Docker 101

Agenda

Docker Enterprise Edition (EE) and Community Edition (CE)Containers are NOT VMs

Working with Docker (Build, Ship, Run)

Container Architecture

But Why?

Getting started

Q & A

Docker Enterprise Edition (EE) and Community Edition (CE)

Docker Enterprise Edition (EE) and Community Edition (CE)

Enterprise Edition (EE)

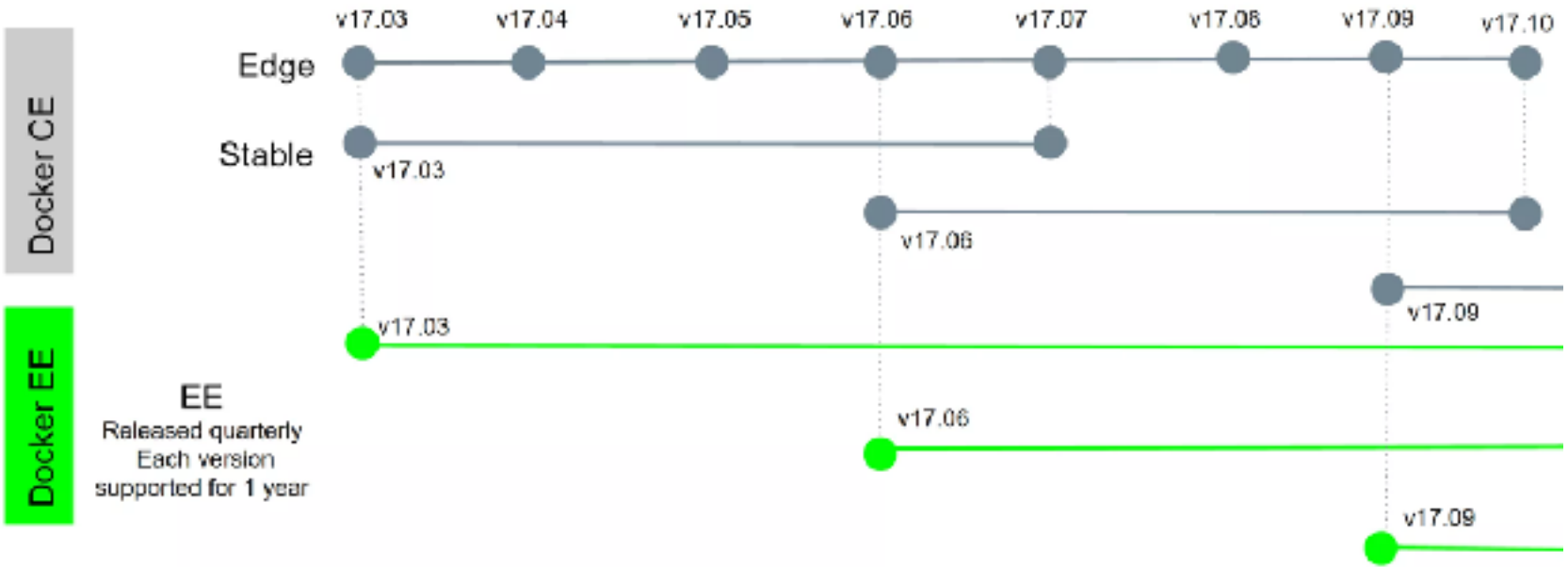
- CaaS enabled platform subscription (integrated container orchestration, management and security)
- Enterprise class support
- Quarterly releases, supported for one year each with backported patches and hotfixes.
- Certified Infrastructure, Plugins, Containers

Community Edition (CE)

- Free Docker platform for “do it yourself” dev and ops
- Monthly Edge release with latest features for developers
- Quarterly release with maintenance for ops

