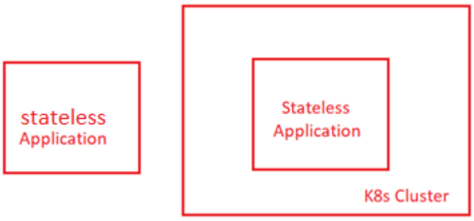
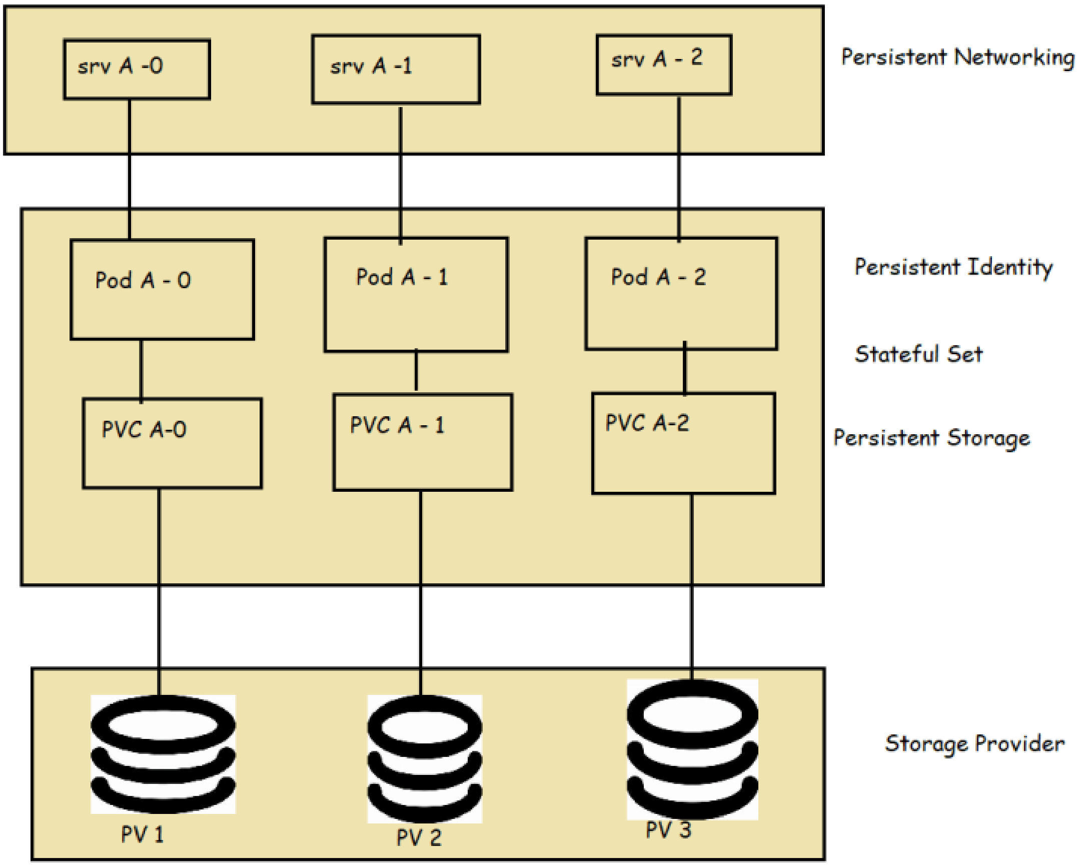
Stateful Service

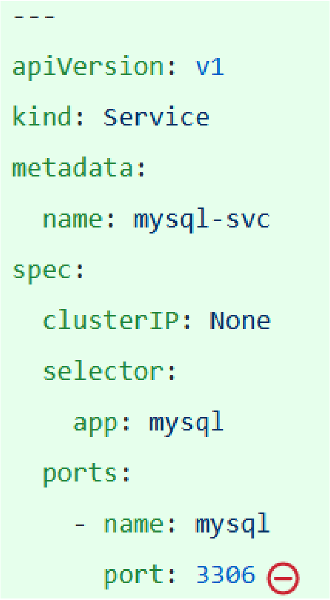
- Overview: Distributed Stateful applications require features such as persisted identity, networking, storage.
- Problem:
  - In real world behind every highly scalable stateless service is a stateful service typically in the shape of datastore
  - In early days of k8s when it lacked support of stateful workloads, the solution was to place stateless applications in k8s cluster and stateful applications outside cluster.



  - Single instance stateful application => We create replica set and also Persistent Volume Claims & Persistent Volumes.
- Solution:
  - K8S has StatefulSets that provides managing stateful applications.

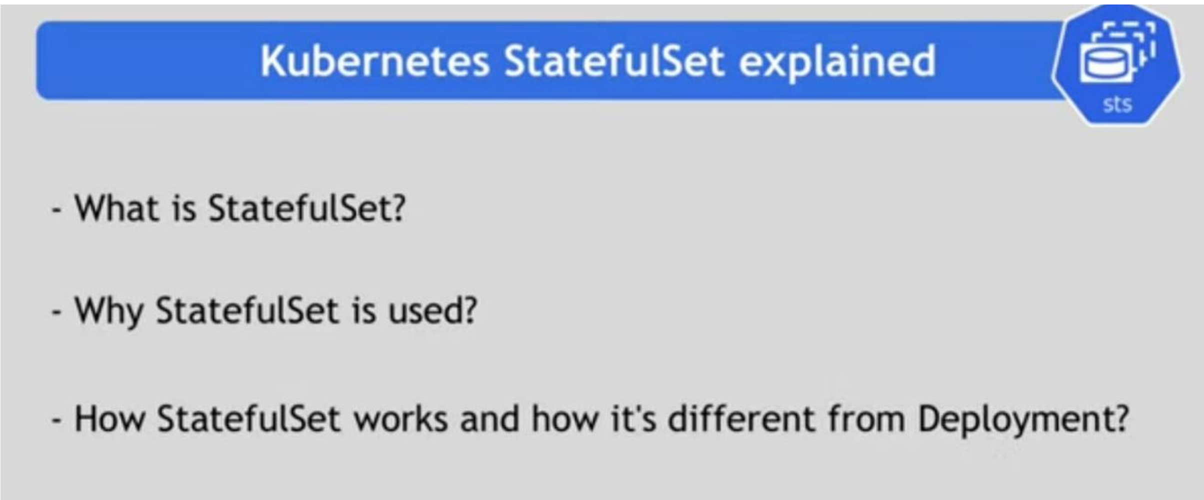


```
---
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: mysql-ss
spec:
  serviceName: mysql-svc
  replicas: 2
  selector:
    matchLabels:
      app: mysql
  template:
    metadata:
      labels:
        app: mysql
    spec:
      containers:
        - image: mysql
          name: mysql
          env:
            - name: MYSQL_ROOT_PASSWORD
              value: qwert456
          ports:
            - containerPort: 3306
          volumeMounts:
            - name: mysql-store
              mountPath: /var/lib/mysql
  volumeClaimTemplates:
    - metadata:
        name: mysql-store
      spec:
        accessModes: ["ReadWriteOnce"]
        resources:
          requests:
            storage: 100Mi
```



