สยามชำนาญกิจ
SIAM CHAMNAN KIT

Docker and Kubernetes

Somkiat Puisungnoen

Somkiat | Home

Somkiat Puisungnoen

Update Info 1 | View Activity Log 10+ | ...

Timeline | About | Friends 3,138 | Photos | More

When did you work at Opendream?

22 Pending Items

Intro

Software Craftsmanship

Software Practitioner at สยามชำนาญกิจ พ.ศ. 2556

Agile Practitioner and Technical at SPRINT3r

Post | Photo/Video | Live Video | Life Event

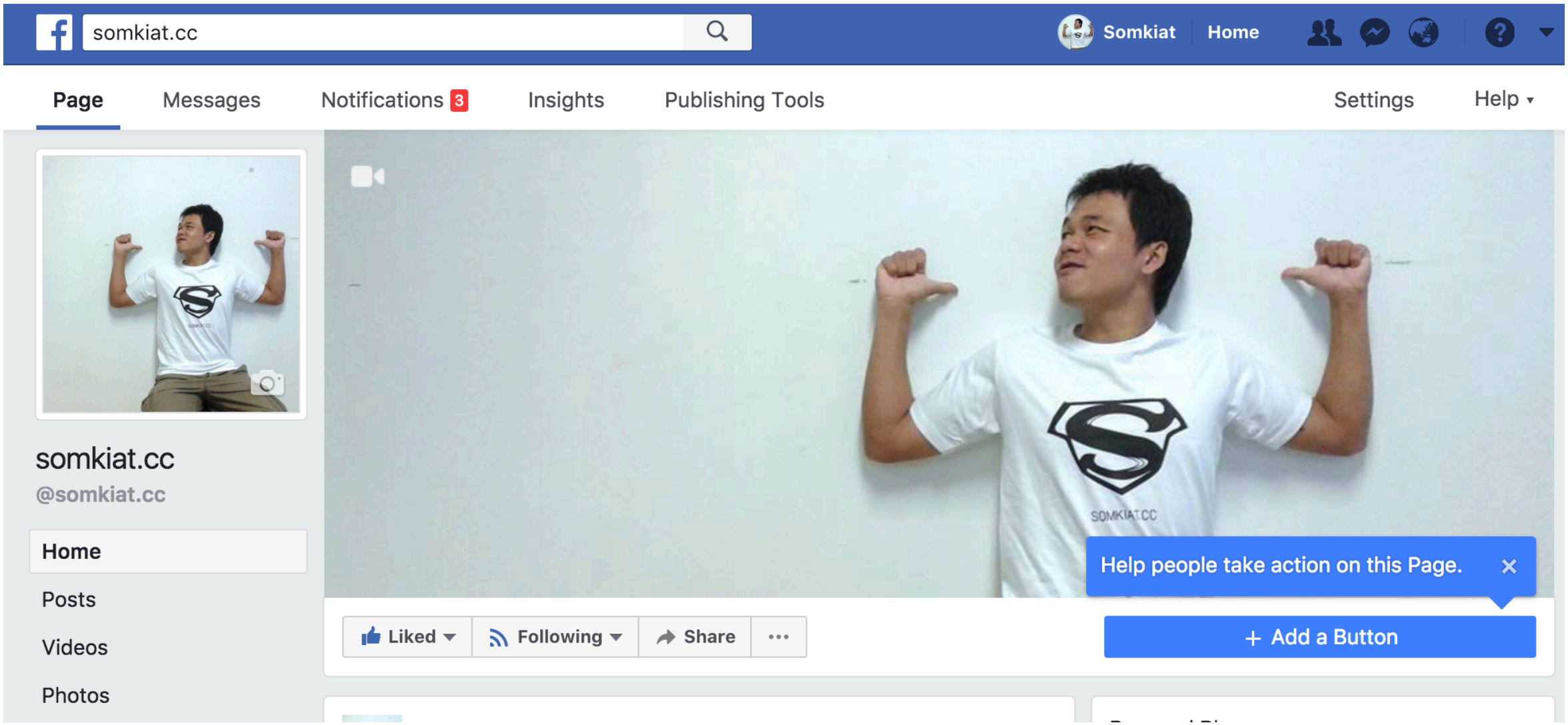
What's on your mind?

