Container

A standard unit of software

that packages up code and all its dependencies

so the application runs quickly and reliably

from one computing environment to another.

- Docker

Container

A standard unit of software

that packages up code and all its dependencies

so the application runs quickly and reliably

from one computing environment to another.

- Docker

```
FROM bowl:1.0.0

RUN apt-get install \
    h2o \
    pebbles \
    plant

COPY fish .

CMD ["fish", "swim"]
```

Container

A standard unit of software

that packages up code and all its dependencies

so the application runs quickly and reliably

from one computing environment to another.

- Docker

```
FROM bowl:1.0.0

RUN apt-get install \
    h2o \
    pebbles \
    plant

COPY fish .

CMD ["fish", "swim"]
```

- Design time: Cohesive full stack of apps and dependencies
- Runtime: Boxed process on an operating system

Container

A standard unit of software

that packages up code and all its dependencies

so the application runs quickly and reliably

from one computing environment to another.

- Docker

```
FROM bowl:1.0.0

RUN apt-get install \
    h2o \
    pebbles \
    plant

COPY fish .

CMD ["fish", "swim"]
```

- Design time: Cohesive full stack of apps and dependencies
- Runtime: Boxed process on an operating system

Why Containers?

Why would you make your life more complicated?

- Build once, run anywhere
- Polyglot
- Disposable
- Modular
- Separation of concerns
- Own your infrastructure
- Rapid scaling

Why Containers?

Why would you make your life more complicated?

Build once, run anywhere
Polyglot
Disposable
Modular
Separation of concerns
Own your infrastructure
Rapid scaling

Replication:
Production abhors singletons

Why Containers?

New reality for service delivery

Why would you make your life more complicated?

Build once, run anywhere
Polyglot
Disposable
Modular
Separation of concerns
Own your infrastructure
Rapid scaling

Replication:
Production abhors singletons

Why Containers?

Why would you make your life more complicated?

Build once, run anywhere
Polyglot
Disposable
Modular
Separation of concerns
Own your infrastructure
Rapid scaling

Replication:
Production abhors singletons

New reality for service delivery

12factor.net

Why Containers?

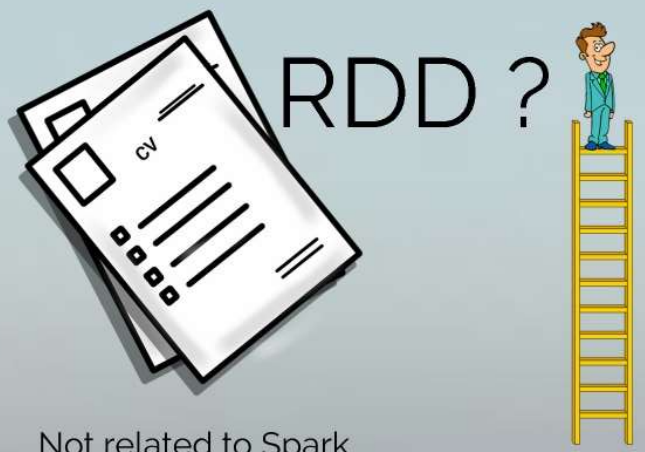
Why would you make your life more complicated?

Build once, run anywhere
Polyglot
Disposable
Modular
Separation of concerns
Own your infrastructure
Rapid scaling

Replication:
Production abhors singletons

New reality for service delivery

12factor.net



Not related to Spark
Resilient Distributed Datasets (RDDs)

