Container Network - Running an NGINX Web Server in a Docker Container

Running an NGINX Web Server in a Docker Container
This guide provides step-by-step instructions to set up and run an NGINX web server inside a Docker container. NGINX is a powerful web server and reverse proxy widely used for serving static content, load balancing, and more. Docker is a platform that allows you to easily create, deploy, and run applications in containers.

alt text

In this guide, we will demonstrate how to set up and run an NGINX web server inside a Docker container. By following these steps, you will learn how to pull the NGINX Docker image, configure a simple HTML file to be served by NGINX, and run the web server inside a container. Additionally, we will cover managing the NGINX container, including starting, stopping, viewing logs, and removing the container. This setup allows for a flexible and efficient way to deploy a web server using Docker's containerization capabilities.

what is Ngins explain in simple manner with basics

What is NGINX? (Explained Simply)

NGINX (pronounced "engine-x") is a **web server** that delivers websites and web applications to users. It is also used as a **reverse proxy**, **load balancer**, and **cache** to improve website performance and security.

Basic Concepts:

1. **Web Server** – It takes requests (like opening a website) and serves the appropriate web page to the user.
2. **Reverse Proxy** – It sits between users and backend servers, forwarding requests to the right place.
3. **Load Balancer** – It distributes incoming traffic across multiple servers to prevent overload.
4. **Caching** – It saves frequently accessed web pages, making them load faster.

Why is NGINX Popular?

- **Fast** – It handles multiple requests at once efficiently.
- **Lightweight** – Uses fewer system resources than traditional web servers.
- **Scalable** – Works well for small and large websites.
- **Secure** – Can block harmful traffic and protect websites.

How NGINX Works?

1. A user enters a website URL (e.g., `example.com`).
2. NGINX receives the request and finds the right content (a webpage, image, video, etc.).
3. It sends the content back to the user's browser.

NGINX is widely used by big companies like Netflix, Instagram, and Airbnb to manage traffic efficiently. 🚀

explain these :
2. Create a Directory for NGINX Content
Create a directory on your system to hold the NGINX configuration files and web content:

mkdir -p ~/nginx/html

This command creates a directory structure to store NGINX-related files.

- `mkdir -p ~/nginx/html`
  - `mkdir` → Creates a new directory.
  - `-p` → Ensures parent directories (`nginx`) are created if they don't exist.
  - `~/nginx/html` → The full path where NGINX will store its web content (HTML files).

This directory will hold the website files that NGINX serves when running inside a Docker container.

root@916dcf806f740625:~/code# docker pull nginx
Using default tag: latest
latest: Pulling from library/nginx
c29f5b76f736: Pull complete
e19db8451adb: Pull complete
24ff42a0d907: Pull complete
c558df217949: Pull complete
976e8f6b25dd: Pull complete
6c78b0ba1a32: Pull complete
84cade77a831: Pull complete
Digest: sha256:91734281c0ebfc6f1aea979cffeed5079cfe786228a71cc6f1f46a228cde6e34
Status: Downloaded newer image for nginx:latest
docker.io/library/nginx:latest
root@916dcf806f740625:~/code# mkdir -p ~/nginx/html
root@916dcf806f740625:~/code#
root@916dcf806f740625:~/code# echo '<h1>Hello, Docker!</h1>' > ~/nginx/html/index.html
root@916dcf806f740625:~/code# docker run --name my-nginx -v
~/nginx/html:/usr/share/nginx/html:ro -p 8080:80 -d nginx
docker: Error response from daemon: Conflict. The container name "/my-nginx" is already in use by container "e3f7e152ae41aee6e3c032ad05cb949407269f0eefb0cc85dad71ee7c0aa14fc". You have to

remove (or rename) that container to be able to reuse that name.
See 'docker run --help'.
root@916dcf806f740625:~/code#

explain these in a very easiest manner in details with basics also correct if any error occur :

Run the NGINX Container
Run the NGINX container, mapping the local directory to the container's web root:

docker run --name my-nginx -v ~/nginx/html:/usr/share/nginx/html:ro -p 8080:80 -d nginx
Explanation of the command options:

--name my-nginx: Names the container my-nginx.
-v ~/nginx/html:/usr/share/nginx/html:ro: Maps the local ~/nginx/html directory to the container's /usr/share/nginx/html directory in read-only mode.
-p 8080:80: Maps port 8080 on the host to port 80 in the container.
-d nginx: Runs the NGINX container in detached mode.

