



Module One

Intro to Docker and Containers

Jay Schmidt

- Over twenty years of experience in tech
- Experience in development, administration, operations, product management, and project management
- Container advocate and user since the beginning
- Believes that technology is there to solve real world problems not to be impressive



Docker's vision

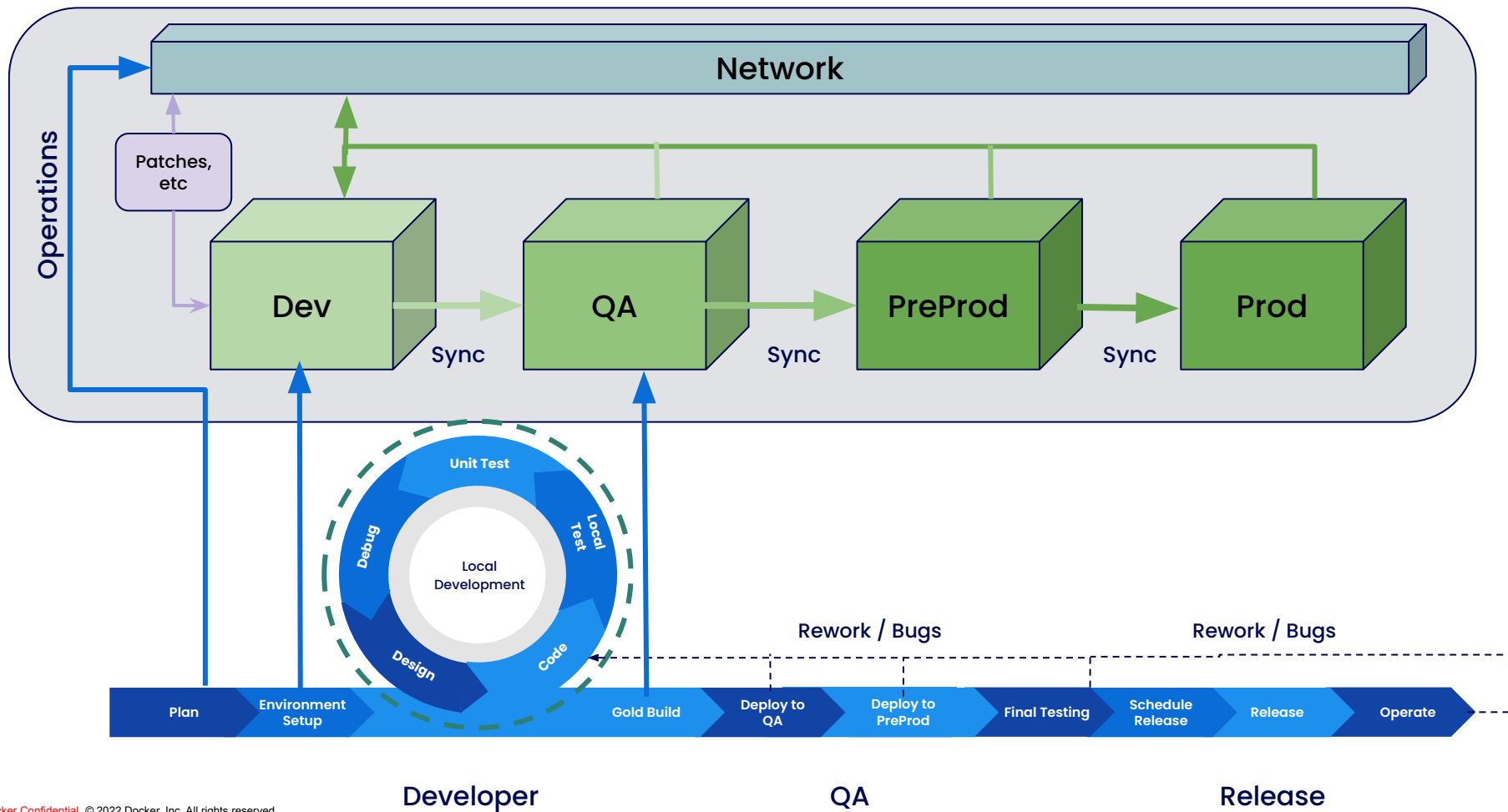
Increase the time developers spend on innovation, and decrease the time they spend on everything else



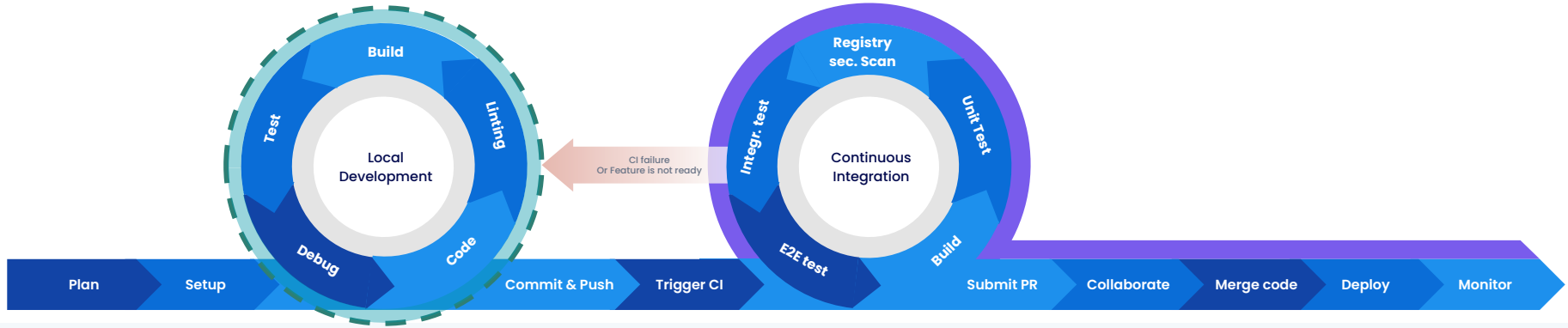


Why Change?

