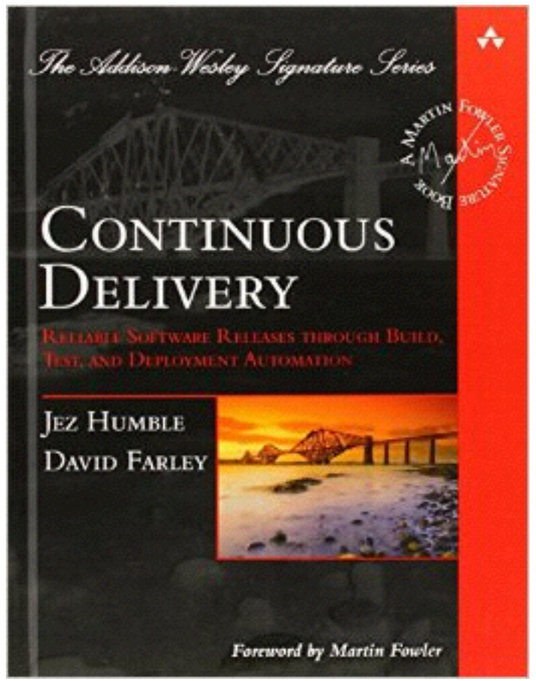
Continuous Delivery Using Docker and Ansible

COURSE INTRODUCTION

Justin Menga
FULL STACK TECHNOLOGIST
@jmenga pseudo.co.de



“Create a Repeatable, Reliable Process for Releasing Software”

Continuous Delivery (2011) - Jez Humble, David Farley

Continuous Delivery

Release Often

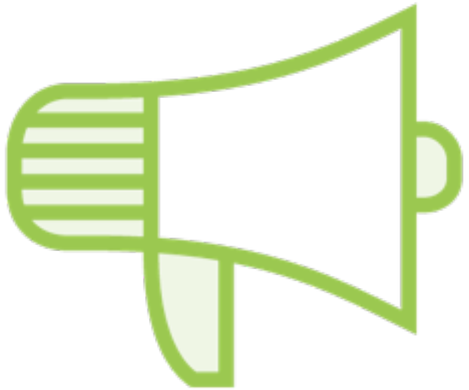
Release Faster

Greater Reliability

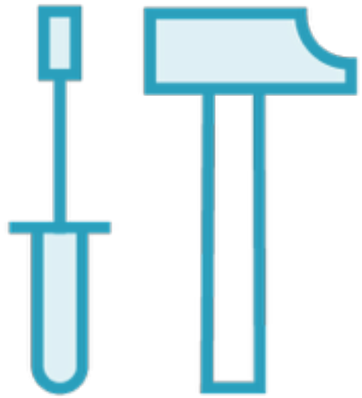
Deliver real business value, as quickly and efficiently as possible

Business Benefits

Innovation



Fast Feedback



Warranty

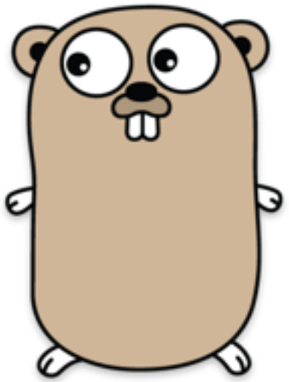
Course Goals


Test


Build


Release


Deploy



Continuous Delivery
Workflow/Pipeline

Docker Compose

```
> make test
> make build
> make release
> make tag
> make publish
```

