SDC 18
SNIA INDIA
May 24-25, 2018
Bangalore, India
STORAGE DEVELOPER CONFERENCE

Container Attached Storage (CAS) for stateful applications on containers

Uma Mukkara
@umamukkara @openebs
MayaData Inc.

Agenda

- Who
- Container Attached Storage
  - What ?
  - Why ?
  - How ?
- Reference implementation of CAS
- Example of Blue Green Deployment

Who ?

MAYADATA

- MayaData delivers Data Agility
- Formerly known as CloudByte
- Sponsor of **OpenEBS** and **Litmus** projects
  - Large Bangalore based OSS contributor
- https://mayaonline.io is SaaS software used to manage stateful workloads on Kubernetes

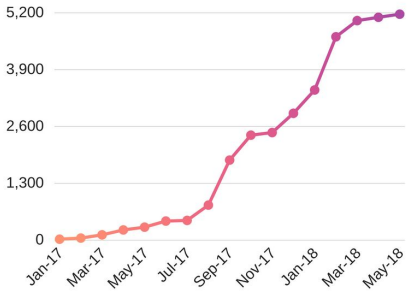
SDC 18
SNIA INDIA

Who ?

OpenEBS

slack.openebs.io

- Popular open source storage project for stateful applications on Kubernetes
- Team of 50+ hackers with storage and DevOps experience



- Leading project in CAS category
- 110+ contributors
- 5K+ github stars
- 1M+ docker pulls

SDC 18 SNIA INDIA

Who ?

- Chaos engineering for stateful applications
- Extends your CI/CD pipelines

Introduction to CAS

- CAS - Container Attached Storage
- It is a new architecture in storage domain
- Applicable only to containers

Genesis of CAS

What if storage for container native applications was itself **container native?**

Storage trends

Microservices / DevOps / Cloud / Kubernetes / incredible acceleration of storage media have changed everything about shared storage.

Shared storage is not optimal now:

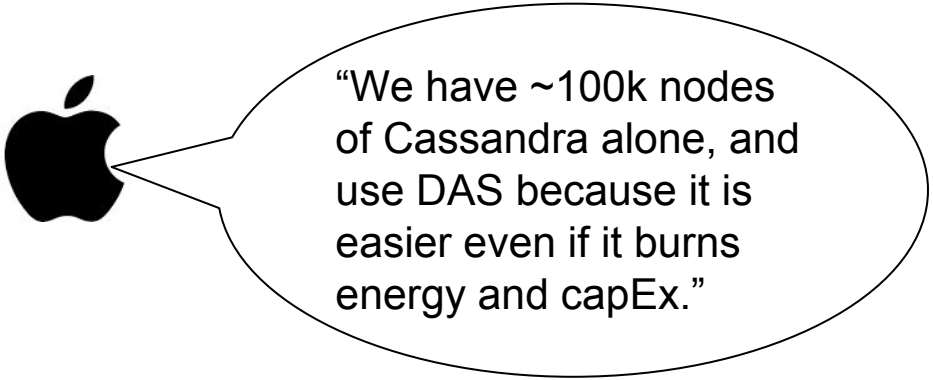
- It used to accelerate your storage
  - **now** it is slower than DAS
- It used to be how you kept your app resilient
  - **now** it increases your blast radius
- It used to be run by one of many IT silos
  - **now** those silos are fading away

DAS vs Distributed

DAS

Benefits:

- Simple
- Ties application to storage
- Predictable for capacity planning
- App deals with resiliency
- Can be faster



Other example DAS users moving to Kubernetes:

NETFLIX Booking.com tenable Walmart

DAS vs Distributed

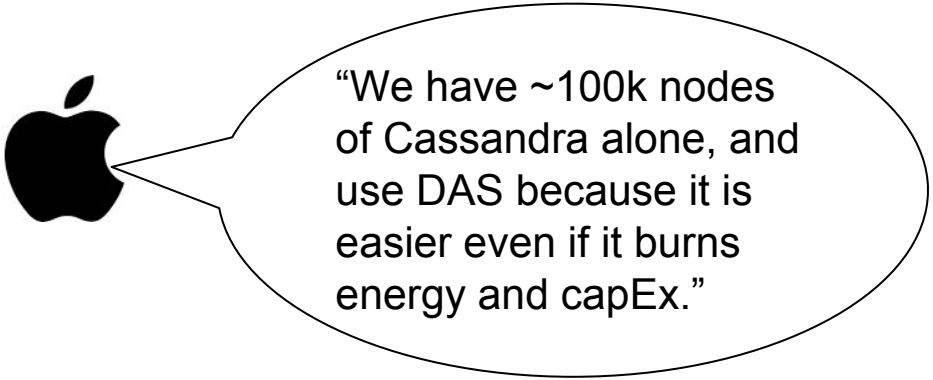
DAS

Benefits:

- Simple
- Ties application to storage
- Predictable for capacity planning
- App deals with resiliency
- Can be faster

Concerns:

- Under-utilized hardware
  - *10% or less utilization*
- Wastes data center
- Difficult to manage
- Lacks storage features
- Cannot be repurposed - made for one workload
- Does not support mobility of workloads via containers
- Cross cloud impossible



Other example DAS users moving to Kubernetes:

NETFLIX Booking.com tenable Walmart

DAS vs Distributed

DAS

Benefits:

- Simple
- Ties application to storage
- Predictable for capacity planning
- App deals with resiliency
- Can be faster

Concerns:

- Under-utilized hardware
  - *10% or less utilization*
- Wastes data center
- Difficult to manage
- Lacks storage features
- Cannot be repurposed - made for one workload
- Does not support mobility of workloads via containers
- Cross cloud impossible

Distributed

Benefits:

- Centralized management
- Greater density and efficiency
- Storage features such as:
  - *Data protection*
  - *Snapshots for versioning*

Concerns:

- Additional complexity
- Enormous blast radius
- Expensive
- Requires storage engineering
- Challenged by container dynamism
- No per microservice storage policy
- I/O blender impairs performance
- Locks customers into vendor
- Cross cloud impossible

DAS vs Distributed

DAS

Benefits:

- Simple
- Ties application to storage
- Predictable for capacity planning
- App deals with resiliency
- Can be faster

Concerns:

- Under-utilized hardware
  - *10% or less utilization*
- Wastes data center
- Difficult to manage
- Lacks storage features
- Cannot be repurposed - made for one workload
- Does not support mobility of workloads via containers
- Cross cloud impossible

Distributed

Benefits:

- Centralized management
- Greater density and efficiency
- Storage features such as:
  - *Data protection*
  - *Snapshots for versioning*

Concerns:

- Additional complexity
- Enormous blast radius
- Expensive
- Requires storage engineering
- Challenged by container dynamism
- No per microservice storage policy
- I/O blender impairs performance
- Locks customers into vendor
- Cross cloud impossible

