@DEVOPSGIRLS

FROM DOCKER TO KUBERNETES

THE FIRST HALF: DOCKER

1. DOCKER COMMANDS
2. DOCKER IMAGES
3. DOCKER TAGS AND REGISTRIES
4. (OPTIONAL) SO HOW DOES IT WORK, REALLY?

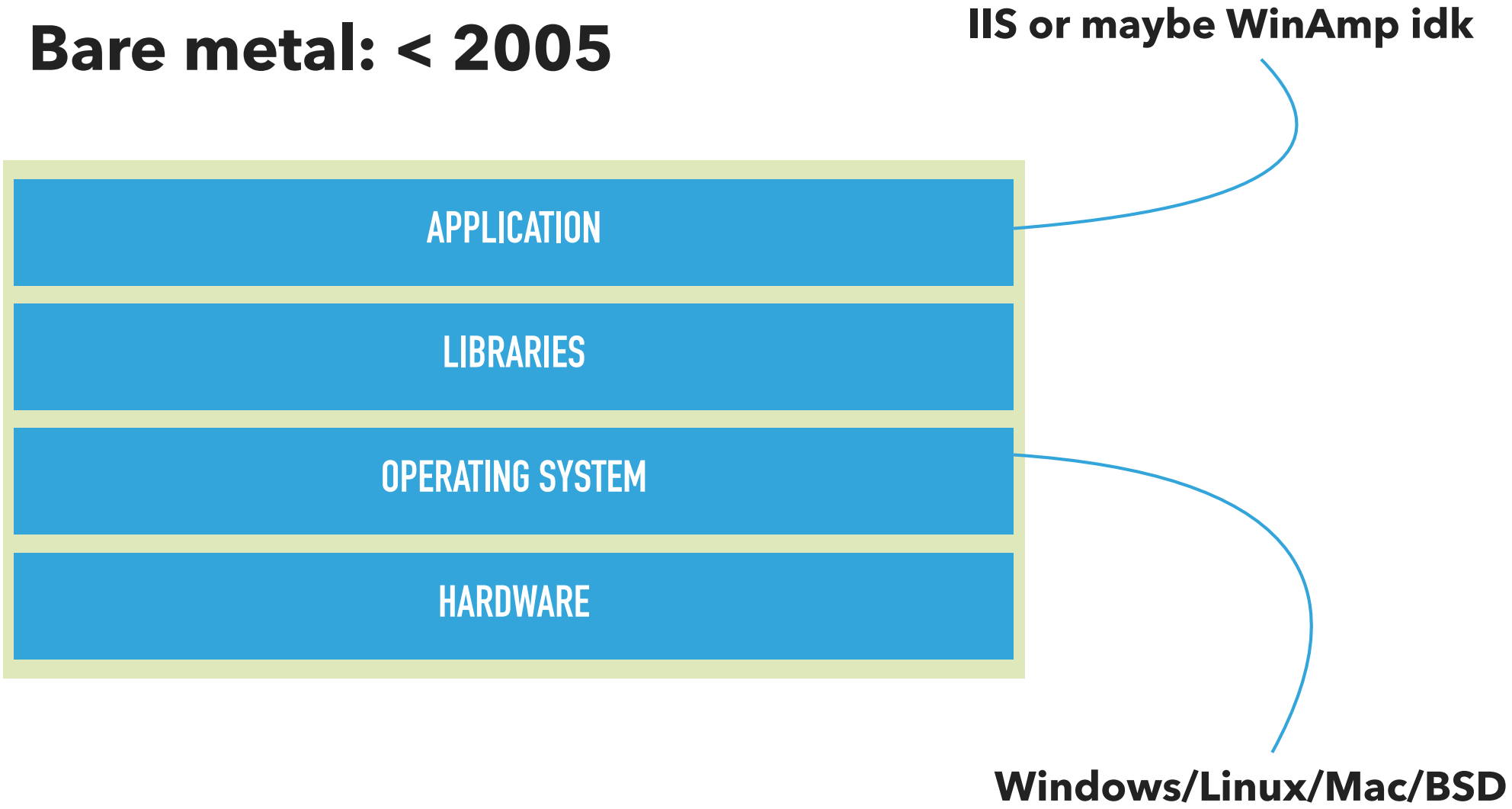
DOCKER

1. FOUNDED THROUGH YCOMBINATOR, 2011
2. DEBUTED AT PYCON, 2013
3. IS A SET OF PRODUCTS THAT USE VIRTUALIZATION TO DELIVER SOFTWARE IN PACKAGES CALLED **CONTAINERS**.

2004

- **16** years ago
- **podcast** became a new word.
- **Mean Girls** was released
- we deployed applications in "**servers**"

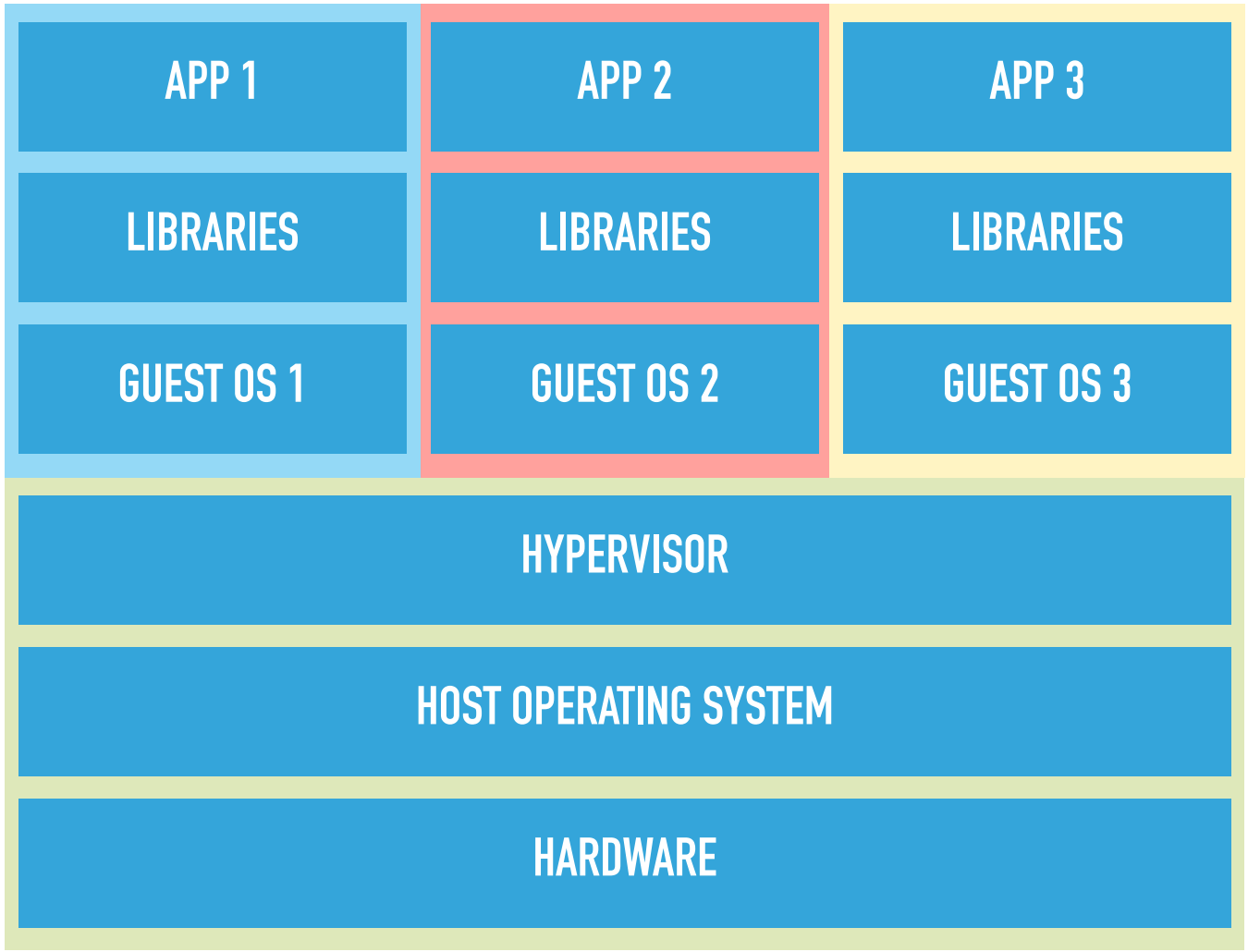
Bare metal: < 2005



Problems:

- Scaling means we have to build another one
- One app per machine

Hypervisors: 2005-onwards

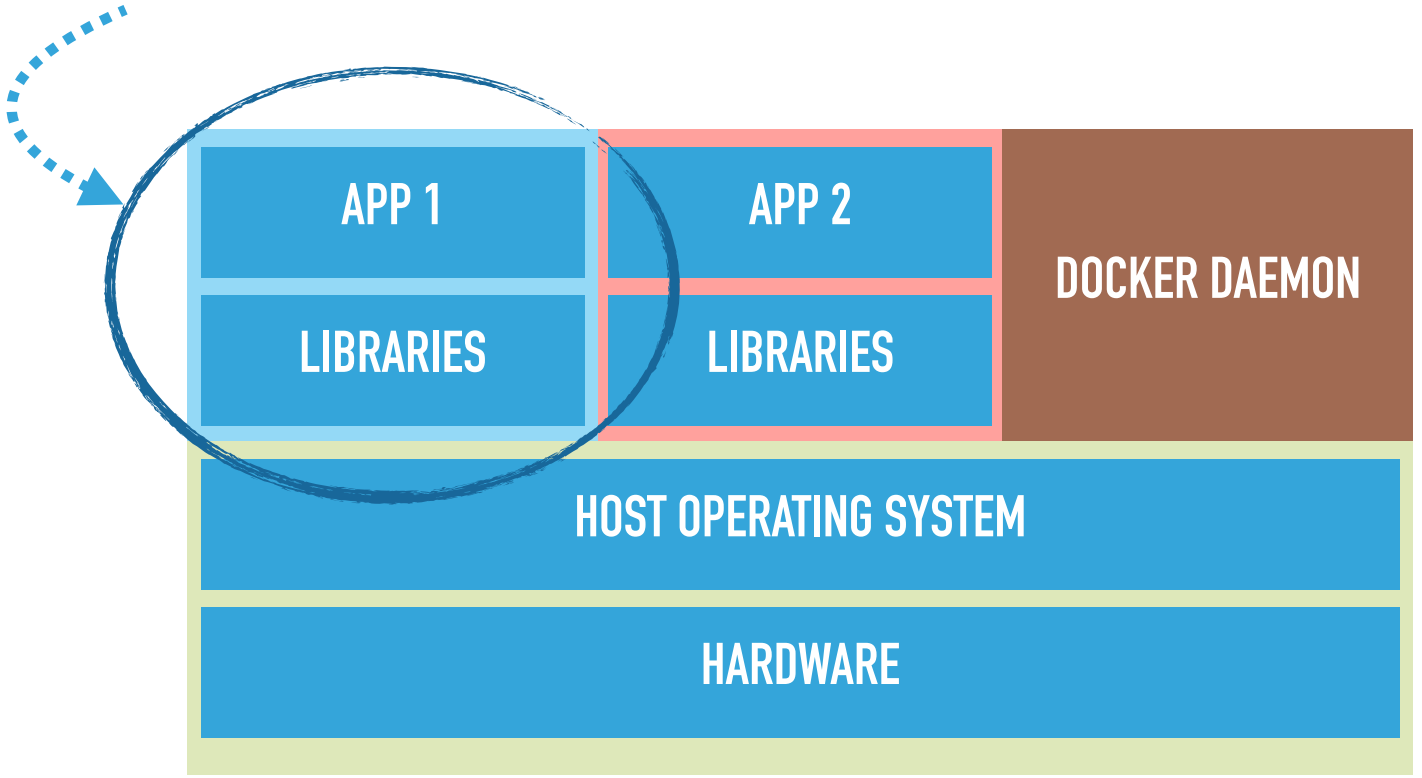


Good: Hardware is abstracted away

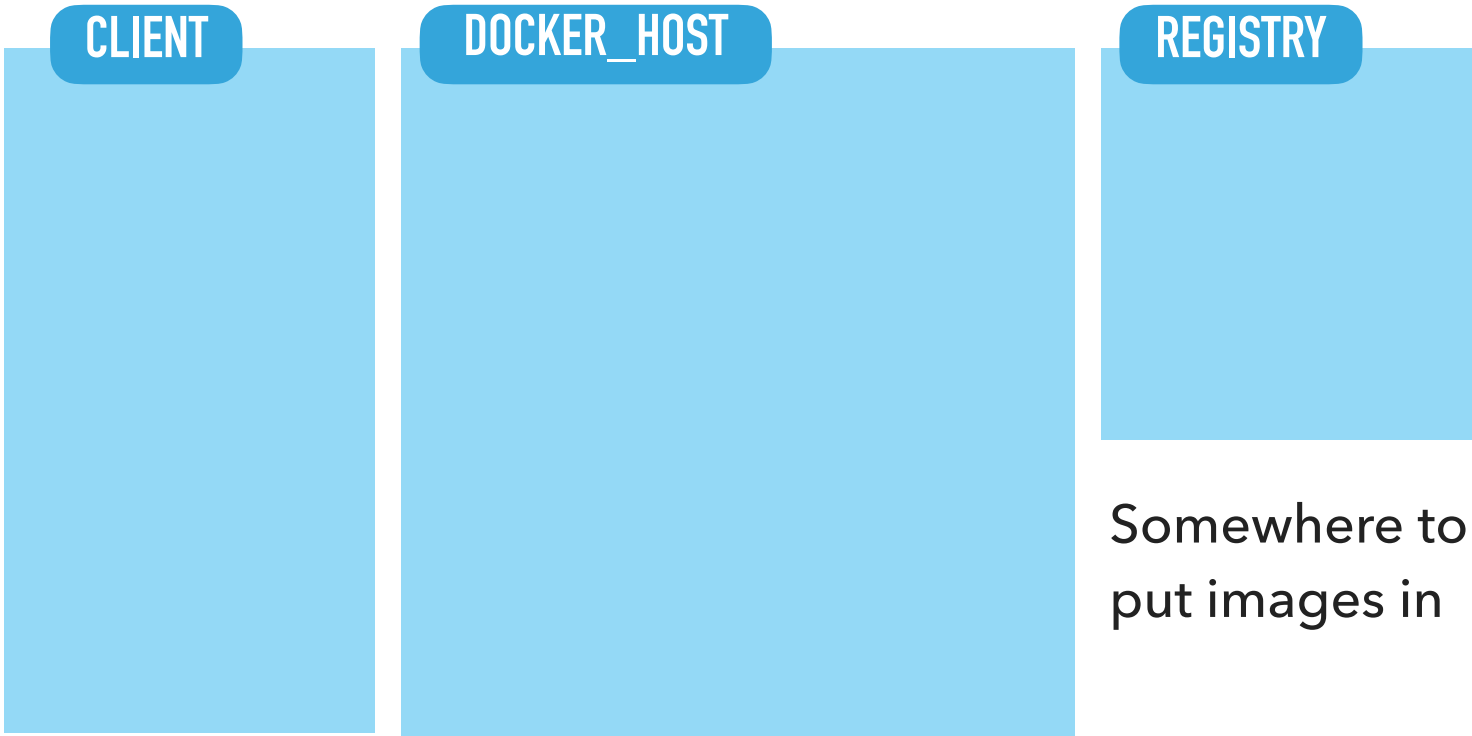
Bad: Software drift

Docker: 2013-Onwards

All you need to worry about!



