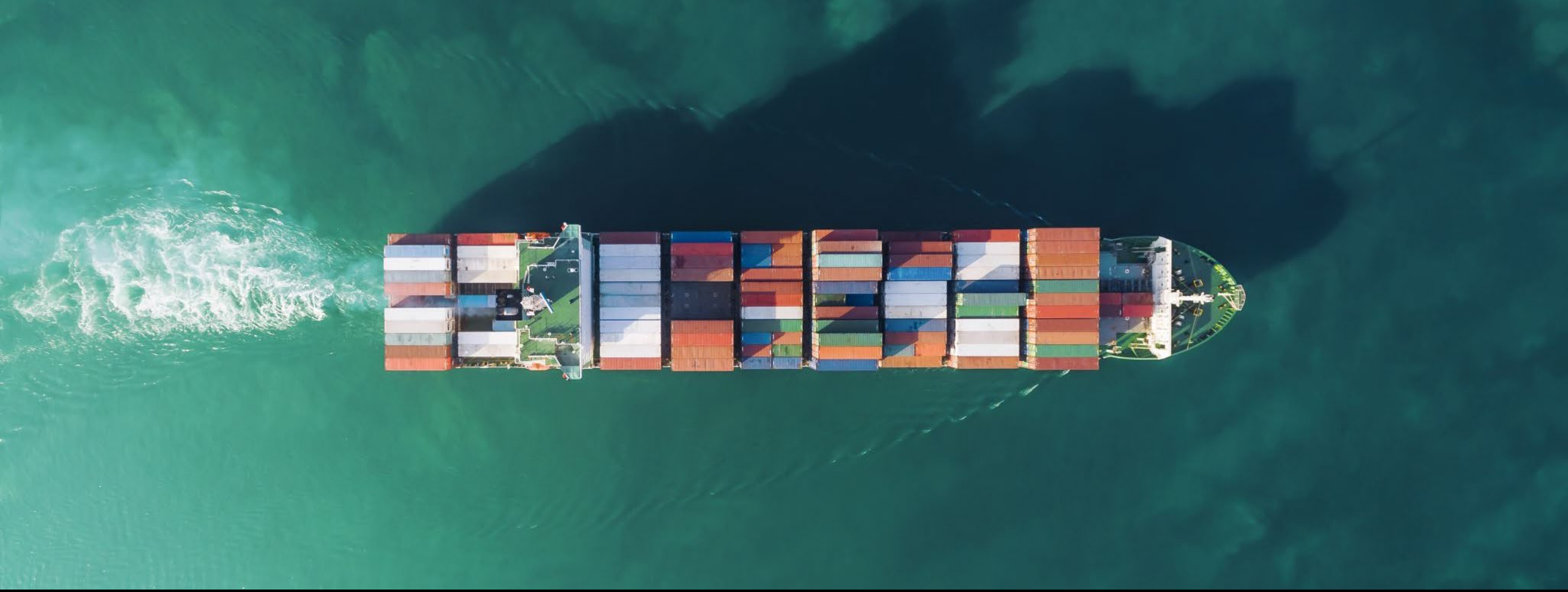


Kubernetes

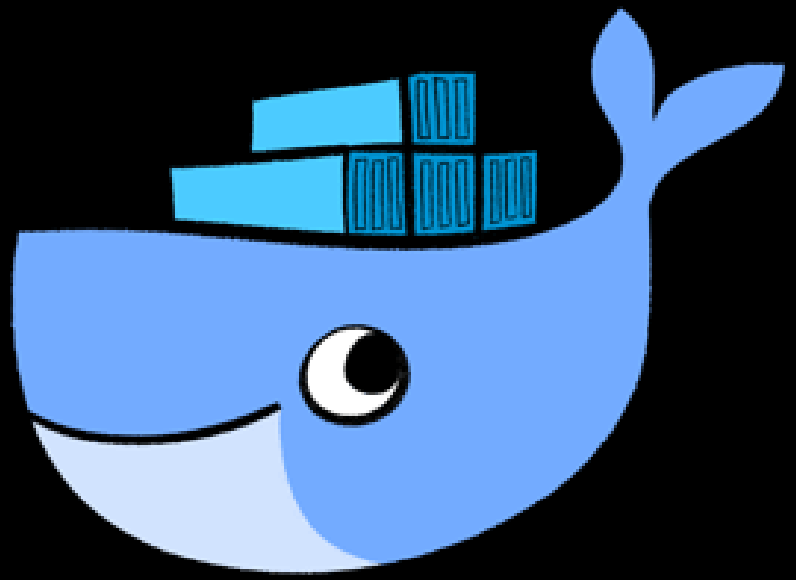
Introduction to K8S

Kubernetes Introduction

“Kubernetes is a powerful open-source system, initially developed by Google, for managing containerized applications in a clustered environment. It aims to provide better ways of managing related, distributed components and services across varied infrastructure.”

Containerized applications

"Containerization is an approach to software development in which an application or service, its dependencies, and its configuration (abstracted as deployment manifest files) are packaged together as a container image. You then can test the containerized application as a unit and deploy it as a container image instance to the host operating system (OS)."

