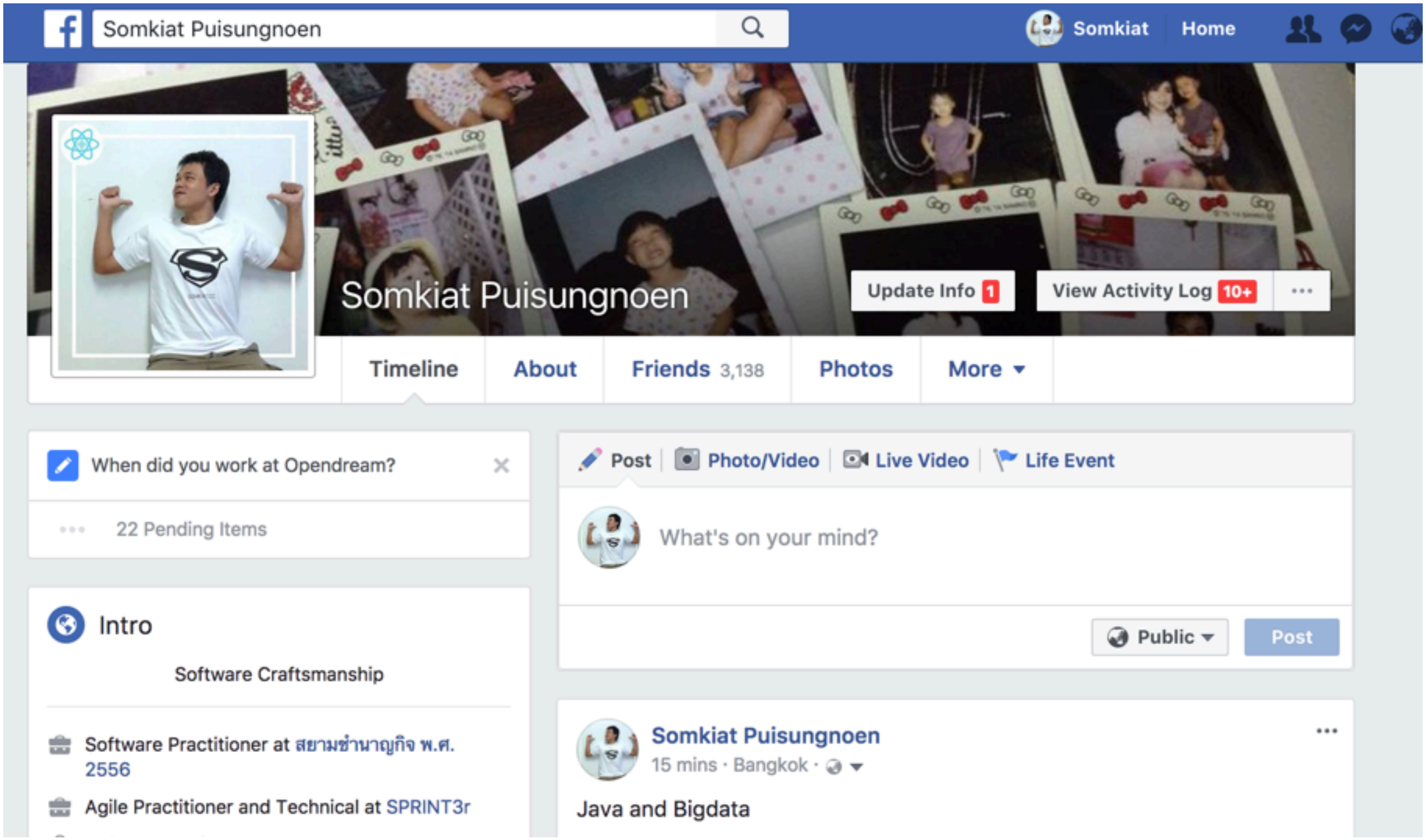
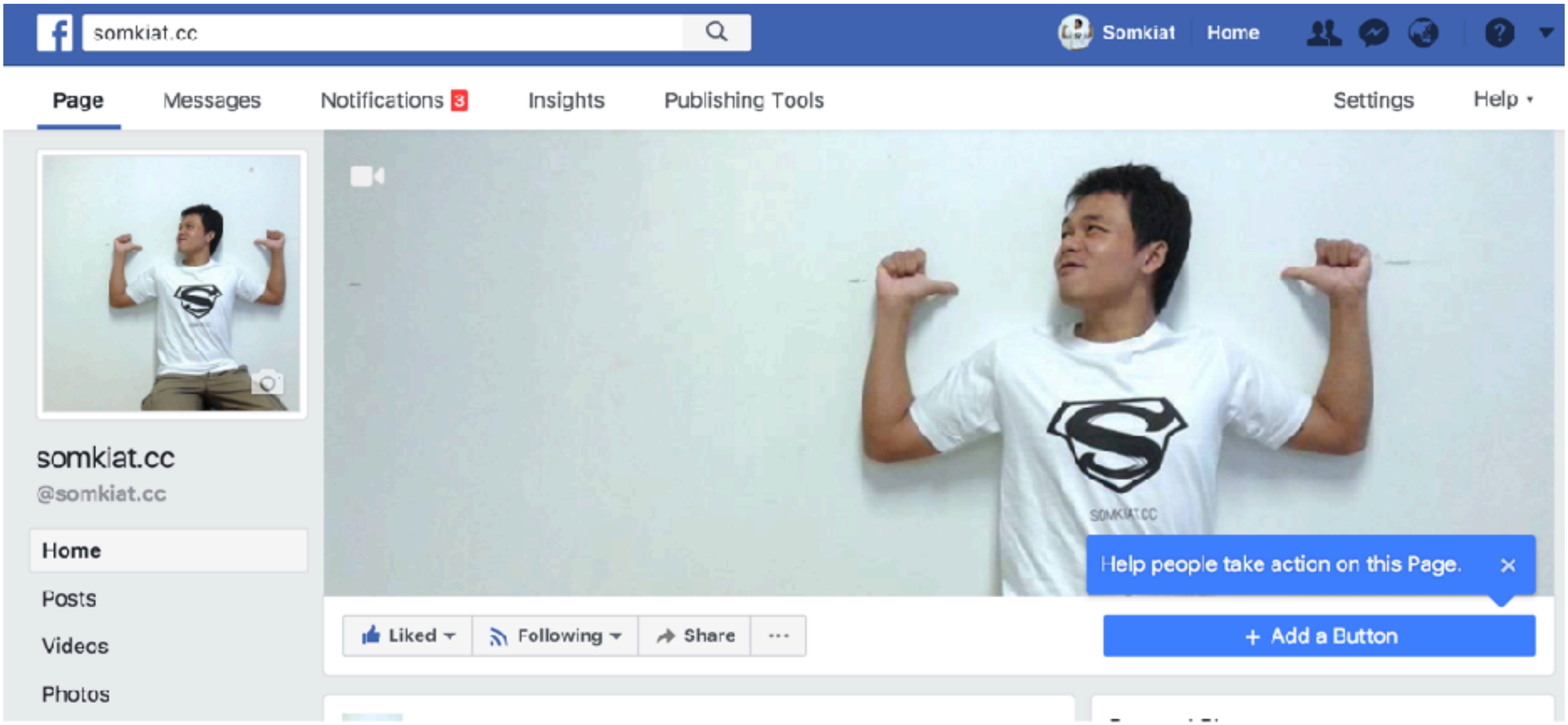
สยามชำนาญกิจ
SIAM CHAMNAN KIT

Kubernetes

In Practice





Agenda Day 1

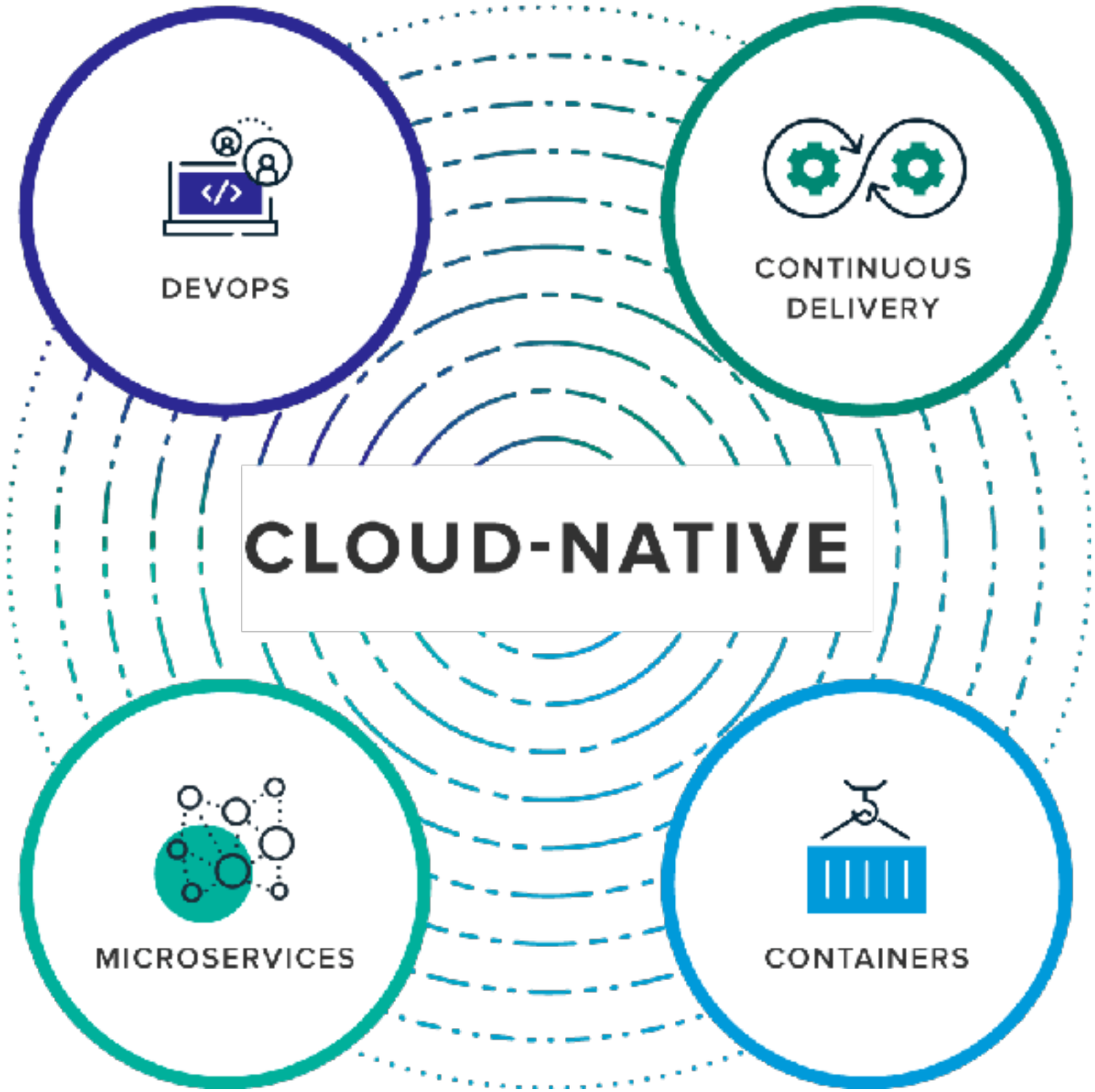
1. Cloud Native Application
2. Kubernetes architecture
3. Key-features
4. Pods and Containers
5. Service
6. Replication Controller (RC)
7. Deployment and ReplicaSet (RS)
8. Volume

Agenda Day 2

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

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

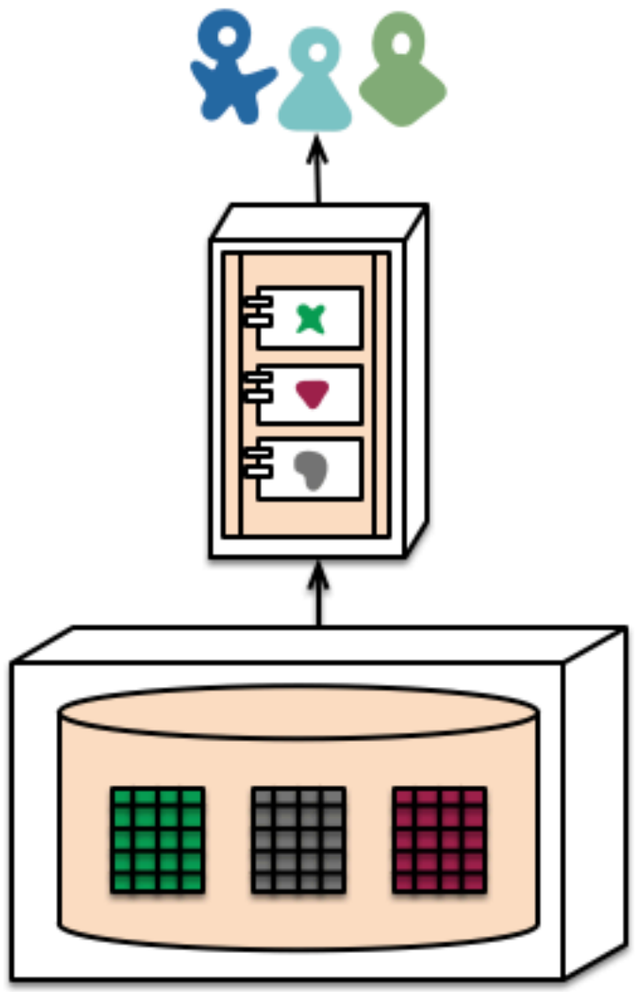
Cloud Native Application



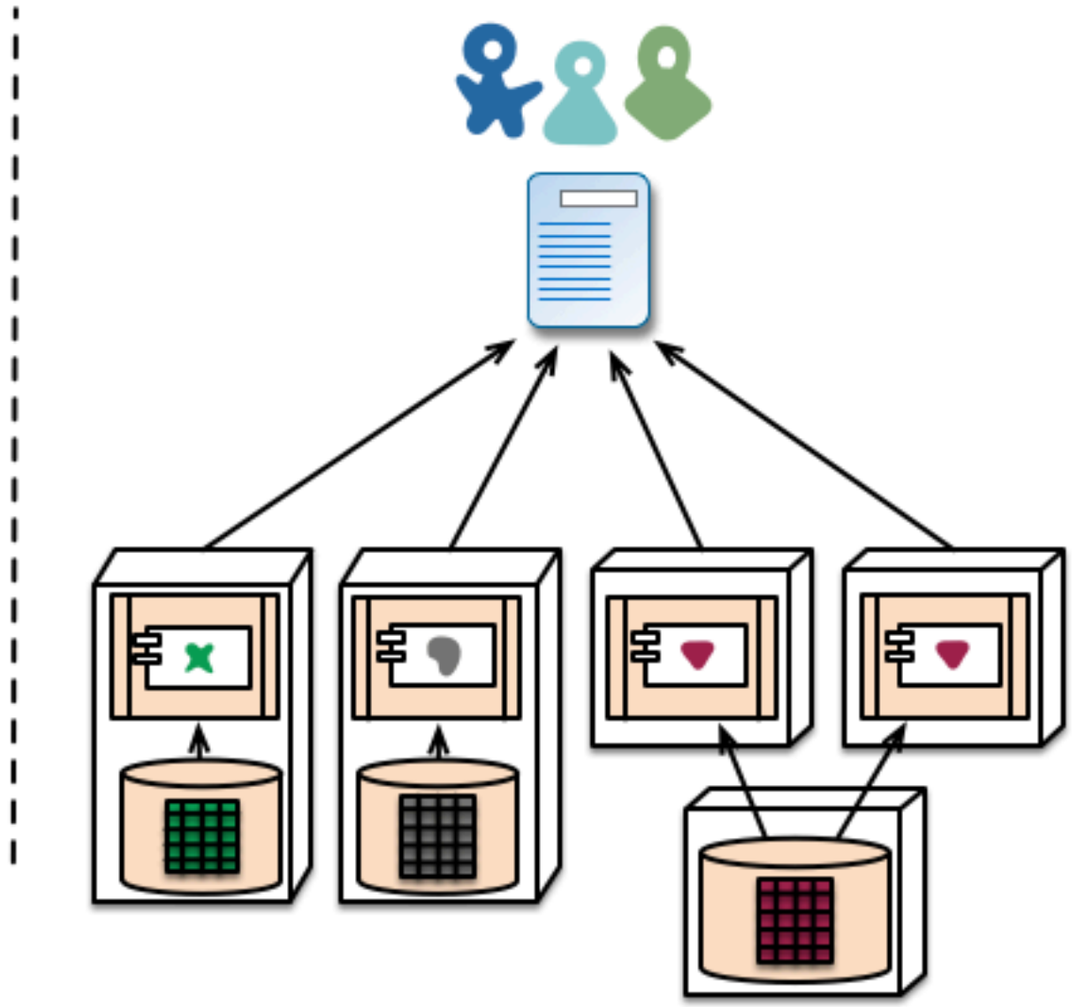
https://pivotal.io/cloud-native

Evolution of Architecture

Microservies

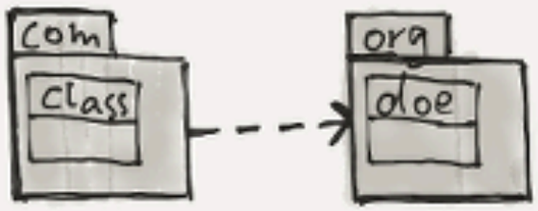


monolith - single database

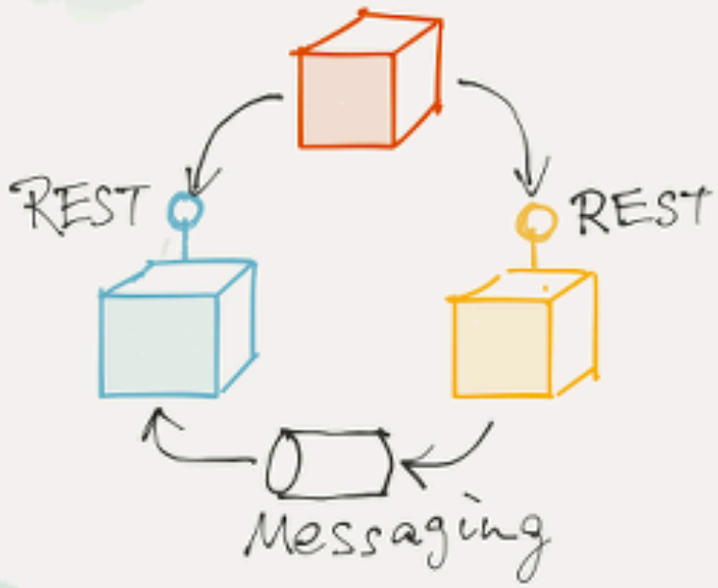


microservices - application databases

Architecture



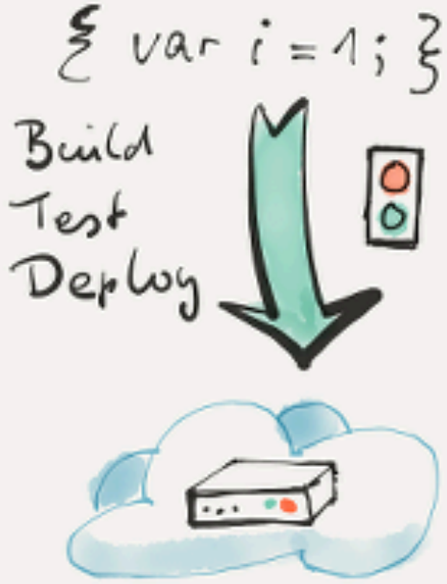
Microservices



People & Teams

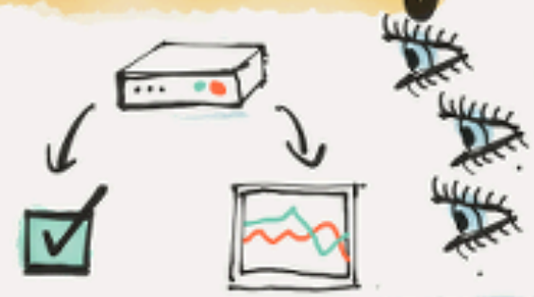
Deployment

Continous Delivery



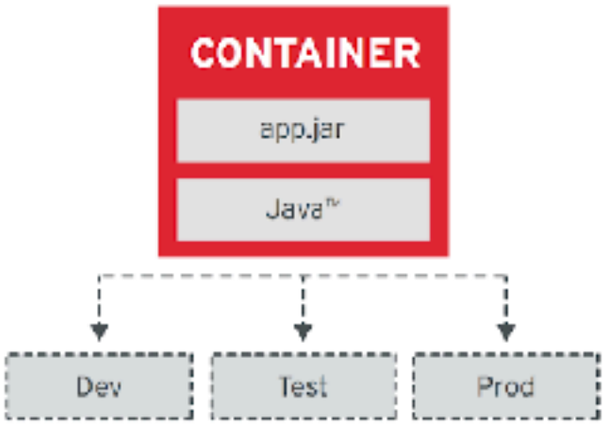
Infrastructure

Monitoring



Container design principles

Image Immutability Principle

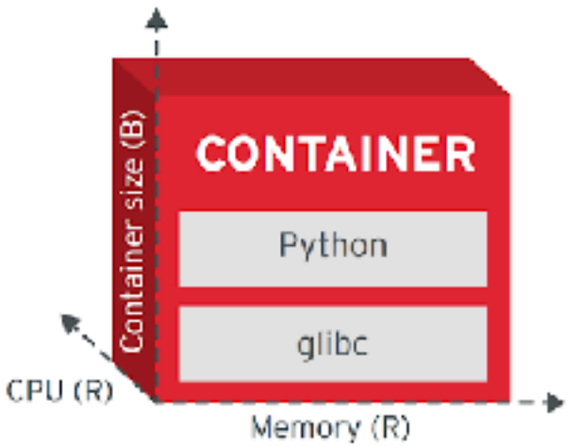


High Observability Principle

Process Disposability Principle

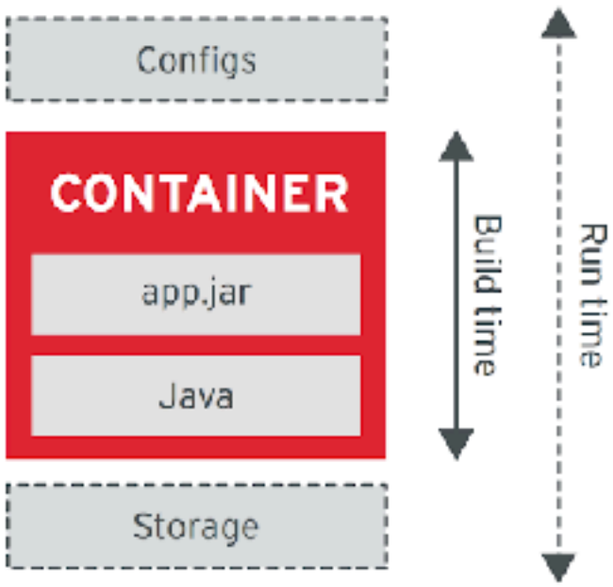
Lifecycle Conformance Principle

Runtime Confinement Principle

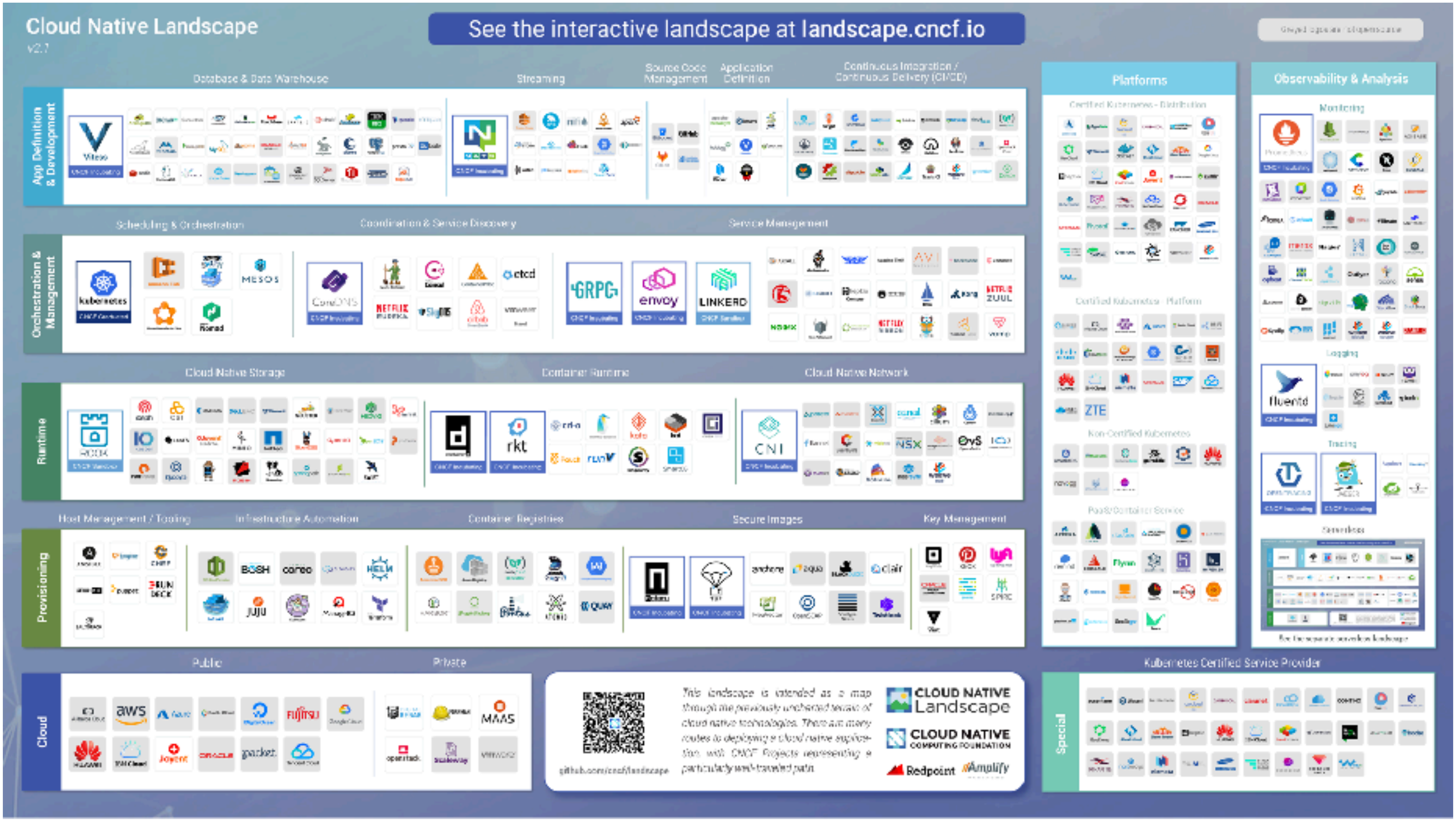


Single Concern Principle

Self-Containment Principle

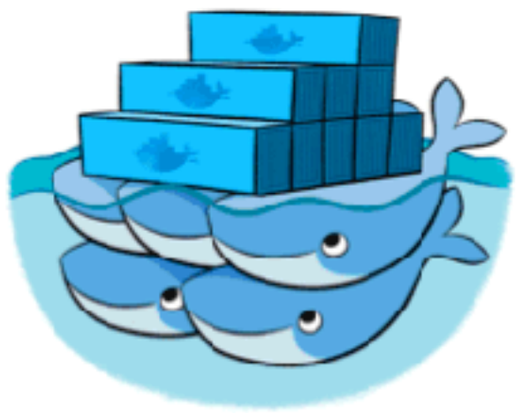


Cloud native landscape



https://github.com/cncf/landscape

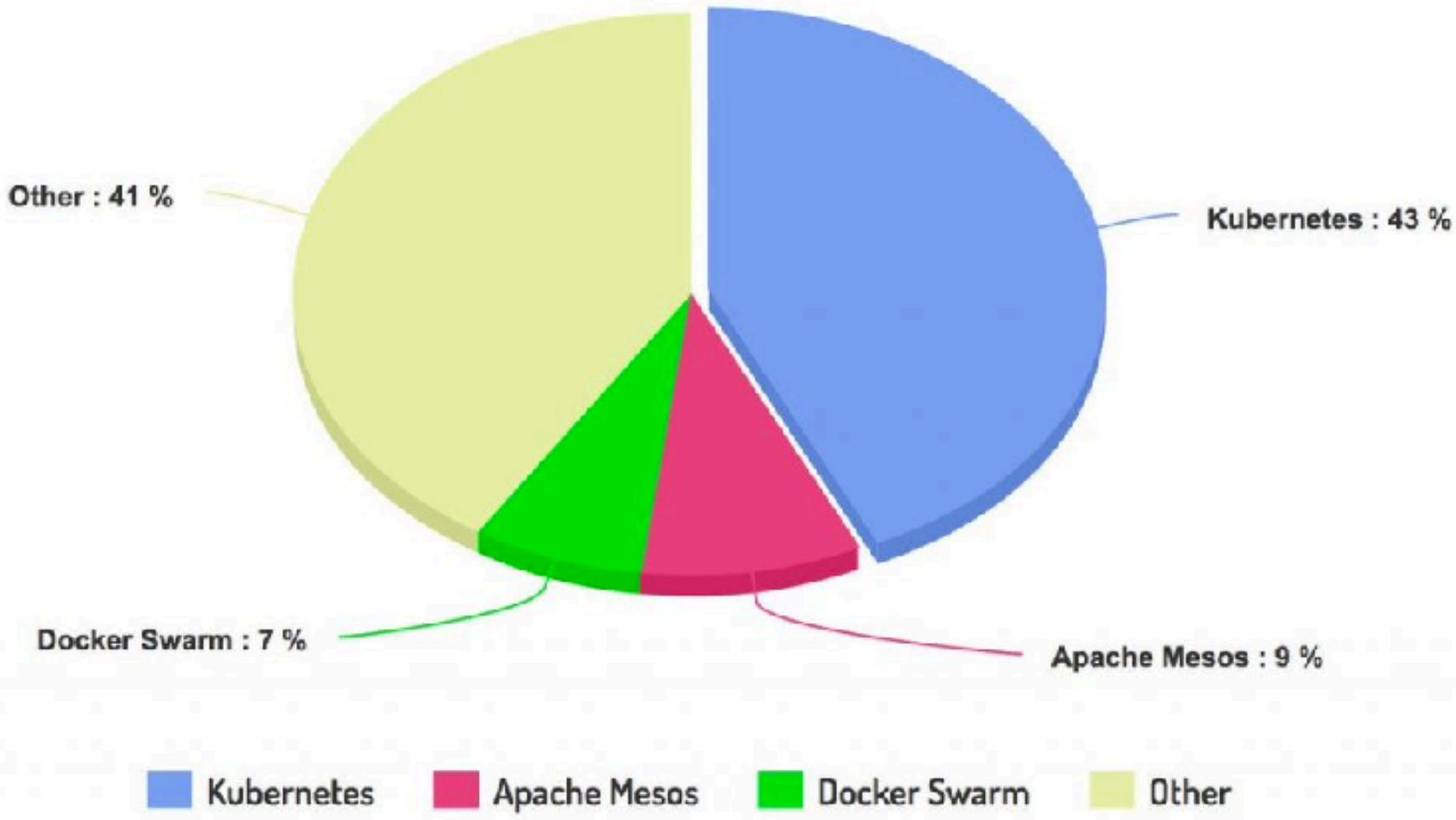
MESOS



Marathon

kubernetes
by Google

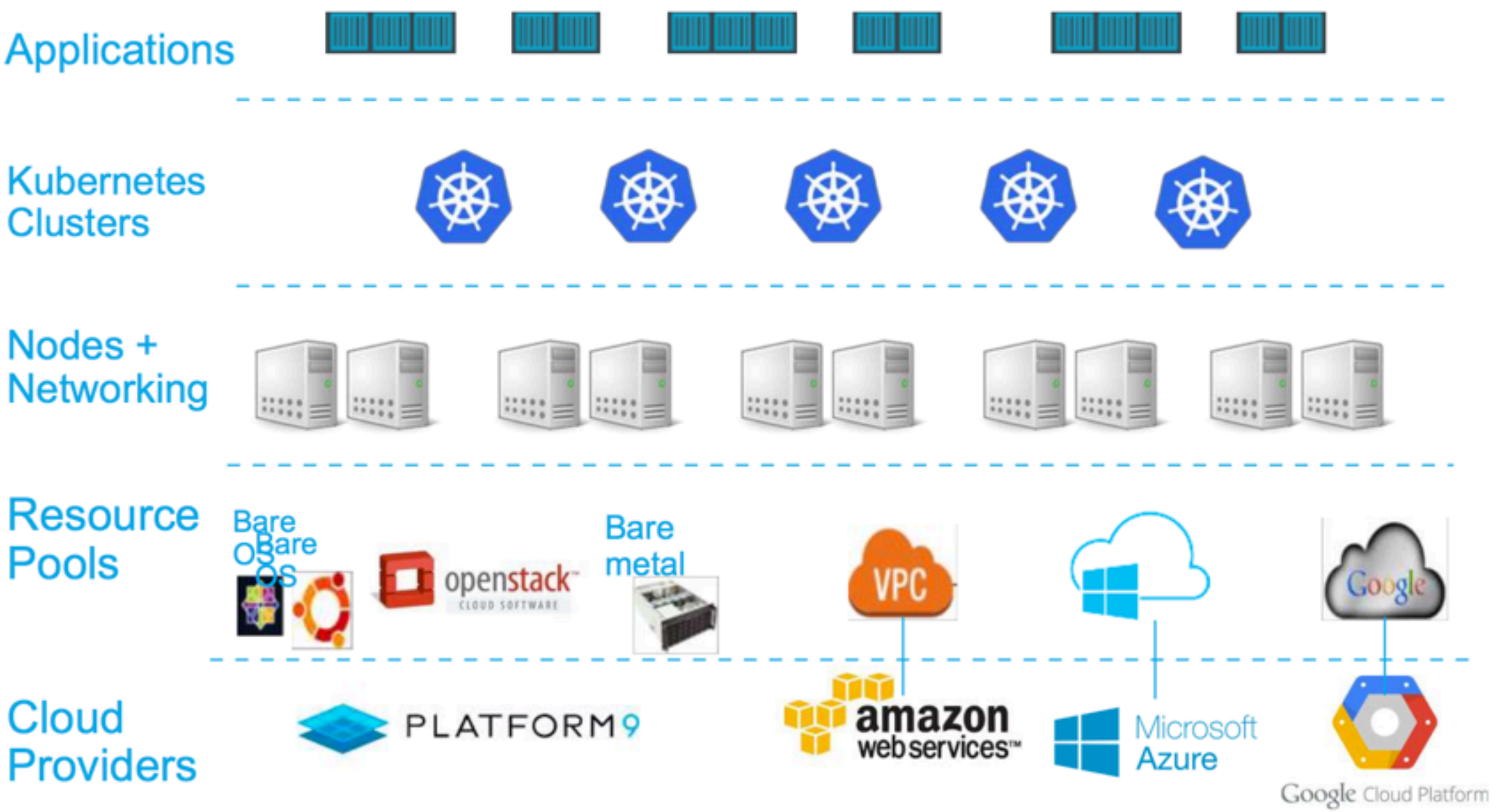
Container Orchestrators in Sysdig's 2017 Docker Usage Report



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

Why Kubernetes ?

Write once, run anywhere



Write once, run anywhere

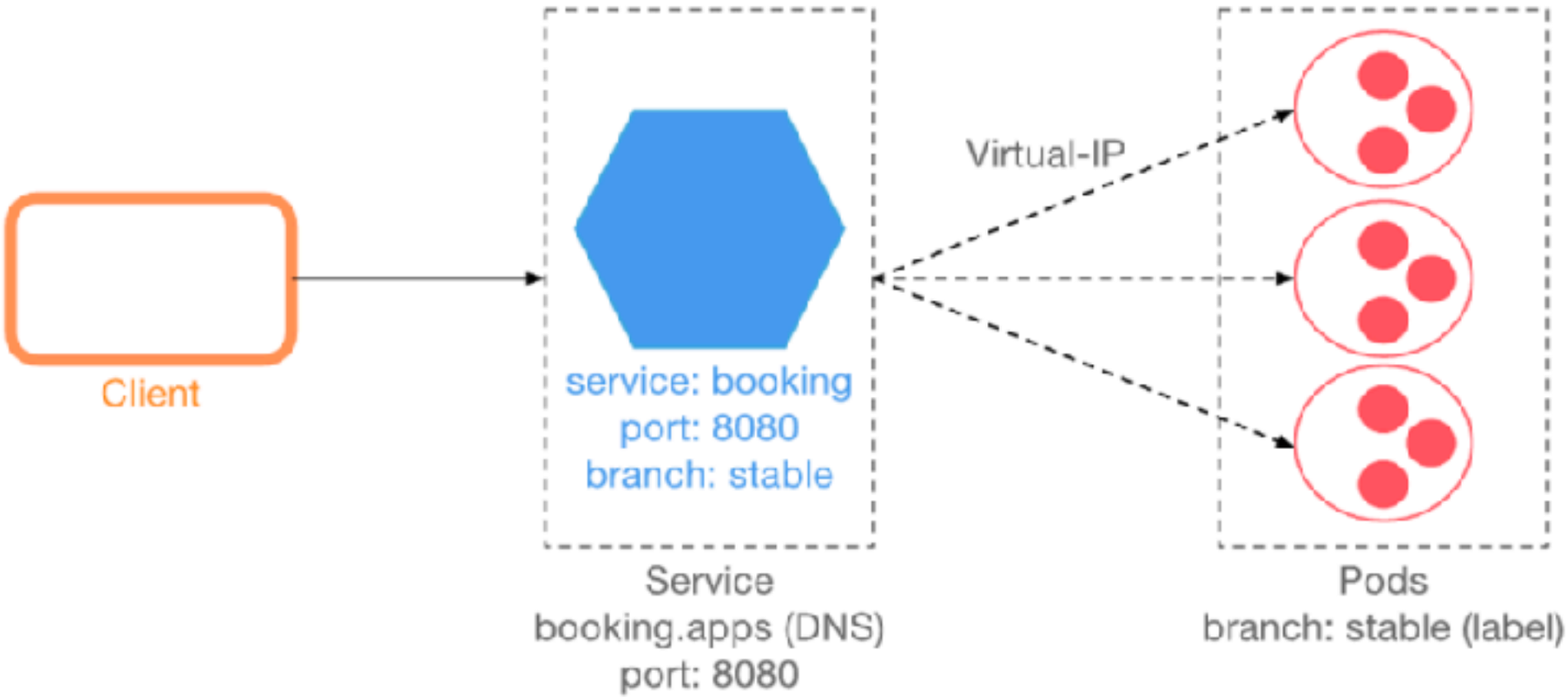
Eliminate infrastructure lock-in
Use containers
Provides management for containers

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

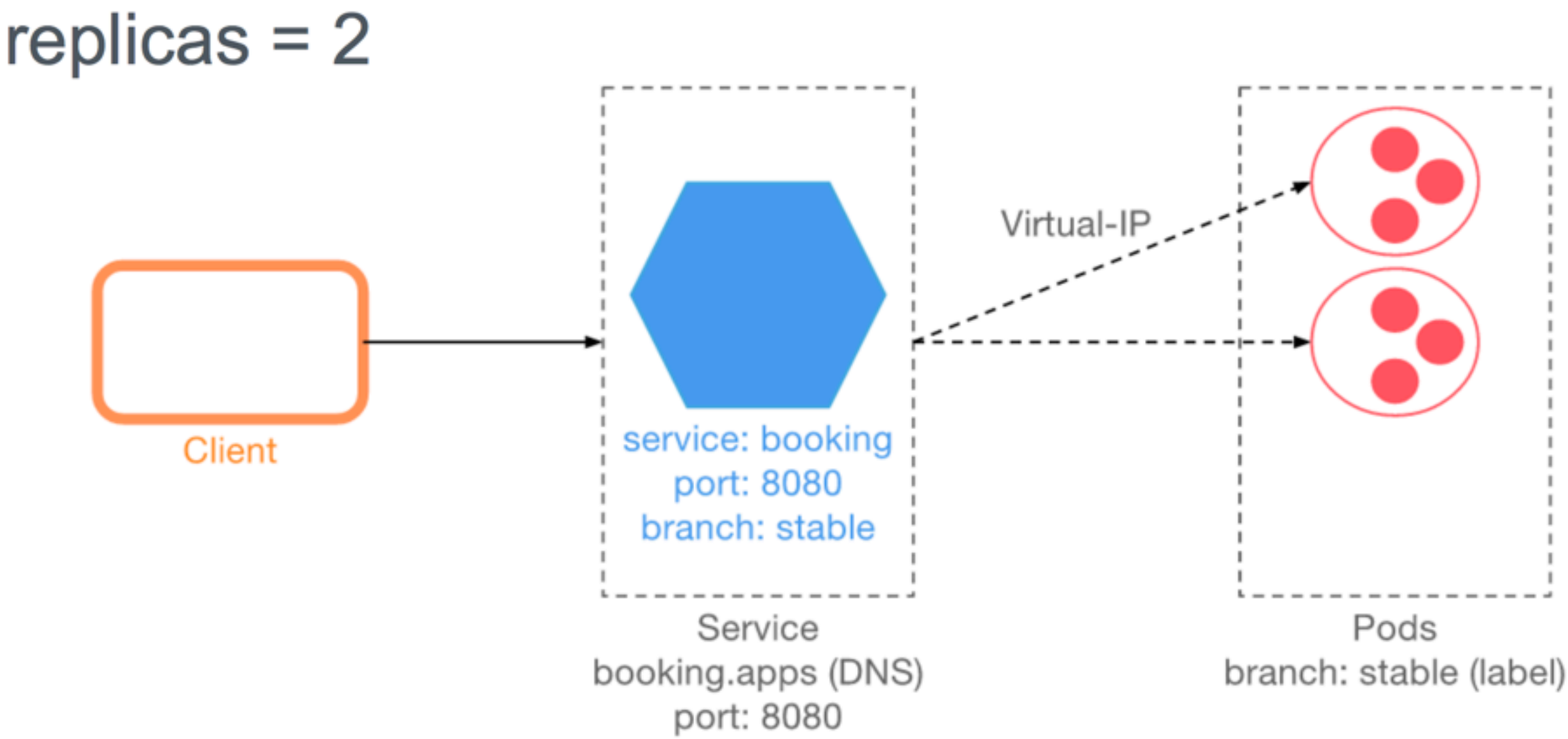
Container/Image boundary similar to class
How to use/manage a collection of container ?



