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 (https://docs.docker.com/get-started/part2/)
3. Scale your app to run multiple containers (https://docs.docker.com/get-started/part3/)
4. Distribute your app across a cluster (https://docs.docker.com/get-started/part4/)
5. Stack services by adding a backend database (https://docs.docker.com/get-started/part5/)
6. Deploy your app to production (https://docs.docker.com/get-started/part6/)

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

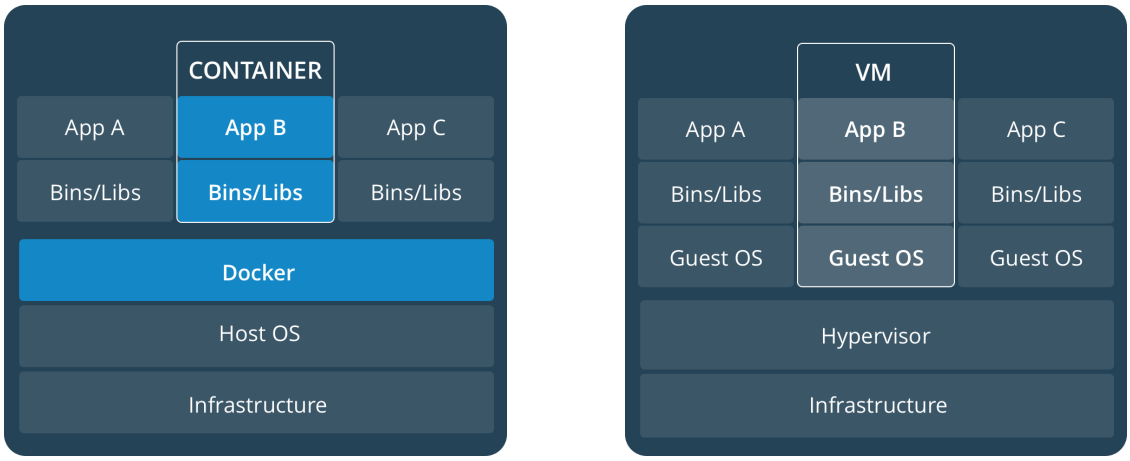
A container is launched by running an image. An **image** is an executable package that includes everything needed to run an application--the code, a runtime, libraries, environment variables, and configuration files.

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.



Prepare your Docker environment

Install a maintained version (https://docs.docker.com/engine/installation/#updates-and-patches) of Docker Community Edition (CE) or Enterprise Edition (EE) on a supported platform (https://docs.docker.com/engine/installation/#supported-platforms).

For full Kubernetes Integration

- Kubernetes on Docker for Mac (https://docs.docker.com/docker-for-mac/kubernetes/) is available in 17.12 Edge (mac45) (https://docs.docker.com/docker-for-mac/edge-release-notes/#docker-community-edition-17120-ce-mac45-2018-01-05) or 17.12 Stable (mac46) (https://docs.docker.com/docker-for-mac/release-notes/#docker-community-edition-17120-ce-mac46-2018-01-09) and higher.
- Kubernetes on Docker for Windows (https://docs.docker.com/docker-for-windows/kubernetes/) is available in 18.02 Edge (win50) (https://docs.docker.com/docker-for-windows/edge-release-notes/#docker-community-edition-18020-ce-rc1-win50-2018-01-26) and higher.

Install Docker (https://docs.docker.com/engine/installation/)

Test Docker version

1. Run `docker --version` and ensure that you have a supported version of Docker:

```
docker --version

Docker version 17.12.0-ce, build c97c6d6
```

2. Run `docker info` or (`docker version` without `--`) to view even more details about your docker installation:

```
docker info

Containers: 0
 Running: 0
 Paused: 0
 Stopped: 0
Images: 0
Server Version: 17.12.0-ce
Storage Driver: overlay2
...
```

To avoid permission errors (and the use of `sudo`), add your user to the `docker` group. Read more (https://docs.docker.com/engine/installation/linux/linux-postinstall/).

Test Docker installation

1. Test that your installation works by running the simple Docker image, hello-world (https://hub.docker.com/_/hello-world/):

```
docker run hello-world

Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
ca4f61b1923c: Pull complete
Digest: sha256:ca0eeb6fb05351dfc8759c20733c91def84cb8007aa89a5bf606bc8b315b9fc7
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.
...
```

2. List the `hello-world` image that was downloaded to your machine:

```
docker image ls
```

3. List the `hello-world` container (spawned by the image) which exits after displaying its message. If it were still running, you would not need the `--all` option:

```
docker container ls --all

CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
54f4984ed6a8        hello-world         "/hello"            20 seconds ago      Exited (0) 19 seconds a
```

Recap and cheat sheet

```
## List Docker CLI commands
docker
docker container --help

## Display Docker version and info
docker --version
docker version
docker info

## Excecute Docker image
docker run hello-world

## List Docker images
docker image ls

## List Docker containers (running, all, all in quiet mode)
docker container ls
docker container ls --all
docker container ls -aq
```

Conclusion of part one

Containerization makes CI/CD (https://www.docker.com/use-cases/cicd) seamless. For example:

- applications have no system dependencies
- updates can be pushed to any part of a distributed application
- resource density can be optimized.

With Docker, scaling your application is a matter of spinning up new executables, not running heavy VM hosts.

On to Part 2 >> (https://docs.docker.com/get-started/part2/)

get started (https://docs.docker.com/glossary/?term=get%20started), setup (https://docs.docker.com/glossary/?term=setup), orientation (https://docs.docker.com/glossary/?term=orientation), quickstart (https://docs.docker.com/glossary/?term=quickstart), intro (https://docs.docker.com/glossary/?term=intro), concepts (https://docs.docker.com/glossary/?term=concepts), containers (https://docs.docker.com/glossary/?term=containers)