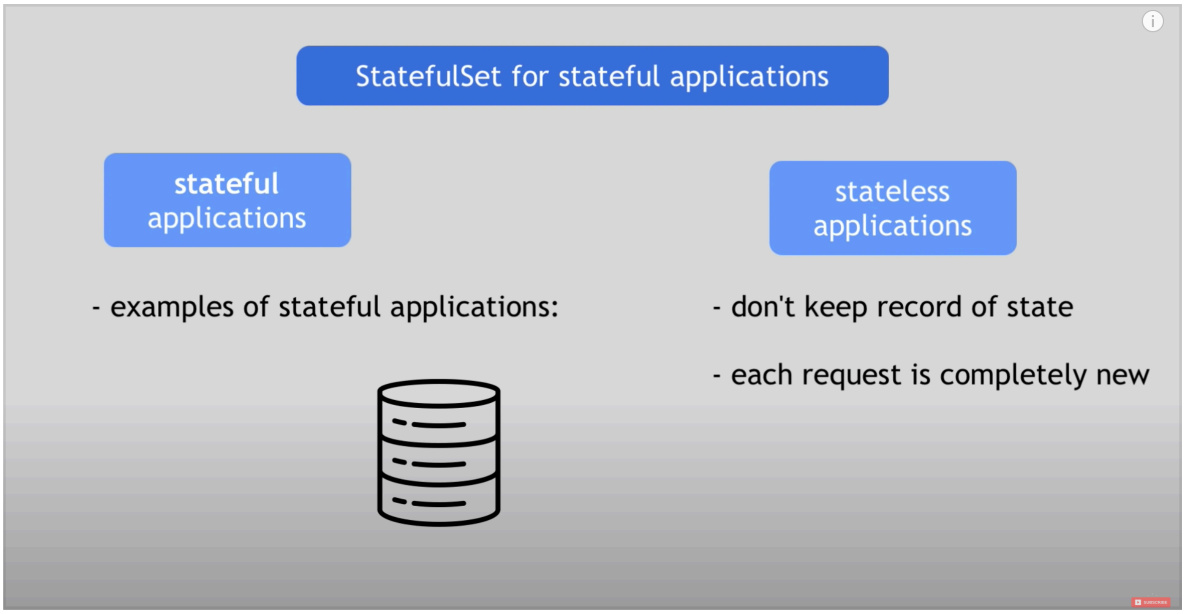
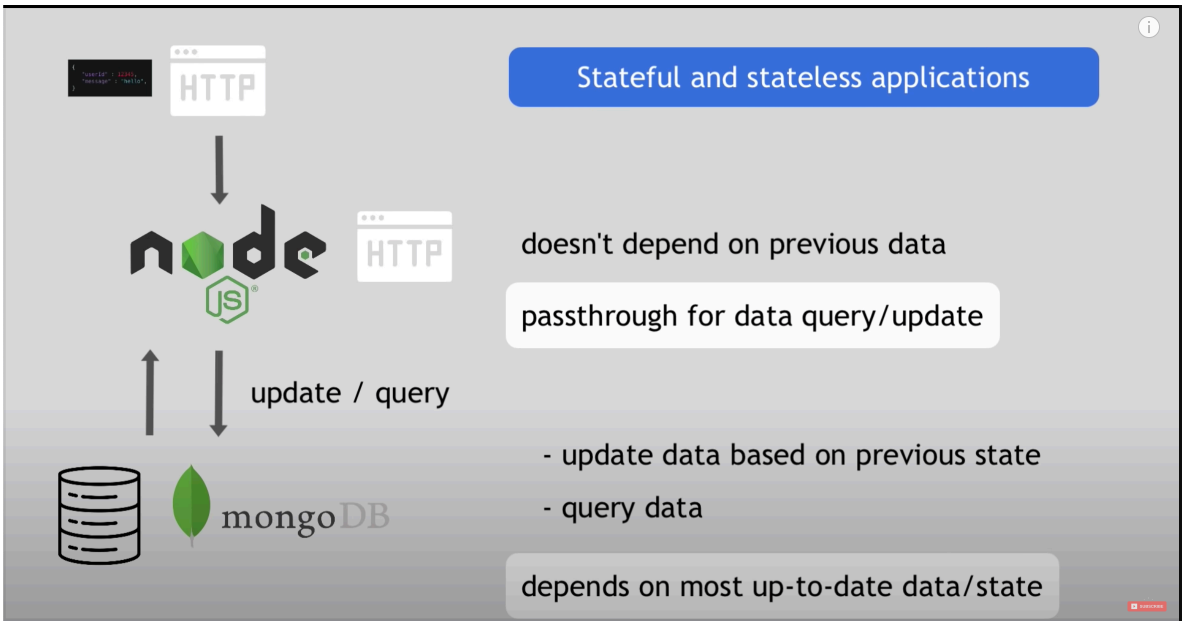
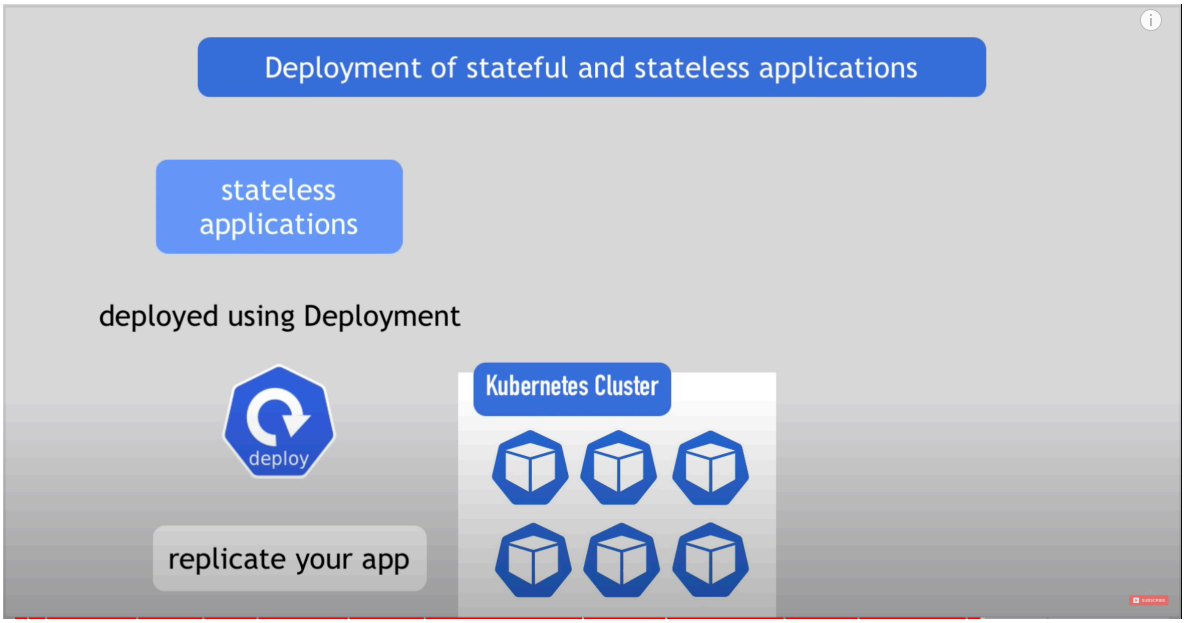
Kubernets StatefulSet

