



Kubernetes

In Practice







Somkiat Puisungnoen

Update Info 1 View Activity Log 10+ ...

Timeline About Friends 3,138 Photos More

When did you work at Opendream? X

... 22 Pending Items

Post Photo/Video Live Video Life Event

What's on your mind?

Public Post

Intro

Software Craftsmanship

Software Practitioner at สยามชั่นนาคุกิจ พ.ศ. 2556

Agile Practitioner and Technical at SPRINT3r

Somkiat Puisungnoen 15 mins · Bangkok · ...

Java and Bigdata



somkiat.cc

Page Messages Notifications 3 Insights Publishing Tools Settings Help ▾

somkiat.cc
@somkiat.cc

Home Posts Videos Photos

Liked Following Share ... + Add a Button

Help people take action on this Page. ×



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

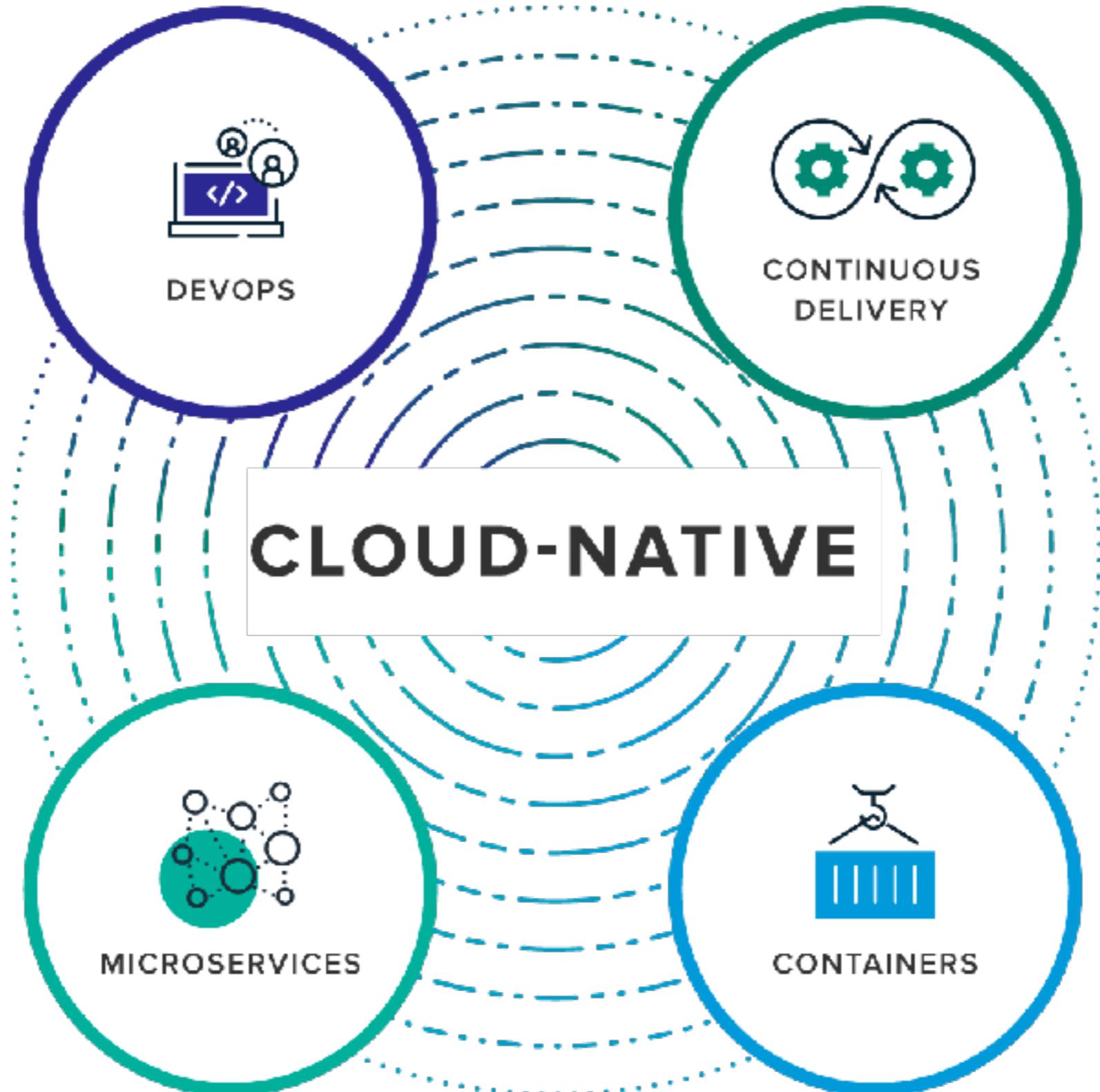


<https://github.com/up1/course-kubernetes-in-practice>



Cloud Native Application





<https://pivotal.io/cloud-native>



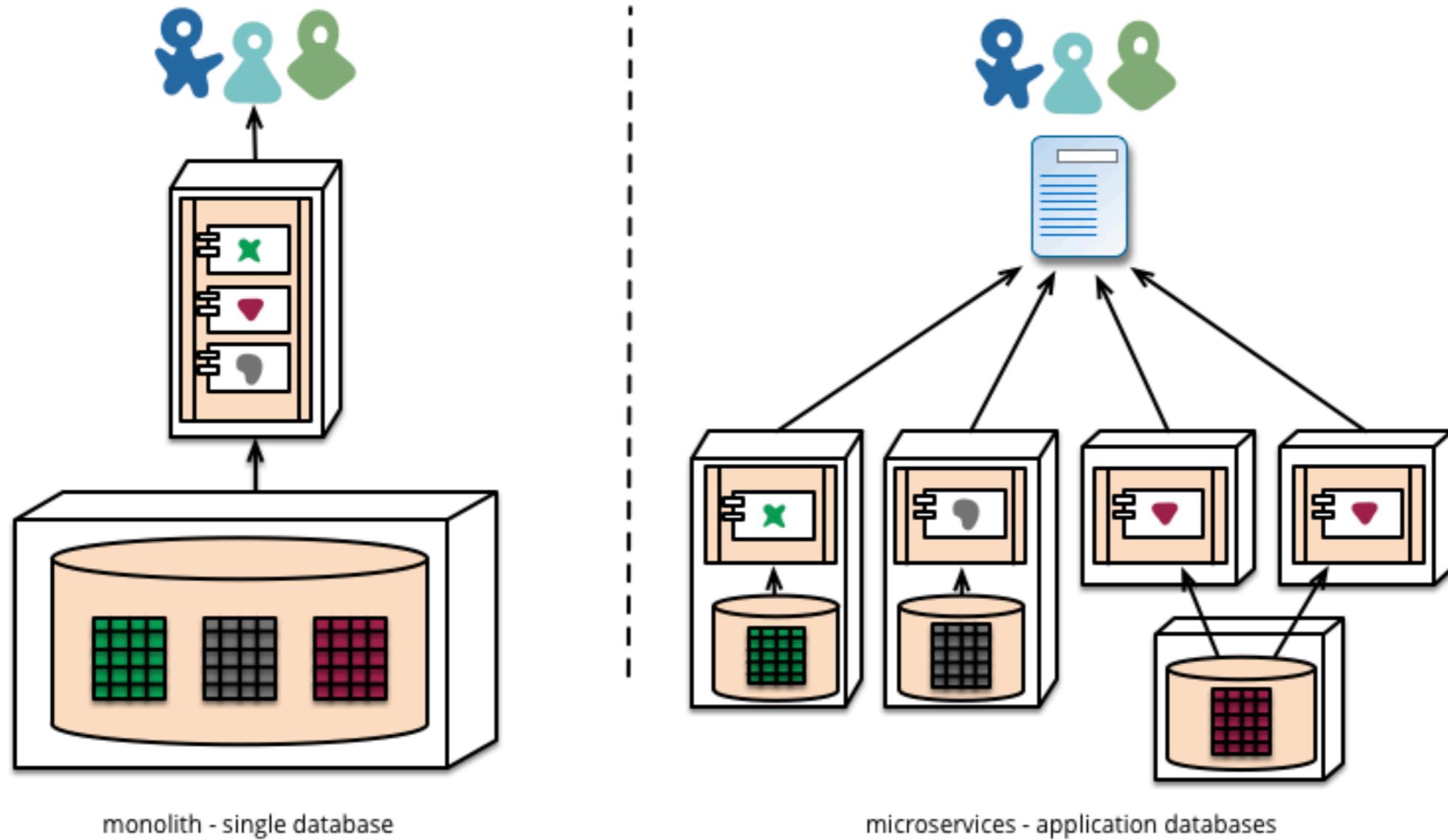
Evolution of Architecture



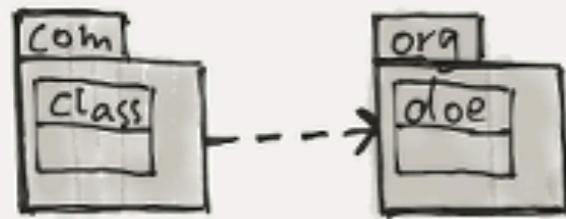
Kubernetes in practice

© 2017 - 2018 Siam Chamnankit Company Limited. All rights reserved.

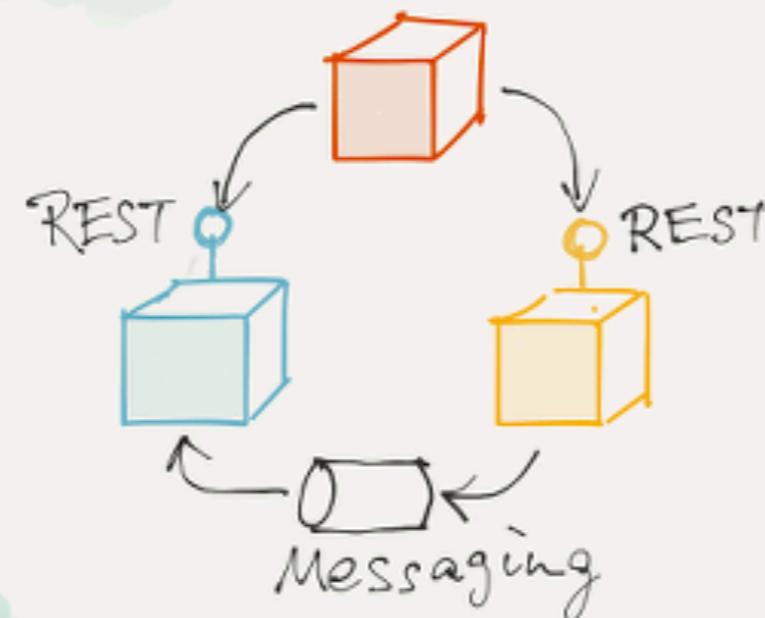
Microservices



Architecture



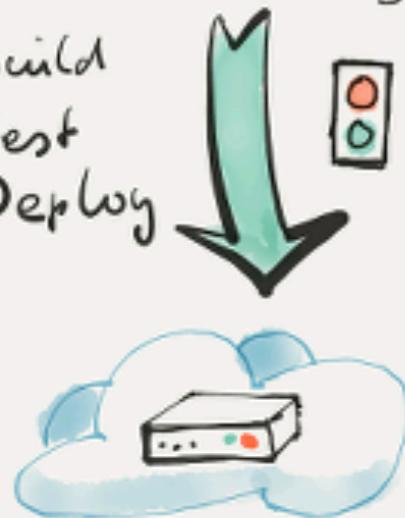
Microservices



Deployment

Continuous Delivery

`{ var i=1; }`
Build
Test
Deploy



Infrastructure

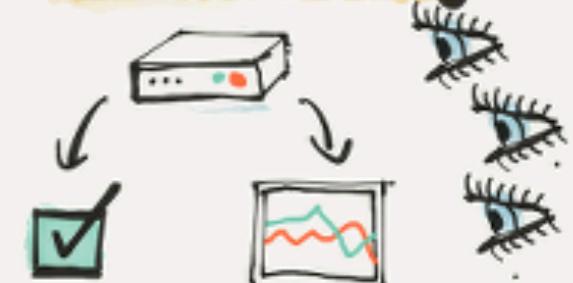


People & Teams



Communication
Collaboration

Monitoring

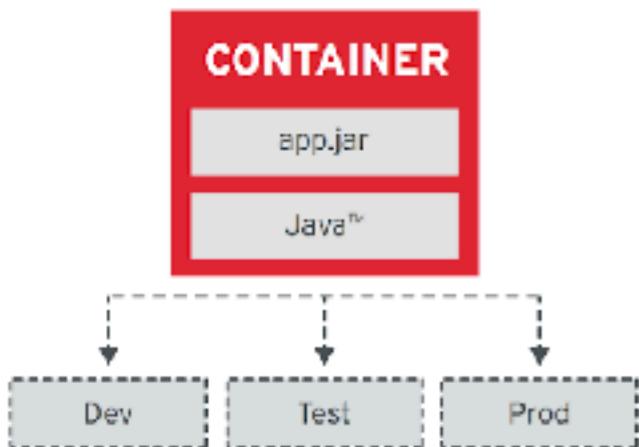


Features & Technology



Container design principles

Image Immutability Principle



High Observability Principle



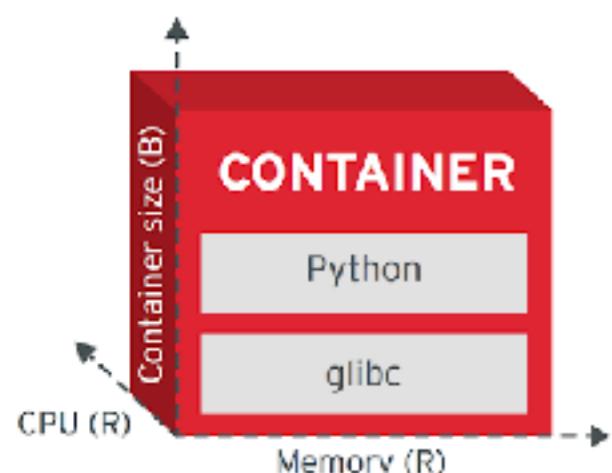
Process Disposability Principle



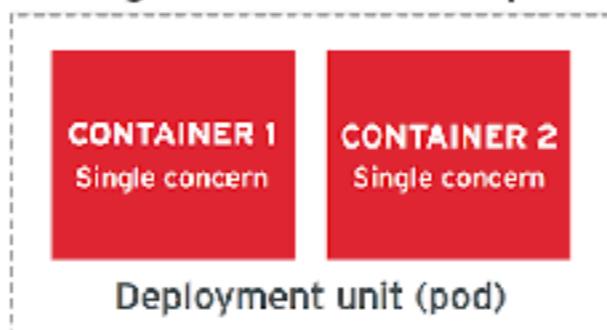
Lifecycle Conformance Principle



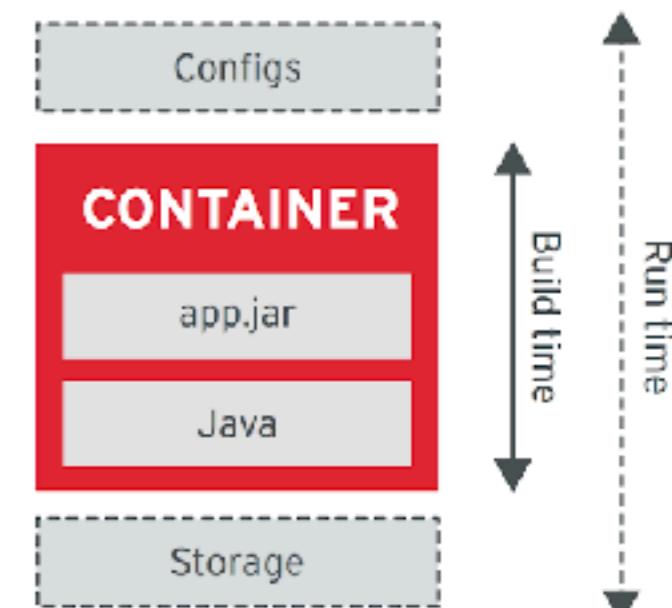
Runtime Confinement Principle



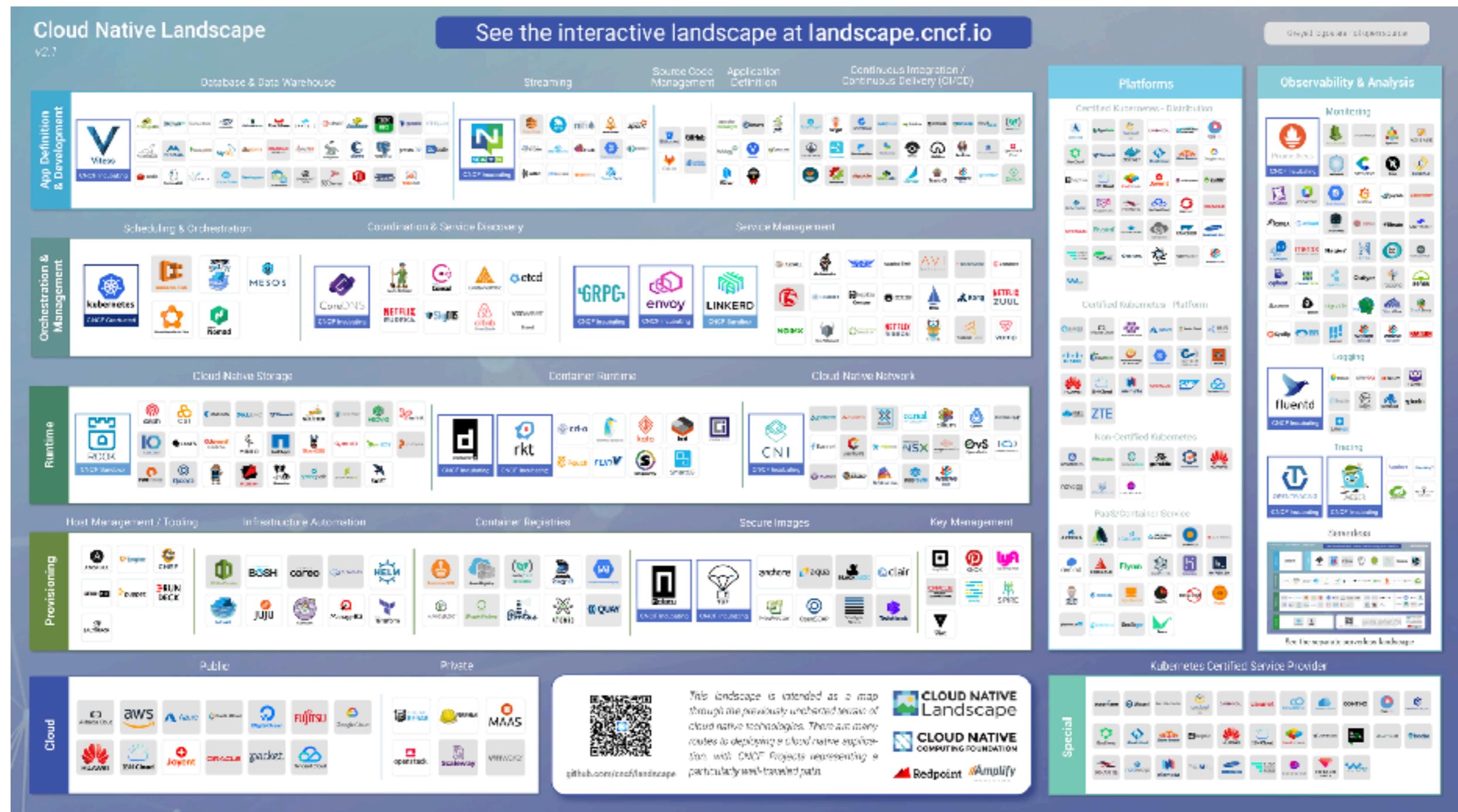
Single Concern Principle



Self-Containment Principle



Cloud native landscape



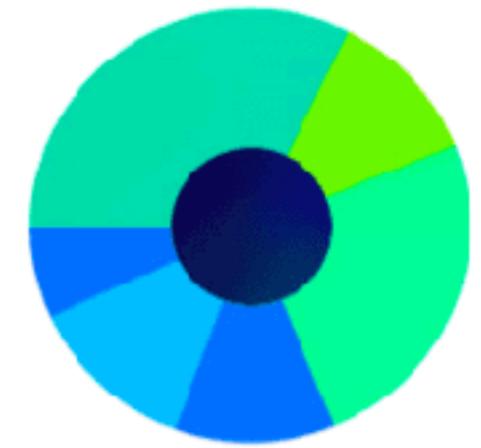
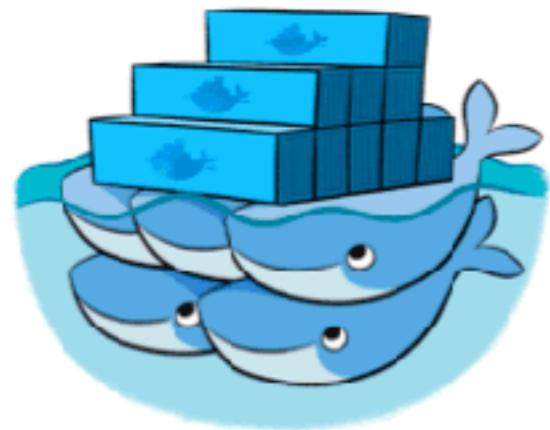
<https://github.com/cncf/landscape>







MESOS



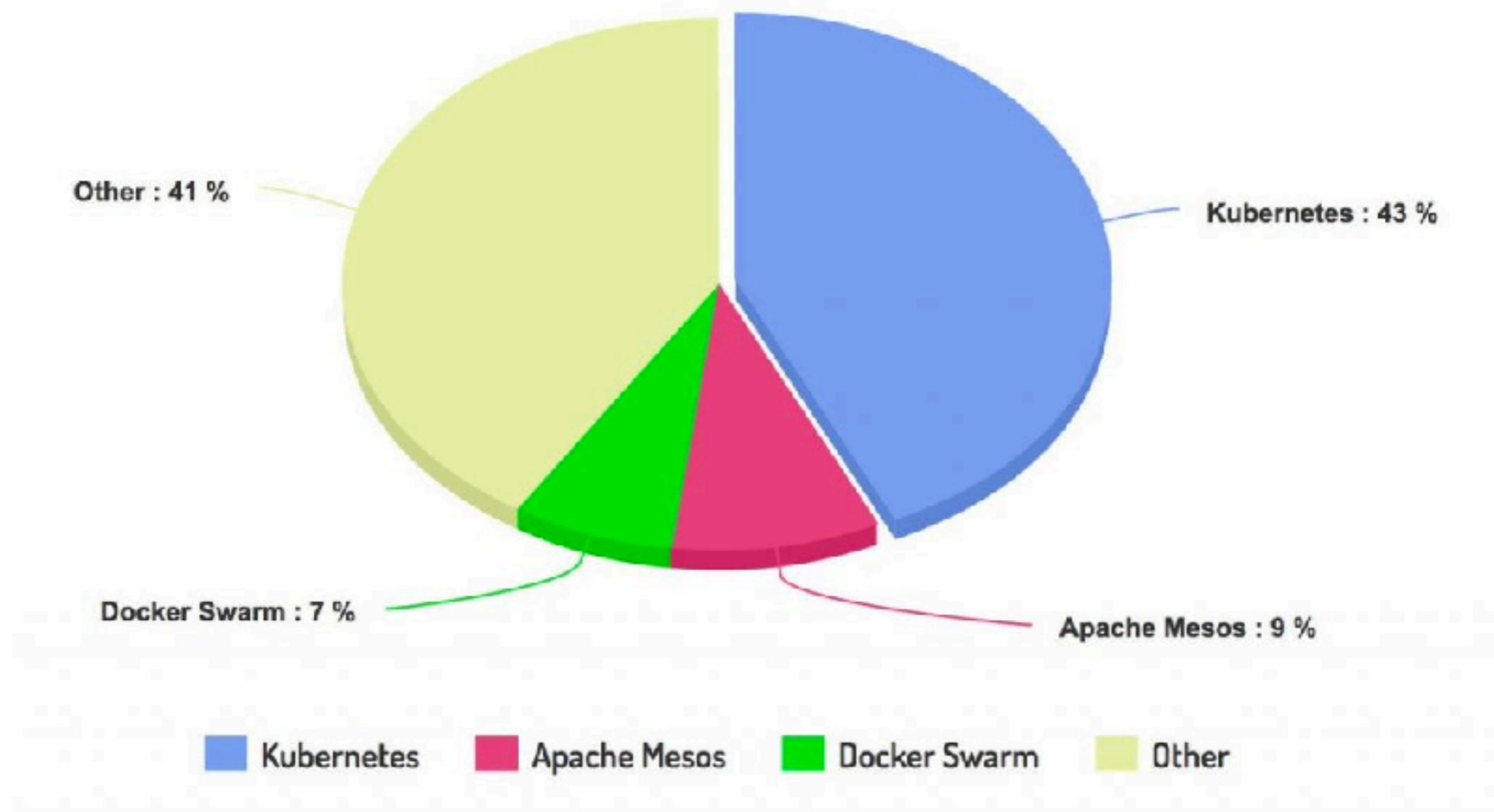
Marathon



kubernetes
by Google



Container Orchestrators in Sysdig's 2017 Docker Usage Report



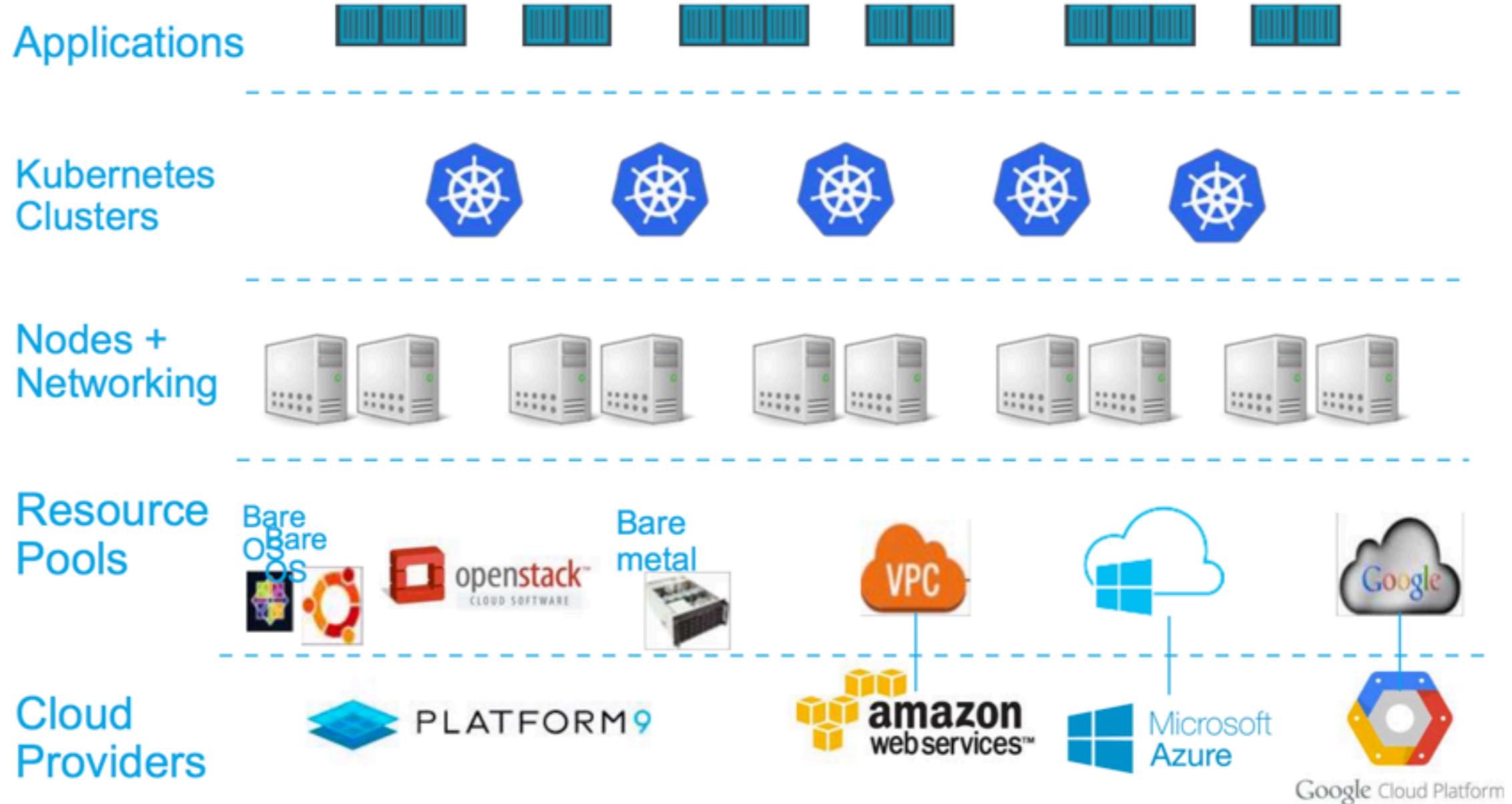
<https://sysdig.com/blog/sysdig-docker-usage-report-2017/>



Why Kubernetes ?



Write once, run anywhere



Write once, run anywhere

Eliminate infrastructure lock-in

Use containers

Provides management for containers



Modular app design

Monolithic app makes everything worse

- Larger teams slow thing down

- Spaghetti dependencies

Lack of ownership for sharing components

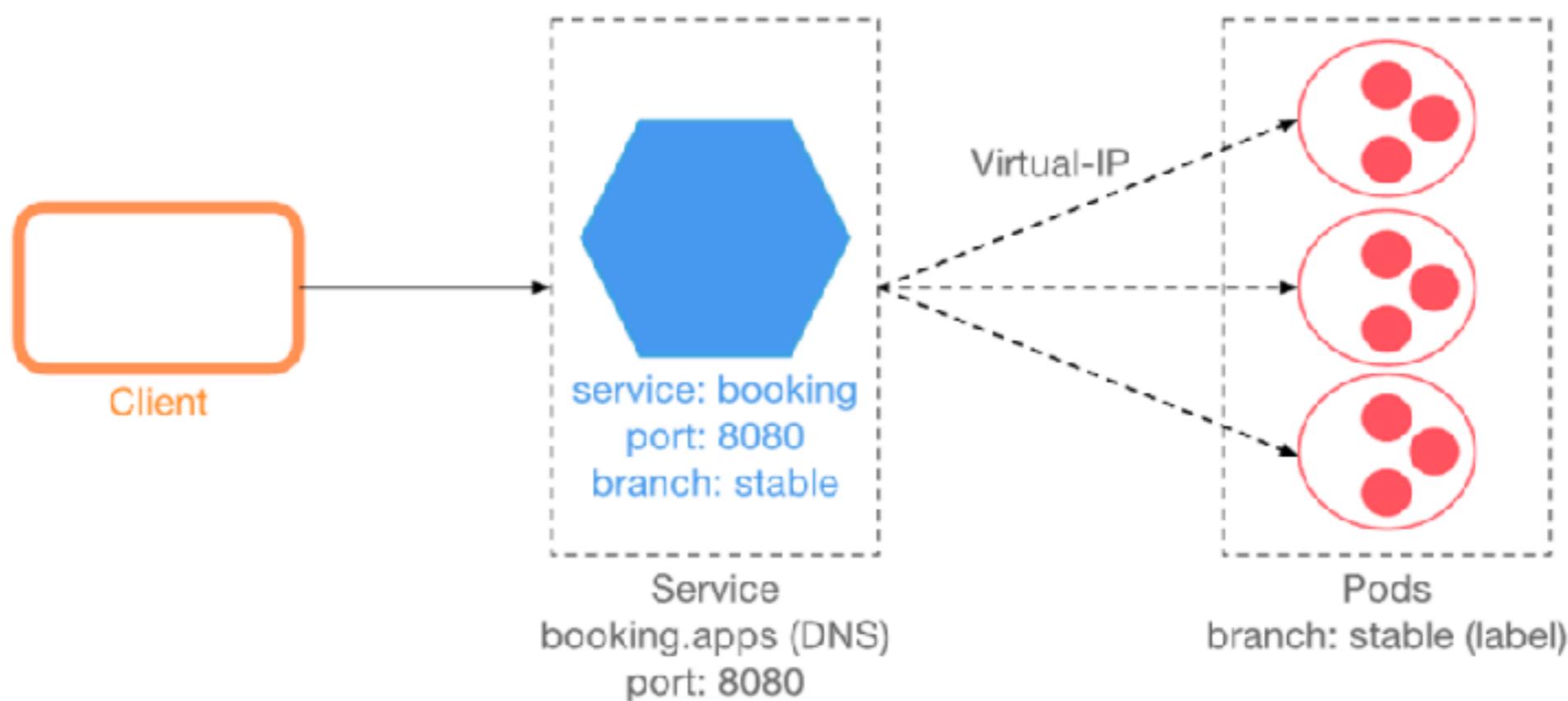
- Complexity to testing

- Slow building



Modular app design

Container/Image boundary similar to class
How to use/manage a collection of container ?