SDC 18
SNIA INDIA

YASSS

Not tunable for containers

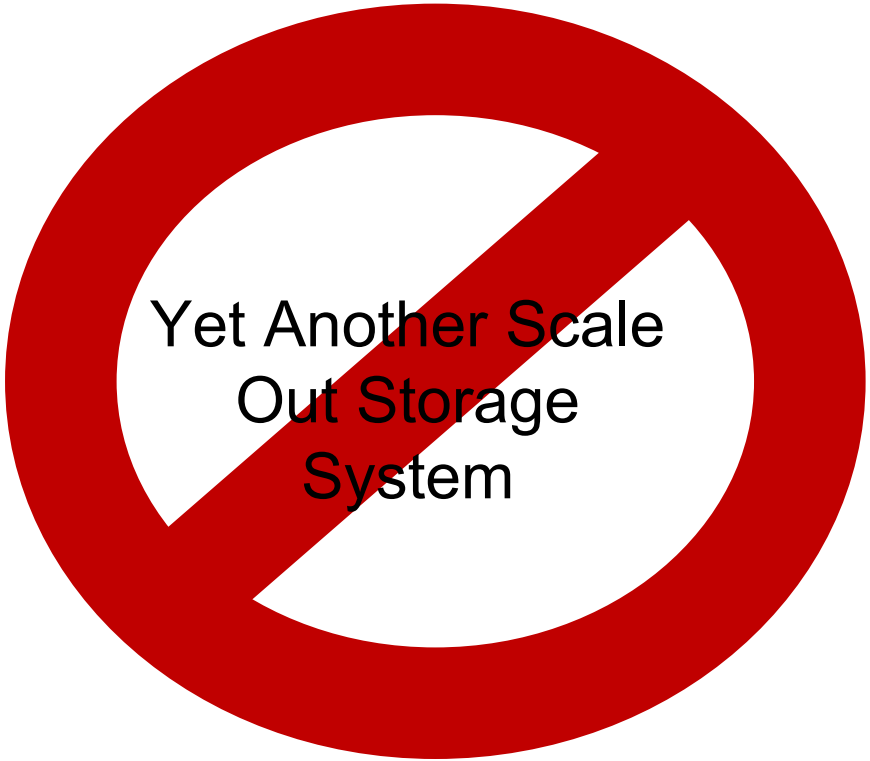
Lock-in & not x-cloud

Latency & Big 0

Monolithic mess

Huge blast radius

!YASSS

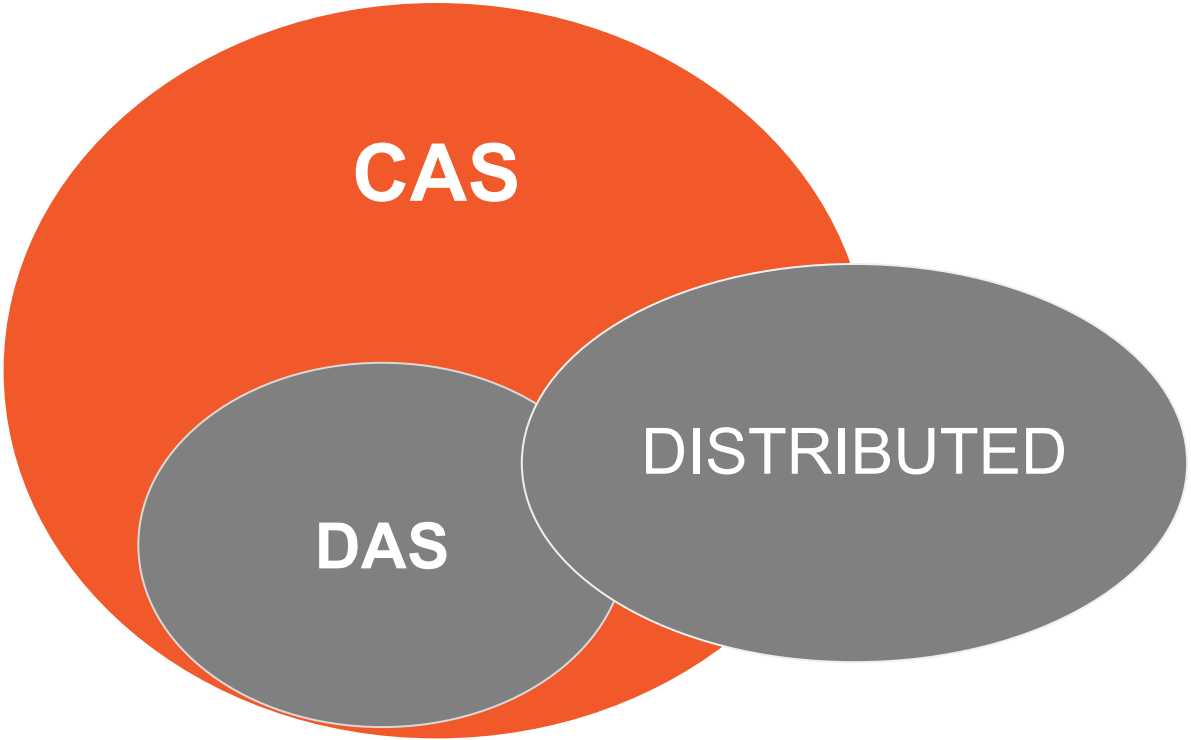


What is needed ?

- Like DAS
- No blast radius
- Container friendly

CAS

CAS = Containers & DAS^{+} & Distributed^{-}



SDC 18
SNIA INDIA

CAS

Tunable per container

Cross SAN portability

Cross cloud portability

Low latency

No special skills needed!

