

# IBM Cloud Container Workshop

## *Part 1 - Containers*



# The Challenge

## Multiplicity of stacks



### Static website

nginx 1.15.6 + modsecurity + openssl + bootstrap 4



### User DB

postgresql + pgv8 + v8



### Queue

Redis + redis-sentinel



### Analytics DB

hadoop + hive + thrift + OpenJDK



### Background workers

Python 3.0 + celery + pyredis + libcurl + ffmpeg + libopencv + nodejs + phantomjs



### Web frontend

Ruby + Rails + sass + Unicorn

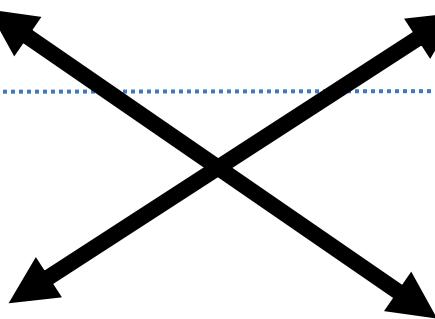


### API endpoint

Python 3.5 + Flask + pyredis + celery + psycopg + postgresql-client

## Multiplicity of HW environments

### Development VM



### Production Servers



### Production Cluster

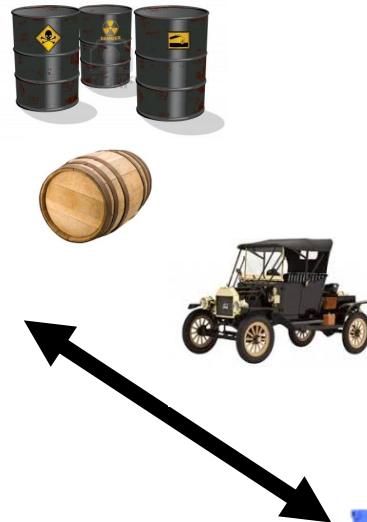


### Customer Data Center

# The Solution: Intermodal Shipping Container



Multiplicity of Goods



Multiplicity of Methods for sorting



# Container System for code

## Multiplicity of stacks



### Static website

nginx 1.15.6 + modsecurity + openssl + bootstrap 4



### User DB

postgresql + pgv8 + v8



### Queue

Redis + redis-sentinel

### Analytics DB

hadoop + hive + thrift + OpenJDK



### Background workers

Python 3.0 + celery + pyredis + libcurl + ffmpeg + libopencv + nodejs + phantomjs



### Web frontend

Ruby + Rails + sass + Unicorn



### API endpoint

Python 3.5 + Flask + pyredis + celery + psycopg + postgresql-client

## Multiplicity of HW environments

### Development VM



### Production Servers



### Customer Data Center



### Public Cloud



### Production Cluster



### Contributor's laptop

# Why it works? Separation of concerns



**Deb**  
The Developer

Worries about what's  
« **inside** » the container

- Her code
- Her libraries
- Her Package Manager
- Her Apps
- Her Data

To her, all Linux servers look  
the same



**Mike**  
The Ops Guy

Worries about what's  
« **outside** » the container

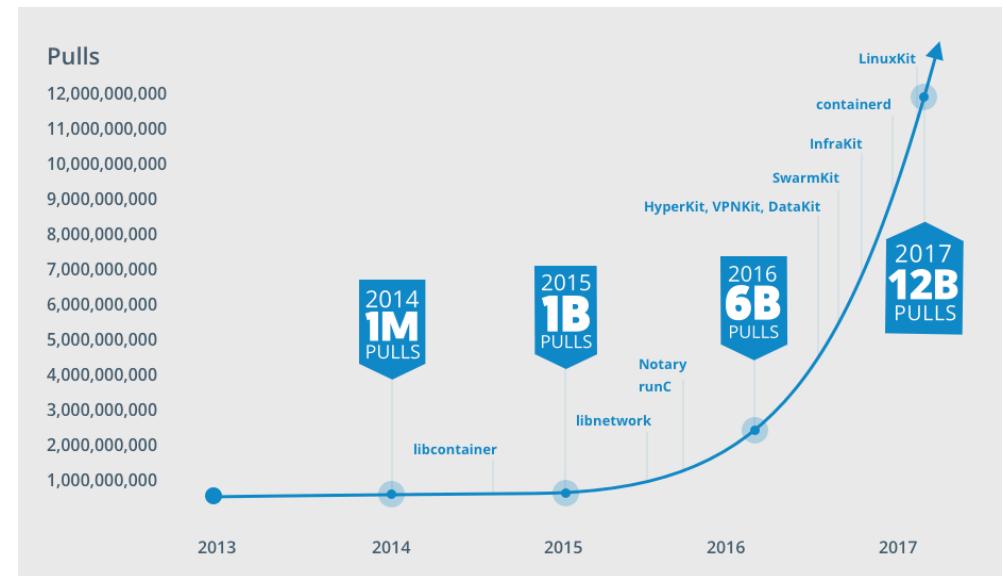
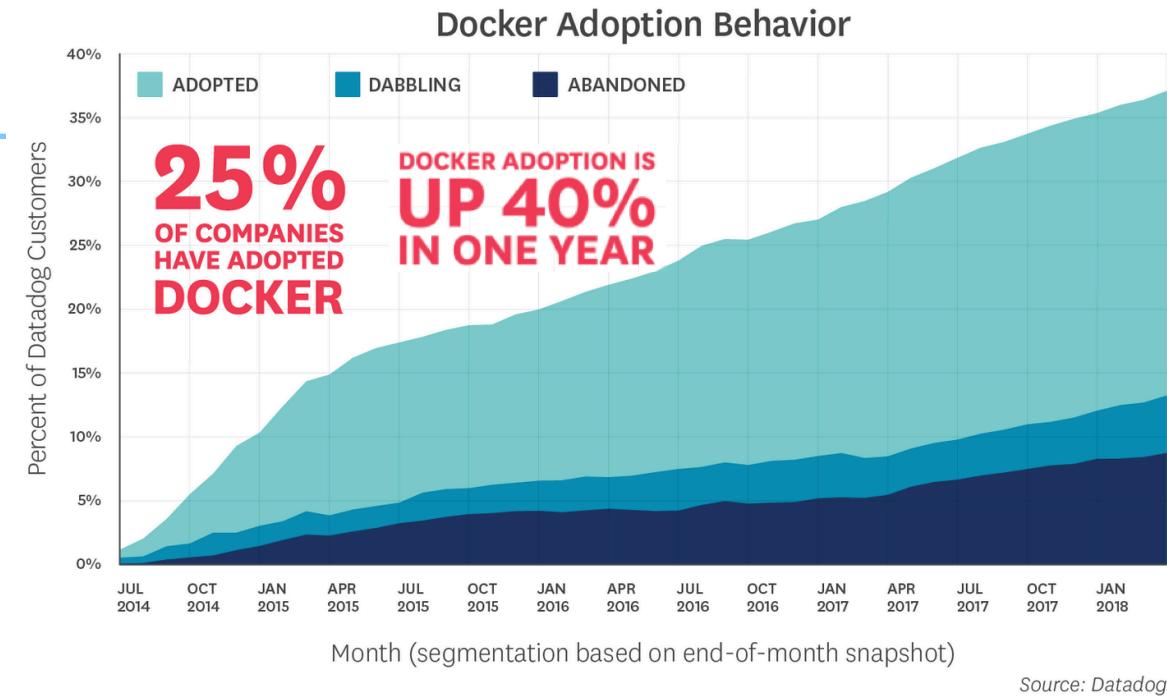
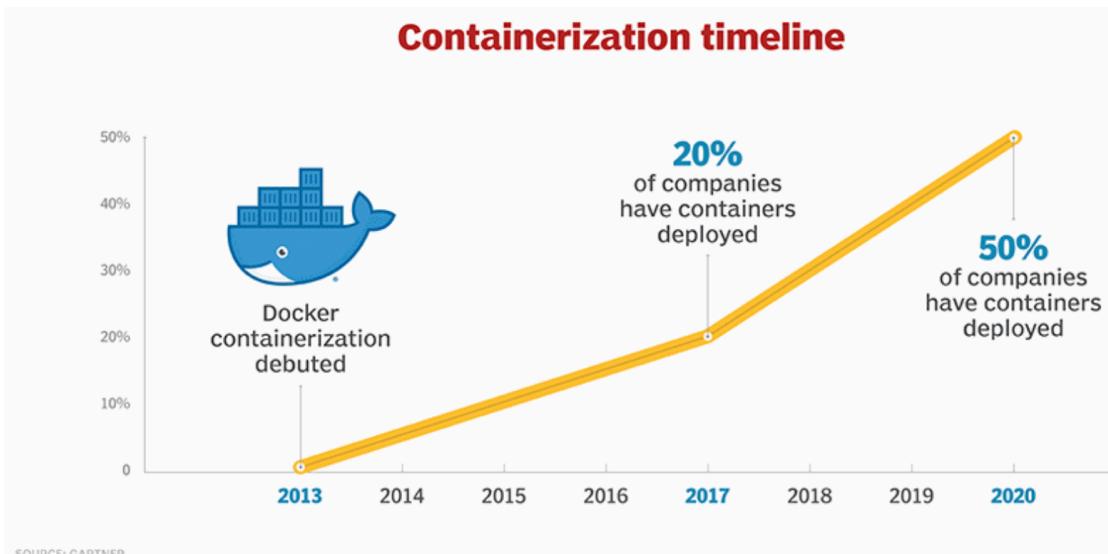
- Logging
- Remote access
- Monitoring
- Network config



