

# Docker JumpStart

## Getting Started with Docker

# Mike Pfeiffer



Blog

<https://mikepfeiffer.io>



Twitter

@Mike\_Pfeiffer

Mike Pfeiffer is a twenty year IT industry veteran, published author, and international conference speaker. He's a former architect for Amazon Web Services and engineer for Microsoft. Today Pfeiffer serves as Chief Technologist at CloudSkills.io.

# Dan Wahlin



Blog

<https://blog.codewithdan.com>



Twitter

@DanWahlin

Dan Wahlin founded Wahlin Consulting (<https://codewithdan.com>) which provides training and architecture services on front-end and back-end web technologies, Microservices, and Docker/Kubernetes. He's published multiple courses on Pluralsight.com and is a Docker Captain, Microsoft MVP/Regional Director, and Google GDE. Dan speaks at multiple conferences and runs the Code with Dan development newsletter.

# Workshop Logistics

- 9 am – 4 pm PST
- Lunch from 12 – 1 pm PST (with breaks in the morning/afternoon)
- Recordings will be available by June 28<sup>th</sup>
- Please mute your microphone
- Ask questions using the chat
- For more details/questions/answers we'll call on you to unmute your mic
- Hands-on labs

# Pre-Reqs

Prior experience with one of the following:

- Command line
- Systems administration
- Software development

# Course Expectations

- Learn Docker from the ground up
- Learn how to work with images and containers
- Learn different container orchestration techniques
- Learning is not effective unless you're relaxed and having fun.  
Have fun and enjoy it! ☺

# Workshop Content

## Day 1

- Getting Started with Docker
- Setting Up Your Environment
- Building and Managing Docker Images
- Running and Managing Docker Containers



# Workshop Content

## Day 2

- Mastering Container Volumes
- Container Networking Demystified
- Docker Compose Orchestration In-Depth
- Container Orchestration with Kubernetes



# Agenda

- Why Docker?
- Getting Started with Docker
- Benefits to Developers and DevOps
- Containers vs. Virtual Machines
- Docker and Microservices