CAS feature comparison

DAS

Benefits:

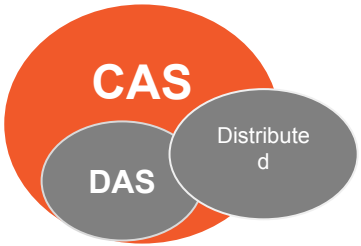
- Simple
- Ties application to storage
- Predictable for capacity planning
- App deals with resiliency
- Can be faster

Concerns:

- Under-utilized hardware
  - *10% or less utilization*
- Wastes data center
- Difficult to manage
- Lacks storage features
- Cannot be repurposed - made for one workload
- Does not support mobility of workloads via containers
- Cross cloud impossible

OpenEBS = "CAS"

- ✓ Simple
- ✓ No new skills required
- ✓ Per microservice storage policy
- ✓ Data protection & snapshots
- ✓ Reduces cloud vendor lock-in
- ✓ Eliminates storage vendor lock-in
- ✓ Highest possible efficiency
- ✓ Large & growing OSS community
- ✓ Natively cross cloud



Distributed

Benefits:

- Centralized management
- Greater density and efficiency
- Storage features such as:
  - *Data protection*
  - *Snapshots for versioning*

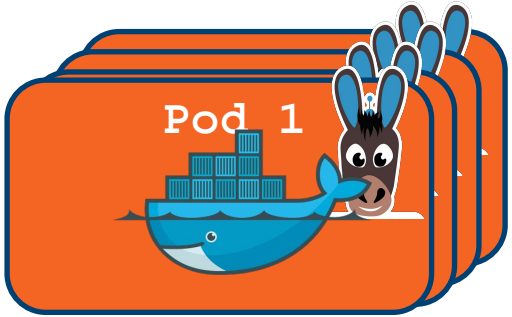
Concerns:

- Additional complexity
- Enormous blast radius
- Expensive
- Requires storage engineering
- Challenged by container dynamism
- No per microservice storage policy
- I/O blender impairs performance
- Locks customers into vendor
- Cross cloud impossible

SDC 18
SNIA INDIA

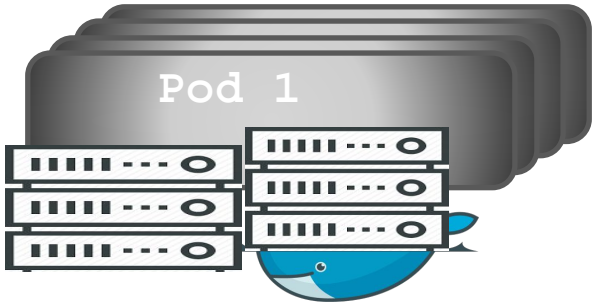
Why CAS ?

- Truly Cloud Native (Storage is a microservice)
- Enables cloud native blue-green deployment of storage (no need to upgrade all volumes at once)
- Highest granularity of storage policies (assume one storage controller handles only one type of workload)



cloud native

VS



CAS benefits

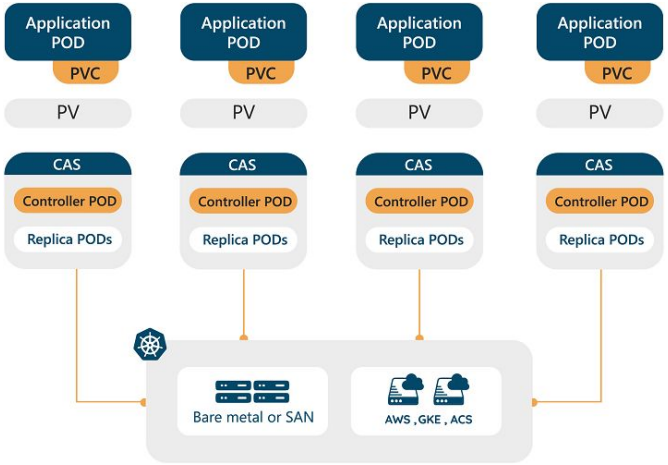
Cloud Native challenges

1. Workloads much smaller
2. Ephemeral duration
3. 10-100x increase in quantity
4. Mobile workloads
5. Immutable not tunable
6. Developers responsible for operations
7. Easiest solutions lead to cloud lock-in

