Learn Enough Docker to be Useful

Part 2: A Delicious Dozen Docker Terms You Need to Know

Jeff Hale
Jan 16, 2019 · 6 min read ★

In Part 1 of this series we explored the conceptual landscape of Docker containers. We discussed the reasons Docker containers are important and several ways to think about them. And we made one into a pizza 🍕. In this article I'll share a dozen additional terms from the Docker ecosystem that you need to know.

Keeping with the food theme from the first article in the series. Here's a dozen delicious Docker donuts. Think of each donut as a Docker container. 😄

Docker Ecosystem Terms

I've broken Docker terms into two categories for easier mental model creation: *Essentials* and *Scaling*. Let's hit the eight essentials first.

Docker Essentials

Docker Platform is Docker's software that provides the ability to package and run an application in a container on any Linux server. Docker Platform bundles code files and dependencies. It promotes easy scaling by enabling portability and reproducibility.

Docker Engine is the client-server application. The Docker company divides the Docker Engine into two products. *Docker Community Edition (CE)* is free and largely based on open source tools. It's probably what you'll be using. *Docker Enterprise* comes with additional support, management, and security features. Enterprise is how the Docker firm keeps the lights on.

Engine makes things run

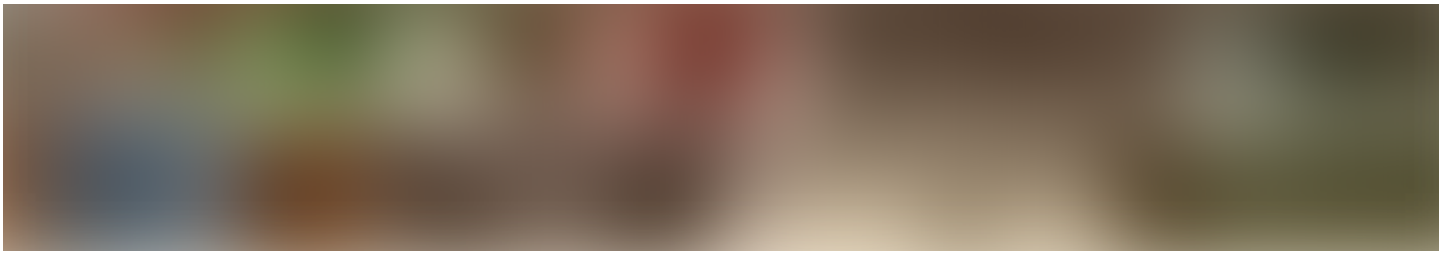
Docker Client is the primary way you'll interact with Docker. When you use the Docker Command Line Interface (CLI) you type a command into your terminal that

starts with `docker`. Docker Client then uses the Docker API to send the command to the Docker Daemon.

Diagram from the Docker docs

Docker Daemon is the Docker server that listens for Docker API requests. The Docker Daemon manages images, containers, networks, and volumes.

Docker Volumes are the best way to store the persistent data that your apps consume and create. We'll have more to say about Docker Volumes in Part 5 of this series. Follow me to make sure you don't miss it.



Volumes

A **Docker Registry** is the remote location where Docker Images are stored. You push images to a registry and pull images from a registry. You can host your own registry or use a provider's registry. For example, AWS and Google Cloud have registries.

Docker Hub is the largest registry of Docker images. It's also the default registry. You can find images and store your own images on Docker Hub for free.

Hubs and spokes

A **Docker Repository** is a collection of Docker images with the same name and different tags. The *tag* is the identifier for the image.

Usually a repository has different versions of the same image. For example, *Python* is the name of the most popular official Docker image repository on Docker Hub. *Python:3.7-slim* refers to the version of the image with the *3.7-slim* tag in the Python repository. You can push a repository or a single image to a registry.

Now let's look at Docker terms related to scaling multiple Docker containers.

Scaling Docker

The following four concepts relate to using multiple containers at once.

