# KUBERNETES(K8S)

## Pilot or Helmsman

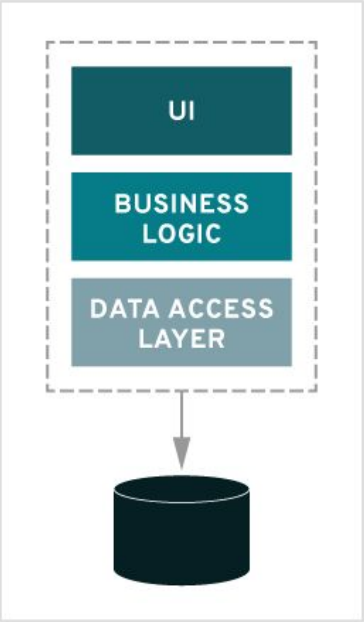
The person holding the software's steering wheel

2nd WEEK

30th August, 2019

# Monolith Architecture to MicroServices Architecture

## MONOLITHIC

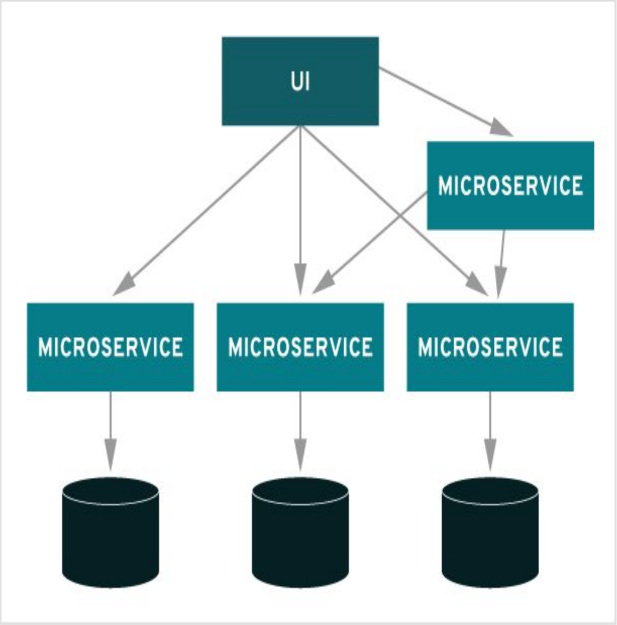


Microservices are about breaking up a complex system into more manageable, **smaller units** of work that can be **developed and deployed independently**.

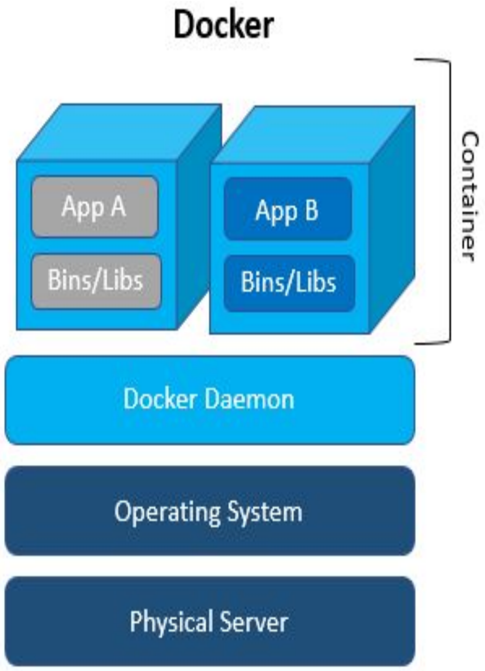
The microservices architecture allows engineers to develop and deploy software apps as discrete, loosely connected modules. A separation of services from servers and the ability to run them independently.

