Orchestrating Containers with Docker Compose



Dan Wahlin Wahlin Consulting

@DanWahlin www.codewithdan.com

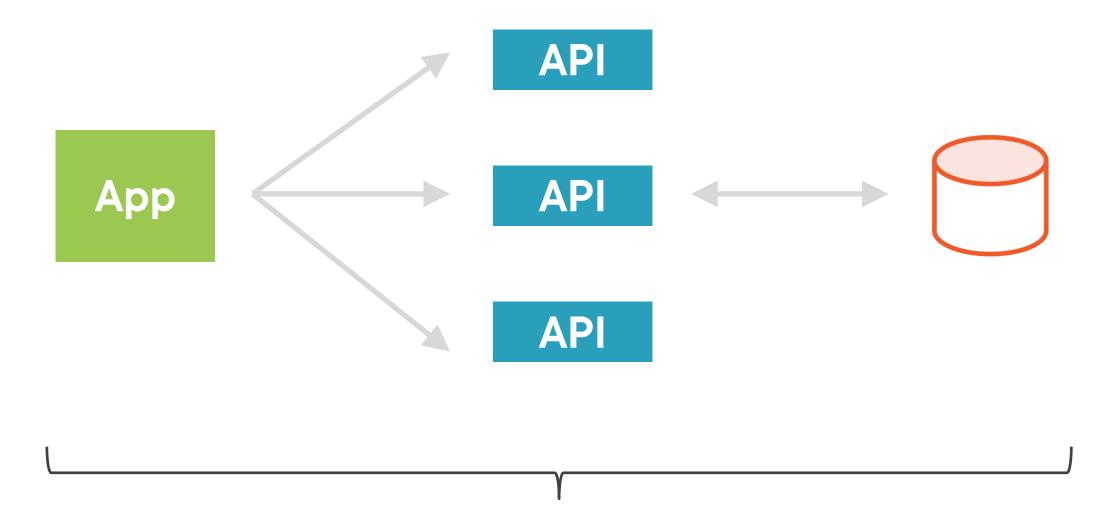
Module Overview



- Docker Compose container properties
- Define ports and volumes
- Define environment variables
- Create a bridge network
- Start and stop containers

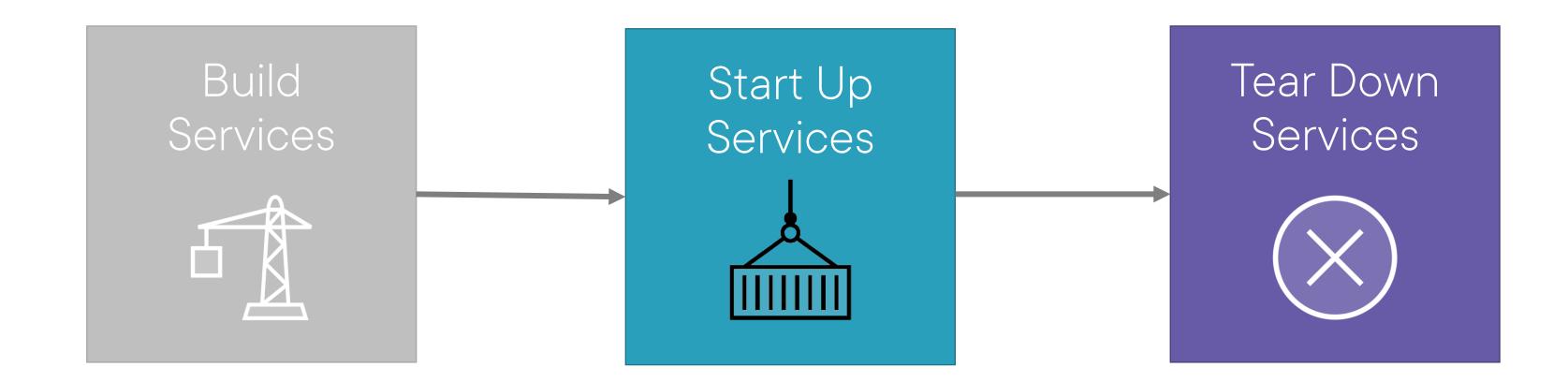
Docker Compose Properties

Orchestrating Multiple Containers



How do you orchestrate multiple containers?