Public | Post

Somkiat Puisungnoen

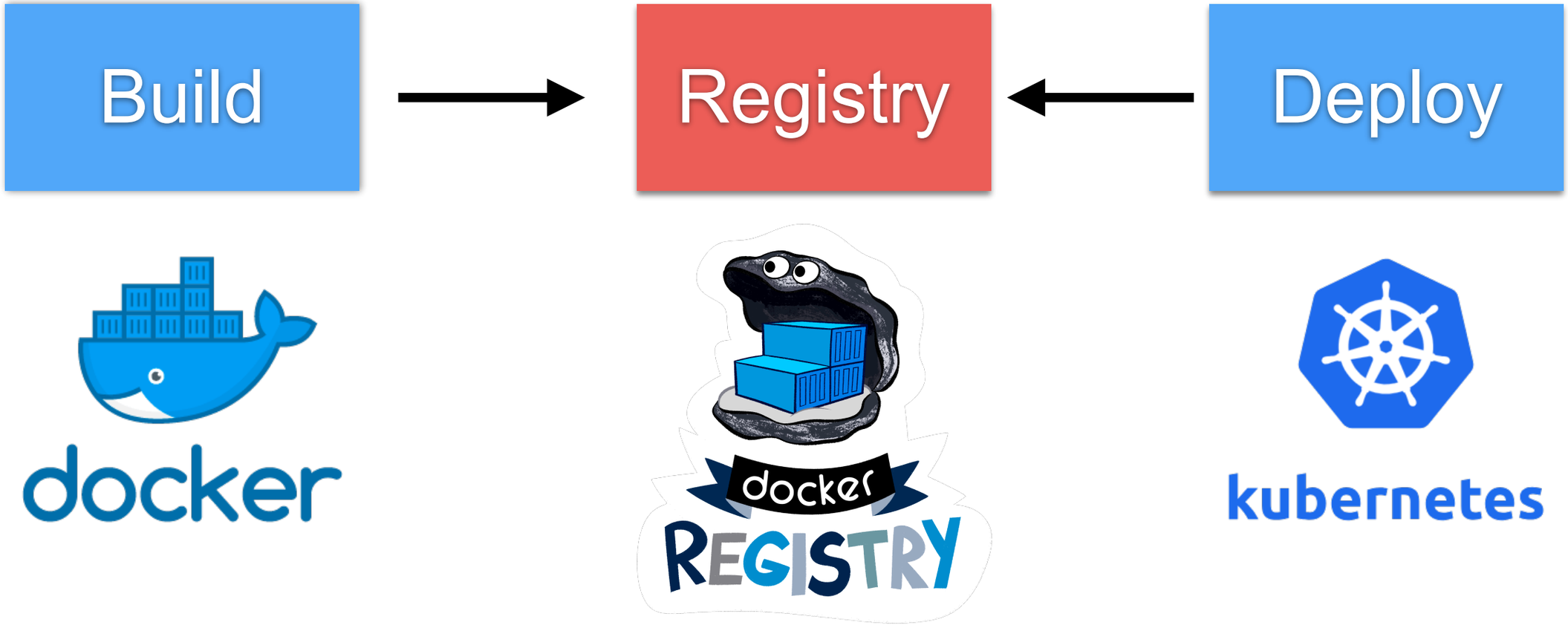
15 mins · Bangkok ·

Java and Bigdata

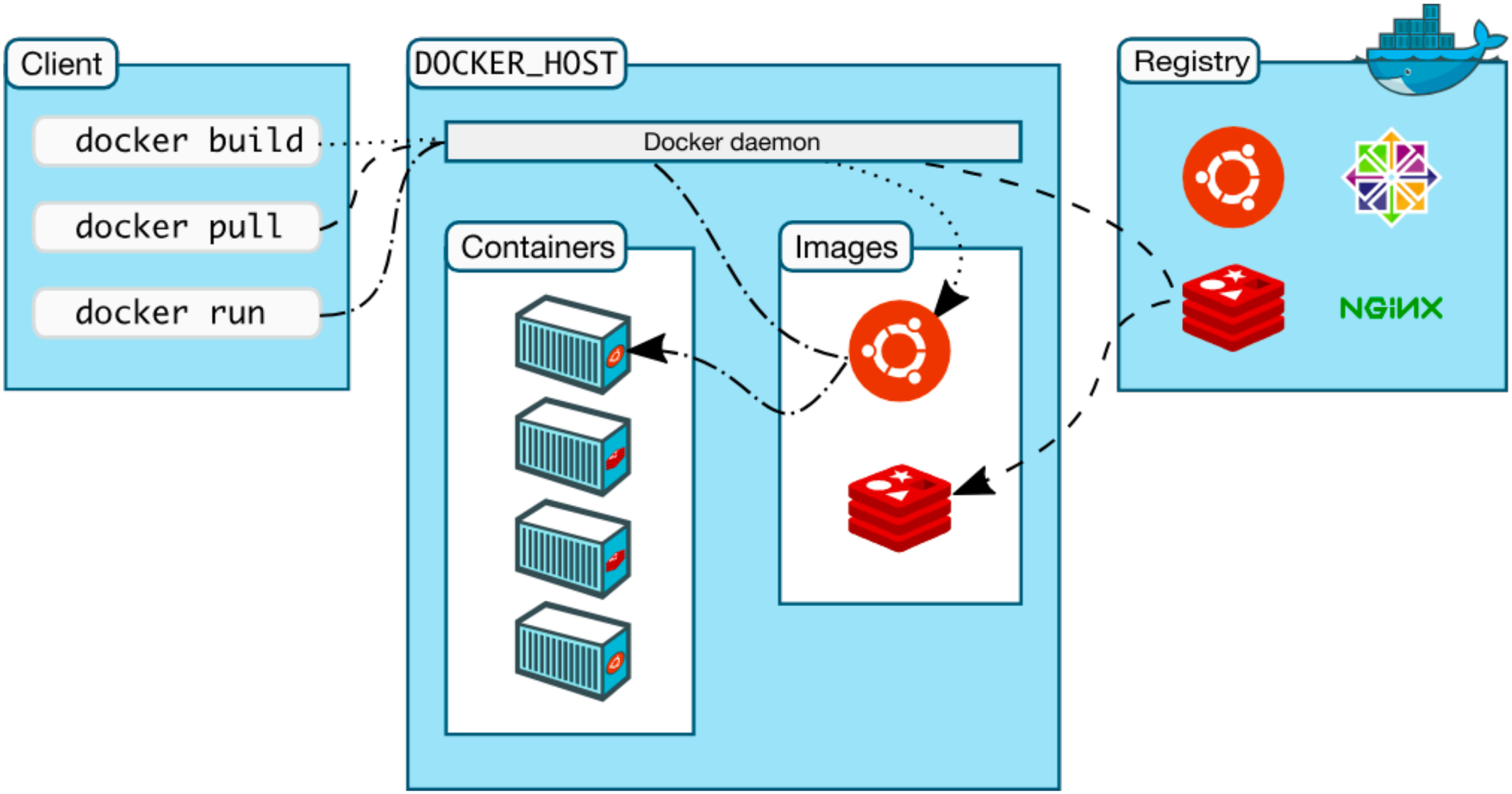


https://github.com/up1/demo-docker-k8s

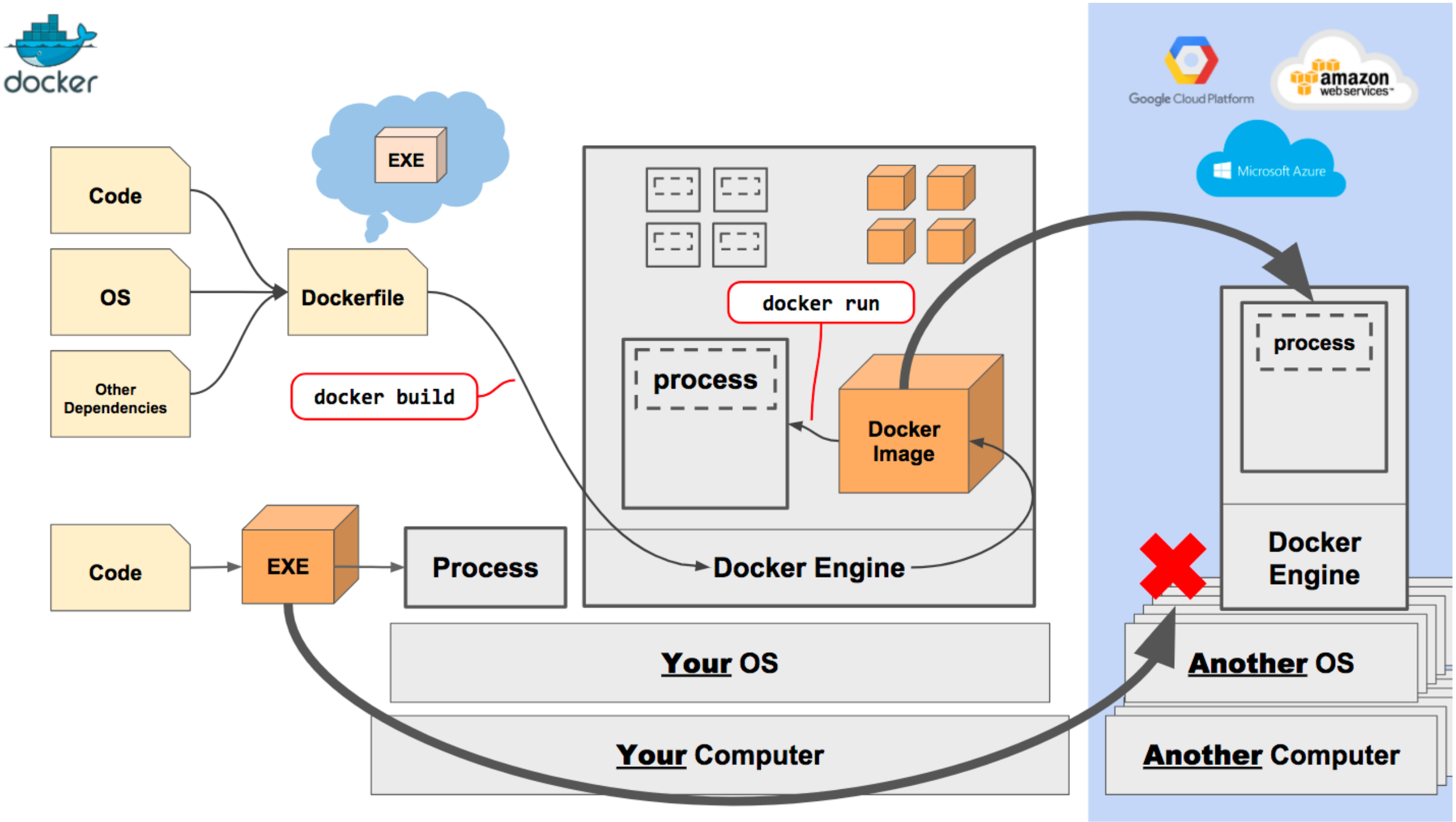
Goals



Docker overview



Docker overview



Docker command

$docker-compose build
$docker-compose up
$docker-compose down

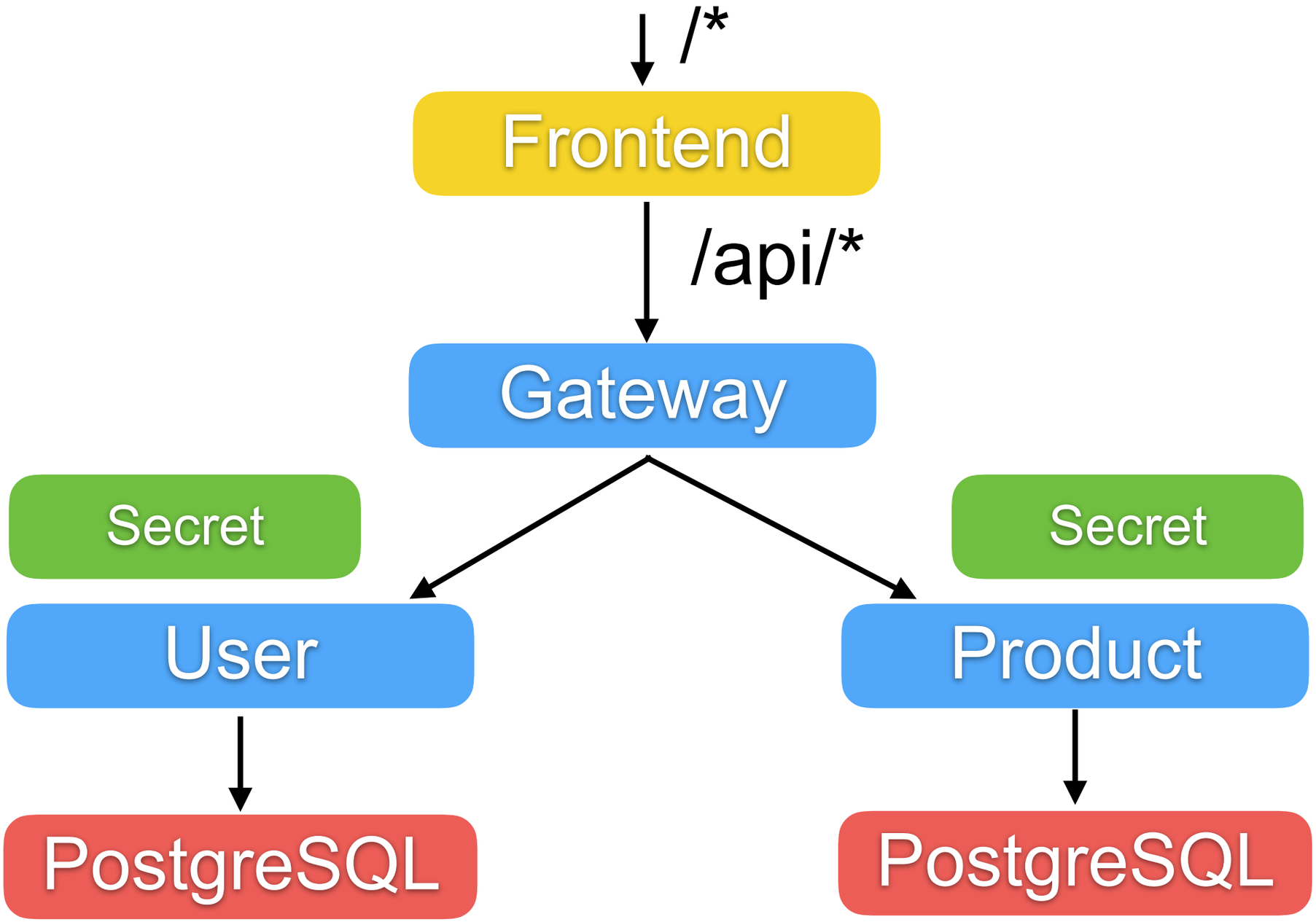
Building Docker image

From container
Dockerfile

Docker compose

All-in-one with simple command
$docker-compose build/up/down

Structure of service



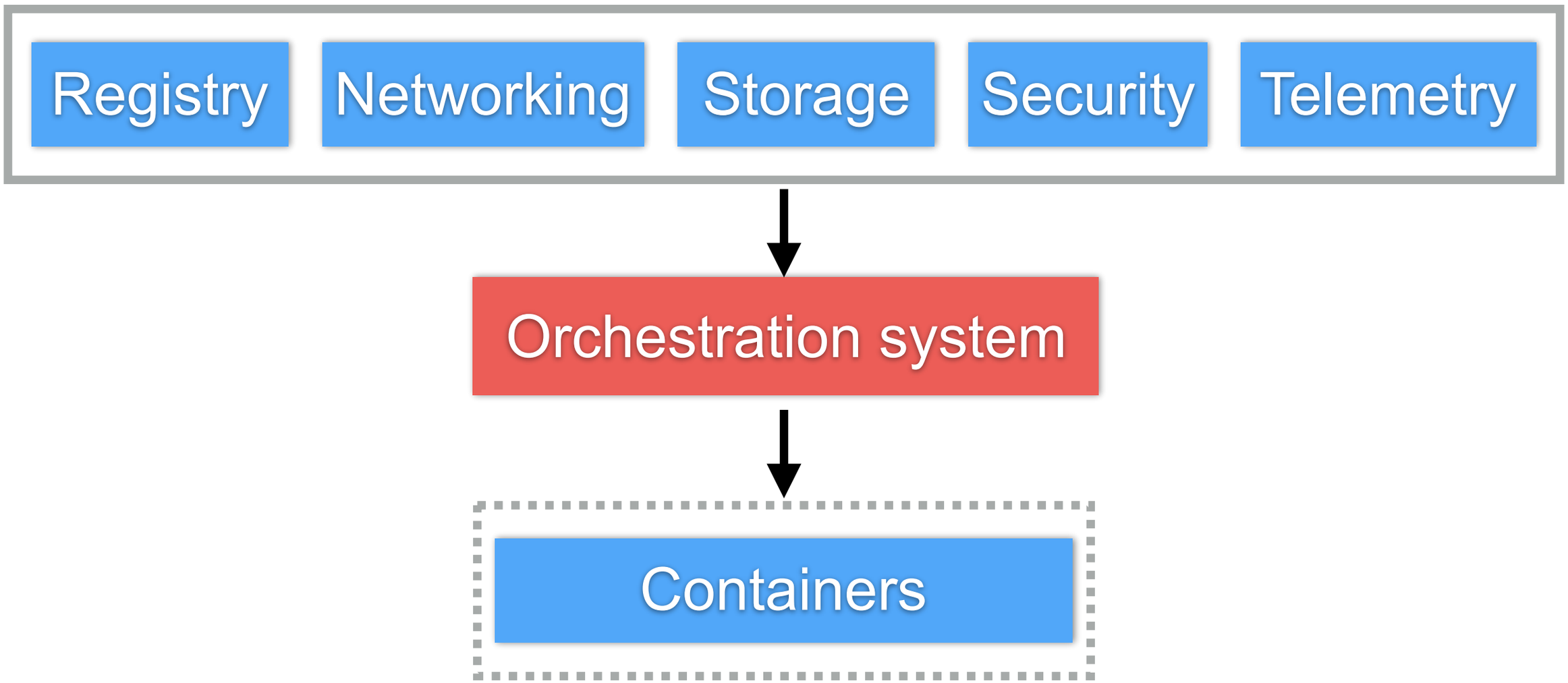
Workshop

https://github.com/up1/demo-docker-k8s

Introduction Kubernetes

Kubernetes

Open source **orchestration** for containers
Docker and rkt



Kubernetes

Provide declarative primitives for **desired state**

Self-healing

Horizontal scaling

Automatic binpacking

Service discovery

Load balancing

Benefits

Deploy application quickly and predictable
Scaling in the fly
Release new feature seamless
Fail-proofing
Limit hardware usage
Agile software development
Portability between OS, host and cloud provider

Kubernetes concepts

Pods

Replication controller

Deployment/ReplicaSet

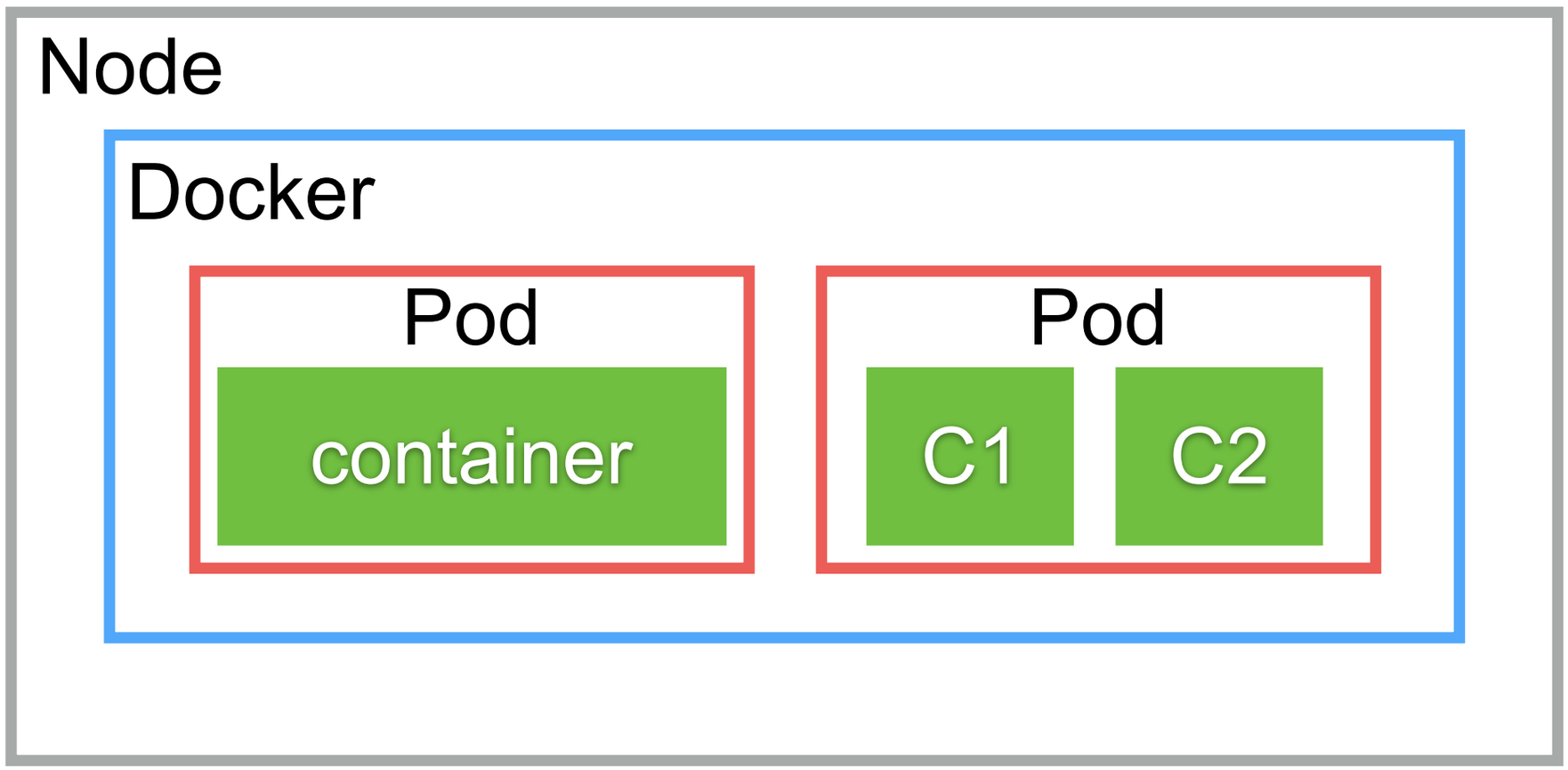
Service

Label

Namespace

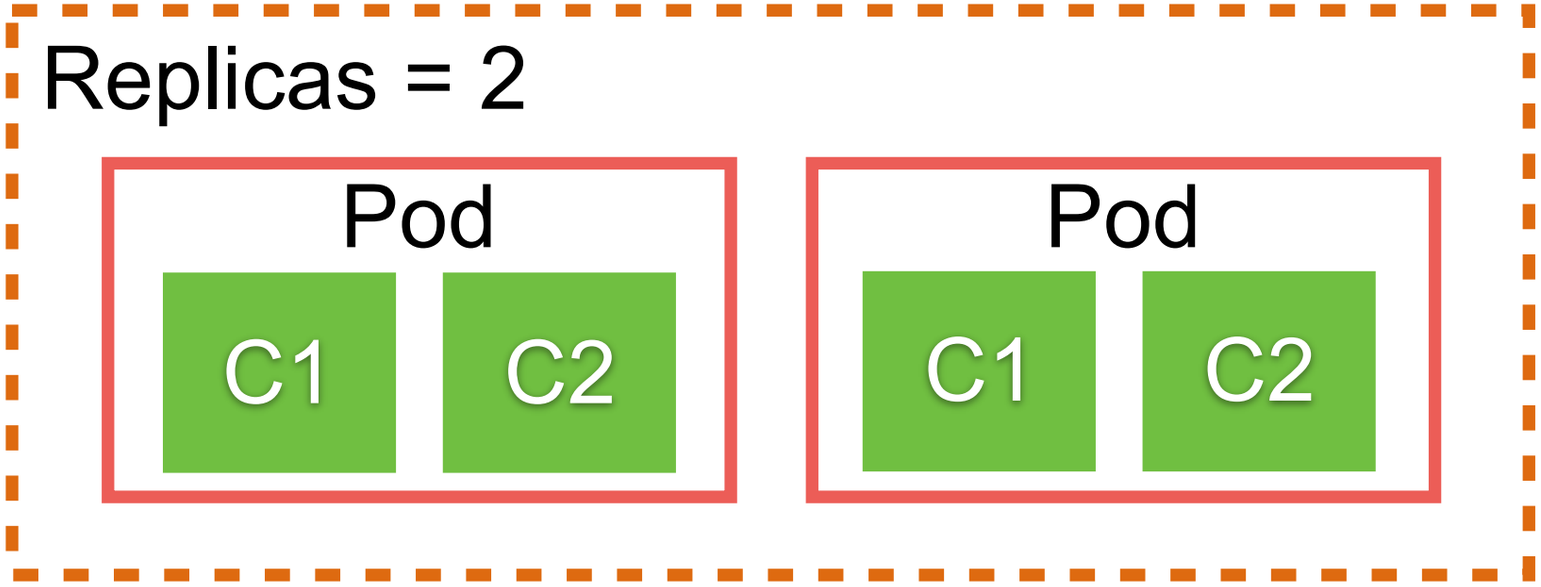
Pods

Colocated group of containers
Share IP, storage and namespace



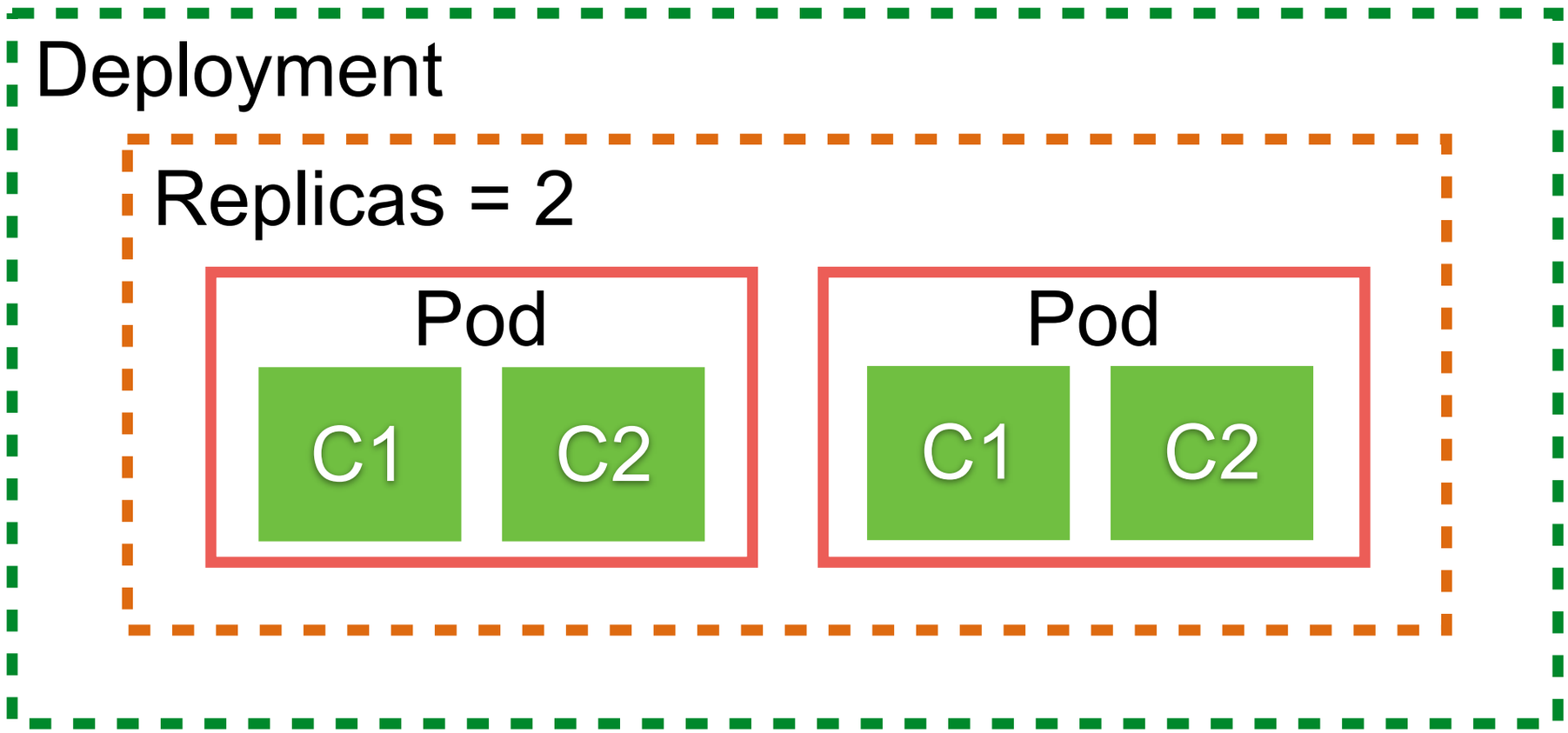
Replication Controller (RC)

Manage the lifecycle of Pod
Ensure specified number are running



ReplicaSet (RS)

Next generation of Replication Controller
Generally create with **Deployment**
Enable Horizontal Pod Autoscaling (HPA)