To achieve this isolation, we use **containerisation**

## MICROSERVICES



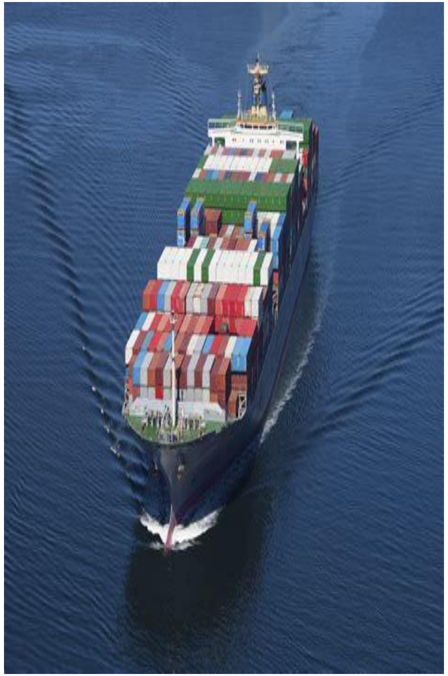
# Containerisation: What, Why and How



Encapsulating an application in a container with its own **operating environment**

Containerisation keeps the separation between the two by **isolating their processes** and hence helping them to achieve their respective goals in tandem.

Containers also provide **modularity** in development.



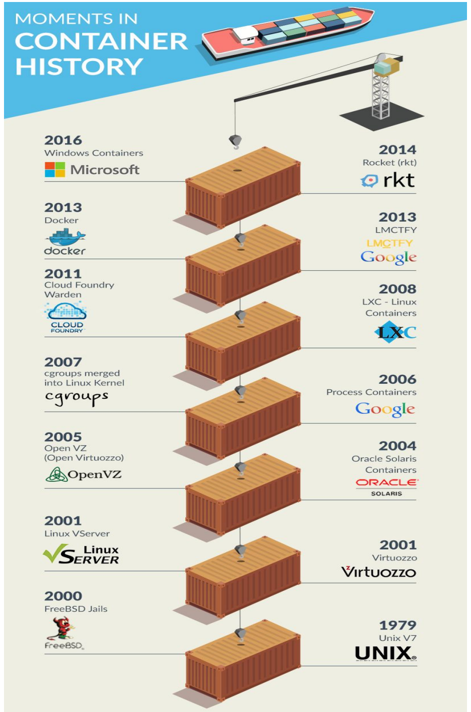
# History of Containers

## 2008 – LXC

LXC was the first and the most complete Linux container. It was developed by using cgroups and Linux namespaces

## 2013 – Docker

It is still rated as the market leader. Though initially it made use of LXC. It differed in the way that it was a complete ecosystem for handling containers.



## 2016 – Windows Con.

After watching the popularity of containers in the Linux operating system, Microsoft released Microsoft Windows Server 2016 with Windows container support

