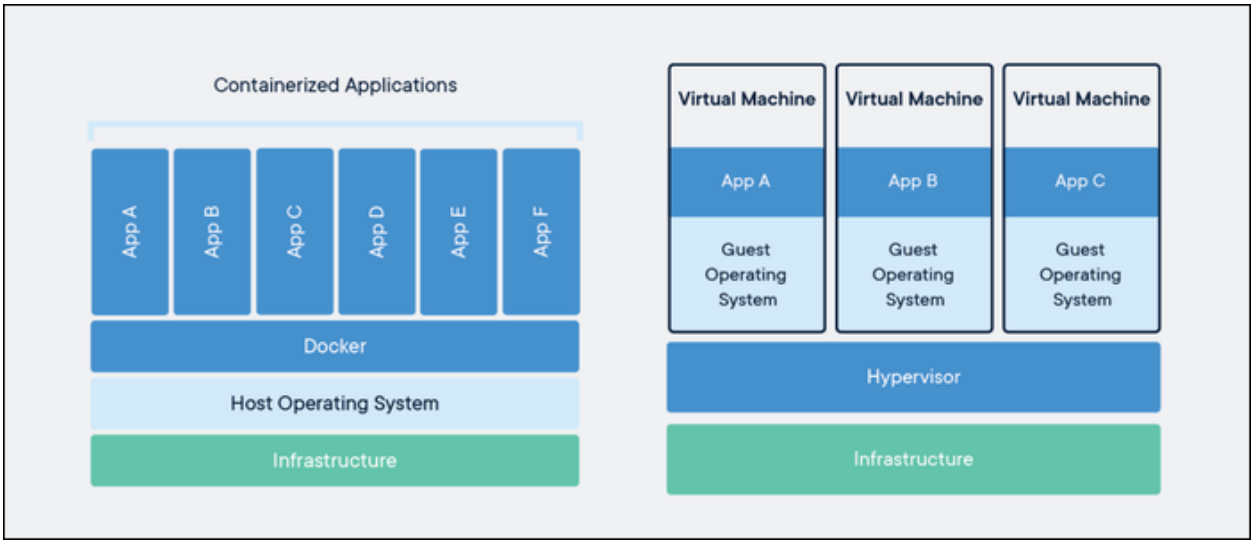
Getting Started with Docker

A Guide to What Docker is, How to Install it, and The Basics of Using it

Introduction

So what is Docker? You can think of Docker as a middleman between full on virtualization and running something normally on your hardware. While virtualization would emulate an entire operating system on your computer, Docker uses something called containers. Containers, unlike VMs, use the host operating system and virtualizes the software running.



Containers only have exactly what they need inside of them, with nothing else. This means they are generally super lightweight so you can run a whole bunch of them on one machine. Instead of needing an entire desktop/virtual machine to run a simple application, like a web server for example, you can just throw it in a container and call it a day.

In a VM you call what manages the VM's the hypervisor (here's more information on hypervisors if you want to learn more), in Docker that's the Docker Engine.

Installing Docker

Docker is available for Linux, MacOS, and Windows, making it something very useful for developers who run into the “but it works on my machine” issue. It can also allow you to create a container, and anyone can run it on nearly any device without caring about the operating system. We will be using a package manager for all the installations. If you want to learn more about them/need to install one on your system, refer to the document Package Managers.

Windows Installation

There are many ways to install Docker on Windows, but we will be using Chocolatey, a package manager for Windows.

In an administrator PowerShell console, run the command. There may be a confirmation request in the console, confirm it or the installation will halt.

choco install docker-desktop

MacOS Installation

We are going to be using Homebrew for this, so install it if you haven't already. All you need to do is run the following command in your terminal if you have Homebrew installed.

brew install --cask docker

Running the Docker Desktop

For Windows and MacOS you need to run the Docker Desktop application that was installed on your local computer during the above steps. You need to do this to start the Docker Engine. Go through the recommended setup options (there is no need to create an account right now, skip where you can).

Verify your Install

To verify your install, run the command “docker run hello-world” and if you get an error message, you have not properly installed Docker.

Note: You will most likely need to restart your CLI to use the command after installation.

Docker Hub account

Now that you have verified your installation, you should create a Docker Hub account. We will talk more about what this is later, but now think of it as GitHub for Docker. To create an account run Docker login and follow the steps there.

Note: It is best to use a PAT (Personal Access Token)

Using Docker

I think the best way to get to know a system is to play around with it, so let's do that! I will guide you through making and using your own Docker container. The result of this will be running a simple Hello World python script!

First, create a directory to store some files and navigate to it. Then create a one-line Python script named hello.py in that directory that prints “Hello World!” or any other string you like.

Dockerfile

Before we can create a container, we need an image. We will talk more about what an image is later, but now we will just focus on how to make one. To make an image, create a file named Dockerfile in your directory. In the contents of this file, paste the following.

```
# The operating system this is running on

FROM ubuntu:latest

# Install Python

RUN set -xe \

  && apt-get update \

  && apt-get install -y python3

# Copy your hello-world.py script into the filesystem of the container

COPY hello-world.py ./hello-world.py

# Define the entrypoint of the container

ENTRYPOINT ["python3", "hello.py"]
```

I recommend reading up more on how to write Dockerfiles. This is just a very basic example of one. Here is Docker’s official reference on how to create them.

Now, we need to use this Dockerfile to create an image. Let’s create a new image called example-image by running this command: “docker build -t example-image .”

Images

A Docker image describes the parameters of the container and how to create it. We create these images from Dockerfiles, just like we just did. Now we don't always have to create our own images, rather there are a lot of them already available out there for us to use whenever we want! You can find a very large collection of them at the Docker Hub.

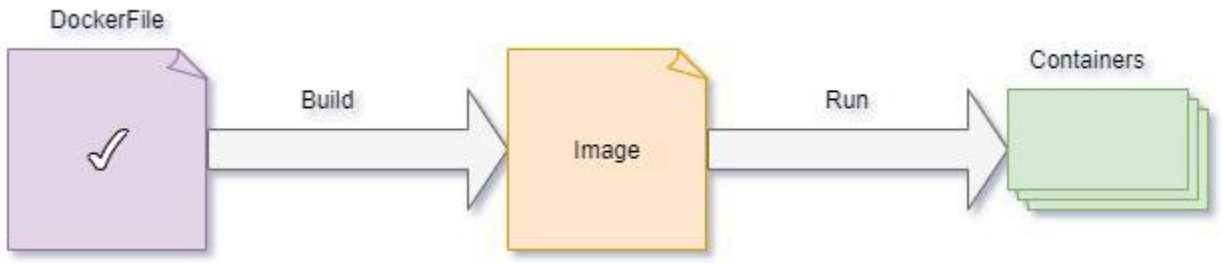
To see our newly created Docker image, run “docker image ls”. You should see the newly created image!

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
example-image	latest	abcd0cf5b7a3	21 minutes ago	44.3MB
hello-world	latest	9c7a54a9a43c	6 months ago	13.3kB

If you would like to remove an image, just run “docker image rm <IMAGE ID>”. Be careful though, no containers can be currently using that image or else you will be unable to delete it.

Containers

Finally, we are going to make a container! This is where Docker gets fun. To recap, we wrote a Dockerfile, built an image, and will run a container from that image. That is the Docker lifecycle, the so called ‘meat and potatoes’ of the whole situation. Understanding that workflow is essential to understanding docker.



To start a container from our new image, we need to run docker run example-image. This will start up our container, and will output “Hello World!” or whatever string you entered for your hello.py.