Fault-tolerant by design

Design for failure

Infrastructure provisioning/re-provisioning

Configuration networking and load balance

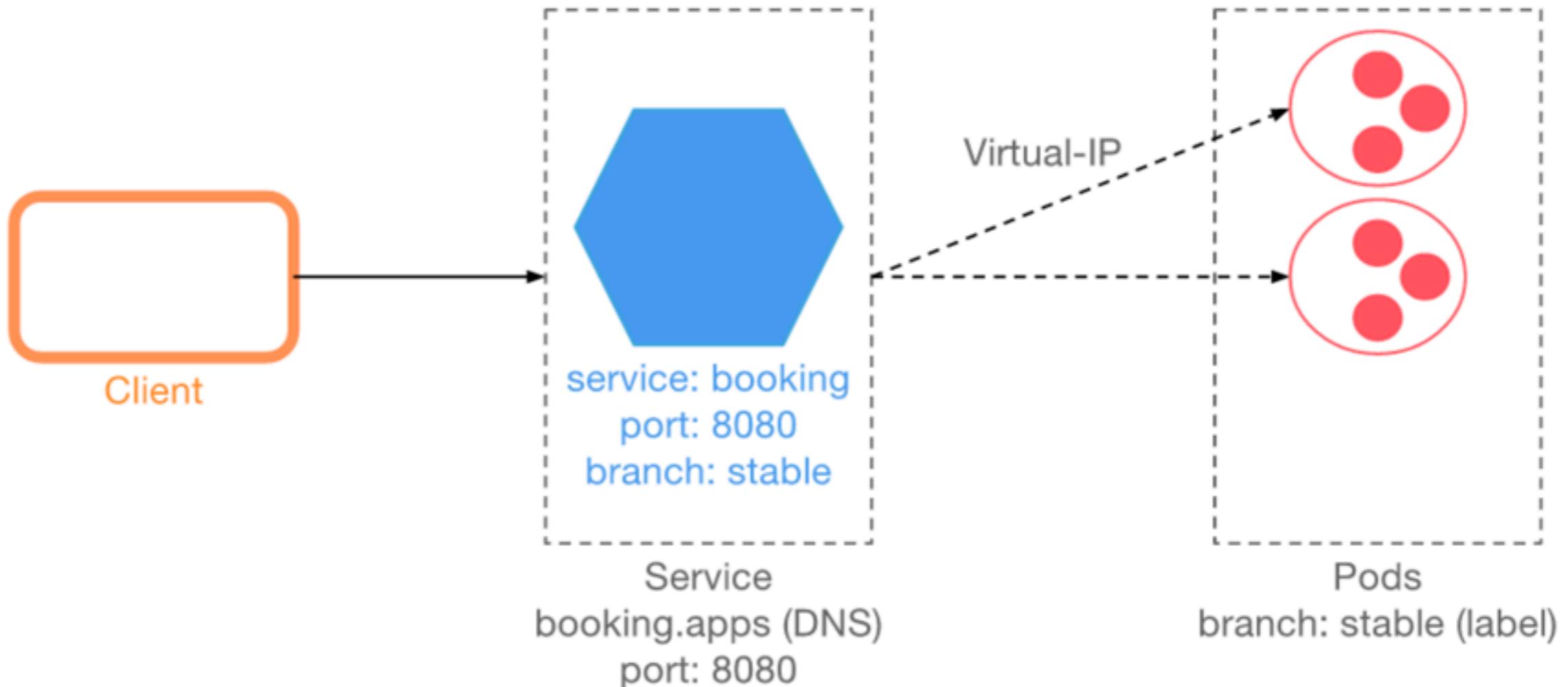
Redundancy (scale-out)

Lifecycle management (Software update)



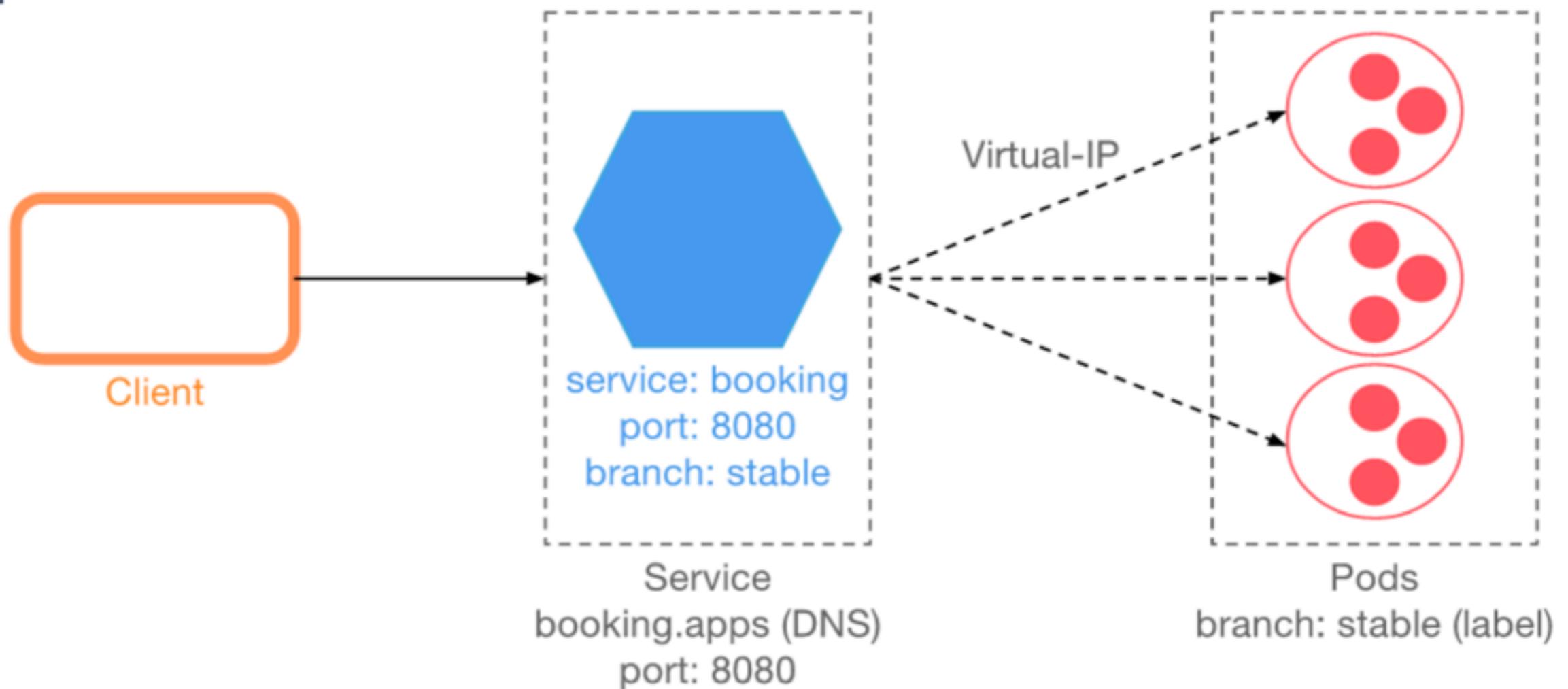
Fault-tolerant by design

replicas = 2



Fault-tolerant by design

replicas = 3



Deployment, not Infrastructure

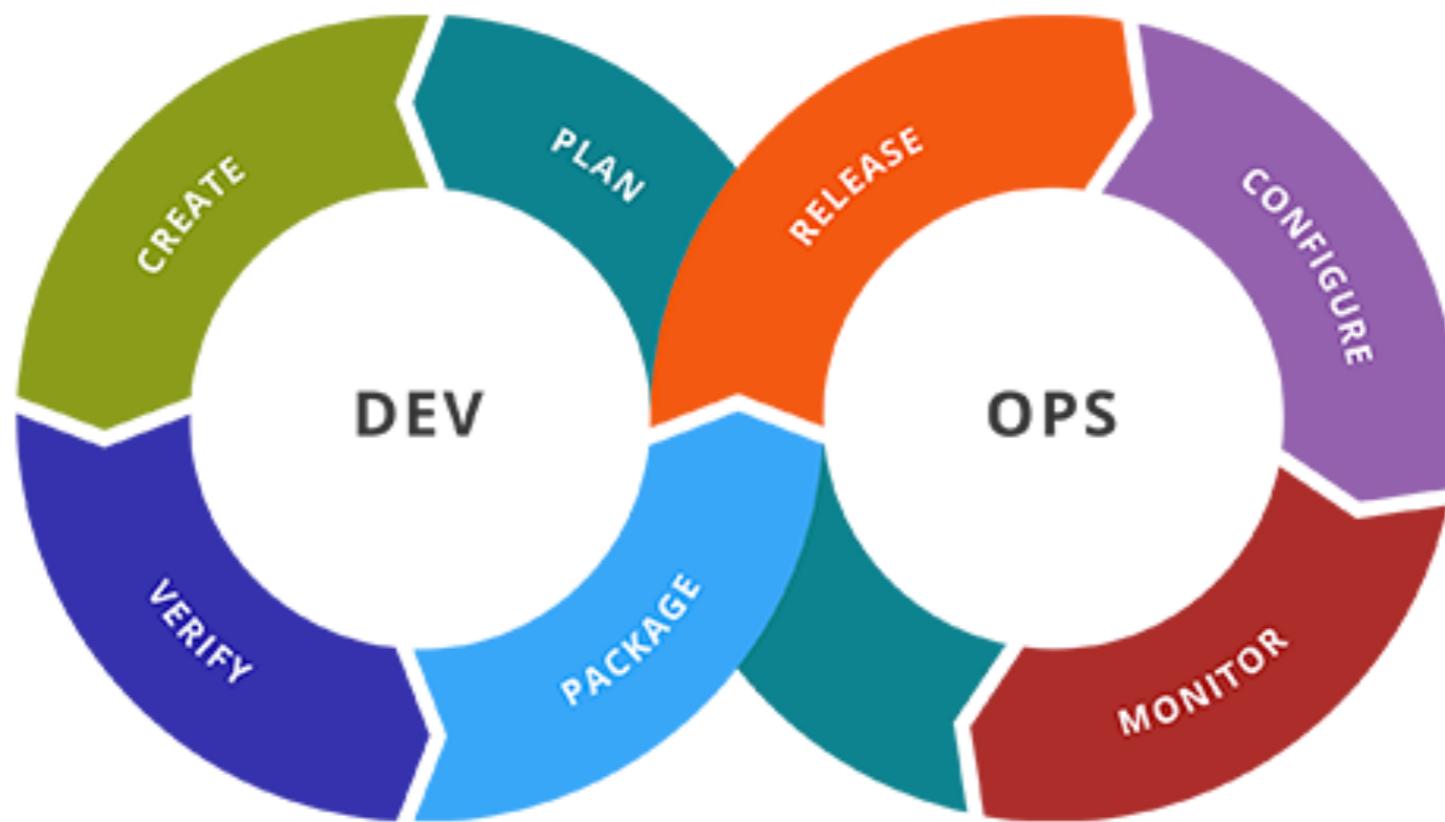
Software deployment is hard
Infrastructure provisioning/re-provisioning
Configuration networking and load balance
Redundancy (scale-out)
Lifecycle management (Software update)



Deployment, not Infrastructure

Kubernetes support for deployment
Controllers are in focus
Scale-out service
Rolling update for new version
Rollback to a previous version
Pause and resume a deployment
Horizontal auto-scaling
Canary deployment





Let's start !!



Installation



Software requirement

minkube

Kubernetes command-line tool



Starting minikube

```
$minikube start  
$minikube status
```



Version of Kubernetes

```
$kubectl get nodes -o yaml
```

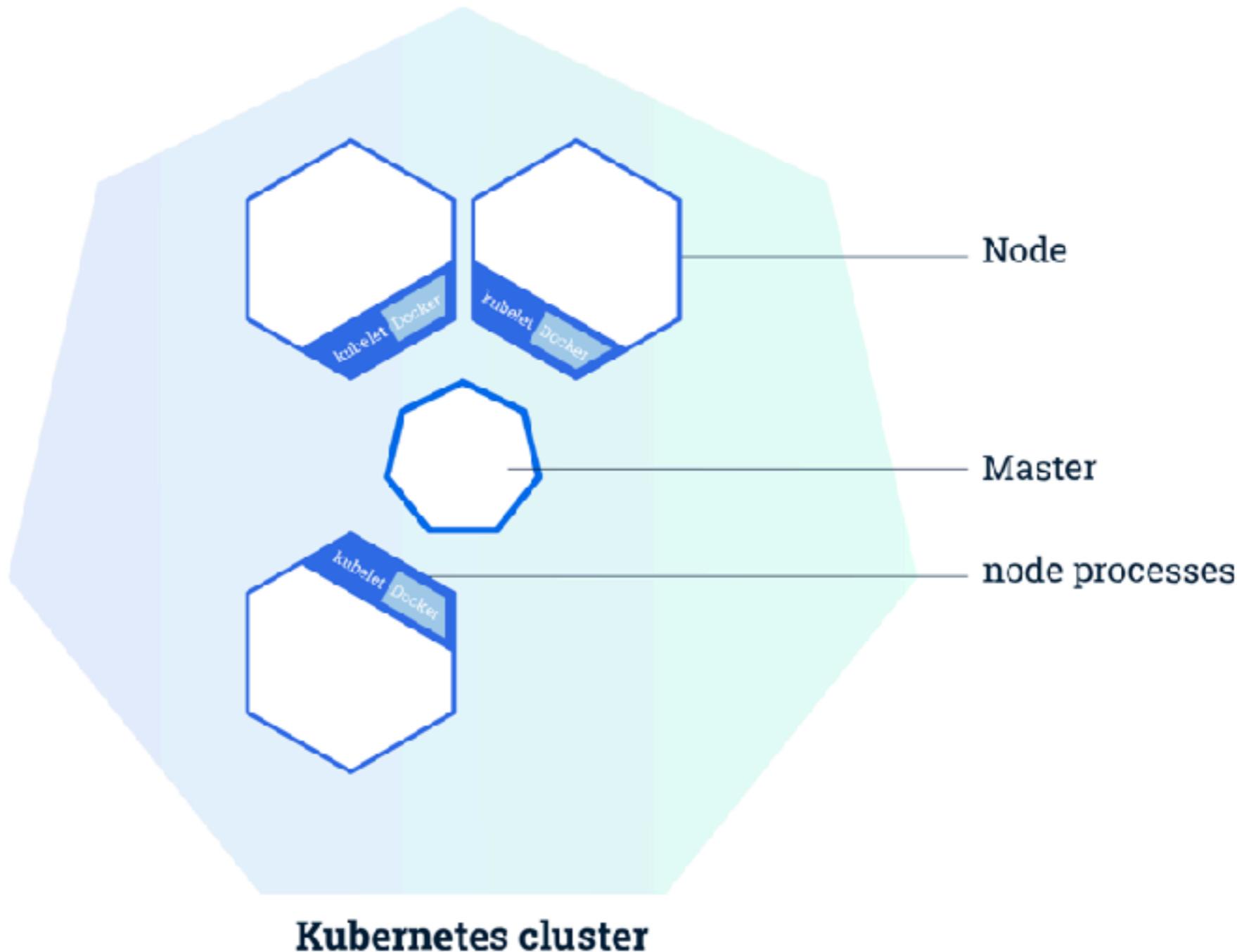
```
nodeInfo:  
  architecture: amd64  
  bootID: 7c769465-27e5-4dd8-a89f-319ba1b8ef57  
  containerRuntimeVersion: docker://17.9.0  
  kernelVersion: 4.9.64  
  kubeProxyVersion: v1.9.4  
  kubeletVersion: v1.9.4  
  machineID: edffd4ca8bf24253a736ea86d2448185  
  operatingSystem: linux  
  osImage: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



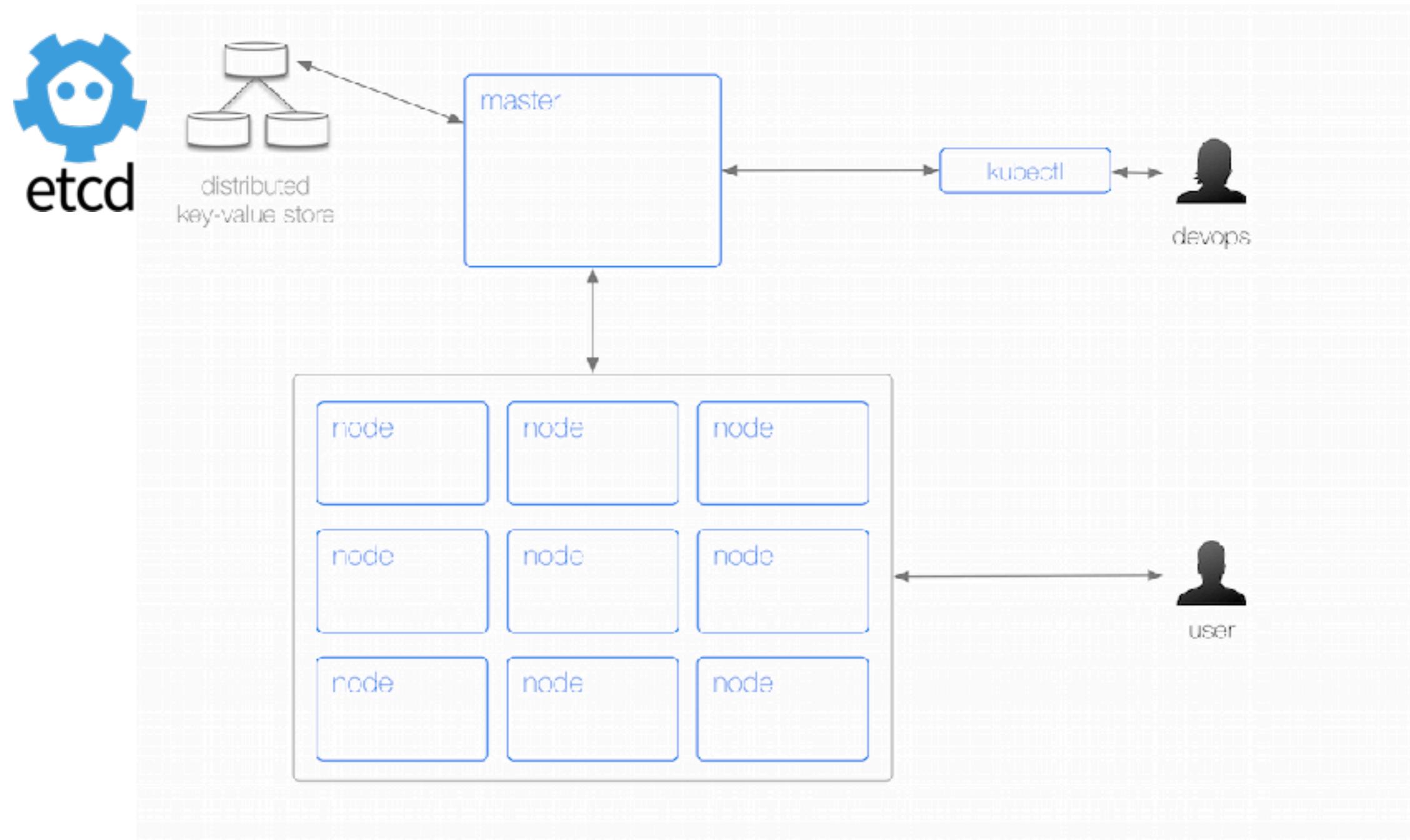
Basic of Kubernetes



Kubernetes cluster



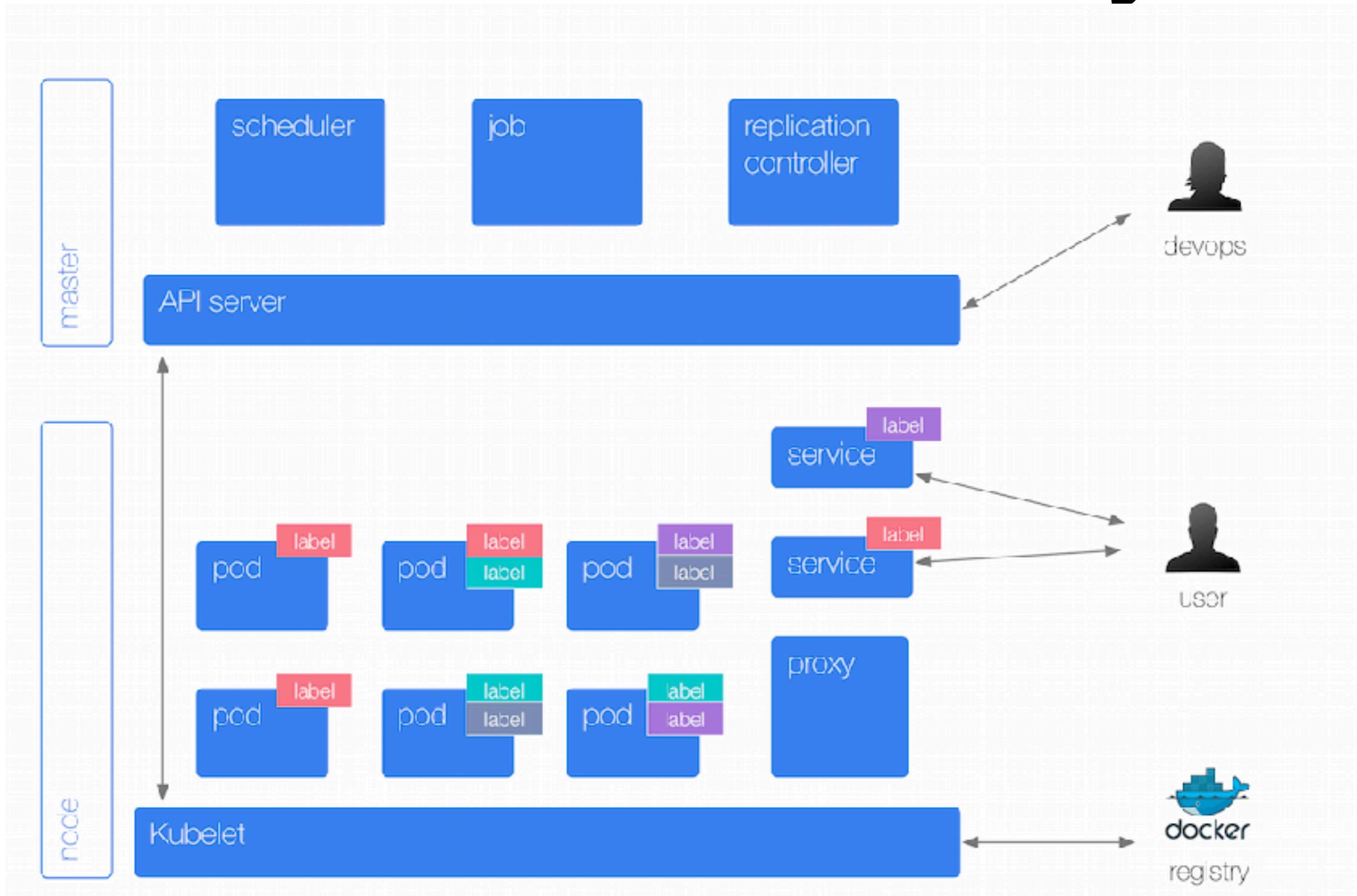
Kubernetes physical layout



<http://k8s.info/cs.html>



Kubernetes abstraction layout



<http://k8s.info/cs.html>



Hello with Kubernetes

file /01-hello/instruction.txt

file /01-hello/instruction2.txt



Hello Kubernetes

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



Try to use somkiat/hello

The screenshot shows the Docker Hub interface for the repository `somkiat/hello`. At the top, there's a dark header bar with a search bar, navigation links for Dashboard, Explore, Organizations, Create, and a user profile for `somkiat`. Below the header, the repository name `somkiat/hello` is displayed with a star icon, indicating it's a public repository. A note says "Last pushed: 2 months ago". There are tabs for Repo Info, Tags, Collaborators, Webhooks, and Settings, with the Tags tab currently selected. A table lists three tags: `v3`, `v2`, and `latest`, each with a compressed size of 5 MB and a last updated date of 2 months ago. Each tag row has a delete icon.

Tag Name	Compressed Size	Last Updated	
v3	5 MB	2 months ago	
v2	5 MB	2 months ago	
latest	5 MB	2 months ago	

<https://hub.docker.com/r/somkiat/hello>



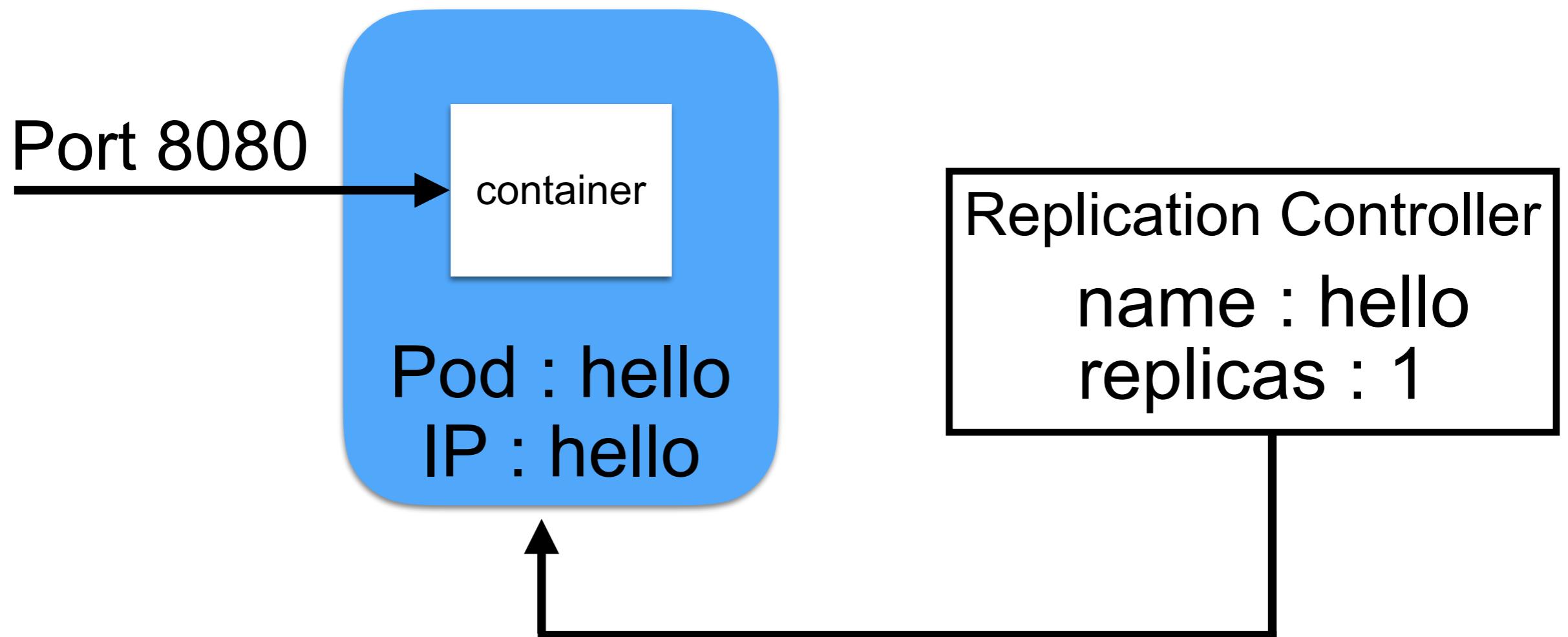
Create a container

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

<https://kubernetes.io/docs/reference/kubectl/conventions/#generators>



Create a container

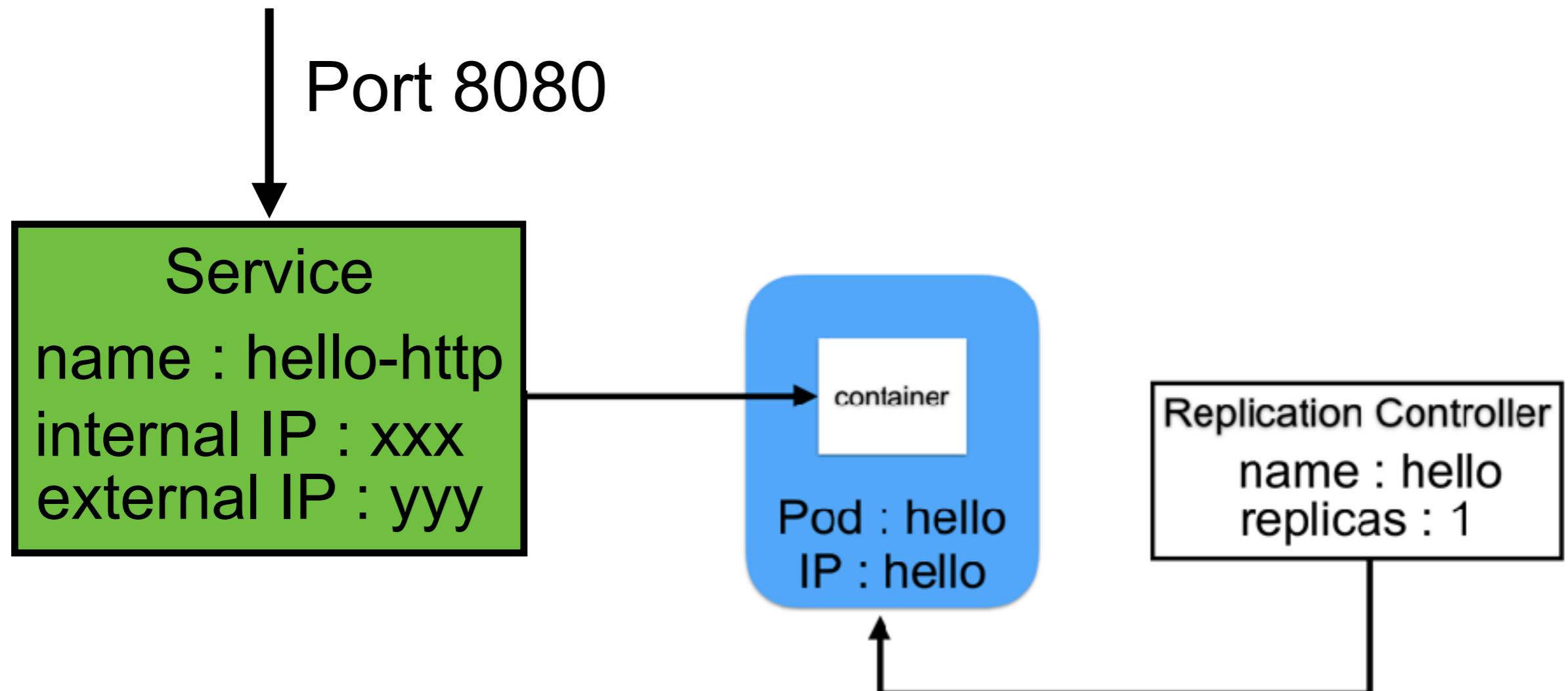


Expose RC with service

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



Expose RC with service



Access service with minikube

