# Docker (An opinionated guide)

Alexander Morton

Financial Cloud

*alex.morton@financial-cloud.com*

July 26, 2018

# Overview

# What's the problem?

*My application needs this file/binary/environment/application to run. Shouldn't take you long to install...*

- Things to consider about the application itself
  - What version do you need?
  - Does it run in a particular version of a certain OS?
  - What does it need access to?
  - What access rights does it need?
  - Does all of the above affect other components of your applications?
  - Does this need to communicate with the outside world?
- Things to consider about production and development
  - **You will usually need to have some development version**
  - Will you have to consider your cloud provider's requirements in addition to your own?
  - Do you need to reconfigure the full application to safely test it?

# What's the solution?

**There is never just one solution**

- Virtual machines have there place if you need a single OS which is completely isolated.
- You may also not want to use the docker engine.
- Cloud provider allow you to more easily setup virtual machines. *The Amazon Machine Image (AMI) would be an example*

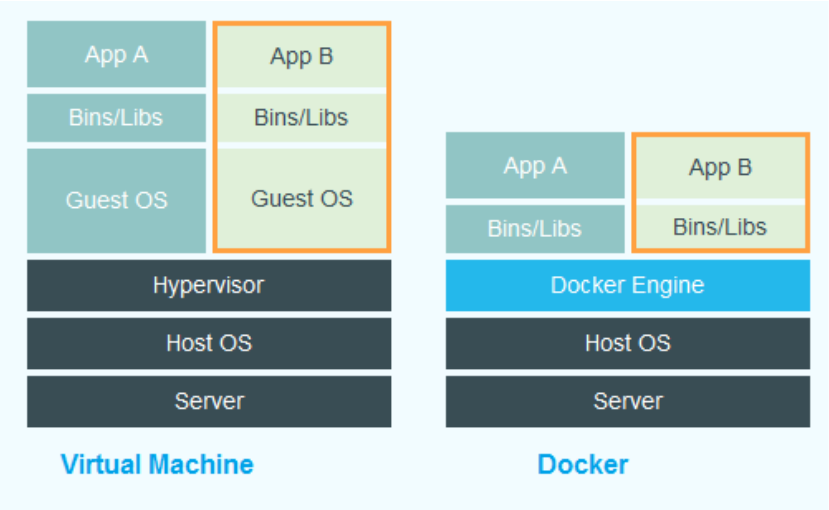



Figure: You have two options to this problem

# Docker solution

"Docker is a computer program that performs operating-system-level virtualization also known as containerization." [Docker Wiki, 2018]

## Basically we use features of the linux kernel

**[Docker Overview, 2016]**

- **Namespaces**: Processes only see resources associated to there namespace.
- **CGroups**: Processes use of resources are isolated into groups.

## What this allows you to do

- Build your application state in layers. If you change a layer then all previous layers taken directly from cache. **Faster builds**
- Only build your application with the stuff it needs. **More efficient when running** **[Container Journal, 2016]**

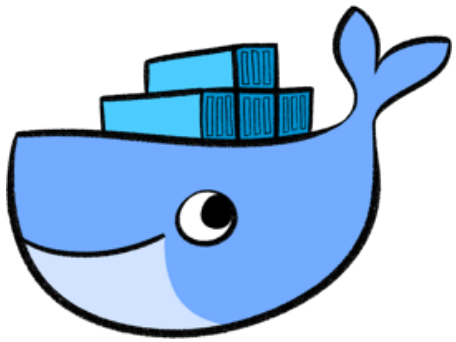


Figure: Docker is nothing more than a bunch of containers on top of a whale [Whale, 2018]

"Docker is the company driving the container movement and the only container platform provider to address every application across the hybrid cloud." [Docker, 2018]

# Docker File Commands

Creating an image can be done with a handful of commands.

[Dockerfile Instructions and Syntax, 2015]

## FROM

This instruction is used to set the base image for subsequent instructions. It is mandatory to set this in the first line of a Dockerfile. You can use it any number of times though.

## MAINTAINER

This is a non-executable instruction used to indicate the author of the Dockerfile.

## RUN

This instruction lets you execute a command on top of an existing layer and create a new layer with the results of command execution.

## CMD

The major difference between CMD and RUN is that CMD doesnt execute anything during the build time. It just specifies the intended command for the image. Whereas RUN actually executes the command during build time.

## EXPOSE

While running your service in the container you may want your container to listen on specified ports.

## COPY/ADD

This instruction is used to copy files and directories from a specified source to a destination (in the file system of the container). ADD is the same but allows remote urls.

## ENTRYPOINT

You can use this instruction to set the primary command for the image.

## VOLUME

You can use the VOLUME instruction to enable access to a location on the host system from a container. Just pass the path of the location to be accessed.

## USER

This is used to set the UID (or username) to use when running the image.

## WORKDIR

This is used to set the currently active directory for other instructions such as RUN, CMD, ENTRYPOINT, COPY and ADD.

## ONBUILD

This instruction adds a trigger instruction to be executed when the image is used as the base for some other image. It behaves as if a RUN instruction is inserted immediately after the FROM instruction of the downstream Dockerfile.