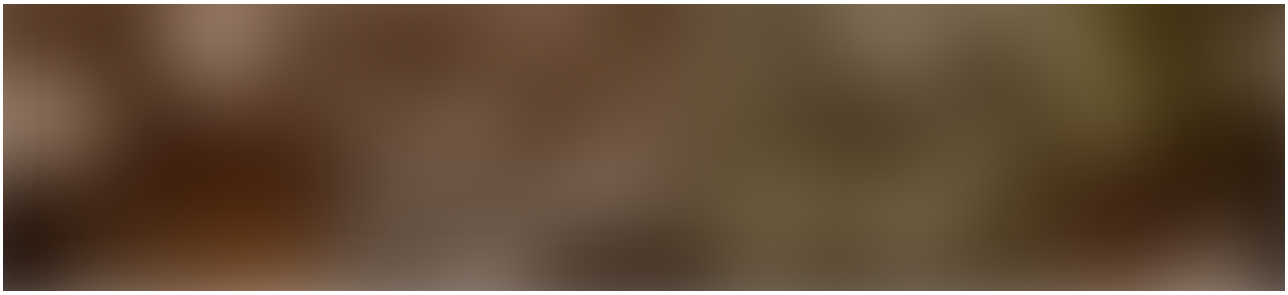
Docker Networking allows you to connect Docker containers together. Connected Docker containers could be on the same host or multiple hosts. For more information

on Docker networking, see this post.

Docker Bridge Network

Docker Compose is a tool that makes it easier to run apps that require multiple Docker containers. Docker Compose allows you to move commands into a `docker-compose.yml` file for reuse. The Docker Compose command line interface (cli) makes it easier to interact with your multi-container app. Docker Compose comes free with your installation of Docker.

Docker Swarm is a product to orchestrate container deployment. The official Docker tutorial has you using Docker Swarm in its fourth section. I would suggest you not spend time on Docker Swarm unless you have a compelling reason to do so.



Bee swarm

Docker Services are the different pieces of a distributed app. From the docs:

> *Services are really just "containers in production." A service only runs one image, but it codifies the way that image runs — what ports it should use, how many replicas of the container should run so the service has the capacity it needs, and so on. Scaling a service changes the number of container instances running that piece of software, assigning more computing resources to the service in the process.*

Docker services allow you to scale containers across multiple Docker Daemons and make Docker Swarms possible.

There you have it: a dozen delicious Docker terms you should know.

Recap

Here's the one line explanation to help you keep these dozen terms straight.

Basics

Platform — the software that makes Docker containers possible
Engine — client-server app (CE or Enterprise)
Client — handles Docker CLI so you can communicate with the Daemon
Daemon — Docker server that manages key things
Volumes — persistent data storage
Registry — remote image storage
Docker Hub — default and largest Docker Registry
Repository — collection of Docker images, e.g. Alpine

Scaling

Networking — connect containers together
Compose — time saver for multi-container apps

Swarm — orchestrates container deployment
Services — containers in production

Because we're keeping with food metaphors, and everyone loves a baker's dozen, we have one more related term for you: *Kubernetes*.

One more donut with extra icing and sprinkles

Kubernetes automates deployment, scaling, and management of containerized applications. It's the clear winner in the container orchestration market. Instead of Docker Swarm, use Kubernetes to scale up projects with multiple Docker containers. Kubernetes isn't an official part of Docker; it's more like Docker's BFF.

I have a whole series on Kubernetes in the works. Kubernetes is pretty awesome.

Now that you know the conceptual landscape and common terms I suggest you try out Docker.

Baking with Docker

If you haven't worked with Docker before, it's time to get in the kitchen and make something!

Docker runs locally on Linux, Mac, and Windows. If you're on a Mac or Windows machine, install the latest stable version of Docker Desktop here. As a bonus, it comes with Kubernetes. If you're installing Docker elsewhere, go here to find the version you need.

After you have Docker installed, do the first two parts of the Docker tutorial. Then meet back here for more Docker fun. In the next four parts of this series we'll dive into Dockerfiles, Docker images, the Docker CLI, and dealing with data. Follow me to make sure you don't miss the adventure.

Part 3 on Dockerfiles is now available here:

Learn Enough Docker to be Useful

Part 3: A Dozen Dandy Dockerfile Instructions

towardsdatascience.com

If you found this article helpful, please share it on Twitter, Facebook, LinkedIn and your favorite forums. Thanks for reading!

Thanks to Kathleen Hale.

Data Science Machine Learning Docker Programming Software Development


Download on the
App Store


GET IT ON
Google Play