# Docker for Web Developers

**Dan Wahlin**
PRESIDENT - WAHLIN CONSULTING

@danwahlin www.codewithdan.com

# Placeholder

Promo Video Demo
of Parts of Course Goes Here

I'll talk over the video
Created after the course is done

# Course Agenda

- Why Use Docker as a Developer?
- Setting Up Your Dev Environment with Docker Toolbox
- Using Docker Client in Your Dev Environment
- Hooking Your Source Code Into a Container
- Linking Docker Containers
- Managing Multiple Containers with Docker Compose
- Deploying Your Containers to Azure
- Reviewing the Case for Docker

# Why Use Docker as a Developer?

**Dan Wahlin**
WAHLIN CONSULTING

@danwahlin www.codewithdan.com

# Module Agenda

What is Docker?

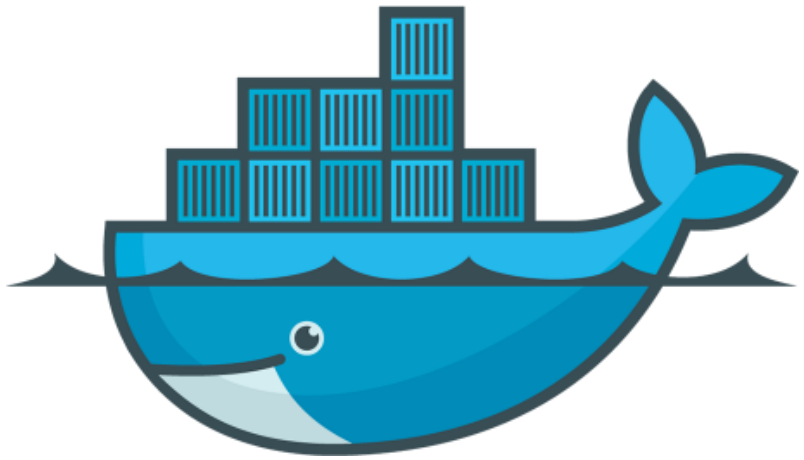
Docker Benefits
(for Web Developers)

Docker Tools

Docker in Action

# What is Docker?

## What is Docker?



Lightweight, open, secure platform

Simplify building, shipping, running apps

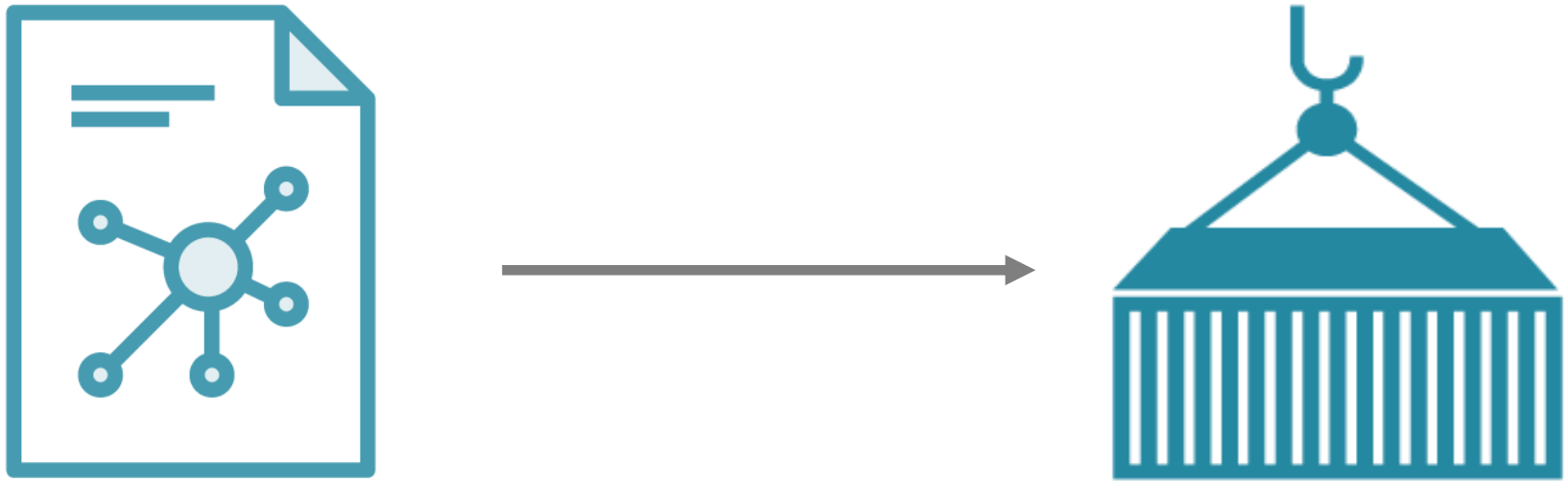
Shipping container system for code

Runs natively on Linux or Windows Server

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

Relies on "images" and "containers"

# The Role of Images and Containers



**Docker Image**

Example: Ubuntu with Node.js and Application Code

**Docker Container**

Created by using an image. Runs your application.

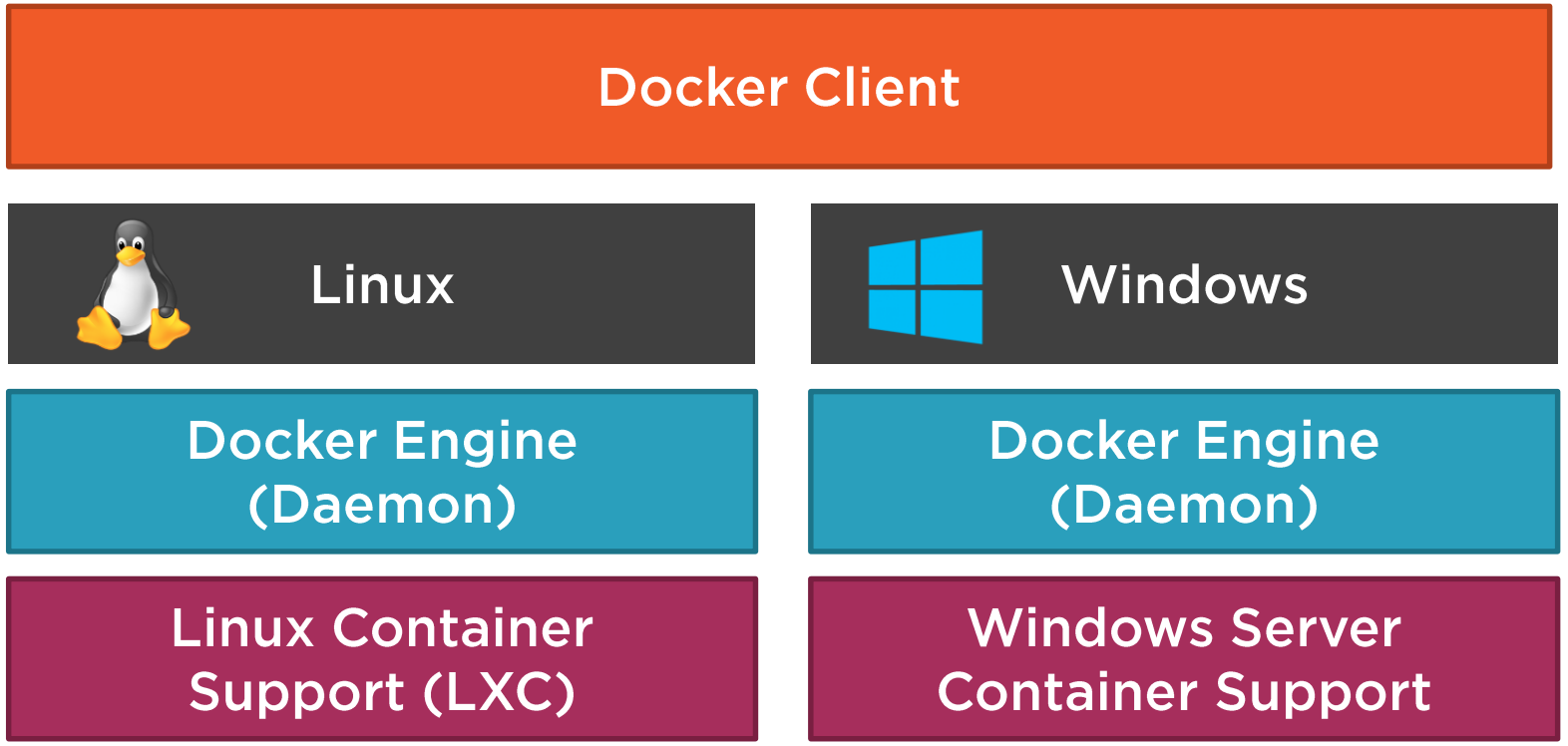
# Image

A **read-only template** composed of layered filesystems used to share common files and create Docker container instances.

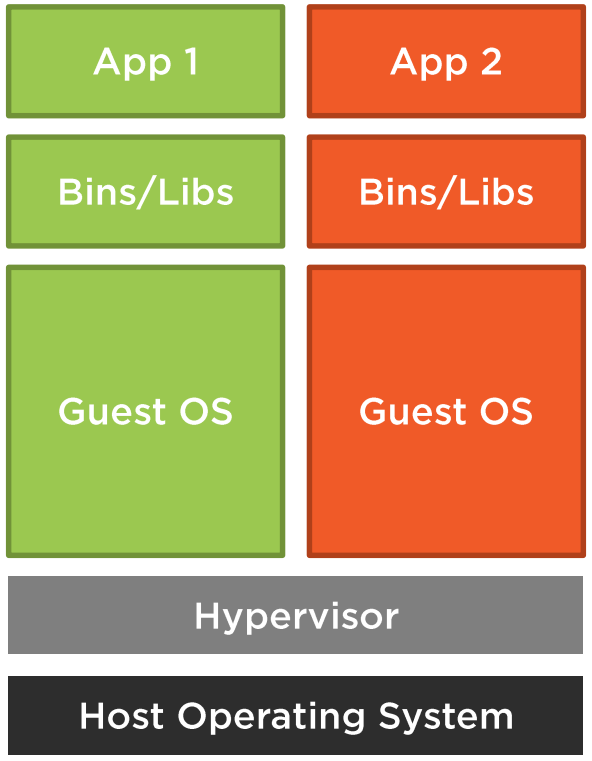
# Container

**An isolated and secured shipping container** created from an image that can be run, started, stopped, moved and deleted.

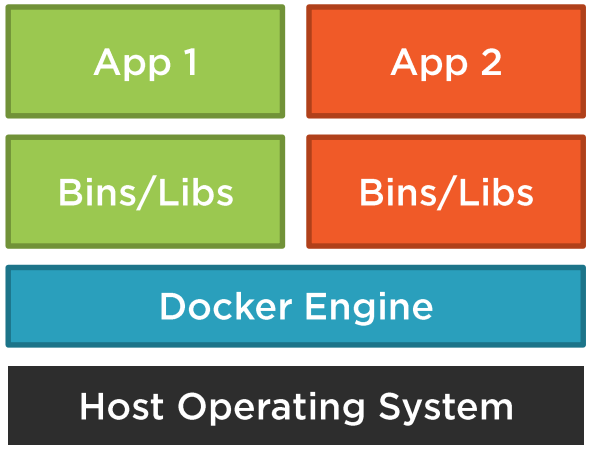
# Where Does Docker Run?



# Docker Containers Versus Virtual Machines




Virtual Machines

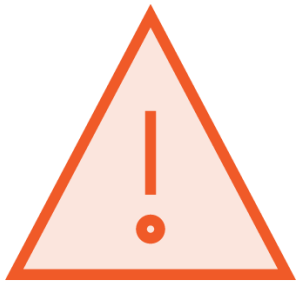



Docker Containers

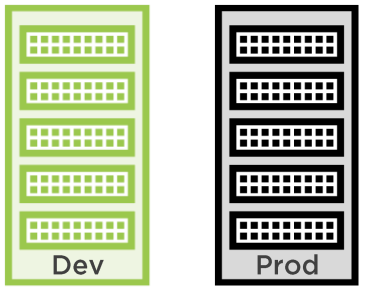
# Docker Benefits (for Web Developers)

# Docker Benefits
(for Web Developers)

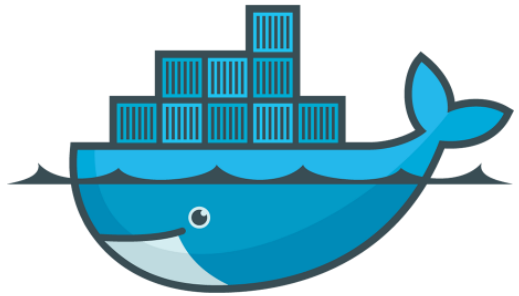
Accelerate Developer Onboarding



Eliminate App Conflicts

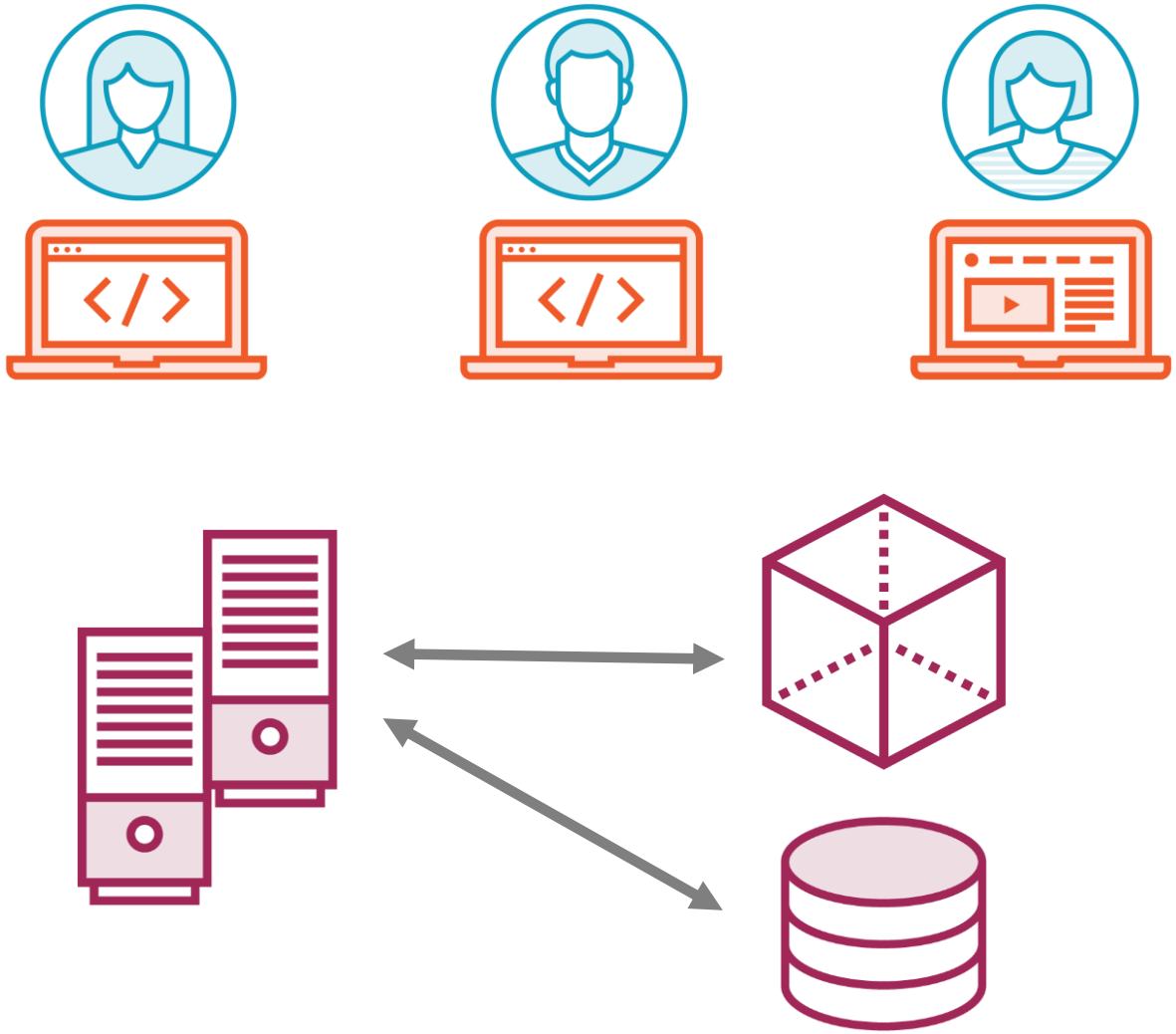



Environment Consistency

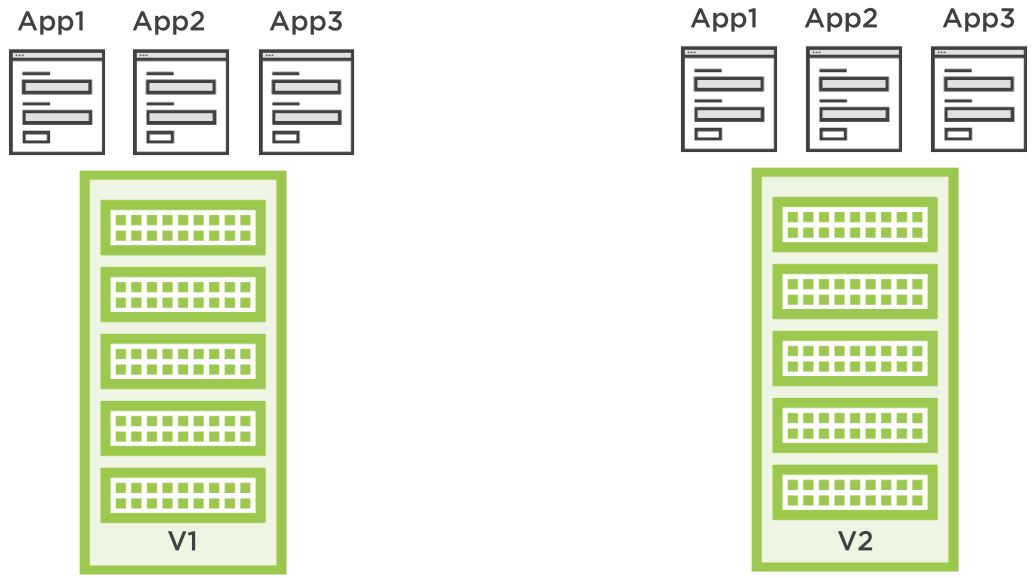


Ship Software Faster

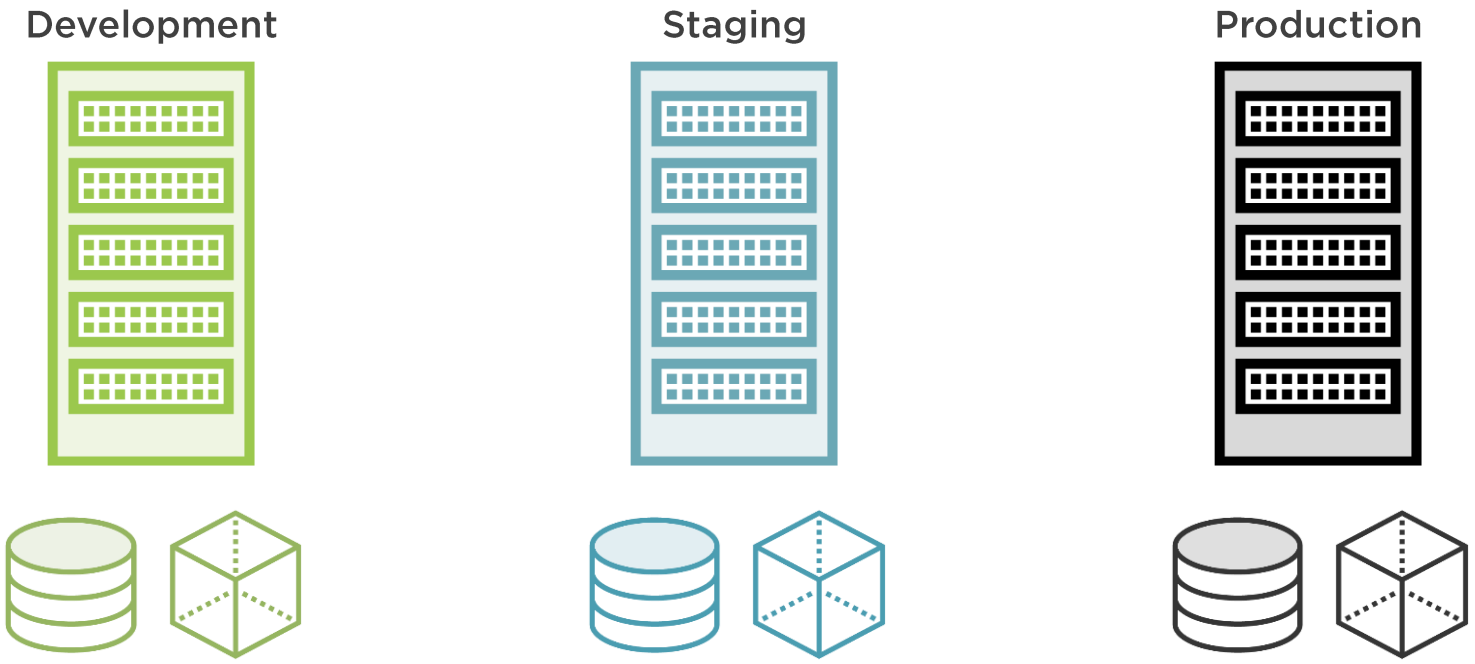
# Accelerate Developer Onboarding



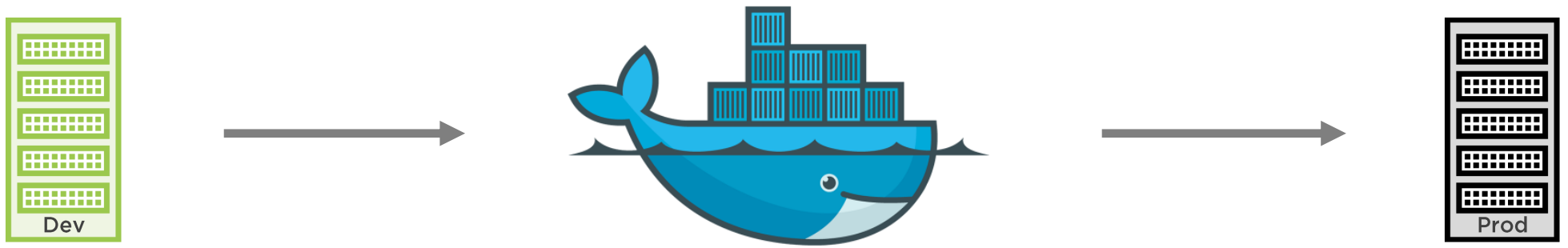
# Eliminate App Conflicts



# Environment Consistency



# Ship Software Faster



# Docker Tools

## Docker Toolbox

Docker Toolbox (Windows 7/8 or Mac)

Provides image and container tools

Virtual Machine (for Windows/Mac)

Works on Windows, Mac, Linux

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

## Docker CE

Docker Community Edition (Windows 10 Pro+ or Mac)

Provides image and container tools

Using Hyper-V/Hyperkit to run VMs

Works on Windows, Mac, Linux

https://www.docker.com/community-edition

# Docker Toolbox vs. Docker CE

| Docker Toolbox | Docker Community Edition |
| --- | --- |
| Windows 7 or 8 | Windows 10+ Pro or Mac |
| Uses Virtual Box | Hyper-V/Hyperkit |
| Will use Docker Machine | Don’t normally need to use Docker Machine |

# Docker Toolbox Tools

Docker Client

Docker Machine

Docker Compose

Docker Kitematic

VirtualBox

Docker Toolbox

# Docker in Action

# Summary

**Docker simplifies building, shipping and running apps**

**Runs natively on Linux and Windows Server**

**Docker is NOT the same as using Virtual Machines**

**Key benefits to Web Developers:**

- Accelerate developer onboarding
- Simplify working with multiple apps
- Consistency between environments
- Ship faster!