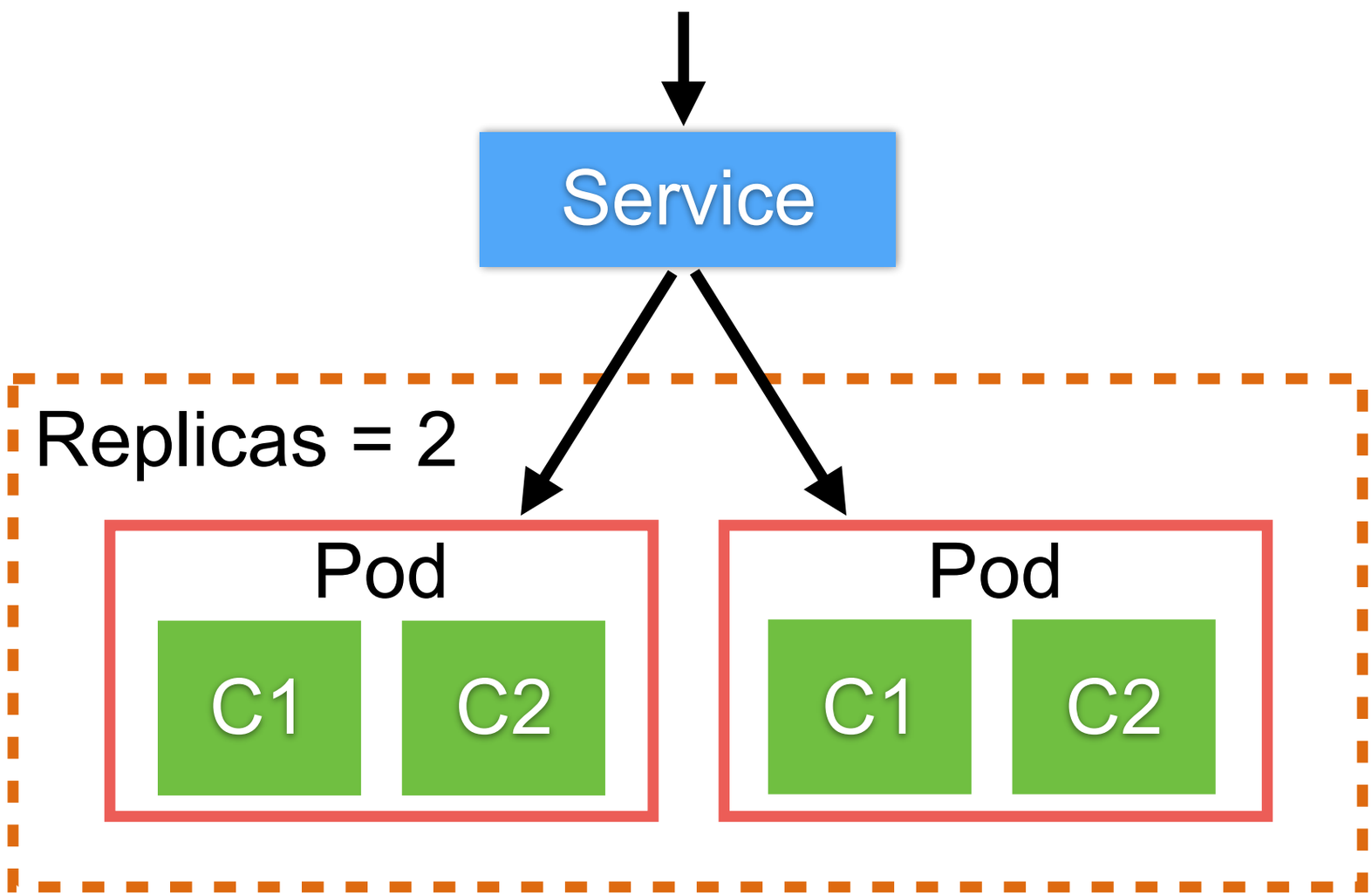
Deployment

Declarative update for Pod and RS
Rolling update
Rollback to previous version

Service

Single and stable name of the set of Pods

Act as a Load Balancer



Publish services

ClusterIP

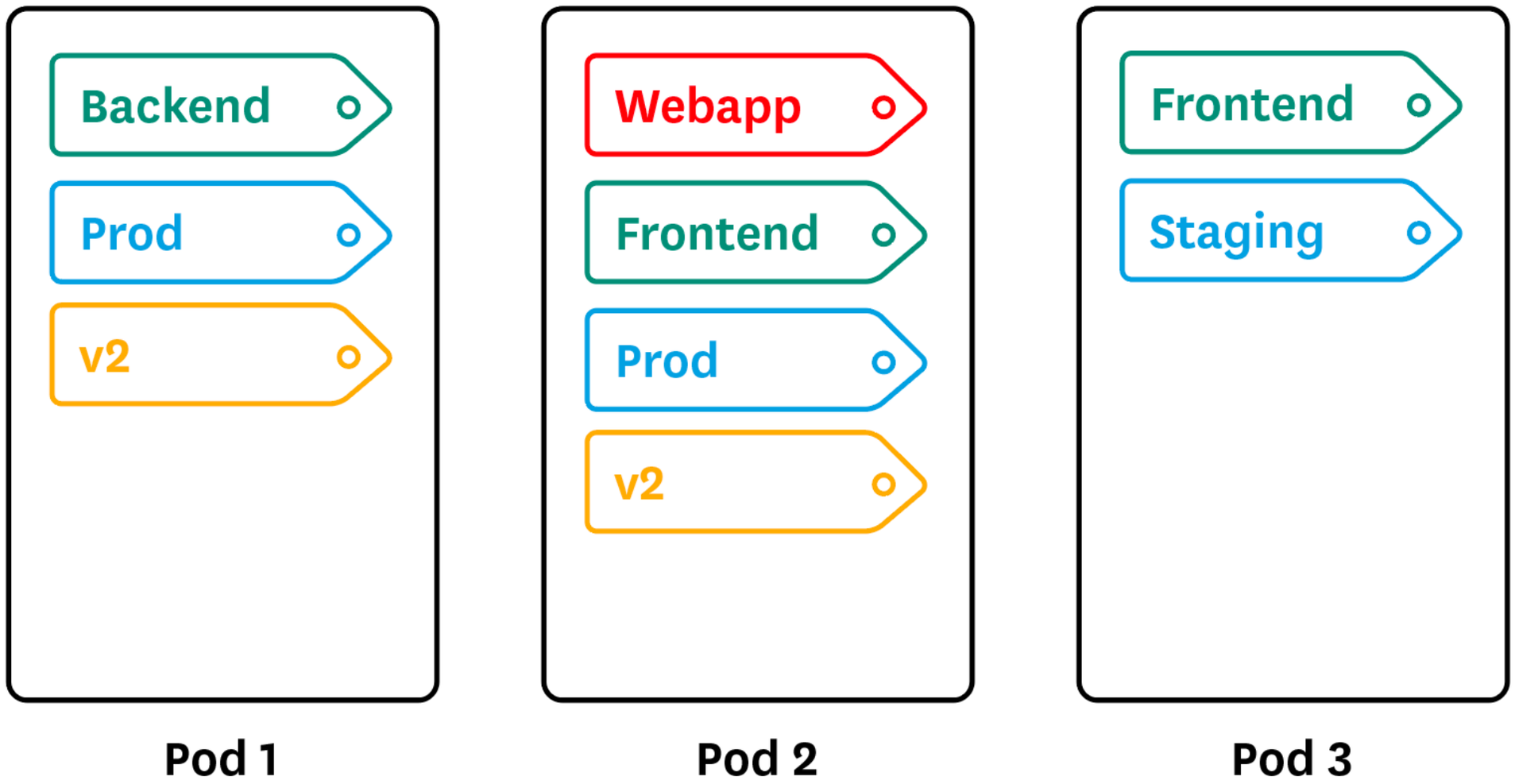
NodePort

LoadBalancer

ExternalName

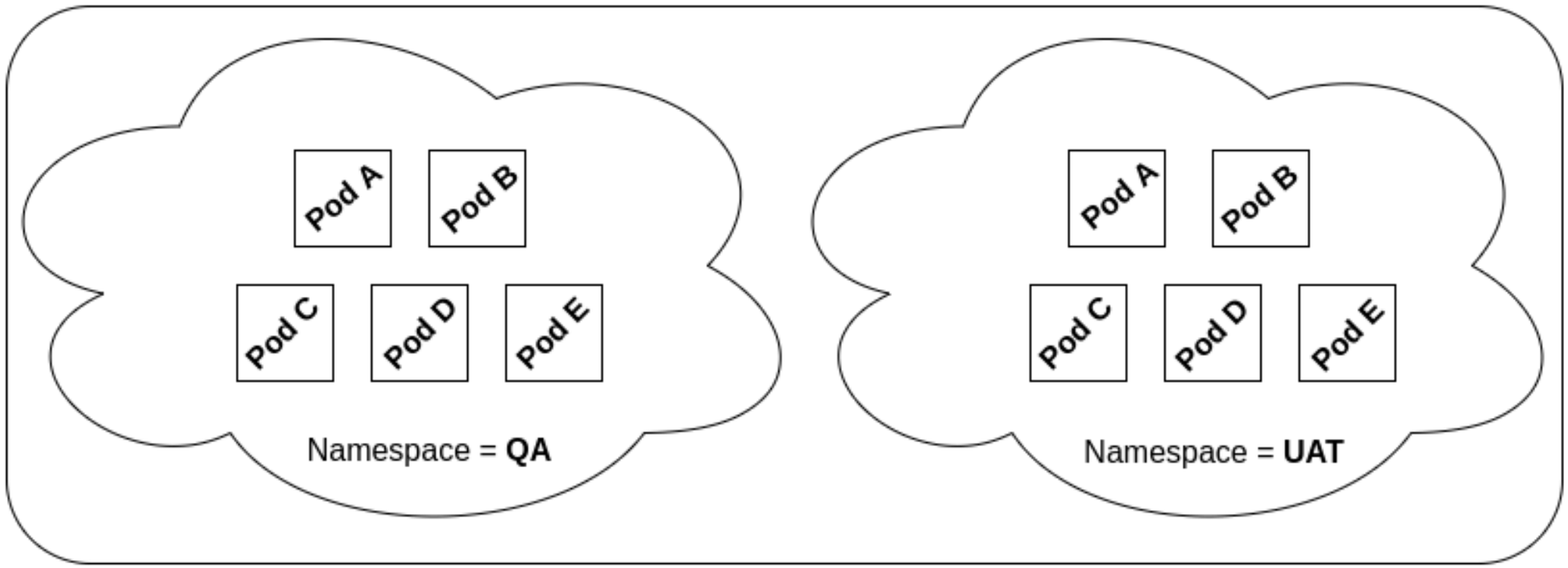
Label

Use to organize and select group of objects



Namespace

Namespace sets up virtual clusters over the same physical cluster



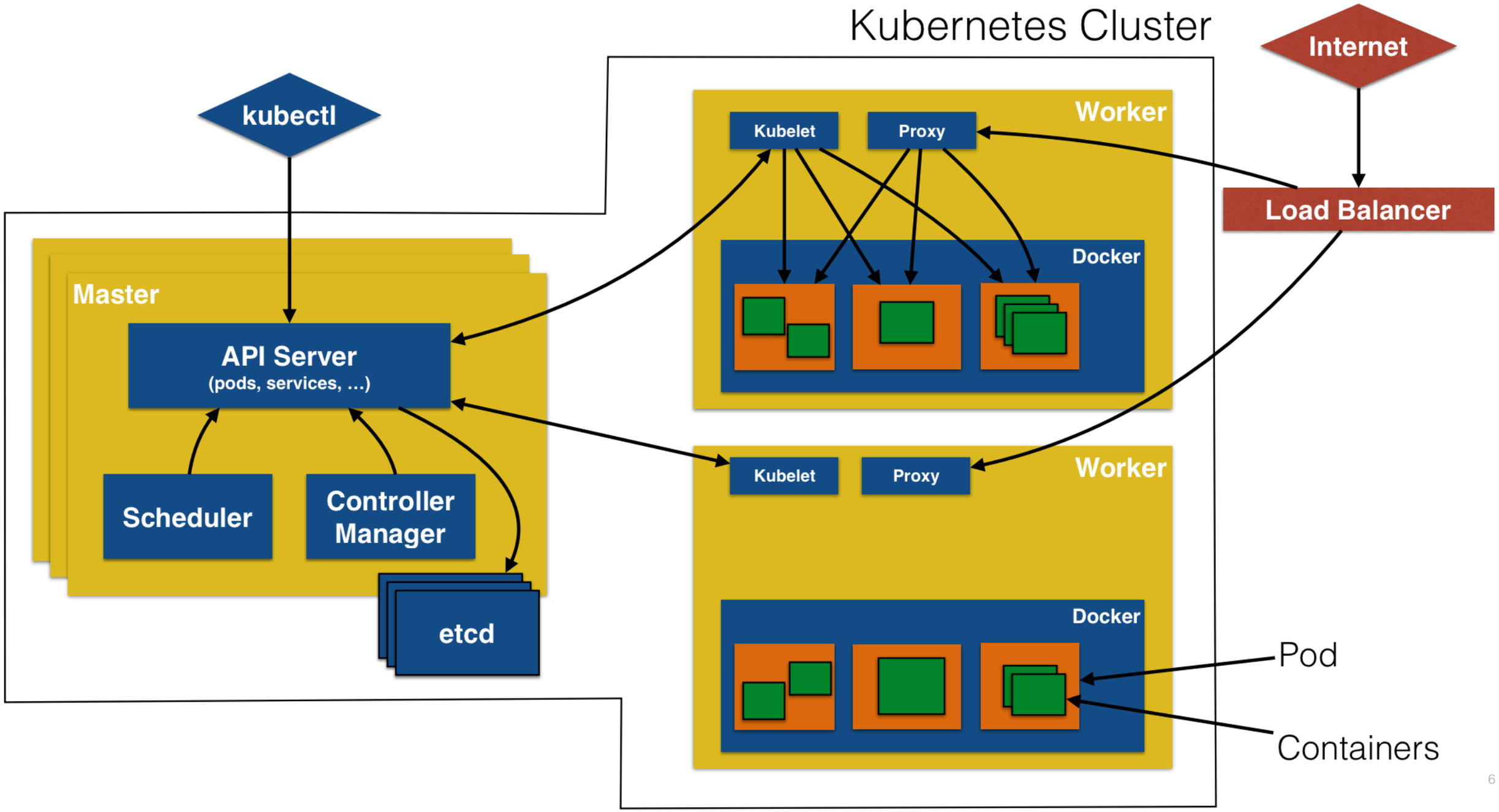
Kubernetes components

Node

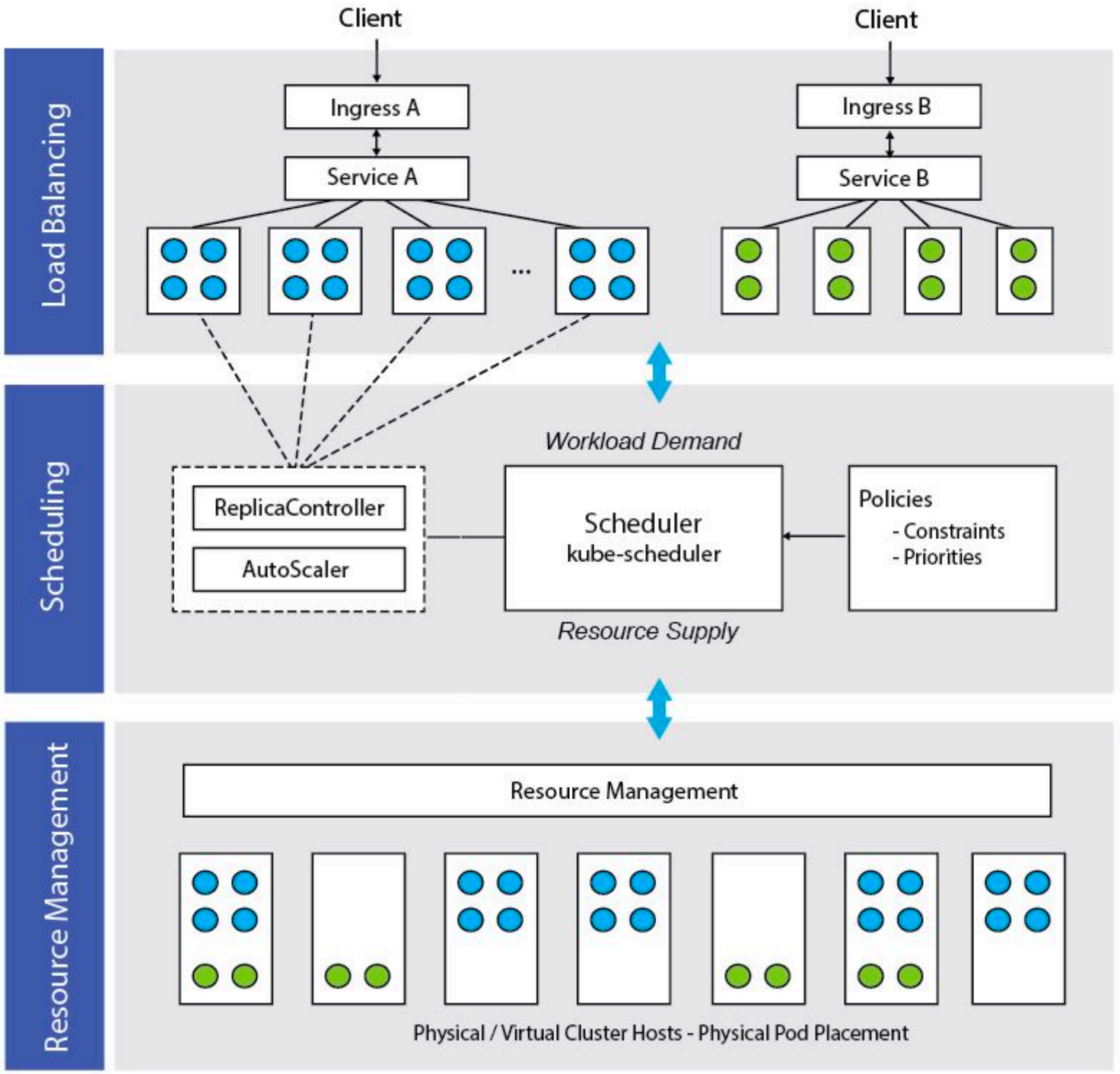
Master node

Worker node

Kubernetes components



Kubernetes Container Orchestration



Kubernetes Scheduling Algorithm

