Google Docs Version of These Slides:

https://docs.google.com/presentation/d/15acG7x_gJ9GMi0cxBUU0EAGQxbd3xTZlvUO9UWzR0Fc/edit?usp=sharing

Deploying Hybrid-OS Applications with Docker EE

Presenter Name

Email / Twitter Handle

Before we get started you will need:

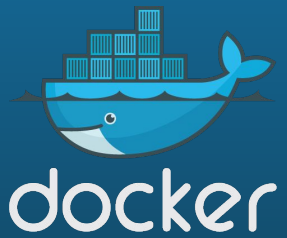
- **RDP client (on Mac: Microsoft Remote Desktop)**
- **SSH client (on Windows: PuTTy)**

docker

What will we be doing today?

- **Slides**: Docker EE Overview
  - **Hands On**: Build an EE Cluster
  - **Hands On**: Deploy a Linux application
- **Slides**: Migrating Traditional Apps with Docker
  - **Hands On**: Migrate and deploy an IIS Web App
- **Slides:** Orchestration capabilities and Docker Compose
  - **Hands On:** Deploy a multi-OS two service app

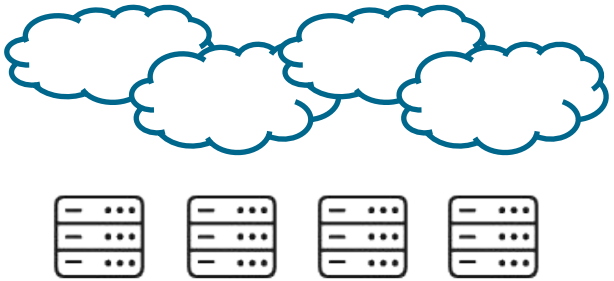
Docker EE Overview



Challenge Fragmented Infrastructure

Four

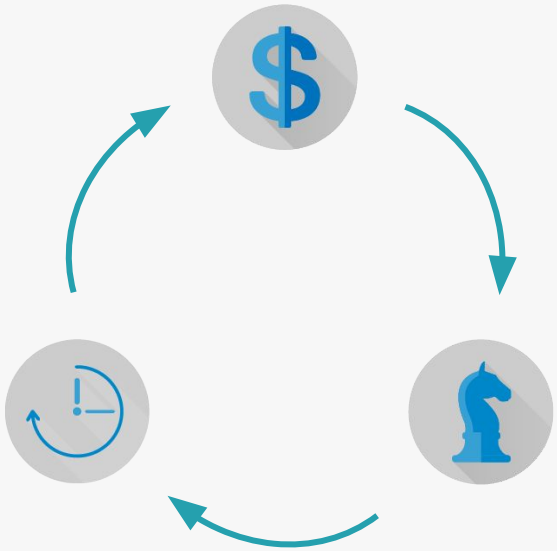
Average number of on-prem platforms (2.3) and public clouds (1.8) actively used by organizations, with another 4 being tested