```
$minikube service hello-http
```

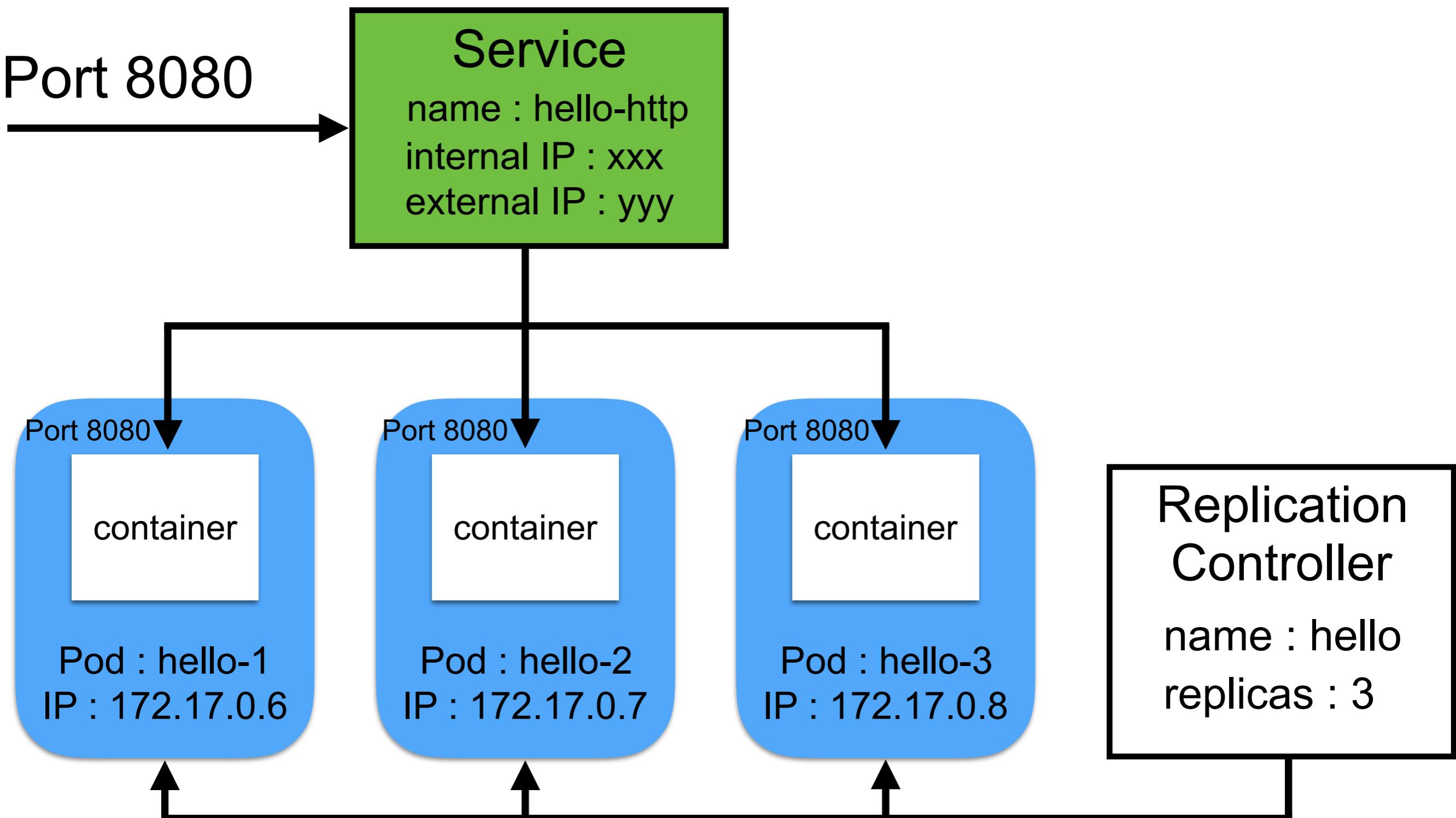


Scaling the application

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



Scaling the application



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-pqnpz	1/1	Running	0	36m	172.17.0.6	minikube
hello-wwsnh	1/1	Running	0	29m	172.17.0.7	minikube



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

The screenshot shows the Kubernetes dashboard interface. At the top, there is a navigation bar with a Kubernetes logo, a search bar, and a '+ CREATE' button. Below the navigation bar, a blue header bar displays the text 'Overview'. On the left side, there is a sidebar under the heading 'Cluster' containing links for Namespaces, Nodes, Persistent Volumes, Roles, and Storage Classes. A dropdown menu for 'Namespaces' is open, showing 'default' as the selected option. Below the sidebar, there are two main sections: 'Workloads' and 'Workloads Statuses'. The 'Workloads Statuses' section contains two green circular charts, both labeled '100.00%' and representing 'Pods' and 'Replication Controllers'. In the 'Workloads' section, there is a table titled 'Pods' listing three entries: 'hello-lw2p7', 'hello-wwsnh', and 'hello-pqnpz', all running on the 'minikube' node. The table includes columns for Name, Node, Status, Restarts, and Age.

Name	Node	Status	Restarts	Age
hello-lw2p7	minikube	Running	0	53 minutes
hello-wwsnh	minikube	Running	0	53 minutes
hello-pqnpz	minikube	Running	0	59 minutes



Try to use jboss/wildfly

The screenshot shows the Docker Hub interface for the jboss/wildfly repository. At the top, there's a dark header bar with a search bar containing 'Search' and icons for Docker Hub. To the right are links for 'Explore', 'Help', 'Sign up' (in a blue button), and 'Sign in'. Below the header, the repository details are shown: 'PUBLIC | AUTOMATED BUILD' followed by the repository name 'jboss/wildfly' with a