Lifecycle

Squaring the circle: Faster releases and better stability



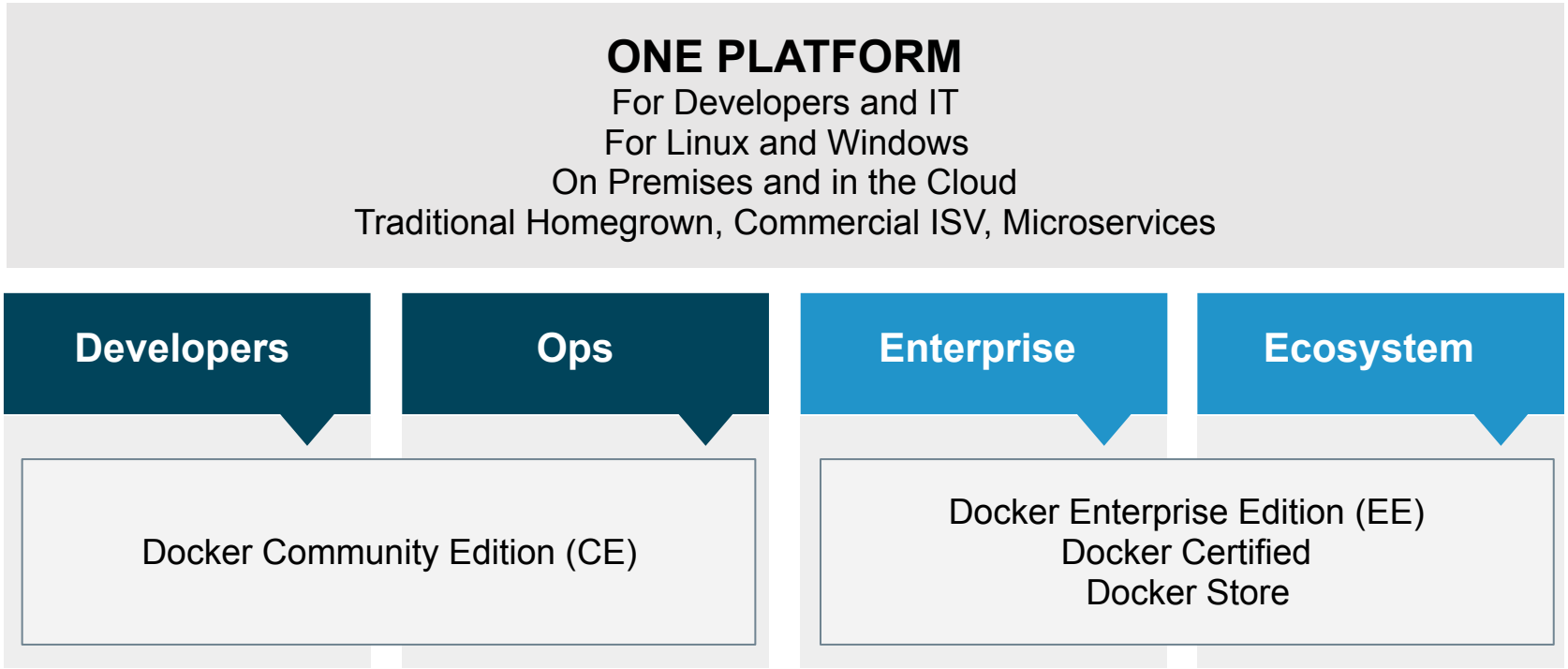
Editions and certification

NEW Certification program for Infrastructure, Plugins and Containers

Docker CE availability

- Docker CE for Mac and Windows 10
- Docker CE for AWS and Azure
- Docker CE for Linux: Debian, Ubuntu, Fedora, CentOS
- (Docker EE is available at no extra cost on Windows Server 2016)