Layers

Infrastructure as code

Layers

Code
Functions
Logic

Infrastructure as code

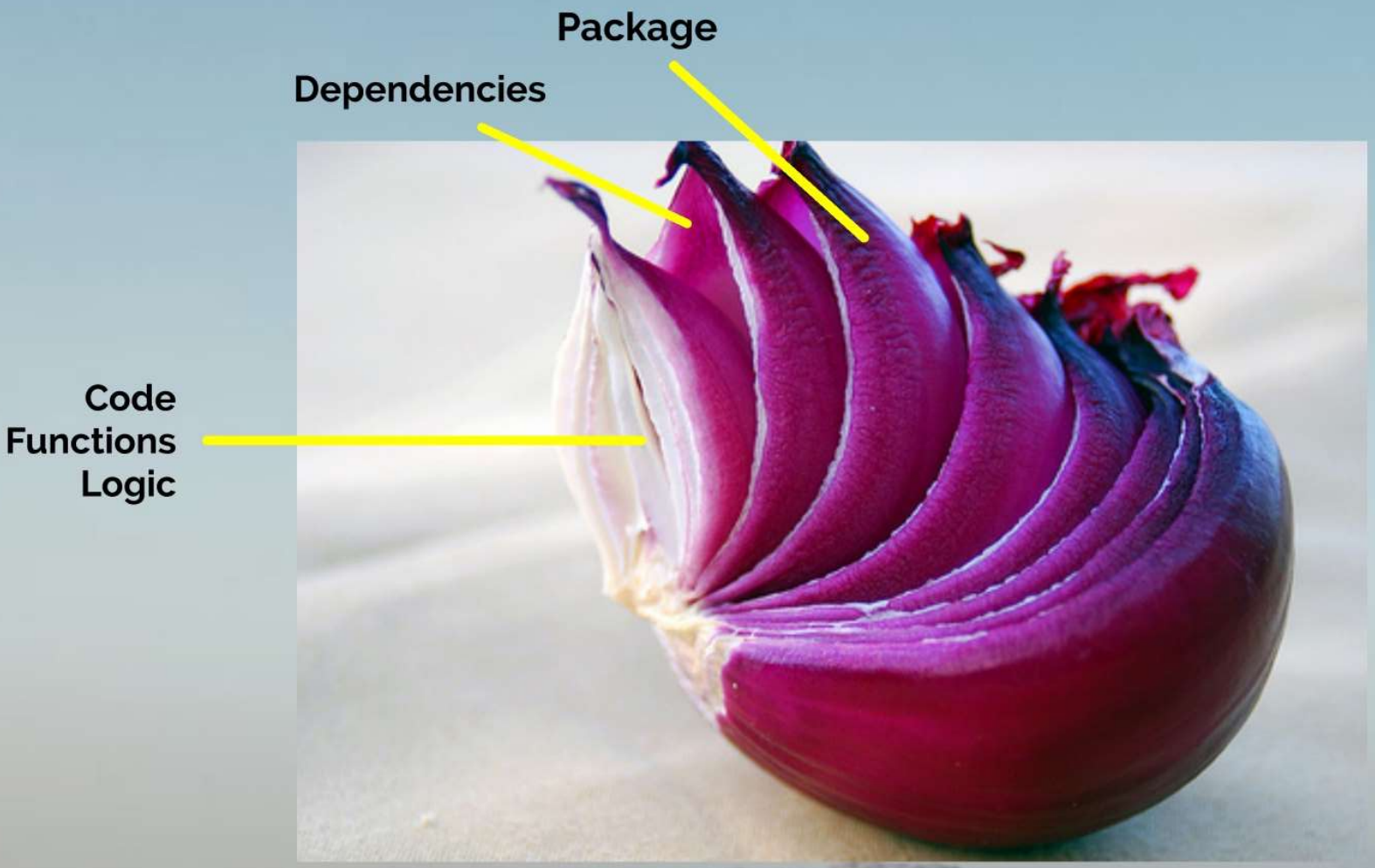
Layers

Dependencies

Code
Functions
Logic

Infrastructure as code

Layers



Infrastructure as code

Layers

API / Web Service
- REST
- gRPC
- GraphQL

Package

Dependencies

Code
Functions
Logic

Infrastructure as code

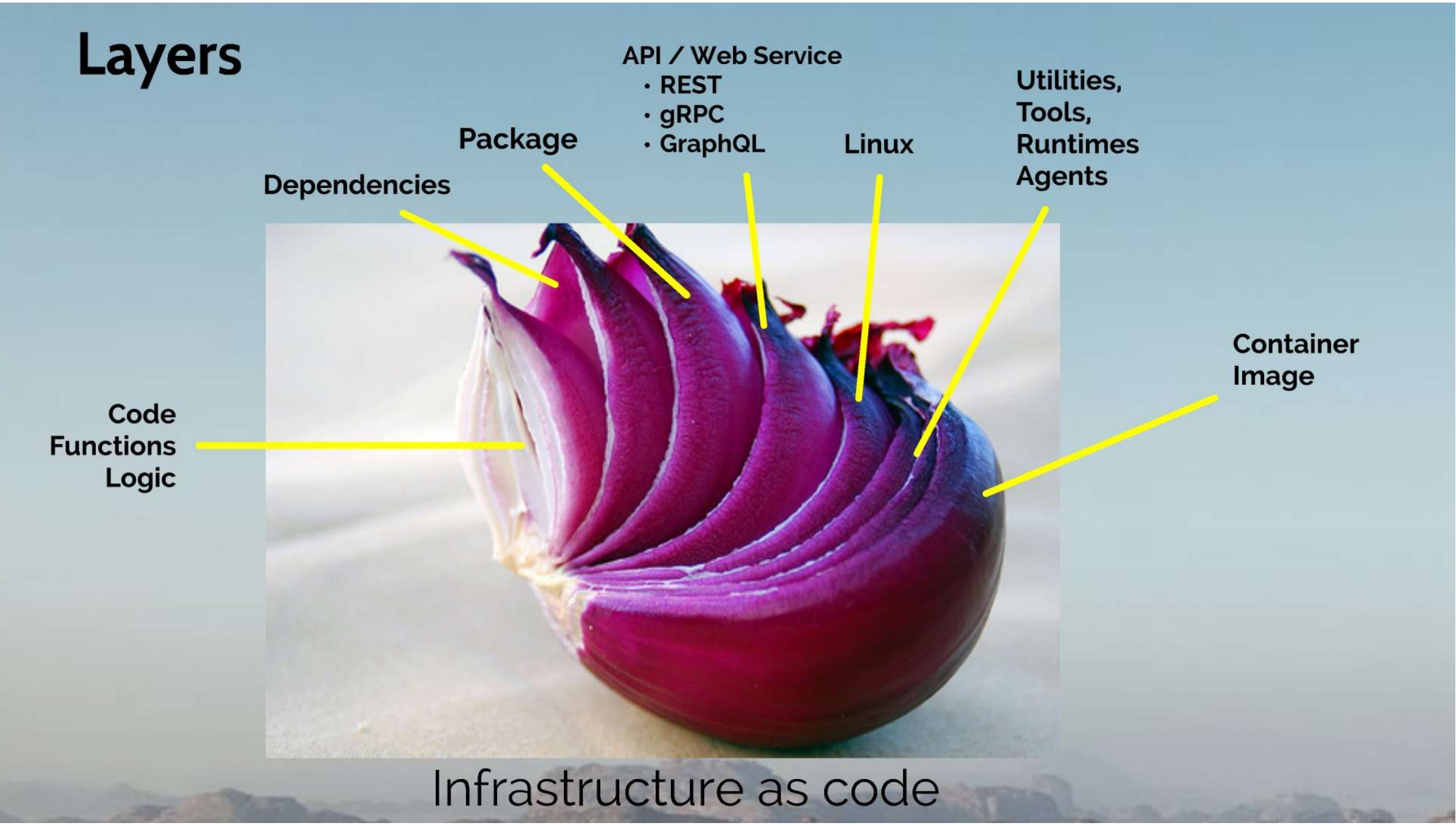
Layers

Infrastructure as code

Layers

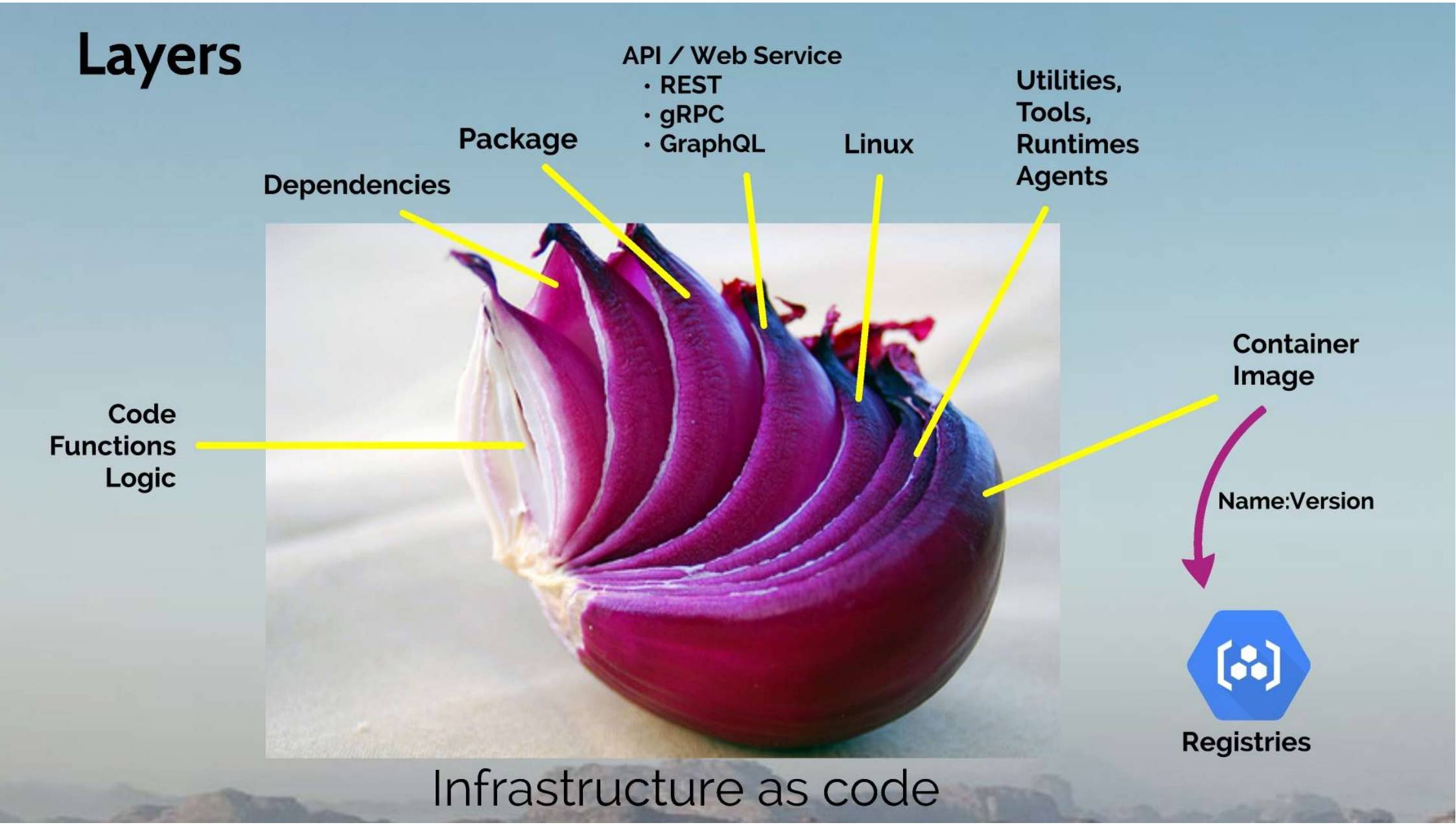
Infrastructure as code

Layers



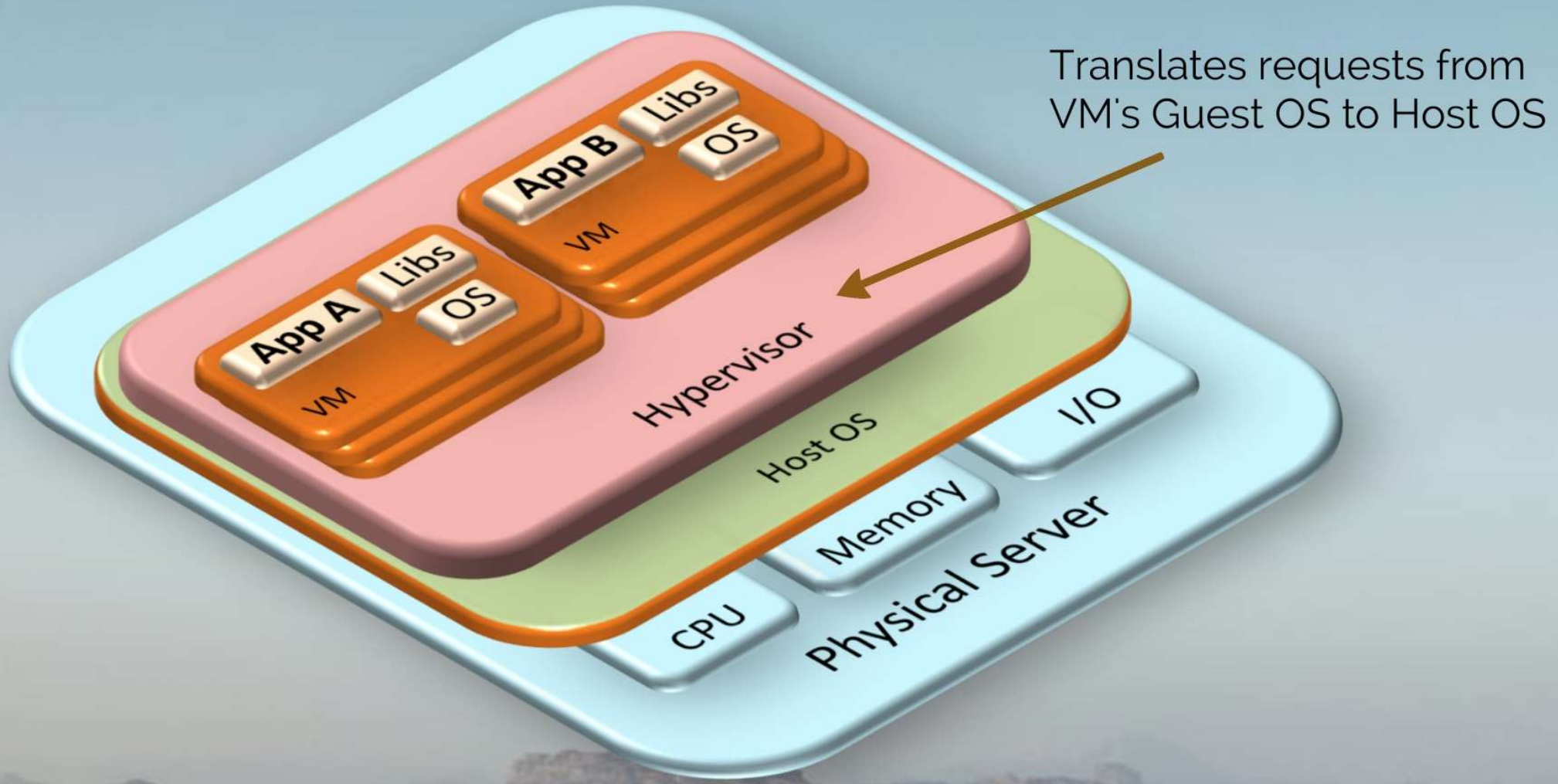
Infrastructure as code

Layers



Infrastructure as code

Before Containers, We had Virtual Machines



