

# Kubernetes KBTG

- Training Day -

Get Slide



Jumpbox®

เจ้าของลิขสิทธิ์ อนุญาติให้ใช้รูปด้านล่างเพื่อการเรียนรู้ส่วนบุคคลเท่านั้น และขอสงวนลิขสิทธิ์ ห้ามมิให้เผยแพร่ในที่สาธารณะ ผู้ละเอียด จะถูกดำเนินคดีตามกฎหมาย



# Jumpbox®

Tech Passion | Sharing | Society

"เราเชื่อว่า การเรียนรู้ทำให้ชีวิตคุณดีขึ้น"

-Jumpbox Team-



**JoJo**  
(Cloud Native Stylist)



**Pae**  
(Full-stack Developer)

เจ้าของ Page [jitrak.dev](https://jitrak.dev)



กรรมการ และประธานคณะกรรมการ

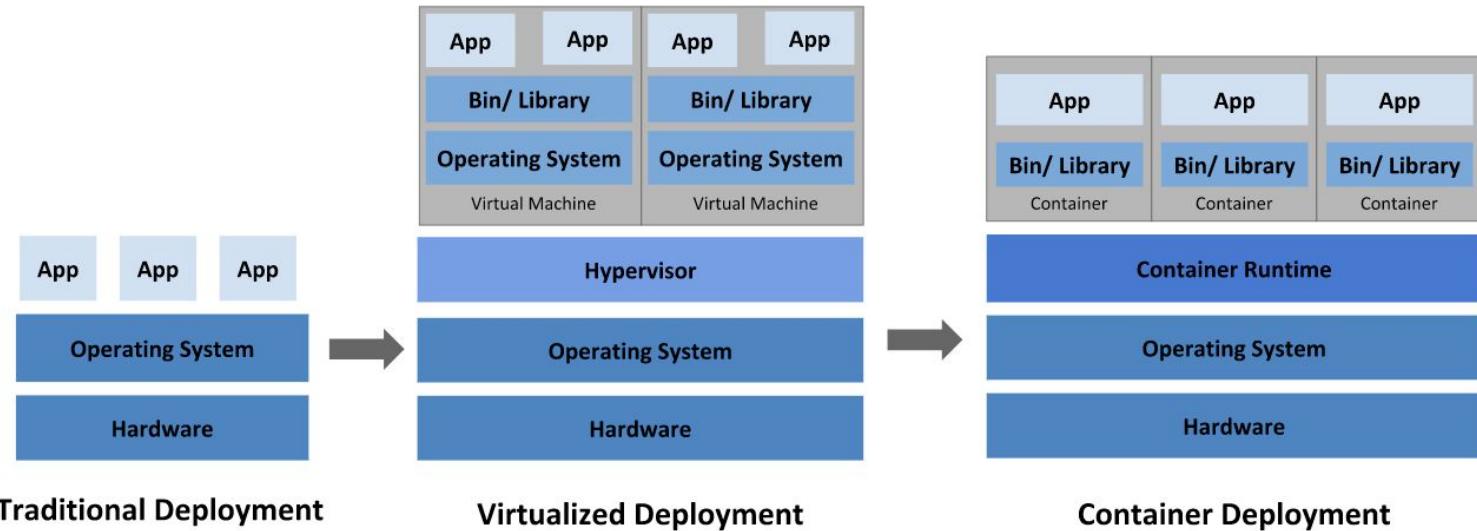


# Agenda

## Day 1

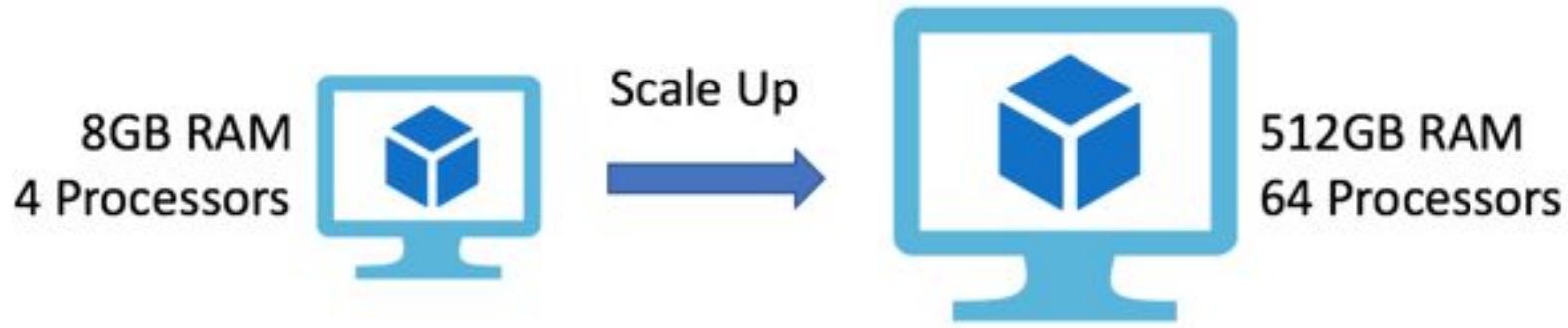
- Evolution of deployment
- Introduction to Orchestrator
- Kubernetes Overview
- Kubernetes Objects
- Kubernetes API
- Workload Objects
- The Tools for interaction with Kubernetes
- Kubernetes Architecture (Basic)
- Kubernetes Component (Basic)
- Kubernetes Context
- Kubernetes Object Management
- Pod
- Deployment Strategy
- Service
- Ingress
- Kubernetes Architecture
- Command and Arguments
- Environment Variables
- ConfigMap
- Secret

# Evolution of deployment



Reference Pictures: [Overview Kubernetes](#)

# Business Growth

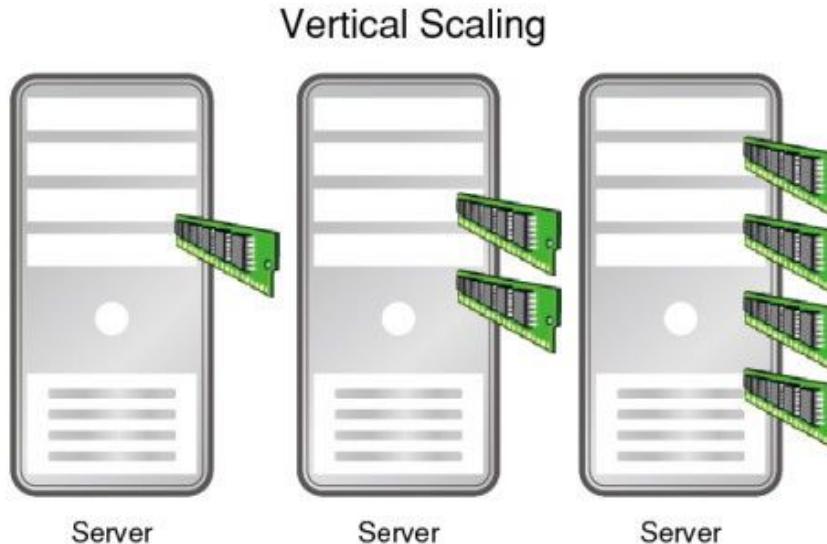


Reference Pictures:

- [Scaling Applications in the Cloud](#)

# Vertical Scaling

Performance Enhancement  
through Scale-Up

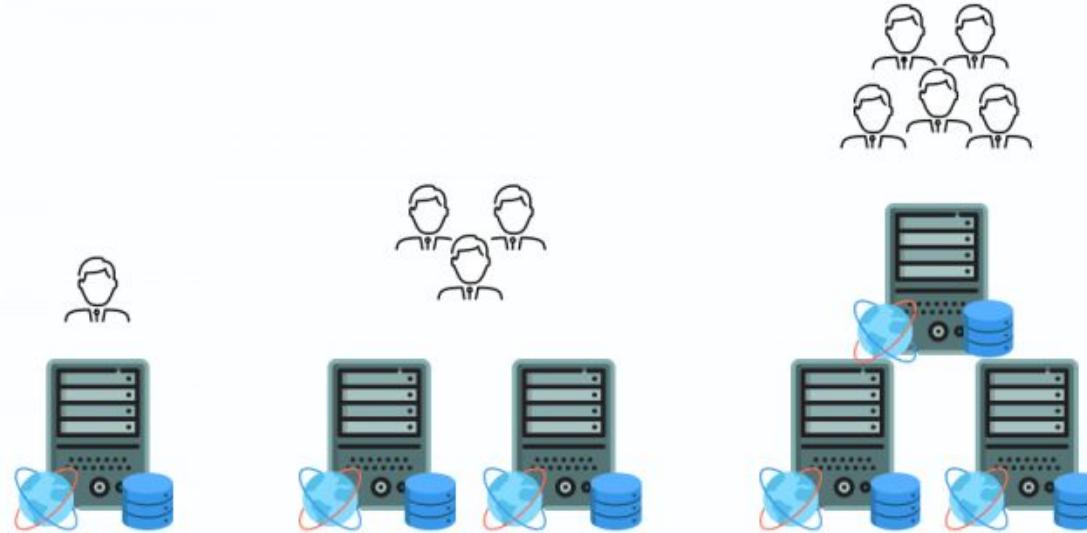


Reference Pictures:

- [Scaling a monolith vertical scaling, horizontal scaling simply defined](#)

# Horizontal Scaling

## Horizontal Scaling



Reference Pictures:

- [Scaling a monolith vertical scaling, horizontal scaling simply defined](#)

# Vertical Scaling

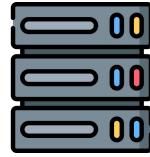
- Autoscaling -



#### Reference Pictures:

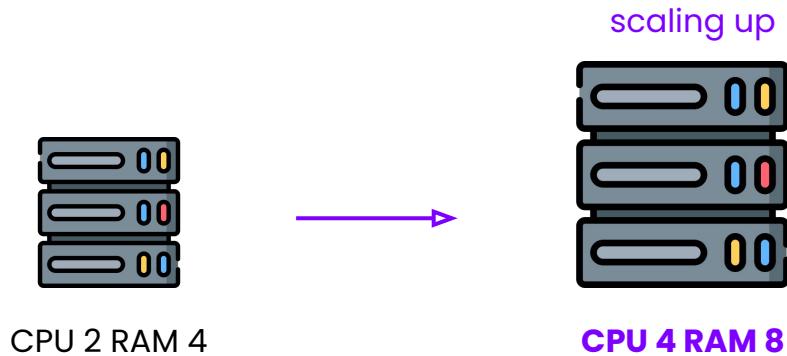
- [Tune Your Paid Search Campaigns To Account For Holiday Volatility](#)

# Vertical Scaling



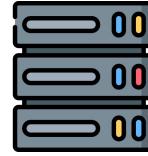
CPU 2 RAM 4

## Vertical Scaling (Cont.)

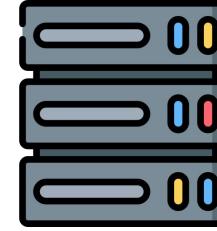


## Vertical Scaling (Cont.)

scaling down



CPU 2 RAM 4

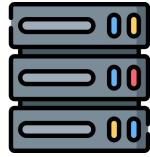


CPU 4 RAM 8

# Horizontal Scaling

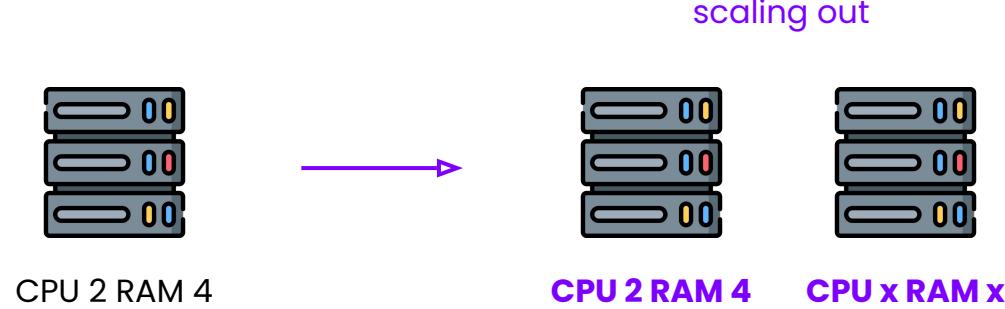
- Autoscaling -

# Horizontal Scaling



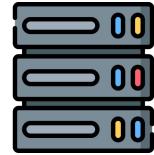
CPU 2 RAM 4

## Horizontal Scaling (Cont.)

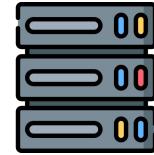


# Horizontal Scaling (Cont.)

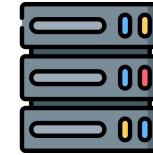
scaling in



CPU 2 RAM 4



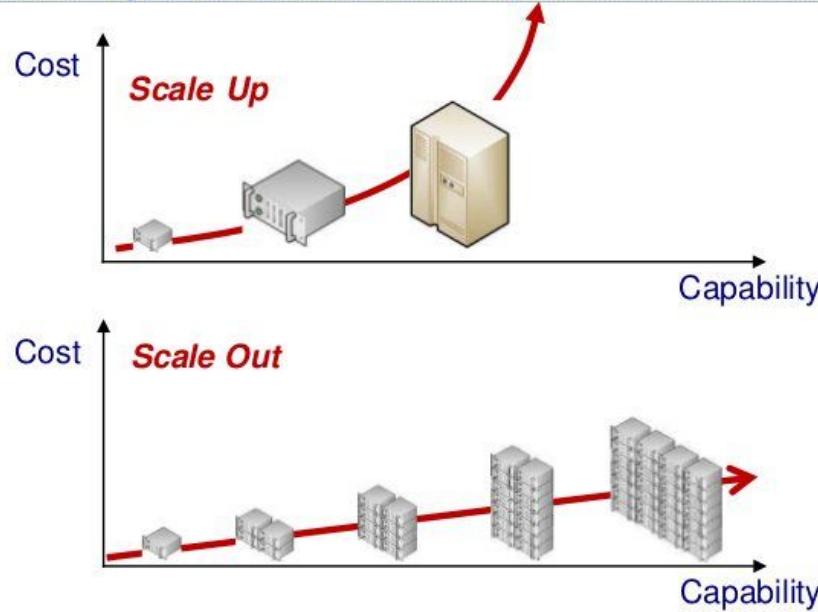
CPU 2 RAM 4



CPU x RAM x

# Economies of scale

## Scale Up vs. Scale Out



[www.scispike.com](http://www.scispike.com)

Copyright © SciSpike 2015

6

Reference Pictures:

- [Scale Up vs. Scale Out](#)



#### Reference Pictures:

- [Tune Your Paid Search Campaigns To Account For Holiday Volatility](#)

# Vertical vs Horizontal scaling

- 4. Autoscaling -

# Vertical vs Horizontal scaling

# Vertical vs Horizontal scaling (Cont.)

## Vertical

### Pros:

- It can sometimes be more cost effective
- It can sometimes be simpler to work with
- It can be easier to maintain

# Vertical vs Horizontal scaling (Cont.)

## Vertical

### Pros:

- It can sometimes be more cost effective
- It can sometimes be simpler to work with
- It can be easier to maintain

### Cons:

- You still have a single point of failure
- It is inherently limited
- It can sometimes cost more

# Vertical vs Horizontal scaling (Cont.)

## Vertical



## Horizontal

### Pros:

- It can sometimes be more cost effective
- It can sometimes be simpler to work with
- It can be easier to maintain

### Cons:

- You still have a single point of failure
- It is inherently limited
- It can sometimes cost more

### Pros:

- It increases availability and resilience/fault-tolerance.
- It can be more cost-effective.
- It can increase performance.

# Vertical vs Horizontal scaling (Cont.)

## Vertical



## Horizontal

### Pros:

- It can sometimes be more cost effective
- It can sometimes be simpler to work with
- It can be easier to maintain

### Cons:

- You still have a single point of failure
- It is inherently limited
- It can sometimes cost more

### Pros:

- It increases availability and resilience/fault-tolerance.
- It can be more cost-effective.
- It can increase performance.

### Cons:

- It can be more complex.
- It generally costs more up-front

# Vertical vs Horizontal scaling (Cont.)

## Vertical



## Horizontal

### Pros:

- It can sometimes be more cost effective
- It can sometimes be simpler to work with
- It can be easier to maintain

### Cons:

- You still have a single point of failure
- It is inherently limited
- It can sometimes cost more

### Pros:

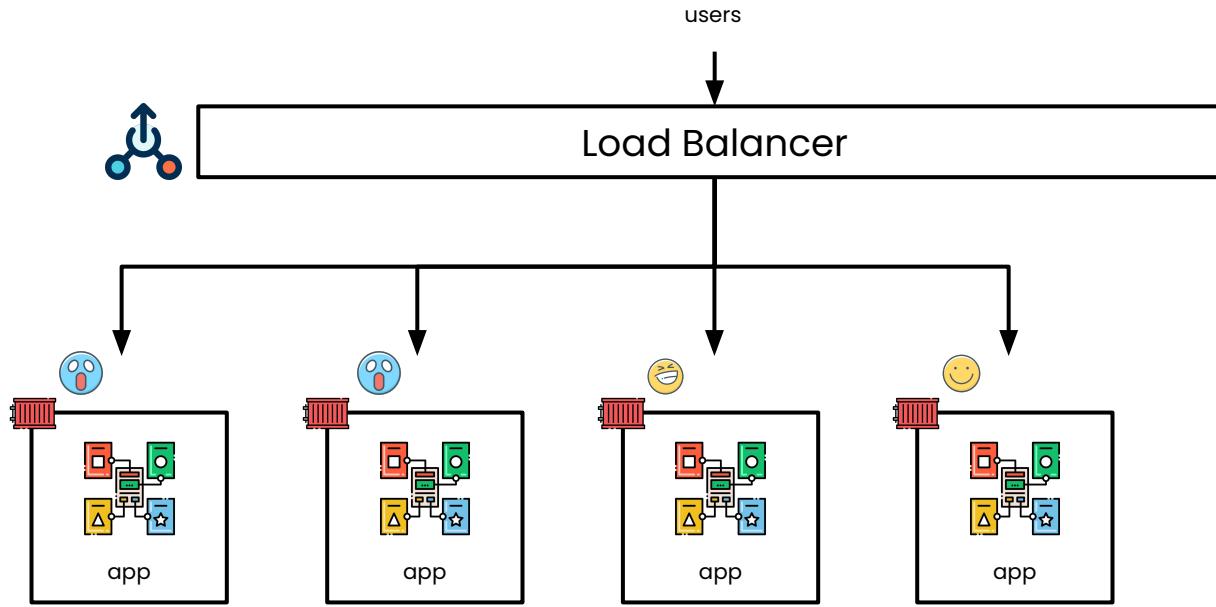
- It increases availability and resilience/fault-tolerance.
- It can be more cost-effective.
- It can increase performance.

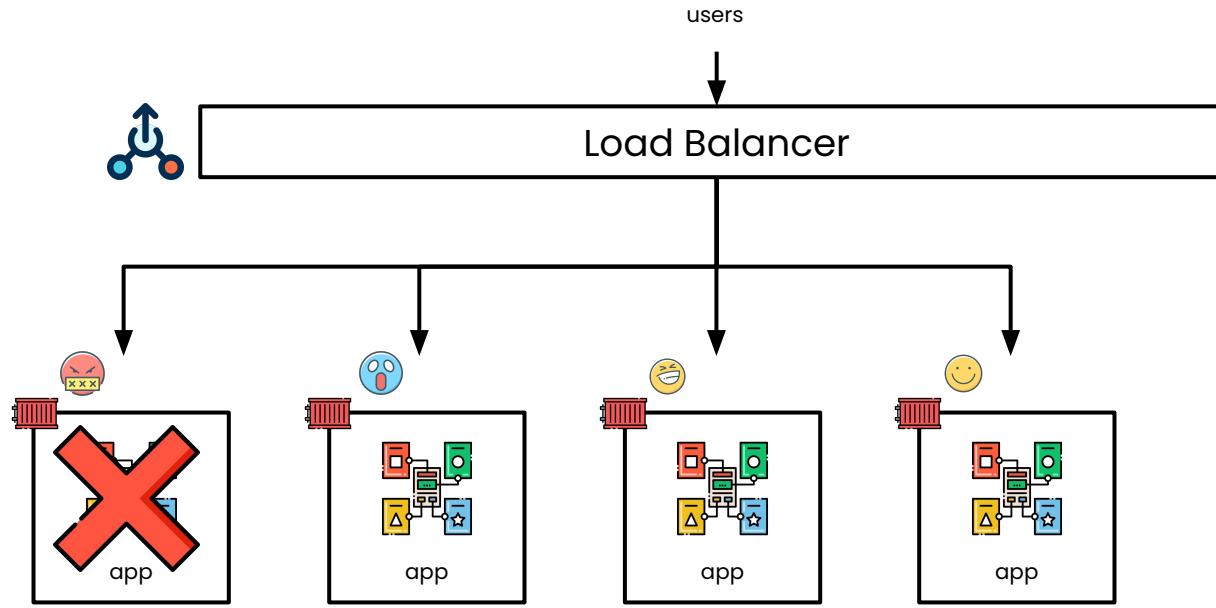
### Cons:

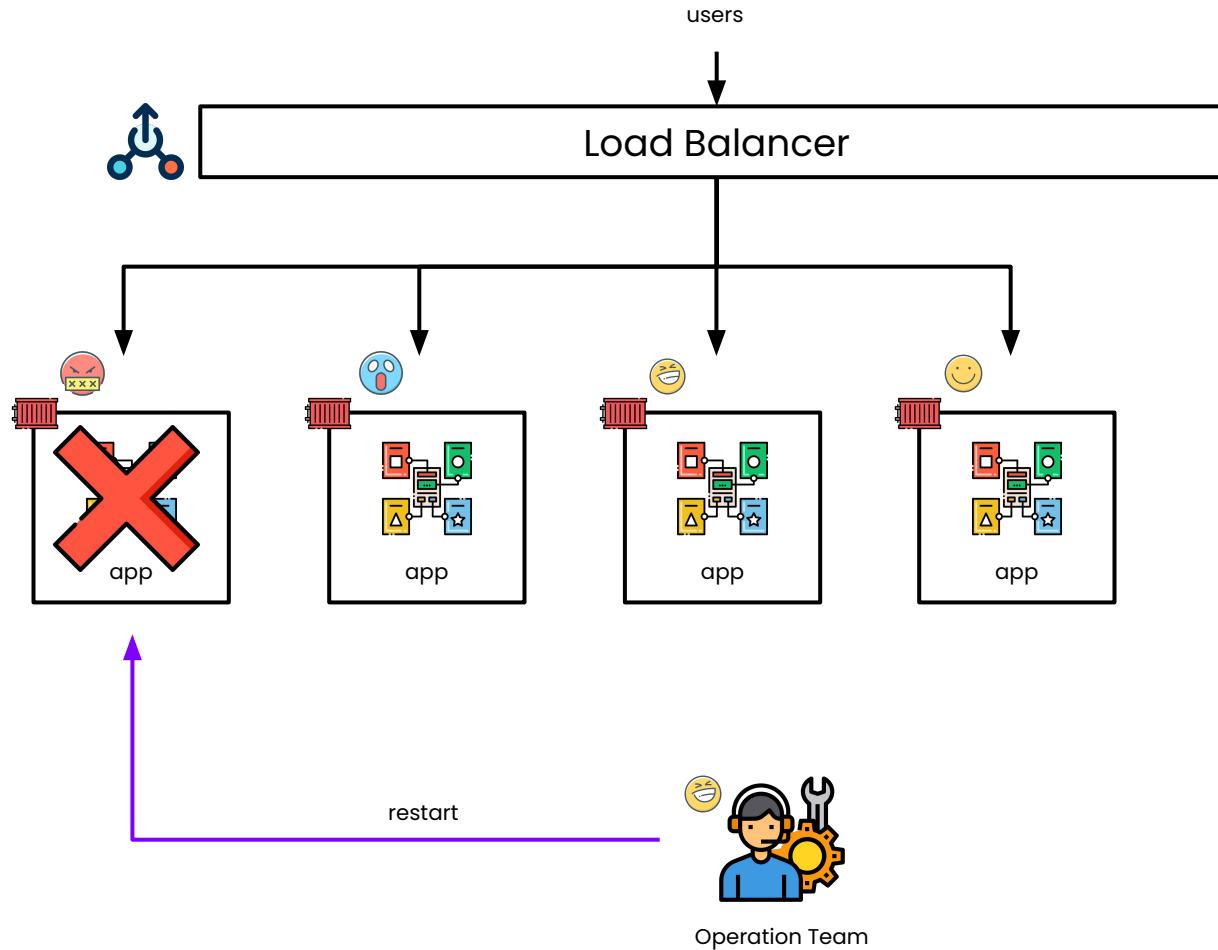
- It can be more complex.
- It generally costs more up-front

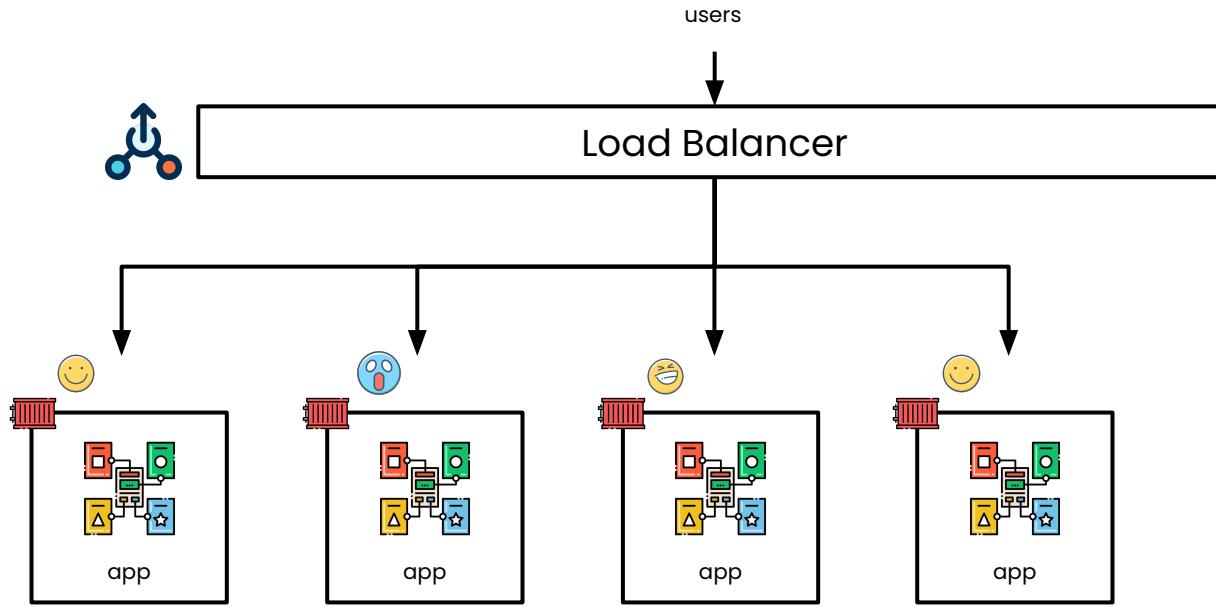
**Come back to the **real world** use case**

# Use containers how to scale out?

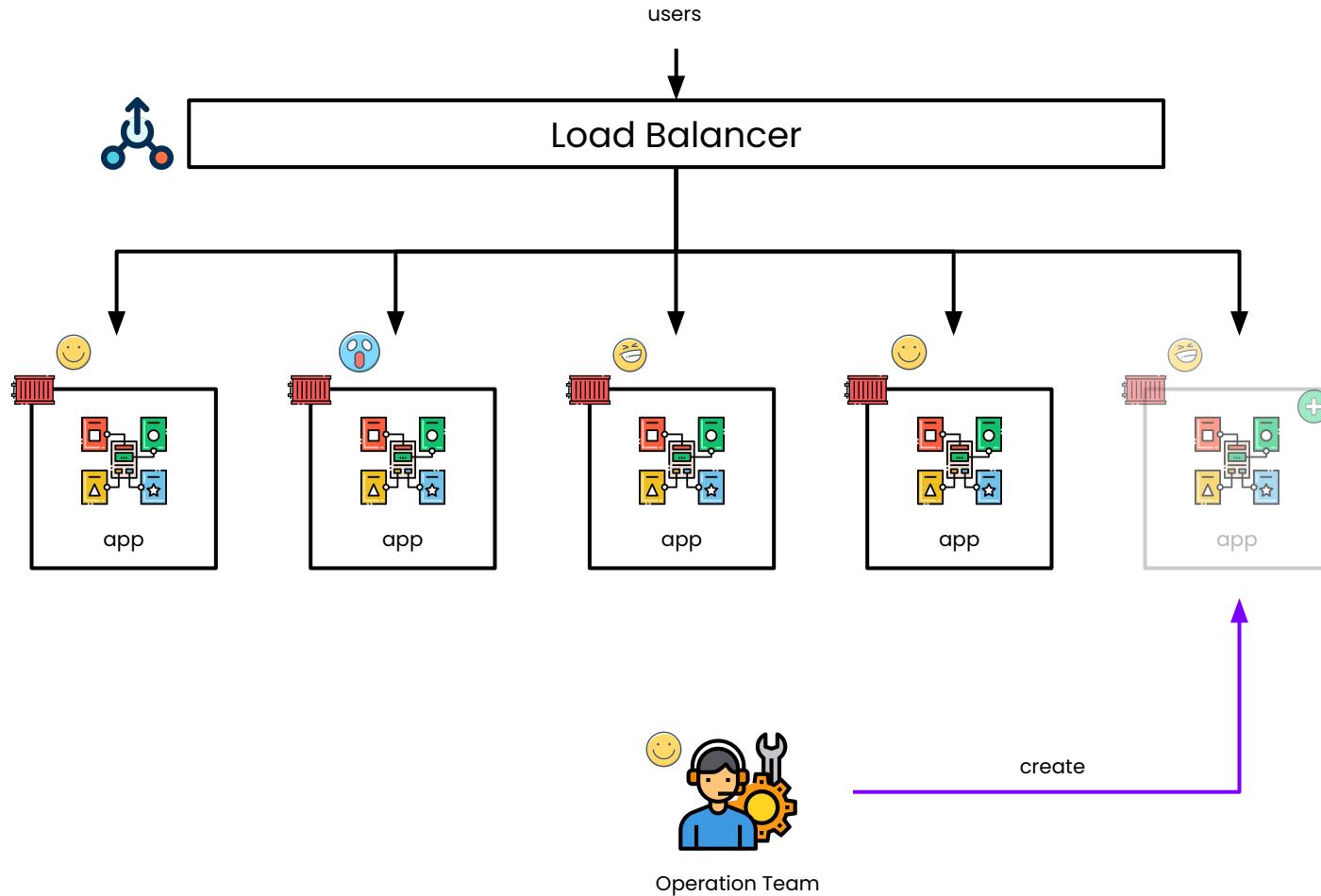


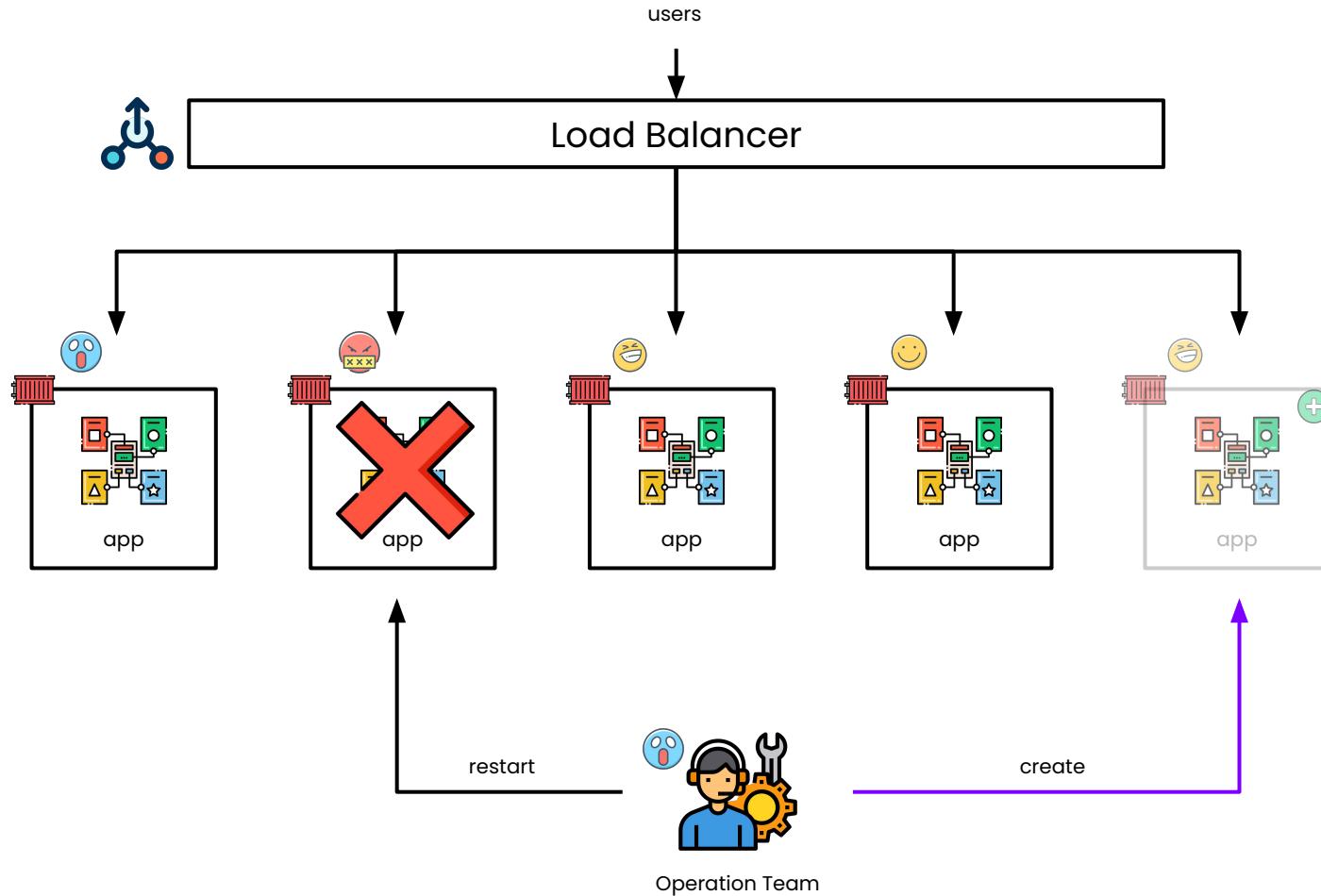






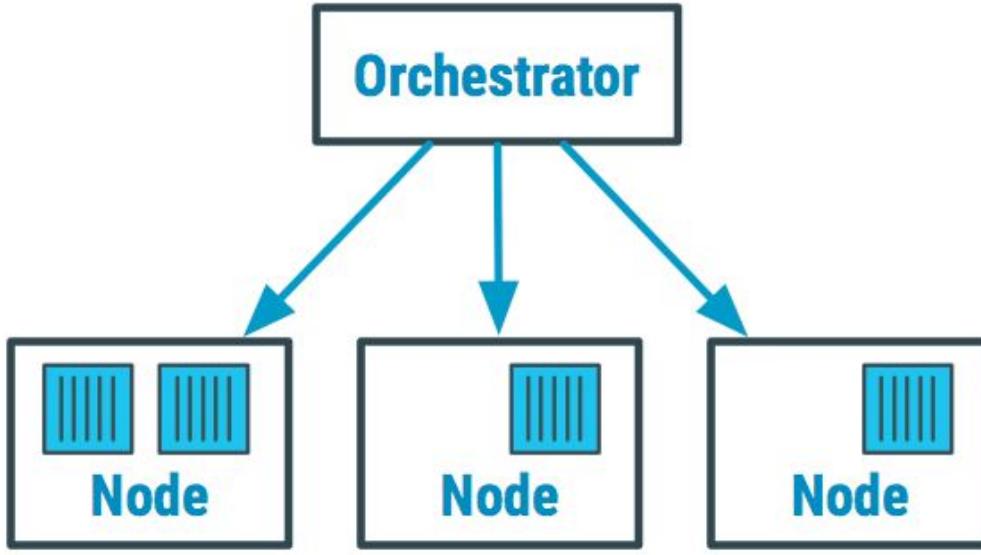
Operation Team





# Introduction to Orchestrator Docker Swarm & Kubernetes

# Container Orchestration



Reference Pictures:

- <https://devopedia.org/images/article/37/8281.1530784485.png>

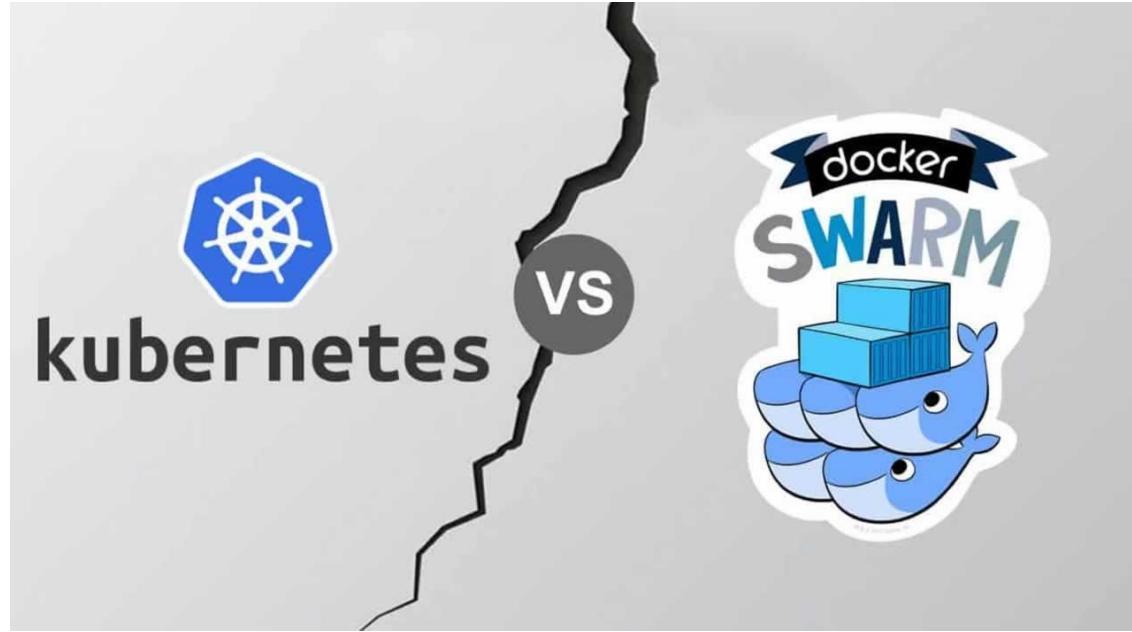
# Container Orchestration tools



Reference Pictures:

- [Managed Container Orchestration with Amazon ECS](#)

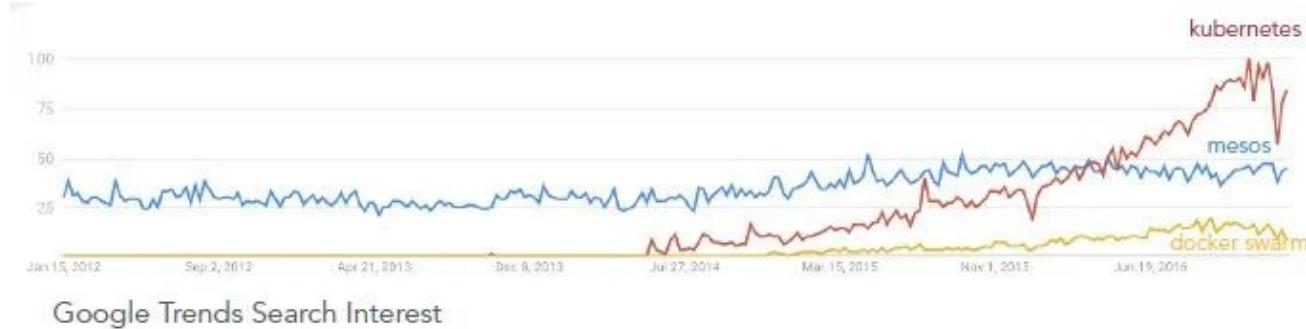
# Kubernetes vs Docker Swarm



## Reference Pictures:

- [Kubernetes vs Docker Swarm | Comparison of Docker and Kubernetes | Kubernetes Training | Edureka](#)

# Kubernetes was the clear winner



## Google Trends Search Interest

### Project Maturity

- Built by the same team that built google's [borg](#), matured within google and outside

### Enterprise Adoption

- Strong - well known [customers](#) running in production

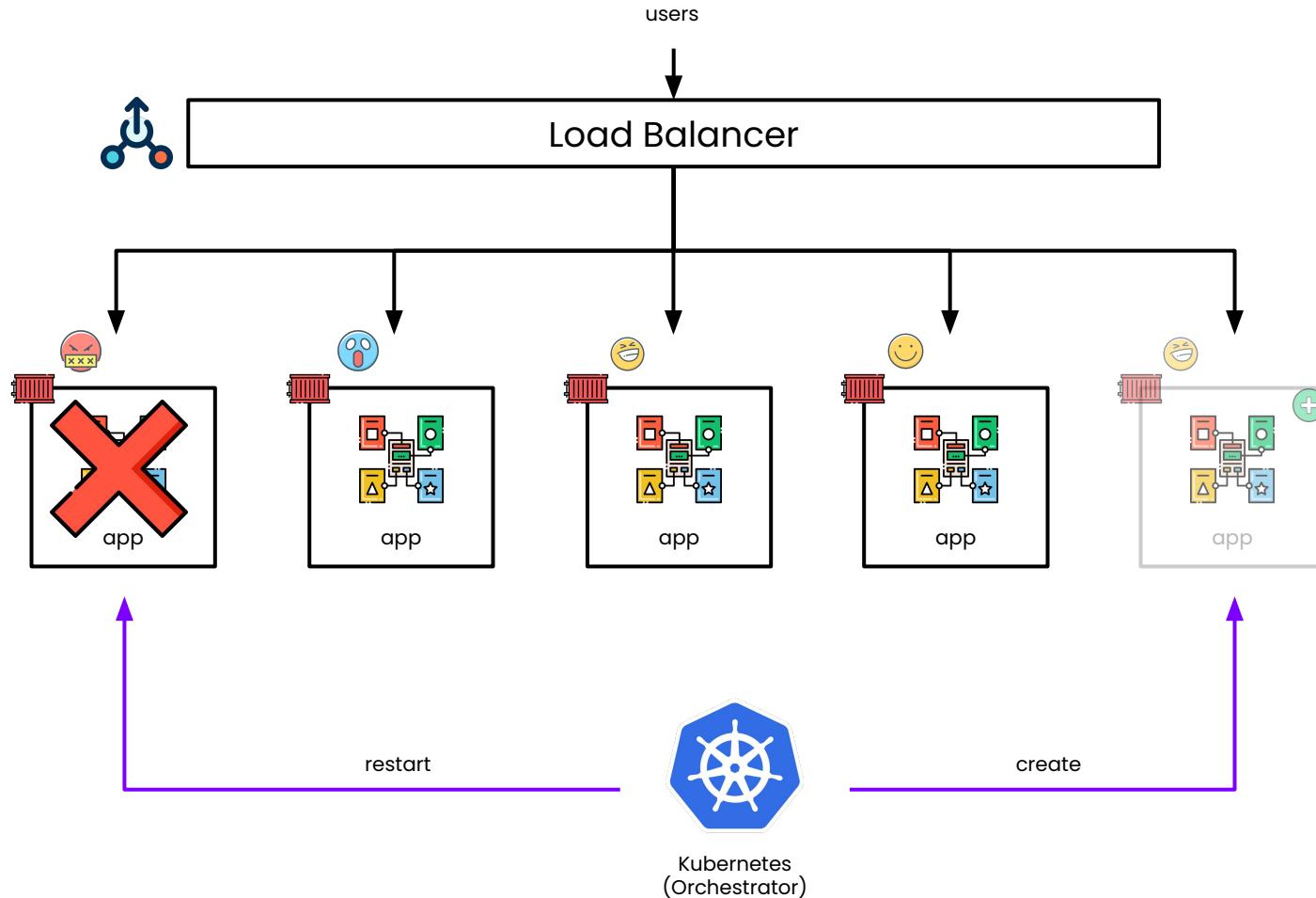
### Community Strength

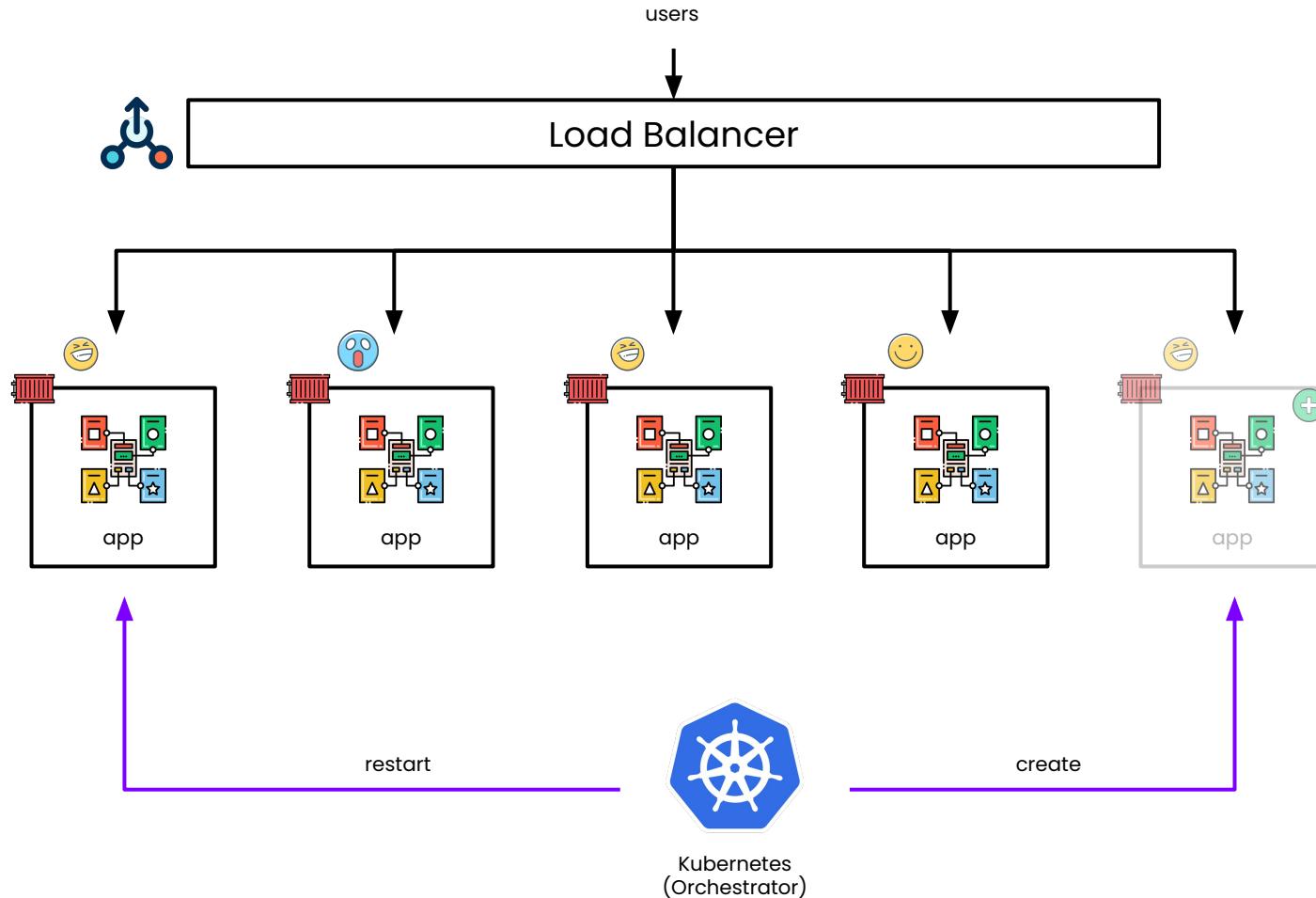
- > 1000 contributors
- 20K github stars
- Most active OSS community - google, red hat, intel, [box](#). Astounding pace of innovation.

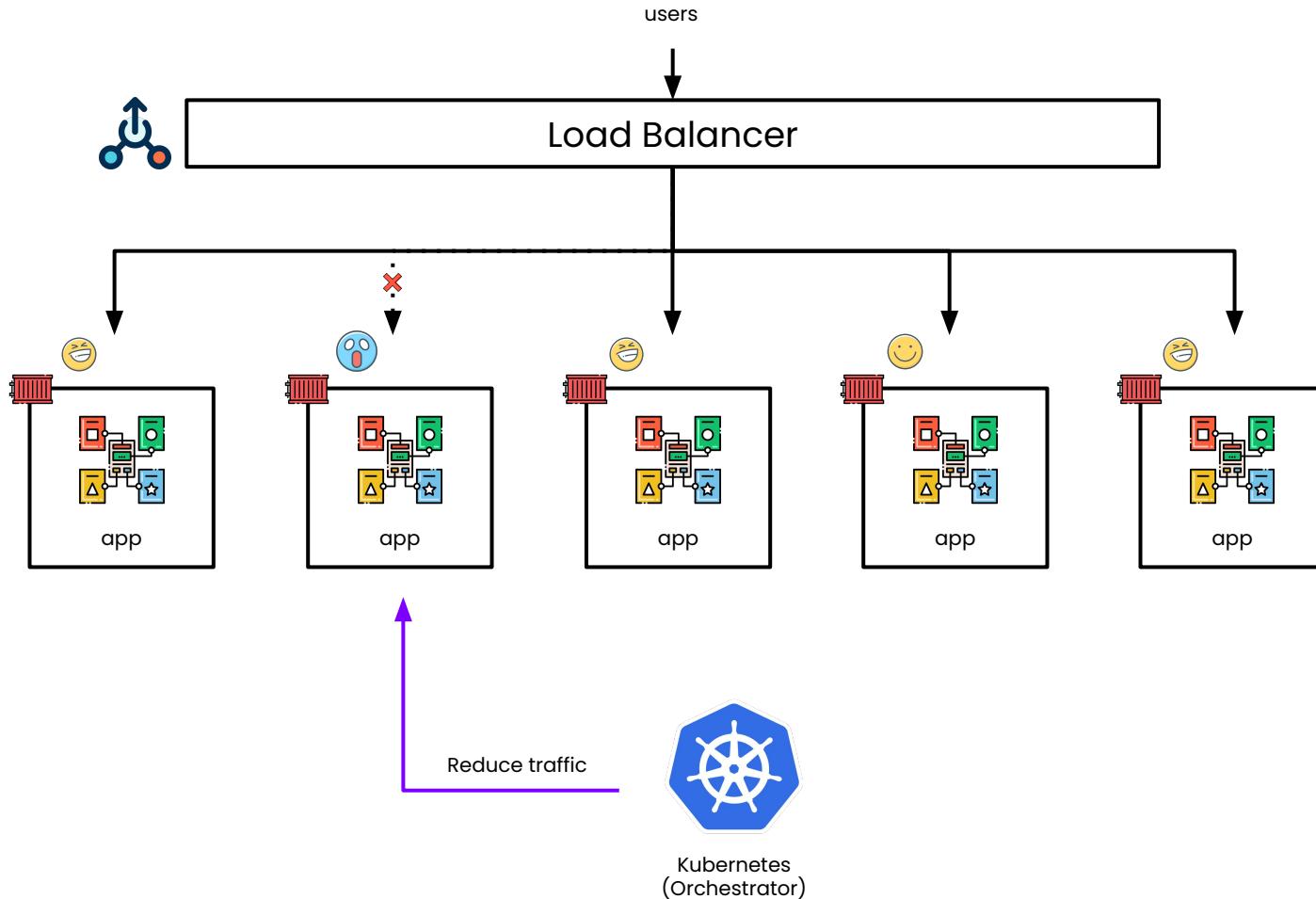
## Reference Pictures:

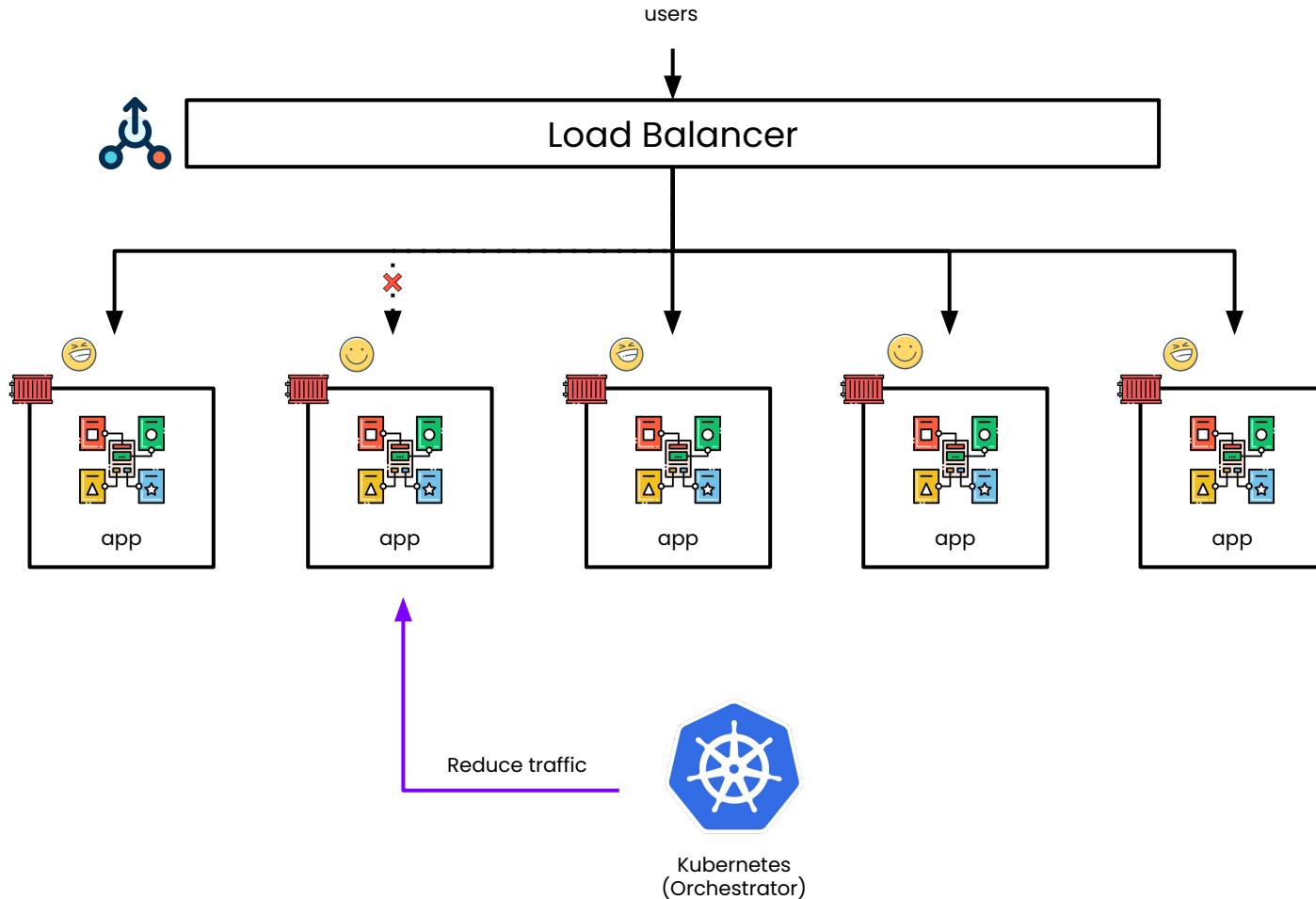
- [The Gravity of Kubernetes](#)

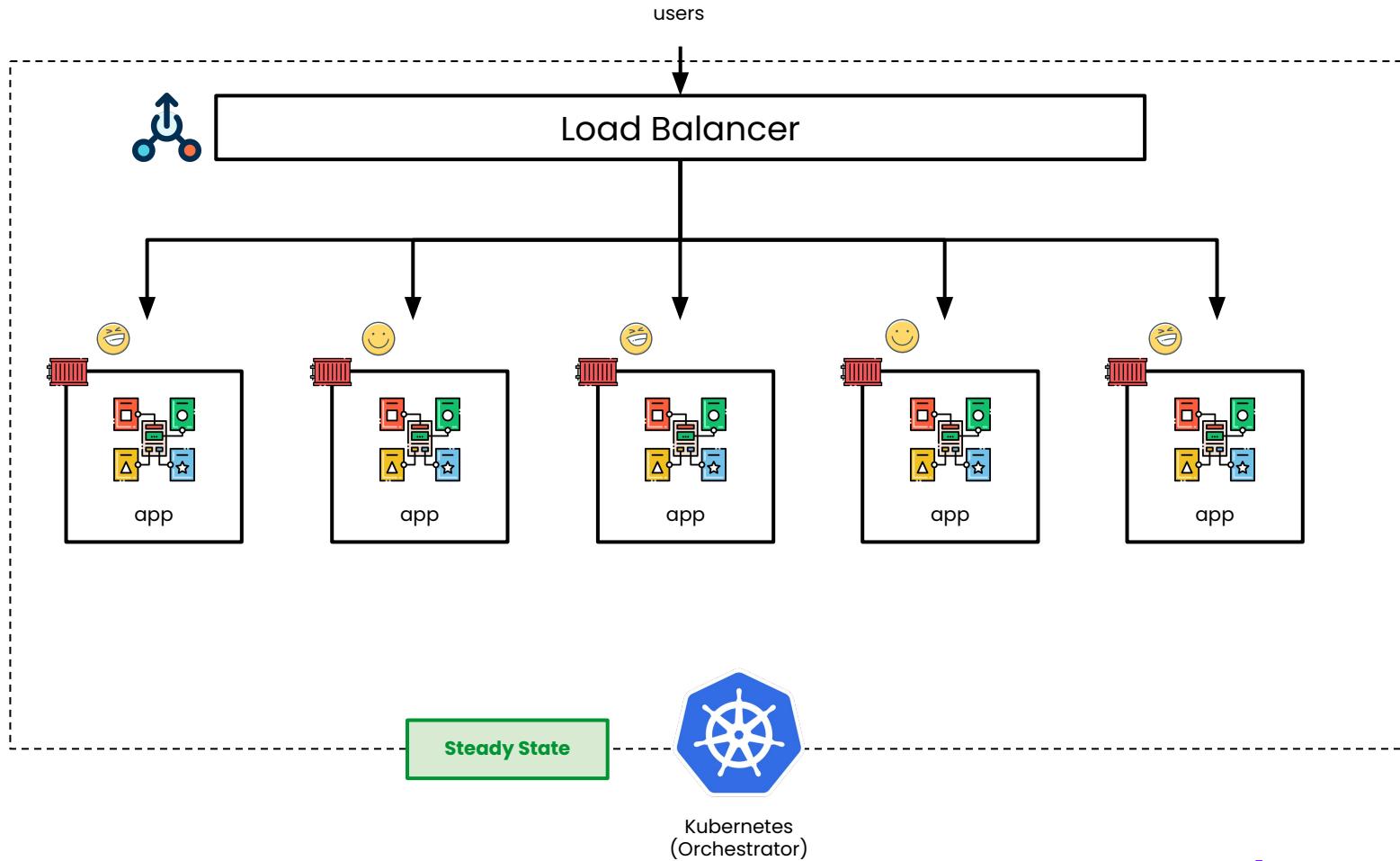
**and now **Kubernetes** comes in**











# Benefit of Kubernetes

- Why do we need Kubernetes? -

# Benefit of Kubernetes

## Benefit of Kubernetes (Cont.)

1. Restart, Redeploy, Create new Application, Rollback Deploy

## Benefit of Kubernetes (Cont.)

1. Restart, Redeploy, Create new Application, Rollback Deploy
2. Deployment Risk free

## Benefit of Kubernetes (Cont.)

1. Restart, Redeploy, Create new Application, Rollback Deploy
2. Deployment Risk free
3. Continuous Deployment

## Benefit of Kubernetes (Cont.)

1. Restart, Redeploy, Create new Application, Rollback Deploy
2. Deployment Risk free
3. Continuous Deployment
4. Scalability

## Benefit of Kubernetes (Cont.)

1. Restart, Redeploy, Create new Application, Rollback Deploy
2. Deployment Risk free
3. Continuous Deployment
4. Scalability
5. Observability

## Benefit of Kubernetes (Cont.)

1. Restart, Redeploy, Create new Application, Rollback Deploy
2. Deployment Risk free
3. Continuous Deployment
4. Scalability
5. Observability
6. Security & Compliance

## Benefit of Kubernetes (Cont.)

1. Restart, Redeploy, Create new Application, Rollback Deploy
2. Deployment Risk free
3. Continuous Deployment
4. Scalability
5. Observability
6. Security & Compliance

## K8s can do

- Service discovery and load balancing
- Storage orchestration
- Automated rollouts and rollbacks
- Automatic bin packing
- Self-healing
- Secret and configuration management

# What is K8s?

is an open-source container-orchestration system for

- Automating software deployment
- Scaling
- Management

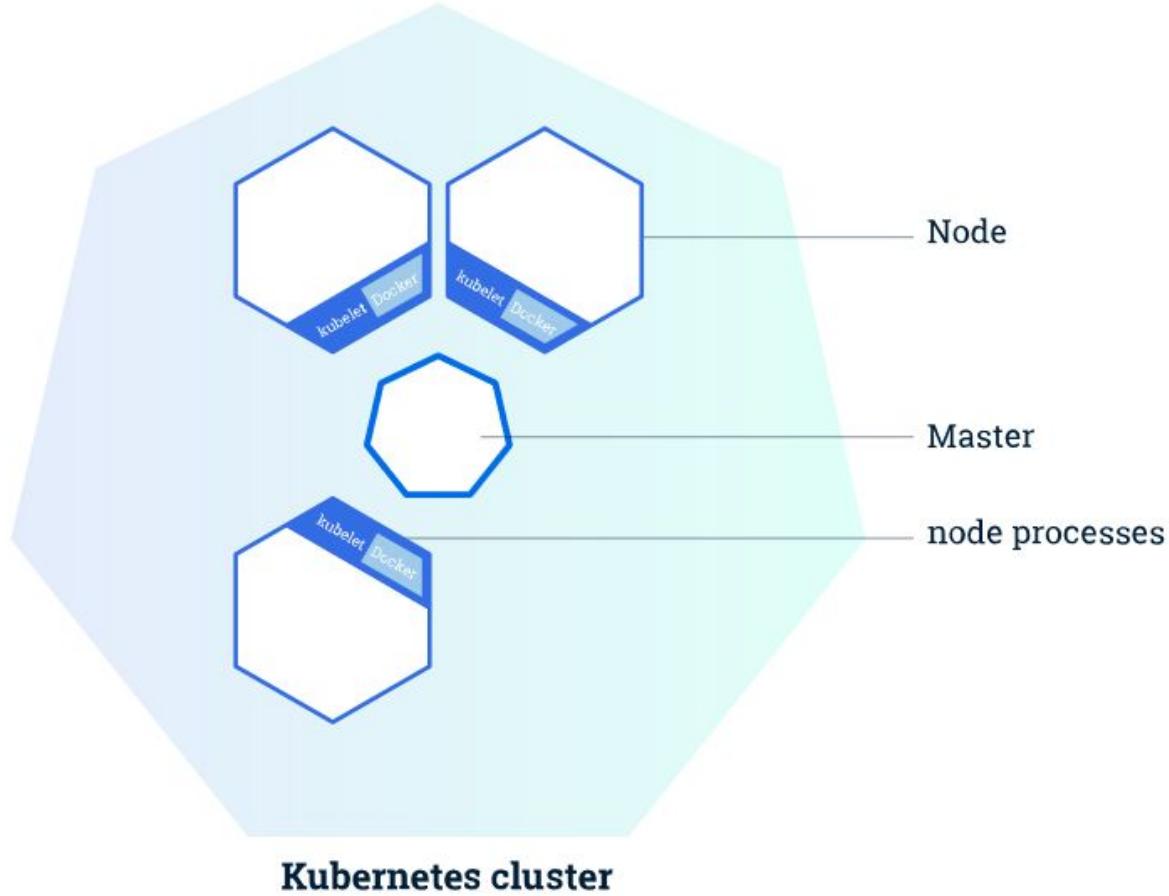
Credit: [Kubernetes Wikipedia](#)

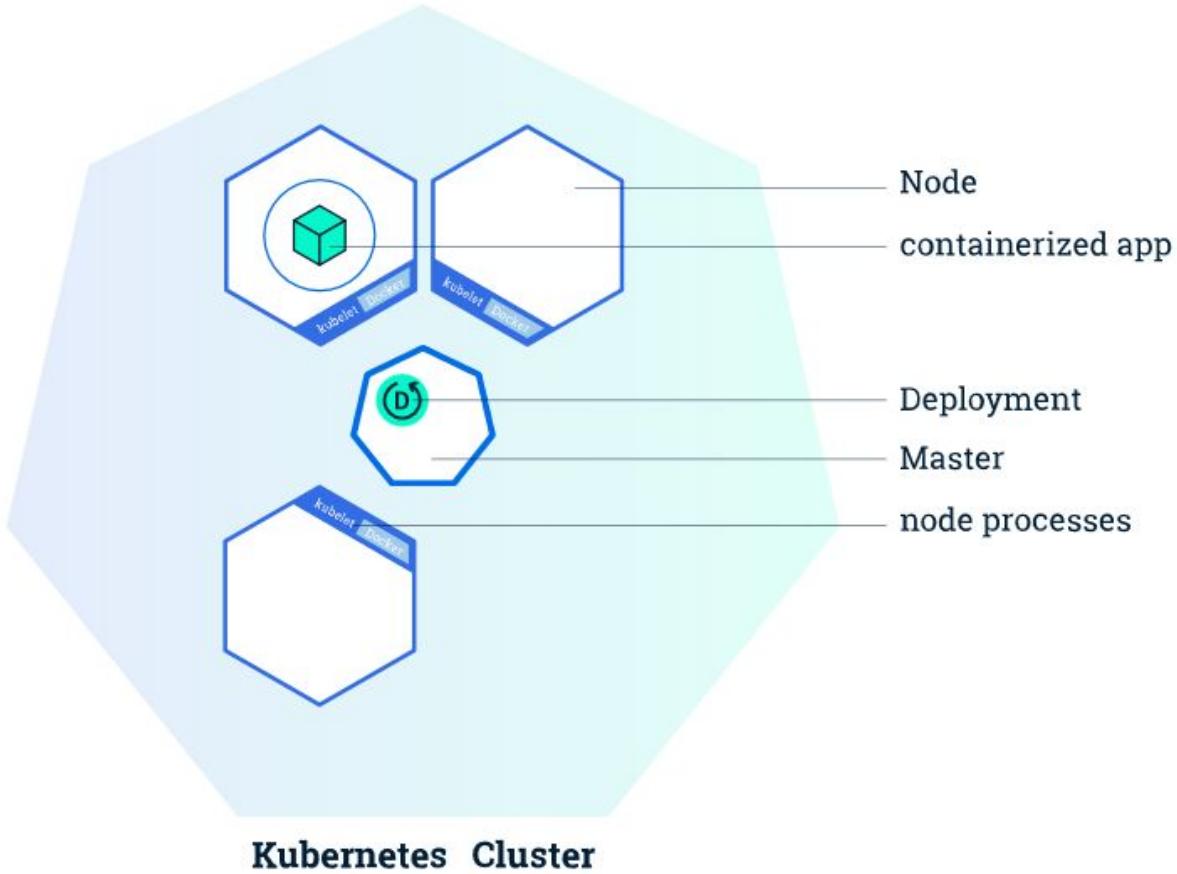
ເນັ້ນຕົວທີ່ໄດ້ຮັບສູນໆທີ່ອກງານເຮັດວຽກຂະໜາດໃຫ້ນັ້ນ ແລະຂອງສຽງລືຊີສິຖົງ ທັນມີໄຟເຫຍແພຣີນທີ່ສາມາຮັນ ຜູ້ອະນຸມັດ ຈະຄຸກດຳເນີນຕົດຕາມງູ່ໝາຍ

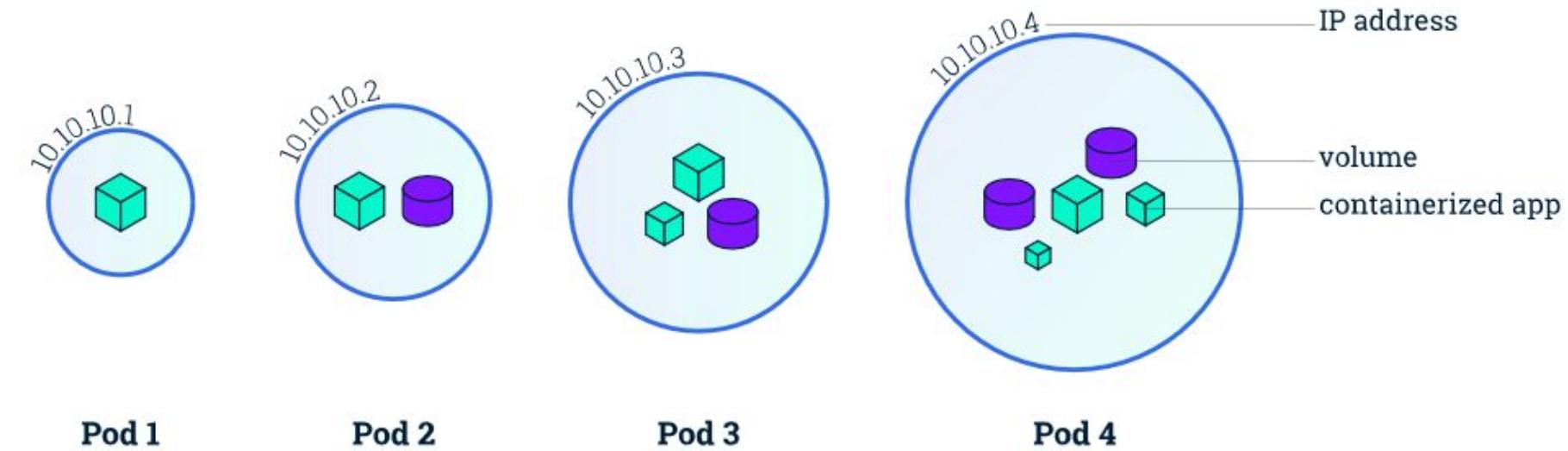
**Jumpbox®**

## Want more story K8s

- [Kubernetes คืออะไร ทำไม Orchestration จึงเป็นหัวใจของ Infrastructure ยุคนี้](#)





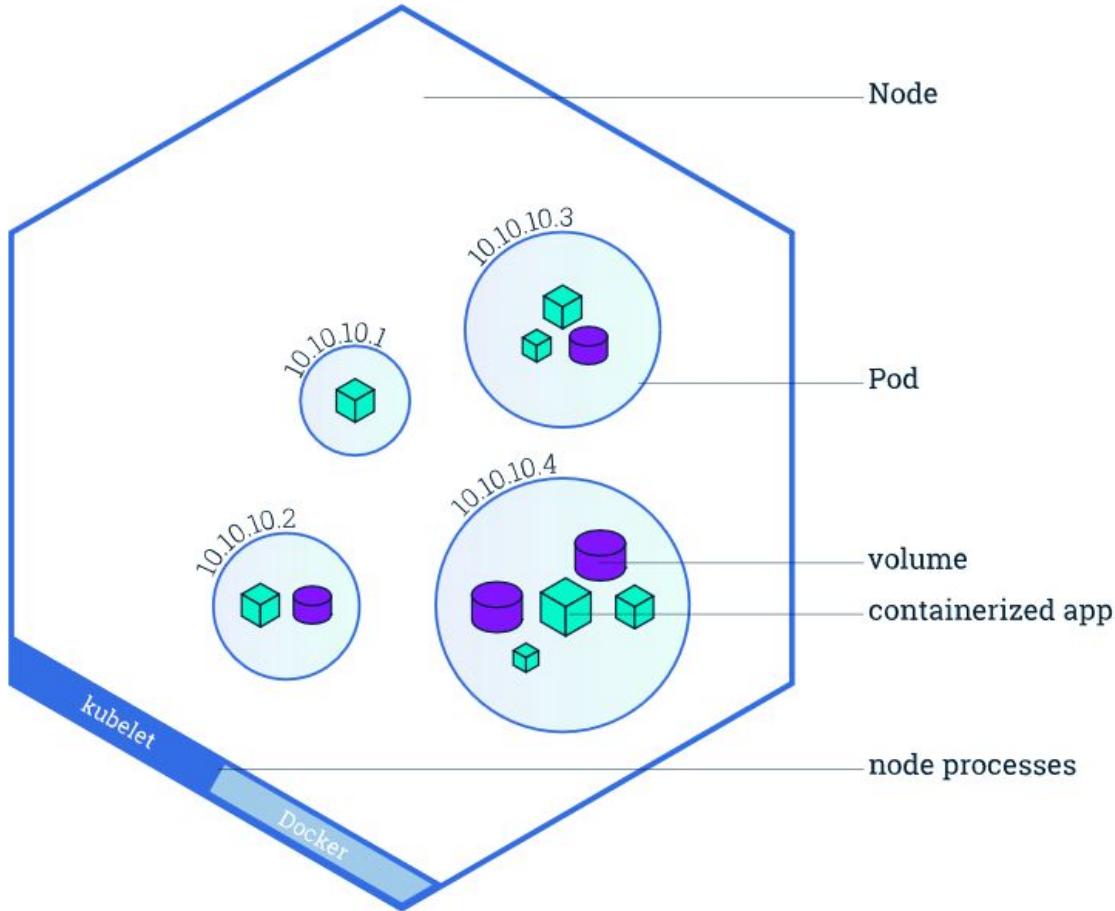


Pod 1

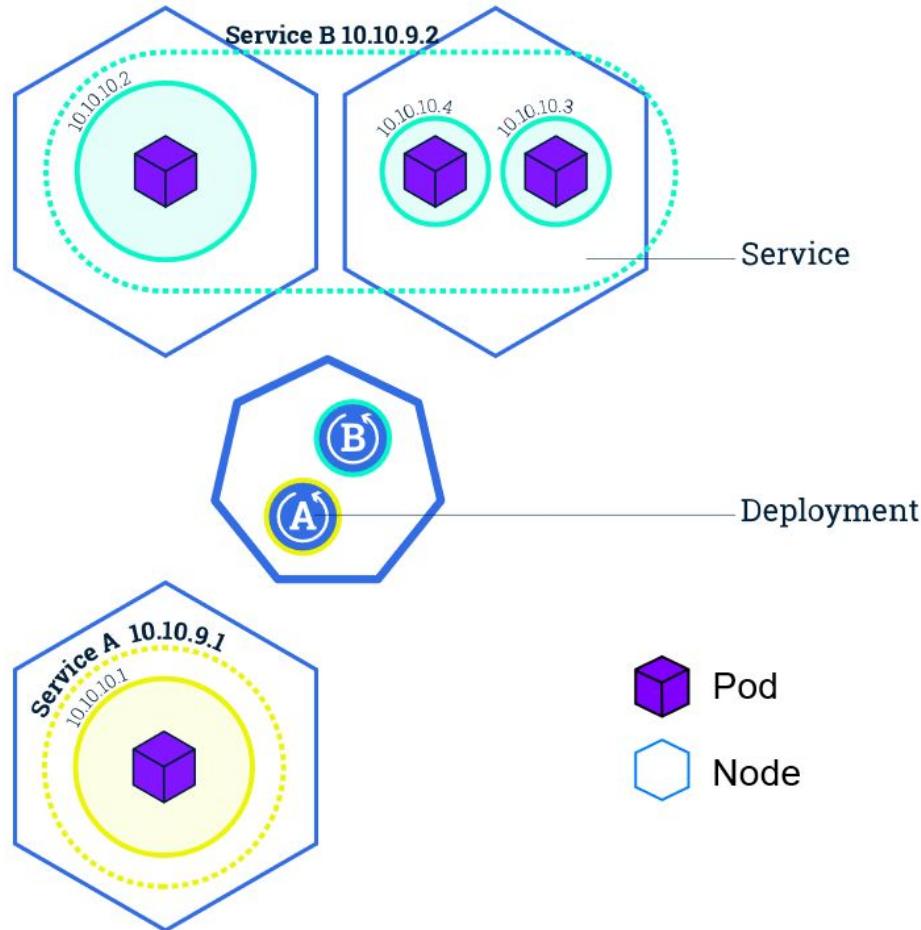
Pod 2

Pod 3

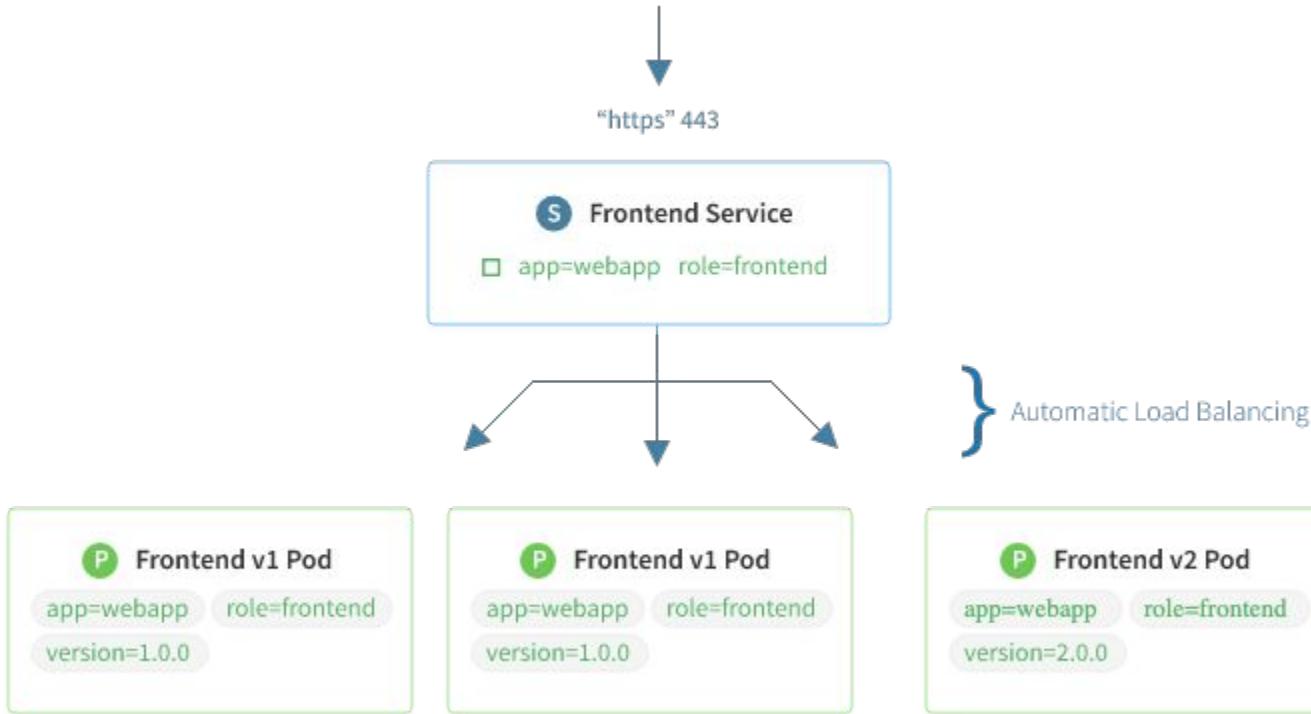
Pod 4



เจ้าของลิขสิทธิ์ อนุญาติให้ใช้ส่วนนี้เพื่อการเรียนรู้ส่วนบุคคลเท่านั้น และขอสงวนลิขสิทธิ์ ห้ามมิให้เผยแพร่ในที่สาธารณะ ผู้อื่นเดิน จดถูกต้องเป็นคิดตามกฎหมาย



เจ้าของลิขสิทธิ์ อนุญาติให้ใช้ส่วนนี้เพื่อการเรียนรู้ส่วนบุคคลเท่านั้น และขอสงวนลิขสิทธิ์ ห้ามมิให้เผยแพร่ในที่สาธารณะ ผู้อื่นเดินทาง จะถูกดำเนินคดีตามกฎหมาย



#### Reference Pictures:

- [Kubernetes – Basics](#)