Run Locally

Jenkins

GitHub

Test

Build

Release

Deploy

Docker Hub

Docker

Continuous Delivery
Workflow/Pipeline

Course Audience

Developers

Wrap test and build actions in Docker

Your applications work on ANY machine

Run production-like environment locally

Fast track your changes to production

Operations

Build Continuous Delivery pipelines

Automate test, build, release and deploy

Setup Continuous Delivery systems

Infrastructure as Code

Architects

Application agnostic methodology

Run on any Continuous Delivery system

Immutable Infrastructure

Infrastructure as Code

Course Prerequisites

Course Prerequisites

Docker

Docker Engine
Docker Compose

Ansible

Configuration Management

Application Delivery

Continuous Integration
Application Testing

Course Prerequisites

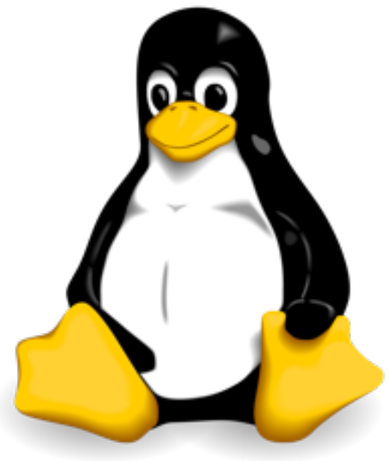
No experience required

Continuous Delivery Workflow
is compatible with anything that runs on Linux

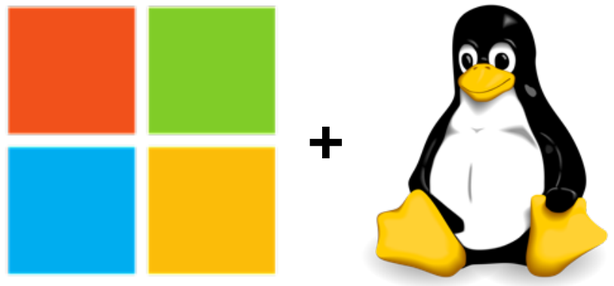
Development Environment

Mac OS X

Linux



Windows + Linux VM



Course Tour

Module 2 - Creating the Sample Application

Sample Application

Unit and Integration Tests

Acceptance Tests

Module 3 - Unit/Integration Testing using Docker

Test Stage

Docker Images

Docker and Docker Compose Building Blocks

Module 4 - Building Artifacts using Docker

Build Stage

Test Stage Consistency

Creating Built Distributions

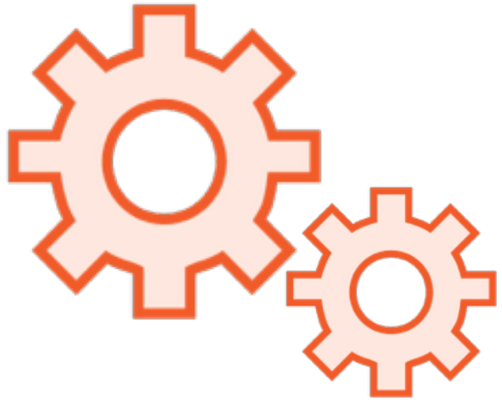
Module 5 - Creating Releases using Docker

Release Stage

Release Image

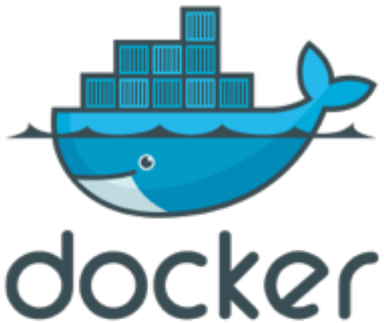
Release Environment

Module 6 - Continuous Delivery Automation



Make Build System

GitHub Repos



Docker Hub Repos

Module 7 - Enhancing the Workflow

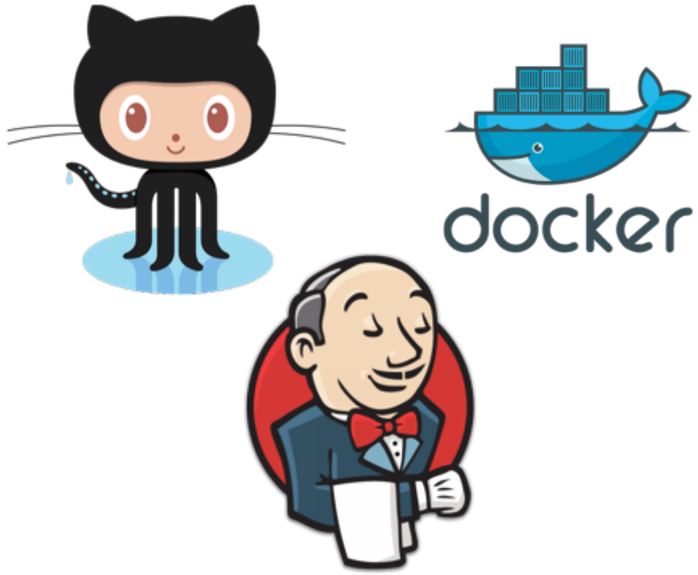
Production Ready

Error/Failure Handling

Tag Publish

Module 8 - Continuous Delivery using Jenkins

Jenkins



GitHub/Docker Hub Integration

Running Jenkins in AWS

Module 9 - Continuous Deployment using Ansible

Ansible Deployment Playbook

Infrastructure as Code

End-to-End Continuous Delivery

Continuous Delivery Overview

Why Continuous Delivery?

