Kubernetes and Service Mesh Workshop

Slides: https://slides.peterj.dev/jfuture-2019

Agenda

Workshop runs from 12:00 PM to 5:00 PM

- Lunch at 14:30 PM

Multiple sections - theory + exercises

- Introduction to Containers
- Docker and Kubernetes
- Istio Service Mesh
  - Traffic Routing
  - Resiliency
  - Security

Introduction

- I am Peter (@pjausovec)
- Software Engineer at Oracle
- Working on "cloud-native" stuff
- Books:
  - Cloud Native: Using Containers, Functions, and Data to Build Next-Gen Apps
  - SharePoint Development
  - VSTO For Dummies
- Courses:
  - Kubernetes Course (https://startkubernetes.com)
  - Istio Service Mesh Course (https://learnistio.com)



Source: https://www.sourcetoday.com/viewpoint/3-data-center-maintenance-tips-mitigate-security-threats



By helpameout - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=20337780

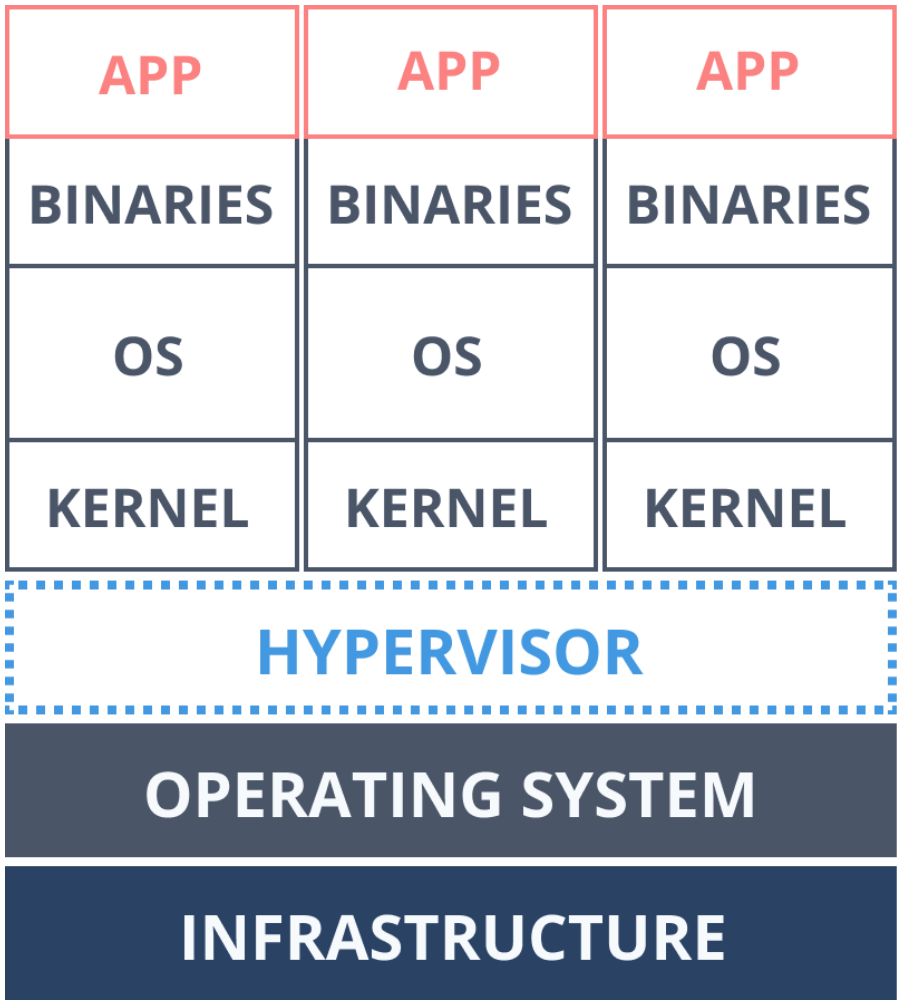
Virtualization

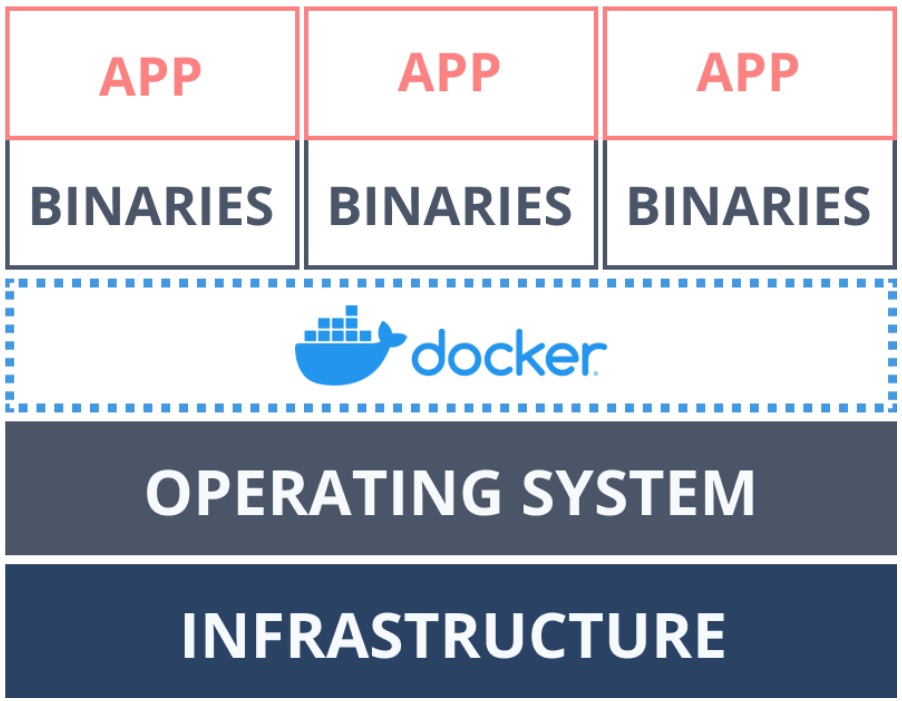
- 2006: VMWare Server
- Run multiple OS on the same host
- Expensive: multiple kernels, OS ...


msc
tex
CAPITAL

Docker

- First public release in 2013
- Containers existed in Linux for >10 years
- Slice the OS to securely run multiple applications
  - Namespaces, cGroups

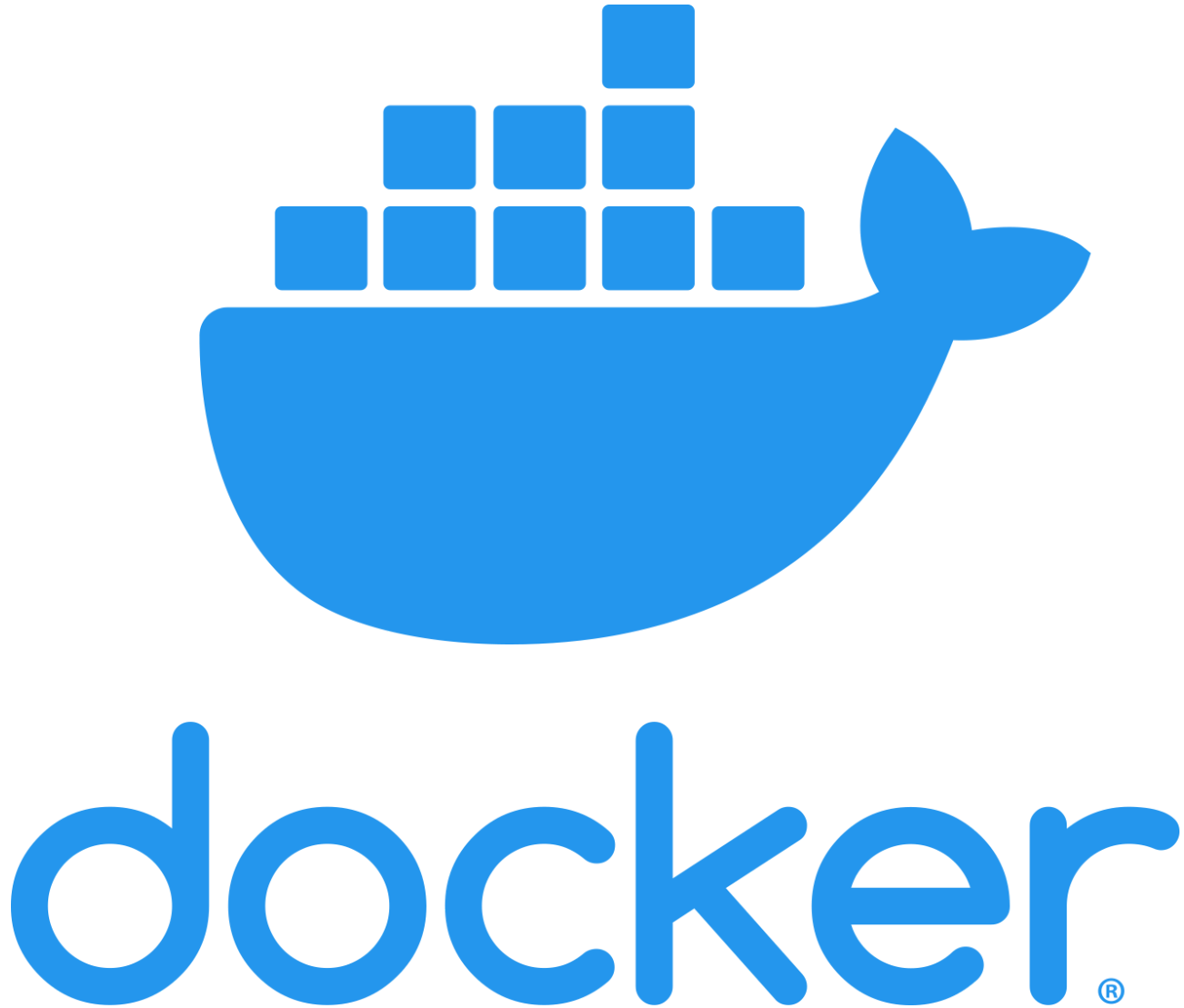

APP
APP
APP
BINARIES
BINARIES
BINARIES
OS
OS
OS
KERNEL
KERNEL
KERNEL
HYPERVISOR
OPERATING SYSTEM
INFRASTRUCTURE


APP
APP
APP
BINARIES
BINARIES
BINARIES
docker
OPERATING SYSTEM
INFRASTRUCTURE



IT WORKS

on my machine


docker

What is Docker?

Docker Engine (daemon) + CLI

Dockerfile

```
1 FROM ubuntu:18.04
2 WORKDIR /app
3 COPY hello.sh /app
4 RUN chmod +x hello.sh
5 RUN apt-get update
6 RUN apt-get install curl -y
7 CMD ["./hello.sh"]
```

Docker image

- Collection of layers from `Dockerfile` (one layer per command)
- Layers are stacked on top of each other
- Each layer is a delta from the layer before it
- All layers are read-only