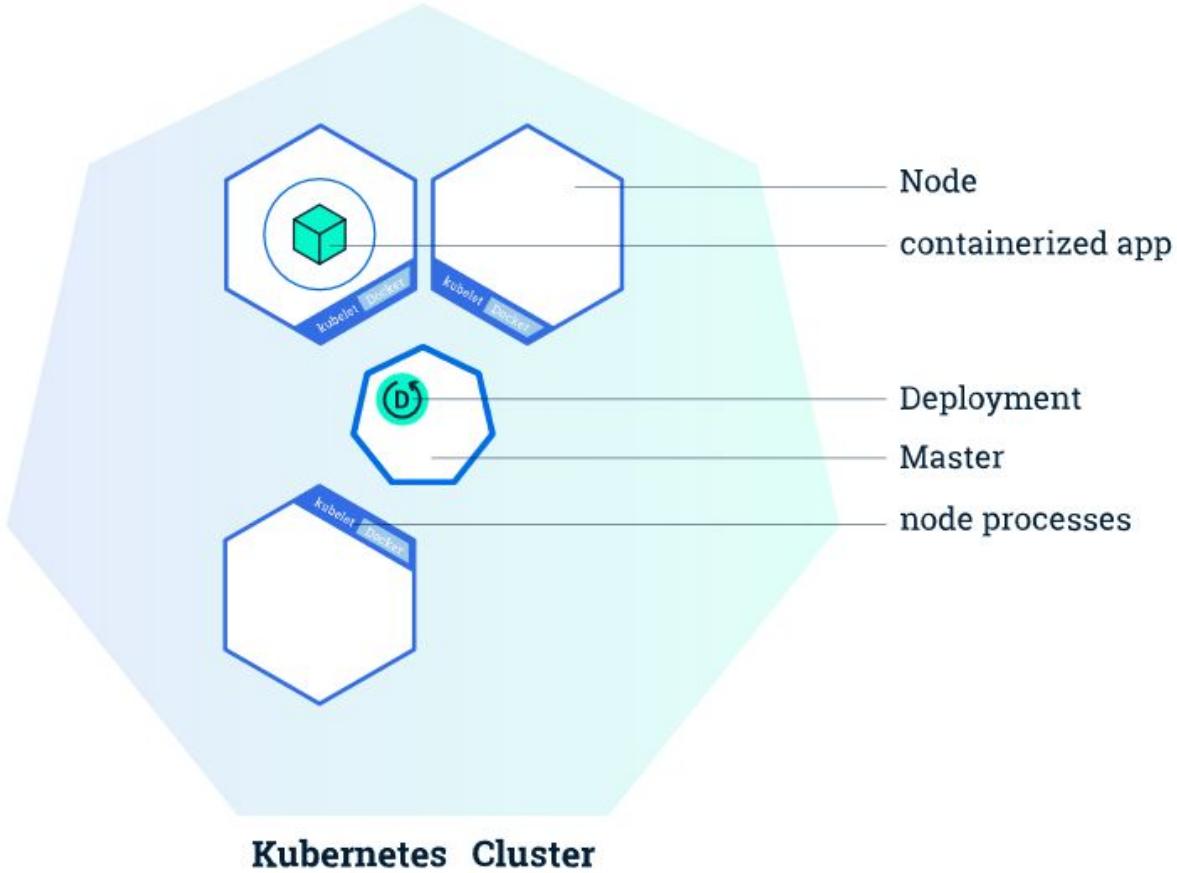


# Activity

# Let's play

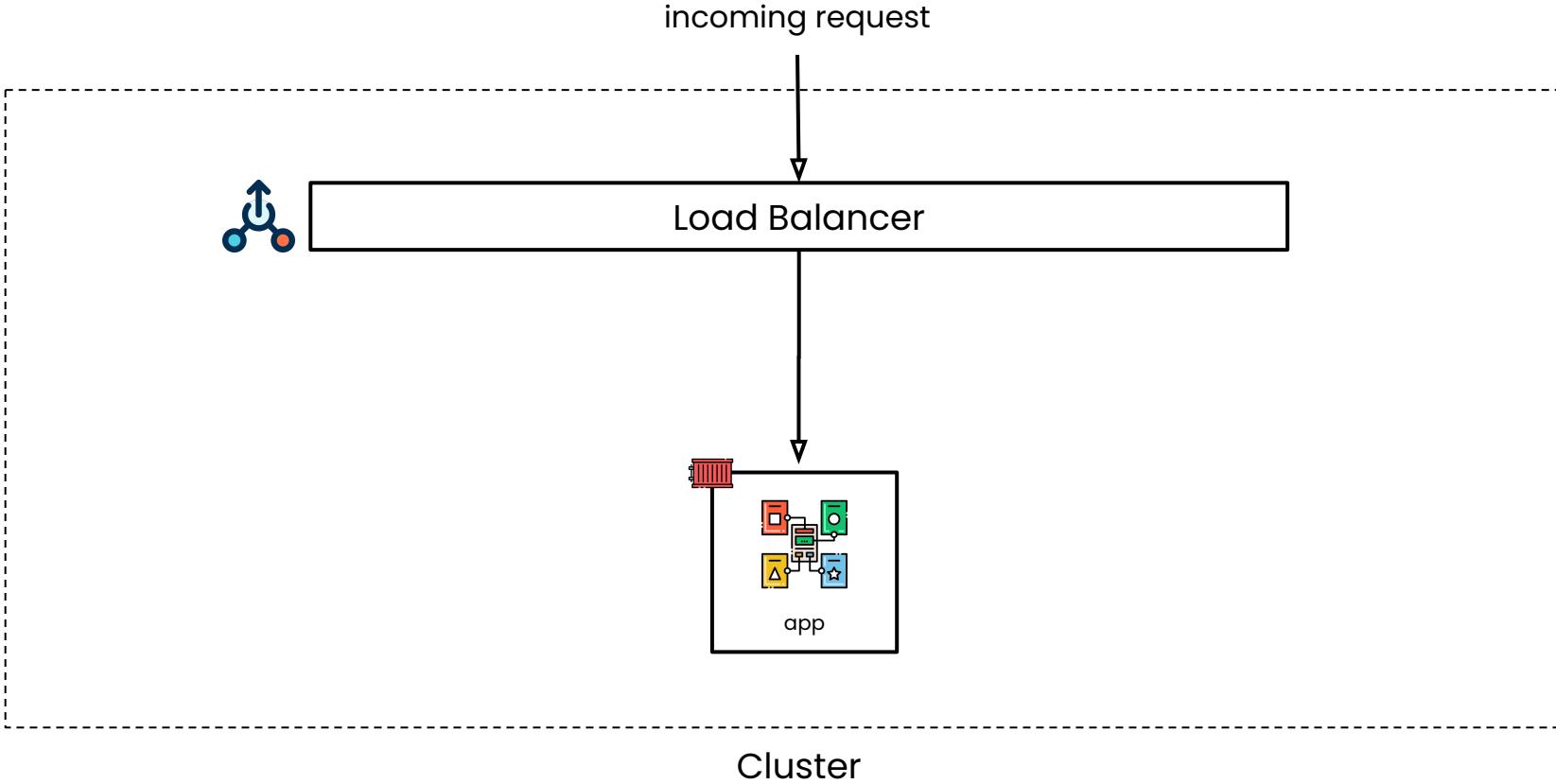
# Interactive Tutorial - Deploying an App

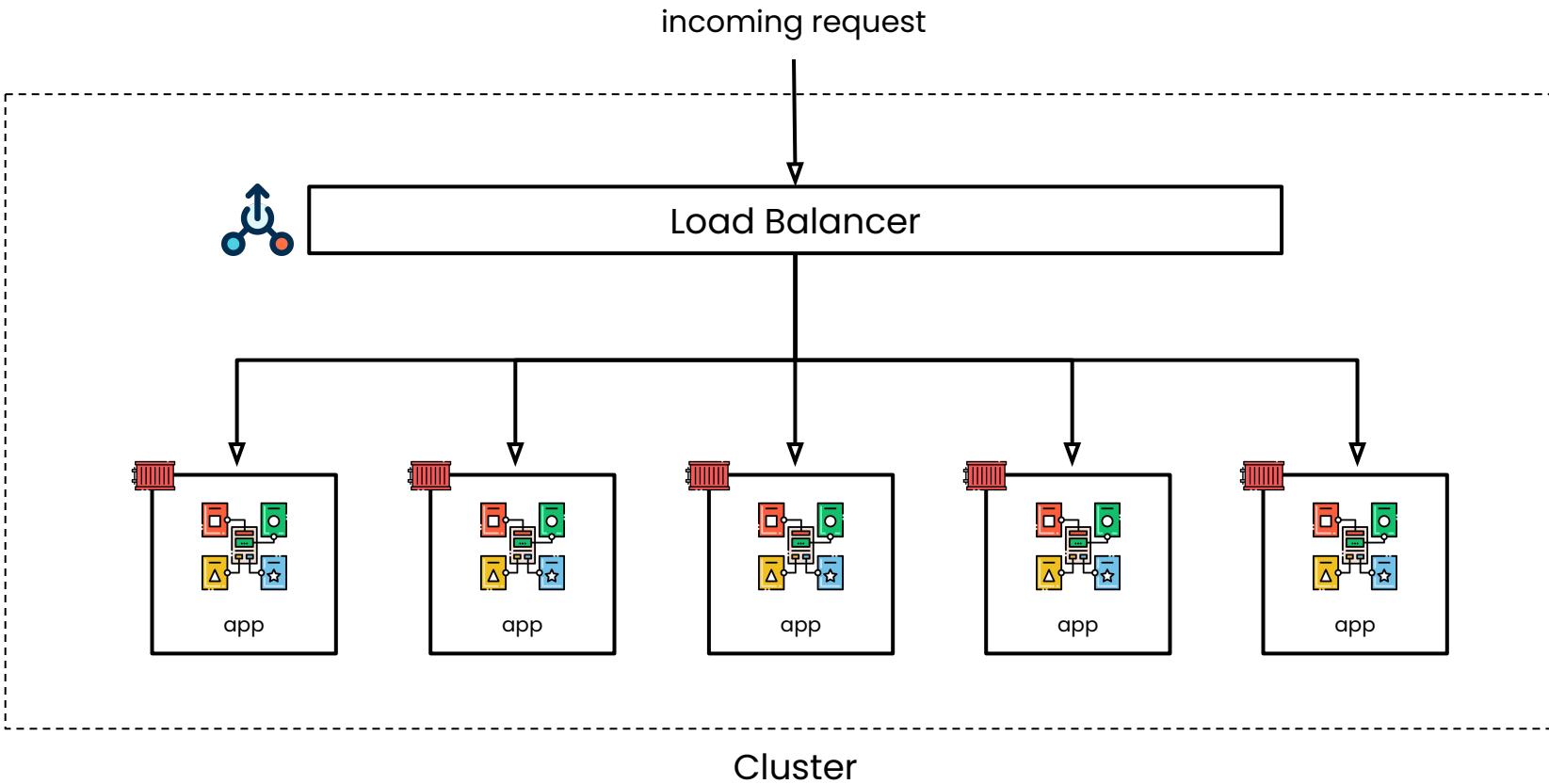
- Play with k8s on local or [KillerCoda](#)
- Notion: [Interactive Tutorial - Deploying an App](#)

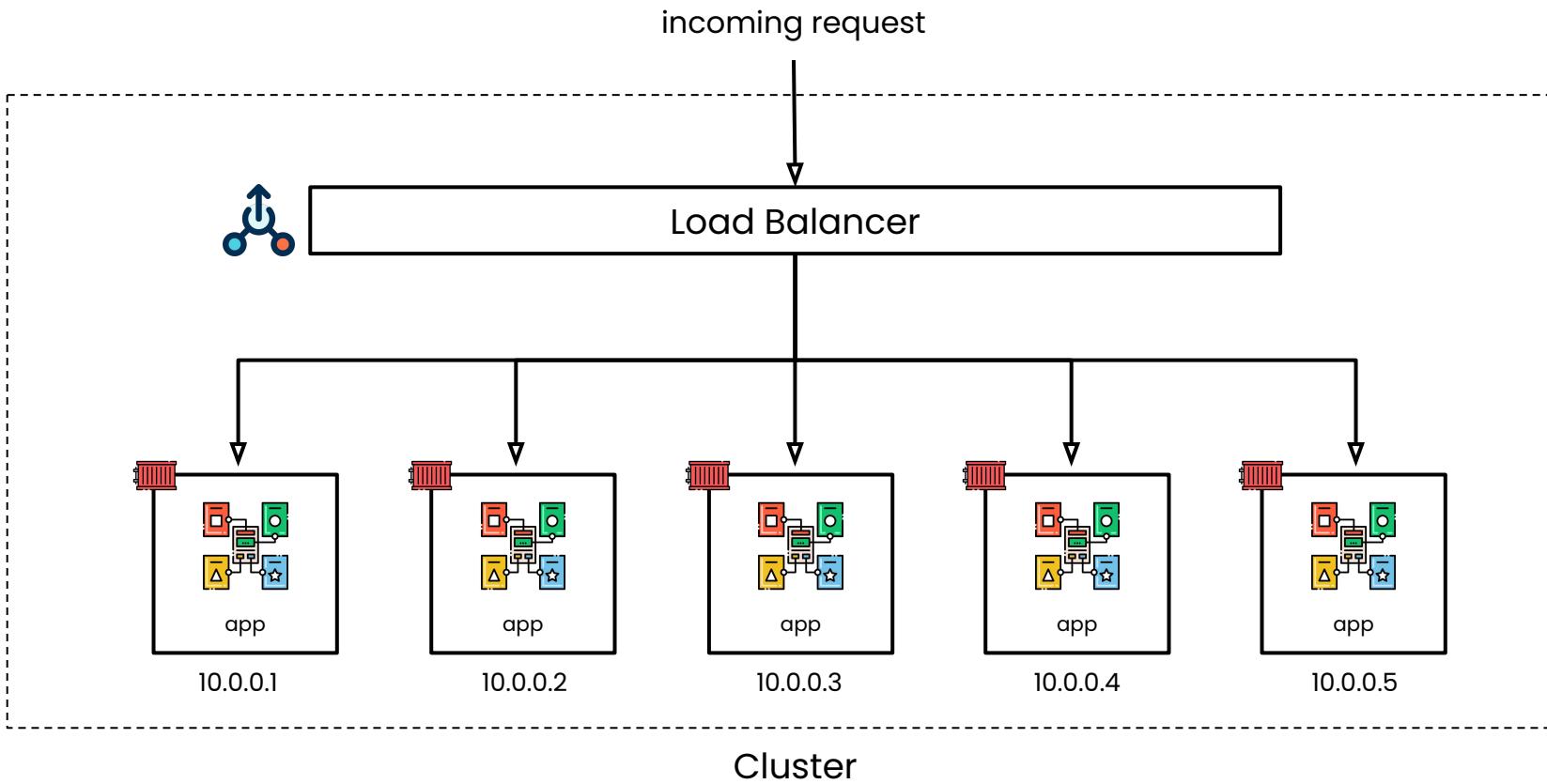


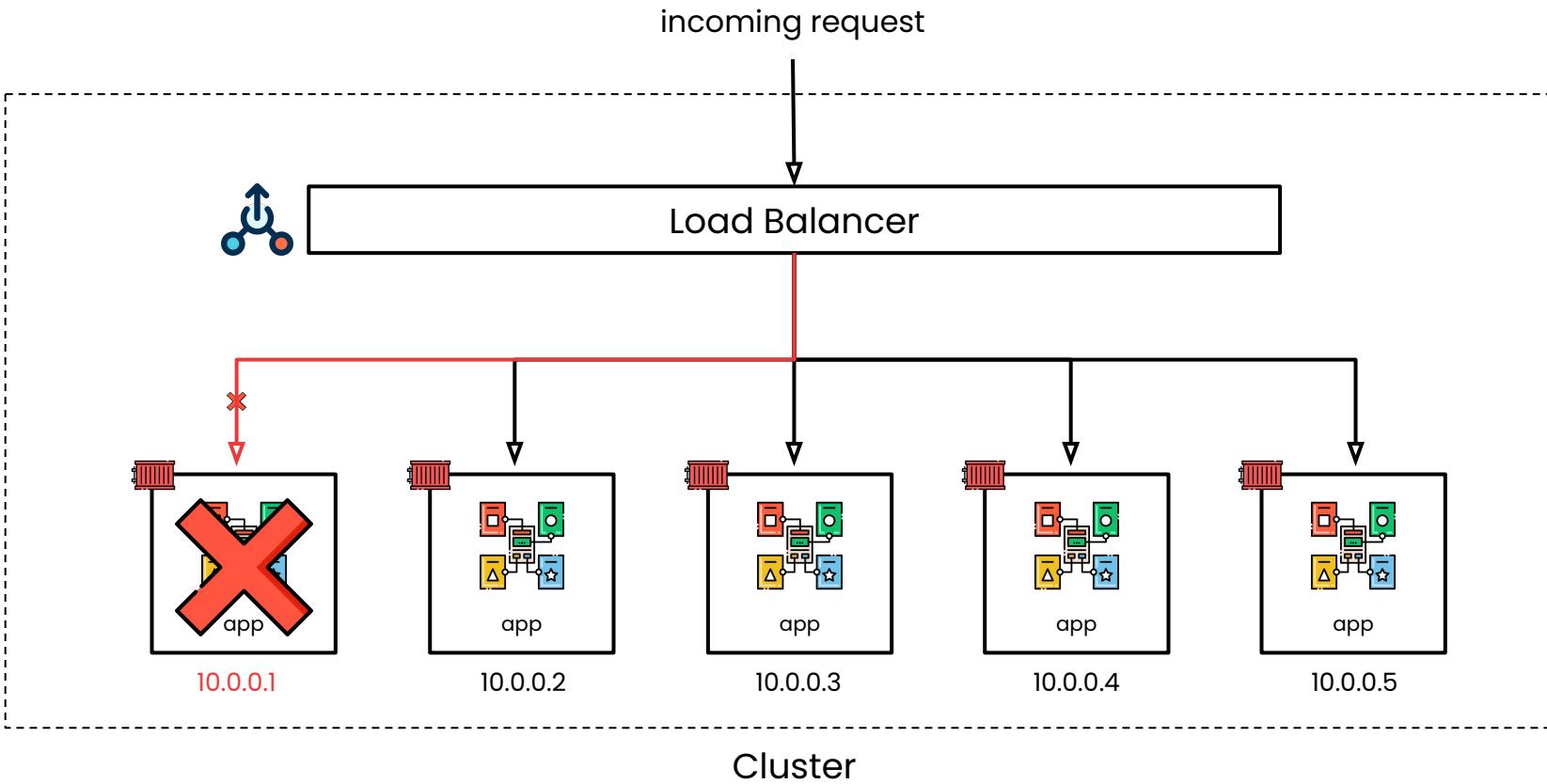
# Interactive Tutorial - Exploring Your App

- Notion: [Interactive Tutorial - Exploring Your App](#)

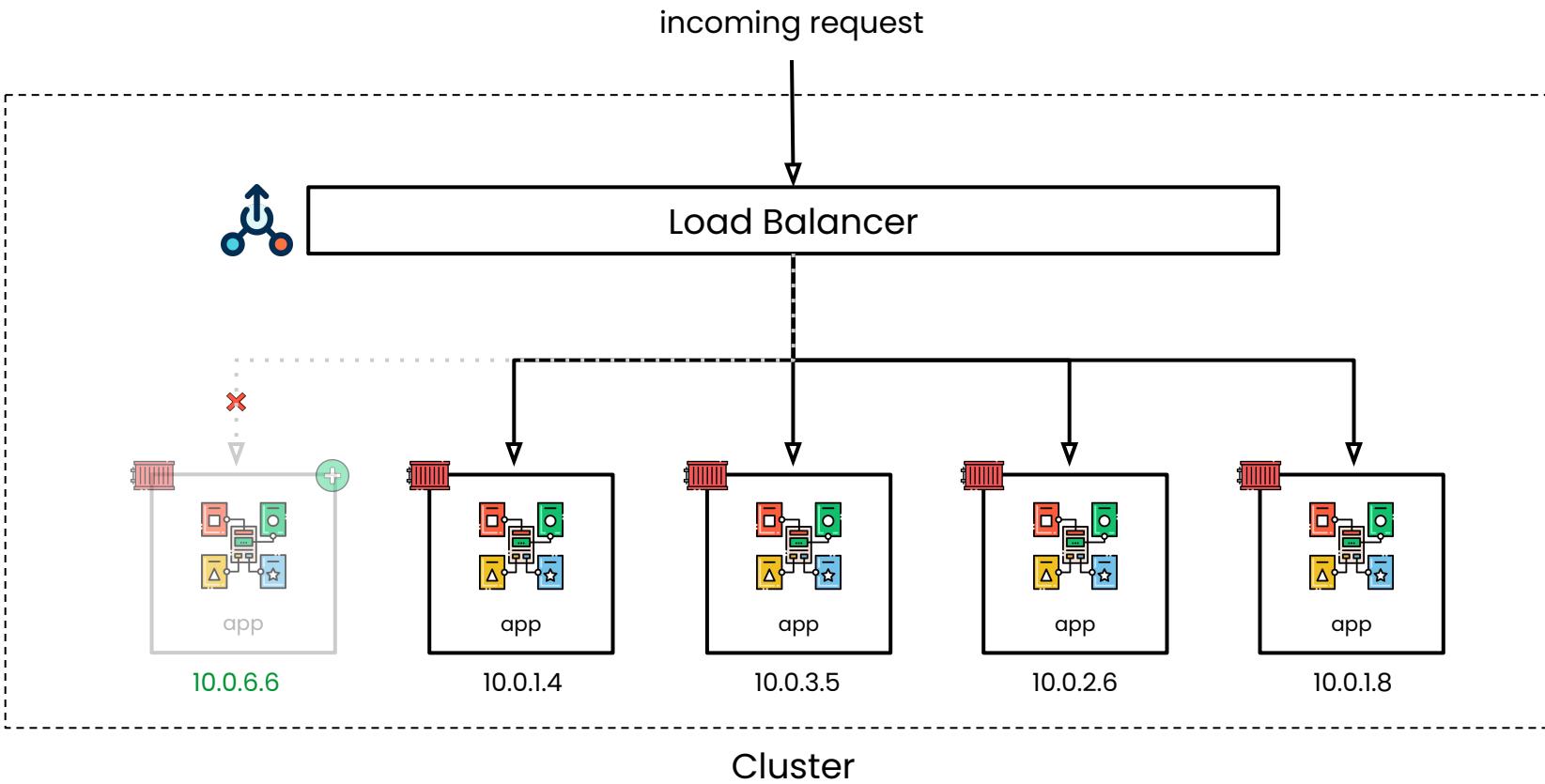




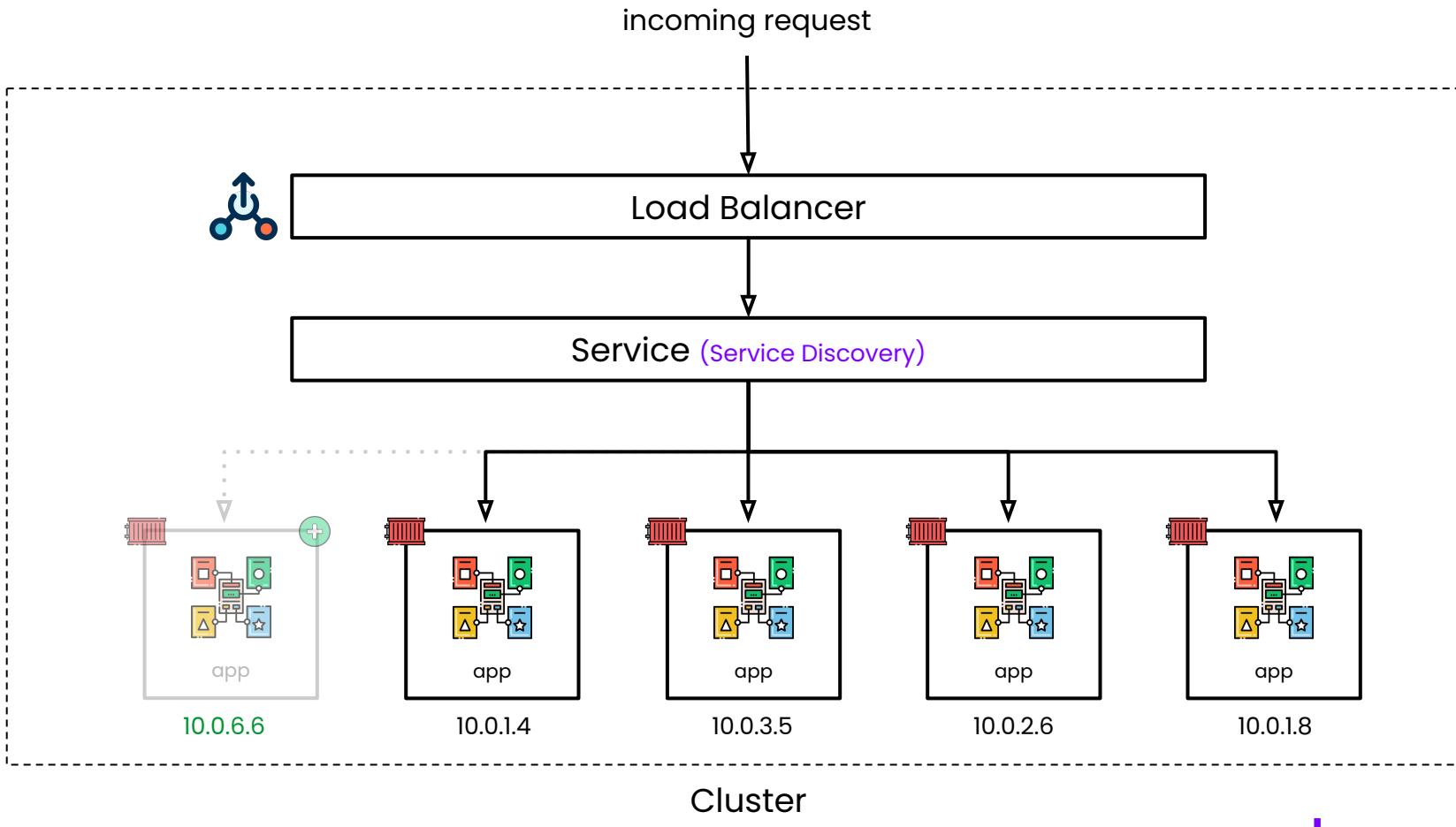




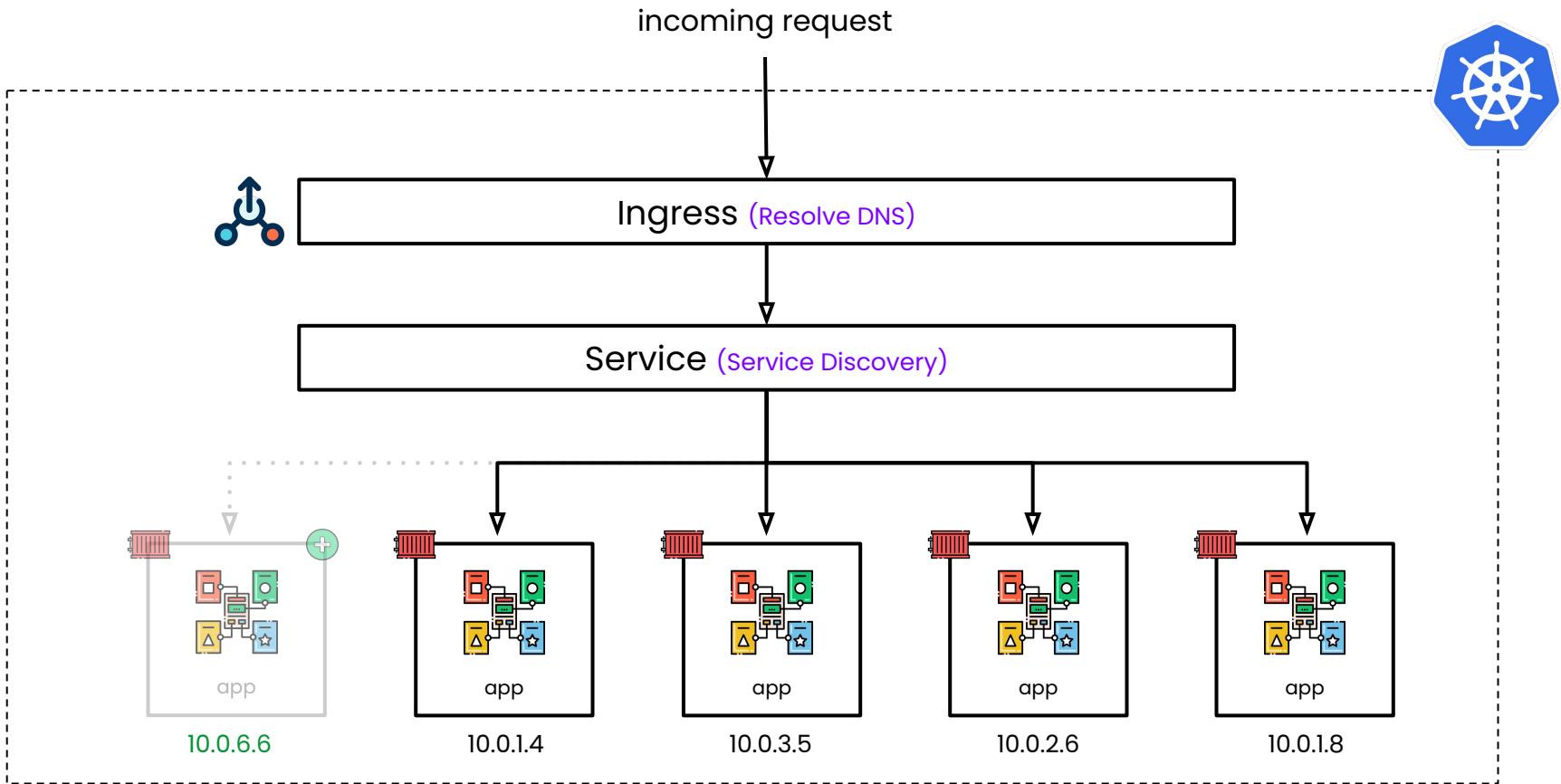
# IP has been changed



# Need to register IP to Loadbalancer



**and now **Kubernetes** comes in**



Kubernetes  
(Orchestrator)

# Kubernetes Overview

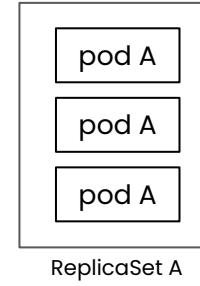
# Kubernetes Overview



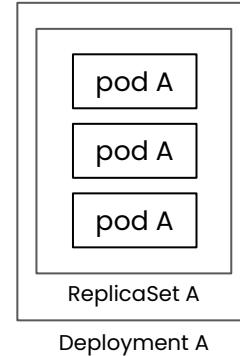
# Kubernetes Overview



# Kubernetes Overview



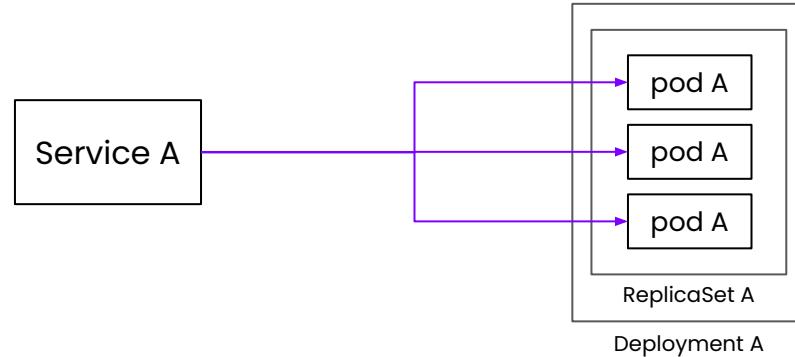
# Kubernetes Overview



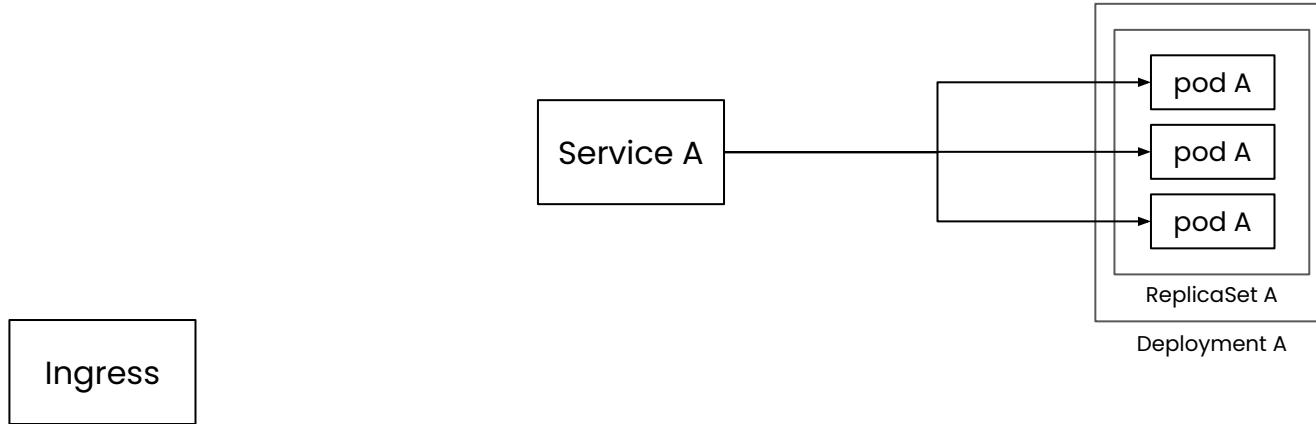
# Kubernetes Overview



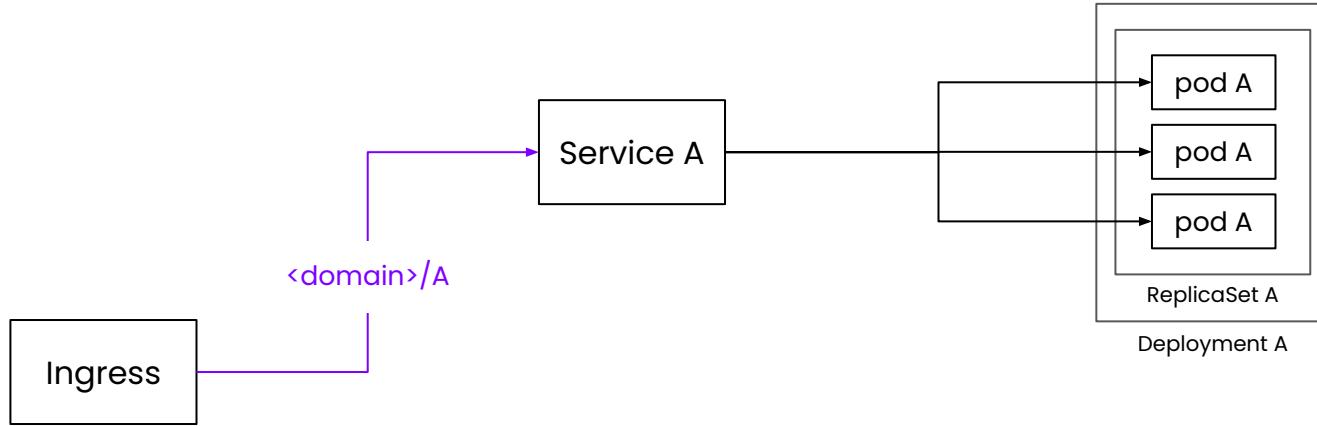
# Kubernetes Overview



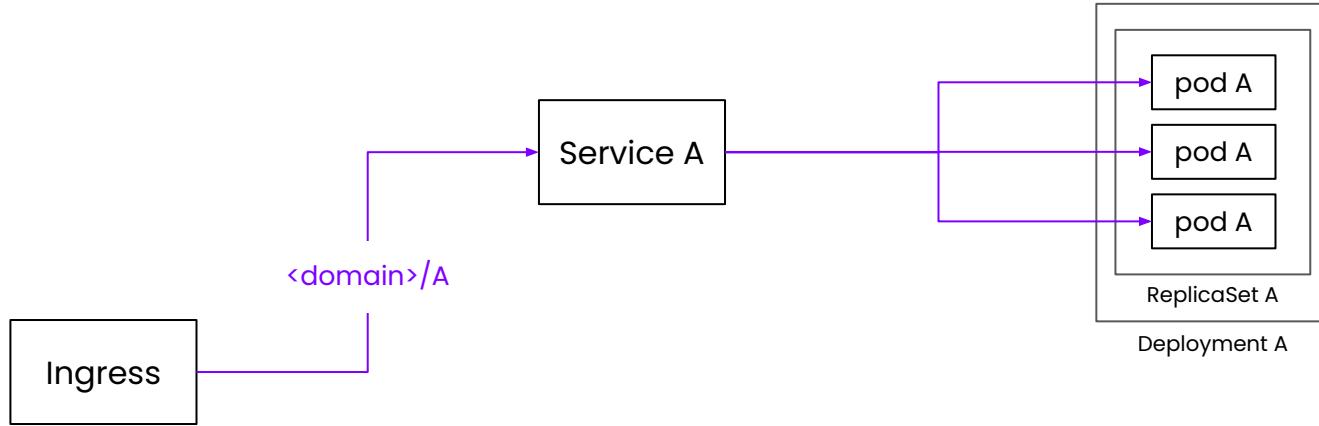
# Kubernetes Overview



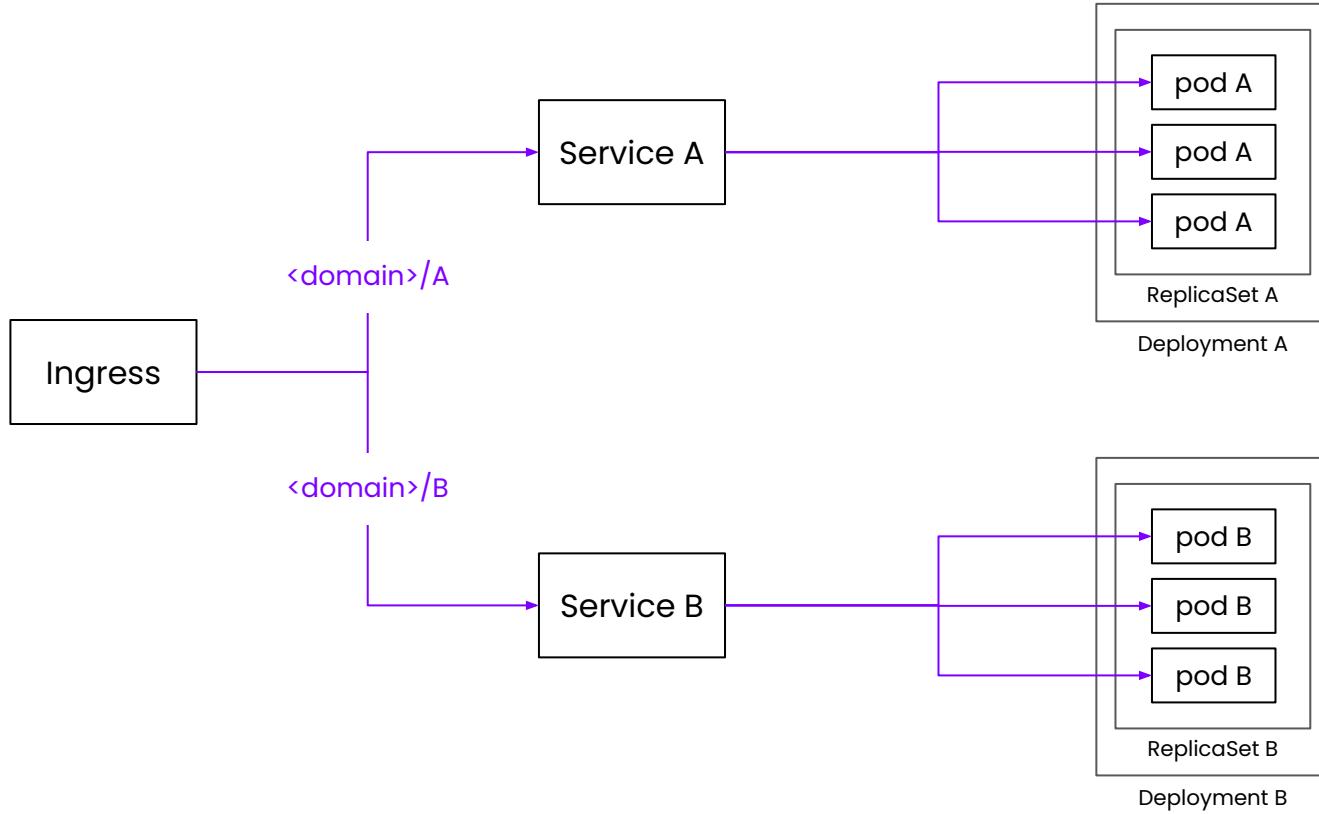
# Kubernetes Overview



# Kubernetes Overview



# Kubernetes Overview



# Kubernetes Overview

- Kubernetes Fundamental -

# Kubernetes Overview (Cont.)

1. Pods
2. Deployment
3. Service
4. Ingress

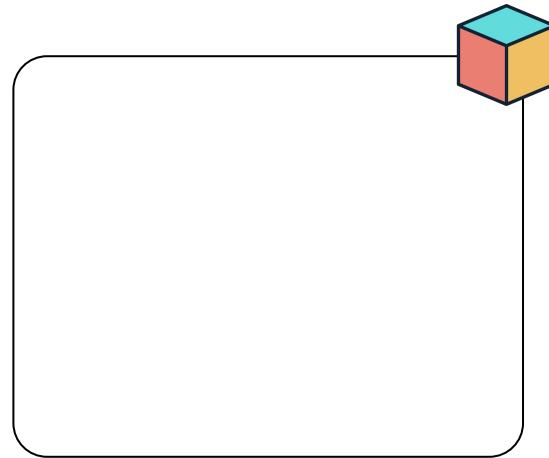
# 1. POD

**Jumpbox®**

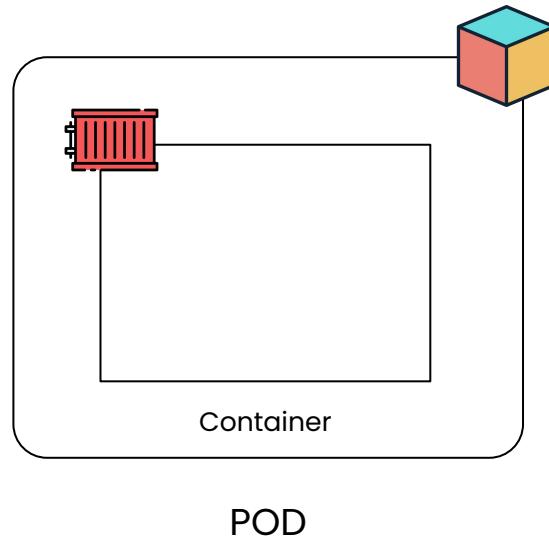
เจ้าของลิขสิทธิ์ อนุญาติให้ใช้สิ่งดังนี้เพื่อการเรียนรู้ส่วนบุคคลเท่านั้น และขอสงวนลิขสิทธิ์ ห้ามมิให้เผยแพร่ในที่สาธารณะ ผู้ละเอียด จะถูกดำเนินคดีตามกฎหมาย

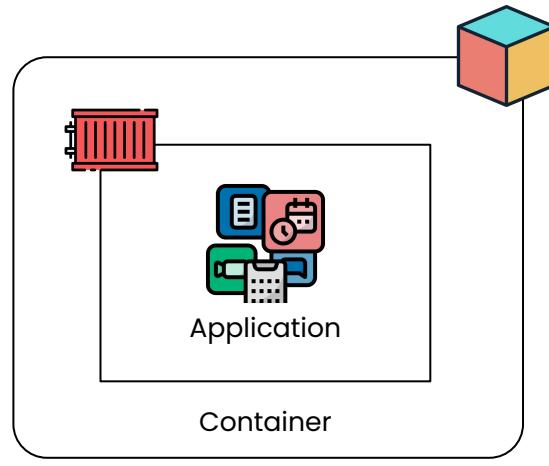
# 1. POD

the **smallest** deployable units of Kubernetes



POD





POD



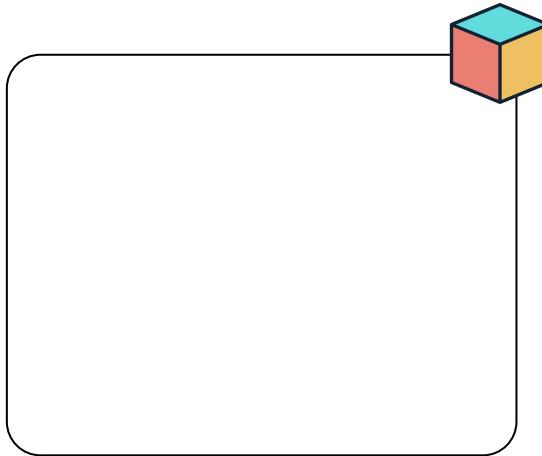
POD of peas

เจ้าของลิขสิทธิ์ อนุญาติให้ใช้สิ่งดังนี้เพื่อการเรียนรู้ส่วนบุคคลเท่านั้น และขอสงวนลิขสิทธิ์ ห้ามมิให้เผยแพร่ในที่สาธารณะ ผู้ละเอียด จะถูกดำเนินคดีตามกฎหมาย

**Jumpbox®**



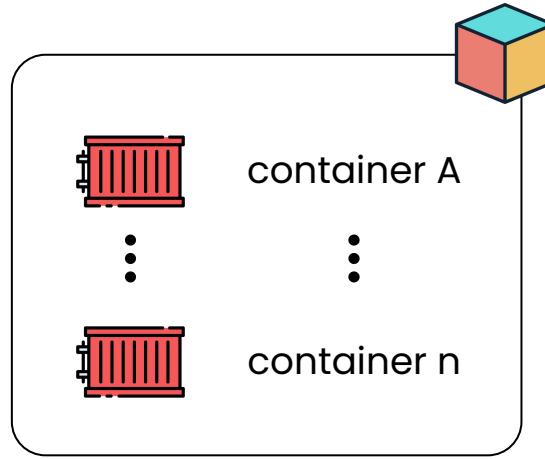
POD of peas



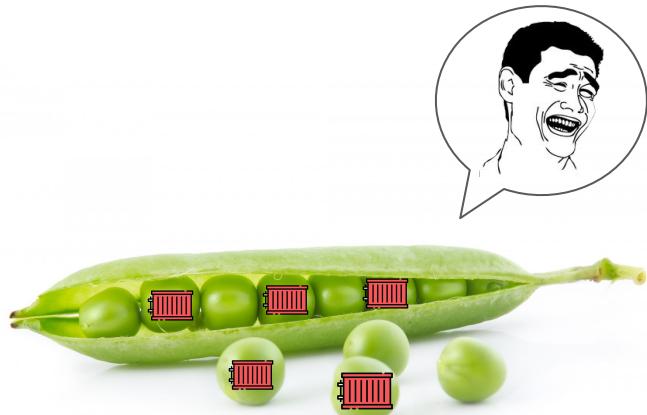
POD



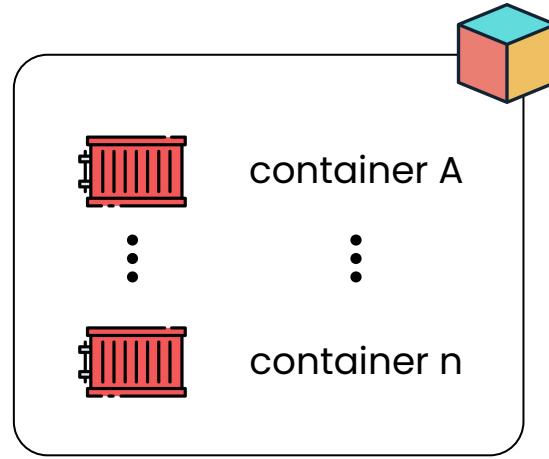
POD of peas



POD of Containers



POD of peas



POD of Containers

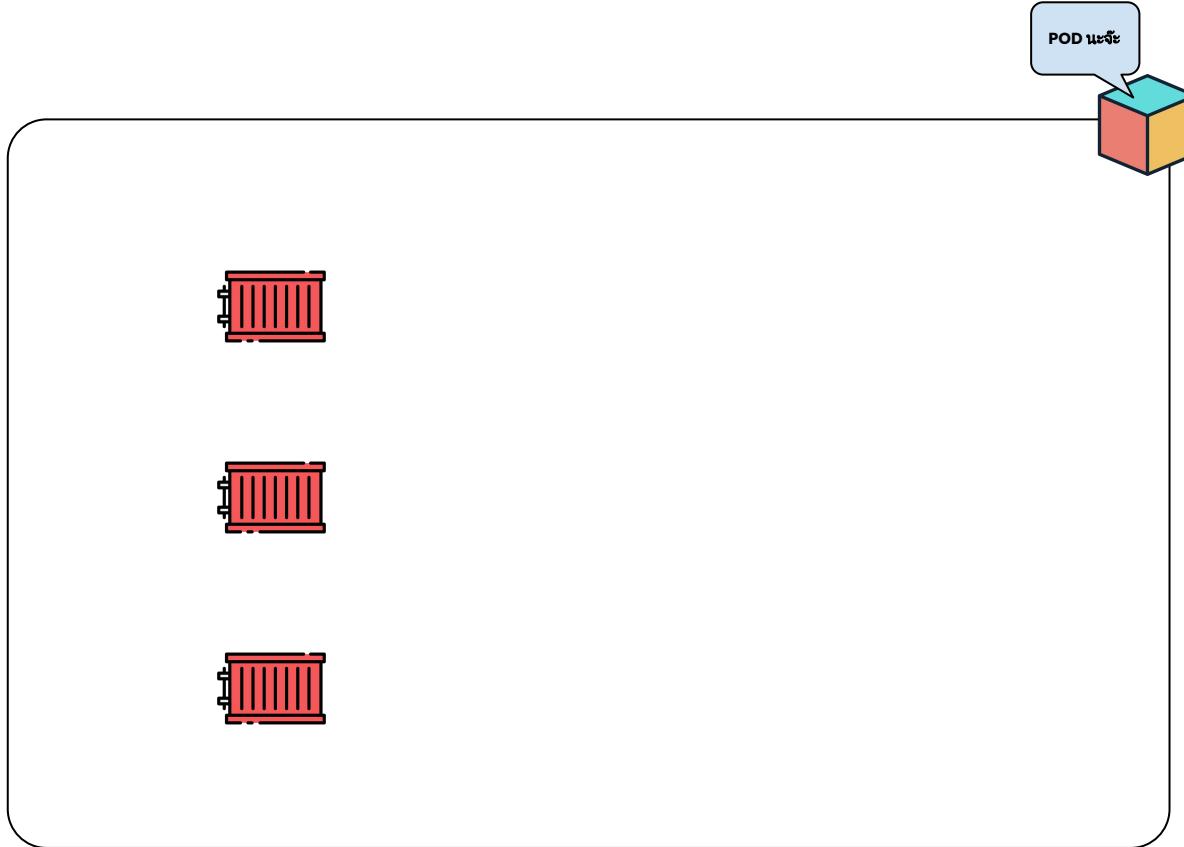
# **Containers in POD**

# **Containers in POD – Storage**

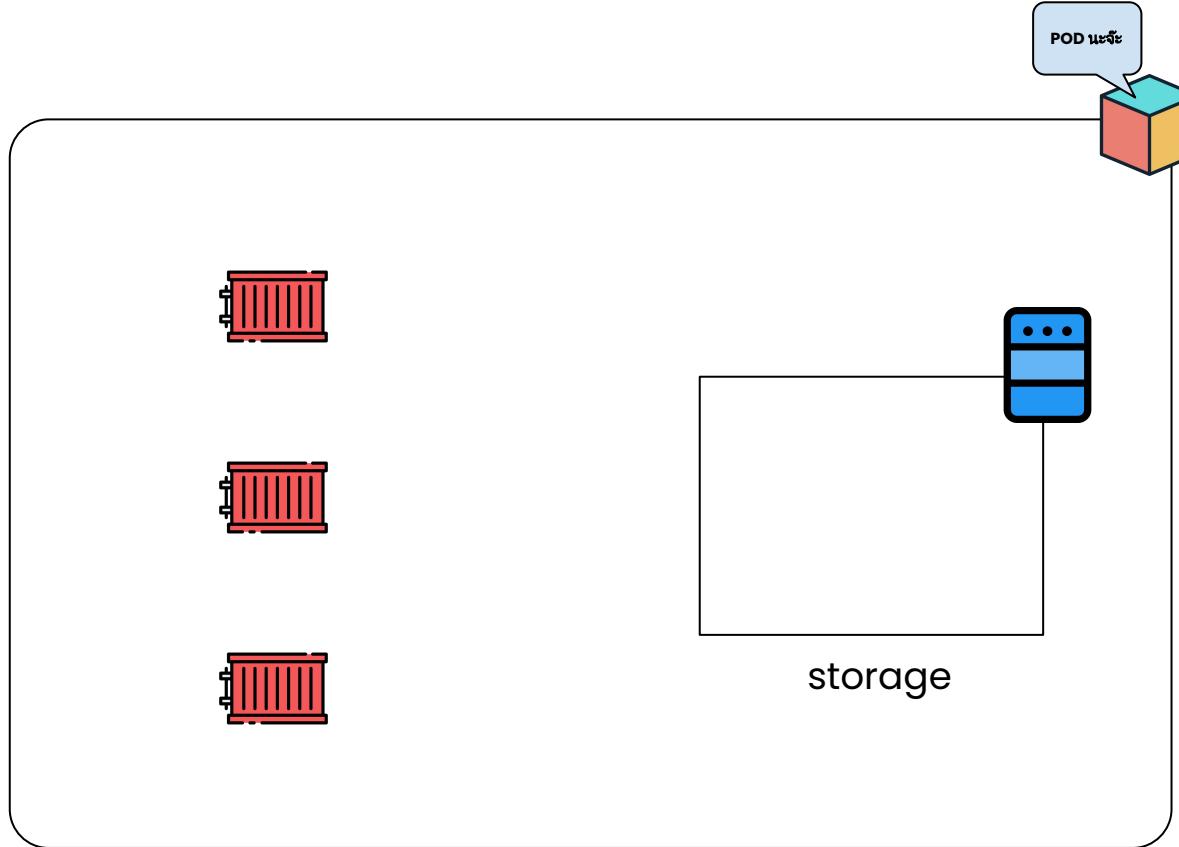
**Jumpbox®**

เจ้าของลิขสิทธิ์ อนุญาติให้ใช้สิ่งดังนี้เพื่อการเรียนรู้ส่วนบุคคลเท่านั้น และขอสงวนลิขสิทธิ์ ห้ามมิให้เผยแพร่ในที่สาธารณะ ผู้อื่นเดินทาง จะถูกดำเนินคดีตามกฎหมาย

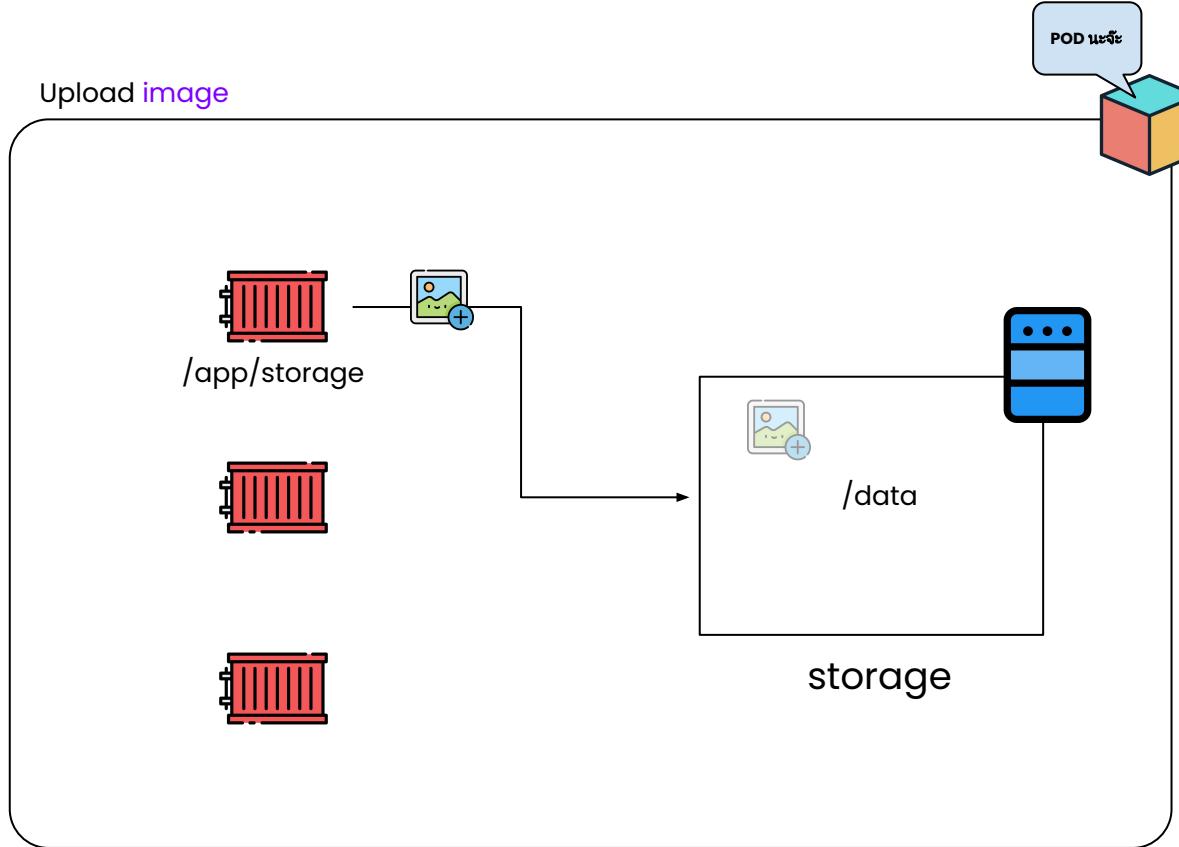
## Containers in POD - Storage (Cont.)



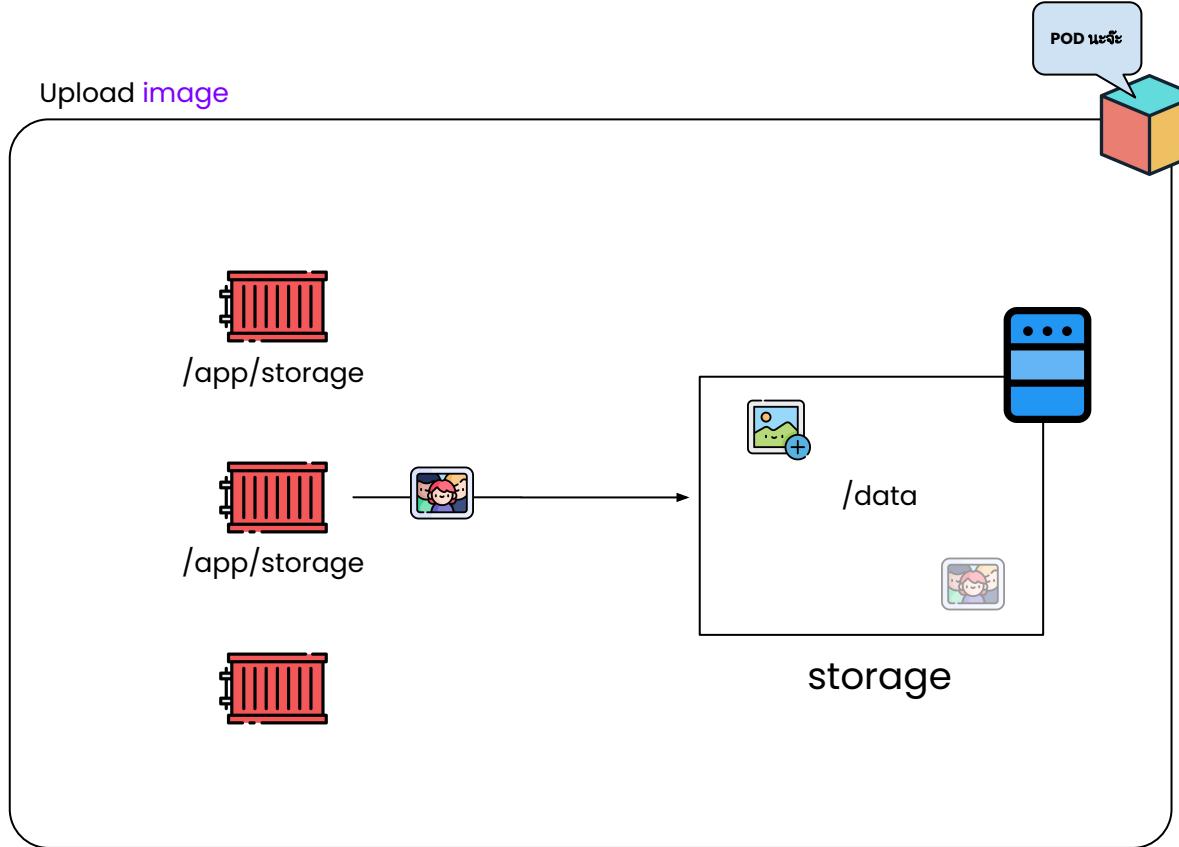
## Containers in POD - Storage (Cont.)



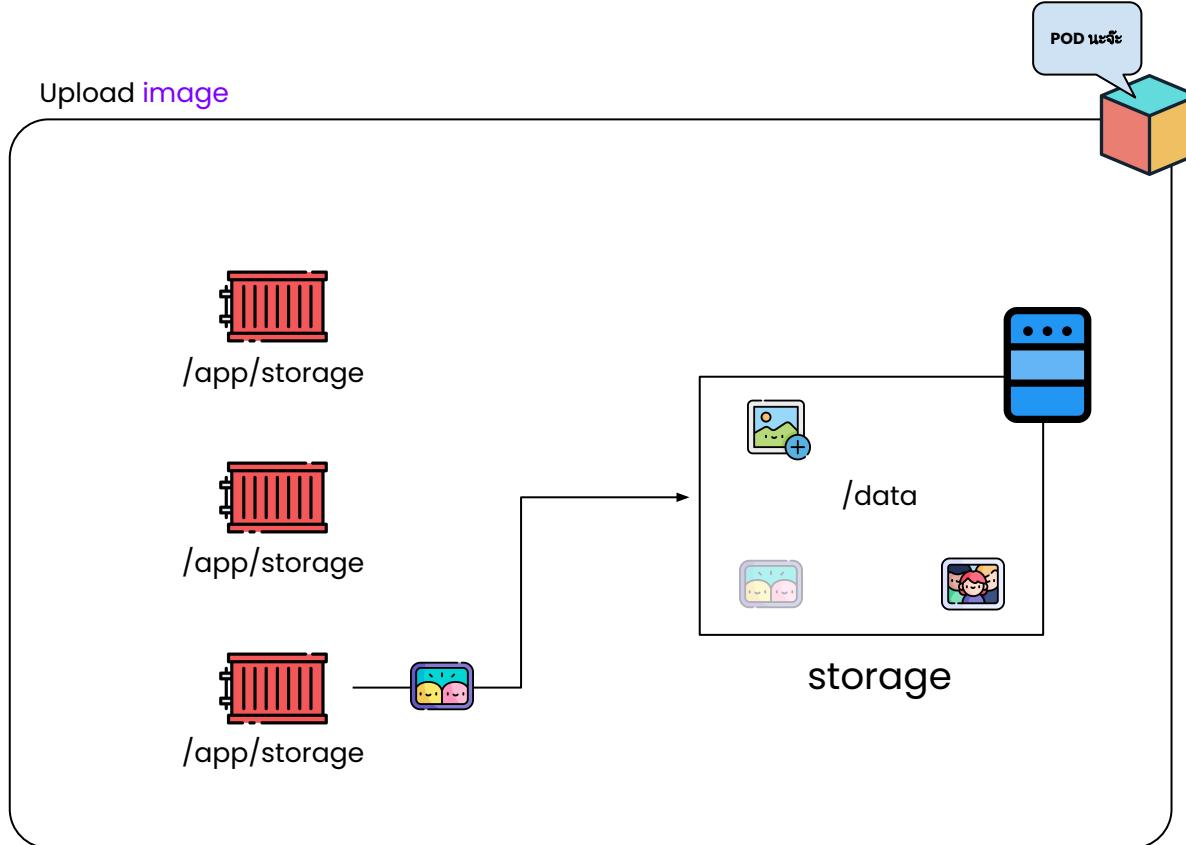
## Containers in POD - Storage (Cont.)



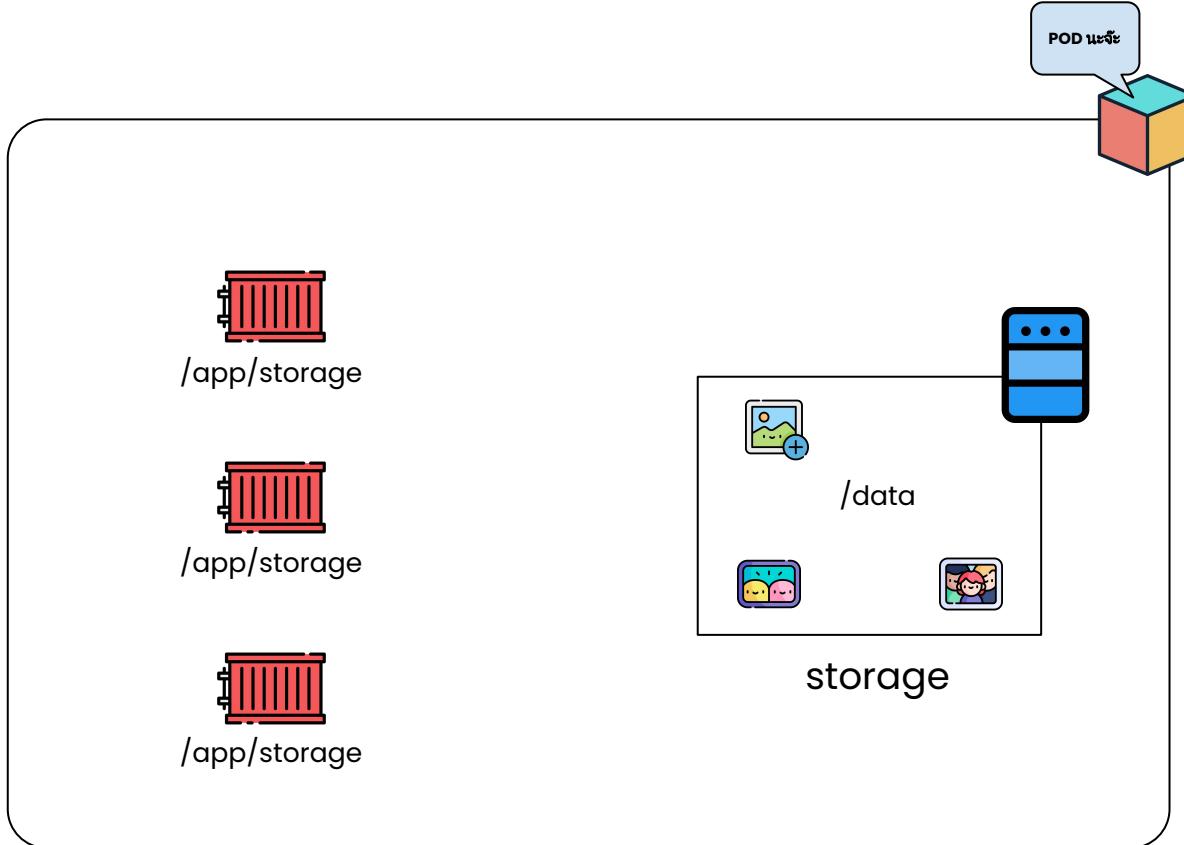
## Containers in POD - Storage (Cont.)



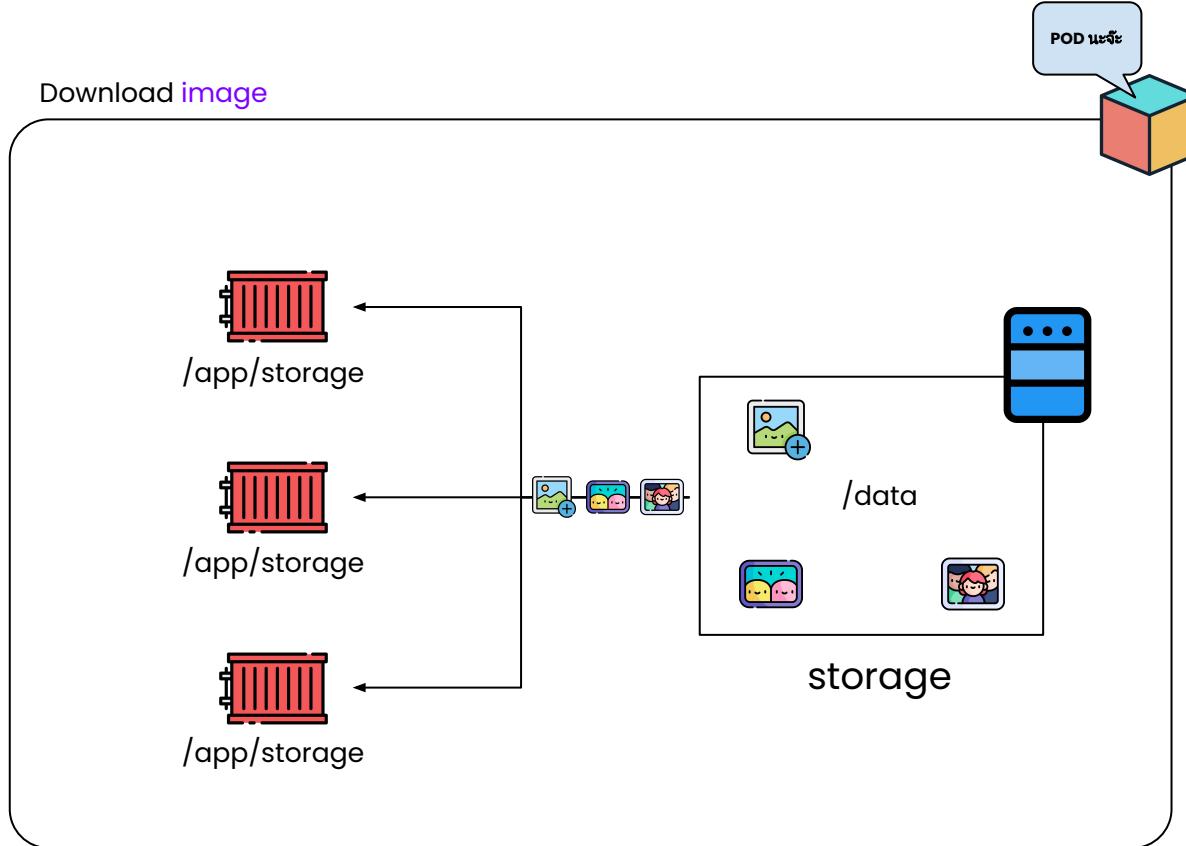
# Containers in POD - Storage (Cont.)



## Containers in POD - Storage (Cont.)



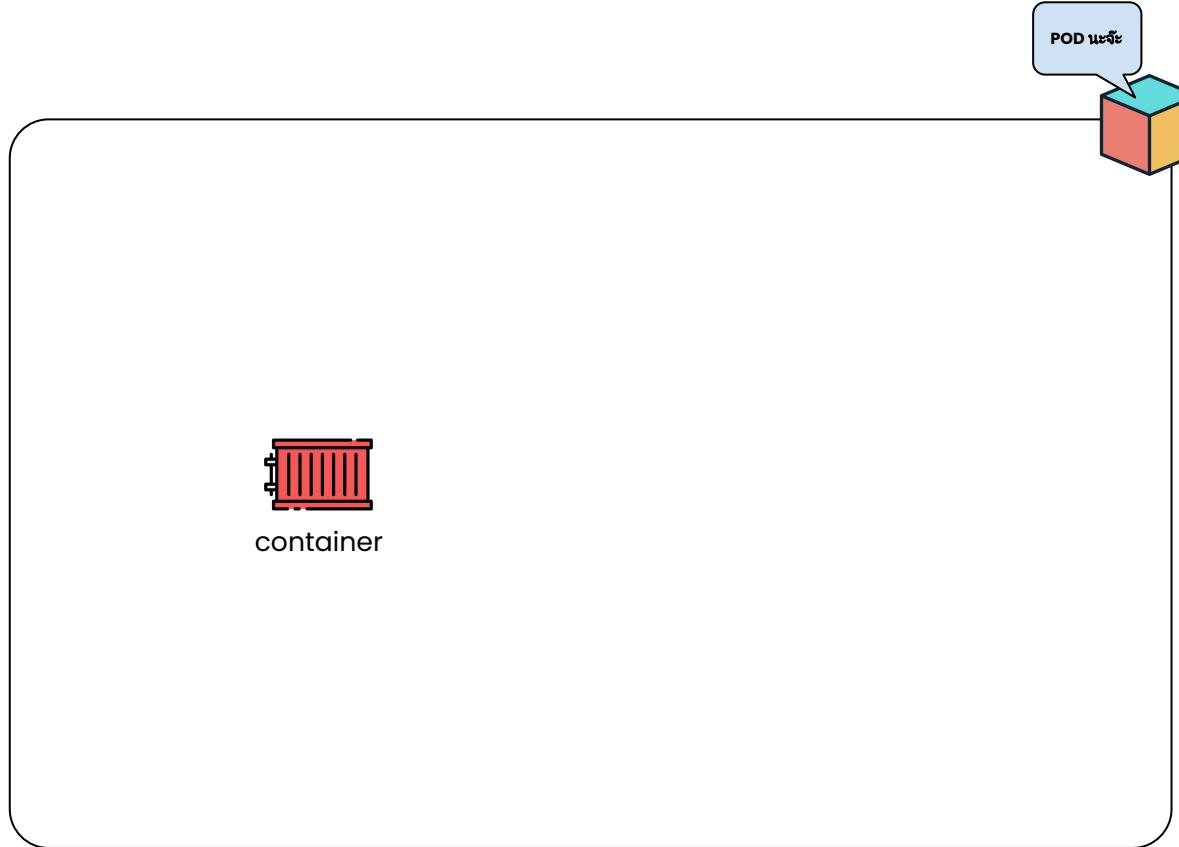
# Containers in POD - Storage (Cont.)



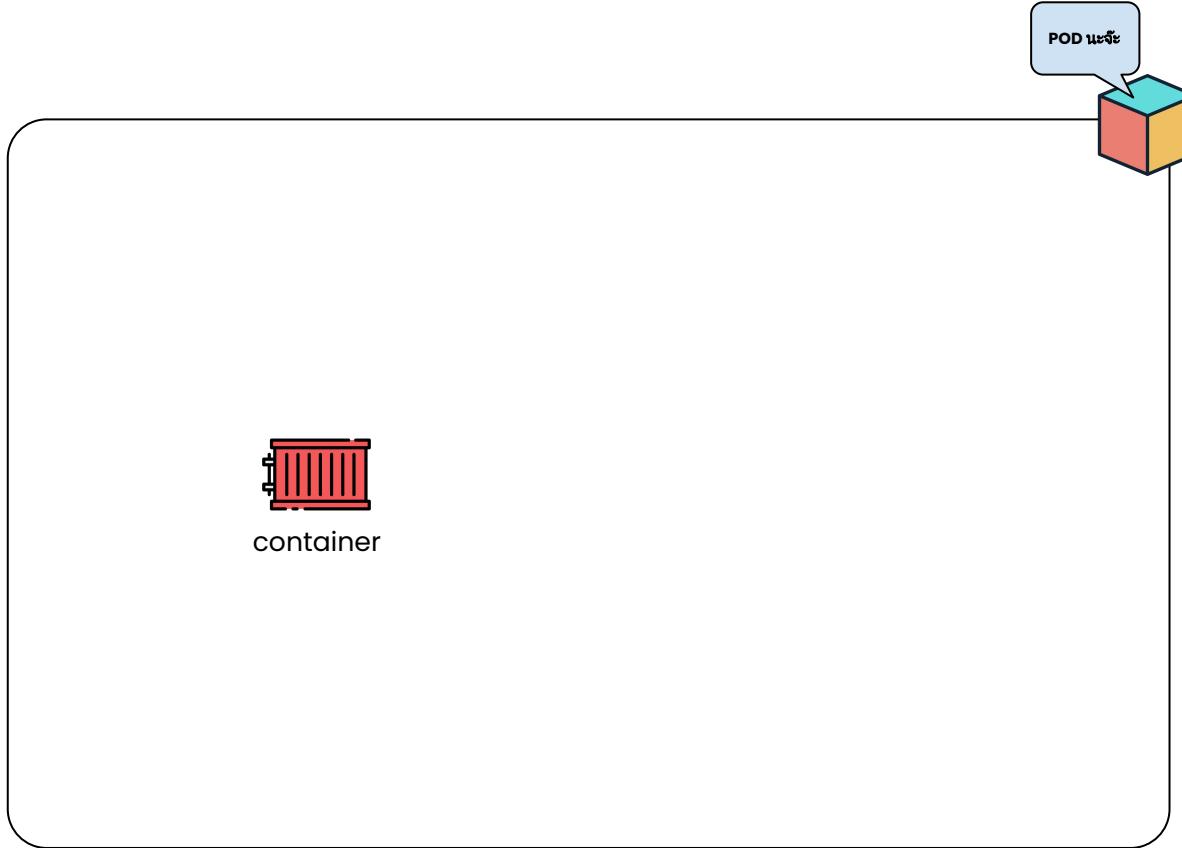
# A container in POD

Jumpbox®

## A container in POD (Cont.)



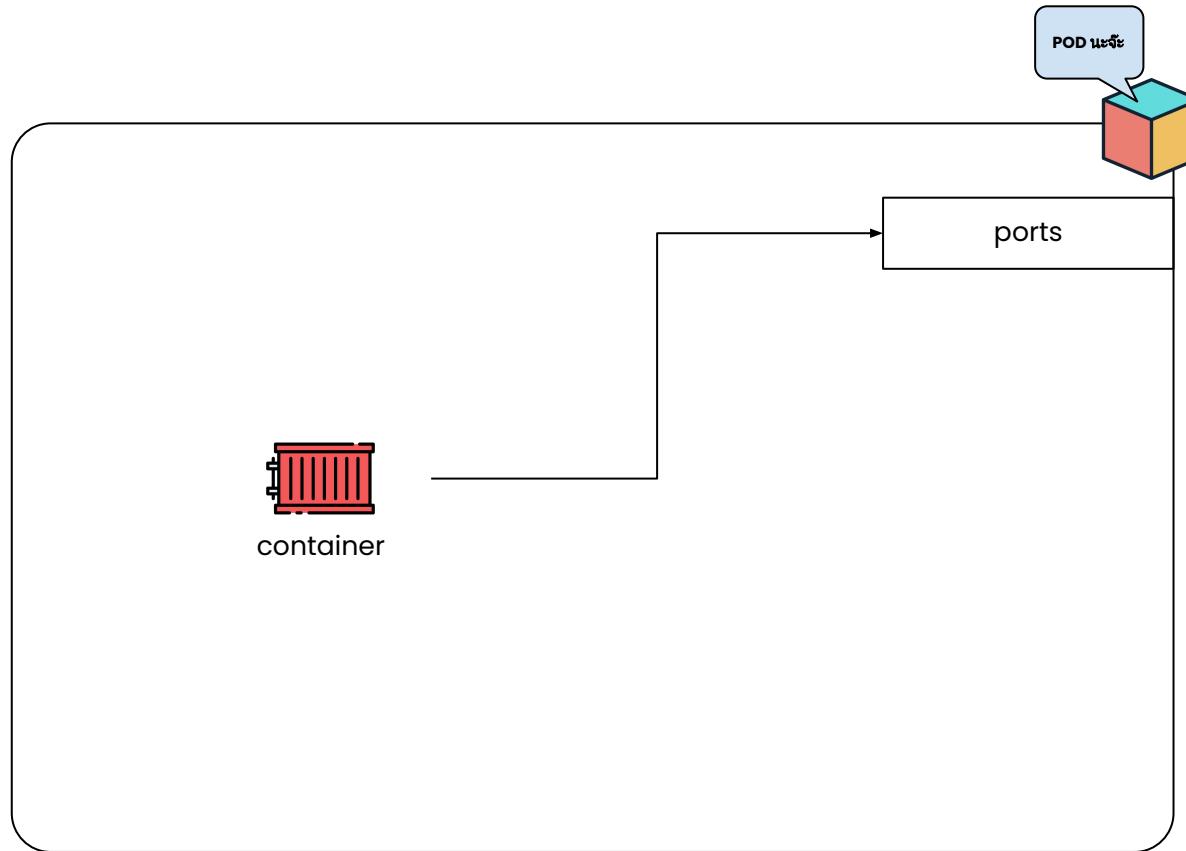
## A container in POD (Cont.)



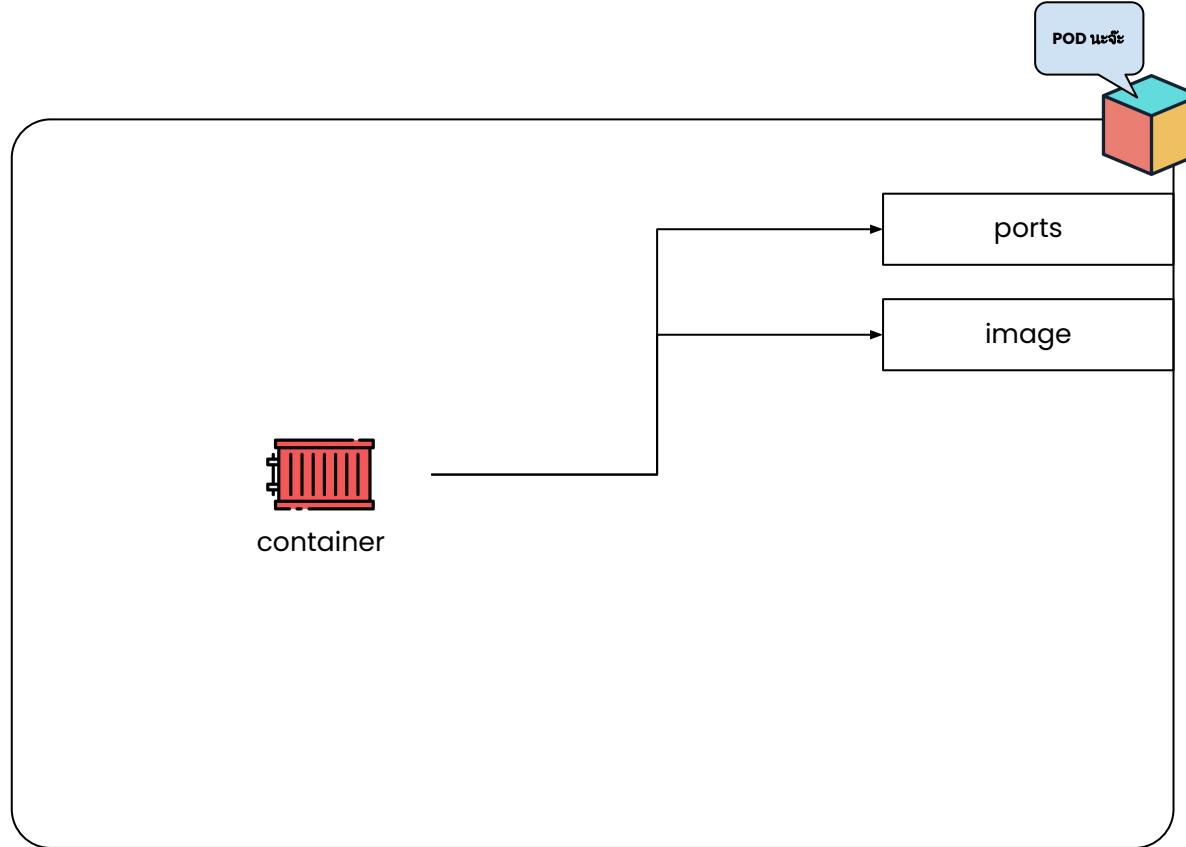
เจ้าของธุรกิจที่อนญาตให้ใช้ส่วนลดนี้เพื่อการเรียนรู้ส่วนบุคคลเท่านั้น และขอสงวนสิทธิ์ห้ามมิให้เผยแพร่ในที่สาธารณะ ผู้ดูแลเมือง จะถูกดำเนินคดีตามกฎหมาย

# Jumpbox®

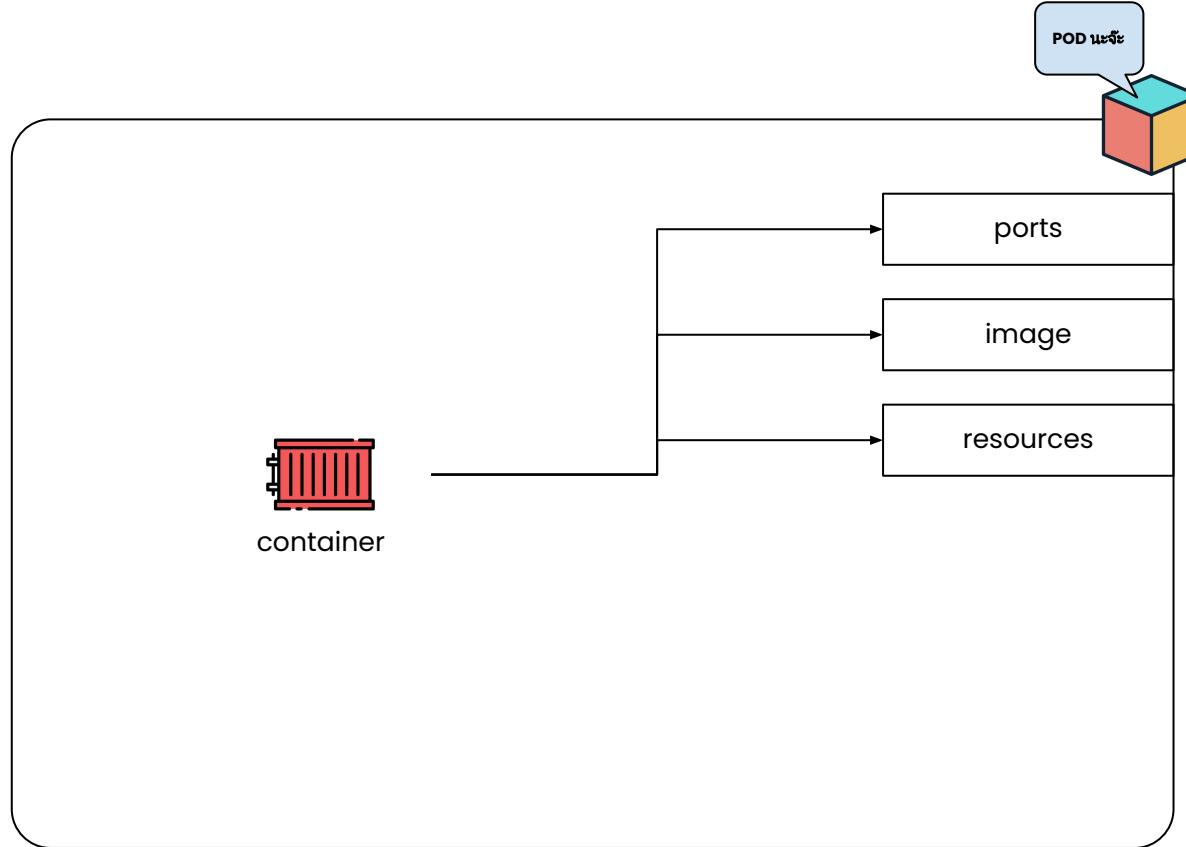
## A container in POD (Cont.)



