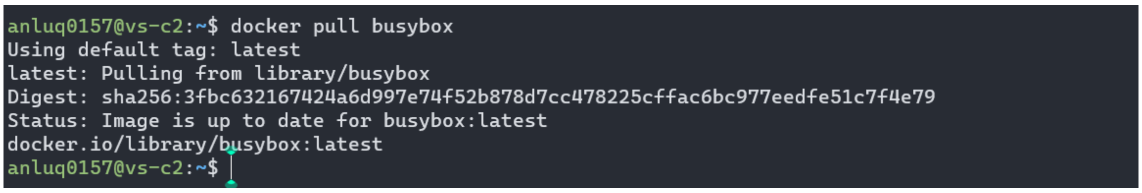


```
anluq0157@vs-c2:~$ docker rm ab0cd58f8cb7
```



```
anluq0157@vs-c2:~$ docker container prune
```

With this two commands We can remove a container by the container ID or We can remove all the containers

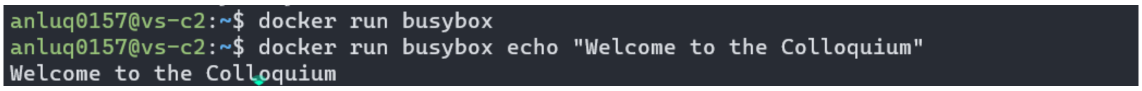


```
anluq0157@vs-c2:~$ docker pull busybox
Using default tag: latest
latest: Pulling from library/busybox
Digest: sha256:3fbc632167424a6d997e74f52b878d7cc478225cffac6bc977eedfe51c7f4e79
Status: Image is up to date for busybox:latest
docker.io/library/busybox:latest
anluq0157@vs-c2:~$
```

This command is used to download a Docker image named "busybox" from the Docker Hub, which is a central repository for Docker images.

The "busybox" image is a lightweight and minimalistic Linux distribution that is often used as a base image for creating other Docker containers.

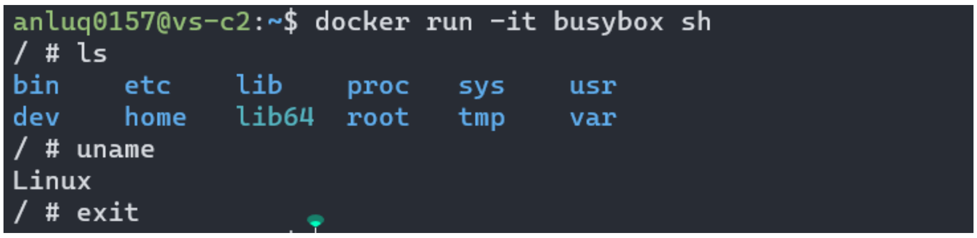
After running this command, you'll have a local copy of the "busybox" image on your system, and you can use it to create containers.



```
anluq0157@vs-c2:~$ docker run busybox
anluq0157@vs-c2:~$ docker run busybox echo "Welcome to the Colloquium"
Welcome to the Colloquium
```

When We run this two commands, The first one creates a new Docker container based on the "busybox" image. It runs a container using the "busybox" image as its base.

In the second one, a Docker container based on the "busybox" image is created, and immediately after that, it executes the "echo" command inside the container, which prints the message "Welcome to colloquium" to the container's standard output. After the message is printed, the container will exit because there are no long-running processes specified in the command.

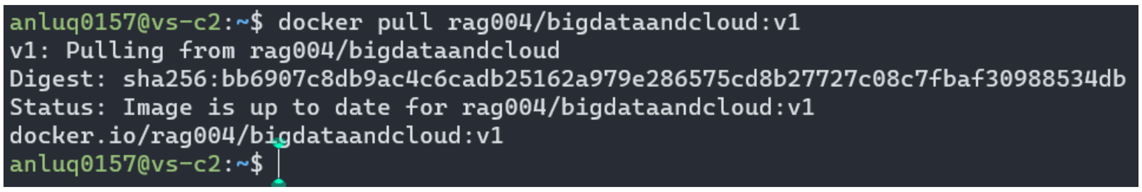


```
anluq0157@vs-c2:~$ docker run -it busybox sh
/ # ls
bin     etc     lib     proc    sys     usr
dev     home    lib64   root    tmp     var
/ # uname
Linux
/ # exit
```

Here, Docker creates a new container based on the "busybox" image. It starts an interactive shell session using "sh" as the shell interpreter.

