Docker & Kubernetes

Why Should I learn Docker?

I get it. You're in the field of data science. You think the DevOps guys can take care of Docker. Your boss didn't ask you to become an expert (unlike mine!). You feel you don't really need to understand Docker.

That's not true, and let me tell you why ?

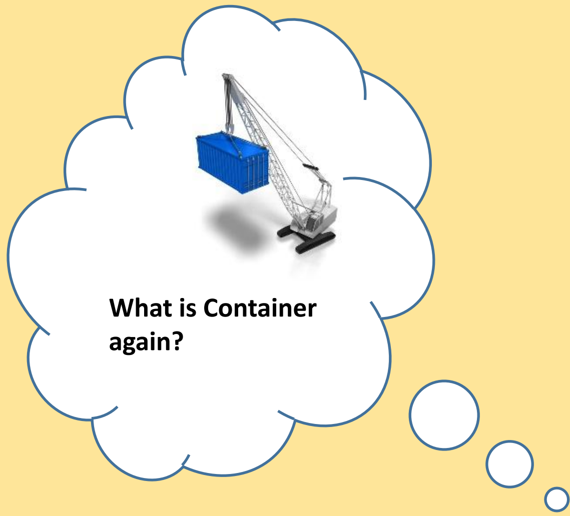
"Not sure why it's not working on your machine, it's working on mine. Do you want me to have a look?"

Ever heard these words uttered at your workplace? Once you (and your team) understand Docker, nobody will ever have to utter those words again. Your code will run smoothly in Ubuntu, Windows, AWS, Azure, Google Cloud, or anywhere, as a matter of fact.

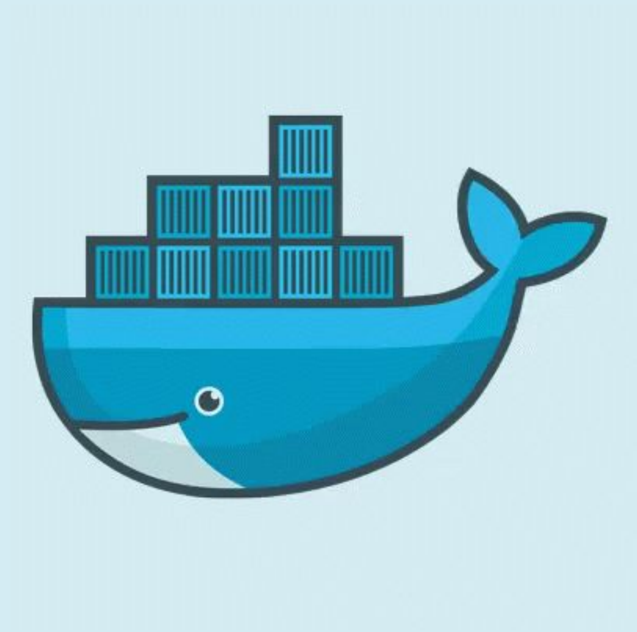
“The applications you build become reproducible anywhere “

What is Docker?

An open, **containerization** platform for developers and sysadmins to build, ship, and run distributed applications. It enables to package and deploy an application or service as an isolated unit containing all of its dependencies whether on laptops, data center VMs, or the cloud.



What is Container?



Containers, or Linux Containers, are a technology that allows us to isolate certain kernel processes and trick them into thinking they're the only ones running in a completely new computer.

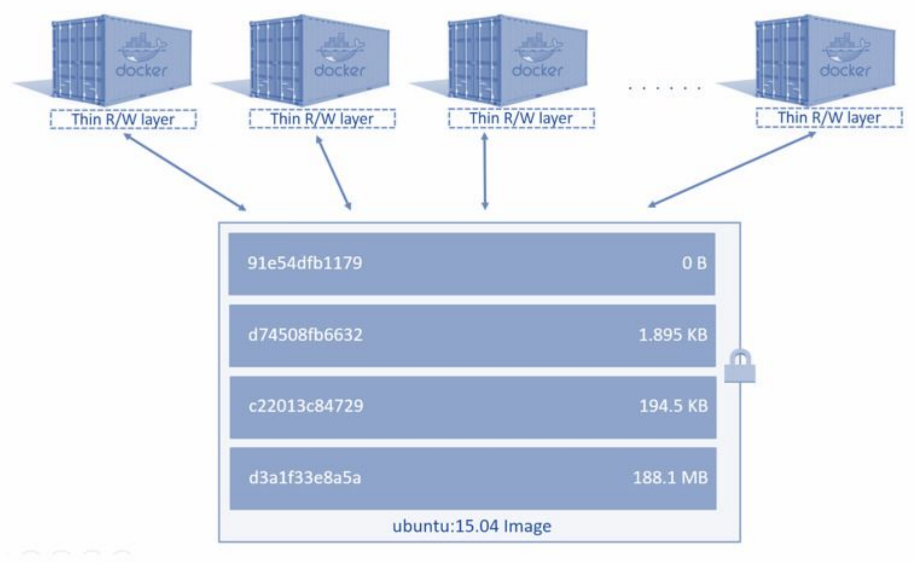
Docker packages software into **standardized units** called containers that have everything the software needs to run including libraries, system tools, code, and runtime. Using Docker, you can quickly deploy and scale applications into any environment and know your code will run.