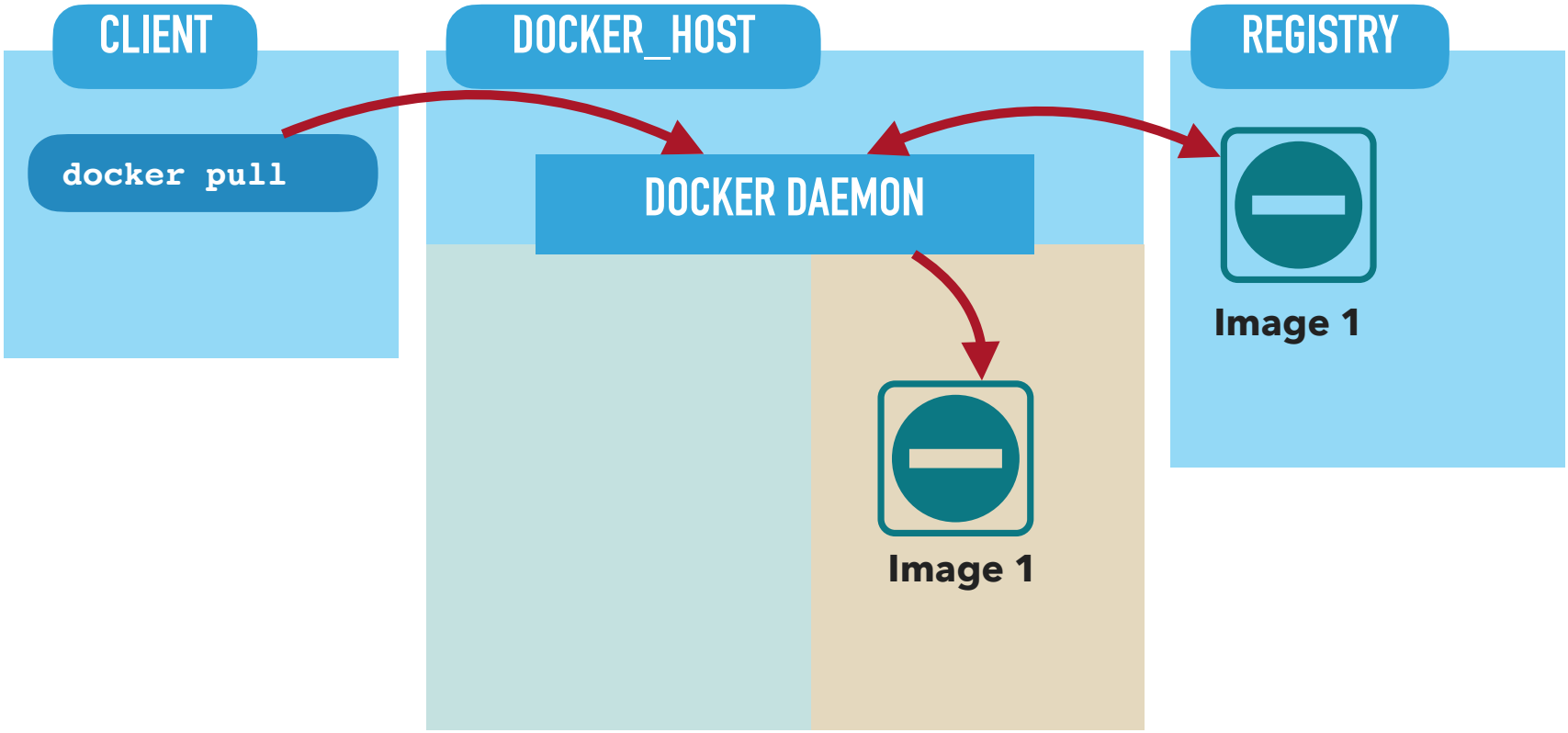
The Docker Ecosystem

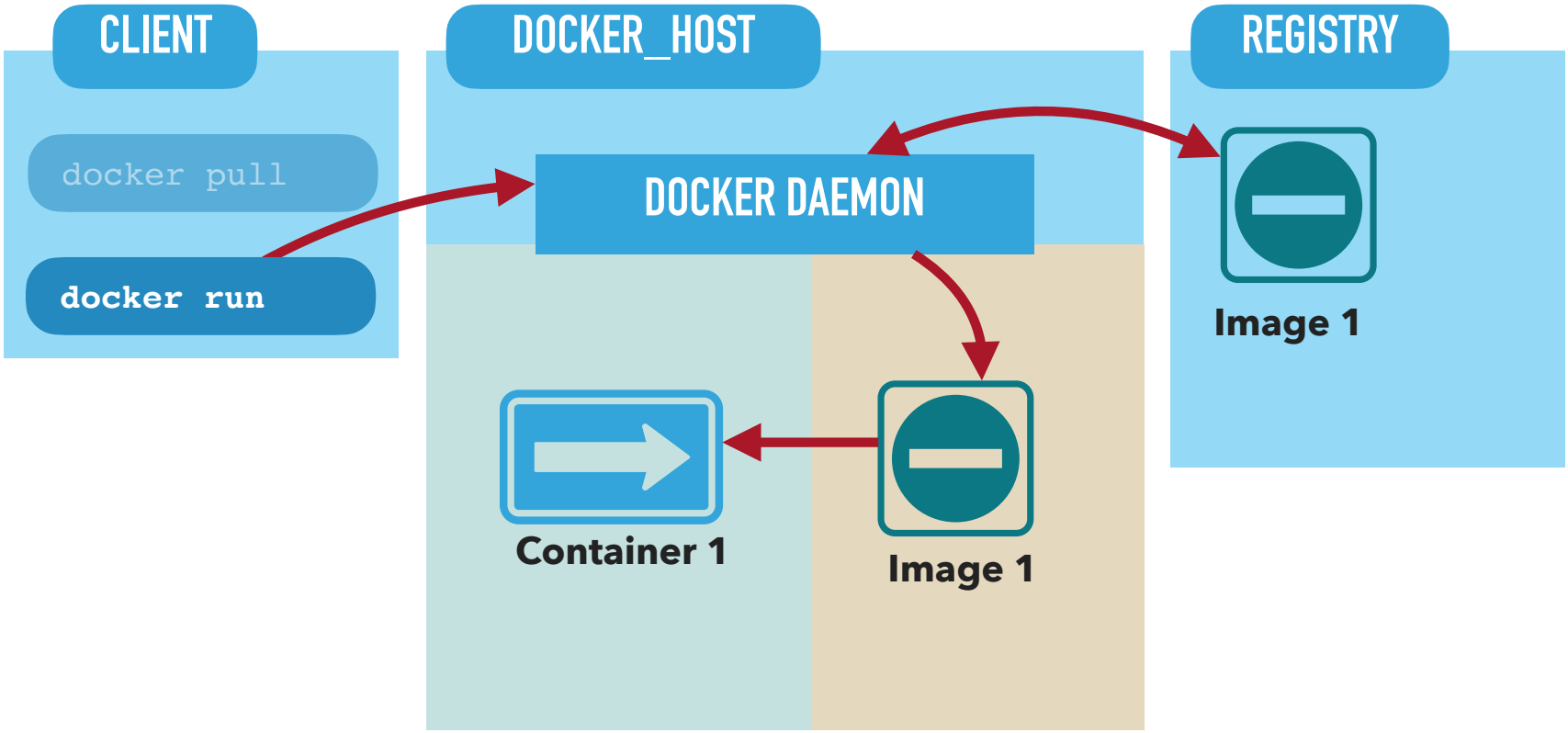


Your CLI

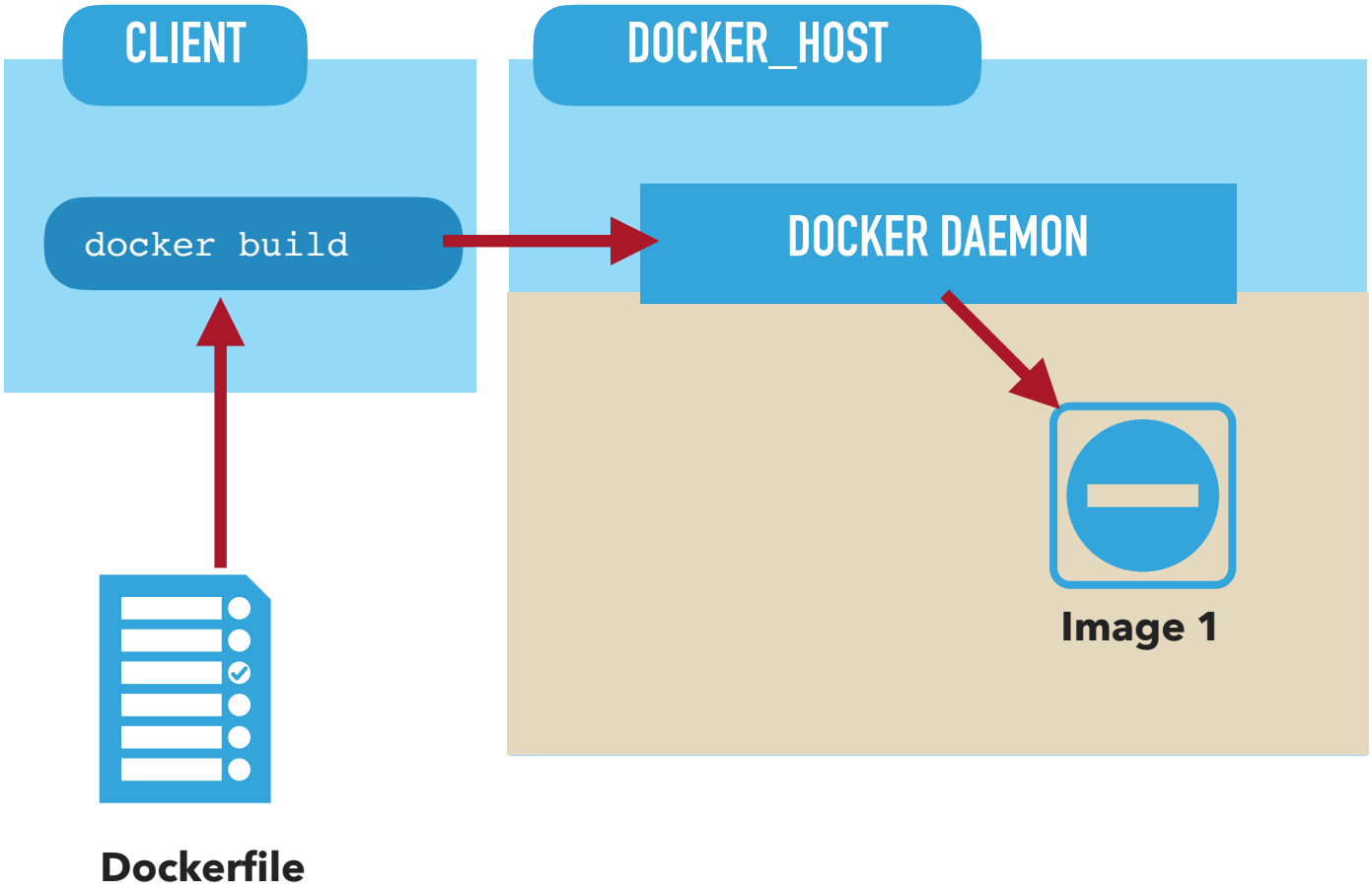
A machine running the Docker daemon (default: your machine)

Somewhere to put images in


CLIENT
DOCKER_HOST
REGISTRY
docker pull
DOCKER DAEMON
Image 1
Image 1


CLIENT
docker pull
docker run
DOCKER_HOST
DOCKER DAEMON
Container 1
Image 1
REGISTRY
Image 1

Part Deux: Dockerfiles


CLIENT
DOCKER_HOST
docker build
DOCKER DAEMON
Image 1
Dockerfile

FROM nginx:latest
RUN apt-get update
RUN apt-get -y install vim
COPY daisy.jpg /var/www/html/daisy.jpg
CMD echo "I love daisies"

CMD echo "I love daisies"

COPY daisy.jpg /var/www/html/daisy.jpg

RUN apt-get -y install vim

RUN apt-get update

FROM nginx:latest

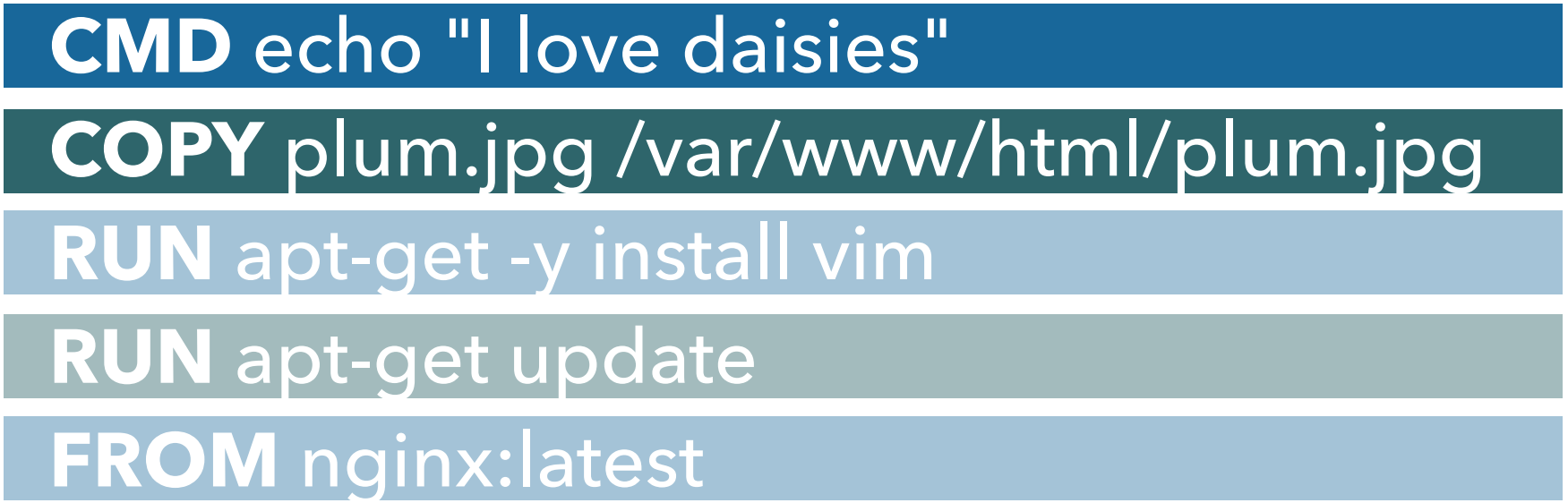
CMD echo "I love hibiscus"