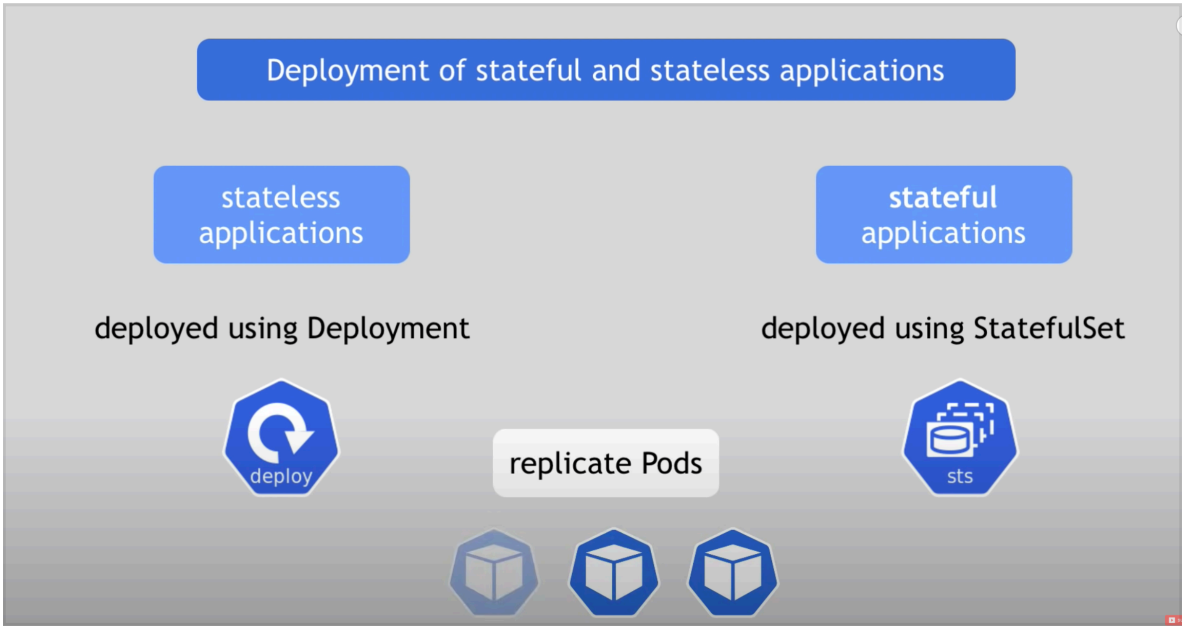
- What id StatefulSet? And Why StatefulSet is used?
  - It is a k8s component which is specifically used for stateful applications.
  - Like dbs, Apps that stores data and keeps track of it and