# Setting up Your Docker Environment

**Dan Wahlin**
WAHLIN CONSULTING

@danwahlin www.codewithdan.com

# Module Agenda

- Installing Docker on Mac
- Installing Docker on Windows
- Getting Started with Docker Kitematic
- Docker Kitematic in Action

# Installing Docker on Mac

# Installing Docker on Windows

# Docker on Windows

Windows 7/8

Install Docker Toolbox

Windows 10+ Pro

Install Docker CE

# Getting Started with Docker Kitematic

# Docker Kitematic Overview

GUI used to provision VMs and work with Images and Containers

Visually Search for Docker images

Create, Run and Manage Containers

# Docker Kitematic in Action

# Summary

Docker Toolbox runs on Windows 7/8

Docker CE runs on Mac or Windows 10 Pro or higher

Docker Kitematic provides a visual way to work with images and containers

# Using Docker Tools

**Dan Wahlin**
WAHLIN CONSULTING

@danwahlin www.codewithdan.com

# Module Agenda

- Getting Started with Docker Machine
- Docker Machine in Action
- Getting Started with Docker Client
- Docker Client in Action
- Docker Commands Review

## Docker Toolbox
## Docker Community Edition

- Docker Client
- Docker Machine
- Docker Compose
- Docker Kitematic
- Virtual Box (Docker Toolbox Only)

# Getting Started with Docker Machine

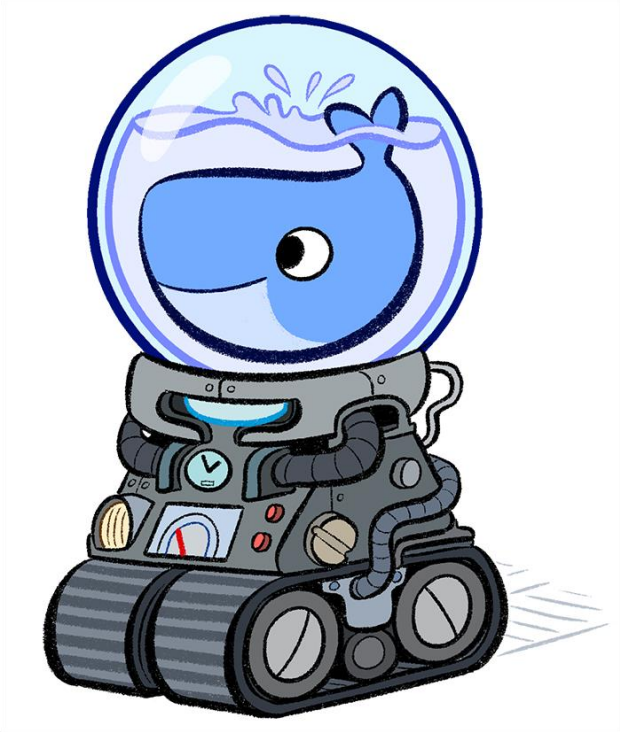
# Docker Machine Overview

Create and Manage Local Machines

Create and Manage Cloud Machines

Configure Docker Client to talk to Machines

# Key Docker Machine Commands



```
docker-machine ls

docker-machine start [machine name]

docker-machine stop [machine name]

docker-machine env [machine name]

docker-machine ip [machine name]
```

# Docker Machine in Action (Mac)

# Docker Machine in Action (Windows)

# Getting Started with Docker Client

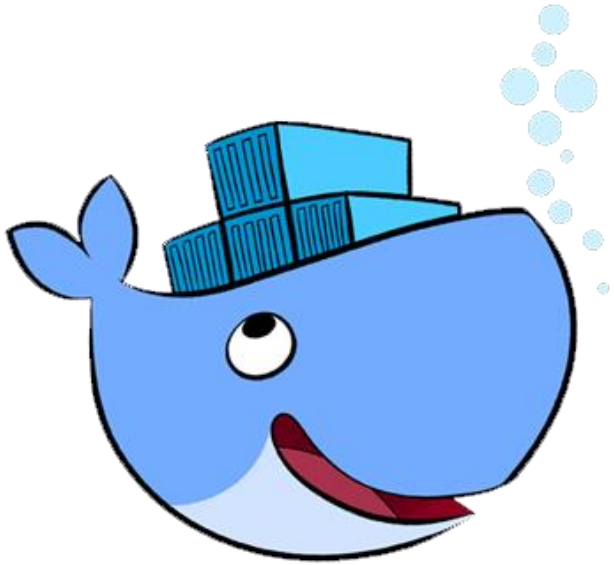
# Docker Client Overview

Interact with Docker Engine

Build and Manage Images

Run and Manage Containers

# Key Docker Client Commands



docker pull [image name]

docker run [image name]

docker images

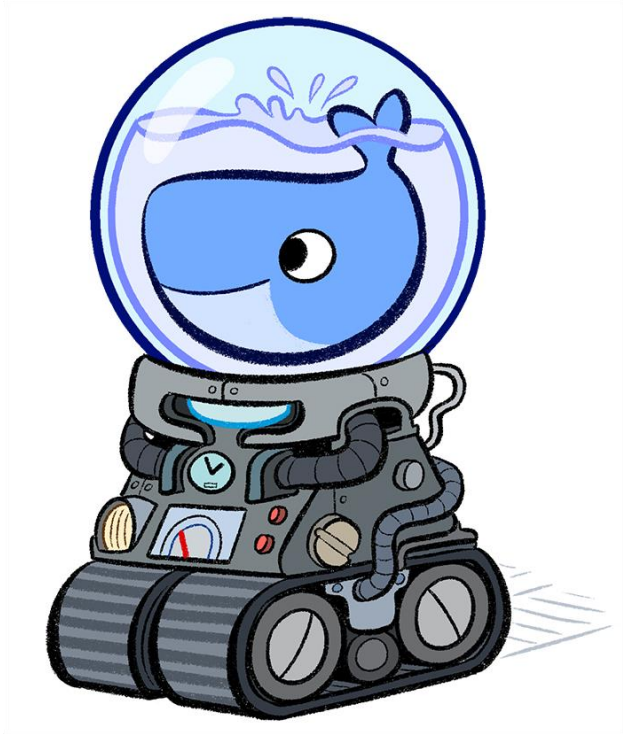
docker ps

# Docker Client in Action (Mac)

# Docker Client in Action (Windows)

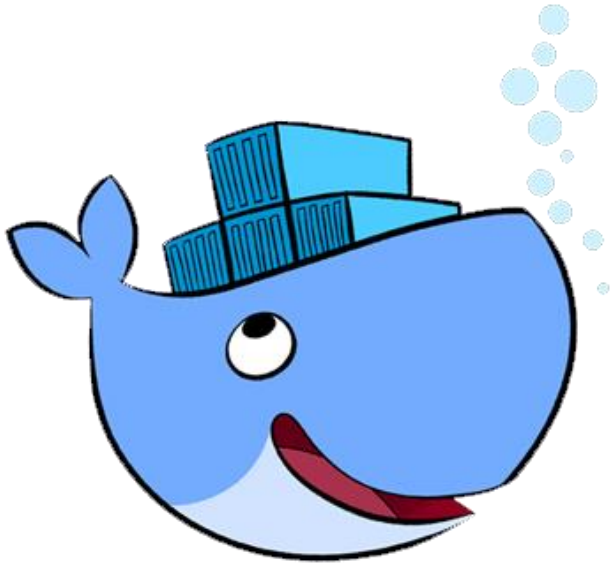
# Docker Commands Review

# Key Docker Machine Commands



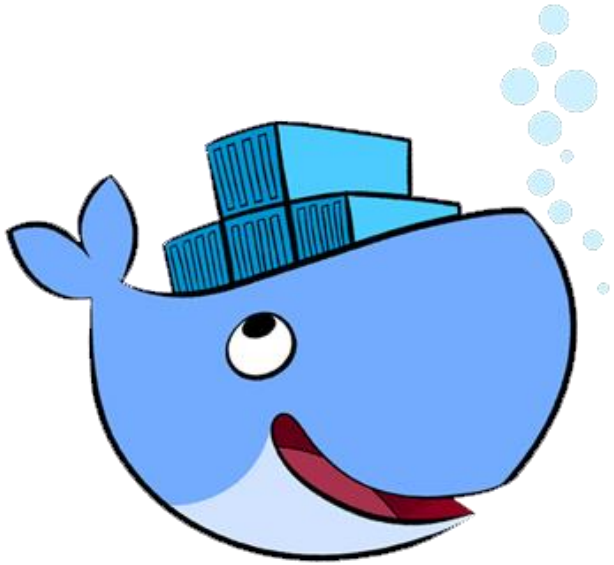
```
docker-machine ls

docker-machine start [machine name]

docker-machine stop [machine name]

docker-machine env [machine name]

docker-machine ip [machine name]

docker-machine status [machine name]
```

# Key Docker Client Image Commands



```
docker pull [image name]

docker images

docker rmi [image ID]
```

# Key Docker Client Container Commands



```
docker run [image name]

docker ps –a

docker rm [container ID]
```

# Summary

Docker Machine creates and manages machines

Docker Client manages images and containers

# Hooking Your Source Code into a Container

**Dan Wahlin**
WAHLIN CONSULTING

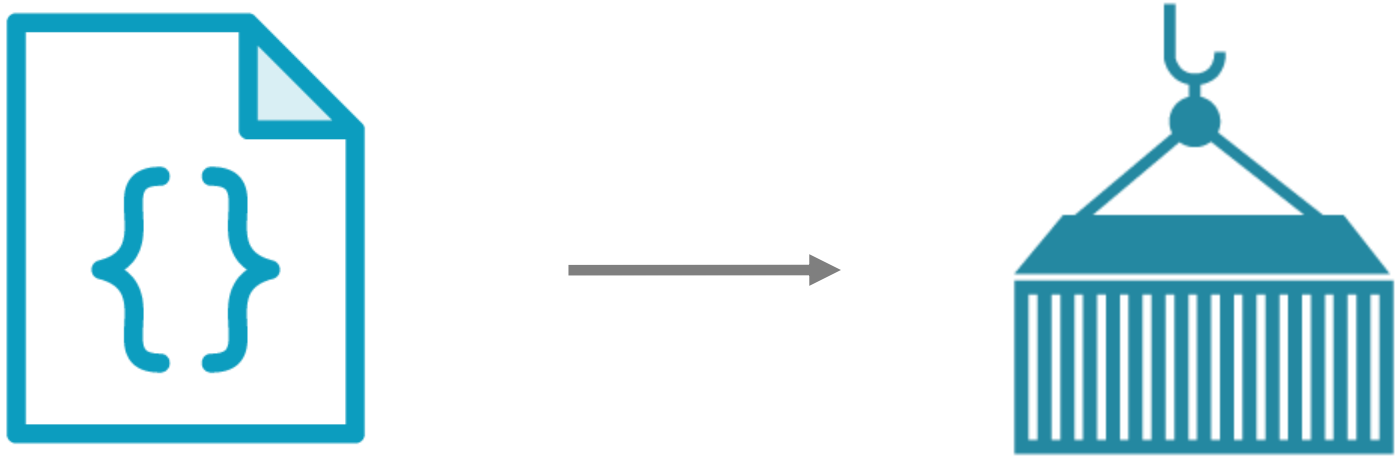
@danwahlin www.codewithdan.com

# Module Agenda

- The Layered File System
- Containers and Volumes
- Source Code, Volumes and Containers
- Hooking a Container Volume to Source Code
- Removing Containers and Volumes

## Question

How do you get source code into a container?



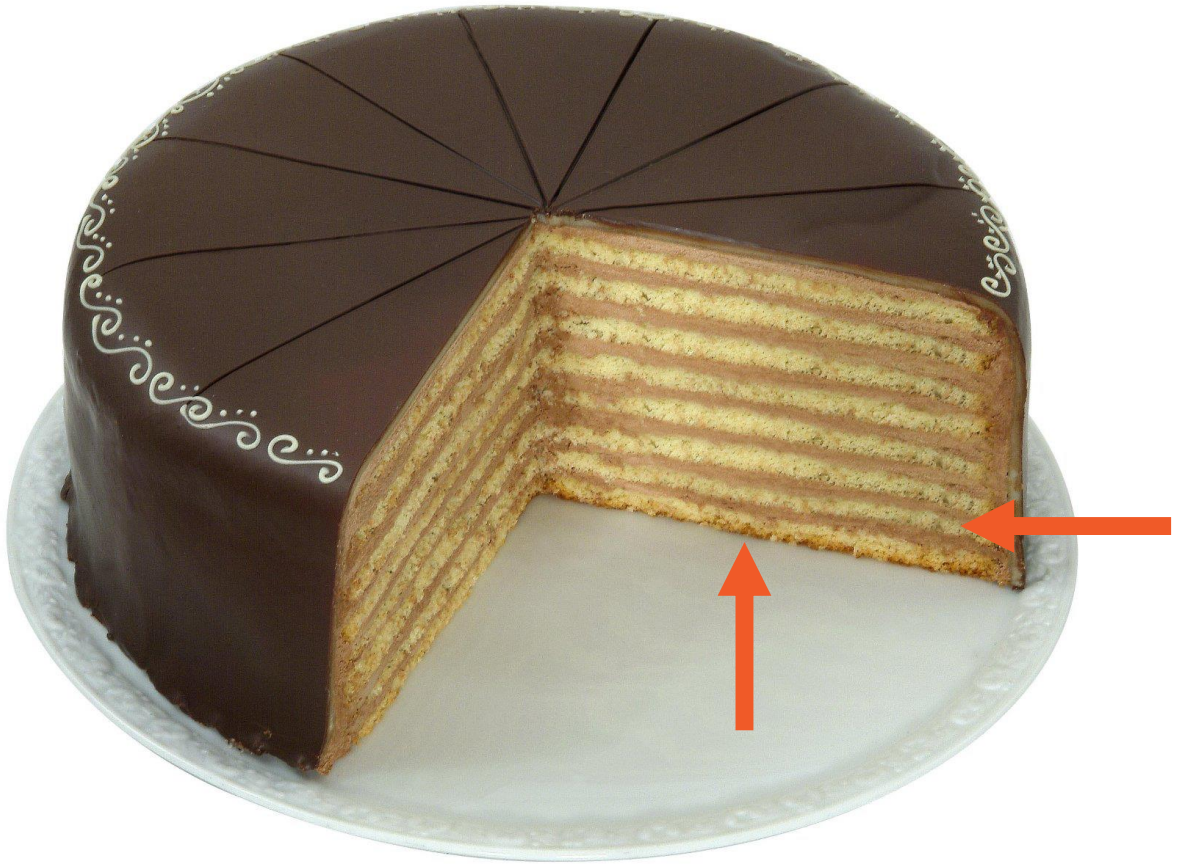
# Answer

1. Create a container volume that points to the source code.

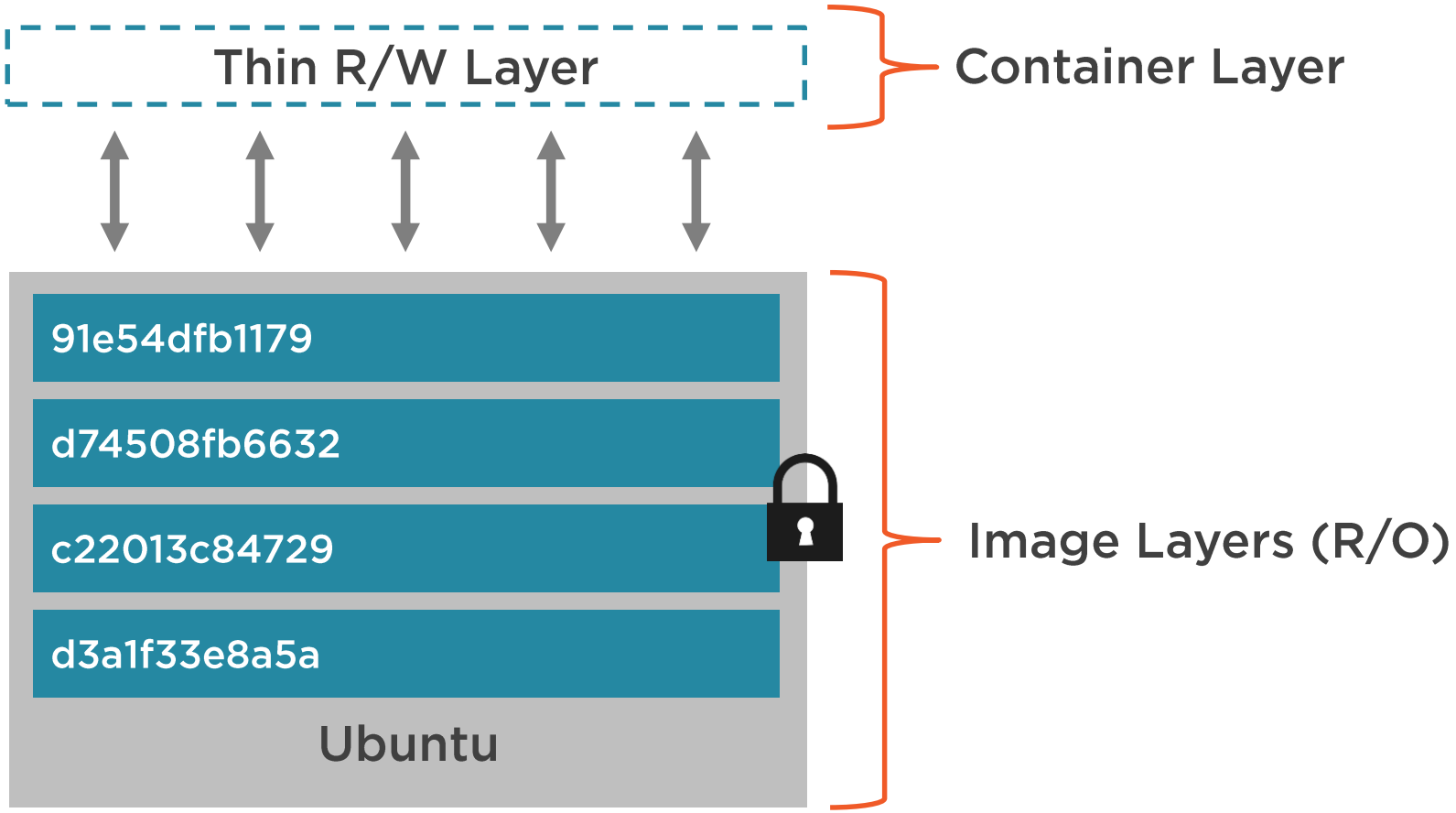
2. Add your source code into a custom image that is used to create a container.