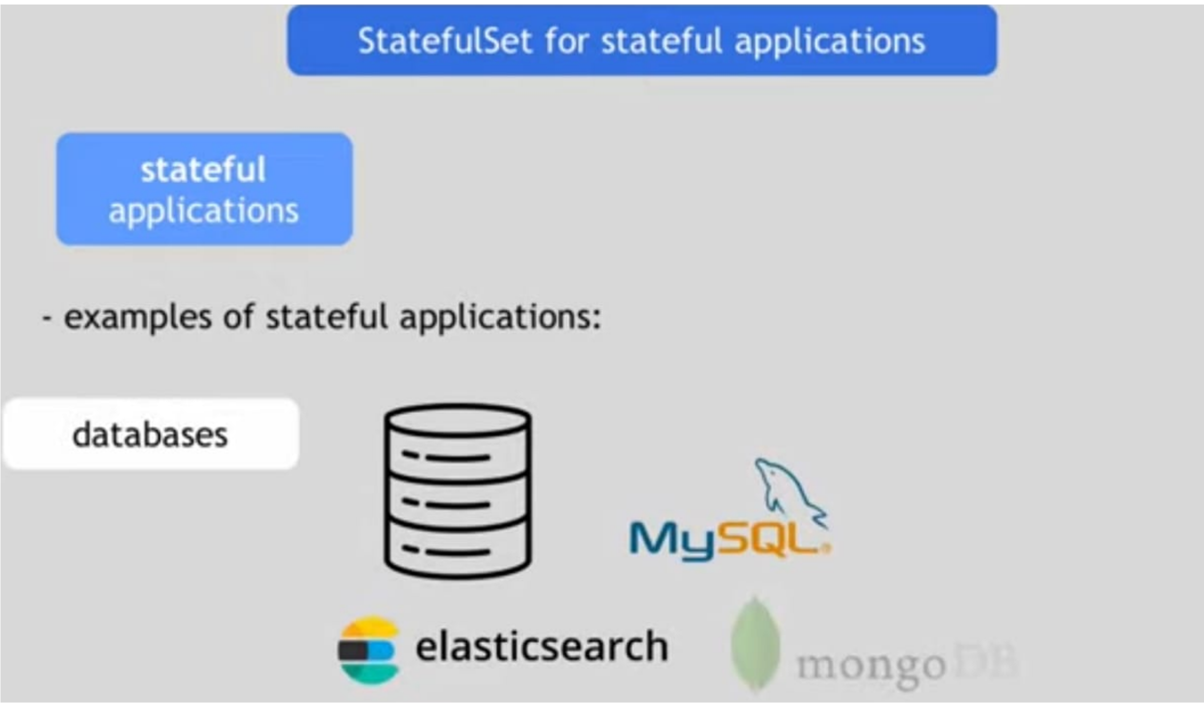
```
---
apiVersion: v1
kind: Service
metadata:
  name: mysql-svc
spec:
  clusterIP: None
  selector:
    app: mysql
  ports:
    - name: mysql
      port: 3306
```

```yaml
---
apiVersion: v1
kind: Pod
metadata:
  name: test
spec:
  containers:
    - image: alpine
      name: alpine
      args: ["sleep", "1d"]
```



Kubernetes StatefulSet explained

- What is StatefulSet?
- Why StatefulSet is used?
- How StatefulSet works and how it's different from Deployment?

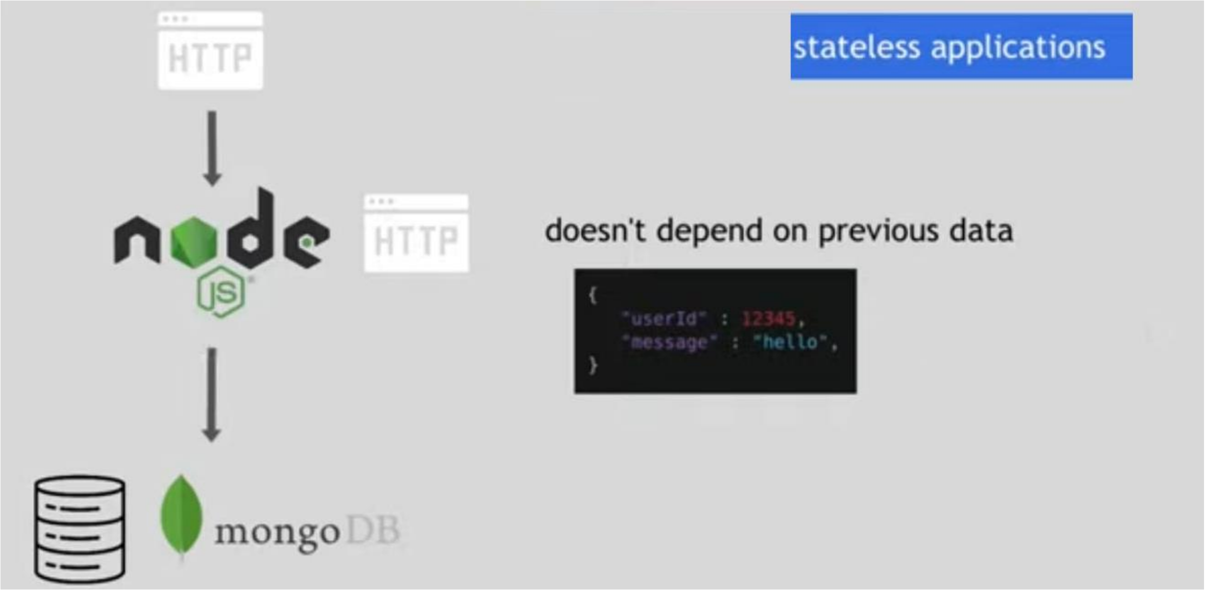


StatefulSet for stateful applications

stateful applications

- examples of stateful applications:

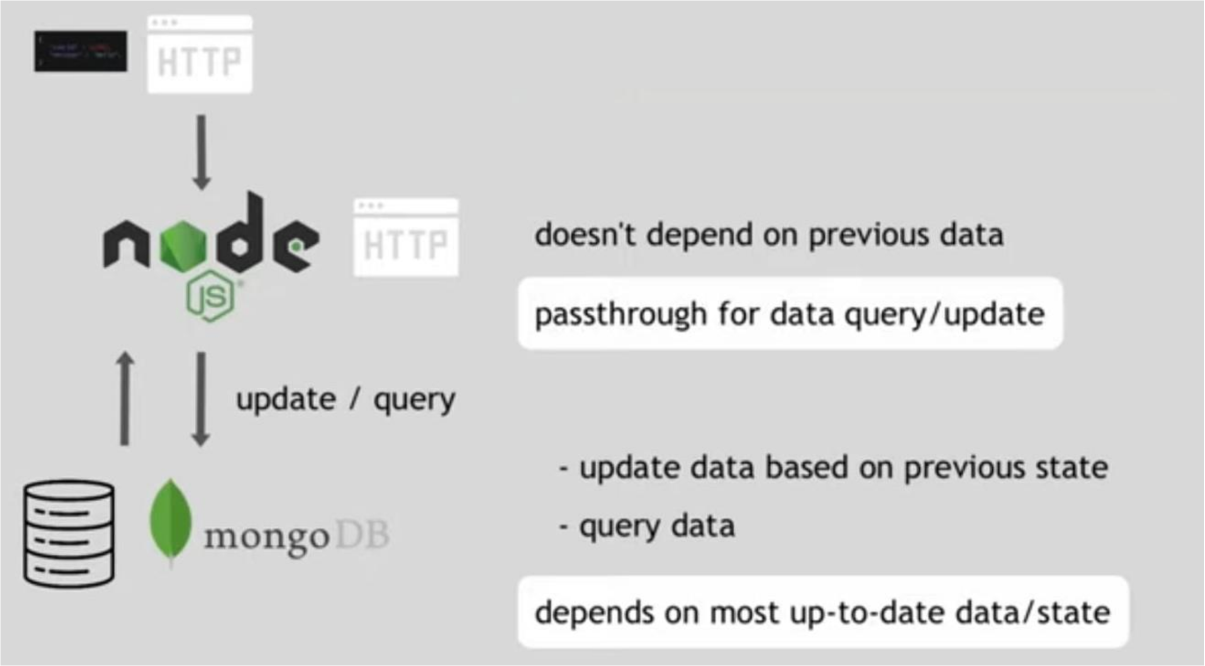
databases



stateless applications

doesn't depend on previous data

```
{
    "userId" : 12345,
    "message" : "hello",
}
```

