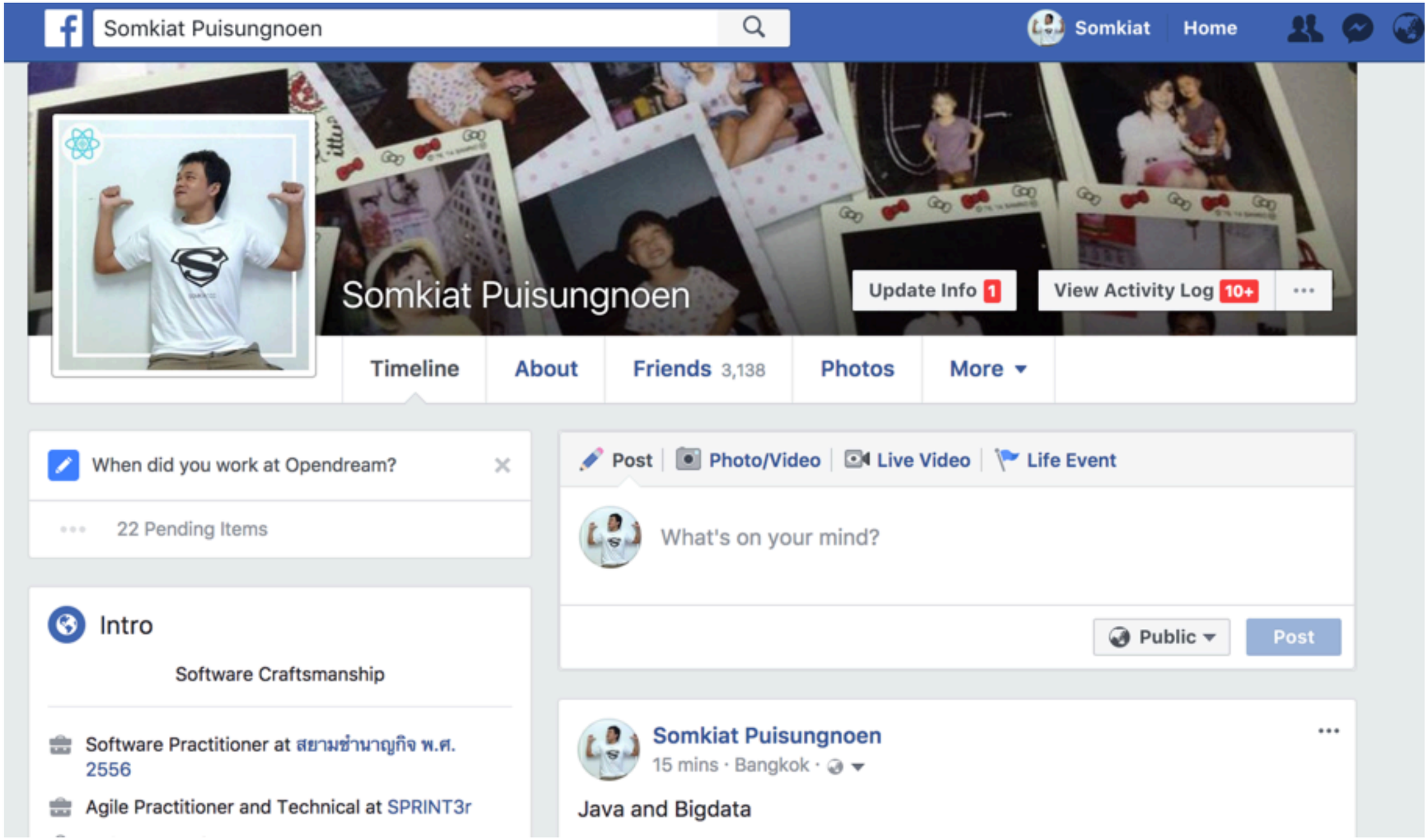
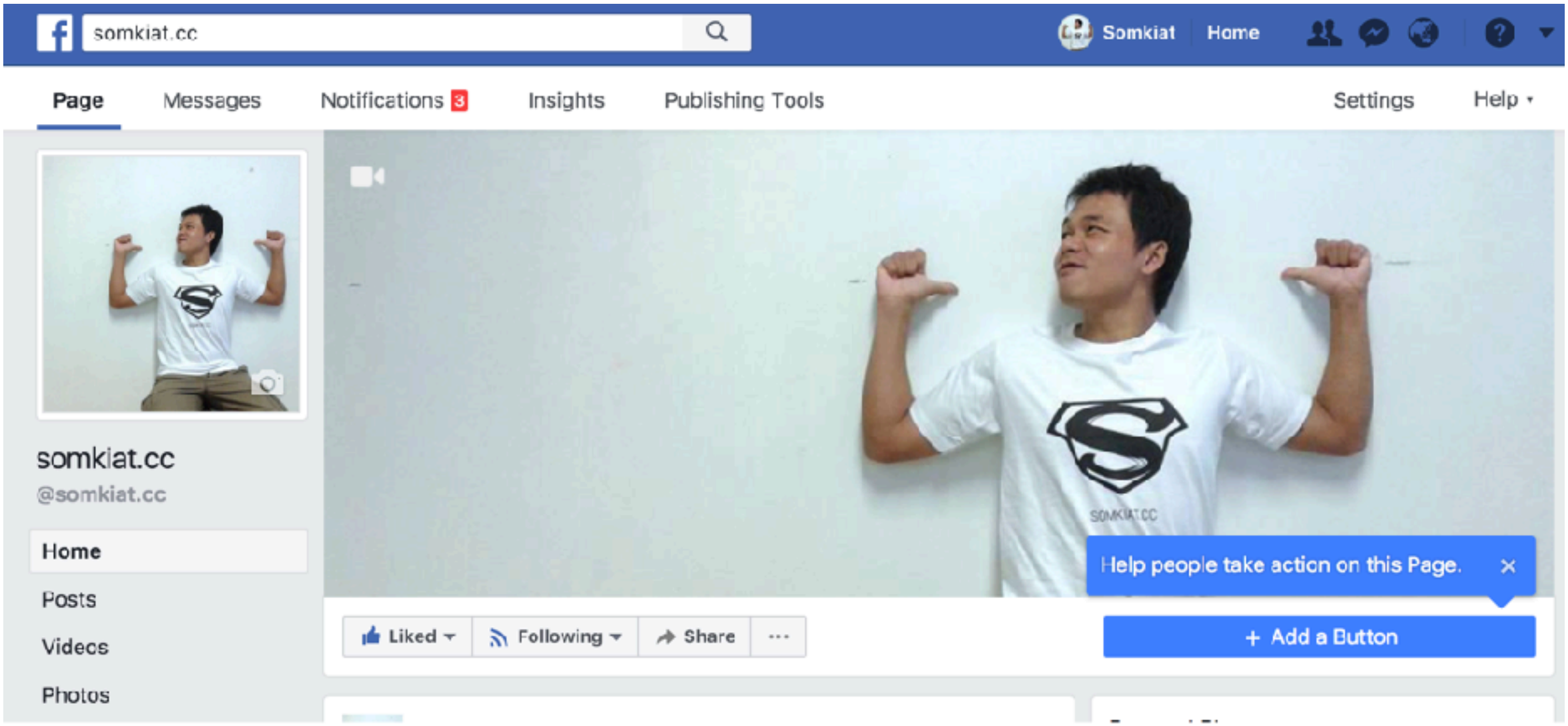
สยามชำนาญกิจ
SIAM CHAMNAN KIT

Kubernetes

In Practice





Agenda Day 1

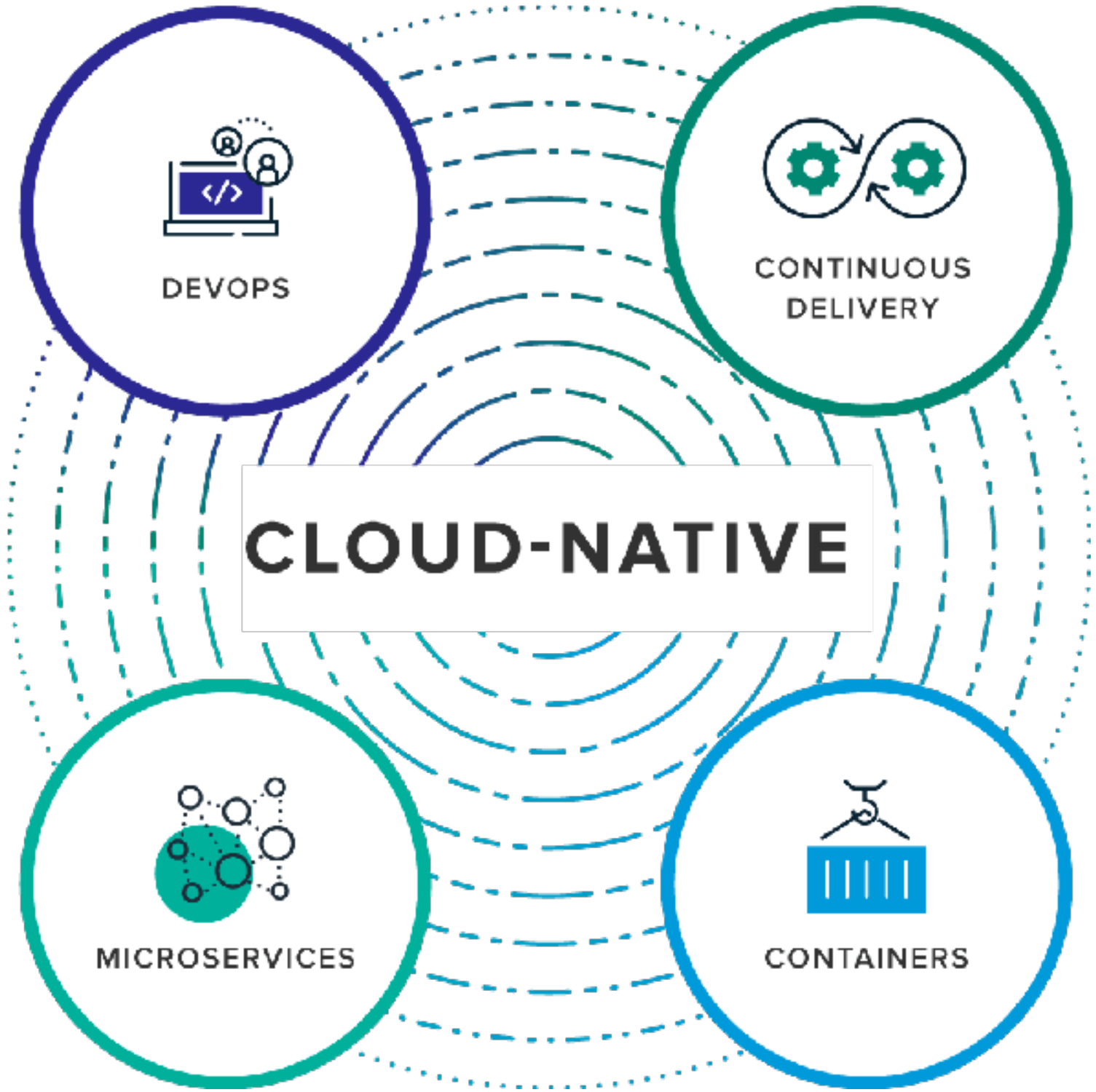
1. Cloud Native Application
2. Kubernetes architecture
3. Key-features
4. Pods and Containers
5. Service
6. Replication Controller (RC)
7. Deployment and ReplicaSet (RS)
8. Volume

Agenda Day 2

1. Resource management
2. Horizontal Pods Autoscaler (HPA)
3. ConfigMap and Secret
4. Log and monitoring
5. Ingress network
6. Batch execution
7. Persistence storage