Docker Compose Workflow



Docker Compose Container Properties

image context dockerfile args

ports volumes environment networks

Define Ports and Volumes

Running a Container and Defining Ports

docker run -p <hostPort>:<containerPort> <imageName>

Defining Ports

```
services:
  node:
    image: nodeapp
    build:
      context: .
      dockerfile: node.dockerfile
    ports:
      - "3000:3000"
```

■ Host:Container ports to use

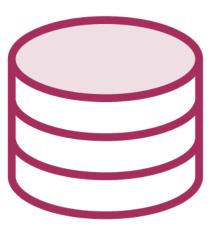
Volume Usage Scenarios

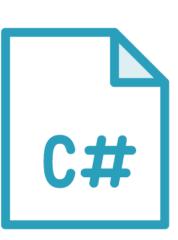
Store log files
Outside of container

Store database files outside of container

Link source code to container

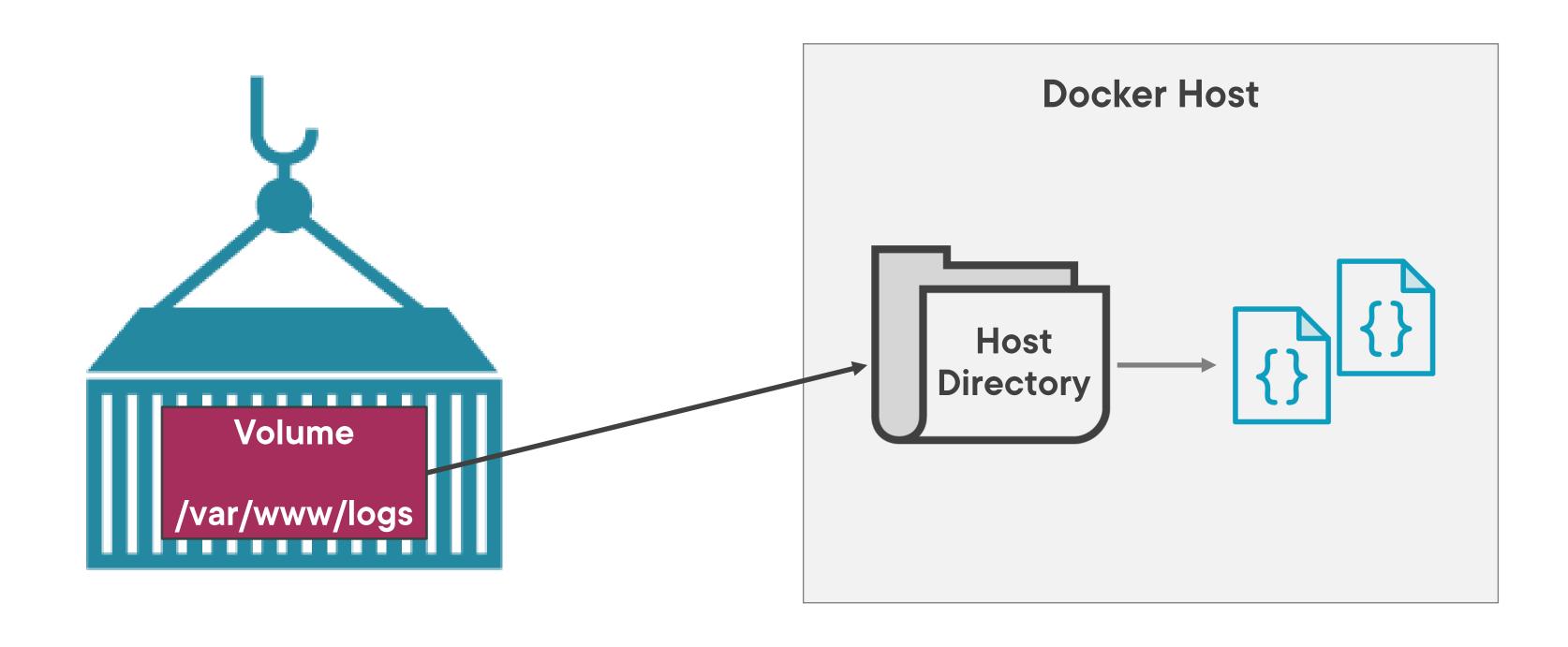






Many more scenarios!

Volume Mounts



Defining a Host Location (Mac/Linux)

Data in folder should be stored on the container host

docker run -p <ports> -v \$(pwd):/var/www/logs <imageToRun>

Print working directory (Mac/Linux)

Defining a Host Location (PowerShell)

Print working directory (Windows PowerShell)

docker run -p <ports> -v \${PWD}:/var/www/logs <imageToRun>

Defining Volumes

```
services:
  node:
    image: nodeapp
    build:
      context: .
      dockerfile: node.dockerfile
    ports:
      - "3000:3000"
    volumes:
      - ./logs:/var/www/logs
```

■ Define volume (host directory and container directory)

Define Environment Variables

Environment Variables and Containers

NODE_ENV=production

APP_VERSION=1.0

LOG_DIR=./logs



Environment Variables

Container

Defining an Environment Variable

Define environment variable that container can access

docker run -p <ports> --env NODE_ENV=production <imageToRun>

Defining Environment Variables

```
services:
  node:
    image: nodeapp
    build:
      context: .
      dockerfile: node.dockerfile
    ports:
      - "3000:3000"
    environment:
      - NODE_ENV=production
      - APP_VERSION=1.0
```

■ Define environment variables that will be available in the running container

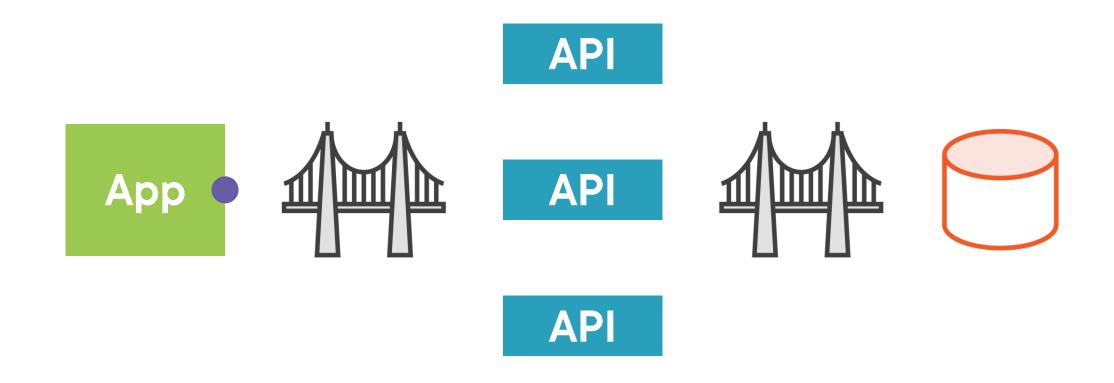
Defining Environment Variables in a File

```
services:
  node:
    image: nodeapp
    build:
      context: .
      dockerfile: node.dockerfile
    ports:
      - "3000:3000"
    env_file:
      - ./common.env
      - ./settings.env
```