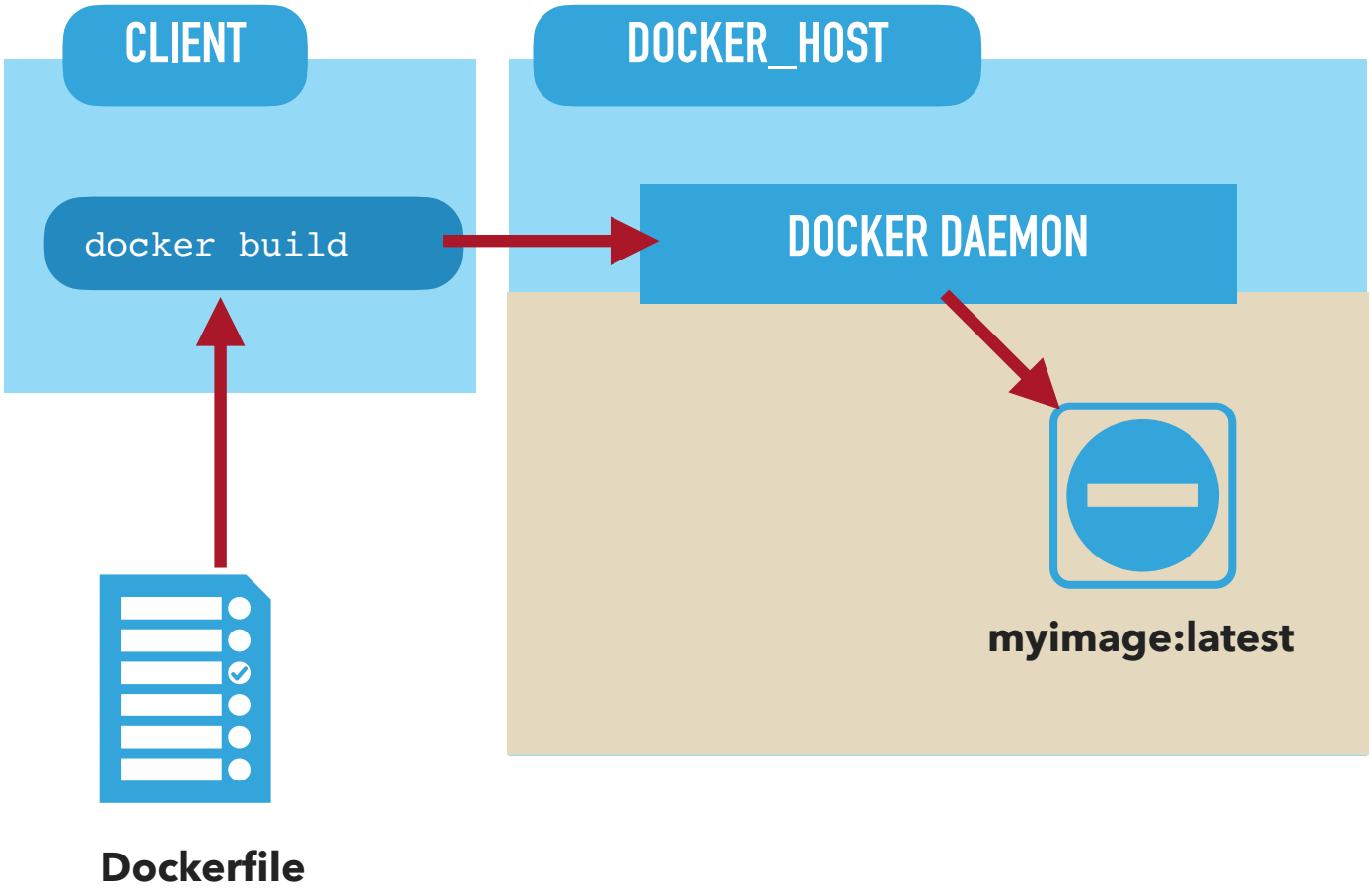
COPY daisy.jpg /var/www/html/daisy.jpg

RUN apt-get -y install vim

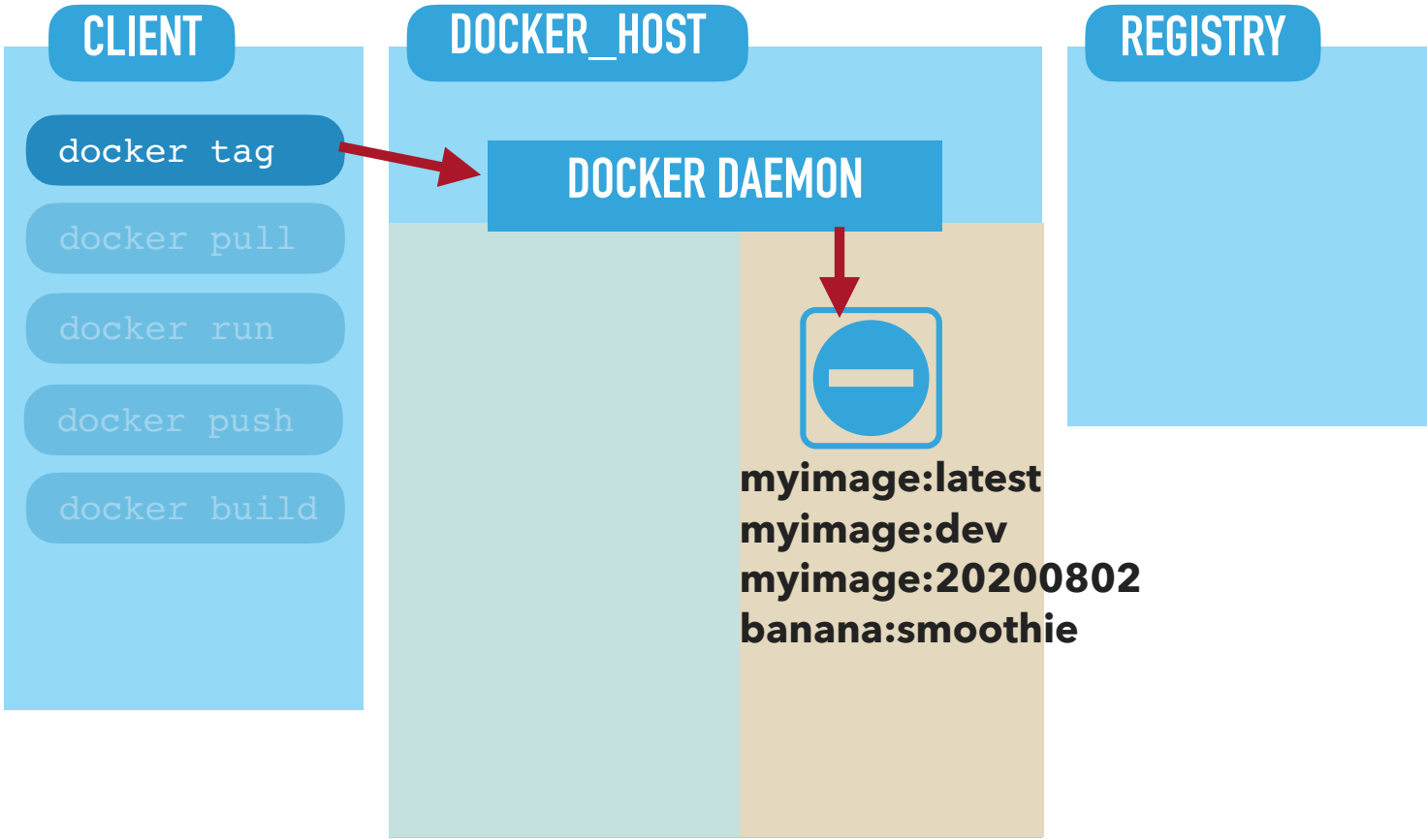
RUN apt-get update

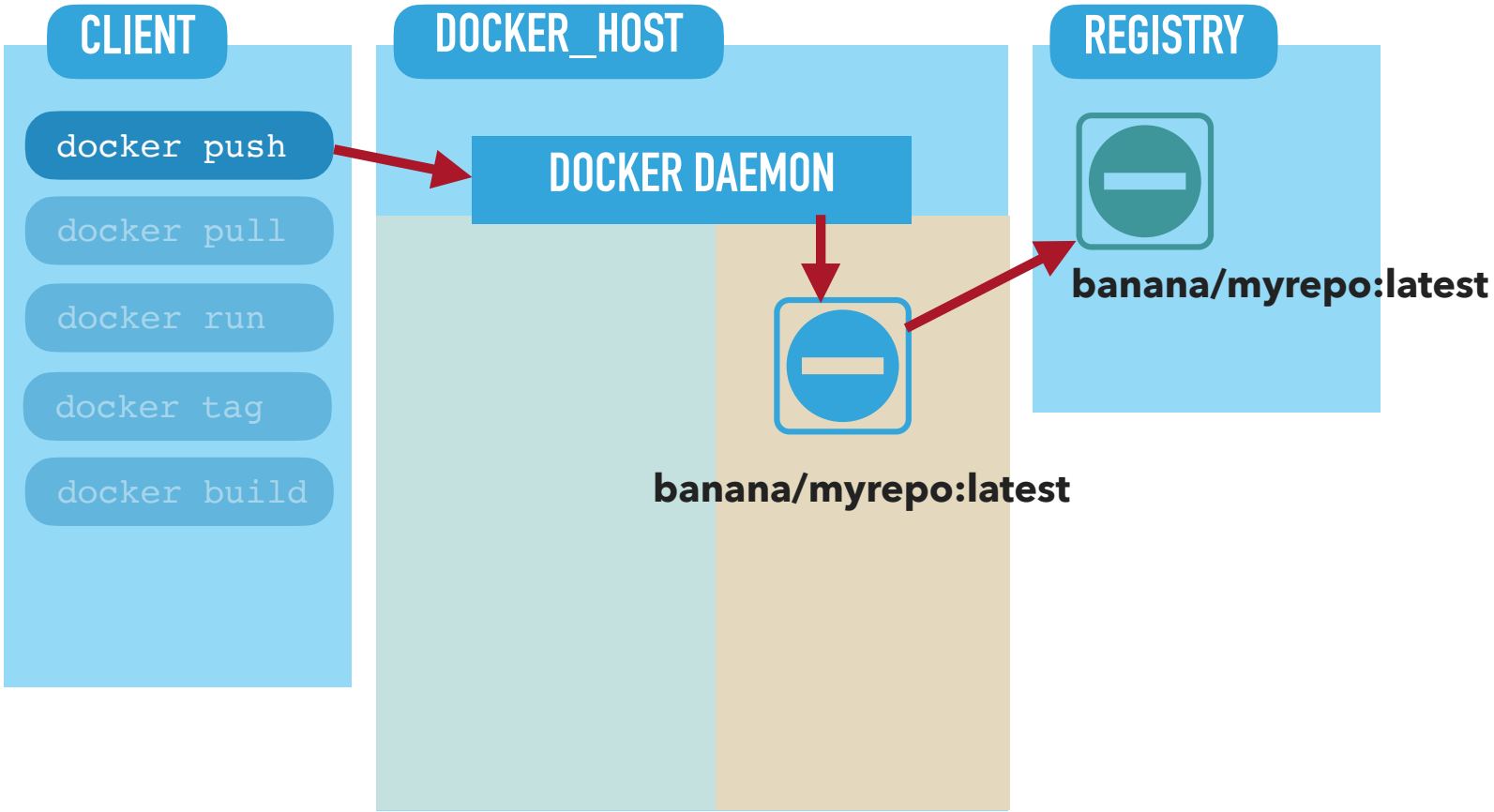
FROM nginx:latest


CMD echo "I love daisies"
COPY plum.jpg /var/www/html/plum.jpg
RUN apt-get -y install vim
RUN apt-get update
FROM nginx:latest