Let's break this down step-by-step in the **simplest way possible** and also fix the error you're facing.

✅ First, What You Are Trying to Do:

You want to:

- Run an NGINX web server **inside Docker**
- Show a simple HTML file (`Hello, Docker!`)
- Access it using your **computer's browser at port `8080`**

✅ The Correct Command:

```
bash

docker run --name my-nginx -v ~/nginx/html:/usr/share/nginx/html:ro -p 8080:80 -d nginx
```

✅ What Each Part Means (Super Simple):

Part	Meaning
`docker run`	Start a new container
`--name my-nginx`	Give the container a name: **my-nginx**
`-v ~/nginx/html:/usr/share/nginx/html:ro`	Share your folder (`~/nginx/html`) with NGINX's web folder inside the container, read-only

Part	Meaning
`-p 8080:80`	Connect your computer's port **8080** to NGINX's default port **80**
`-d`	Run the container **in the background** (detached mode)
`nginx`	Use the **nginx** image you downloaded earlier

⚠️ The Error You Got:

```
csharp

Conflict. The container name "/my-nginx" is already in use...
```

This means:
You **already have a container** named `my-nginx` running or stopped. Docker doesn't allow two containers with the same name.

✅ How to Fix It

🅰️ Option 1: Remove the old container (safe if you don't need it anymore)

```
bash

docker rm -f my-nginx
```

Then run your command again:

```
bash

docker run --name my-nginx -v ~/nginx/html:/usr/share/nginx/html:ro -p 8080:80 -d nginx
```

🅱️ Option 2: Use a different name

```
bash

docker run --name my-nginx2 -v ~/nginx/html:/usr/share/nginx/html:ro -p 8080:80 -d nginx
```

✅ How to Check If It's Working

Open your browser and go to:

```
arduino

http://localhost:8080
```

You should see:

```
Hello, Docker!
```

Would you like to learn how to stop or remove the container later too?

what does this means:
-d Run the container in the background (detached mode)

Great question! Let's break it down in a super easy way:

🔹 `-d` means Detached Mode

When you use `-d` in the `docker run` command:

```
bash

docker run -d nginx
```

You are telling Docker:

> **"Run this container in the background — I don't want to see its logs or output in my terminal."**

✅ Simple Example:

- **Without `-d` (Attached mode):**

```
bash

docker run nginx
```

- Your terminal **gets "stuck"** showing logs from NGINX.
- You **can't type anything else** until you stop it.
- You'll see things like:

```
bash

127.0.0.1 - - [date] "GET / HTTP/1.1" 200 ...
```

- **With `-d` (Detached mode):**

```
bash

docker run -d nginx
```

- The container runs **in the background**
- Your terminal is **free to use**
- You get back a container ID like:

```
nginx

e3f7e152ae41aee6e3c032ad05cb949407269f0eefb0cc85dad71ee7c0aa14fc
```

🔁 Bonus: How to View Output Later

If you want to **see what's happening inside**, you can use:

```
bash

docker logs <container-name>
```

Example:

```
bash

docker logs my-nginx
```

So, to sum it up:

`-d`	Detached Mode	Background Running	Terminal Free

Let me know if you want to try both modes with an example!

