**What is Docker?**

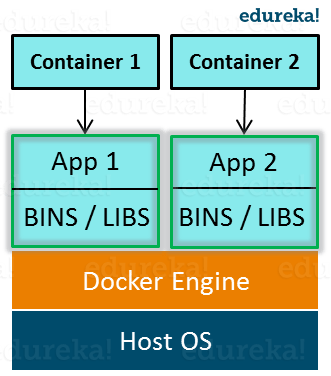
Build, Ship, and Run Any App, Anywhere

Docker is a tool designed to make it easier to deploy, and run applications by using containers.

Containers allow a developer to package up an application with all of the parts it needs, such as libraries and other dependencies, and ship it all out as one package.

Docker is a containerization platform that packages your application and all its dependencies together in the form of Containers to ensure that your application works seamlessly in any environment.

* Each application will run on a separate container and will have its own set of libraries and dependencies.
* It also ensures that there is process level isolation, meaning each application is independent of other applications, giving developers surety that they can build applications that will not interfere with one another.



Though Docker containers are sometimes referred to as ‘light-weight VMs’, they are not VMs (virtual machines). Both serve different purposes, and hence one cannot replace the other. Their underlying architecture is what differentiates Docker from Virtual machines. While VMs are based on server virtualization technology, Docker uses container virtualization.

Docker is a tool

Docker is designed to make it easier to create, deploy, and run applications by using containers.

Containers allow a developer to package up an application with all of the parts it needs, such as libraries and other dependencies.

Docker is a bit like a virtual machine.

Docker is open source.

Docker is a tool that is designed to benefit both developers and system administrators.

A container is a lightweight, resource-controlled runtime environment, running on a host or virtual machine

**Docker Images**

It contains a list of commands and instruction on how to build and run a container. So basically Images contains all the data and metadata required to fire up a container(also called blueprint).We can't lunch a container without specifying Images.

Images may come from two sources:

* image repository, which is Docker Hub.
* you can create your own images. We will talk about this in more details, because this part is awesome. Docker images are layered, so you can build them layer by layer, and you can build your images starting from other images.

**Dockerfile**

A Dockerfile is a text file that defines a Docker image. You’ll use a Dockerfile to create your own custom Docker image, in other words to define your custom environment to be used in a Docker container.

A Dockerfile is a file that you create which in turn produces a Docker image when you build it.

A Dockerfile is a text file that Docker reads in from top to bottom. It contains a bunch of instructions which informs Docker HOW the Docker image should get built.

You can relate it to cooking. In cooking you have recipes. A recipe lets you know all of the steps you must take in order to produce whatever you’re trying to cook.

The act of cooking is building the recipe.

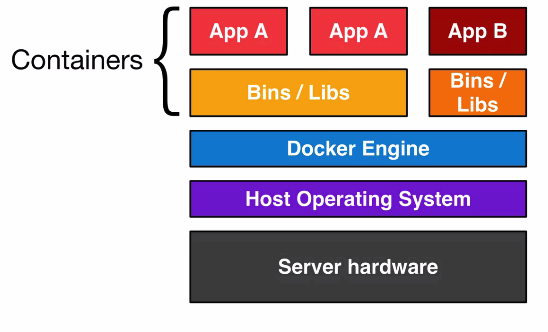
A Dockerfile is a recipe (or blueprint if that helps) for building Docker images, and the act of running a separate build command produces the Docker image from that recipe.

**Docker Container**

Containers are lunch from Images so we can say container is the running instance of an Images. Container is a runtime construct, unlike Images which is build time construct.

Here’s an easy thing to relate to if you’re a software developer:

In the world of object oriented programming, you often deal with classes. You can think of a Docker image as a class, where as a Docker container is an instance of that class.



**Differences between a Dockerfile, Docker Image and Docker Container**

DockerFile--(Build)-->DockerImage --(run)-->DockerContainer

**DockerFile** is what you or developer write code to do something (ex- Install)

**Docker Image** is you get when you build dockerfile .

**Docker Container** is you get when you run your Docker image

We can get Docker Image from docker hub by pulling and then run it to get container .