Docker image



Image names

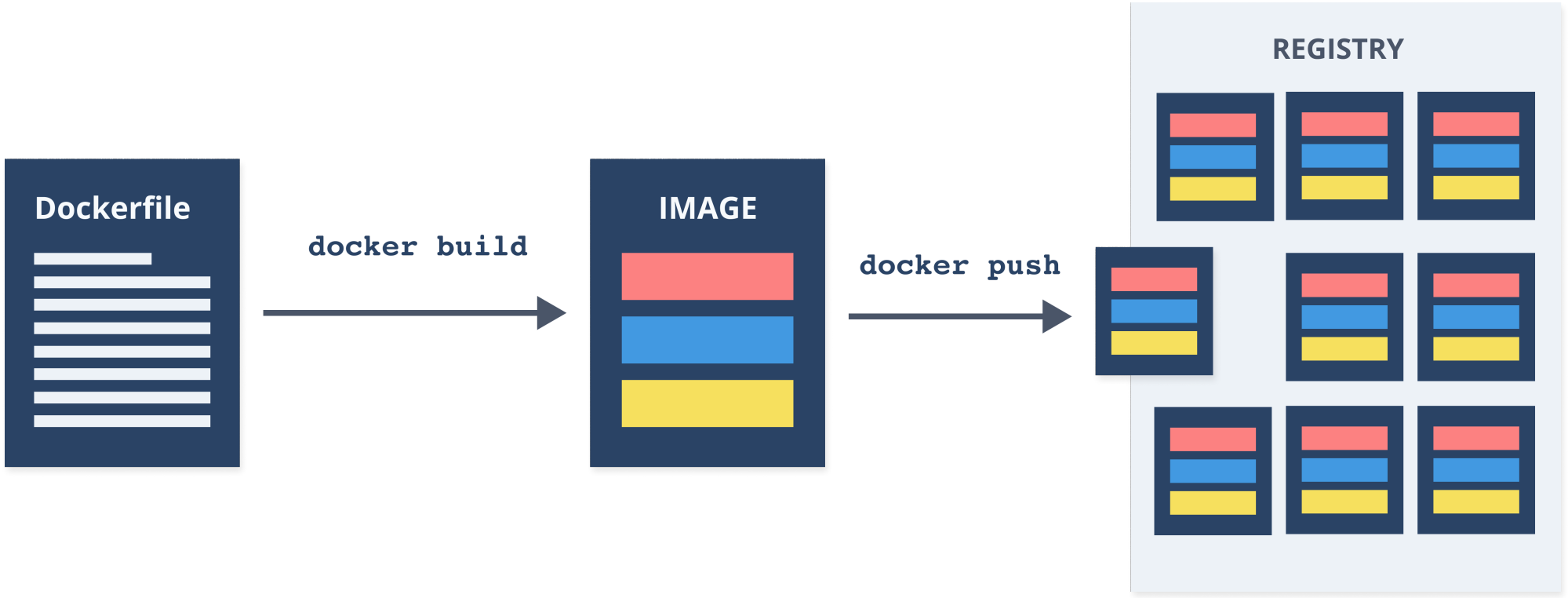
- Image = repository + image name + tag

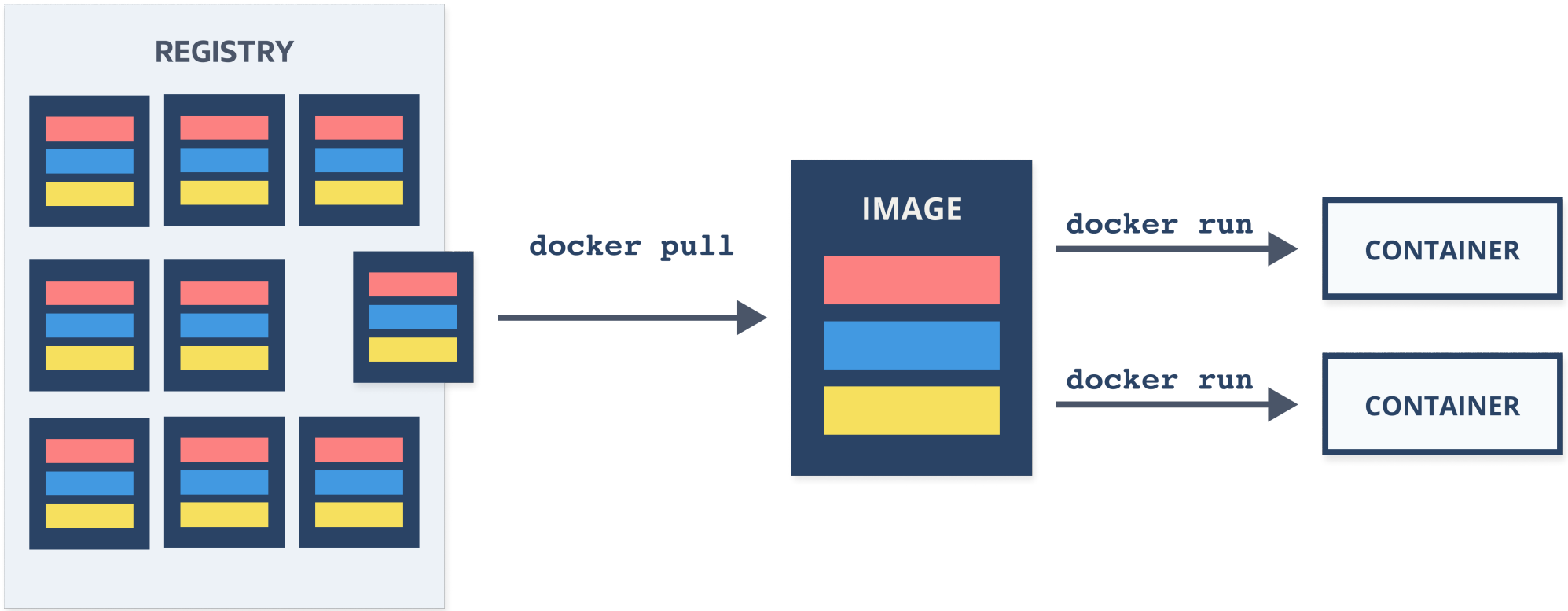
```
mycompany/hello-world:1.0.1
```

- All images get a default tag called *latest*
- Tag = version or variant of an image

Docker Registry

- Place to store your Docker images
  - Public and private repositories
  - Docker Hub (https://hub.docker.com)
  - Every cloud provider has its own
- You can also store images locally, on your Docker host


Dockerfile
docker build
IMAGE
docker push
REGISTRY


REGISTRY
docker pull
IMAGE
docker run
CONTAINER
docker run
CONTAINER



Container Orchestration

- Provision and deploy containers onto nodes
- Resource management/scheduling containers
- Health monitoring
- Scaling
- Connect to networking
- Internal load balancing

Kubernetes Overview

- Most popular choice for cluster management and scheduling container-centric workloads
- Open source project for running and managing containers
- K8S = KuberneteS

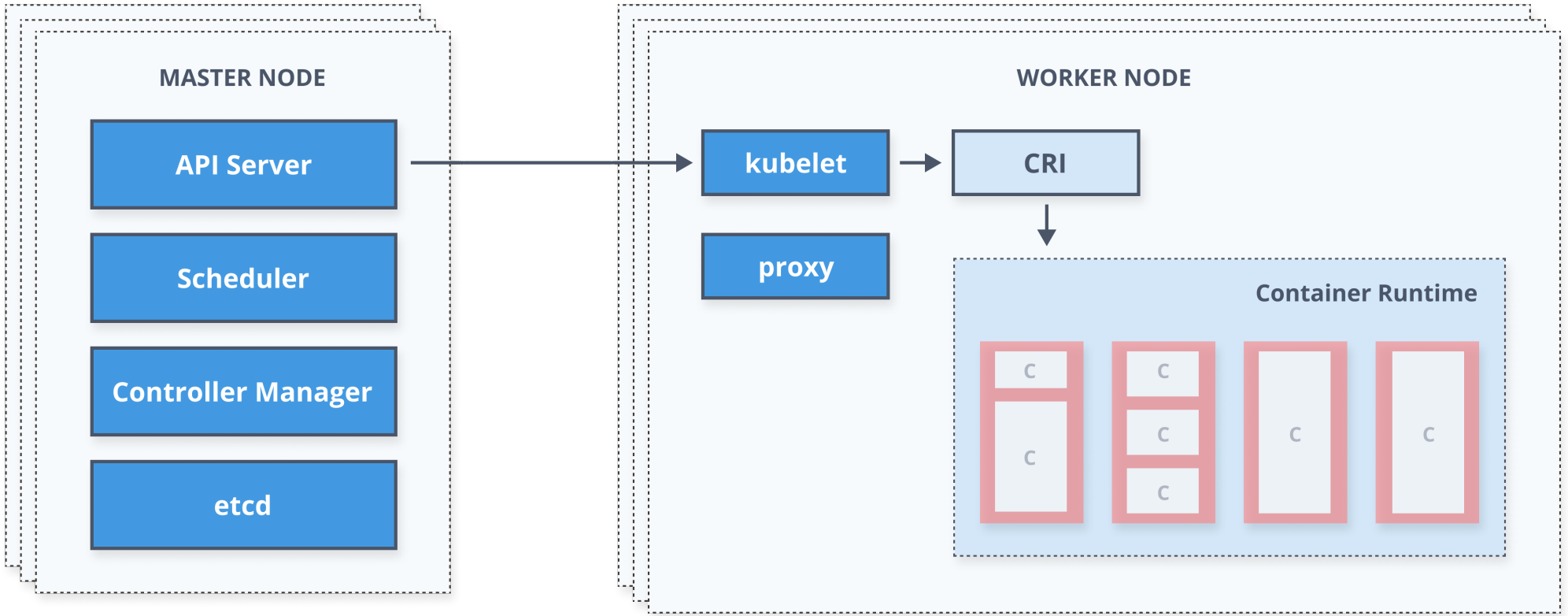
Definitions

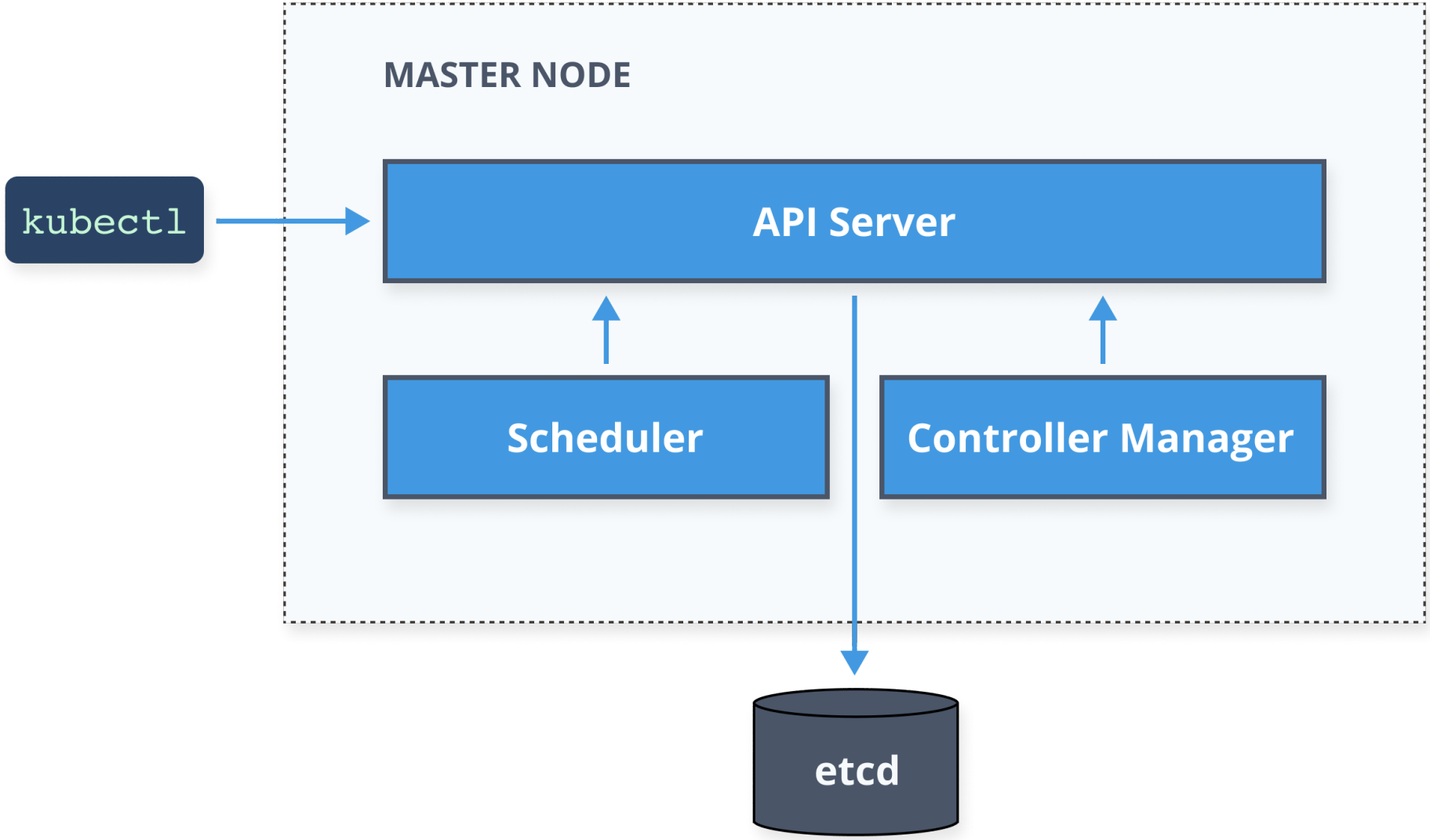
Portable, extensible, open-source platform for managing containerized workloads and services

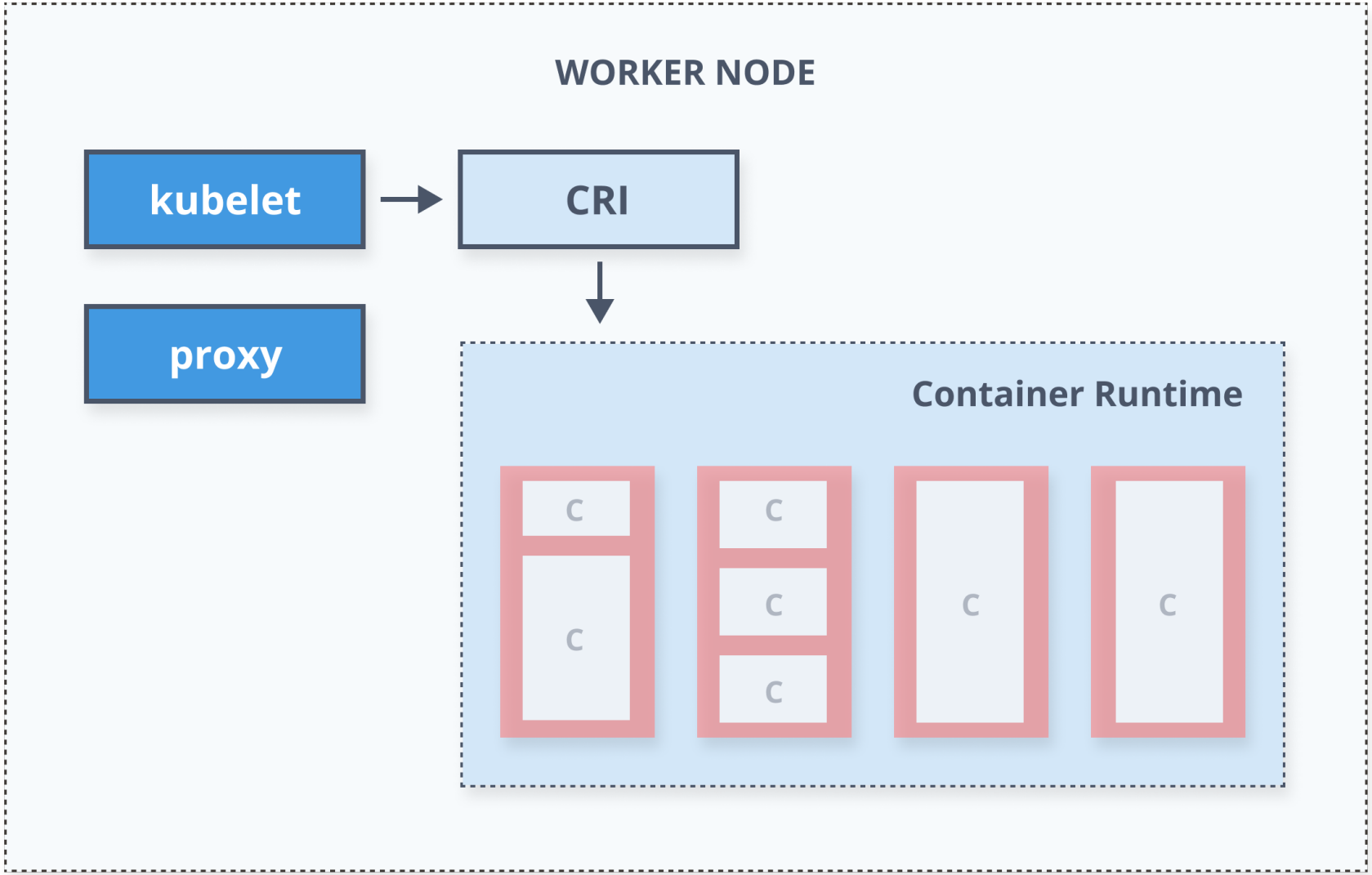
Container-orchestration system for automating application deployment, scaling, and management

