What is a Docker image?

A Docker image is a snapshot of source code, libraries, dependencies, tools, and everything else (except the Operating System kernel!) that a container needs to run.

There are many pre-built images that you can use. For example, some come with Ubuntu (a Linux distribution). Others come with Ubuntu and Python already installed. You can also make your own images that already have Flask installed (as well as other dependencies your app needs).

> **COMES WITH UBUNTU?**
>
> In the last lecture I mentioned that Docker containers use the host OS kernel, so why does the container need Ubuntu?
>
> Remember that operating systems are kernel + programs/libraries. Although the container uses the host kernel, we may still need a lot of programs/libraries that Ubuntu ships with. An example might be a C language compiler!

This is how you define a Docker image. I'll guide you through how to do this in the next lecture, but bear with me for a second:

```
FROM python:3.10
EXPOSE 5000
WORKDIR /app
RUN pip install flask
COPY . .
CMD ["flask", "run", "--host", "0.0.0.0"]
```

This is a `Dockerfile`, a definition of how to create a Docker image. Once you have this file, you can ask Docker to create the Docker image. Then, after creating the Docker image, you can ask Docker to run it as a container.

```
Dockerfile ---build--> docker image ---run--> docker container
```

In this `Dockerfile` you can see the first line: `FROM python:3.10`. This tells Docker to first download the `python:3.10` image (an image someone else already created), and once that image is created, run the following commands.

WHAT'S IN THE PYTHON IMAGE?

The `python:3.10` image is also built using a `Dockerfile`! You can see the `Dockerfile` for it here.

You can see it comes `FROM` another image. There is usually a chain of these, images built upon other images, until you reach the base image. In this case, the base image is running Debian (a Linux distribution).

▸ Where is the base image!?

So, why the chain?

Three main reasons:

1. So you don't have to write a super long and complex `Dockerfile` which contains everything you need.
2. So pre-published images can be shared online, and all you have to do is download them.
3. So when your own images use the same base image, Docker in your computer only downloads the base image once, saving you a lot of disk space.

Back to our `Dockerfile`. The commands after `FROM...` are specific to our use case, and do things like install requirements, copy our source code into the image, and tell Docker what command to run when we start a container from this image.

This separation between images and containers is interesting because once the image is created you can ship it across the internet and:

- Share it with other developers.
- Deploy it to servers.

Plus once you've downloaded the image (which can take a while), starting a container from it is almost instant since there's very little work to do.