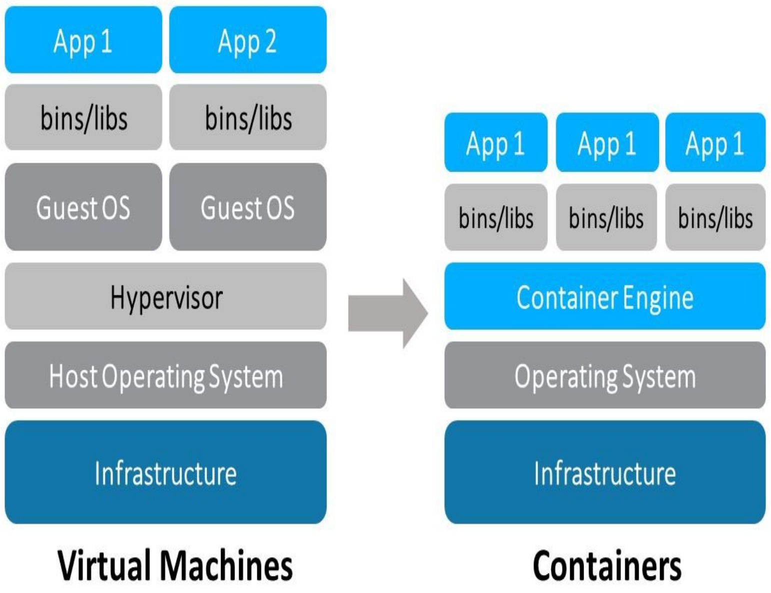
## Other Containers

CoreOS Rkt

Kata Container

Kontena

# Virtual Machine vs Containers



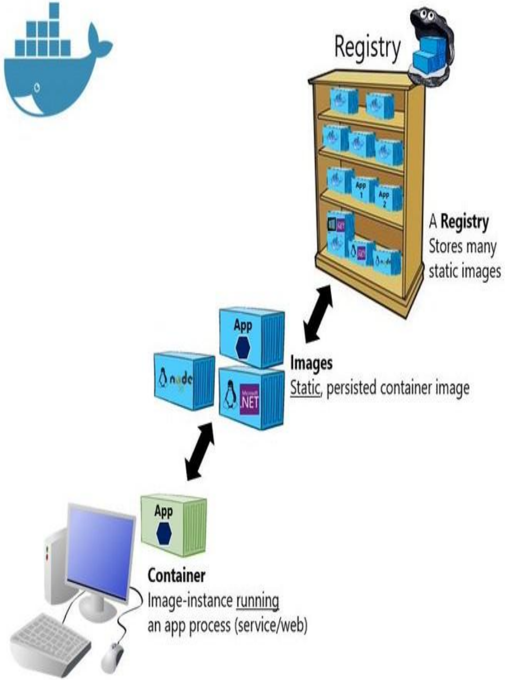
## Hardware Virtualisation

one processor acts like a different virtual processor and more users can access the same physical processor at the same time.

## Operating System Virtualisation

Container instances that are created on the container layer use the host OS only – this layer doesn’t have its own virtual OS and applications.

# Docker - The world’s leading containerisation Platform

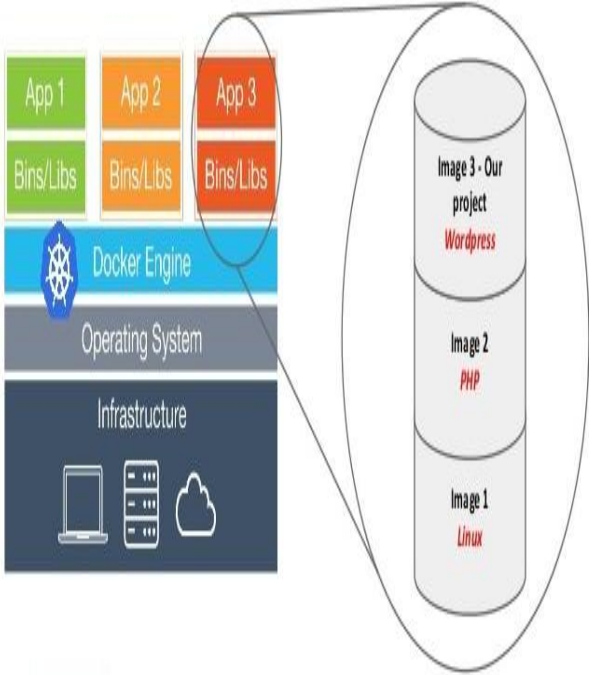


## Docker

Container technology that allows you to **containerize** applications and It helps run containerised application

## Docker Compose

Docker compose is used for configuring and starting multiple containers on the **same** host



# What is Kubernetes?

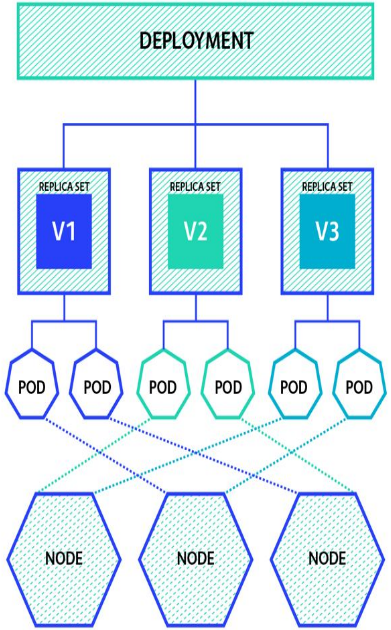
## K8S Inception

Kubernetes was launched in 2014, after more than a decade of experience of running production workloads at Google with Google’s internal container cluster managers, **Borg** and **Omega**.