# The Layered File System

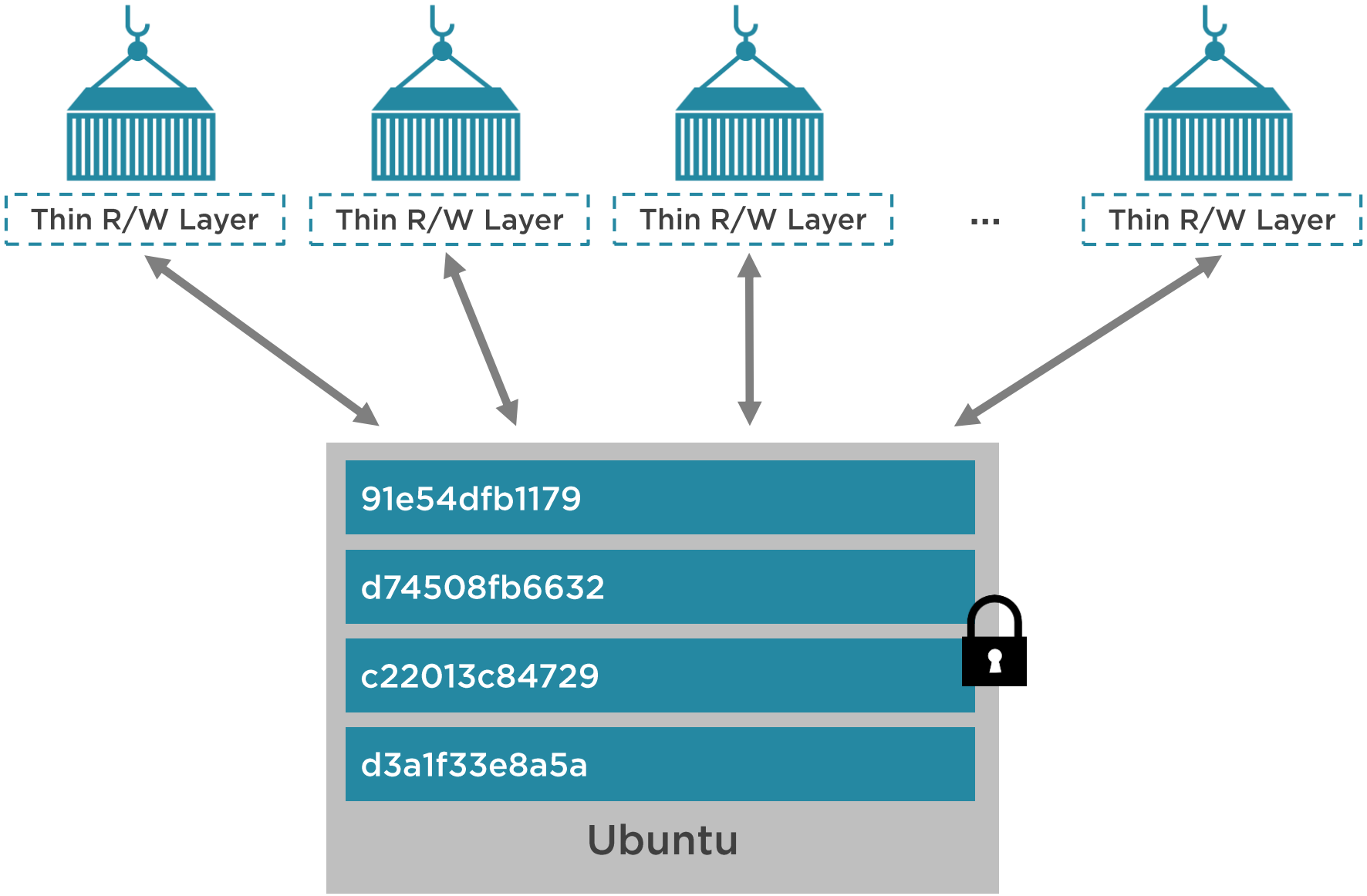
# Layers (from a dessert perspective)



# Images, Containers and File Layers

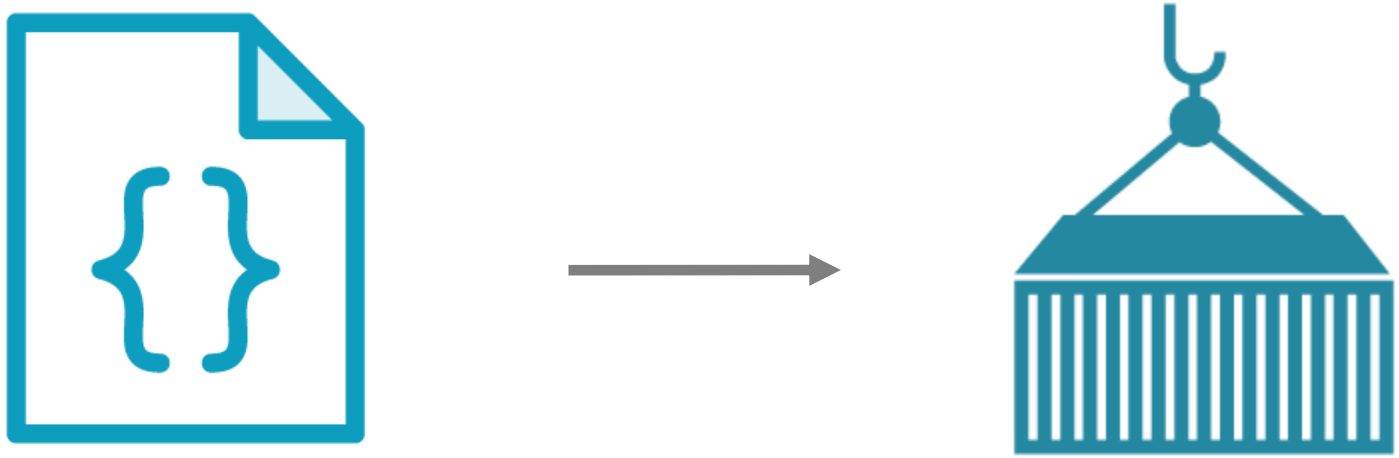


# Containers Can Share Image Layers



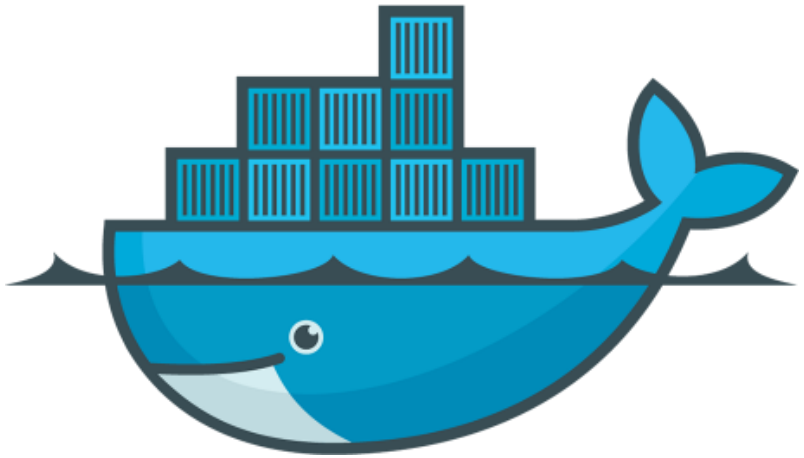
## Question

How do you get source code into a container?



# Containers and Volumes

# Docker Volumes



What is a Volume?

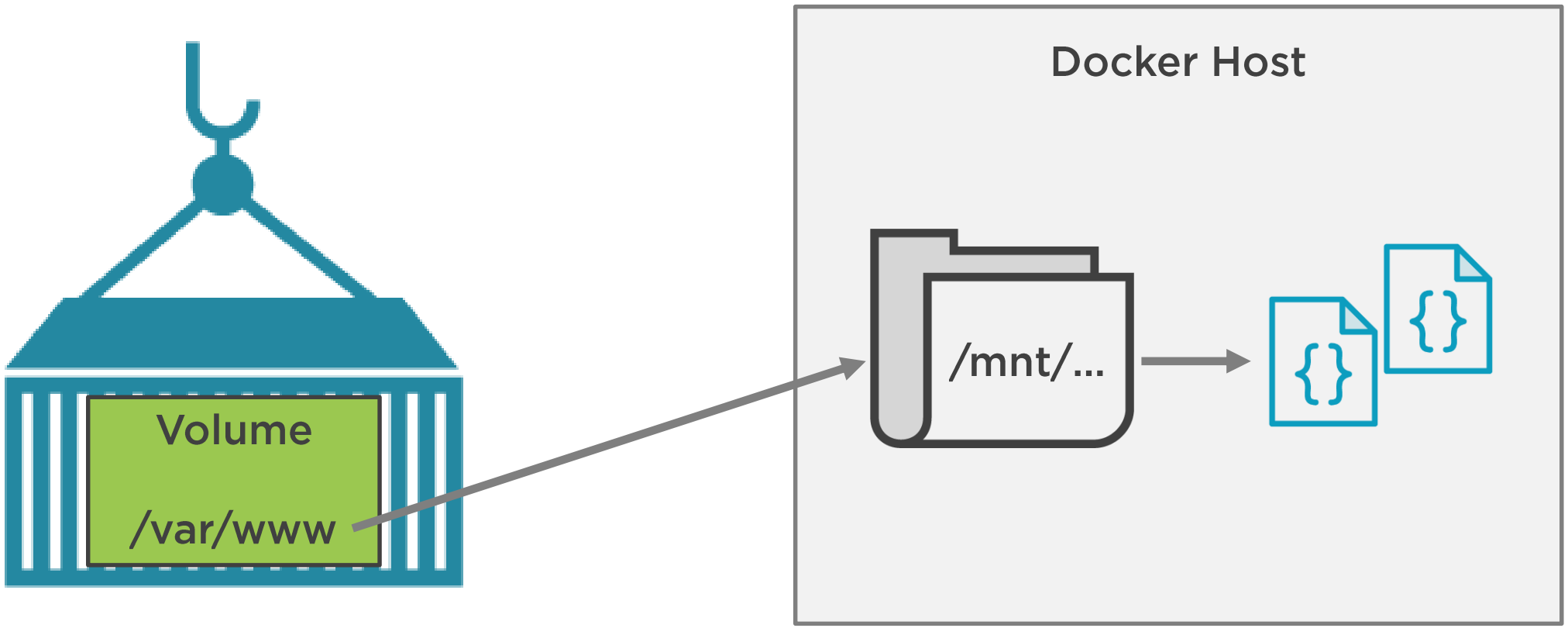
Special type of directory in a container typically referred to as a "data volume"

Can be shared and reused among containers

Updates to an image won't affect a data volume

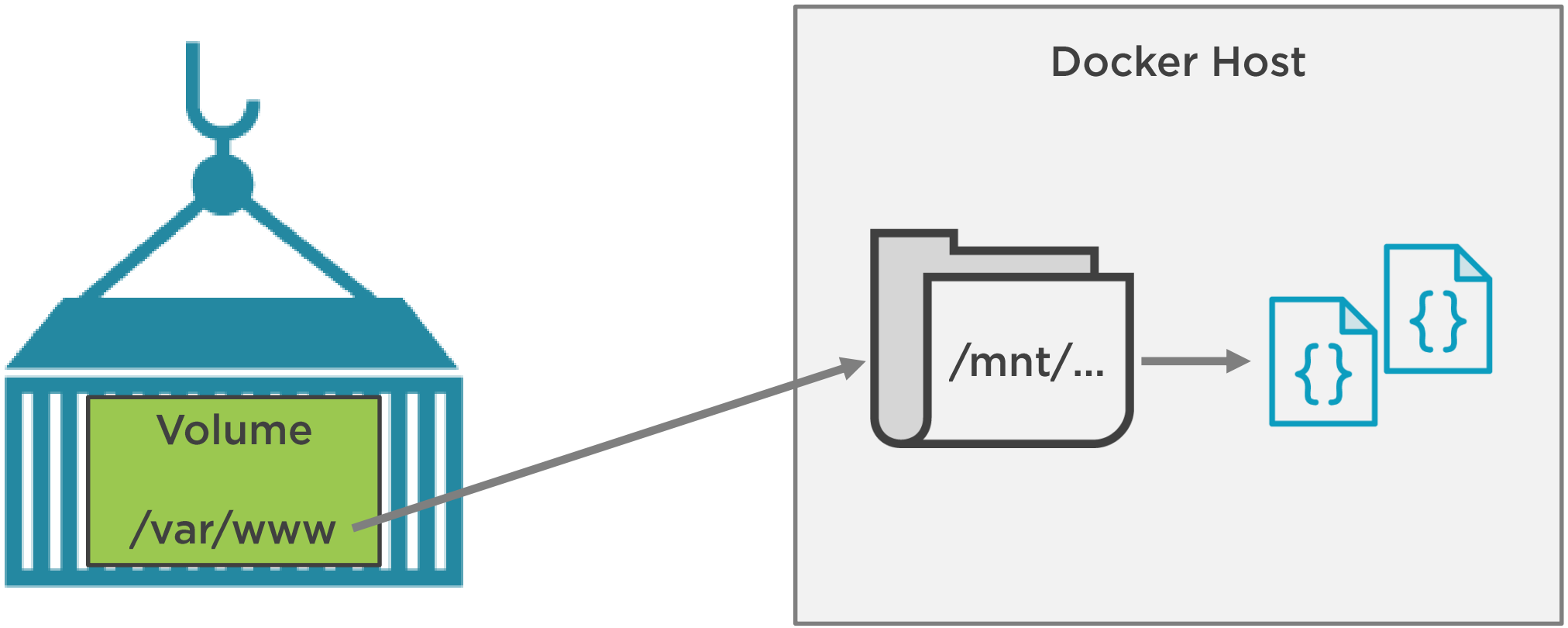
Data volumes are persisted even after the container is deleted

