First steps with Docker

Installing Docker

Docker supports Linux, MacOS, and Windows platforms. It's straightforward to install Docker on most platforms and we'll get to that in a bit. Docker Inc. provides Community and Enterprise editions of the Docker platform. The Enterprise edition has the same features as the Community edition, but it provides additional support and certified containers, plugins, and infrastructure. For the purpose of this book and for most general development and production use, the Community edition is suitable, thus we will be using that in this book.

Installing Docker on Windows

You need to meet certain prerequisites before you can install Docker on Windows. These include the following:

- Hyper-V support
- Hardware virtualization support, typically be enabled from your system BIOS
- Only 64-bit editions of Windows 10 (Pro/Education/Enterprise editions having the Anniversary Update v1607) are supported at the moment

Notice that this looks like what a virtualization setup would require, and you learned in the previous chapter that Docker is not virtualization. So why does Docker for Windows require features required for virtualization?

The short answer is that Docker relies on numerous features, such as namespaces and cgroups, and these are not available on Windows. To get around this limitation, Docker for Windows creates a lightweight Hyper-V container running a Linux kernel. At the time of writing, Docker includes experimental support for Native containers that allow for creation of containers without the need for Hyper-V. Let's focus on installing Docker CE for Windows. This section assumes that all prerequisites have been met and that Hyper-V is enabled. Head over to https://store.docker.com/editions/community/docker-ce-desktop-windows to download Docker CE.

You may be required to restart your system, as enabling Hyper-V is a Windows system feature. If it's not installed, this feature will be installed and that requires a restart to enable the feature. Once the install is complete, open a command prompt window and type the command shown below to check that Docker is installed and is working correctly :

```
docker run --rm hello-world
```

If the install went fine, you should see this response :

```
docker run --rm hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
ca4f61b1923c: Pull complete
Digest: sha256:66ef312bbac49c39a89aa9bcc3cb4f3c9e7de3788c944158df3ee0176d32b751
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
```

This message shows that your installation appears to be working correctly.
…

Installing on Mac OS

Installing Docker for Mac is much like installing any other application. Go to https://store.docker.com/editions/community/docker-ce-desktop-mac, click the Get Docker for CE Mac (stable) link, and double-click the file to run the installer that is downloaded. Drag the Docker whale to the Applications folder to install it, as shown in Figure 2.

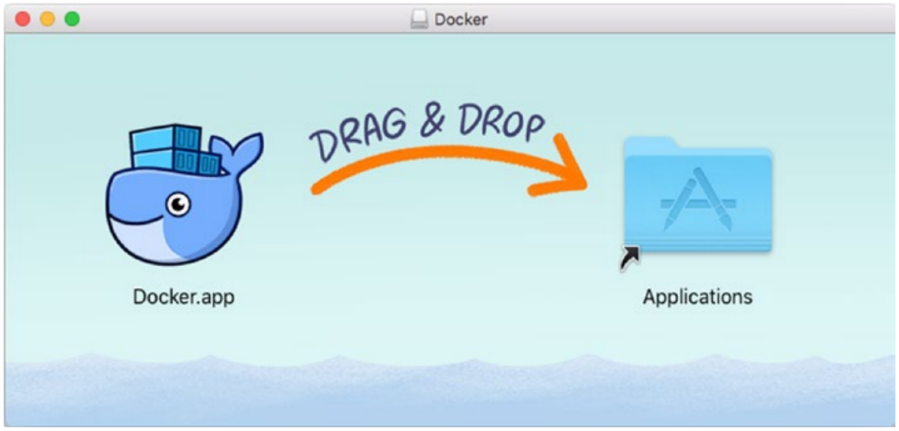

Figure 2. Installing Docker for Mac

Once Docker is installed, open the Terminal app and run the command below to confirm the install was successful :

docker run --rm hello-world

If the install went fine, you should see the response :

docker run --rm hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
ca4f61b1923c: Pull complete
Digest: sha256:66ef312bbac49c39a89aa9bcc3cb4f3c9e7de3788c944158df3ee0176d32b751
Status: Downloaded newer image for hello-world:latest
Hello from Docker!
This message shows that your installation appears to be working correctly.
…

Installing on Linux

To install Docker on Linux, visit https://www.docker.com/community-edition. Select the distro you're using and follow the commands to install Docker.
The following section outlines the steps needed to install Docker on Ubuntu.