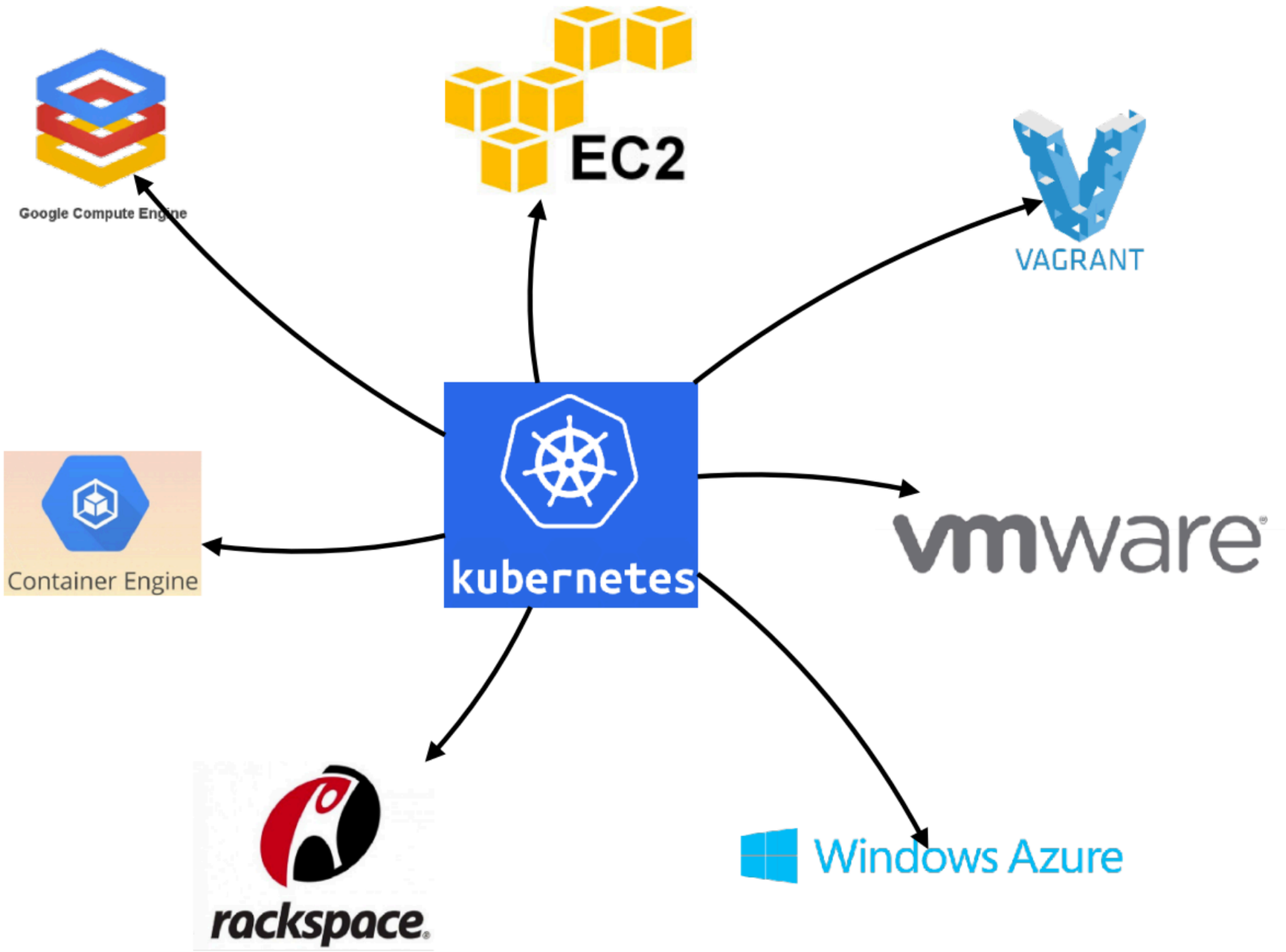
kubectl

Command-line
Control the Kubernetes cluster manager
CRUD Kubernetes resources

kubectl

$kubectl create
$kubectl get
$kubectl describe
$kubectl delete

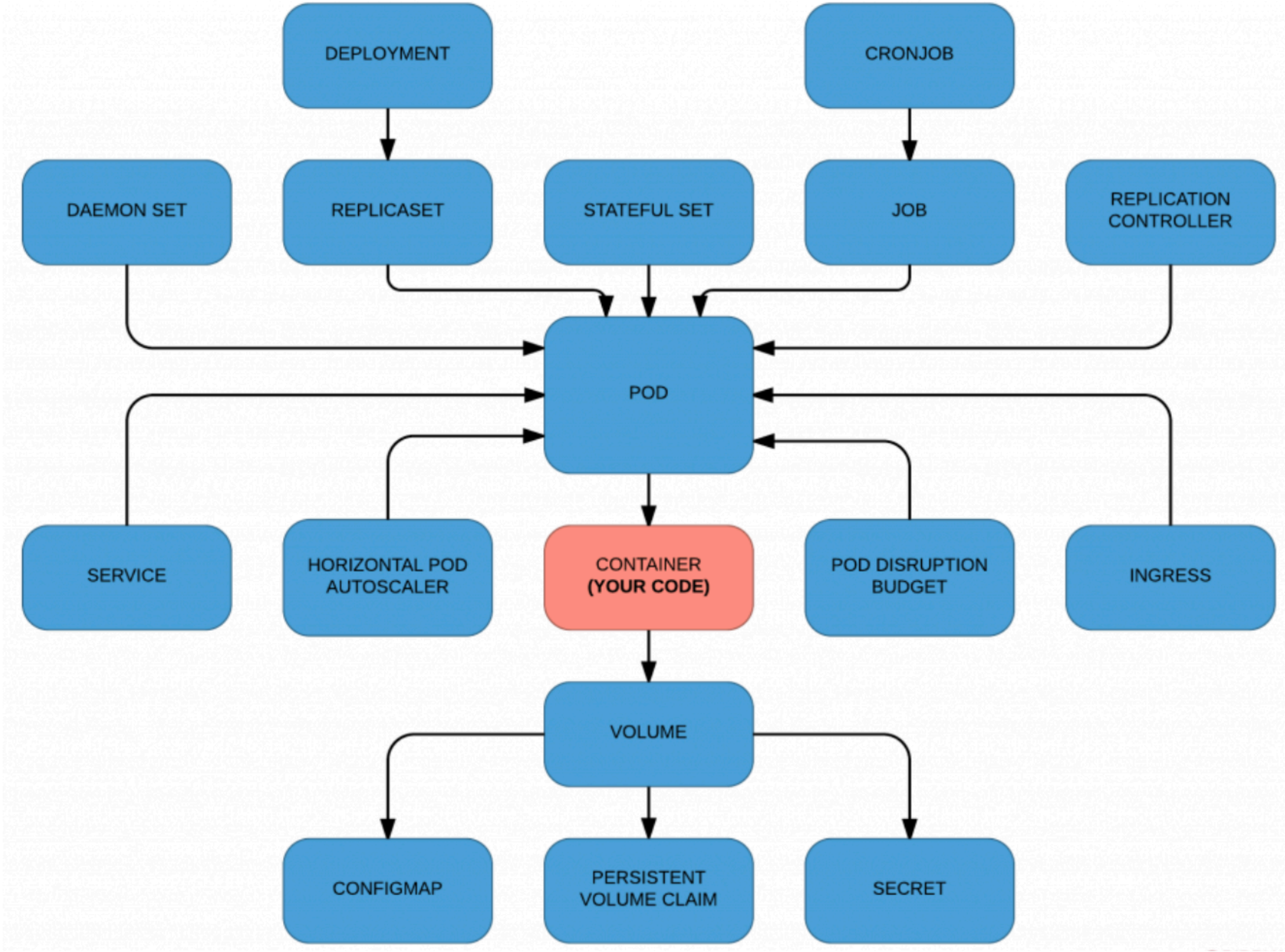
Kubernetes on Cloud



Let’s start on Google Cloud

Workshop

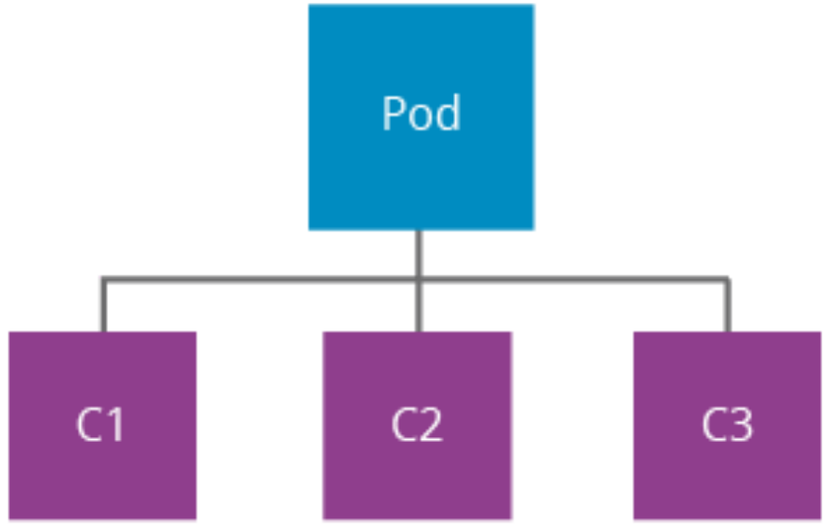
Components



Pods

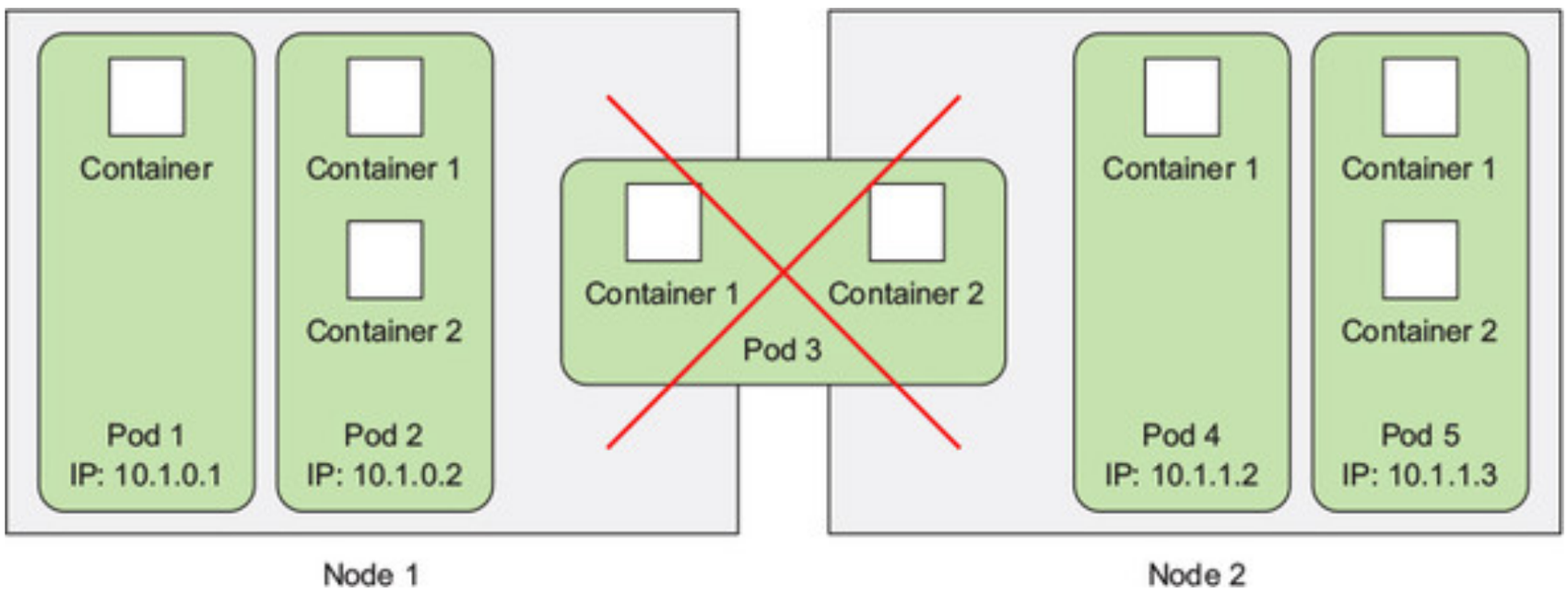
Pods

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



Pods

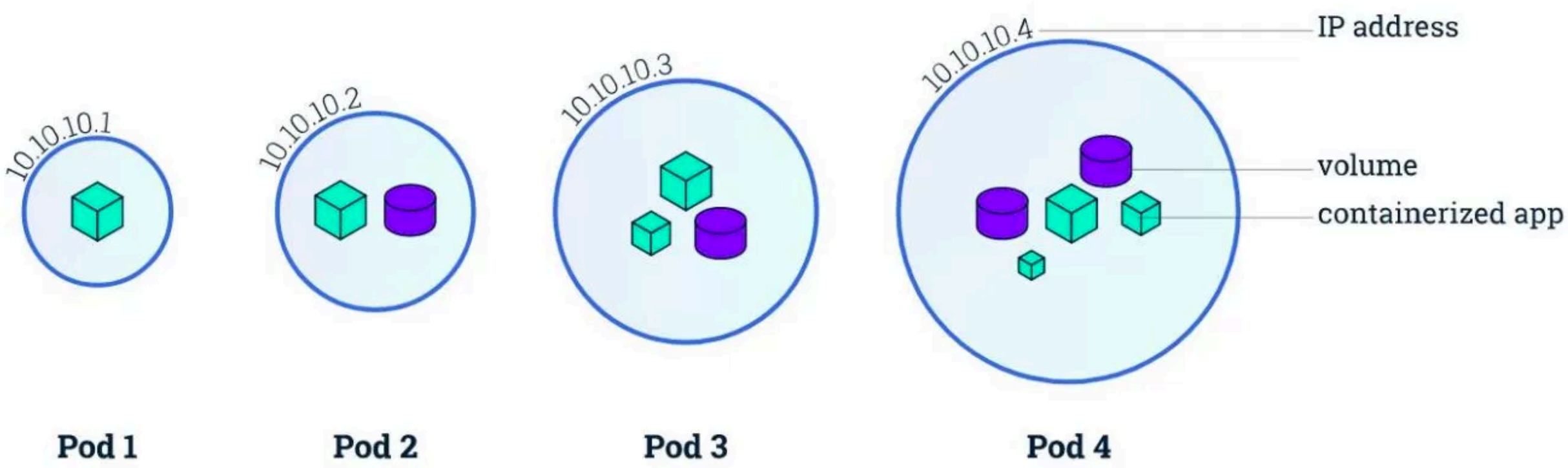
1 pods = 1 container
1 pods = N containers



All containers in same Pods

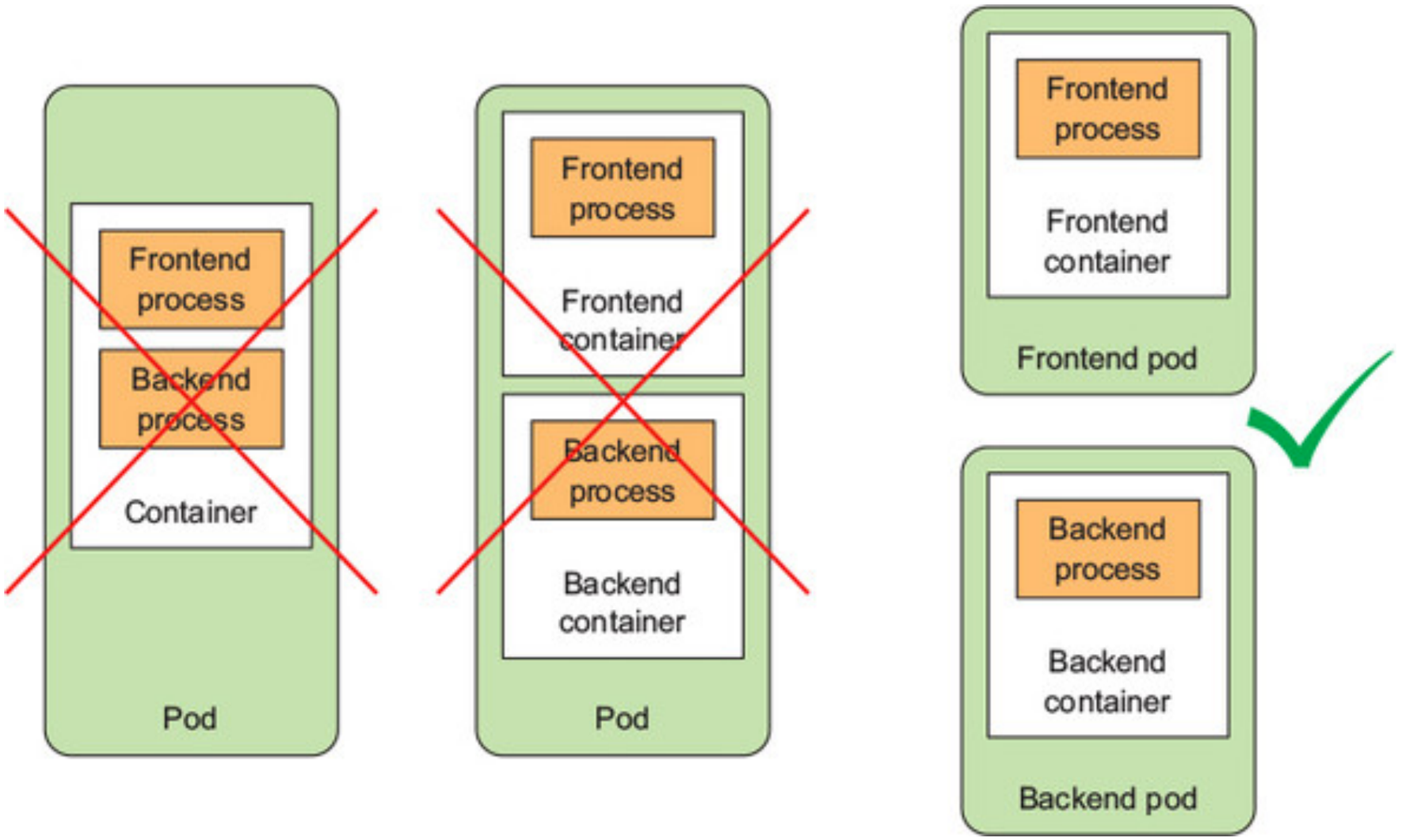
Share process ID
Share network interface
Share Hostname/IP/Port
Share Unix Time Sharing (UTS)