# Volume Overview

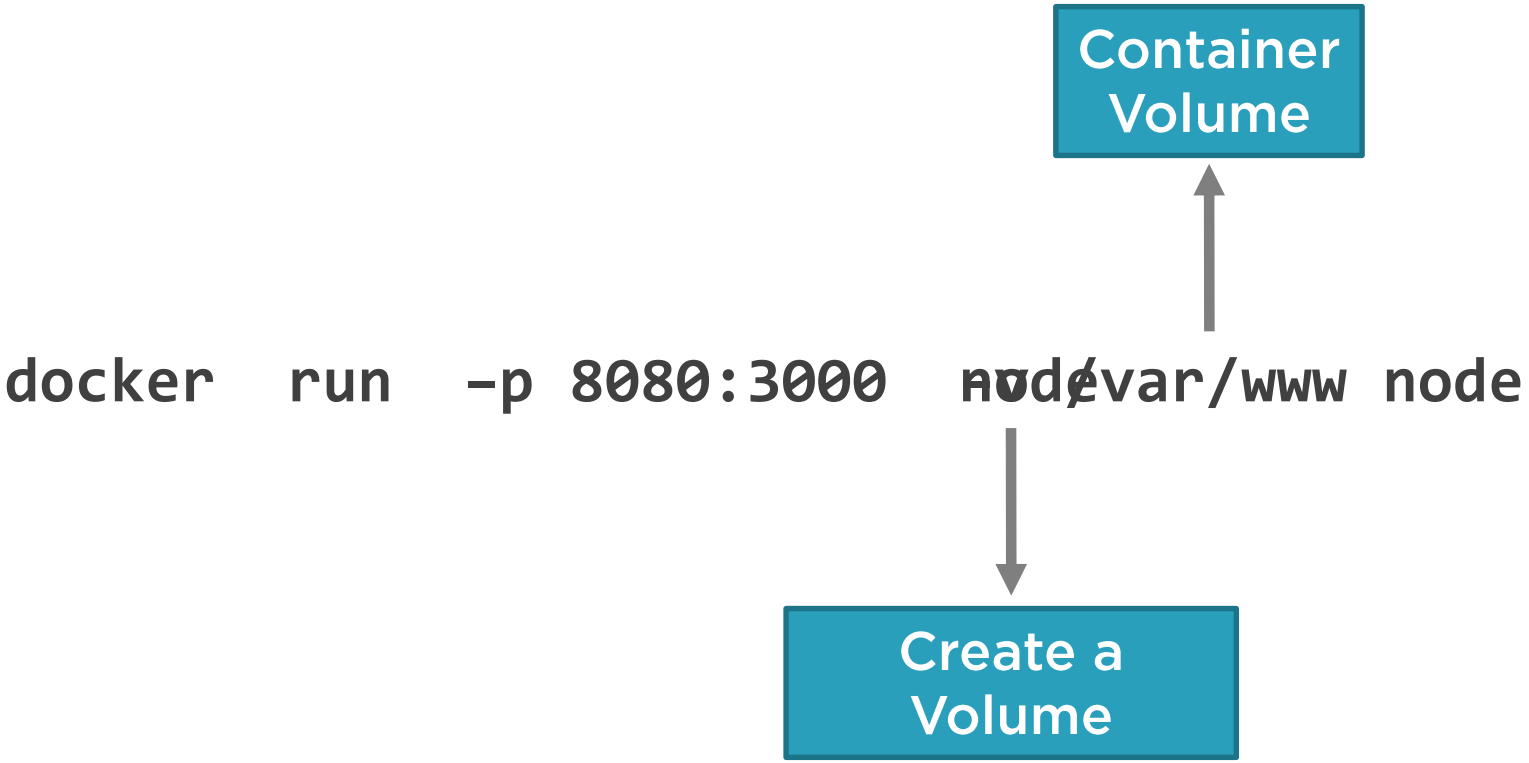


# Source Code, Volumes and Containers

# Understanding Volumes

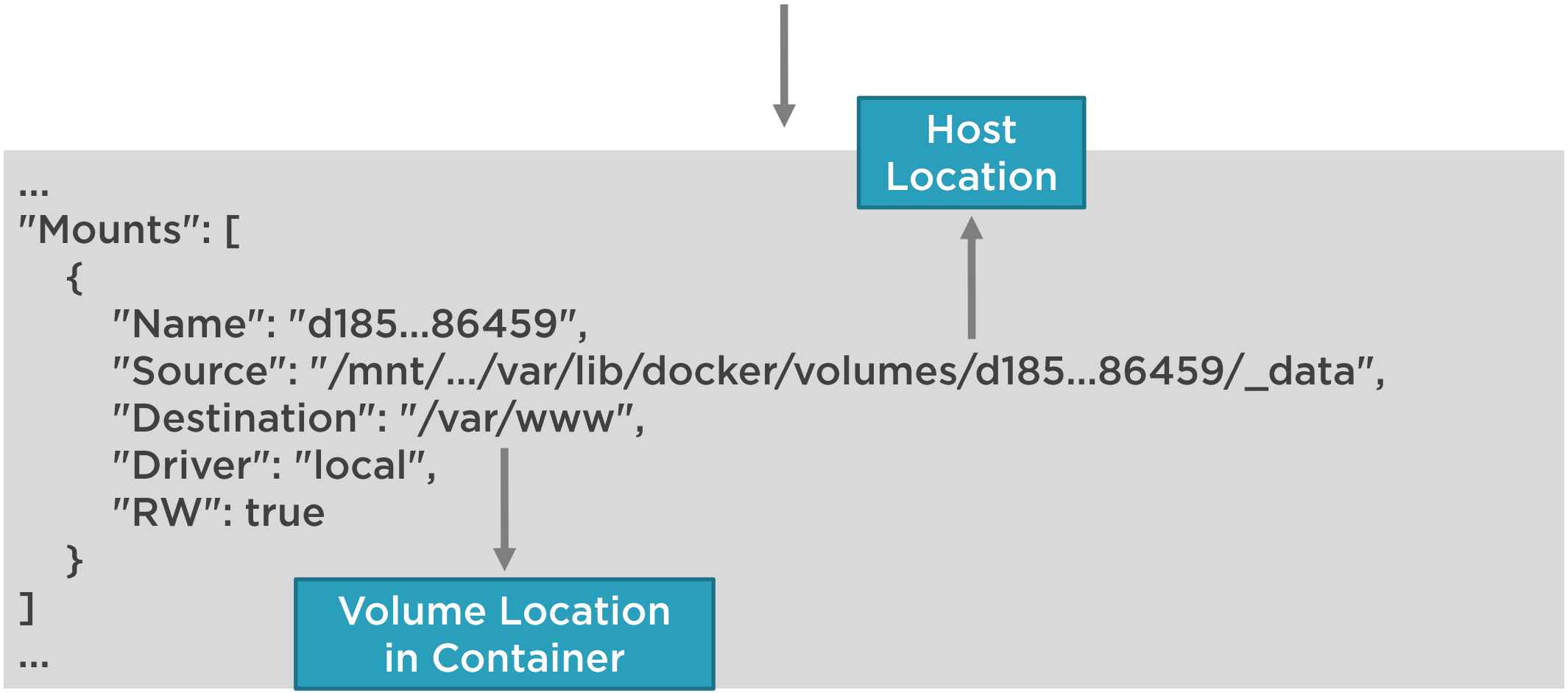


# Creating a Data Volume

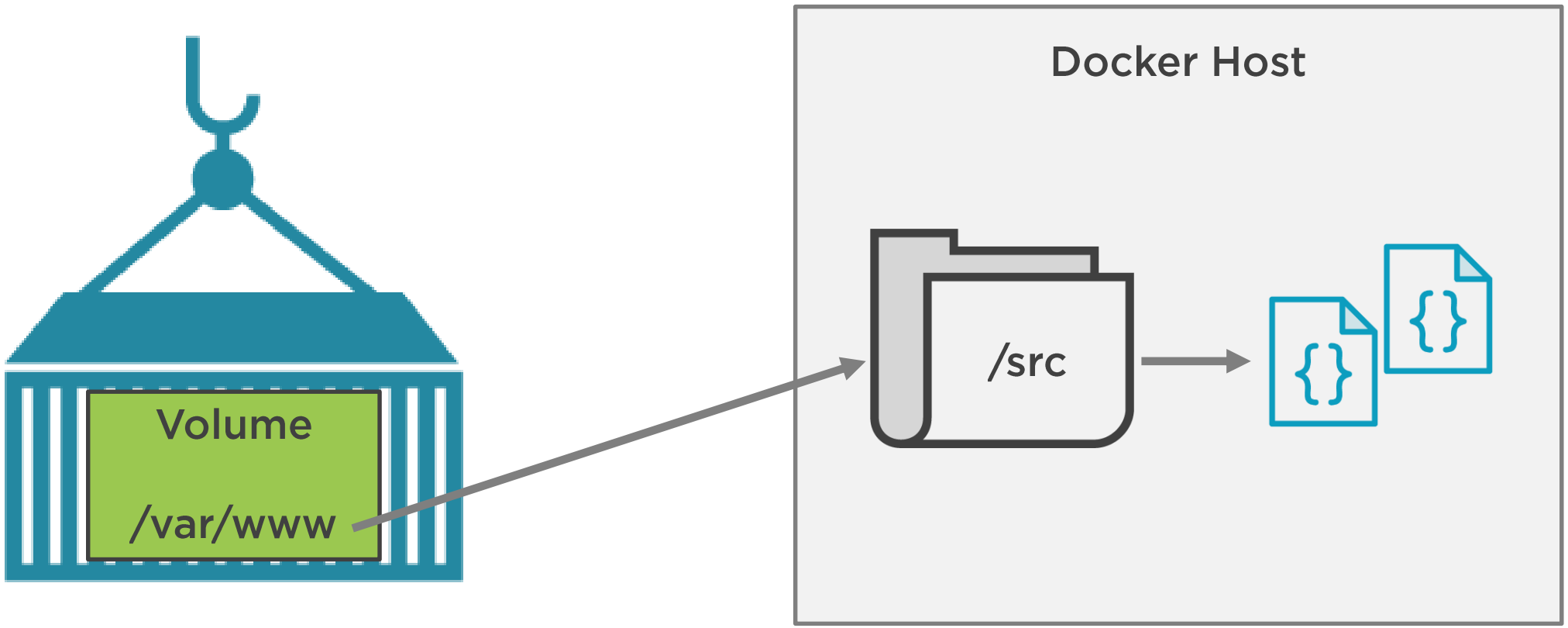


# Locating a Volume

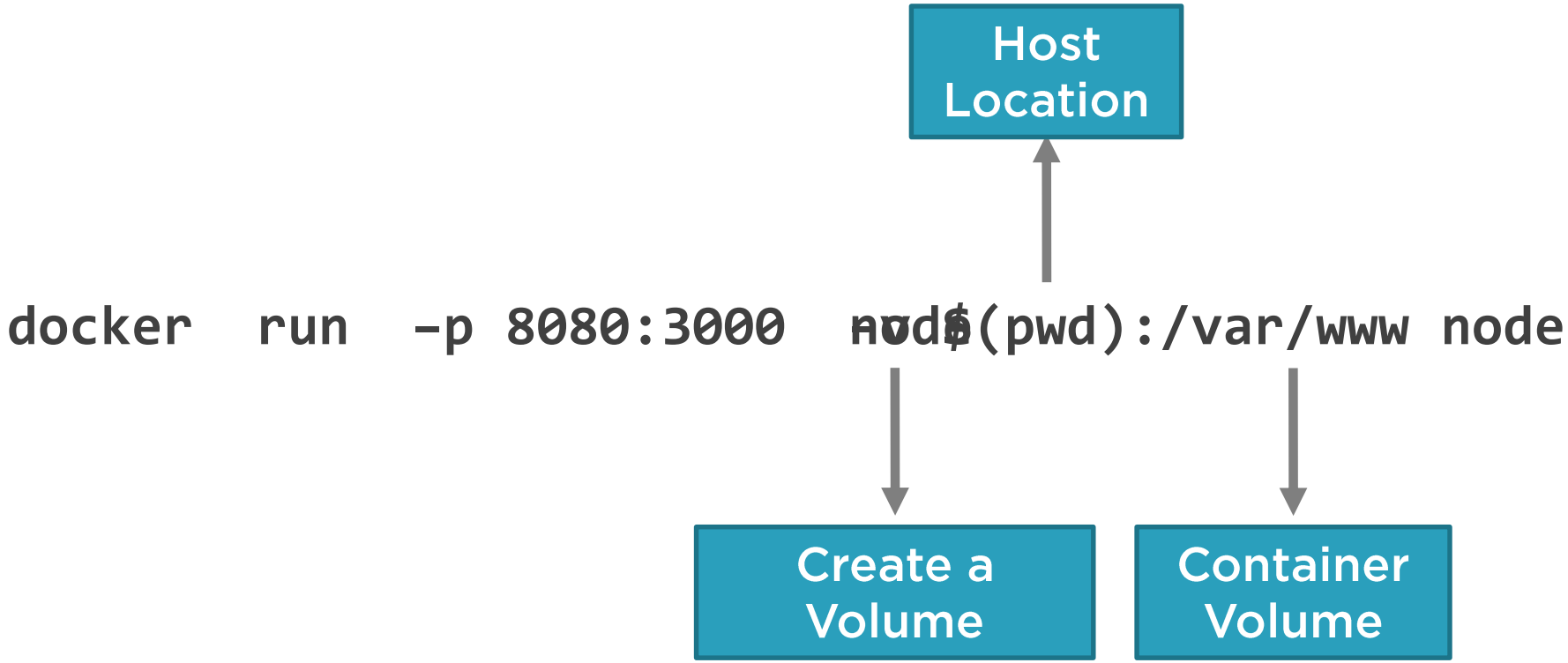
```
docker inspect mycontainer
```



# Customizing Volumes

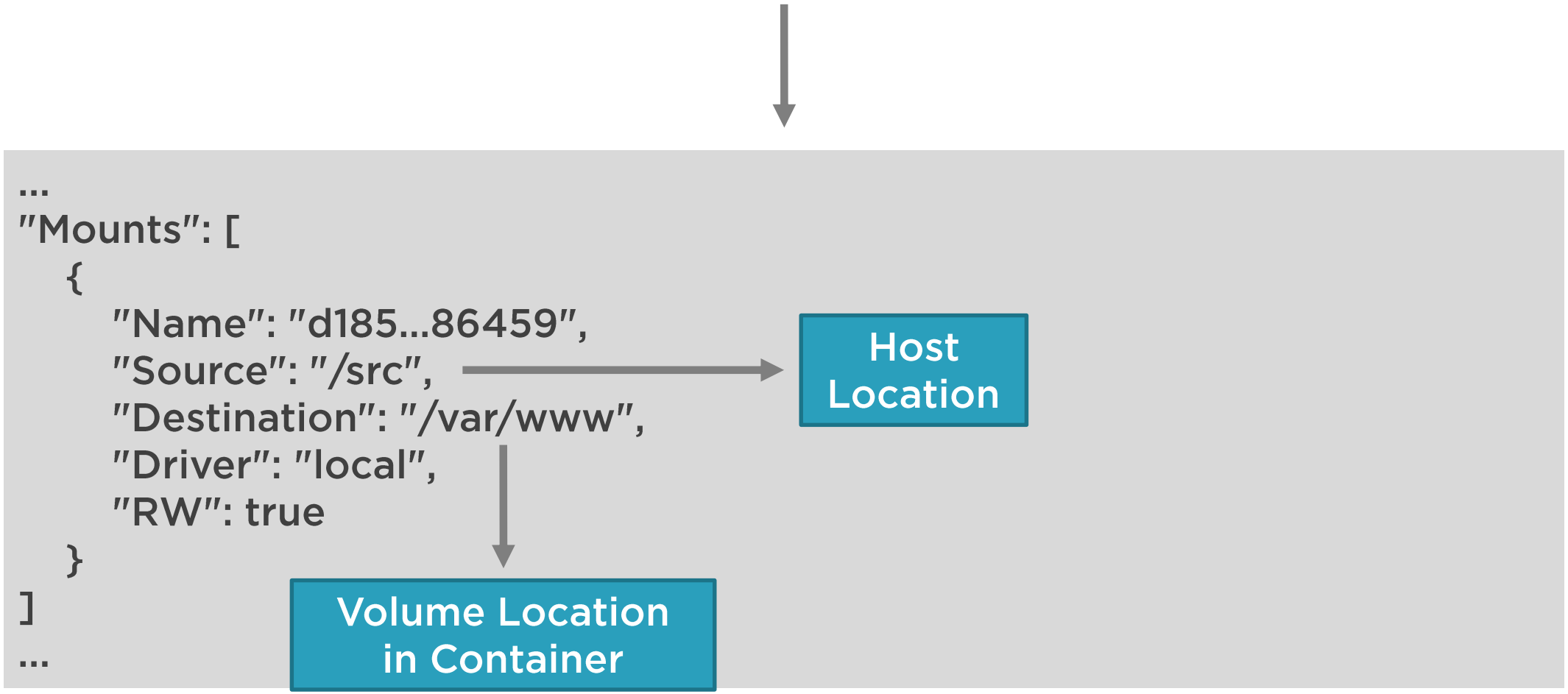


# Customizing the Host Location for a Data Volume



# Locating a Volume

```
docker inspect mycontainer
```



# Hooking a Volume to Node.js Source Code

# Hooking a Volume to ASP.NET Source Code

# Removing Containers and Volumes

# Creating a Docker Managed Volume

```
docker  run  -p 8080:3000  -v /var/www node
```

# Docker Controlled Volumes

```
docker inspect mycontainer
```

```
...
"Mounts": [
    {
        "Name": "d185...86459",
        "Source": "/mnt/.../var/lib/docker/volumes/d185...86459/_data",
        "Destination": "/var/www",
        "Driver": "local",
        "RW": true
    }
]
...
```

# Removing Volumes

```
docker rm -v lastContainer
```

# Summary

Docker images and containers rely on a "layered file system"

Source Code can be hooked to a container using volumes:
docker run –v

Volumes are persisted on the Docker Host

Volumes can be removed using:
docker rm –v lastContainer

# Building Custom Images with Dockerfile

**Dan Wahlin**
WAHLIN CONSULTING

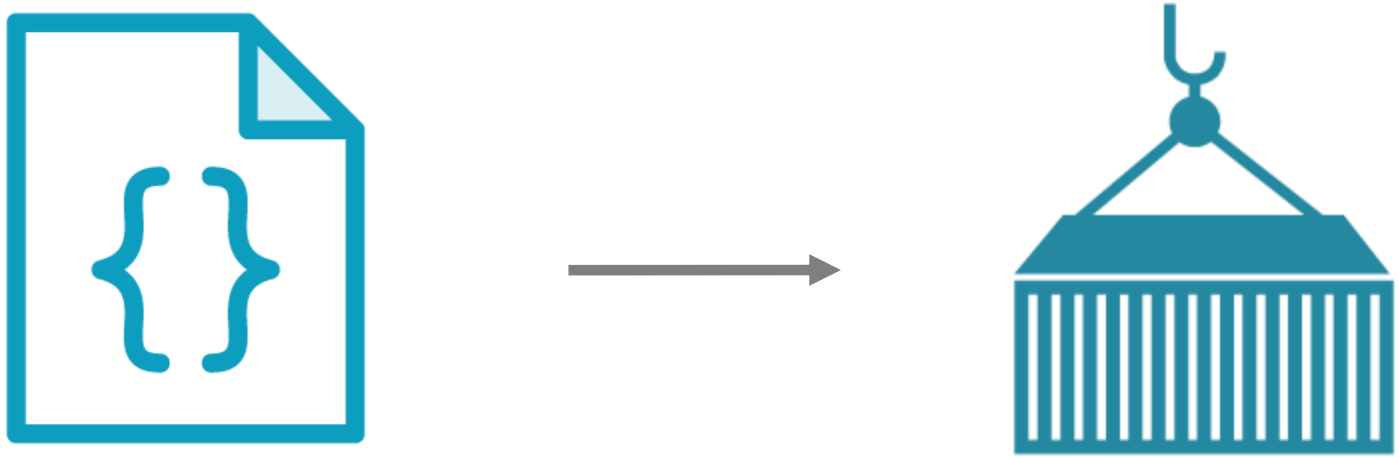
@danwahlin www.codewithdan.com

# Module Agenda

- Getting Started with Dockerfile
- Creating a Custom DockerFile
- Building a Custom Image
- Publishing an Image to Docker Hub

## Question

How do you get source code into a container?



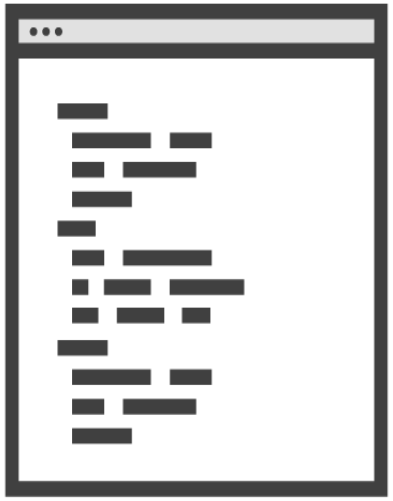
# Answer

1. Create a container volume that points to the source code.

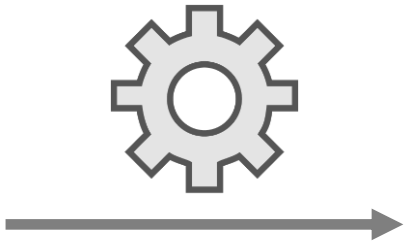
2. Add your source code into a custom image that is used to create a container.

# Getting Started with Dockerfile

# Dockerfile and Images



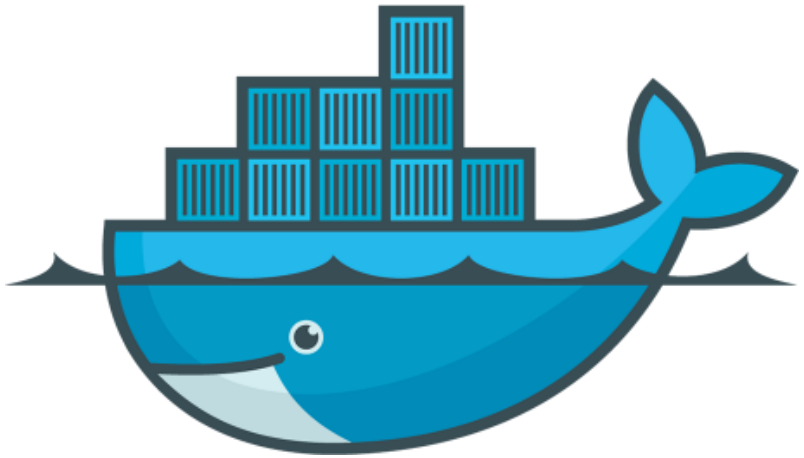
Dockerfile



docker build

Docker Image

# Dockerfile Overview



Text file used to build Docker images

Contains build instructions

Instructions create intermediate image that can be cached to speed up future builds

Used with "docker build" command

# Key Dockerfile Instructions

| | | |
|---|---|---|
| FROM | MAINTAINER | RUN |
| COPY | ENTRYPOINT | WORKDIR |
| EXPOSE | ENV | VOLUME |

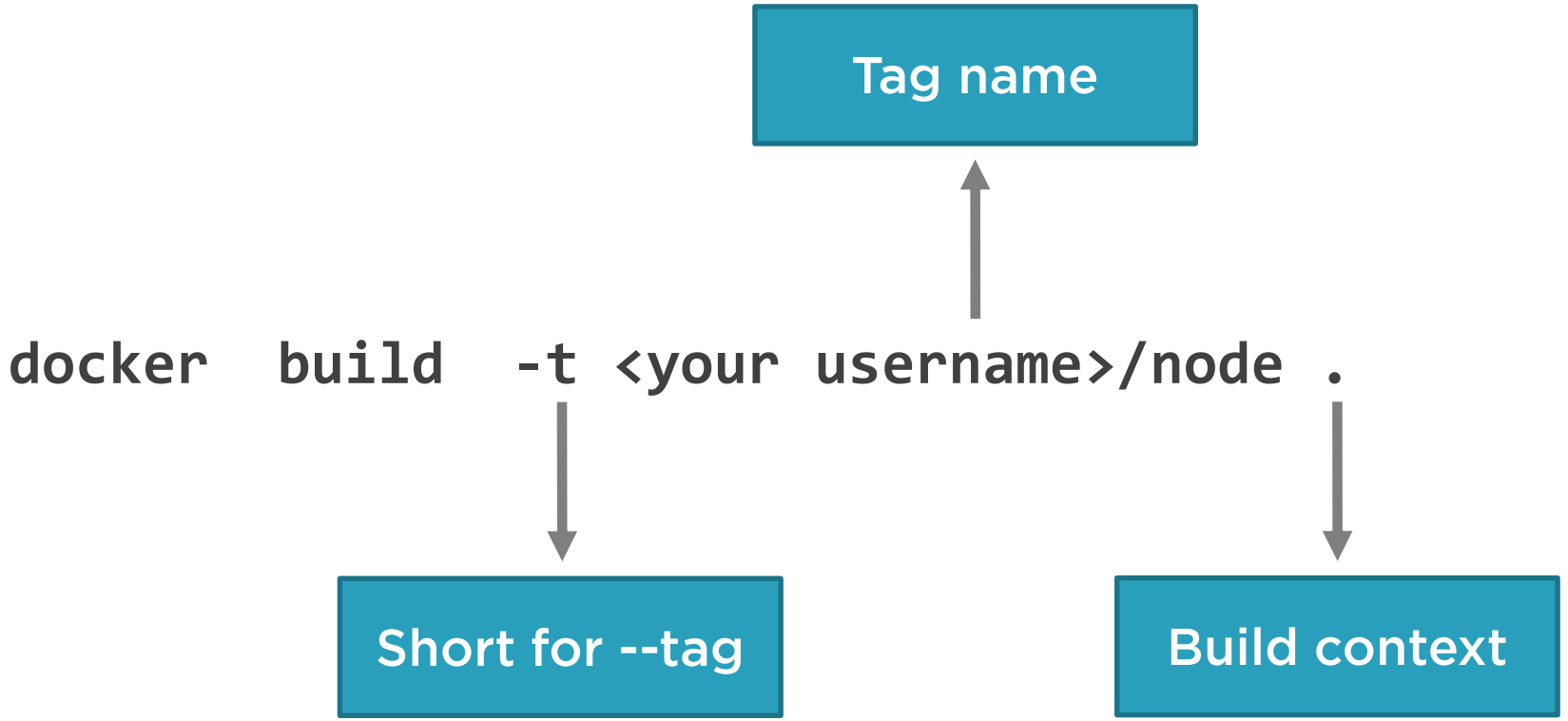
# Dockerfile Example

```
FROM          node
MAINTAINER    Dan Wahlin
COPY          .  /var/www
WORKDIR       /var/www
RUN           npm install
EXPOSE        8080
ENTRYPOINT    ["node", "server.js"]
```

# Creating a Custom Node.js Dockerfile

# Building a Node.js Image

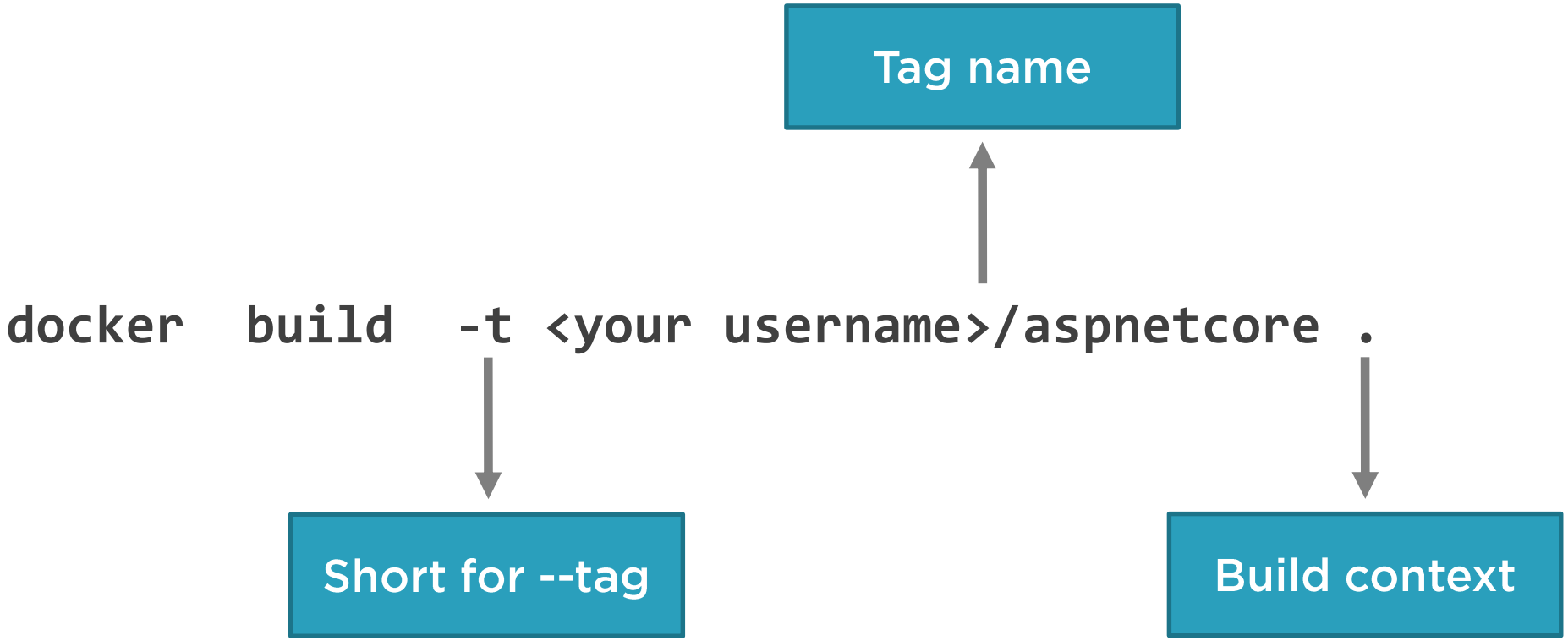
# Building a Custom Image



# Creating a Custom ASP.NET Core Dockerfile

# Building an ASP.NET Core Image

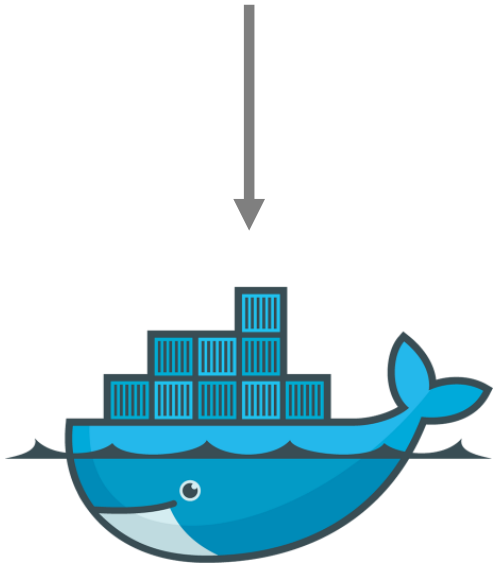
# Building a Custom Image



# Publishing an Image to Docker Hub

# Publishing an Image to Docker Hub

```
docker  push  <your username>/node
```



Docker Registry

# Summary

Dockerfile is a simple text file with instructions that is used to create an image

