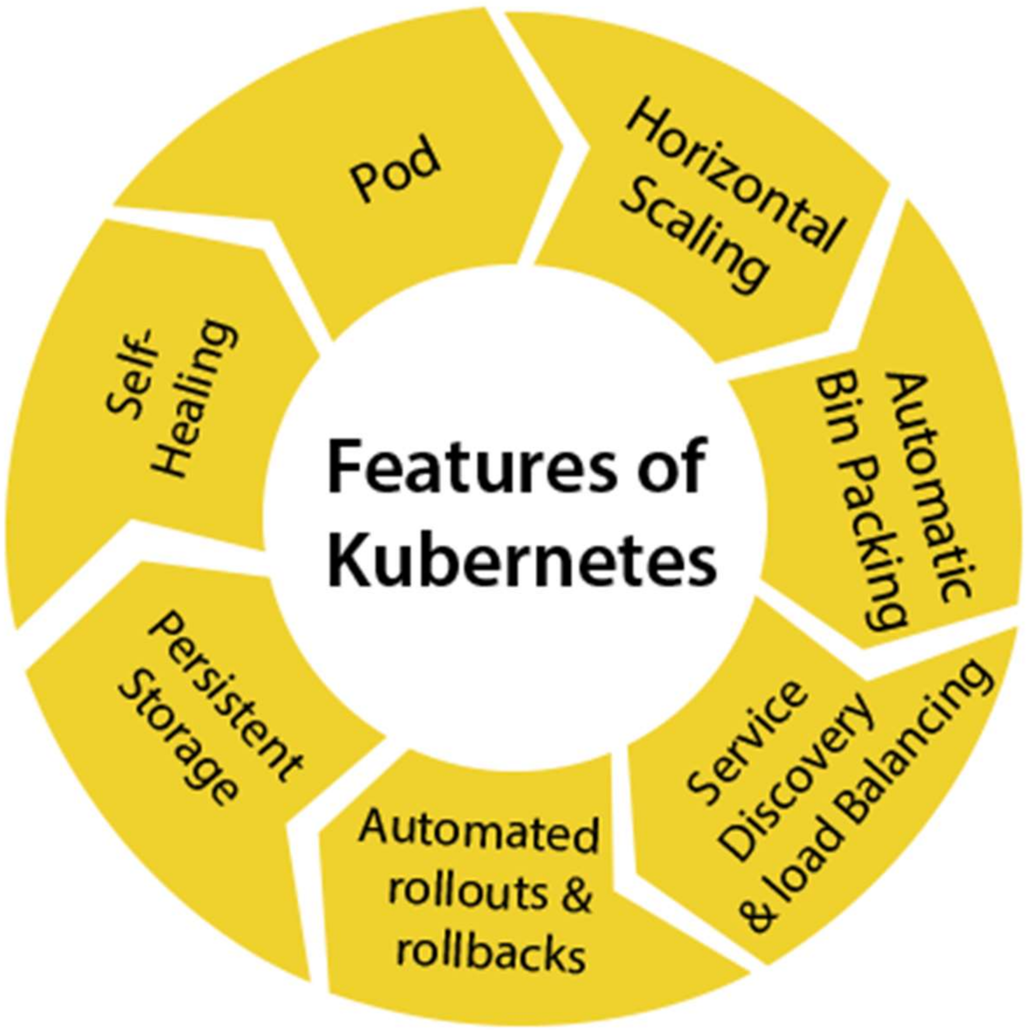
Introduction to Kubernetes

Features


Features of Kubernetes
Pod
Horizontal Scaling
Automatic Bin Packing
Service Discovery & load Balancing
Automated rollouts & rollbacks
Persistent Storage
Self-Healing

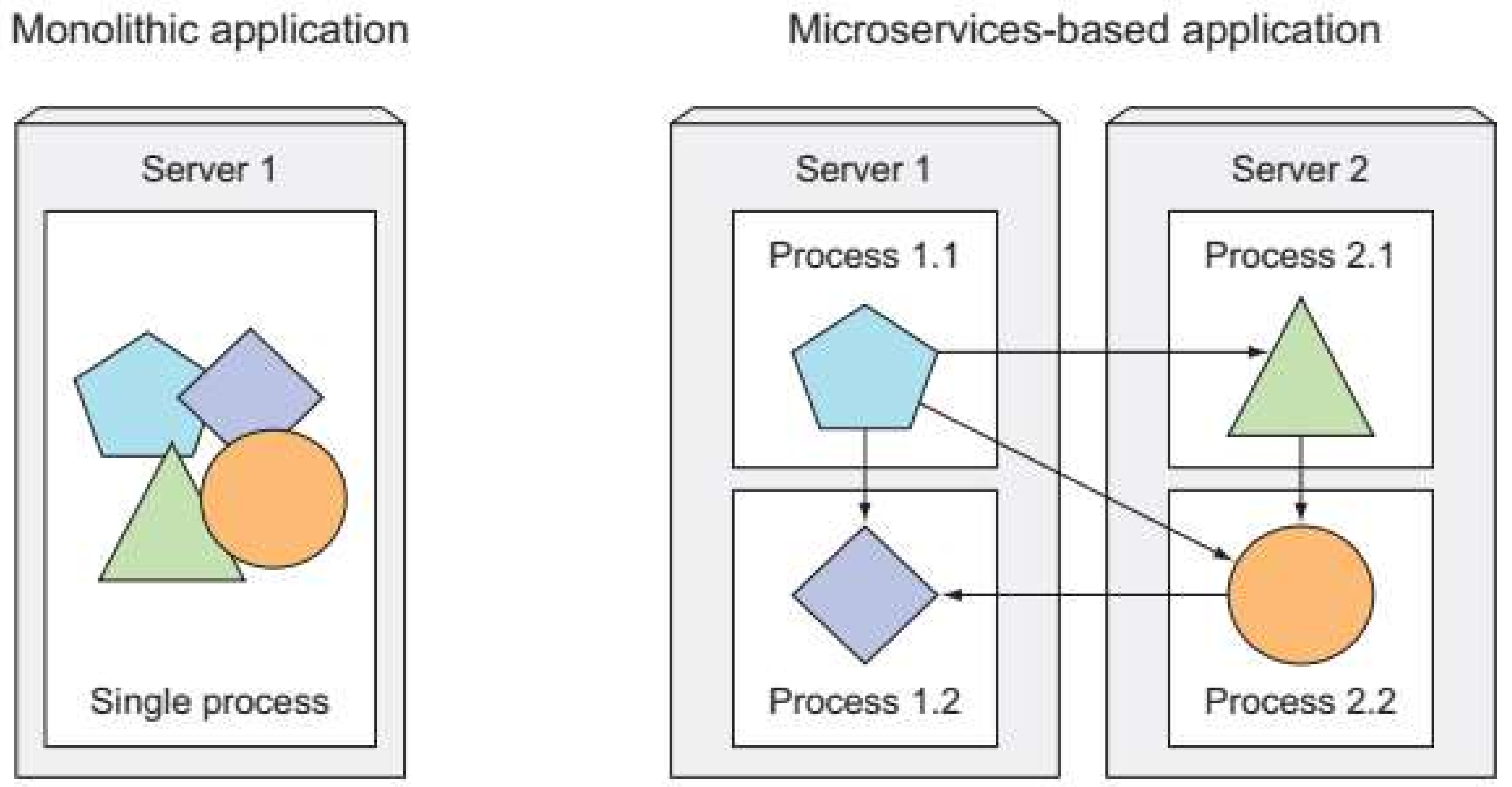
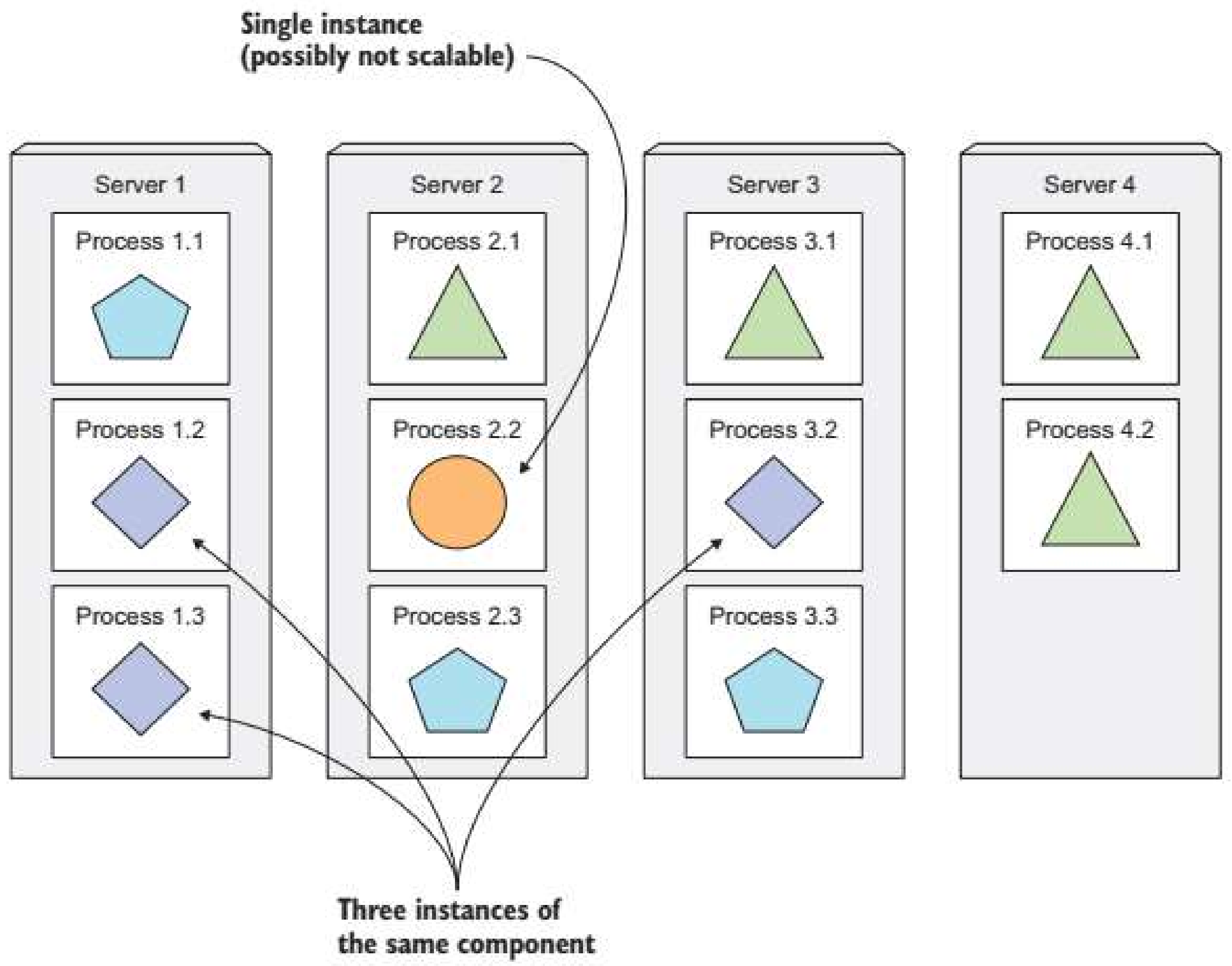
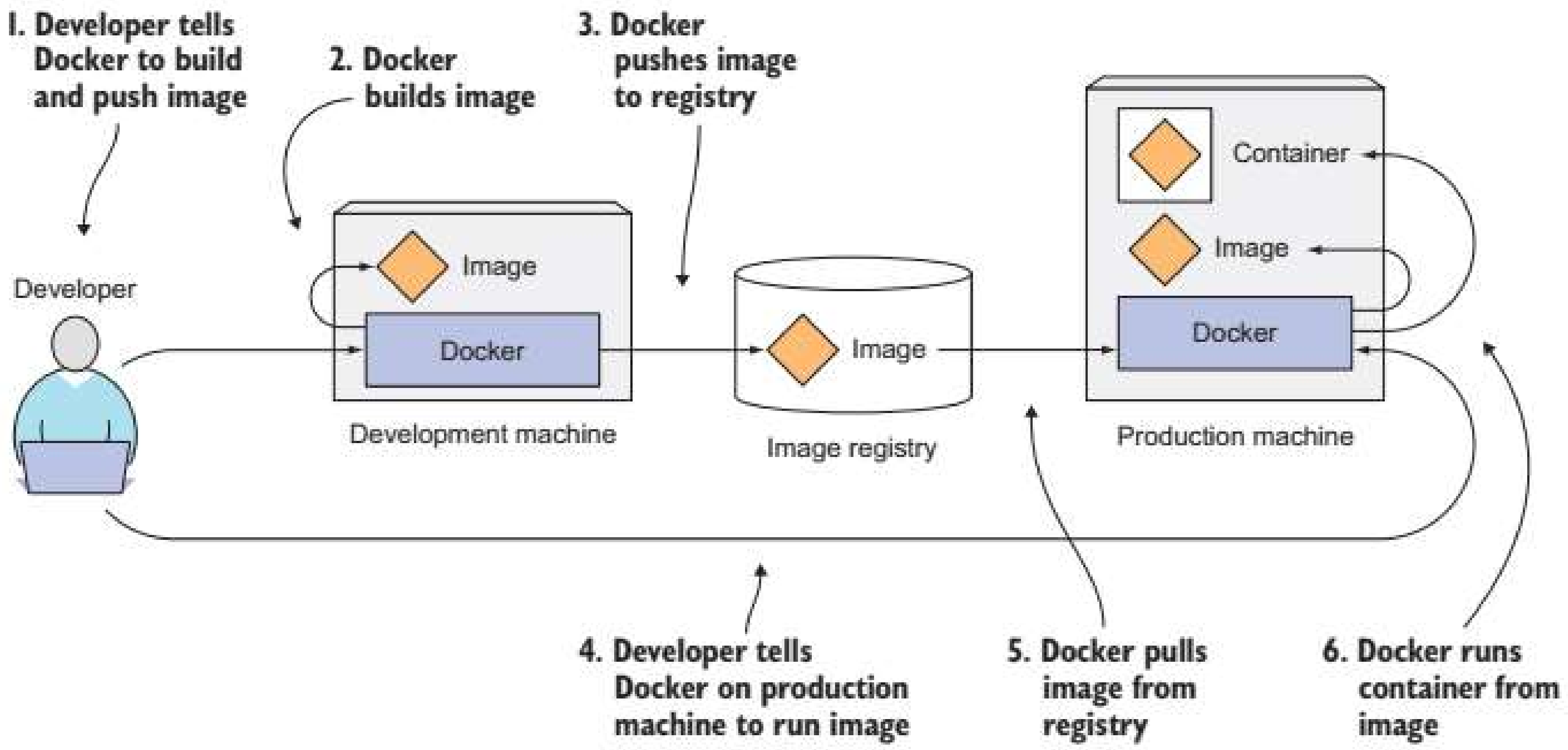