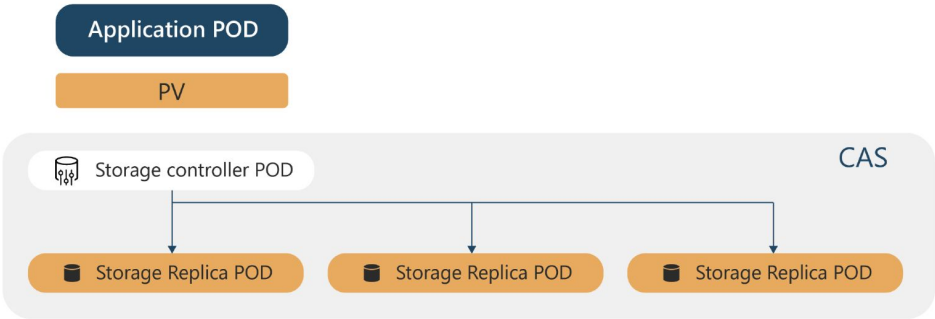
CAS Answers

1. Keep data local
2. Run controller in the POD
3. 10-100x more controllers
4. Follow the workloads
5. Stateless control of state
6. Make storage just another microservice each team controls. No YASSS needed!
7. Cross cloud cMotion - thanks to OpenEBS & MayaOnline