Challenge: Keeping the Lights On

80% IT Budget

Spent on maintenance & upkeep



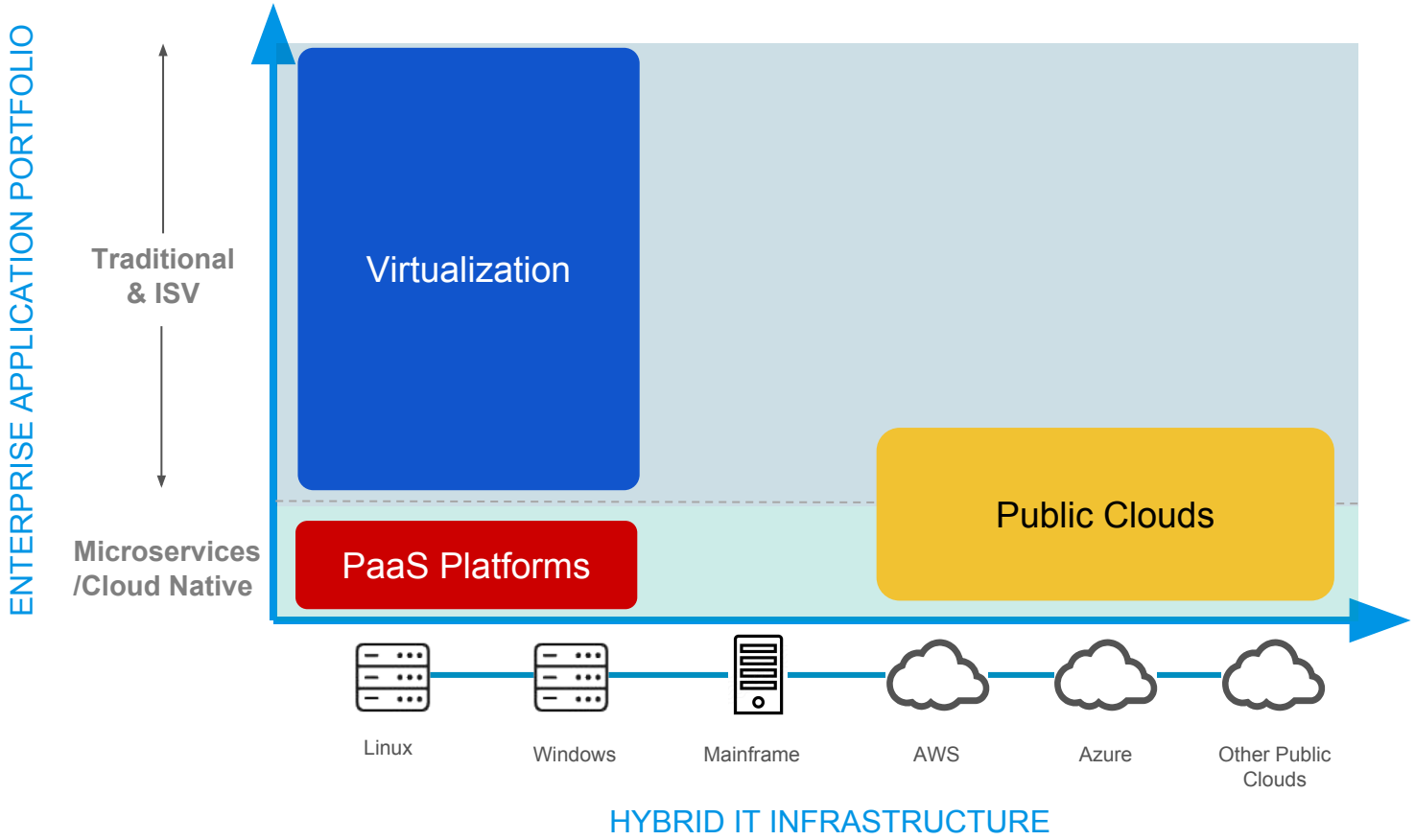
Application Changes

become too complex, difficult, and/or costly to implement

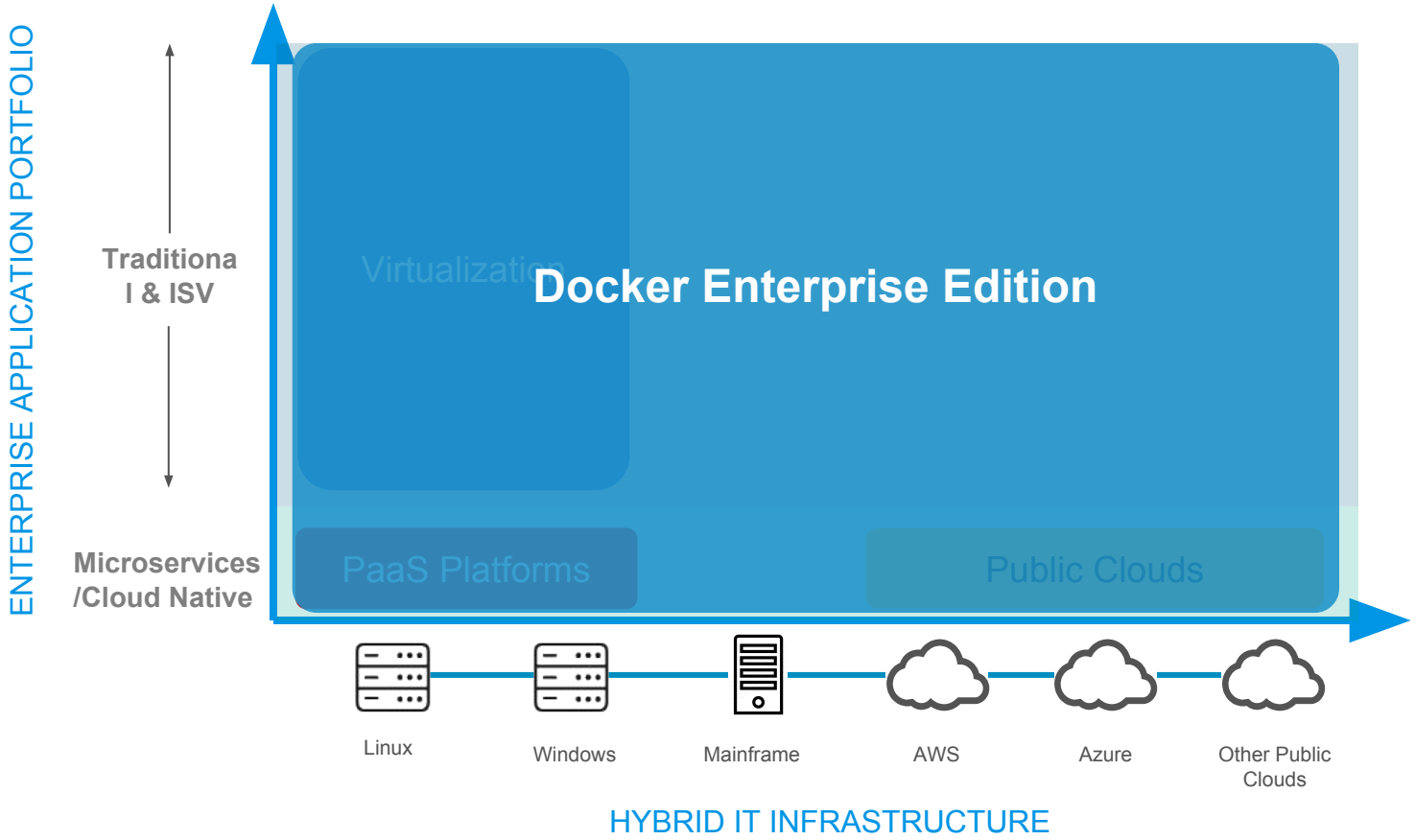
Stuck Keeping the Lights on

Making it difficult to keep up with accelerating standards

Challenge: Narrow Solutions Create Islands



Docker EE: Unify management & security



Docker allows you to invest in tomorrow while you save costs today

Engine for Innovation

Gain a competitive edge through rapid application and infrastructure modernization through a streamlined software supply chain

Driver of Cost Savings

Reduce your infrastructure costs by 50% and maintenance costs by 90%

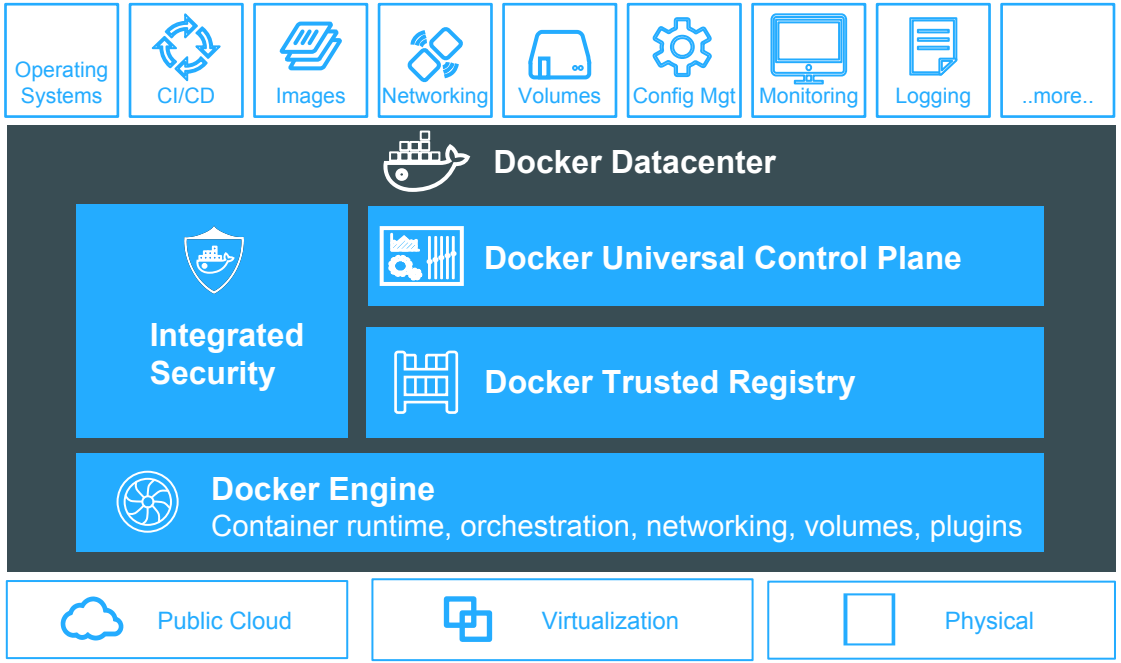
Docker Enterprise Edition Capabilities

Enterprise Edition

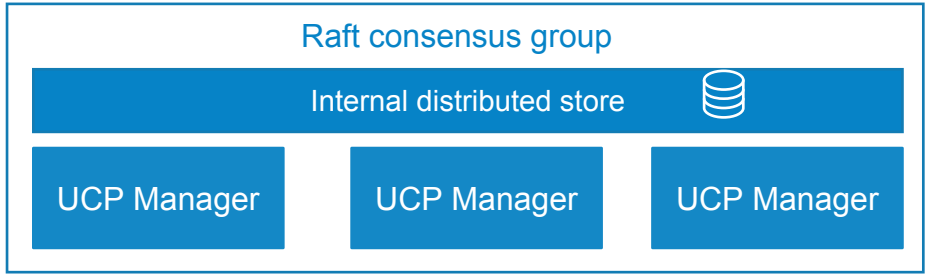
- Certification and Support
- Integrated App and Cluster Management
- Optimized Container Engine

Certified Containers		Certified Plugins
Application Composition, Deployment and Reliability		
Policy Management	Secure Access and User Management	Application and Cluster Management
Image Scanning and Monitoring	Content Trust and Verification	Image Management
Security	Network	Volumes
Distributed State	Container Runtime	Orchestration
Certified Infrastructure		