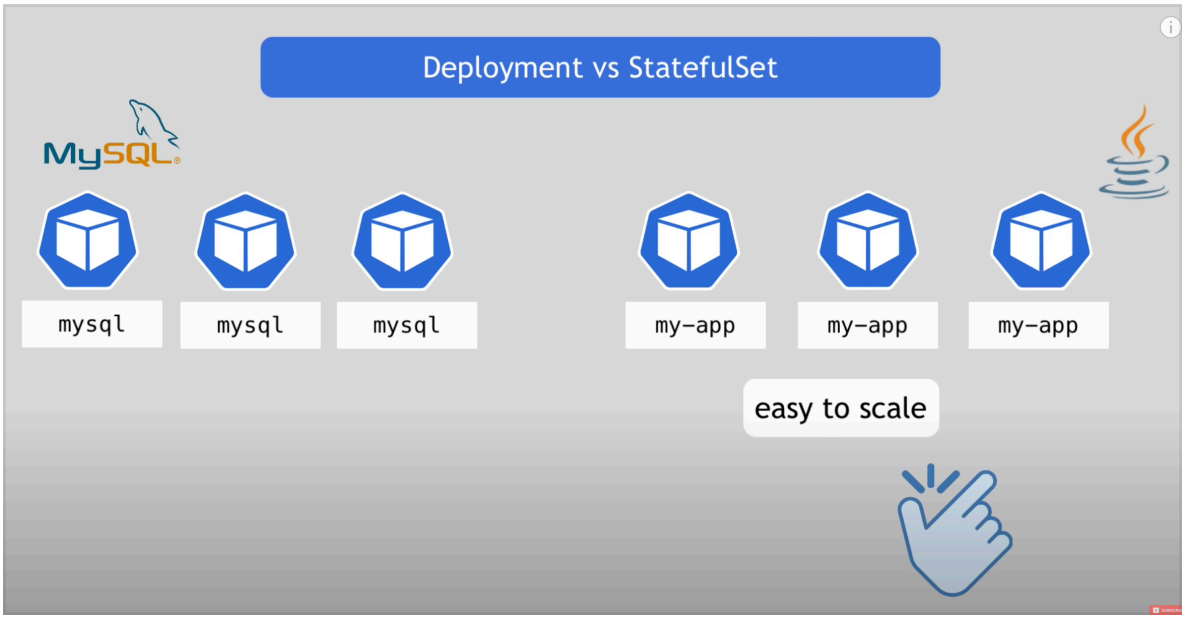


  - Because of this difference , they are both deployment different ways using different components