## Purpose of K8S

Kubernetes, at a basic level, is a system for **running and coordinating containerised applications** across a cluster of machines. It is a platform designed to completely **manage the life cycle of containerised applications and services** using methods that provide predictability, scalability and high availability.

# Kubernetes’ Prominent Resources

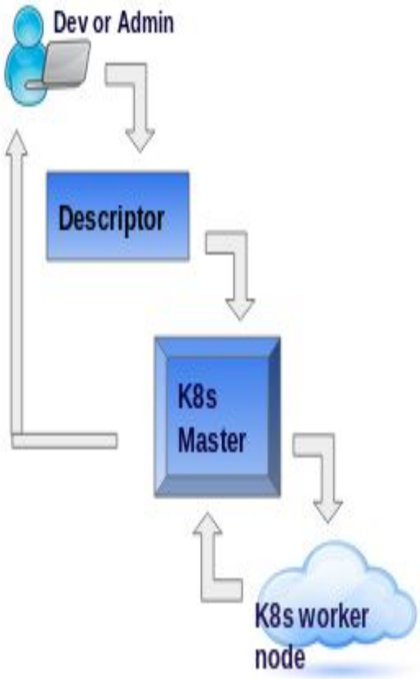


**Container:** This is the smallest unit in the Kubernetes world.

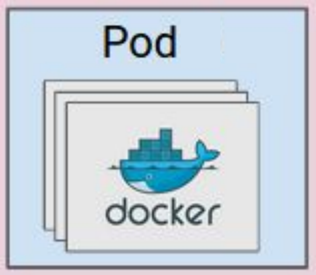
**Pod:** This is a management unit in Kubernetes comprising one or more containers.

**Node:** A node is the host on which the containers run.

**Deployment:** A pod by itself is mortal but with a deployment, Kubernetes can make sure that the number of pods that a user specifies is always up and running in the system.

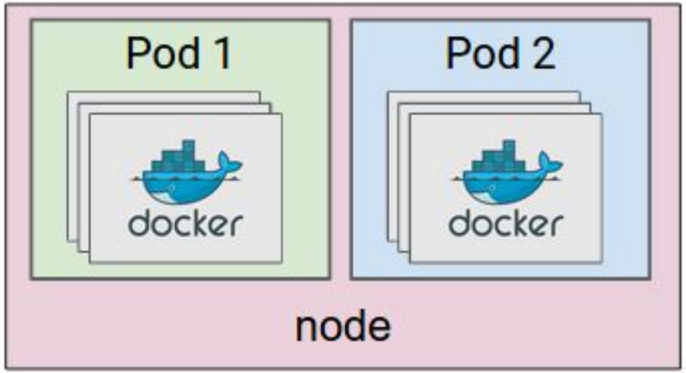


# Kubernetes Pod Game



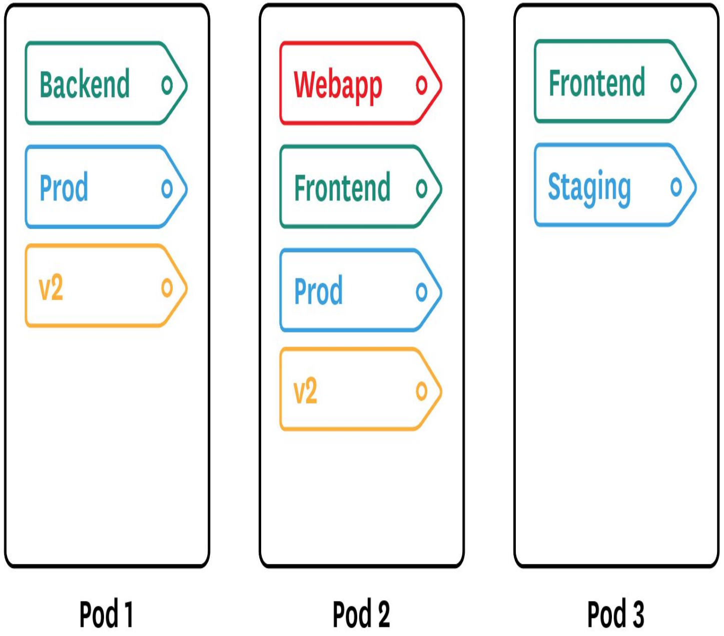
## Pod (po)

