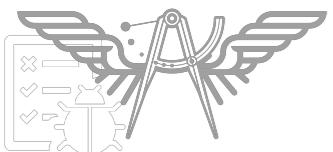


Kubernetes

In Practice





Somkiat Puisungnoen

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

Intro

Software Craftsmanship

Software Practitioner at สยามชัมนาณกิจ พ.ศ. 2556

Agile Practitioner and Technical at SPRINT3r

Post Photo/Video Live Video Life Event

What's on your mind?

Public Post

Somkiat Puisungnoen 15 mins · Bangkok · ⚙️

Java and Bigdata



Page

Messages

Notifications 3

Insights

Publishing Tools

Settings

Help ▾



somkiat.cc

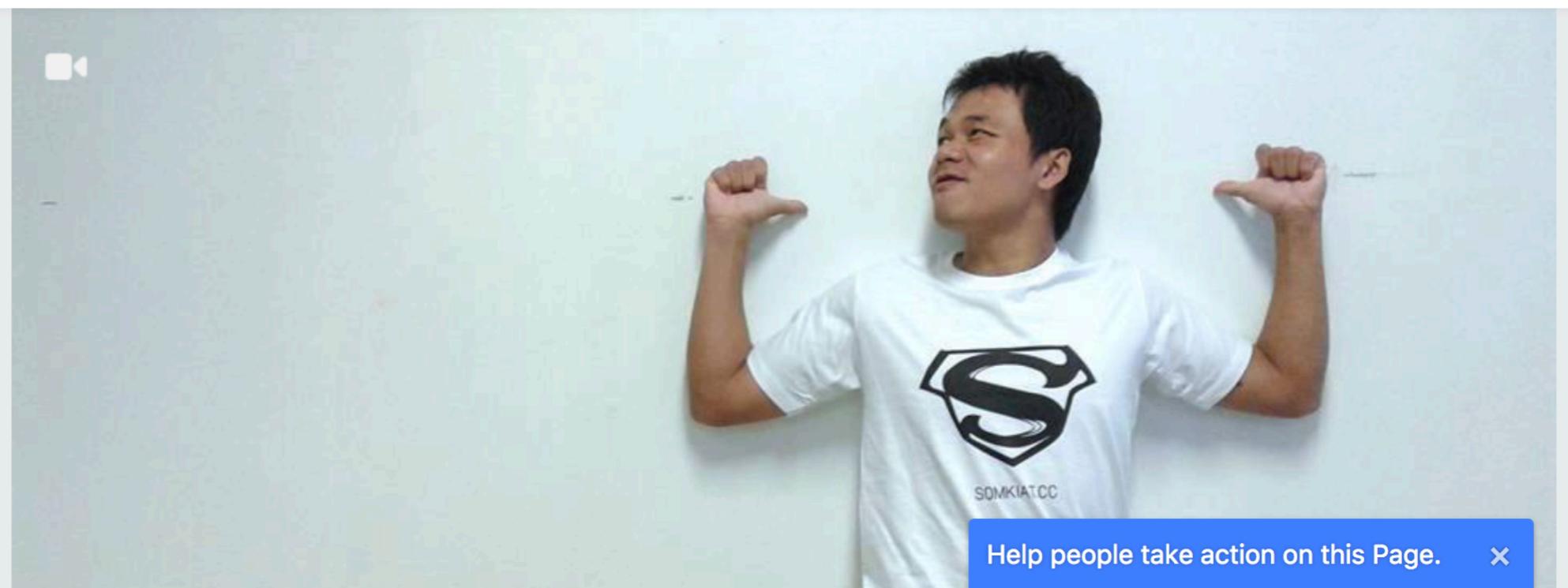
@somkiat.cc

Home

Posts

Videos

Photos



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. Working with Persistence storage

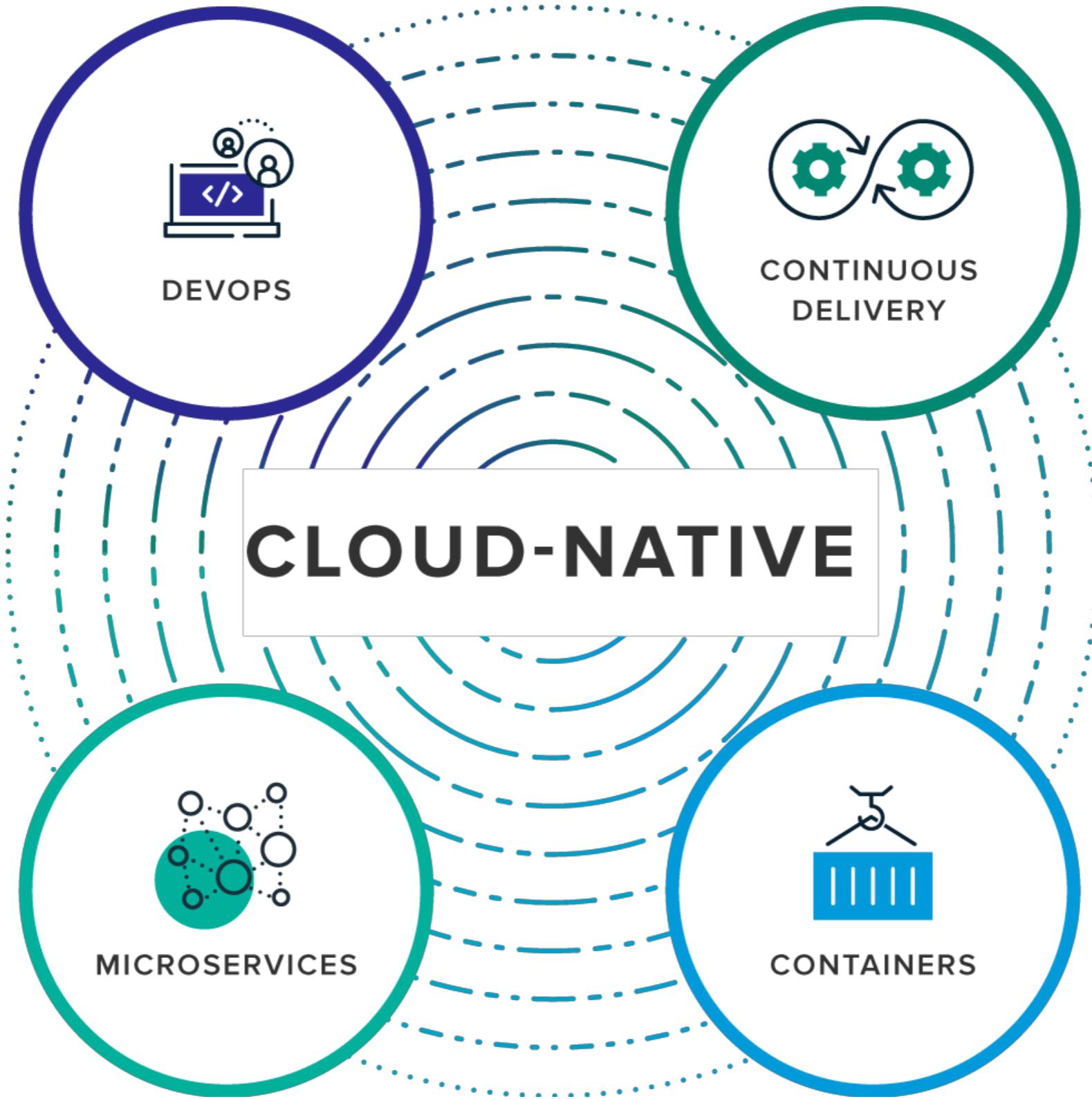


<https://github.com/up1/course-kubernetes-in-practice>



Cloud Native Application





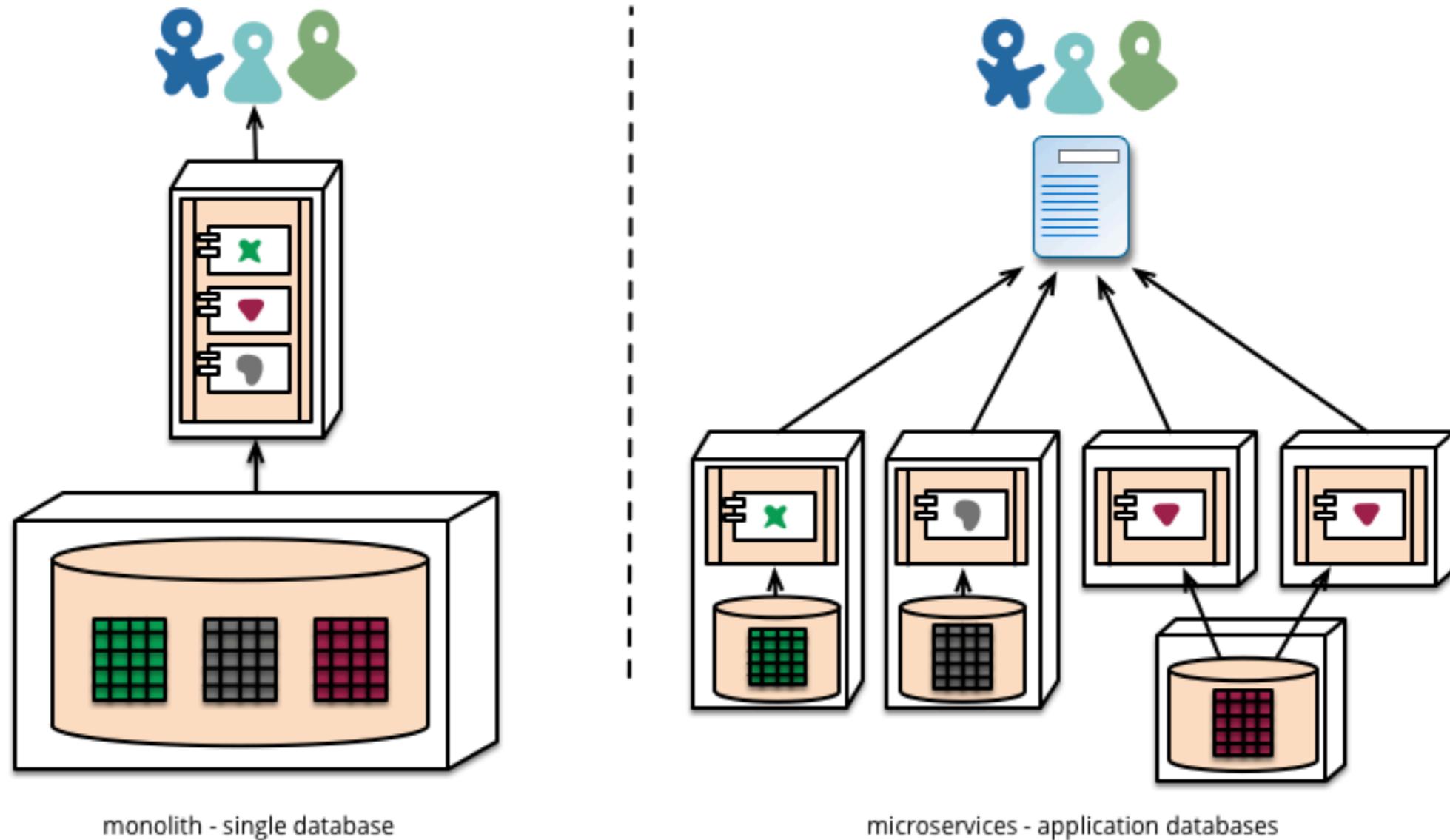
<https://pivotal.io/cloud-native>



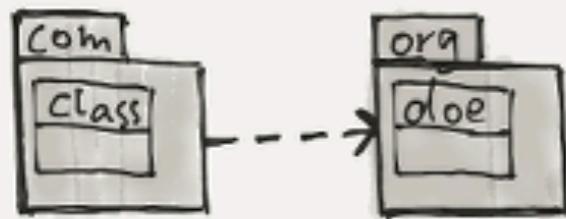
Evolution of Architecture



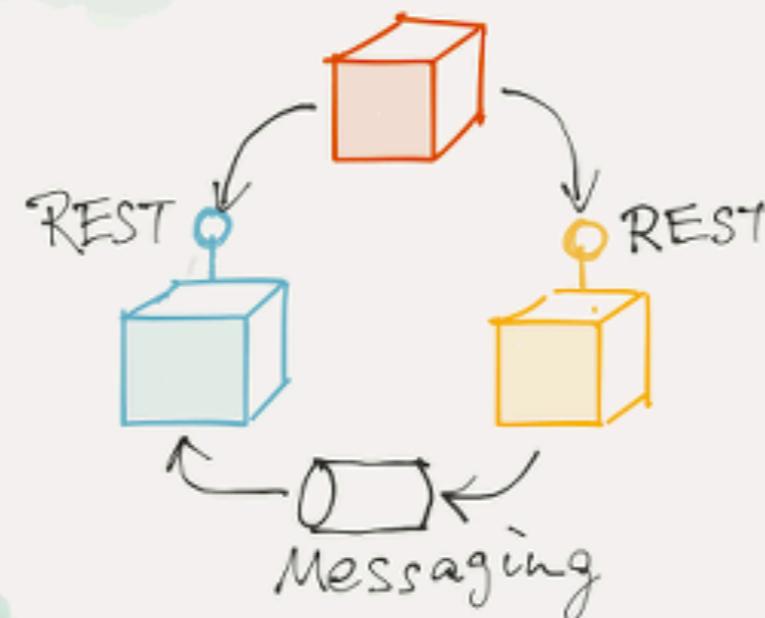
Microservices



Architecture



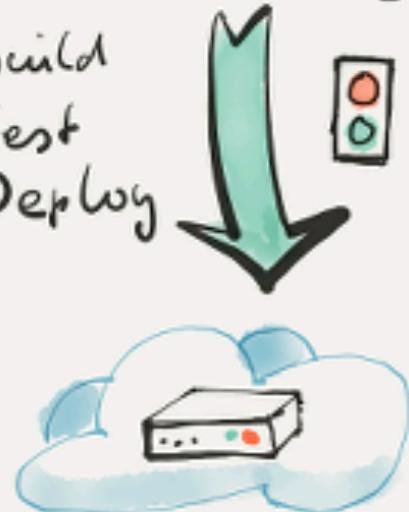
Microservices



Deployment

Continuous Delivery

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



Infrastructure

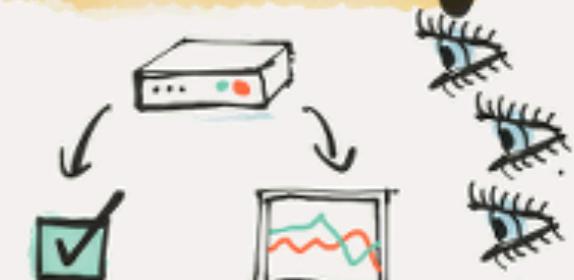


People & Teams



Communication
Collaboration

Monitoring

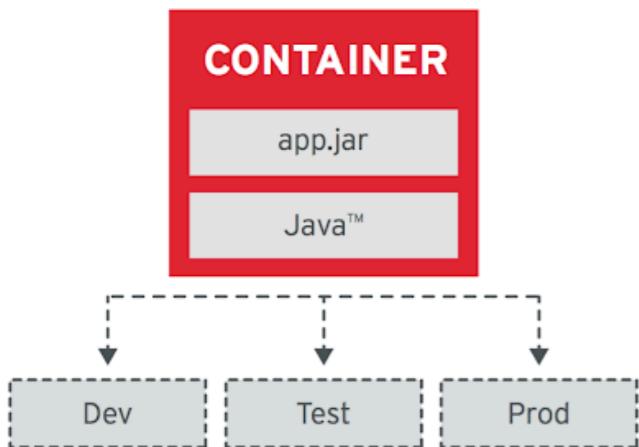


Features & Technology



Container design principles

Image Immutability Principle



High Observability Principle



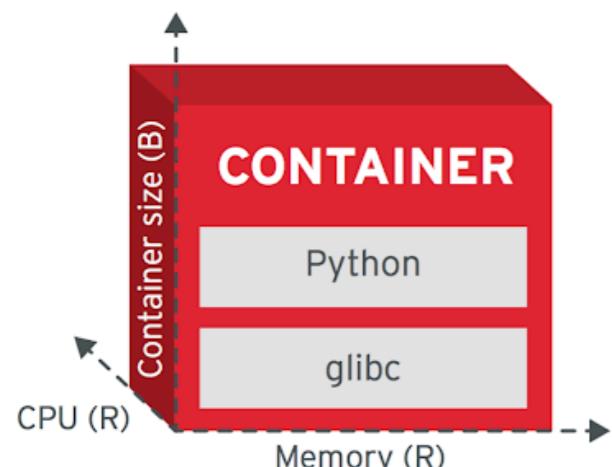
Process Disposability Principle



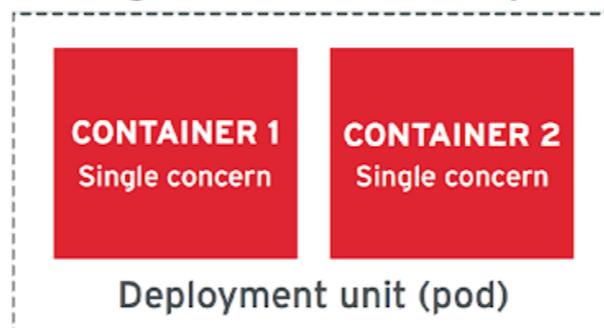
Lifecycle Conformance Principle



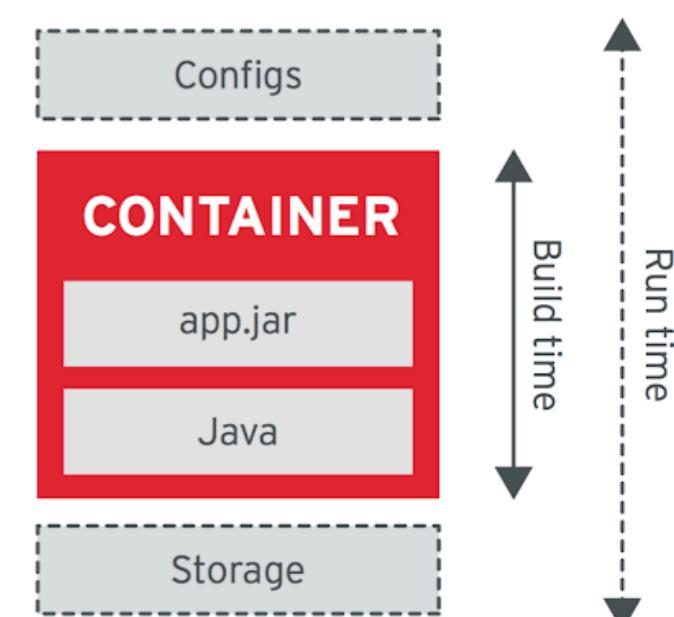
Runtime Confinement Principle



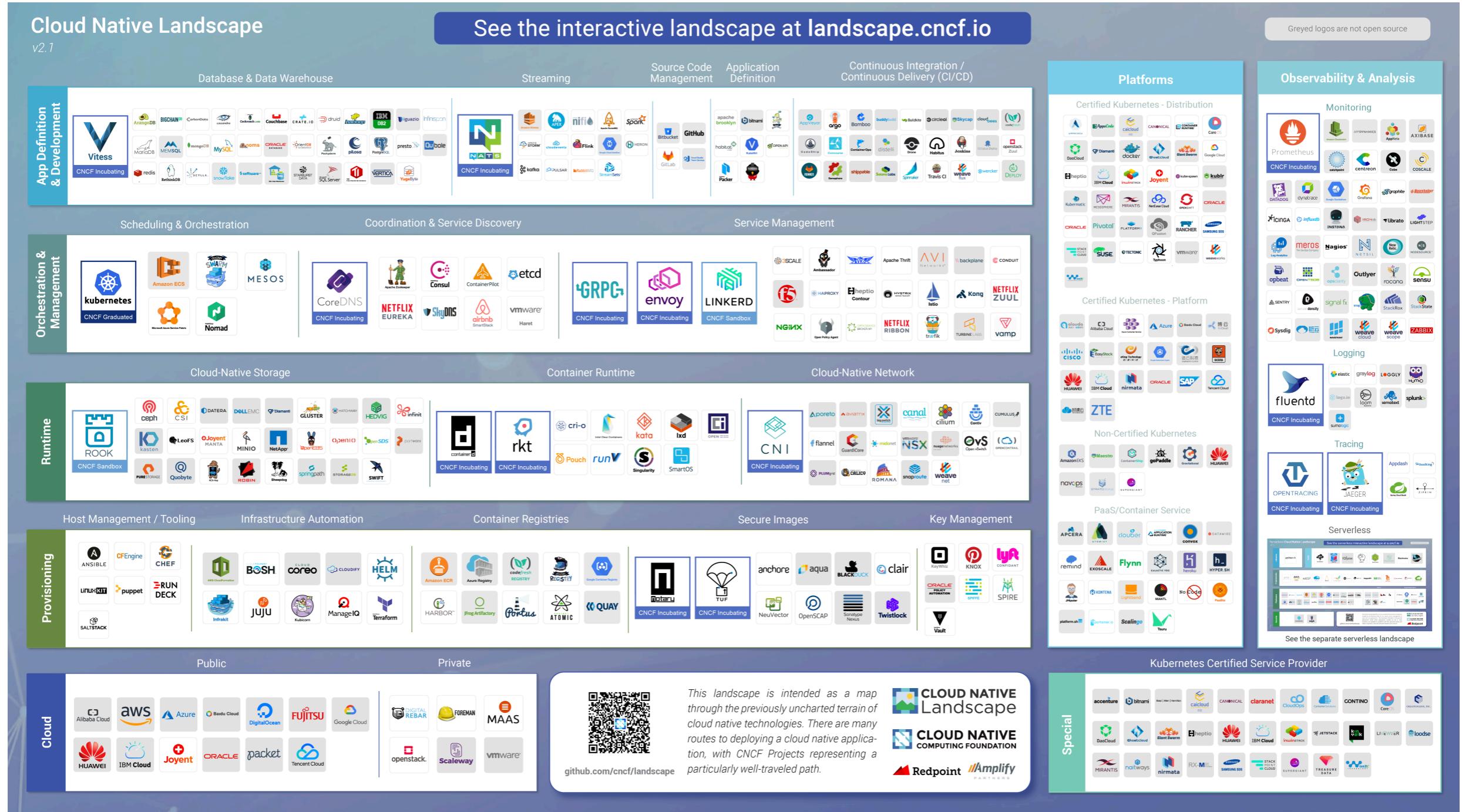
Single Concern Principle



Self-Containment Principle



Cloud native landscape



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



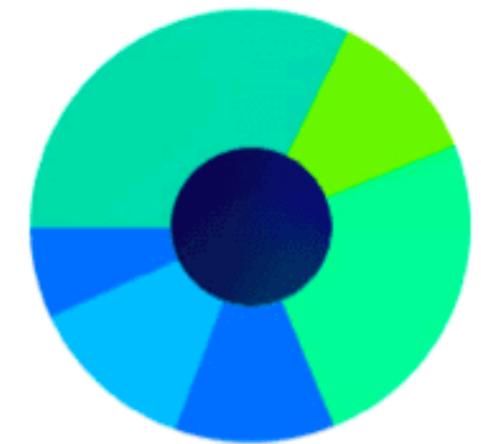
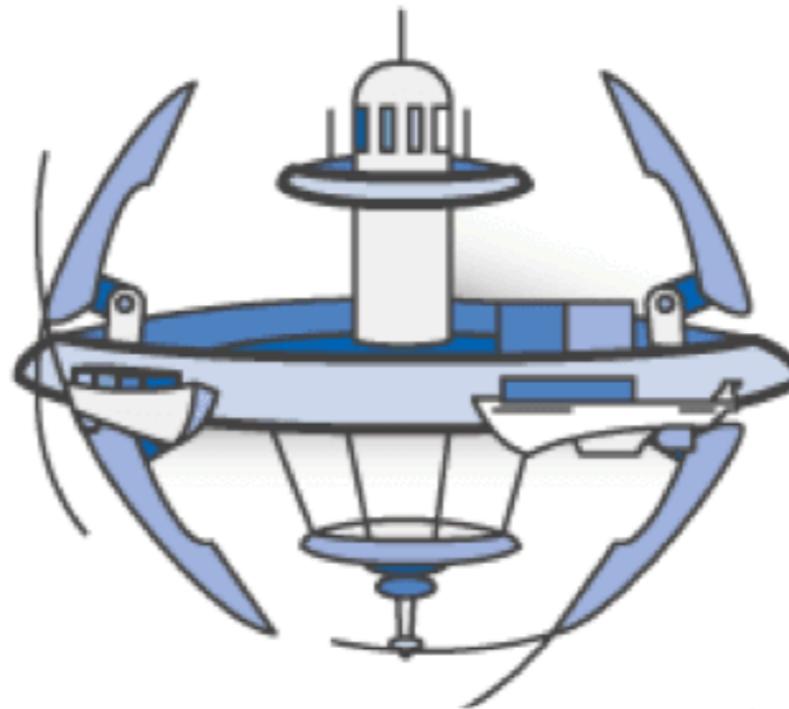
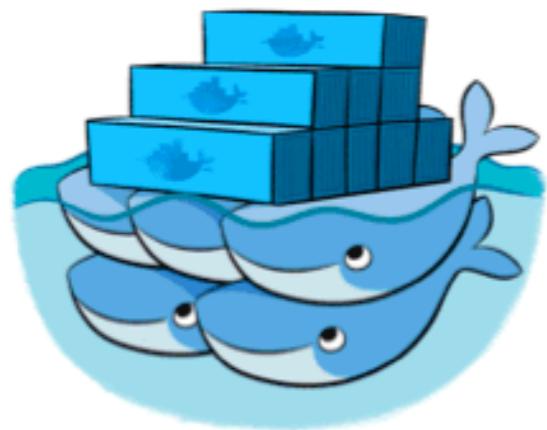
Kubernetes in practice

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





MESOS



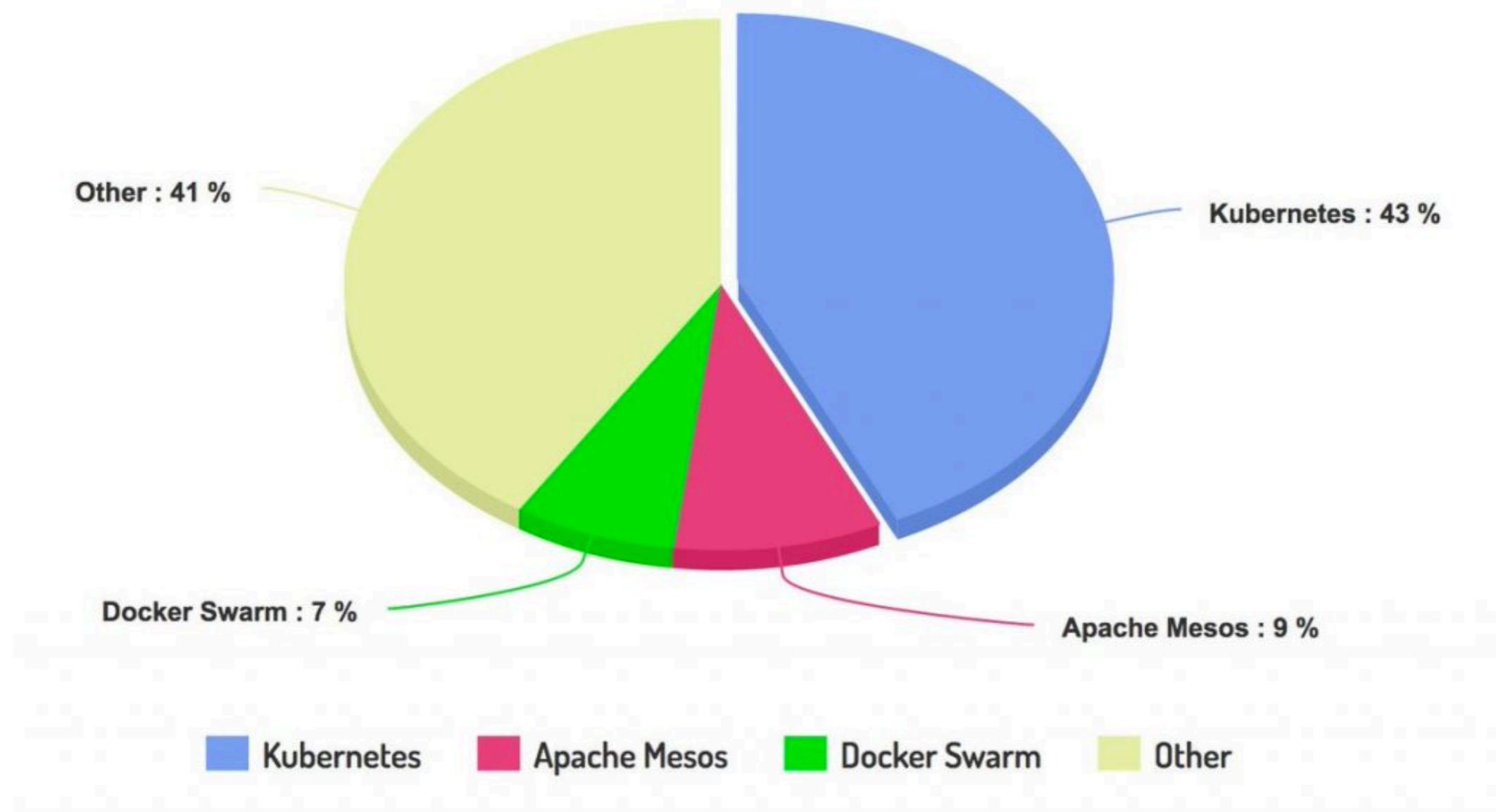
Marathon



kubernetes
by Google



Container Orchestrators in Sysdig's 2017 Docker Usage Report



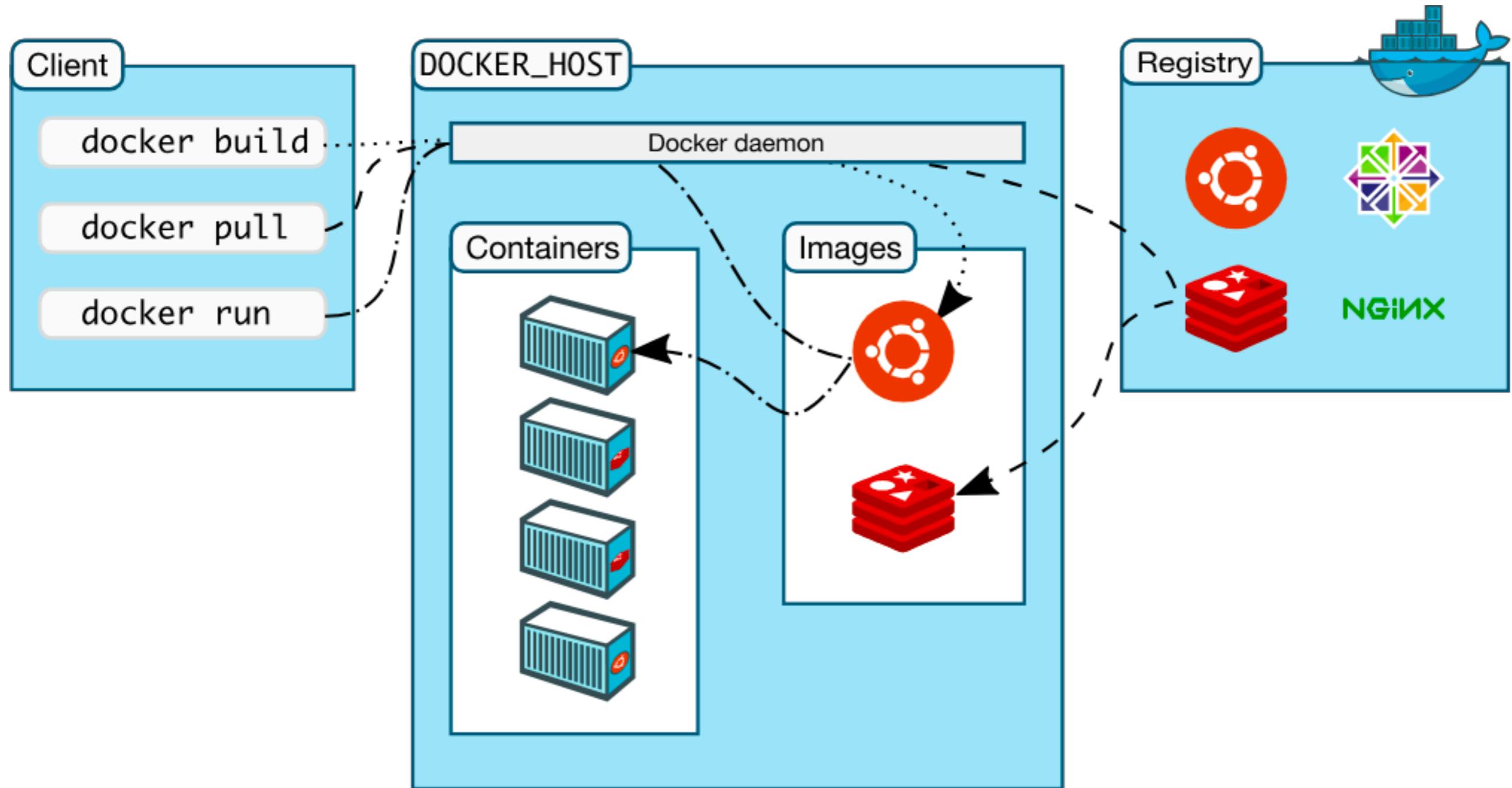
<https://sysdig.com/blog/sysdig-docker-usage-report-2017/>



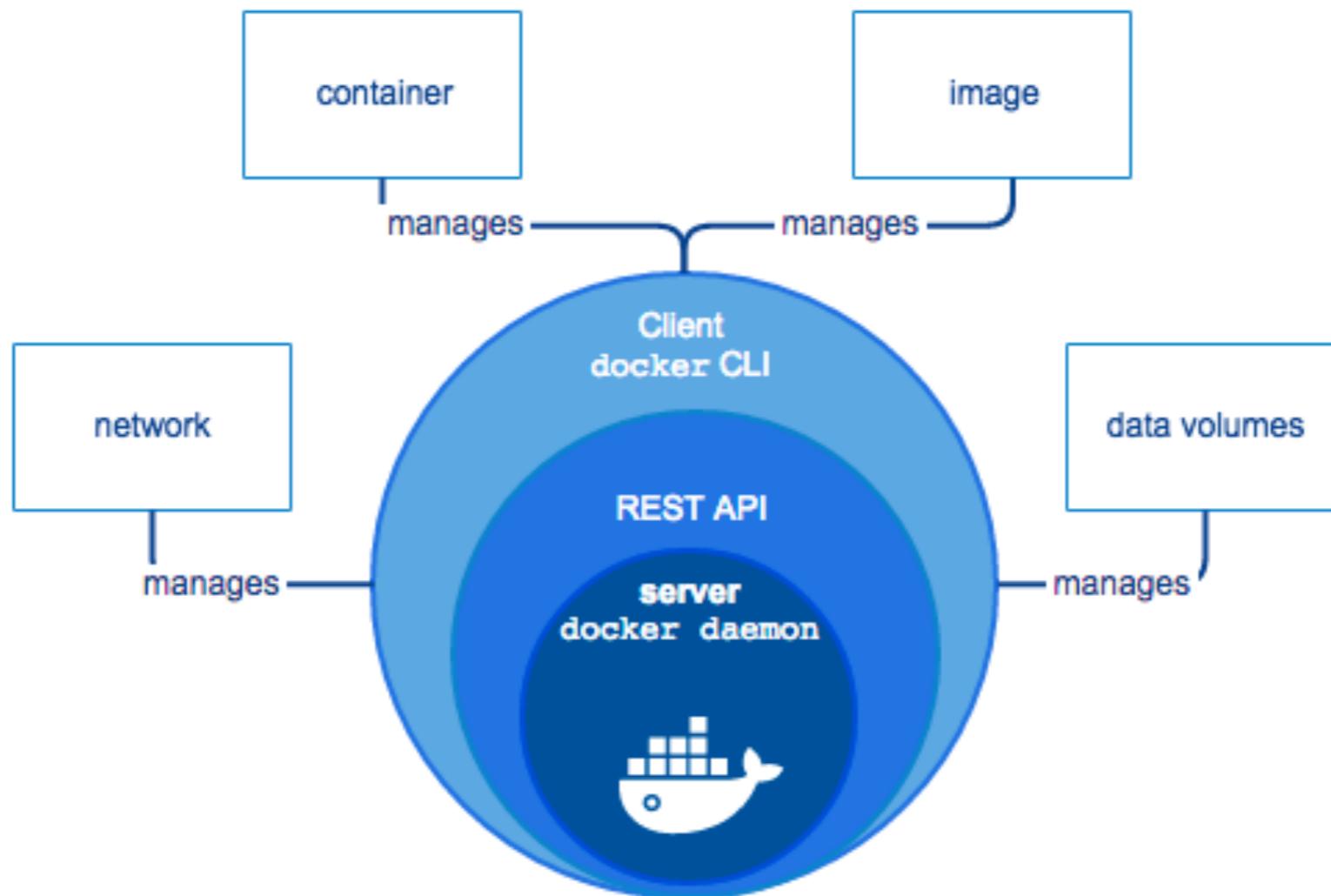
Recap Docker



How docker works ?



Docker Engine

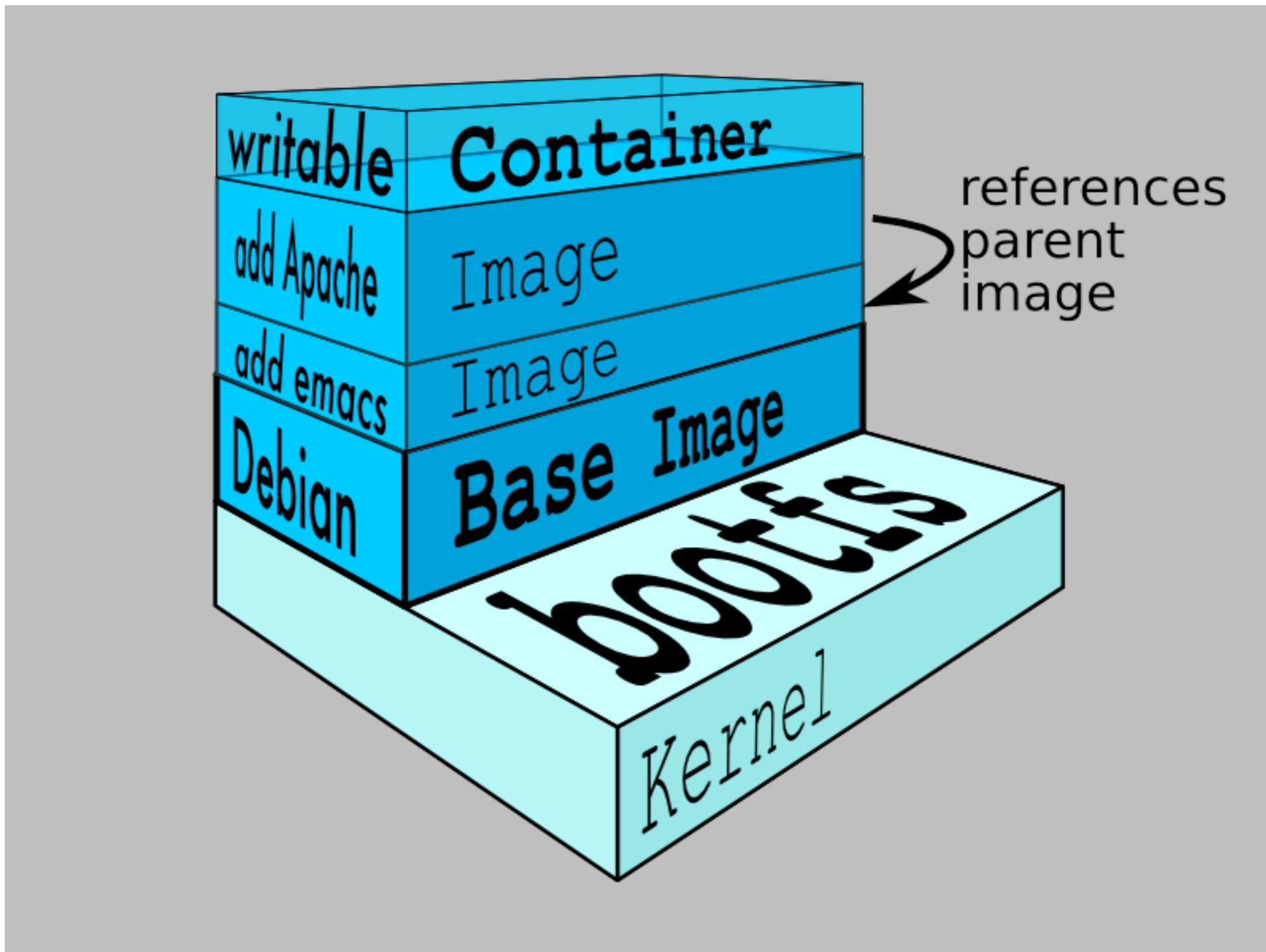


Concepts of Docker

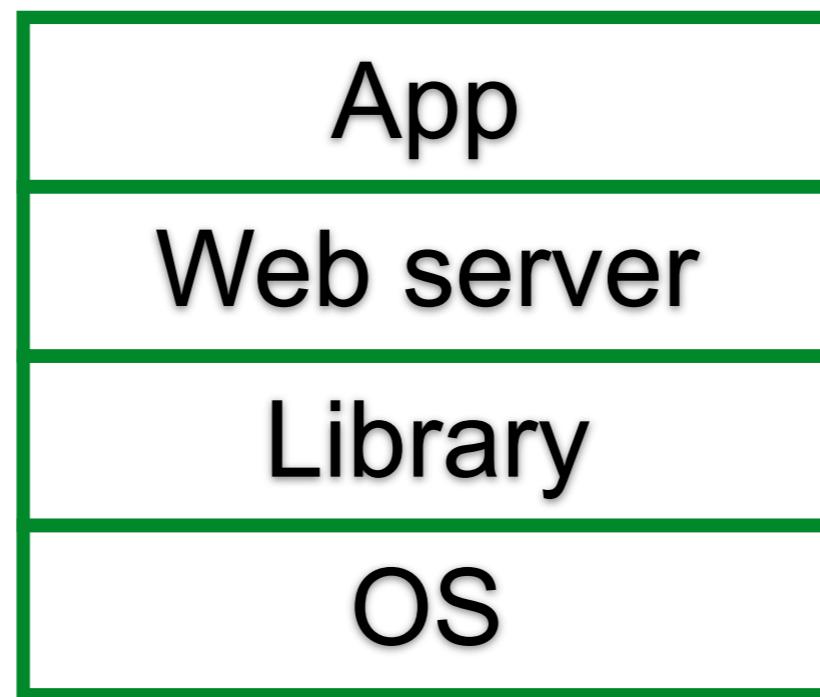
Image
Layer
Container
Docker registry



Docker Image and Layer



Example of Java



Example of Java



Essential for Docker

Command line (CLI)

Dockerfile

Docker compose



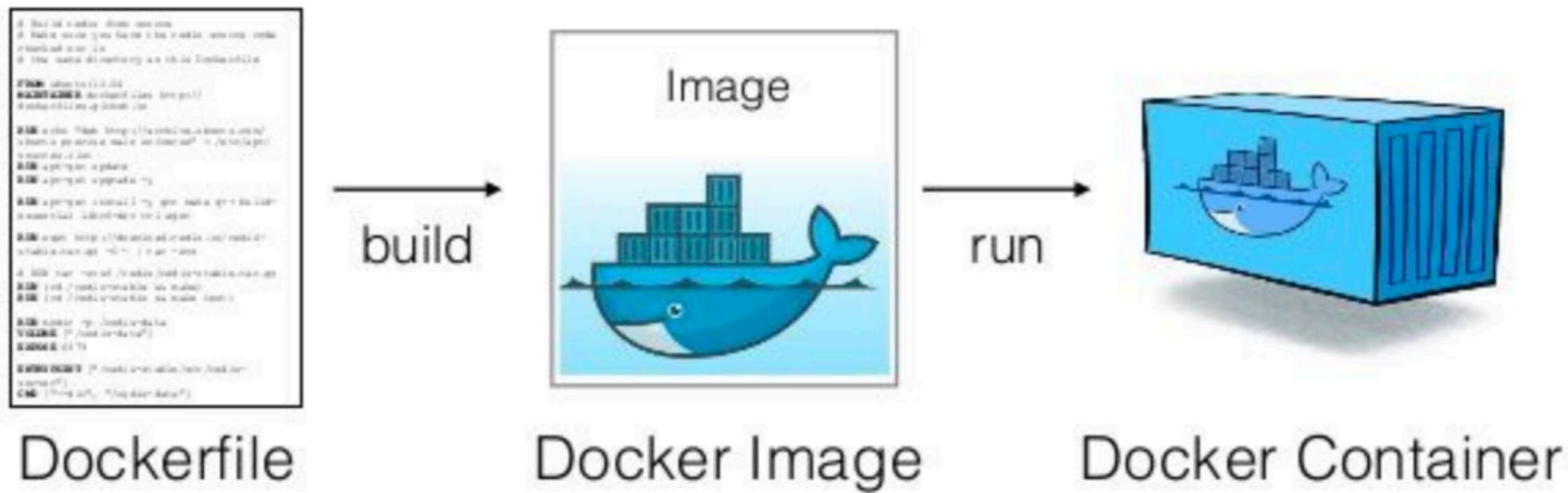
Using docker via command line

\$docker image
\$docker container
\$docker network
\$docker volume

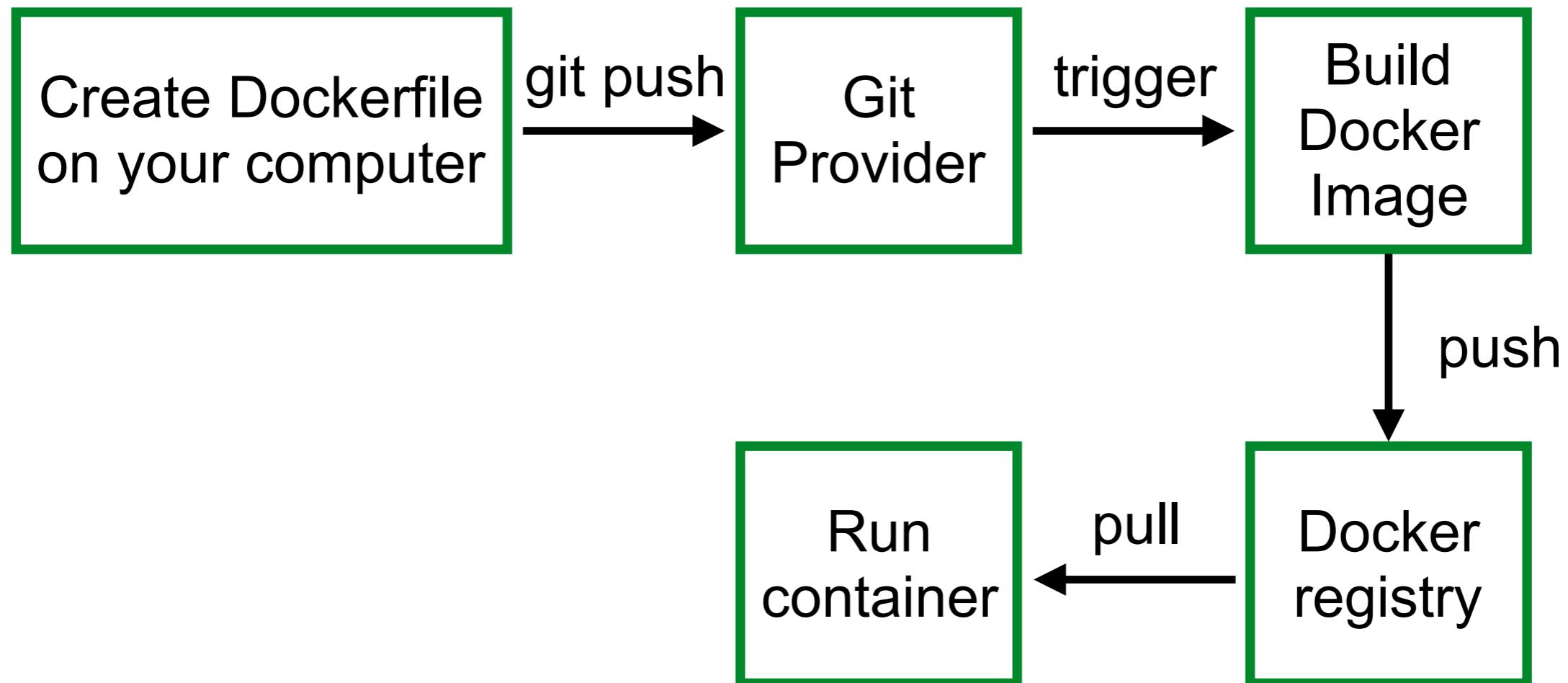


Dockerfile

Recipe to build a Docker image



Automated process

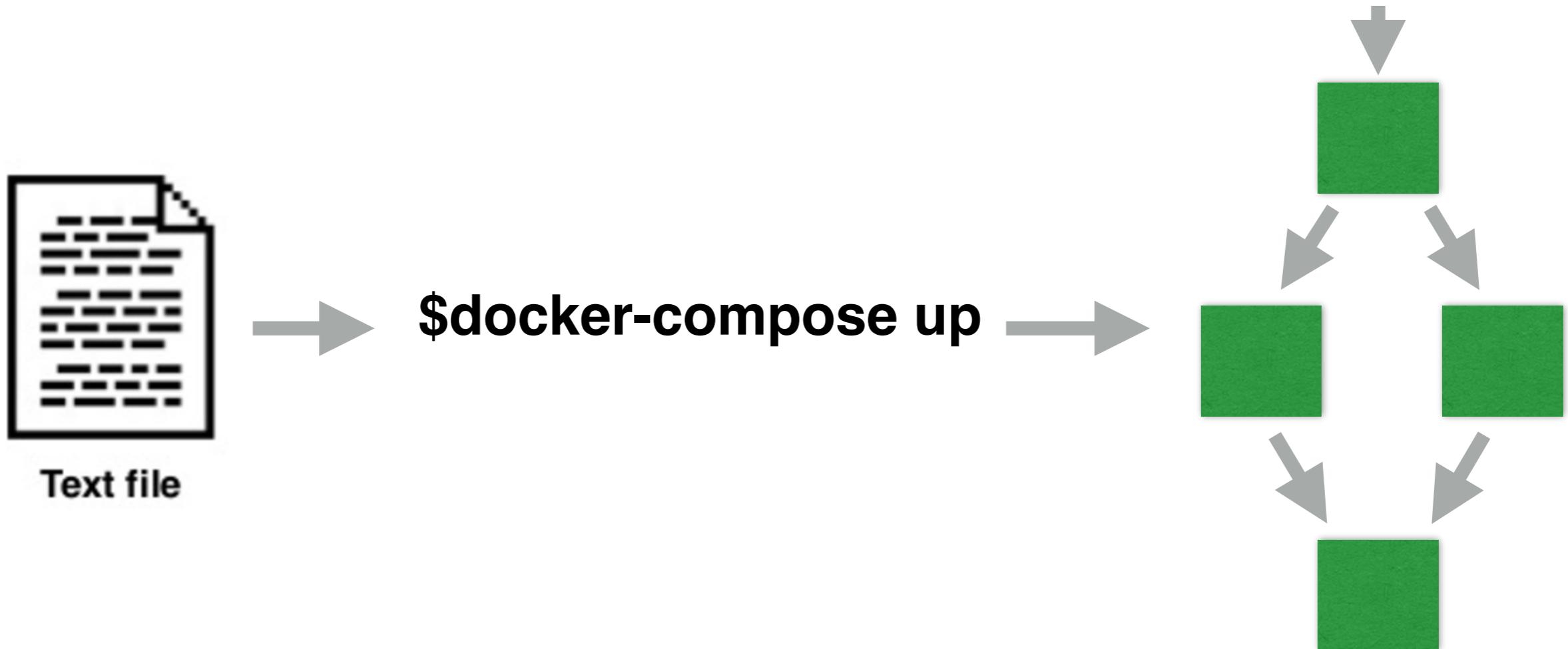


Workshop with Dockerfile



Docker compose

Running your app in one command-line



<https://docs.docker.com/compose/>



Docker compose

Create file docker-compose.yml

```
version: '3'  
services:  
  web:  
    container_name: web  
    build: .  
    ports:  
      - 80:5000  
    environment:  
      - REDIS_HOST=redis  
  
  redis:  
    container_name: redis  
    image: redis  
    links:  
      - web
```



Using docker via command line

\$docker-compose build

\$docker-compose up

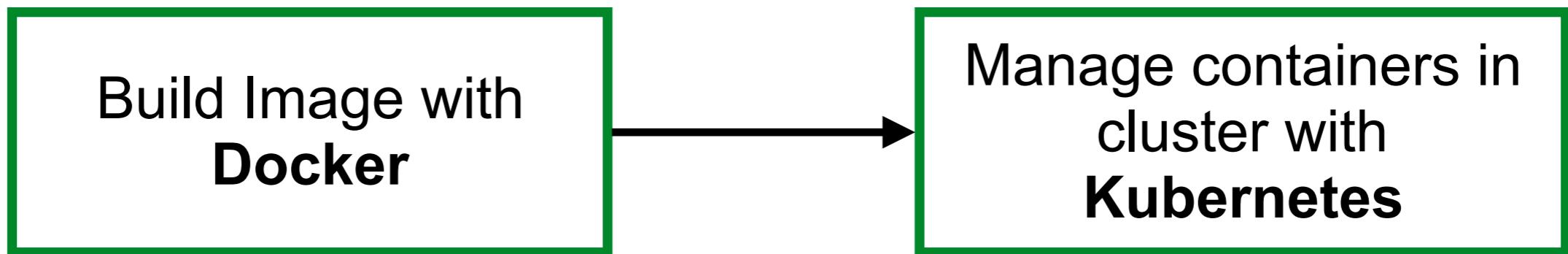
\$docker-compose down



Workshop with Docker compose



Docker and Kubernetes



Docker on production

Using label for image and container

Delete unused objects

Limit a container's resources (cpu, mem)

Logging

Monitoring



Docker on production

Using label for image and container

Delete unused objects

Limit a container's resources (cpu, mem)

Logging

Monitoring

Security !!!



DevSecOps

Using label for image and container
Delete unused objects

Limit a container's resources (cpu, mem)

Logging
Monitoring
Security !!!



DevSecOps

OWSAP Docker Top 10

OWASP Docker Top 10



The banner features the OWASP logo (a stylized orange flower inside a circle) and the text "OWASP" in large orange letters, with "Open Web Application Security Project" in smaller text below it. The background is blue with a subtle leaf pattern.

[hide]

- [1 About Docker Top 10](#)
- [2 Description](#)
- [3 Why not "Container Security"](#)

https://www.owasp.org/index.php/OWASP_Docker_Top_10

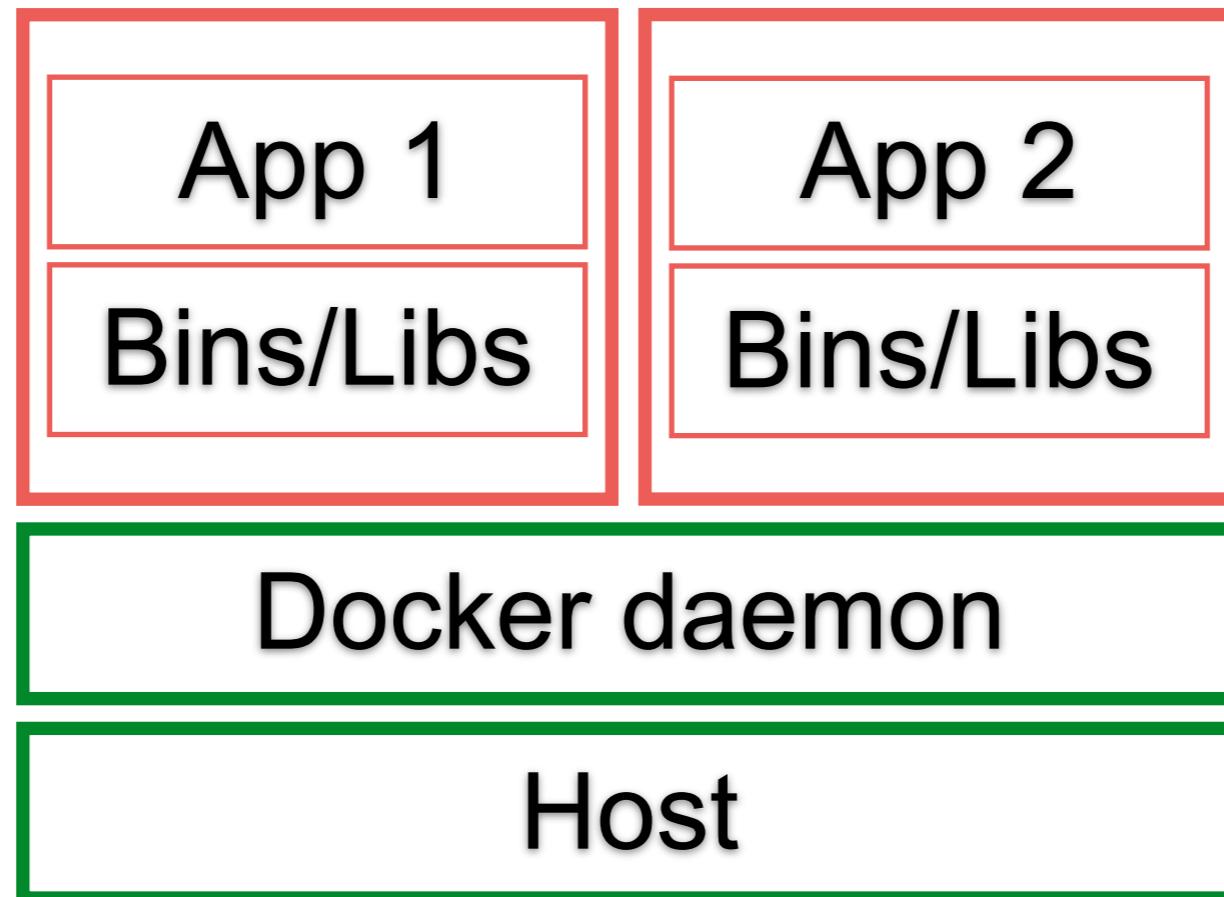


Different layers of security

Docker image building
Docker runtime
Docker engine
Docker host



Different layers of security



How to handle security ?