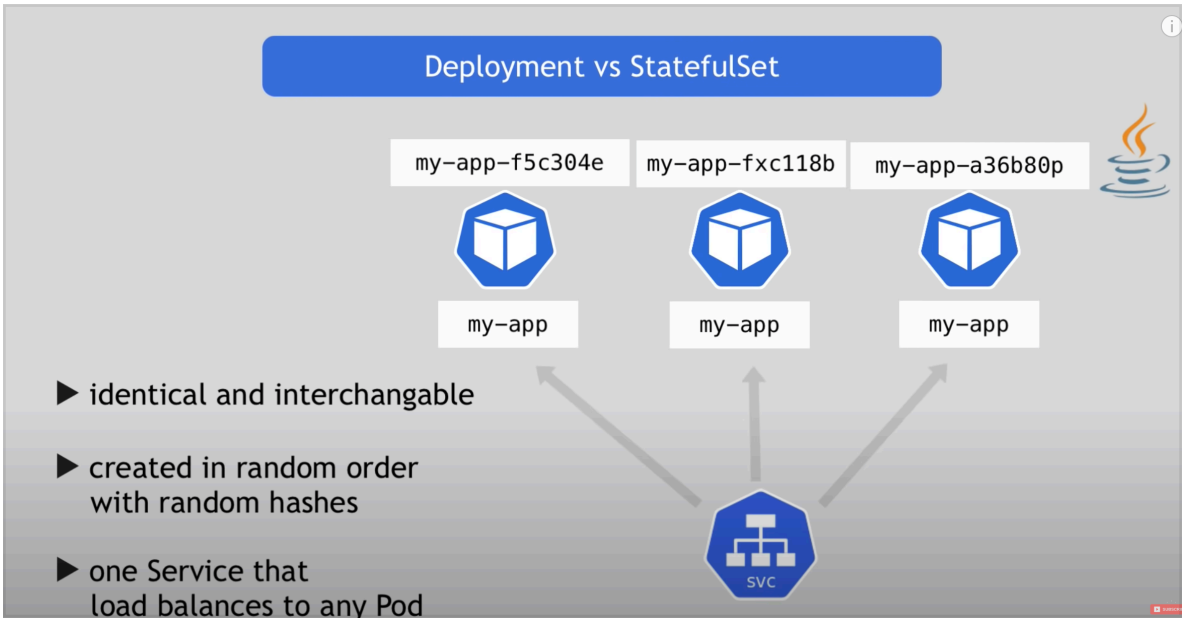




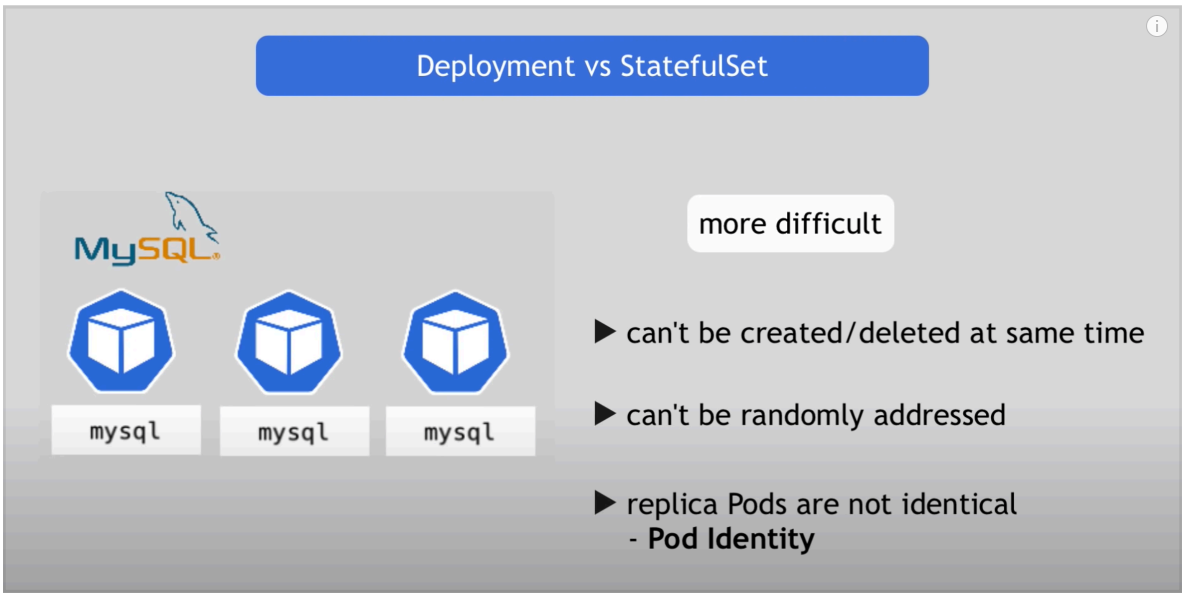
  - Bothe manage Pods based on container specification
  - Configure storage the same way
  - The question that could arise is what is the difference b/n them then, well replicationg statefulset is more difficult and has other requirements
- How it works and how its different from Deployment
  - Consider this


Deployment vs StatefulSet
MySQL
mysql
mysql
mysql
my-app
my-app
my-app
easy to scale

