Get Started, Part 2: Containers

Prerequisites

- Install Docker version 1.13 or higher.
- Read the orientation in Part 1.
- Give your environment a quick test run to make sure you're all set up:

```
docker run hello-world
```

Introduction

It's time to begin building an app the Docker way. We start at the bottom of the hierarchy of such an app, which is a container, which we cover on this page. Above this level is a service, which defines how containers behave in production, covered in Part 3. Finally, at the top level is the stack, defining the interactions of all the services, covered in Part 5.

- Stack
- Services
- **Container** (you are here)

Your new development environment

In the past, if you were to start writing a Python app, your first order of business was to install a Python runtime onto your machine. But, that creates a situation where the environment on your machine needs to be perfect for your app to run as expected, and also needs to match your production environment.

With Docker, you can just grab a portable Python runtime as an image, no installation necessary. Then, your build can include the base Python image right alongside your app code, ensuring that your app, its dependencies, and the runtime, all travel together.

These portable images are defined by something called a Dockerfile.

Define a container with Dockerfile

Dockerfile defines what goes on in the environment inside your container. Access to resources like networking interfaces and disk drives is virtualized inside this environment, which is isolated from the rest of your system, so you need to map ports to the outside world, and be specific about what files you want to “copy in” to that environment. However, after doing that, you can expect that the build of your app defined in this Dockerfilebehaves exactly the same wherever it runs.