Architetture Software a Microservizi
Corso di Laurea Magistrale in Informatica

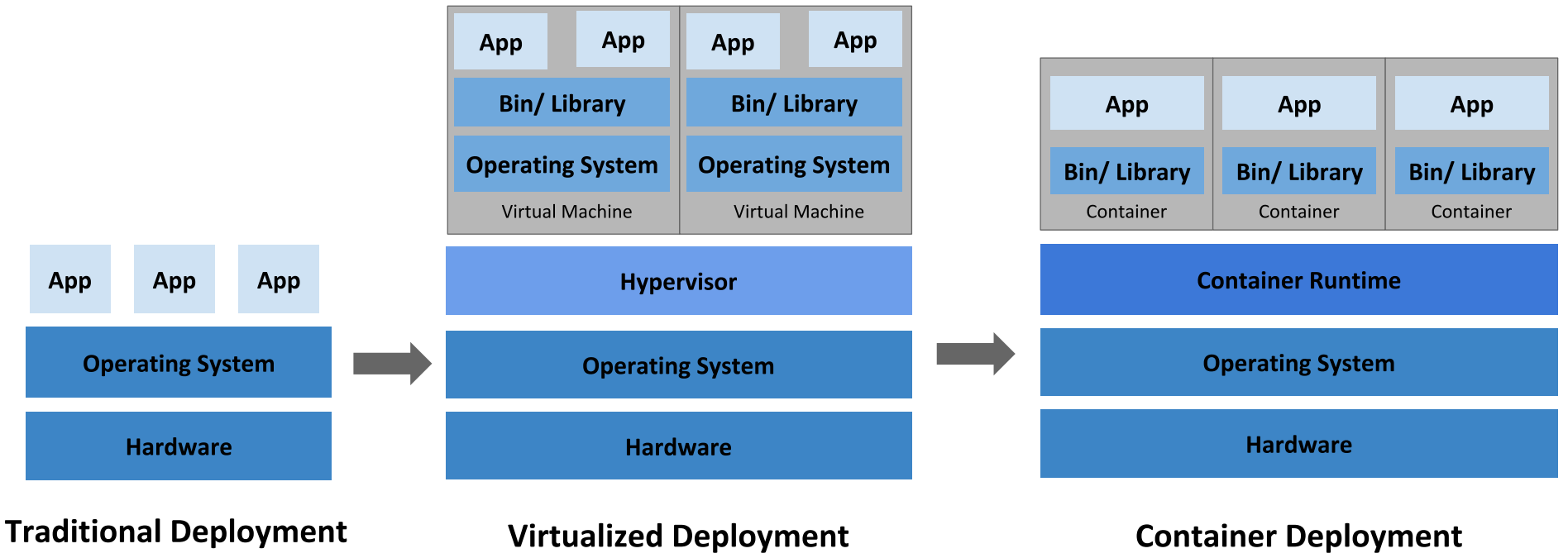
Light virtualization (containers)

Davide Rossi
Dipartimento di Informatica – Scienze e Ingegneria
Università di Bologna

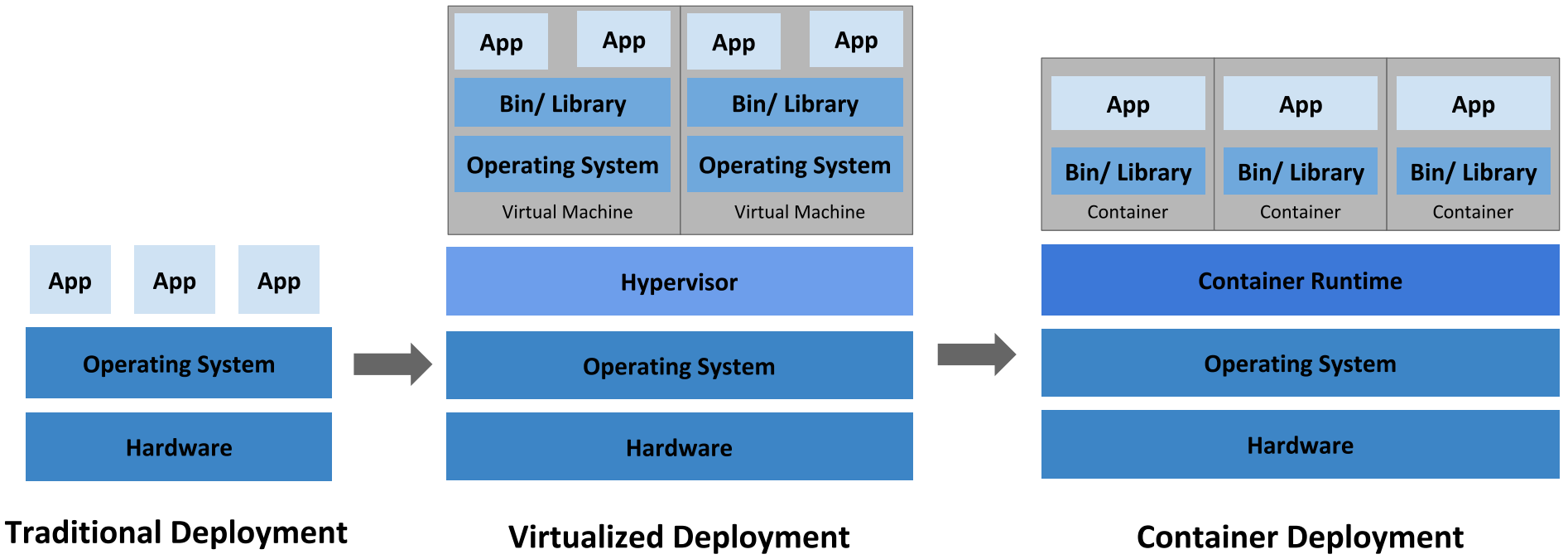
What is a container

- A container is an isolated partition of the OS
- Containers are directly supported by the OS (that is: Docker is just a management tool)

Containers and *light* virtualization



Containers and *light* virtualization



Notice that on Windows containers are really *light* VMs, i.e. they run on top of a hypervisor.

Containers on Linux

Linux manages containers using

- namespaces
  - create partitions for OS managed abstractions
- cgroups
  - limit processes access to resources