CAS architecture example

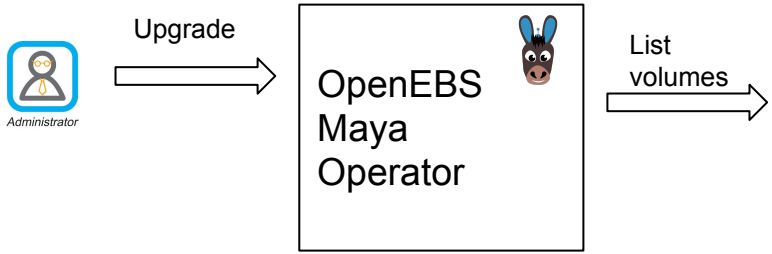


- Each volume has a controller and three replica pods



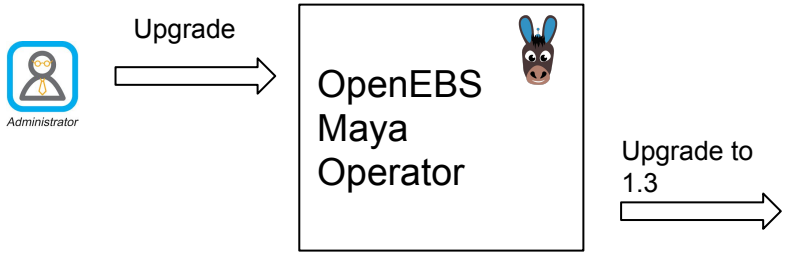
SDC 18
SNIA INDIA

Blue Green Deployment example with CAS



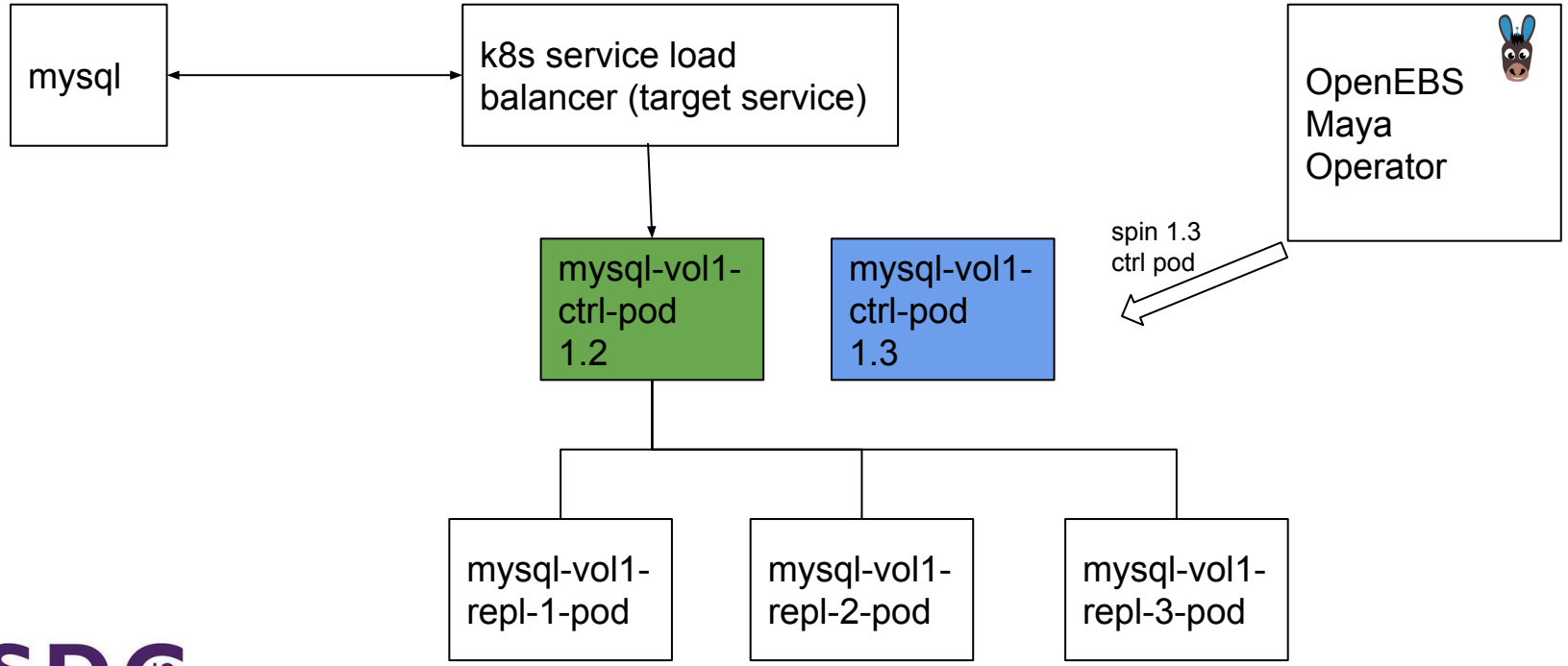
#	volume name	ctrl ver	rep1 ver	rep2 ver	rep3 ver
1	mysql-vol1	1.2	1.2	1.2	1.2
2	postgre-vol1	1.1	1.1	1.1	1.1
3	postgre-vol2	1.2	1.2	1.2	1.2
4	mongoDB-vol1	1.2	1.2	1.2	1.2

