INTRODUCTION TO DOCKER

QUICK INVENTORY

- Who has never heared of Docker?
- Who has used or is using Docker?

GOALS

- Learn the basics about Docker
  - Images & containers
  - Data storage
  - Application stacks
  - Docker layers
- Use cases

If there are questions please do not hesitate to ask them!!

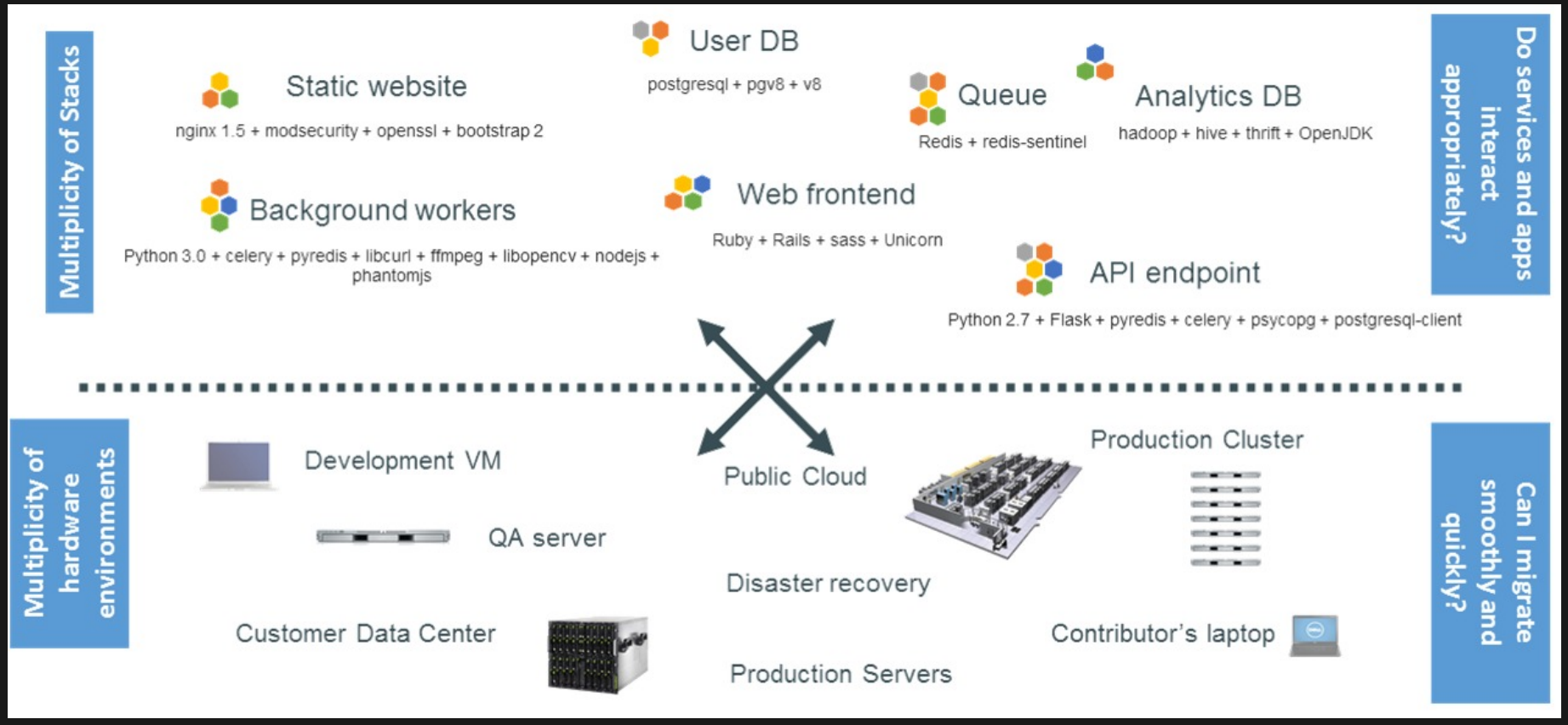
WHAT IS DOCKER?

A software **containerisation** platform

For packaging and running applications!

