

Case Study: Implementing Port Forwarding with Docker.

Introduction:

Port forwarding in Docker plays a crucial role in enabling communication between containers and the host system or between containers themselves. In this case study, we'll explore how to set up port forwarding for Apache HTTP Server (httpd) and Ngnix containers.

Example 1: Running Apache HTTP Server Inside a Docker Container

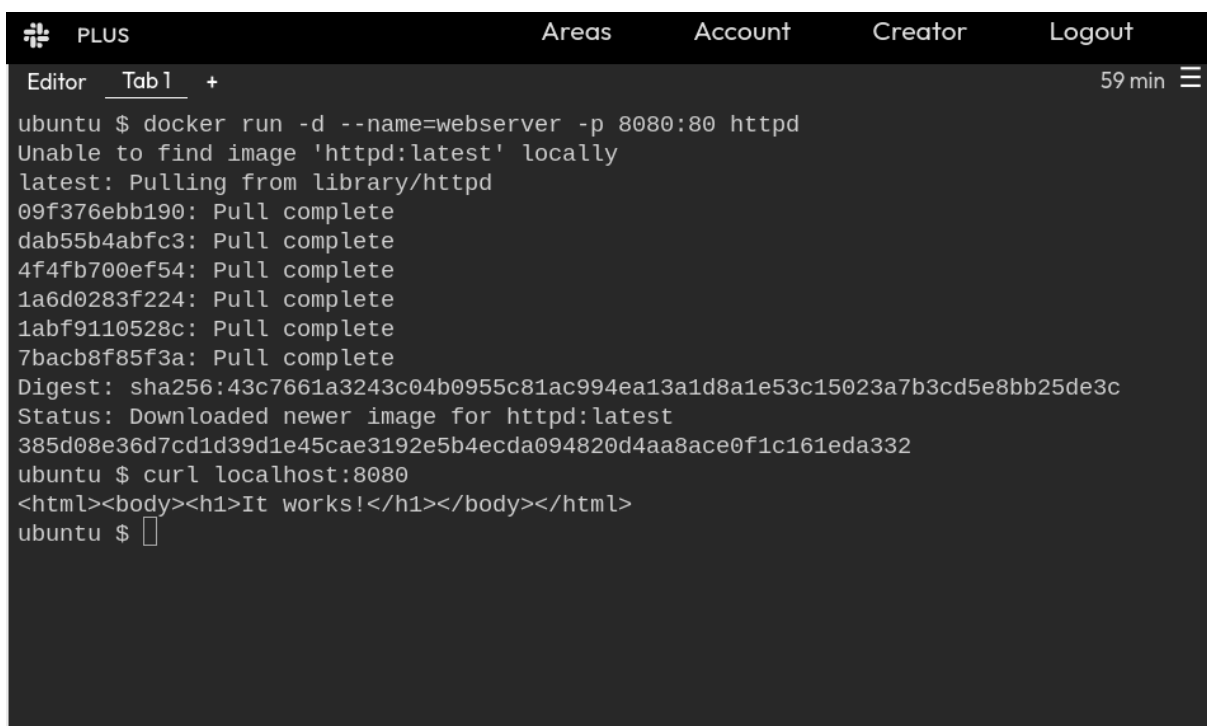
Scenario:

A web developer wants to run a website using Apache HTTP Server inside a Docker container and access it from the host system's web browser.

Implementation:

1. Run the Docker Container with Port Forwarding:

```
...  
docker run -dit --name=webserver -p 8080:80 httpd  
...
```



The screenshot shows a terminal window with a dark background. At the top, there is a navigation bar with a hamburger menu icon, the word 'PLUS', and tabs for 'Areas', 'Account', 'Creator', and 'Logout'. Below the navigation bar, there is a tab labeled 'Editor' and 'Tab 1' with a plus sign. The terminal output shows the following commands and their results:

```
ubuntu $ docker run -d --name=webserver -p 8080:80 httpd  
Unable to find image 'httpd:latest' locally  
latest: Pulling from library/httpd  
09f376ebb190: Pull complete  
dab55b4abfc3: Pull complete  
4f4fb700ef54: Pull complete  
1a6d0283f224: Pull complete  
1abf9110528c: Pull complete  
7bacb8f85f3a: Pull complete  
Digest: sha256:43c7661a3243c04b0955c81ac994ea13a1d8a1e53c15023a7b3cd5e8bb25de3c  
Status: Downloaded newer image for httpd:latest  
385d08e36d7cd1d39d1e45cae3192e5b4ecda094820d4aa8ace0f1c161eda332  
ubuntu $ curl localhost:8080  
<html><body><h1>It works!</h1></body></html>  
ubuntu $
```

2. Access the Apache Server:

Open a web browser on the host system and navigate to <http://localhost:8080> to access the website served by the Apache HTTP Server running inside the Docker container.

