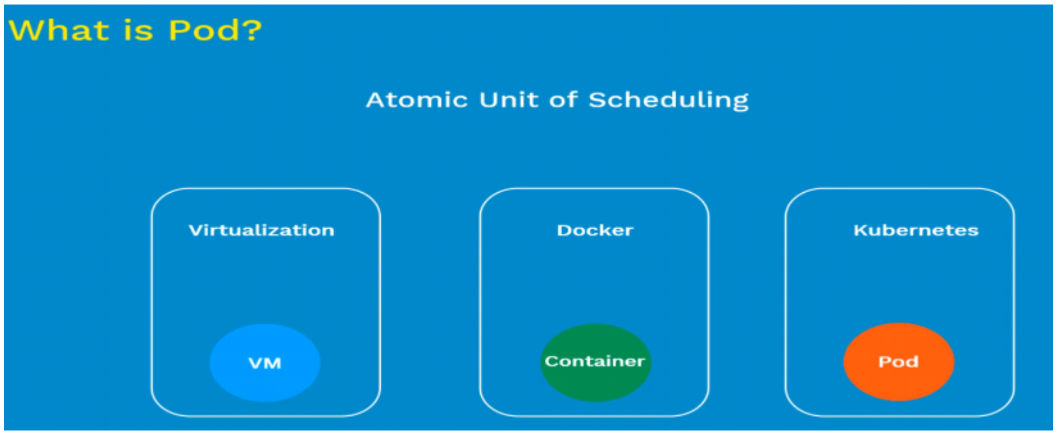
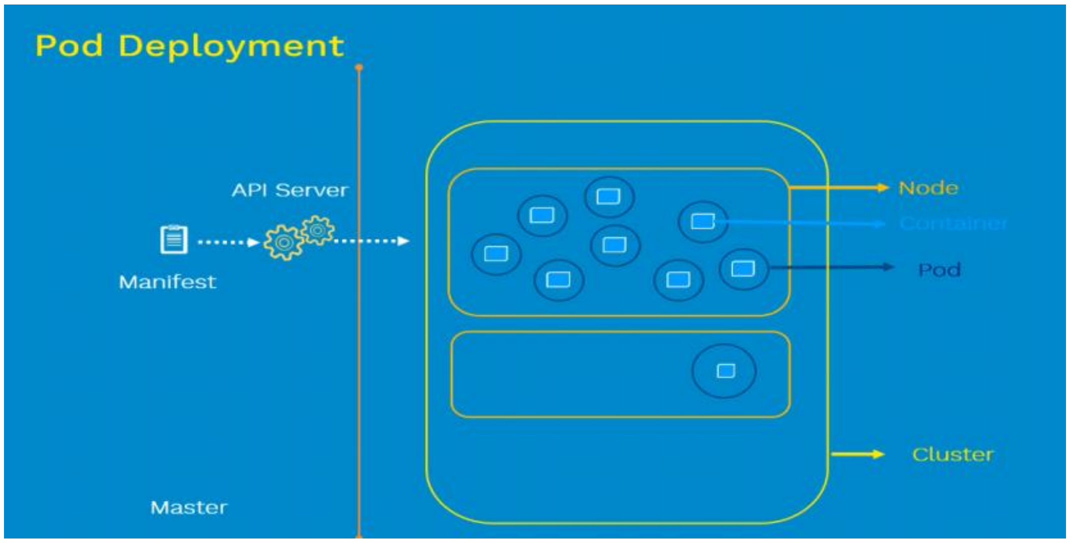
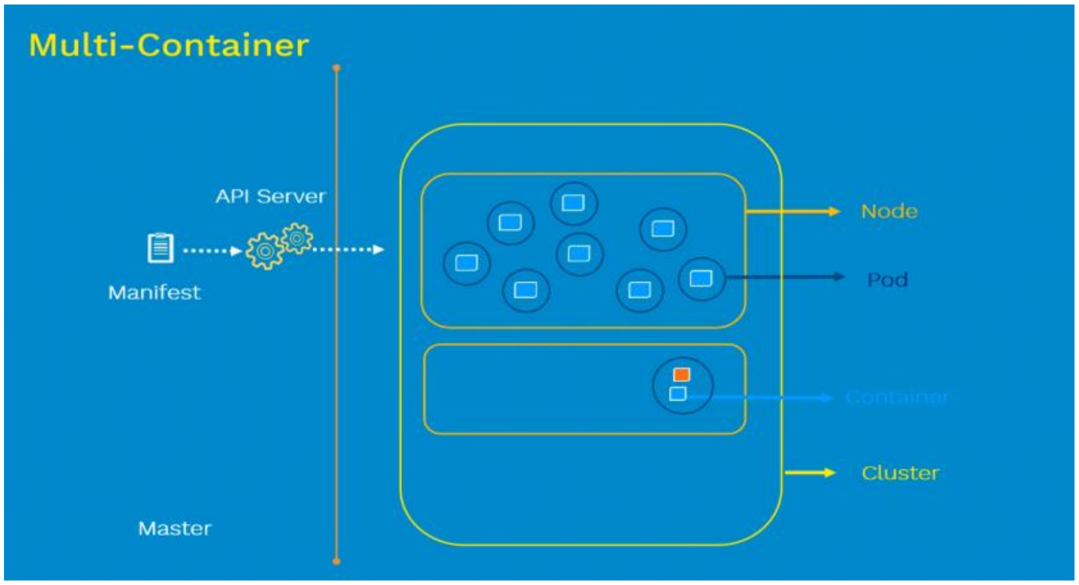
Kubernetes

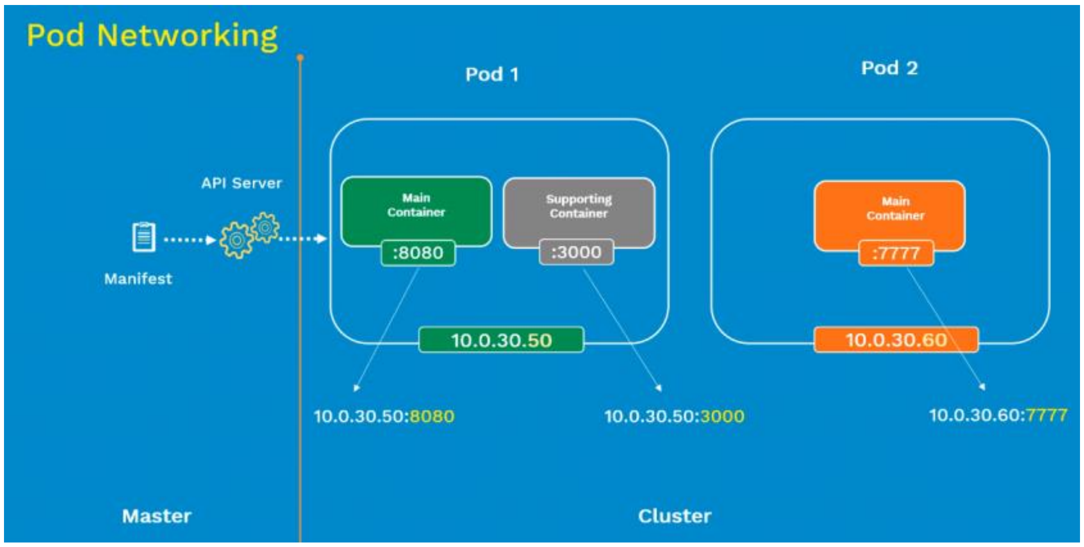
Pods:-It is the Atomic unit of Scheduling.