Kernel namespaces

Control groups (cgroups)

Ulimit

User namespaces

Apparmor and seccomp

<https://docs.docker.com/engine/security/>



Docker image recommendations

Do not run software as root

Do not trust community pages

Use minimal base image

Build always on a fresh base image



Docker image recommendations

Use specific version (don't latest)

Add HEALTHCHECK command

Do not store secrets to Dockerfile

Do not install unnecessary software



Docker host recommendations

Stay up-to-date
Use centralized logging



Docker runtime

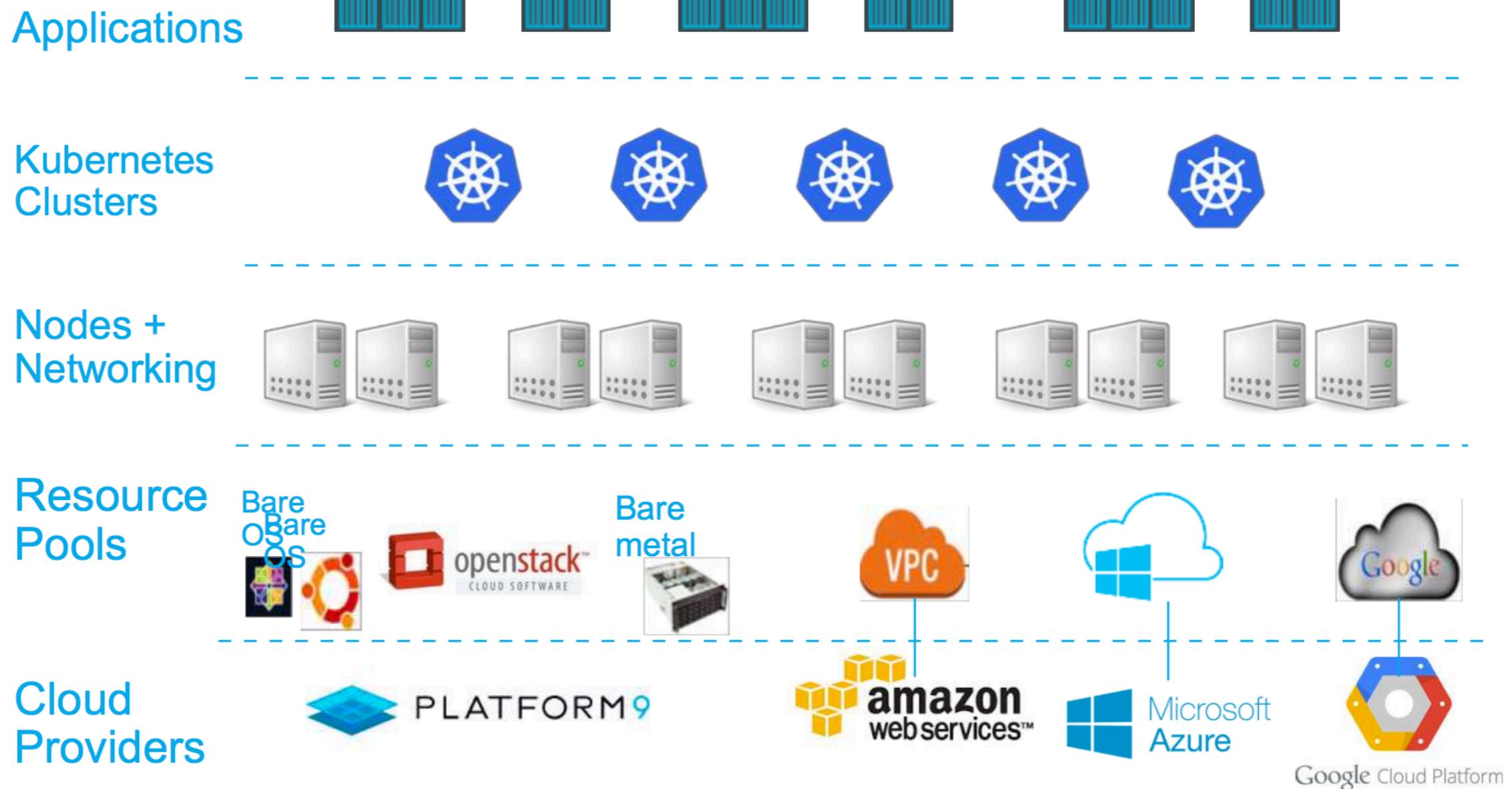
Use docker-compose instead of CLI
Add default flag



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

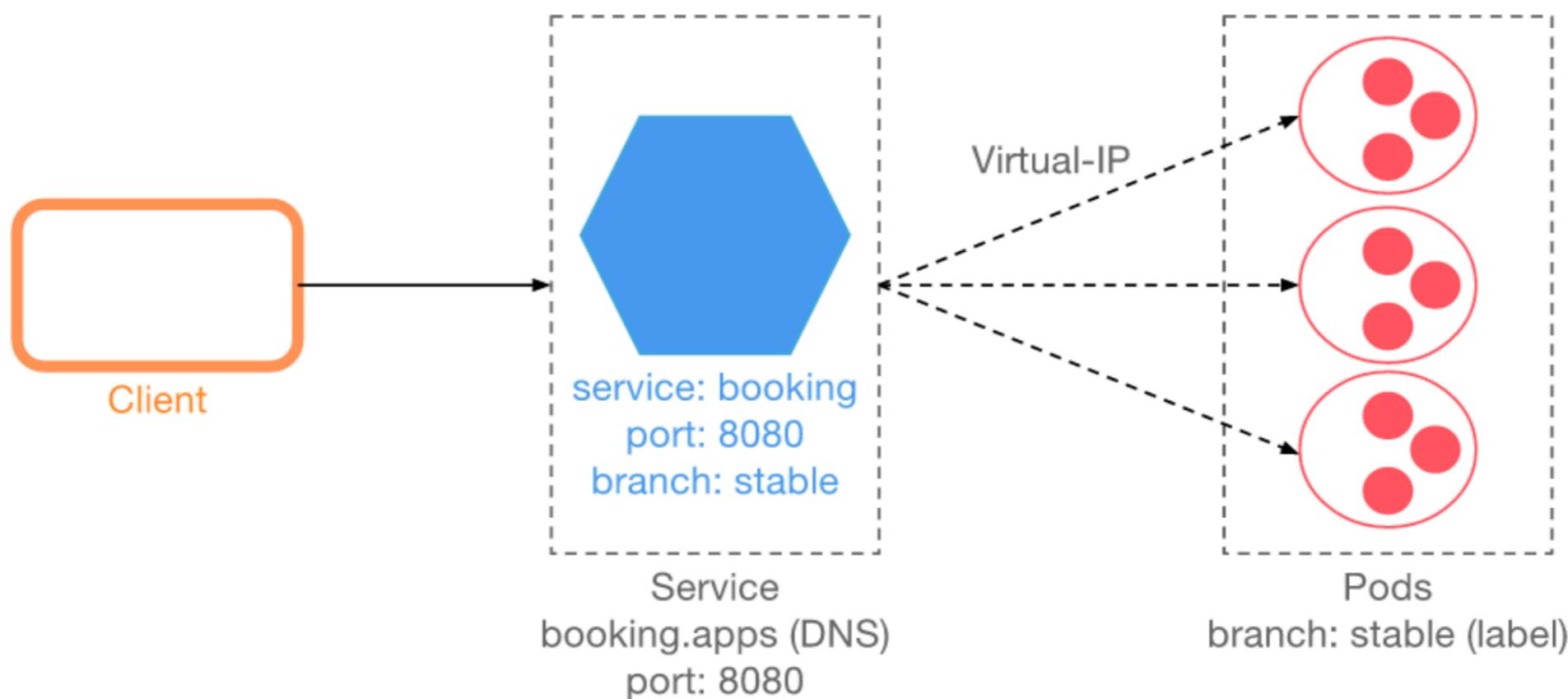
- Complexity to testing

- 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

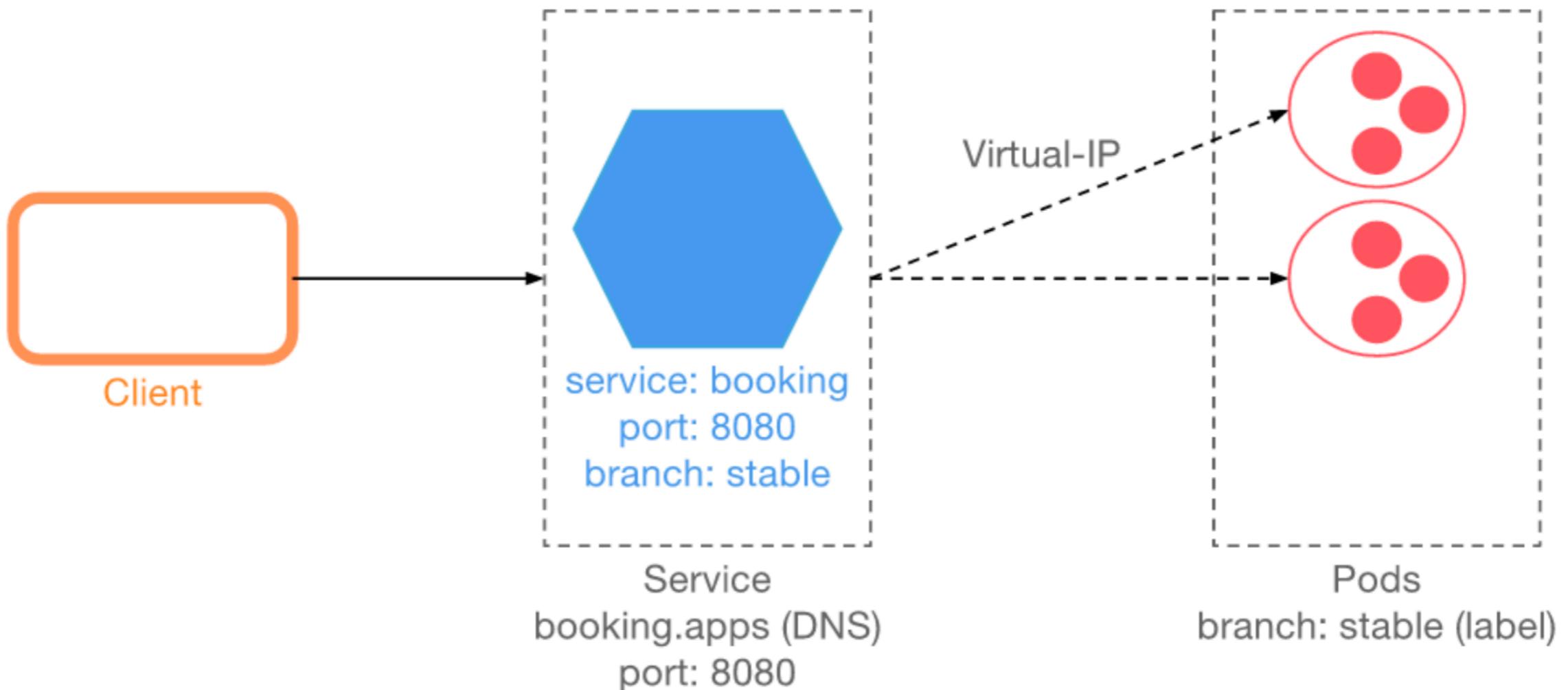
Redundancy (scale-out)

Lifecycle management (Software update)



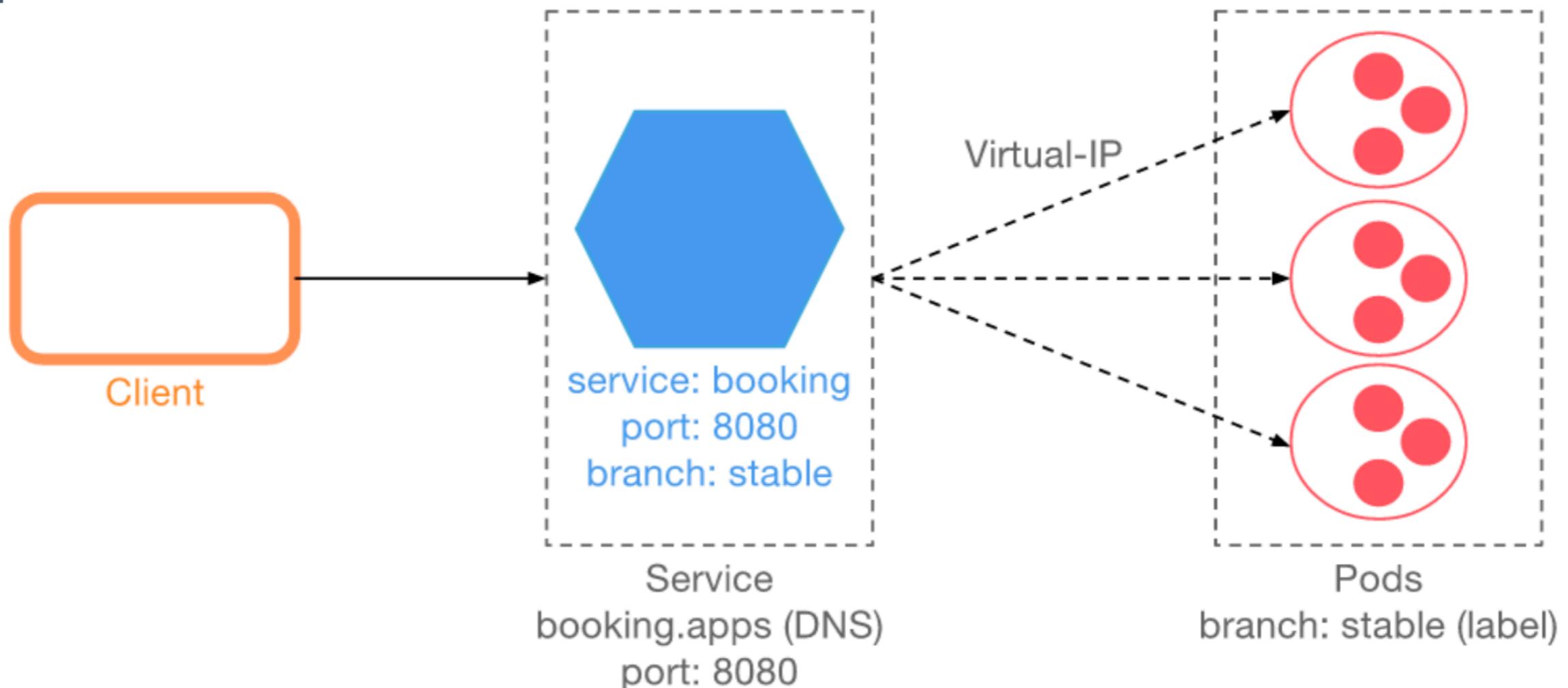
Fault-tolerant by design

replicas = 2



Fault-tolerant by design

replicas = 3



Deployment, not Infrastructure

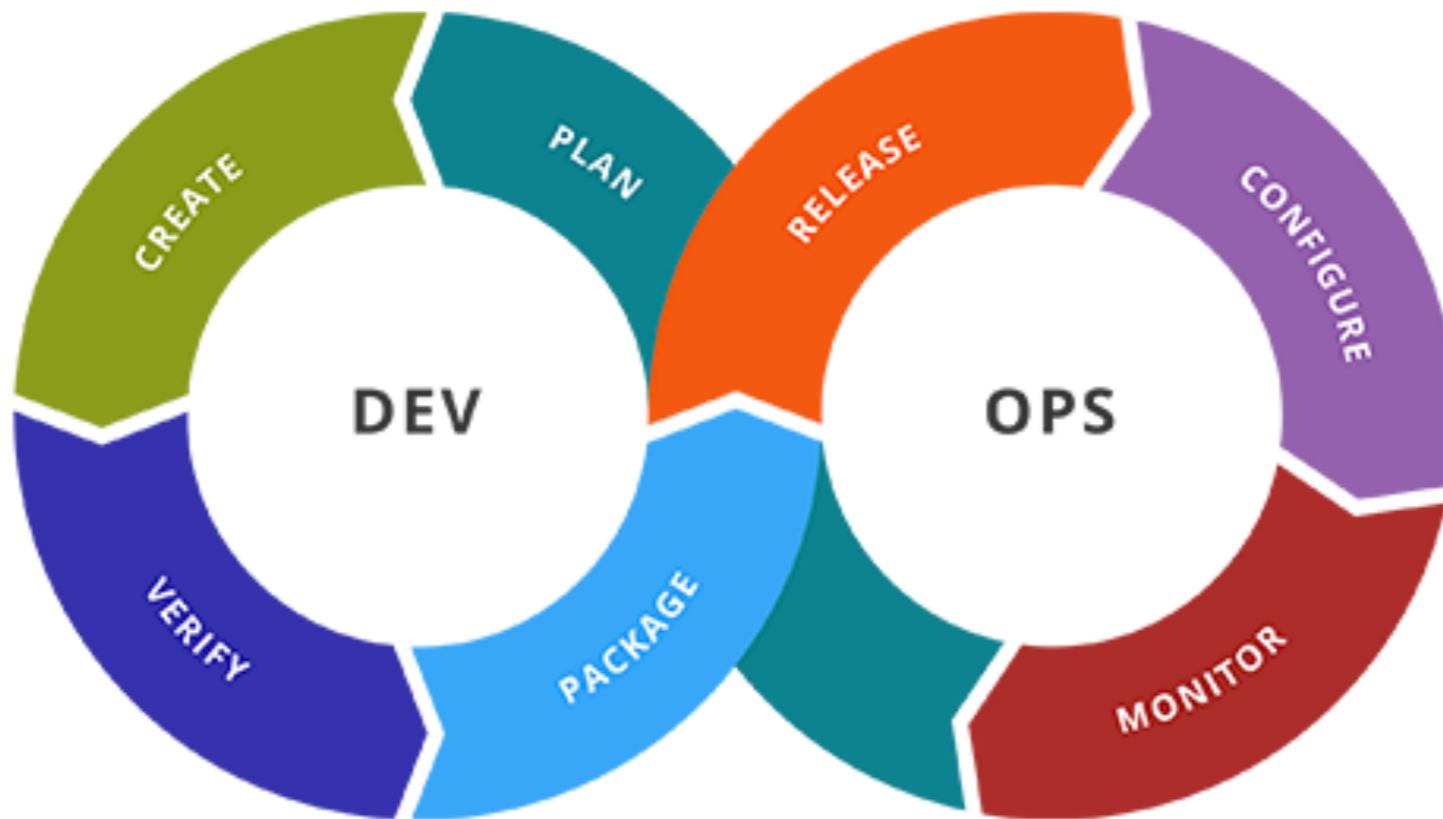
Software deployment is hard
Infrastructure provisioning/re-provisioning
Configuration networking and load balance
Redundancy (scale-out)
Lifecycle management (Software update)



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



minikube

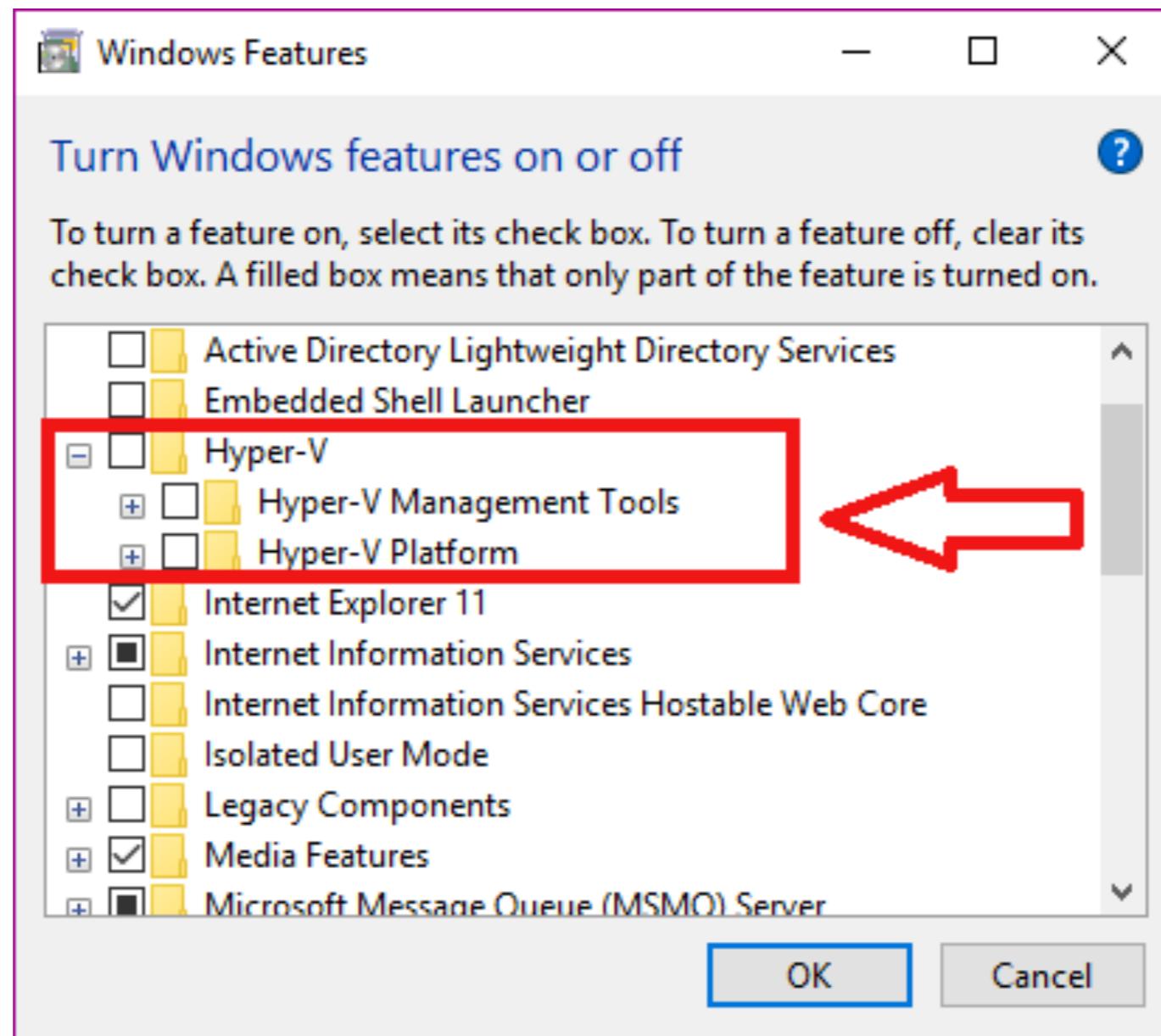


<https://github.com/kubernetes/minikube>



For Windows OS

Disable Hyper-V



Starting minikube

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



Software requirement

Kubernetes command-line tool (kubectl)

<https://kubernetes.io/docs/tasks/tools/install-kubectl/#install-kubectl-binary-via-curl>



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
```



Play with K8S

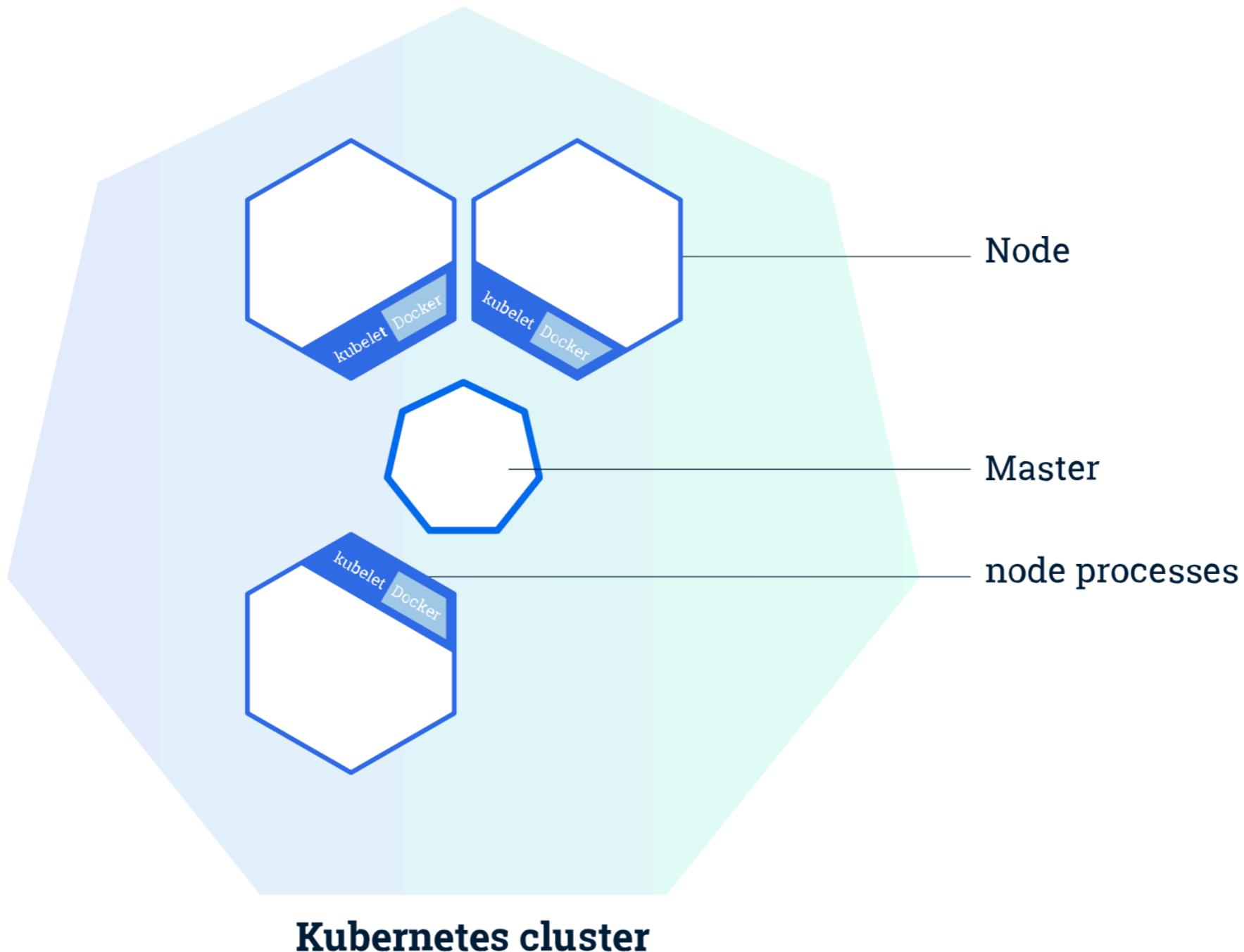
<https://labs.play-with-k8s.com/>



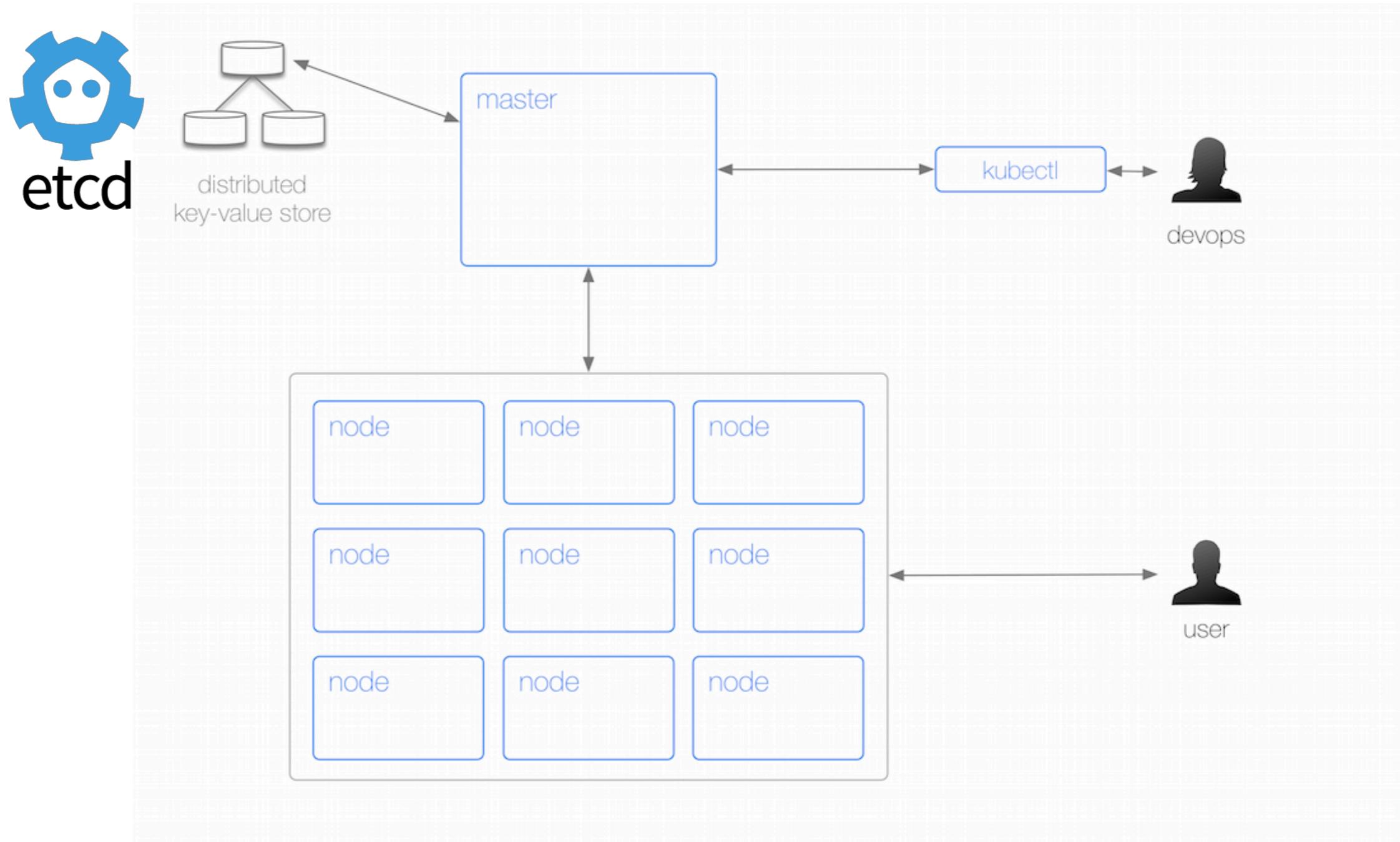
Basic of Kubernetes



Kubernetes cluster



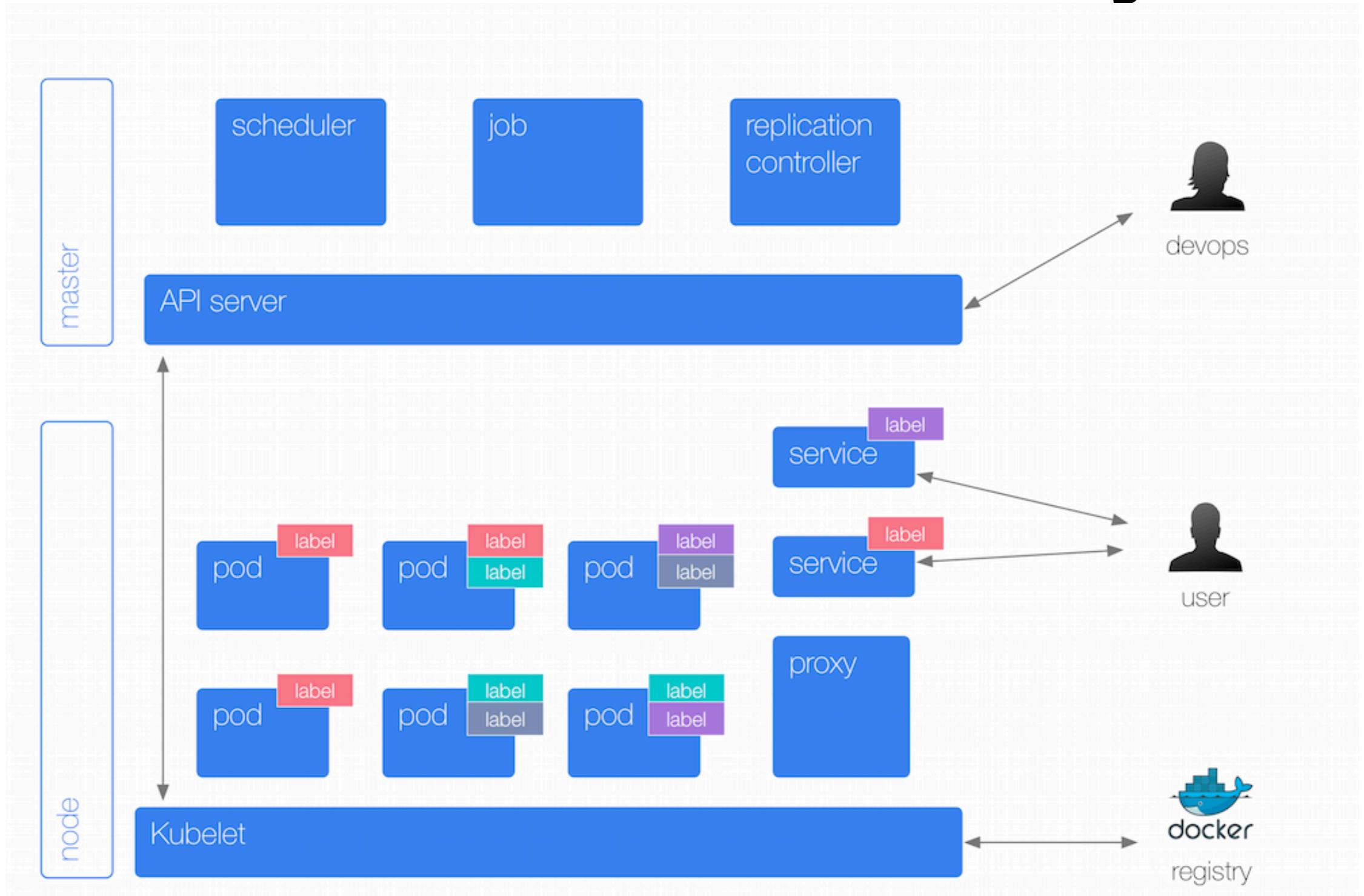
Kubernetes physical layout



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



Kubernetes abstraction layout



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



Hello with Kubernetes

file /01-hello/instruction.txt



Hello with Kubernetes

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 is a search bar and navigation links for Dashboard, Explore, Organizations, Create, and a user profile for `somkiat`. Below the header, it says "PUBLIC REPOSITORY". The repository name `somkiat/hello` is displayed with a star icon. A note indicates "Last pushed: 2 months ago". Below this, there are tabs for Repo Info, Tags, Collaborators, Webhooks, and Settings. The "Tags" tab is selected, showing a table with three rows of tags:

| Tag Name | Compressed Size | Last Updated | Action |
|----------|-----------------|--------------|--------|
| 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
```



Create a container

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

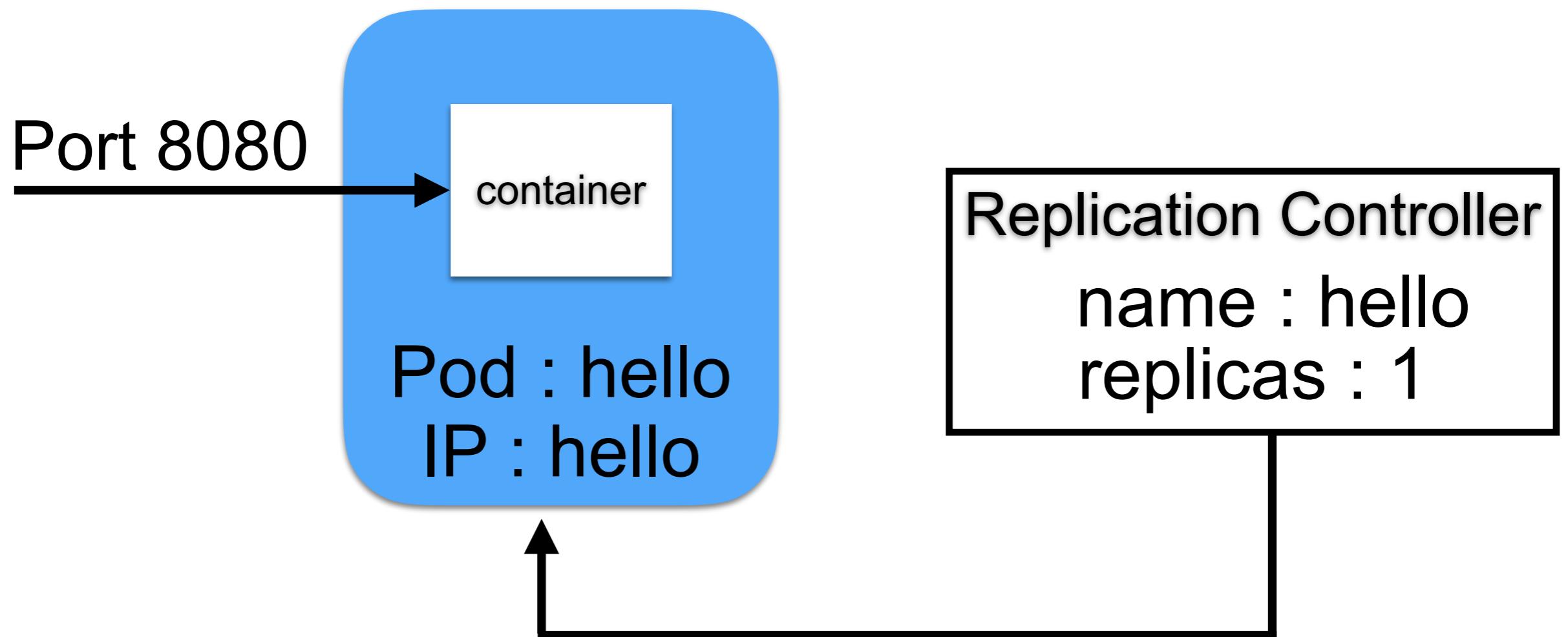


Create Replication Controller (RC)