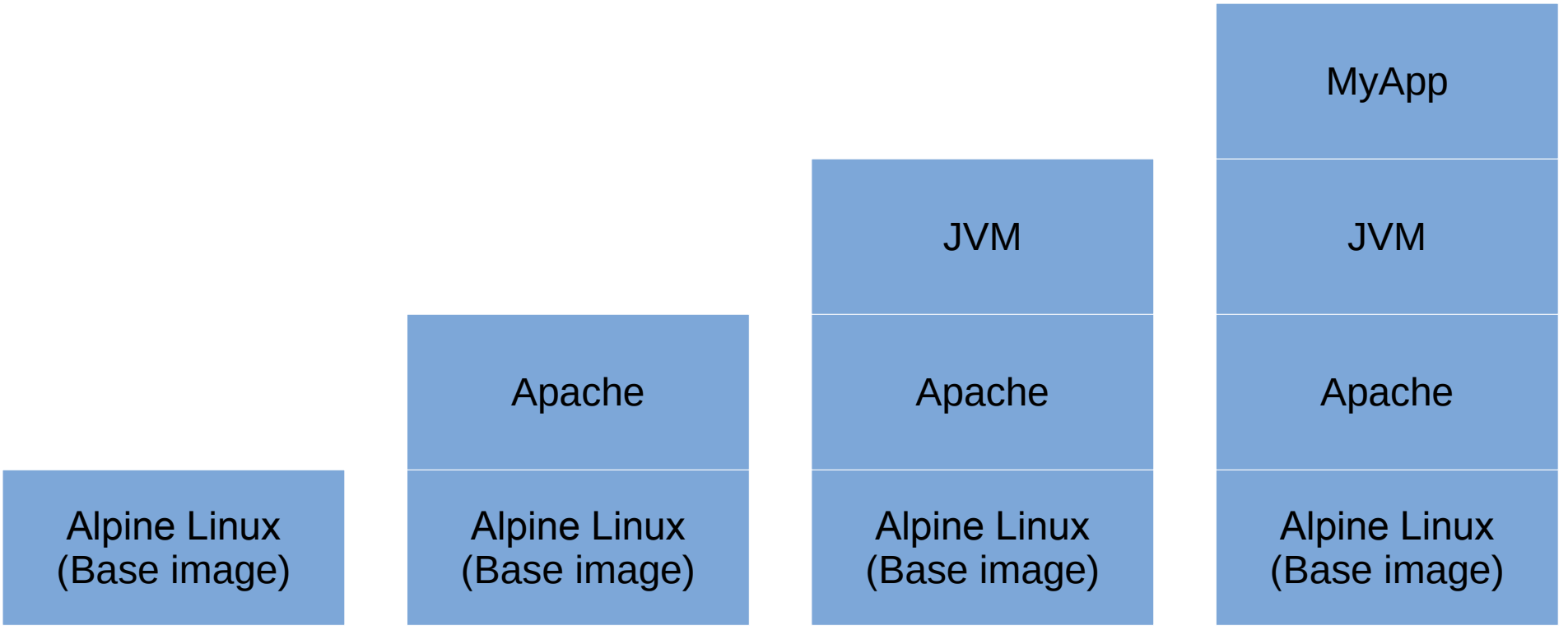
Filesystem

- A container usually has its own FS
- While this could just be a chrooted partition of the host FS, it is usually a loopback/*layered* filesystem

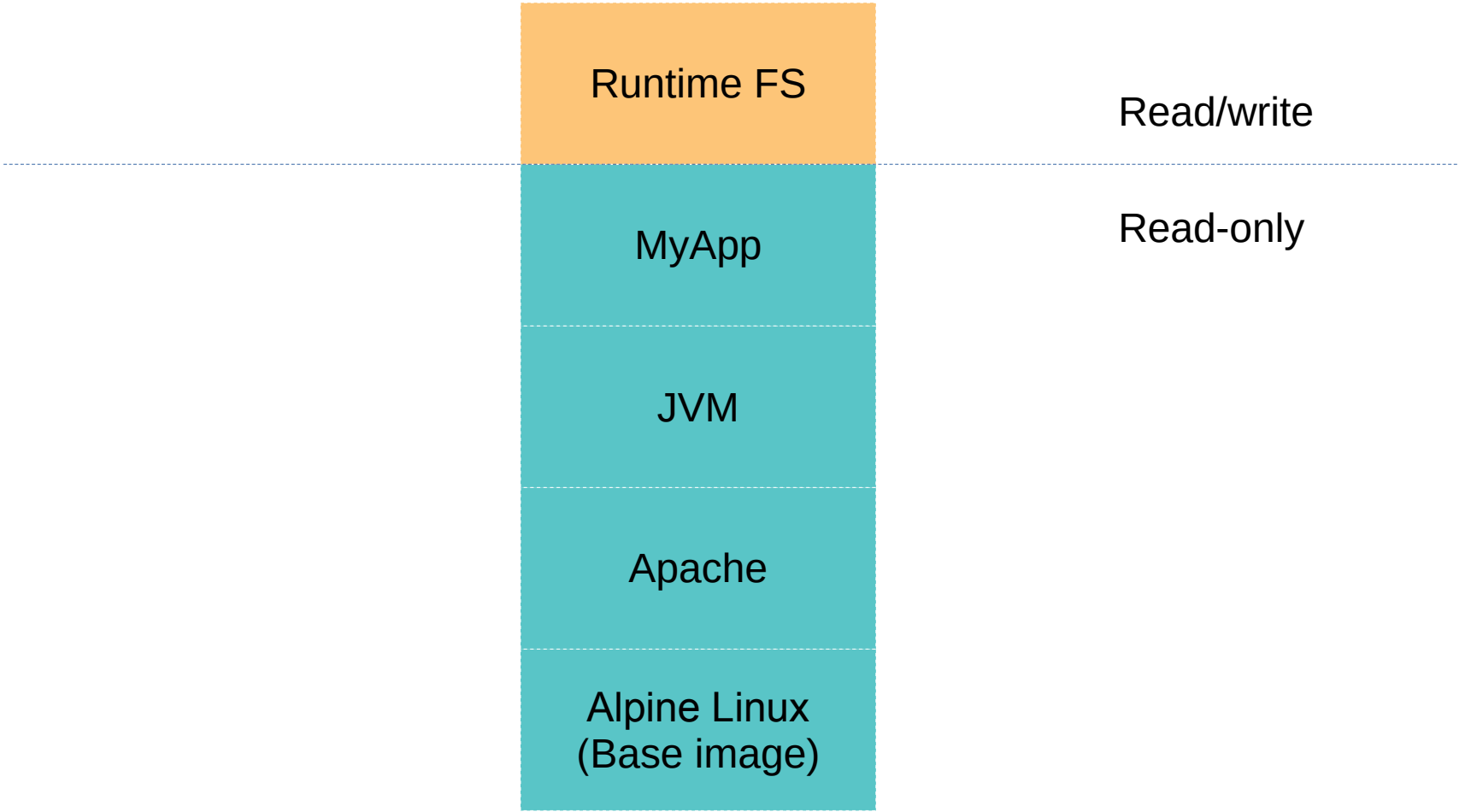
Container images

- Bundles containing all that is needed to setup a container
  - FS content
  - Initial process
  - Misc metadata & configuration bits