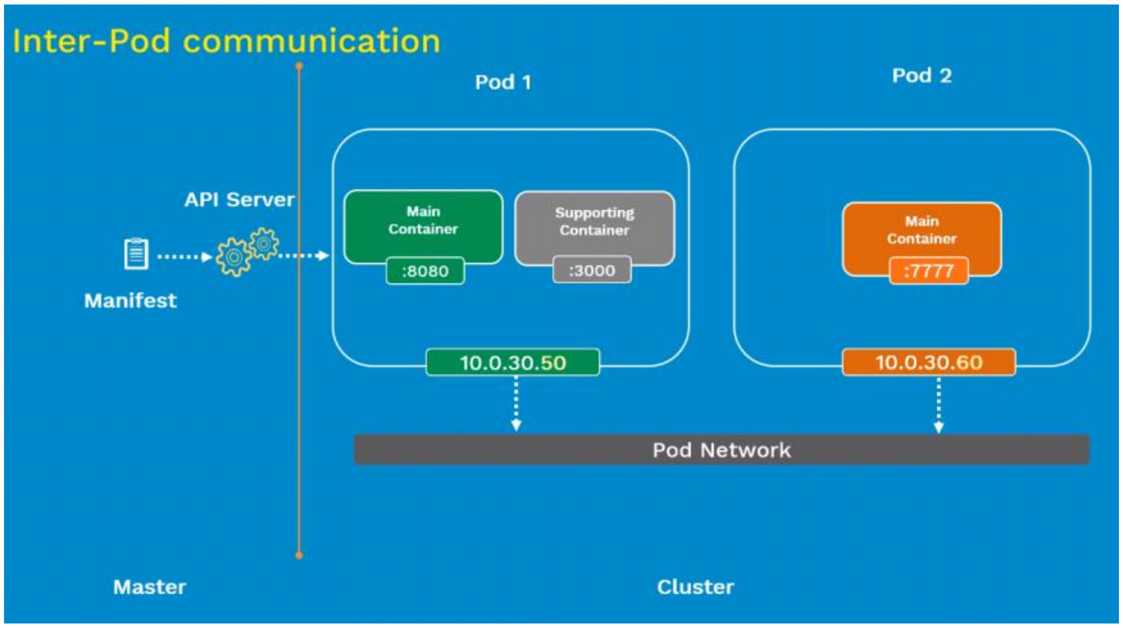


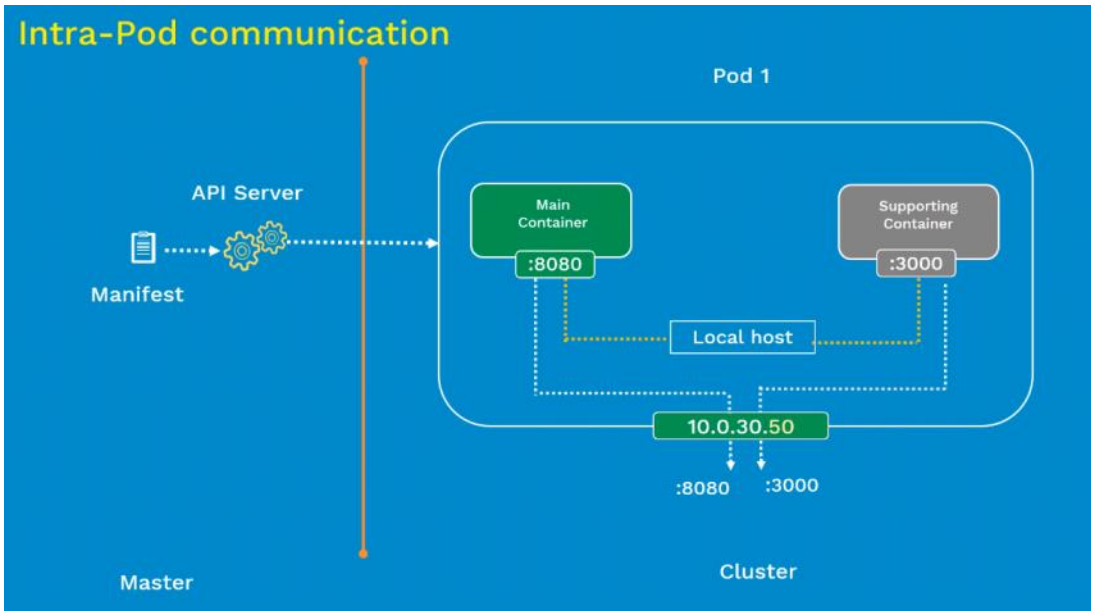
A **Kubernetes pod** is a group of containers that are deployed together on the same host. If you frequently deploy single containers, you can generally replace the word "**pod**" with "container" and accurately understand the concept