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



Create a container

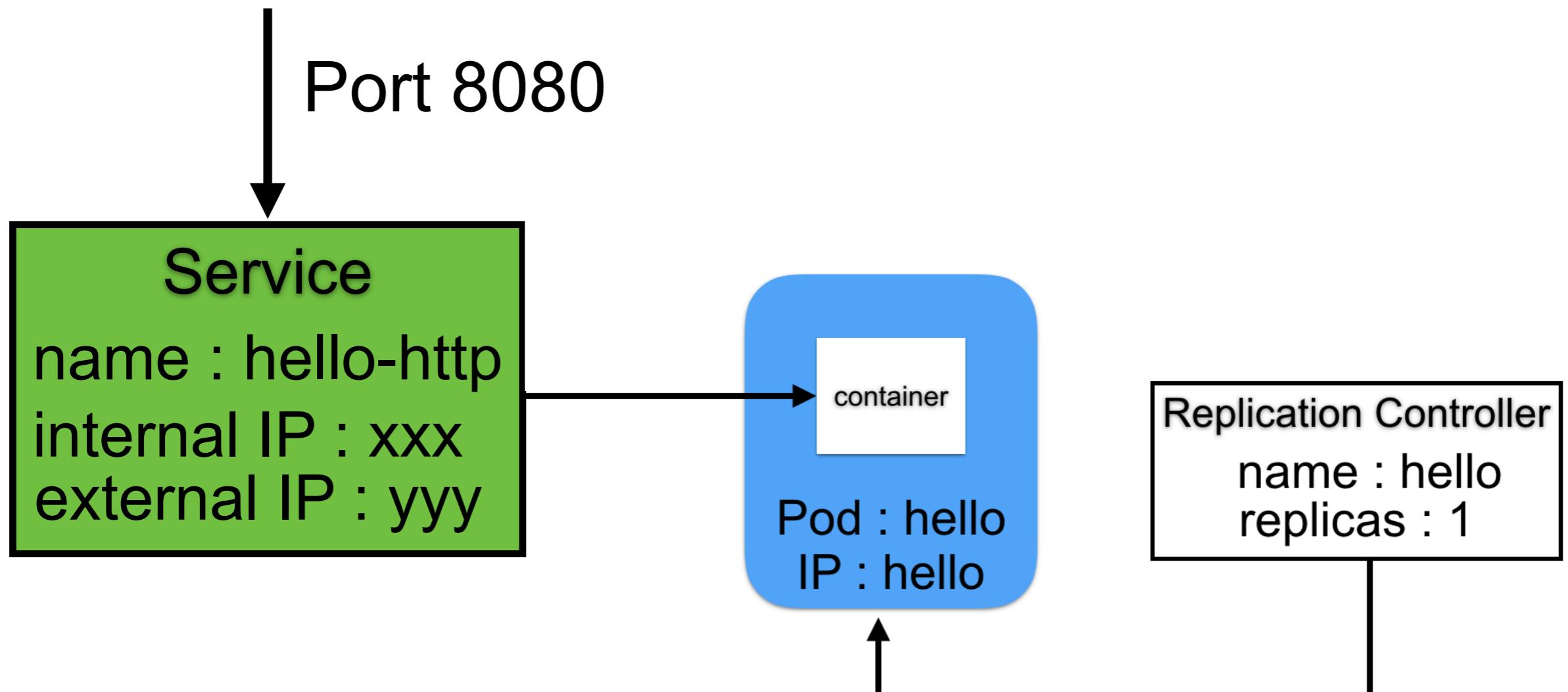


Expose RC with service

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



Expose RC with service



Access service with minikube

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

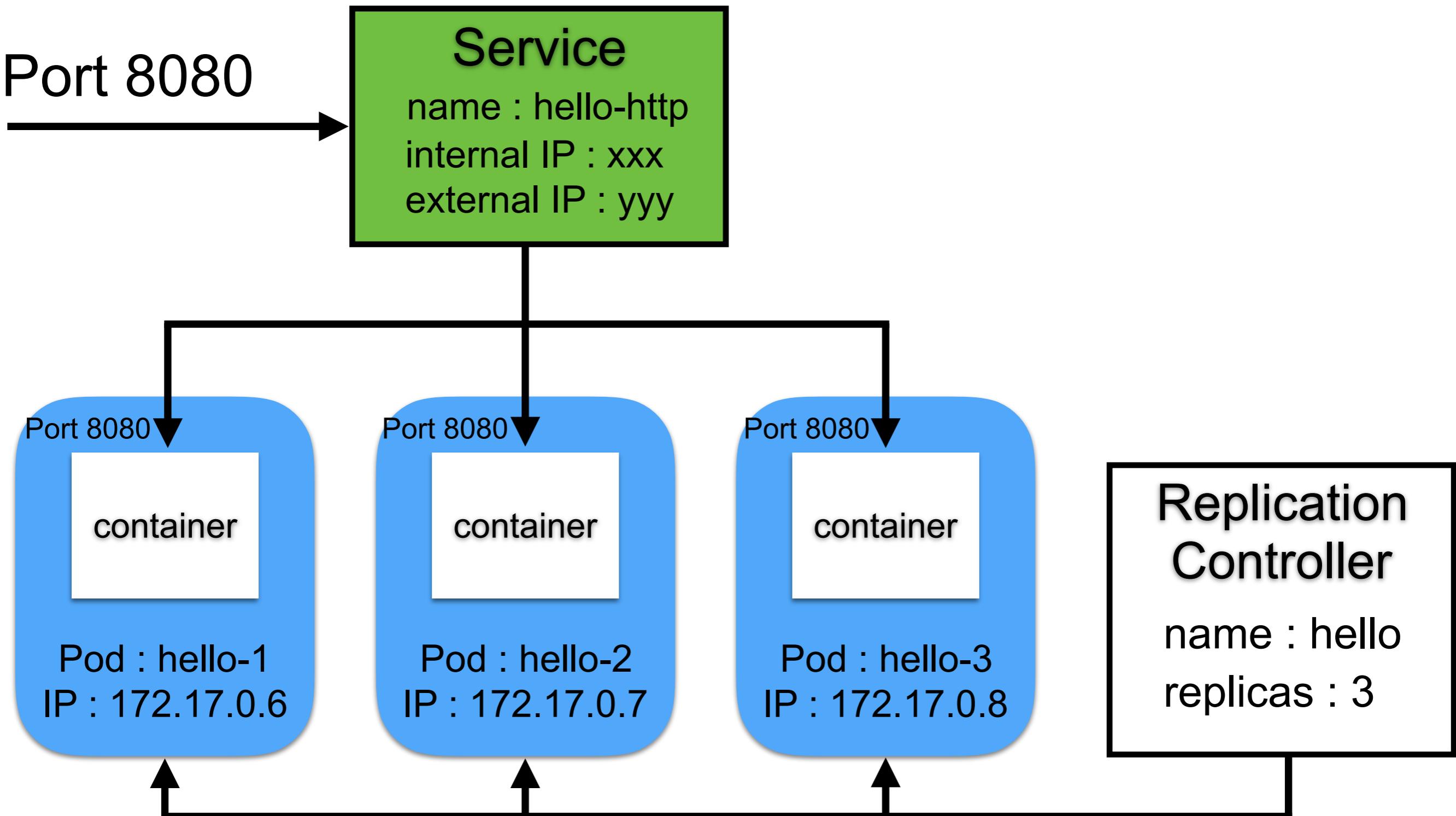


Scaling the application

```
$ 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 'Namespace' is open, showing 'default' as the selected option. The main content area is titled 'Workloads' and contains a section titled 'Workloads Statuses' with two green circular charts, both showing '100.00%' for 'Pods' and 'Replication Controllers'. Below this, there is a table titled 'Pods' listing three entries: 'hello-lw2p7', 'hello-wwsnh', and 'hello-pqnpz'. Each entry includes columns for Name, Node, Status, Restarts, and Age. The 'Name' column is sorted by name, 'Status' is sorted by status, and 'Age' is sorted by age. The 'Status' column for all three entries shows 'Running'. The 'Age' column shows values of '53 minutes' and '59 minutes'. Each row has a set of three icons at the end.

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



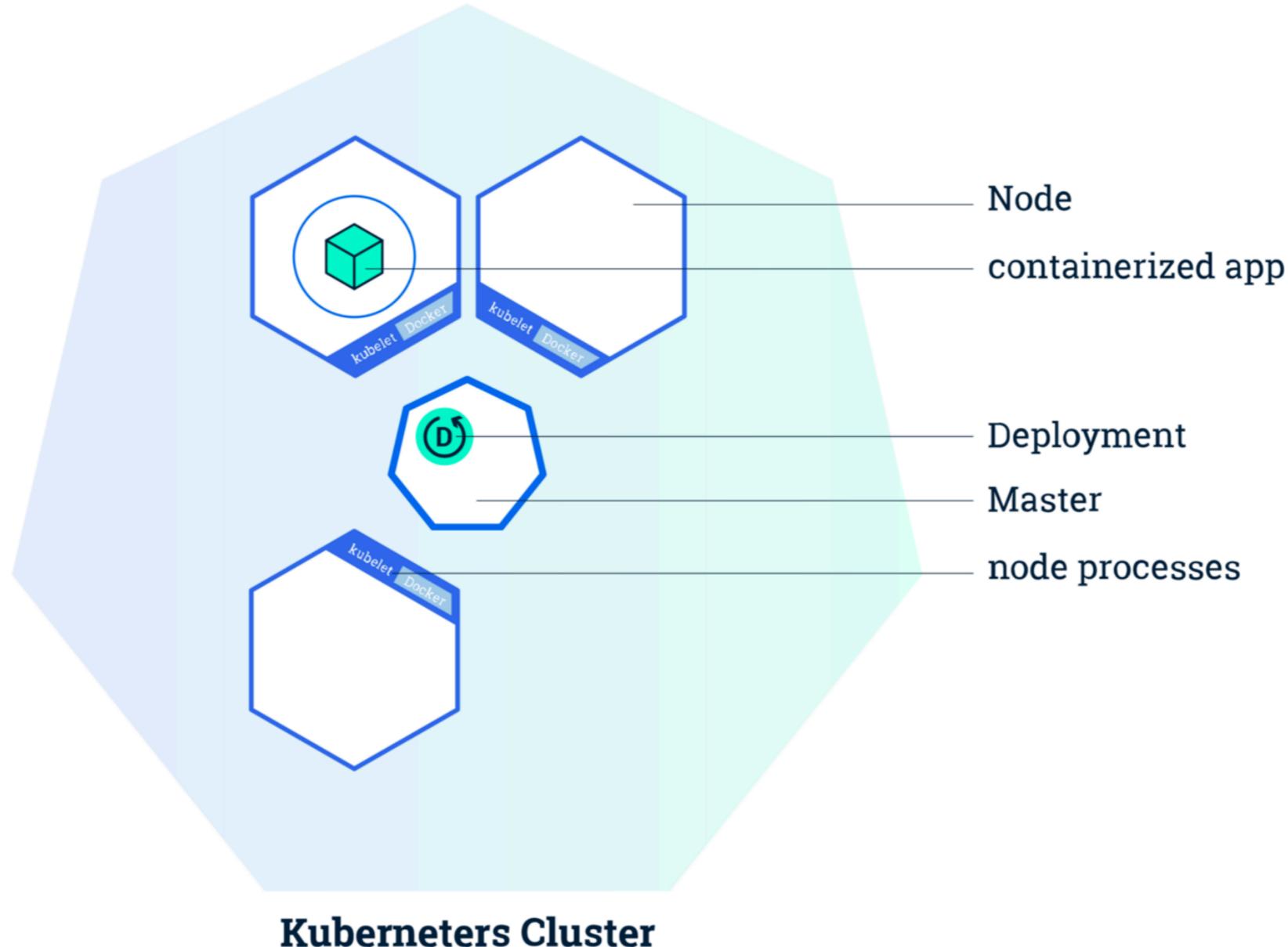
Welcome to Kubernetes



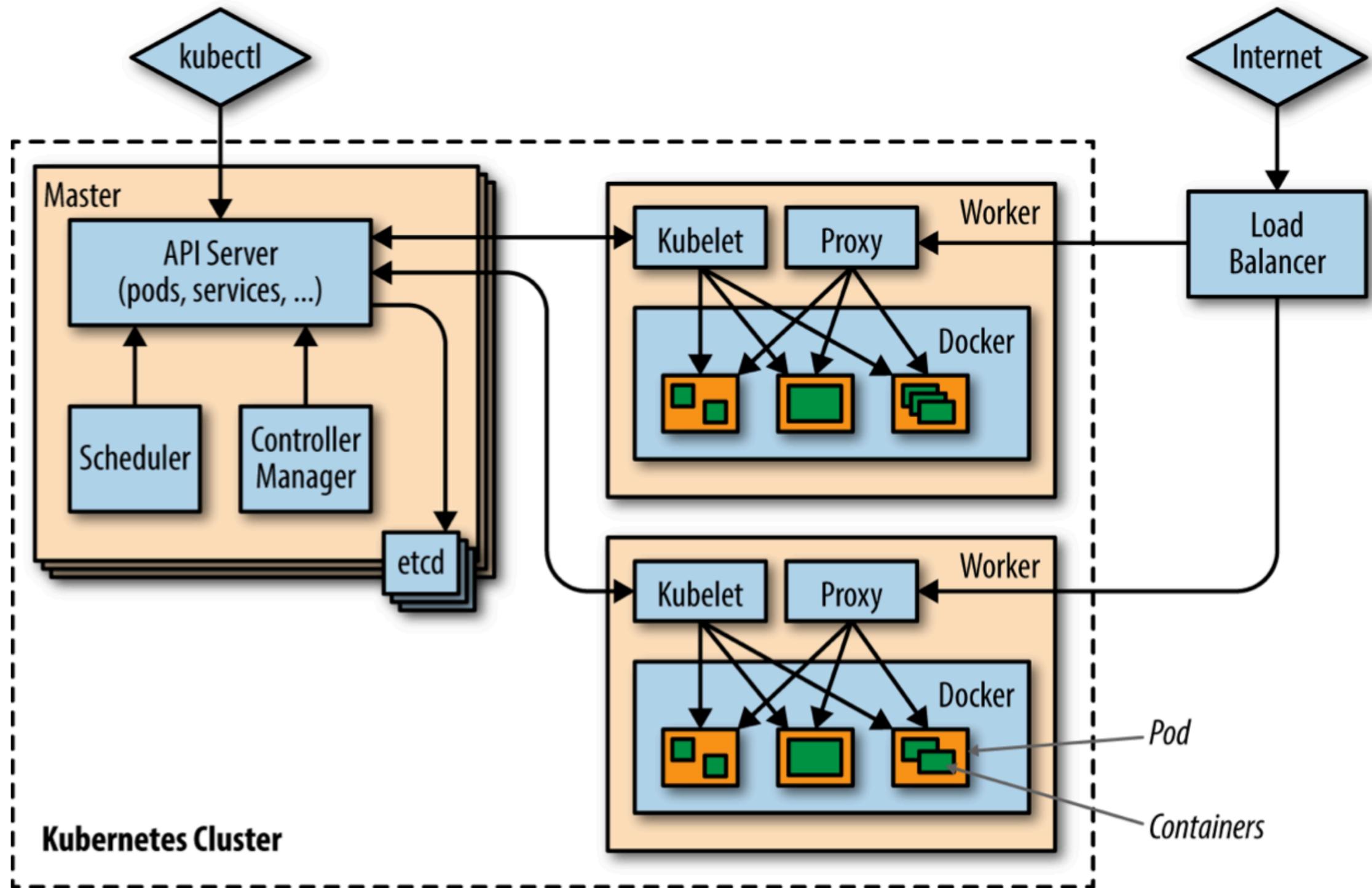
Kubernetes Architecture



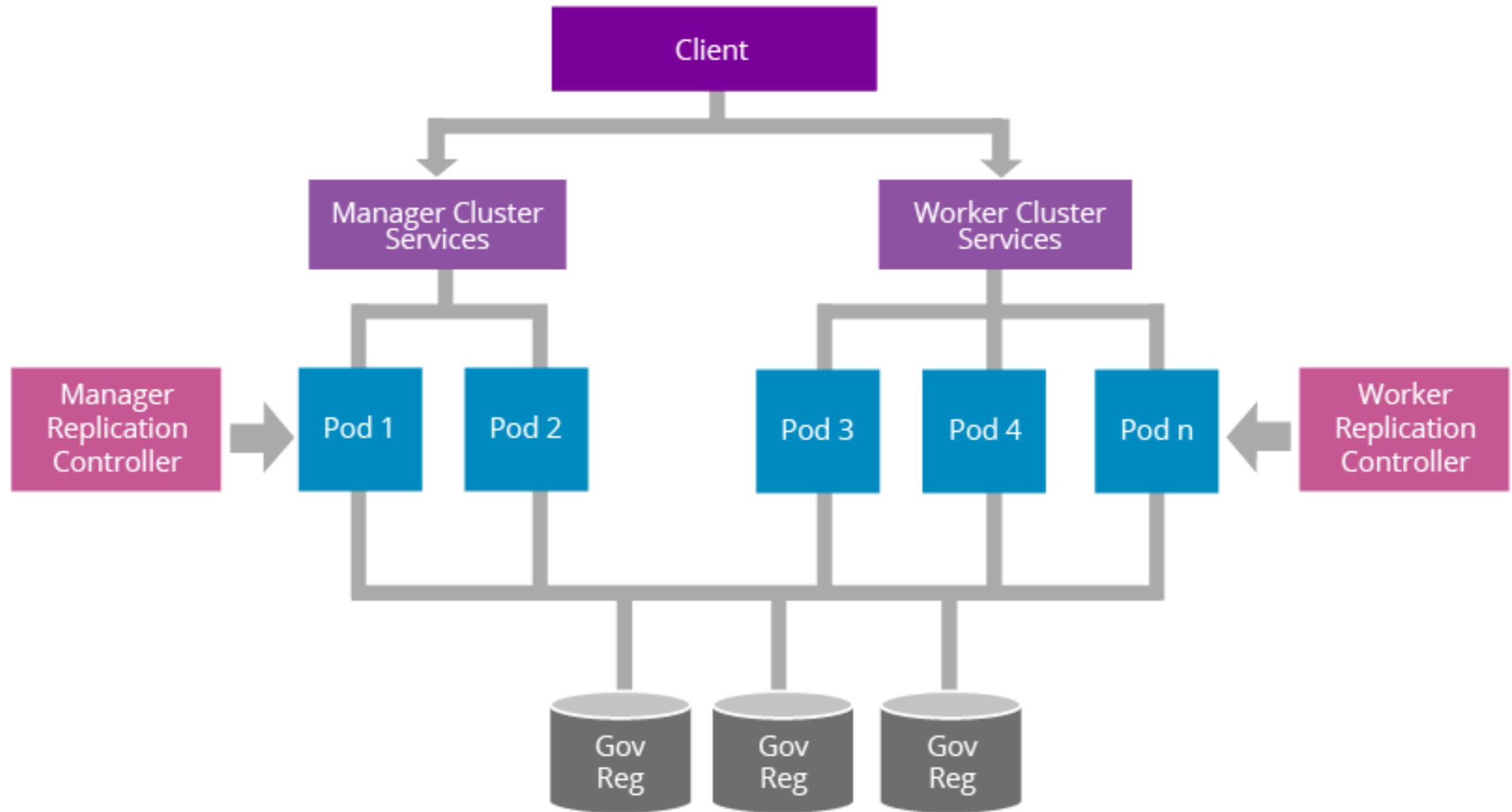
Kubernetes cluster



Kubernetes architecture



Kubernetes architecture



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



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

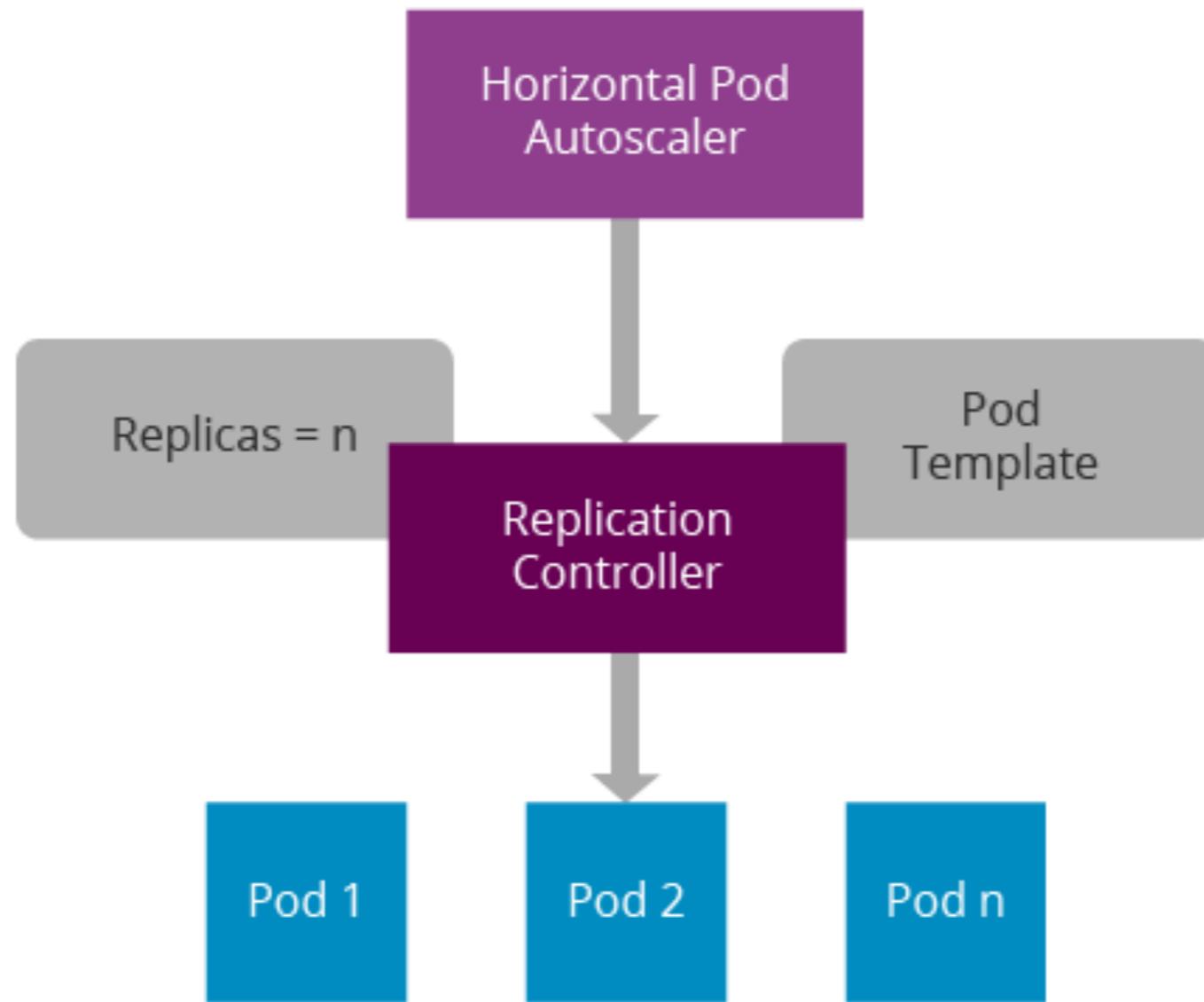
| CPU Requests | CPU Limits | Memory Requests | Memory Limits |
|--------------|------------|-----------------|---------------|
| 5m (0%) | 0 (0%) | 50Mi (2%) | 0 (0%) |
| 260m (13%) | 0 (0%) | 110Mi (5%) | 170Mi (8%) |
| 0 (0%) | 0 (0%) | 0 (0%) | 0 (0%) |
| 0 (0%) | 0 (0%) | 0 (0%) | 0 (0%) |



Horizontal Pod Autoscaler (HPA)



Horizontal Pod Autoscaler



Horizontal Pod Autoscaler

Monitor workload on Pods (based on CPU)
and automatic scaling-up application

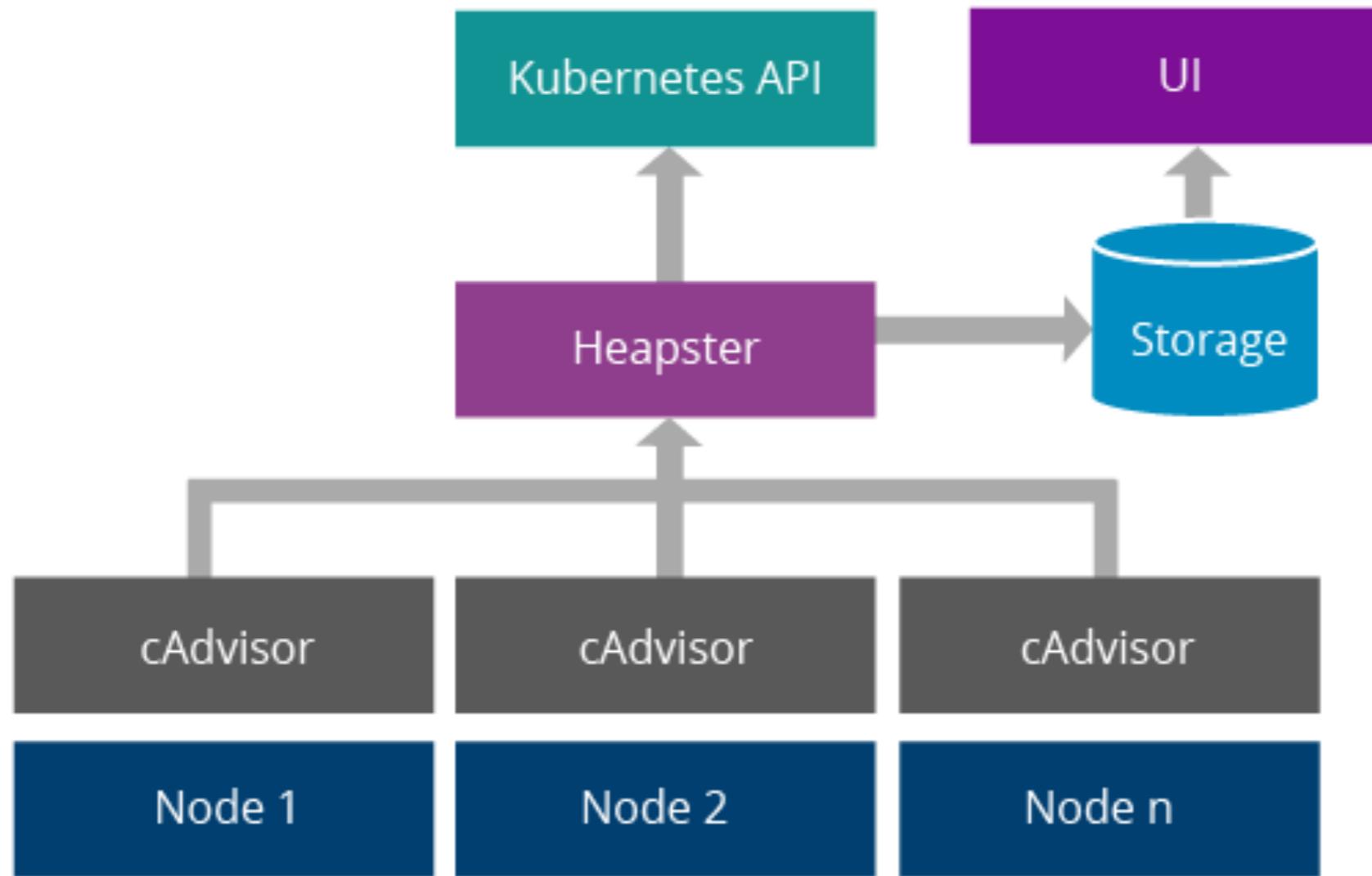
Scaling-up and down for your app need !!



Resource usage monitoring



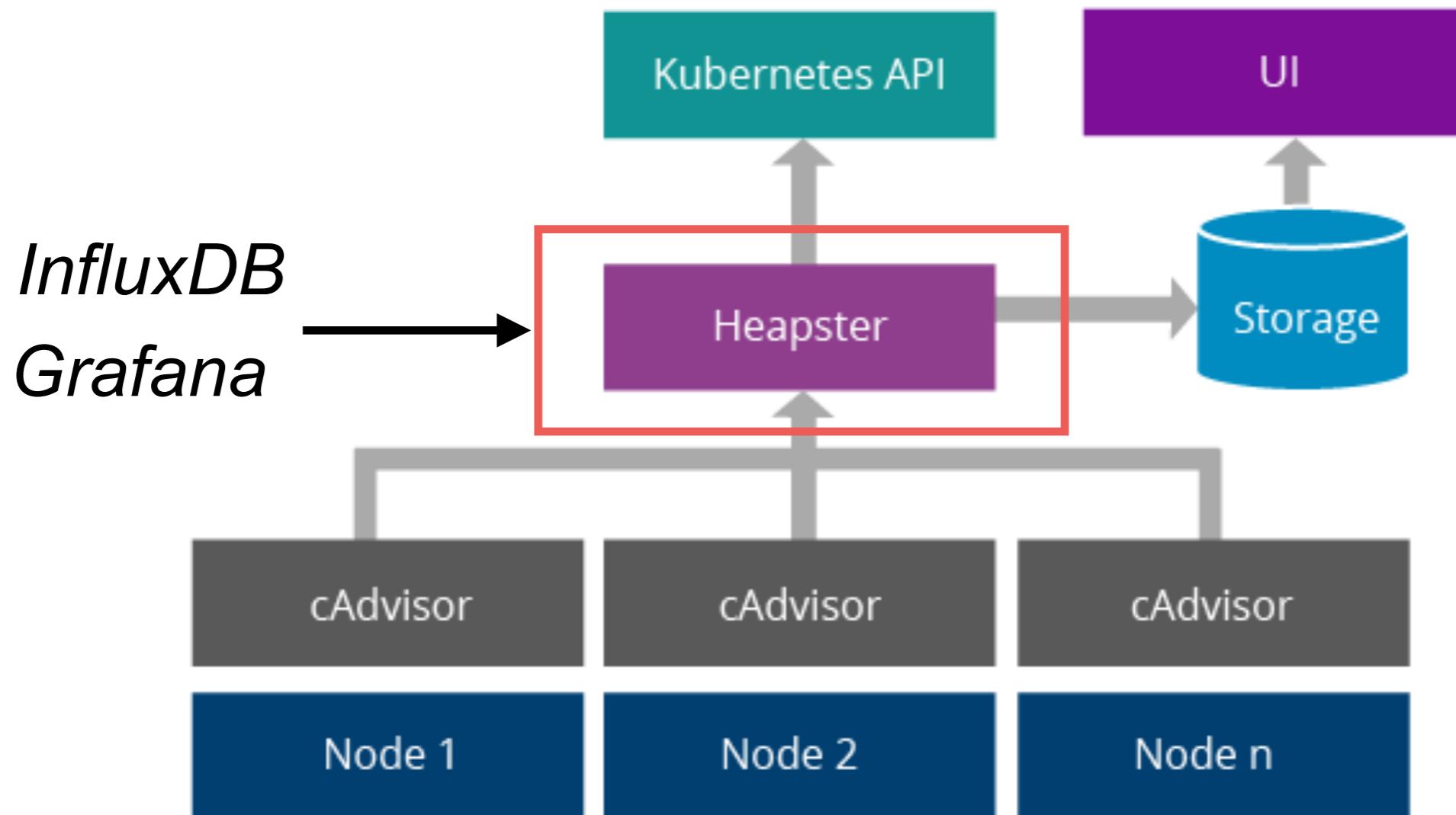
Resource usage monitoring



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



Resource usage monitoring



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



Heapster deprecation timeline

| Kubernetes | Action | Policy and Support |
|------------|---------------------|--|
| v 1.11 | Initial deprecation | No new feature + fix bugs |
| v 1.12 | Setup removal | The optional to install heapster in Kubernetes |
| v 1.13 | Removal | No new fix bug, move to retired organization |

<https://github.com/kubernetes/heapster/blob/master/docs/deprecation.md>



Try Kubernetes Metric Server

Starting from Kubernetes 1.8

Resources usage metrics => CPU and memory

Use via Metric API and Metric server

<https://github.com/kubernetes-incubator/metrics-server>

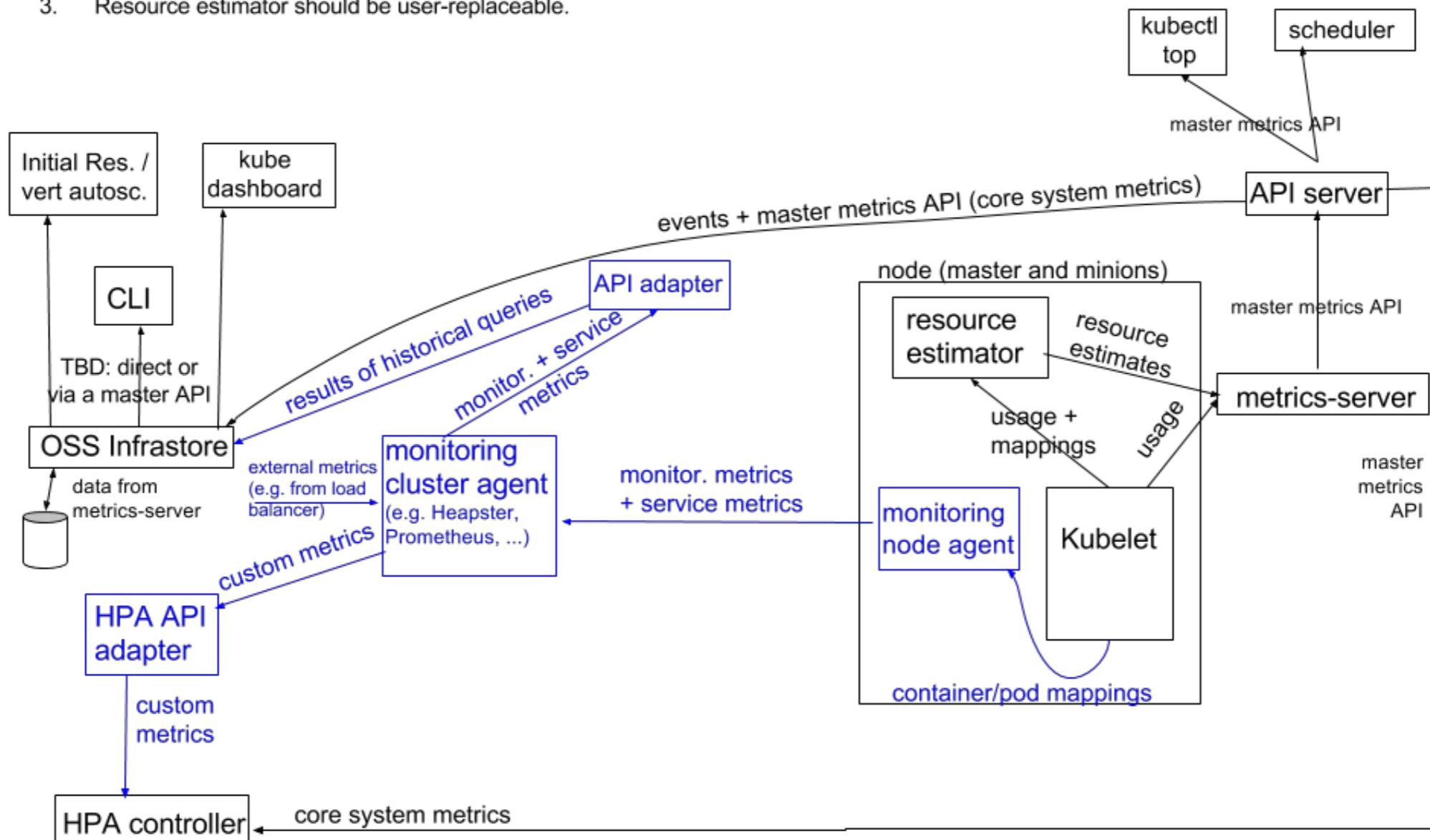


Architecture diagram

Monitoring architecture proposal: OSS
(arrows show direction of metrics flow)

Notes

1. Arrows show direction of metrics flow.
2. Monitoring pipeline is in blue. It is user-supplied and optional.
3. Resource estimator should be user-replaceable.



https://github.com/kubernetes/community/blob/master/contributors/design-proposals/instrumentation/monitoring_architecture.md

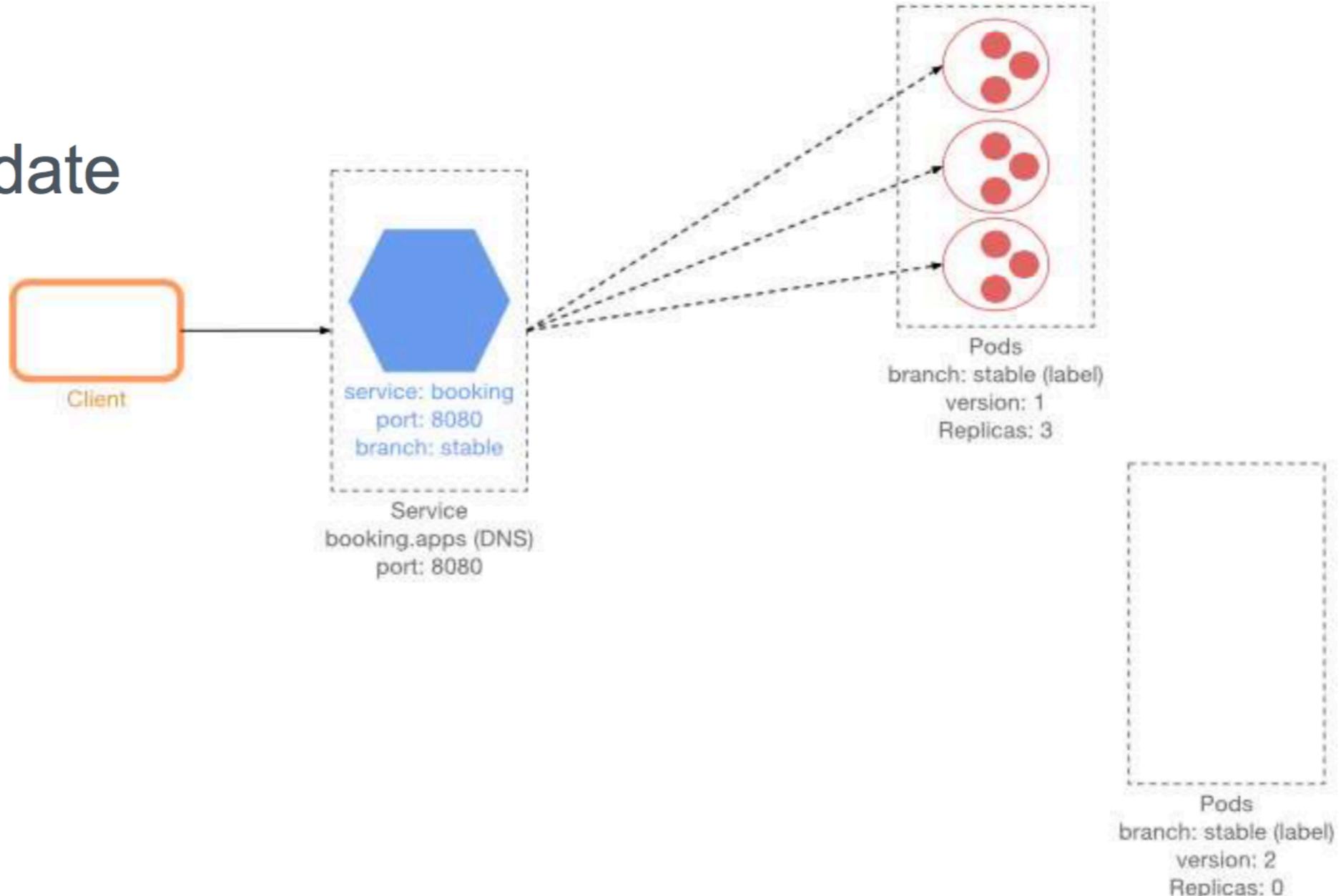


Automated rollouts & rollbacks



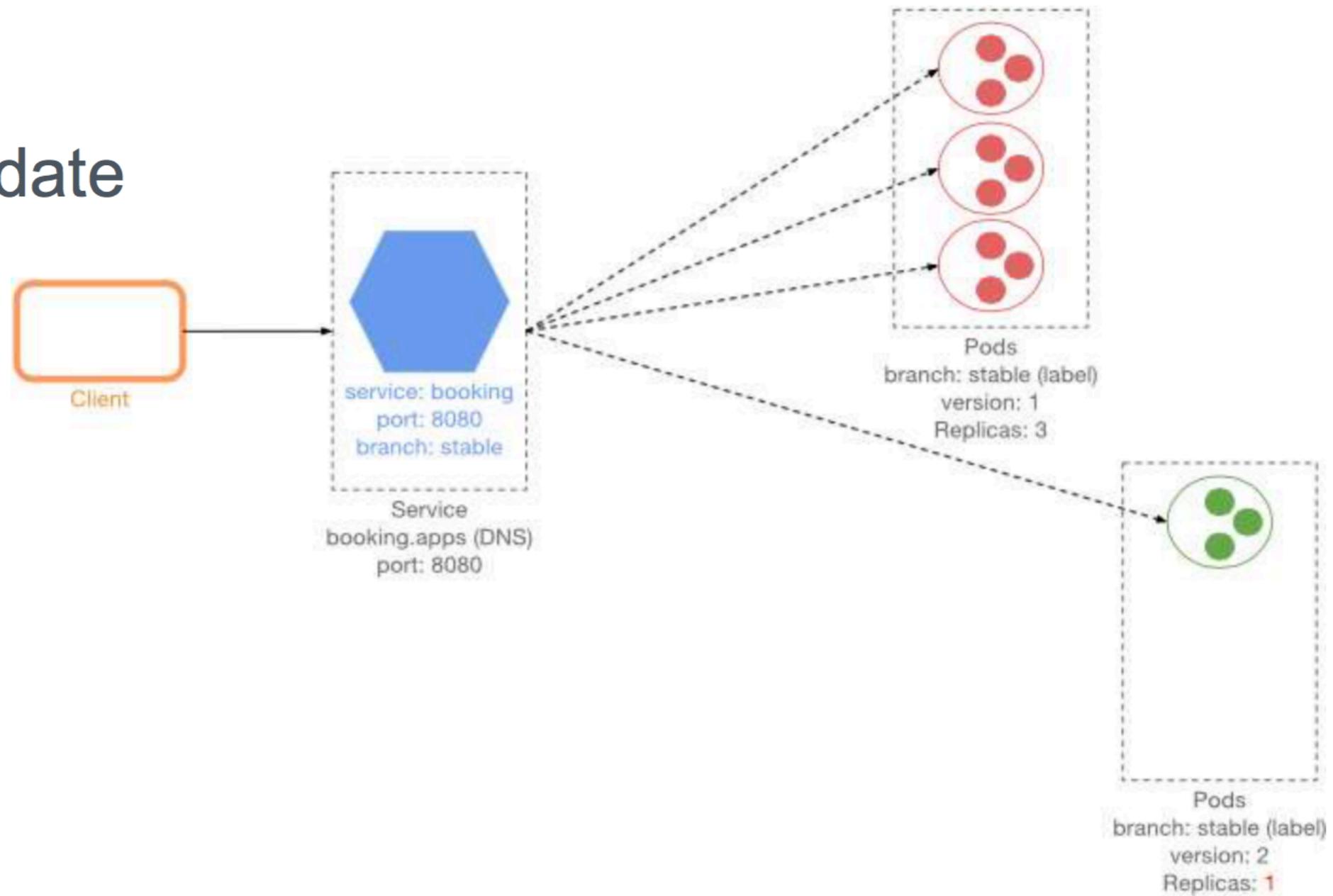
Rolling update (1)

Rolling Update



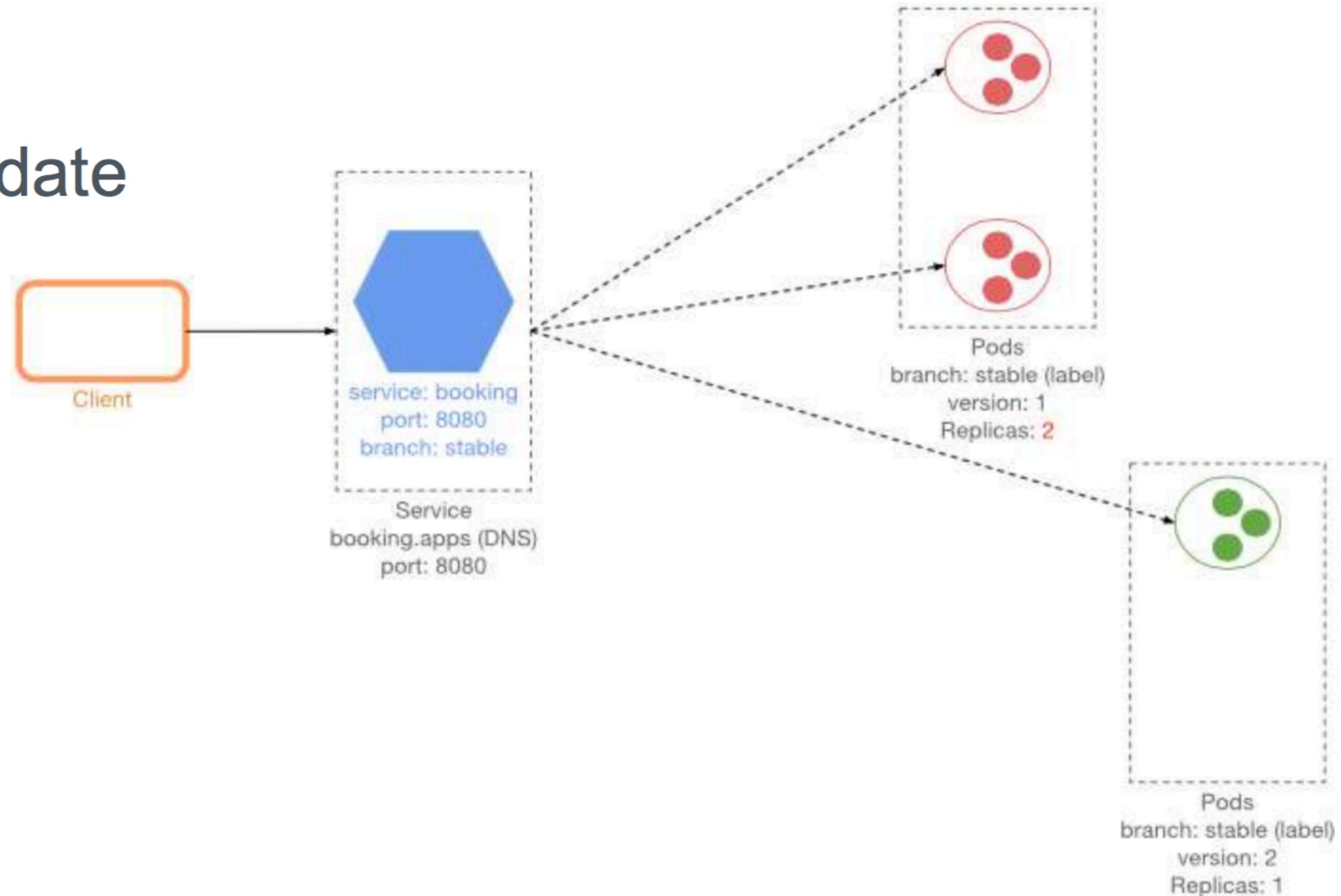
Rolling update (2)

Rolling Update



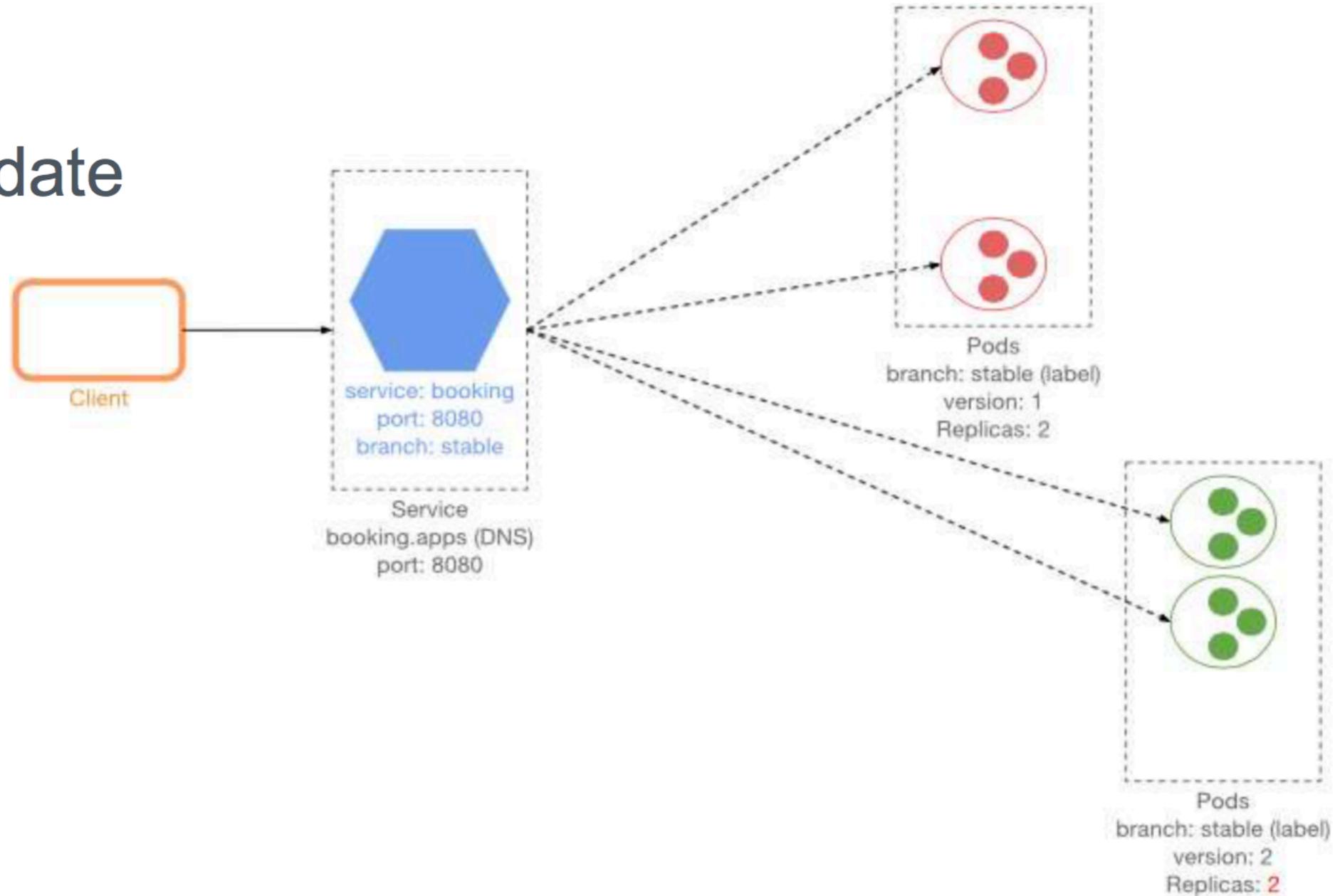
Rolling update (3)

Rolling Update



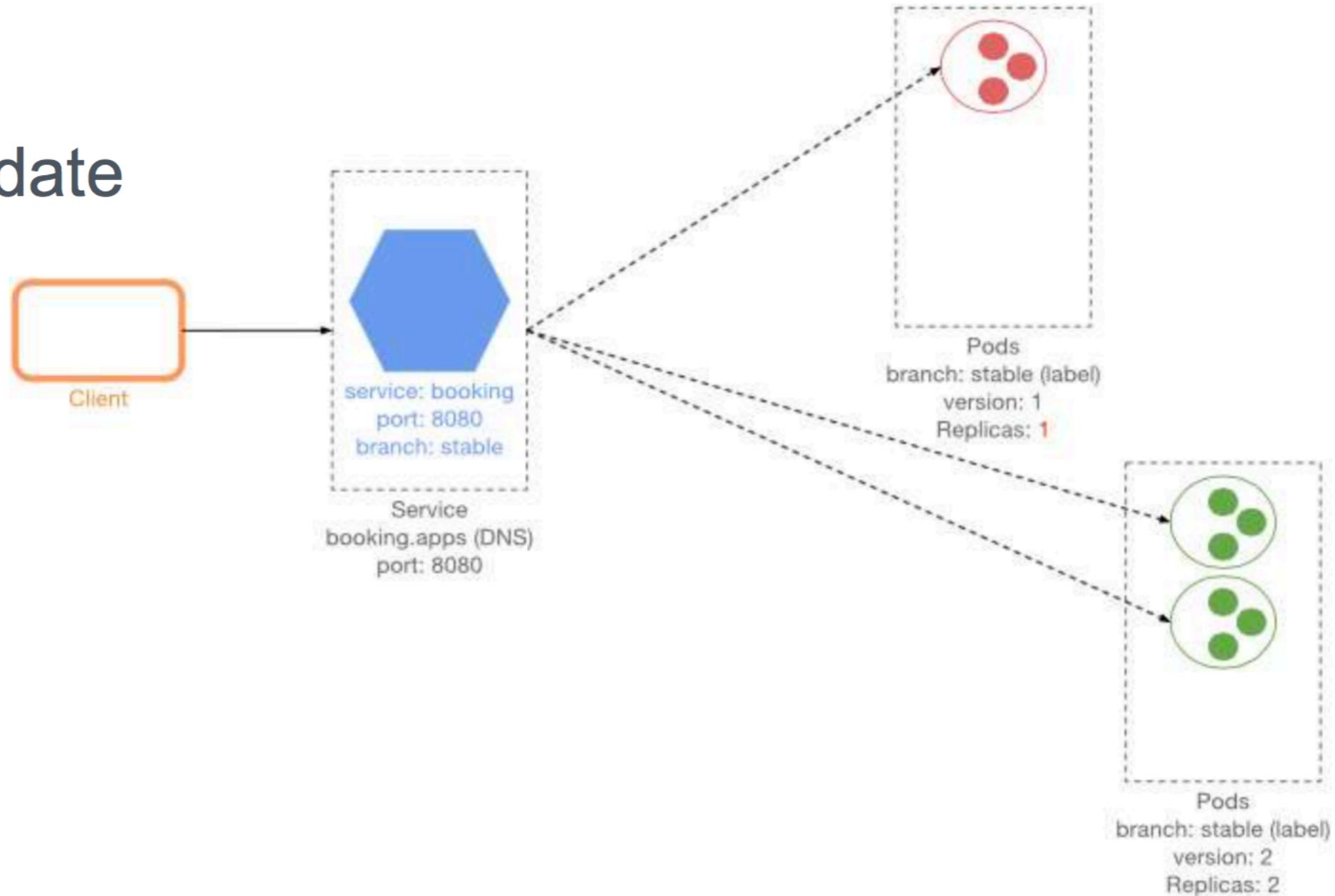
Rolling update (4)

Rolling Update



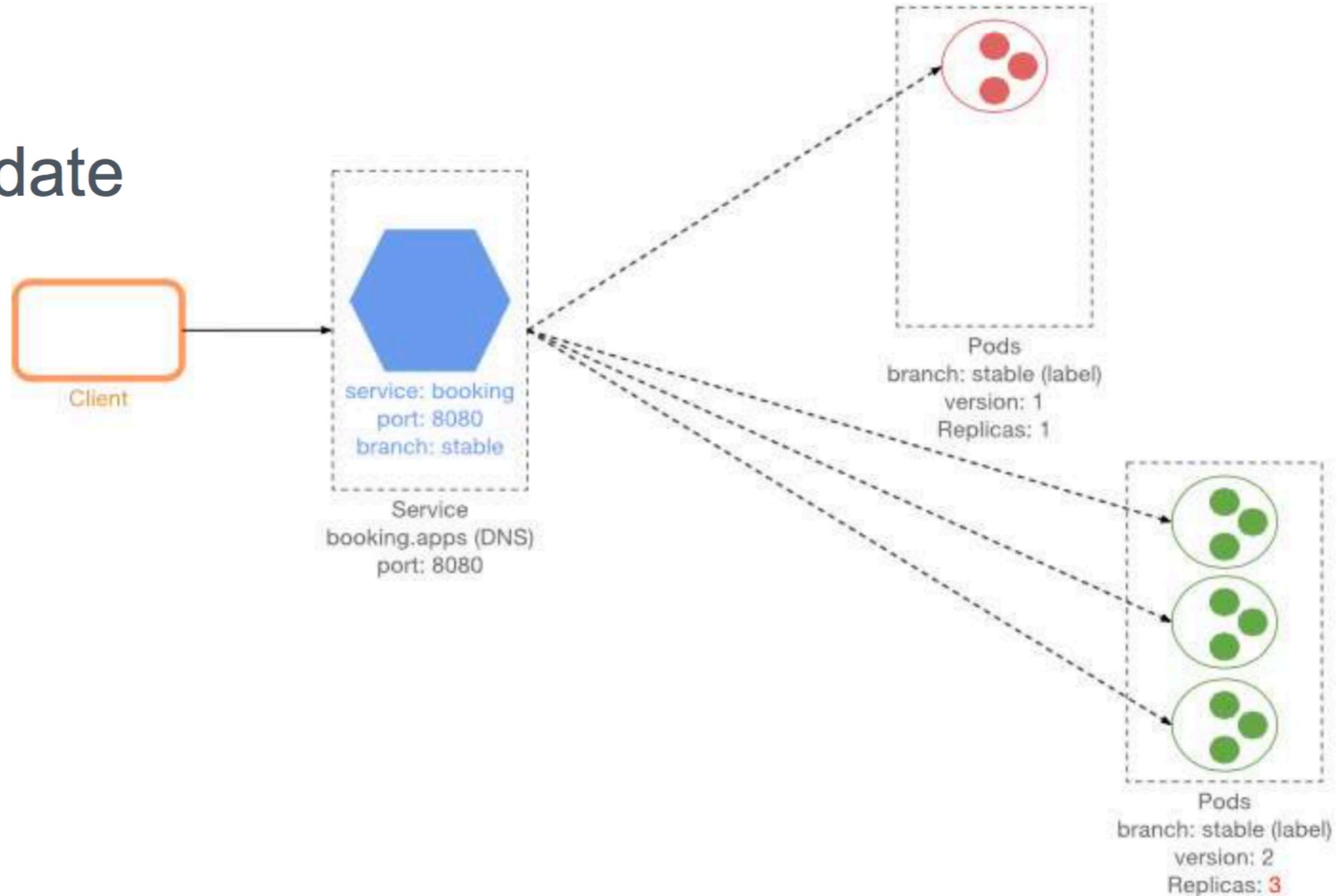
Rolling update (5)

Rolling Update



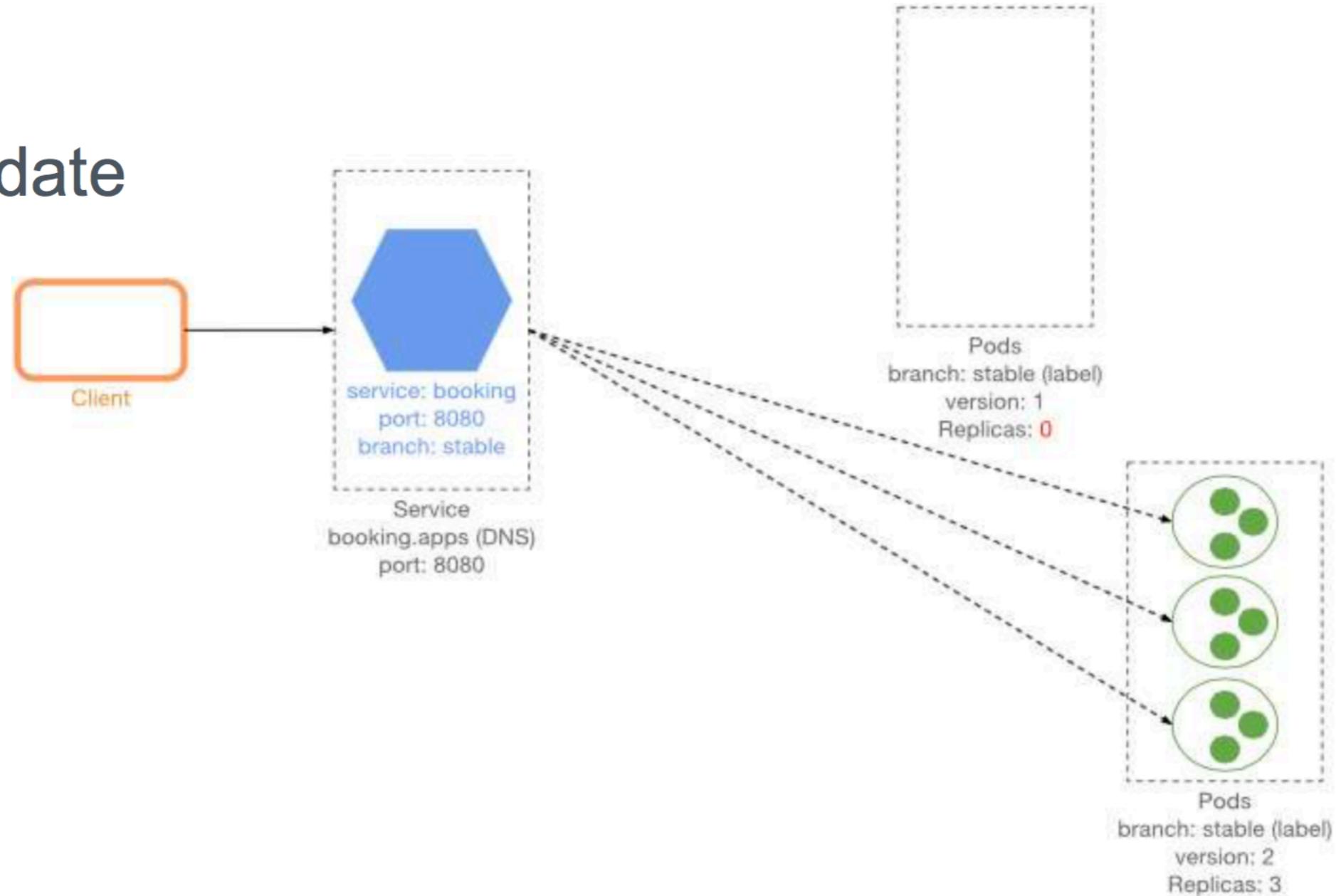
Rolling update (6)

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

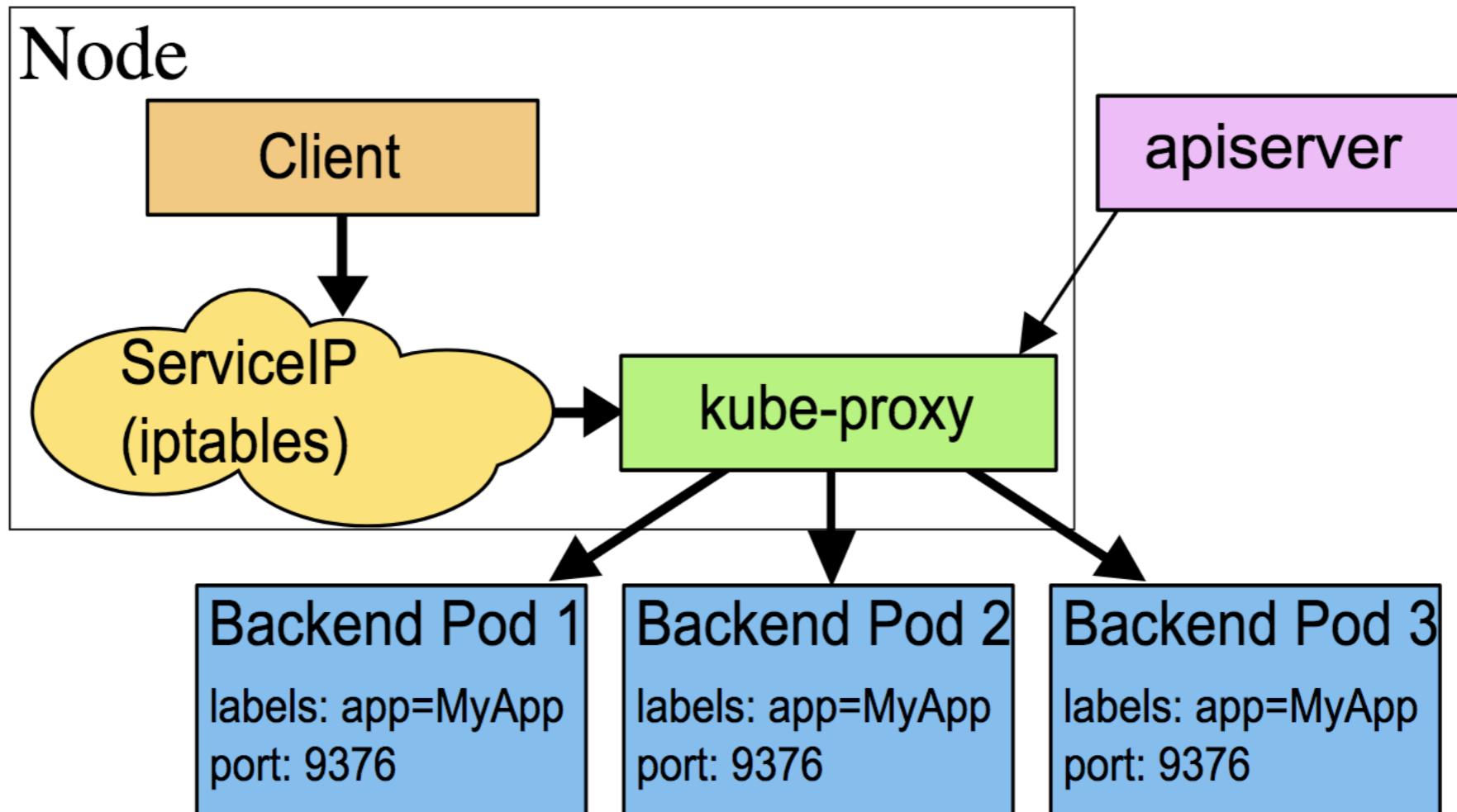
Discovery use for service to look Pod

Support load balance between Pods (replica)

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



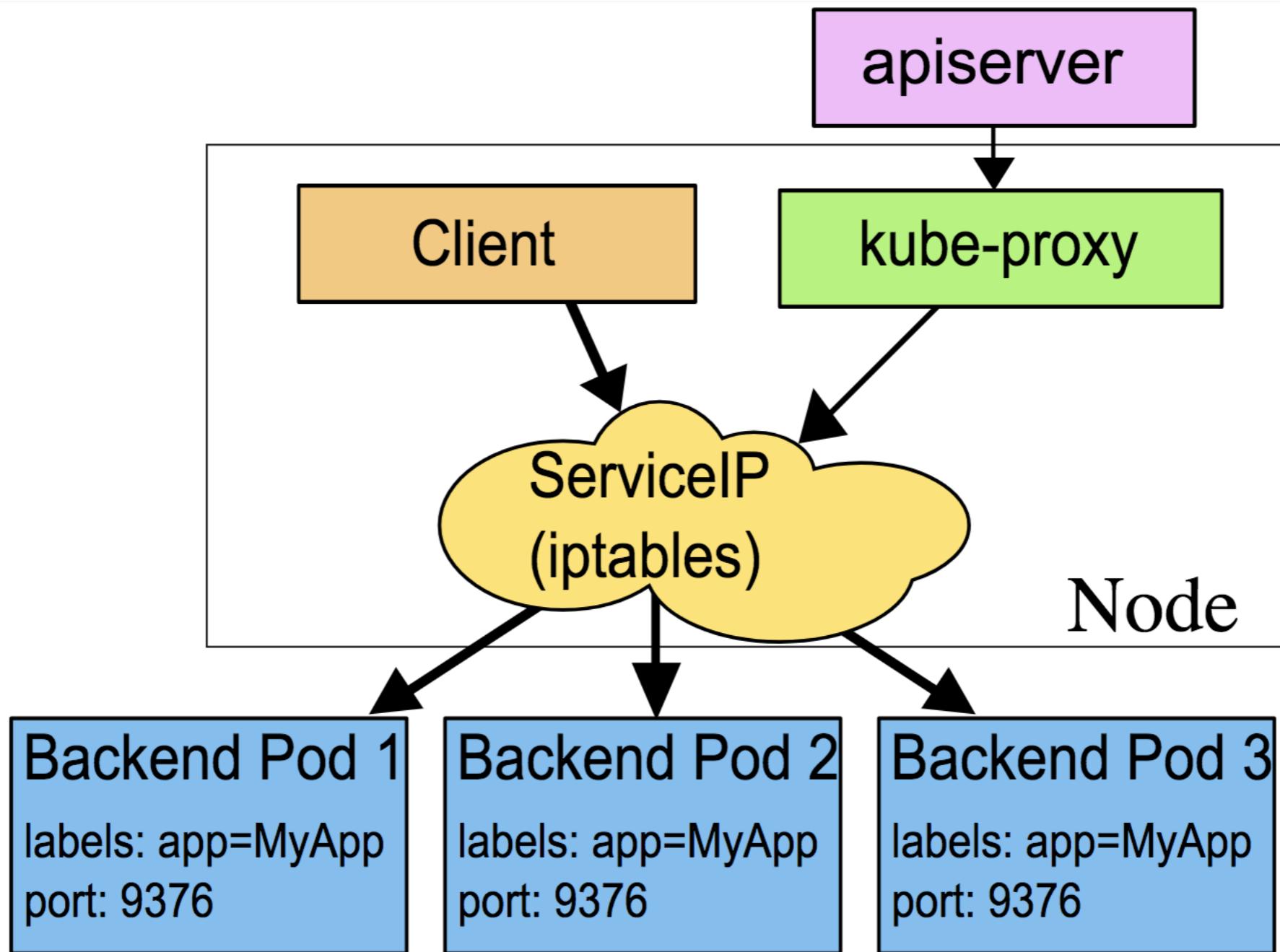
Proxy-mode: userspace



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



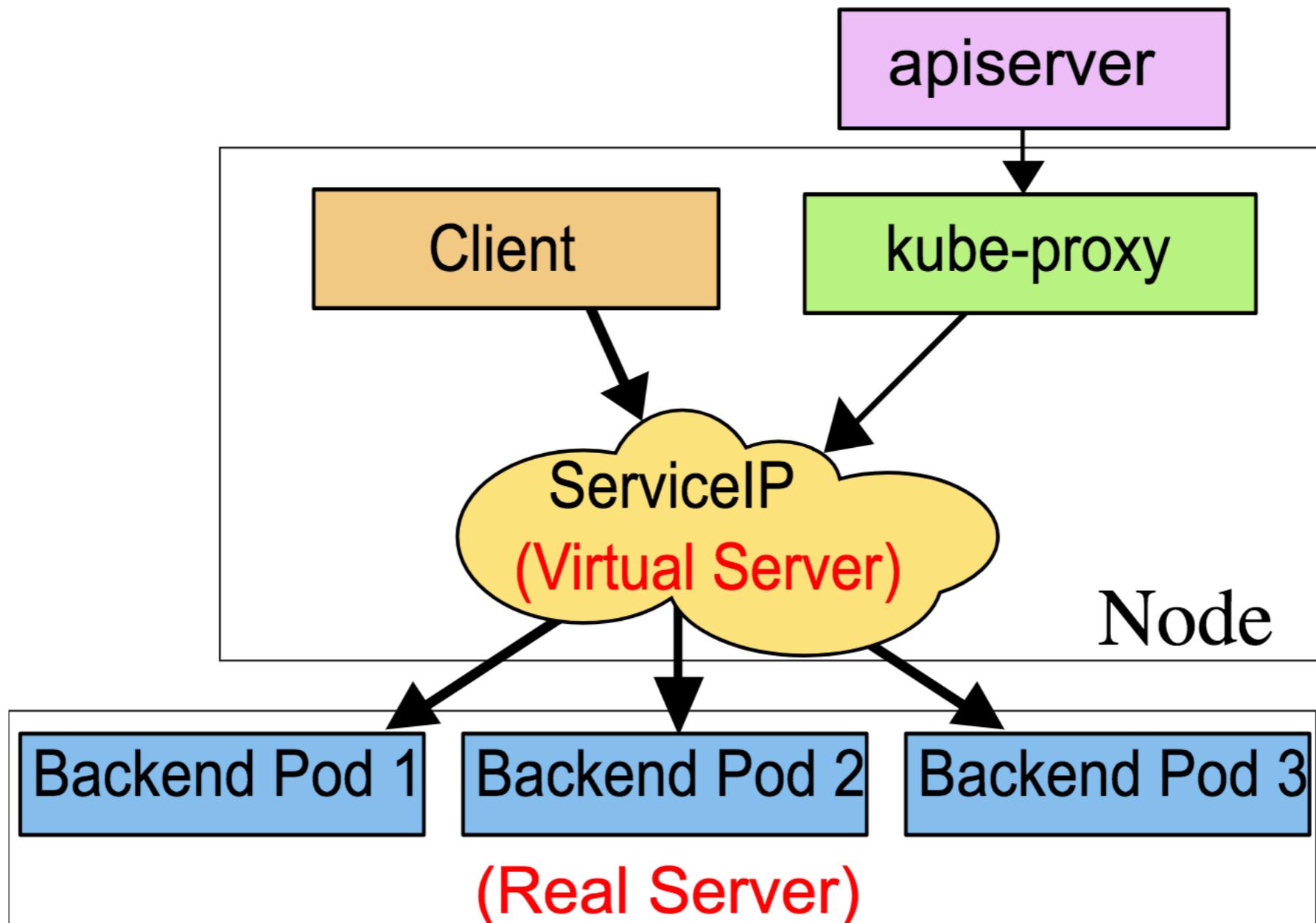