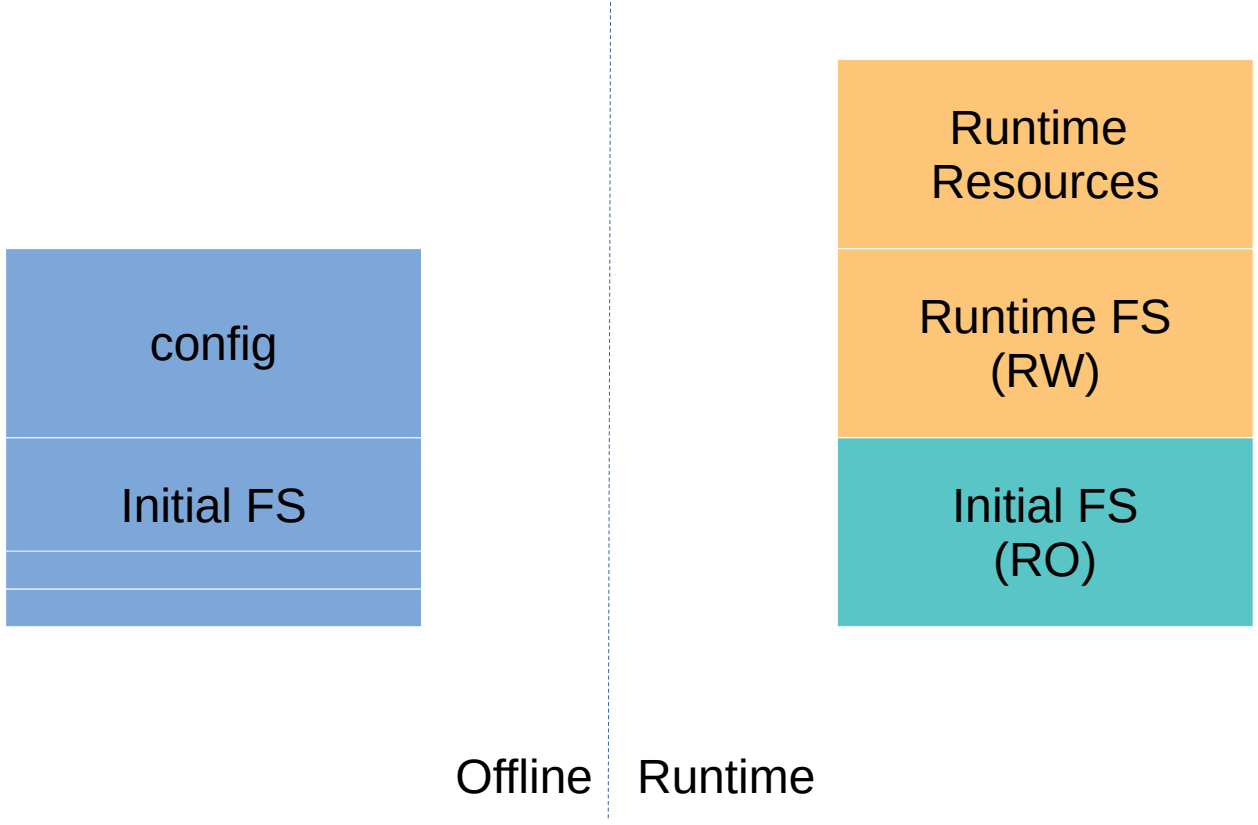
Layered FS images



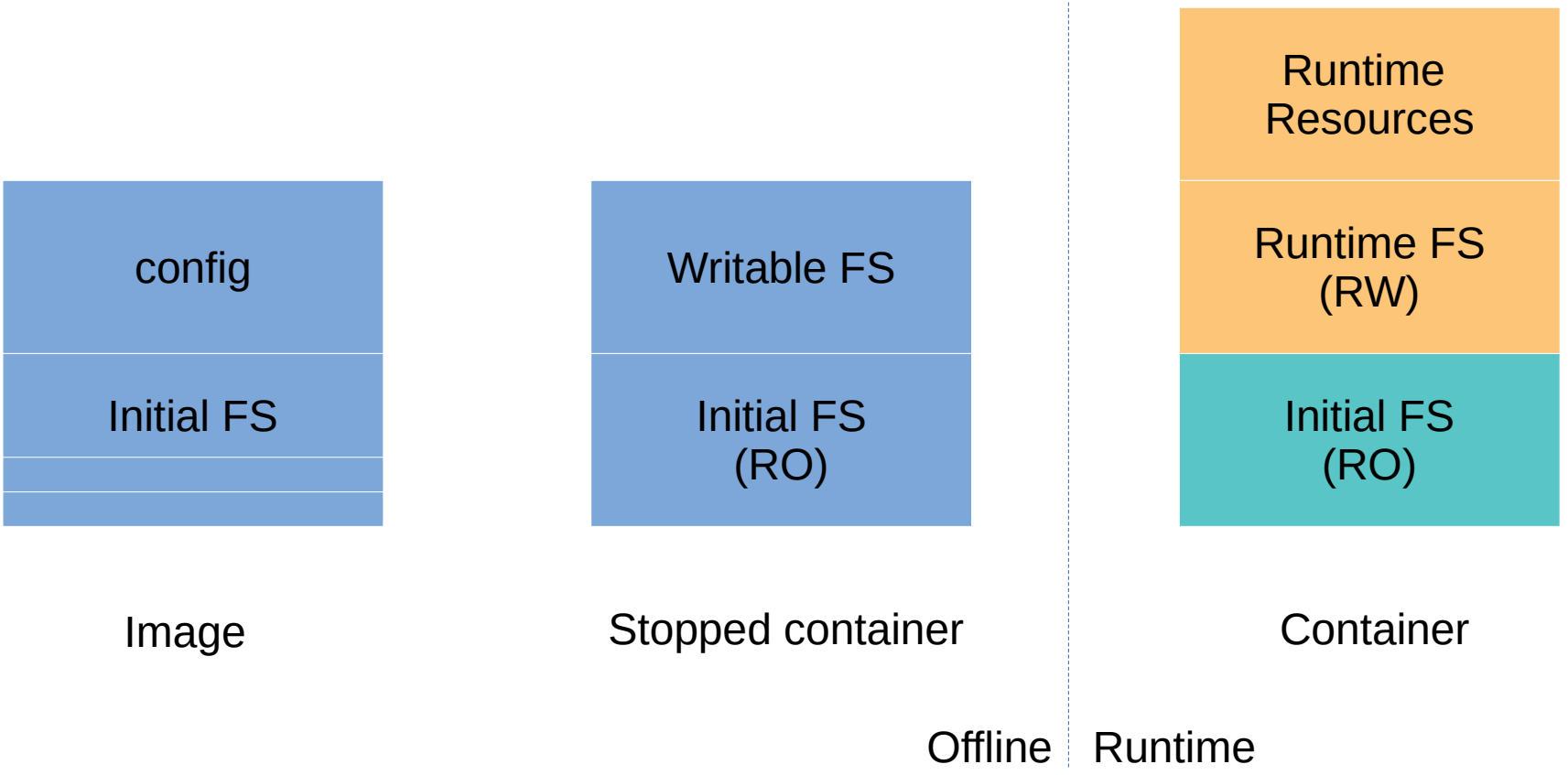
Runtime FS


Runtime FS
MyApp
JVM
Apache
Alpine Linux
(Base image)
Read/write
Read-only