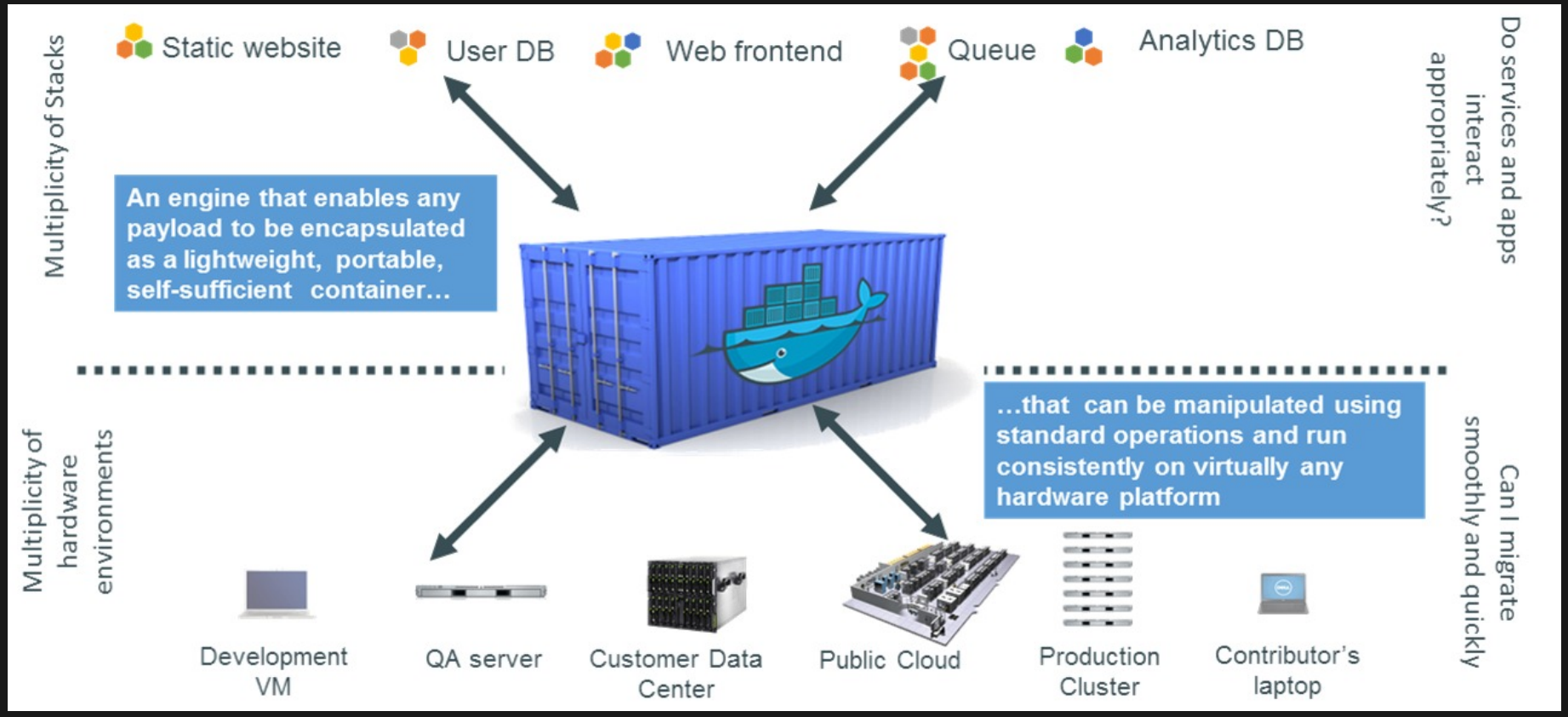
WHAT IS DOCKER TRYING TO SOLVE?

The famous “it works on my machine” headache!



DOCKER SOLUTION

Create reusable platform independent packages



DOCKER VOCABULARY

- **Docker image**
  - Blueprint of an application

DOCKER VOCABULARY

- **Docker image**
  - Blueprint of an application
- **Docker container**
  - Instance of a started image