What is Docker Image?

- Docker Images are made up of multiple layers that are stacked on top of each other and represented as a single object. These are the read-only template that is used to create a Docker container. Because containers are intended to be fast and lightweight, images tend to be small. The official Alpine Linux image is about 5MB in size and official Ubuntu image is of 40MB.

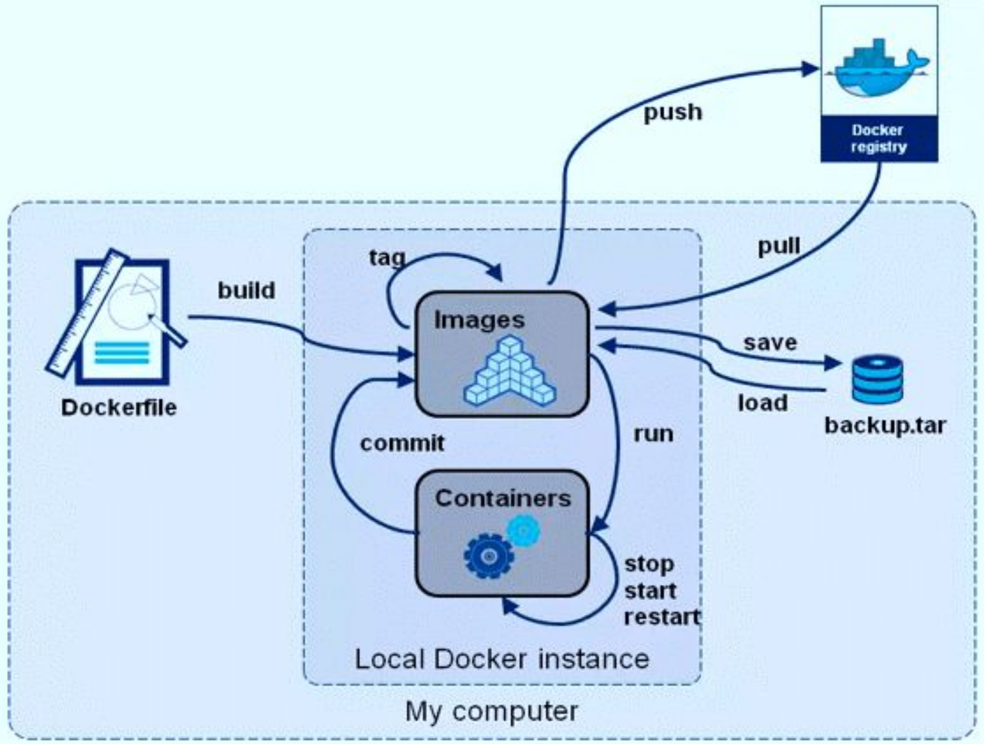
- These images are very similar to the VM image, but there is some difference between them:

- VM image is used to create VM machine and Docker images are used to create Docker containers.
- VM image is big in size while Docker images are lightweight.



Note: Each container has its own writable container layer, and all changes are stored in this container layer, multiple containers can share access to the same underlying image and yet have their own data state.

What is Docker Image Registry?



- **Image Registry** is centralized storage used to store container images, which makes these images easily shareable. We can pull and push our images into this repository. There are three main types of registries: Docker Hub, Third-party Registry services and Self-hosted registry.