star icon indicating it's favorited. A note says 'Last pushed: a month ago'. Below this, there are tabs for 'Repo Info', 'Tags', 'Dockerfile', and 'Build Details', with 'Build Details' being the active tab. Under 'Short Description', it says 'WildFly application server image'. Under 'Full Description', it says 'WildFly Docker image'.

<https://hub.docker.com/r/jboss/wildfly/>



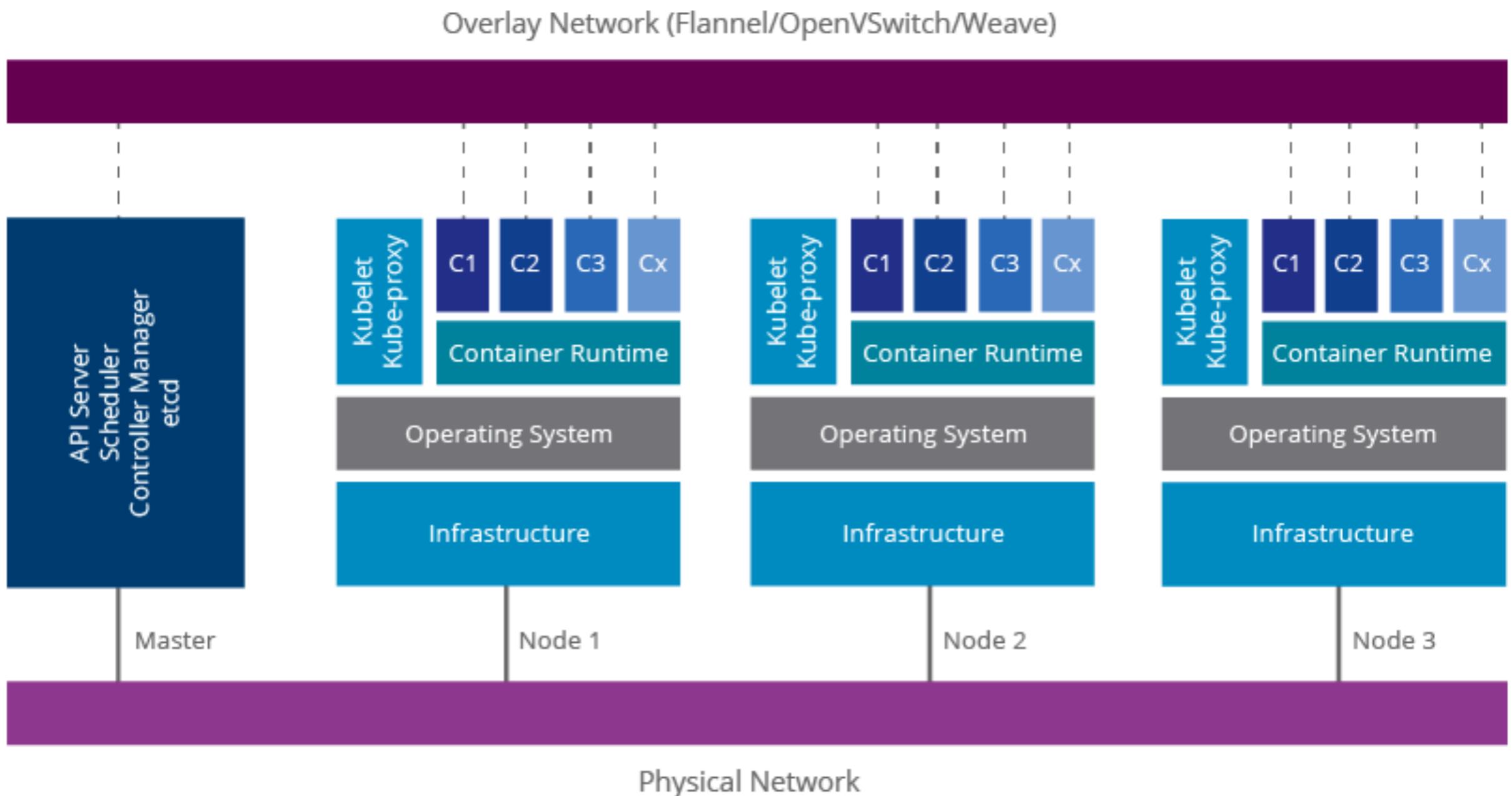
Welcome to Kubernetes



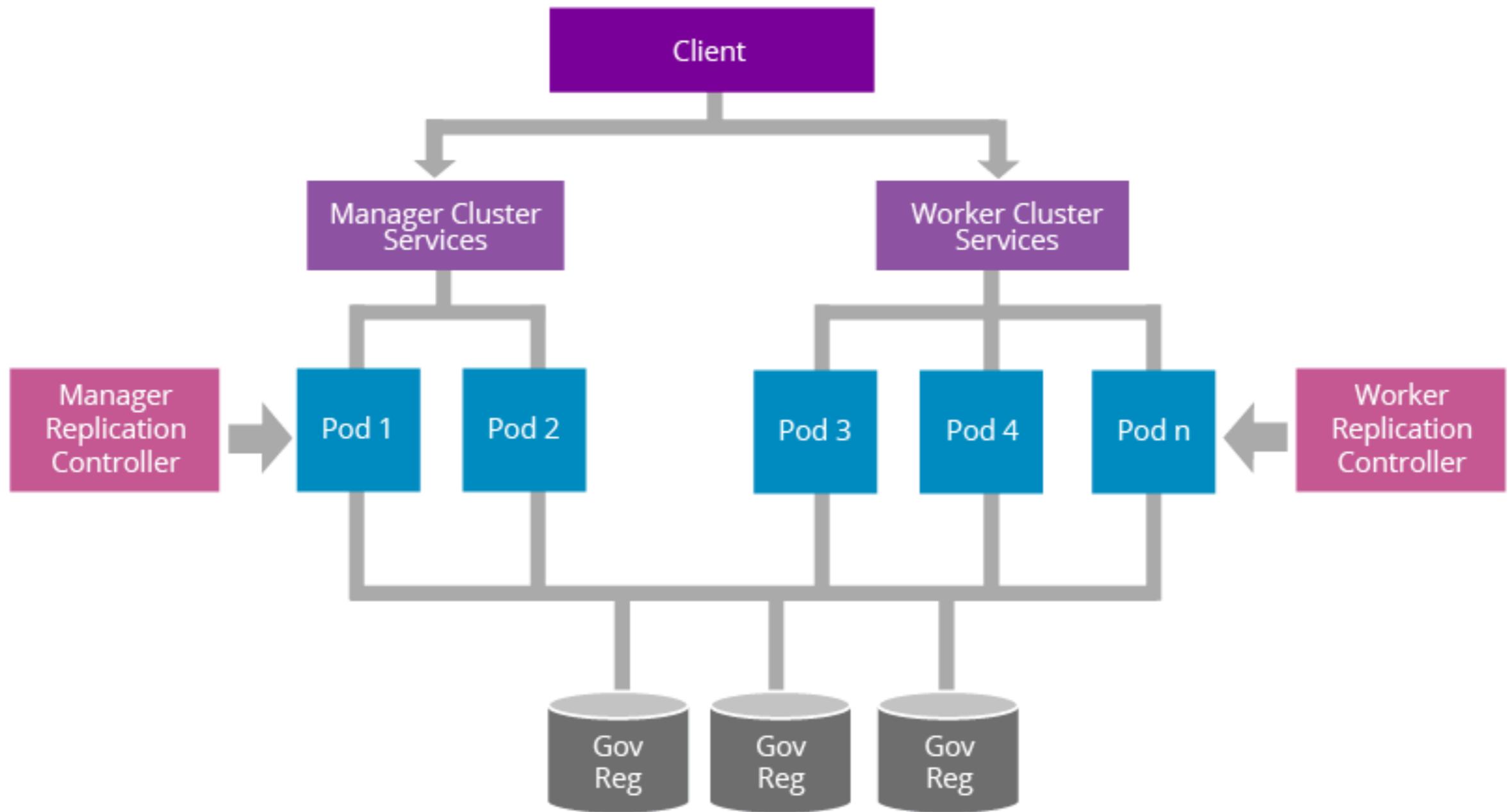
Kubernetes Architecture



Kubernetes architecture



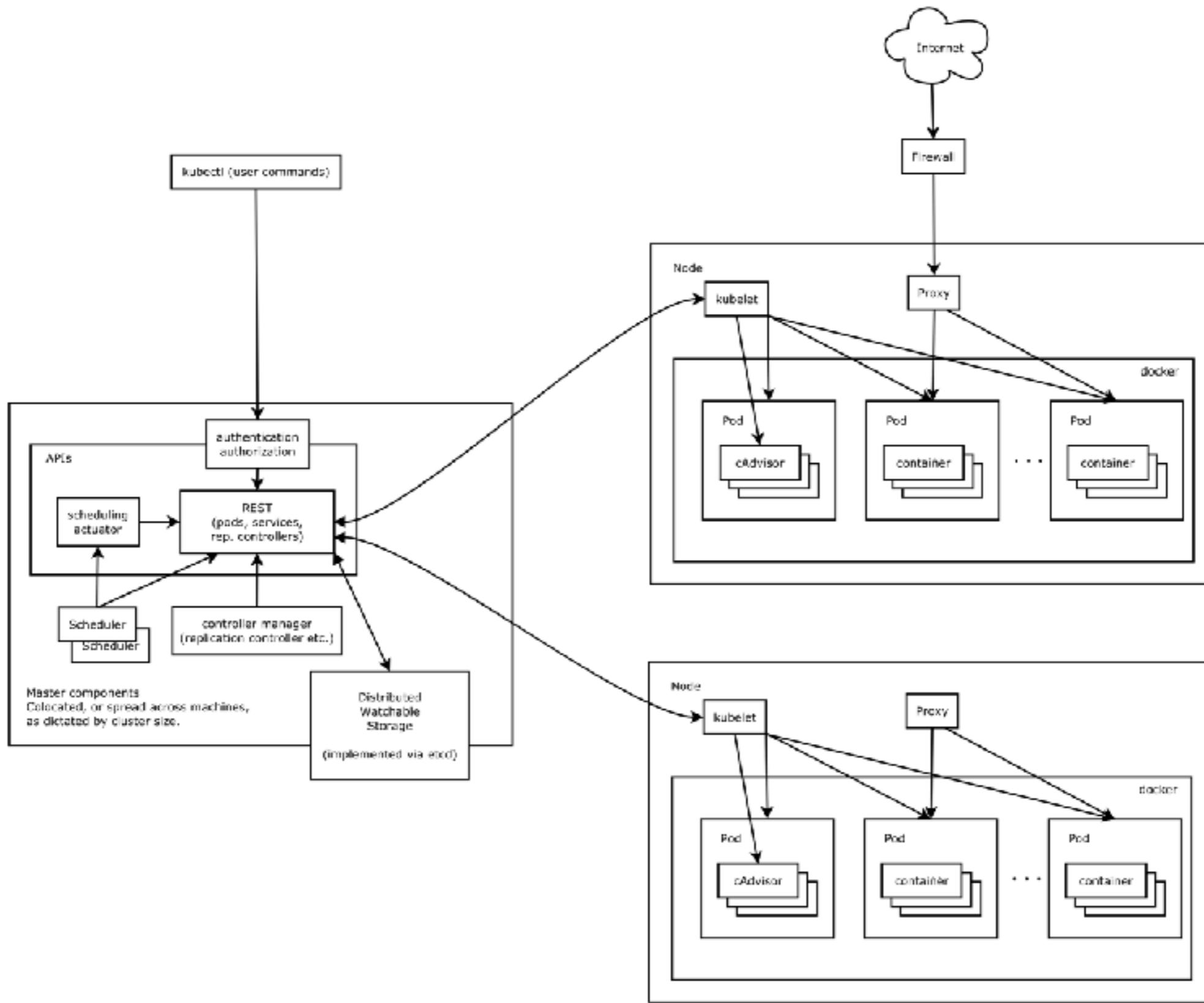
Kubernetes architecture



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



Kubernetes architecture



Key features

<https://kubernetes.io/>



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 binpacking



Automatic binpacking

Limit of CPU/Memory can define on Pods
Schedule will select a node that have enough resources



Automatic binpacking

When reach the Memory limit ?

1. Kill current Pod
2. If Pod have restart flag, try to create in other node



Automatic binpacking

When reach the CPU limit ?

1. Schedule not kill Pod
2. Schedule waiting it back to normal state



Automatic binpacking

Try to check resources of node

\$kubectl describe node

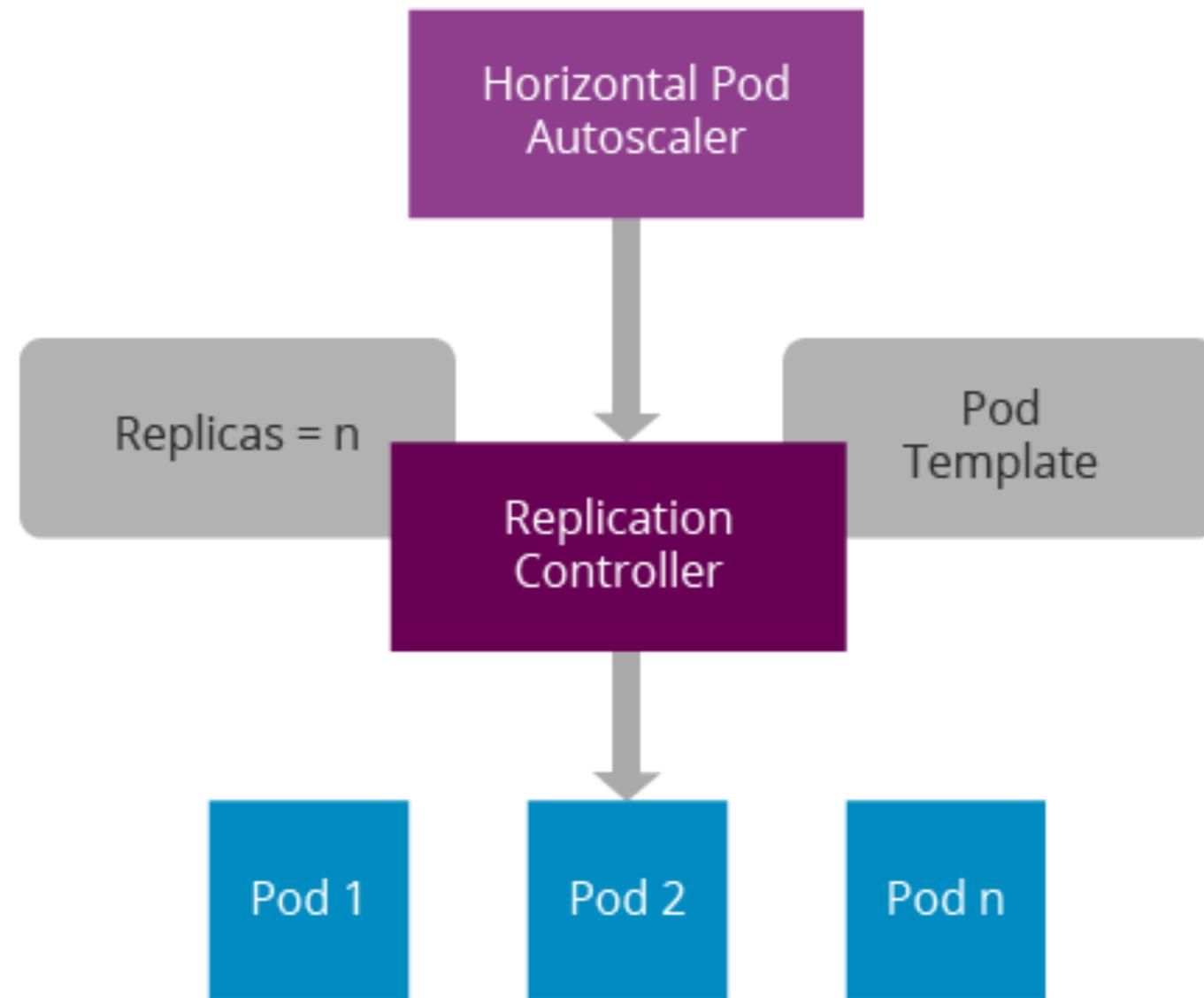
CPU Requests	CPU Limits	Memory Requests	Memory Limits
5m (0%)	0 (0%)	50Mi (2%)	0 (0%)
260m (13%)	0 (0%)	110Mi (5%)	170Mi (8%)
0 (0%)	0 (0%)	0 (0%)	0 (0%)
0 (0%)	0 (0%)	0 (0%)	0 (0%)



Horizontal Pod Autoscaler



Horizontal Pod Autoscaler

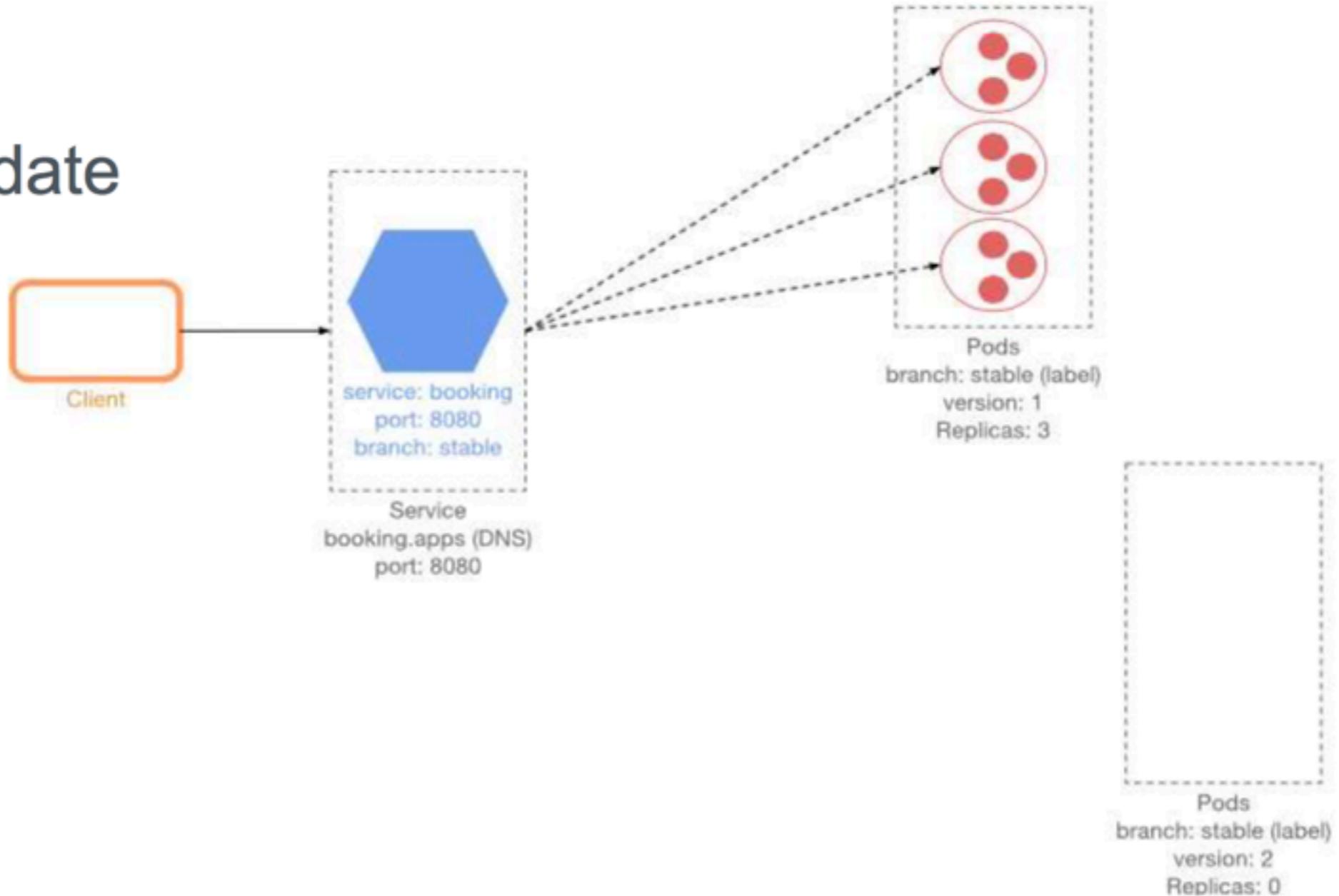


Automated rollouts & rollbacks



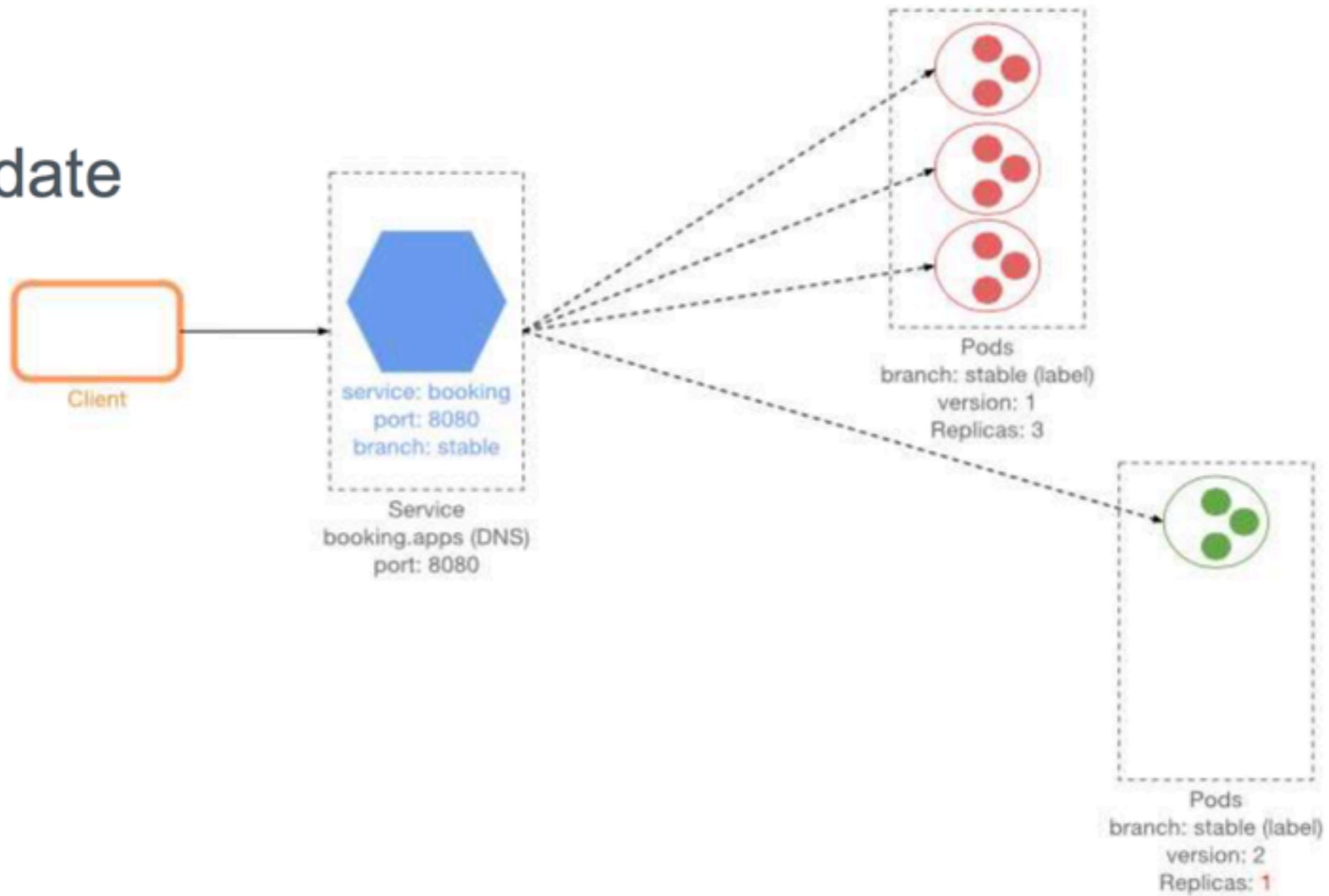
Rolling update (1)

Rolling Update



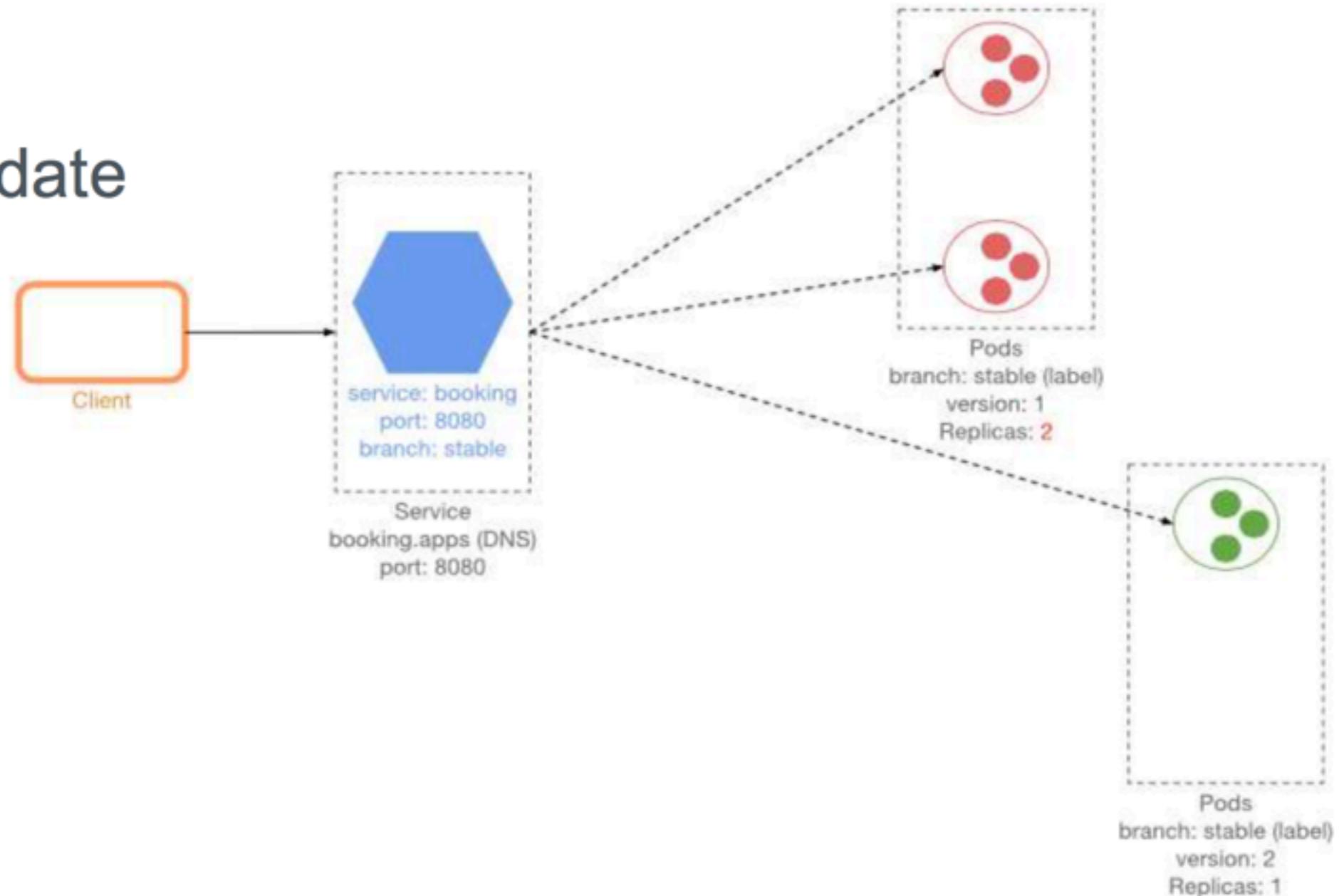
Rolling update (2)

Rolling Update



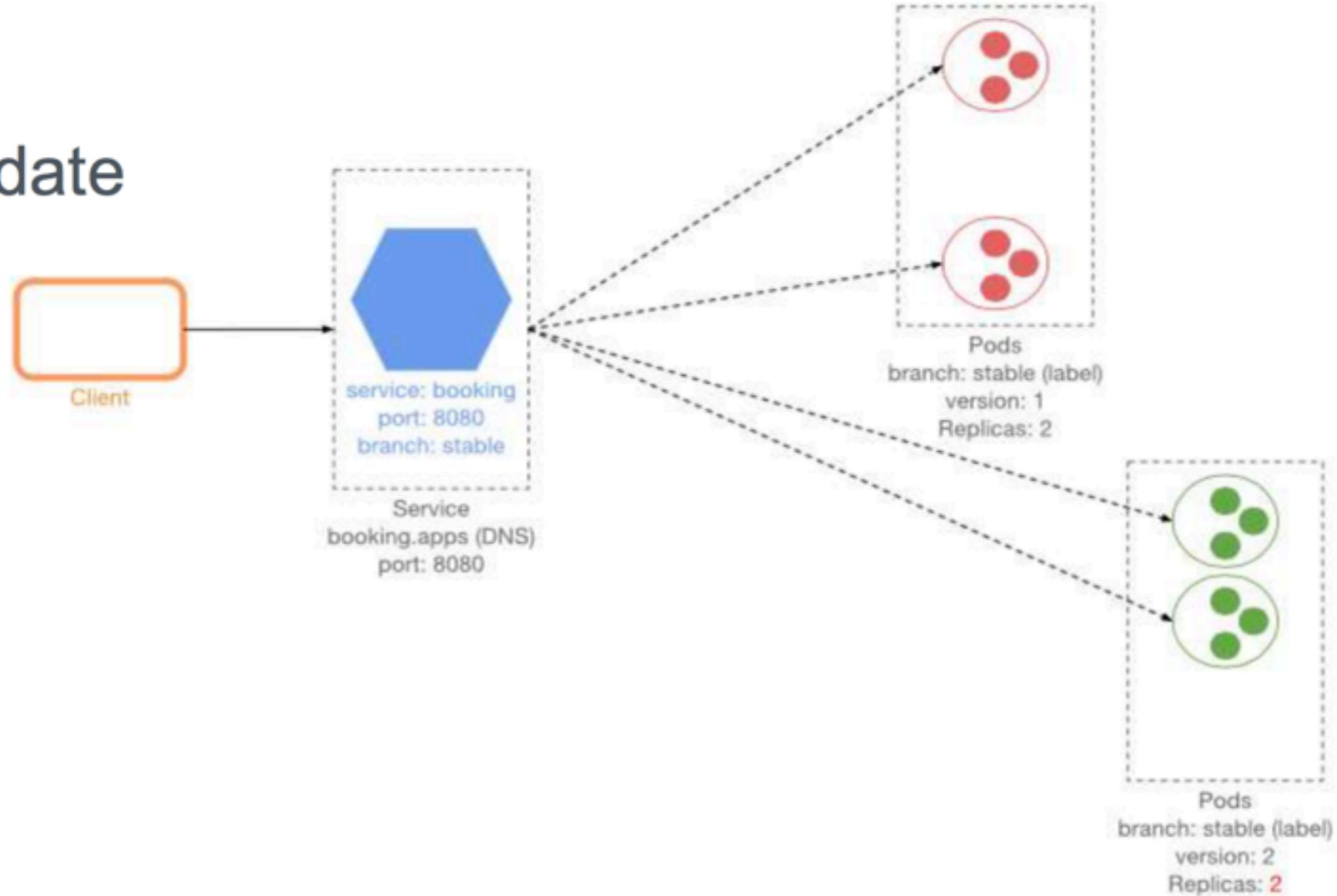
Rolling update (3)

Rolling Update



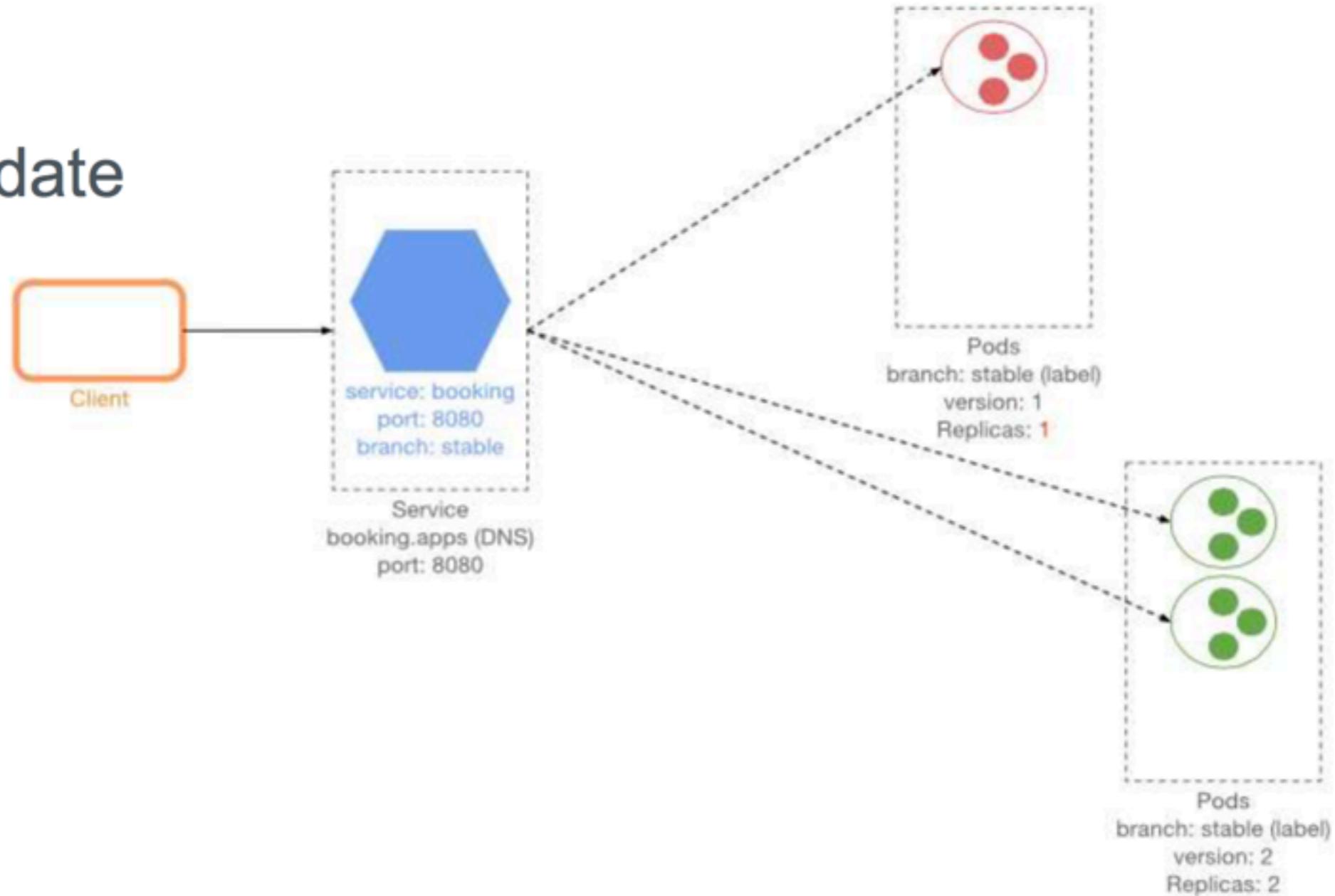
Rolling update (4)

Rolling Update