Each Dockerfile starts with a FROM instruction

Custom images are built using:
docker build –t <username>/imageName

Images can be pushed to Docker Hub

# Communicating Between Docker Containers

**Dan Wahlin**
WAHLIN CONSULTING

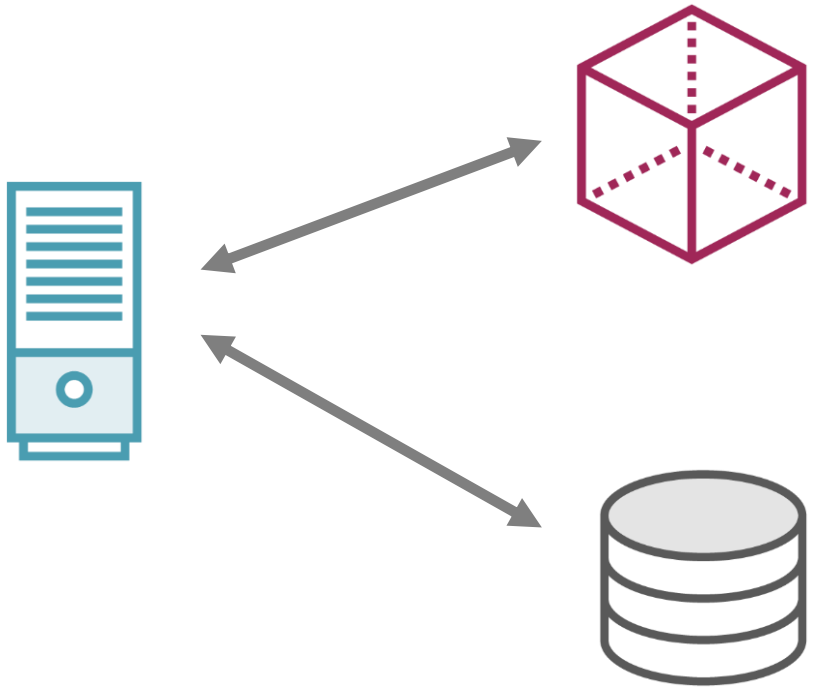
@danwahlin www.codewithdan.com

# Module Agenda

- Getting Started with Container Linking
- Linking Containers by Name (legacy linking)
- Container Linking in Action
- Getting Started with Container Networks
- Container Networks in Action
- Linking Multiple Containers

# Getting Started with Container Linking

# The Need for Linked Containers



# Docker Container Linking Options

Use Legacy Linking

Add Containers to a Bridge Network

# Linking Containers by Name (legacy linking)

# Steps to Link Containers

Run a Container with a Name

1

Link to Running Container by Name

2

Repeat for Additional Containers

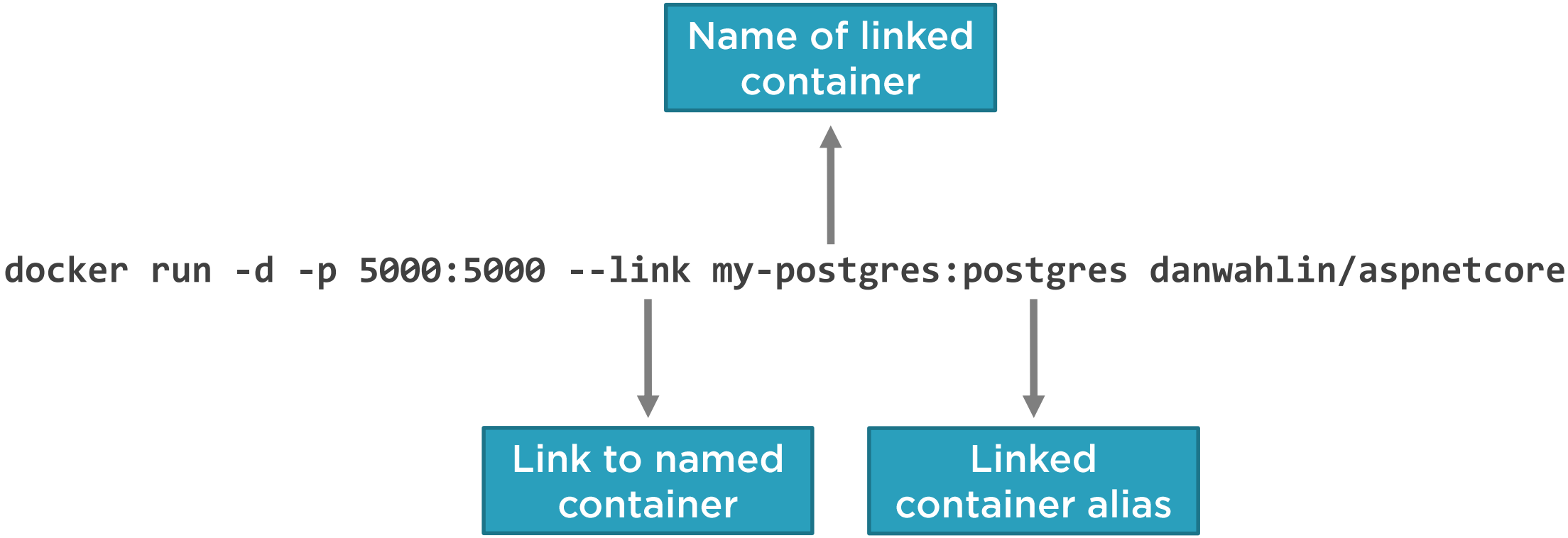
3

# 1 Run a Container with a Name

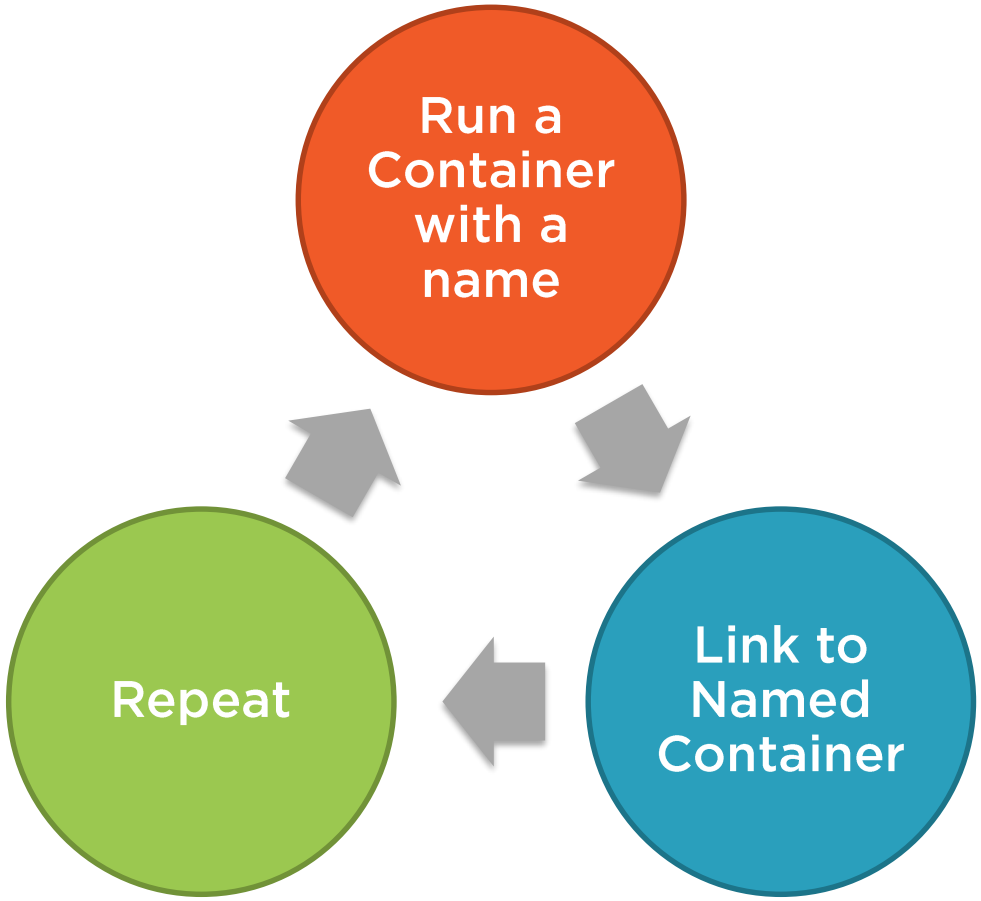
```
docker run -d --name my-postgres postgres
```

Define a name for the container

# 2 Link to Running Container By Name



# 3 Repeat for Additional Containers

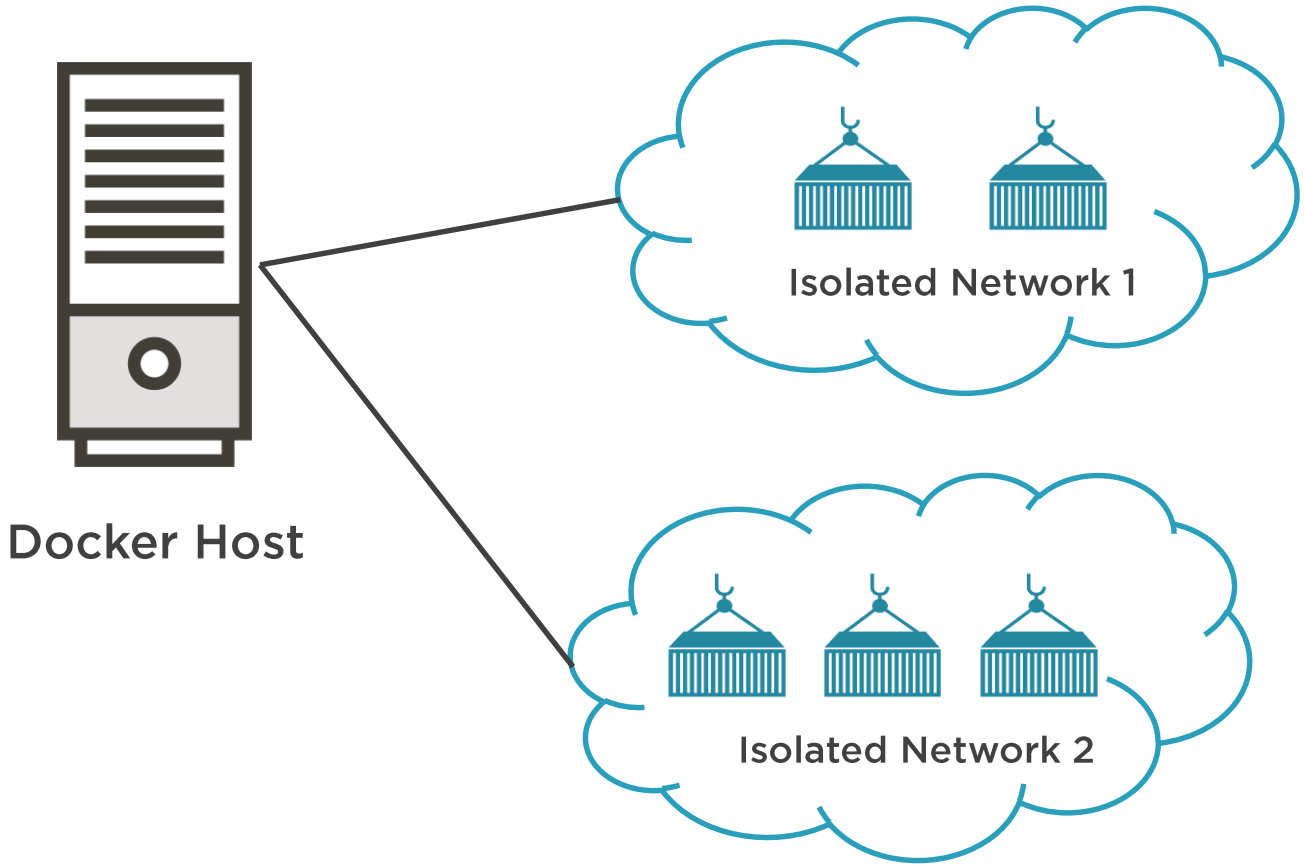


# Linking Node.js and MongoDB Containers

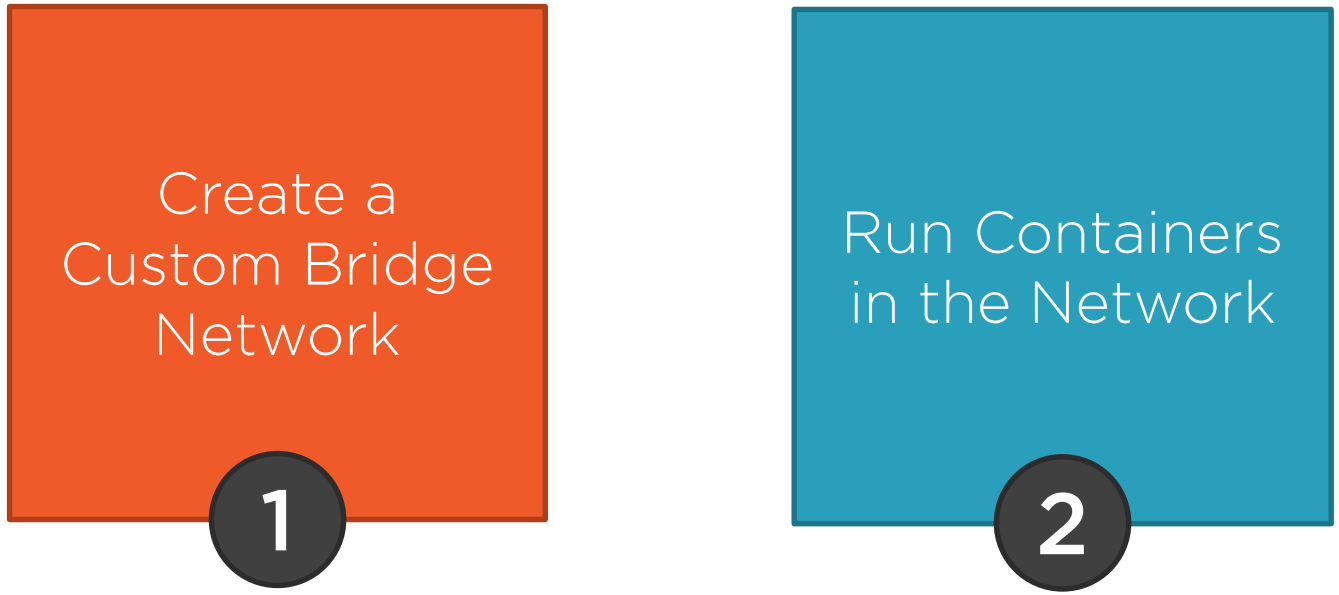
# Linking ASP.NET Core and PostgreSQL Containers

# Getting Started with Container Networks

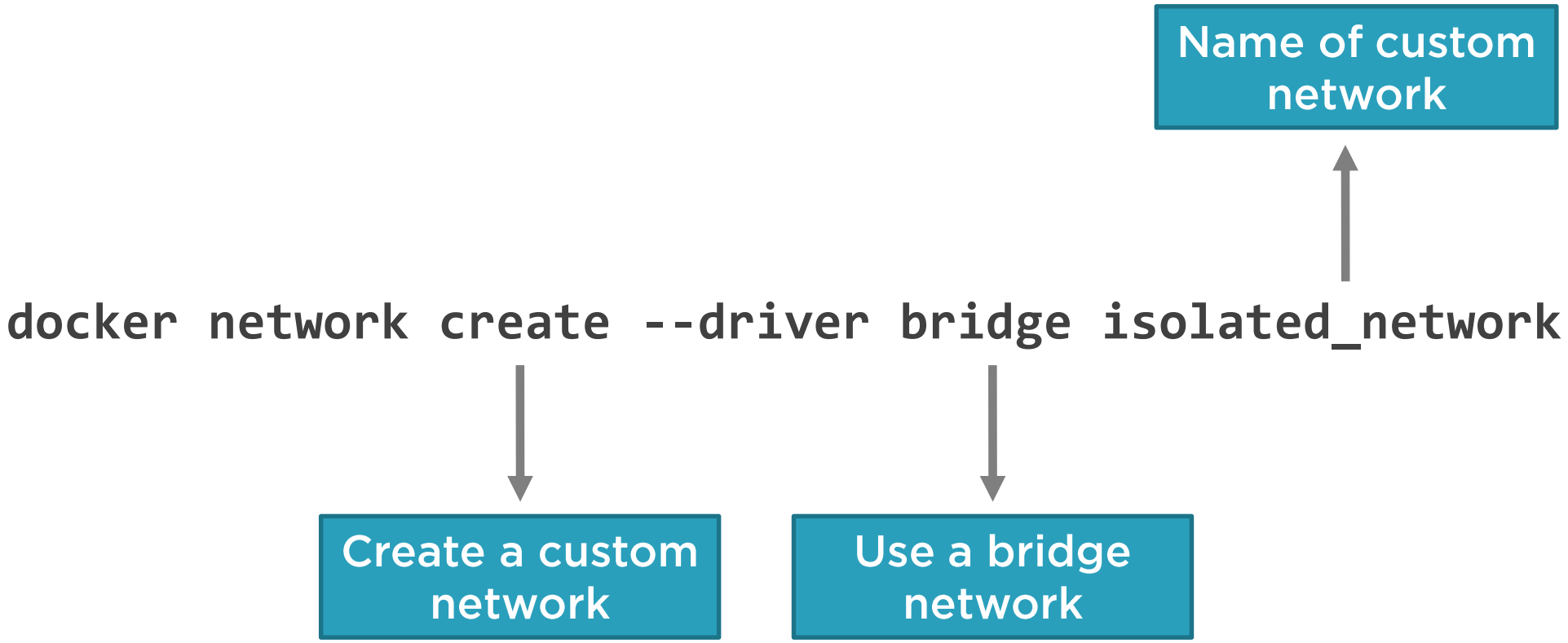
# Understanding Container Networks



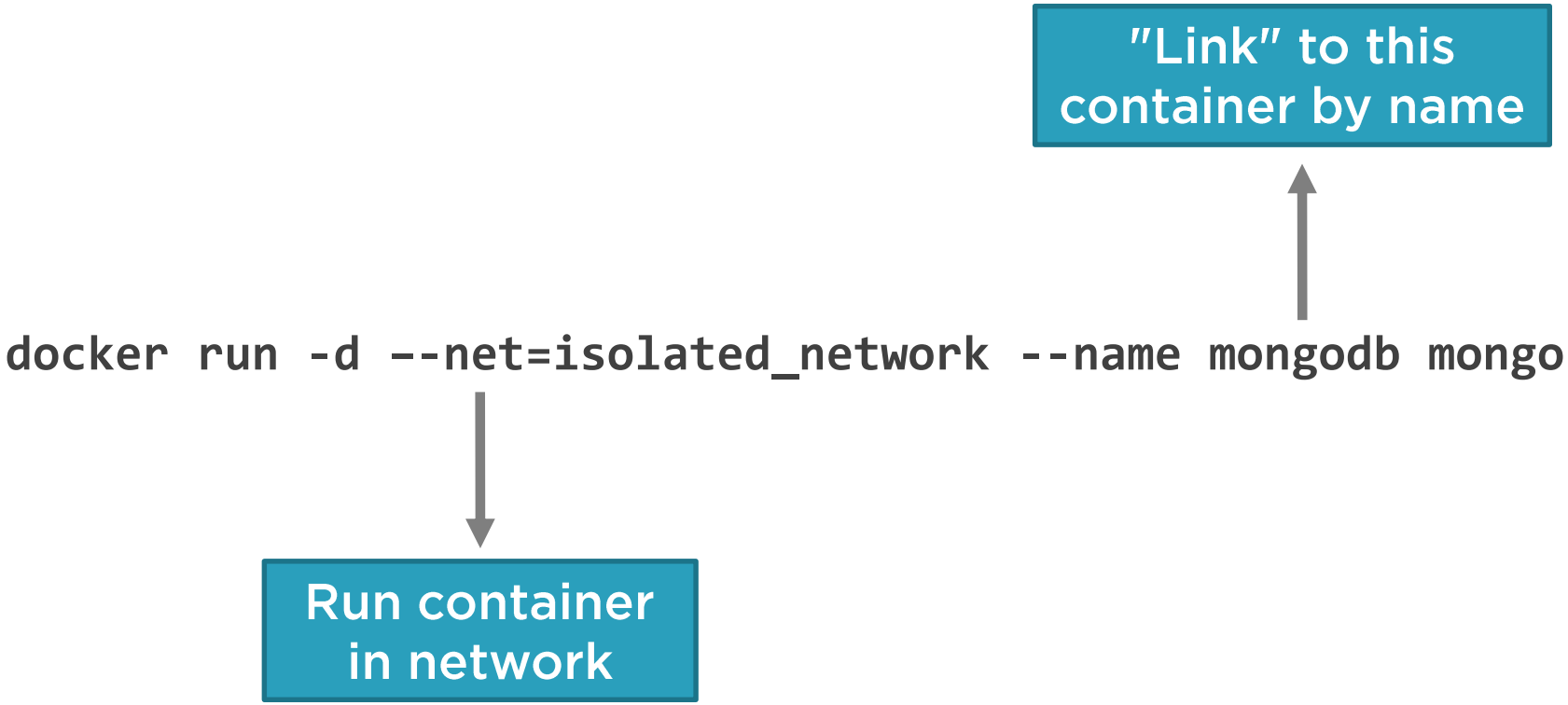
# Steps to Create a Container Network



# 1 Create a Custom Container Network



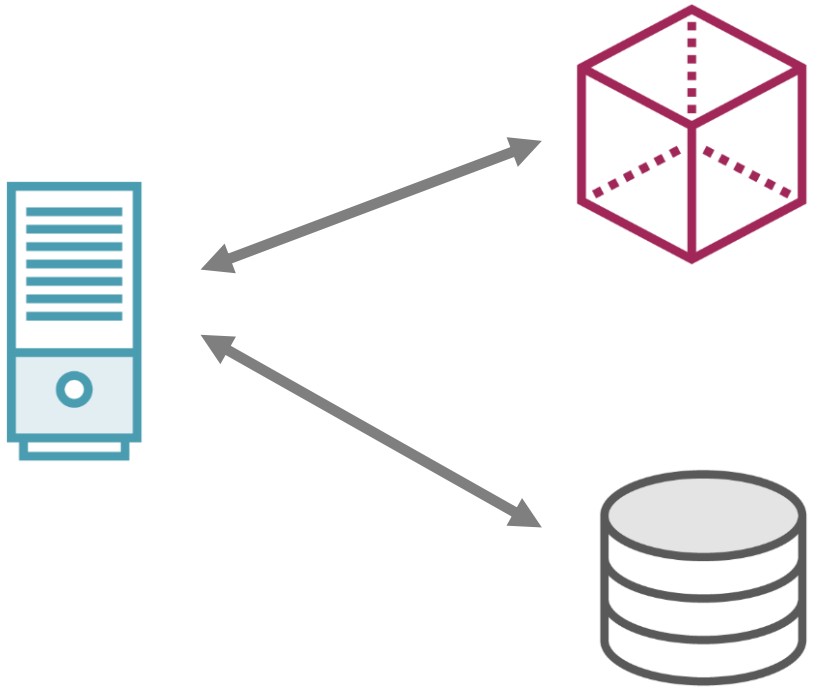
# 2 Run Containers in the Container Network