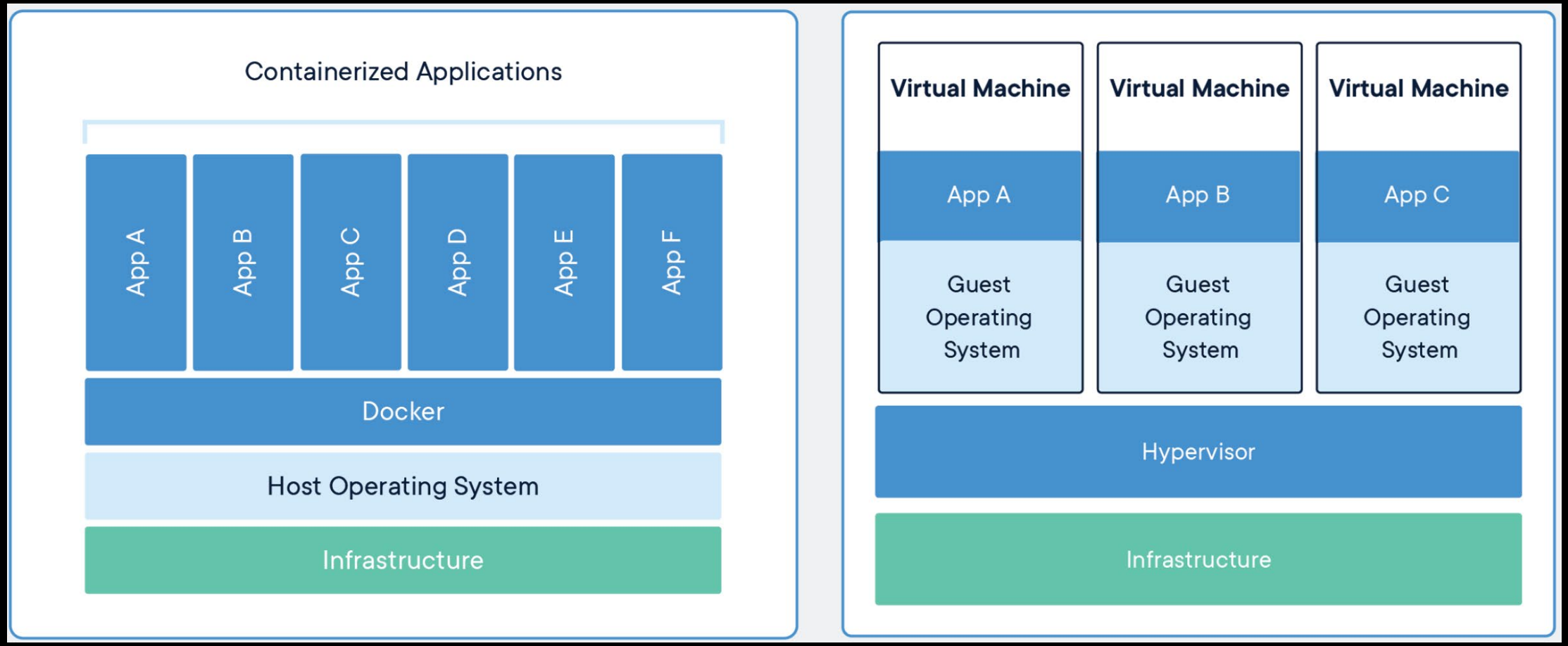


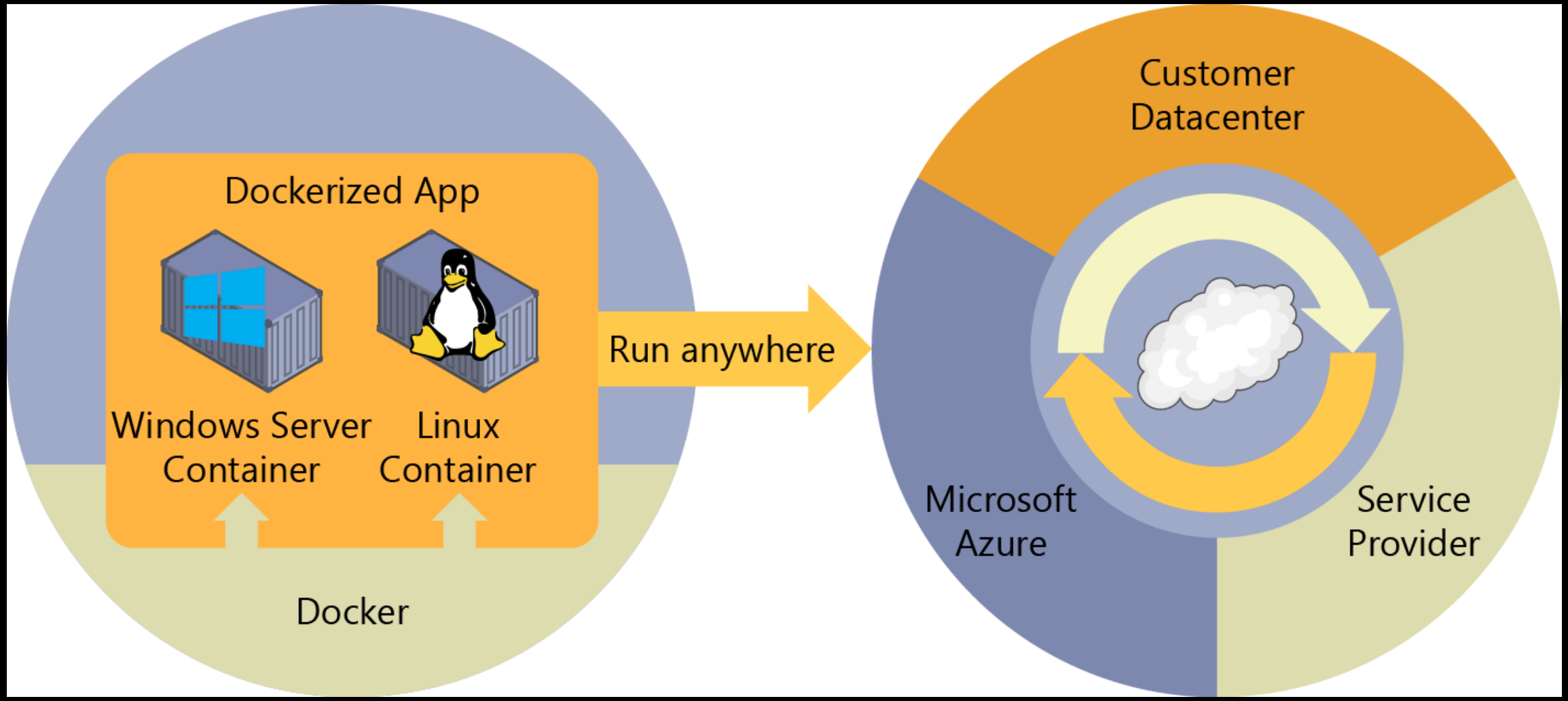
Intro to Docker

”Ship my machine”

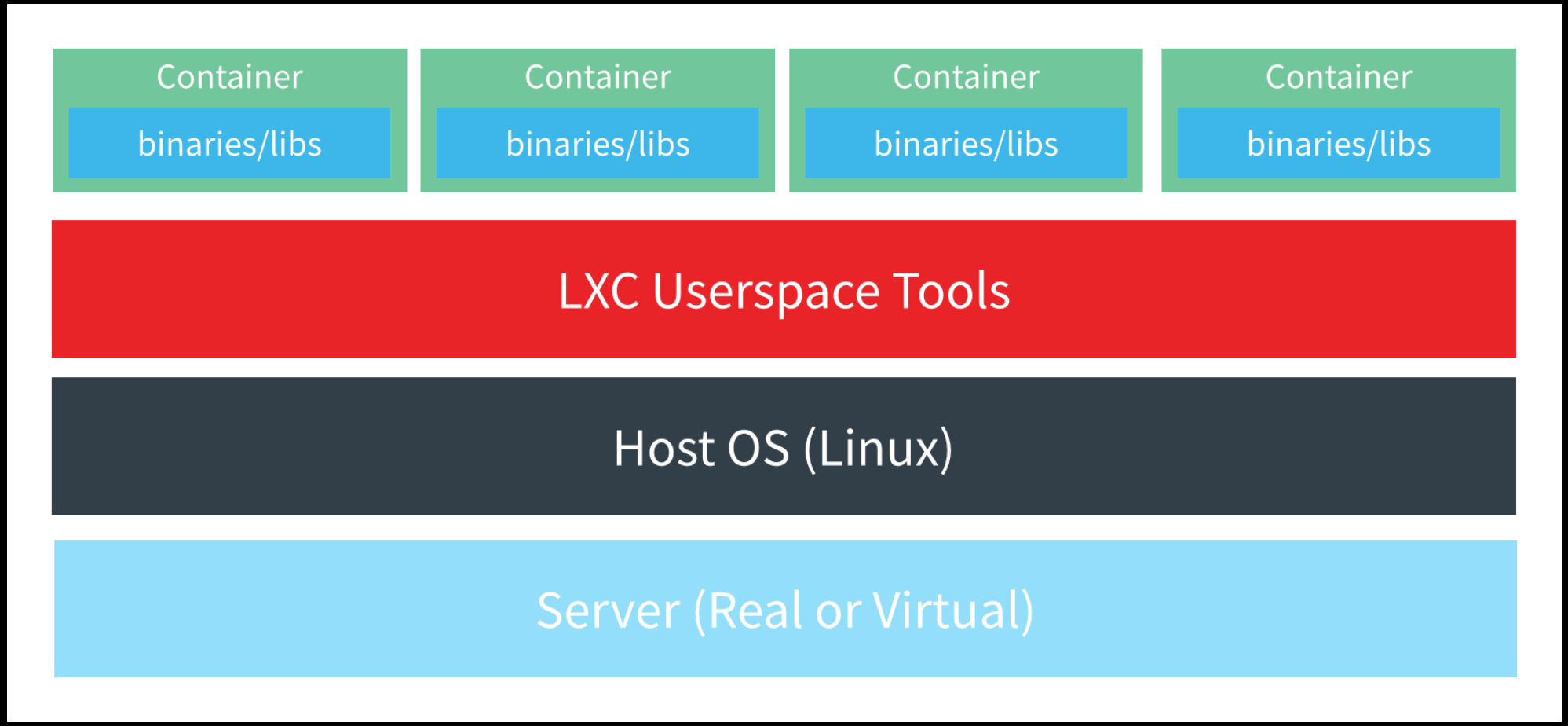
- Since 2014
- Containers have been around since 2000
- Google has lmctfy, running their apps (Gmail, Docs, Search)
- Originally for Linux (build on LXC)

Containers vs VMs

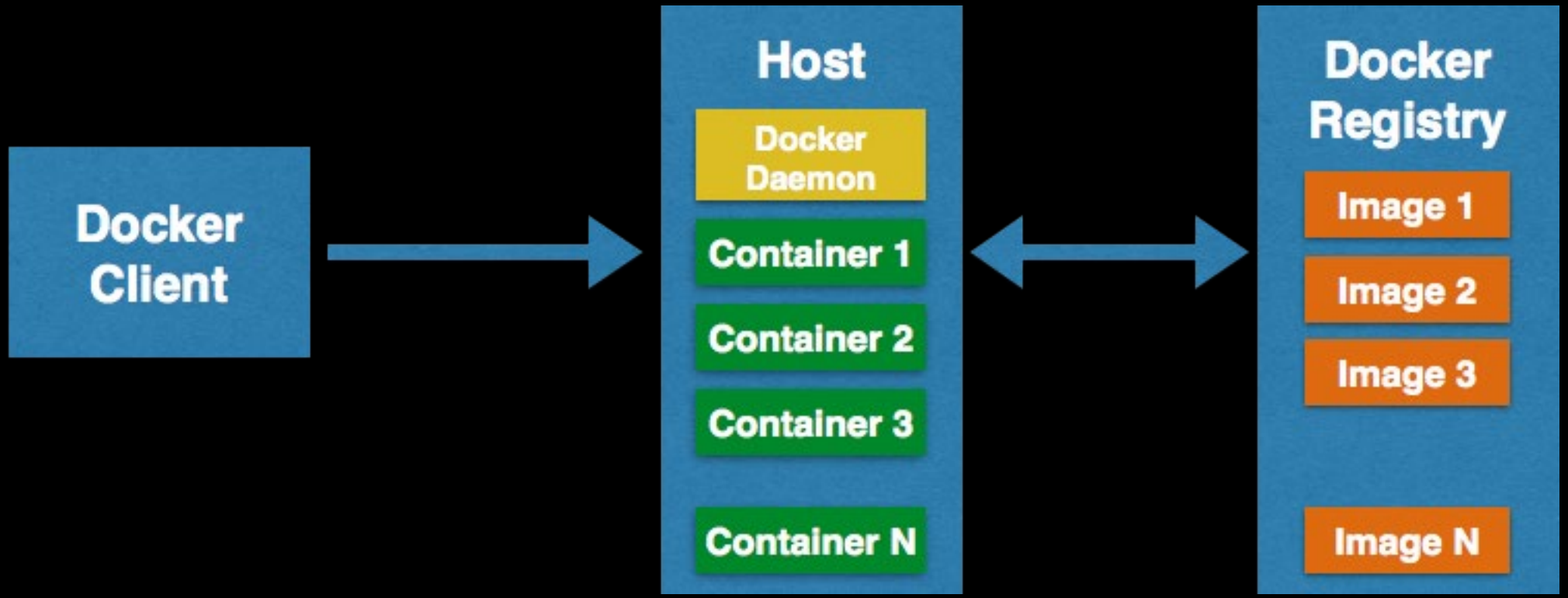



Dockerized App
Windows Server Container
Linux Container
Docker
Run anywhere
Customer Datacenter
Microsoft Azure
Service Provider

How does it work?