https://kubernetes.io

Kubernetes Architecture

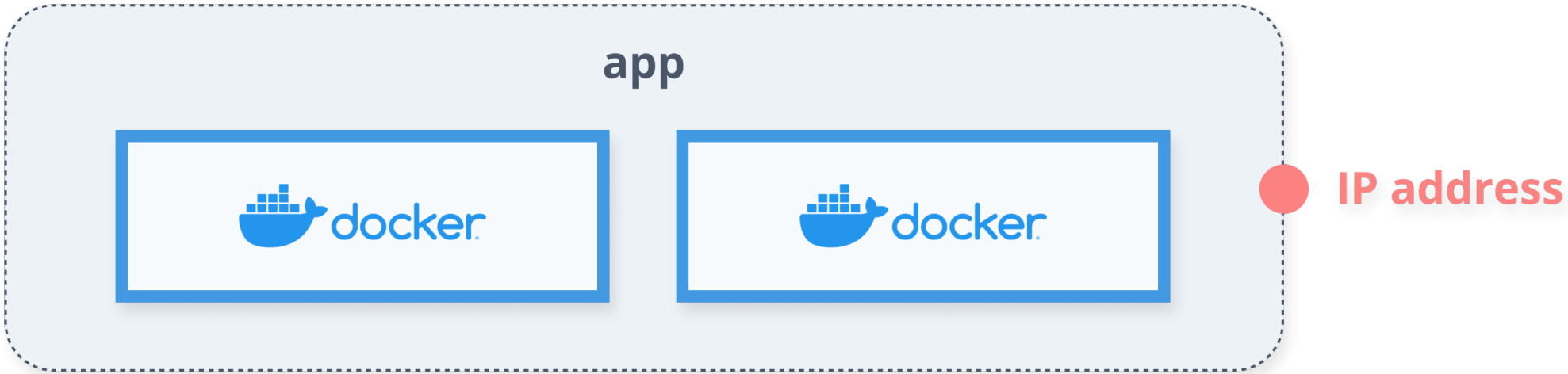



MASTER NODE
kubectl
API Server
Scheduler
Controller Manager
etcd


WORKER NODE
kubelet
CRI
proxy
Container Runtime
C
C
C
C
C
C
C
C

Kubernetes Resources

- Multiple resources defined in the Kubernetes API
  - namespaces
  - pods
  - services
  - secrets
  - ...
- Custom resources as well!
  - CRD (Custom Resource Definition)


app
docker
docker
IP address

Pods

- Smallest deployable/manageable/scalable unit
- Logic grouping of containers
  - All running on the same node
  - Share namespace, network, and volumes
- Has a unique IP
- Controlled by a ReplicaSet

Pods

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: myapp-pod
  labels:
    app: myapp
spec:
  containers:
  - name: myapp-container
    image: busybox
    command: ['sh', '-c', 'echo Hello Minsk! && sleep 3600']
```

ReplicaSet

- Ensures specified number of pod replicas is running
  - Creates and deletes pods as needed
- Selector + Pod template + Replica count

```yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: myapp
  labels:
    app: myapp
spec:
  replicas: 5
  selector:
    matchLabels:
      app: myapp
  template:
    metadata:
      labels:
        app: myapp
    spec:
      containers:
      - name: myapp-container
        image: busybox
        command: ['sh', '-c', 'echo Hello Minsk! && sleep 3600']
```

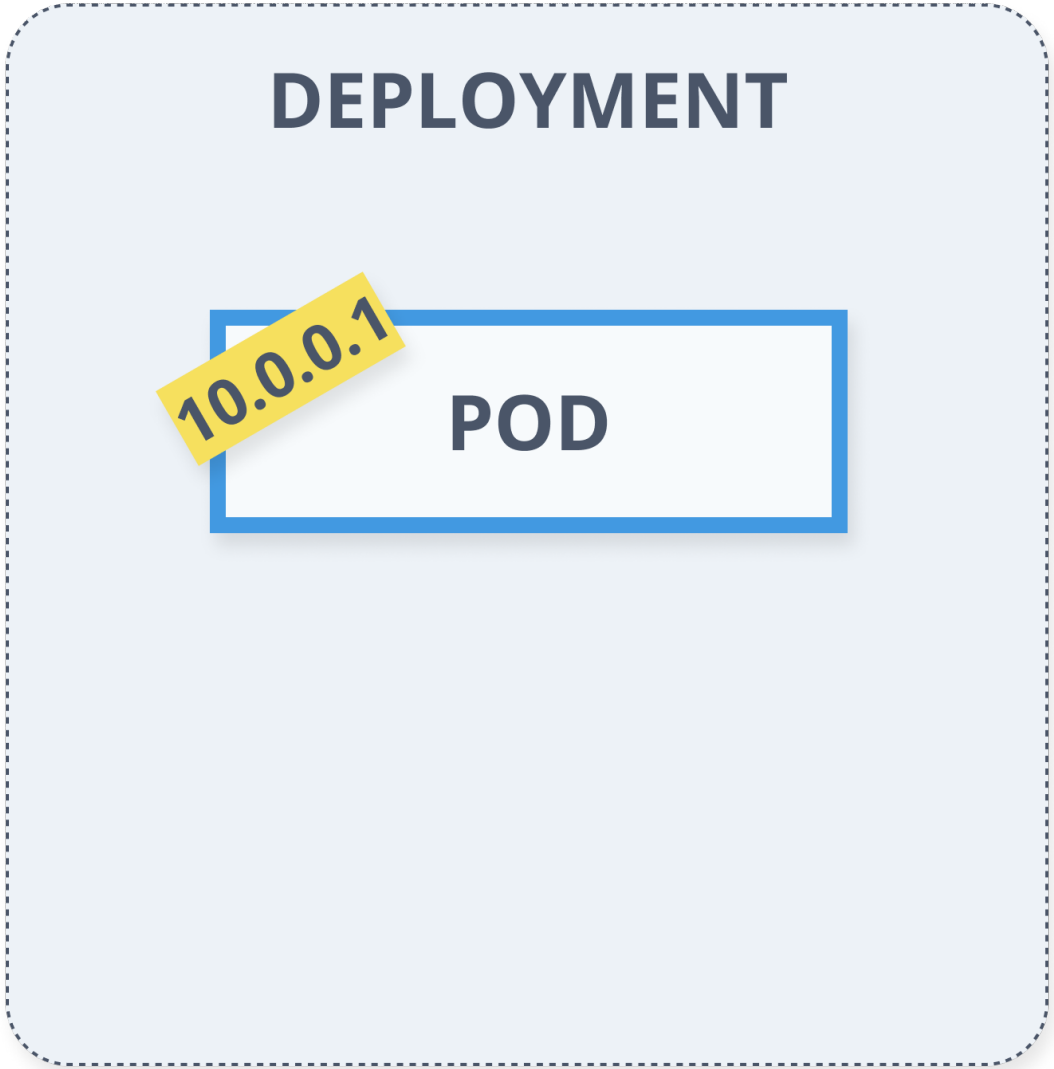
Demo

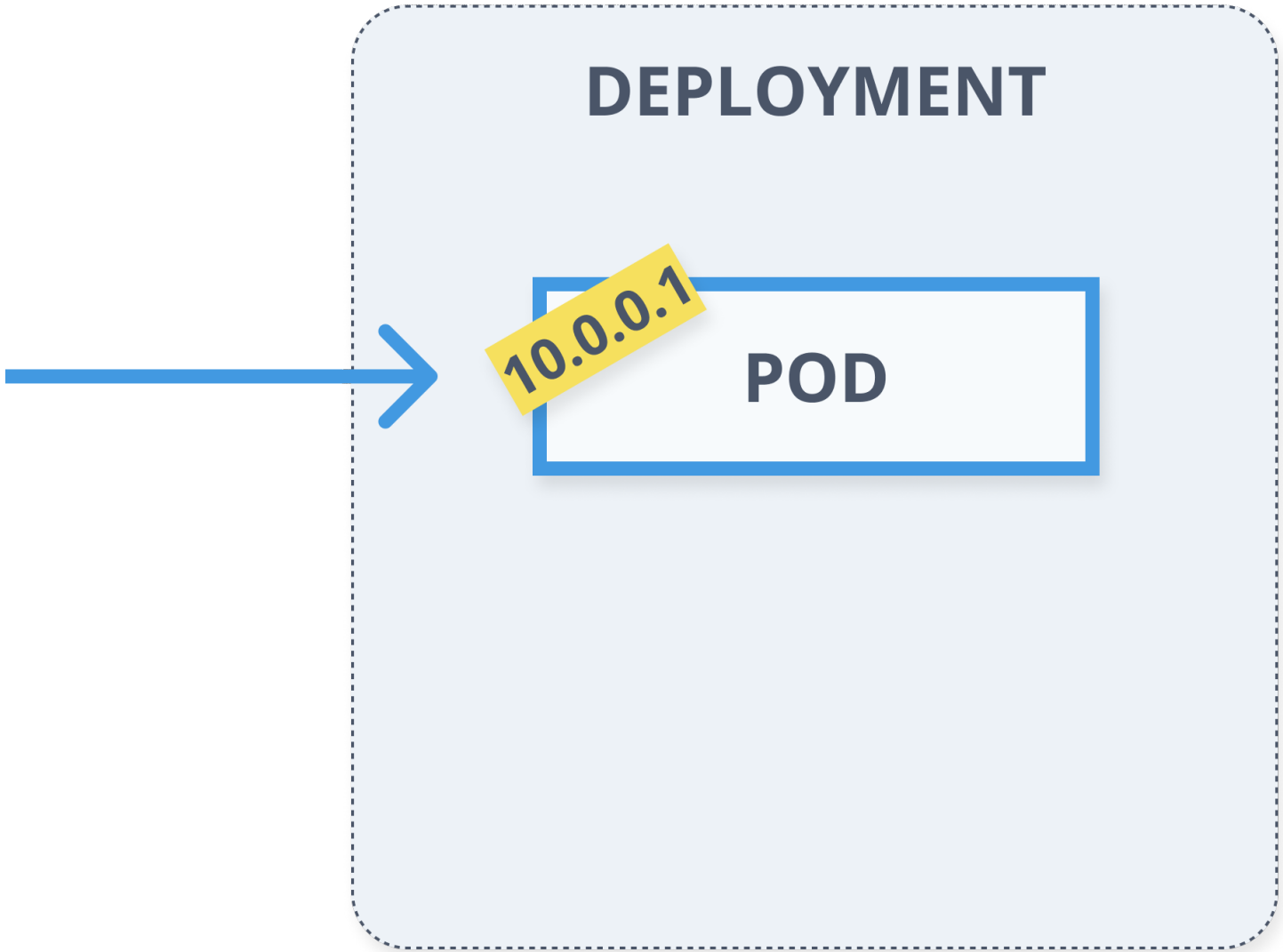
Pods and ReplicaSets

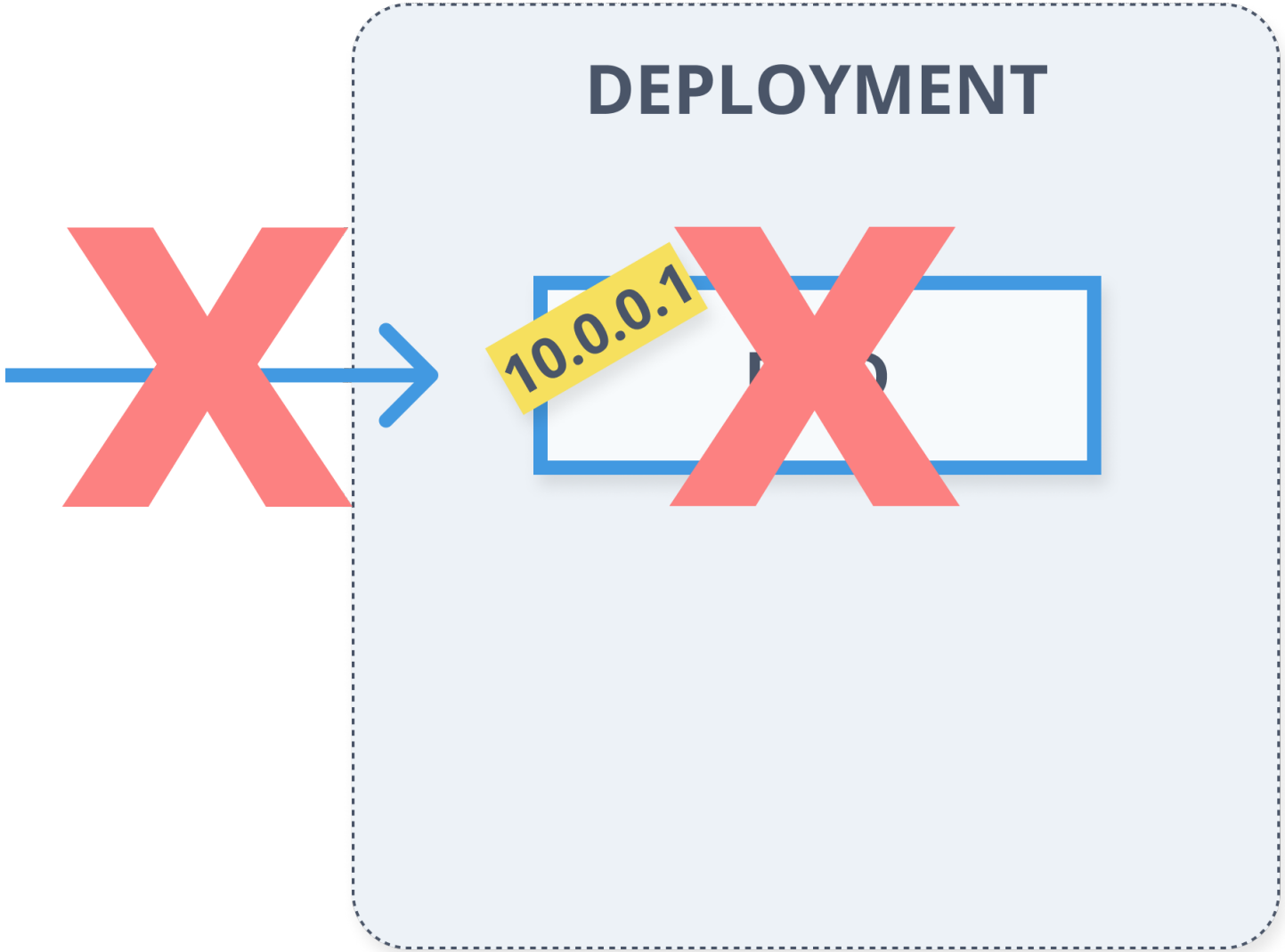
Deployments

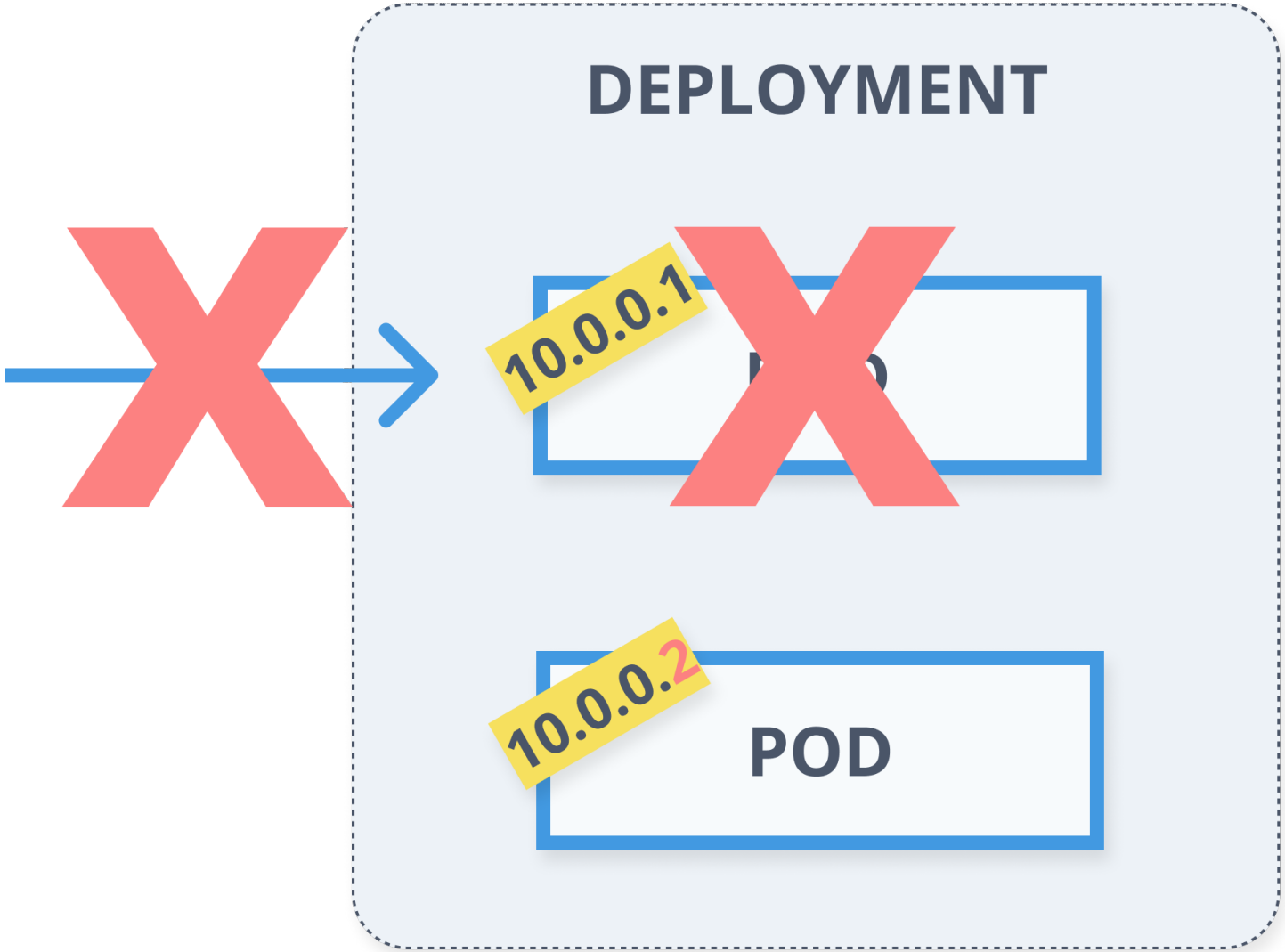
- Describes desired state
- Manages updates
  - Controlled roll-out from actual state to the desired state

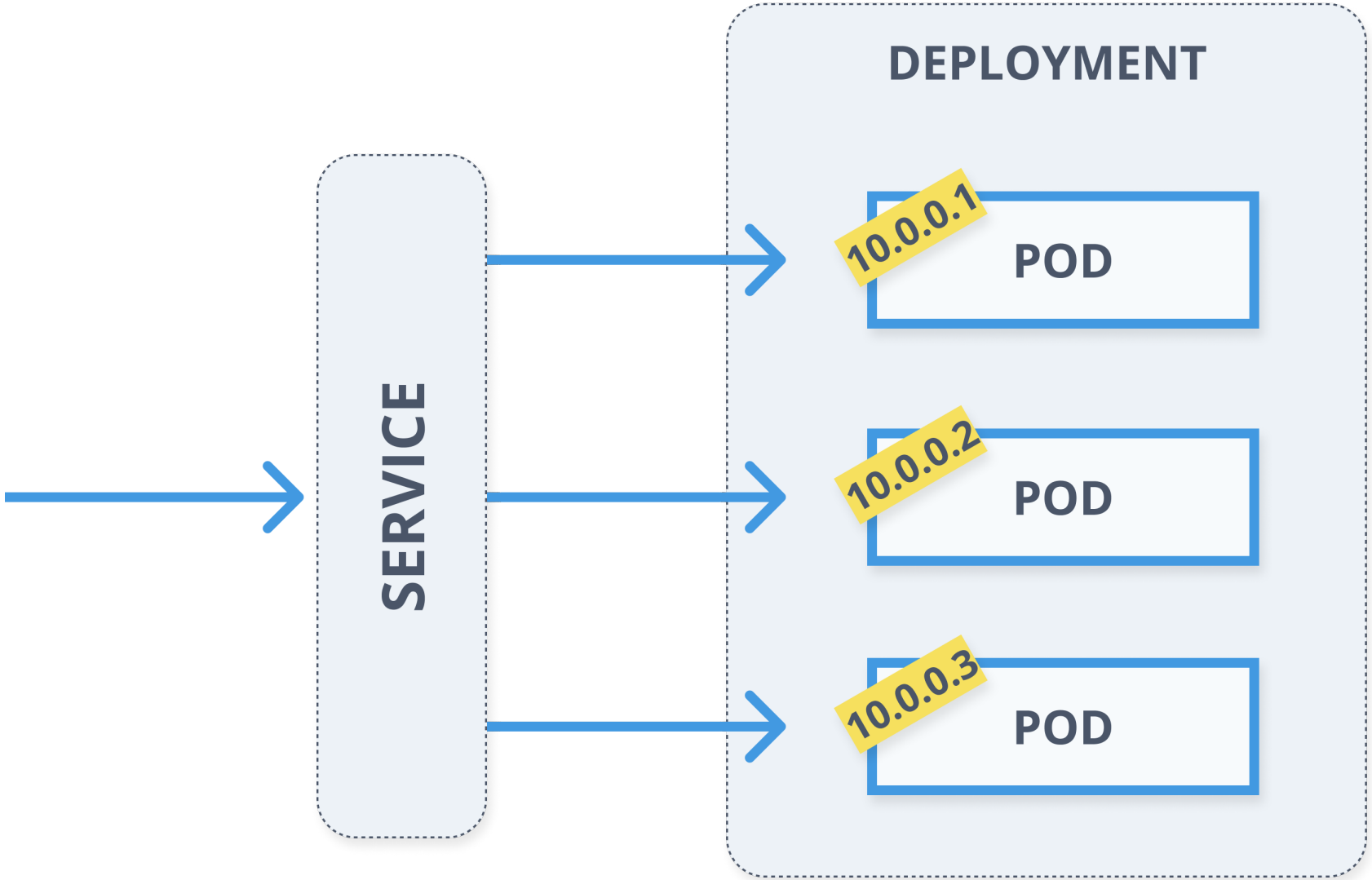
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: myapp
  labels:
    app: myapp
spec:
  replicas: 5
  selector:
    matchLabels:
      app: myapp
  template:
    metadata:
      labels:
        app: myapp
    spec:
      containers:
        - name: myapp-container
          image: busybox
          command: ['sh', '-c', 'echo Hello Minsk! && sleep 3600']
```