Before Containerization



After Containerization





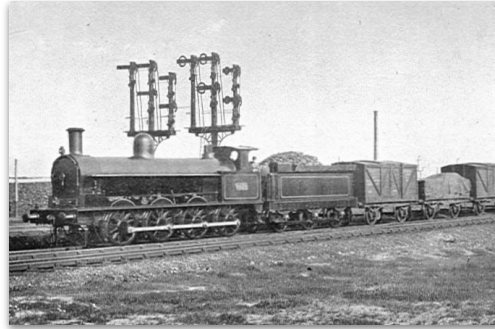
Why Containers?

Shipping over the years



Shipping by sea

Slow. Unstandardized. High chance of loss and theft.



Industrial revolution

Faster. Still unstandardized.
Highlights inefficiencies.



Shipping container

Standardization causes massive improvement to throughput



Container (Shipping) Specifications

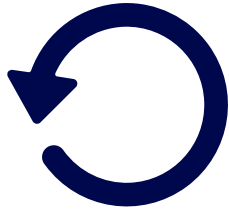
- Allows for interoperability with all users / vendors that follow the ISO Specification
- Codified under ISO 6346:1995
- Container dimensions are regulated under this code
 - Height:
 - Standard containers are 8 ft. 6 in
 - Other heights measuring from 4 ft. to 9 ft. 6 in.
 - Width
 - The majority of ISO containers have a width of 8 ft.
 - C, D, E, and F 8ft. to 8ft. 2.43in
 - L, M, N, and P exceeding 8ft. 2.43in.
 - Length
 - Common container lengths include 20 ft. and 40 ft.
 - Other available lengths comprise 24 ft., 28 ft., 44 ft., 45 ft., 46 ft., 53 ft., and 56 ft.



Note: This is illustrative; please don't do a deep dive into Shipping Containers.



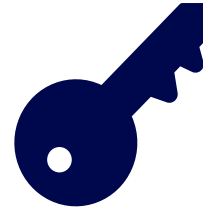
Net results



Port turnaround
times dropped
from 3 weeks to
24 hours



Shipping costs
dropped from
\$5.86/ton to
\$0.16/ton, a 97%
reduction in cost



Theft/damage
dropped
significantly due to
fewer touches

Shipping software



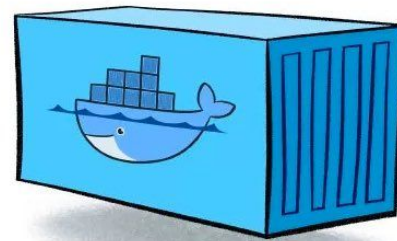
Per-app deploys

Slow. Unstandardized. High chance of drift and misconfiguration.



Mobile/internet age

Need to deploy more often.
Highlights gaps and environmental issues.



Container spec

Standardized packaging makes it easy to build, share, and run applications

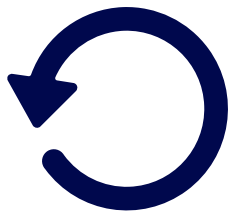


Container (Software) Specifications

- Docker created the Open Container Initiative in June 2015
- Currently owned by the Linux Foundation
- Currently defines three specifications
 - **image-spec** - defines image structures and manifests
 - **runtime-spec** - defines how to run OCI images
 - **distribution-spec** - defines the API protocol to push, pull, and discover content



Positive results



93% of survey respondents reported accelerated application development and deployment with containers and Kubernetes and a 26% increase in developer productivity.

CNCF, 2020 /
Portworx



87% of surveyed organizations reported cost savings after containerizing along with 6x higher availability

451 Research, 2020
Sysdig



Up to 50% increase in server utilization, along with 21% less deployments with containers.

Diamanti, Netflix



Where are containers?

Dev environments Microservices
Software development Monoliths
In production CI/CD pipelines

Education

Classroom environments Reproducible research

Data sciences Edge computing
Emerging fields

AI/ML model training Distribution/usage of models

Home automation

Private game servers

DIYers

Media streaming

Network monitoring

NAS servers

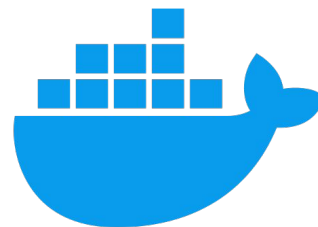


Why does OCI Matter?

OCI is the cornerstone of containerization, providing standardization and interoperability that fuels the container ecosystem's growth and ensures portability, security, and long-term stability.