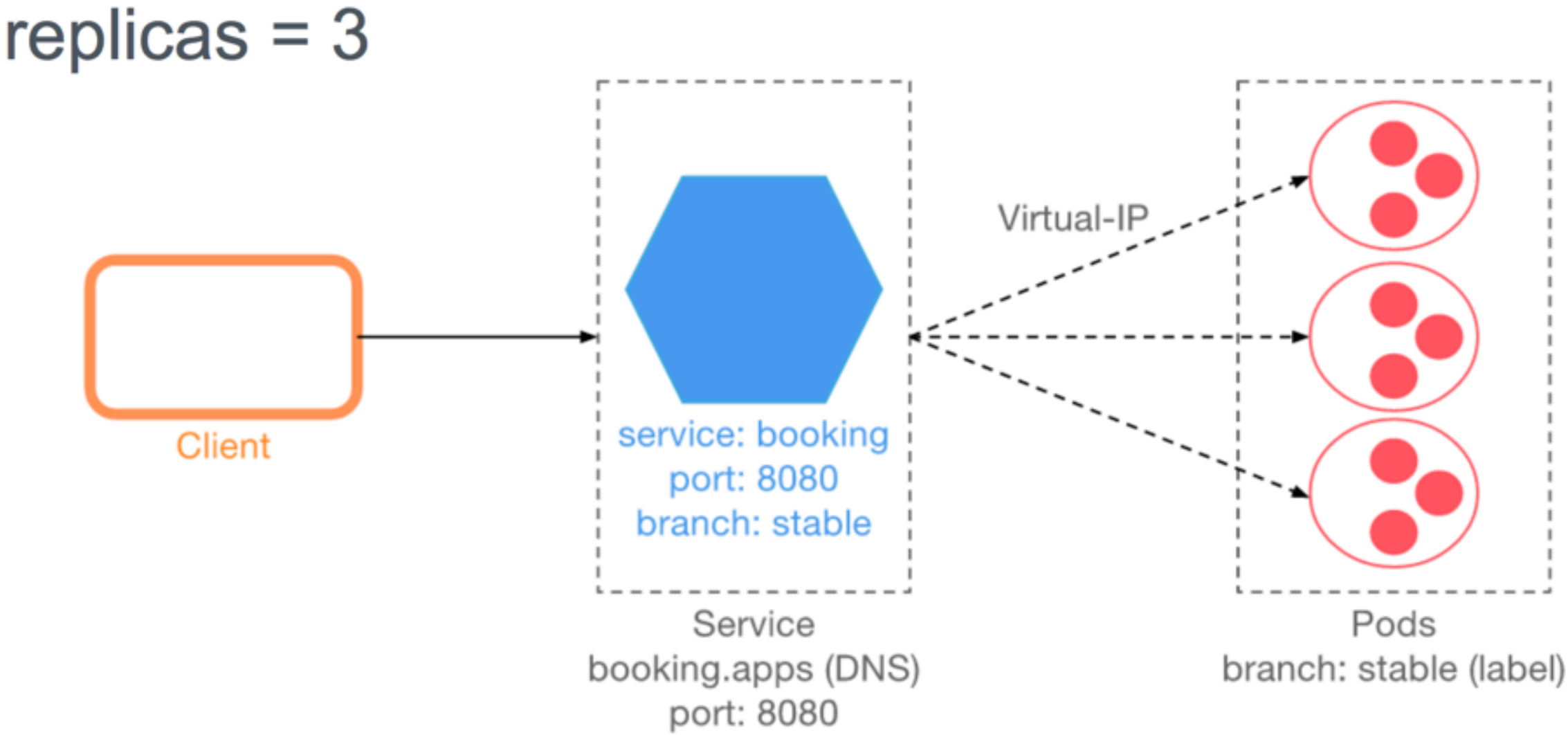
Fault-tolerant by design

Design for failure
Infrastructure provisioning/re-provisioning
Configuration networking and load balance
Redundancy (scale-out)
Lifecycle management (Software update)

Fault-tolerant by design



Fault-tolerant by design

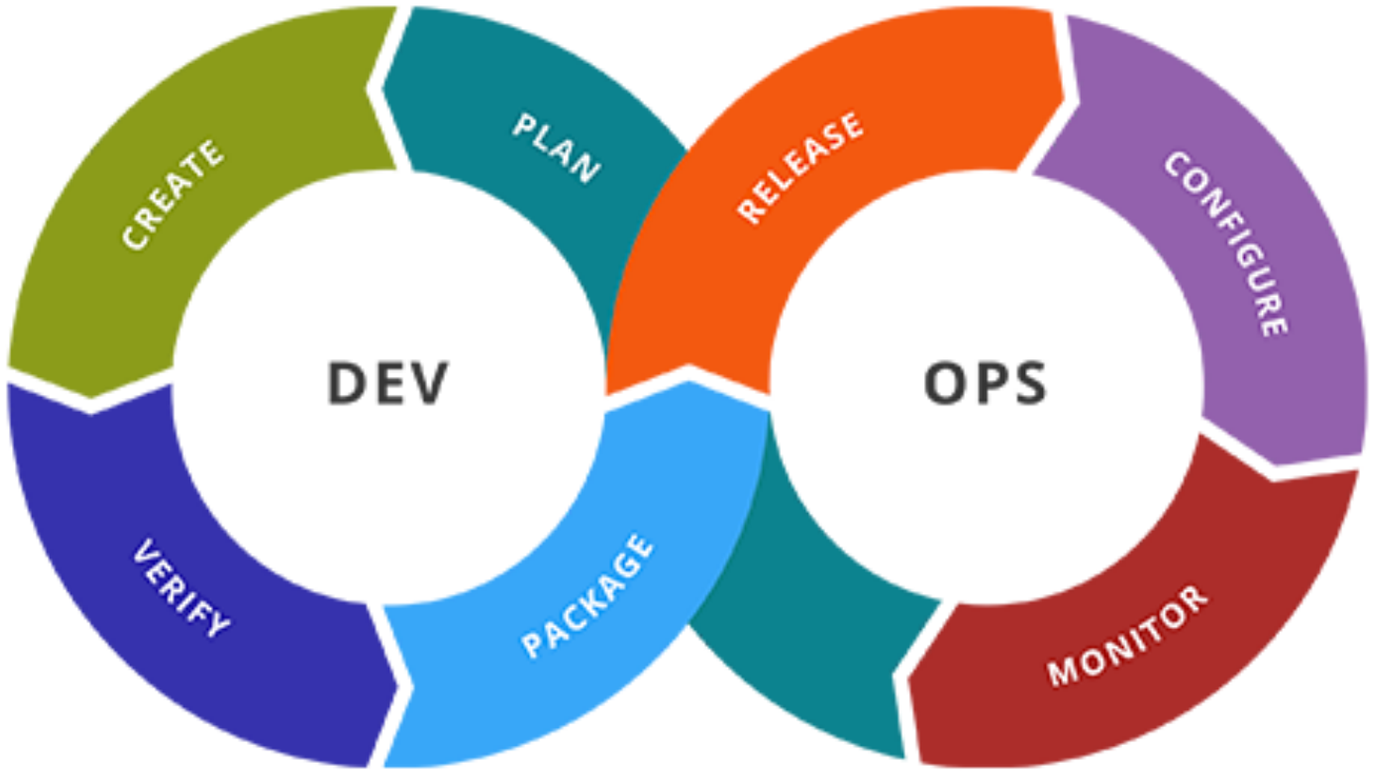


Deployment, not Infrastructure

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

https://github.com/kubernetes/minikube

Starting minikube

$minkube start
$minikube status

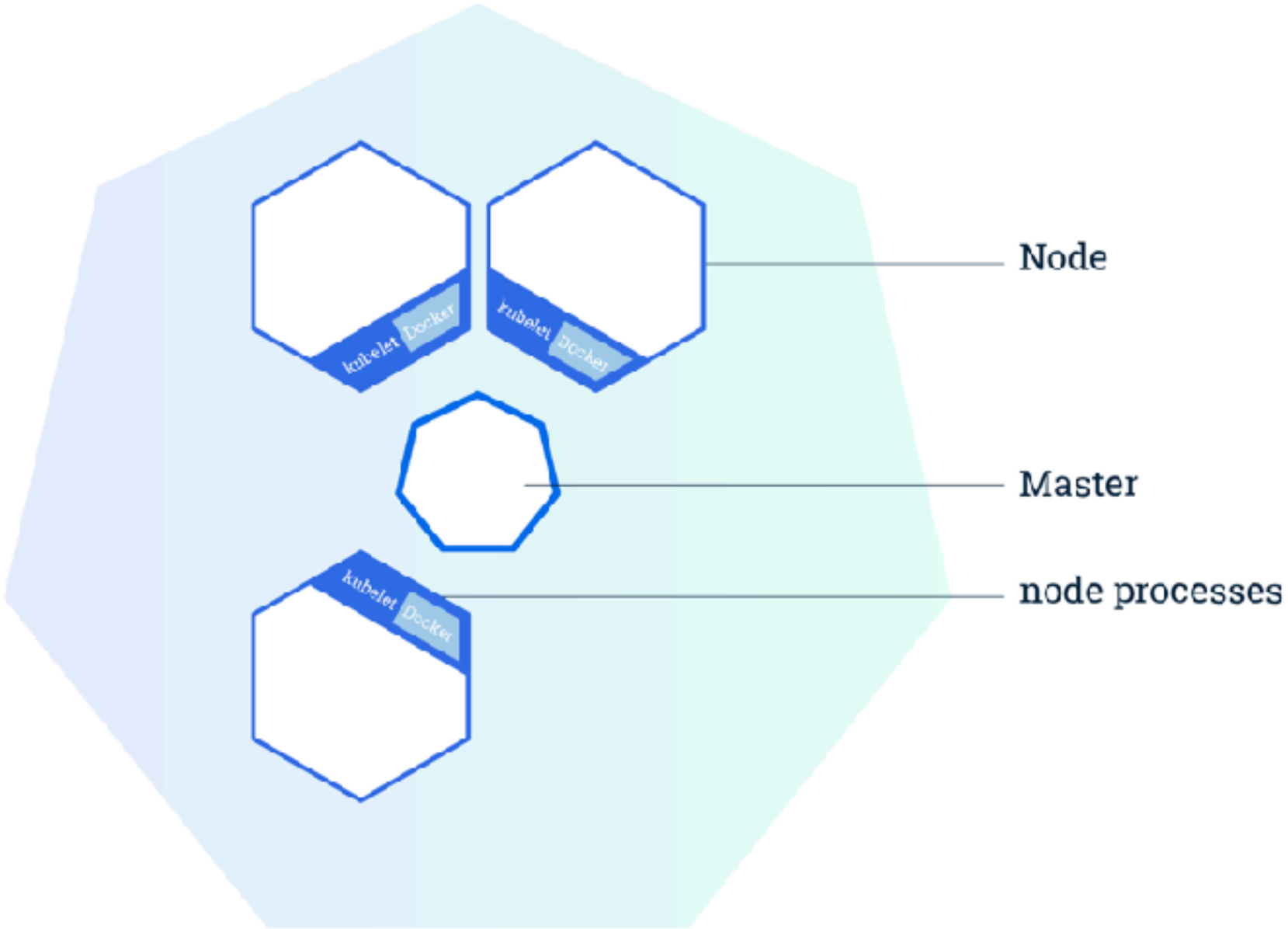
Version of Kubernetes

$kubectl get nodes -o yaml

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

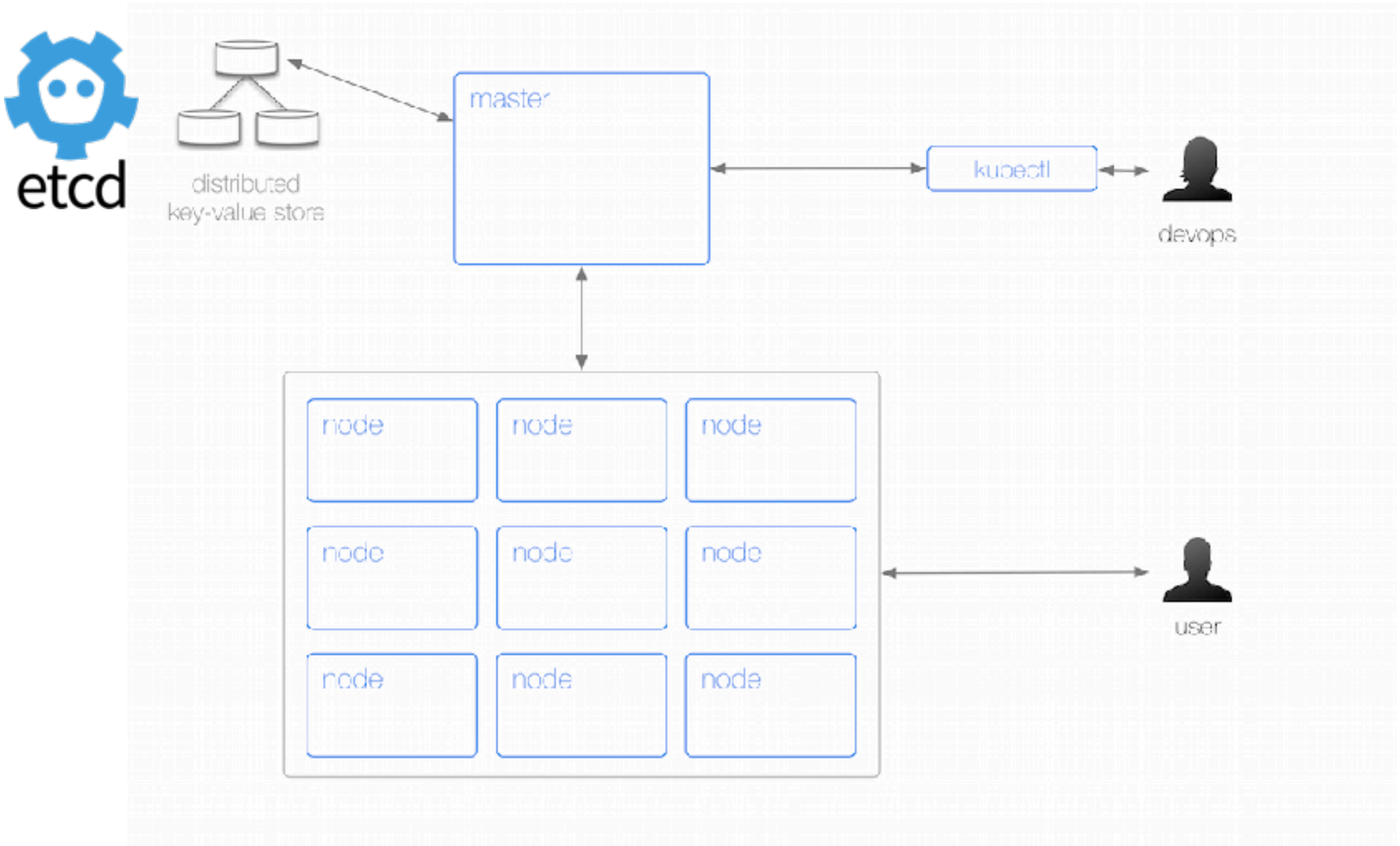
Basic of Kubernetes

Kubernetes cluster



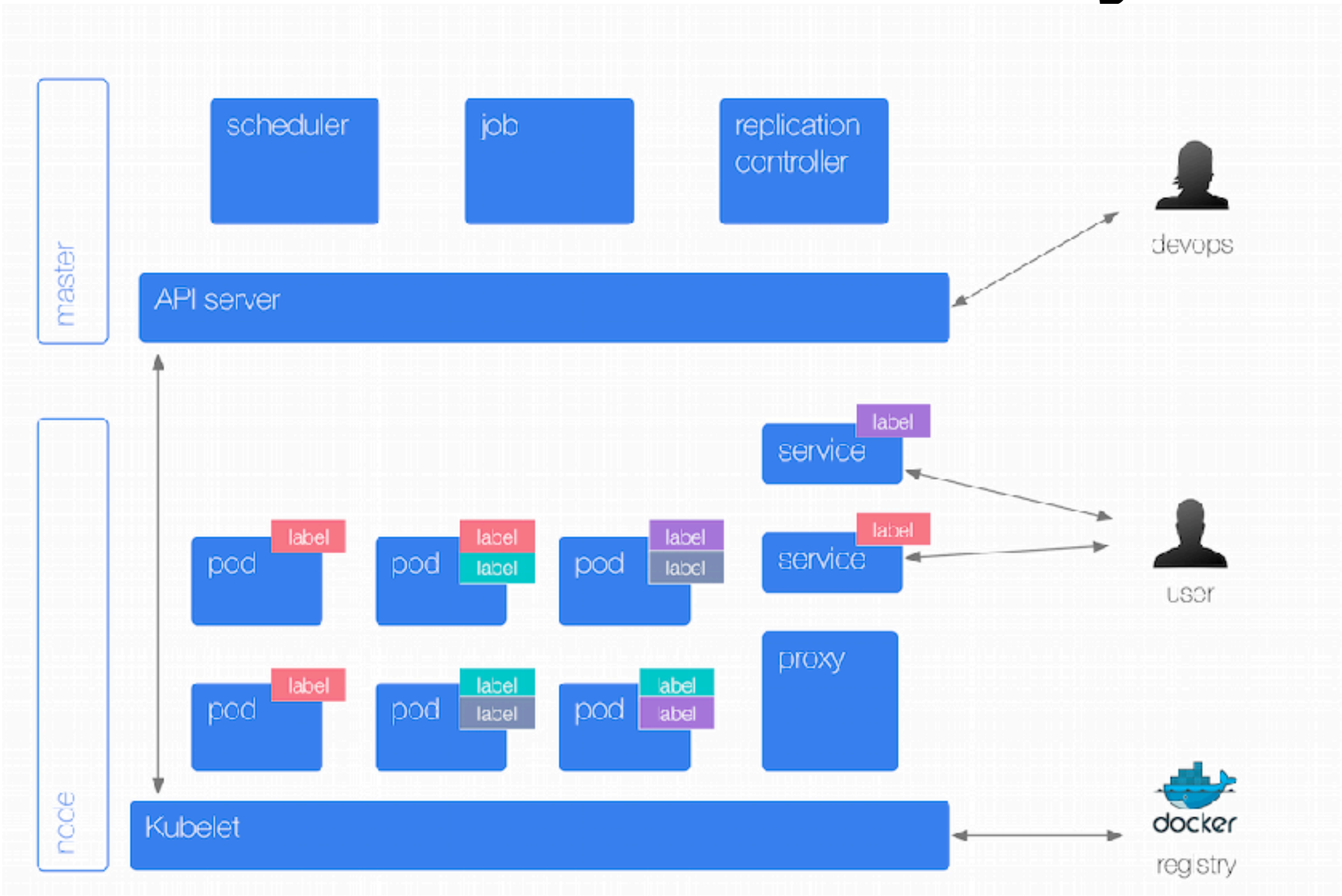
Kubernetes cluster

Kubernetes physical layout



http://k8s.info/cs.html

Kubernetes abstraction layout



http://k8s.info/cs.html

Hello with Kubernetes

file /01-hello/instruction.txt

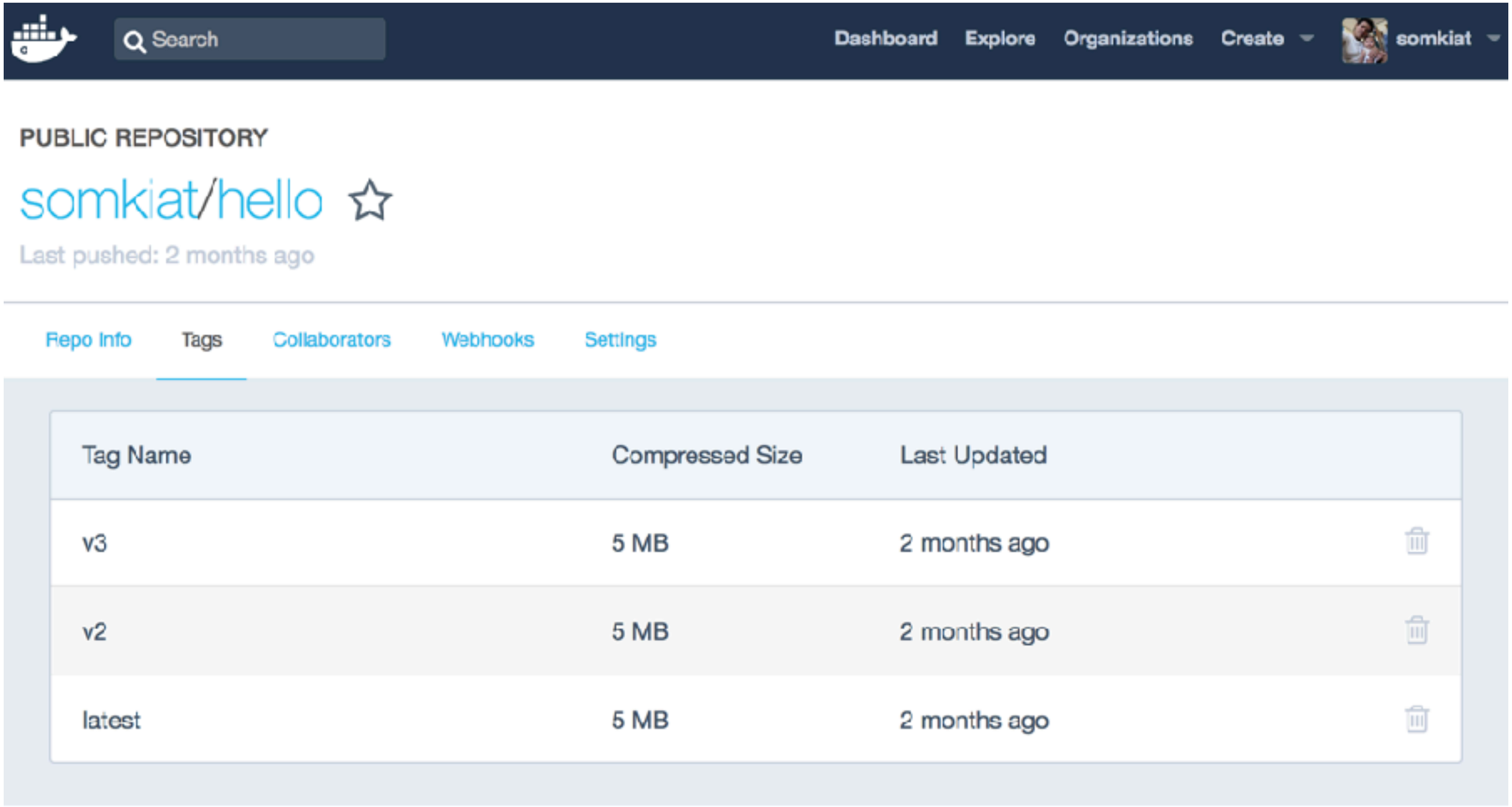
Hello with Kubernetes

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



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

Create a container

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

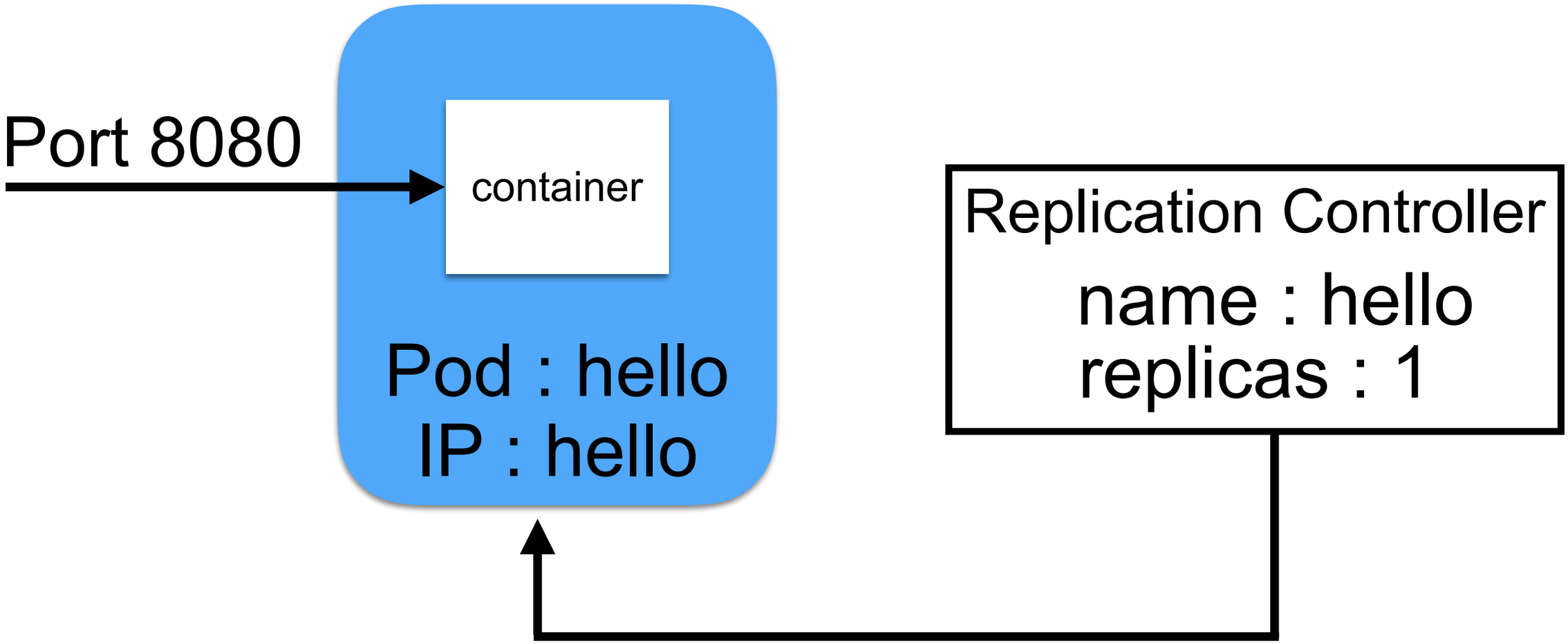
Create a container

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

Create Replication Controller (RC)

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

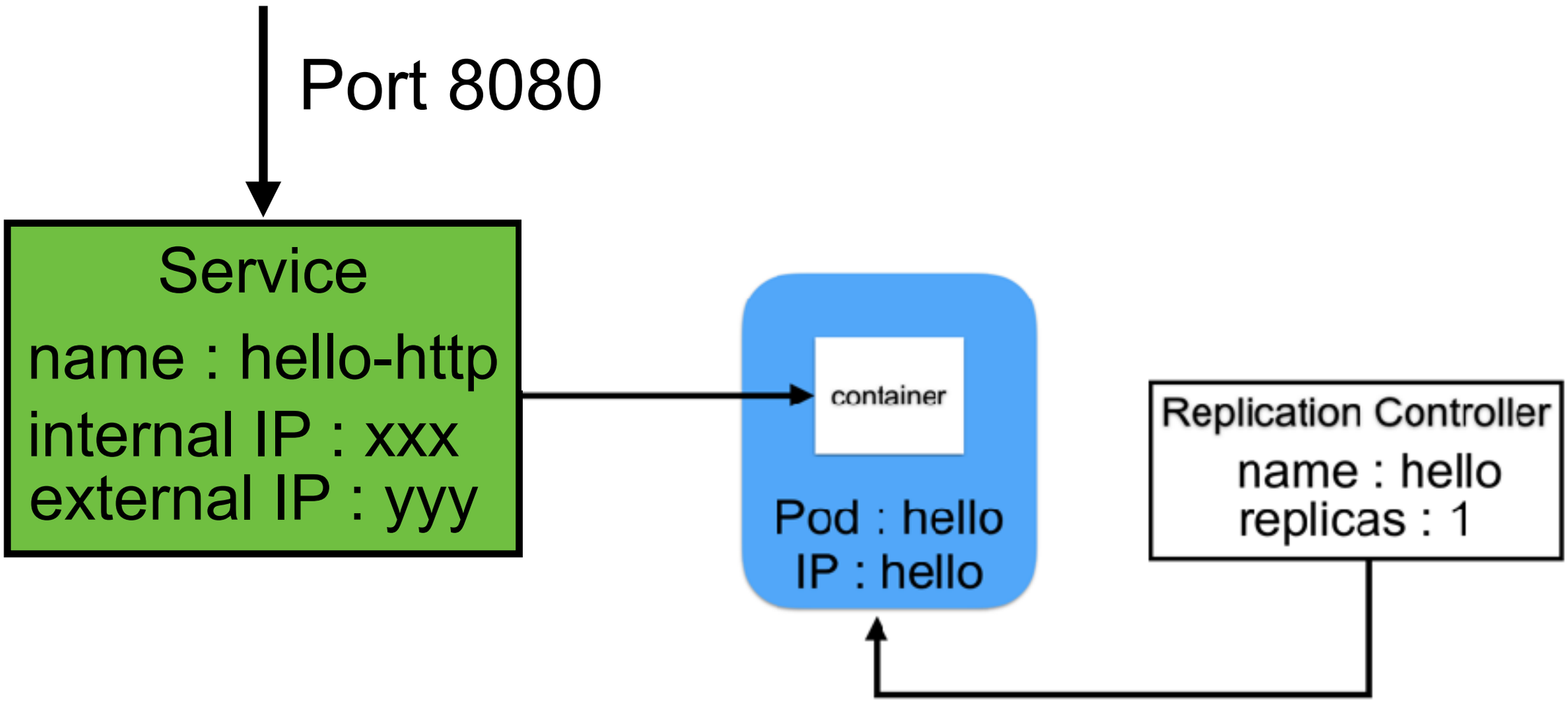
Create a container



Expose RC with service

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

Expose RC with service



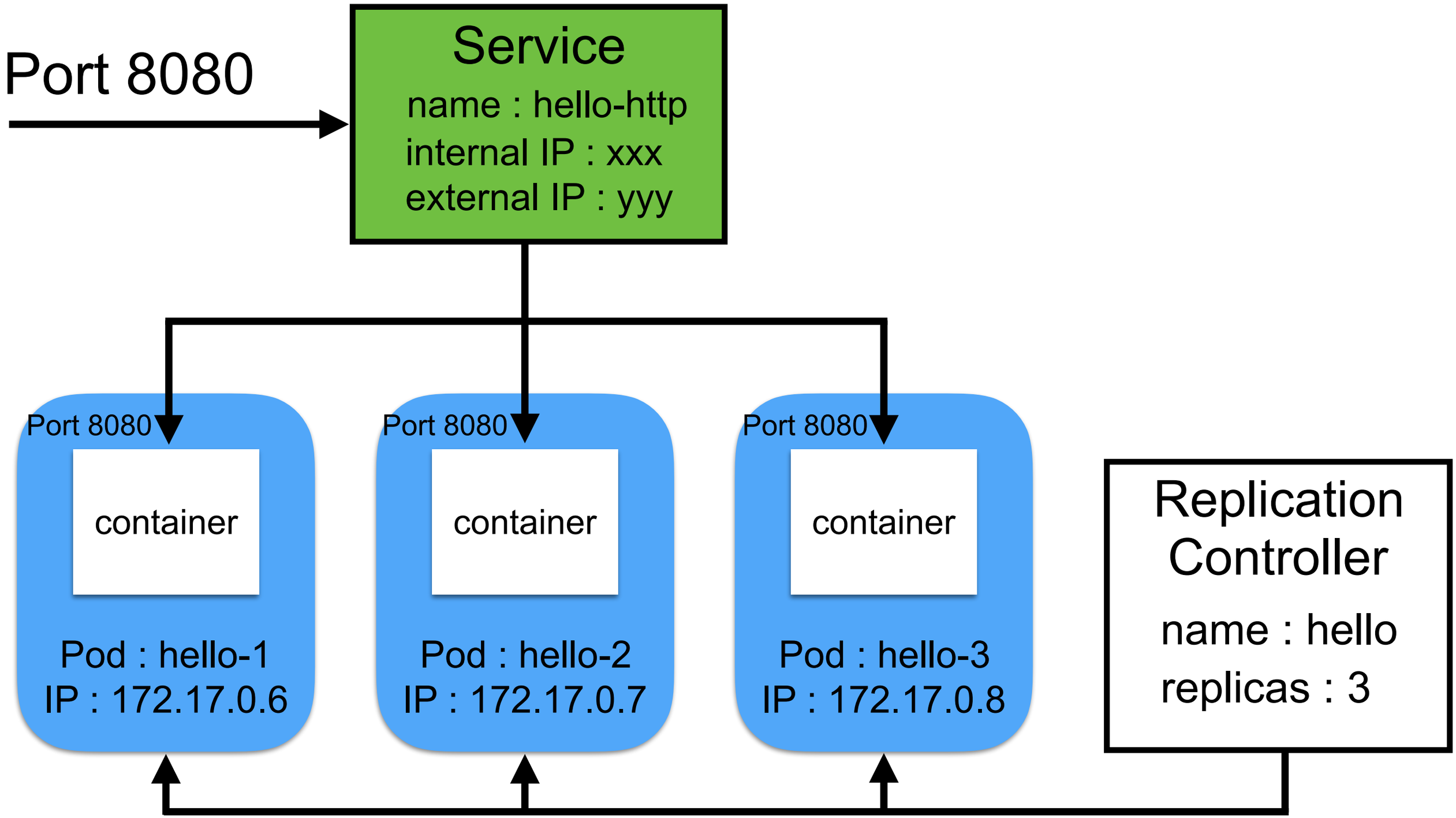
Access service with minikube

$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-pqnzp   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

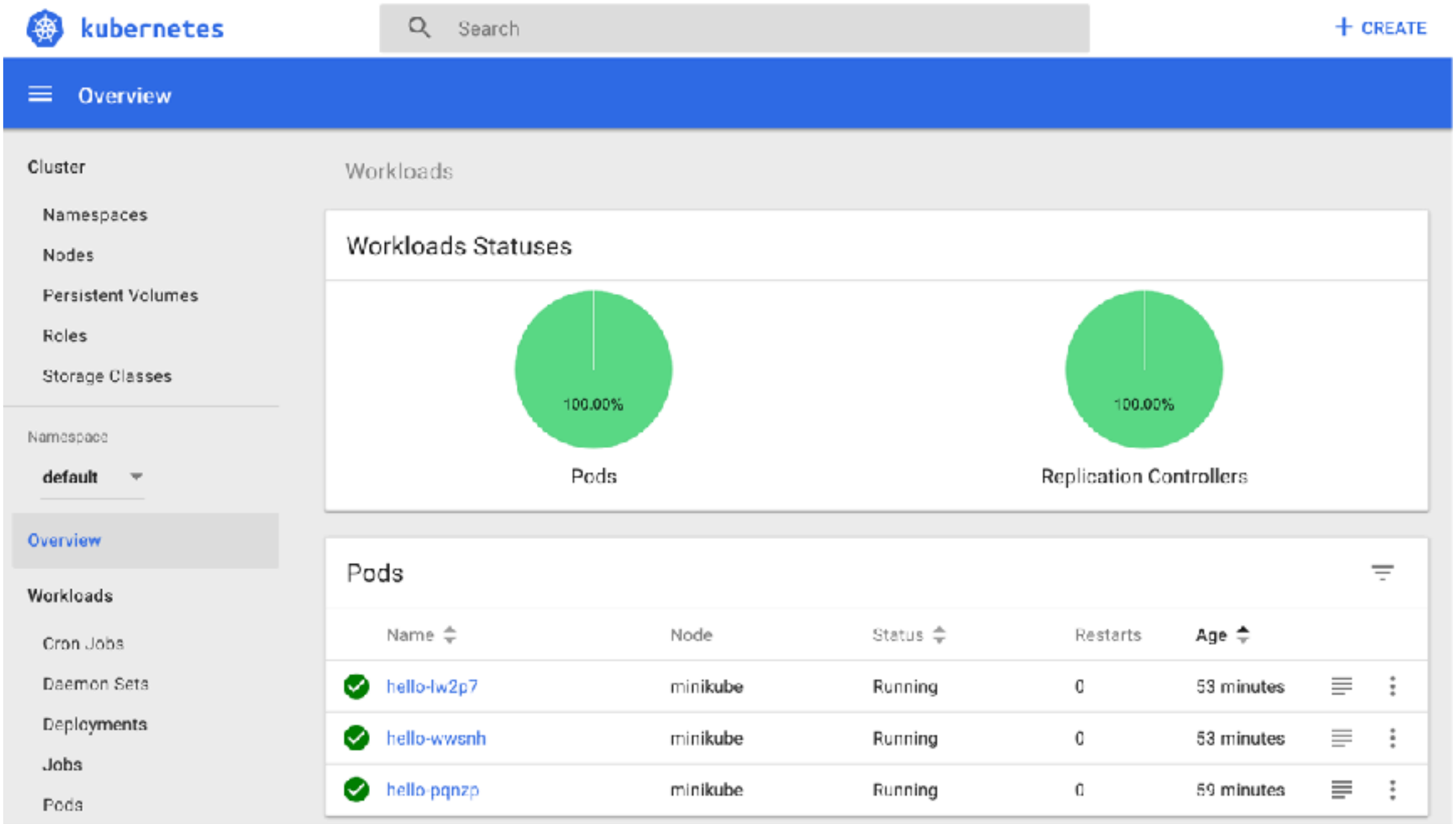
$curl http://$SERVICE
Hello, "/" on **hello-wwsnh**

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

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

Kubernetes dashboard

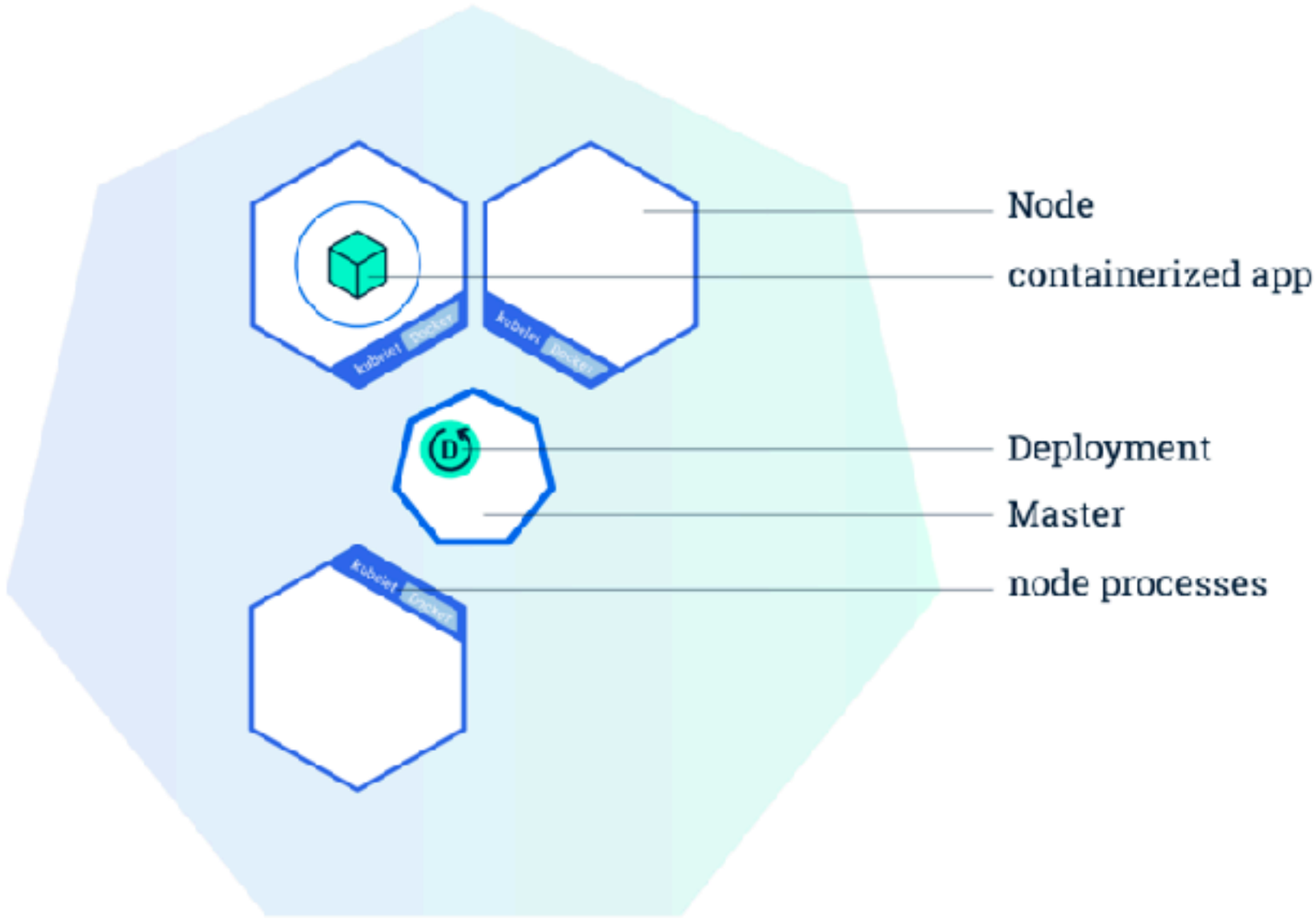
$minikube dashboard



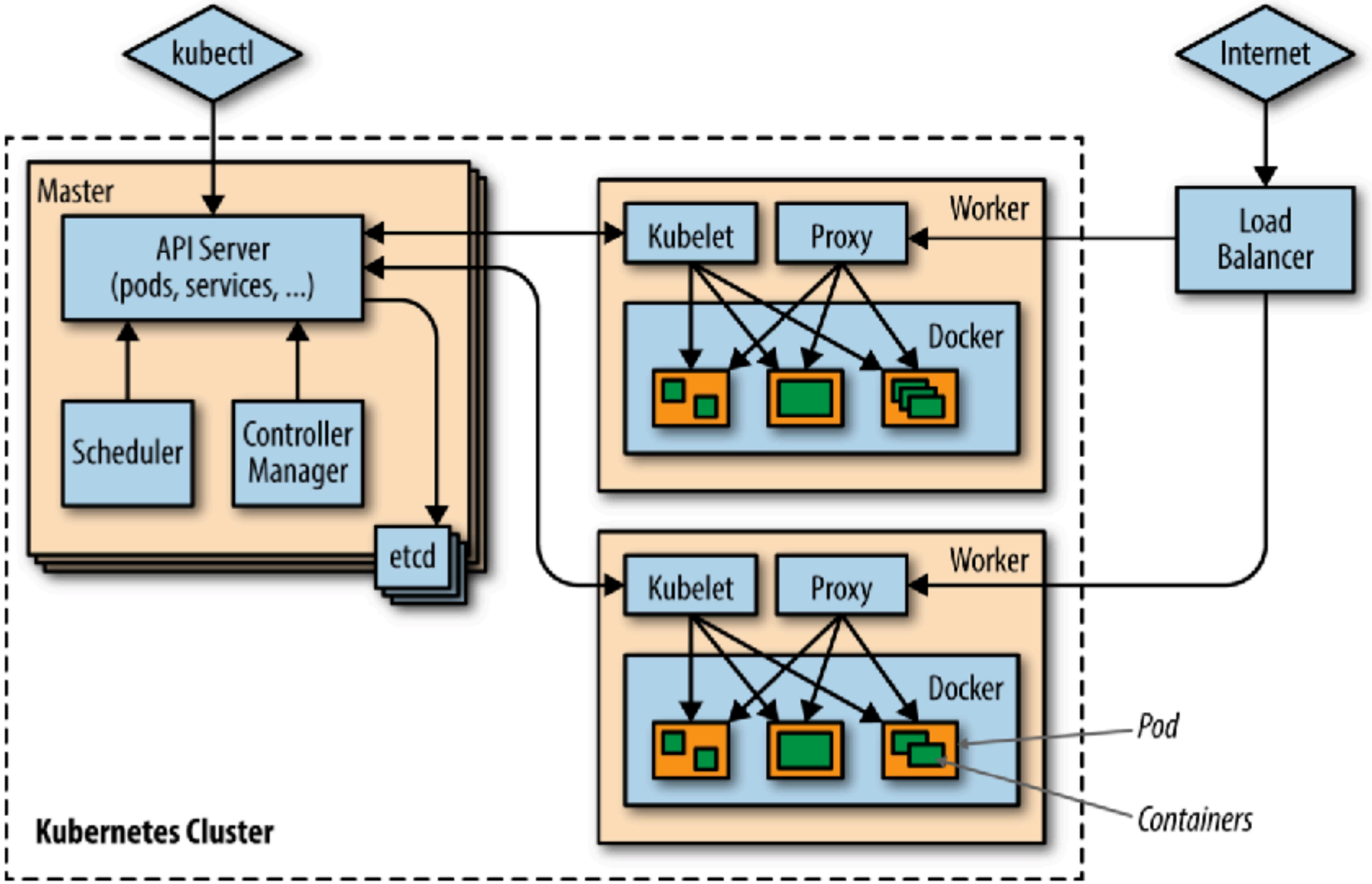
Welcome to Kubernetes

Kubernetes Architecture

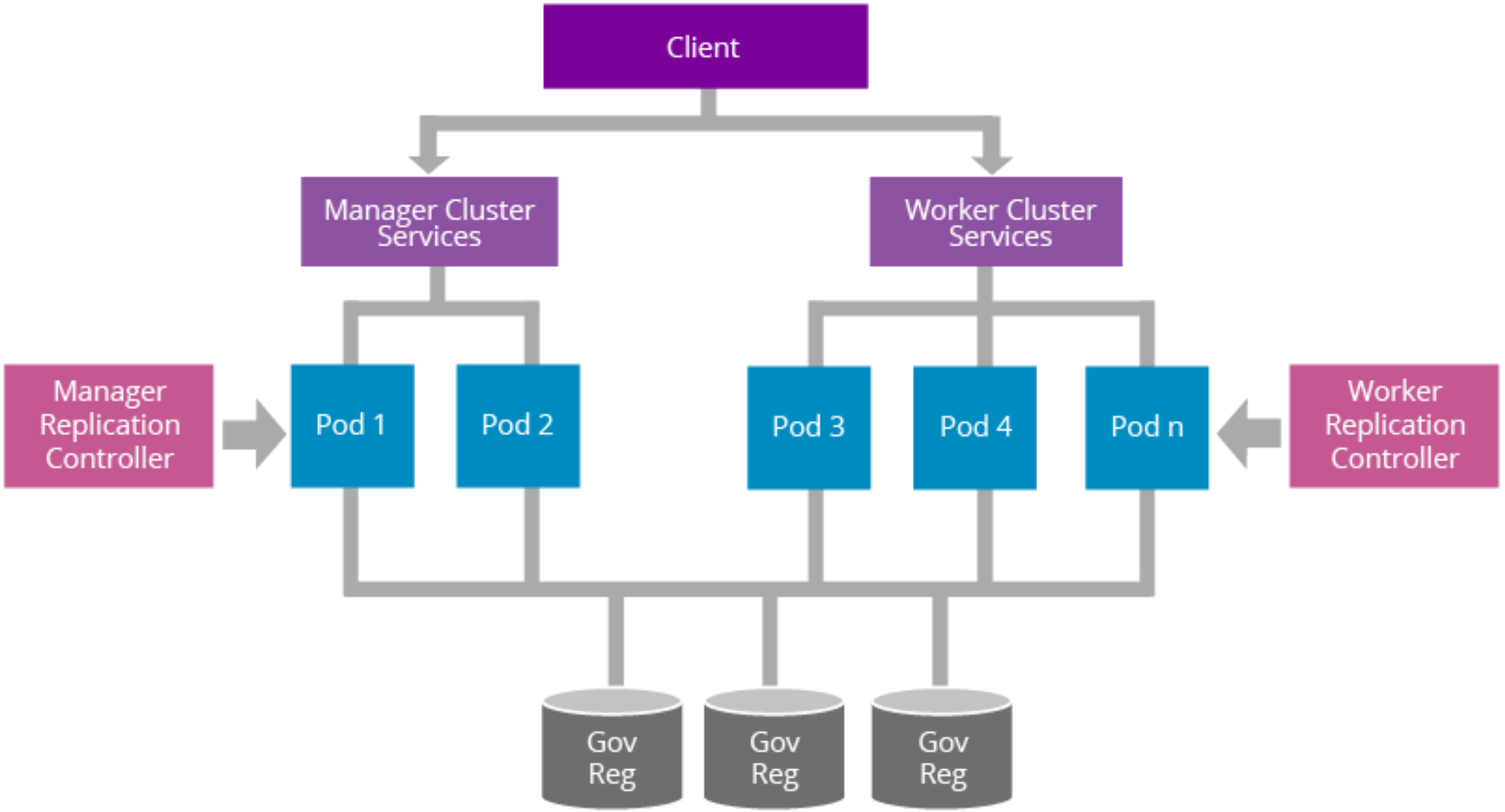
Kubernetes cluster



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

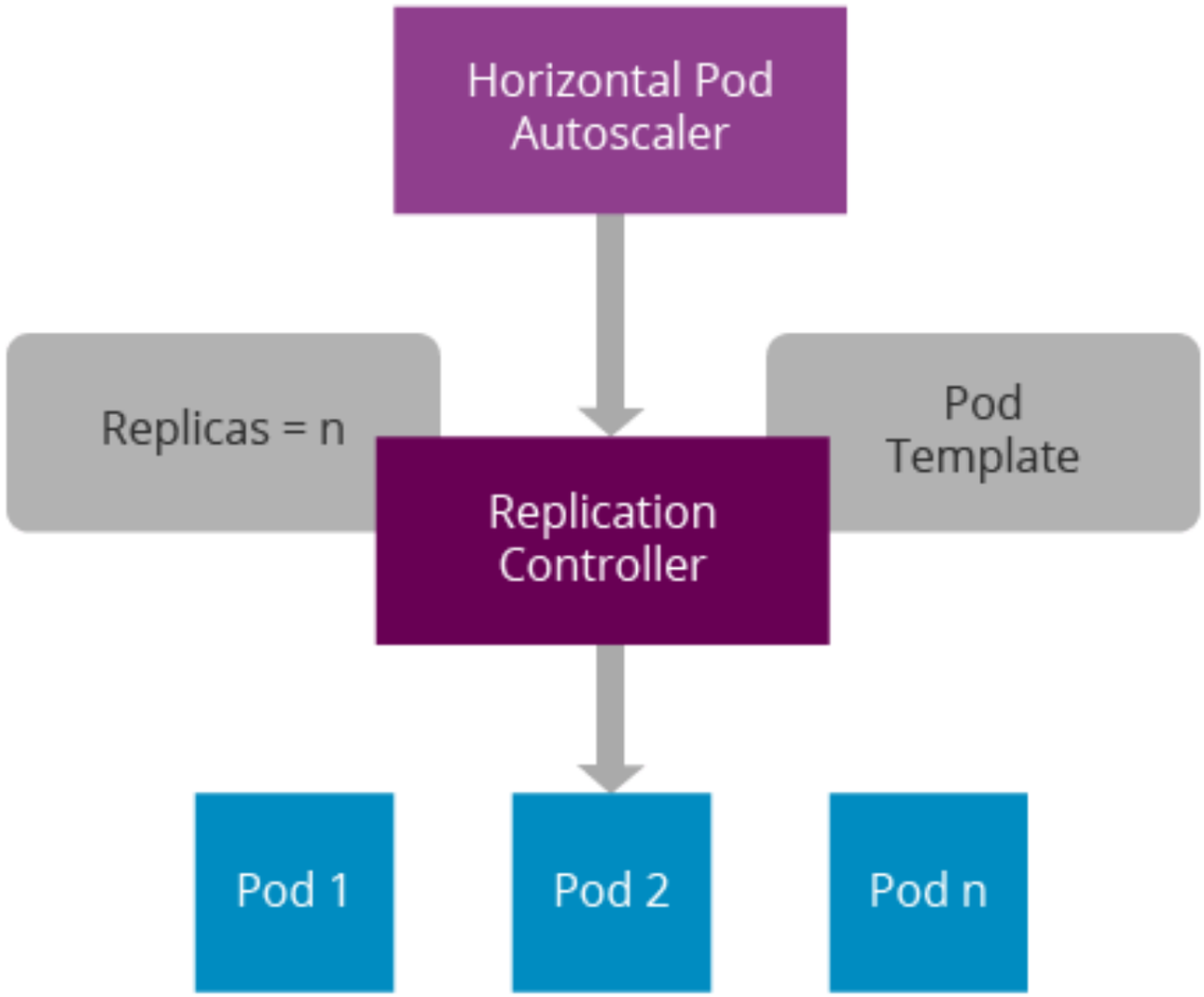
Try to check resources of node

$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



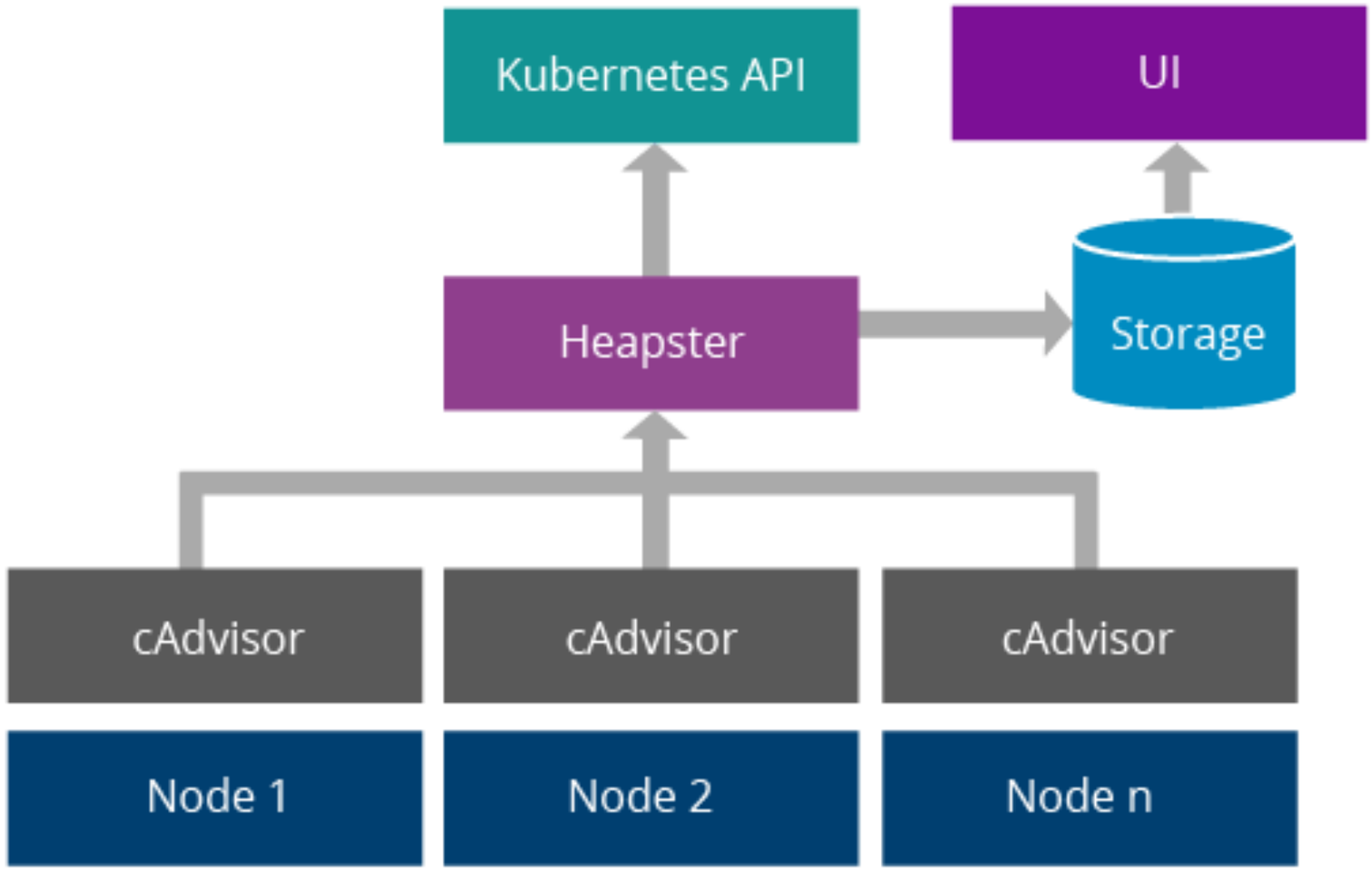
Horizontal Pod Autoscaler

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

Scaling-up and down for your app need !!