Figure 1.1 Components inside a monolithic application vs. standalone microservices


Single instance
(possibly not scalable)
Server 1
Process 1.1
Process 1.2
Process 1.3
Server 2
Process 2.1
Process 2.2
Process 2.3
Server 3
Process 3.1
Process 3.2
Process 3.3
Server 4
Process 4.1
Process 4.2
Three instances of
the same component


1. Developer tells Docker to build and push image
2. Docker builds image
3. Docker pushes image to registry
Developer
Image
Docker
Development machine
Image
Image registry
Container
Image
Docker
Production machine
4. Developer tells Docker on production machine to run image
5. Docker pulls image from registry
6. Docker runs container from image

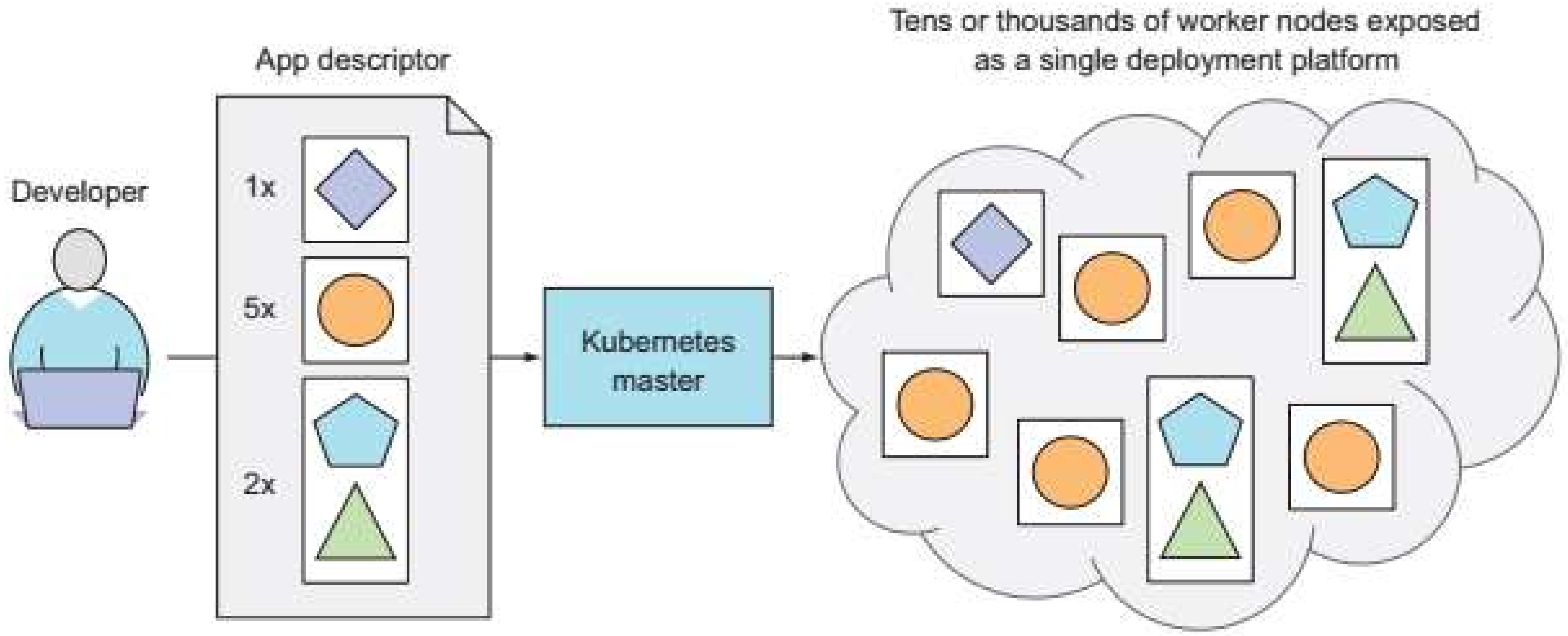

Developer
App descriptor
1x
5x
2x
Kubernetes master
Tens or thousands of worker nodes exposed as a single deployment platform

Figure 1.8 Kubernetes exposes the whole datacenter as a single deployment platform.