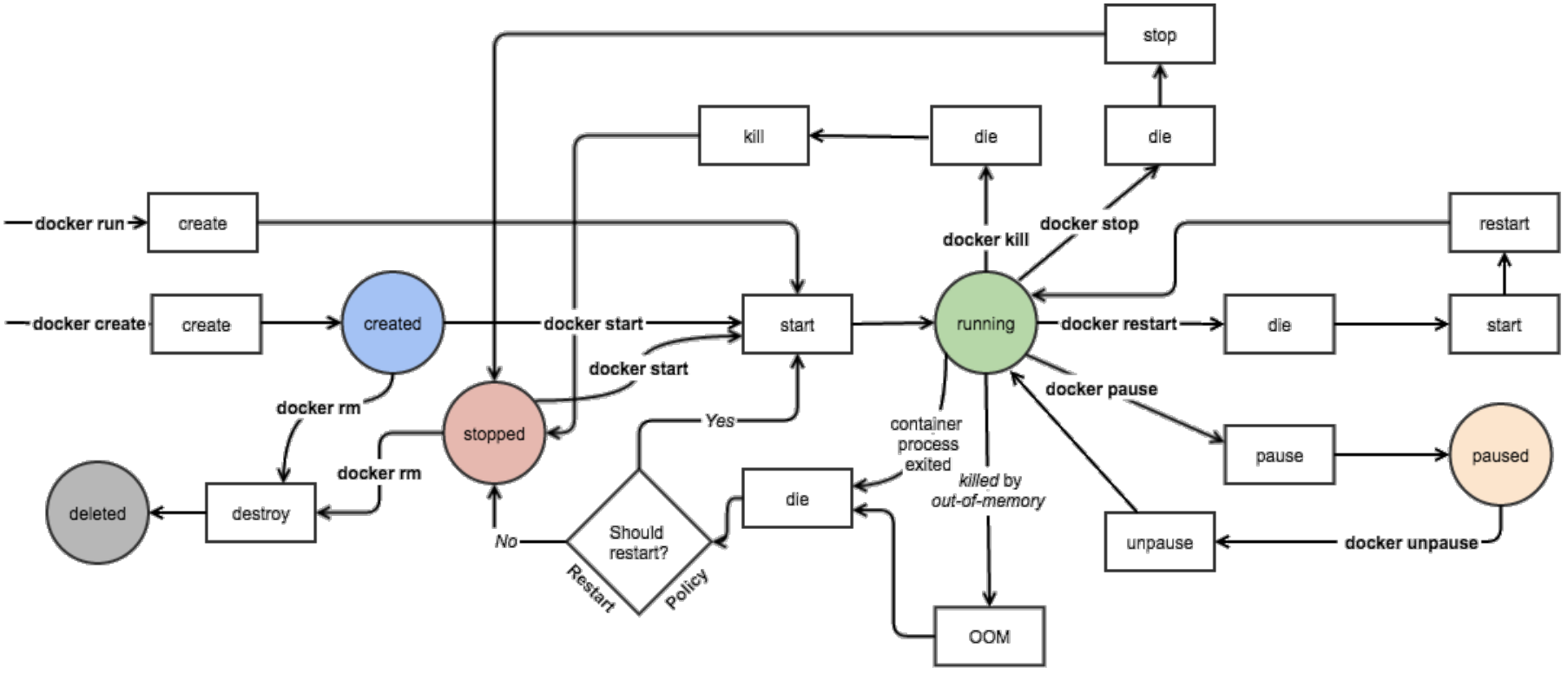
Images vs containers



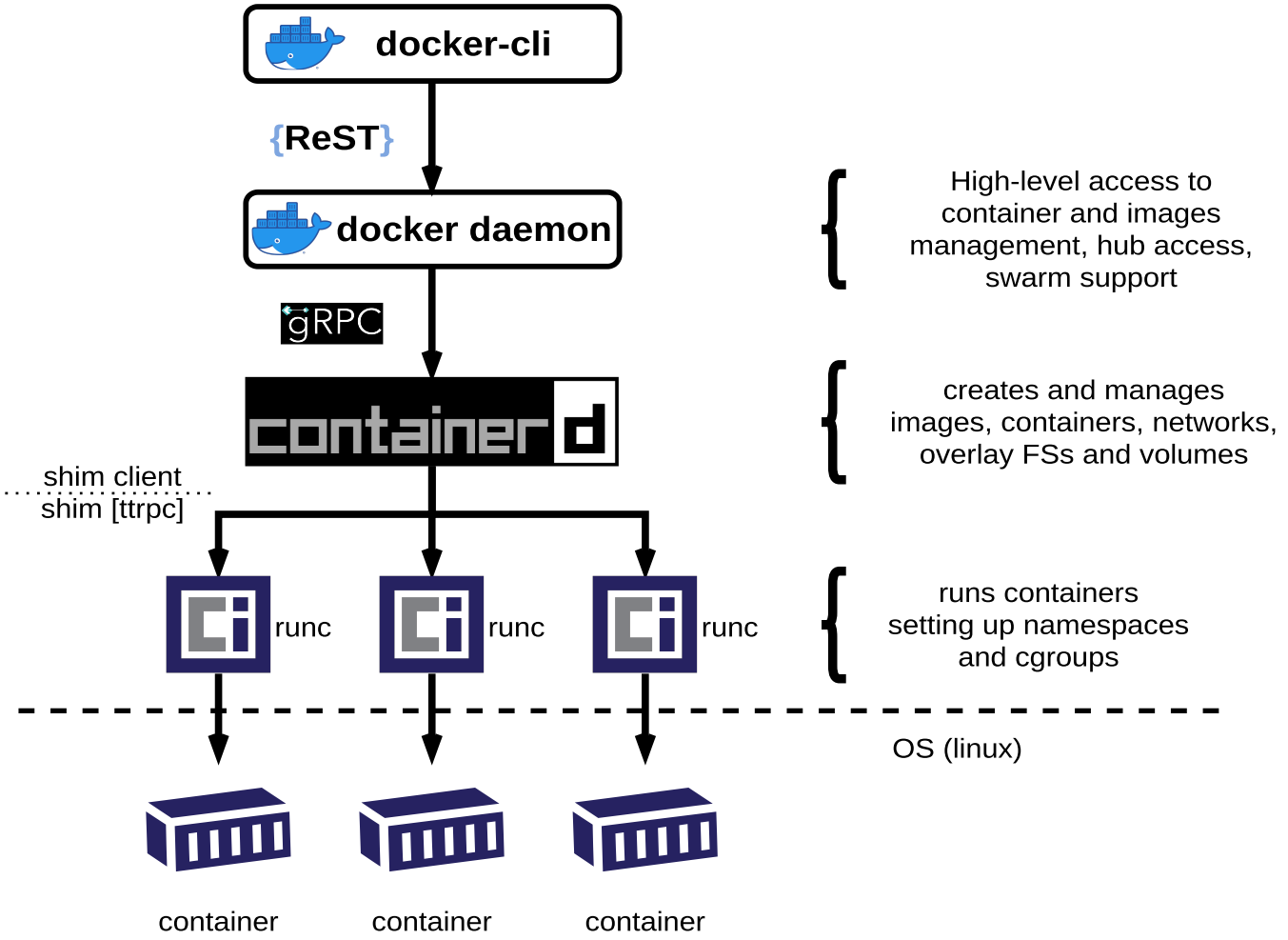
The Docker view



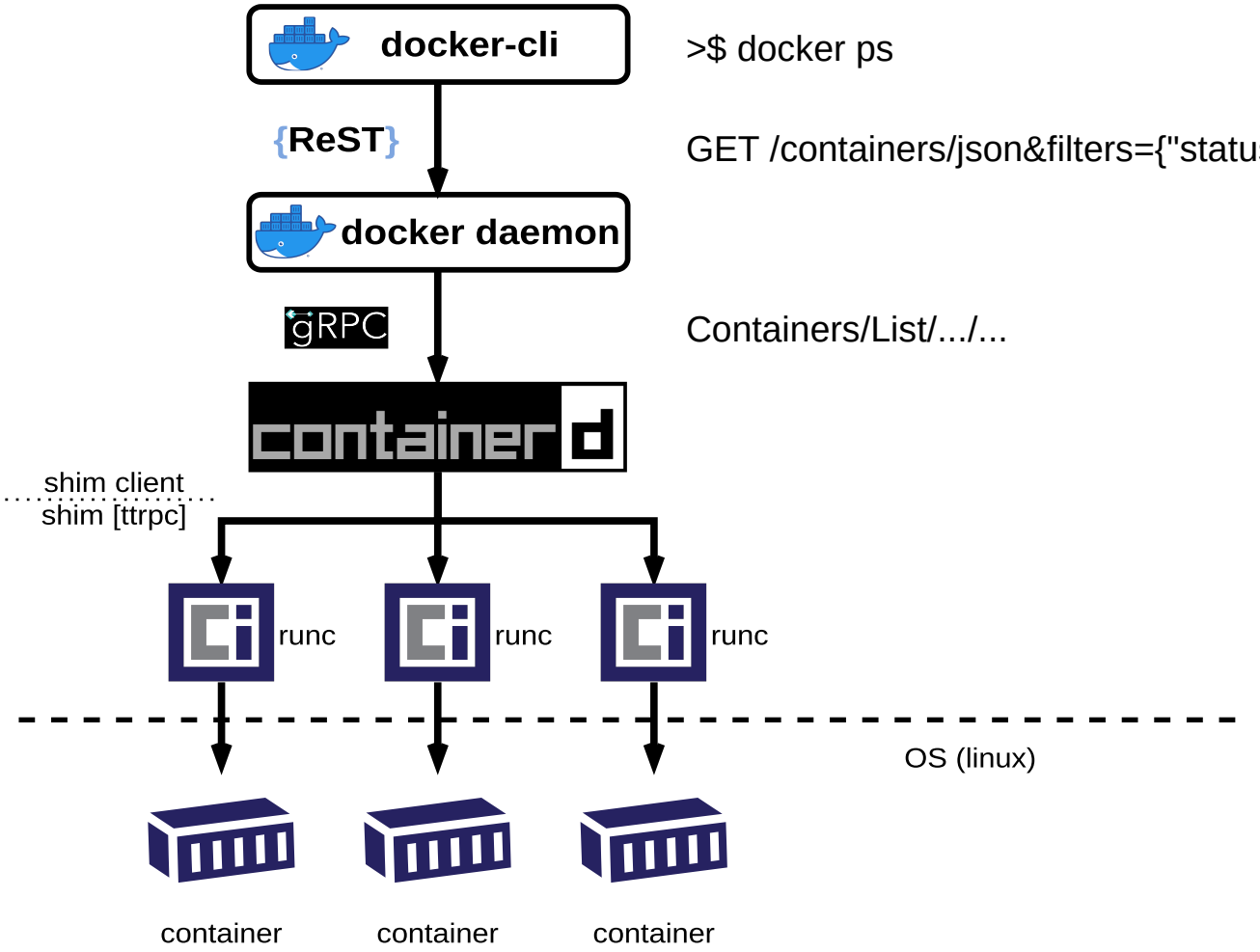
Specifically...