Access HTTP services which run in your environment

The services need to run on all interfaces (like 0.0.0.0) and not just localhost

Host 1

Common Ports

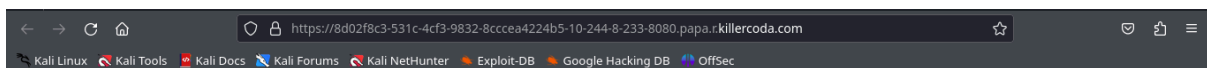
80

8080

Custom Ports

8080

Access



It works!

Example 2: Running Ngnix Inside a Docker Container

Scenario:

An organization wants to deploy a website using NGINX inside a Docker container and ensure it is accessible from the host system's web browser.

Implementation:

1.Run the Docker Container with Port Forwarding::

...

```
docker run -d --name=ng -p 9090:80 nginx
```

...

PLUS

Areas

Account

Creator

Logout

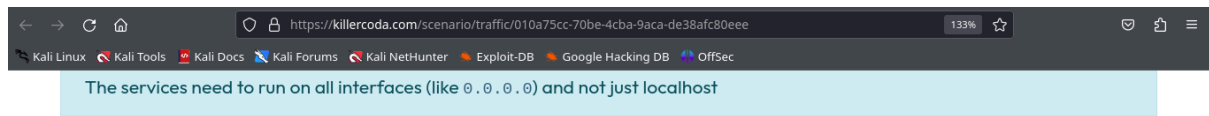
Editor Tab 1 +

37 min

```
ubuntu $ docker run -d --name=mysever -p 9090:80 nginx
9ad4ea1cab53ba68ce74d6820ccb27ac54753758b136774bd2bbf992158a4c12
ubuntu $ docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PO
RTS           NAMES
9ad4ea1cab53   nginx    "/docker-entrypoint...." 7 seconds ago  Up 6 seconds  0.
0.0.0:9090->80/tcp, :::9090->80/tcp  mysever
ubuntu $
```

2. Access the NGINX Server:

Open a web browser on the host system and navigate to <http://localhost:9090> to access the website served by NGINX running inside the Docker container.



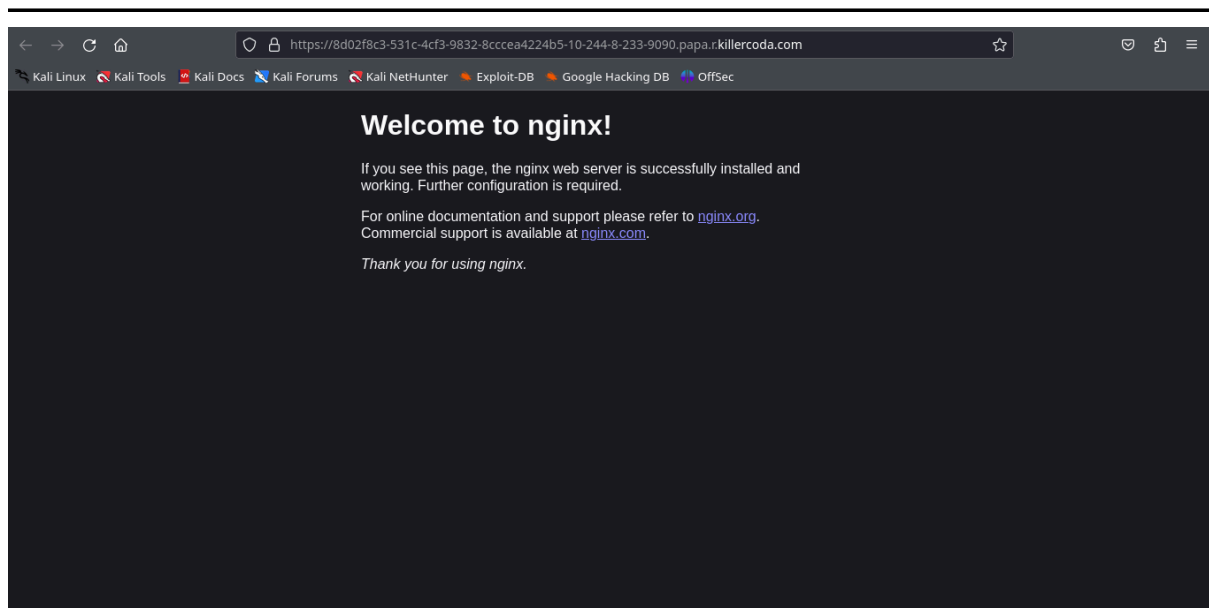
Host 1

Common Ports

80 8080

Custom Ports

9090 Access



Conclusion:

Port forwarding in Docker simplifies the process of accessing services running inside containers from the host system or other containers. By following the examples provided in this case study, users can effectively set up port forwarding for Apache HTTP Server and Nginx containers, enabling seamless communication between containers and the host system.