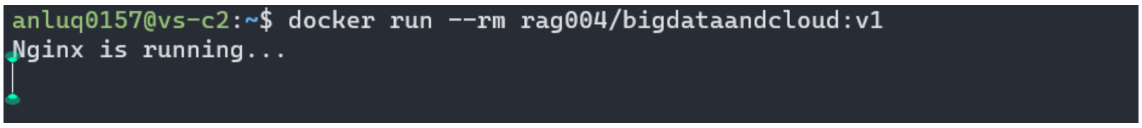
We will be placed inside the container, and We can now interact with the shell just like We would on a regular Linux system. We can run commands like "cd", "ls", and more in order to navigate the file system, and perform other shell operations.

Hosting a webapp in Docker:



```
anluq0157@vs-c2:~$ docker pull rag004/bigdataandcloud:v1
v1: Pulling from rag004/bigdataandcloud
Digest: sha256:bb6907c8db9ac4c6cadb25162a979e286575cd8b27727c08c7fbaf30988534db
Status: Image is up to date for rag004/bigdataandcloud:v1
docker.io/rag004/bigdataandcloud:v1
anluq0157@vs-c2:~$
```

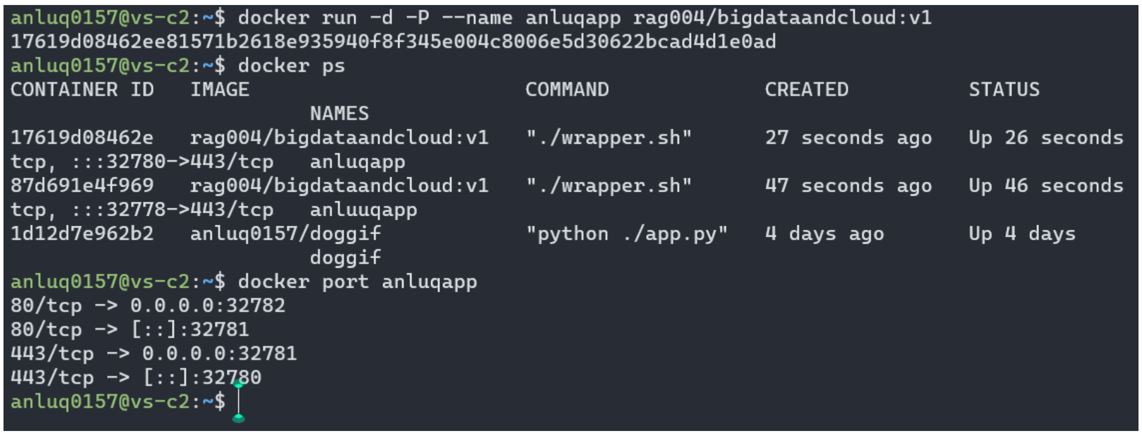
After running this command, We will have a local copy of the specified Docker image (rag004/bigdataandcloud:v1) on our system, and We can use it to create containers based on this image. This is a common step when working with Docker to acquire the necessary images for running containers in your environment.



```
anluq0157@vs-c2:~$ docker run --rm rag004/bigdataandcloud:v1
Nginx is running...
```

This command runs a container based on the specified image and removes the container automatically after it finishes executing its command.

If We want to interact with the Nginx server running inside the container, We can typically access it using a web browser or tools like curl. We may need to determine the IP address or port mapping of the container to access the Nginx web server, depending on how the container is configured; For this We will use the following commands after exiting with CTRL-D from the one before:



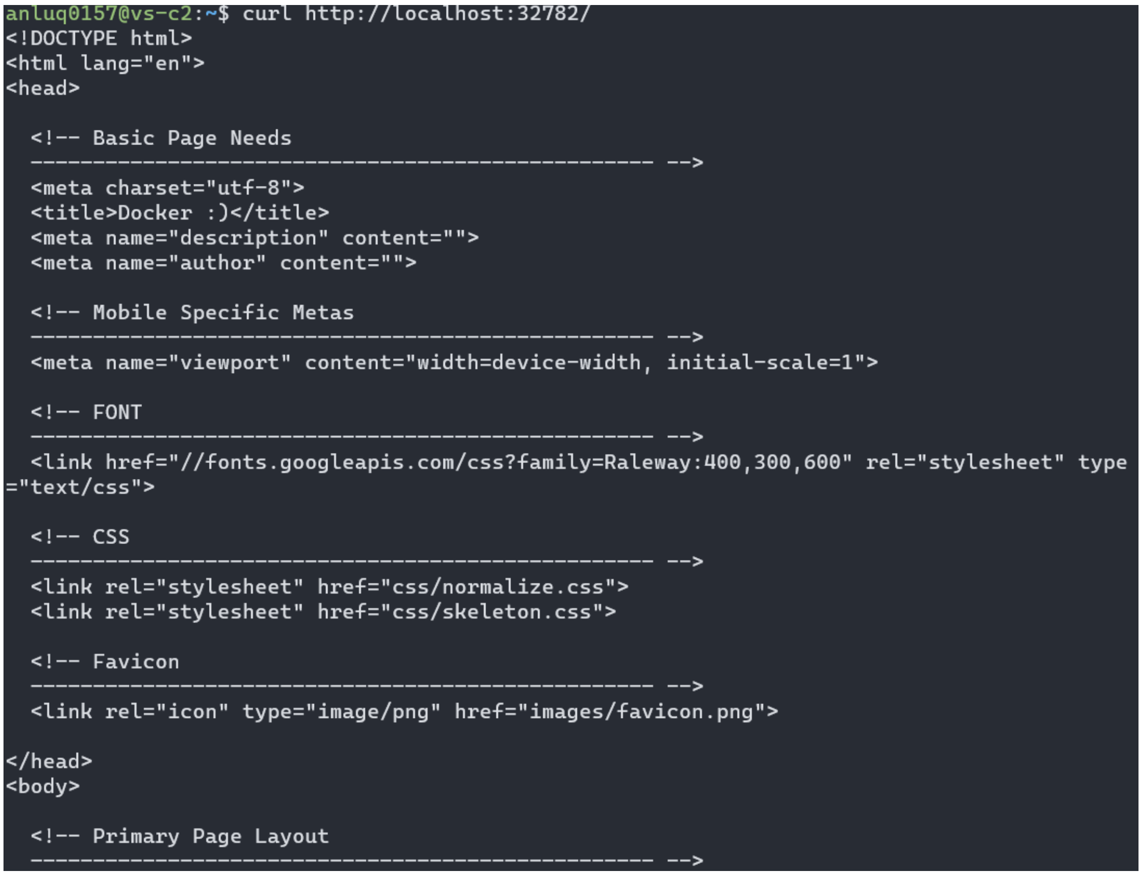
```
anluq0157@vs-c2:~$ docker run -d -P --name anluqapp rag004/bigdataandcloud:v1
17619d08462ee81571b2618e935940f8f345e004c8006e5d30622bcad4d1e0ad
anluq0157@vs-c2:~$ docker ps
CONTAINER ID   IMAGE                         COMMAND             CREATED          STATUS
                         NAMES
17619d08462e   rag004/bigdataandcloud:v1   "./wrapper.sh"      27 seconds ago   Up 26 seconds
tcp, :::32780->443/tcp   anluqapp
87d691e4f969   rag004/bigdataandcloud:v1   "./wrapper.sh"      47 seconds ago   Up 46 seconds
tcp, :::32778->443/tcp   anluuqapp
1d12d7e962b2   anluq0157/doggif              "python ./app.py"   4 days ago       Up 4 days
                         doggif
anluq0157@vs-c2:~$ docker port anluqapp
80/tcp -> 0.0.0.0:32782
80/tcp -> [::]:32781
443/tcp -> 0.0.0.0:32781
443/tcp -> [::]:32780
anluq0157@vs-c2:~$
```

In the first command the "-d" flag is to run the container in detached mode meaning you have access to the terminal and the container is running in background, "-P" publish ports for our container and "--name" provides a name to our container

Then, with "docker ps" We see our container running in the background and with "docker port anluqapp" We see the port of the container.