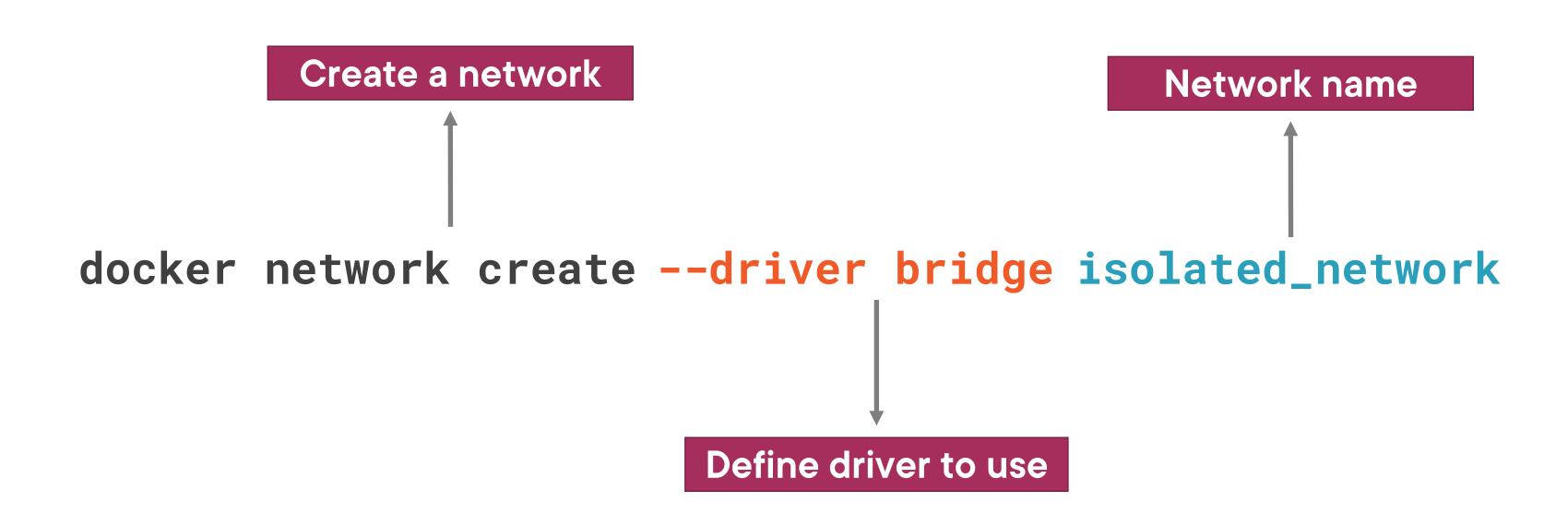
■ Pull in environment variables from these files and make them available to the container

Create a Bridge Network

Using a Bridge Network to Communicate



Create a Bridge Network



Defining a Bridge Network

```
version: '3.x'
services:
  node:
    container_name: nodeapp
    image: nodeapp
    build:
    ports:
      - "3000:3000"
    networks:
      nodeapp-network
networks:
  nodeapp-network:
    driver: bridge
```