Pod is a co-located **group of containers** and represent the basic building block in kubernetes. A pod of containers allows you to **run closely related processes together** and provide them with the **same environment**.



Kubernetes achieves this by configuring Docker to have all **containers of a pod share the same set of Linux namespaces instead of each container having its own set**. Because all containers of a pod run under the **same Network and UTS namespaces**. they all share the same hostname and network interfaces. But by default, the **filesystem of each container is fully isolated from other containers.** A pod never span two nodes

# Label It



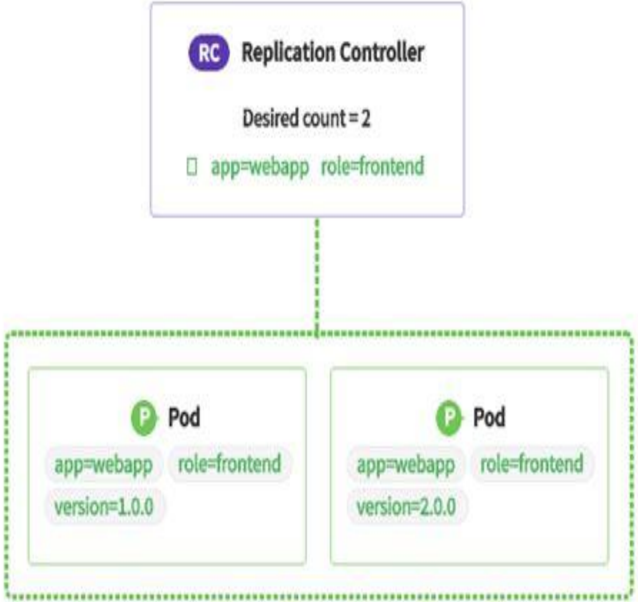
## Label

A label is an arbitrary **key-value pair** attach to a resource. A resource can have more than one label, as long as the keys of those labels are unique within that resource.

## Purpose

Labelling leads to much-better-organized system. Using Labels, Kubernetes resources can be selected, deleted as group

# Replication Controller and Replication Sets



## Replication Controller (rc)

A Replication Controller is a Kubernetes resource that ensures its pods are always kept running. It notices the missing pod and creates a replacement pod