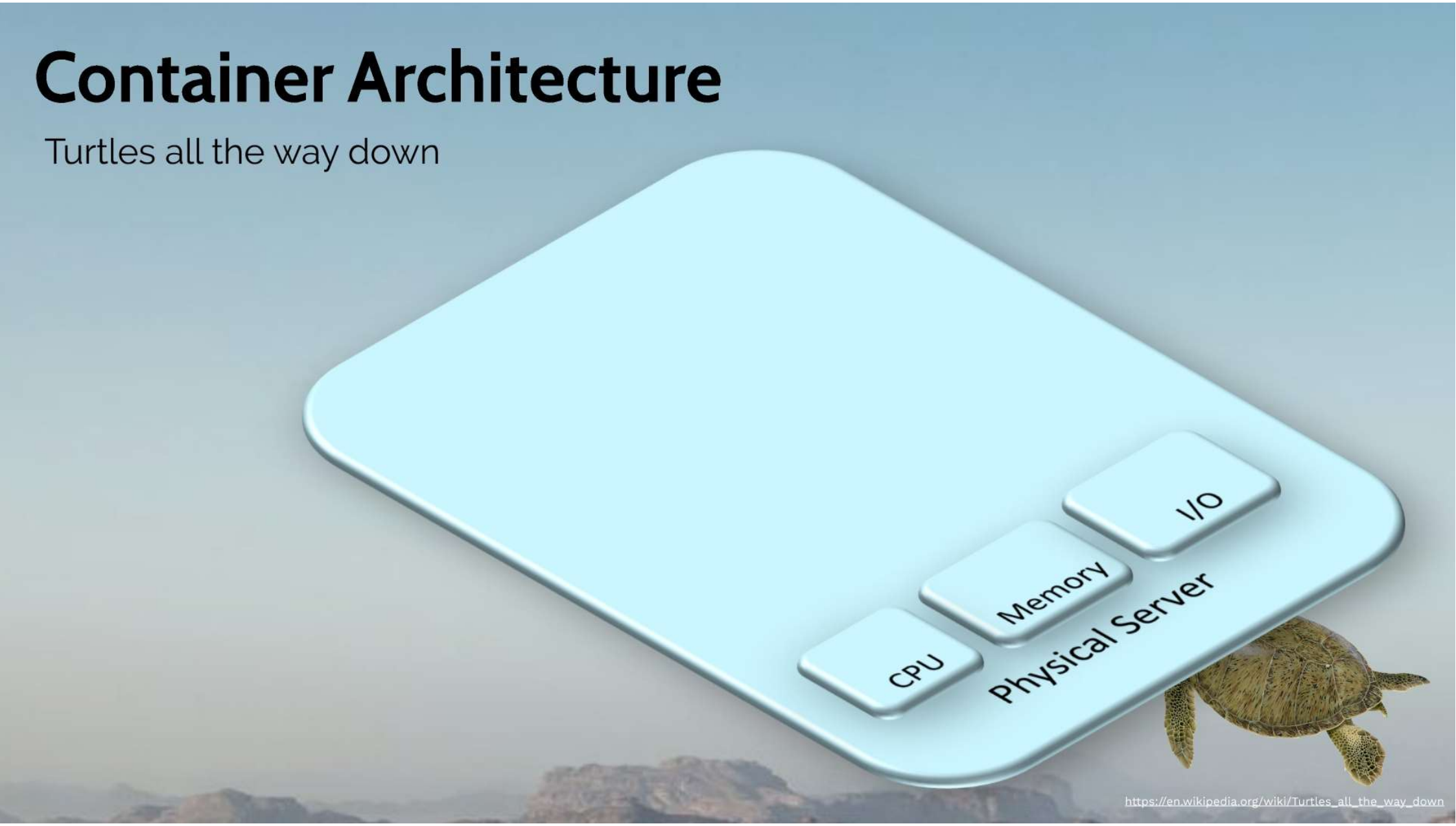
Container Architecture

Turtles all the way down

https://en.wikipedia.org/wiki/Turtles_all_the_way_down

Container Architecture

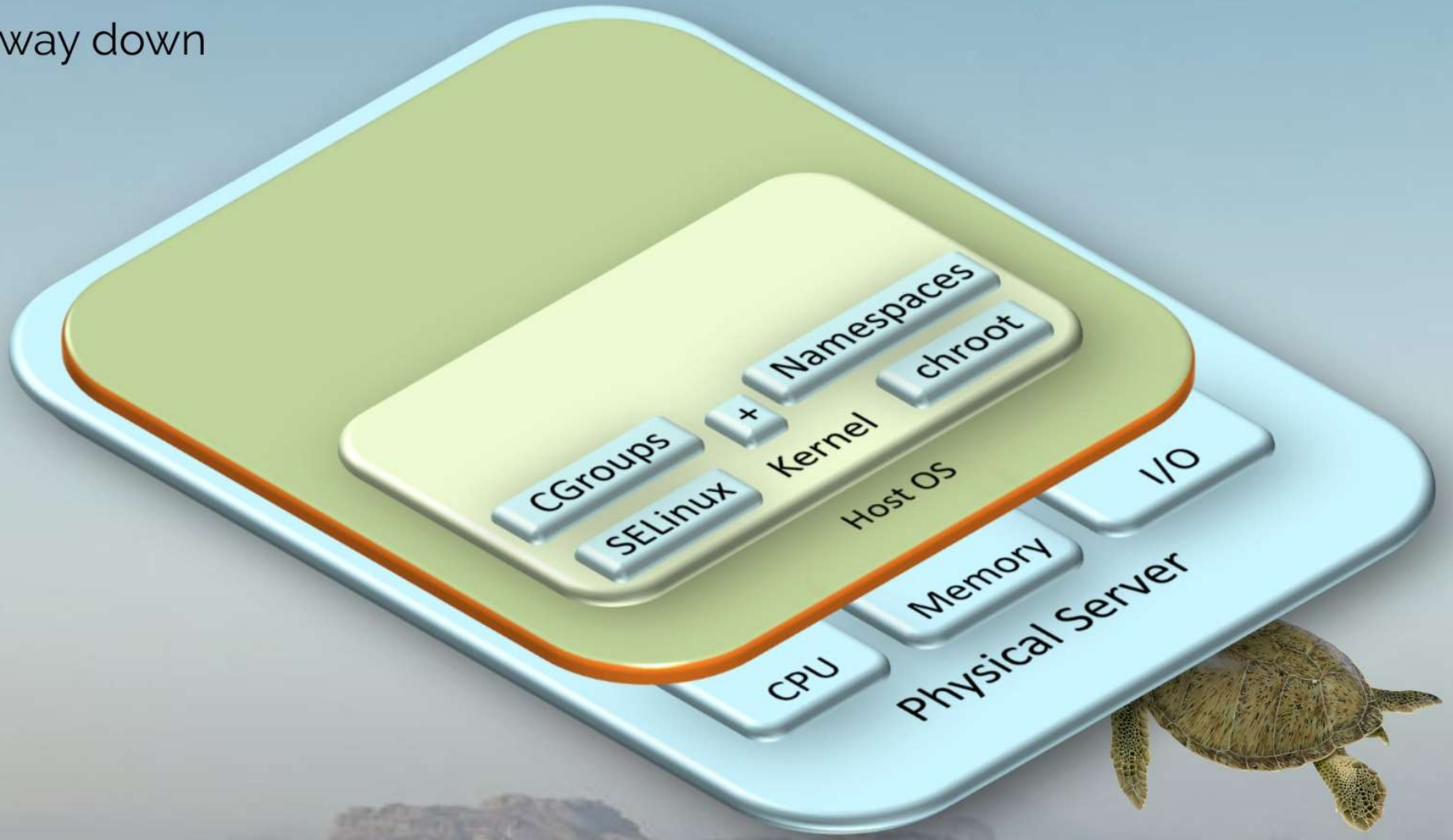
Turtles all the way down



https://en.wikipedia.org/wiki/Turtles_all_the_way_down

Container Architecture

Turtles all the way down

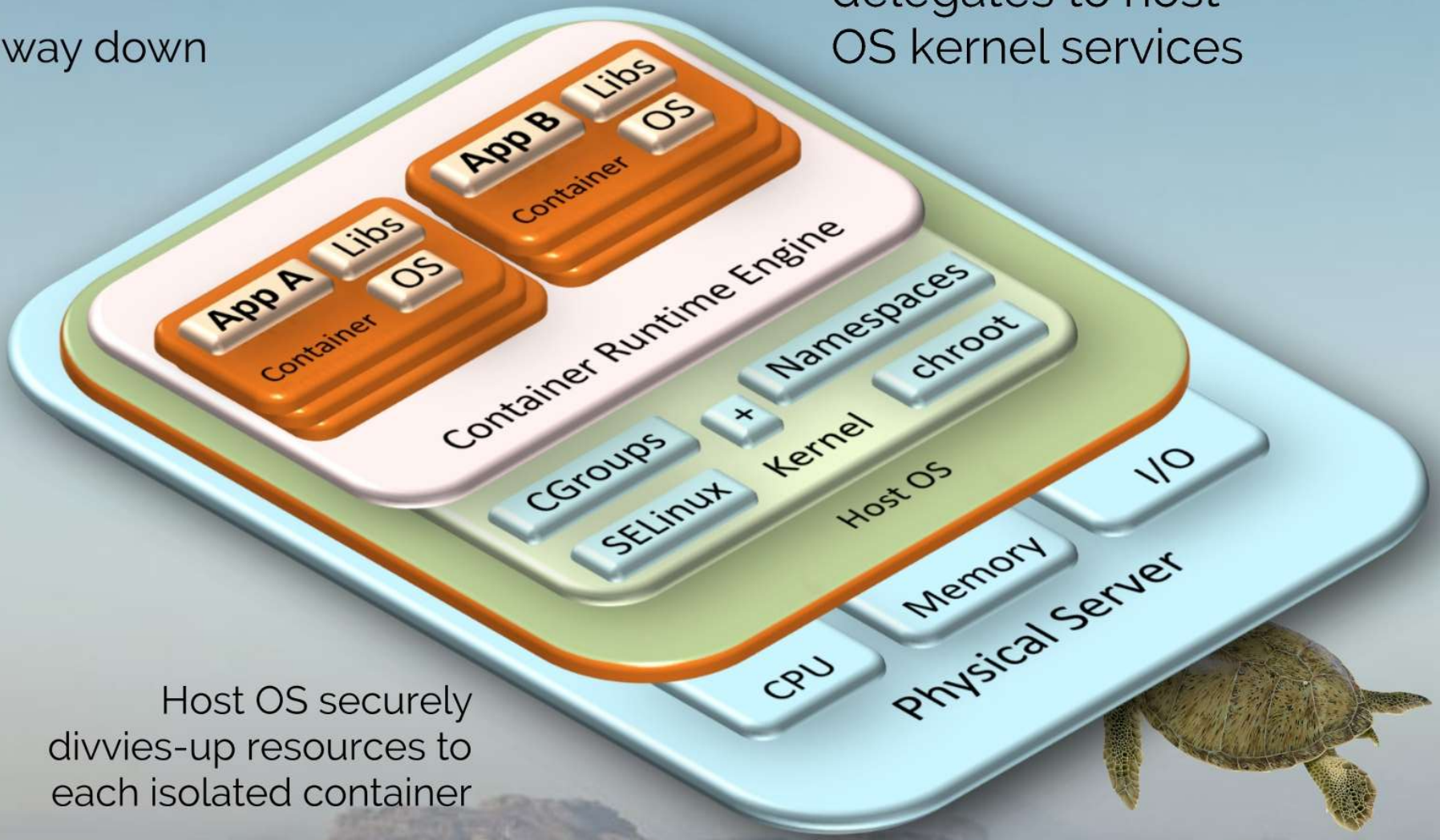


https://en.wikipedia.org/wiki/Turtles_all_the_way_down

Container Architecture

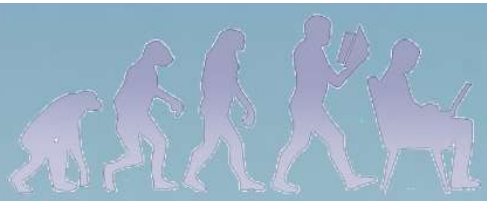
Turtles all the way down

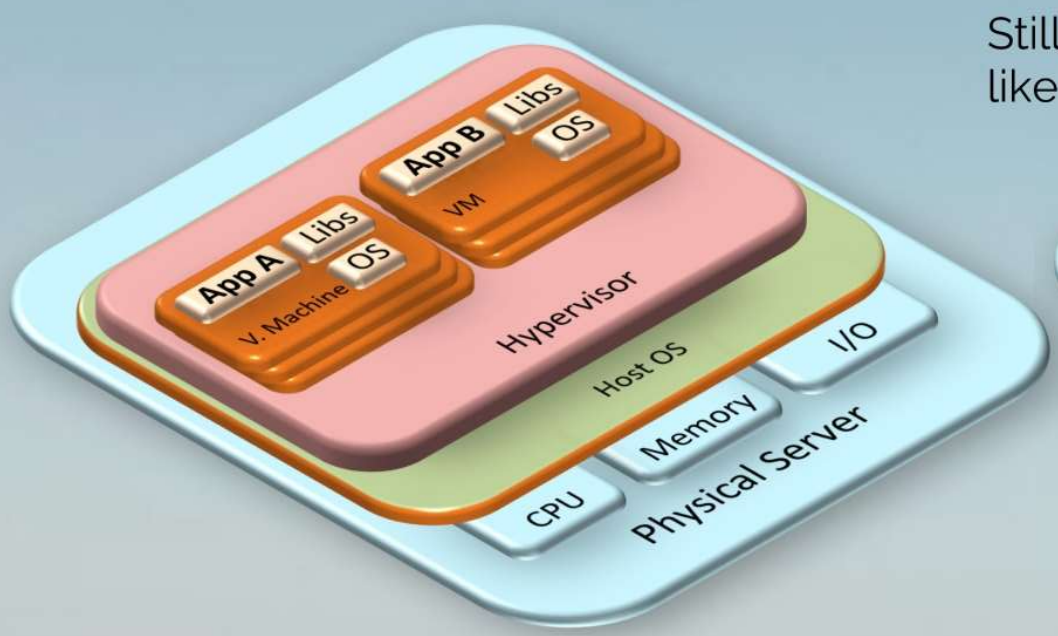
OS in container
delegates to host
OS kernel services