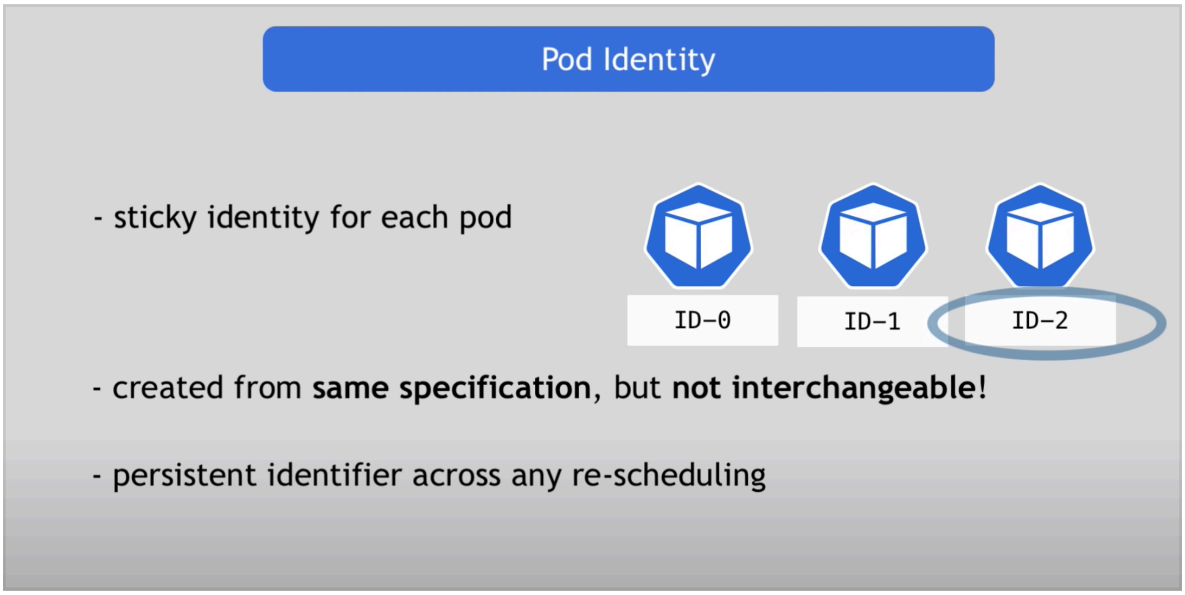
And


Deployment vs StatefulSet
my-app-f5c304e
my-app-fxc118b
my-app-a36b80p
my-app
my-app
my-app
identical and interchangable
created in random order with random hashes
one Service that load balances to any Pod
SVC