All containers start, stop, copy,  
attach, migrate ... the exact  
same way

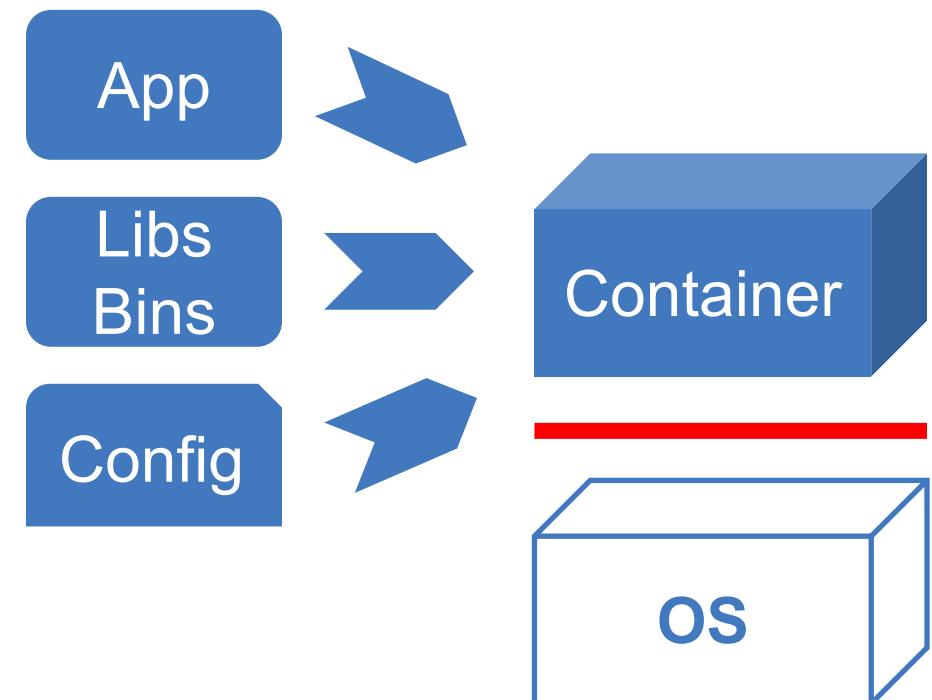
# Market Dynamics and Use Cases

- Container Adoption Drivers
  - Microservice Patterns
  - Cloud Native Applications
  - Hybrid Cloud
  - CD/CI in DevOps
  - Modernizing Applications
- All industries are impacted



# Containers

- A standard way to **package** an application and all its dependencies so that it can be moved between environments and **run** without changes.
- Containers work by **isolating** the differences between applications **inside** the container so that everything **outside** the container can be standardized.

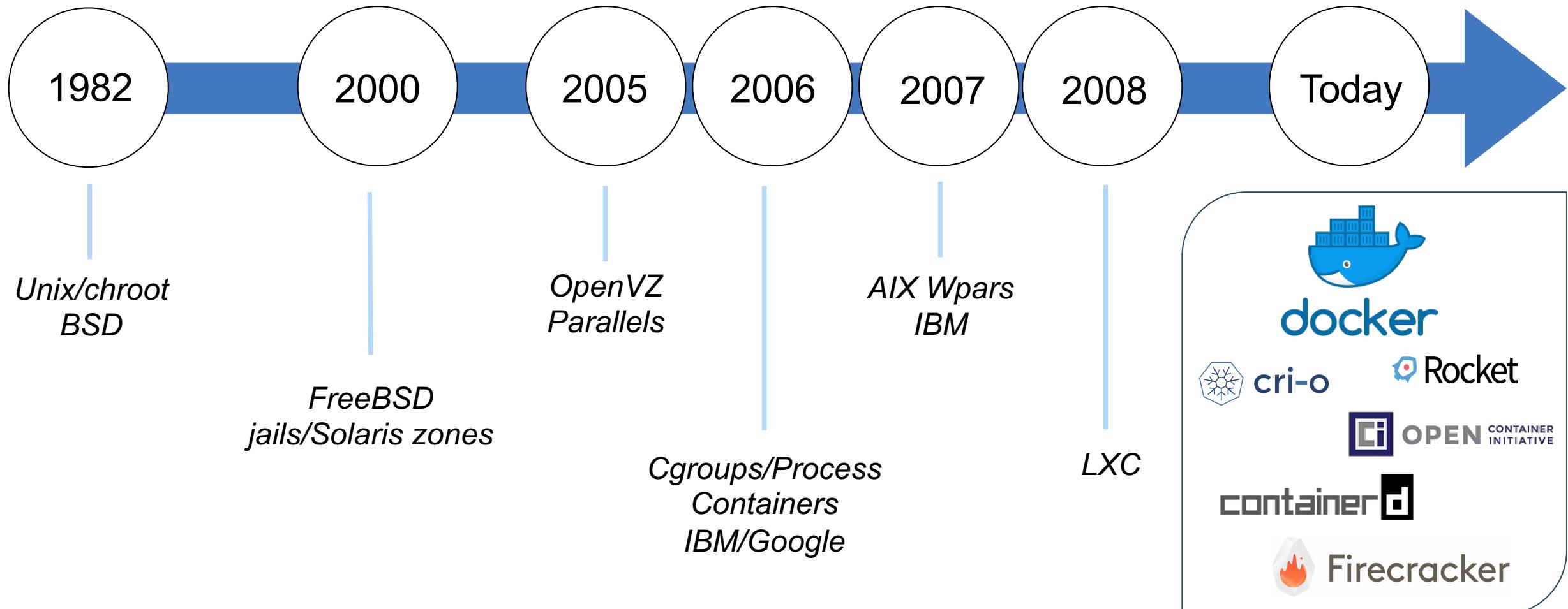


# Why Customers are interested in Containers

---

- #1 : Application Portability
  - Isolated containers package the application, dependencies and configurations together. These containers can then seamlessly move across environments and infrastructures.
- #2 : Ship More Software
  - Accelerate development & deployment, CI and CD pipelines by eliminating headaches of setting up environments and dealing with differences between environments. On average, Docker users ship software 7X more frequently<sup>1</sup>.
- #3 : Resource Efficiency
  - Lightweight containers run on a single machine and share the same OS kernel while images are layered file systems sharing common files to make efficient use of RAM and disk and start instantly.