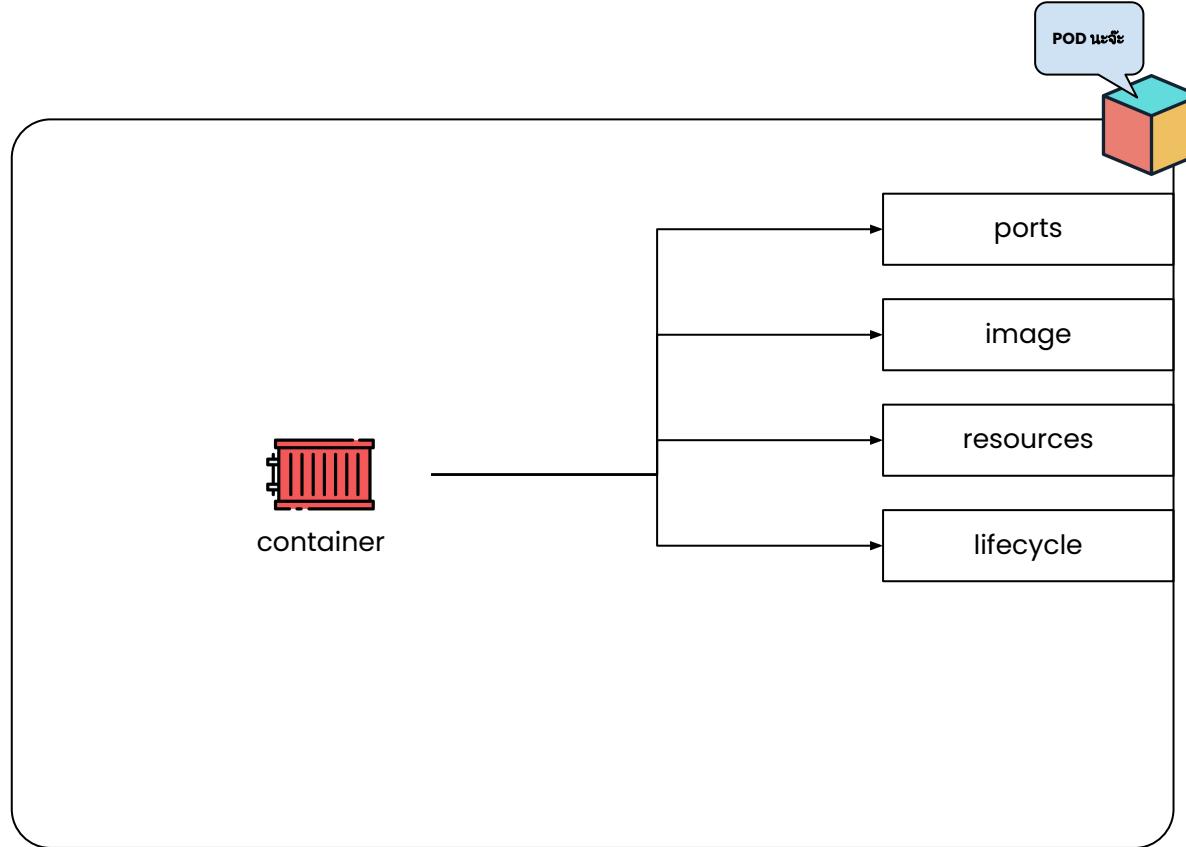
## A container in POD (Cont.)



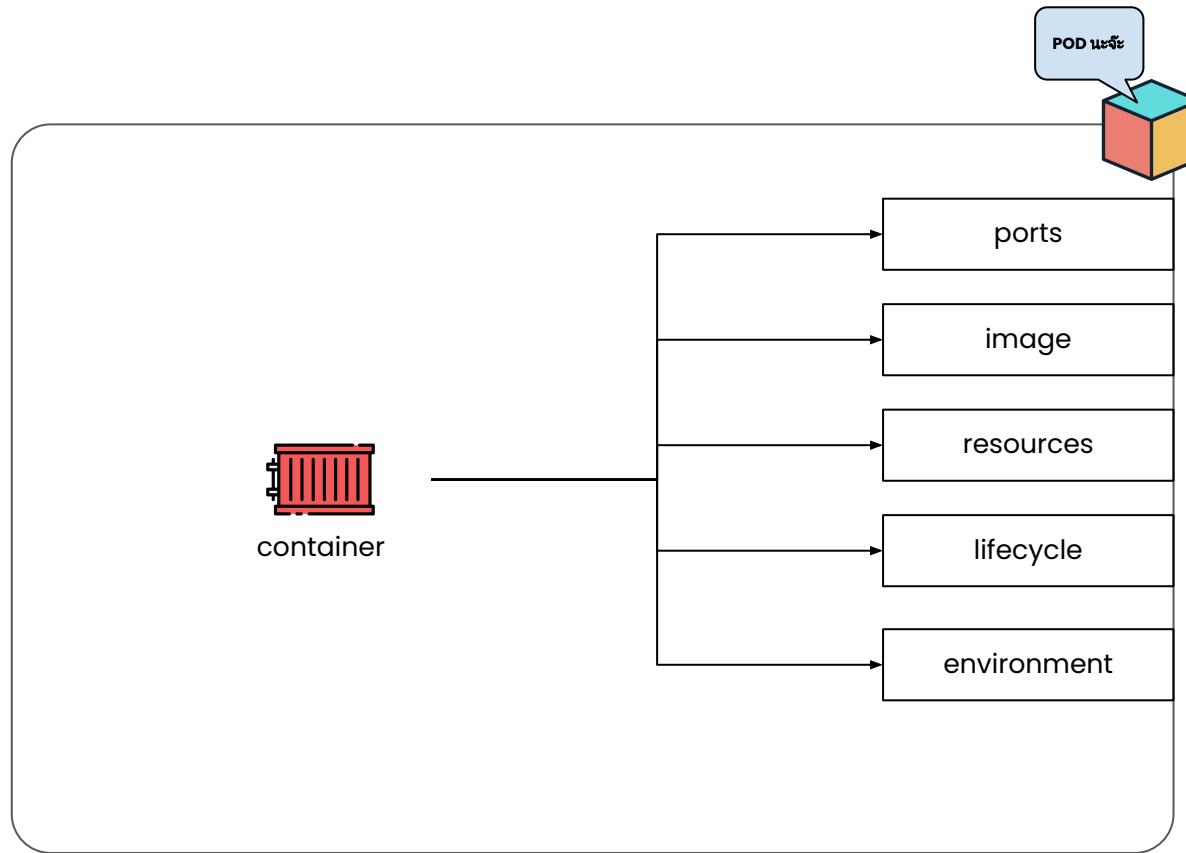
## A container in POD (Cont.)



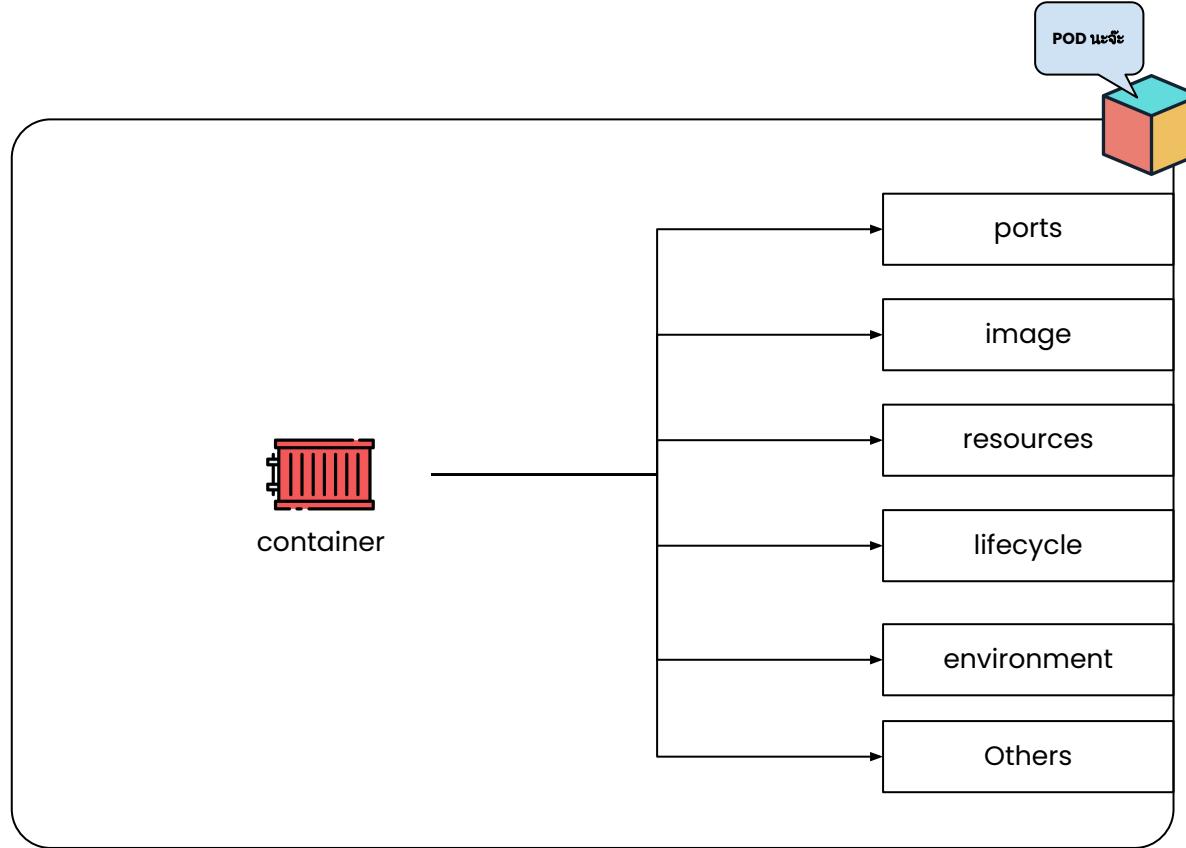
## A container in POD (Cont.)



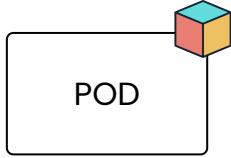
## A container in POD (Cont.)

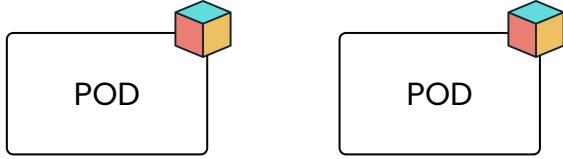


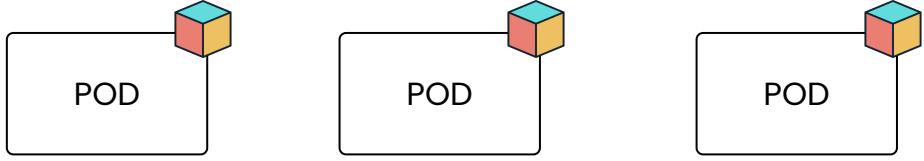
## A container in POD (Cont.)

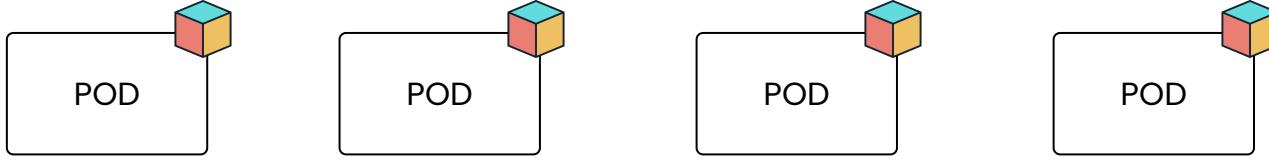


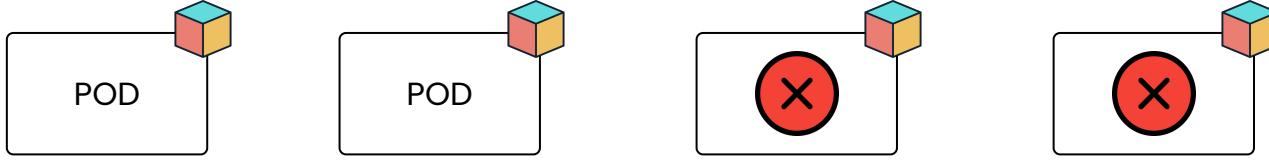
## 2. Deployment

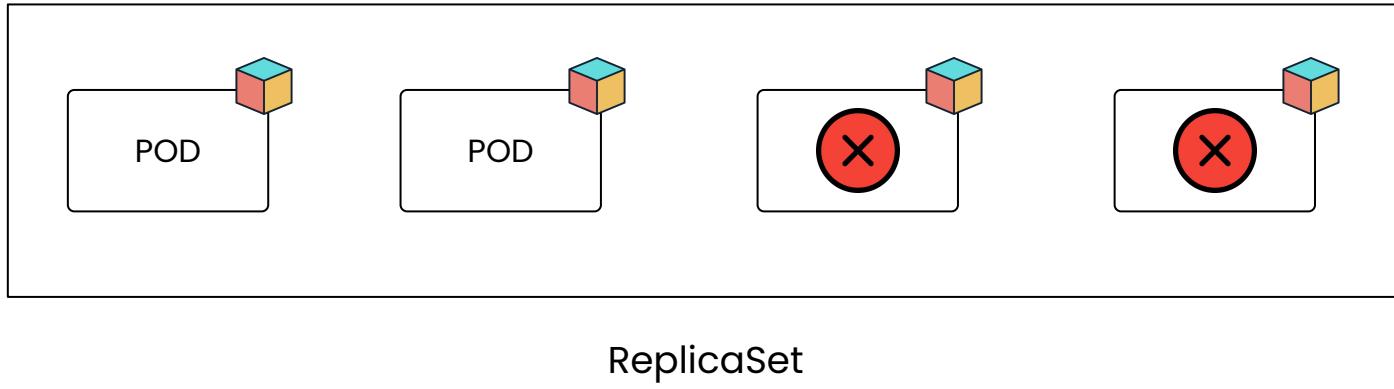


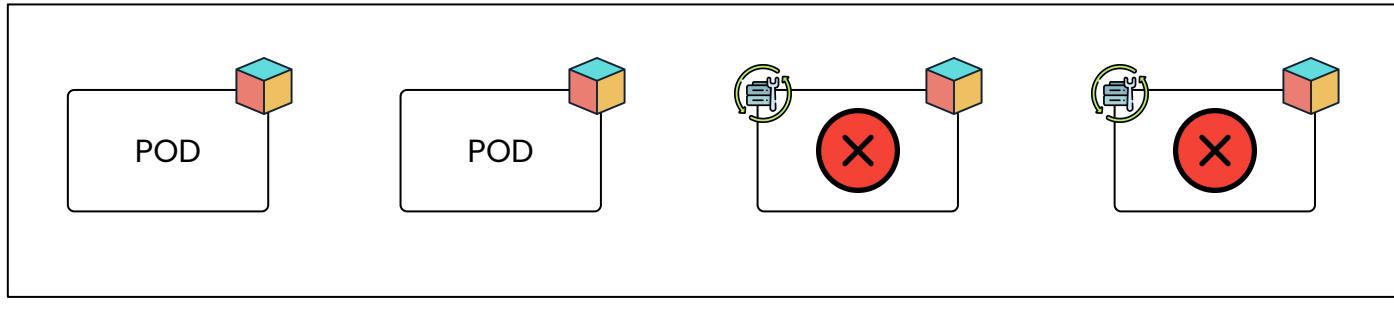




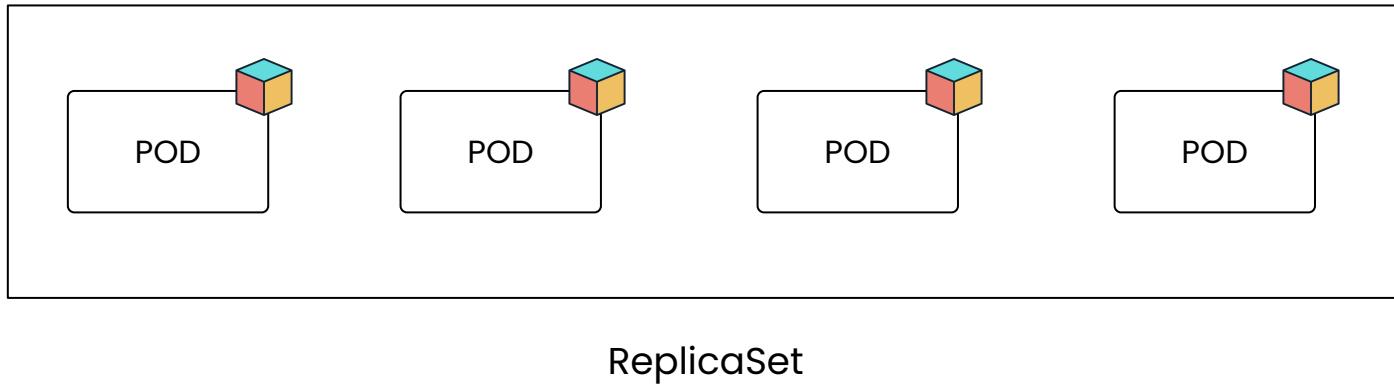








ReplicaSet

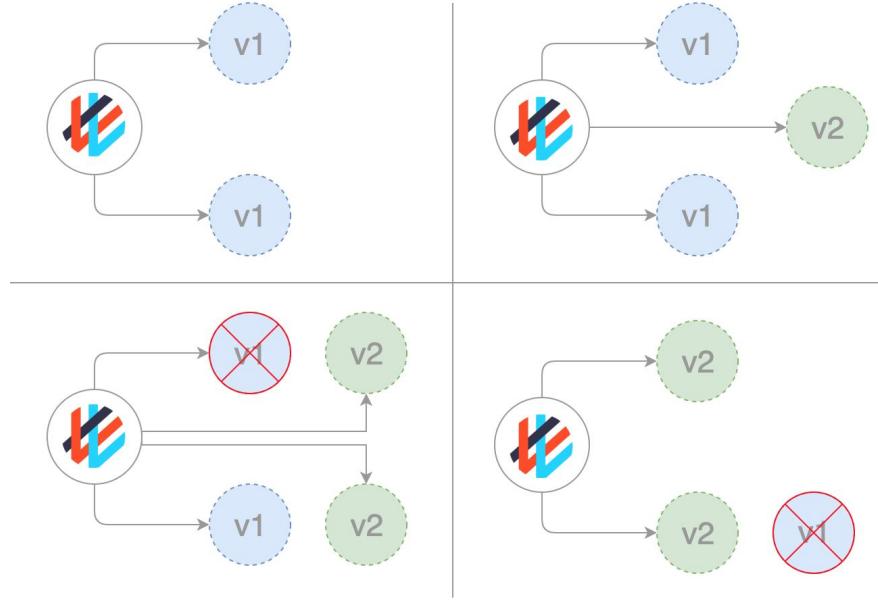


## 2. Deployment

## 2. Deployment

Strategy for deployment process

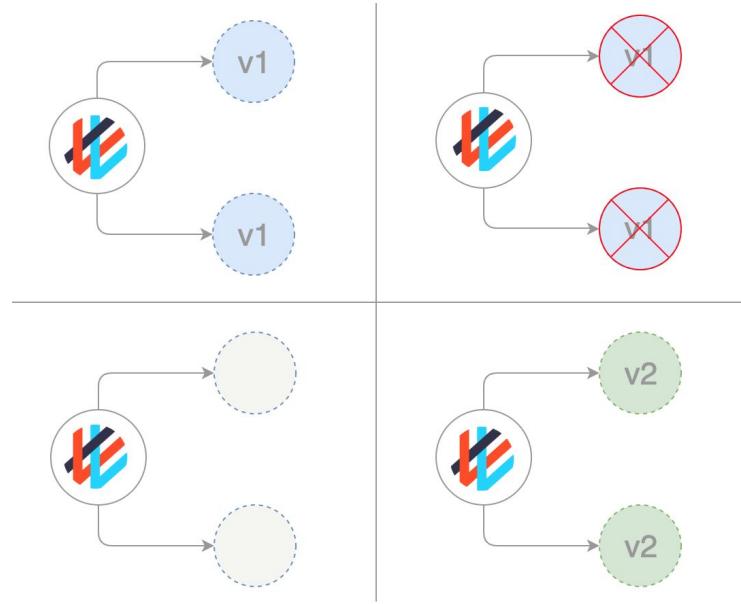
# Rolling Deployment



Reference Pictures:

- [Kubernetes Deployment Strategies](#)

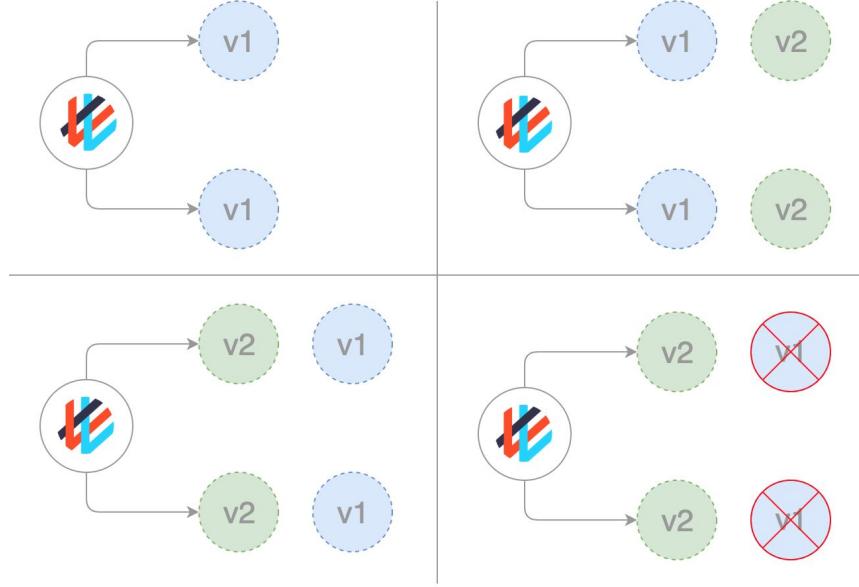
# Recreate



Reference Pictures:

- [Kubernetes Deployment Strategies](#)

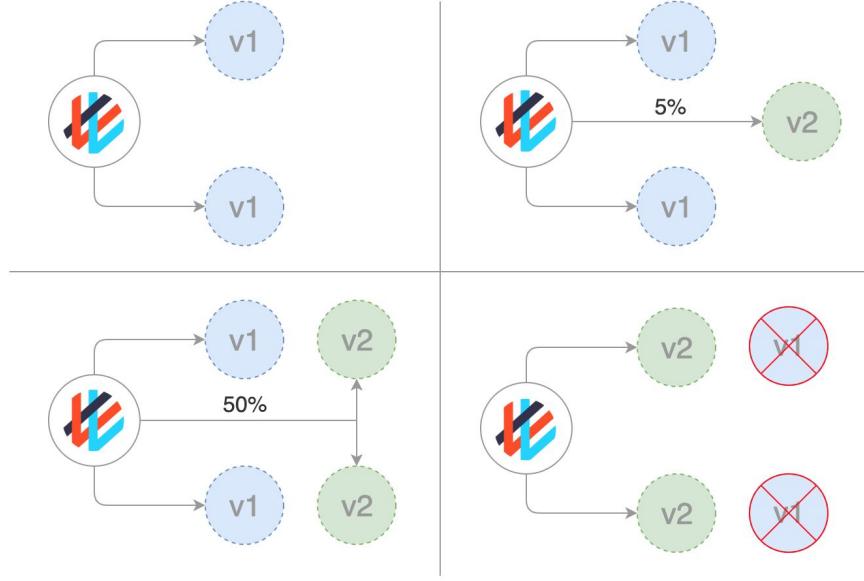
# Blue / Green deployments



Reference Pictures:

- [Kubernetes Deployment Strategies](#)

# Canary



Reference Pictures:

- [Kubernetes Deployment Strategies](#)

# Deployment strategy type

- **strategy** (DeploymentStrategy)

*Patch strategy: retainKeys*

The deployment strategy to use to replace existing pods with new ones.

*DeploymentStrategy describes how to replace existing pods with new ones.*

- **strategy.type** (string)

Type of deployment. Can be "Recreate" or "RollingUpdate". Default is RollingUpdate.

- **strategy.rollingUpdate** (RollingUpdateDeployment)

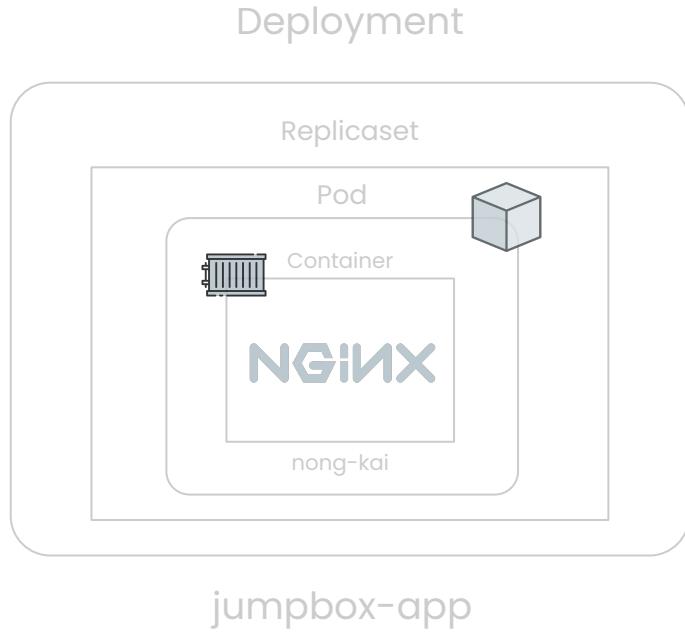
Rolling update config params. Present only if DeploymentStrategyType = RollingUpdate.

*Spec to control the desired behavior of rolling update.*

Reference Pictures:

- [Deployment](#)

# Deployment Strategy Structure (Cont.)



```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: jumpbox-app
spec:
  replicas: 1
  strategy: {}
  selector:
    matchLabels:
      app: jumpbox
  template:
    metadata:
      labels:
        app: jumpbox
    spec:
      containers:
        - image: nginx
          name: nong-kai
```

# Deployment - Strategy (Cont.)

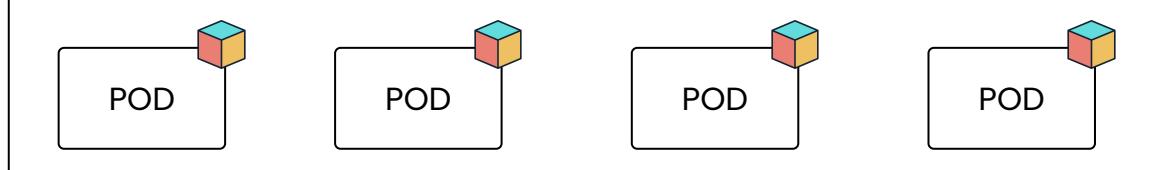


## Deployment

เจ้าของลิขสิทธิ์ อนุญาติให้ใช้สิ่งดังนี้เพื่อการเรียนรู้ส่วนบุคคลเท่านั้น และขอสงวนลิขสิทธิ์ ห้ามมิให้เผยแพร่ในที่สาธารณะ ผู้ละเอียด จะถูกดำเนินคดีตามกฎหมาย

**Jumpbox®**

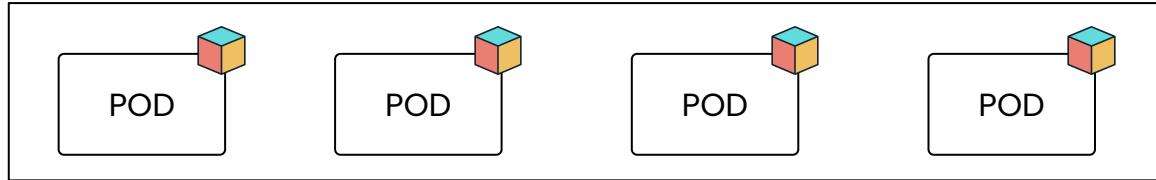
# Deployment - Strategy (Cont.)



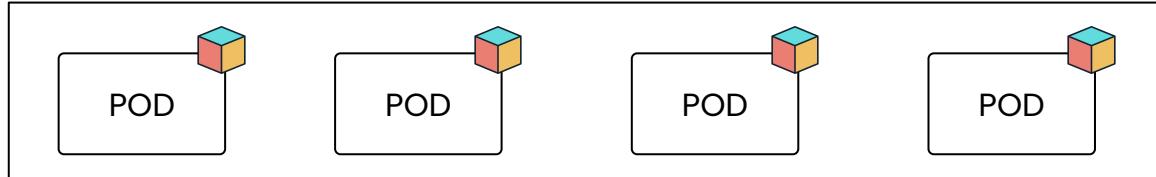
ReplicaSet: v1

Deployment

# Deployment - Strategy (Cont.)



ReplicaSet: v1

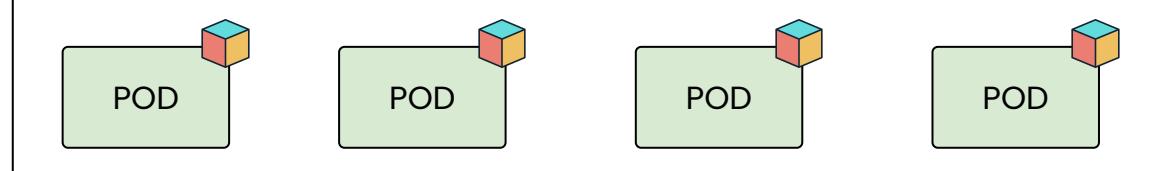


ReplicaSet: v2

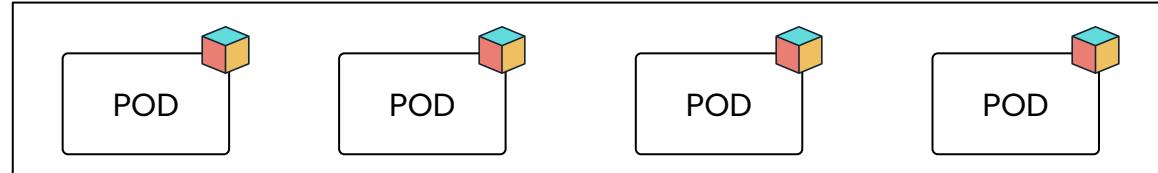
Deployment

# Deployment - Strategy (Cont.)

Green



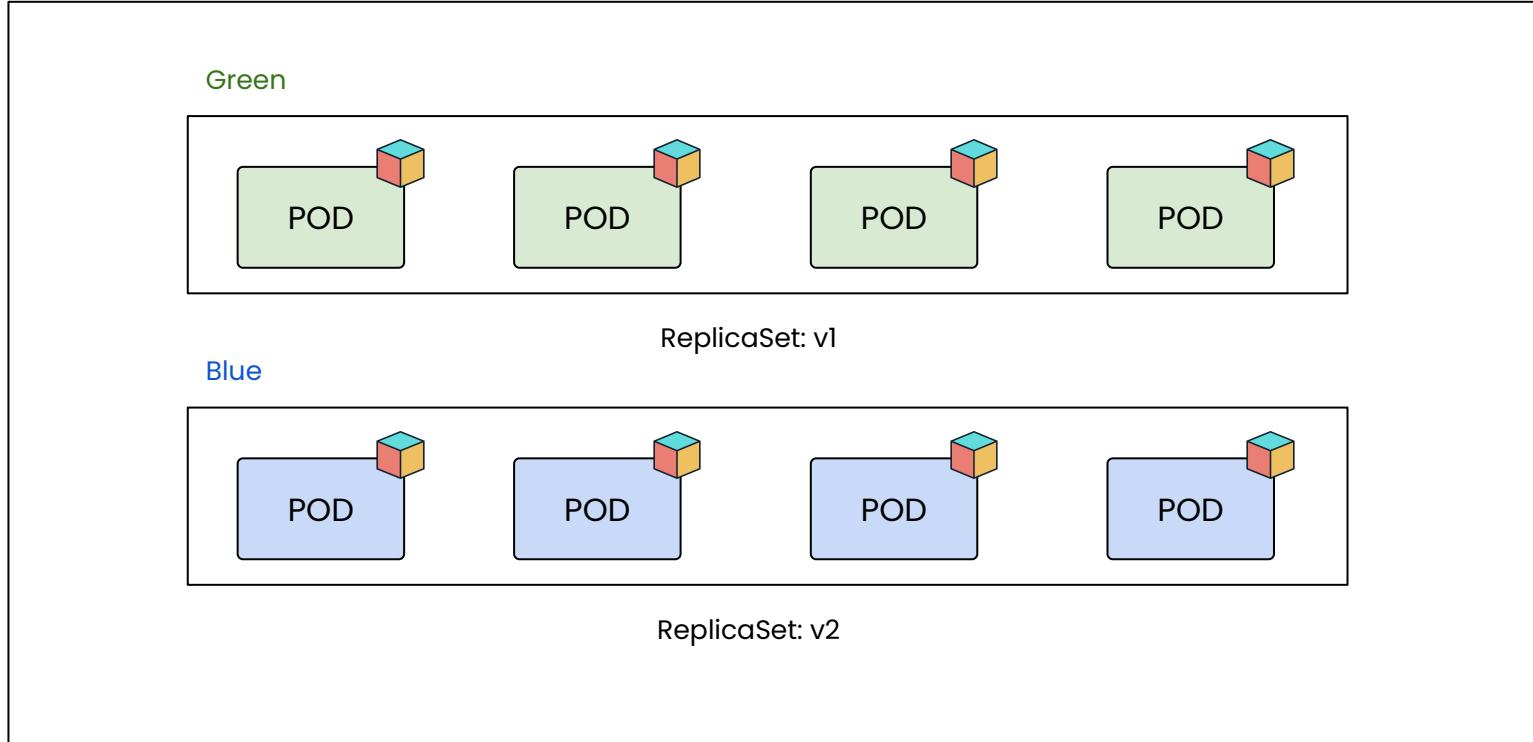
ReplicaSet: v1



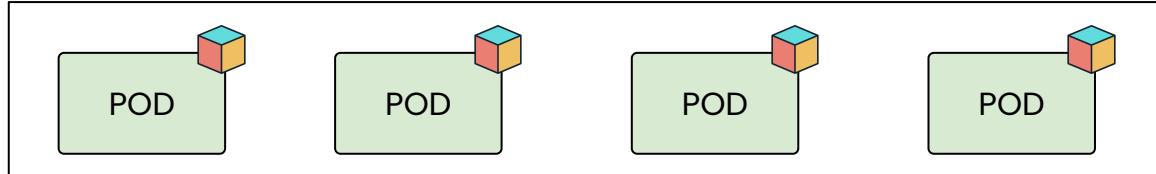
ReplicaSet: v2

Deployment

# Deployment - Strategy (Cont.)



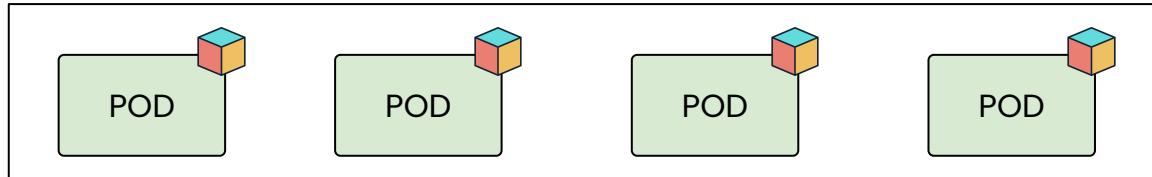
# Deployment - Strategy (Cont.)



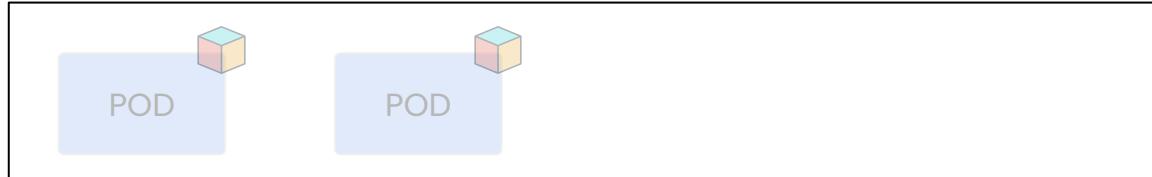
ReplicaSet: v1

Deployment

# Deployment - Strategy (Cont.)



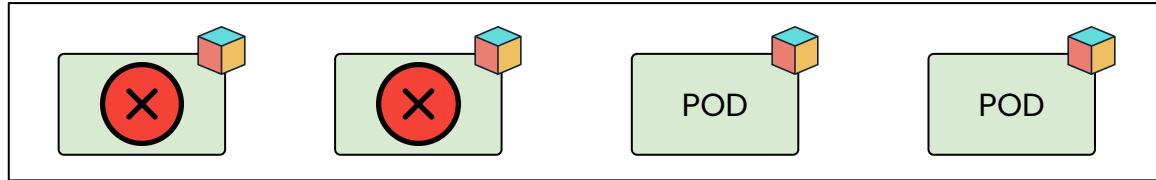
ReplicaSet: v1



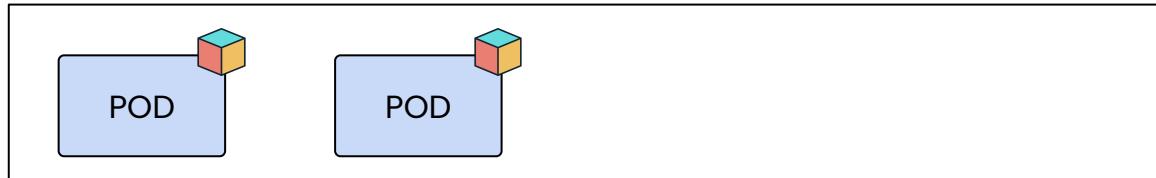
ReplicaSet: v2

Deployment

# Deployment - Strategy (Cont.)



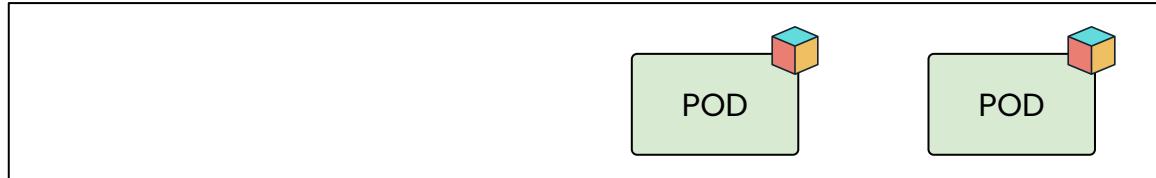
ReplicaSet: v1



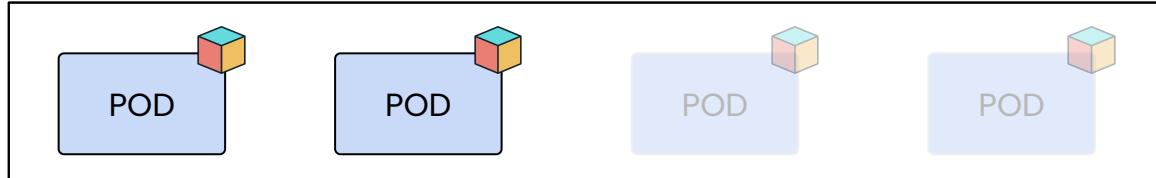
ReplicaSet: v2

Deployment

# Deployment - Strategy (Cont.)



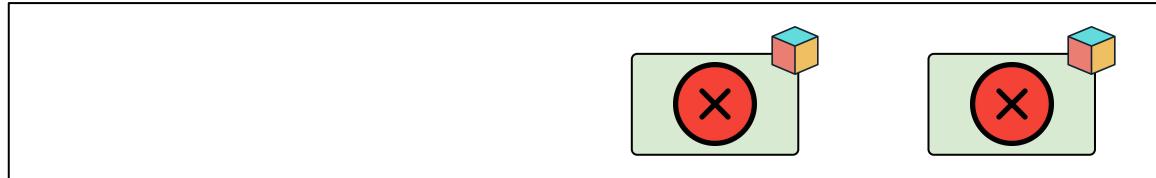
ReplicaSet: v1



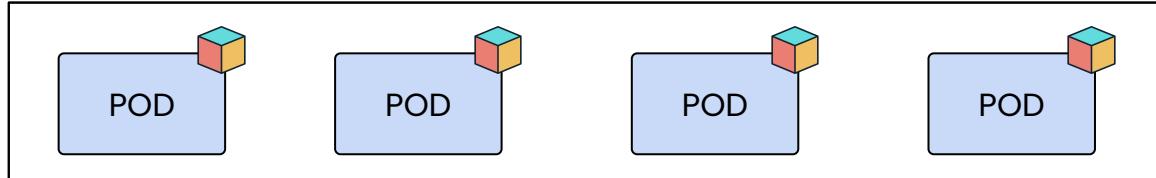
ReplicaSet: v2

Deployment

# Deployment - Strategy (Cont.)



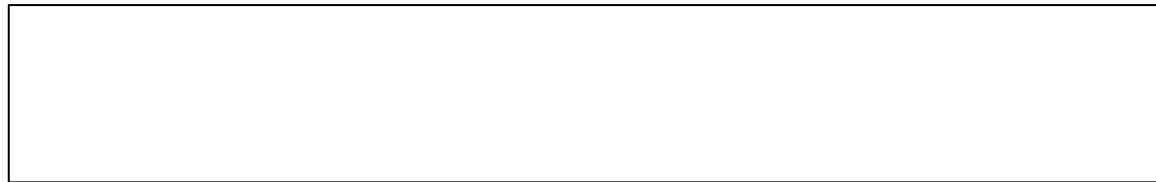
ReplicaSet: v1



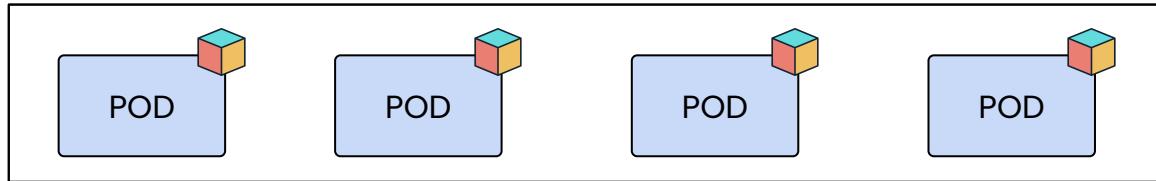
ReplicaSet: v2

Deployment

# Deployment - Strategy (Cont.)



ReplicaSet: v1



ReplicaSet: v2

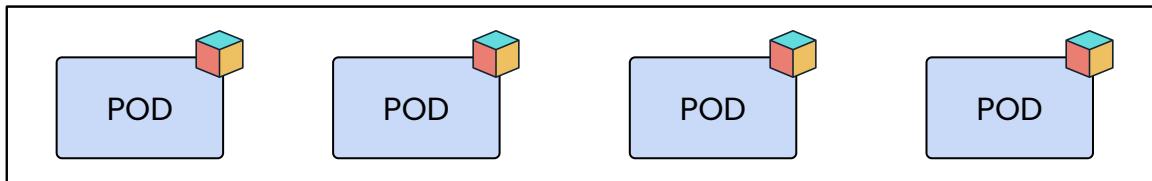
Deployment

# Deployment - Strategy (Cont.)

revisionHistory: 1



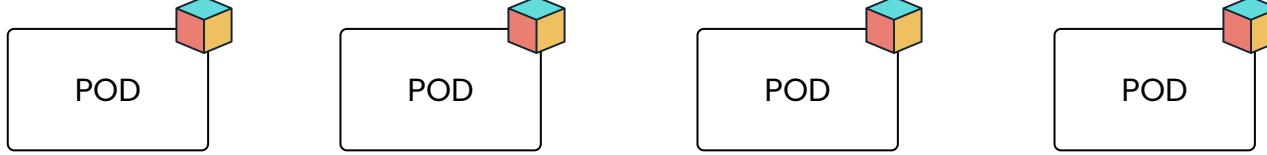
ReplicaSet: v1



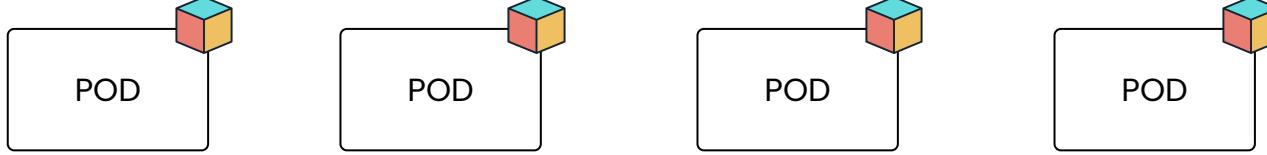
ReplicaSet: v2

Deployment

## **3. Service**

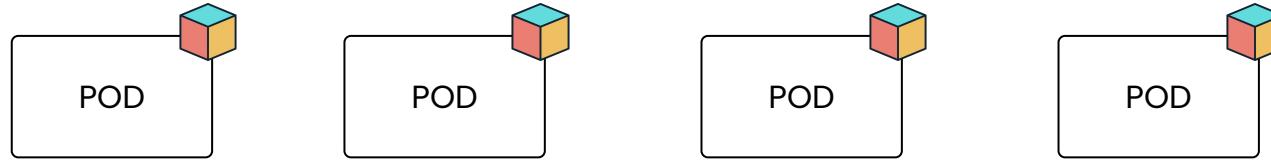


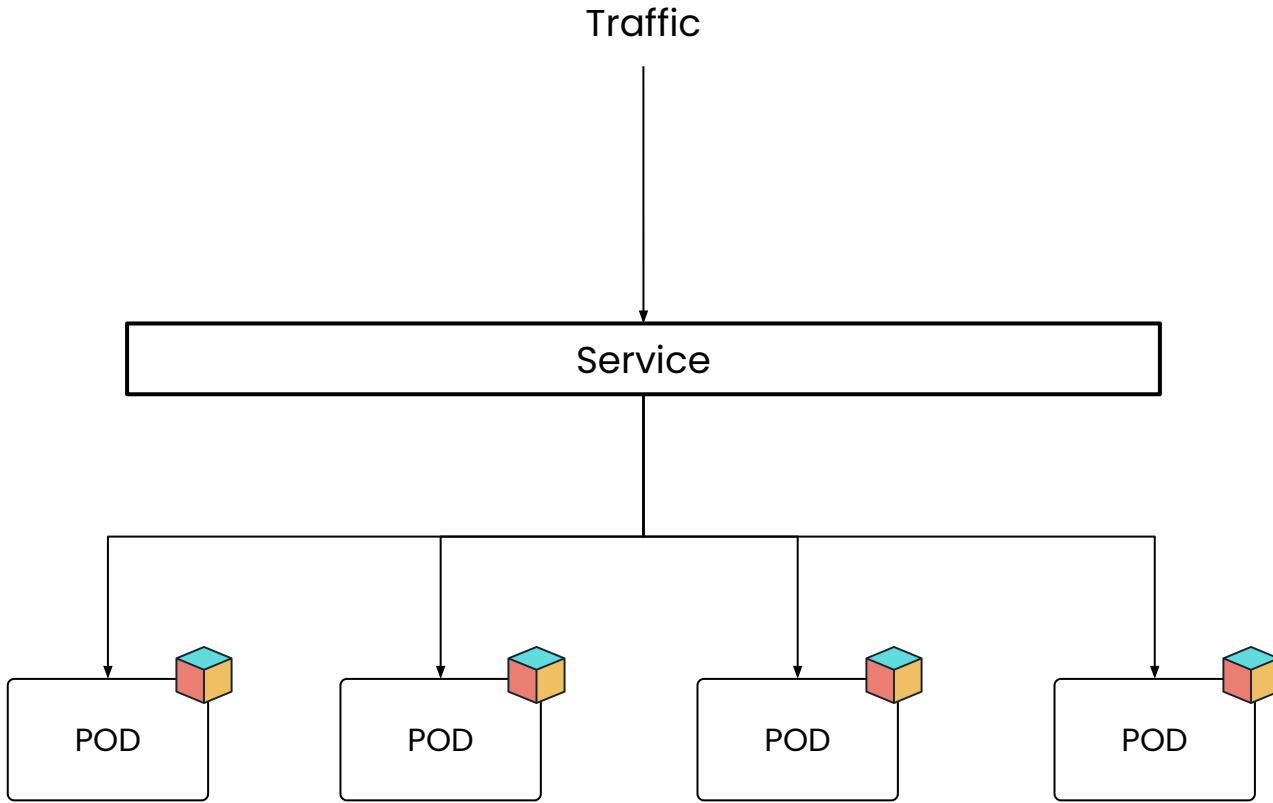
## Traffic



Traffic

Service





## 4. Ingress

Service A

Service B

Service ...

Service n

# Users

Service A

Service B

Service ...

Service n

Users

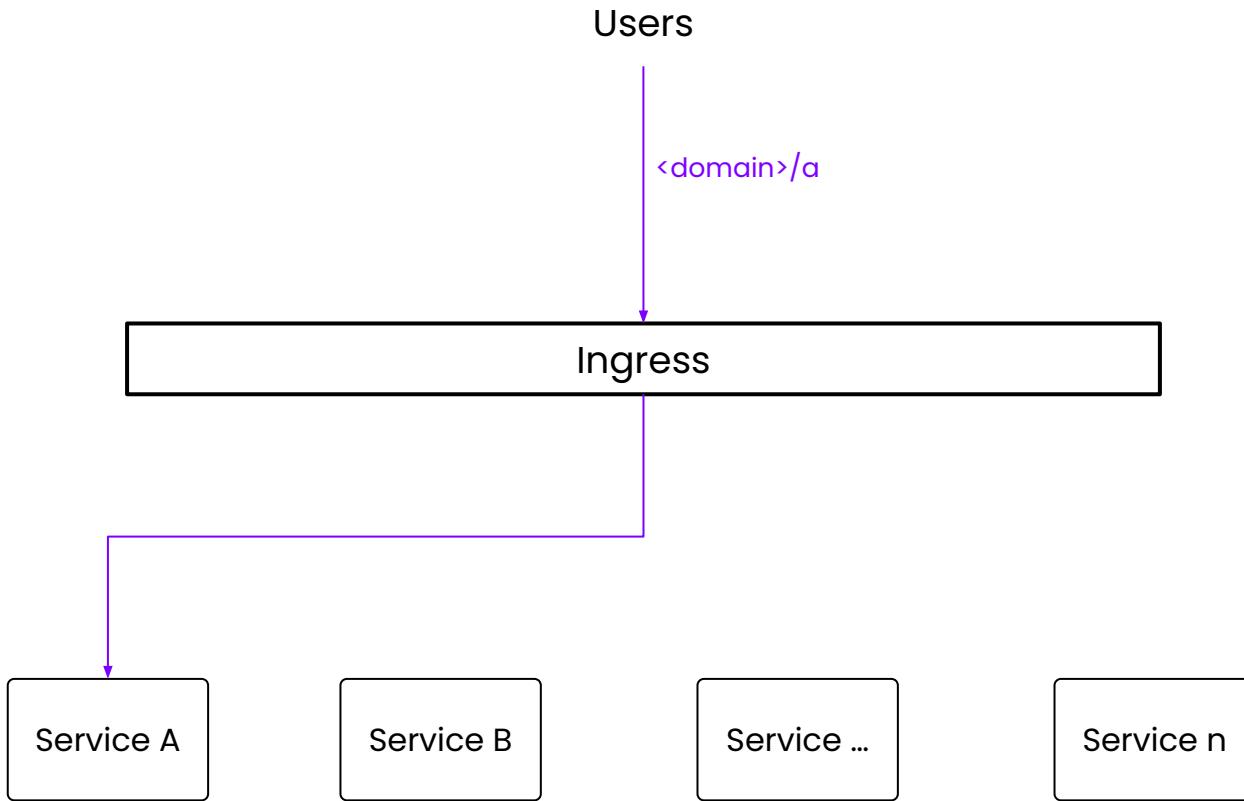
Ingress

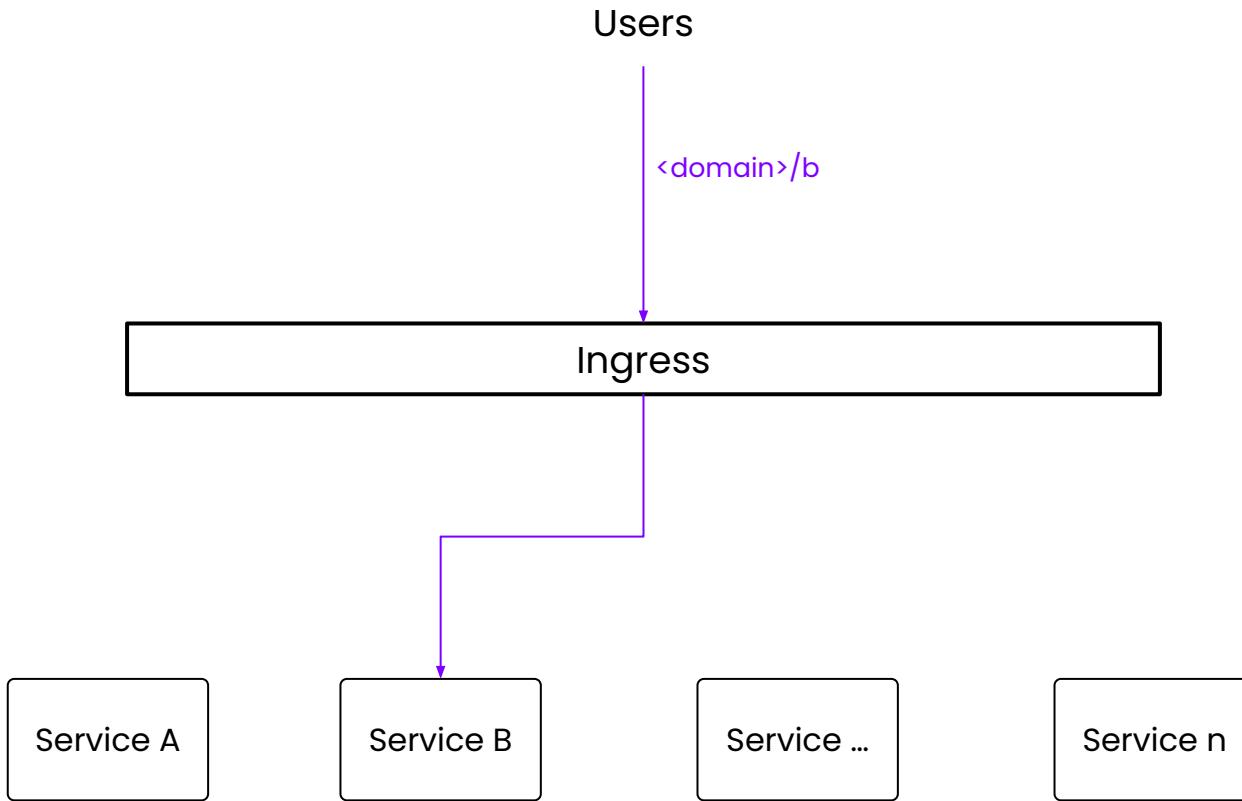
Service A

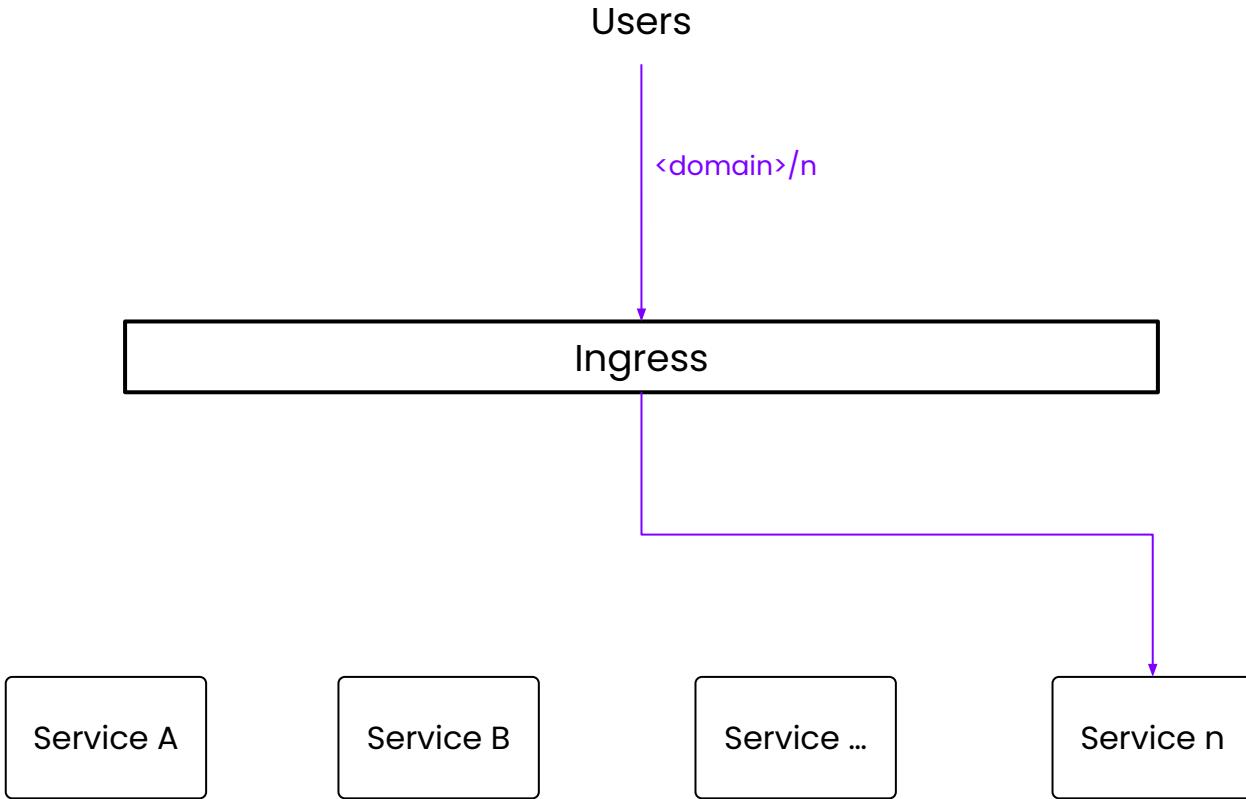
Service B

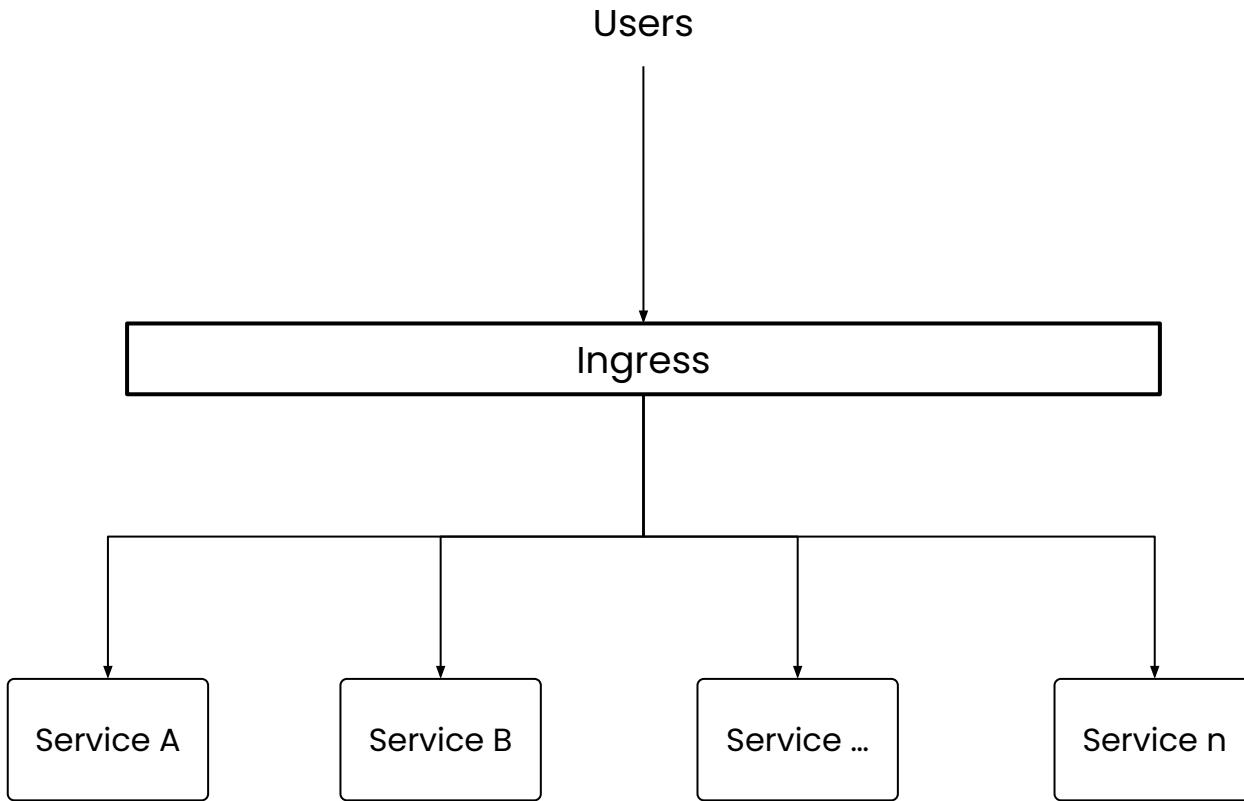
Service ...

Service n



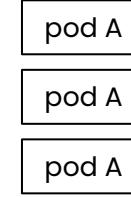






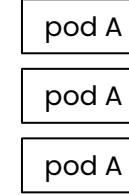
# Kubernetes Overview

# Kubernetes Overview

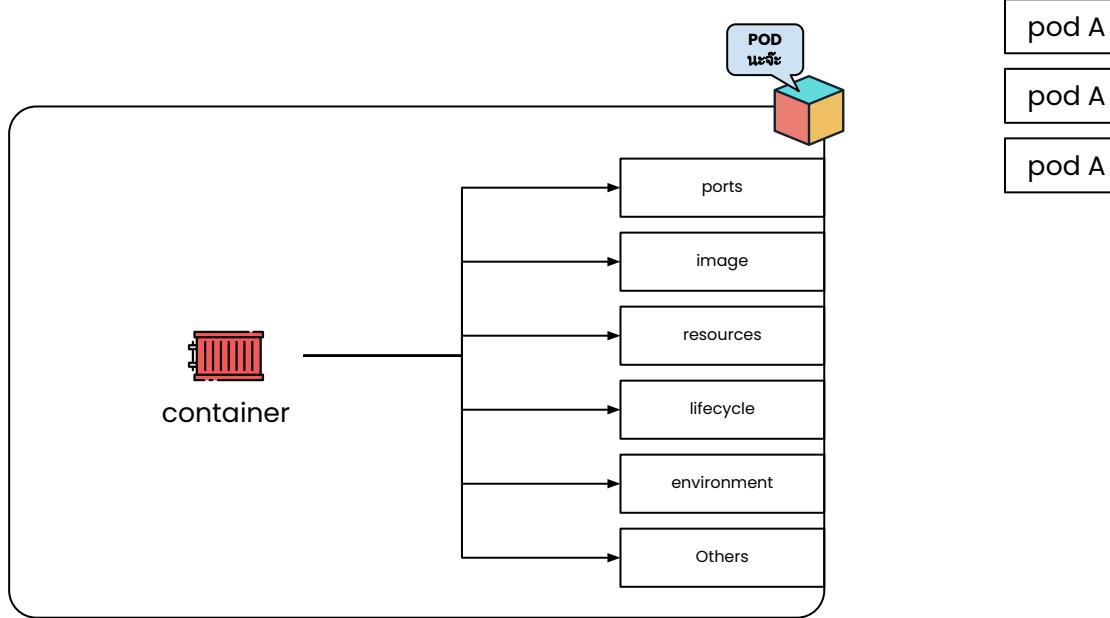


# Kubernetes Overview

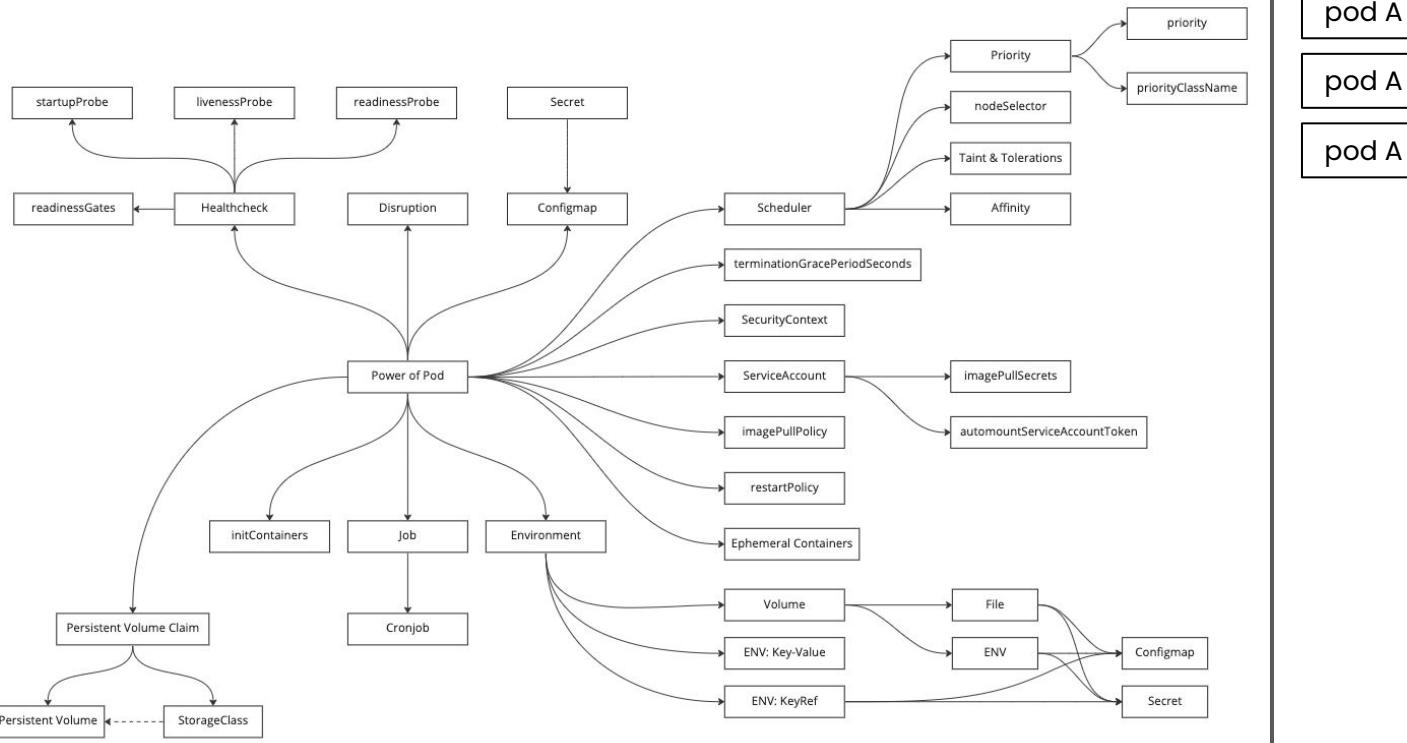
the **smallest** deployable  
units of Kubernetes



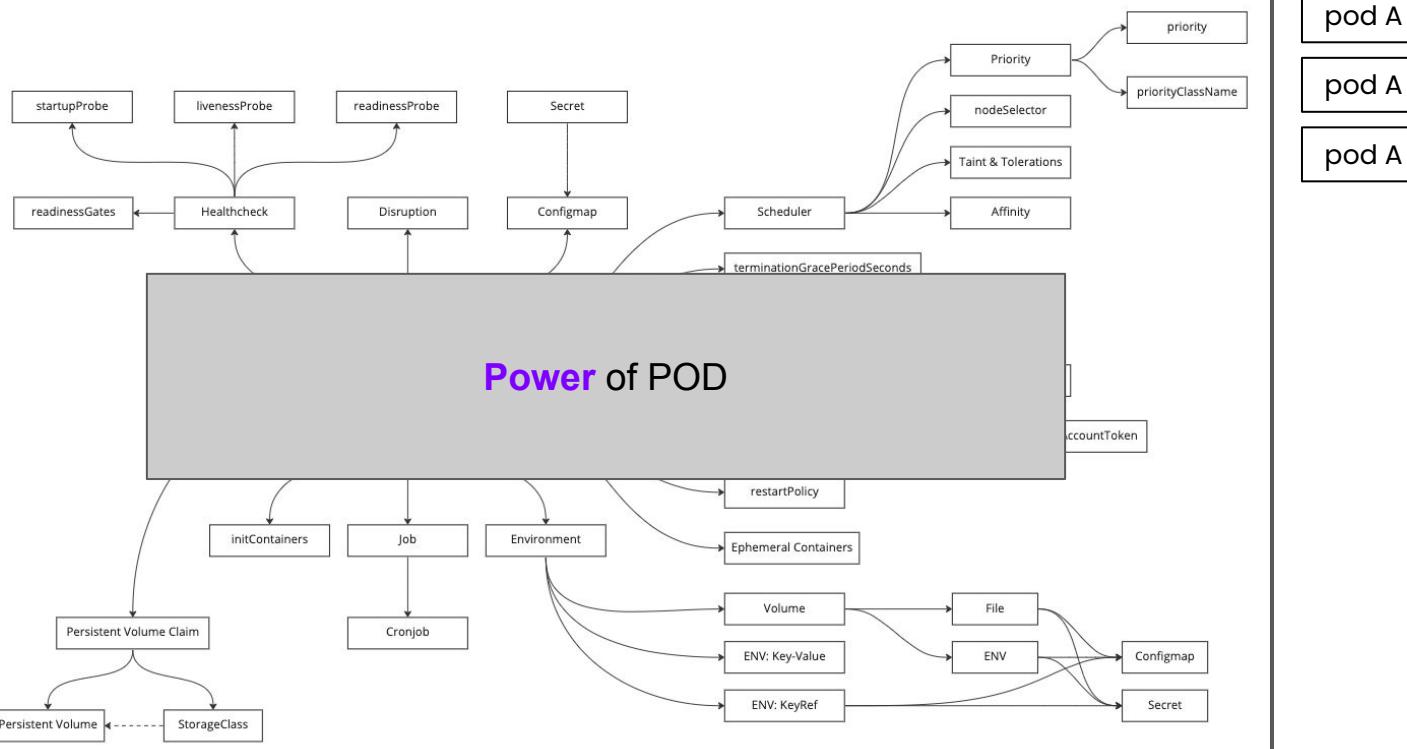
# Kubernetes Overview



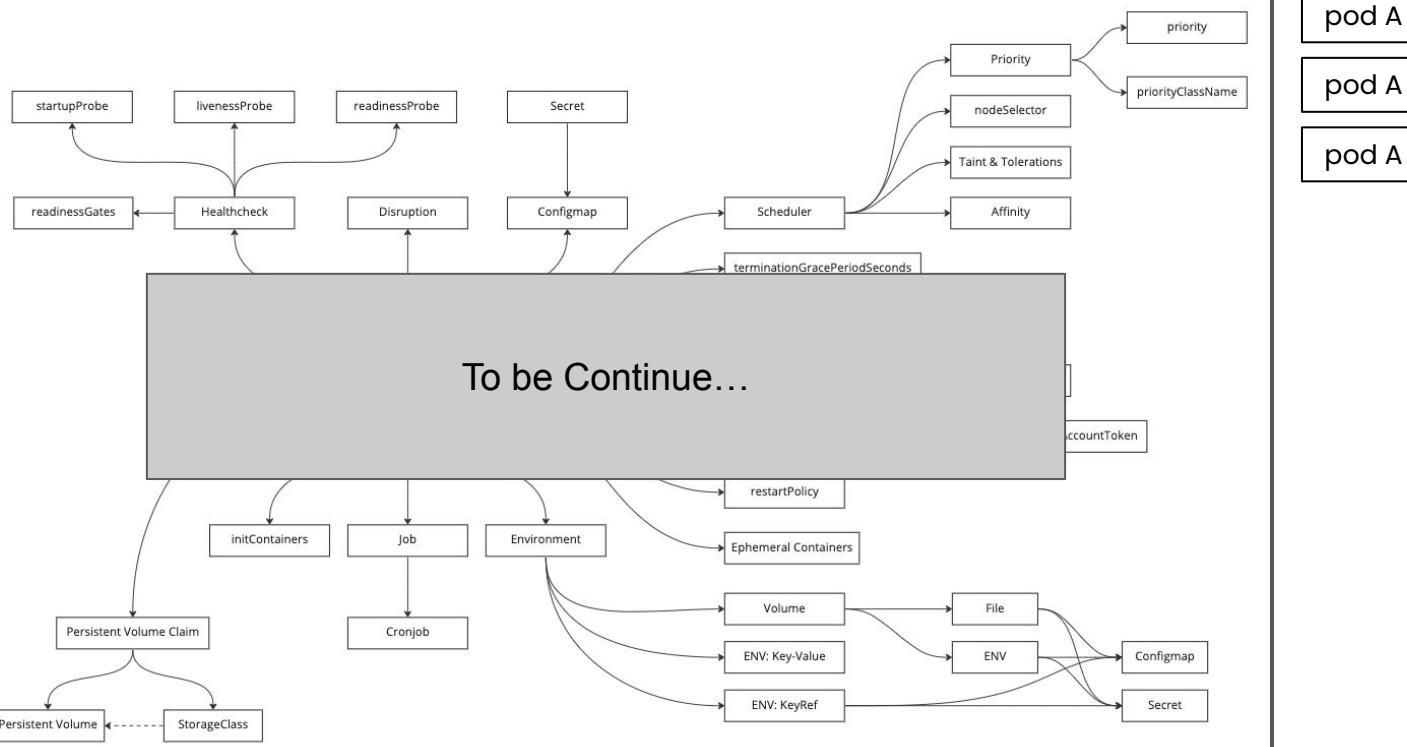
# Kubernetes Overview



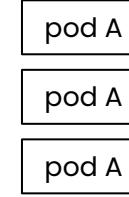
# Kubernetes Overview



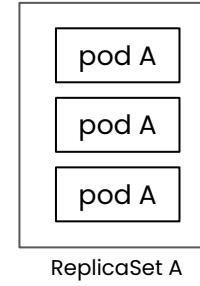
# Kubernetes Overview



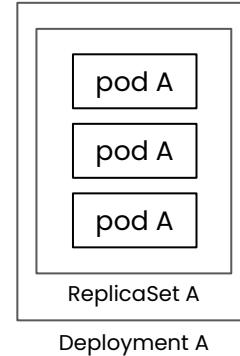
# Kubernetes Overview



# Kubernetes Overview



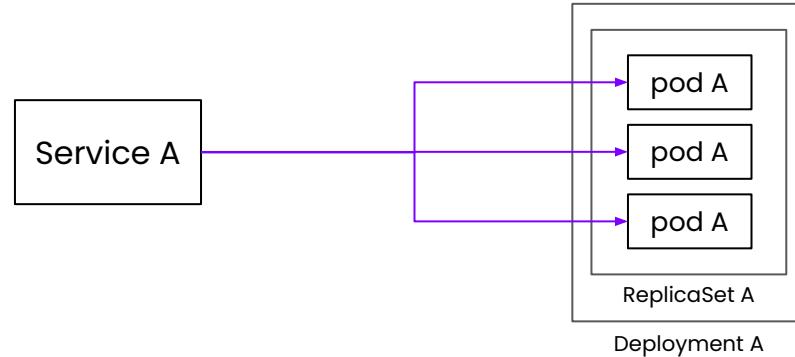
# Kubernetes Overview



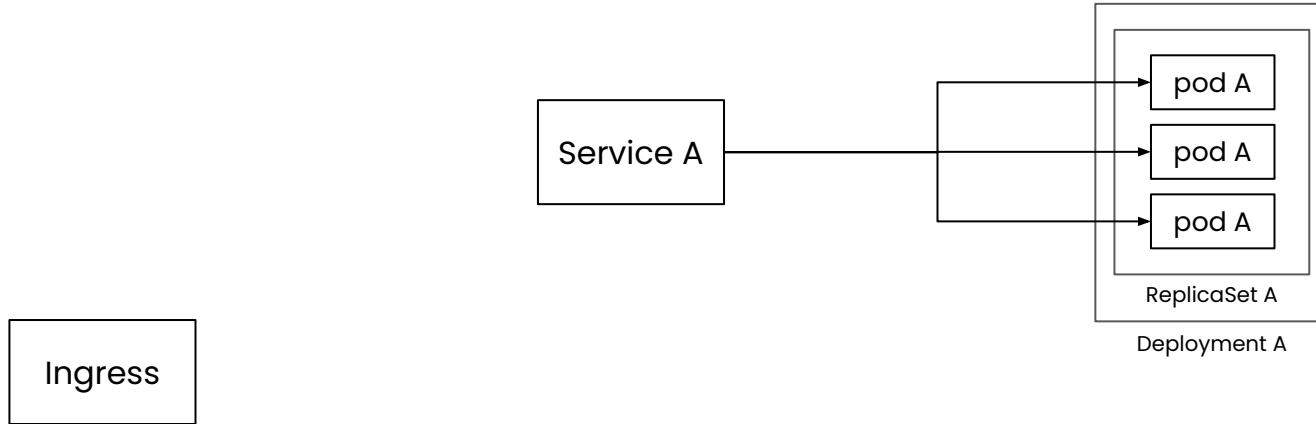
# Kubernetes Overview



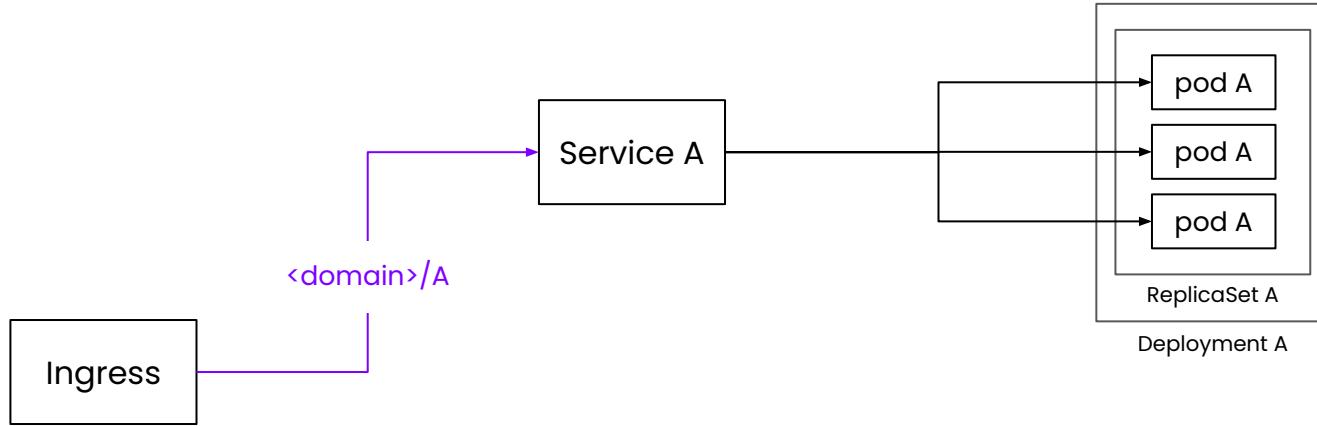
# Kubernetes Overview



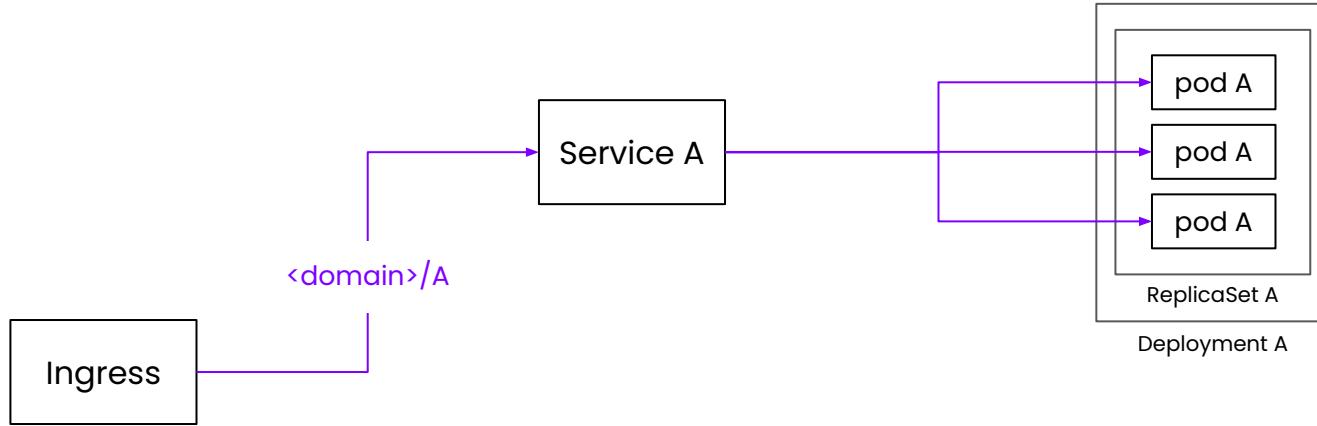
# Kubernetes Overview



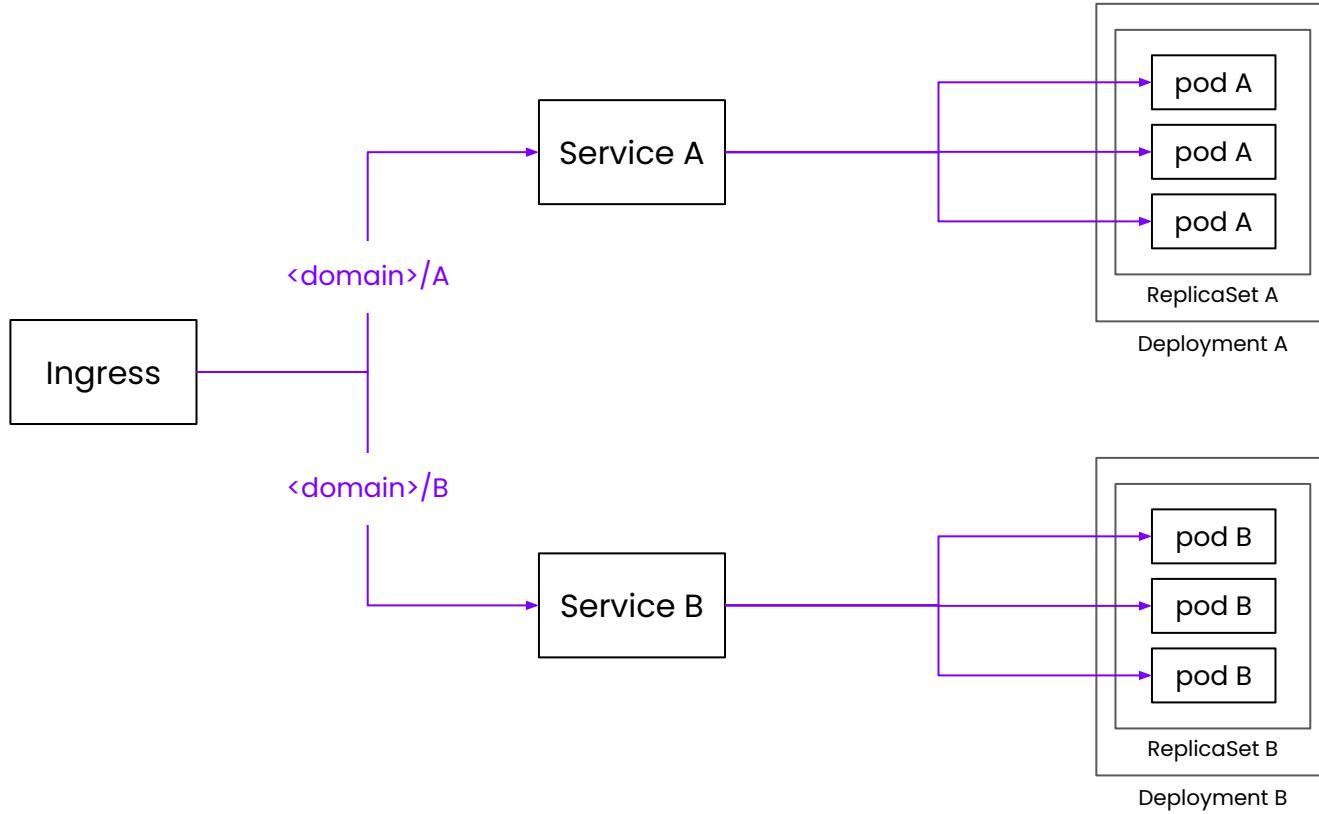
# Kubernetes Overview



# Kubernetes Overview



# Kubernetes Overview



# Kubernetes Objects

- Working with Kubernetes -

# Kubernetes Objects

## Kubernetes Objects (Cont.)

One of the core concepts of Kubernetes is providing a lot of mostly **abstract resources**, also called **objects**

# Kubernetes Objects (Cont.)

One of the core concepts of Kubernetes is providing a lot of mostly abstract resources, also called **objects**

- that you can use to describe how your workload should be handled.