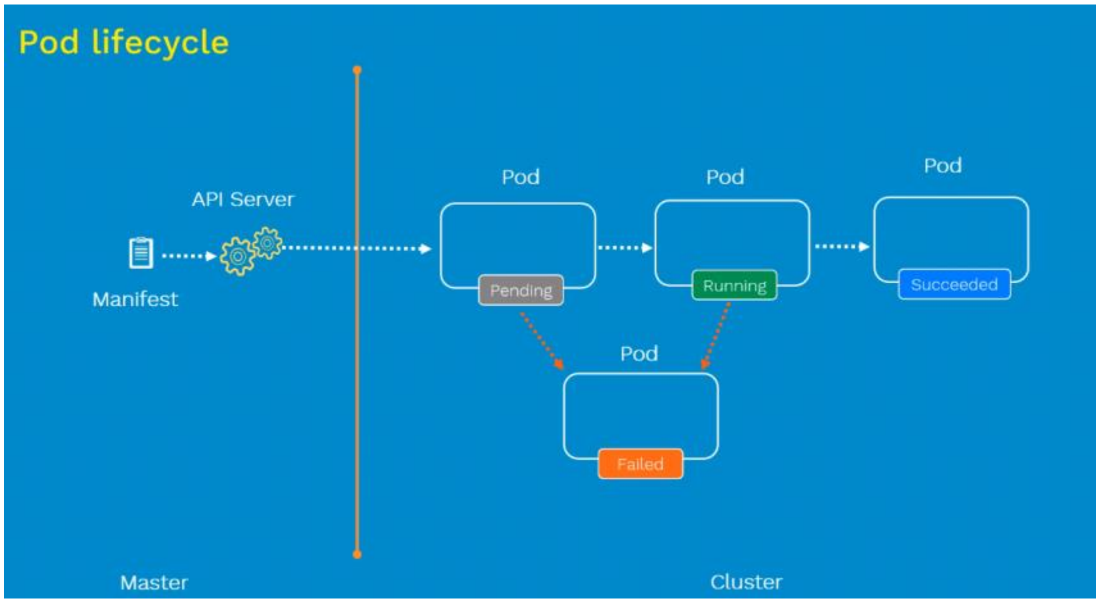


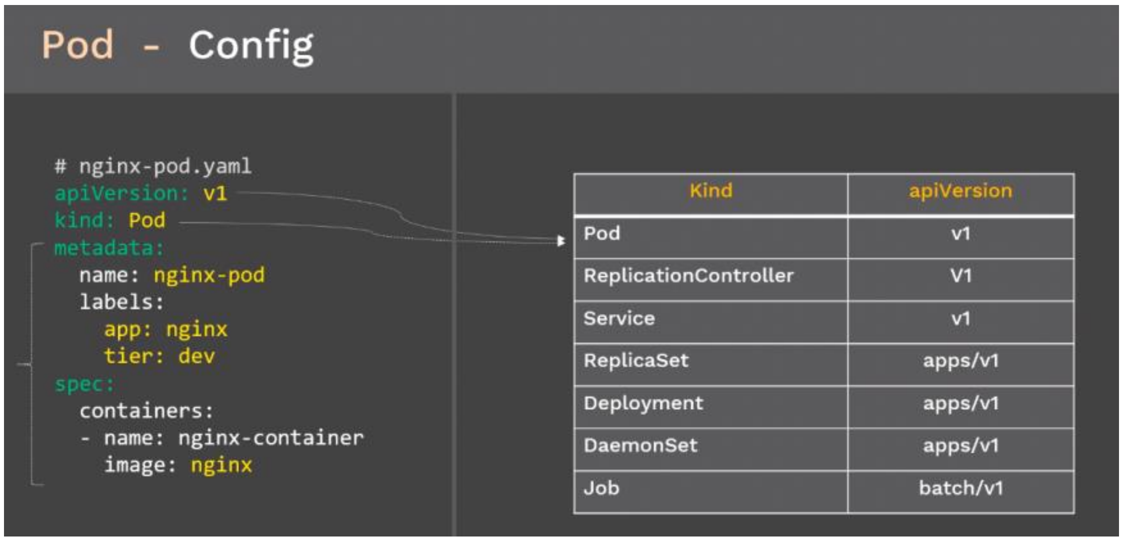



Pod Networking
Pod 1
Pod 2
API Server
Manifest
Main Container
:8080
Supporting Container
:3000
Main Container
:7777
10.0.30.50
10.0.30.60
10.0.30.50:8080
10.0.30.50:3000
10.0.30.60:7777
Master
Cluster


Inter-Pod communication
Pod 1
Pod 2
API Server
Manifest
Main Container
:8080
Supporting Container
:3000
Main Container
:7777
10.0.30.50
10.0.30.60
Pod Network
Master
Cluster


Intra-Pod communication
Pod 1
API Server
Manifest
Main Container
:8080
Supporting Container
:3000
Local host
10.0.30.50
:8080
:3000
Master
Cluster



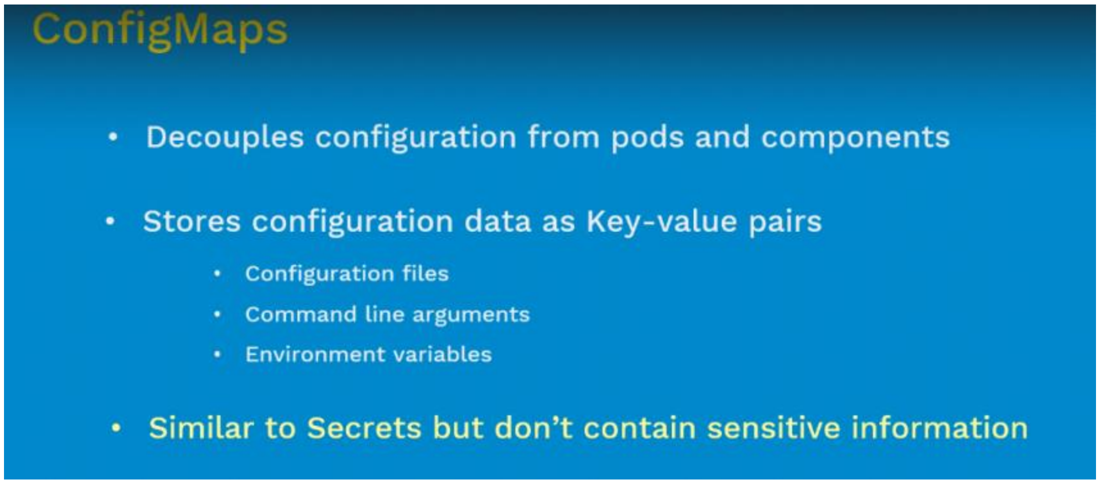


Pod - Config

```
# nginx-pod.yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx-pod
  labels:
    app: nginx
    tier: dev
spec:
  containers:
  - name: nginx-container
    image: nginx
```

Kind	apiVersion
Pod	v1
ReplicationController	V1
Service	v1
ReplicaSet	apps/v1
Deployment	apps/v1
DaemonSet	apps/v1
Job	batch/v1

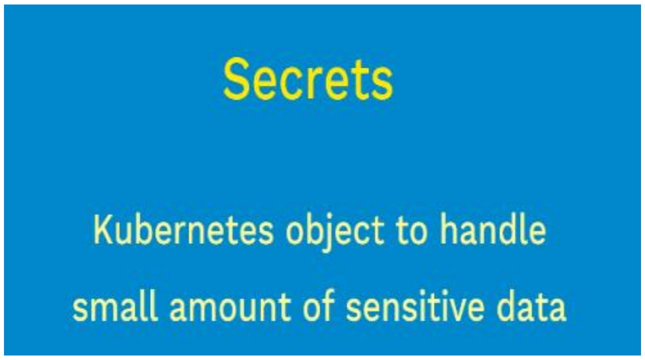
DEMO:-Pod Creation



ConfigMaps

- Decouples configuration from pods and components
- Stores configuration data as Key-value pairs
  - Configuration files
  - Command line arguments
  - Environment variables
- Similar to Secrets but don't contain sensitive information