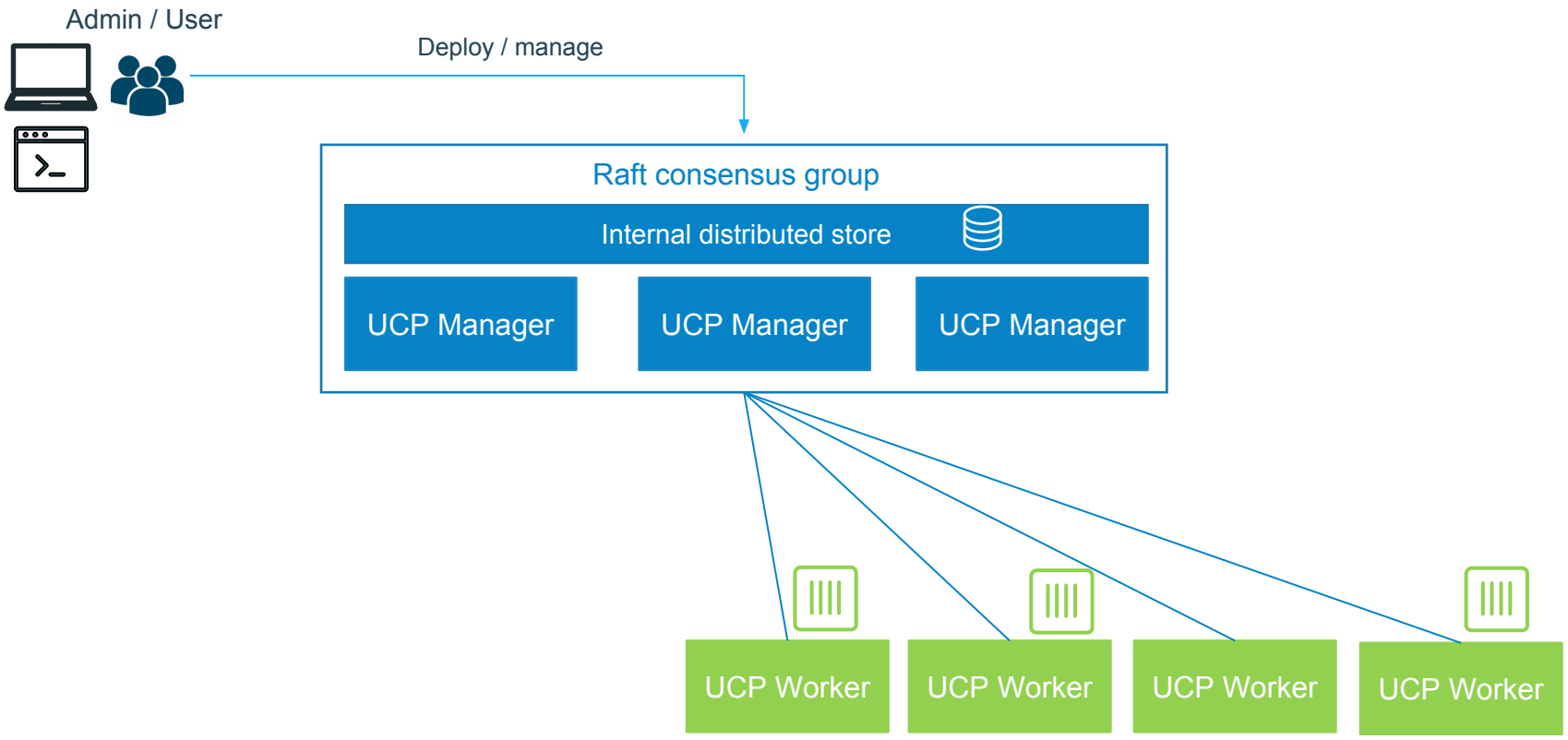
Docker Datacenter



Docker Datacenter Architecture

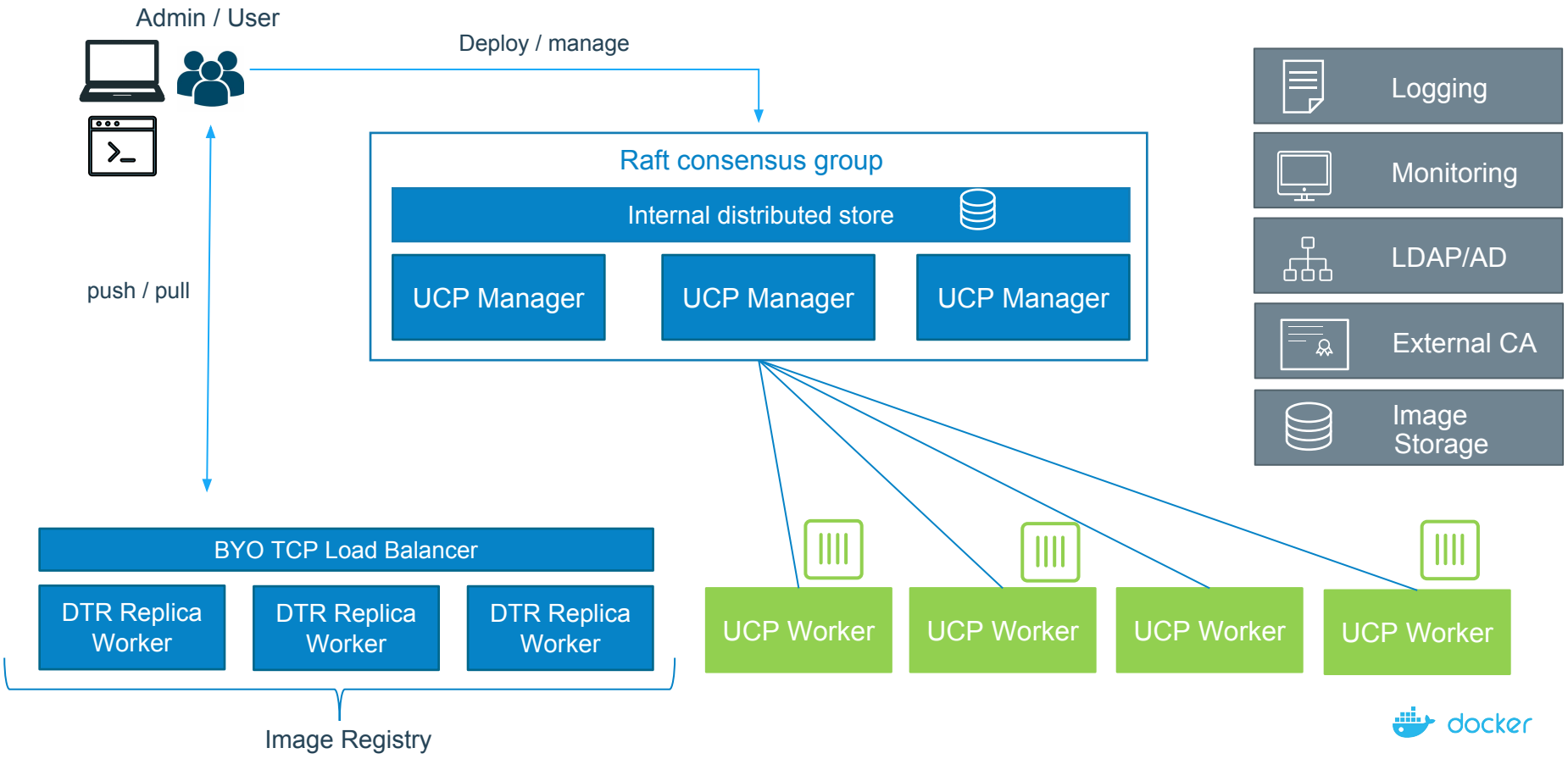


Docker Datacenter Architecture



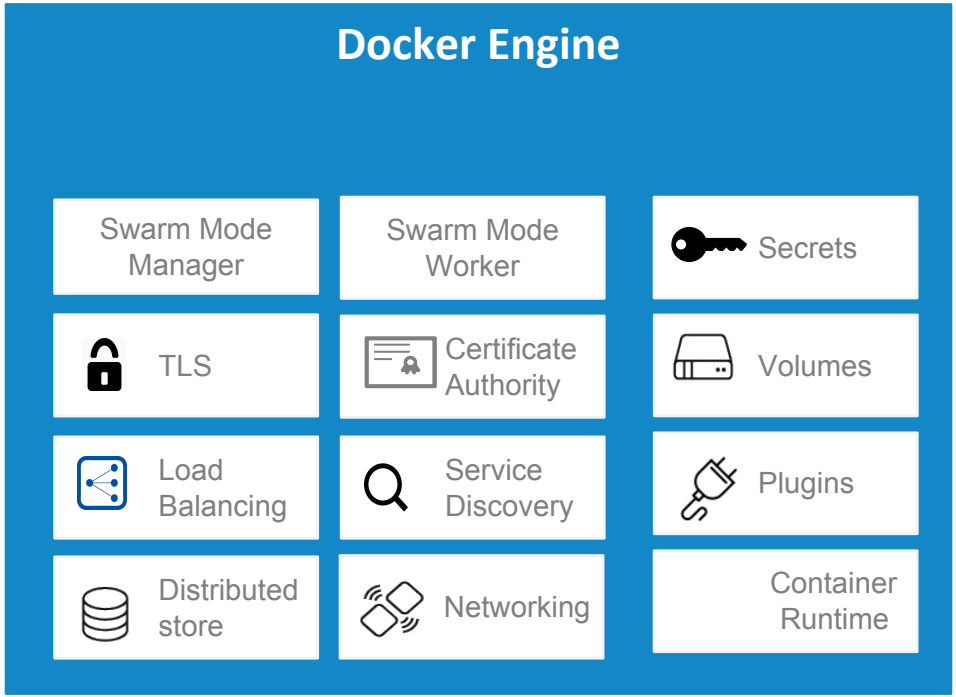
docker

Docker Datacenter Architecture



The building block: Docker Engine

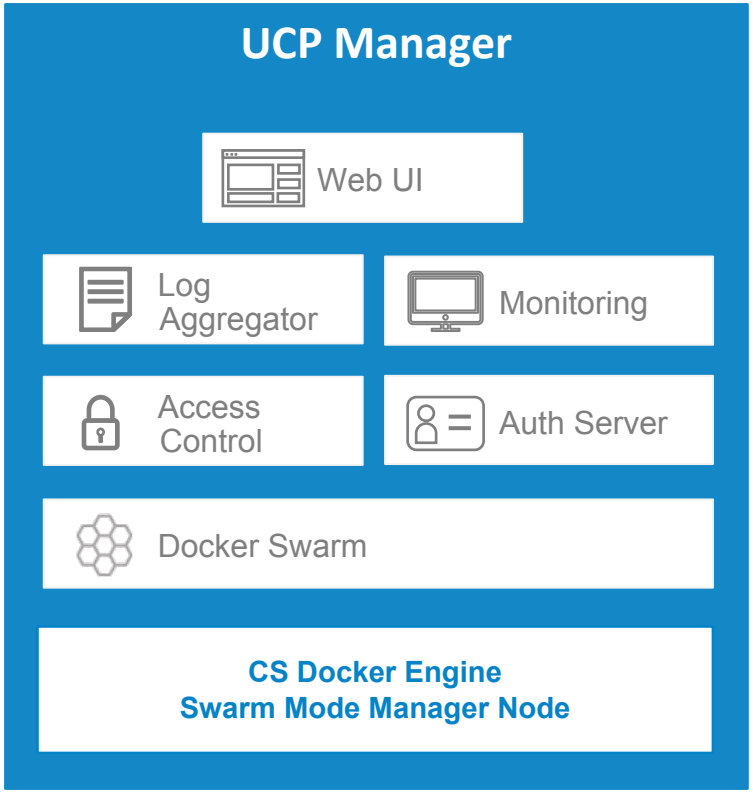
Built in orchestration with scheduling, networking and security



- Powerful yet simple, built in orchestration
- Declarative app services
- Built in container centric networking
- Built in default security
- Extensible with plugins, drivers and open APIs

docker

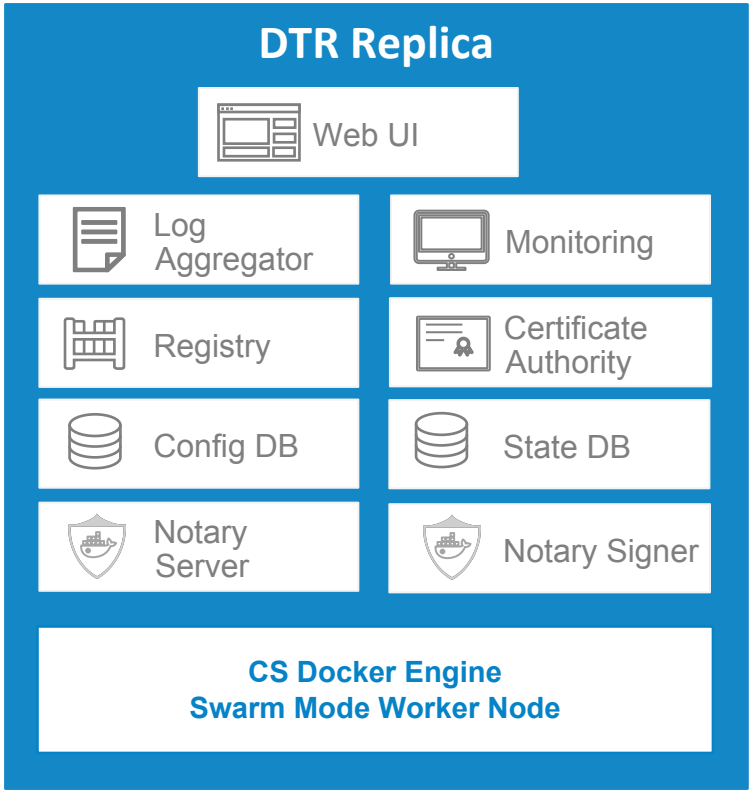
Deep Dive: UCP Manager Nodes



- Point and click UI to manage nodes, services, containers and networks
- CLI and API support
- Secure access control with LDAP/AD support and granular RBAC
- Content security policy

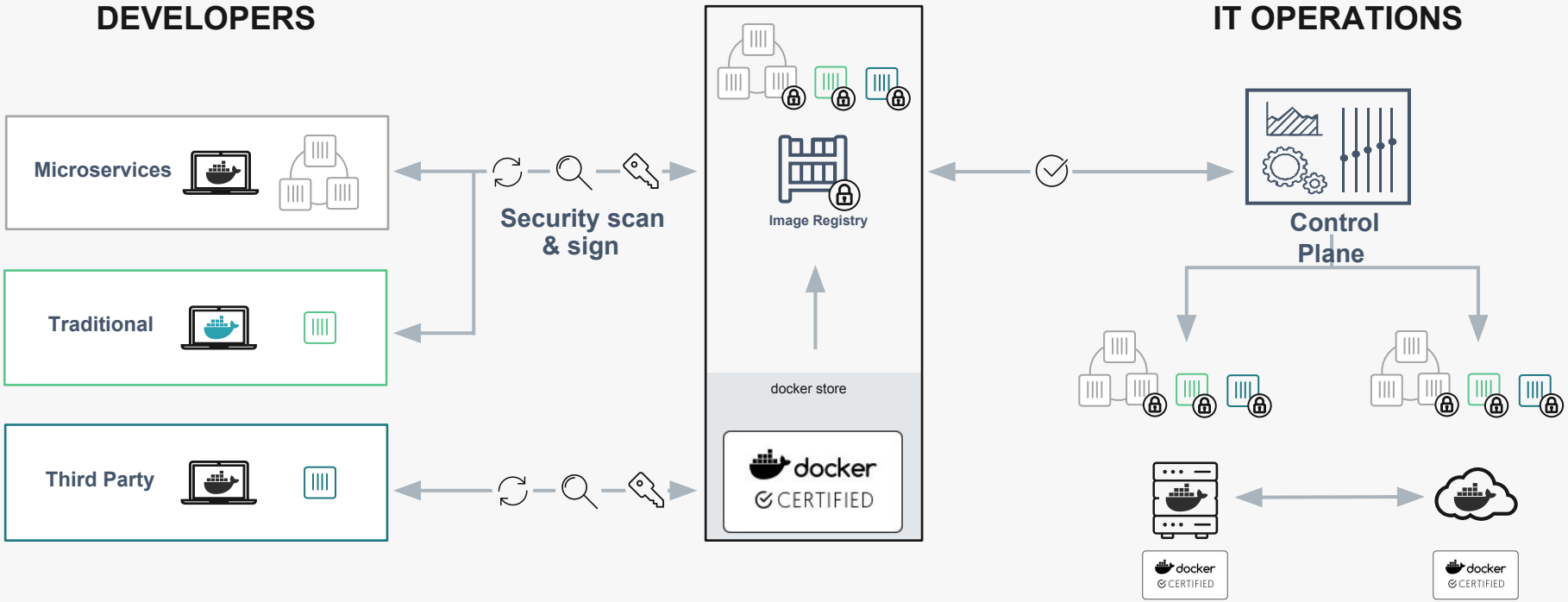
docker

Deep Dive: DTR Replica Worker Nodes