The parts


Docker Client
Host
Docker Daemon
Container 1
Container 2
Container 3
Container N
Docker Registry
Image 1
Image 2
Image 3
Image N

Docker terminology

- Container image
- Dockerfile
- Build
- Container
- Tag
- Registry
- Compose
- Cluster
- Orchestrator

Docker Host

- Run as daemon
- Build images
- Download images
- Starting & stopping containers
- Rest API for remote management
- Infrastructure: storage, networking, memory, cpu

Stateful or Stateless

- Docker is designed to be stateless
- No persistent storage of state!

- Workaround: Volumes

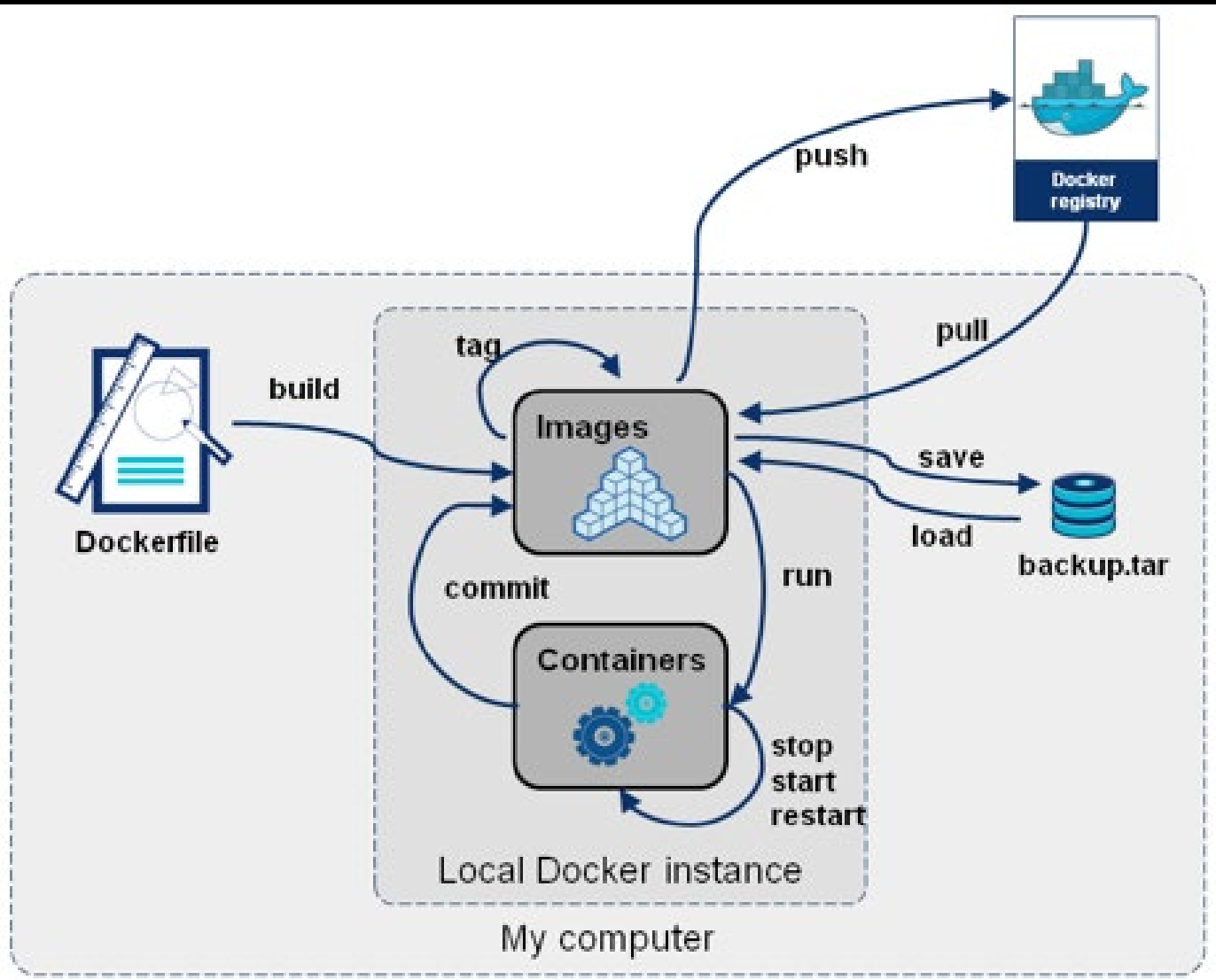
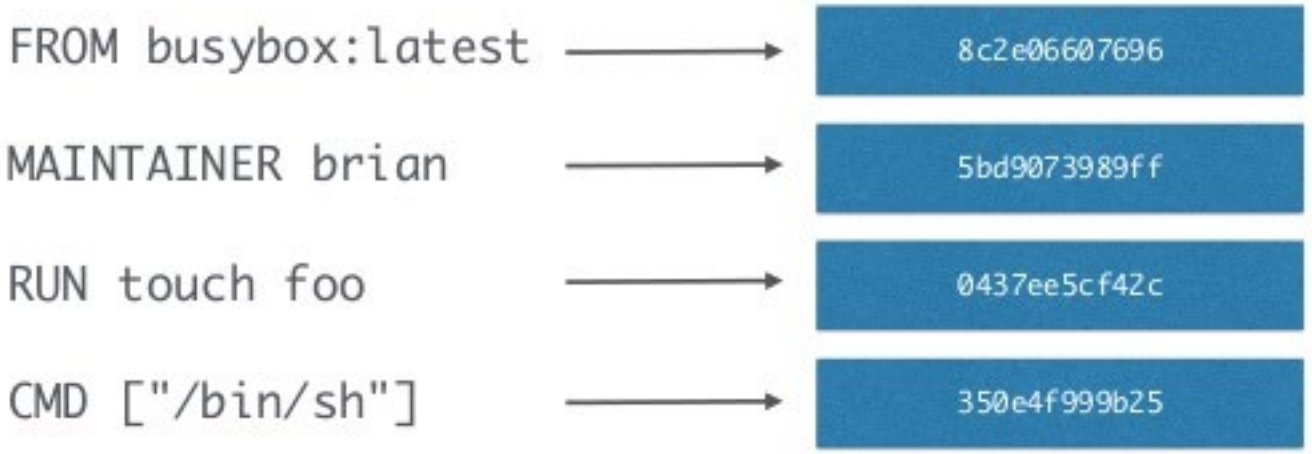
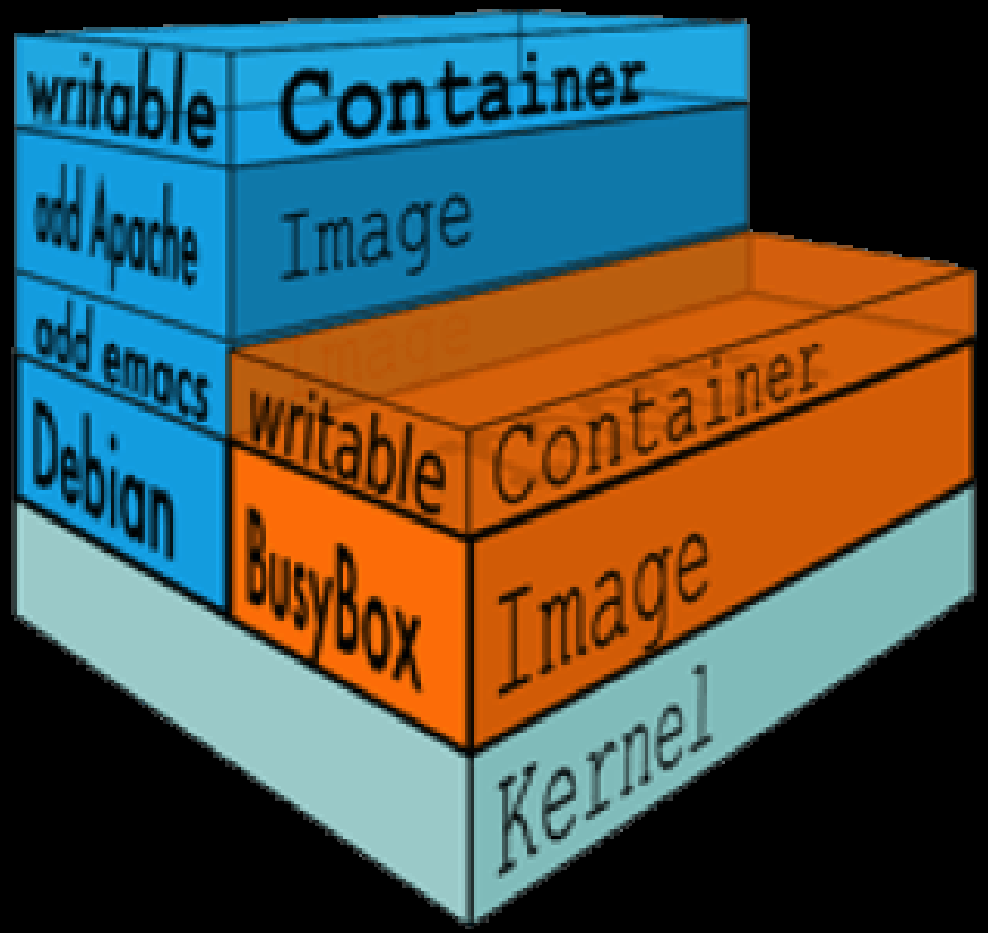