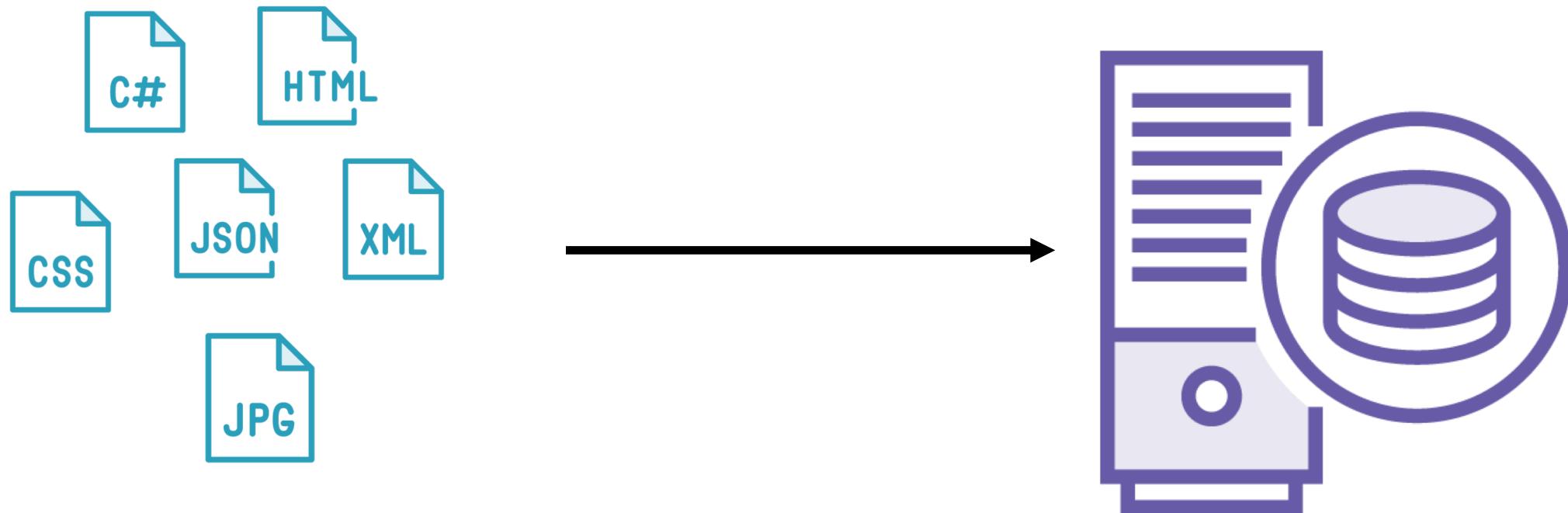


# Why Docker?



Shipping  
Applications  
the Traditional  
Way

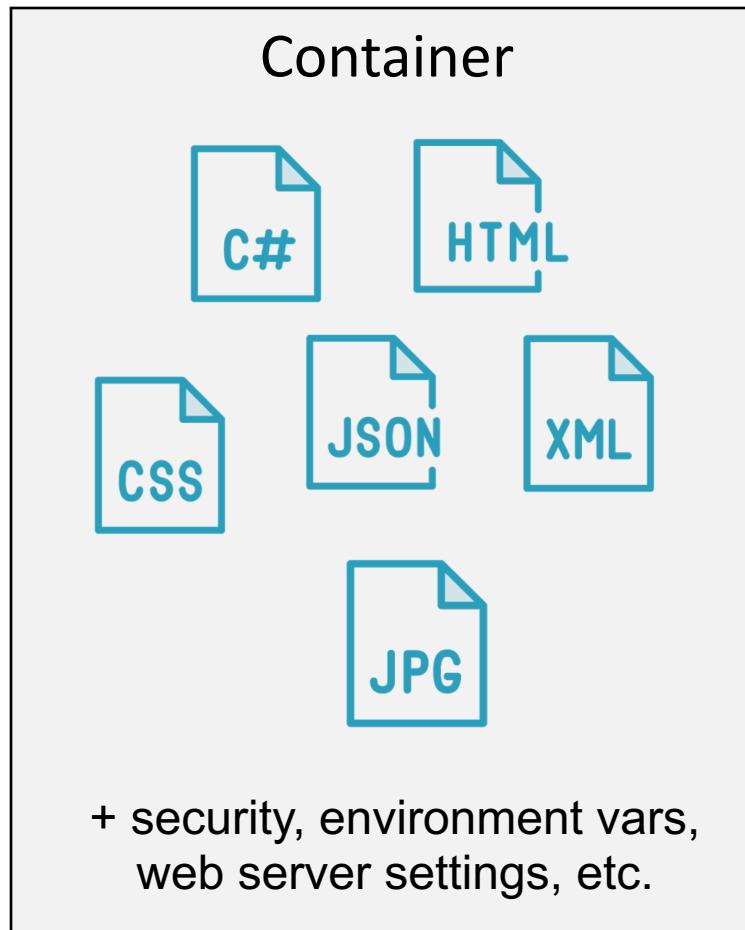
# Shipping Applications



# Shipping Applications with Containers

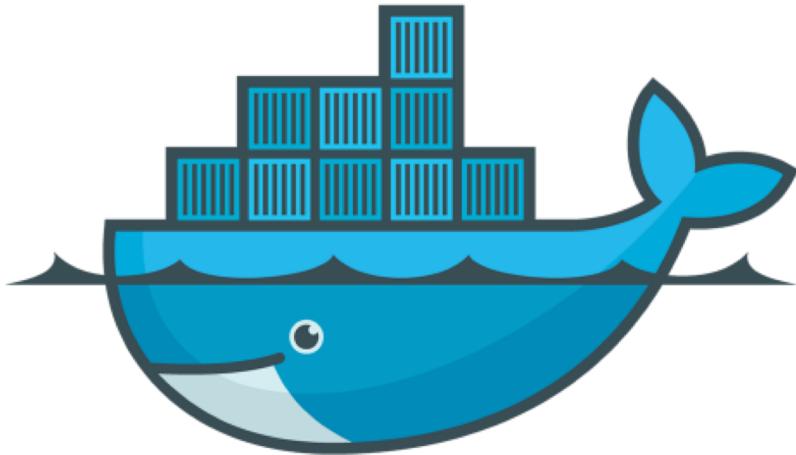


# Shipping Applications using Containers



# Getting Started with Docker

What is Docker?