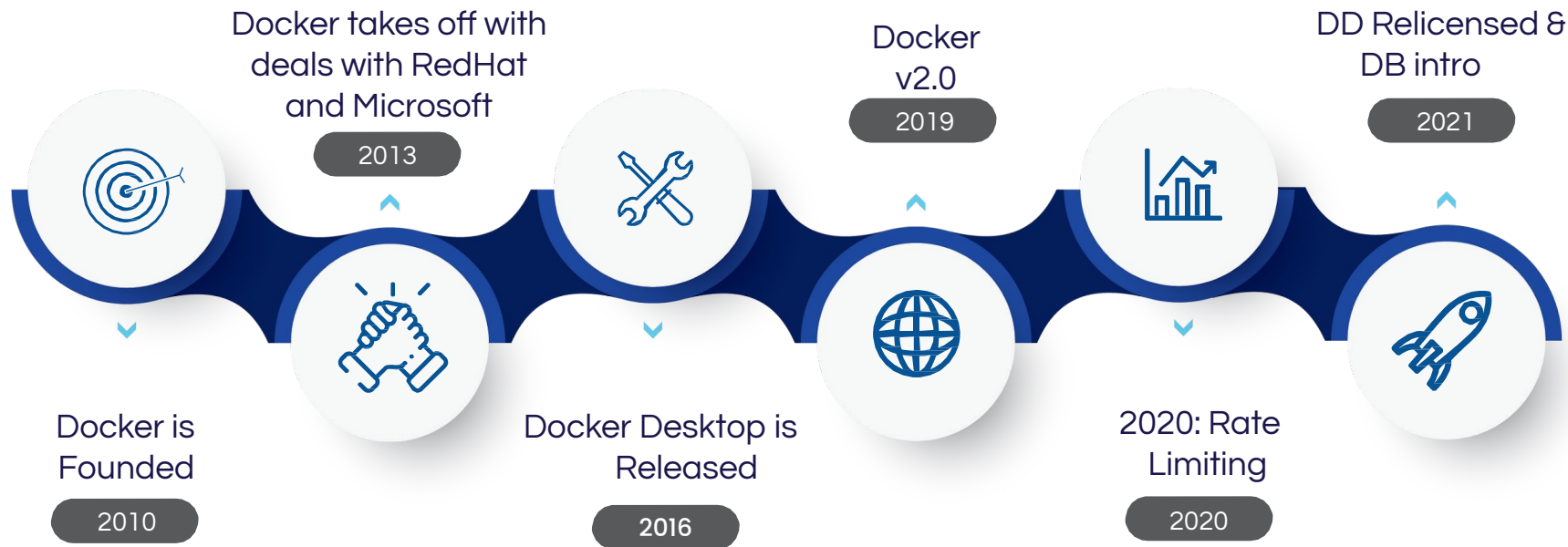
Some OCI Compliant Tooling





About Docker...

Brief History Timeline



At **Docker** We Believe It Is
Critical To **Empower**
Developers

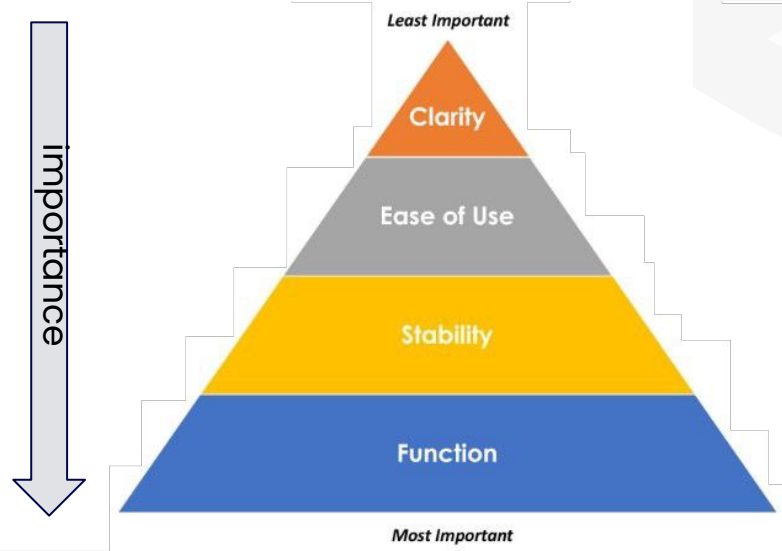


**Developer Experience Is A
Leading Indicator Of A
Successful Company**



Side Effects Of Bad Dx

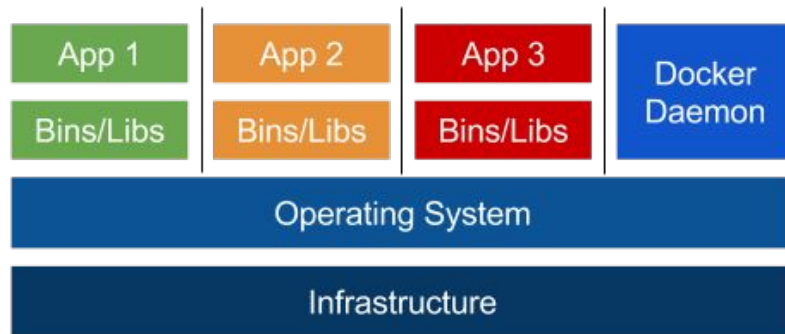
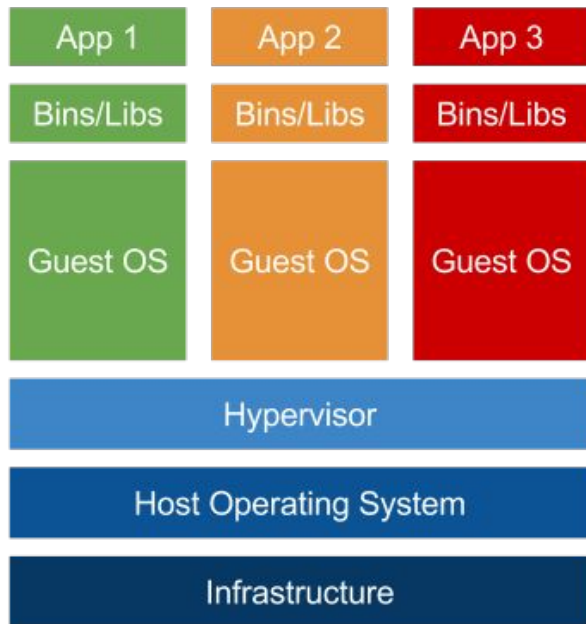
- Inability To Retain Top Talent
- Buggy Software
- Change Is Slow
- Low Customer Satisfaction
- Outages
- Excessive Spending