solve this problem:
root@d5f5202f9e4f7d86:~/code# docker pull nginx
Using default tag: latest
latest: Pulling from library/nginx
6e909acdb790: Pull complete
5eaa34f5b9c2: Pull complete
417c4bccf534: Pull complete
e7e0ca015e55: Pull complete
373fe654e984: Pull complete
97f5c0f51d43: Pull complete
c22eb46e871a: Pull complete
Digest: sha256:124b44bfc9ccd1f3cedf4b592d4d1e8bddb78b51ec2ed5056c52d3692baebc19
Status: Downloaded newer image for nginx:latest
docker.io/library/nginx:latest
root@d5f5202f9e4f7d86:~/code# mkdir -p ~/nginx/html
root@d5f5202f9e4f7d86:~/code# echo '<h1>Hello, Docker!</h1>' > ~/nginx/html/index.html
root@d5f5202f9e4f7d86:~/code# docker run --name my-nginx -v
~/nginx/html:/usr/share/nginx/html:ro -p 8080:80 -d nginx
16250fdc80e73e2f85a89fa95aeecdc4f03b88136d83999aa7b18ccdc0c45e61
docker: Error response from daemon: driver failed programming external connectivity on endpoint

```
my-nginx (303106e2808dd8a31e821438432420979cac0960e08f76cdb709ae6a8cf554e9): Error
starting userland proxy: listen tcp4 0.0.0.0:8080: bind: address already in use.
root@d5f5202f9e4f7d86:~/code# docker run --name my-nginx -v
~/nginx/html:/usr/share/nginx/html:ro -p 8080:80 -d nginx
docker: Error response from daemon: Conflict. The container name "/my-nginx" is already in use by
container "16250fdc80e73e2f85a89fa95aeecdc4f03b88136d83999aa7b18ccdc0c45e61". You have to
remove (or rename) that container to be able to reuse that name.
See 'docker run --help'.
root@d5f5202f9e4f7d86:~/code# root@d5f5202f9e4f7d86:~/nginx/html# docker run --name my-
nginx -v ~/nginx/html:/usr/share/nginx/html:ro -p 8080:80 -d nginx
docker: Error response from daemon: Conflict. The container name "/my-nginx" is already in use by
container "16250fdc80e73e2f85a89fa95aeecdc4f03b88136d83999aa7b18ccdc0c45e61". You have to
remove (or rename) that container to be able to reuse that name.
See 'docker run --help'.
root@d5f5202f9e4f7d86:~/nginx/html#
```

You're facing **two main issues** here:

❌ Problem 1: Port 8080 is already in use

Error:

```
perl

Error starting userland proxy: listen tcp4 0.0.0.0:8080: bind: address already in use.
```

🔍 Why?

Something else (maybe another container or program) is already using **port 8080** on your machine.

❌ Problem 2: Container with the same name already exists

Error:

```
csharp

Conflict. The container name "/my-nginx" is already in use...
```

🔍 Why?

You already created a container named `my-nginx`, and Docker doesn't allow duplicate names.

✅ How to Fix It (Step-by-Step)

✅ Step 1: Check what's using port 8080

Run this:

```bash
sudo lsof -i :8080
```

This will show you what is using port 8080. If it's an old Docker container, go to step 2.

✅ Step 2: Stop and remove the old container

Run:

```bash
docker ps -a
```

You'll see your container listed, like:

```javascript
CONTAINER ID   IMAGE    NAME       STATUS
16250fdc80e7   nginx    my-nginx   Exited/Error
```

Now remove it:

```bash
docker rm -f my-nginx
```

✅ Step 3: Run it again with a different port (optional)

If port 8080 is busy and you can't free it, try a different port like 8081:

```bash
docker run --name my-nginx -v ~/nginx/html:/usr/share/nginx/html:ro -p 8081:80 -d nginx
```

Then go to:

```arduino
http://localhost:8081
```

✅ **Summary**:

- `port 8080` was **already in use**
- `my-nginx` container name was **already taken**
- You fixed it by stopping/removing the container and/or using a different port