Third-Party Registry Services

Red Hat Quay

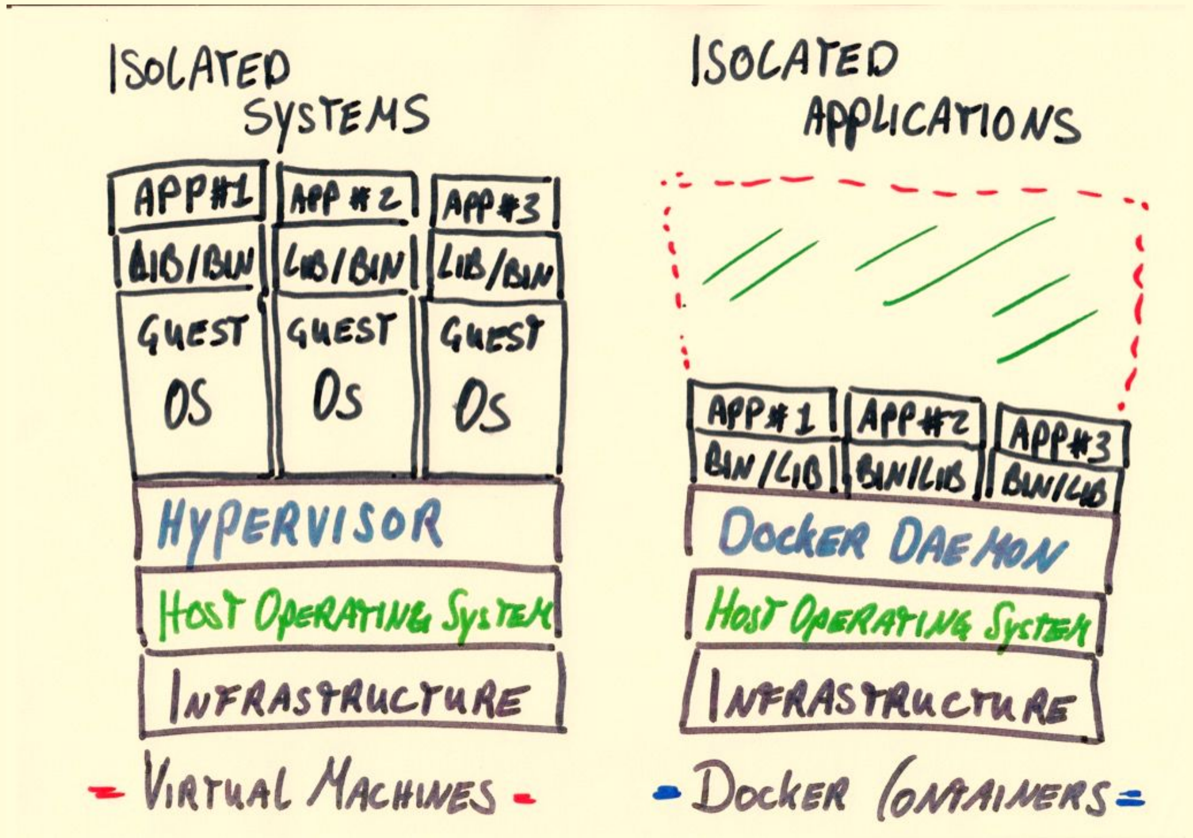
Google Container Registry

Azure Container Registry

Amazon ECR

Third-party Registry services provides a central point to store and manage the user’s private Docker container images and related artifacts. We can even maintain control over who can access, view, or download Docker Images.
Examples: Red Hat Quay, Amazon ECR, Azure Container Registry, Google Container Registry

Is VM and Container Same?

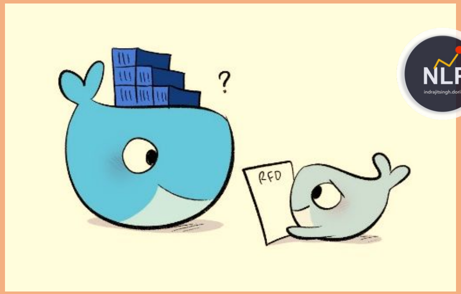


- Different from Virtual Machines, a container can share the kernel of the operating system while only having their different binaries/libraries loaded with them.

- In other words, you don't need to have whole different OS (called guest OS) installed inside your host OS. You can have several containers running within a single OS without having several different guest OS's installed.