DEPLOYMENT
10.0.0.1
POD


DEPLOYMENT
10.0.0.1
POD


DEPLOYMENT
10.0.0.1


DEPLOYMENT
10.0.0.1
10.0.0.2
POD


DEPLOYMENT
SERVICE
10.0.0.1
POD
10.0.0.2
POD
10.0.0.3
POD

Services

- Define a logical set of pods
  - Pods are determined using labels (selector)
- Reliable, fixed IP address
- Automatic DNS entries
  - E.g. `hello-web.default`

```yaml
kind: Service
apiVersion: v1
metadata:
  name: myapp
  labels:
    app: myapp
spec:
  selector:
    app: myapp
  ports:
    - port: 80
      name: http
      targetPort: 3000
```

Services

ClusterIP

- Service is exposed on a cluster-internal iP (default)

NodePort

- Uses the same static port on each cluster node to expose the service

LoadBalancer

- Uses cloud providers' load balancer to expose the service

ExternalName

- Maps the service to a DNS name

Ingress

- Exposes HTTP/HTTPS routes from outside the cluster to services within the cluster
- Ingress controller uses a load balancer to handle the traffic (based on the ingress rules)
- Fanout and name based virtual hosting support:
  - blog.example.com -> blog-service.default
  - chat.example.com -> chat-service.default
  -

```yaml
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: ingress-example
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  rules:
  - host: blog.example.com
    http:
      paths:
      - path: /
        backend:
          serviceName: blog-service
          servicePort: 3000
      - path: /api
        backend:
          serviceName: blog-api
          servicePort: 8080
```

Config Maps

- Stores configuration values (key-value pairs)
- Values consumed in pods as:
  - environment variables
  - files
- Helps separating app code from configuration
- Needs to exist before they are consumed by pods (unless marked as optional)
- Need to be in the same namespace as pods