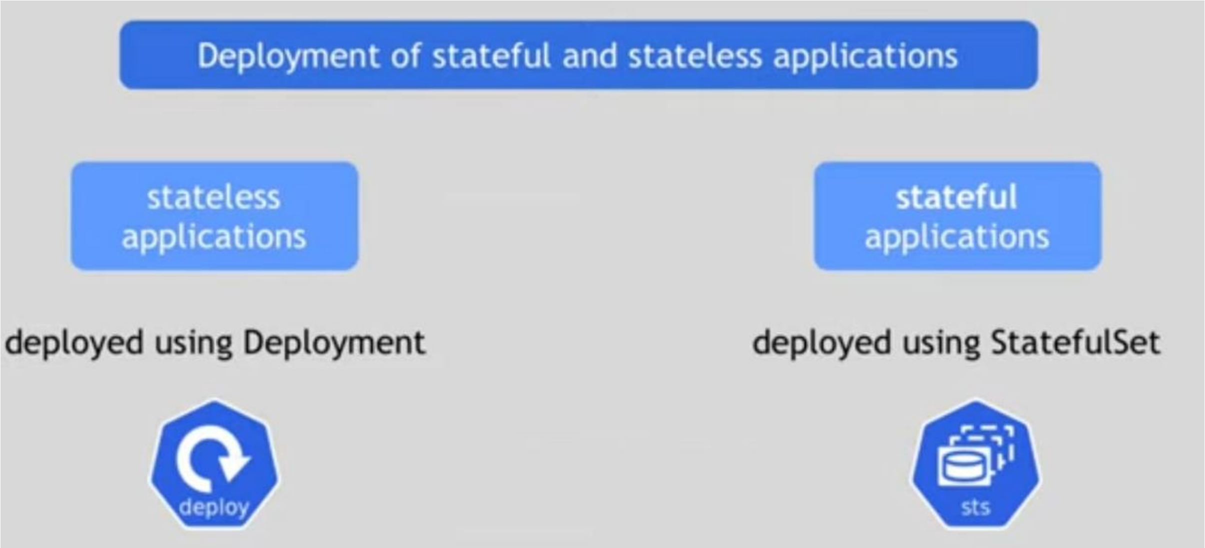


doesn't depend on previous data

passthrough for data query/update

- update data based on previous state
- query data

depends on most up-to-date data/state



Deployment of stateful and stateless applications

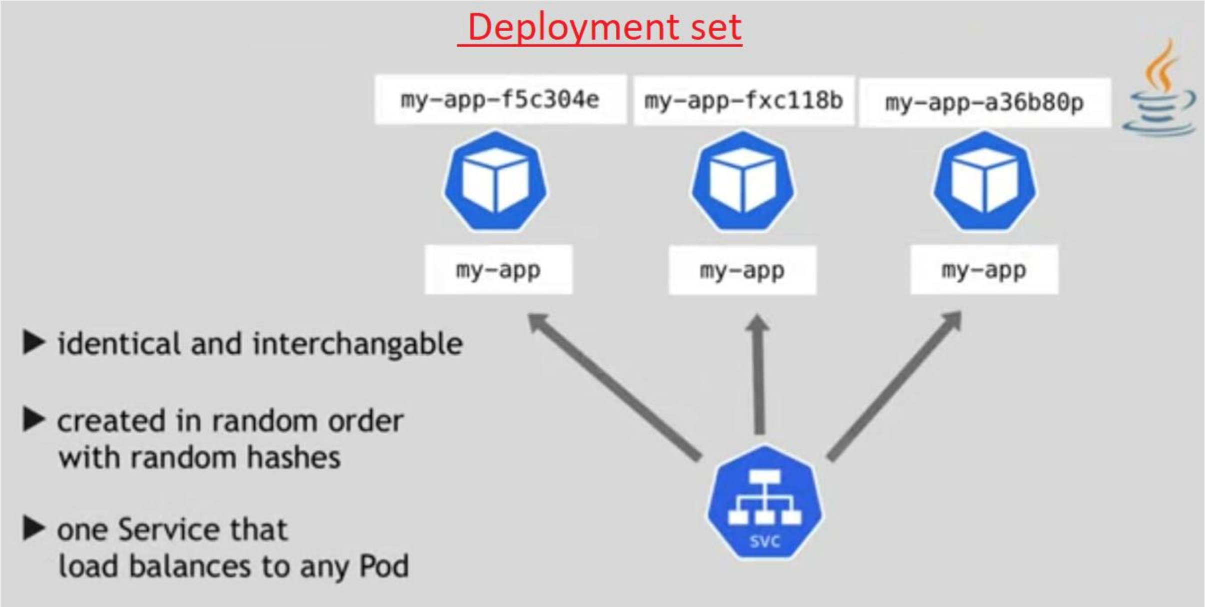
stateless applications

deployed using Deployment

stateful applications

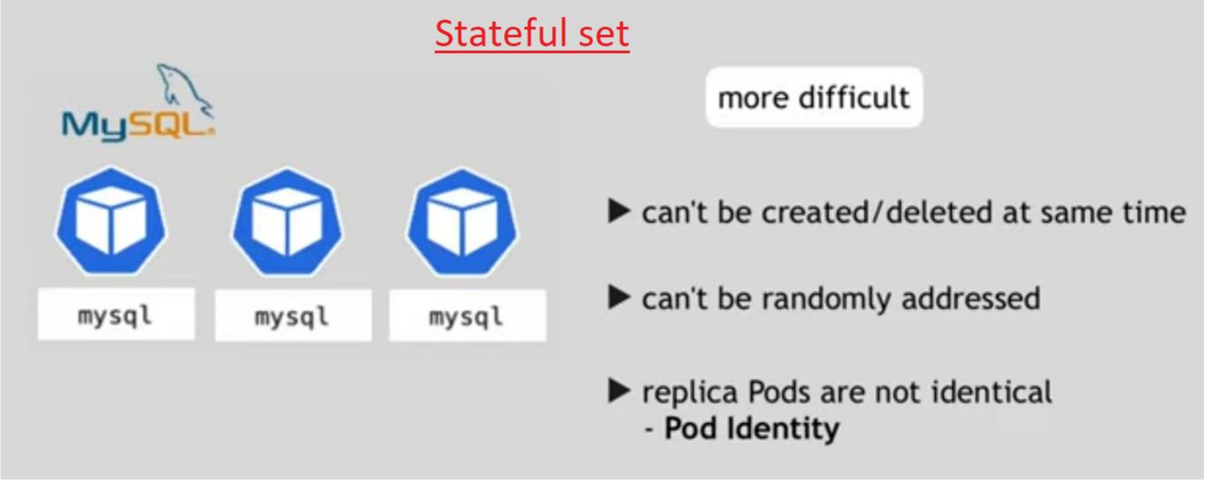
deployed using StatefulSet

Deployment set



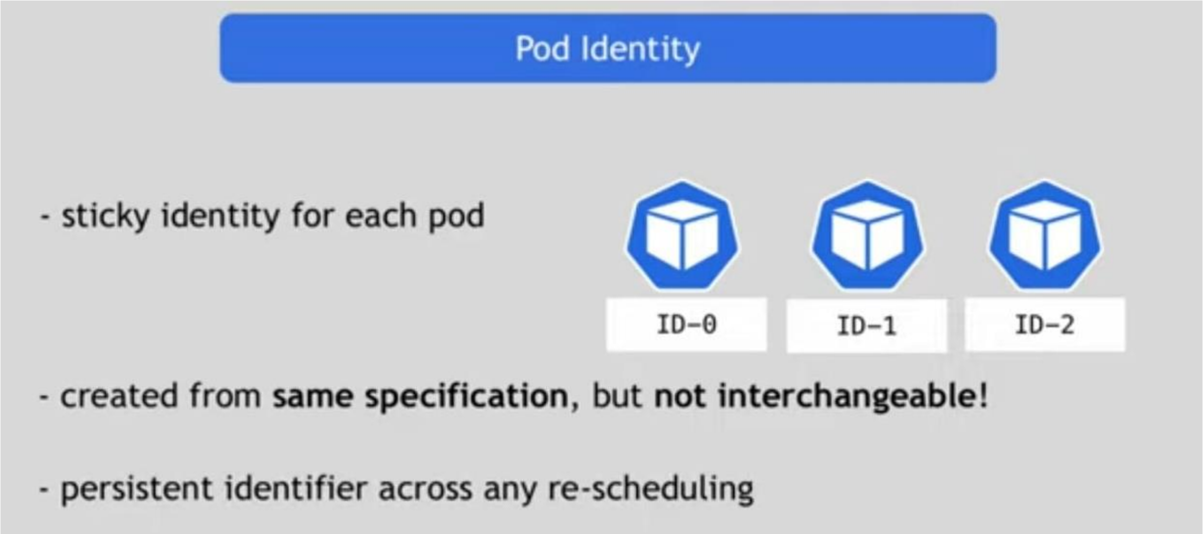
- identical and interchangable
- created in random order with random hashes
- one Service that load balances to any Pod

Stateful set



more difficult

- can't be created/deleted at same time
- can't be randomly addressed
- replica Pods are not identical
  - **Pod Identity**