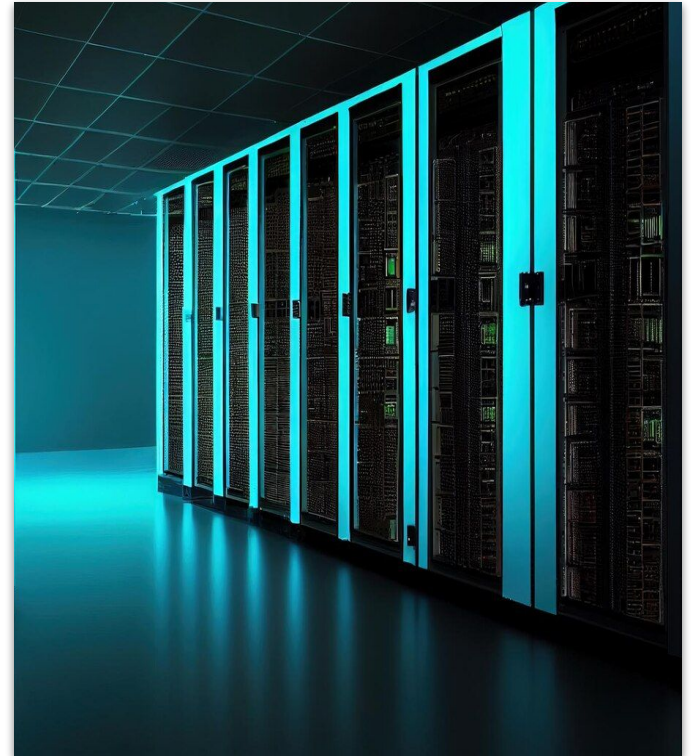
Containerization Benefits

But What About VMs?



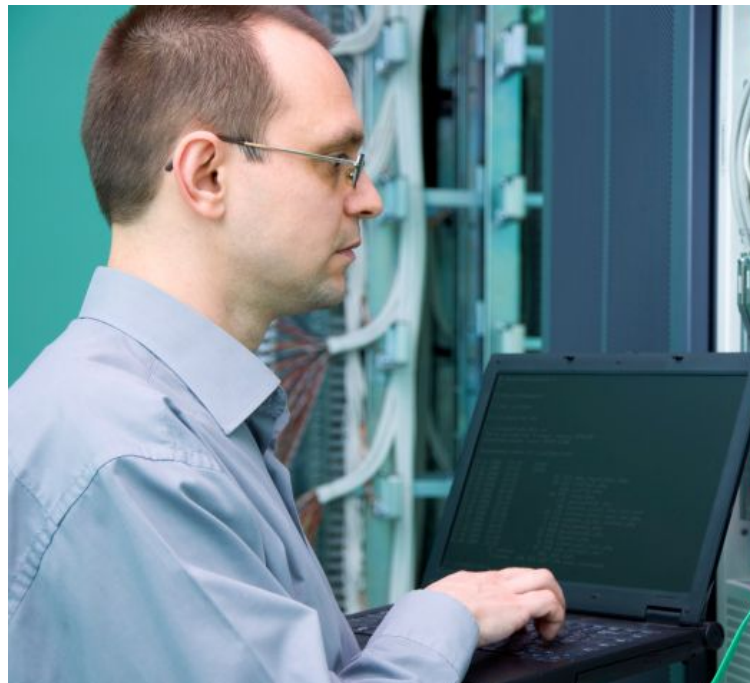
Reduction in Infrastructure Costs

- Containers are more efficient than virtual machines
 - Shared host kernel
 - Easier to share resources
- Containers reduce overhead
 - Each container is self-contained
 - Host is only responsible for container runtime



Reduction in Management Costs

- Containers are easier to manage
 - Versioned
 - Simple Migration Path
- Easier Host Management
 - OCI Runtime
 - Storage
- Easier Scalability
 - Add/Remove Containers



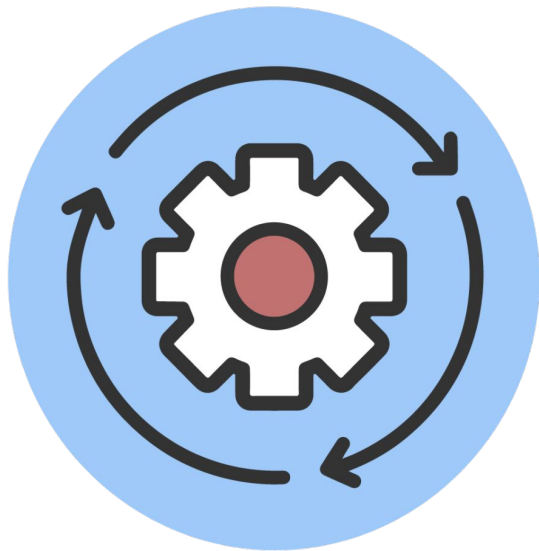
Developer Velocity

- Use Prebuilt Images
 - Official / Verified Images
 - Easier Upgrades
- Reduce “Cold Start” Problem
 - Stand Up / Tear Down Environments
- Consistent Environment
 - “Works on My Machine”



Moving to Production

- Many OCI Compliant Options
 - Kubernetes / OpenShift
 - NOMAD
 - Serverless
- Large Ecosystem of Tooling
 - Management
 - Security
 - Authentication / Management
 - Virus / Malware