Pods



Organize container across Pods

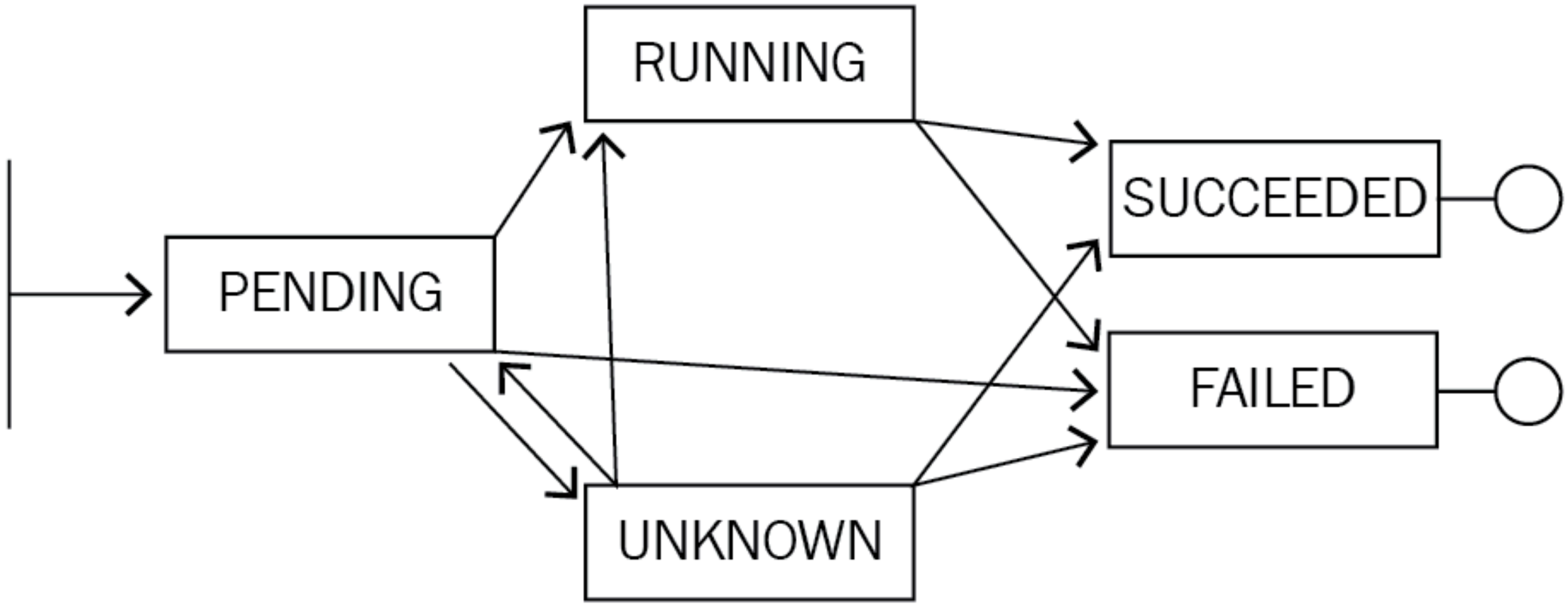
Split multi-tiers app into multiple pods



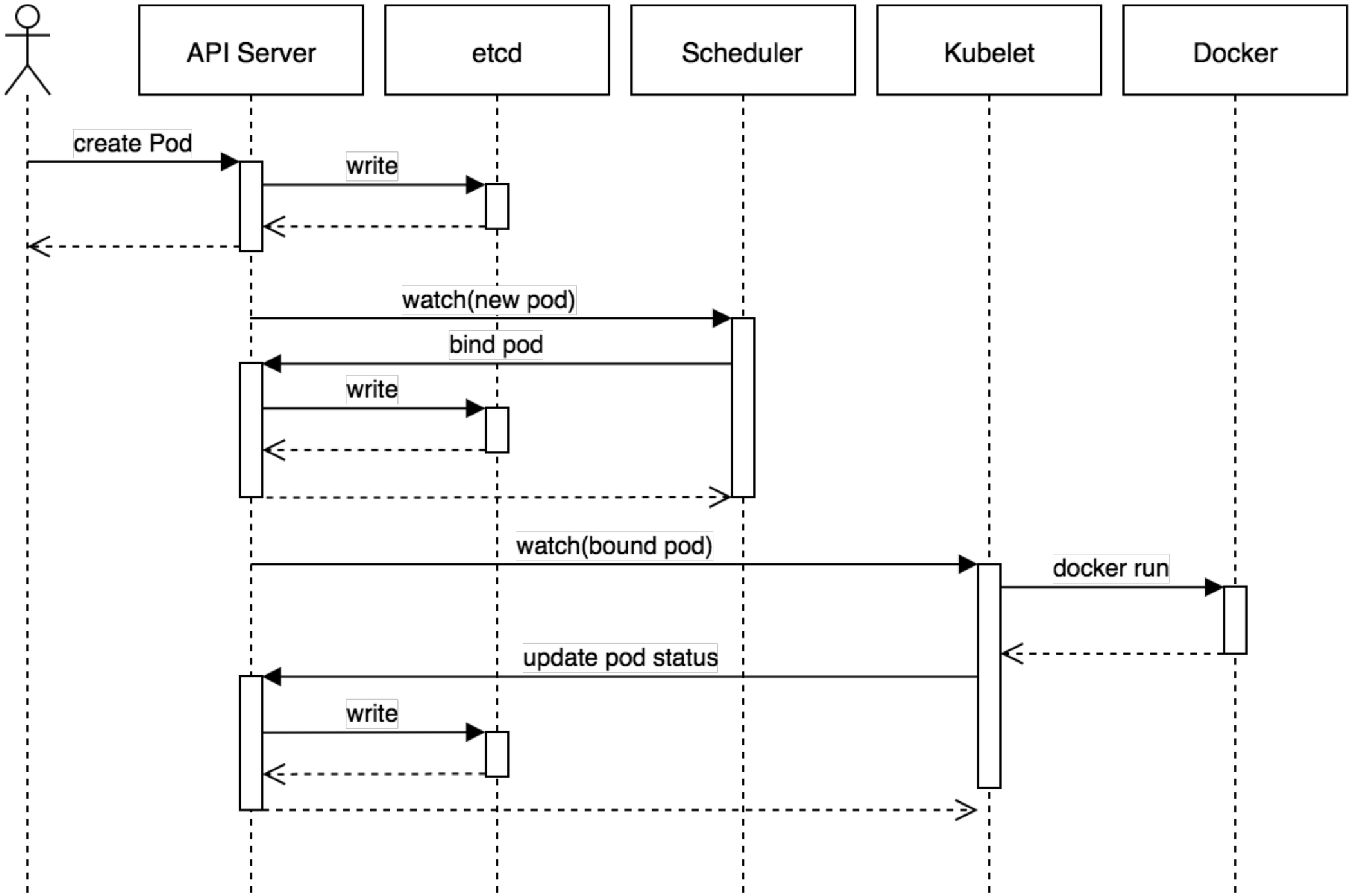
Pods lifecycle

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/starting/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



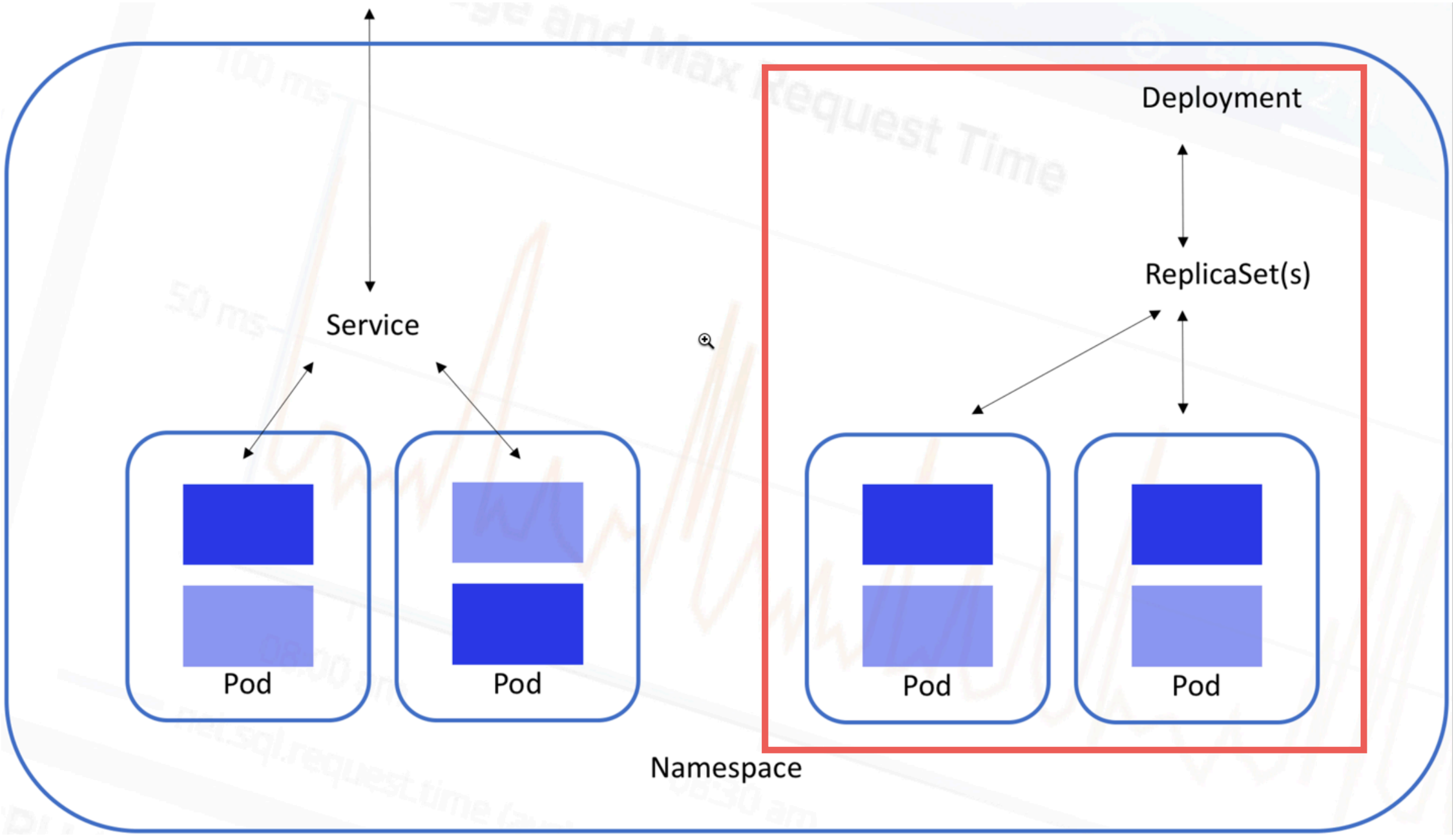
How to create Pods ?



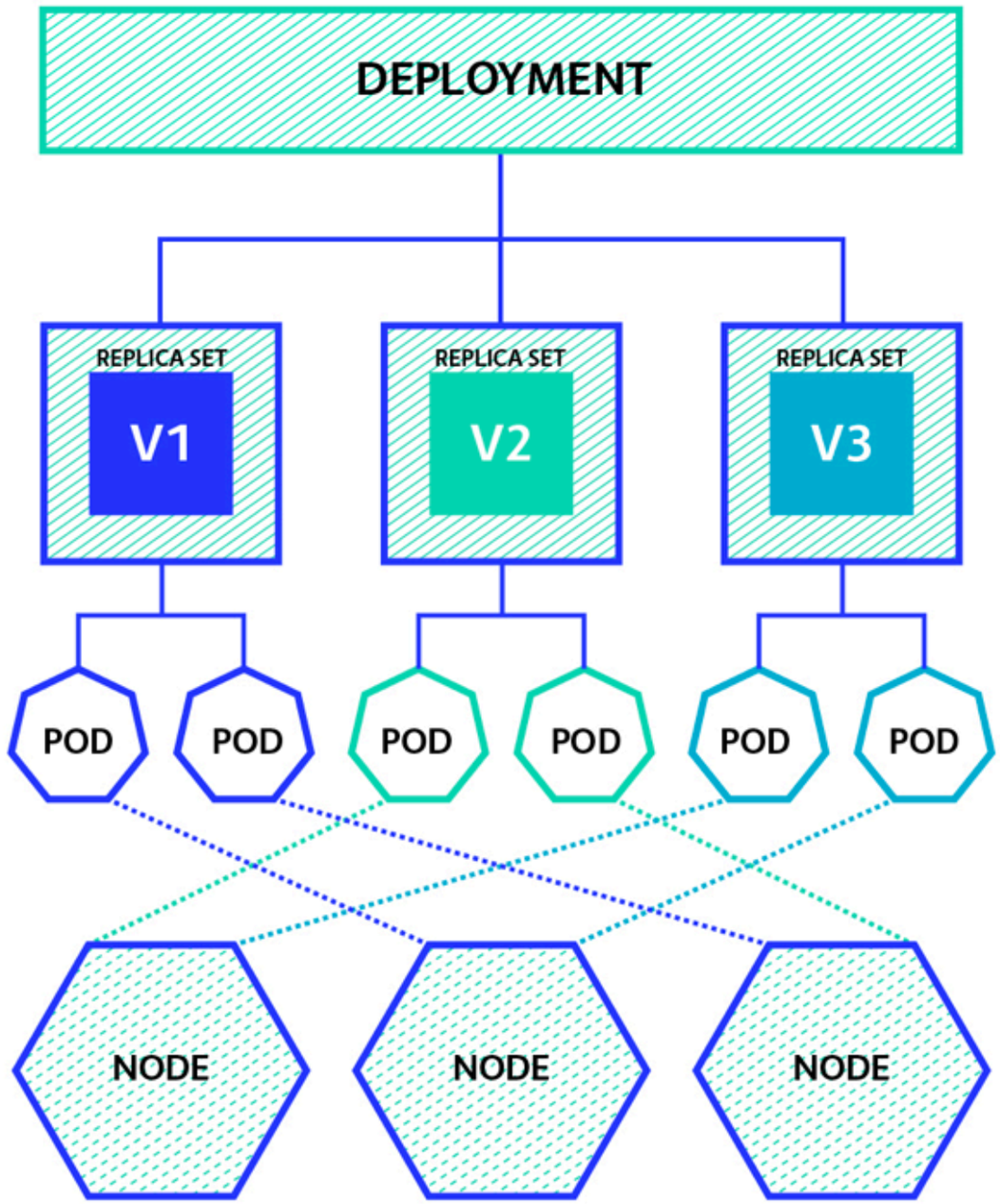
https://blog.heptio.com/core-kubernetes-jazz-improv-over-orchestration-a7903ea92ca

Deployment

Deployment and ReplicaSet



Deployment and ReplicaSet



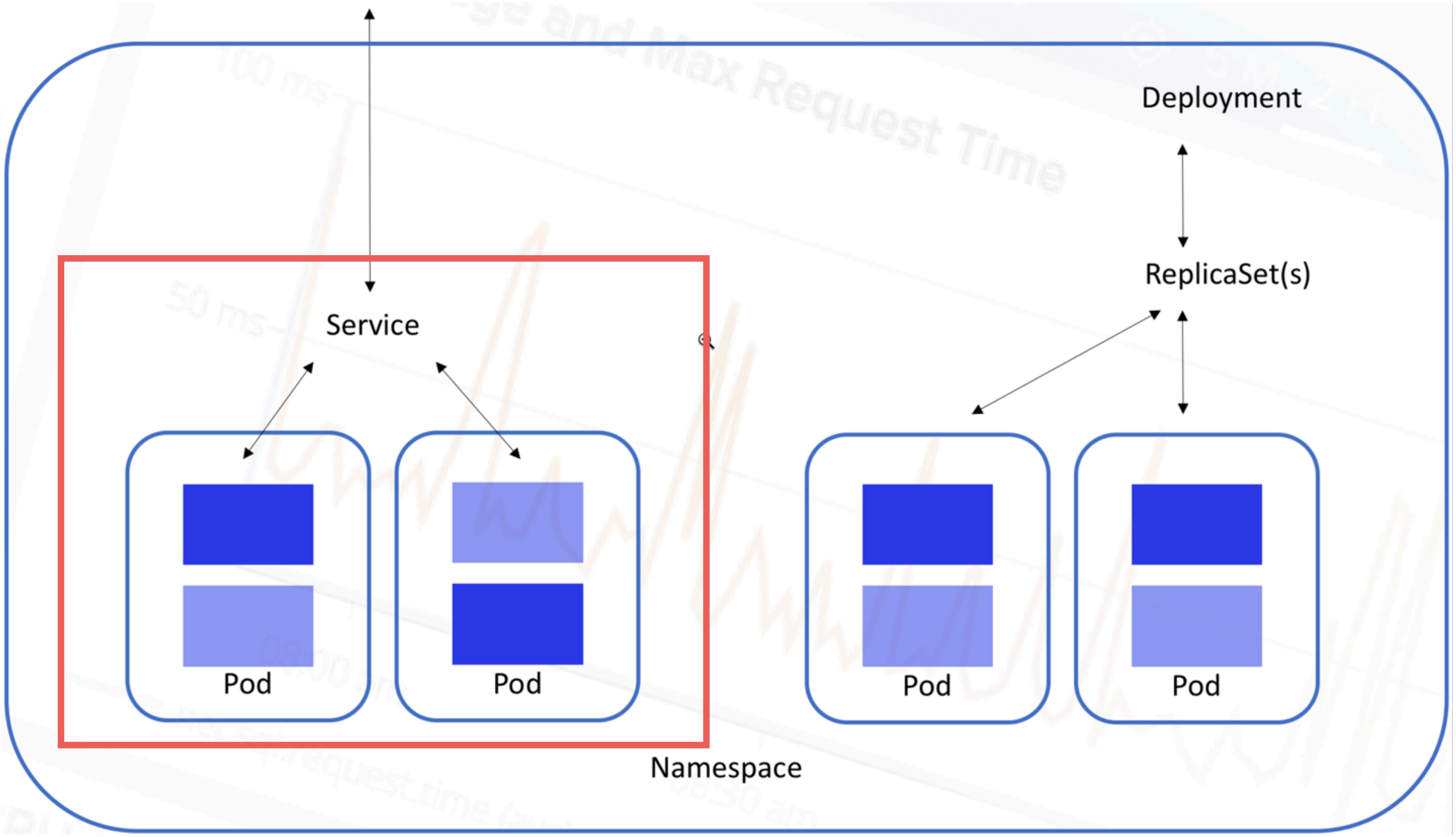
https://thenewstack.io/kubernetes-deployments-work/