Docker registry
push
pull
tag
build
Images
save
load
backup.tar
Dockerfile
commit
run
Containers
stop
start
restart
Local Docker instance
My computer

Image Layers

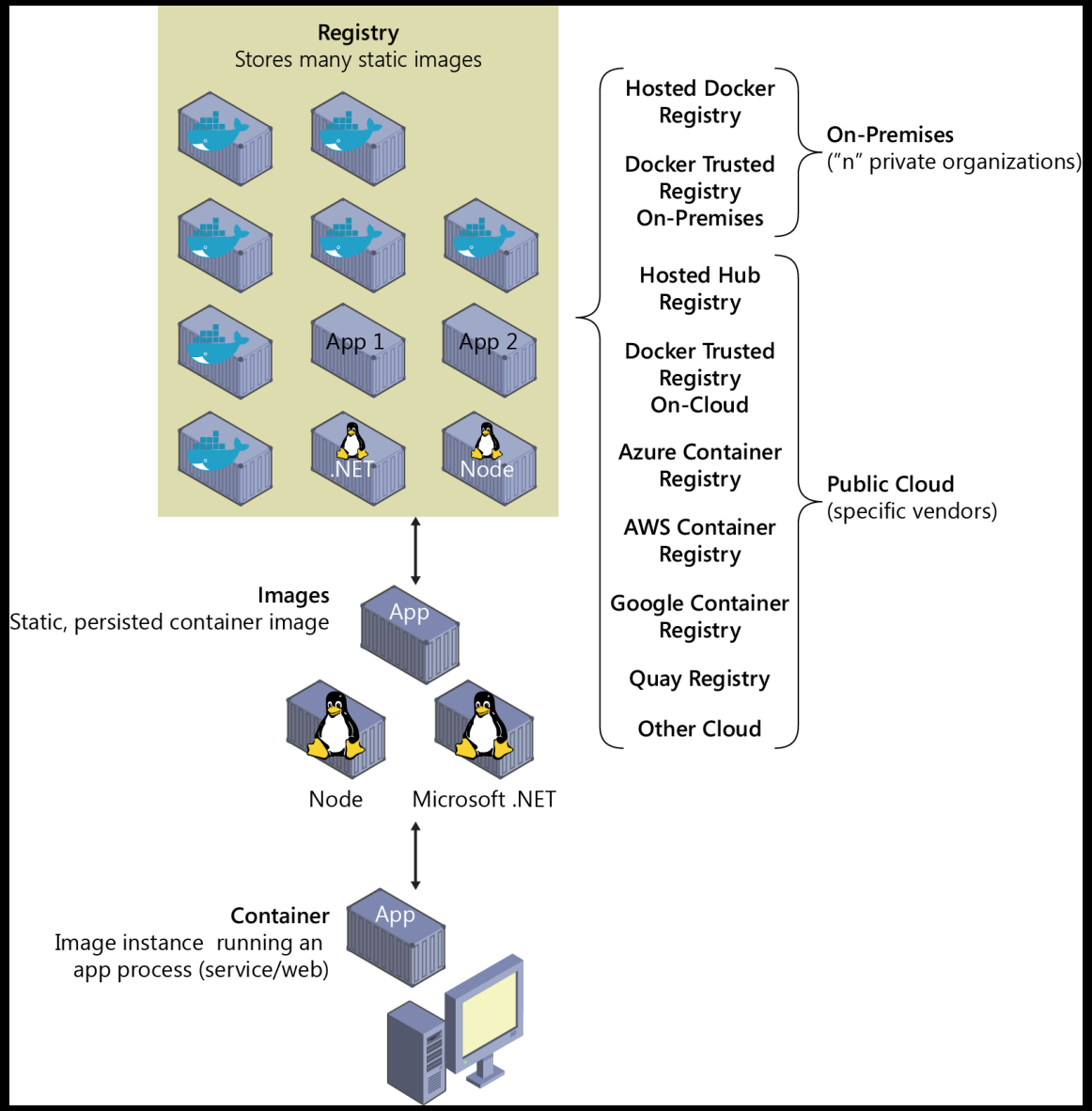
- Each Dockerfile instruction generates a new layer



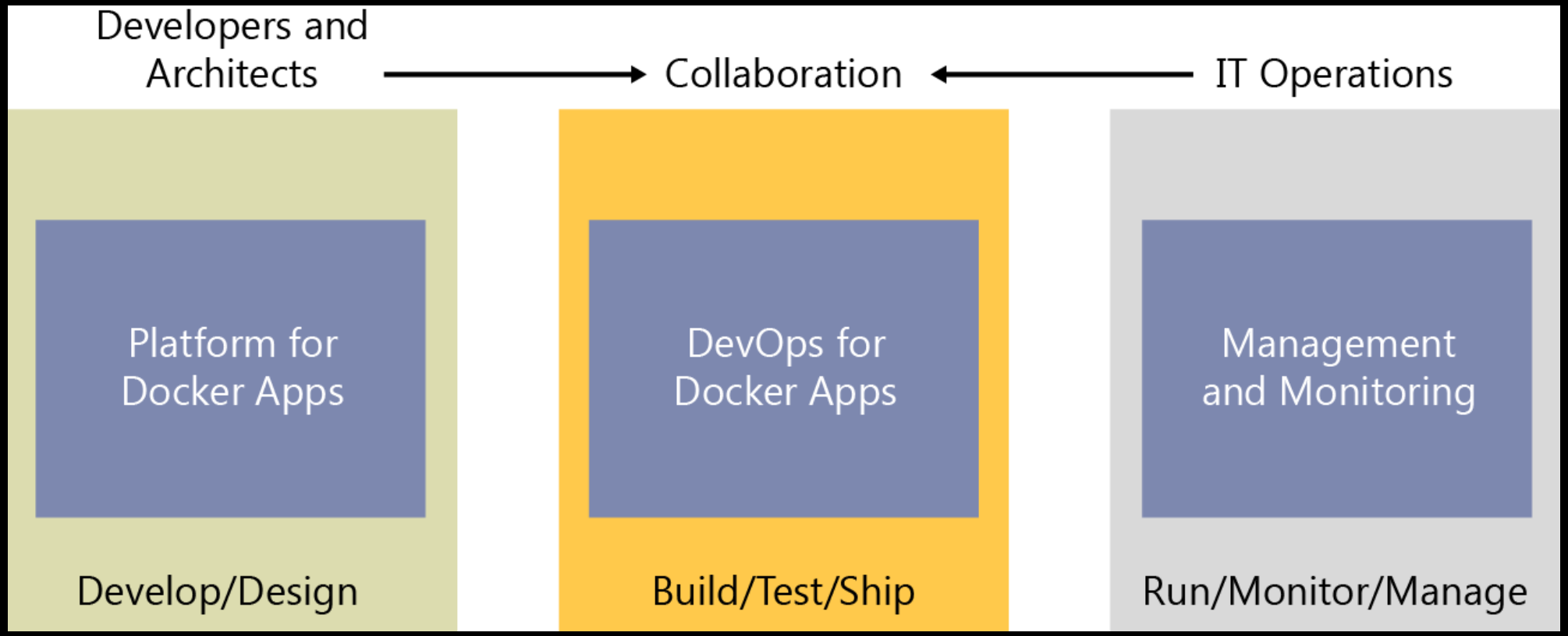

writable
Container
add Apache
Image
add emacs
writable
Container
Debian
BusyBox
Image
Kernel