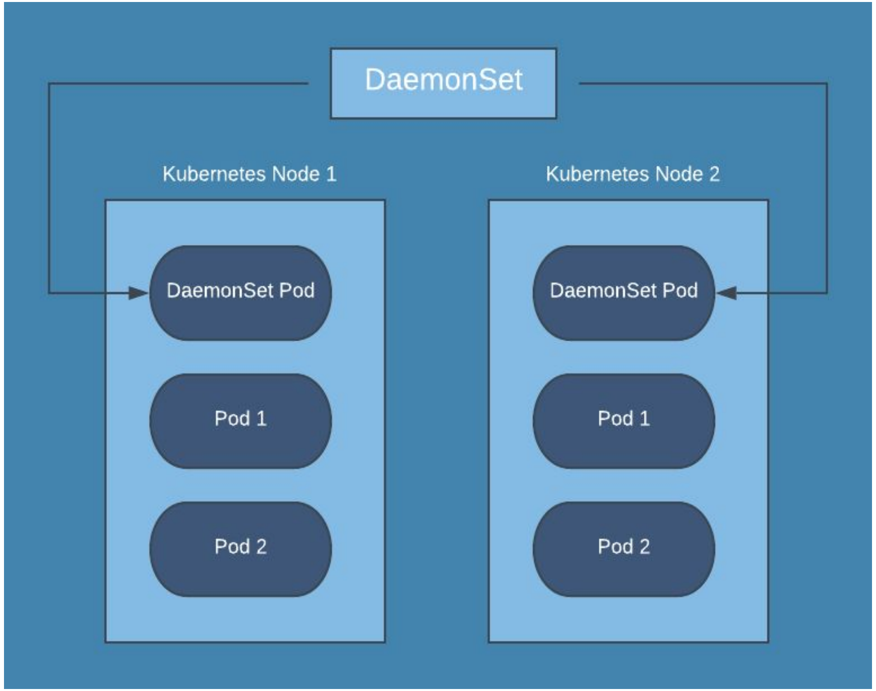
## 3 Key Parts

1.**Pod Selector** 2. **Replica** 3.**Pod Template**

## Replica Sets (rs)

Replica Set has more **expressive pod selectors**

# DaemonSets



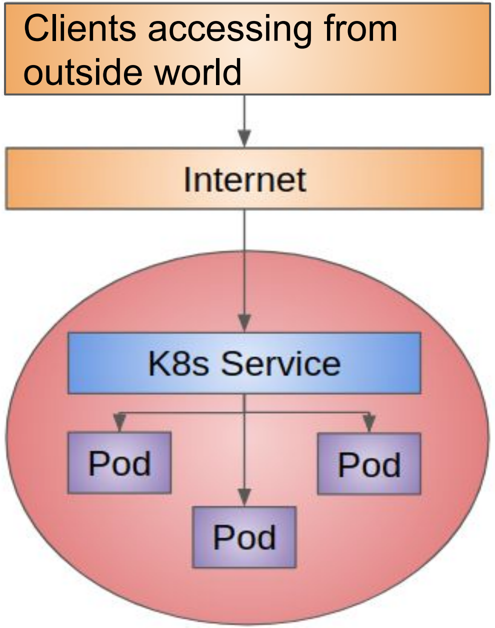
## DaemonSet (ds)

DaemonSets run only a **single pod replica on each node**, whereas Replica-Sets scatter them around the whole cluster randomly.

## Purpose

log collector and resource monitor

# Kubernetes Services



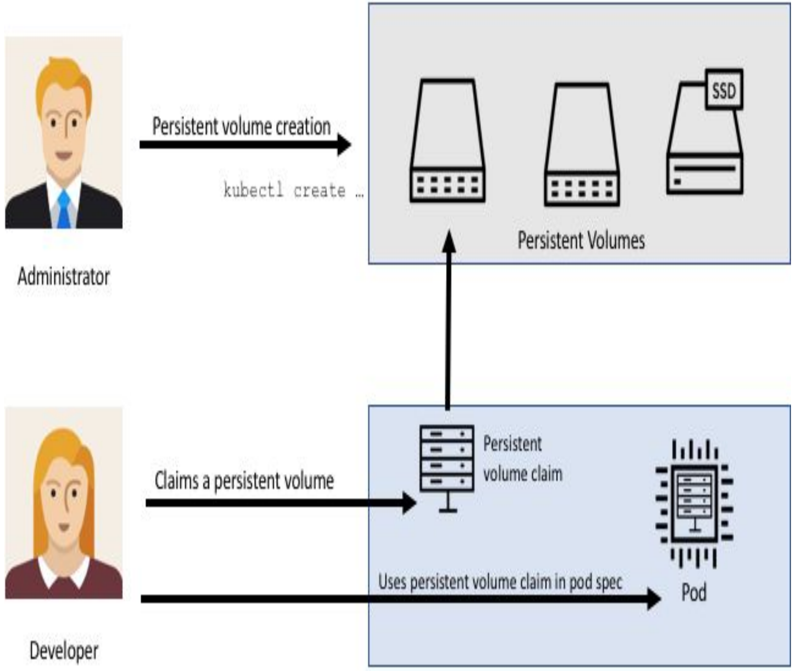
## K8S Services (svc)