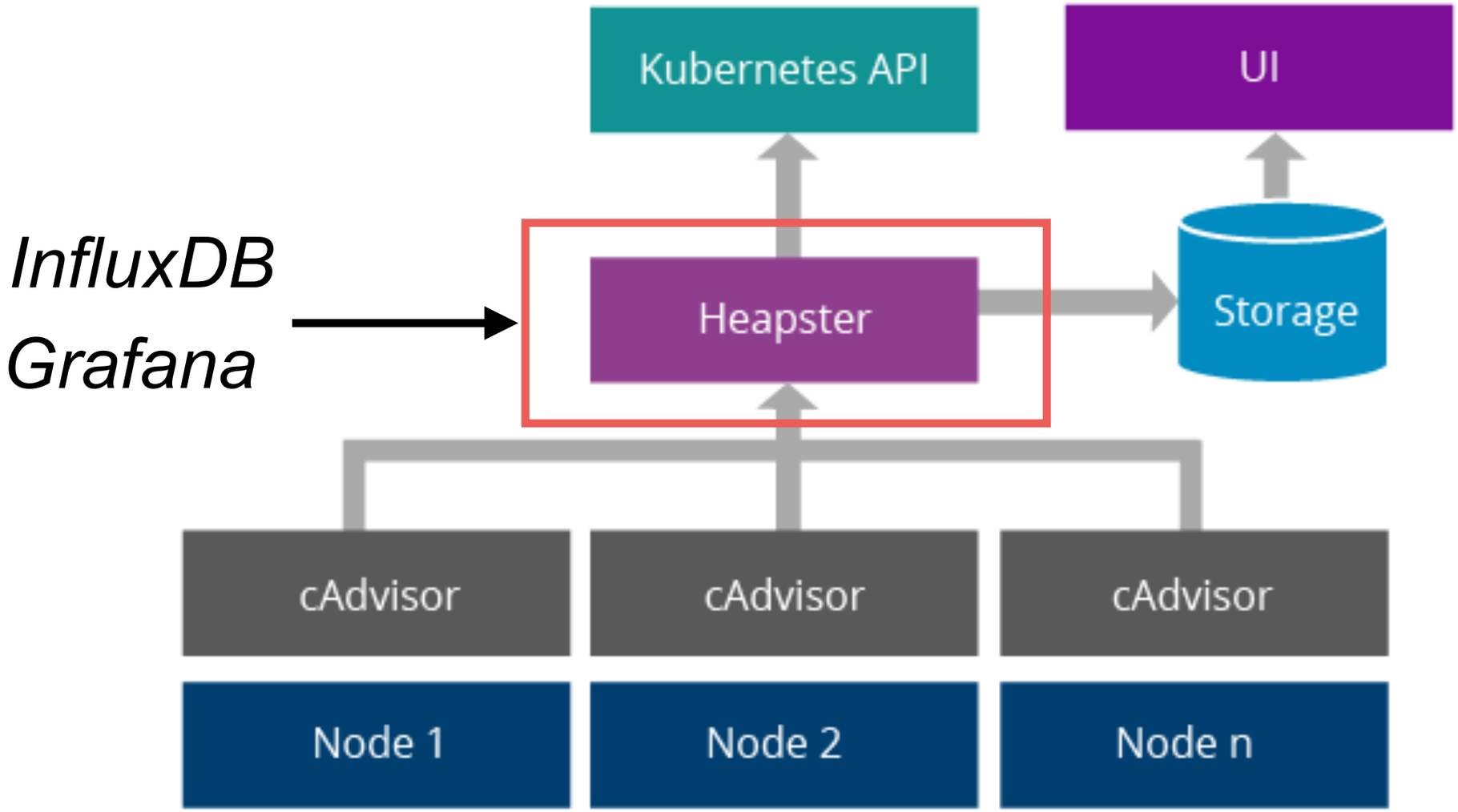
Resource usage monitoring

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

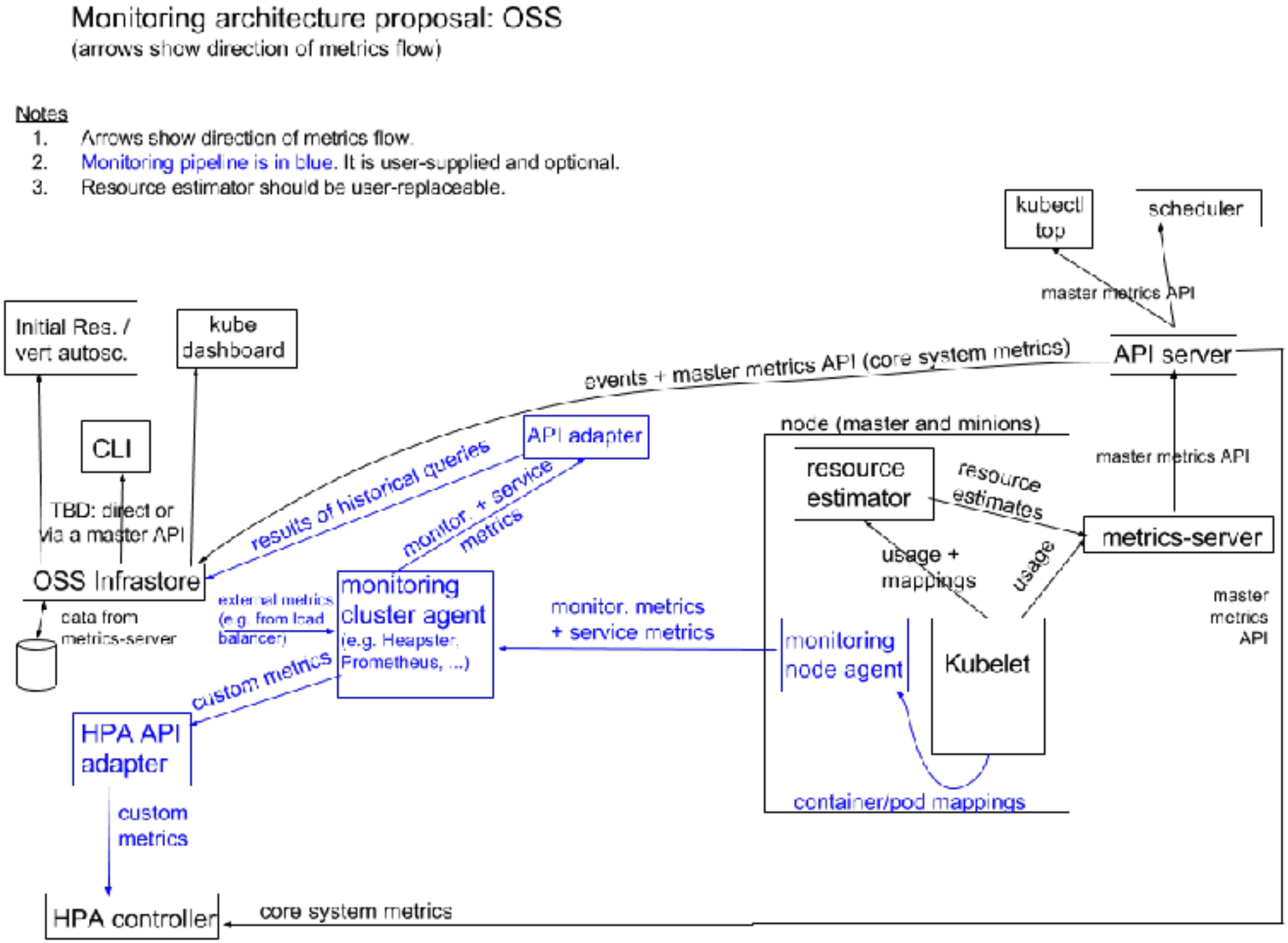
https://github.com/kubernetes/heapster/blob/master/docs/deprecation.md

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

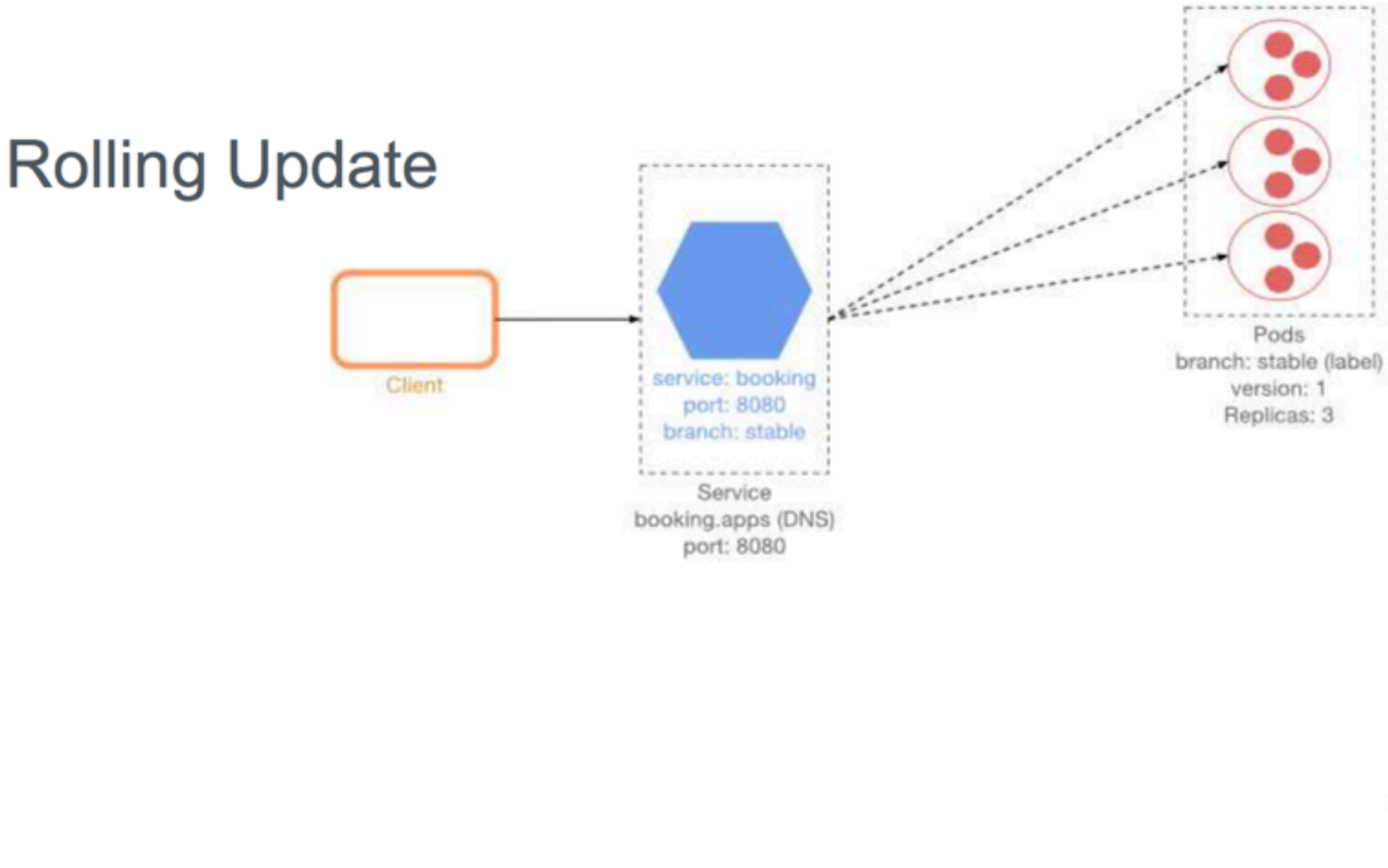
Architecture diagram



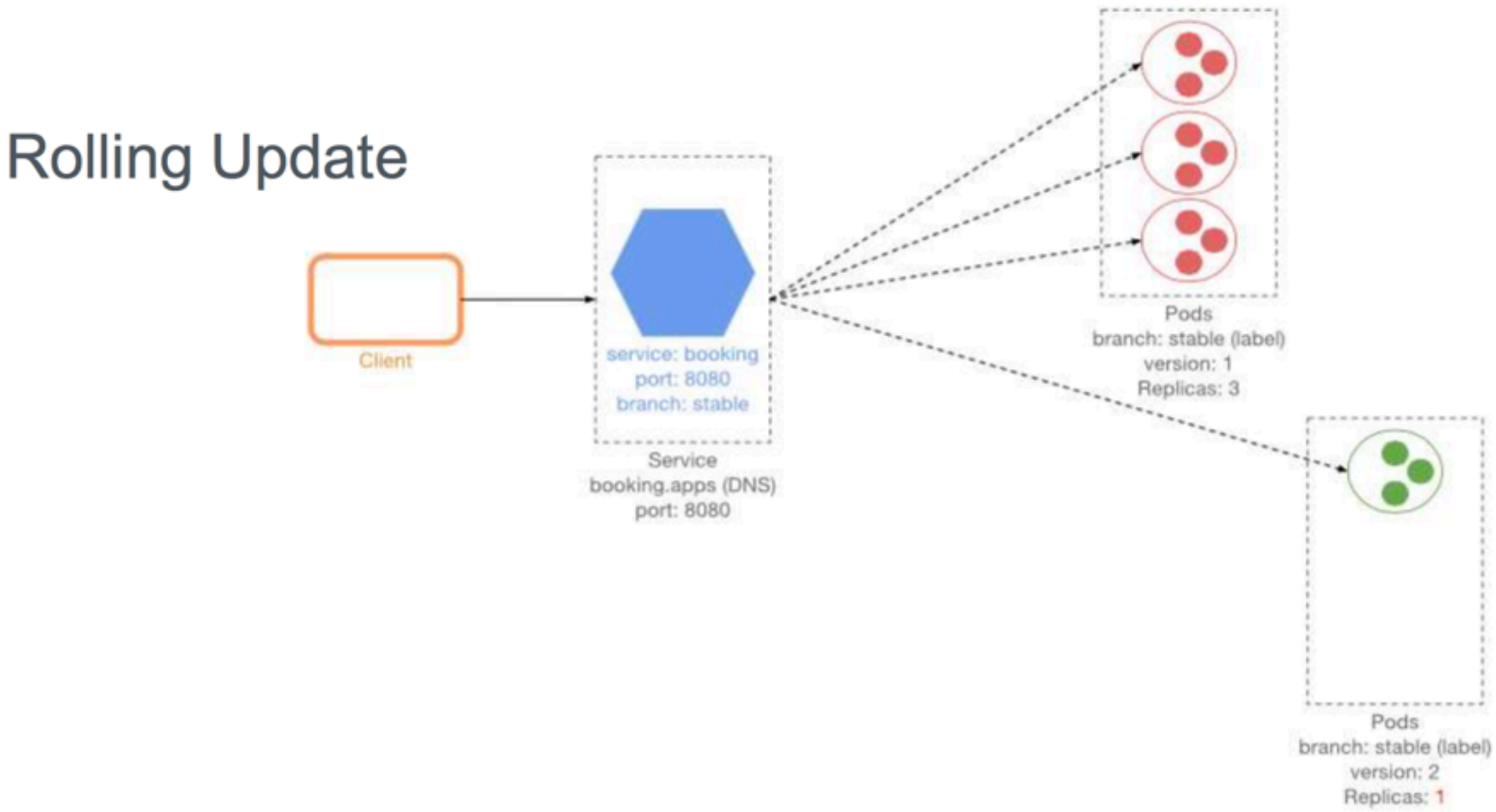
https://github.com/kubernetes/community/blob/master/contributors/design-proposals/instrumentation/monitoring_architecture.md

Automated rollouts & rollbacks

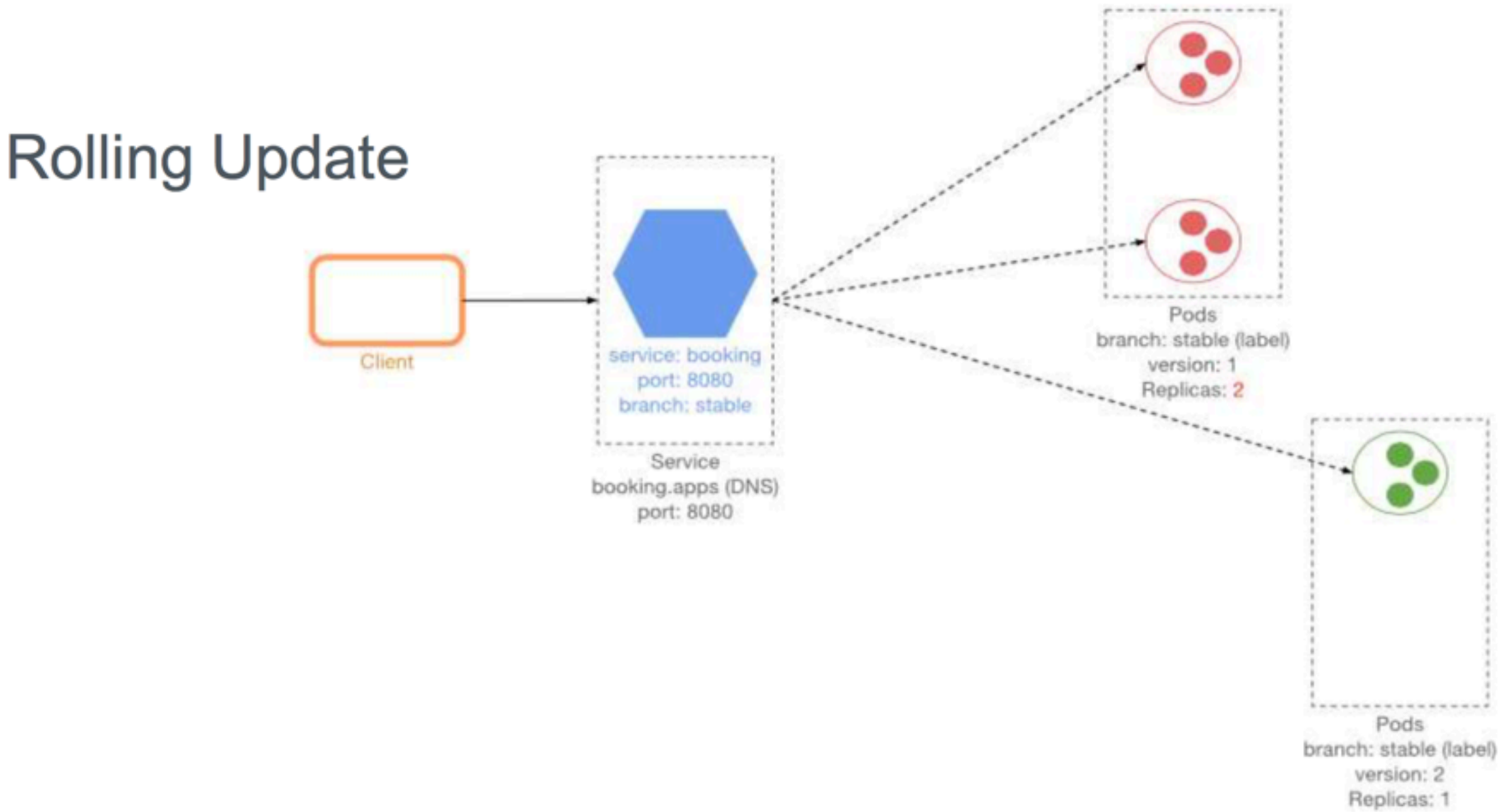
Rolling update (1)



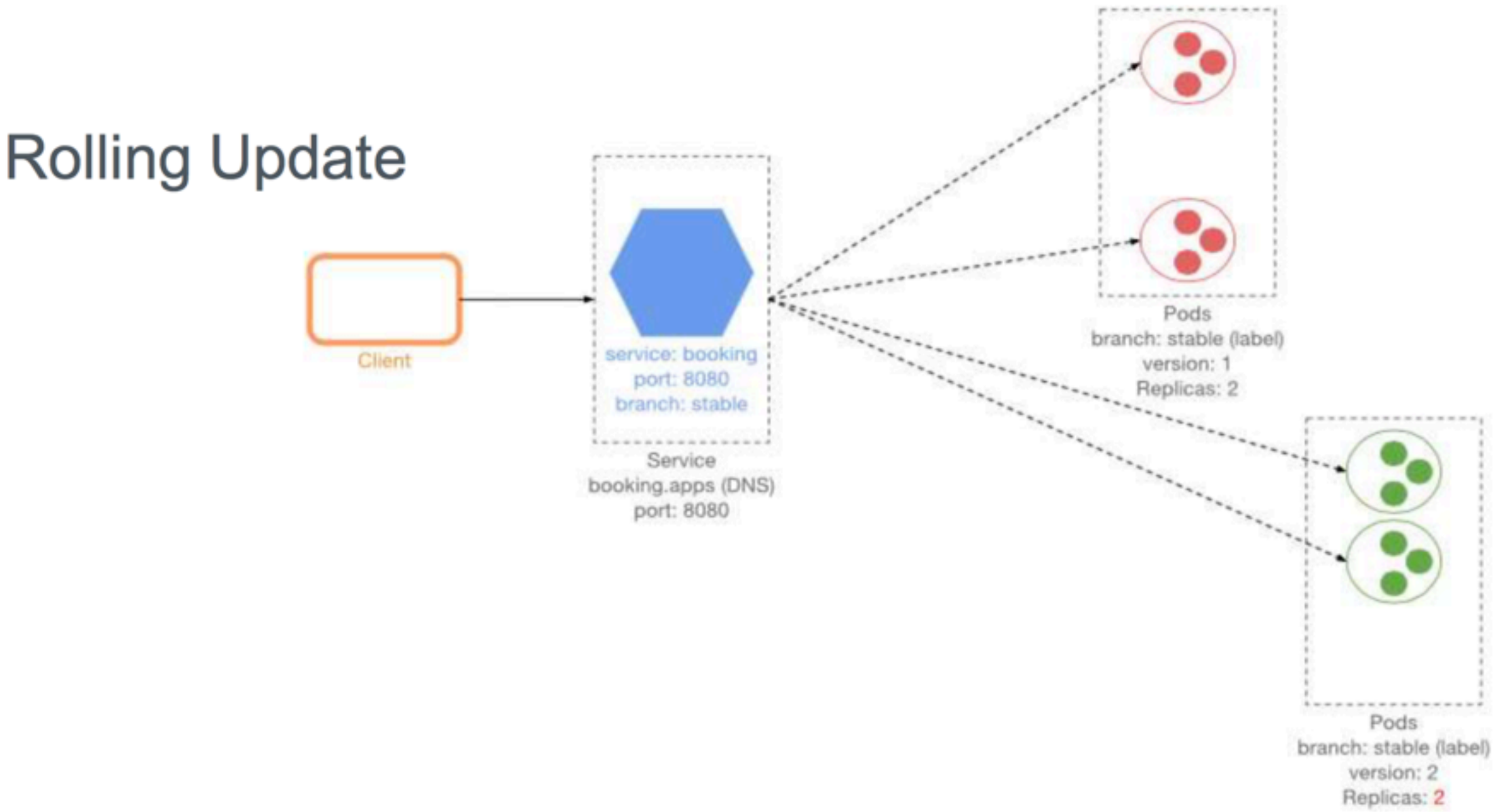
Rolling update (2)



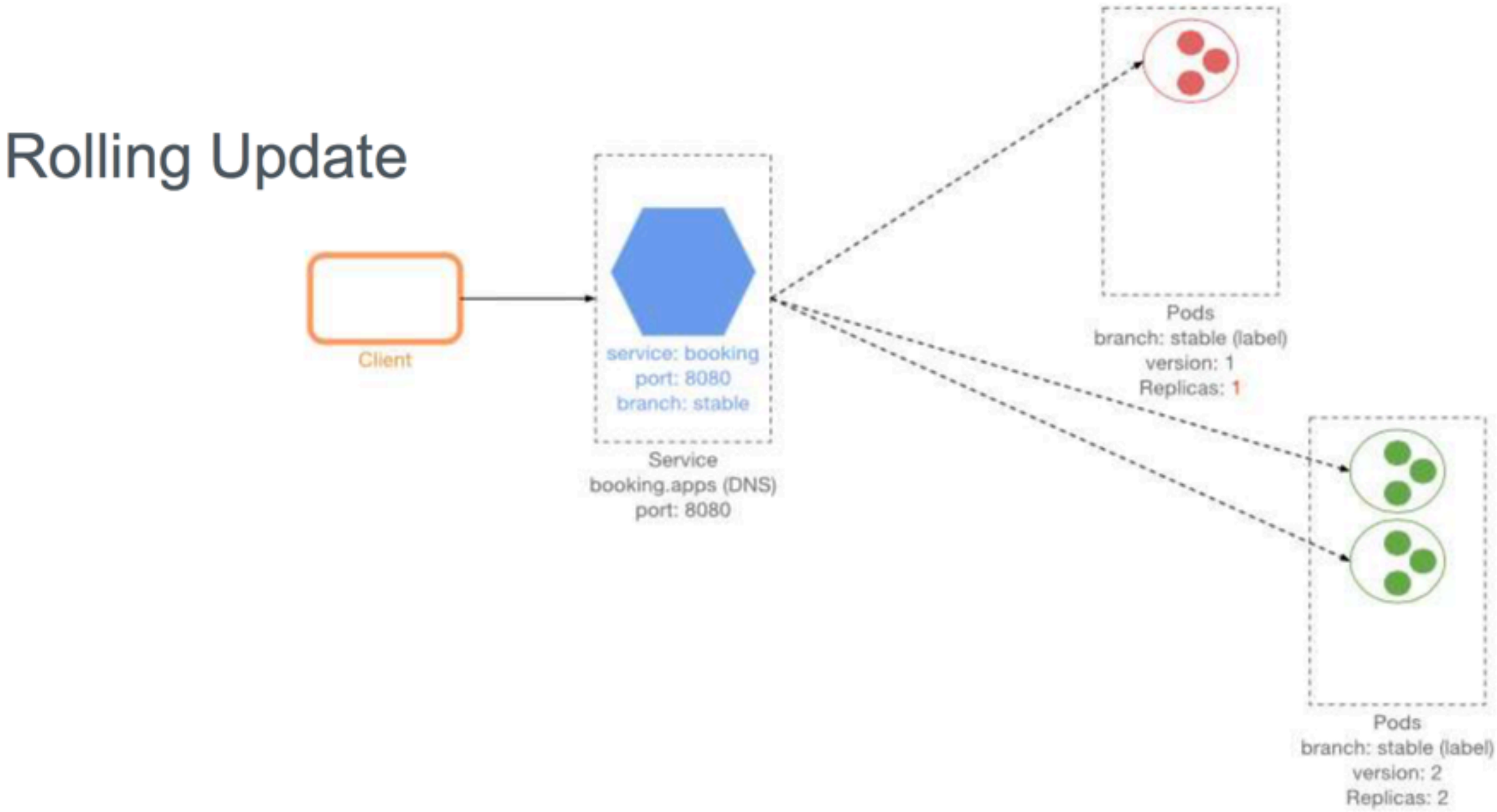
Rolling update (3)



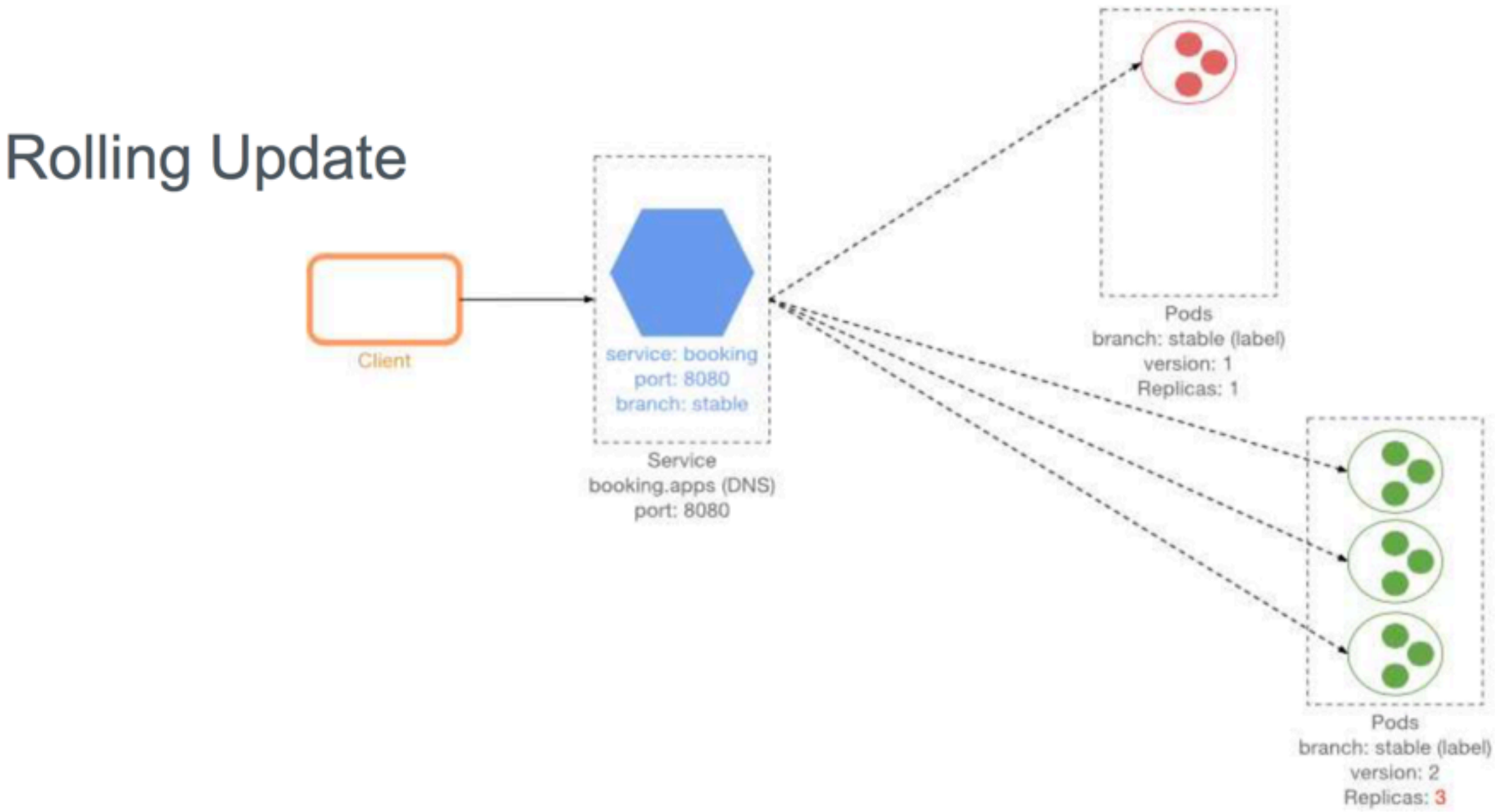
Rolling update (4)



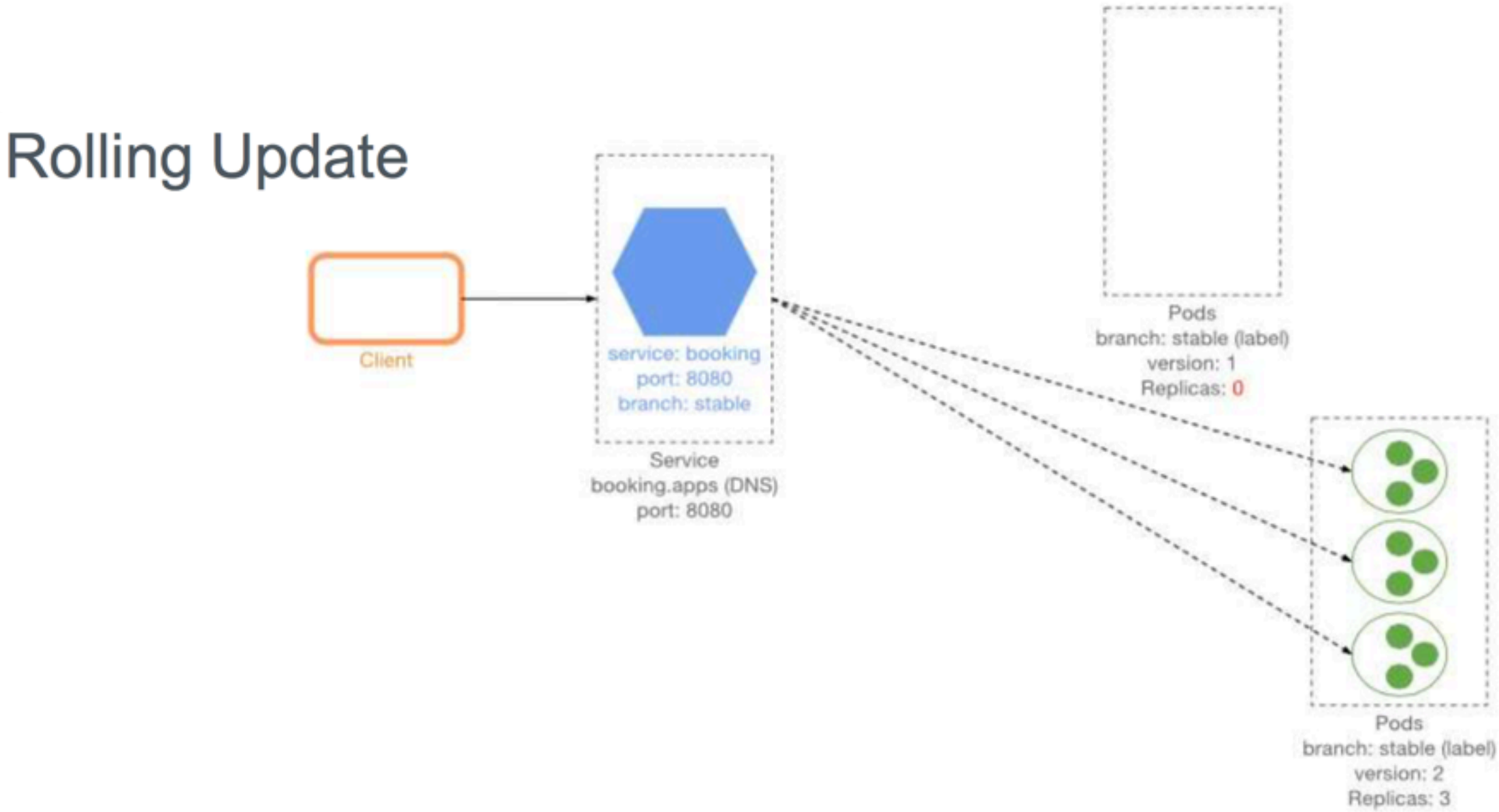
Rolling update (5)



Rolling update (6)



Rolling update (7)



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

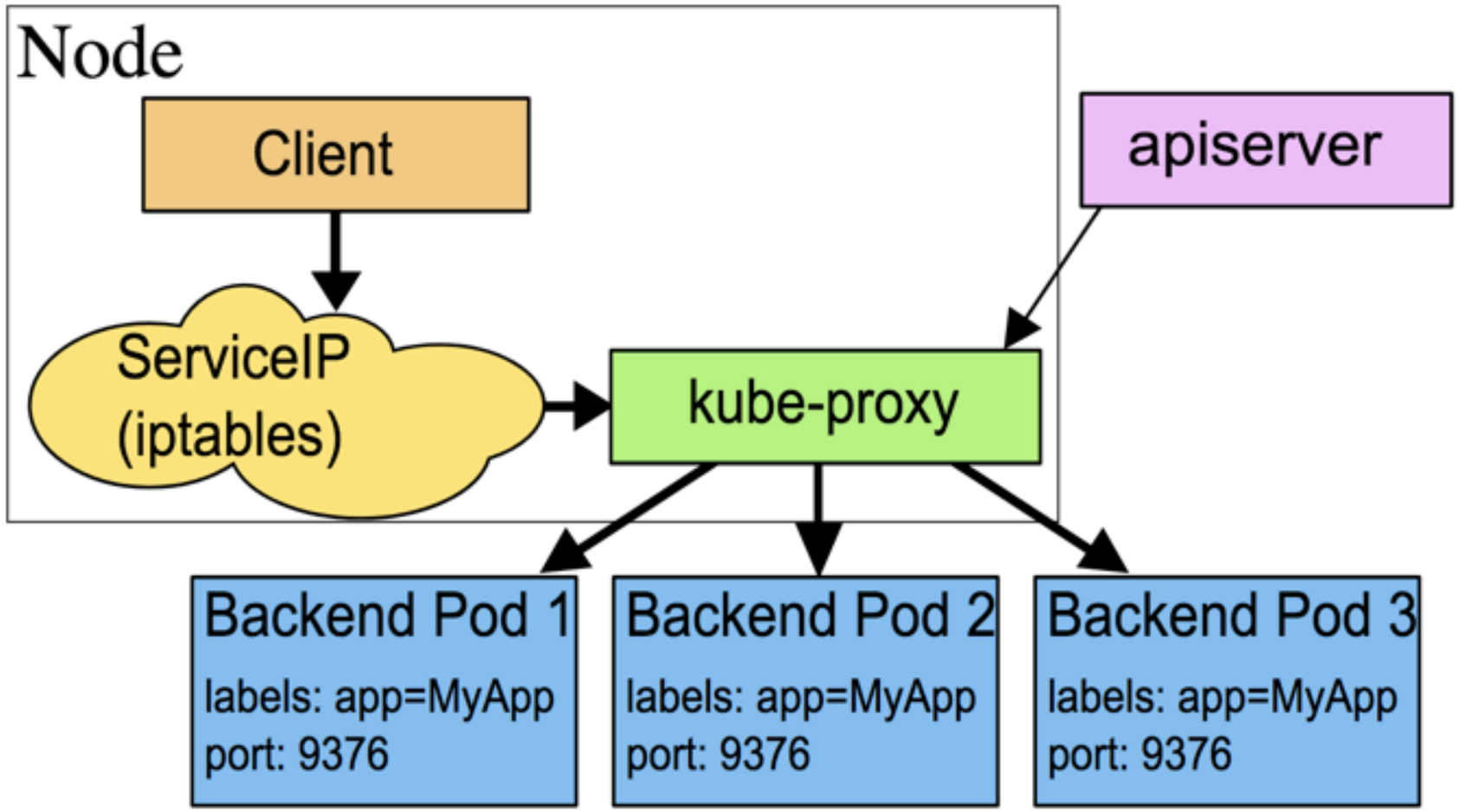
Service is the connector/proxy for client to connect with Pod

Discovery use for service to look Pod

Support **load balance** between Pods (replica)

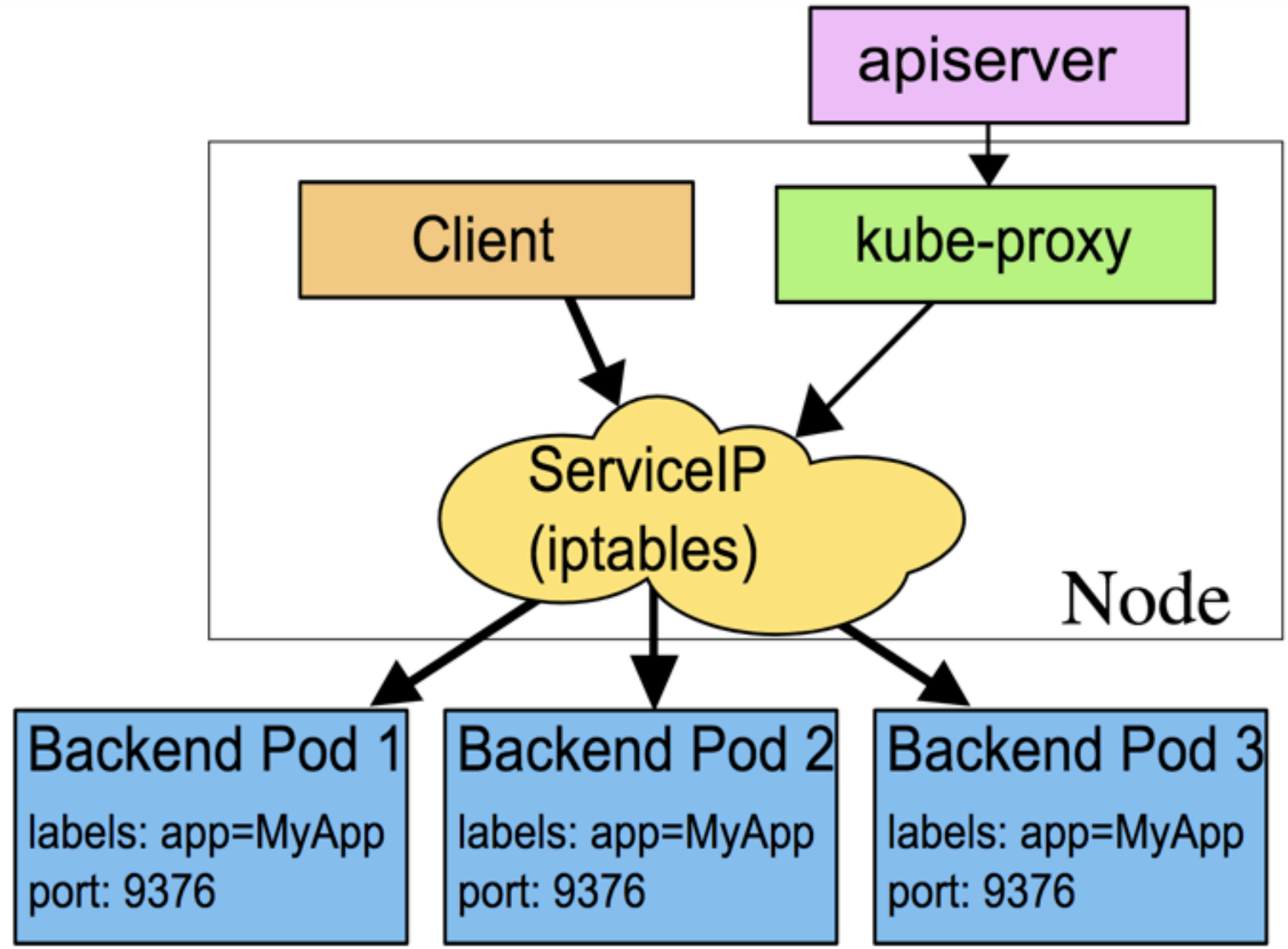
https://kubernetes.io/docs/concepts/services-networking/service/

Proxy-mode: userspace



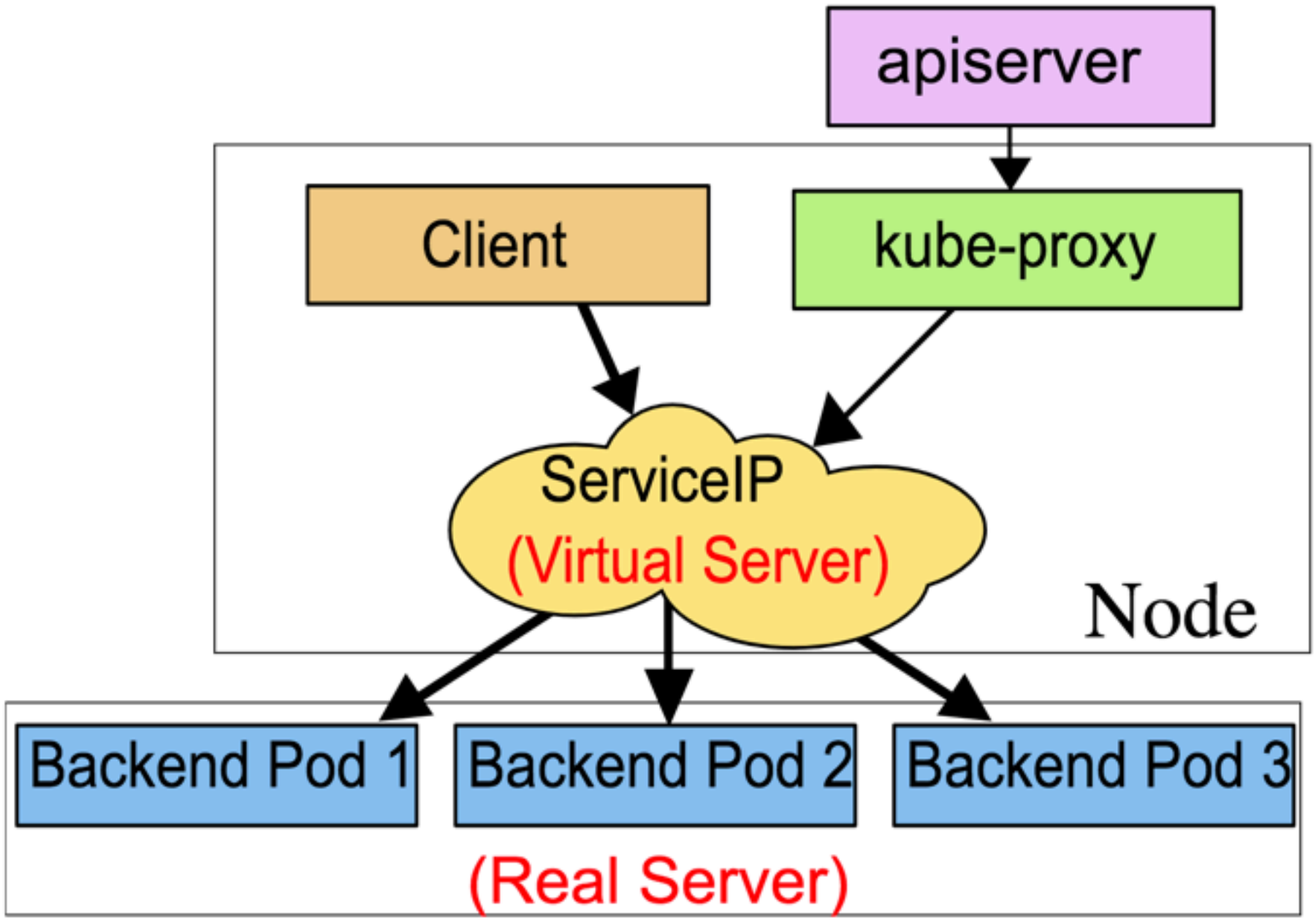
Note that in the above diagram, `clusterIP` is shown as `ServiceIP`.