# A Brief Container History



- A Linux Foundation Collaborative Project
- Free from specific vendor control / an open ecosystem
- Includes:
  - a runtime specification
  - reference runtime\* (runc)
  - an image format specification
  - an image distribution spec (2019)

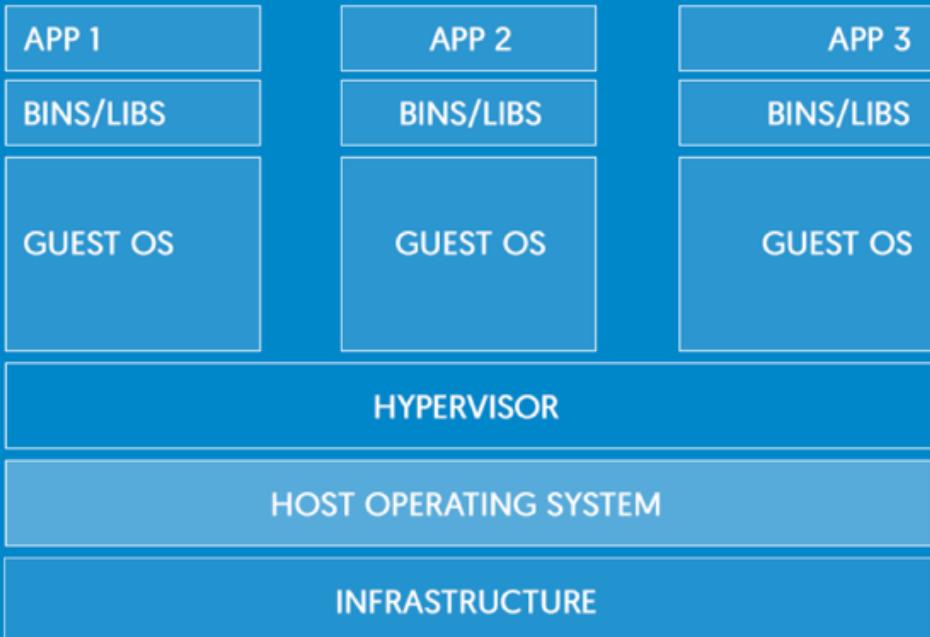
\*seeded with *runc + libcontainer* by Docker

A dark grey rectangular box containing information about the Open Container Initiative. At the top left is the OCI logo. To its right, the word 'OPEN' is displayed in large, bold, grey capital letters, with 'CONTAINER INITIATIVE' in smaller, dark blue capital letters to its right.

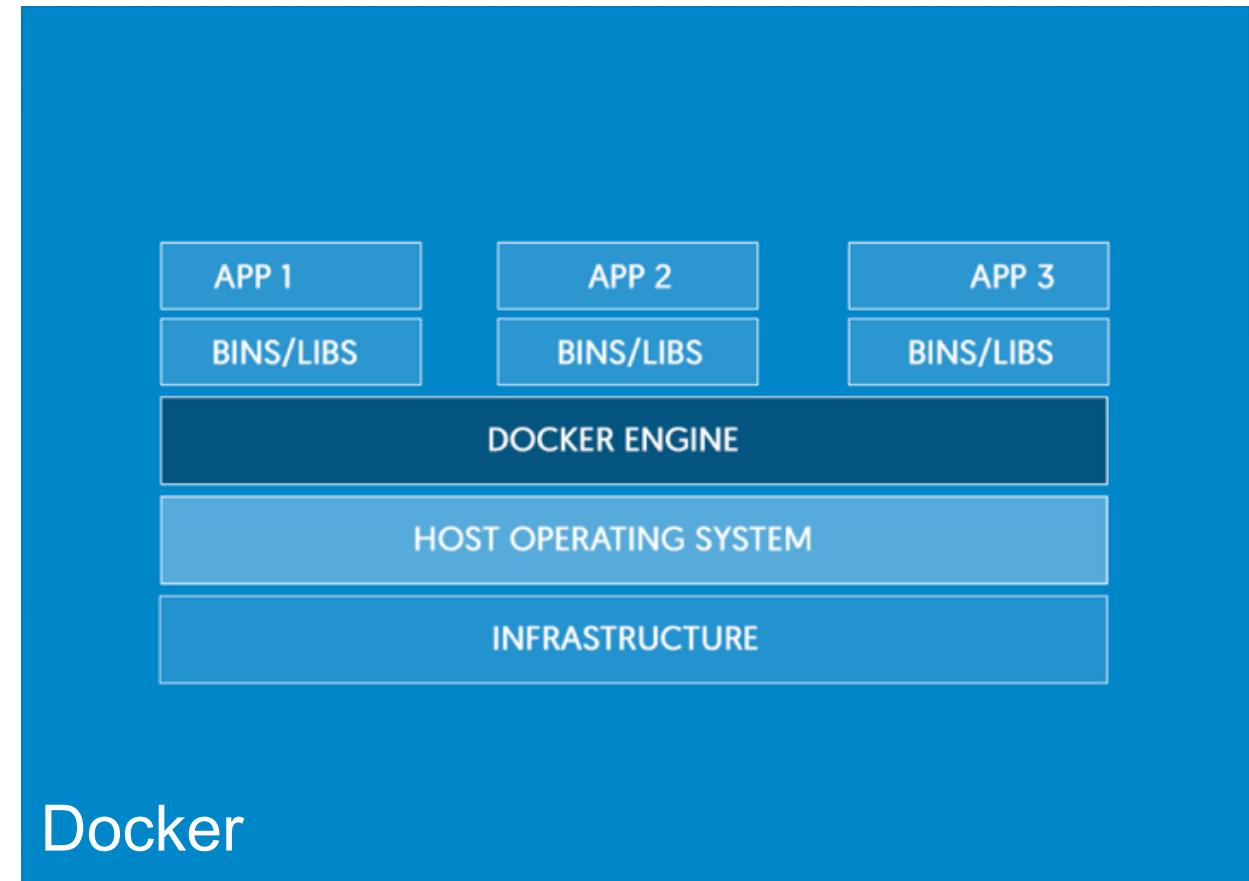
> Announced June 20th, 2015  
> Charter signed on December 8th, 2015  
> 37 member companies  
> Initial specifications reached 1.0 in June 2017

