

[Installing and Using Docker Toolbox for Mac OSX and Windows](#)

One of the most compelling reasons to run Docker on your local machine is the speed at which you can deploy and build lab environments. As a huge fan of Vagrant, I love the ability to spin up environments such as the [sandbox labs I've been using for a long time with Vagrant and VirtualBox](#).

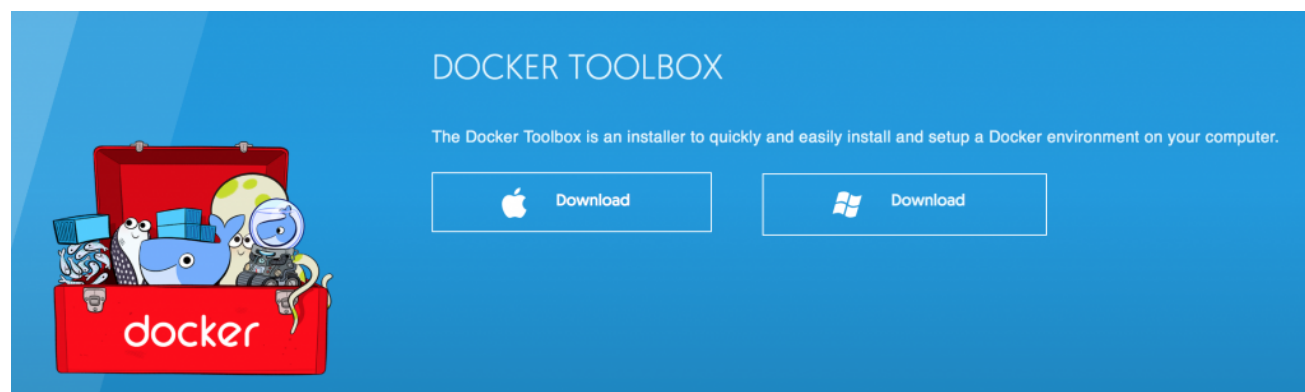
Switching to Docker as an option for many of my quick labs has also meant the same ability to run as an abstraction on top of my laptop so that I don't end up in dependency hell with development libraries and underlying infrastructure needs that quickly begin to conflict as I do more testing and development.

Installing Docker Toolbox on Mac OSX or Windows

The best way to get started is to run the Docker Toolbox platform which deploys a Docker environment with popular and important Docker tools including:

- docker-engine
- docker-compose
- docker-machine
- Kitematic

Navigate over to <https://www.docker.com/products/docker-toolbox> to get your appropriate version:



Rather than document the steps on a continuously changing set of screens, I recommend that you follow the installation process with the tools you desire using the guides provided by Docker here: <https://docs.docker.com/toolbox/overview/>

Once you're installed, you can kick the tires on Docker using your first Docker Hello World test container using the `docker run hello-world` command:

```
DiscoPosse — All your shell are belong to us — -bash — 80x30
[Eric's-MBP-2:~ DiscoPosse$
[Eric's-MBP-2:~ DiscoPosse$
[Eric's-MBP-2:~ DiscoPosse$ docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
78445dd45222: Pull complete
Digest: sha256:c5515758d4c5e1e838e9cd307f6c6a0d620b5e07e6f927b07d05f6d12a1ac8d7
Status: Downloaded newer image for hello-world:latest

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

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://cloud.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/engine/userguide/

Eric's-MBP-2:~ DiscoPosse$ █
```

You can see that the container image was not local, so a download process started and then the container was launched. As long as you see the results like above, you're in business!

We will be using this as a baseline for a lot of other examples in the blog. As usual, this is meant to emulate a basic Docker configuration and does not really reflect a multi-node deployment with overlay networking. The goal is to be able to quickly and easily launch containers using Docker Engine for a number of admin tasks that can replace what we may have been doing inside dedicated workstations or sandbox virtual machines in the past.