Simplify building, shipping, running apps

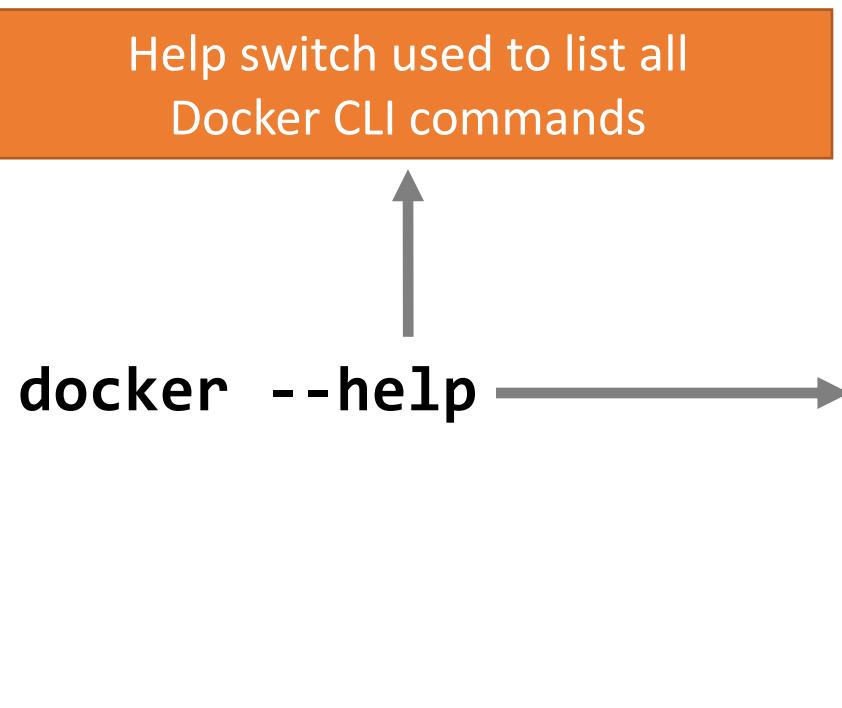
Shipping container system for code

Runs natively on Linux or Windows Server 2016+

Runs on Windows or Mac Development machines  
(with a virtual machine)

Relies on images and containers

# Docker Client CLI



```
Dans-iMac:~ danwahlin$ docker --help

Usage: docker COMMAND

A self-sufficient runtime for containers

Options:
  --config string          Location of client config files (default
                           "/Users/danwahlin/.docker")
  -D, --debug              Enable debug mode
  --help                   Print usage
  -H, --host list           Daemon socket(s) to connect to
  -l, --log-level string   Set the logging level
                           ("debug"|"info"|"warn"|"error"|"fatal") (default
                           "info")
  --tls                   Use TLS; implied by --tlsverify
  --tlscacert string       Trust certs signed only by this CA (default
                           "/Users/danwahlin/.docker/ca.pem")
  --tlscert string         Path to TLS certificate file (default
```

# Docker Client Version

Get Docker client CLI version



**docker -v**

```
Dans-iMac:~ danwahlin$ docker -v
Docker version 17.05.0-ce, build 89658be
Dans-iMac:~ danwahlin$
```

```
C:\Users\dwahl>docker -v
Docker version 17.05.0-ce, build 89658be
C:\Users\dwahl>
```