The Docker Platform



docker

Containers are not VMs

Docker containers are NOT VMs

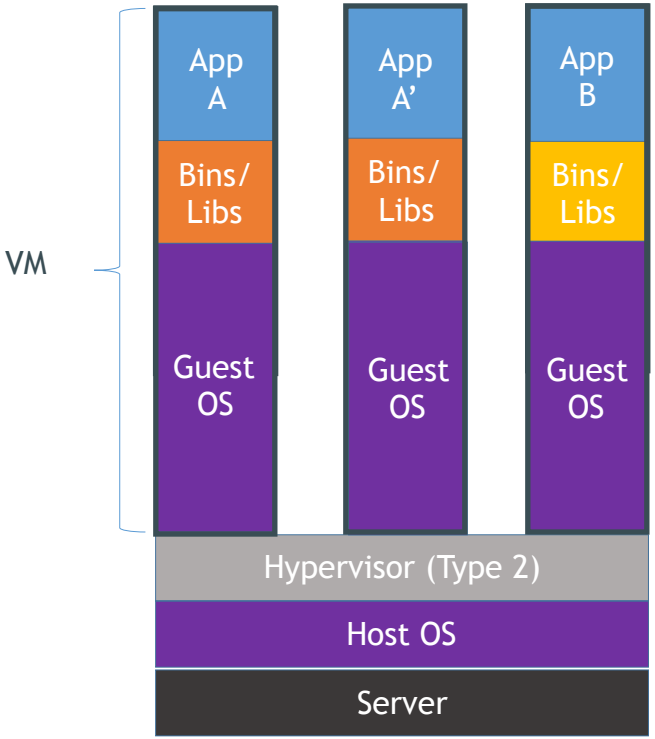
- Easy connection to make
- Fundamentally different architectures
- Fundamentally different benefits

