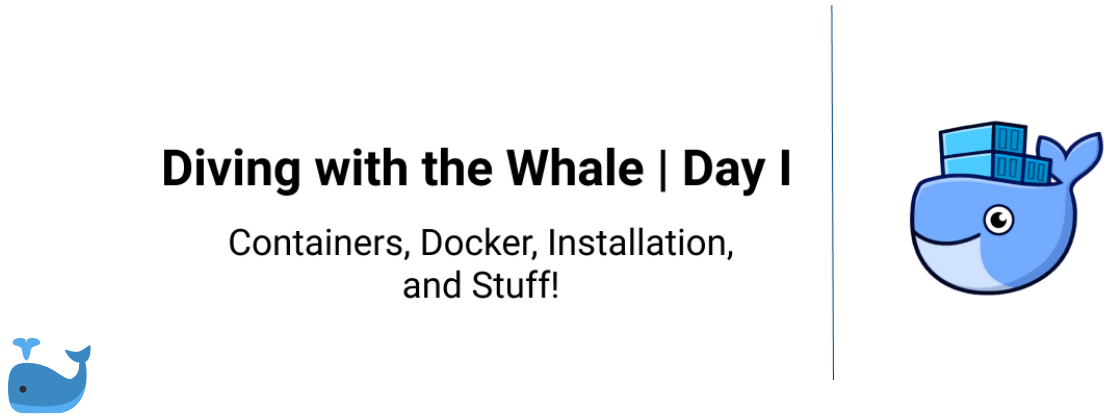


Diving with the Whale - Docker Day I - Intro & Installation

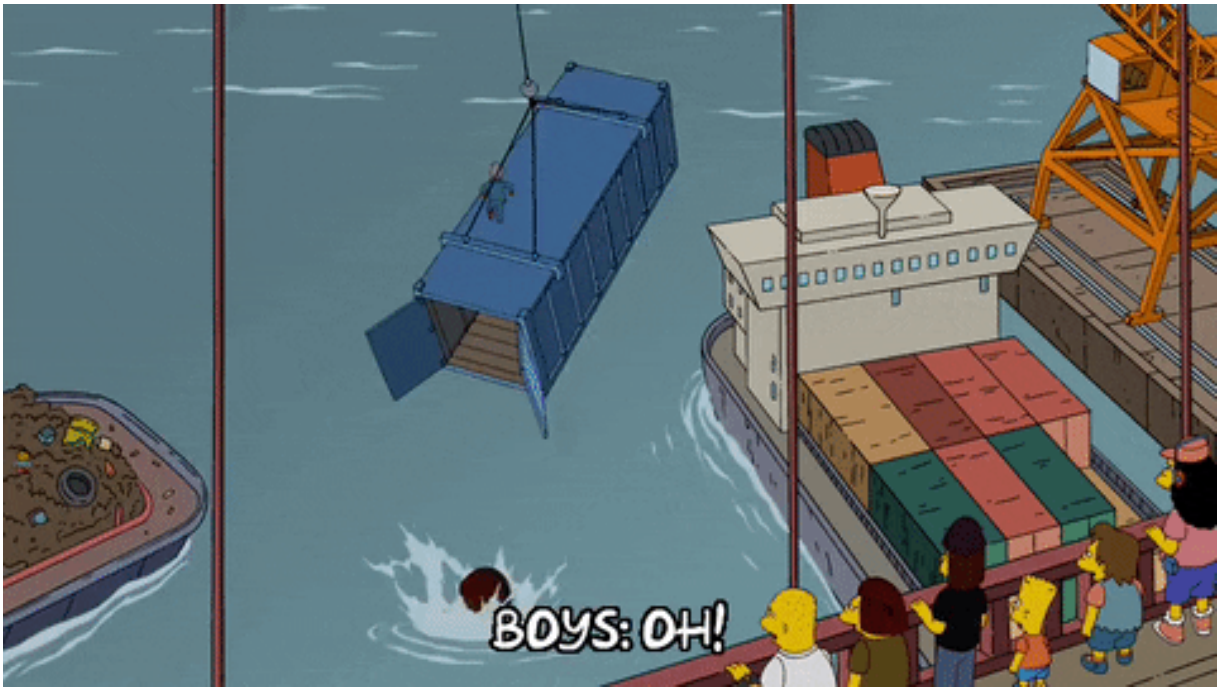
Author : Ahmed Ayman

> "Believe it or not, Evergreen vs. the Suez Canal was **NOT** the biggest container event of the year. #DockerCon is!"
> - Docker official page on Facebook.