# Docker Client Information

Get information about host, images,  
containers and more

**docker info** →

```
Dans-iMac:~ danwahlin$ docker info
Containers: 0
Running: 0
Paused: 0
Stopped: 0
Images: 67
Server Version: 17.05.0-ce
Storage Driver: overlay2
Backing Filesystem: extfs
Supports d_type: true
Native Overlay Diff: true
Logging Driver: json-file
Cgroup Driver: cgroupfs
Plugins:
    Volume: local
    Network: bridge host ipvlan macvlan null overlay
Swarm: inactive
Runtimes: runc
Default Runtime: runc
Init Binary: docker-init
containerd version: 9048e5e50717ea4497b757314bad98ea3763c145
runc version: 9c2d8d184e5da67c95d601382adf14862e4f2228
init version: 949e6fa
```

# Benefits to Developers and DevOps

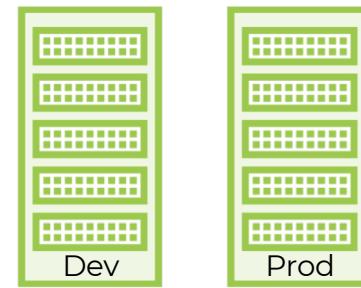
# Docker Benefits



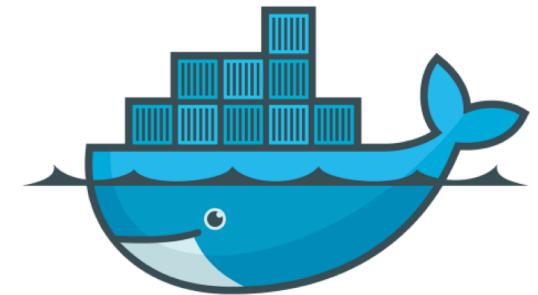
Accelerate  
Developer  
Onboarding



Eliminate App  
Conflicts

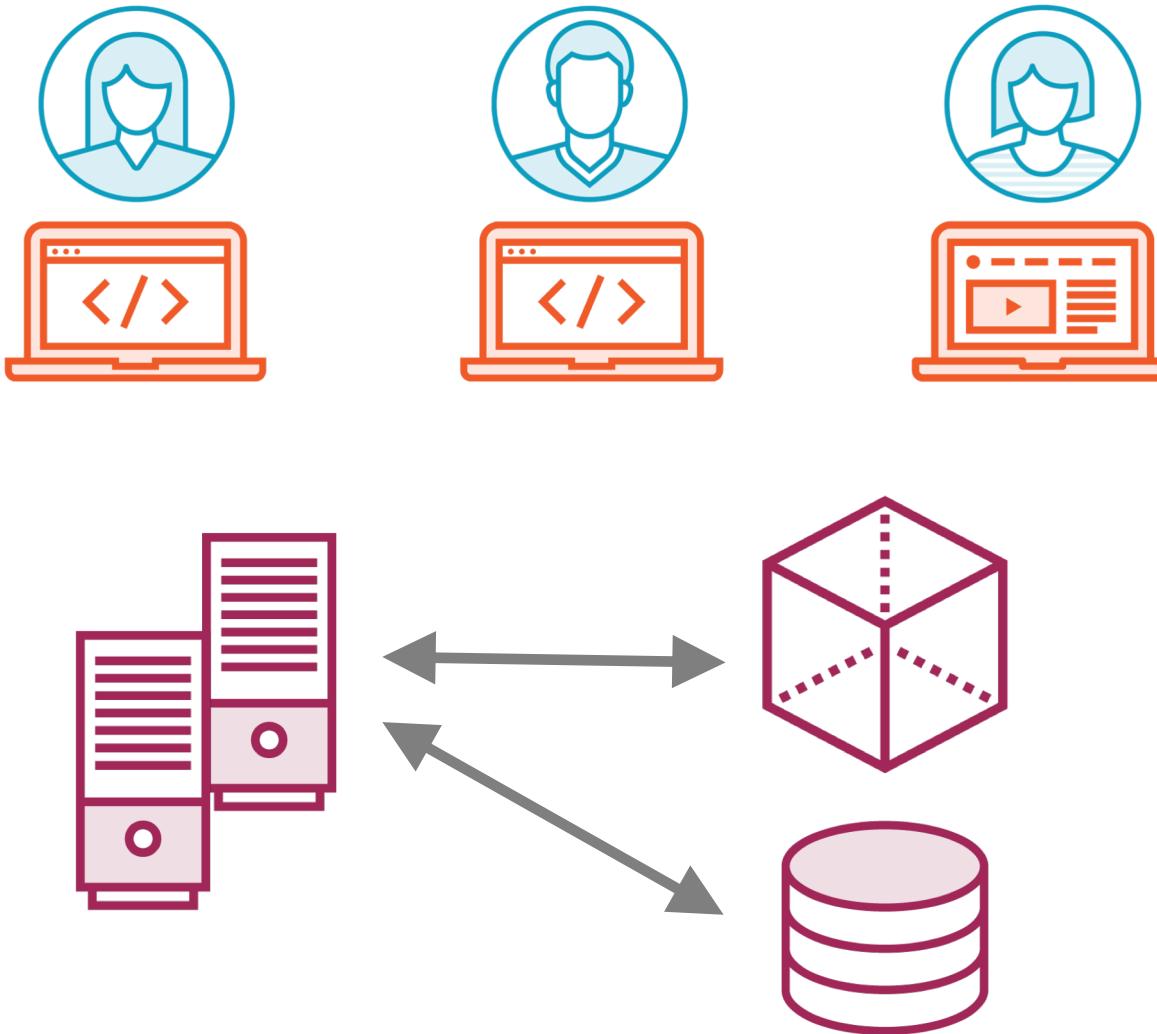


Environment  
Consistency

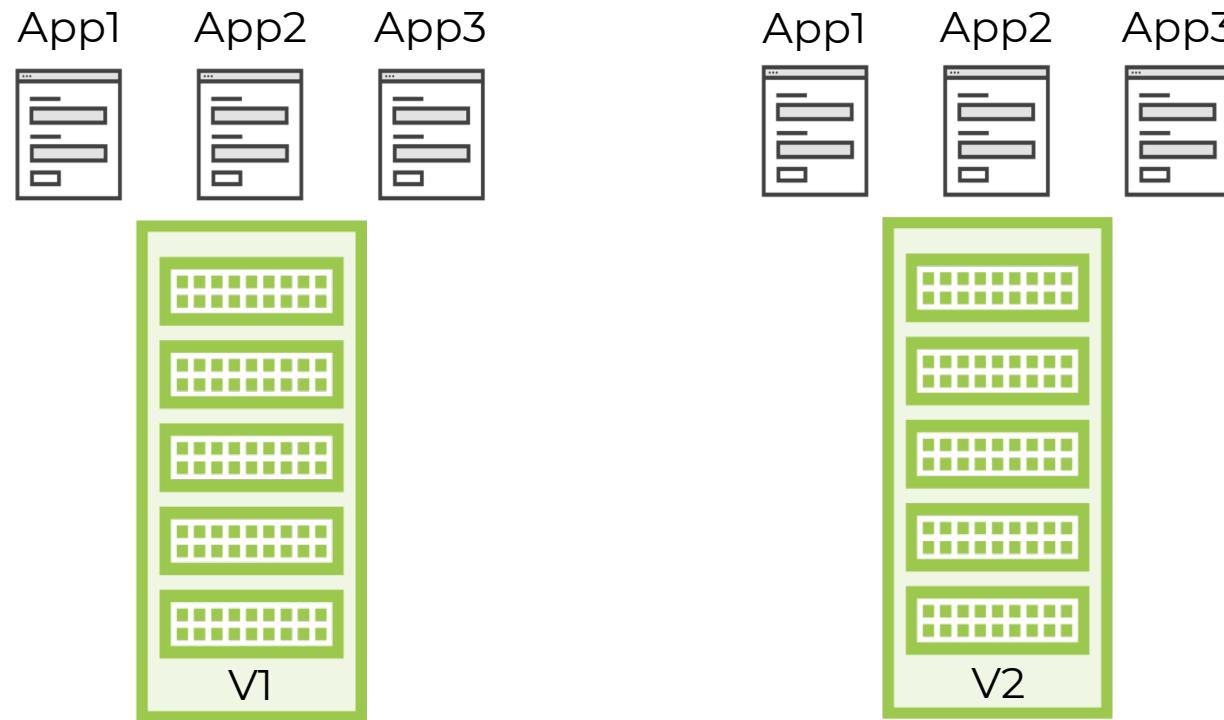


Ship Software  
Faster

# Accelerate Developer Onboarding

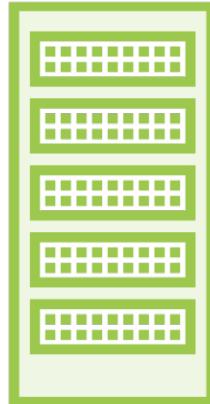


# Eliminate App Conflicts

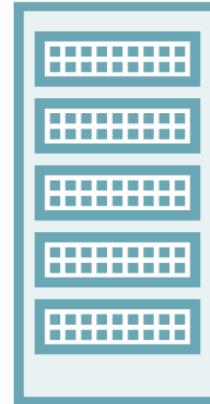