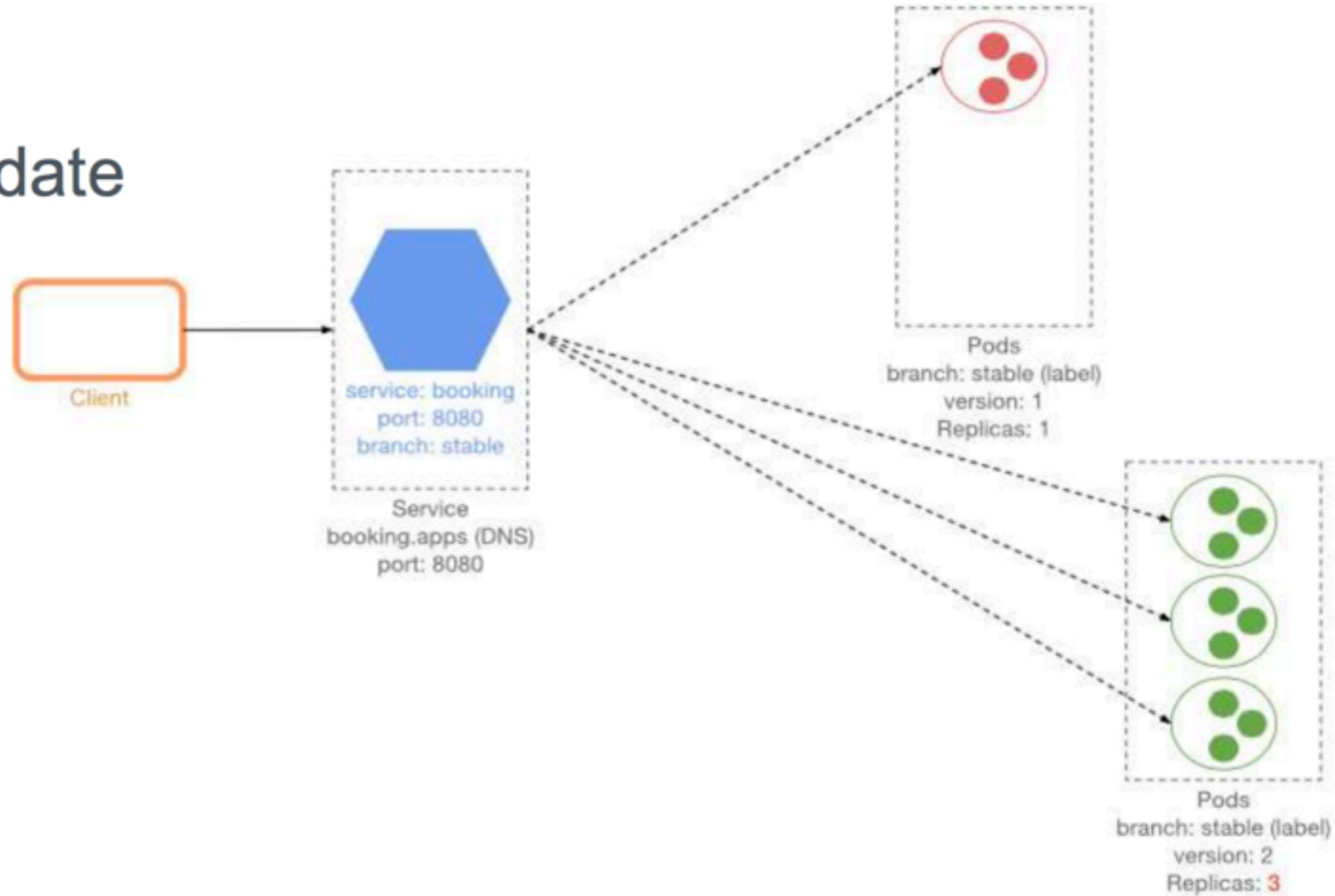
Rolling update (5)

Rolling Update



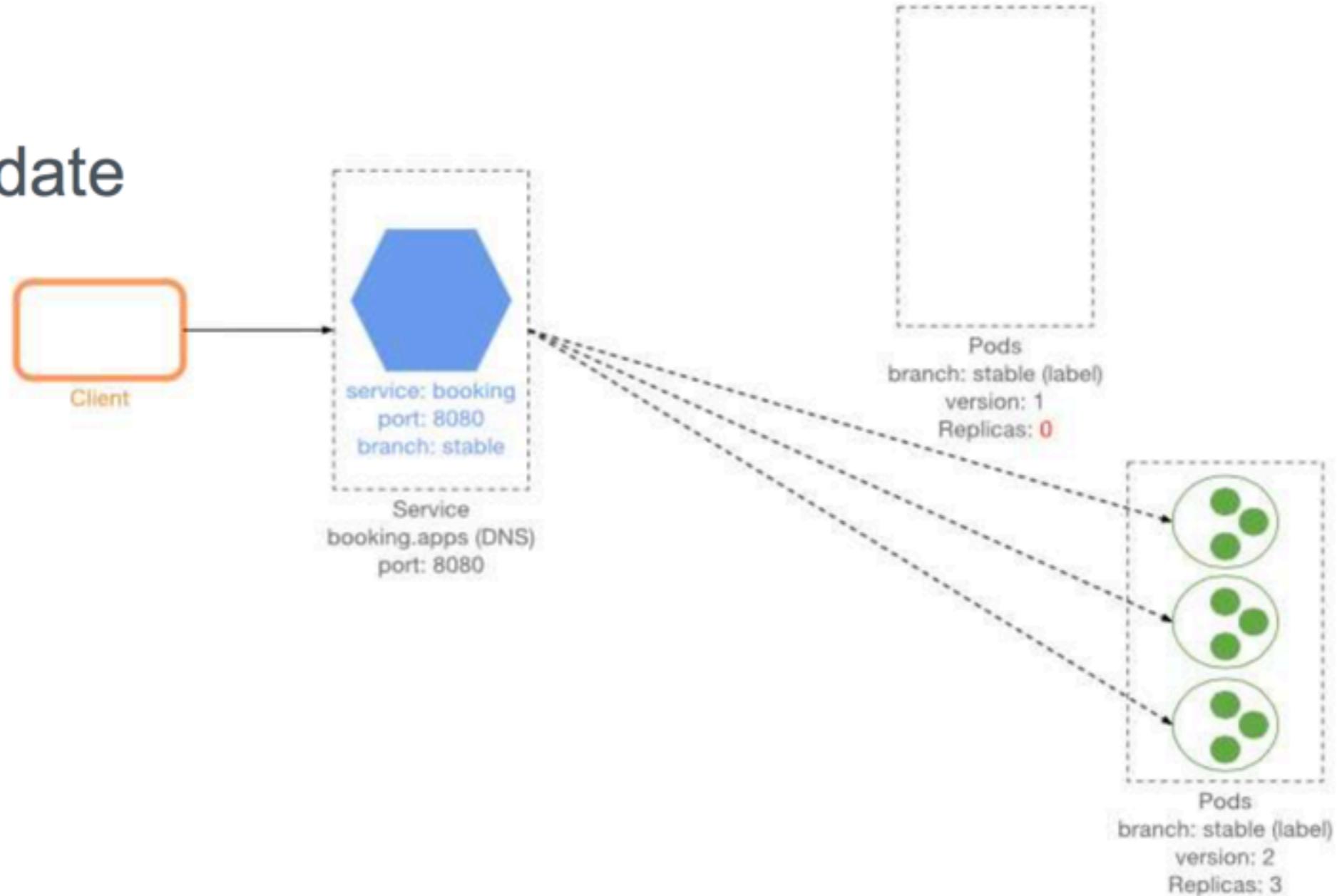
Rolling update (6)

Rolling Update



Rolling update (7)

Rolling Update



Storage orchestration



Storage orchestration

Support many type of storages

1. Local storage
2. Network storage (NFS, GlusterFS, Ceph)
3. Cloud storage (GCE, AWS, Azure)



Self-healing



Provide well known ports for Kubernetes services



Service Discovery and Load balance

Service is the connector/proxy for client to connect with Pod

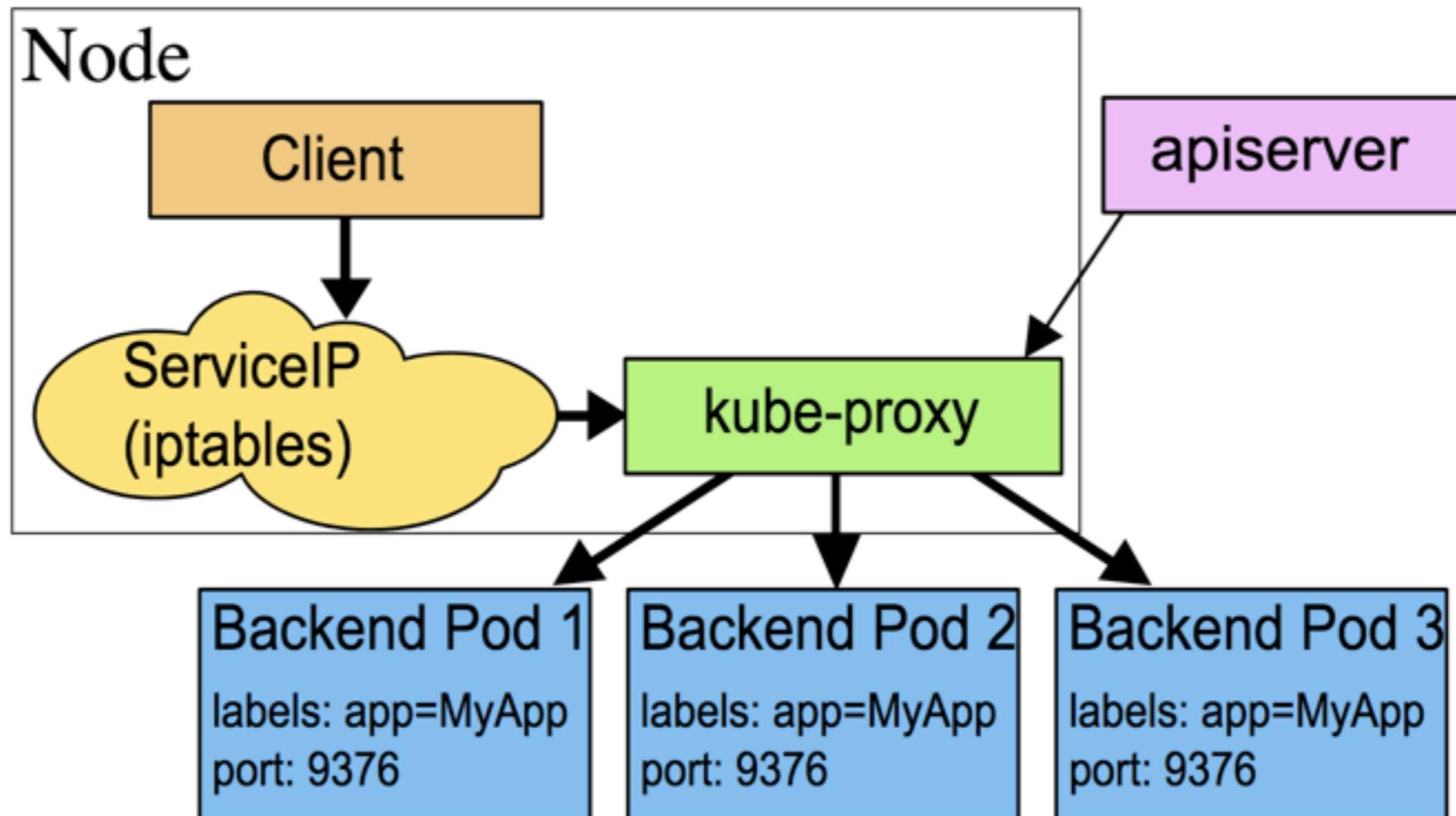
Discovery use for service to look Pod

Support load balance between Pods (replica)

<https://kubernetes.io/docs/concepts/services-networking/service/>



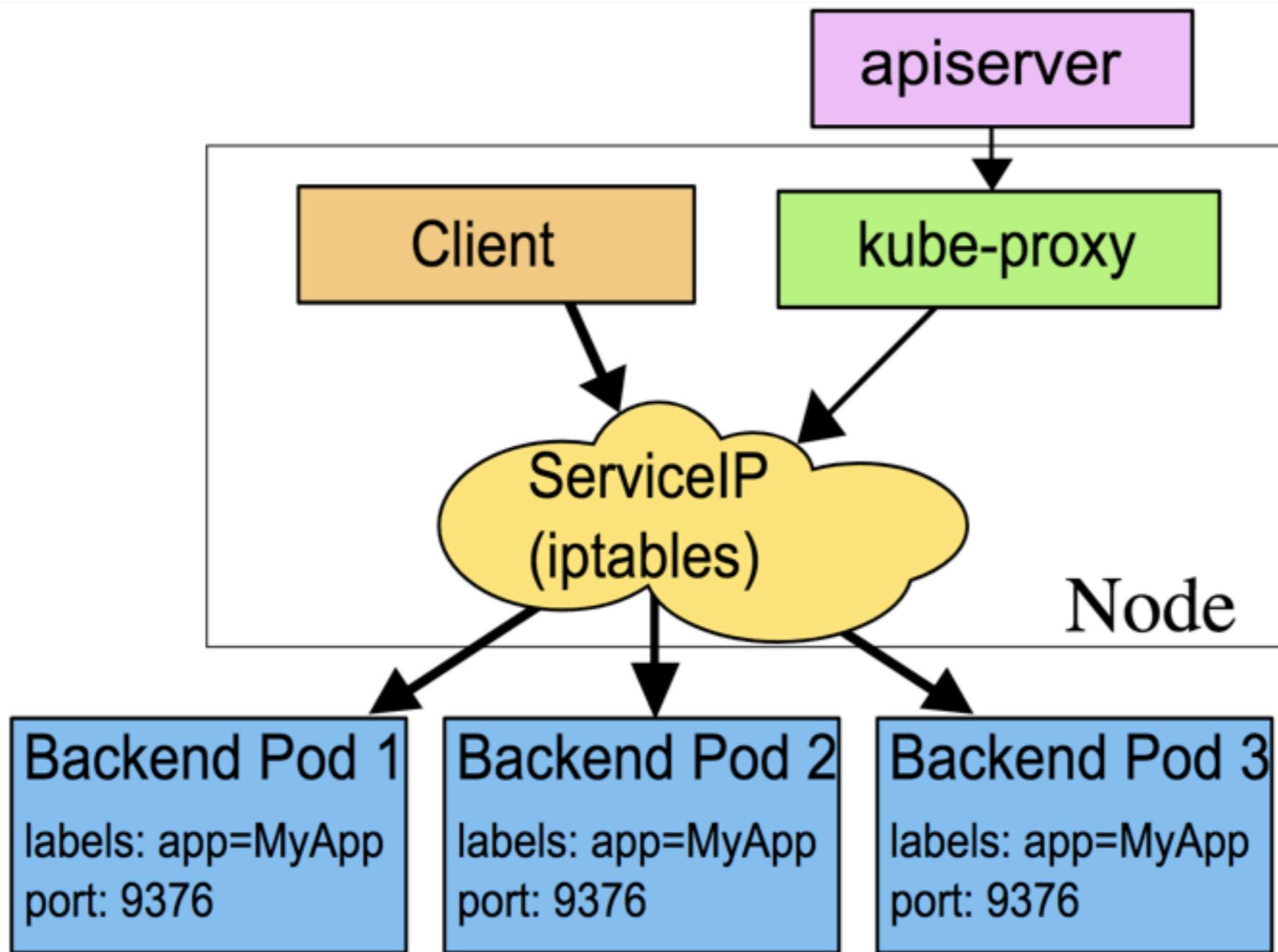
Proxy-mode: userspace



Note that in the above diagram, `clusterIP` is shown as `ServiceIP`.



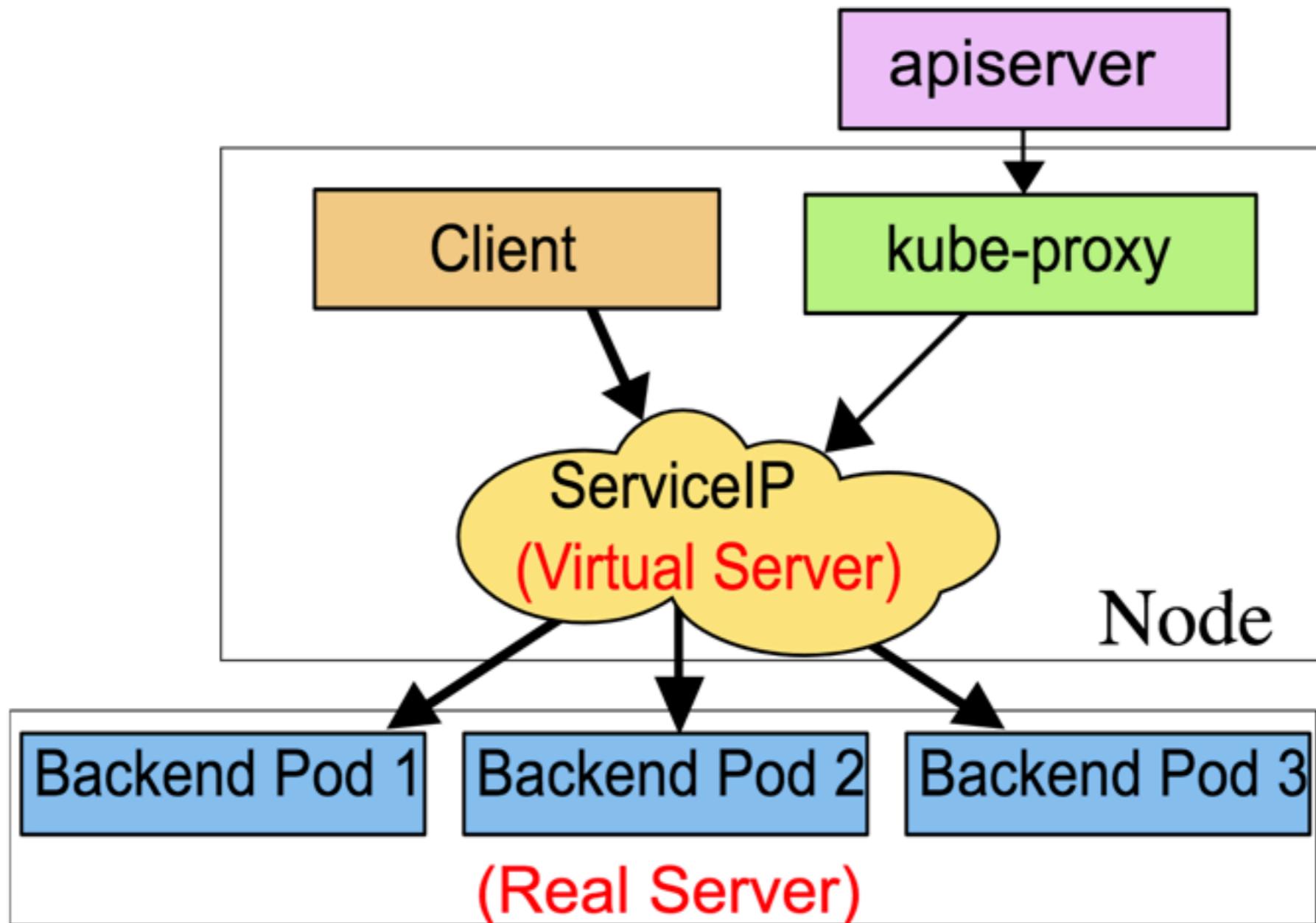
Proxy-mode: iptables



Note that in the above diagram, `clusterIP` is shown as `ServiceIP`.



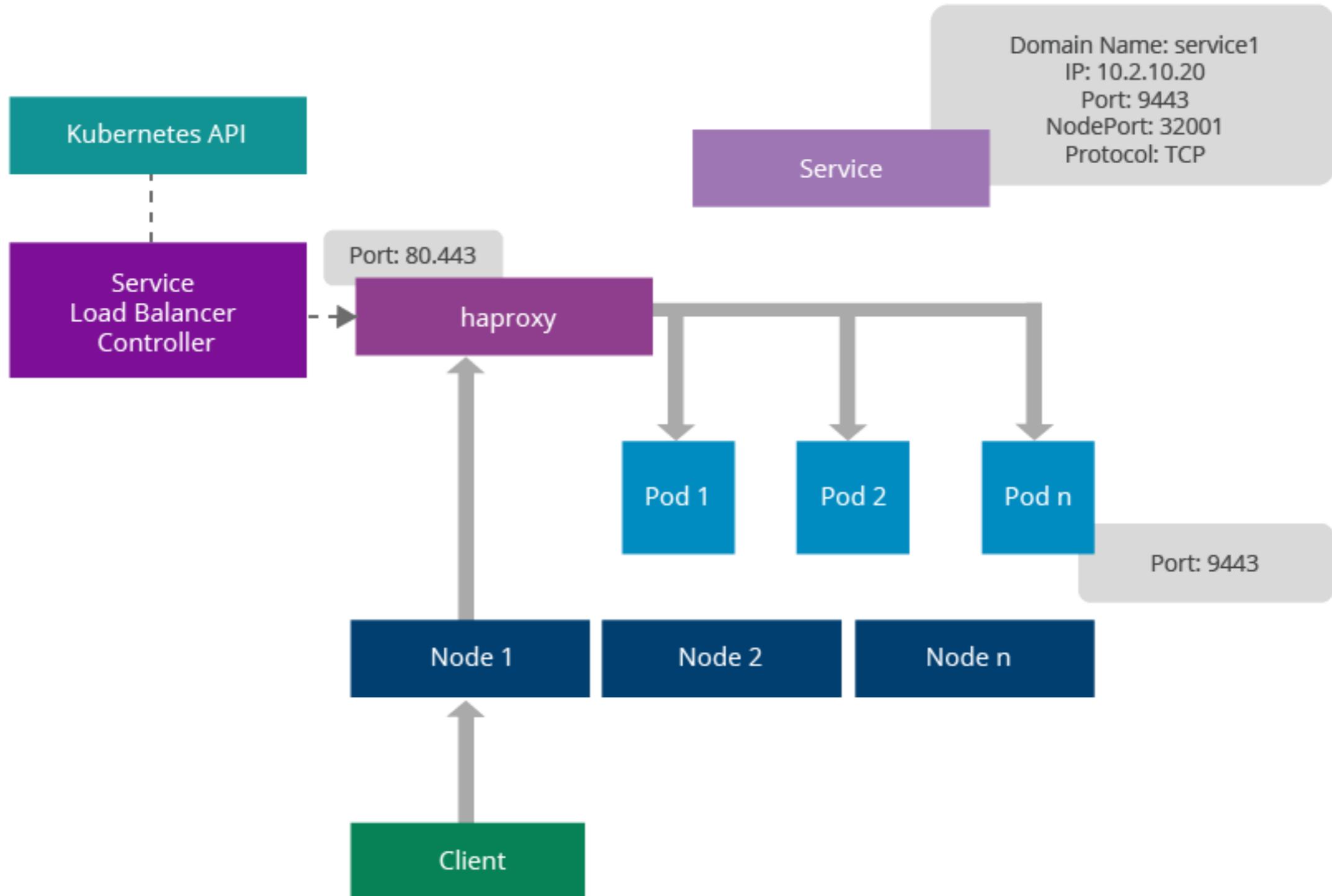
Proxy-mode: ipvs (beta 1.9)



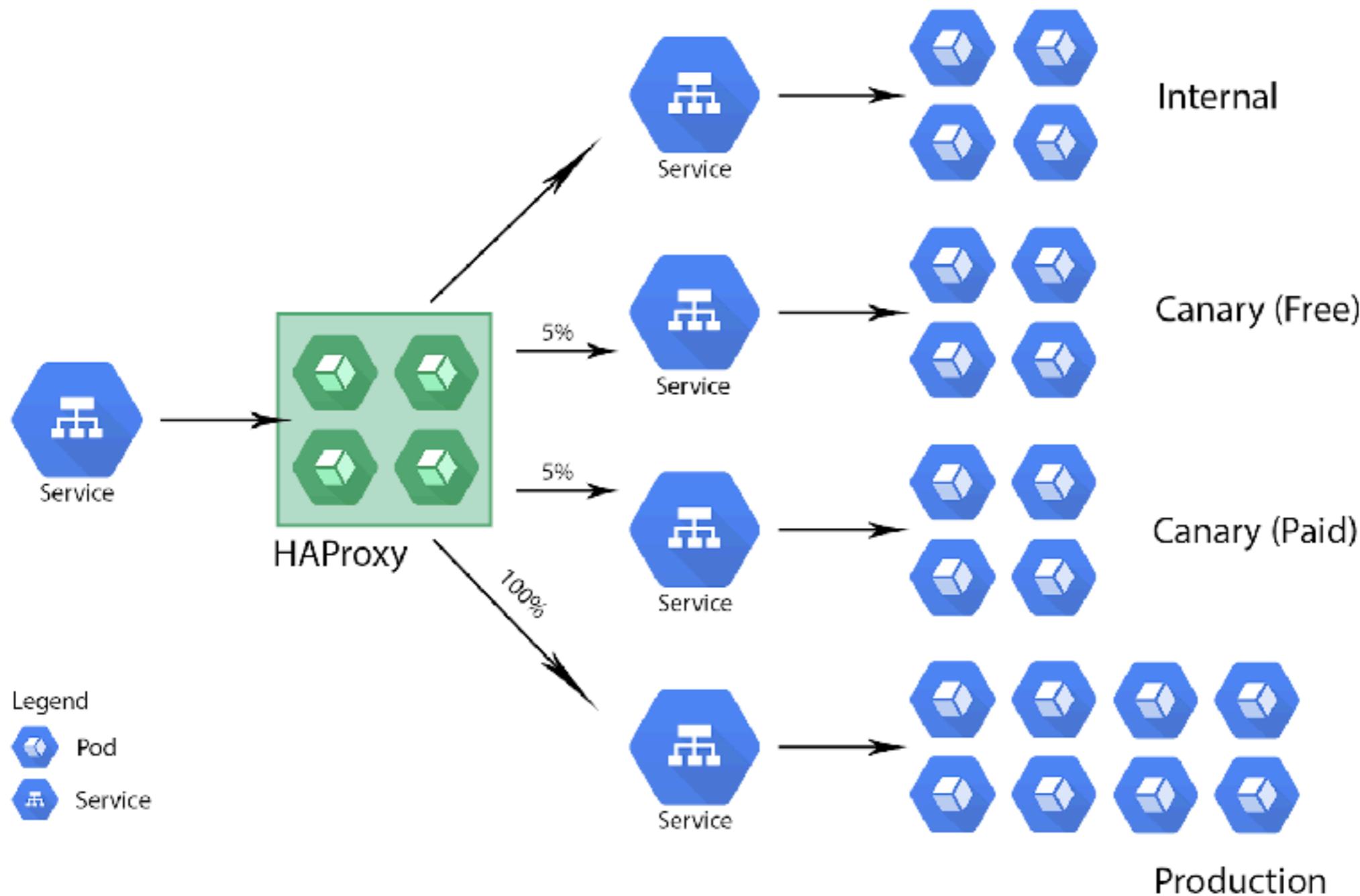
Sticky session management using Service Load Balancers



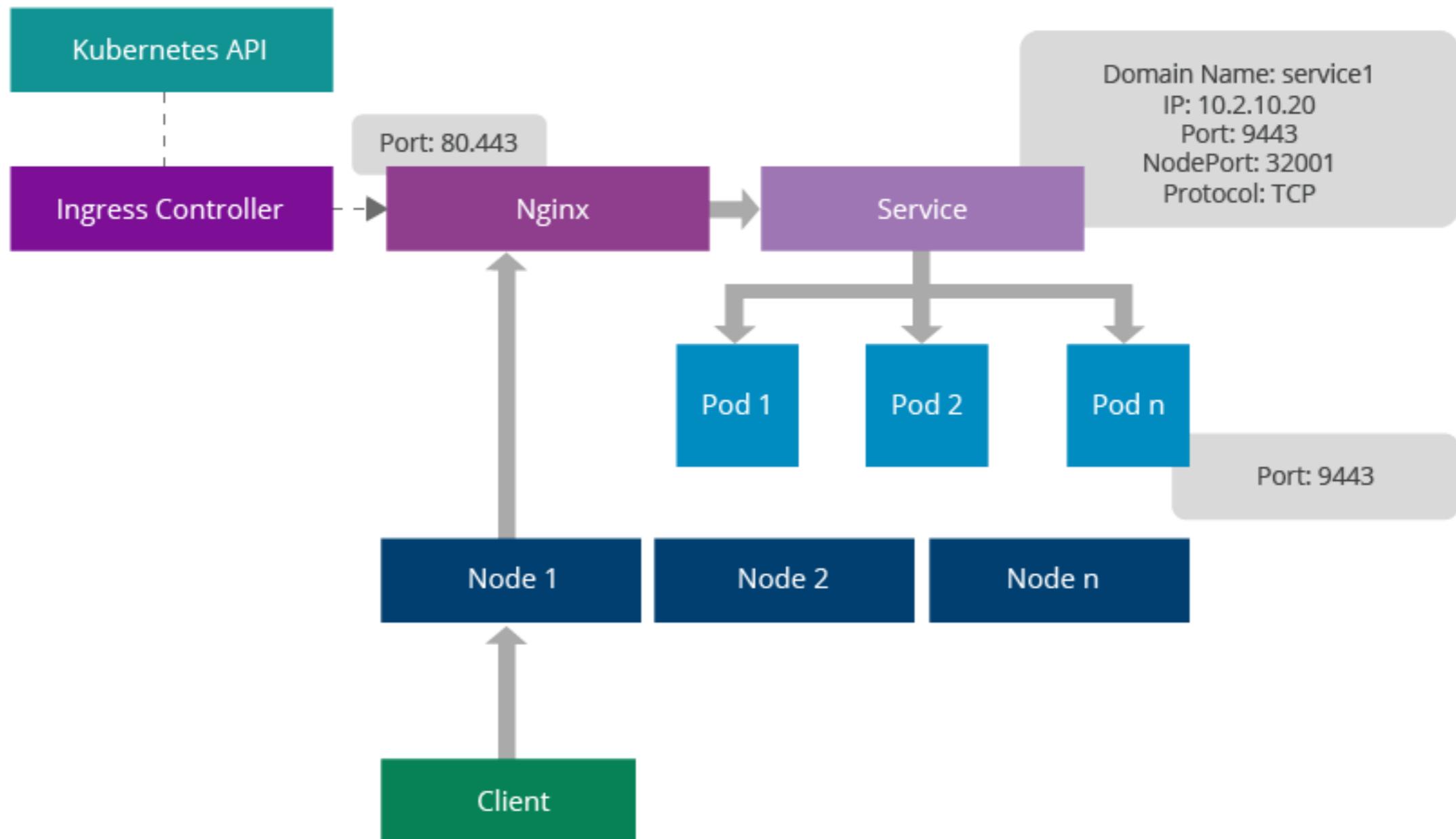
Service Load Balancer



Service Load Balancer



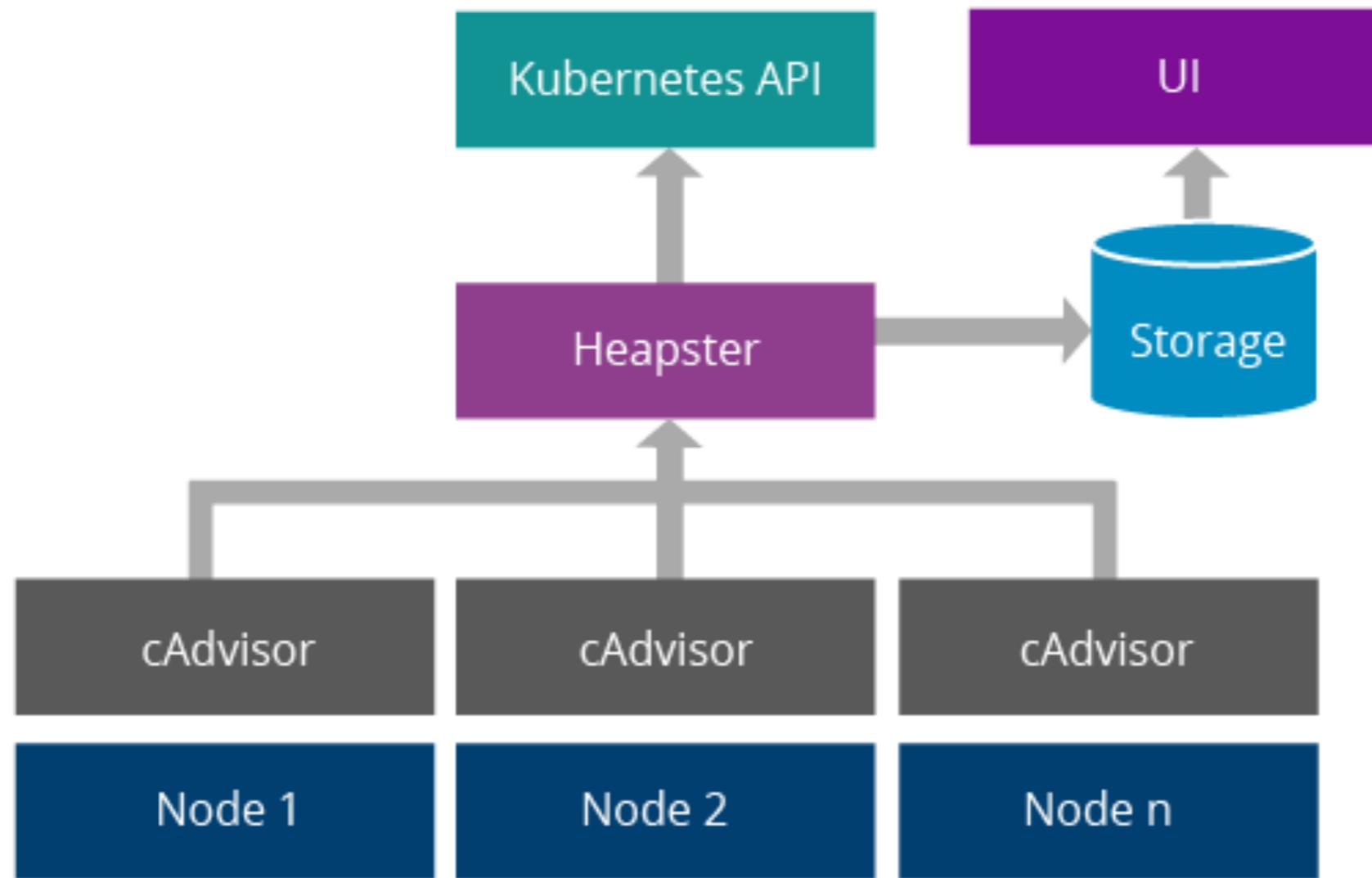
Ingress controller



Resource usage monitoring



Resource usage monitoring



<https://github.com/kubernetes/heapster>



Secret and configuration management



Secret and configuration mgt

Keep confidential data to running app in
encryption format

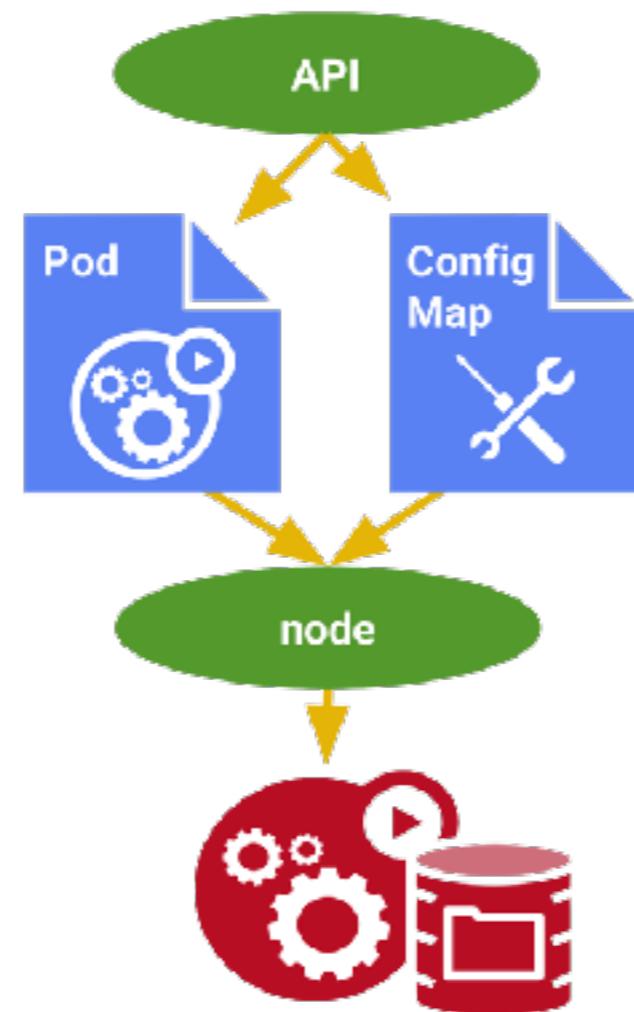
\$kubectl get secret

NAME	TYPE	DATA	AGE
default-token-26swm	kubernetes.io/service-account-token	3	2h



Secret and configuration mgt

Use **ConfigMap** to define all configuration that reference from Pod

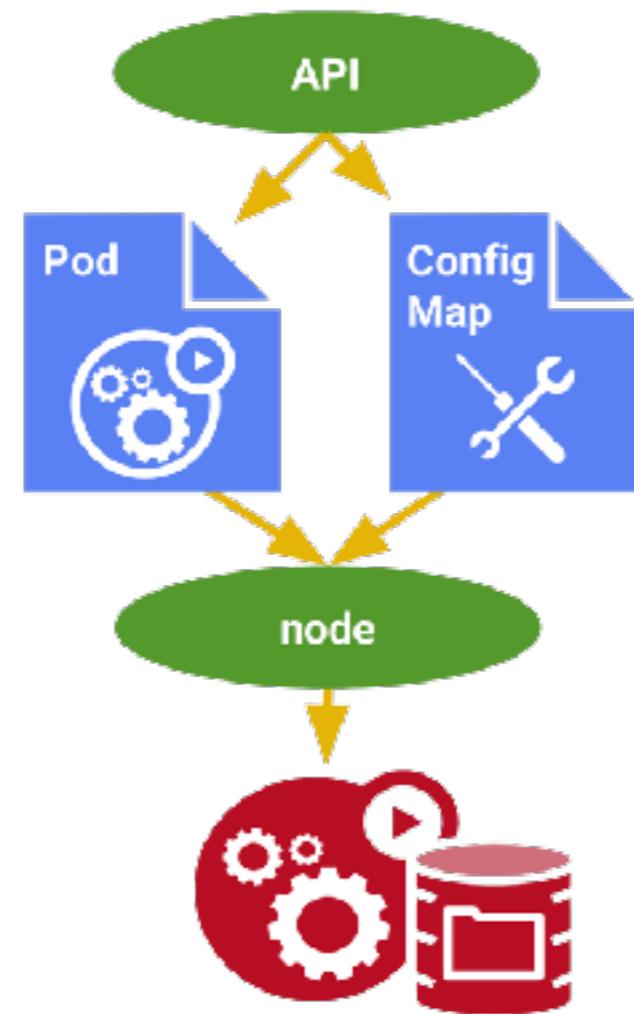


<https://kubernetes.io/docs/tasks/configure-pod-container/configure-pod-configmap/>