Proxy-mode: iptables



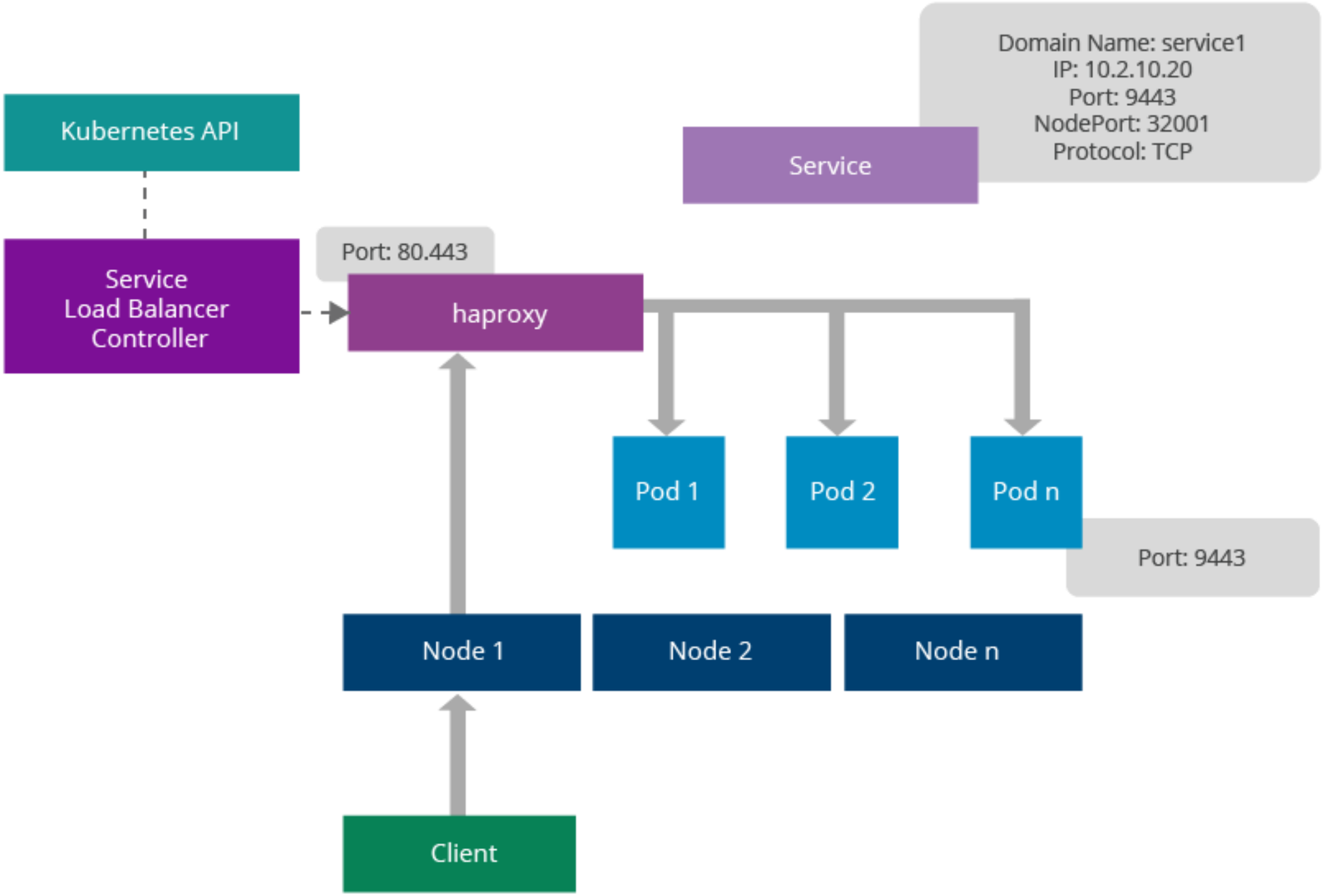
Note that in the above diagram, `clusterIP` is shown as `ServiceIP`.

Proxy-mode: ipvs (beta 1.9)

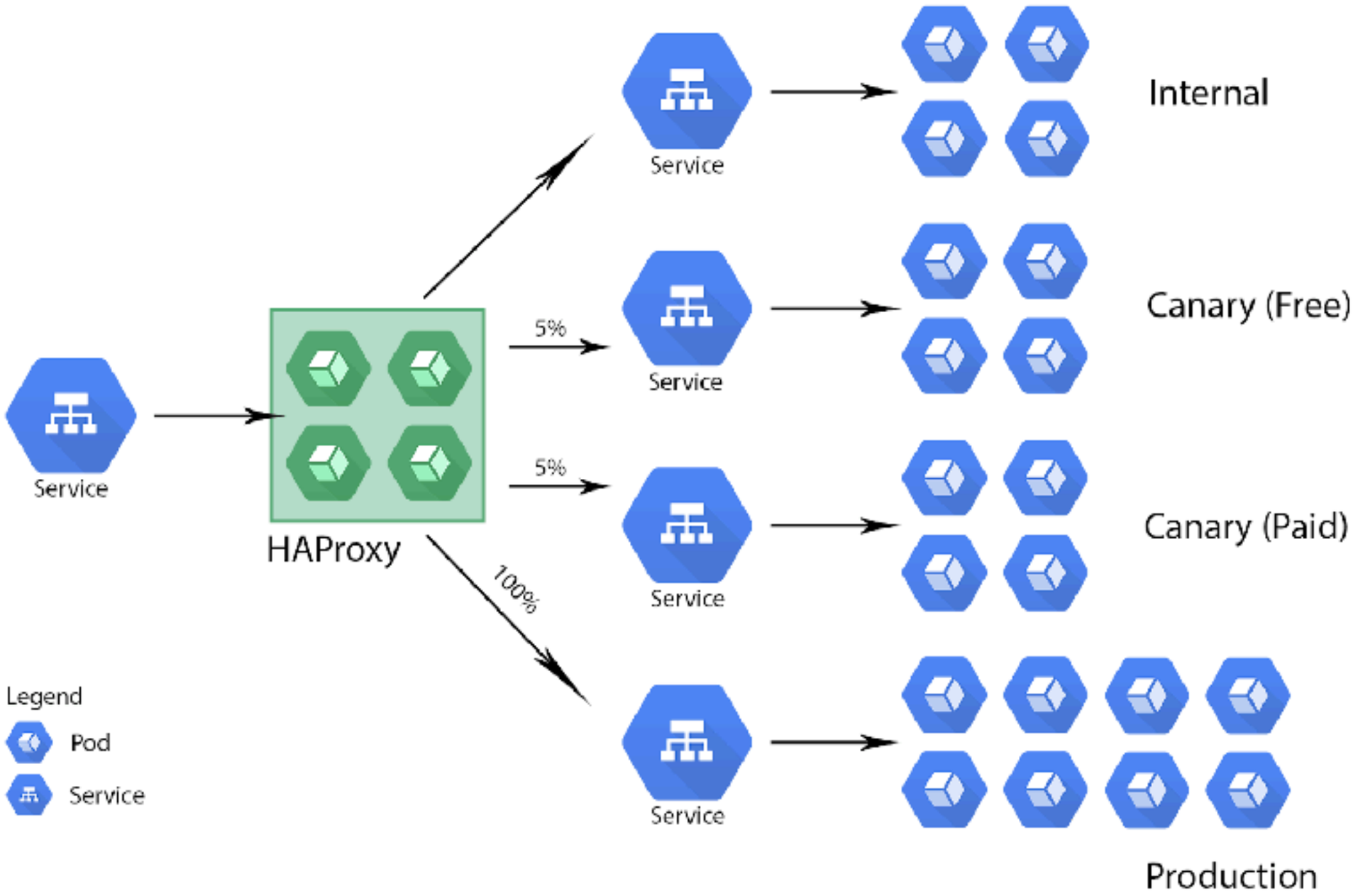


Sticky session management using Service Load Balancers

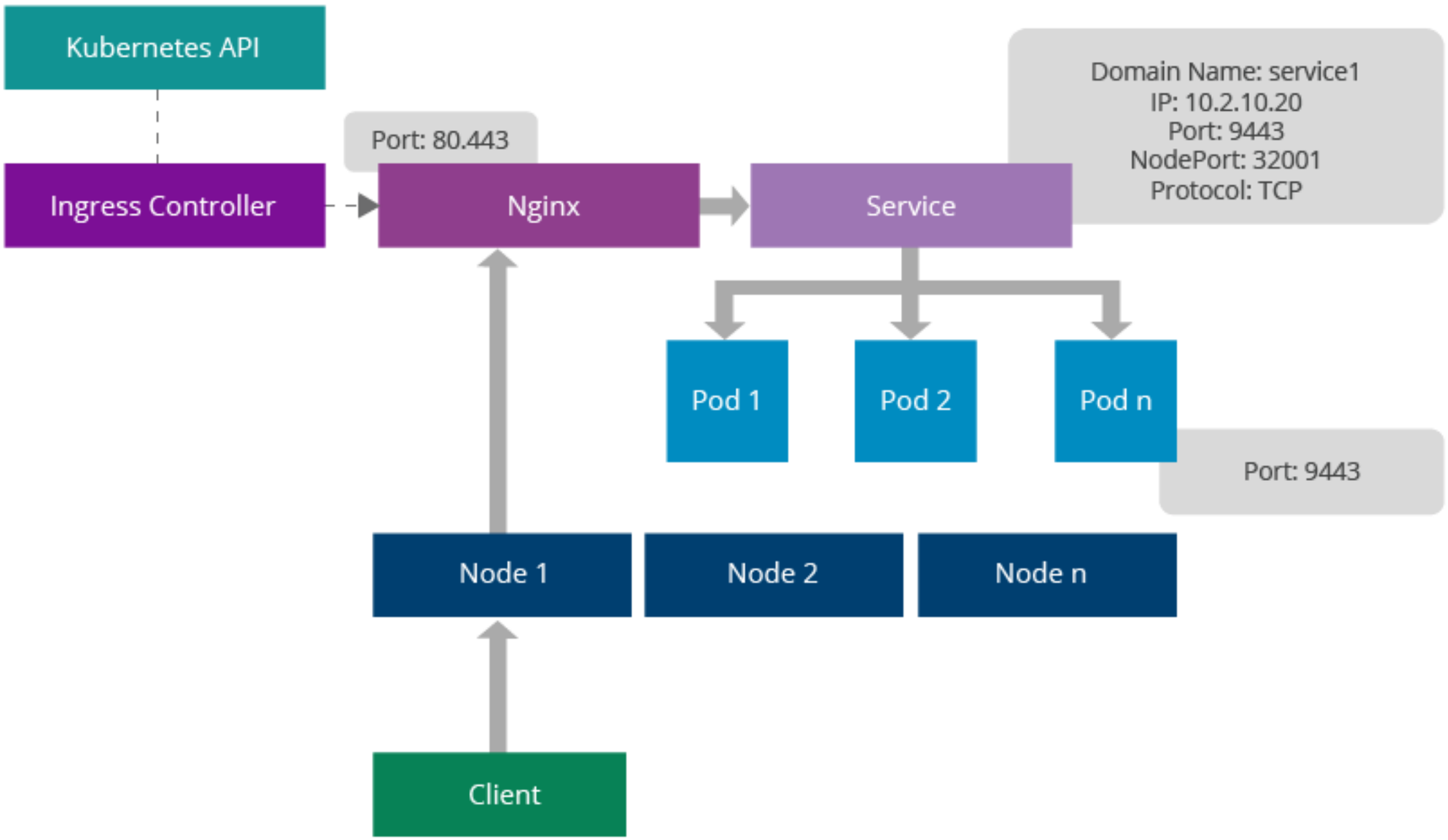
Service Load Balancer



Service Load Balancer



Ingress controller



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

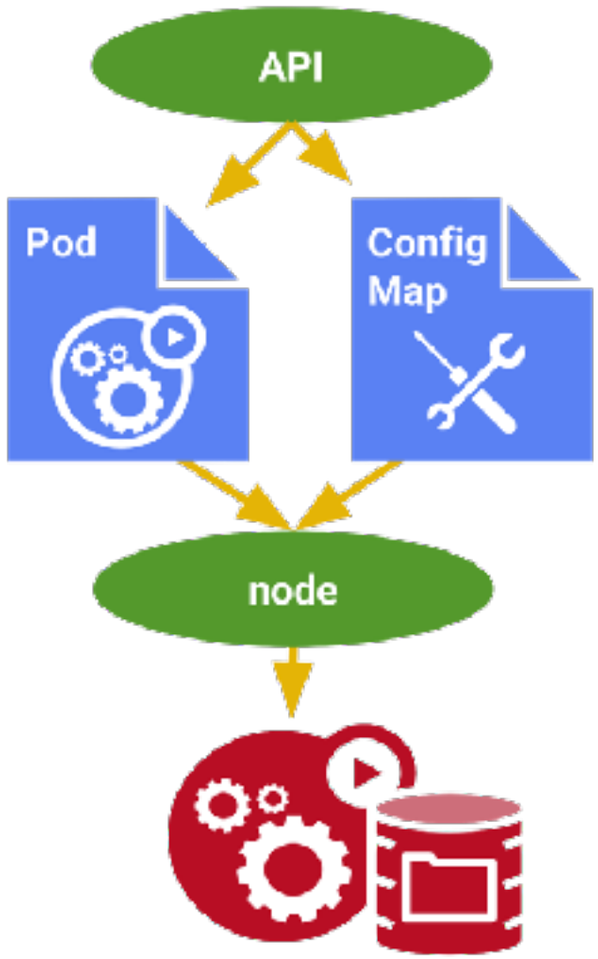
Keep confidential data to running app in encryption format

$kubectl get secret

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

Secret and configuration mgt

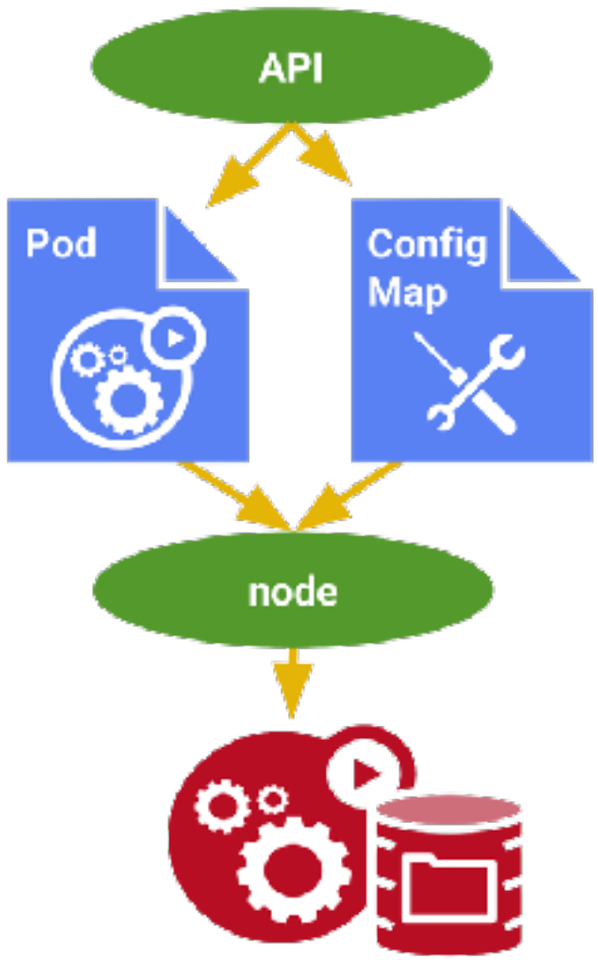
Use **ConfigMap** to define all configuration that reference from Pod



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

Secret and configuration mgt

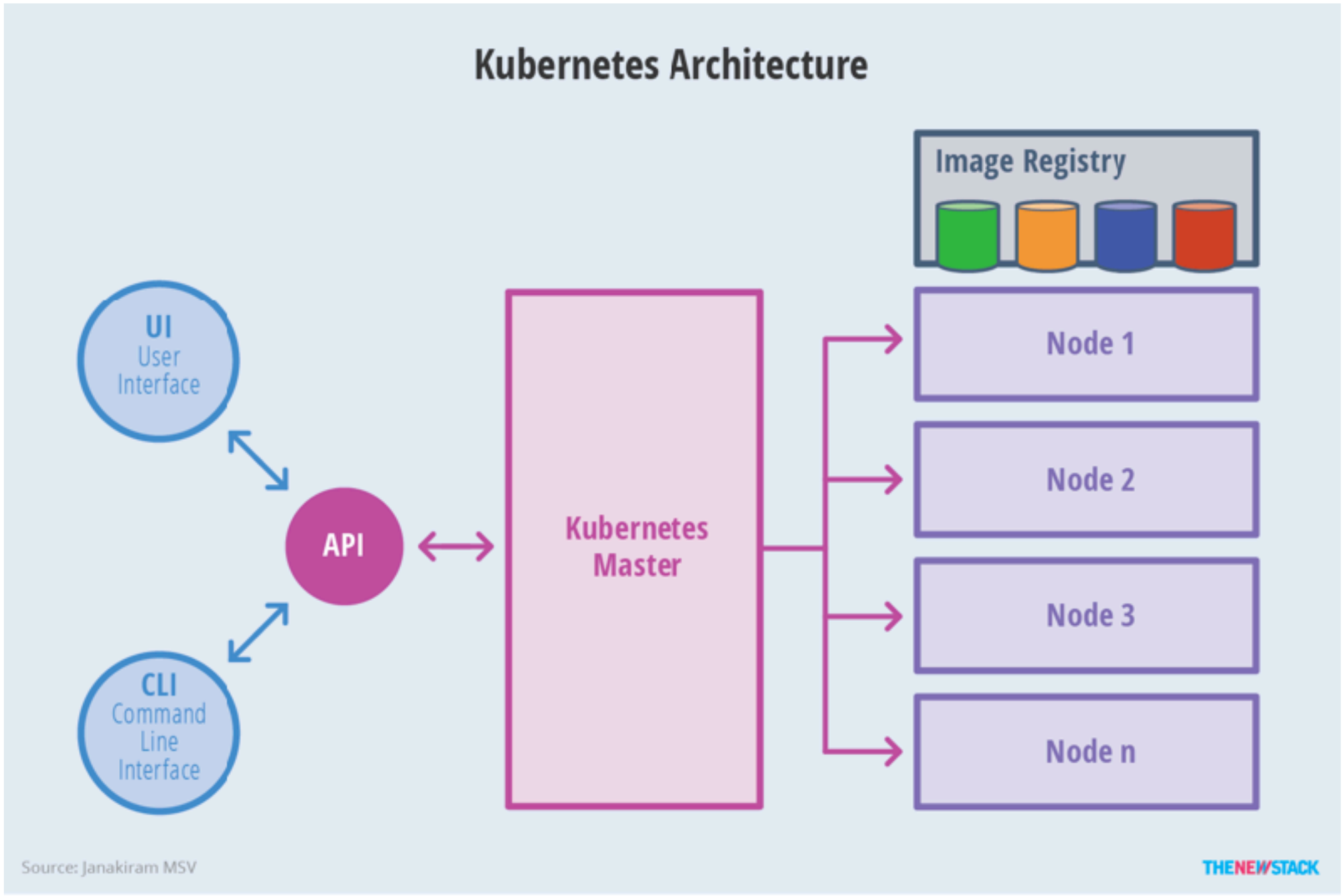
Come from **12 factor app** :: config come from the environment



https://12factor.net/

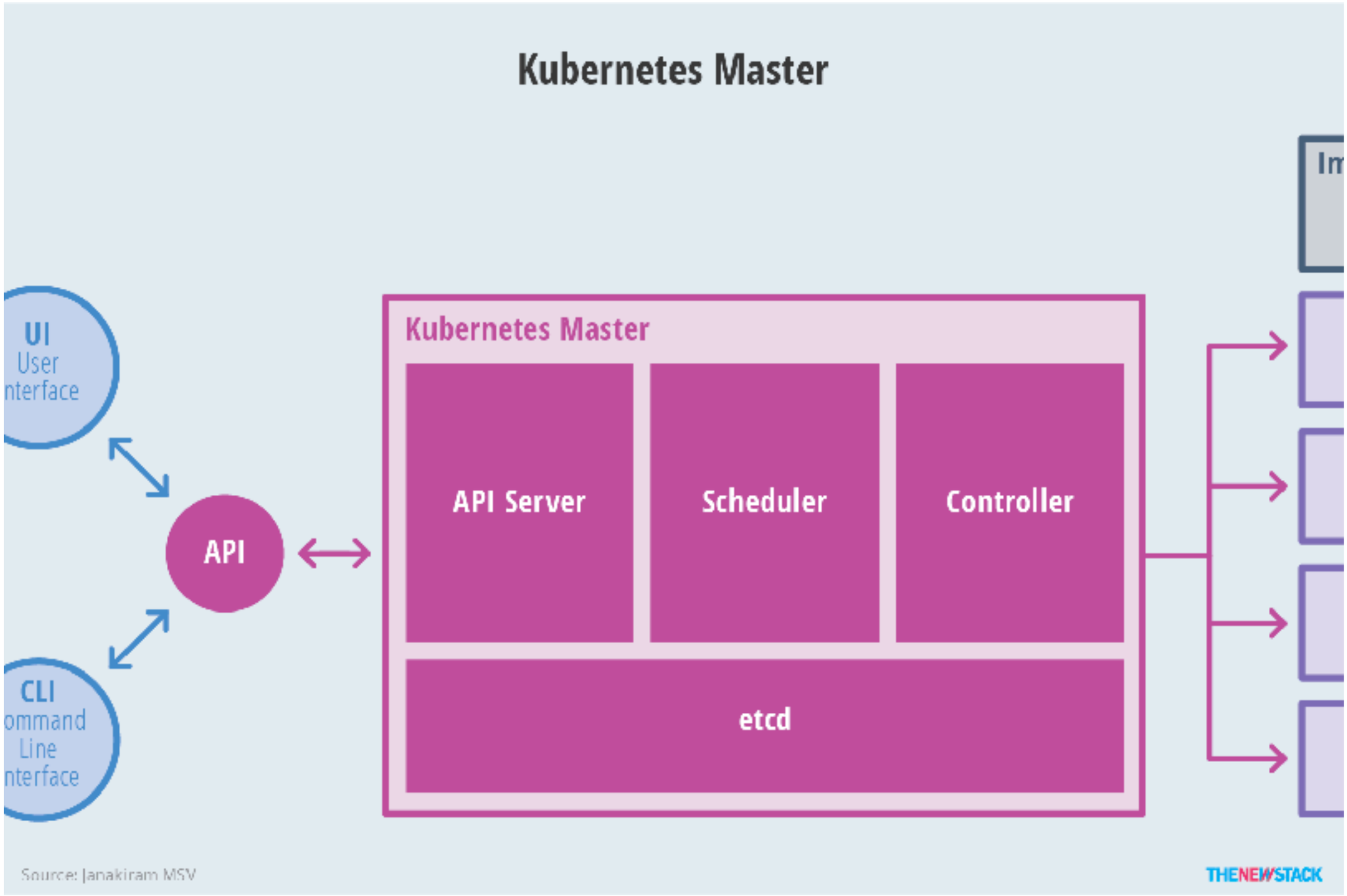
System architecture

System architecture



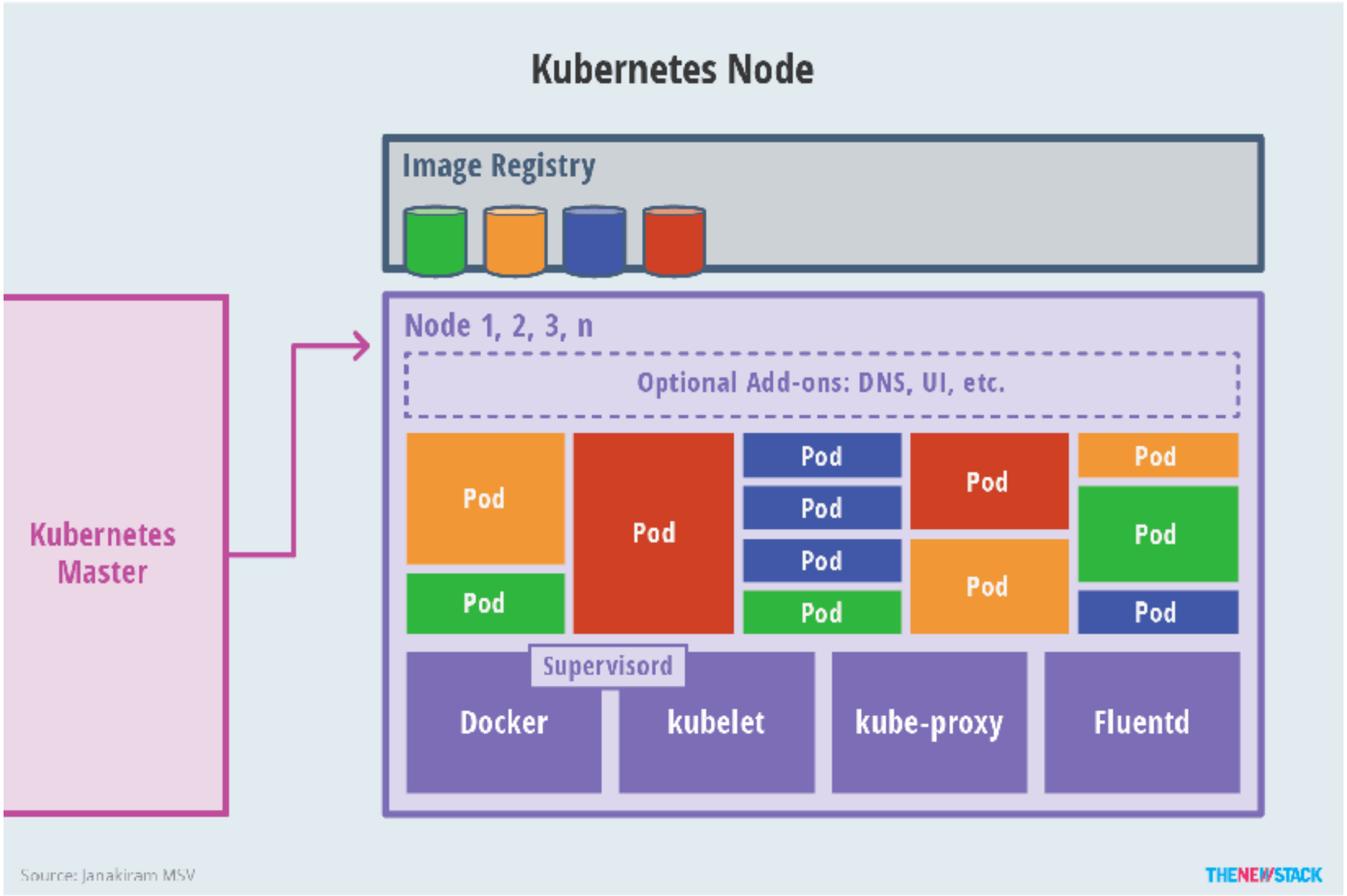
https://thenewstack.io/kubernetes-an-overview/

Master



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

Node

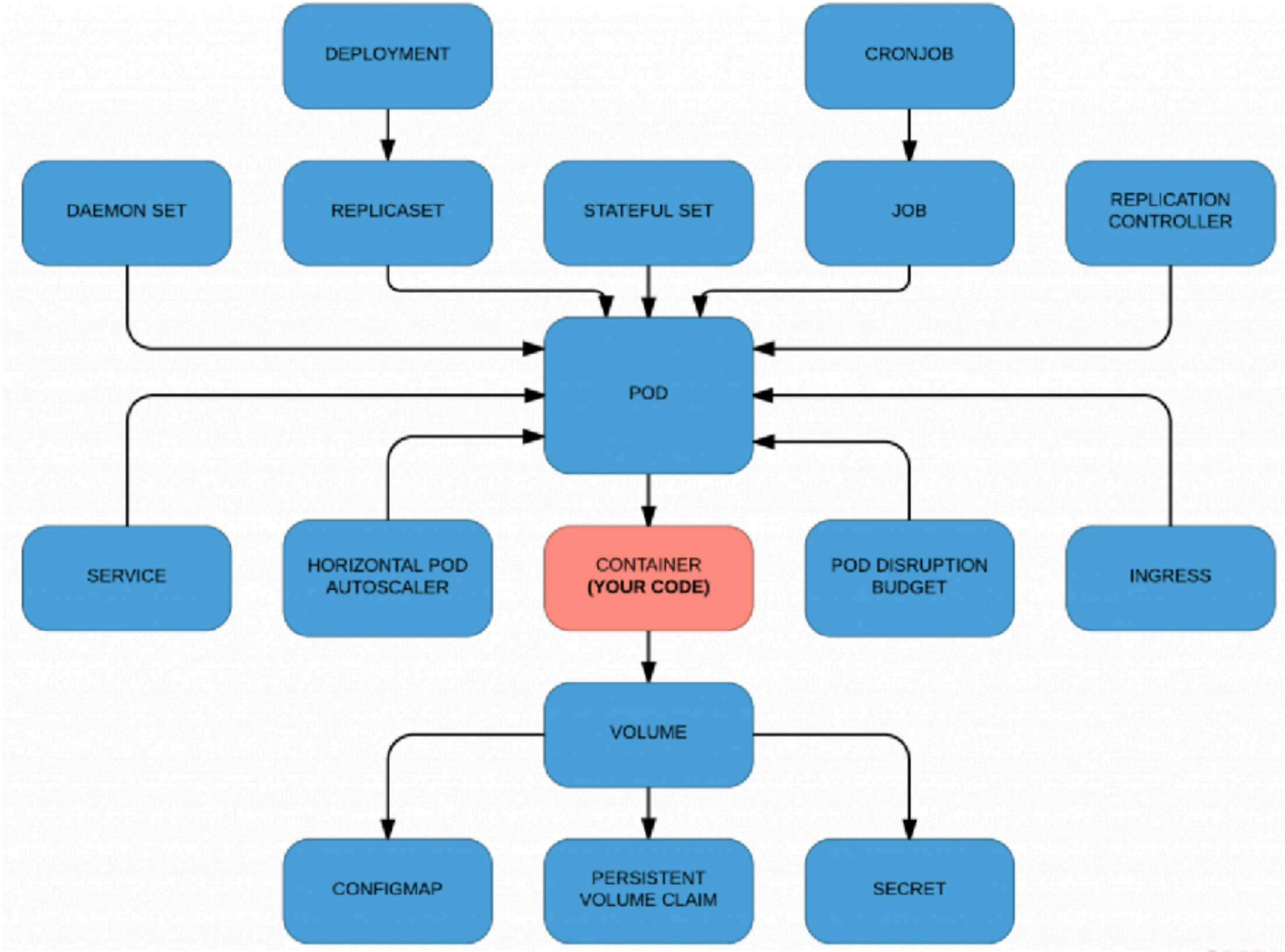


https://thenewstack.io/kubernetes-an-overview/

Core concepts of Kubernetes

Core concepts

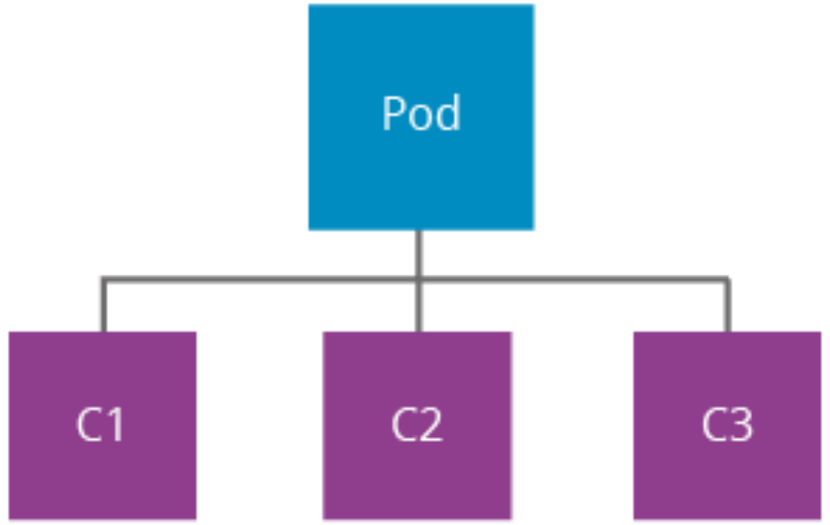
Pods vs containers
Services
Replication Controllers
Deployments
ConfigMap and Secret
Volumes
StatefulSets



Pods vs containers

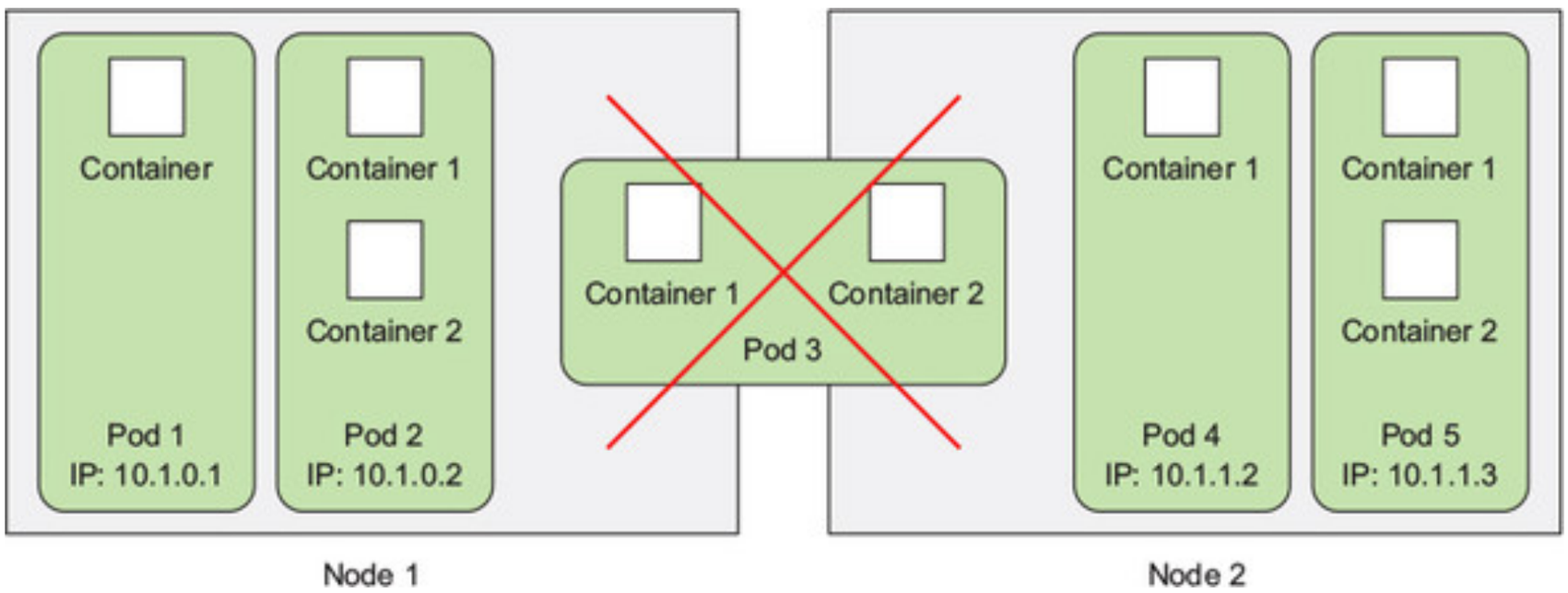
Pods

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



Pods

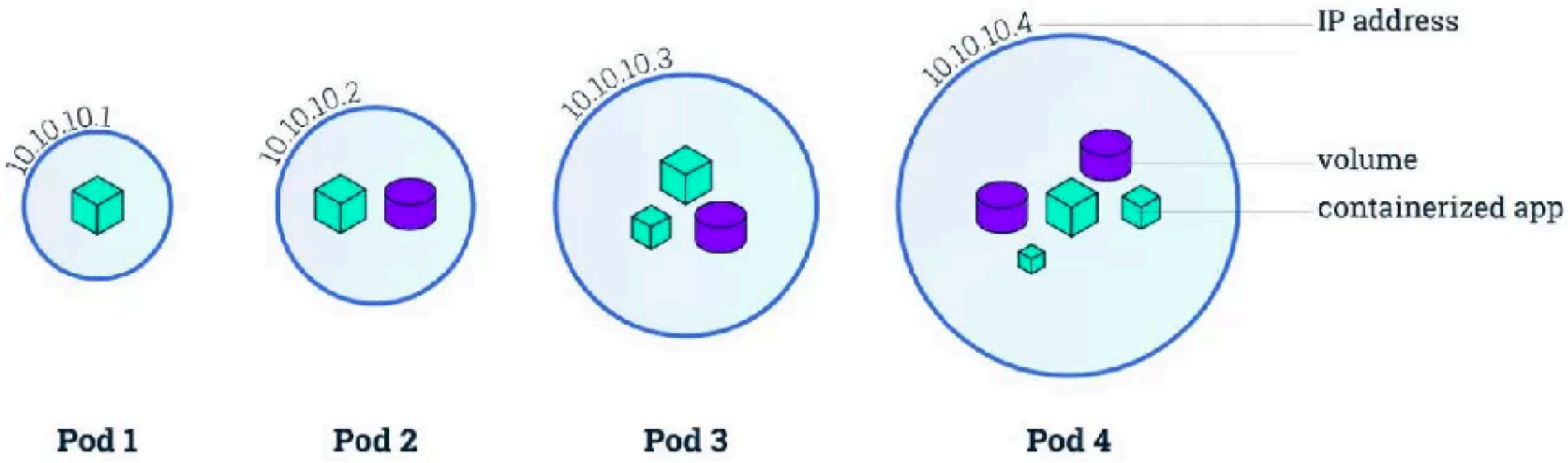
1 pods = 1 container
1 pods = N containers



All containers in same Pods

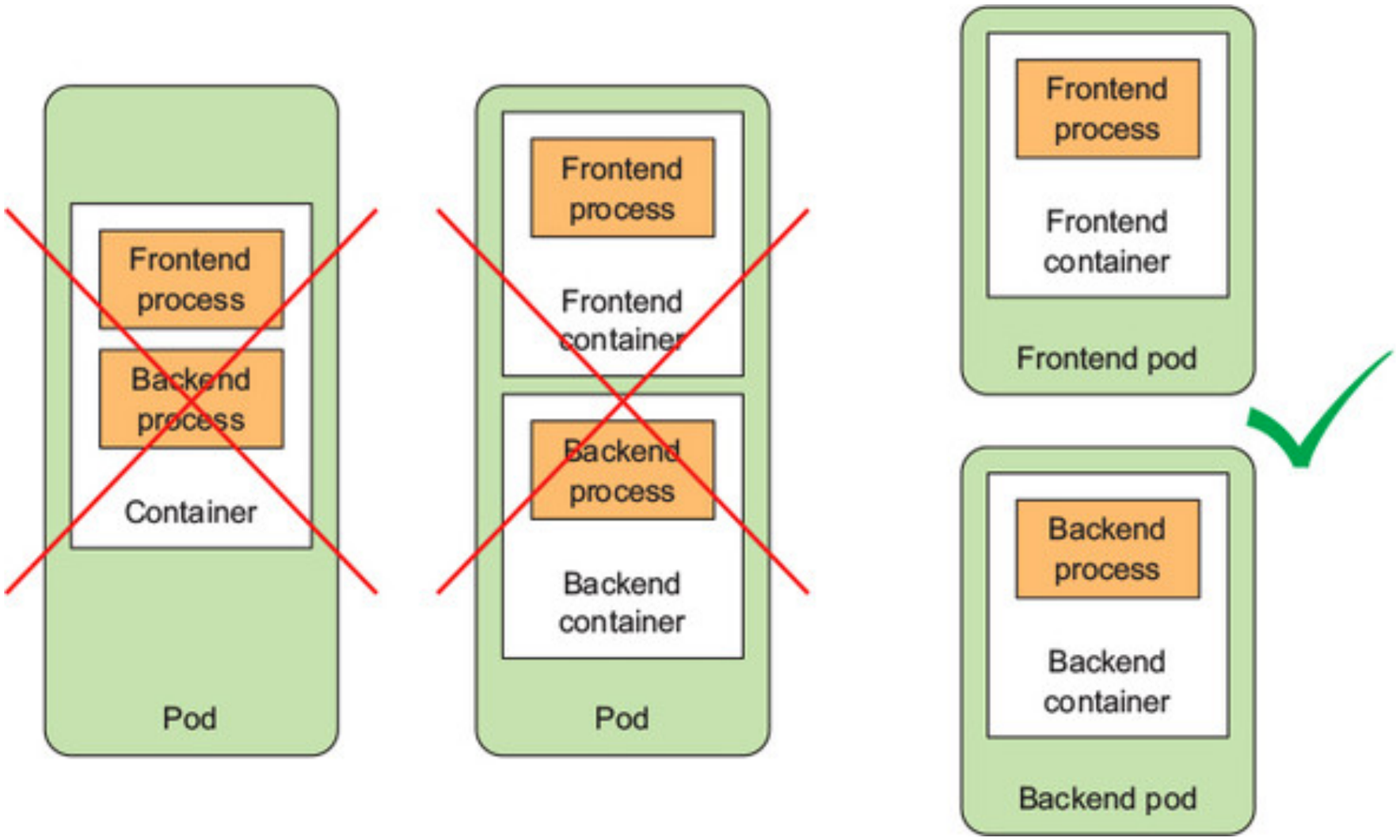
Share process ID
Share network interface
Share Hostname/IP/Port
Share Unix Time Sharing (UTS)