Other Options

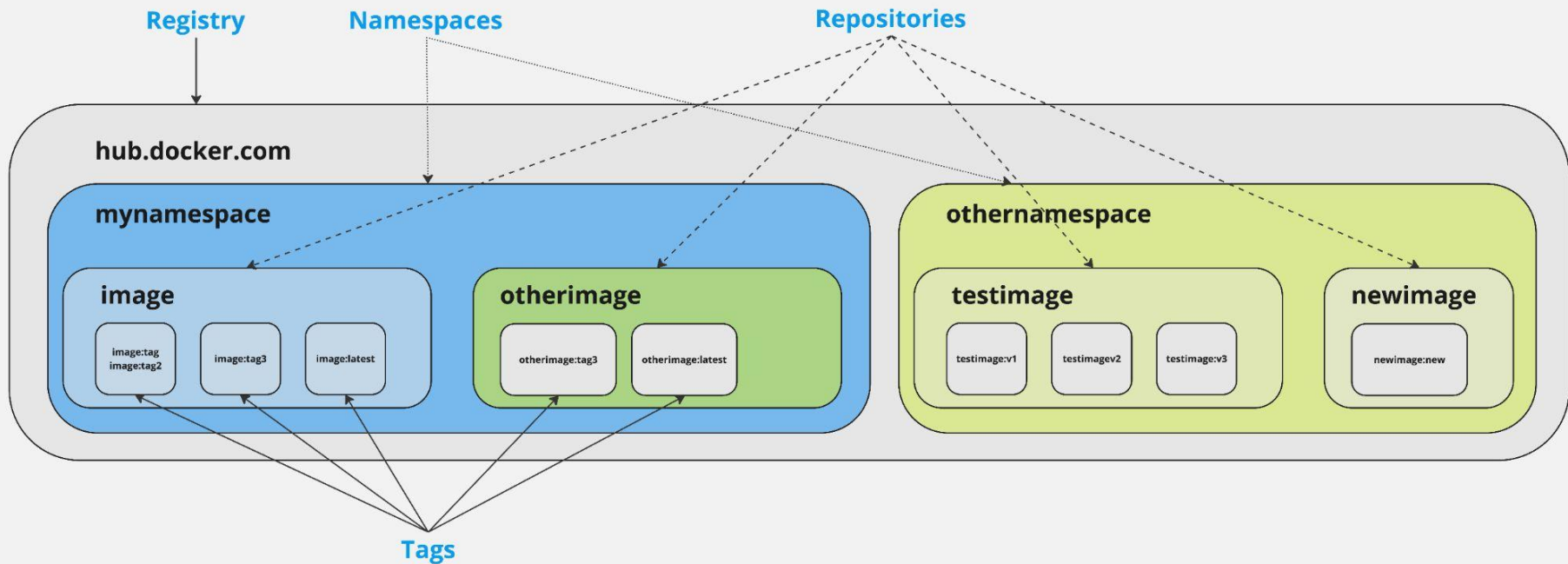
- Virtual Machines
 - Resource Overhead
 - Management Overhead
- Cloud Based VMs
 - Management Overhead
 - Price
- Physical Servers
 - Resource Overhead
 - Lack of Flexibility
 - Management Overhead
- Development Systems
 - Configuration Management
 - Resource Management





Important Terms and Definitions

Registries, Repos, Repositories, Oh My!



Code Repository

- A storage location for source code
 - Code is pushed/pulled from the repository during its lifecycle
 - Can be SaaS or Local
 - Can be Public or Private
- Examples:
 - Github
 - Gitlab
 - BitBucket



OCI Image

- An image that complies with the OCI specifications
- Other names
 - Docker Image
 - Container Image
- Includes everything necessary to create / run a piece of software
 - Running Image == Container
- Built from Dockerfile + Base Images + Code / Files / Packages
- Images are annotated with tags
 - "latest"
 - "test"
 - "v1.1"



Image Registry

- A storage location for container images
 - Many repositories exist inside a container registry
 - Images are pushed/pulled from a repository in the registry
 - Can be SaaS or Local
 - Can be Public or Private
- Examples:
 - Docker Hub
 - Amazon ECR
 - Harbor
 - JFrog Artifactory



Namespace

- A string used as a container for repositories
 - Can be a username
 - Can be an organization name
 - Multiple namespaces exist in a registry
 - Each namespace has one to many repositories
- Examples
 - Docker Hub User Namespace: jayschmidt
 - Docker Hub Organization Namespace: virington



Image Repository

- A collection of related container images
 - A repository can contain many images
 - Images can have one to many tags
 - Repositories live inside a registry
- A fully qualified image consists of:
 - `registry/namespace/repository:tag`
- Examples
 - Registry: `hub.docker.com`
 - Namespace: `jayschmidt`
 - Repository: `ratg`
 - Tag: `latest`
 - All together: `hub.docker.com/jayschmidt/ratg:latest`



Container

- A running instance of a container image
- Also called
 - Docker Container
 - OCI Container
- OCI images run on any OCI compliant runtime environment
 - “containerd”
 - “runc”
 - “Sysbox”
- Isolates the application and its environment to run on any machine