https://github.com/up1/course-kubernetes-in-practice

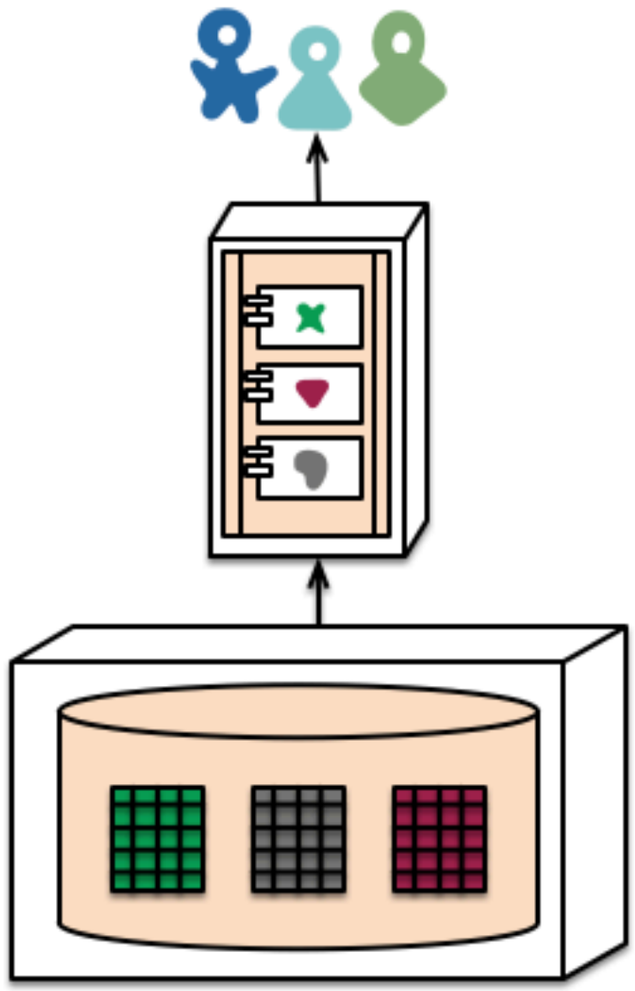
Cloud native



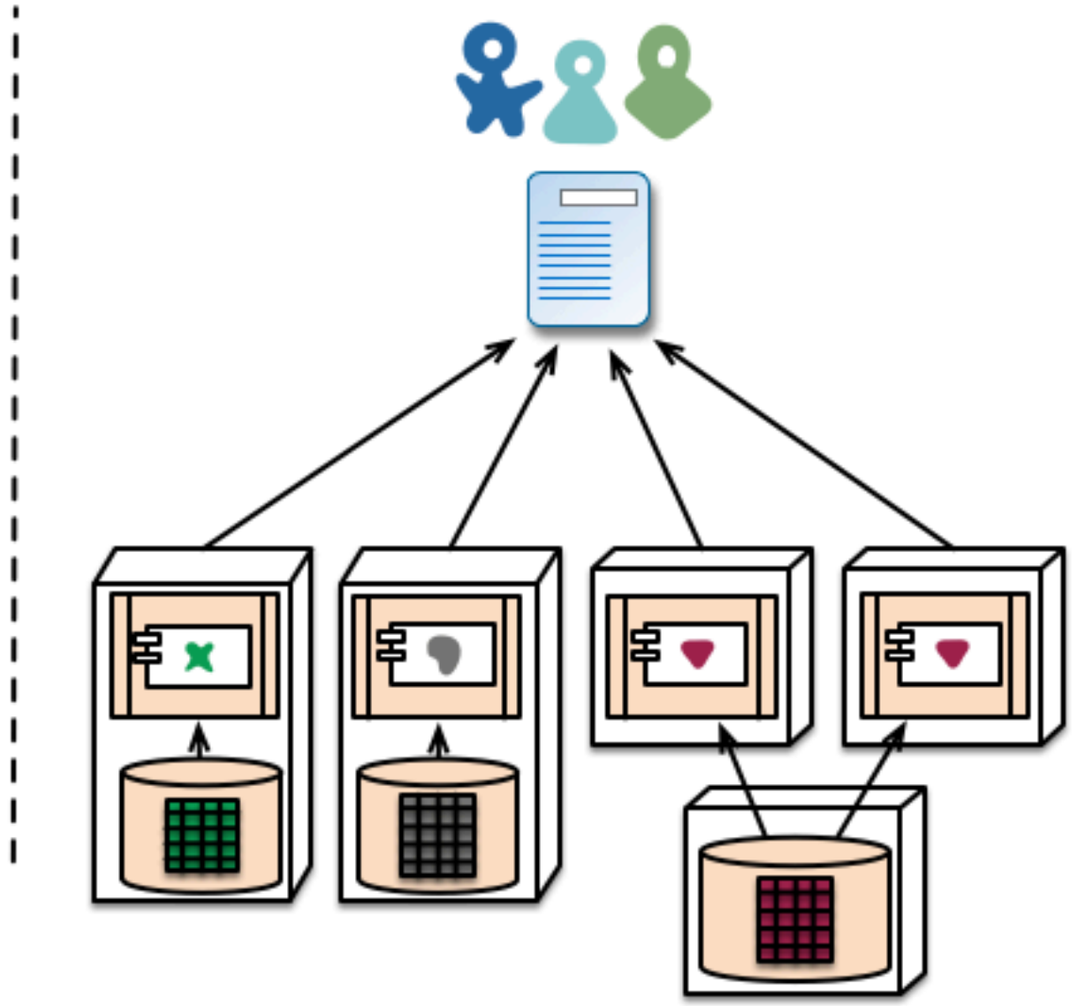
https://pivotal.io/cloud-native

Evolution of Architecture

Microservies

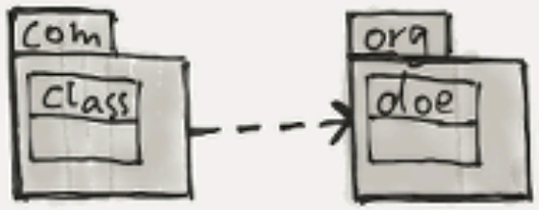


monolith - single database

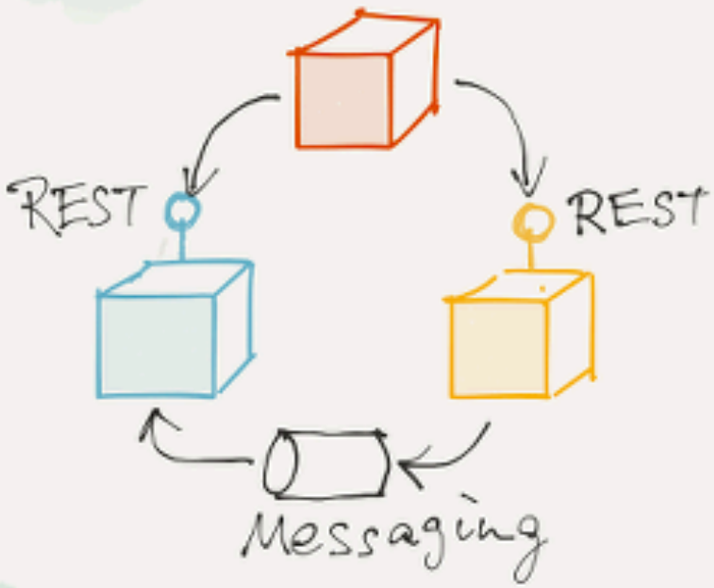


microservices - application databases

Architecture



Microservices

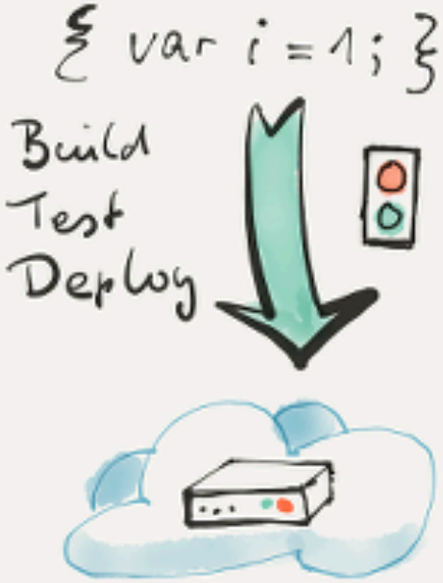


People & Teams

Communication
Collaboration

Deployment

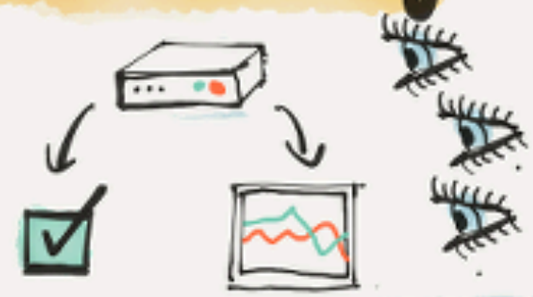
Continous Delivery



Infrastructure

Provisioning

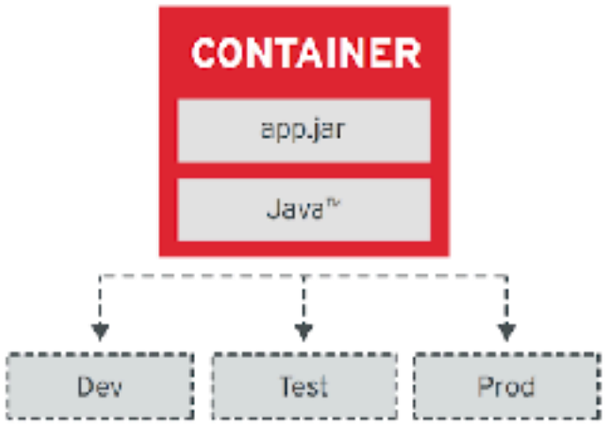
Monitoring



Features & Technology

Container design principle

Image Immutability Principle

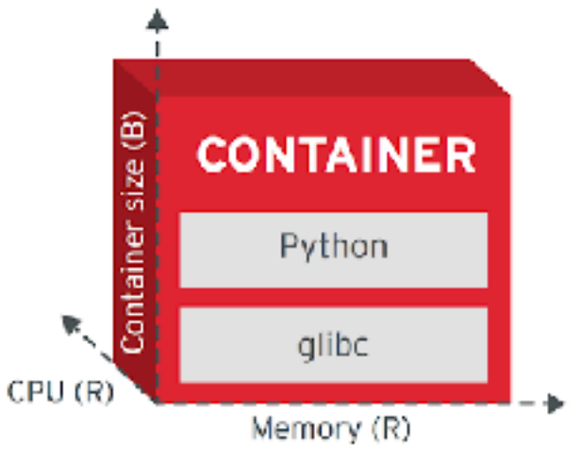


High Observability Principle

Process Disposability Principle

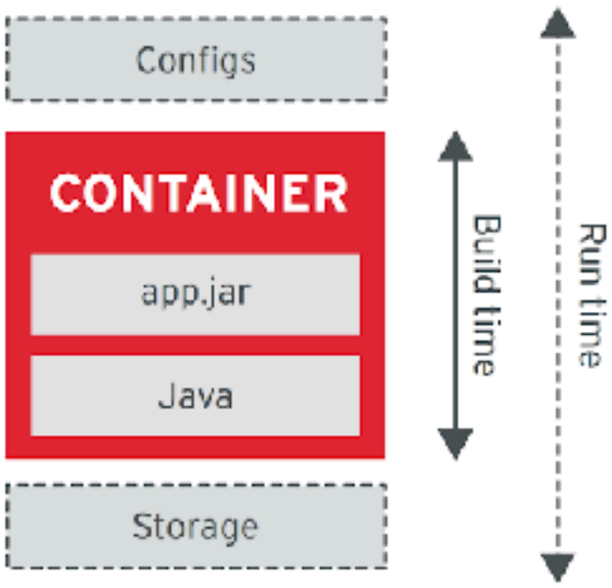
Lifecycle Conformance Principle

Runtime Confinement Principle

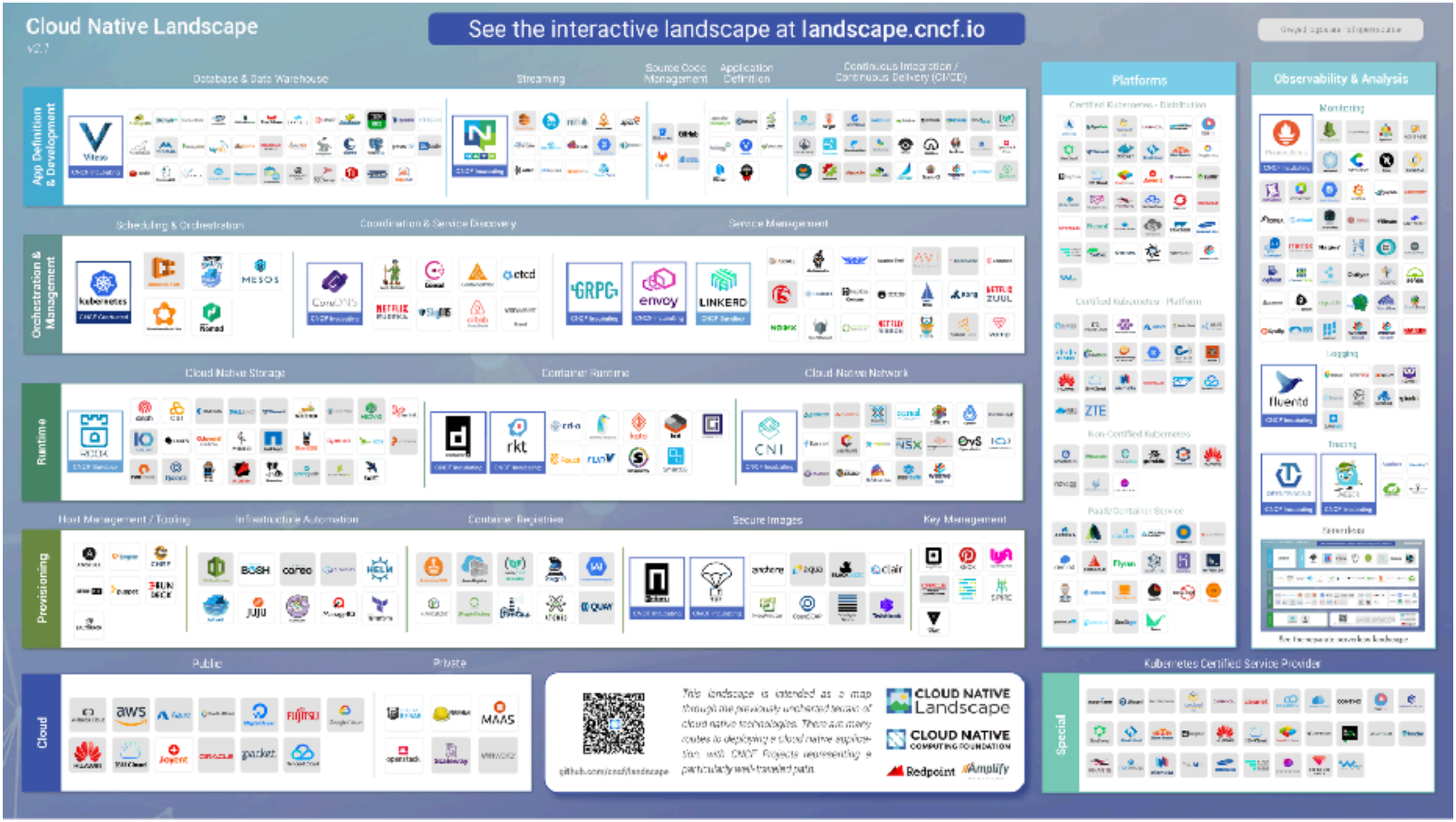


Single Concern Principle

Self-Containment Principle



Cloud native landscape



https://github.com/cncf/landscape

Installation

Software requirement

minkube
Kubernetes command-line tool

Starting minikube

$minkube start
$minikube status

Version of Kubernetes

$kubectl get nodes -o yaml

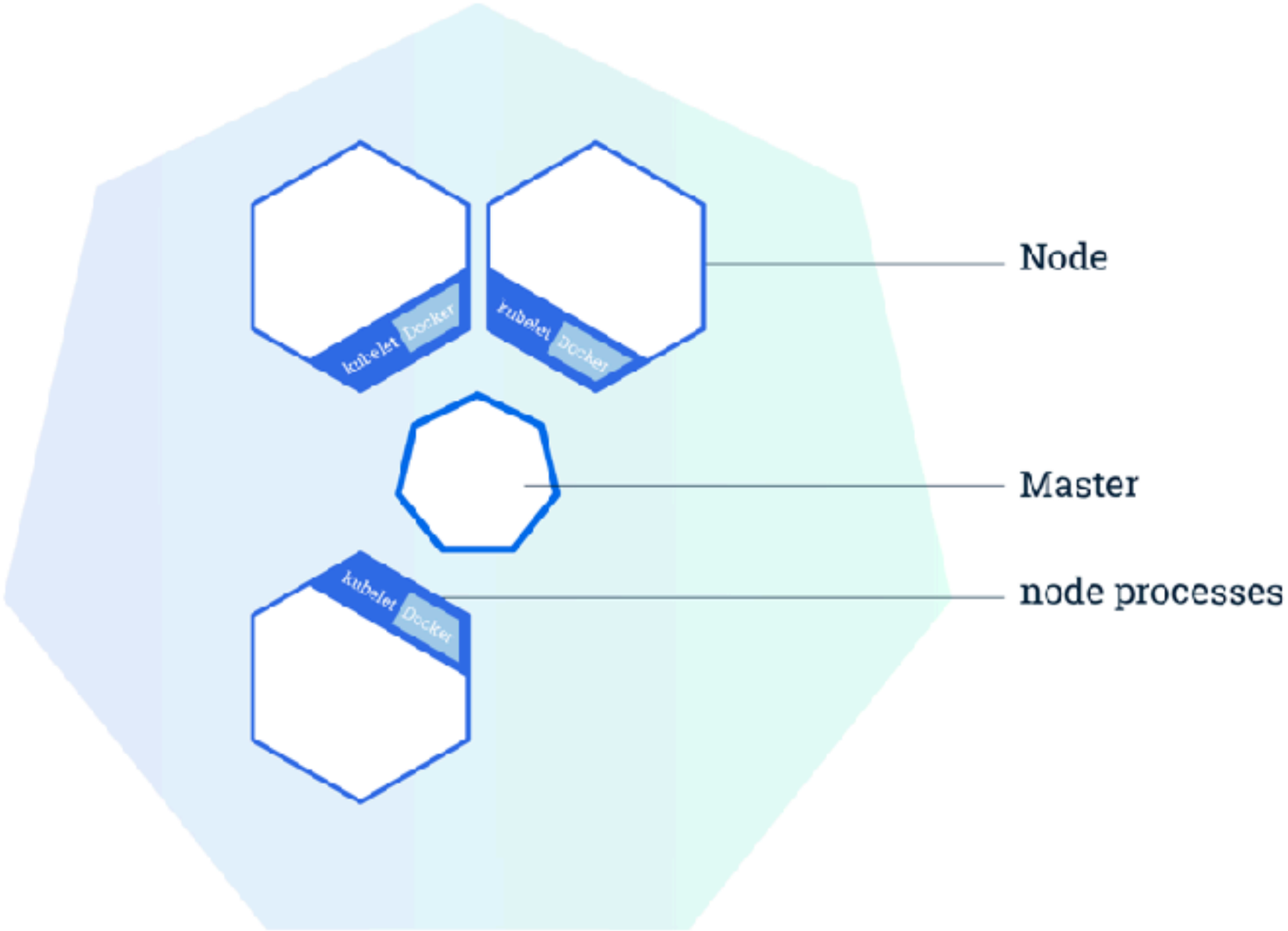
```
nodeInfo:
  architecture: amd64
  bootID: 7c769465-27e5-4dd8-a89f-319ba1b8ef57
  containerRuntimeVersion: docker://17.9.0
  kernelVersion: 4.9.64
  kubeProxyVersion: v1.9.4
  kubeletVersion: v1.9.4
  machineID: edffd4ca8bf24253a736ea86d2448185
  operatingSystem: linux
  osImage: Buildroot 2017.11
  systemUUID: 98BAE0EF-8C9E-45DF-9AC7-F0E05806B189
```

Basic of Kubernetes

Kubernetes (k8s)

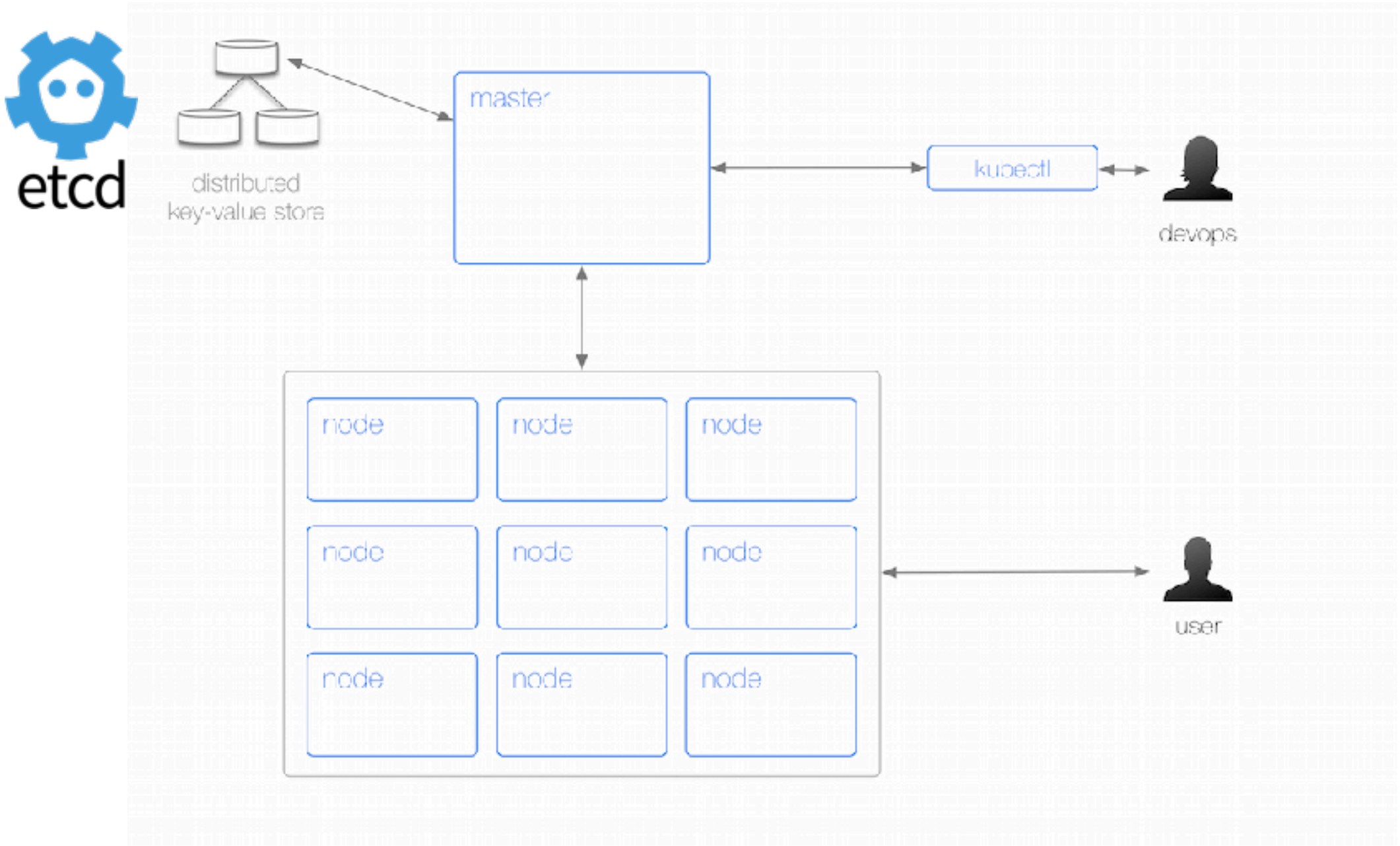
Container orchestration tool
TODO
TODO

Kubernetes cluster



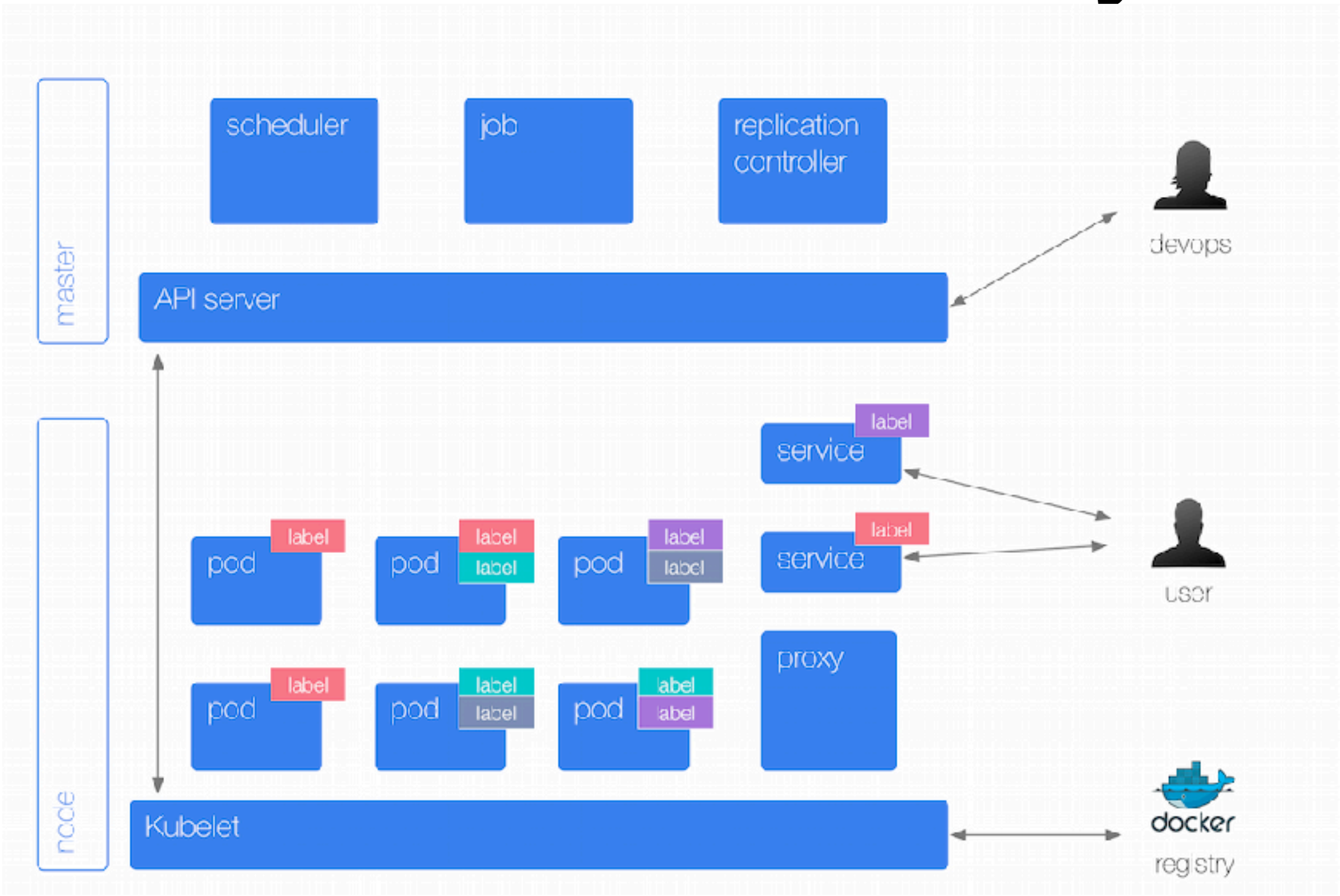
Kubernetes cluster

Kubernetes physical layout



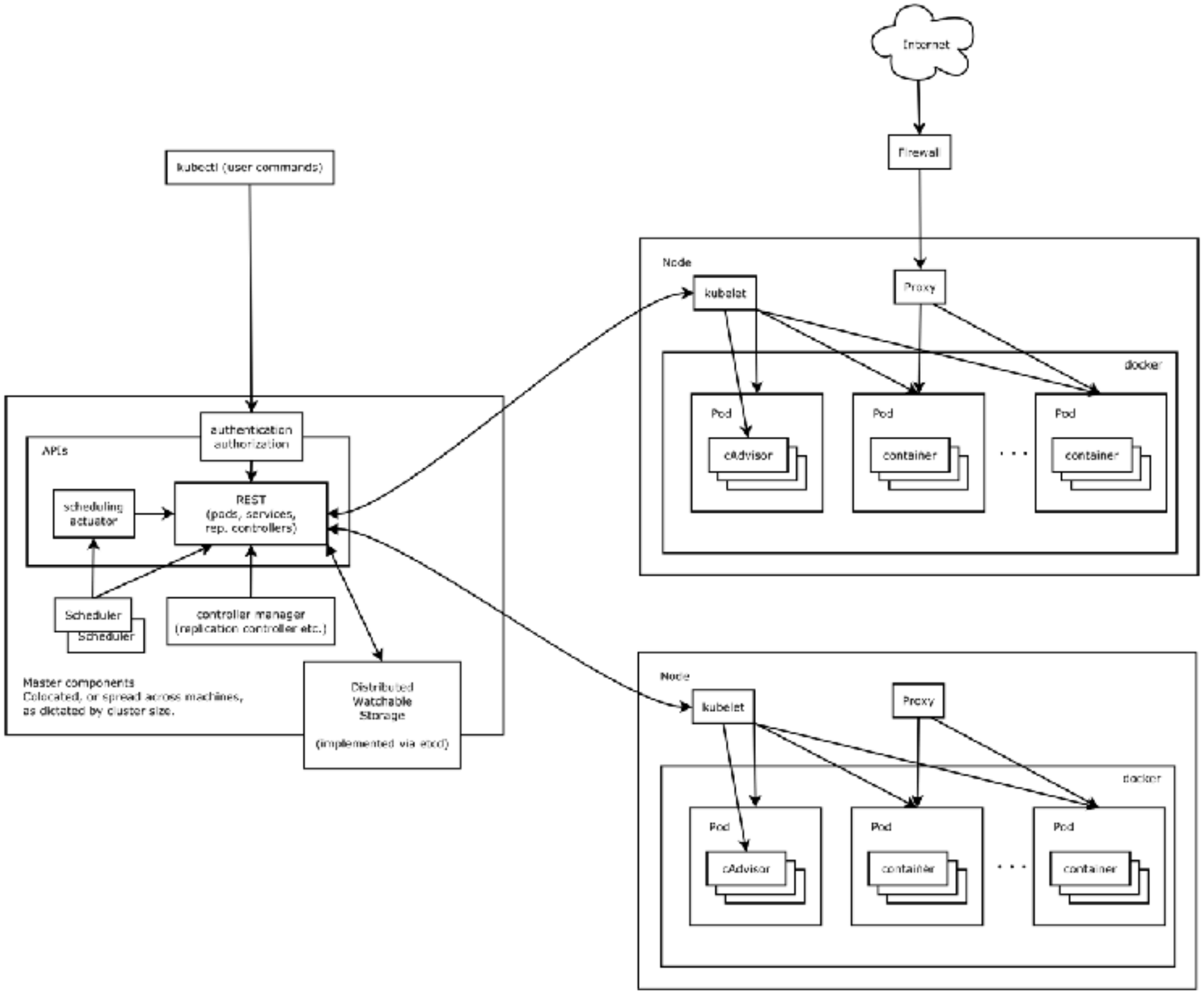
http://k8s.info/cs.html

Kubernetes abstraction layout



http://k8s.info/cs.html

Kubernetes architecture



Hello with Kubernetes

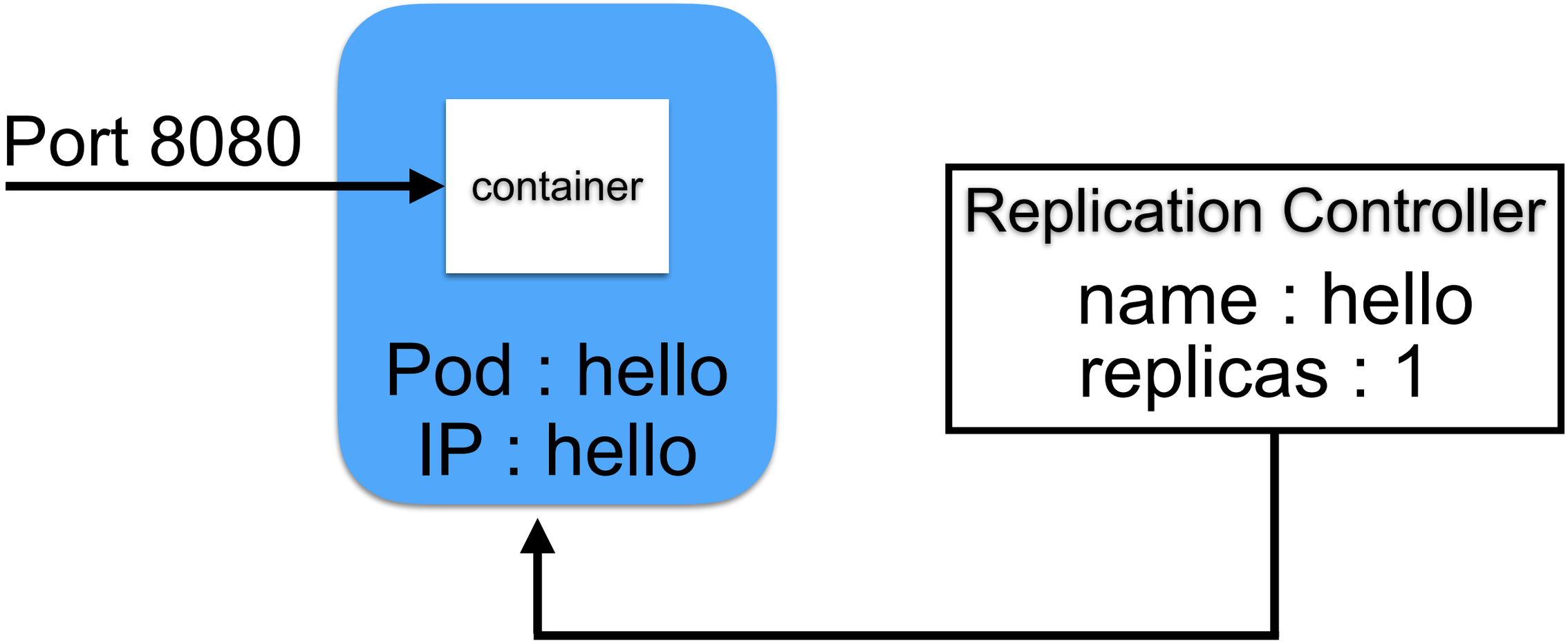
Hello Kubernetes

Working with **kubectl** in command line
Try to run image from Docker hub

Create a container

$kubectl run **hello** --image=somkiat/hello --port=8080 **--generator=run/v1**

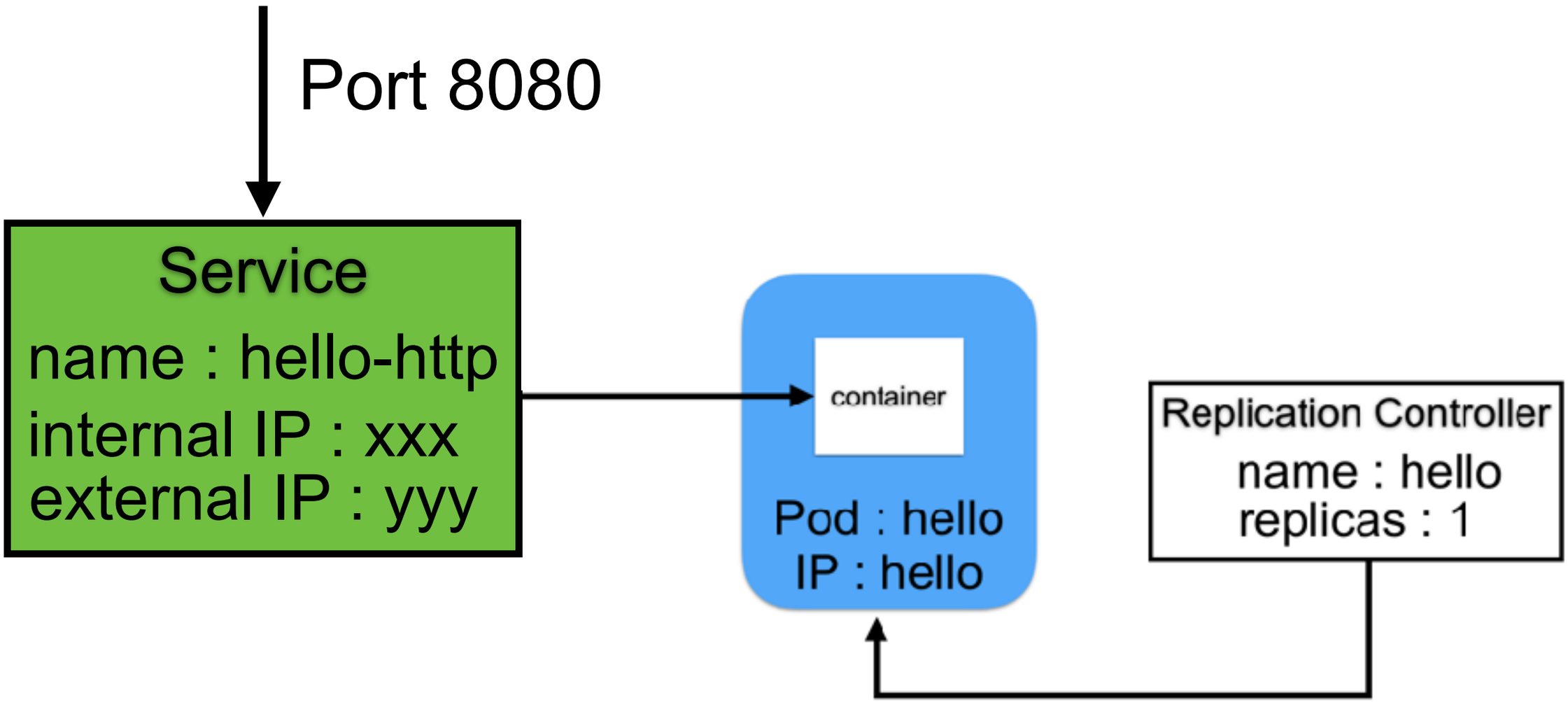
Create a container



Expose RC with service

$kubectl expose rc hello
--type=LoadBalancer --name **hello-http**

Expose RC with service



Access service with qinikube

$minikube service hello-http

Scaling the application

$kubectl **scale** rc hello **--replicas=3**

List of RC

$kubectl get rc

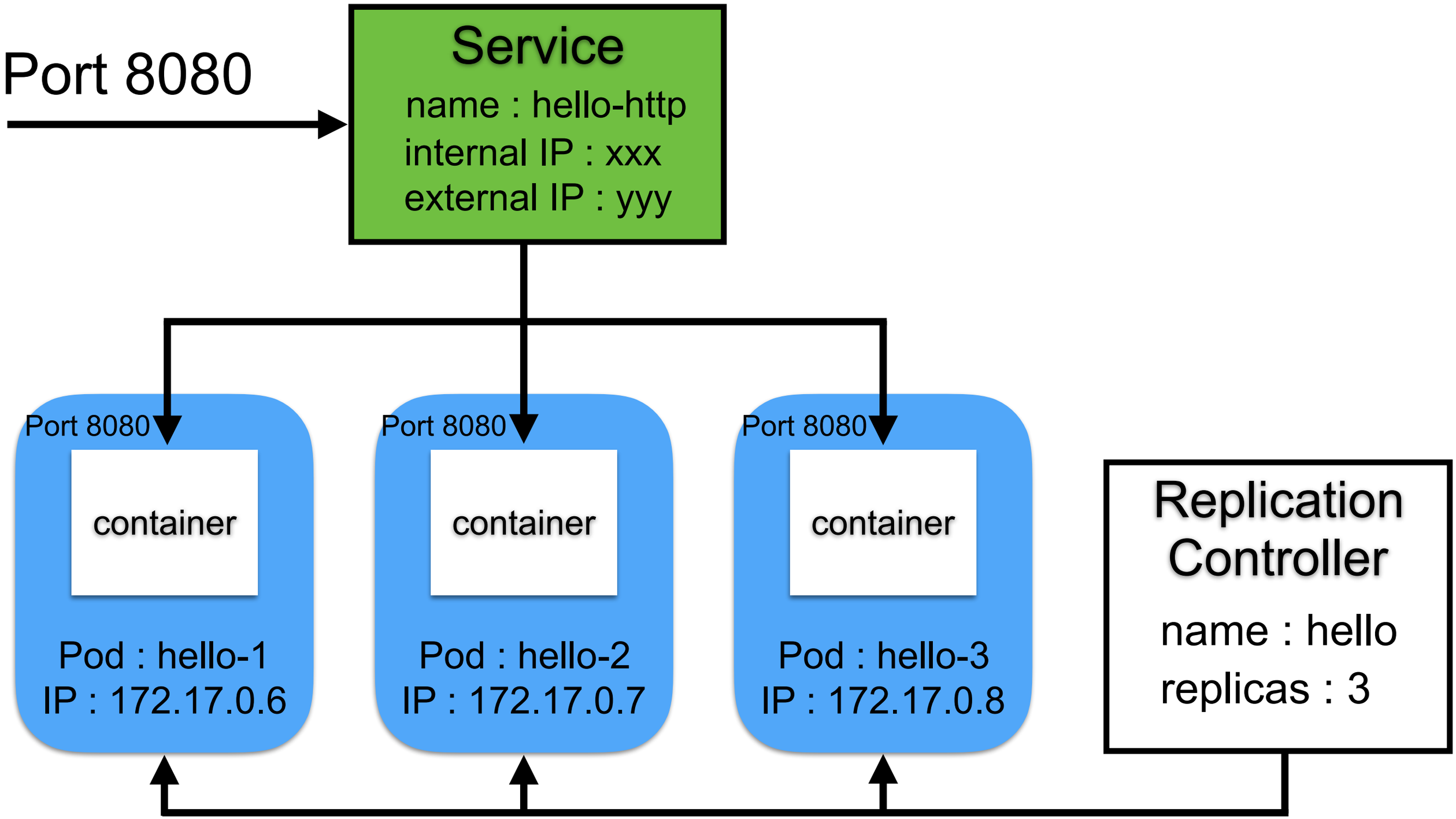
```
NAME      DESIRED   CURRENT   READY     AGE
hello     3         3         3         35m
```