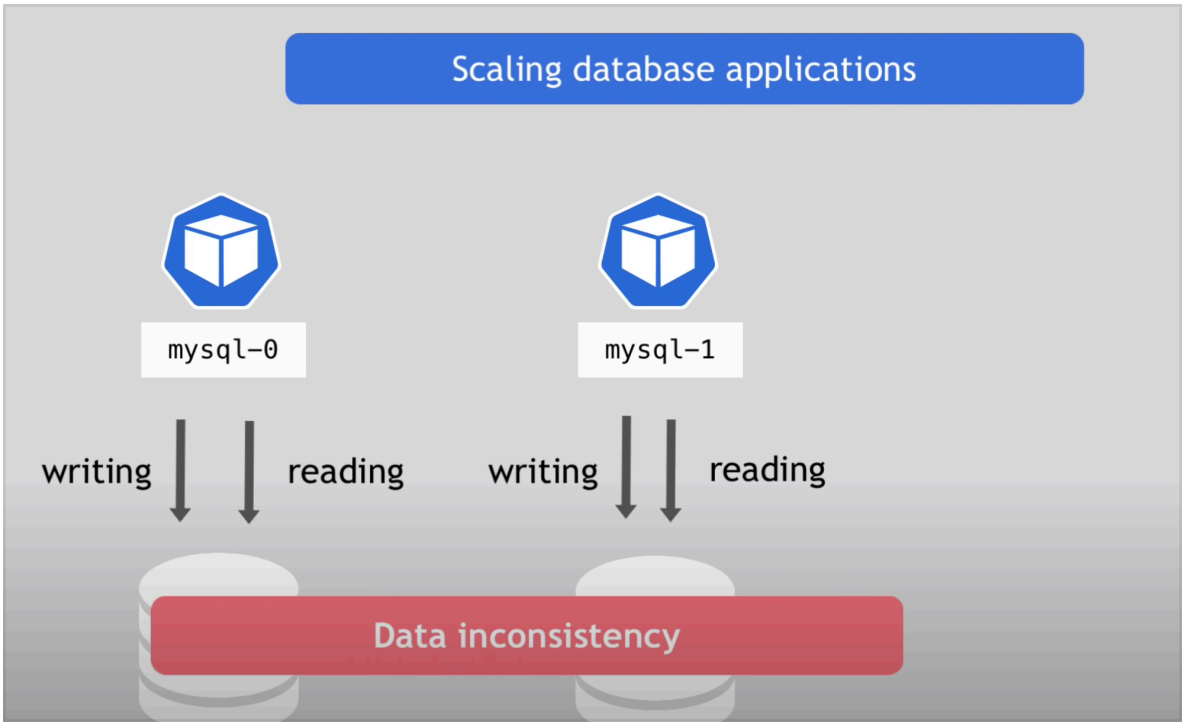
And



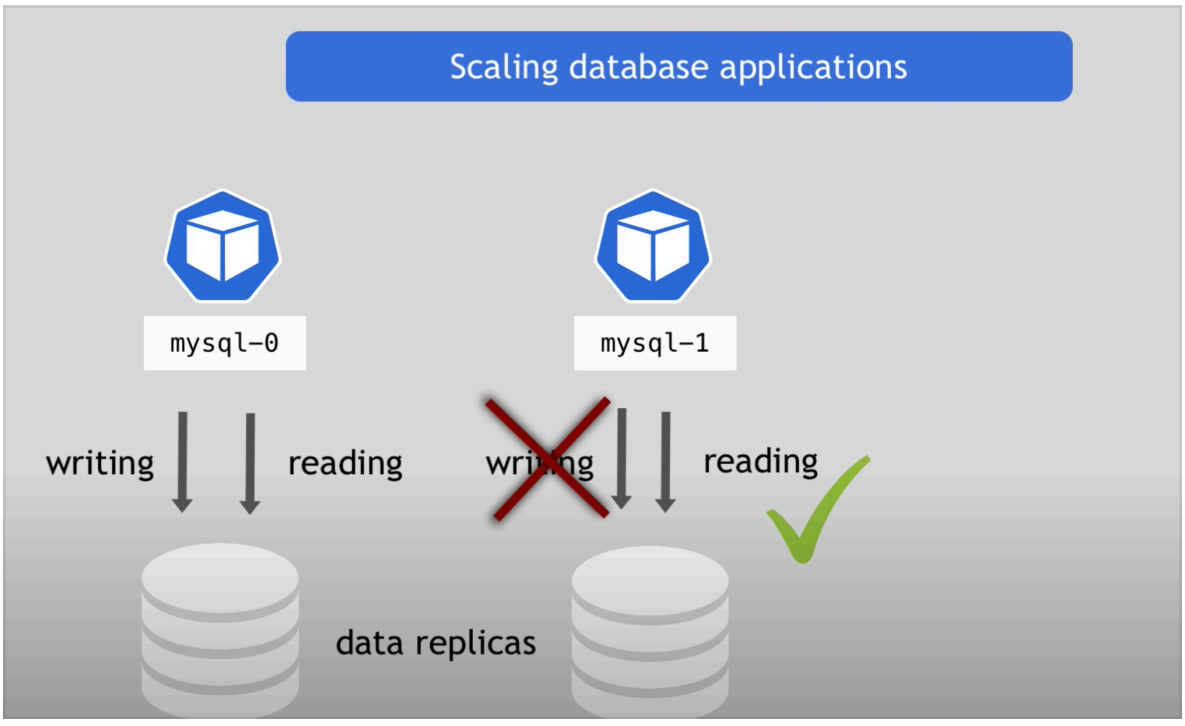
- They cannot be randomly addressed and the reason for the is replica pos are not identical - Pod identity



- Why is this identity necessary



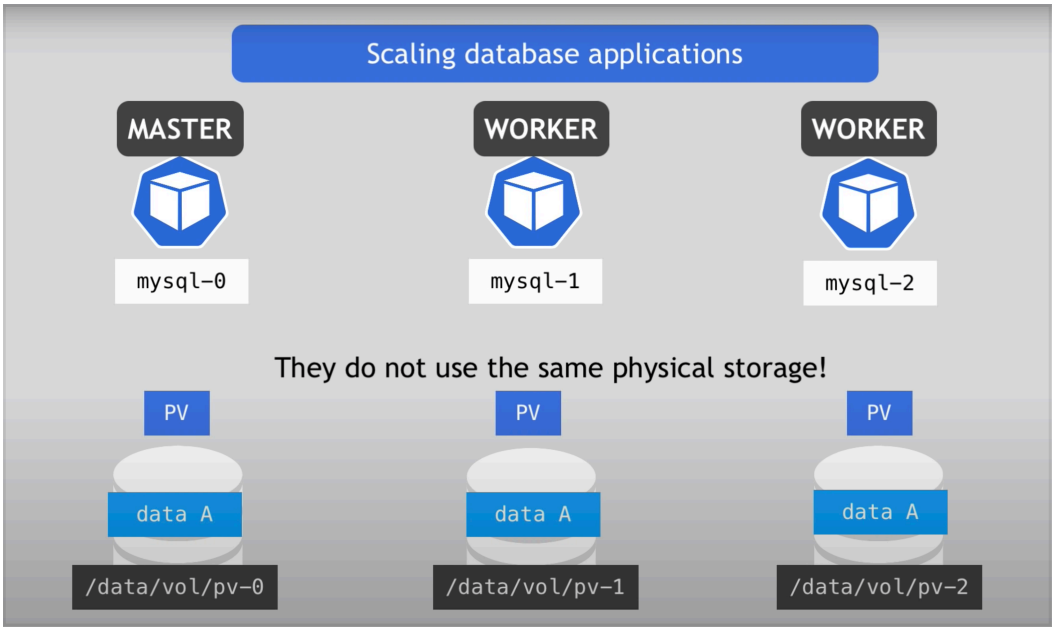
- Reading at the same time is allowed from multiple pods but not for writing, it is only allowed to the one pod at a time.