Pods



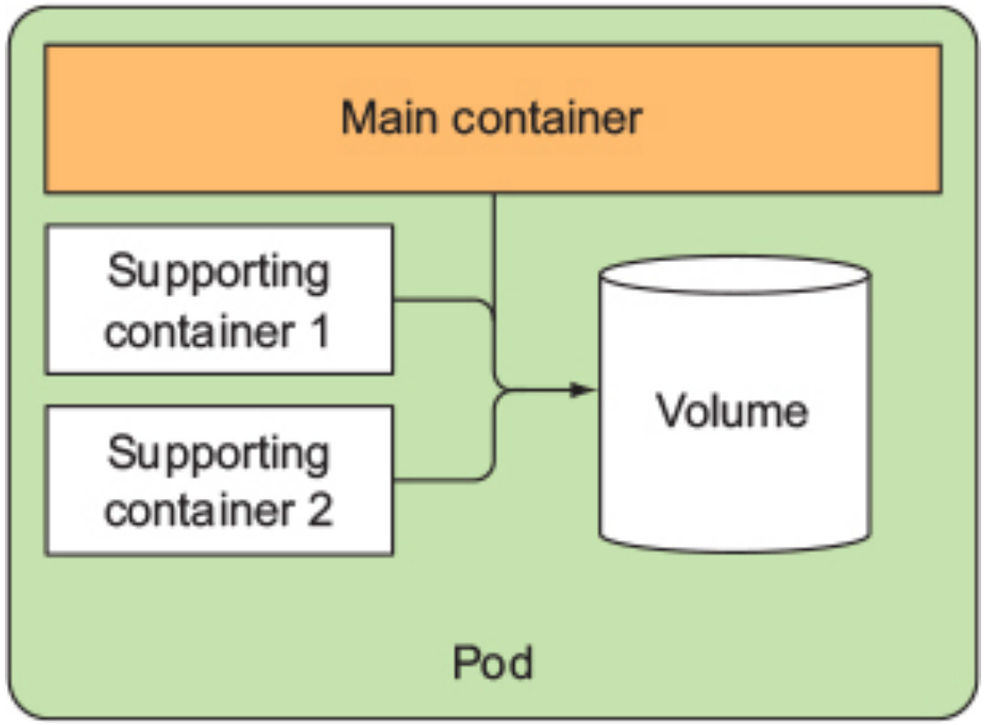
Organize container across Pods

Split multi-tiers app into multiple pods



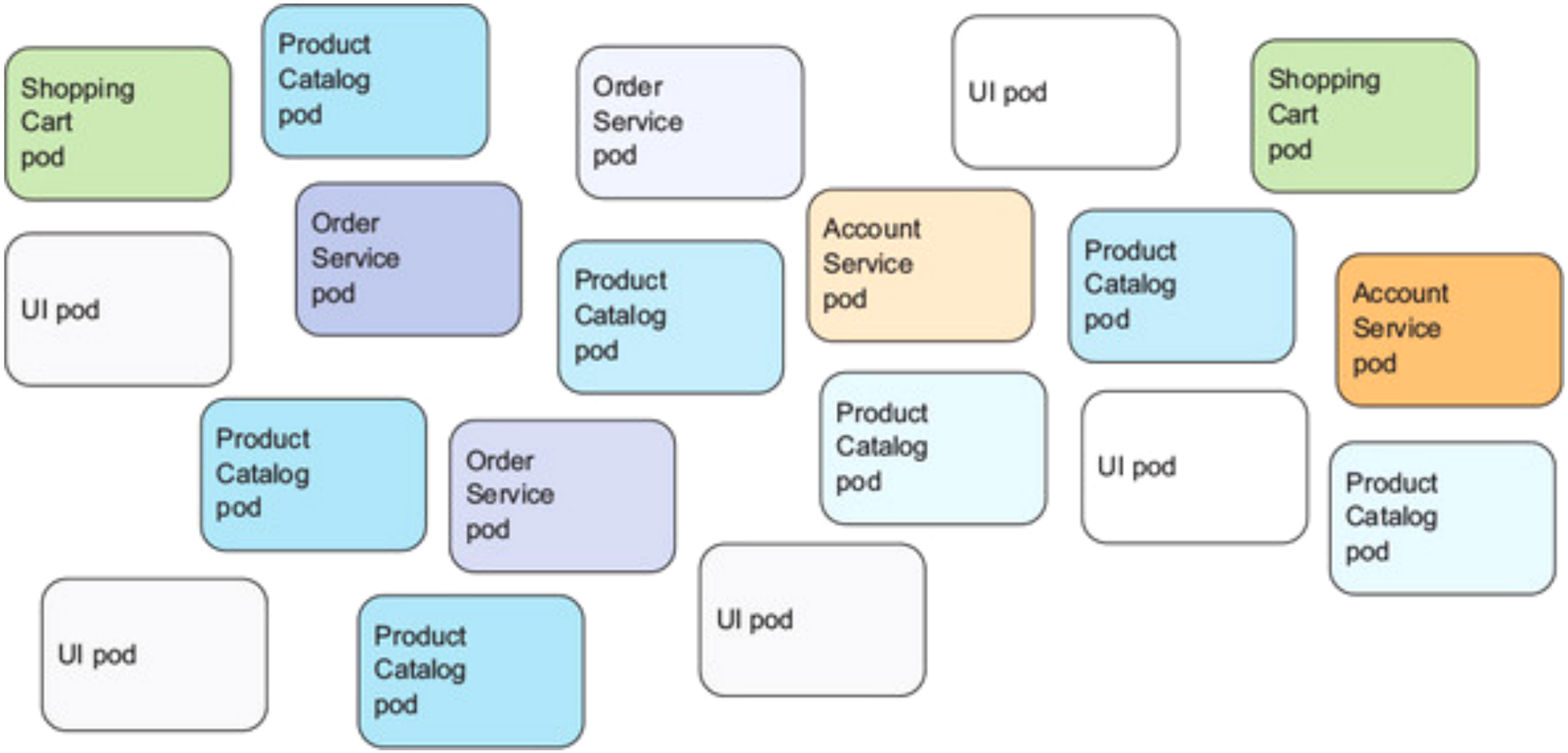
Organize container across Pods

When to use multiple containers in a pods

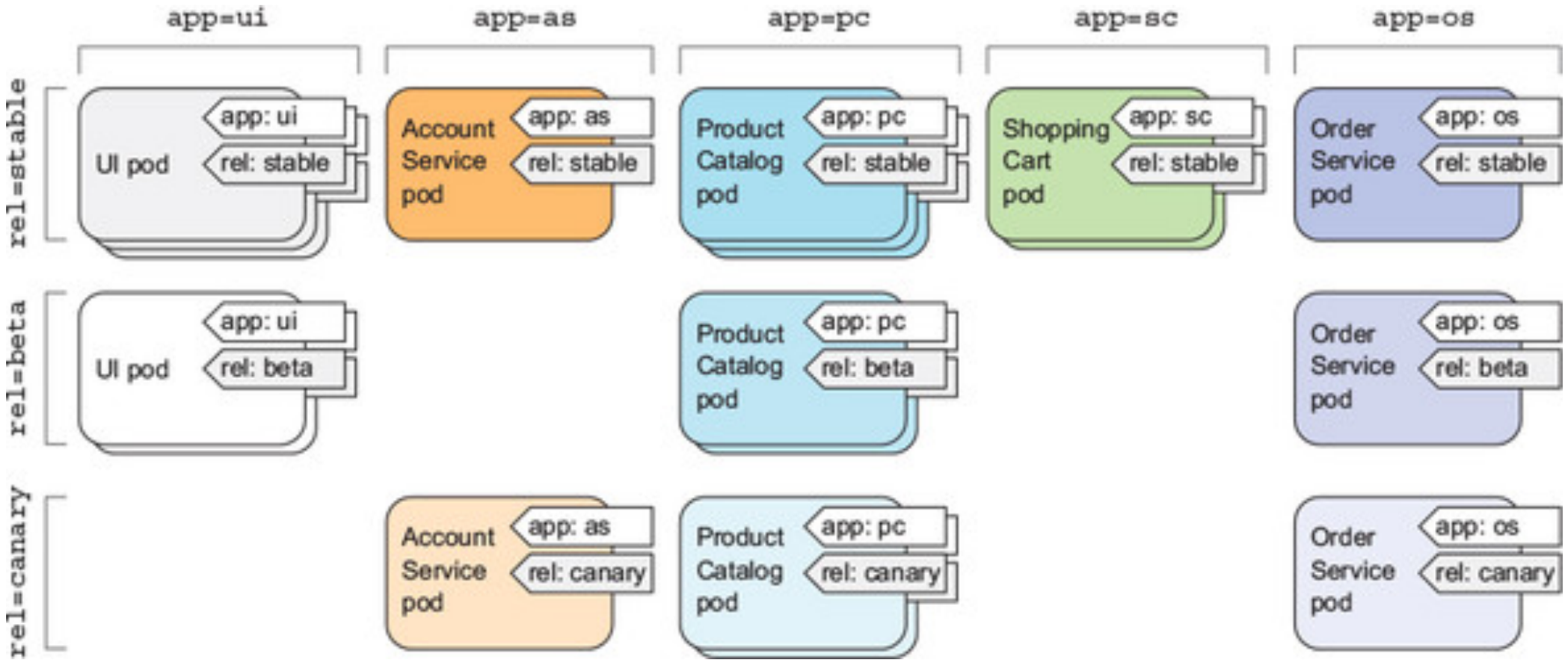


Organize pods with Labels

Organize pods with Labels

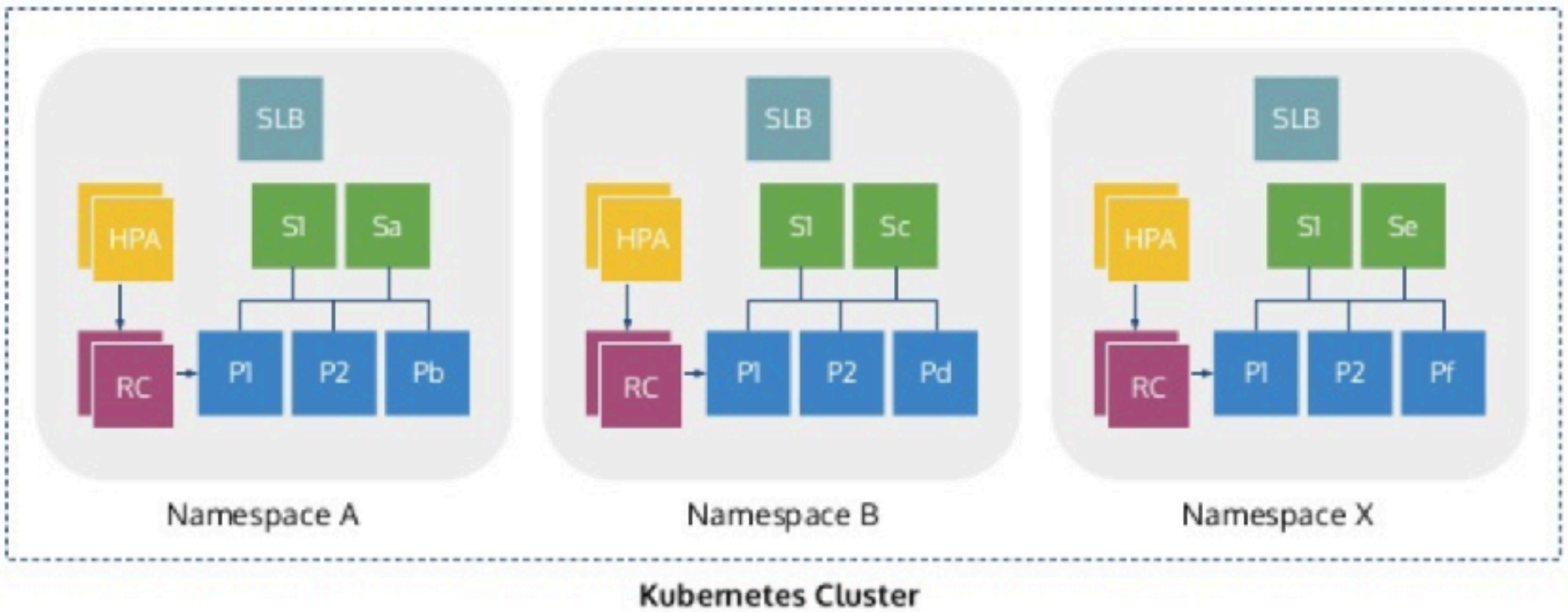


Organize pods with Labels



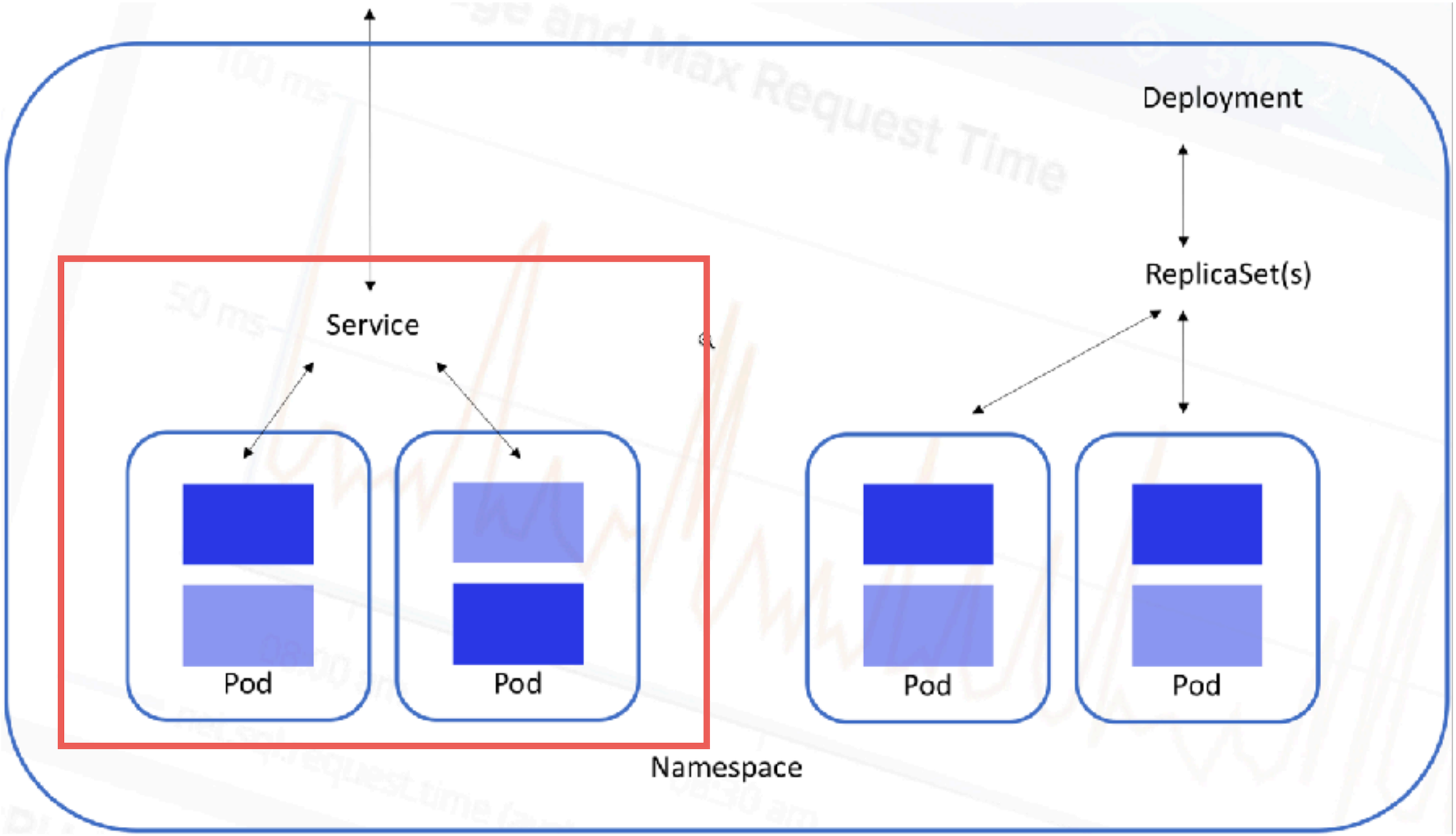
Pods namespaces

Allow different teams to use the same cluster

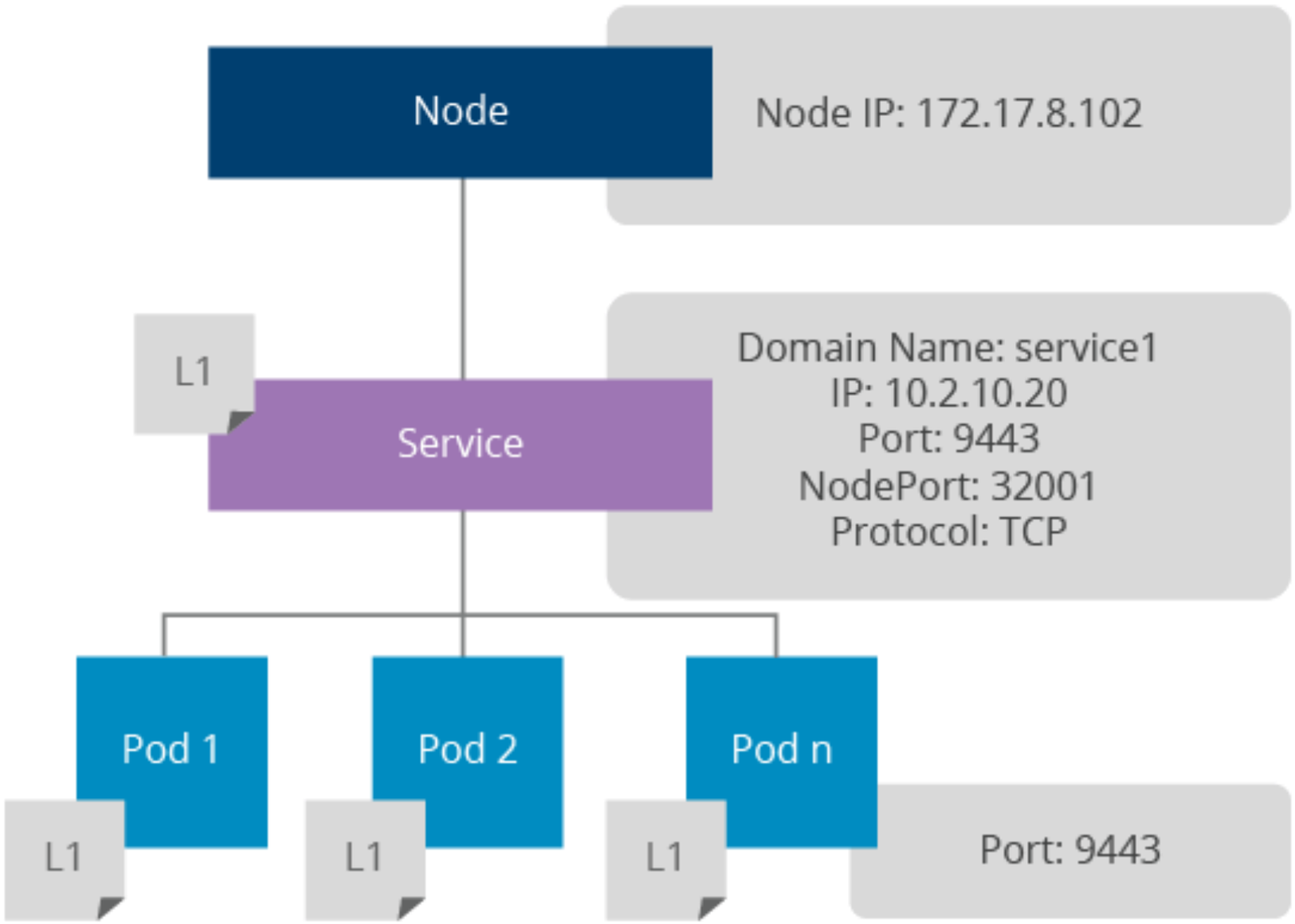


Services discovery and Load balancing

Services



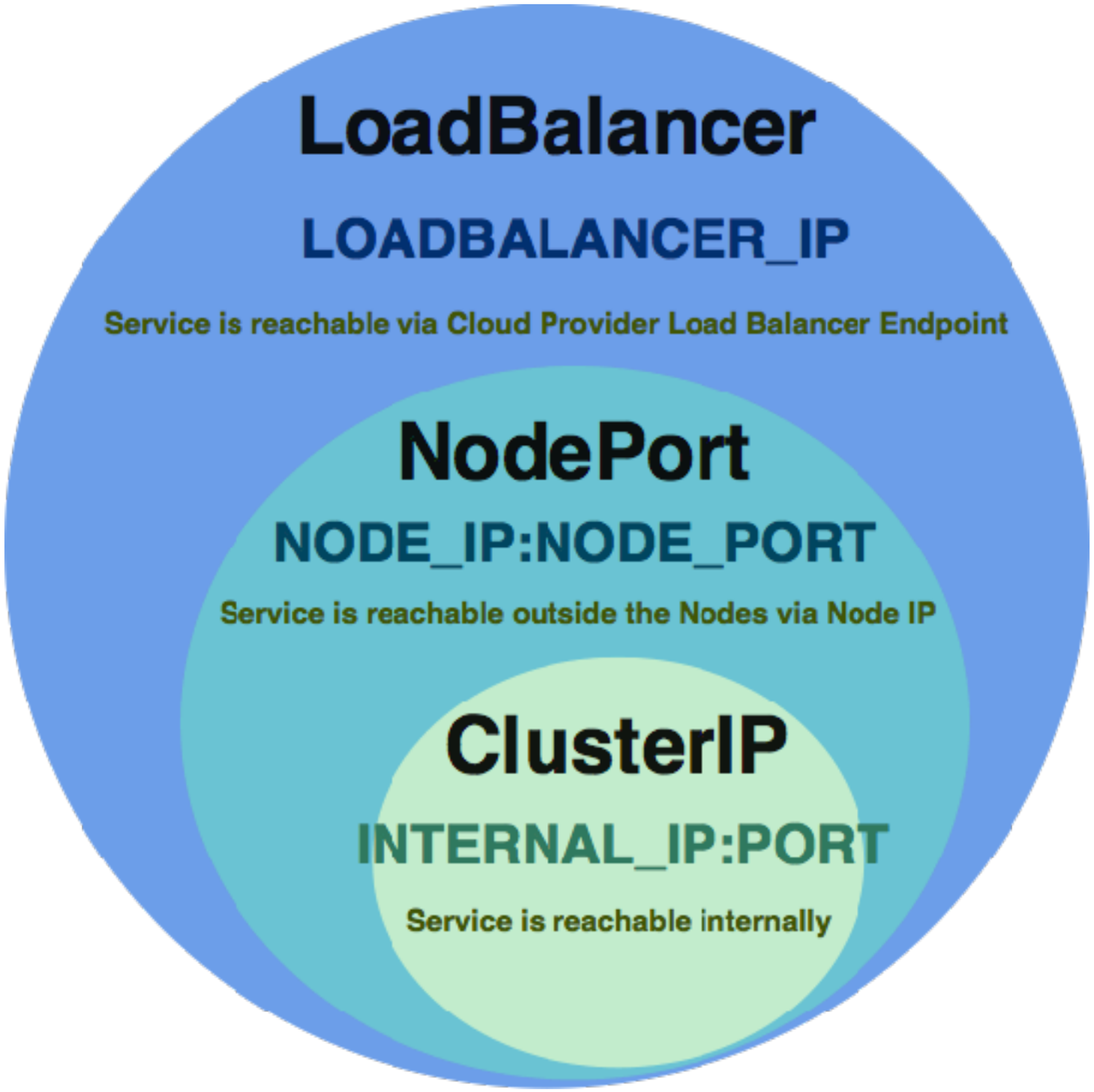
Services



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

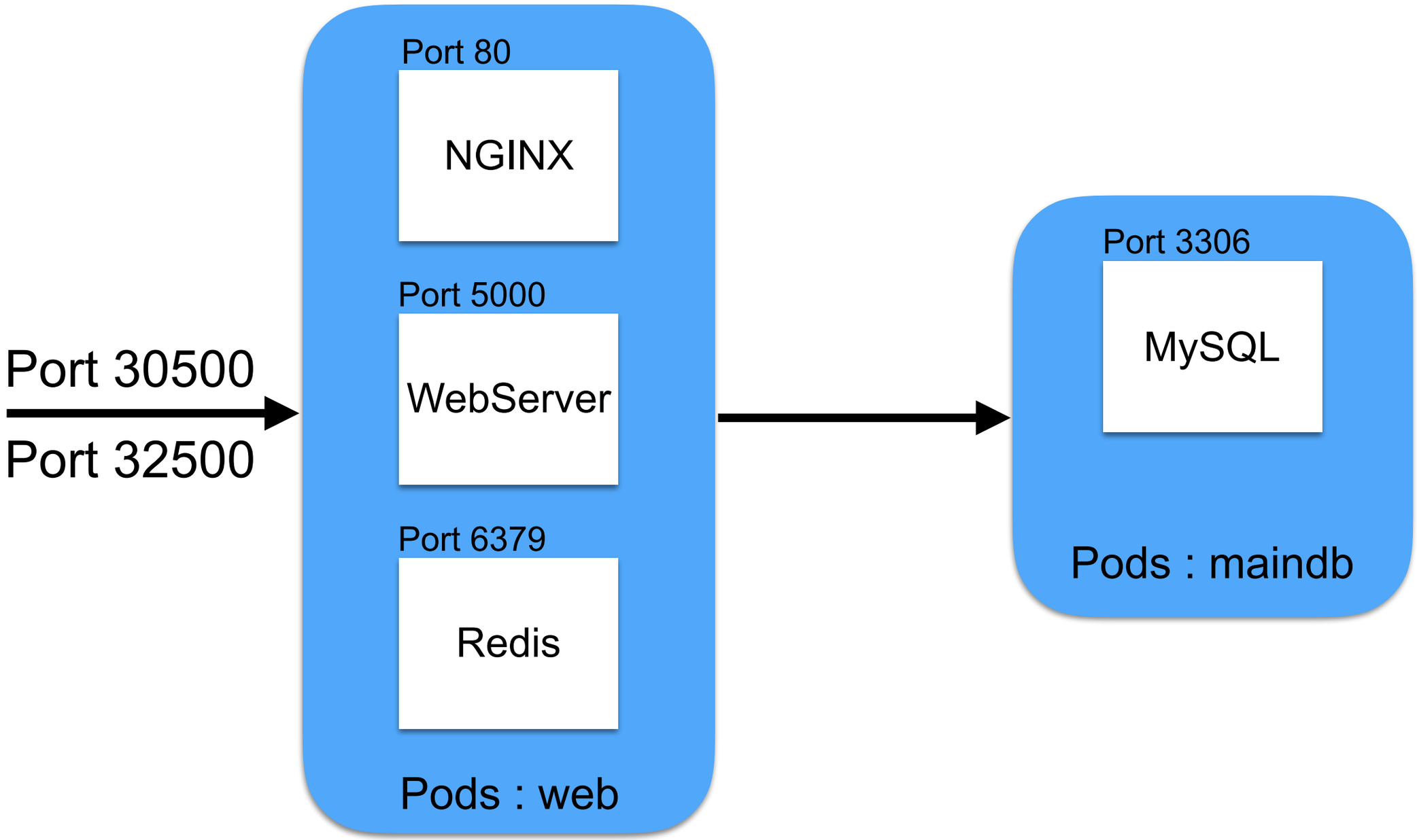
file /02-pod-service/single-pod/
workshop_instruction.txt

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

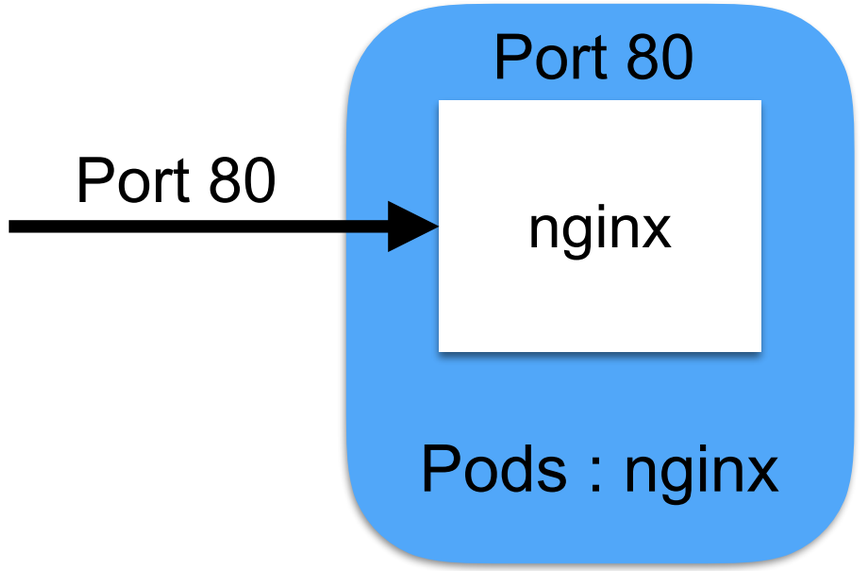
file /02-pod-service/expose/expose-pod-service.txt

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.

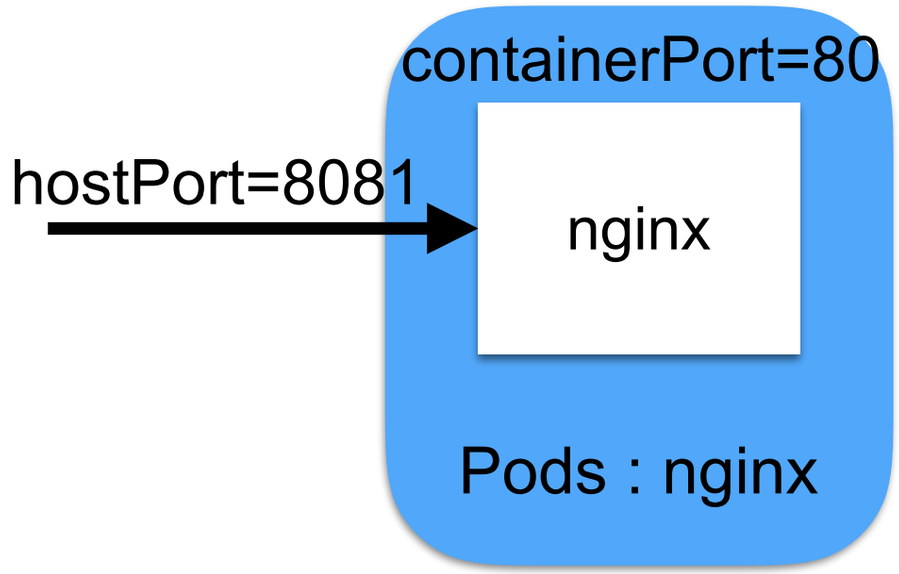


HostNetwork

```yaml
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.

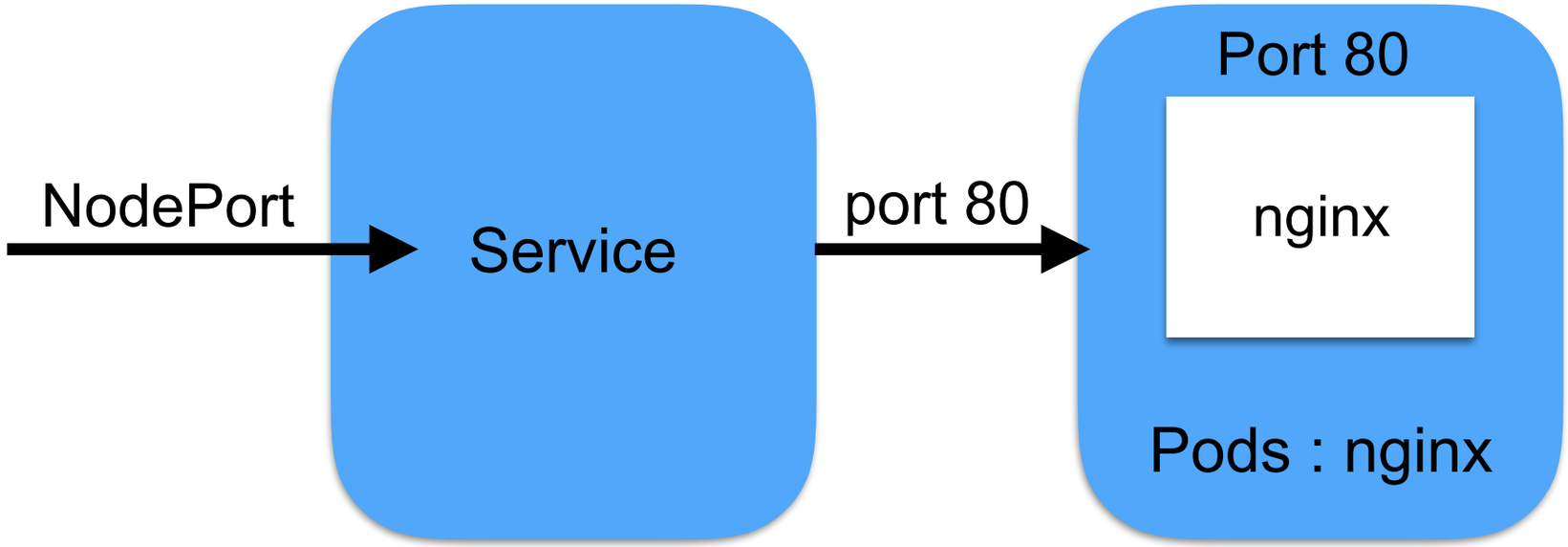


HostPort

```yaml
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.



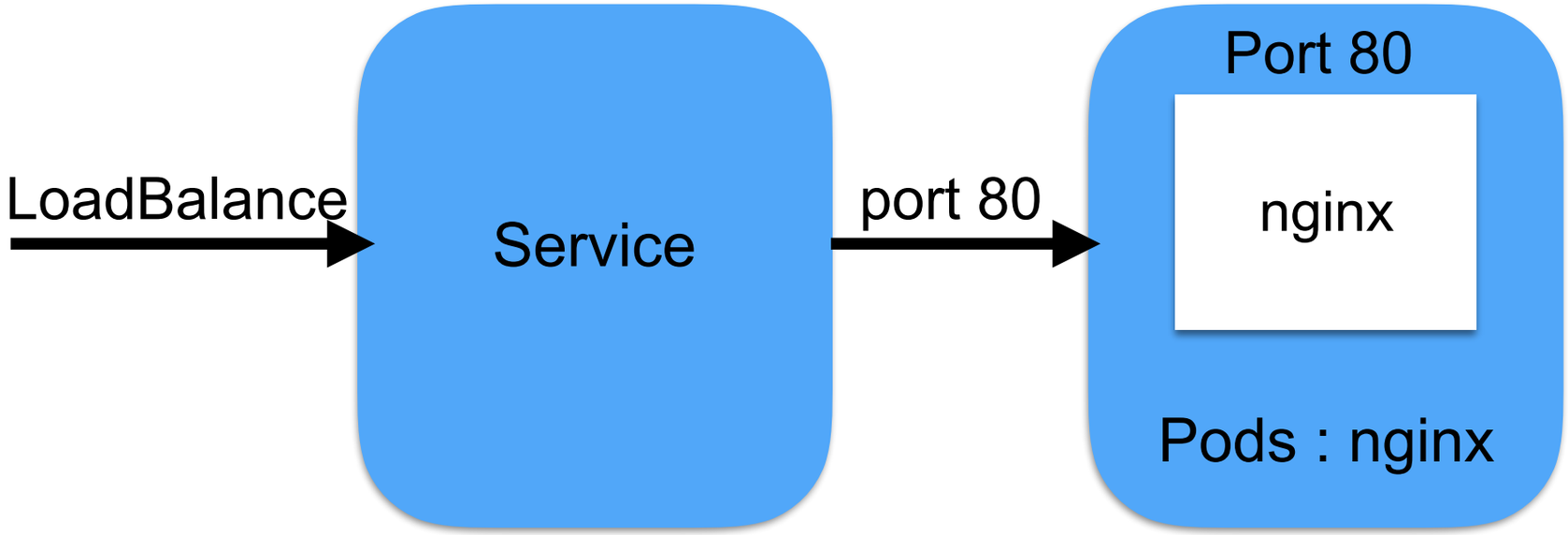
NodePort

```yaml
kind: Service
apiVersion: v1
metadata:
  name: nginx
spec:
  type: NodePort
  ports:
    - port: 80
      nodePort: 30001
  selector:
    name: nginx
```

```yaml
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.



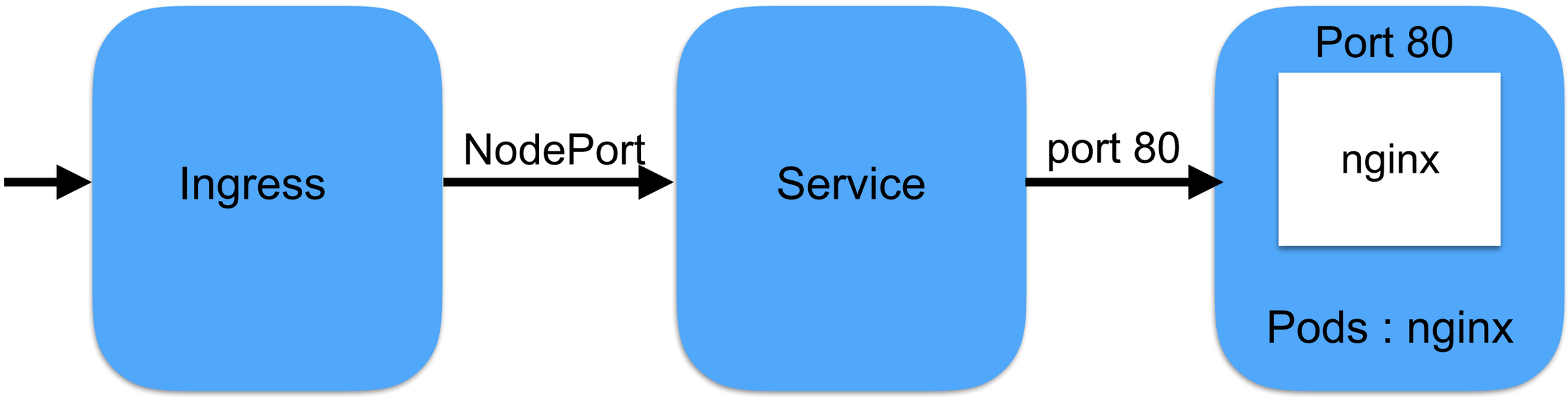
LoadBalancer

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

```yaml
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

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

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

Replication Controller (RC)

Replication Controller

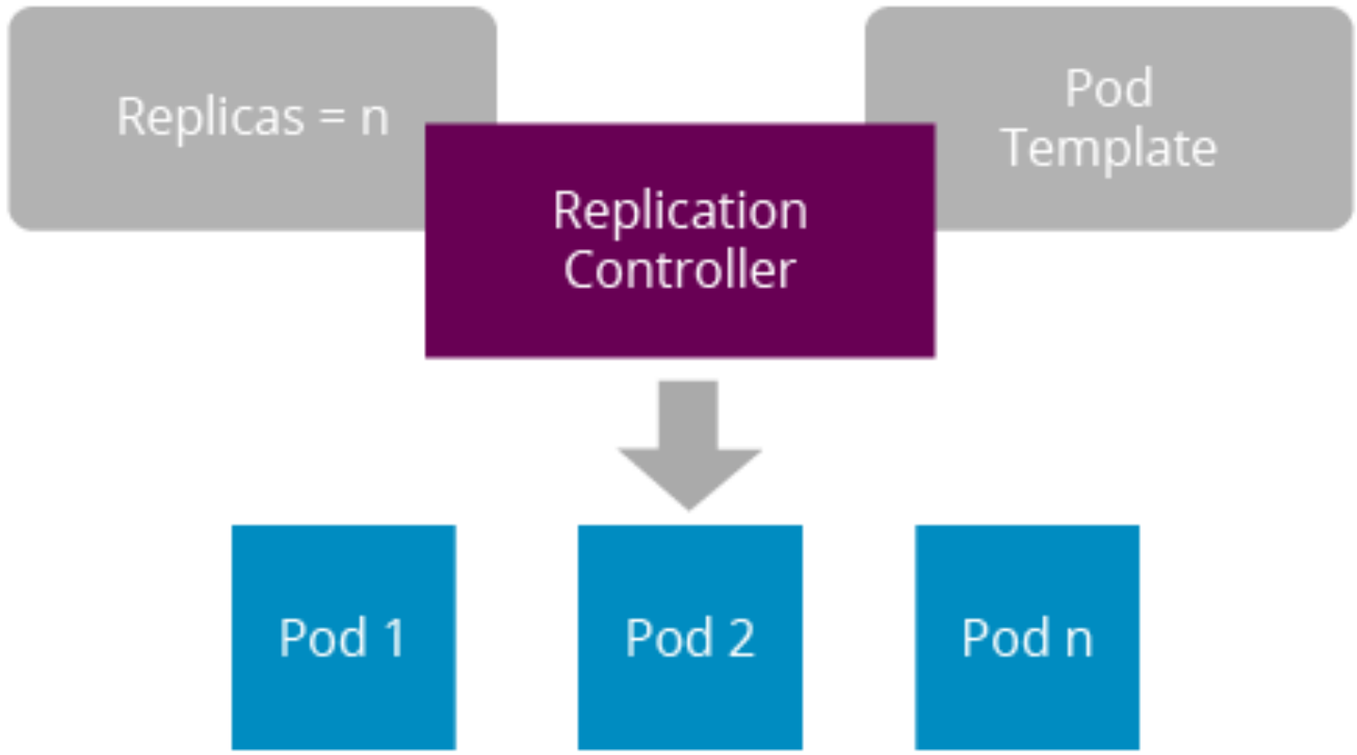
Create and maintain Pods
Keep copy of Pods by design
Maintain on cluster level
Auto-healing if Pods crash with any reason
Ensure Pods is up and run with desired number

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

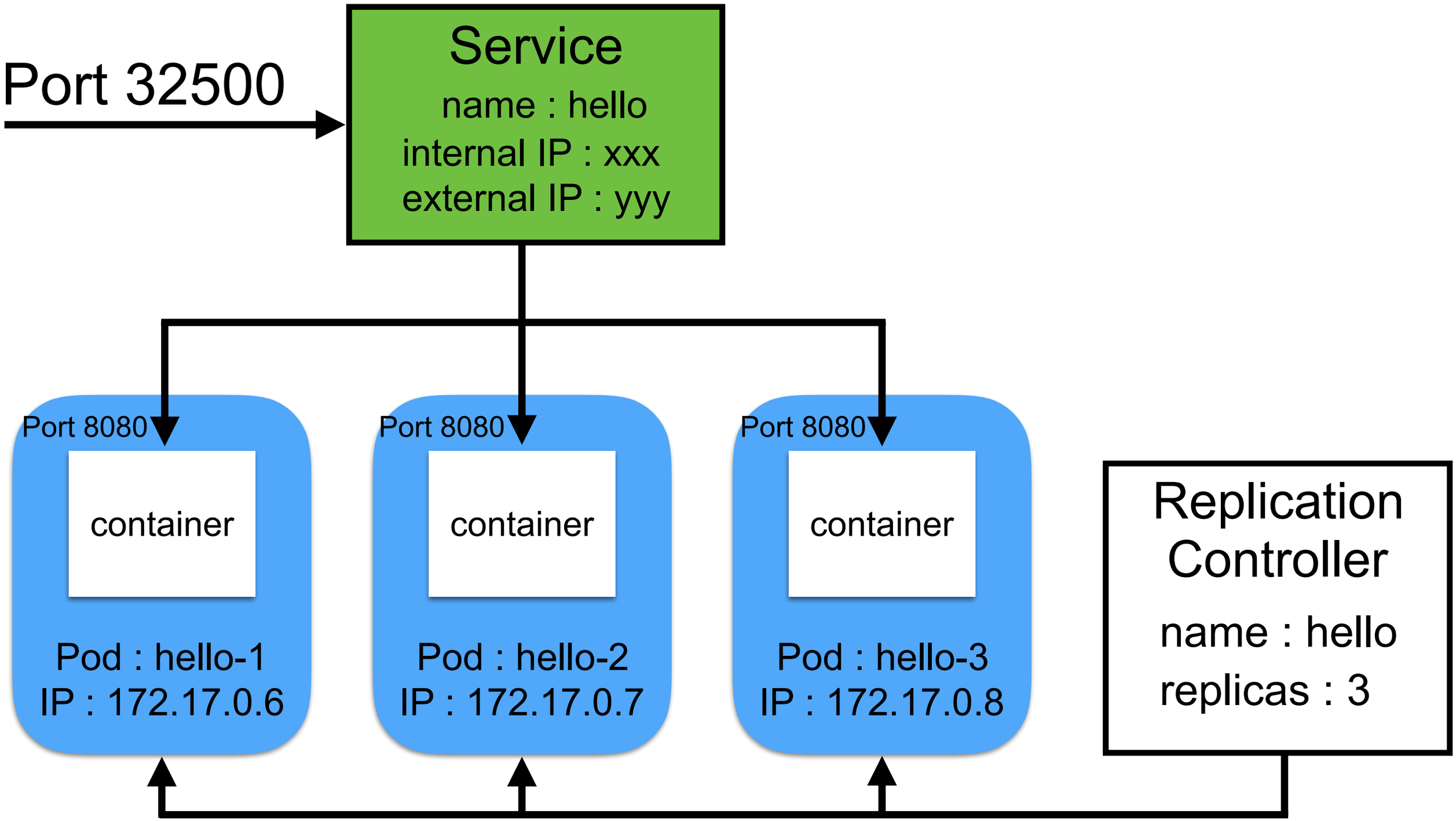
$kubectl **describe rc <name>**

```
  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

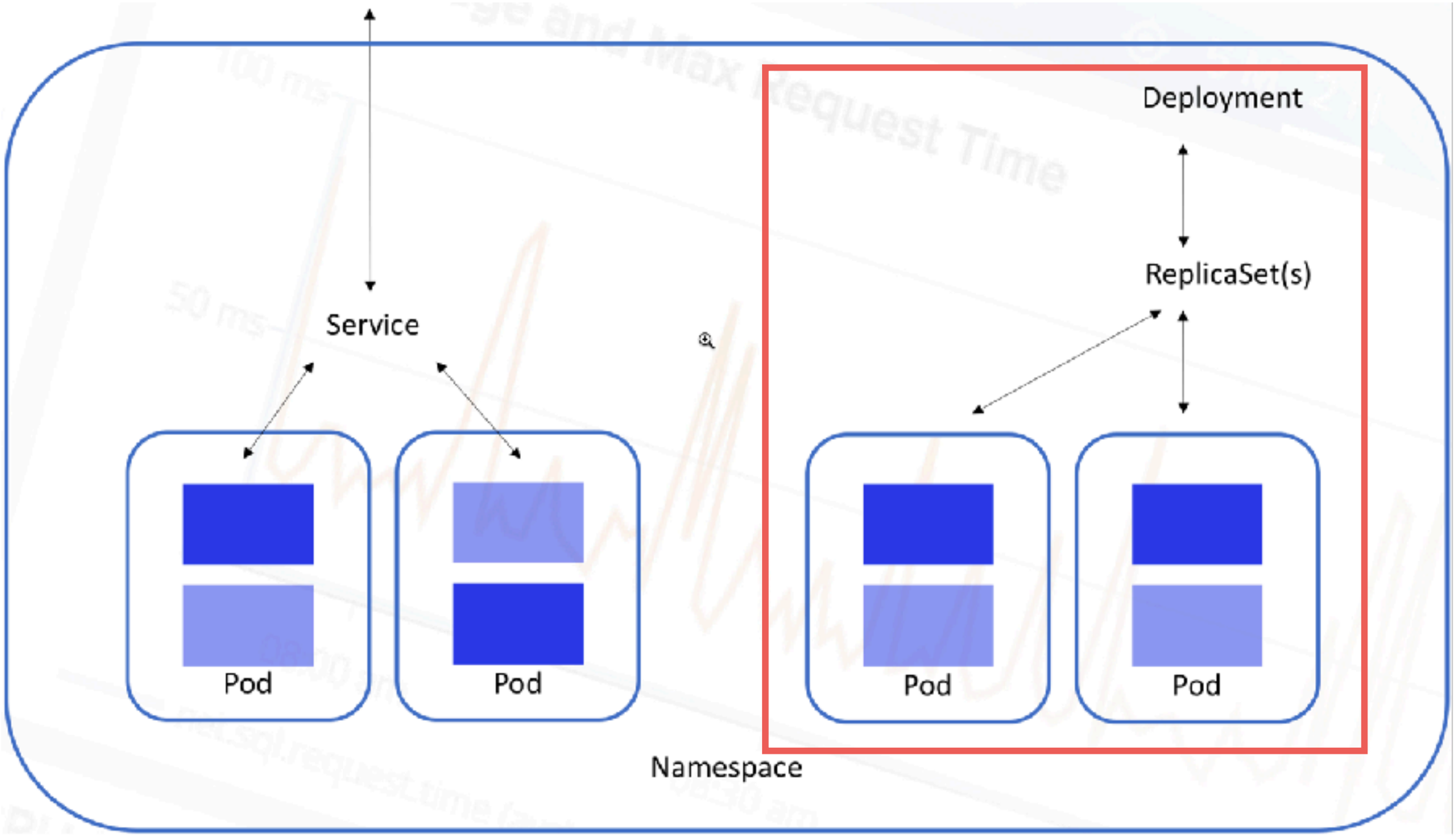
Scaling the application



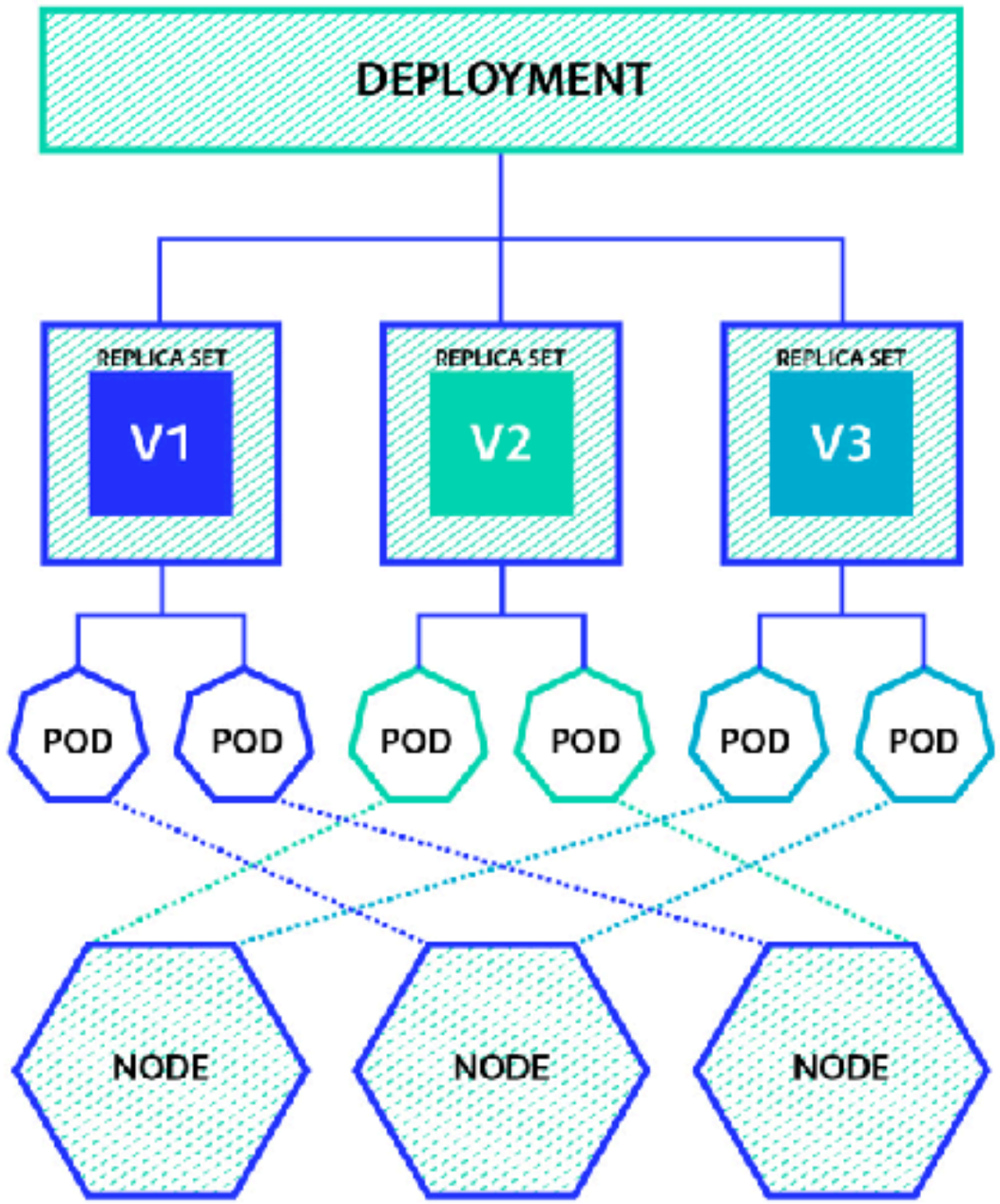
Deployment and ReplicaSet (RS)

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

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

Set-based requirement