# Kubernetes Objects (Cont.)

One of the core concepts of Kubernetes is providing a lot of mostly abstract resources, also called **objects**

- that you can use to describe how your workload should be handled.
- Some of them are used to **handle problems of container orchestration**, like
  - scheduling, self-healing, and others are there to solve some inherent problems of containers.

# Kubernetes Objects (Cont.)

One of the core concepts of Kubernetes is providing a lot of mostly abstract resources, also called **objects**

- that you can use to describe how your workload should be handled.
- Some of them are used to handle problems of container orchestration, like
  - scheduling, self-healing, and others are there to solve some inherent problems of containers.
- Kubernetes objects can be distinguished between **workload-oriented** objects that are used for handling container workloads and **infrastructure-oriented objects**, that for example handle configuration, networking and security.

# Kubernetes Objects (Cont.)

One of the core concepts of Kubernetes is providing a lot of mostly abstract resources, also called **objects**

- that you can use to describe how your workload should be handled.
- Some of them are used to handle problems of container orchestration, like
  - scheduling, self-healing, and others are there to solve some inherent problems of containers.
- Kubernetes objects can be distinguished between **workload-oriented** objects that are used for handling container workloads and **infrastructure-oriented objects**, that for example handle configuration, networking and security.
- Some of these objects can be **put into a namespace**, while others are available **across the whole cluster**.

## Kubernetes Objects (Cont.)

# Kubernetes Objects (Cont.)

As a user, we can describe these objects in the popular **data-serialization language YAML** and send them to the **api-server**, where they **get validated before they are created**.

```
apiVersion: v1
kind: Pod
metadata:
  name: jumpbox-app
spec:
  containers:
  - name: jumpbox
    image: busybox
    command: ["sh", "-c"]
    args: ["echo Hello Jumpbox!! && sleep 3600"]
```

**YAML file**

# Kubernetes Objects (Cont.)

As a user, we can describe these objects in the popular data-serialization language **YAML** and send them to the api-server, where they get validated before they are created.

**The fields highlighted are required fields. They include:**

## **apiVersion**

- Each object can be versioned. That means the data structure of the **object can change** between different versions.

```
apiVersion: v1
kind: Pod
metadata:
  name: jumpbox-app
spec:
  containers:
    - name: jumpbox
      image: busybox
      command: ["sh", "-c"]
      args: ["echo Hello Jumpbox!! && sleep 3600"]
```

YAML file

# Kubernetes Objects (Cont.)

As a user, we can describe these objects in the popular data-serialization language **YAML** and send them to the api-server, where they get validated before they are created.

**The fields highlighted are required fields. They include:**

```
apiVersion: v1
kind: Pod
metadata:
  name: jumpbox-app
spec:
  containers:
    - name: jumpbox
      image: busybox
      command: ["sh", "-c"]
      args: ["echo Hello Jumpbox!! && sleep 3600"]
```

YAML file

## apiVersion

- Each object can be versioned. That means the data structure of the **object can change** between different versions.

## kind

- The **kind of object** that should be created.

# Kubernetes Objects (Cont.)

As a user, we can describe these objects in the popular data-serialization language **YAML** and send them to the api-server, where they get validated before they are created.

**The fields highlighted are required fields. They include:**

```
apiVersion: v1
kind: Pod
metadata:
  name: jumpbox-app
spec:
  containers:
    - name: jumpbox
      image: busybox
      command: ["sh", "-c"]
      args: ["echo Hello Jumpbox!! && sleep 3600"]
```

YAML file

## apiVersion

- Each object can be versioned. That means the data structure of the **object can change** between different versions.

## kind

- The **kind of object** that should be created.

## metadata

- Data that can be **used to identify it**. A **name** is required for each object and **must be unique**. You can use namespaces if you need multiple objects with the same name.

# Kubernetes Objects (Cont.)

As a user, we can describe these objects in the popular data-serialization language **YAML** and send them to the api-server, where they get validated before they are created.

**The fields highlighted are required fields. They include:**

```
apiVersion: v1
kind: Pod
metadata:
  name: jumpbox-app
spec:
  containers:
    - name: jumpbox
      image: busybox
      command: ["sh", "-c"]
      args: ["echo Hello Jumpbox!! && sleep 3600"]
```

YAML file

## apiVersion

- Each object can be versioned. That means the data structure of the **object can change** between different versions.

## kind

- The **kind of object** that should be created.

## metadata

- Data that can be **used to identify it**. A name is required for each object and **must be unique**. You can use namespaces if you need multiple objects with the same name.

## spec

- The **specification of the object**. Here you can describe your **desired state**. Be cautious, since the structure for the **object can change with its version!**

# Kubernetes Objects (Cont.)

As a user, we can describe these objects in the popular **data-serialization language YAML** and send them to the **api-server**, where they **get validated before they are created**.

```
apiVersion: v1
kind: Pod
metadata:
  name: jumpbox-app
spec:
  containers:
    - name: jumpbox
      image: busybox
      command: ["sh", "-c"]
      args: ["echo Hello Jumpbox!! && sleep 3600"]
```

YAML file

**The fields highlighted are required fields. They include:**

#### apiVersion

- Each object can be versioned. That means the data structure of the **object can change** between different versions.

#### kind

- The **kind of object** that should be created.

#### metadata

- Data that can be **used to identify it**. A **name** is required for **each object and must be unique**. You can use namespaces if you need multiple objects with the same name.

#### spec

- The **specification of the object**. Here you can describe your **desired state**. Be cautious, since the structure for the **object can change with its version!**

# Kubernetes API

- Alpha:
  - The version names contain `alpha` (for example, `v1alpha1` ).
  - Built-in alpha API versions are disabled by default and must be explicitly enabled in the `kube-apiserver` configuration to be used.
  - The software may contain bugs. Enabling a feature may expose bugs.
  - Support for an alpha API may be dropped at any time without notice.
  - The API may change in incompatible ways in a later software release without notice.
  - The software is recommended for use only in short-lived testing clusters, due to increased risk of bugs and lack of long-term support.

Reference Pictures:

- [API Overview](#)

# Kubernetes API

- Beta:

- The version names contain `beta` (for example, `v2beta3` ).
- Built-in beta API versions are disabled by default and must be explicitly enabled in the `kube-apiserver` configuration to be used (**except** for beta versions of APIs introduced prior to Kubernetes 1.22, which were enabled by default).
- Built-in beta API versions have a maximum lifetime of 9 months or 3 minor releases (whichever is longer) from introduction to deprecation, and 9 months or 3 minor releases (whichever is longer) from deprecation to removal.
- The software is well tested. Enabling a feature is considered safe.
- The support for a feature will not be dropped, though the details may change.
- The schema and/or semantics of objects may change in incompatible ways in a subsequent beta or stable API version. When this happens, migration instructions are provided. Adapting to a subsequent beta or stable API version may require editing or re-creating API objects, and may not be straightforward. The migration may require downtime for applications that rely on the feature.
- The software is not recommended for production uses. Subsequent releases may introduce incompatible changes. Use of beta API versions is required to transition to subsequent beta or stable API versions once the beta API version is deprecated and no longer served.

**Note:**

Please try beta features and provide feedback. After the features exit beta, it may not be practical to make more changes.

Reference Pictures:

- [API Overview](#)

# Kubernetes API

- Stable:
  - The version name is `vX` where `X` is an integer.
  - Stable API versions remain available for all future releases within a Kubernetes major version, and there are no current plans for a major version revision of Kubernetes that removes stable APIs.

Reference Pictures:

- [API Overview](#)

# Kubernetes API (Cont.)

## API Versioning:

- e.g., **v1alpha1**, API may be buggy and unstable, disabled in production clusters
- e.g., **v1beta1**, API may have bugs but generally is stable, may be incompatible between releases, enabled in production clusters
- e.g., **v1**, API is stable and become generally available, supports backward compatibility, suitable for production

## Reference Pictures:

- [Introduction to Kubernetes API – the way to understand the concept of Kubernetes Operators.](#)

# Kubernetes API (Cont.)

We can split objects into basic one:

- Pod
- Service
- Volume
- Namespace

and more high-level objects built on the logic of the basic ones:

- ReplicaSet
- Deployment
- StatefulSet
- Job

Reference Pictures:

- [Introduction to Kubernetes API – the way to understand the concept of Kubernetes Operators.](#)

# Workload Objects

- Working with Kubernetes -

Working just with Pods would not be flexible enough in a container orchestration platform.

For example,

For example,

if a Pod is lost because a node failed, it is **gone forever**.

To **make sure** that a **defined number** of Pod copies runs all the time,

To **make sure** that a defined number of Pod copies runs all the time, we can **use controller** objects that **manage the pod for us.**

# Workload Objects

Jumpbox®

เจ้าของลิขสิทธิ์ อนุญาติให้ใช้สิ่งดังนี้เพื่อการเรียนรู้ส่วนบุคคลเท่านั้น และขอสงวนลิขสิทธิ์ ห้ามมิให้เผยแพร่ในที่สาธารณะ ผู้ลงทะเบียน จะถูกดำเนินคดีตามกฎหมาย

# Workload Objects (Cont.)

## Kubernetes Objects:

- ReplicationController
- ReplicaSet
- Deployment
- StatefulSet
- DaemonSet
- Job
- CronJob

Reference:

- [Workload](#)

# Workload Objects (Cont.)

## Kubernetes Objects:

- **ReplicationController**
  - A ReplicationController ensures that a specified number of pod replicas are running at any one time. In other words, a ReplicationController makes sure that a pod or a homogeneous set of pods is always up and available.
  - A Deployment that configures a ReplicaSet is now the recommended way to set up replication.

Reference:

- [Workload](#)

# Workload Objects (Cont.)

## Kubernetes Objects (Cont.):

- **ReplicaSet**
  - A controller object that **ensures a desired number** of pods is running at any given time.
  - ReplicaSets can be **used to scale out applications and improve their availability**.
  - They do this by starting multiple copies of a pod definition.

# Workload Objects (Cont.)

## Kubernetes Objects (Cont.):

- **ReplicaSet**
  - A controller object that **ensures a desired number** of pods is running at any given time.
  - ReplicaSets can be used to scale out applications and improve their availability.
  - They do this by starting multiple copies of a pod definition.
- **Deployment**
  - The most **feature-rich** object in Kubernetes. A Deployment can be used to **describe the complete application lifecycle**, they do this by **managing multiple ReplicaSets** that get updated when the application is changed by providing a new container image, for example. Deployments are **perfect to run stateless** applications in Kubernetes.

# Workload Objects (Cont.)

## Kubernetes Objects (Cont.):

- **ReplicaSet**
  - A controller object that **ensures a desired number** of pods is running at any given time.
  - ReplicaSets can be used to scale out applications and improve their availability.
  - They do this by starting multiple copies of a pod definition.
- **Deployment**
  - The most feature-rich object in Kubernetes. A Deployment can be used to describe the complete application lifecycle, they do this by managing multiple ReplicaSets that get updated when the application is changed by providing a new container image, for example. Deployments are **perfect to run stateless** applications in Kubernetes.
- **StatefulSet**
  - Considered a bad practice for a long time, StatefulSets can be **used to run stateful** applications like databases on Kubernetes. Stateful applications have special requirements that don't fit the ephemeral nature of pods and containers. In contrast to Deployments, StatefulSets try to **retain IP addresses of pods** and give them a **stable name, persistent storage** and more graceful handling of scaling and updates.

# Workload Objects (Cont.)

## Kubernetes Objects (Cont.):

- **DaemonSet**
  - Ensures that a copy of a Pod **runs on all (or some) nodes** of your cluster.
  - DaemonSets are perfect to run **infrastructure-related workload**, for example monitoring or logging tools.

Reference:

- [Workload](#)

# Workload Objects (Cont.)

## Kubernetes Objects (Cont.):

- **DaemonSet**
  - Ensures that a copy of a Pod **runs on all (or some) nodes** of your cluster.
  - DaemonSets are perfect to run infrastructure-related workload, for example monitoring or logging tools.
- **Job**
  - Creates one or more Pods that **execute a task and terminate afterwards**. Job objects are perfect to run **one-shot scripts** like database migrations or administrative tasks.

Reference:

- [Workload](#)

# Workload Objects (Cont.)

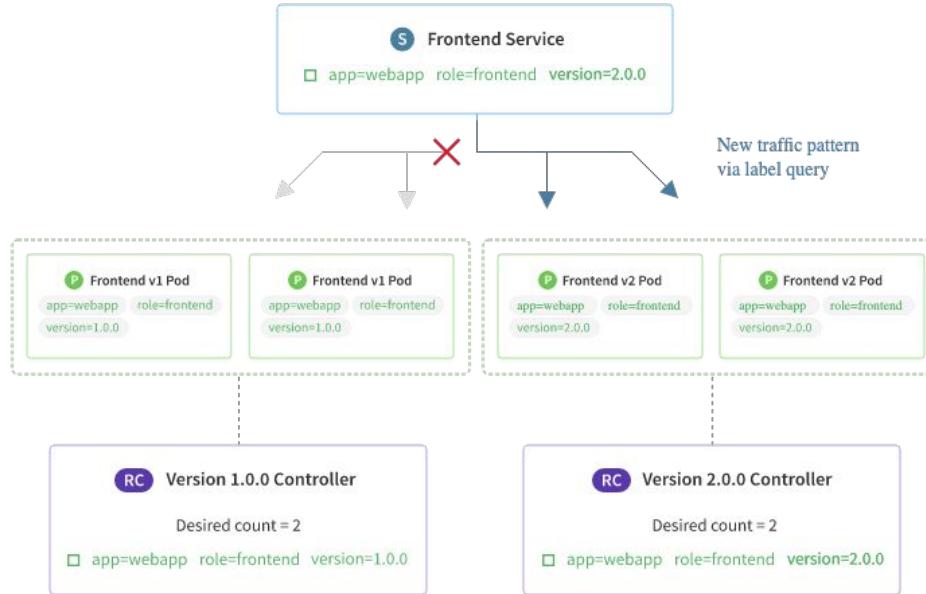
## Kubernetes Objects (Cont.):

- **DaemonSet**
  - Ensures that a copy of a Pod **runs on all (or some) nodes** of your cluster.
  - DaemonSets are perfect to run infrastructure-related workload, for example monitoring or logging tools.
- **Job**
  - Creates one or more Pods that **execute a task and terminate afterwards**. Job objects are perfect to run one-shot scripts like database migrations or administrative tasks.
- **CronJob**
  - CronJobs add a **time-based configuration to jobs**. This allows running Jobs periodically, for example doing a backup job every night at 4am.

Reference:

- [Workload](#)

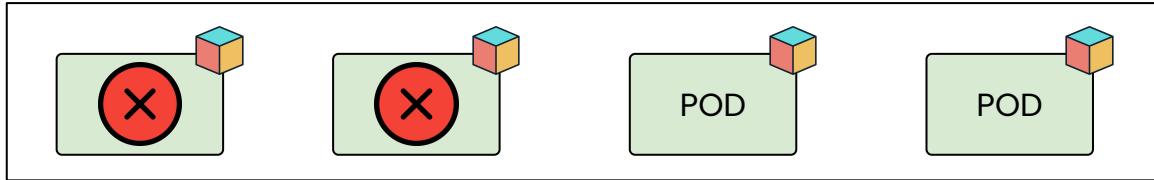
# Replication Controller (Traffic Shift)



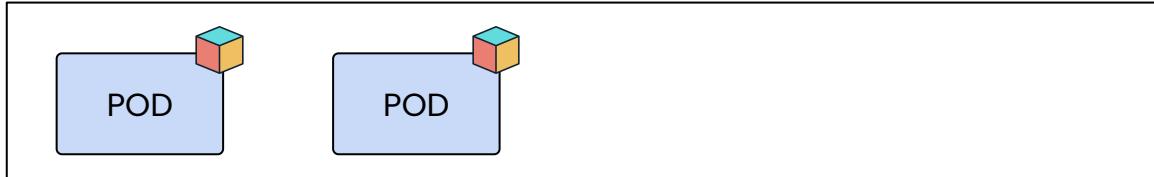
Reference:

- [Docker & Kubernetes](#)

# ReplicaSet



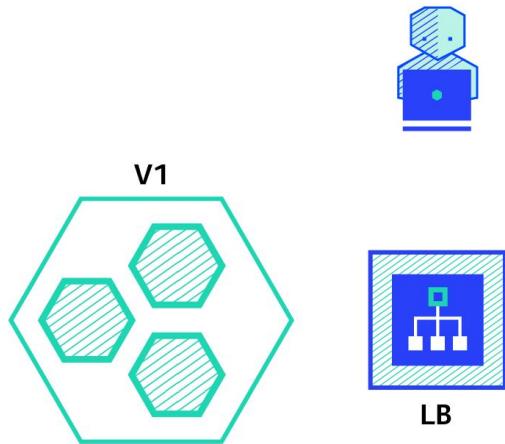
ReplicaSet: v1



ReplicaSet: v2

Deployment

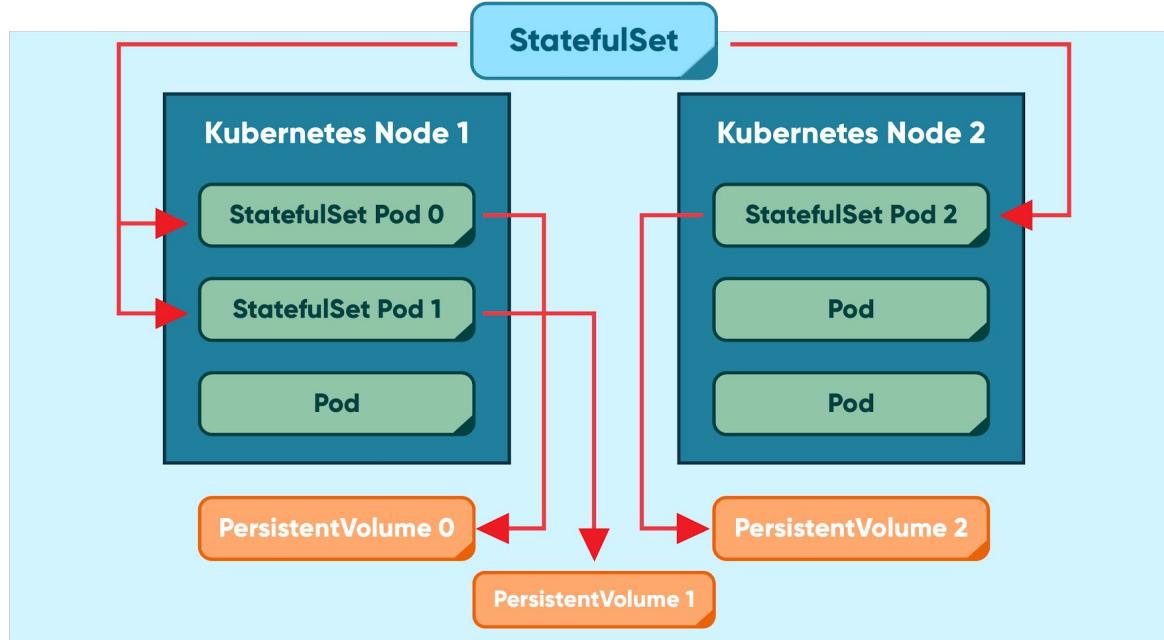
# Deployment



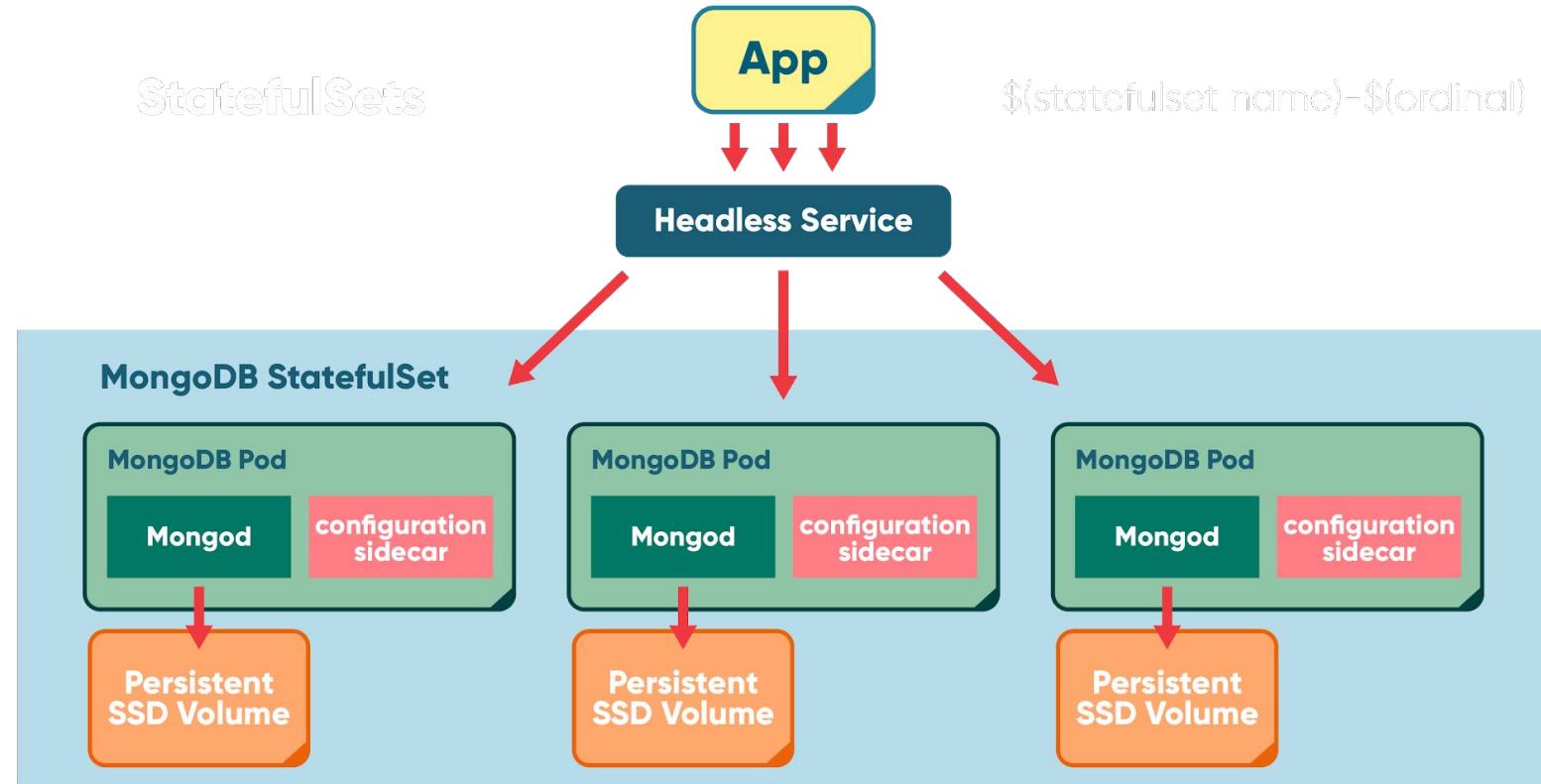
Reference:

- [Deployment Strategies](#)

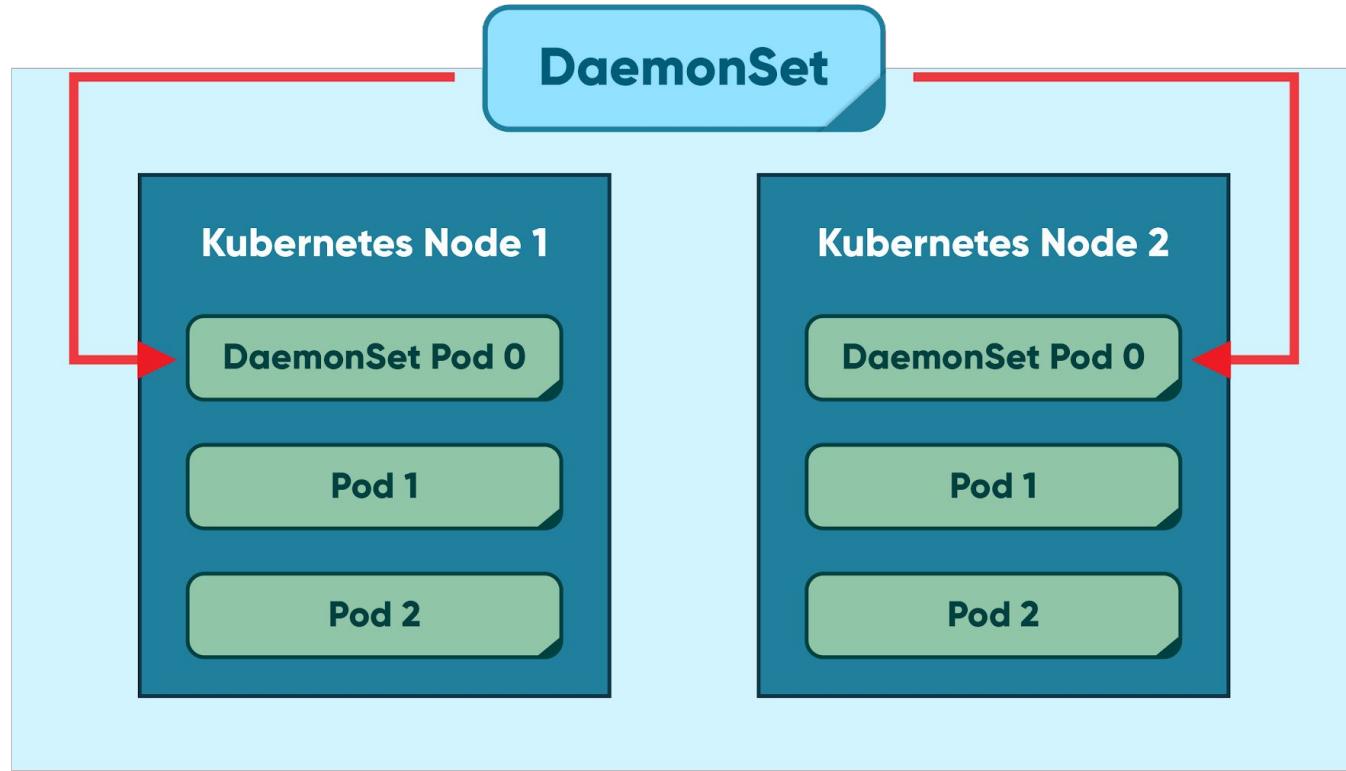
# StatefulSet



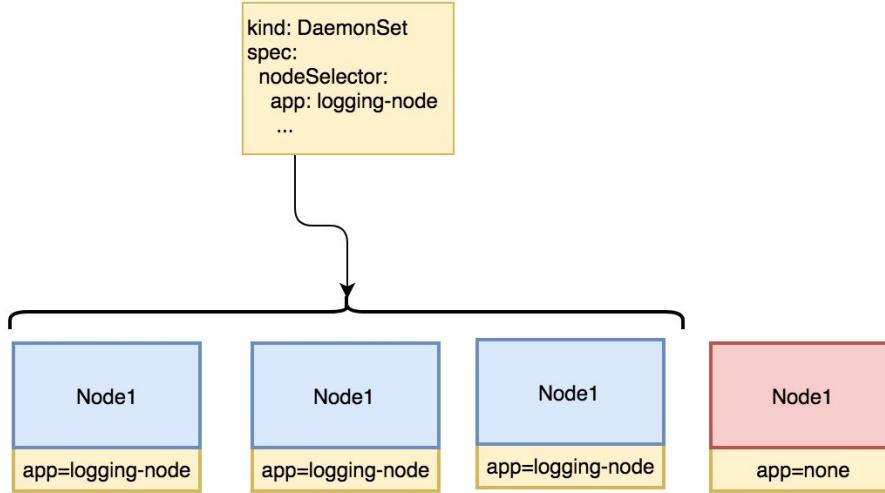
## StatefulSet (Cont.)



# DaemonSet



## DaemonSet (Cont.)



Reference:

- [Kubernetes 101 DaemonSets #5](#)

# Job

## Jobs

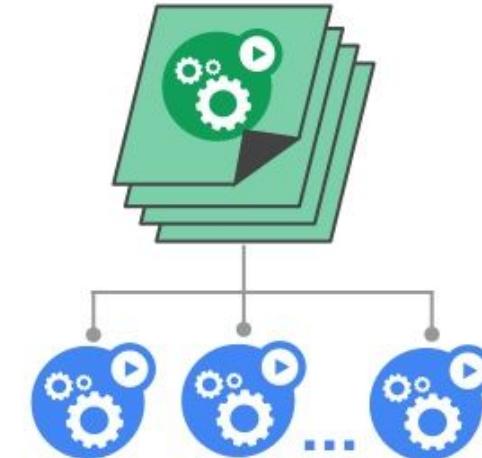
Run-to-completion, as opposed to run-forever

- Express parallelism vs. required completions
- Workflow: restart on failure
- Build/test: don't restart on failure

Aggregates success/failure counts

Built for batch and big-data work

Status: **GA** in Kubernetes v1.2



Google Cloud Platform

Reference:

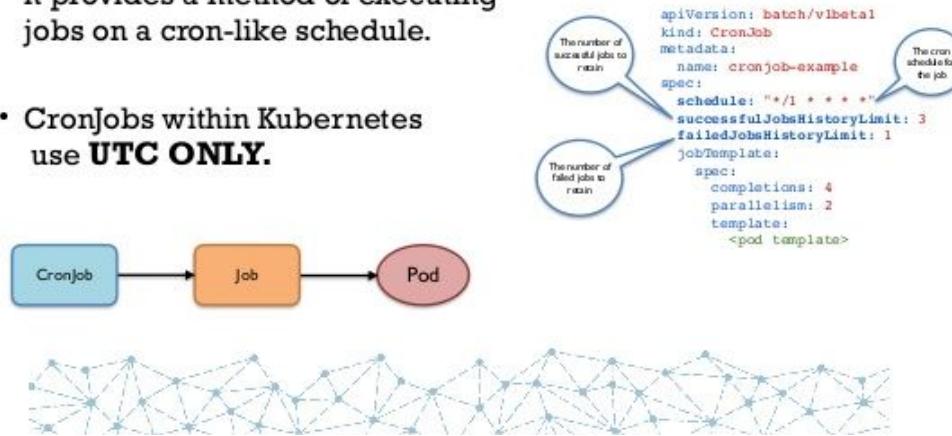
- [What's new in Kubernetes](#)

# CronJob

A Cron Job creates Jobs on a time-based schedule. One CronJob object is like one line of a crontab (cron table) file. It runs a job periodically on a given schedule, written in Cron format.

## CRONJOB

- An extension of the Job Controller, it provides a method of executing jobs on a cron-like schedule.
- CronJobs within Kubernetes use **UTC ONLY**.



Reference:

- [What's new in Kubernetes](#)
- [DevJam 2019 – Introduction to Kubernetes](#)

# Kubernetes Replication Controller, Replica Set and Deployments: Understanding replication options

Link: [Kubernetes Replication Controller, Replica Set and Deployments](#)

# Interacting with Kubernetes

- Working with Kubernetes -

# Interacting with Kubernetes

Jumpbox®

เจ้าของลิขสิทธิ์ อนุญาติให้ใช้สิ่งดังนี้เพื่อการเรียนรู้ส่วนบุคคลเท่านั้น และขอสงวนลิขสิทธิ์ ห้ามมิให้เผยแพร่ในที่สาธารณะ ผู้อื่นเดินทาง จะถูกดำเนินคดีตามกฎหมาย

# Interacting with Kubernetes (Cont.)

Let's take a look at the **basic kubectl** commands.

You can use the **--help** flag to view them:

```
$ kubectl --help
```

```
kubectl controls the Kubernetes cluster manager.

Find more information at: https://kubernetes.io/docs/reference/kubectl/overview/

Basic Commands (Beginner):
  create Create a resource from a file or from stdin
  expose Take a replication controller, service, deployment or pod and expose it as a new Kubernetes service
  run Run a particular image on the cluster
  set Set specific features on objects

Basic Commands (Intermediate):
  explain Get documentation for a resource
  get Display one or many resources
  edit Edit a resource on the server
  delete Delete resources by file names, stdin, resources and names, or by resources and label selector
```

Reference:

- [kubectl](#)

# Interacting with Kubernetes (Cont.)

To access the API, users can use the official command line interface client called [kubectl](#). Lets look at some basic commands for everyday Kubernetes usage.

## Reference:

- [kubectl](#)

# Interacting with Kubernetes (Cont.)

To access the API, users can use the official command line interface client called kubectl. Lets look at some basic commands for everyday Kubernetes usage.

You can **list the available objects in your cluster with the following command:**

```
$ kubectl api-resources
```

NAME	SHORTNAMES	APIVERSION	NAMESPACED	KIND
...				
configmaps	cm	v1	true	ConfigMap
...				
namespaces	ns	v1	false	Namespace
nodes	no	v1	false	Node
persistentvolumeclaims	pvc	v1	true	PersistentVolumeClaim
...				
pods	po	v1	true	Pod
...				
services	svc	v1	true	Service

Reference:

- [kubectl](#)

## Interacting with Kubernetes (Cont.)

If you want to know more about an object, `kubectl` has a built-in `explain` function!. It can be used to learn more about how to write yaml file.

# Interacting with Kubernetes (Cont.)

If you want to know more about an object, `kubectl` has a built-in `explain` function!. It can be used to learn more about how to write yaml file.

## Let's learn more about pods:

```
$ kubectl explain pod
```

```
KIND:      Pod
VERSION:   v1

DESCRIPTION:
    Pod is a collection of containers that can run on a host. This resource is
    created by clients and scheduled onto hosts.

FIELDS:
    apiVersion <string>
        APIVersion defines the versioned schema of this representation of an
        ...
    ...
    kind <string>
    ...
    metadata <Object>
    ...
    spec <Object>
```

# Interacting with Kubernetes (Cont.)

If you want to know more about an object, `kubectl` has a built-in `explain` function!. It can be used to learn more about how to write yaml file.

## Let's learn more about pods (Cont.):

```
$ kubectl explain pod.spec
```

```
KIND:      Pod
VERSION:   v1

RESOURCE: spec <Object>

DESCRIPTION:
  Specification of the desired behavior of the pod. More info:
  https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#spec-and-status

  PodSpec is a description of a pod.

FIELDS:
  activeDeadlineSeconds <integer>
    Optional duration in seconds the pod may be active on the node relative to
    StartTime before the system will actively try to mark it failed and kill
    associated containers. Value must be a positive integer.
  ...
  ...
```

# The **Tools** for interaction with Kubernetes:

## Reference:

- [Helm](#)
- [ArtifactHub](#)

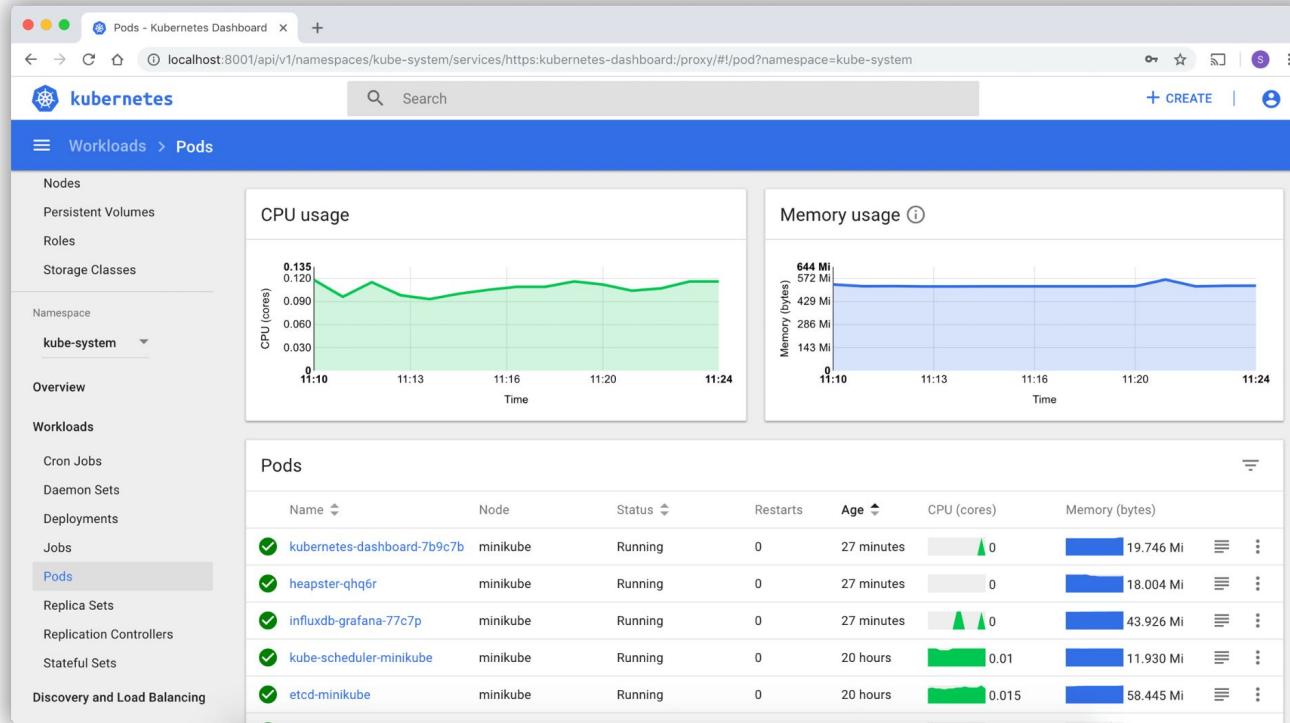
# The **Tools** for interaction with Kubernetes:

- [kubernetes/dashboard](#)
- [derailed/k9s](#)
- [Lens](#)
- [Rancher](#)
- [VMware Tanzu Octant](#)

## Reference:

- [Helm](#)
- [ArtifactHub](#)

# Web UI (Dashboard)



# Problem

- ~~Security with official Web UI~~
- Convenient

# The **Tools** for interaction with Kubernetes:

- [kubernetes/dashboard](#)
- [derailed/k9s](#)
- [Lens](#)
- [Rancher](#)
- [VMware Tanzu Octant](#)

Despite the numerous CLI tools and GUIs, there are also advanced tools that **allow the creation of templates and the packaging of Kubernetes objects**. Probably the most frequently used tool in connection with Kubernetes today is [\*\*Helm\*\*](#).

## Reference:

- [Helm](#)
- [ArtifactHub](#)

# The **Tools** for interaction with Kubernetes:

- [kubernetes/dashboard](#)
- [derailed/k9s](#)
- [Lens](#)
- [Rancher](#)
- [VMware Tanzu Octant](#)

Despite the numerous CLI tools and GUIs, there are also advanced tools that allow the creation of templates and the packaging of Kubernetes objects. Probably the most frequently used tool in connection with Kubernetes today is **Helm**.

**Helm** is a **package manager** for Kubernetes, which allows easier updates and interaction with objects. Helm packages Kubernetes objects in **so-called Charts**, which can be **shared with others via a registry**. To get started with Kubernetes, you can search the [\*\*ArtifactHub\*\*](#) to find your favorite software packages, ready to deploy.

## Reference:

- [Helm](#)
- [ArtifactHub](#)



The  
package manager  
for Kubernetes

Helm is the best way to find, share,  
and use software built for Kubernetes.



Reference:

- [Helm](#)
- [ArtifactHub](#)

# Kubernetes Architecture (Basic)

- Kubernetes Fundamental -

- Process in Kubernetes
- Kubernetes Component (Basic)



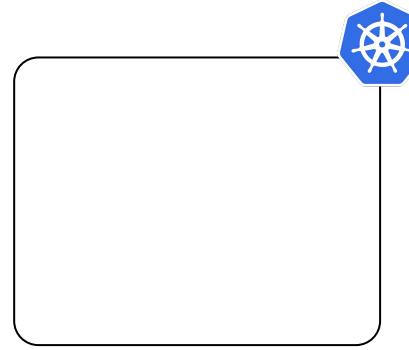
Me

เจ้าของลิขสิทธิ์ อนุญาติให้ใช้สิ่งที่มีลักษณะเด่นๆ ของการเรียนรู้ส่วนบุคคลเท่านั้น และขอสงวนลิขสิทธิ์ ห้ามมิให้เผยแพร่ในที่สาธารณะ ผู้ลงทะเบียน จะถูกดำเนินคดีตามกฎหมาย

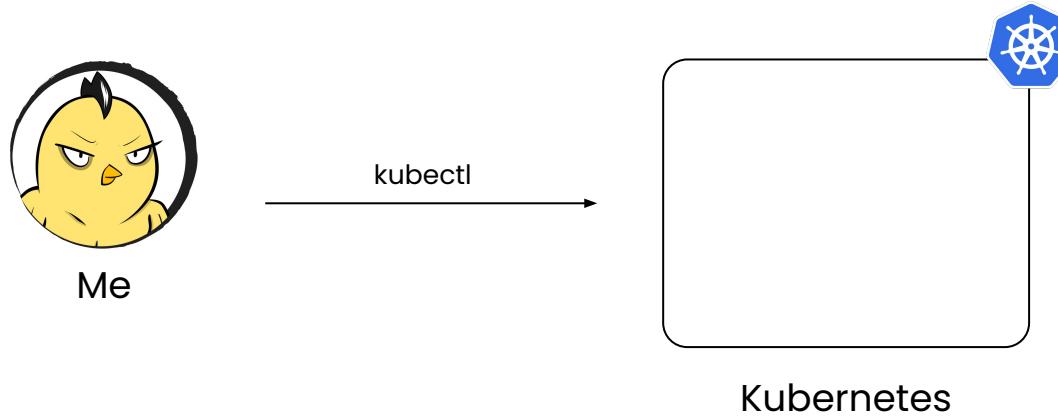
**Jumpbox®**



Me



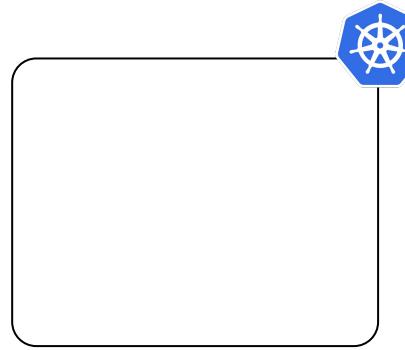
Kubernetes





Me

CLI  
(Command Line Interface)  
kubectl

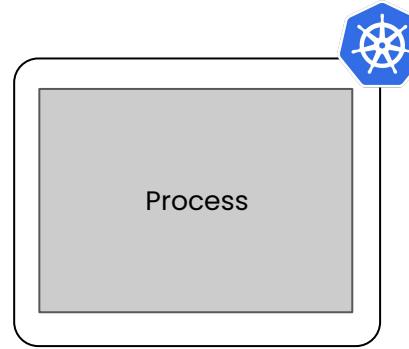


Kubernetes



Me

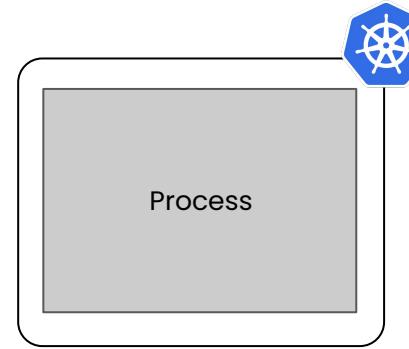
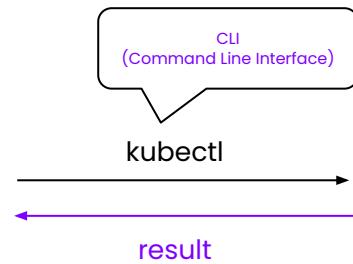
CLI  
(Command Line Interface)  
kubectl



Kubernetes



Me



Kubernetes

# Process in Kubernetes

- Kubernetes Architecture (Basic) -

# Process in Kubernetes (Cont.)

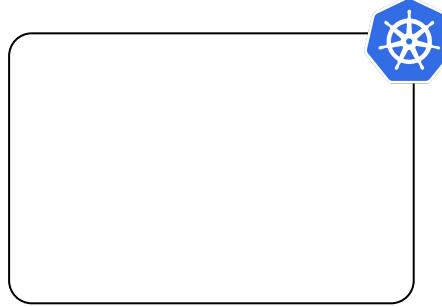


Me

## Process in Kubernetes (Cont.)

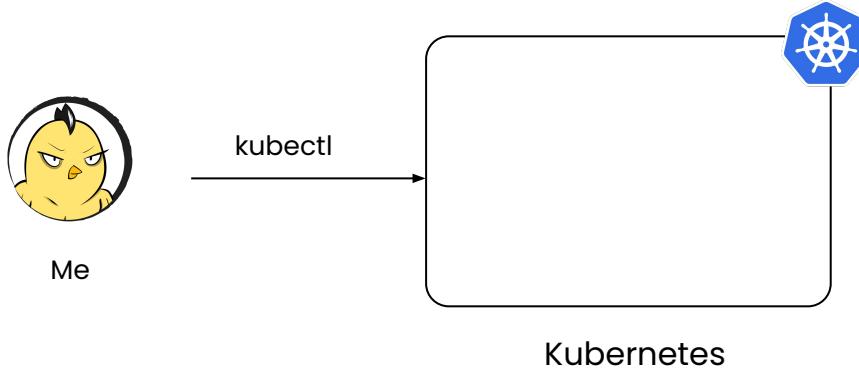


Me

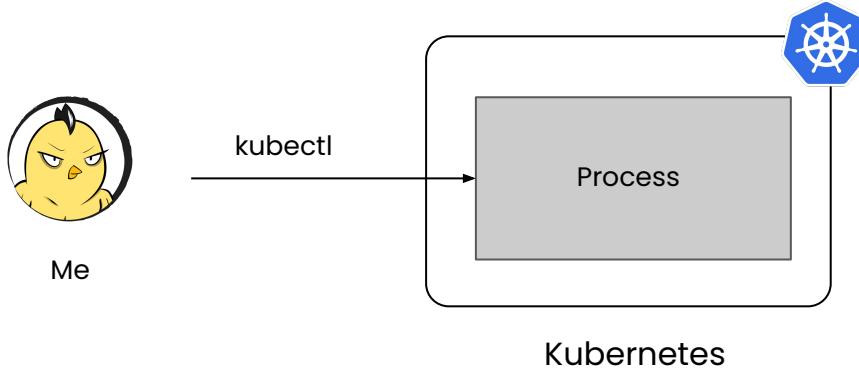


Kubernetes

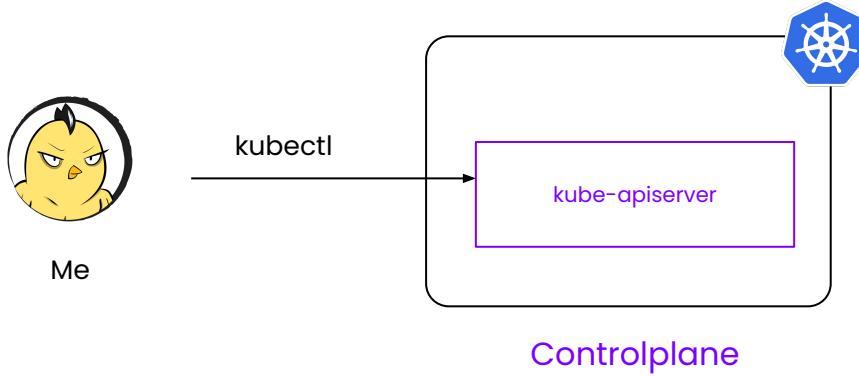
## Process in Kubernetes (Cont.)



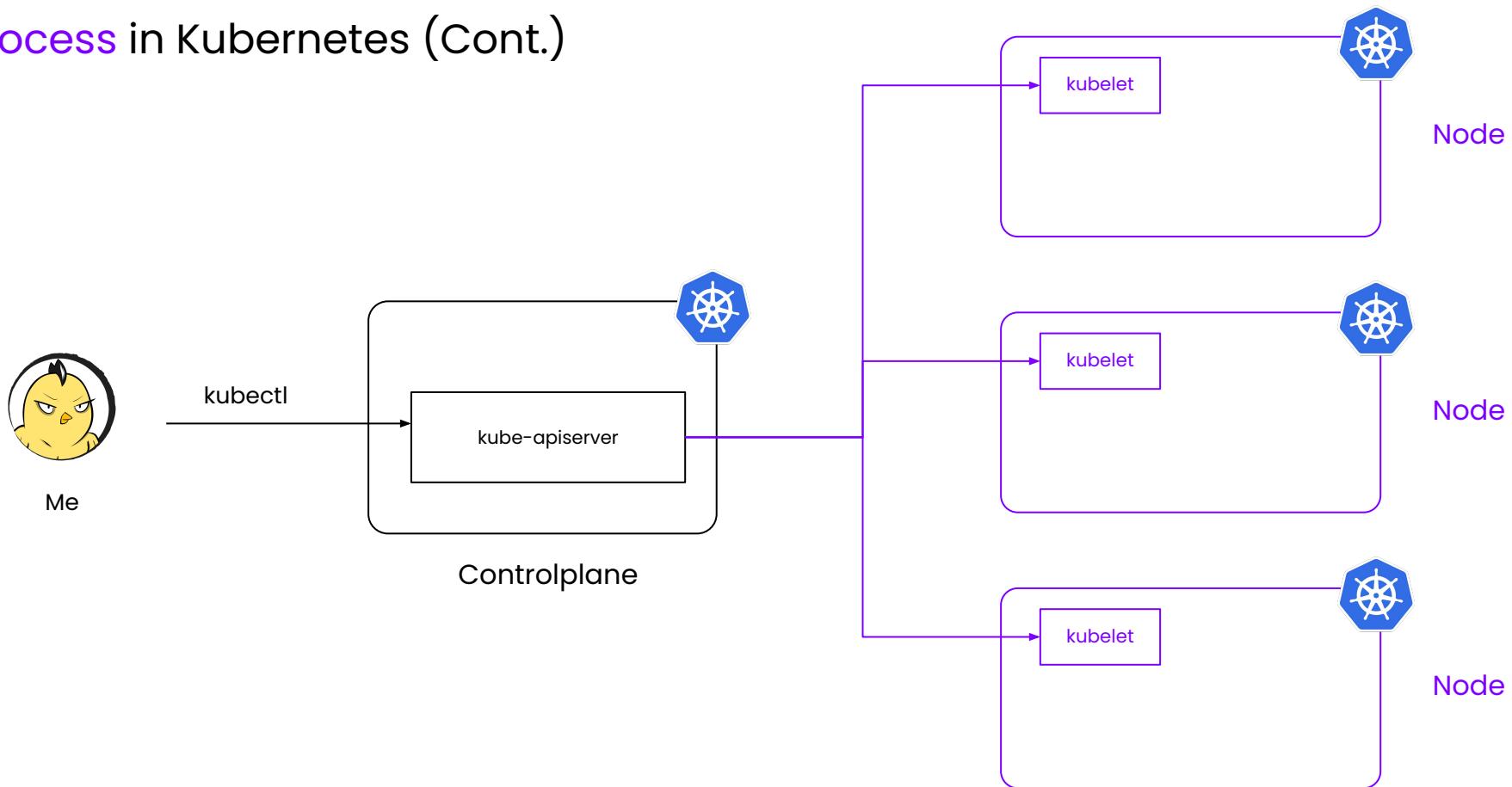
## Process in Kubernetes (Cont.)



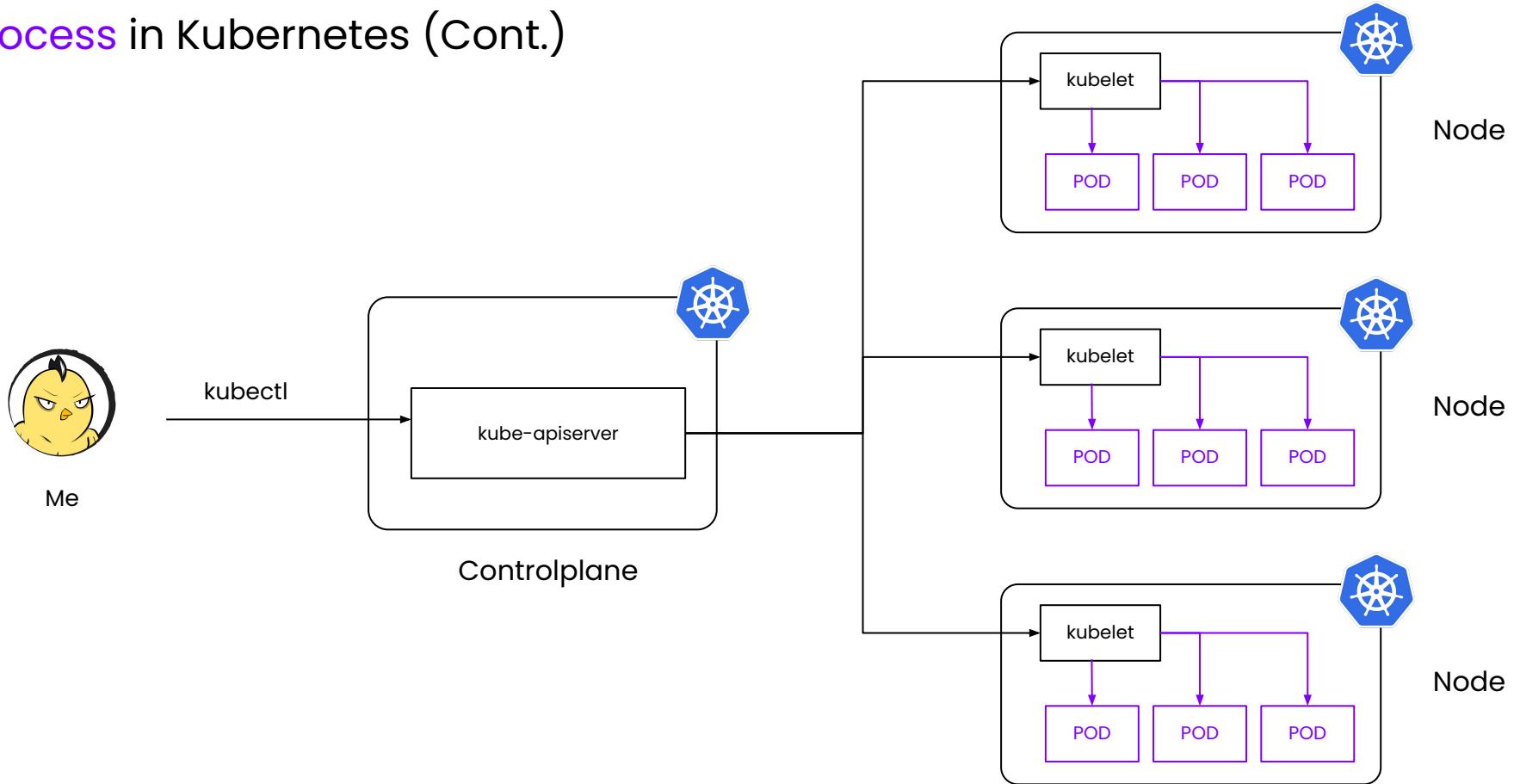
## Process in Kubernetes (Cont.)



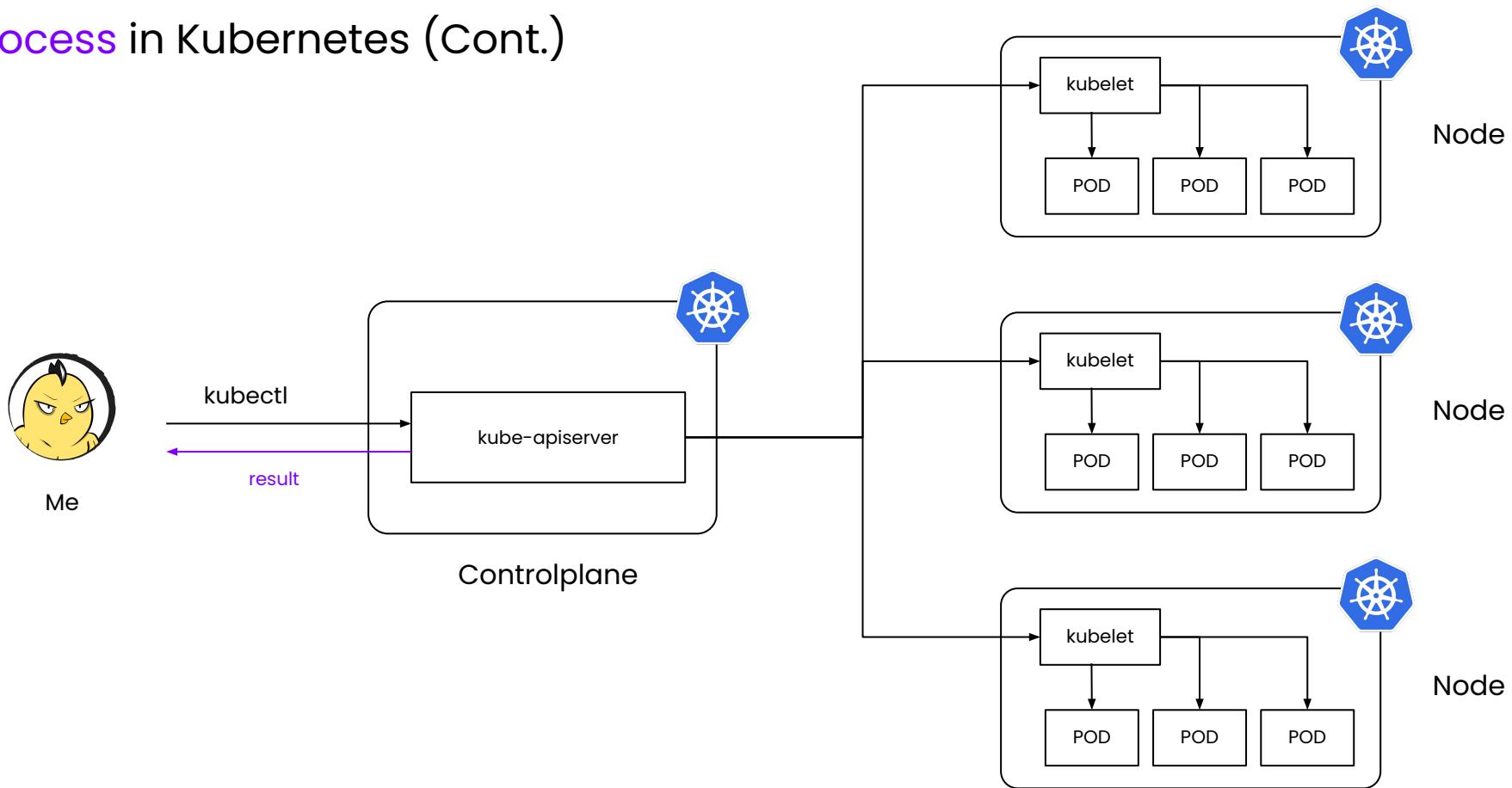
## Process in Kubernetes (Cont.)



## Process in Kubernetes (Cont.)



## Process in Kubernetes (Cont.)



# Kubernetes Component (Basic)

- Kubernetes Architecture (Basic) -

# Kubernetes Component (Basic)

# Kubernetes Component (Basic)

## 1. Kubernetes Client

# Kubernetes Component (Basic)

## 1. Kubernetes Client

- **kubectl**



# Kubernetes Component (Basic)

## 1. Kubernetes Client

- **kubectl**

## 2. Kubernetes Controlplane



Controlplane

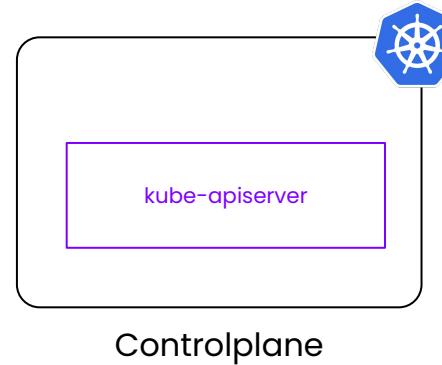
# Kubernetes Component (Basic)

## 1. Kubernetes Client

- **kubectl**

## 2. Kubernetes Controlplane

- **kube-apiserver**



# Kubernetes Component (Basic)

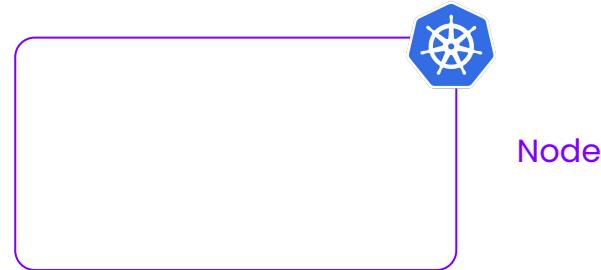
## 1. Kubernetes Client

- **kubectl**

## 2. Kubernetes Controlplane

- **kube-apiserver**

## 3. Kubernetes Node



# Kubernetes Component (Basic)

## 1. Kubernetes Client

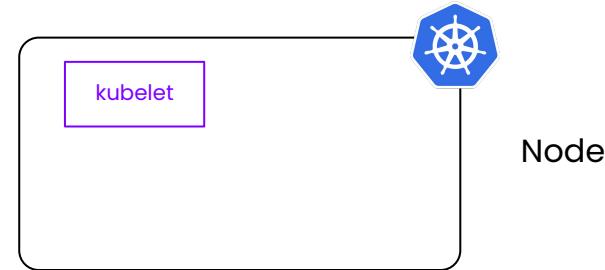
- **kubectl**

## 2. Kubernetes Controlplane

- **kube-apiserver**

## 3. Kubernetes Node

- **kubelet**



# Kubernetes Component (Basic)

## 1. Kubernetes Client

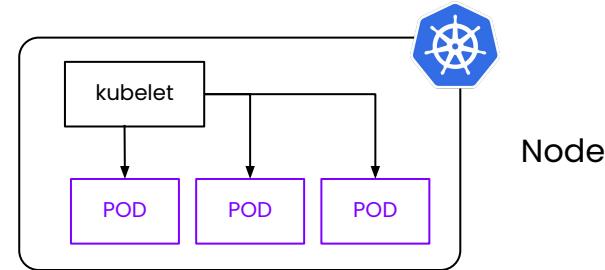
- **kubectl**

## 2. Kubernetes Controlplane

- **kube-apiserver**

## 3. Kubernetes Node

- **kubelet**
- **pod**



# Kubernetes Context

- Kubernetes Fundamental -

## Kubernetes Context (Cont.)

## Kubernetes Context (Cont.)

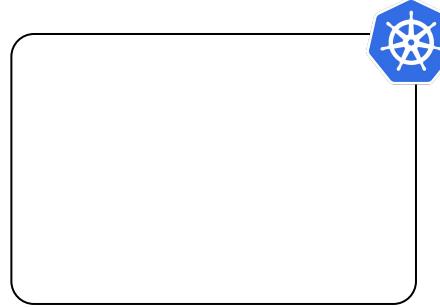


Me

# Kubernetes Context (Cont.)

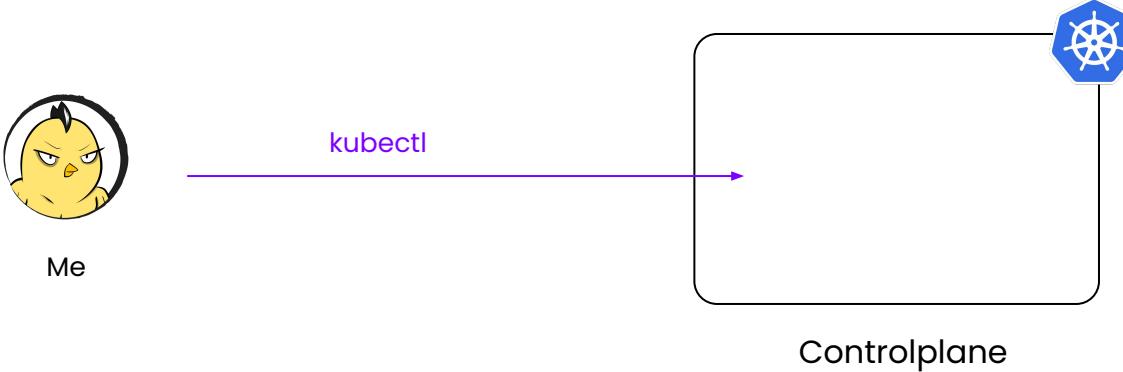


Me

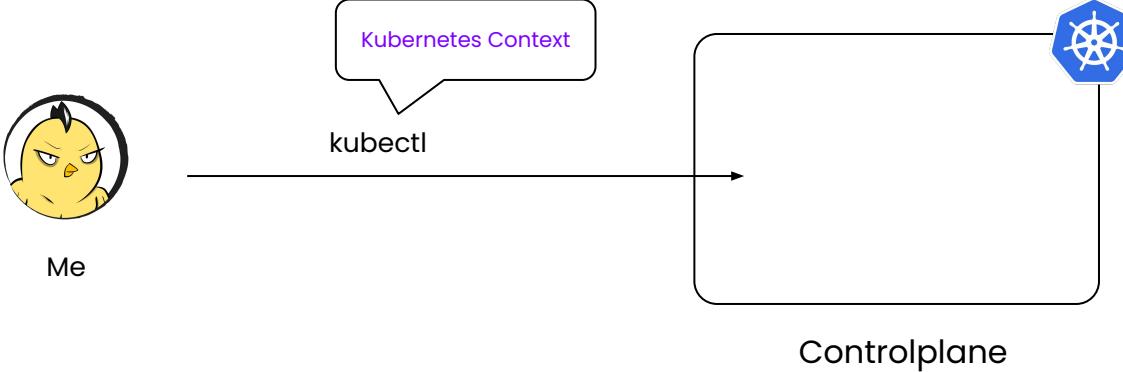


Controlplane

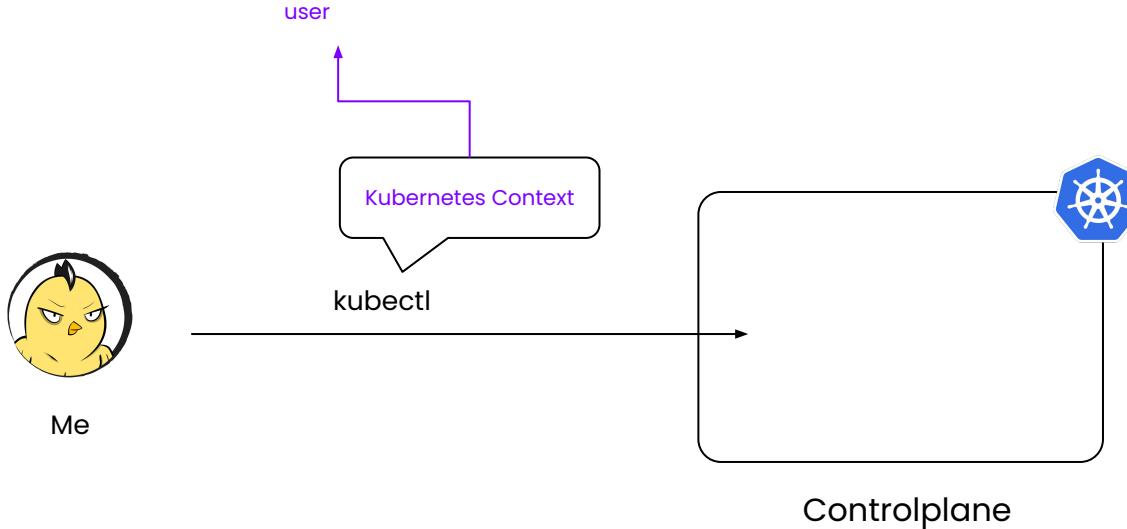
## Kubernetes Context (Cont.)



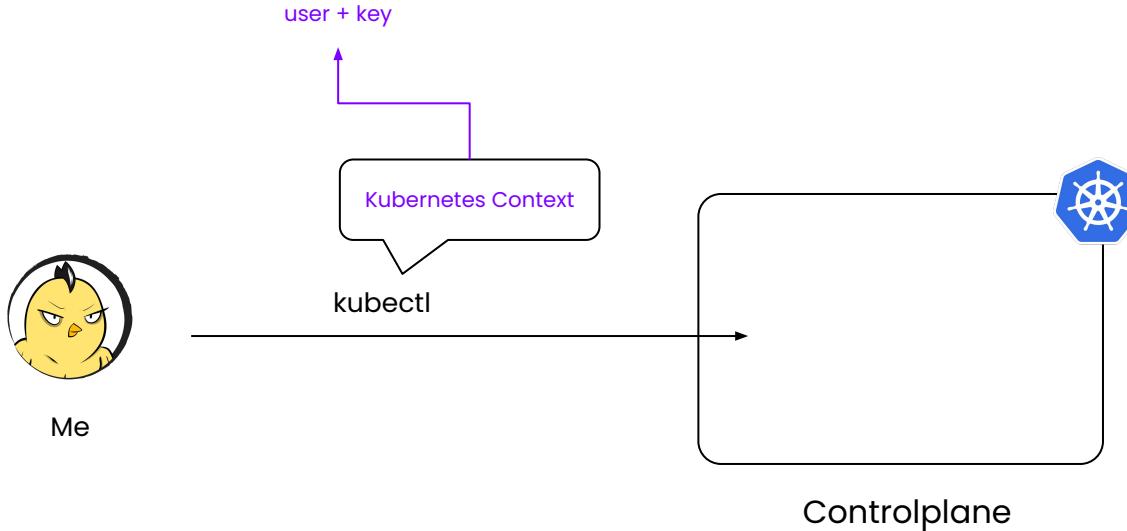
# Kubernetes Context (Cont.)



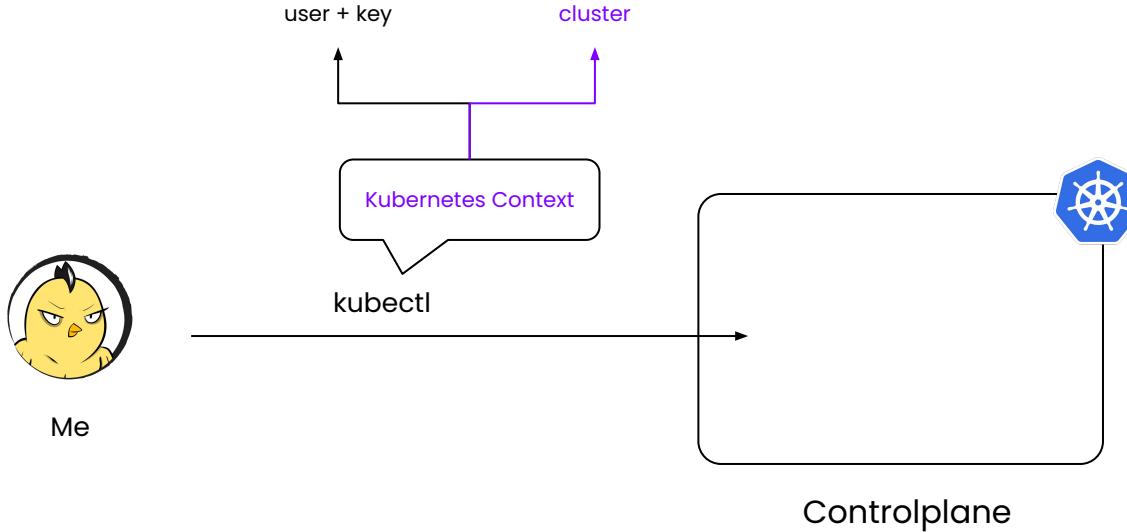
# Kubernetes Context (Cont.)



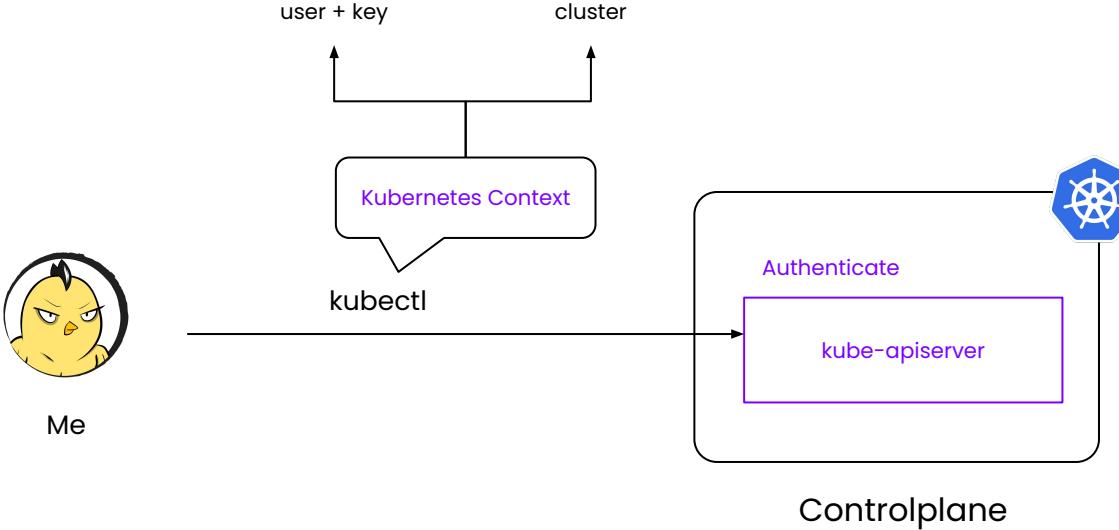
# Kubernetes Context (Cont.)



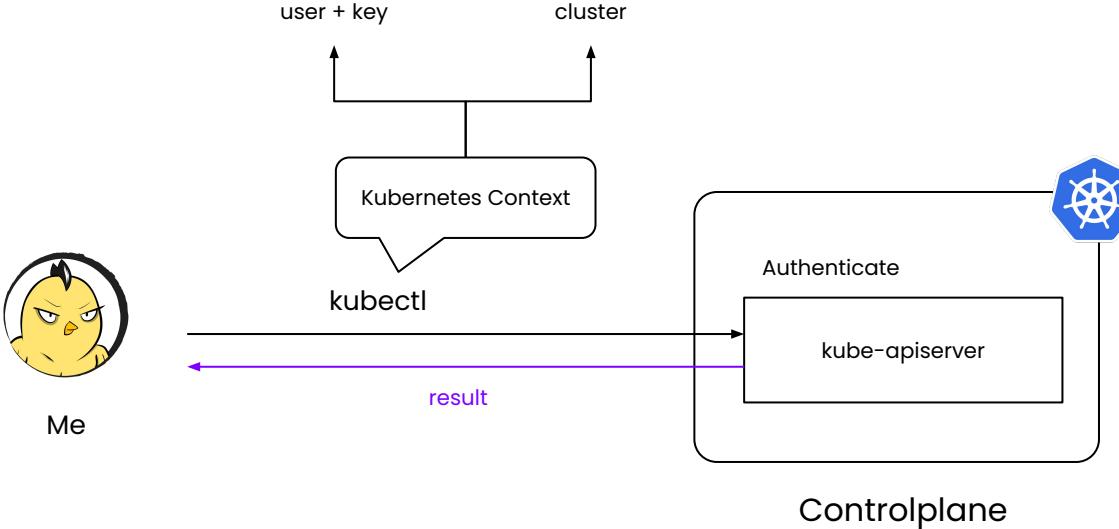
# Kubernetes Context (Cont.)



# Kubernetes Context (Cont.)



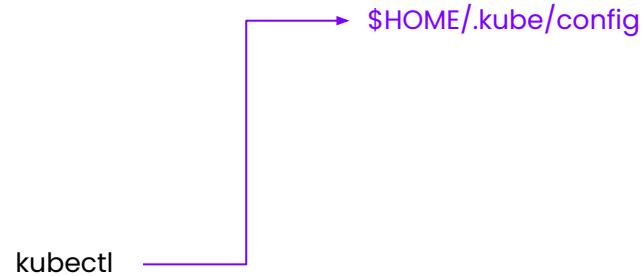
# Kubernetes Context (Cont.)



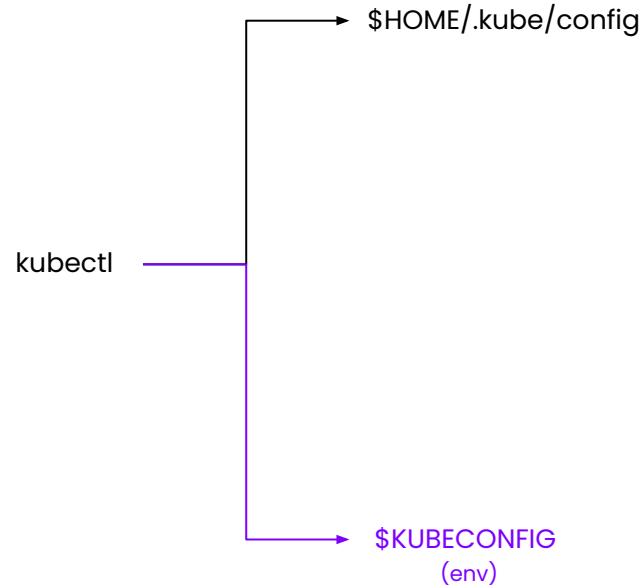
## Kubernetes Context - path (Cont.)

kubectl

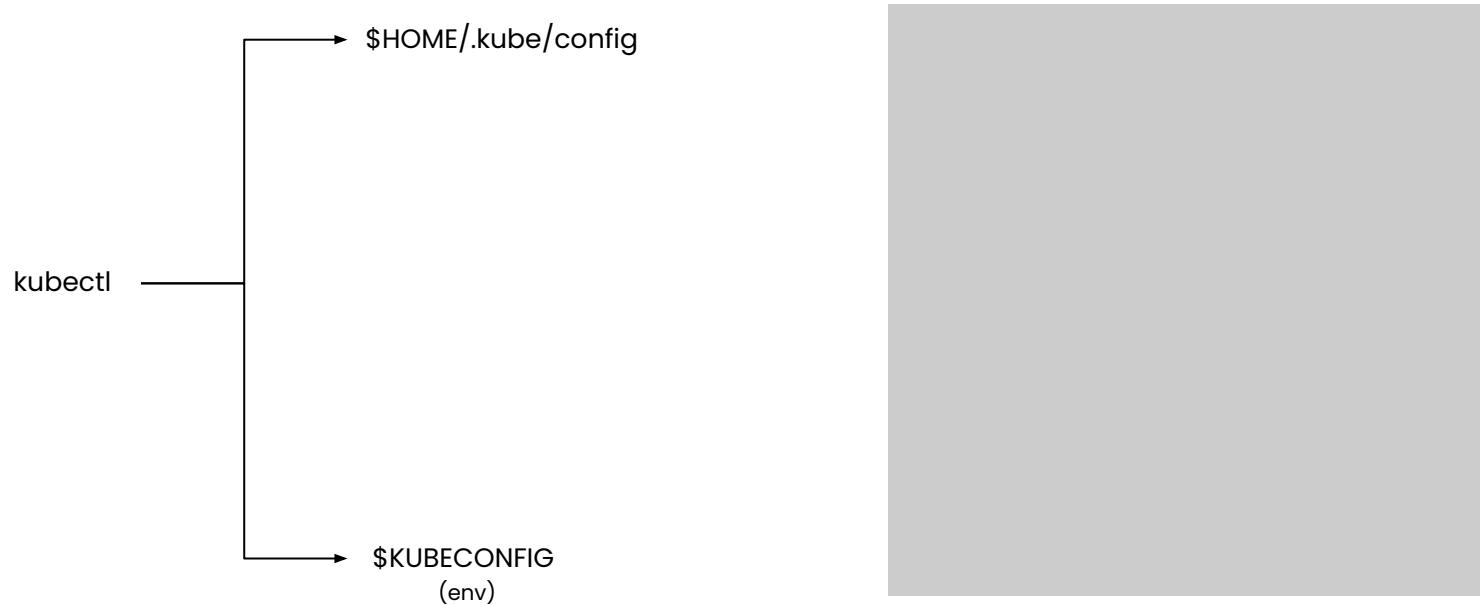
## Kubernetes Context - path (Cont.)



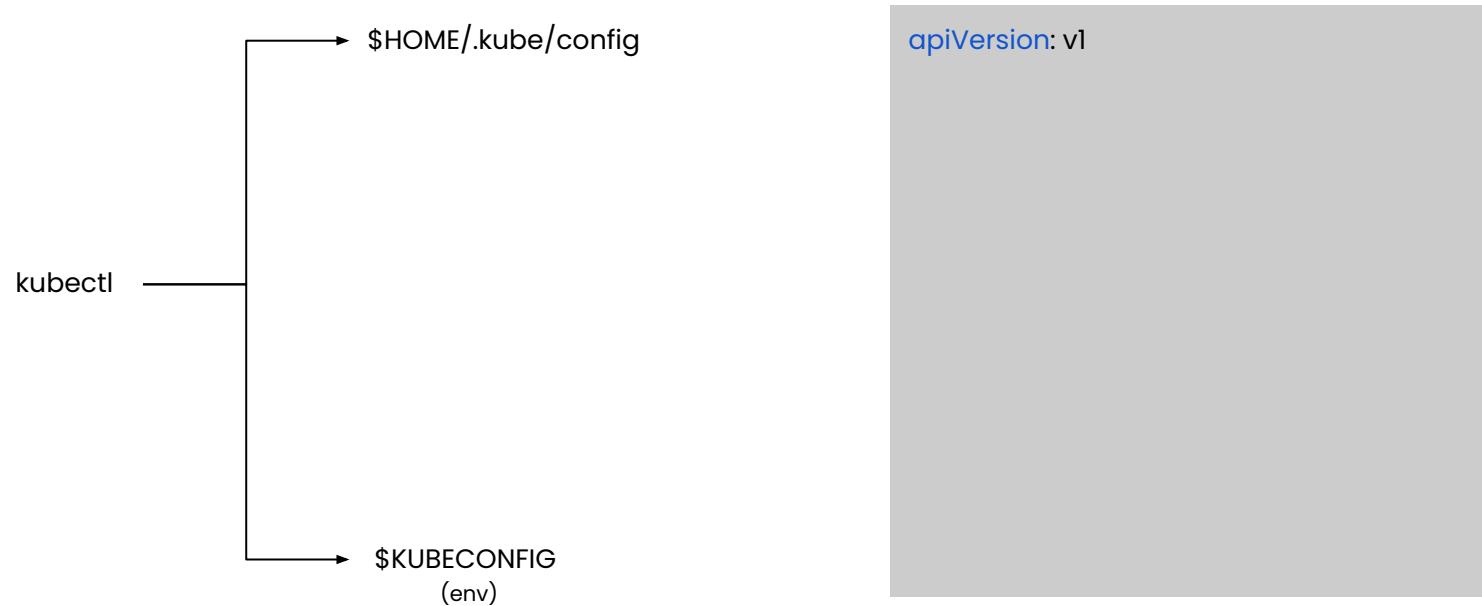
## Kubernetes Context - path (Cont.)