Deployment and ReplicaSet

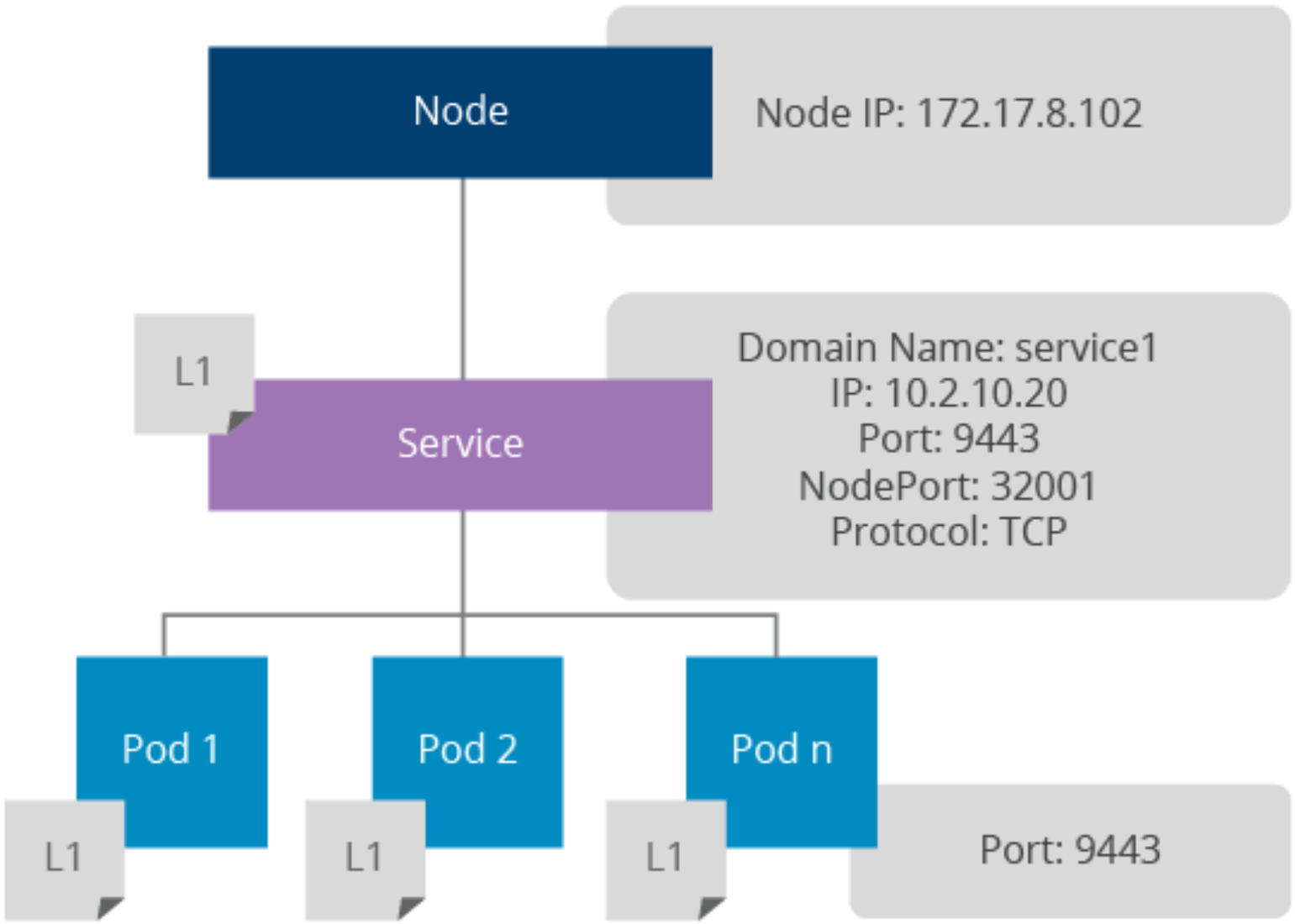
Next-generation of RC
Provide function to maintain versioning of Pods
Update new version (Rollout)
Revert to old version (Rollback)
Scale a deployment
Pause/Resume process