CLIENT
DOCKER_HOST
docker build
DOCKER DAEMON
myimage:latest
Dockerfile

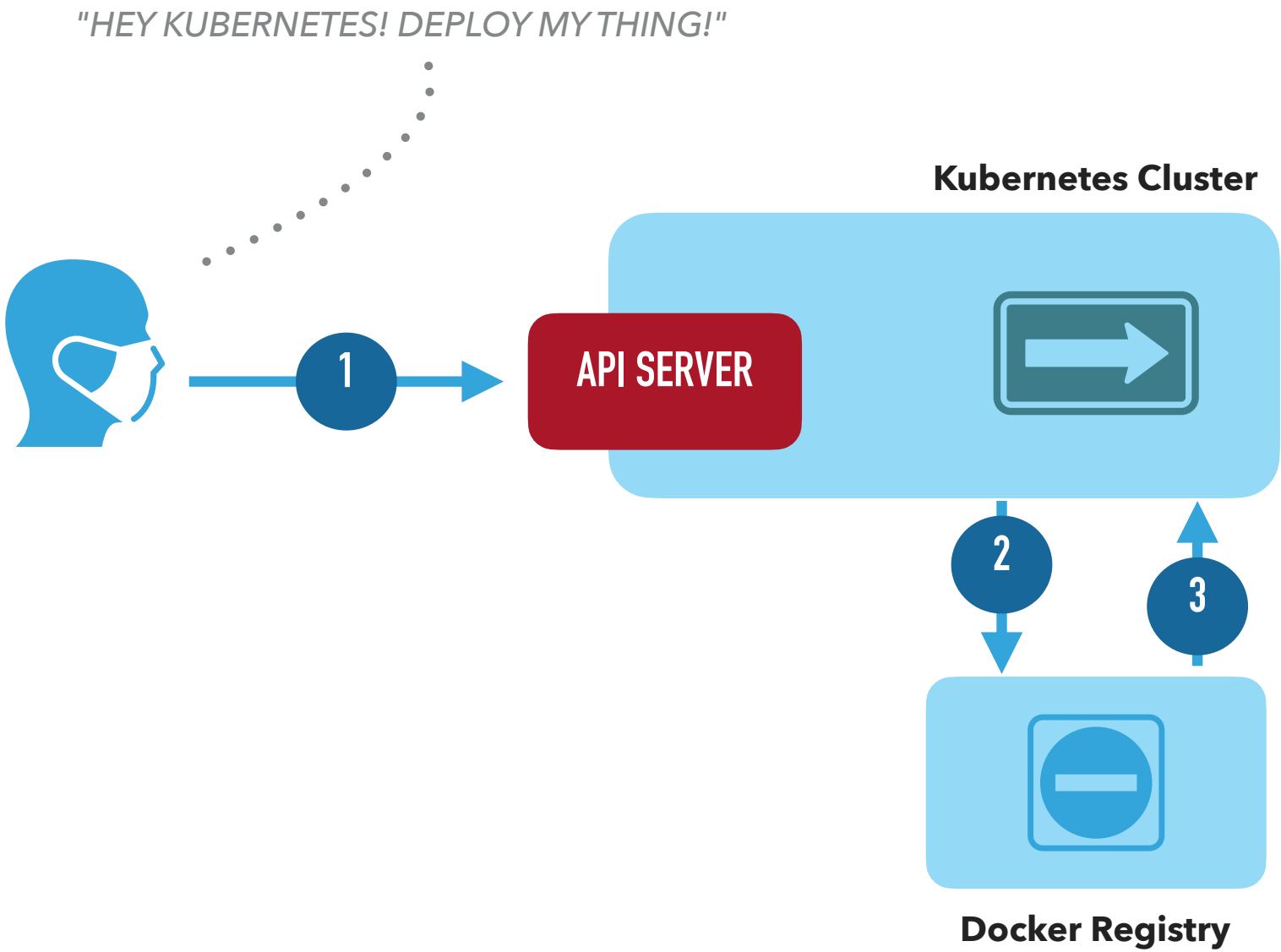
Part 3: Tags and Pushing

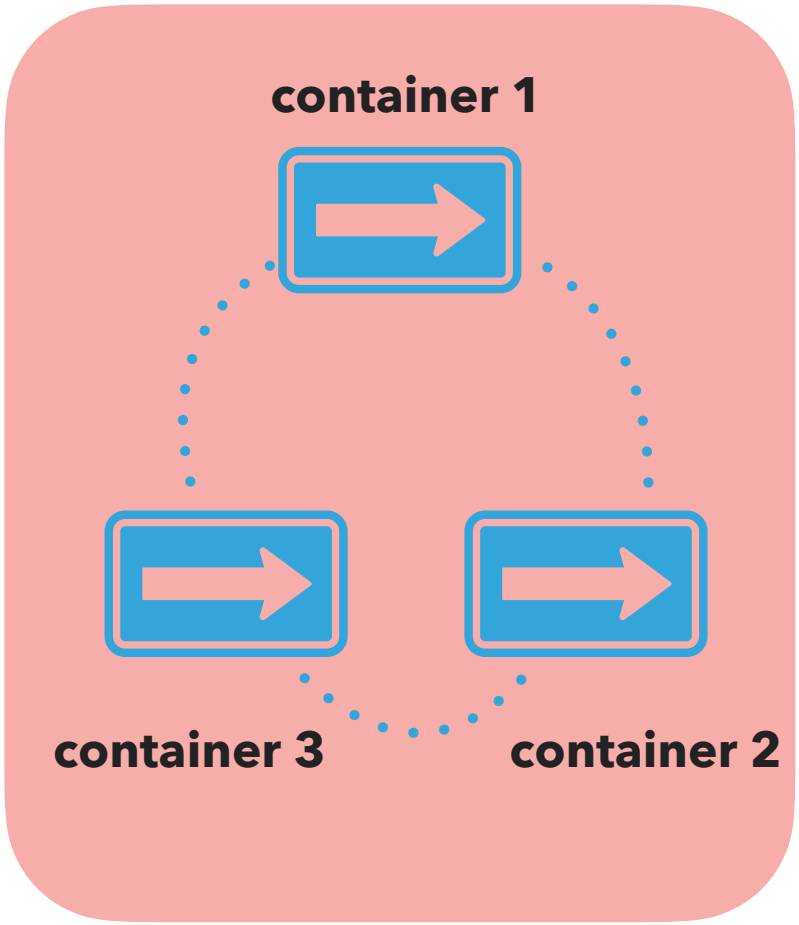

CLIENT
DOCKER_HOST
REGISTRY
docker tag
docker pull
docker run
docker push
docker build
DOCKER DAEMON
myimage:latest
myimage:dev
myimage:20200802
banana:smoothie


CLIENT
DOCKER_HOST
REGISTRY
docker push
docker pull
docker run
docker tag
docker build
DOCKER DAEMON
banana/myrepo:latest
banana/myrepo:latest