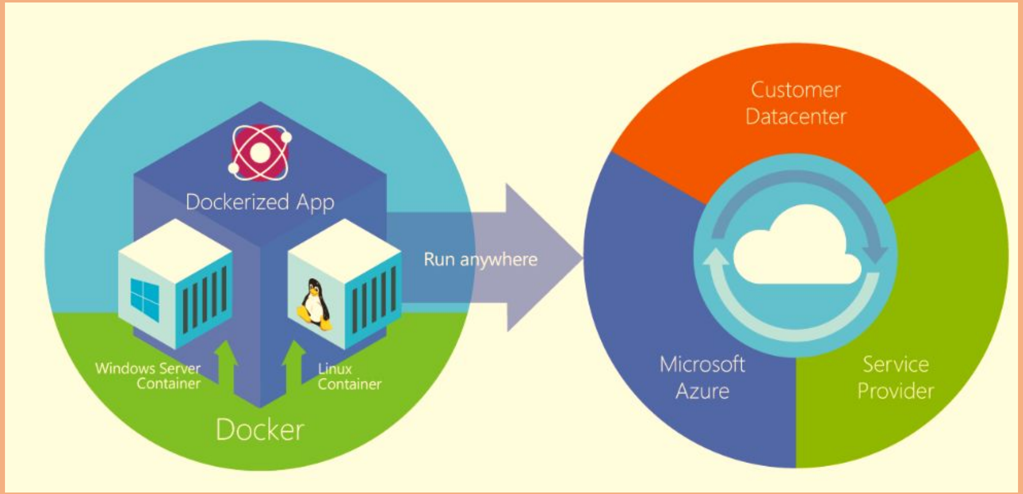
This makes containers much smaller, faster, and more efficient. While a VM can take about a minute to spin up and can weigh several Gigabytes, a container weighs, on average, 400 to 600mb (the biggest ones).

Comparing VMs with Container



- A container image is a way to package an app or service and deploy it in a reliable and reproducible way. You could say that Docker isn't only a technology but also a philosophy and a process.

- When using Docker, you won't hear developers say, "It works on my machine, why not in production?" They can simply say, "It runs on Docker", because the packaged Docker application can be executed on any supported Docker environment, and it runs the way it was intended to on all deployment targets (such as Dev, QA, staging, and production)



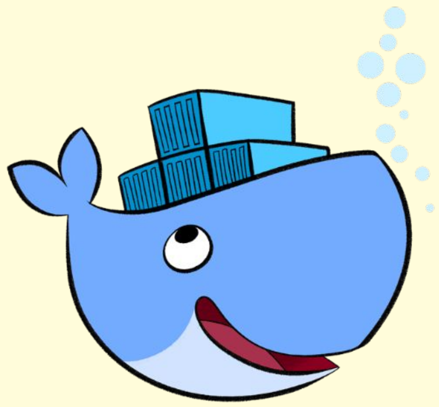
How does DOCKER work ?

- Docker packages an application and all its dependencies in a virtual container that can run on any Linux server. This is why we call them containers. Because they have all the necessary dependencies contained in a single piece of software.

- Docker is composed of the following elements:

  - a Daemon, which is used to build, run, and manage the containers
  - a high-level API which allows the user to communicate with the Daemon,
  - and a CLI, the interface we use to make this all available.

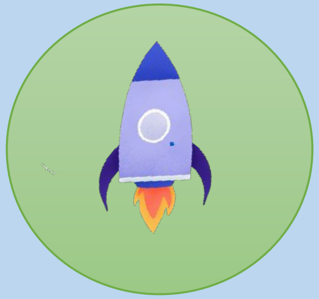
- Docker works by providing a standard way to run your code.
- Docker is an operating system for containers. Similar to how a virtual machine virtualizes (removes the need to directly manage) server hardware, containers virtualize the operating system of a server.
- Docker is installed on each server and provides simple commands you can use to build, start, or stop containers.



Why use DOCKER ?

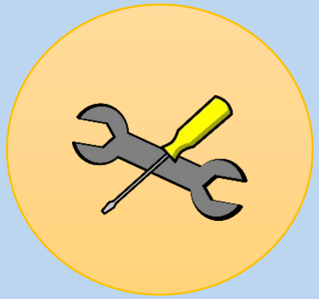
- Using Docker lets you ship code faster, standardize application operations, seamlessly move code, and save money by improving resource utilization. With Docker, you get a single object that can reliably run anywhere. Docker's simple and straightforward syntax gives you full control. Wide adoption means there's a robust ecosystem of tools and off-the-shelf applications that are ready to use with Docker.

