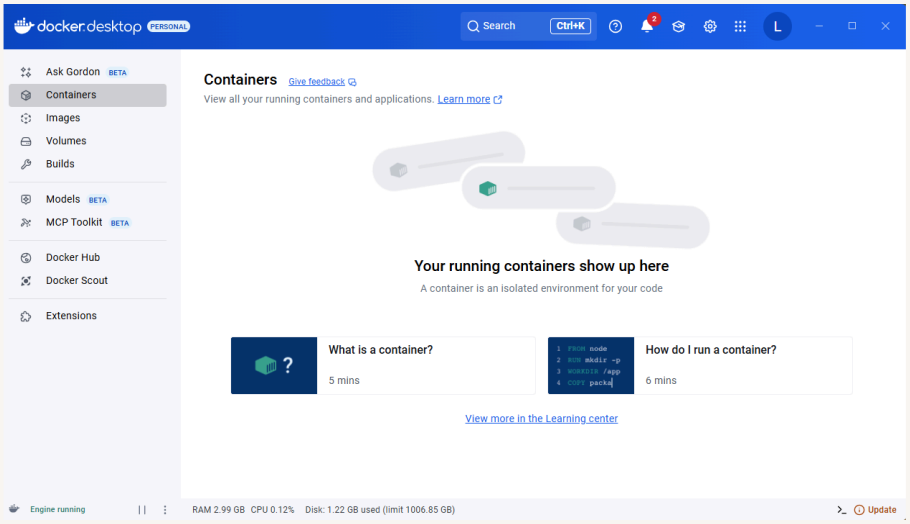
nextwork.org

Deploy an App with Docker

Louis Moyo



Louis Moyo
NextWork Student

Introducing Today's Project!

What is Docker?

Docker is a way to package an app and all of its dependencies into a lightweight container that runs the same everywhere. It’s useful because it gives consistency, isolation, and fast, repeatable deployments. In this project I used Docker to build an image from nginx:latest, copy in my index.html, and expose port 80. I could run it locally and then deploy the same image on AWS Elastic Beanstalk, which meant what worked on my machine worked in the cloud too.

One thing I didn't expect...

I didn’t expect Elastic Beanstalk to be so strict about the ZIP layout and version labels. Reusing a label or zipping a folder (instead of the two files at the root) caused aborted deployments. I also hit port-80 conflicts locally and learned to stop/remove containers. The takeaway: keep version labels unique, zip only Dockerfile + index.html at the root, and use the EB CLI/CloudShell to deploy cleanly.

This project took me...

This project took me 3–4 hours end-to-end. The most time was spent diagnosing EB deployment errors and learning the right CLI flow. It was most rewarding to see my page live at a public URL and realise the same container ran identically on my laptop and in AWS.

L
Louis Moyo
NextWork Student

Understanding Containers and Docker

Containers

Containers are lightweight, portable units that package an app with all its dependencies so it runs the same anywhere. They're faster than virtual machines, start in seconds, and use fewer resources. Containers are useful because they ensure consistency across environments, isolate apps for stability, and scale easily to handle demand. This makes developing, testing, and deploying applications more reliable and efficient.

A container image is a blueprint for creating containers. It packages the app code, dependencies, libraries, and configuration into one file so containers built from it run consistently on any system with Docker.

Docker

Docker is a platform that lets you build, package, and run applications inside containers, ensuring they work consistently across different environments. Docker Desktop is a graphical application that makes using Docker easier on your local computer. It includes Docker Engine, tools, and a dashboard so you can manage containers, images, volumes, and networks without only relying on terminal commands.

L

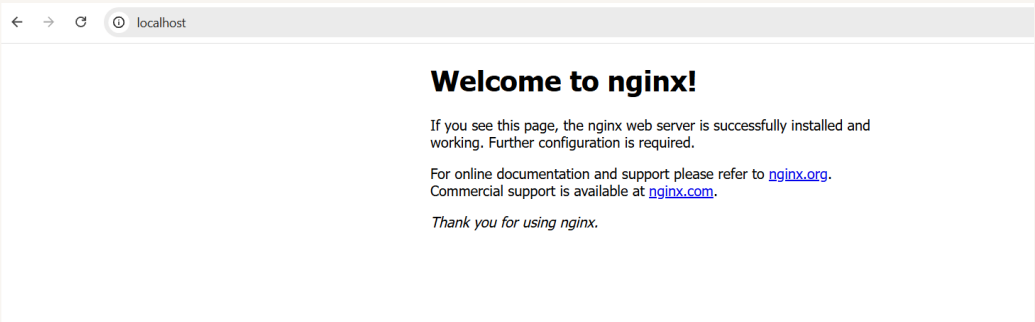
Louis Moyo
NextWork Student

The Docker daemon is the background process that runs on your computer and manages Docker containers. It listens for commands from the Docker client (like docker run) and does the heavy lifting of building, running, and distributing containers. The daemon handles container lifecycle, networking, storage, and ensures your containers run as defined in their images. Without the daemon, Docker commands wouldn't actually create or run containers.