1. Update the apt index:
   sudo apt-get update

2. Install the necessary packages required to use a repository over HTTPS:
    sudo apt-get install \
    apt-transport-https \
    ca-certificates \
    curl \
    software-properties-common
3. Install Docker's official GPG key:
    curl -fsSL https://download.docker.com/linux/ubuntu/gpg
    | sudo apt-key add -
4. Add Docker's stable repository:
    sudo add-apt-repository \
     "deb [arch=amd64] https://download.docker.com/linux/ubuntu \
    $(lsb_release -cs) \
    stable"

5. Update the apt package index:
    sudo apt-get update
6. Install Docker:
    sudo apt-get install docker-ce

Check that Docker for Linux is working :

docker run --rm hello-world

You should have this response :
docker run --rm hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
ca4f61b1923c: Pull complete
Digest: sha256:66ef312bbac49c39a89aa9bcc3cb4f3c9e7de3788c944158df3ee0176d32b751
Status: Downloaded newer image for hello-world:latest
Hello from Docker!
This message shows that your installation appears to be working
correctly.
…

First use of Docker

To makes things easy to read and understand, you can use a tool called jq for processing Docker's JSON output. You can download and install jq from https://stedolan.github.io/jq/

Open a terminal window and type the following command:

docker info

If well installed, you should see configuration information of your Docker configuration.

Working with Docker Images

Let’s look at the available Docker images. To do this, type the following command:

docker image ls

Here’s a listing of the images available locally :

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
hello-world	latest	f2a91732366c	2 months ago	1.85kB

If you had pulled more images or run more containers, you’d have seen a bigger list. Let’s look at the hello-world image now. To do this, type the following:

docker image inspect hello-world

The docker inspect command provides a lot of information about the image. Of importance are the image properties Env, Cmd, and Layers, which tell us about these environment variables. They tell us which executable runs when the container is started and the layers associated with these environment variables.

```
docker image inspect hello-world | jq .[].Config.Env
[
  #"PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"
]
```

Here’s the startup command on the container:

```
docker image inspect hello-world | jq .[].Config.Cmd
[
  "/hello"
]
```

Here are the layers associated with the image:

```
docker image inspect hello-world | jq .[].RootFS.Layers
[
  #"sha256:f999ae22f308fea973e5a25b57699b5daf6b0f1150ac2a5c2ea9d7fecee50fdf"
]
```

Working with a Real-World Docker Images

Let’s look at a more complex image now. Nginx is a very popular reverse proxy server for HTTP/S (among others), as well as a load balancer and a webserver. To pull down the nginx image, type the following:

docker pull nginx

```
Using default tag: latest
latest: Pulling from library/nginx
e7bb522d92ff: Pull complete
6edc05228666: Pull complete
cd866a17e81f: Pull complete
```

Digest: sha256:285b49d42c703fdf257d1e2422765c4ba9d3e37768d6ea83d7fe2043dad6e63d
Status: Downloaded newer image for nginx:lates

Notice the first line:

Using default tag: latest

Every Docker image has an associated tag. Tags typically include names and version labels. While it is not mandatory to associate a version tag with a Docker image name, these tags make it easier to roll back to previous versions. Without a tag name, Docker must fetch the image with the latest tag. You can also provide a tag name to force-fetch a tagged image. Docker Store lists the different tags associated with the image. If you're looking for a specific tag/version, it's best to check Docker Store. Figure 3 shows a typical tag listing of an image.