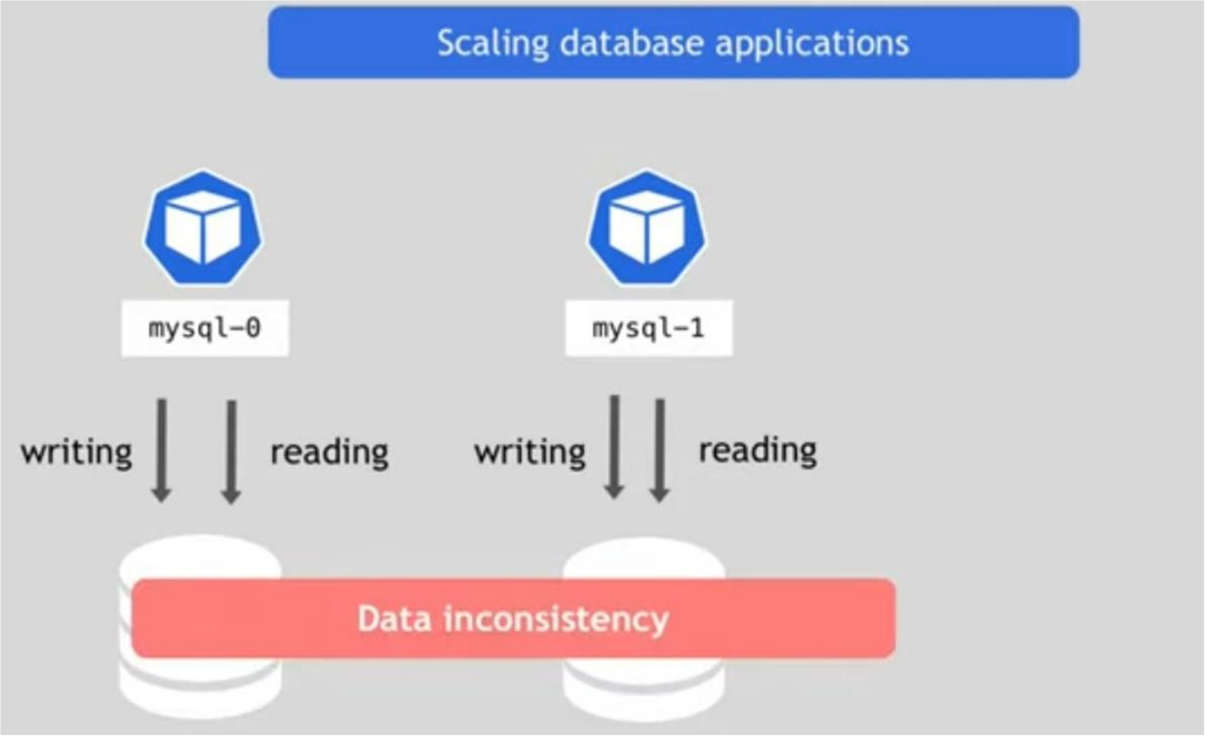
Pod Identity

- sticky identity for each pod

ID-0 ID-1 ID-2

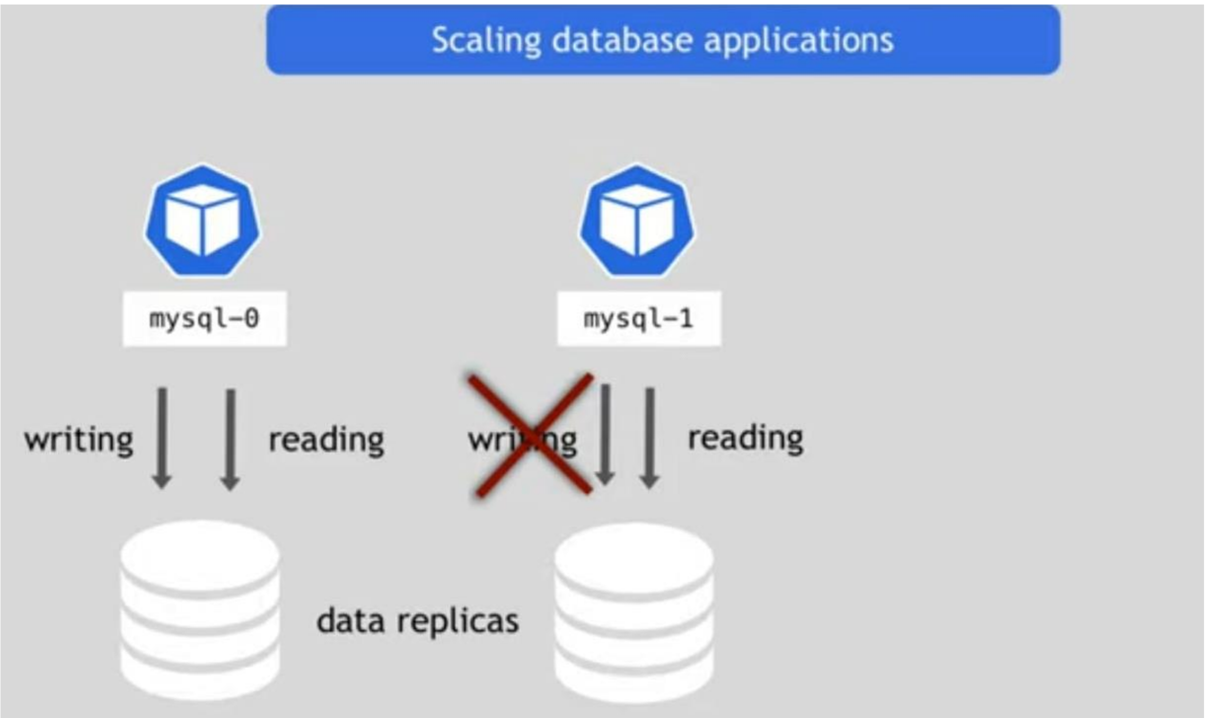
- created from **same specification**, but **not interchangeable!**