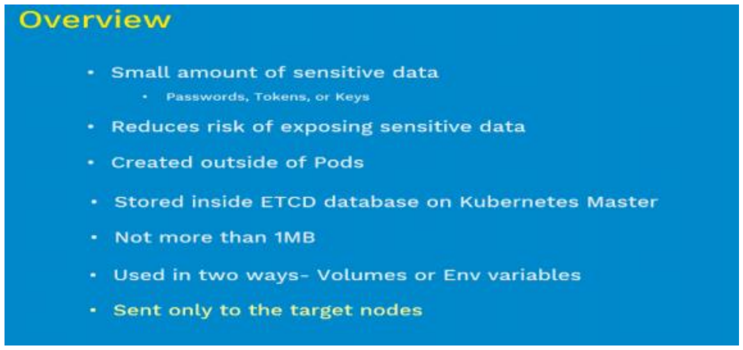
DEMO-Config Maps



Secrets

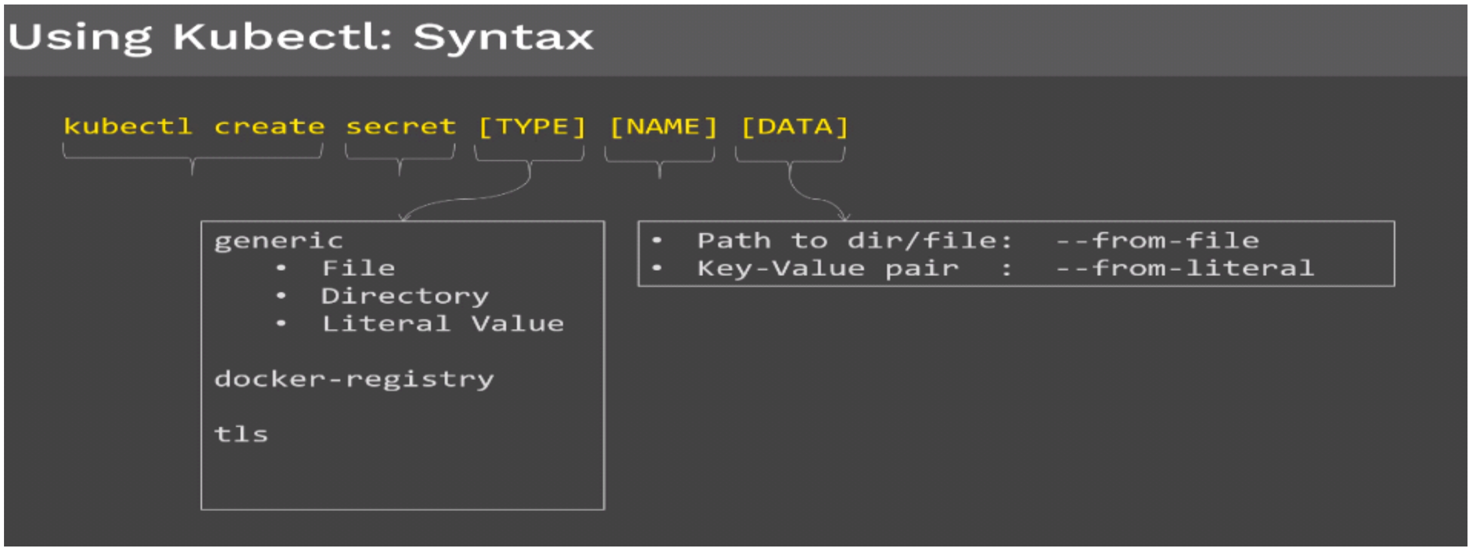
Kubernetes object to handle

small amount of sensitive data



Overview

- Small amount of sensitive data
  - Passwords, Tokens, or Keys
- Reduces risk of exposing sensitive data
- Created outside of Pods
- Stored inside ETCD database on Kubernetes Master
- Not more than 1MB
- Used in two ways- Volumes or Env variables
- Sent only to the target nodes



Using Kubectl: Syntax

```
kubectl create secret [TYPE] [NAME] [DATA]
```

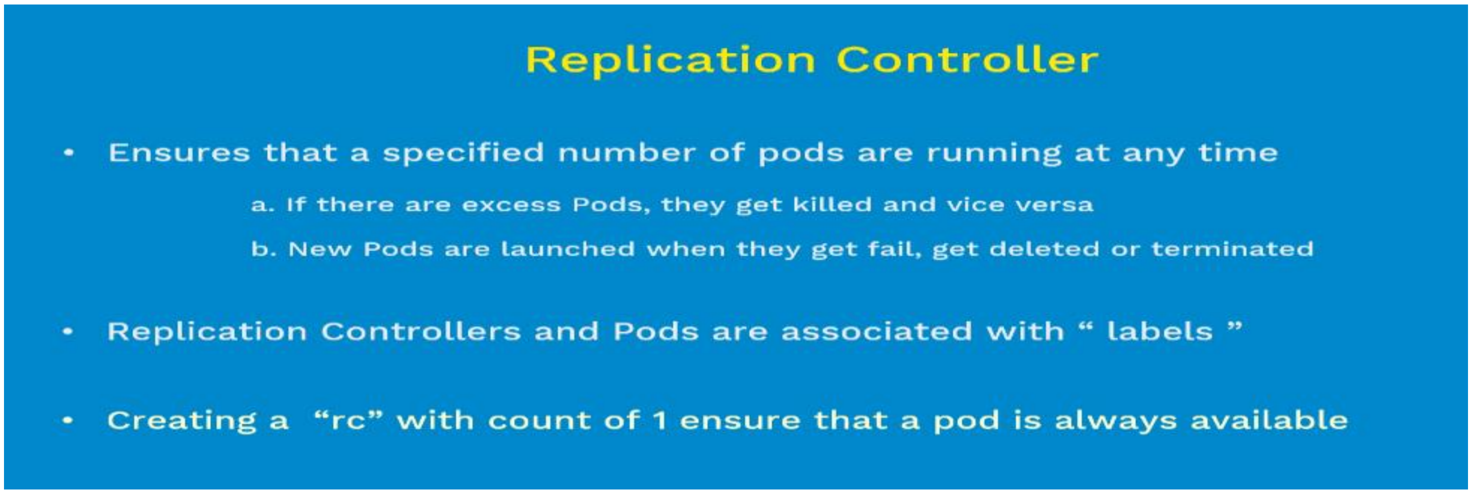
```
generic
    •  File
    •  Directory
    •  Literal Value

docker-registry

tls
```

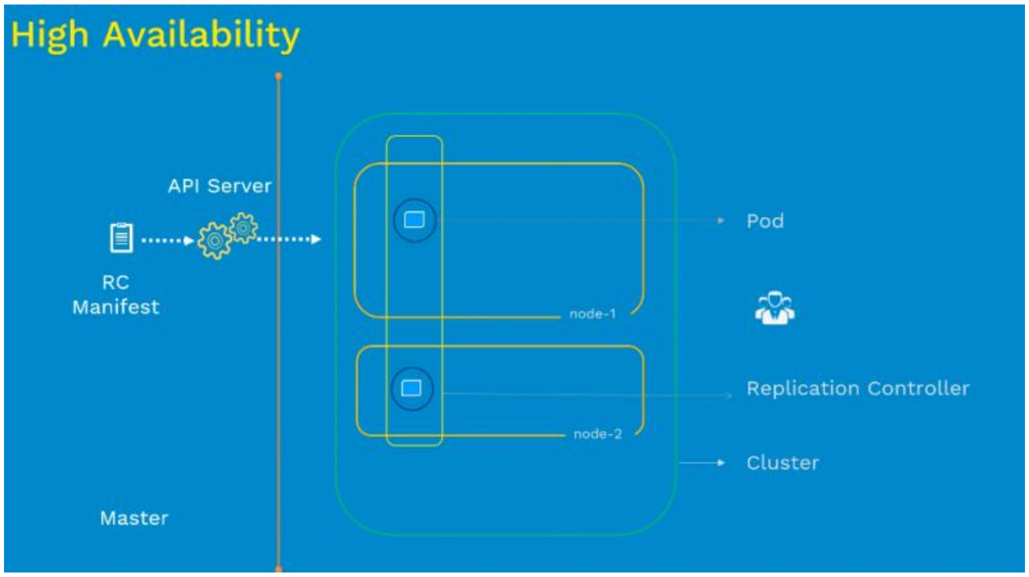
```
•  Path to dir/file:   --from-file
•  Key-Value pair  :   --from-literal
```