Release Often
(Fast)

Release Small
(Focused)

MVP

Minimum Viable Product

Test your ideas in the real world

Build better smarter products

Reduce Risk

Smaller Changes = Less Risk

Continuously Deploy

- To at least one environment
- Always deploying means you get much better at deployments

Small scope of change

- Easier to deploy
- Easier to fix

Real Progress

Each change is real progress

- You know it’s done because you deployed it successfully

Build Confidence

- In ability to release new features
- In ability to detect and fix issues

How to achieve Continuous Delivery?

Methodology

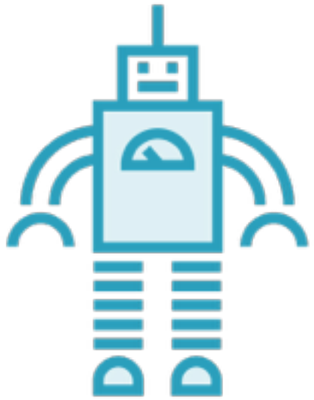
Application Agnostic

Portability

Separation of Concerns

And more...

- Immutable Infrastructure
- Infrastructure as Code



Automation

Saves time and money

Repeatable and consistent results

Automate as much as possible

- Tests
- Creating environments
- Deployments
- Monitoring
- etc...

Testing

Quantitative Measurement

Production Readiness

Automation and Consistency

Just one more thing...

People

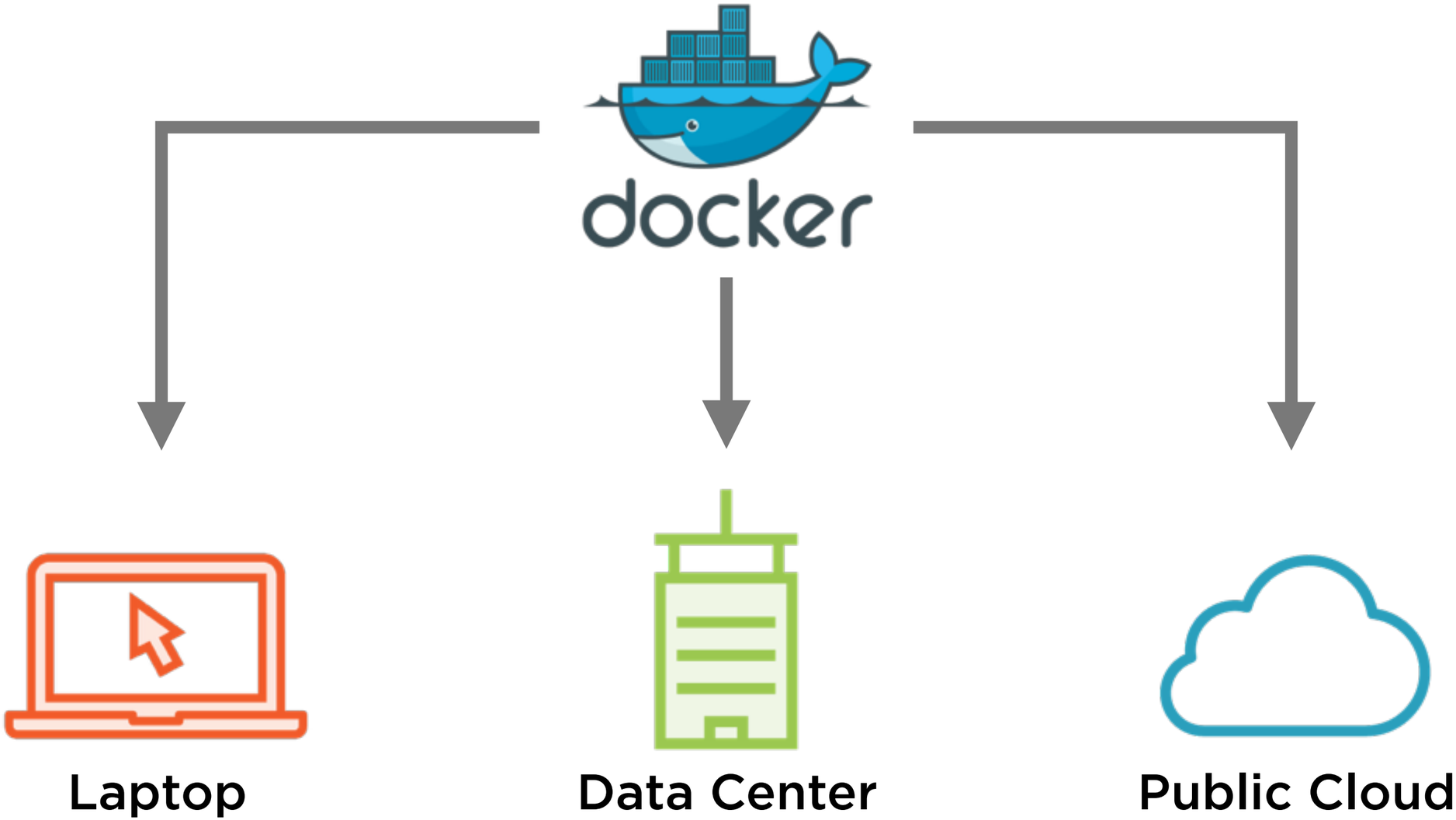
Why Docker?