Proxy-mode: iptables



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



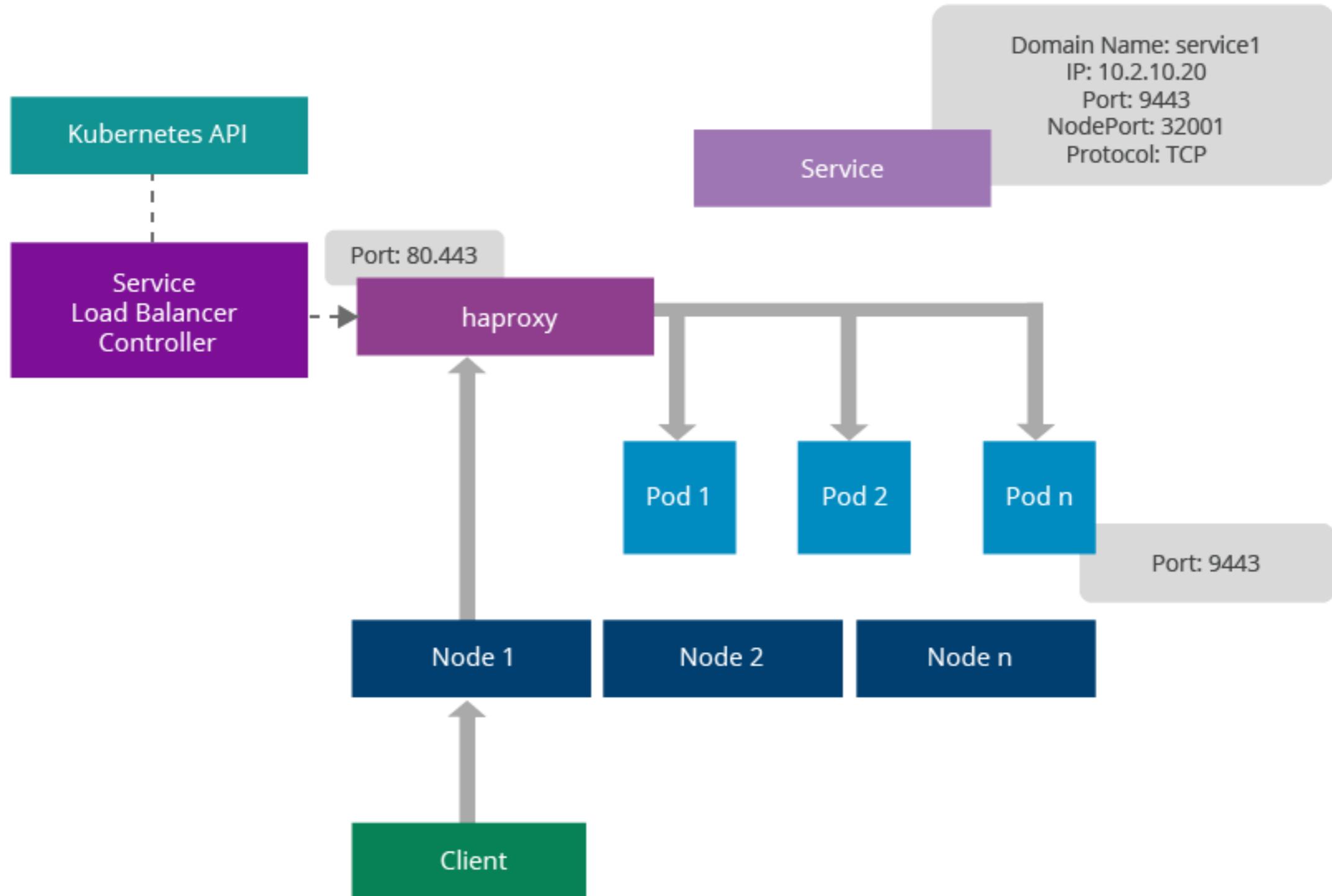
Proxy-mode: ipvs (beta 1.9)



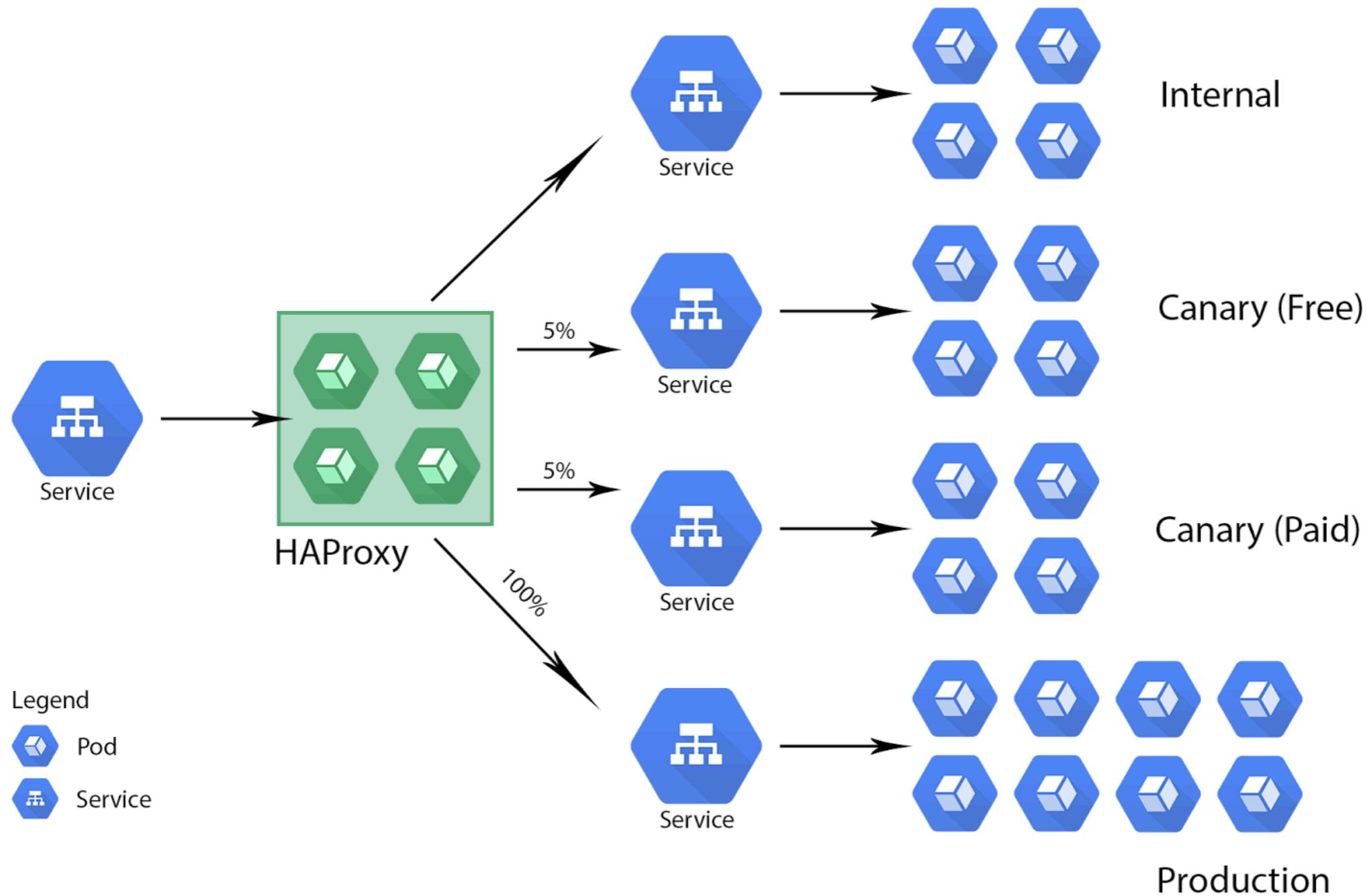
Sticky session management using Service Load Balancers



Service Load Balancer



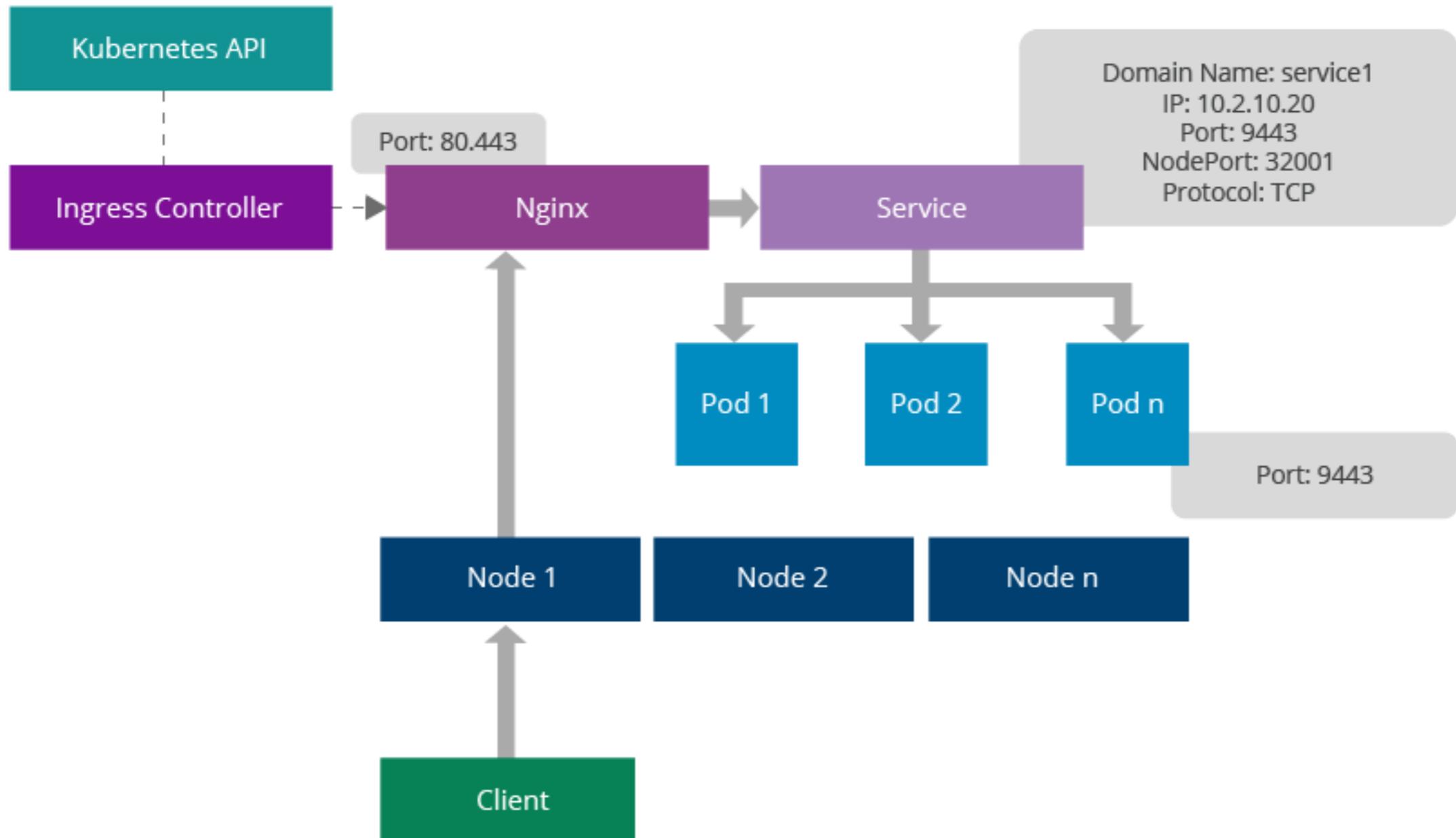
Service Load Balancer



<https://kubernetes.io/blog/2017/04/multi-stage-canary-deployments-with-kubernetes-in-the-cloud-onprem/>



Ingress controller



Secret and configuration management



Secret and configuration mgt

From **12factor** app

Store config in environments
(dev/qa/staging/prod)

<https://12factor.net/>



Secret and configuration mgt

Keep confidential data to running app in
encryption format

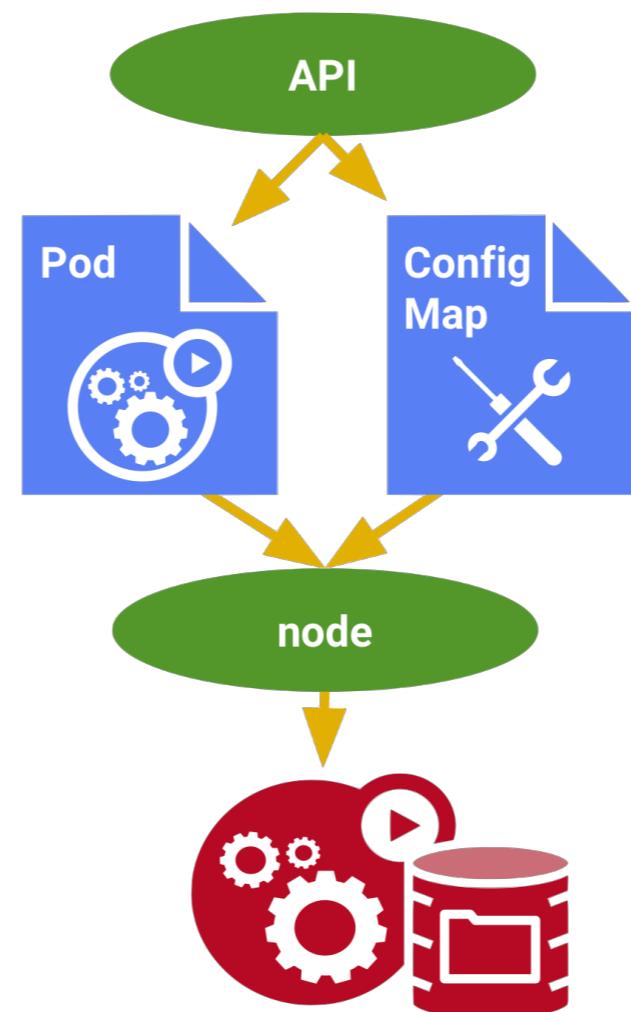
\$kubectl get secret

| 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

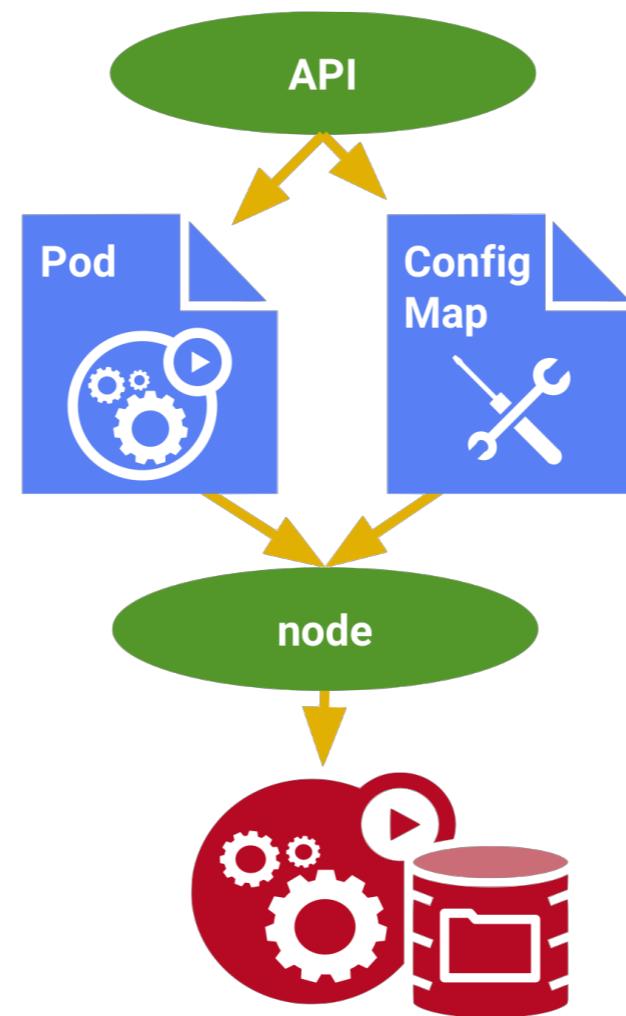


<https://kubernetes.io/docs/tasks/configure-pod-container/configure-pod-configmap/>



Secret and configuration mgt

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



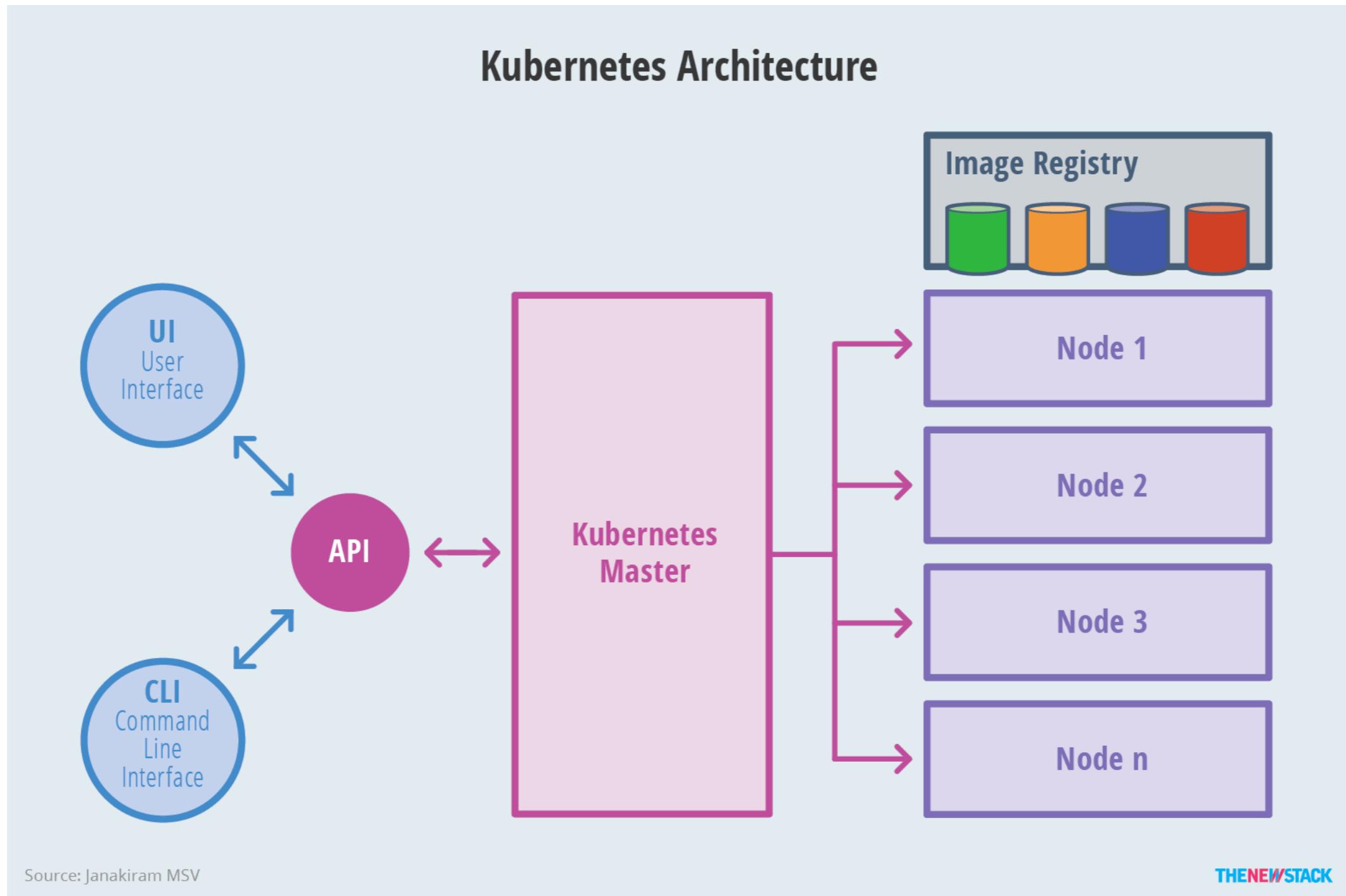
<https://12factor.net/>



System architecture



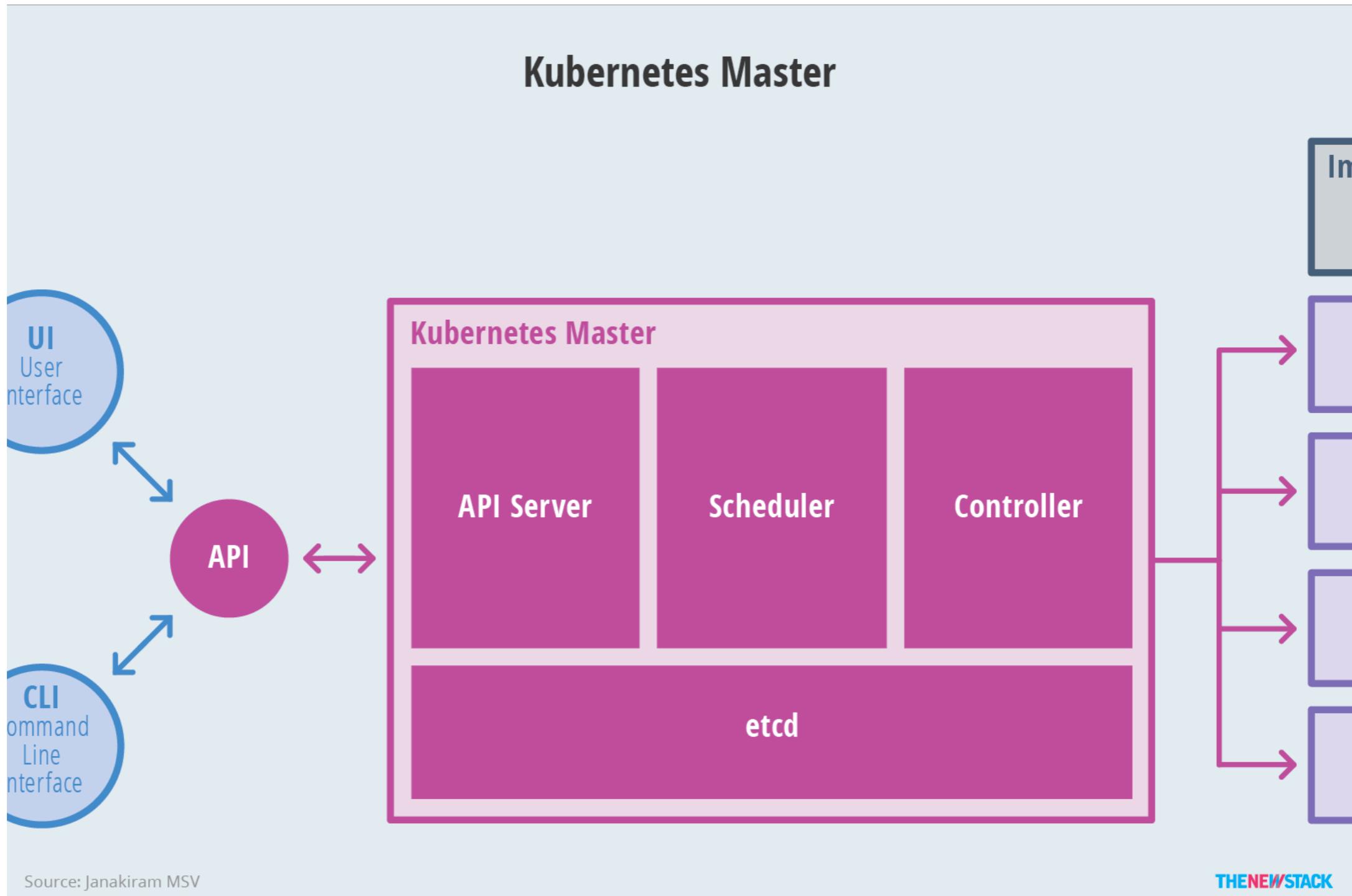
System architecture



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



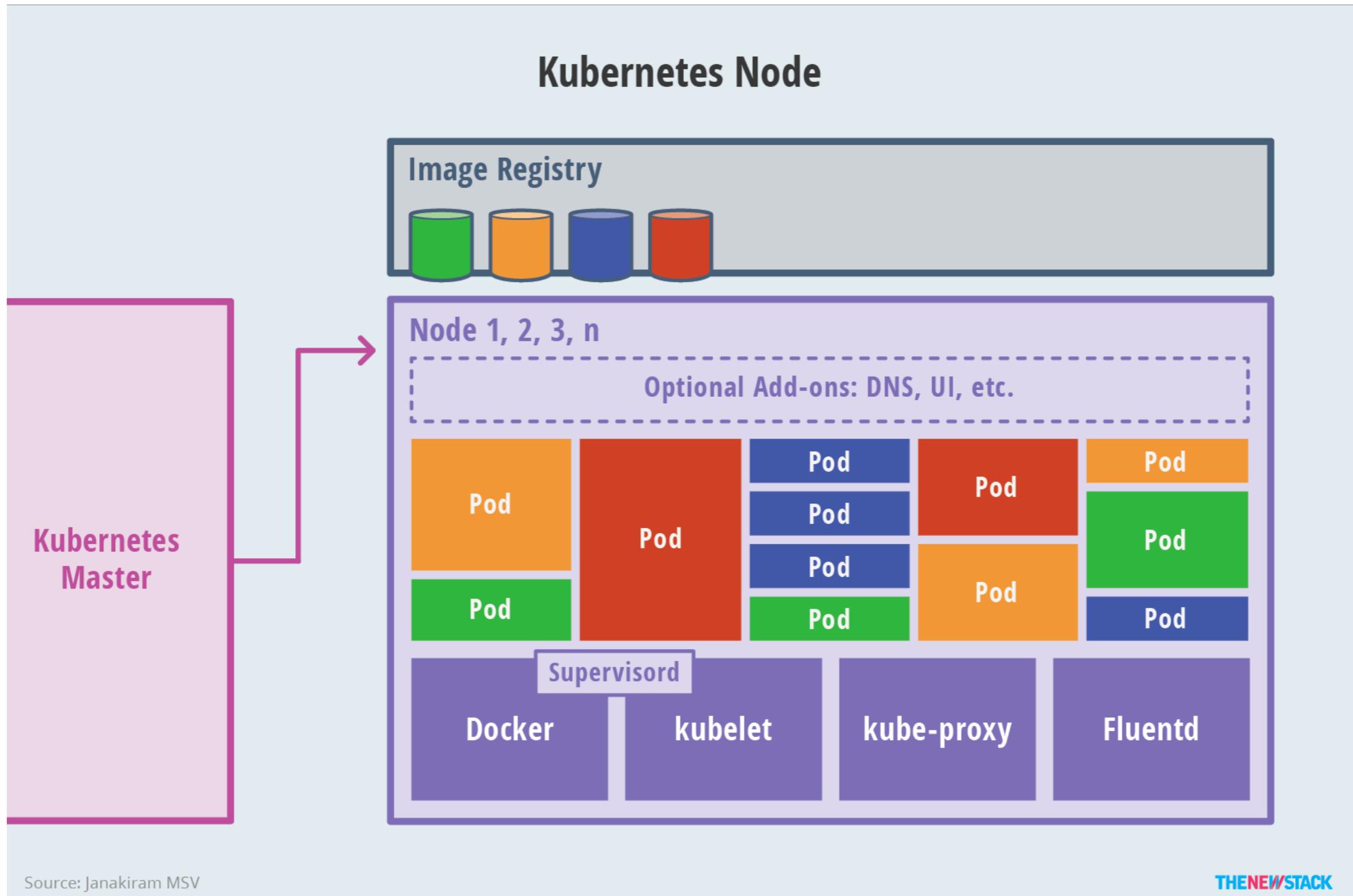
Master



<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

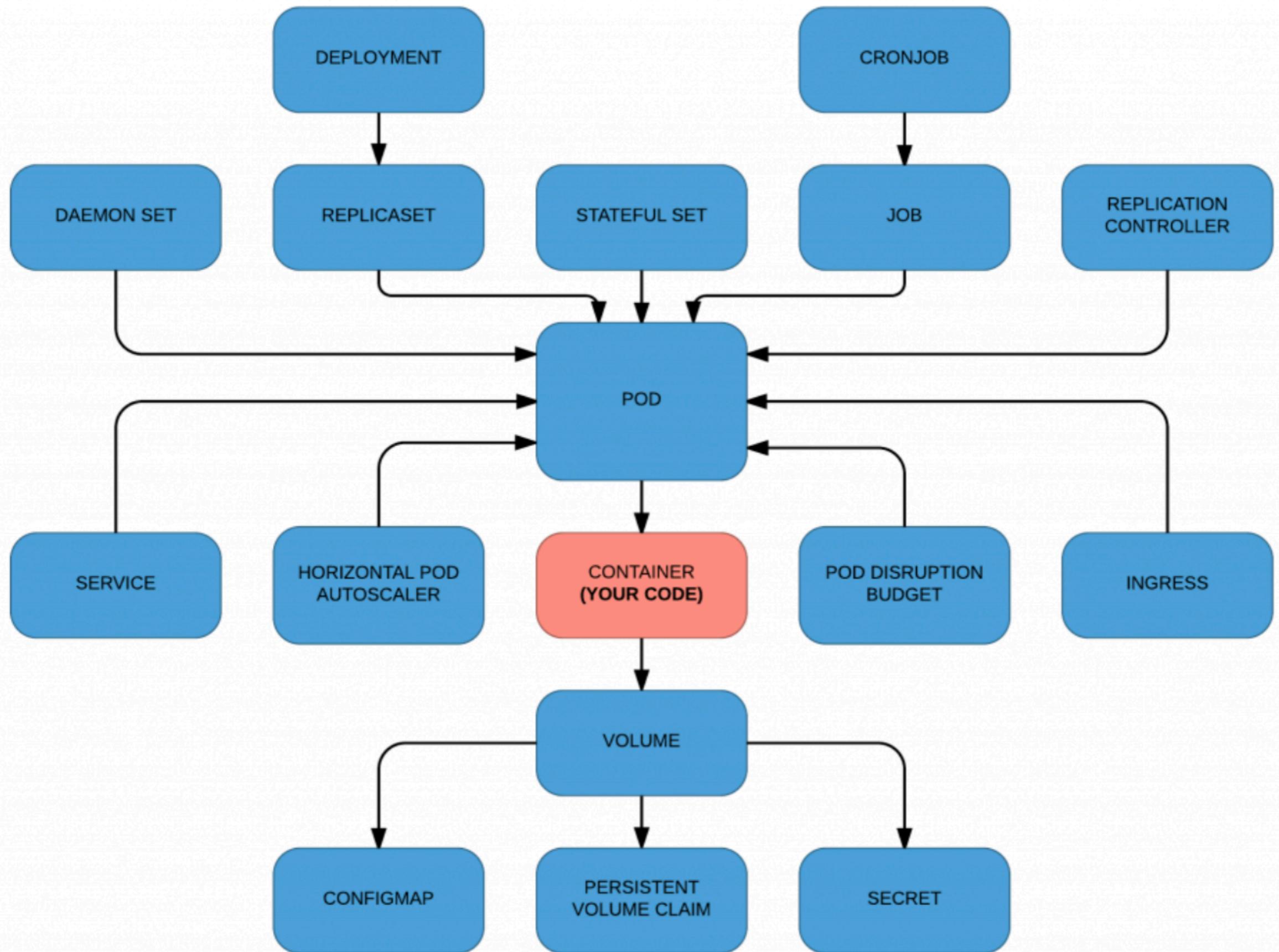
Deployments

ConfigMap and Secret

Volumes

StatefulSets



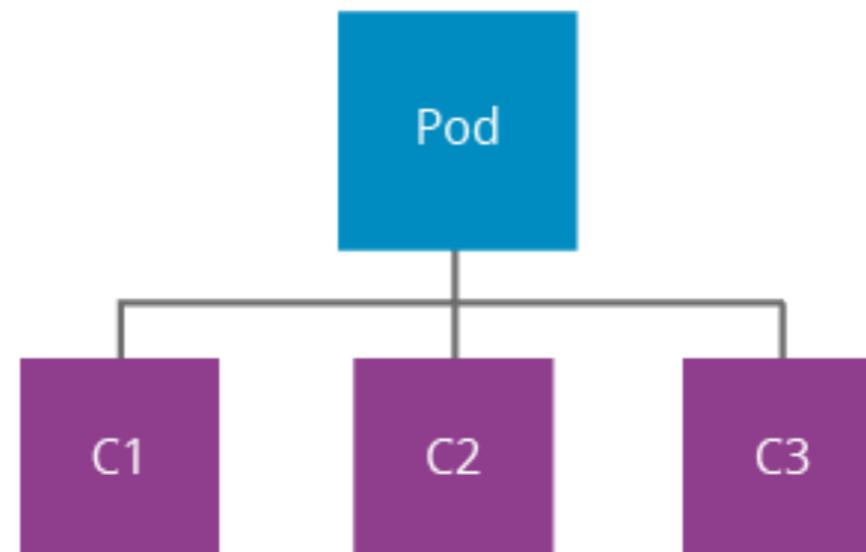


Pods vs containers



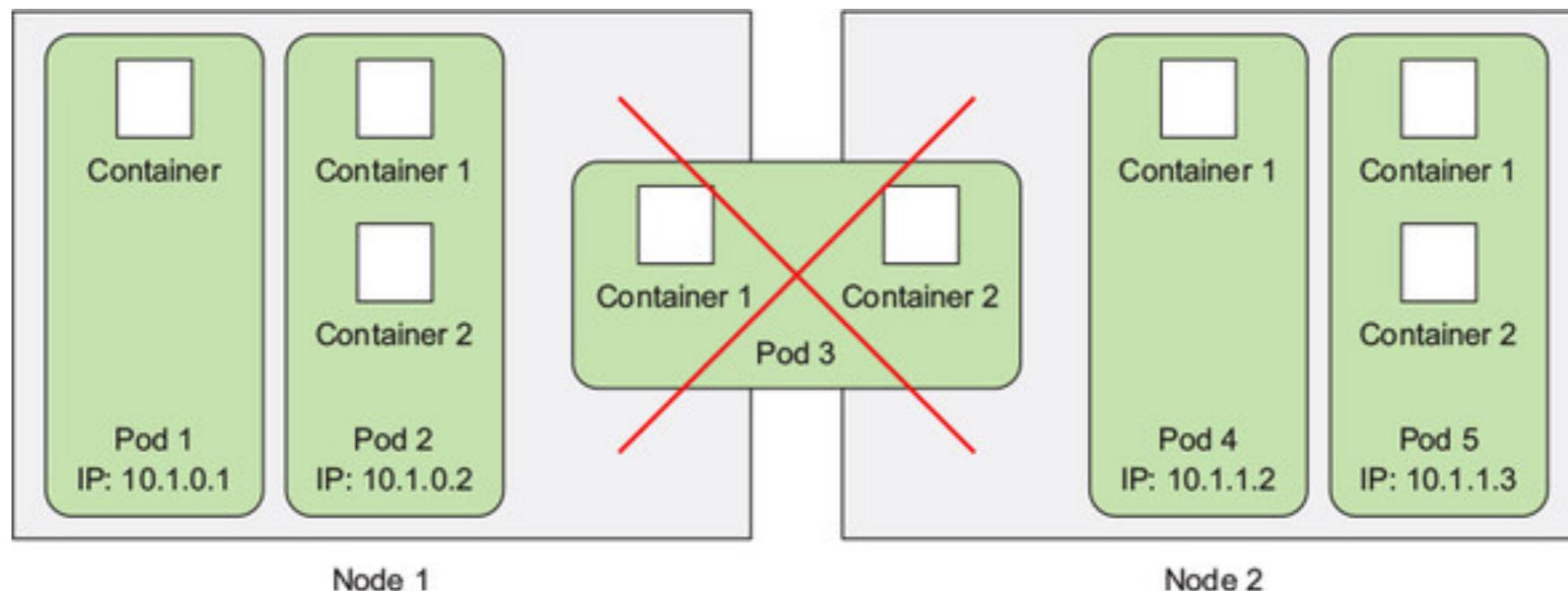
Pods

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

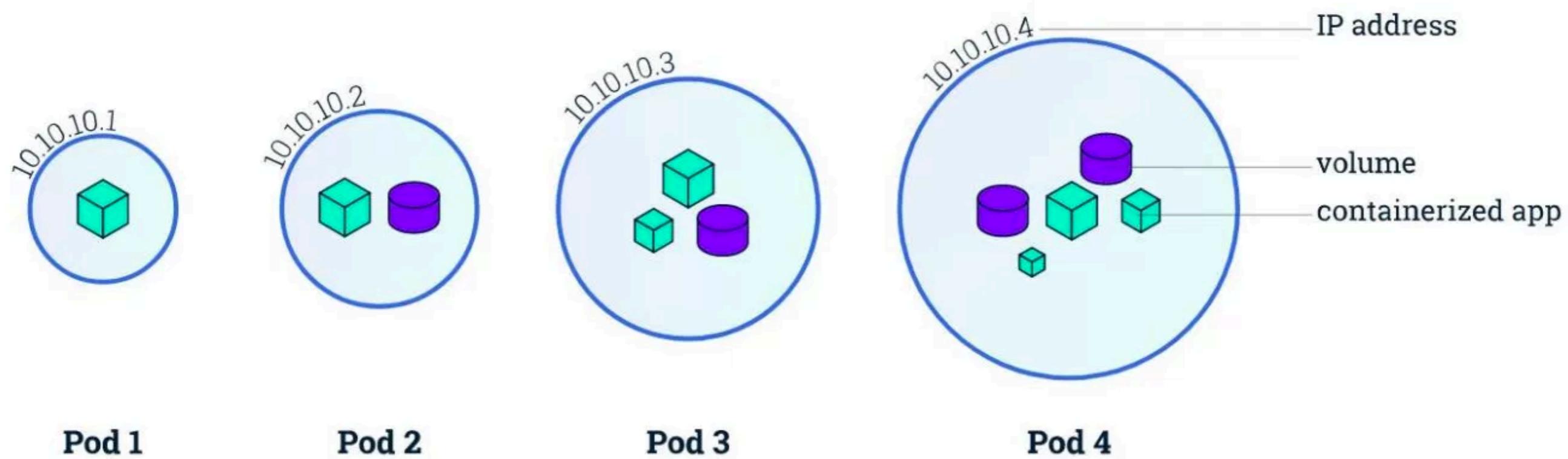
Share network interface

Share Hostname/IP/Port

Share Unix Time Sharing (UTS)



Pods

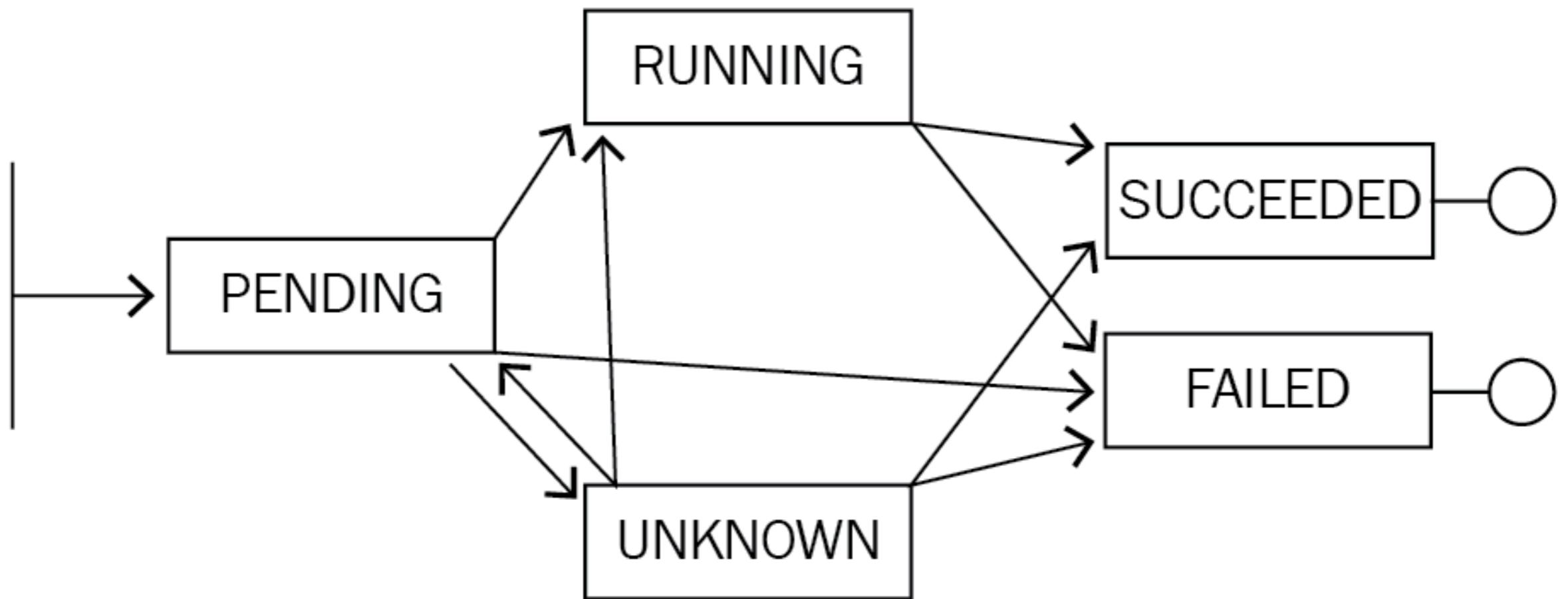


Pods lifecycle

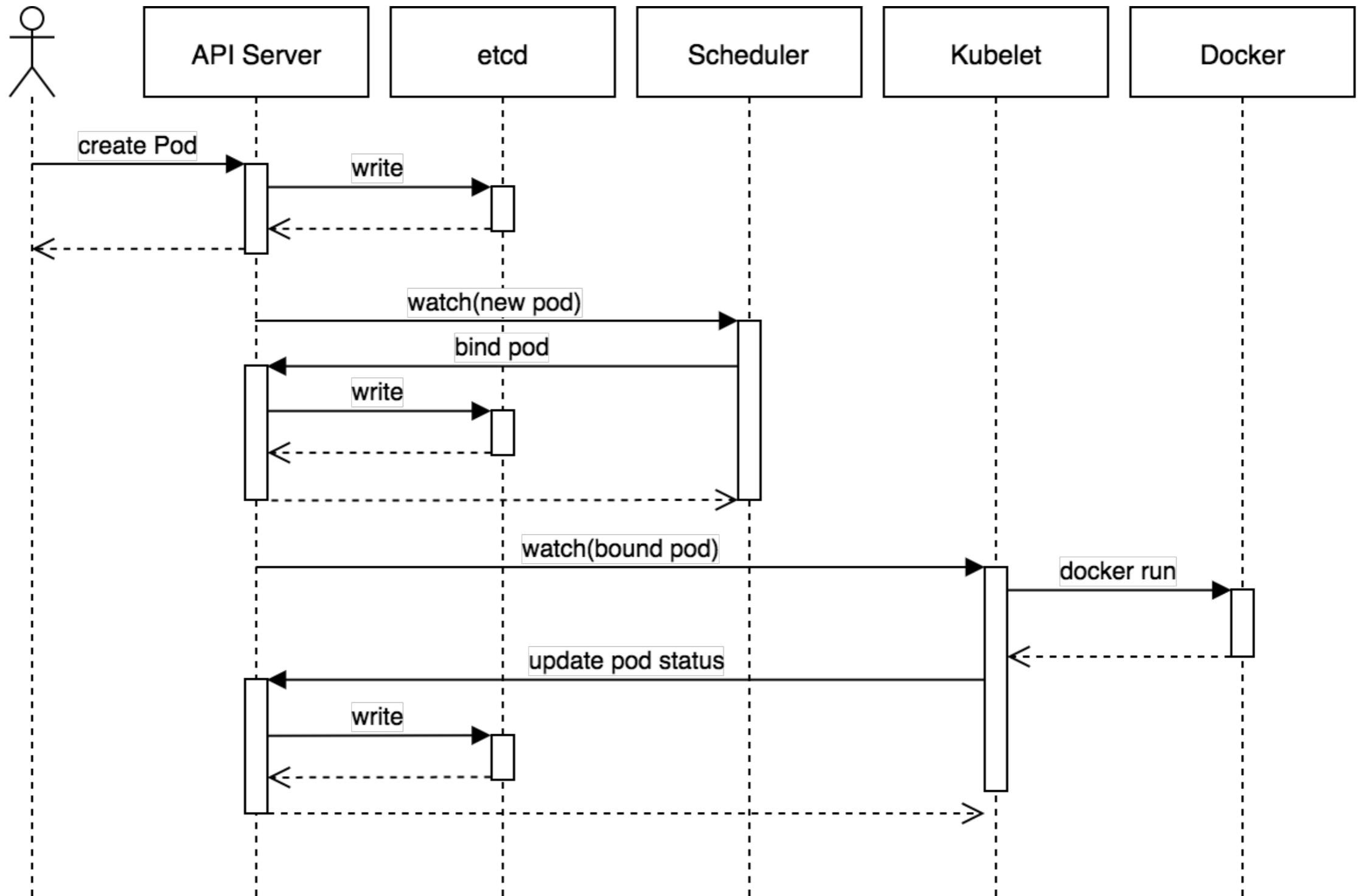
| Phase | Description |
|-----------|---|
| Pending | Accepted by kubernetes but container not created yet |
| Running | Pods bound to the node, all containers created and at least one container is running/start/restarting |
| Successed | Containers exited with status 0 |
| Failed | All containers exit and at least one exited with non-zero status |
| Unknow | State of Pods can't be determined due to communication issues with its node |



Pods lifecycle



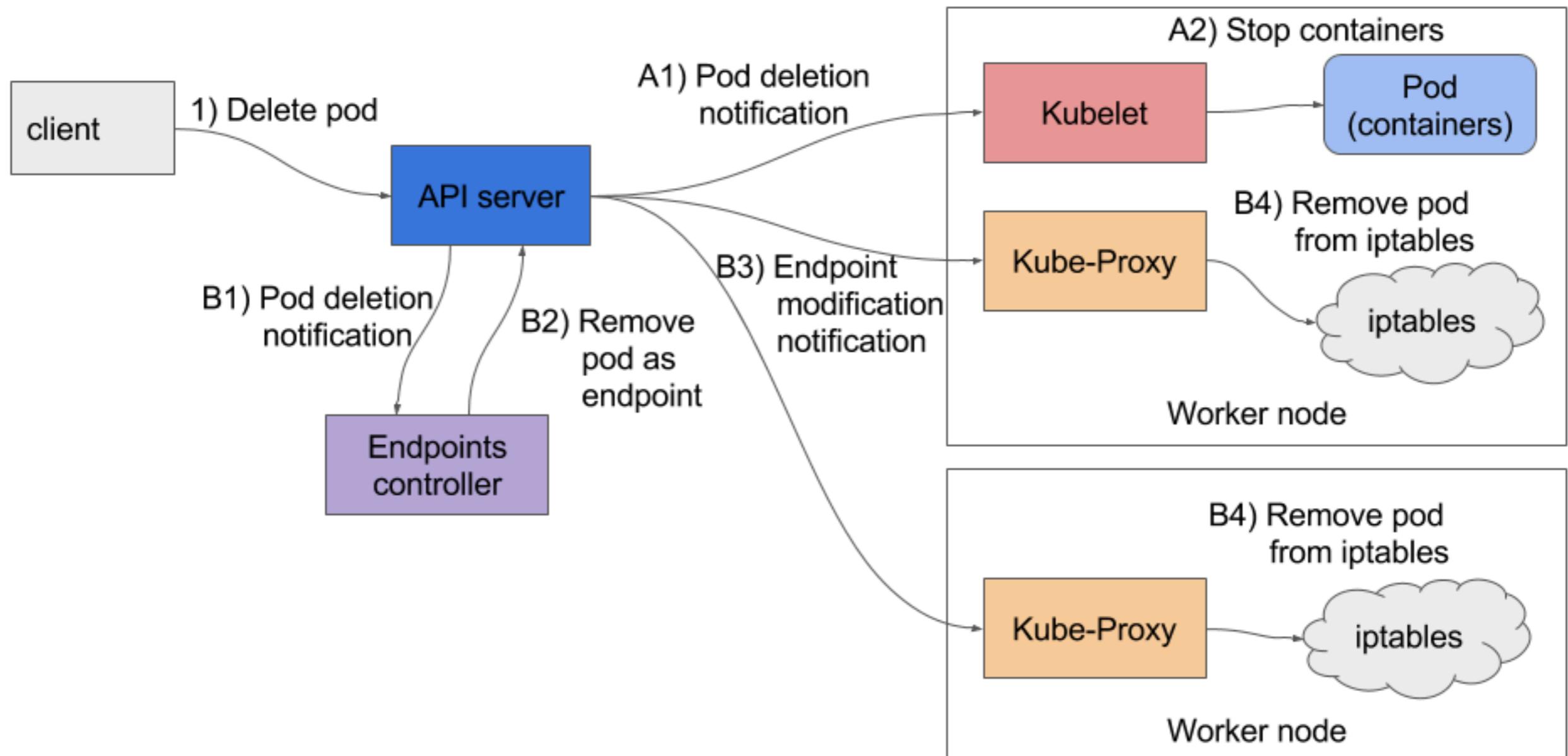
How to create Pods ?



<https://blog.heptio.com/core-kubernetes-jazz-improv-over-orchestration-a7903ea92ca>



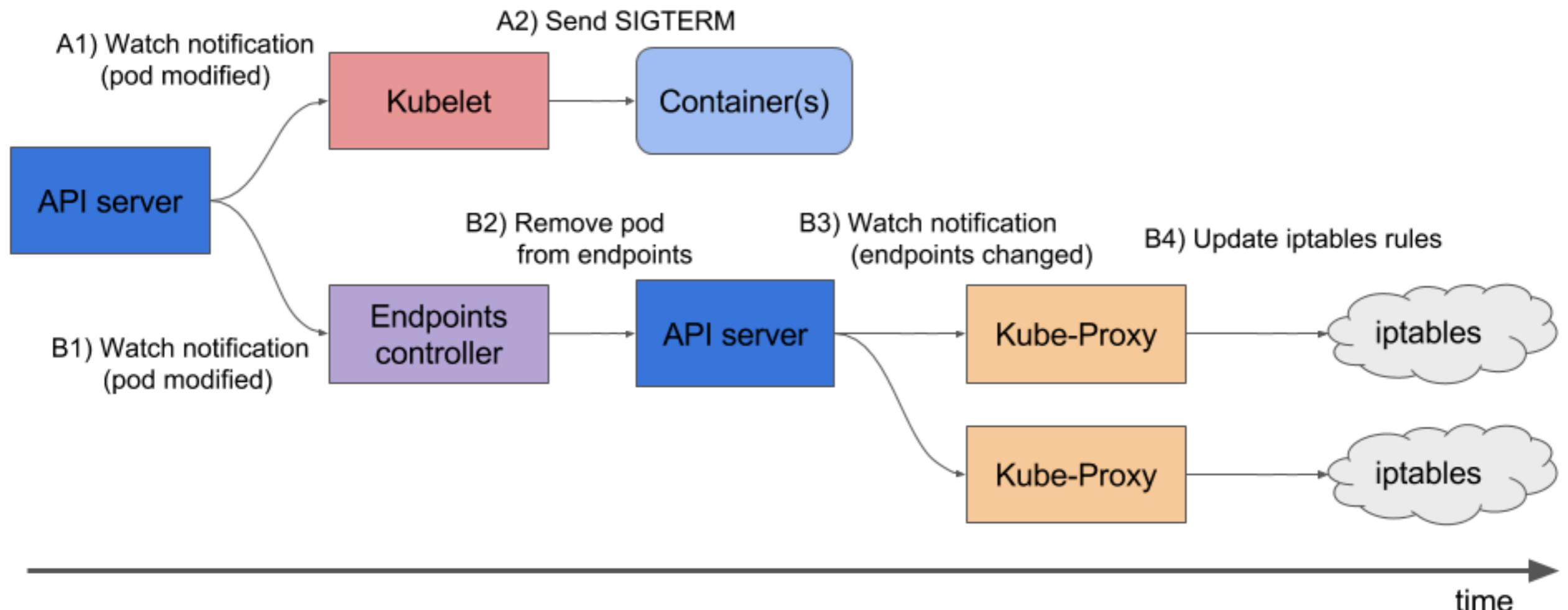
How to delete Pods ?



<https://freecontent.manning.com/handling-client-requests-properly-with-kubernetes/>



How to delete Pods ?

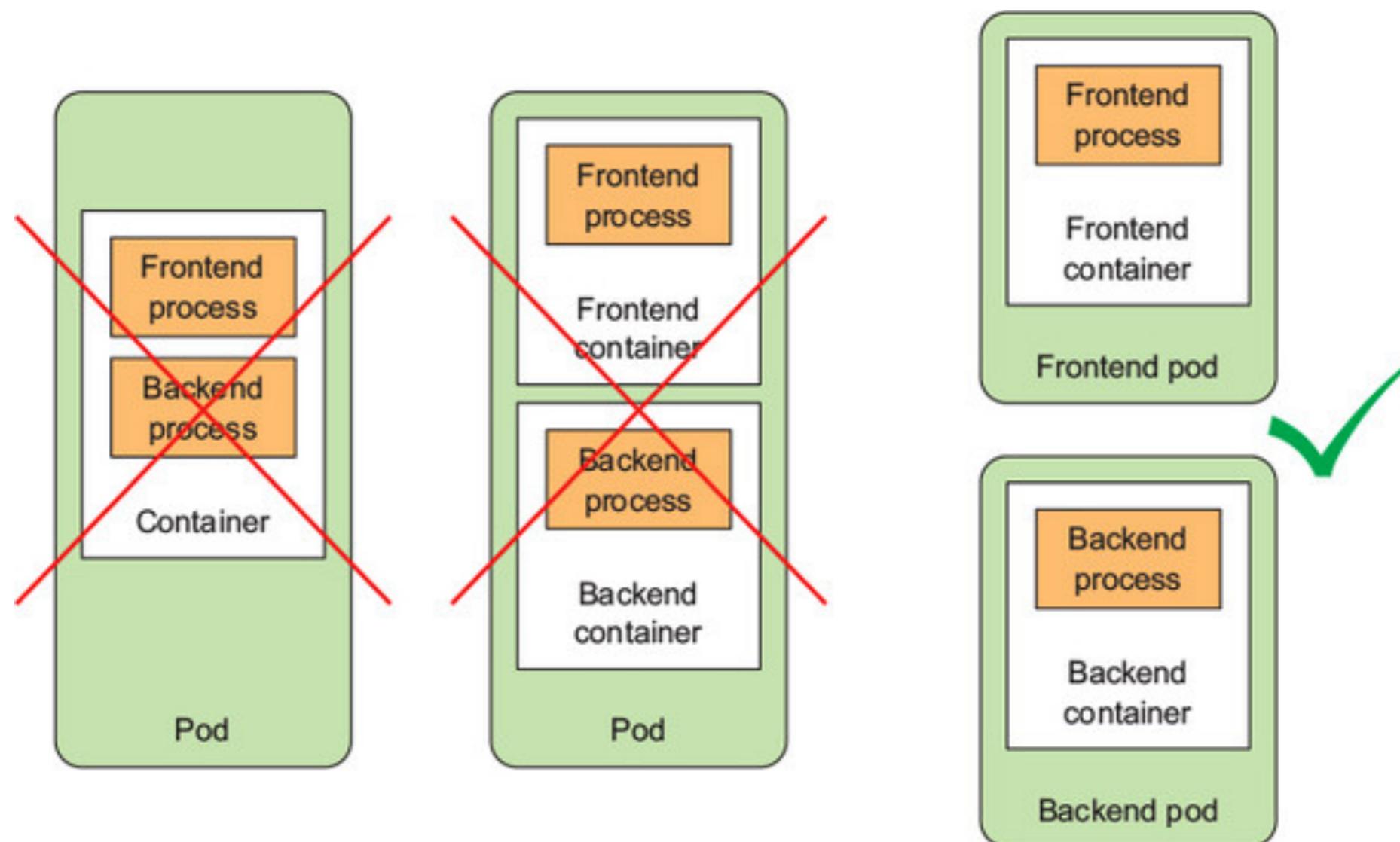


<https://freecontent.manning.com/handling-client-requests-properly-with-kubernetes/>



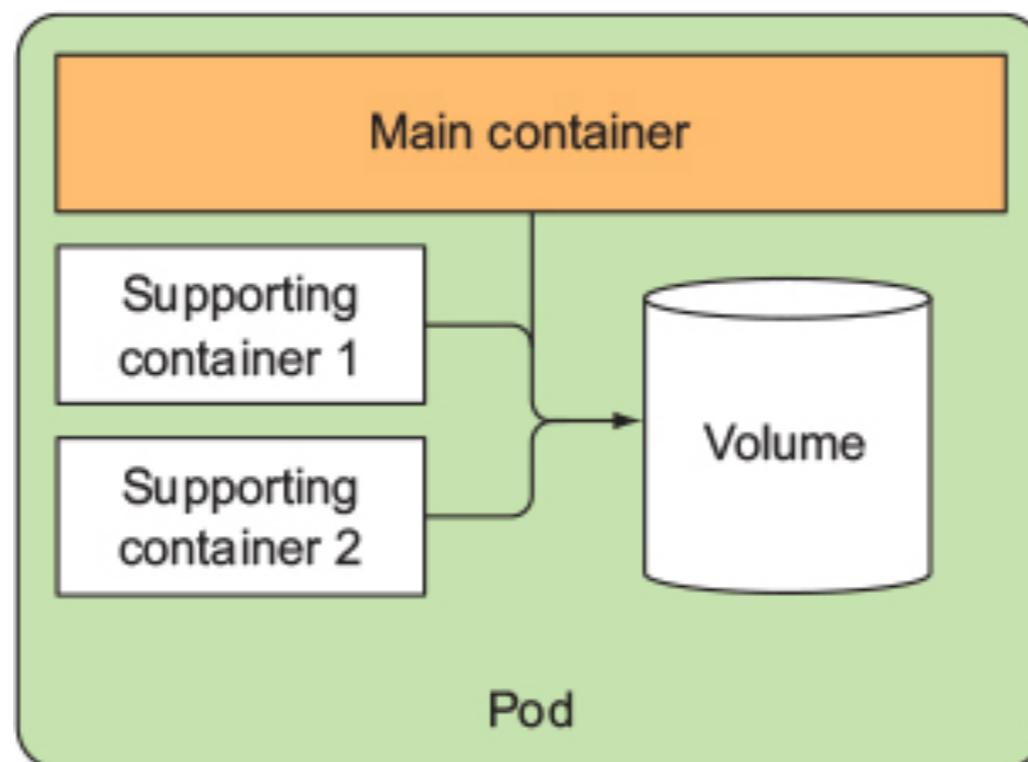
Organize container across Pods

Split multi-tiers app into multiple pods



Organize container across Pods

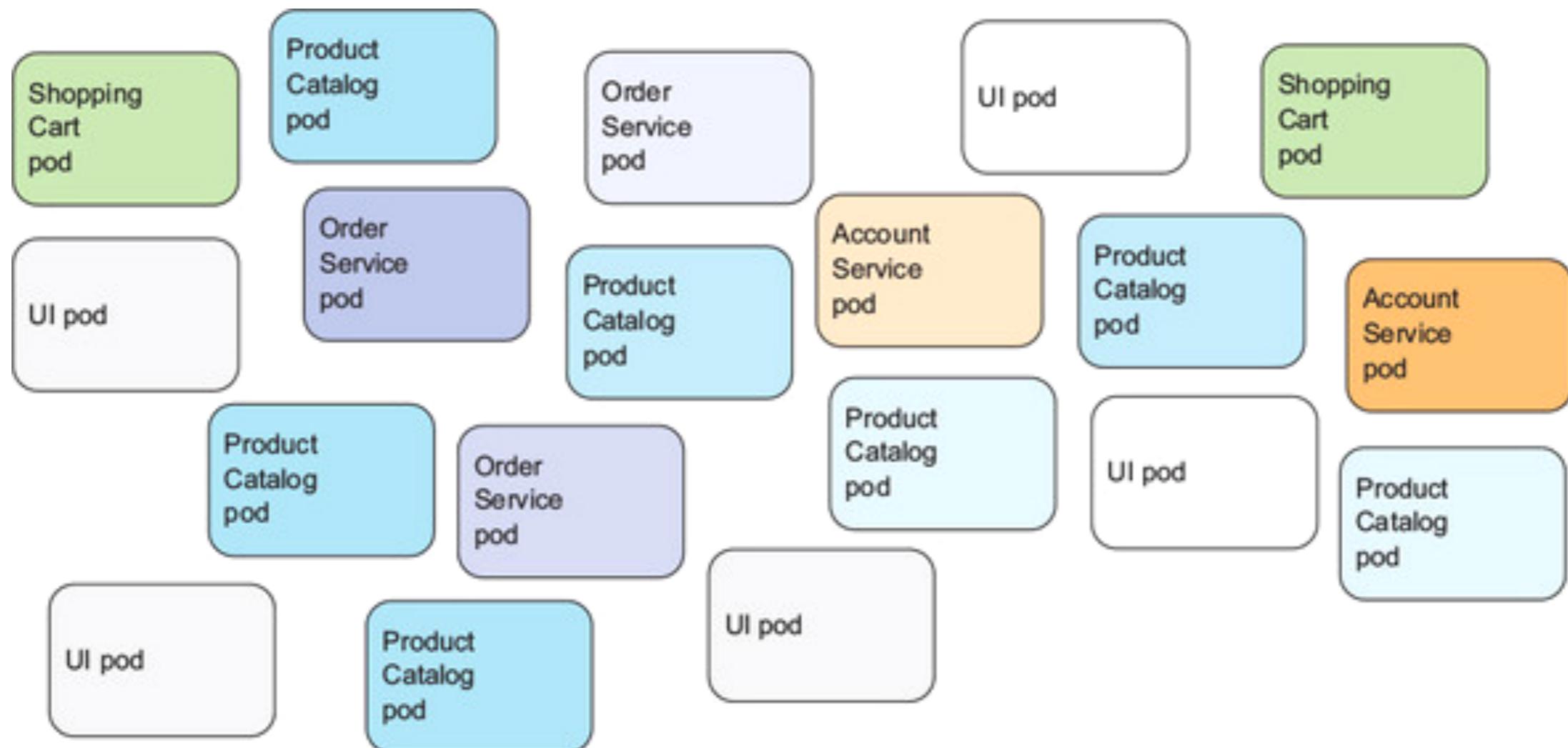
When to use multiple containers in a pods



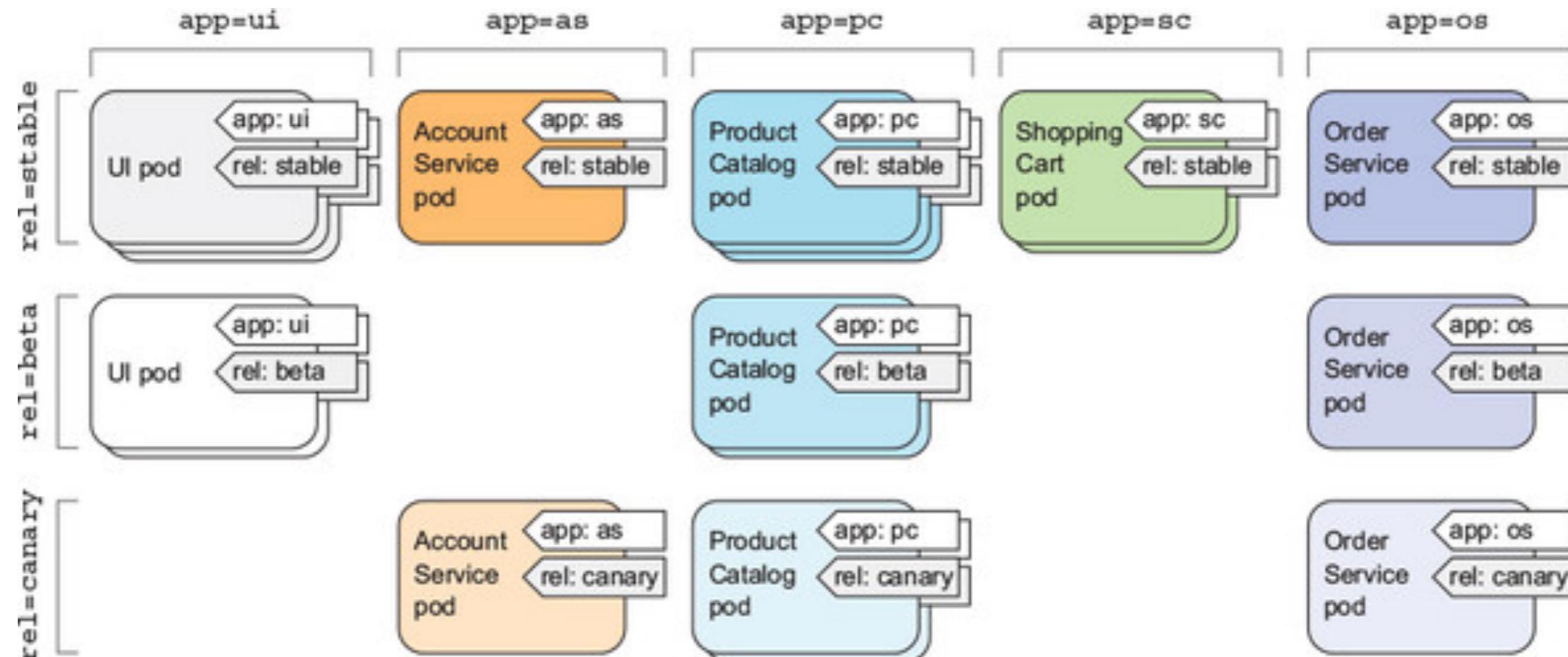
Organize pods with Labels



Organize pods with Labels

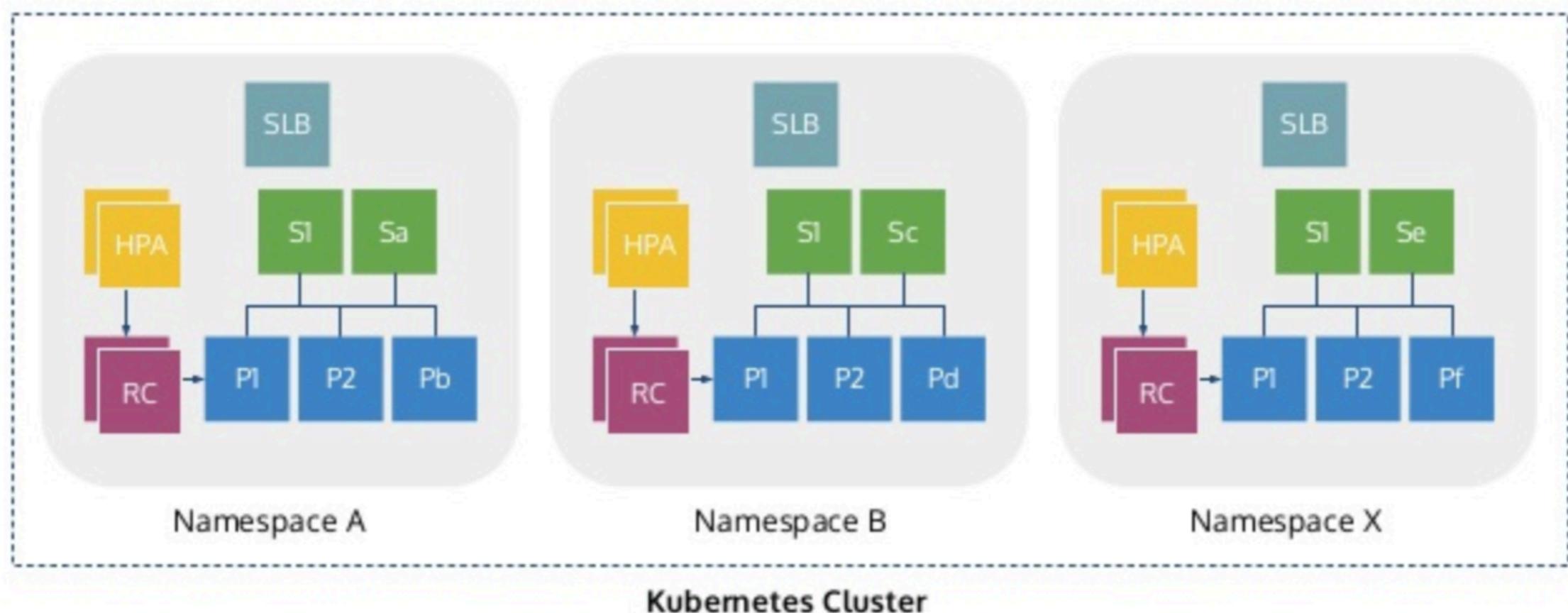


Organize pods with Labels



Pods namespaces

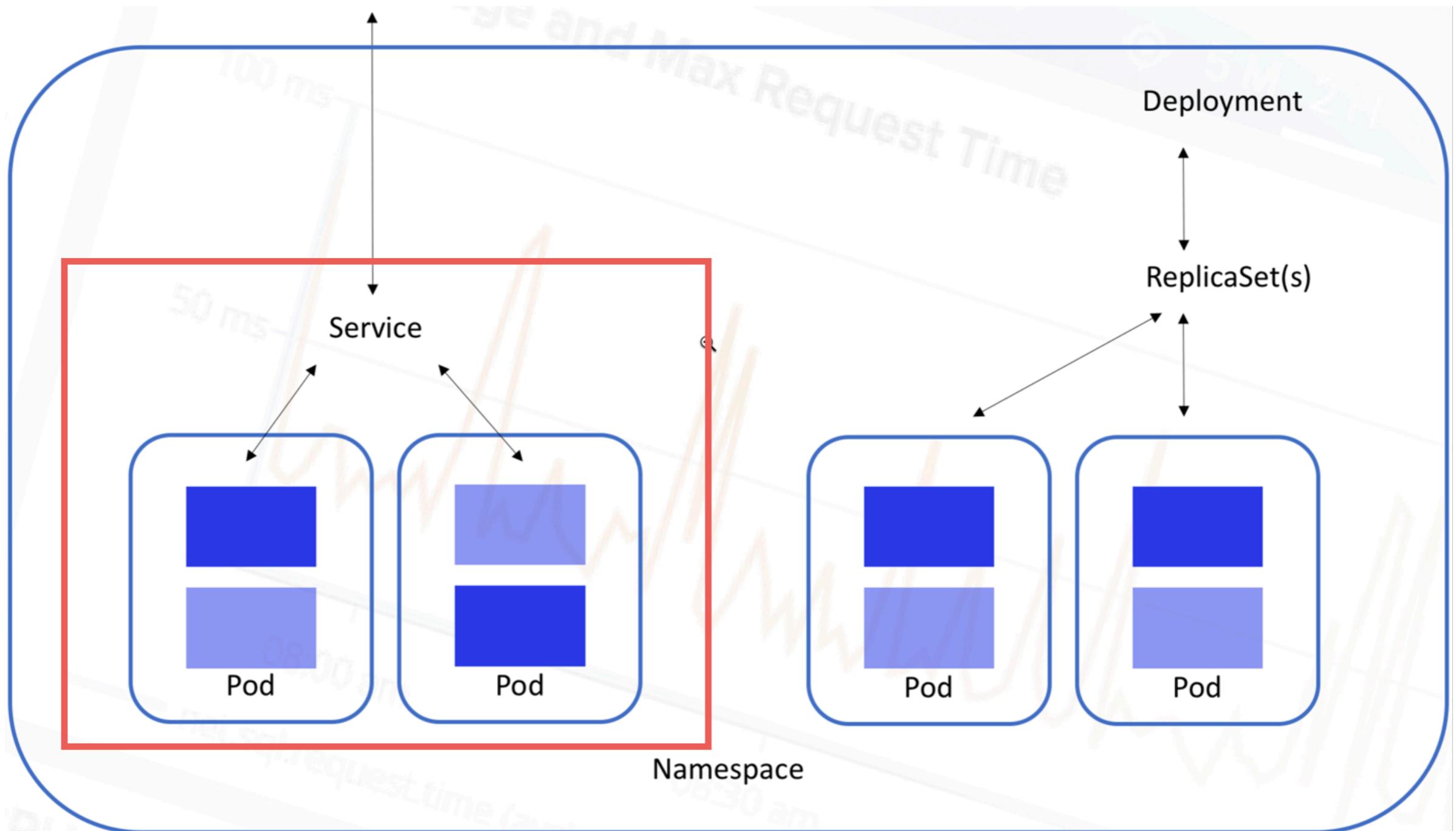
Allow different teams to use the same cluster



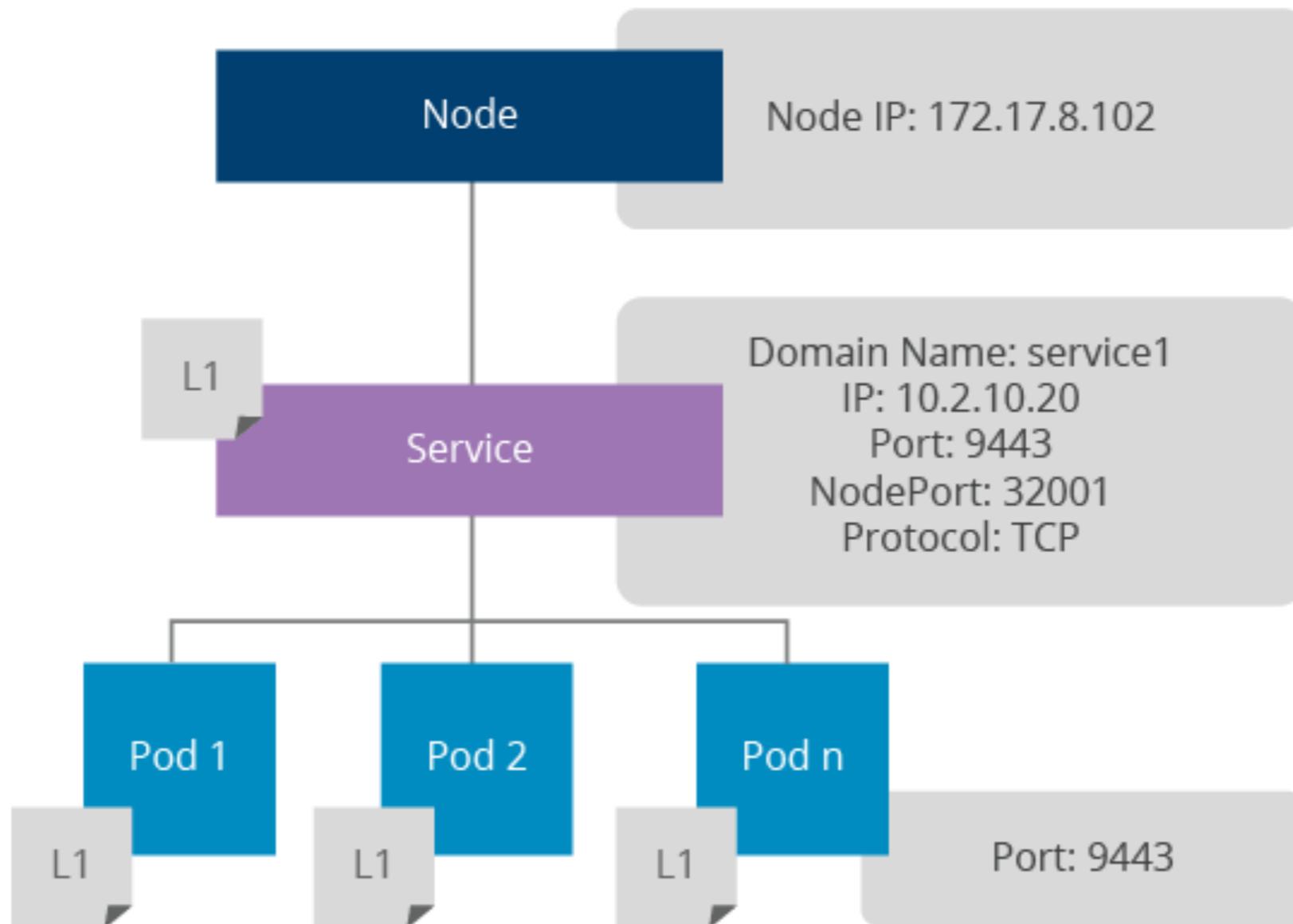
Services discovery and Load balancing



Services



Services

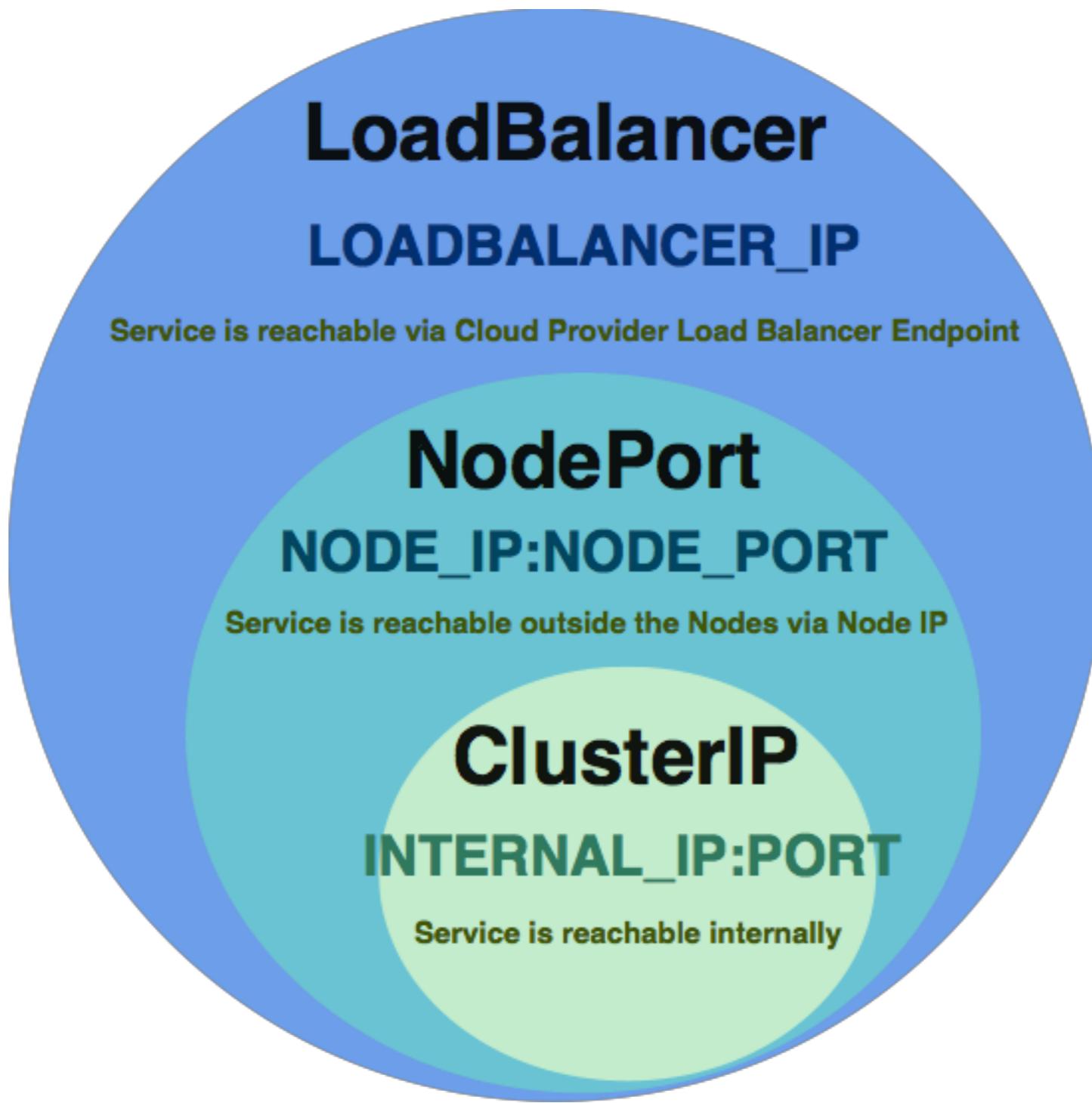


Services

- 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



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



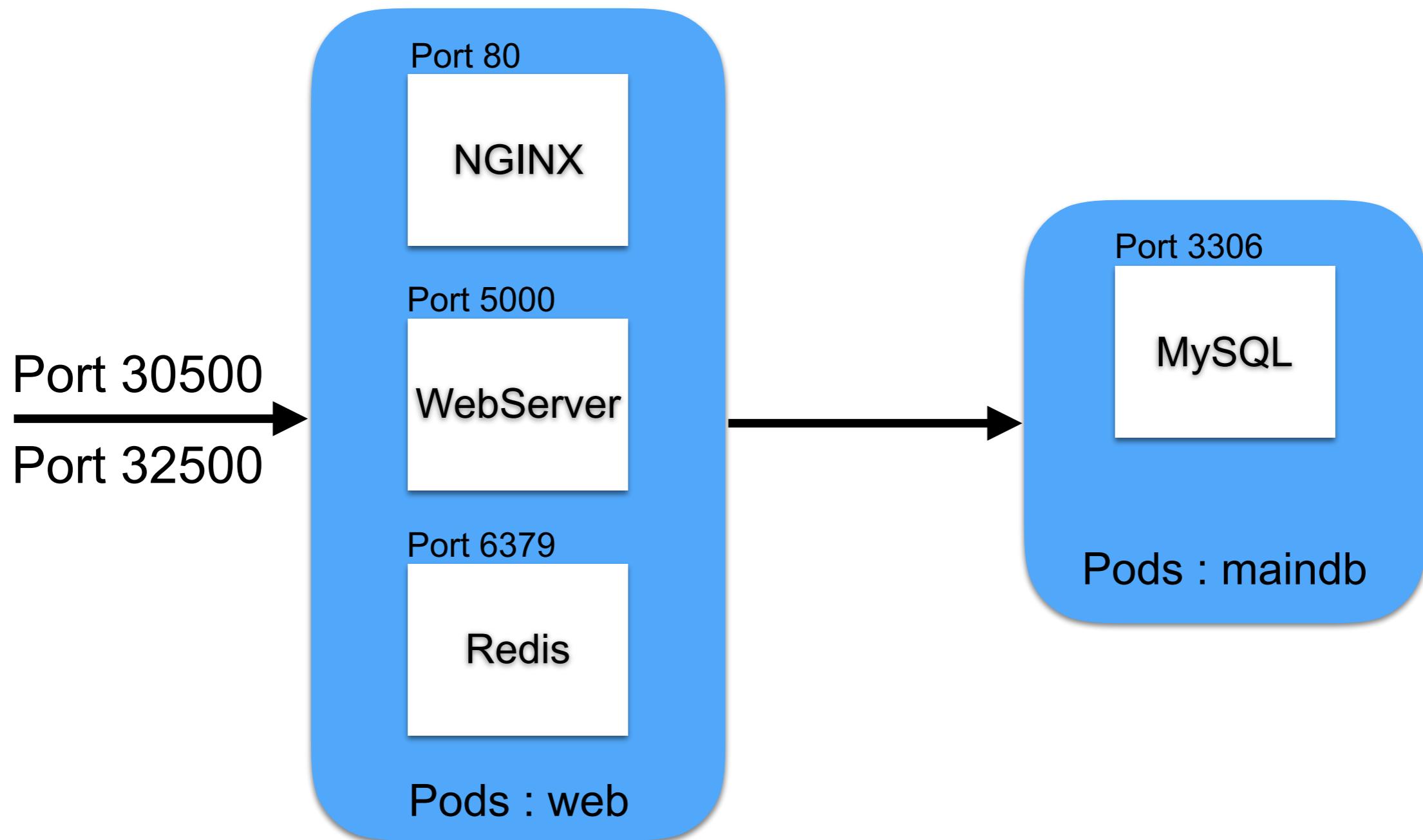
Workshop :: Pods & Services

N container per Pods

file /02-pod-service/multiple-pod/
workshop_instruction.txt



Workshop



Workshop :: Expose port of Pods to outside cluster

file /02-pod-service/expose/expose-pod-service.txt



Expose Pods to outside cluster

HostNetwork

HostPort

NodePort of service

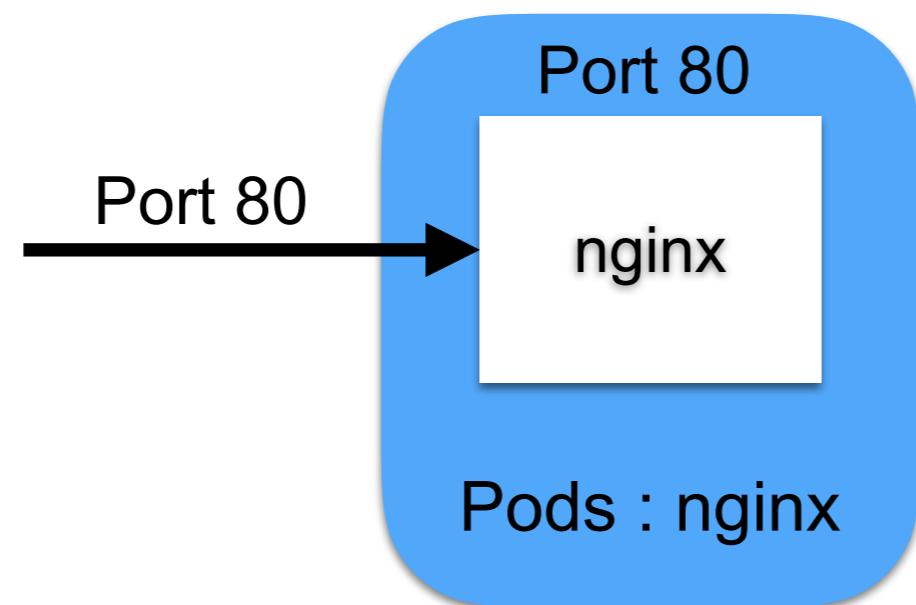
LoadBalance of service

Ingress controller



HostNetwork

Applications running in such a pod can directly see the network interfaces of the host machine where the pod was started.



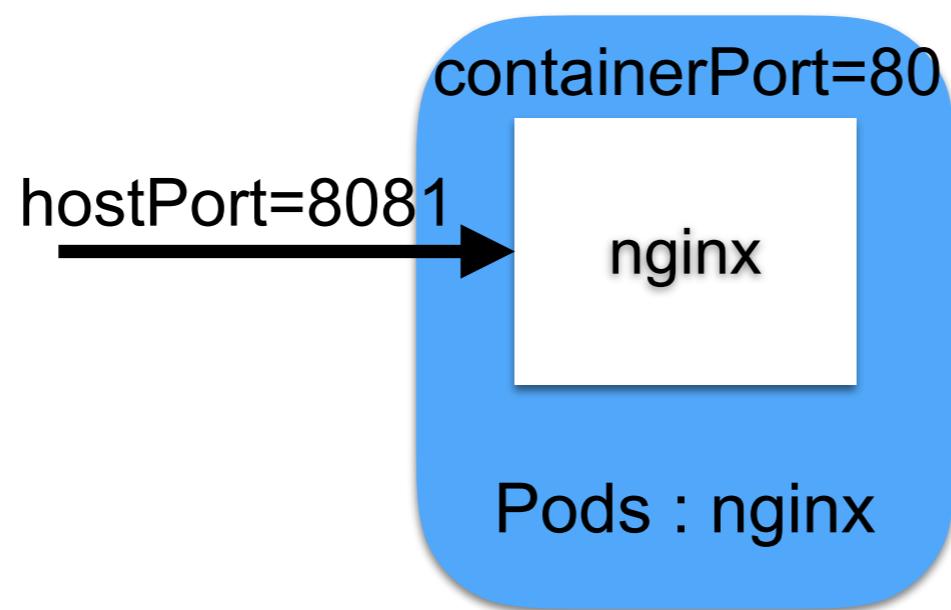
HostNetwork

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  hostNetwork: true
  containers:
    - name: nginx
      image: nginx
```