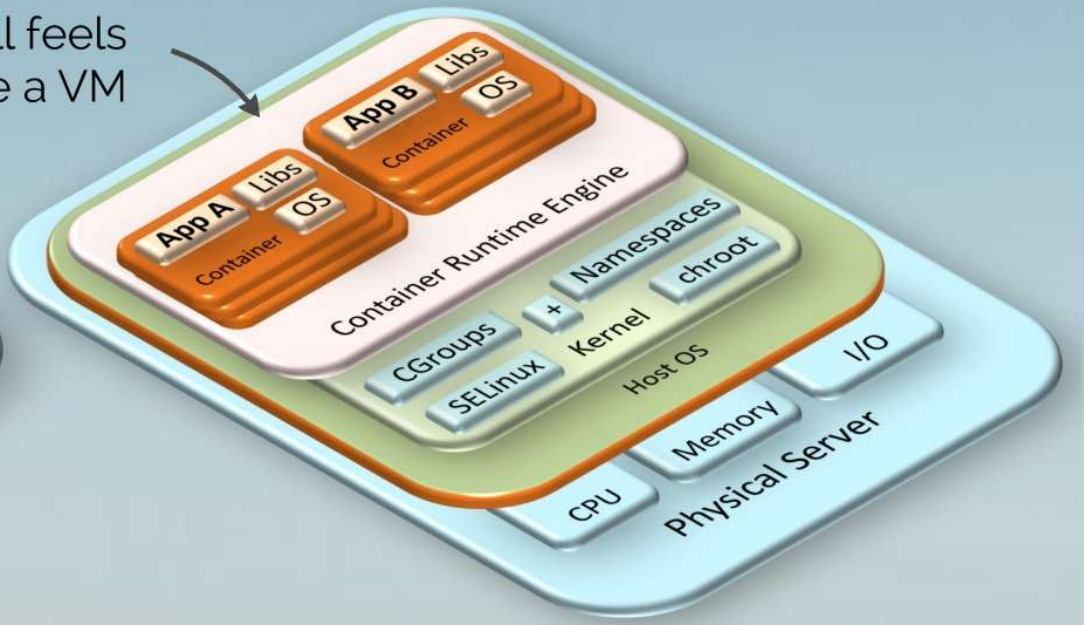


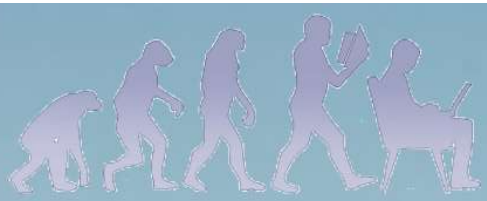
Host OS securely
divvies-up resources to
each isolated container

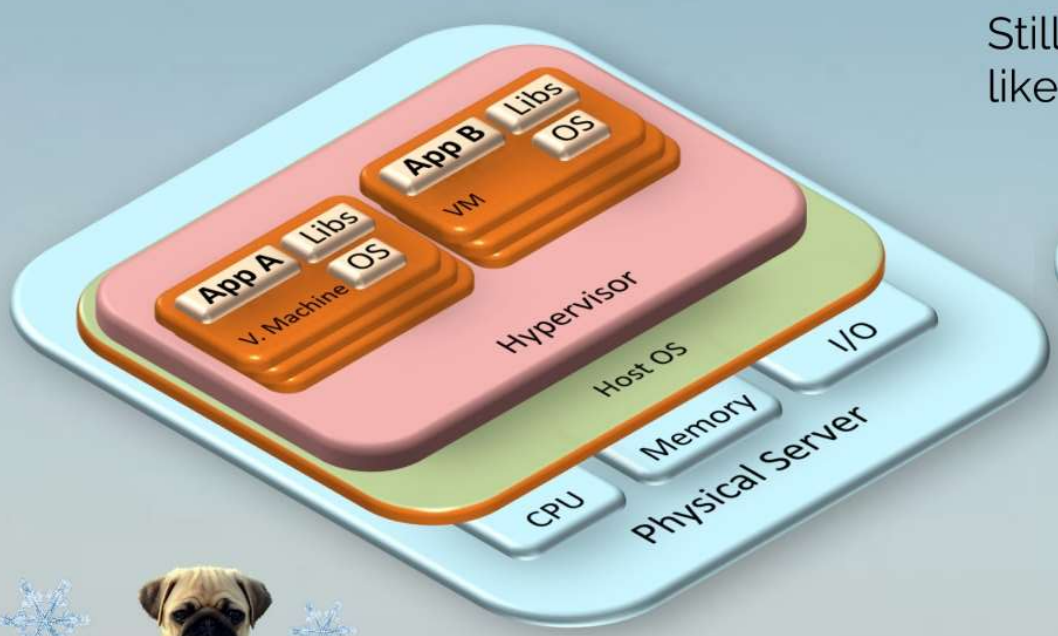
https://en.wikipedia.org/wiki/Turtles_all_the_way_down