Secret and configuration mgt

Come from **12 factor app** :: config come from the environment



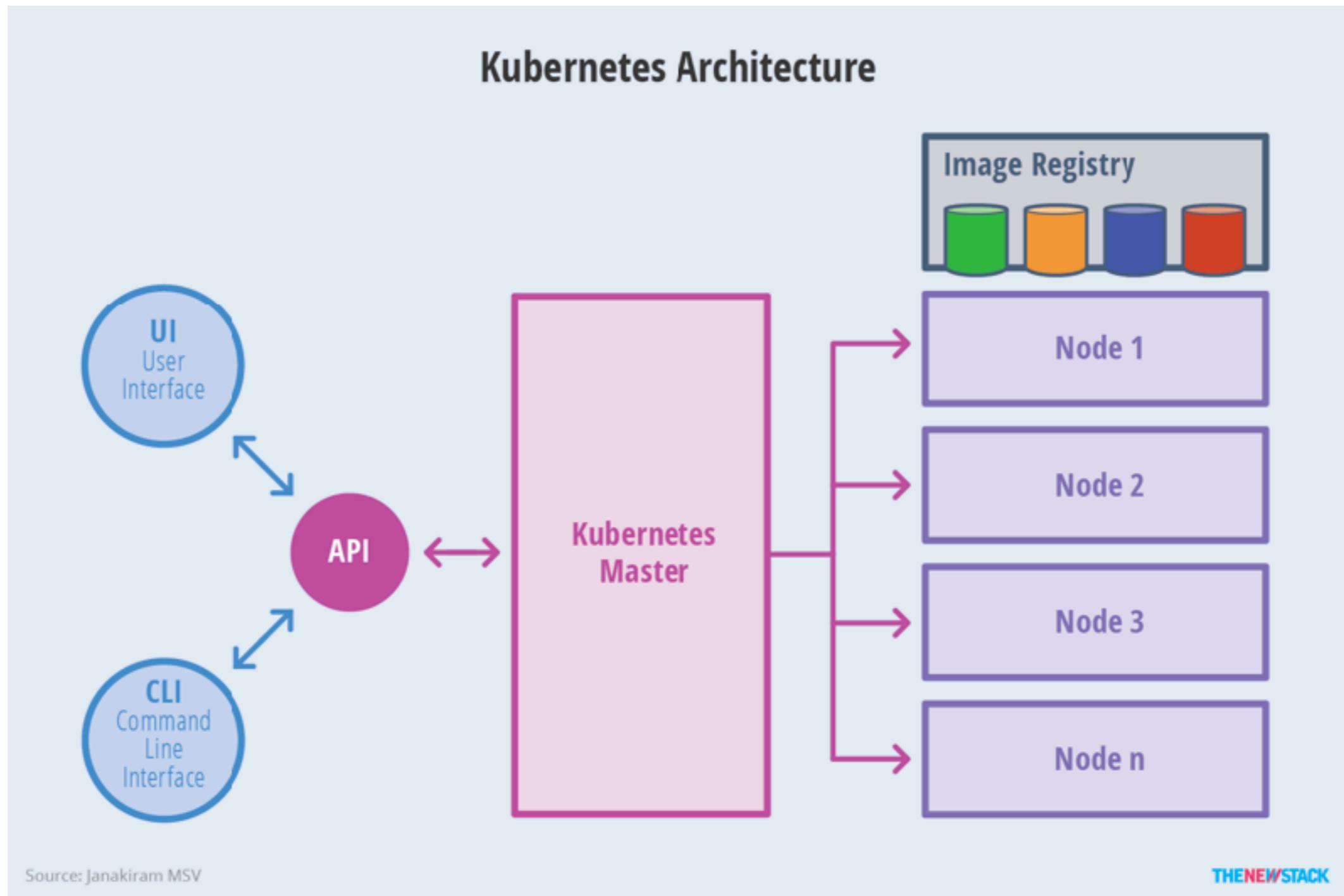
<https://12factor.net/>



System architecture



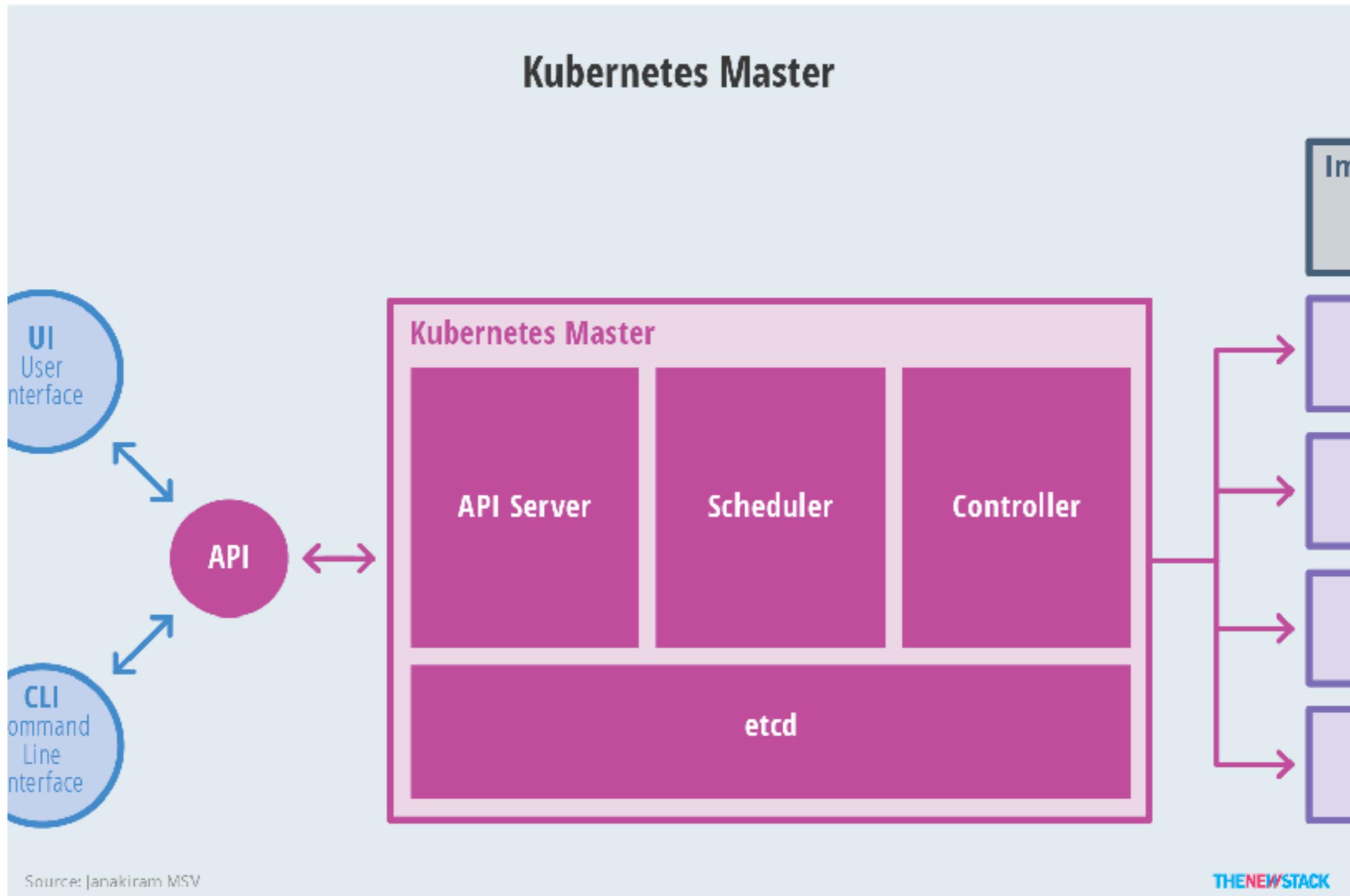
System architecture



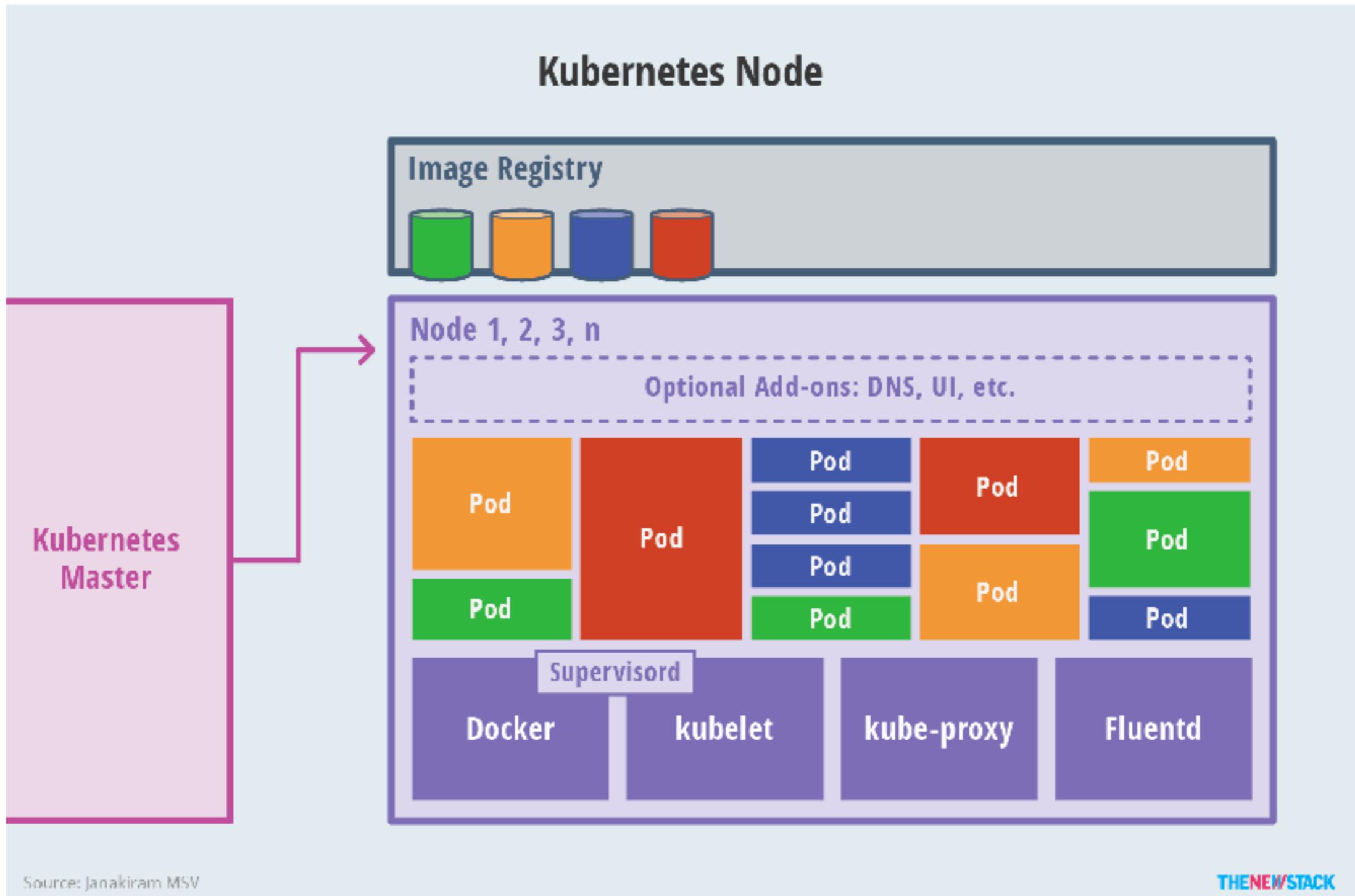
<https://thenewstack.io/kubernetes-an-overview/>



Master



Node



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



Core concepts of Kubernetes



Core concepts

Pods vs containers

Services

Replication Controllers

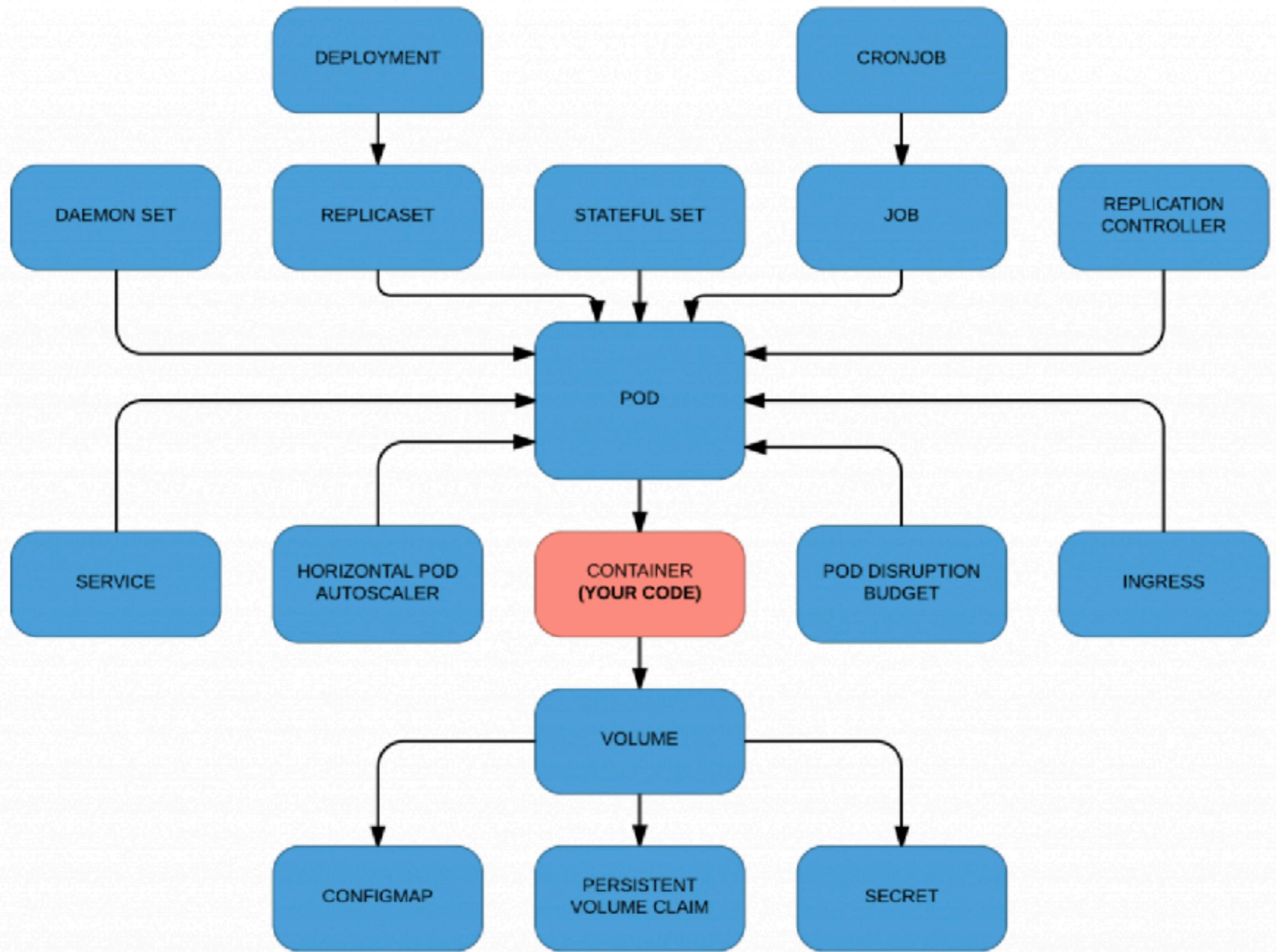
Deployments

ConfigMap and Secret

Volumes

StatefulSets



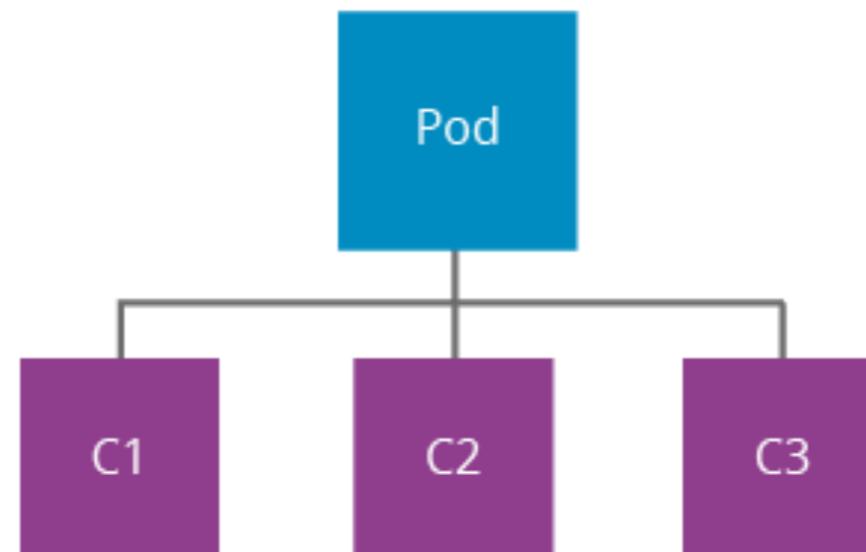


Pods vs containers



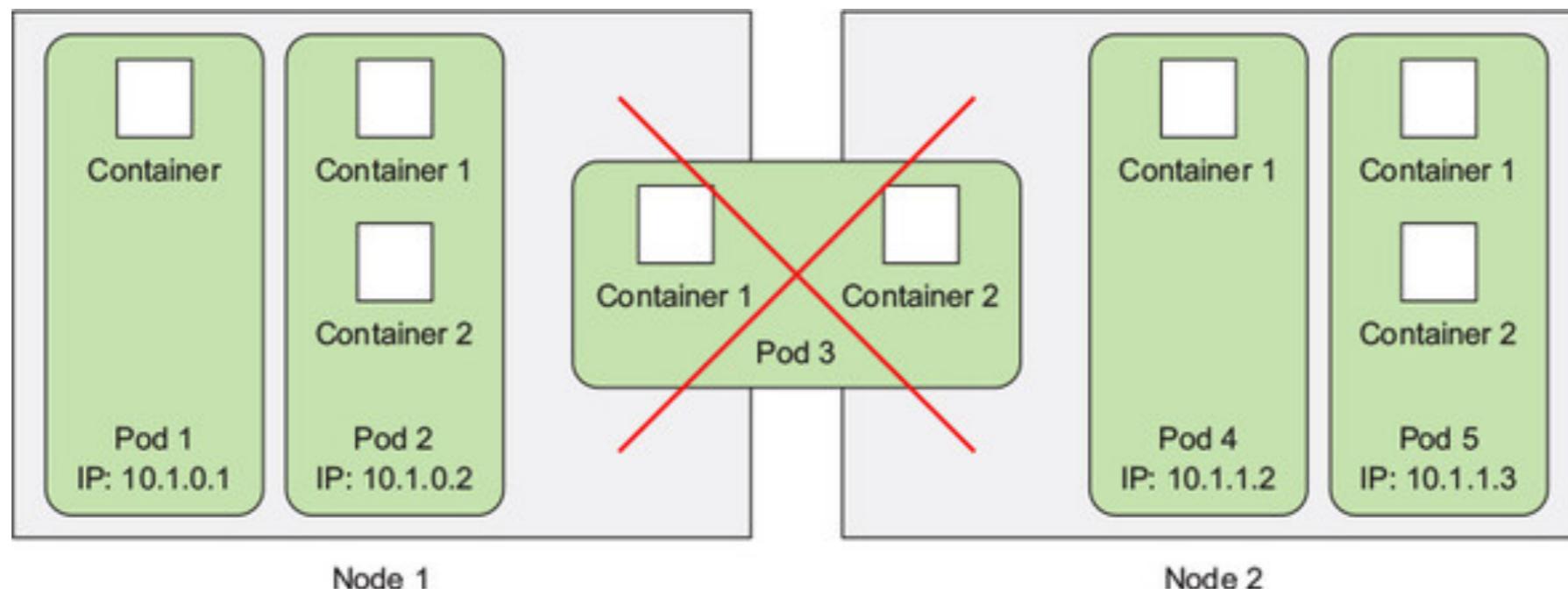
Pods

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

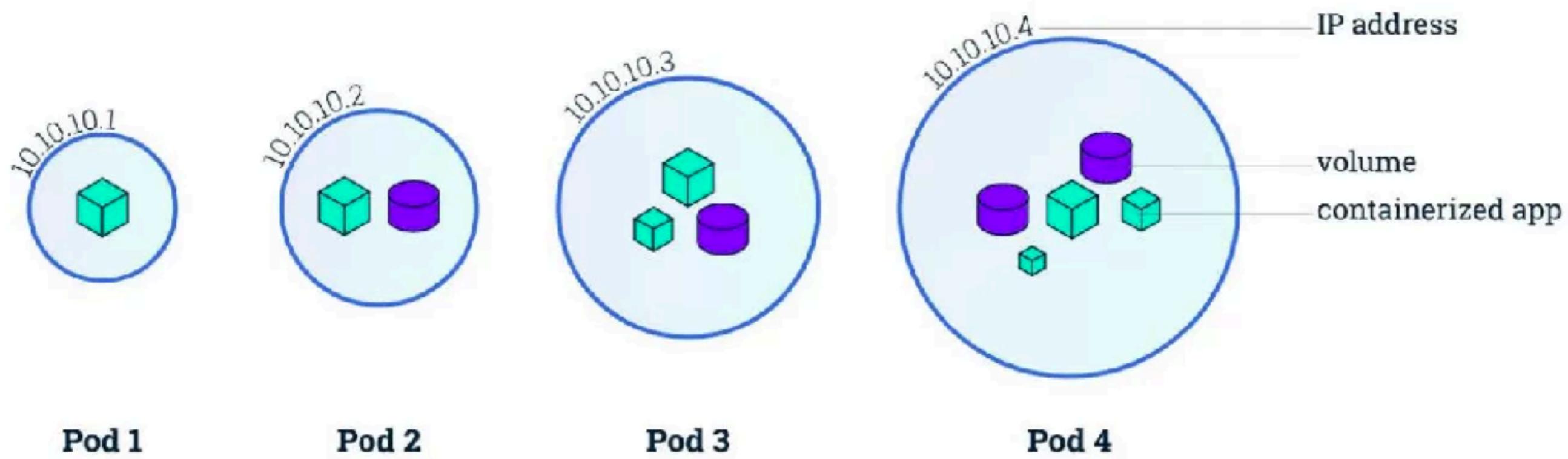
Share network interface

Share Hostname/IP/Port

Share Unix Time Sharing (UTS)

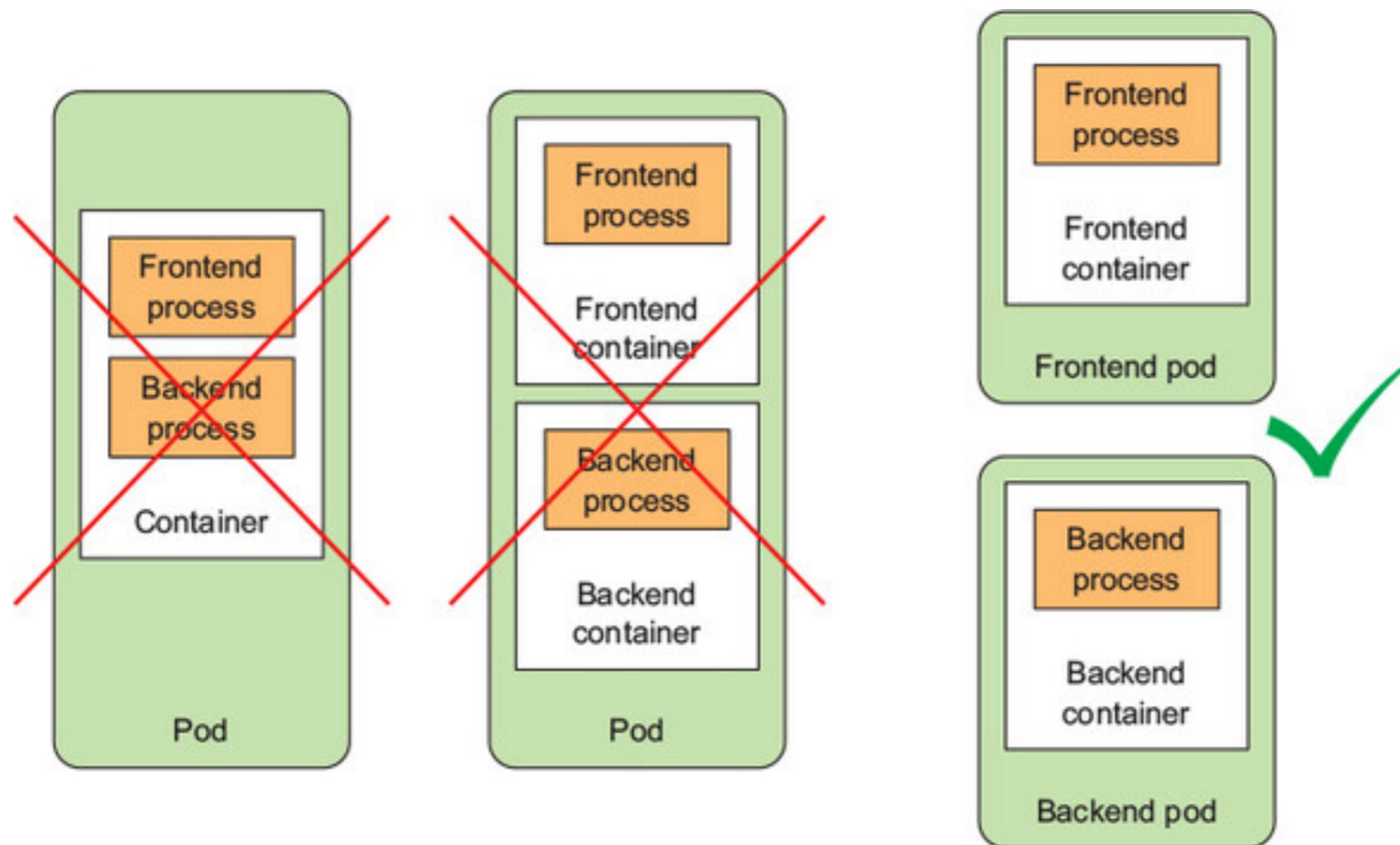


Pods



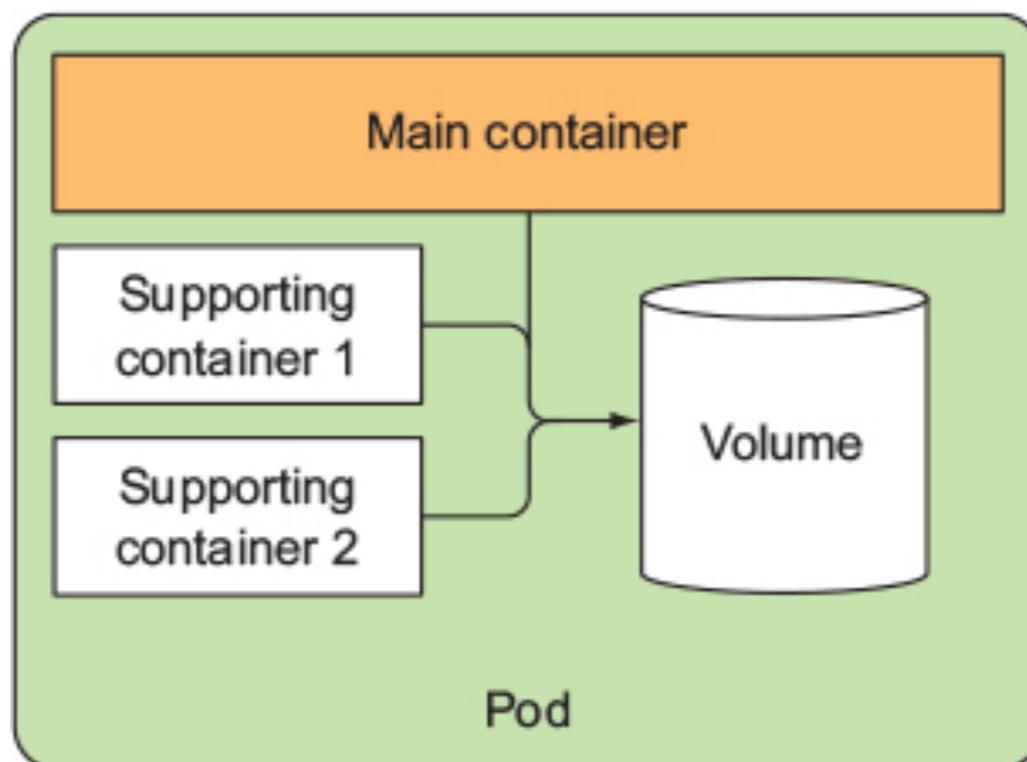
Organize container across Pods

Split multi-tiers app into multiple pods

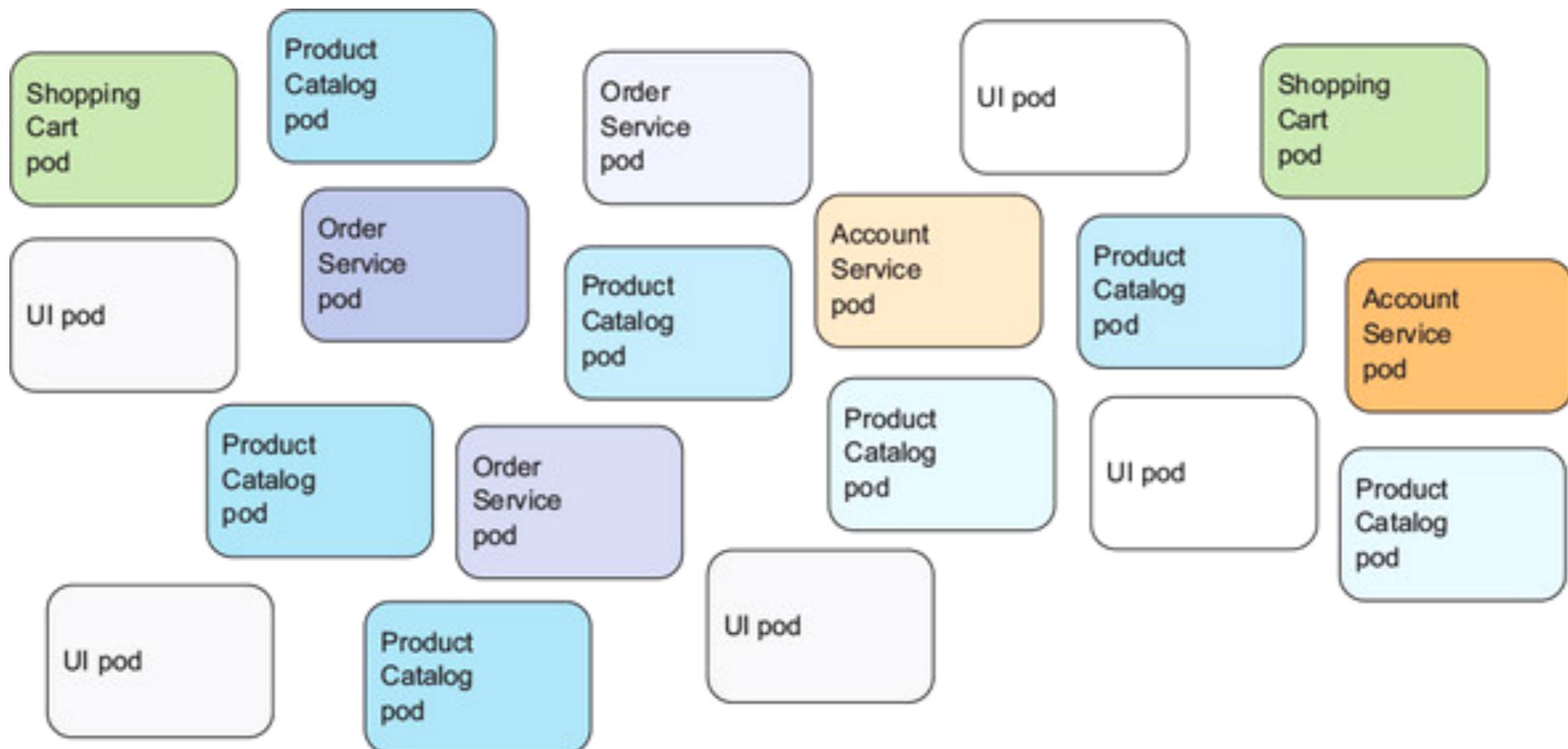


Organize container across Pods

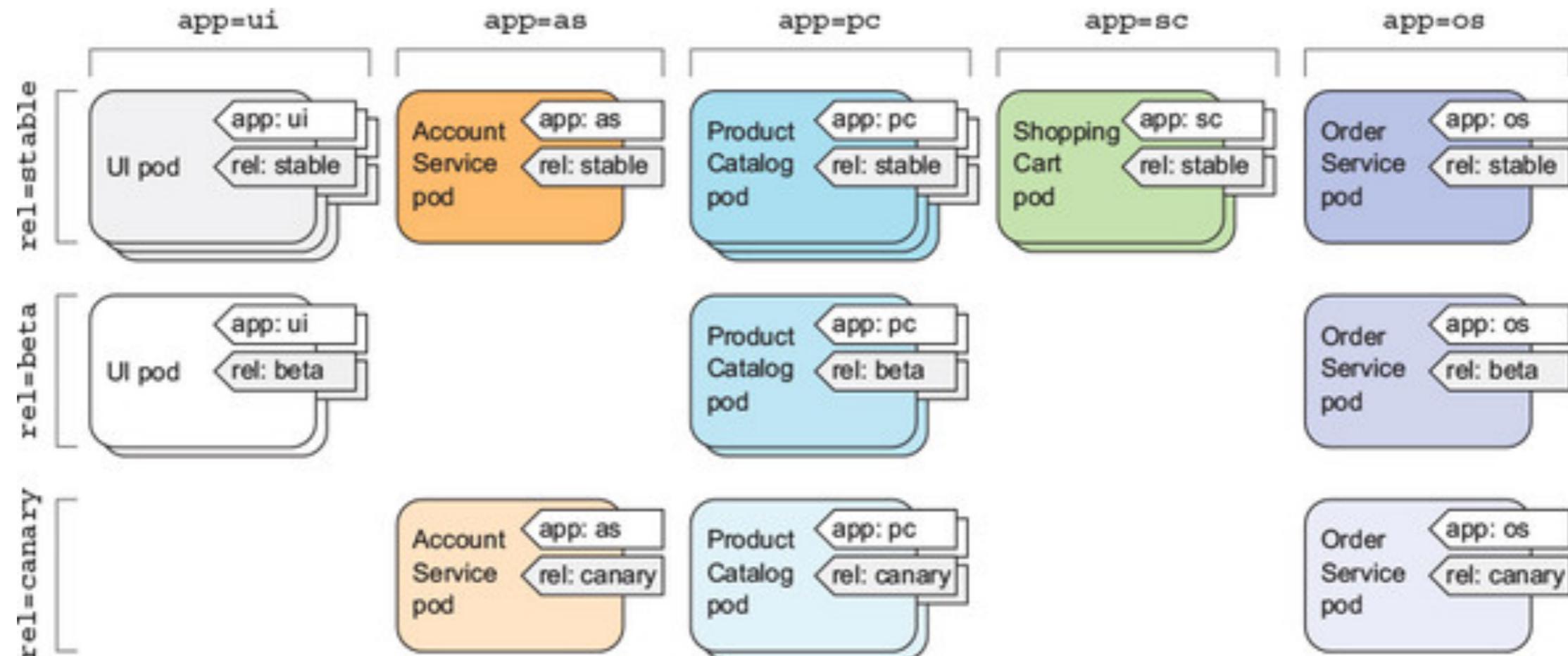
When to use multiple containers in a pods



Organize pods with Labels

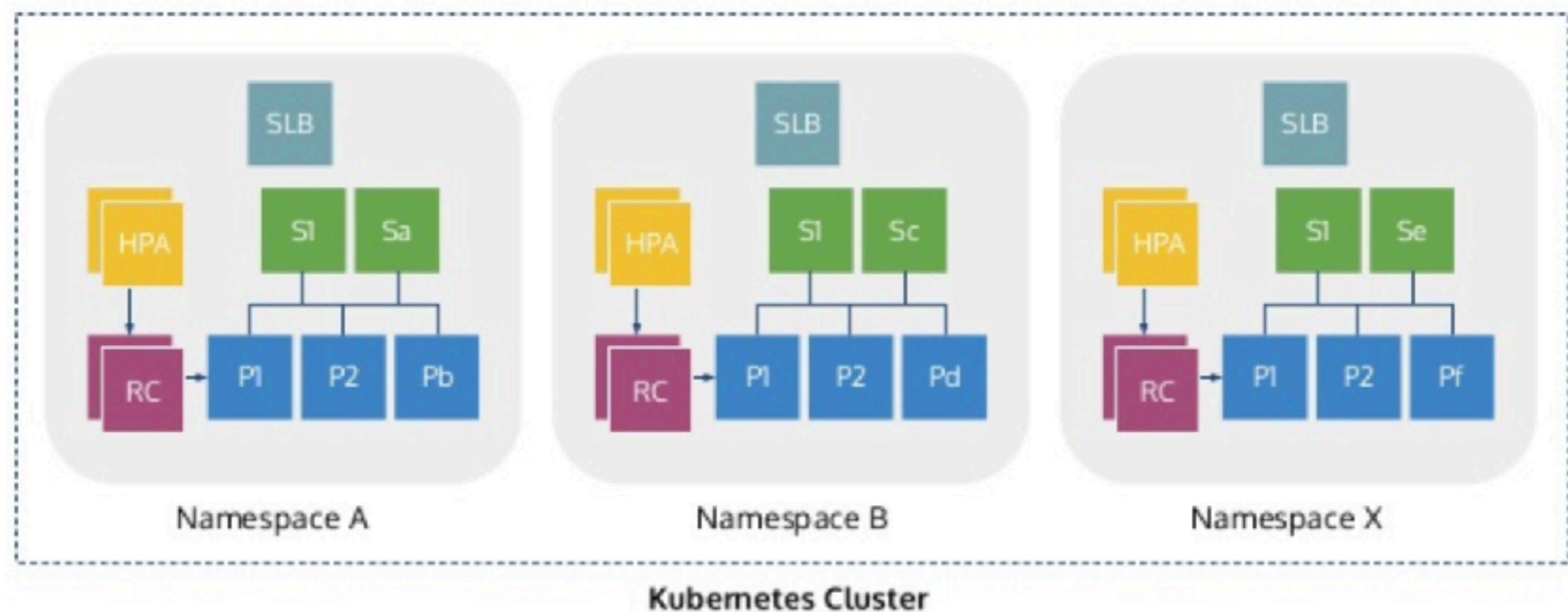


Organize pods with Labels



Pods namespaces

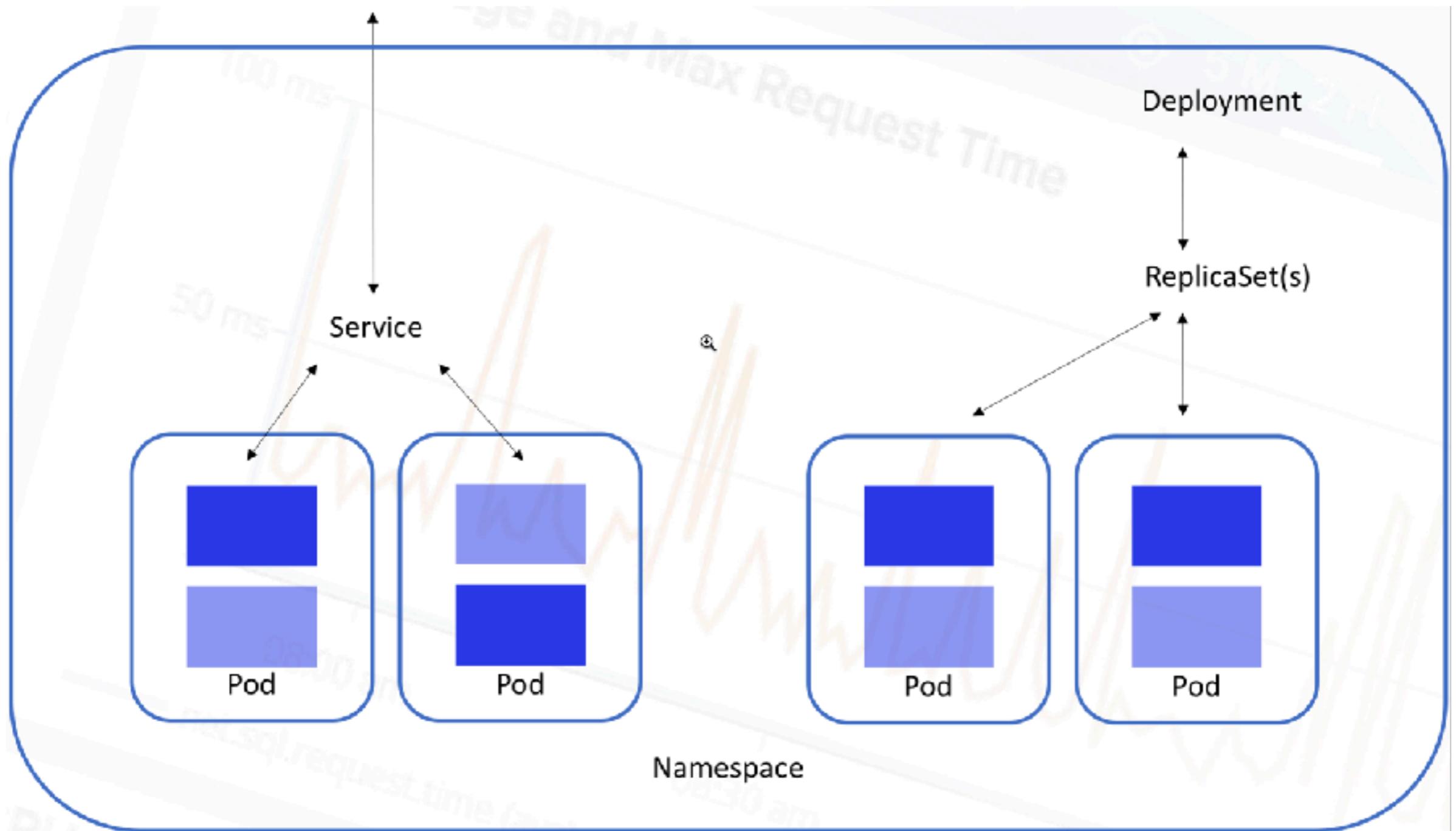
Allow different teams to use the same cluster



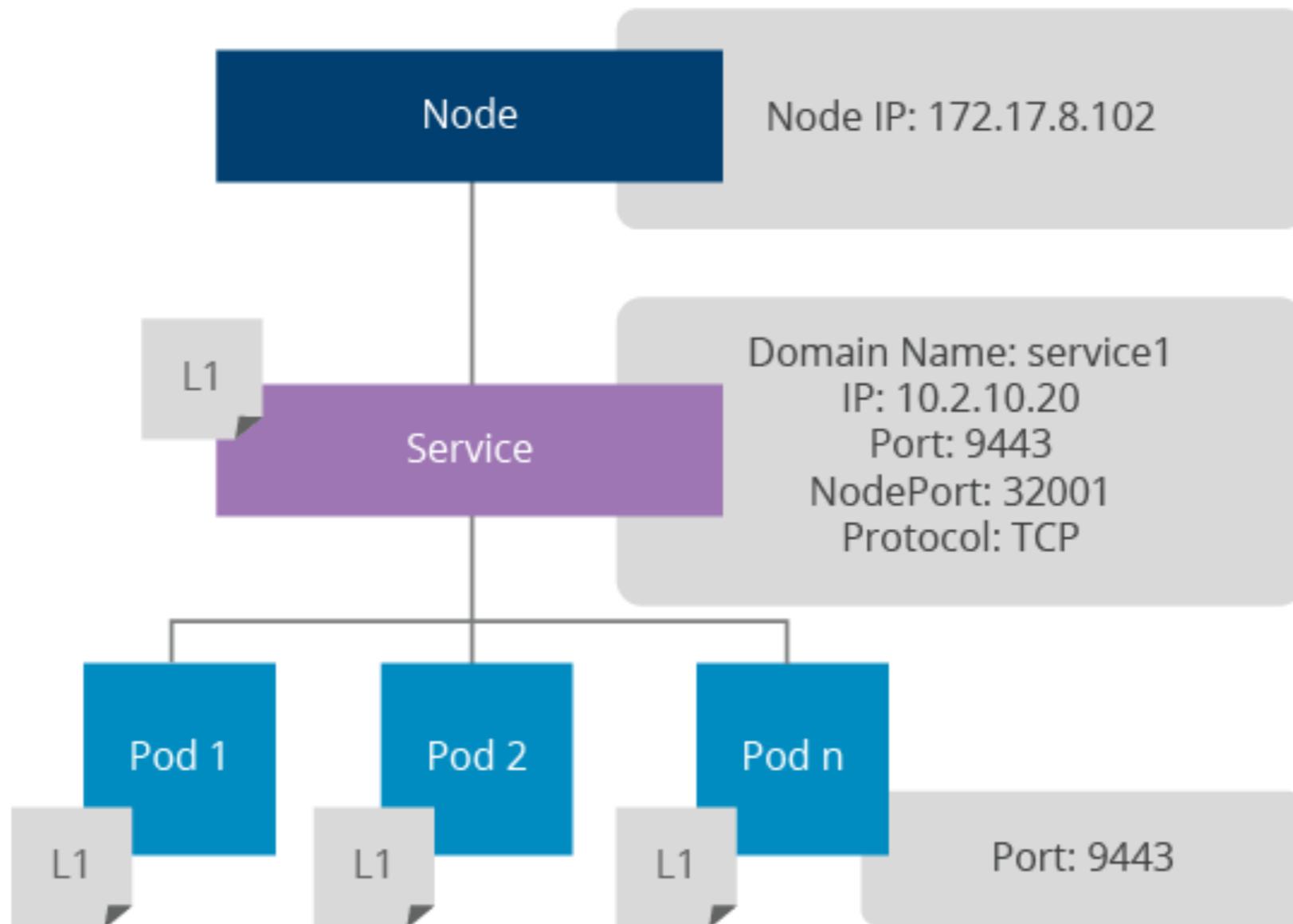
Services discovery and Load balancing



Services



Services

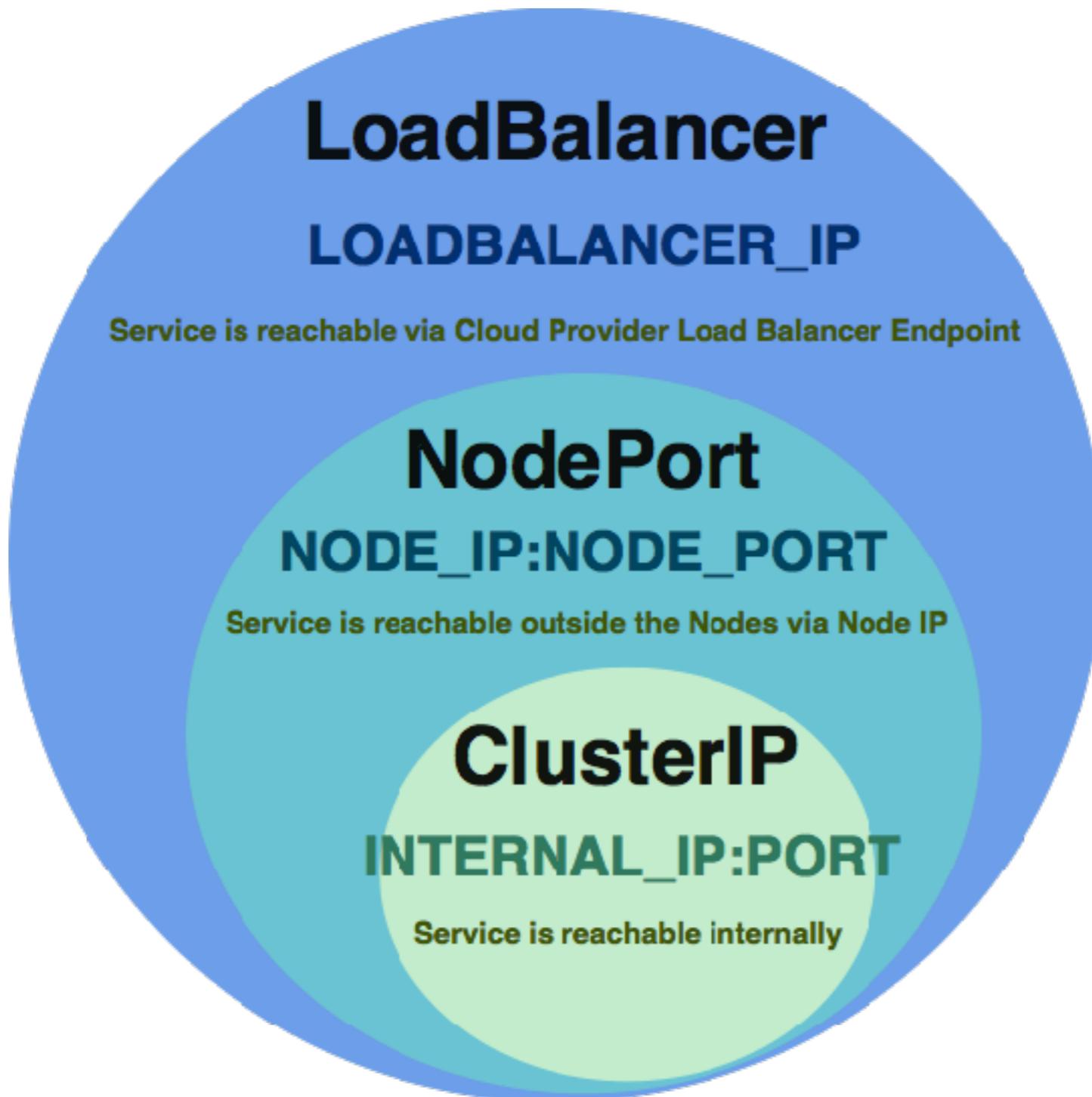


Services

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



3 types of services



Kubernetes Object Management

1. Imperative command
2. Imperative object configuration
3. Declarative object configuration

<https://kubernetes.io/docs/concepts/overview/object-management-kubectl/overview/>



1. Imperative command