HostPort

The container port will be exposed to the external network at **<hostIP>:<hostPort>**, where the **hostIP** is the IP address of the Kubernetes node where the container is running and the **hostPort** is the port requested by the user.



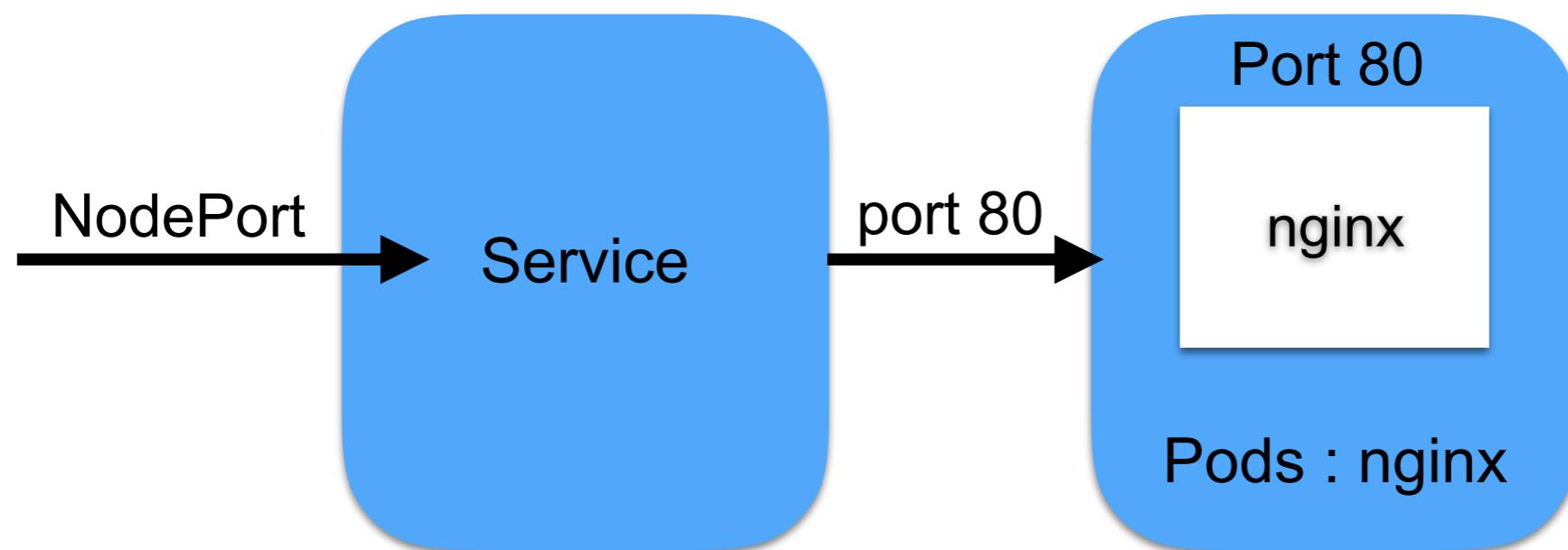
HostPort

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
    - name: nginx
      image: nginx
      ports:
        - containerPort: 80
          hostPort: 8081
```



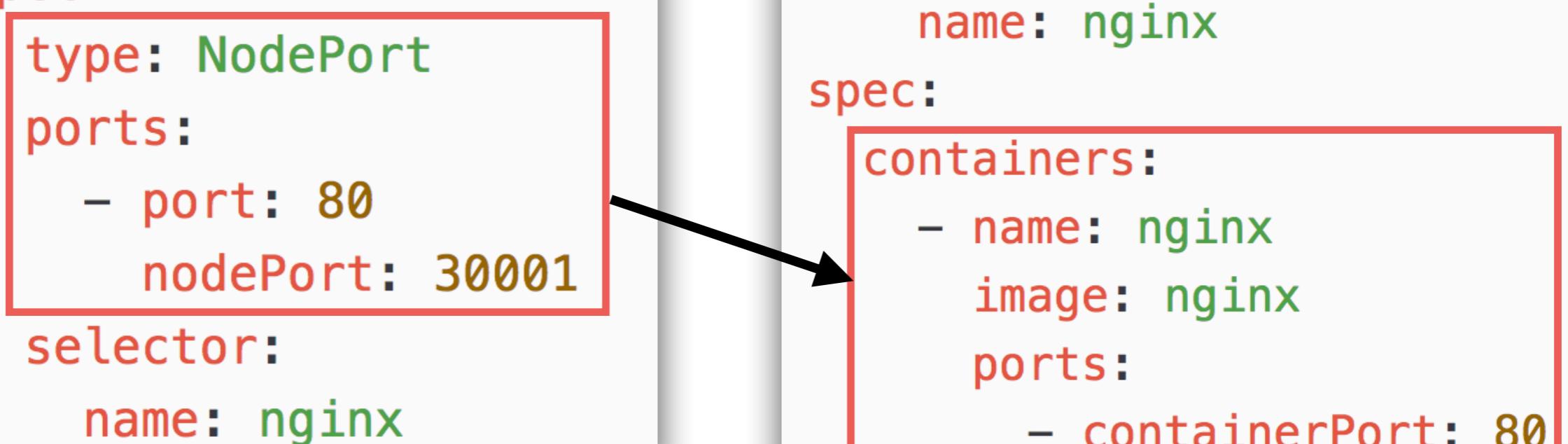
NodePort

When creating a NodePort service, the user can specify a port from the range **30000-32767**, and each Kubernetes node will proxy that port to the pods selected by the service.



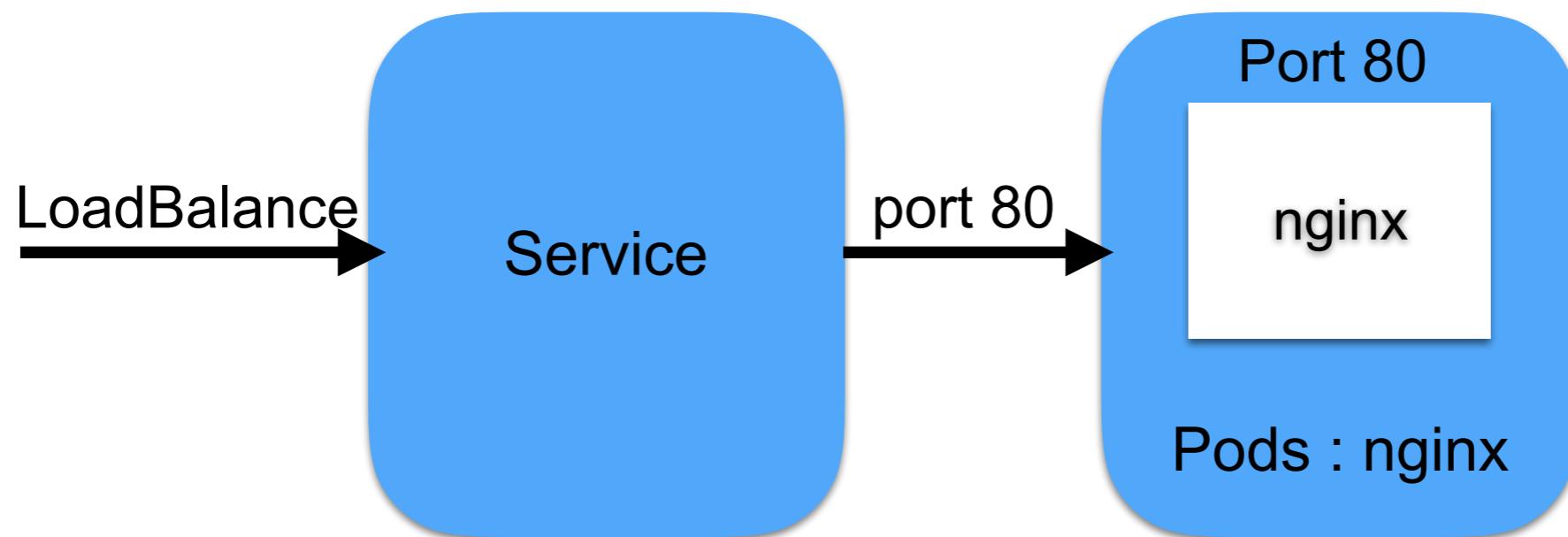
NodePort

```
kind: Service
apiVersion: v1
metadata:
  name: nginx
spec:
  type: NodePort
  ports:
    - port: 80
      nodePort: 30001
  selector:
    name: nginx
apiVersion: v1
kind: Pod
metadata:
  name: nginx
  labels:
    name: nginx
spec:
  containers:
    - name: nginx
      image: nginx
      ports:
        - containerPort: 80
```



LoadBalancer

a cloud provider has to be enabled in the configuration of the Kubernetes cluster. As of version 1.6, Kubernetes can provision load balancers on AWS, Azure, CloudStack, GCE and OpenStack.



LoadBalancer

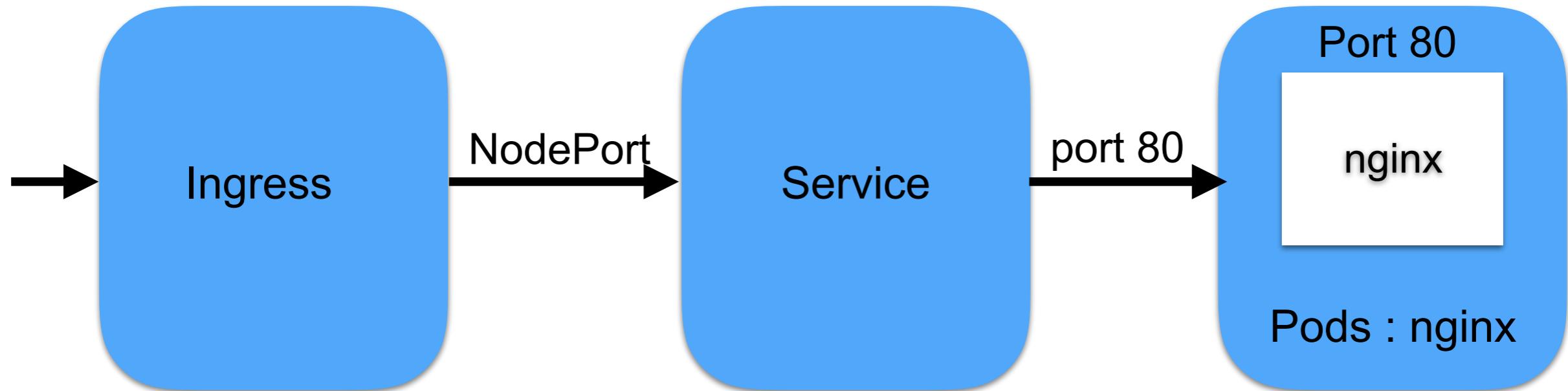
```
kind: Service
apiVersion: v1
metadata:
  name: nginx
spec:
  type: LoadBalancer
  ports:
    - port: 80
selector:
  name: nginx
```

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
  labels:
    name: nginx
spec:
  containers:
    - name: nginx
      image: nginx
      ports:
        - containerPort: 80
```



Ingress controller

The Ingress controller is deployed as a Docker container on top of Kubernetes. Its Docker image contains a load balancer like nginx or HAProxy and a controller daemon.



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



Ingress controller

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: nginx
spec:
  rules:
    - host: nginx.kube.example.com
      http:
        paths:
          - web:
              serviceName: nginx
              servicePort: 80
```

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
  labels:
    name: nginx
spec:
  containers:
    - name: nginx
      image: nginx
      ports:
        - containerPort: 80
```



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



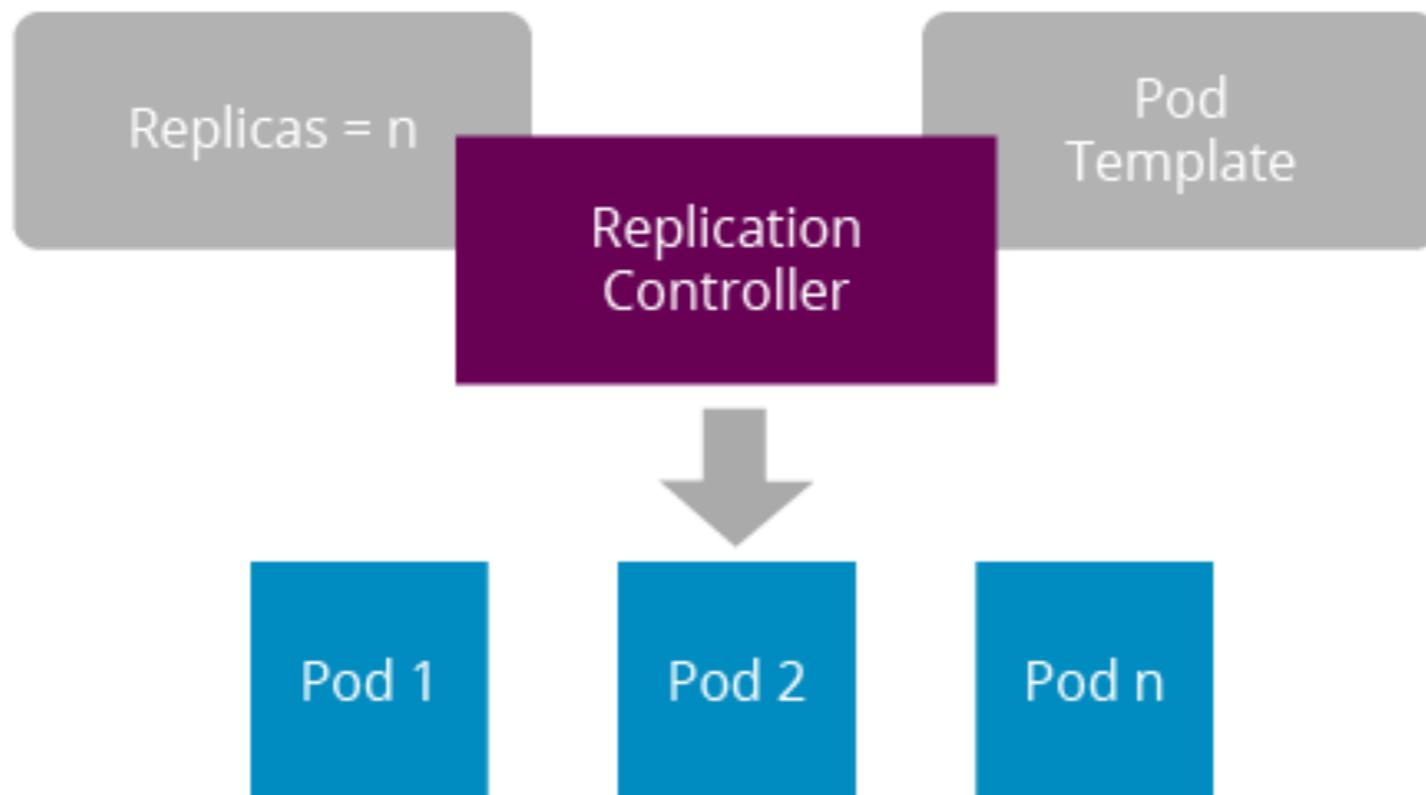
Replication Controller

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 |

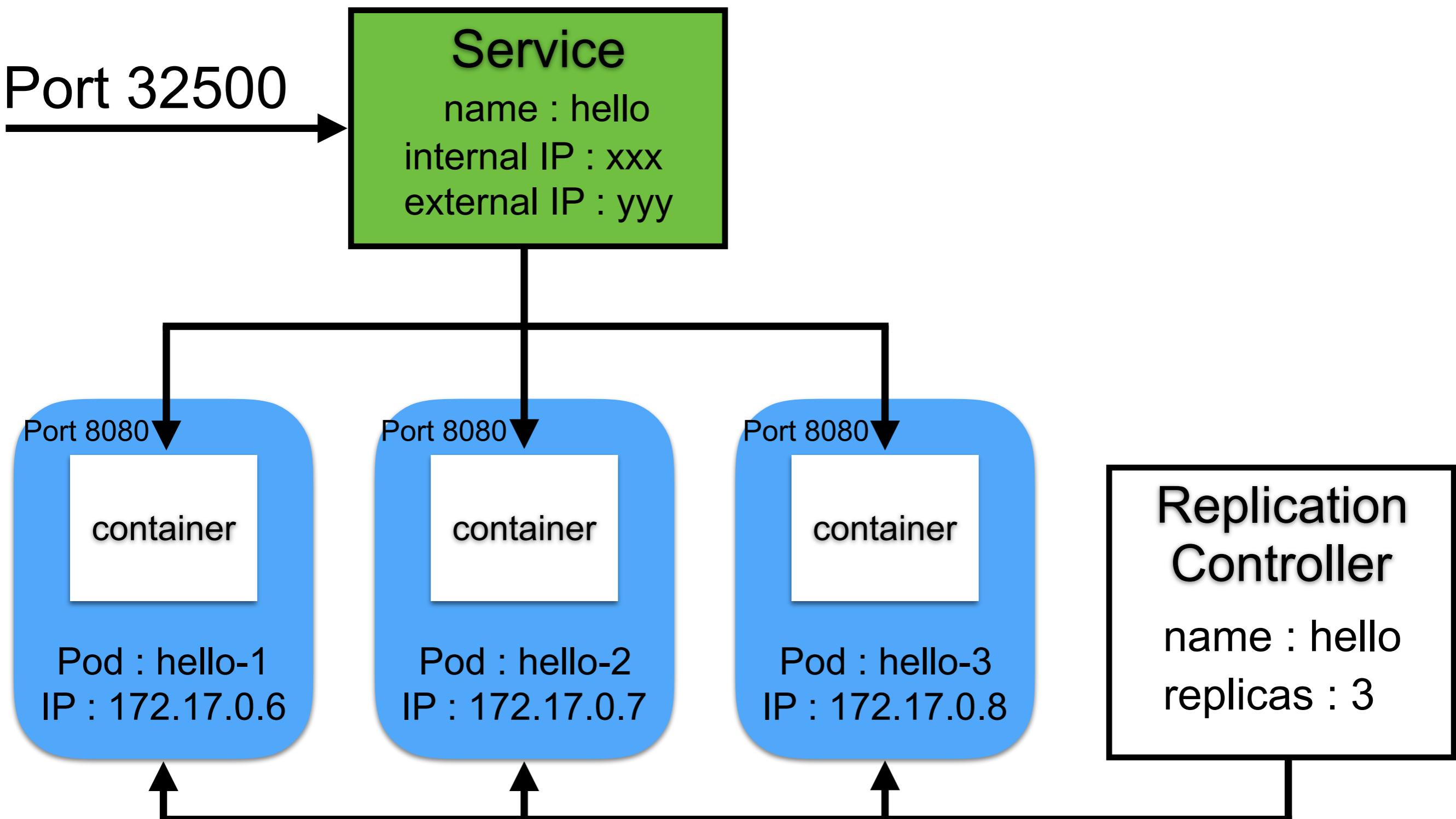


Workshop Replication Controller

file /03-replication-controller



Scaling the application

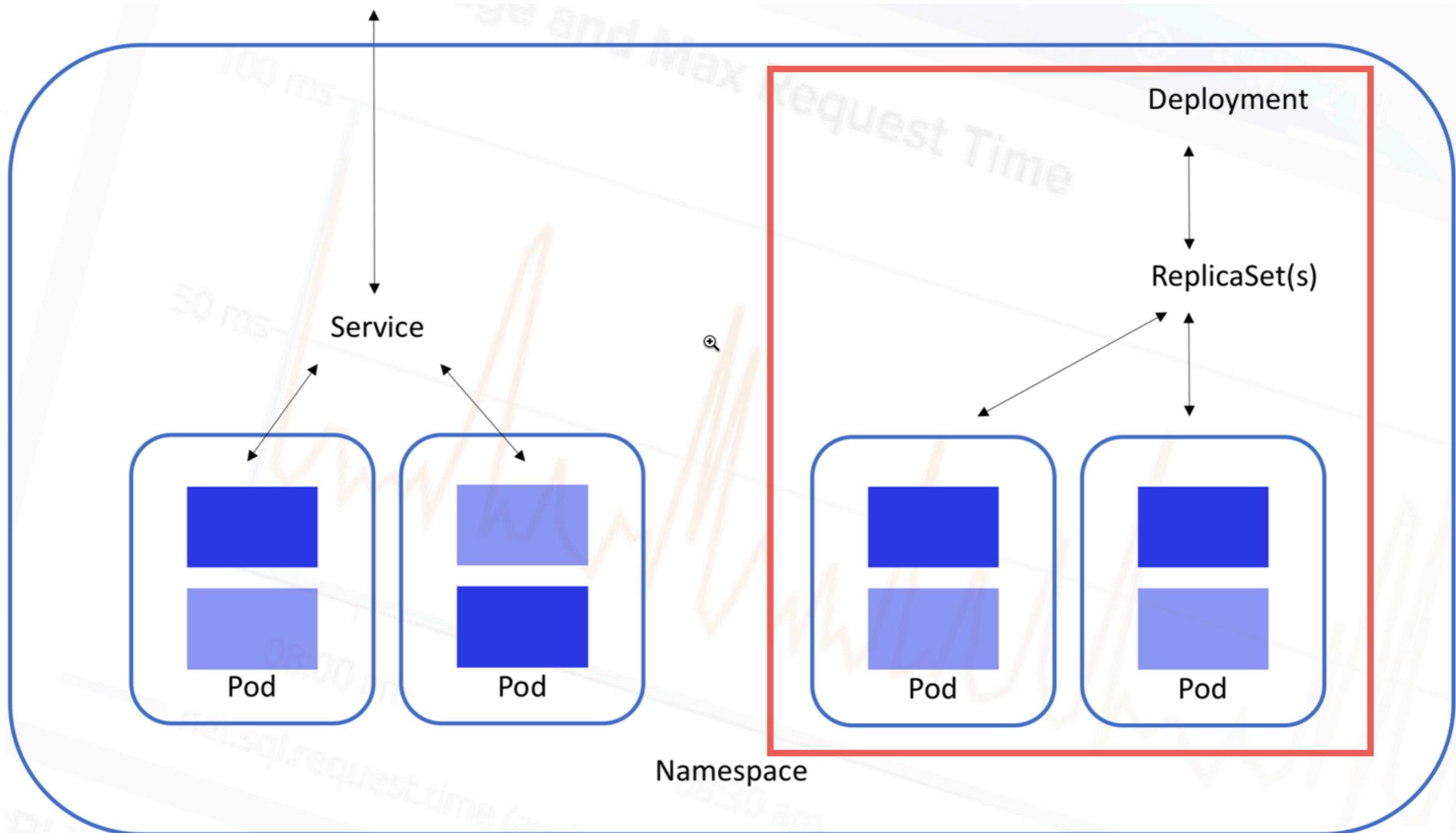


Deployment and ReplicaSet (RS)

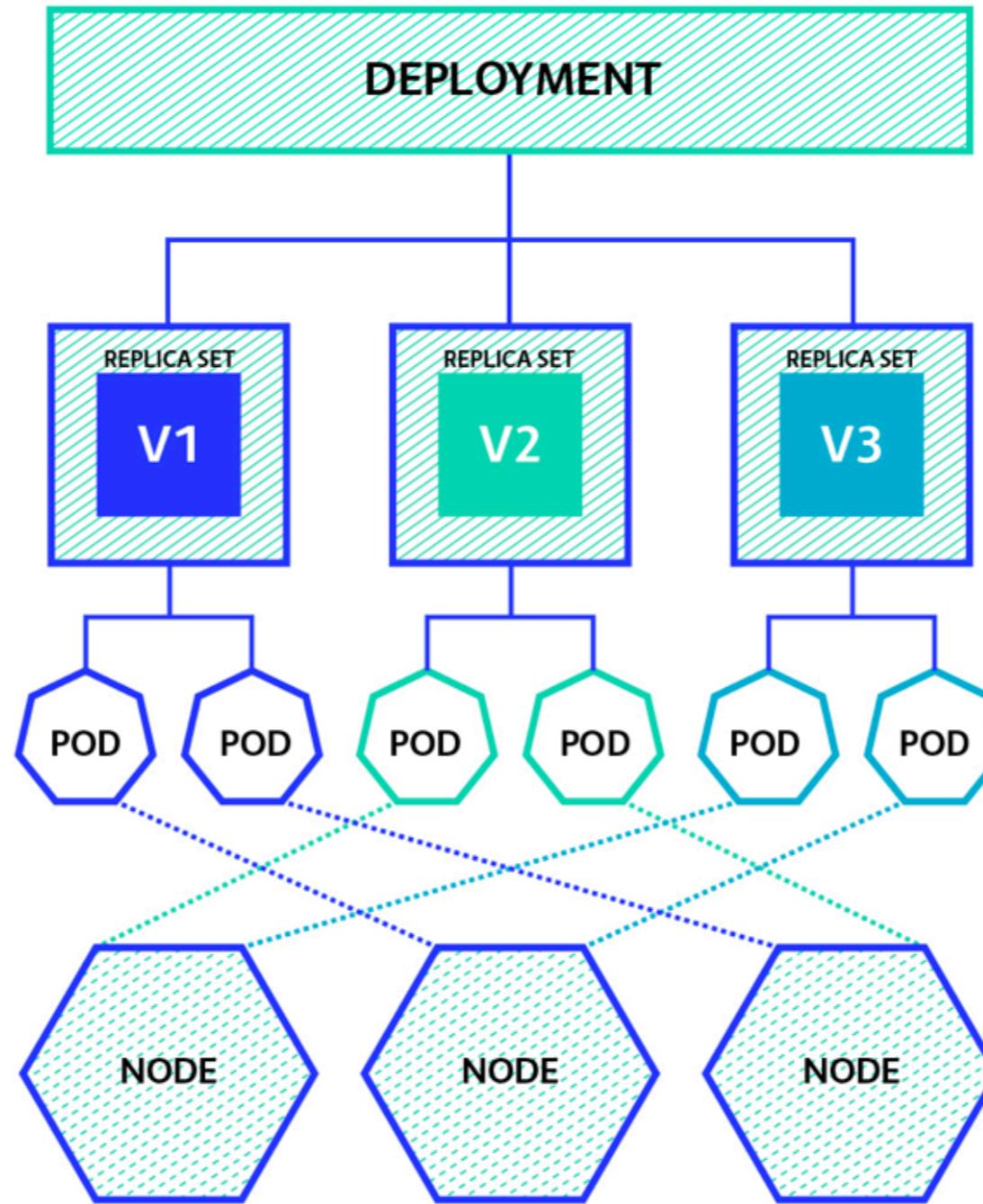
file /04-deployment-and-replica-set



Deployment and ReplicaSet



Deployment and ReplicaSet



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



Deployment and ReplicaSet

Next-generation of RC

Provide function to maintain versioning of Pods

Update new version (Rollout)

Revert to old version (Rollback)

Scale a deployment

Pause/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

<https://kubernetes.io/docs/concepts/overview/working-with-objects/labels/>



Label selector

Equality-based requirement

```
spec:  
  replicas: 3  
  selector:  
    app: nginx  
    environment: production  
    tier: frontend
```

Set-based requirement

```
spec:  
  replicas: 3  
  selector:  
    matchLabels:  
      tier: frontend  
    matchExpressions:  
      - {key: tier, operator: In, values: [frontend]}
```



Set-based requirement can be
mixed with equality-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":{},"matchExpressions":[]},  
      "template":{"metadata":{"labels":{"app":"web"}},  
      "spec":{"containers":[{"image":"nginx:1.13.6-alpine",  
      "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



Horizontal Pods Autoscale (HPA)

File /working-with-java/03-hpa



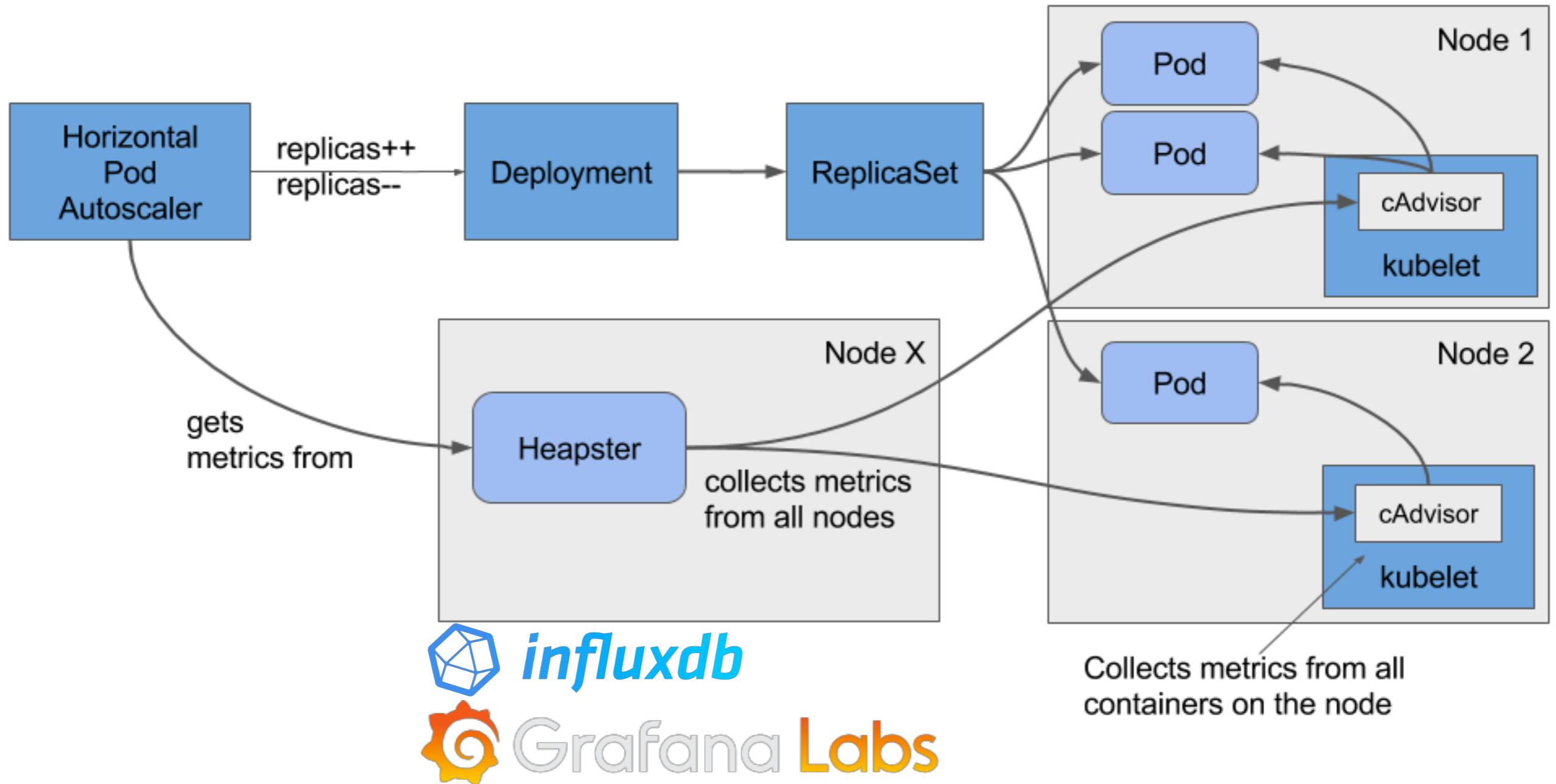
Horizontal Pod Autoscaler

Monitor workload on Pods (based on CPU)
and automatic scaling-up application

Scaling-up and down for your app need !!