Physical server

VMs

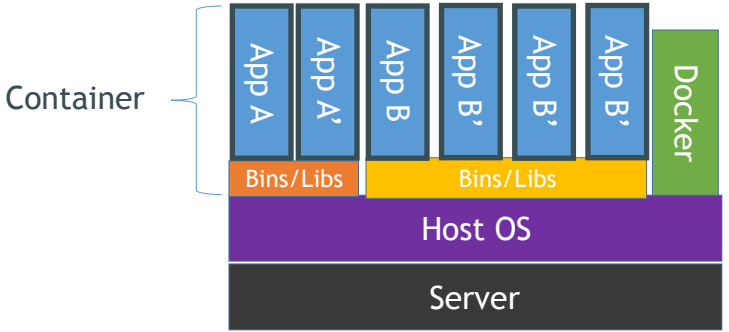
Containers

Containers vs. VMs

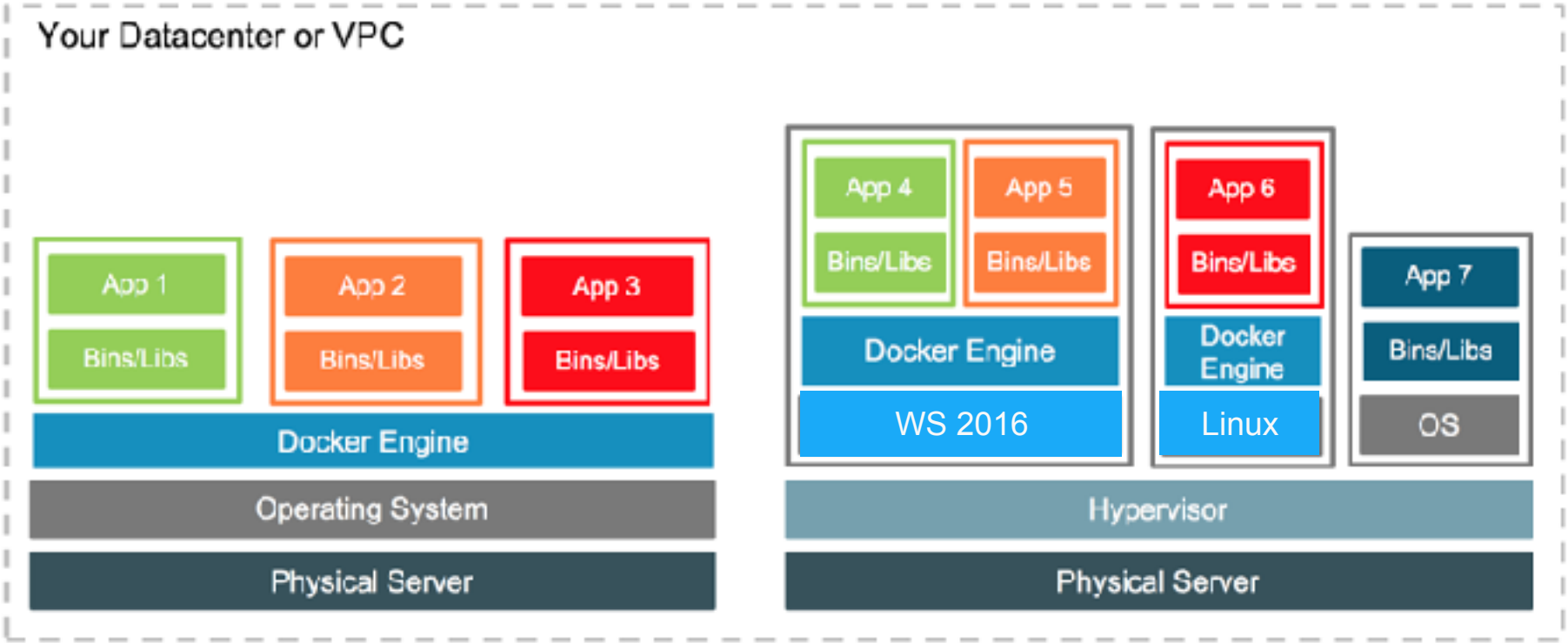


Containers are isolated, but share OS and, where appropriate, bins/ libraries

…result is significantly faster deployment, much less overhead, easier migration, faster restart