Why Docker?

Speed

Portability

Docker Anywhere...



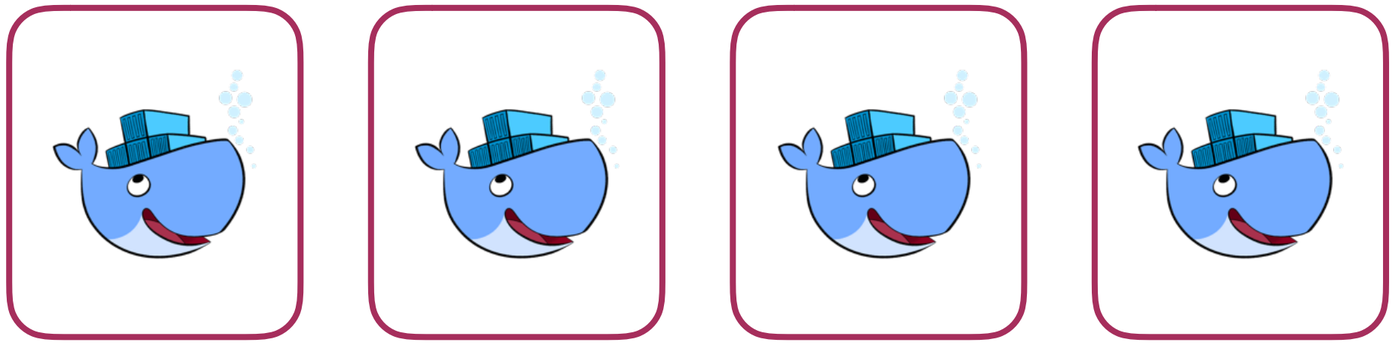
Docker Simplifies Your Infrastructure

Containers

Diverse Requirements

Infrastructure

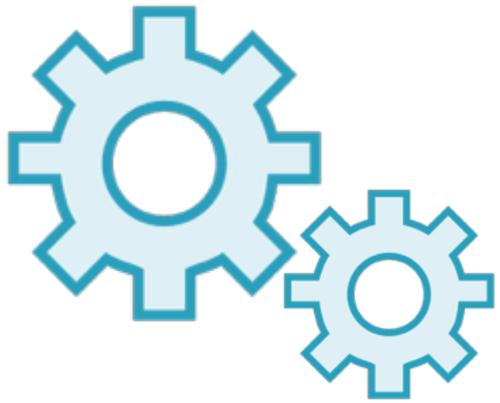
Simple Requirements



Why Docker?