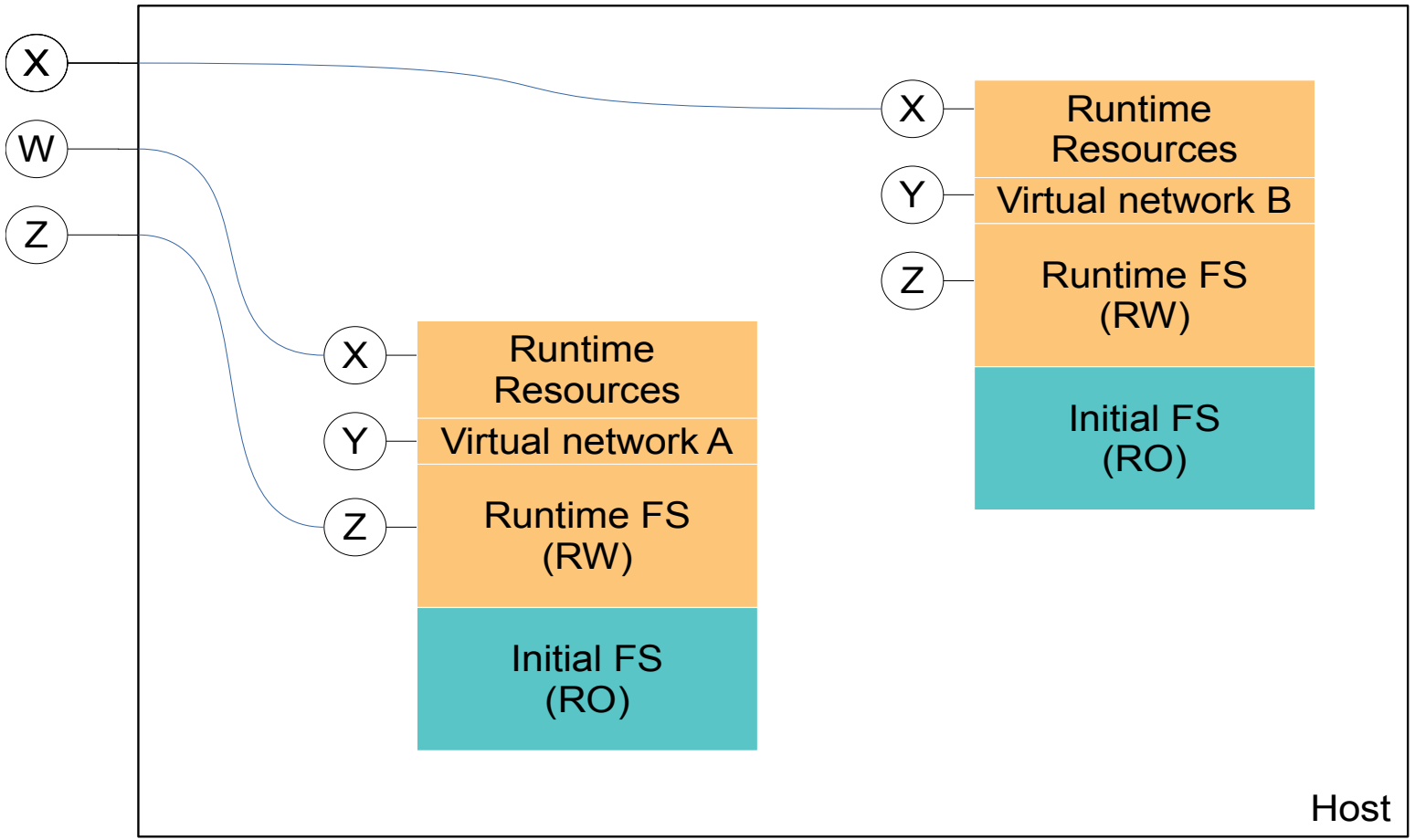
The Docker stack



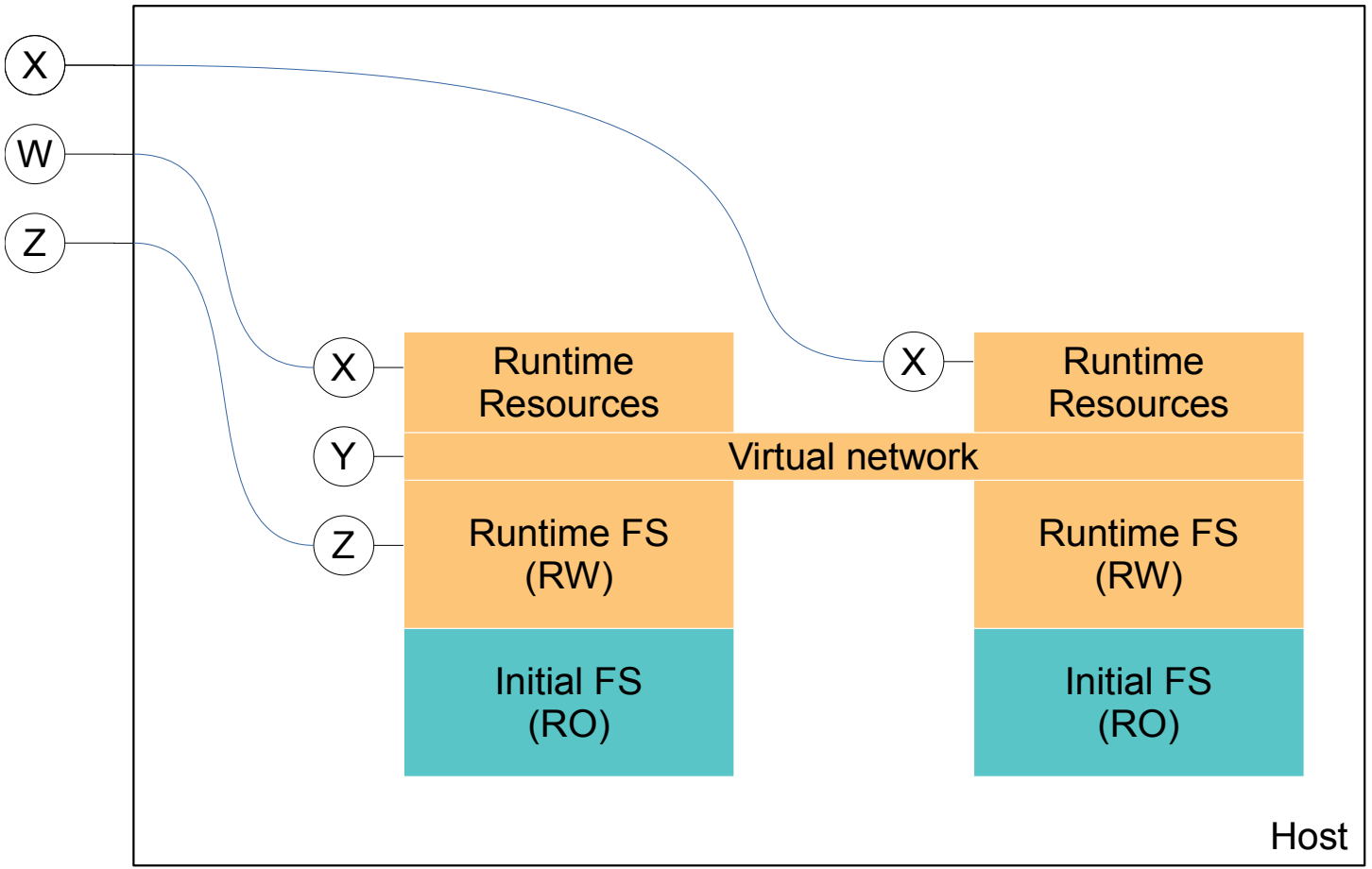
The Docker stack



Networking


X
W
Z
X
Runtime Resources
Virtual network A
Runtime FS (RW)
Initial FS (RO)
X
Y
Z
Runtime Resources
Virtual network B
Runtime FS (RW)
Initial FS (RO)
Host

Intra containers



Dockerfile

- Note: using docker and a dockerfile is one of the possible ways to create a container image compliant with the Open Container Initiative (OCI) image specification
- A dockerfile uses a DSL (the Dockerfile syntax) that is parsed by Docker (or other tools) to create container images

Dockerfile structure

```
#syntax=docker/dockerfile:1
FROM golang:1.21-alpine
WORKDIR /src
COPY . .
RUN go mod download
RUN go build -o /bin/client ./cmd/client
RUN go build -o /bin/server ./cmd/server
ENTRYPOINT [ "/bin/server" ]
```

Multi-stage builds

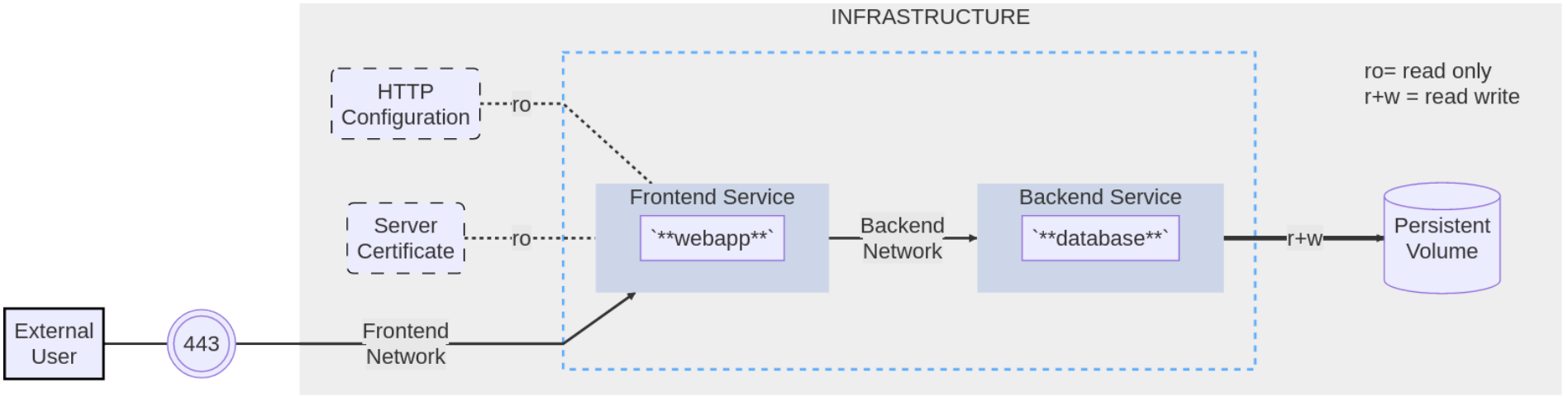
```
1  #syntax=docker/dockerfile:1
2  FROM golang:1.21-alpine as build
3  WORKDIR /src
4  COPY . .
5  RUN go mod download
6  RUN go build -o /bin/client ./cmd/client