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: hello-kube-config
  namespace: default
data:
  count: 1
  hello: world
```

Secrets

- For storing and managing small amount of sensitive data (passwords, tokens, keys)
- Referenced as files in a volume, mounted from a secret
- Base64 encoded
- Types: generic, Docker registry, TLS

Namespaces

- Provides unique scope for resources
  - `my-namespace.my-service`
  - `another-namespace.my-service`
- (Most) Kubernetes resources live inside a namespace
- Can't be nested

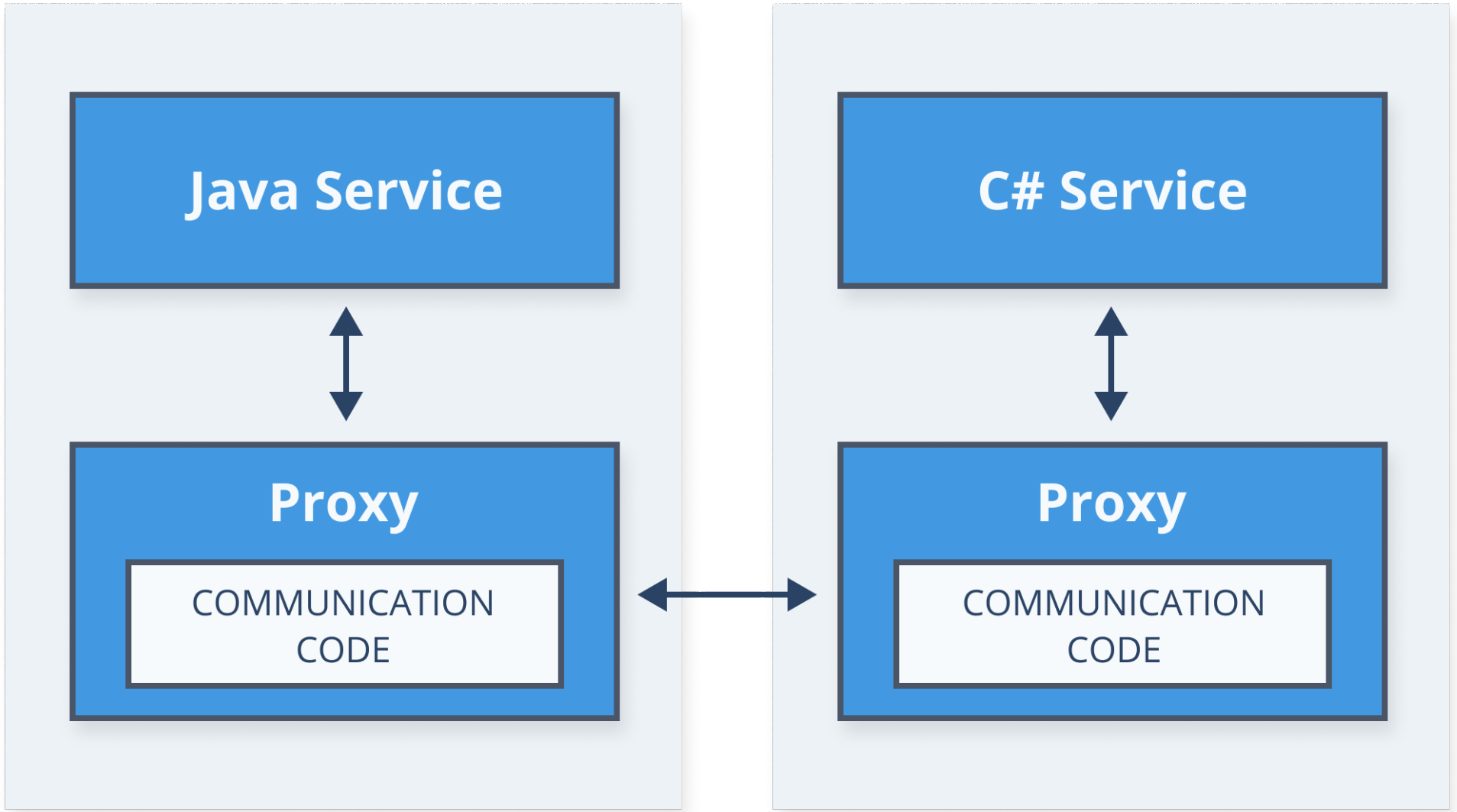
Exercises - Kubernetes

`./kubernetes`

https://github.com/peterj/jfuture

Service Mesh

Dedicated infrastructure layer to connect, manage, and secure workloads by managing the communication between them


Java Service
C# Service
Proxy
COMMUNICATION CODE
Proxy
COMMUNICATION CODE

Service Mesh - Architecture

Data plane (proxies)

- Run next to each service instance (or one per host)
  - Istio uses Envoy proxy
- Intercept all incoming/outgoing requests (`iptables`)
- Configure on how to handle traffic
- Emits metric

Control plane

- Validates rules
- Translates high-level rules to proxy configuration
- Updates the proxies/configuration
- Collects metrics from proxies

Service Mesh - Features

Connect

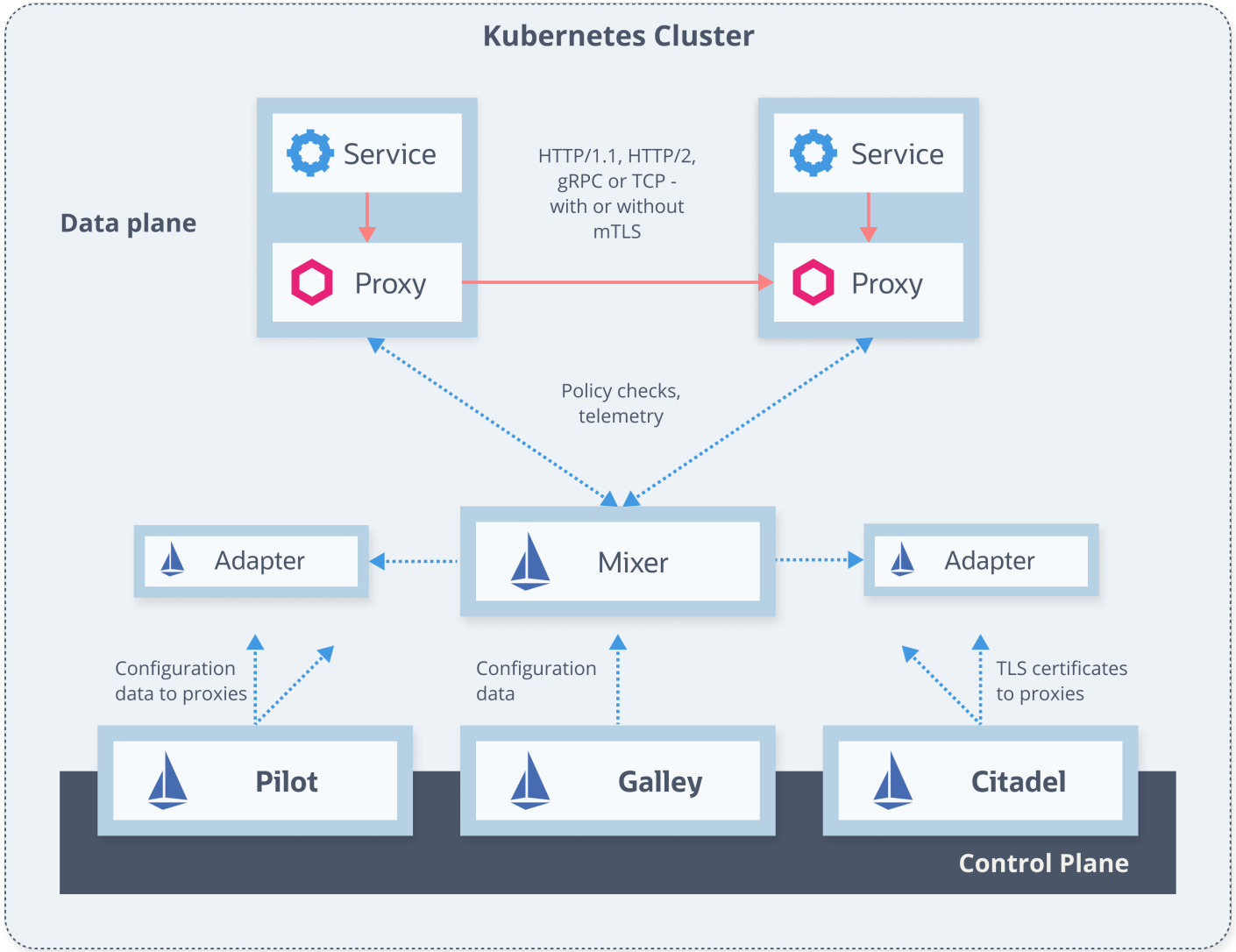
- Layer 7 routing and traffic management
  - %-based traffic split (URIs, header, scheme, method, ...)
  - Circuit breakers, timeouts and retries

Manage

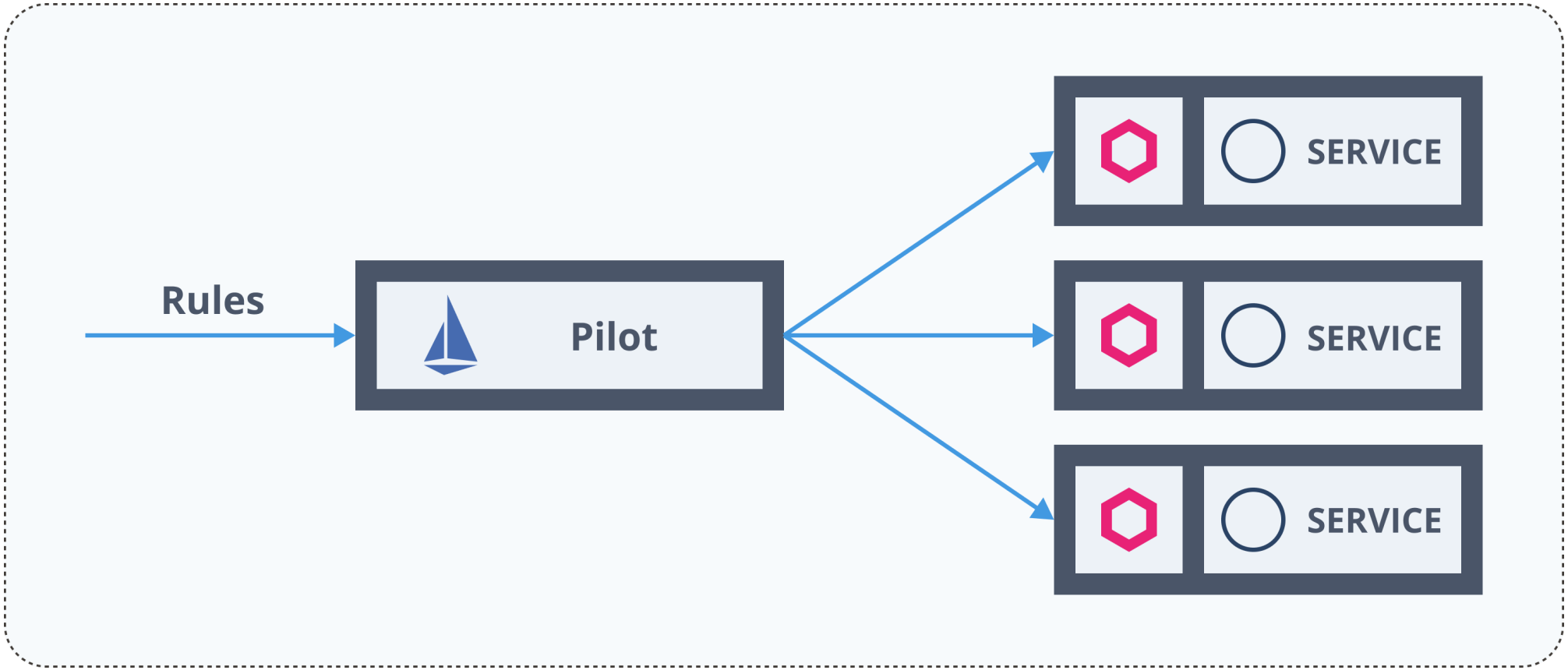
- Telemetry (proxies collect metrics automatically)
- Visibility into service communication without code changes

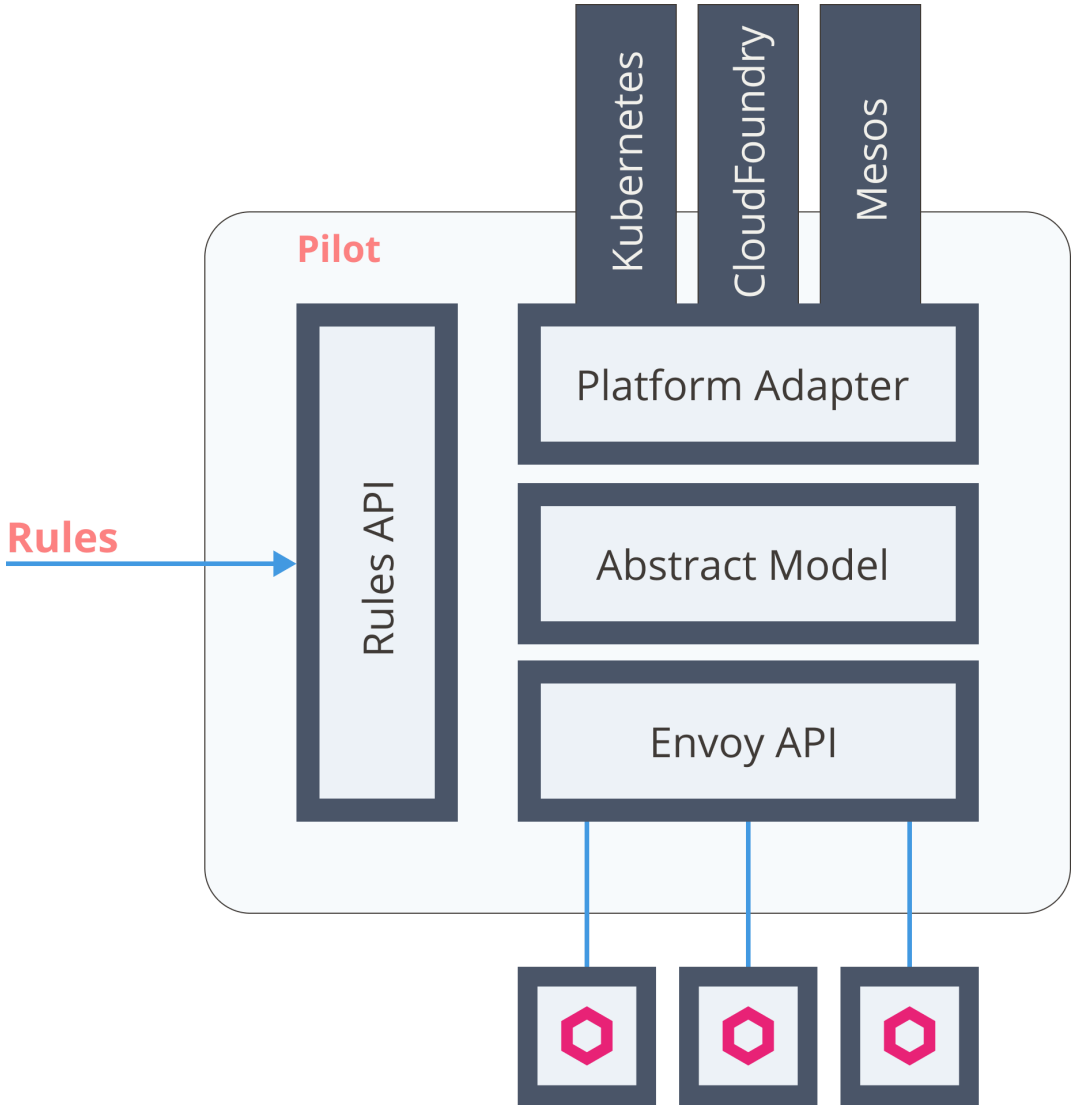
Secure

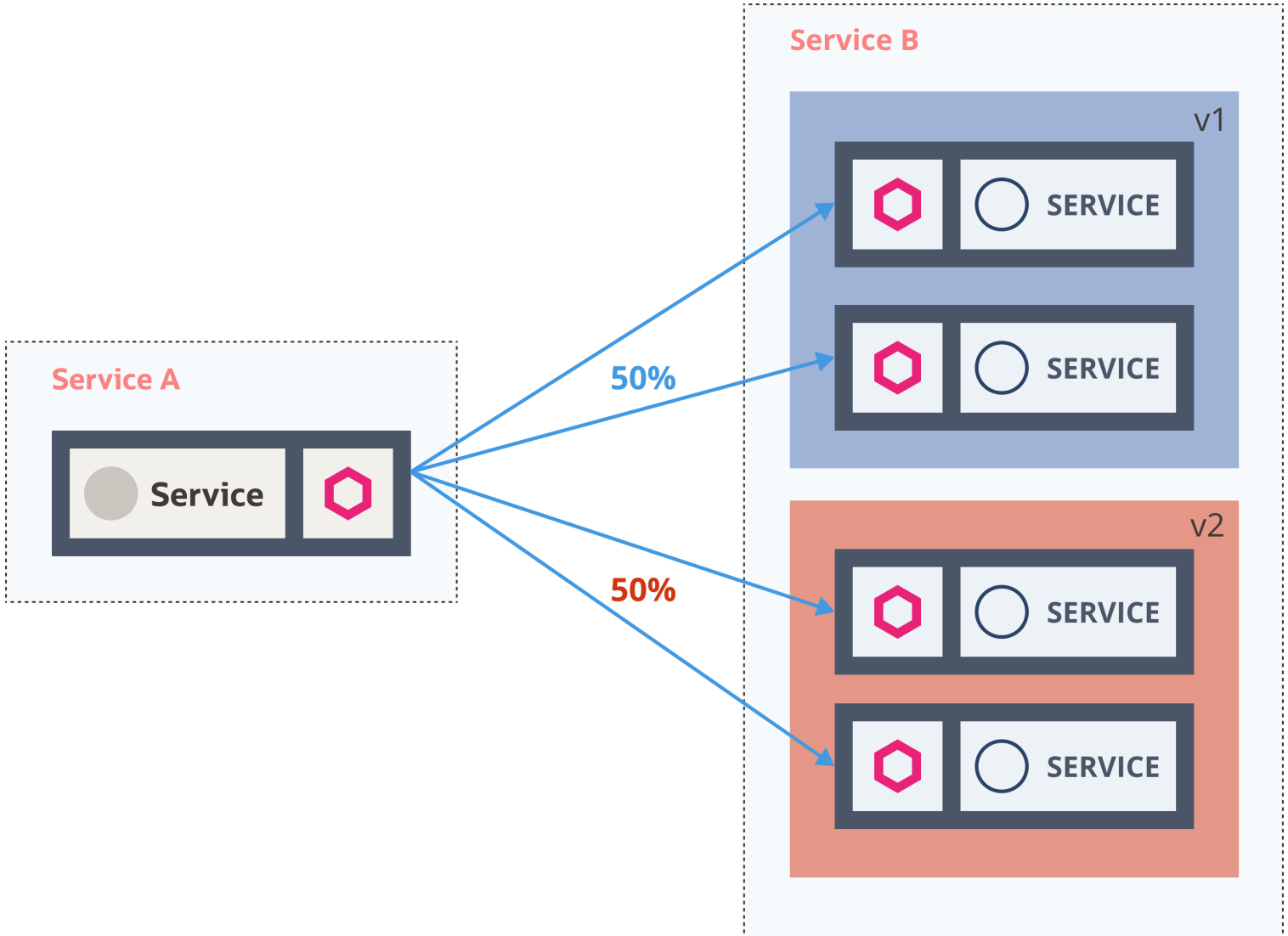
- Secure communication between services (mutual TLS)
- Identity + cert for each service


Kubernetes Cluster
Data plane
Service
Proxy
HTTP/1.1, HTTP/2,
gRPC or TCP -
with or without
mTLS
Service
Proxy
Policy checks,
telemetry
Adapter
Mixer
Adapter
Configuration
data to proxies
Configuration
data
TLS certificates
to proxies
Pilot
Galley
Citadel
Control Plane

Traffic Management


Rules
Pilot
SERVICE
SERVICE
SERVICE


Kubernetes
CloudFoundry
Mesos
Pilot
Platform Adapter
Rules API
Rules
Abstract Model
Envoy API


Service B
v1
SERVICE
SERVICE
Service A
50%
Service
50%
v2
SERVICE
SERVICE

Service Mesh - Istio

Traffic Management Resources

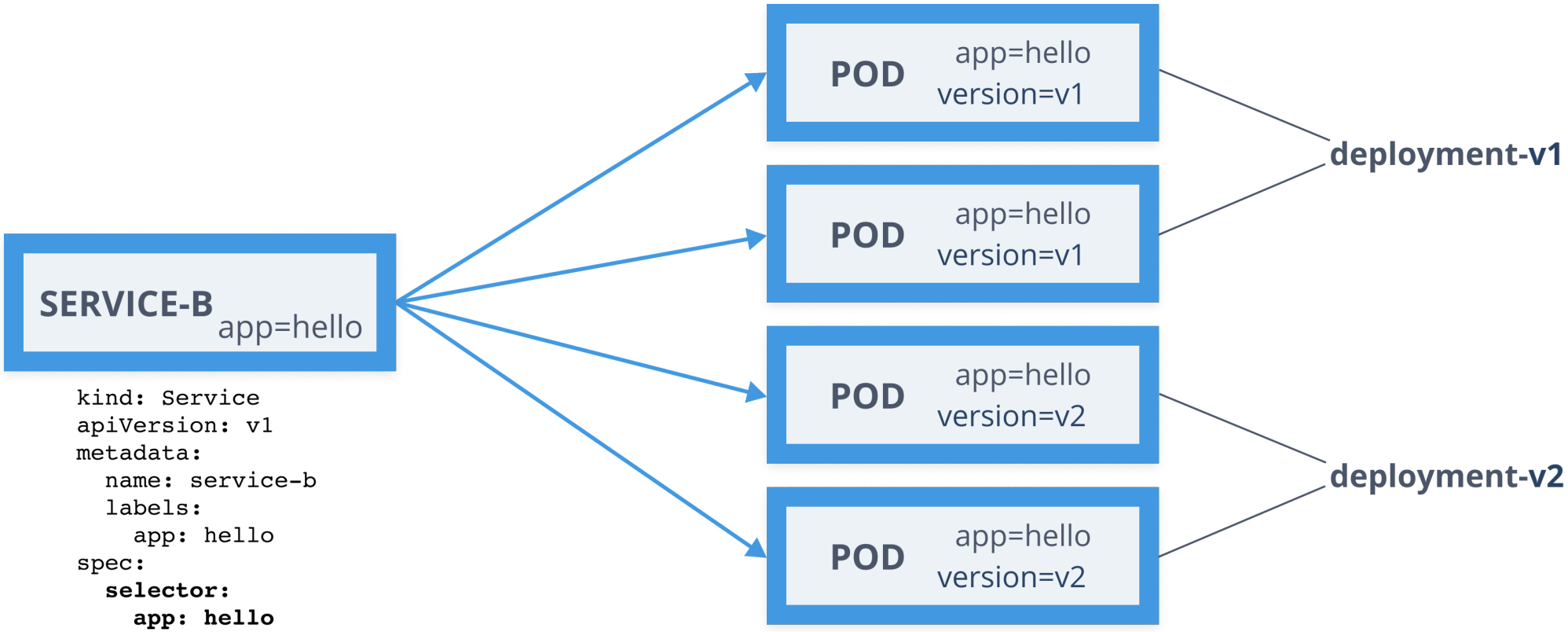
- Gateway
- VirtualService
- DestinationRule
- ServiceEntry
- Sidecar

Service Mesh - Virtual Service

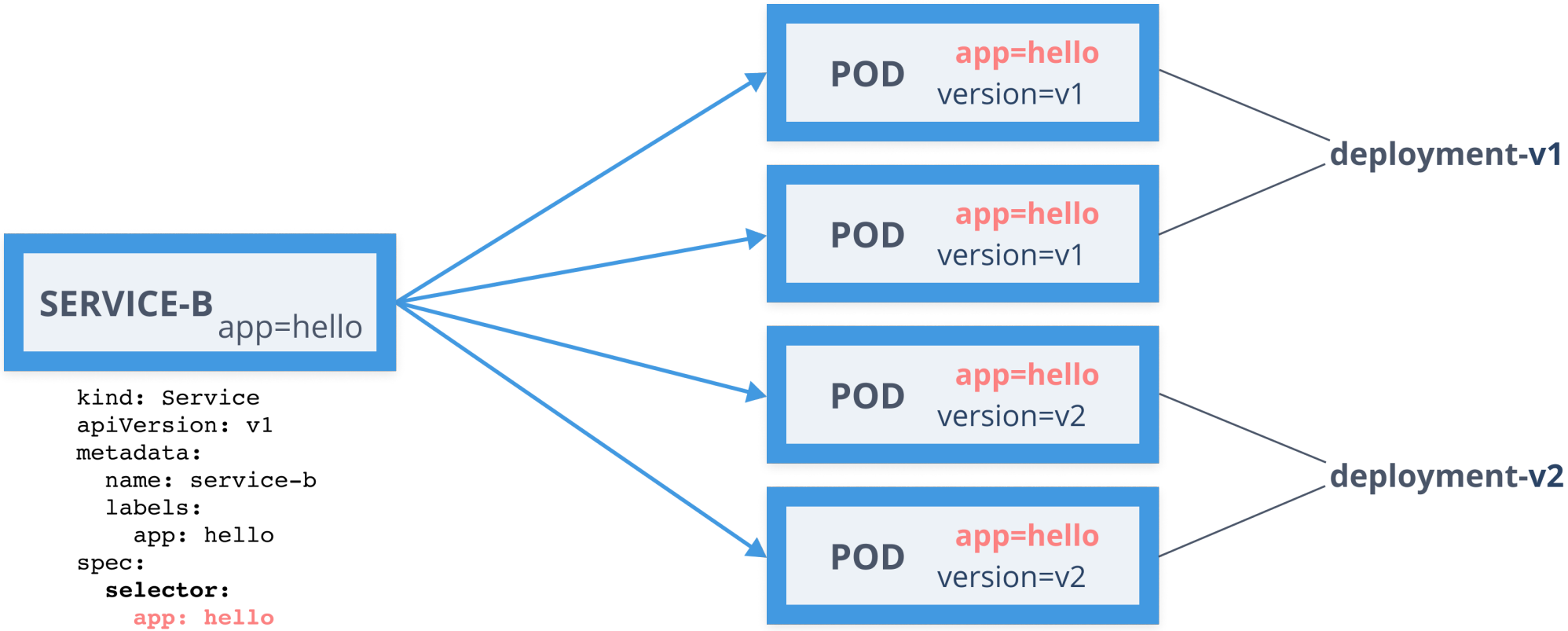
```yaml
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: service-b
spec:
  hosts:
    - service-b.default.svc.cluster.local
  http:
    - route:
        - destination:
            host: service-b
            subset: v1
          weight: 98
        - destination:
            host: service-b
            subset: v2
          weight: 2
```

Service Mesh - Destination Rule