Services

Services



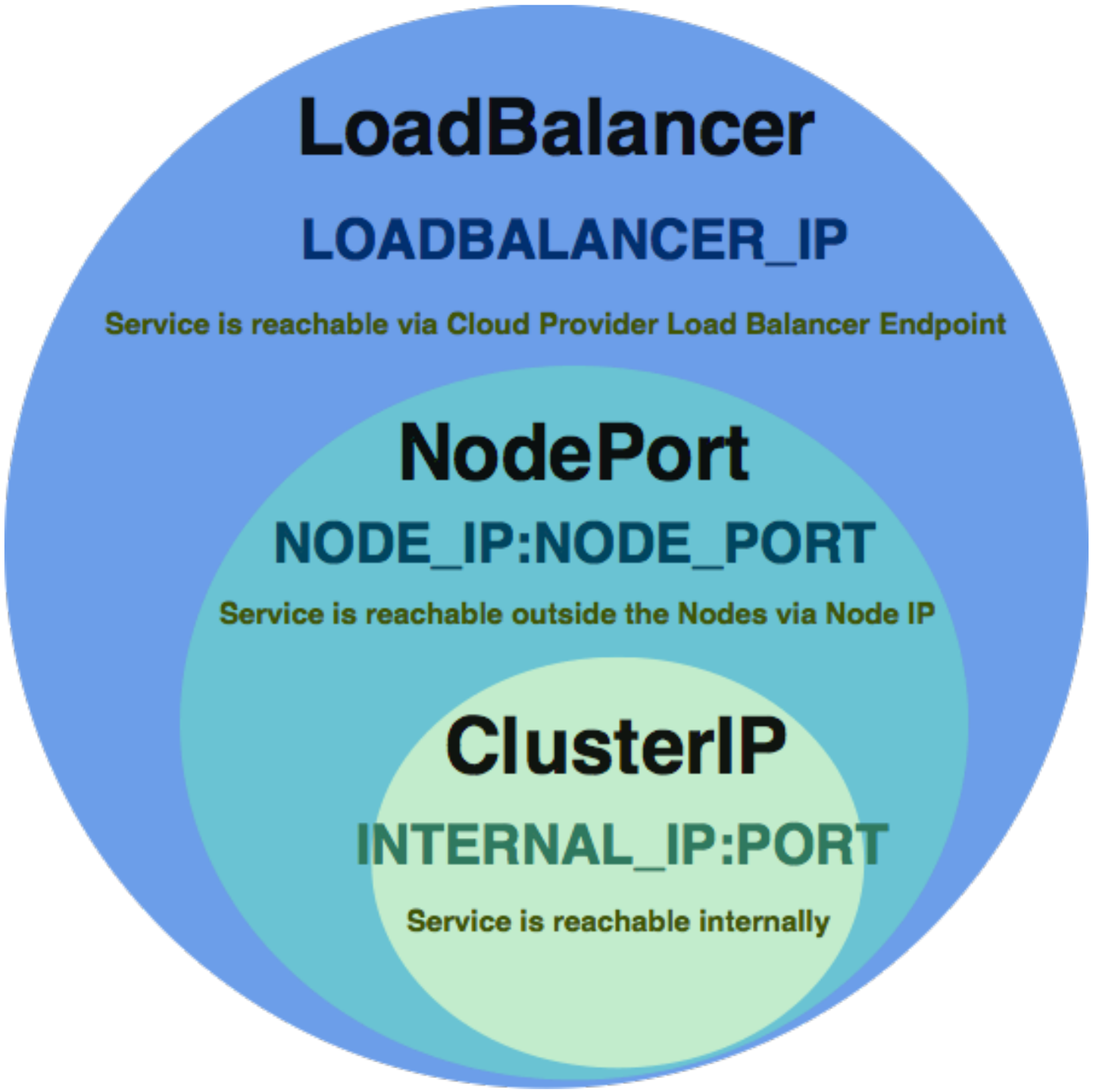
Services



Services

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

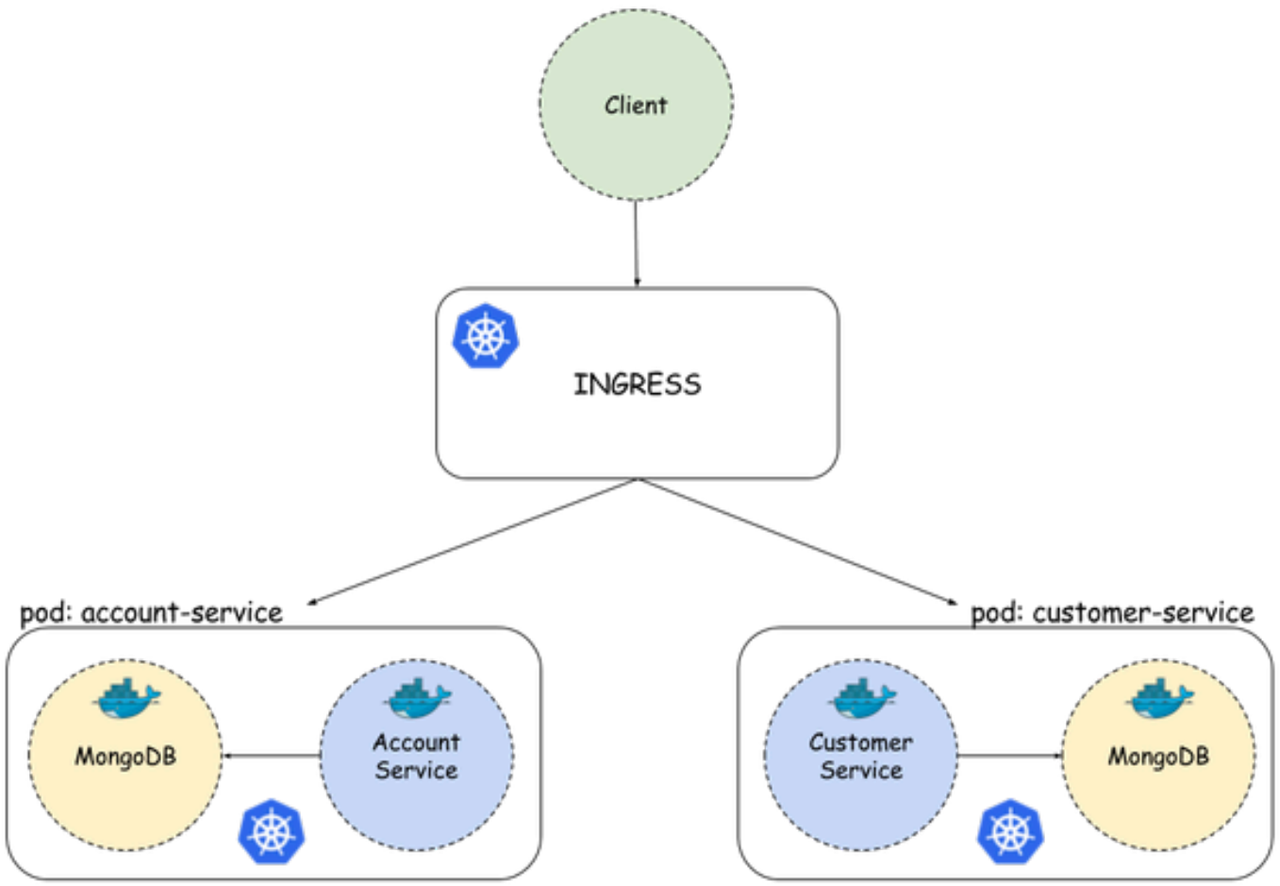
3 types of services



Ingress controller

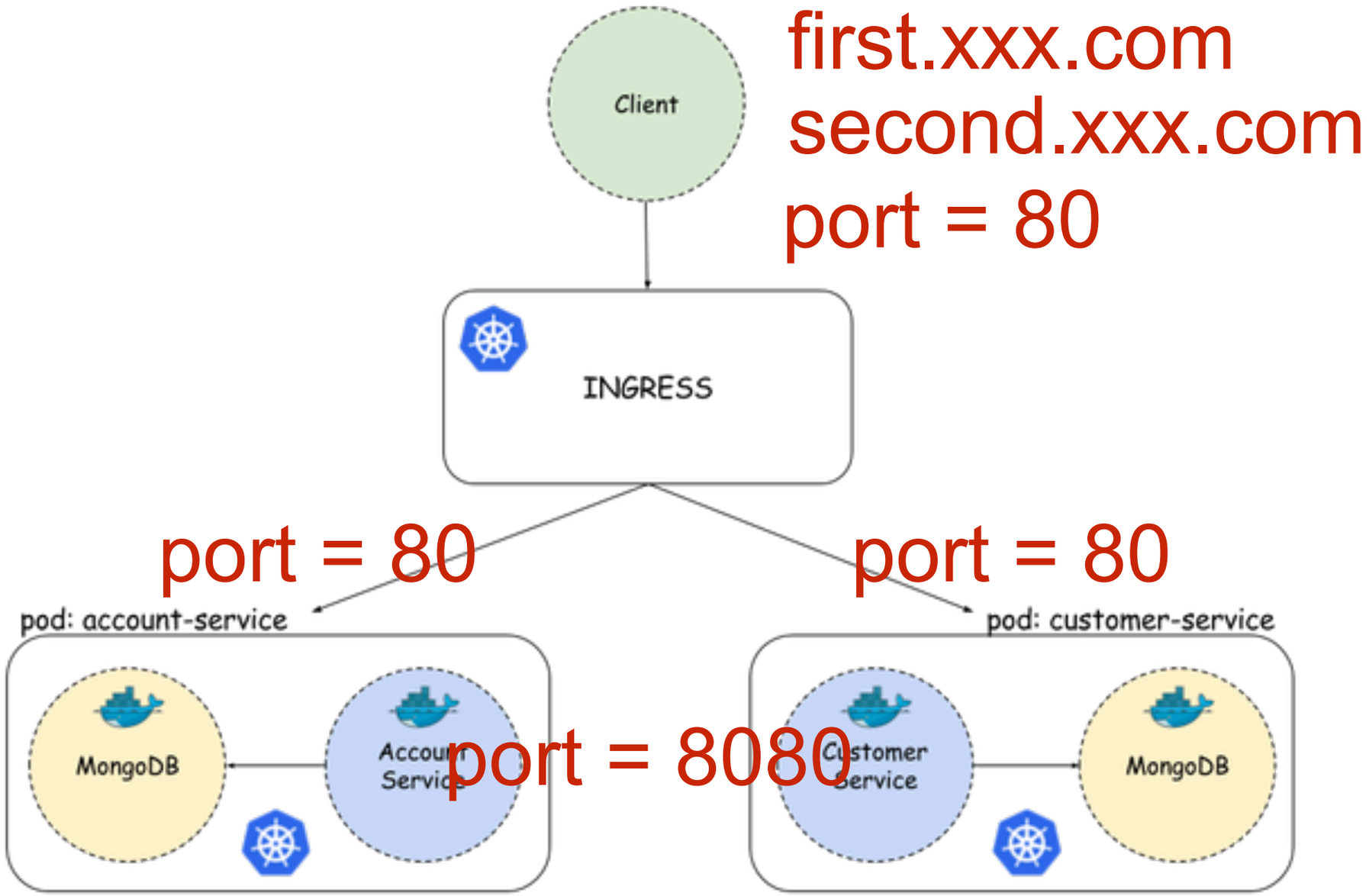
Ingress controller

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

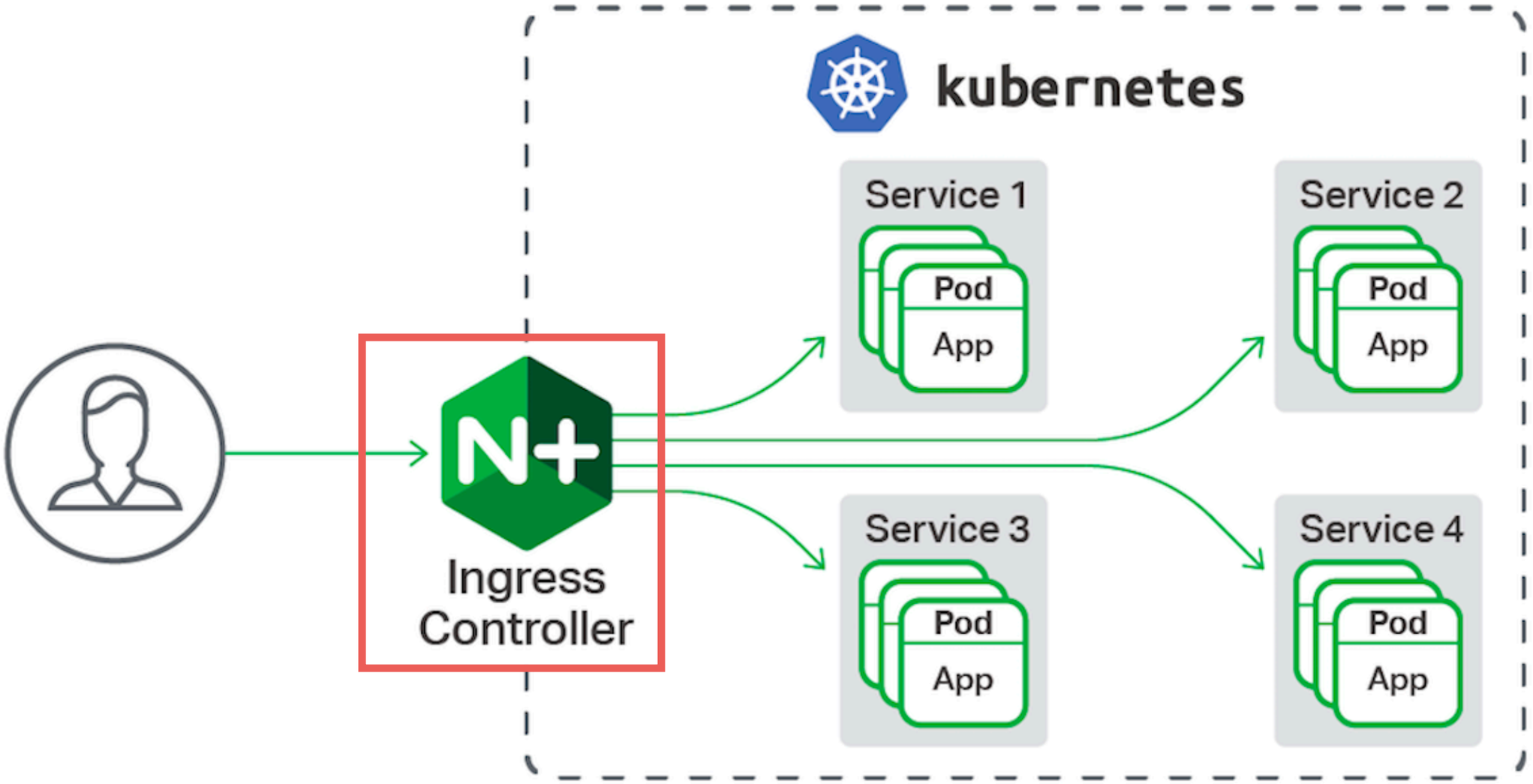


https://kubernetes.io/docs/concepts/services-networking/ingress/

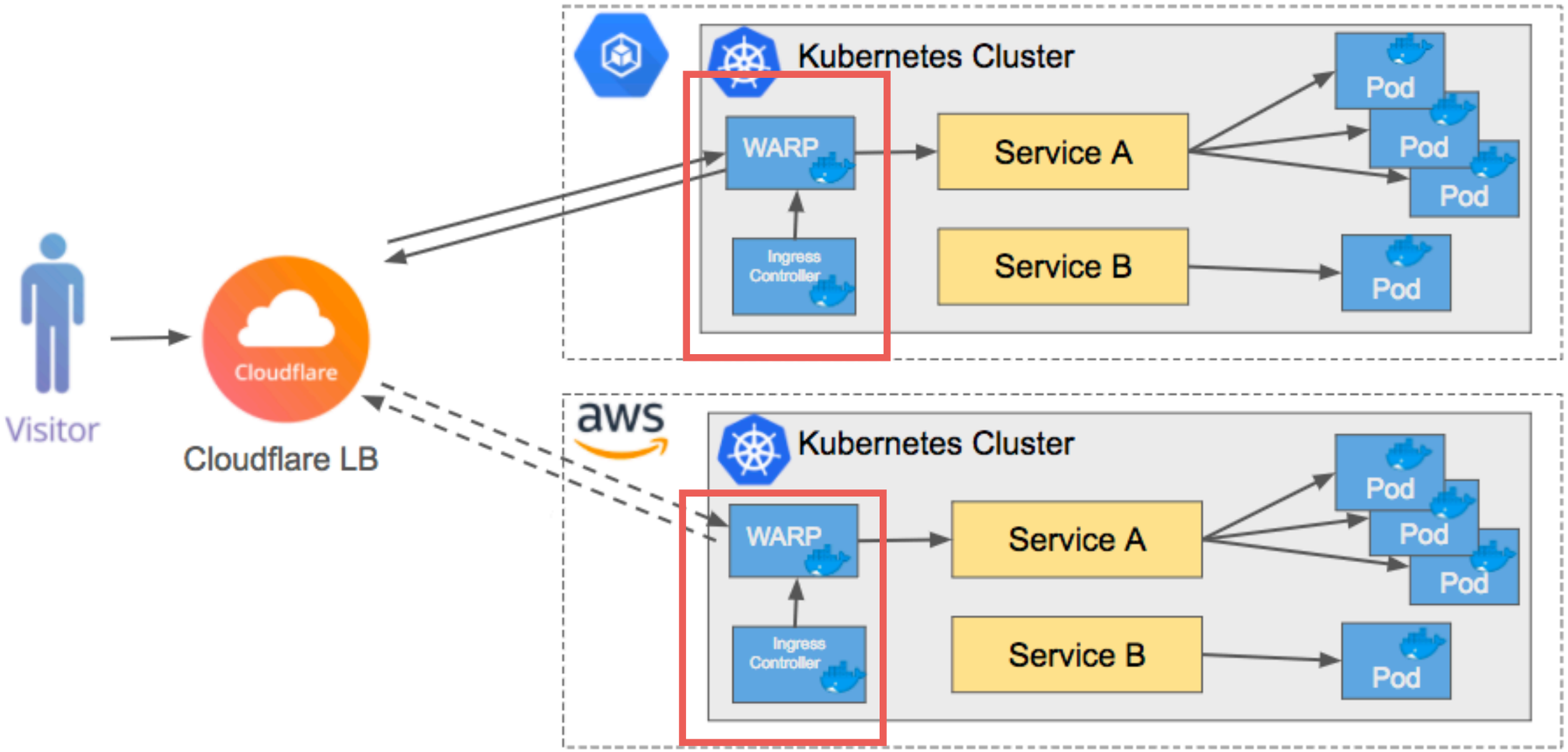
Ingress controller



Ingress Network



Ingress Network



ConfigMap and Secret

ConfigMap

Configuration parameters
Key-value pairs

Environment variables

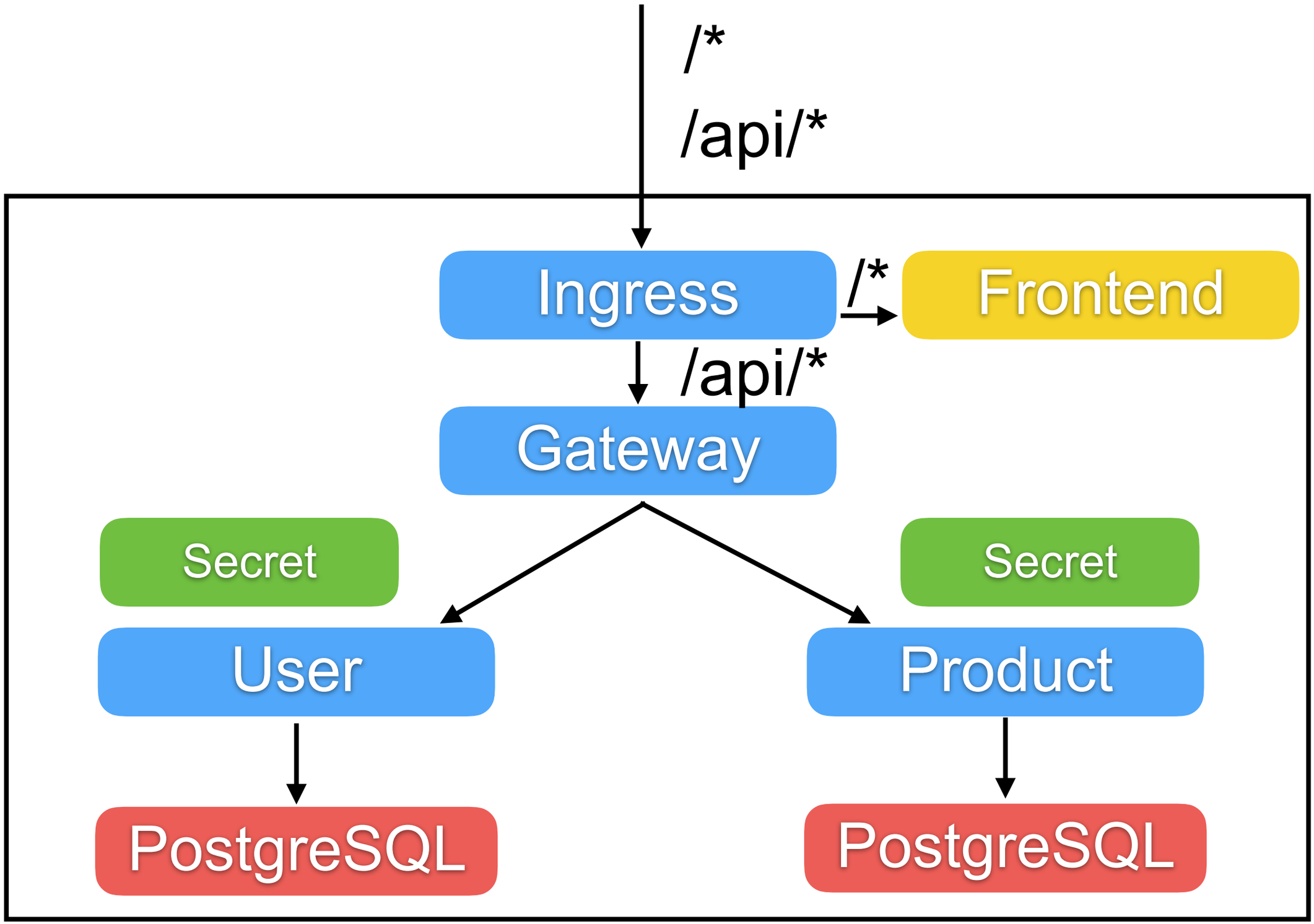
Container command line arguments

Volume

Secret

Way to Kubernetes to distributed **credentials**
keys, **passwords** or **secret data** to Pod
Native way of Kubernetes
Other way use **external vault services**