<https://opencontainers.org>  
<https://github.com/opencontainers>

# VMs, Containers and Docker



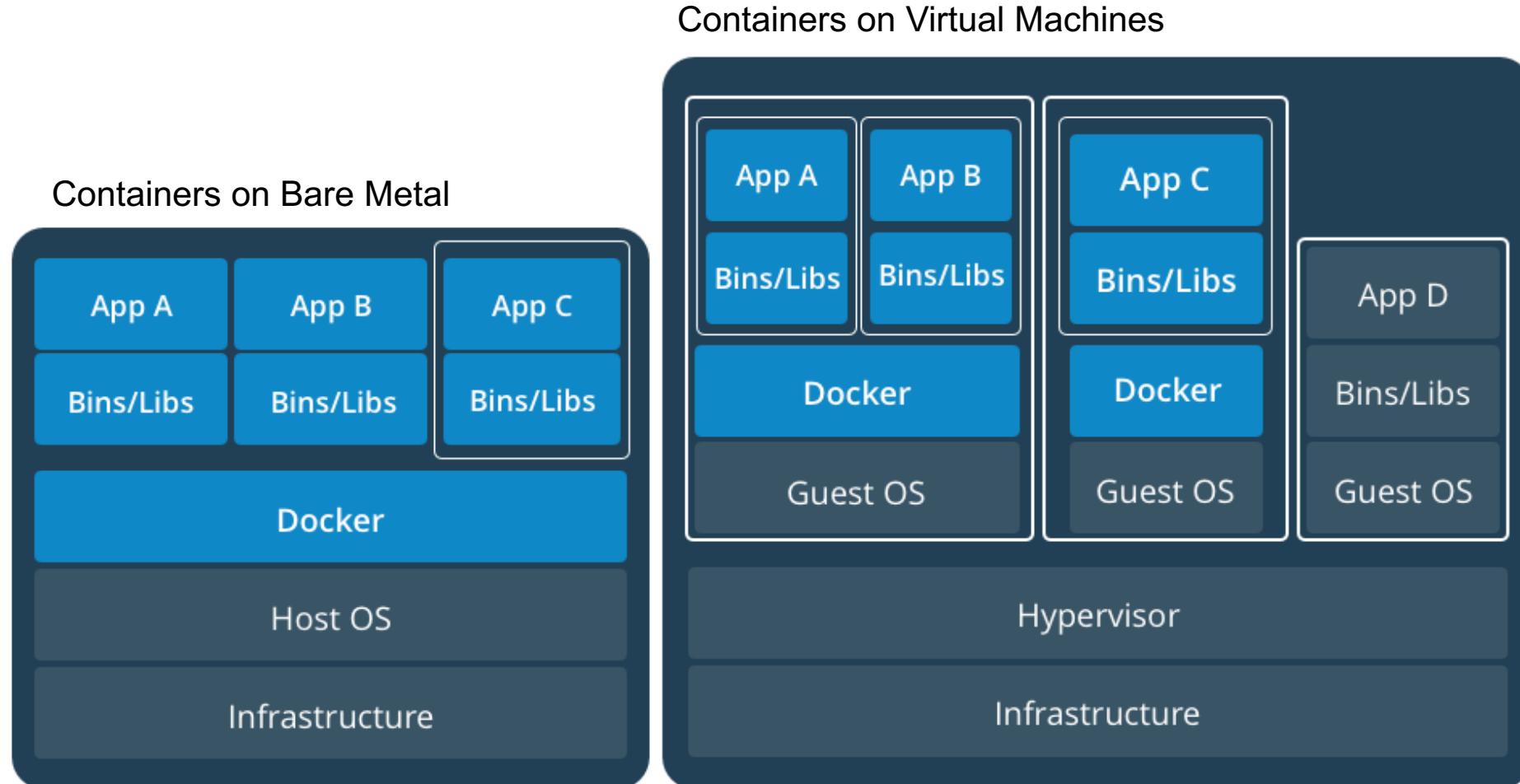
VM



Docker

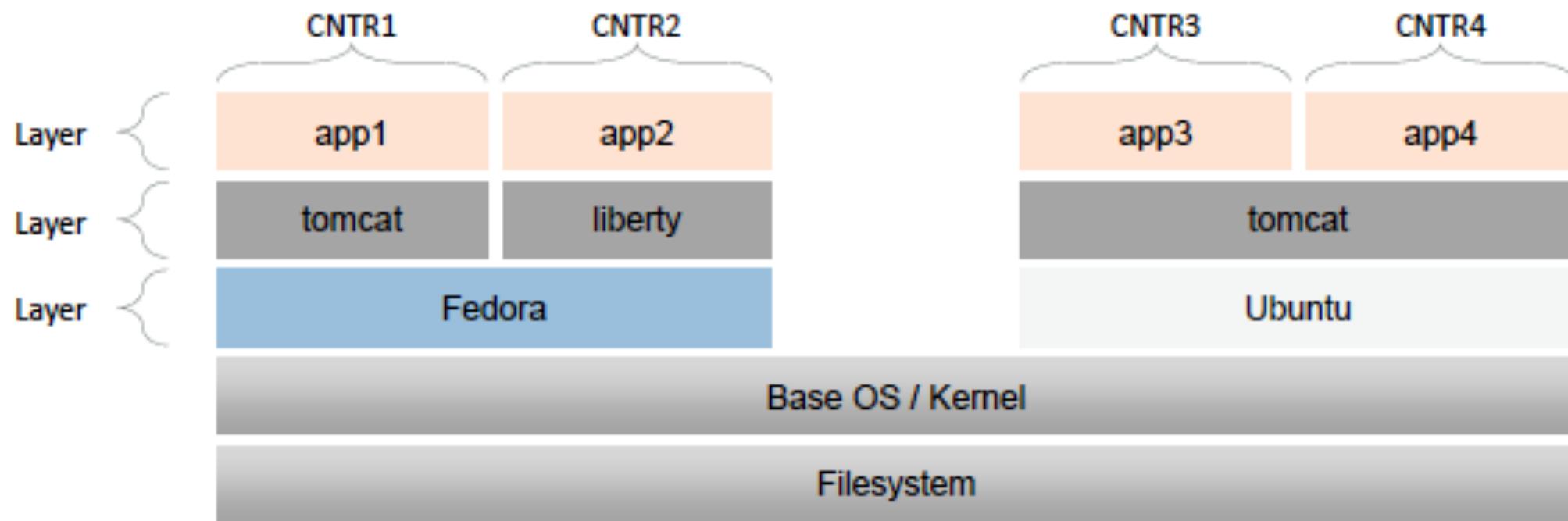
Docker = Linux namespaces + cgroups + overlay (union) file system + image format

# Containers and VMs Together



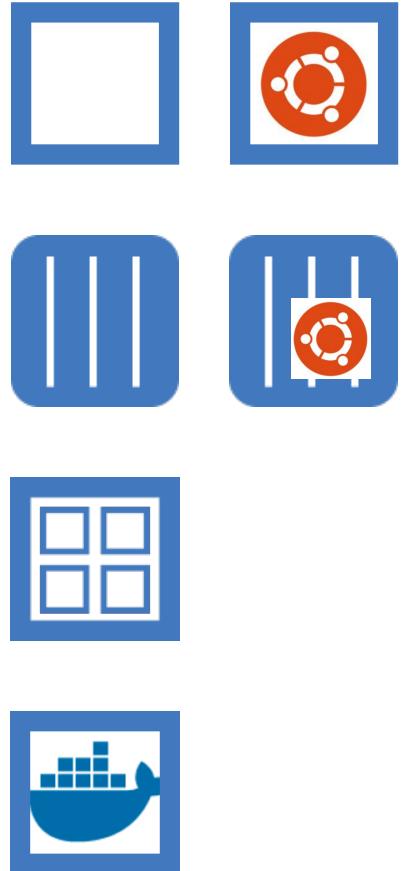
# Docker Containers

- Docker uses a copy-on-write (union) filesystem
- New files (& edits) are only visible to current/above layers (used for reuse)



# Docker Terminology

- Image
  - A read-only snapshot of a container stored in a registry to be used as a template for building containers. At rest.
- Container
  - The image when it is ‘running.’ The standard unit for app service
- Registry
  - Stores, distributes and manages Docker images
- Engine
  - The software that executes commands for containers. Networking and volumes are part of Engine. Can be clustered together.
- Control Plane
  - Management plane for container and cluster orchestration