- Point and click UI to manage repos, images and team collaboration
- Image management with labels, tag store and garbage collection
- HA and redundant system
- Content security with built in image signing and verification
- Wide variety of storage driver support for image store

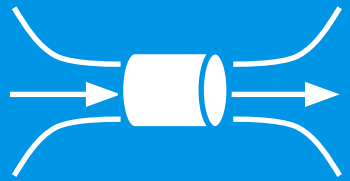
Docker EE CaaS In Action



Unique Advantages of Docker Enterprise Edition

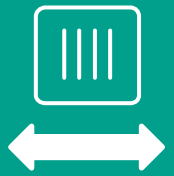
Secure Hybrid Orchestration

- Define application-centric policies and boundaries
- Manage diverse applications across mixed infrastructure with secure segmentation



Secure, Automated Supply Chain

- Streamline the app delivery process across all apps (Linux and Windows, traditional and microservices)

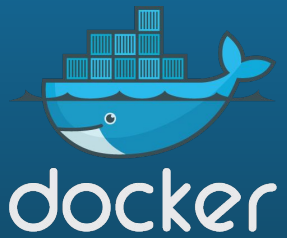


Infrastructure Independence

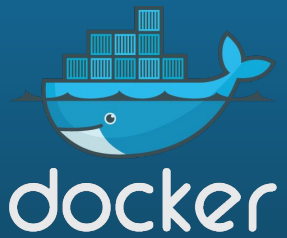
- Consistently manage all applications across any infrastructure
- Easily “lift and shift” apps onto new infrastructure