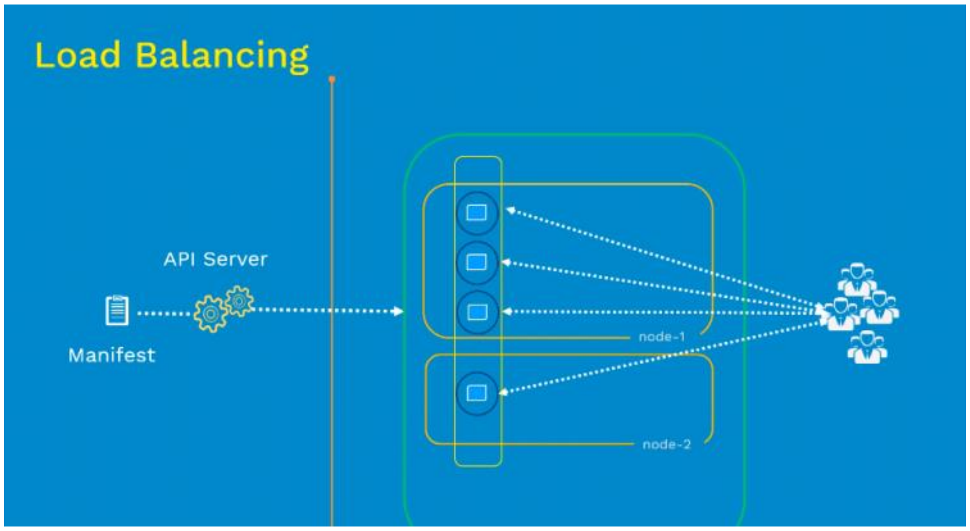
Secert Demo



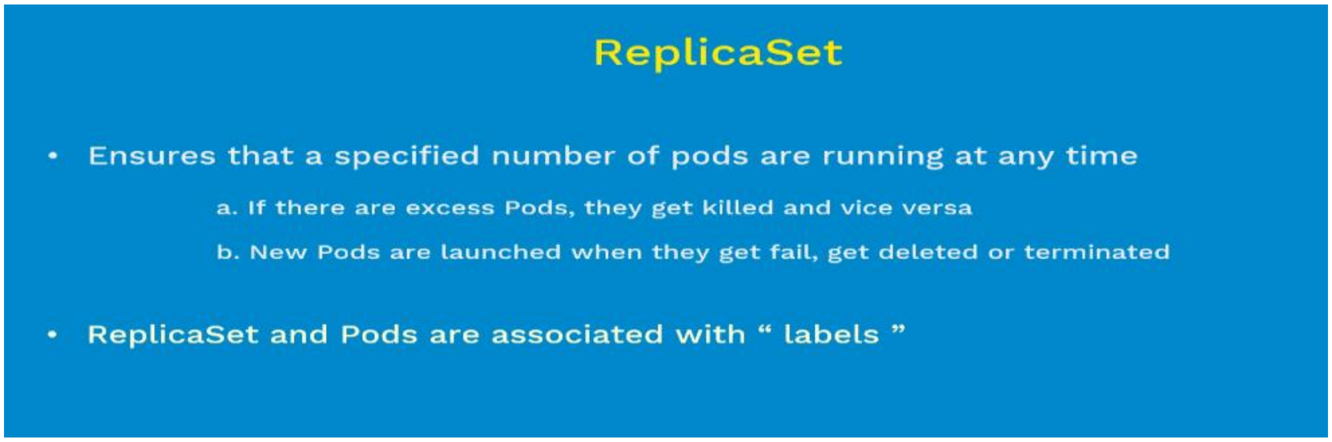
Replication Controller

- Ensures that a specified number of pods are running at any time
  - a. If there are excess Pods, they get killed and vice versa
  - b. New Pods are launched when they get fail, get deleted or terminated
- Replication Controllers and Pods are associated with " labels "
- Creating a "rc" with count of 1 ensure that a pod is always available