Evolution: VMs  to Container Runtimes

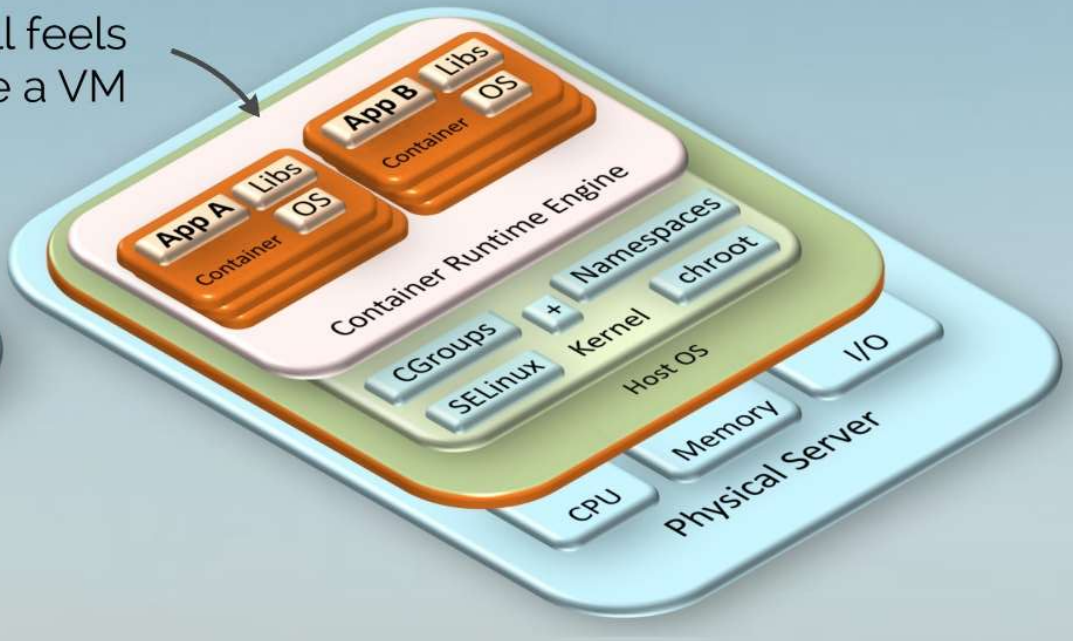




Evolution: VMs  to Container Runtimes

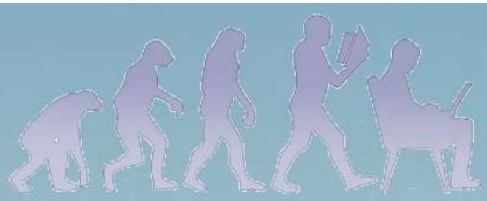


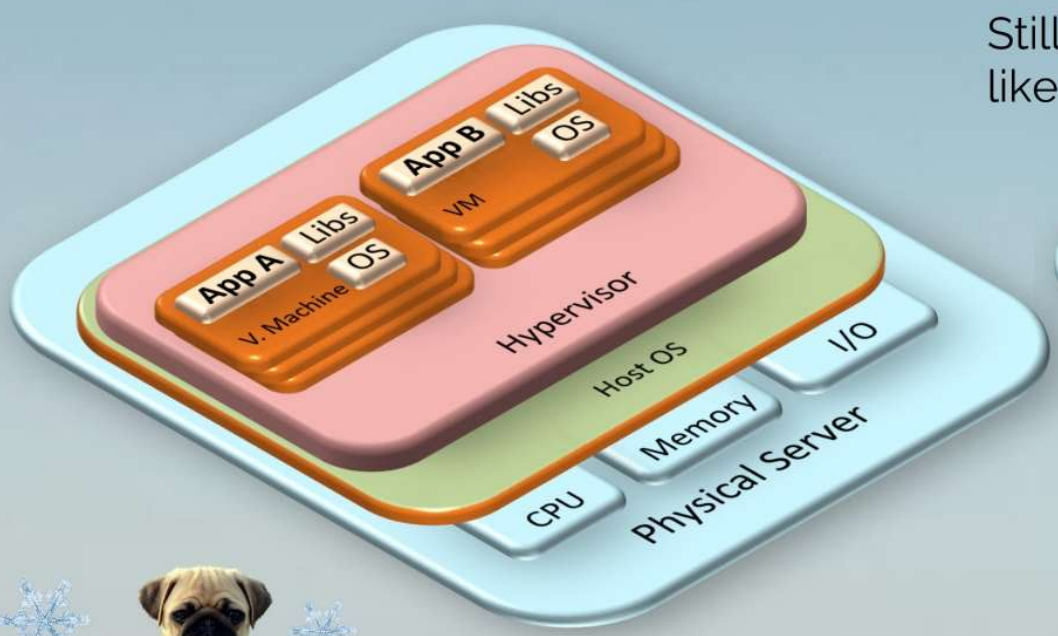
Still feels like a VM



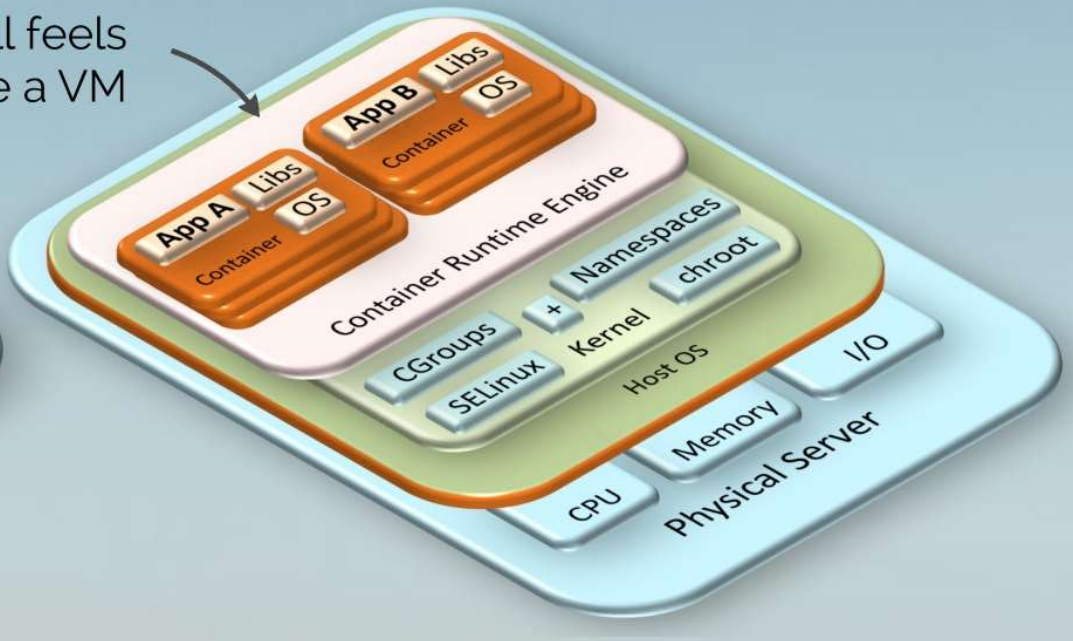
Unique servers, snowflakes GBs - minutes to start

https://martinfowler.com/bliki/SnowflakeServer.html

Evolution: VMs  to Container Runtimes



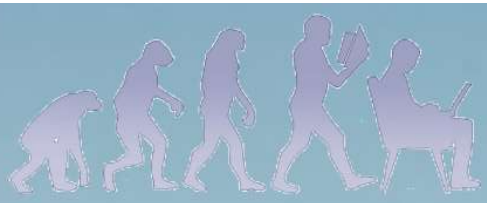
Still feels like a VM

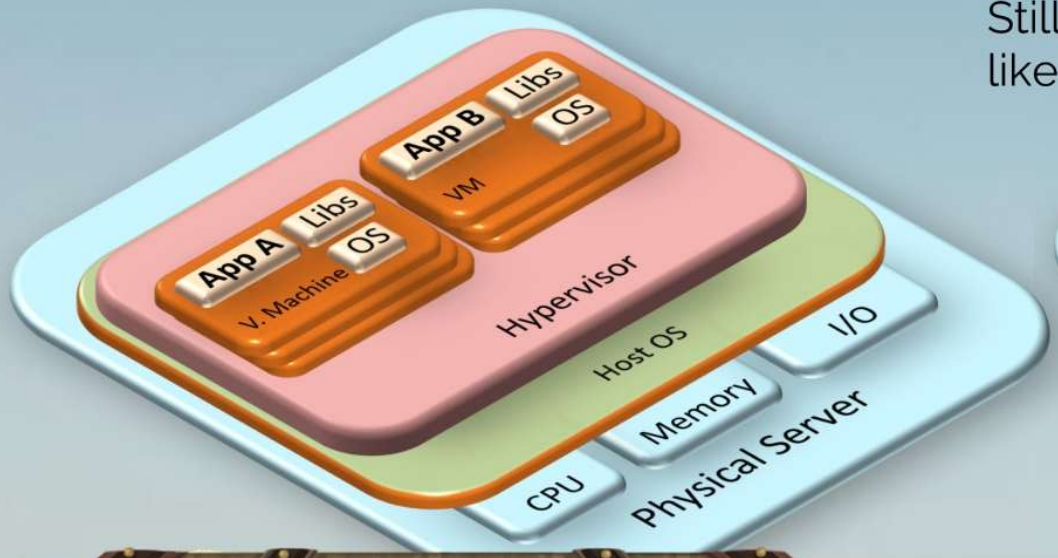


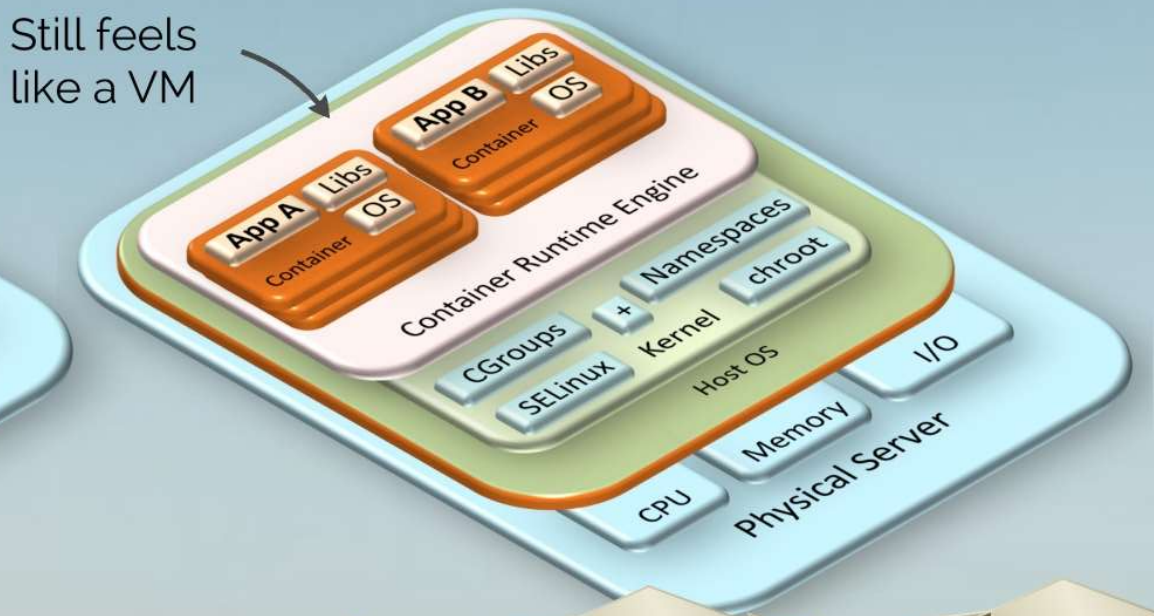
Unique servers, snowflakes GBs - minutes to start

https://martinfowler.com/bliki/SnowflakeServer.html

Disposable. ephemeral IPs MBs - seconds to start

Evolution: VMs  to Container Runtimes





Kernel Sleight of Hand for Containers

Kernel Sleight of Hand for Containers

chroot
Limit where
you run

Kernel Sleight of Hand for Containers

chroot
Limit where
you run

Security-Enhanced Linux
Limit what
you can access

Kernel Sleight of Hand for Containers

chroot
Limit where
you run

Control Groups
Limit what you
can use

Security-Enhanced Linux
Limit what
you can access

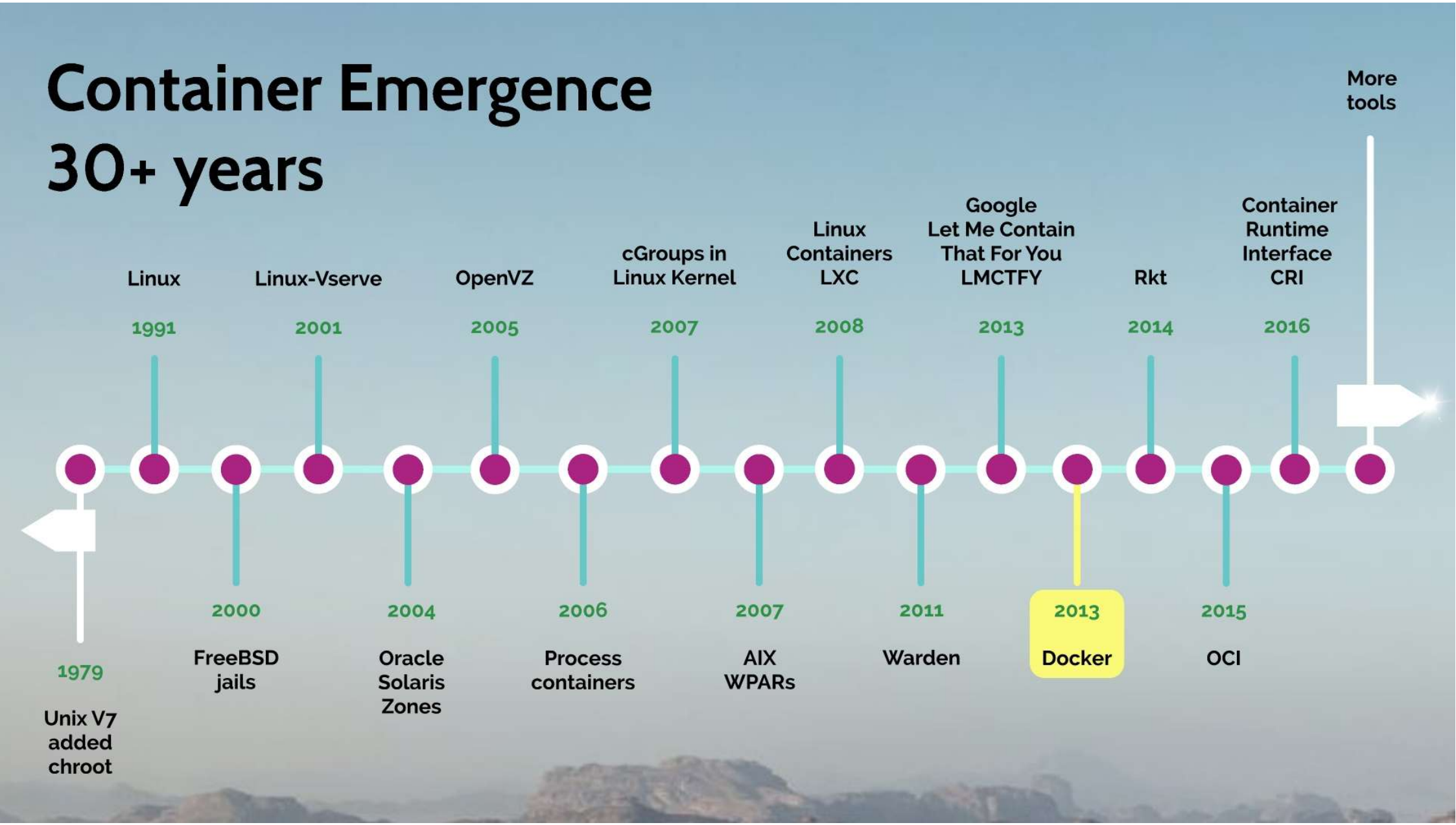
Kernel Sleight of Hand for Containers

chroot
Limit where
you run

Control Groups
Limit what you
can use

Namespaces
Limit what
you can see

Security-Enhanced Linux
Limit what
you can access


Container Emergence
30+ years
1979
Unix V7 added chroot
1991
Linux
2000
FreeBSD jails
2001
Linux-Vserve
2004
Oracle Solaris Zones
2005
OpenVZ
2006
Process containers
2007
cGroups in Linux Kernel
2007
AIX WPARs
2008
Linux Containers LXC
2011
Warden
2013
Google Let Me Contain That For You LMCTFY
2013
Docker
2014
Rkt
2015
OCI
2016
Container Runtime Interface CRI
More tools

Terminology

A few common terms

Container Image
A file of bytes following the container image format that defines the container. Stored in a repository. Often composed of multiple layers.