```
$kubectl run nginx --image nginx
```

```
$kubectl create deployment nginx --image nginx
```



2. Imperative object configuration

```
$kubectl create -f nginx.yaml
```

```
$kubectl delete -f nginx.yaml
```



3. Declarative object configuration

```
$kubectl apply -f configs/  
$kubectl apply -R -f configs/
```



Kubernetes Object Management

Management technique	Operates on	Recommended environment	Supported writers	Learning curve
Imperative commands	Live objects	Development projects	1+	Lowest
Imperative object configuration	Individual files	Production projects	1	Moderate
Declarative object configuration	Directories of files	Production projects	1+	Highest

Warning: A Kubernetes object should be managed **using only one technique**. Mixing and matching techniques for the same object results in undefined behavior.



Workshop :: Pods & Services

1 container per Pods

**file /02-pod-service/single-pod/
workshop_instruction.txt**



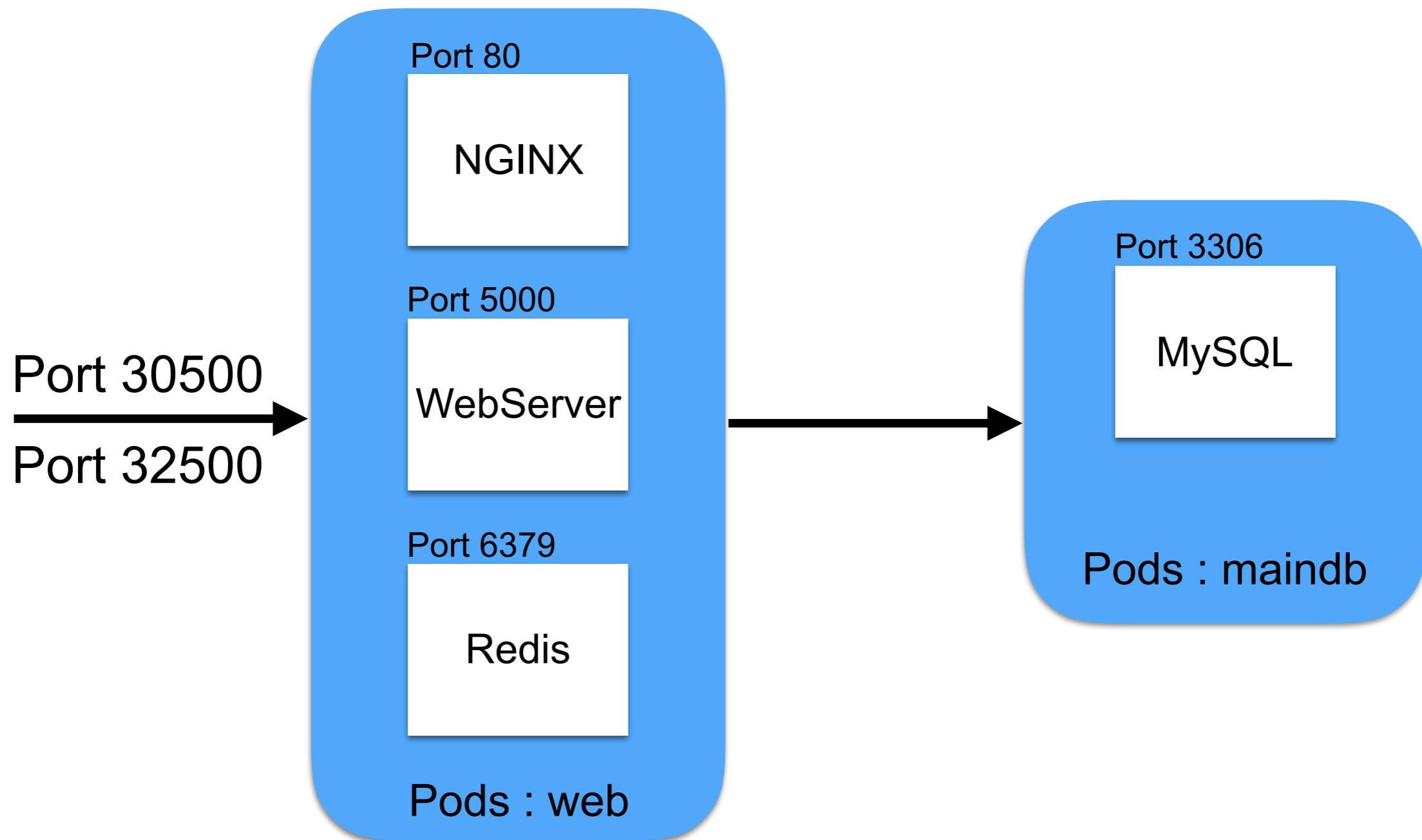
Workshop :: Pods & Services

N container per Pods

file /01-starter/instruction.txt



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



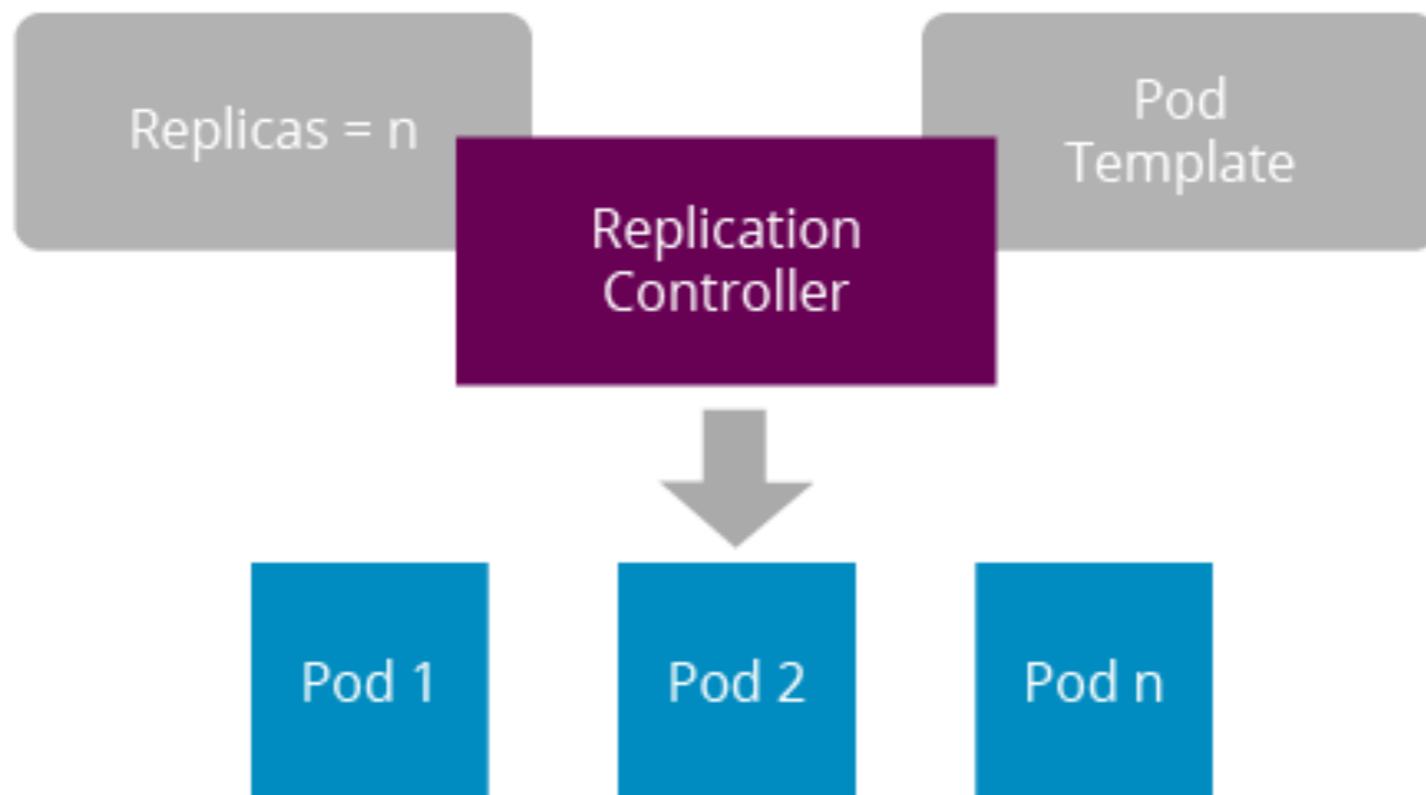
Replication Controller

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

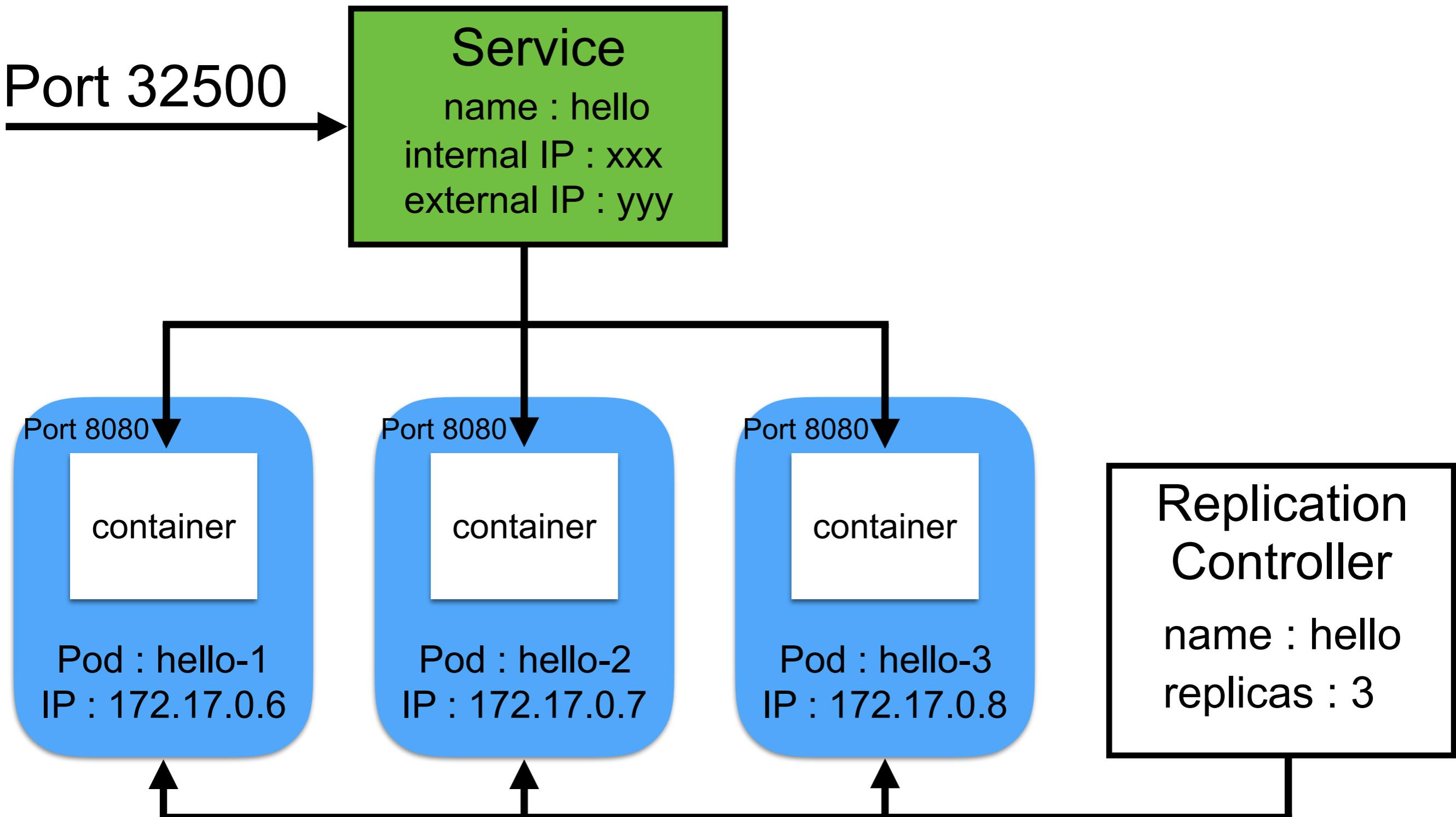


Workshop Replication Controller

file /03-replication-controller



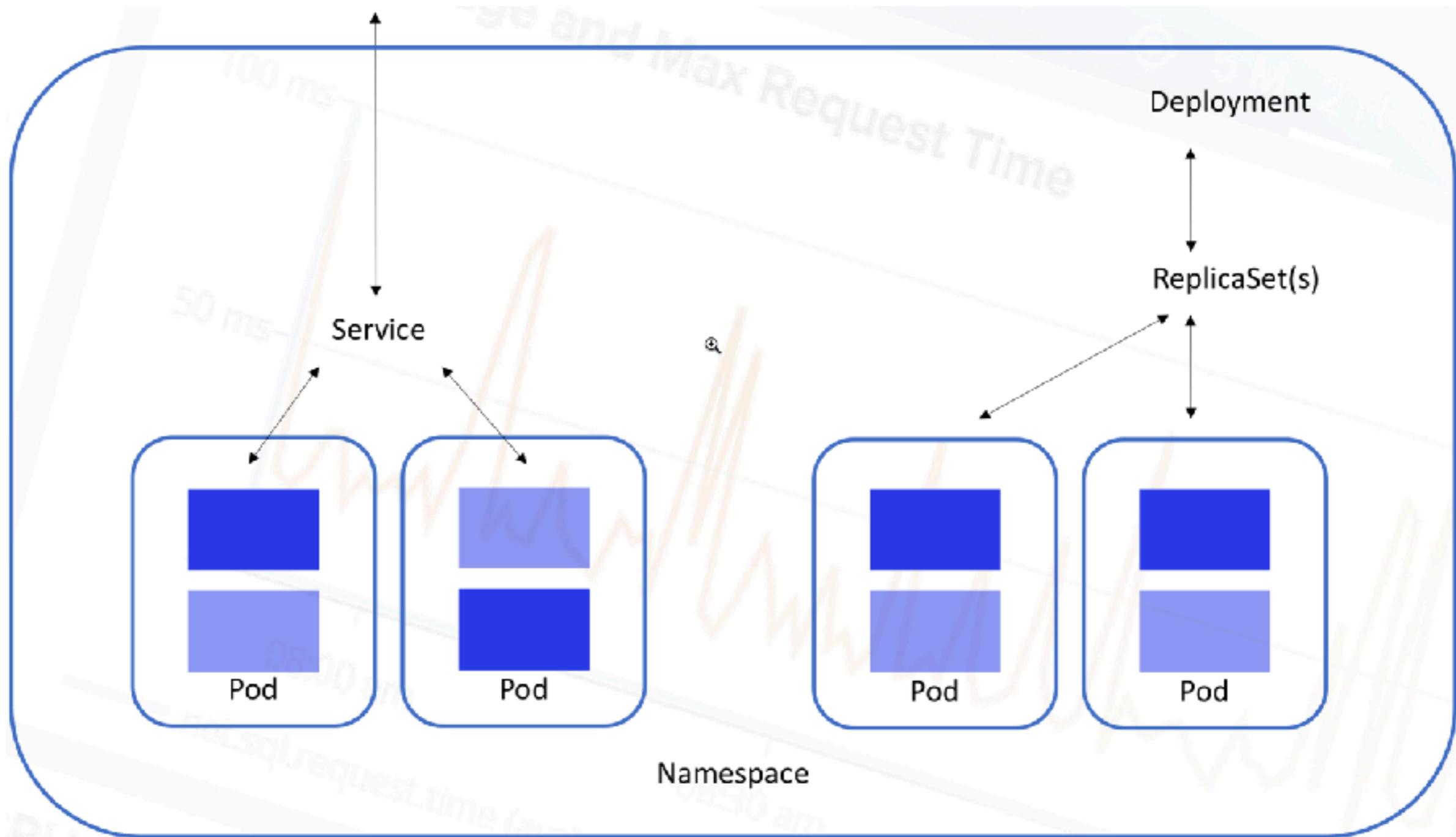
Scaling the application



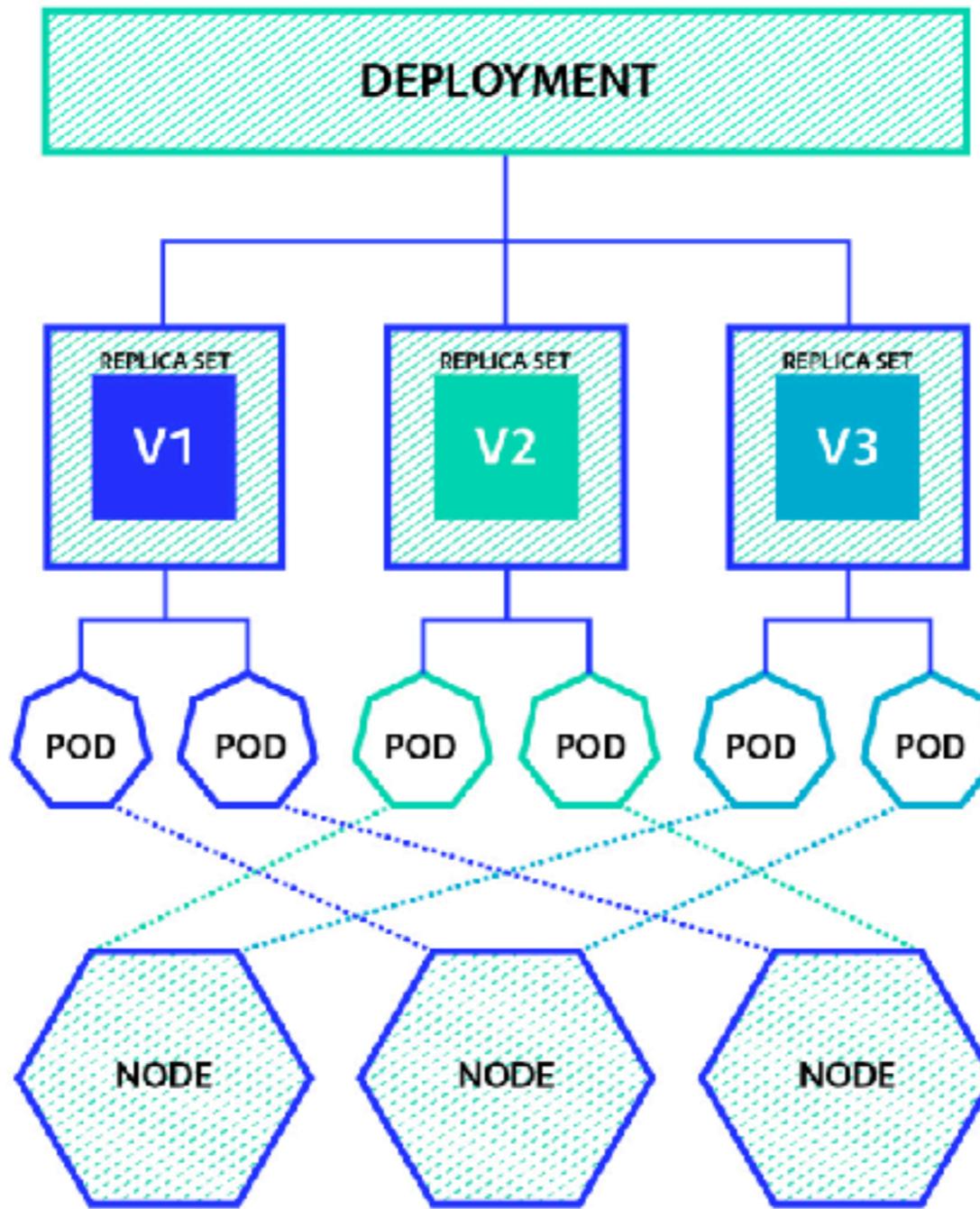
Deployment and ReplicaSet (RS)



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 file it