## Kubernetes Context - path (Cont.)



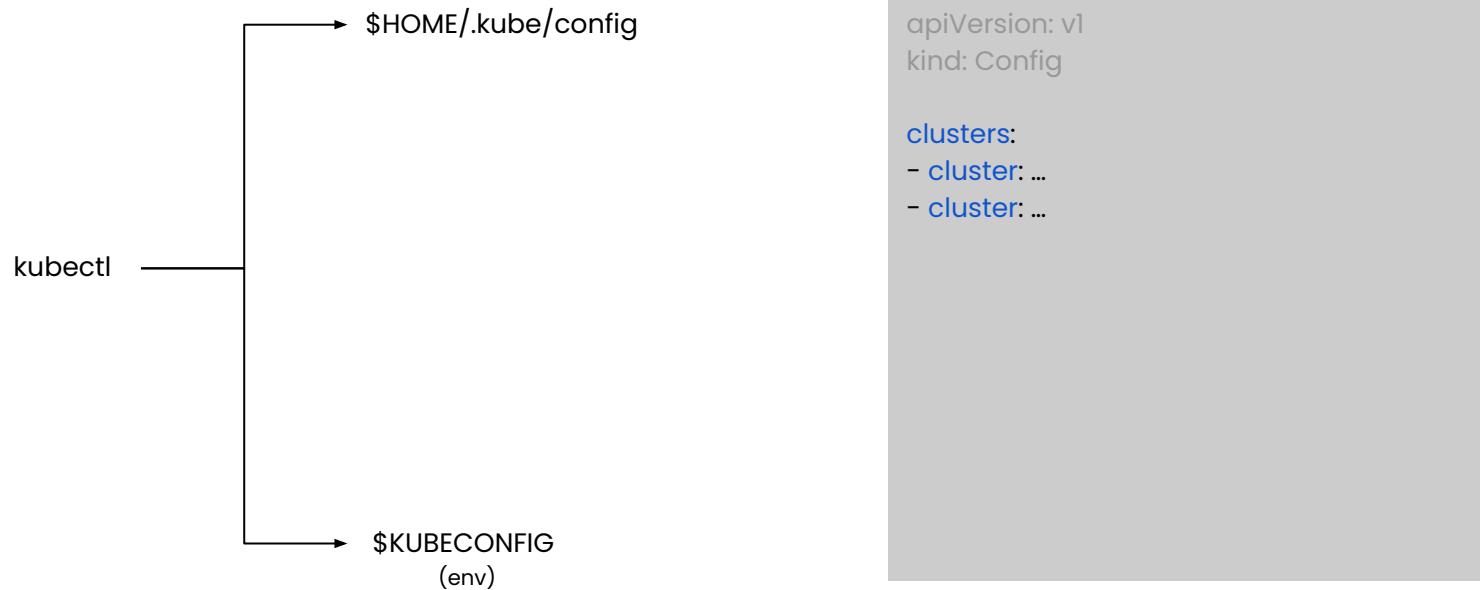
## Kubernetes Context - path (Cont.)



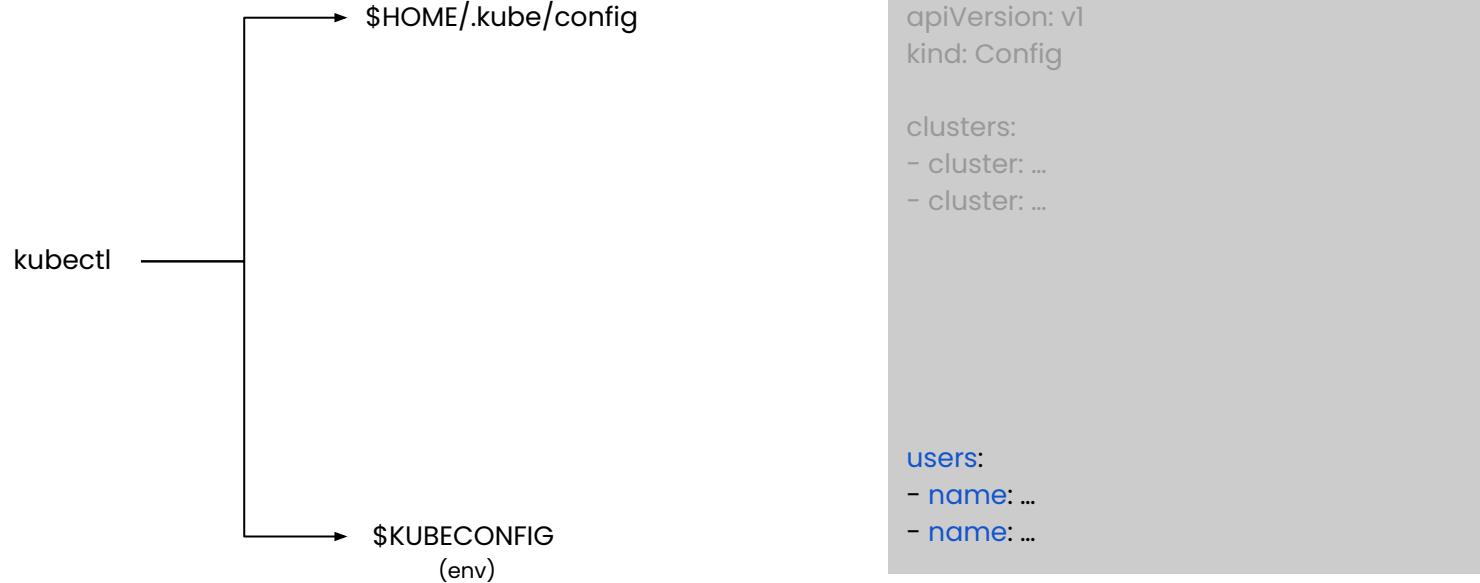
## Kubernetes Context - path (Cont.)



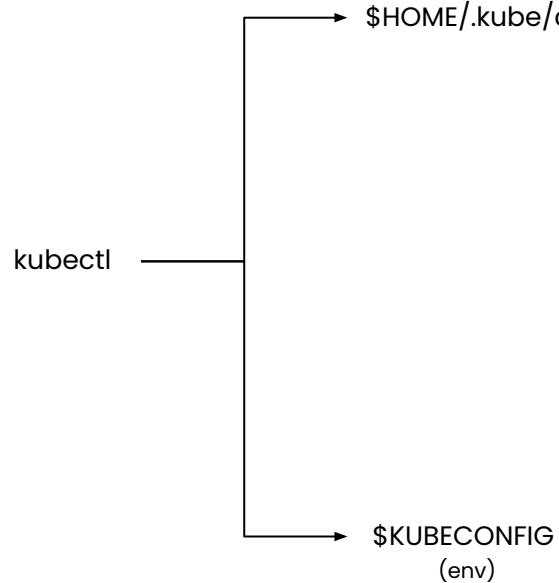
## Kubernetes Context - path (Cont.)



## Kubernetes Context - path (Cont.)



## Kubernetes Context - path (Cont.)



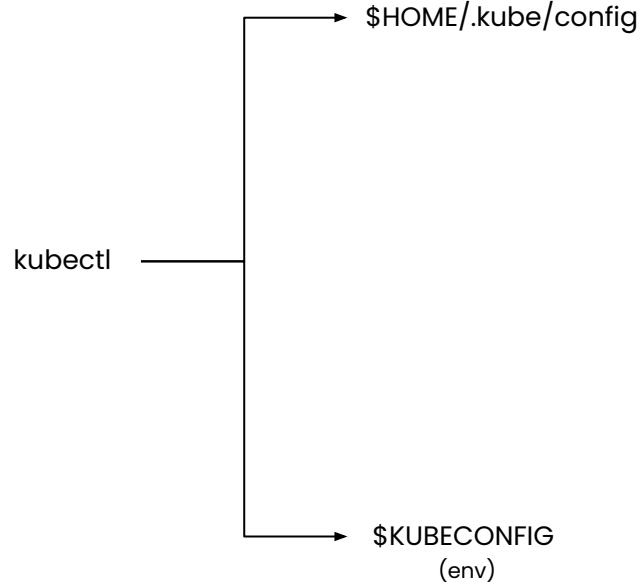
```
apiVersion: v1
kind: Config

clusters:
- cluster: ...
- cluster: ...

contexts:
- context:
  cluster: cluster-1
  user: user-1
  name: context-1
- context: ...

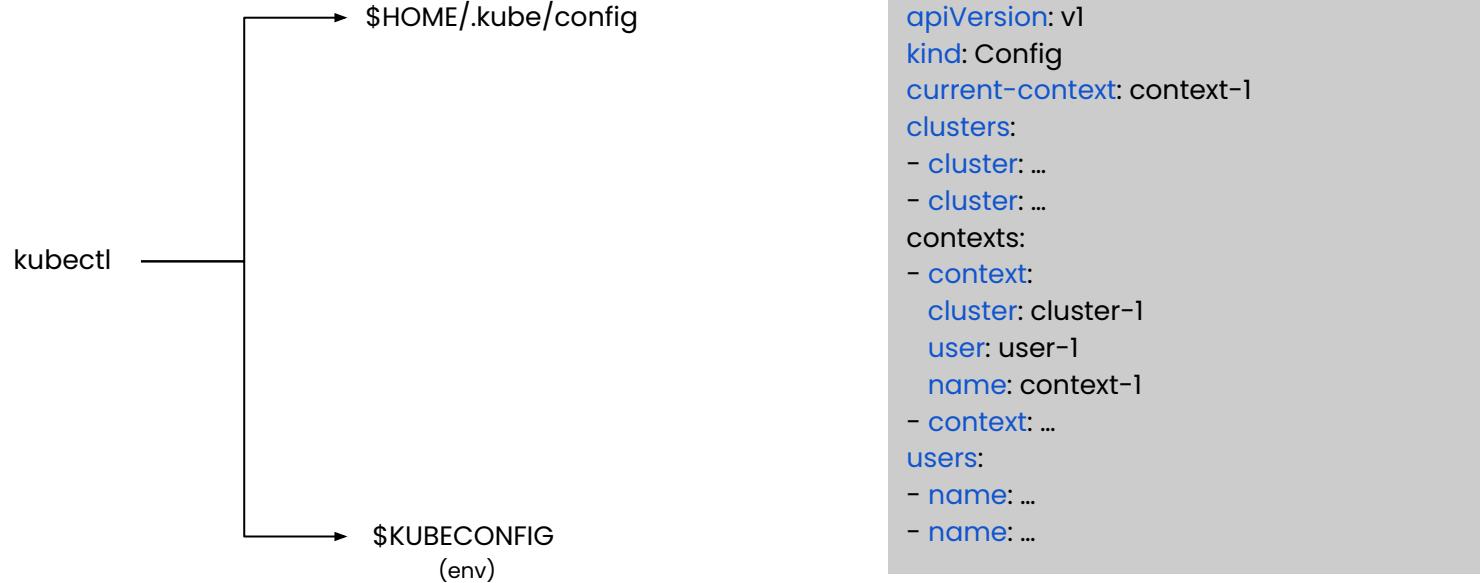
users:
- name: ...
- name: ...
```

## Kubernetes Context - path (Cont.)

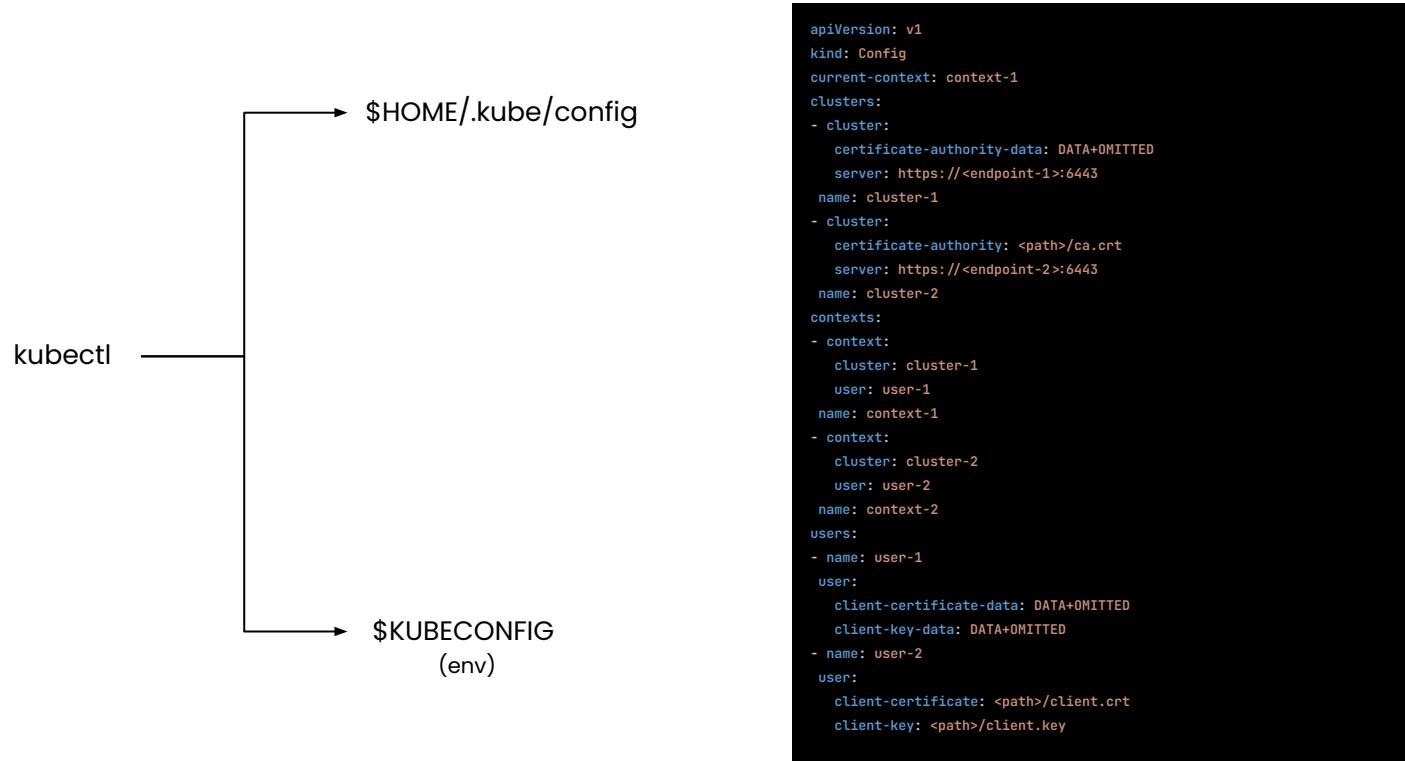


```
apiVersion: v1
kind: Config
current-context: context-1
clusters:
- cluster: ...
- cluster: ...
contexts:
- context:
  cluster: cluster-1
  user: user-1
  name: context-1
- context: ...
users:
- name: ...
- name: ...
```

## Kubernetes Context - path (Cont.)



# Kubernetes Context - path (Cont.)



# Kubernetes Basic Command (Context)

- Kubernetes Fundamental -

# Kubernetes Command (Context) (Cont.)

# Kubernetes Command (Context) (Cont.)

## 1. **kubectl config view**

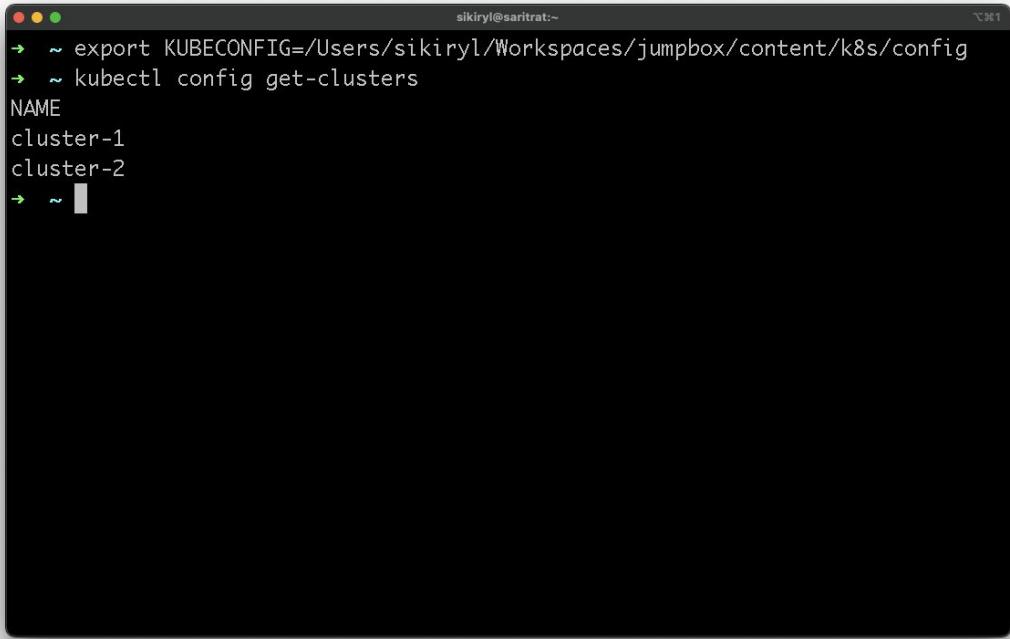
# Kubernetes Command (Context) (Cont.)

## 1. **kubectl config view**

```
sikiryl@saritrat:~
→ ~ export KUBECONFIG=/Users/sikiryl/Workspaces/jumpbox/content/k8s/config
→ ~ kubectl config view
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data: DATA+OMITTED
    server: https://<endpoint-1>:6443
    name: cluster-1
- cluster:
    certificate-authority: <path>/ca.crt
    server: https://<endpoint-2>:6443
    name: cluster-2
contexts:
- context:
    cluster: cluster-1
    user: user-1
    name: context-1
- context:
    cluster: cluster-2
    user: user-2
```

# Kubernetes Command (Context) (Cont.)

1. **kubectl config view**
2. **kubectl config get-clusters**

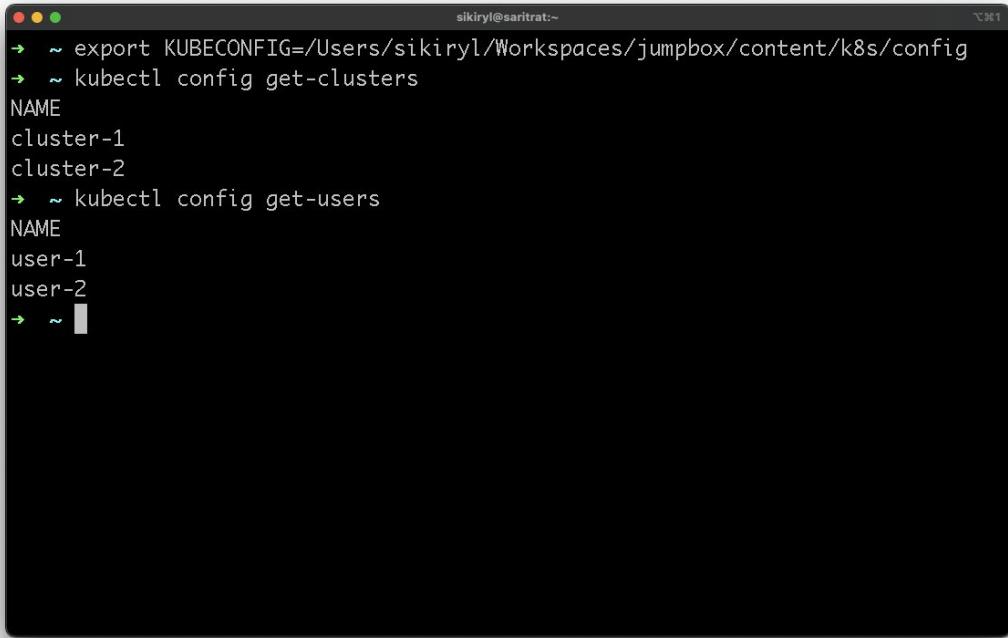


```
sikiryl@saritrat:~
→ ~ export KUBECONFIG=/Users/sikiryl/Workspaces/jumpbox/content/k8s/config
→ ~ kubectl config get-clusters
NAME
cluster-1
cluster-2
→ ~ |
```

A screenshot of a macOS terminal window titled "sikiryl@saritrat:~". The window shows two command-line entries. The first entry is "export KUBECONFIG=/Users/sikiryl/Workspaces/jumpbox/content/k8s/config". The second entry is "kubectl config get-clusters". The output of the second command is displayed below it, showing two clusters named "cluster-1" and "cluster-2". The terminal has a dark background with light-colored text.

# Kubernetes Command (Context) (Cont.)

1. **kubectl config view**
2. **kubectl config get-clusters**
3. **kubectl config get-users**



```
sikiryl@saritrat:~
→ ~ export KUBECONFIG=/Users/sikiryl/Workspaces/jumpbox/content/k8s/config
→ ~ kubectl config get-clusters
NAME
cluster-1
cluster-2
→ ~ kubectl config get-users
NAME
user-1
user-2
→ ~
```

A screenshot of a macOS terminal window titled "sikiryl@saritrat:~". The terminal shows the execution of three commands: "export KUBECONFIG=/Users/sikiryl/Workspaces/jumpbox/content/k8s/config", "kubectl config get-clusters", and "kubectl config get-users". The output for "get-clusters" lists two clusters: "cluster-1" and "cluster-2". The output for "get-users" lists two users: "user-1" and "user-2". The terminal has a dark background with light-colored text.

# Kubernetes Command (Context) (Cont.)

1. **kubectl config view**
2. **kubectl config get-clusters**
3. **kubectl config get-users**
4. **kubectl config get-contexts**

```
sikiryl@saritrat:~
→ ~ export KUBECONFIG=/Users/sikiryl/Workspaces/jumpbox/content/k8s/config
→ ~ kubectl config get-clusters
NAME
cluster-1
cluster-2
→ ~ kubectl config get-users
NAME
user-1
user-2
→ ~ kubectl config get-contexts
CURRENT   NAME      CLUSTER      AUTHINFO      NAMESPACE
*         context-1  cluster-1    user-1
                           cluster-2    user-2
→ ~
```

# Kubernetes Command (Context) (Cont.)

1. **kubectl config view**
2. **kubectl config get-clusters**
3. **kubectl config get-users**
4. **kubectl config get-contexts**
5. **kubectl config current-context**

```
sikiryl@saritrat:~
→ ~ export KUBECONFIG=/Users/sikiryl/Workspaces/jumpbox/content/k8s/config
→ ~ kubectl config get-clusters
NAME
cluster-1
cluster-2
→ ~ kubectl config get-users
NAME
user-1
user-2
→ ~ kubectl config get-contexts
CURRENT   NAME      CLUSTER      AUTHINFO      NAMESPACE
*          context-1  cluster-1    user-1
                           cluster-2    user-2
→ ~ kubectl config current-context
context-1
→ ~ █
```

## Kubernetes Command (Context) (Cont.)

1. **kubectl config view**
2. **kubectl config get-clusters**
3. **kubectl config get-users**
4. **kubectl config get-contexts**
5. **kubectl config current-context**
- 6.



# Activity

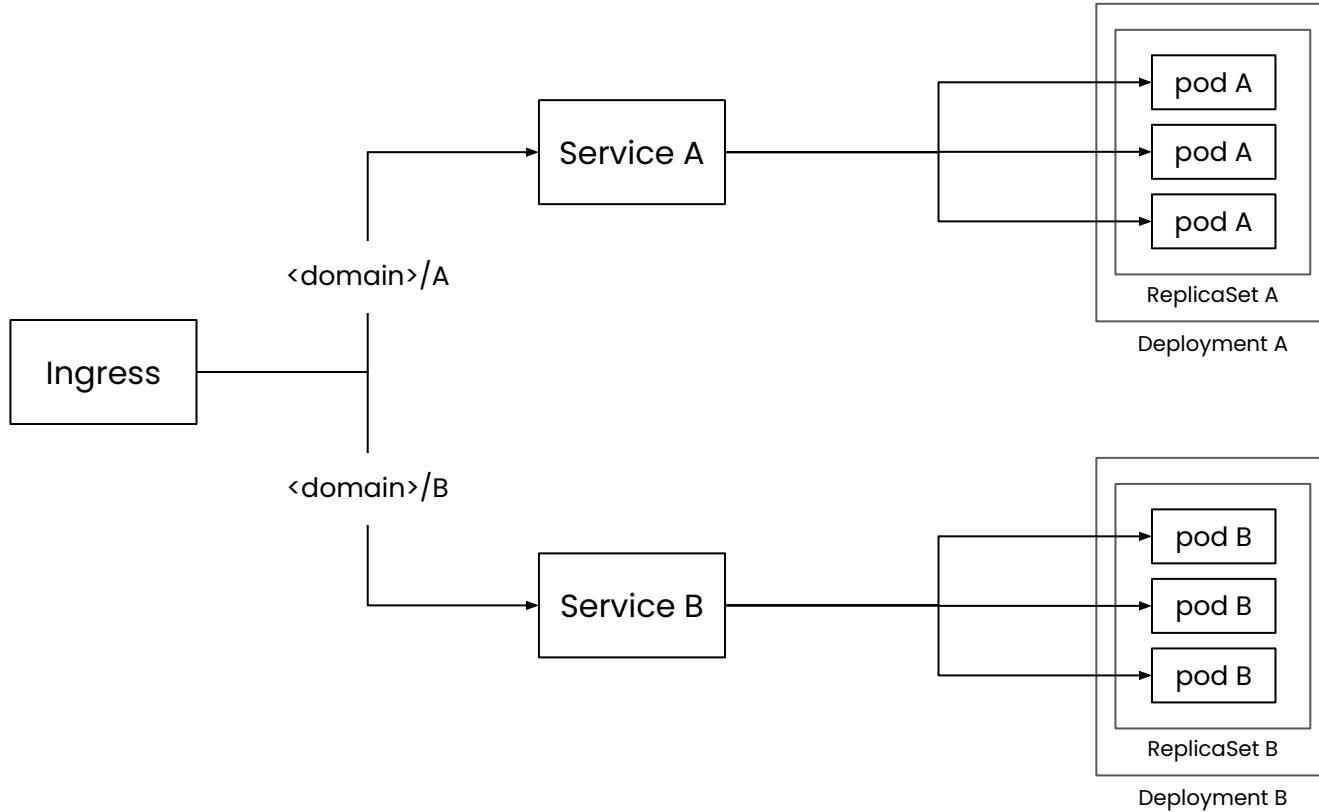
# Pod in Action

- Kubernetes Fundamental -

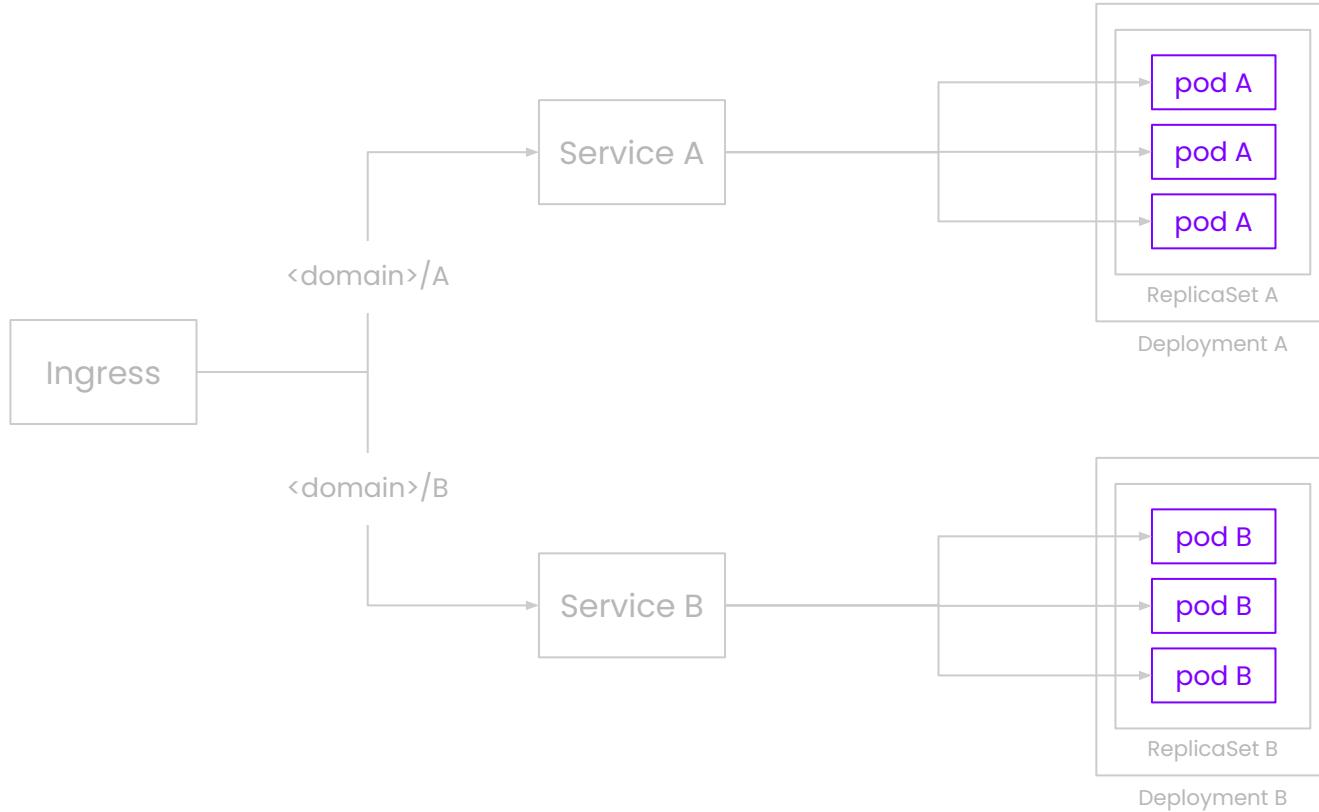
# 1. POD

the **smallest** deployable units of Kubernetes

# Kubernetes in Action



# Kubernetes in Action

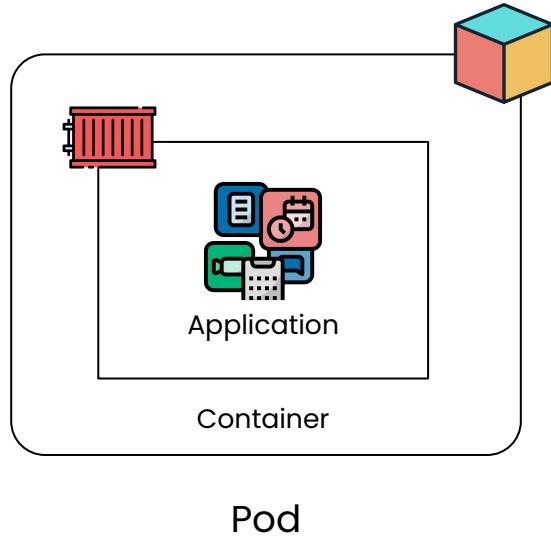


## How to create Pod? (Cont.)

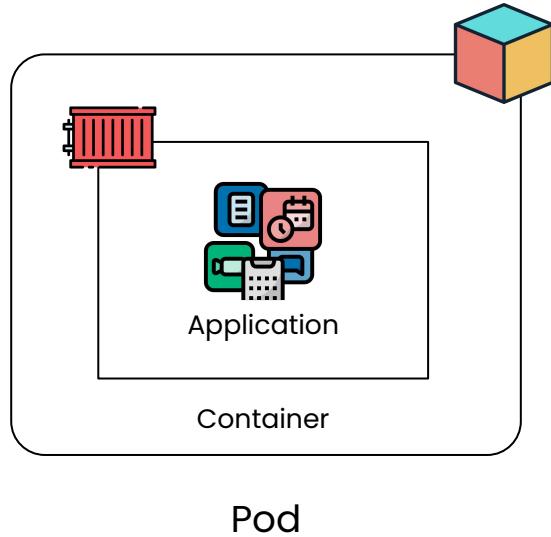
Jumpbox®

เจ้าของลิขสิทธิ์ อนุญาติให้ใช้สิ่งดังนี้เพื่อการเรียนรู้ส่วนบุคคลเท่านั้น และขอสงวนลิขสิทธิ์ ห้ามมิให้เผยแพร่ในที่สาธารณะ ผู้อื่นเดินทาง จะถูกดำเนินคดีตามกฎหมาย

## How to create Pod? (Cont.)



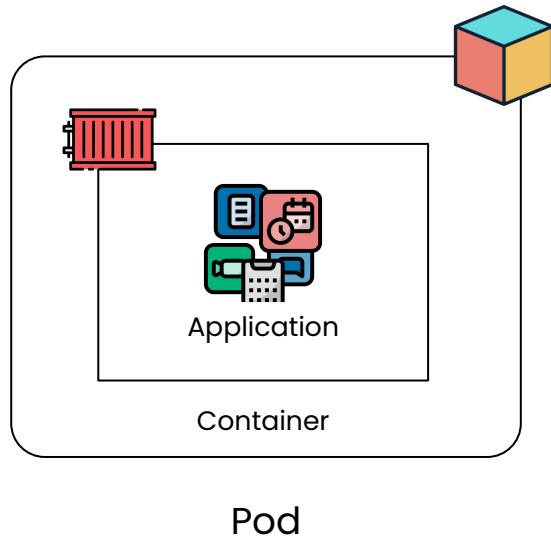
## How to create Pod? (Cont.)



```
$ kubectl run <name> --image <image>
```

1. Imperative command

## How to create Pod? (Cont.)



```
$ kubectl run <name> --image <image>
```

1. Imperative command



```
$ kubectl apply -f <file.yaml or file.json>
```

2. Declarative file

# Kubernetes Object Management

# Kubernetes Object Management (Cont.)

## Management techniques

### Warning:

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

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

### Reference Pictures:

- [Object Management](#)

## Imperative commands

Run an instance of the nginx container by creating a Deployment object.

- `kubectl run nginx --image nginx`

Do the same thing using a different syntax:

- `kubectl create deployment nginx --image nginx`

# Imperative object configuration

Create the objects defined in a configuration file:

- `kubectl create -f nginx.yaml`

Delete the objects defined in two configuration files:

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

Update the objects defined in a configuration file by overwriting the live configuration:

- `kubectl replace -f nginx.yaml`

## Declarative object configuration

Process all object configuration files in the configs directory, and create or patch the live objects. You can first diff to see what changes are going to be made, and then apply:

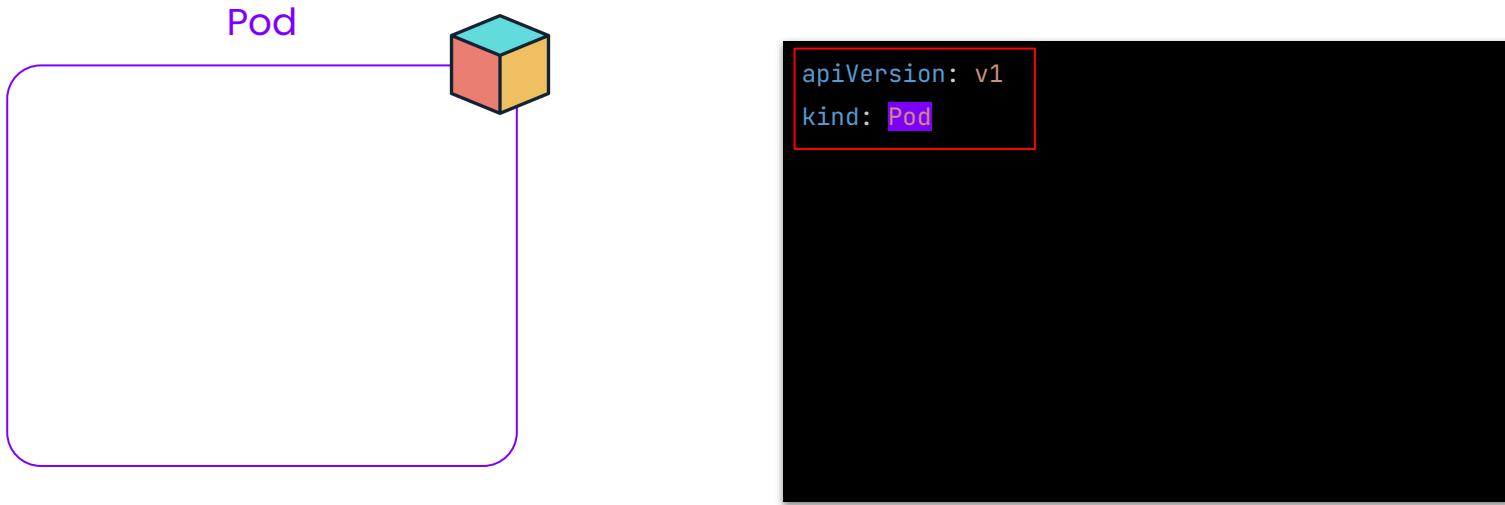
- `kubectl diff -f configs/`
- `kubectl apply -f configs/`

Recursively process directories:

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

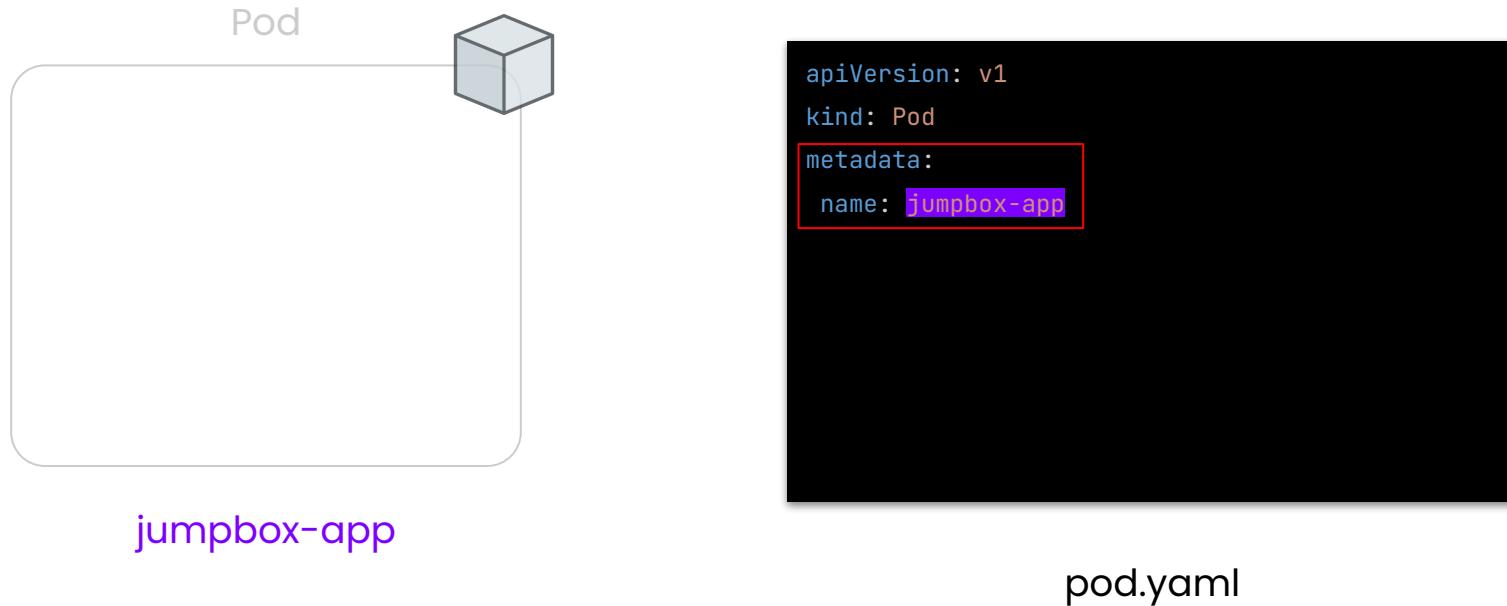
## Pod Structure with **Declarative** file (Cont.)

## Pod Structure with Declarative file (Cont.)

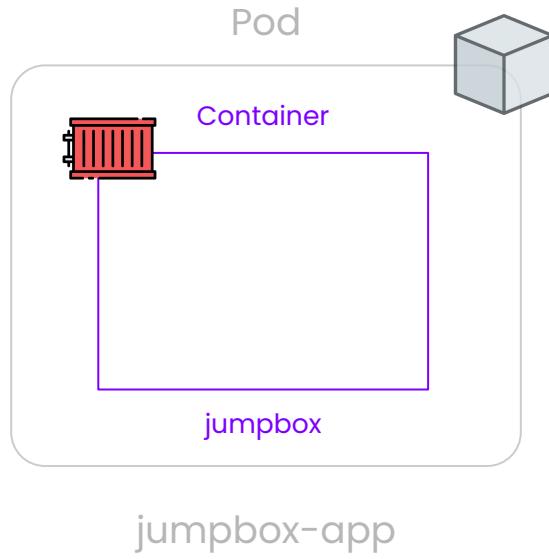


pod.yaml

## Pod Structure with Declarative file (Cont.)



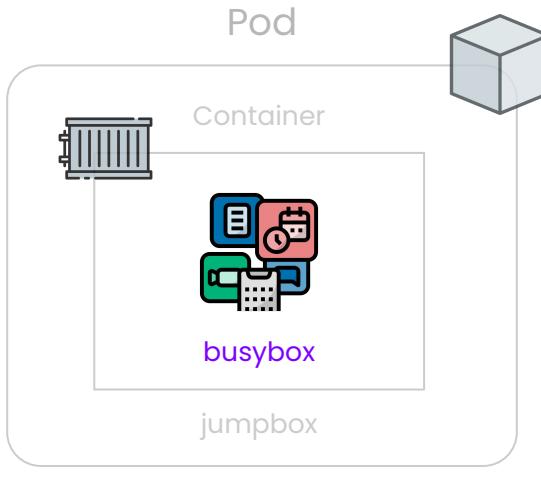
# Pod Structure with Declarative file (Cont.)



```
apiVersion: v1
kind: Pod
metadata:
  name: jumpbox-app
spec:
  containers:
    - name: jumpbox
```

pod.yaml

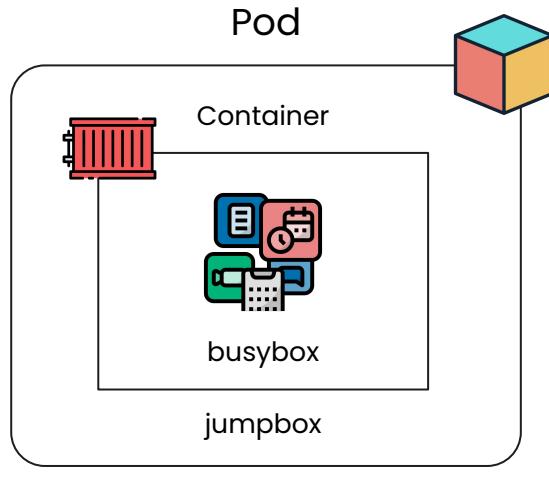
# Pod Structure with Declarative file (Cont.)



```
apiVersion: v1
kind: Pod
metadata:
  name: jumpbox-app
spec:
  containers:
    - name: jumpbox
      image: busybox
      command: ["sh", "-c"]
      args: ["echo Hello Jumpbox!! && sleep 3600"]
```

pod.yaml

## Pod Structure with Declarative file (Cont.)



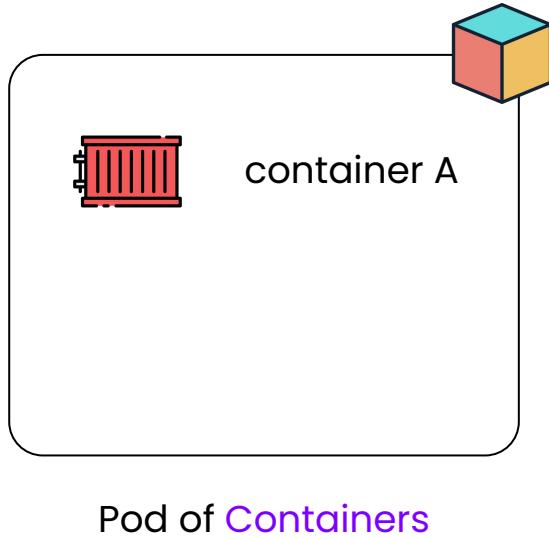
jumpbox-app

```
apiVersion: v1
kind: Pod
metadata:
  name: jumpbox-app
spec:
  containers:
    - name: jumpbox
      image: busybox
      command: ["sh", "-c"]
      args: ["echo Hello Jumpbox!! && sleep 3600"]
```

pod.yaml

## Multiple container in Pod (Cont.)

## Multiple container in Pod (Cont.)

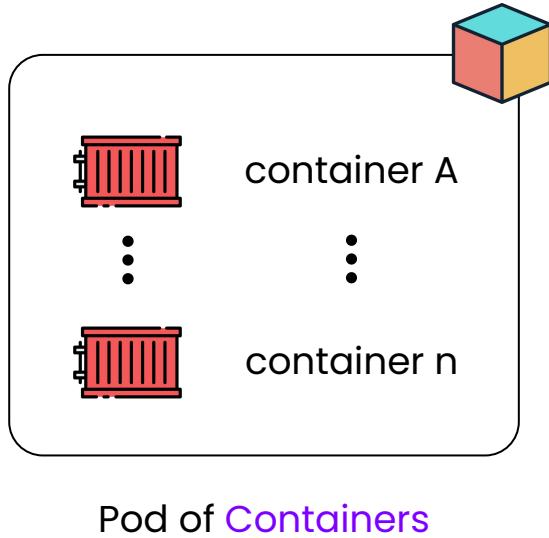


```
apiVersion: v1
kind: Pod
metadata:
  name: jumpbox-app
spec:
  containers:
    - name: jumpbox-app
      ...

```

pod.yaml

## Multiple container in Pod (Cont.)



```
apiVersion: v1
kind: Pod
metadata:
  name: jumpbox-app
spec:
  containers:
    - name: jumpbox-app
      ...
    - name: <container-name-2>
      ...
    - name: <container-name-n>
```

pod.yaml

Jumpbox®

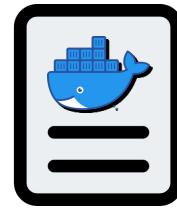
## Pod: YAML related to Dockerfile (Cont.)

# Pod: YAML related to Dockerfile (Cont.)



pods.yaml

```
apiVersion: v1
kind: Pod
metadata:
  name: jumpbox-app
spec:
  containers:
  - name: jumpbox
    image: busybox
    command: [ "sh", "-c" ]
    args: [ "echo Hello Jumpbox!! && sleep 3600" ]
```



Dockerfile

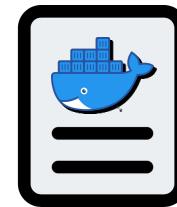
```
FROM busybox
ENTRYPOINT [ "sh", "-c" ]
CMD [ "echo Hello Jumpbox!! && sleep 3600" ]
```

## Pod: YAML related to Dockerfile (Cont.)



pods.yaml

# Override



Dockerfile

```
apiVersion: v1
kind: Pod
metadata:
  name: jumpbox-app
spec:
  containers:
    - name: jumpbox
      image: busybox
      command: ["sh", "-c"]
      args: ["echo Hello Jumpbox!! && sleep 3600"]
```

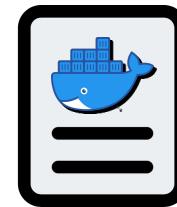
```
FROM busybox
ENTRYPOINT [ "sh", "-c" ]
CMD [ "echo Hello Jumpbox!! && sleep 3600" ]
```

## Pod: YAML related to Dockerfile (Cont.)



pods.yaml

# Override



Dockerfile

```
apiVersion: v1
kind: Pod
metadata:
  name: jumpbox-app
spec:
  containers:
    - name: jumpbox
      image: busybox
      command: ["sh", "-c"]
      args: ["echo Hello Jumpbox!! && sleep 3600"]

```

The diagram illustrates how a specific command in a Kubernetes pod's Dockerfile is overridden by a command in the pod's configuration file. A red arrow points from the highlighted command in the pod's configuration to the corresponding line in the Dockerfile, indicating that the pod's command takes precedence. Conversely, a cyan arrow points from the Dockerfile's image definition back to the pod's configuration, showing that the Dockerfile's image definition is used.

```
FROM busybox
ENTRYPOINT [ "sh", "-c" ]
CMD [ "echo Hello Jumpbox!! && sleep 3600" ]
```

## Pod: YAML related to Dockerfile (Cont.)



pods.yaml

# Override



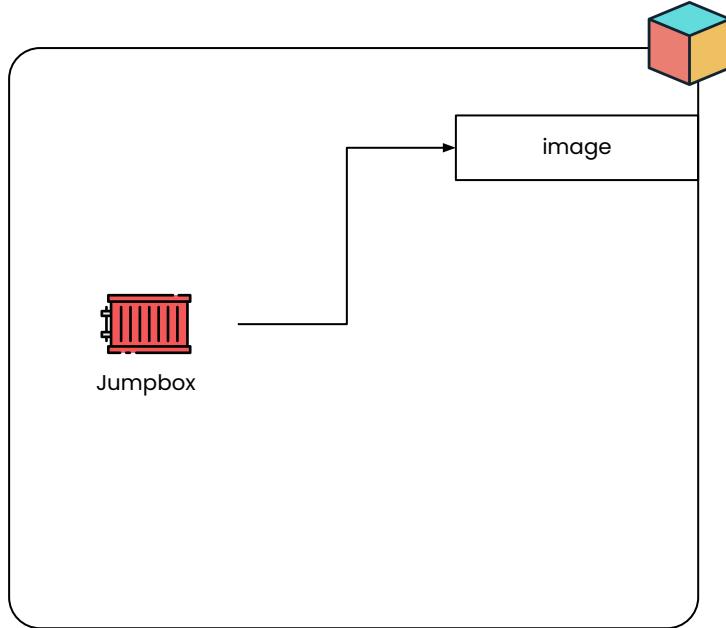
Dockerfile

```
apiVersion: v1
kind: Pod
metadata:
  name: jumpbox-app
spec:
  containers:
    - name: jumpbox
      image: busybox
      command: ["sh", "-c"]
      args: ["echo Hello Jumpbox!! && sleep 3600"]
```

```
FROM busybox
ENTRYPOINT [ "sh", "-c" ]
CMD [ "echo Hello Jumpbox!! && sleep 3600" ]
```

# A container in POD

## A container in POD (Cont.)

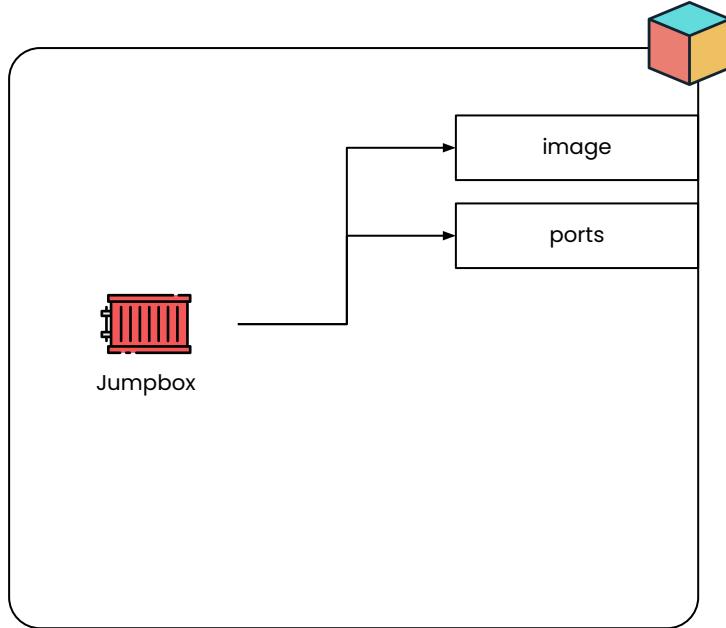


Logical view

```
...  
spec:  
  containers:  
    - name: jumpbox  
      image: busybox
```

Declarative file

## A container in POD (Cont.)

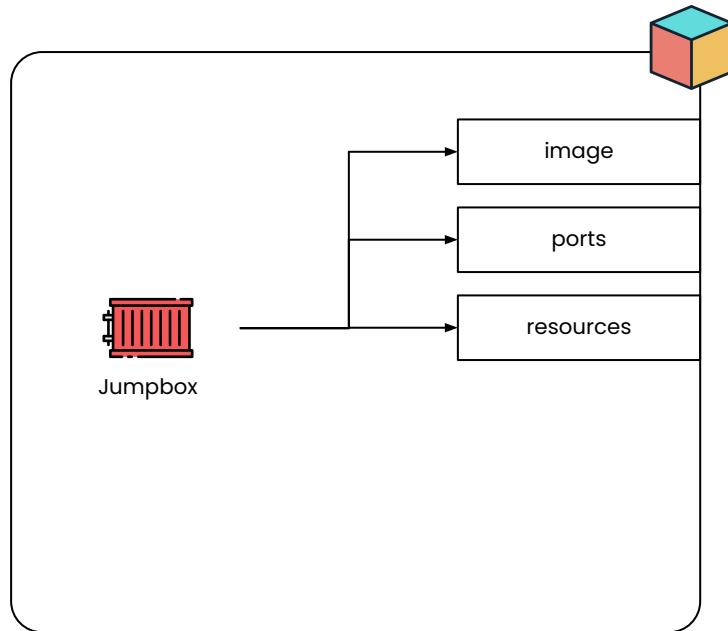


Logical view

```
...  
spec:  
  containers:  
    - name: jumpbox  
      image: busybox  
      ports:  
        - containerPort: 80
```

Declarative file

## A container in POD (Cont.)

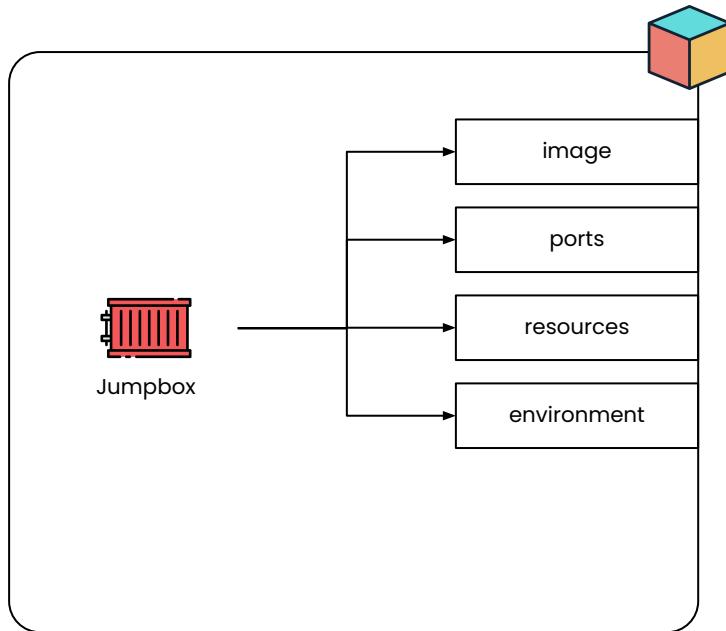


Logical view

```
...  
spec:  
  containers:  
    - name: jumpbox  
      image: busybox  
      ports:  
        - containerPort: 80  
      resources:  
        ...
```

Declarative file

## A container in POD (Cont.)

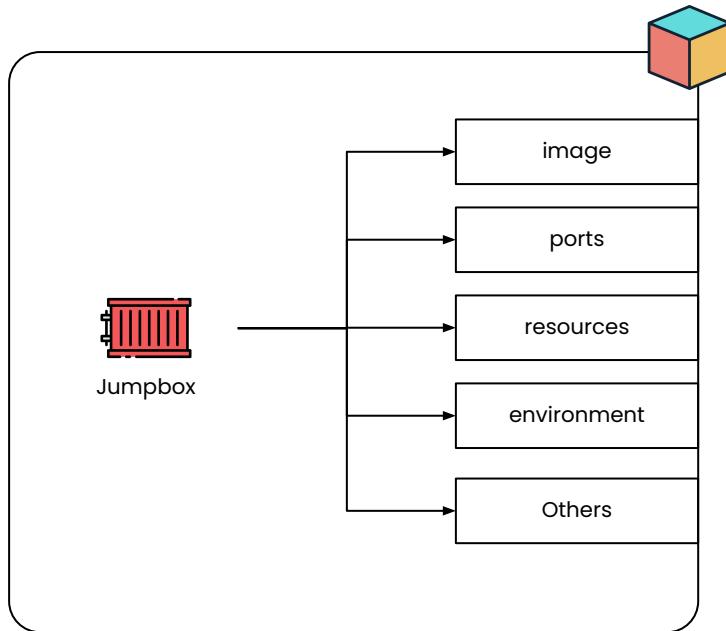


Logical view

```
...
spec:
  containers:
    - name: jumpbox
      image: busybox
      ports:
        - containerPort: 80
      resources:
        ...
      env:
        - name: DEMO_GREETING
          value: "Hello from the environment"
```

Declarative file

## A container in POD (Cont.)



Logical view

```
...
spec:
  containers:
    - name: jumpbox
      image: busybox
      ports:
        - containerPort: 80
      resources:
        ...
      env:
        - name: DEMO_GREETING
          value: "Hello from the environment"
      command: ["sh", "-c"]
      args: ["echo Hello Jumpbox!! && sleep 3600"]
```

Declarative file

**Jumpbox®**

# PODs

**With Real Action Simulator**

**Click Here**



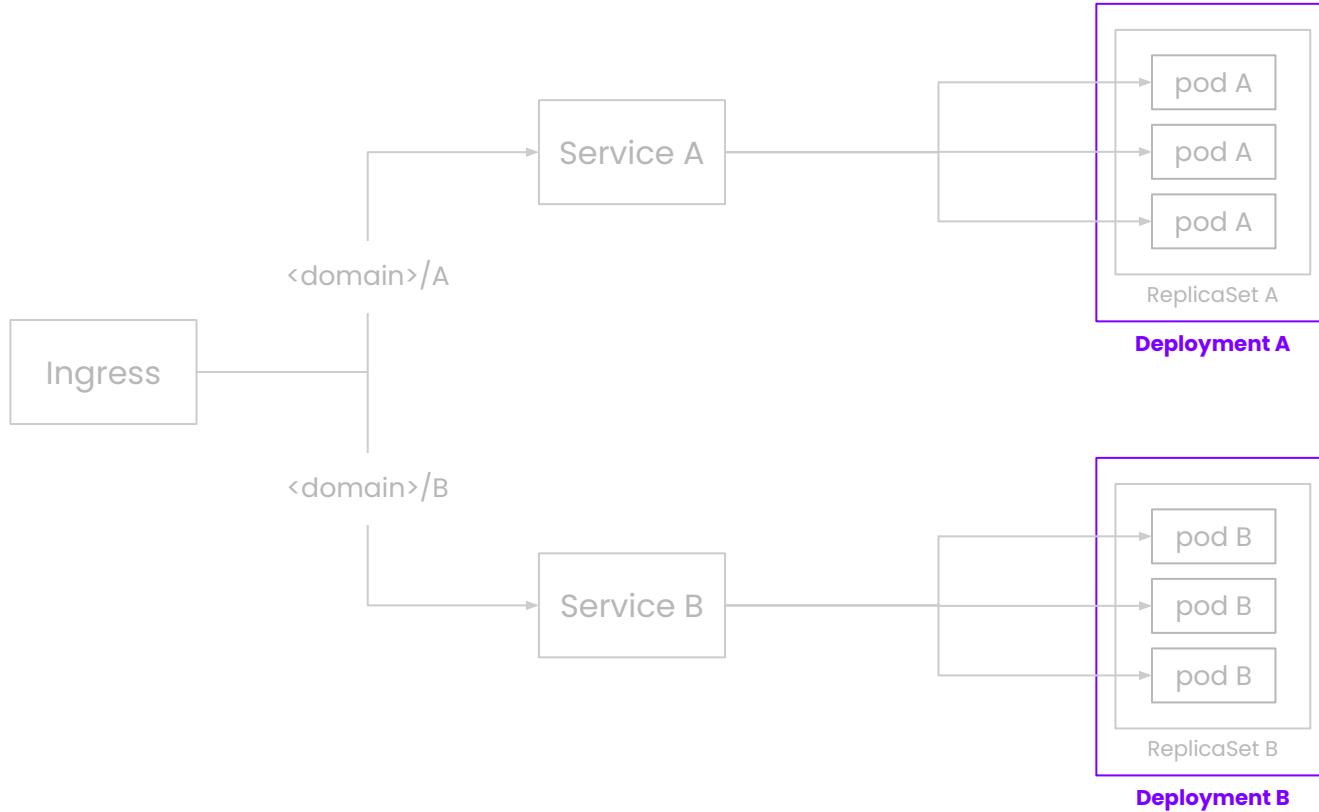
เจ้าของลิขสิทธิ์ อนุญาติให้ใช้สิ่งที่มีการเรียบเรียงอุปกรณ์ท่านนั้น และขอสงวนลิขสิทธิ์ ห้ามมิให้เผยแพร่ในที่สาธารณะ ผู้ละเอียด จะถูกดำเนินคดีตามกฎหมาย

**Jumpbox®**

# Deployment

- Working with Kubernetes -

# Kubernetes in Action



# A Pod Template in Deployment

## Deployment Structure with **Declarative** file (Cont.)

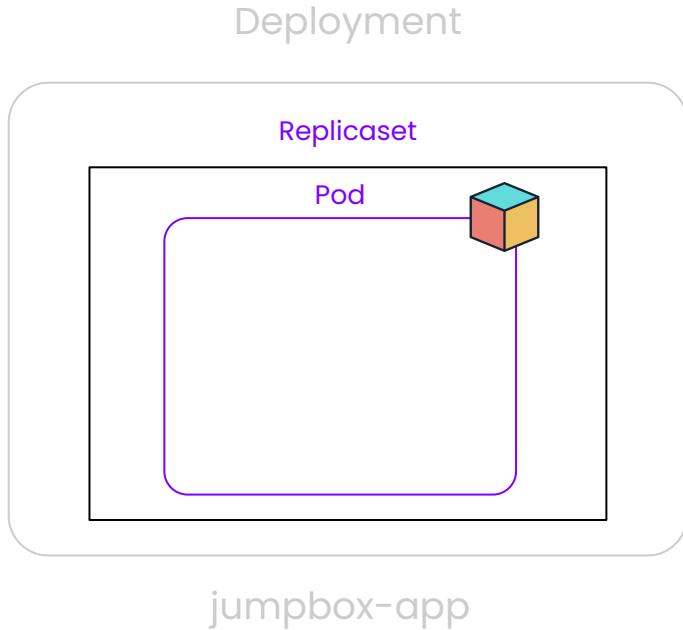
# Deployment Structure with **Declarative** file (Cont.)

Deployment

jumpbox-app

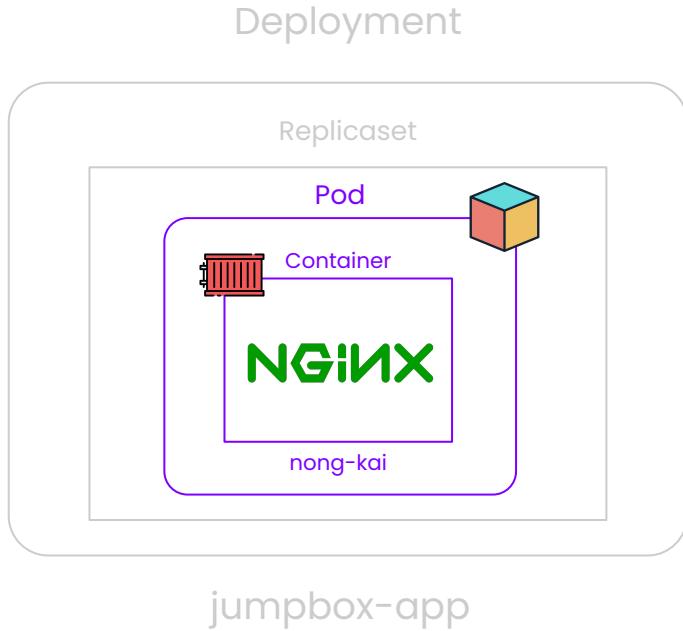
```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: jumpbox-app
```

# Deployment Structure with Declarative file (Cont.)



```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: jumpbox-app
spec:
  replicas: 1
```

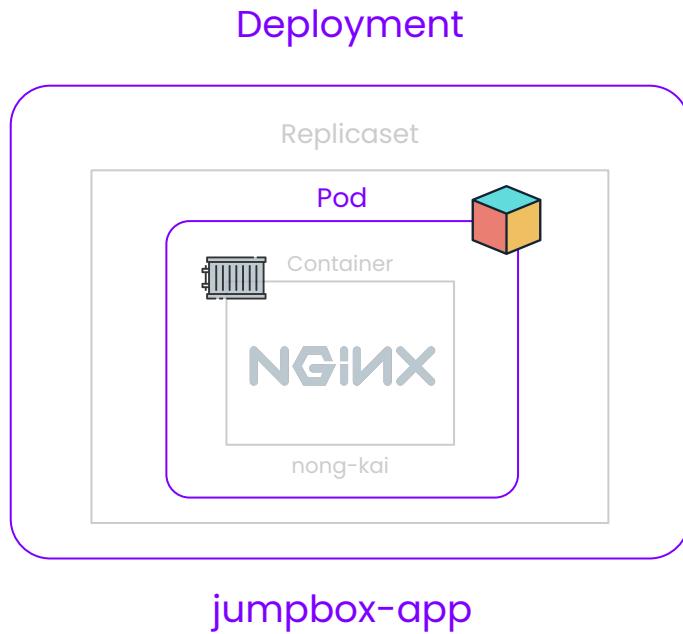
# Deployment Structure with Declarative file (Cont.)



```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: jumpbox-app
spec:
  replicas: 1
  strategy: {}
  selector:
    matchLabels:
      app: jumpbox
  template:
    metadata:
      labels:
        app: jumpbox
    spec:
      containers:
        - image: nginx
          name: nong-kai
```

Pod.spec (pod template)

# Deployment Structure with Declarative file (Cont.)

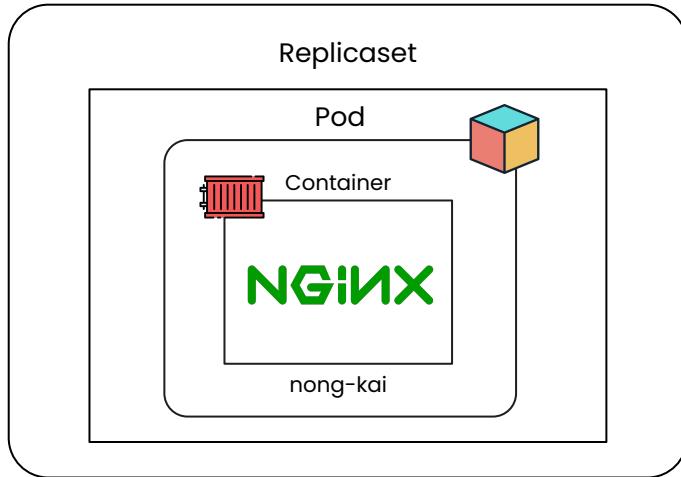


```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: jumpbox-app
spec:
  replicas: 1
  strategy: {}
  selector:
    matchLabels:
      app: jumpbox
  template:
    metadata:
      labels:
        app: jumpbox
    spec:
      containers:
        - image: nginx
          name: nong-kai
```

Deployment select Pods by label

# Deployment Structure with Declarative file (Cont.)

## Deployment



jumpbox-app

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: jumpbox-app
spec:
  replicas: 1
  strategy: {}
  selector:
    matchLabels:
      app: jumpbox
  template:
    metadata:
      labels:
        app: jumpbox
    spec:
      containers:
        - image: nginx
          name: nong-kai
```

Deployment select Pods by label

Pods definition

# Deployment

## With Real Action Simulator

[Click Here](#)



เจ้าของลิขสิทธิ์ อนุญาติให้ใช้ส่วนหนึ่งของการเรียนรู้ด้านบุคคลท่านนี้ และขอสงวนลิขสิทธิ์ หัวมีไฟเบอร์ในที่สาธารณะ ผู้อื่นเมต จะถูกดำเนินคดีตามกฎหมาย

Jumpbox®

# Deployment

Strategy for deployment process

# Deployment

Strategy for deployment process

- RollingUpdate
- Recreate

# Rolling strategy

# Deployment - Strategy (Cont.)

Strategy: **rollingUpdate**

Deployment

# Deployment - Strategy (Cont.)

Strategy: **rollingUpdate**

- maxSurge 50%

Deployment

# Deployment - Strategy (Cont.)

Strategy: **rollingUpdate**

- maxSurge 50%
- maxUnavailable: 0%

Deployment

# Deployment Strategy

# Deployment Strategy

## 1. Rolling Update

# Deployment Strategy

## 1. RollingUpdate

### a. maxSurge

```
...  
spec:  
  strategy:  
    rollingUpdate:  
      maxSurge: 25%
```

# Deployment Strategy

## 1. RollingUpdate

### a. maxSurge

```
...  
spec:  
  strategy:  
    rollingUpdate:  
      maxSurge: 25%
```

- It specifies the maximum number of Pods that can be created over the desired number of Pods.

# Deployment Strategy

## 1. RollingUpdate

### a. maxSurge

```
...  
spec:  
  strategy:  
    rollingUpdate:  
      maxSurge: 25%
```

- It specifies the maximum number of Pods that can be created over the desired number of Pods.
- The value can be an absolute number or percentage ( $\geq 0$ ,  $\geq 0\%$ )

# Deployment Strategy

## 1. RollingUpdate

### a. maxSurge

```
...  
spec:  
  strategy:  
    rollingUpdate:  
      maxSurge: 25%
```

- It specifies the maximum number of Pods that can be created over the desired number of Pods.
- The value can be an absolute number or percentage ( $\geq 0, \geq 0\%$ )
- The value **cannot be 0 if MaxUnavailable is 0**

# Deployment Strategy

## 1. RollingUpdate

### a. maxSurge

```
...  
spec:  
  strategy:  
    rollingUpdate:  
      maxSurge: 25%
```

- It specifies the maximum number of Pods that can be created over the desired number of Pods.
- The value can be an absolute number or percentage ( $\geq 0$ ,  $\geq 0\%$ )
- The value cannot be 0 if MaxUnavailable is 0
- The absolute number is calculated from the percentage by **rounding up**.

# Deployment Strategy

## 1. RollingUpdate

### a. maxSurge

```
...  
spec:  
  strategy:  
    rollingUpdate:  
      maxSurge: 25%
```

- It specifies the maximum number of Pods that can be created over the desired number of Pods.
- The value can be an absolute number or percentage ( $\geq 0, \geq 0\%$ )
- The value cannot be 0 if maxUnavailable is 0
- The absolute number is calculated from the percentage by rounding up.
- The default value is 25%.

# Deployment Strategy

## 1. RollingUpdate

### a. maxSurge

```
...  
spec:  
  strategy:  
    rollingUpdate:  
      maxSurge: 25%
```

- It specifies the maximum number of Pods that can be created over the desired number of Pods.
- The value can be an absolute number or percentage ( $\geq 0, \geq 0\%$ )
- The value cannot be 0 if MaxUnavailable is 0
- The absolute number is calculated from the percentage by rounding up.
- The **default value** is 25%.

# Deployment Strategy

## 1. RollingUpdate

- a. maxSurge
- b. maxUnavailable

```
...  
spec:  
  strategy:  
    rollingUpdate:  
      maxSurge: 25%  
      maxUnavailable: 25%
```

# Deployment Strategy

## 1. RollingUpdate

### a. maxSurge

### b. maxUnavailable

- It specifies the maximum number of Pods that can be **unavailable** during the update process.