List of Pods

$kubectl get pod -o wide

```
NAME          READY     STATUS    RESTARTS   AGE       IP             NODE
hello-lw2p7   1/1       Running   0          29m       172.17.0.8     minikube
hello-pqnzp   1/1       Running   0          36m       172.17.0.6     minikube
hello-wwsnh   1/1       Running   0          29m       172.17.0.7     minikube
```

Scaling the application



Testing

export SERVICE=192.168.99.100:32614

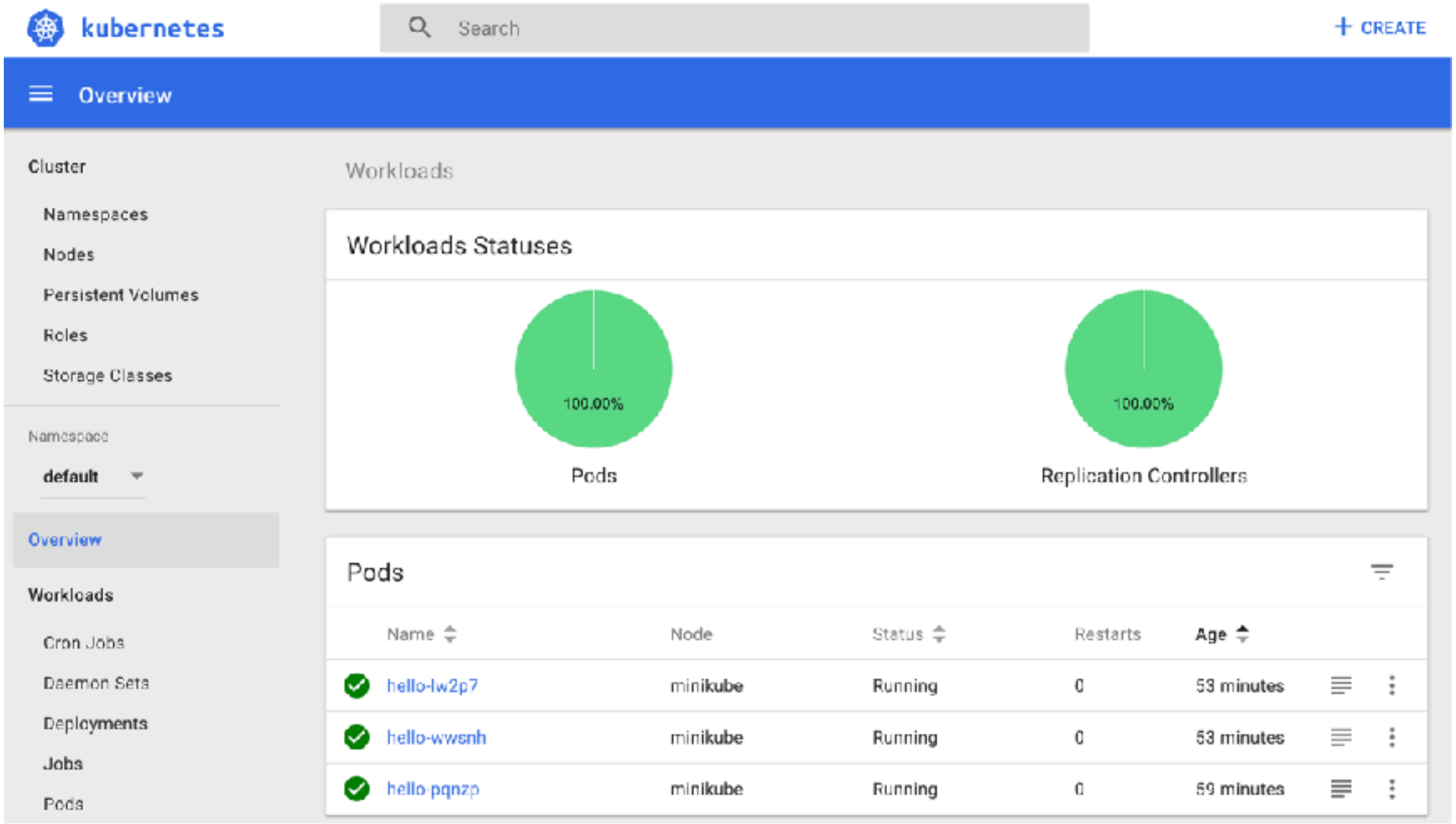
$curl http://$SERVICE
Hello, "/" on **hello-wwsnh**

$curl http://$SERVICE
Hello, "/" on **hello-pqnzp**

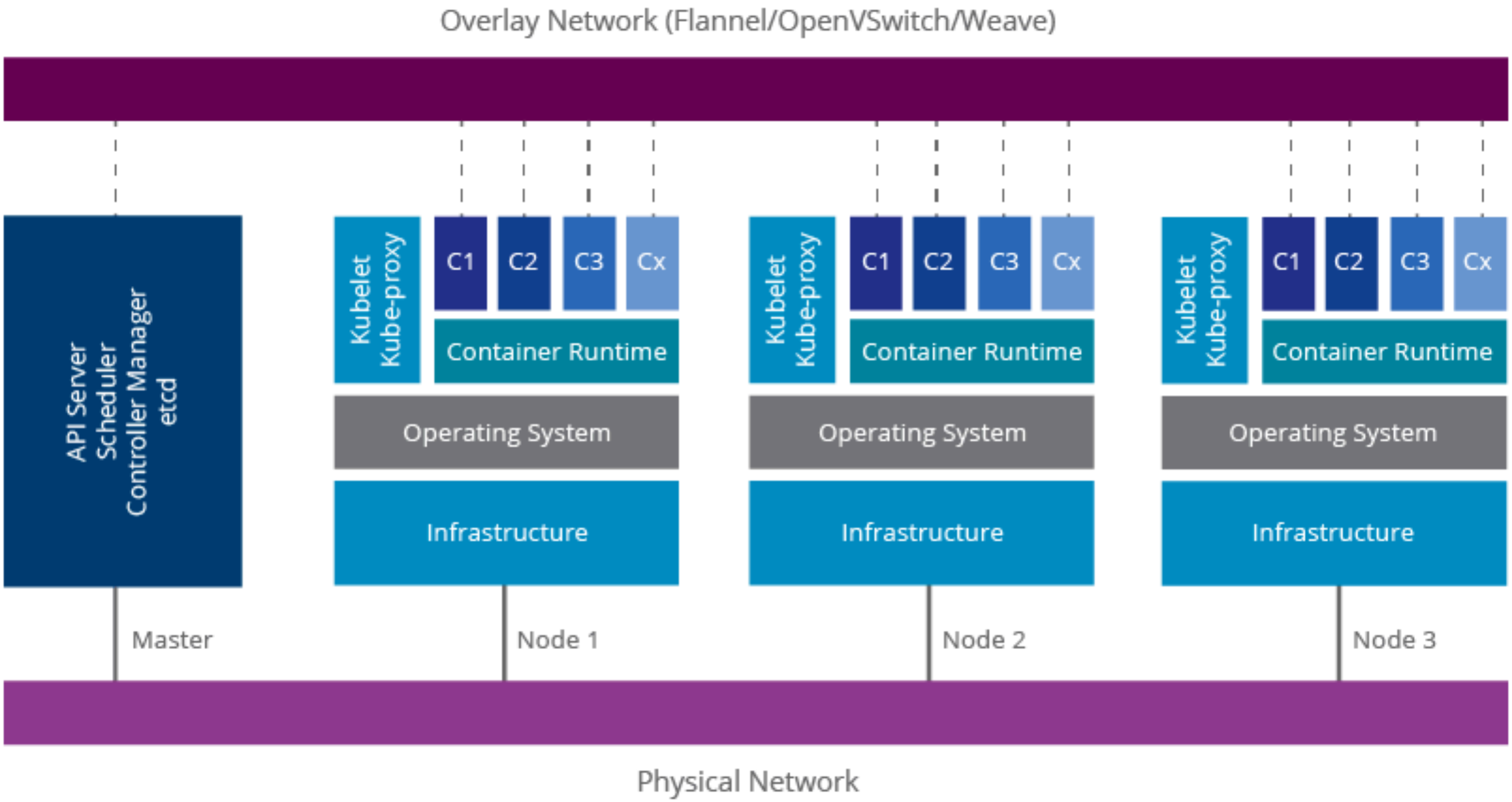
$curl http://$SERVICE
Hello, "/" on **hello-lw2p7**

Kubernetes dashboard

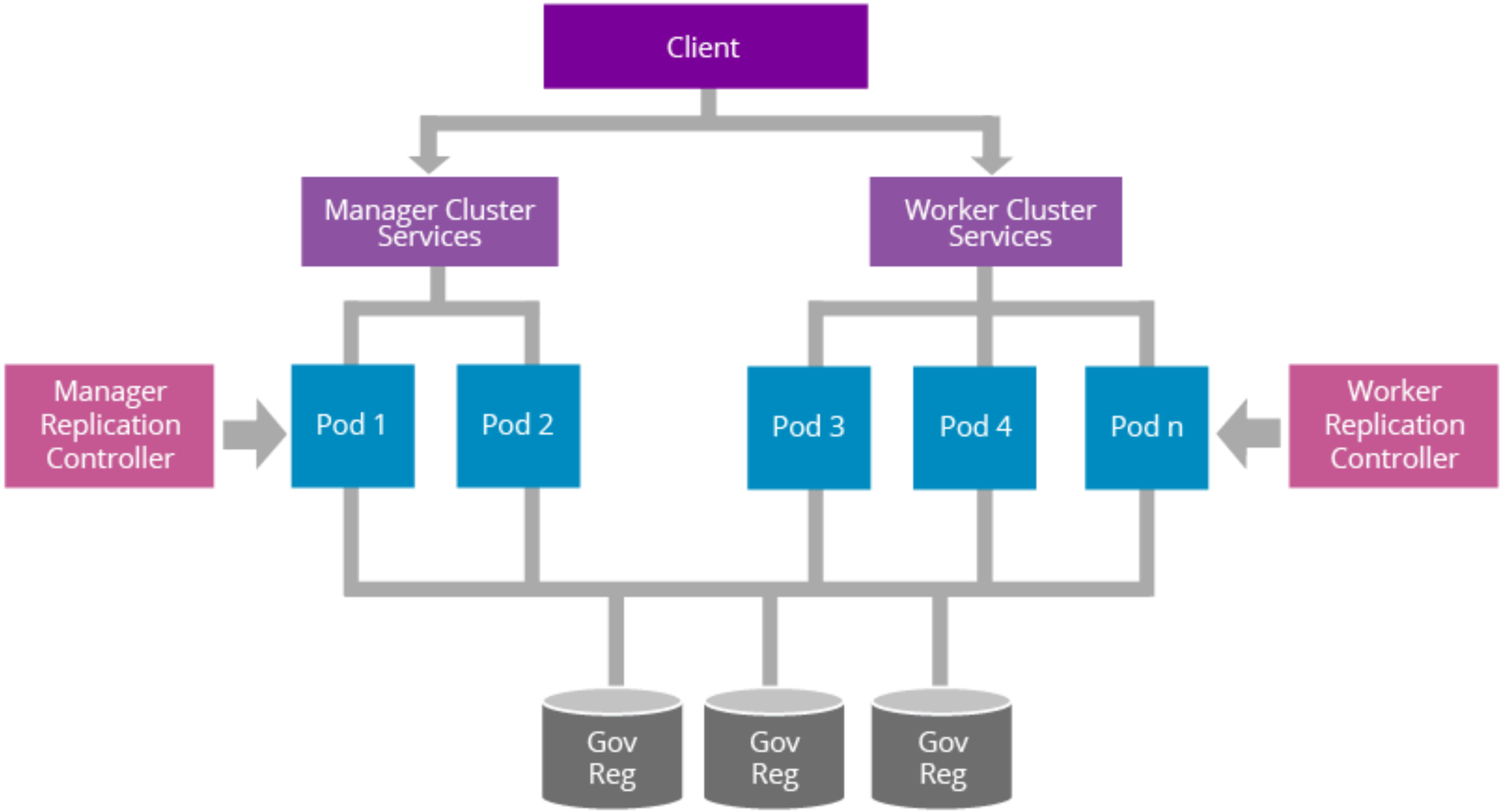
$minikube dashboard



Kubernetes architecture



Kubernetes architecture



https://wso2.com/whitepapers/a-reference-architecture-for-deploying-wso2-middleware-on-kubernetes/

Key features

Key features

Automatic binpacking
Horizontal Pod Autoscaler (HPA)
Automated rollouts and rollbacks
Storage orchestration

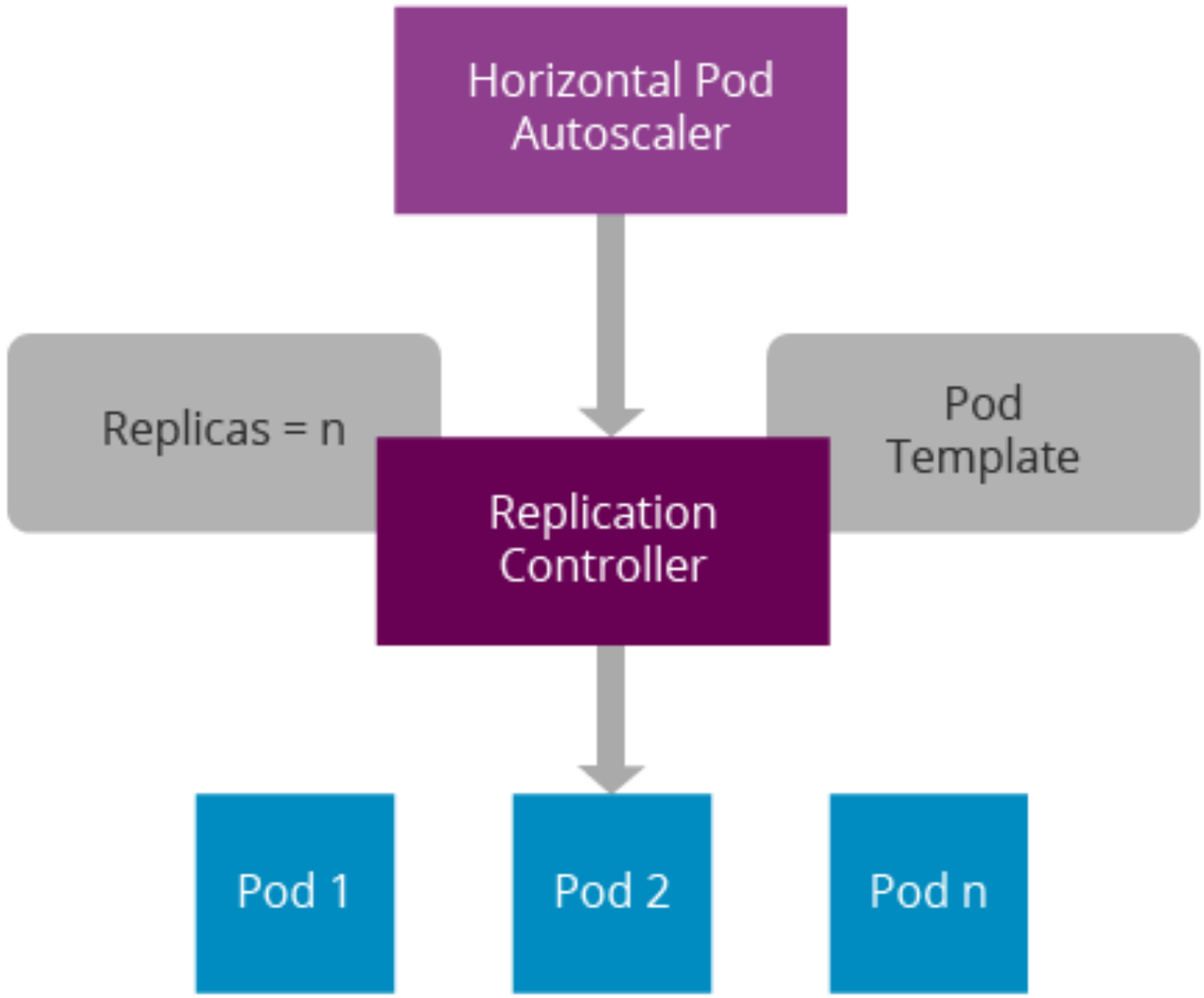
Key features

Self-healing
Service discovery and Load Balancing (LB)
Secret and config management
Batch execution

Automatic bin packing

Horizontal Pod Autoscaler

Horizontal Pod Autoscaler

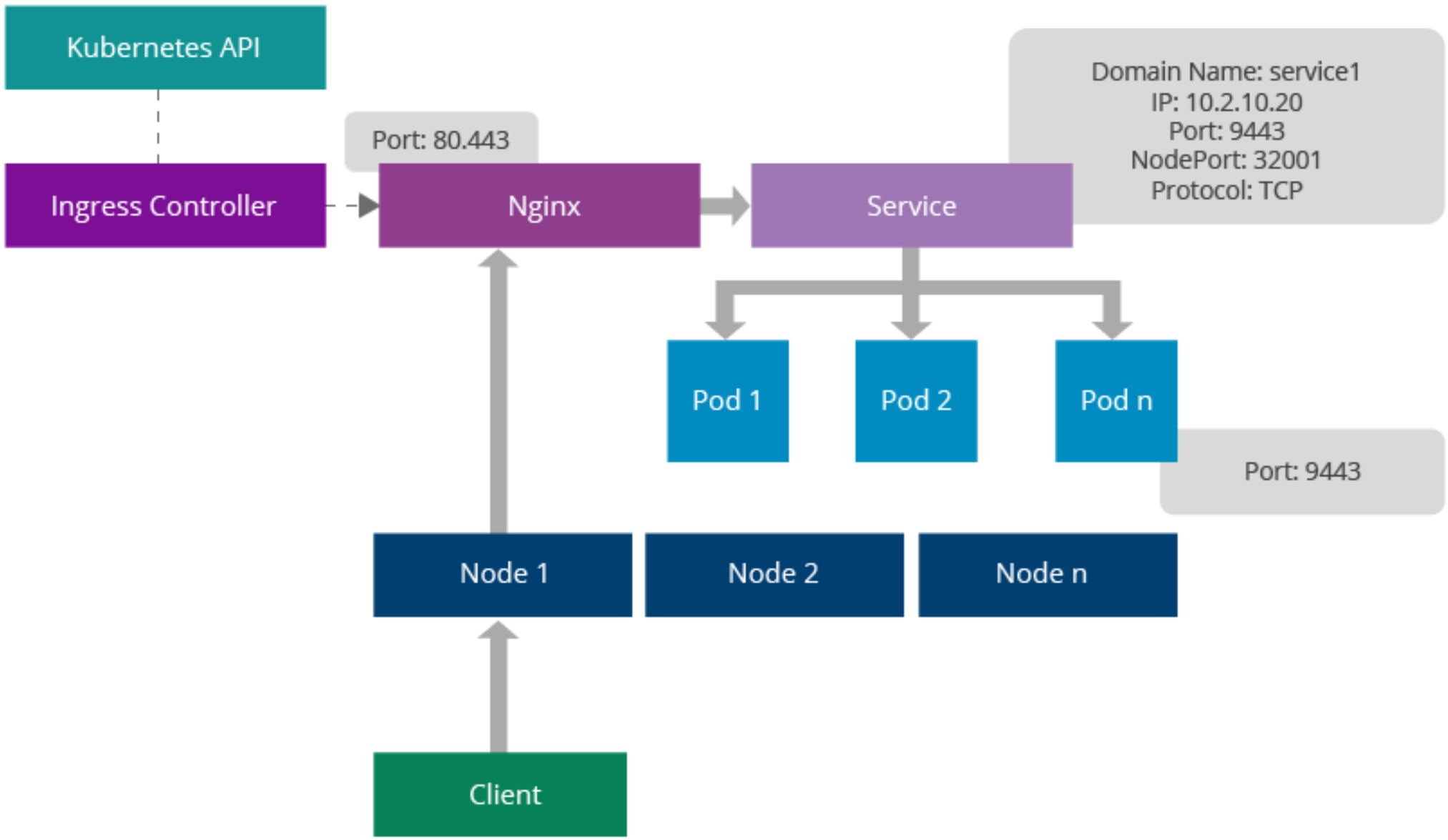


Automated rollouts & rollbacks

Storage orchestration

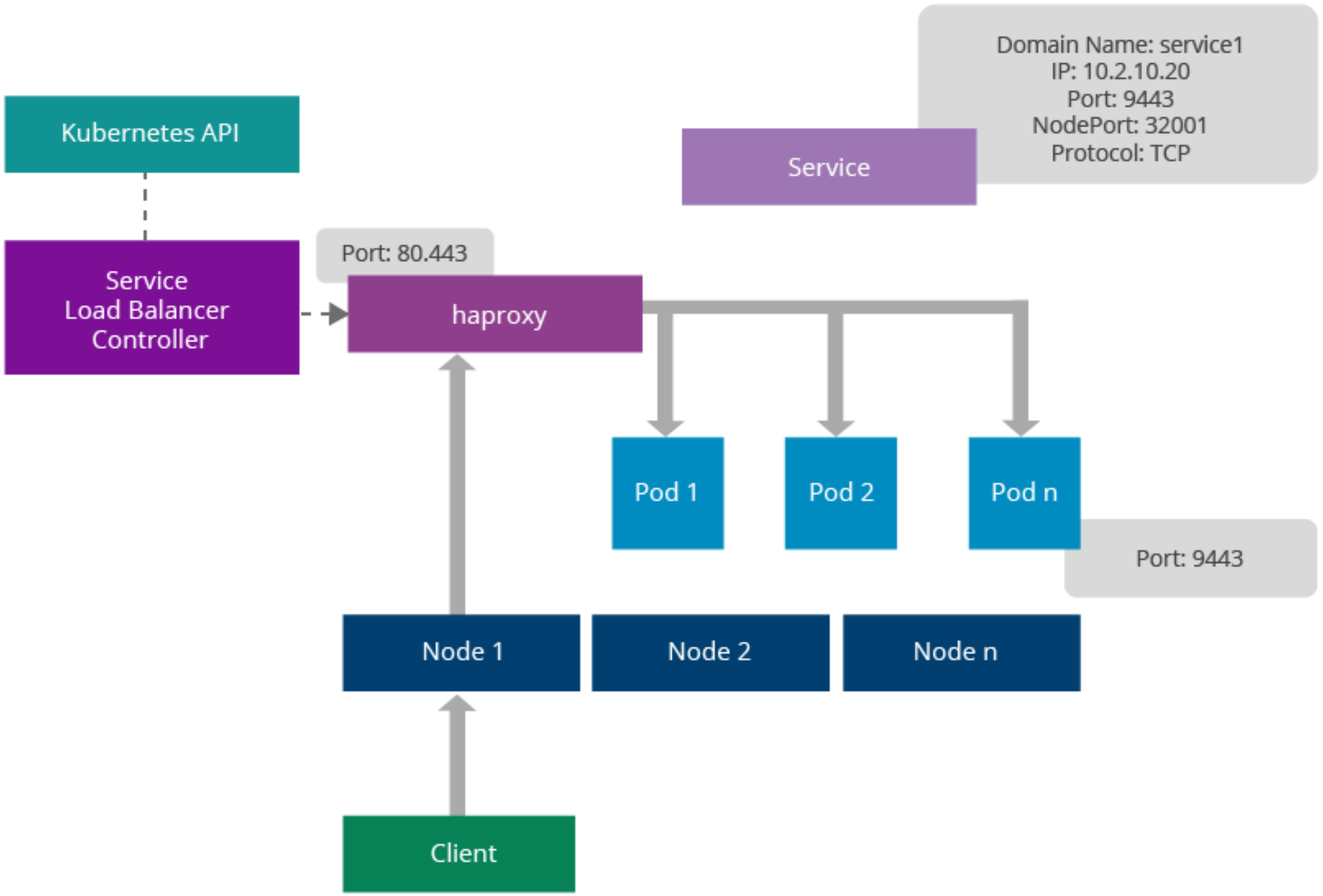
Provide well known ports for Kubernetes services