# will be 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,"selector":{"matchLabels":{"app":"web"}}, "template":{"metadata":{"labels":{"app":"web"}}, "spec":{"containers":[{"image":"nginx:1.13","ports":[{"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>
```

```
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/ReplicaSet

file /04-deployment-and-replica-set

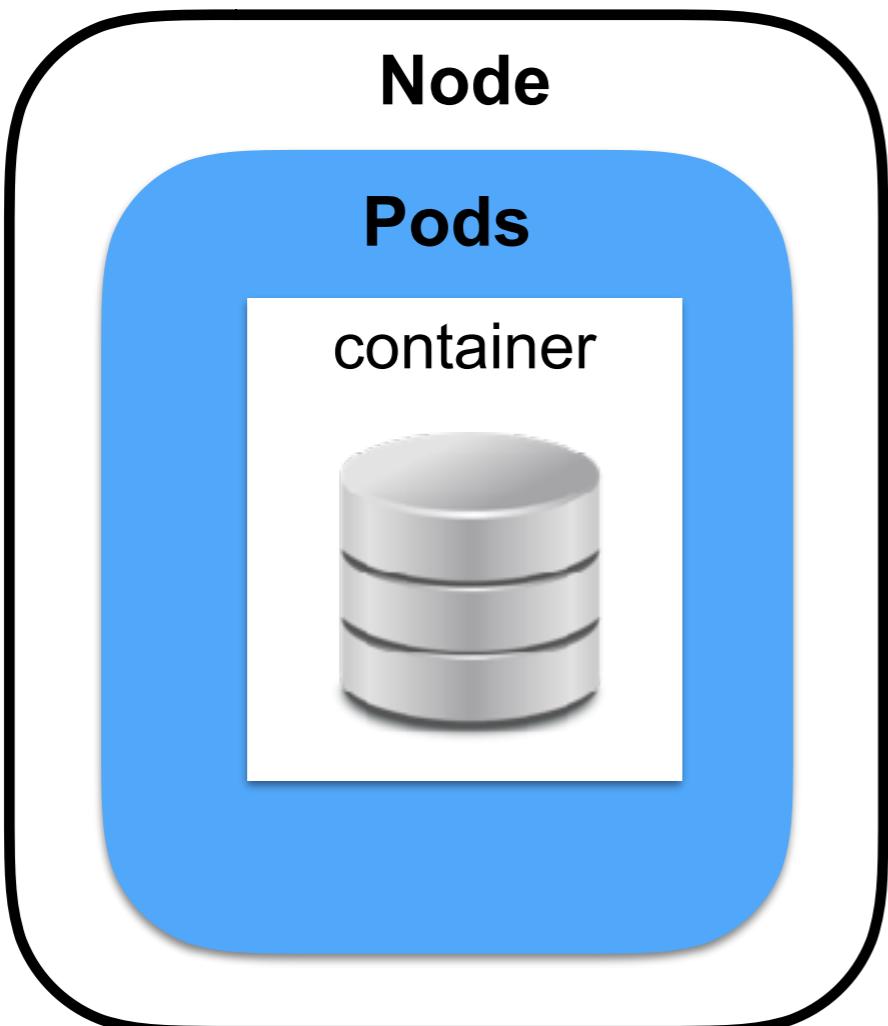


Volume

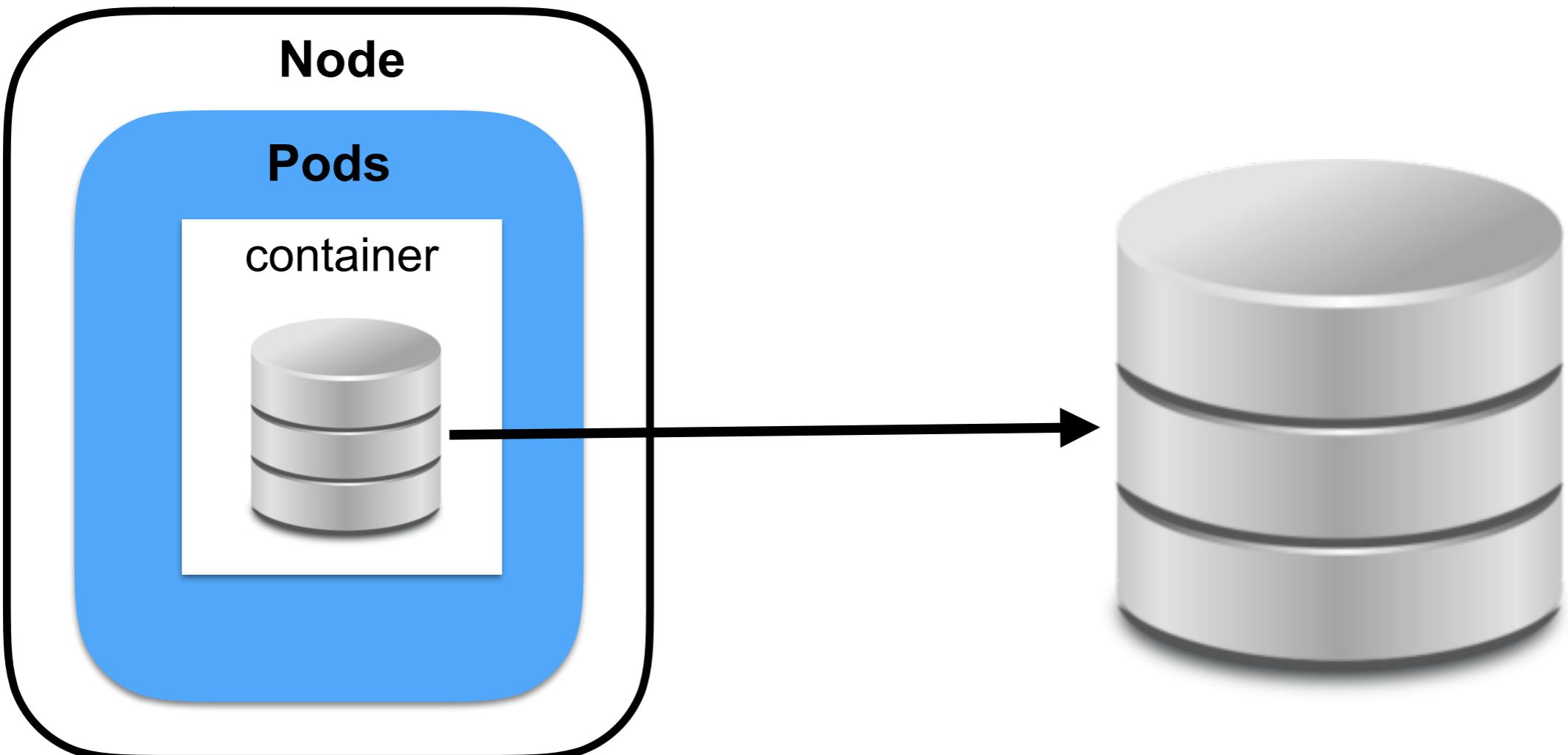
<https://kubernetes.io/docs/concepts/storage/volumes/#resources>



Volume



Volume



Volume

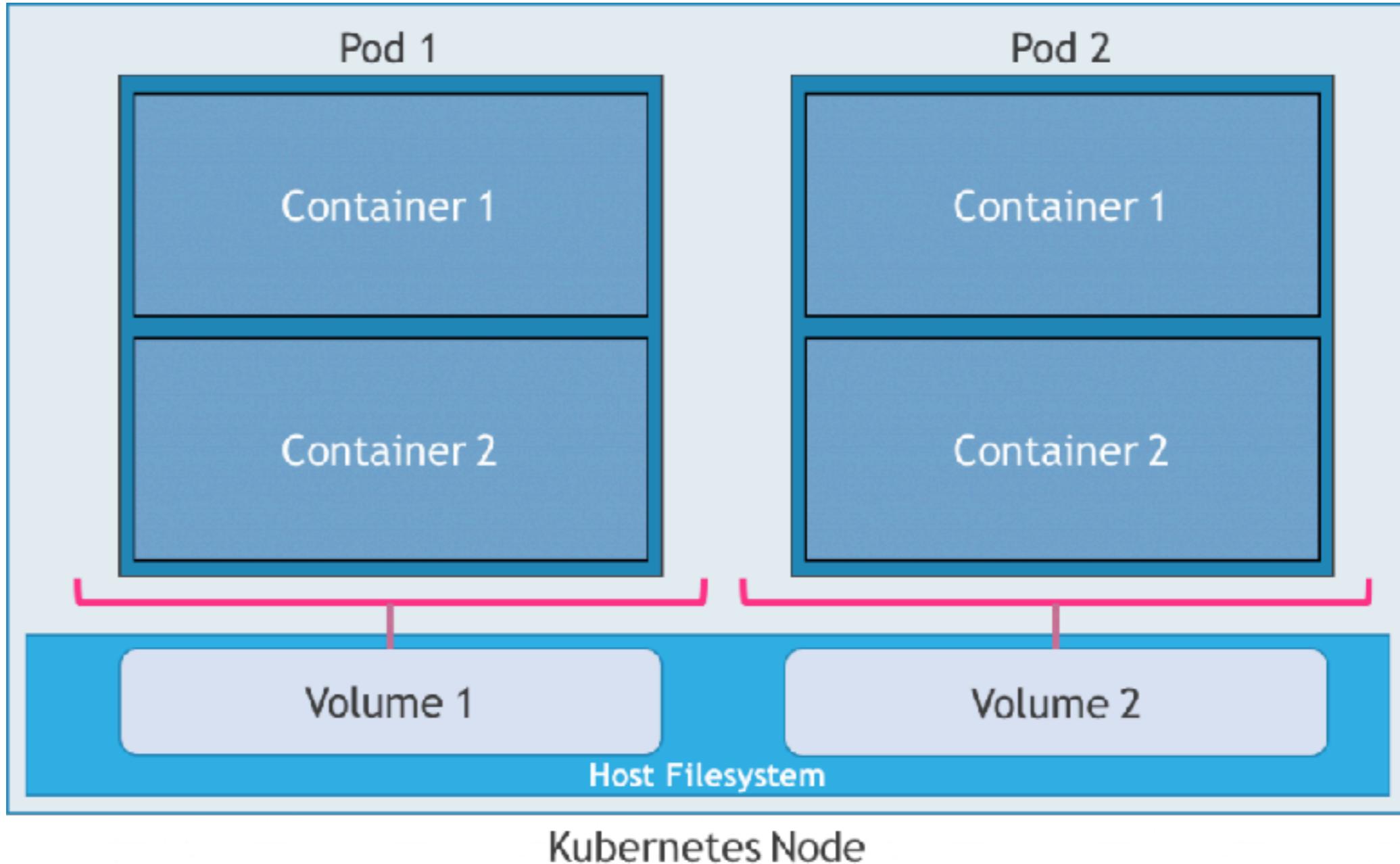
Containers in the same Pods share storage
(EmptyDir)

- 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



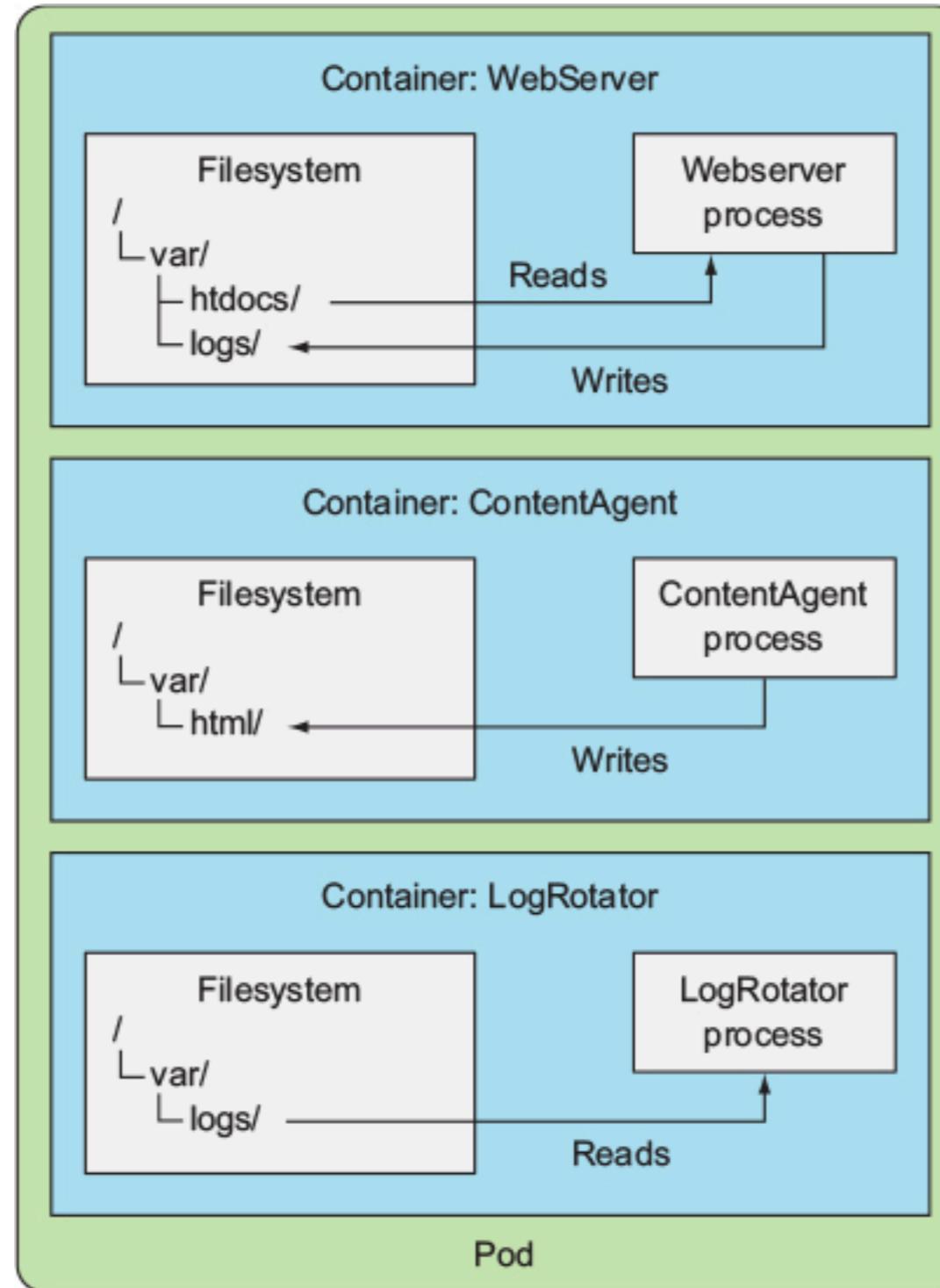
Containers shared volume



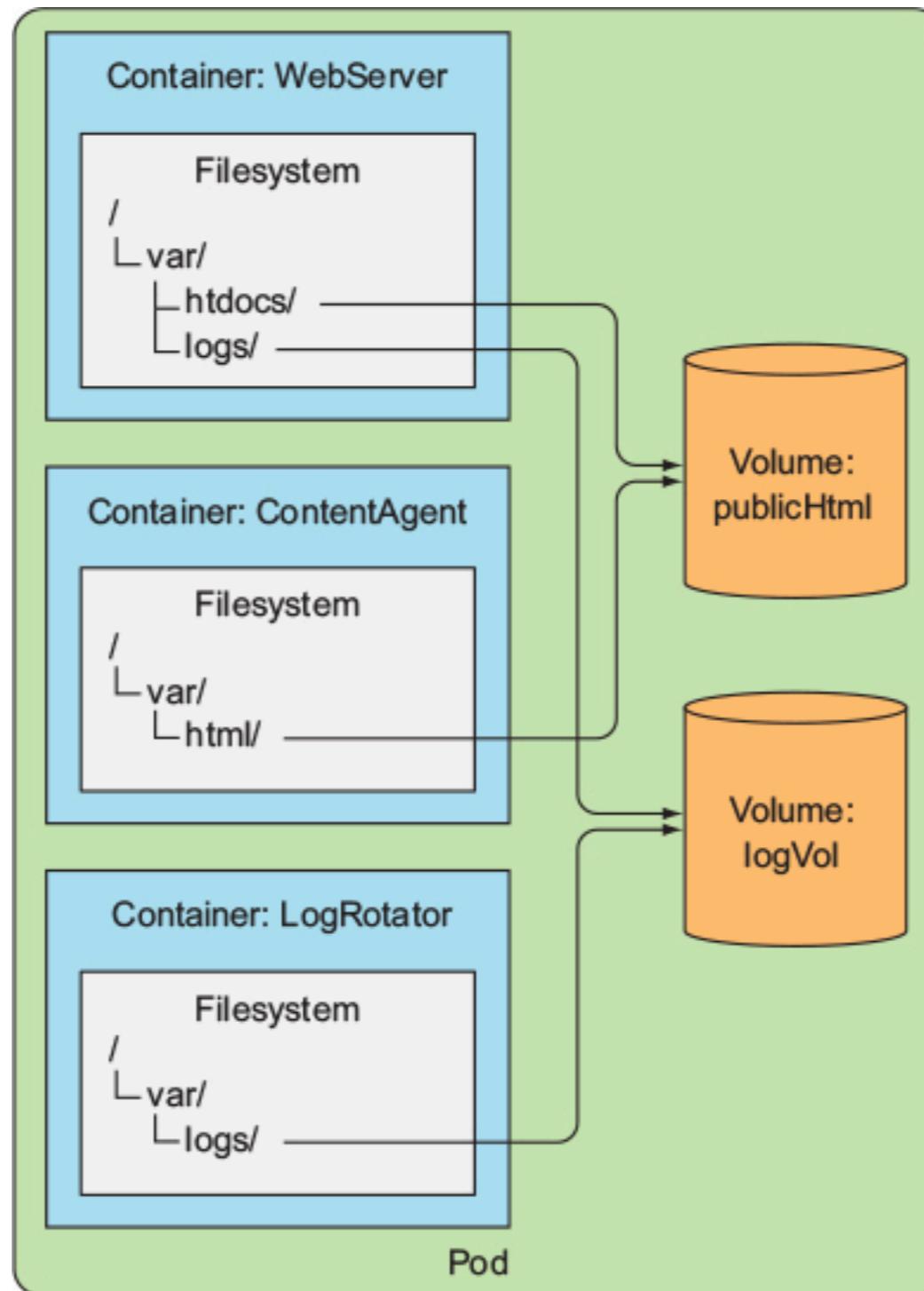
<https://thenewstack.io/strategies-running-stateful-applications-kubernetes-volumes/>



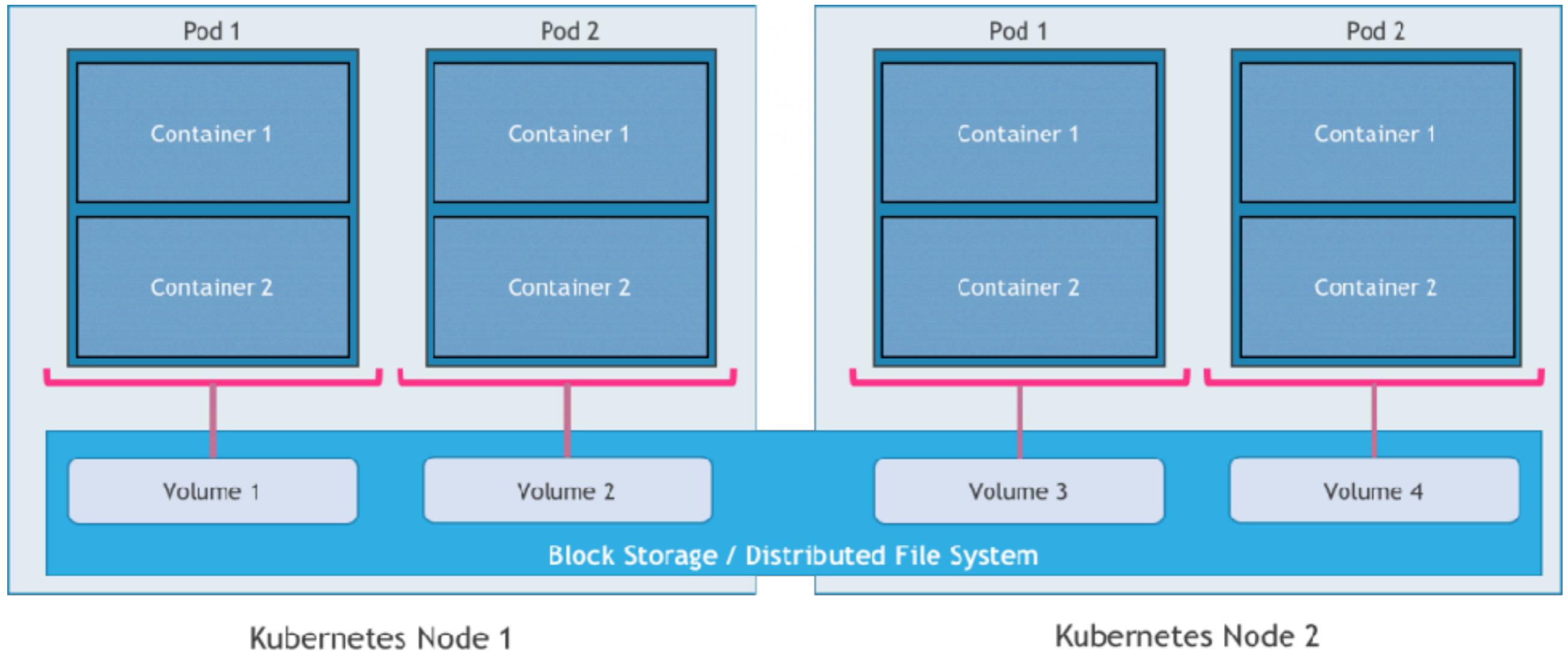
Without shared storage



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

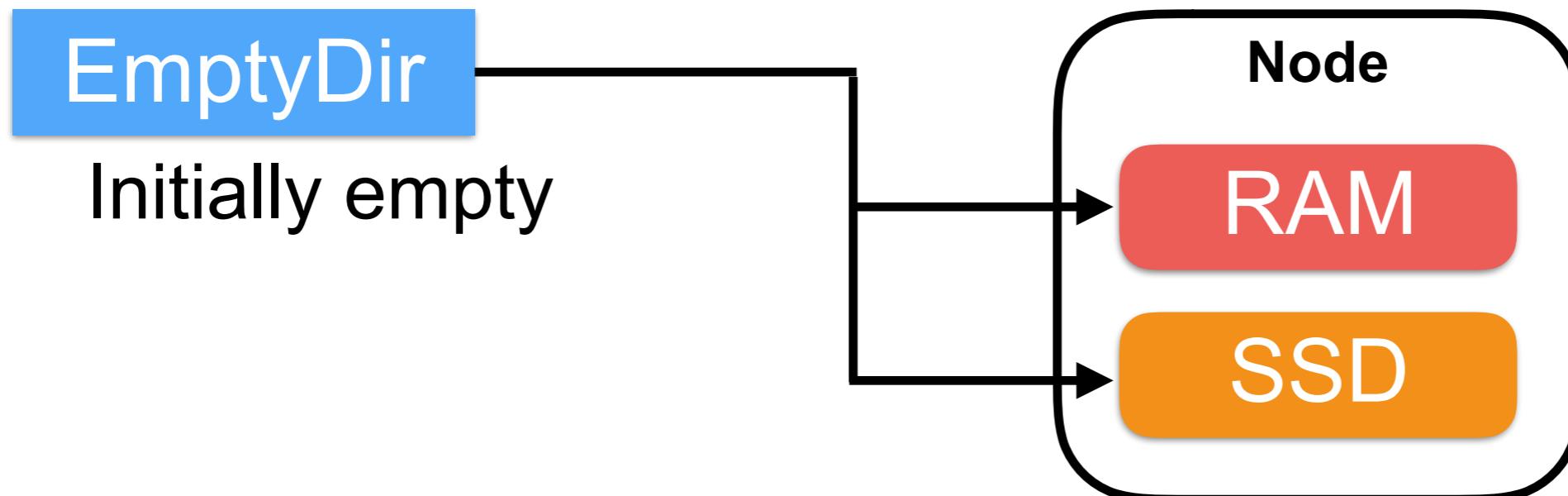
NFS

Cloud Volume

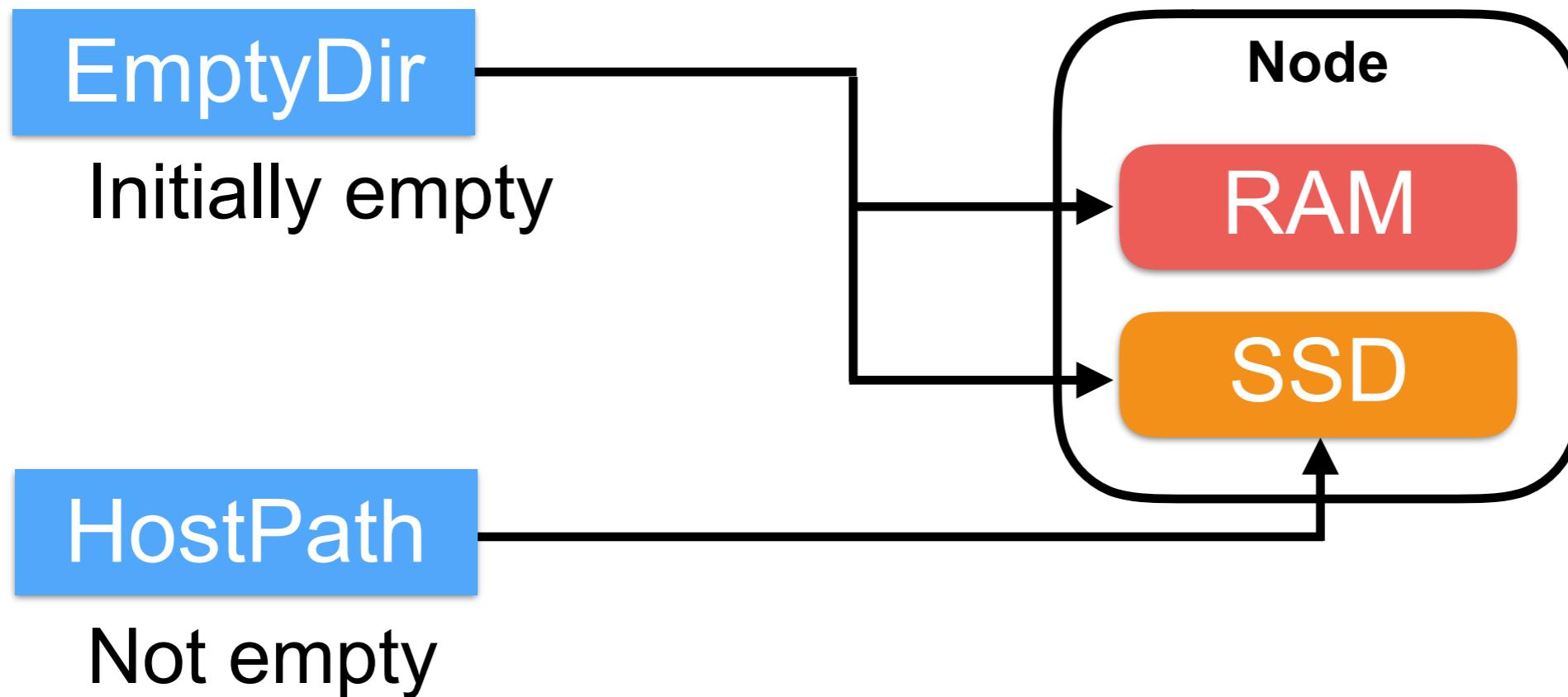
Persistent Volume Claim



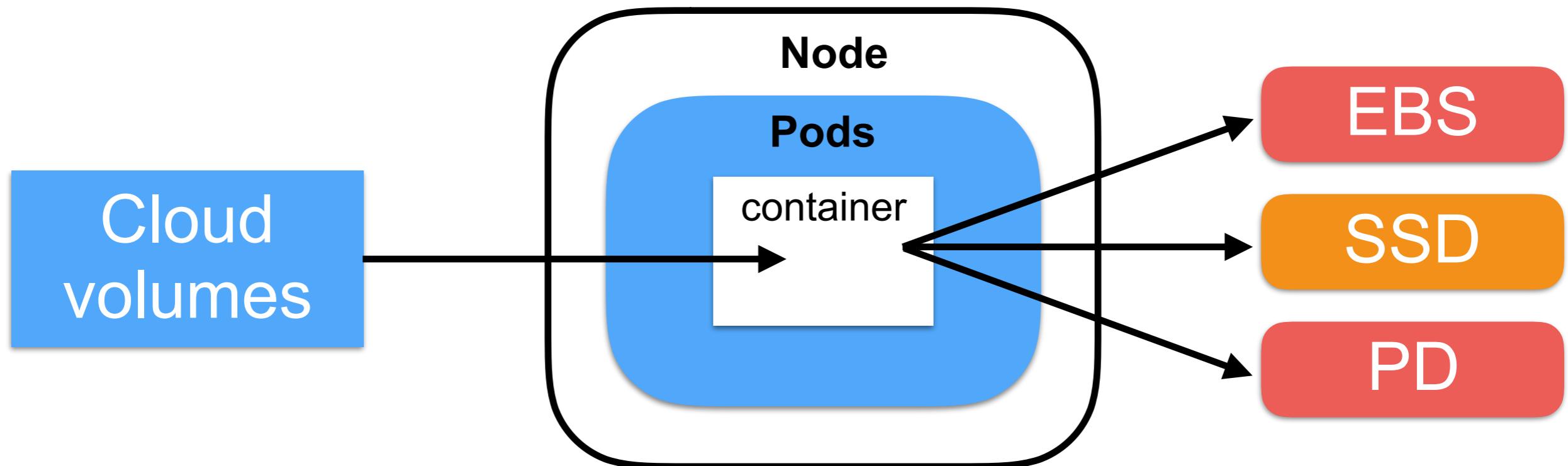
Kubernetes volumes



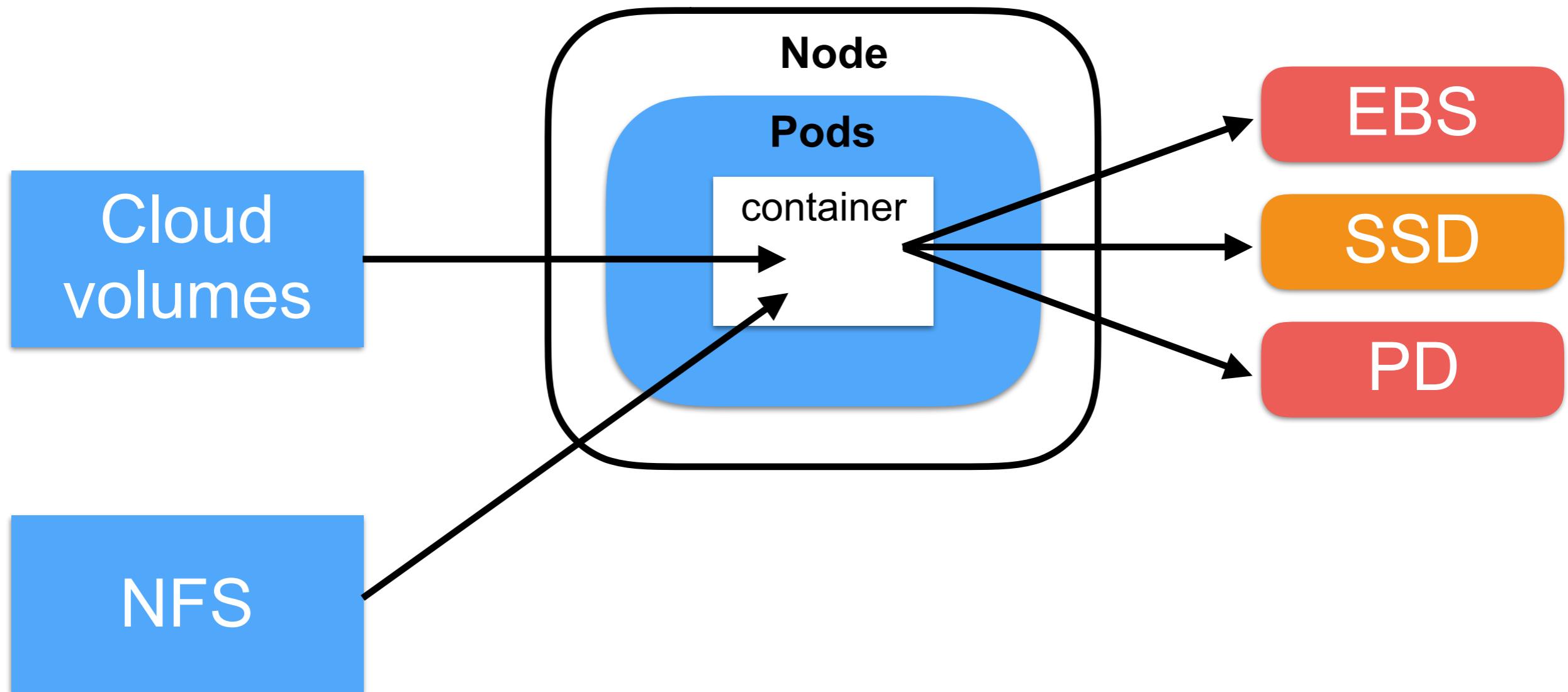
Kubernetes volumes



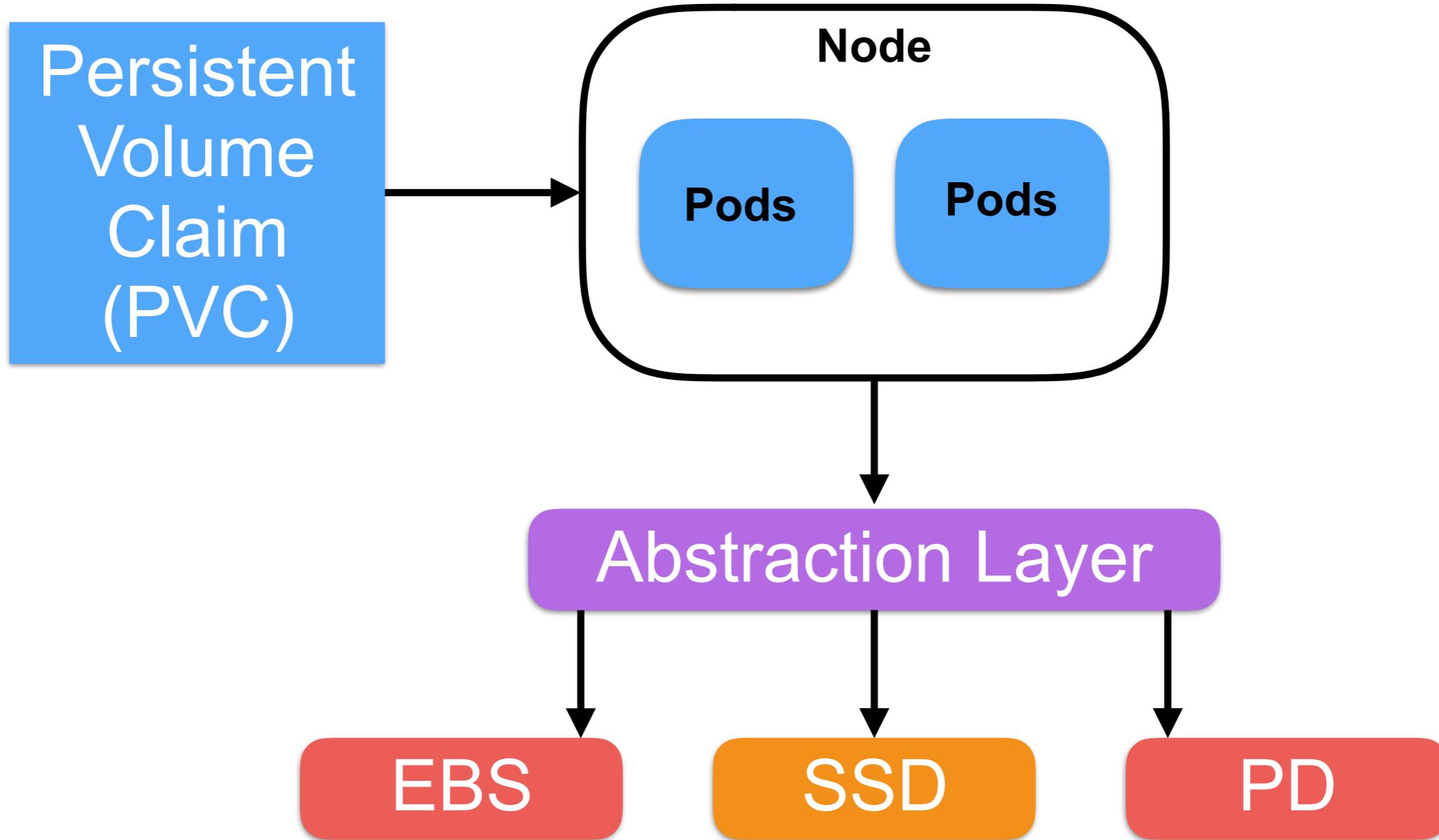
Kubernetes volumes



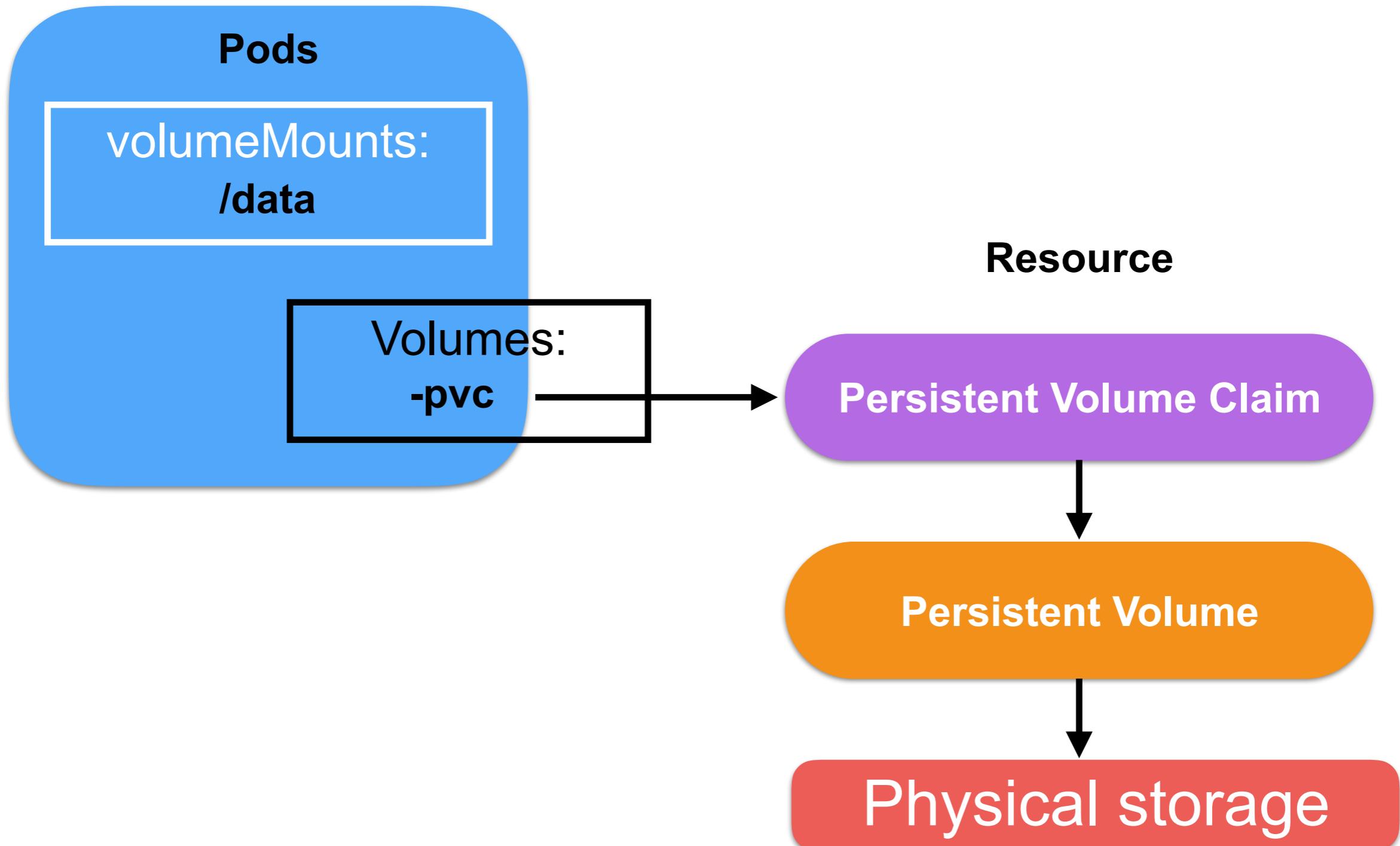
Kubernetes volumes



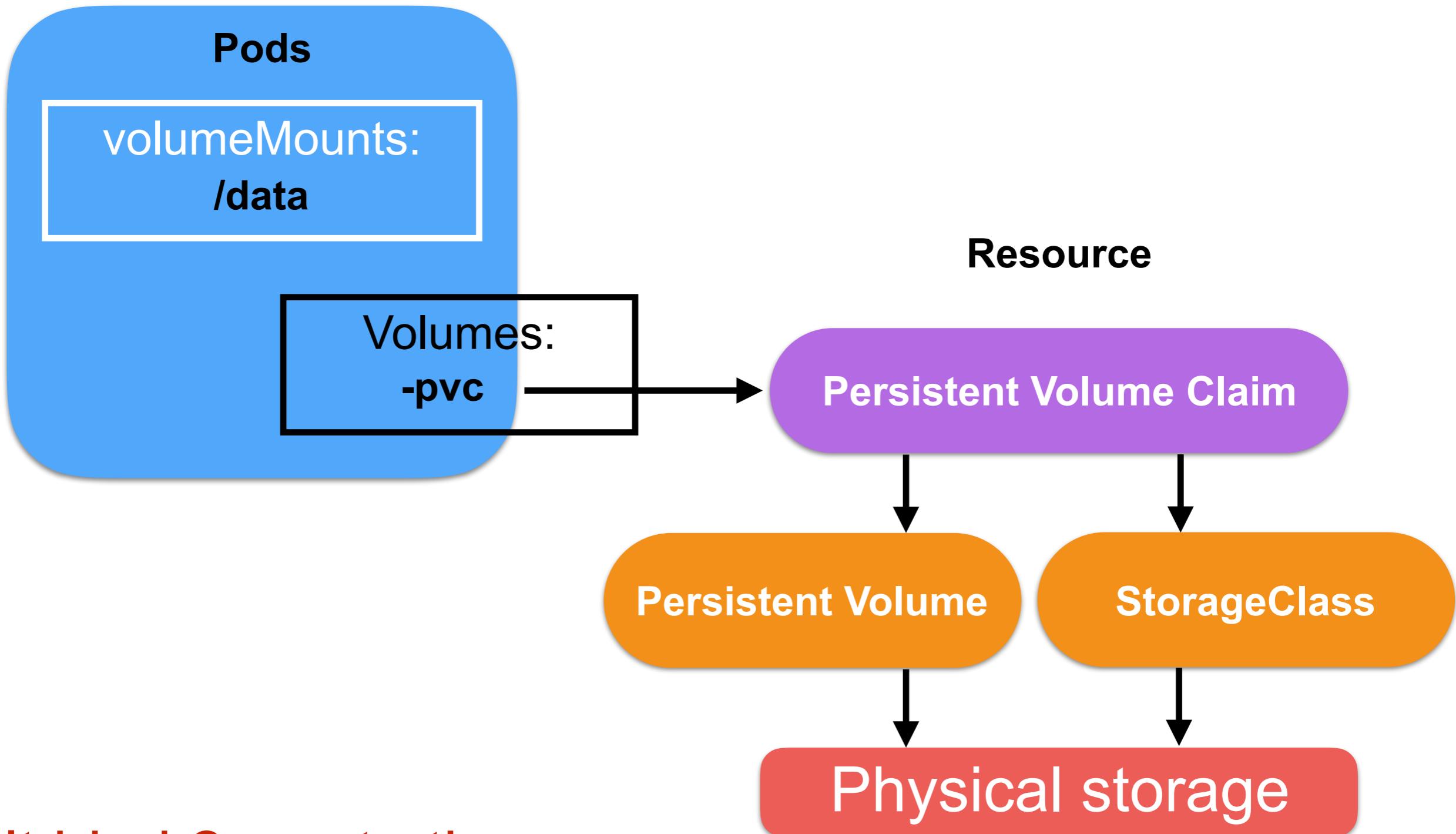
Kubernetes volumes



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
```