Containers Overview

What are Containers ?

Containers, in short, contain applications in a way that keep them **isolated** from the **host system** that they run on.
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**.

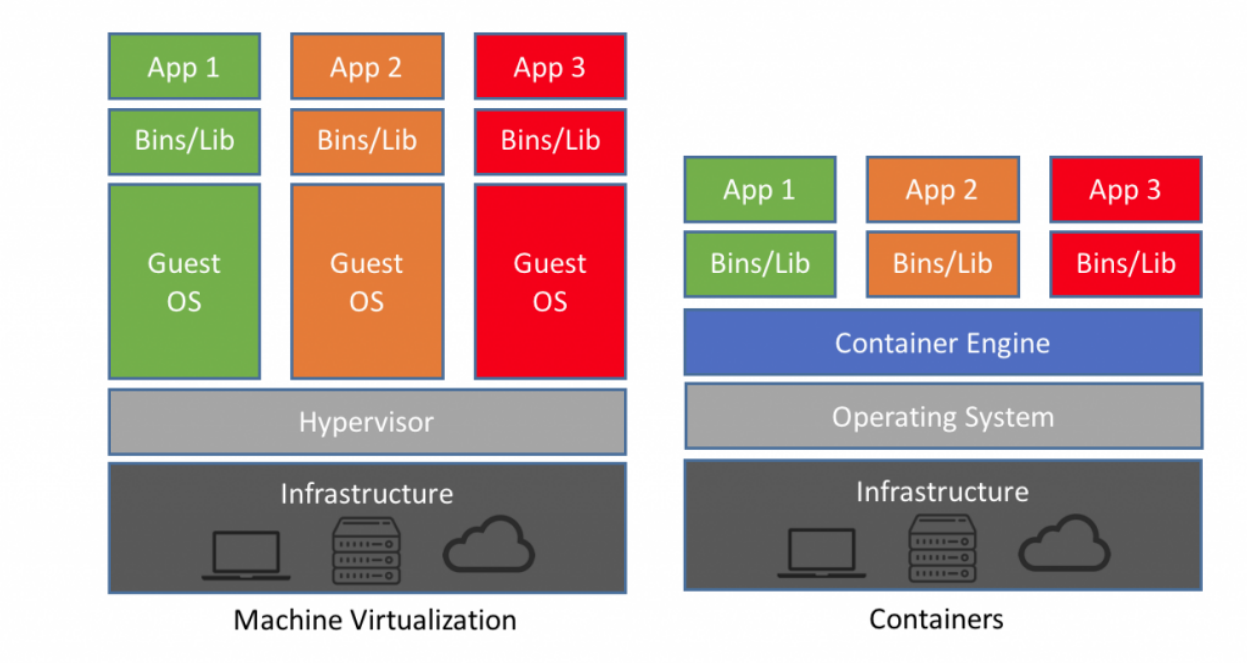


WE WILL NOT DO THAT!

Just as shipping containers, which allow items to be transported by ship, train, or truck independent of the items inside, software containers act as a standard unit of deployment that can contain different code and dependencies which can be deployed across different environments.

So why should we use a container ?

Containers VS Virtual Machines



Containers

Containers are an abstraction at the app layer that packages code and dependencies together. Multiple containers can run on the same machine and **share the OS kernel** with other containers, each running as isolated processes in user space.
Containers take up less space than VMs (container images are typically tens of MBs in size), can handle more applications and require fewer VMs and Operating systems.

VMs

Virtual machines (VMs) are an abstraction of physical hardware turning one server into many servers.
The hypervisor allows multiple VMs to run on a single machine. Each VM includes a full copy of an operating system, the application, necessary binaries and libraries – taking up tens of GBs. VMs can also be slow to boot.

Containers ≠ Docker

Docker is a Container Engine and not the only one, actually it's not even the first one!
But we can say it's almost the best one.

Do you know any other Container Engines ?


LXC


Rocket

Docker

What is Docker ?

Docker is a platform which packages an application and all its dependencies together in the form of container image and run these containers.

When we say dependencies, we mean an application need Unix environment, with java installed, with some scripts or libraries installed, with certain folders created, and application installed into a specific location.