Getting & Building images

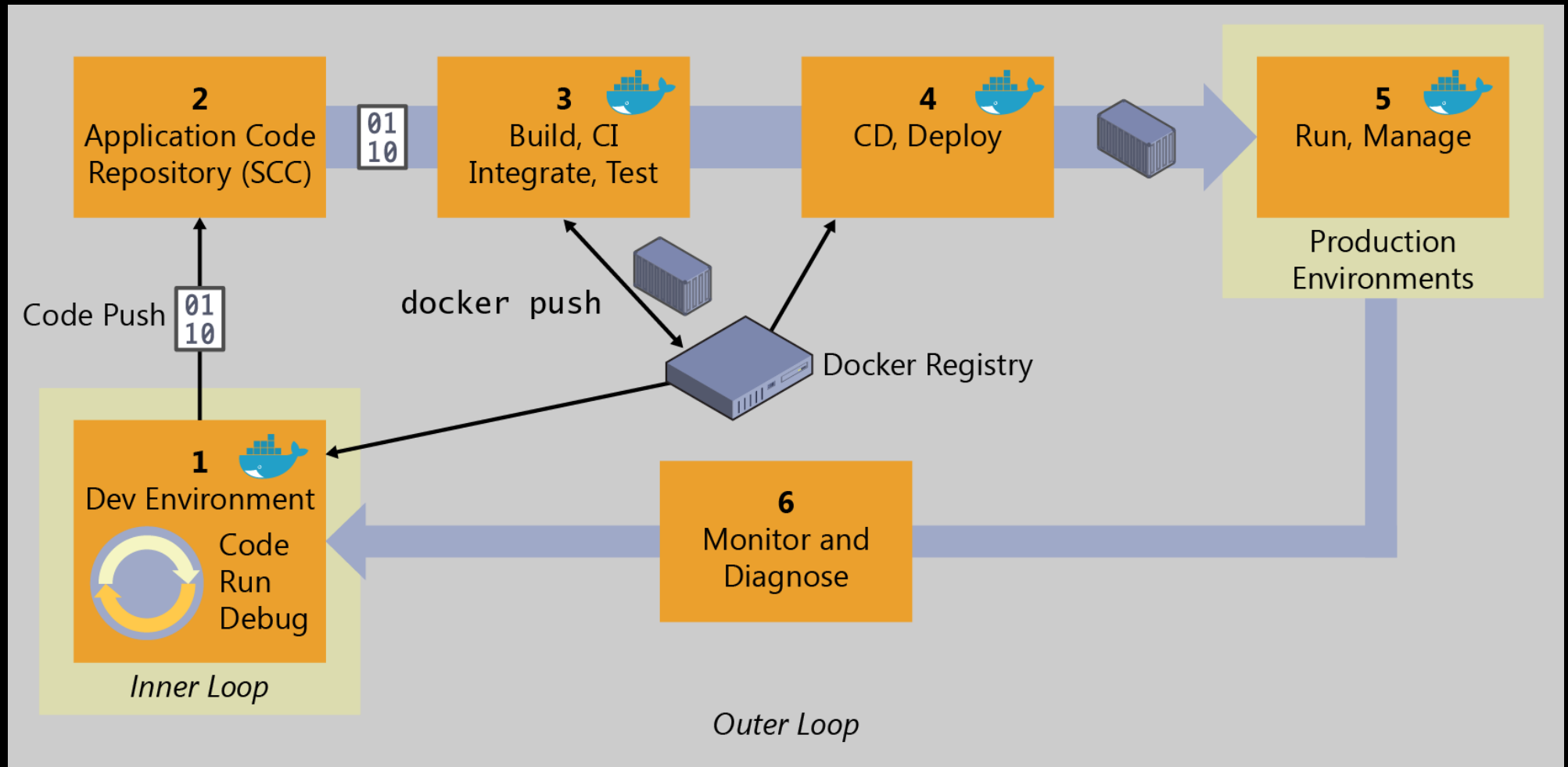
- docker pull <image-name>
- docker run <image-name>

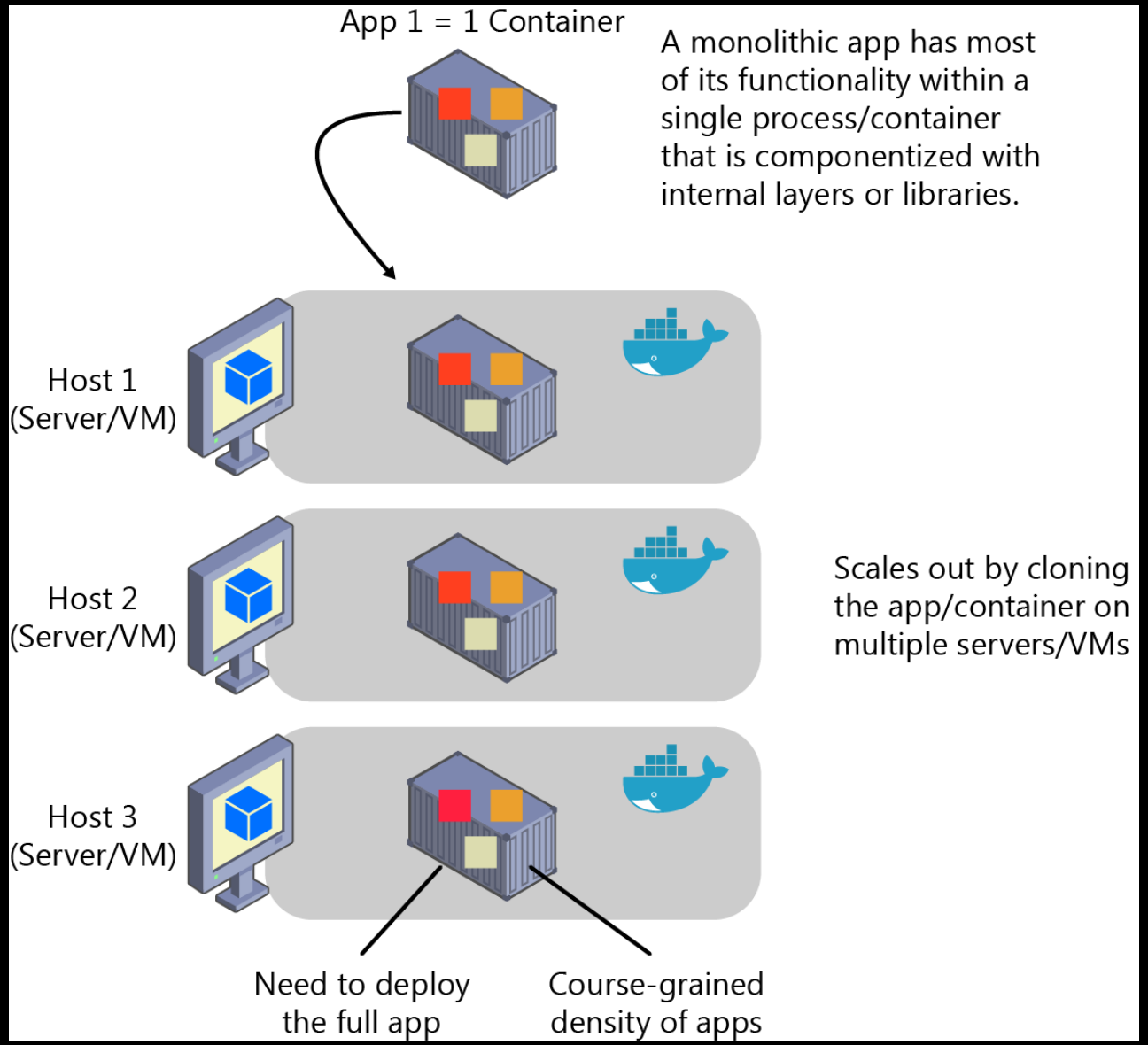

Registry
Stores many static images
App 1
App 2
.NET
Node
Images
Static, persisted container image
App
Node
Microsoft .NET
Container
Image instance running an app process (service/web)
App
Hosted Docker Registry
Docker Trusted Registry On-Premises
On-Premises
("n" private organizations)
Hosted Hub Registry
Docker Trusted Registry On-Cloud
Azure Container Registry
AWS Container Registry
Google Container Registry
Quay Registry
Other Cloud
Public Cloud
(specific vendors)

