Get Started, Part 1: Orientation and setup

Estimated reading time: 4 minutes

Welcome! We are excited that you want to learn Docker. The *Docker Get Started Tutorial* teaches you how to:

1. Set up your Docker environment (on this page)
2. Build an image and run it as one container
3. Scale your app to run multiple containers
4. Distribute your app across a cluster
5. Stack services by adding a backend database
6. Deploy your app to production

Docker concepts

Docker is a platform for developers and sysadmins to **develop, deploy, and run** applications with containers. The use of Linux containers to deploy applications is called *containerization*. Containers are not new, but their use for easily deploying applications is.

Containerization is increasingly popular because containers are:

- Flexible: Even the most complex applications can be containerized.
- Lightweight: Containers leverage and share the host kernel.
- Interchangeable: You can deploy updates and upgrades on-the-fly.
- Portable: You can build locally, deploy to the cloud, and run anywhere.
- Scalable: You can increase and automatically distribute container replicas.
- Stackable: You can stack services vertically and on-the-fly.

Images and containers

A container is launched by running an image. An **image** is an executable package that includes everything needed to run an application--the code, a runtime, libraries, environment variables, and configuration files.

A **container** is a runtime instance of an image--what the image becomes in memory when executed (that is, an image with state, or a user process). You can see a list of your running containers with the command, `docker ps`, just as you would in Linux.

Containers and virtual machines

A **container** runs *natively* on Linux and shares the kernel of the host machine with other containers. It runs a discrete process, taking no more memory than any other executable, making it lightweight.

By contrast, a **virtual machine** (VM) runs a full-blown "guest" operating system with *virtual* access to host resources through a hypervisor. In general, VMs provide an environment with more resources than most applications need.