- persistent identifier across any re-scheduling



Scaling database applications

Data inconsistency



Scaling database applications

mysql-0

writing

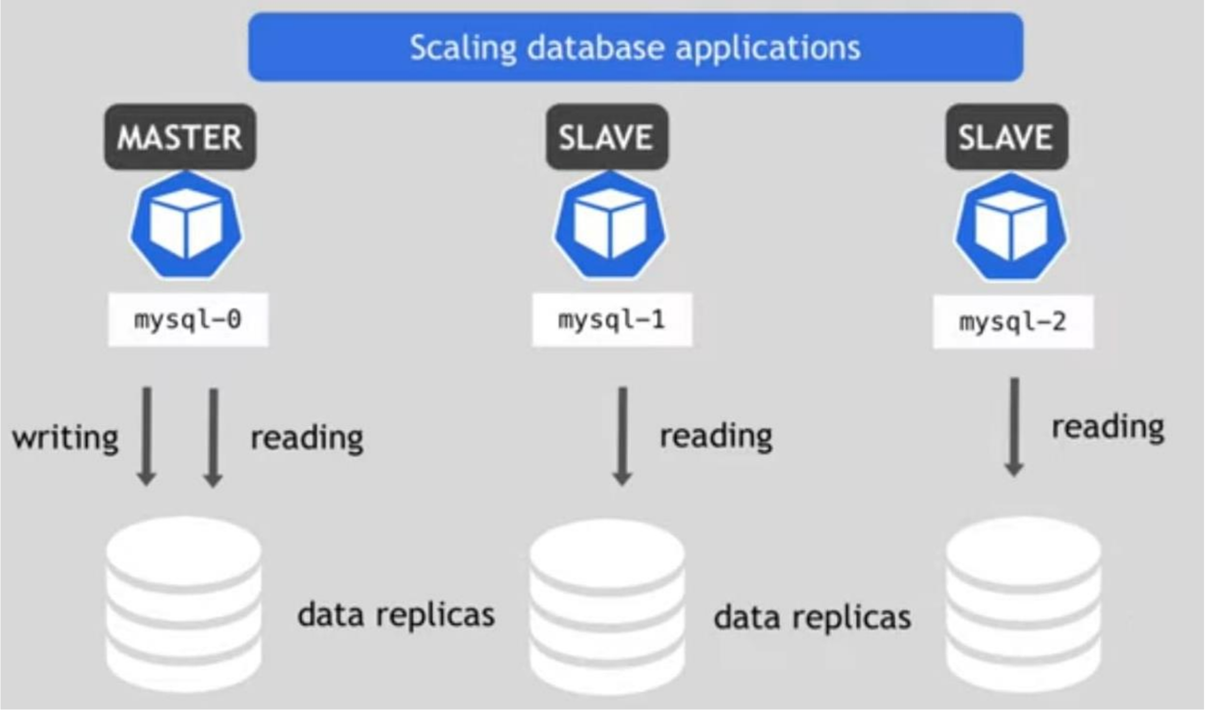
reading

mysql-1

~~writing~~

reading

data replicas



Scaling database applications

MASTER

mysql-0

writing

reading

SLAVE

mysql-1

reading

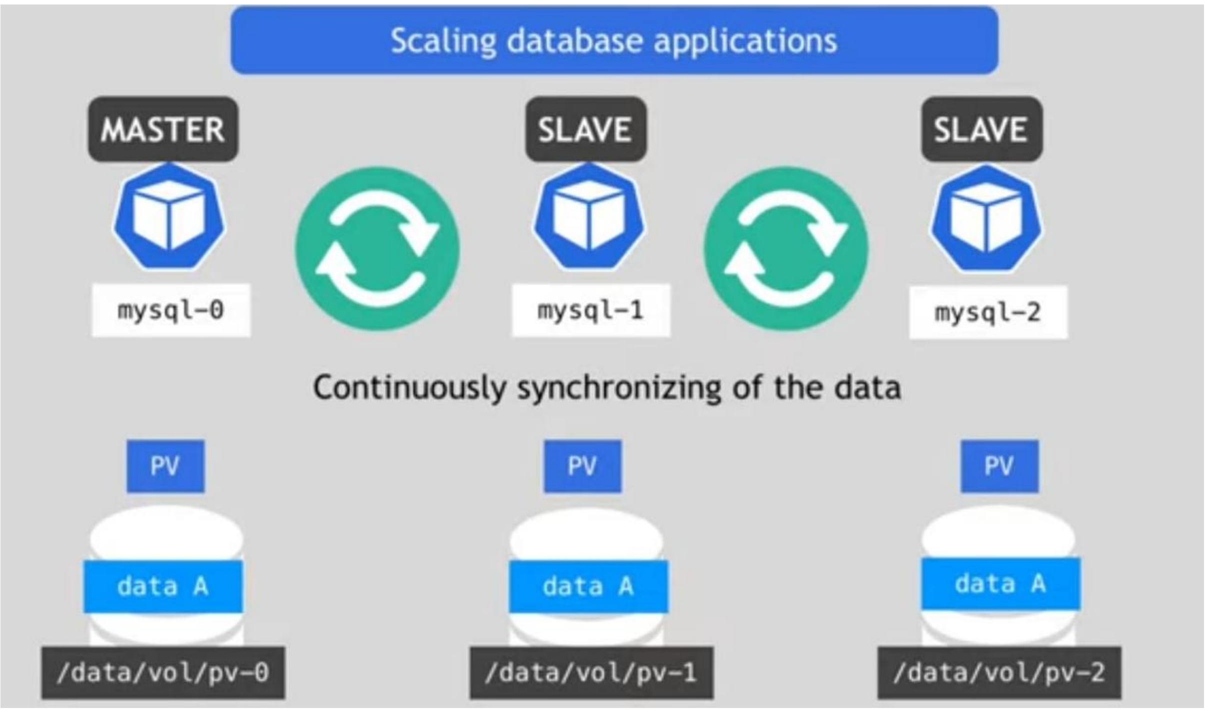
SLAVE

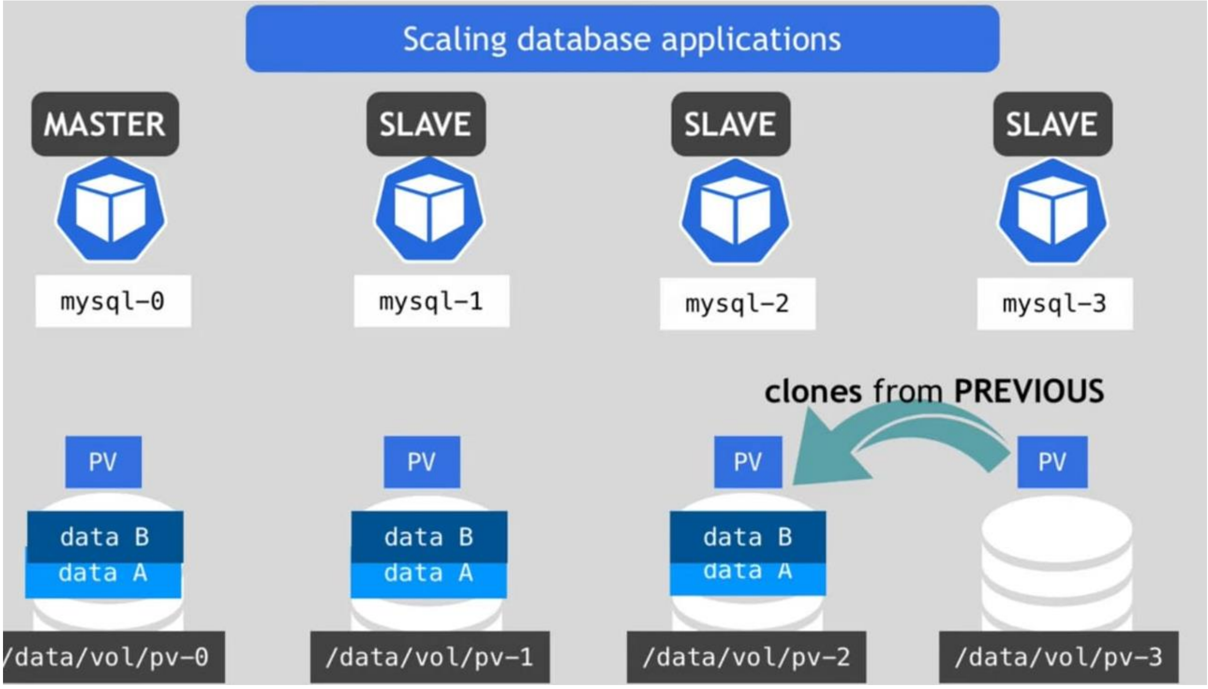
mysql-2

reading

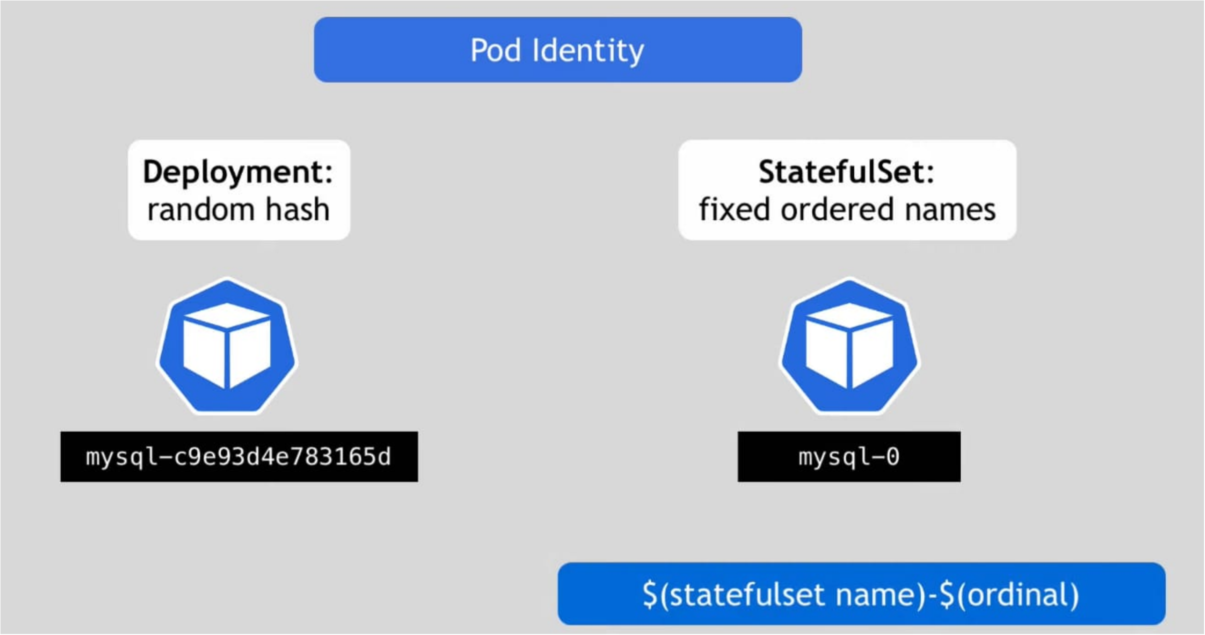
data replicas

data replicas





Always, new pod copy data from previous pod.