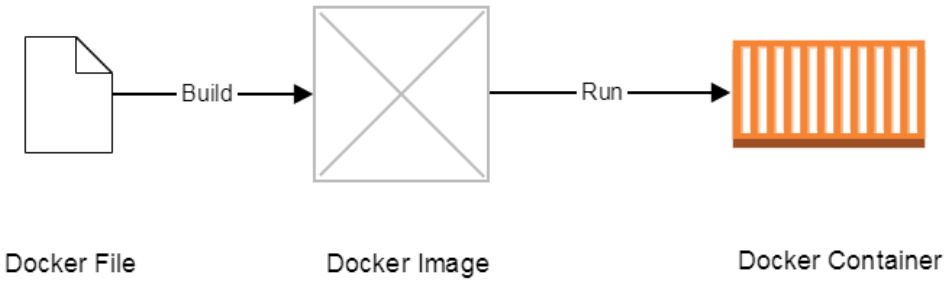
So a developer can build a container image having different applications installed on it and give it to his peer developers and the QA team.
Then the QA team would only need to run the container image to replicate the developer's environment.

Why Docker ?

- Consistent development, testing, and deployment.
- Fixes "It works on my machine" problem!
- Docker allows for rapid development.
- Scalability
  - Due to their size, containers can easily and quickly scale to meet demands
- Portability
  - Applications packaged into Docker Images can run anywhere Docker can be installed

This is the story of a Dockerized app

> Once upon a time a dockerized app was born a **Docker-file**, then a developer fellow came and built the docker-file to grow up and become a **Docker-Image** , After that everyone working on the app took the docker-image and ran it on their machines to finally breathe and become a **Docker-Container**!
> - **Ahmed Ayman**, 2021 - yeah am quoting my-self lol.



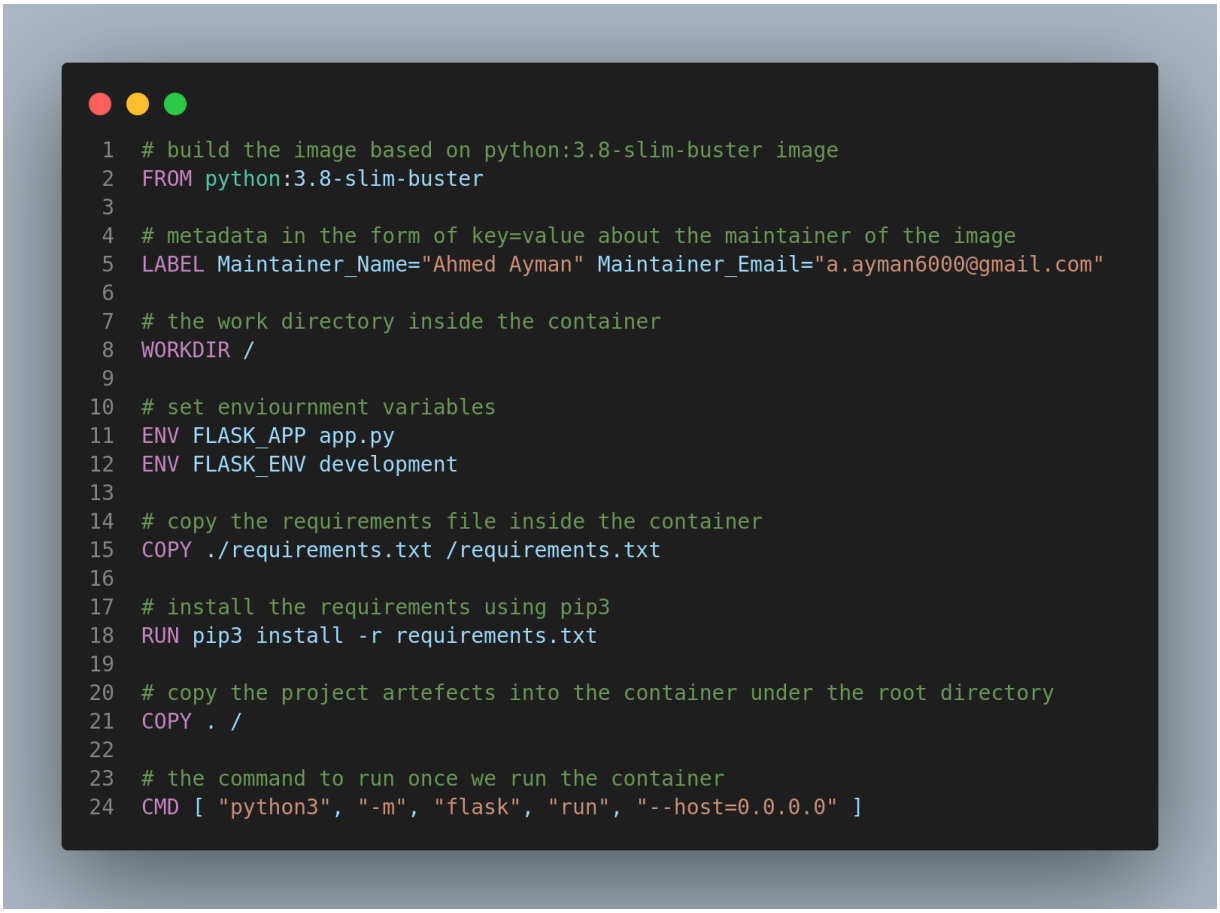
What is Docker-file ?