Exercise 1:
Build a Docker EE Cluster

Exercise 2:
Deploy a Linux application



Migrating Traditional Applications With Docker EE



How Do YOU Get Things To Change

Modernize Traditional Apps with Docker Enterprise Edition to get

portability, security and efficiency of apps without changing the code

You have to cut into the 80%

To Fuel The Innovation

MetLife

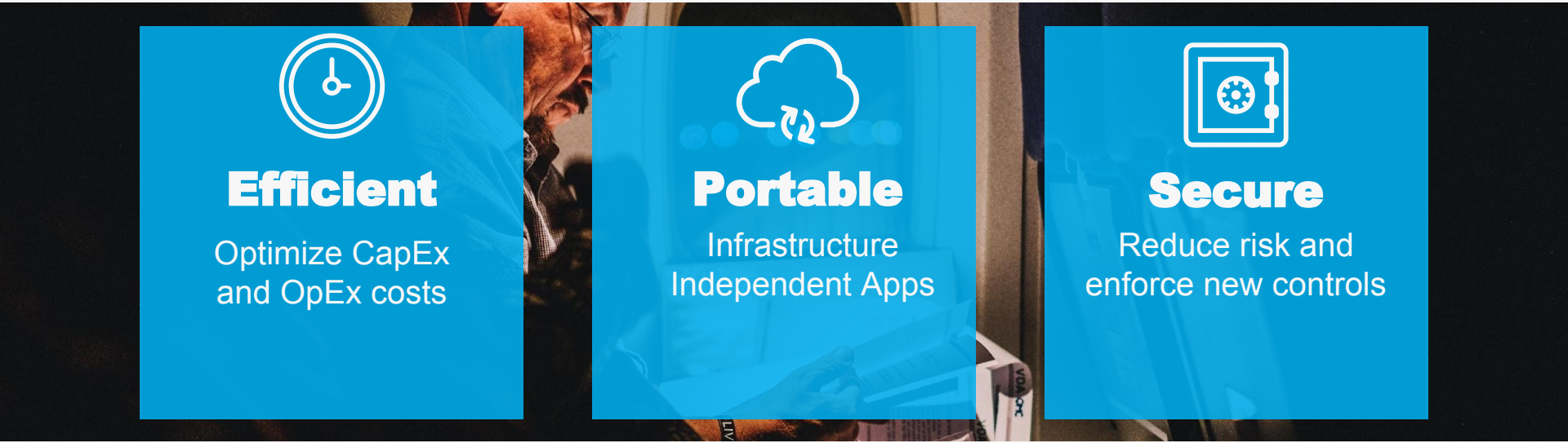
Wanted to modernize one of their legacy java applications running in their datacenter so they could reduce costs and establish a common governance model for all their applications

Priorities going into the MTA Engagement

- TCO reduction
- Common governance model
- Security
- Self-service platform
- Standard deployment model

	Before	After	
Time from Commit to Deploy	7 Days	5 Mins	99% Faster
Infrastructure Utilization	4 Servers	2 Servers	50% Decrease
MTTR for Security Patches	7 Days	2 Hours	84x Decrease in MTTR
Average Time To Scale	7 Days	5 Mins	99% Increase

Docker EE Gives Legacy Applications Modern Capabilities



Size of Infrastructure

77%

Reduction

Deployment Speed

99%

Faster

MTTR for Patching

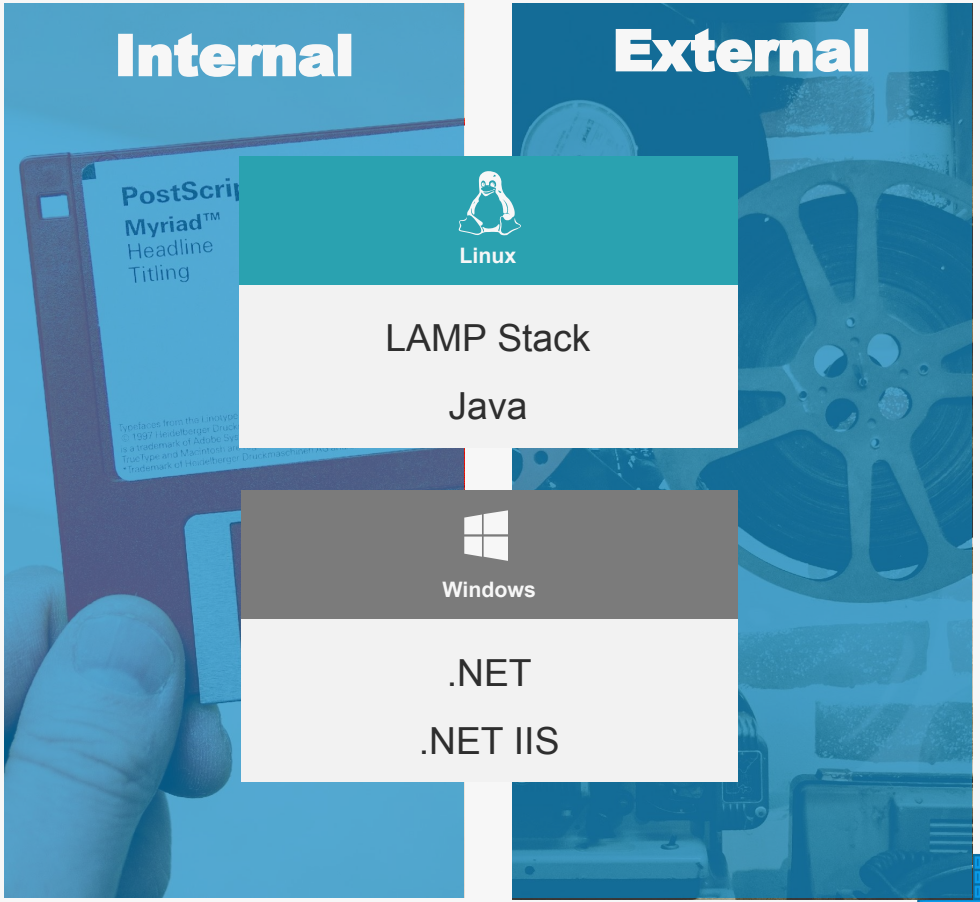
99%

Faster

What Is A Legacy App
Really?

The date in which that code was written isn't the only indicator that you're dealing with a legacy application. There's several other behaviors to keep an eye out for.

- Contains a lot of lost knowledge
- Most updates are band-aid fixes.
- Dynamic scaling isn't possible, or takes way too long.
- Maintenance windows are a quarterly or bi-annual event. And contain more anxiety than confidence



Methodology: Docker EE Modernizes
Apps and Infrastructure

The quickest way to cut into that 80%

Existing Application

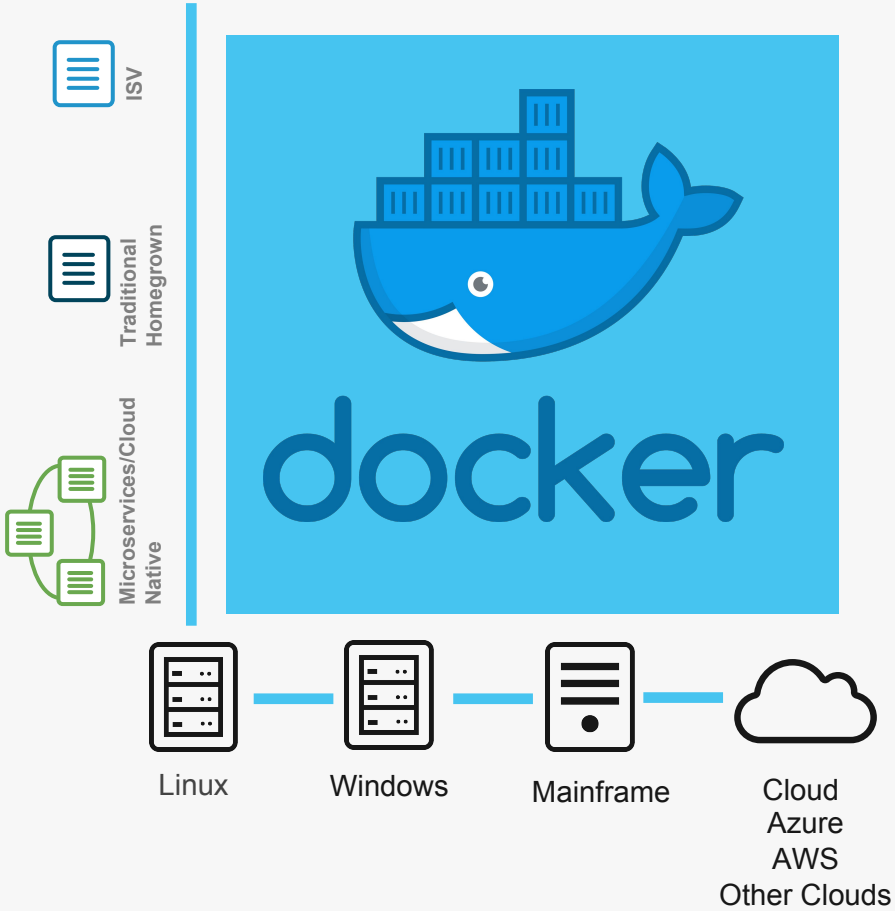
Convert to a **container** with Docker EE