Components in Autoscaling



<https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale-walkthrough/>



1. Enable hipster in minikube

\$minikube addons enable heapster

\$minikube addons start heapster

\$minikube addons open heapster



See all pods from heapster

\$kubectl get pods --all-namespaces

| NAMESPACE | NAME | READY | STATUS | RESTARTS | AGE |
|-------------|---|-------|---------|----------|-----|
| default | mongo-controller-xdh28 | 1/1 | Running | 0 | 9h |
| default | spring-boot-service-deployment-74f5df876f-gxn24 | 1/1 | Running | 0 | 4h |
| kube-system | default-http-backend-kf92s | 1/1 | Running | 0 | 4h |
| kube-system | heapster-2ssh2 | 1/1 | Running | 0 | 8h |
| kube-system | influxdb-grafana-vk6j6 | 2/2 | Running | 0 | 8h |
| kube-system | kube-addon-manager-minikube | 1/1 | Running | 2 | 1d |
| kube-system | kube-dns-54cccfbd8-xkpl6 | 3/3 | Running | 6 | 1d |
| kube-system | kubernetes-dashboard-77d8b98585-n49sg | 1/1 | Running | 4 | 1d |
| kube-system | metrics-server-bb9ffc6b8-5mgnv | 1/1 | Running | 0 | 8h |
| kube-system | nginx-ingress-controller-nrk4d | 1/1 | Running | 0 | 4n |
| kube-system | storage-provisioner | 1/1 | Running | 2 | 1d |



See all services from heapster

\$kubectl get service --all-namespaces

| NAMESPACE | NAME | TYPE | CLUSTER-IP | EXTERNAL-IP | PORT(S) | AGE |
|-------------|----------------------|-----------|----------------|-------------|--------------------|-----|
| default | kubernetes | ClusterIP | 10.96.0.1 | <none> | 443/TCP | 1d |
| default | mongo-service | ClusterIP | 10.99.242.178 | <none> | 27017/TCP | 9h |
| default | spring-boot-service | NodePort | 10.98.105.78 | <none> | 80:32132/TCP | 4h |
| kube-system | default-http-backend | NodePort | 10.102.221.170 | <none> | 80:30001/TCP | 1h |
| kube-system | heapster | ClusterIP | 10.100.221.108 | <none> | 80/TCP | 8h |
| kube-system | kube-dns | ClusterIP | 10.96.0.10 | <none> | 53/UDP, 53/TCP | 1d |
| kube-system | kubernetes-dashboard | NodePort | 10.99.243.227 | <none> | 80:30000/TCP | 1d |
| kube-system | metrics-server | ClusterIP | 10.107.38.180 | <none> | 443/TCP | 8h |
| kube-system | monitoring-grafana | NodePort | 10.104.125.177 | <none> | 80:30002/TCP | 8h |
| kube-system | monitoring-influxdb | ClusterIP | 10.99.175.232 | <none> | 8083/TCP, 8086/TCP | 8h |



See result in Grafana

<http://192.168.99.100:30002>

The screenshot shows the Grafana Home Dashboard. At the top, there is a header bar with a gear icon, a 'Home' button, and a search bar containing the URL '192.168.99.100:30002/?orgId=1'. Below the header, the title 'Home Dashboard' is displayed. The main content area features a 'Getting Started with Grafana' section with five steps: 'Install Grafana' (checkmark), 'Create your first data source' (checkmark), 'Create your first dashboard' (checkmark), 'Add Users' (highlighted in green), and 'Install apps & plugins' (cloud icon). To the left, there are sections for 'Starred dashboards' (Pods, Cluster) and 'Recently viewed dashboards'. To the right, there are sections for 'Installed Apps' (None installed, Browse Grafana.net), 'Installed Panels' (None installed, Browse Grafana.net), and 'Installed Datasources' (None installed, Browse Grafana.net).



2. Create metric server

Starting from Kubernetes 1.8, resource usage metrics, such as container CPU and memory usage, are available in Kubernetes through the Metrics API. These metrics can be either accessed directly by user, for example by using `kubectl top` command, or used by a controller in the cluster, e.g. Horizontal Pod Autoscaler, to make decisions.

<https://github.com/kubernetes-incubator/metrics-server>



3. Deploy Springboot service

```
$kubectl create -f boot-deployment.yaml  
$kubectl create -f boot-service.yaml
```



4. Create HPA in command line

```
$kubectl autoscale  
deployment/spring-boot-service-deployment  
--min=1 --max=5 --cpu-percent=5
```

min = minimum of replica

max = maximum of replica

cpu-percent = average of % of CPU usage



4. Create HPA in command line

\$kubectl get hpa

REFERENCE

Deployment/spring-boot-service-deployment

| | TARGETS | MINPODS | MAXPODS |
|--|---------|---------|---------|
| | 0%/5% | 1 | 5 |



4. Create HPA in command line

**Scale-up can only happen if there was no rescaling
within the last 3 minutes**

**Scale-down will wait for 5 minutes from the last
rescaling**

<https://github.com/kubernetes/community/blob/master/contributors/design-proposals/autoscaling/horizontal-pod-autoscaler.md#autoscaling-algorithm>



5. Load testing with command line

`http://spring-boot-service.default.svc.cluster.local:
8080/user`



6. See result (1)

\$kubectl get pods

| NAME | READY | STATUS | RESTARTS | AGE |
|---|-------|---------|----------|-----|
| load-test-6f86656986-9qvdl | 0/1 | Error | 0 | 4m |
| mongo-controller-xdh28 | 1/1 | Running | 0 | 1h |
| spring-boot-service-deployment-86c7b47b7b-9l9sn | 1/1 | Running | 0 | 15m |
| spring-boot-service-deployment-86c7b47b7b-9srkg | 1/1 | Running | 0 | 1m |
| spring-boot-service-deployment-86c7b47b7b-kbtrl | 1/1 | Running | 0 | 1m |
| spring-boot-service-deployment-86c7b47b7b-pqbsm | 1/1 | Running | 0 | 1m |



6. See result (2)

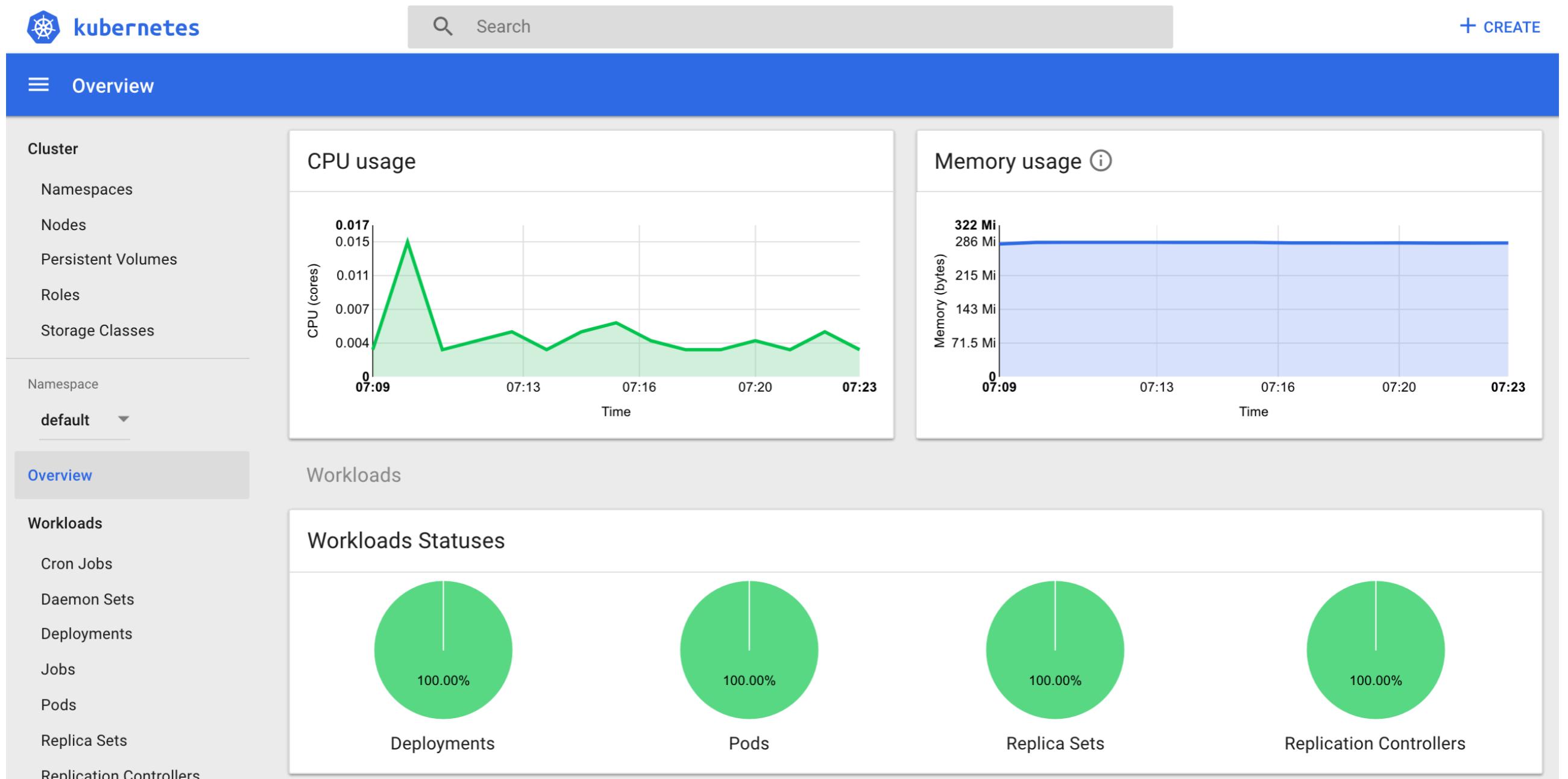
\$kubectl describe hpa

```
Metrics:                               ( current / target )
  resource cpu on pods  (as a percentage of request): 134% (269m) / 5%
Min replicas:                           1
Max replicas:                           5
Conditions:
  Type        Status  Reason
  ----        ----  -----
  AbleToScale False   BackoffBoth
  ScalingActive True    ValidMetricFound
  ScalingLimited True   TooManyReplicas
Events:
  Type      Reason
  ----      -----
  Normal    SuccessfulRescale
  Warning   FailedGetResourceMetric
  Warning   FailedUpdateStatus
  Warning   FailedComputeMetricsReplicas
```

The output shows the Horizontal Pod Autoscaler (HPA) configuration and its current state. It includes metrics (resource CPU usage), minimum and maximum replicas, and conditions (e.g., AbleToScale, ScalingActive, ScalingLimited). The Events section highlights successful rescaling and several warning messages related to failed metric fetching and updates.



7. See result in dashboard (1)



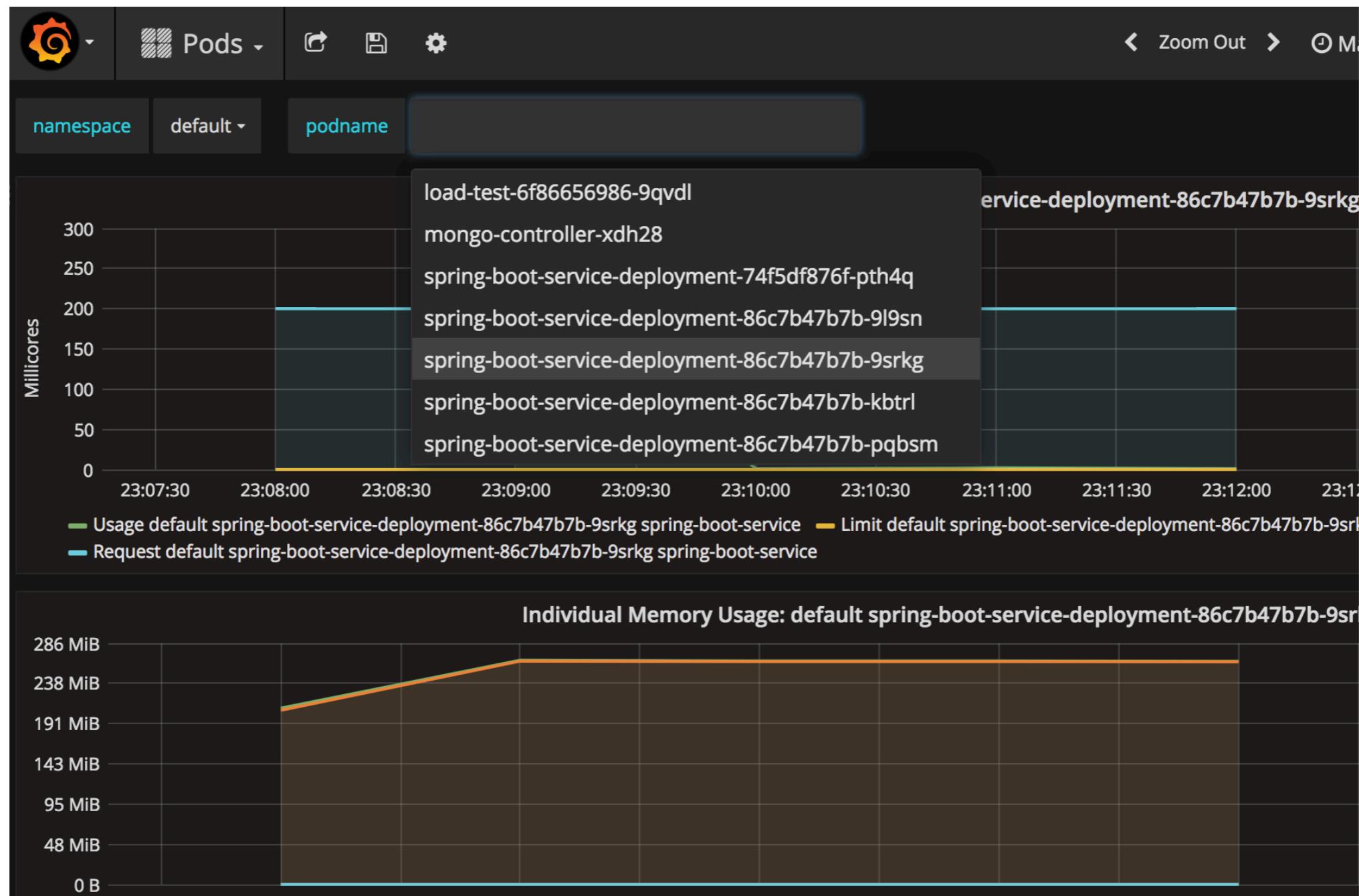
7. See result in dashboard (2)

| Pods | | | | | | | |
|--------------------------|----------|---------|----------|------------|---|----------------|--|
| Name | Node | Status | Restarts | Age | CPU (cores) | Memory (bytes) | |
| ✓ spring-boot-service-de | minikube | Running | 0 | 2 minutes |  0.249 | 265.926 Mi |   |
| ✓ spring-boot-service-de | minikube | Running | 0 | 2 minutes |  0.03 | 254.156 Mi |   |
| ✓ spring-boot-service-de | minikube | Running | 0 | 2 minutes |  0.001 | 94.840 Mi |   |
| ✓ load-test-6f86656986- | minikube | Running | 1 | 5 minutes |  0 | 688 Ki |   |
| ✓ spring-boot-service-de | minikube | Running | 0 | 16 minutes |  0.018 | 33.211 Mi |   |
| ✓ mongo-controller-xdh2 | minikube | Running | 0 | an hour |  0.004 | 24.453 Mi |   |



8. See result

In Grafana dashboard



Stateful Application Deployment



Stateless vs Stateful

Stateless

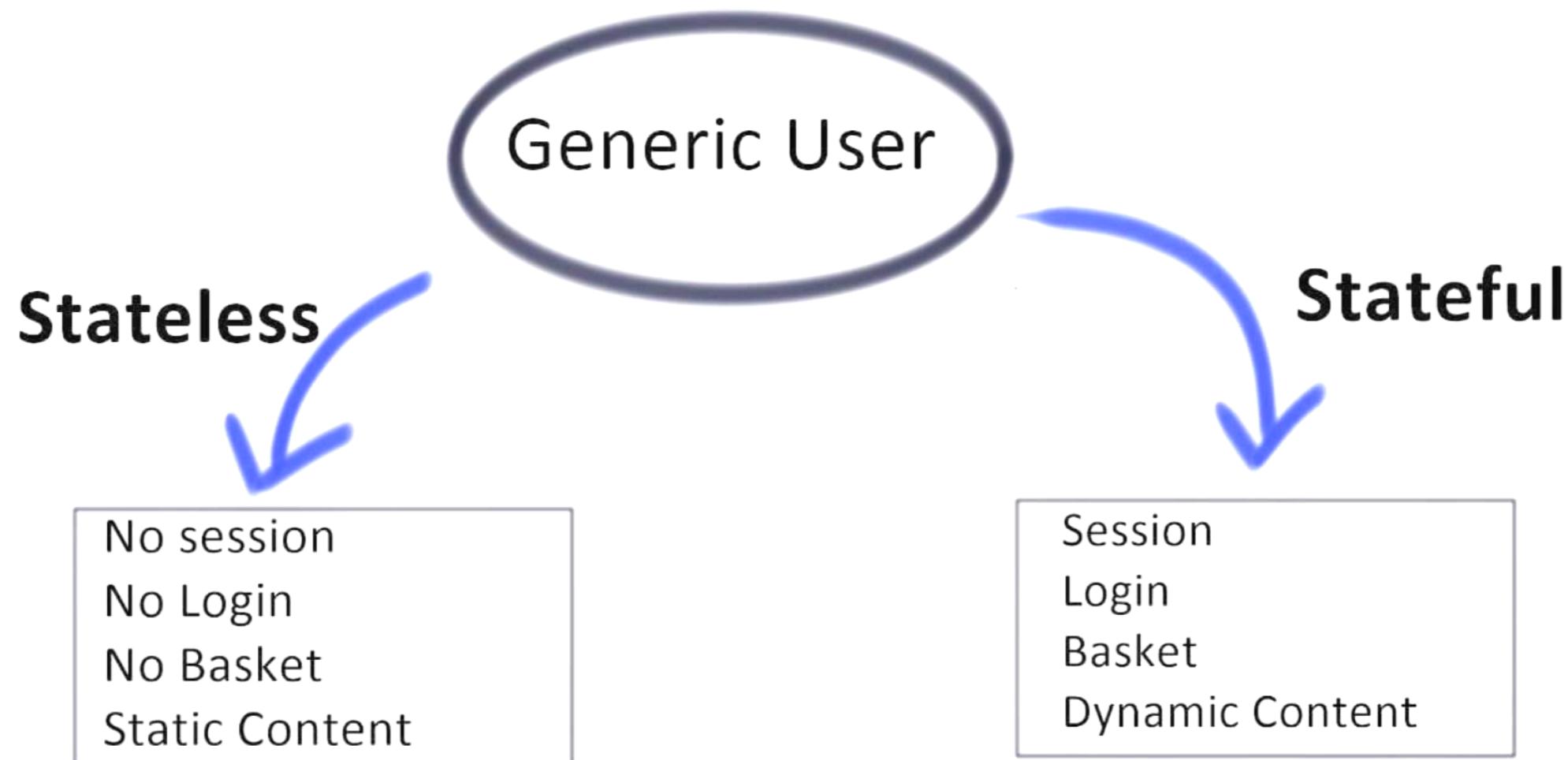
No information about a transaction/request is maintained after it is processed.

Stateful

State information is kept even after a transaction/request has been processed.



Stateless vs Stateful



Container world

Container designed for **Stateless** application

All Load Balance/dispatch process is **not** aware about **state** of application



Solution for Stateful

Share centralize storage pool for all node

Web and Application server

- Keep session and files in storage pool
- Every servers are read/write data on the same place



Solution for Stateful

Share centralize storage pool for all node

Database server

- Active/Active
- Active/Standby
- Keep data in storage pool

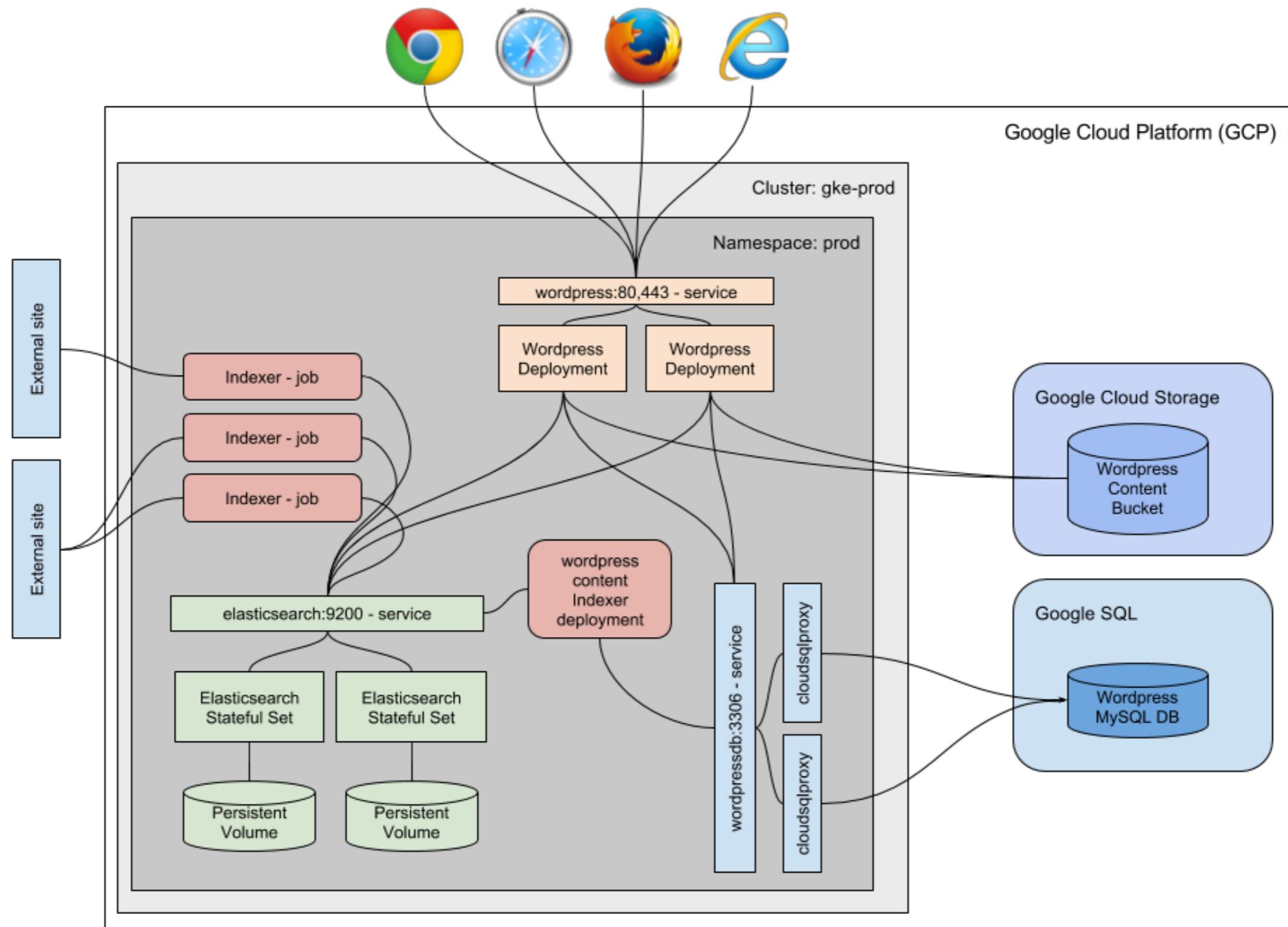


Stateful with Kubernetes

Volume
Statefulset



Stateful with Kubernetes



<http://blogs.avalonconsult.com/blog/enterprise-web/cloud-computing/architecting-for-kubernetes/>

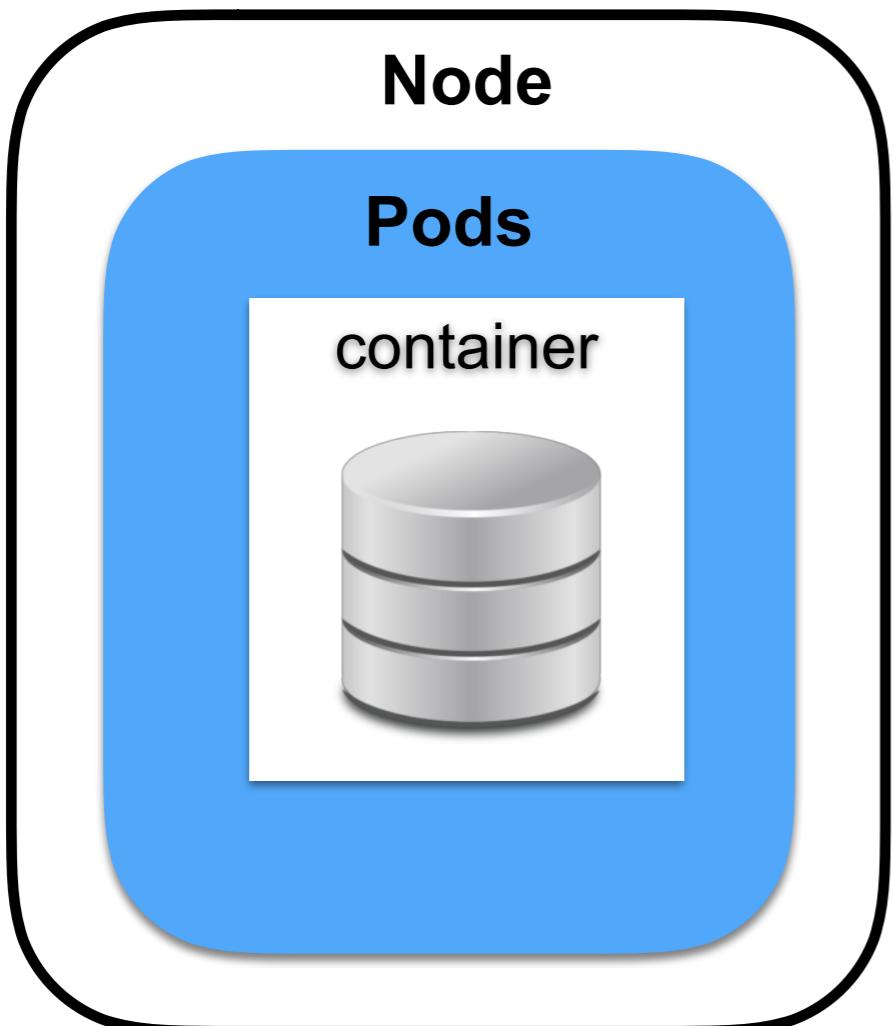


Volume in Kubernetes

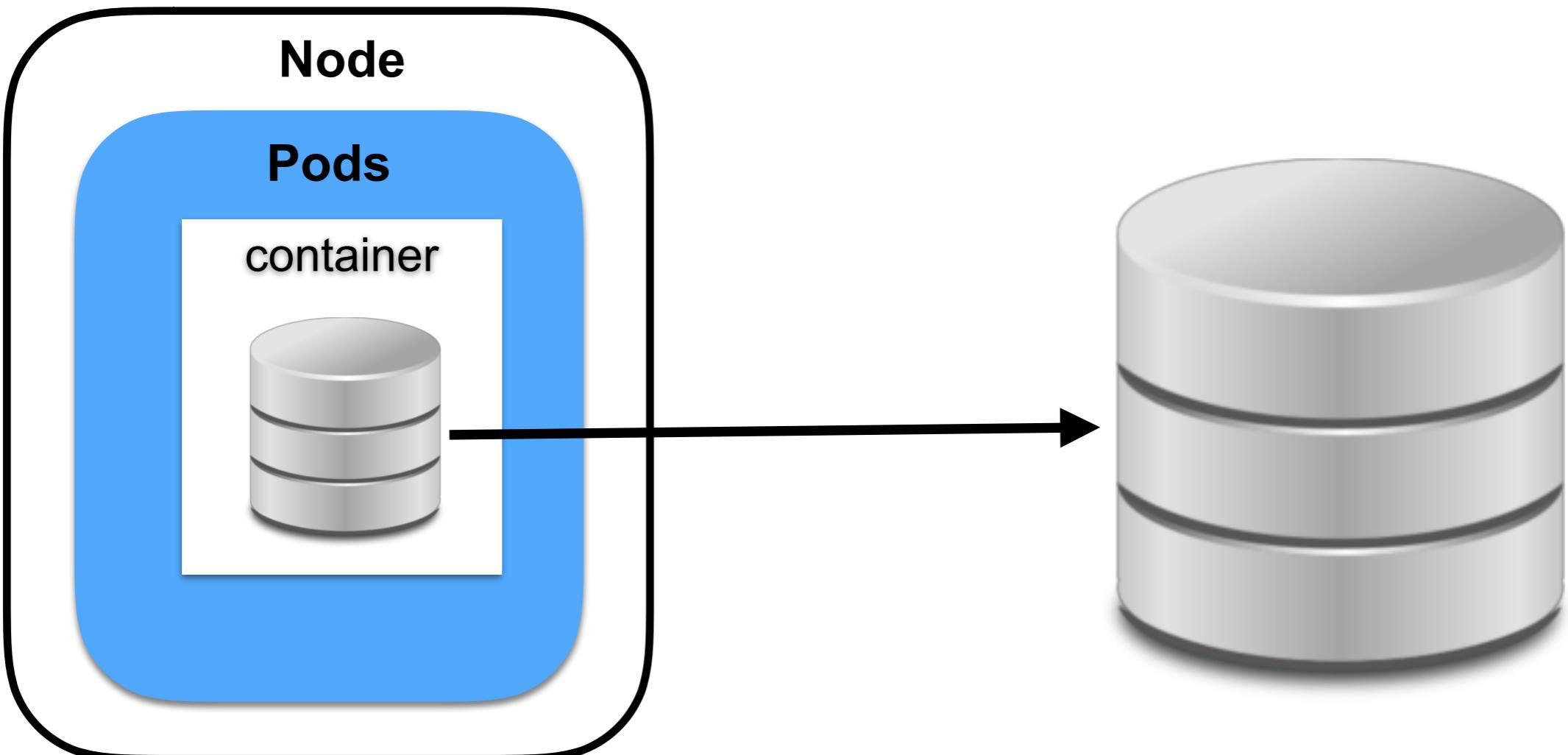
<https://kubernetes.io/docs/concepts/storage/volumes/>



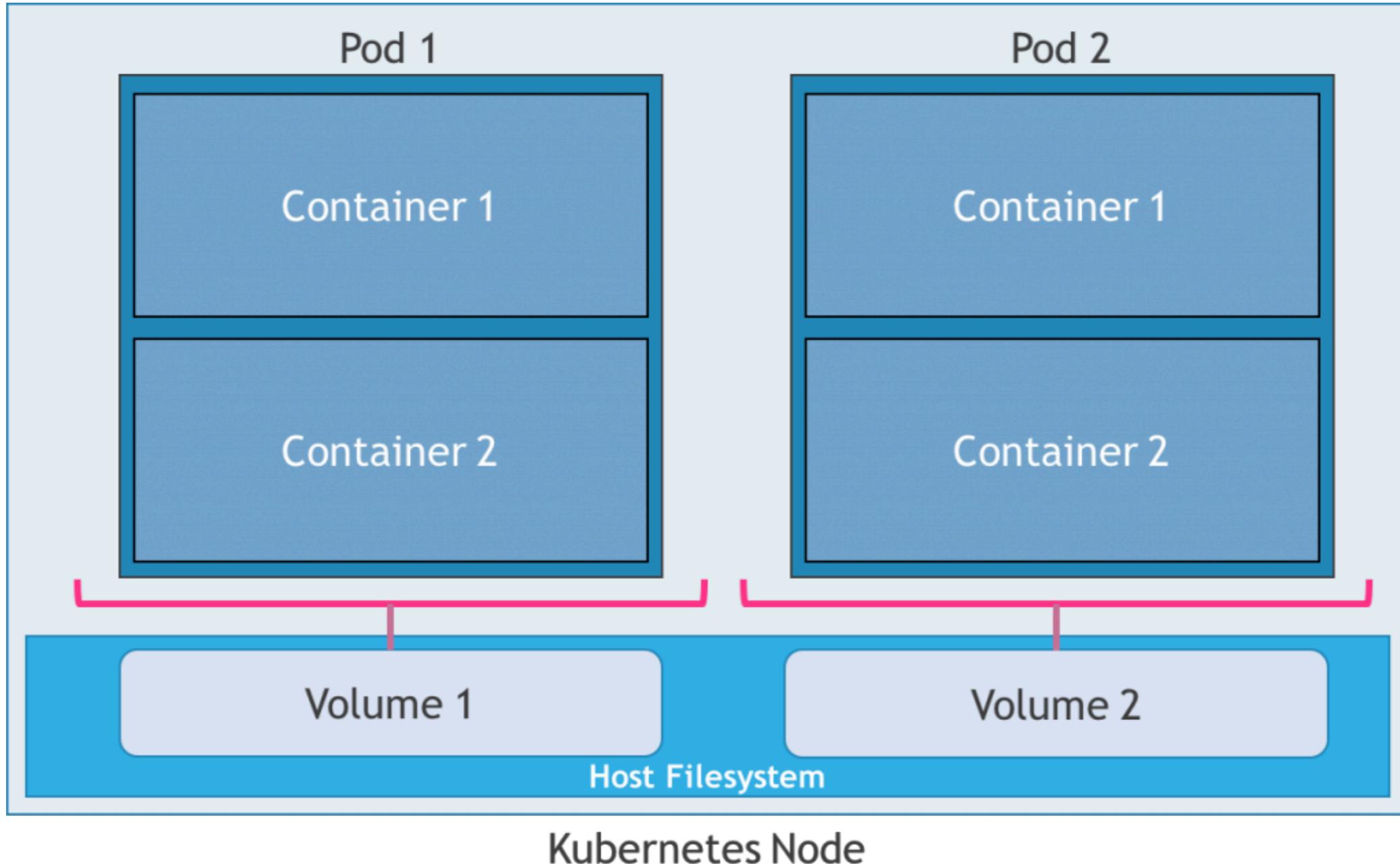
Volume



Volume



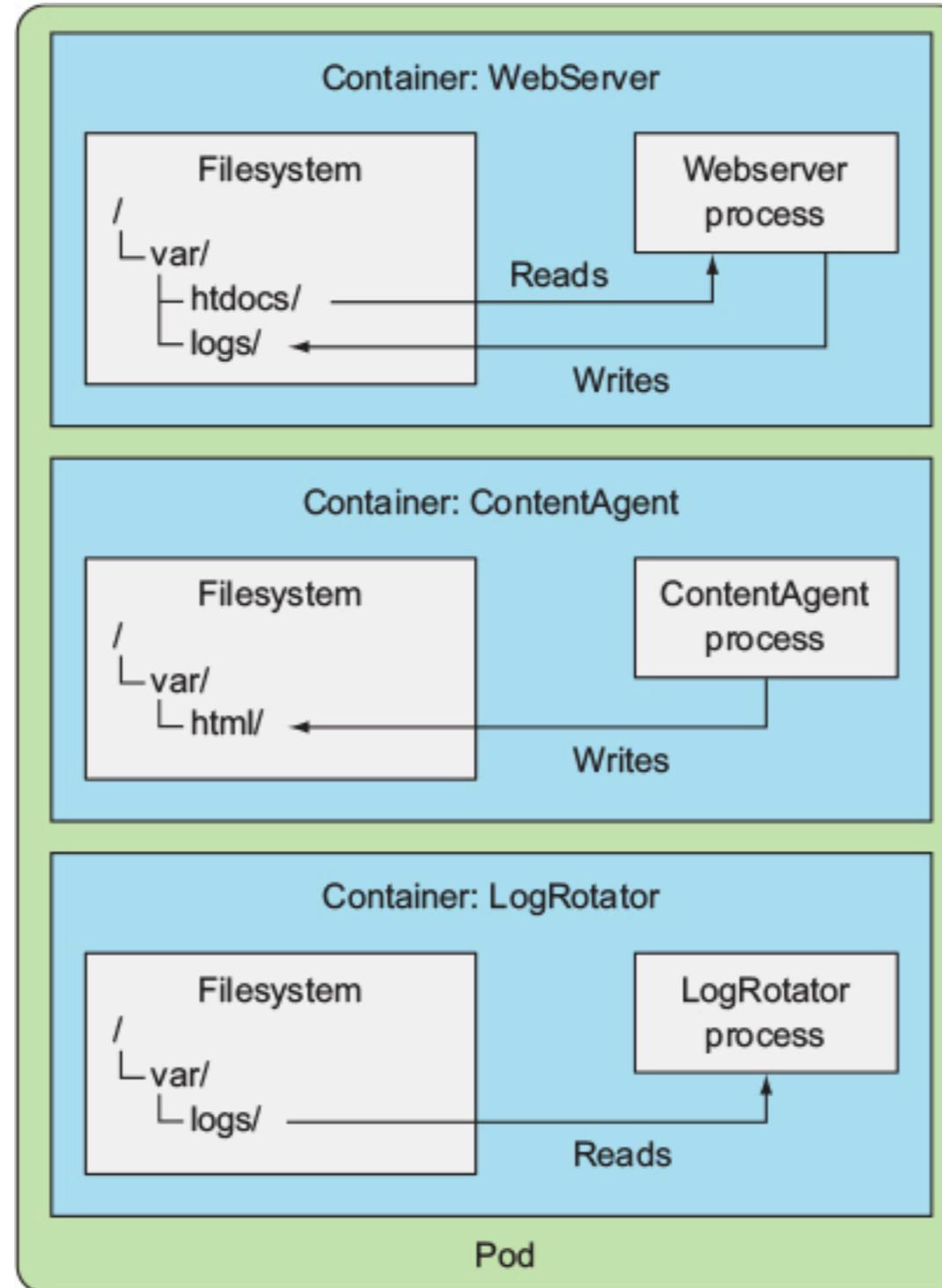
Containers shared volume



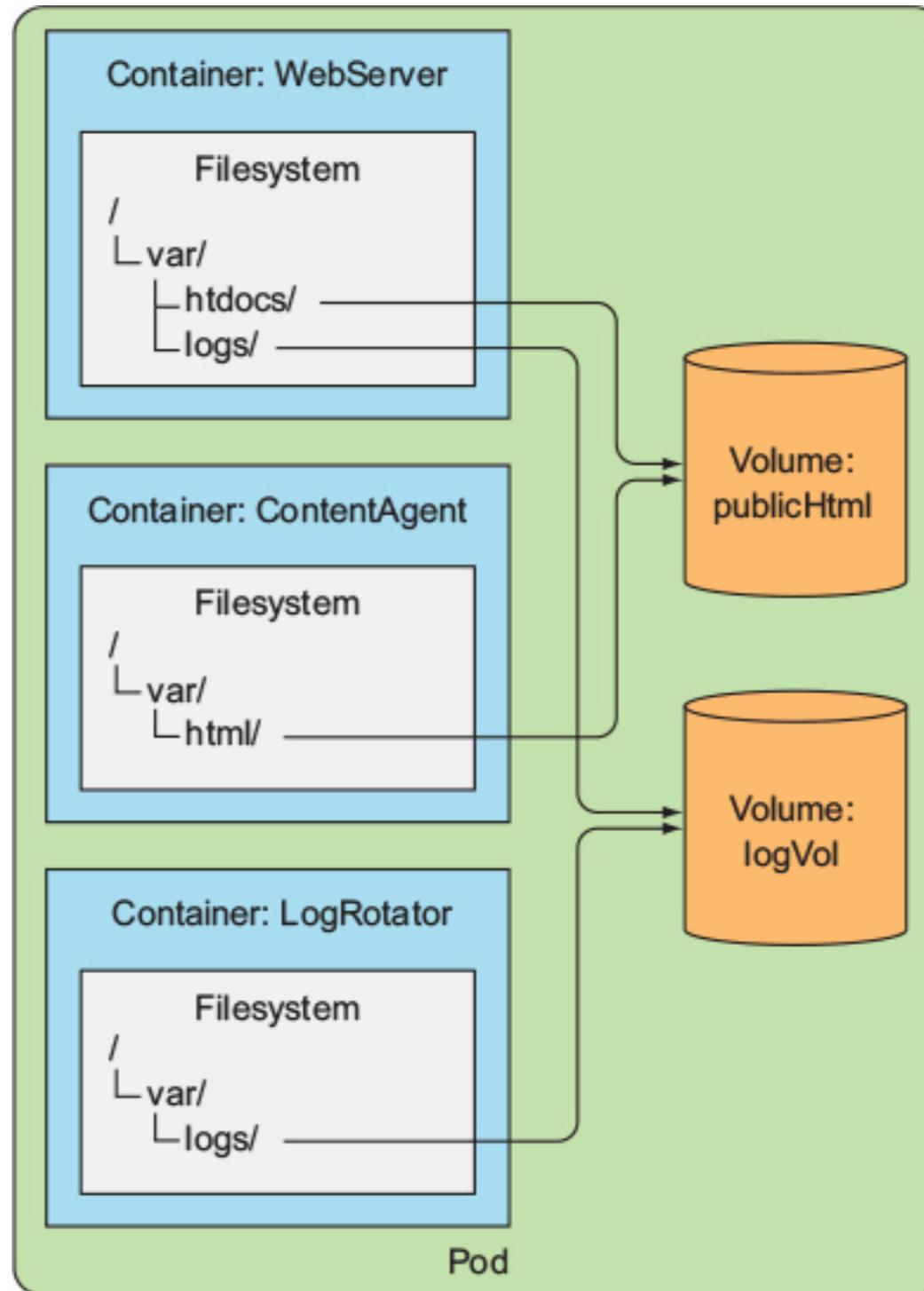
<https://thenewstack.io/strategies-running-stateful-applications-kubernetes-volumes/>



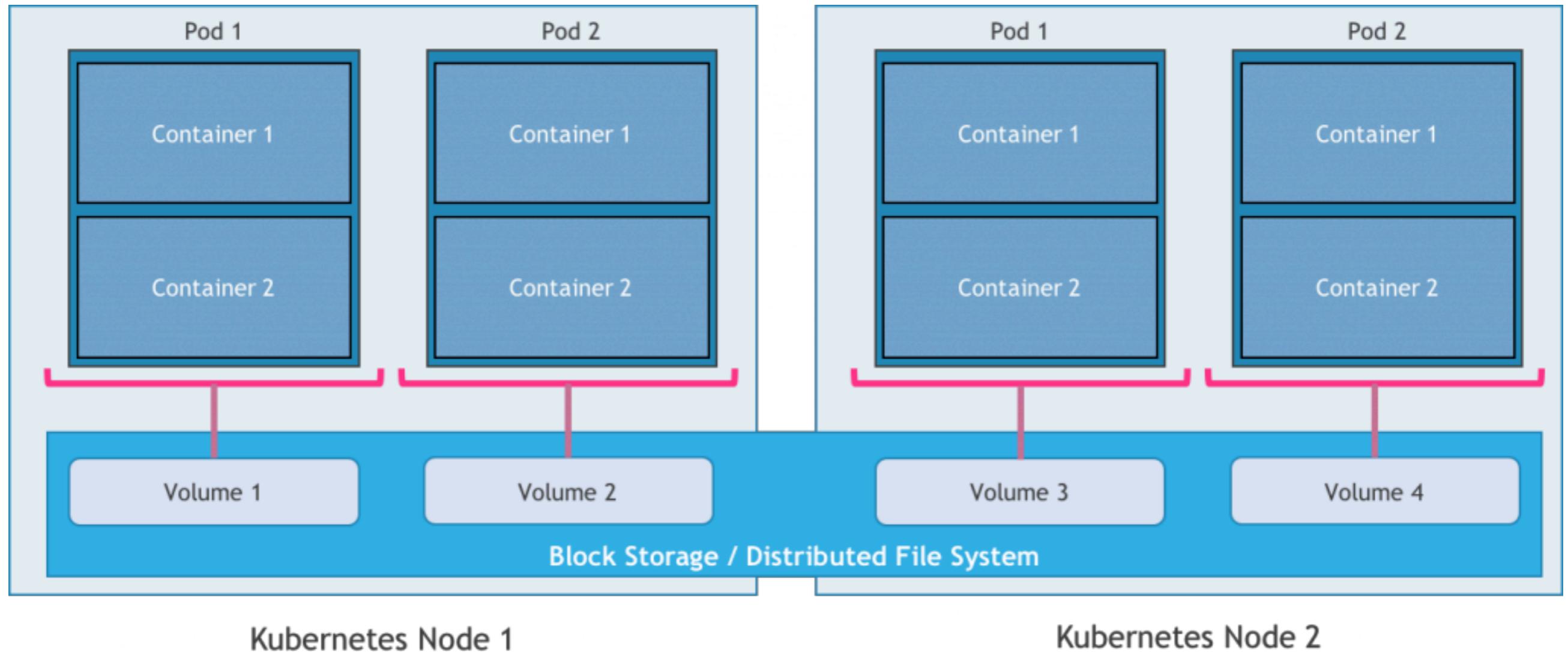
Without shared storage



Containers shared volume



Nodes shared volume



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



Volume

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



Volume on Kubernetes

`EmptyDir`

`hostPath`

`gitRepo`

`NFS`

`icePersistentDisk`

`Flocker`

Persistent Volume Claim (PVC)