A Kubernetes Service is a resource you create to make a single, **constant point of entry to a group of pods** providing the same service. Each service has an IP address and port that never change while the service exists

## With-in and From Outside

1. Cluster IP (with-in)
2. Node Port Service
3. Load balancing Service
4. Ingress

# K8s Volume


Persistent volume creation
kubectl create ...
Administrator
SSD
Persistent Volumes
Claims a persistent volume
Persistent volume claim
Uses persistent volume claim in pod spec
Pod
Developer

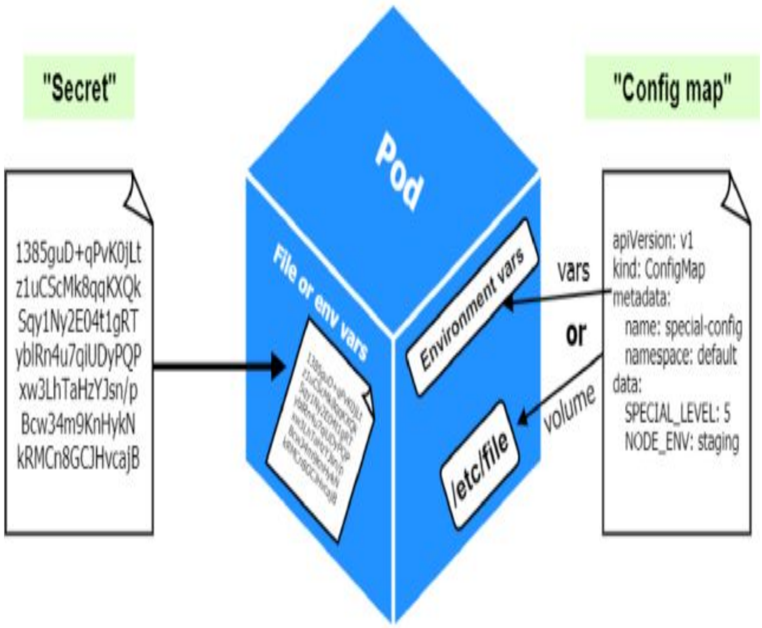
## Volume

A volume's contents will persist across container restarts. After a container is restarted, the new container can see all the files that were written to the volume by the previous container. Also, if a pod contains multiple containers, the volume can be used by all of them at once.

## Volume Types

emptyDir, hostPath , gitRepo, nfs, cloud provider-specific storage (gcePersistentDisk, awsElastic-BlockStore, azureDisk) ..

# ConfigMaps and Secrets



```
1385guD+qPvK0jLt
z1uCScMk8qqKXQk
Sqy1Ny2E04t1gRT
ybIRn4u7qlUDyPQP
xw3LhTaHzYJsn/p
Bcw34m9KnHykN
kRMCn8GCJHvcajB
```