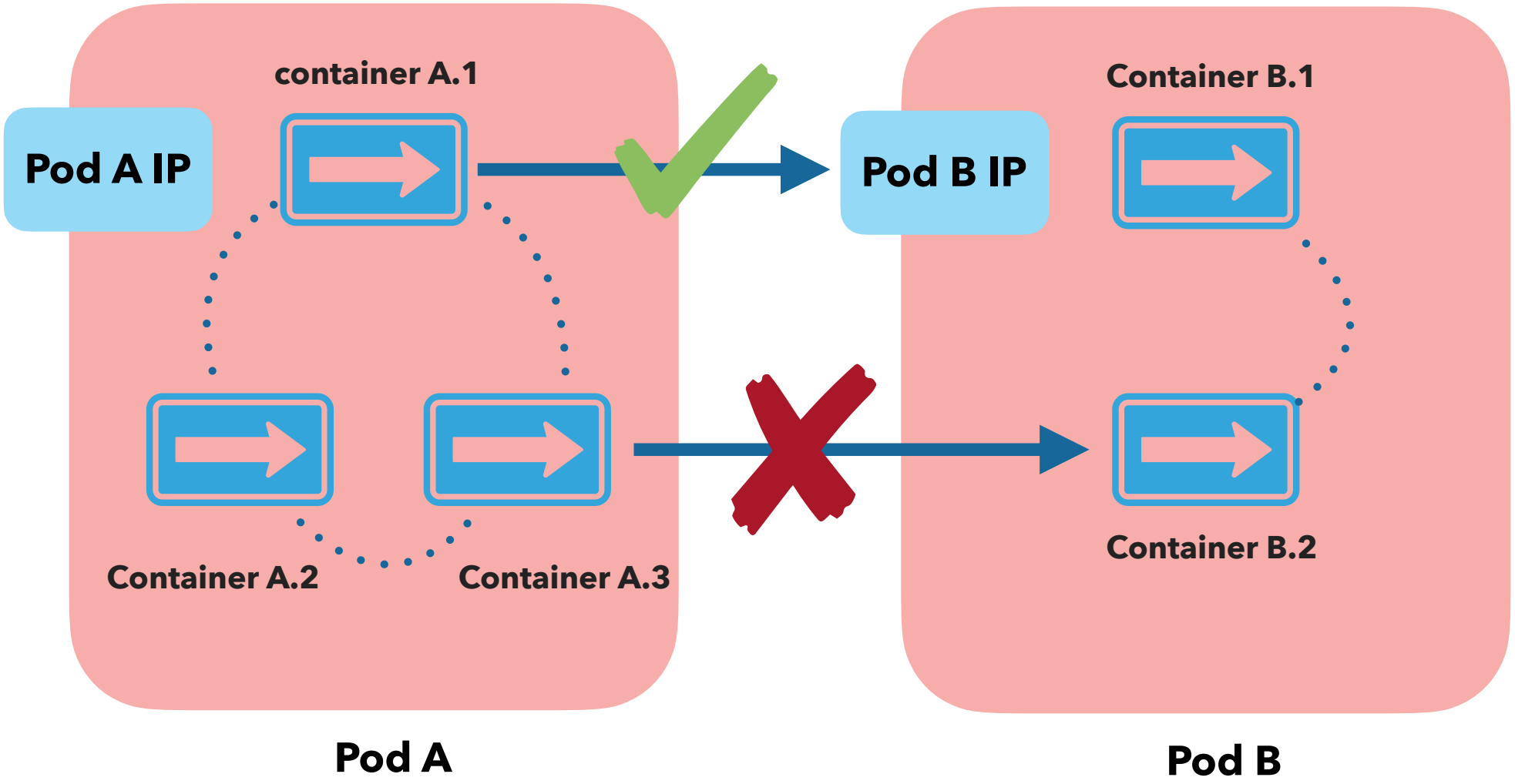
THE SECOND HALF: KUBERNETES

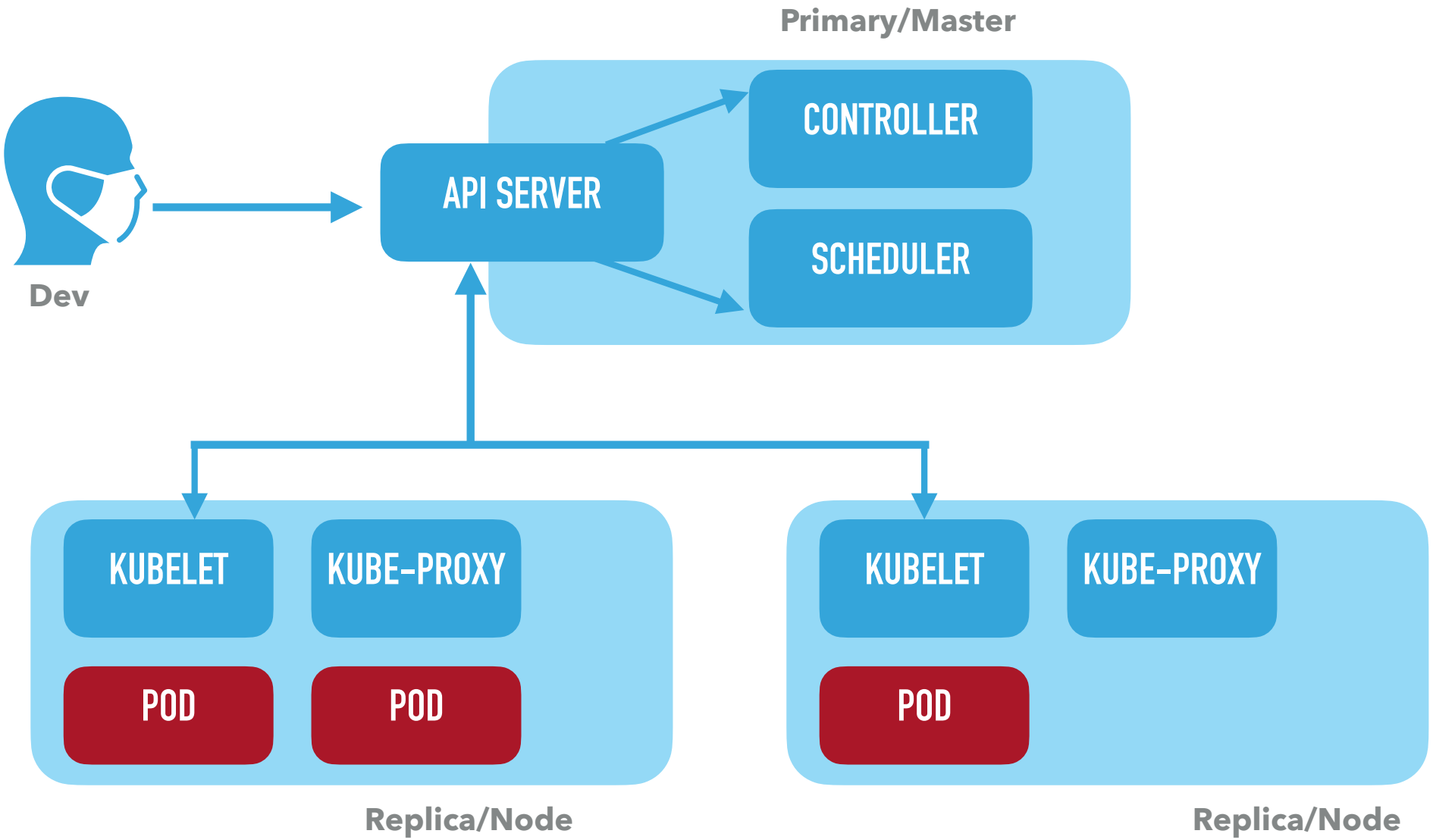
1. KUBERNETES COMMANDS AND PODS
2. SERVICES AND DEPLOYMENTS
3. LABELS
4. (OPTIONAL) SO HOW DOES IT WORK, REALLY?

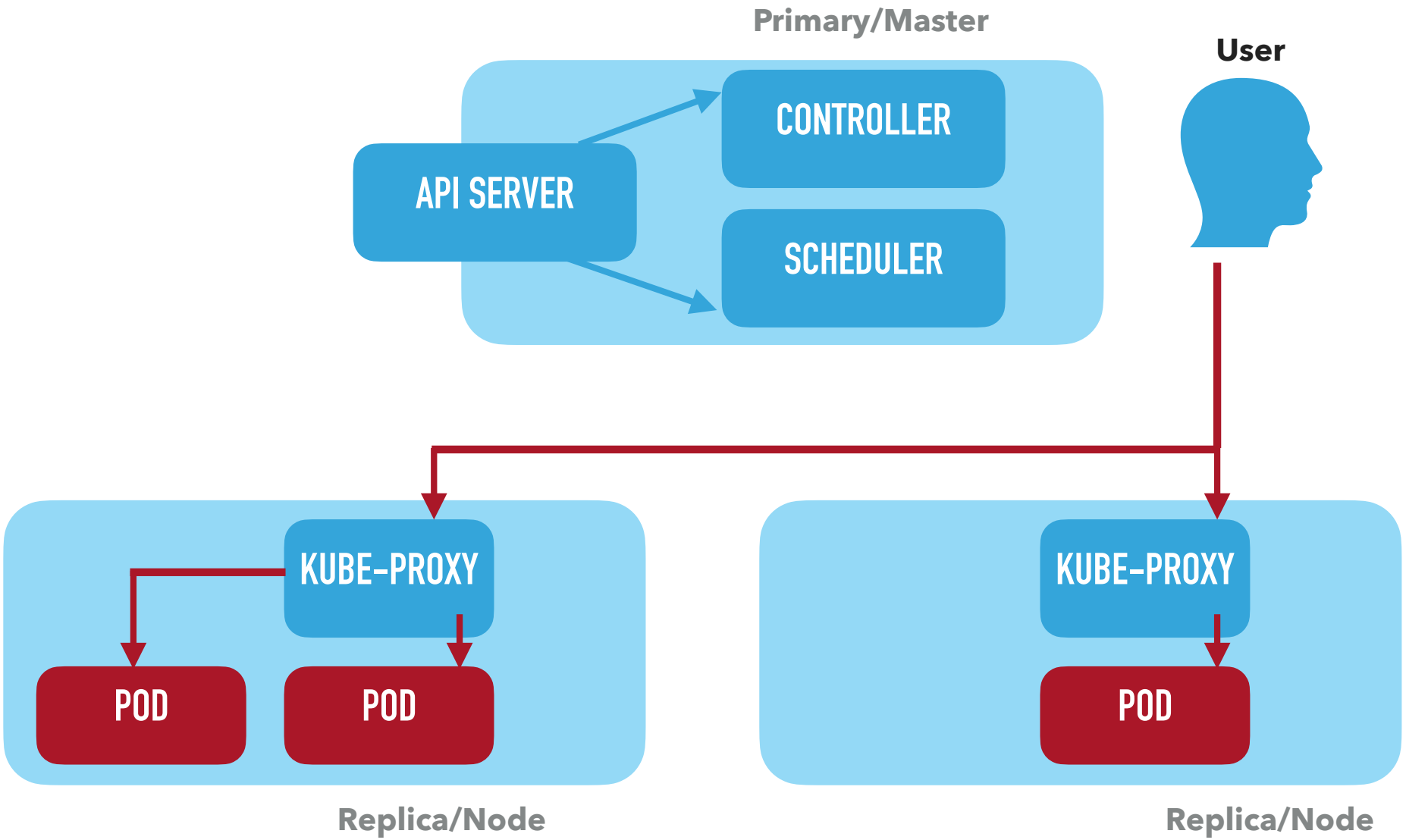

"HEY KUBERNETES! DEPLOY MY THING!"
Kubernetes Cluster
API SERVER
1
2
3
Docker Registry