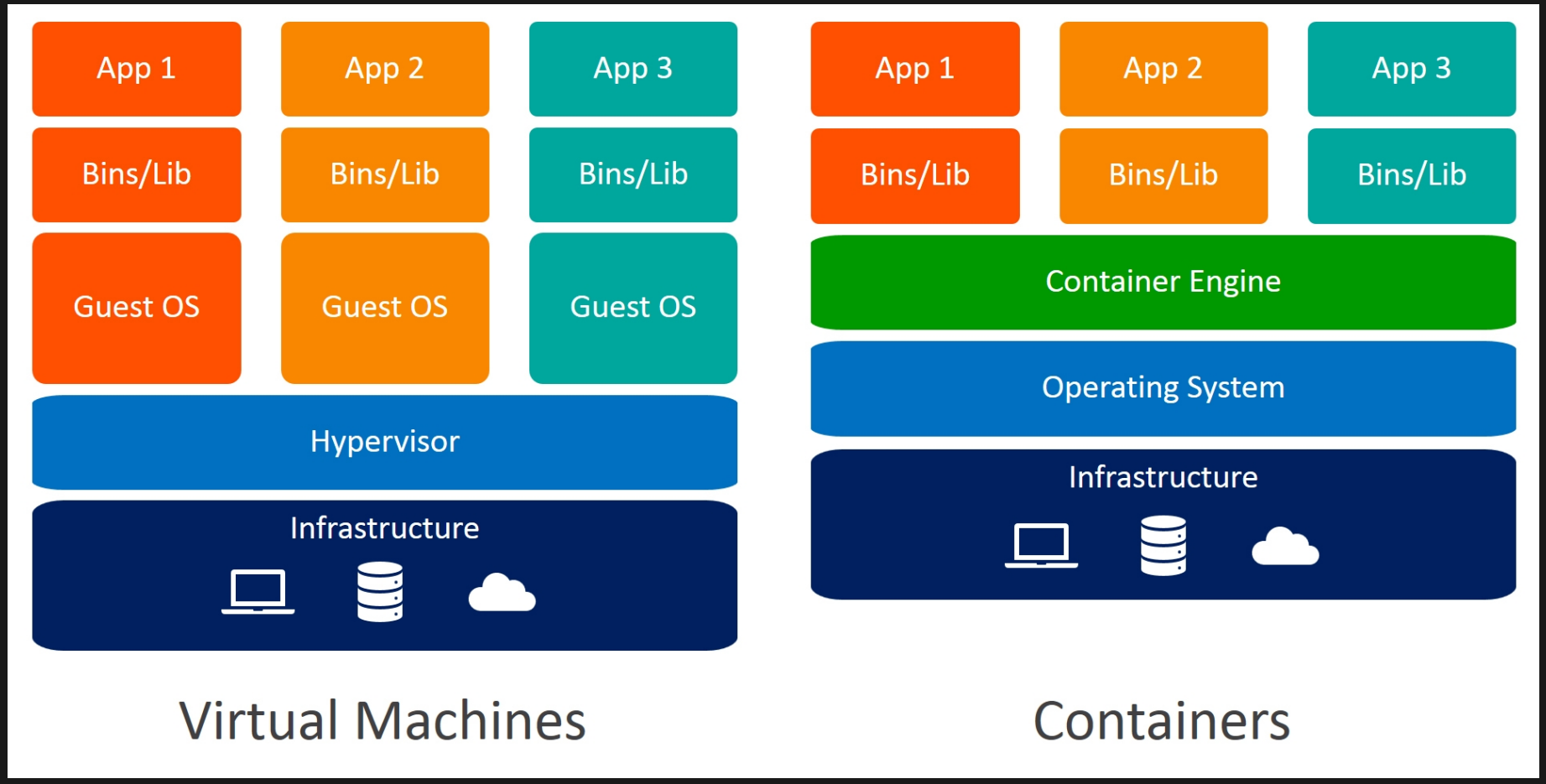
DOCKER VOCABULARY

- **Docker image**
  - Blueprint of an application
- **Docker container**
  - Instance of a started image
- **Docker engine**
  - Creates, ships and runs Docker containers