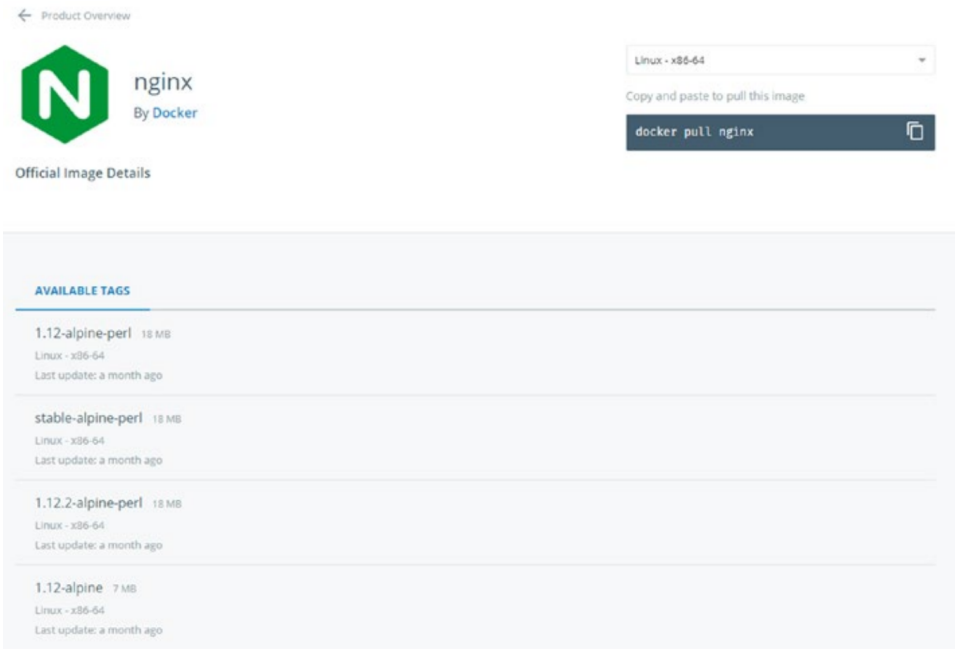

Figure 3. Docker Store listing of nginx and the available tags

Let's try to pull the 1.12-alpine-perl version of nginx. This command is the same as before; you only have to append the tag with a colon to explicitly mention the tag:

docker pull nginx:1.12-alpine-perl

1.12-alpine-perl: Pulling from library/nginx
550fe1bea624: Pull complete
20a55c7b3b0e: Pull complete
552be5624b14: Pull complete
40fc04944e91: Pull complete
Digest: #sha256:b7970b06de2b70acca1784ab92fb06d60f4f714e901a55b6b5211c22a446dbd2
Status: Downloaded newer image for nginx:1.12-alpine-perl

The different hex numbers that you see are the associated layers of the image.

By default, Docker pulls the image from Docker Hub. You can manually specify a different registry, which is useful if the Docker images are not available on Docker Hub and are instead stored elsewhere, such as an on-premise hosted artifactory. To do this, you have to prepend the registry path to the image name. So, if the registry is hosted on docker-private.registry and is being served on 1337 port, the pull command will now be:

docker pull docker-private.registry:1337/nginx

If the registry needs authentication, you can log in to the registry by typing docker login:

docker login docker-private.registry:1337

Now that you have the image, try to start a container. To start a container and run the associated image, you have to type docker run.

docker run -p 80:80 nginx

Host will wait for connections while console remains open. You must detach nginx with « -d » option in order to take back control of the console.

Let’s try making a curl request to see if the nginx webserver is running:

curl http://localhost:80

This confirms that our nginx container is indeed up and running. In this, we see an extra flag called -p. This flag tells Docker to publish the exposed port from the Docker container to the host.

The first parameter after the flag is the port on the Docker host that must be published and the second parameter refers to the port within the container. We can confirm that the image publishes the port using the docker inspect command:

```
docker image inspect nginx | jq .[].Config.ExposedPorts
{
  "80/tcp": {}
}
```

We can change the port on which the service is published on the Docker host by changing the first parameter after the -p flag:

docker run -p 8080:80 nginx

Now, try running a curl request to port 8080:

curl http://localhost:8080

You should see the same response. To list all the running containers, you can type docker ps:

docker ps

The point to note is the NAMES column. Docker automatically assigns a random name when a container is started. Since you'd like more meaningful names, you can provide a name to the container by providing -n required-name as the parameter.

Another point to note is that when we created a second container with port publishing to port 8080, the other container continues to run. To stop the container, you have to type docker stop:

docker stop <container-id>

where container-id is available from the list. If the stop was successful, Docker will echo the container ID back. If the container refuses to stop, you can issue a kill command to force stop and kill the container(docker kill <container-id>). Type docker ps to confirm that the containers are no longer running.
So, what about the stopped container, where are they? By default, docker ps only shows the active, running containers. To list all the containers, type the following:

docker ps -a

Even though the containers have been stopped and/or killed, these containers continue to exist in the local filesystem. You can remove the containers by typing the following:

docker rm <container-id>

retry docker ps -a

Similarly, you can list all the images present in the system by typing the following:

docker image ls

Let's try to remove the nginx image. In this case, Docker refuses to remove the image because there is a reference to this image from another container. Until we remove all the containers that use a particular image, we will not be able to remove the image altogether.

Now, go on docker Hub, and try to run another image of your choice !