# Environment Consistency

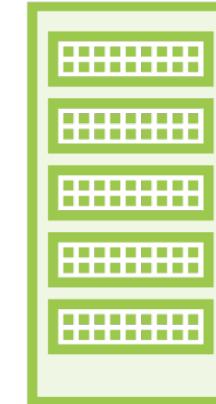
Development



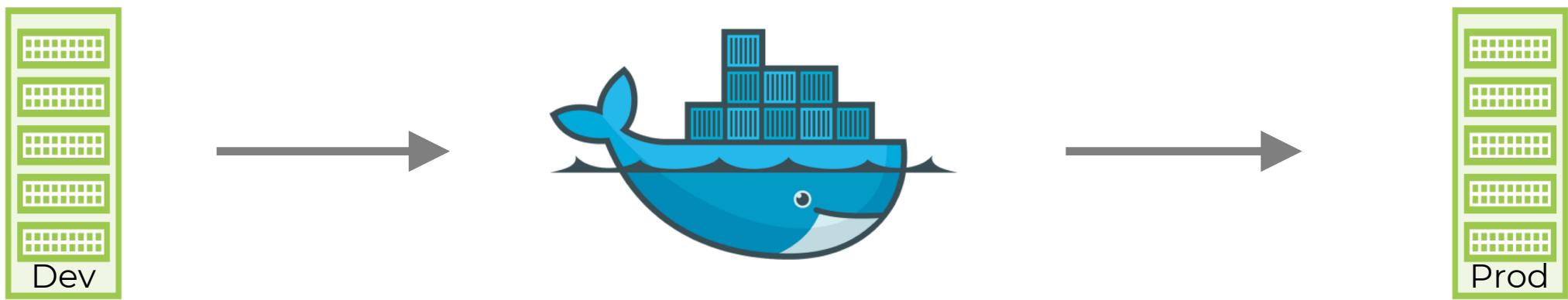
Staging



Production



# Ship Software Faster



# Container Benefits Review

- Onboard developers more quickly to new/existing projects
- "Containerize" existing applications
- Convert monolithic applications to microservices
- Provide environment consistency
- Deploy more reliably
- Ship software faster
- Reduce infrastructure costs



# Containers vs. Virtual Machines

# Where Does Docker Run?

Docker Client



Linux



Windows

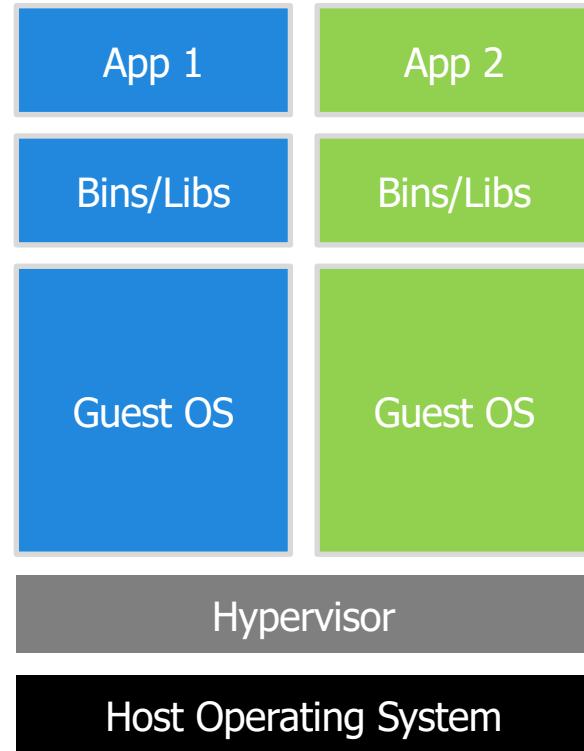
Docker Engine  
(Daemon)