Let me know if you want a command to auto-check and clean these issues for future runs!

```
root@d5f5202f9e4f7d86:~/code# docker run --name my-nginx -v
~/nginx/html:/usr/share/nginx/html:ro -p 8080:80 -d nginx
edbecff71473493de899e593043756b69801e289d587370130d40695f5078a37
docker: Error response from daemon: driver failed programming external connectivity on endpoint
my-nginx (9eeca75184e1c9b88e40f12abcb3d5a3c737f23add21e73f6c12ea620a10986a): Error
starting userland proxy: listen tcp4 0.0.0.0:8080: bind: address already in use.
root@d5f5202f9e4f7d86:~/code# docker run --name my-nginx -v
~/nginx/html:/usr/share/nginx/html:ro -p 9080:80 -d nginx
docker: Error response from daemon: Conflict. The container name "/my-nginx" is already in use by
container "edbecff71473493de899e593043756b69801e289d587370130d40695f5078a37". You have
to remove (or rename) that container to be able to reuse that name.
See 'docker run --help'.
root@d5f5202f9e4f7d86:~/code# docker run --name my-nginx2 -v
~/nginx/html:/usr/share/nginx/htm
l:ro -p 9080:80 -d nginx
f7352f3648559c81d0350ba37fadfcc50331febb6406326224ba2c7f58655e63
root@d5f5202f9e4f7d86:~/code# docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED         STATUS         PORTS
NAMES
f7352f364855   nginx     "/docker-entrypoint...."   44 seconds ago   Up 43 seconds   0.0.0.0:9080-
>80/tcp, :::9080->80/tcp   my-nginx2
root@d5f5202f9e4f7d86:~/code# curl http://localhost:8080
Found. Redirecting to ./?folder=/root/nginx/htmlroot@d5f5202f9e4f7d86:~/code# curl
http://localhost:9080
<h1>Hello, Docker!</h1>
root@d5f5202f9e4f7d86:~/code# docker logs my-nginx2
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Sourcing /docker-entrypoint.d/15-local-resolvers.envsh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
2025/03/26 10:20:52 [notice] 1#1: using the "epoll" event method
2025/03/26 10:20:52 [notice] 1#1: nginx/1.27.4
2025/03/26 10:20:52 [notice] 1#1: built by gcc 12.2.0 (Debian 12.2.0-14)
2025/03/26 10:20:52 [notice] 1#1: OS: Linux 5.10.51
2025/03/26 10:20:52 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2025/03/26 10:20:52 [notice] 1#1: start worker processes
2025/03/26 10:20:52 [notice] 1#1: start worker process 29
2025/03/26 10:20:52 [notice] 1#1: start worker process 30
2025/03/26 10:20:52 [notice] 1#1: start worker process 31
2025/03/26 10:20:52 [notice] 1#1: start worker process 32
172.17.0.1 - - [26/Mar/2025:10:21:06 +0000] "GET / HTTP/1.1" 200 24
"https://67aa3ccddb2c69e7e975ceff.bm-north.lab.poridhi.io/?folder=/root/code" "Mozilla/5.0
(Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/134.0.0.0
```

```
Safari/537.36" "103.170.231.217, 172.20.0.4"
172.17.0.1 - - [26/Mar/2025:10:22:30 +0000] "GET / HTTP/1.1" 200 24 "-" "curl/7.68.0" "-"
root@d5f5202f9e4f7d86:~/code# sudo lsof -i :8080
sudo: lsof: command not found
root@d5f5202f9e4f7d86:~/code#
```

Awesome — you're almost there and you've already solved most of the issues 🎉
Let's do a quick recap and final explanation to clear everything up:

✅ What You Did Correctly:

1. ✅ **Pulled the NGINX Docker image**:

   ```bash
   docker pull nginx
   ```

2. ✅ **Created a custom HTML file** to serve:

   ```bash
   echo '<h1>Hello, Docker!</h1>' > ~/nginx/html/index.html
   ```

3. ✅ **Faced port 8080 conflict**, so you used port `9080` instead 👋

   ```bash
   docker run --name my-nginx2 -v ~/nginx/html:/usr/share/nginx/html:ro -p 9080:80 -d nginx
   ```

4. ✅ **Confirmed it's working** by using `curl`:

   ```bash
   curl http://localhost:9080
   ```

   Output:

   ```css
   <h1>Hello, Docker!</h1>
   ```

❌ Remaining Minor Issue: Port 8080 still in use

You tried this:

```bash
sudo lsof -i :8080
```

But got:

```
bash

sudo: lsof: command not found
```

✅ Fix: Install `lsof` to Check What's Using Port 8080

To install `lsof` on Ubuntu/Debian systems:

```
bash

apt update
apt install lsof -y
```

Then check:

```
bash

lsof -i :8080
```

This will show you what program or container is using port 8080. You can then decide to:

- Stop the process using that port
- Or just keep using another port like `9080` (which you're doing fine now!)

🔄 Optional Cleanup Commands

- 🔍 See all containers (running + stopped):

  ```
  bash

  docker ps -a
  ```

- 🗑️ Remove a stopped container (like your old `my-nginx`):

  ```
  bash

  docker rm my-nginx
  ```

- 💥 Force-remove any container (even if running):

  ```
  bash

  docker rm -f my-nginx
  ```

- ❌ Remove a container **by ID** (if name not working):

  ```
  bash

  docker rm -f edbecff71473
  ```

✅ Final Confirmation

Your working container:

```
bash

docker ps
```

Output:

```
nginx

f7352f364855   nginx   ...   0.0.0.0:9080->80/tcp   my-nginx2
```

Means it's all good.
Open a browser and go to:

```
arduino

http://localhost:9080
```

And you'll see your page:
Hello, Docker!