```yaml
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 f
# reopened with the relevant failures.
#
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  annotations:
    deployment.kubernetes.io/revision: "6"
    kubectl.kubernetes.io/last-applied-configuration: |
      {"apiVersion":"apps/v1","kind":"Deployment","metadata":{"annotations":{},
","namespace":"default"},"spec":{"replicas":3,"revisionHistoryLimit":1,"selecto
mplate":{"metadata":{"labels":{"app":"web"}},"spec":{"containers":[{"image":"sc
s":[{"containerPort":8080,"protocol":"TCP"}]}]}}}}
  creationTimestamp: 2018-04-01T17:17:51Z
  generation: 10
  labels:
    app: web
  name: hello
  namespace: default
  resourceVersion: "33405"
  selfLink: /apis/extensions/v1beta1/namespaces/default/deployments/hello
  uid: 9b773859-35d0-11e8-9d36-0800275c6c60
```

Modify YAML file and apply

$kubectl apply -f hello_deployment.yml

Show history of rollout

$kubectl rollout history deployment/hello

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

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

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

Try to rollback to revision

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

Try to scale a deployment

$kubectl scale deployment hello --replicas=5

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

Pause and resume rollout

$kubectl **rollout pause** deployment/hello

$kubectl **rollout resume** deployment/hello

Workshop
Deployment/ReplicaSet

file /04-deployment-and-replica-set

Horizontal Pods Autoscale (HPA)

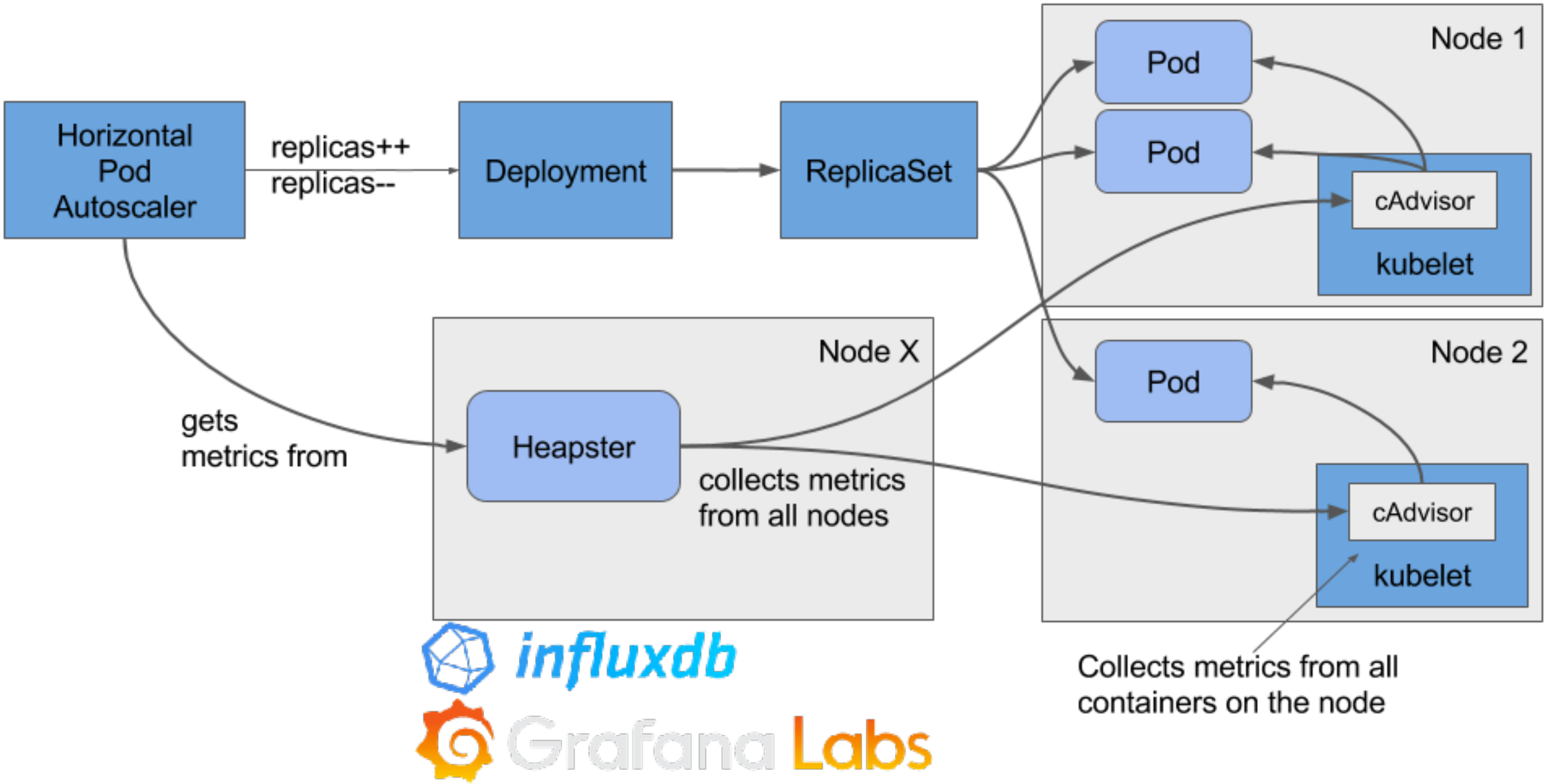
File /working-with-java/03-hpa

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-54cccfbdf8-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          4h
kube-system   storage-provisioner                         1/1       Running   2          1d
```

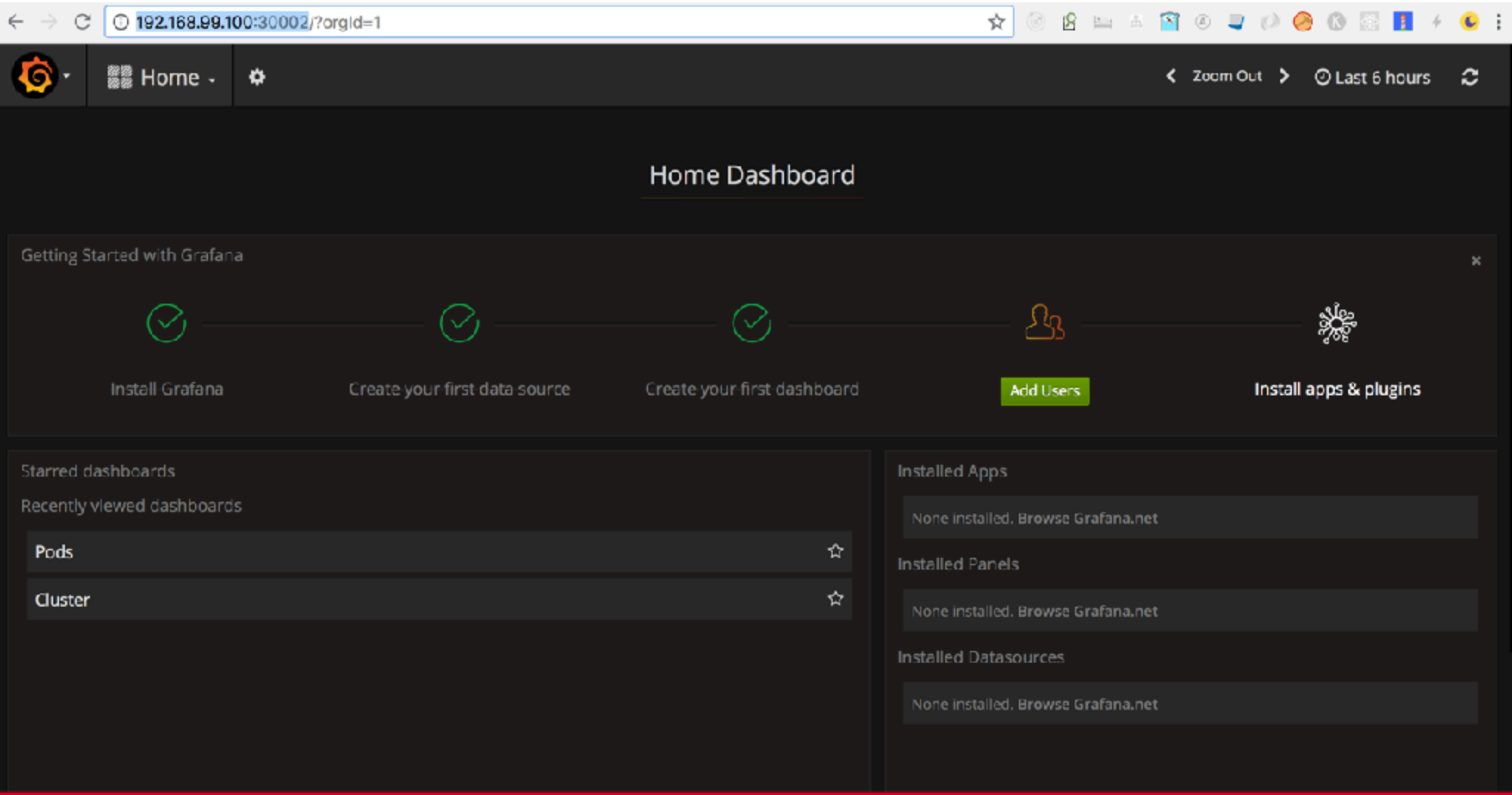
See all services from heapster

$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	10.102.224.179	<none>	80:30001/TCP	4h
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



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	TARGETS	MINPODS	MAXPODS
Deployment/spring-boot-service-deployment	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

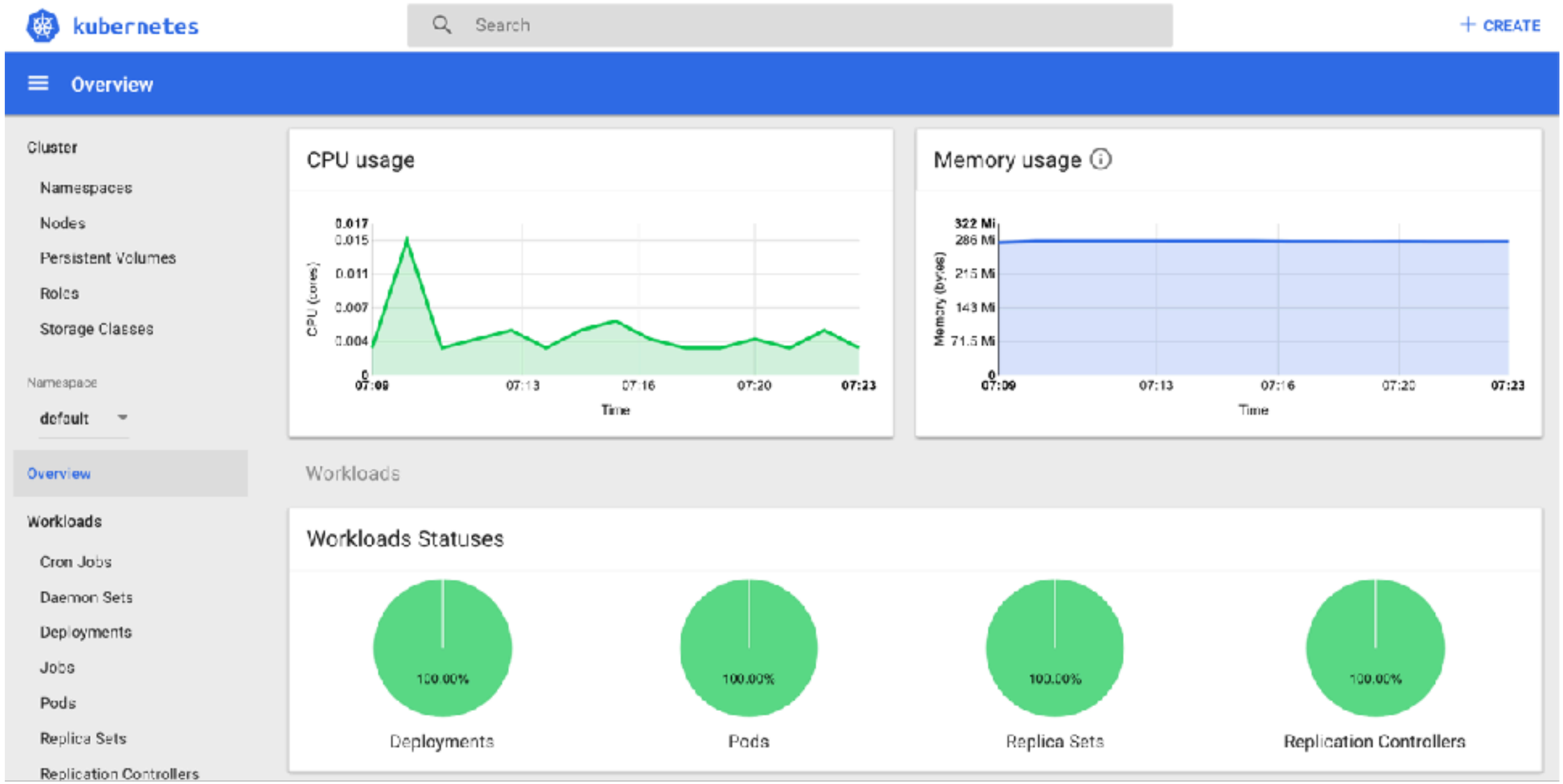
```
NAME                                          READY     STATUS    RESTARTS   AGE
load-test-6f86656986-9qvdl                    0/1       Error     0          4m
mongo-controller-xdh28                        1/1       Running   0          1h
spring-boot-service-deployment-86c7b47b7b-9l9sn   1/1       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            Message
  ----            ------  ------            -------
  AbleToScale     False   BackoffBoth       the time since the previous scale is still within both the downscale and upscale fo
rbidden windows
  ScalingActive   True    ValidMetricFound  the HPA was able to succesfully calculate a replica count from cpu resource utiliza
tion (percentage of request)
  ScalingLimited  True    TooManyReplicas   the desired replica count is more than the maximum replica count
Events:
  Type     Reason                        Age   From                       Message
  ----     ------                        ----  ----                       -------
  Normal   SuccessfulRescale             1m    horizontal-pod-autoscaler  New size: 4; reason: cpu resource utilization (percen
tage of request) above target
  Warning  FailedGetResourceMetric       19s   horizontal-pod-autoscaler  unable to get metrics for resource cpu: unable to fet
ch metrics from API: the server could not find the requested resource (get pods.metrics.k8s.io)
  Warning  FailedUpdateStatus            19s   horizontal-pod-autoscaler  Operation cannot be fulfilled on horizontalpodautosca
lers.autoscaling "spring-boot-service-deployment": the object has been modified; please apply your changes to the latest versio
n and try again
  Warning  FailedComputeMetricsReplicas  19s   horizontal-pod-autoscaler  failed to get cpu utilization: unable to get metrics
for resource cpu: unable to fetch metrics from API: the server could not find the requested resource (get pods.metrics.k8s.io)
```

7. See result in dashboard (1)



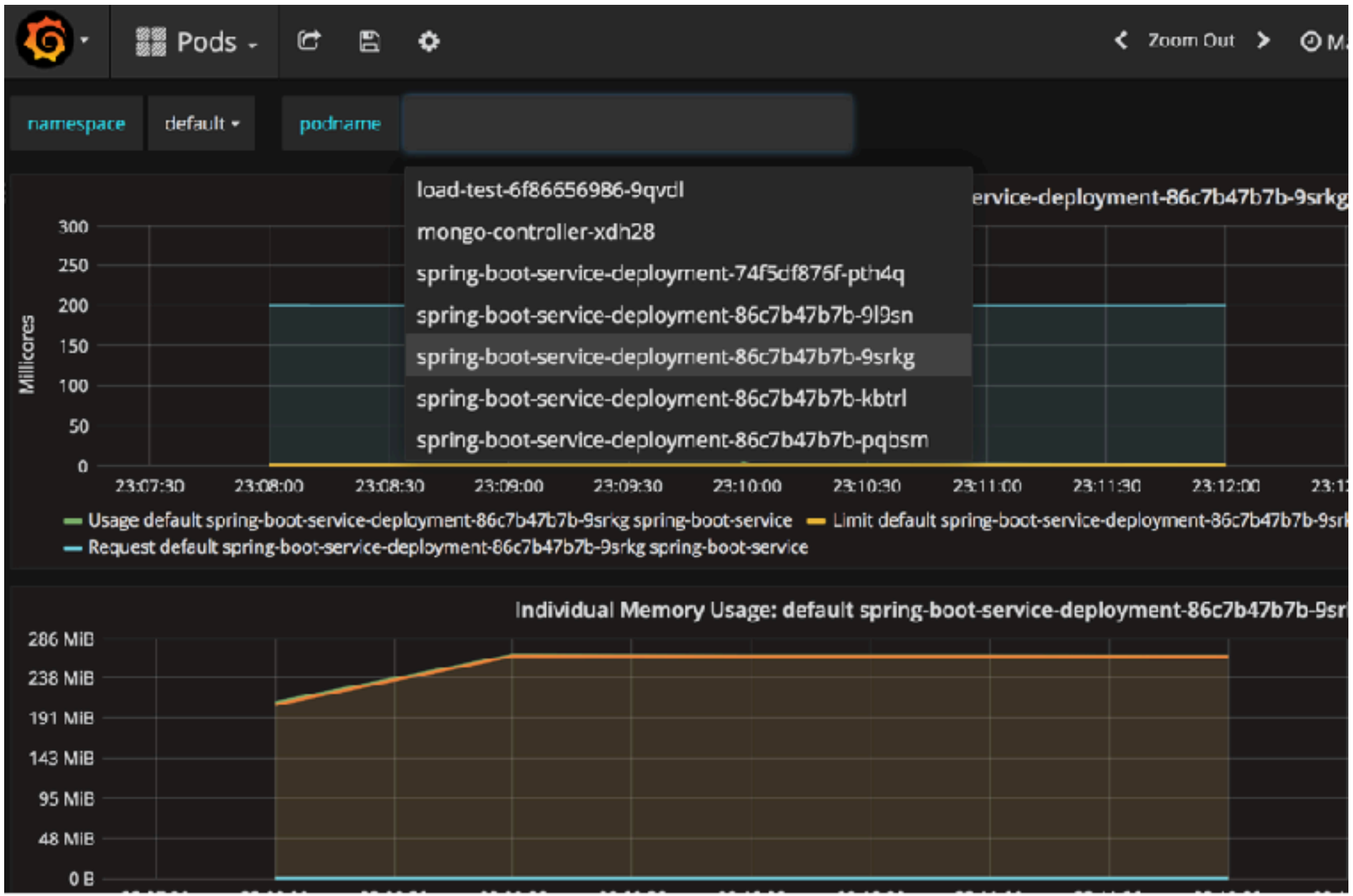
7. See result in dashboard (2)

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

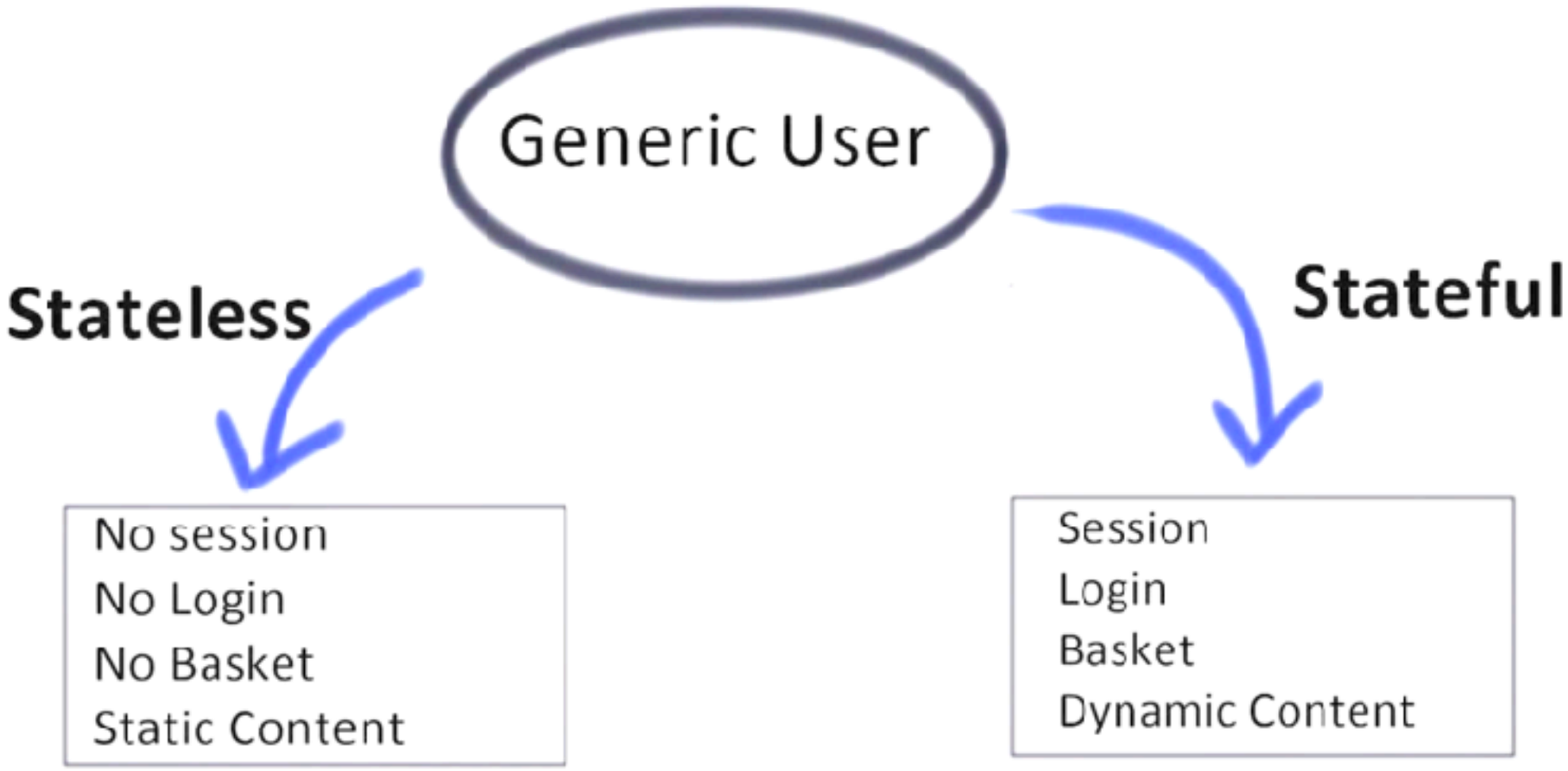
Stateless

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

Stateful

State information is keep 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

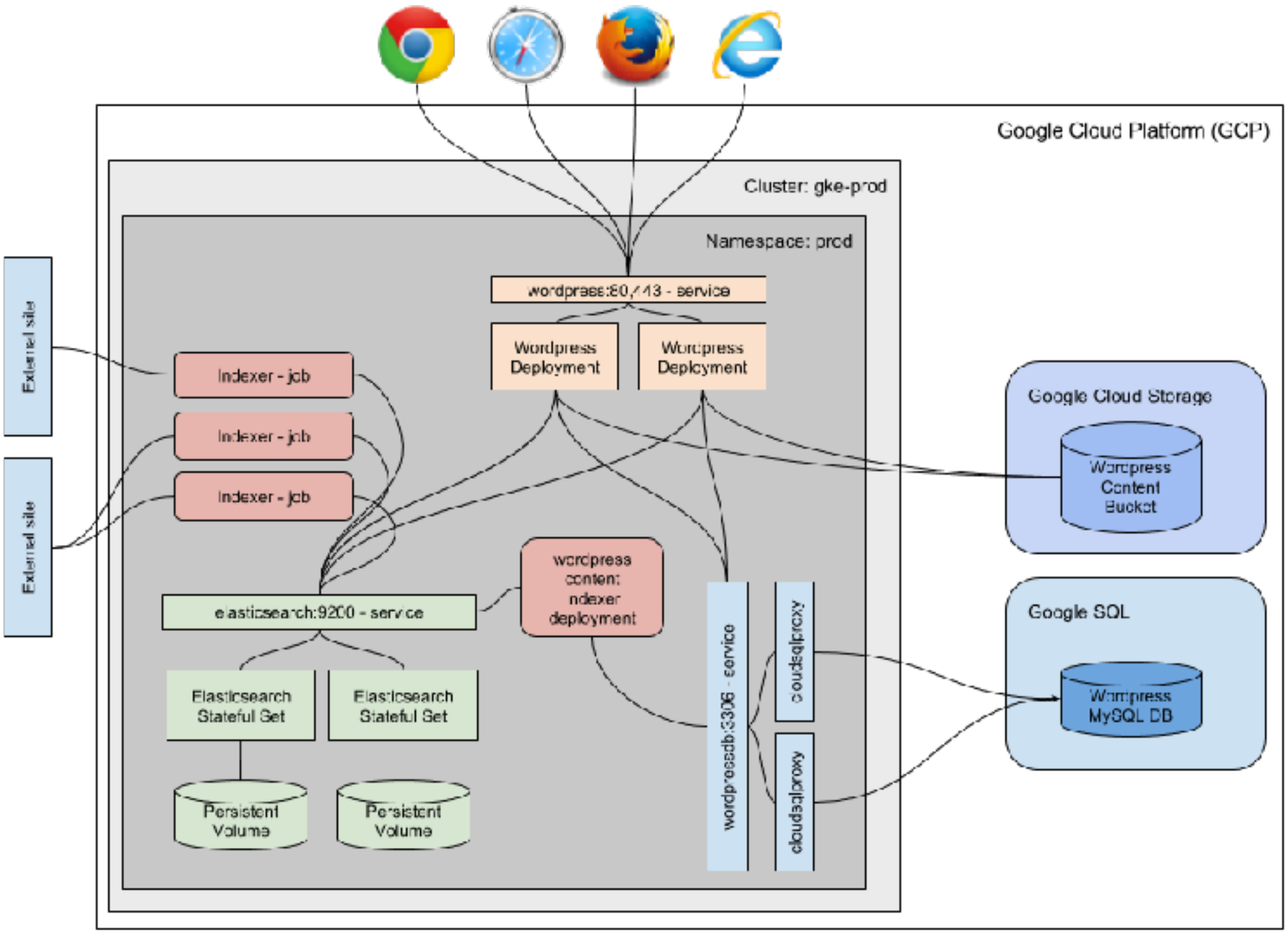
Database server

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

Stateful with Kubernetes

Volume
Statefulset

Stateful with Kubernetes

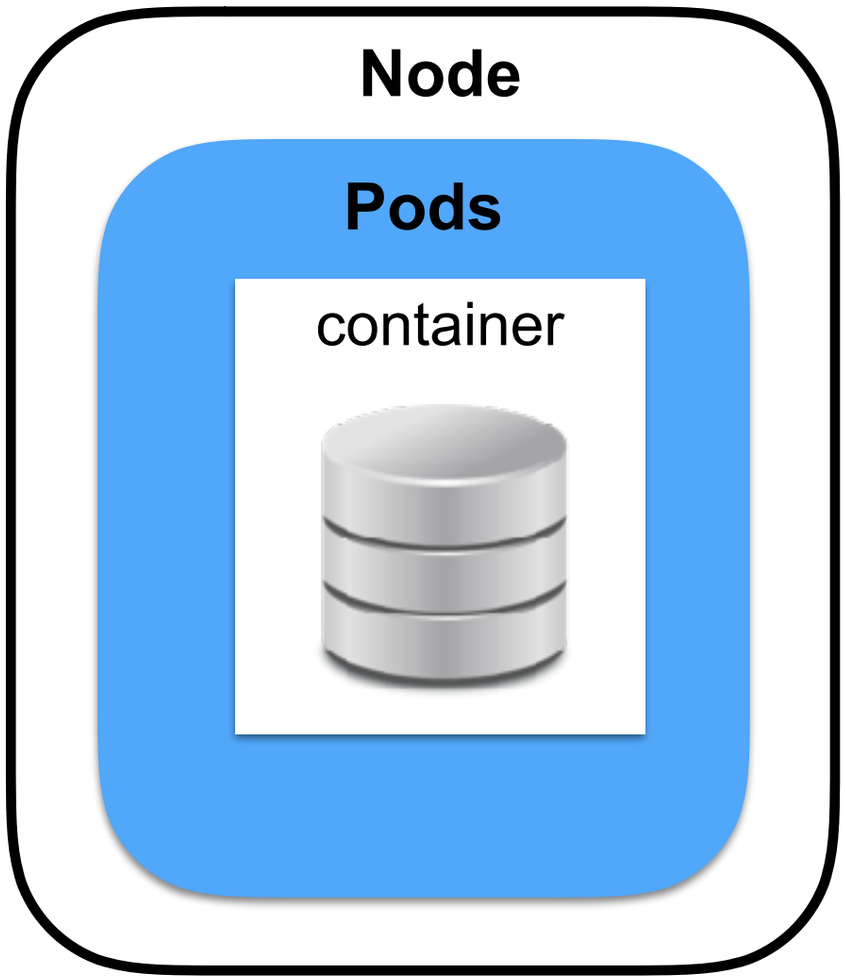


http://blogs.avalonconsult.com/blog/enterprise-web/cloud-computing/architecting-for-kubernetes/

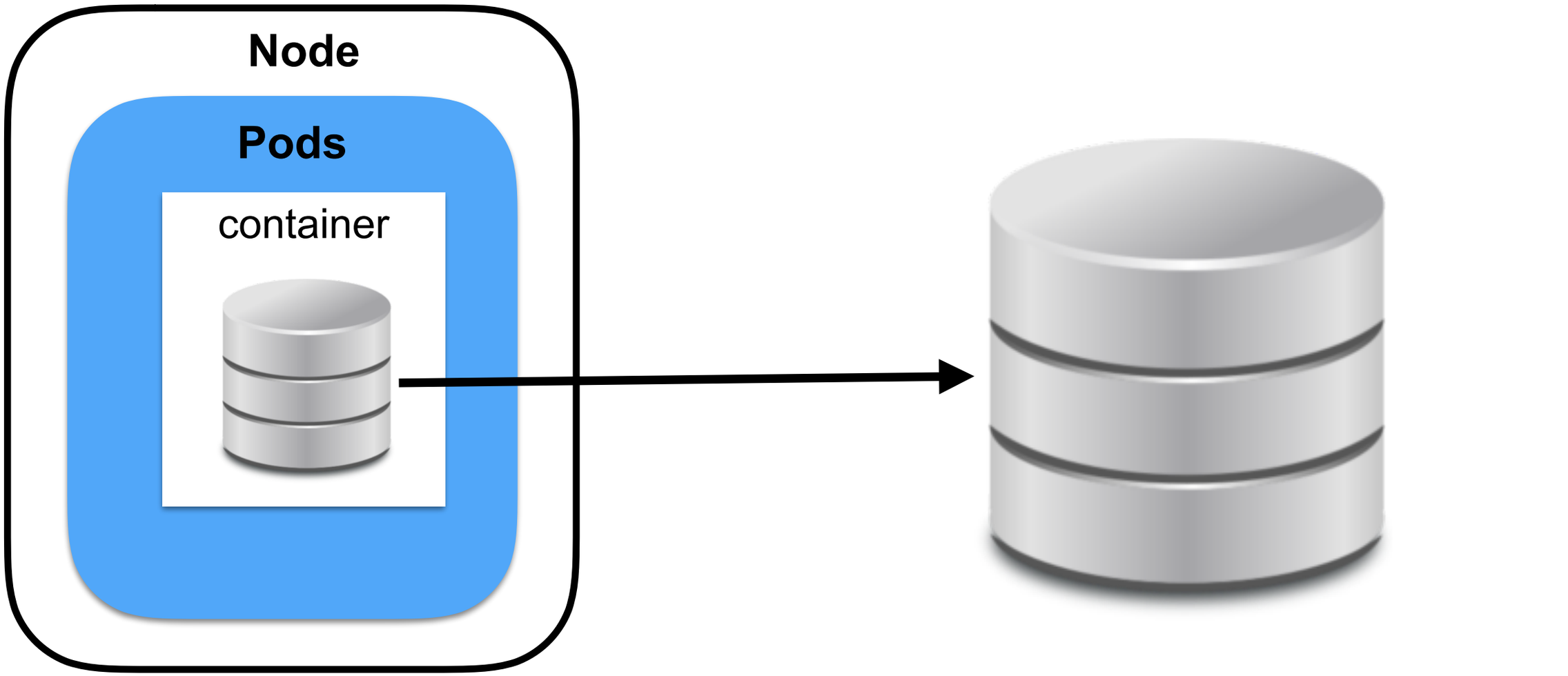
Volume in Kubernetes

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

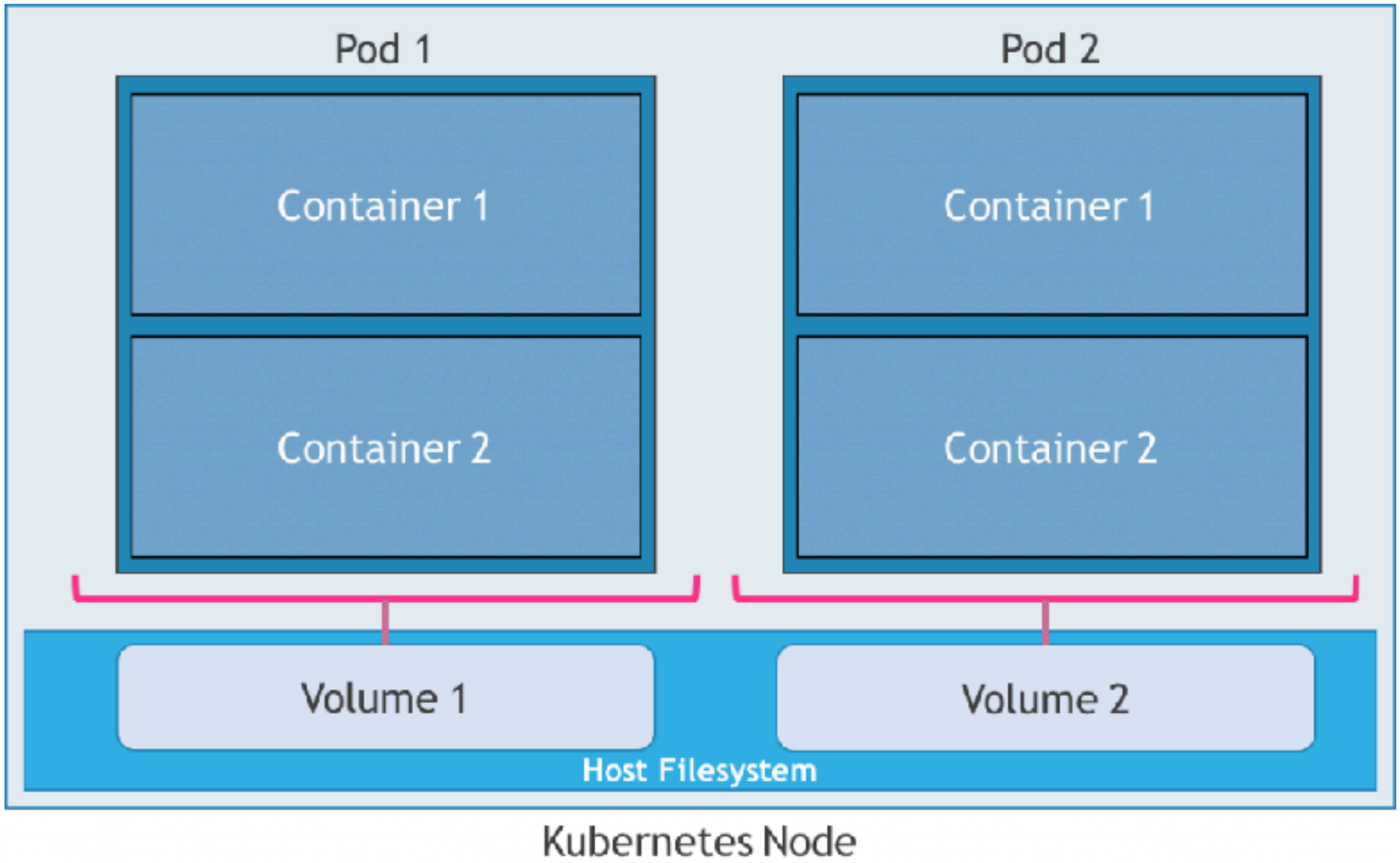
Volume



Volume

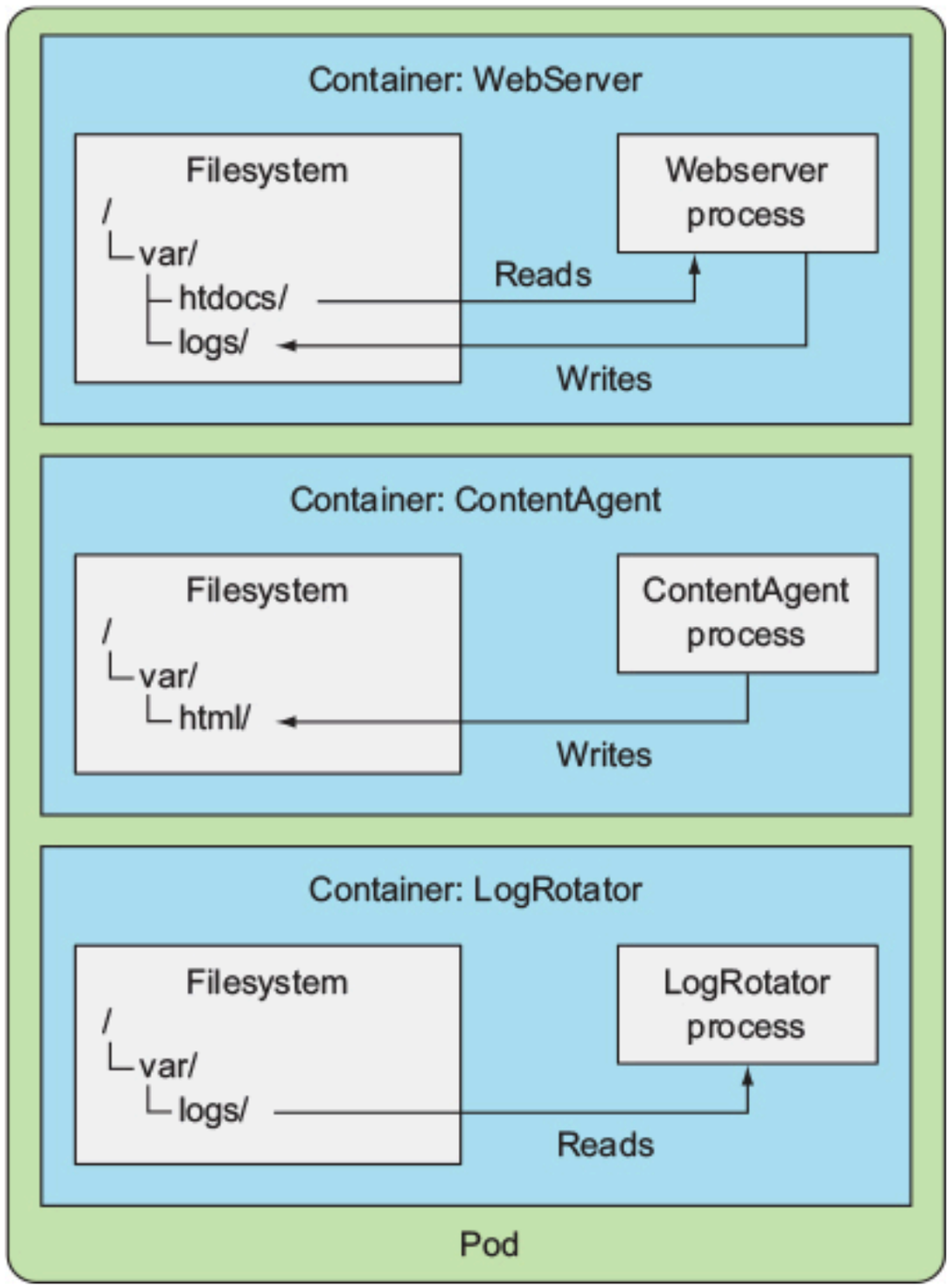


Containers shared volume

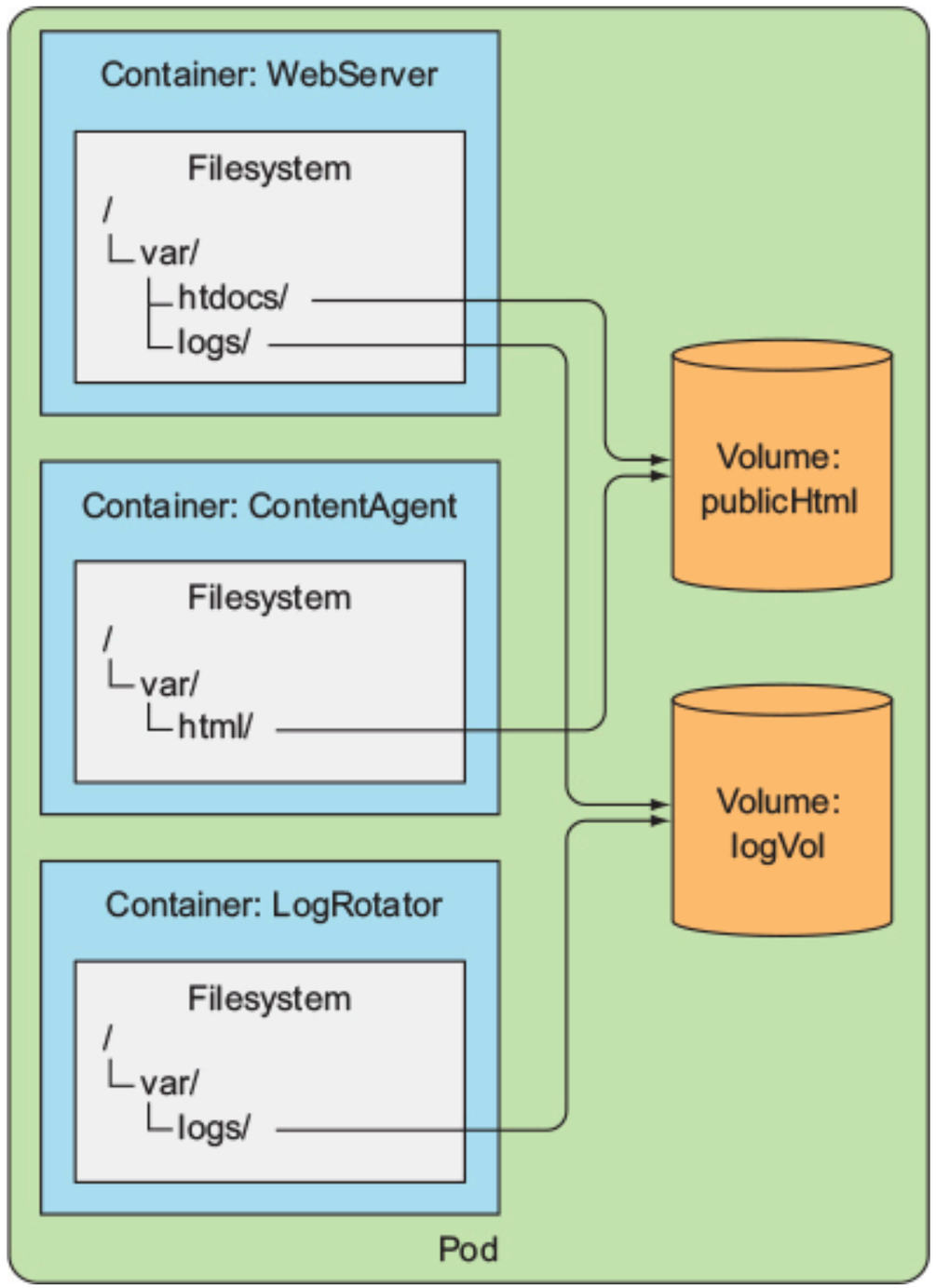


https://thenewstack.io/strategies-running-stateful-applications-kubernetes-volumes/

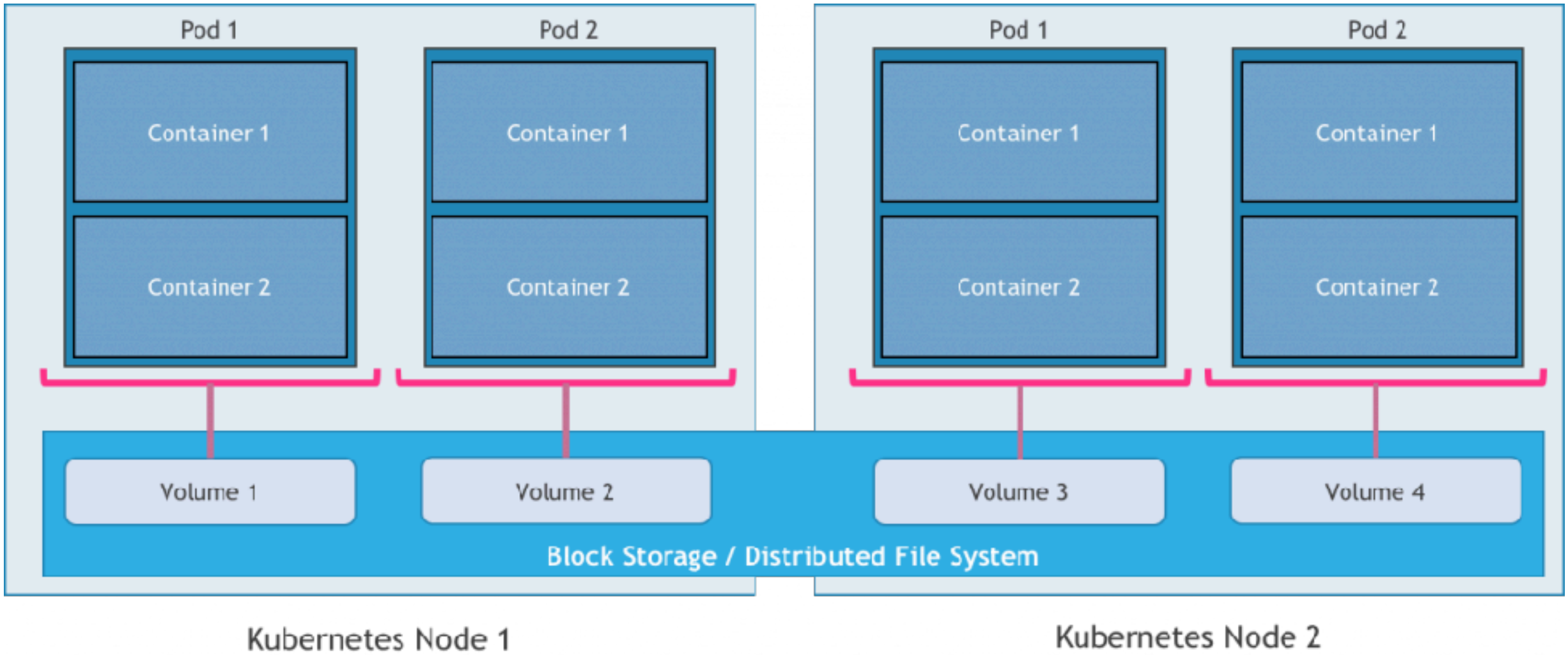
Without shared storage



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

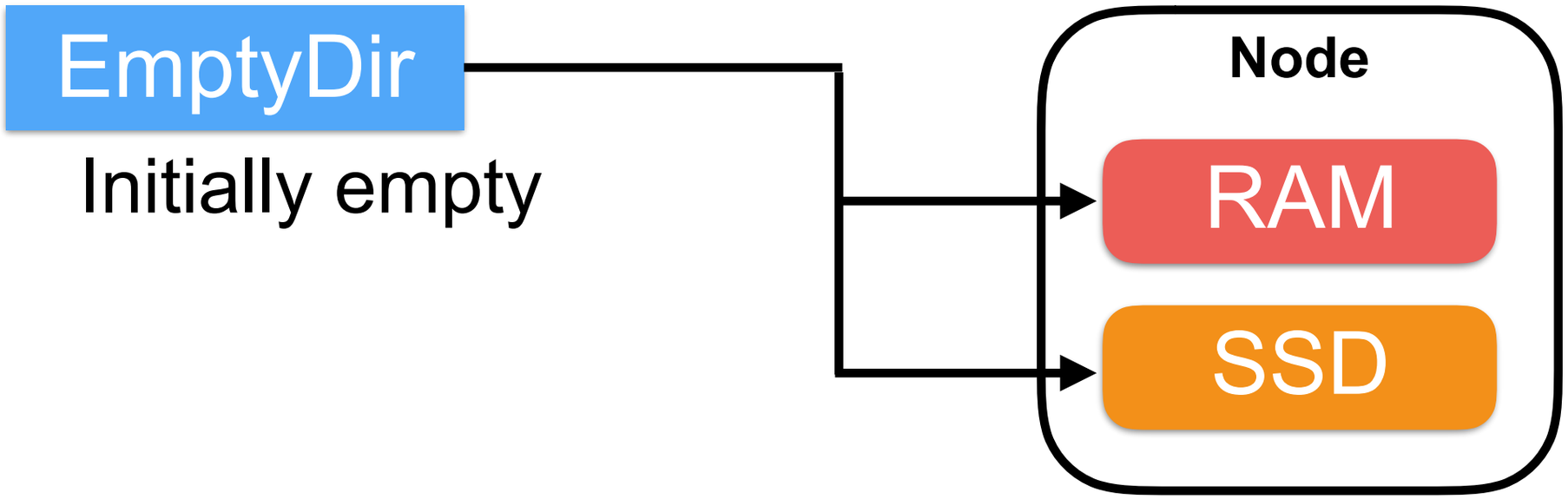
EmptyDir
hostPath
gitRepo
NFS
icePersistentDisk
Flocker
Persistent Volume Claim (PVC)