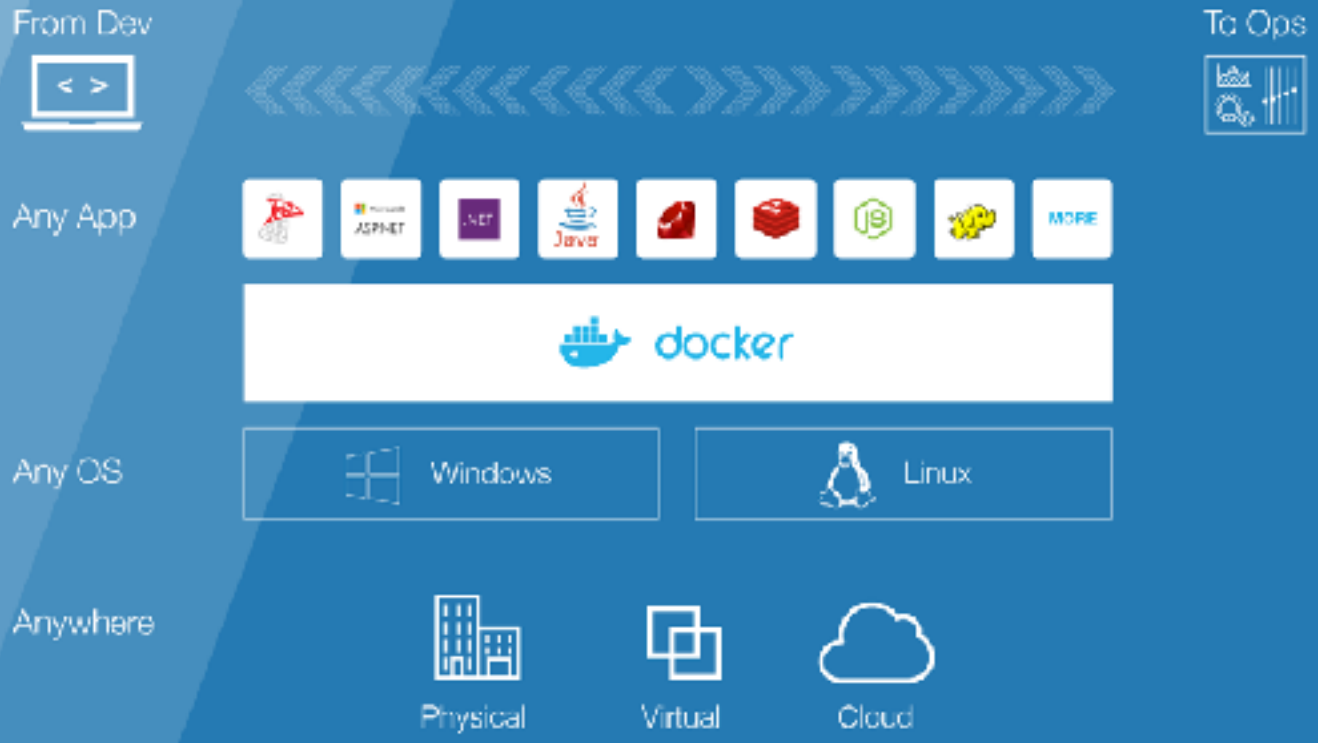


They’re different, not mutually exclusive



Build, Ship, and Run

Build, Ship, Run, Any App Anywhere



Some Docker vocabulary

Docker Image

The basis of a Docker container. Represents a full application

Docker Container

The standard unit in which the application service resides and executes

Docker Engine

Creates, ships and runs Docker containers deployable on a physical or virtual, host locally, in a datacenter or cloud service provider

Registry Service (Docker Hub or Docker Trusted Registry)

Cloud or server based storage and distribution service for your images

Basic Docker Commands

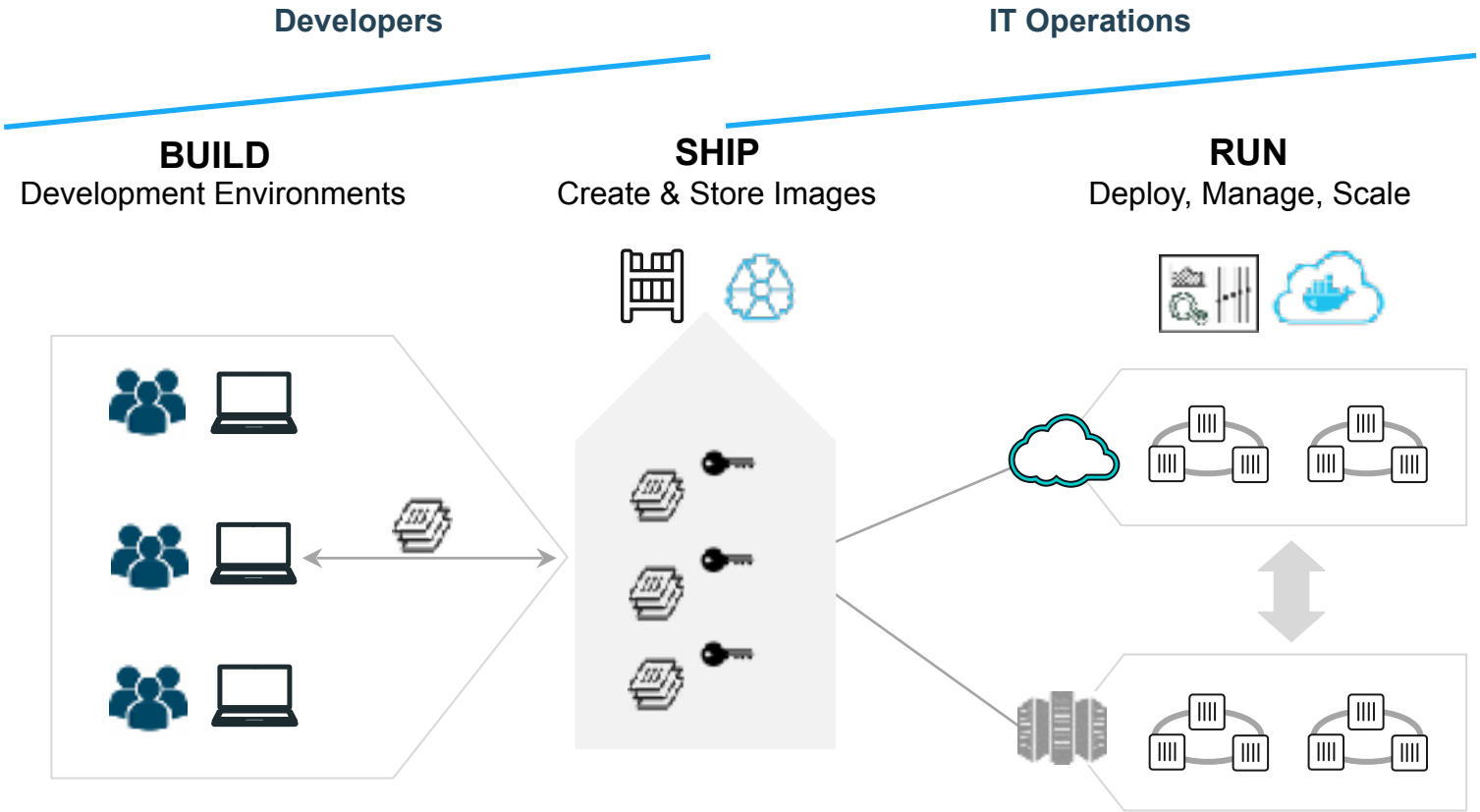
```
$ docker pull/catweb:1.0

$ docker images

$ docker run –d –p 5000:5000 ––name catweb mikegcoleman/catweb:latest

$ docker ps

$ docker stop catweb (or <container id>)

$ docker rm catweb (or <container id>)

$ docker rmi mikegcoleman/catweb:latest (or <image id>)

$ docker build –t mikegcoleman/catweb:2.0 .

$ docker push mikegcoleman/catweb:2.0
```