Checking if our webapp is working:

In Terminal:

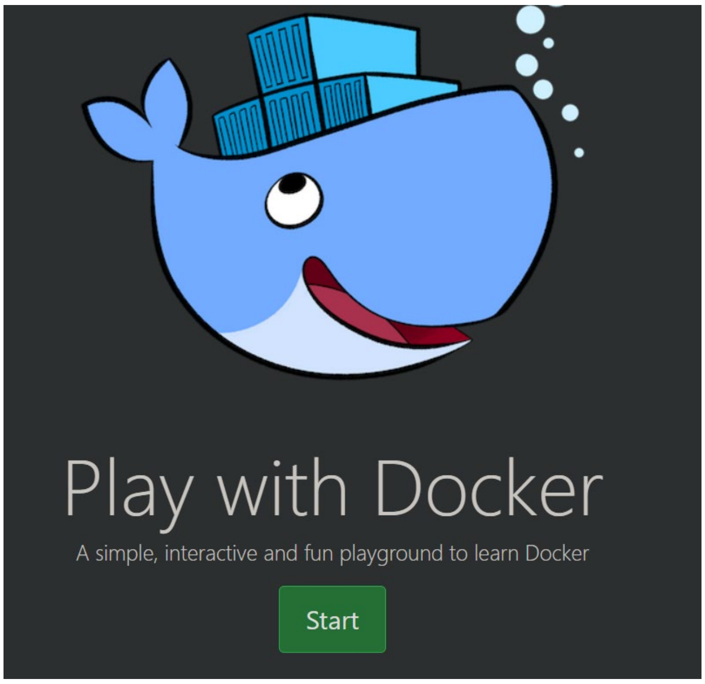


```
anluq0157@vs-c2:~$ curl http://localhost:32782/
<!DOCTYPE html>
<html lang="en">
<head>

  <!-- Basic Page Needs
  –––––––––––––––––––––––––––––––––––––––––––––––––– -->
  <meta charset="utf-8">
  <title>Docker :)</title>
  <meta name="description" content="">
  <meta name="author" content="">

  <!-- Mobile Specific Metas
  –––––––––––––––––––––––––––––––––––––––––––––––––– -->
  <meta name="viewport" content="width=device-width, initial-scale=1">

  <!-- FONT
  –––––––––––––––––––––––––––––––––––––––––––––––––– -->
  <link href="//fonts.googleapis.com/css?family=Raleway:400,300,600" rel="stylesheet" type
="text/css">

  <!-- CSS
  –––––––––––––––––––––––––––––––––––––––––––––––––– -->
  <link rel="stylesheet" href="css/normalize.css">
  <link rel="stylesheet" href="css/skeleton.css">

  <!-- Favicon
  –––––––––––––––––––––––––––––––––––––––––––––––––– -->
  <link rel="icon" type="image/png" href="images/favicon.png">

</head>
<body>

  <!-- Primary Page Layout
  –––––––––––––––––––––––––––––––––––––––––––––––––– -->
```

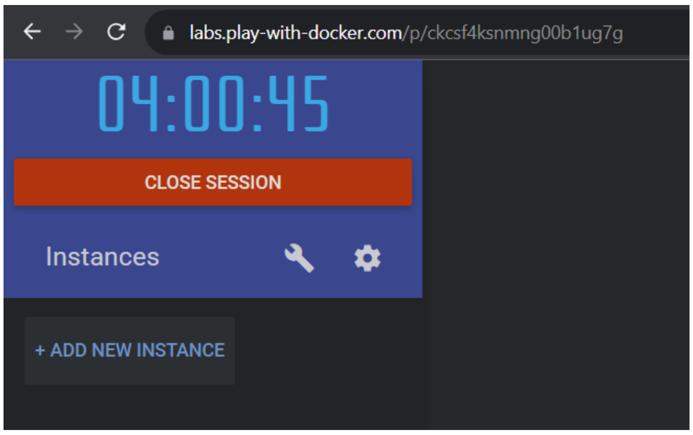
We can stop the container writing: **docker stop anluqapp**

On Play with Docker:

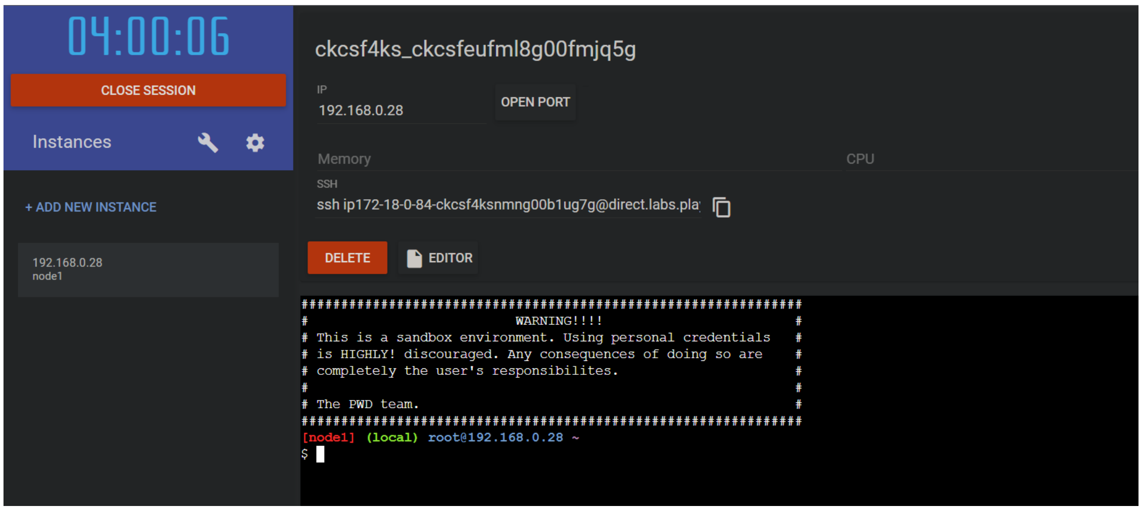
We go to the link: https://labs.play-with-docker.com/