Running an Nginx Image

Nginx is a lightweight, high-performance web server used to serve web pages and handle requests. It is popular because it can manage large amounts of traffic efficiently, act as a reverse proxy, load balancer, and is often used in containers for simplicity and speed.

The command I ran to start a new container was: docker run -d -p 80:80 nginx This tells Docker to run the Nginx image in detached mode (-d) and map port 80 on my computer to port 80 inside the container so I can access the web page in my browser.

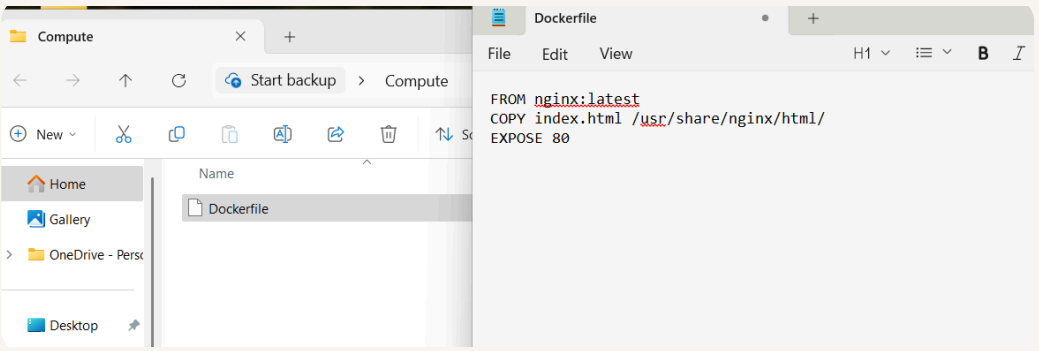


Creating a Custom Image

A Dockerfile is a plain-text recipe that tells Docker exactly how to build an image - step by step. Each instruction (e.g., FROM, COPY) creates a cached layer so builds are reproducible and fast. You run it with docker build to produce a portable image that runs the same everywhere.

My Dockerfile tells Docker three things: 1. Use a base image — FROM nginx:latest pulls the official NGINX image to start from. 2. Add my web content — COPY index.html /usr/share/nginx/html/ replaces NGINX's default page with my own index.html so it's what gets served. 3. Declare the network port — EXPOSE 80 documents that the container listens on HTTP port 80 (I'll actually publish it when running, e.g., docker run -p 80:80 ...).

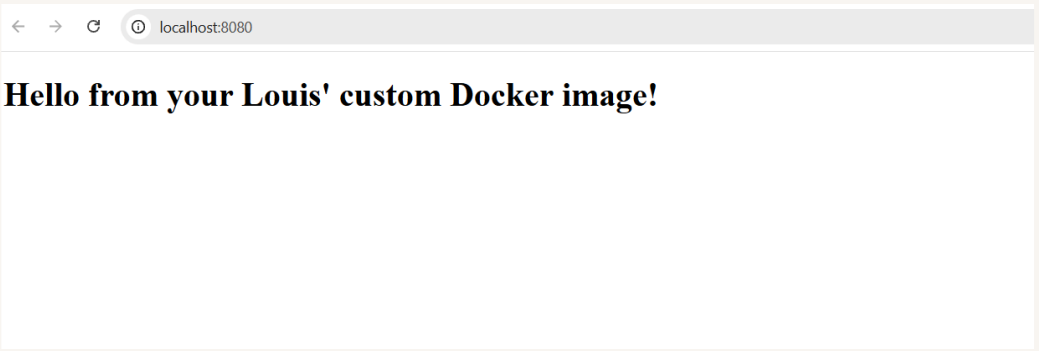
The command I used to build a custom image with my Dockerfile was docker build -t my-web-app . The . at the end means "use the current directory as the build context (where Docker reads the Dockerfile and any files it needs, like index.html).



Running My Custom Image

There was an error when I ran my custom image because port 80 on my PC was already in use. Docker returned: “Bind for 0.0.0.0:80 failed: port is already allocated.” I resolved this by finding/stopping the container on port 80 (docker ps --filter "publish=80", docker stop <ID>), then running my image on a free port: docker run -d -p 8080:80 my-web-app After that I could open http://localhost:8080 and see my page.