SHIP MORE SOFTWARE FASTER



Docker users on average ship software 7x more frequently than non-Docker users. Docker enables you to ship isolated services as often as needed.

STANDARDIZE OPERATIONS

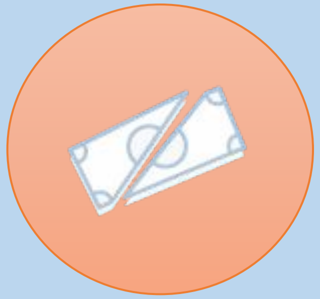


Small containerized applications make it easy to deploy, identify issues, and roll back for remediation.

SEAMLESSLY MOVE

Docker-based applications can be seamlessly moved from local development machines to production deployments on AWS.

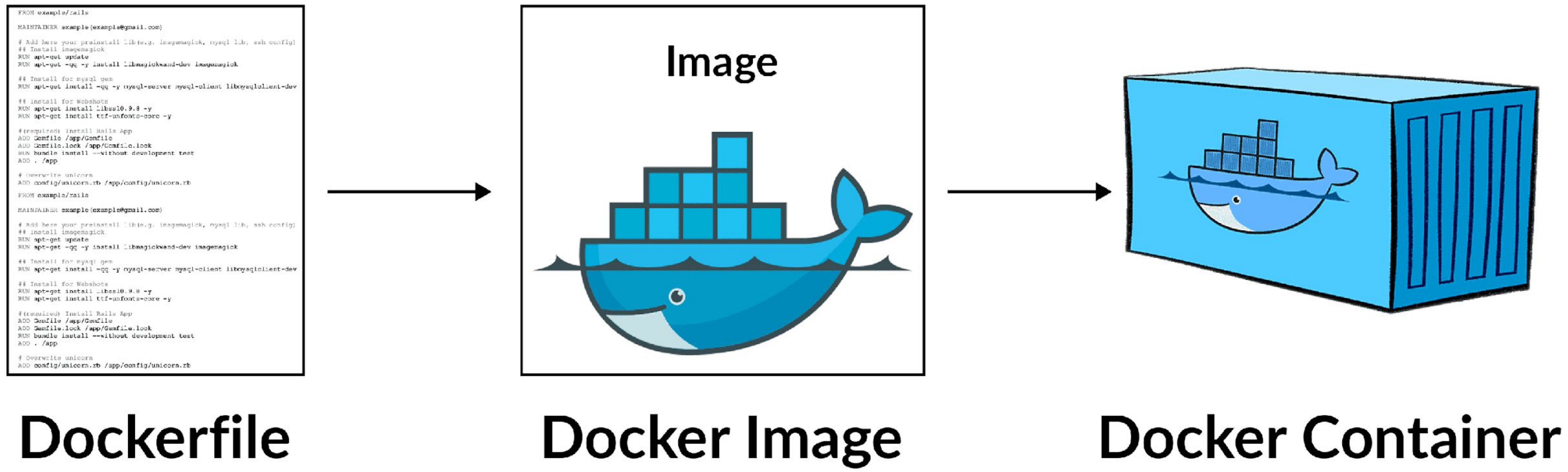
SAVE MONEY



Docker containers make it easier to run more code on each server, improving your utilization and saving you money.

What is DOCKER File?

- A Dockerfile is a document file that contains collections of commands that will be executed in the docker environment for building a new docker image. This file is written in **YAML** Language. These images consist of read-only layers each of which represents a Dockerfile instruction. It is a more systematic, flexible and efficient way to build a Docker image.