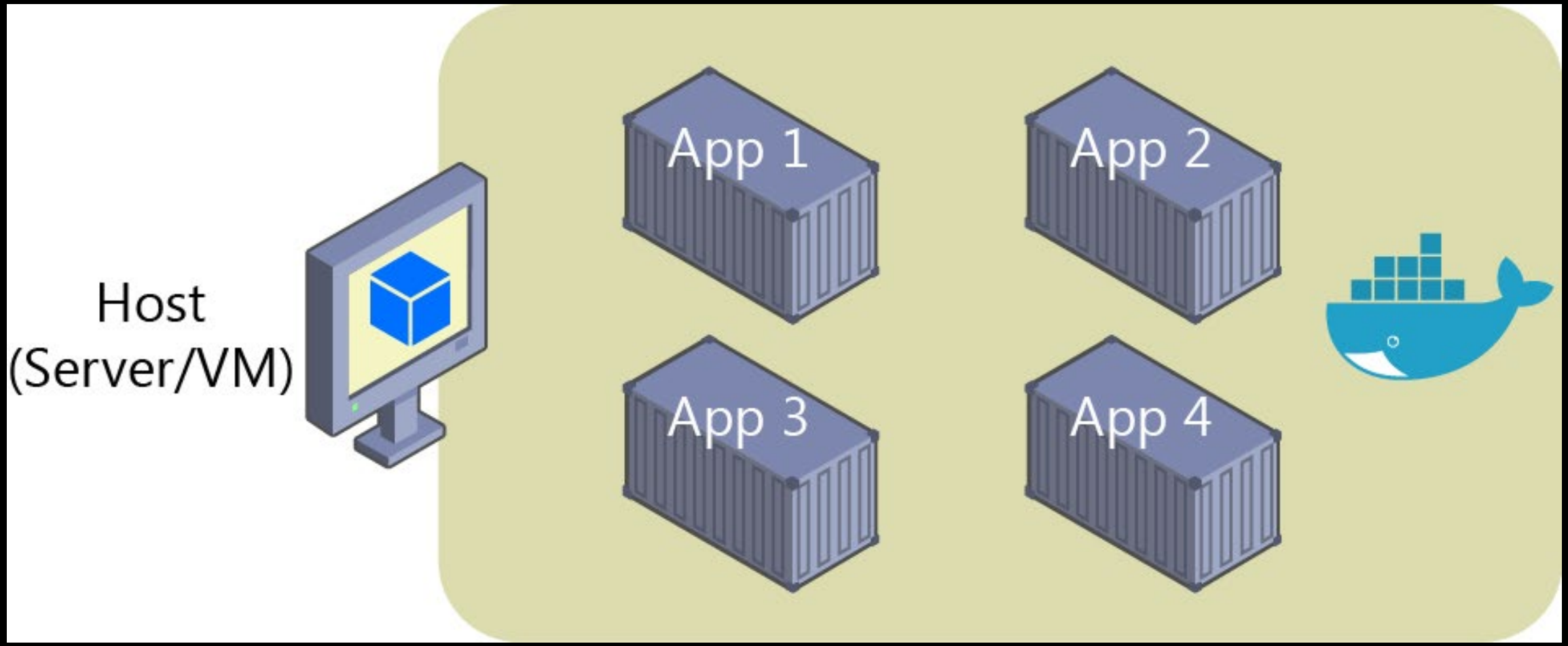
Workflow

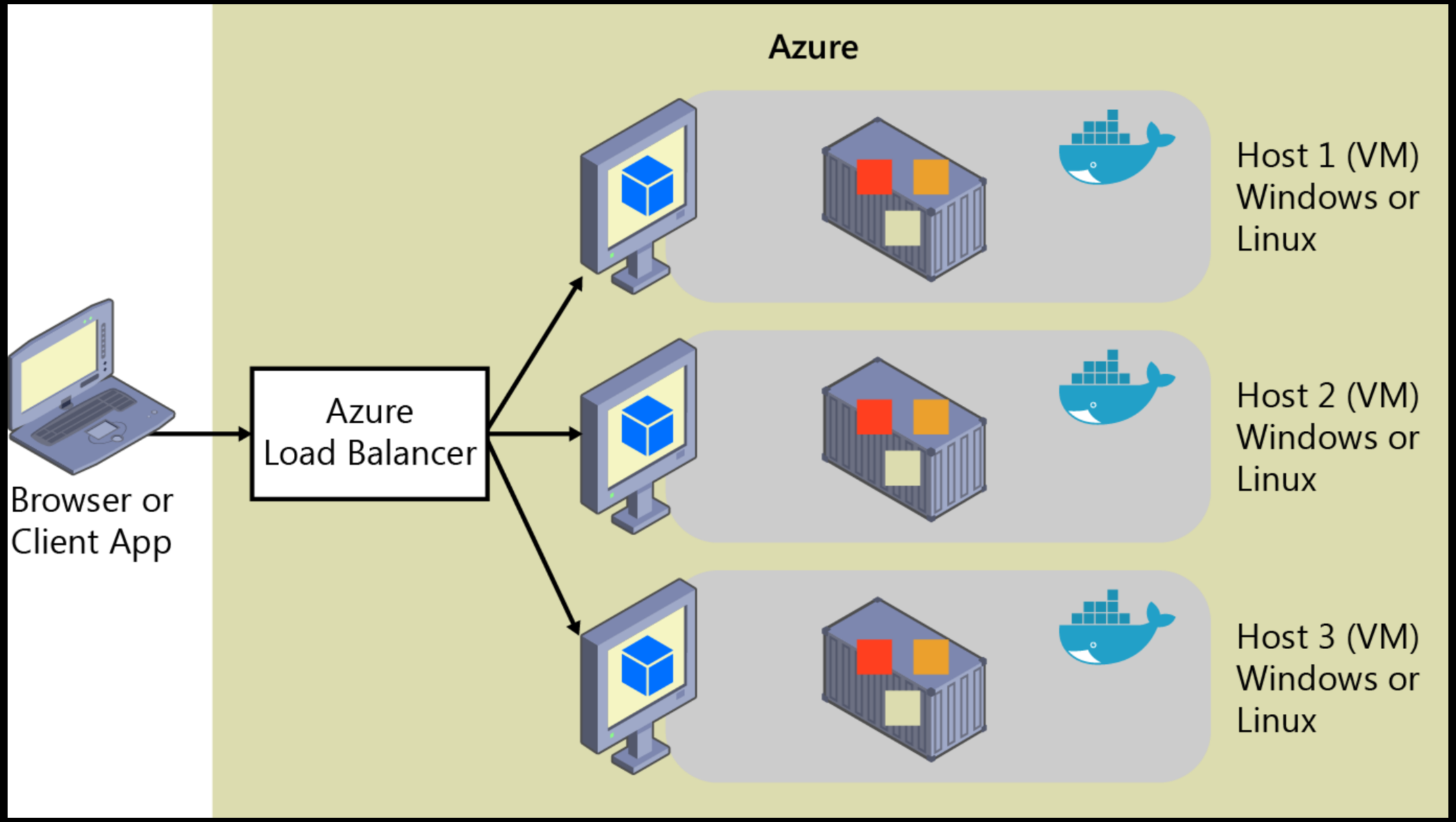

Developers and Architects
Collaboration
IT Operations
Platform for Docker Apps
DevOps for Docker Apps
Management and Monitoring
Develop/Design
Build/Test/Ship
Run/Monitor/Manage

Workflow


2
Application Code
Repository (SCC)
01
10
3
Build, CI
Integrate, Test
4
CD, Deploy
5
Run, Manage
Production
Environments
Code Push
01
10
docker push
Docker Registry
1
Dev Environment
Code
Run
Debug
Inner Loop
6
Monitor and
Diagnose
Outer Loop


App 1 = 1 Container
A monolithic app has most of its functionality within a single process/container that is componentized with internal layers or libraries.
Host 1 (Server/VM)
Host 2 (Server/VM)
Host 3 (Server/VM)
Scales out by cloning the app/container on multiple servers/VMs
Need to deploy the full app
Course-grained density of apps


Host
(Server/VM)
App 1
App 2
App 3
App 4


Azure
Browser or
Client App
Azure
Load Balancer
Host 1 (VM)
Windows or
Linux
Host 2 (VM)
Windows or
Linux
Host 3 (VM)
Windows or
Linux