# Docker Commands (CLI)

```
phil:[~]: docker version
```

**Client:**

**Version:** 18.03.0-ce

**API version:** 1.37

**Go version:** go1.9.4

**Git commit:** 0520e24

**Built:** Wed Mar 21 23:06:22 2018

**OS/Arch:** darwin/amd64

**Experimental:** false

**Orchestrator:** swarm

**Server:**

**Engine:**

**Version:** 18.03.0-ce

**API version:** 1.37 (minimum version 1.12)

**Go version:** go1.9.4

**Git commit:** 0520e24

**Built:** Wed Mar 21 23:14:32 2018

**OS/Arch:** linux/amd64

**Experimental:** true

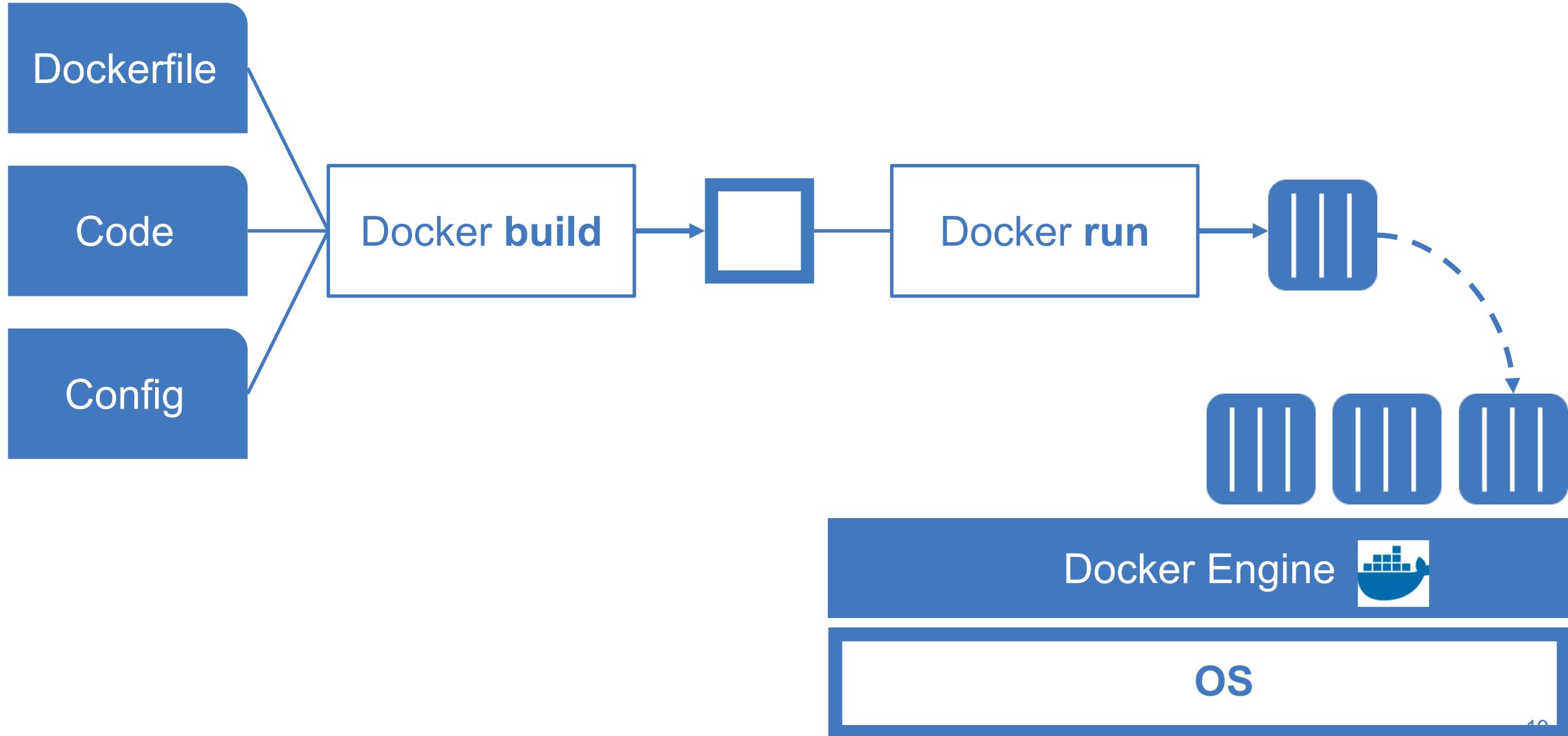
```
[phil:[~]: docker
Usage: docker COMMAND
A self-sufficient runtime for containers
Options:
  --config string      Location of client config files (default "/Users/phil/.docker")
  -D, --debug          Enable debug mode
  -H, --host list      Daemon socket(s) to connect to
  -l, --log-level string
  --tls                Set the logging level ("debug"|"info"|"warn"|"error"|"fatal") (default "info")
  --tlscacert string  Use TLS; implied by --tlsverify
  --tlscert string    Trust certs signed only by this CA (default "/Users/phil/.docker/ca.pem")
  --tlskey string      Path to TLS certificate file (default "/Users/phil/.docker/cert.pem")
  --tlsv1               Path to TLS key file (default "/Users/phil/.docker/key.pem")
  --tlsv1.1             Use TLS and verify the remote
  --tlsv1.2             Print version information and quit
Management Commands:
  checkpoint  Manage checkpoints
  config       Manage Docker configs
  container   Manage containers
  image        Manage images
  network     Manage networks
  node         Manage Swarm nodes
  plugin      Manage plugins
  secret      Manage Docker secrets
  service     Manage services
  swarm        Manage Swarm
  system      Manage Docker
  trust        Manage trust on Docker images
  volume      Manage volumes
Commands:
  attach       Attach local standard input, output, and error streams to a running container
  build        Build an image from a Dockerfile
  commit      Create a new image from a container's changes
  cp           Copy files/folders between a container and the local filesystem
  create      Create a new container
  deploy      Deploy a new stack or update an existing stack
  diff         Inspect changes to files or directories on a container's filesystem
  events      Get real time events from the server
  exec        Run a command in a running container
  export      Export a container's filesystem as a tar archive
  history    Show the history of an image
  images      List images
  import      Import the contents from a tarball to create a filesystem image
  info         Display system-wide information
  inspect    Return low-level information on Docker objects
  kill        Kill one or more running containers
  load        Load an image from a tar archive or STDIN
  login      Log in to a Docker registry
  logout    Log out from a Docker registry
  logs       Fetch the logs of a container
  pause      Pause all processes within one or more containers
  port        List port mappings or a specific mapping for the container
  ps          List containers
  pull        Pull an image or a repository from a registry
  push        Push an image or a repository to a registry
  rename    Rename a container
  restart   Restart one or more containers
  rm         Remove one or more containers
  rmi       Remove one or more images
  run        Run a command in a new container
```

# Most Useful Docker Commands

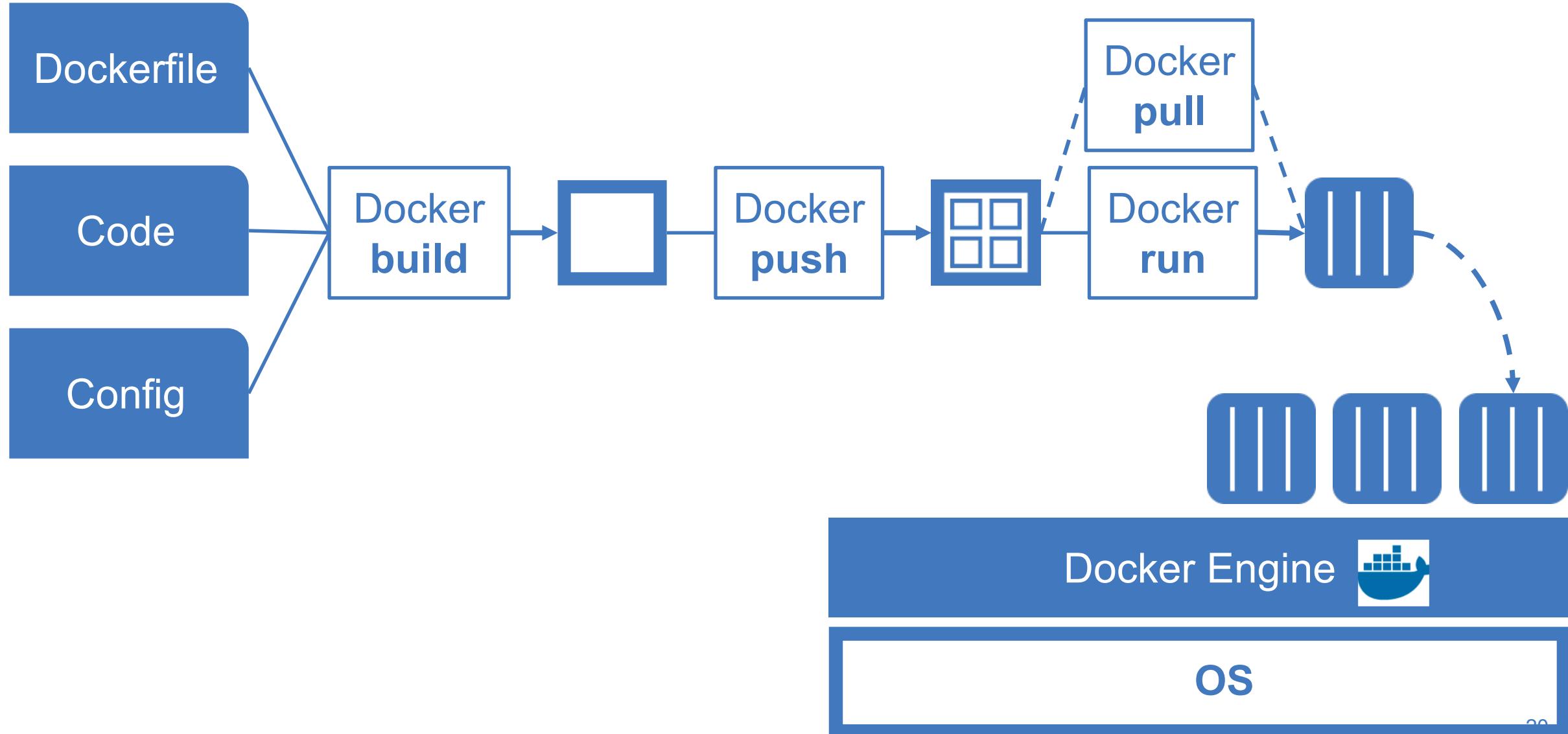
---

- **docker build** used to build the image with the help of the Dockerfile
- **docker push** push the image into a registry
- **docker images** list images in a registry
- **docker run** run the container or a command in a container
- **docker ps** list containers
- **docker kill** kill one or more container
- **docker exec** run a command in a running container
- **docker top** display the running processes in a container
- **docker container** manages container details
- **docker network** manages networks for containers