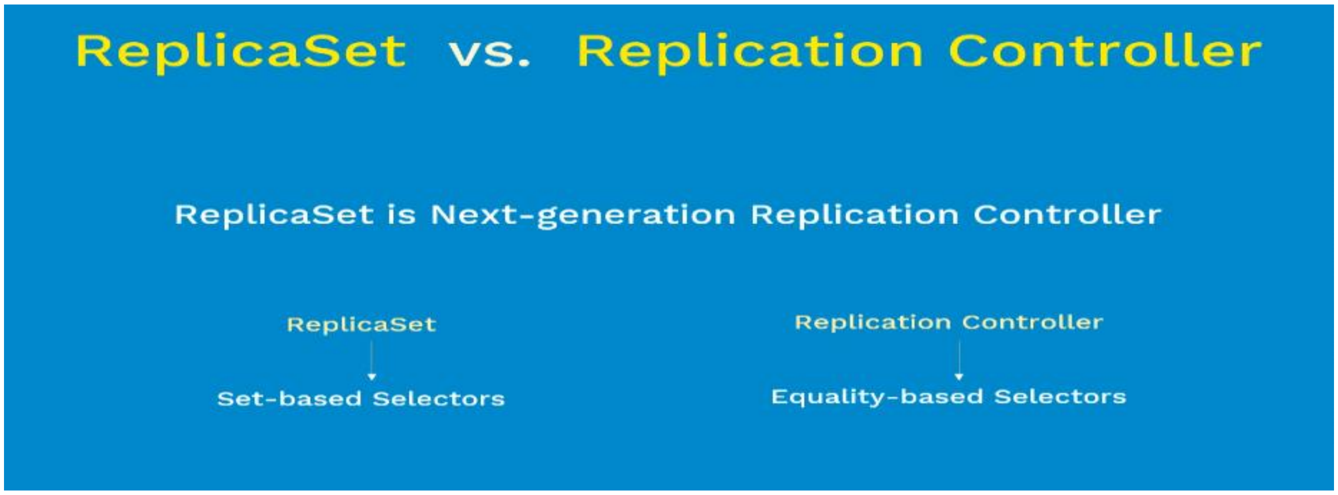


Replication Controller Demo



ReplicaSet

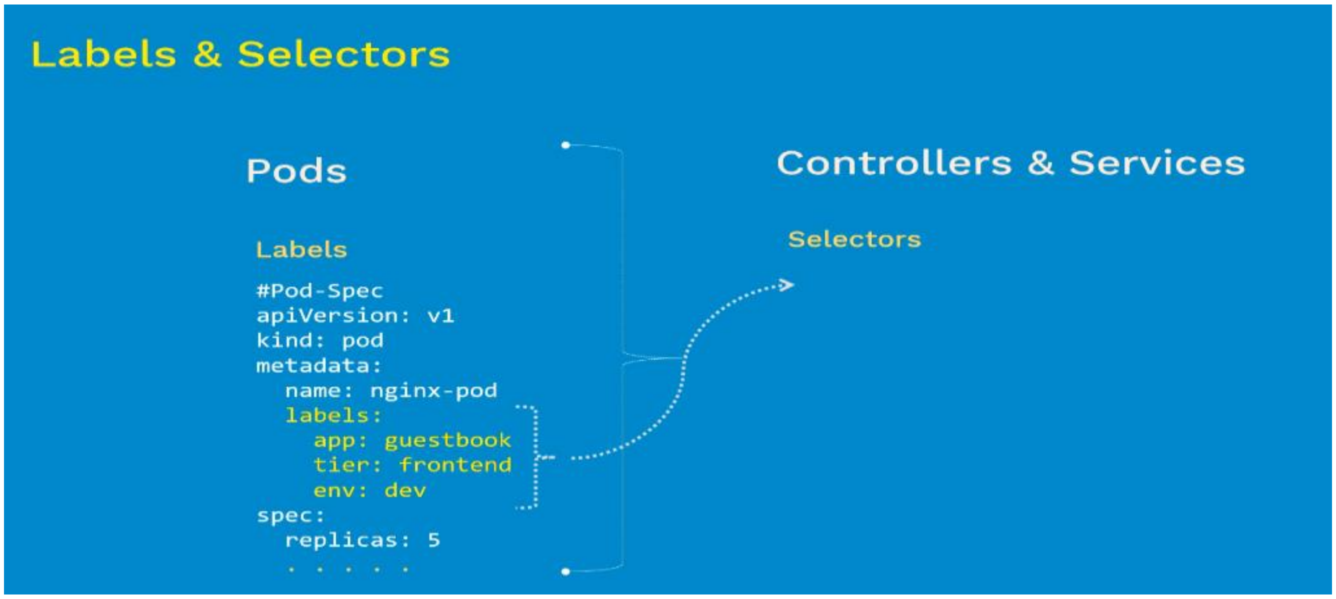
- Ensures that a specified number of pods are running at any time
  - a. If there are excess Pods, they get killed and vice versa
  - b. New Pods are launched when they get fail, get deleted or terminated
- ReplicaSet and Pods are associated with " labels "



ReplicaSet vs. Replication Controller

ReplicaSet is Next-generation Replication Controller

ReplicaSet	Replication Controller
↓	↓
Set-based Selectors	Equality-based Selectors



Labels & Selectors

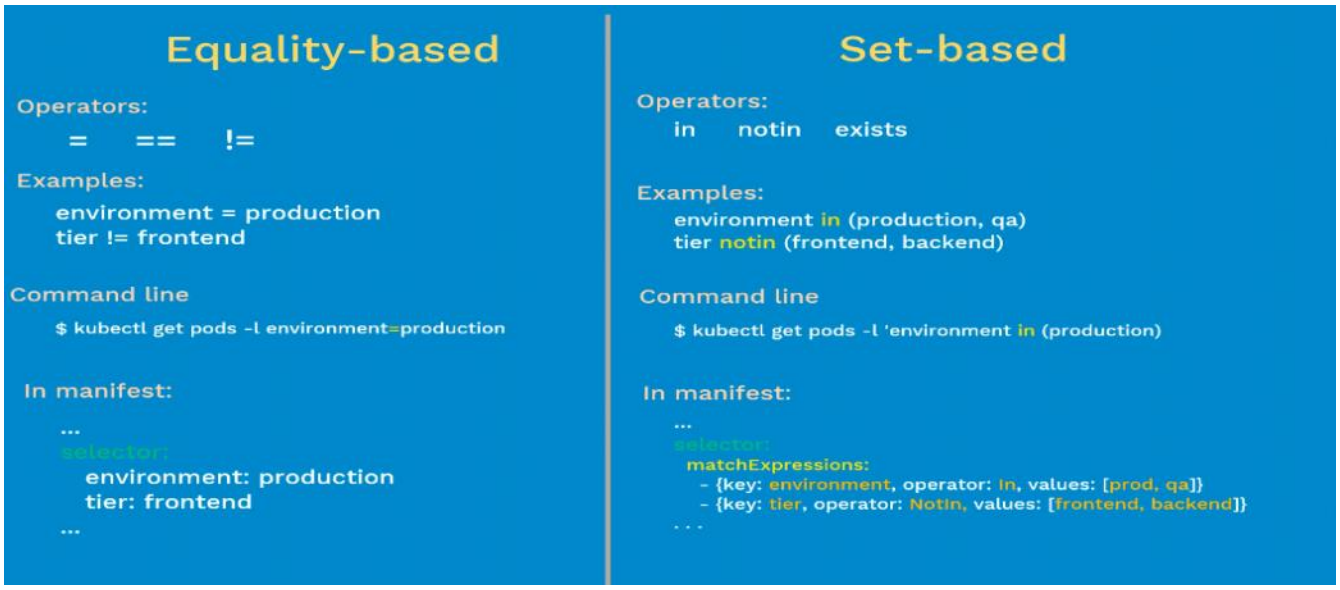
Pods

Labels

```
#Pod-Spec
apiVersion: v1
kind: pod
metadata:
  name: nginx-pod
  labels:
    app: guestbook
    tier: frontend
    env: dev
spec:
  replicas: 5
  . . . . .
```

Controllers & Services

Selectors



Equality-based

Operators:

= == !=

Examples:

```
environment = production
tier != frontend
```

Command line

```
$ kubectl get pods -l environment=production
```

In manifest:

```
...
selector:
  environment: production
  tier: frontend
...
```

Set-based

Operators:

in notin exists

Examples:

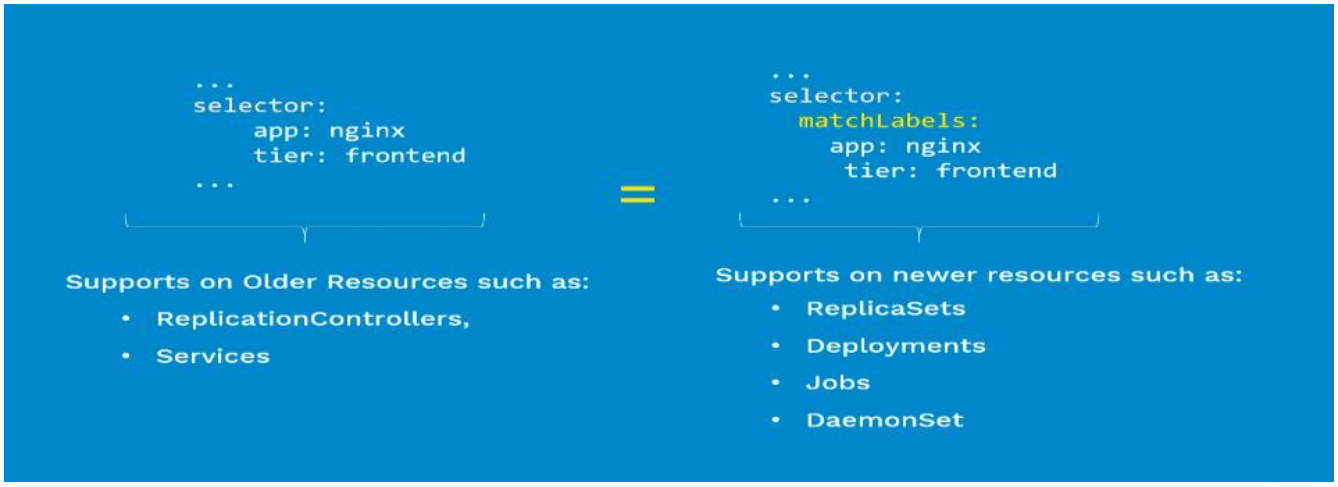
```
environment in (production, qa)
tier notin (frontend, backend)
```