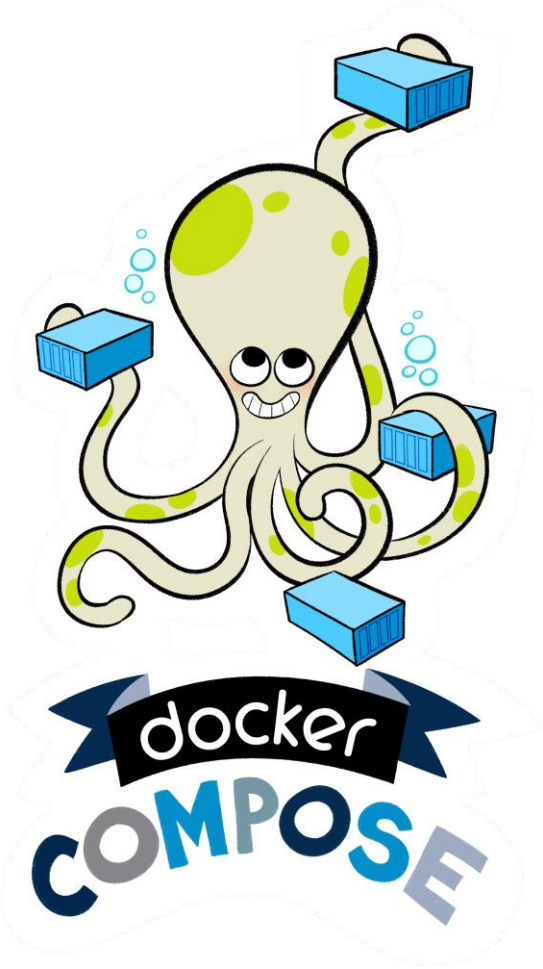
# Container Networks in Action

# Linking Multiple Containers

# Is There an Easier Way?



# Docker Compose Can Simplify Container Linking



# Summary

Docker containers communicate using link or network functionality

The –-link switch provides "legacy linking"

The --net command-line switch can be used to setup a bridge network

Docker Compose can be used to link multiple containers to each other

# Managing Containers with Docker Compose

**Dan Wahlin**
WAHLIN CONSULTING

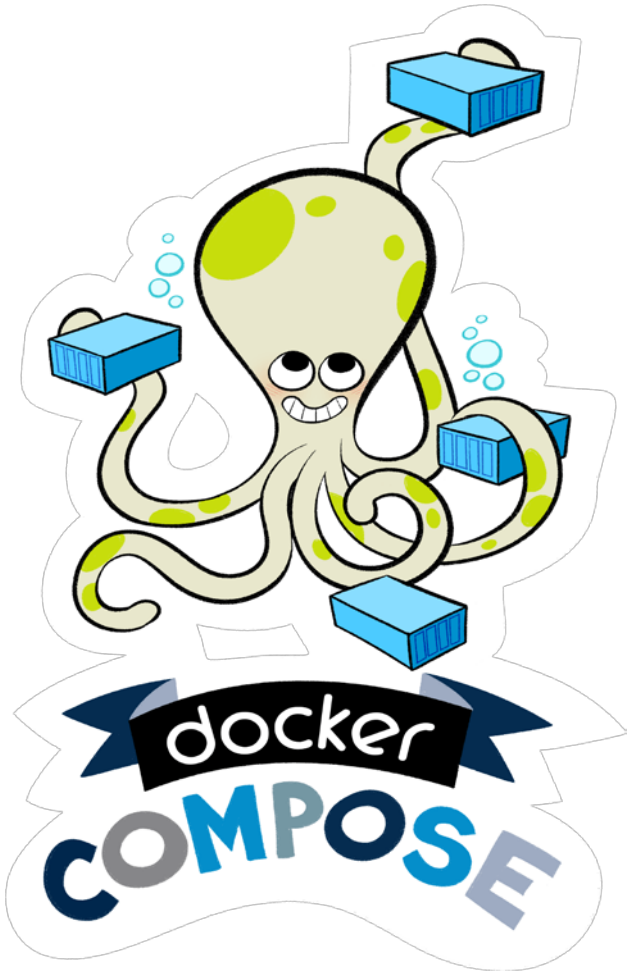
@danwahlin www.codewithdan.com

# Module Agenda

- Getting Started with Docker Compose
- The docker-compose.yml File
- Docker Compose Commands
- Docker Compose in Action
- Setting Up Development Environment Services
- Creating a Custom docker-compose.yml File
- Managing Development Environment Services

# Getting Started with Docker Compose

# Docker Compose Manages Your Application Lifecycle

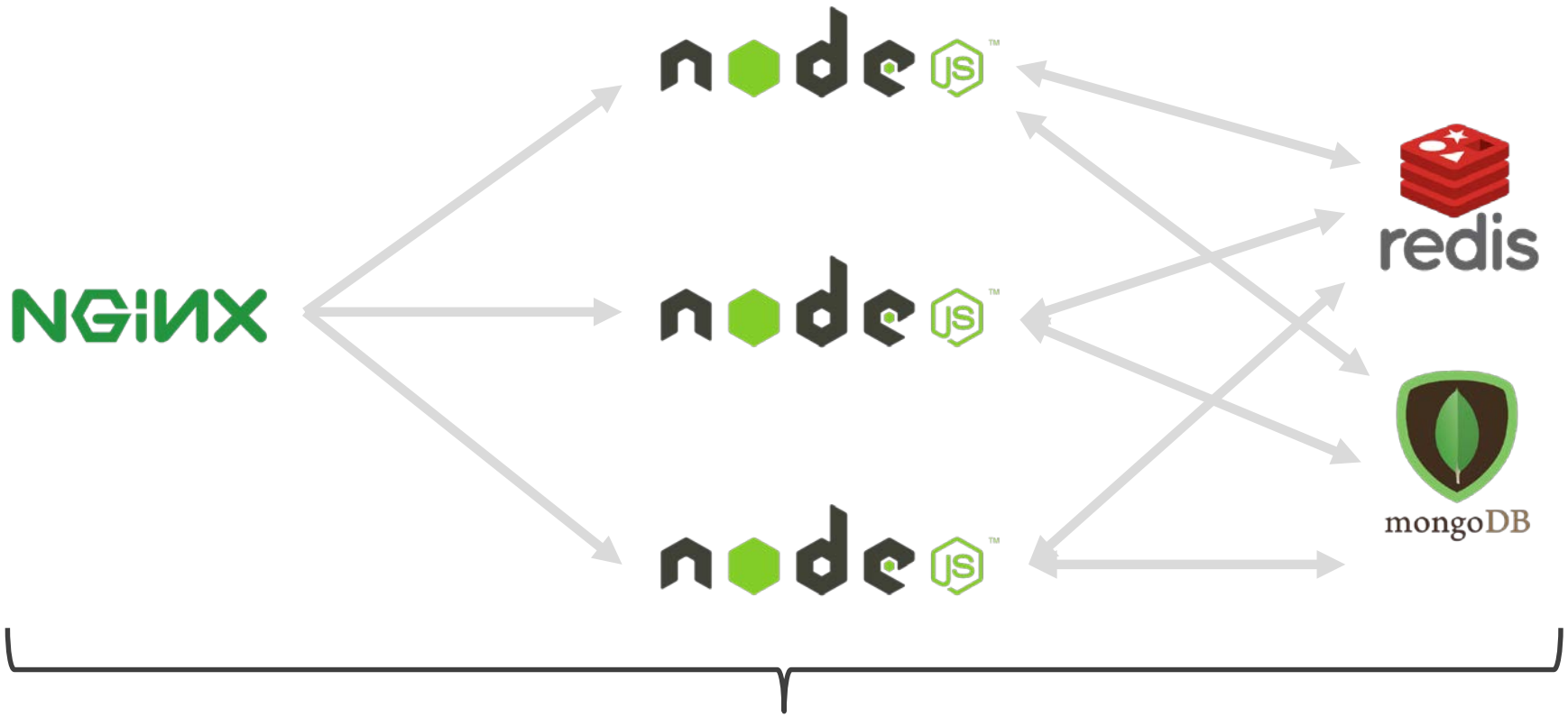


# Docker Compose Features


**Manages the whole application lifecycle:**

- Start, stop and rebuild services
- View the status of running services
- Stream the log output of running services
- Run a one-off command on a service

# The Need for Docker Compose

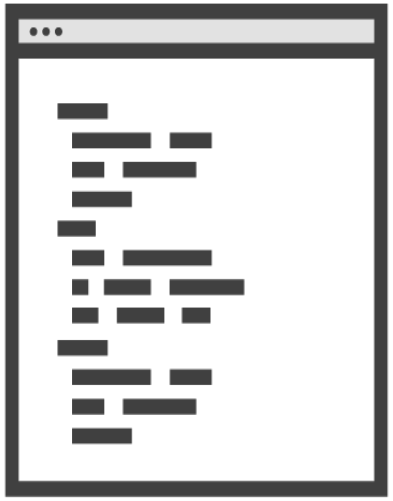



**Docker Compose**
**(docker-compose.yml)**

# Docker Compose Workflow

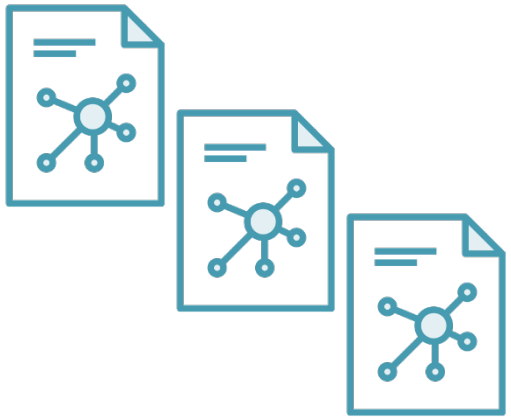
# The docker-compose.yml File

# The Role of the Docker Compose File



docker-compose.yml
(service configuration)

Docker Compose
Build



Docker Images
(services)

# Docker Compose and Services

version: '2'

services:


docker-compose.yml

# Key Service Configuration Options

| | | |
|---|---|---|
| **build** | **environment** | **image** |
| **networks** | **ports** | **volumes** |

# docker-compose.yml Example

```
version: '2'
services:
  node:
    build:
      context: .
      dockerfile: node.dockerfile
    networks:
      -nodeapp-network

  mongodb:
     image: mongo
     networks:
       - nodeapp-network

networks:
  nodeapp-network
    driver: bridge
```

# Docker Compose Commands

# Key Docker Compose Commands


docker-compose build

docker-compose up

docker-compose down

docker-compose logs

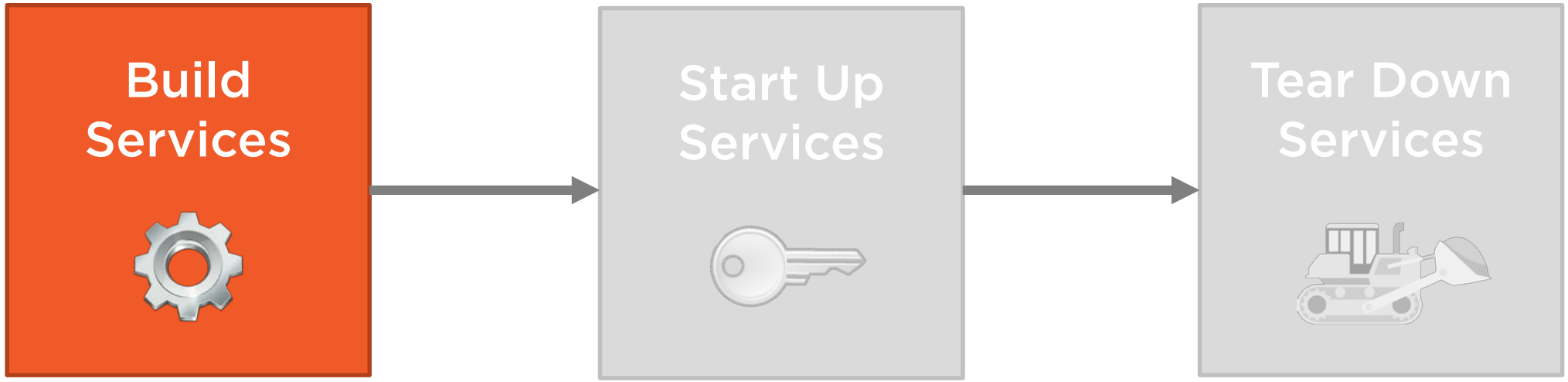
docker-compose ps

docker-compose stop

docker-compose start

docker-compose rm

# Building Services



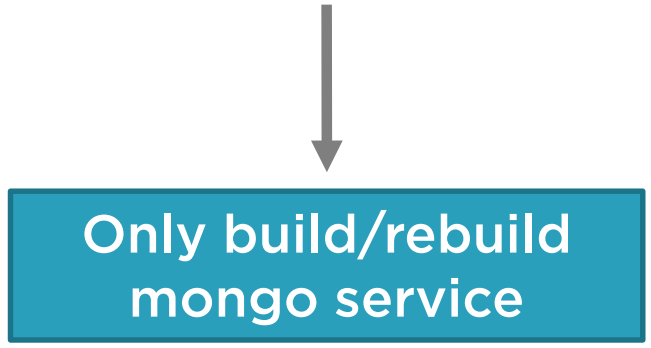
# Building Services

```
docker-compose build
```

Build or rebuild services defined in docker-compose.yml

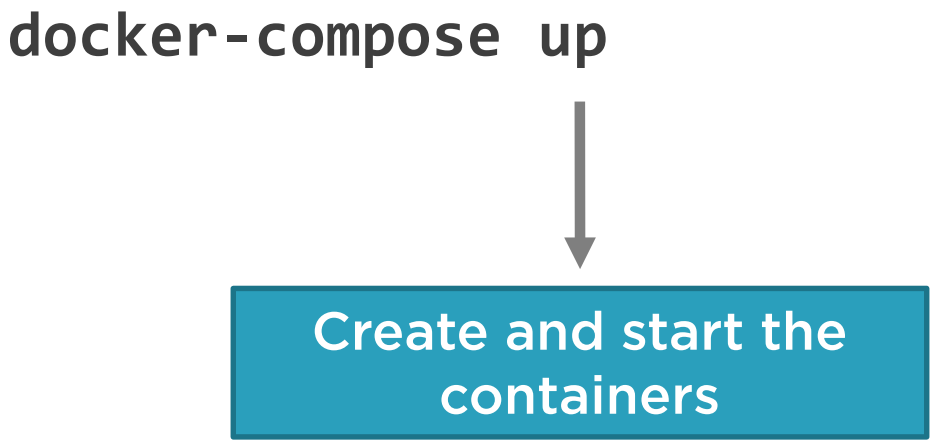
# Building Specific Services

```
docker-compose build mongo
```