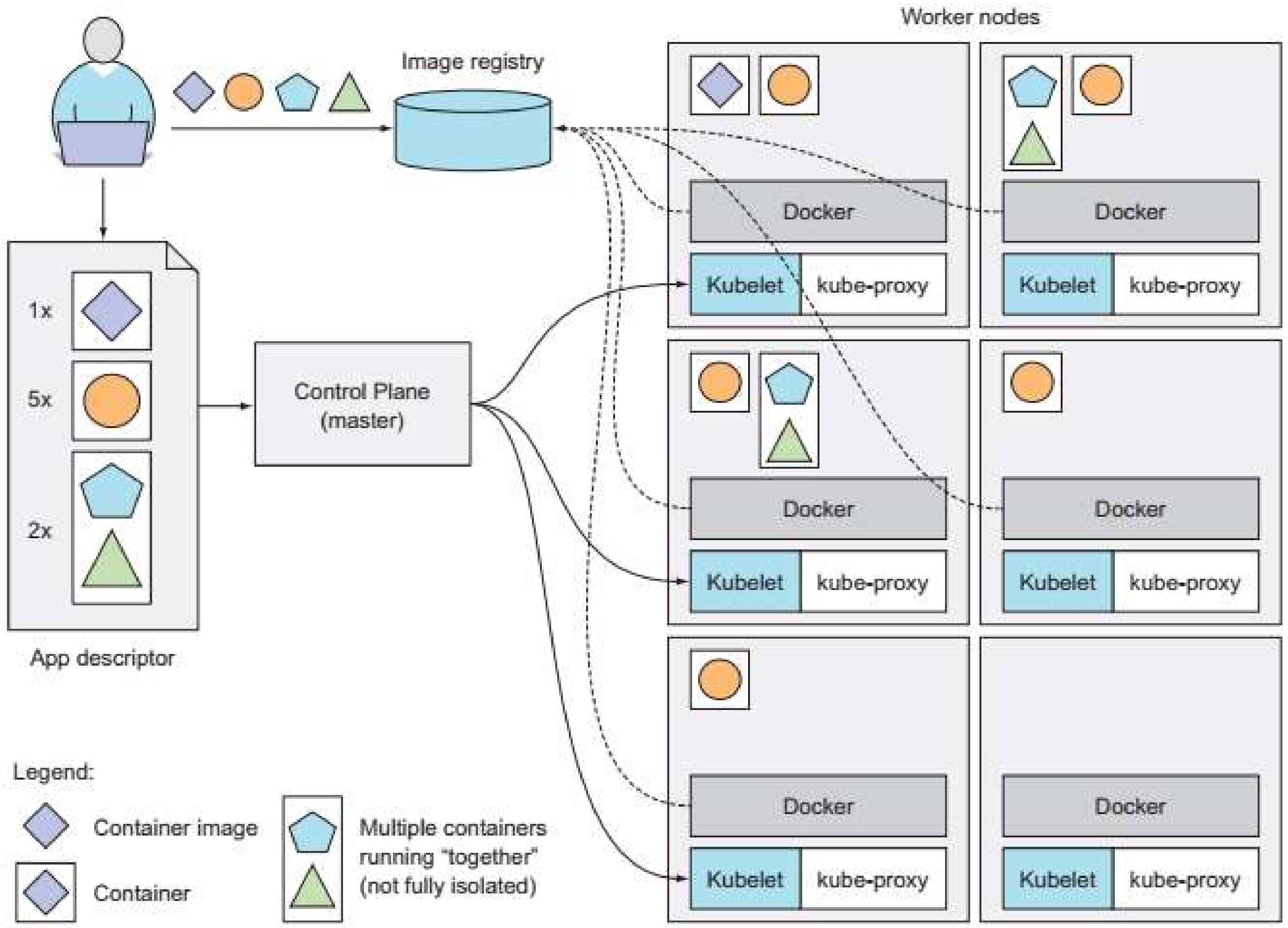


Figure 1.10 A basic overview of the Kubernetes architecture and an application running on top of it

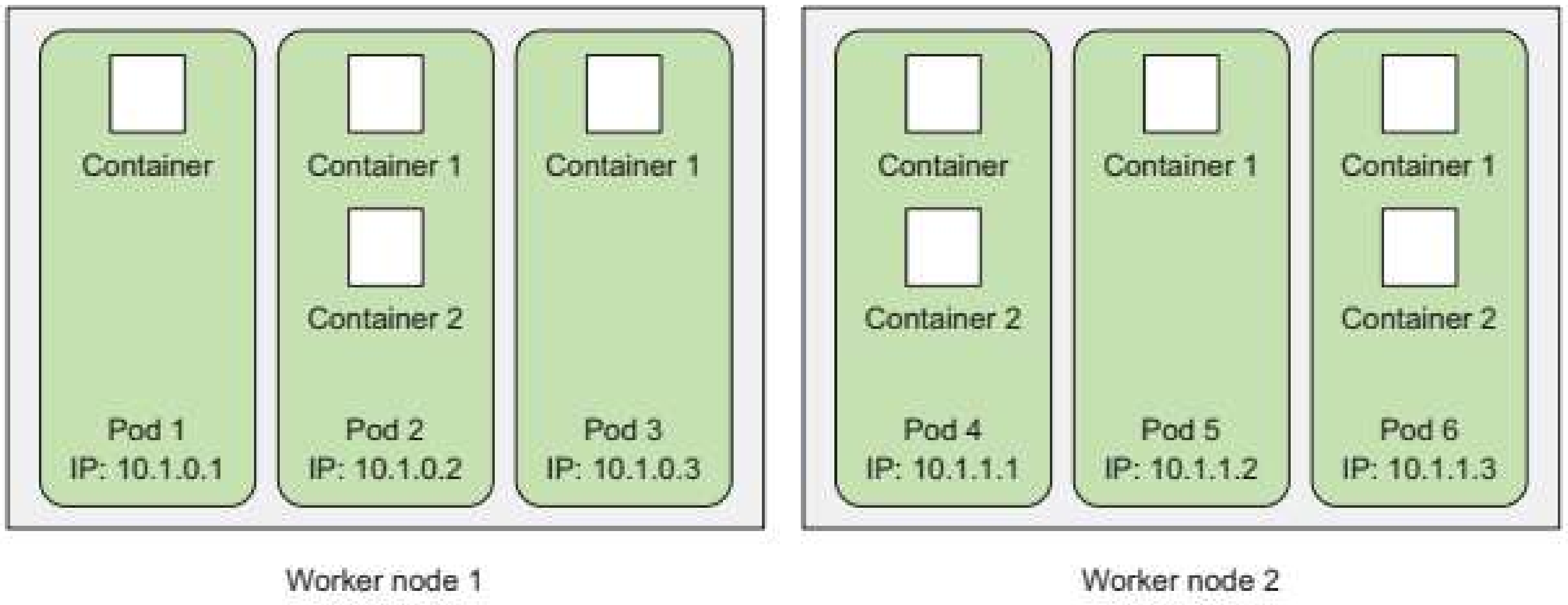


Figure 2.5 The relationship between containers, pods, and physical worker nodes

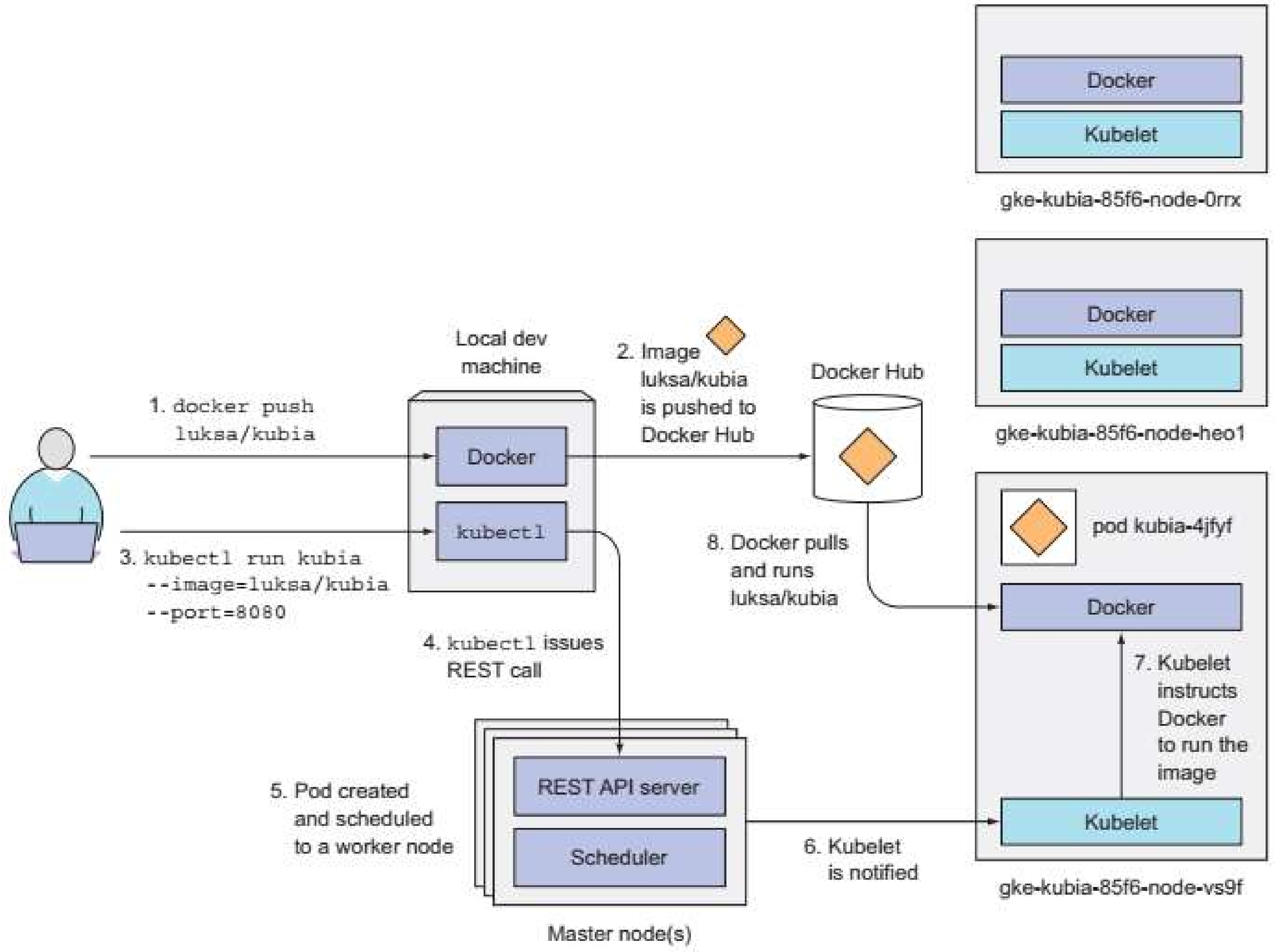

Docker
Kubelet
gke-kubia-85f6-node-0rrx
Docker
Kubelet
gke-kubia-85f6-node-heo1
Local dev machine
2. Image luksa/kubia is pushed to Docker Hub
Docker Hub
1. docker push luksa/kubia
Docker
kubectl
pod kubia-4jfyf
3. kubectl run kubia --image=luksa/kubia --port=8080
8. Docker pulls and runs luksa/kubia
Docker
4. kubectl issues REST call
7. Kubelet instructs Docker to run the image
REST API server
Scheduler
5. Pod created and scheduled to a worker node
6. Kubelet is notified
Kubelet
gke-kubia-85f6-node-vs9f
Master node(s)