Ingress controller

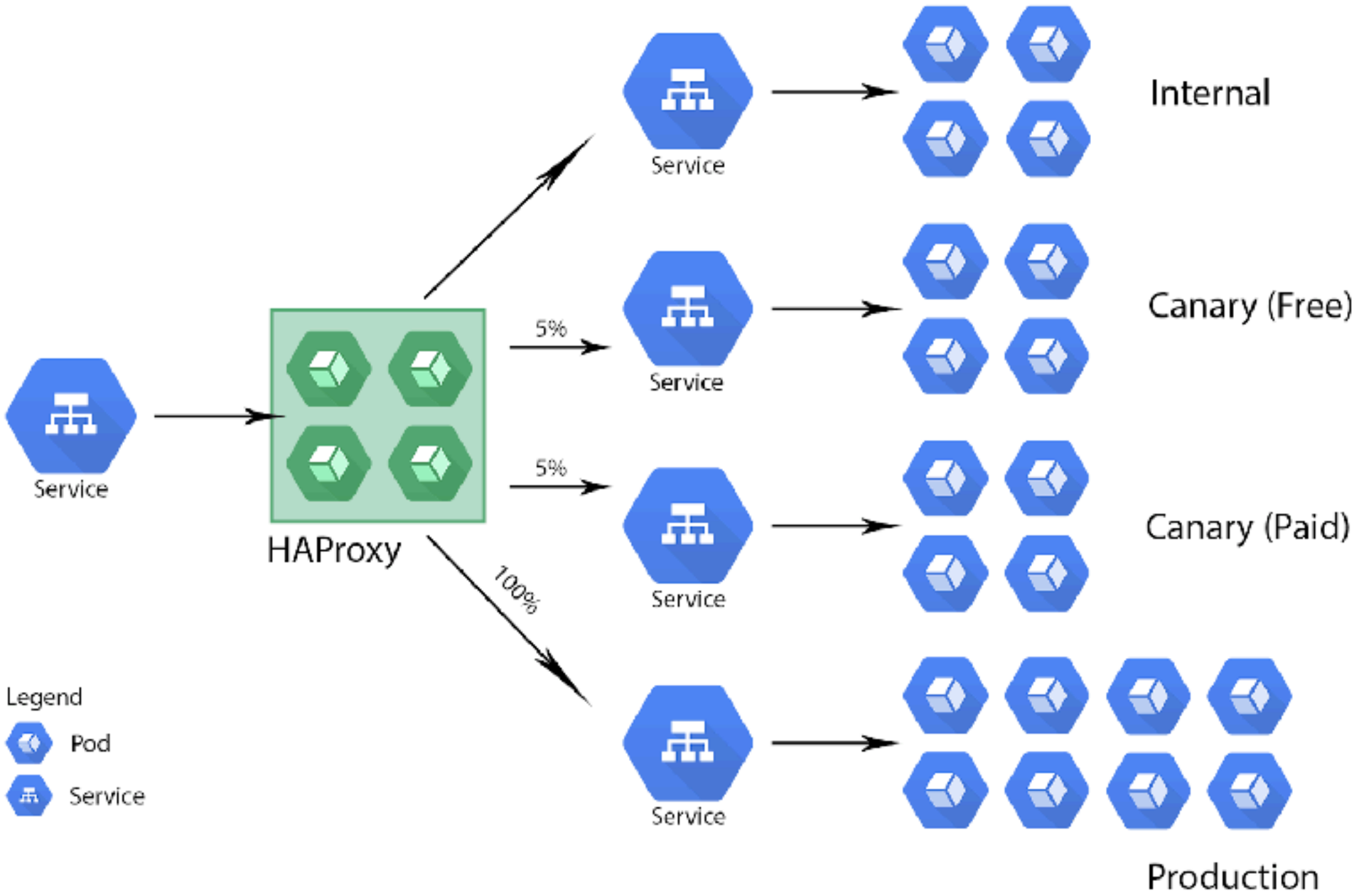


Sticky session management using Service Load Balancers

Service Load Balancer

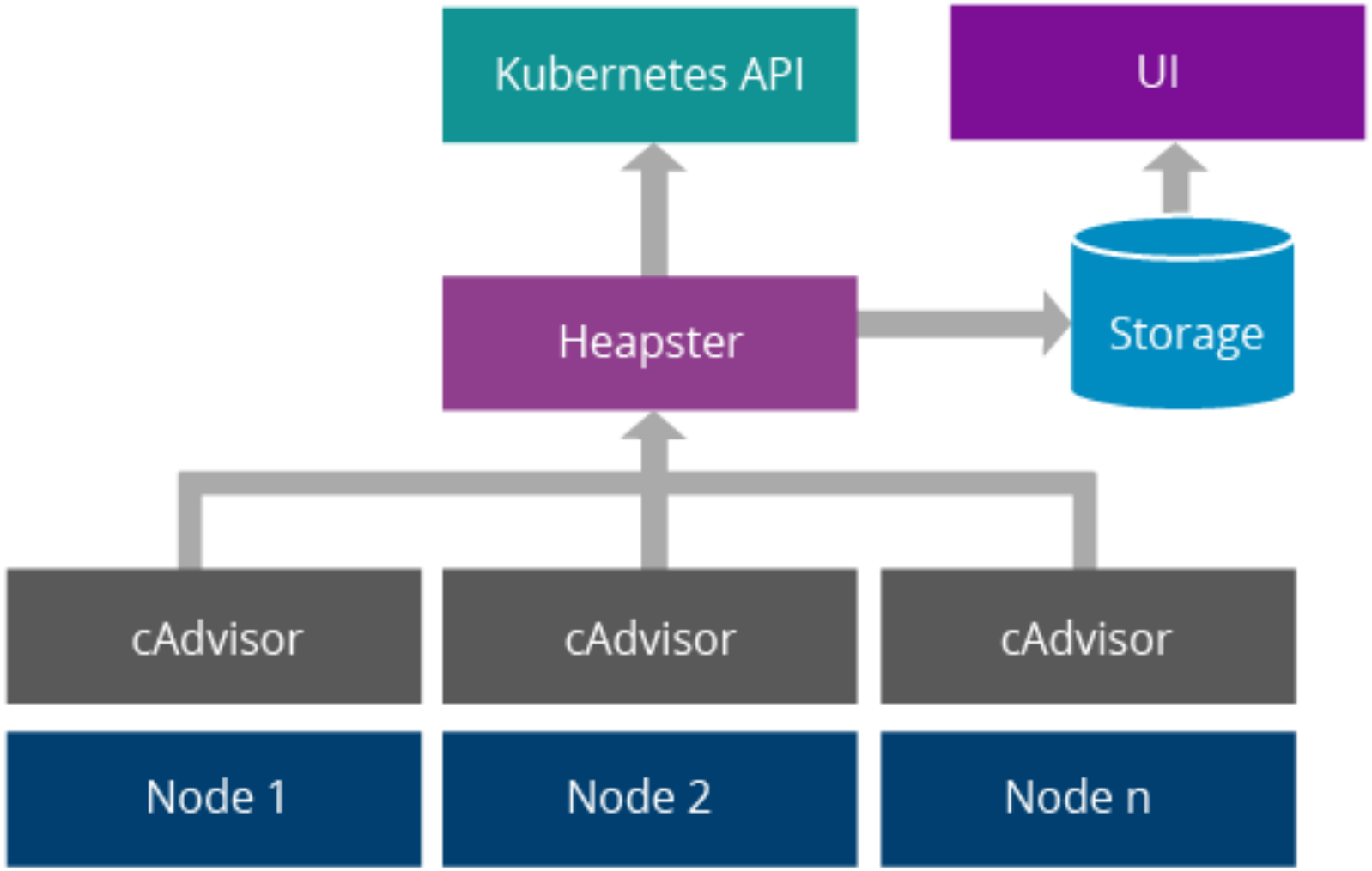


Service Load Balancer



Resource usage monitoring

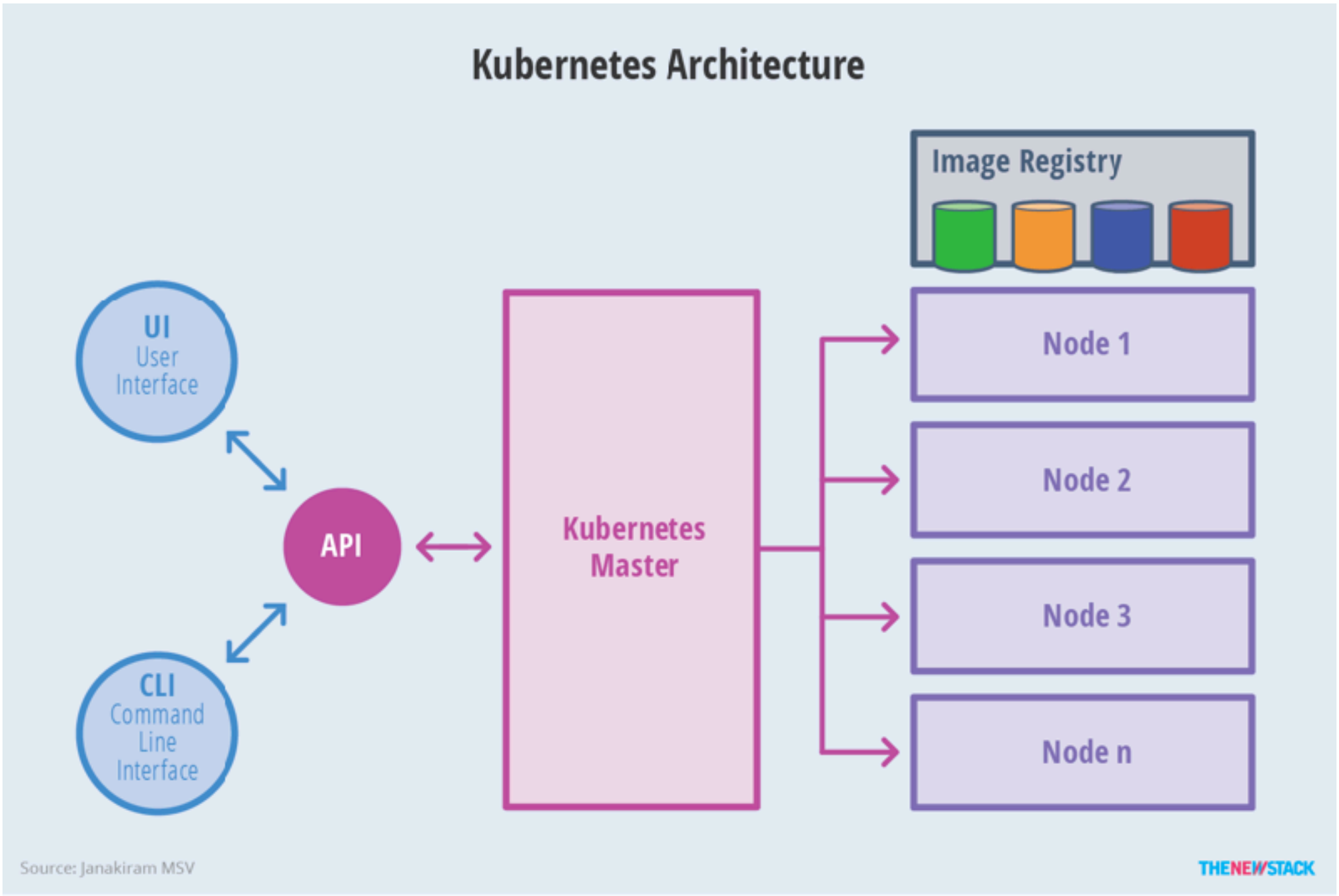
Resource usage monitoring



https://github.com/kubernetes/heapster

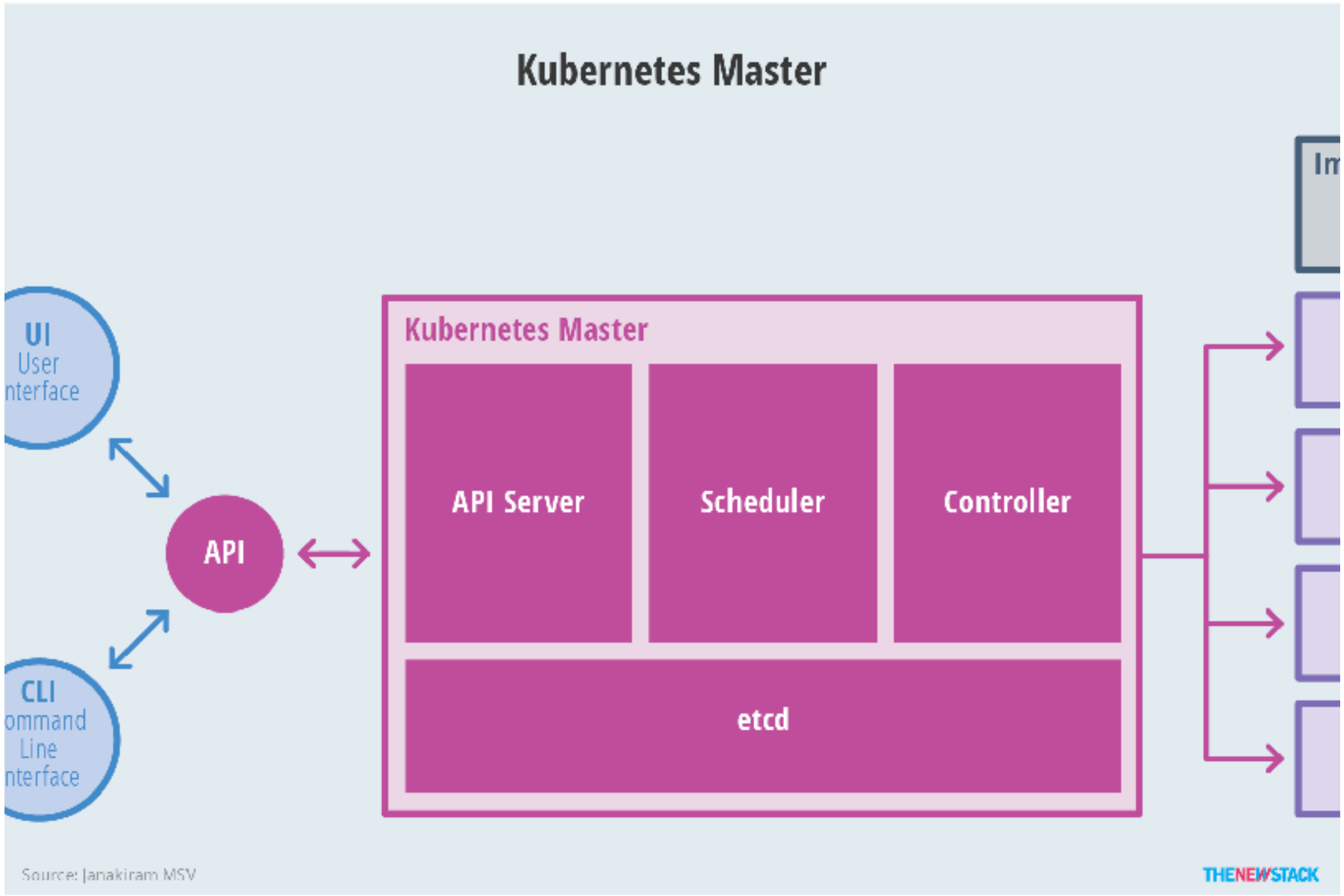
System architecture

System architecture



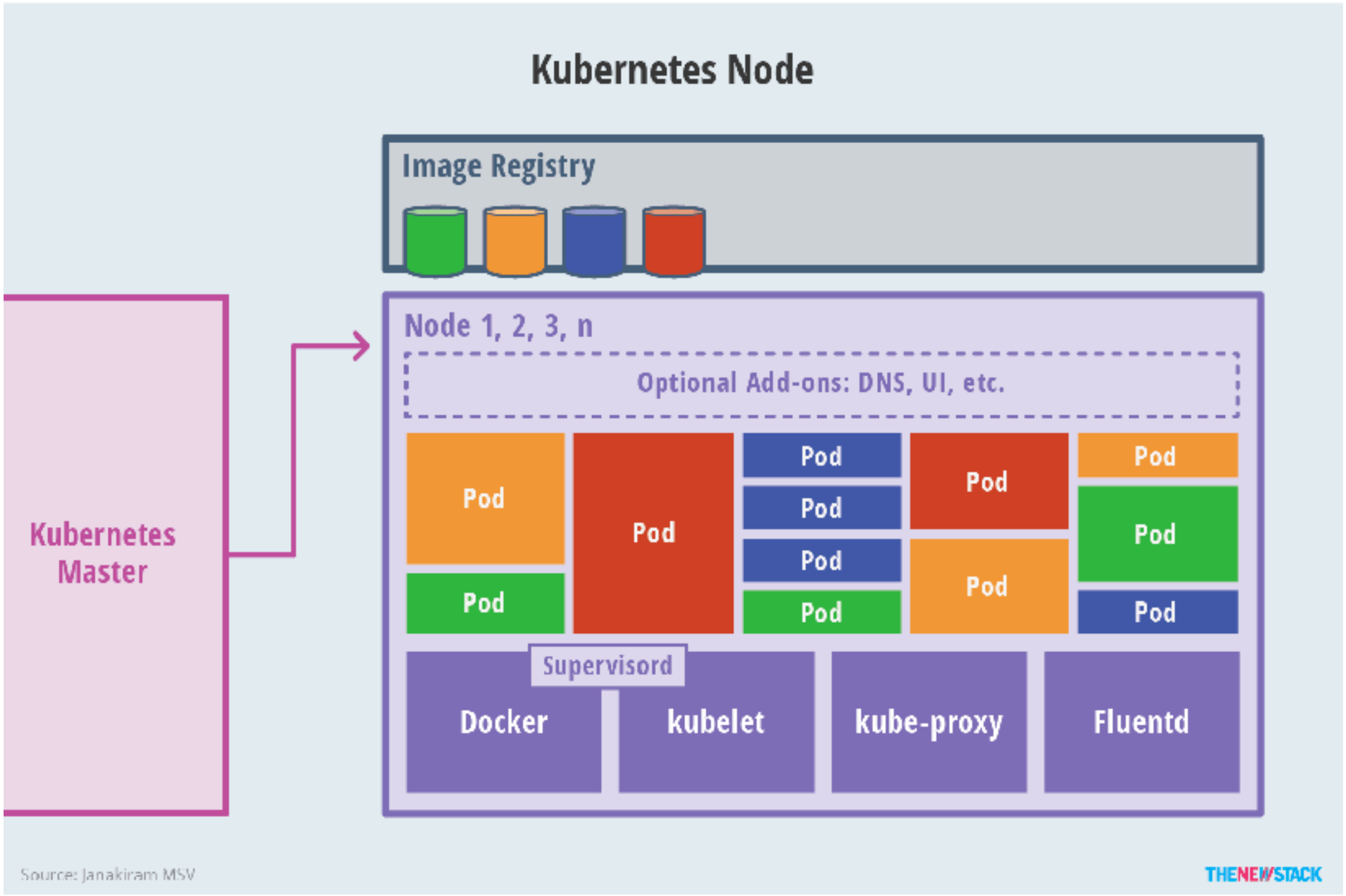
https://thenewstack.io/kubernetes-an-overview/

Master



https://thenewstack.io/kubernetes-an-overview/

Node



https://thenewstack.io/kubernetes-an-overview/

Core concepts of Kubernetes

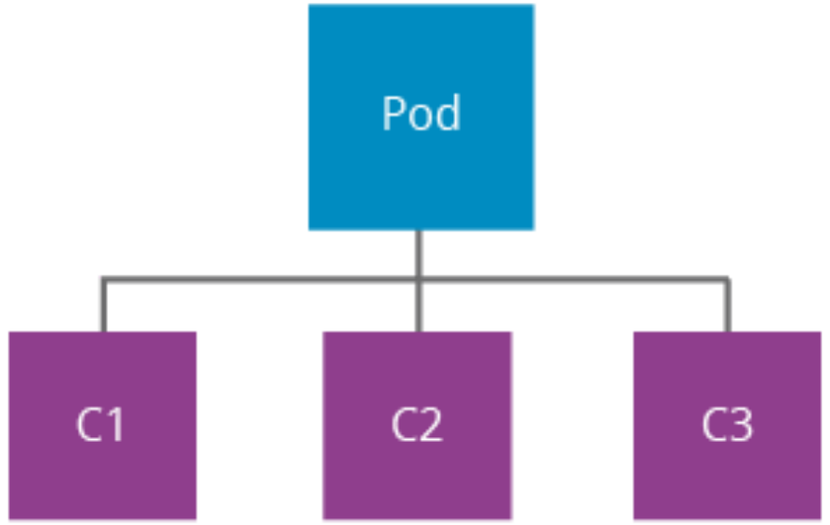
Core concepts

Pods vs containers
Services
Replication Controllers
Deployments
ConfigMap and Secret
Volumes
StatefulSets

Pods vs containers

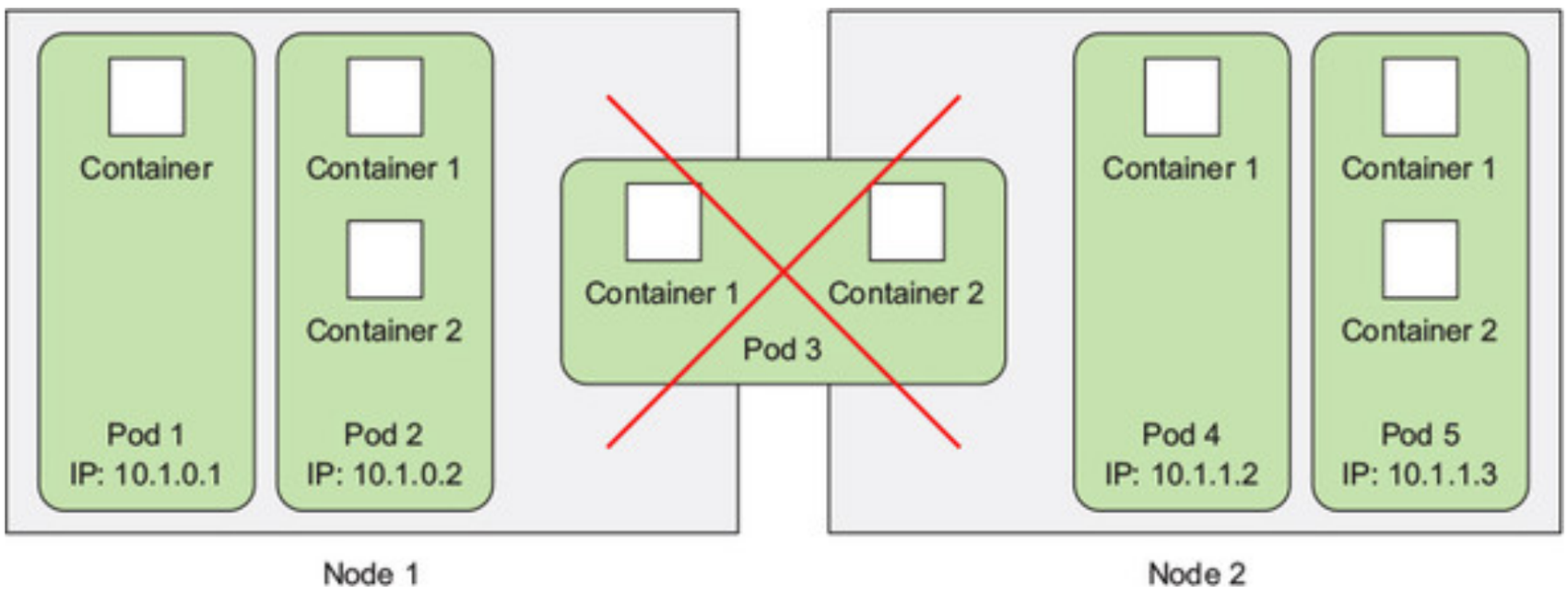
Pods

Small group of co-located containers
Optionally shared volume between containers
Basic deployment unit in Kubernetes



Pods

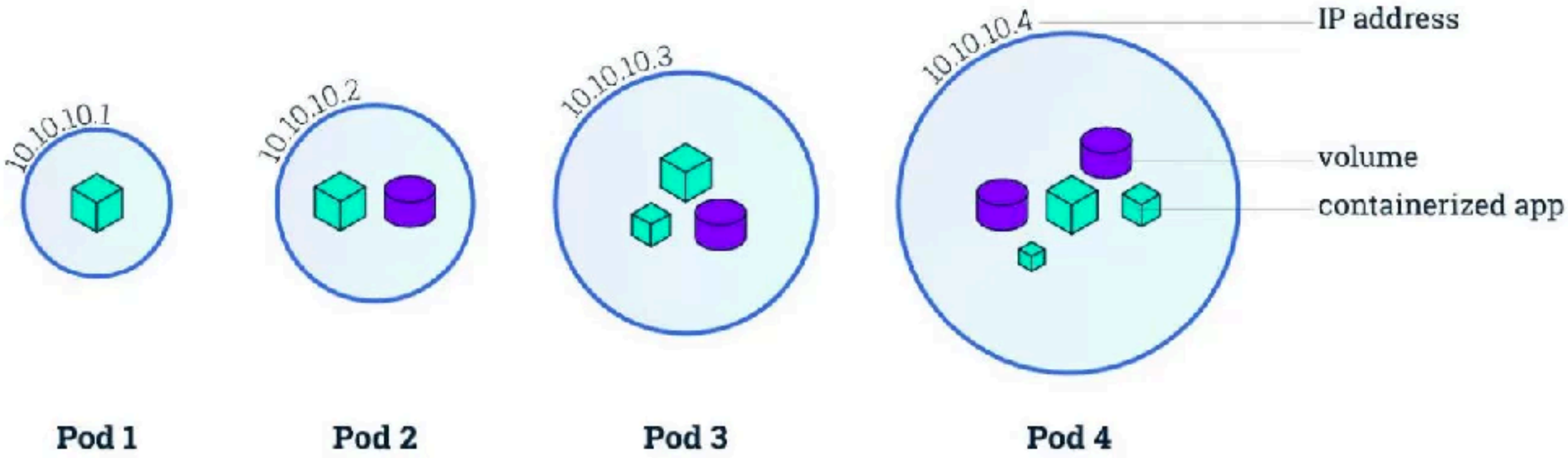
1 pods = 1 container
1 pods = N containers



All containers in same Pods

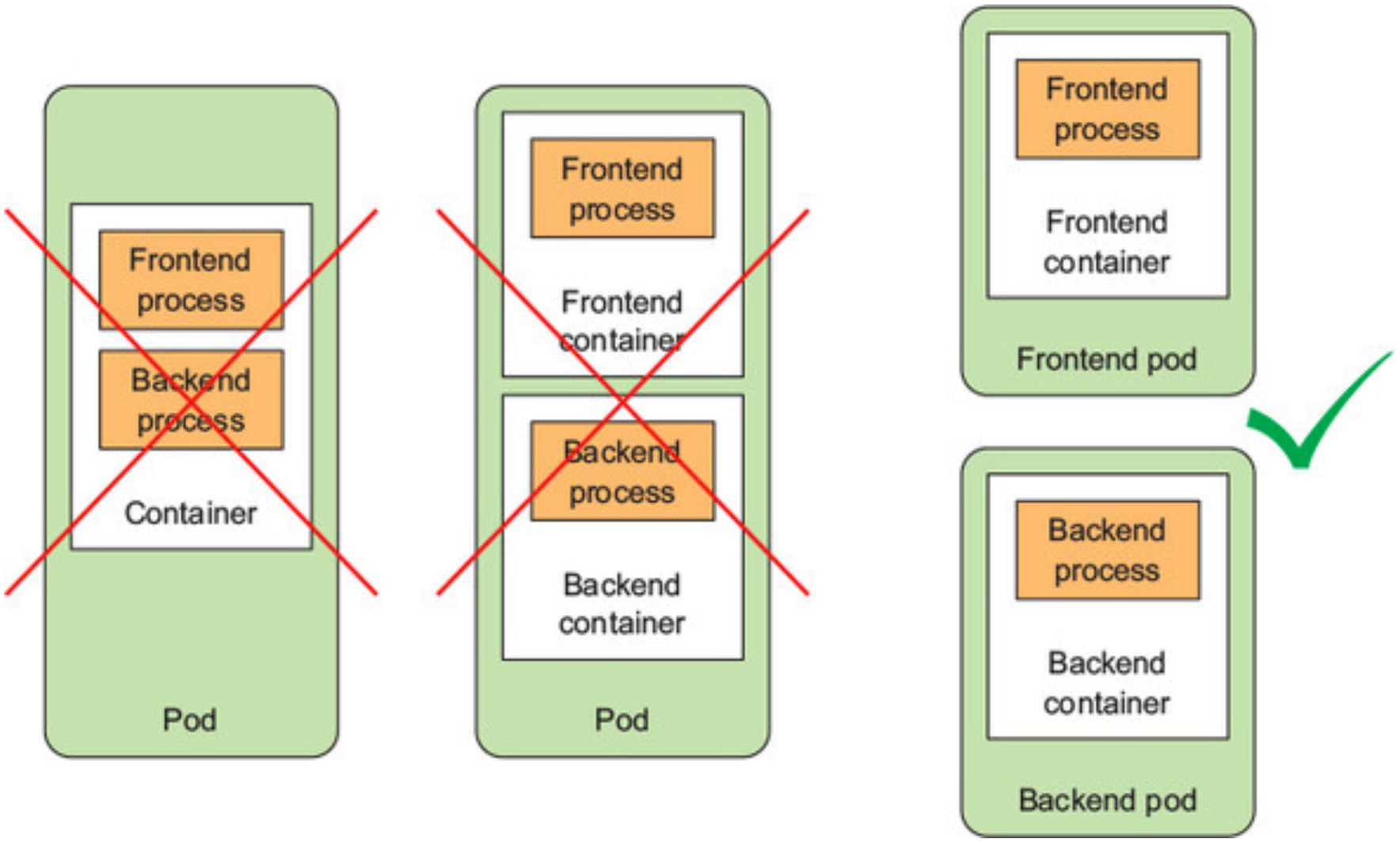
Share process ID
Share network interface
Share Hostname/IP/Port
Share Unix Time Sharing (UTS)