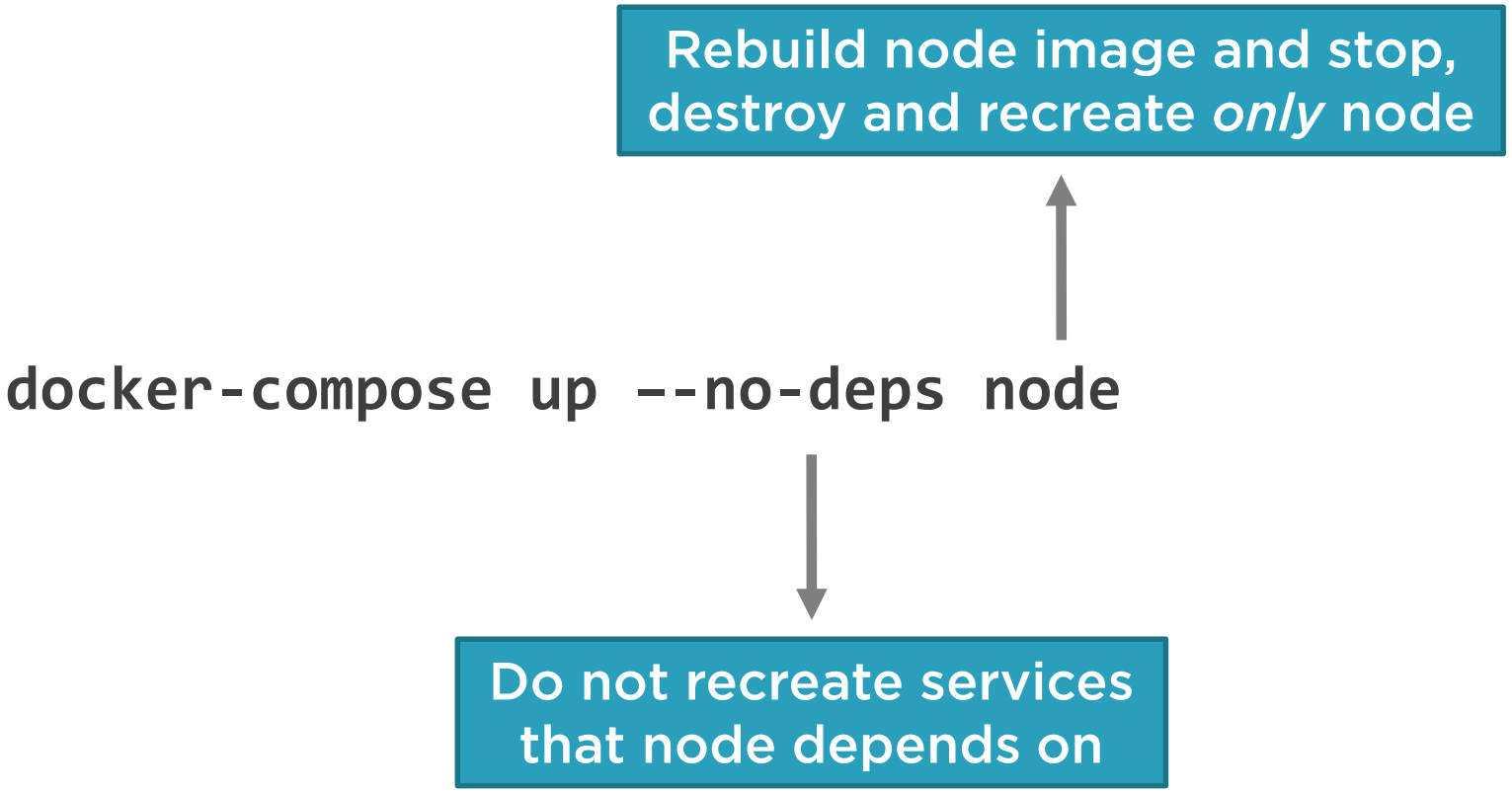


# Starting Services Up

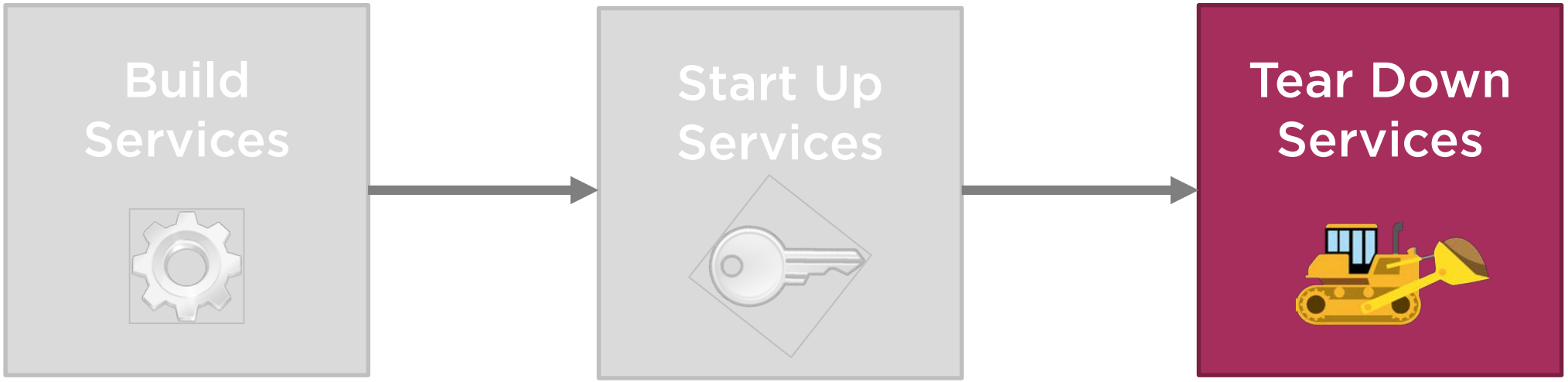
# Creating and Starting Containers



# Creating and Starting Containers



# Tearing Down Services

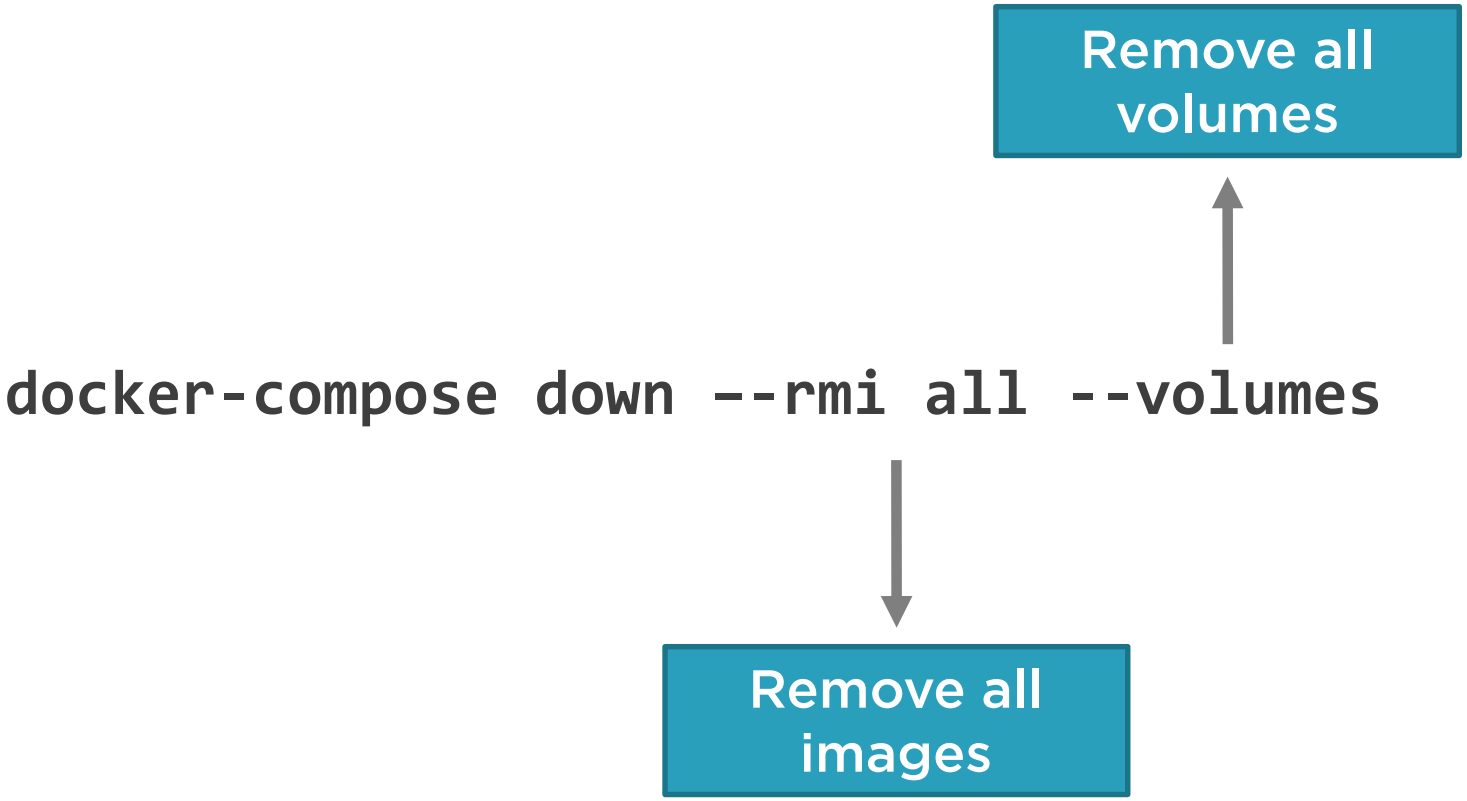


# Stop and Remove Containers

```
docker-compose down
```

Take all of the containers down (stop and remove)

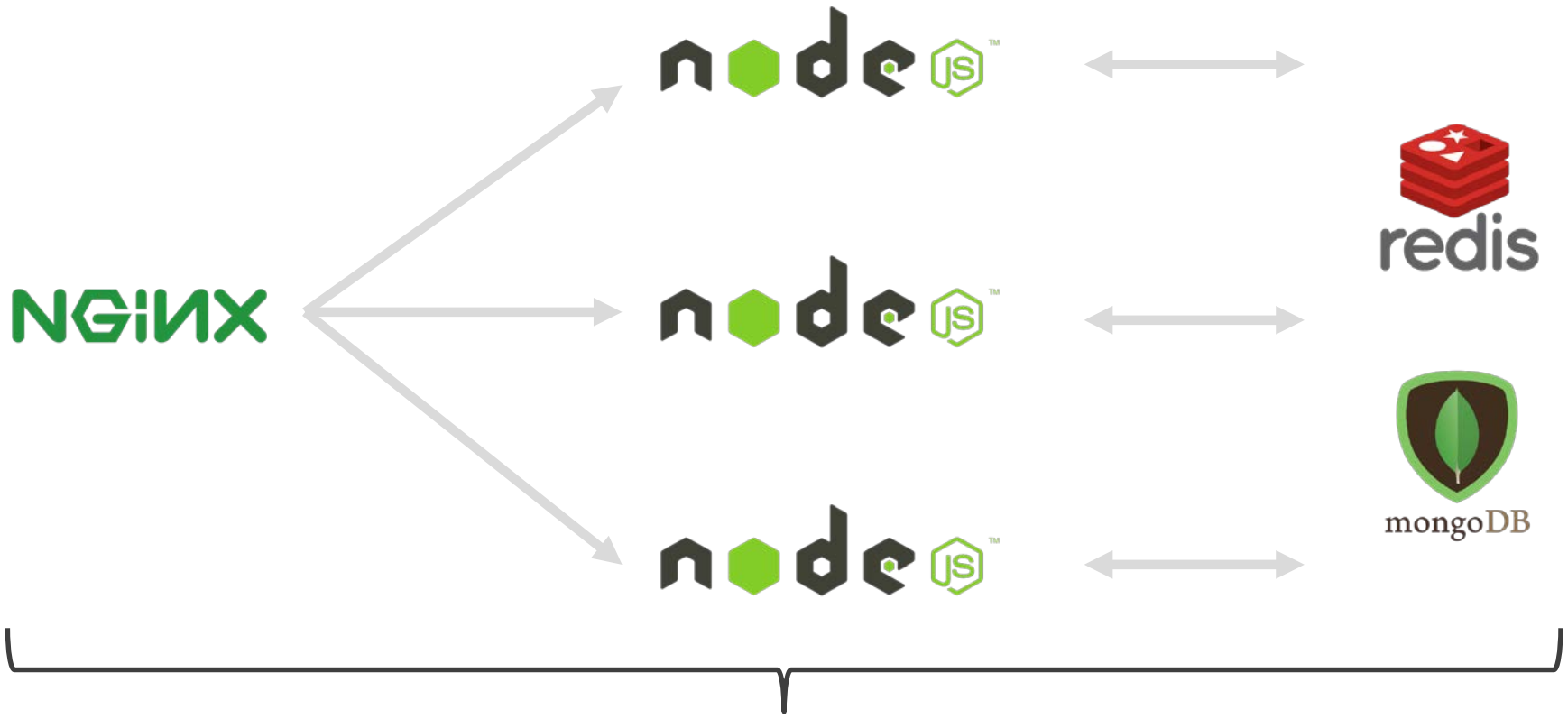
# Stop and Remove Containers, Images, Volumes



# Docker Compose in Action

# Setting up Development Environment Services

# Development Environment Services




**Docker Compose**
**(docker-compose.yml)**

# Creating a Custom docker-compose.yml File

# Managing Development Environment Services

# Summary

Docker Compose simplifies the process of building, starting and stopping services

docker-compose.yml defines services

Excellent way to manage services used in a development environment

Key Docker Compose commands include "build", "up" and "down"

# Moving to Kubernetes

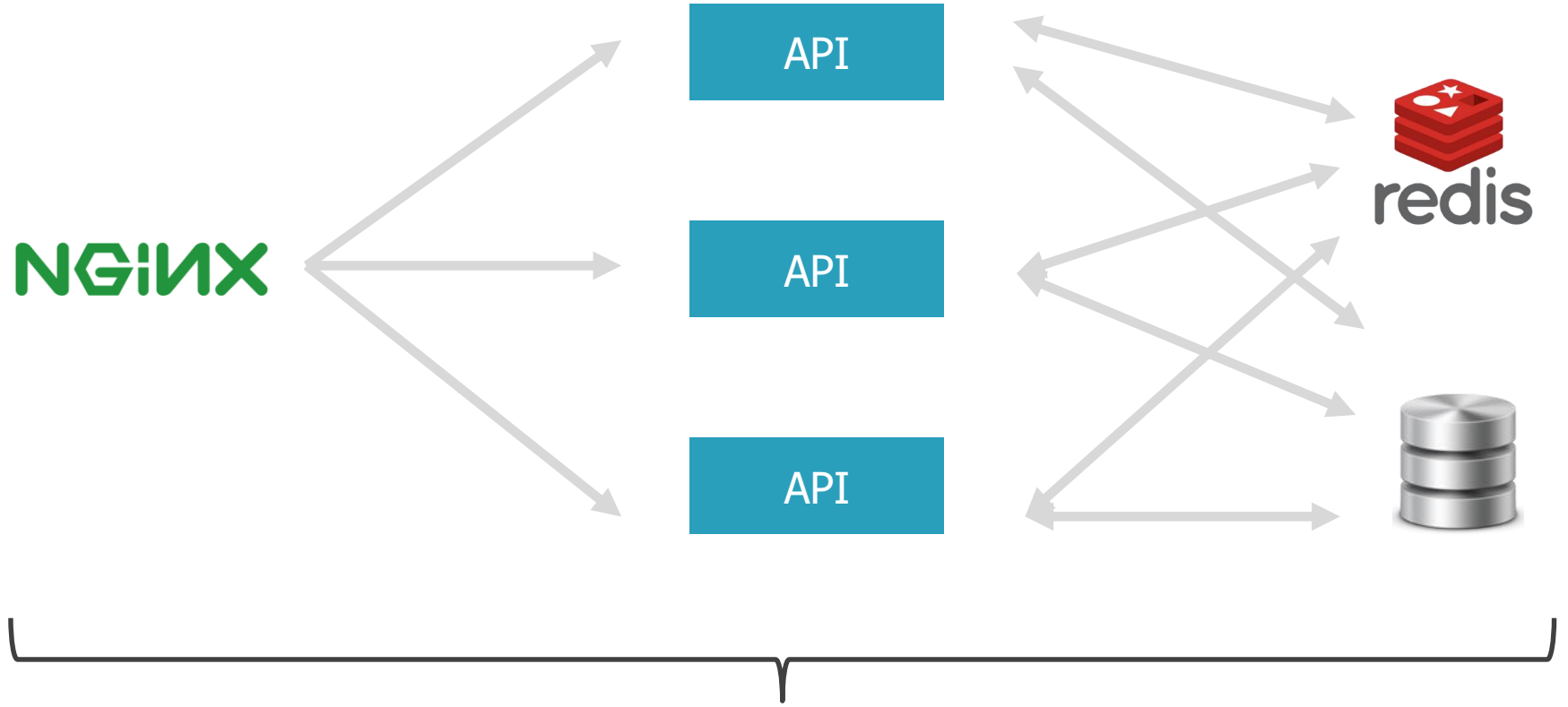
**Dan Wahlin**
WAHLIN CONSULTING

@danwahlin www.codewithdan.com

# Module Agenda

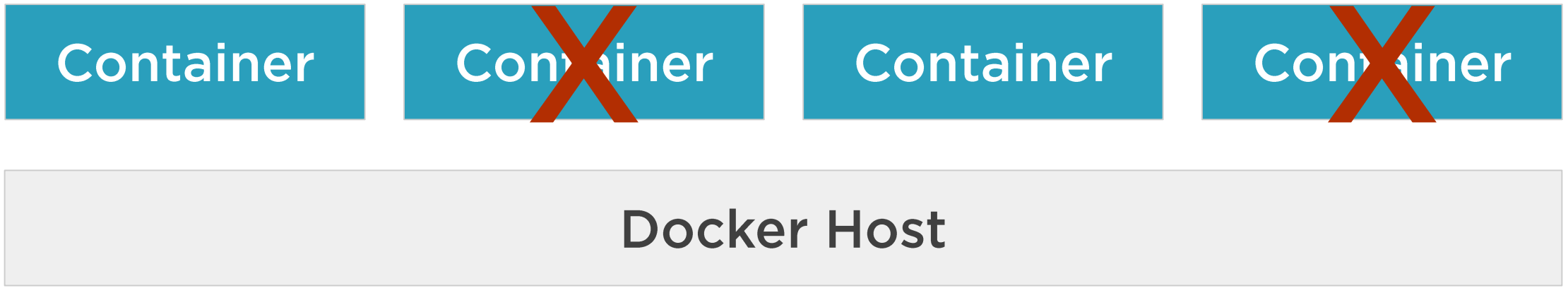
- Beyond Docker Compose
- Introduction to Kubernetes
- Converting from Docker Compose to Kubernetes
- Running Containers in Kubernetes
- Stopping and Removing Containers in Kubernetes

# Beyond Docker Compose

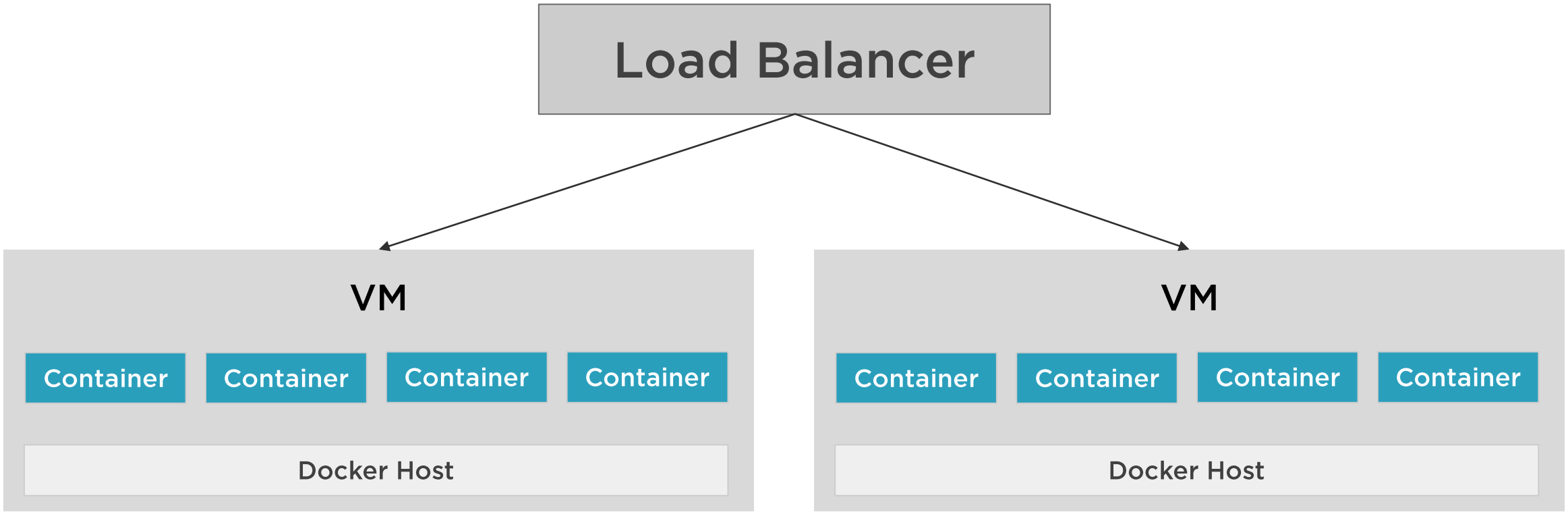



How do you manage all of these containers in test/stage/production?

# How Are You Deploying/Managing Containers?



# How Do You Scale and Load Balance VMs?



# It Would Be Nice if You Could...

- Package up an app, provide a manifest, and let something else manage it for us
- Not worry about the management of containers
- Eliminate single points of failure and self-heal containers
- Have a robust way to scale and load balance containers
- Update containers without bringing down the application
- Have robust networking and persistent storage options

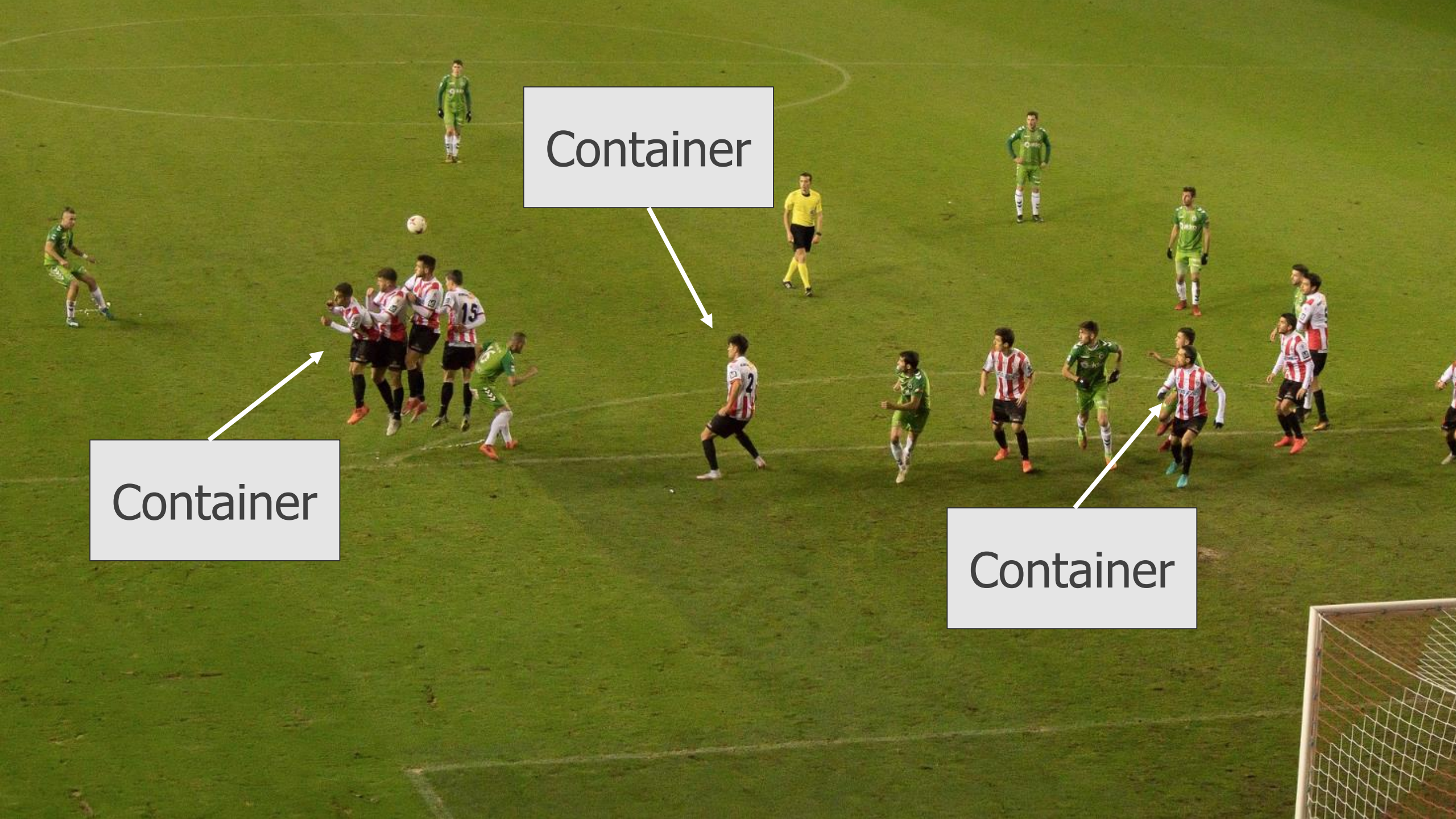
What if we could define the containers we want and then hand it off to a system that manages it all for us?

Welcome to Kubernetes!

# Introduction to Kubernetes

"Kubernetes is an open-source system for automating deployment, scaling, and management of containerized applications."

**https://kubernetes.io**


Container
Container
Container

“Kubernetes is the coach of a container team.”


Container
Container
Container
Kubernetes
(conductor)

“Kubernetes is the conductor of a container orchestra.”