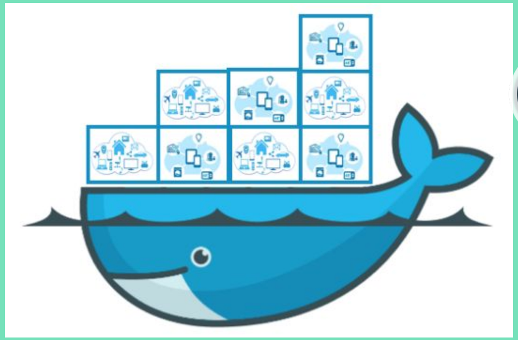


```
FROM example/rails

MAINTAINER example(example@gmail.com)

# Add here your preinstall lib(e.g. imagemagick, mysql lib, ssh config)
## Install imagemagick
RUN apt-get update
RUN apt-get -qq -y install libmagickwand-dev imagemagick

## Install for mysql gem
RUN apt-get install -qq -y mysql-server mysql-client libmysqlclient-dev

## Install for Webshots
RUN apt-get install libssl0.9.8 -y
RUN apt-get install ttf-unfonts-core -y

#(required) Install Rails App
ADD Gemfile /app/Gemfile
ADD Gemfile.lock /app/Gemfile.lock
RUN bundle install --without development test
ADD . /app

# Overwrite unicorn
ADD config/unicorn.rb /app/config/unicorn.rb
FROM example/rails

MAINTAINER example(example@gmail.com)

# Add here your preinstall lib(e.g. imagemagick, mysql lib, ssh config)
## Install imagemagick
RUN apt-get update
RUN apt-get -qq -y install libmagickwand-dev imagemagick

## Install for mysql gem
RUN apt-get install -qq -y mysql-server mysql-client libmysqlclient-dev

## Install for Webshots
RUN apt-get install libssl0.9.8 -y
RUN apt-get install ttf-unfonts-core -y

#(required) Install Rails App
ADD Gemfile /app/Gemfile
ADD Gemfile.lock /app/Gemfile.lock
RUN bundle install --without development test
ADD . /app

# Overwrite unicorn
ADD config/unicorn.rb /app/config/unicorn.rb
```

Common Dockerfile statements

Command	Purpose
FROM	To specify the base image which we want to use.
WORKDIR	To define the working directory for any commands that follow in the Dockerfile.
RUN	To install a package or any application.
COPY	To copy over files or directories from a specific location
ADD	Same as COPY, but we can also use a URL instead of a local file / directory and we can extract a tar file from the source directly into the destination.
ENTRYPOINT	Command that will always be executed when the container starts. If not specified, the default is /bin/sh -c
CMD	To define a default command to run when your container starts.
EXPOSE	To define which port through which to access your container application.
LABEL	To add metadata to the image.

Example Dockerfile



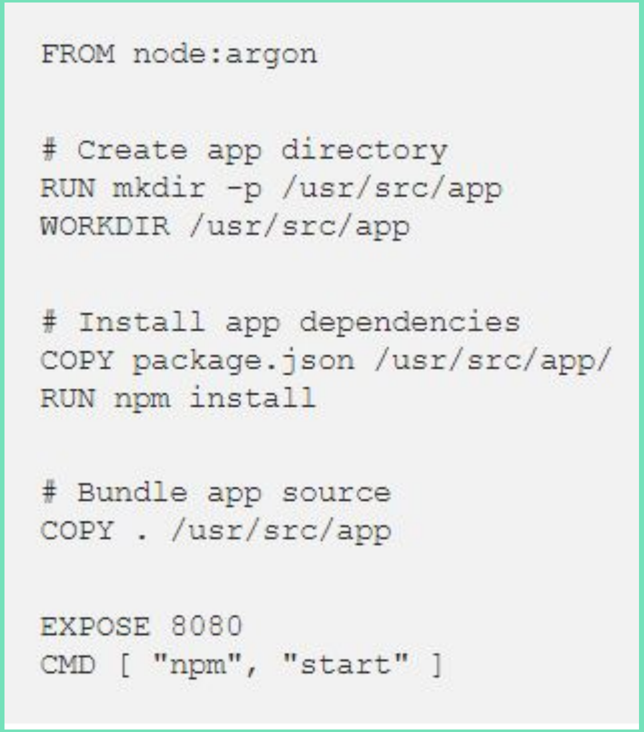
```
# Using the official Ubuntu as base

FROM Ubuntu

RUN apt-get update

RUN apt-get install -y nginx

COPY index.nginx-debian.html
 /var/www/html

EXPOSE 80

COPY [“./start.sh”, ”/root/start.sh”]

ENTRYPOINT /root/start.sh
```