Pods



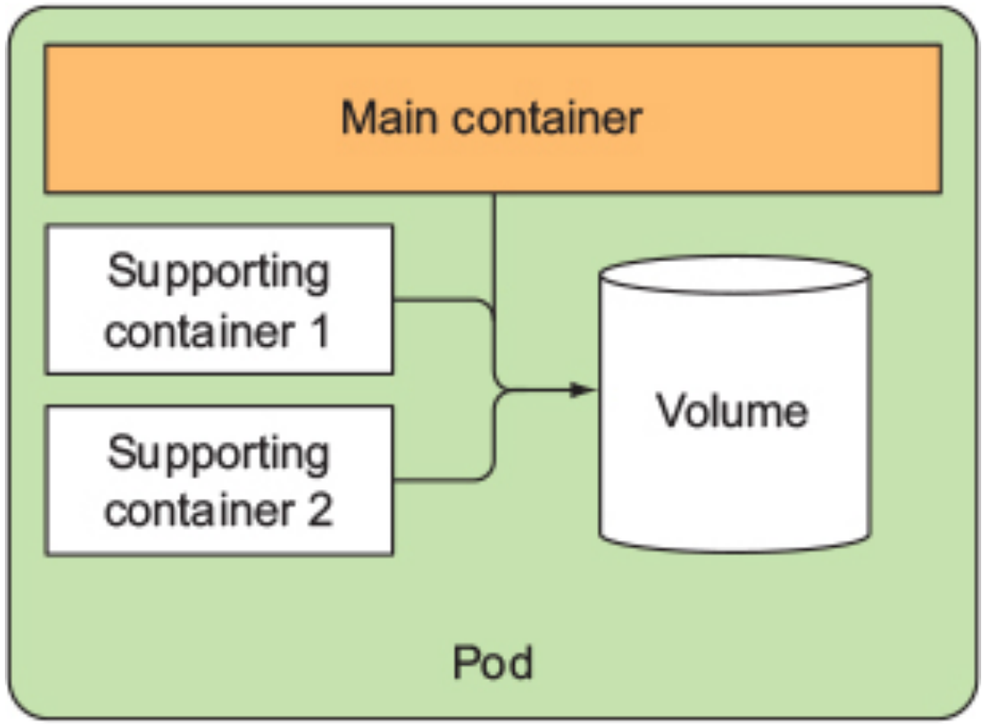
Organize container across Pods

Split multi-tiers app into multiple pods

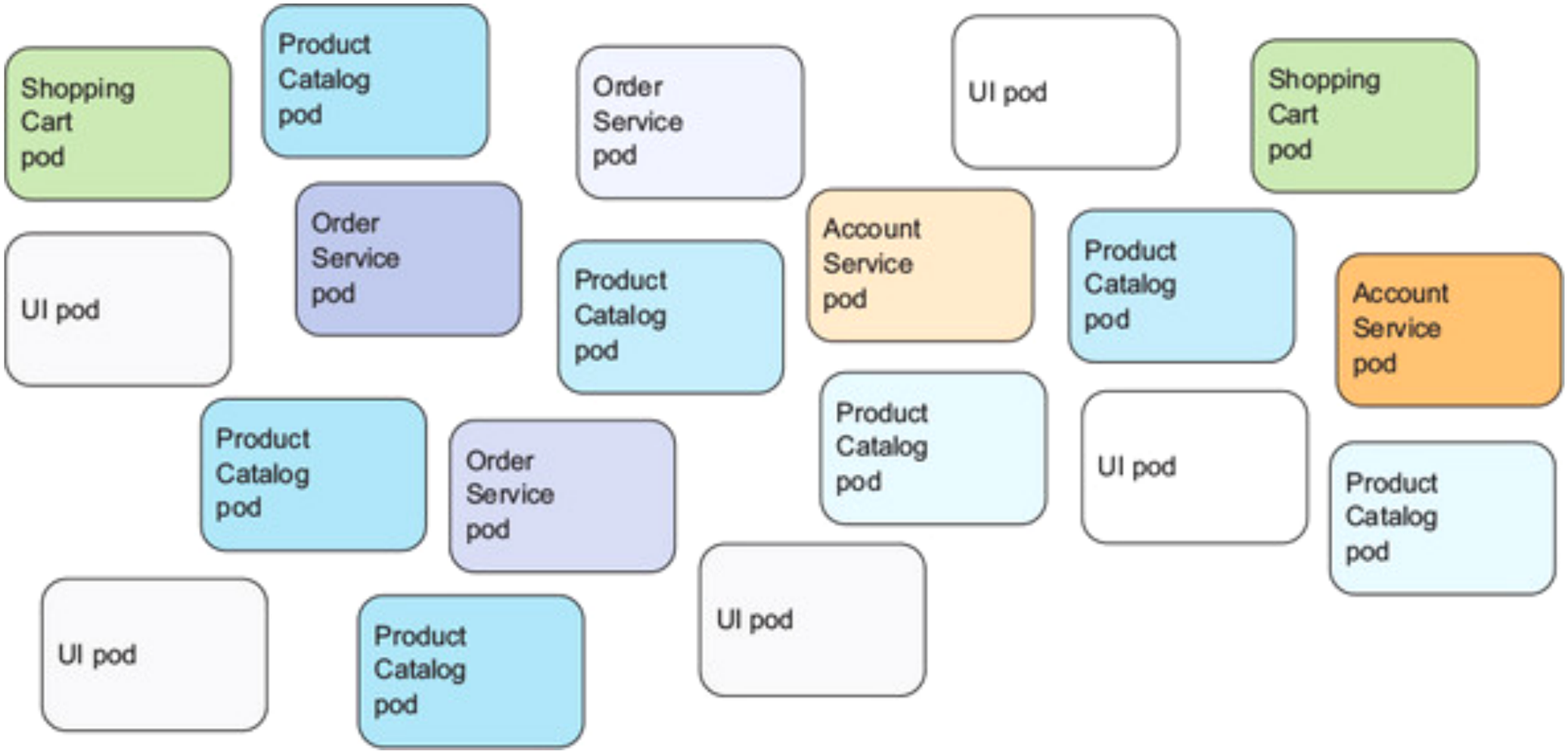


Organize container across Pods

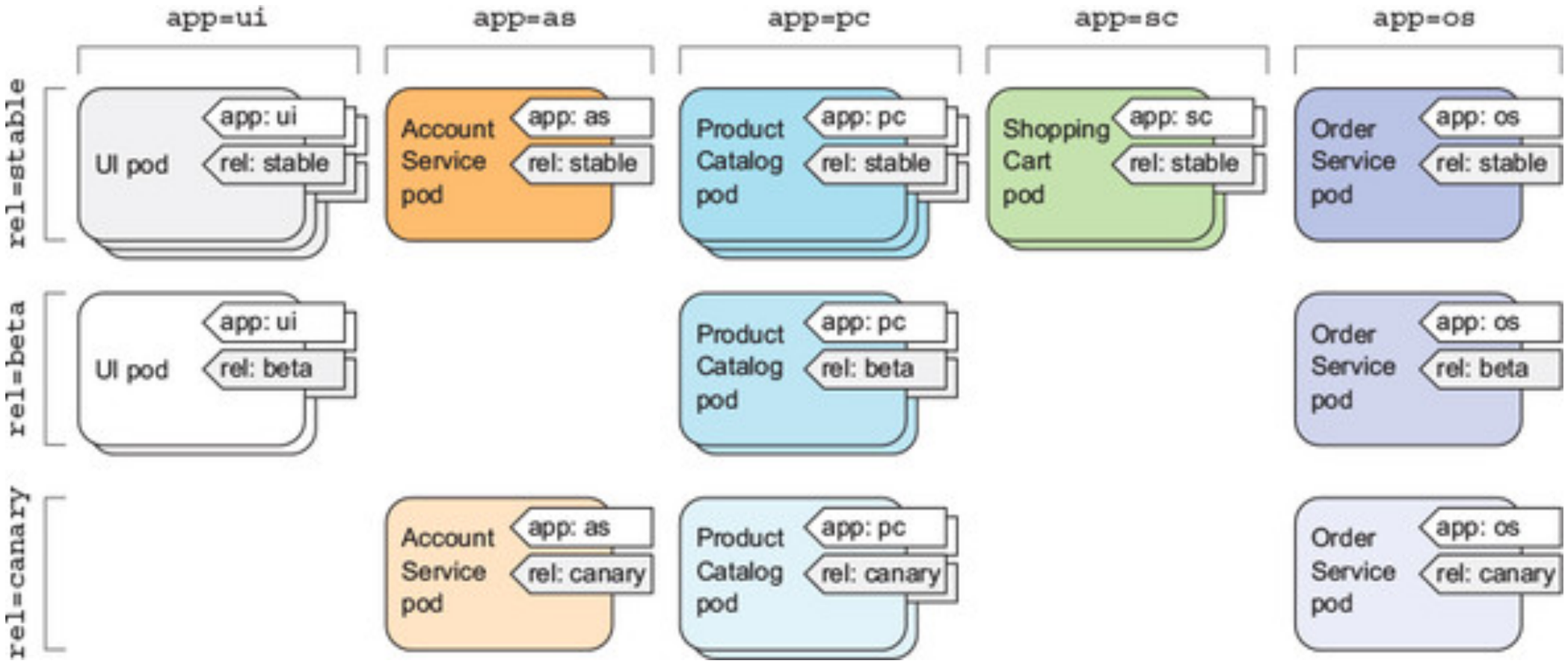
When to use multiple containers in a pods



Organize pods with Labels

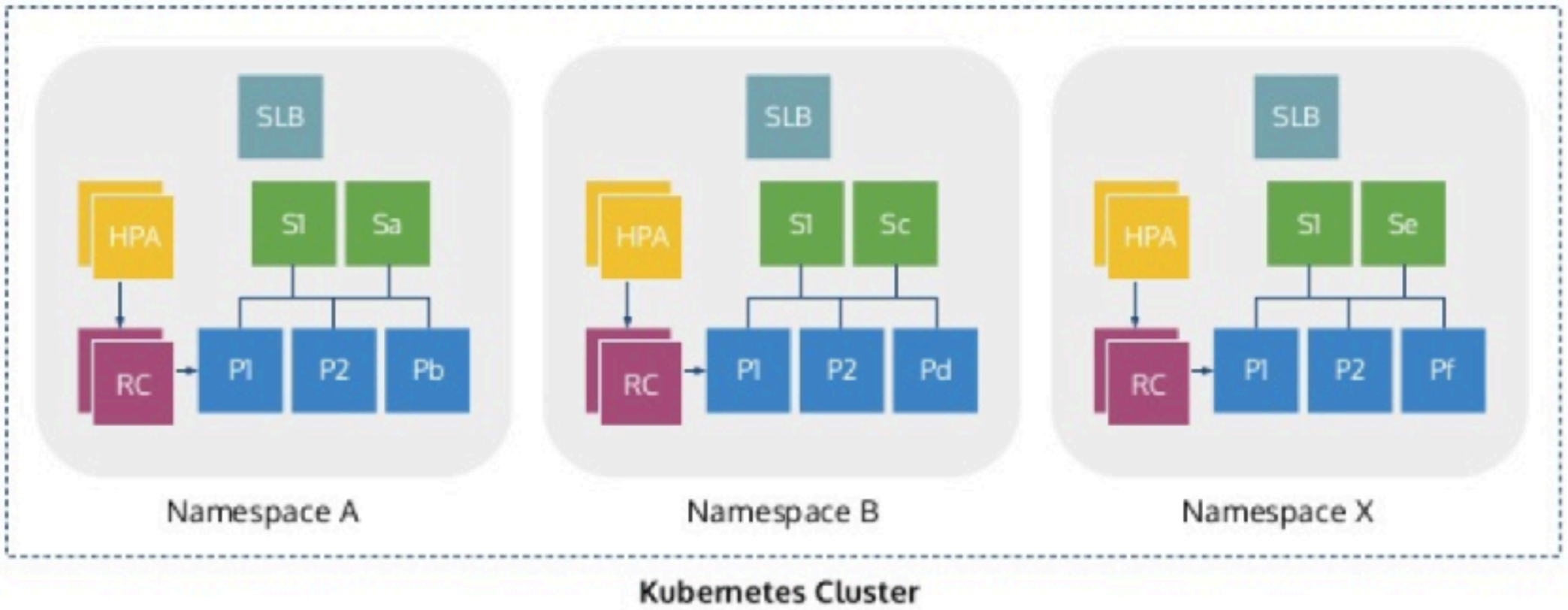


Organize pods with Labels



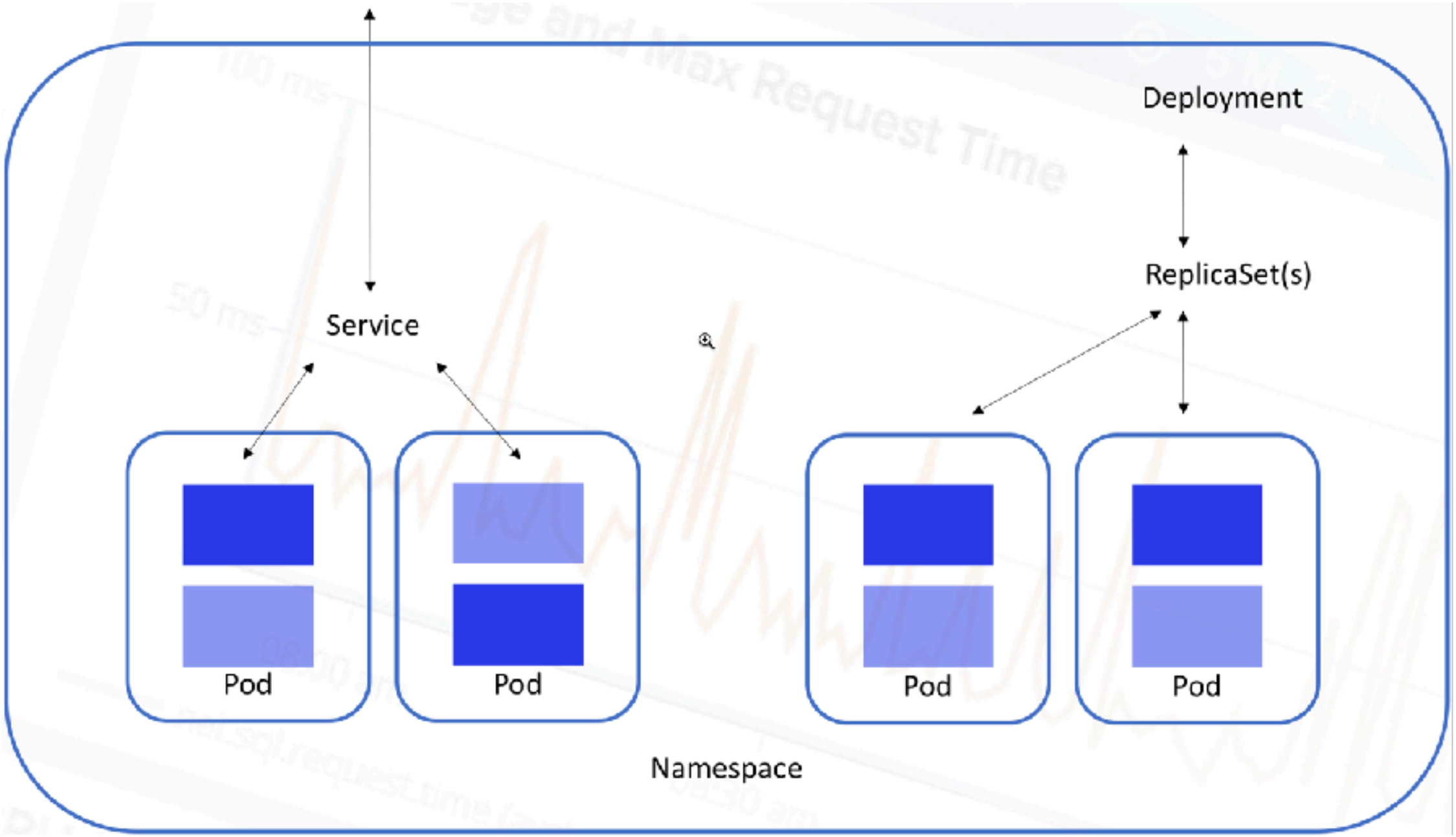
Pods namespaces

Allow different teams to use the same cluster

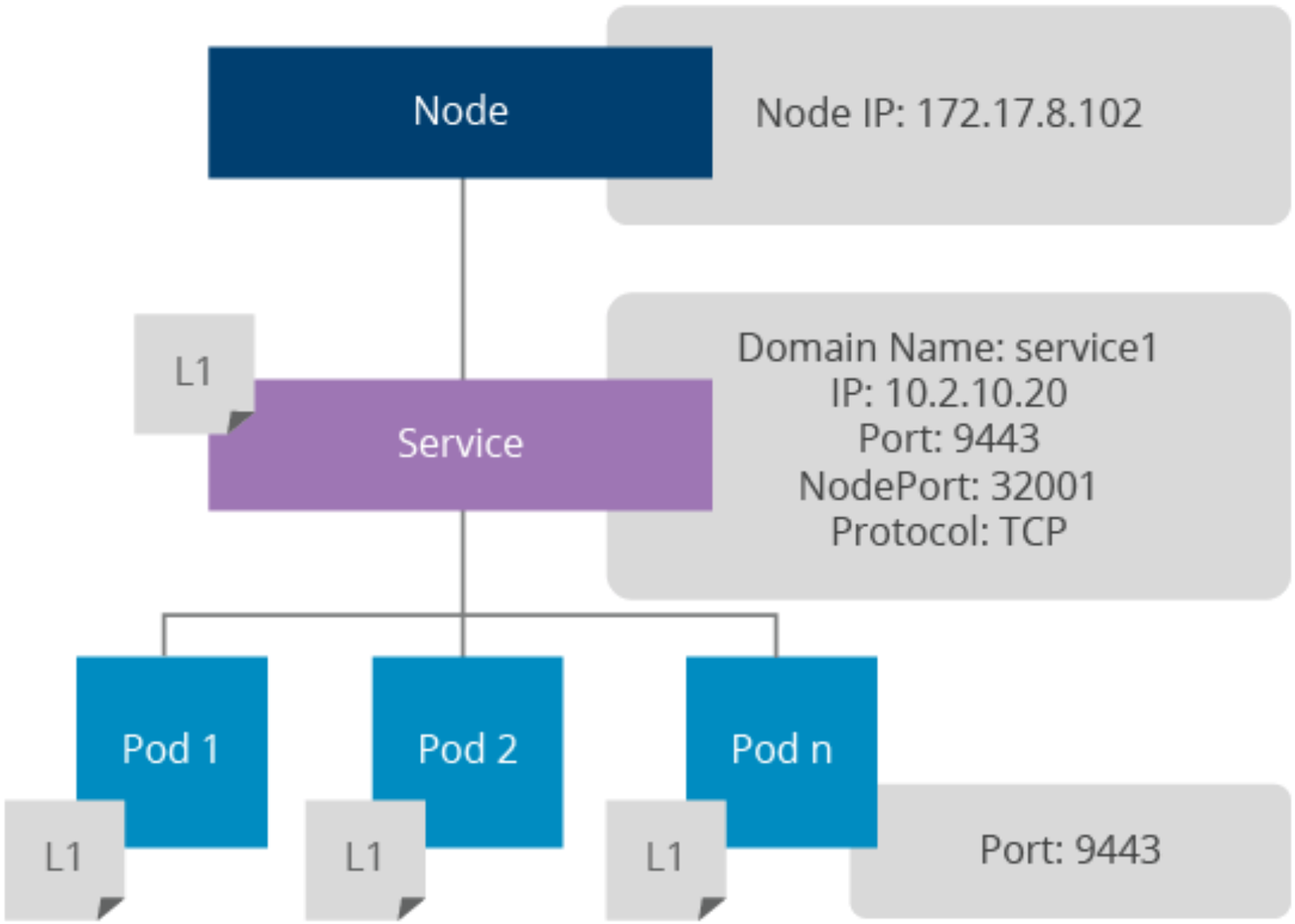


Services discovery and Load balancing

Services



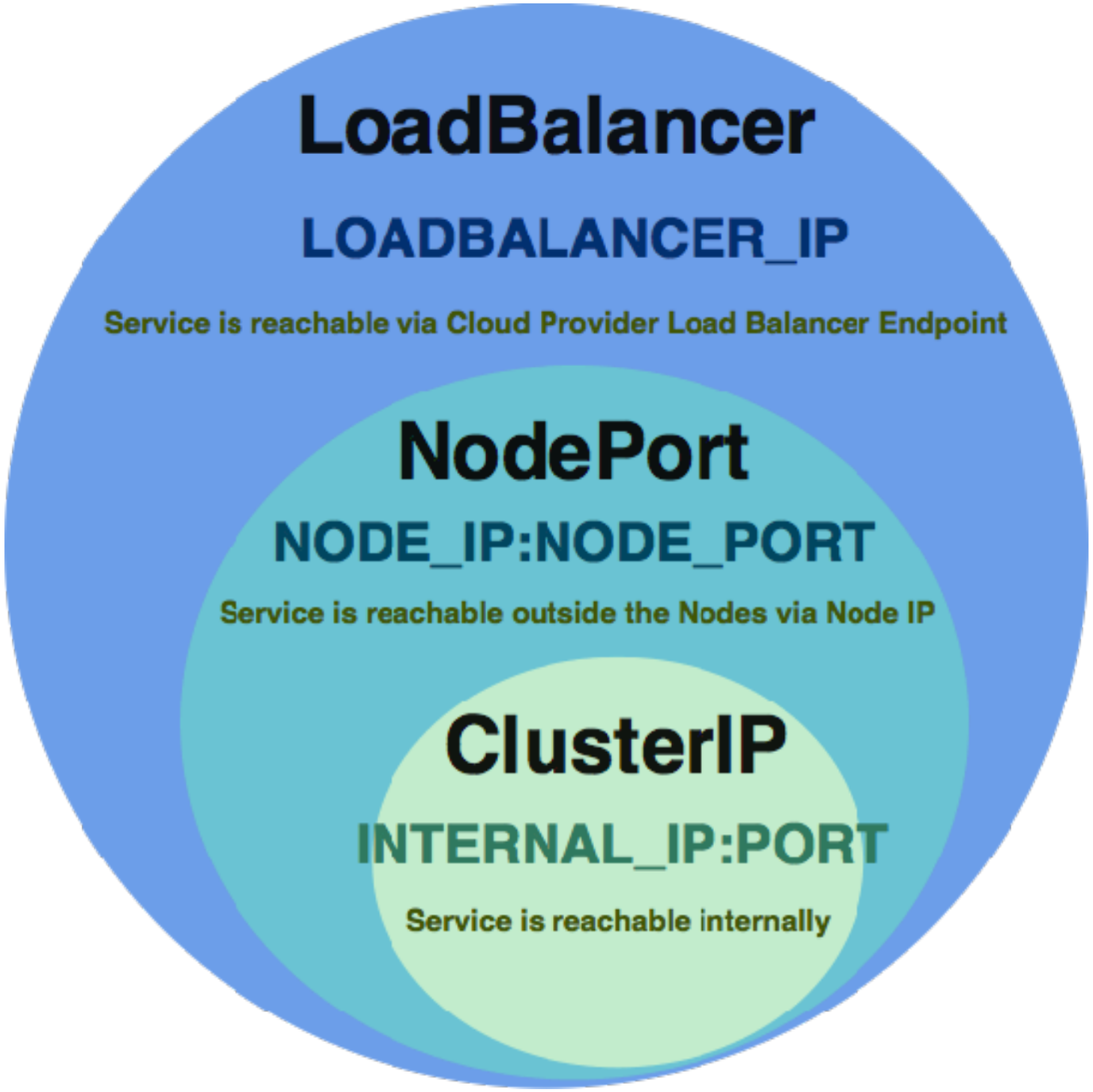
Services



Services

Independent from Pods
Abstraction layer of Pods
Provide load balance
Expose access Pods/Load balance
Find Pods by label selector