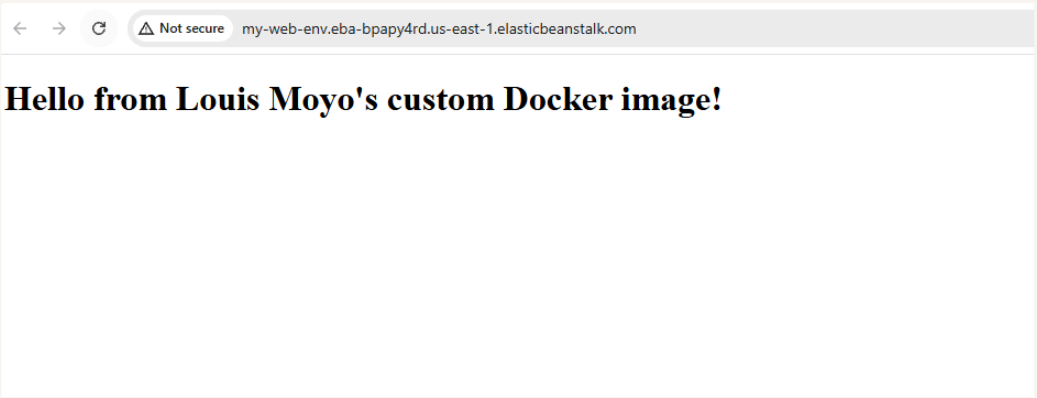
In this example, the container image is my-web-app:latest - the read-only blueprint built from my Dockerfile (FROM nginx, COPY index.html, EXPOSE 80). It packages the web server and my index.html. The container is the running instance created from that image, e.g. my-web-container, with runtime settings like port mapping 8080:80. It has a writable layer, can be started/stopped/removed, and I can run many containers from the same image.


localhost:8080
Hello from your Louis' custom Docker image!

Elastic Beanstalk

Elastic Beanstalk is AWS's platform-as-a-service for deploying web apps without managing servers. You upload your code or Docker image and EB automatically provisions EC2, networking, load balancing, auto-scaling, and health monitoring, then gives you a public URL. It handles rolling updates and basic logs/metrics so you can focus on your application, not the infrastructure.

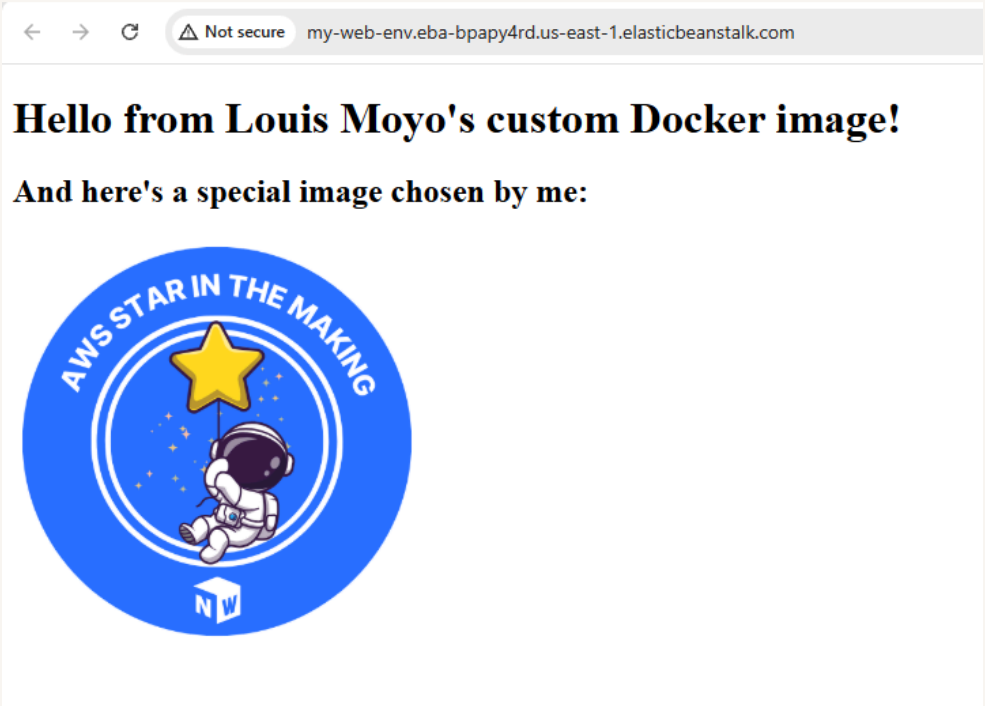
Deploying my custom image with Elastic Beanstalk took 10 minutes end-to-end. It tppl 2–3 min to set up IAM roles and EB app ,6–7 min for the environment to provision and become healthy, and 1 min for the eb deploy upload. Subsequent updates took 30–60 seconds each.


Not secure
my-web-env.eba-bpapy4rd.us-east-1.elasticbeanstalk.com
Hello from Louis Moyo's custom Docker image!

Deploying App Updates

I updated index.html by keeping the H1 and adding an H2 plus an <img> pointing to a public URL with simple responsive styling. I verified the change locally by opening the HTML file in Chrome and confirming the new heading and image displayed correctly.

My app updates didn't show up because Elastic Beanstalk was still serving the previous bundle-my edits existed only on my computer. To deploy my changes, I only had to create a fresh zip with just Dockerfile and index.html at the zip root, go to my EB environment, click Upload and deploy, choose the new zip, and give it a new version label (e.g., Version Two). After the deploy completed, I opened the environment URL and hard-refreshed to see the new H2 and image. (CLI: eb deploy.)


Not secure
my-web-env.eba-bpapy4rd.us-east-1.elasticbeanstalk.com
Hello from Louis Moyo's custom Docker image!
And here's a special image chosen by me:
AWS STAR IN THE MAKING

The place to learn & showcase your skills

Check out nextwork.org for more projects