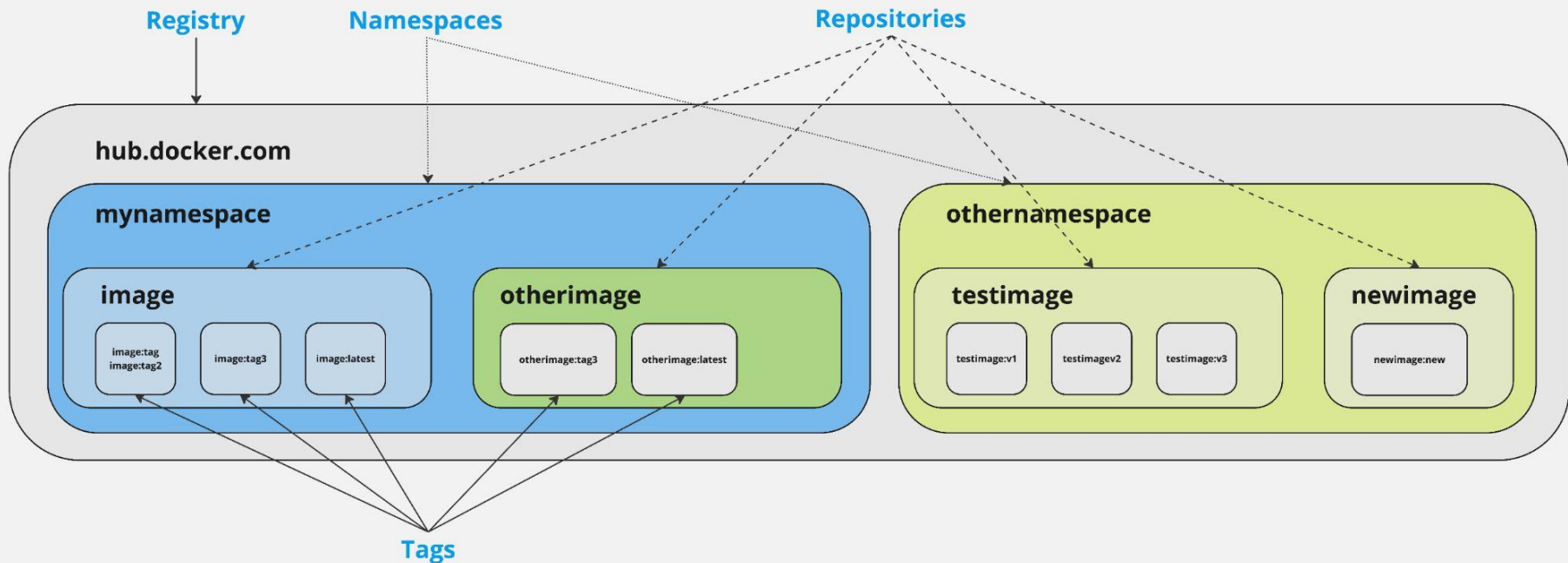


Container Orchestrator

- A system designed to manage containers
 - Controls lifecycle management
 - Provides monitoring / management / security
- Some examples:
 - Kubernetes (EKS, AKS, Microk8s, K3s, KinD, Talos, etc)
 - Hashicorp NOMAD
 - Mirantis Container Runtime
 - Serverless (Fargate, ECS, etc)



Registries, Repos, Repositories, Oh My!



Quick Summary

- Code Repositories
 - Where application code is stored and managed
- Image Registries
 - Where container images are stored after they are built
- Container Images
 - Blueprint of the app, ready to be deployed
 - Stored in Repositories inside Registries
- Containers
 - Running instances of the image, isolated and lightweight
- Orchestrators
 - System to run/manage/monitor/secure multiple containers





Containerization Journey

Your Containerization Journey

- Almost nobody starts containerization with a blank slate
- There is a mix of new applications and existing applications
- The best practice is to understand your companies' willingness to containerize applications, and create standard processes for doing so
- This process must include a triage/prioritization step



Application Modernization Triage

What are the considerations for modernizing applications?

- **Business priorities** – What are the business priorities, and how do they align to modernizing this application?
- **Application knowledge** – does anyone really understand how the application works or how its deployed?
- **Application tech stack** – how do the technologies used in this application align to our current application technology standards?
- **Application lifespan** – is this application still needed in its current form?
- **Organizational capacity** – how much capacity do the development /testing/operations teams have to work on this application?
- **Cost/Risk** – What are the costs and risks of running the current monolith vs the costs and risks of modernizing the application?



Application Modernization Strategies

5 R's first popularized by Gartner

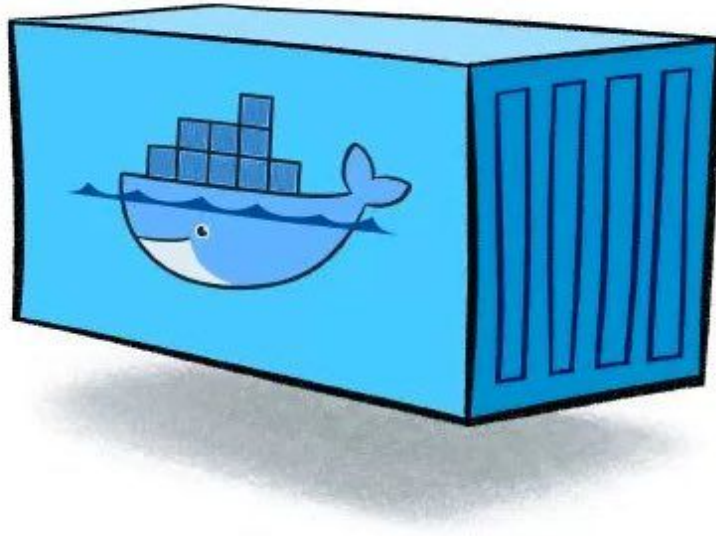
- **Rehost** – Minimal/No changes “Lift and Shift”
- **Refactor** – Light modifications to the application
- **Rearchitect** – Significant modifications/splitting of the application
- **Rebuild** – Rewriting/Redesigning the application
- **Replace** – Retire application and replace with other systems