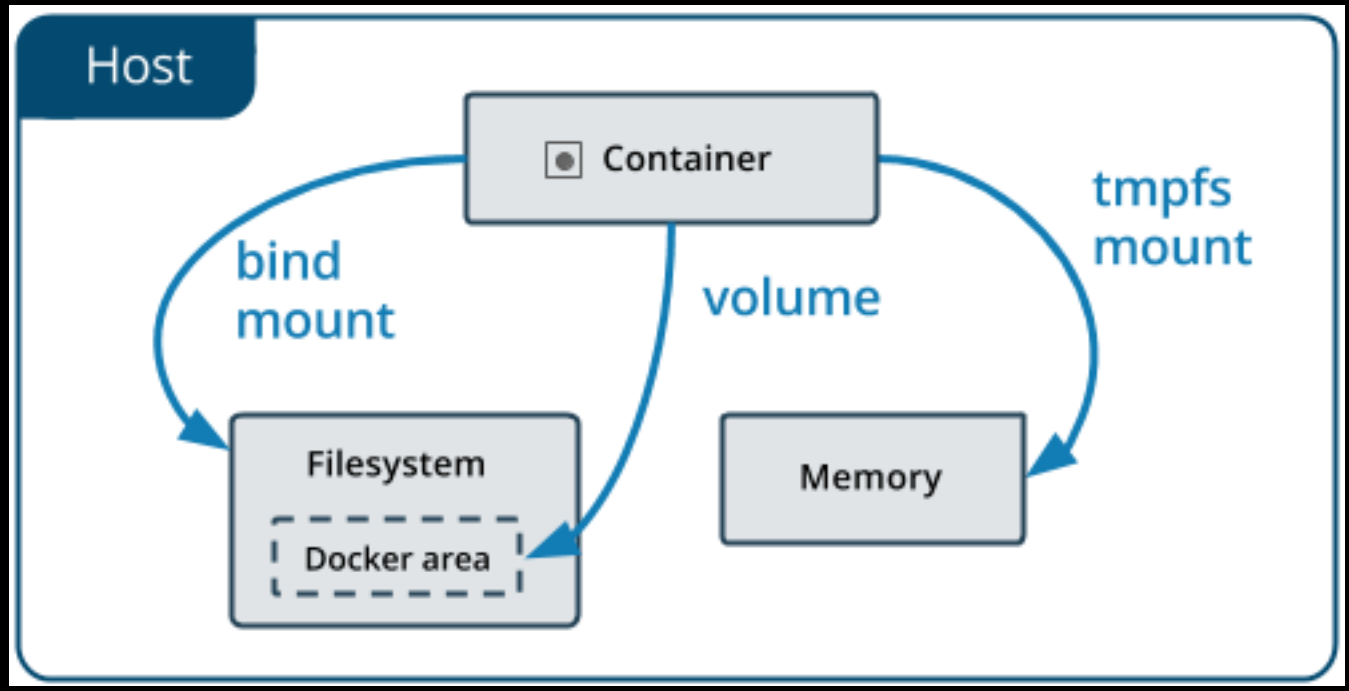
Stateful or Stateless

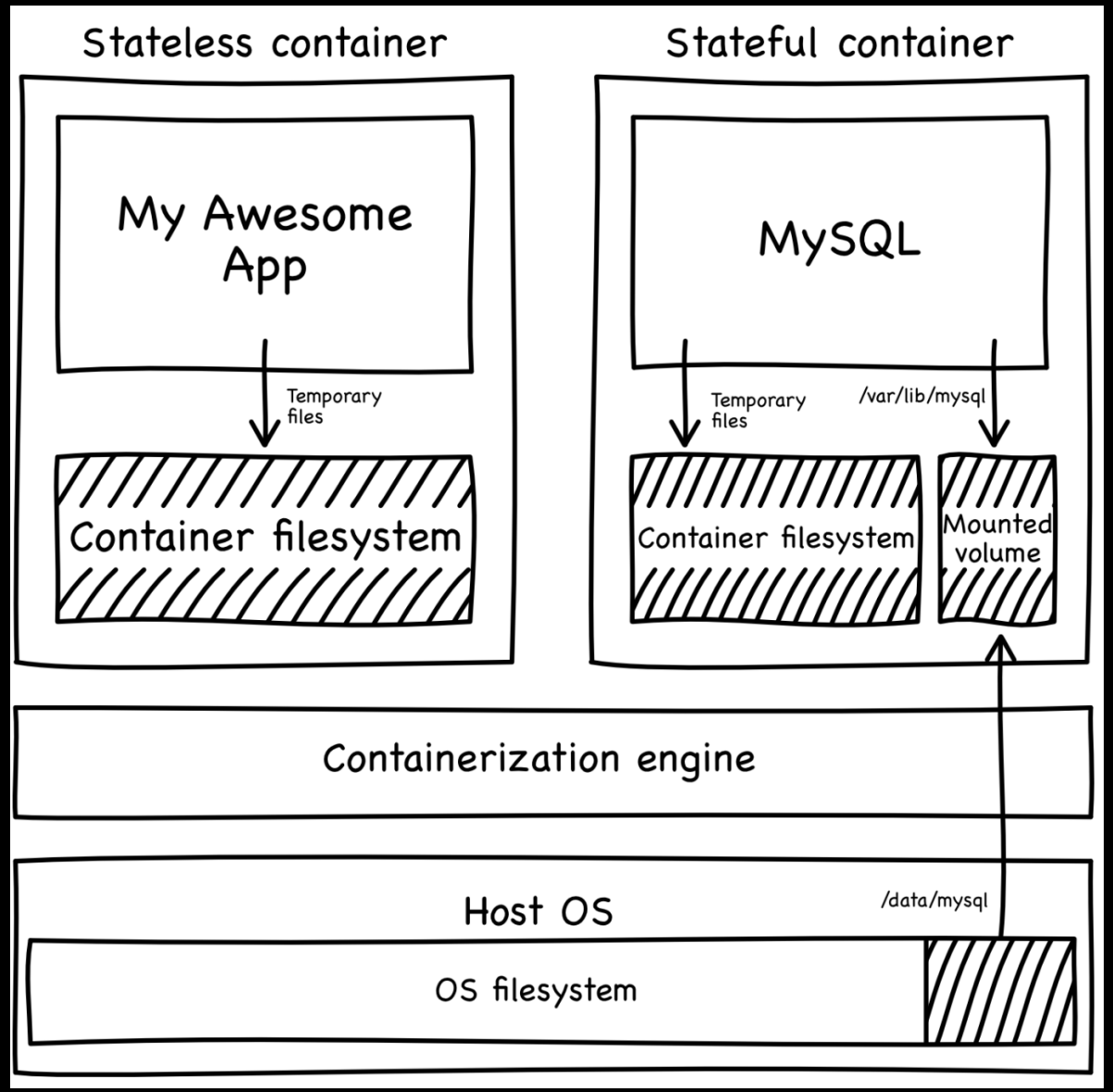
- Docker is designed to be stateless
- No persistent storage of state!

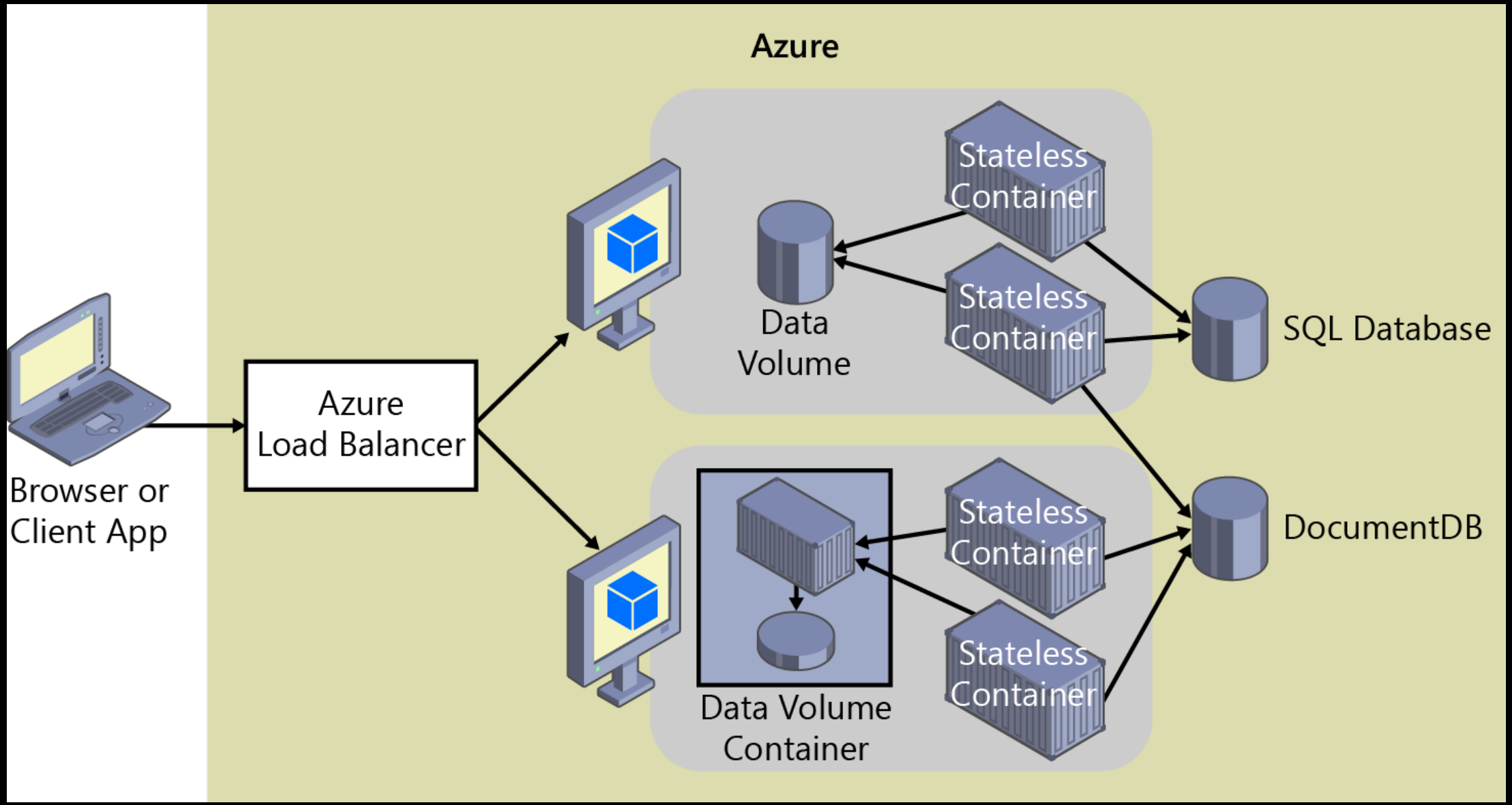
- Workaround: Volumes

Storage

- Volume
  - Part of host file system
  - Managed by Docker
- Bind mount
  - Anywhere on host
  - Inc. system files
- Tmpfs




Stateless container
Stateful container
My Awesome App
MySQL
Temporary files
Temporary files
/var/lib/mysql
Container filesystem
Container filesystem
Mounted volume
Containerization engine
Host OS
/data/mysql
OS filesystem


Azure
Stateless Container
Stateless Container
Data Volume
SQL Database
Browser or Client App
Azure Load Balancer
Stateless Container
Stateless Container
DocumentDB
Data Volume Container

Orchestration

Orchestration

- Automation of all aspects of coordinating and managing containers

- Focused on managing the life cycle of containers and their dynamic environments

Orchestration - Why?

- Automate the following tasks at scale:
  - Configuring and scheduling of containers
  - Provisioning and deployments of containers
  - Availability of containers
  - The configuration of applications in terms of the containers that they run in
  - Scaling of containers to equally balance application workloads across infrastructure
  - Allocation of resources between containers
  - Load balancing, traffic routing and service discovery of containers
  - Health monitoring of containers
  - Securing the interactions between containers.

Orchestration choices

- **Docker Swarm**
  Build into Docker, simple/fast orchestration

- **Mesos**
  Very powerful. Containerized apps and vms side-by-side
  Paypal, Twitter & Uber use it.