Figure 2.6 Running the `luksa/kubia` container image in Kubernetes

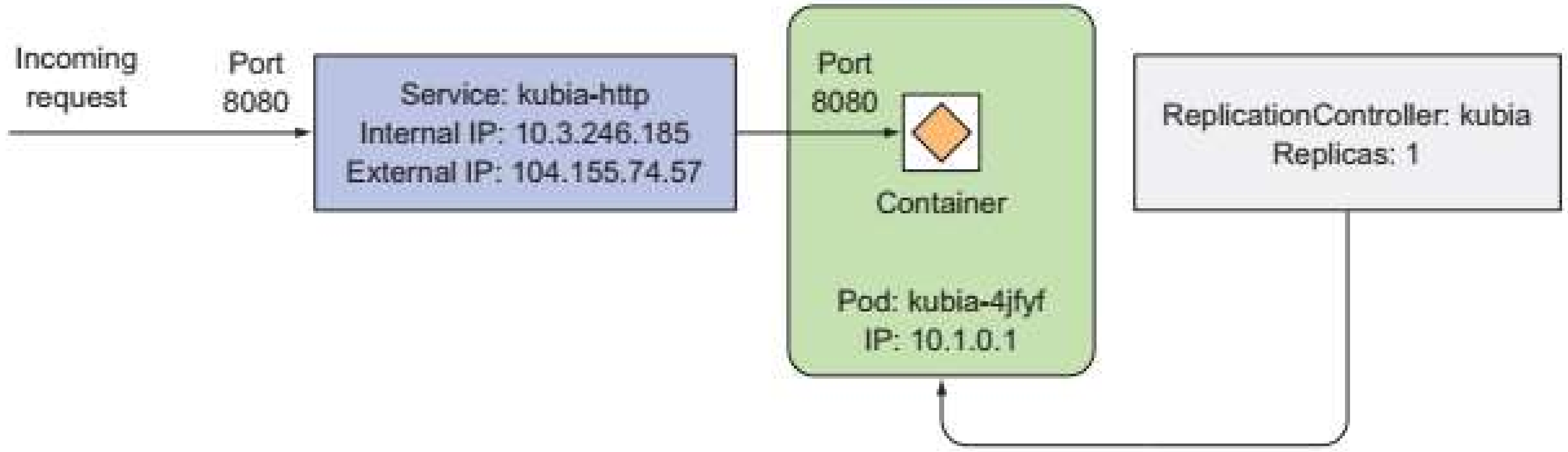


Figure 2.7 Your system consists of a ReplicationController, a Pod, and a Service.

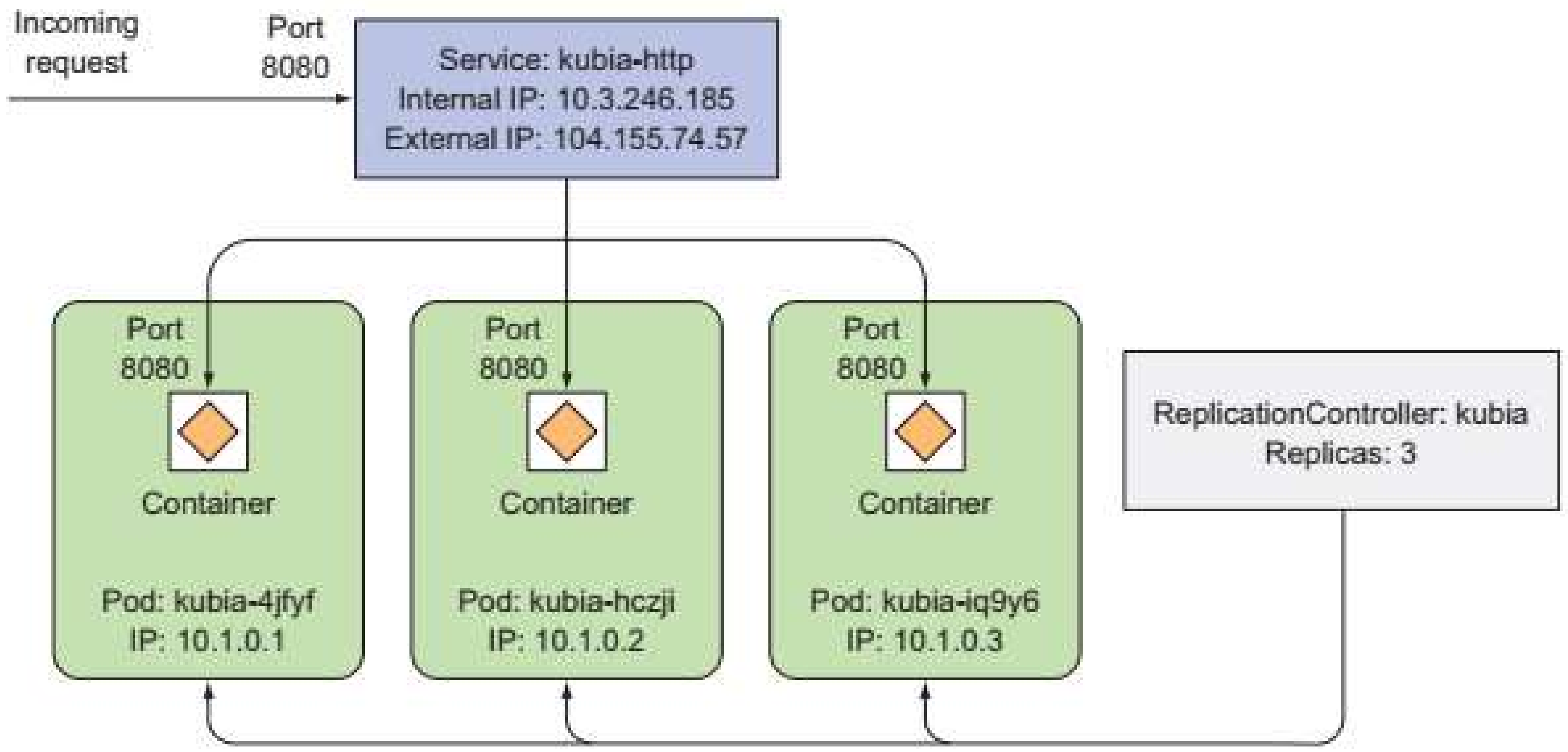
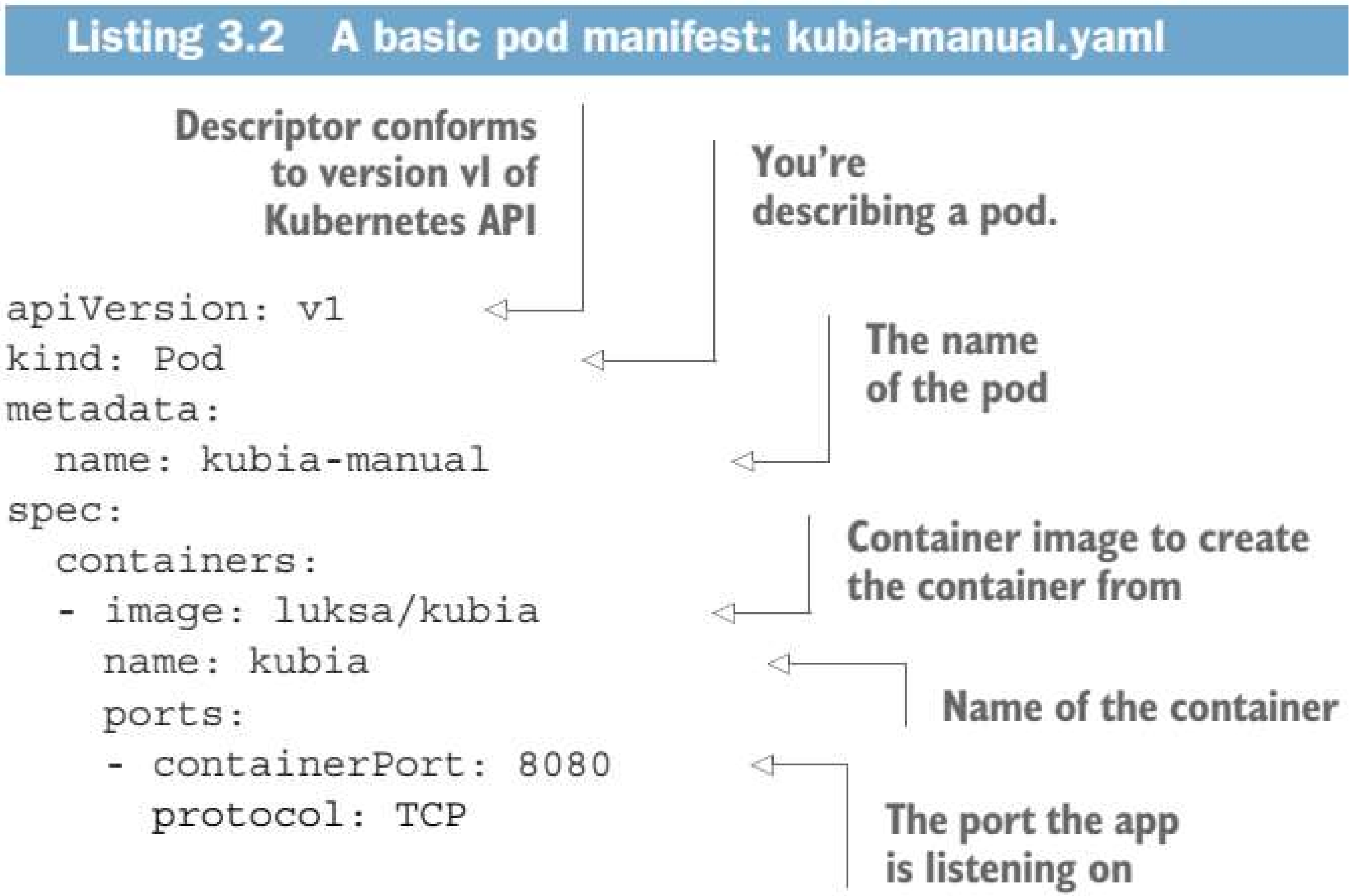


Figure 2.8 Three instances of a pod managed by the same ReplicationController and exposed through a single service IP and port.