Migrating to a Container

The Challenge: Containerize a VM or Bare Metal Workload



Current Deployment

Virtual Machine

Wordpress

MySQL

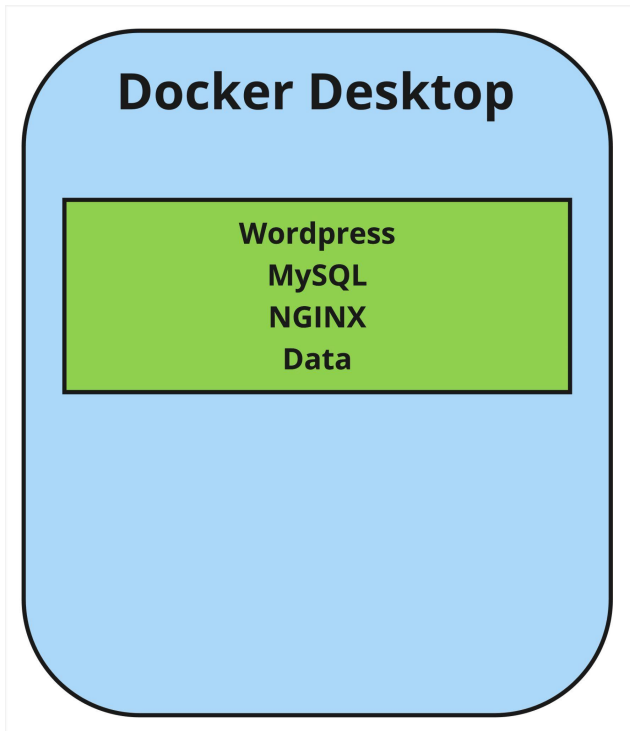
NGINX

All Data

- Ubuntu 20.04 LTS
- WordPress application
- MySQL for data storage
- Data directory for media and uploads
- NGINX for serving the site
- All data stored locally



Simple solution – why not Lift and Shift?



Pros:

- Everything from the VM
- But now in an image!
- Repeatable and shareable

Cons:

- Missing out on a lot of container benefits
- May not be easily scalable

Bottom Line:

- Choose what works best for you...
- ...but understand the tradeoffs

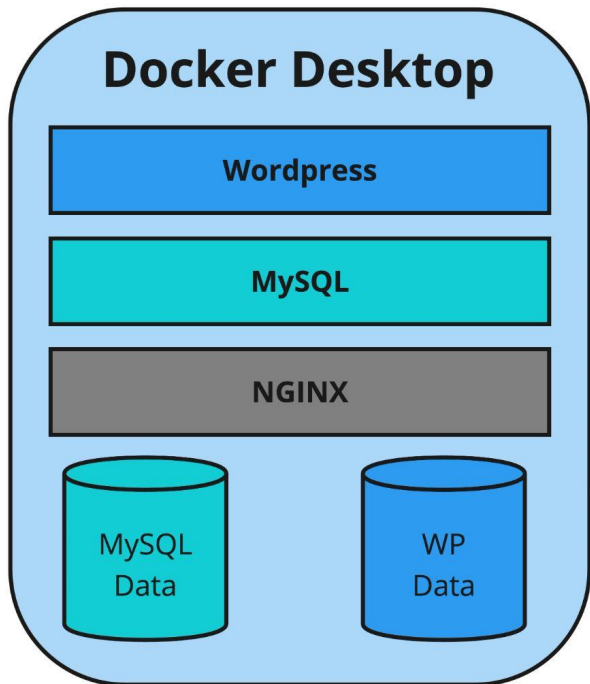


Remember why are we doing this

- Isolation
 - Each component runs in its own environment
- Version Control
 - Easily manage and update software versions
- Scalability
 - Scale components independently based on demand
- Portability
 - Run the application consistently across different environments
 - This can include other OCI compliant runtimes/orchestrators



A Better Container Deployment



- Ubuntu 20.04 LTS Base
- WordPress container
- MySQL container
- NGINX container
- Wordpress Volume
- MySQL Volume



Wordpress



wordpress  Docker Official Image ·  1B+ ·  5.7K

The WordPress rich content management system can utilize plugins, widgets, and themes.

CONTENT MANAGEMENT SYSTEM

















Overview

Tags

Quick reference

- Maintained by:
[the Docker Community](#) 
- Where to get help:
[the Docker Community Slack](#)  [Server Fault](#)  [Unix & Linux](#)  or [Stack Overflow](#) 

Supported tags and respective Dockerfile links

- [6.6.2-php8.1-apache](#)  [6.6-php8.1-apache](#)  [6-php8.1-apache](#)  [php8.1-apache](#)  [6.6.2-php8.1](#)  [6.6-php8.1](#)  [6-php8.1](#)  [php8.1](#) 
- [6.6.2-php8.1-fpm](#)  [6.6-php8.1-fpm](#)  [6-php8.1-fpm](#)  [php8.1-fpm](#) 
- [6.6.2-php8.1-fpm-alpine](#)  [6.6-php8.1-fpm-alpine](#)  [6-php8.1-fpm-alpine](#)  [php8.1-fpm-alpine](#) 



MySQL



mysql

 Docker Official Image ·  1B+ ·  10K+

MySQL is a widely used, open-source relational database management system (RDBMS).