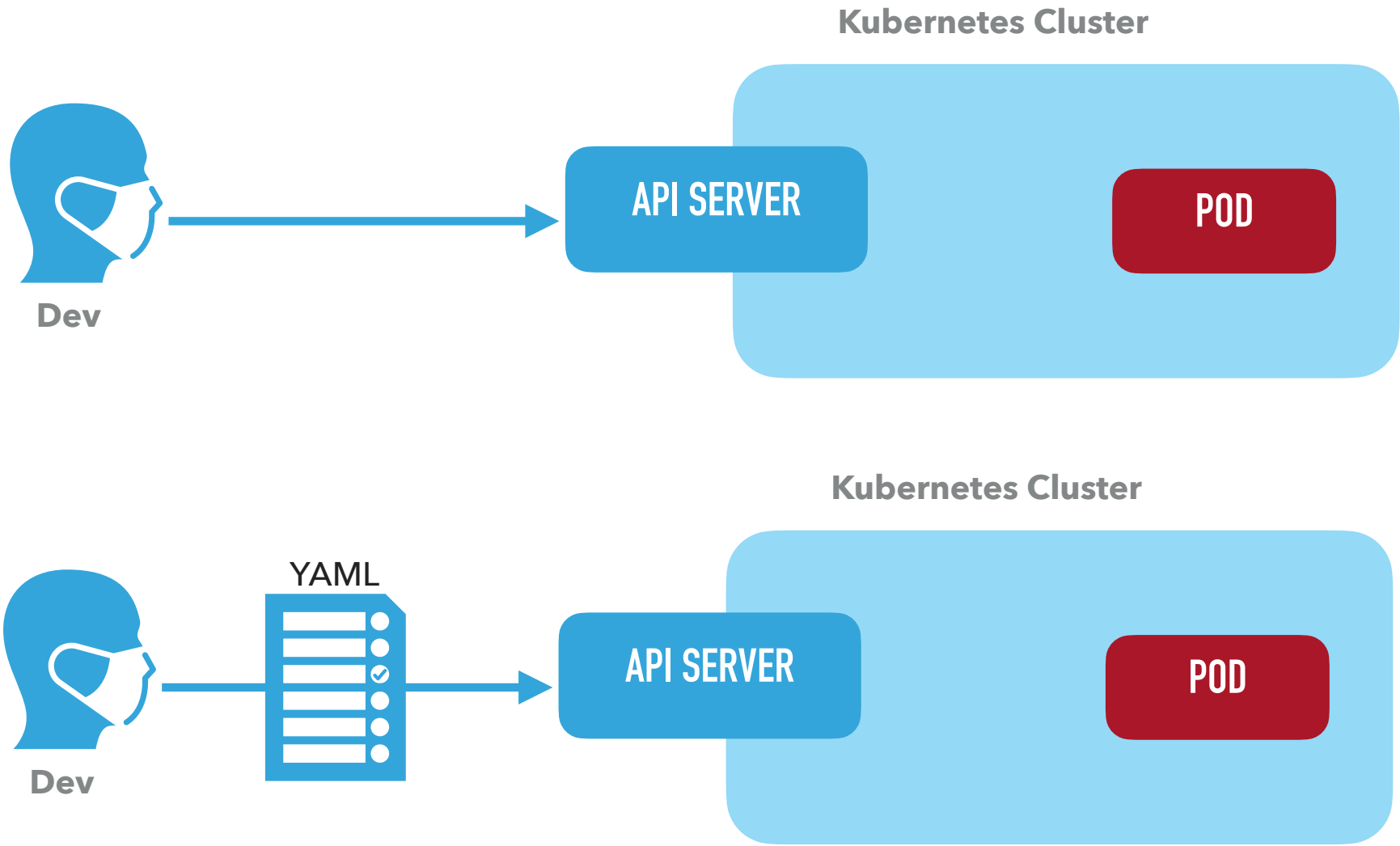


Pod

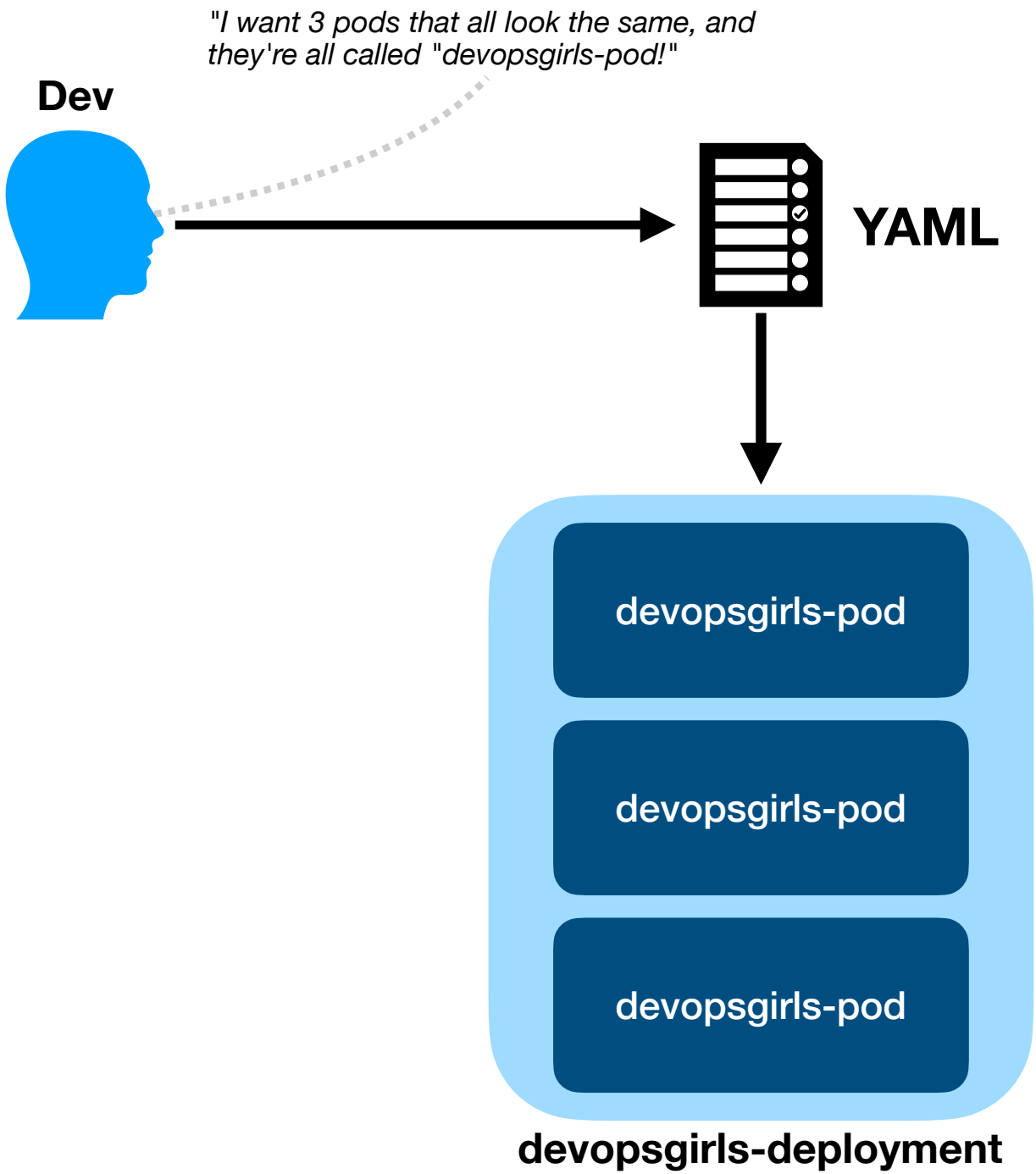

container A.1
Pod A IP
Container A.2
Container A.3
Pod A
Container B.1
Pod B IP
Container B.2
Pod B


Primary/Master
CONTROLLER
API SERVER
SCHEDULER
Dev
KUBELET
KUBE-PROXY
POD
POD
KUBELET
KUBE-PROXY
POD
Replica/Node
Replica/Node

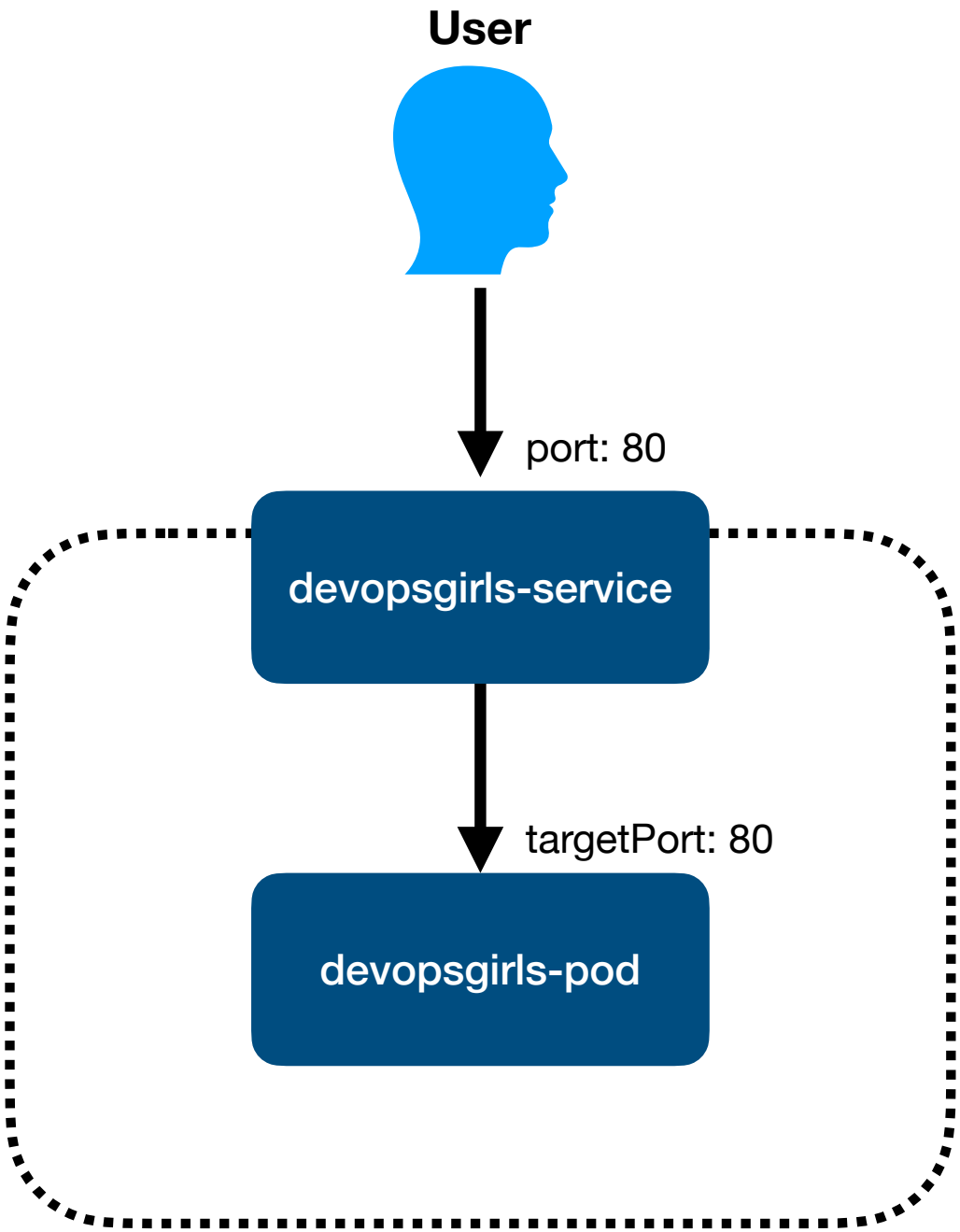

Primary/Master
User
CONTROLLER
API SERVER
SCHEDULER
KUBE-PROXY
KUBE-PROXY
POD
POD
POD
Replica/Node
Replica/Node