- **Kubernetes**
  Extremely portable. Cloud provider integrations.
  Originally by Google (Borg project), CNCF main project, backed by IBM,Amazon,Microsoft,Intel,Cisco, RedHat

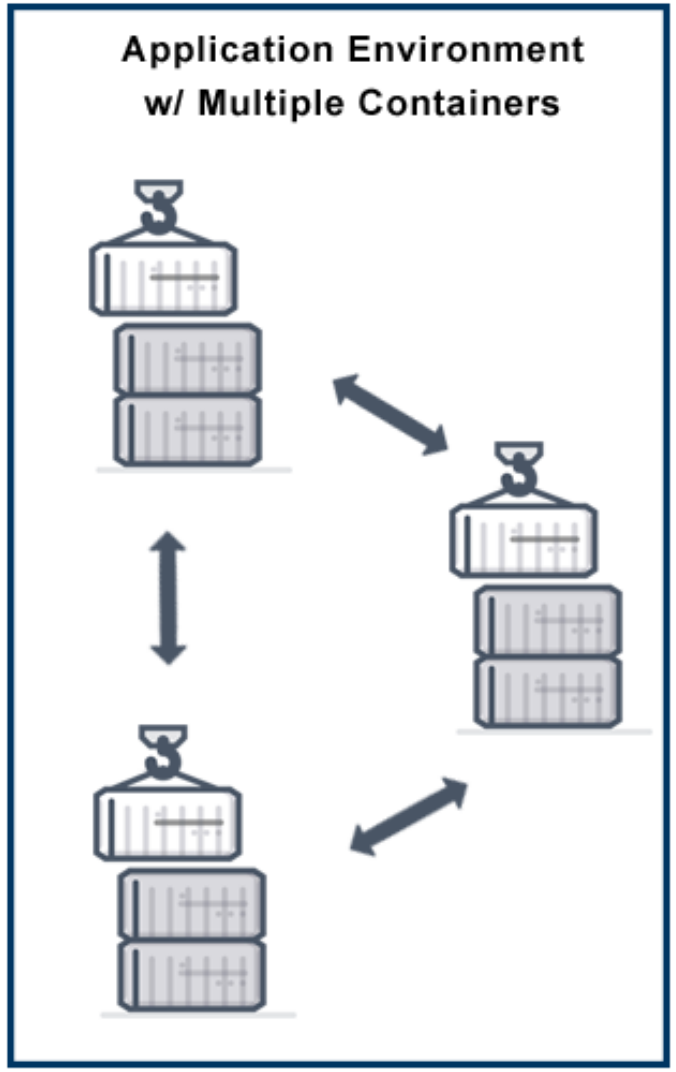
Container Orchestration Software
(Docker, Openshift & Kubernetes)

docker

OPENSHIFT

Automate:

- Configuration
- Provisioning
- Availability
- Scaling
- Security
- Resource allocation
- Load balancing
- Health monitoring



- Docker
- Docker Compose
- Docker Swarm / Stack
- Docker Machine

docker-compose.yml

```yaml
version: "3"
services:
  web:
    # replace username/repo:tag with your name and image details
    image: username/repo:tag
    deploy:
      replicas: 5
      resources:
        limits:
          cpus: "0.1"
          memory: 50M
      restart_policy:
        condition: on-failure
    ports:
      - "80:80"
    networks:
      - webnet
networks:
  webnet:
```

Docker Stack

- Set of interrelated services
- Orchestrated & scaled together

SWARM

- 1.12.0 and up
- Create a cluster (swarm) from Docker CLI
- Different roles (set on runtime)
  - Managers
  - Workers
- Scaling
- Failover
- Multi-host network (vnet over machines)
- Service discovery
- TLS auth/sec

Docker Swarm - Example components

```
docker swarm init
docker stack deploy -c docker-compose.yml <name>
docker service ls
docker service inspect <name>
docker service scale <name>=replica_count
docker network ls
docker network inspect <name>

docker exec –t <id> /bin/bash
apt-get update && apt-get install iputils-ping
cat /etc/resolv.conf
```

Local registry in Swarm

```
docker service create --name registry --publish
5000:5000 registry:2
```

Management Tools

- Consul
- ZooKeeper (SmartStack=+HAProxy, Nerve, Synapse)
- Etcd
- Serf
- Etc etc etc

Introduction to K8S

Kubernetes

- Manages container-based applications
  - Along with networking and storage requirements
  - Focused on application workloads instead of infrastructure components
- Makes it easier to orchestrate large solutions using a variety of containers
  - Application containers
  - Storage containers
  - Middleware containers
  - Even more…
- Applications are described declaratively
  - Use YAML files to describe application
  - Kubernetes handles management and deployment

Kubernetes terminology

- Nodes
  - Individual VM running containerized applications
- Pools
  - Groups of nodes with identical configurations
- Pods
  - Single instance of an application
  - It’s possible for a pod to contain multiple containers within the same node
- Deployments
  - One or more identical pods managed by Kubernetes
- Manifests
  - YAML file describing a deployment

Pods and services

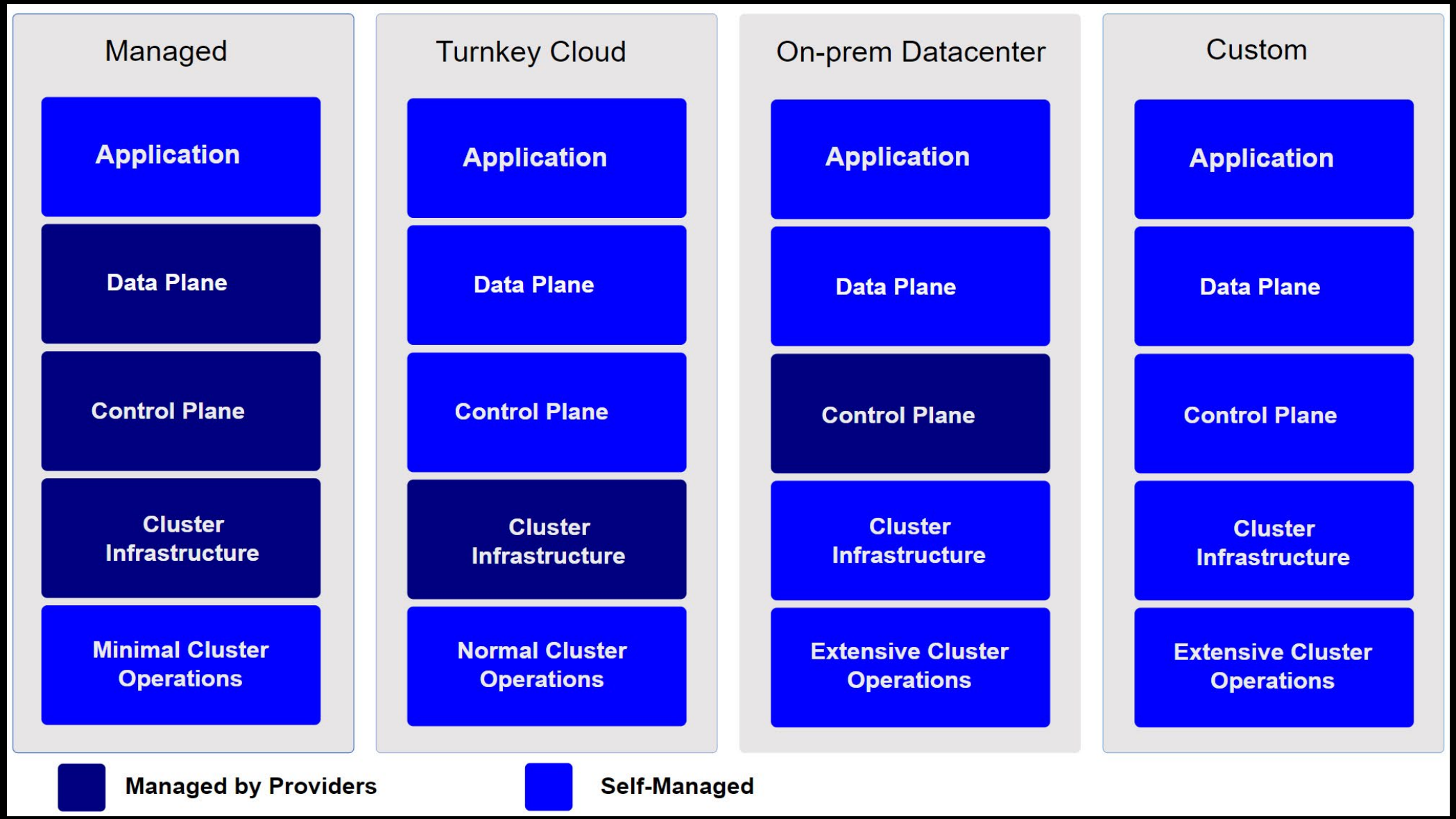
- Kubernetes concept of a pod, which is one or more containers deployed together on one host, and the smallest compute unit that can be defined, deployed, and managed

Managed

- Amazon EKS
- Azure Kubernetes Service (AKS)
- Digital Ocean
- Google Kubernetes Engine (GKE)
- IBM Cloud Kubernetes Service

Turnkey

- Amazon EC2
- Google Compute Engine (GCE)


Managed
Application
Data Plane
Control Plane
Cluster Infrastructure
Minimal Cluster Operations
Turnkey Cloud
Application
Data Plane
Control Plane
Cluster Infrastructure
Normal Cluster Operations
On-prem Datacenter
Application
Data Plane
Control Plane
Cluster Infrastructure
Extensive Cluster Operations
Custom
Application
Data Plane
Control Plane
Cluster Infrastructure
Extensive Cluster Operations
Managed by Providers
Self-Managed