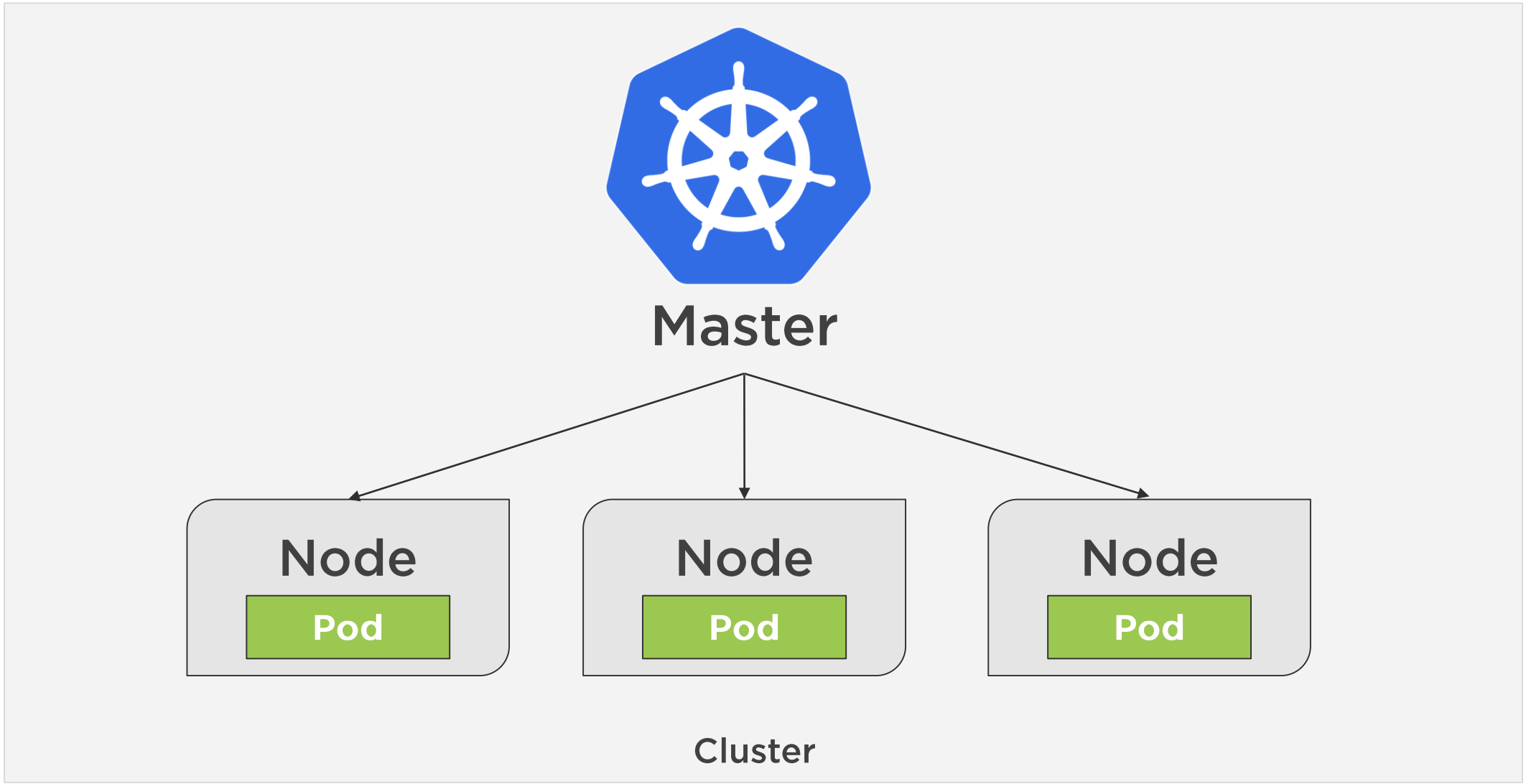
# Kubernetes Overview

- Container and cluster management
- Supported by all major cloud platforms
- Provides a declarative way to define a cluster's state using manifest files (YAML)
- Interact with Kubernetes using kubectl

# Key Features

- Service Discovery/ Load Balancing
- Storage Orchestration
- Automate Rollouts/Rollbacks
- Manage Workloads
- Self-healing
- Secret and Configuration Management
- Horizontal Scaling
- More

# Kubernetes - The Big Picture



# Running Kubernetes Locally

# Running Kubernetes Locally

**Minikube**

https://github.com/kubernetes/minikube

**Docker Desktop**

https://www.docker.com/products/docker-desktop

# Key Kubernetes Concepts

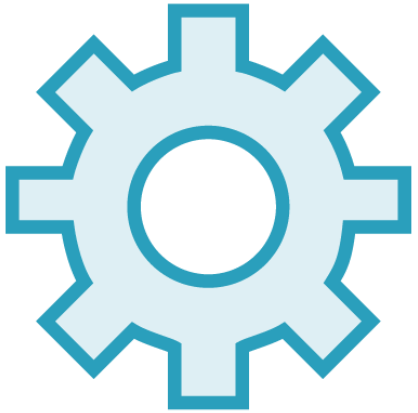
# Deployment

Describe desired state

Can be used to replicate pods

Support rolling updates and rollbacks

# Service

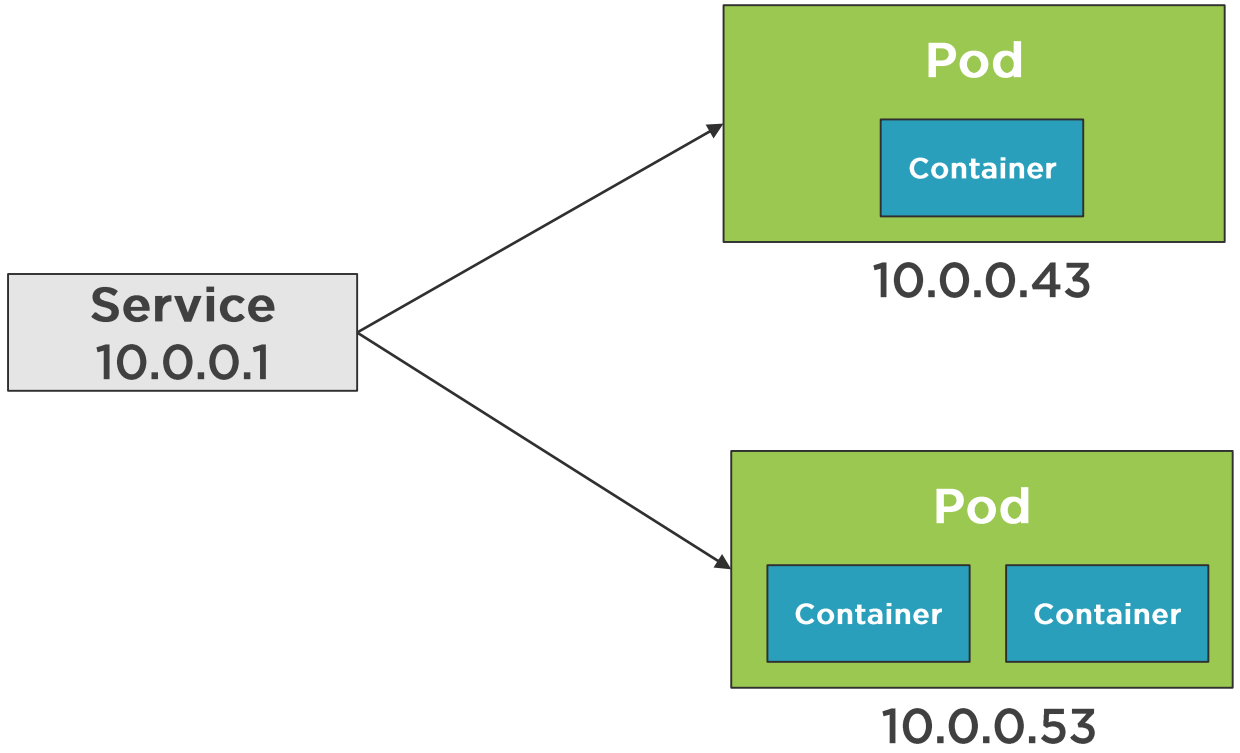


Pods live and die

Services abstract pod IP addresses from consumers

Load balances between pods

# Services


Pod
Container
10.0.0.43
Service
10.0.0.1
Pod
Container
Container
10.0.0.53

# Converting from Docker Compose to Kubernetes

# Migrating from Docker Compose to Kubernetes

**Compose on Kubernetes**

https://github.com/docker/compose-on-kubernetes

**Kompose**

http://kompose.io

# Running Containers in Kubernetes


kubectl version

kubectl get [deployments | services | pods]

kubectl run nginx-server --image=nginx:alpine

kubectl apply –f [fileName | folderName]

kubectl port-forward [name-of-pod] 8080:80

# Stopping and Removing Containers in Kubernetes


kubectl delete –f [fileName | folderName]

# Summary

Kubernetes provides a robust solution for automating deployment, scaling, and management of containers

Provides a way to move to a desired state

Relies on YAML (or JSON) files to represent desired state

Nodes and pods play a central role

A container runs in a pod

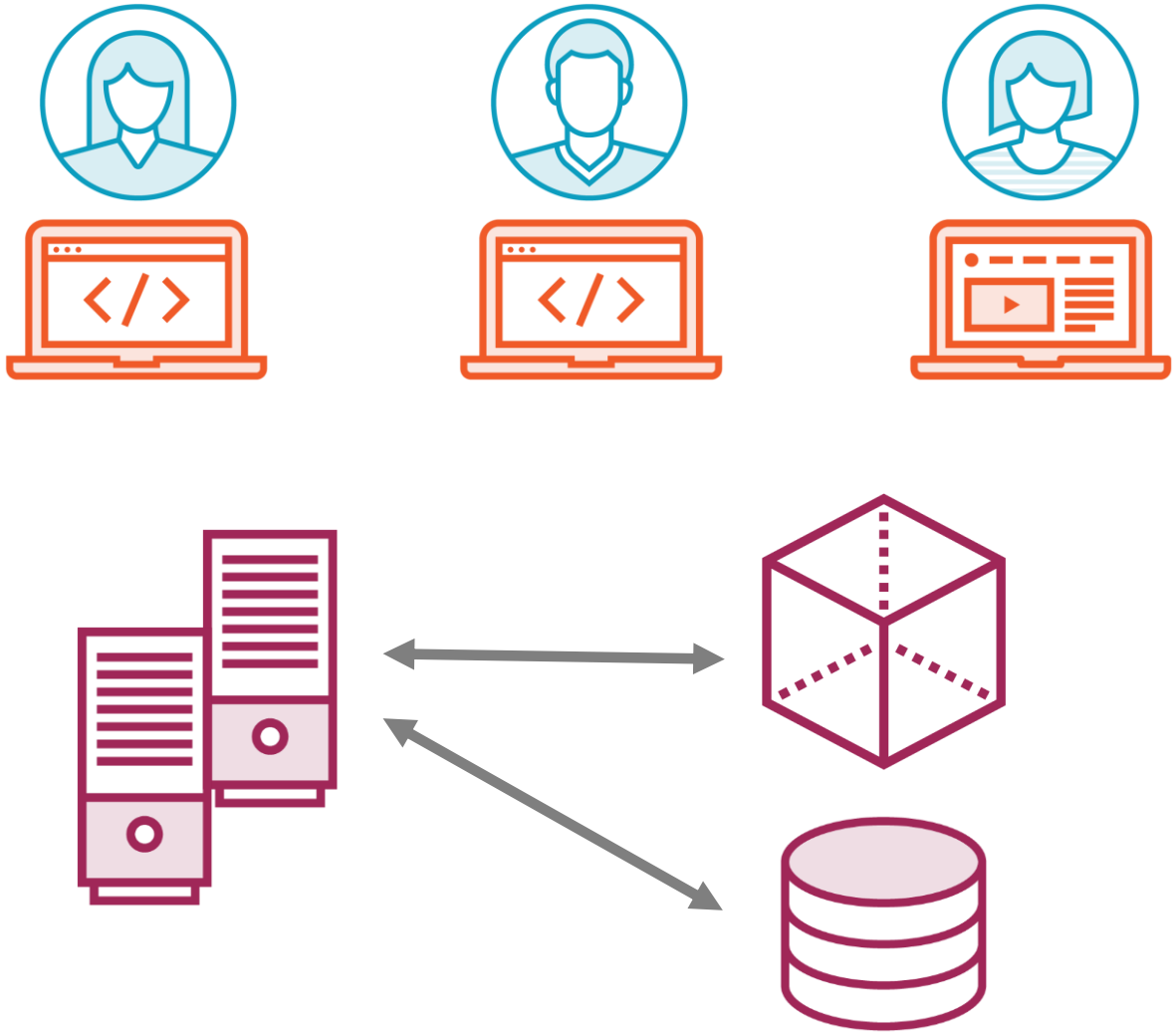
kubectl can be used to issue commands and interact with the Kubernetes API

# Reviewing the Case for Docker

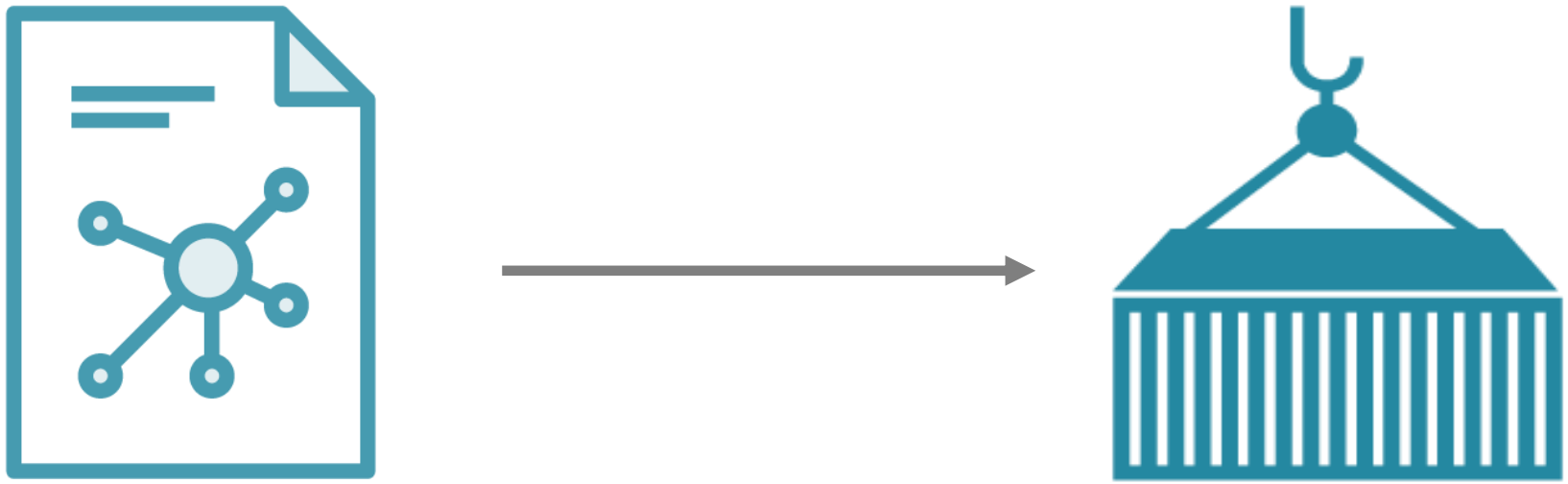
**Dan Wahlin**
WAHLIN CONSULTING

@danwahlin www.codewithdan.com

# Docker Web Development Benefits



# Images and Containers



**Docker Image**

**Docker Container**

# Docker Toolbox

Docker Client

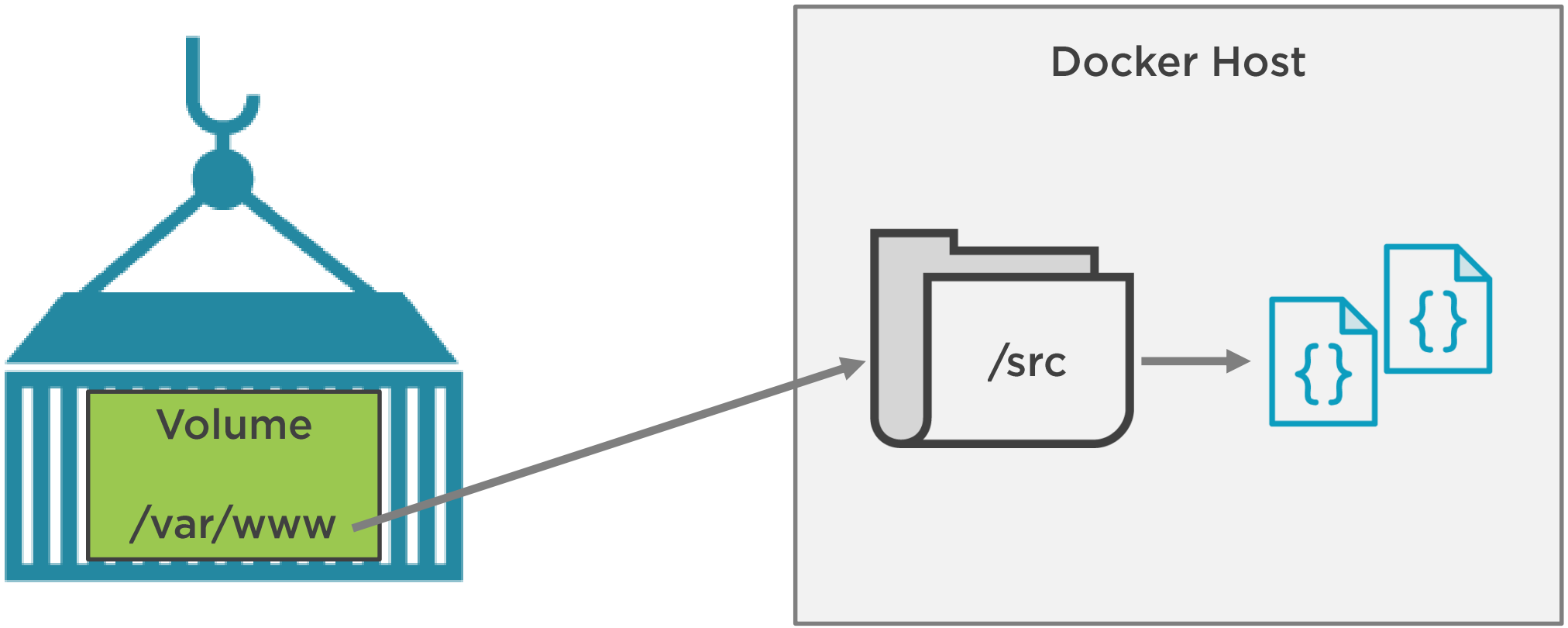
Docker Kitematic

Docker Machine

VirtualBox

Docker Compose

# Docker Volumes



# Dockerfile and Custom Images

```
FROM          node
MAINTAINER    Dan Wahlin
COPY          . /var/www
WORKDIR       /var/www
RUN           npm install
EXPOSE        8080
ENTRYPOINT    ["node", "server.js"]
```

# Docker Compose

```
version: '2'
services:
  node:
    build:
      context: .
      dockerfile: node.dockerfile
    networks:
      -nodeapp-network

  mongodb:
     image: mongo
     networks:
       - nodeapp-network

networks:
  nodeapp-network
    driver: bridge
```

# Containers and Docker Cloud





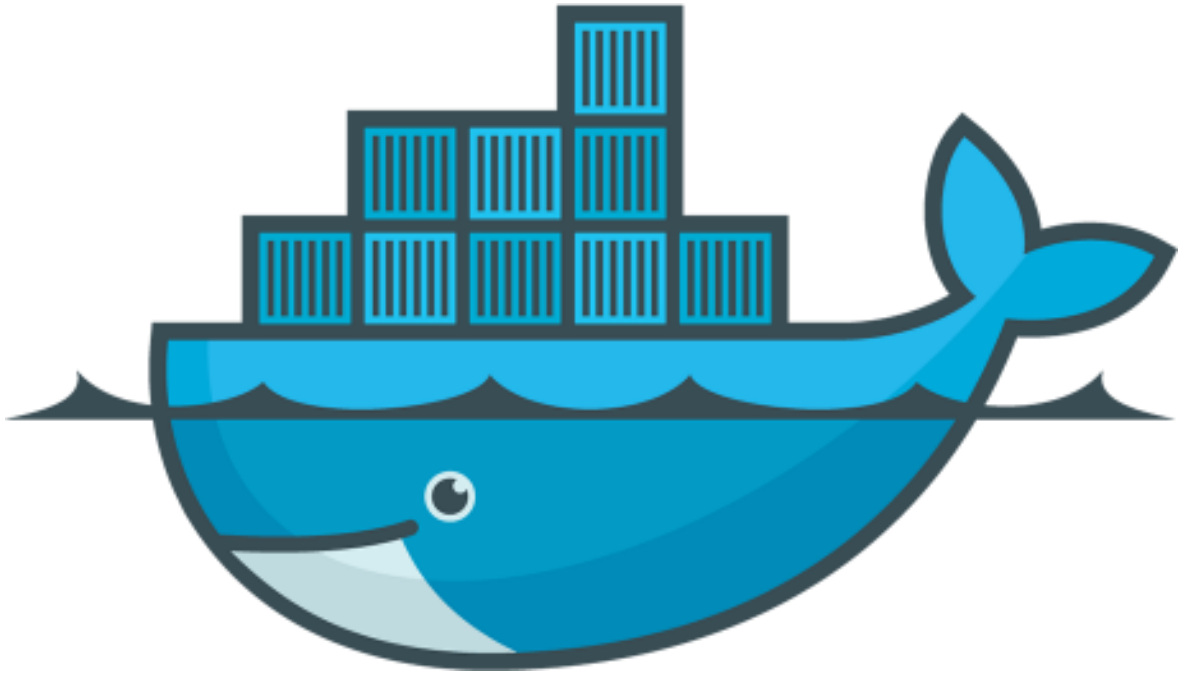


Microsoft Azure

Amazon Web Services

Digital Ocean

Others...



# Docker