Listing 3.2 A basic pod manifest: kubia-manual.yaml

```
apiVersion: v1
kind: Pod
metadata:
  name: kubia-manual
spec:
  containers:
  - image: luksa/kubia
    name: kubia
    ports:
    - containerPort: 8080
      protocol: TCP
```

Descriptor conforms to version vl of Kubernetes API

You're describing a pod.

The name of the pod

Container image to create the container from

Name of the container

The port the app is listening on

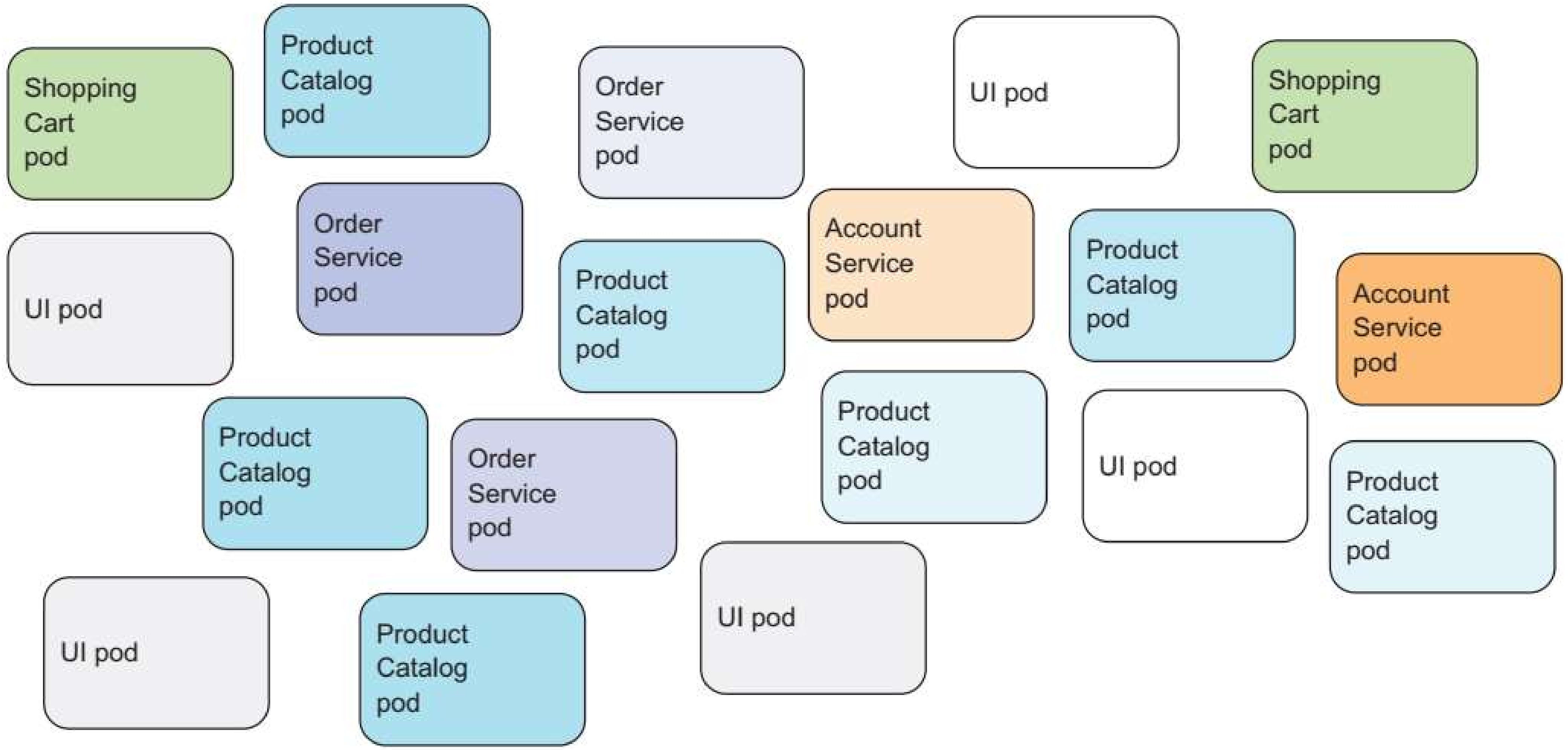


Figure 3.6 Uncategorized pods in a microservices architecture

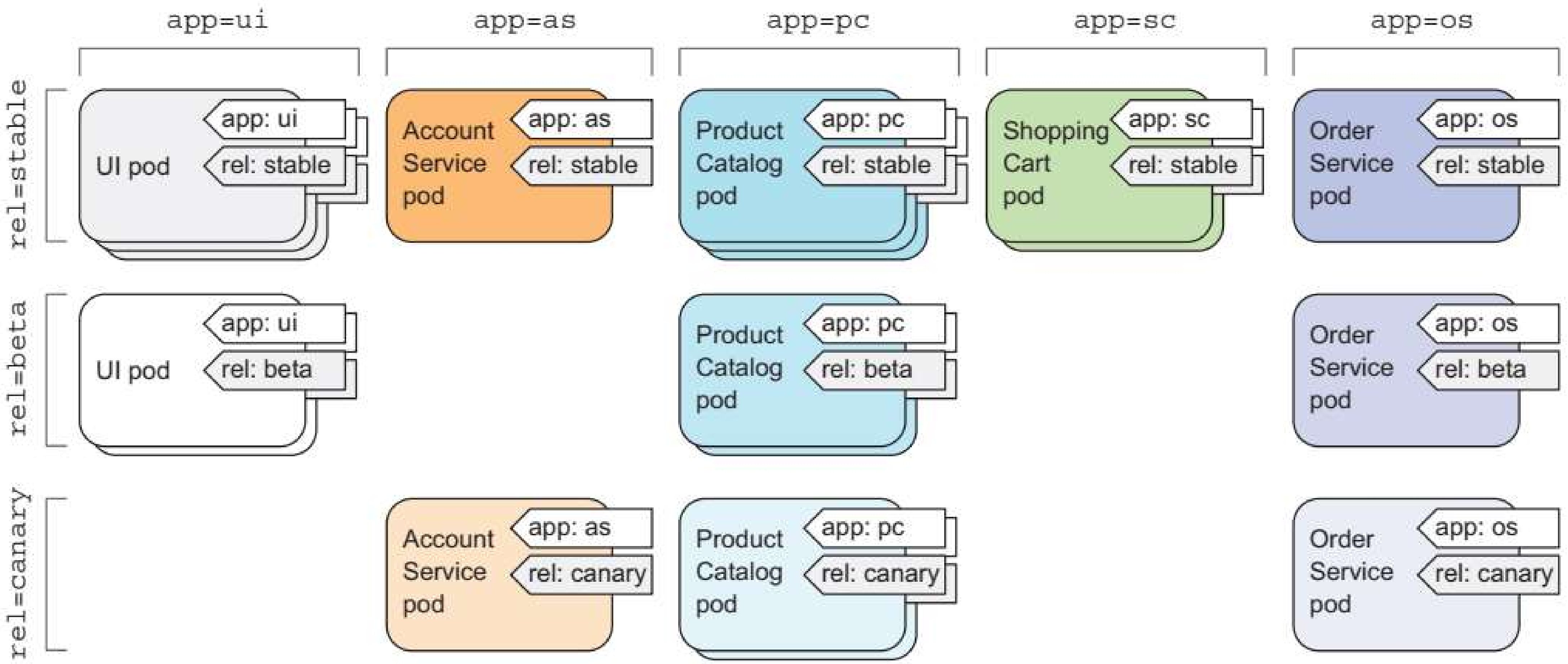


Figure 3.7 Organizing pods in a microservices architecture with pod labels

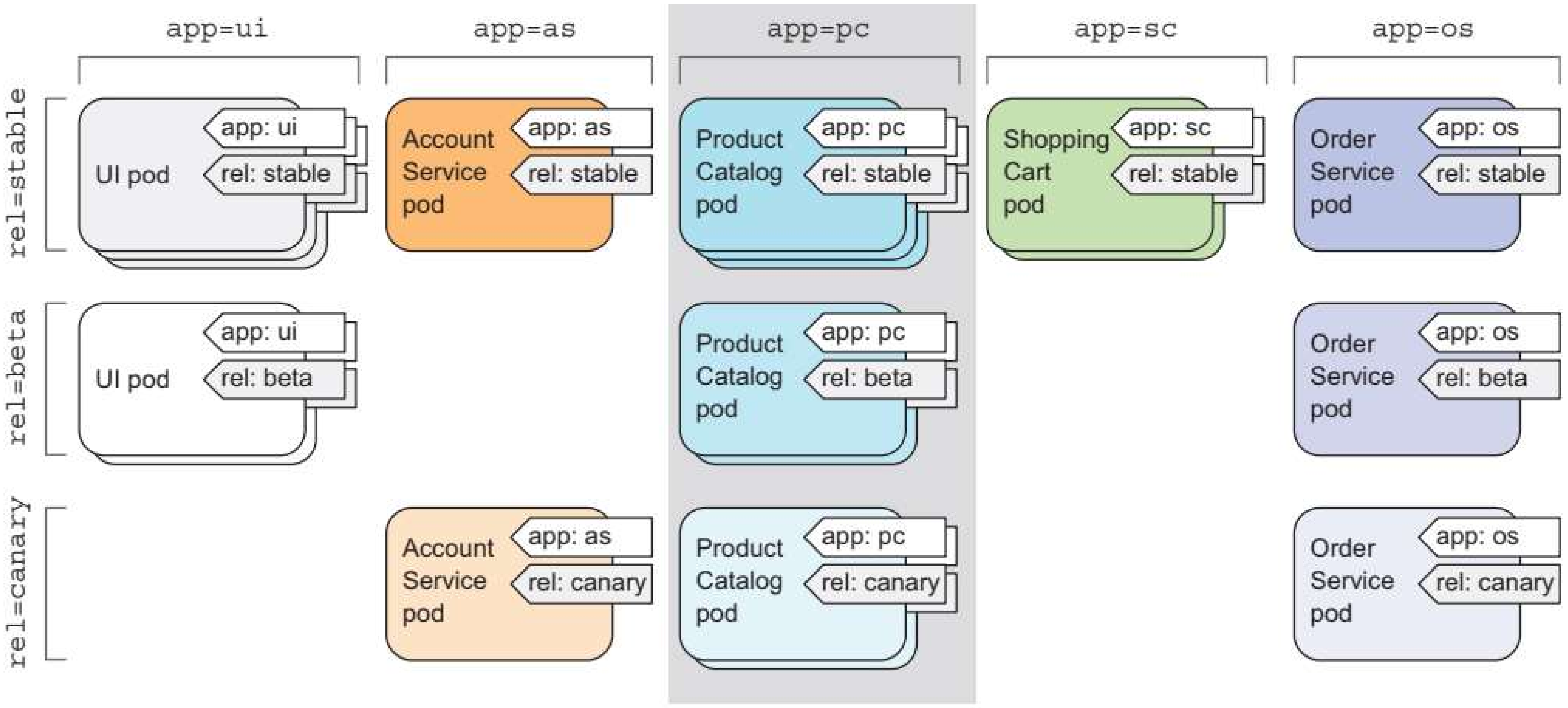


Figure 3.8 Selecting the product catalog microservice pods using the "app=pc" label selector