Image Layer
Container images are typically many layers of images. Each layer defines distinct and reusable container definitions. FROM, COPY, ADD and RUN add new layers

Container Engine
Process on the Host OS that accepts requests to pull images, calls the container runtime and cleans up after containers terminate.

Registry
Library or datastore of multiple repositories. Hosts and manages the repo files and manifests (add, remove, list, security).

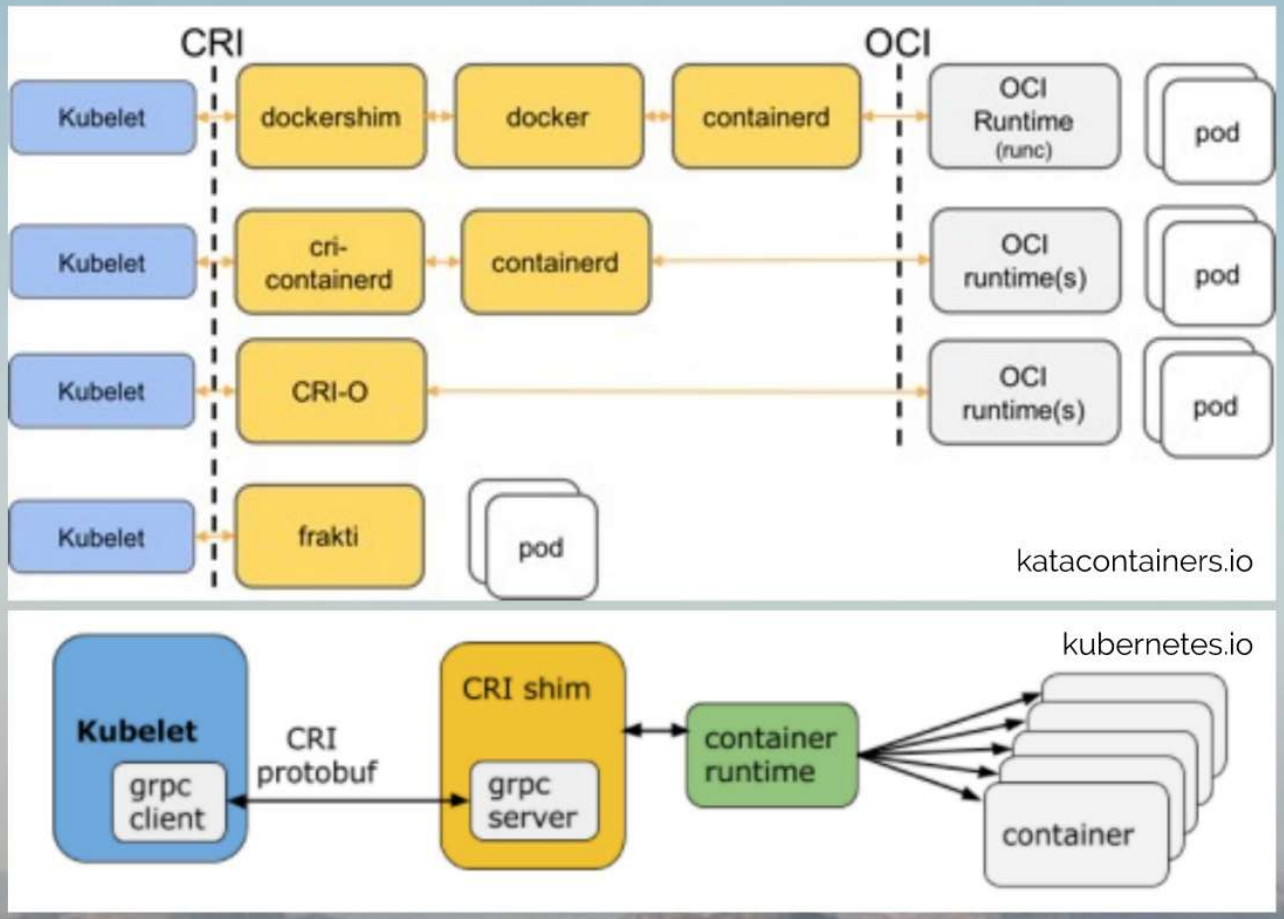
Repository
Listing of one or more versions of a container along with its images layers and manifests. Follows established image manifest schema.

Tag
A label that uniquely identifies multiple container images in a single repository.

Base Image
Most containers build upon base container images layers. and build more layers on top of them. Encourages reuse.