7  RUN go build -o /bin/server ./cmd/server
8
9  FROM scratch
10 COPY --from=build /bin/client /bin/server /bin/
11 ENTRYPOINT [ "/bin/server" ]
```

Compose

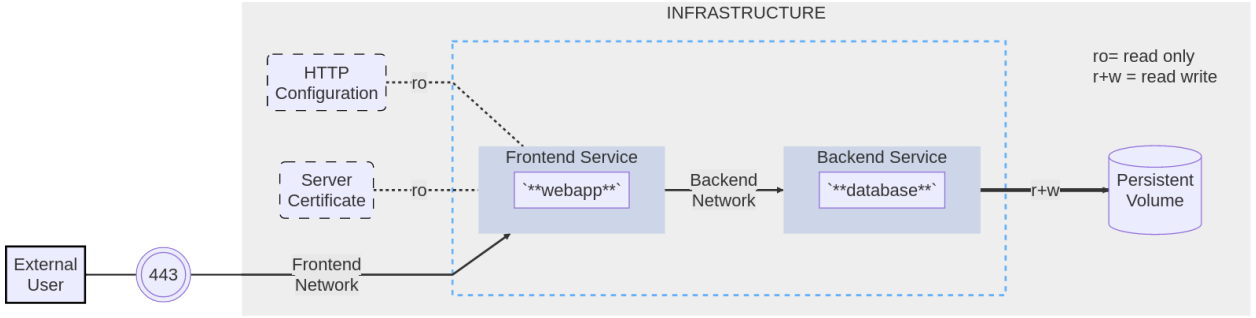
A standardized specification for container-based applications (Docker compose is the reference implementation).

It allows to decoratively describe how to set up and execute an application composed by multiple cooperating containers.

Compose example



Compose example




```yaml
services:
  frontend:
    image: awesome/webapp
    ports:
      - "443:8043"
    networks:
      - front-tier
      - back-tier
    configs:
      - httpd-config
    secrets:
      - server-certificate

  backend:
    image: awesome/database
    volumes:
      - db-data:/etc/data
    networks:
      - back-tier

volumes:
  db-data:
    driver: flocker
    driver_opts:
      size: "10GiB"

configs:
  httpd-config:
    external: true

secrets:
  server-certificate:
    external: true

networks:
  # The presence of these objects is sufficient to define them
  front-tier: {}
  back-tier: {}
```