```yaml
apiVersion: networking.istio.io/v1alpha3
kind: DestinationRule
metadata:
  name: service-b
spec:
  host: service-b.default.svc.cluster.local
  subsets:
  - name: v1
    labels:
      version: v1
  - name: v2
    labels:
      version: v2
  trafficPolicy:
  tls:
    mode: ISTIO_MUTUAL
```




```
kind: Service
apiVersion: v1
metadata:
  name: service-b
  labels:
    app: hello
spec:
  selector:
    app: hello
  ports:
    -...
```



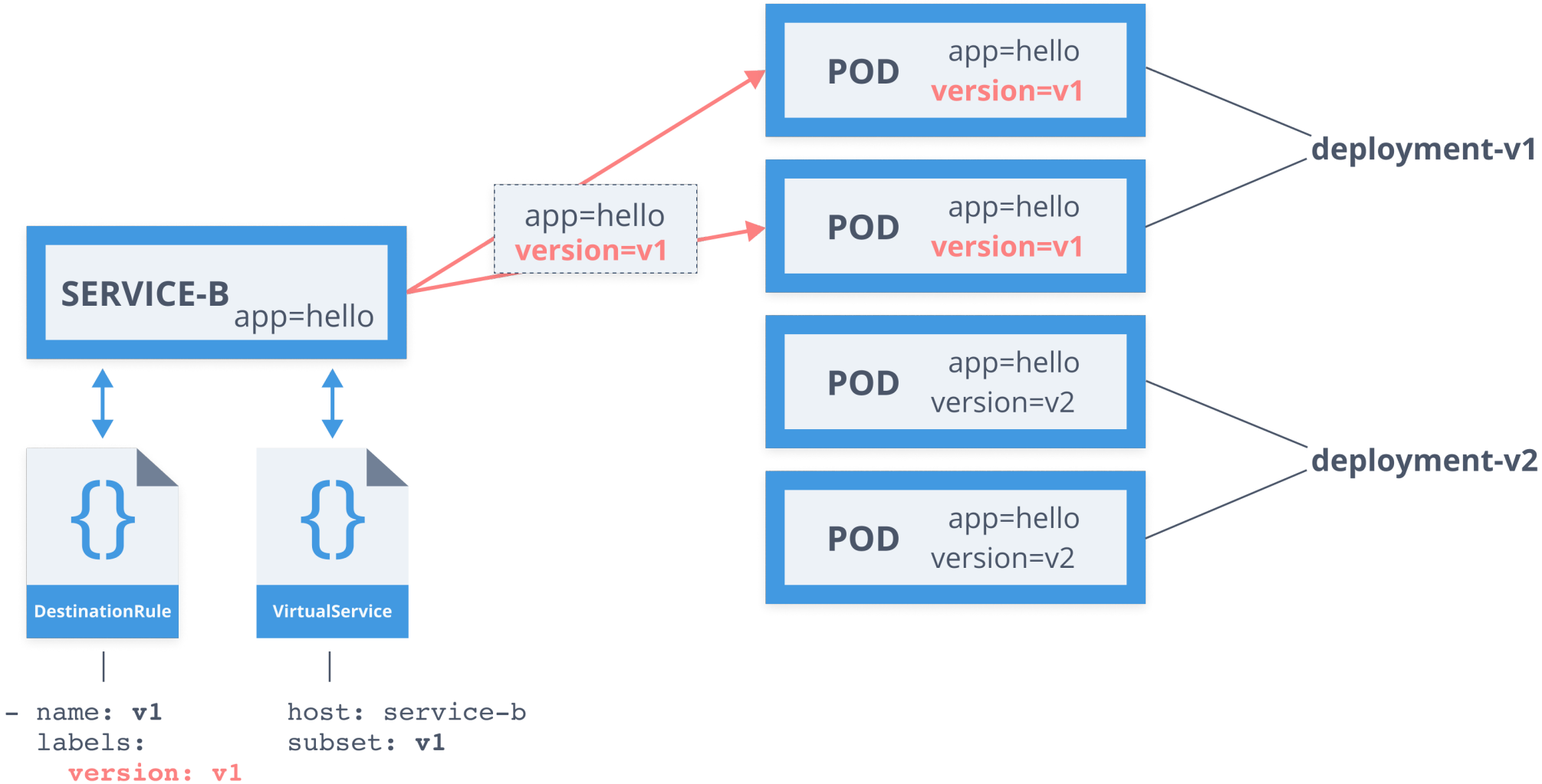