■ Put service in "nodeapp-network" bridge network

◄ Define a bridge network

Start and Stop Containers

Key Docker Compose Commands



docker-compose build



docker-compose up



docker-compose down

Creating and Starting Containers

docker-compose up -d

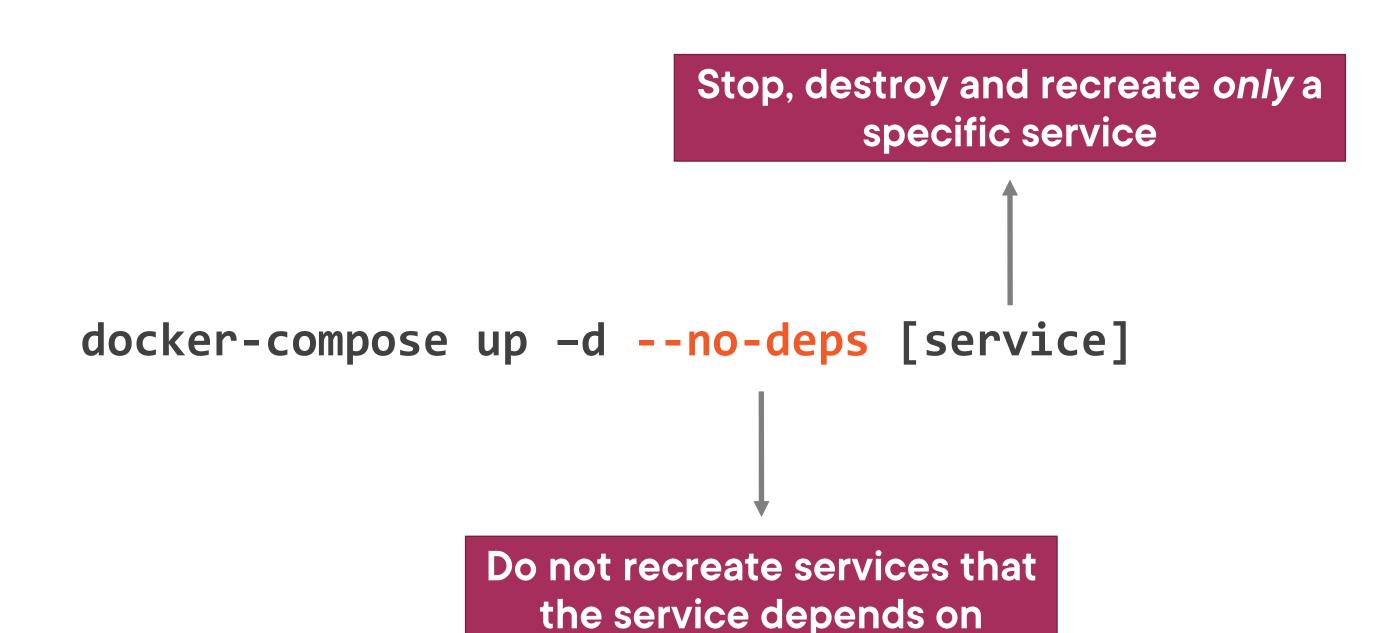
Run in detached mode

Defining Dependencies

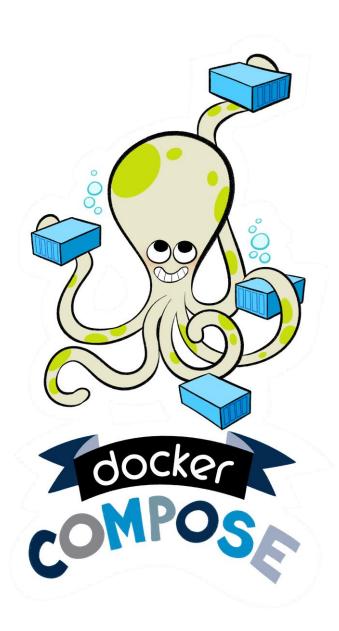
```
services:
  node:
    image: nodeapp
    build:
      context: .
      dockerfile: node.dockerfile
    ports:
      - "3000:3000"
    volumes:
      - ./logs:/var/www/logs
    depends_on:
      - mongodb
```

■ This service depends on another service named "mongodb" so start the other service first.

Creating and Starting Containers



Key Docker Compose Container Commands



docker-compose ps
docker-compose stop
docker-compose start
docker-compose rm

Using Docker Compose Commands

Key Docker Compose Commands



docker-compose --help docker-compose build docker-compose up docker-compose up -d docker-compose up -d --no-deps [service] docker-compose down docker-compose ps docker-compose stop [service] docker-compose start [service]

Summary



- Docker Compose can be used to orchestrate multiple containers
- Ports, volumes, and environment variables can be defined in Docker Compose files
- Bridge networks are used for container communication and can be defined in Docker Compose files
- Docker Compose commands can be used to start, stop, list, and remove containers