Pod Identity

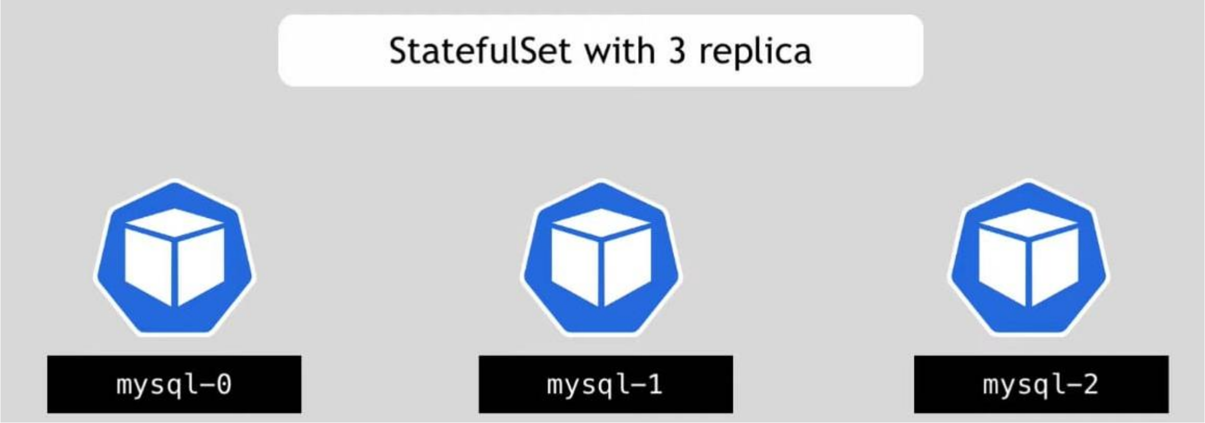
Deployment:
random hash

mysql-c9e93d4e783165d

StatefulSet:
fixed ordered names

mysql-0

$(statefulset name)-$(ordinal)