Docker Engine  
(Daemon)

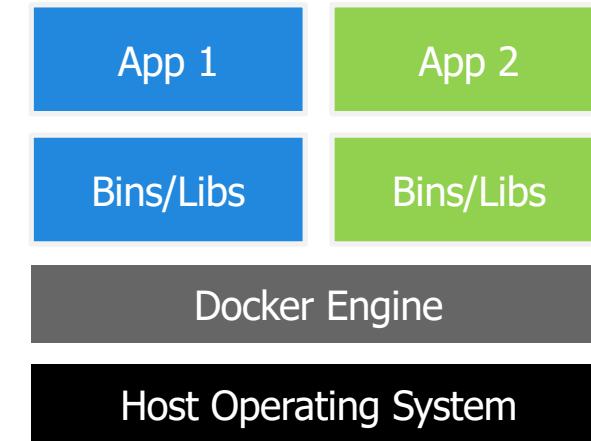
Linux Container  
Support (LXC)

Windows Server  
Container Support

# Docker Containers Versus Virtual Machines



Virtual Machines



Docker Containers

# Docker and Microservices



Containers  
simplify  
shipping  
microservices

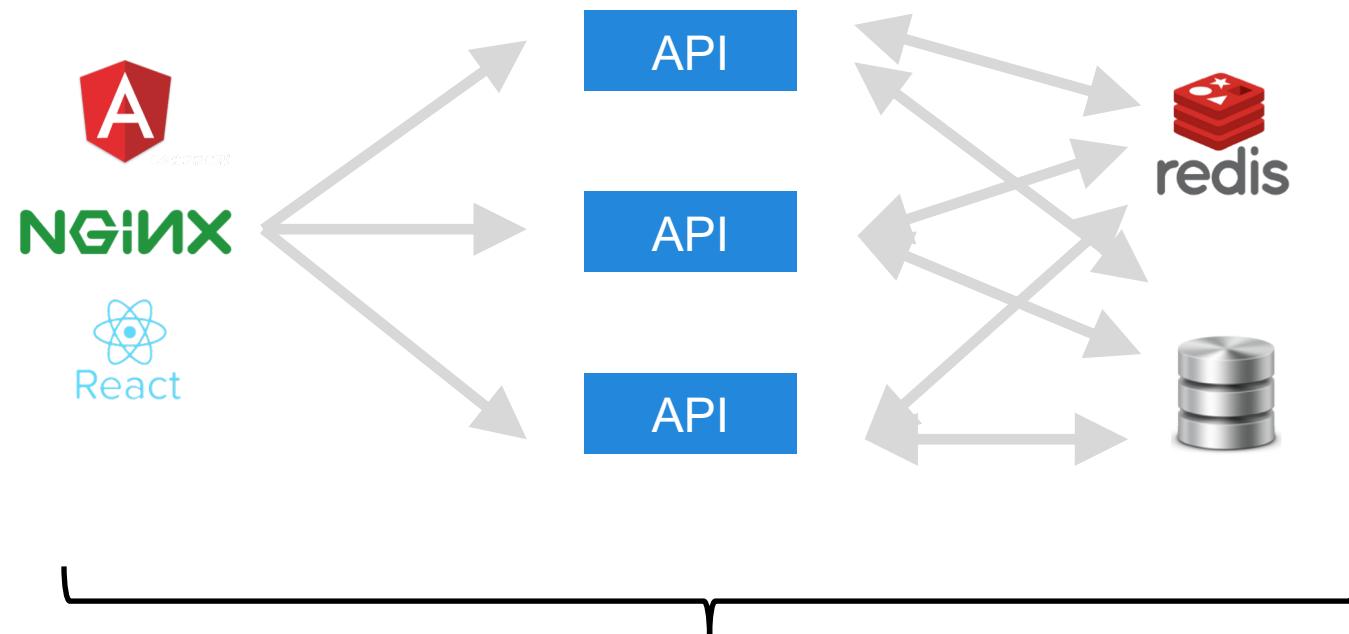
Microservice 1

Microservice 2

Microservice 3

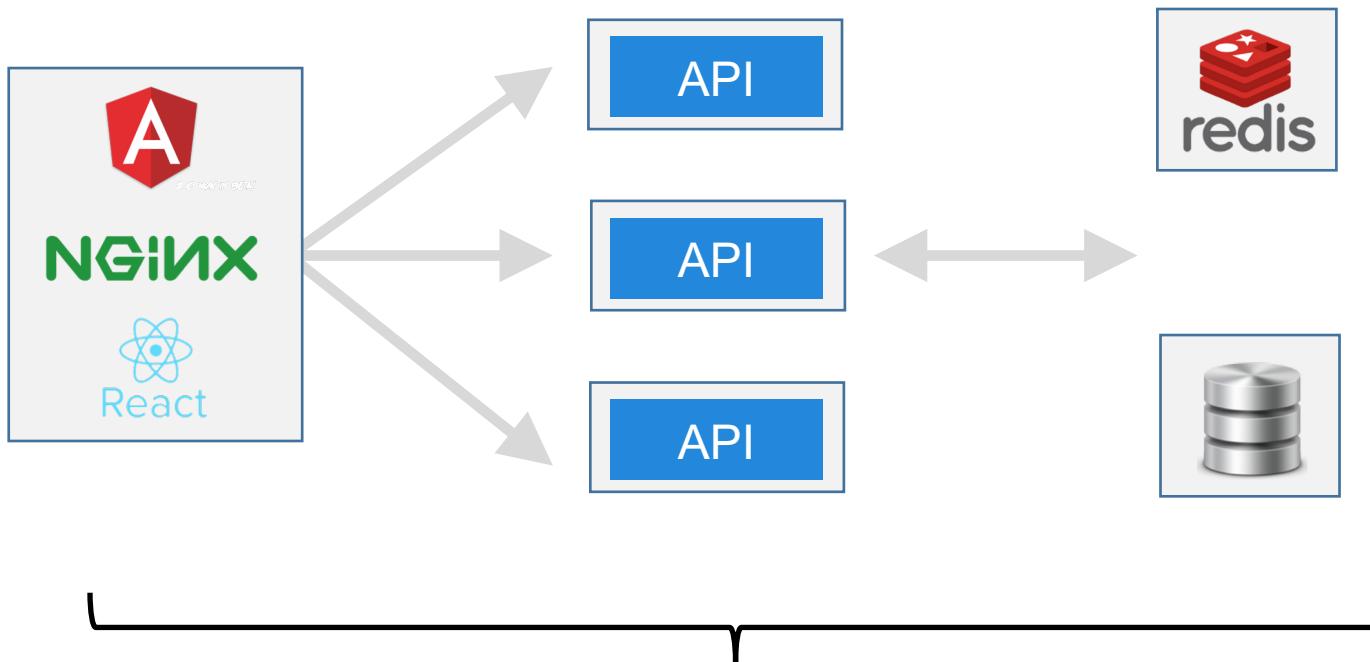
IMO 9455038  
**APL ZEEBRUGGE**  
PANAMA

# Running Microservices



How do you make all of these services work **exactly** the same in development/staging/production?

# Docker and Microservices



Containers can provide consistency between environments

# Summary

- Deploying applications isn't as simple as moving code between environments
- Containers provide a way to consistently move applications between environments
- Images are read-only templates that can be used to create containers
- Docker containers are very different from virtual machines
- Docker can be used to implement microservices