```
kind: Service
apiVersion: v1
metadata:
  name: service-b
  labels:
    app: hello
spec:
  selector:
    app: hello
  ports:
    -...
```

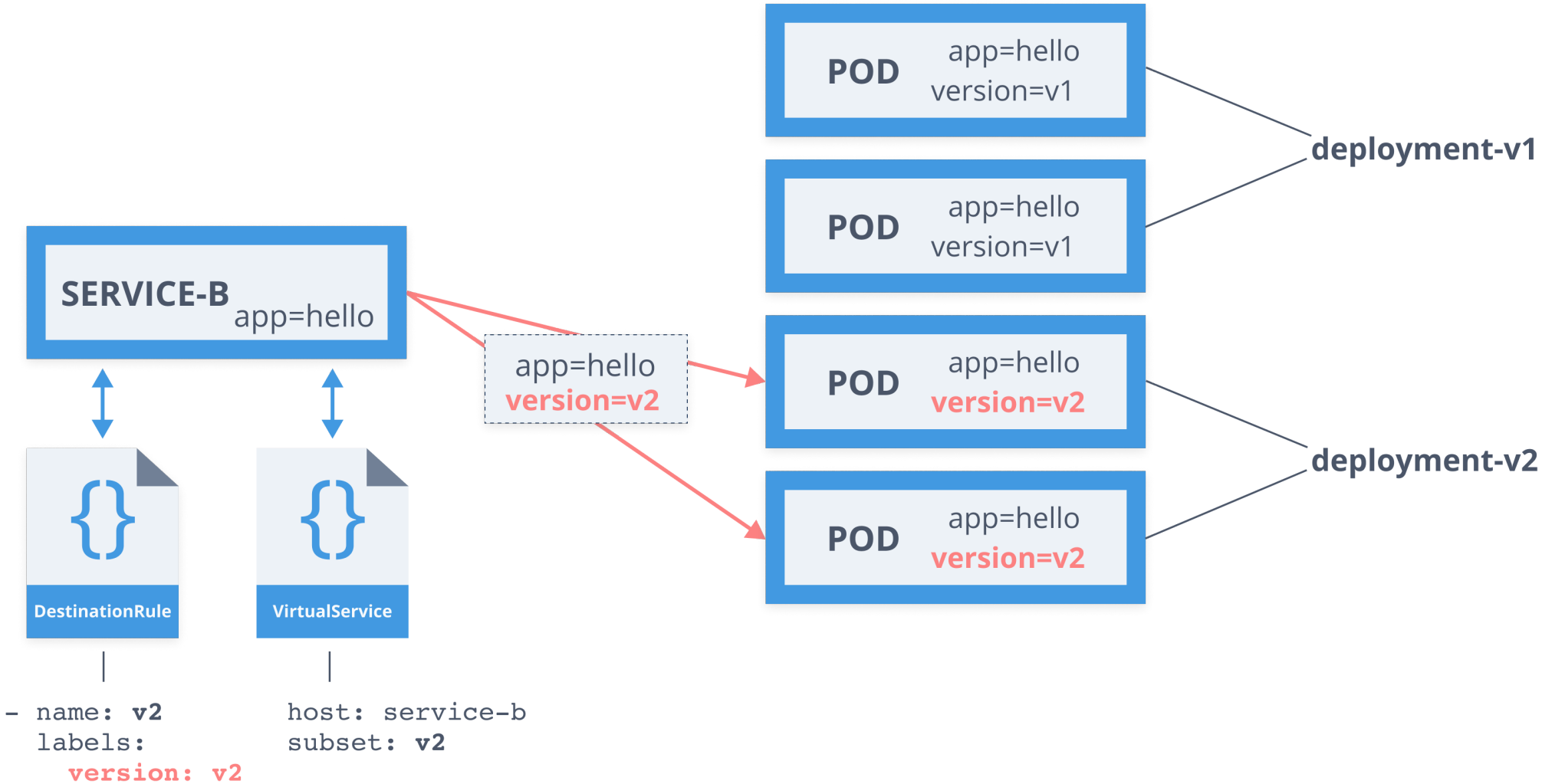
Destination rule

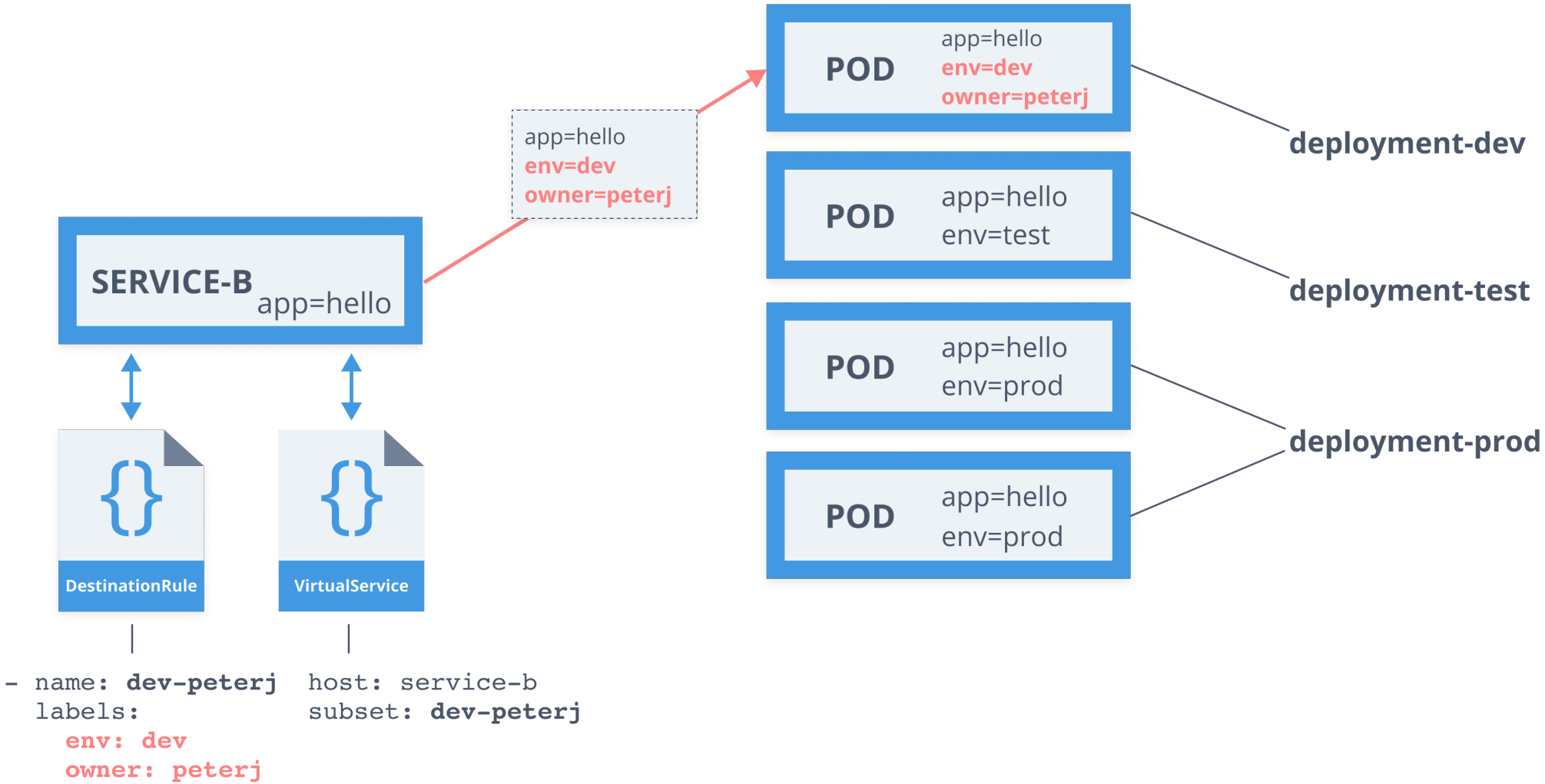
```yaml
apiVersion: networking.istio.io/v1alpha3
kind: DestinationRule
metadata:
  name: service-b
spec:
  host: service-b.default.svc.cluster.local
  subsets:
    - name: v1
      labels:
        version: v1
    - name: v2
      labels:
        version: v2
```

Virtual service

```
1  ...
2  http:
3    - route:
4        - destination:
5            host: service-b
6            subset: v1
7          weight: 30
```


SERVICE-B
app=hello
app=hello
version=v1
POD
app=hello
version=v1
POD
app=hello
version=v1
POD
app=hello
version=v2
POD
app=hello
version=v2
deployment-v1
deployment-v2
DestinationRule
VirtualService
- name: v1
labels:
version: v1
host: service-b
subset: v1


SERVICE-B
app=hello
app=hello
version=v2
POD
app=hello
version=v1
POD
app=hello
version=v1
POD
app=hello
version=v2
POD
app=hello
version=v2
deployment-v1
deployment-v2
DestinationRule
VirtualService
- name: v2
labels:
version: v2
host: service-b
subset: v2

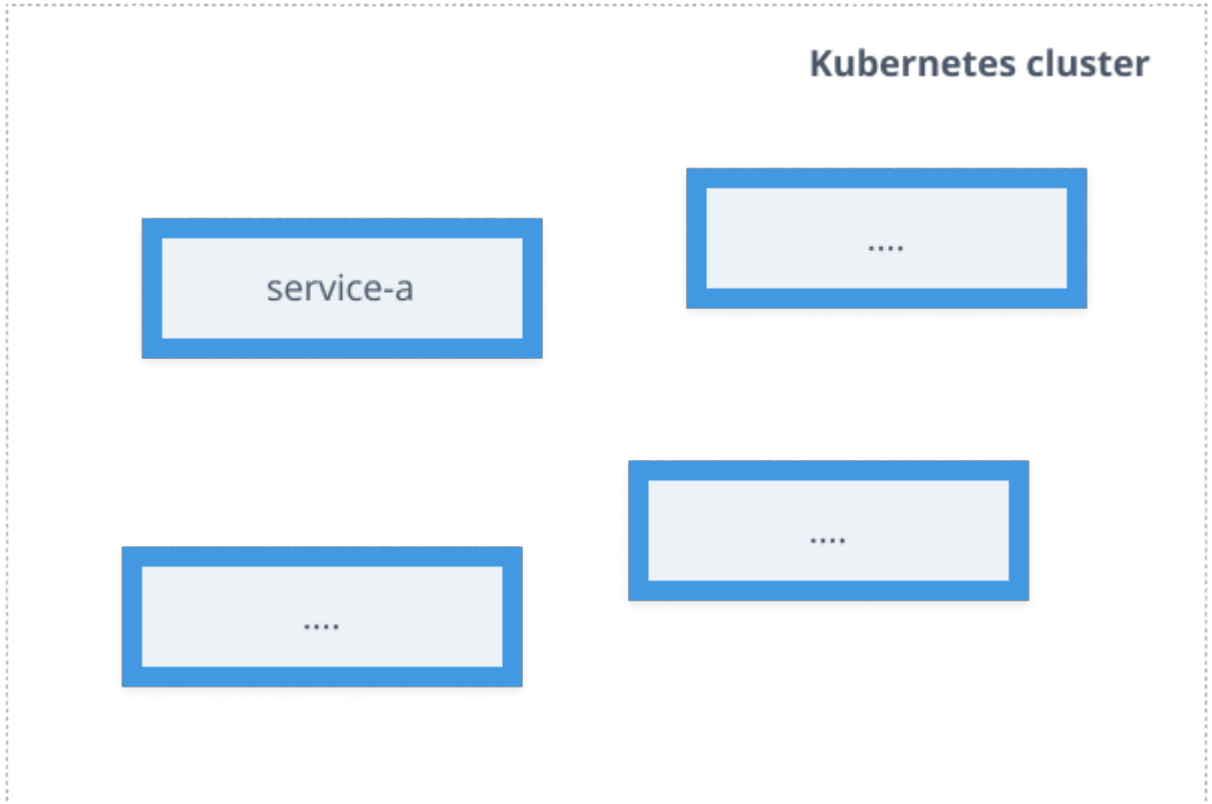

SERVICE-B app=hello
app=hello
env=dev
owner=peterj
POD app=hello env=dev owner=peterj
POD app=hello env=test
POD app=hello env=prod
POD app=hello env=prod
deployment-dev
deployment-test
deployment-prod
DestinationRule
VirtualService
- name: dev-peterj
labels:
env: dev
owner: peterj
host: service-b
subset: dev-peterj

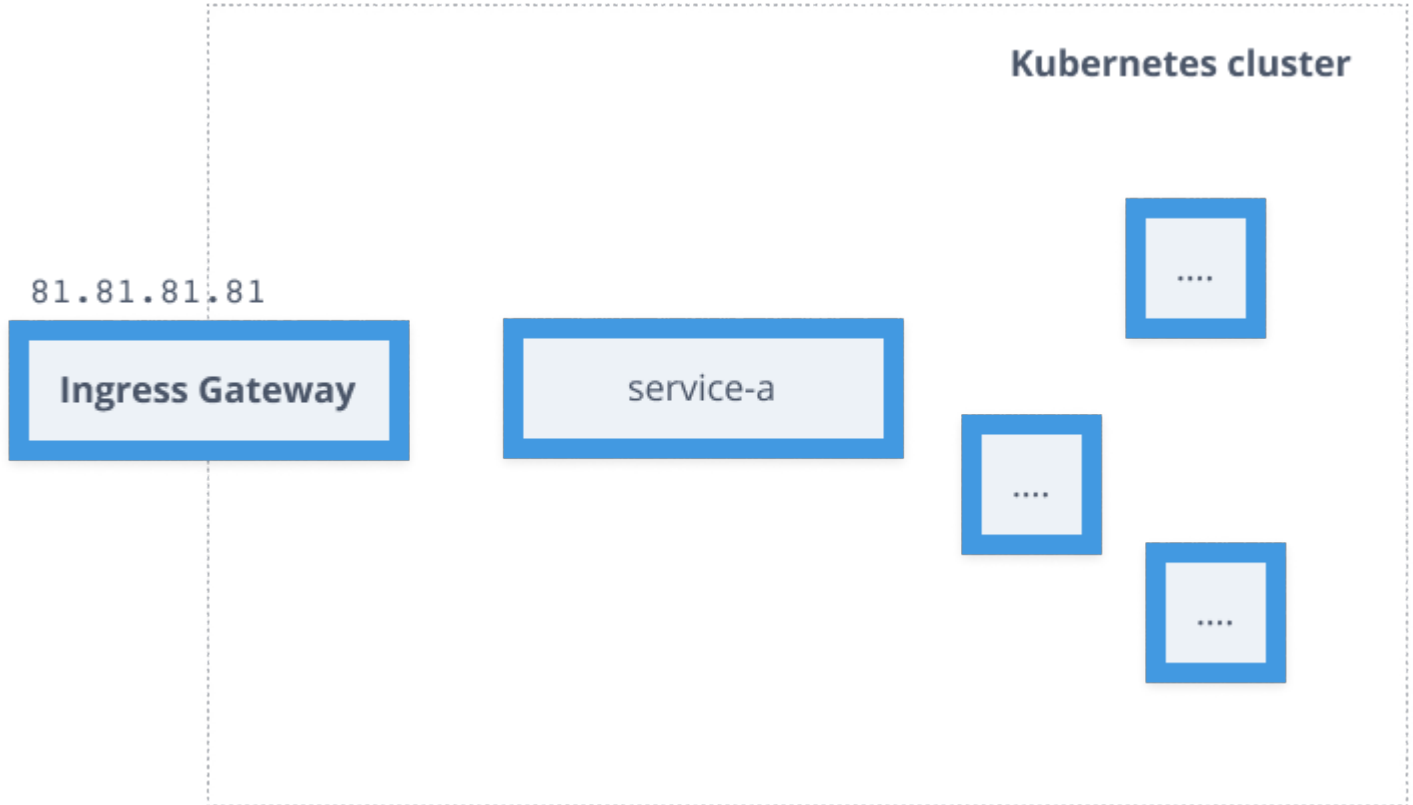
Service Mesh - Service Entry