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: special-config
  namespace: default
data:
  SPECIAL_LEVEL: 5
  NODE_ENV: staging
```

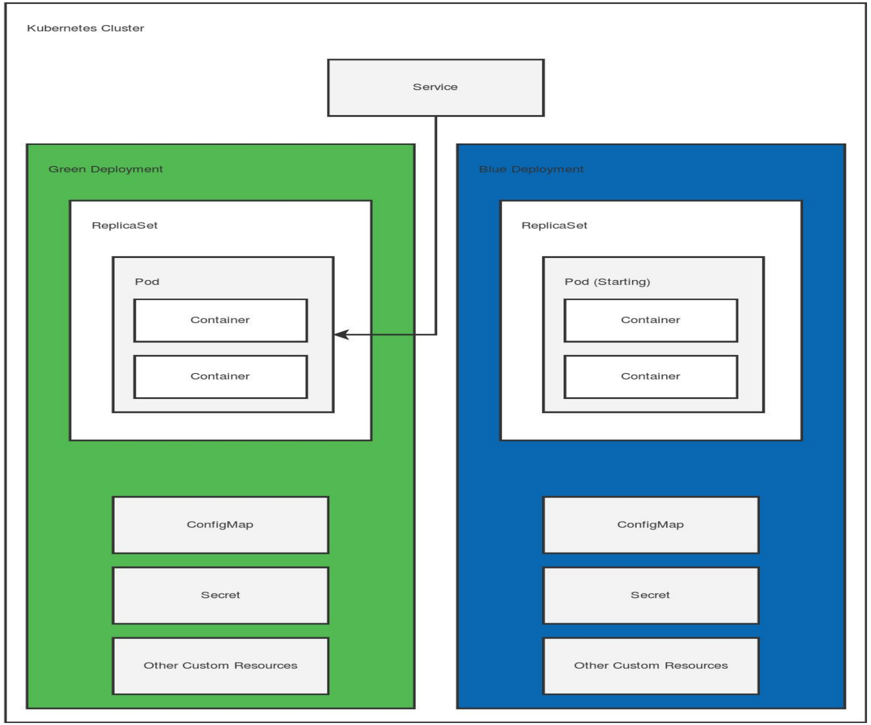
## ConfigMaps

Kubernetes allows separating configuration options into a separate object called a ConfigMap, which is a map containing **key/value** pairs with the values ranging from **short literals to full config files.**

## Secrets

Secrets are also map containing **key/value** and can be **files**. Secret to store any data that is sensitive in nature and needs to be kept under key.

# Deployments



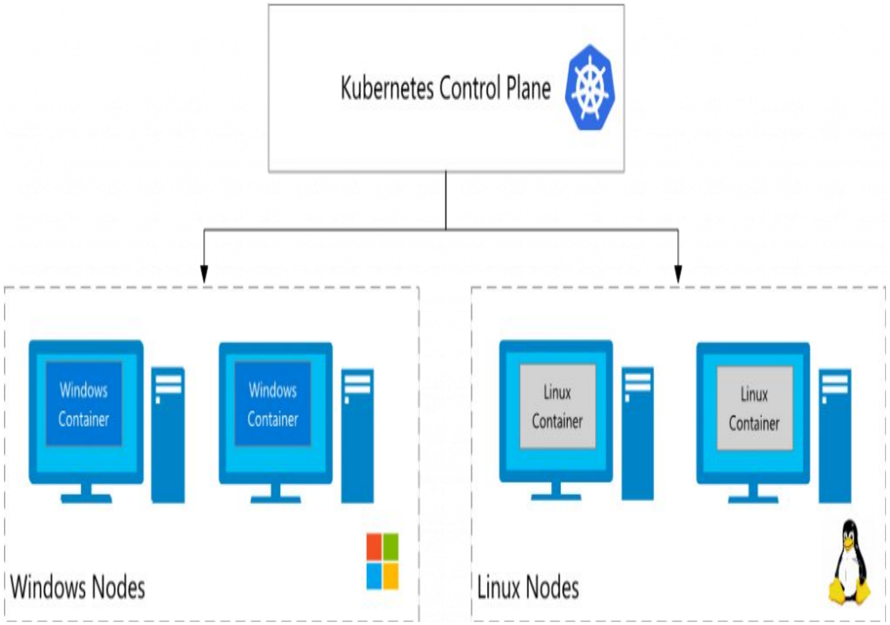
## Deployment

Deployment creates a **ReplicaSet** which in turn make sure desired pod is running in the cluster.

Recreate or Rolling update

Rolling back a rollout is possible because Deployments keep a revision history.

# Windows containers in Kubernetes



Windows applications constitute a large portion of the services and applications that run in many organisations.

Adding Windows nodes in your existing Linux cluster

**Windows Server 2019** is the only Windows operating system supported, enabling Kubernetes Node on Windows

# It is the World of Containers

Thank You