Command line

```
$ kubectl get pods -l 'environment in (production)
```

In manifest:

```
...
selector:
  matchExpressions:
    - {key: environment, operator: In, values: [prod, qa]}
    - {key: tier, operator: NotIn, values: [frontend, backend]}
...
```



```
...
selector:
    app: nginx
    tier: frontend
...
```

=

```
...
selector:
  matchLabels:
    app: nginx
     tier: frontend
...
```

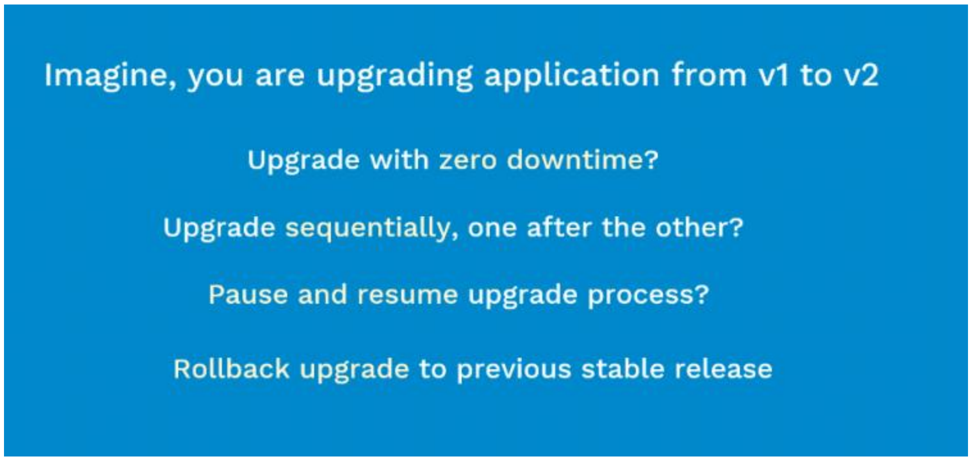
Supports on Older Resources such as:

- ReplicationControllers,
- Services

Supports on newer resources such as:

- ReplicaSets
- Deployments
- Jobs
- DaemonSet

Replica Set Demo



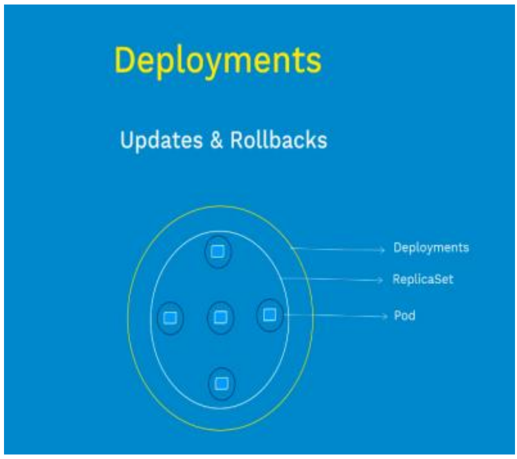
Imagine, you are upgrading application from v1 to v2

Upgrade with zero downtime?

Upgrade sequentially, one after the other?

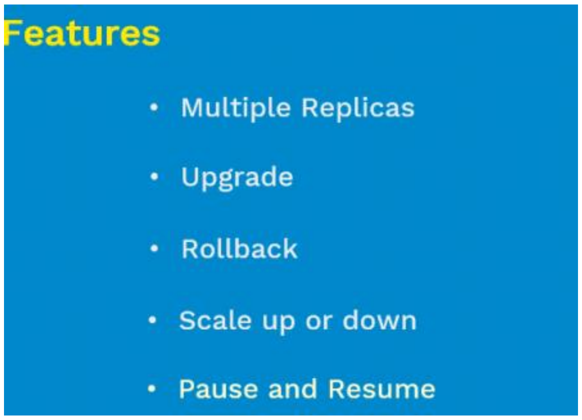
Pause and resume upgrade process?

Rollback upgrade to previous stable release



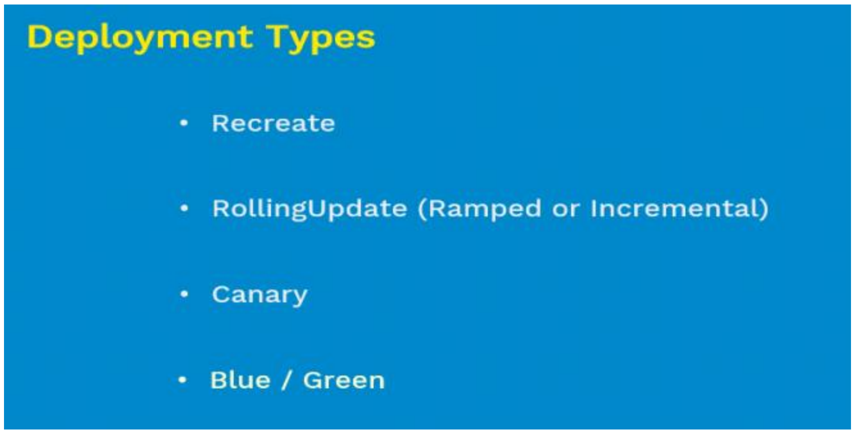
Deployments

Updates & Rollbacks



Features

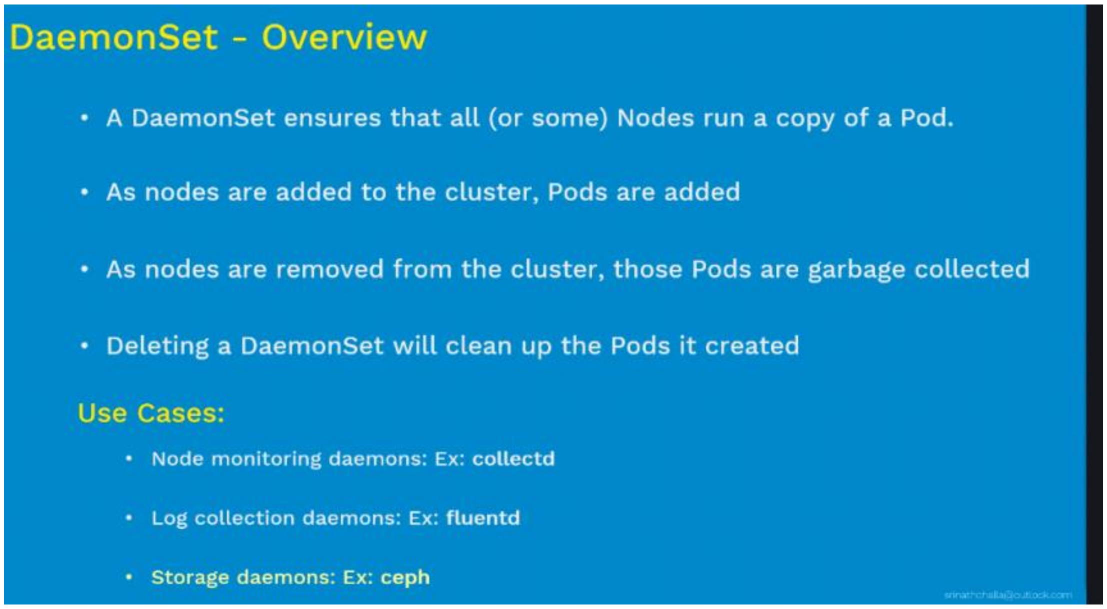
- Multiple Replicas
- Upgrade
- Rollback
- Scale up or down
- Pause and Resume