### Images [like vm]

* Read only template used to create containers
* Buuilt by you or other Docker users
* Stored in the Docker Hub or your local Registry

### Containers [like a runing machine]

* Isolated application platform
* Contains everything needed to run your application
* Based on images

Docker - Hub

Docker Hub is a registry service on the cloud that allows you to download Docker images that are built by other communities. You can also upload your own Docker built images to Docker hub.

Website: <http://hub.docker.com>

**Docker Run Sample image which is already existing in docker hub**

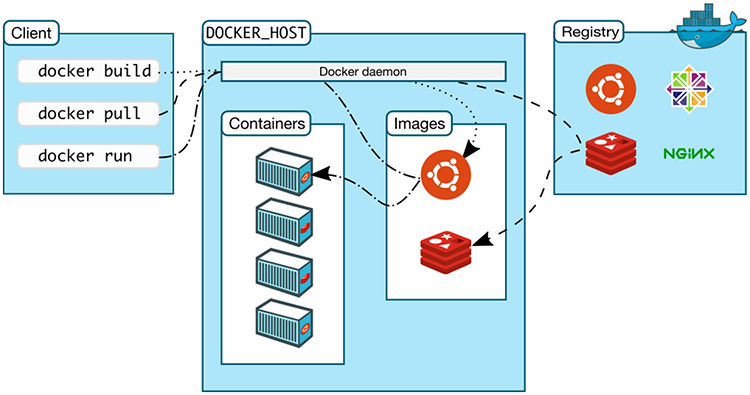
docker run hello-world

Note:1.Only .Net core supports linux containers

2 Dotnet Framework supports only windows containers.

Note: **The Docker daemon** is a service that runs on your [host operating system](https://nickjanetakis.com/blog/comparing-virtual-machines-vs-docker-containers).

### How the Client Talks to the Docker Host



* **First we have the client on the left**, this is where we’re running various Docker commands. The client could be installed on your laptop running Windows, MacOS or a server running Linux, it doesn’t matter.
* **Next up, we have the Docker host**. This is typically referred to as the server running the Docker daemon. That makes sense right? It’s the host that happens to be running the Docker daemon.

It’s very simple to configure the Docker client to connect to a remote Docker host. This is one way you’re able to run Docker on MacOS and Windows.

In that case, the Docker daemon ends up running in a virtual machine that uses Linux, and the Docker client is configured to connect to that remote Docker host.

The key take away here is, the client and daemon does not need to be on the same box.

NOTE: Below are other possible commands which can be used in Dockerfile

|  |  |
| --- | --- |
| Command | Description |
| ADD | Copies a file from the host system onto the container |
| CMD | The command that runs when the container starts |
| ENTRYPOINT |  |
| ENV | Sets an environment variable in the new container |
| EXPOSE | Opens a port for linked containers |
| FROM | The base image to use in the build. This is mandatory and must be the first command in the file. |
| MAINTAINER | An optional value for the maintainer of the script |
| ONBUILD | A command that is triggered when the image in the Dcokerfile is used as a base for another image |
| RUN | Executes a command and save the result as a new layer |
| USER | Sets the default user within the container |
| VOLUME | Creates a shared volume that can be shared among containers or by the host machine |
| WORKDIR | Set the default working directory for the container |

Now run the above image created on the Docker Container using below command.

**Other Popular Docker commands:**

* docker run – Runs a command in a new container.
* docker start – Starts one or more stopped containers
* docker stop – Stops one or more running containers
* docker build – Builds an image form a Docker file
* docker pull – Pulls an image or a repository from a registry
* docker push – Pushes an image or a repository to a registry
* docker export – Exports a container’s filesystem as a tar archive
* docker exec – Runs a command in a run-time container
* docker search – Searches the Docker Hub for images
* docker attach – Attaches to a running container
* docker commit – Creates a new image from a container’s changes