We click on Start,



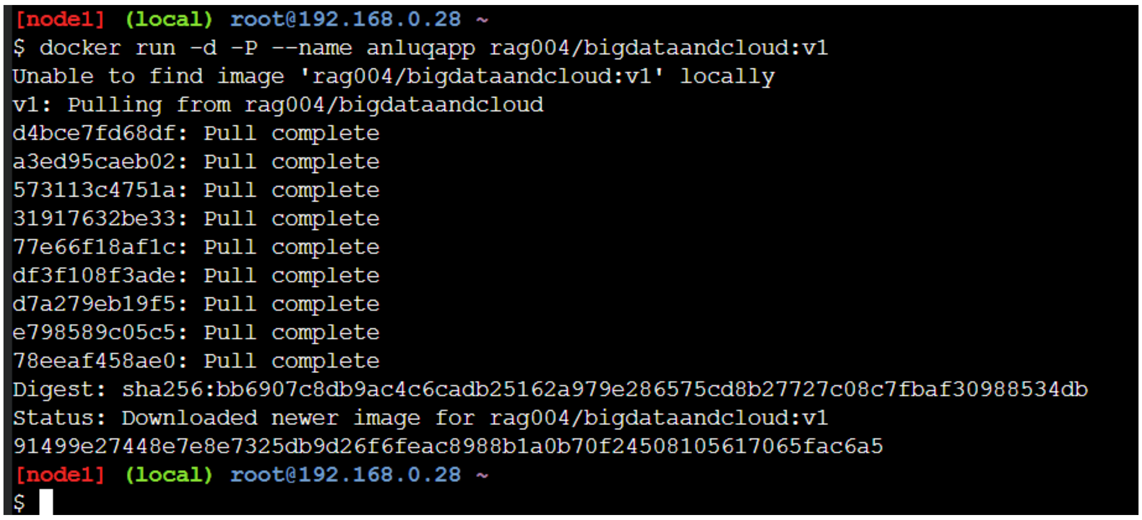
We click on Add New Instance,



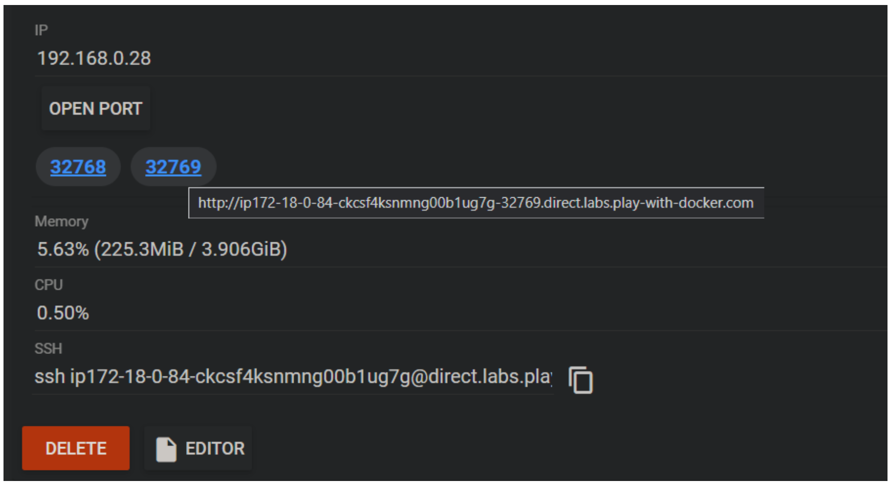
```
###############################################################
#                          WARNING!!!!                        #
# This is a sandbox environment. Using personal credentials   #
# is HIGHLY! discouraged. Any consequences of doing so are    #
# completely the user's responsibilites.                      #
#                                                             #
# The PWD team.                                               #
###############################################################
[node1] (local) root@192.168.0.28 ~
$
```

This creates a VM instance for you to run docker commands.

Now We will do all the steps We did in the server:

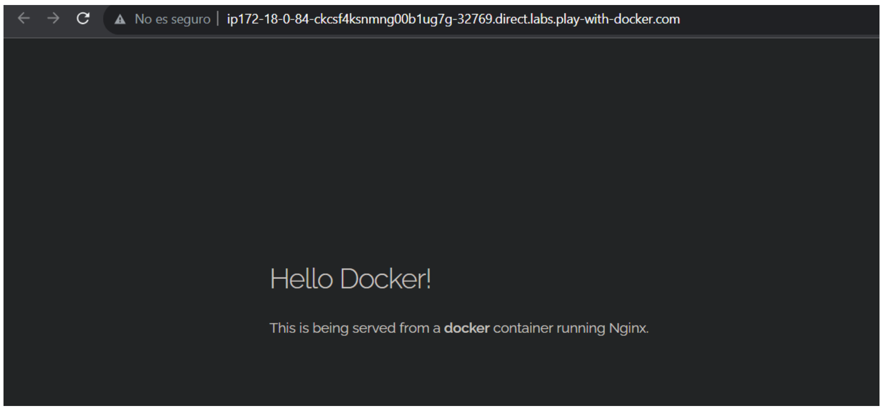


```
[node1] (local) root@192.168.0.28 ~
$ docker run -d -P --name anluqapp rag004/bigdataandcloud:v1
Unable to find image 'rag004/bigdataandcloud:v1' locally
v1: Pulling from rag004/bigdataandcloud
d4bce7fd68df: Pull complete
a3ed95caeb02: Pull complete
573113c4751a: Pull complete
31917632be33: Pull complete
77e66f18af1c: Pull complete
df3f108f3ade: Pull complete
d7a279eb19f5: Pull complete
e798589c05c5: Pull complete
78eeaf458ae0: Pull complete
Digest: sha256:bb6907c8db9ac4c6cadb25162a979e286575cd8b27727c08c7fbaf30988534db
Status: Downloaded newer image for rag004/bigdataandcloud:v1
91499e27448e7e8e7325db9d26f6feac8988b1a0b70f24508105617065fac6a5
[node1] (local) root@192.168.0.28 ~
$
```



IP
192.168.0.28
OPEN PORT
32768 32769
http://ip172-18-0-84-ckcsf4ksnmng00b1ug7g-32769.direct.labs.play-with-docker.com
Memory
5.63% (225.3MiB / 3.906GiB)
CPU
0.50%
SSH
ssh ip172-18-0-84-ckcsf4ksnmng00b1ug7g@direct.labs.pla

Here We click in the open port 32769 in order to see the webpage;