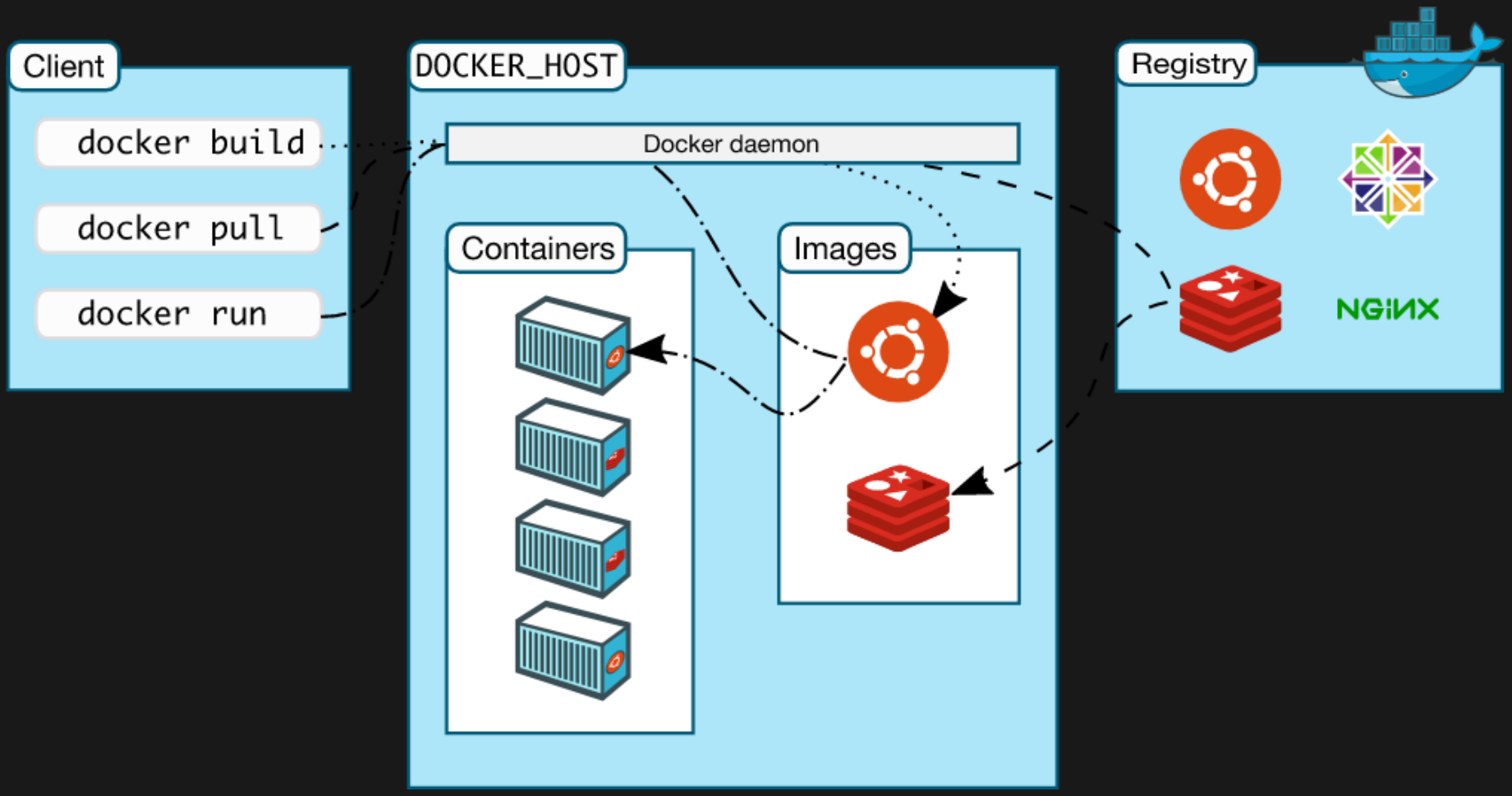
DOCKER VOCABULARY

- **Docker image**
  - Blueprint of an application
- **Docker container**
  - Instance of a started image
- **Docker engine**
  - Creates, ships and runs Docker containers
- **Docker registry**
  - Storage for images

VIRTUAL MACHINE VS CONTAINER



DOCKER ARCHITECTURE



DOCKER USE CASES

1. Application isolation
2. Developer productivity
3. Rapid deployment
4. Simplifying configuration
5. Code pipeline management

DOCKER IMAGES AND CONTAINERS

- Image
  - An immutable template for containers
  - Can be created, pulled and pushed towards public and private registries
    - e.g. Docker Hub

DOCKER IMAGES AND CONTAINERS

- Image
  - An immutable template for containers
  - Can be created, pulled and pushed towards public and private registries
    - e.g. Docker Hub
- Container
  - Instance of an image
  - Can be started, stopped, restarted