# Docker Supply Chain (local)



# Docker Supply Chain (with a registry)



# Registries

- Hosting image repositories
  - You can define your own registry
  - A registry is managed by a registry container
- Public and Private registries
  - Public Registry like Docker Hub
  - <https://hub.docker.com>
- Login into the registry
  - Docker login domain:port

The screenshot shows the Docker Hub website interface. At the top, there's a search bar with placeholder text "Search for great content (e.g., mysql)". Below the search bar are navigation links: Explore, Repositories, Organizations, Get Help, and a user account dropdown for manugenard. The main content area has tabs for Docker EE, Docker CE, Containers (which is selected and highlighted in blue), and Plugins. On the left, there are filters for Docker Certified (with an unchecked checkbox) and Images (with checkboxes for Verified Publisher and Official Images). Categories listed include Analytics, Application Frameworks, Application Infrastructure, Application Services, Base Images, Databases, DevOps Tools, and Featured Images. The central part of the page displays a list of available images, starting with "Oracle Database Enterprise Edition" (Docker Certified, By Oracle, Updated 2 years ago), "Oracle Database 12c Enterprise Edition" (Container, Docker Certified, Linux, x86-64, Databases), "Oracle Java 8 SE (Server JRE)" (Docker Certified, By Oracle, Updated 5 days ago), and "MySQL Server Enterprise Edition" (Docker Certified, By Oracle, Updated 6 months ago). Each image entry includes its logo, name, certification status, publisher, and last update date.

# IBM Software on Docker Hub

The image shows two screenshots of the Docker Hub interface. The left screenshot displays a list of repositories under the 'ibmcom' organization, with 112 repositories shown. The right screenshot provides a detailed view of the 'ibmcom/websphere-liberty' repository.

**Left Screenshot: IBM Software Repositories**

Repositories (112)

- ibmcom/mq
- ibmcom/mqlight
- ibmcom/ibmjava
- ibmcom/datapower
- ibmcom/websphere-liberty
- ibmcom/websphere-traditional

**Right Screenshot: ibmcom/websphere-liberty Repository Details**

PUBLIC REPOSITORY  
**ibmcom/websphere-liberty** ☆  
Last pushed: 20 days ago

Repo Info Tags

Short Description  
Official IBM WebSphere Application Server for Developers Liberty image.

Full Description

Supported tags and respective Dockerfile links

- kernel ([ga/developer/kernel/Dockerfile](#))
- common ([ga/developer/common/Dockerfile](#))
- webProfile6 ([ga/developer/webProfile6/Dockerfile](#))
- webProfile7 ([ga/developer/webProfile7/Dockerfile](#))
- javaee7, latest ([ga/developer/javaee7/Dockerfile](#))
- beta ([beta/Dockerfile](#))

The images in this repository are a mirror of those in the [websphere-liberty](#) official repository.  
For more information about this image and its history, please see [the relevant manifest file](#) ([library/websphere-liberty](#)). This image is updated via pull requests to the [docker-library/official-images](#) GitHub repo.

# Dockerfile

- Build an image automatically
- Specifies base image and instructions:
  - **FROM** <existing image>
  - **ADD** <local file> <path inside image>
  - **RUN** <cmd>
  - **EXPOSE** <port>
  - **ENV** <name> <value>
  - **CMD** <cmd>

```
# Use latest jboss/base-jdk:7 image as the base
FROM jboss/base-jdk:7

# Set the WILDFLY_VERSION env variable
ENV WILDFLY_VERSION 8.1.0.Final

# Add the WildFly distribution to /opt
RUN cd $HOME && curl http://download.jboss.org/wildfly/$WILDFLY_VERSION.tar.gz | tar zx && mv $HOME/wildfly-$WILDFLY_VERSION /opt/jboss/wildfly

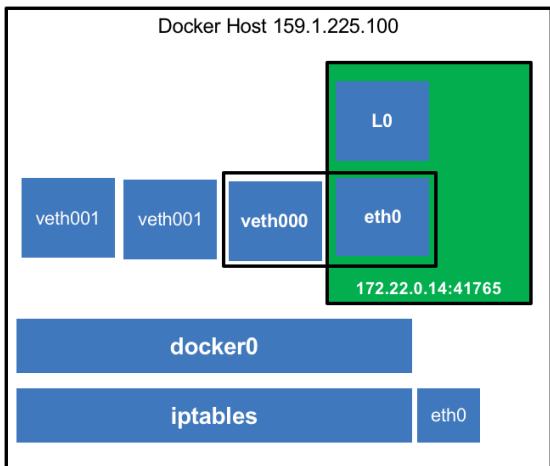
# Set the JBOSS_HOME env variable
ENV JBOSS_HOME /opt/jboss/wildfly

# Expose the ports we're interested in
EXPOSE 8080 9990

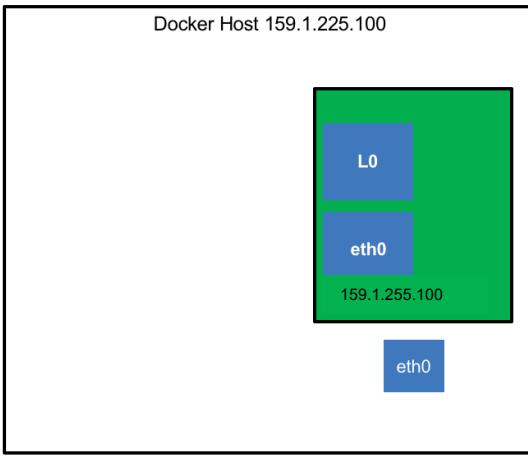
# Set the default command to run on boot
CMD ["/opt/jboss/wildfly/bin/standalone.sh", "-b", "0.0.0.0"]
```

# Docker Networking in a Host

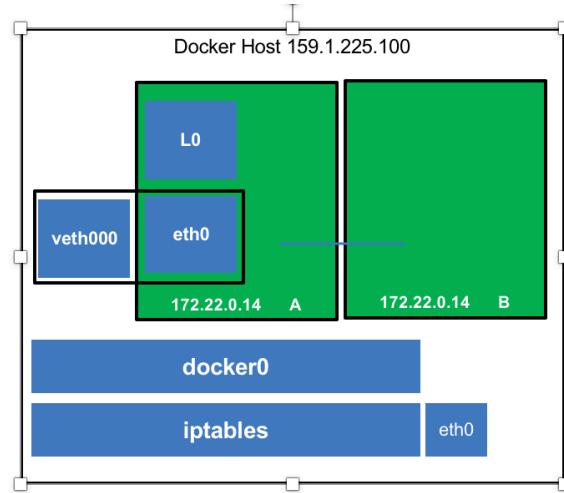
**Bridge Mode**



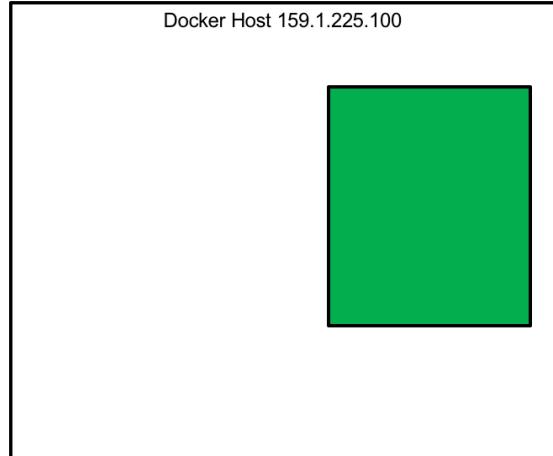
**Host Mode**



**Container Mode**



**No Networking**



```
docker run -d --net=bridge nginx:1.9.1
```

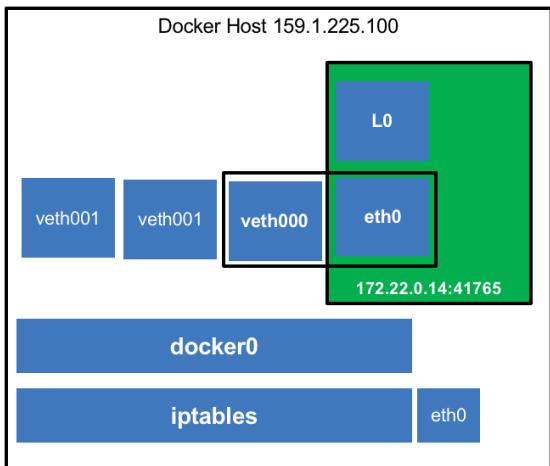
```
docker run -d -P --net=host ubuntu:14.04
```

```
docker run -it --net=container:anothercontainer ubuntu:14.04 ip addr ...
```