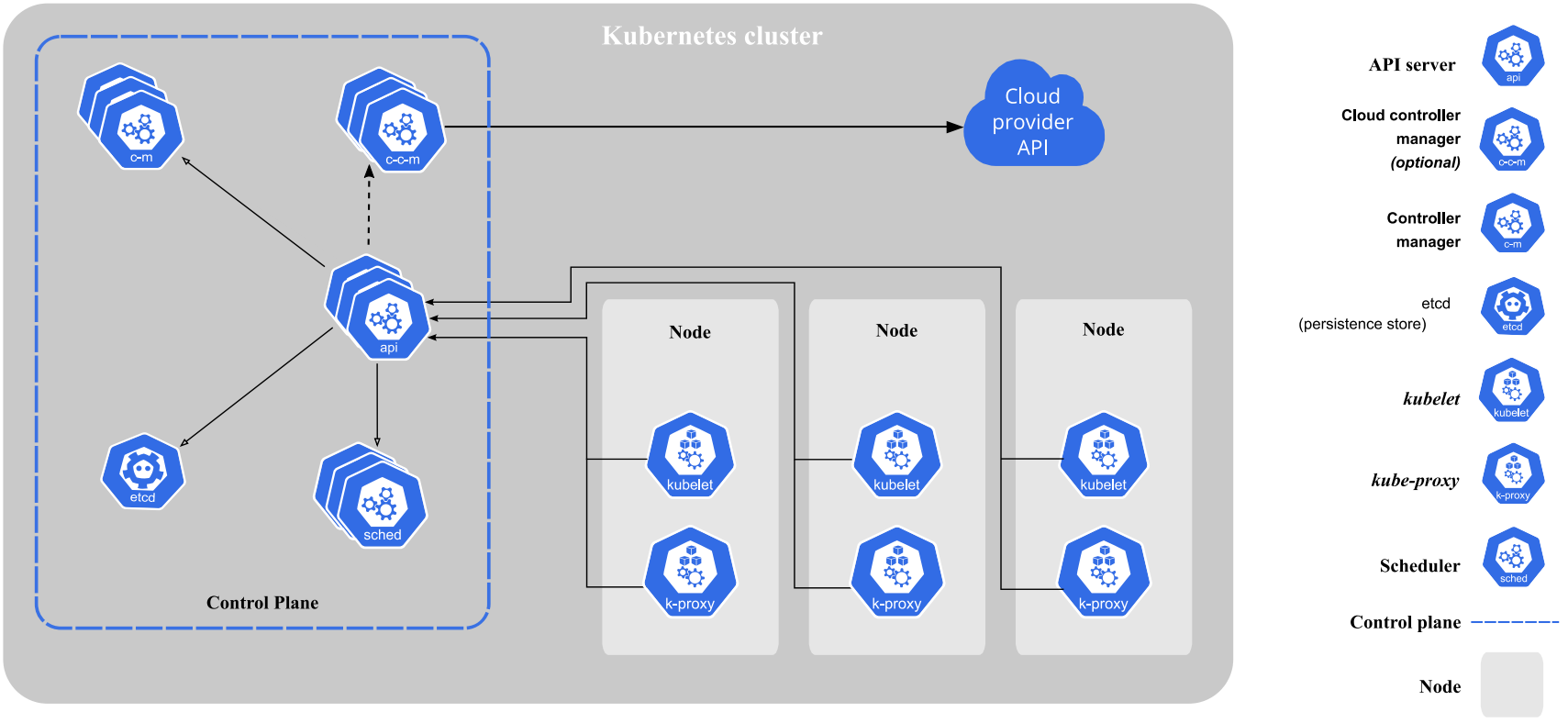
Kubernetes (K8s)

- Kubernetes allocates containers in multiple *nodes*

Kubernetes (K8s)

- Kubernetes allocates *containerized applications* in a *cluster* composed by multiple *nodes*
- Providing support for automating deployment, scaling, networking and management

K8s components



K8s

- Kubernetes manages the state of the system as a set of persistent *objects*
- Kind of objects in K8s:
  Pod, Deployment, ReplicaSet, StatefulSet, DaemonSet, PersistentVolume, Service, Ingress, Namespace, ConfigMap, Secret, Job

K8s general concepts

- Object management techniques
  - Imperative (commands/configurations)
  - Declarative configurations – a desired state is declared, K8s' machinery works to reach that state
- A simple matching mechanism between *labels* (key-value pairs) and *selectors* is used to associate objects

Pod objects

- Pods are ephemeral deployment units
- A pod is a group of one or more containers, with shared storage and network resources
- Pods are always co-located and co-scheduled
- *Think of them as a compose with **less** isolation*

Isolation objects

- Namespace
  - Isolate partition within a cluster

Workload objects

- ReplicaSet
  - Stable set of replica pods
- Deployment
  - Declarative updates for Pods and ReplicaSets
- StatefulSet
  - Manages replicas with identities

Workload objects

- DaemonSet
  - Manages pod to be allocated to nodes
- Job
  - Manages (retriable) pods executions

Storage objects

- PersistentVolume
  - Abstract storage resources (to be matched by corresponding `persistentVolumeClaims`)

Configuration objects

- ConfigMap
  - Key-value storage that can be consumed by pods
- Secret
  - Key-value storage intended to hold confidential data

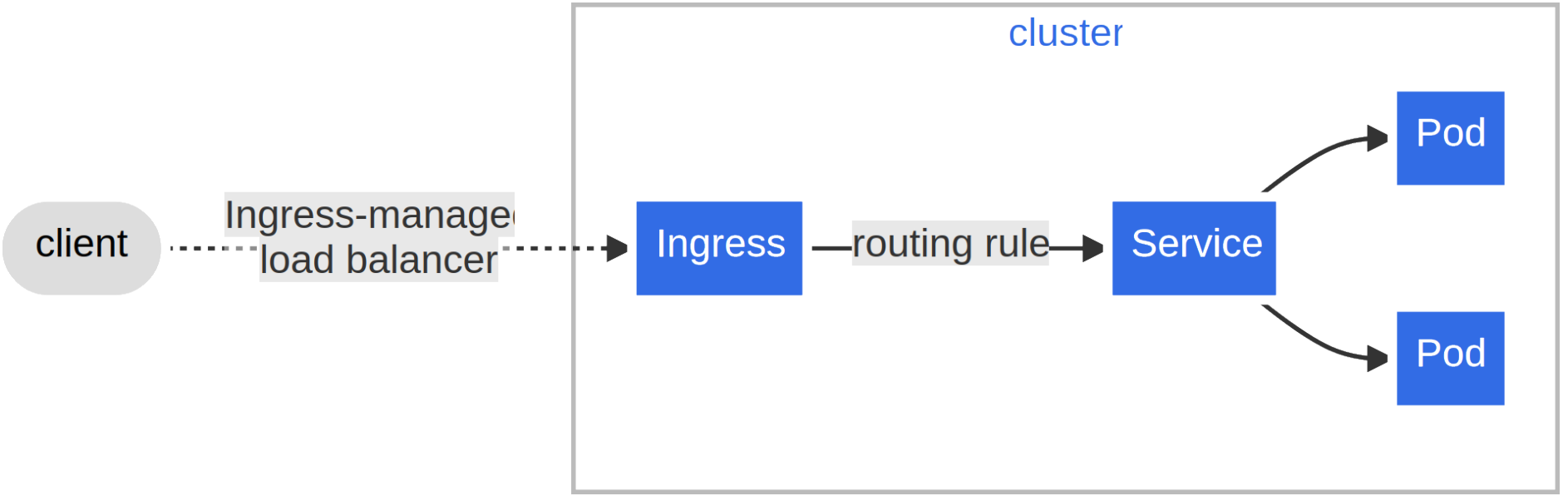
Access objects

- Service
  - Exposes a network application (running in pods in the cluster)
    Services are registered in the cluster’s internal DNS
- Ingress
  - Manages external access to services in a cluster