REUSE EXISTING IMAGES

```
# search for image 'hello' in registry
$ docker search hello
$ docker search --filter=stars=3 --no-trunc hello
$ docker search --filter=is-official=true --no-trunc hello

# docker pull <registry>:<port>/<image>:<tag>
$ docker pull hello-world \n
$ docker pull ubuntu         'default latest'
$ docker pull ubuntu:latest 'equal to previous'
$ docker pull ubuntu:14.0.4 'with tag'

# list images on local system
$ docker image ls

# run an image and create a container
$ docker run hello-world
$ docker run ubuntu echo "hello world"
```

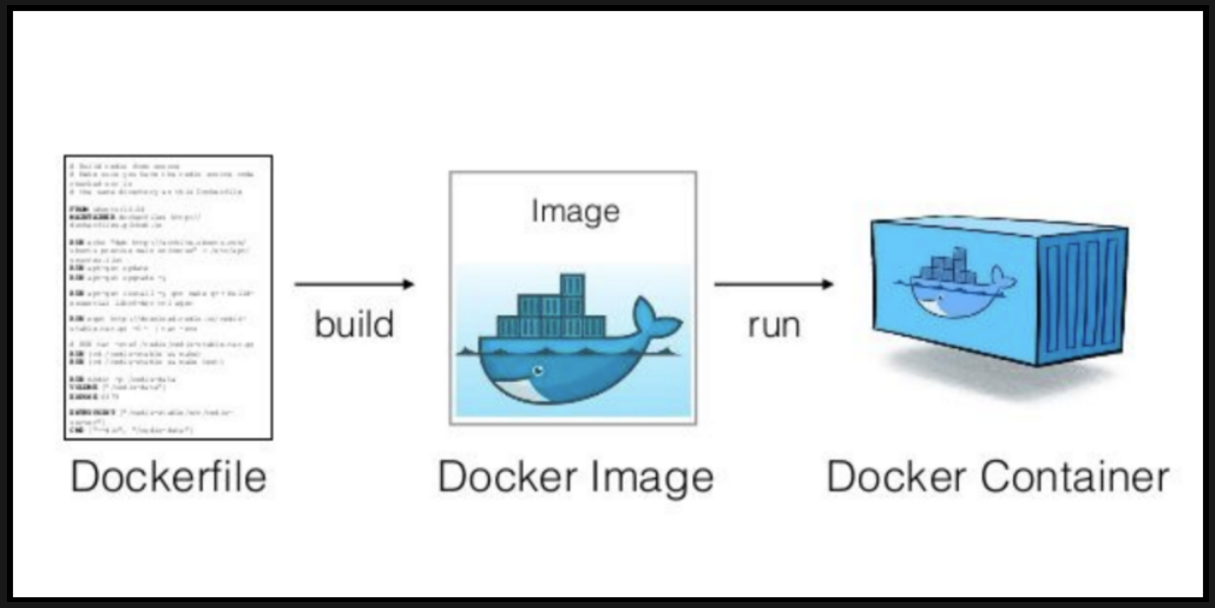
DEMO

- Search for a Docker image (cli / dockerhub)
- Download Docker image
- Run hello-world
- Run ubuntu
- Get some system information

CREATE YOUR OWN IMAGES

Dockerfile

Each Dockerfile is a script (DSL), composed of various commands and arguments to perform actions on a base image in order to create a new one.



DOCKERFILE COMMANDS

Command	Overview
FROM	Specify base image
RUN	Execute specified command
ENTRYPOINT	Specify the command to execute the container
CMD	Specify the command at the time of container execution (can be overwritten)
COPY	Simple copy of files / directories from host machine to container image
ADD	COPY + unzip / download from URL (**not recommended**)
ENV	Add environment variables
EXPOSE	Open designated port
WORKDIR	Change current directory
MAINTAINER	**deprecated**
	now `LABEL maintainer="maintainer@example.com"`should be specified as

DOCKERFILE EXAMPLE 1

```
# file: Dockerfile
# select a base image
FROM tomcat:10.0.0

# add a file to the image
COPY index.html /usr/local/tomcat/webapps/ROOT/index.html
```

```
# build the image
$ docker build --file Dockerfile --tag tomcat:test .
# equal to (when in the Dockerfile folder!):
# -> docker build -t tomcat-test .
```

```
# run container and publish ports
$ docker run --publish 8080:8080 tomcat:test
# equal to:
# -> docker run -p 8080:8080 tomcat:test
```