In this demo DockerFile,

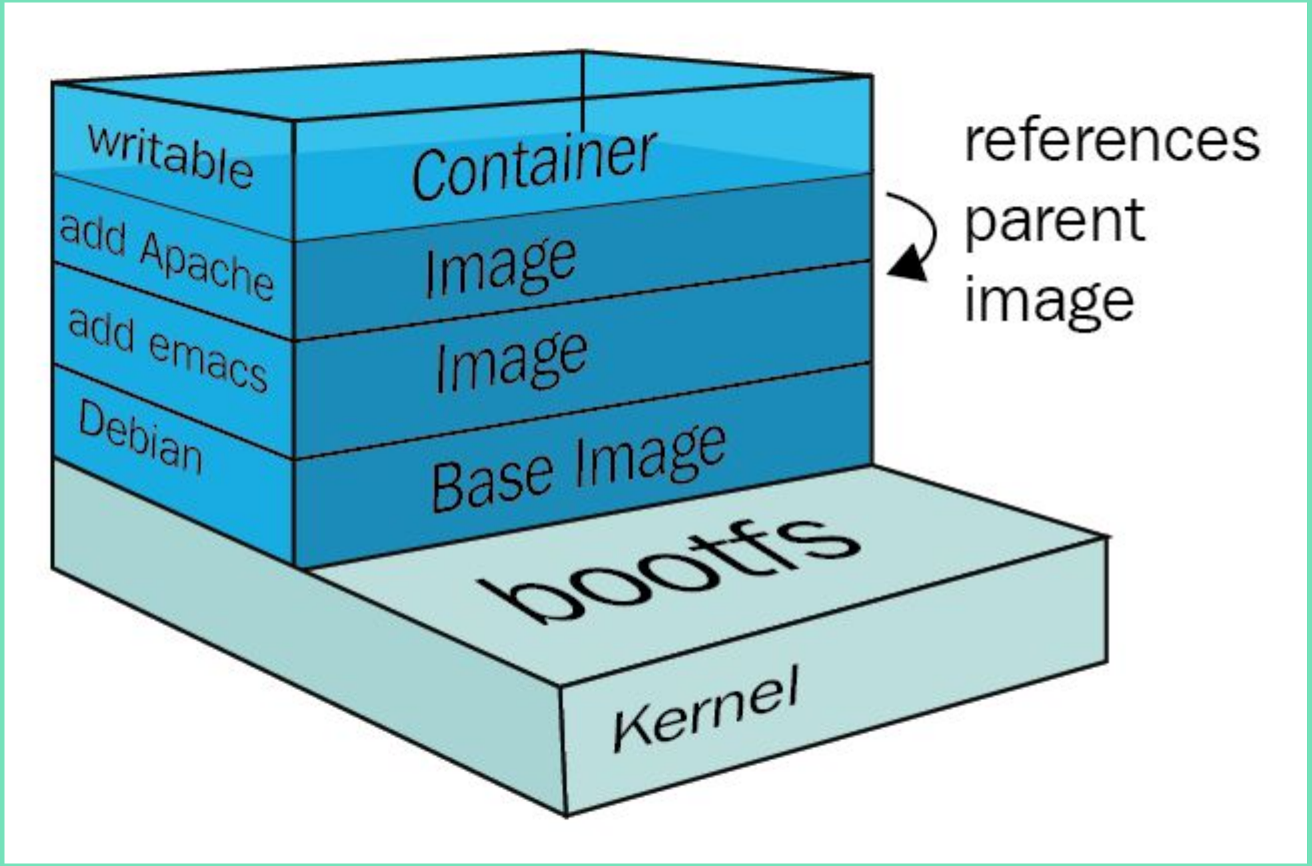
- The first line “#using the official Ubuntu as a base” is a comment. You can add comments to the Docker File with the help of the **#** command
- The next line has to start with the **FROM** keyword. It tells docker, which base image is to be used. In our example, we are creating an image from the **ubuntu** image.
- The **RUN** command is used to run instructions against the image. In our case, we first update our Ubuntu system and then install the Nginx server on our **ubuntu** image.
- The **COPY** command is used to copy a file inside the /var/www/html folder.
- The next command is **EXPOSE**, which is used to expose the port number.
- The **ENTRYPOINT** command will run the **/root/start.sh** when the container starts.

To build an image from the dockerfile, we use the **build** command:

Another Example Dockerfile



```
FROM node:argon

# Create app directory
RUN mkdir -p /usr/src/app
WORKDIR /usr/src/app

# Install app dependencies
COPY package.json /usr/src/app/
RUN npm install

# Bundle app source
COPY . /usr/src/app

EXPOSE 8080
CMD [ "npm", "start" ]
```



Thank You

You can contact me : indrajitsingh37@gmail.com

https://www.linkedin.com/in/indrajitsinghds/