Speed

Portability



Automation

Dockerfile

```
FROM ubuntu:trusty
MAINTAINER Justin Menga

ENV MY_ENV_VAR=some_value

RUN apt-get install nginx -y

COPY my_file /path/to/my_file

COMMAND [“start.sh”,”-x opt”]
```

- Image Metadata
- Environment Variables
- Commands to run on build
- Copy files to image
- Command to run on start

docker-compose.yml

```
app:
  image: myorg/myrepo:latest
  links:
    - db
  volumes:
    - /path/to/host:/path
  environment:
    MYSQL_DB: todobackend
  ...



db:
  image: mysql
  ...
```

- “app” service (aka container)
- Image the service is based from
- List of service dependencies

- List of volumes to mount

- Environment variables

- “db” service (another container)

Continuous Delivery Architecture

Continuous Delivery Workflow

Test

Build

Release

Deploy

Test Stage

Unit and Integration Testing

Use Docker to wrap test runners

- e.g. maven, gradle (Java)
- e.g. manage.py (Python Django)

Benefits of using Docker

- Predefined environment
- Portable
- Consistent
- Repeatable

Build Stage

Build Application Artifacts

- e.g. Python Wheels
- e.g. Java JAR files

Must represent as tested application state

Creates a Deployable Artifact

- i.e. a built distribution
- Pre-compiled, pre-built
- Installation requires no development dependencies

Release Stage

Build Docker Release Image

- Includes minimal runtime environment
- Installs application artifacts

Release Environment

- A production-like environment
- Use an external test runner to run acceptance tests

Tag and Publish

- Only if acceptance tests pass
- Docker Hub

Deploy Stage

Deploy release image to:

- At least one environment
- e.g. Development Environment
- e.g. QA or Staging Environments
- Perhaps even production

Fully automated deployments

- Using orchestration tools such as Ansible
- Leverage Infrastructure automation tools such as AWS CloudFormation

Demo

Preparing your Environment

- Choosing a Virtualization Platform
- Installing Brew
- Installing Docker Tools
- Installing Ansible
- Installing other Tools
- GitHub, Docker Hub and AWS
- Creating a Docker Virtual Machine
- Creating the Course Root Folder

Choosing a Virtualization Platform

Choosing a Virtualization Platform

VirtualBox
virtualbox.org

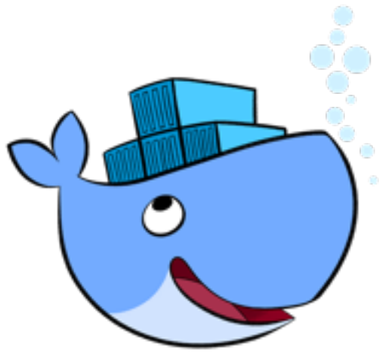
VMWare Fusion
vmware.com/fusion

Parallels
parallels.com

Installing Brew

Installing Docker

Installing Docker Tools

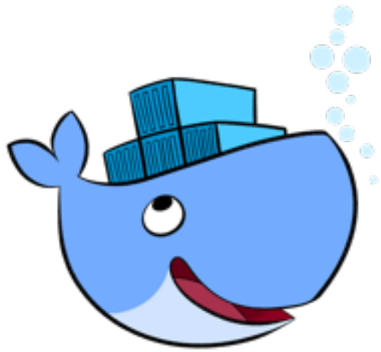


Docker
docker.com/docker-engine

Docker Compose
docker.com/docker-compose

Docker Machine
docker.com/docker-machine

Installing Docker Tools



Docker

docker.com/docker-engine

Tested on 1.9, 1.10 and 1.11

(Version 1.10+ recommended)

Docker Compose

docker.com/docker-compose

Tested on 1.5, 1.6 and 1.7

(Version 1.6+ recommended)

Docker Machine

docker.com/docker-machine

Tested on 0.5, 0.6 and 0.7

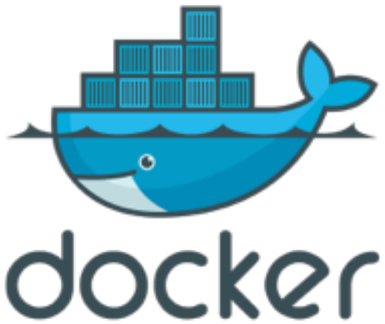
(Version 0.6+ recommended)

Installing Ansible

Installing Other Tools

GitHub, Docker Hub and AWS

GitHub
github.com



Docker Hub
hub.docker.com

AWS
aws.amazon.com/free

Creating a Docker Virtual Machine

Setting up a Course Folder

Summary

Course Introduction

- Course Goals and Outline
- Continuous Delivery Overview
- Why Docker?
- Preparing the local environment
- Creating a Docker Machine