CRI and OCI

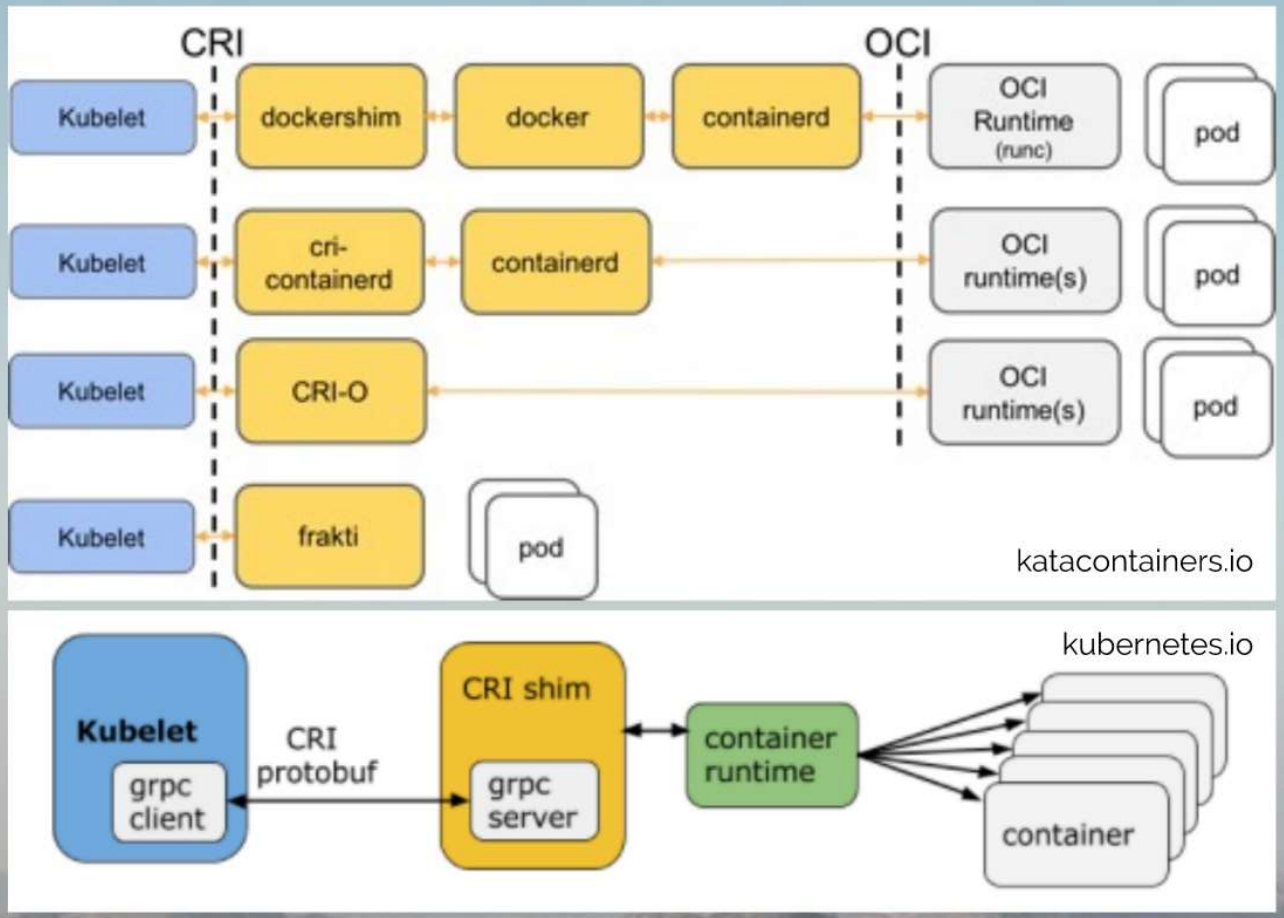
OPEN CONTAINER INITIATIVE



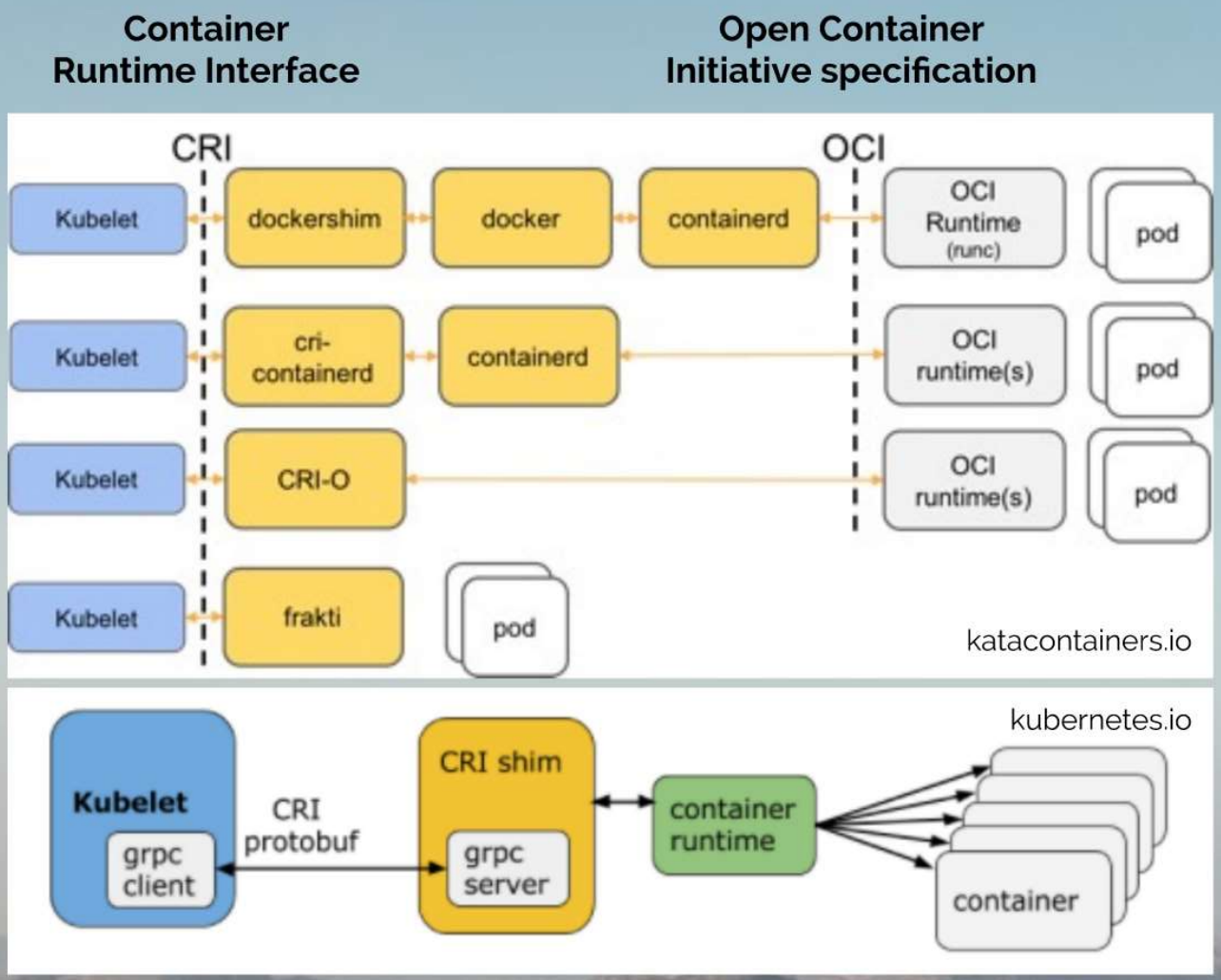
CRI and OCI

OPEN CONTAINER INITIATIVE

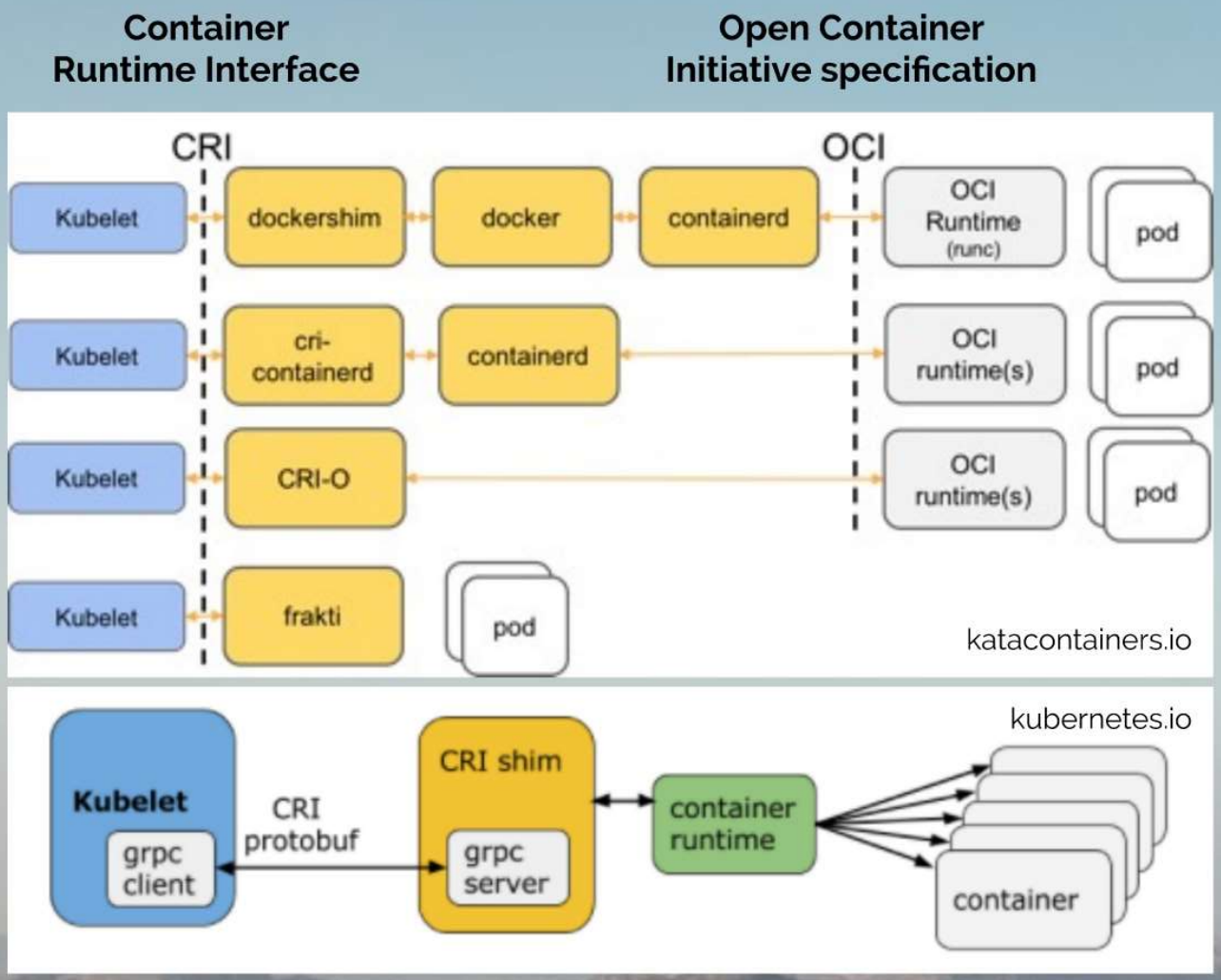
Container Runtime Interface



CRI and OCI


Container
Runtime Interface
Open Container
Initiative specification
CRI
OCI
Kubelet
dockershim
docker
containerd
OCI
Runtime
(runc)
pod
Kubelet
cri-
containerd
containerd
OCI
runtime(s)
pod
Kubelet
CRI-O
OCI
runtime(s)
pod
Kubelet
frakti
pod
katacontainers.io
kubernetes.io
Kubelet
grpc
client
CRI
protobuf
CRI shim
grpc
server
container
runtime
container

CRI and OCI

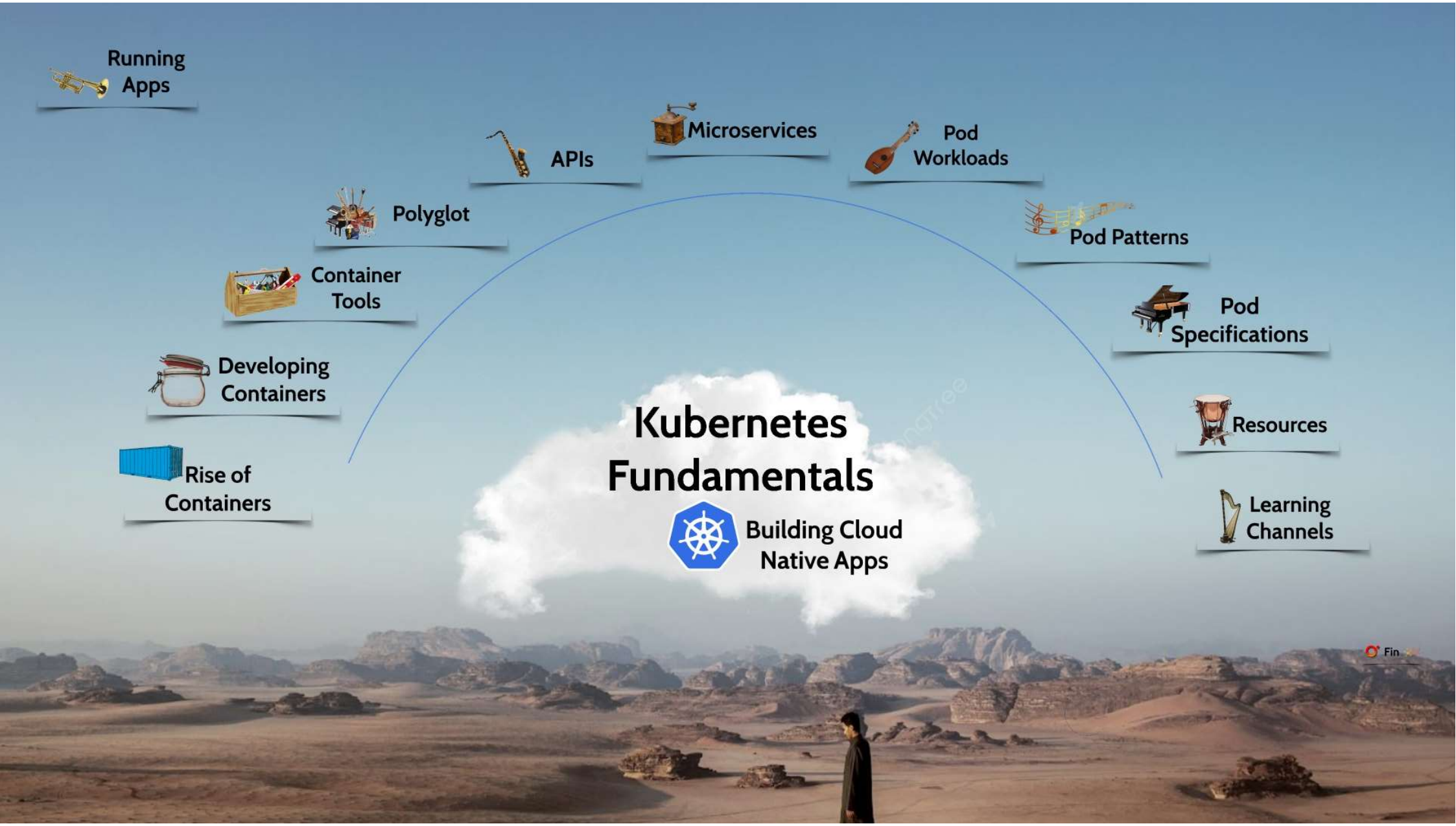


OPEN CONTAINER INITIATIVE

OCI container runtimes with Kubelet CRI

- Docker (nix Q4 2020)
- cri-o (OpenShift)
- **cri-containerd** (default)
  - runc
  - gVisor
  - Kata containers
  - Firecracker
  - Nabla containers
- crun (c based)
- Rktlet (CRI for Rkt)
- rktnetes
- Frakti
- Singularity
- * Virtual Kubelet

CNCF Container Runtime Engines


Running Apps
APIs
Microservices
Pod Workloads
Polyglot
Pod Patterns
Container Tools
Pod Specifications
Developing Containers
Resources
Rise of Containers
Learning Channels
Kubernetes Fundamentals
Building Cloud Native Apps

Distillation Factors

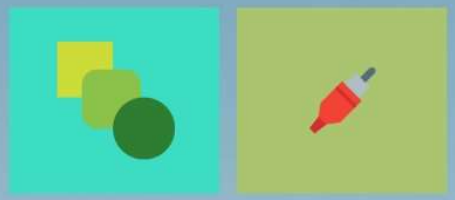
8 Factors

Distillation Factors

8 Factors

High cohesion

All features in the container are well defined, and used.
"Do one thing and do it well." -- Unix



Low coupling

Exposed interfaces are minimal, simple, and stable.