```
...  
spec:  
  strategy:  
    rollingUpdate:  
      maxSurge: 25%  
      maxUnavailable: 25%
```

# Deployment Strategy

## 1. RollingUpdate

### a. maxSurge

### b. maxUnavailable

- It specifies the maximum number of Pods that can be unavailable during the update process.
- The value can be an absolute number or percentage ( $\geq 0$ ,  $\geq 0\%$ )

```
...  
spec:  
  strategy:  
    rollingUpdate:  
      maxSurge: 25%  
      maxUnavailable: 25%
```

# Deployment Strategy

## 1. RollingUpdate

### a. maxSurge

### b. maxUnavailable

```
...  
spec:  
  strategy:  
    rollingUpdate:  
      maxSurge: 25%  
      maxUnavailable: 25%
```

- It specifies the maximum number of Pods that can be unavailable during the update process.
- The value can be an absolute number or percentage ( $\geq 0, \geq 0\%$ )
- The value **cannot be 0 if maxSurge is 0**

# Deployment Strategy

## 1. RollingUpdate

### a. maxSurge

### b. maxUnavailable

```
...  
spec:  
  strategy:  
    rollingUpdate:  
      maxSurge: 25%  
      maxUnavailable: 25%
```

- It specifies the maximum number of Pods that can be unavailable during the update process.
- The value can be an absolute number or percentage ( $\geq 0, \geq 0\%$ )
- The value cannot be 0 if maxSurge is 0
- The absolute number is calculated from the percentage by **rounding down**.

# Deployment Strategy

## 1. RollingUpdate

### a. maxSurge

### b. maxUnavailable

```
...  
spec:  
  strategy:  
    rollingUpdate:  
      maxSurge: 25%  
      maxUnavailable: 25%
```

- It specifies the maximum number of Pods that can be unavailable during the update process.
- The value can be an absolute number or percentage ( $\geq 0, \geq 0\%$ )
- The value cannot be 0 if maxSurge is 0
- The absolute number is calculated from the percentage by rounding down.
- The **default value** is 25%.

# Deployment Strategy

## 1. RollingUpdate

### a. maxSurge

### b. maxUnavailable

```
...  
spec:  
  strategy:  
    rollingUpdate:  
      maxSurge: 25%  
      maxUnavailable: 25%
```

- It specifies the maximum number of Pods that can be **unavailable** during the update process.
- The value can be an **absolute number or percentage ( $\geq 0, \geq 0\%$ )**
- The value **cannot be 0 if maxSurge is 0**
- The absolute number is calculated from the percentage by **rounding down**.
- The **default value** is 25%.

# Example

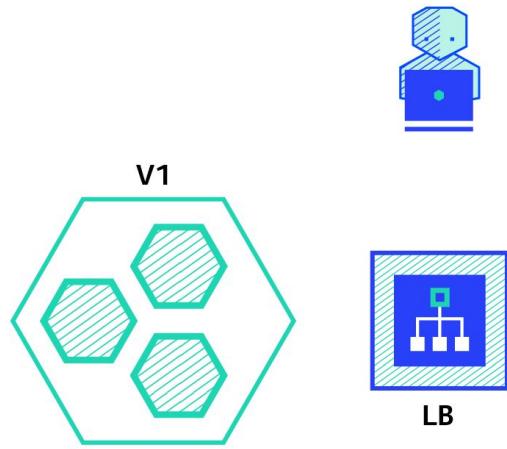
Deployment **RollingUpdate** Strategy

# Example

## Deployment RollingUpdate Strategy

- maxSurge: 50%
- maxUnavailable: 0%

# Deployment



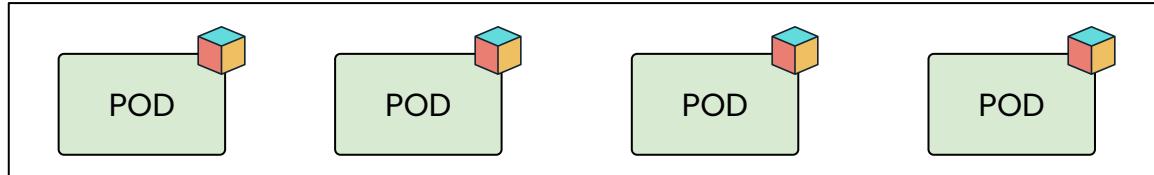
Credit: <https://blog.container-solutions.com/deployment-strategies>

ເຫັນອອລີສີທີ່ ອຸນຍາດໃຫ້ຮັບຄຸນໆທີ່ກ່ຽວຂ້ອງການເປັນນູ້ຮັວນບຸກຄະຫຼາກທ່ານັ້ນ ແລະຂອງສຽນລີສີທີ່ ທັນມີໄຟເຫັນພົນໍາທີ່ສ່າງຮານະ ຜູ້ອະນິດ ຈະຖຸກດຳເນີນຄືຕາມງູ່ໝາຍ

**Jumpbox®**

## Example: RollingUpdate (Cont.)

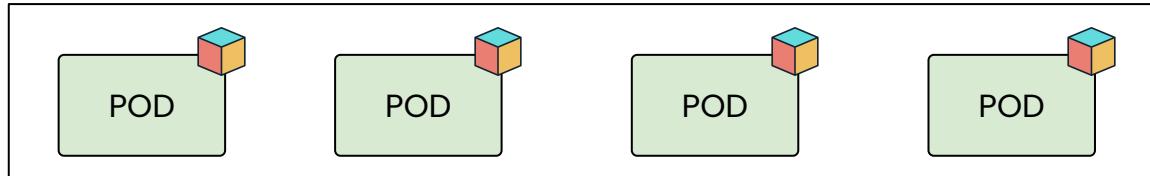
maxSurge: 50%  
maxUnavailable: 0%



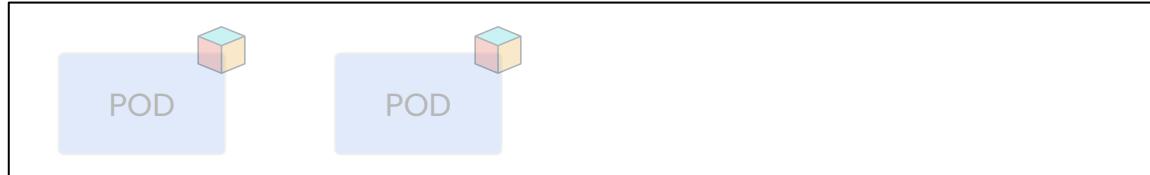
Deployment

## Example: RollingUpdate (Cont.)

maxSurge: 50%  
maxUnavailable: 0%



ReplicaSet: v1

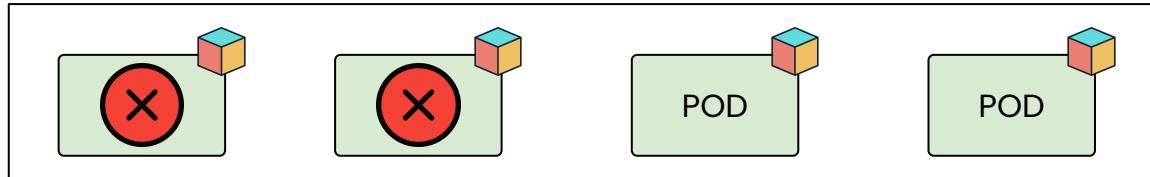


ReplicaSet: v2

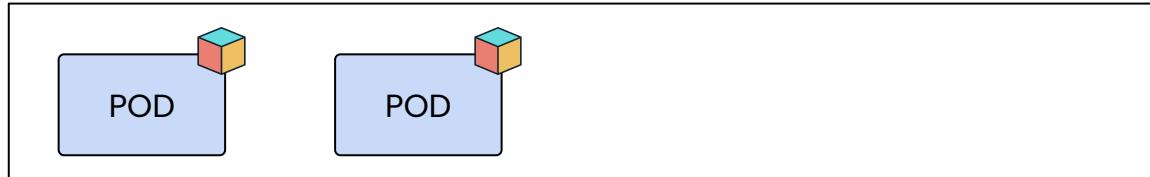
Deployment

## Example: RollingUpdate (Cont.)

maxSurge: 50%  
maxUnavailable: 0%



ReplicaSet: v1

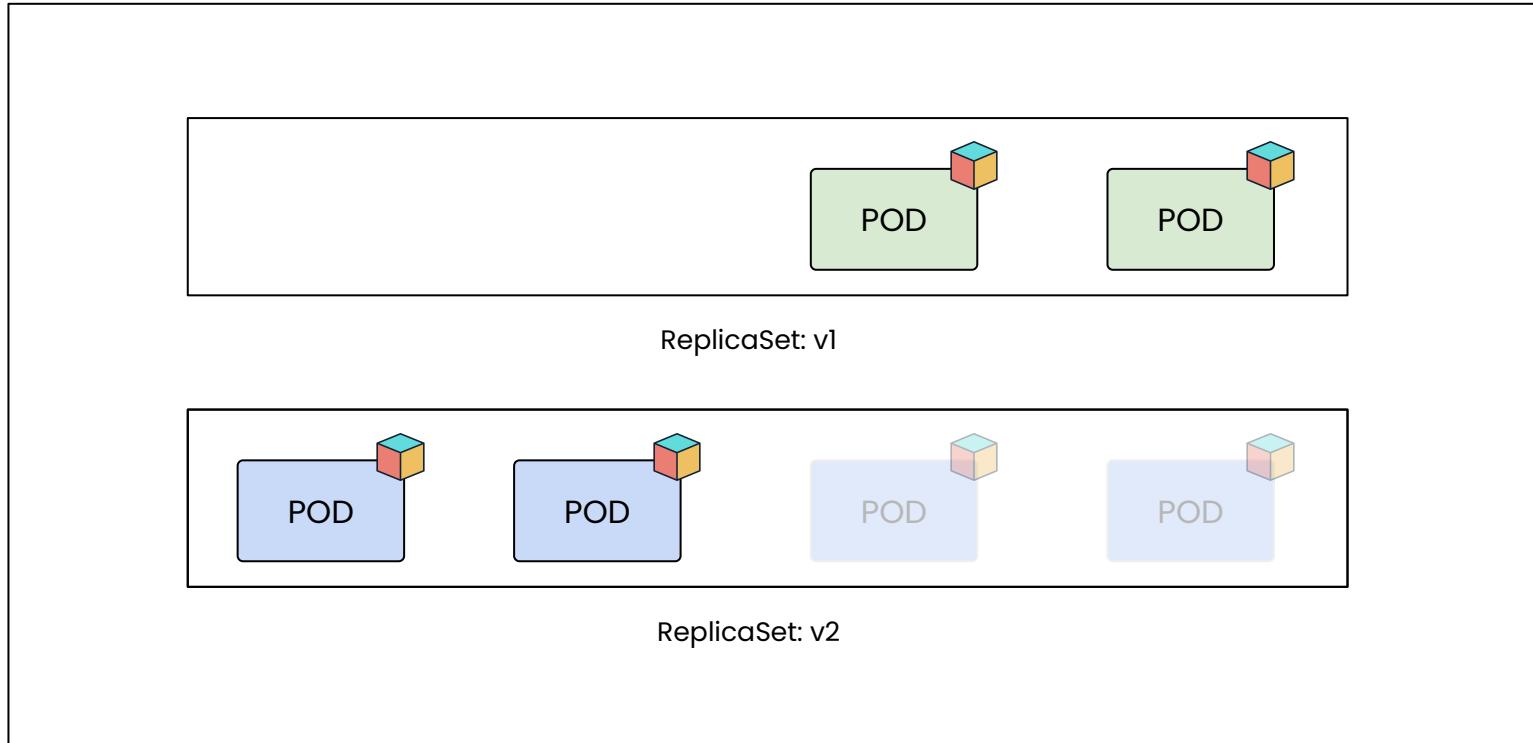


ReplicaSet: v2

Deployment

## Example: RollingUpdate (Cont.)

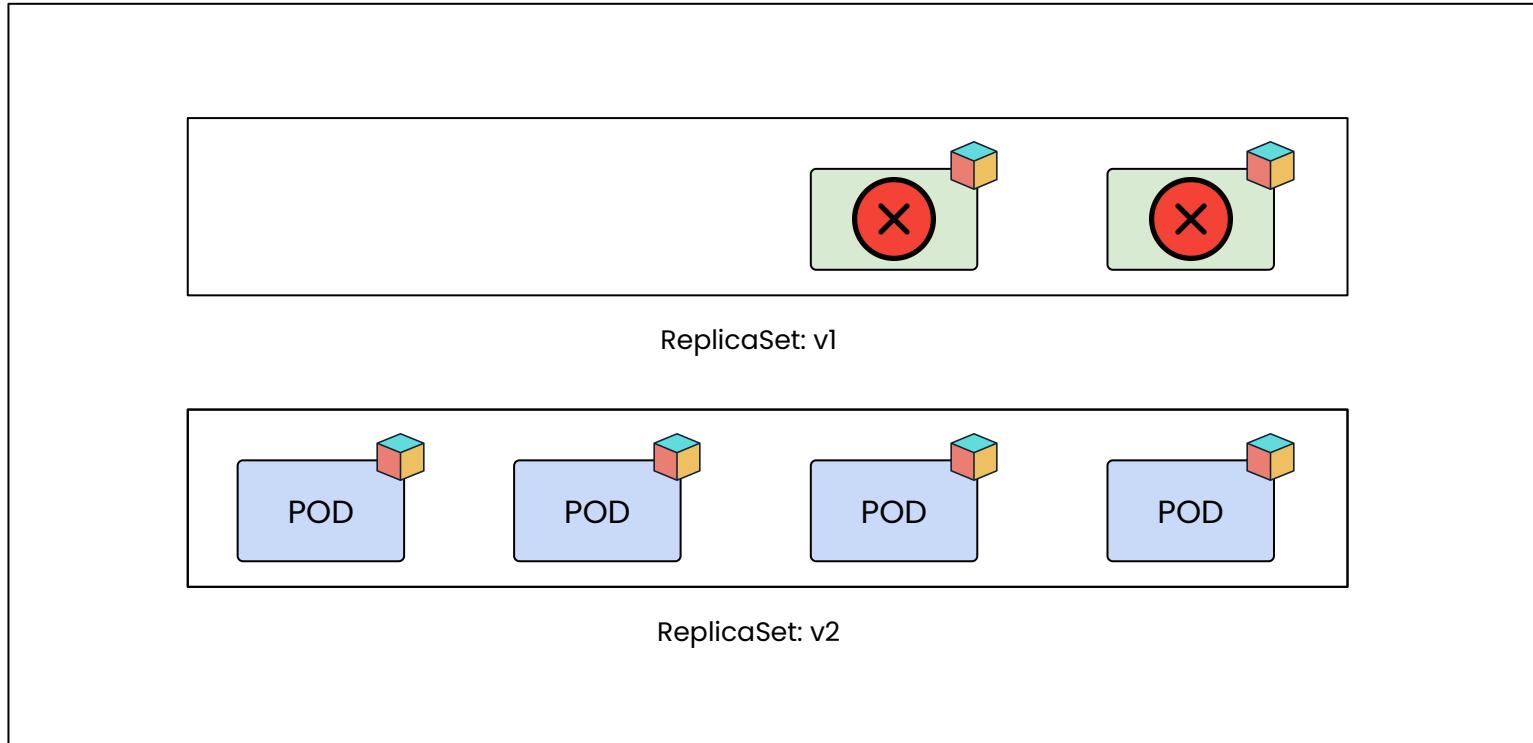
maxSurge: 50%  
maxUnavailable: 0%



Deployment

## Example: RollingUpdate (Cont.)

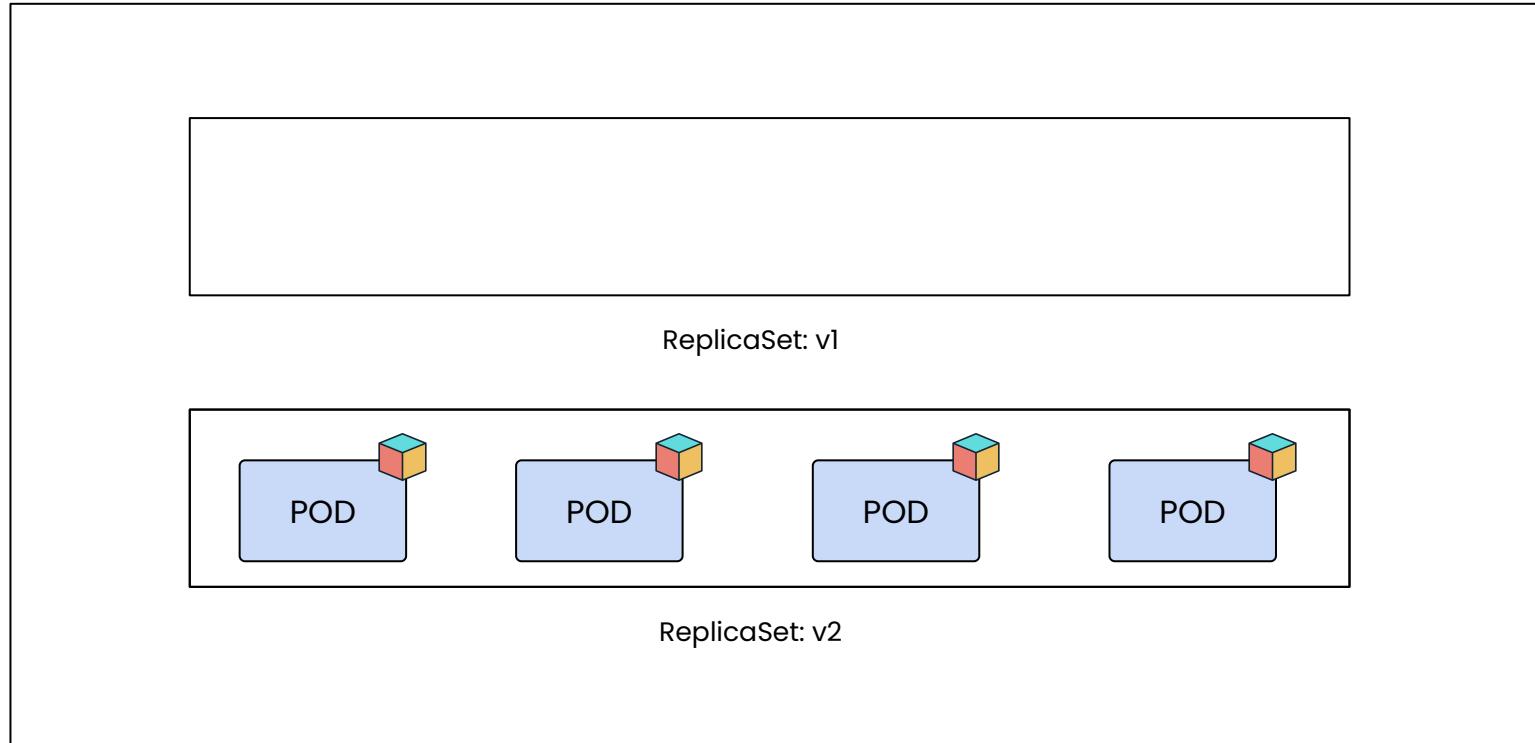
maxSurge: 50%  
maxUnavailable: 0%



Deployment

## Example: RollingUpdate (Cont.)

maxSurge: 50%  
maxUnavailable: 0%

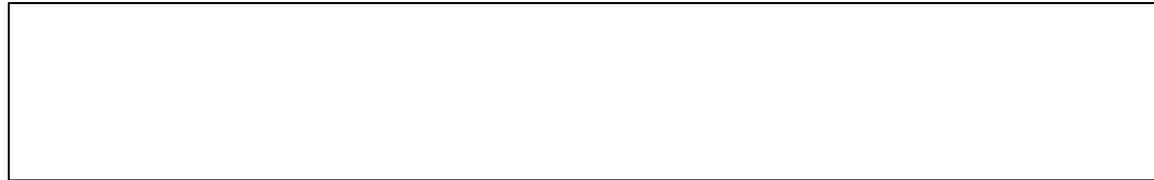


Deployment

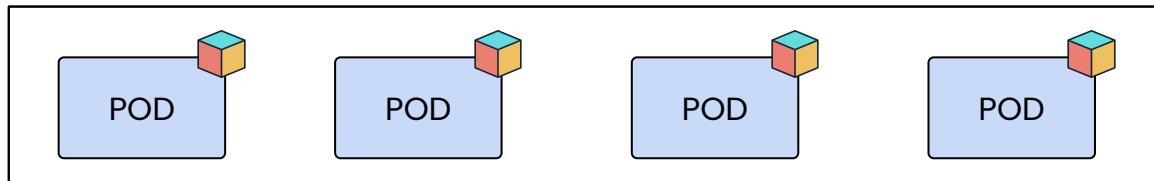
## Example: RollingUpdate (Cont.)

maxSurge: 50%  
maxUnavailable: 0%

revisionHistory: 1



ReplicaSet: v1



ReplicaSet: v2

Deployment

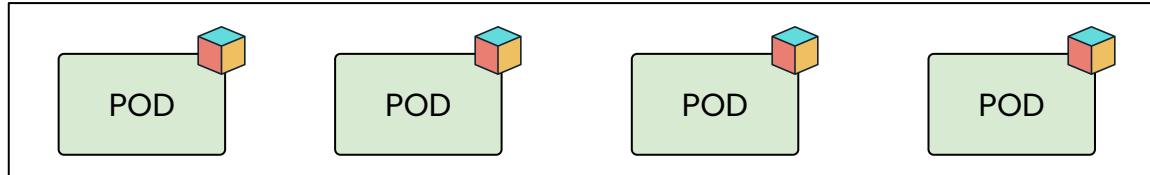
# Example 2

## Deployment RollingUpdate Strategy

- maxSurge: 25%
- maxUnavailable: 25%

## Example 2: RollingUpdate (Cont.)

maxSurge: 25%  
maxUnavailable: 25%

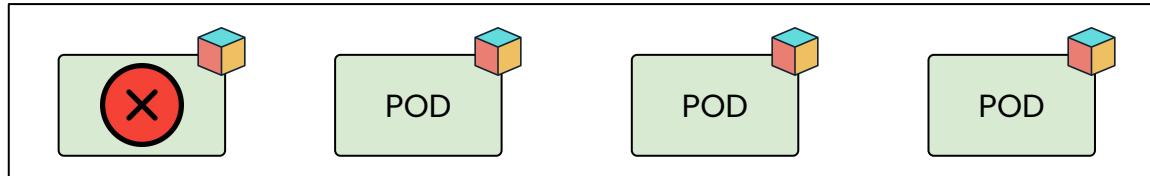


ReplicaSet: v1

Deployment

## Example 2: RollingUpdate (Cont.)

maxSurge: 25%  
maxUnavailable: 25%



ReplicaSet: v1

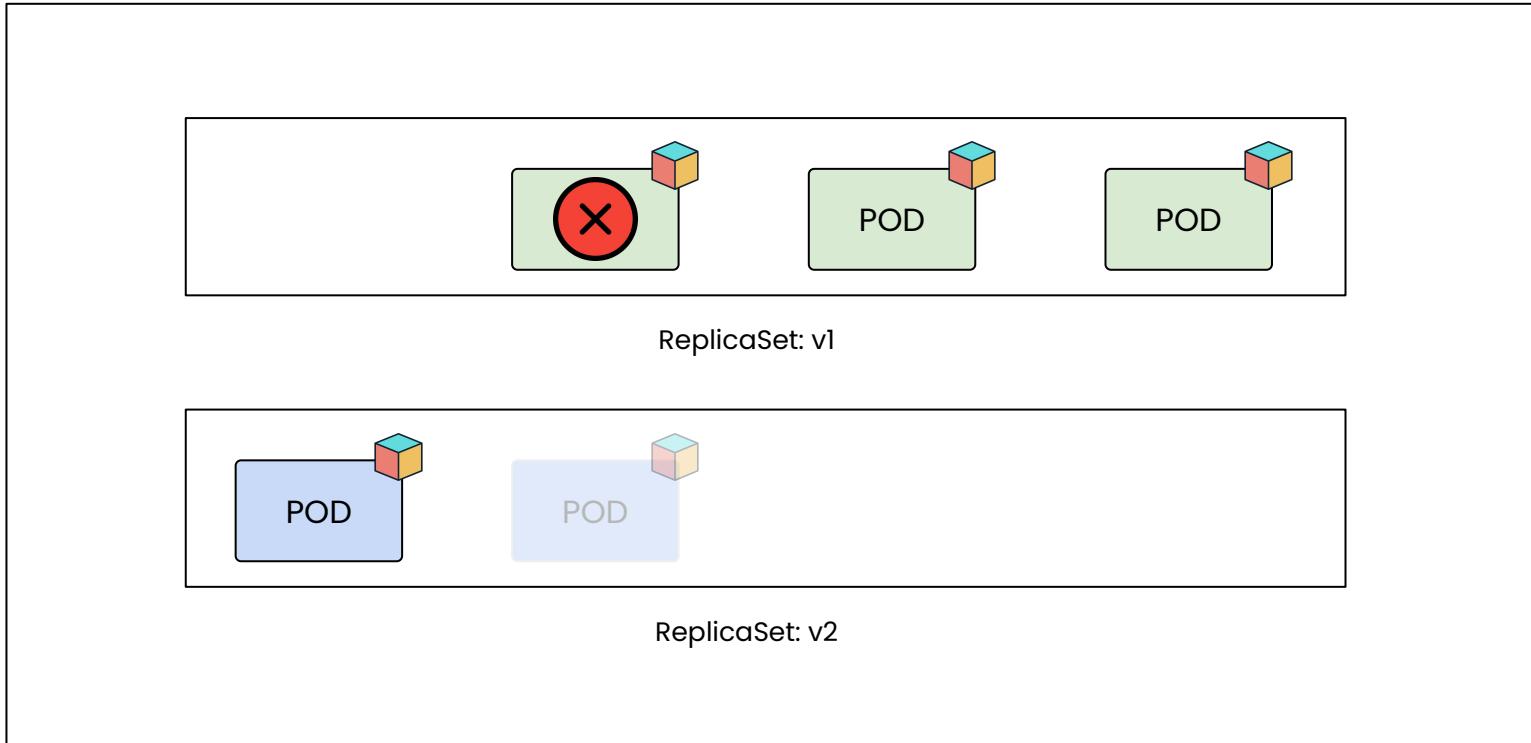


ReplicaSet: v2

Deployment

## Example 2: RollingUpdate (Cont.)

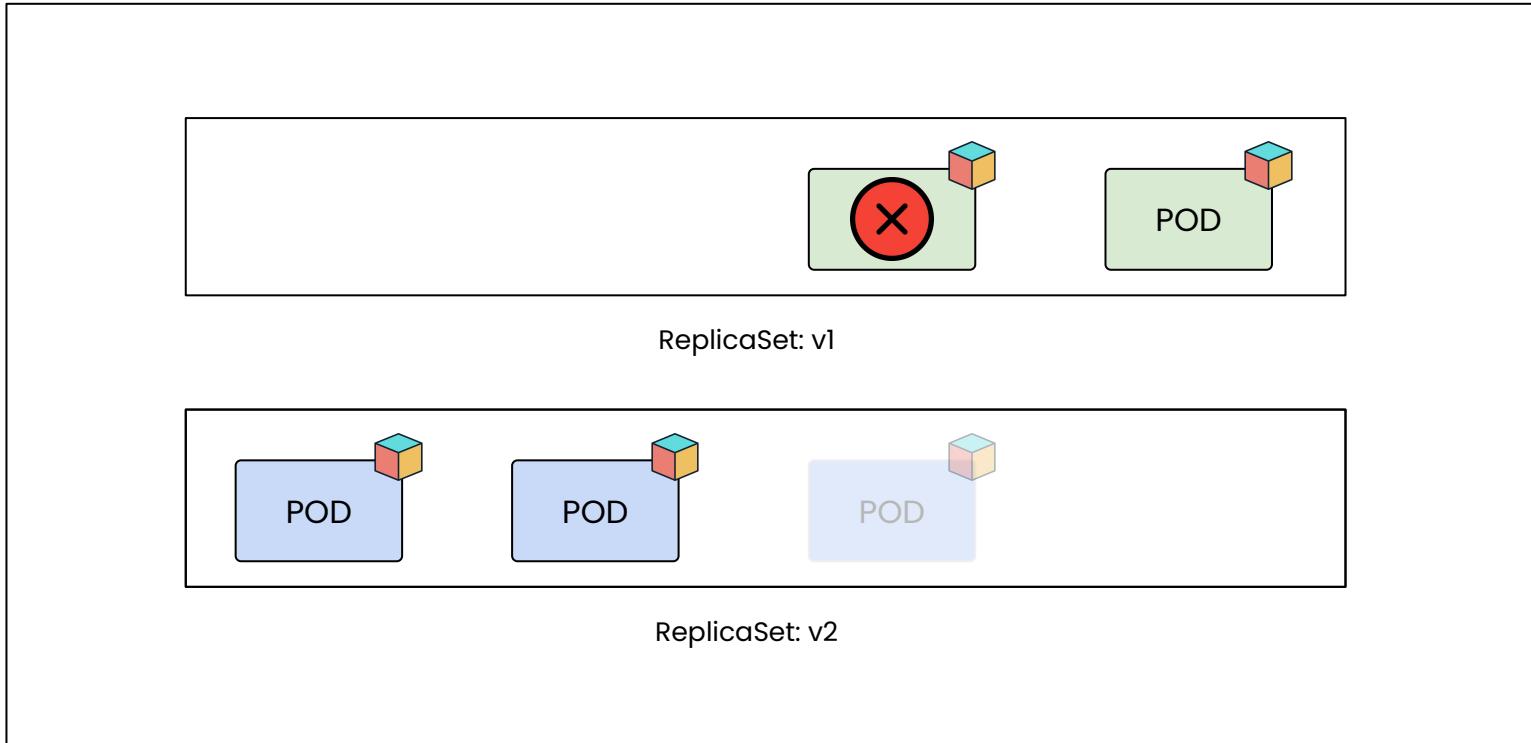
maxSurge: 25%  
maxUnavailable: 25%



Deployment

## Example 2: RollingUpdate (Cont.)

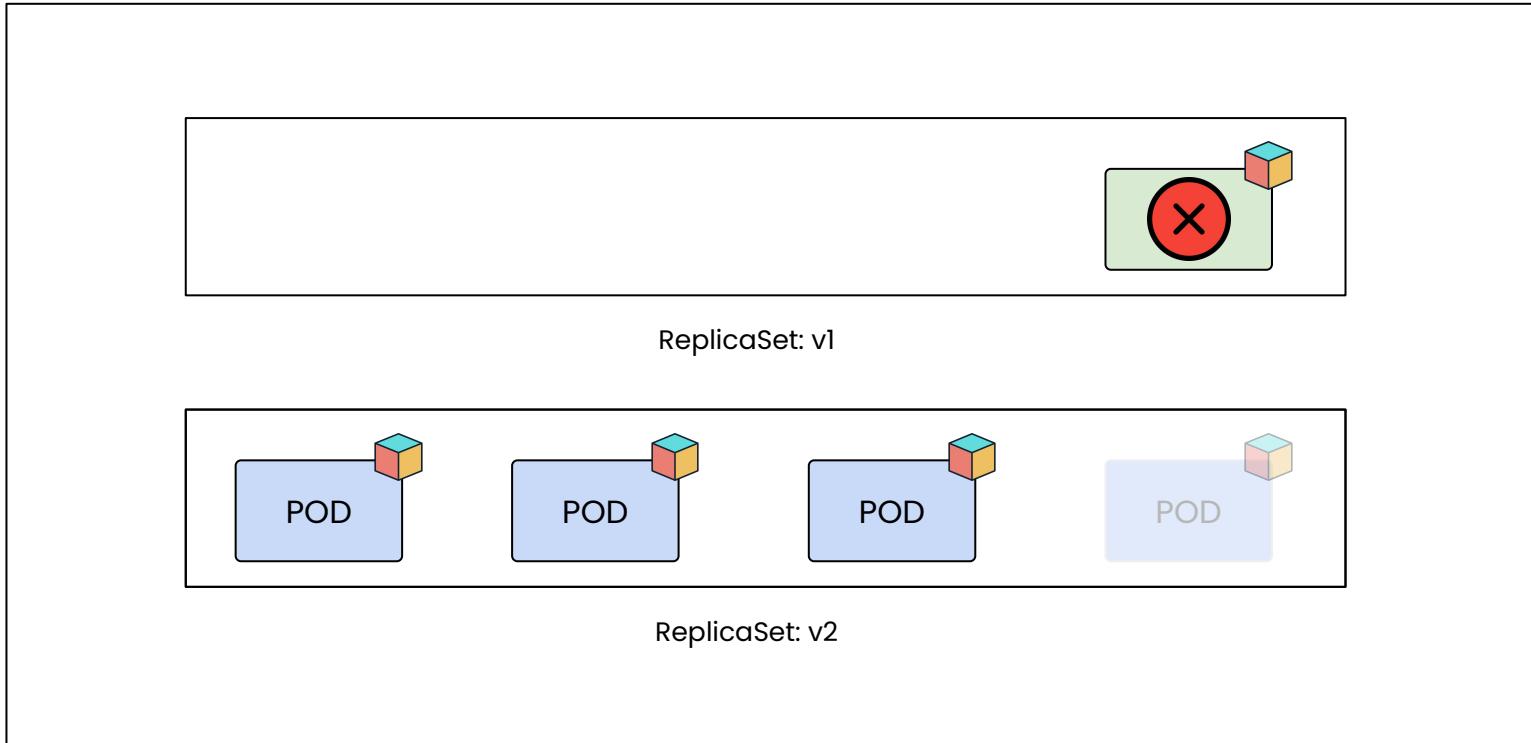
maxSurge: 25%  
maxUnavailable: 25%



Deployment

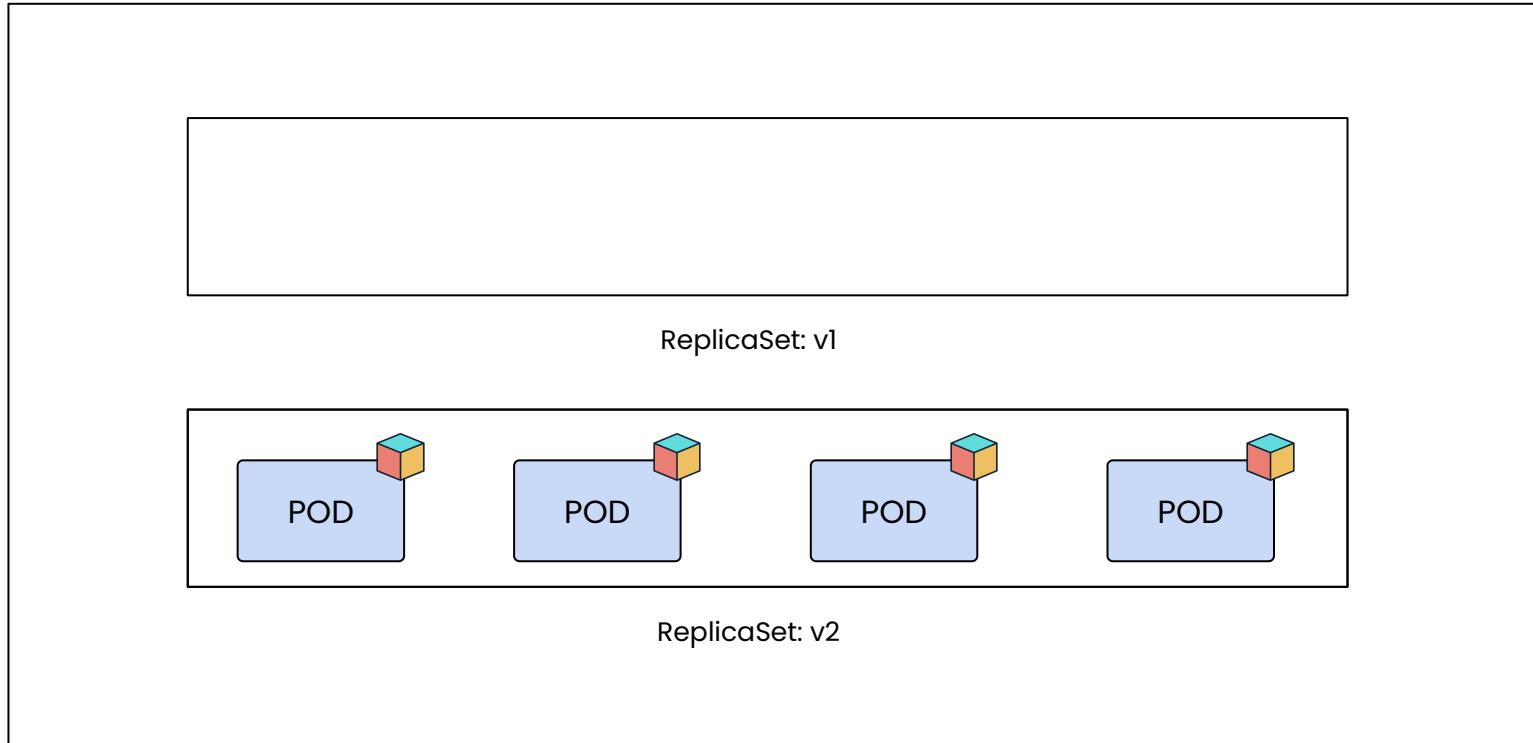
## Example 2: RollingUpdate (Cont.)

maxSurge: 25%  
maxUnavailable: 25%



## Example 2: RollingUpdate (Cont.)

maxSurge: 25%  
maxUnavailable: 25%



Deployment

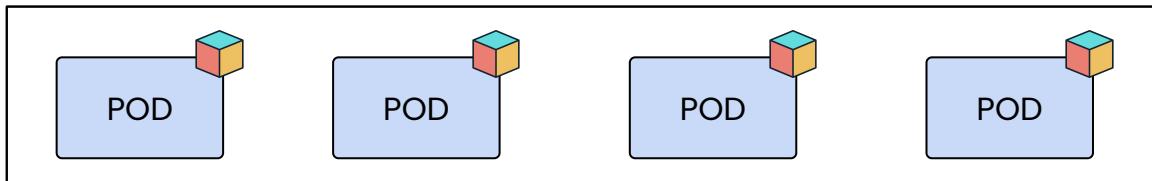
## Example 2: RollingUpdate (Cont.)

maxSurge: 25%  
maxUnavailable: 25%

revisionHistory: 1



ReplicaSet: v1



ReplicaSet: v2

Deployment

# Deployment Strategy

- 1. RollingUpdate**
  - a. maxSurge**
  - b. maxUnavailable**
- 2. Recreate**

```
...  
spec:  
  strategy:  
    type: Recreate
```

# Deployment Strategy

## 1. RollingUpdate

a. maxSurge

b. maxUnavailable

## 2. Recreate

- This will only guarantee Pod termination previous to creation for upgrades.

```
...  
spec:  
  strategy:  
    type: Recreate
```

# Deployment Strategy

## 1. RollingUpdate

### a. maxSurge

### b. maxUnavailable

- This will only guarantee Pod termination previous to creation for upgrades.
- All Pods of the old revision will be terminated immediately when you upgrade new version.

## 2. Recreate

```
...  
spec:  
  strategy:  
    type: Recreate
```

# Deployment Strategy

## 1. RollingUpdate

### a. maxSurge

### b. maxUnavailable

## 2. Recreate

- This will only guarantee Pod termination previous to creation for upgrades.
- All Pods of the old revision will be terminated immediately when you upgrade new version.
- Successful removal is awaited before any Pod of the new revision is created.

```
...  
spec:  
  strategy:  
    type: Recreate
```

# Deployment Strategy

## 1. RollingUpdate

### a. maxSurge

### b. maxUnavailable

## 2. Recreate

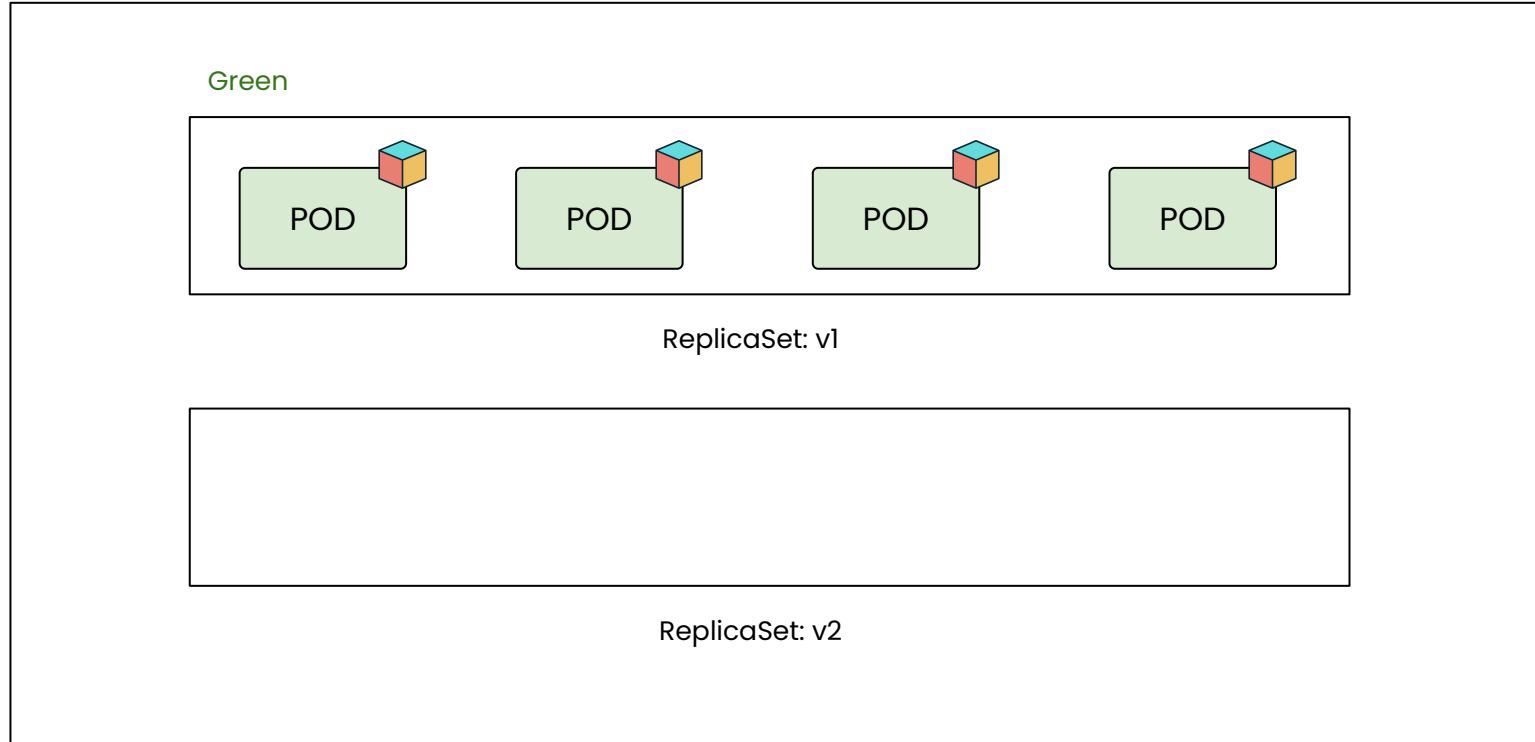
- This will only guarantee Pod termination previous to creation for upgrades.
- All Pods of the old revision will be terminated immediately when you upgrade new version.
- Successful removal is awaited before any Pod of the new revision is created.

```
...  
spec:  
  strategy:  
    type: Recreate
```

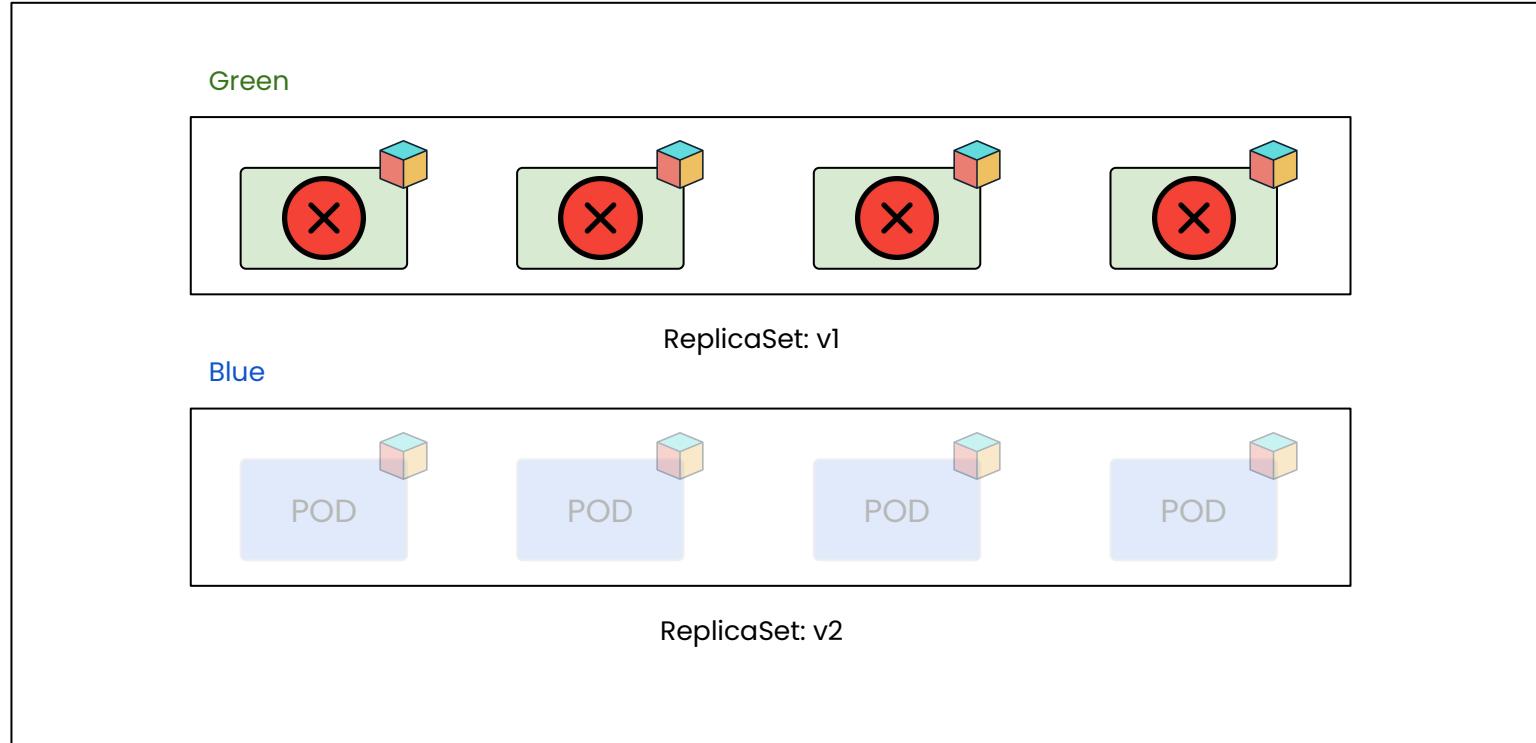
# Example

Deployment **Recreate** Strategy

## Example: Recreate (Cont.)

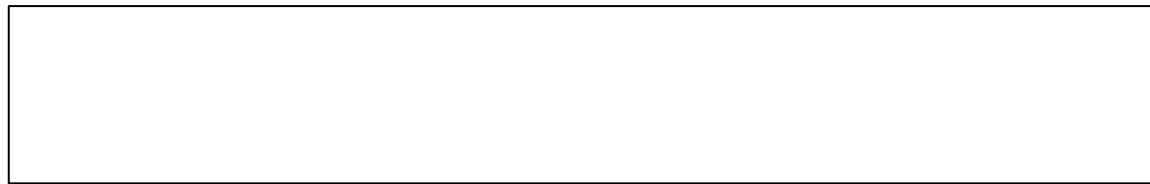


## Example: Recreate (Cont.)

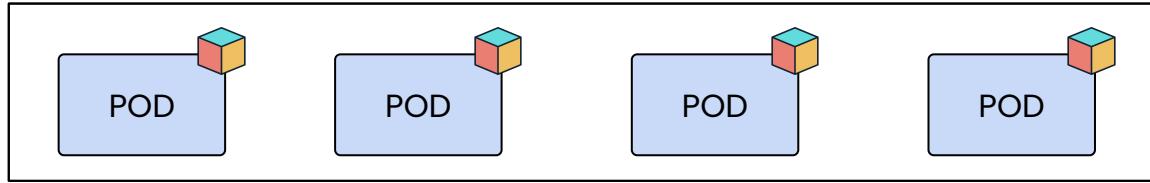


Deployment

## Example: Recreate (Cont.)



ReplicaSet: v1



ReplicaSet: v2

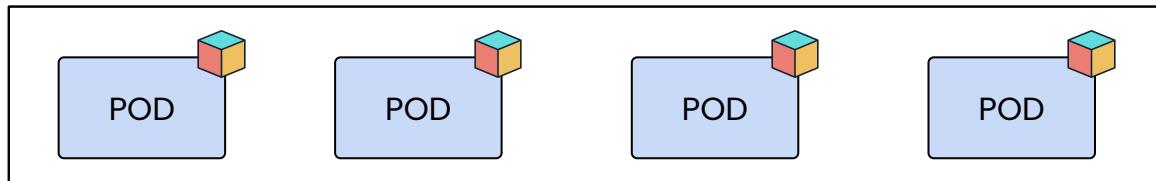
Deployment

## Example: Recreate (Cont.)

revisionHistory: 1



ReplicaSet: v1



ReplicaSet: v2

Deployment

# Kubernetes in Action

1. Pods
2. Deployment
  - a. Strategy
  - b. revisionHistory

# Kubernetes in Action

## 1. Pods

## 2. Deployment

### a. Strategy

### b. revisionHistory

- It field that specifies the number of old ReplicaSets to retain to allow rollback.

```
...  
spec:  
  revisionHistoryLimit: 10  
strategy:  
  ...
```

# Kubernetes in Action

## 1. Pods

## 2. Deployment

### a. Strategy

### b. revisionHistory

- It field that specifies the number of old ReplicaSets to retain to allow rollback.
- These old ReplicaSets **consume resources** in etcd and crowd the output of `$ kubectl get ns`

```
...  
spec:  
  revisionHistoryLimit: 10  
strategy:  
  ...
```

# Kubernetes in Action

## 1. Pods

## 2. Deployment

### a. Strategy

### b. revisionHistory

- It field that specifies the number of old ReplicaSets to retain to allow rollback.
- These old ReplicaSets consume resources in etcd and crowd the output of `$ kubectl get rs`
- The configuration of each [Deployment revision is stored](#) in its ReplicaSets

```
...  
spec:  
  revisionHistoryLimit: 10  
strategy:  
  ...
```

# Kubernetes in Action

## 1. Pods

## 2. Deployment

### a. Strategy

### b. revisionHistory

- It field that specifies the number of old ReplicaSets to retain to allow rollback.
- These old ReplicaSets consume resources in etcd and crowd the output of `$ kubectl get rs`
- The configuration of each Deployment revision is stored in its ReplicaSets
- once an old ReplicaSet is deleted, you lose the ability to rollback to that revision of Deployment

```
...  
spec:  
  revisionHistoryLimit: 10  
strategy:  
  ...
```

# Kubernetes in Action

## 1. Pods

## 2. Deployment

### a. Strategy

### b. revisionHistory

- It field that specifies the number of old ReplicaSets to retain to allow rollback.
- These old ReplicaSets consume resources in etcd and crowd the output of `$ kubectl get rs`
- The configuration of each Deployment revision is stored in its ReplicaSets
- once an old ReplicaSet is deleted, you lose the ability to rollback to that revision of Deployment
- By default, 10 old ReplicaSets will be kept.

```
...  
spec:  
  revisionHistoryLimit: 10  
strategy:  
  ...
```

# Kubernetes in Action

## 1. Pods

## 2. Deployment

### a. Strategy

### b. revisionHistory

- It field that specifies the number of old ReplicaSets to retain to allow rollback.
- These old ReplicaSets consume resources in etcd and crowd the output of `$ kubectl get ns`
- The configuration of each Deployment revision is stored in its ReplicaSets
- once an old ReplicaSet is deleted, you lose the ability to rollback to that revision of Deployment
- By default, 10 old ReplicaSets will be kept.

```
...  
spec:  
  revisionHistoryLimit: 10  
strategy:  
  ...
```

# Deployment Strategy

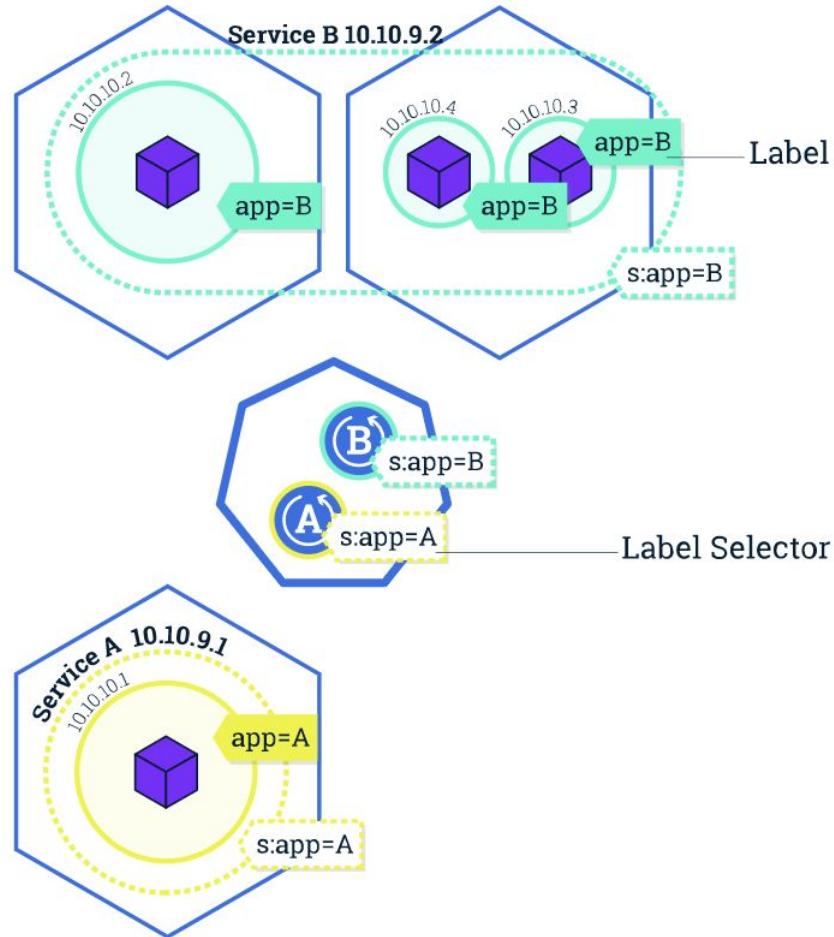
With Real Action Simulator

[Click Here](#)



# Service

- Working with Kubernetes -

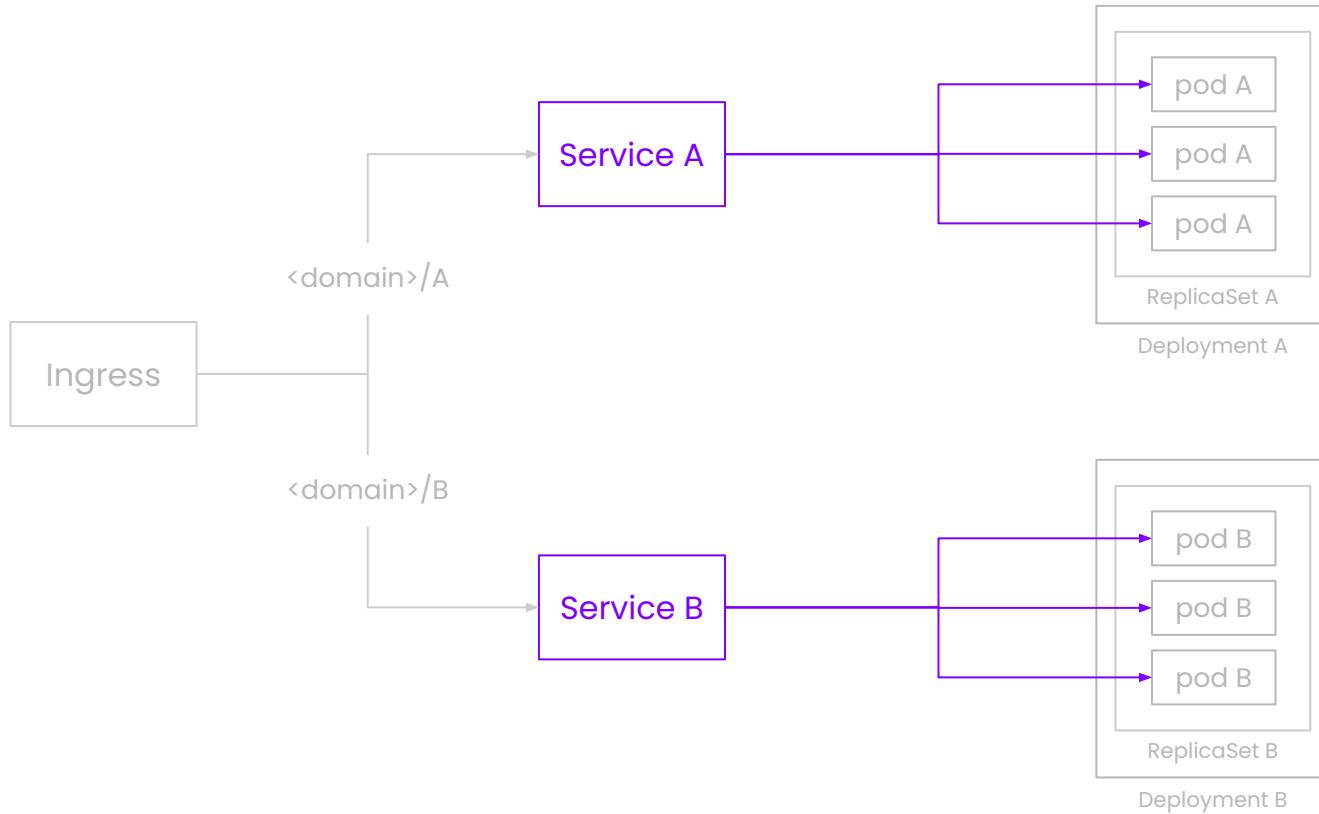


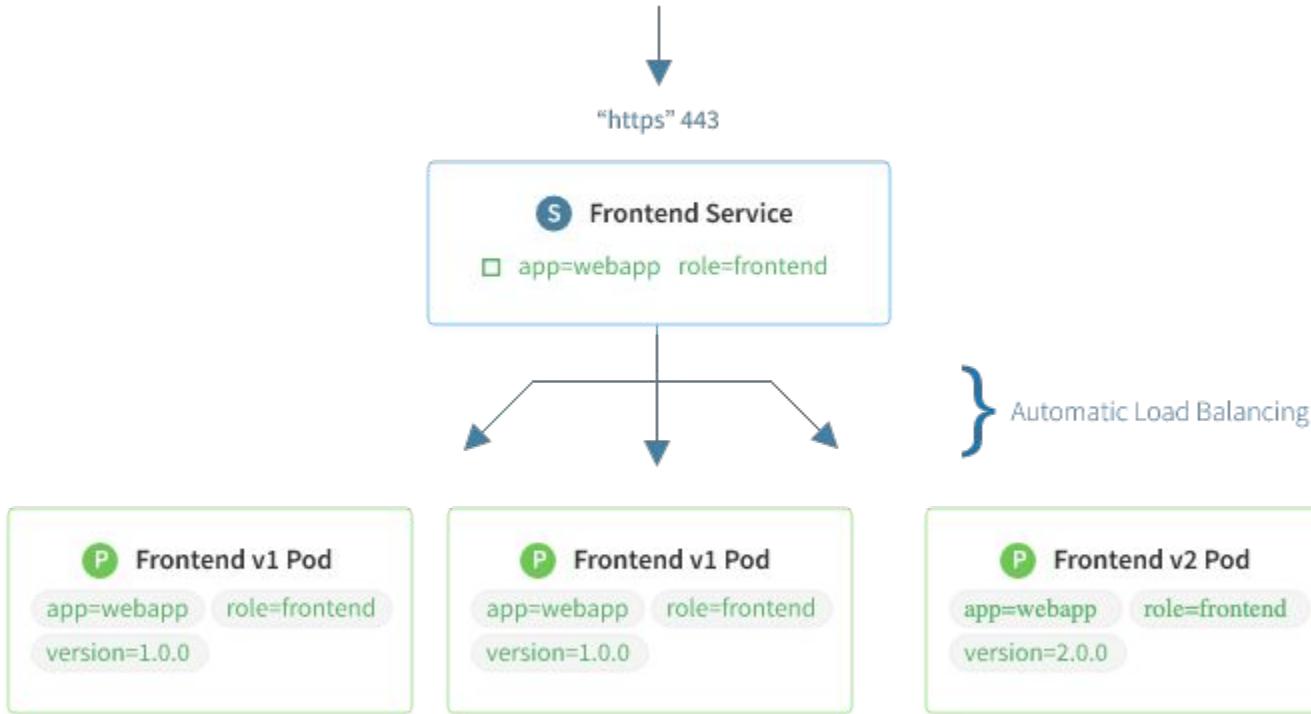
เจ้าของลิสติ๊ก อุบลฯได้ใช้วัสดุนี้เพื่อการเรียนรู้ส่วนบุคคลเท่านั้น และขอสงวนลิขสิทธิ์ ห้ามมิให้เผยแพร่ในที่สาธารณะ ผู้อื่นเดินทาง จดถูกคัดเป็นคดีตามกฎหมาย

# What is service?

- The idea of a Service is to group a set of Pod endpoints into a single resource.
- You can configure various ways to access the grouping.
- By default, you get a stable cluster IP address that clients inside the cluster can use to contact Pods in the Service

# Kubernetes Service





#### Reference Pictures:

- [Kubernetes – Basics](#)

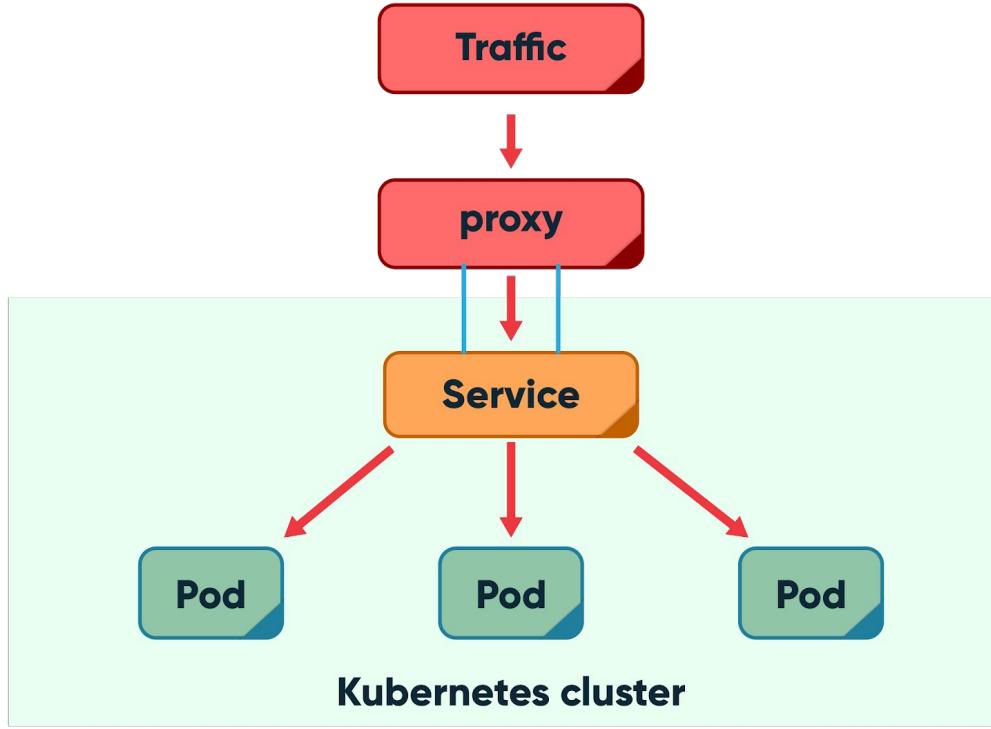
# Service

- Publishing Services (ServiceTypes)
- Headless Services

# ServiceTypes

- ClusterIP
  - Exposes the Service on a cluster-internal IP.
  - Default ServiceType
- NodePort
  - Exposes the Service on each Node's IP at a static port (the NodePort)
- LoadBalancer
  - Exposes the Service externally using a cloud provider's load balancer.
- ExternalName
  - Maps the Service to the contents of the externalName field (e.g. foo.bar.example.com), by returning a CNAME record

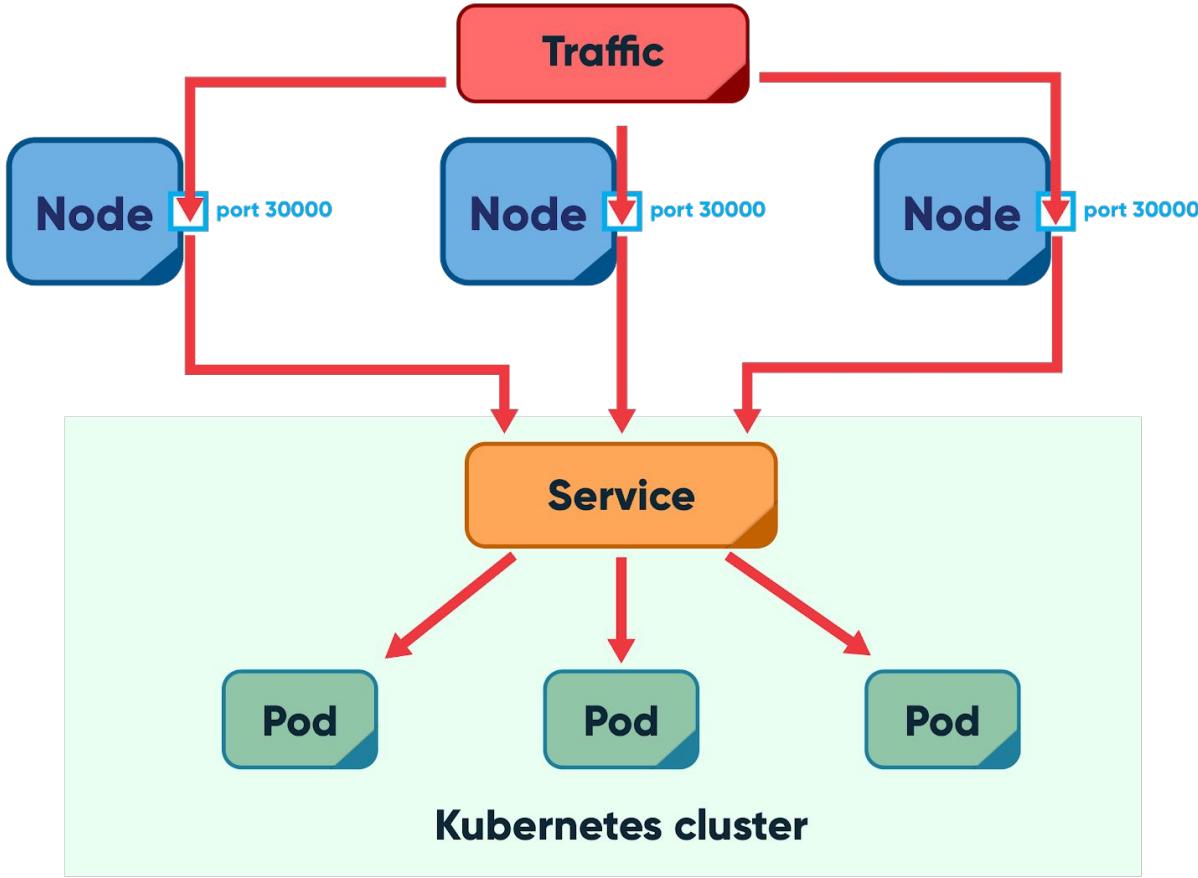
# ClusterIP



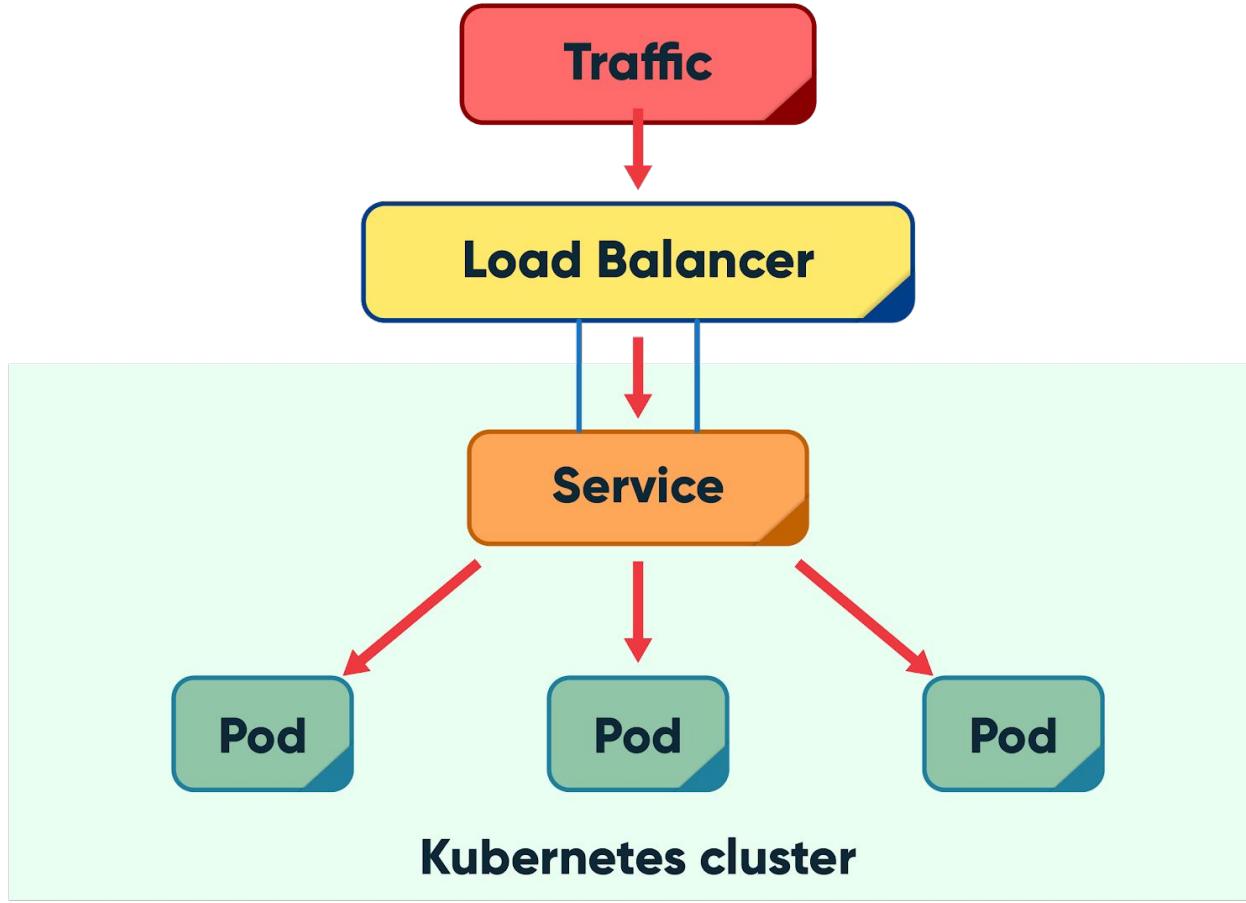
Reference Pictures:

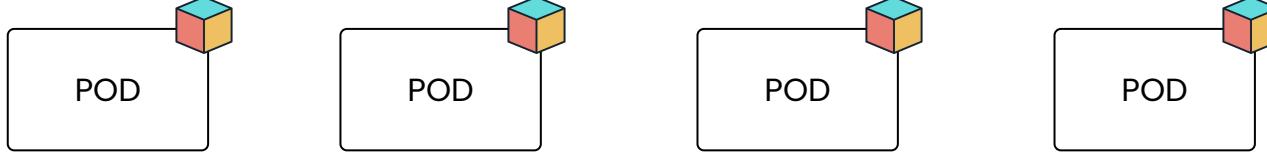
- [Service Types](#)

# NodePort

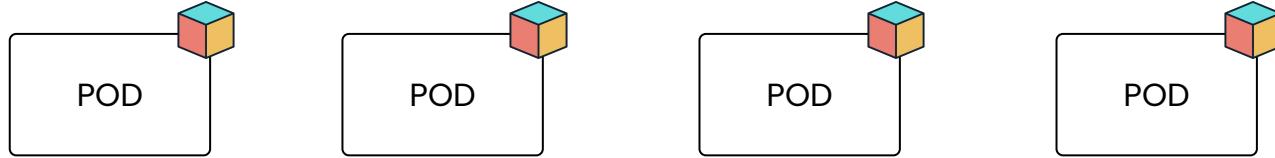


# LoadBalancer



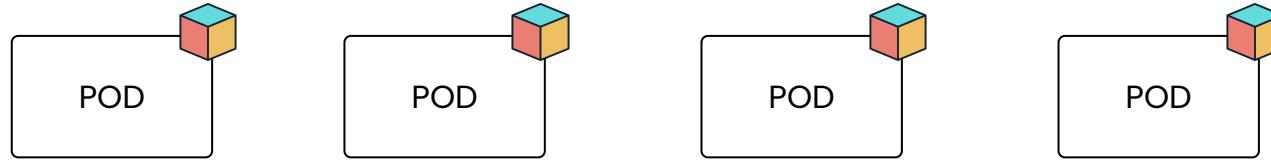


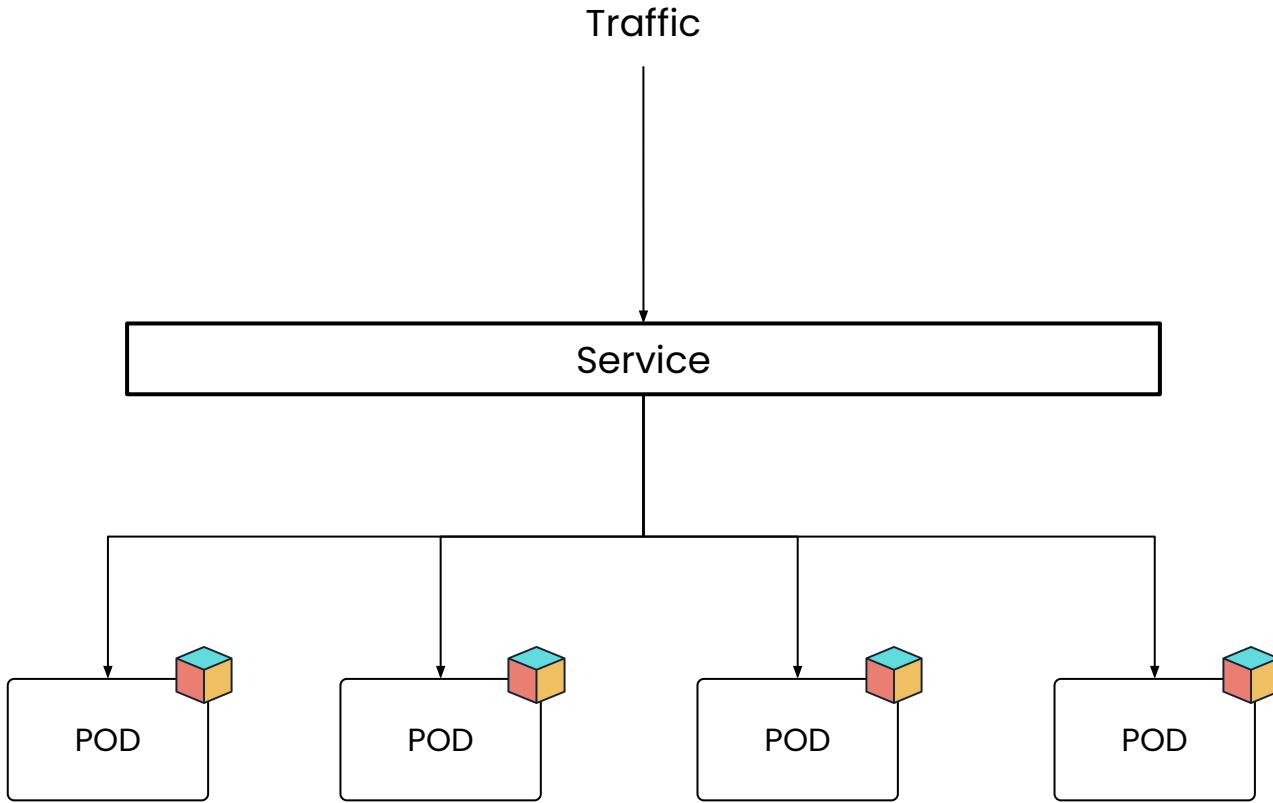
## Traffic



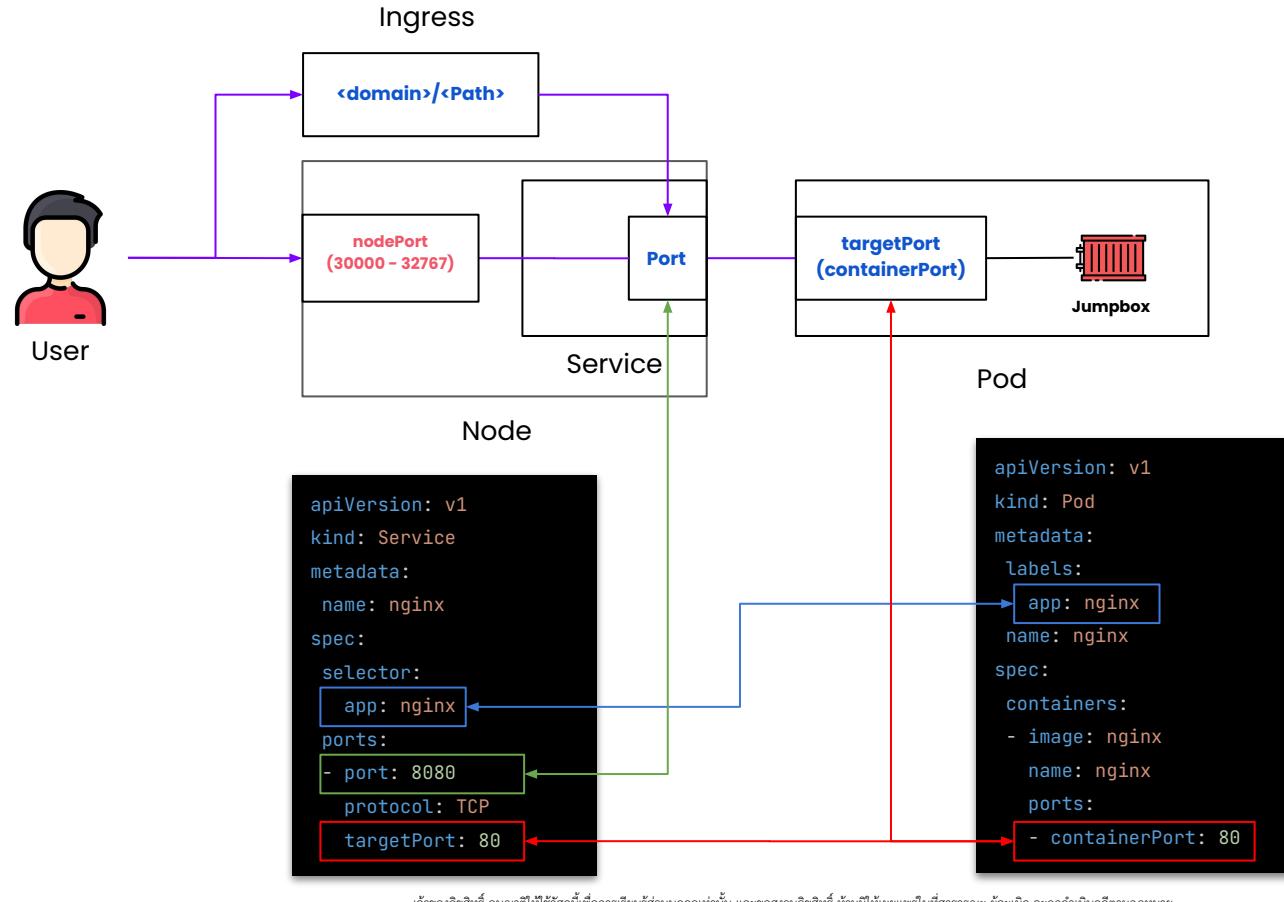
Traffic

Service





# Service Logical View

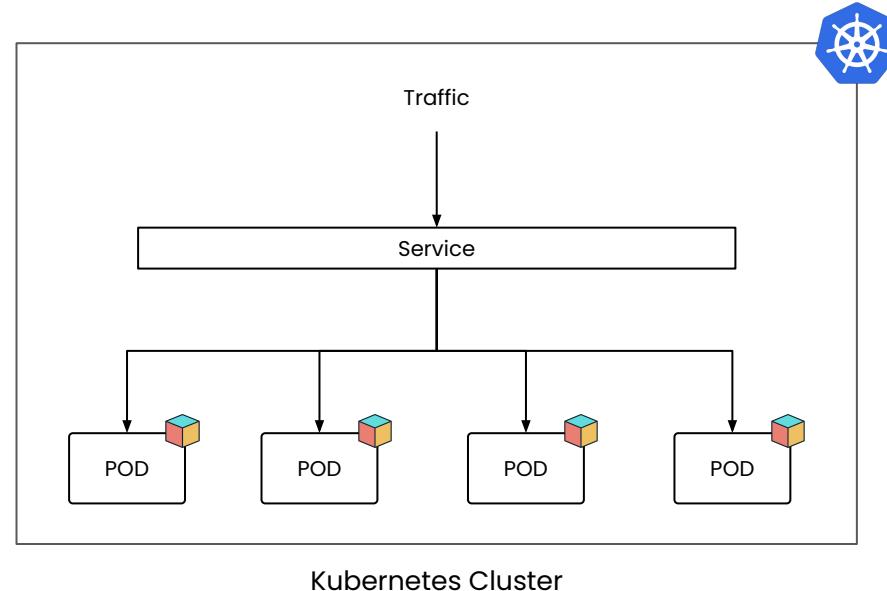


ເຈົ້າຂອງລືສີທີ່ ອຸນຍາດໃຫ້ຮັບສຸດໜີ່ທີ່ການເບີນຊັ້ນບຸກຄຸດທ່ານັ້ນ ແລະຂອງສຽນລືສີທີ່ ທັນມີໄຟເຫັນພຽນໃຫ້ສາງຮັນ ຜູ້ອະນຸມີ ຈະຖຸດຳເນີນຄືຕໍ່ຕາມກູ່ໝາຍ

**Jumpbox®**

# Kubernetes Service Type

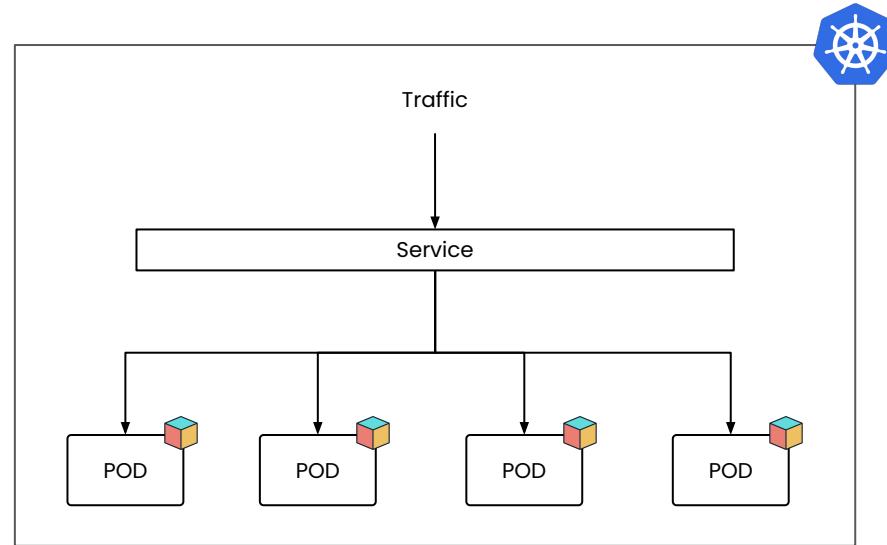
## 1. Cluster IP



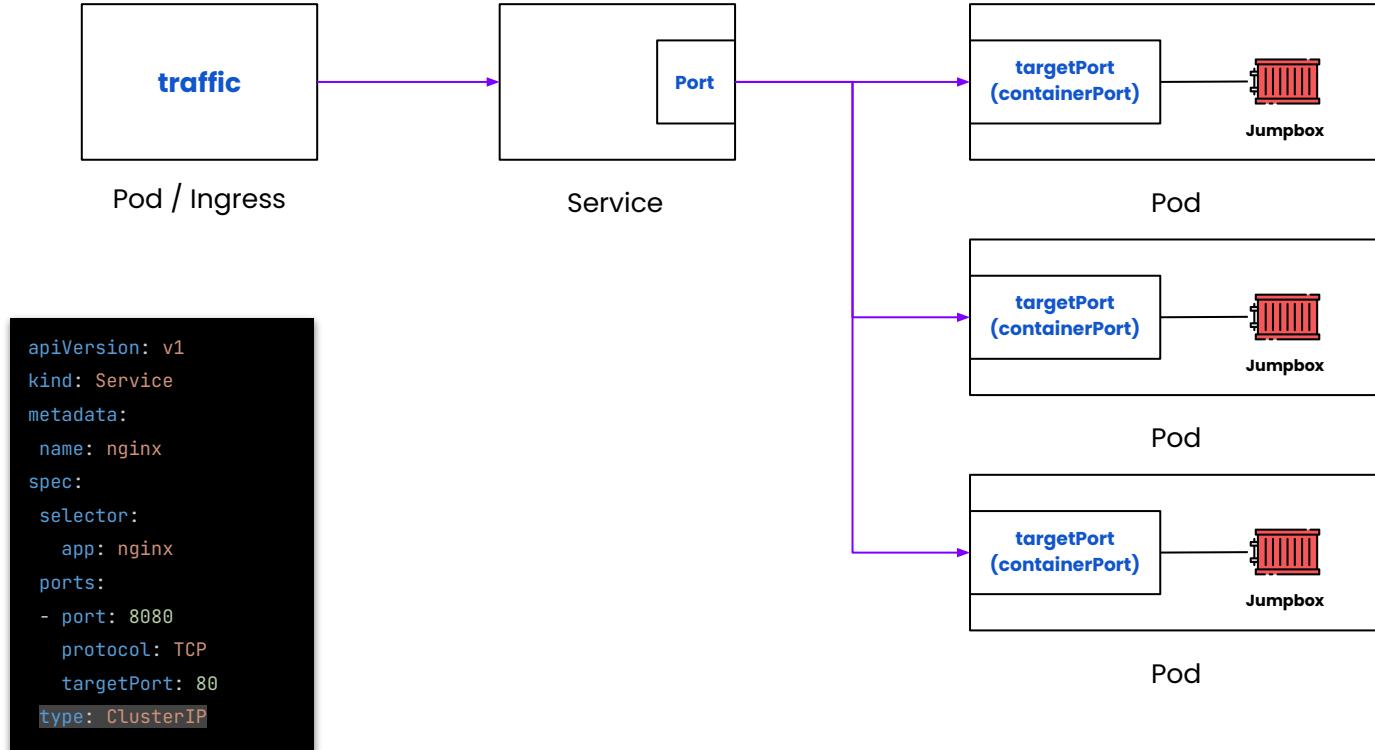
# Service type: ClusterIP

- The **default** Kubernetes service.
- It gives you a service **inside your cluster** that **other apps inside (traffic)** your cluster can access.
- There is **no external access**.