Dockerfile – Linux Example

```
# our base image
FROM alpine:latest

# Install python and pip
RUN apk add --update py-pip

# upgrade pip
RUN pip install --upgrade pip

# install Python modules needed by the Python app
COPY requirements.txt /usr/src/app/
RUN pip install --no-cache-dir -r /usr/src/app/requirements.txt

# copy files required for the app to run
COPY app.py /usr/src/app/
COPY templates/index.html /usr/src/app/templates/

# tell the port number the container should expose
EXPOSE 5000

# run the application
CMD ["python", "/usr/src/app/app.py"]
```

- Instructions on how to build a Docker image
- Looks very similar to “native” commands
- Important to optimize your Dockerfile

Put it all together: Build, Ship, Run Workflow



docker

Demo

Build, Ship, and Run

Now you try it!

Visit http://docs.docker.com/installation

Install the right version of Docker for your machine

- Docker for Mac
- Docker for Windows

After Docker is installed, run Catweb

- `docker run –d –p 5000:5000 --name catweb mikegcoleman/catweb`

Browse to port 5000 on your machine

- `http://localhost:5000`

Docker Container Architecture

Image Layers

Docker File System

- Logical file system by grouping different file system primitives into branches (directories, file systems, subvolumes, snapshots)
- Each branch represents a layer in a Docker image
- Allows images to be constructed / deconstructed as needed vs. a huge monolithic image (ala traditional virtual machines)
- When a container is started a writeable layer is added to the “top” of the file system

Copy on Write

Super efficient:

- Sub second instantiation times for containers
- New container can take <1 Mb of space