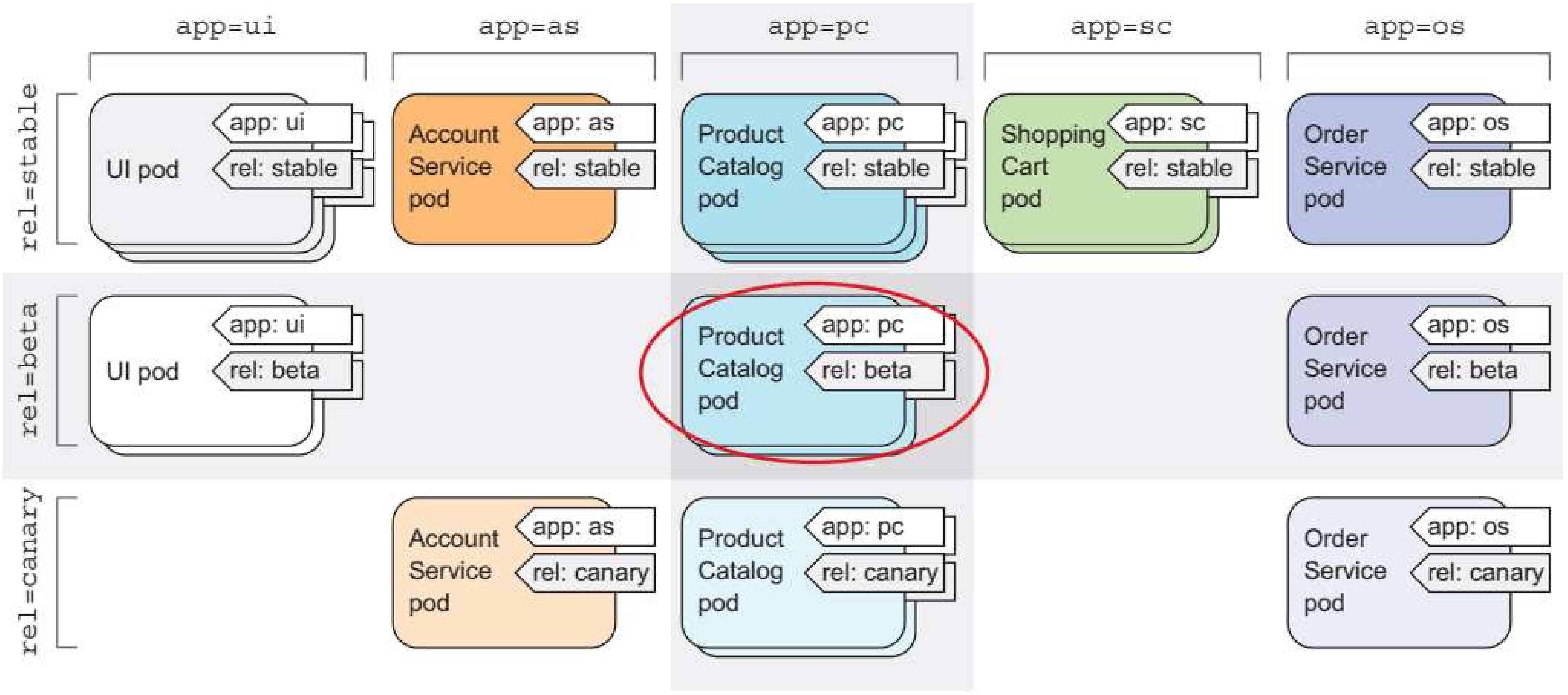


Figure 3.9 Selecting pods with multiple label selectors

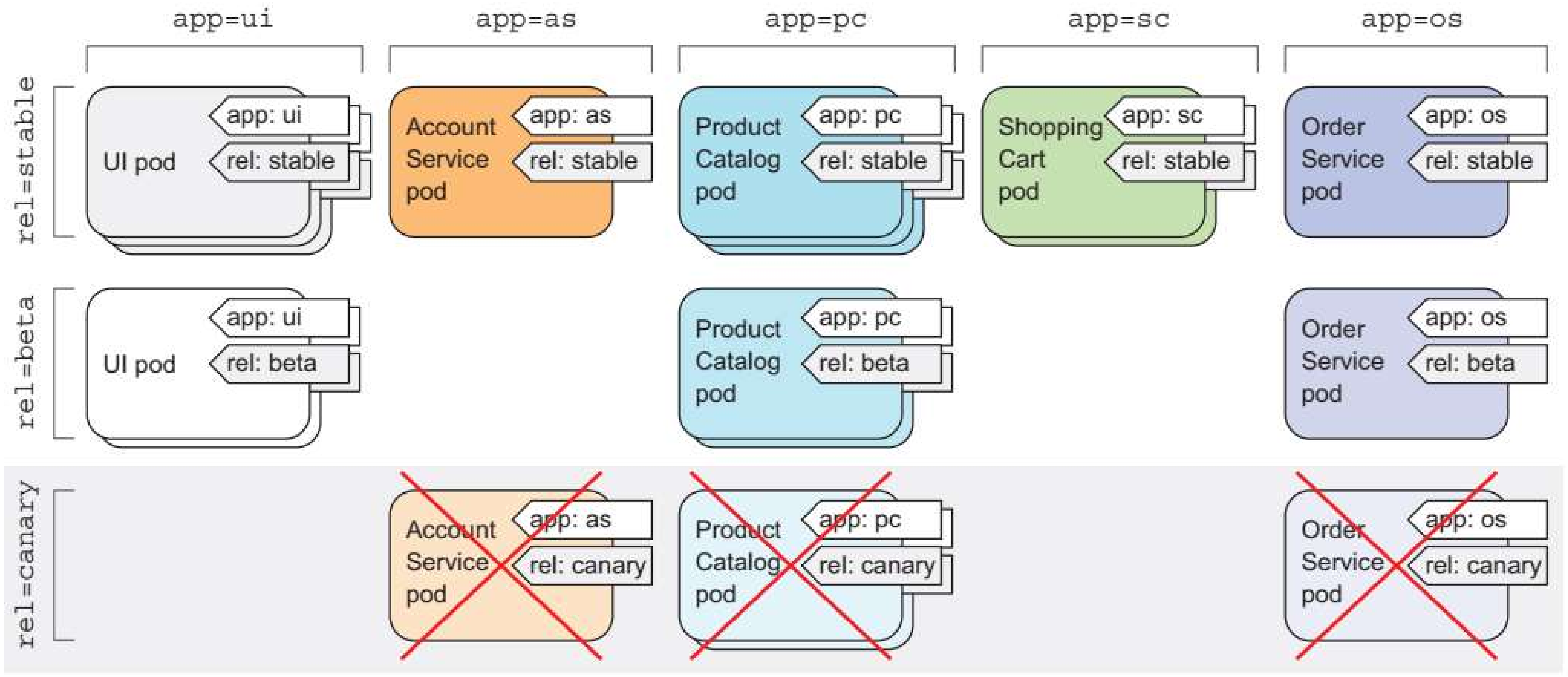


Figure 3.10 Selecting and deleting all canary pods through the `rel=canary` label selector