```
apiVersion: v1
kind: Service
metadata:
  name: nginx
spec:
  selector:
    app: nginx
  ports:
  - port: 8080
    protocol: TCP
    targetPort: 80
  type: ClusterIP
```

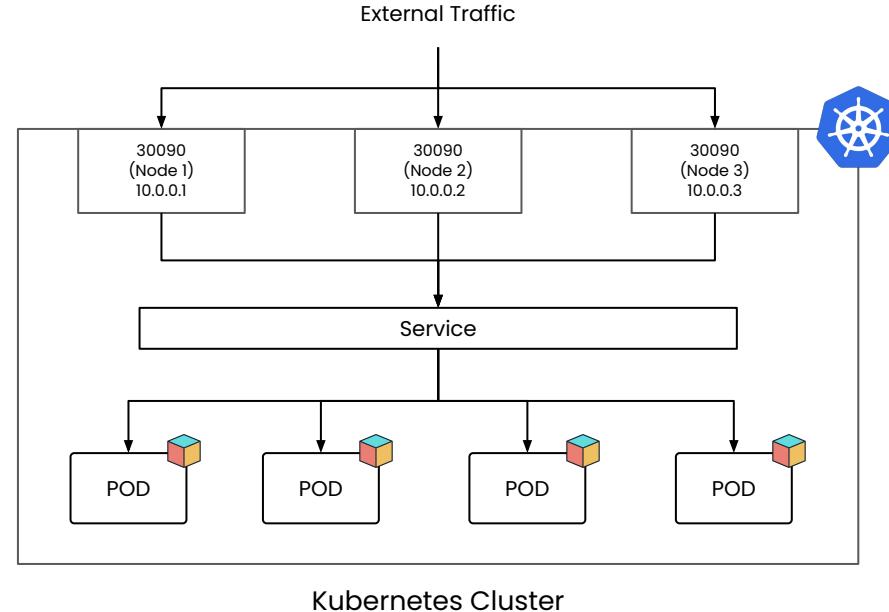


# Service type: ClusterIP (Cont.)



# Kubernetes Service Type

1. Cluster IP
2. NodePort

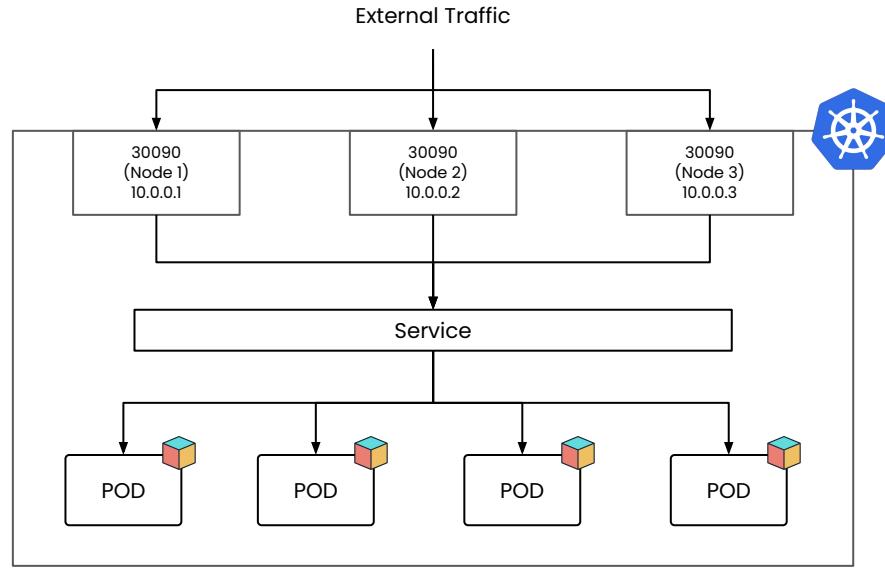


# Service type: NodePort (Cont.)

- The most primitive way to get **external traffic directly** to your service.
- NodePort, as the name implies, opens a specific port on all the Nodes (the VMs), and any traffic that is sent to this port is forwarded to the service.

NodePort Range: 30000 - 32767

```
apiVersion: v1
kind: Service
metadata:
  name: nginx
spec:
  selector:
    app: nginx
  ports:
  - port: 8080
    protocol: TCP
    targetPort: 80
    nodePorts: 30090
    type: NodePort
```

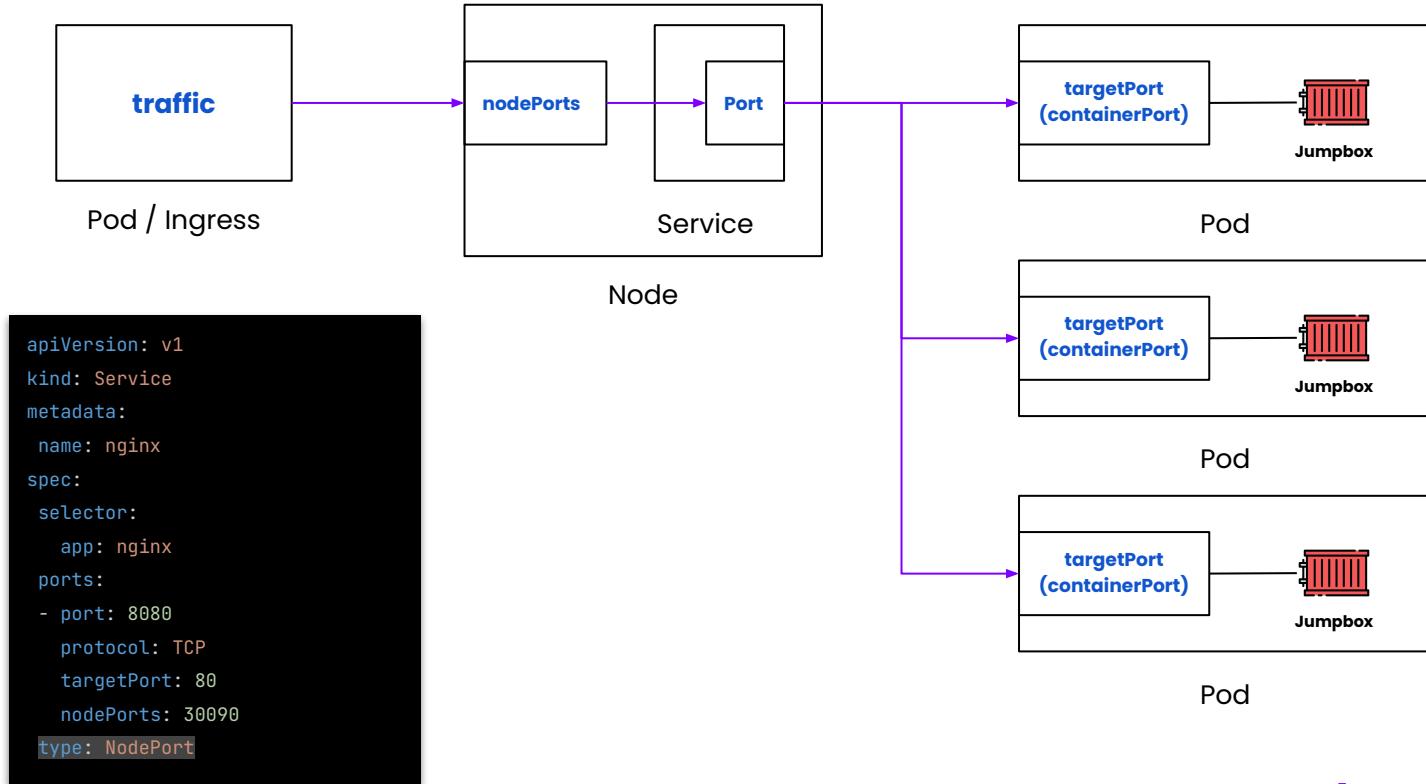


Kubernetes Cluster

เจ้าของลิขสิทธิ์ อนุญาติให้ใช้ส่วนที่ของการเรียนรู้ด้านบุคคลทั่วไป และขอสงวนลิขสิทธิ์ ห้ามมิให้เผยแพร่ในที่สาธารณะ ผู้อื่นเนิด จะถูกดำเนินคดีตามกฎหมาย

Jumpbox®

# Service type: NodePort (Cont.)

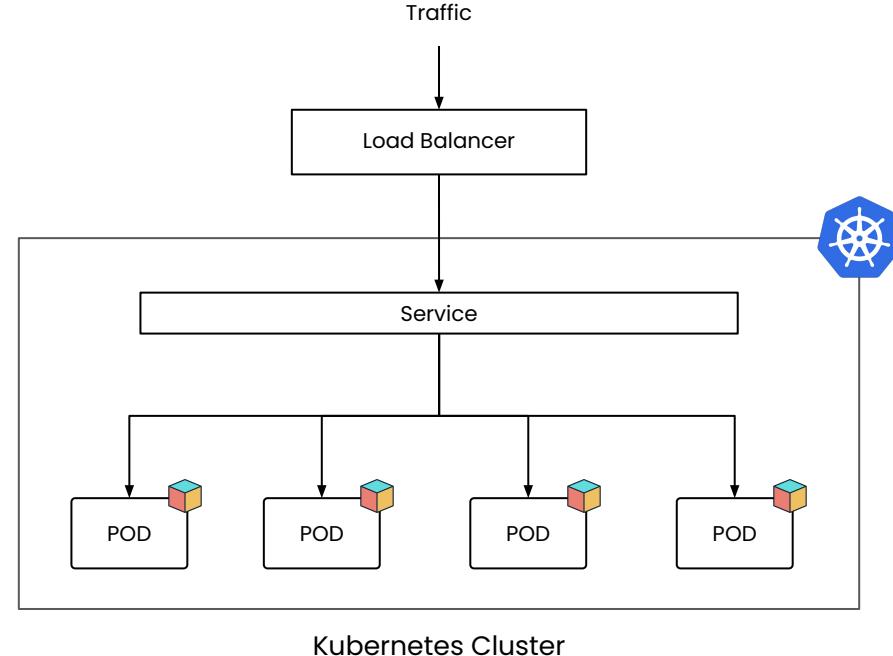


เจ้าของลิขสิทธิ์ อนุญาติให้ใช้ส่วนนี้เพื่อการเรียนรู้ส่วนบุคคลเท่านั้น และขอสงวนลิขสิทธิ์ ห้ามมิให้เผยแพร่ในที่สาธารณะ ผู้ละเมิด จะถูกดำเนินคดีตามกฎหมาย

**Jumpbox®**

# Kubernetes Service Type

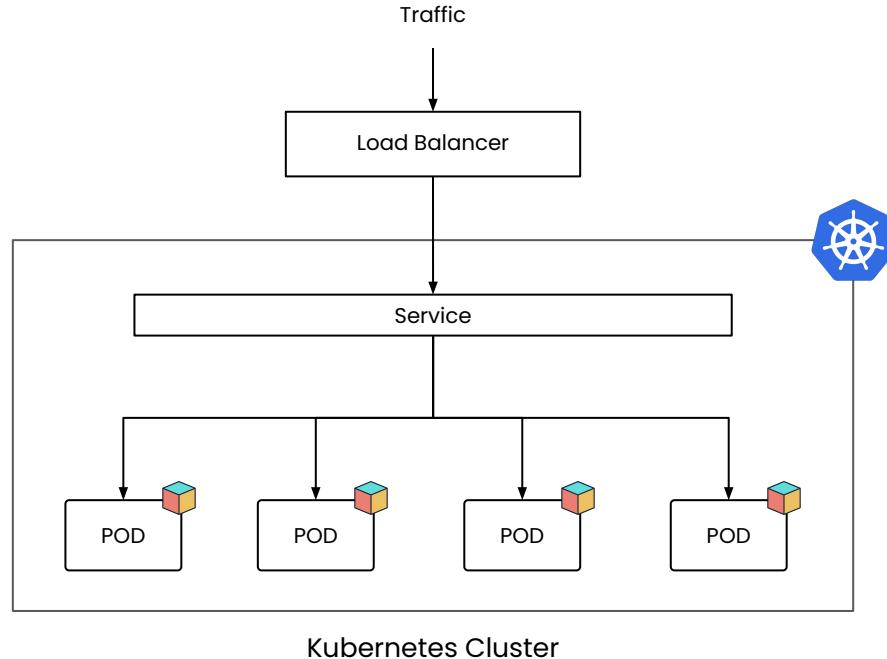
1. Cluster IP
2. NodePort
3. Load Balancer



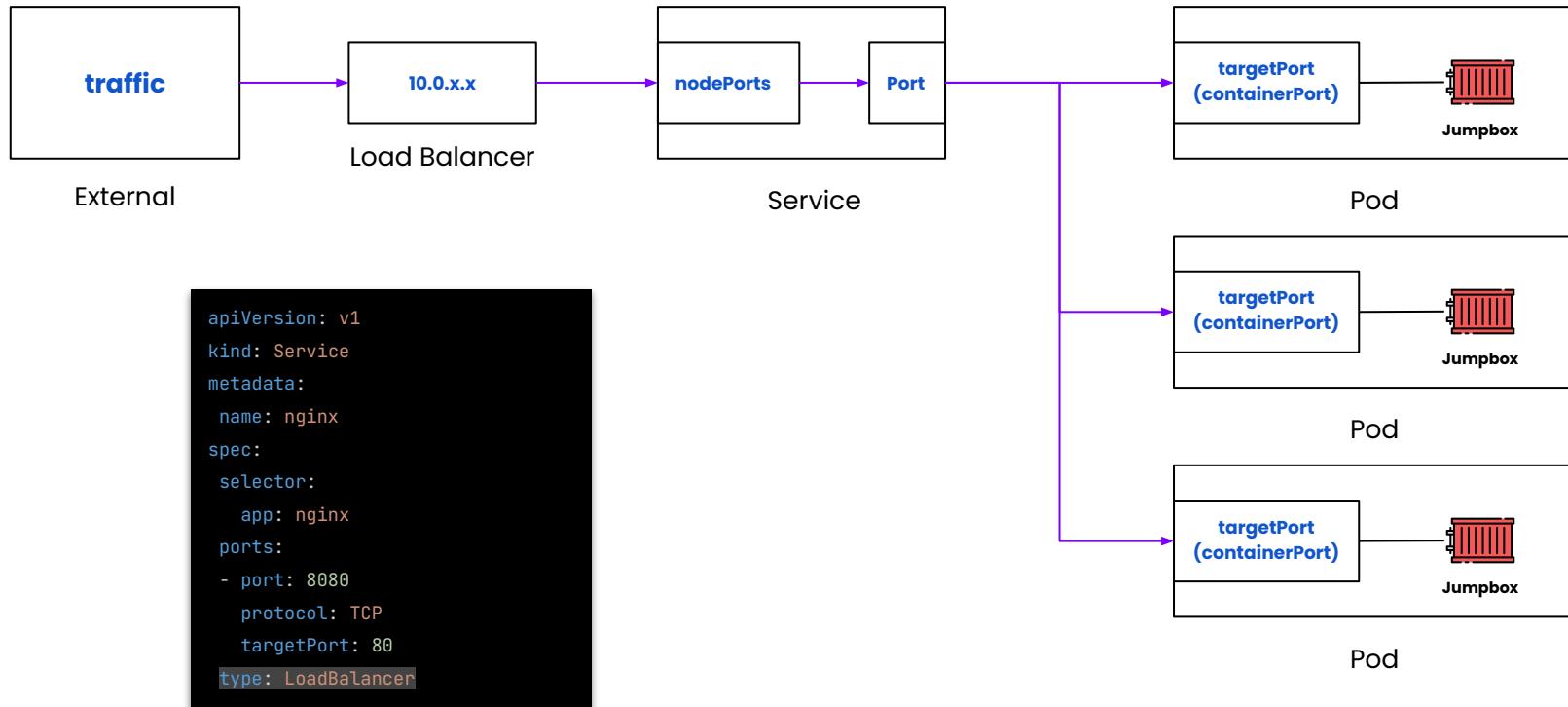
# Service type: LoadBalancer

the standard way to expose a service to the Internet

```
apiVersion: v1
kind: Service
metadata:
  name: nginx
spec:
  selector:
    app: nginx
  ports:
  - port: 8080
    protocol: TCP
    targetPort: 80
  type: LoadBalancer
```

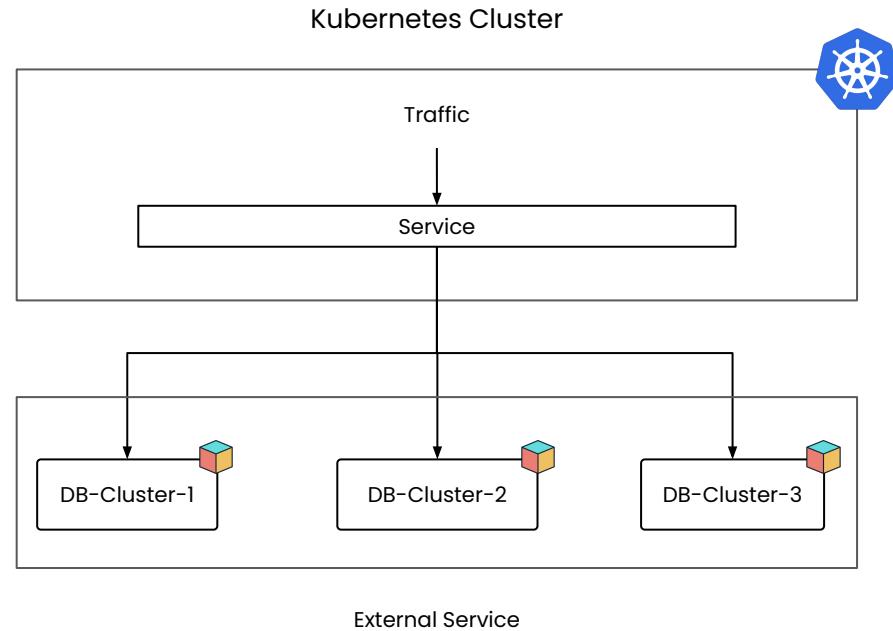


# Service type: LoadBalancer (Cont.)



# Kubernetes Service Type

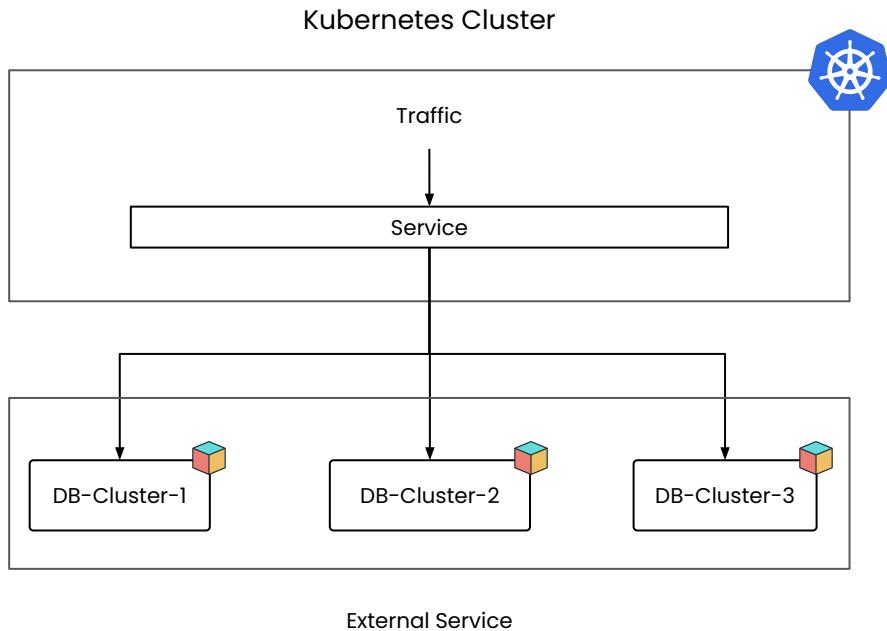
1. Cluster IP
2. NodePort
3. Load Balancer
4. External Name



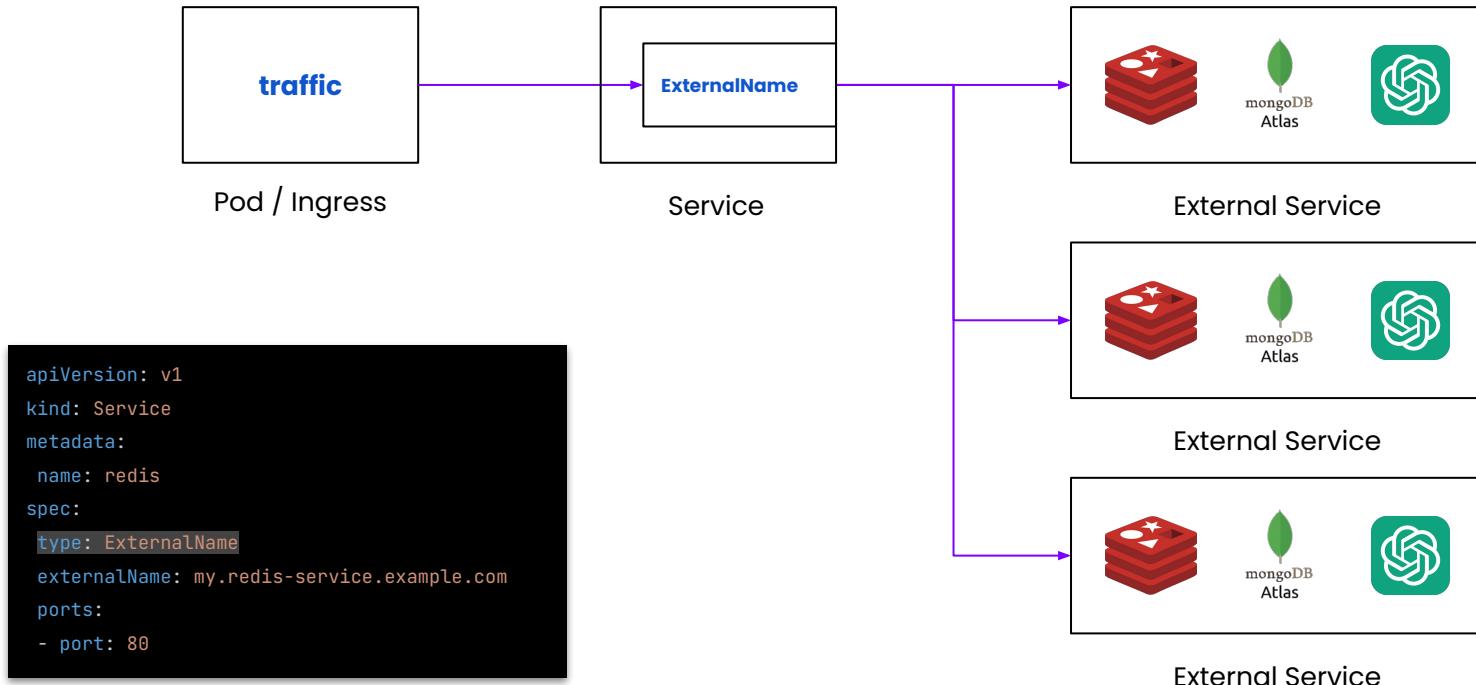
# Service type: ExternalName (Cont.)

- It map a Service to a DNS name, not to a typical selector such as my-service or cassandra.
- You specify these Services with the `spec.type: externalName` parameter.

```
apiVersion: v1
kind: Service
metadata:
  name: redis
spec:
  type: ExternalName
  externalName: my.redis-service.example.com
  ports:
    - port: 80
```

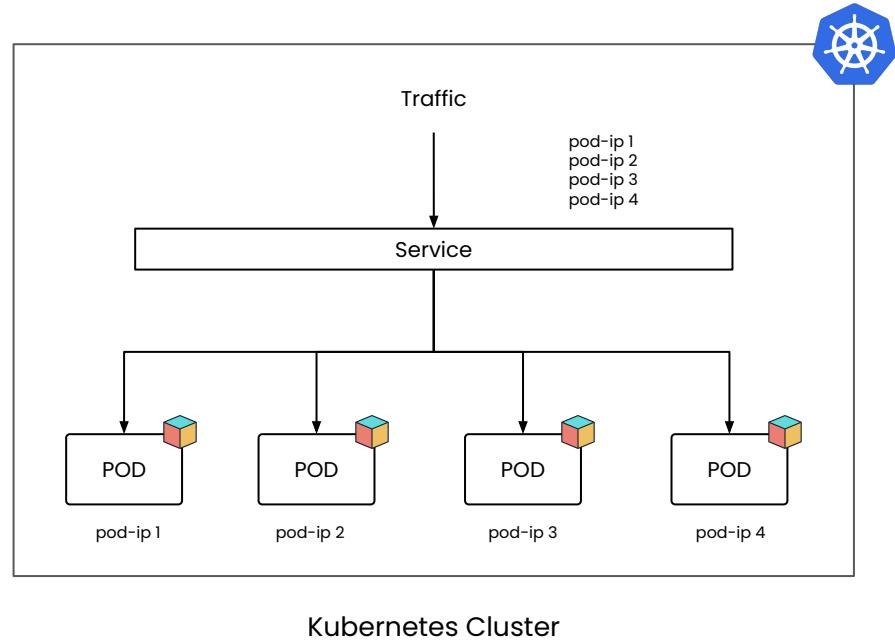


# Service type: ExternalName (Cont.)



# Kubernetes Service Type

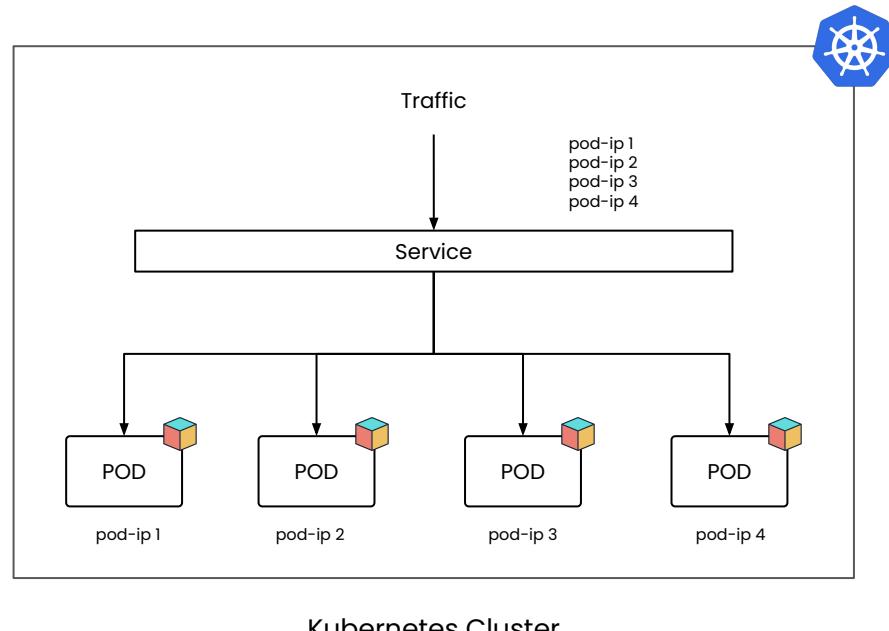
1. Cluster IP
2. NodePort
3. Load Balancer
4. External Name
5. Headless



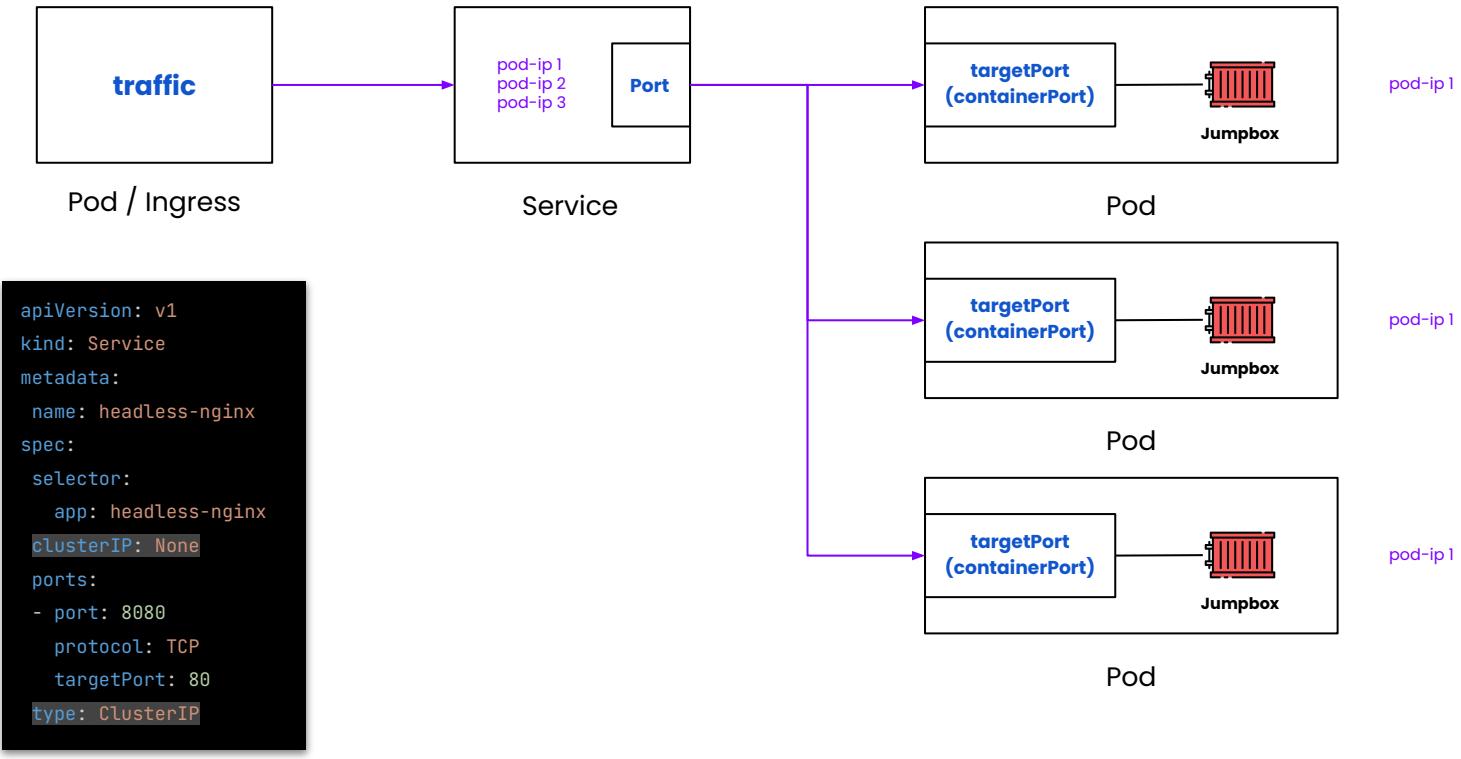
## Service type: ClusterIP, clusterIP: None (Cont.)

- Kubernetes allows clients to **find the IP addresses of the pods** in a service
- Clients can therefore do a **simple DNS A record lookup** and **get the IPs of all the pods** that are part of the service.
- You specify these Services with the **spec.type: ClusterIP** and **clusterIP: None** parameter.

```
apiVersion: v1
kind: Service
metadata:
  name: headless-nginx
spec:
  selector:
    app: headless-nginx
  clusterIP: None
  ports:
    - port: 8080
      protocol: TCP
      targetPort: 80
  type: ClusterIP
```



## Service type: ClusterIP, clusterIP: None (Cont.)



# Service

With Real Action Simulator

[Click Here](#)



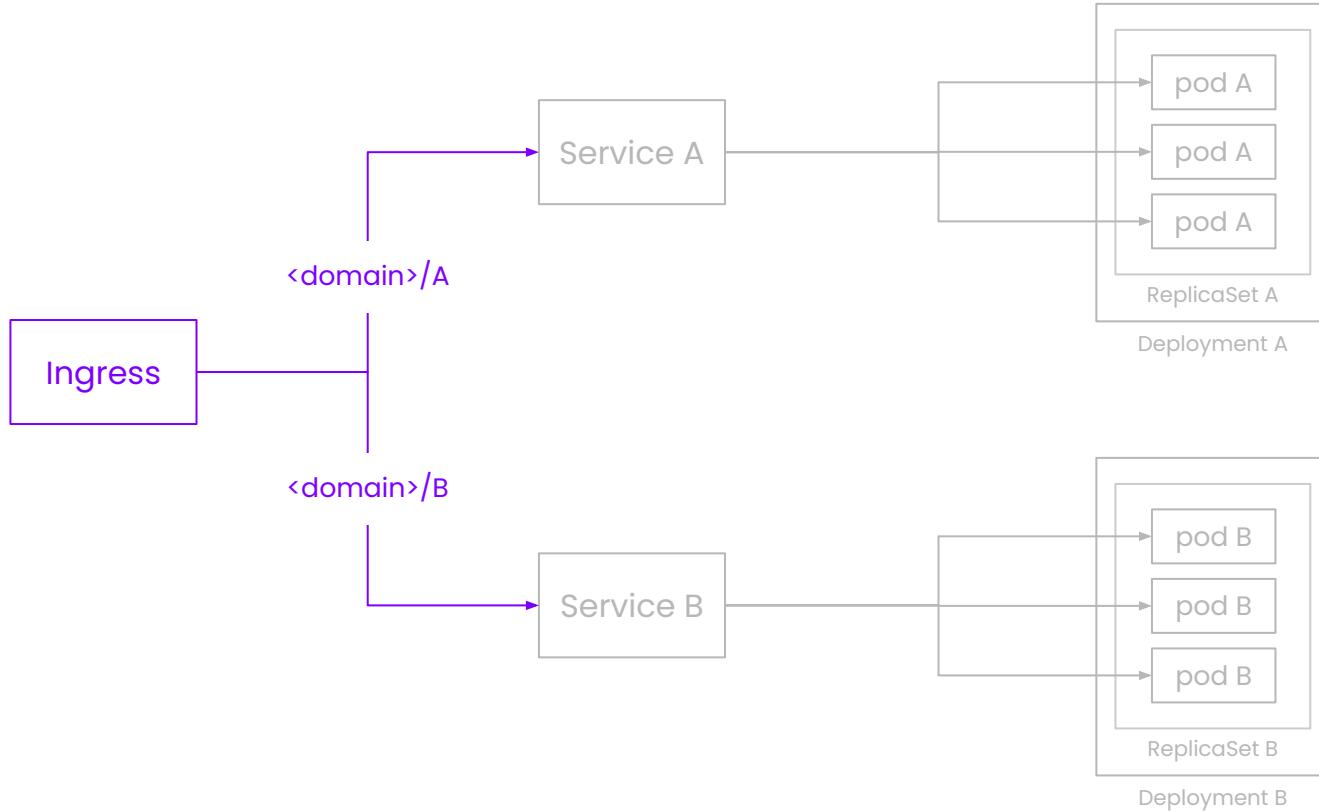
เจ้าของลิขสิทธิ์ อนุญาติให้ใช้ส่วนหนึ่งของการเรียนรู้ส่วนบุคคลเท่านั้น และขอสงวนลิขสิทธิ์ ห้ามมิให้เผยแพร่ในที่สาธารณะ ผู้อื่นเดิน จดถูกต้องเป็นคัดลอก

Jumpbox®

# Ingress

- Working with Kubernetes -

# Kubernetes Overview



W054313196	ingress-nginx	istio ingress	traefik 2.0	kong	contour	haproxy	citrix ingress controller	F5 Networks	voyager	AWS ALB Ingress	Tyk
auth	basic, digest, external auth	JWT	basic, digest and forward auth in alpha	Basic Auth, HMAC, JWT, Key, LDAP, OAuth 2.0, PASETO, plus paid Kong Enterprise options like OpenID Connect	-	basic	basic	Wide range of auth options with APM module	basic,oauth		Basic, Token, OpenID, HMAC, OAuth 2.0, Custom, mTLS, JWT (lua, js, gRPC, go)
Tracing	yes	yes	yes	yes	-	-	-	-	-	-	yes
istio integration	-	yes	-	yes	-	-	-	-	-	-	-
linkerd2	yes	-	yes	-	-	yes	-	-	-	-	-
canary/shadow	canary		canary, mirroring	canary	canary	-	canary	Blue-Green Deployment, A/B Deployment	-		With goreplay plugin <a href="https://github.com/buger/goreplay">https://github.com/buger/goreplay</a>
scope	cross-namespace	cross-namespace	cross-namespace	cross-namespace	cross namespace	optional cross-namespace	cross-namespace	cross-namespace	cross-namespace	cross-namespace	cross-namespace
backend service discovery	dynamic	dynamic	dynamic	dynamic	dynamic	dynamic	dynamic	dynamic	dynamic	dynamic	dynamic
based on	nginx	envoy	traefik	nginx + openresty	envoy	haproxy	Citrix ADC	F5 ADC	haproxy	AWS ALB	Go - Tyk
routing	host.path(with regex)	host.user	host.path	host.path (with regex), method, header	host.path	host.path	host.path	- Full Ingress support - Openshift Routes - Any L3/L4/L7 info when using AS3 Extension	host.path		host, path, method, header RE2 Regexp
protocol	http,https,tcp (separate lb),udp,grpc,fastcgi,IP C socket	tcp,http,https,gr pc	http,https,grpc,tcp + tls	http,https, grpc, tcp, tcp+tls	http,https,tcp,grpc	http,tcp	http,https,tcp,ssl-tcp,udp	tcp, http, https	http,https,tcp	http, https	http, https, gRPC, TCP, TLS-TCP
link	<a href="https://kubernetes.github.io/ingress-nginx/">https://kubernetes.github.io/ingress-nginx/</a>	<a href="https://istio.io/docs/tasks/traffic-management/ingress/">https://istio.io/docs/tasks/traffic-management/ingress/</a>	<a href="https://docs.traefik.io/providers/kubernetes-crd/">https://docs.traefik.io/providers/kubernetes-crd/</a>	<a href="https://github.com/Kong/kubernetes-ingress-controller">https://github.com/Kong/kubernetes-ingress-controller</a>	<a href="https://github.com/projectcontour/contour">https://github.com/projectcontour/contour</a>	<a href="https://www.haproxy.com/blog/proxy-ingress_controller_for_kubernetes/">https://www.haproxy.com/blog/proxy-ingress_controller_for_kubernetes/</a>	<a href="https://github.com/citrix-k8s-ingress-controller">https://github.com/citrix-k8s-ingress-controller</a>	<a href="https://github.com/istio/api/blob/master/networking/v1alpha3/gateway.proto">https://github.com/istio/api/blob/master/networking/v1alpha3/gateway.proto</a>	<a href="https://appcode.com/products/voyager/">https://appcode.com/products/voyager/</a>	<a href="https://github.com/kubernetes-sigs/aws-alb-ingress-controller/helm-chart/#using-the-ingress-controller">https://github.com/kubernetes-sigs/aws-alb-ingress-controller/helm-chart/#using-the-ingress-controller</a>	<a href="https://github.com/TykTechnologies/tyk-helm-chart">https://github.com/TykTechnologies/tyk-helm-chart</a>

Credit:

[https://docs.google.com/spreadsheets/d/16bxRgpO1H\\_Bn-5xVZ1WrR\\_I-0A-GOI6egmhvqqLMOmg/edit#gid=1612037324](https://docs.google.com/spreadsheets/d/16bxRgpO1H_Bn-5xVZ1WrR_I-0A-GOI6egmhvqqLMOmg/edit#gid=1612037324)

เจ้าของลิขสิทธิ์ อนุญาติให้ใช้ส่วนที่เพื่อการเรียนรู้ส่วนบุคคลเท่านั้น และขอสงวนลิขสิทธิ์ ห้ามมิให้เผยแพร่ในที่สาธารณะ ผู้ละเมิด จะถูกดำเนินคดีตามกฎหมาย

Jumpbox®

Service A

Service B

Service ...

Service n

# Users

Service A

Service B

Service ...

Service n

Users

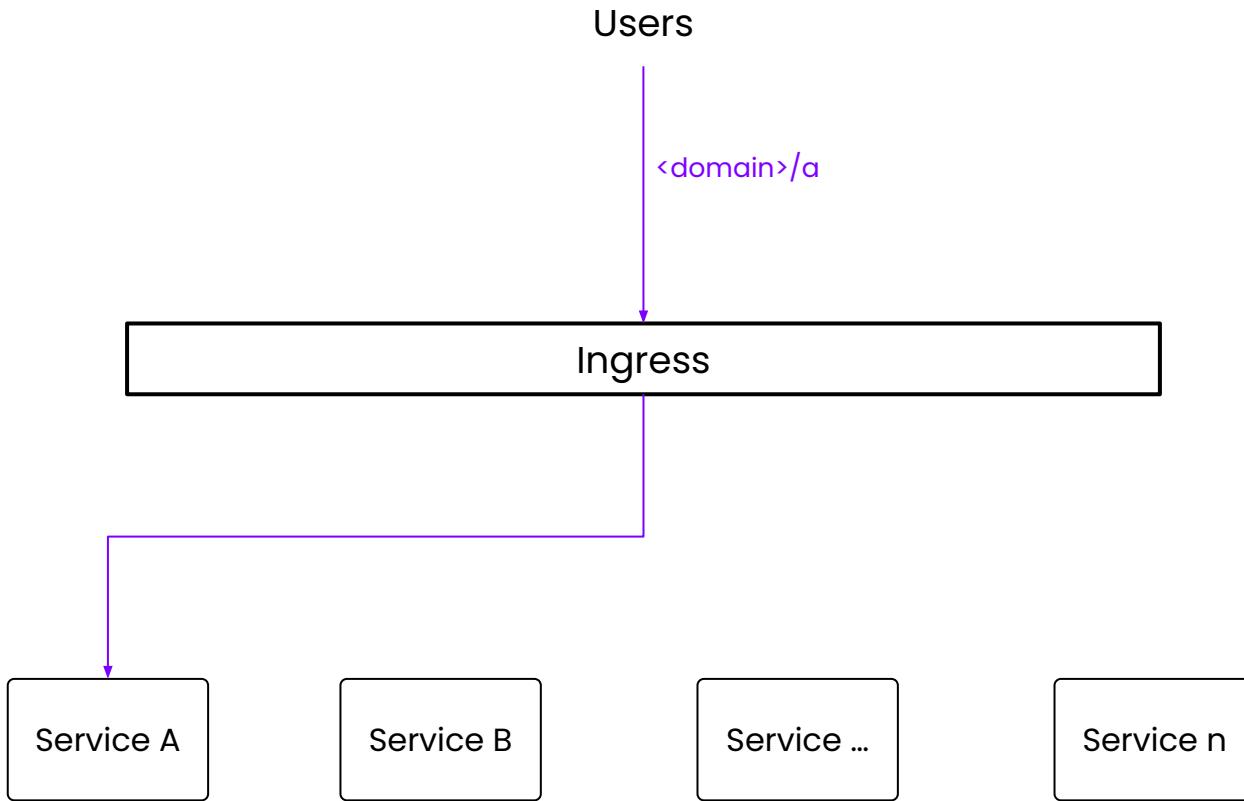
Ingress

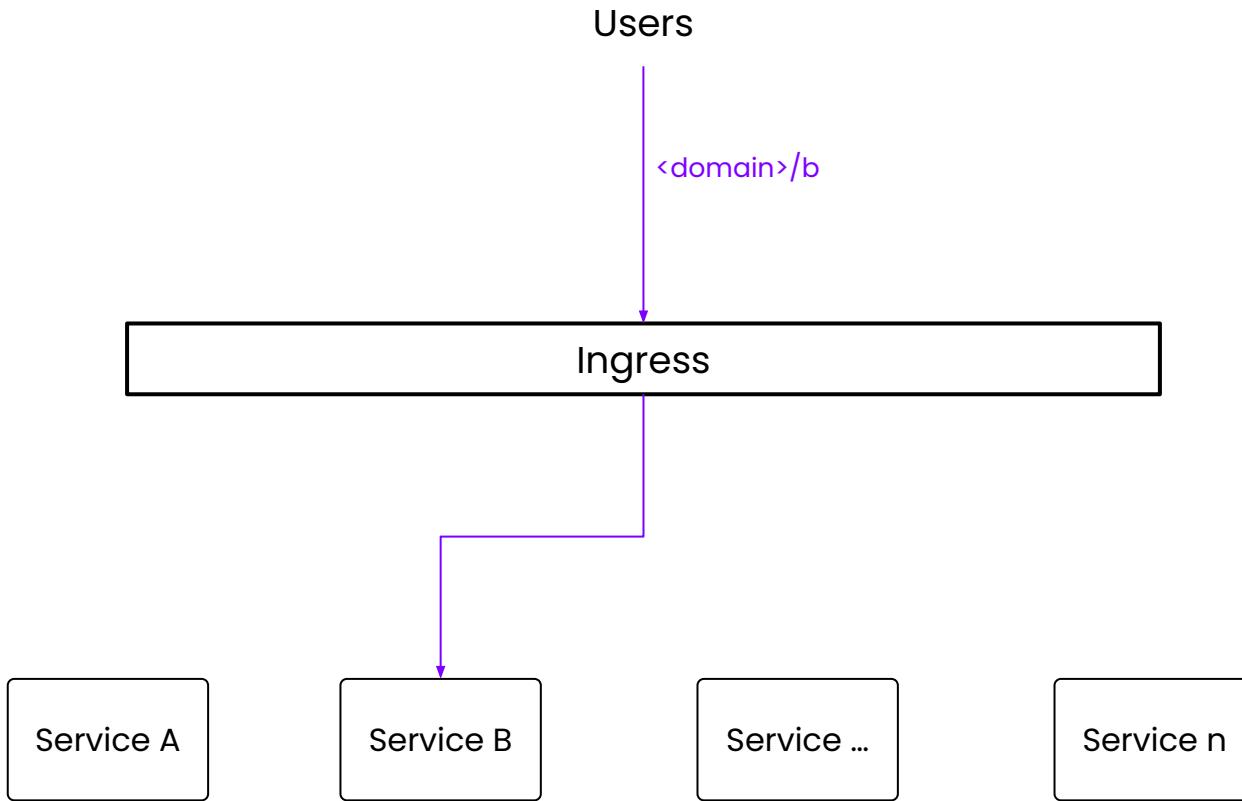
Service A

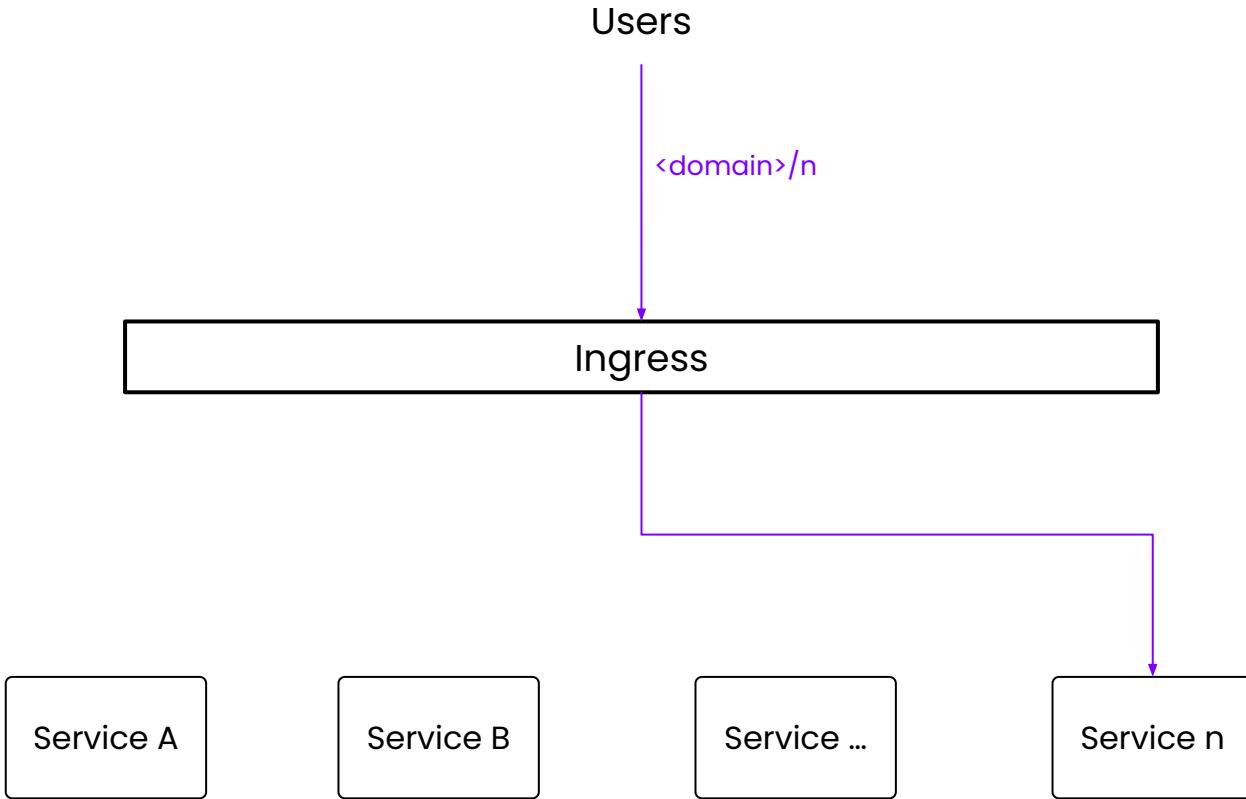
Service B

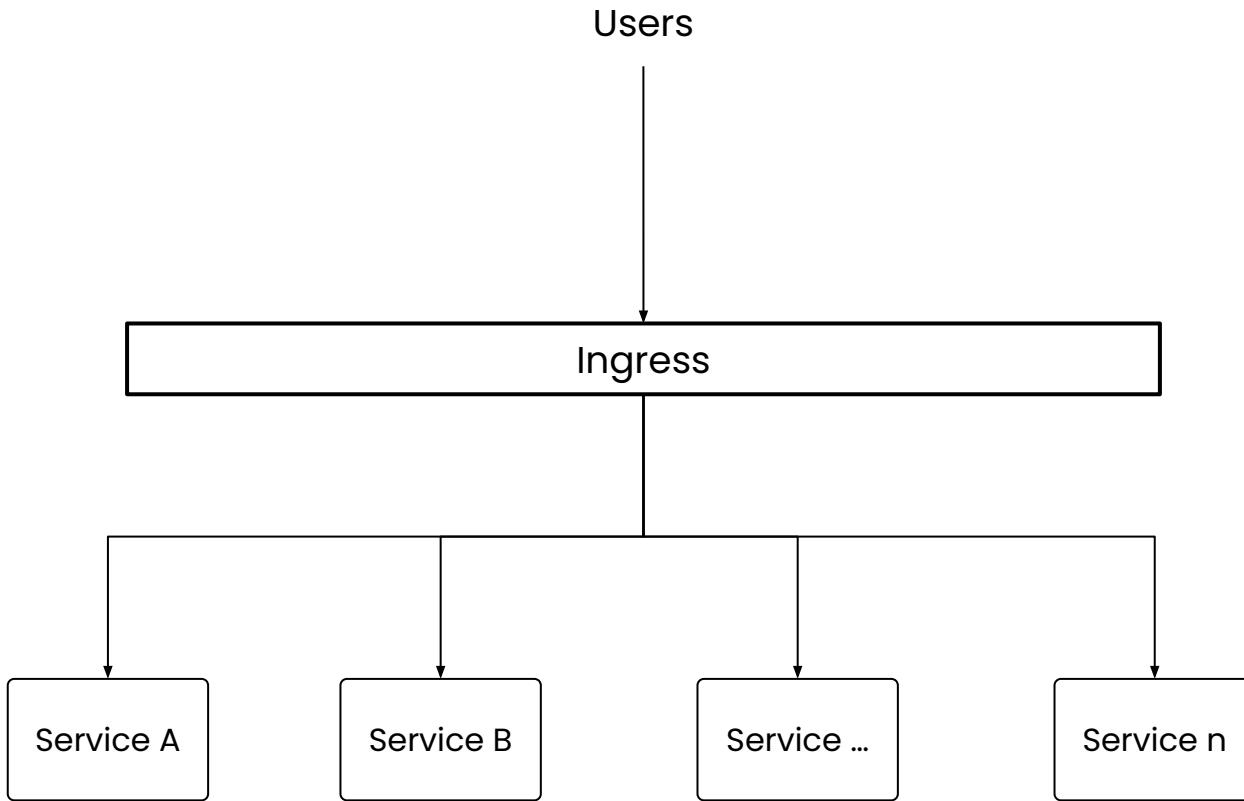
Service ...

Service n





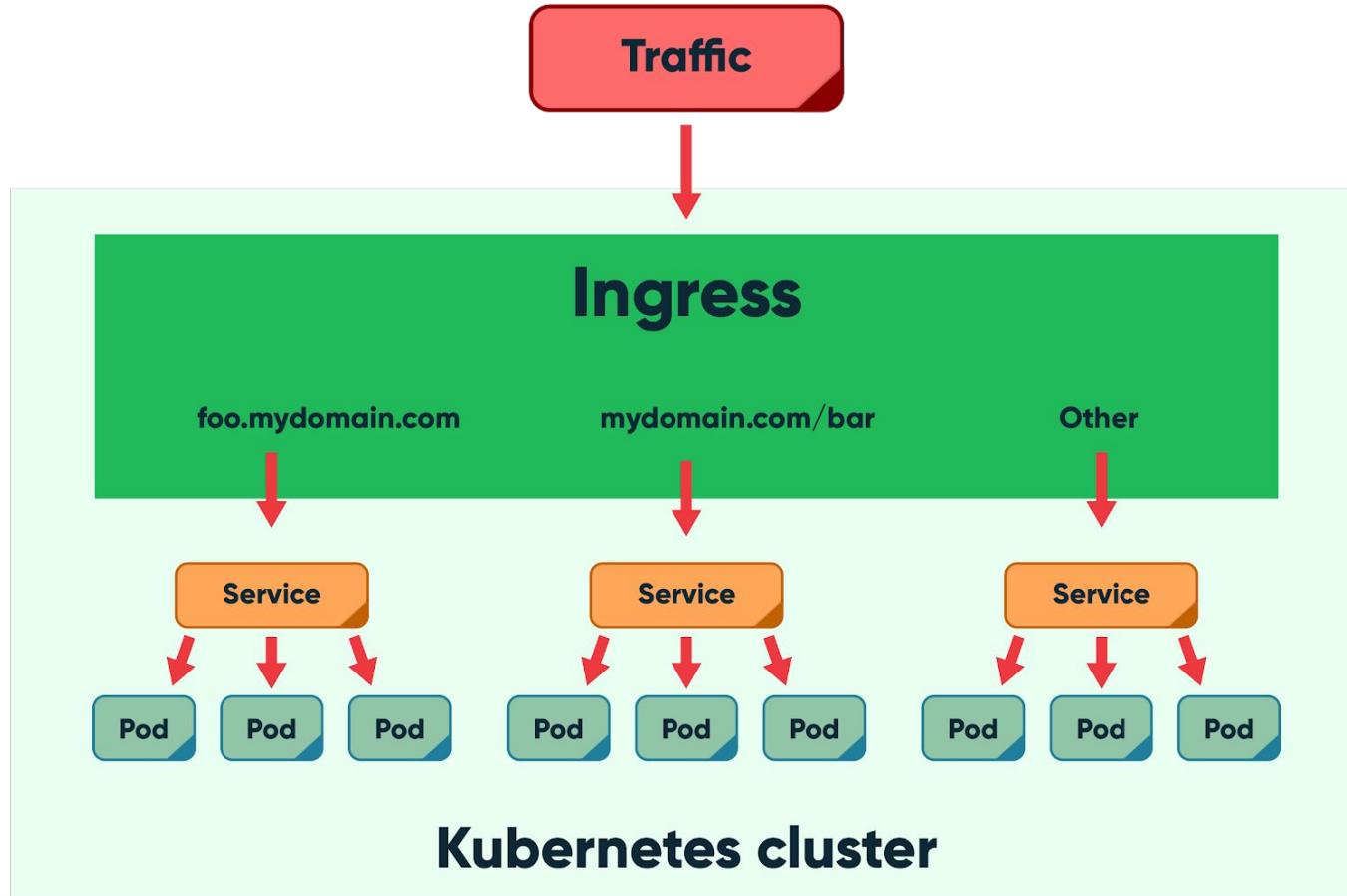




# Ingress

- Exposes HTTP and HTTPS routes from outside the cluster to services within the cluster.
- Traffic routing is controlled by rules defined on the Ingress resource.

# Ingress



# Ingress Structure

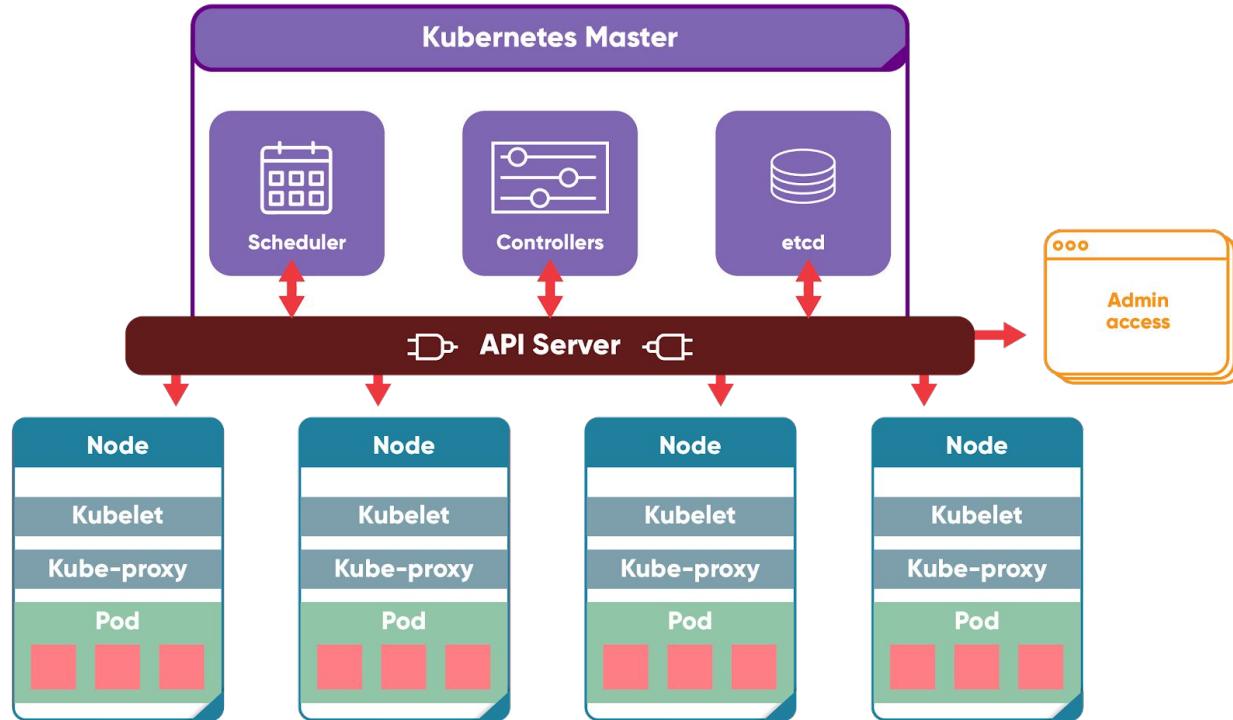
```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  annotations:
    nginx.ingress.kubernetes.io/backend-protocol: HTTPS
    nginx.ingress.kubernetes.io/rewrite-target: /$2
  name: nginx
spec:
  ingressClassName: nginx
  rules:
  - http:
      paths:
      - backend:
          service:
            name: nginx
          port:
            number: 80
        path: /api(/|$)(.*)
        pathType: Prefix
```

```
apiVersion: v1
kind: Service
metadata:
  name: nginx
spec:
  selector:
    app: nginx
  ports:
    port: 80
    protocol: TCP
    targetPort: 80
  type: ClusterIP
```

# Kubernetes Architecture

# Kubernetes Architecture

## Kubernetes cluster



# Command and Arguments

- Inject Data Into Applications -

- Command and Arguments
- Environment Variables

# Command and Arguments

# Command and Arguments (Cont.)

```
apiVersion: v1
kind: Pod
metadata:
  name: command-demo
  labels:
    purpose: demonstrate-command
spec:
  containers:
  - name: command-demo-container
    image: debian
    command: ["printenv"]
    args: ["HOSTNAME", "KUBERNETES_PORT"]
  restartPolicy: OnFailure
```

## Reference:

- [Command and Argument](#)

เจ้าของลิขสิทธิ์ อนุญาติให้ใช้ส่วนนี้เพื่อการเรียนรู้ส่วนบุคคลเท่านั้น และขอสงวนลิขสิทธิ์ ห้ามมิให้เผยแพร่ในที่สาธารณะ ผู้ละเมิด จะถูกดำเนินคดีตามกฎหมาย

# Command and Arguments (Cont.)

```
apiVersion: v1
kind: Pod
metadata:
  name: command-demo
  labels:
    purpose: demonstrate-command
spec:
  containers:
    - name: command-demo-container
      image: debian
      command: ["printenv"]
      args: ["HOSTNAME", "KUBERNETES_PORT"]
  restartPolicy: OnFailure
```

## Reference:

- [Command and Argument](#)

เจ้าของลิขสิทธิ์ อนุญาติให้ใช้ส่วนนี้เพื่อการเรียนรู้ส่วนบุคคลเท่านั้น และขอสงวนลิขสิทธิ์ ห้ามมิให้เผยแพร่ในที่สาธารณะ ผู้ละเมิด จะถูกดำเนินคดีตามกฎหมาย

## Command and Arguments (Cont.)

```
apiVersion: v1
kind: Pod
metadata:
  name: command-demo
  labels:
    purpose: demonstrate-command
spec:
  containers:
    - name: command-demo-container
      image: debian
      command: ["printenv"]
      args: ["HOSTNAME", "KUBERNETES_PORT"]
  restartPolicy: OnFailure
```

- When you **create a Pod**, you can define a **command and arguments** for the containers that run in the Pod.

### Reference:

- [Command and Argument](#)

เจ้าของลิขสิทธิ์ อนุญาติให้ใช้ส่วนนี้เพื่อการเรียนรู้ส่วนบุคคลเท่านั้น และขอสงวนลิขสิทธิ์ ห้ามมิให้เผยแพร่ในที่สาธารณะ ผู้อื่นเดินทาง เดินทางโดยไม่มีสิทธิ์

# Command and Arguments (Cont.)

```
apiVersion: v1
kind: Pod
metadata:
  name: command-demo
  labels:
    purpose: demonstrate-command
spec:
  containers:
    - name: command-demo-container
      image: debian
      command: ["printenv"]
      args: ["HOSTNAME", "KUBERNETES_PORT"]
  restartPolicy: OnFailure
```

- When you create a Pod, you can define a command and arguments for the containers that run in the Pod.
- To define a command, include the **command field** in the configuration file.

## Reference:

- [Command and Argument](#)

เจ้าของลิขสิทธิ์ อนุญาติให้ใช้ส่วนนี้เพื่อการเรียนรู้ส่วนบุคคลเท่านั้น และขอสงวนลิขสิทธิ์ ห้ามมิให้เผยแพร่ในที่สาธารณะ ผู้ละเมิด จะถูกดำเนินคดีตามกฎหมาย

# Command and Arguments (Cont.)

```
apiVersion: v1
kind: Pod
metadata:
  name: command-demo
  labels:
    purpose: demonstrate-command
spec:
  containers:
    - name: command-demo-container
      image: debian
      command: ["printenv"]
      args: ["HOSTNAME", "KUBERNETES_PORT"]
  restartPolicy: OnFailure
```

- When you create a Pod, you can define a command and arguments for the containers that run in the Pod.
- To define a command, include the `command` field in the configuration file.
- To define arguments for the command, include the `args` field in the configuration file.

## Reference:

- [Command and Argument](#)

# Command and Arguments (Cont.)

```
apiVersion: v1
kind: Pod
metadata:
  name: command-demo
  labels:
    purpose: demonstrate-command
spec:
  containers:
    - name: command-demo-container
      image: debian
      command: ["printenv"]
      args: ["HOSTNAME", "KUBERNETES_PORT"]
  restartPolicy: OnFailure
```

- When you create a Pod, you can define a command and arguments for the containers that run in the Pod.
- To define a command, include the command field in the configuration file.
- To define arguments for the command, include the args field in the configuration file.
- The command and arguments that you define **cannot be changed after the Pod is created.**

## Reference:

- [Command and Argument](#)

# Command and Arguments (Cont.)

```
apiVersion: v1
kind: Pod
metadata:
  name: command-demo
  labels:
    purpose: demonstrate-command
spec:
  containers:
    - name: command-demo-container
      image: debian
      command: ["printenv"]
      args: ["HOSTNAME", "KUBERNETES_PORT"]
  restartPolicy: OnFailure
```

- When you **create a Pod**, you can define a **command and arguments** for the containers that run in the Pod.
- To define a command, include the **command field** in the configuration file.
- To define arguments for the command, include the **args field** in the configuration file.
- The command and arguments that you define **cannot be changed after the Pod is created**.

## Reference:

- [Command and Argument](#)

# Environment Variables

- Inject Data Into Applications -

# Define an **environment variable** for a container

## Define an **environment variable** for a container (Cont.)

- When you create a Pod, you can set environment variables for the containers that run in the Pod.
- To set environment variables, include the `env` or `envFrom` field in the configuration file.

Reference:

- [environment variable](#)

## Define an **environment variable** for a container (Cont.)

- When you create a Pod, you can set environment variables for the containers that run in the Pod.
- To set environment variables, include the env or envFrom field in the configuration file.

### **env:**

- allows you to **set environment variables** for a container, specifying a value directly for each variable that you name.

### Reference:

- [environment variable](#)

# Define an **environment variable** for a container (Cont.)

- When you create a Pod, you can set environment variables for the containers that run in the Pod.
- To set environment variables, include the env or envFrom field in the configuration file.

## **env:**

- allows you to set environment variables for a container, specifying a value directly for each variable that you name.

## **envFrom:**

- allows you to set environment variables for a container by referencing either a **ConfigMap** or a **Secret**.
- When you use envFrom, all the **key-value pairs** in the referenced ConfigMap or Secret are **set as environment variables** for the container.
- You can also specify a common prefix string.

## Reference:

- [environment variable](#)

# Define an **environment variable** for a container (Cont.)

- When you create a Pod, you can set environment variables for the containers that run in the Pod.
- To set environment variables, include the `env` or `envFrom` field in the configuration file.

## **env:**

- allows you to **set environment variables** for a container, specifying a value directly for each variable that you name.

## **envFrom:**

- allows you to set environment variables for a container by referencing either a **ConfigMap** or a **Secret**.
- When you use envFrom, all the **key-value pairs** in the referenced ConfigMap or Secret are **set as environment variables** for the container.
- You can also specify a common prefix string.

## Reference:

- [environment variable](#)

## Define an **environment variable** for a container (Cont.)

```
apiVersion: v1
kind: Pod
metadata:
  name: envar-demo
  labels:
    purpose: demonstrate-envvars
spec:
  containers:
  - name: envar-demo-container
    image: gcr.io/google-samples/node-hello:1.0
    env:
    - name: DEMO_GREETING
      value: "Hello from the environment"
    - name: DEMO_JUMPBOX
      value: "Jumpbox is Here"
```

### Reference:

- [environment variable](#)

เจ้าของลิขสิทธิ์ อนุญาติให้ใช้ส่วนนี้เพื่อการเรียนรู้ส่วนบุคคลเท่านั้น และขอสงวนลิขสิทธิ์ ห้ามมิให้เผยแพร่ในที่สาธารณะ ผู้อื่นเดินทาง จะถูกดำเนินคดีตามกฎหมาย

**Jumpbox®**

## Define an **environment variable** for a container (Cont.)

```
apiVersion: v1
kind: Pod
metadata:
  name: envar-demo
  labels:
    purpose: demonstrate-envvars
spec:
  containers:
    - name: envar-demo-container
      image: gcr.io/google-samples/node-hello:1.0
      env:
        - name: DEMO_GREETING
          value: "Hello from the environment"
        - name: DEMO_JUMPBOX
          value: "Jumpbox is Here"
```



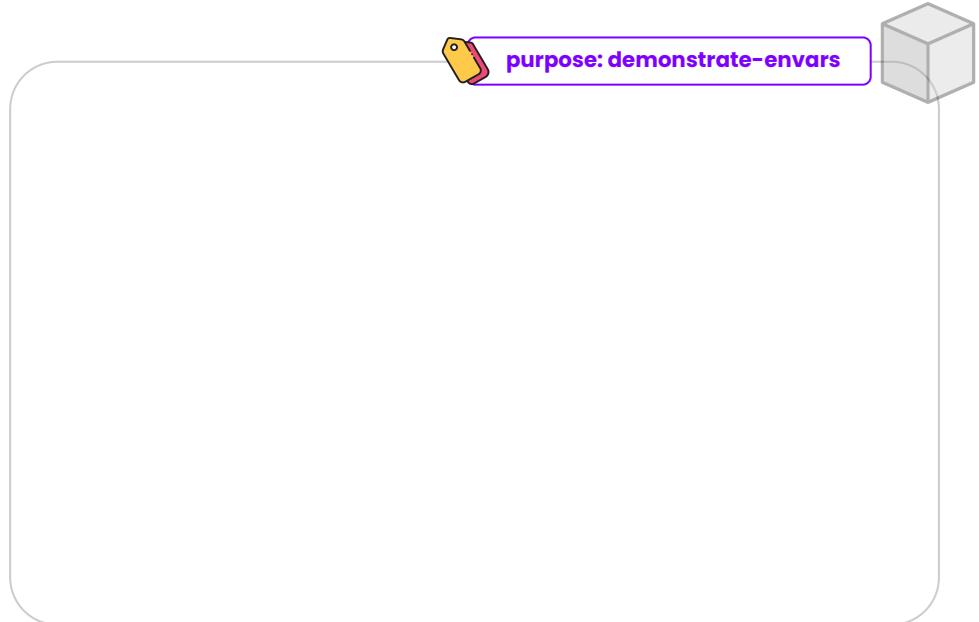
**Pod: envar-demo**

### Reference:

- [environment variable](#)

# Define an **environment variable** for a container (Cont.)

```
apiVersion: v1
kind: Pod
metadata:
  name: envar-demo
  labels:
    purpose: demonstrate-envvars
spec:
  containers:
    - name: envar-demo-container
      image: gcr.io/google-samples/node-hello:1.0
      env:
        - name: DEMO_GREETING
          value: "Hello from the environment"
        - name: DEMO_JUMPBOX
          value: "Jumpbox is Here"
```

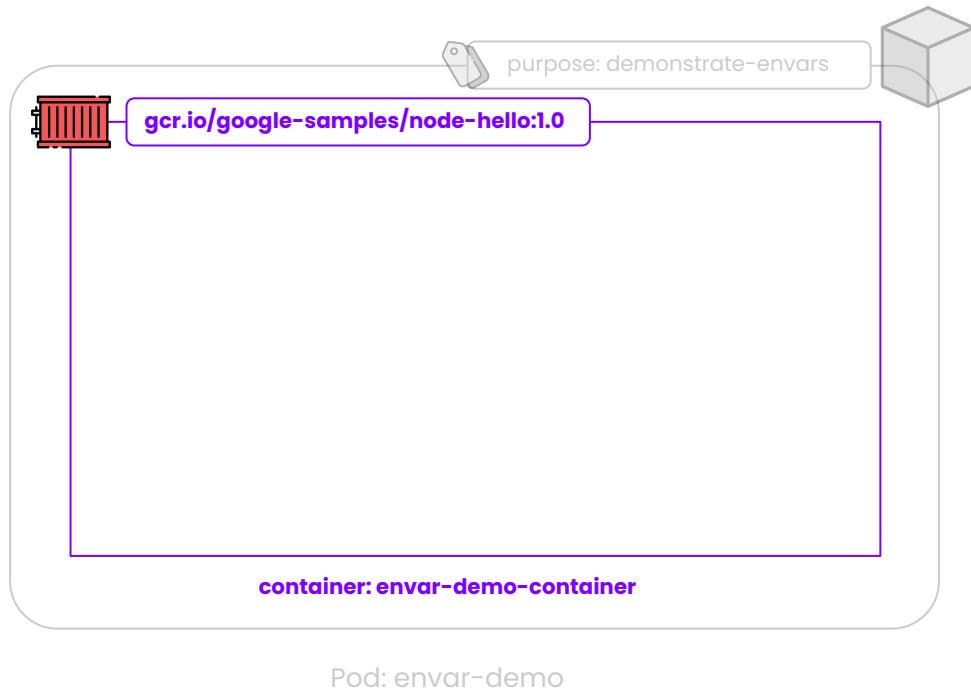


## Reference:

- [environment variable](#)

## Define an **environment variable** for a container (Cont.)

```
apiVersion: v1
kind: Pod
metadata:
  name: envar-demo
  labels:
    purpose: demonstrate-envars
spec:
  containers:
    - name: envar-demo-container
      image: gcr.io/google-samples/node-hello:1.0
      env:
        - name: DEMO_GREETING
          value: "Hello from the environment"
        - name: DEMO_JUMPBOX
          value: "Jumpbox is Here"
```



### Reference:

- [environment variable](#)

# Define an **environment variable** for a container (Cont.)

```
apiVersion: v1
kind: Pod
metadata:
  name: envar-demo
  labels:
    purpose: demonstrate-envvars
spec:
  containers:
    - name: envar-demo-container
      image: gcr.io/google-samples/node-hello:1.0
      env:
        - name: DEMO_GREETING
          value: "Hello from the environment"
        - name: DEMO_JUMPBOX
          value: "Jumpbox is Here"
```

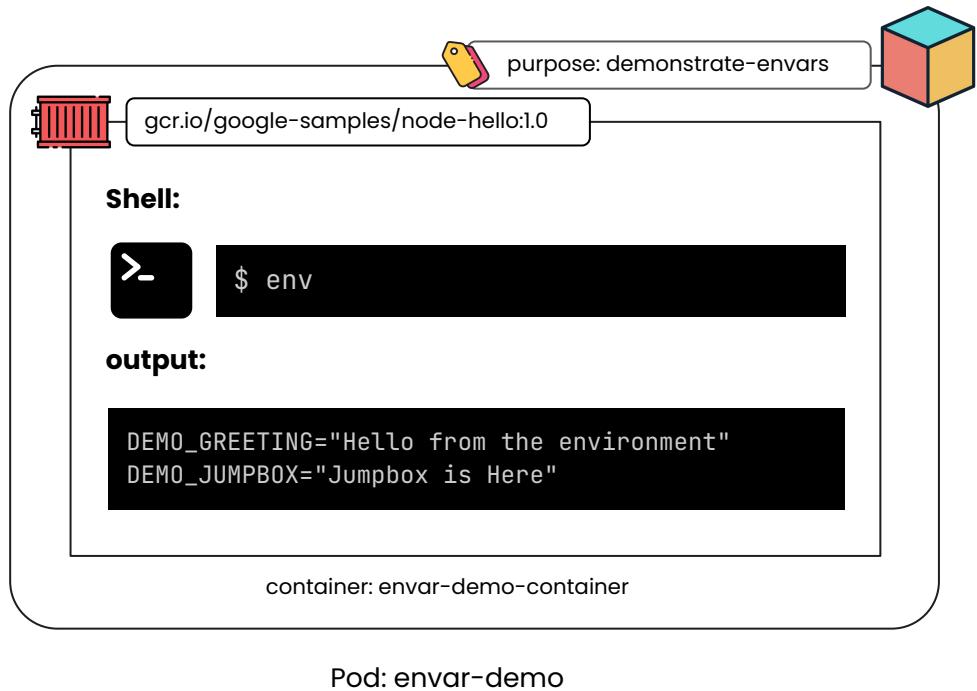


## Reference:

- [environment variable](#)

# Define an **environment variable** for a container (Cont.)

```
apiVersion: v1
kind: Pod
metadata:
  name: envar-demo
  labels:
    purpose: demonstrate-envvars
spec:
  containers:
    - name: envar-demo-container
      image: gcr.io/google-samples/node-hello:1.0
      env:
        - name: DEMO_GREETING
          value: "Hello from the environment"
        - name: DEMO_JUMPBOX
          value: "Jumpbox is Here"
```



## Reference:

- [environment variable](#)

## Define an **dependent variable** for a container

## Define an **dependent variable** for a container (Cont.)

```
spec:  
  containers:  
    ...  
    env:  
      - name: SERVICE_PORT  
        value: "80"  
      - name: SERVICE_IP  
        value: "172.17.0.1"  
      - name: UNCHANGED_REFERENCE  
        value: "$(PROTOCOL)://$(SERVICE_IP):$(SERVICE_PORT)"  
      - name: PROTOCOL  
        value: "https"  
      - name: SERVICE_ADDRESS  
        value: "$(PROTOCOL)://$(SERVICE_IP):$(SERVICE_PORT)"  
      - name: ESCAPED_REFERENCE  
        value: "$$(PROTOCOL)://$(SERVICE_IP):$(SERVICE_PORT)"
```

### Reference:

- [dependent variable](#)

## Define an **dependent variable** for a container (Cont.)

```
spec:  
  containers:  
    ...  
    env:  
      - name: SERVICE_PORT  
        value: "80"  
      - name: SERVICE_IP  
        value: "172.17.0.1"  
      - name: UNCHANGED_REFERENCE  
        value: "$(PROTOCOL)://$(SERVICE_IP):$(SERVICE_PORT)"  
      - name: PROTOCOL  
        value: "https"  
      - name: SERVICE_ADDRESS  
        value: "$(PROTOCOL)://$(SERVICE_IP):$(SERVICE_PORT)"  
      - name: ESCAPED_REFERENCE  
        value: "$$(PROTOCOL)://$(SERVICE_IP):$(SERVICE_PORT)"
```

### Reference:

- [dependent variable](#)

## Define an **dependent variable** for a container (Cont.)

```
spec:  
  containers:  
    ...  
    env:  
      - name: SERVICE_PORT  
        value: "80"  
      - name: SERVICE_IP  
        value: "172.17.0.1"  
      - name: UNCHANGED_REFERENCE  
        value: "$(PROTOCOL)://$(SERVICE_IP):$(SERVICE_PORT)"  
      - name: PROTOCOL  
        value: "https"  
      - name: SERVICE_ADDRESS  
        value: "$(PROTOCOL)://$(SERVICE_IP):$(SERVICE_PORT)"  
      - name: ESCAPED_REFERENCE  
        value: "$$(PROTOCOL)://$(SERVICE_IP):$(SERVICE_PORT)"
```

### Reference:

- [dependent variable](#)

# Define an **dependent variable** for a container (Cont.)

```
spec:  
  containers:  
    ...  
    env:  
      - name: SERVICE_PORT  
        value: "80"  
      - name: SERVICE_IP  
        value: "172.17.0.1"  
      - name: UNCHANGED_REFERENCE  
        value: "$(PROTOCOL)://$(SERVICE_IP):$(SERVICE_PORT)"  
      - name: PROTOCOL  
        value: "https"  
      - name: SERVICE_ADDRESS  
        value: "$(PROTOCOL)://$(SERVICE_IP):$(SERVICE_PORT)"  
      - name: ESCAPED_REFERENCE  
        value: "$$(PROTOCOL)://$(SERVICE_IP):$(SERVICE_PORT)"
```

## Shell:

```
$ echo  
"$UNCHANGED_REFERENCE\\n$SERVICE_ADDRESS\\n$ESCAPED_REFERENCE"
```

## Reference:

- [dependent variable](#)

# Define an **dependent variable** for a container (Cont.)

```
spec:  
  containers:  
    ...  
    env:  
      - name: SERVICE_PORT  
        value: "80"  
      - name: SERVICE_IP  
        value: "172.17.0.1"  
      - name: UNCHANGED_REFERENCE  
        value: "$(PROTOCOL)://$(SERVICE_IP):$(SERVICE_PORT)"  
      - name: PROTOCOL  
        value: "https"  
      - name: SERVICE_ADDRESS  
        value: "$(PROTOCOL)://$(SERVICE_IP):$(SERVICE_PORT)"  
      - name: ESCAPED_REFERENCE  
        value: "$$(PROTOCOL)://$(SERVICE_IP):$(SERVICE_PORT)"
```

## Shell:

```
$ echo  
"$UNCHANGED_REFERENCE\\n$SERVICE_ADDRESS\\n$ESCAPED_REFERENCE"
```

## output:

```
UNCHANGED_REFERENCE=$(PROTOCOL)://172.17.0.1:80  
SERVICE_ADDRESS=https://172.17.0.1:80  
ESCAPED_REFERENCE=$$(PROTOCOL)://172.17.0.1:80
```

## Reference:

- [dependent variable](#)

## Pod fields as values for environment variables

# Pod fields as values for environment variables (Cont.)

```
apiVersion: v1
kind: Pod
metadata:
  namespace: kai
  name: envar-demo
spec:
  containers:
  - env:
    - name: MY_POD_NAME
      valueFrom:
        fieldRef:
          fieldPath: metadata.name
    - name: MY_POD_NAMESPACE
      valueFrom:
        fieldRef:
          fieldPath: metadata.namespace
```

## Reference:

- [Expose Pod Information](#)

เจ้าของลิชีสิทธ์ อนุญาติให้ใช้สิ่งนี้เพื่อการเรียนรู้ด้านบุคคลทั่วไป และขอสงวนลิขสิทธ์ ห้ามมิให้เผยแพร่ในที่สาธารณะ ผู้อื่นเดินทาง จนถูกดำเนินคดีตามกฎหมาย

# Pod fields as values for environment variables (Cont.)

```
apiVersion: v1
kind: Pod
metadata:
  namespace: kai
  name: envar-demo
spec:
  containers:
    - env:
        - name: MY_POD_NAME
          valueFrom:
            fieldRef:
              fieldPath: metadata.name
        - name: MY_POD_NAMESPACE
          valueFrom:
            fieldRef:
              fieldPath: metadata.namespace
```

## Reference:

- [Expose Pod Information](#)

เจ้าของลิชีสิทธ์ อนุญาติให้ใช้สิ่งนี้เพื่อการเรียนรู้ด้านบุคคลทั่วไป และขอสงวนลิขสิทธ์ ห้ามมิให้เผยแพร่ในที่สาธารณะ ผู้อื่นเนต จะถูกดำเนินคดีตามกฎหมาย

# Pod fields as values for environment variables (Cont.)

```
apiVersion: v1
kind: Pod
metadata:
  namespace: kai
  name: envar-demo
spec:
  containers:
    - env:
        - name: MY_POD_NAME
          valueFrom:
            fieldRef:
              fieldPath: metadata.name
        - name: MY_POD_NAMESPACE
          valueFrom:
            fieldRef:
              fieldPath: metadata.namespace
```

## Reference:

- [Expose Pod Information](#)

เจ้าของลิชีสิทธ์ อนุญาติให้ใช้สิ่งนี้เพื่อการเรียนรู้ด้านบุคคลทั่วไป และขอสงวนลิขสิทธ์ ห้ามมิให้เผยแพร่ในที่สาธารณะ ผู้อื่นเดิน จดถูกต้องเป็นคือเจ้าของลิขสิทธ์

# Pod fields as values for environment variables (Cont.)

```
apiVersion: v1
kind: Pod
metadata:
  namespace: kai
  name: envar-demo
spec:
  containers:
    - env:
        - name: MY_POD_NAME
          valueFrom:
            fieldRef:
              fieldPath: metadata.name
        - name: MY_POD_NAMESPACE
          valueFrom:
            fieldRef:
              fieldPath: metadata.namespace
```

## Shell:

```
$ echo "$MY_POD_NAME\n$MY_POD_NAMESPACE"
```

## Reference:

- [Expose Pod Information](#)

เจ้าของลิขสิทธิ์ อนุญาติให้ใช้ส่วนนี้เพื่อการเรียนรู้ส่วนบุคคลเท่านั้น และขอสงวนลิขสิทธิ์ ห้ามมิให้เผยแพร่ในที่สาธารณะ ผู้ละเมิด จะถูกดำเนินคดีตามกฎหมาย

# Pod fields as values for environment variables (Cont.)