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

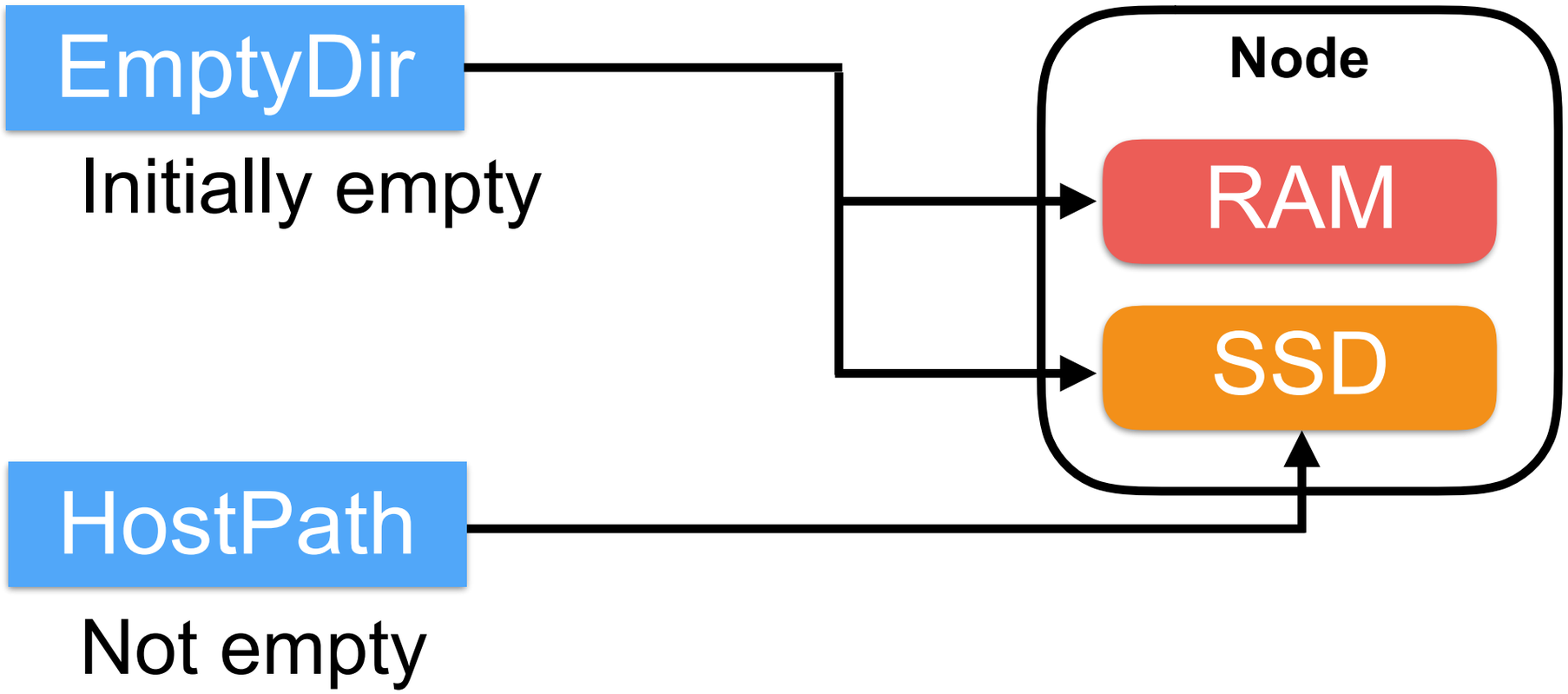
Volume on Kubernetes

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

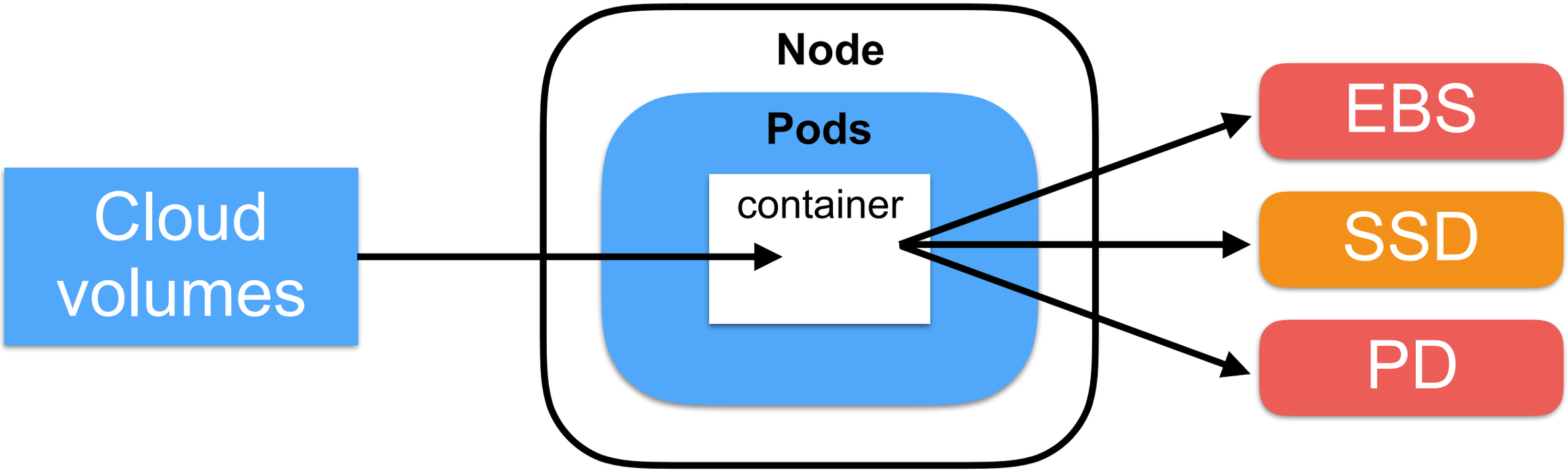
Volume on Kubernetes



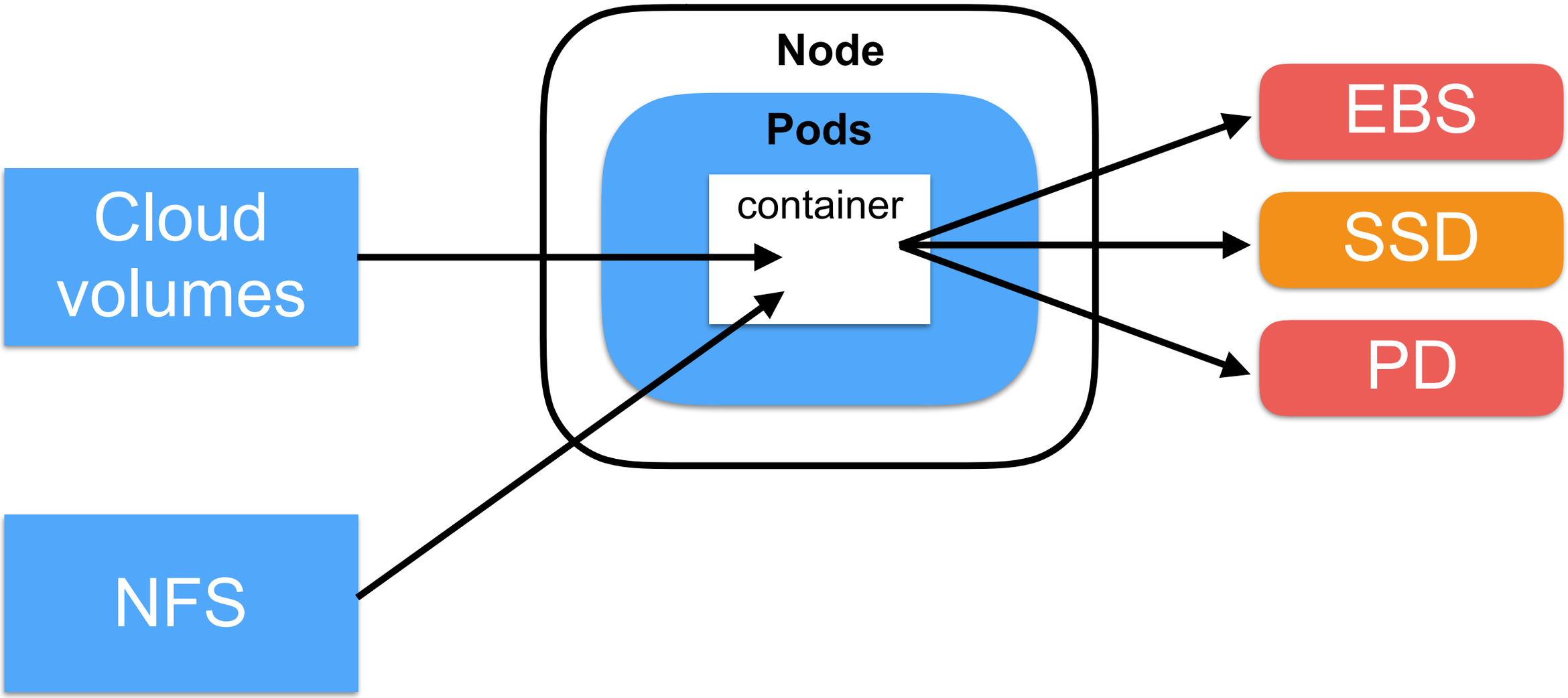
Volume on Kubernetes



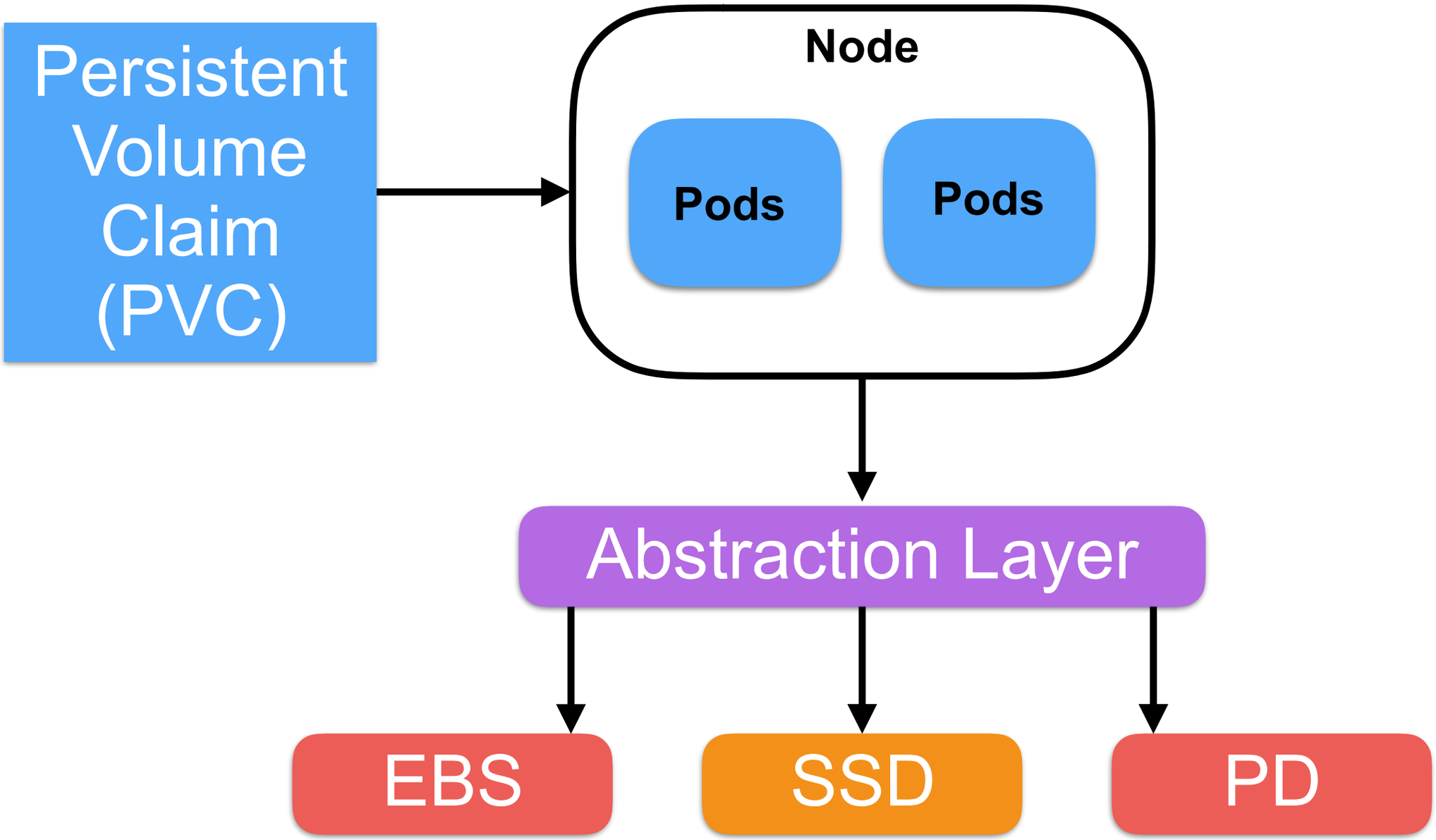
Volume on Kubernetes



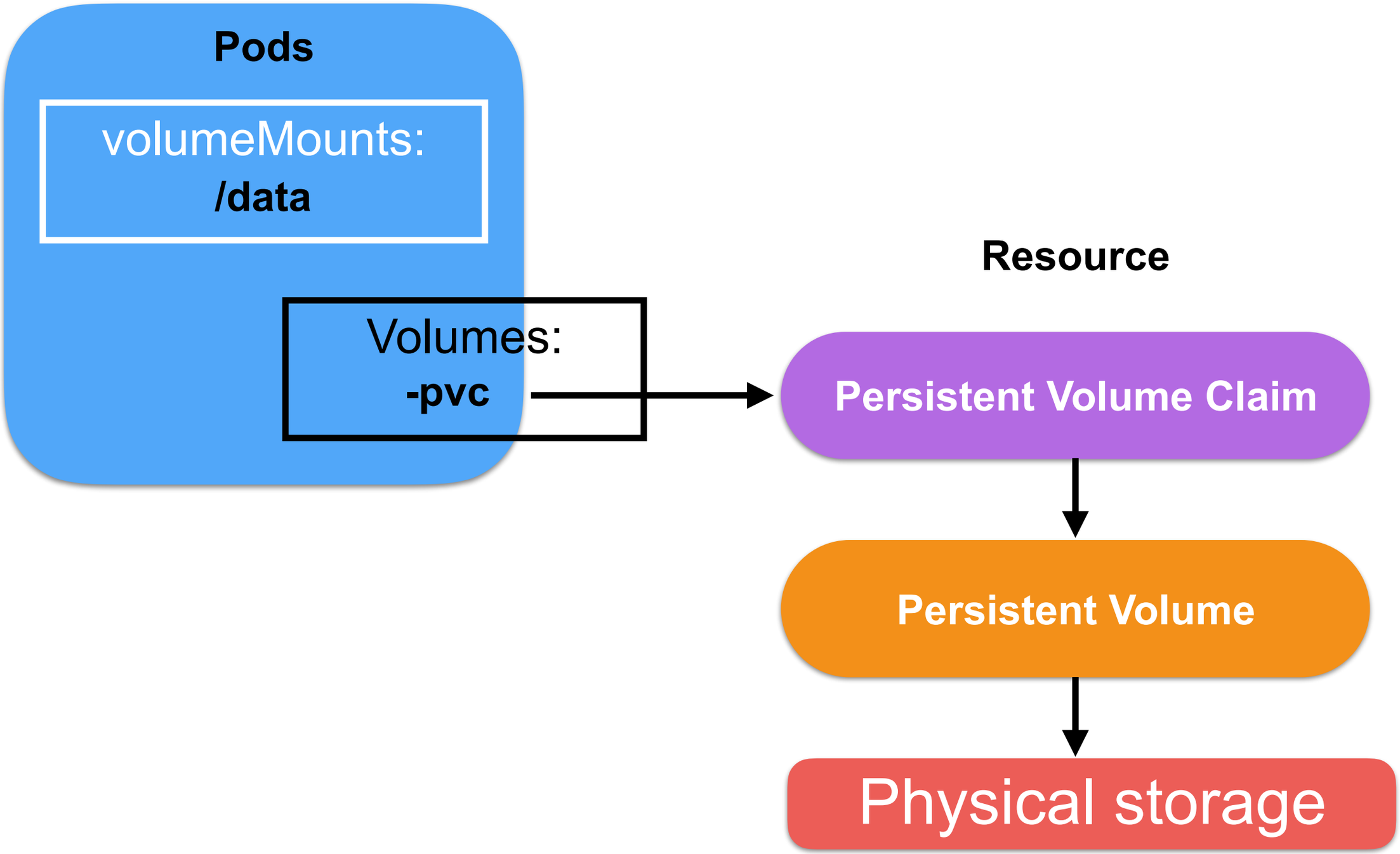
Volume on Kubernetes



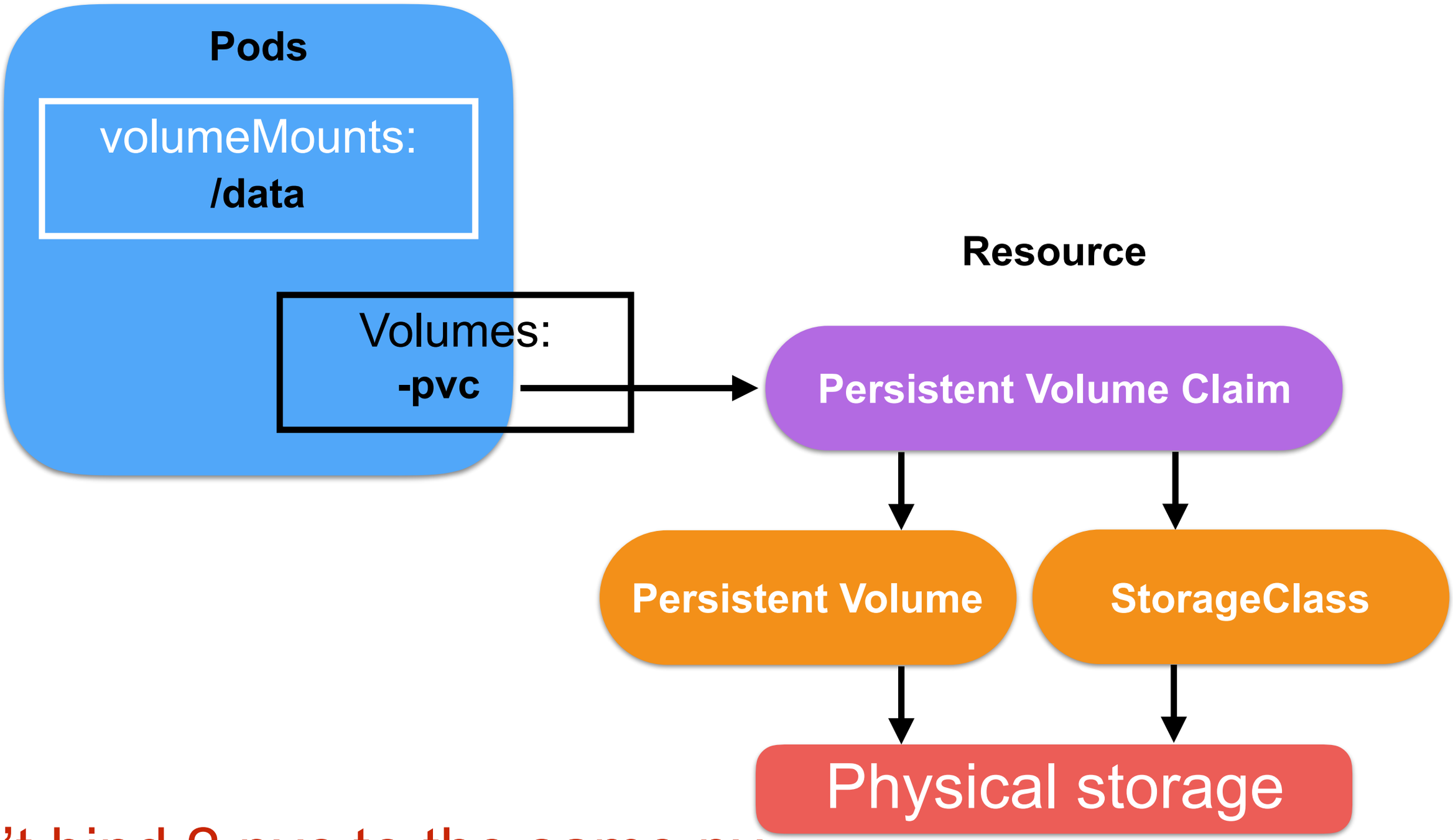
Volume on Kubernetes



Volume on Kubernetes



Volume on Kubernetes



Can’t bind 2 pvc to the same pv.

Persistent Volume

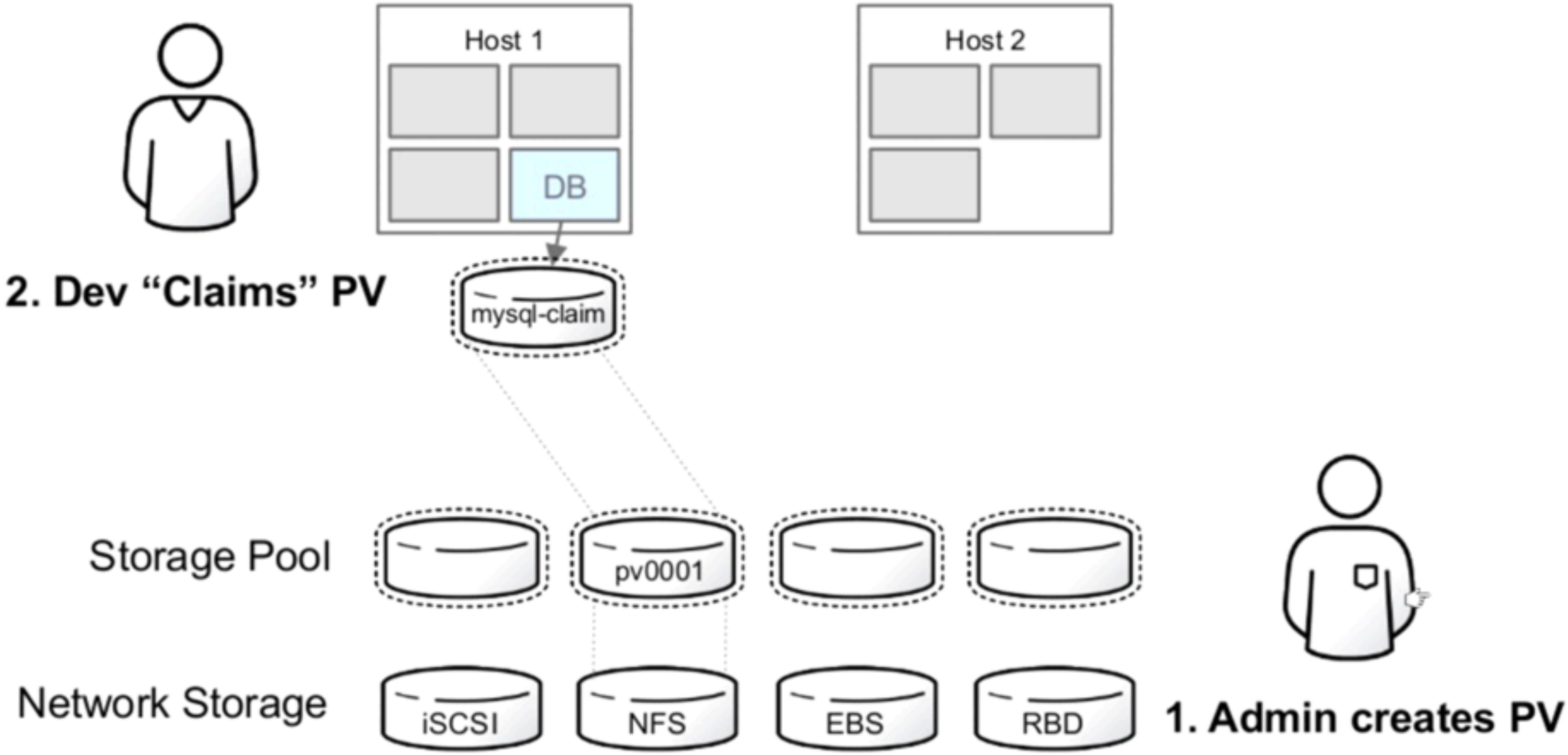
Provide and APIs for user and administrator
Abstraction layer to hide detail of storage

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)

```yaml
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)

```yaml
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

Read**W**rite**O**nce (RWO)

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

Read**O**nly**M**any (ROX)

the volume can be mounted read-only by many nodes

Read**W**rite**M**any (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

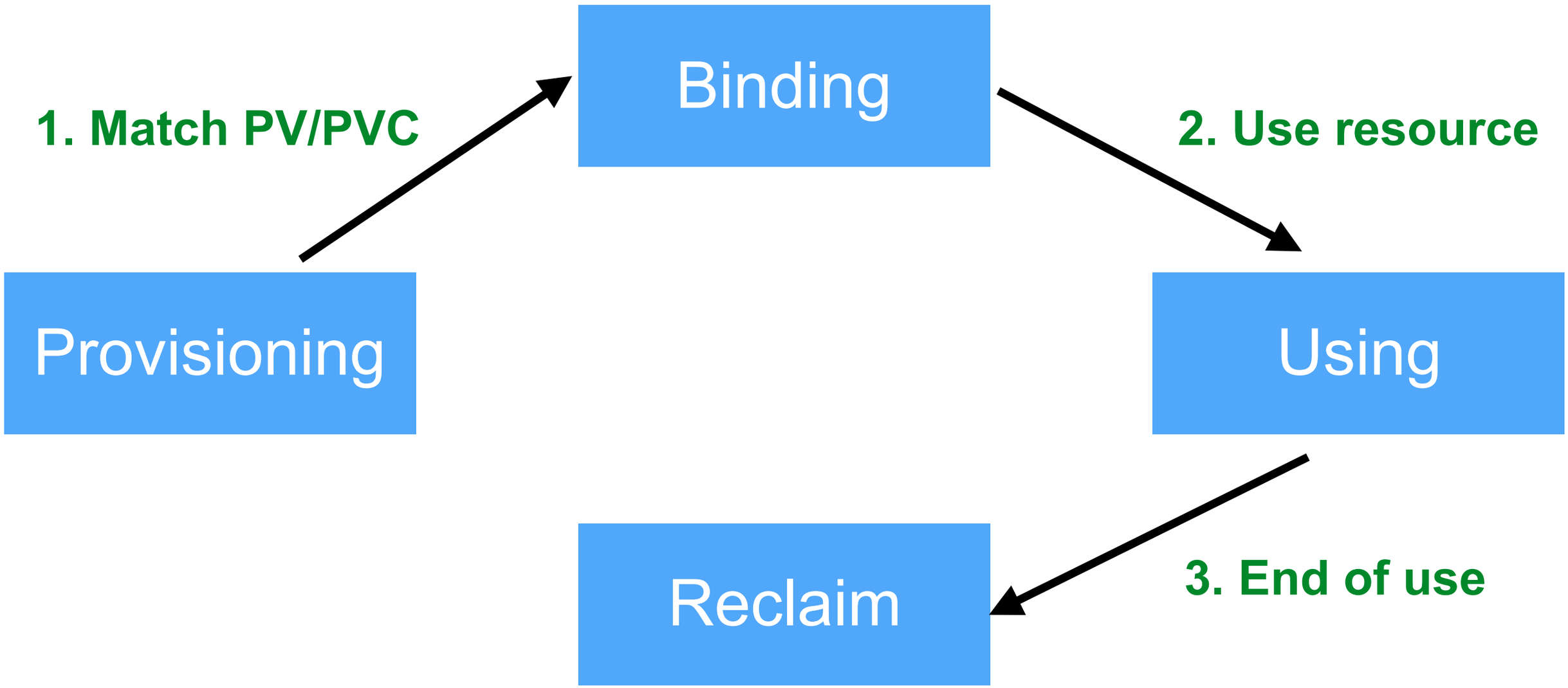
3. Released

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

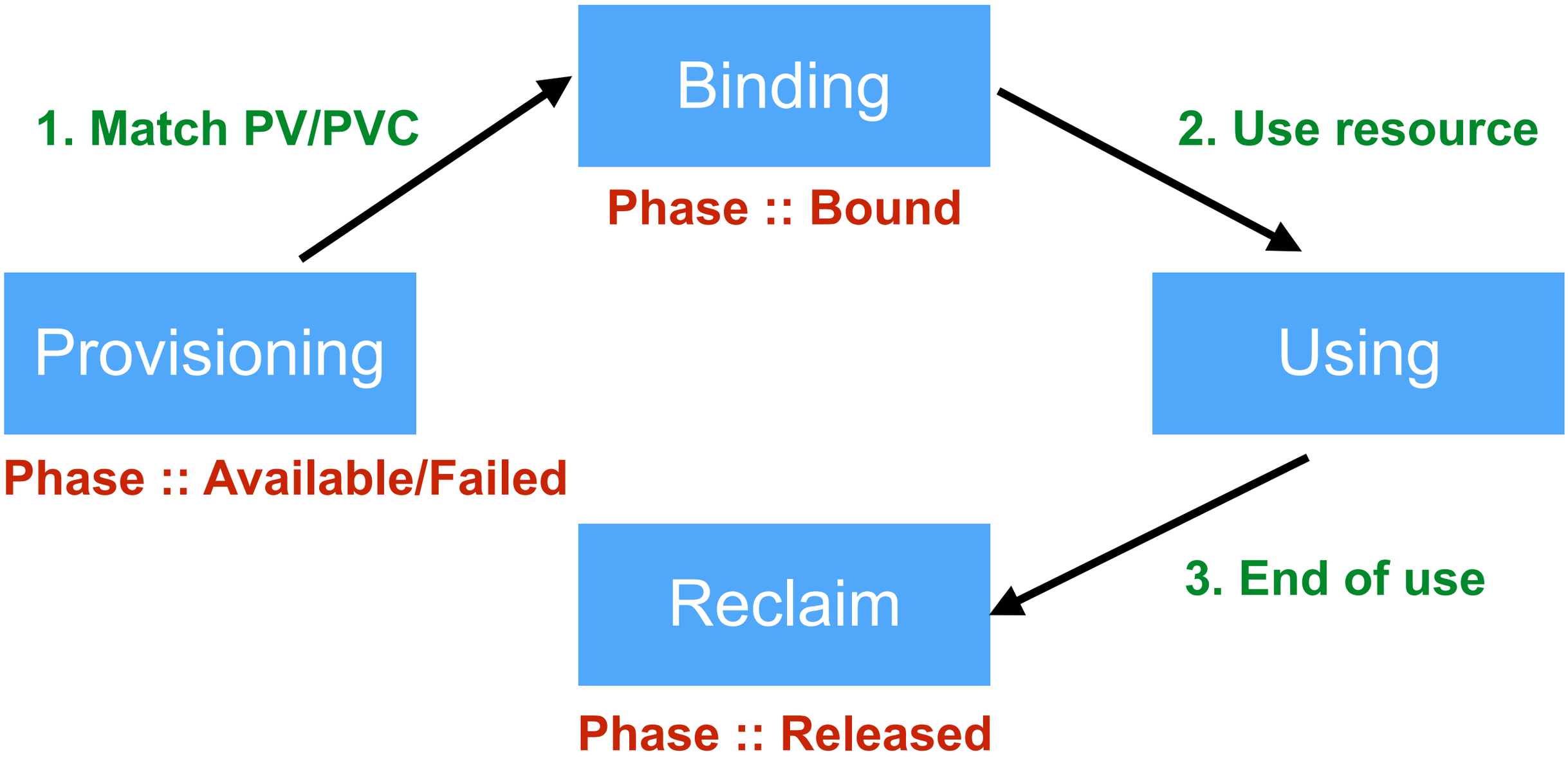
4. Failed

Volume has failed its automatic reclamation

Lifecycle of volume



Phase of volume

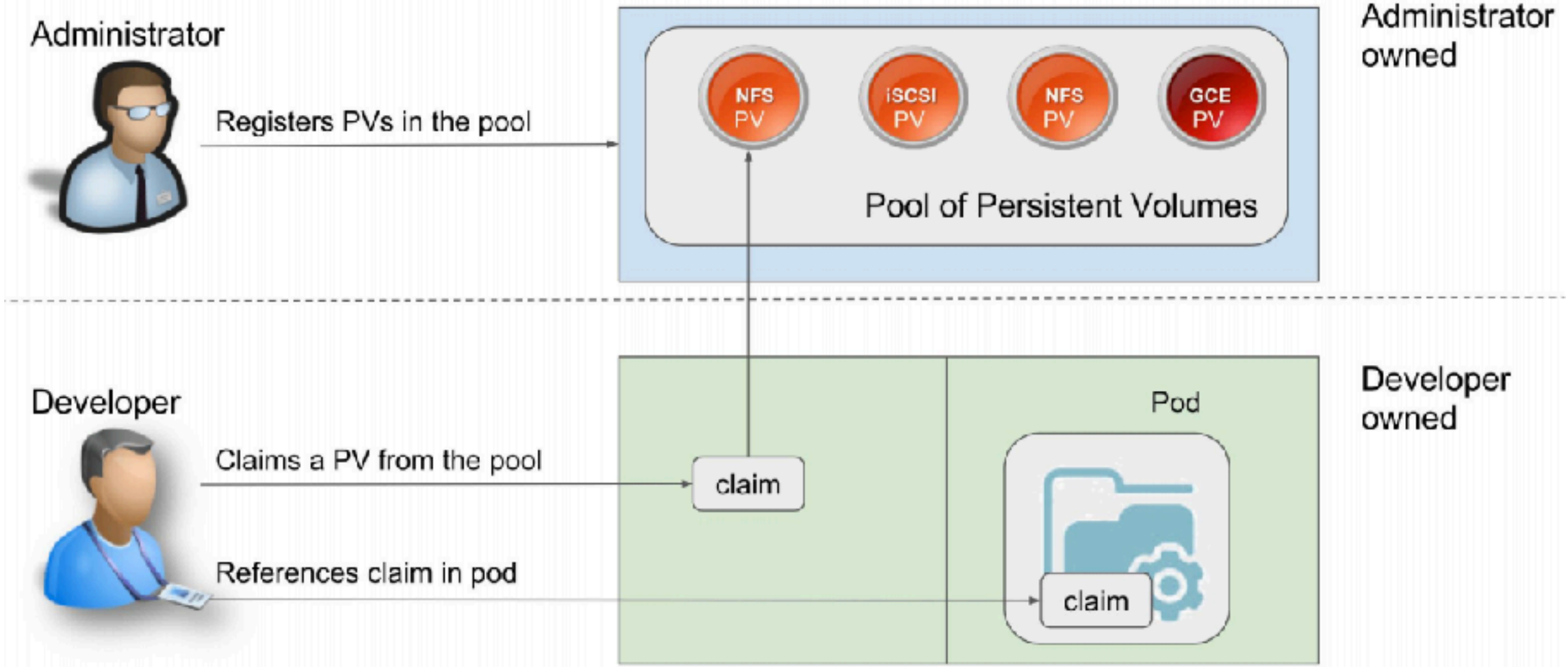


Provisioning

Static
Dynamic

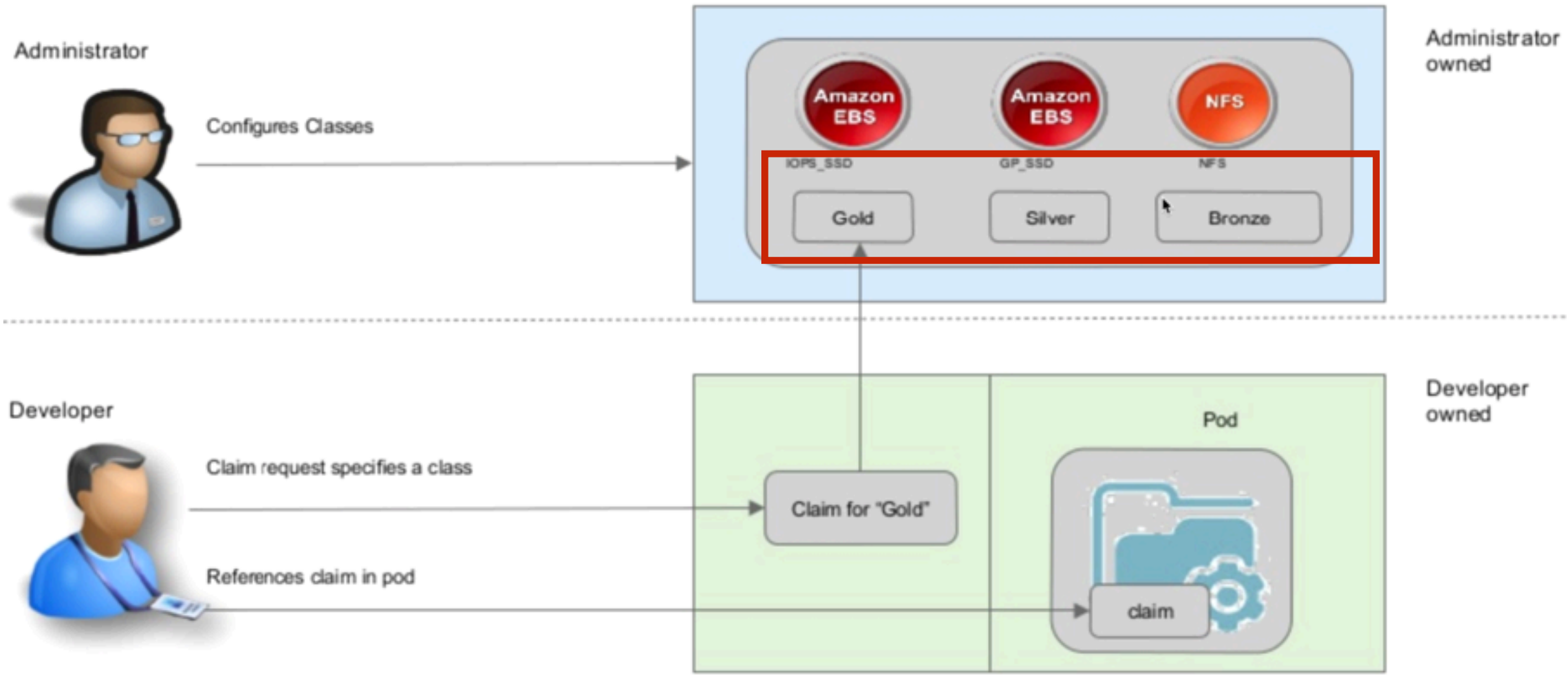
Static

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

```yaml
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
```

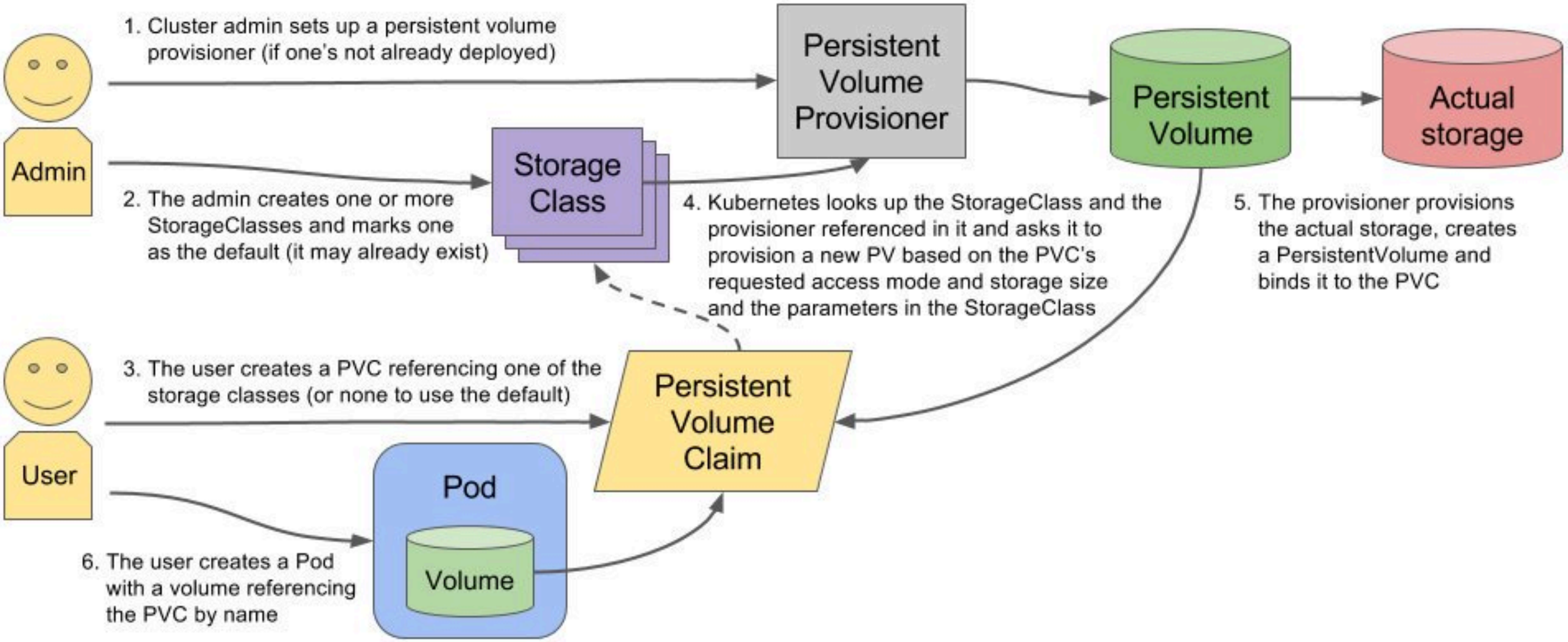
Workshop with Volume

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

```yaml
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
```

```yaml
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