DOCKERFILE EXAMPLE 2

```
# select a base image
FROM azul/zulu-openjdk:11

# add a file to the image
ADD service.jar

# expose a port
EXPOSE 8080

# run the application
CMD ["java","-jar","/service.jar"]
```

```
# build the image
$ docker build --file Dockerfile --tag my-app:0.1 .
```

```
# run container and publish ports
$ docker run --publish 80:8080 my-app:0.1
```

```
# ship/push the image
$ docker push my-app:0.1
```

DEMO

- Build docker image
- Run docker image (results in a container)
- Push the docker image

INFORMATION & CLEANUP

```
# show running containers
$ docker ps

# show running/stopped containers
$ docker ps -a

# cleanup
$ docker system prune -a
$ docker container prune
```

DOCKER DATA STORAGE

- In-container (bad idea!)
- Bind mounts (mount to local file system)
- Volumes (data is managed by docker)

```
# start interactive docker container, mount local folder
$ docker run -it --rm -v $(pwd):/my-data openjdk:11-jdk bash

# create volume
$ docker volume create my-volume

# list volumes
$ docker volume ls

# start docker container and use docker volume
$ docker run -it --rm -v my-volume:/my-data alpine sh
```

DEMO

- Use local filesystem mount
- Use Docker volumes

DOCKER COMPOSE

Managing multiple containers via the commandline is cumbersome! So we we need a way to easily handle multiple containers at once.

WITHOUT / WITH COMPOSE

- Without compose
  - build and run one container at a time
  - manually connect containers
- With compose
  - define multi container stack in a file
  - single command to start entire stack
  - handles container dependencies

3 STEPS OF COMPOSE

1. Define the services that make your application stack in a Docker compose file
2. (optional) Define your application environment in a Dockerfile (compose build!)
3. Run the CLI:

```
$ docker-compose up
```

COMPOSE EXAMPLE

```
version: '3.9'

services:
  db:
    image: mysql:5.7
    restart: always
    # some code omitted!!
    volumes:
      - db_data:/var/lib/mysql
  wordpress:
    image: wordpress:latest
    ports:
      - "8000:80"
    restart: always
    environment:
      # some code omitted!!
      WORDPRESS_DB_HOST: db:3306
    depends_on:
      - db

volumes:
  db_data:
```

COMPOSE COMMANDS

```
$ docker-compose up
$ docker-compose up -d
$ docker-compose down
$ docker-compose down --volumes
```

DEMO

- Start full stack using Docker Compose
  - Wordpress
  - MySql with admin tooling
  - Using Compose to build