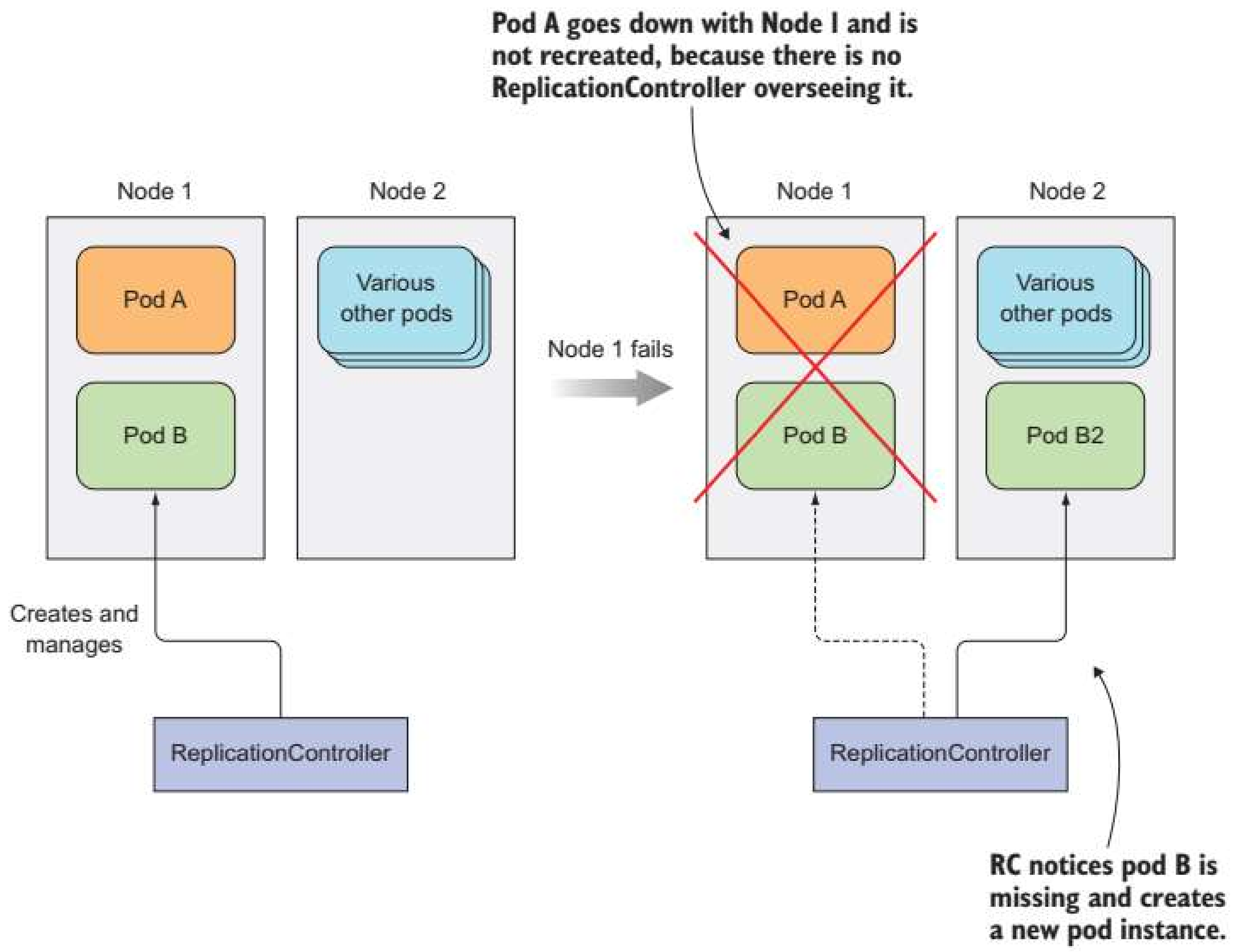

Pod A goes down with Node I and is not recreated, because there is no ReplicationController overseeing it.
Node 1
Node 2
Pod A
Pod B
Various other pods
Creates and manages
ReplicationController
Node 1 fails
Node 1
Node 2
Pod A
Pod B
Various other pods
Pod B2
ReplicationController
RC notices pod B is missing and creates a new pod instance.

Figure 4.1 When a node fails, only pods backed by a ReplicationController are recreated.

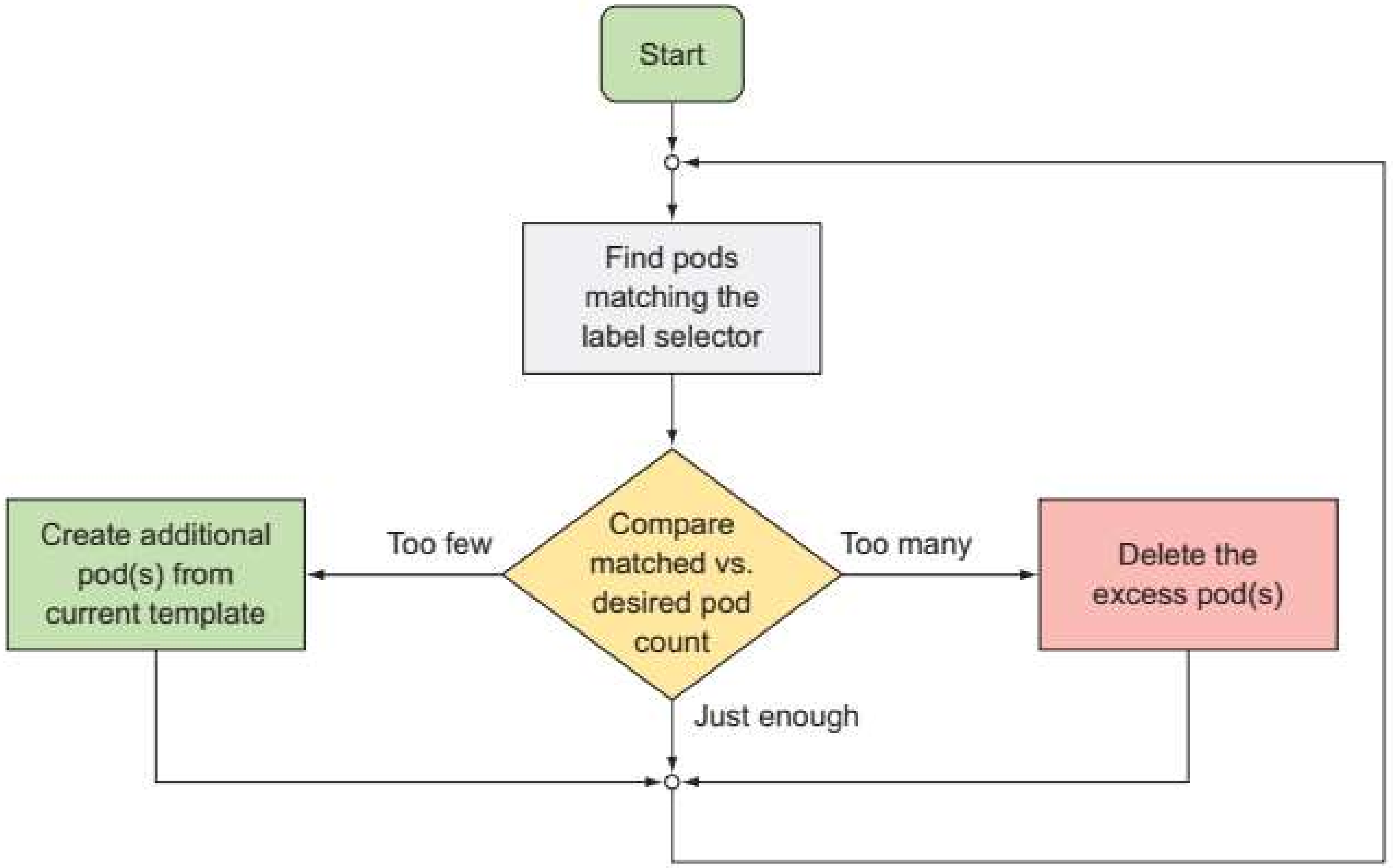

Start
Find pods matching the label selector
Compare matched vs. desired pod count
Too few
Create additional pod(s) from current template
Too many
Delete the excess pod(s)
Just enough

Figure 4.2 A ReplicationController's reconciliation loop

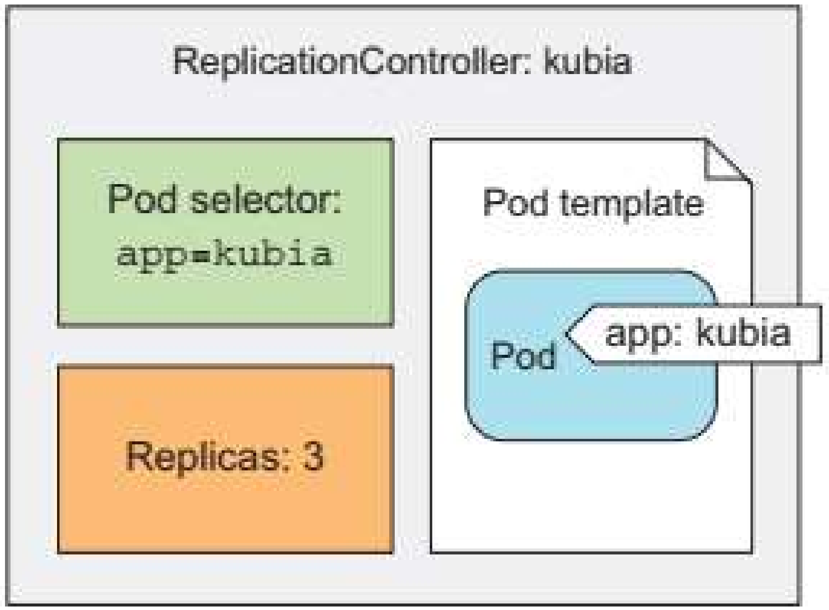


Figure 4.3 The three key parts of a ReplicationController (pod selector, replica count, and pod template)

Listing 4.4 A YAML definition of a ReplicationController: kubia-rc.yaml

```
apiVersion: v1
kind: ReplicationController
metadata:
  name: kubia
spec:
  replicas: 3
  selector:
    app: kubia
  template:
    metadata:
      labels:
        app: kubia
    spec:
      containers:
      - name: kubia
        image: luksa/kubia
        ports:
        - containerPort: 8080
```

This manifest defines a ReplicationController (RC)