Text Document with commands telling docker engine exactly how to build the docker image.

For example, a docker-file will have instructions like

1. Use Unix image as a base image
2. Update the package-manager repository
3. Install python
4. Copy the project files on it
5. Open ports
6. Run the app using this command

here is an example docker-file of a flask web app



```dockerfile
# build the image based on python:3.8-slim-buster image
FROM python:3.8-slim-buster

# metadata in the form of key=value about the maintainer of the image
LABEL Maintainer_Name="Ahmed Ayman" Maintainer_Email="a.ayman6000@gmail.com"

# the work directory inside the container
WORKDIR /

# set enviournment variables
ENV FLASK_APP app.py
ENV FLASK_ENV development

# copy the requirements file inside the container
COPY ./requirements.txt /requirements.txt

# install the requirements using pip3
RUN pip3 install -r requirements.txt

# copy the project artefects into the container under the root directory
COPY . /

# the command to run once we run the container
CMD [ "python3", "-m", "flask", "run", "--host=0.0.0.0" ]
```

What is a Docker-Image ?

A Docker image is a read-only template that contains a set of instructions for creating a container that can run on the Docker platform on any machine that has only docker installed on it. It provides a convenient way to package up applications

Docker image is created from the Docker-file. And the Docker-file contains instructions to include environment, framework, artifacts, configurations etc, So the Docker image actually contains all these.

An image includes everything, from operating system to all the dependencies (such as frameworks) plus deployment and execution configuration to be used by a container runtime.

Having everything within the image allows you to migrate images between different environments and be confident that if it works in one environment, then it will work in another. **"IT WORKS ON MY MACHINE" PROBLEM SOLVED!**

What is a Docker Container?

A single running/stopped instance of a docker image.

A container represents the execution of a **Single** application, process, or service. When scaling a service, you create multiple instances of a container from the same image.

You can think of the relation between the image and it's container(s) as many things

- the Image as a class, and the container as an object or instance of that class
- the image as a Linux distro burned on USB, and the container an installed Linux distro on a machine using this image burned on the USB
- the relation between the image and the container is one image to many containers, as the image can have multiple containers spanned from it, but each container can come only from one image.

What is a Registry?

A place to store Docker Images and pull them whenever you want.

A storage and content delivery system, holding named Docker images.

It can be either a user's local repository or a central repository that is available to public like Docker Hub, which allows multiple users to share registry. A little similar to git and GitHub.

Examples:

- Azure Container Registry
- Google Cloud Registry
- Docker Hub (Default for docker)

How is complete process looks like when using docker?

- Create an app or a service, and build the artifacts
- Write/Update Docker-file
- Build Docker image from Docker-file
- Push Docker image to Registry (optional)
- Run Docker image to create Docker container instance.

Docker Installation - we're finally here! -

Docker Installation Documentation

Docker Installation on Linux

if you don't use Linux, go install Linux then get back ****kidding**** you'll find the steps for installing docker on windows & Mac in the documentation.

Install using the repository

1. Set up the repository

```
$ sudo apt-get remove docker docker-engine docker.io containerd runc
$ sudo apt-get update
```

```
$ sudo apt-get install \
    apt-transport-https \
    ca-certificates \
    curl \
    gnupg \
    lsb-release
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg
$ echo \
  "deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/ubuntu \
  $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

2. Install Docker Engine

```
$ sudo apt-get update
$ sudo apt-get install docker-ce docker-ce-cli containerd.io
```

3. Verify that Docker Engine is installed correctly by running the hello-world image.

```
$ sudo docker run hello-world
```

4. Create a docker group

```
$ sudo groupadd docker
```

5. Add your user to the docker group.

```
$ sudo usermod -aG docker $USER
```

6. Activate the changes to groups

```
$ newgrp docker
```

7. Verify that you can run docker commands without Sudo

```
$ docker run hello-world
```