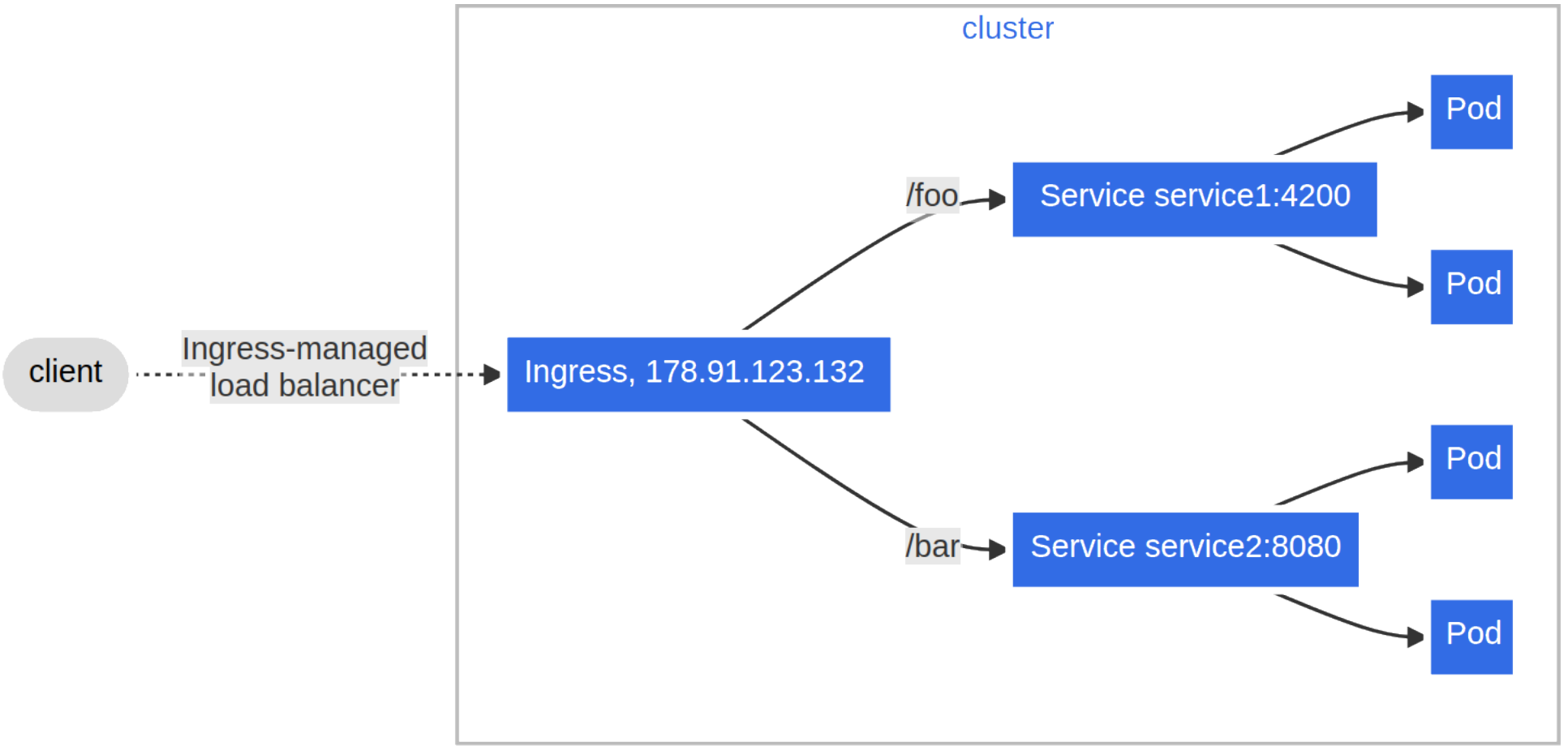
Service types

- ClusterIP
  - Allocates a cluster-wide internal virtual IP address on all nodes. Traffic to this IP is redirected to the pod’s endpoints via OS-provided mechanisms
- NodePort
  - Exposes the ClusterIP on all nodes (on the same port). Incoming traffic is subject to NAT
- LoadBalancer
  - Configures an external load balancer to access NodePorts

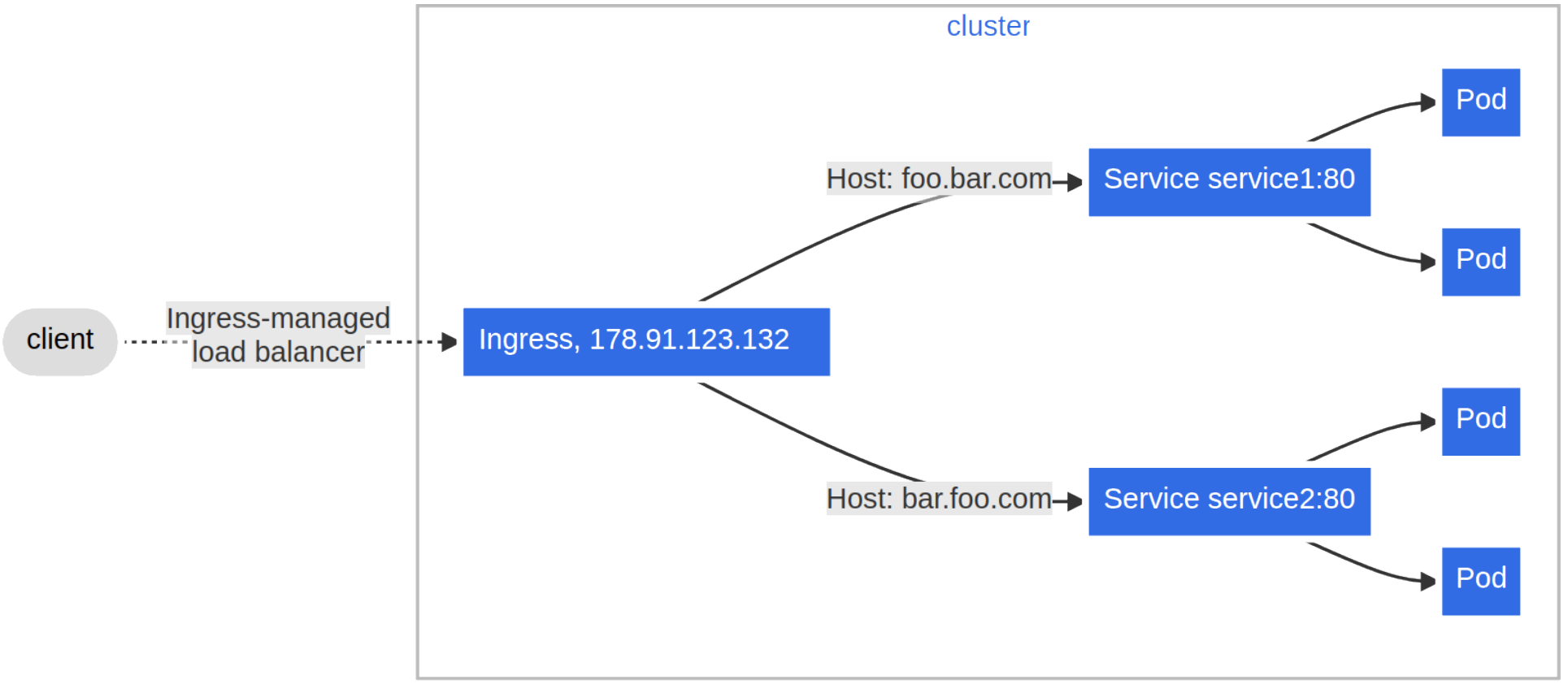
Ingress


cluster
client
Ingress-manage
load balancer
Ingress
routing rule
Service
Pod
Pod

Ingress


cluster
Pod
/foo
Service service1:4200
Pod
client
Ingress-managed
load balancer
Ingress, 178.91.123.132
Pod
/bar
Service service2:8080
Pod

Ingress


cluster
Pod
Host: foo.bar.com
Service service1:80
Pod
client
Ingress-managed
load balancer
Ingress, 178.91.123.132
Pod
Host: bar.foo.com
Service service2:80
Pod