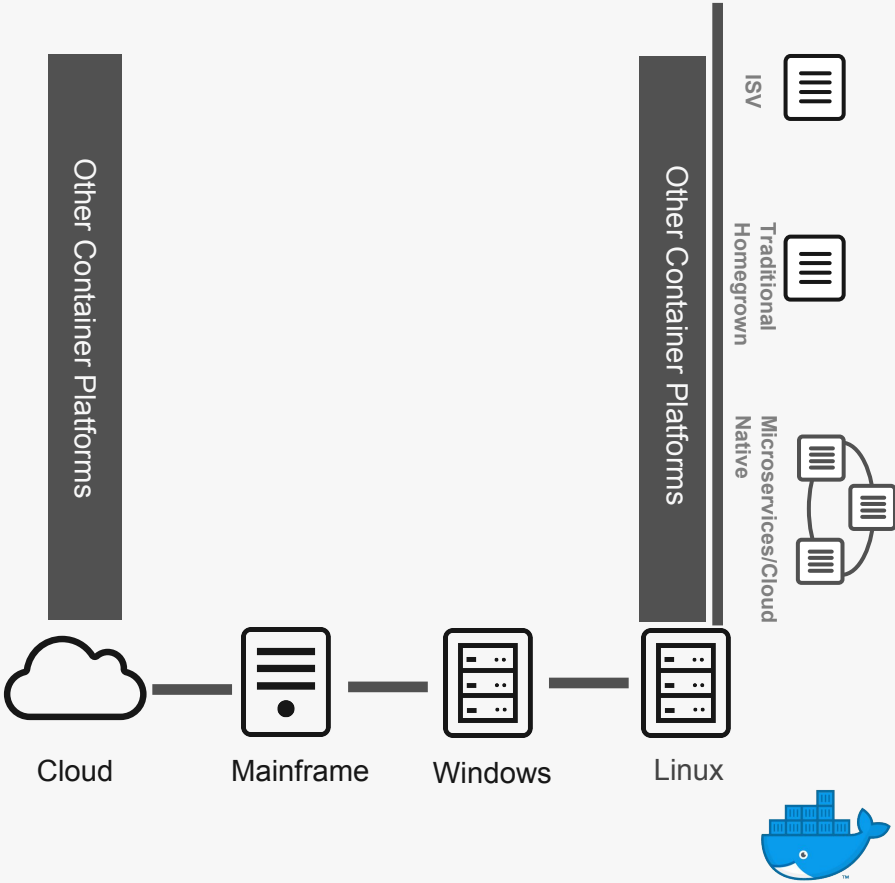
Modern Infrastructure
Built on premise, in the cloud, or as part of a hybrid environment.

Modern Methodologies
Integrate to CI/CD and automation system

Modern Microservices
Add new services or start peeling off services from monolith code base

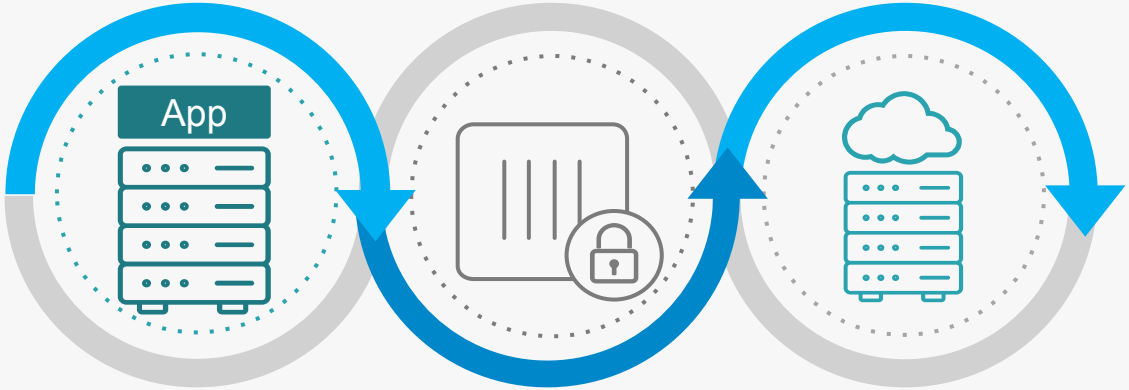
Why Docker for MTA





Our Exercise Today

The quickest way to cut into that 80%



Existing Application

Convert to a **container** with Docker EE

Modern Infrastructure
Built on premise, in the cloud, or as part of a hybrid environment.

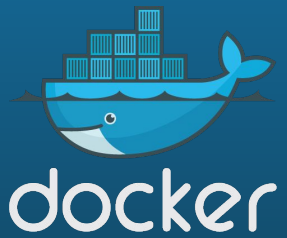
Migrate and Deploy a Windows App

- Simple static web app built on IIS stored inside a VM
- Image2Docker creates Dockerfiles from VMs
- Supported Artifacts
  - Microsoft Windows Server Roles and Features
  - Microsoft Windows Add/Remove Programs (ARP)
  - Microsoft Windows Server Domain Name Server (DNS)
  - Microsoft Windows Internet Information Services (IIS)
  - Apache Web Server

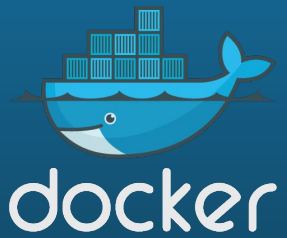
Windows Tweet App Partial Dockerfile

```
FROM microsoft/windowsservercore

RUN Add-WindowsFeature Web-Server

EXPOSE 80

RUN Set-WebConfigurationProperty -pspath 'MACHINE/WEBROOT/APPHOST' -filter
 'system.applicationHost/log' -name 'centralLogFileMode' -value 'CentralW3C';

WORKDIR C:\

COPY start.ps1 .
COPY index.html C:\inetpub\wwwroot
COPY windows.png C:\inetpub\wwwroot

CMD .\start.ps1
```

Exercise 3:
Migrate and deploy a Windows web app



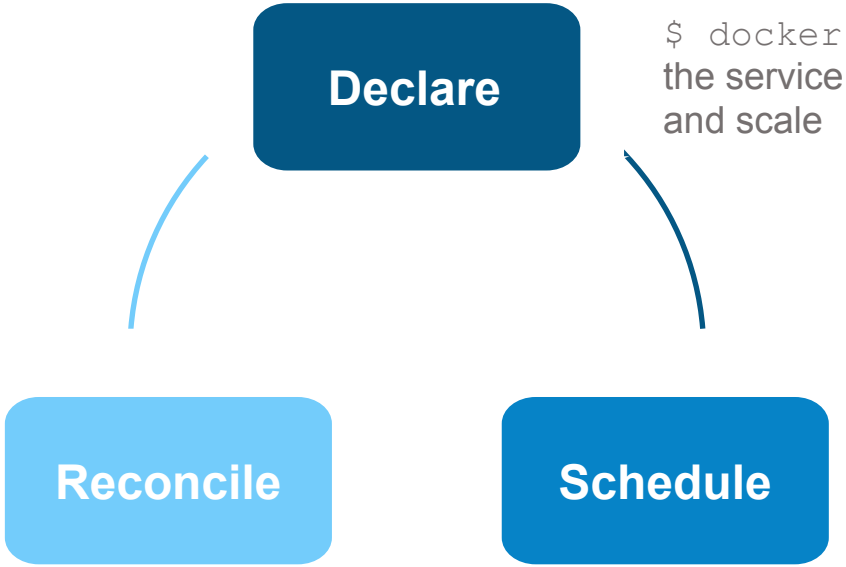
Orchestration Capabilties & Docker Compose