```yaml
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
```

```yaml
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
```

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

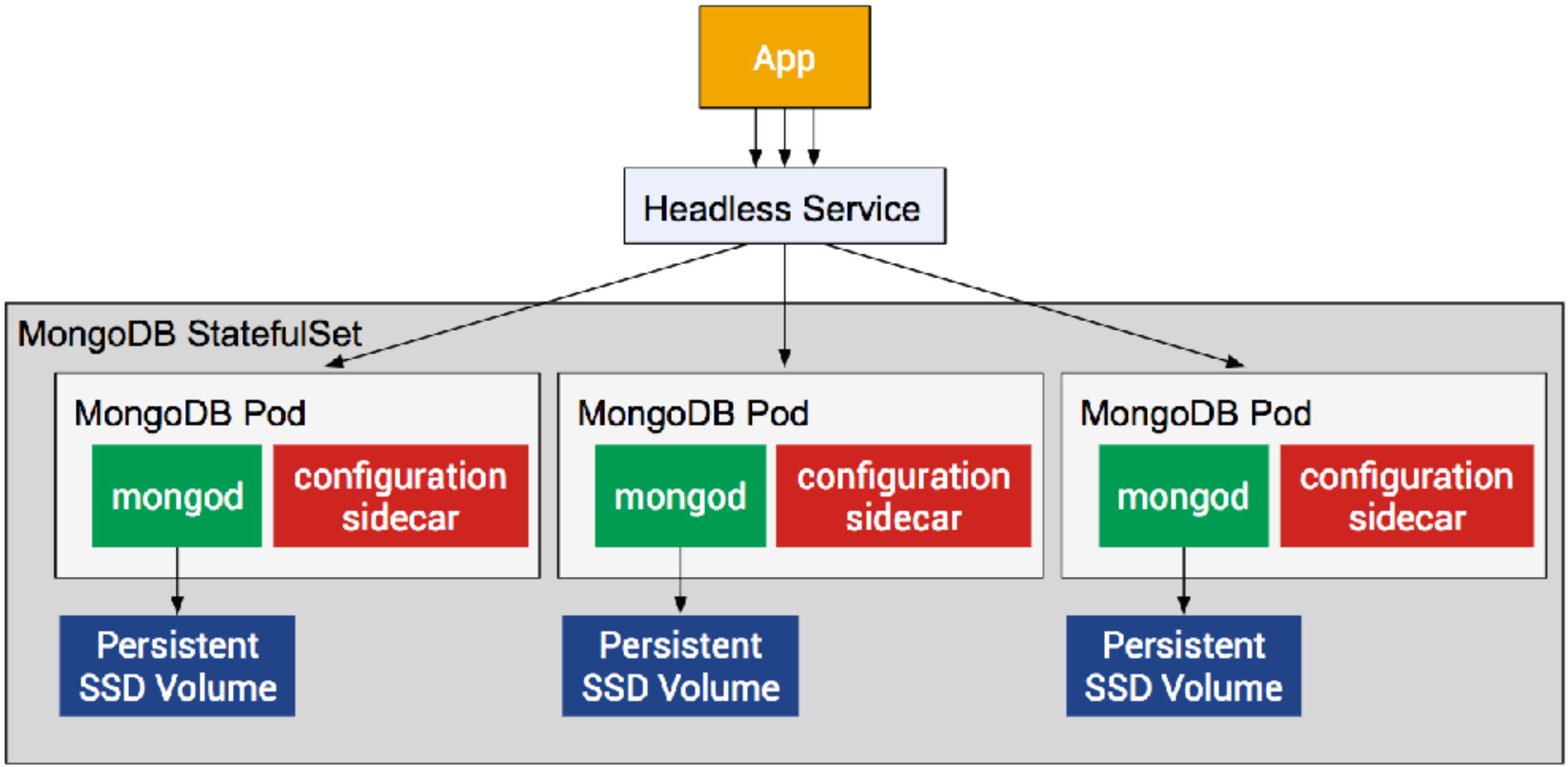
Each Pods is suffix with a predictable (mysql-01, mysql-02)

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

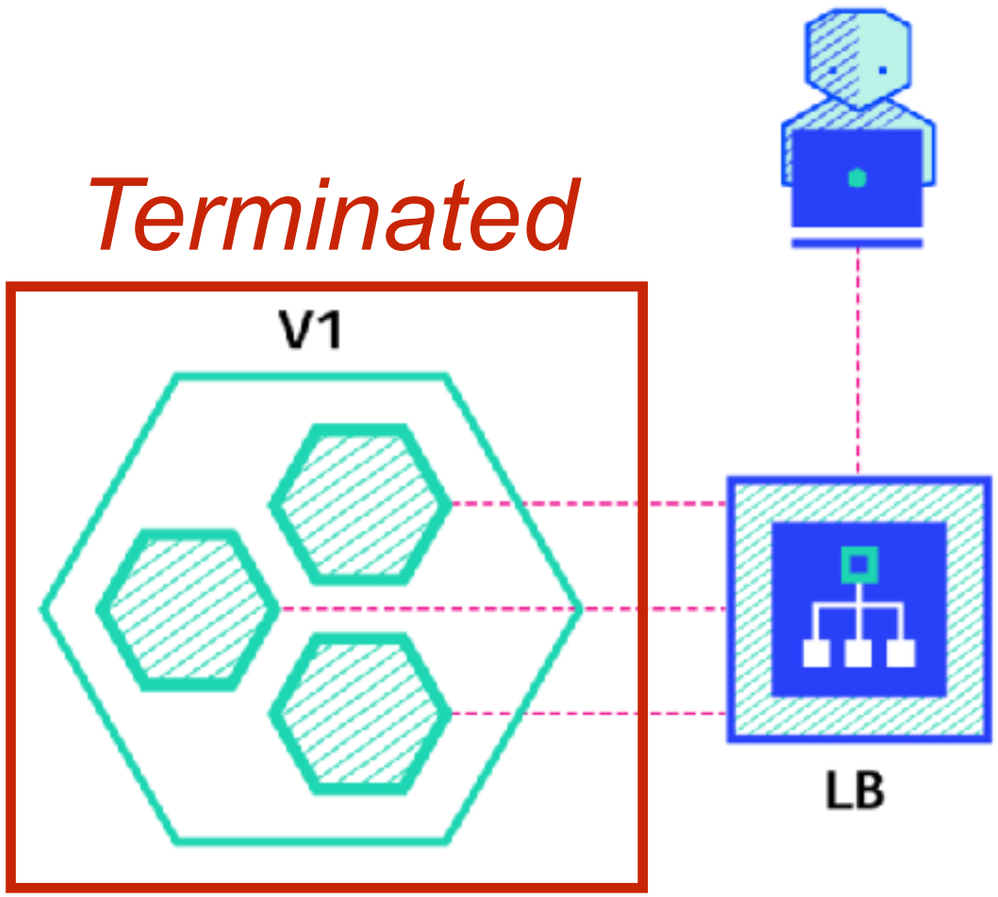
Strategies to deploy

Recreate
Ramped
Blue/Green
Canary
A/B testing
Shadow

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

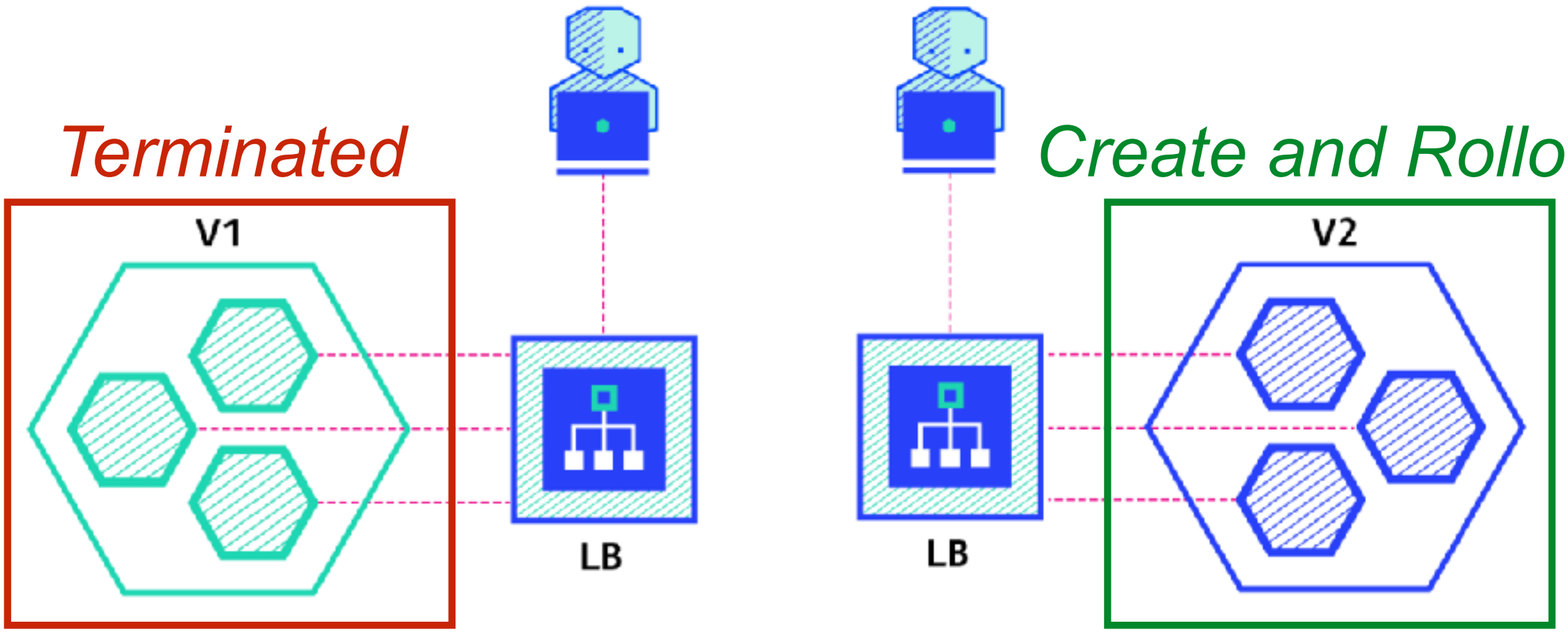
1. Recreate

Version A is terminated then version B is rollout



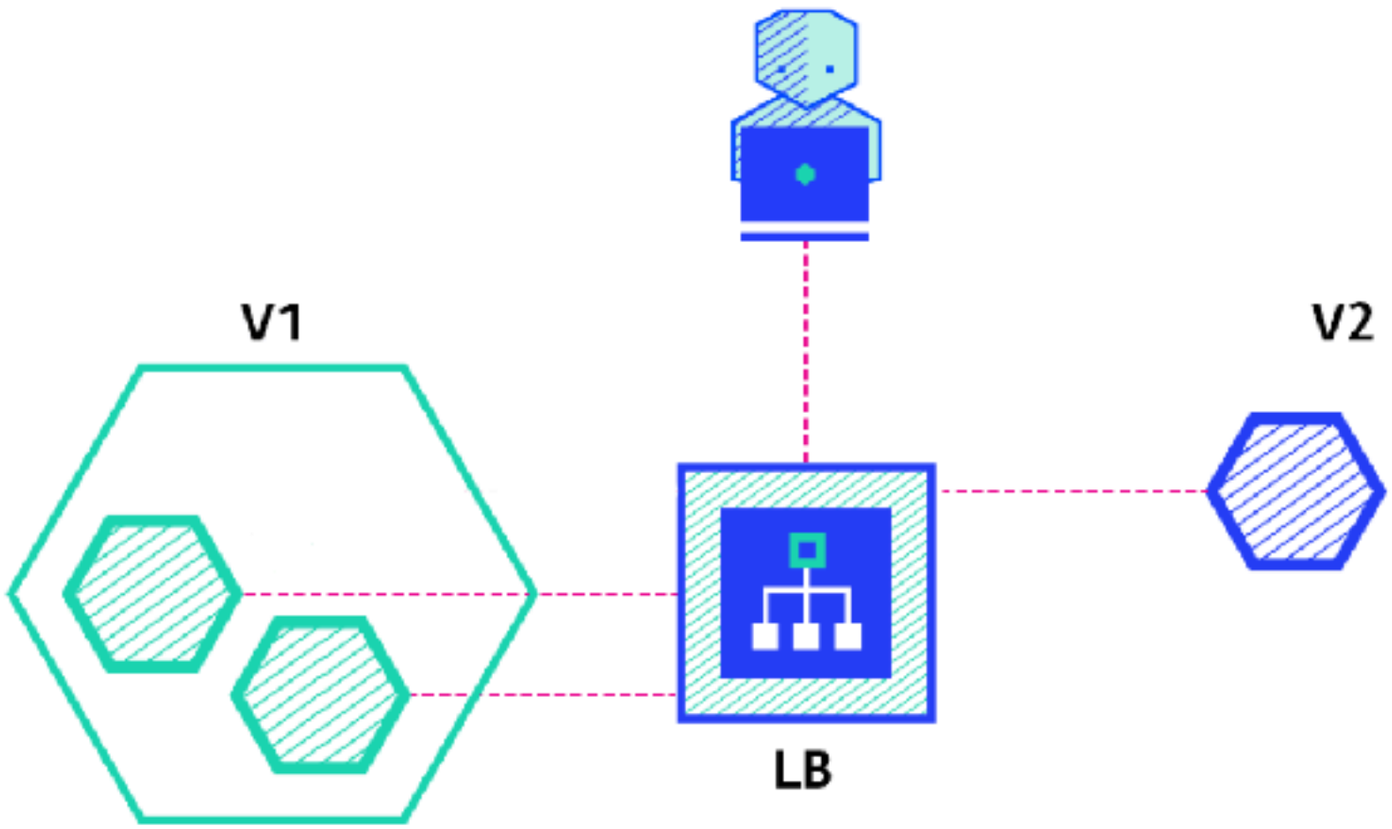
1. Recreate

Version A is terminated then version B is rollout



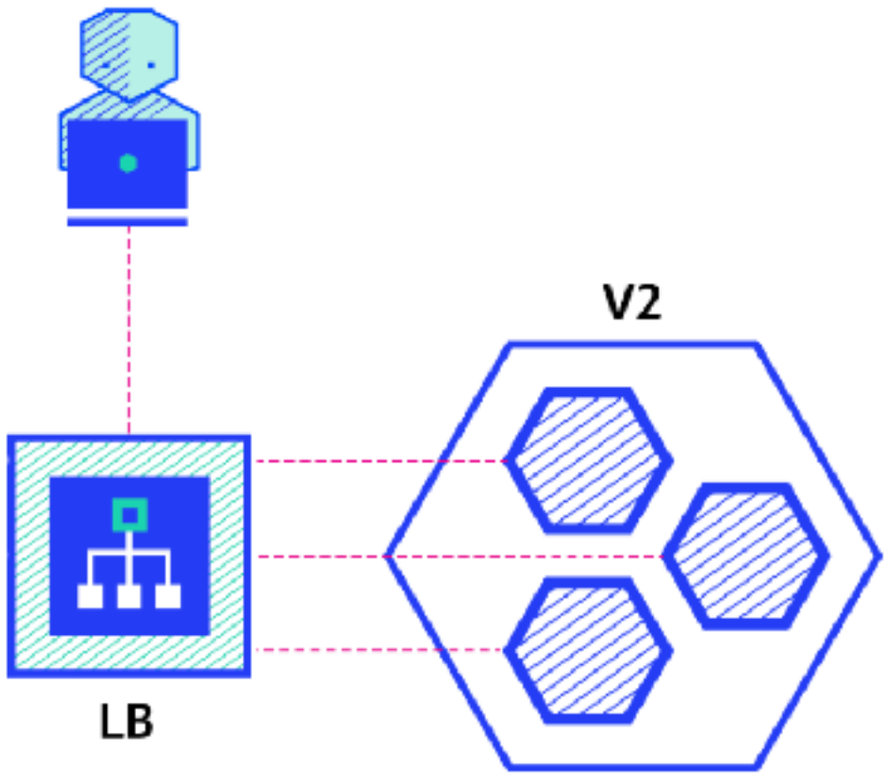
2. Ramped

Slow roll out by replace instance one-by-one



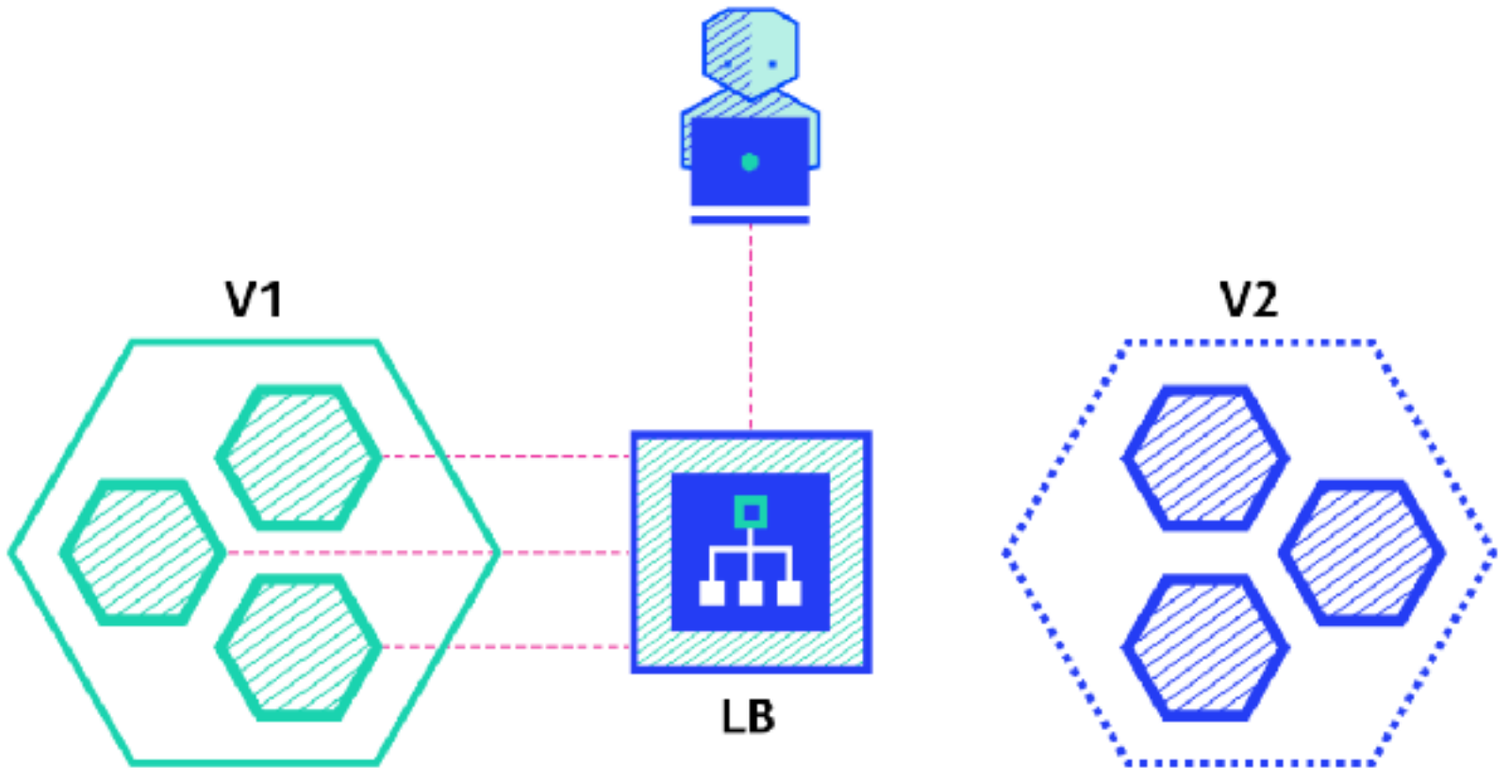
2. Ramped

Slow roll out by replace instance one-by-one



3. Blue/Green

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

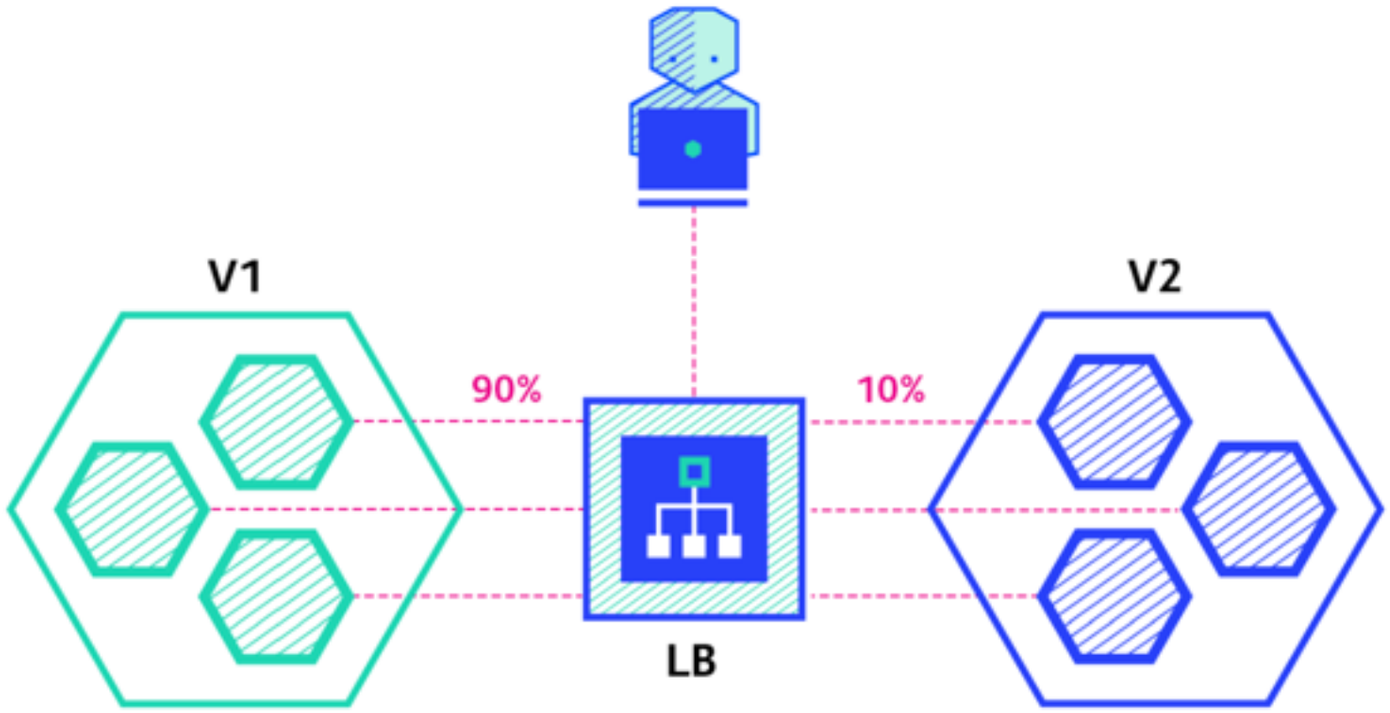


4. Canary

Shift production traffic from version A to B

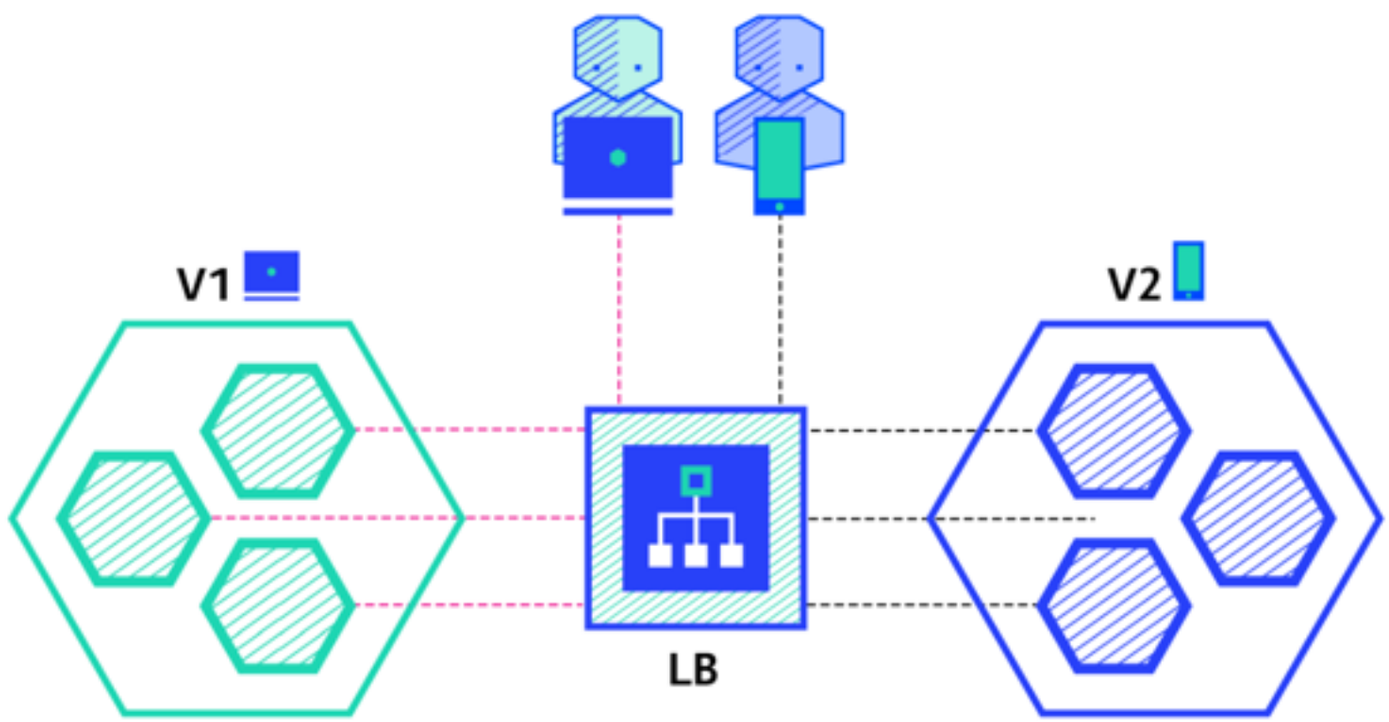
Traffic is split based-on weight

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



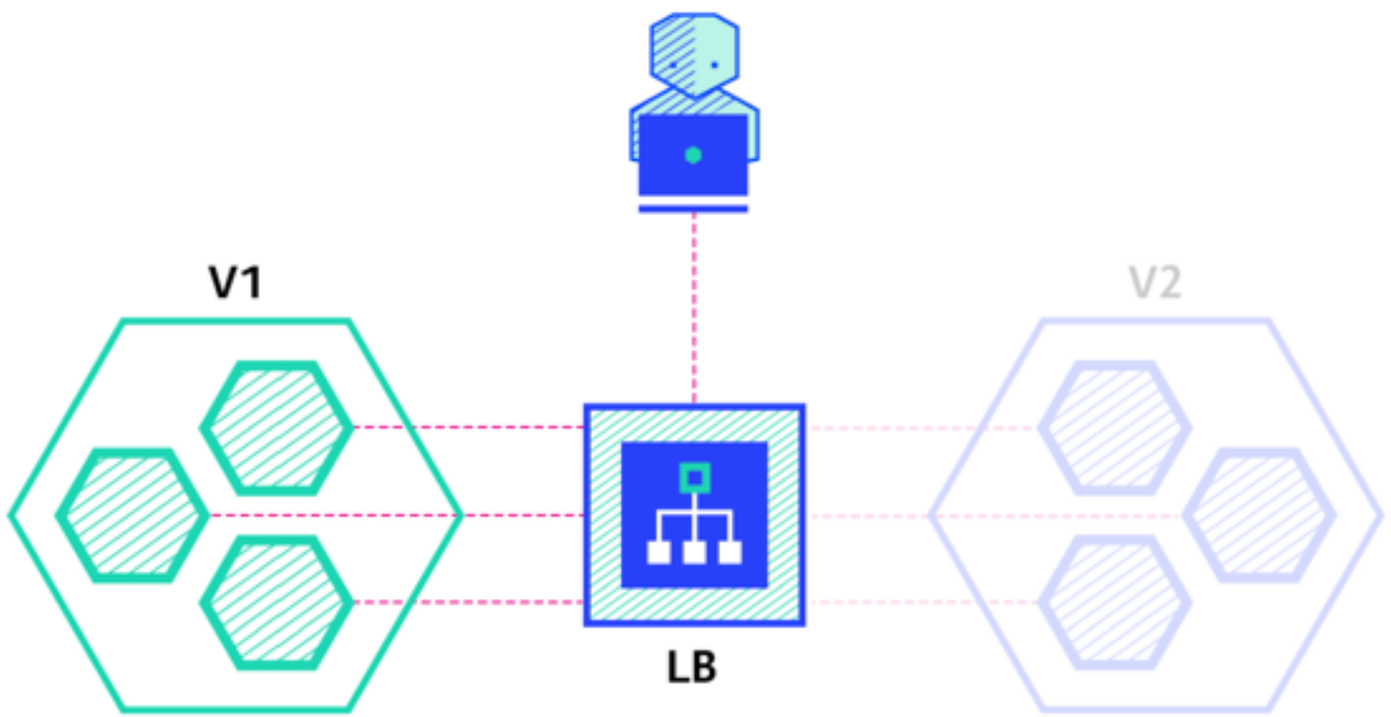
5. A/B testing

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

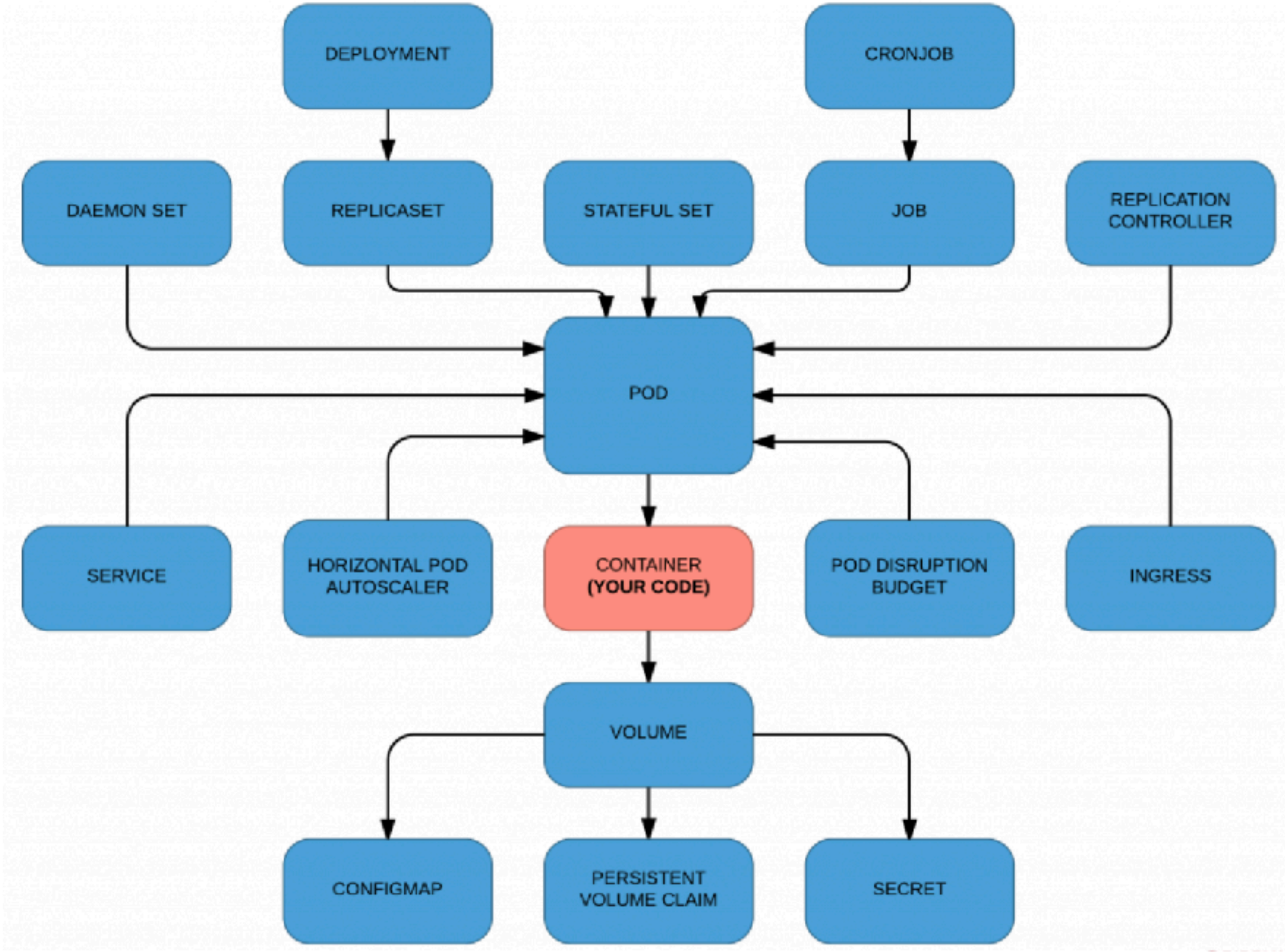
When it comes to production, a ramped or blue/green deployment is usually a good fit, but proper testing of the new platform is necessary.

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.

Container Solutions

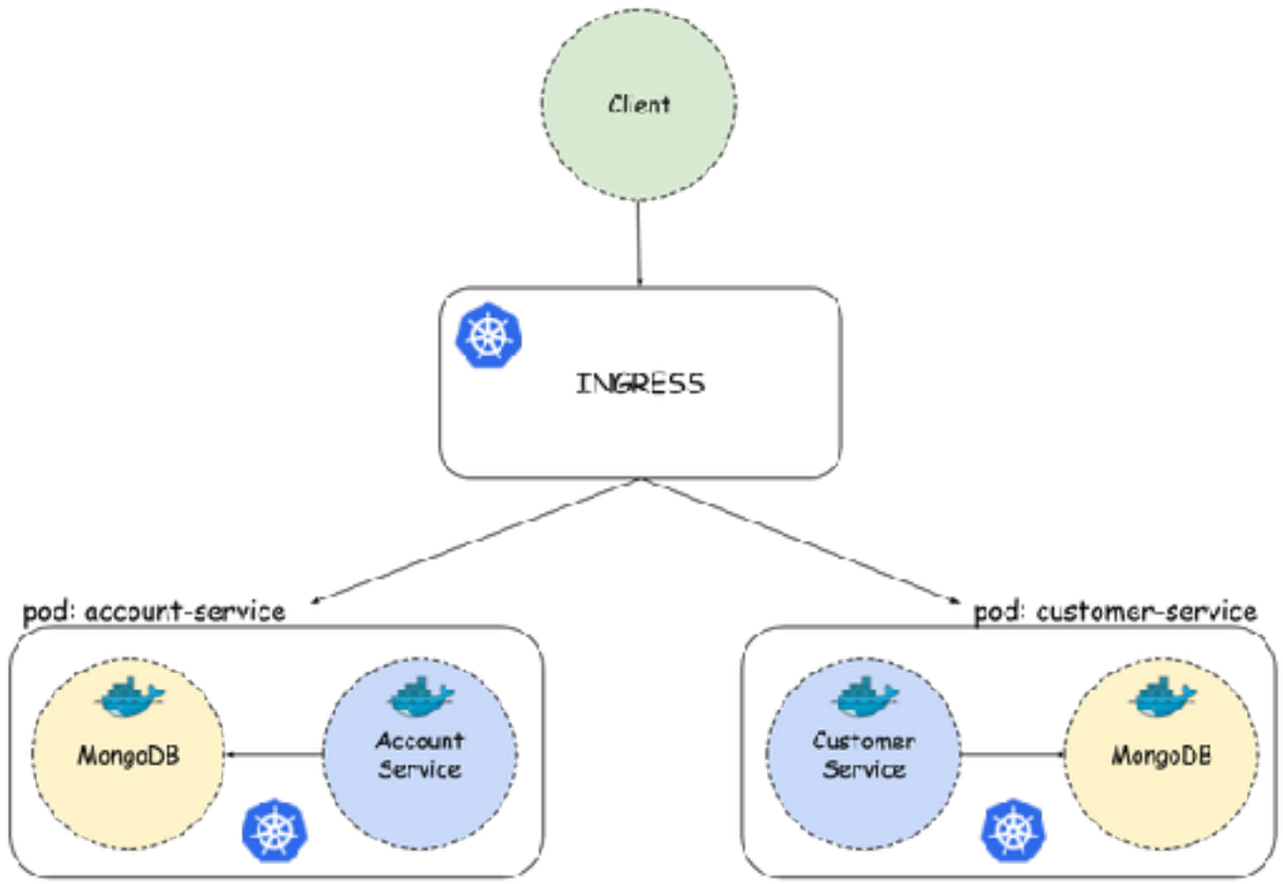
Strategy	ZERO DOWNTIME	REAL TRAFFIC TESTING	TARGETED USERS	CLOUD COST	ROLLBACK DURATION	NEGATIVE IMPACT ON USER	COMPLEXITY OF SETUP
RECREATE version A is terminated then version B is rolled out	✗	✗	✗	■□□	■■■	■■■	□□□
RAMPED version B is slowly rolled out and replacing version A	✓	✗	✗	■□□	■■■	■□□	■□□
BLUE/GREEN 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

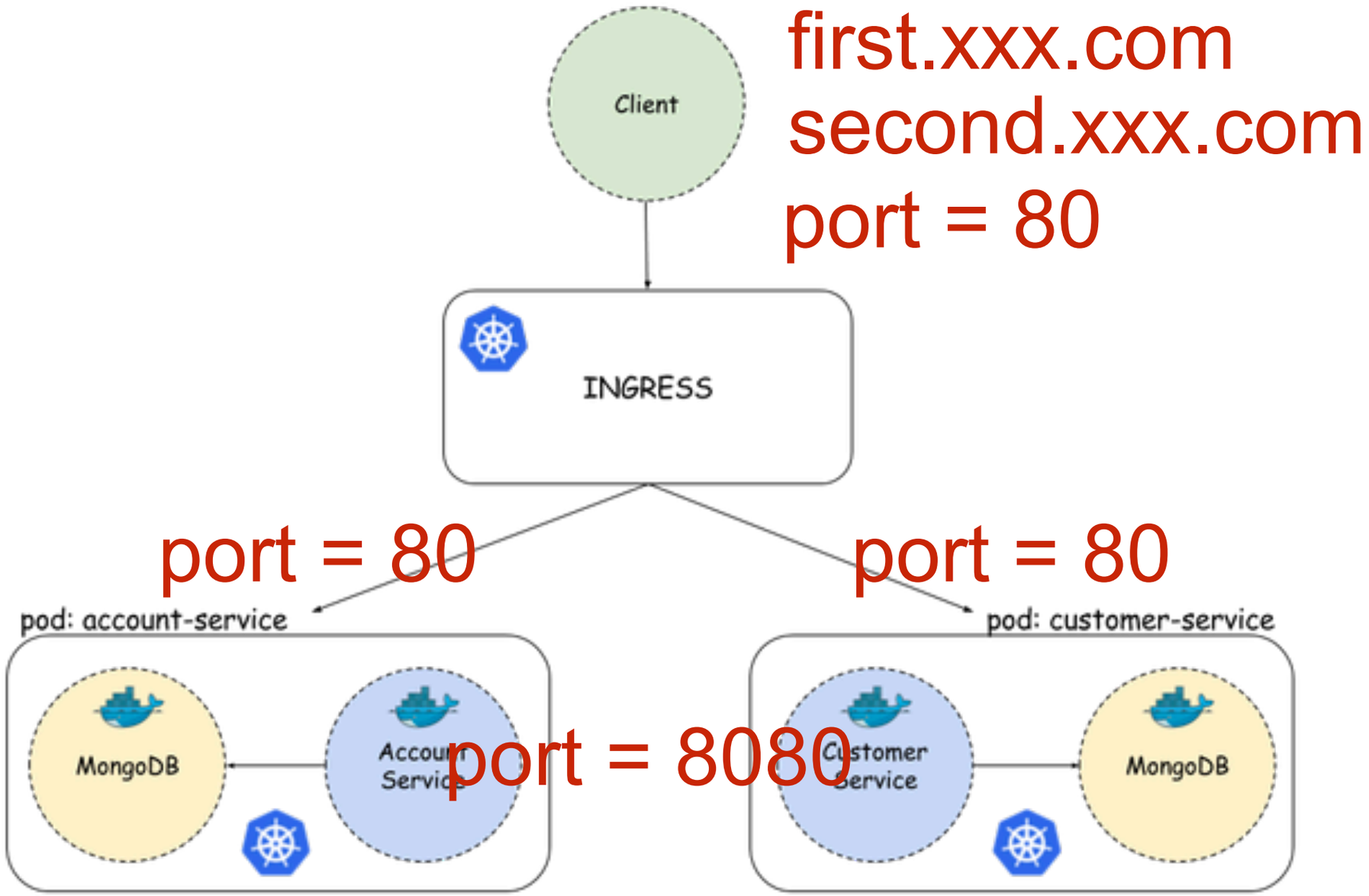
Ingress Network

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

Job and Cronjob

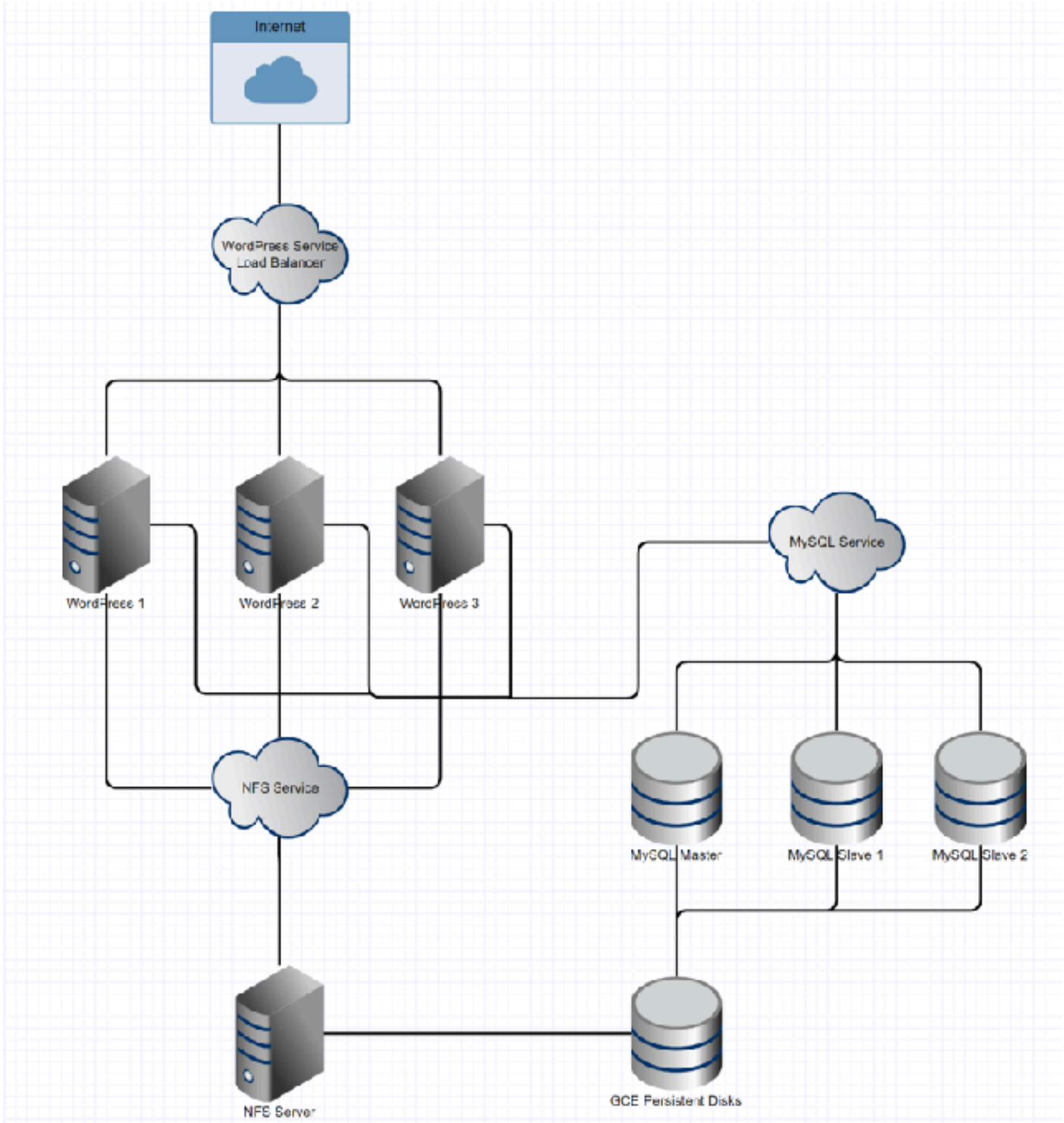
Thank you

Wordpress and MySQL

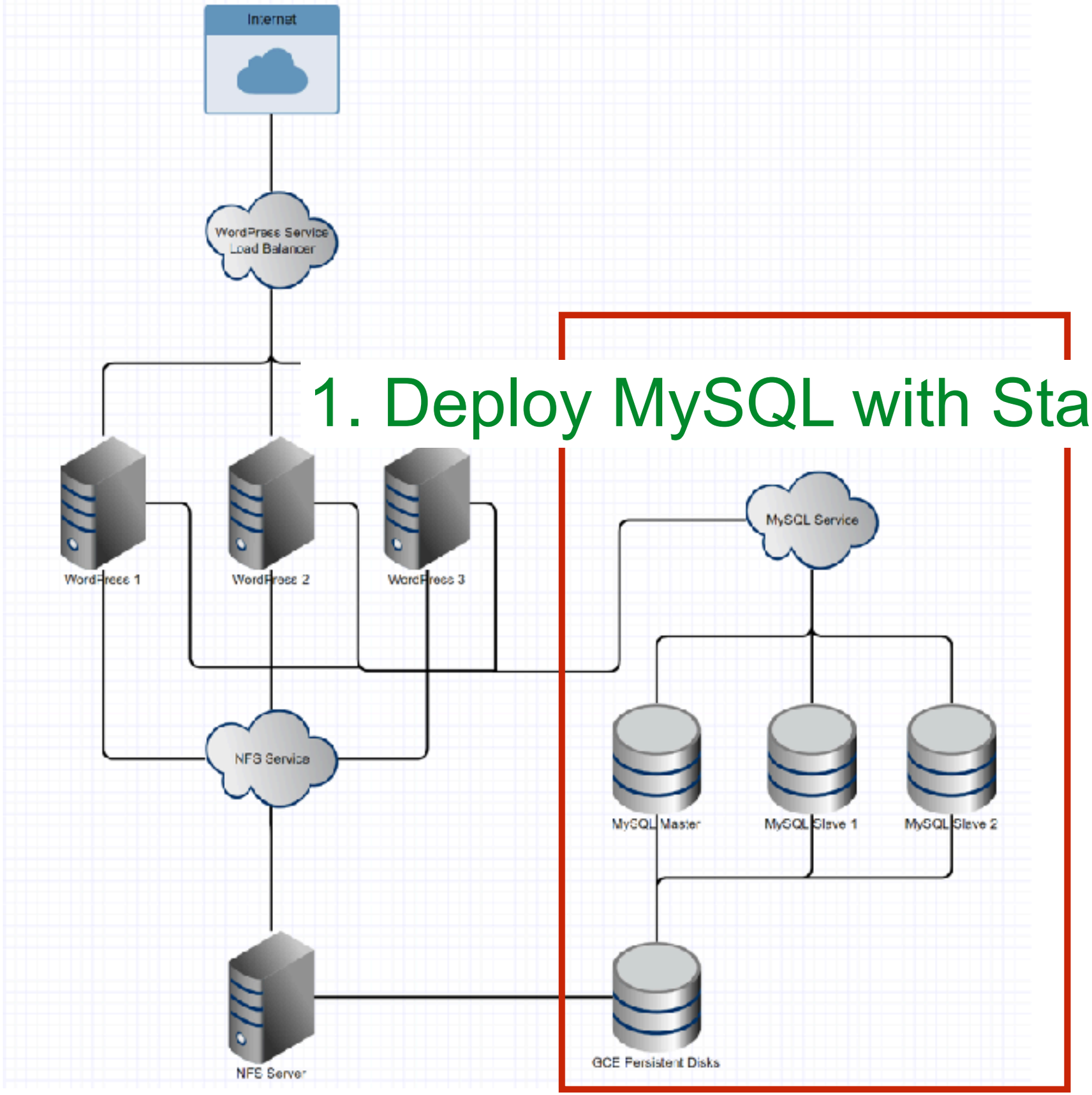
File /workshop-wordpress

Architecture

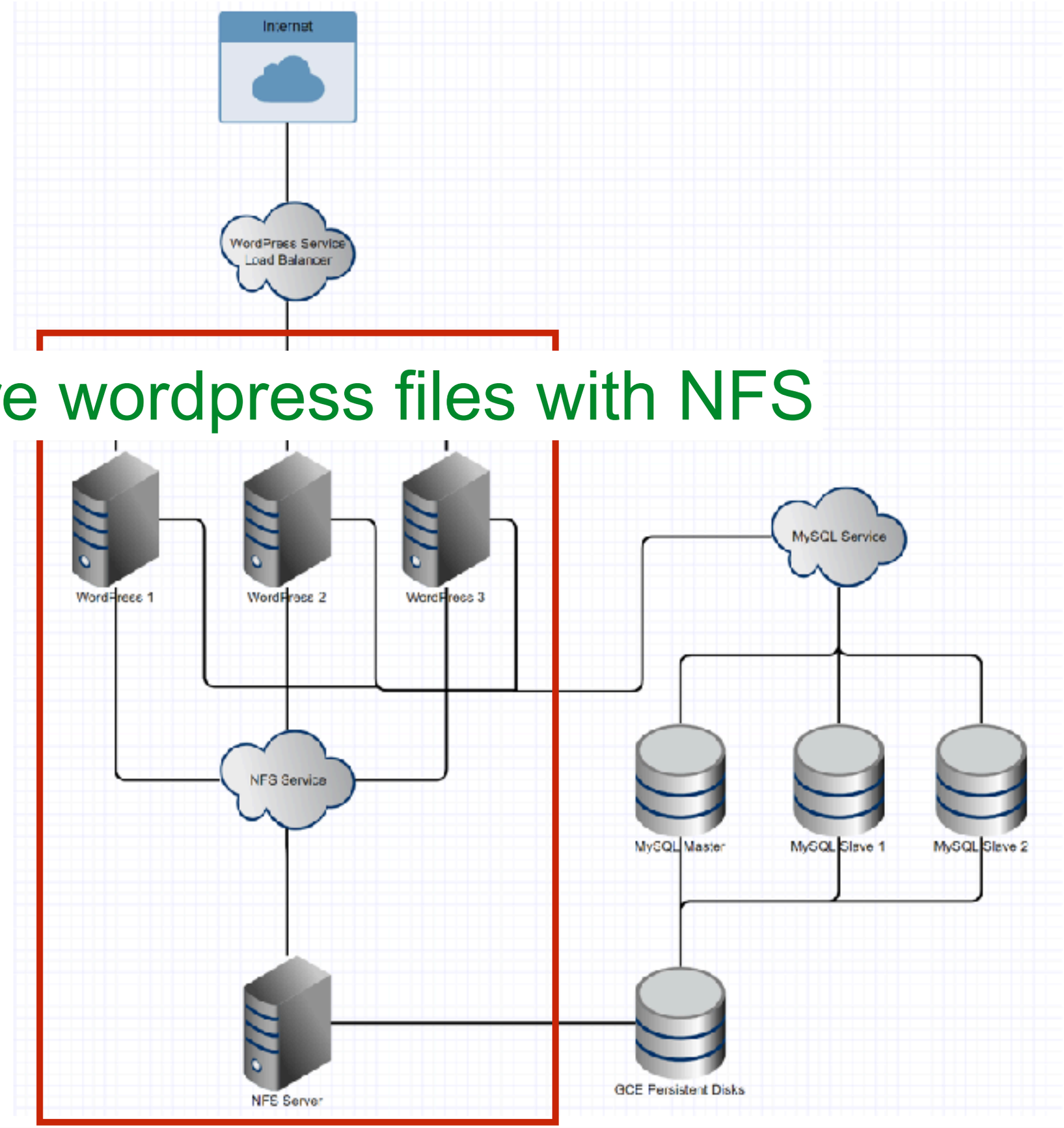
Storage for Wordpress application file with **NFS**
Database cluster with **MySQL**
Use Kubernetes load balance and **service** networking



1. Deploy MySQL with StatefulSet



2. Share wordpress files with NFS



3. Expose port with service