Kubernetes Cluster
API SERVER
POD
Dev
Kubernetes Cluster
YAML
API SERVER
POD
Dev

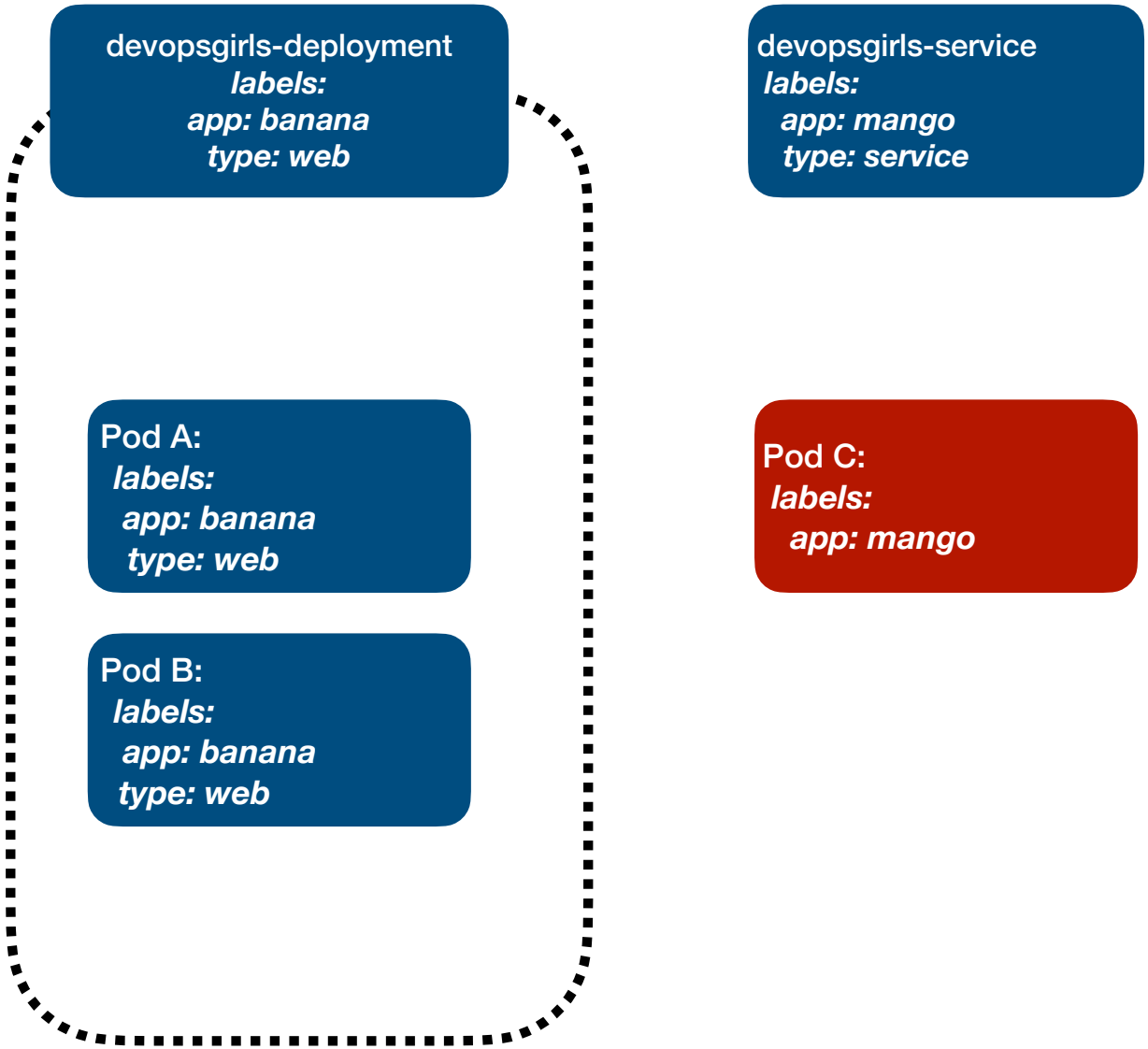
```
---
apiVersion: v1
kind: Pod
spec:
  containers:
    - image: "nginx:alpine"
      name: nginx
      ports:
         - containerPorts: 80
```


"I want 3 pods that all look the same, and they're all called "devopsgirls-pod!"
Dev
YAML
devopsgirls-pod
devopsgirls-pod
devopsgirls-pod
devopsgirls-deployment

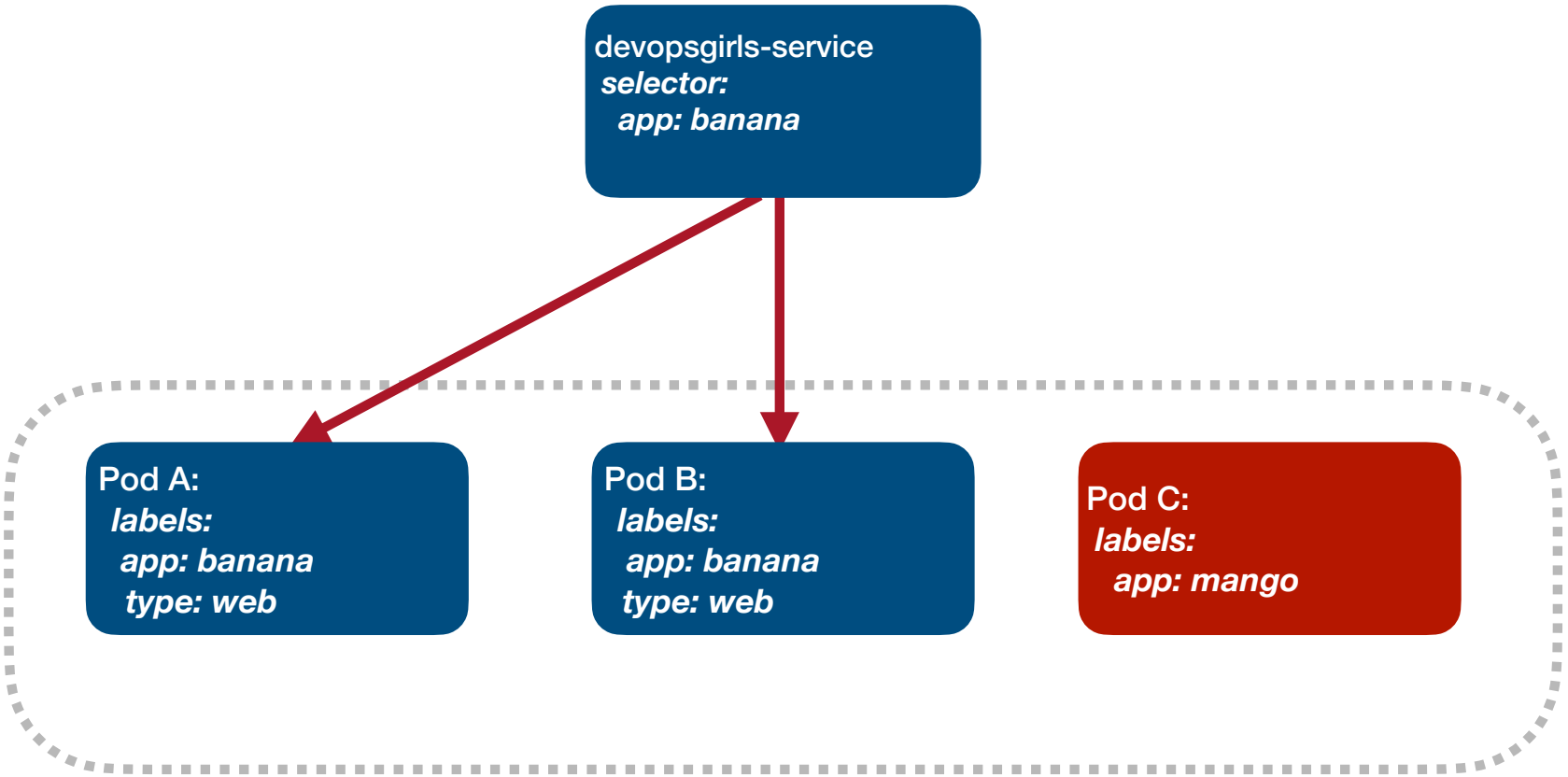
```
kind: Deployment
metadata:
  name: "devopsgirls-deployment"
spec:
  replicas: 1
  template:
   spec:
      containers:
        - image: "nginx:alpine"
          name: nginx
          ports:
            - containerPort: 80
```


User
port: 80
devopsgirls-service
targetPort: 80
devopsgirls-pod

```
---
apiVersion: v1
kind: Service
metadata:
  name: "devopsgirls-service"
spec:
  ports:
    - port: 80
      targetPort: 80
  selector:
    app: "devopsgirls"
```


devopsgirls-deployment
labels:
app: banana
type: web
devopsgirls-service
labels:
app: mango
type: service
Pod A:
labels:
app: banana
type: web
Pod B:
labels:
app: banana
type: web
Pod C:
labels:
app: mango

```
kind: Deployment
metadata:
  name: "devopsgirls-deployment"
spec:
  replicas: 1
  template:
    metadata:
      labels:
        app: "devopsgirls"
    spec:
      containers:
        - image: "nginx:alpine"
          name: nginx
          ports:
            - containerPort: 80
```


devopsgirls-service
selector:
app: banana
Pod A:
labels:
app: banana
type: web
Pod B:
labels:
app: banana
type: web
Pod C:
labels:
app: mango

```
---
apiVersion: v1
kind: Service
metadata:
  name: "devopsgirls-service"
spec:
  ports:
    - port: 80
      targetPort: 80
  selector:
    app: "devopsgirls"
```