Deployment Types

- Recreate
- RollingUpdate (Ramped or Incremental)
- Canary
- Blue / Green

Deployment Demo



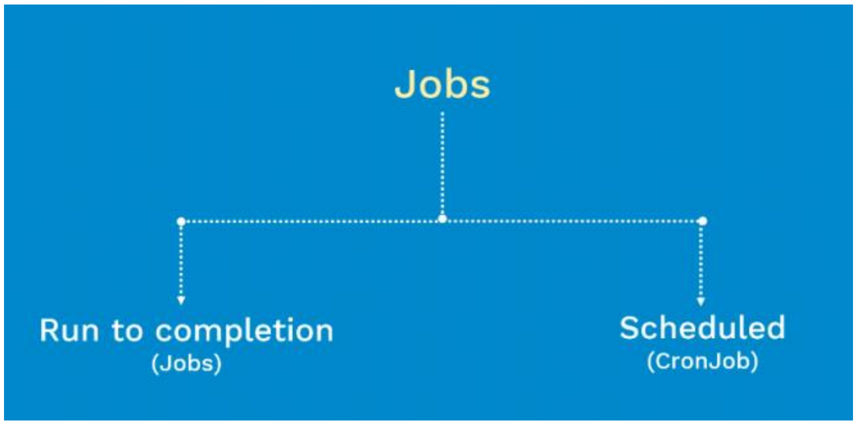
DaemonSet - Overview

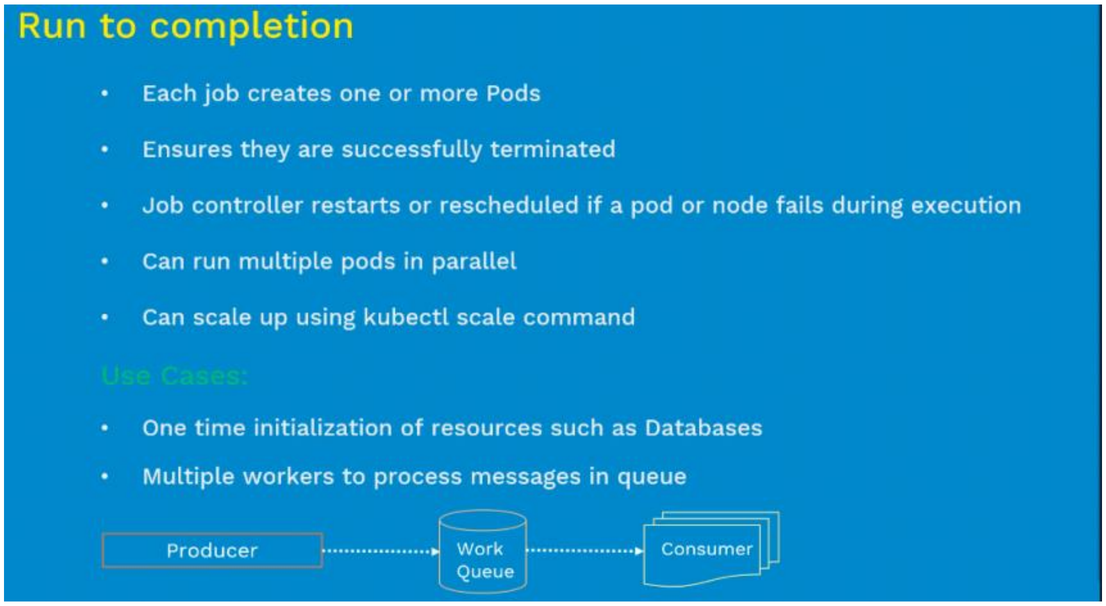
- A DaemonSet ensures that all (or some) Nodes run a copy of a Pod.
- As nodes are added to the cluster, Pods are added
- As nodes are removed from the cluster, those Pods are garbage collected
- Deleting a DaemonSet will clean up the Pods it created

Use Cases:

- Node monitoring daemons: Ex: **collectd**
- Log collection daemons: Ex: **fluentd**
- Storage daemons: Ex: **ceph**

DaemonSet Demo





Run to completion

- Each job creates one or more Pods
- Ensures they are successfully terminated
- Job controller restarts or rescheduled if a pod or node fails during execution
- Can run multiple pods in parallel
- Can scale up using kubectl scale command

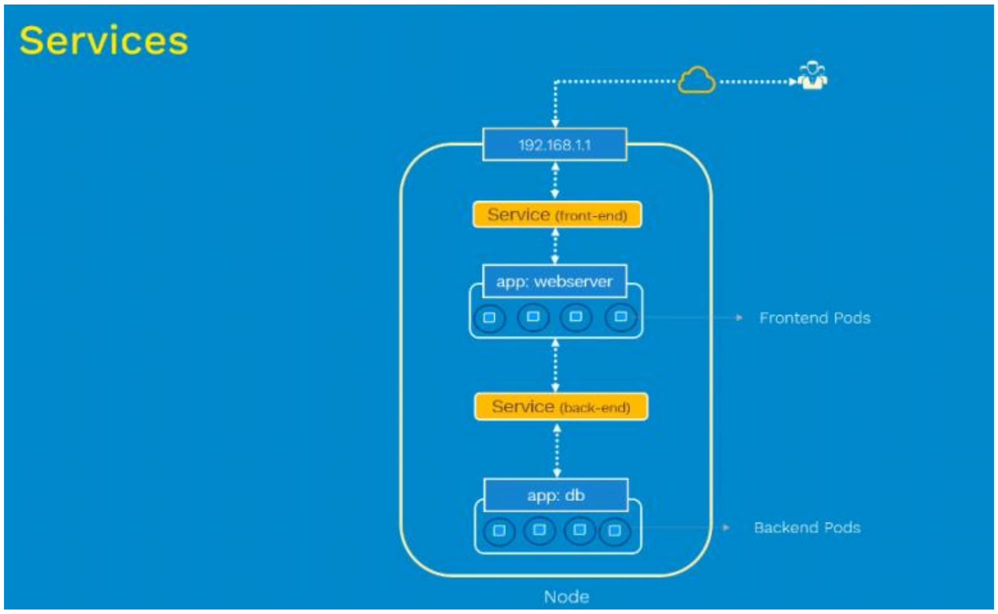
Use Cases:

- One time initialization of resources such as Databases
- Multiple workers to process messages in queue

Producer
Work Queue
Consumer

JOB Demo

Services


192.168.1.1
Service (front-end)
app: webserver
Frontend Pods
Service (back-end)
app: db
Backend Pods
Node