3 types of services



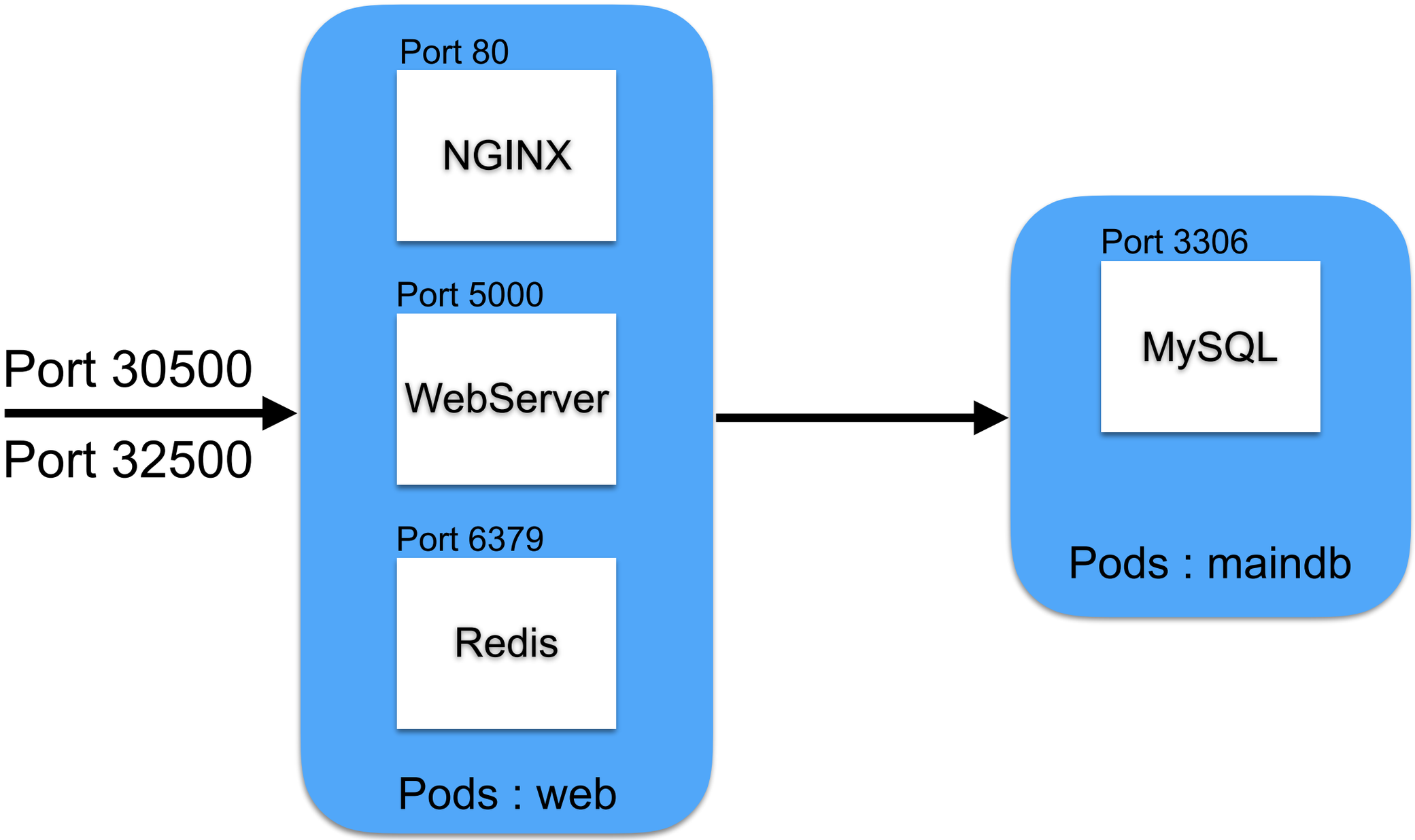
Workshop :: Pods & Services

1 container per Pods

Workshop :: Pods & Services

N container per Pods

Workshop



Replication Controller (RC)

Replication Controller

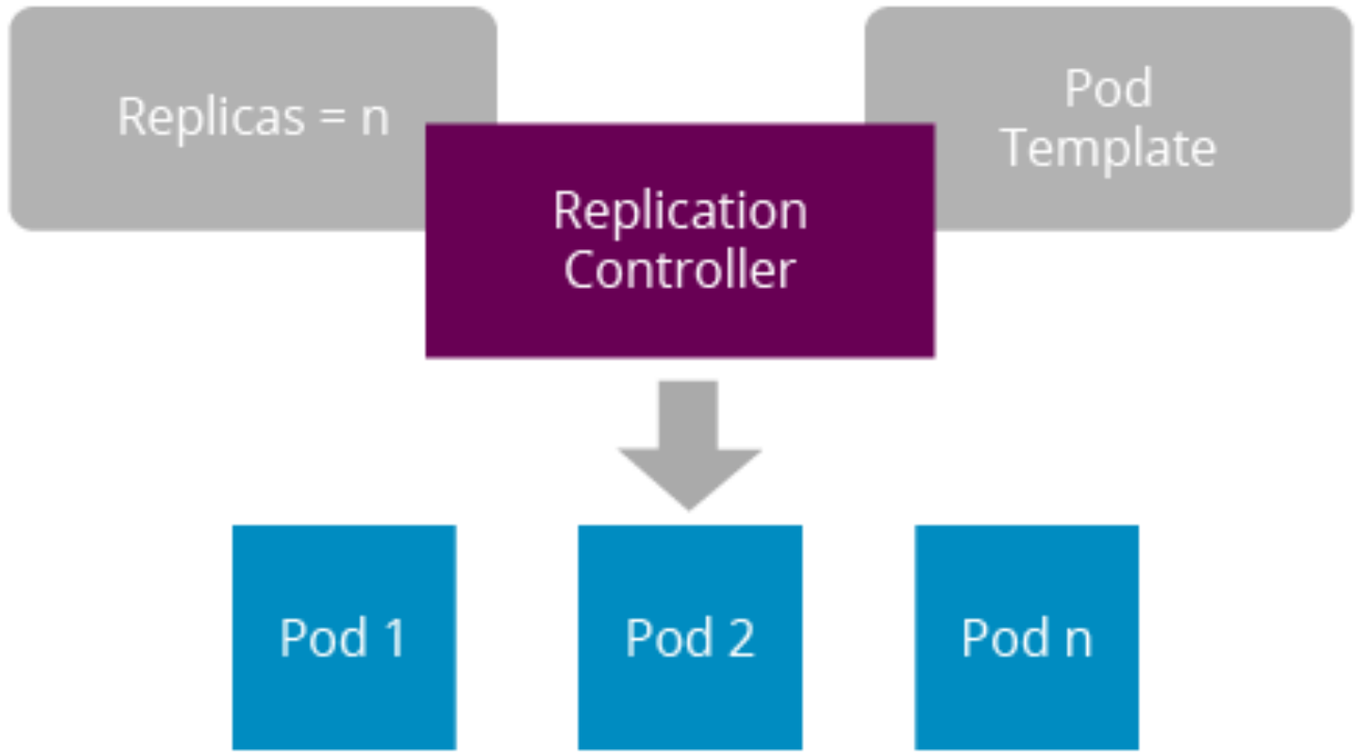
Create and maintain Pods
Keep copy of Pods by design
Maintain on cluster level
Auto-healing if Pods crash with any reason
Ensure Pods is up and run with desired number

Replication Controller

Running based on label type
Equality-based requirement

```
selector:
  name: web
  version: "1.0"
  module: WebServer
  environment: development
```

Replication Controller



Scale up replicas of RC

$kubectl **scale** <option>
--replicas=<number>
<type or name>

```
bash-3.2$ kubectl get rc
NAME      DESIRED   CURRENT   READY     AGE
hello     5         5         5         15m
bash-3.2$ kubectl get pods
NAME          READY     STATUS    RESTARTS   AGE
hello-jptv7   1/1       Running   0          43s
hello-kdbsc   1/1       Running   0          43s
hello-rn545   1/1       Running   0          15m
hello-w6t5q   1/1       Running   0          43s
hello-x6x8k   1/1       Running   0          43s
```

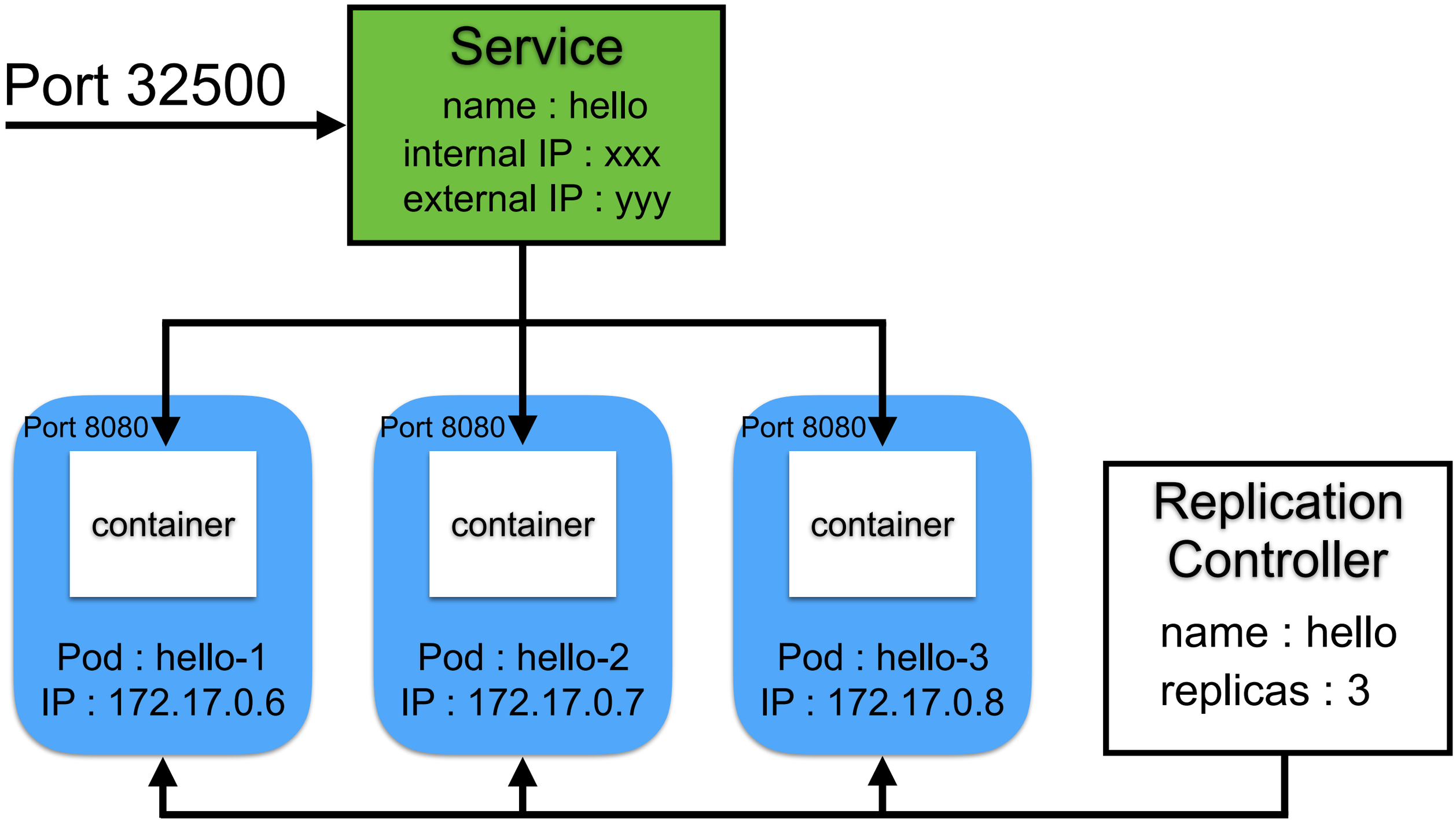
Detail of RC

$kubectl **describe rc <name>**

```
  Containers:
   hello:
    Image:        somkiat/hello:latest
    Port:         8080/TCP
    Host Port:    0/TCP
    Environment:  <none>
    Mounts:       <none>
  Volumes:        <none>
Events:
Type    Reason             Age   From                    Message
----    ------             ----  ----                    -------
Normal  SuccessfulCreate   16m   replication-controller  Created pod: hello-rn545
Normal  SuccessfulCreate   16m   replication-controller  Created pod: hello-pfwnj
Normal  SuccessfulCreate   16m   replication-controller  Created pod: hello-jfqfl
Normal  SuccessfulCreate   13m   replication-controller  Created pod: hello-6pfqm
Normal  SuccessfulCreate   11m   replication-controller  Created pod: hello-l4lqc
Normal  SuccessfulCreate   11m   replication-controller  Created pod: hello-pfgr2
Normal  SuccessfulDelete   9m    replication-controller  Deleted pod: hello-l4lqc
Normal  SuccessfulDelete   9m    replication-controller  Deleted pod: hello-pfwnj
Normal  SuccessfulDelete   9m    replication-controller  Deleted pod: hello-6pfqm
Normal  SuccessfulDelete   9m    replication-controller  Deleted pod: hello-pfgr2
Normal  SuccessfulCreate   2m    replication-controller  Created pod: hello-kdbsc
Normal  SuccessfulCreate   2m    replication-controller  Created pod: hello-jptv7
Normal  SuccessfulCreate   2m    replication-controller  Created pod: hello-x6x8k
```

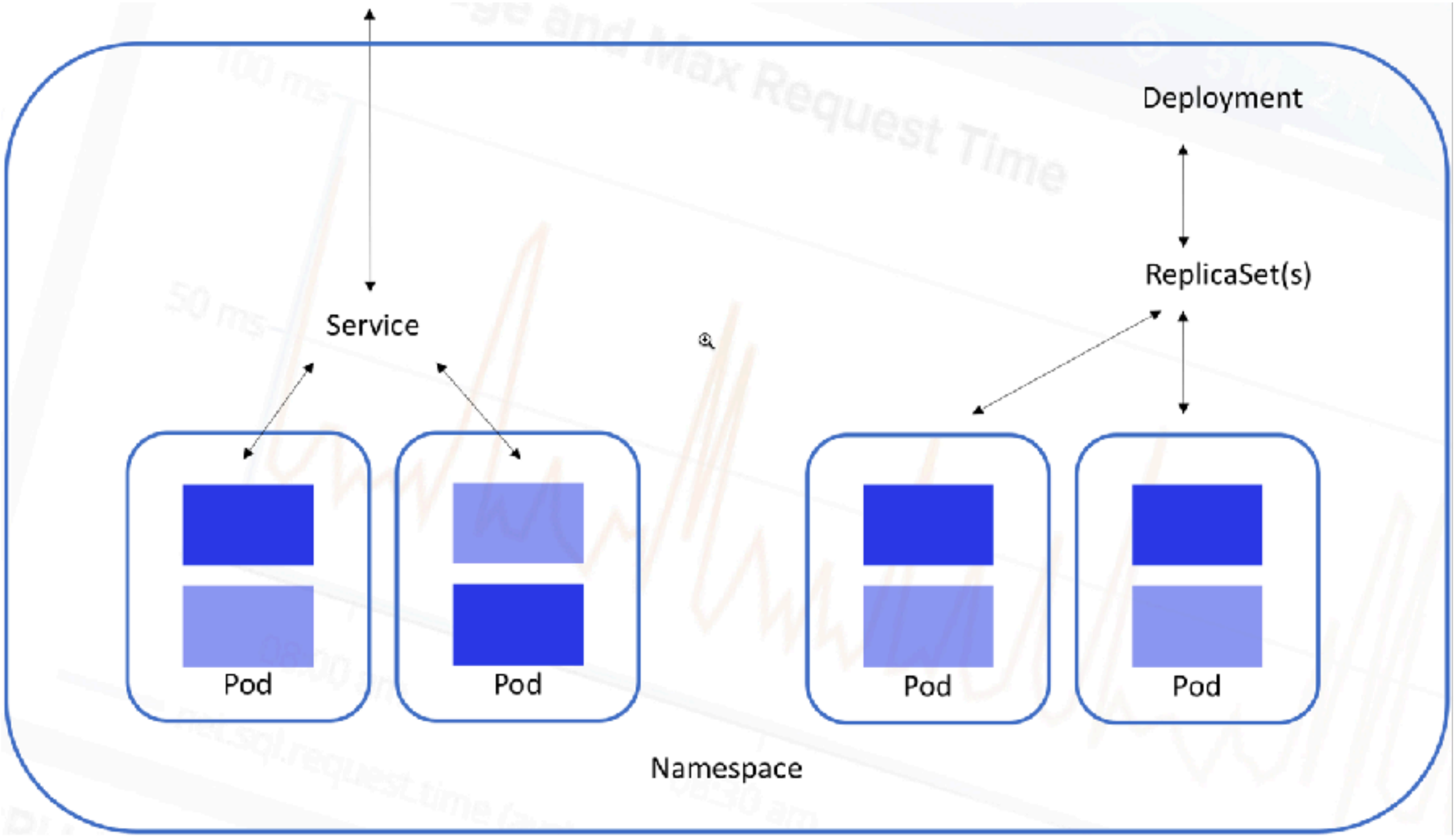
Workshop
Replication Controller

Scaling the application

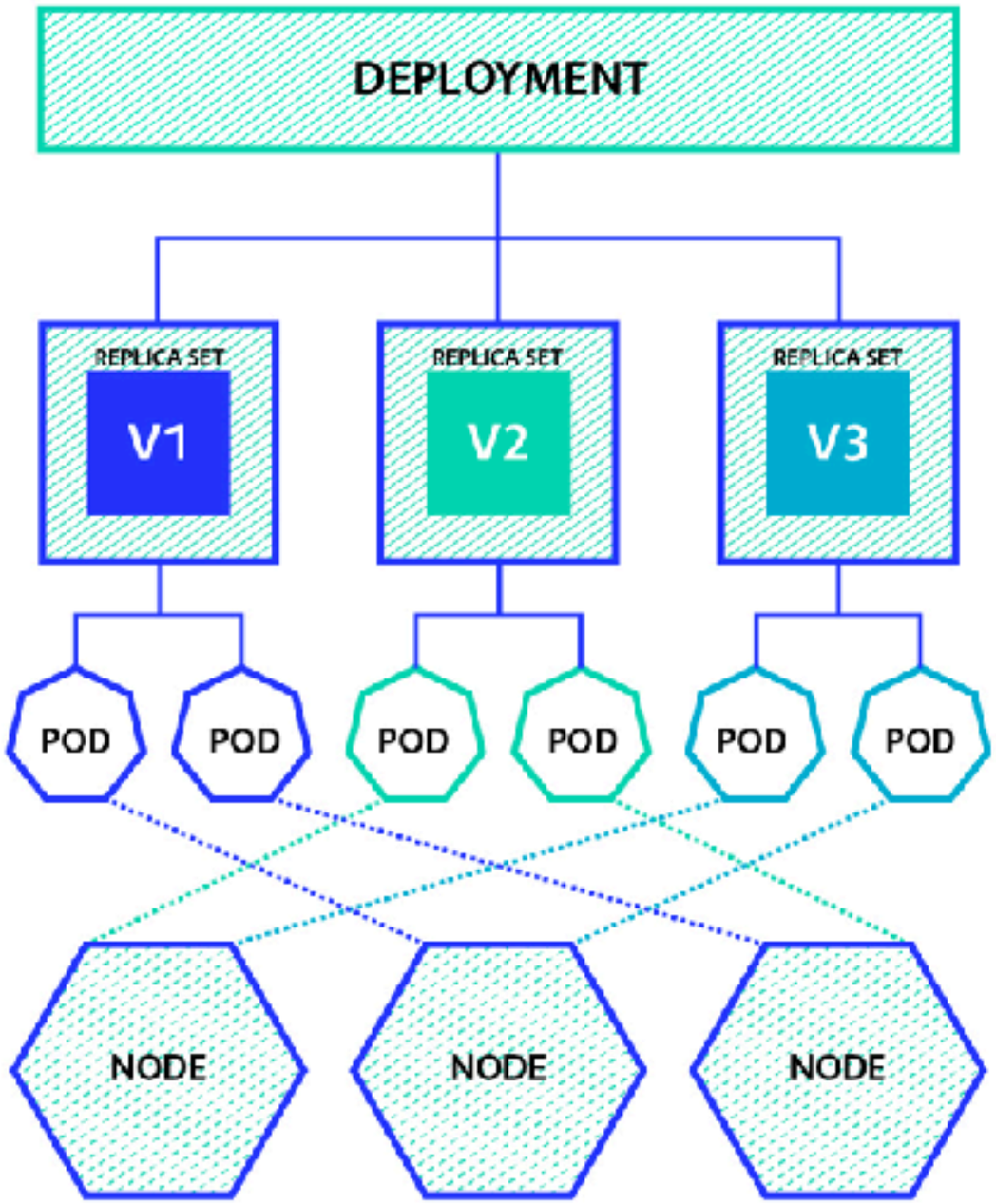


Deployment and ReplicaSet (RS)

Deployment and ReplicaSet



Deployment and ReplicaSet



https://thenewstack.io/kubernetes-deployments-work/

Deployment and RS

Next-generation of RC
Provide function to maintain versioning of Pods
Update new version (Rollout)
Revert to old version (Rollback)
Scale a deployment
Pause process
Resume process

New version from Deployment

Create new RS
Start to scale as desired
Scale down existing RS to 0
Delete existing RS

Deployment/RS vs RC

RS more dynamic than RC

RS support label with methods:

Equality-based requirement

Set-based requirement

Rollout strategy

Set online

Edit online

Modify YAML file and apply

Set online

$kubectl set **image** deployment/hello hello=somkiat/hello:v2

```
Waiting for rollout to finish: 1 out of 3 new replicas have been updated...
Waiting for rollout to finish: 1 out of 3 new replicas have been updated...
Waiting for rollout to finish: 1 out of 3 new replicas have been updated...
Waiting for rollout to finish: 1 out of 3 new replicas have been updated...
Waiting for rollout to finish: 1 out of 3 new replicas have been updated...
Waiting for rollout to finish: 1 out of 3 new replicas have been updated...
Waiting for rollout to finish: 2 out of 3 new replicas have been updated...
Waiting for rollout to finish: 2 out of 3 new replicas have been updated...
Waiting for rollout to finish: 2 old replicas are pending termination...
Waiting for rollout to finish: 1 old replicas are pending termination...
Waiting for rollout to finish: 1 old replicas are pending termination...
deployment "hello" successfully rolled out
```

Edit online

$kubectl **edit deployment** hello

```
# Please edit the object below. Lines beginning with a '#' will be ignored,
# and an empty file will abort the edit. If an error occurs while saving this f
# reopened with the relevant failures.
#
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  annotations:
    deployment.kubernetes.io/revision: "6"
    kubectl.kubernetes.io/last-applied-configuration: |
      {"apiVersion":"apps/v1","kind":"Deployment","metadata":{"annotations":{},
","namespace":"default"},"spec":{"replicas":3,"revisionHistoryLimit":1,"selecto
mplate":{"metadata":{"labels":{"app":"web"}},"spec":{"containers":[{"image":"sc
s":[{"containerPort":8080,"protocol":"TCP"}]}]}}}}
  creationTimestamp: 2018-04-01T17:17:51Z
  generation: 10
  labels:
    app: web
  name: hello
  namespace: default
  resourceVersion: "33405"
  selfLink: /apis/extensions/v1beta1/namespaces/default/deployments/hello
  uid: 9b773859-35d0-11e8-9d36-0800275c6c60
```

Modify YAML file and apply

$kubectl apply -f hello_deployment.yml

Show history of rollout

$kubectl rollout history deployment/hello

```
deployments "hello"
REVISION  CHANGE-CAUSE
2         <none>
3         <none>
4         <none>
```

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: hello
  labels:
    app: web
spec:
  replicas: 3
  revisionHistoryLimit: 1
  selector:
    matchLabels:
      app: web
```

spec.revisionHistory = 2 (default)
spec.revisionHistory = 0 (clean all)

Try to rollback to revision

$kubectl rollout undo deployment/hello —to-revision=2

Try to scale a deployment

$kubectl scale deployment hello --replicas=5