Blue Green Deployment example with CAS

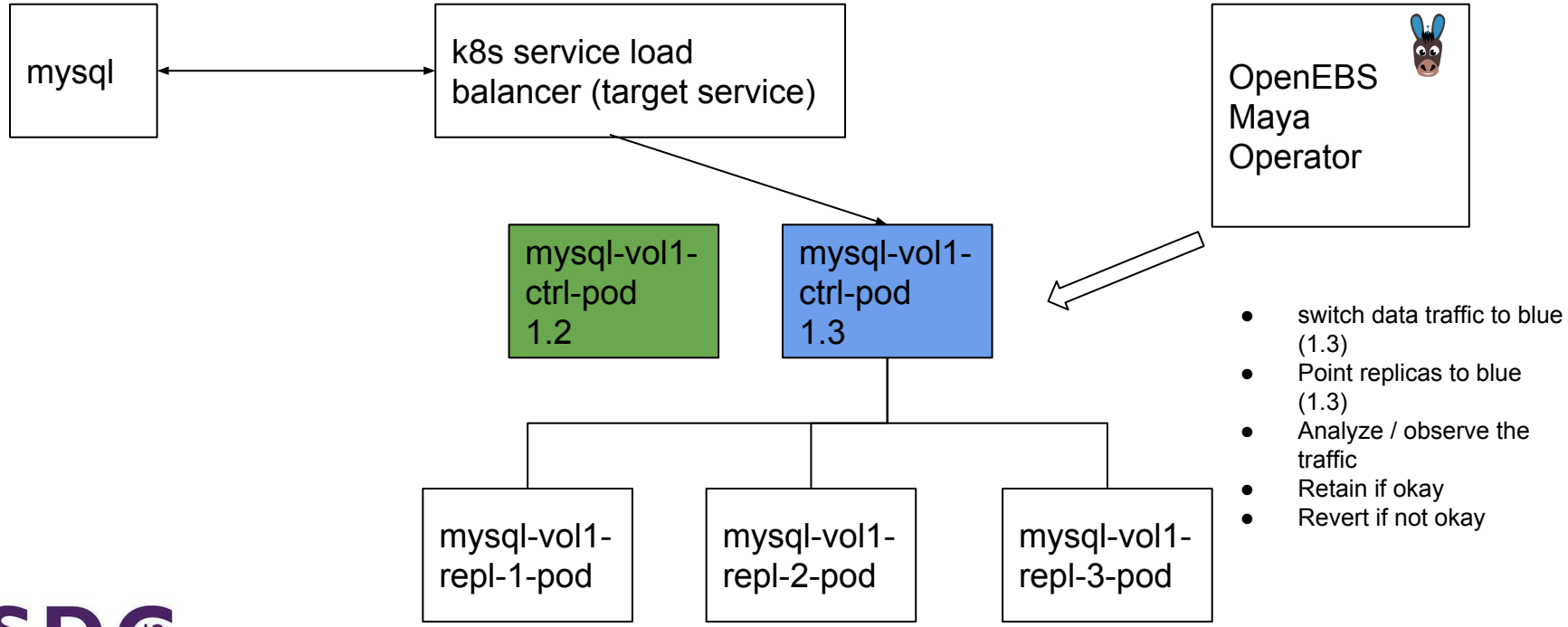


#	volume name	ctrl ver	rep1 ver	rep2 ver	rep3 ver
1	mysql-vol1	1.2	1.2	1.2	1.2
2	postgre-vol1	1.1	1.1	1.1	1.1
3	postgre-vol2	1.2	1.2	1.2	1.2
4	mongoDB-vol1	1.2	1.2	1.2	1.2