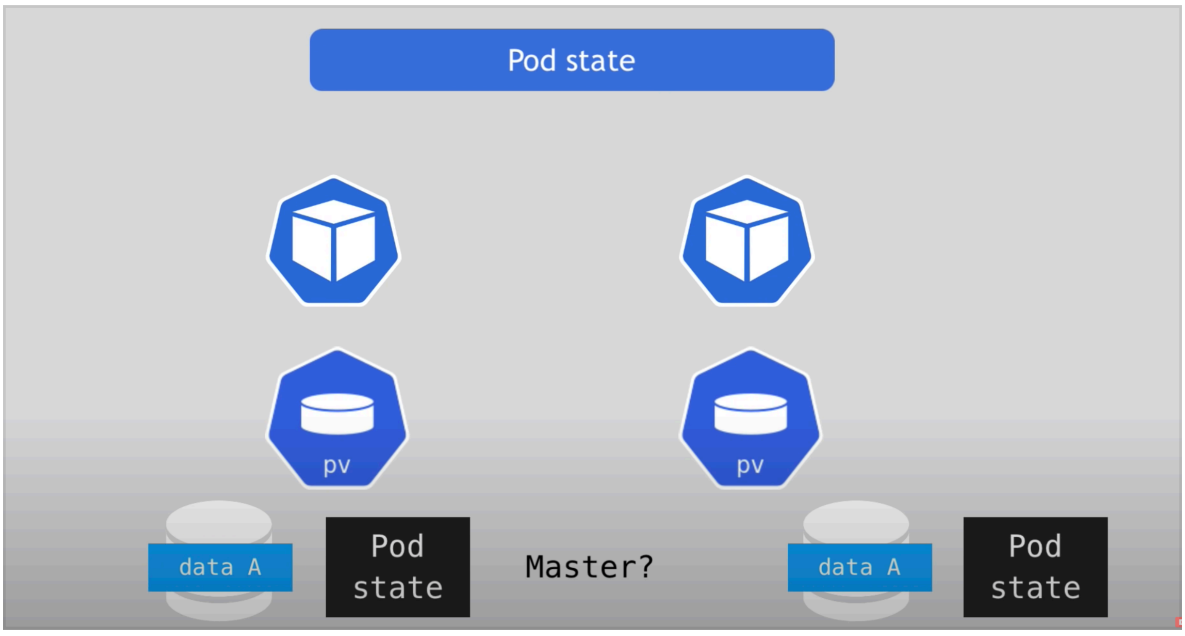


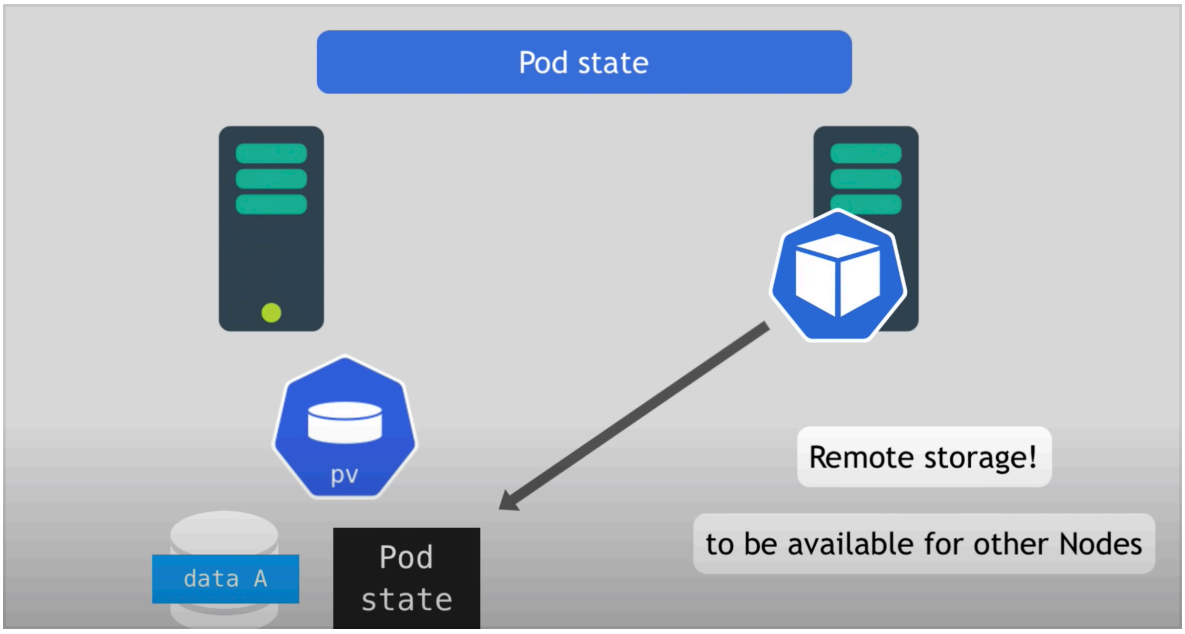
- And the pod that is allowed to update the data is called the master and the others called slaves or worker nodes.
- This is the first diff b/n dev and statefulset, they are not same or identical but there is a master pod and slaves and there is also difference b/n those slaves pods in terms of storage that is the next point
- They don't use or have access to the same physical storage even though they have the same data but they are not using the same

physical storage of the data, they each have thier own replicas of the storage that each one of them can access for itself