Distillation Factors

8 Factors

High cohesion

All features in the container are well defined, and used.
"Do one thing and do it well." -- Unix

Low coupling

Exposed interfaces are minimal, simple, and stable.

Immutable

Container not capable of or susceptible to change.

Idempotent

Multiple instances all behave the same, over time.

Distillation Factors

8 Factors

High cohesion

All features in the container are well defined, and used.
"Do one thing and do it well." -- Unix

Low coupling

Exposed interfaces are minimal, simple, and stable.

Immutable

Container not capable of or susceptible to change.

Idempotent

Multiple instances all behave the same, over time.

Attack phobic

Security attack vectors are resisted as potential surfaces have been minimized.

Small images

Less is more. Reduce I/O pressure. Images are revised, stored, transported, downloaded and scaled.

Distillation Factors

8 Factors

High cohesion

All features in the container are well defined, and used.
"Do one thing and do it well." -- Unix

Low coupling

Exposed interfaces are minimal, simple, and stable.

Immutable

Container not capable of or susceptible to change.

Idempotent

Multiple instances all behave the same, over time.

Attack phobic

Security attack vectors are resisted as potential surfaces have been minimized.

Small images

Less is more. Reduce I/O pressure. Images are revised, stored, transported, downloaded and scaled.

Rapid startup

Containers are embraced as ephemeral, fragile, expendable, and scalable cattle.

Resource frugality

Every bit and CPU tick is a limited resource as containers are scaled throughout your datacenter.

Slim toolkit

COMING SOON

SlimAI

Inspects, optimizes, and secures container images by removing unnecessary components

Java application images:

from ubuntu:14.04 - 743.6 MB → 100.3 MB

Node.js application images:

ubuntu:14.04 - 432MB → 14MB (by 30.85 x)

debian:jessie - 406MB → 25.1MB (16.21 x)

node:alpine - 66.7MB → 34.7MB (1.92 x)

node:distroless - 72.7MB → 39.7MB (1.83 x)

Python application images:

ubuntu:14.04 - 438MB → 16.8MB (by 25.99 x)

python:2.7-alpine - 84.3MB → 23.1MB (by 3.65 x)

python:2.7.15 - 916MB → 27.5MB (by 33.29 x)

centos:7 - 647MB → 23MB (by 28.57 x)

centos/python-27-centos7 - 700MB → 24MB (29.01 x)

python2.7:distroless - 60.7MB → 18.3MB (by 3.32 x)

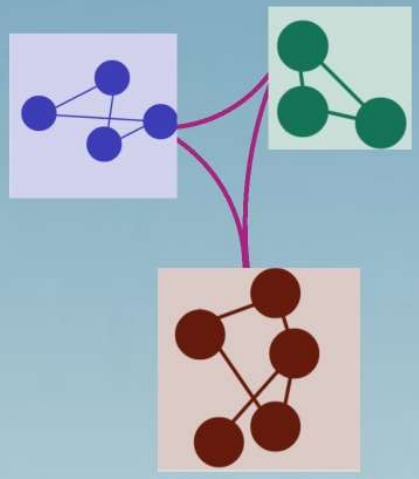
Two Delicious Architectures

Containers

Two Delicious Architectures

Containers

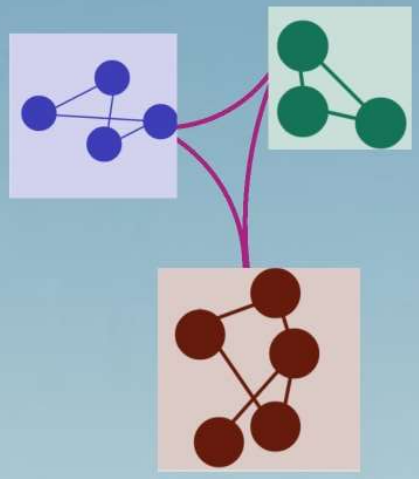
Microservices



Two Delicious Architectures

Containers

Microservices



Tasty Combination

Microservices

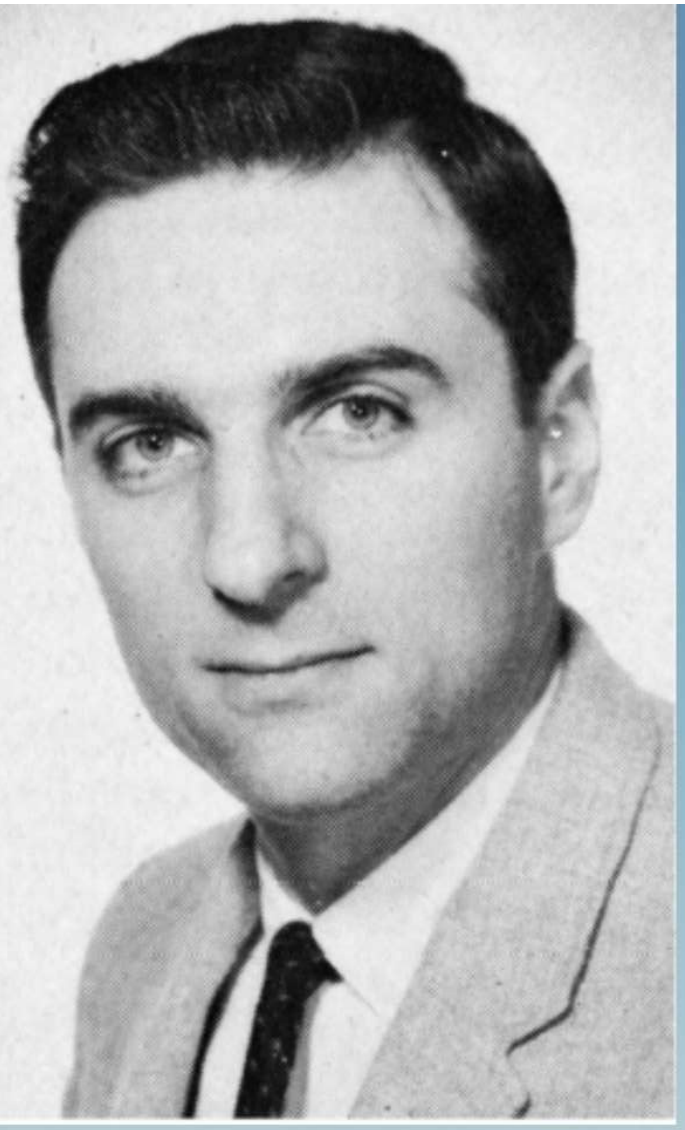
- Apps as suites of independently deployable services
- Modular: High cohesion, low coupling
- Often own small datastore
- Scaling at a micro level
- Do one thing and do it well
- Sometimes they feel like OO classes
- Monolith antimatter

Microservices defined

James Lewis

- Unix philosophy

Martin Fowler



Team organization

"Any organization that designs a system (defined more broadly here than just information systems) will inevitably produce a design whose structure is a copy of the organization's communication structure."

— Melvin E. Conway, 1968

Each team continuously delivers solutions, independently

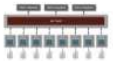
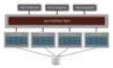
Team

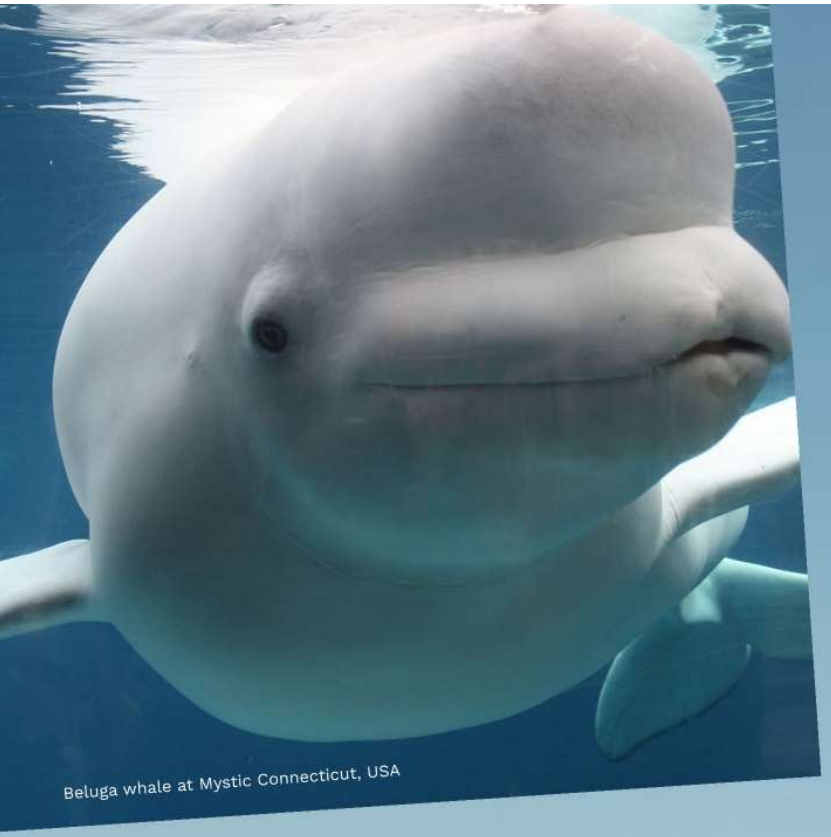
Communication and size

Jeff Bezos

- No matter how large your company gets, individual teams shouldn't be larger than what two pizzas can feed.
- API Mandate: All teams only communication through APIs, else you are fired.

space-based architecture

	 layered monolith	 microkernel	 microservices	 service-based	 event-driven	 space-based
agility	★	★★★	★★★★★	★★★★	★★★★	★★★
deployment	★	★★★	★★★★★	★★★★	★★★	★★★
testability	★★	★★★	★★★★★	★★★★	★★	★
performance	★★★	★★★	★★	★★★	★★★★★	★★★★★
scalability	★	★	★★★★★	★★★	★★★★	★★★★★
elasticity	★	★	★★★★	★★	★★★	★★★★★
simplicity	★★★★★	★★★★	★	★★★	★	★
fault-tolerance	★	★	★★★★★	★★★★	★★★★★	★★★
evolvability	★	★★★	★★★★★	★★★★	★★★★★	★★★
total cost	★★★★★	★★★★★	★	★★★★	★★★	★★



Deployment

ReplicaSet

DaemonSet

StatefulSet

Job &
CronJob

Five kinds

Pods used by built-in workload resources

kind: Deployment

- Most common
- Rollout / Rollback
- Pause
- Image updates

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.7.9
        ports:
        - containerPort: 80
```