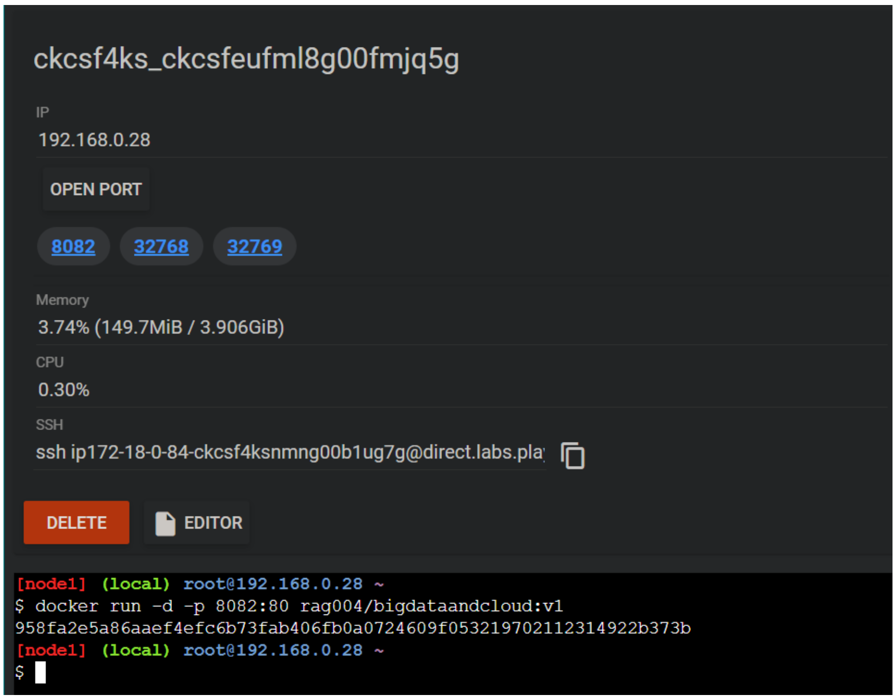


No es seguro | ip172-18-0-84-ckcsf4ksnmng00b1ug7g-32769.direct.labs.play-with-docker.com

Hello Docker!

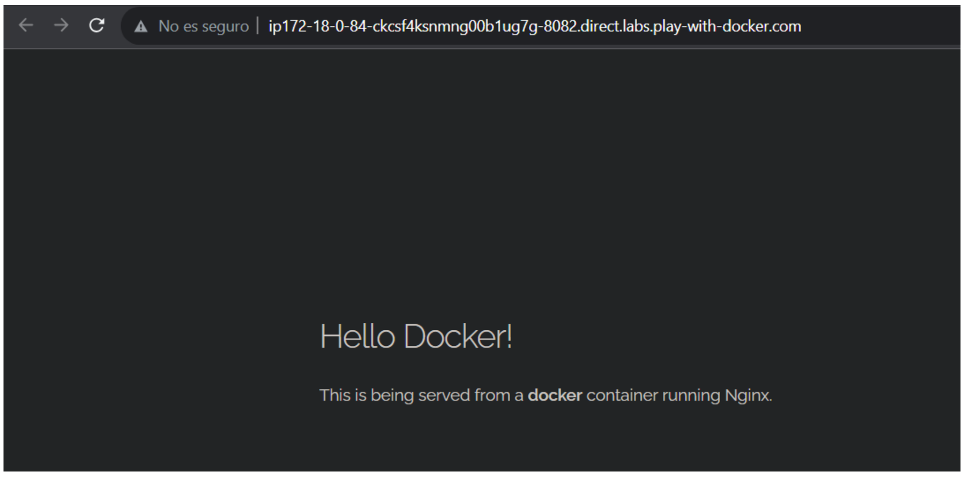
This is being served from a **docker** container running Nginx.

Also, We can specify a custom port for our container:



ckcsf4ks_ckcsfeufml8g00fmjq5g
IP
192.168.0.28
OPEN PORT
8082 32768 32769
Memory
3.74% (149.7MiB / 3.906GiB)
CPU
0.30%
SSH
ssh ip172-18-0-84-ckcsf4ksnmng00b1ug7g@direct.labs.pla

```
[node1] (local) root@192.168.0.28 ~
$ docker run -d -p 8082:80 rag004/bigdataandcloud:v1
958fa2e5a86aaef4efc6b73fab406fb0a0724609f053219702112314922b373b
[node1] (local) root@192.168.0.28 ~
$ 
```


No es seguro | ip172-18-0-84-ckcsf4ksnmng00b1ug7g-8082.direct.labs.play-with-docker.com

Hello Docker!

This is being served from a **docker** container running Nginx.

This is all! Thank you for reading!

Best Regards,

Antonio.