```
bash-3.2$ kubectl scale deployment hello --replicas=3
deployment.extensions "hello" scaled
bash-3.2$ kubectl get deployment -o wide
NAME      DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE       CONTAINERS   IMAGES           SELECTOR
hello     3         3         3            3           21m       hello        somkiat/hello    app=web
bash-3.2$ kubectl scale deployment hello --replicas=5
deployment.extensions "hello" scaled
bash-3.2$ kubectl get deployment -o wide
NAME      DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE       CONTAINERS   IMAGES           SELECTOR
hello     5         5         5            5           21m       hello        somkiat/hello    app=web
bash-3.2$ kubectl scale deployment hello --replicas=1
deployment.extensions "hello" scaled
bash-3.2$ kubectl get deployment -o wide
NAME      DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE       CONTAINERS   IMAGES           SELECTOR
hello     1         1         1            1           21m       hello        somkiat/hello    app=web
```

Pause and resume rollout

$kubectl **rollout pause** deployment/hello

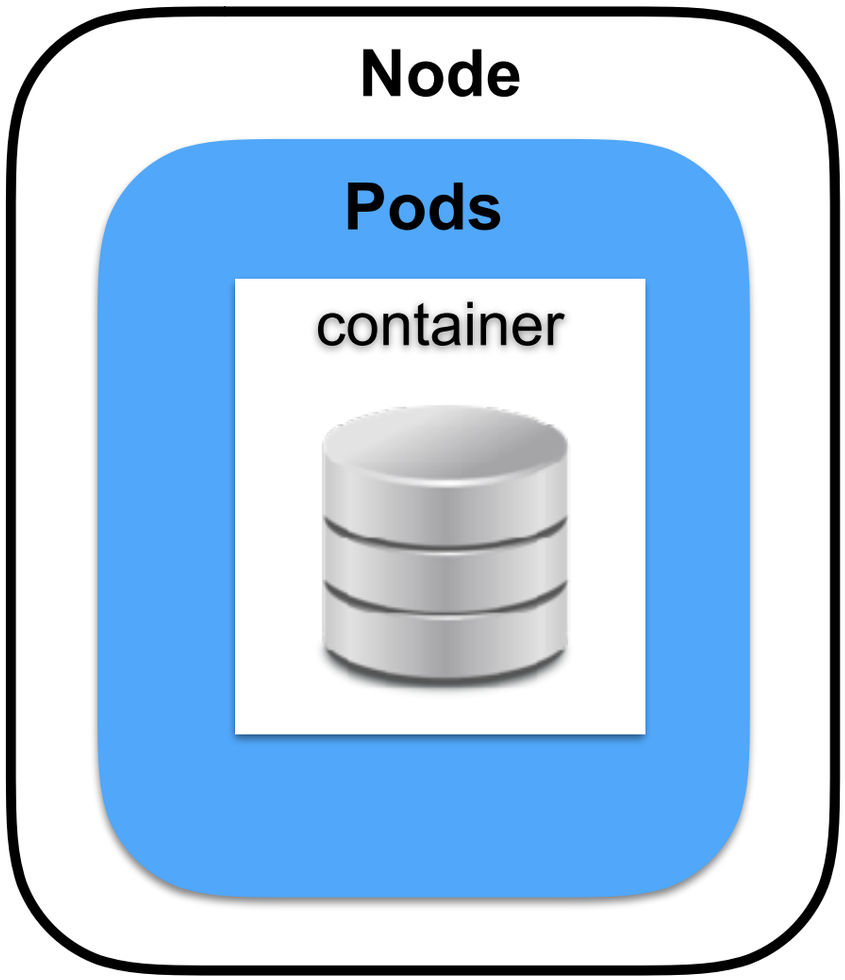
$kubectl **rollout resume** deployment/hello

Workshop Deployment/RS

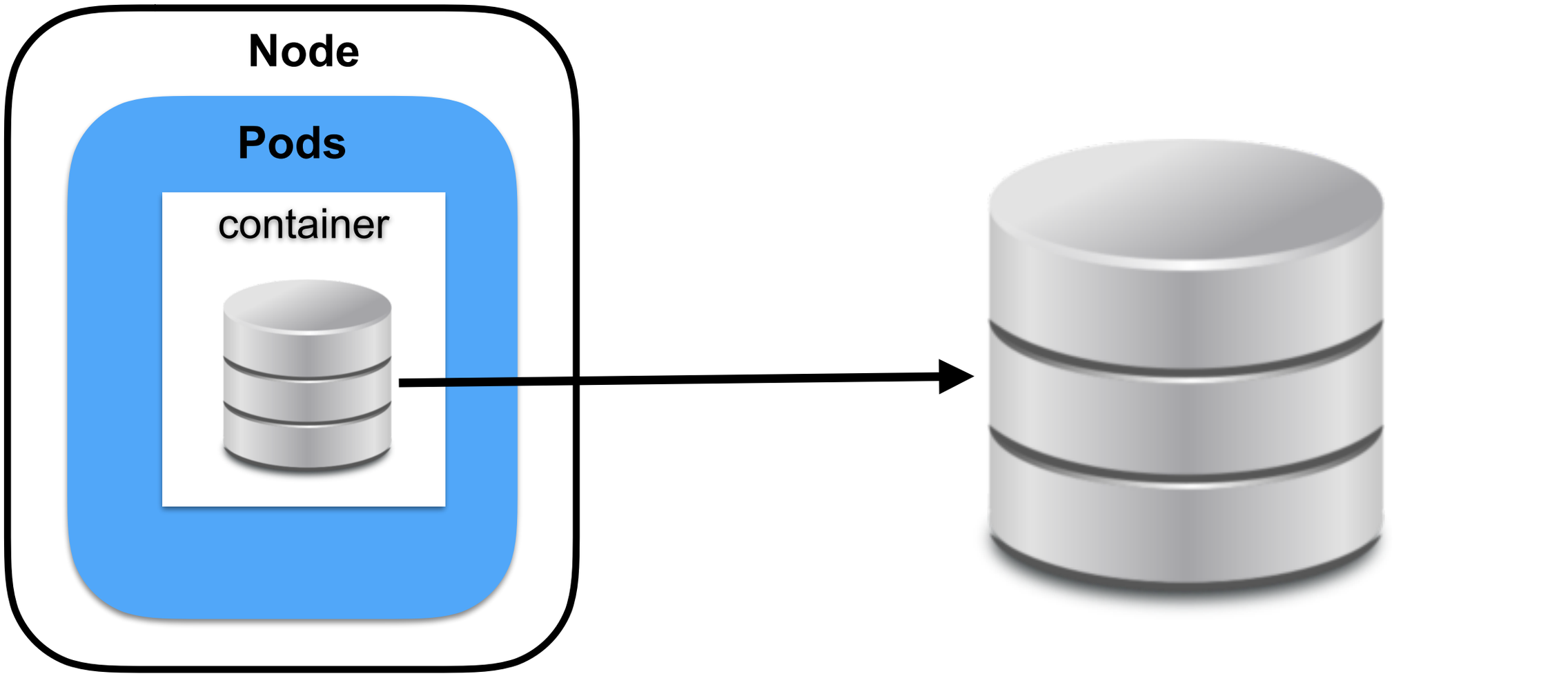
Volume

https://kubernetes.io/docs/concepts/storage/volumes/#resources

Volume



Volume



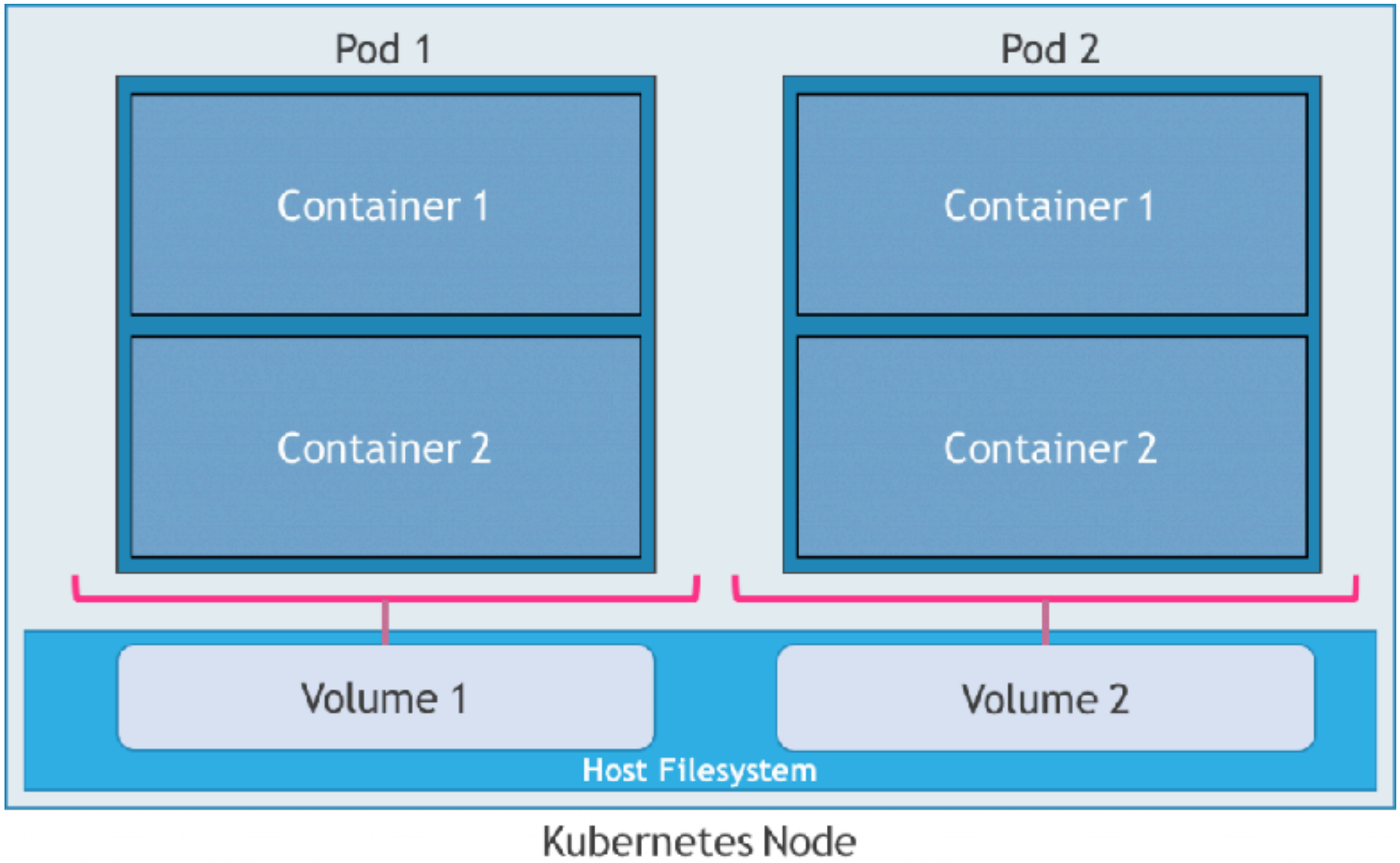
Volume

Containers in the same Pods share storage
(EmptyDir)

- Read and write for all containers in Pods
- Container crash not effect to this storage
- When Pods is deleted then Empty will deleted

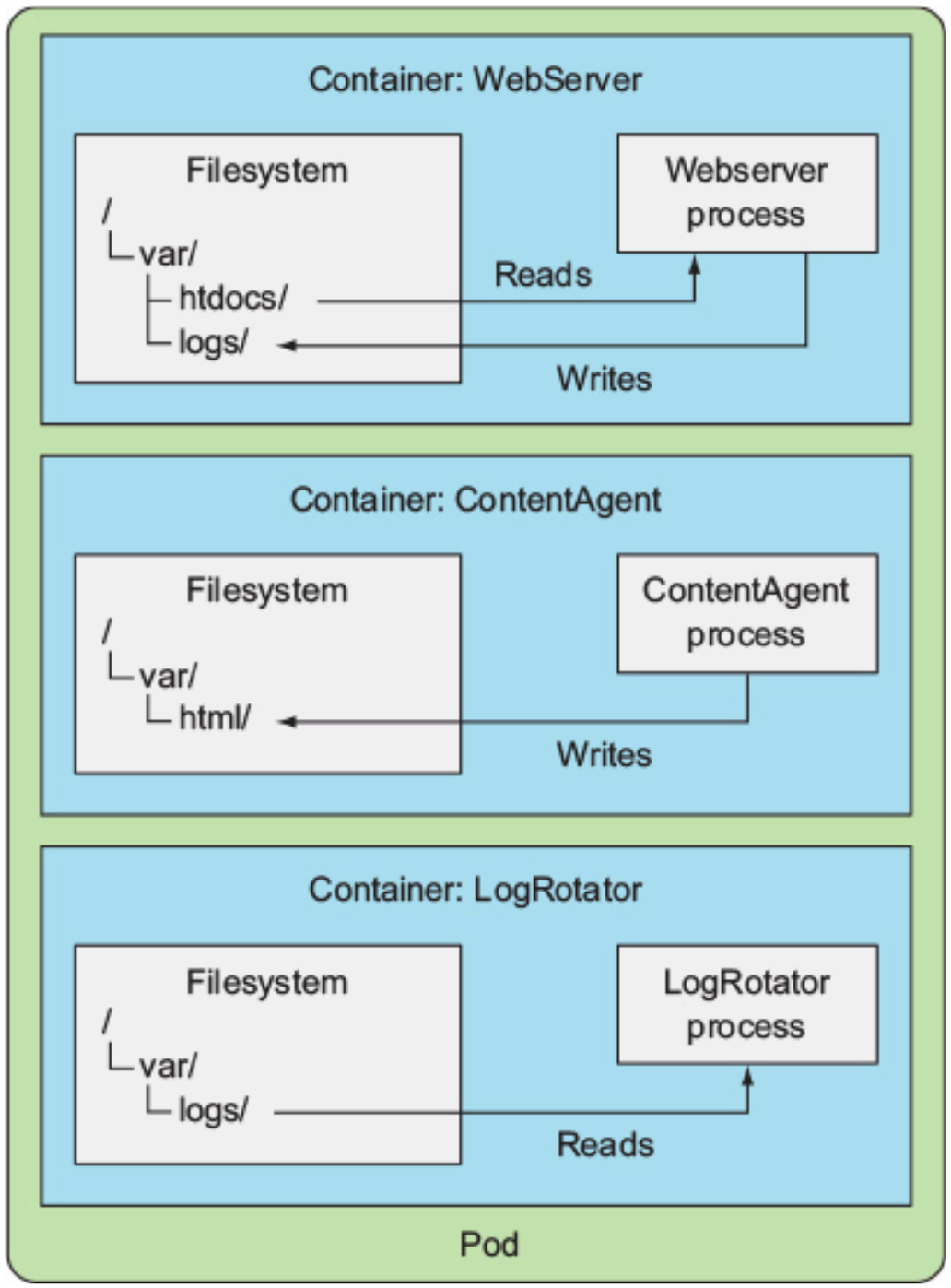
Data of Pods/container is **ephemeral**
all data may loss when Pods is restarted

Containers shared volume

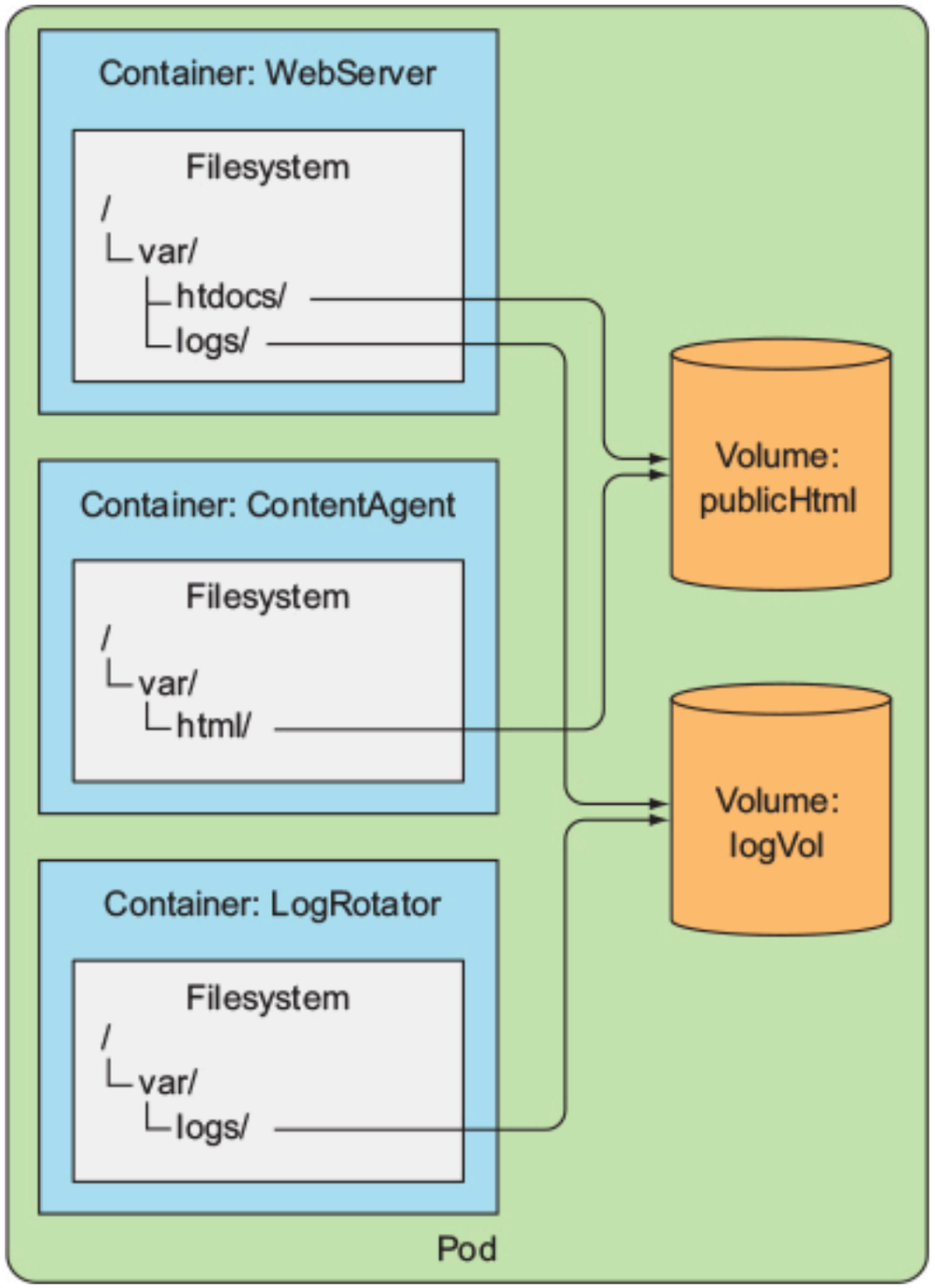


https://thenewstack.io/strategies-running-stateful-applications-kubernetes-volumes/

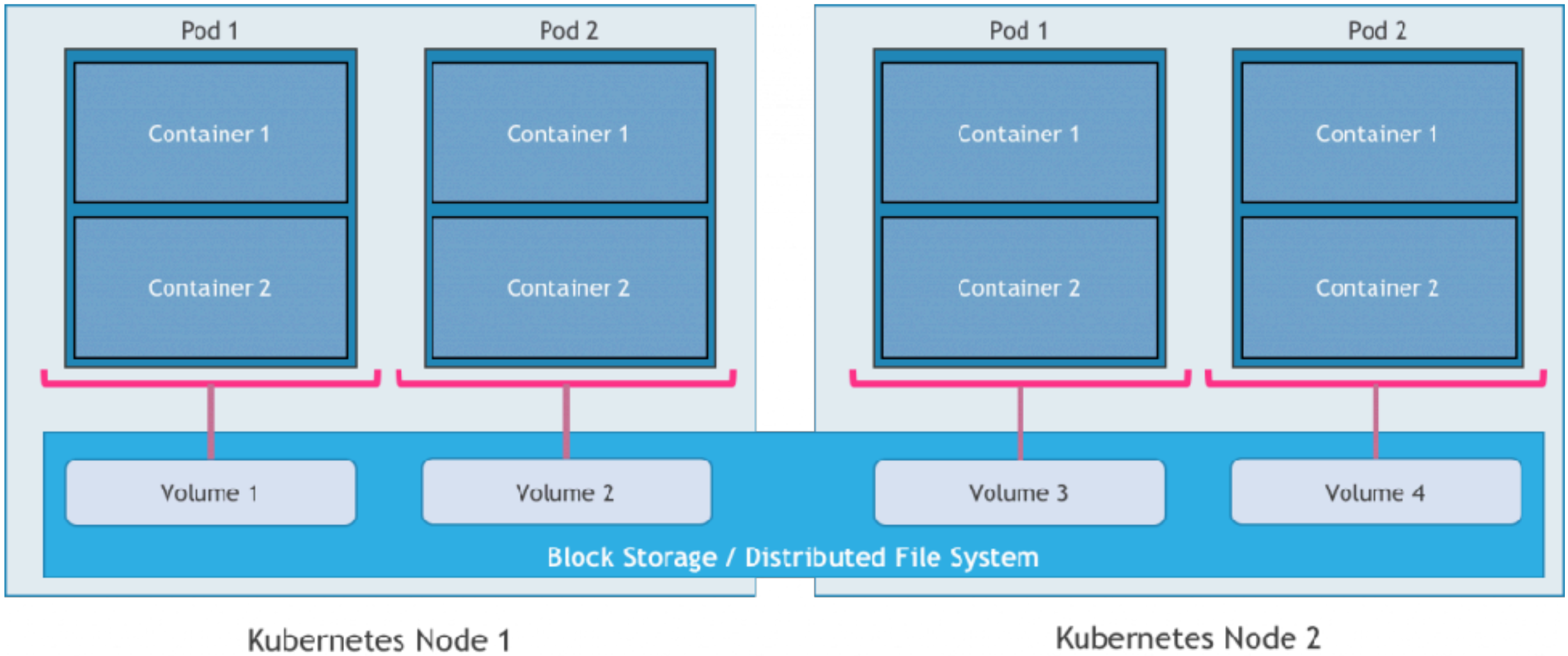
Without shared storage



Containers shared volume



Nodes shared volume



https://thenewstack.io/strategies-running-stateful-applications-kubernetes-volumes/

Volume on Kubernetes

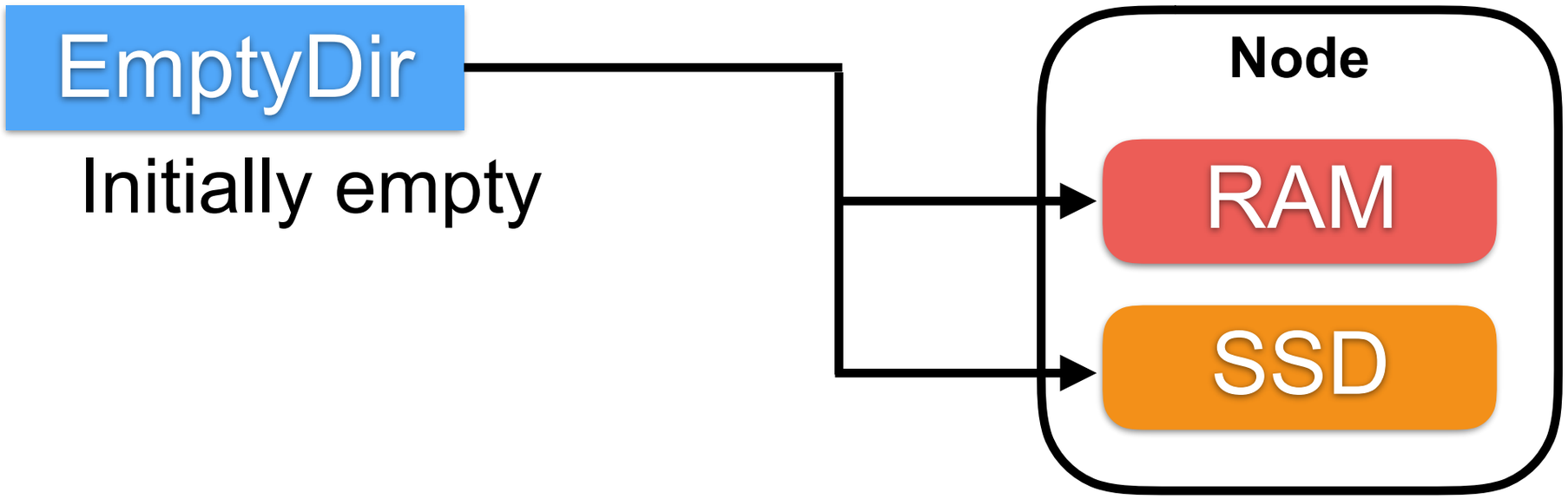
EmptyDir
hostPath
gitRepo
NFS
icePersistentDisk
Flocker
Persistent Volume Claim (PVC)
configMap, secret

https://kubernetes.io/docs/concepts/storage/volumes/

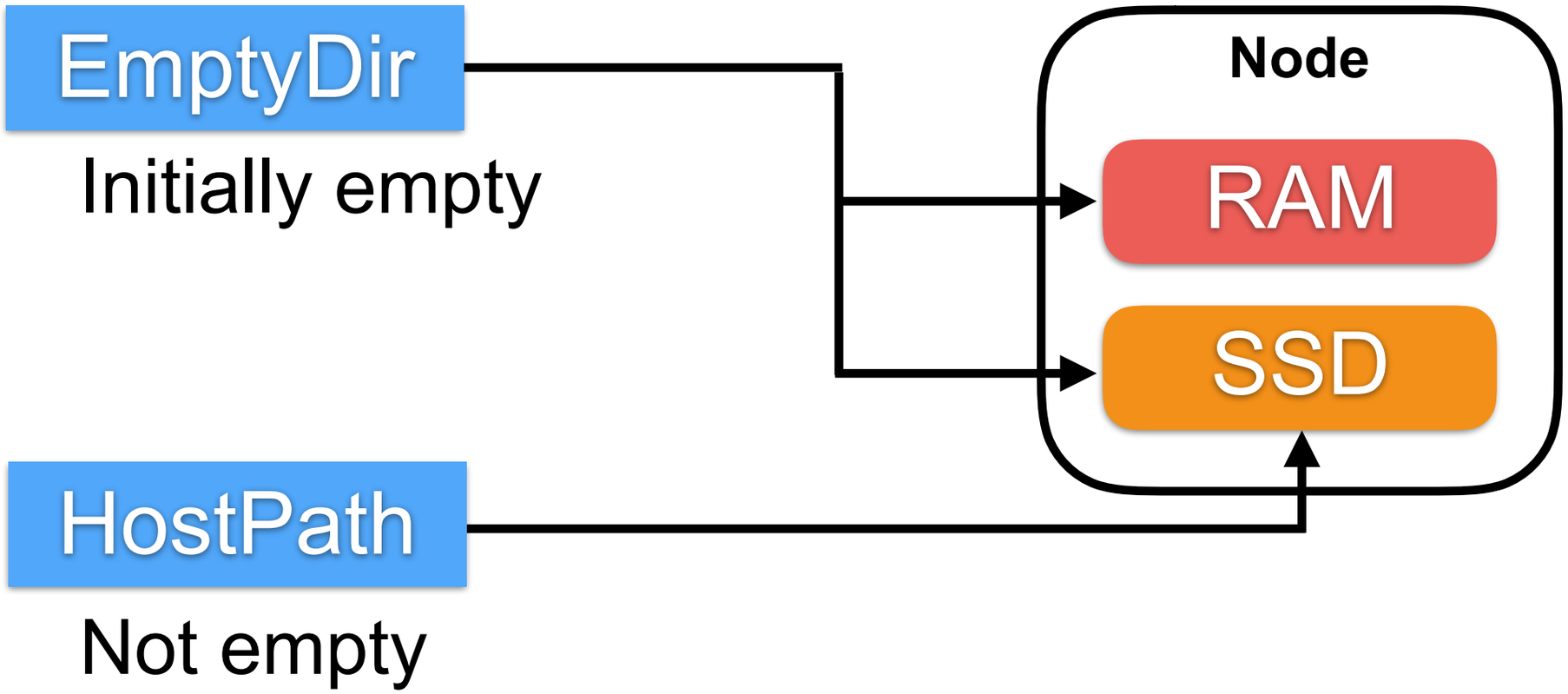
Kubernetes volumes

- EmptyDir
- HostPath
- NFS
- Cloud Volume
- Persistent Volume Claim

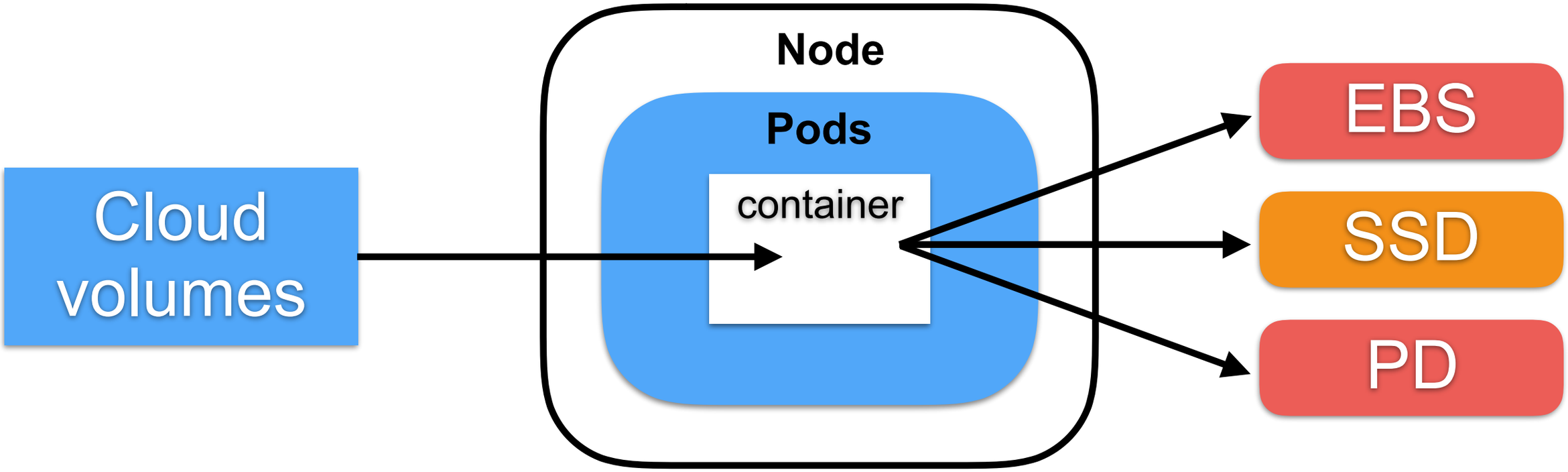
Kubernetes volumes



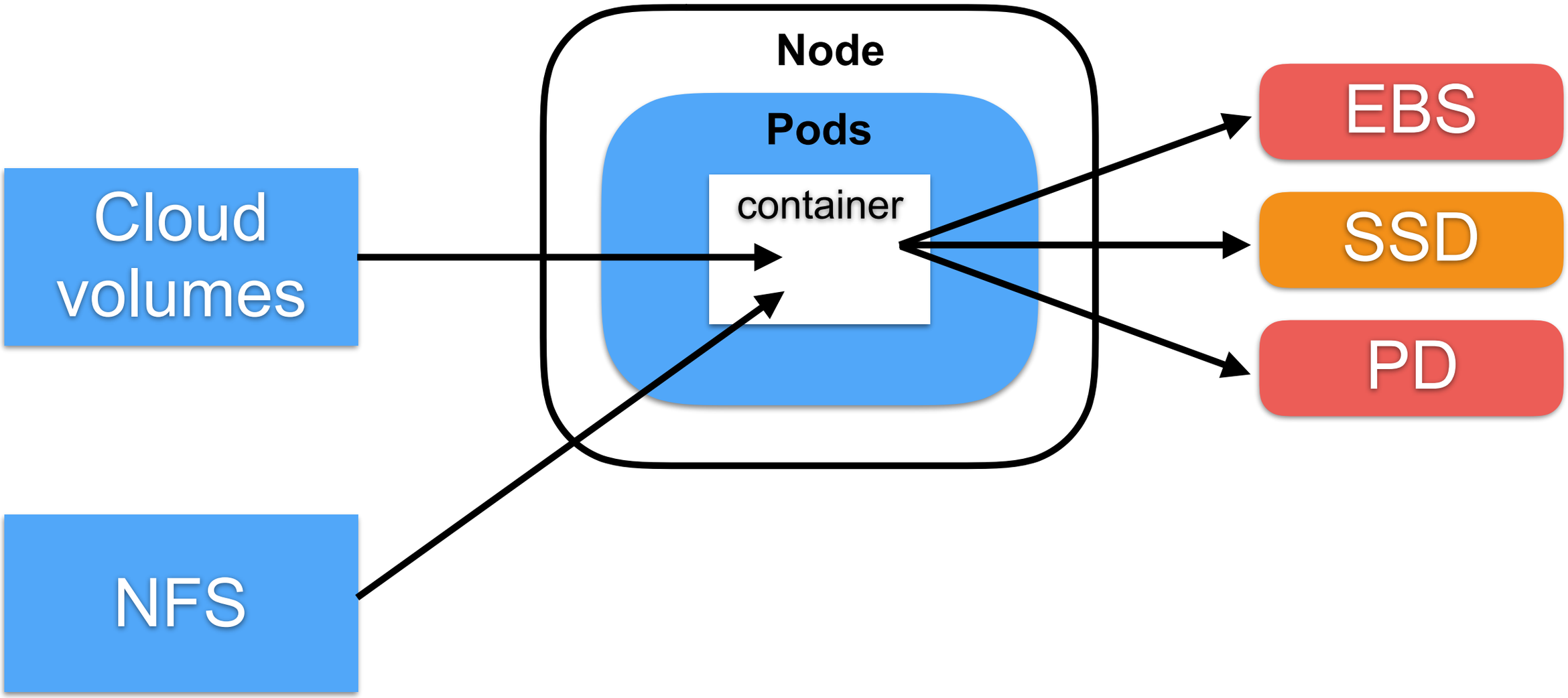
Kubernetes volumes



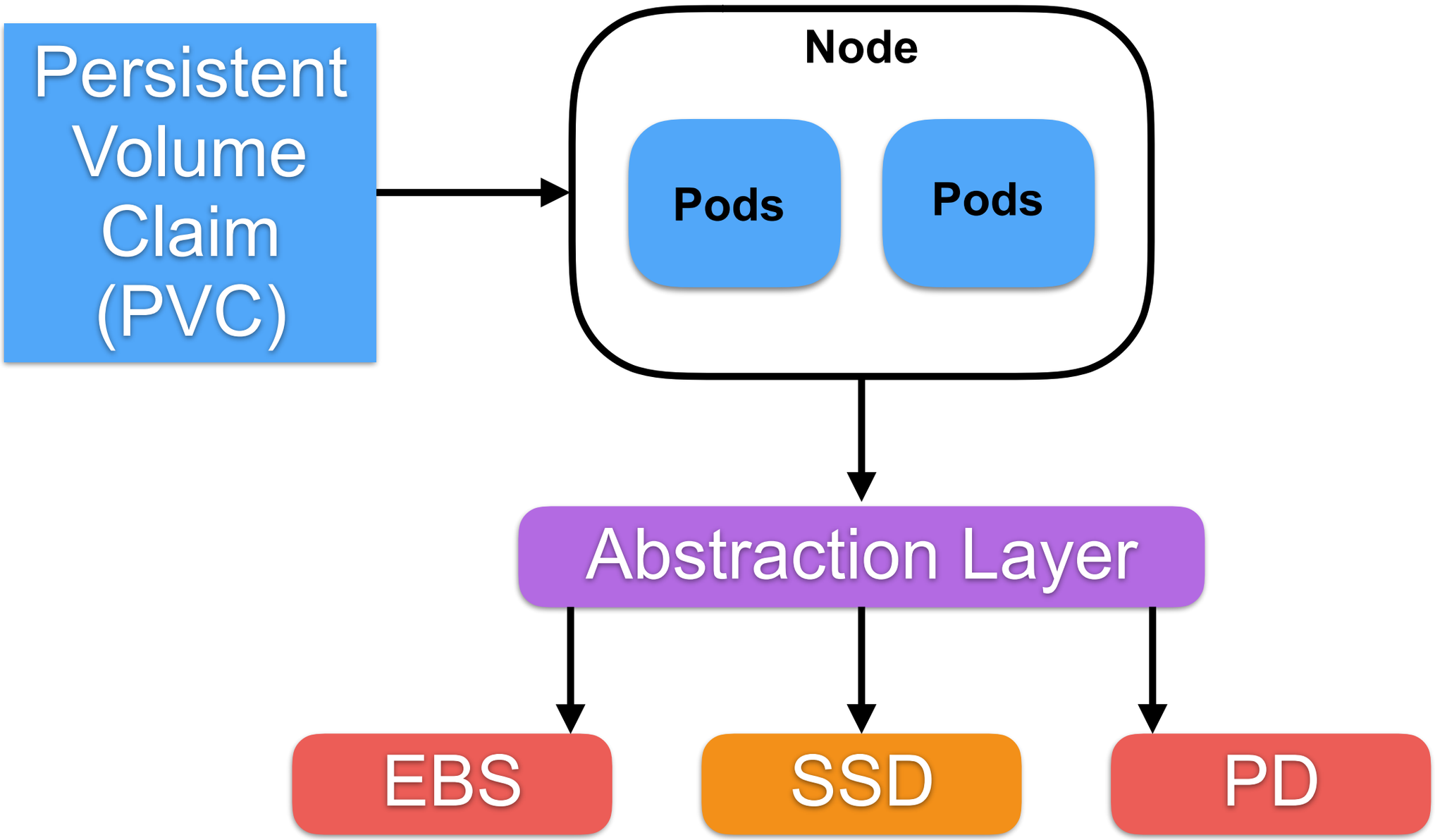
Kubernetes volumes



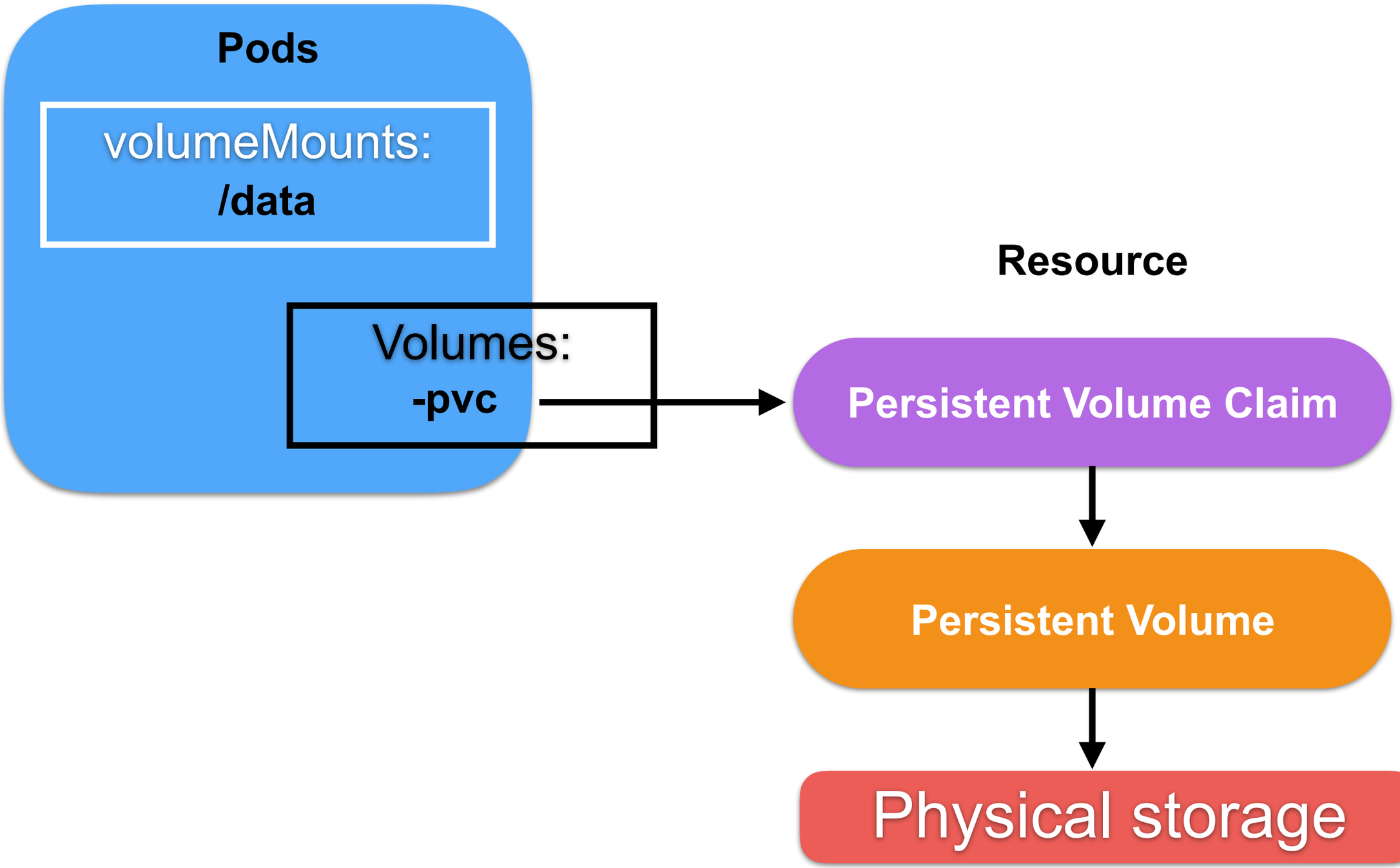
Kubernetes volumes



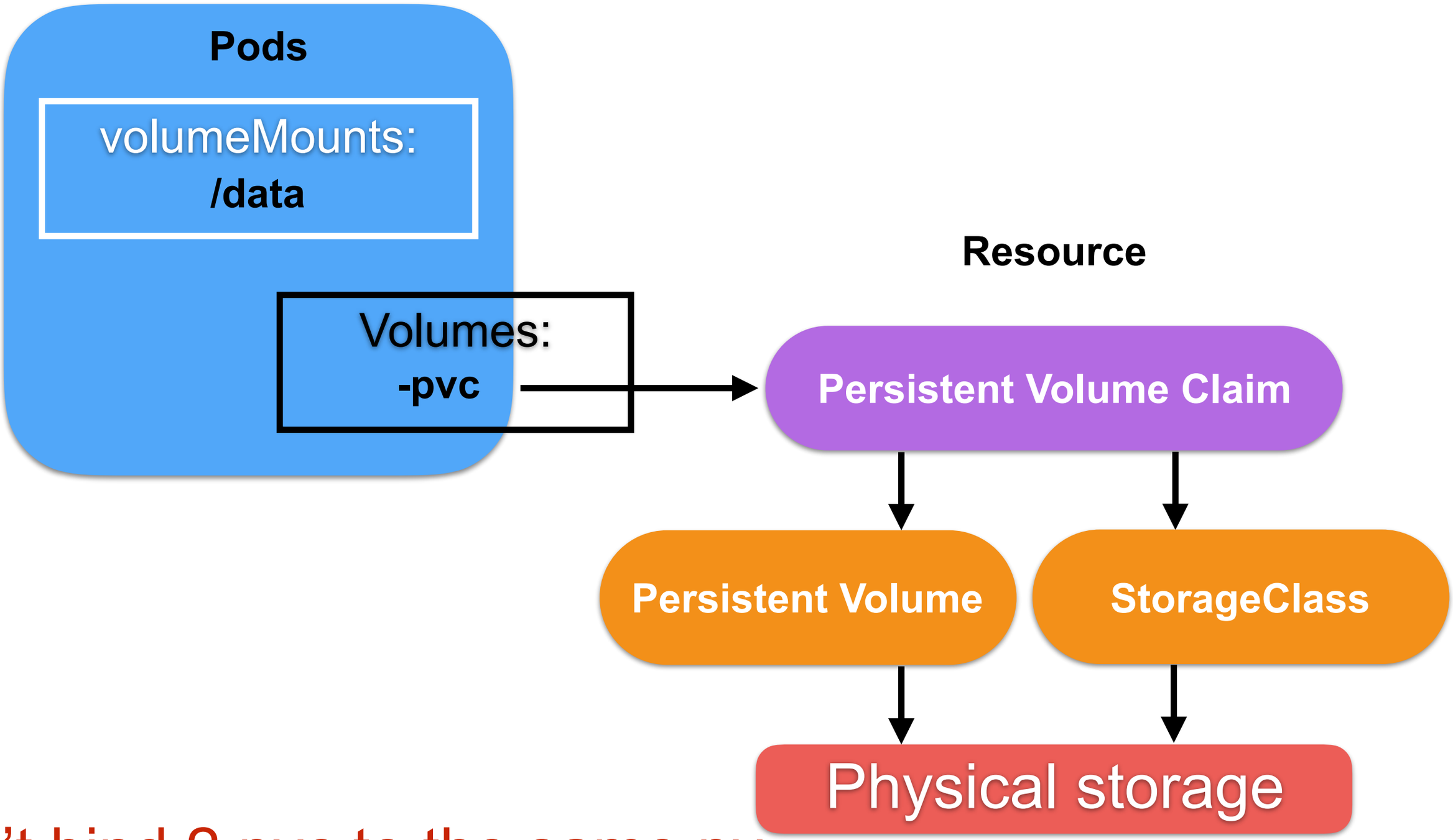
Kubernetes volumes



Persistent volume



Persistent volume



Can’t bind 2 pvc to the same pv.

EmptyDir

Shares volume/storage in same Pods

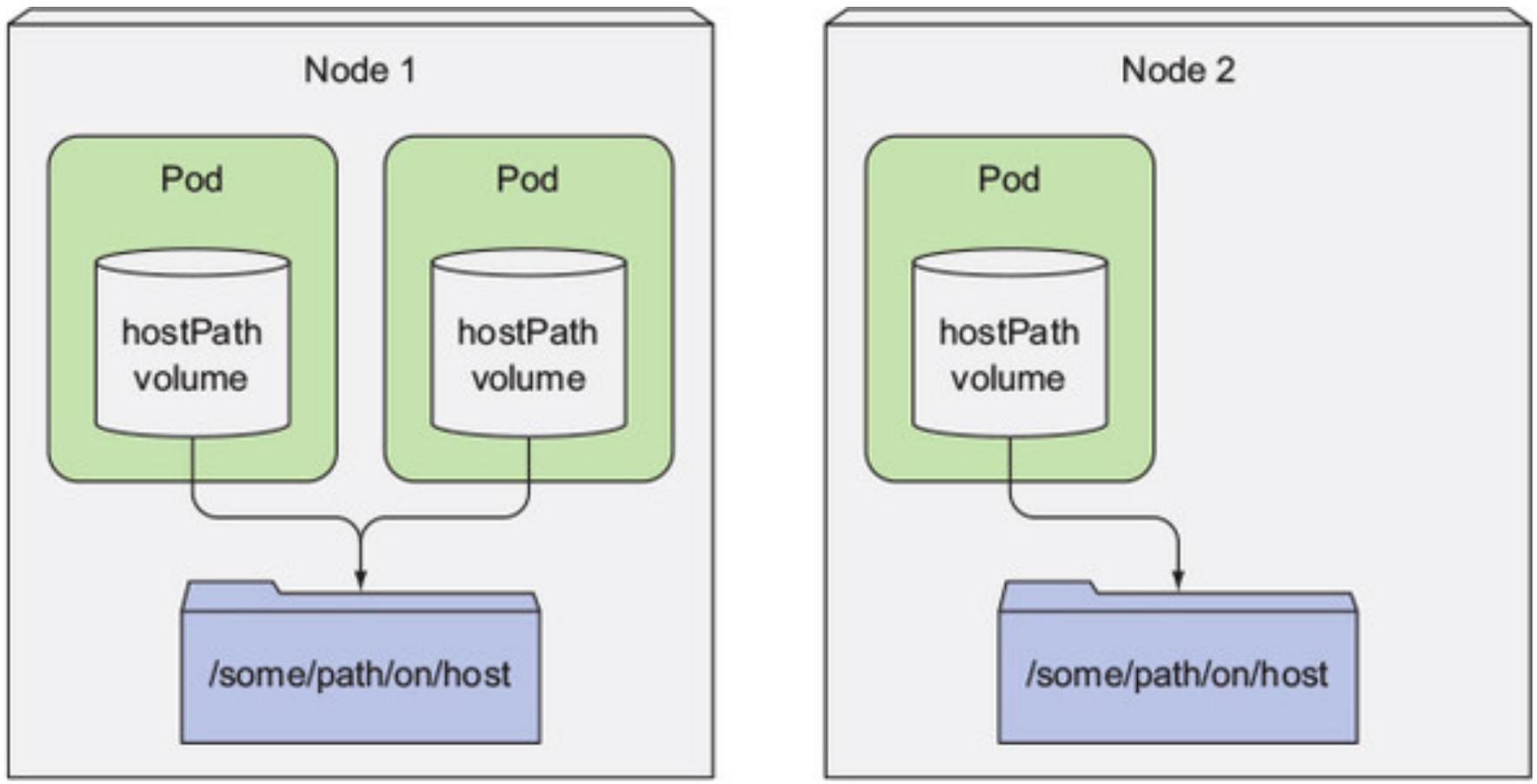
```
    - image: nginx:alpine
      name: web-server
      volumeMounts:
        - name: html
          mountPath: /usr/share/nginx/html
          readOnly: true
      ports:
        - containerPort: 80
          protocol: TCP