Services \ Tasks

- Services provide a piece of functionality
  - Based on a Docker image
  - An application is made up of 1-n services
- Replicated Services and Global Services
- Tasks are the containers that actually do the work
  - A service has 1-n tasks

How service deployment works



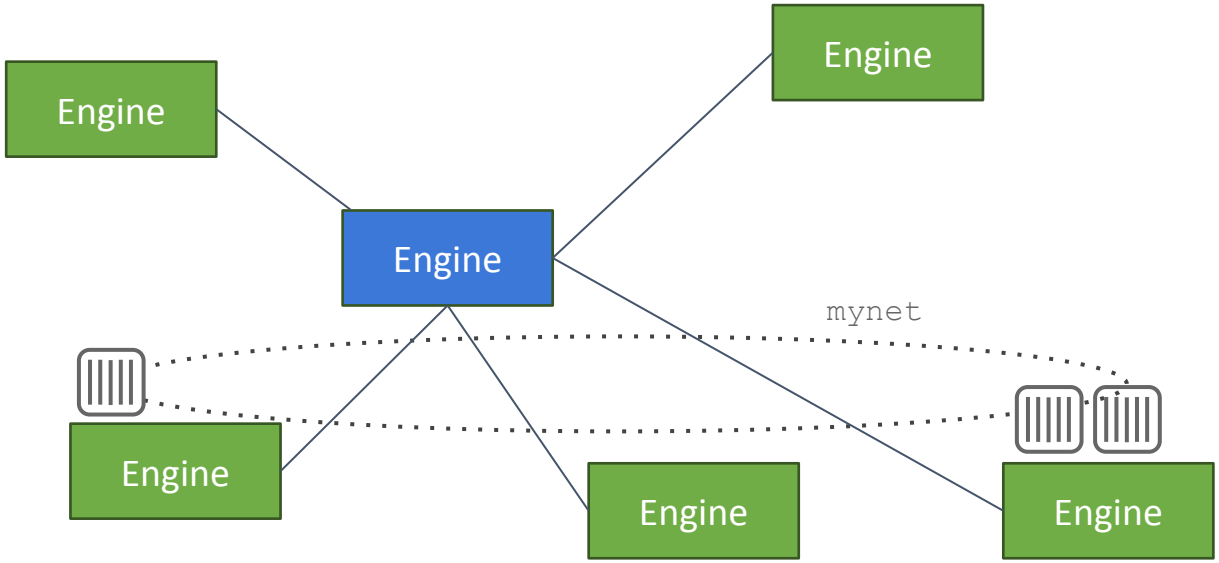
`$ docker service create` declares the service name, network, image:tag and scale

Managers break down service into tasks, schedules them and workers execute tasks

Engines check to see what is running and compared to what was declared to “true up” the environment

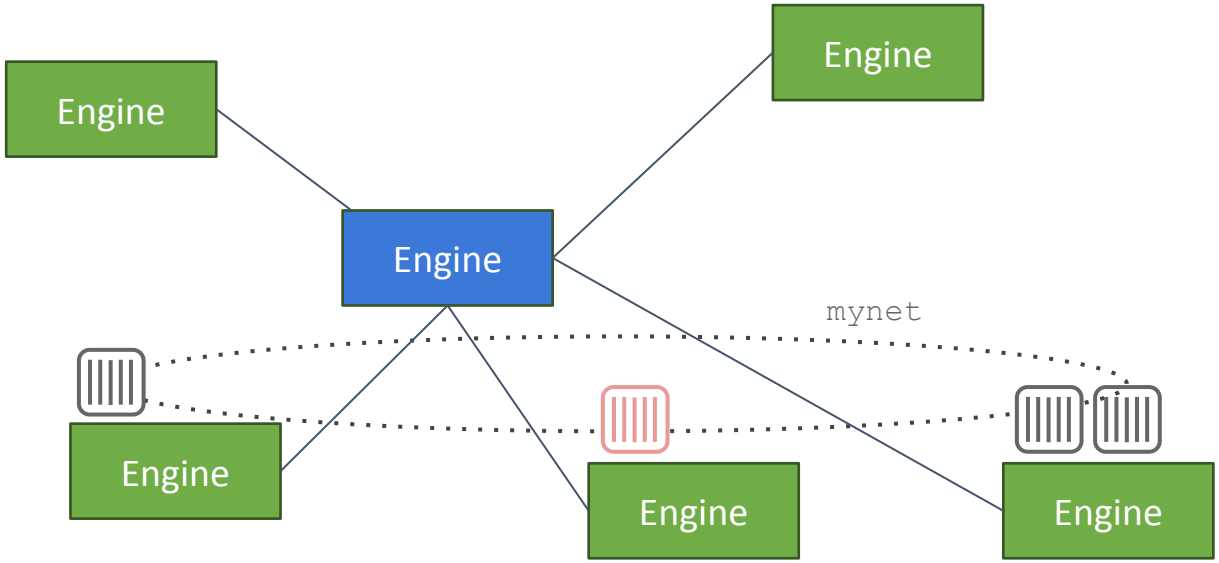
docker

Services




```
$ docker service create --replicas 3 --name frontend --network mynet
 --publish 80:80/tcp frontend_image:latest
```

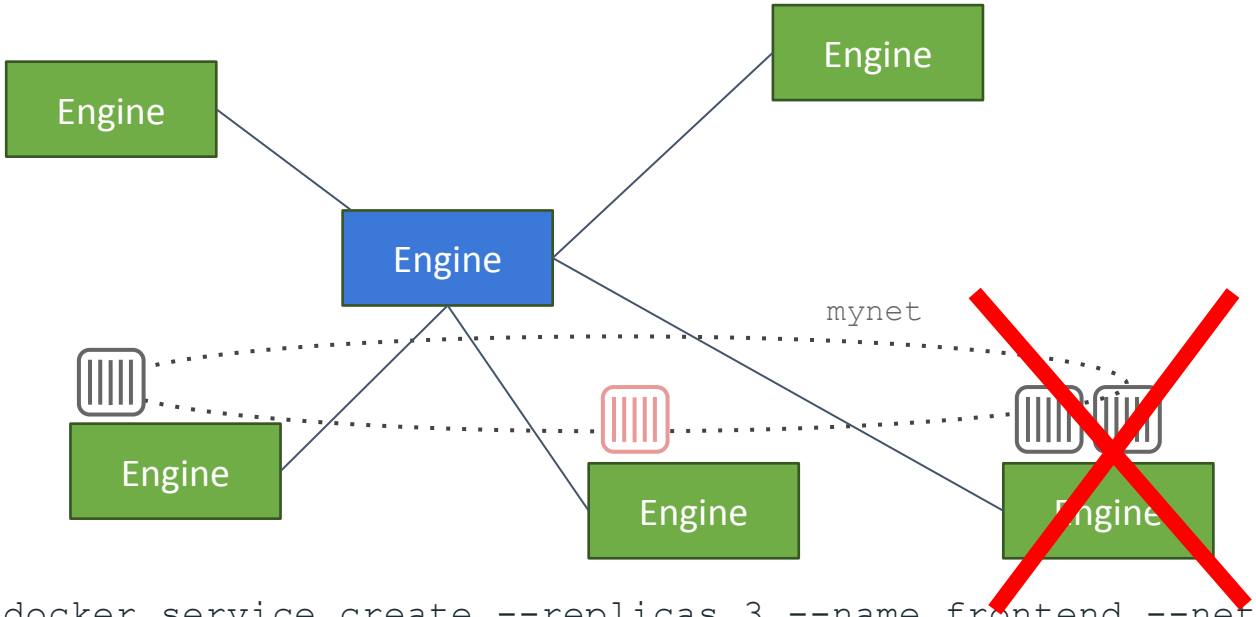
Services




```
$ docker service create --replicas 3 --name frontend --network mynet
 --publish 80:80/tcp frontend_image:latest
```

```
$ docker service create --name redis --network mynet redis:latest
```