# Example docker file

## Example (Base Image for all ours apps in production)

```
FROM node:carbon

# Set working directory for container
ENV HOME=/usr/app/nodeApp
WORKDIR $HOME

# Need to install aws-cli so the container can get the environment file on startup.
RUN apt-get update
RUN apt-get install -qq -y python-pip libpython-dev
RUN curl -O https://bootstrap.pypa.io/get-pip.py
RUN python get-pip.py
RUN pip install awscli

# Install vim for development
RUN apt-get install vim -y

RUN mkdir -p $HOME
RUN mkdir -p $HOME/logs

# Give everyone including node user the ability to execute/read/write to app directory.
# Need to write to app so we can run startup script.
RUN chmod -R o+rwx /usr/app/nodeApp

COPY ./.aws $HOME/.aws
RUN aws s3 cp s3://npmrc/.npmrc ./.npmrc

RUN npm i -g pm2
COPY ./setup.sh $HOME
```

# Example docker file

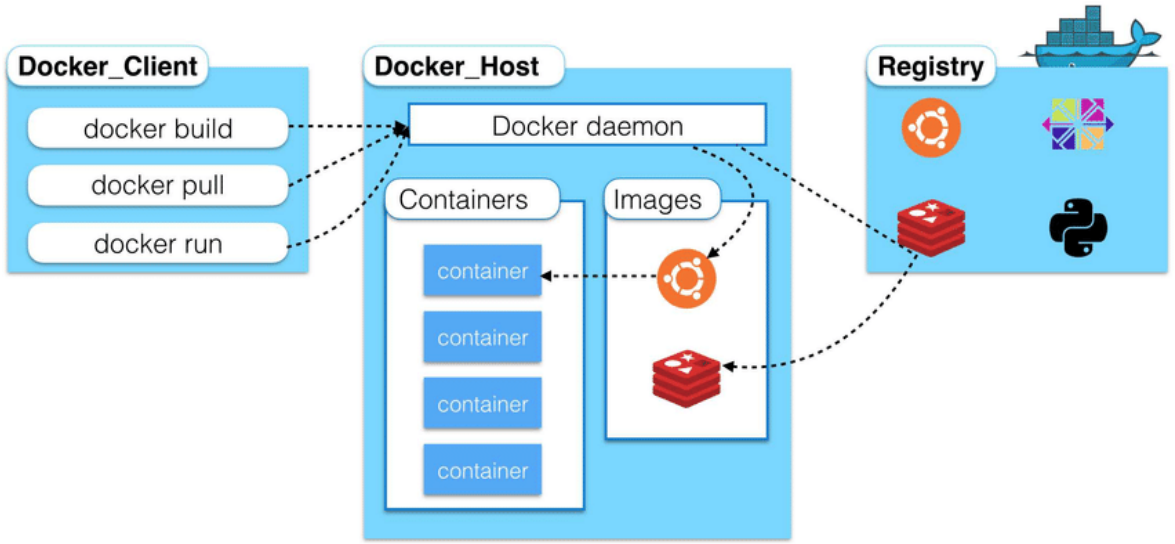
## Example (We expand on that base image for each app)

```
FROM 0000000000.dkr.ecr.eu-west-3.amazonaws.com/production-env:latest

COPY ./package.json $HOME/
COPY ./dist $HOME/dist/
COPY ./UI/dist $HOME/UI/dist/

ENV NODE_ENV=production
RUN npm install

ENV JSON=rules-engine.json

# Setup will get env file and set VPC dependent env variables.
CMD ["bash","/usr/app/nodeApp/setup.sh"]
```

# Docker CLI

**Now we have a docker file what do we do next?**

- Install docker
  - Easy for mac and linux. Windows is another story
- Pull in other peoples images to base your project on
- Build your own image from the base image
- Run the image
  - A running image is the container.
  - You can then access most containers with a bash shell.



# Conclusion

- Lots of videos/blogs on the subject
- Docker is a useful tool for the novice and the tech genius
- Containerization underpins many leading edge technologies
  - AWS Fargate and serverless lambdas are two example we use here
- Docker Rivals
  - Don't confuse container orchestration and container engine
  - If you interested in docker alternatives rkt is one which works with kubernetes.
  - Don't forget VM still have a place.
- Even if you don't use containers in production they can make development a lot easier
- We haven't even touched on container orchestration and networking

# References


Docker (2018)

What is Docker

https://www.docker.com/what-docker

Docker Wiki (2018)

Docker Wiki

https://en.wikipedia.org/wiki/Docker_(software)

Container Journal (2018)

Docker: Not Faster than VMs, but More Efficient

https://containerjournal.com/2016/11/21/docker-not-faster-vms-just-efficient/

Docker Overview (2016)

What Is Docker? How Does It Work?

https://devopscube.com/what-is-docker/

Dockerfile Instructions and Syntax (2015)

Dockerfile Instructions and Syntax

https://deis.com/blog/2015/dockerfile-instructions-syntax/

# The End