Dockerfile

A Dockerfile is analogous to a recipe, specifying the required ingredients for an application. Let's take a simple example of a nodejs application whose Dockerfile looks as follows:

FROM node:boron

Create app directory

WORKDIR /home/code

Install app dependencies

RUN npm install

EXPOSE 8080

CMD ["npm", "start"]

Notice EXPOSE 8080. You may have seen this statement in most of the Dockerfile(s) across Docker hub.

The EXPOSE instruction informs Docker that the container listens on the specified network ports at runtime. EXPOSE does not make the ports of the container accessible to the host.

A simpler explanation

The **EXPOSE** instruction exposes the specified port and makes it available only for **inter-container communication**. Let's understand this with the help of an example.

Let's say we have two containers, a nodejs application and a redis server. Our node app needs to communicate with the redis server for several reasons.

For the node app to be able to talk to the redis server, the redis container needs to expose the port. Have a look at the **Dockerfile** of official redis image and you will see a line saying **EXPOSE 6379**. This is what helps the two containers to communicate with each other.

So when your nodejs app container tries to connect to the 6379 port of the redis container, the EXPOSE instruction is what makes this possible.

Note: For the node app server to be able to communicate with the redis container, it's important that both the containers are running in the same docker network.

Binding the container port with the host

So EXPOSE helps in inter-container communication. What if, there's a need to bind the port of the container with that of the host machine on which the container is running?

Pass the -p (lower case p) as a option to the docker run instruction as follows

docker run -p <HOST_PORT>:<CONTAINER:PORT> IMAGE_NAME

Find out more about this in the official documentation.