Structure of service



Workshop

https://github.com/up1/demo-docker-k8s

Application Deployment Strategies

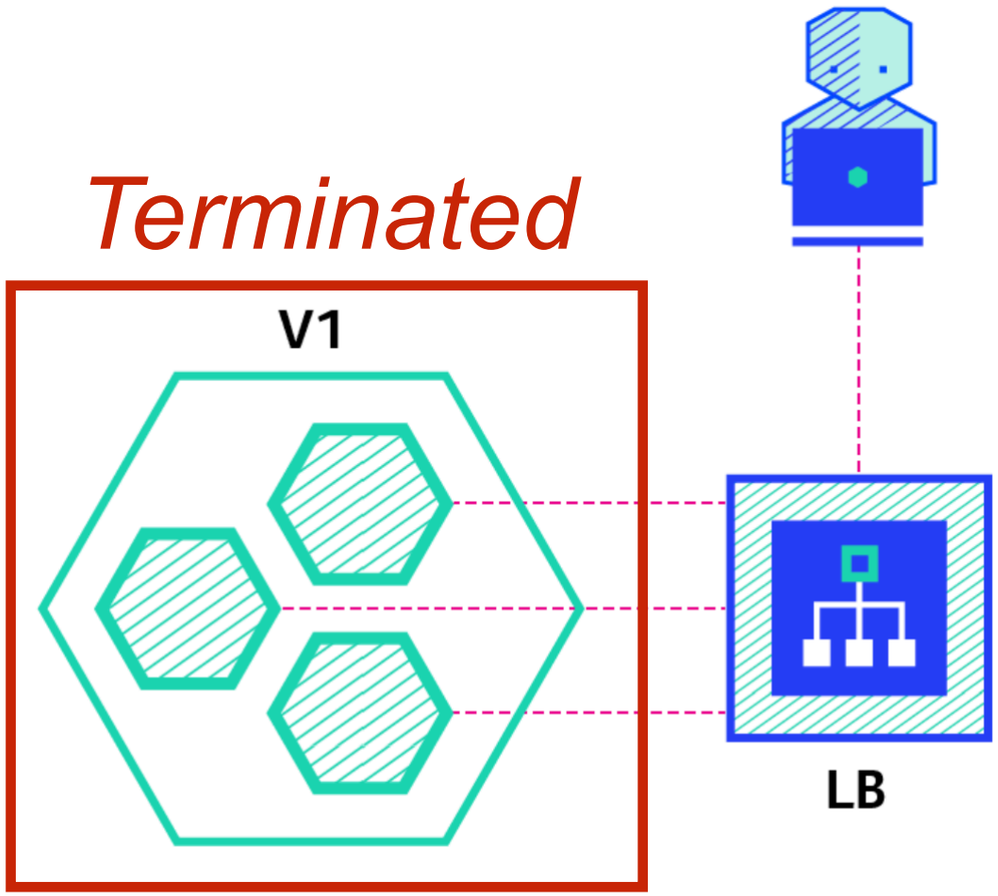
Strategies to deploy

Recreate
Ramped
Blue/Green
Canary
A/B testing
Shadow

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

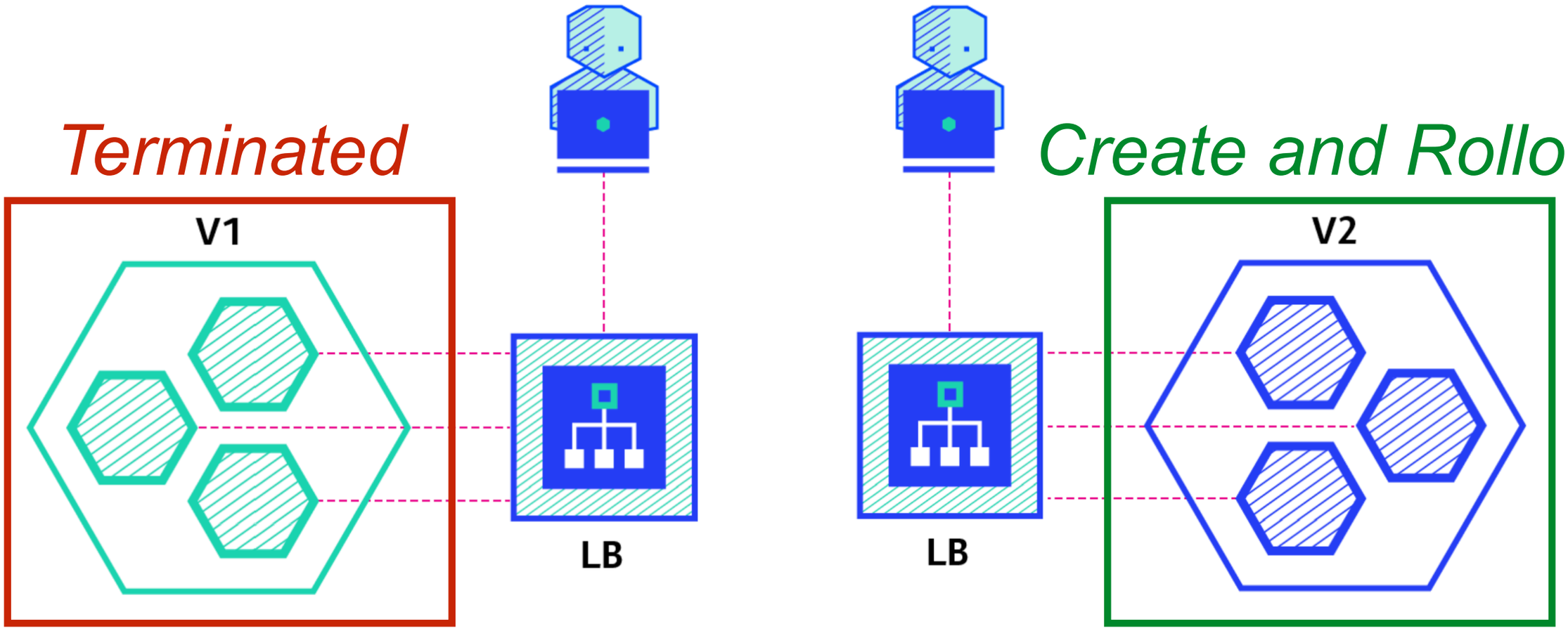
1. Recreate

Version A is terminated then version B is rollout



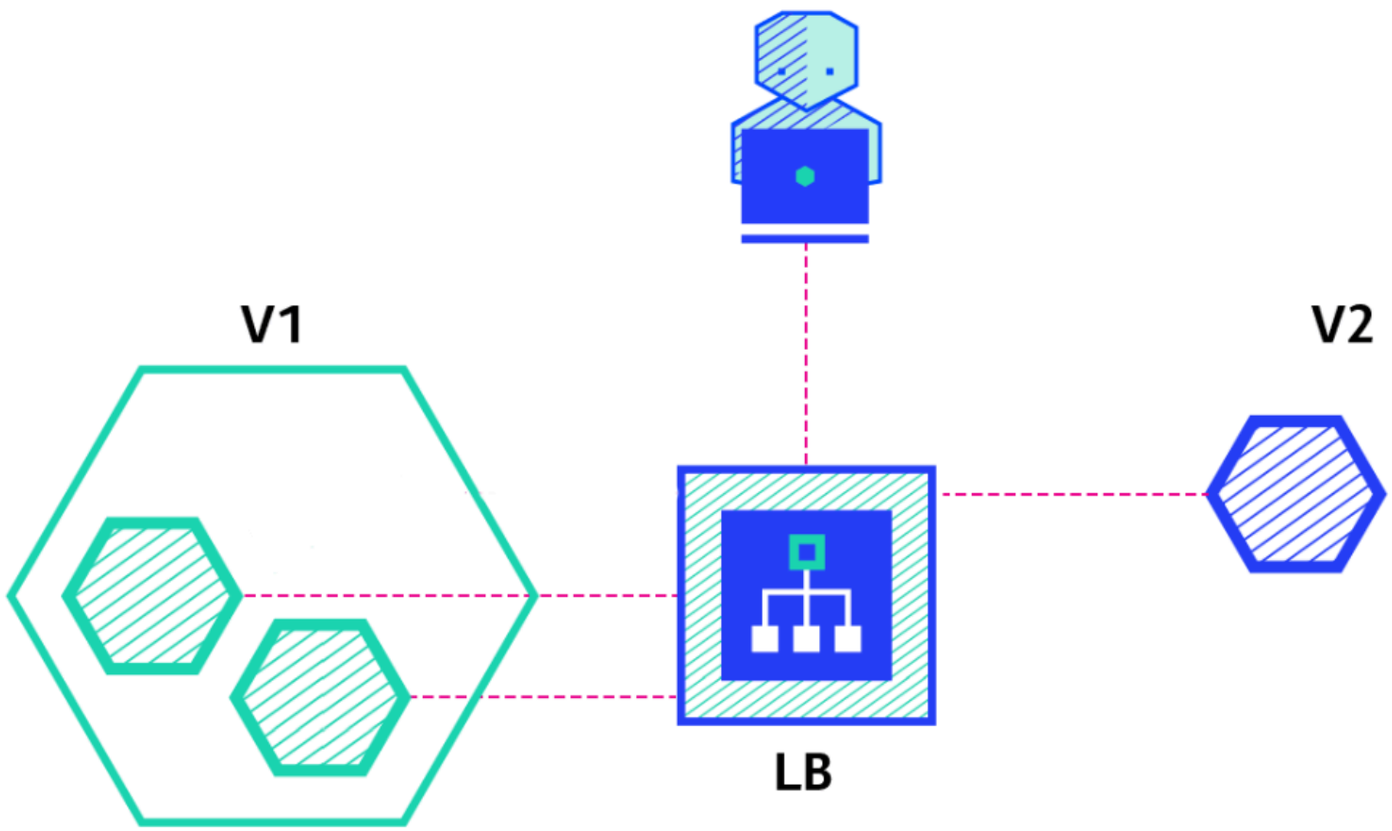
1. Recreate

Version A is terminated then version B is rollout



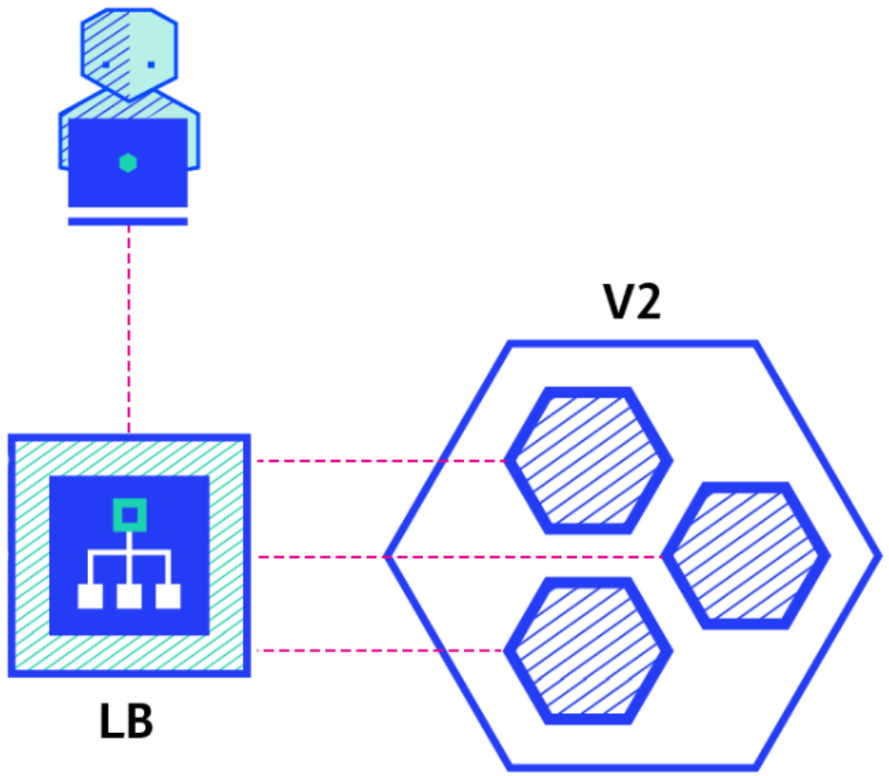
2. Ramped

Slow roll out by replace instance one-by-one



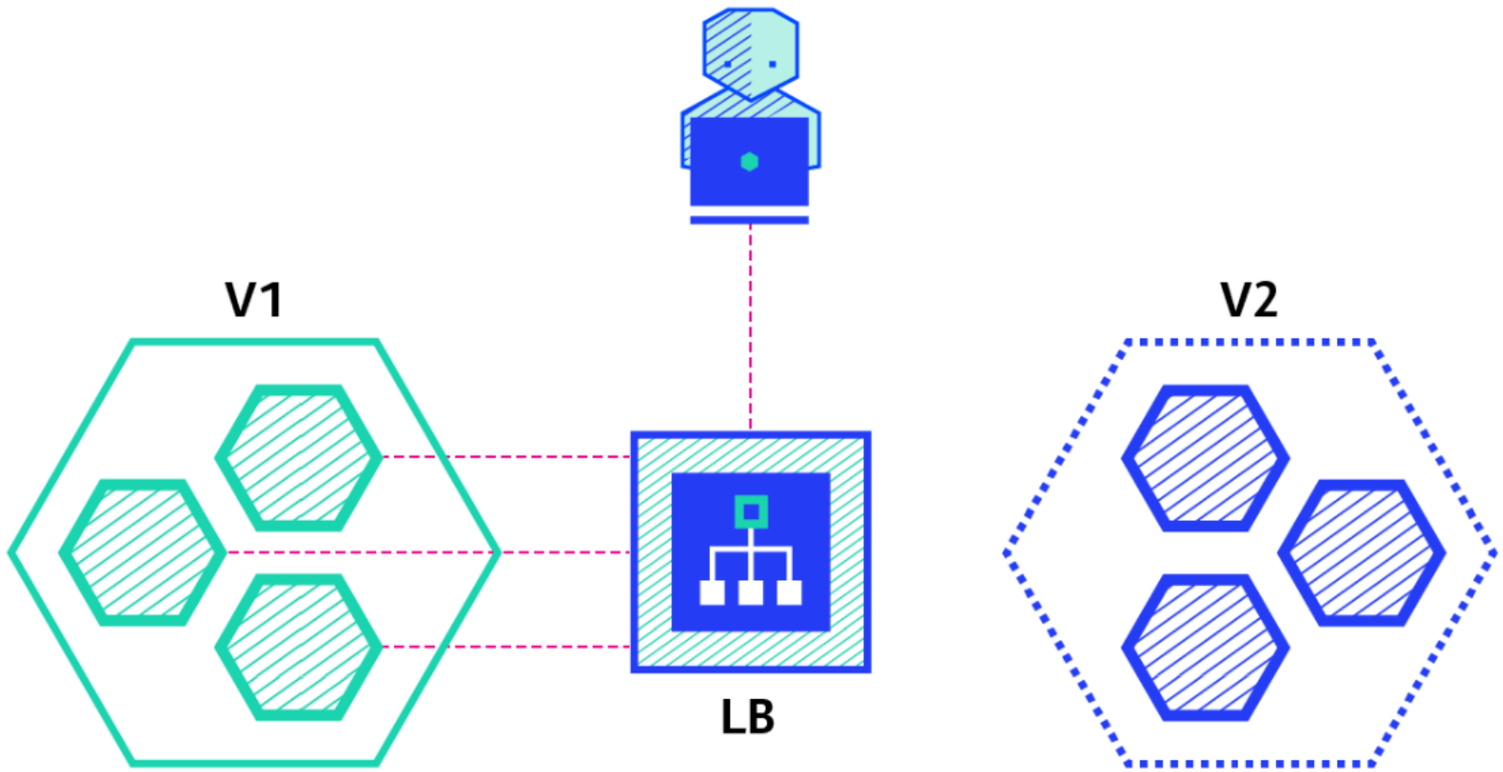
2. Ramped

Slow roll out by replace instance one-by-one



3. Blue/Green

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

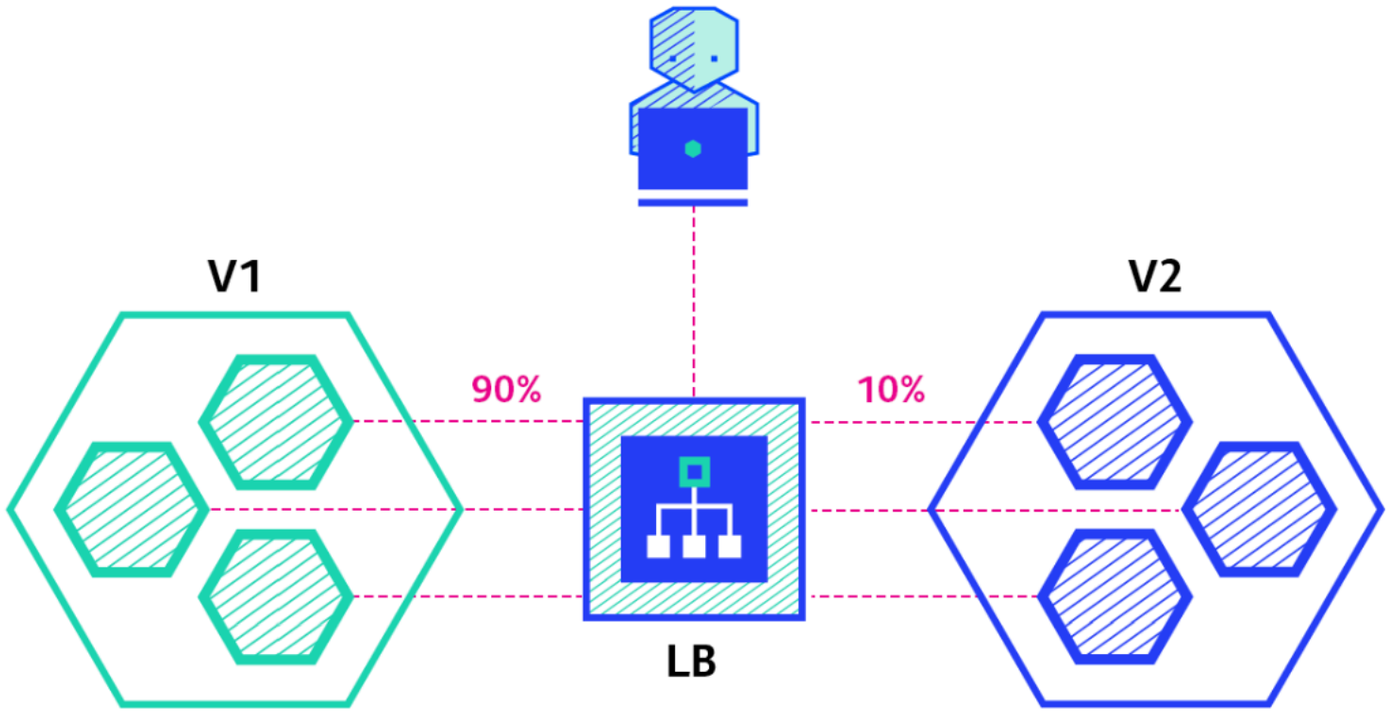


4. Canary

Shift production traffic from version A to B

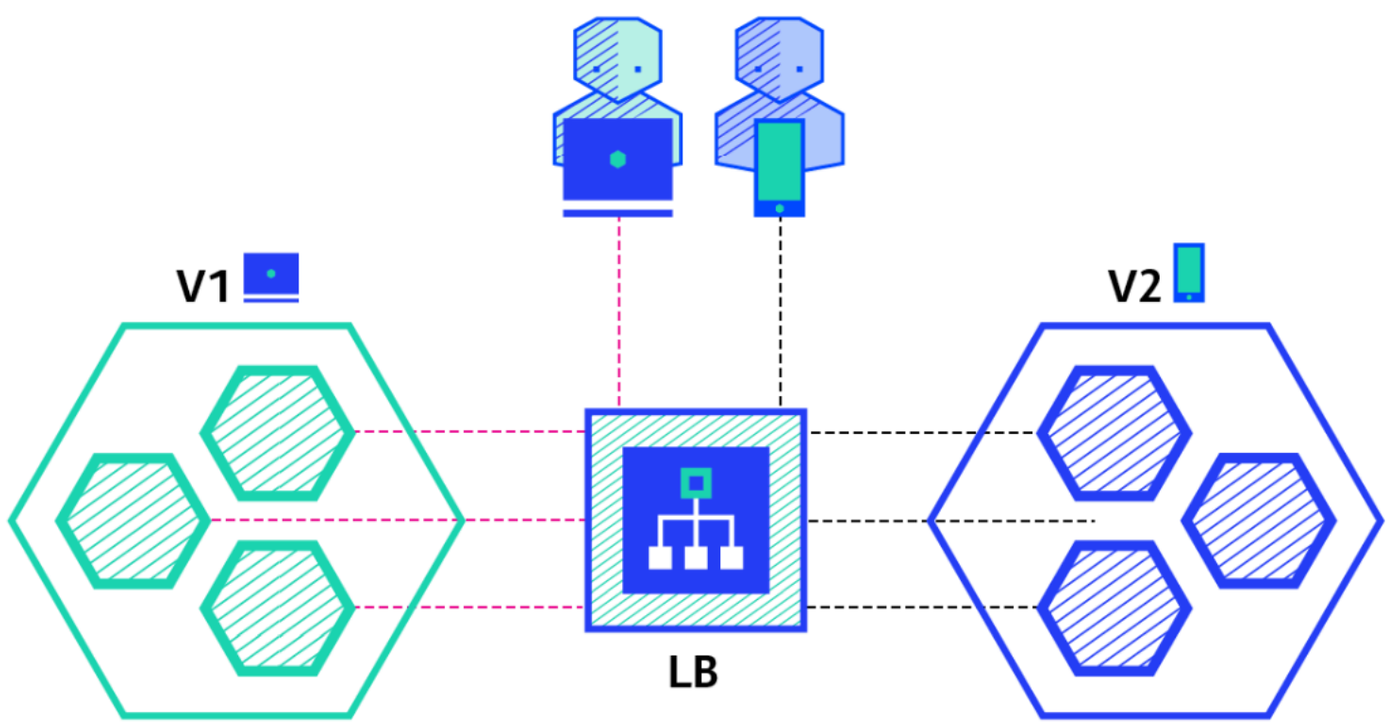
Traffic is split based-on weight

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



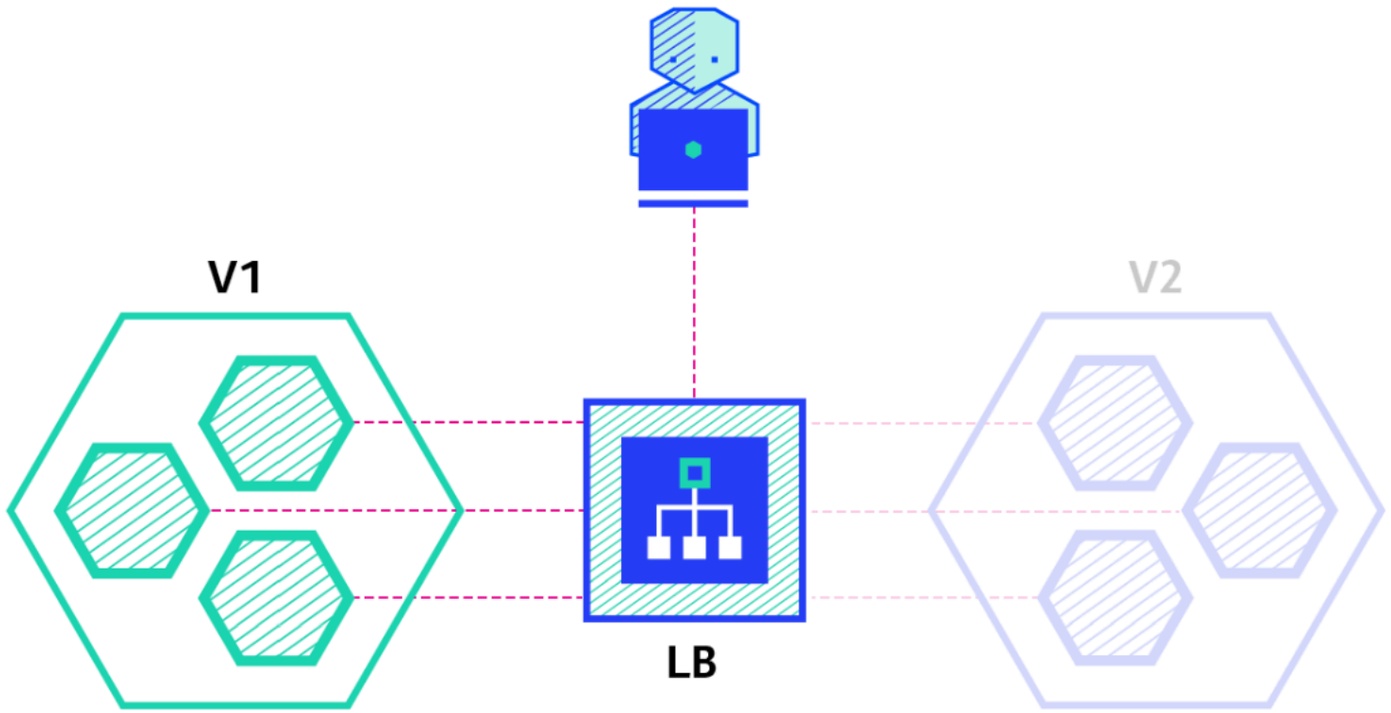
5. A/B testing

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	✓	✓	✗	■■■	□□□	□□□	■■■