DATABASES & STORAGE


























Overview

Tags

Quick reference

- Maintained by:
[the Docker Community and the MySQL Team](#) 
- Where to get help:
[the Docker Community Slack](#) , [Server Fault](#) , [Unix & Linux](#) , or [Stack Overflow](#) 

Supported tags and respective Dockerfile links

- [9.0.1](#)  [9.0](#)  [9](#)  [innovation](#)  [latest](#)  [9.0.1-oraclelinux9](#)  [9.0-oraclelinux9](#)  [9-oraclelinux9](#)  [innovation-oraclelinux9](#)  [oraclelinux9](#)  [9.0.1-oracle](#)  [9.0-oracle](#)  [9-oracle](#)  [innovation-oracle](#) [oracle](#)
- [8.4.2](#)  [8.4](#)  [8](#)  [lts](#)  [8.4.2-oraclelinux9](#)  [8.4-oraclelinux9](#)  [8-oraclelinux9](#)  [lts-oraclelinux9](#)  [8.4.2-oracle](#)  [8.4-oracle](#)  [8-oracle](#)  [lts-oracle](#) 



NGINX



nginx  Docker Official Image ·  1B+ ·  10K+

Official build of Nginx.

WEB SERVERS





















Overview

Tags

Quick reference

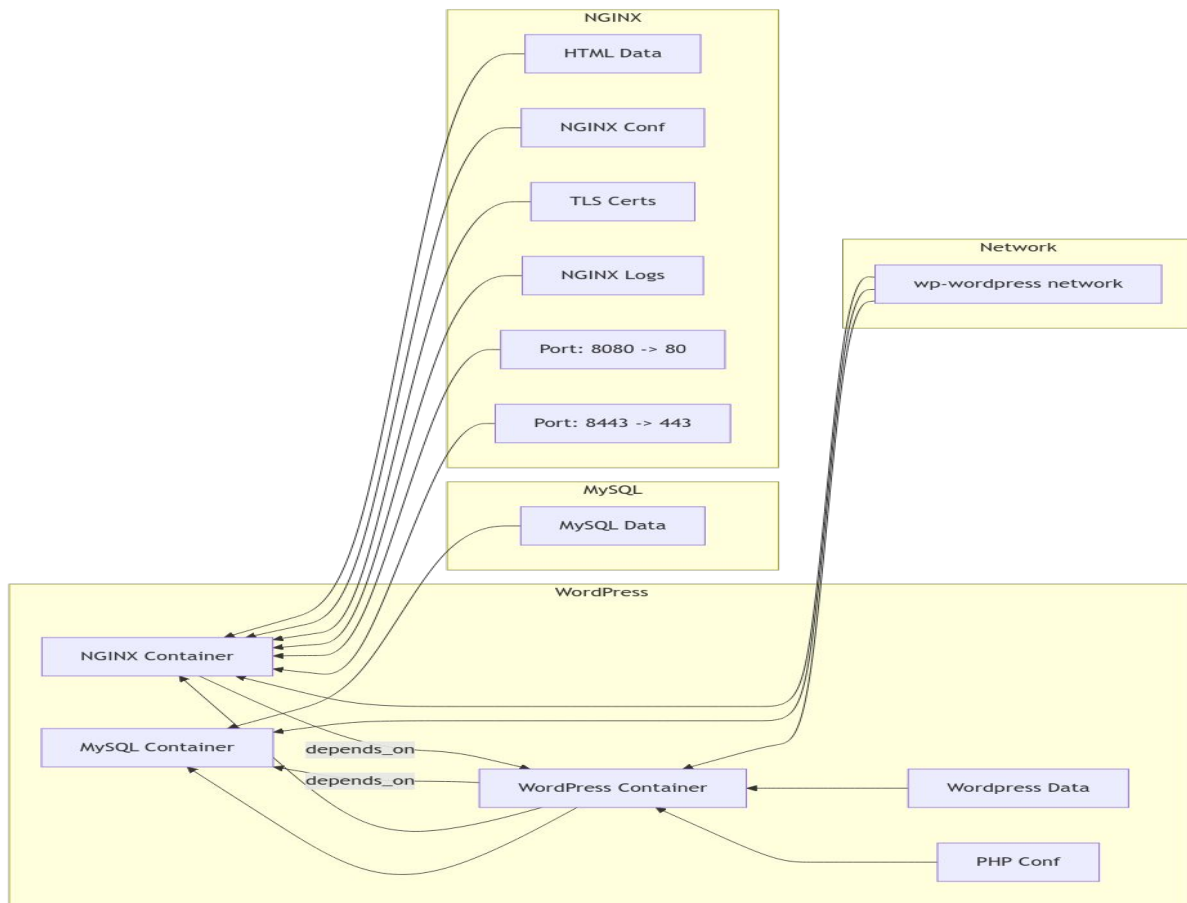
- Maintained by:
[the NGINX Docker Maintainers](#) 
- Where to get help:
[the Docker Community Slack](#) , [Server Fault](#) , [Unix & Linux](#) , or [Stack Overflow](#) 

Supported tags and respective Dockerfile links

- [1.27.2](#)  [mainline](#)  [1](#)  [1.27](#)  [latest](#)  [1.27.2-bookworm](#)  [mainline-bookworm](#)  [1-bookworm](#)  [1.27-bookworm](#)  [bookworm](#) 
- [1.27.2-perl](#)  [mainline-perl](#)  [1-perl](#)  [1.27-perl](#)  [perl](#)  [1.27.2-bookworm-perl](#)  [mainline-bookworm-perl](#)  [1-bookworm-perl](#)  [1.27-bookworm-perl](#)  [bookworm-perl](#) 



What it Looks Like



Regardless of your application's **architecture, containerization** offers a **consistent** and **portable** environment for deployment and scaling.





Questions and Answers