The name of this ReplicationController

The desired number of pod instances

The pod selector determining what pods the RC is operating on

The pod template for creating new pods

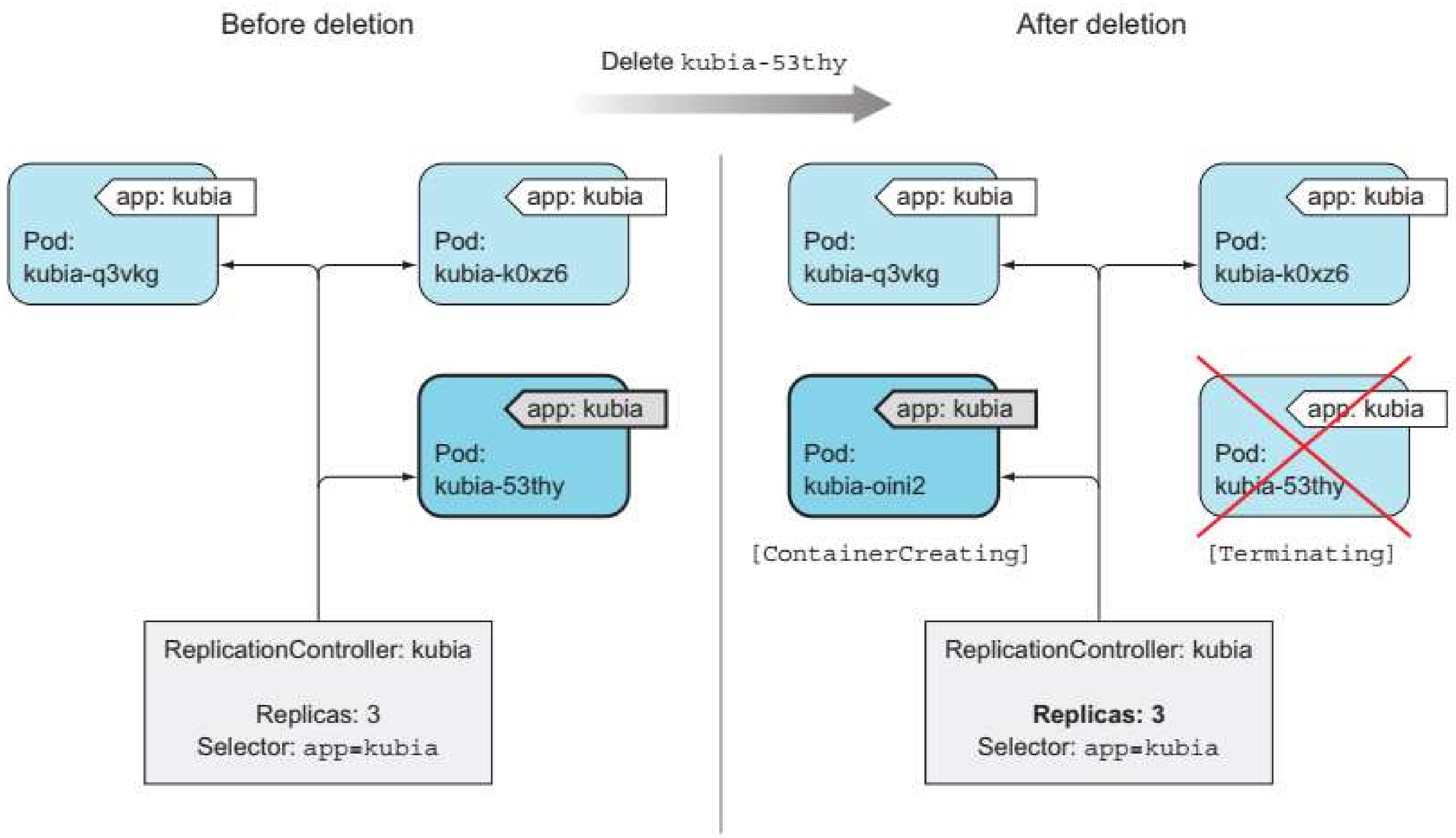


Figure 4.4 If a pod disappears, the ReplicationController sees too few pods and creates a new replacement pod.

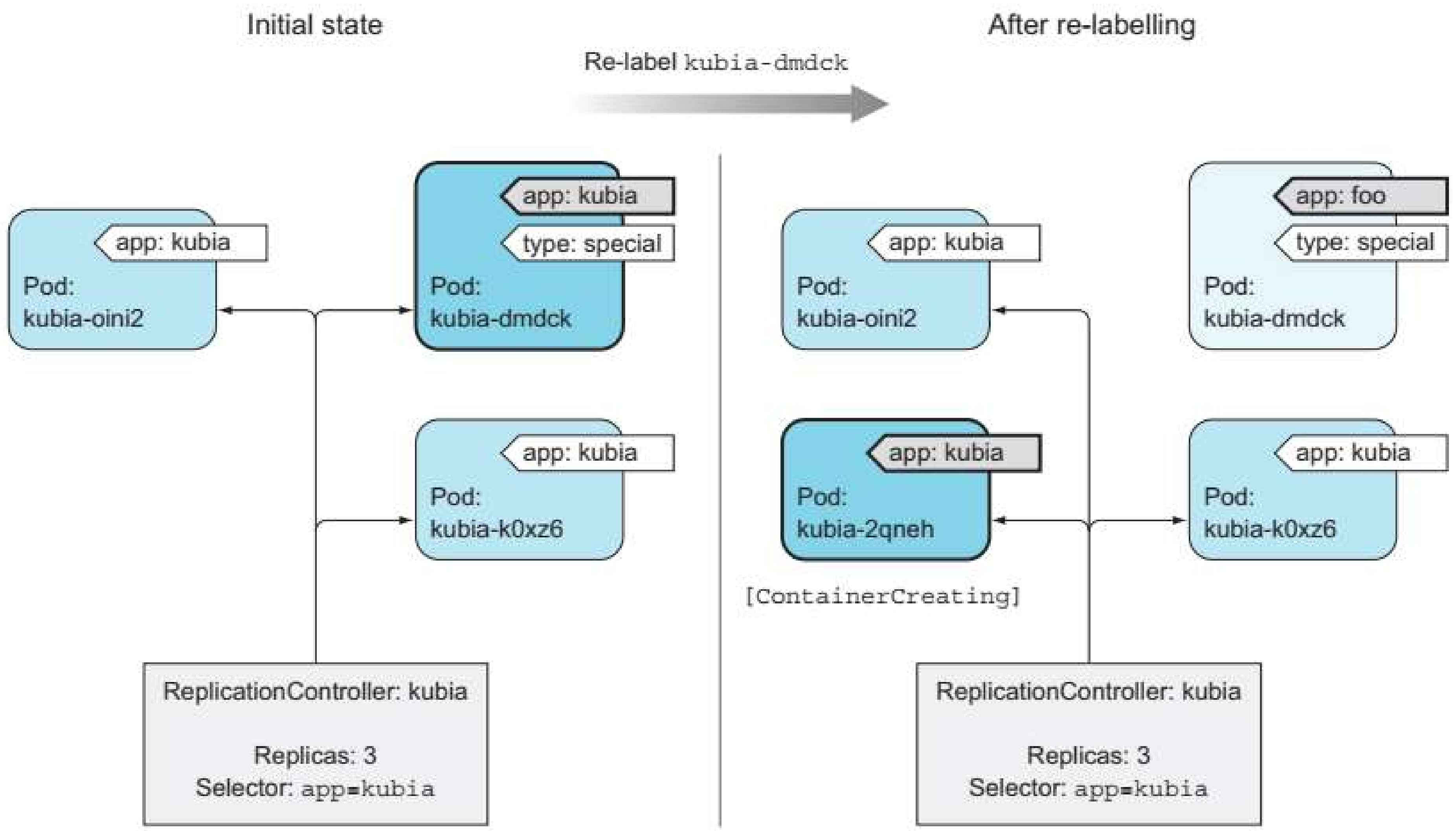
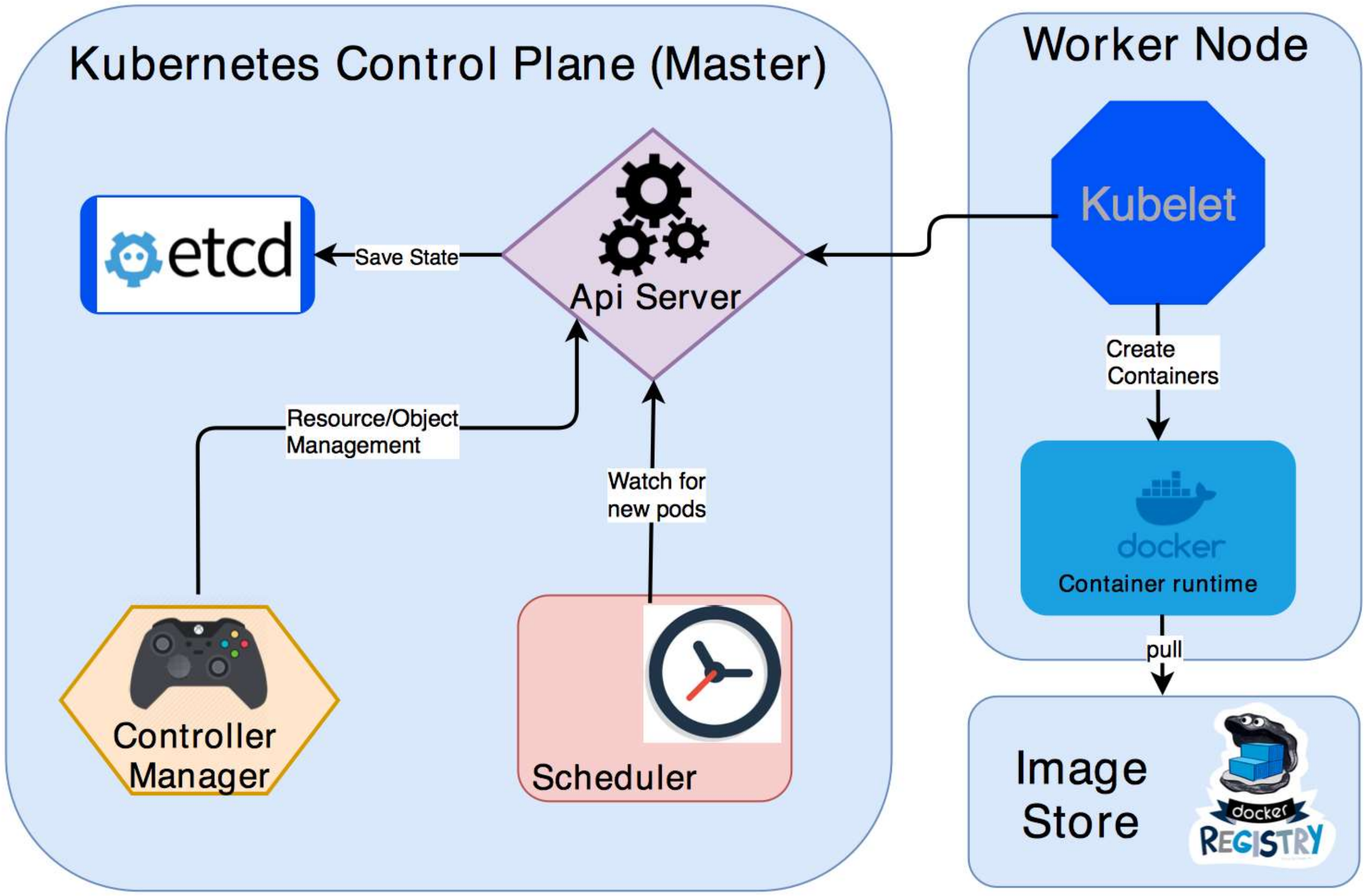


Figure 4.5 Removing a pod from the scope of a ReplicationController by changing its labels

Kubernetes Architecture



Thanks