Containers appears to be a copy of the original image

But, it is really just a link to the original shared image

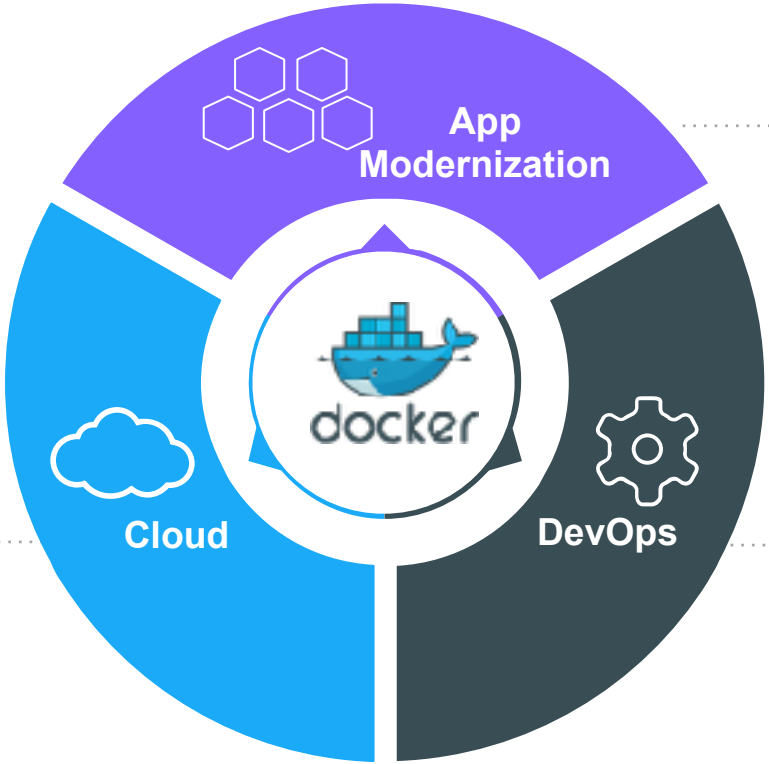
If someone writes a change to the file system, a copy of the affected file/ directory is “copied up”

What about data persistence?

- Volumes allow you to specify a directory in the container that exists outside of the docker file system structure
- Can be used to share (and persist) data between containers
- Directory persists after the container is deleted
  - Unless you explicitly delete it
- Can be created in a Dockerfile or via CLI

But, Why?

Enterprises are looking to Docker for critical transformations



3 out 4

Top initiatives revolve around applications

80%

Docker is central to cloud strategy

44%

Looking to adopt DevOps

State of App development Survey: Q1 2016

Docker delivers speed, flexibility and savings

Agility

13X

More software releases

65%

Reduction in developer onboarding time

Portability

41%

Move workloads across private/ public clouds

Eliminate

"works on my machine" issues

Control

62%

Report reduction in MTTR

10X

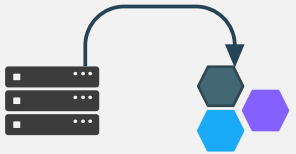
Cost reduction in maintaining existing applications

State of App development Survey: Q1 2016, Cornell University case study

One platform delivers one journey for all applications

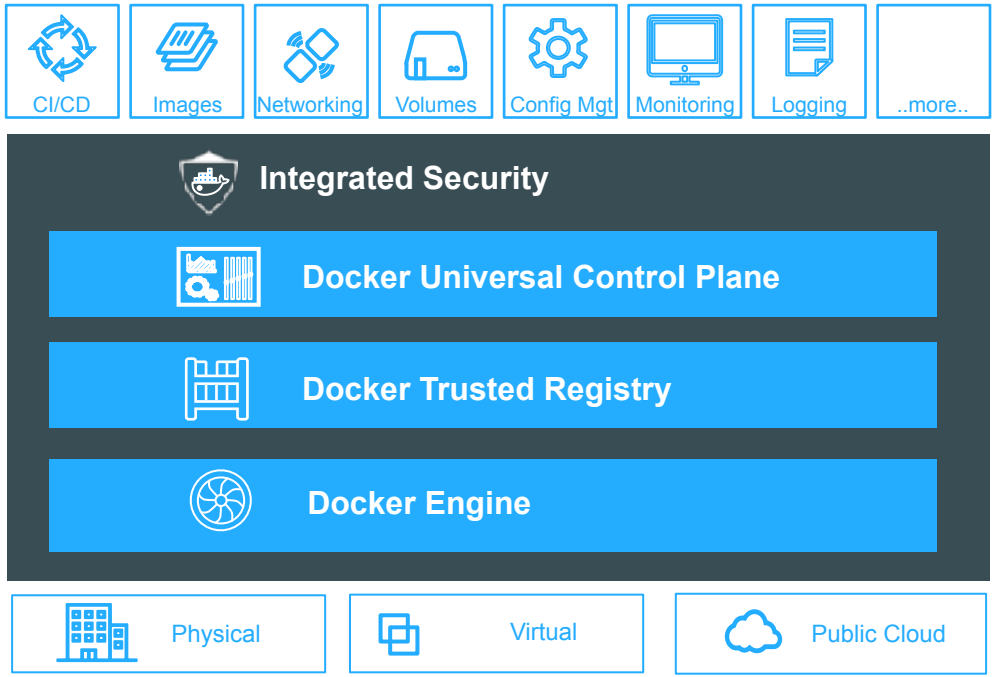
1. **Containerize Legacy Applications**
   Lift and shift for portability and efficiency