```
apiVersion: v1
kind: Pod
metadata:
  namespace: kai
  name: envar-demo
spec:
  containers:
    - env:
        - name: MY_POD_NAME
          valueFrom:
            fieldRef:
              fieldPath: metadata.name
        - name: MY_POD_NAMESPACE
          valueFrom:
            fieldRef:
              fieldPath: metadata.namespace
```

## Shell:

```
$ echo "$MY_POD_NAME\n$MY_POD_NAMESPACE"
```

## Output:

```
MY_POD_NAME=envar-demo
MY_POD_NAMESPACE=kai
```

## Reference:

- [Expose Pod Information](#)

เจ้าของลิขสิทธิ์ อนุญาติให้ใช้ส่วนนี้เพื่อการเรียนรู้ส่วนบุคคลเท่านั้น และขอสงวนลิขสิทธิ์ ห้ามมิให้เผยแพร่ในที่สาธารณะ ผู้ละเมิด จะถูกดำเนินคดีตามกฎหมาย

# Environment

With Real Action Simulator

[Click Here](#)



เจ้าของลิขสิทธิ์ อนุญาติให้ใช้ส่วนหนึ่งของการเรียนรู้ส่วนบุคคลเท่านั้น และขอสงวนลิขสิทธิ์ ห้ามมิให้เผยแพร่ในที่สาธารณะ ผู้อื่นเดิน จดถูกดัดแปลงโดย

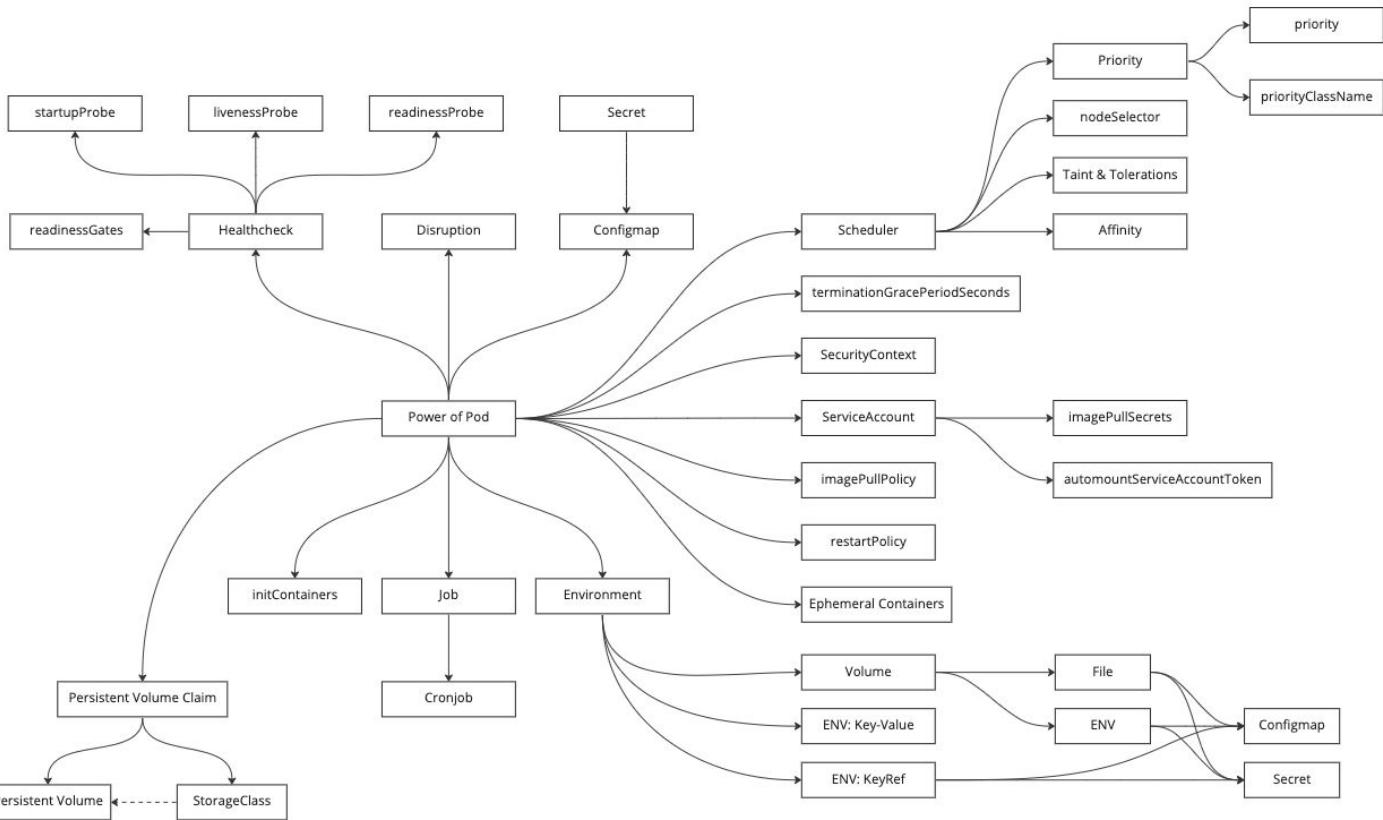
**Jumpbox®**

# ConfigMap

- Working with Kubernetes -

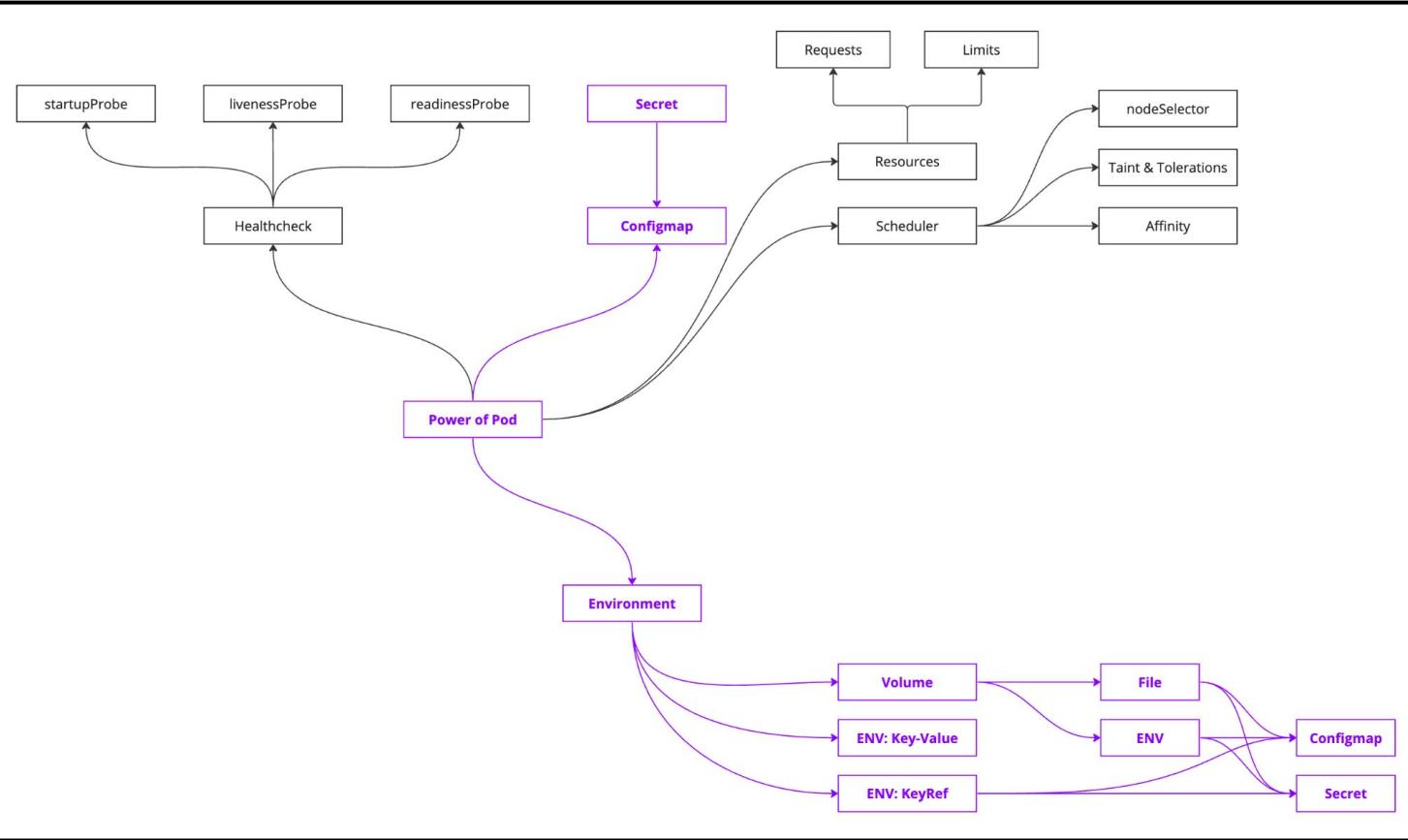
- ENV
- ENV: KeyRef
- Volume

# Power of Pod Mapping



เจ้าของลิขสิทธิ์ อนุญาตให้ใช้ส่วนนี้เพื่อการเรียนรู้ส่วนบุคคลเท่านั้น และขอสงวนลิขสิทธิ์ ห้ามมิให้เผยแพร่ในที่สาธารณะ ผู้อื่นเมตตา จะถูกดำเนินคดีตามกฎหมาย

# Power of Pod Mapping



ເຈົ້າຂອງລືສີເກີ້ວ ອຸນຍາດໃຫ້ຕົວສຸນ໌ທີ່ເກີ້ວໄດ້ຮັບຮັບມາຈົດລົງຈະບຸຄຸລາທ່ານີ້ ແລະຂອງຮັບຮັບມາຈົດລົງຈຳເປົ້າທີ່ ທັນມີໄທເກີ້ວພຽນໃໝ່ສໍາຄັນນະ ຜູ້ລະເມີດ ຈະຖຸກດຳເນີນຄືດຕິມາງງົມນາຍ

# ConfigMaps

## Configure a Pod to Use a ConfigMap

Many applications rely on configuration which is used during either application initialization or runtime. Most times, there is a requirement to adjust values assigned to configuration parameters. ConfigMaps are a Kubernetes mechanism that let you inject configuration data into application pods.

The ConfigMap concept allow you to decouple configuration artifacts from image content to keep containerized applications portable. For example, you can download and run the same container image to spin up containers for the purposes of local development, system test, or running a live end-user workload.

This page provides a series of usage examples demonstrating how to create ConfigMaps and configure Pods using data stored in ConfigMaps.

### Reference Pictures:

- [Configure a Pod to Use a ConfigMap](#)

# ConfigMaps (Cont.)

- Kubernetes objects for inject in containers with configuration data
- Configuration data can be consumed in pods in a variety of ways
- ConfigMap can be used to:
  - Populate the value of environment variables.
  - Set command-line arguments in a container.
  - Populate configuration files in a volume.

Reference Pictures:

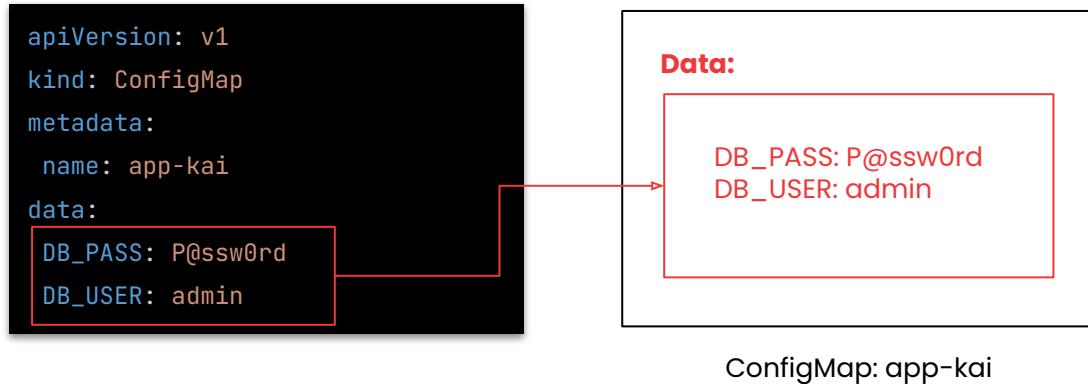
- [Kubernetes Webinar - Using ConfigMaps & Secrets](#)

# ConfigMaps Structure

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: app-kai
data:
  DB_PASS: P@ssw0rd
  DB_USER: admin
```

# ConfigMaps Structure (Cont.)

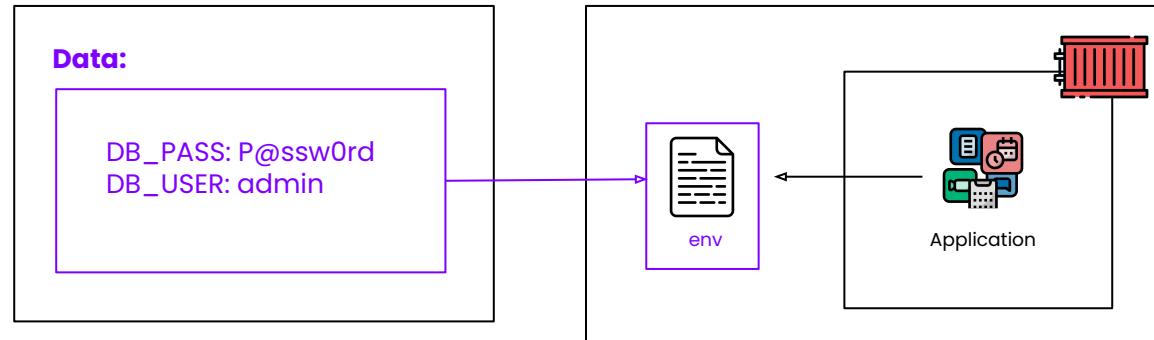
- Allow you to **decouple configuration** from image content to keep containerized applications portable



# ConfigMaps Structure (Cont.)

- Allow you to decouple configuration from image content to keep containerized applications portable
- It is used to setting configuration **data separately** from application code

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: app-kai
data:
  DB_PASS: P@ssw0rd
  DB_USER: admin
```

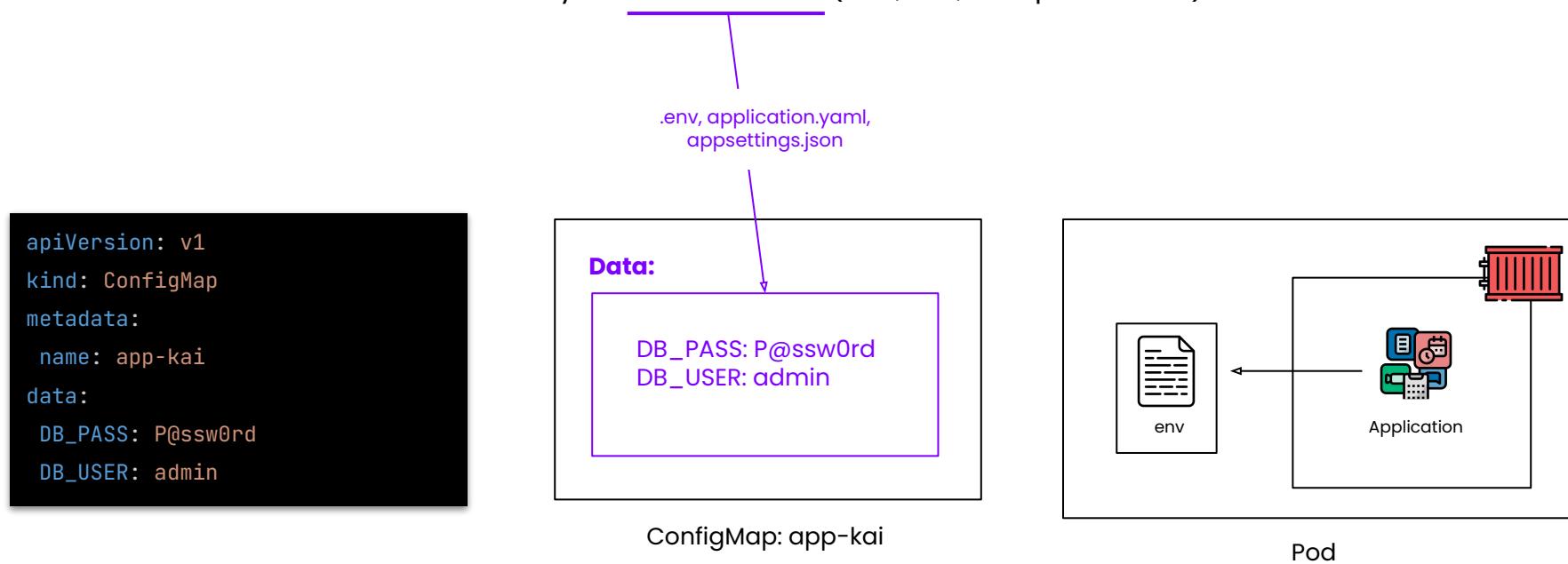


ConfigMap: app-kai

Pod

# ConfigMaps Structure (Cont.)

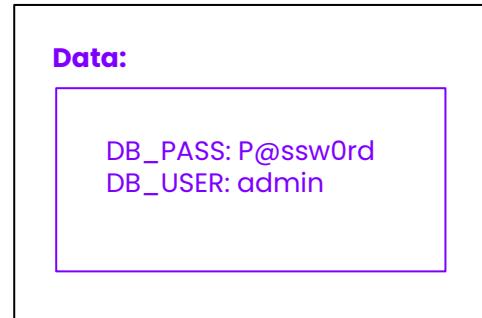
- Allow you to decouple configuration from image content to keep containerized applications portable
- It is used to setting configuration data separately from application code
- It is like `.env` file which can vary on environments (dev, uat, and production)



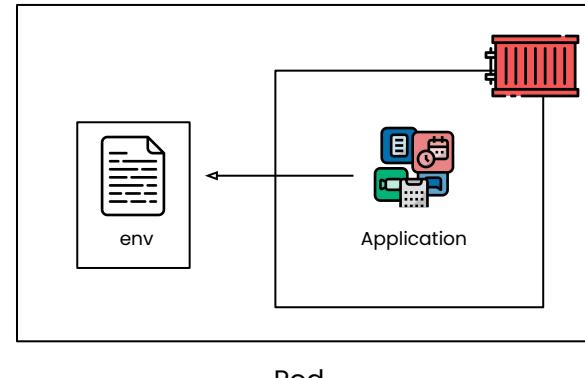
# ConfigMaps Structure (Cont.)

- Allow you to decouple configuration from image content to keep containerized applications portable
- It is used to setting configuration data separately from application code
- It is like .env file which can vary on environments (dev, uat, and production)
- Does **not support large chunks of data**.

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: app-kai
data:
  DB_PASS: P@ssw0rd
  DB_USER: admin
```



ConfigMap: app-kai



Pod

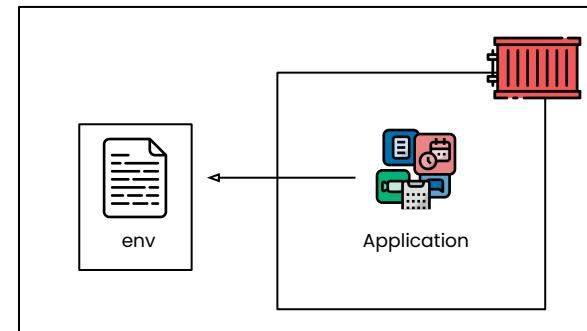
# ConfigMaps Structure (Cont.)

- Allow you to decouple configuration from image content to keep containerized applications portable
- It is used to setting configuration data separately from application code
- It is like .env file which can vary on environments (dev, uat, and production)
- Does not support large chunks of data.
- The size of configmap cannot exceed 1MB.

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: app-kai
data:
  DB_PASS: P@ssw0rd
  DB_USER: admin
```



ConfigMap: app-kai

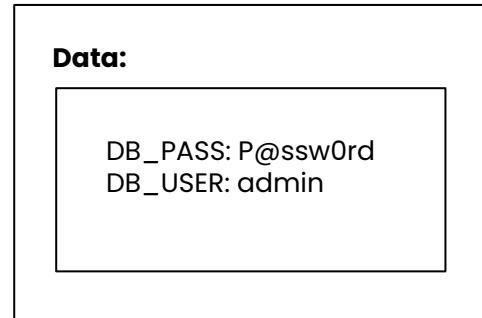


Pod

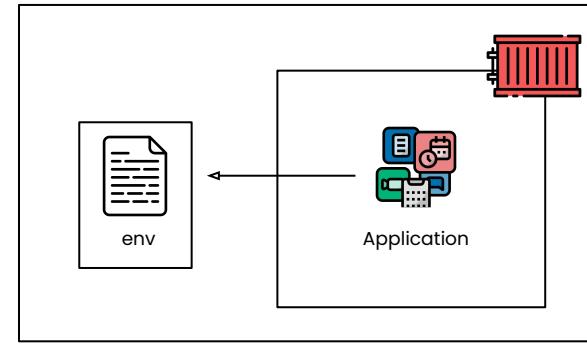
# ConfigMaps Structure (Cont.)

- Allow you to **decouple configuration** from image content to keep containerized applications portable
- It is used to setting configuration **data separately** from application code
- It is like **.env file** which can vary on **environments** (dev, uat, and production)
- Does **not support large** chunks of **data**.
- The **size** of configmap **cannot exceed 1MB**.

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: app-kai
data:
  DB_PASS: P@ssw0rd
  DB_USER: admin
```



ConfigMap: app-kai



Pod

# Environment Variable

- Configure a Pod to Use a ConfigMap -

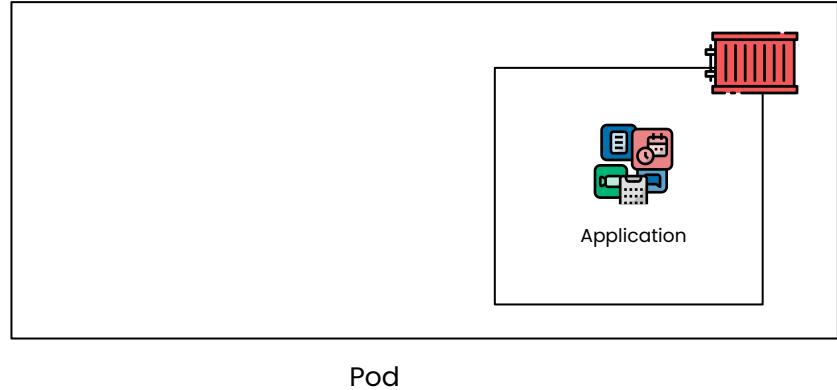
# Configure a Pod to Use a ConfigMap - Environment Variable

# Configure a Pod to Use a ConfigMap - Environment Variable (Cont.)

```
apiVersion: v1
kind: Pod
metadata:
  name: dapi-test-pod
spec:
  containers:
    - name: test-container
      image: registry.k8s.io/busybox
      command: [ "/bin/echo", "$(SPECIAL_LEVEL_KEY) $(SPECIAL_TYPE_KEY)" ]
      env:
        - name: SPECIAL_LEVEL_KEY
          valueFrom:
            configMapKeyRef:
              name: special-config
              key: SPECIAL_LEVEL
        - name: SPECIAL_TYPE_KEY
          valueFrom:
            configMapKeyRef:
              name: special-type
              key: SPECIAL_TYPE
  restartPolicy: Never
```

# Configure a Pod to Use a ConfigMap - Environment Variable (Cont.)

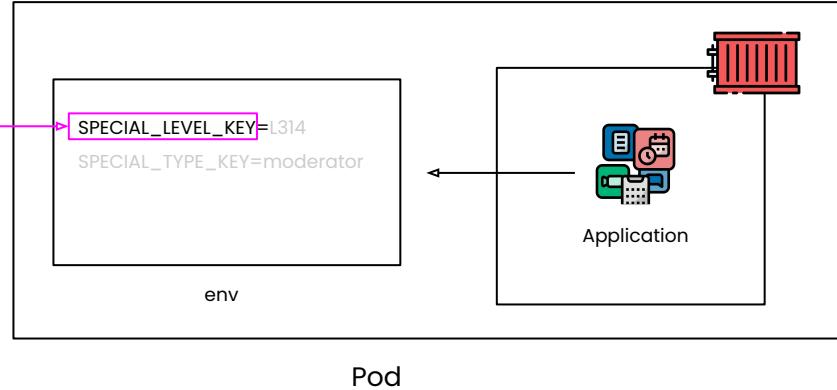
```
apiVersion: v1
kind: Pod
metadata:
  name: dapi-test-pod
spec:
  containers:
    - name: test-container
      image: registry.k8s.io/busybox
      command: [ "/bin/echo", "$(SPECIAL_LEVEL_KEY) $(SPECIAL_TYPE_KEY)" ]
    env:
      - name: SPECIAL_LEVEL_KEY
        valueFrom:
          configMapKeyRef:
            name: special-config
            key: SPECIAL_LEVEL
      - name: SPECIAL_TYPE_KEY
        valueFrom:
          configMapKeyRef:
            name: special-type
            key: SPECIAL_TYPE
  restartPolicy: Never
```



Pod

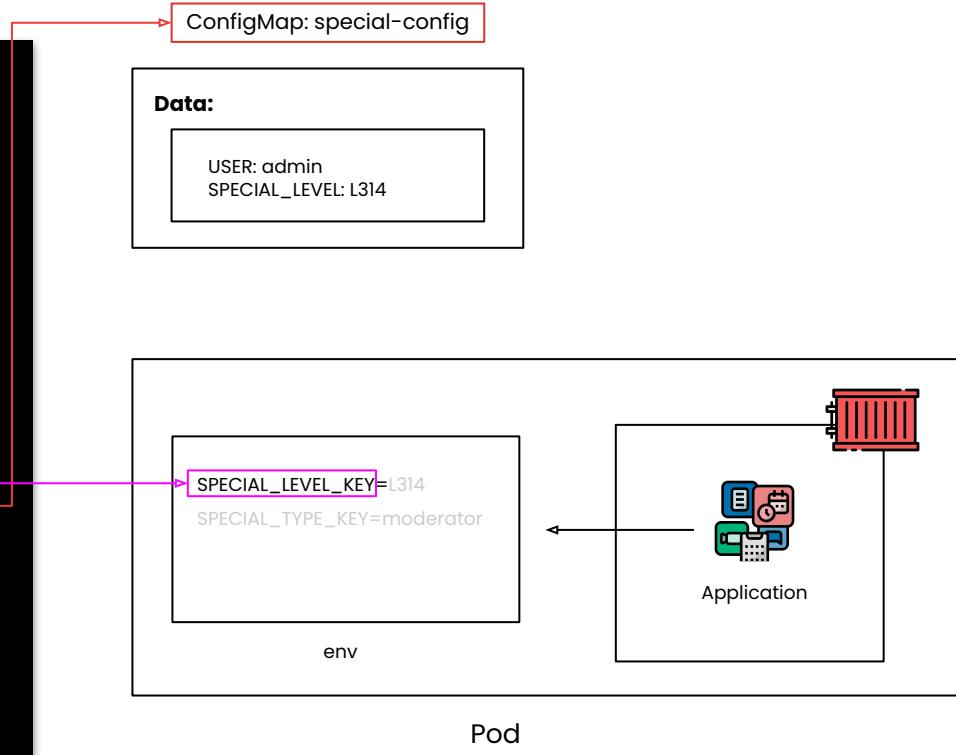
# Configure a Pod to Use a ConfigMap - Environment Variable (Cont.)

```
apiVersion: v1
kind: Pod
metadata:
  name: dapi-test-pod
spec:
  containers:
    - name: test-container
      image: registry.k8s.io/busybox
      command: [ "/bin/echo", "$(SPECIAL_LEVEL_KEY) $(SPECIAL_TYPE_KEY)" ]
    env:
      - name: SPECIAL_LEVEL_KEY
        valueFrom:
          configMapKeyRef:
            name: special-config
            key: SPECIAL_LEVEL
      - name: SPECIAL_TYPE_KEY
        valueFrom:
          configMapKeyRef:
            name: special-type
            key: SPECIAL_TYPE
  restartPolicy: Never
```



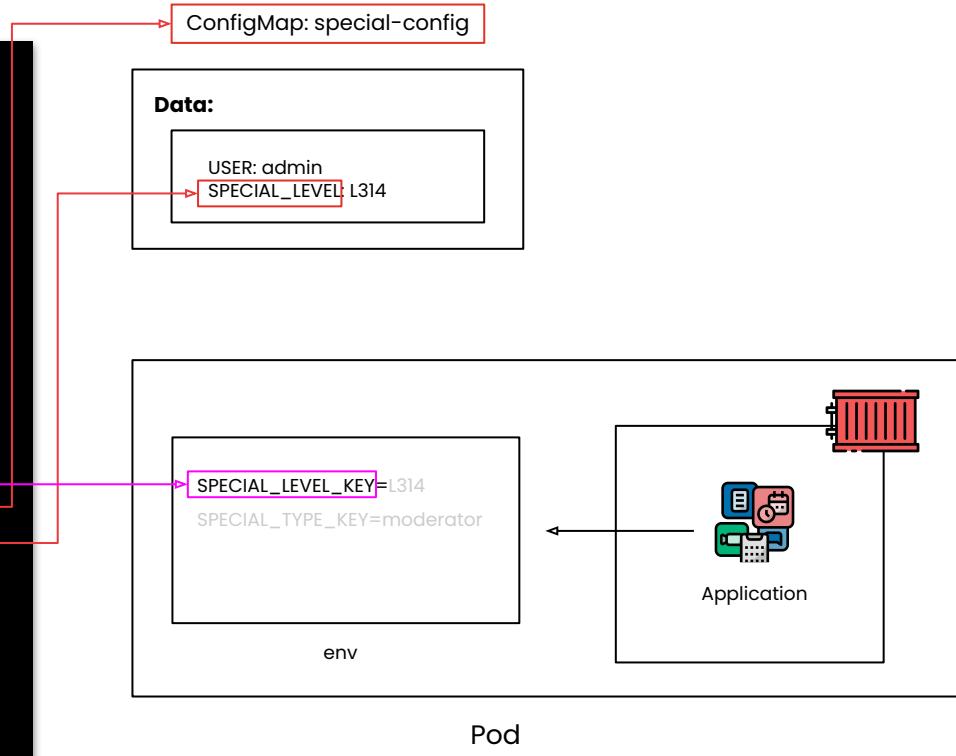
# Configure a Pod to Use a ConfigMap - Environment Variable (Cont.)

```
apiVersion: v1
kind: Pod
metadata:
  name: dapi-test-pod
spec:
  containers:
    - name: test-container
      image: registry.k8s.io/busybox
      command: [ "/bin/echo", "$(SPECIAL_LEVEL_KEY) $(SPECIAL_TYPE_KEY)" ]
    env:
      - name: SPECIAL_LEVEL_KEY
        valueFrom:
          configMapKeyRef:
            name: special-config
            key: SPECIAL_LEVEL
      - name: SPECIAL_TYPE_KEY
        valueFrom:
          configMapKeyRef:
            name: special-type
            key: SPECIAL_TYPE
  restartPolicy: Never
```



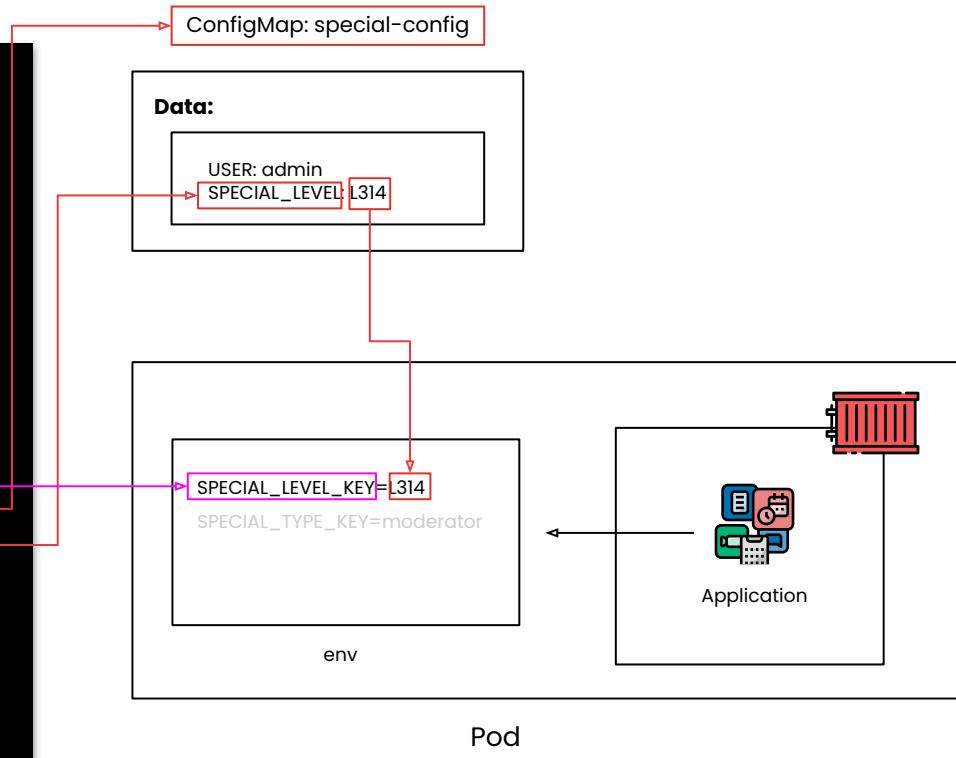
# Configure a Pod to Use a ConfigMap - Environment Variable (Cont.)

```
apiVersion: v1
kind: Pod
metadata:
  name: dapi-test-pod
spec:
  containers:
    - name: test-container
      image: registry.k8s.io/busybox
      command: [ "/bin/echo", "$(SPECIAL_LEVEL_KEY) $(SPECIAL_TYPE_KEY)" ]
    env:
      - name: SPECIAL_LEVEL_KEY
        valueFrom:
          configMapKeyRef:
            name: special-config
            key: SPECIAL_LEVEL
      - name: SPECIAL_TYPE_KEY
        valueFrom:
          configMapKeyRef:
            name: special-type
            key: SPECIAL_TYPE
  restartPolicy: Never
```



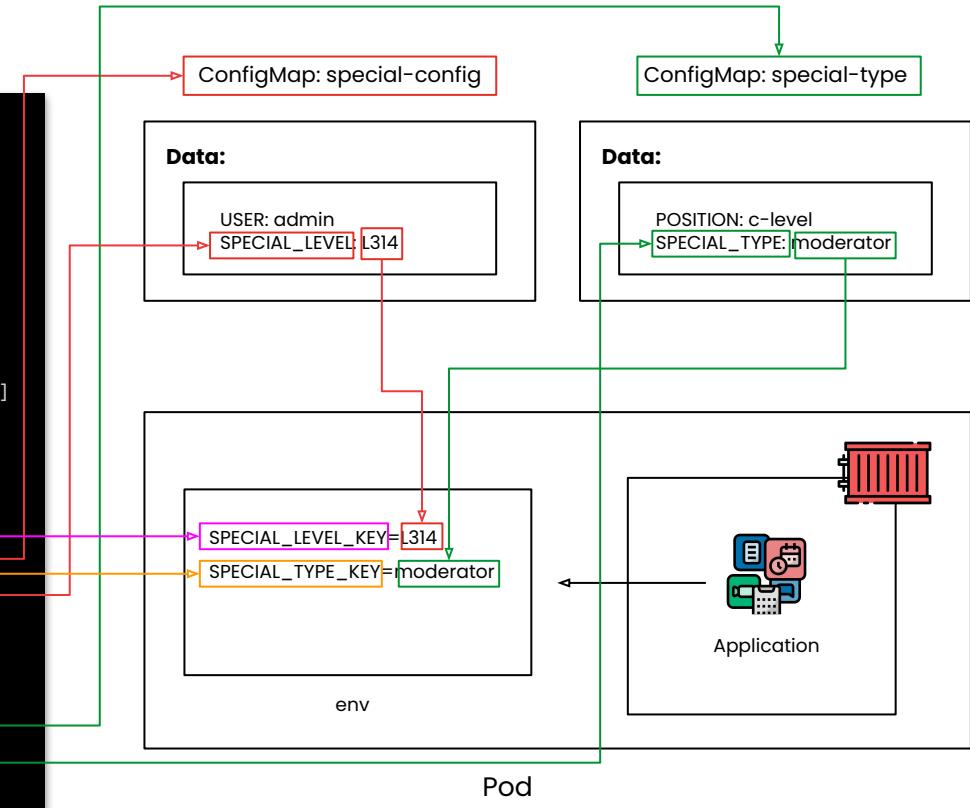
# Configure a Pod to Use a ConfigMap - Environment Variable (Cont.)

```
apiVersion: v1
kind: Pod
metadata:
  name: dapi-test-pod
spec:
  containers:
    - name: test-container
      image: registry.k8s.io/busybox
      command: [ "/bin/echo", "$(SPECIAL_LEVEL_KEY) $(SPECIAL_TYPE_KEY)" ]
    env:
      - name: SPECIAL_LEVEL_KEY
        valueFrom:
          configMapKeyRef:
            name: special-config
            key: SPECIAL_LEVEL
      - name: SPECIAL_TYPE_KEY
        valueFrom:
          configMapKeyRef:
            name: special-type
            key: SPECIAL_TYPE
  restartPolicy: Never
```



# Configure a Pod to Use a ConfigMap - Environment Variable (Cont.)

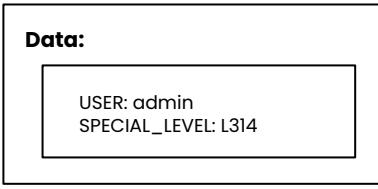
```
apiVersion: v1
kind: Pod
metadata:
  name: dapi-test-pod
spec:
  containers:
    - name: test-container
      image: registry.k8s.io/busybox
      command: [ "/bin/echo", "$(SPECIAL_LEVEL_KEY) $(SPECIAL_TYPE_KEY)" ]
    env:
      - name: SPECIAL_LEVEL_KEY
        valueFrom:
          configMapKeyRef:
            name: special-config
            key: SPECIAL_LEVEL
      - name: SPECIAL_TYPE_KEY
        valueFrom:
          configMapKeyRef:
            name: special-type
            key: SPECIAL_TYPE
  restartPolicy: Never
```



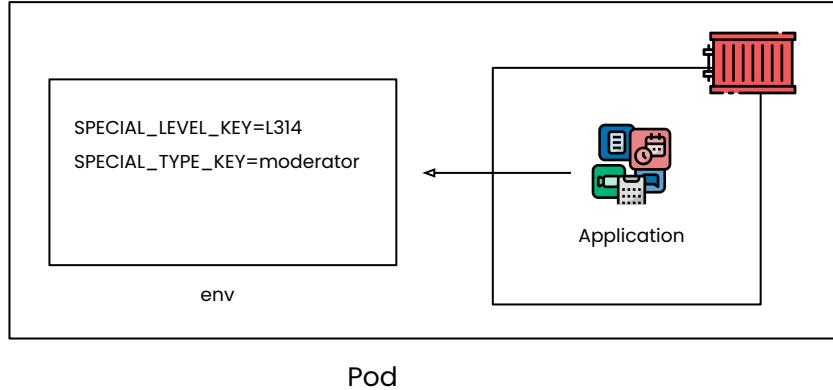
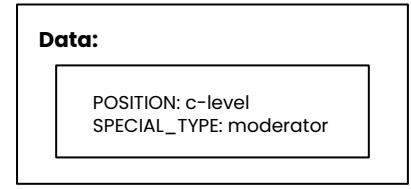
# Configure a Pod to Use a ConfigMap - Environment Variable (Cont.)

```
apiVersion: v1
kind: Pod
metadata:
  name: dapi-test-pod
spec:
  containers:
    - name: test-container
      image: registry.k8s.io/busybox
      command: [ "/bin/echo", "$(SPECIAL_LEVEL_KEY) $(SPECIAL_TYPE_KEY)" ]
      env:
        - name: SPECIAL_LEVEL_KEY
          valueFrom:
            configMapKeyRef:
              name: special-config
              key: SPECIAL_LEVEL
        - name: SPECIAL_TYPE_KEY
          valueFrom:
            configMapKeyRef:
              name: special-type
              key: SPECIAL_TYPE
  restartPolicy: Never
```

ConfigMap: special-config



ConfigMap: special-type



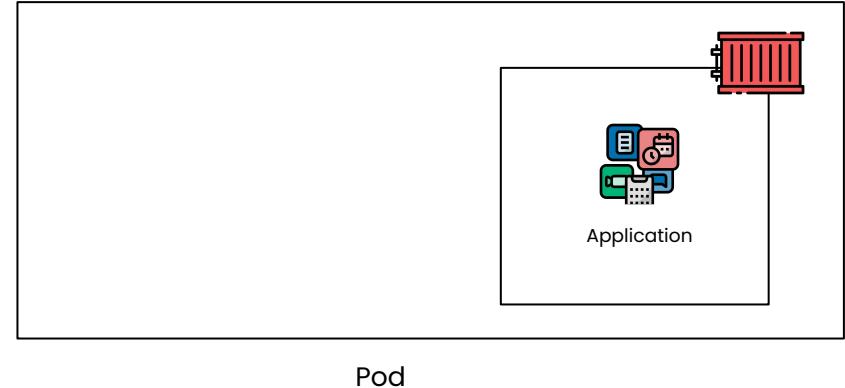
# All key-value as Environment

- Configure a Pod to Use a ConfigMap -

# Configure a Pod to Use a ConfigMap - All key-value as Environment

# Configure a Pod to Use a ConfigMap - All key-value as Environment (Cont.)

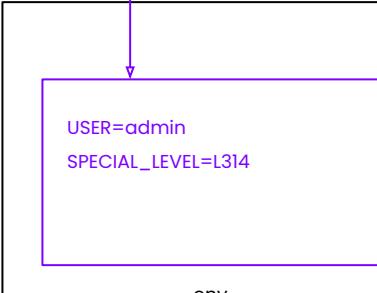
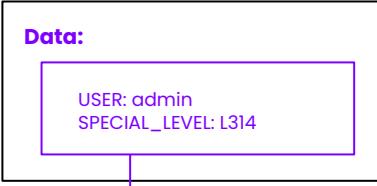
```
apiVersion: v1
kind: Pod
metadata:
  name: dapi-test-pod
spec:
  containers:
    - name: test-container
      image: registry.k8s.io/busybox
      command: [ "/bin/sh", "-c", "env" ]
      envFrom:
        - configMapRef:
            name: special-config
  restartPolicy: Never
```



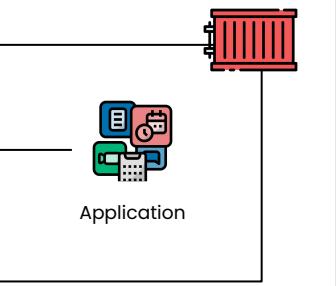
# Configure a Pod to Use a ConfigMap - All key-value as Environment (Cont.)

```
apiVersion: v1
kind: Pod
metadata:
  name: dapi-test-pod
spec:
  containers:
    - name: test-container
      image: registry.k8s.io/busybox
      command: [ "/bin/sh", "-c", "env" ]
      envFrom:
        - configMapRef:
            name: special-config
  restartPolicy: Never
```

ConfigMap: special-config



Pod



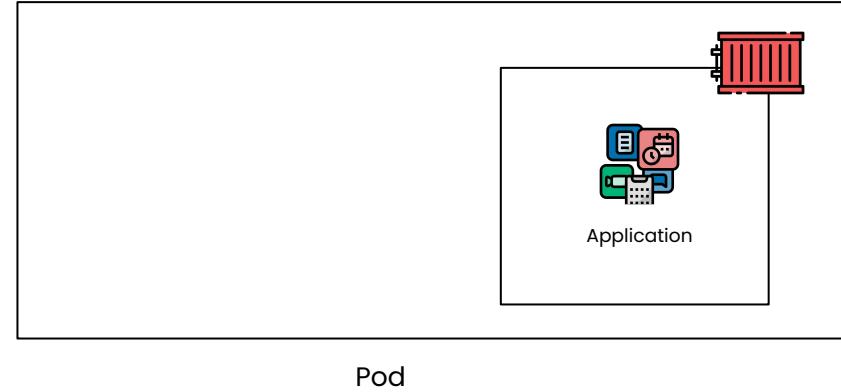
# All key-value as file(s)

- Configure a Pod to Use a ConfigMap -

## Configure a Pod to Use a ConfigMap - All key-value as file(s)

# Configure a Pod to Use a ConfigMap - All key-value as file(s) (Cont.)

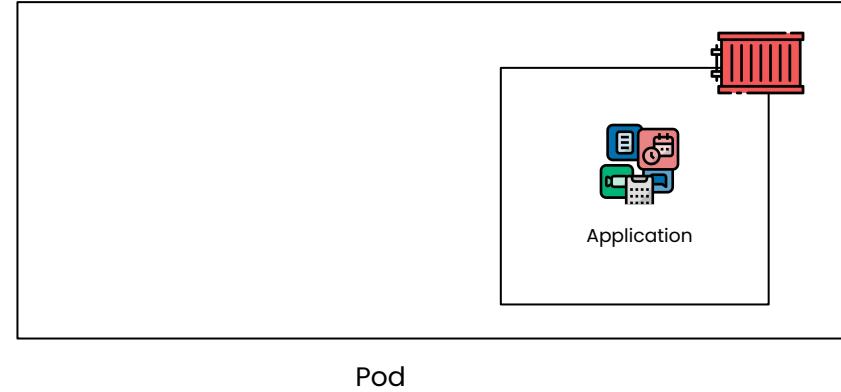
```
apiVersion: v1
kind: Pod
metadata:
  name: dapi-test-pod
spec:
  containers:
    - name: test-container
      image: registry.k8s.io/busybox
      command: [ "/bin/sh", "-c", "ls /etc/config/" ]
    volumeMounts:
      - name: config-volume
        mountPath: /etc/config
  volumes:
    - name: config-volume
      configMap:
        name: special-config
  restartPolicy: Never
```



Pod

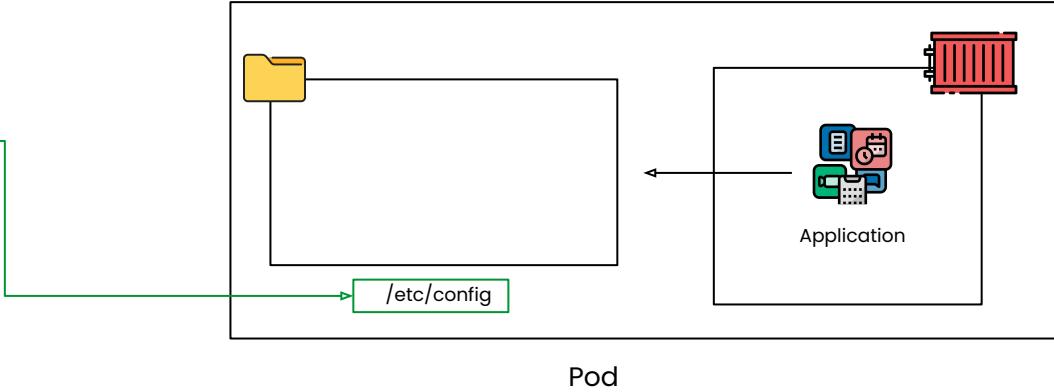
# Configure a Pod to Use a ConfigMap - All key-value as file(s) (Cont.)

```
apiVersion: v1
kind: Pod
metadata:
  name: dapi-test-pod
spec:
  containers:
    - name: test-container
      image: registry.k8s.io/busybox
      command: [ "/bin/sh", "-c", "ls /etc/config/" ]
      volumeMounts:
        - name: config-volume
          mountPath: /etc/config
  volumes:
    - name: config-volume
      configMap:
        name: special-config
  restartPolicy: Never
```

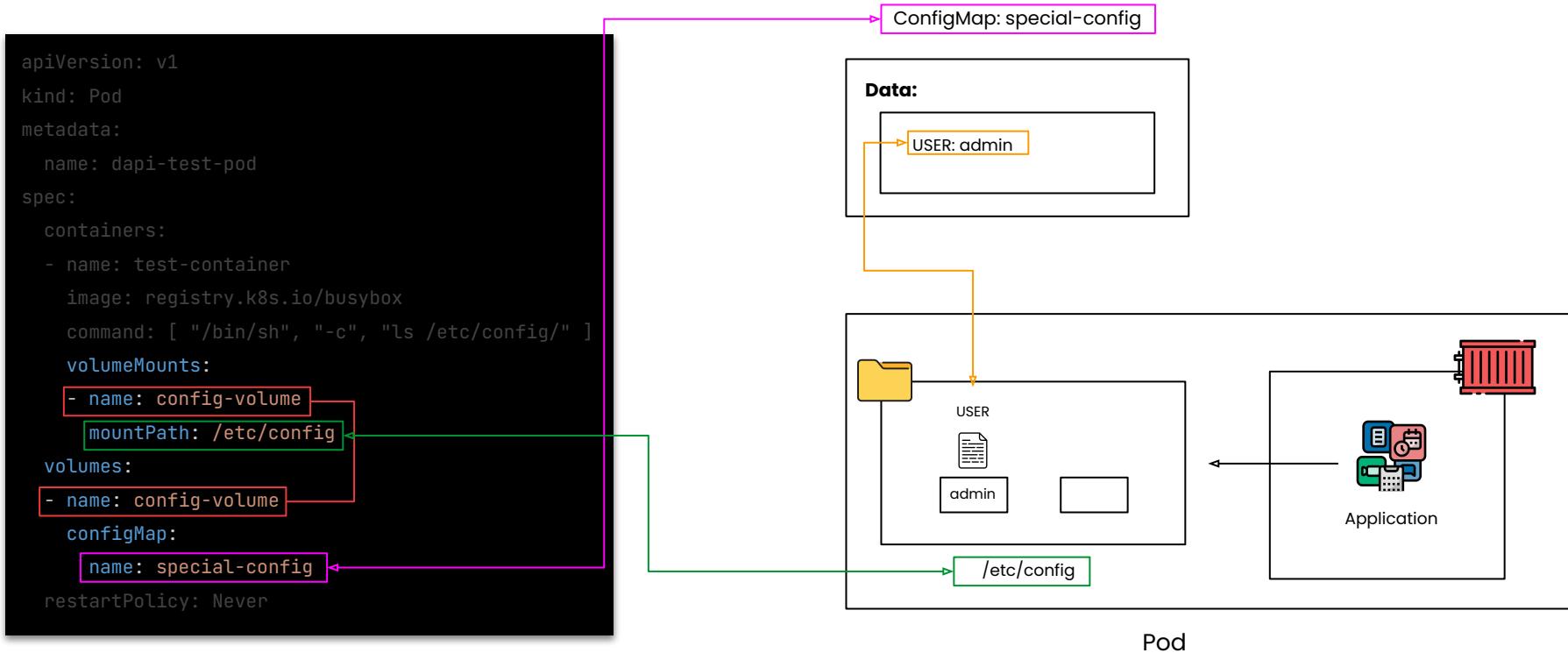


# Configure a Pod to Use a ConfigMap - All key-value as file(s) (Cont.)

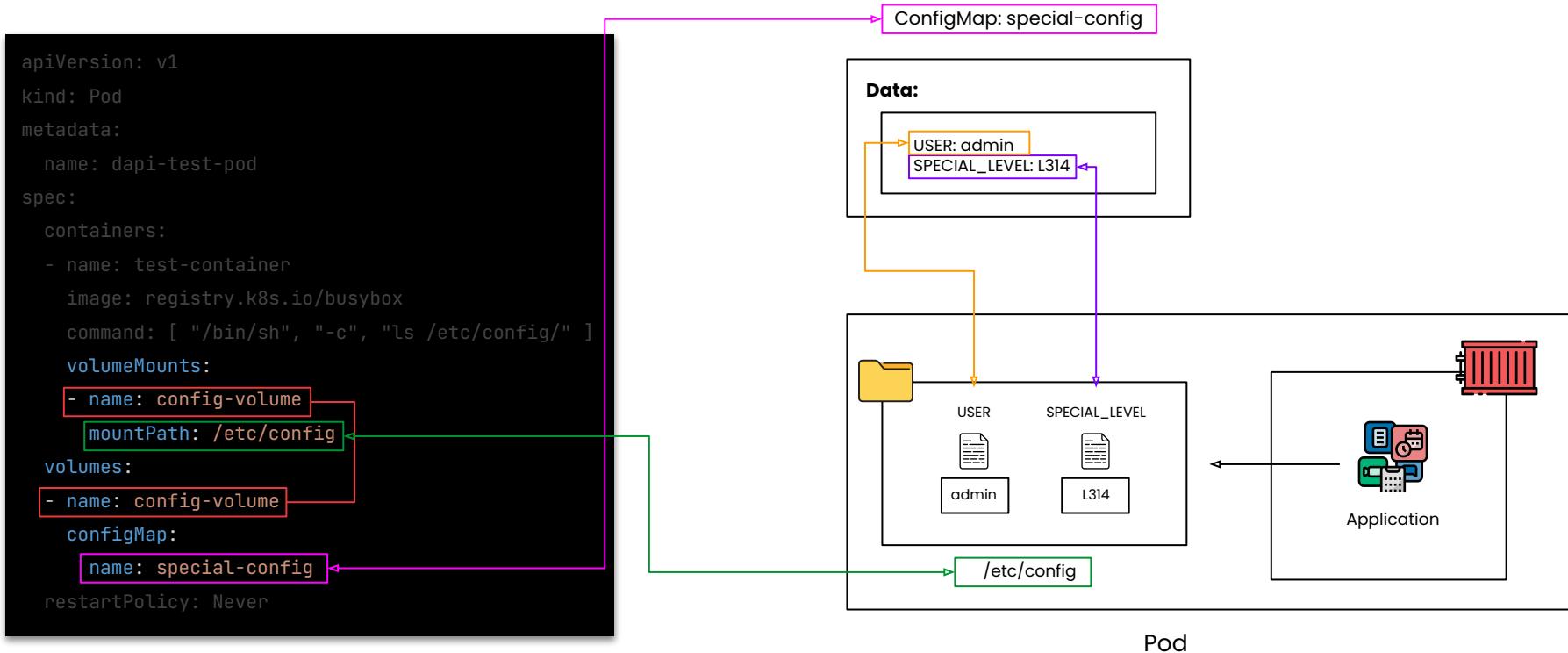
```
apiVersion: v1
kind: Pod
metadata:
  name: dapi-test-pod
spec:
  containers:
    - name: test-container
      image: registry.k8s.io/busybox
      command: [ "/bin/sh", "-c", "ls /etc/config/" ]
      volumeMounts:
        - name: config-volume
          mountPath: /etc/config
  volumes:
    - name: config-volume
      configMap:
        name: special-config
  restartPolicy: Never
```



# Configure a Pod to Use a ConfigMap - All key-value as file(s) (Cont.)



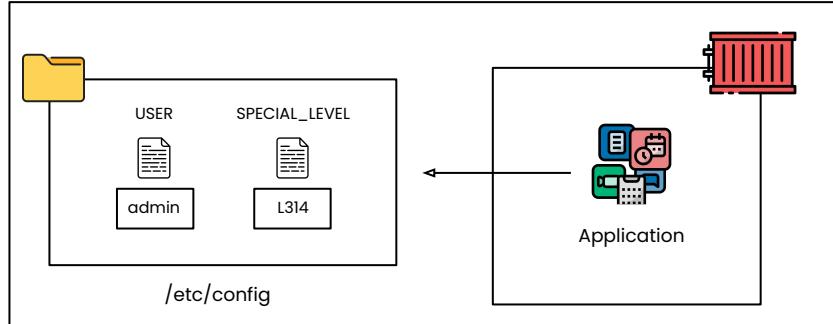
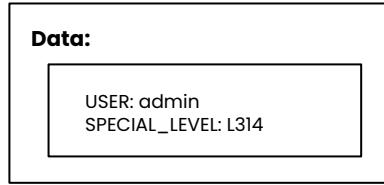
# Configure a Pod to Use a ConfigMap - All key-value as file(s) (Cont.)



# Configure a Pod to Use a ConfigMap - All key-value as file(s) (Cont.)

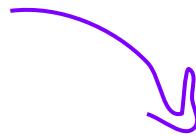
```
apiVersion: v1
kind: Pod
metadata:
  name: dapi-test-pod
spec:
  containers:
    - name: test-container
      image: registry.k8s.io/busybox
      command: [ "/bin/sh", "-c", "ls /etc/config/" ]
  volumeMounts:
    - name: config-volume
      mountPath: /etc/config
  volumes:
    - name: config-volume
      configMap:
        name: special-config
  restartPolicy: Never
```

ConfigMap: special-config



# Action Dojo

- Kata: small exercise to do by yourself



## Activity Time !!

- Kata -

# Configmap

With Real Action Simulator

[Click Here](#)



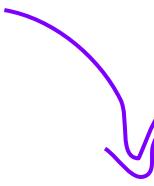
# Action Dojo

- Kata: small exercise to do by yourself



## Activity Time !!

- kata -



## Then Break !!

- 10 minutes -

# Secret

- Working with Kubernetes -

- ENV
- ENV: KeyRef
- Volume

# Secret

เจ้าของลิขสิทธิ์ อนุญาติให้ใช้รับสุนั�น์เพื่อการเรียนรู้ส่วนบุคคลเท่านั้น และขอสงวนลิขสิทธิ์ ห้ามมิให้เผยแพร่ในที่สาธารณะ ผู้ละเอียด จะถูกดำเนินคดีตามกฎหมาย

**Jumpbox®**

## Secret (Cont.)

# Uses for Secrets

You can use Secrets for purposes such as the following:

- Set environment variables for a container.
- Provide credentials such as SSH keys or passwords to Pods.
- Allow the kubelet to pull container images from private registries.

The Kubernetes control plane also uses Secrets; for example, bootstrap token Secrets are a mechanism to help automate node registration.

Reference Pictures:

- [Secrets](#)

## Secret (Cont.)

- Kubernetes secret objects let you store and **manage sensitive information**, such as passwords, OAuth tokens, and ssh keys.

```
apiVersion: v1
kind: Secret
metadata:
  name: mysql
data:
  username: YWRtaW4=
  password: UEBzc3cwcmQ=
```

## Secret (Cont.)

- Kubernetes secret objects let you store and manage sensitive information, such as passwords, OAuth tokens, and ssh keys.
- Secrets are [similar to ConfigMaps](#) but are specifically intended to [hold confidential data](#).

```
apiVersion: v1
kind: Secret
metadata:
  name: mysql
data:
  username: YWRtaW4=
  password: UEBzc3cwcmQ=
```

## Secret (Cont.)

### **Caution:**

Kubernetes Secrets are, by default, stored unencrypted in the API server's underlying data store (etcd). Anyone with API access can retrieve or modify a Secret, and so can anyone with access to etcd. Additionally, anyone who is authorized to create a Pod in a namespace can use that access to read any Secret in that namespace; this includes indirect access such as the ability to create a Deployment.

# Secret (Cont.)

## **Caution:**

Kubernetes Secrets are, by default, stored unencrypted in the API server's underlying data store (etcd). Anyone with API access can retrieve or modify a Secret, and so can anyone with access to etcd. Additionally, anyone who is authorized to create a Pod in a namespace can use that access to read any Secret in that namespace; this includes indirect access such as the ability to create a Deployment.

## **In order to safely use Secrets, take at least the following steps:**

1. [Enable Encryption at Rest](#) for Secrets.
2. [Enable or configure RBAC rules](#) with least-privilege access to Secrets.
3. Restrict Secret access to specific containers.
4. [Consider using external Secret store providers](#).

For more guidelines to manage and improve the security of your Secrets, refer to [Good practices for Kubernetes Secrets](#).

# Environment Variable

- Configure a Pod to Use a Secret -

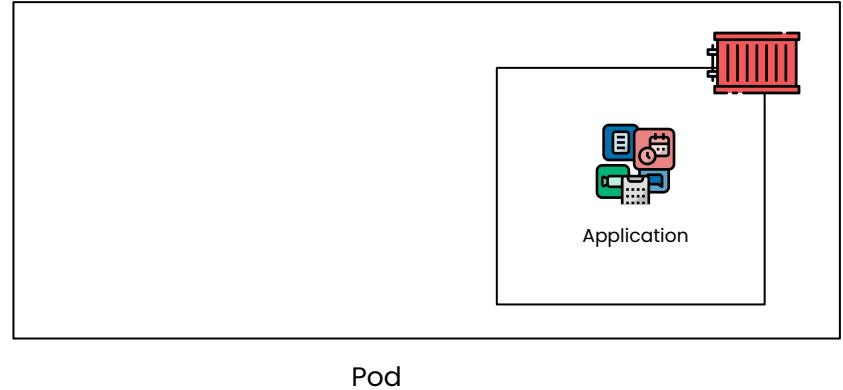
# Configure a Pod to Use a Secret – Environment Variable

## Configure a Pod to Use a Secret – Environment Variable (Cont.)

```
apiVersion: v1
kind: Pod
metadata:
  name: envvars-multiple-secrets
spec:
  containers:
    - name: envvars-test-container
      image: nginx
      env:
        - name: BACKEND_USERNAME
          valueFrom:
            secretKeyRef:
              name: backend-user
              key: backend-username
        - name: DB_USERNAME
          valueFrom:
            secretKeyRef:
              name: db-user
              key: db-username
```

# Configure a Pod to Use a Secret – Environment Variable (Cont.)

```
apiVersion: v1
kind: Pod
metadata:
  name: envvars-multiple-secrets
spec:
  containers:
    - name: envvars-test-container
      image: nginx
      env:
        - name: BACKEND_USERNAME
          valueFrom:
            secretKeyRef:
              name: backend-user
              key: backend-username
        - name: DB_USERNAME
          valueFrom:
            secretKeyRef:
              name: db-user
              key: db-username
```



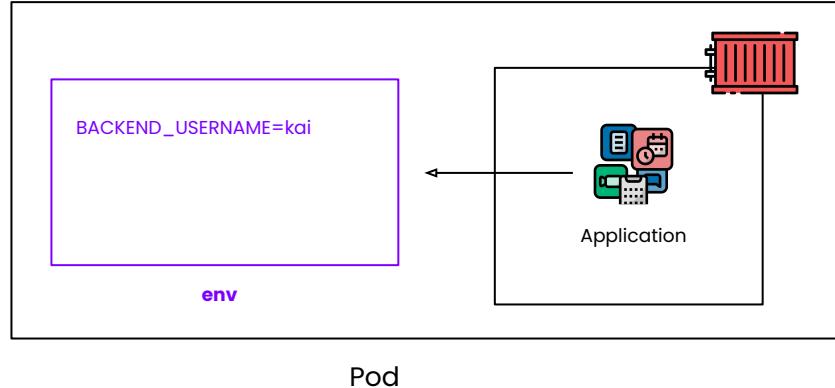
# Configure a Pod to Use a Secret – Environment Variable (Cont.)

```
apiVersion: v1
kind: Pod
metadata:
  name: envvars-multiple-secrets
spec:
  containers:
    - name: envvars-test-container
      image: nginx
      env:
        - name: BACKEND_USERNAME
          valueFrom:
            secretKeyRef:
              name: backend-user
              key: backend-username
        - name: DB_USERNAME
          valueFrom:
            secretKeyRef:
              name: db-user
              key: db-username
```

Secret: backend-user

## Data:

```
backend-role: amVhYgo=
backend-username: a2FpCg==
```



# Configure a Pod to Use a Secret – Environment Variable (Cont.)

```
apiVersion: v1
kind: Pod
metadata:
  name: envvars-multiple-secrets
spec:
  containers:
    - name: envvars-test-container
      image: nginx
      env:
        - name: BACKEND_USERNAME
          valueFrom:
            secretKeyRef:
              name: backend-user
              key: backend-username
        - name: DB_USERNAME
          valueFrom:
            secretKeyRef:
              name: db-user
              key: db-username
```

Secret: backend-user

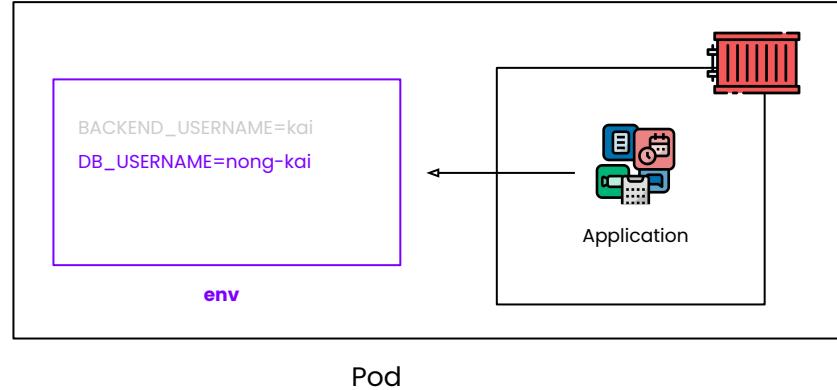
## Data:

```
backend-role: amVhYgo=
backend-username: a2FpCg==
```

Secret: db-user

## Data:

```
db-role: cGVkCg==
db-username: bm9uZyIrYWKK
```



# Configure a Pod to Use a Secret – Environment Variable (Cont.)

```
apiVersion: v1
kind: Pod
metadata:
  name: envvars-multiple-secrets
spec:
  containers:
    - name: envvars-test-container
      image: nginx
      env:
        - name: BACKEND_USERNAME
          valueFrom:
            secretKeyRef:
              name: backend-user
              key: backend-username
        - name: DB_USERNAME
          valueFrom:
            secretKeyRef:
              name: db-user
              key: db-username
```

Secret: backend-user

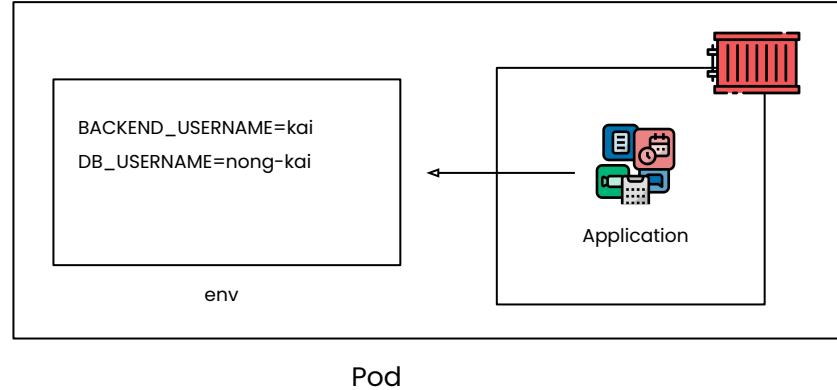
Data:

```
backend-role: amVhYgo=
backend-username: a2FpCg==
```

Secret: db-user

Data:

```
db-role: cGVkCg==
db-username: bm9uZylrYWKK
```



# All key-value as Environment

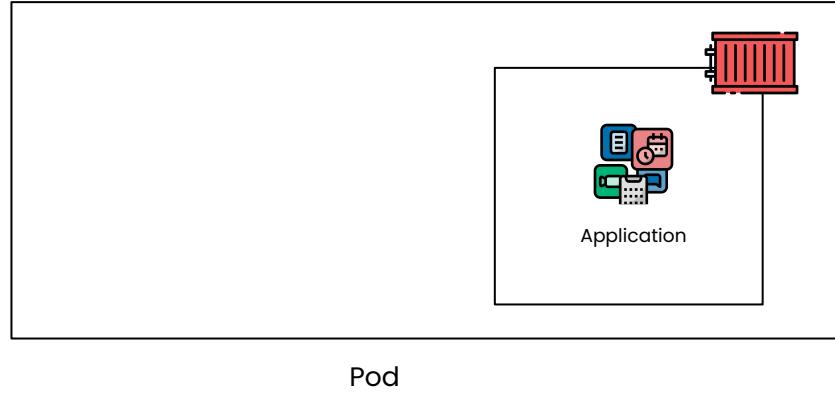
- Configure a Pod to Use a Secret -

## Configure a Pod to Use a Secret – All key-value (Cont.)

```
apiVersion: v1
kind: Pod
metadata:
  name: envfrom-secret
spec:
  containers:
  - name: envvars-test-container
    image: nginx
    envFrom:
    - secretRef:
        name: test-secret
```

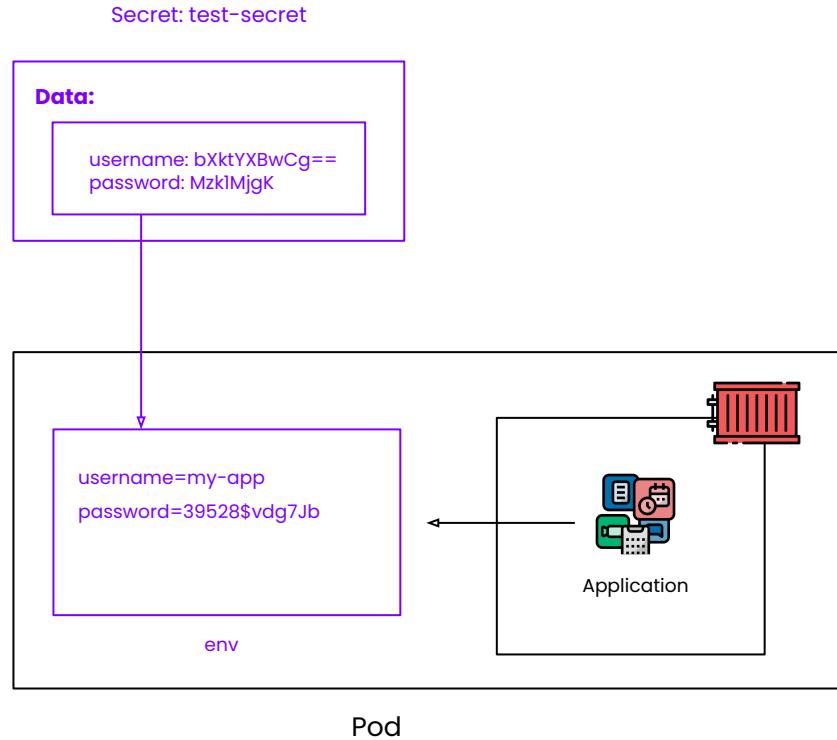
## Configure a Pod to Use a Secret – All key-value (Cont.)

```
apiVersion: v1
kind: Pod
metadata:
  name: envfrom-secret
spec:
  containers:
    - name: envvars-test-container
      image: nginx
      envFrom:
        - secretRef:
            name: test-secret
```



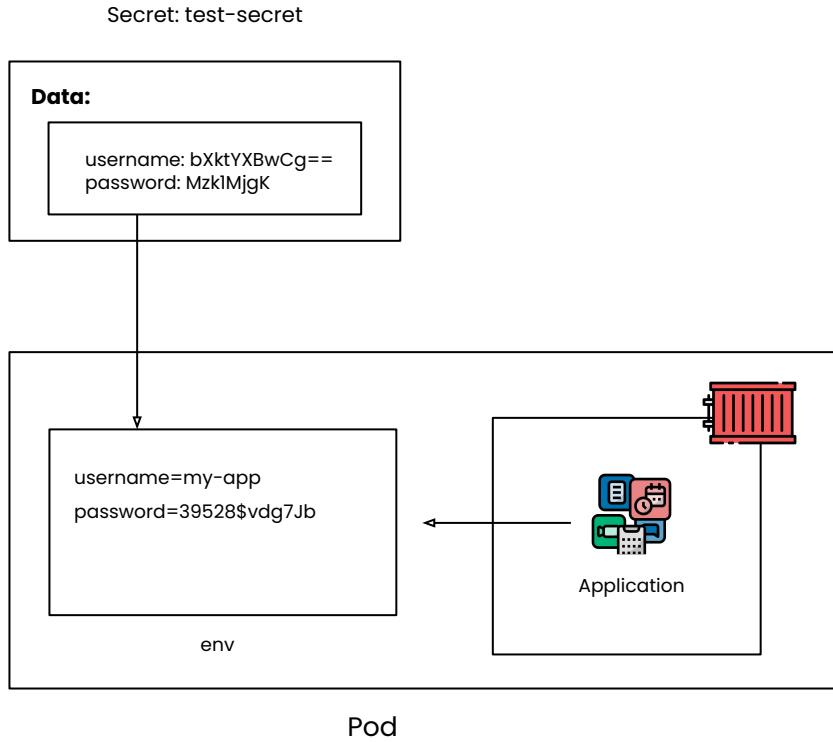
## Configure a Pod to Use a Secret – All key-value (Cont.)

```
apiVersion: v1
kind: Pod
metadata:
  name: envfrom-secret
spec:
  containers:
    - name: envvars-test-container
      image: nginx
      envFrom:
        - secretRef:
            name: test-secret
```



## Configure a Pod to Use a Secret – All key-value (Cont.)

```
apiVersion: v1
kind: Pod
metadata:
  name: envfrom-secret
spec:
  containers:
    - name: envvars-test-container
      image: nginx
      envFrom:
        - secretRef:
            name: test-secret
```



# All key-value as file(s)

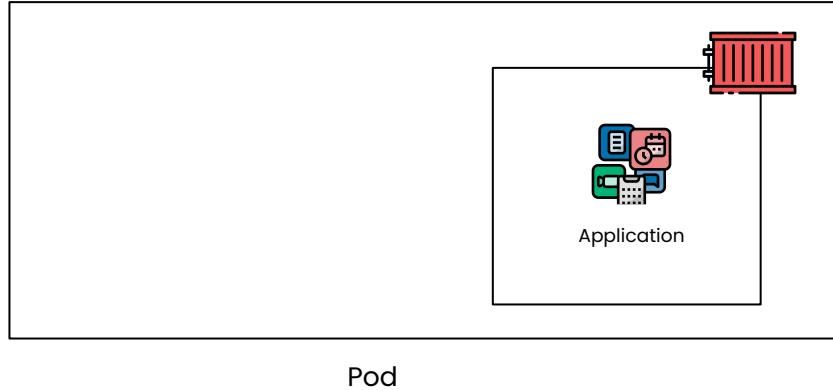
- [Configure a Pod to Use a Secret](#) -

## Configure a Pod to Use a Secret – All key-value as file(s) (Cont.)

```
apiVersion: v1
kind: Pod
metadata:
  name: secret-test-pod
spec:
  containers:
    - name: test-container
      image: nginx
      volumeMounts:
        - name: secret-volume
          mountPath: /etc/secret-volume
          readOnly: true
  volumes:
    - name: secret-volume
      secret:
        secretName: test-secret
```

## Configure a Pod to Use a Secret – All key-value as file(s) (Cont.)

```
apiVersion: v1
kind: Pod
metadata:
  name: secret-test-pod
spec:
  containers:
    - name: test-container
      image: nginx
      volumeMounts:
        - name: secret-volume
          mountPath: /etc/secret-volume
          readOnly: true
      volumes:
        - name: secret-volume
          secret:
            secretName: test-secret
```

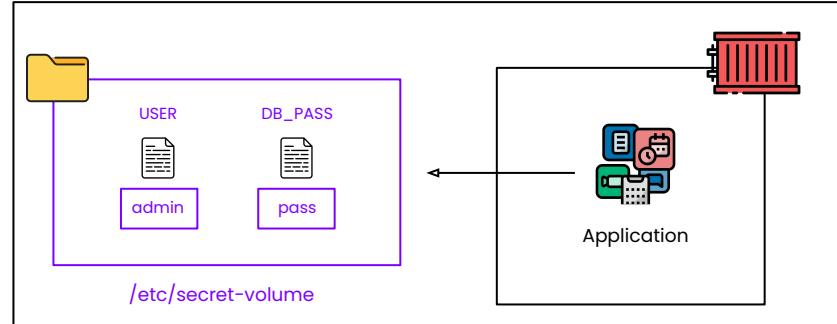
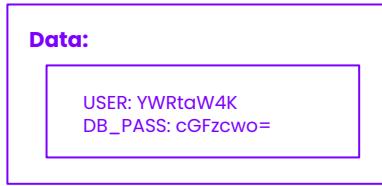


Pod

# Configure a Pod to Use a Secret – All key-value as file(s) (Cont.)

```
apiVersion: v1
kind: Pod
metadata:
  name: secret-test-pod
spec:
  containers:
    - name: test-container
      image: nginx
      volumeMounts:
        - name: secret-volume
          mountPath: /etc/secret-volume
          readOnly: true
    volumes:
      - name: secret-volume
        secret:
          secretName: test-secret
```

Secret: test-secret

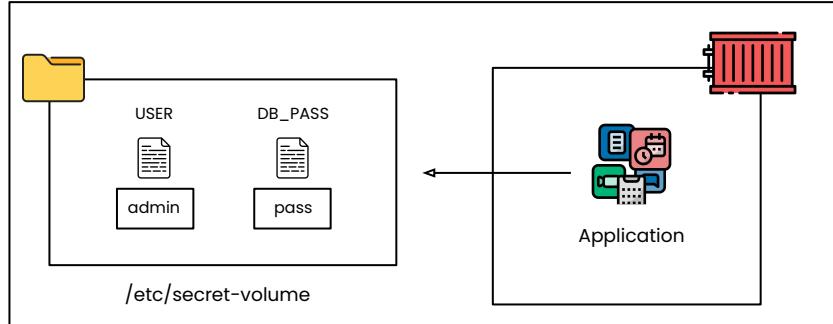
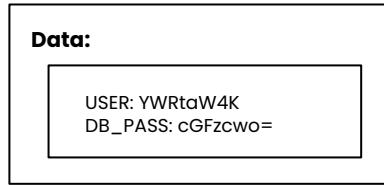


Pod

# Configure a Pod to Use a Secret – All key-value as file(s) (Cont.)

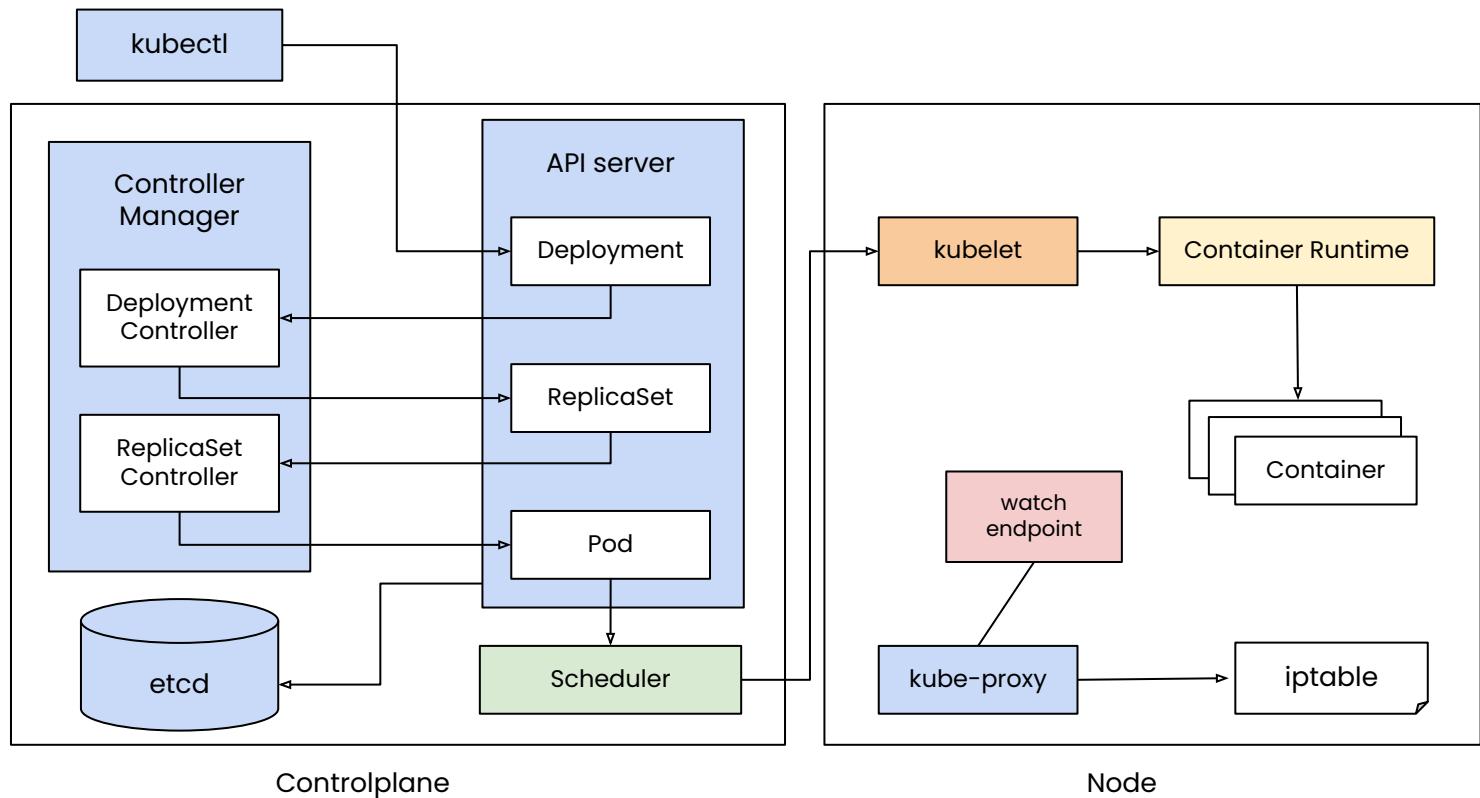
```
apiVersion: v1
kind: Pod
metadata:
  name: secret-test-pod
spec:
  containers:
    - name: test-container
      image: nginx
      volumeMounts:
        - name: secret-volume
          mountPath: /etc/secret-volume
          readOnly: true
      volumes:
        - name: secret-volume
          secret:
            secretName: test-secret
```

Secret: test-secret

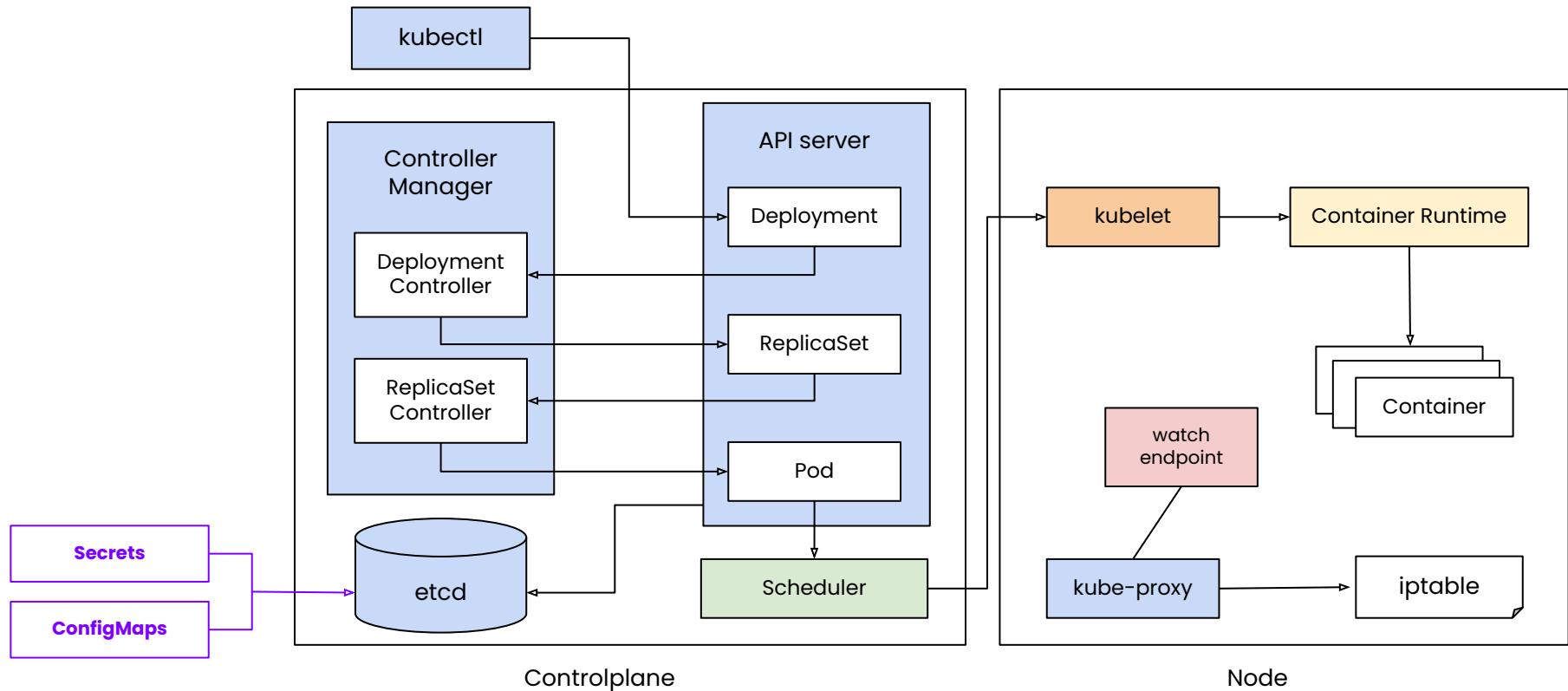


Pod

# Where the ConfigMaps and Secret stored in?



# Where the ConfigMaps and Secret stored in?



# Secret

With Real Action Simulator

[Click Here](#)



เจ้าของลิขสิทธิ์ อนุญาติให้ใช้ส่วนหนึ่งของการเรียนรู้ส่วนบุคคลเท่านั้น และขอสงวนลิขสิทธิ์ ห้ามมิให้เผยแพร่ในที่สาธารณะ ผู้อื่นเดิน จดถูกต้องเป็นคัดลอกกฎหมาย

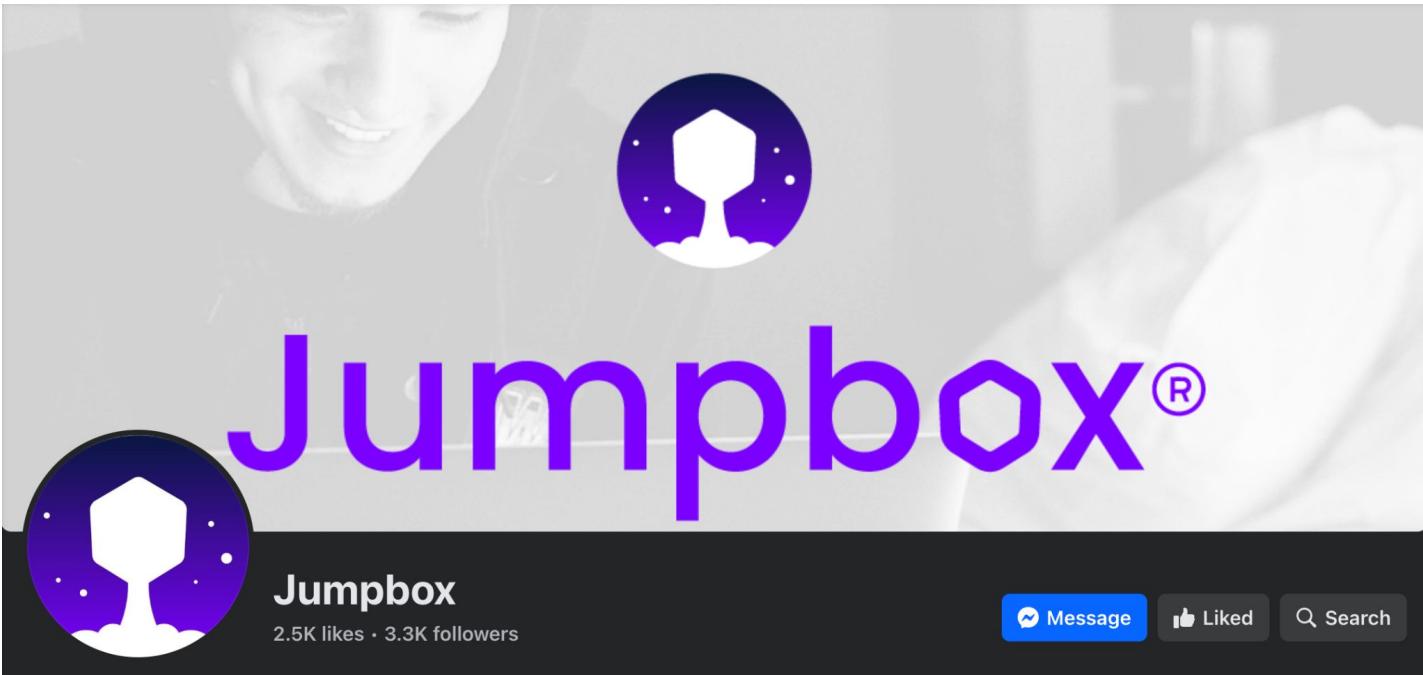
Jumpbox®



# Jumpbox®

Tech Passion | Sharing | Society

# facebook



<https://www.facebook.com/jumpbox.academy>

เจ้าของลิขสิทธิ์ อนุญาติให้ใช้ส่วนหนึ่งของการเรียนรู้ส่วนบุคคลเท่านั้น และขอสงวนลิขสิทธิ์ ห้ามมิให้เผยแพร่ในที่สาธารณะ ผู้อื่นเดินทาง เดินทางโดยไม่ได้รับอนุญาต

Jumpbox®



## Jumpbox

@jumpbox.academy • 1.92K subscribers • 30 videos

More about this channel ...[more](#)

<https://www.youtube.com/@jumpbox.academy>

# Contact Us



**Jumpbox**



**@jumpbox**



**admin@jumpbox.co**



**063-245-2168 (JoJo)**

**062-796-1559 (Beau)**



"เราเชื่อว่า การเรียนรู้ทำให้ชีวิตคุณดีขึ้น"

-Jumpbox Team-



# Jumpbox®

เจ้าของลิขสิทธิ์ อนุญาติให้ใช้สิ่งที่มีการเรียบเรียงร่วมกับบุคคลท่านนั้น และขอสงวนลิขสิทธิ์ ห้ามมิให้เผยแพร่ในที่สาธารณะ ผู้อื่นเดินทาง จะถูกดำเนินคดีตามกฎหมาย