```yaml
apiVersion: networking.istio.io/v1alpha3
kind: ServiceEntry
metadata:
  name: movie-db
spec:
  hosts:
    - api.themoviedb.org
  ports:
    - number: 443
      name: https
      protocol: HTTPS
   resolution: DNS
   location: MESH_EXTERNAL
```

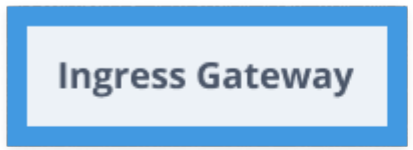
Service Mesh - Gateway

```
apiVersion: networking.istio.io/v1alpha3
kind: Gateway
metadata:
  name: gateway
spec:
  selector:
     istio: ingressgateway
  servers:
    - port:
         number: 80
        name: http
        protocol: HTTP
    hosts:
     - "hello.example.com"
```


Kubernetes cluster
service-a
....
....
....


Kubernetes cluster
81.81.81.81
Ingress Gateway
service-a
....
....
....

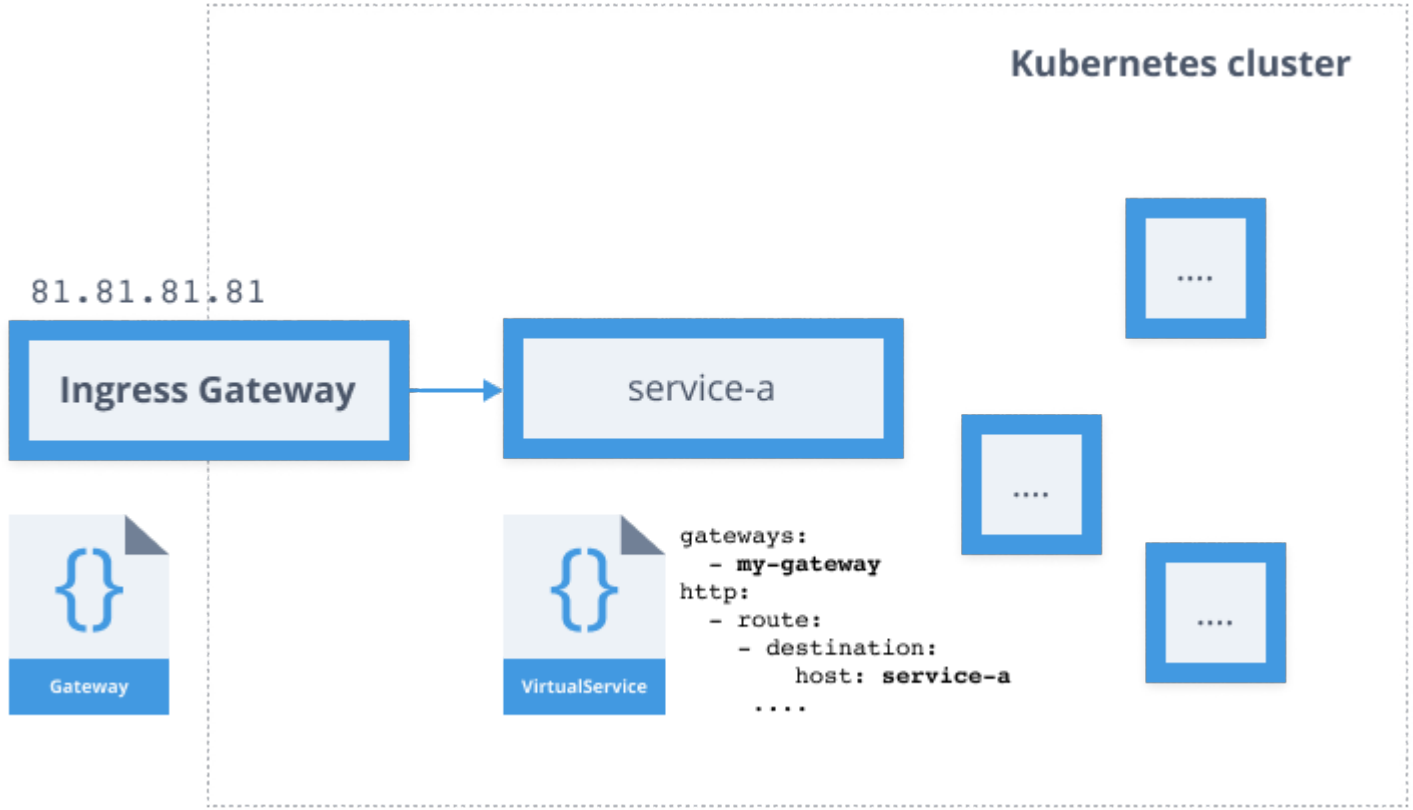
81.81.81.81



Domain Registrar - DNS settings

CNAME: **www.service-a.com**
A: **81.81.81.81**

```
servers:
   - port:
        number: 80
       name: http
       protocol: HTTP
   hosts:
    - "www.service-a.com"
```


Kubernetes cluster
81.81.81.81
Ingress Gateway
service-a
....
....
....
Gateway
VirtualService
gateways:
- my-gateway
http:
- route:
- destination:
host: service-a
....

Service Mesh - Sidecar

```yaml
apiVersion: networking.istio.io/v1alpha3
kind: Sidecar
metadata:
  name: default
  namespace: prod-us-west-1
spec:
  egress:
    - hosts:
        - 'prod-us-west-1/*'
        - 'prod-apis/*'
        - 'istio-system/*'
```

Service Mesh - Traffic Management

- Define subsets in DestinationRule
- Define route rules in VirtualService
- Define one or more destinations with weights

Demo

Istio Traffic Routing

Exercises - Traffic Management

`./istio/traffic`

https://github.com/peterj/jfuture

Service Resiliency

Resiliency

Ability to recover from failures and continue to function

Return the service to a fully functioning state after failure

Resiliency

High availability

- Healthy
- No significant downtime
- Responsive
- Meeting SLAs

Disaster recovery

- Design can't handle the impact of failures
- Data backup & archiving

Resiliency Strategies

- Load Balancing
- Timeouts and retries
- Circuit breakers and bulkhead pattern
- Data replication
- Graceful degradation
- Rate limiting

Service Mesh - Timeouts

```
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
    name: service-b
spec:
    hosts:
    - service-b.default.svc.cluster.local
    http:
    - route:
        - destination:
            host: service-b
            subset: v1
      timeout: 5s
```

Service Mesh - Retries

```yaml
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: service-b
spec:
  hosts:
    - service-b.default.svc.cluster.local
  http:
    - route:
        - destination:
            host: service-b
            subset: v1
      retries:
        attempts: 3
        perTryTimeout: 3s
        retryOn: gateway-error,connect-failure
```

Service Mesh - Circuit Breakers

```yaml
apiVersion: networking.istio.io/v1alpha3
kind: DestinationRule
metadata:
  name: service-b
spec:
  host: service-b.default.svc.cluster.local
   trafficPolicy:
     tcp:
       maxConnections: 1
     http:
       http1MaxPendingRequests: 1
       maxRequestsPerConnection: 1
     outlierDetection:
       consecutiveErrors: 1
       interval: 1s
       baseEjectionTime: 3m
       maxEjectionPercent: 100
```

Service Mesh - Delays

```
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: service-b
spec:
  hosts:
    - service-b.default.svc.cluster.local
  http:
    - route:
        - destination:
            host: service-b
            subset: v1
      fault:
        delay:
            percentage: 50
        fixedDelay: 2s
```

Service Mesh - Aborts

```
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: service-b
spec:
  hosts:
    - service-b.default.svc.cluster.local
  http:
    - route:
        - destination:
            host: service-b
            subset: v1
      fault:
        abort:
          percentage: 50
        httpStatus: 404
```

Demo

Service Resiliency

Exercises - Resiliency

`./istio/resiliency`

https://github.com/peterj/jfuture

Security


1 2 3
4 5 6
7 8 9
* 0 #
ELITE

Access Control

Can a principal perform an action on an object?

Access Control

Can a principal perform an action on an object?

Principal = user

Action = delete

Object = file

Authentication (authn)

- Verify credential is valid/authentic
- Istio: X.509 certificates
- Identity encoded in certificate

Authorization (authz)

- Is principal allowed to perform an action on an object?
- Istio: RBAC policies
  - Role-based access control

Authentication and authorization work together

Identity - SPIFFE

- SPIFFE (Secure Production Identity Framework for Everyone)
- Specially formed X.509 certificate with an ID (e.g. `spiffe://cluster.local/ns/default/sa/default`)
- Kubernetes: service account is used

https://spiffe.io/

Identity - SPIFFE

Concepts:

- Identity (URI)
  - `spiffe://cluster.local/ns/default/sa/default`
- Encoding of identity into SVID (SPIFFE Verifiable Identity Document)
  - X.509 and Subject Alternate Name (SAN) field
- API for issuing and retrieving SVIDs

Key management

- Citadel
  - Certificate authority (CA)
  - Signs certificate requests that create X.509 certificates
- Citadel (node) agents (SDS - secret discovery service)
  - Broker between Citadel and Envoy proxies
- Envoy

Mutual TLS (mTLS)

Flow

1. Traffic from client gets routed to the client side proxy
2. Client side proxy starts mTLS handshake
    - Secure naming check: verify service account in the cert can run the target service
3. Client and server side proxies establish mTLS connection
4. Server side proxy forwards traffic to the server service

Istio Auth Policies

Authn policy:

- Controls how proxies communicate with one another
- mTLS on/off

Authz policy:

- Requires authn
- Configures which identities are allowed to communicate

Configuring mTLS/JWT

- Policy resource (`authentication.istio.io/v1alpha1.Policy`)
- Scope:
  - Mesh < namespace < service
- Also supports JWT

Configuring authorization

- Who can talk to whom
  - Uses RBAC (role-based access control)
- Service role
  - Actions that can be performed on service by any principal with the role
- Service role binding
  - Assigns roles to principals (principals = service identities = ServiceAccounts)

Configuring RBAC

- ClusterRbacConfig resource (`rbac.istio.io/v1alpha1`)
- Multiple modes:
  - On, off
  - On with inclusion, on with exclusion

Exercises - Security

`./istio/security`

https://github.com/peterj/jfuture

Istio vs. Linkerd vs. Consul Connect?

- Linkerd: Kubernetes only
- Consul: agent per-node + proxies
- Linkerd: no circut breaking*
- Consul: no failure injection

https://github.com/linkerd/linkerd2/issues/2846

How to get started?

- Do you need a service mesh?
- Start small and slow:
  - Learn and understand the resources
  - Apply to a subset of services
  - Understand the metrics, logs, dashboards
  - Repeat

https://istio.io

Thank you

FEEDBACK

https://forms.gle/xJW4TVqCfXbhTftm8

Slides: https://slides.peterj.dev/jfuture-2019

Exercises: https://github.com/peterj/jfuture

Contact

- @pjausovec
- peterj.dev

Table of Contents

- Introduction to Containers
- Docker Overview
- Kubernetes Overview
- Service Mesh
- Traffic Management
- Resiliency
- Security