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



Volume on Kubernetes

EmptyDir

HostPath

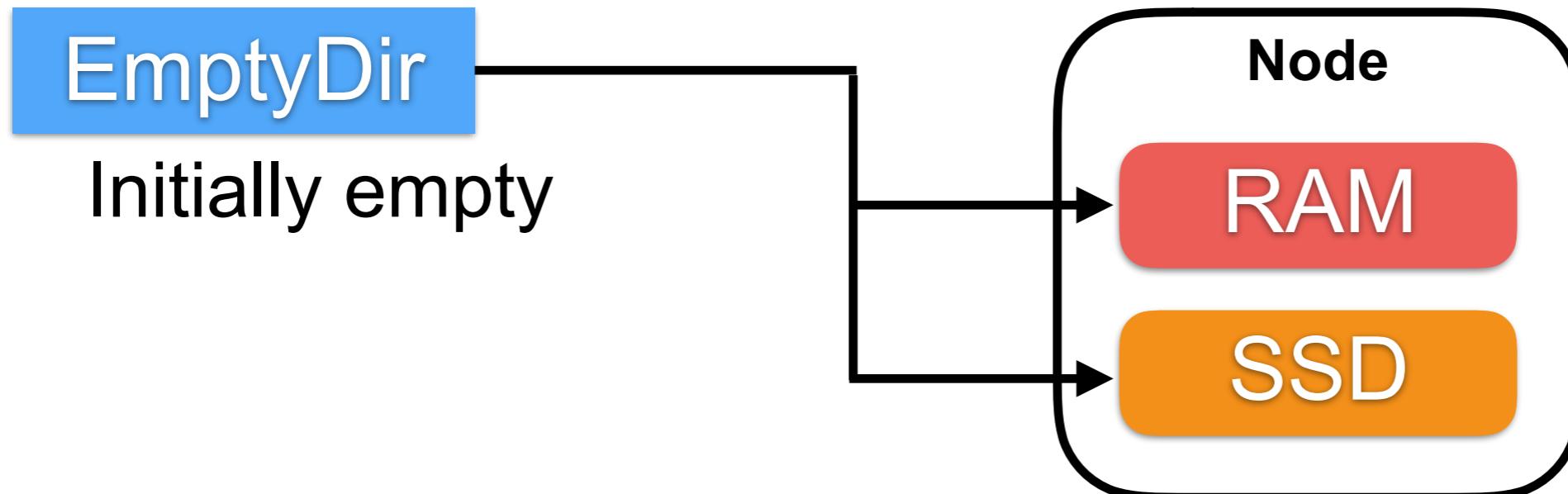
NFS

Cloud Volume

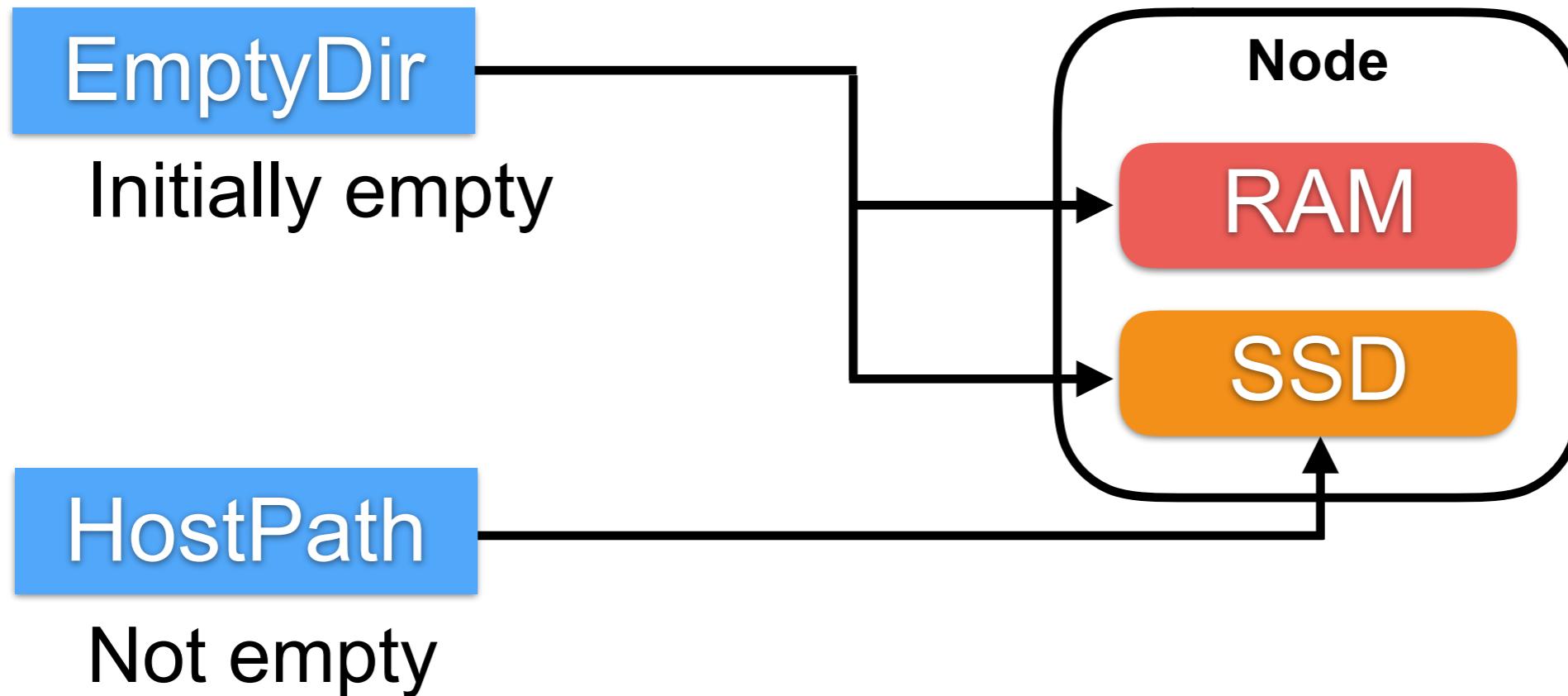
Persistent Volume Claim



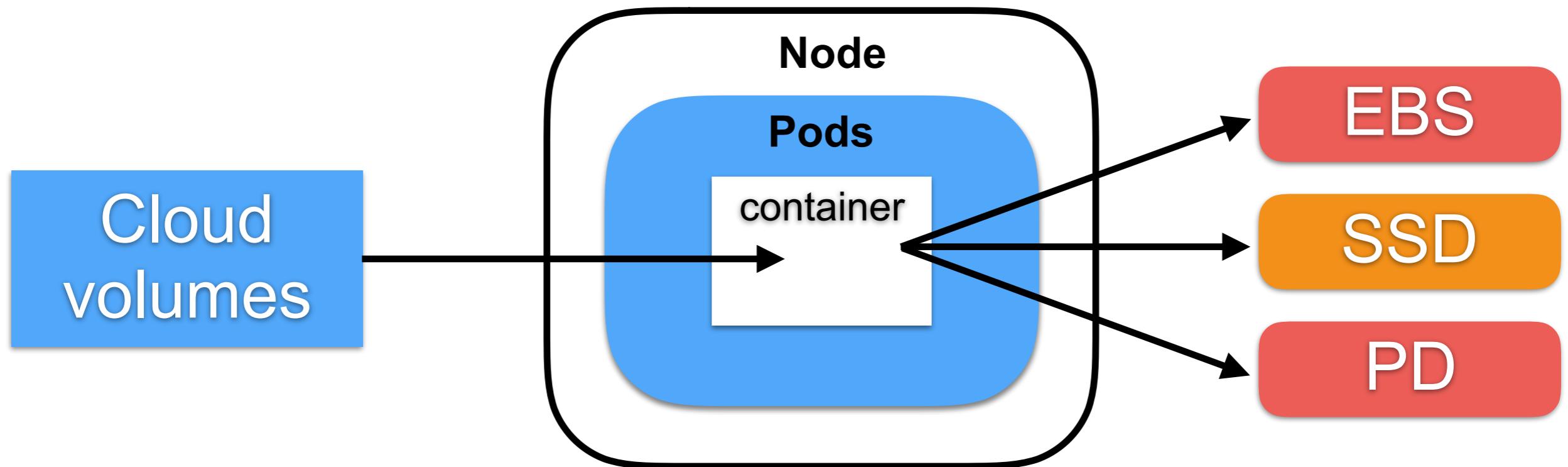
Volume on Kubernetes



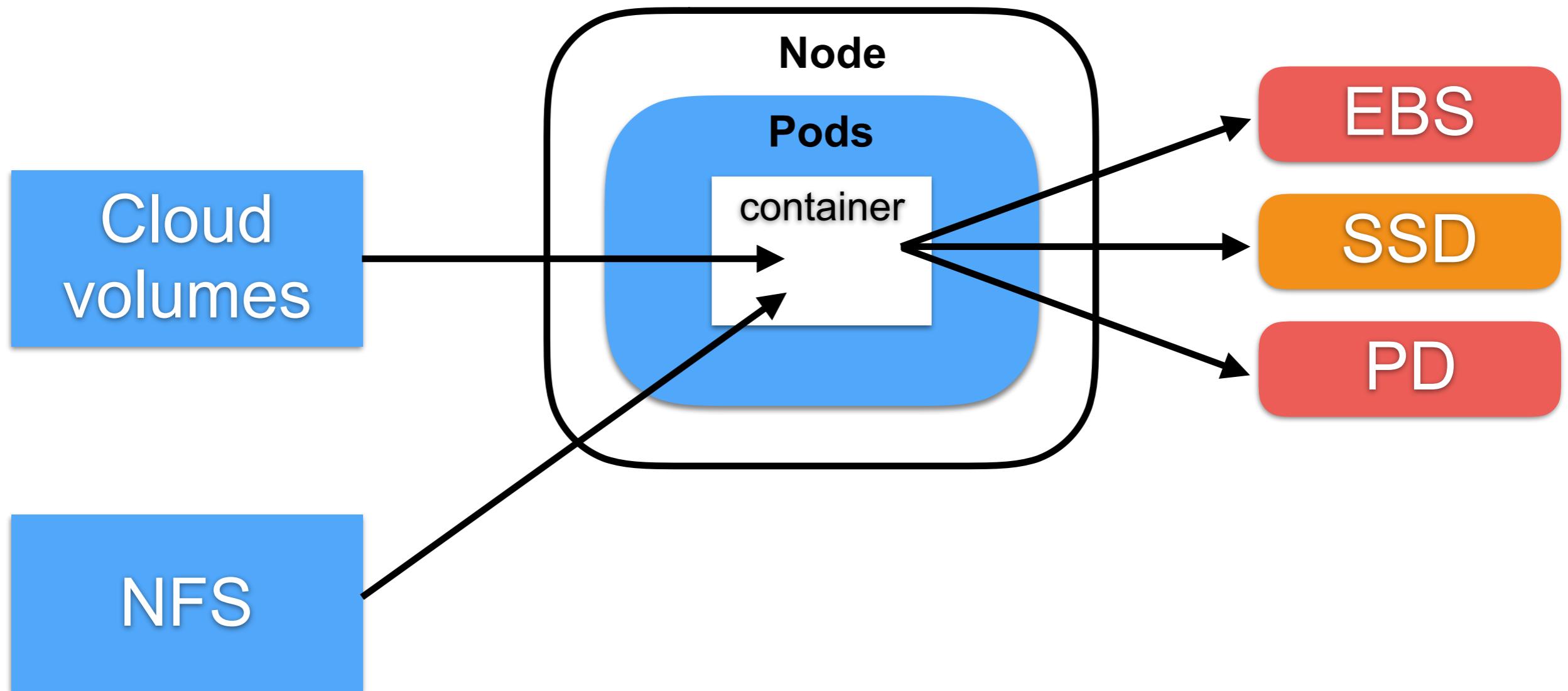
Volume on Kubernetes



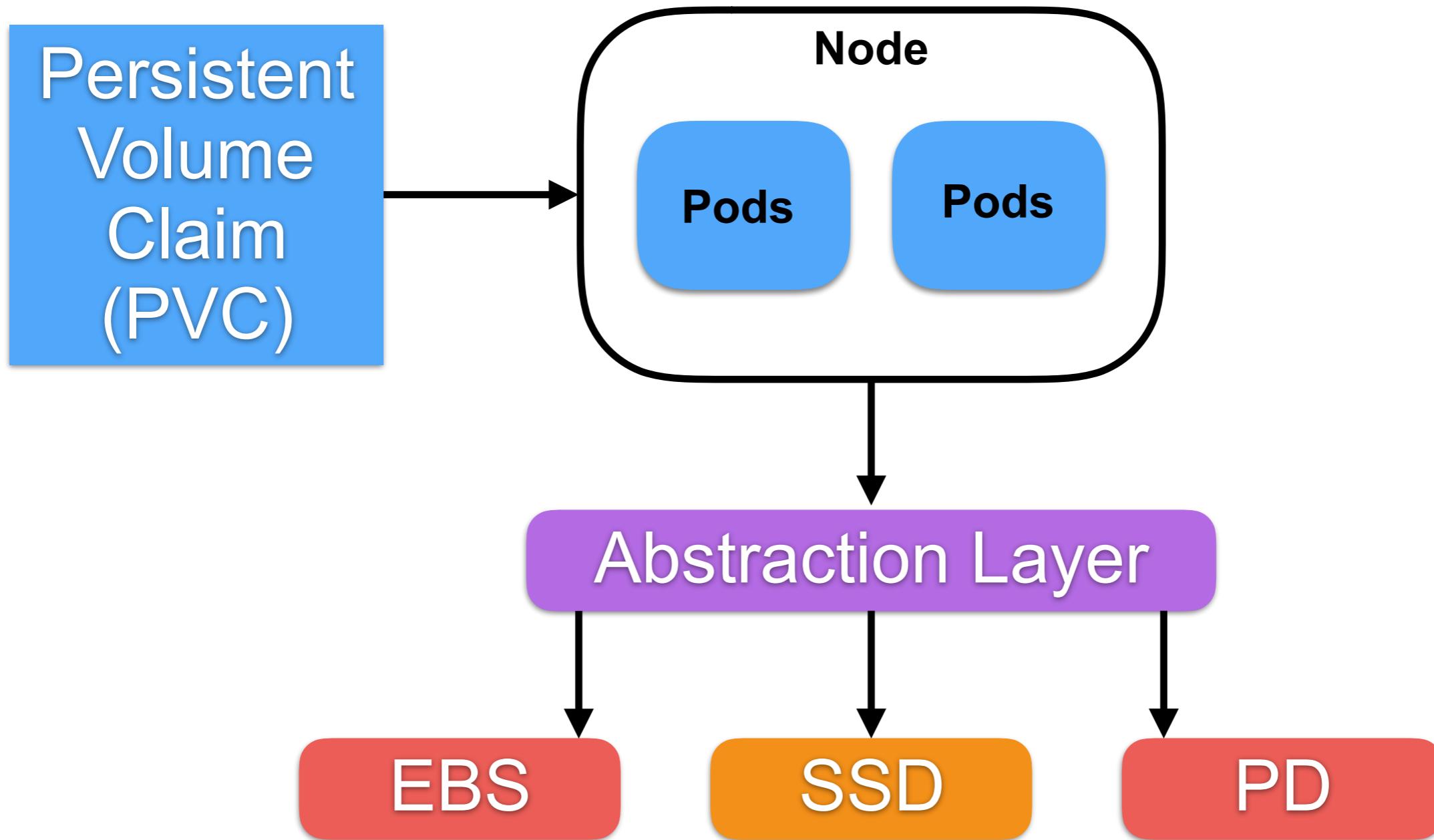
Volume on Kubernetes



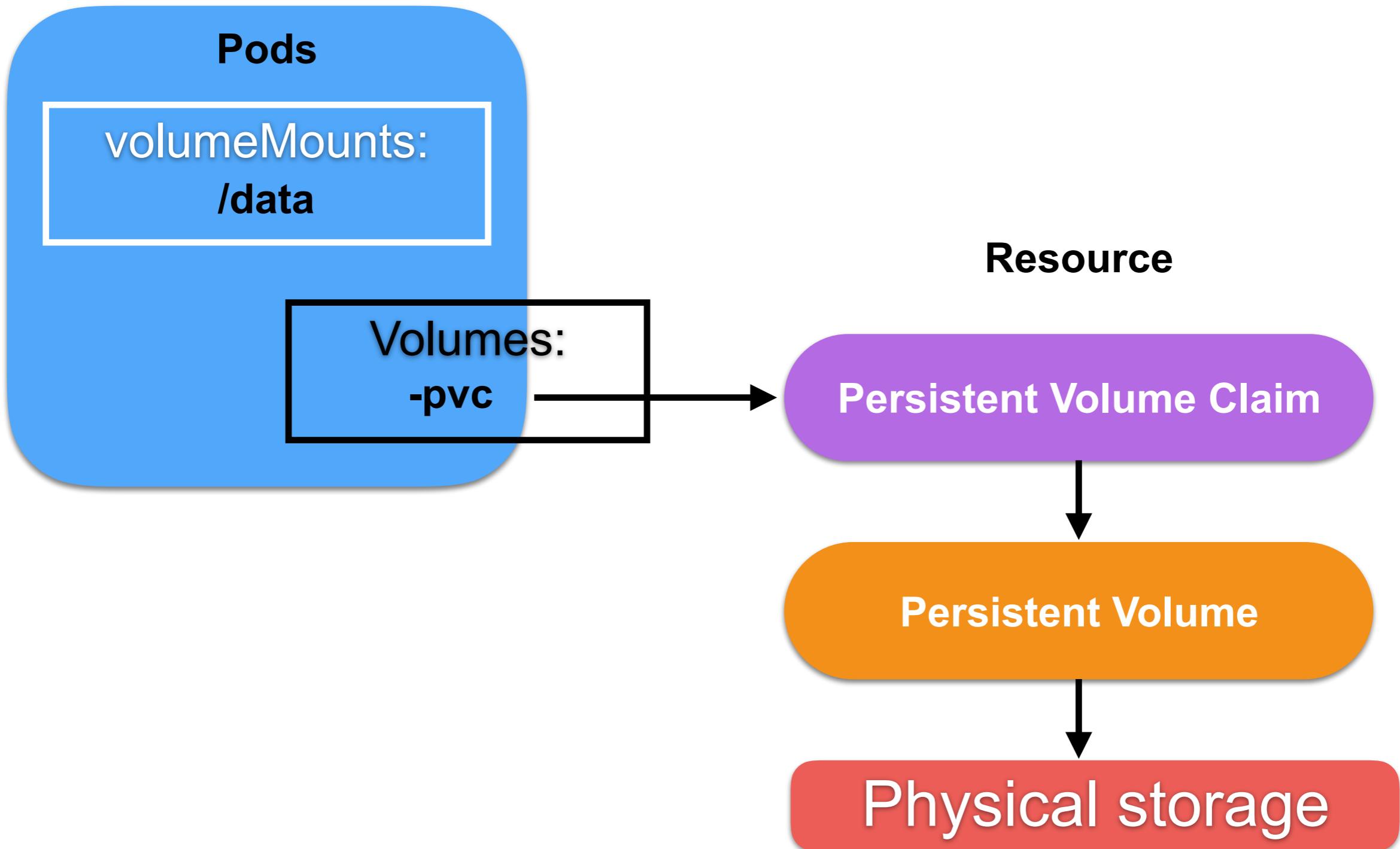
Volume on Kubernetes



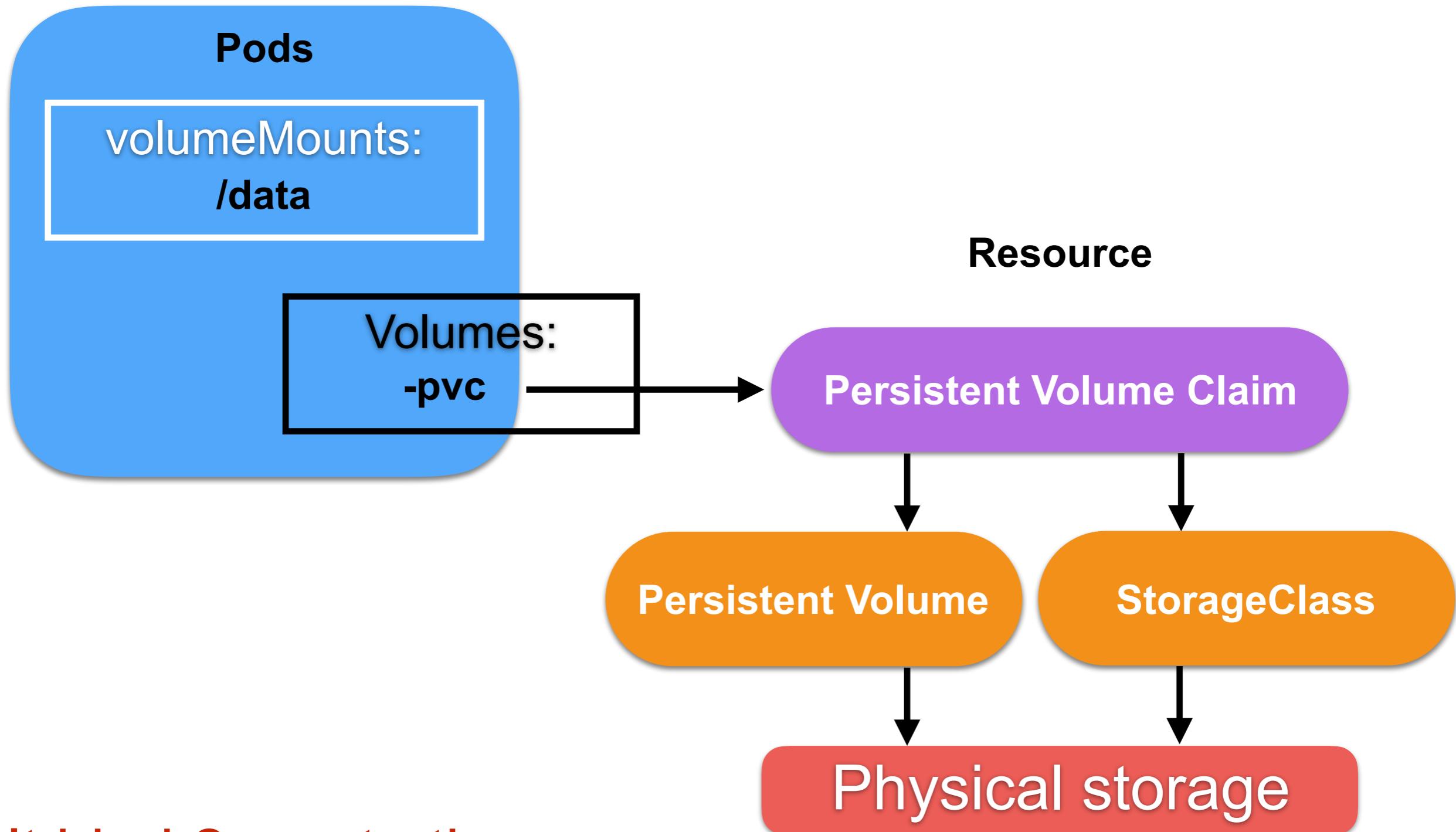
Volume on Kubernetes



Volume on Kubernetes



Volume on Kubernetes



Can't bind 2 pvc to the same pv.



Persistent Volume

Provide and APIs for user and administrator
Abstraction layer to hide detail of storage

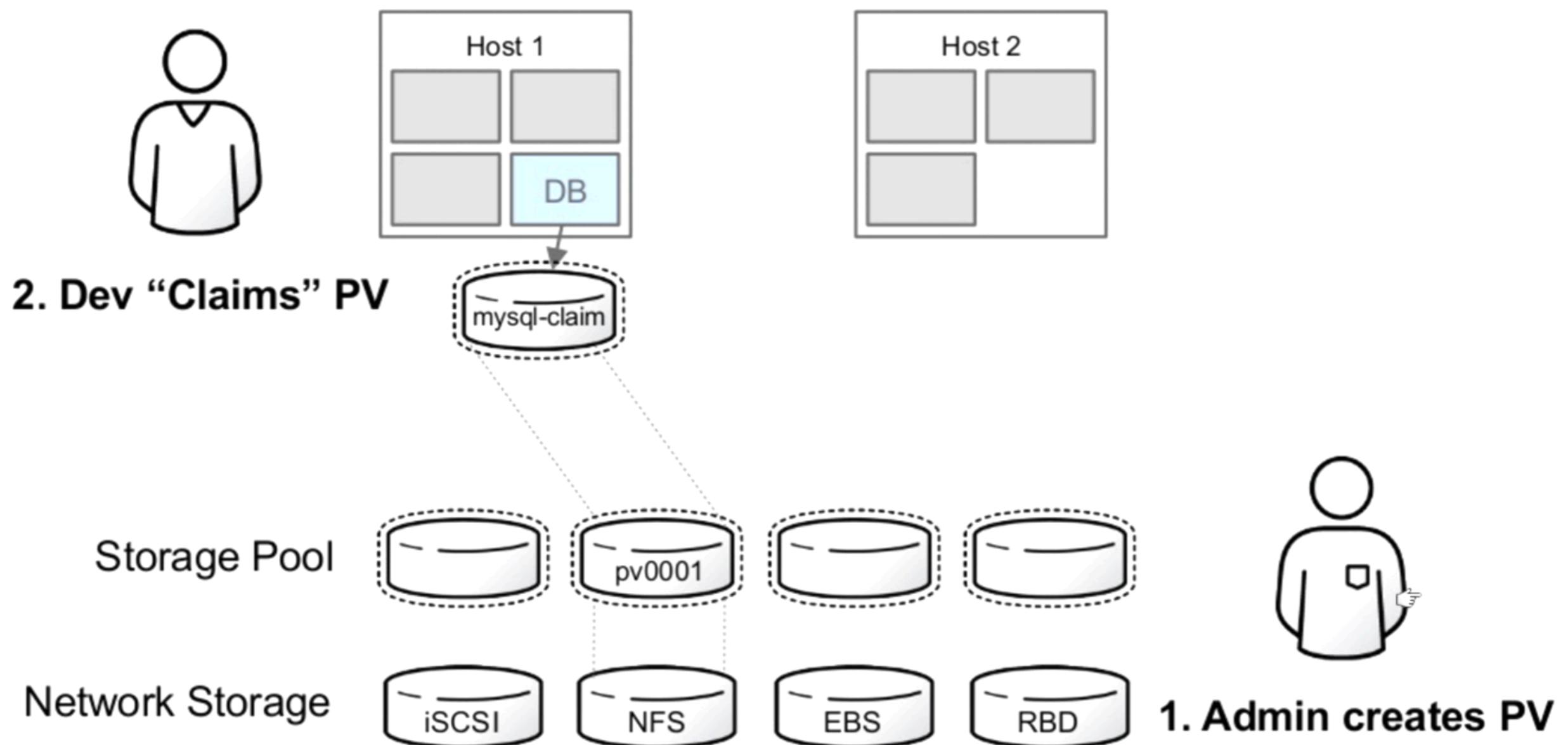
Provide 2 apis for manage your storage

1. PersistentVolume (PV)
2. PersistentVolumeClaim (PVC)

<https://kubernetes.io/docs/concepts/storage/persistent-volumes/>



Persistent Volume



PersistentVolume (PV)

PV is a piece of network storage in the cluster
Piece of storage that independence from Pods
Long lifecycle independent of any Pods
Provisioning by administrator



PersistentVolume (PV)

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv0003
spec:
  capacity:
    storage: 5Gi
  volumeMode: Filesystem
  accessModes:
    - ReadWriteOnce
  persistentVolumeReclaimPolicy: Recycle
  storageClassName: slow
  mountOptions:
    - hard
    - nfsvers=4.1
  nfs:
    path: /tmp
    server: 172.17.0.2
```



PersistentVolumeClaim (PVC)

Request for resources/storage by user
Allows specific resource requests eg. size,
access modes



PersistentVolumeClaim (PVC)

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: myclaim
spec:
  accessModes:
    - ReadWriteOnce
  volumeMode: Filesystem
  resources:
    requests:
      storage: 8Gi
  storageClassName: slow
  selector:
    matchLabels:
      release: "stable"
    matchExpressions:
      - {key: environment, operator: In, values: [dev]}
```



Access modes

ReadWriteOnce (RWO)

the volume can be mounted as read-write by a single node

ReadOnlyMany (ROX)

the volume can be mounted read-only by many nodes

ReadWriteMany (RWX)

the volume can be mounted as read-write by many nodes



Access modes

| Volume Plugin | ReadWriteOnce | ReadOnlyMany | ReadWriteMany |
|----------------------|---------------|--------------|---------------|
| AWSElasticBlockStore | ✓ | - | - |
| AzureFile | ✓ | ✓ | ✓ |
| AzureDisk | ✓ | - | - |
| CephFS | ✓ | ✓ | ✓ |
| Cinder | ✓ | - | - |
| FC | ✓ | ✓ | - |
| FlexVolume | ✓ | ✓ | - |
| Flocker | ✓ | - | - |
| GCEPersistentDisk | ✓ | ✓ | - |
| Glusterfs | ✓ | ✓ | ✓ |
| HostPath | ✓ | - | - |
| iSCSI | ✓ | ✓ | - |
| Quobyte | ✓ | ✓ | ✓ |
| NFS | ✓ | ✓ | ✓ |



Lifecycle of volume and claim



Lifecycle of volume

1. Provisioning
2. Binding
3. Using
4. Reclaim (retain, recycle, delete)



Phase of volume

1. Available

Free resource that is not yet bound to a claim

2. Bound

Volume is bound to a claim

3. Released

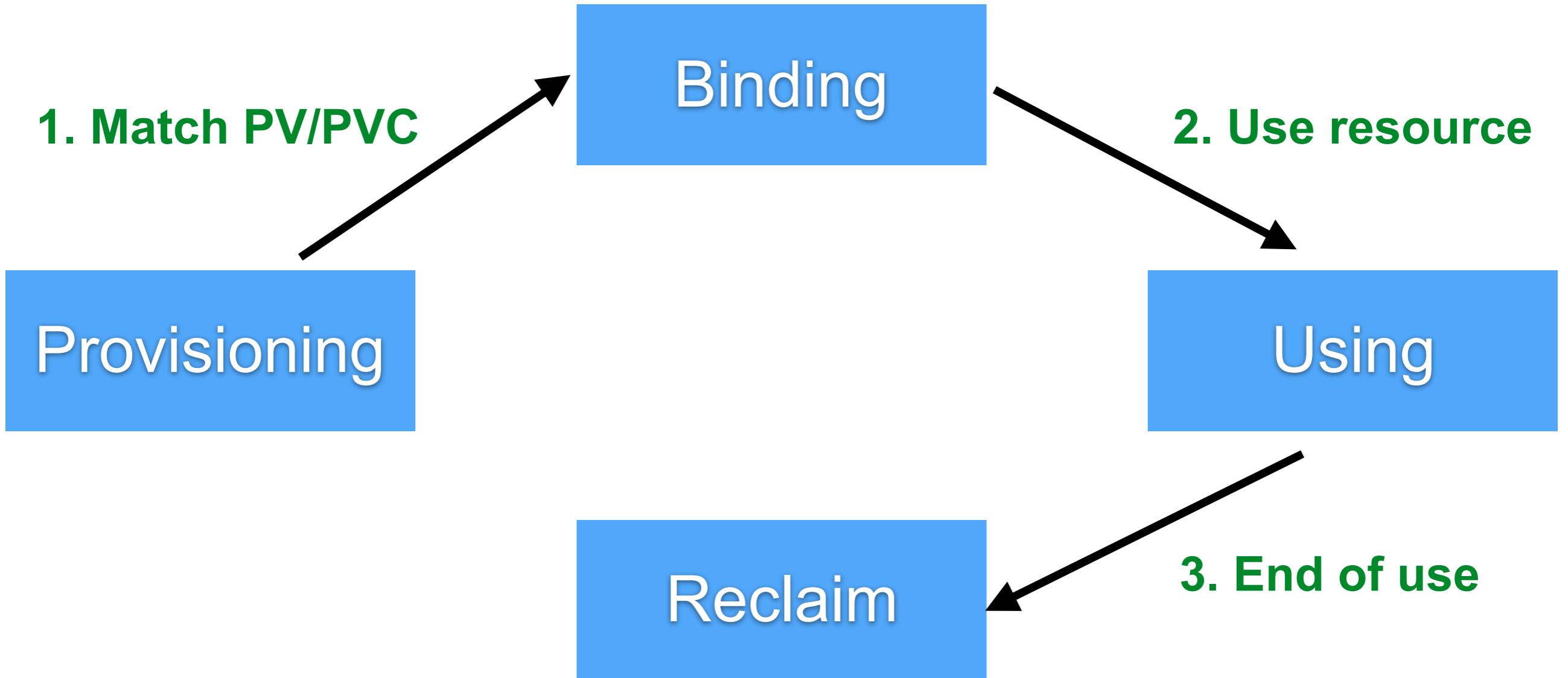
The claim has been deleted, but the resource is not yet reclaimed by the cluster

4. Failed

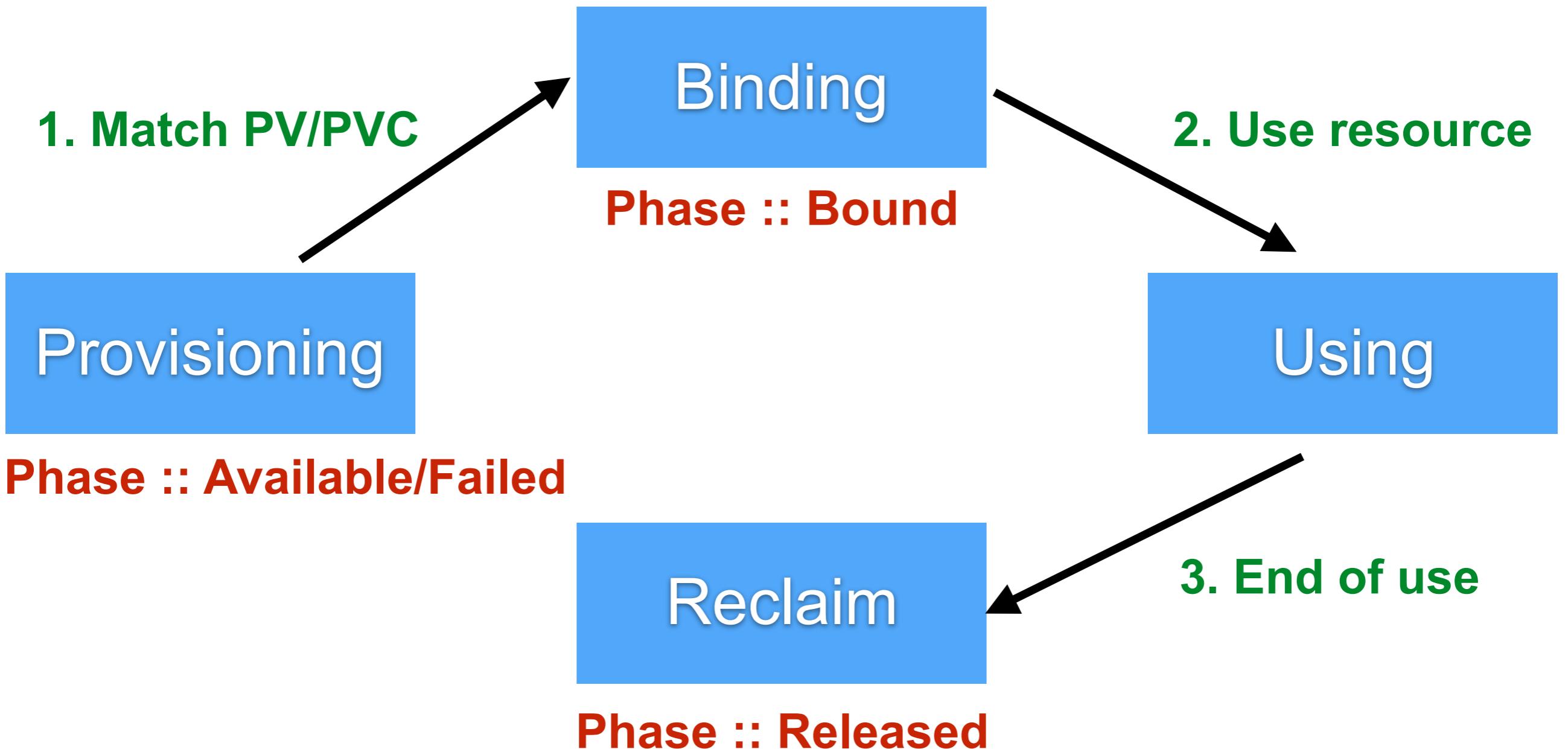
Volume has failed its automatic reclamation



Lifecycle of volume



Phase of volume



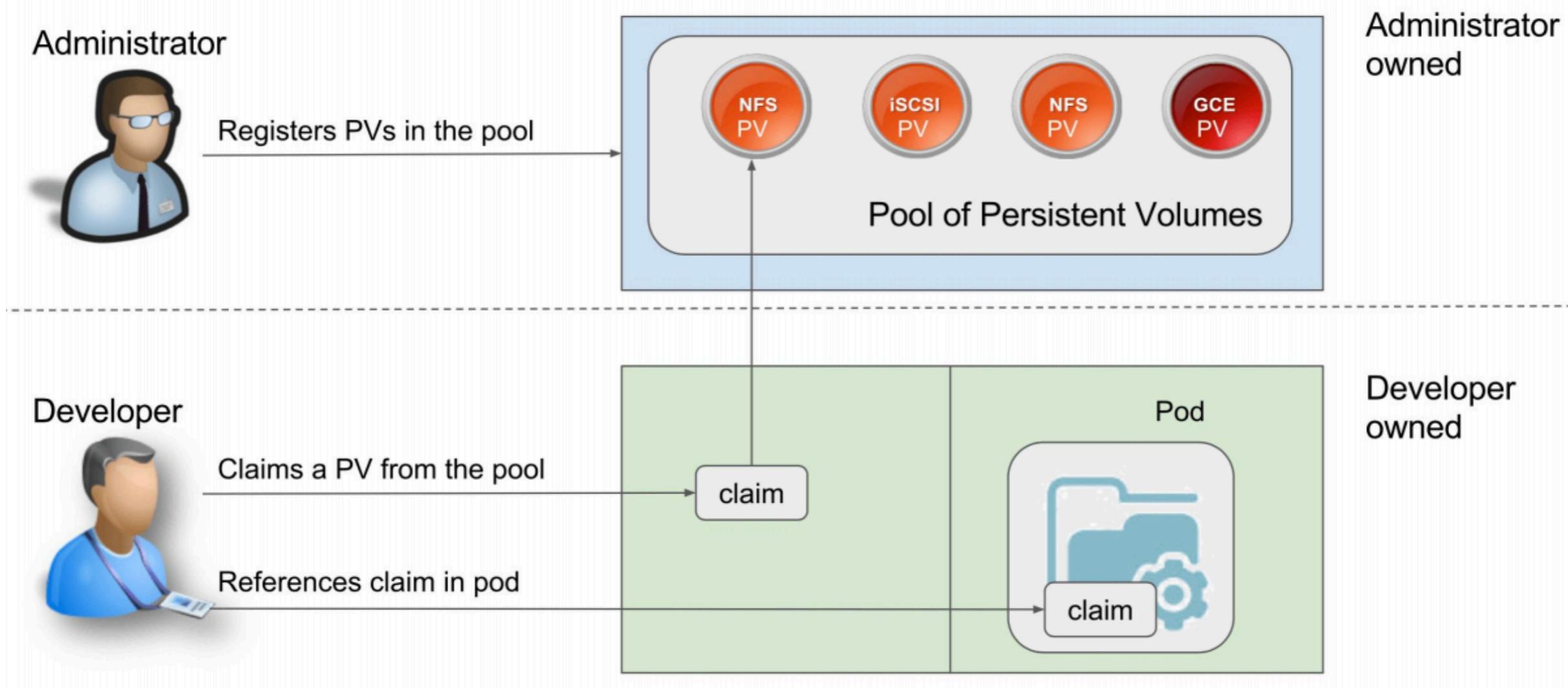
Provisioning

Static
Dynamic



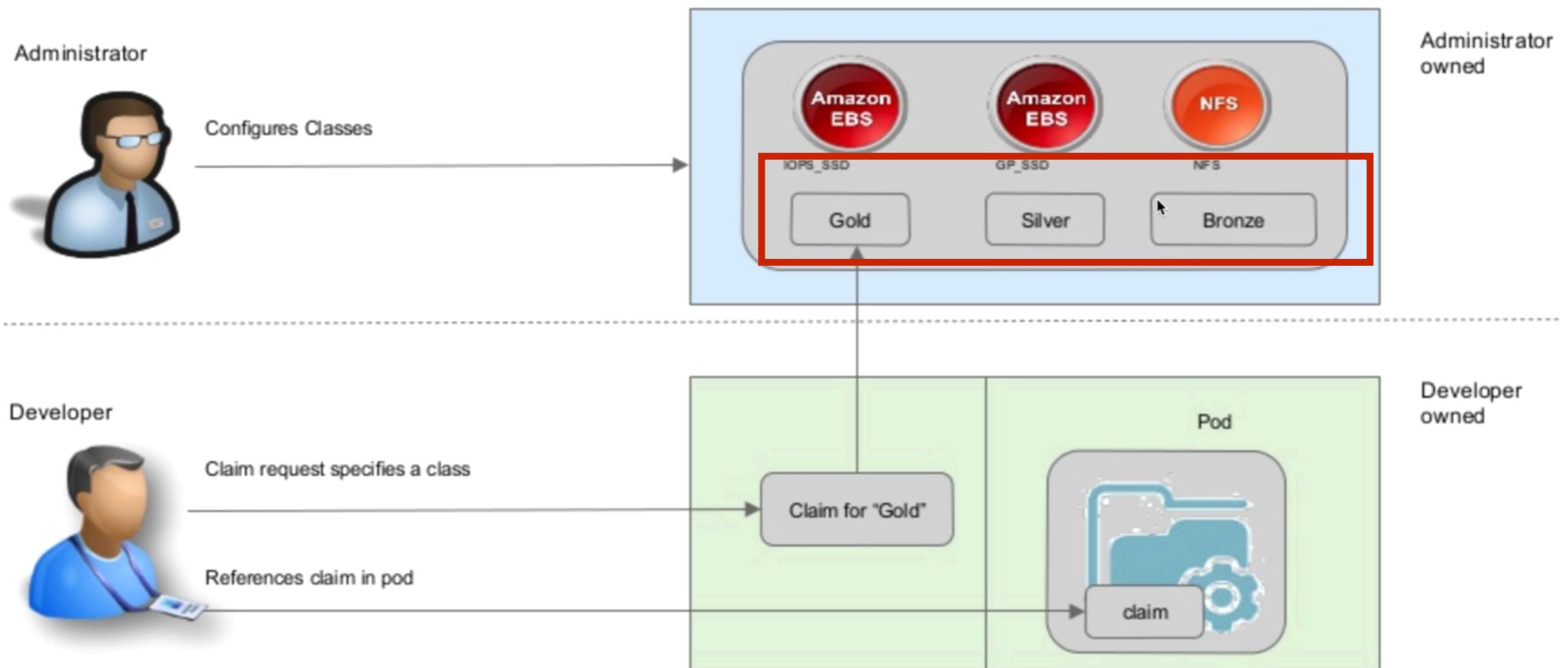
Static

Administrator create all PV
User need to know the detail of storage in each PV



Dynamic

Provisioning based on **StorageClass**
Create and configure by administrator



Using PV/PVC from Pods

```
kind: Pod
apiVersion: v1
metadata:
  name: mypod
spec:
  containers:
    - name: myfrontend
      image: dockerfile/nginx
      volumeMounts:
        - mountPath: "/var/www/html"
          name: mypd
  volumes:
    - name: mypd
      persistentVolumeClaim:
        claimName: myclaim
```



Workshop with Volume

File /working-with-volume/nfs

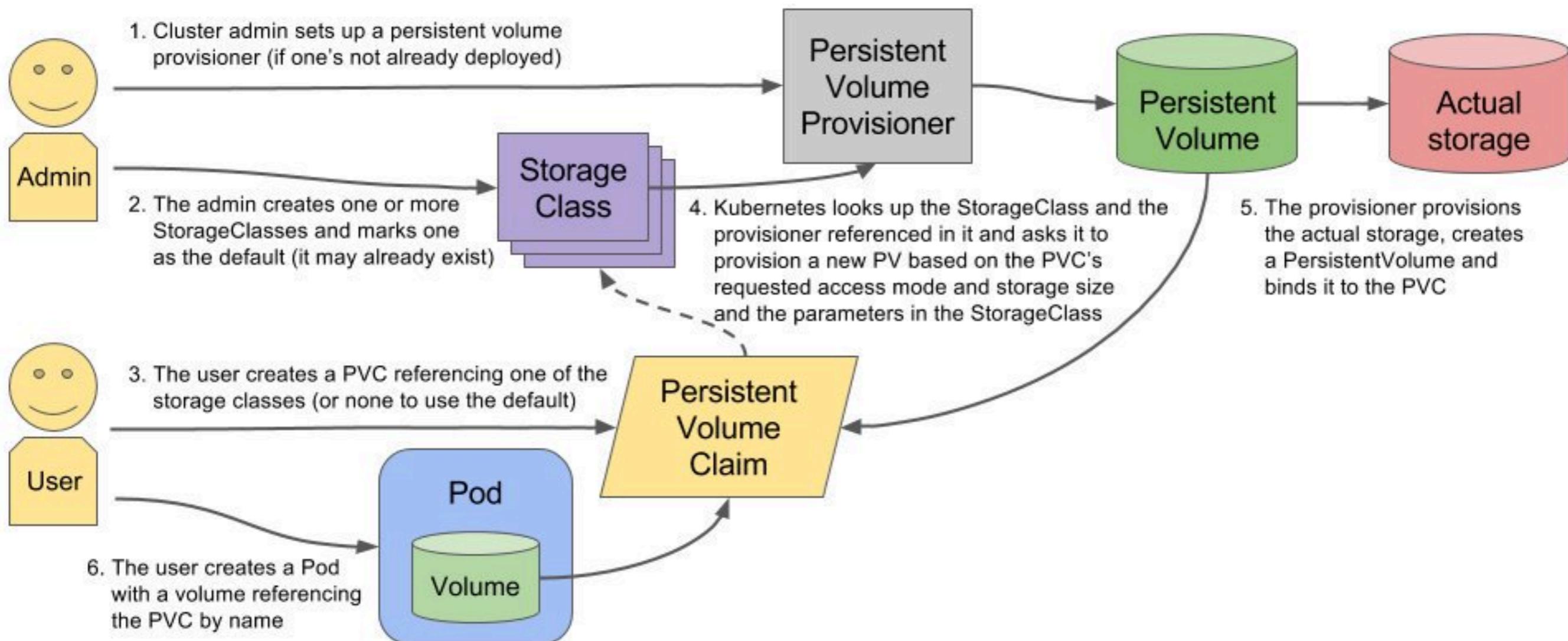


Step to working with PV/PVC

1. Create PV
2. Create PVC
3. Create Deployment
4. Testing



Step to working with PV/PVC



Setup storage engine

NFS (Network File System)
Cloud storage



Create PV and PVC

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-nfs
  labels:
    name: pvc-nfs
    environment: development
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: ""
resources:
  requests:
    storage: 500Mi
selector:
  matchLabels:
    name: pv-nfs
    environment: development
```

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-nfs
  labels:
    name: pv-nfs
    environment: development
spec:
  capacity:
    storage: 1Gi
  storageClassName: ""
  accessModes:
    - ReadWriteMany
  nfs:
    server: 10.148.0.2
    path: "/var/nfs/general"
```



Use PV with PVC from deployment

```
spec:  
  containers:  
    - name: nginx  
      image: nginx:1.8  
    ports:  
      - containerPort: 80  
    volumeMounts:  
      - name: try-nfs  
        mountPath: "/usr/share/nginx/html"  
  volumes:  
    - name: try-nfs  
      persistentVolumeClaim:  
        claimName: pvc-nfs
```

```
apiVersion: v1  
kind: PersistentVolumeClaim  
metadata:  
  name: pvc-nfs  
  labels:  
    name: pvc-nfs  
    environment: development  
spec:  
  accessModes:  
    - ReadWriteMany  
  storageClassName: ""  
  resources:  
    requests:  
      storage: 500Mi  
  selector:  
    matchLabels:  
      name: pv-nfs  
      environment: development
```



Dynamic provisioning with StorageClass



StatefulSet

<https://kubernetes.io/docs/concepts/workloads/controllers/statefulset/>



StatefulSet

Bringing the concept of ReplicaSets to **stateful** Pods

Enable running Pods in **cluster** mode

Ideal for deploy **highly** available database workload



StatefulSet for application

Stable, unique network identifiers

Stable, persistent storage

Ordered, graceful deployment and scaling

Ordered, graceful deletion and termination



Key concepts

Depend on a **Headless service** for Pods

Each Pods get s DNS name accessible to other Pods

Leverage PV and PVC

Each Pods is suffix with a predictable
(mysql-01, mysql-02)



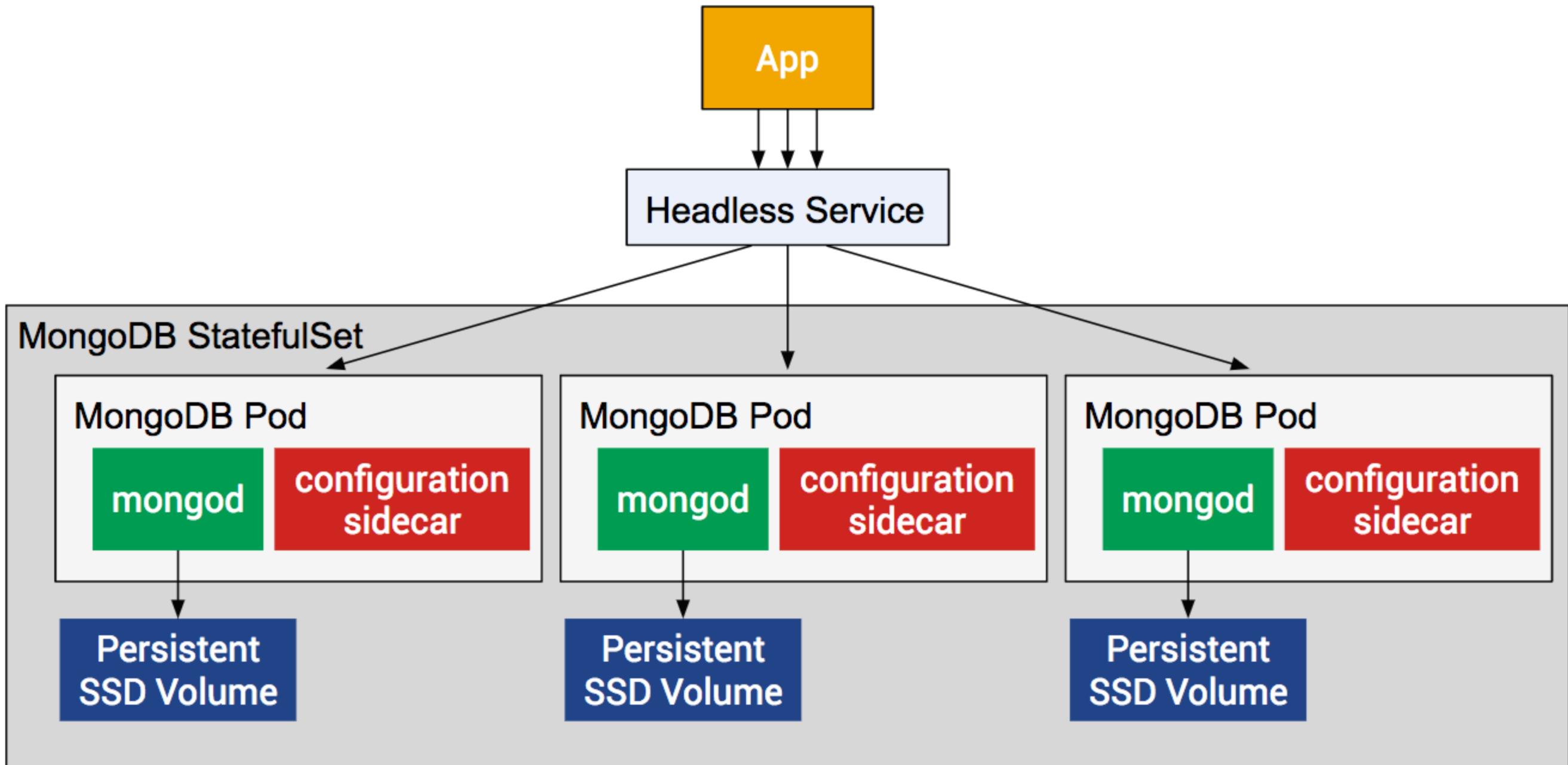
Key concepts

Pods are created sequentially

Pods are terminated in LiFo (Last in, First out)



StatefulSet



<https://kubernetes.io/blog/2017/01/running-mongodb-on-kubernetes-with-statefulsets/>



Key concepts

Pods are created sequentially

Pods are terminated in LiFo (Last in, First out)



Workshop with StatefulSet

File /working-with-stateful-set



Application Deployment Strategies



Strategies to deploy

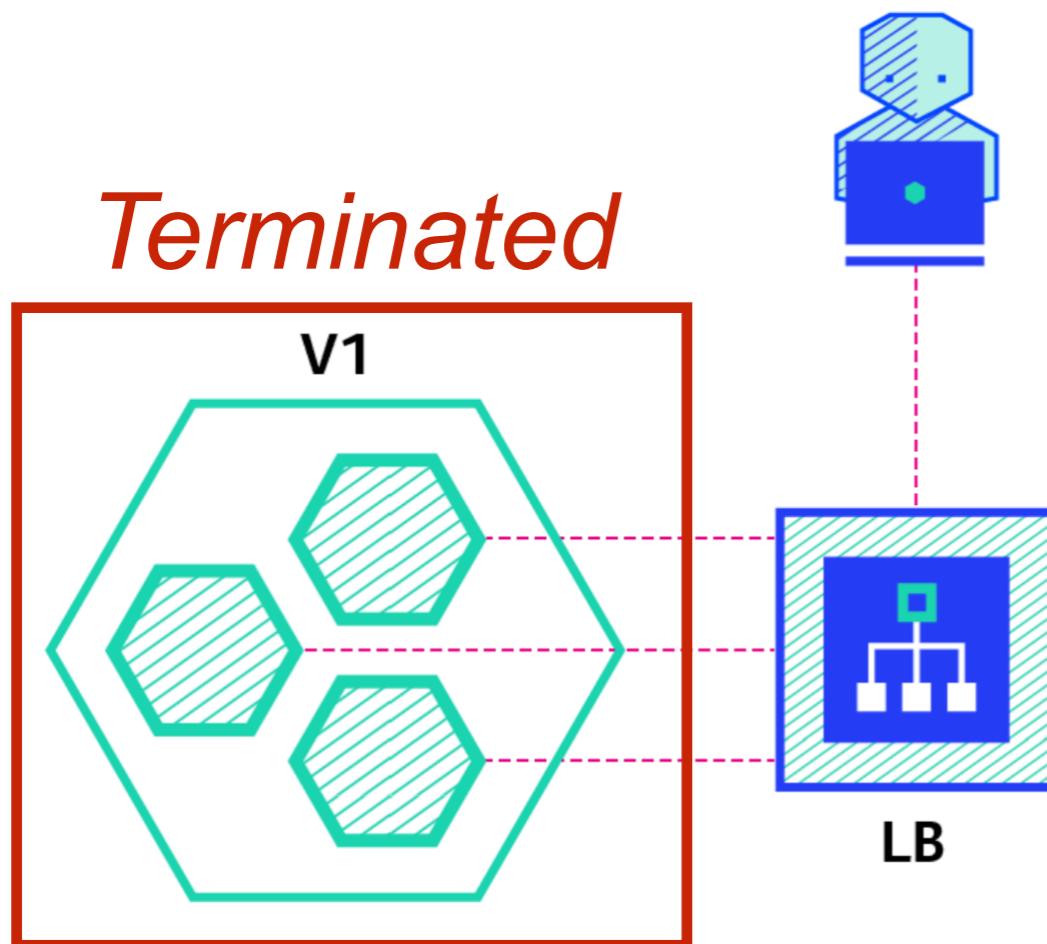
Recreate
Ramped
Blue/Green
Canary
A/B testing
Shadow