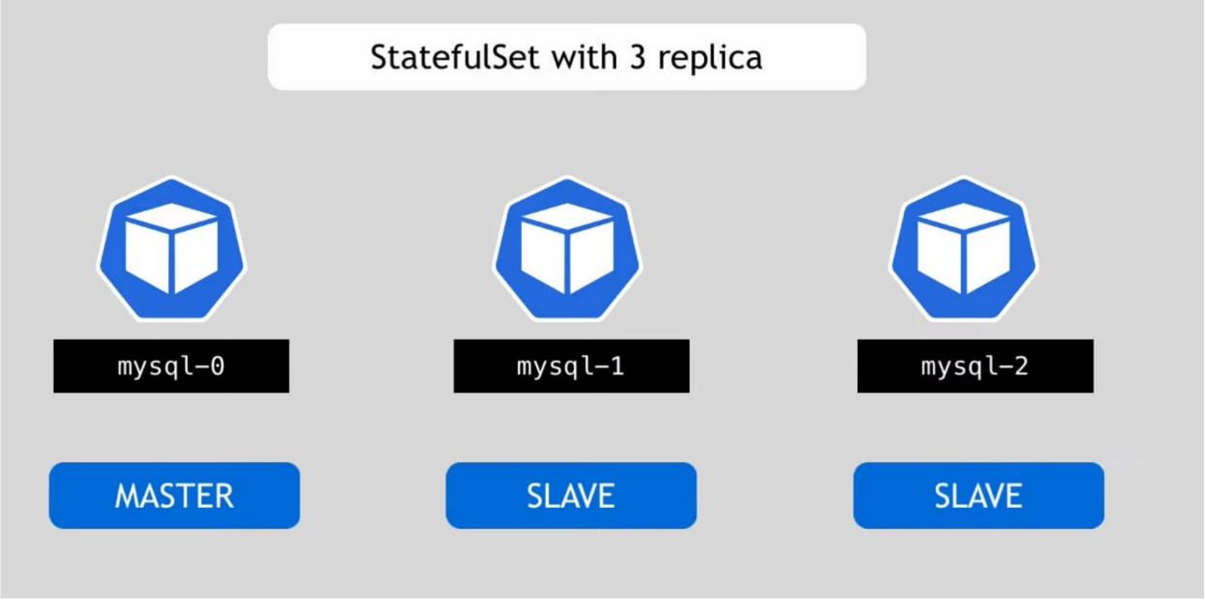


StatefulSet with 3 replica

mysql-0

mysql-1

mysql-2



StatefulSet with 3 replica

mysql-0

mysql-1

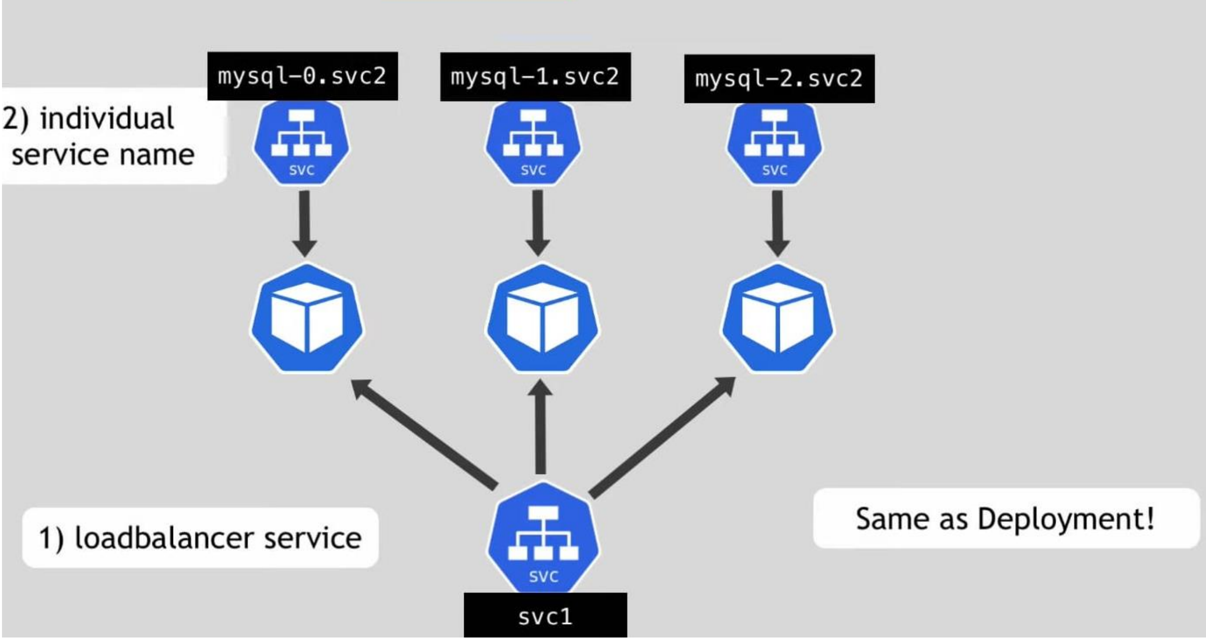
mysql-2

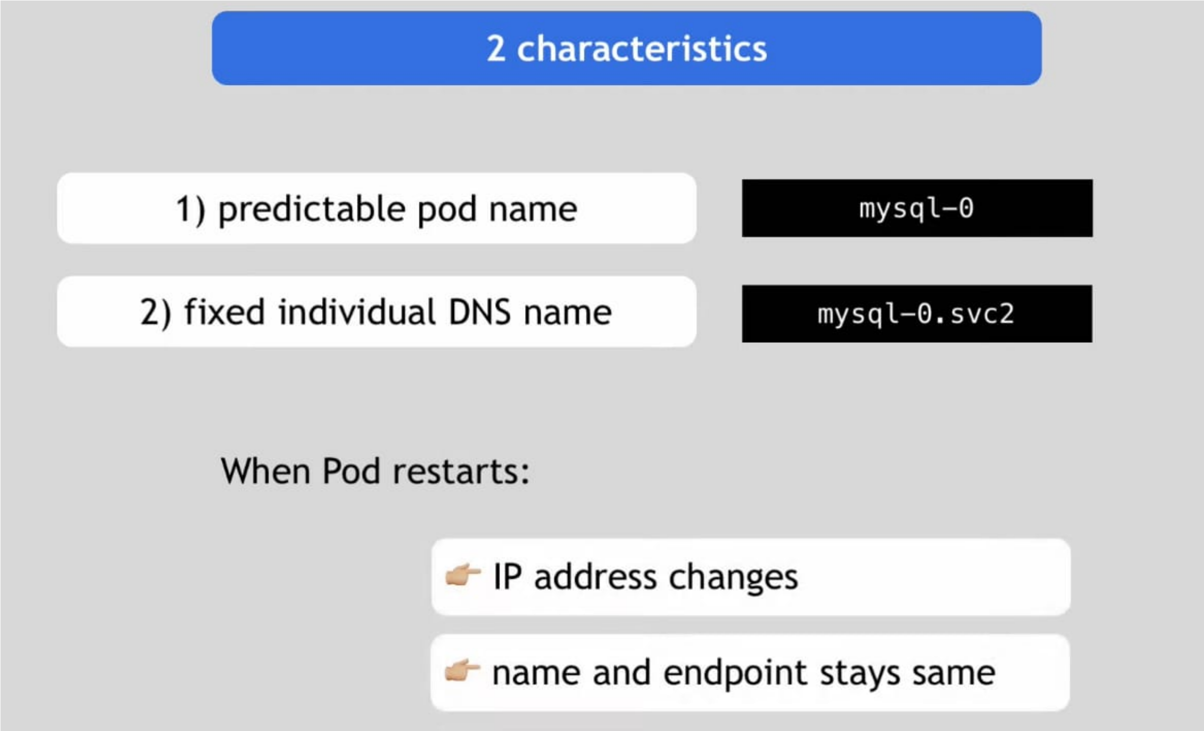
MASTER

SLAVE

SLAVE

Stateful set



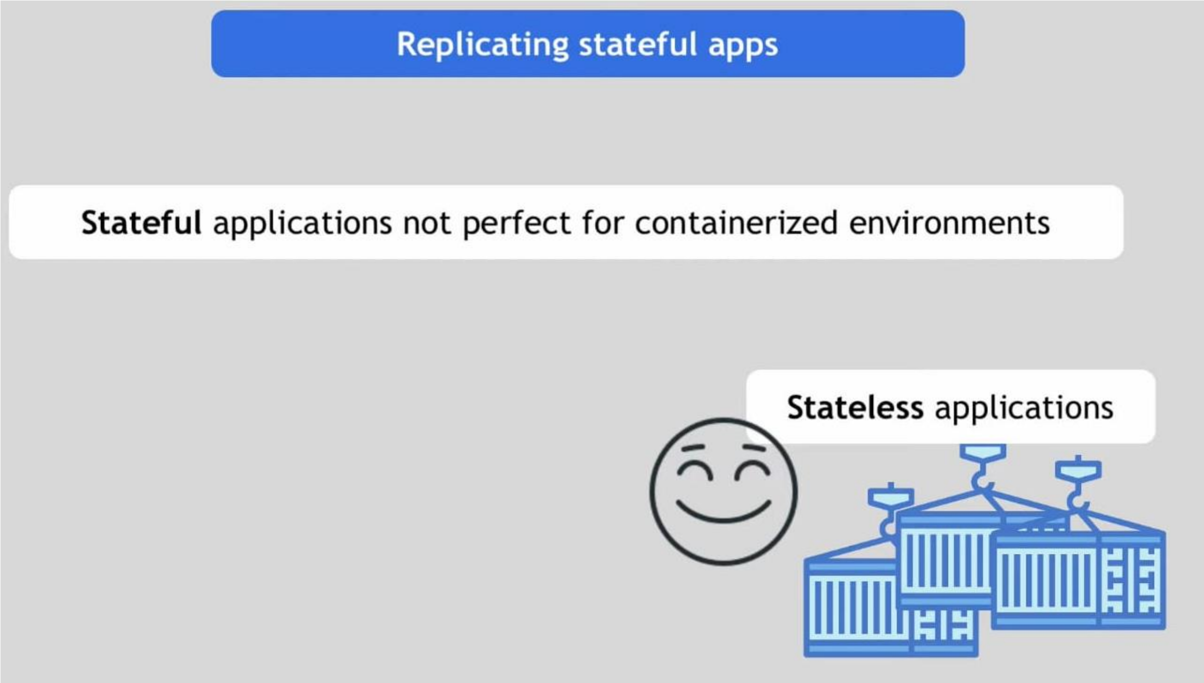


2 characteristics

1) predictable pod name	`mysql-0`
2) fixed individual DNS name	`mysql-0.svc2`

When Pod restarts:

- 👉 IP address changes
- 👉 name and endpoint stays same



Replicating stateful apps

Stateful applications not perfect for containerized environments

Stateless applications