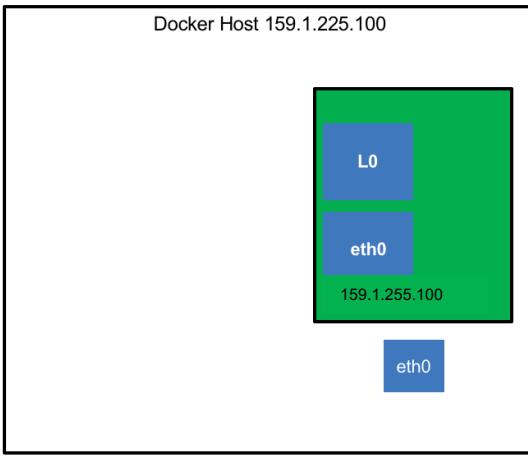
```
docker run -d --net=none ubuntu:latest
```

# Docker Networking in a Host

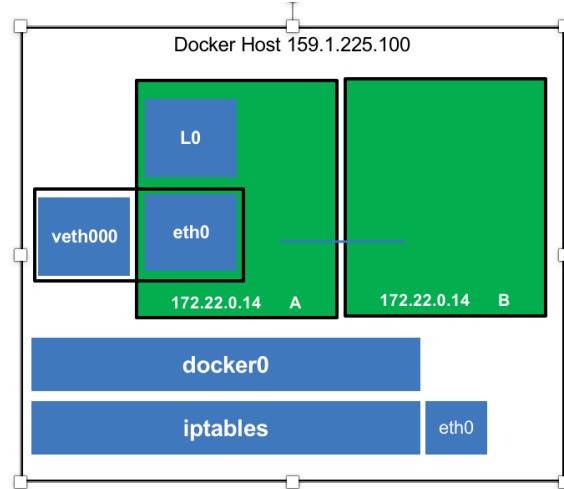
Bridge Mode



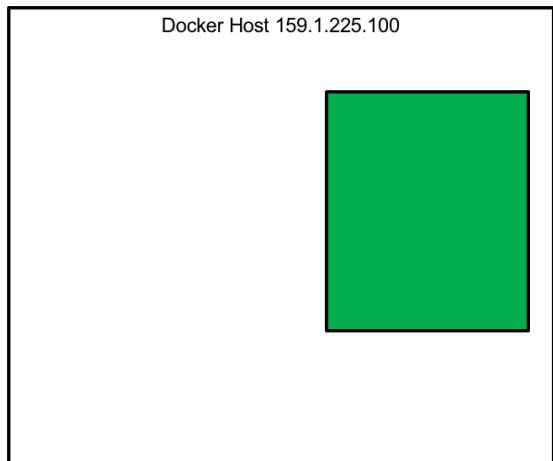
Host Mode



Container Mode



No Networking



```
docker run -d --net=bridge nginx:1.9.1
```

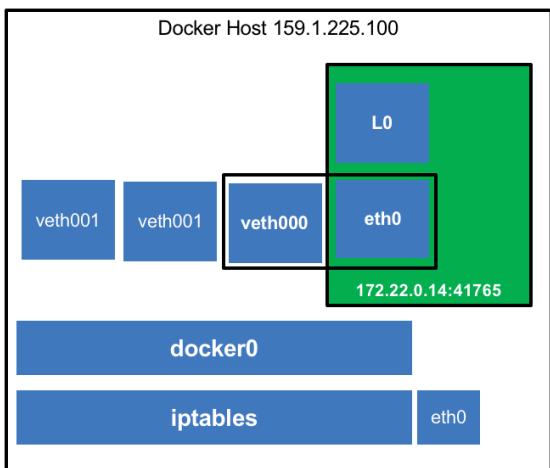
```
docker run -d -P --net=host ubuntu:14.04
```

```
docker run -it --net=container:anothercontainer ubuntu:14.04 ip addr ...
```

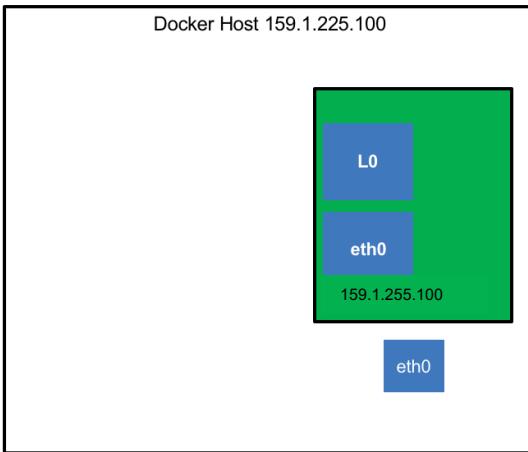
```
docker run -d --net=none ubuntu:latest
```

# Docker Networking in a Host

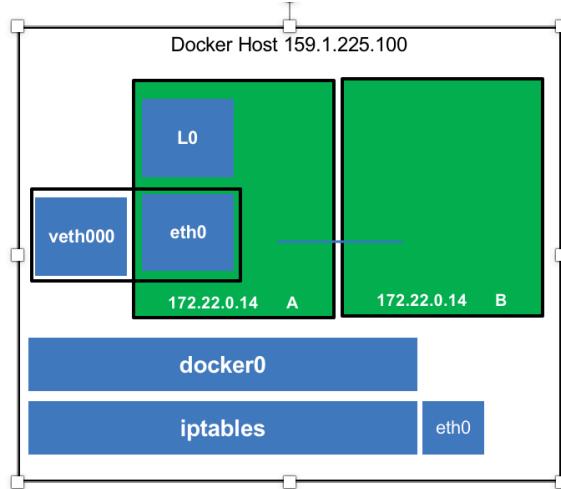
Bridge Mode



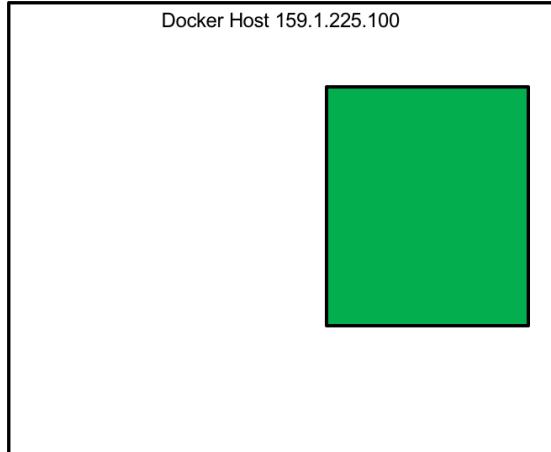
Host Mode



Container Mode



No Networking



```
docker run -d --net=bridge nginx:1.9.1
```

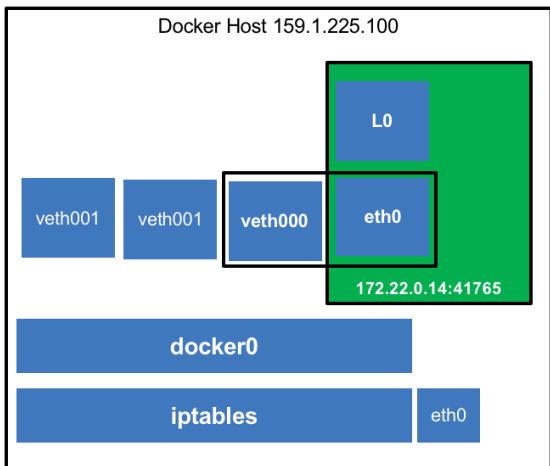
```
docker run -d -P --net=host ubuntu:14.04
```

```
docker run -it --net=container:anothercontainer ubuntu:14.04 ip addr ...
```