  volumes:
    - name: html
      emptyDir:
        medium: Memory
```

HostPath

Mount file/dir from host to Pods
Use when Pods need data from host
Some hostPath need permission for access
Resource scheduling can’t check readiness

HostPath



HostPath

Keep data of MongoDB on worker node

```
spec:
  volumes:
    - name: mongodb-data
      hostPath:
        path: /tmp/mongodb
  containers:
    - image: mongo
      name: mongodb
      volumeMounts:
        - name: mongodb-data
          mountPath: /data/db
```

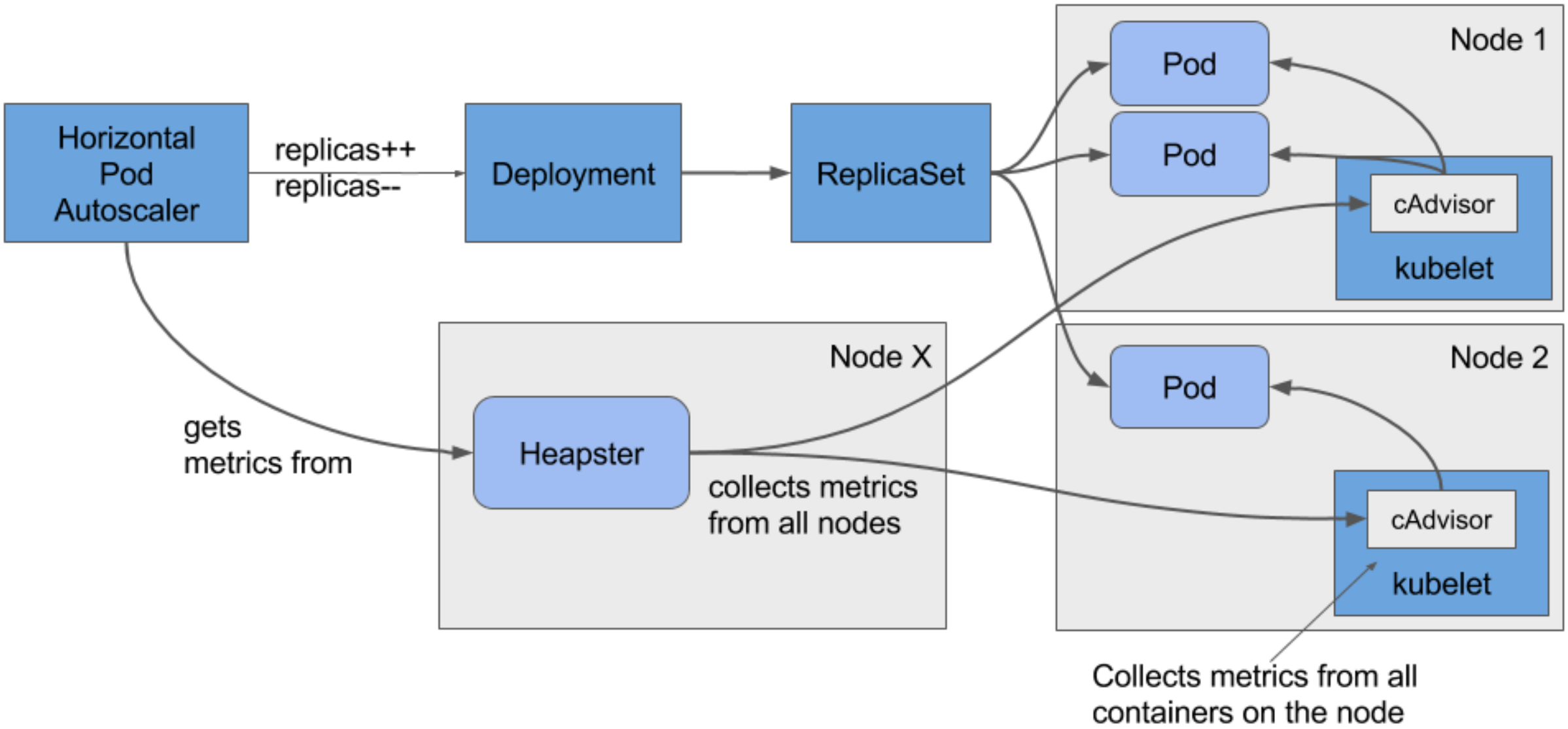
Persistent volume (PV)

TODO
TODO
TODO

Workshop Volume

Horizontal Pods Autoscaler (HPA)

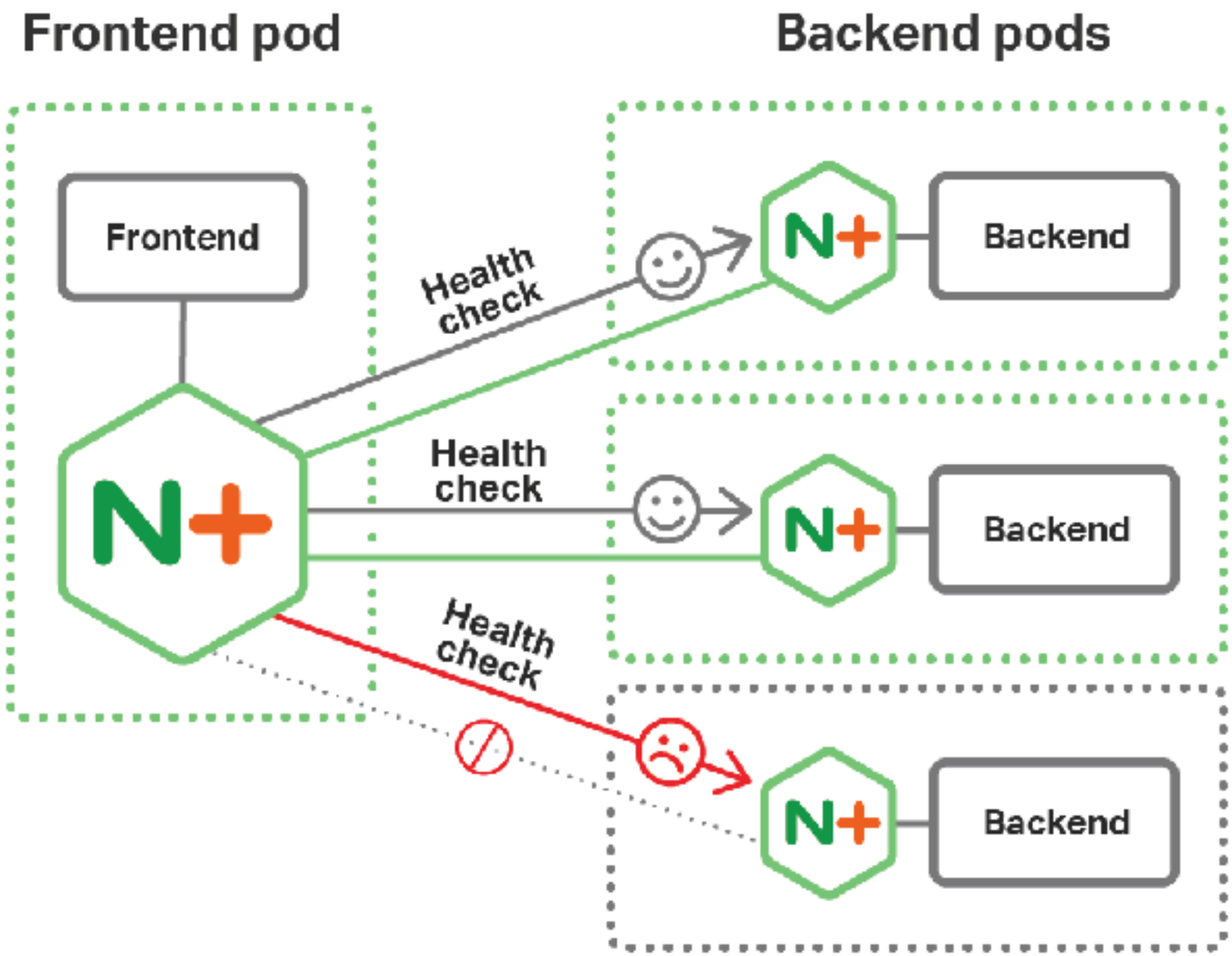
Components of autoscaling



Workshop HPA

Liveness and Readiness probe

Health check of system



https://www.nginx.com/blog/microservices-openshift-fabric-model-nginx-mra/

Liveness and Readiness

TODO
TODO
TODO
TODO
TODO

ConfigMap and Secret

StatefulSet