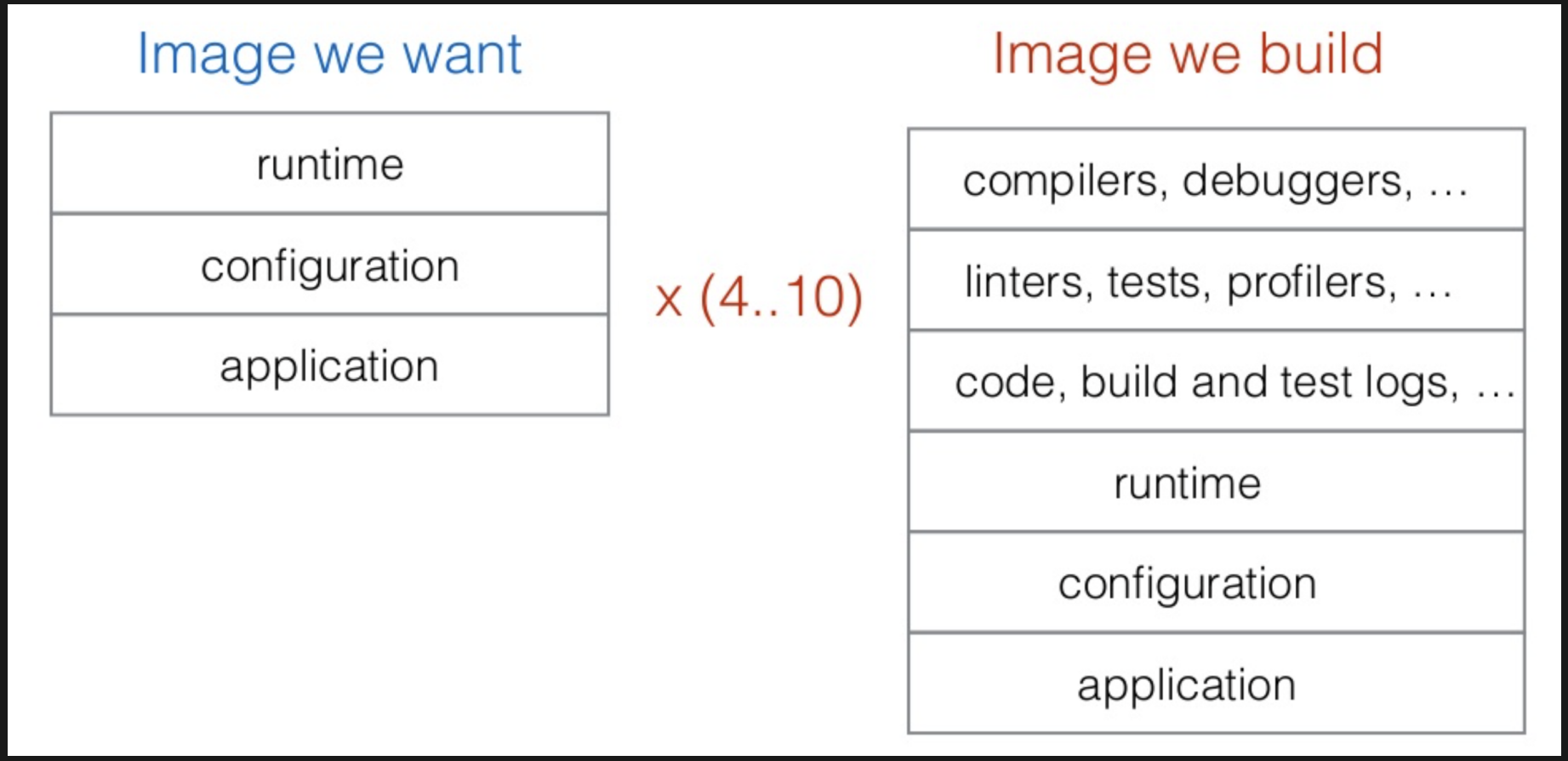
DOCKER LAYERS

- Each `build` command generates a layer
- A complete `build` creates a SINGLE image

Dockerfile

```
FROM ubuntu:15.04
COPY . /app
RUN make /app
CMD python /app/app.py
```

PROBLEMS WITH DOCKER BUILD



MULTISTAGE BUILD

- Build the application
- Perform static code analysis
- Run unit tests
- Run integration tests
- Package application

DO WE HAVE A PROBLEM?

```
FROM openjdk:11-jdk
WORKDIR /opt/app/src
# Copies in our code and compiles
COPY /src/*.java ./
RUN javac *.java

# Perform static code analysis
RUN apt-get install sonar-scanner
RUN sonar-scanner -Dsonar.host.url=http://host.docker.internal
RUN apt-get remove sonar-scanner

# Run unit tests
# Run integration tests

# Create application
RUN rm -rf /src/*.java
CMD java HelloWorld
```

DOCKER BUILDER PATTERN

- 2 Dockerfiles
  - 1st for build tools
  - 2nd for runtime
- Drawbacks
  - 2 Dockerfiles
  - Orchestration needed

DOCKER MULTISTAGE BUILDS

- Single Dockerfile with multiple stages
- Same build!! Local and CI

```
# Copies in our code and compiles
FROM openjdk:11-jdk as builder
WORKDIR /opt/app/src
COPY /src/*.java ./
RUN javac *.java

# Perform static code analysis
FROM sonarsource/sonar-scanner-cli as sonarqube
WORKDIR /opt/app/src
COPY --from=builder /opt/app/src ./
RUN sonar-scanner -Dsonar.host.url=http://host.docker.internal

# Run unit tests
# Run integration tests

# Create application
FROM openjdk:11-jdk as application
WORKDIR /opt/app/src
COPY --from=builder /opt/app/src/*.class ./
CMD java HelloWorld
```

DEMO

- Perform multistage build
  - Compile code
  - Static code analysis
  - Create image

AS A DEVELOPER

- I need to build my software
  - Use a container
- I need a database or something else…
  - dont install local but use a compose file!
- I need to run my software
  - build using docker and run using docker

MORE INFORMATION

- Docker Curriculum
- Docker
- Docker Hub
- Get started with Docker Compose
- Docker Compose File Format
- Docker Awesome

QUESTIONS?

Created by Marco Pas.