<https://thenewstack.io/deployment-strategies/>



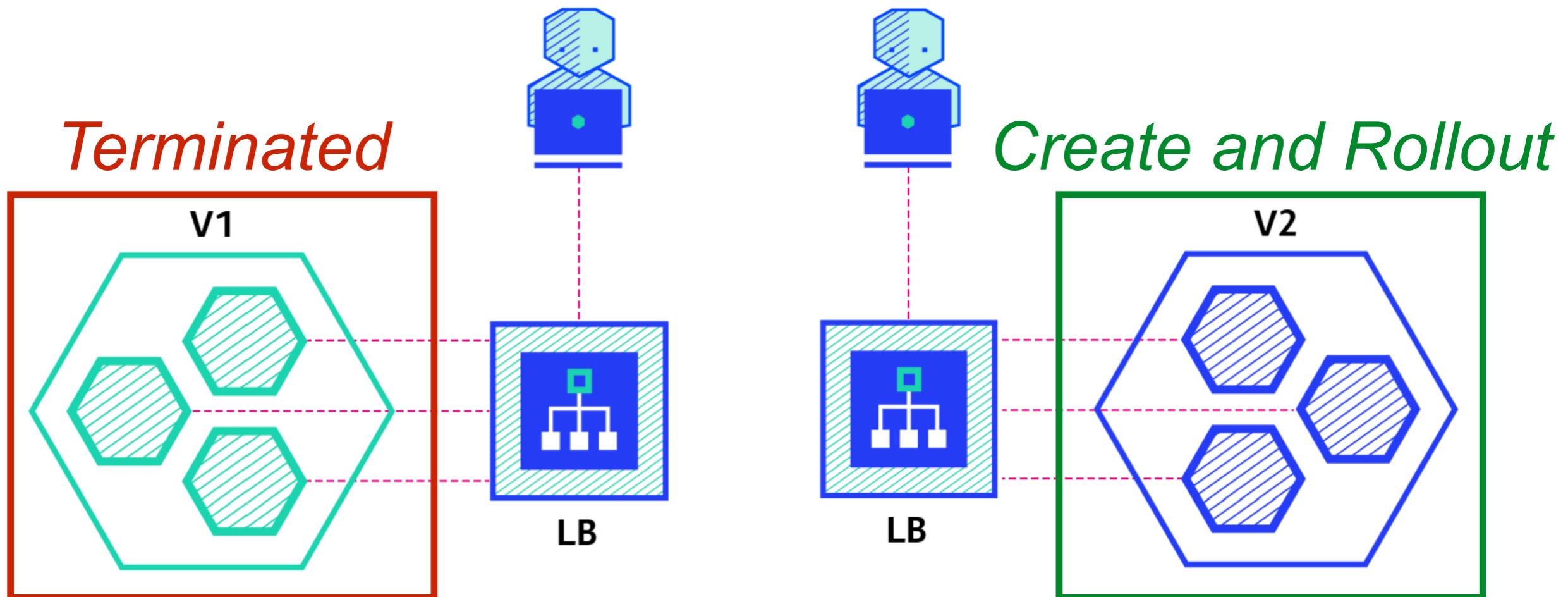
1. Recreate

Version A is terminated then version B is rollout



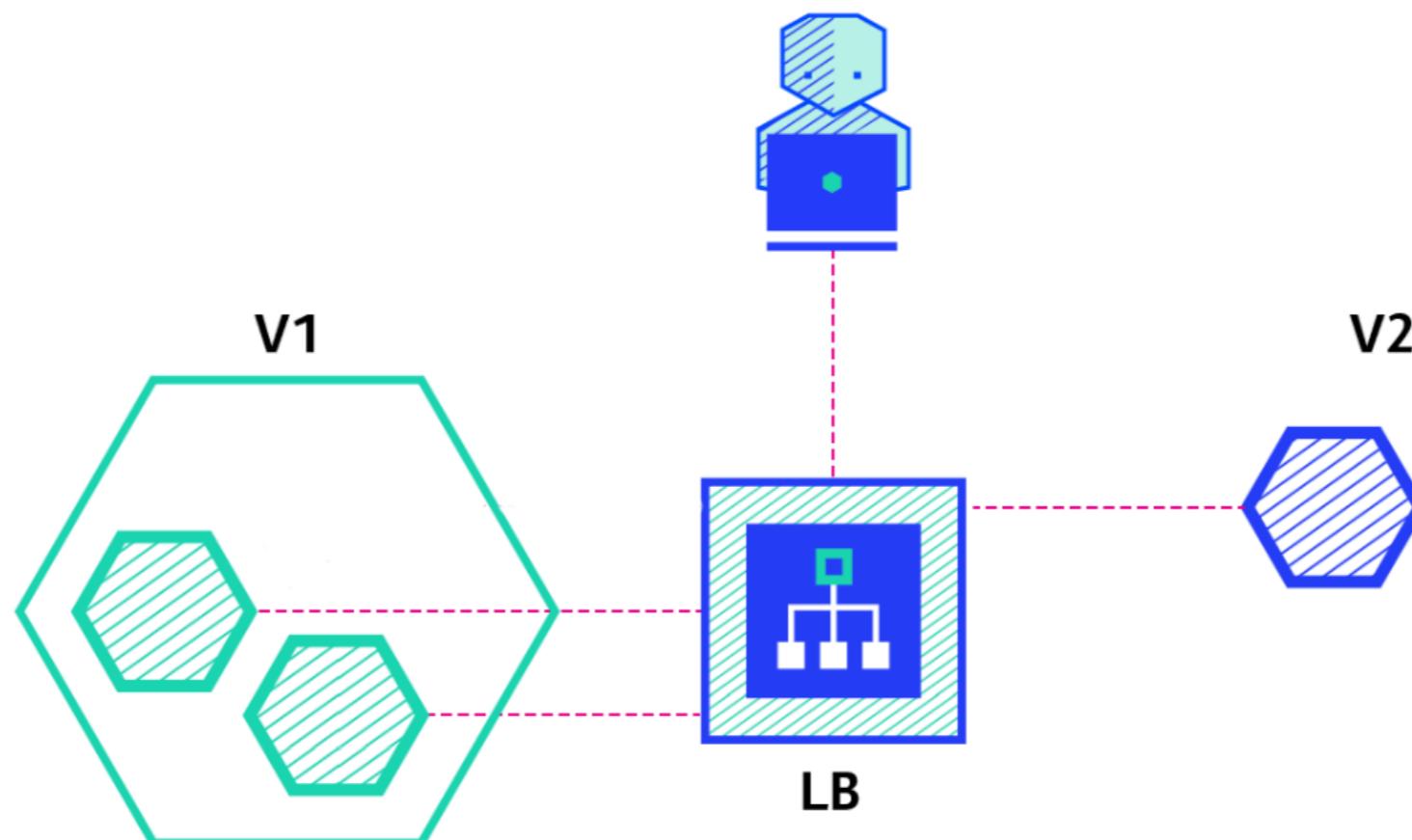
1. Recreate

Version A is terminated then version B is rollout



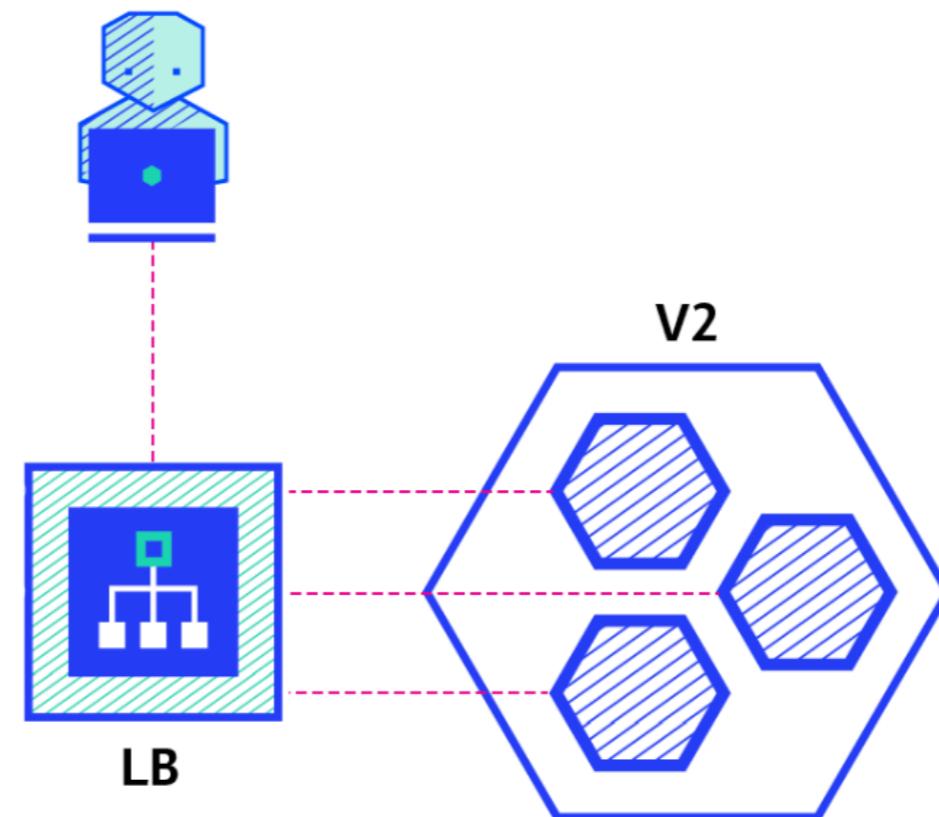
2. Ramped

Slow roll out by replace instance one-by-one



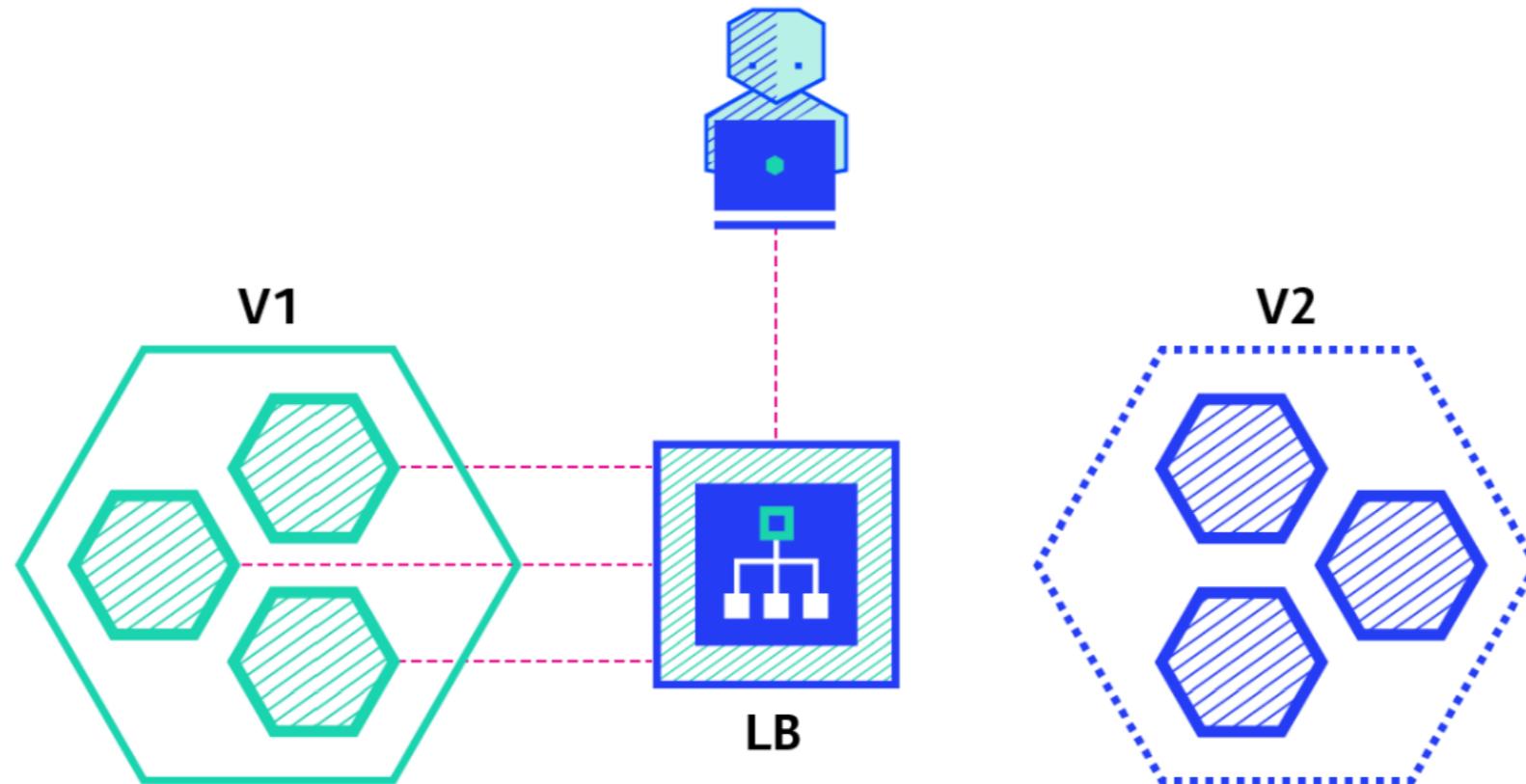
2. Ramped

Slow roll out by replace instance one-by-one



3. Blue/Green

Current version is called **Blue**
New version is called **Green**

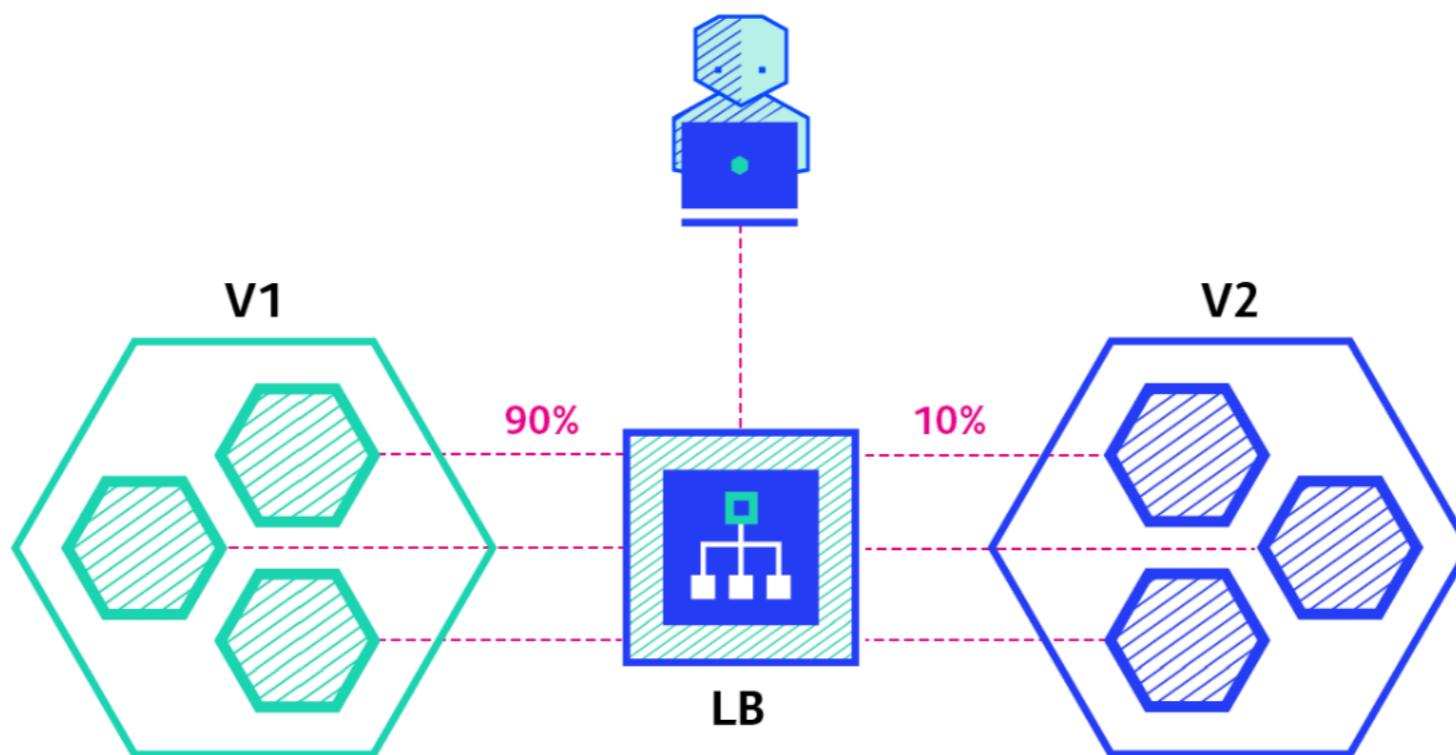


4. Canary

Shift production traffic from version A to B

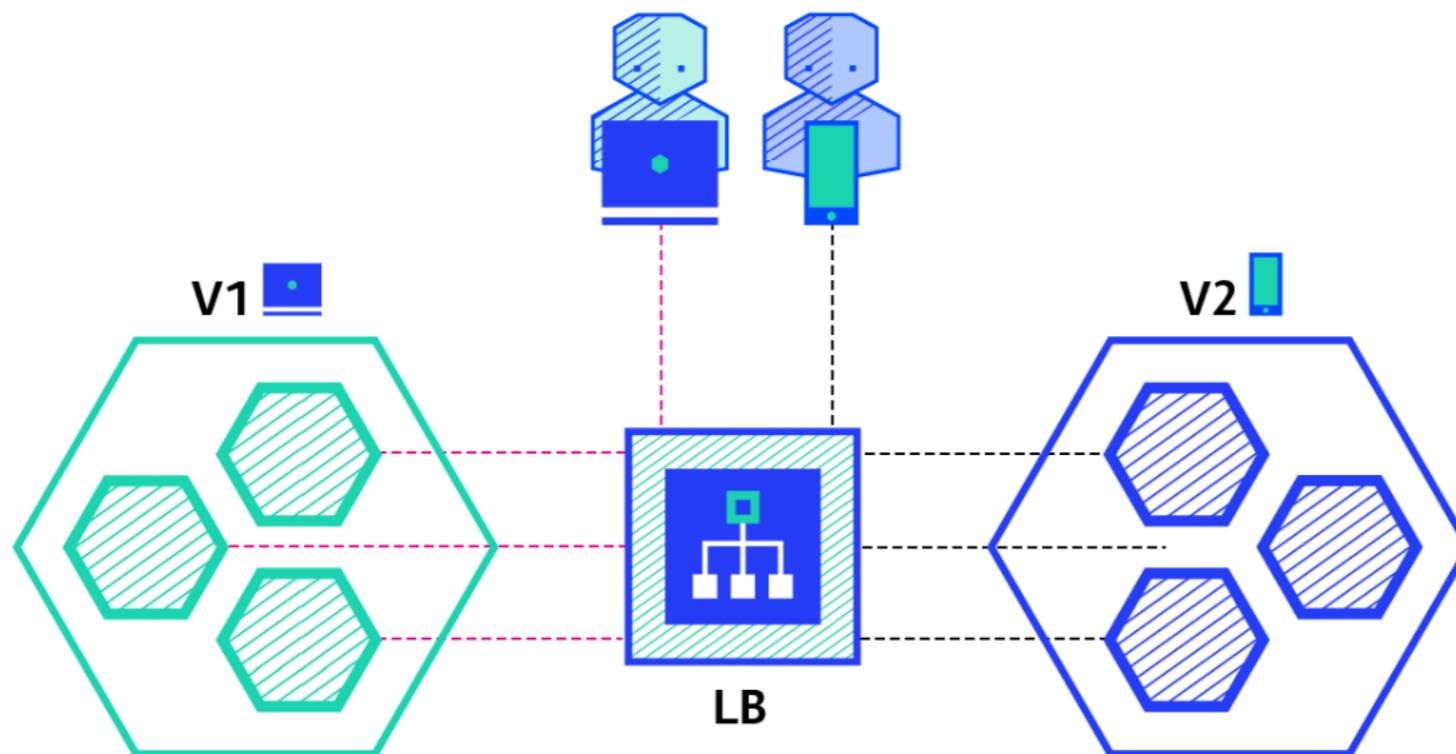
Traffic is split based-on weight

Use when tests are lacking/not reliable and less confident in system



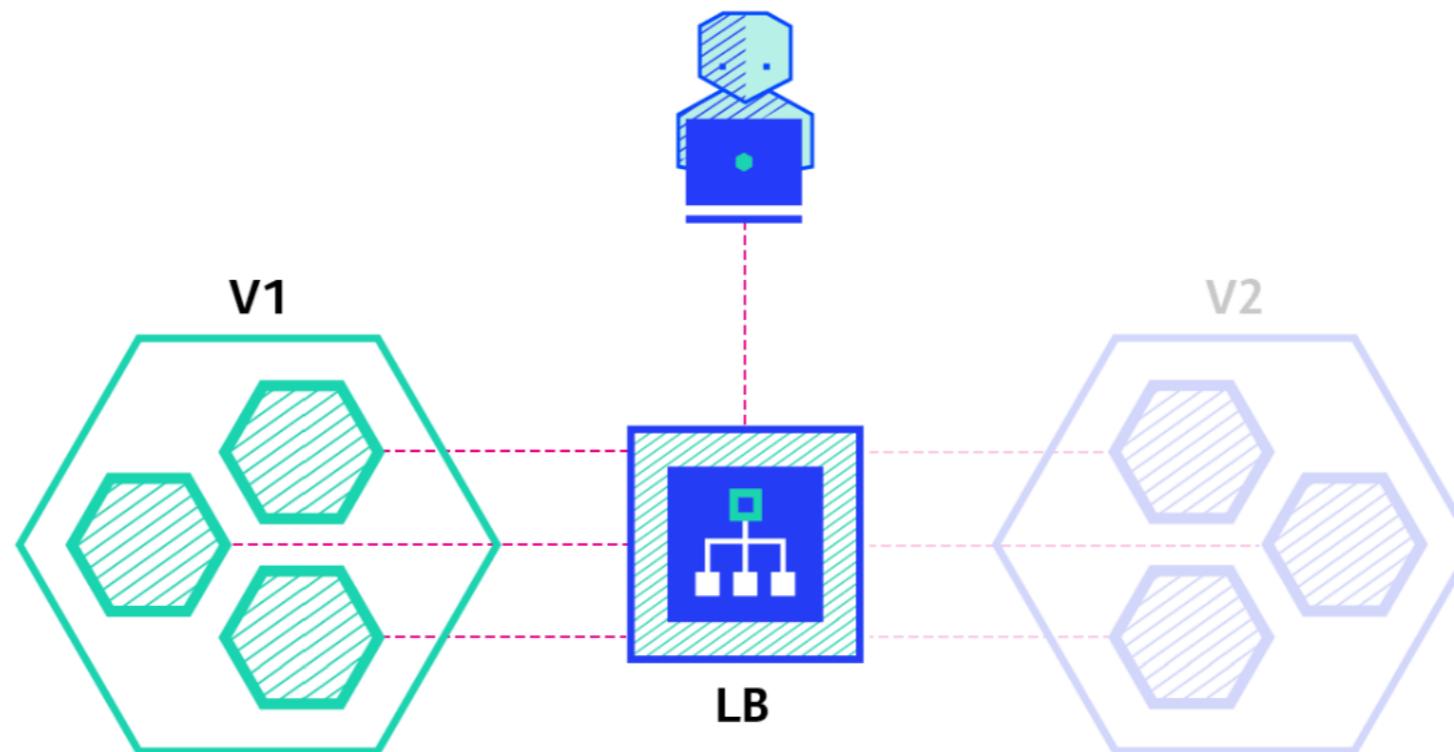
5. A/B testing

Routing the subset of users to new services under the specific condition



6. Shadow

Release version B alongside version A
Send request's A to B without production impact



DEPLOYMENT STRATEGIES

When it comes to production, a ramped or blue/green deployment is usually a good fit, but proper testing of the new platform is necessary.

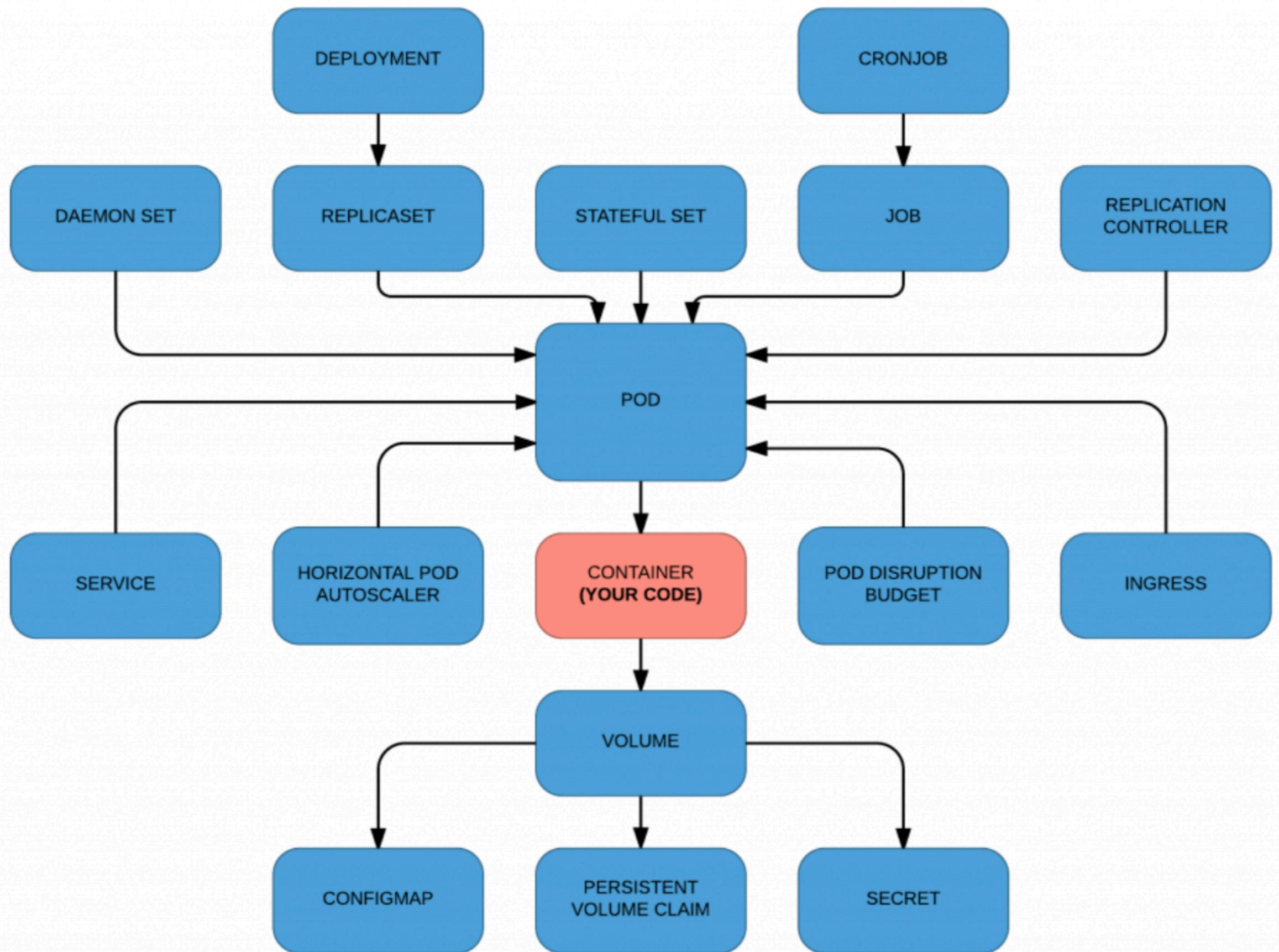
Blue/green and shadow strategies have more impact on the budget as it requires double resource capacity. If the application lacks in tests or if there is little confidence about the impact/stability of the software, then a canary, a/b testing or shadow release can be used.

If your business requires testing of a new feature amongst a specific pool of users that can be filtered depending on some parameters like geolocation, language, operating system or browser features, then you may want to use the a/b testing technique.



| Strategy | ZERO DOWNTIME | REAL TRAFFIC TESTING | TARGETED USERS | CLOUD COST | ROLLBACK DURATION | NEGATIVE IMPACT ON USER | COMPLEXITY OF SETUP |
|--|---------------|----------------------|----------------|------------|-------------------|-------------------------|---------------------|
| RECREATE version A is terminated then version B is rolled out | ✗ | ✗ | ✗ | ■ ■ ■ | ■ ■ ■ | ■ ■ ■ | □ □ □ |
| RAMPED version B is slowly rolled out and replacing version A | ✓ | ✗ | ✗ | ■ ■ ■ | ■ ■ ■ | ■ □ □ | ■ □ □ |
| BLUE/GREEN version B is released alongside version A, then the traffic is switched to version B | ✓ | ✗ | ✗ | ■ ■ ■ | □ □ □ | ■ ■ □ | ■ ■ □ |
| CANARY version B is released to a subset of users, then proceed to a full rollout | ✓ | ✓ | ✗ | ■ ■ ■ | ■ □ □ | ■ □ □ | ■ ■ □ |
| A/B TESTING version B is released to a subset of users under specific condition | ✓ | ✓ | ✓ | ■ ■ ■ | ■ □ □ | ■ □ □ | ■ ■ ■ |
| SHADOW version B receives real world traffic alongside version A and doesn't impact the response | ✓ | ✓ | ✗ | ■ ■ ■ | □ □ □ | □ □ □ | ■ ■ ■ |



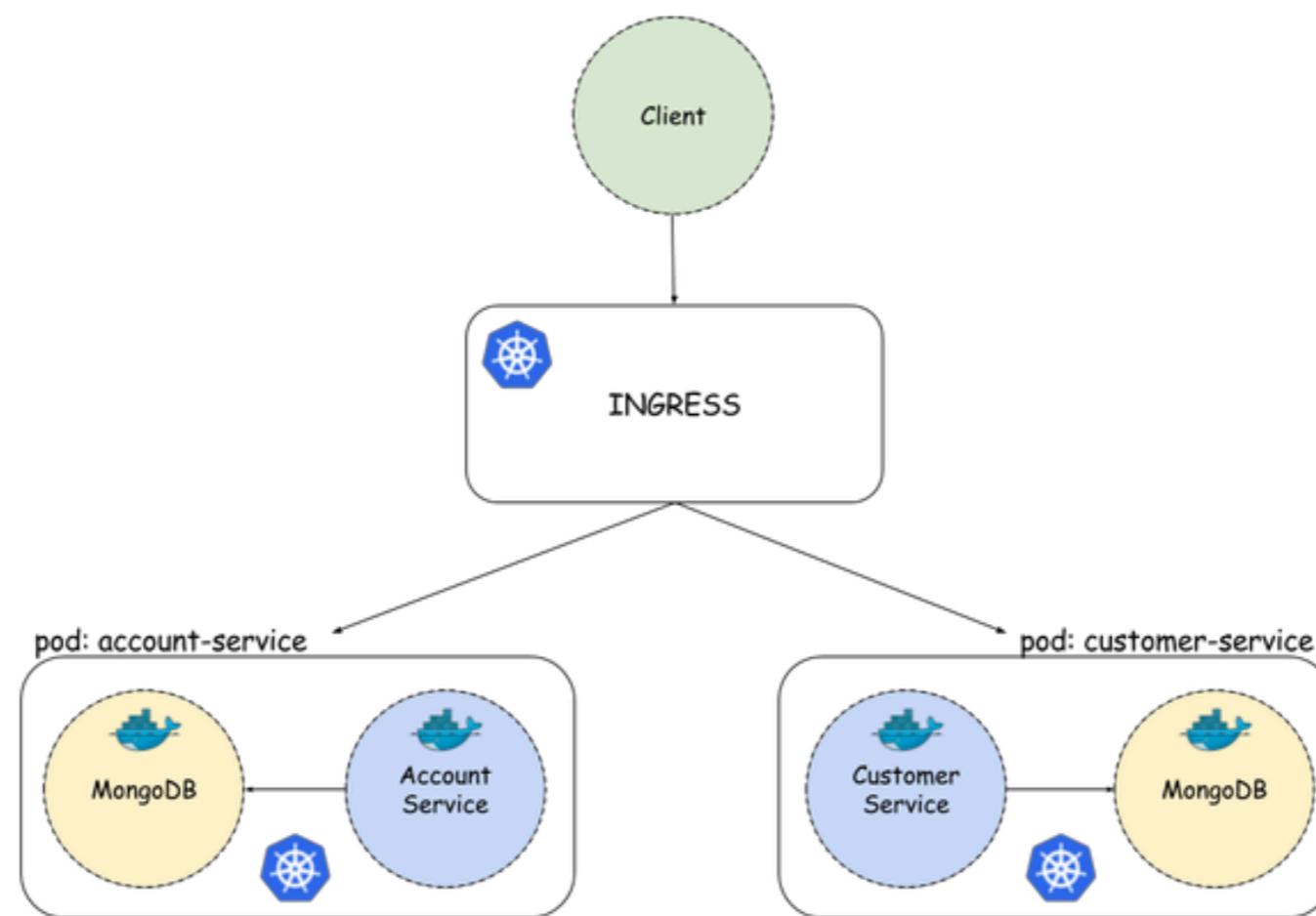


Ingress Network



Ingress Network

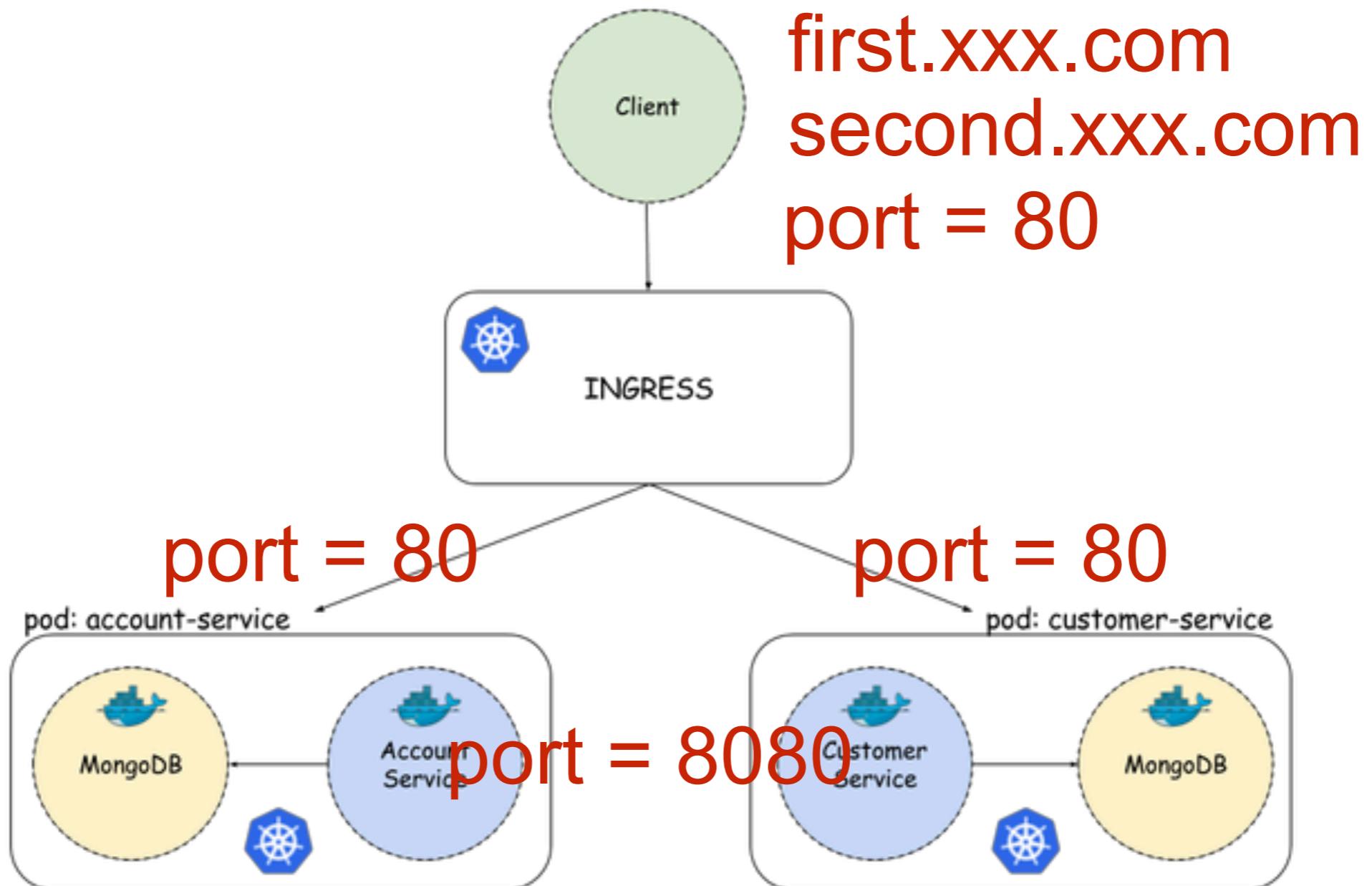
How to handle multiple services in same port ?
How to limit protocol to access ?



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



Ingress Network



Create ingress

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: ingresswebtest
spec:
  rules:
    - host: first.xxx.com
      http:
        paths:
          - backend:
              serviceName: spring-boot-service
              servicePort: 80
    - host: second.xxx.com
      http:
        paths:
          - backend:
              serviceName: spring-boot-service
              servicePort: 80
```



Service

```
apiVersion: v1
kind: Service
metadata:
  name: spring-boot-service
spec:
  selector:
    app: spring-boot-service
  type: NodePort
  ports:
    - port: 80
      name: http
      targetPort: 8080
      protocol: TCP
```



Workshop

Ingress Network

File /working-with-java/05-ingress



Liveness and Readiness probe

<https://kubernetes.io/docs/tasks/configure-pod-container/configure-liveness-readiness-probes/>



Type of health checks

Readiness
Liveness



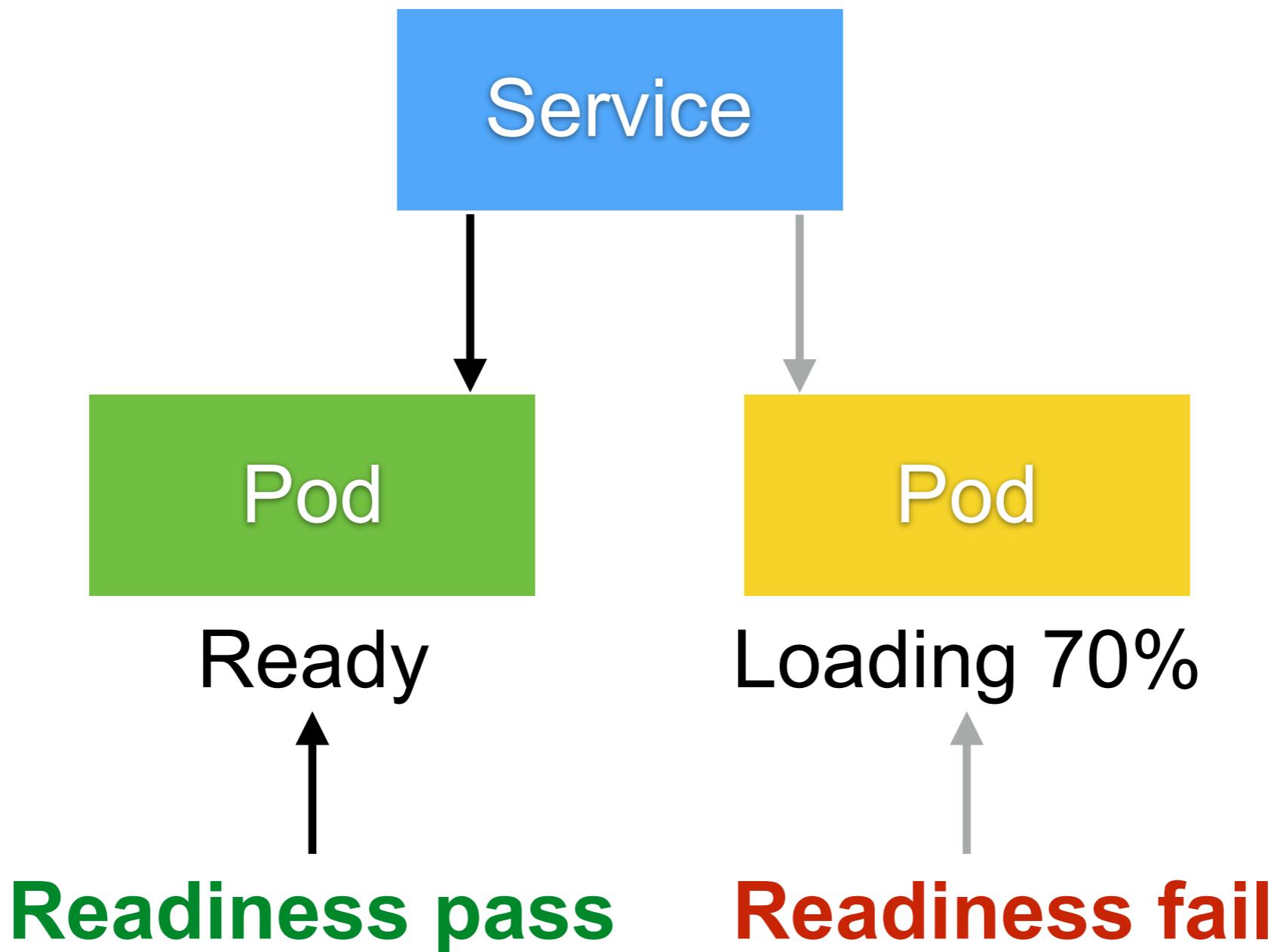
Readiness

Readiness probes are designed to let Kubernetes know when your app is ready to serve traffic.

If a readiness probe starts to **fail**, Kubernetes stops sending traffic to the pod until it passes.



Readiness



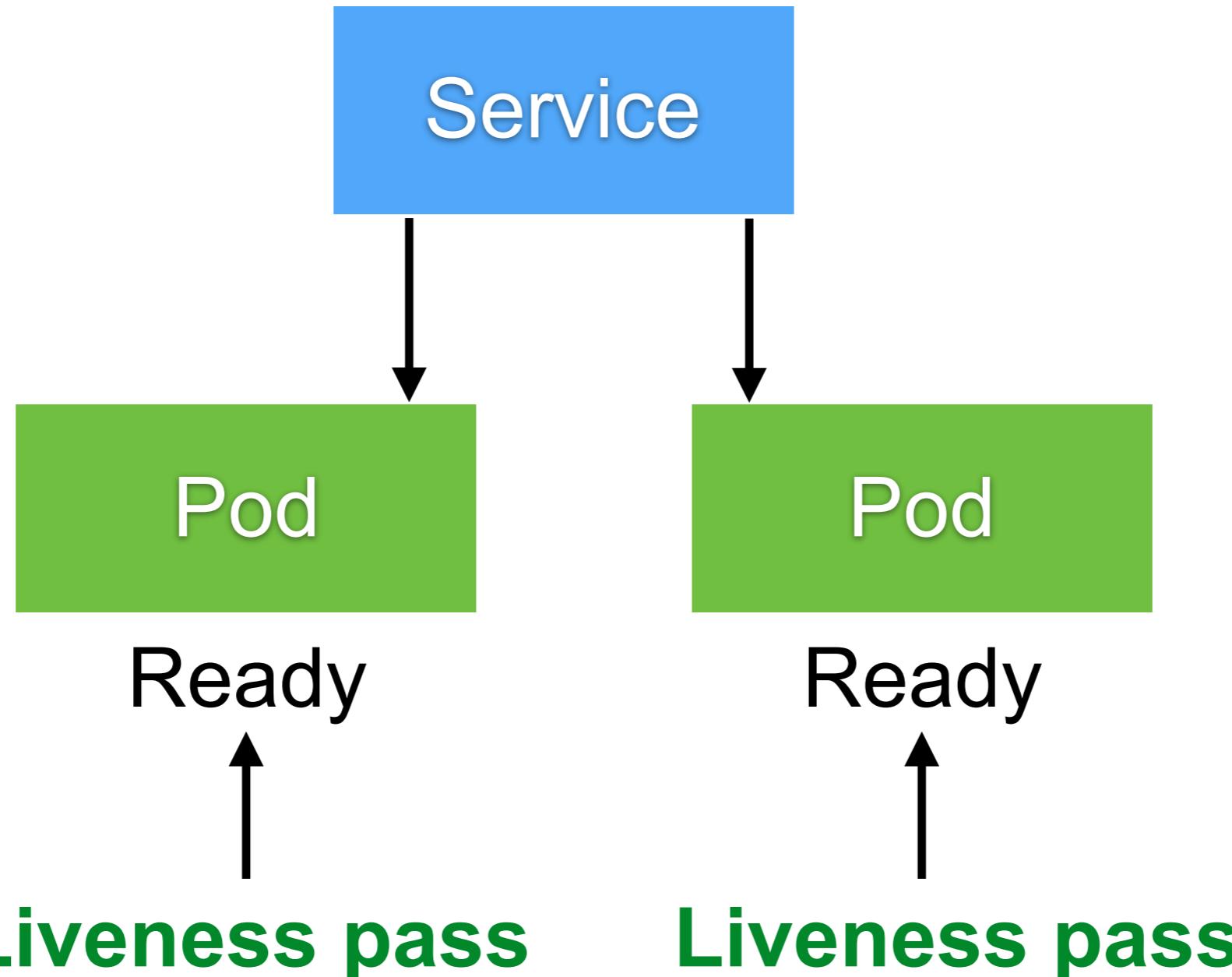
Liveness

Liveness probes let Kubernetes know if your app is alive or dead. If your app is **alive**, then Kubernetes leaves it alone

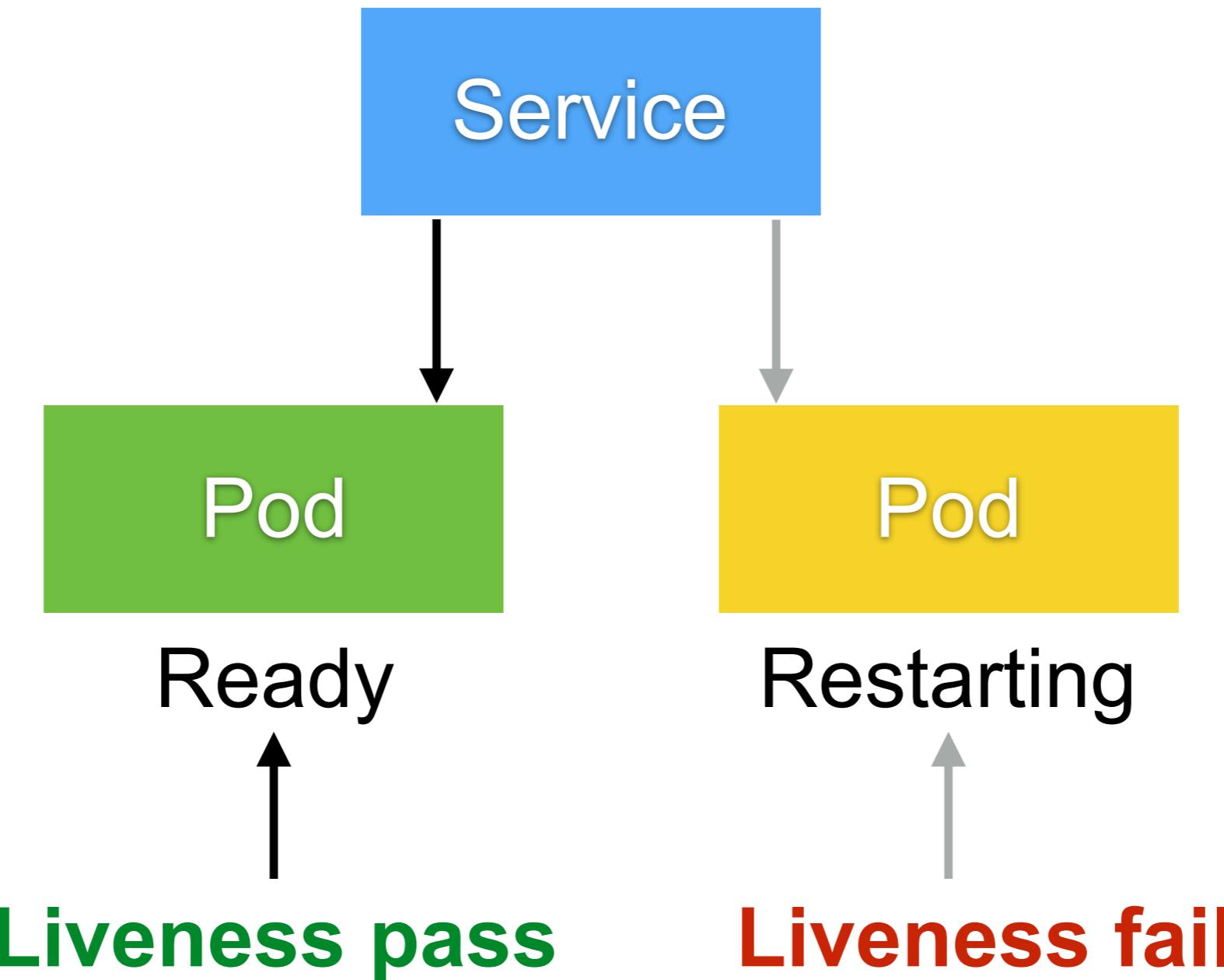
If your app is **dead**, Kubernetes removes the Pod and starts a new one to replace it.



Liveness (Pass)



Liveness (Fail)



Type of probes

HTTP
Command
TCP



HTTP probe

```
spec:  
  containers:  
    - name: liveness  
      image: k8s.gcr.io/liveness  
      args:  
        - /server  
      livenessProbe:  
        httpGet:  
          path: /healthz  
          port: 8080  
        httpHeaders:  
          - name: X-Custom-Header  
            value: Awesome
```



Command probe

```
spec:  
  containers:  
    - name: liveness  
      image: k8s.gcr.io/busybox  
      livenessProbe:  
        exec:  
          command:  
            - cat  
            - /tmp/healthy
```



TCP probe

```
spec:  
  containers:  
    - name: goproxy  
      image: k8s.gcr.io/goproxy:0.1  
      ports:  
        - containerPort: 8080  
      readinessProbe:  
        tcpSocket:  
          port: 8080  
      initialDelaySeconds: 5  
      periodSeconds: 10  
    livenessProbe:  
      tcpSocket:  
        port: 8080  
      initialDelaySeconds: 15  
      periodSeconds: 20
```



Configuring probes

initialDelaySeconds
periodSeconds
timeoutSeconds
successThreshold
failureThreshold



Configuring probes

initialDelaySeconds

Number of seconds after the container has started before liveness or readiness probes are initiated.

periodSeconds

How often (in seconds) to perform the probe.
Default to 10 seconds. Minimum value is 1.

timeoutSeconds

Number of seconds after which the probe times out.
Defaults to 1 second. Minimum value is 1.



Configuring probes

successThreshold

Minimum consecutive successes for the probe to be considered successful after having failed. **Defaults to 1.** Must be 1 for liveness. Minimum value is 1.

failureThreshold

When a Pod starts and the probe fails, Kubernetes will try failureThreshold times before giving up. Giving up in case of liveness probe means restarting the Pod. In case of readiness probe the Pod will be marked Unready. **Defaults to 3.** Minimum value is 1.



Workshop Ingress Network

File /workshop/readiness-and-liveness



Wordpress and MySQL

File /workshop-wordpress

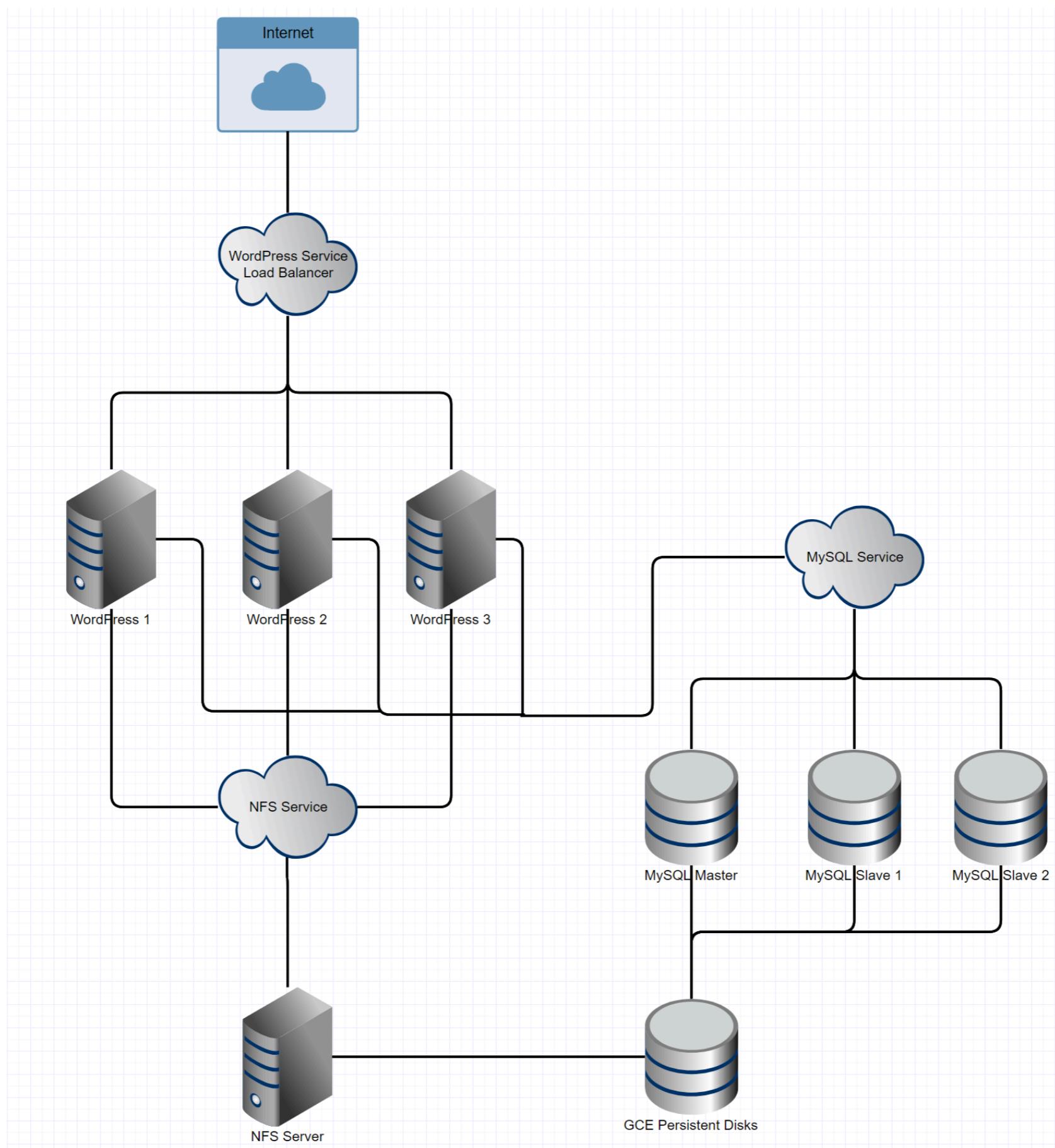


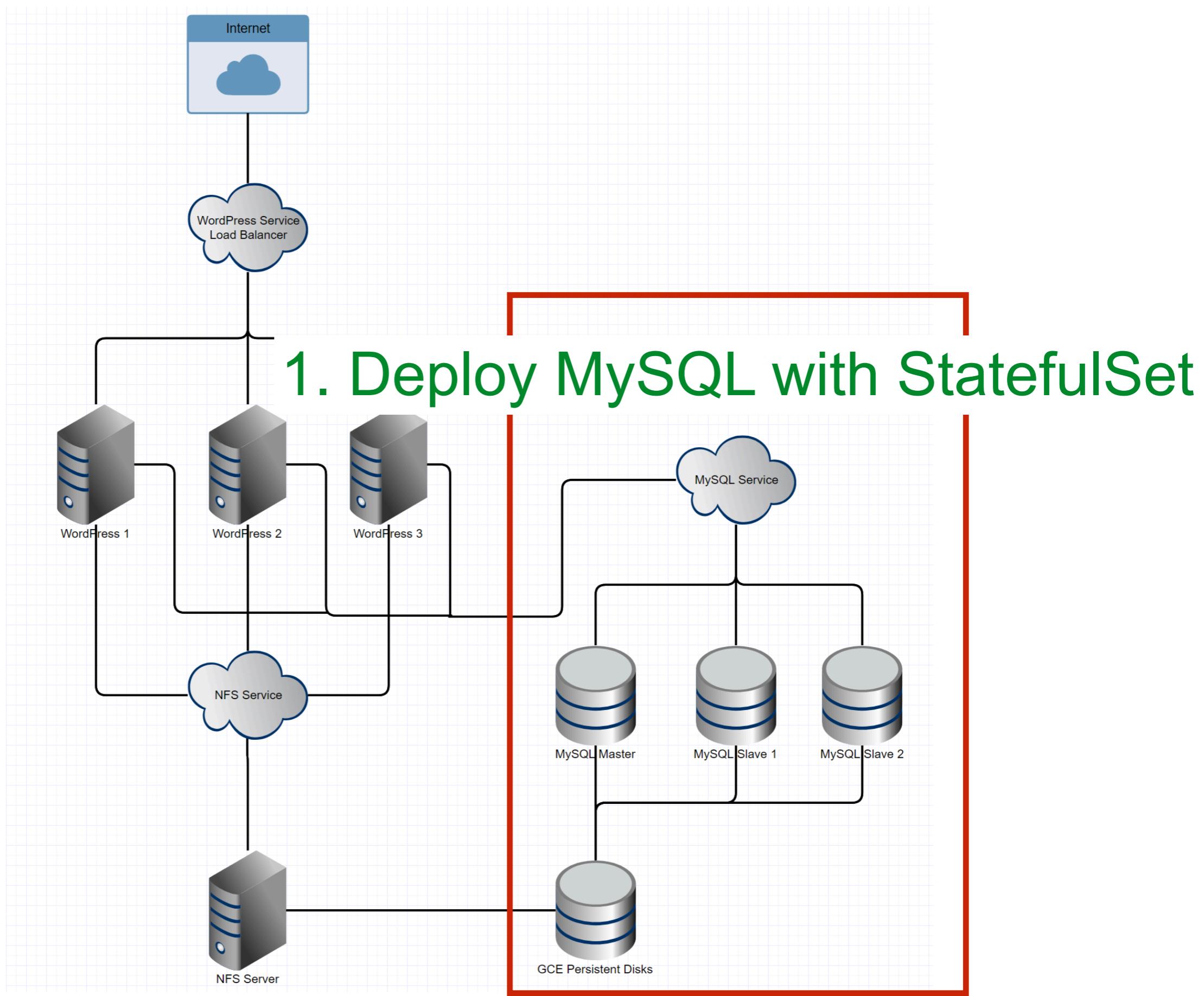
Architecture

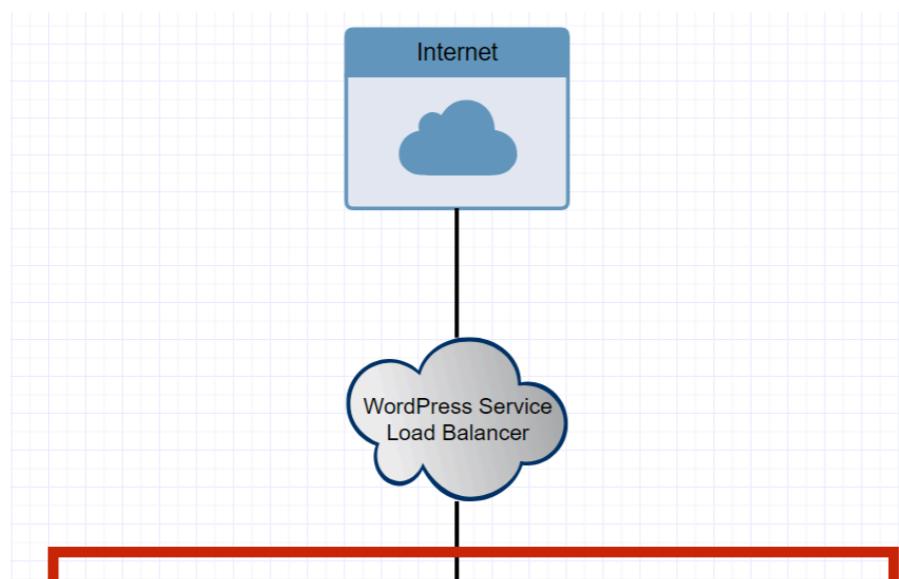
Storage for Wordpress application file with **NFS**
Database cluster with **MySQL**

Use Kubernetes load balance and **service** networking

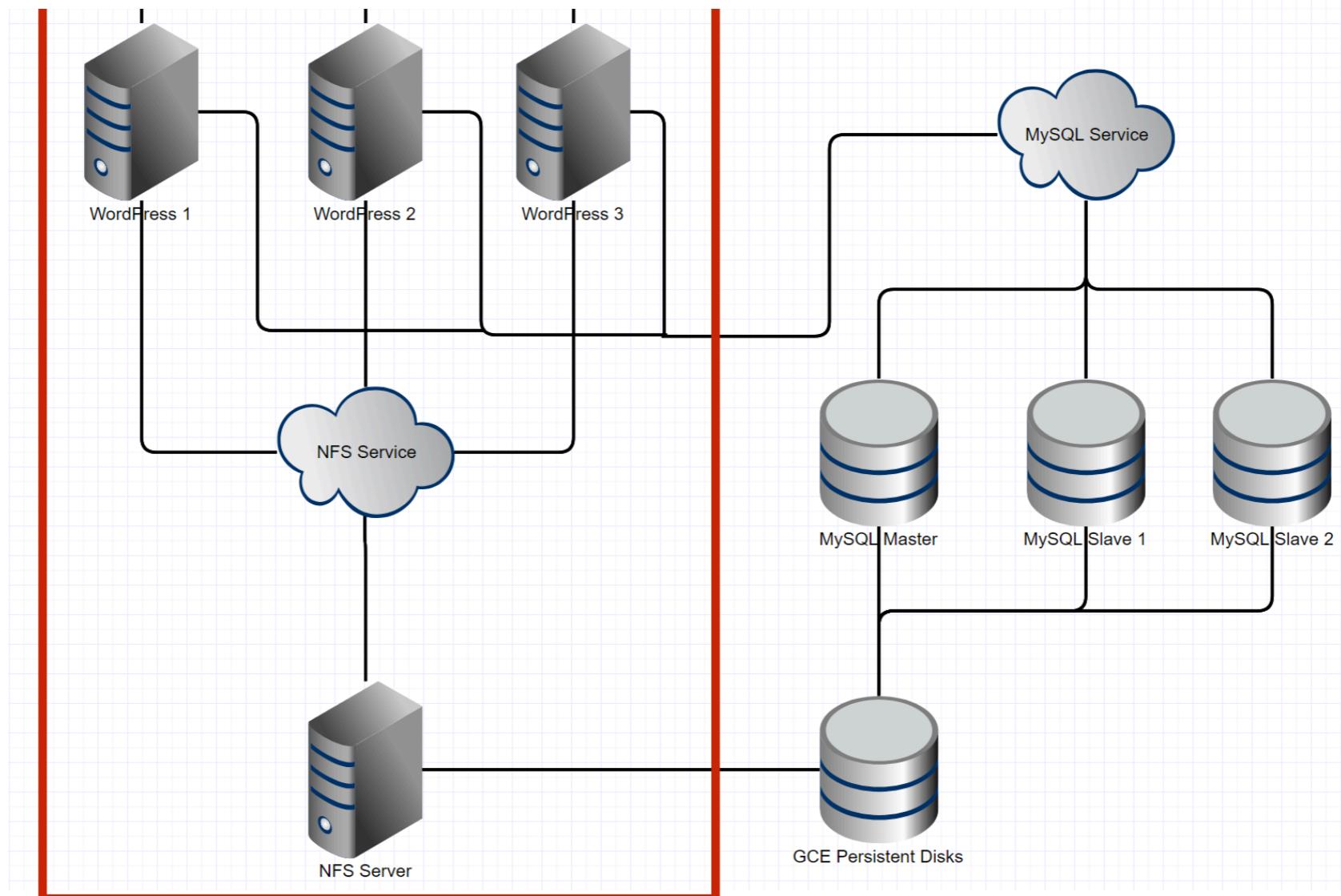




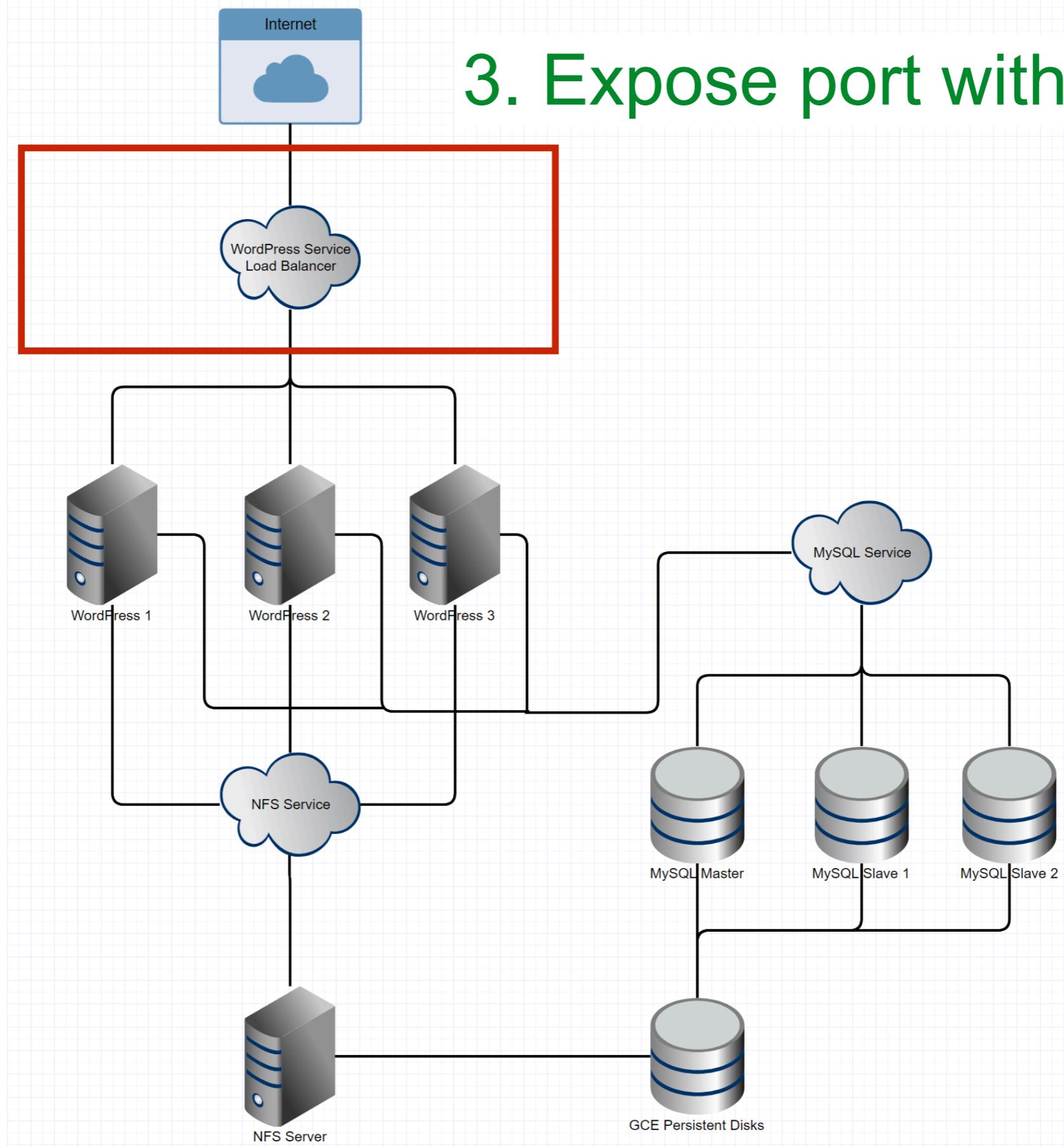




2. Share wordpress files with NFS



3. Expose port with service



Logging

<https://kubernetes.io/docs/concepts/cluster-administration/logging/>



Logging Levels

Pods
Node
Cluster



Demo Logging with Papertrail

File /workshop/logging



Job and Cronjob



Thank you