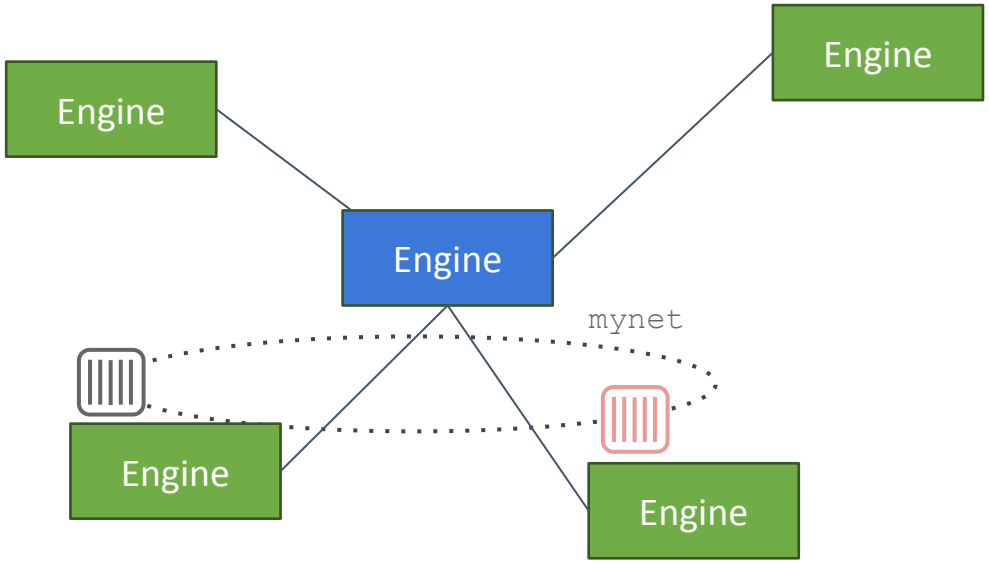
Node Failure




```
$ docker service create --replicas 3 --name frontend --network mynet
 --publish 80:80/tcp frontend_image:latest
```

```
$ docker service create --name redis --network mynet redis:latest
```

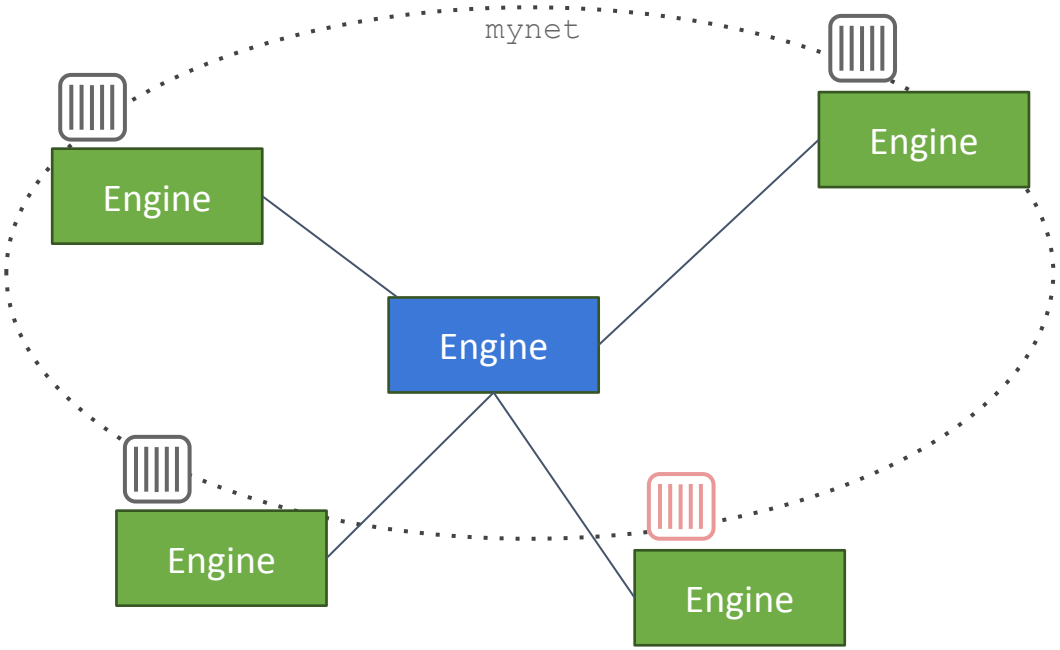
Desired State ≠ Actual State




```
$ docker service create --replicas 3 --name frontend --network mynet
 --publish 80:80/tcp frontend_image:latest
```

```
$ docker service create --name redis --network mynet redis:latest
```

Converge Back to Desired State



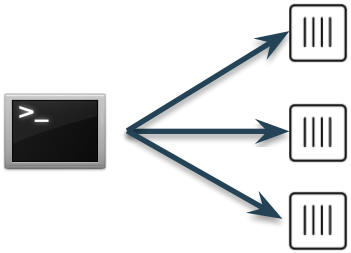

```
$ docker service create --replicas 3 --name frontend --network mynet
 --publish 80:80/tcp frontend_image:latest
```

```
$ docker service create --name redis --network mynet redis:latest
```

Docker Compose: Multi Container Applications

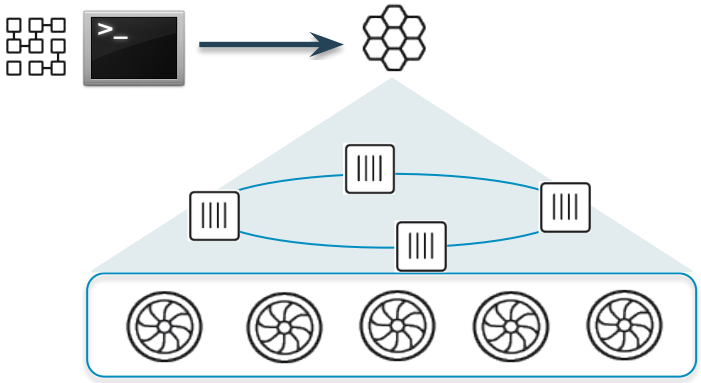
Without Compose

- Build and run one container at a time
- Manually connect containers together
- Must be careful with dependencies and start up order

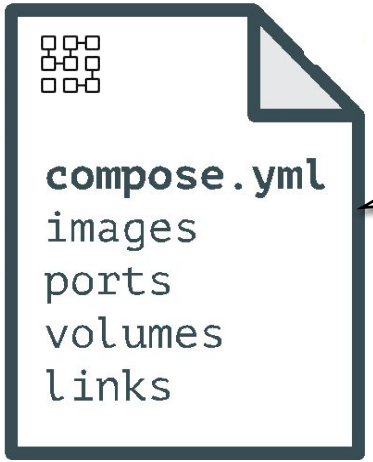


With Compose

- Define multi container app in compose.yml file
- Single command to deploy entire app
- Handles container dependencies
- Works with Docker Swarm, Networking, Volumes, Universal Control Plane



Docker Compose: Multi Container Applications

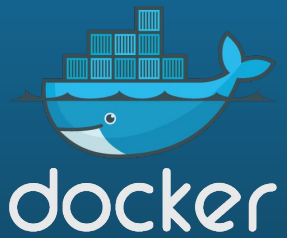


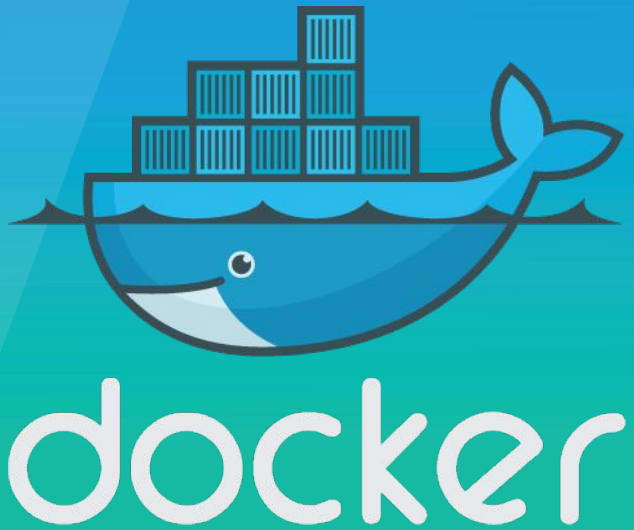

```yaml
containers:
  web:
     build: .
     command: python app.py
     ports:
     - "5000:5000"
     volumes:
     - .:/code
     environment:
     - PYTHONUNBUFFERED=1
  redis:
     image: redis:latest
     command: redis-server --appendonly yes
```

Stacks: Multi-Service Applications

- A stack is a collection of related services
- Stacks are a Docker primitive
  - `docker stack deploy`
  - `docker stack ps`
  - `docker stack rm`
- Implemented via a docker compose file

Exercise 4:
Deploy and Manage a Hybrid-OS Application




docker