2. **Transform Legacy to Microservices**
   Look for shared services to transform



3. **Accelerate New Applications**
   Greenfield innovation

Containers in production with Docker Datacenter



- Enterprise container orchestration, management and security for dev and ops
- Available today for Linux environments
- Q4 2016 beta for Windows environments

docker

Getting started!

Docker on Linux

- Create a Linux VM (or use physical), and install Docker
  - Requires kernel 3.10
- Stable builds
  - `curl –sSL https://get.docker.com/ | sh`
- Test and experimental builds
  - `curl –sSL https://test.docker.com/ | sh`
  - `curl –sSL https://experimental.docker.com/ | sh`
- Can also manually install (see docs)

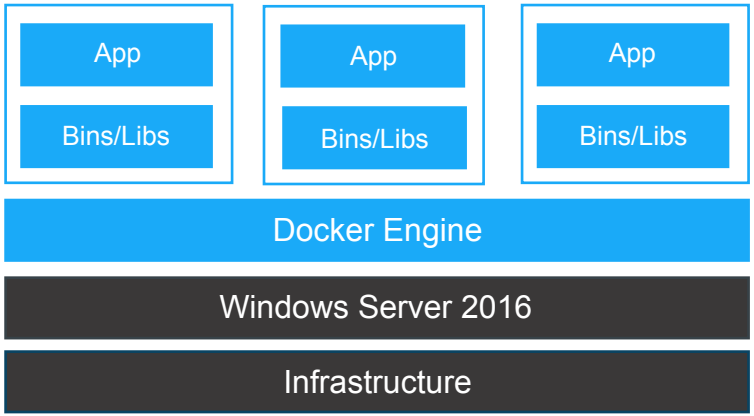
Docker for Windows / Mac

- Currently in public beta
- Easy to install: Get up and running on Docker in minutes
- Leverages Hyper-V (Windows) or xhyv (Mac)
  - Docker for Windows requires Windows Pro 10, Enterprise, or Education
- Full API / CLI compatibility
- OS integration for increased stability and speed

Docker for Azure / AWS

- Currently in private beta
  - https://beta.docker.com/
- Easily deploy Docker 1.12 Swarm clusters (Linux)
- Scale up and down easily
- Integrate with underlying platform (i.e. load balancers)

Docker + Windows Server = Windows Containers



- Native Windows containers powered by Docker Engine
- Windows kernel engineered with new primitives to support containers
- Deep integration with 2+ years of engineering collaboration in Docker Engine and Windows Server
- Microsoft is top 5 Docker open source project contributor and a Docker maintainer

docker

Walk, Jog, Run

Walk:
- Setup your preferred Docker environment
- Fire up some prebuilt images (nginx, hello-world, mikegcoleman/catweb)

Jog:
- Pick a well documented solution (Wordpress, Jenkins, etc)
- Build it for yourself (blogs are your friend)

Run:
- Extend one your Walk solution or Dockerize an existing project
- Build your own Dockerfiles
- Experiment with Docker Compose and Swarm Mode

Thank You.

Questions?

Hands-on Labs

http://github.com/docker/labs/tree/master/beginner


docker