- This means that each pod replica at any time must have the same data as the other ones and in order to achieve that they have to continuously synchronise their data.
  - And since master is the only one allowed to change the data and the slaves need to take care of their own data storage obviously slaves must know about each such change so they can update their own data storage to be up to date for the next query requests.
  - And there is a mechanism in such clustered db setup that allows for continuous data sycnhronization,
    - Master changes data and all slaves update this own data storage to keep in sync and make sure each pod has the same state
  - An when new pod joins the existing setup it needs to create its won storage and take care of syncronizing it, what happens is that it first clones the data from the previous pod not just any pod but always from the previous pod and once it has up to date cloned, it starts continuous synczn as well to listen for any updates by master pod
  - And also it is interesting to note that temporary data storage is theoretically possible in stateful appn but when all the pods die the data will be lost, so it is best practice to use data persistence (PV) for stateful applications
  - The way to do that is configuring Persistent Volumes (PV) for your stateful set - https://youtu.be/X48VuDVv0do?t=11346,
    - To re attachment the ID’s to the pod after restoring the POD (And also in case of rescheduling Pods after being destroyed) it is impotent use remote storage

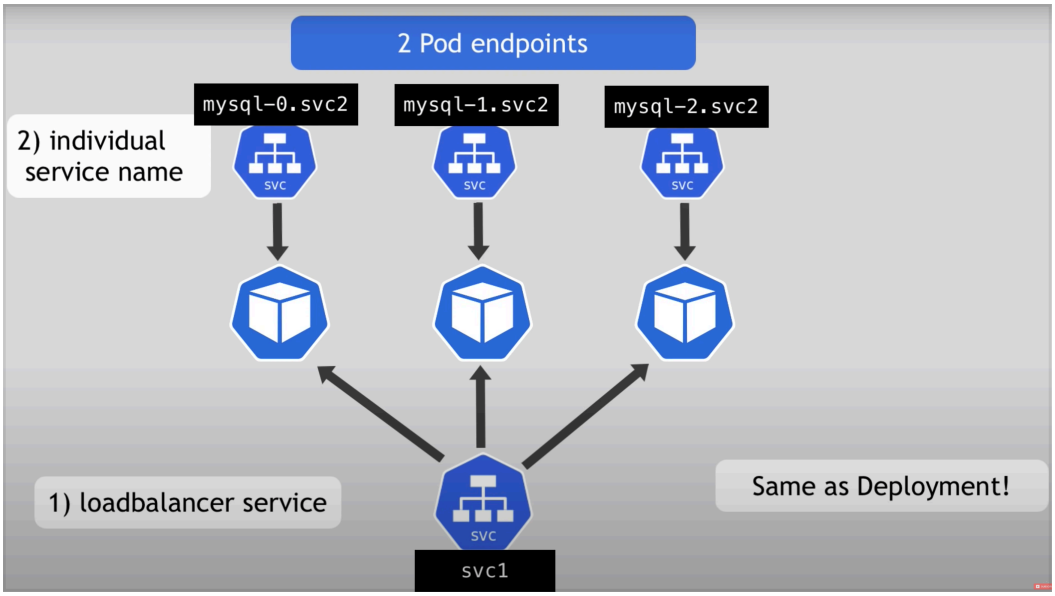

Scaling database applications
MASTER
WORKER
WORKER
mysql-0
mysql-1
mysql-2
They do not use the same physical storage!
PV
PV
PV
data A
data A
data A
/data/vol/pv-0
/data/vol/pv-1
/data/vol/pv-2


Pod state
pv
pv
data A
Pod state
Master?
data A
Pod state



- And the last difference is that this statefulset Pods Identity, it gets fixed ordered names, ${statefulset name}-${ordinal} (mysql-0),
  - Important note to consider is that Next pod is only created, if previous is up and running
    - Deletion starts from the last pod and will not delete synchronously
  - All these are in place. To protect data and state that this stateful apps are dependent on
  -

- Pod endpoints
  - Each pods in a SS gets its own DNS endpoint from a service , same as Deployment plus individual service name for each Pod which deployment pods don’t have


2 Pod endpoints
mysql-0.svc2
mysql-1.svc2
mysql-2.svc2
2) individual service name
svc
svc
svc
1) loadbalancer service
Same as Deployment!
svc
svc1