Let me know if you want help:

- Creating a custom NGINX config
- Hosting multiple pages/sites
- Making it auto-start after reboot
- Or running this on a server/VPS!

```
root@d5f5202f9e4f7d86:~/code# lsof -i :8080
COMMAND  PID USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
node    1173 root   21u  IPv4   3012      0t0  TCP *:http-alt (LISTEN)
node    1173 root   23u  IPv4   7420      0t0  TCP 10.62.20.239:http-alt->10.62.0.1:42218 (ESTABLISHED)
node    1173 root   28u  IPv4  12585      0t0  TCP 10.62.20.239:http-alt->10.62.0.1:41322
(ESTABLISHED)
node    1957 root   21u  IPv4   5427      0t0  TCP 10.62.20.239:http-alt->10.62.0.1:42280 (ESTABLISHED)
node    2820 root   21u  IPv4  12150      0t0  TCP 10.62.20.239:http-alt->10.62.0.1:41396
(ESTABLISHED)
root@d5f5202f9e4f7d86:~/code#
```

Great! Your output shows that **port 8080** (which is also called `**http-alt**`) is being used by a **Node.js application** (`**node**` process with PID `**1173**`, `**1957**`, and `**2820**`).

🔍 What This Means:

- You **cannot run NGINX on port 8080** right now because it's already being used by that Node.js app.
- That's why Docker was throwing this error:

```
perl

Error starting userland proxy: listen tcp4 0.0.0.0:8080: bind: address already in use
```

✅ Options to Fix:

🅰️ Use a Different Port (Recommended, Already Working)

You're already using port `9080` successfully. You can continue using that:

```
bash

http://localhost:9080
```

🅱️ Stop the Node.js Process (Only if you don't need it)

If you want to **free port 8080**, you can stop the process using it:

1. Identify the main process using 8080:

```
yaml

PID 1173 → node
```

2. Kill it:

```
bash

kill 1173
```

3. Then re-run your Docker NGINX container on port 8080:

```
bash

docker run --name my-nginx -v ~/nginx/html:/usr/share/nginx/html:ro -p 8080:80 -d nginx
```

⚠️ **Be careful:** If that Node.js app is important (e.g. another service or web app), killing it will stop that app.

✅ Summary:

Action	Result
Use port 9080	✅ Safe, no conflicts
Stop Node.js	🛑 Frees port 8080 but may affect other apps

Action	Result
Keep both running	Just use different ports (e.g., 8080 for Node.js, 9080 for NGINX)

Let me know if you want help running **both apps behind one domain** using NGINX as a **reverse proxy** — that's a cool setup too!

how to free the port 8080

To **free port 8080**, you need to **stop the process** that is currently using it. In your case, it's a **Node.js process** with **PID 1173** (based on the output from `lsof`).

✅ Step-by-Step to Free Port 8080

🔍 Step 1: Confirm what's using the port

You've already done this:

```bash
lsof -i :8080
```

And saw this:

```ruby
COMMAND PID  USER  ...  NAME
node    1173 root  ...  TCP *:http-alt (LISTEN)
```

So, **Node.js with PID 1173** is using it.

🛑 Step 2: Stop the Process

👉 Option 1: Gracefully stop it

```bash
kill 1173
```

👉 Option 2: Force kill (if it doesn't stop)

```bash
kill -9 1173
```

✅ Step 3: Verify it's freed

Run:

```
bash

lsof -i :8080
```

If nothing shows up, that means **port 8080 is now free** and you can run your NGINX container on it.

🚀 Now Run Your NGINX Container Again

```
bash

docker run --name my-nginx -v ~/nginx/html:/usr/share/nginx/html:ro -p 8080:80 -d nginx
```

Then visit:

```
arduino

http://localhost:8080
```

You should see your NGINX server serving `**Hello, Docker!**`

Let me know if you want to permanently block a program from restarting and using port 8080 again (e.g., via systemd or PM2)!