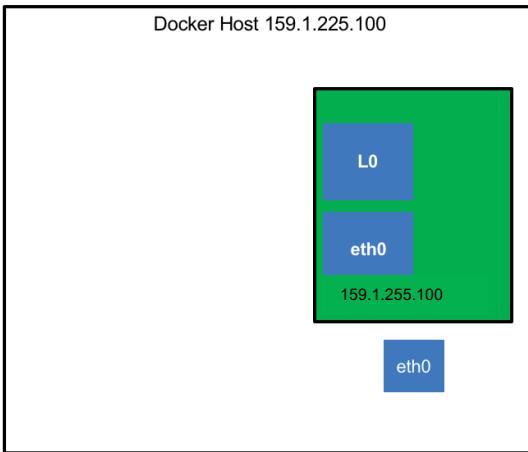
```
docker run -d --net=none ubuntu:latest
```

# Docker Networking in a Host

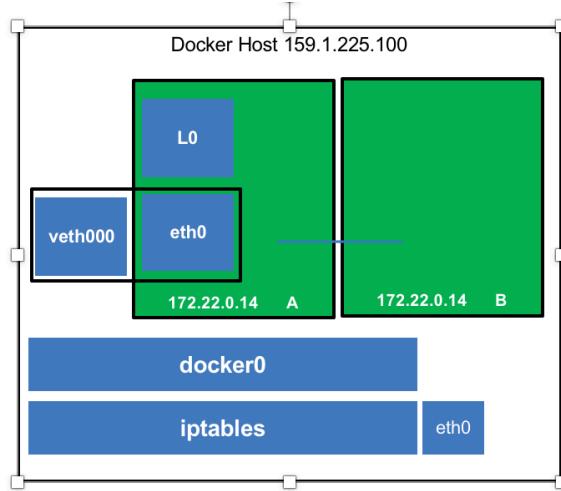
Bridge Mode



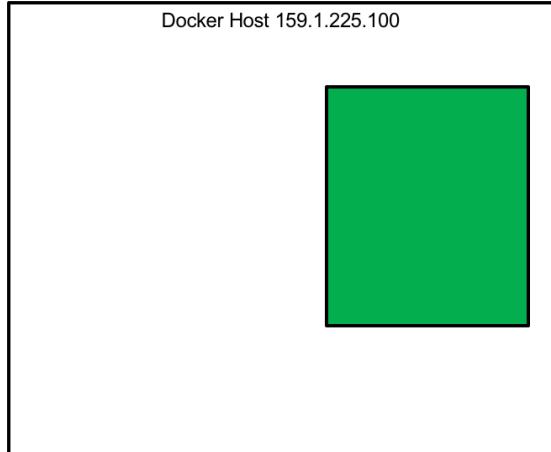
Host Mode



Container Mode



No Networking



```
docker run -d --net=bridge nginx:1.9.1
```

```
docker run -d -P --net=host ubuntu:14.04
```

```
docker run -it --net=container:anothercontainer ubuntu:14.04 ip addr ...
```

```
docker run -d --net=none ubuntu:latest
```

# Docker Networking for Multiple Hosts

---

- SDN = Software Defined Network
- L2 solution (overlay network) :
  - Docker Networking (default)
  - Flannel
  - Weave Net
  - Open vSwitch
  - OpenVPN
- Project Calico (L3 solution & BGP)

# Docker-Compose

- Compose is a tool for defining and running multi-container Docker applications.
- With Compose, you use a YAML file to configure your application's services.
- Then, with a **single command**, you create and start all the services from your configuration.
  - `docker-compose up`

## docker-compose.yml

```
version: '3'
services:
  web:
    build: .
    ports:
      - "5000:5000"
    volumes:
      - .:/code
      - logvolume01:/var/log
    links:
      - redis
  redis:
    image: redis
volumes:
  logvolume01: {}
```

# IBM and Docker Partnership

---

- Strategic partnership announced December, 2014
  - <https://www-03.ibm.com/press/us/en/pressrelease/45597.wss>
- Partnership extended February, 2016
  - IBM initially only partner to resell and support Docker Datacenter
- Objective: Deliver next generation enterprise-grade, portable, distributed applications that are composed of interoperable Docker containers
  - Enables hybrid cloud use cases for the enterprise
  - IBM Cloud Container Service since 2014
- Initiatives Underway especially with IBM Cloud Private

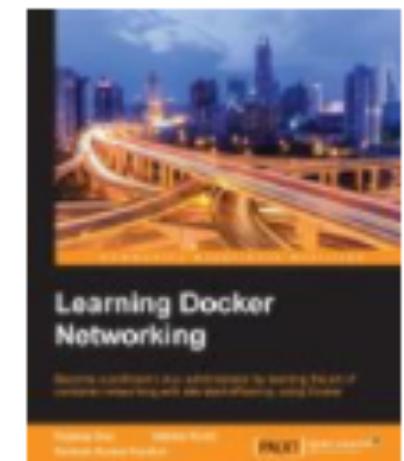
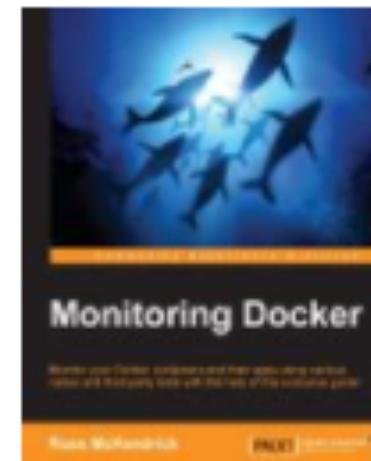
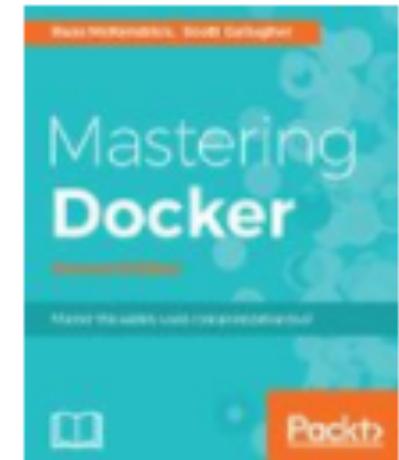
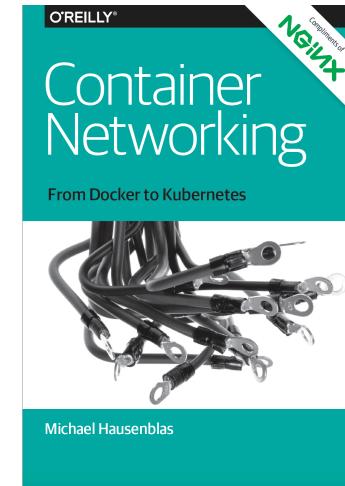
# Advantages of Containers

---

- Containers are **portable**
  - Any platform with a container engine can run containers
- Containers are **easy to manage**
  - Container images are easy to share, download, and delete
    - Especially with Docker registries
  - Container instances are easy to create and delete
  - Each container instance is easy and fast to start and stop
- Containers provide “just enough” **isolation**
  - Processes share the operating system kernel but are segregated
- Containers use hardware more **efficiently**
  - Greater density than virtual machines
  - Especially Docker containers, which can share layers
- Containers are **immutable**

# Books, eBooks and links

- Mastering Docker (second edition)
  - Learning Docker Networking
  - Container Networking
  - Monitoring Docker
- 
- <https://docs.docker.com/>

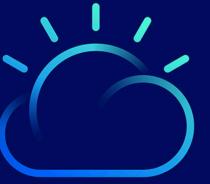




# Creating a docker image & containerizing an application

Demo





# Preparation Lab & Docker Lab

## Labs



# Labs

---

- <https://ibm.biz/container-ws>

## PrepareLab

- Installing Docker on your laptop
- Installaing the ibmcloud (ic) commands

## DockerLab

- Working with Docker
- Building Docker images
- Running Web Application



The image shows a presentation slide titled "IBM Cloud Containers Workshop" in large bold letters at the top. Below the title, it says "IBM Cloud Containers Workshop" and "version 1.0". Underneath, there is a section titled "Agenda" which includes "Day One" and a bulleted list of topics: "Introduction / IBM Cloud", "Docker Overview and Lab :", and two links: "<https://fdescollonges.github.io/IBM-Cloud-Containers-Workshop/1-PrepareLab/>" and "<https://fdescollonges.github.io/IBM-Cloud-Containers-Workshop/2-DockerLab/>".