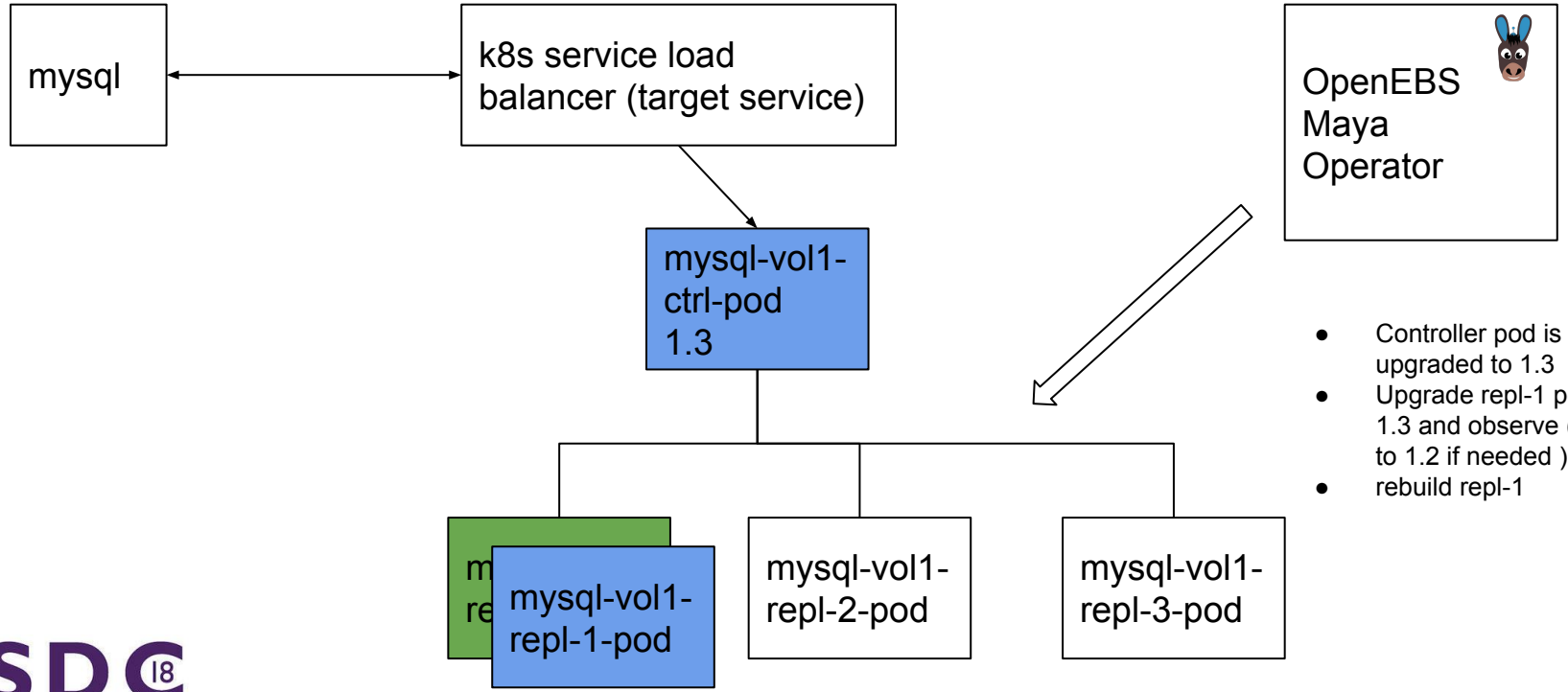
Blue Green deployment of mysql-vol1



Blue Green deployment of mysql-vol1



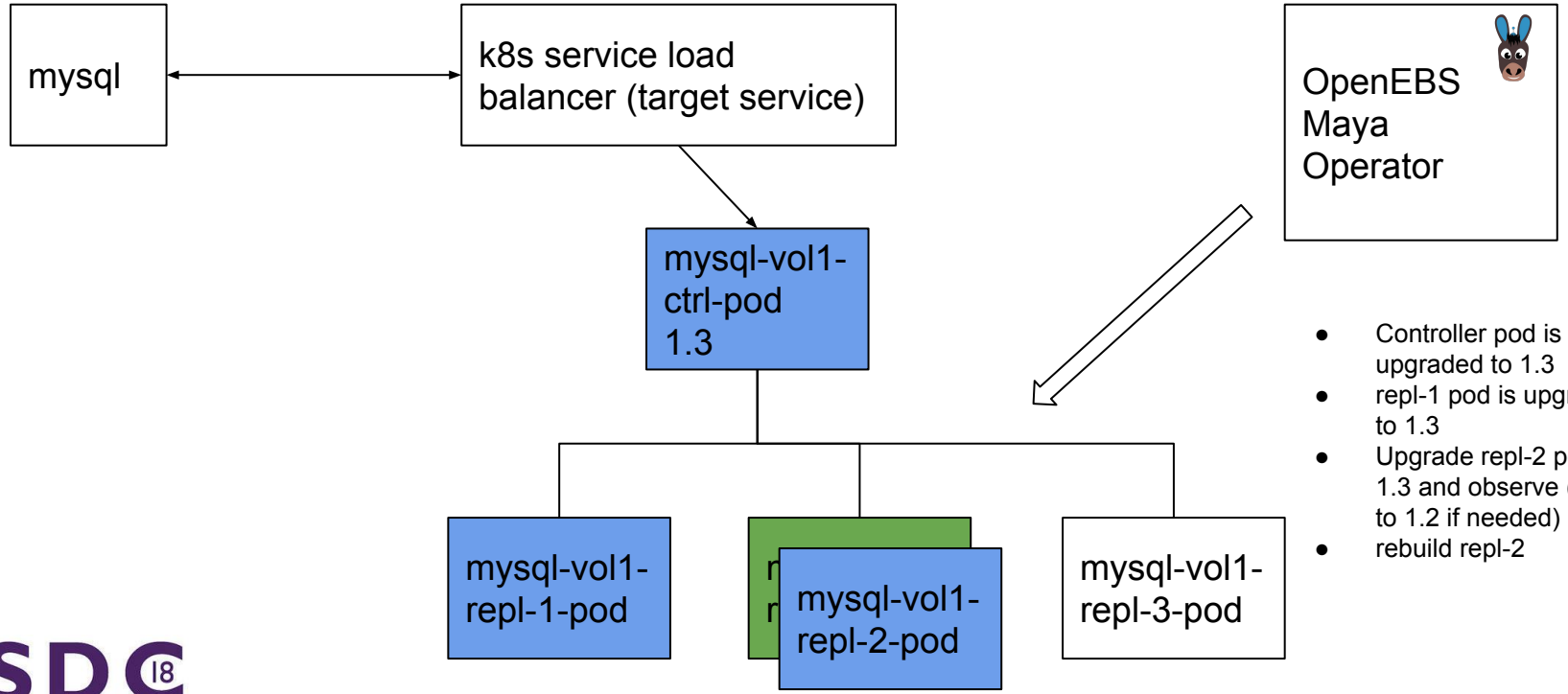
Blue Green deployment of mysql-vol1



- Controller pod is upgraded to 1.3
- Upgrade repl-1 pod to 1.3 and observe (revert to 1.2 if needed)
- rebuild repl-1

SDC 18
SNIA INDIA

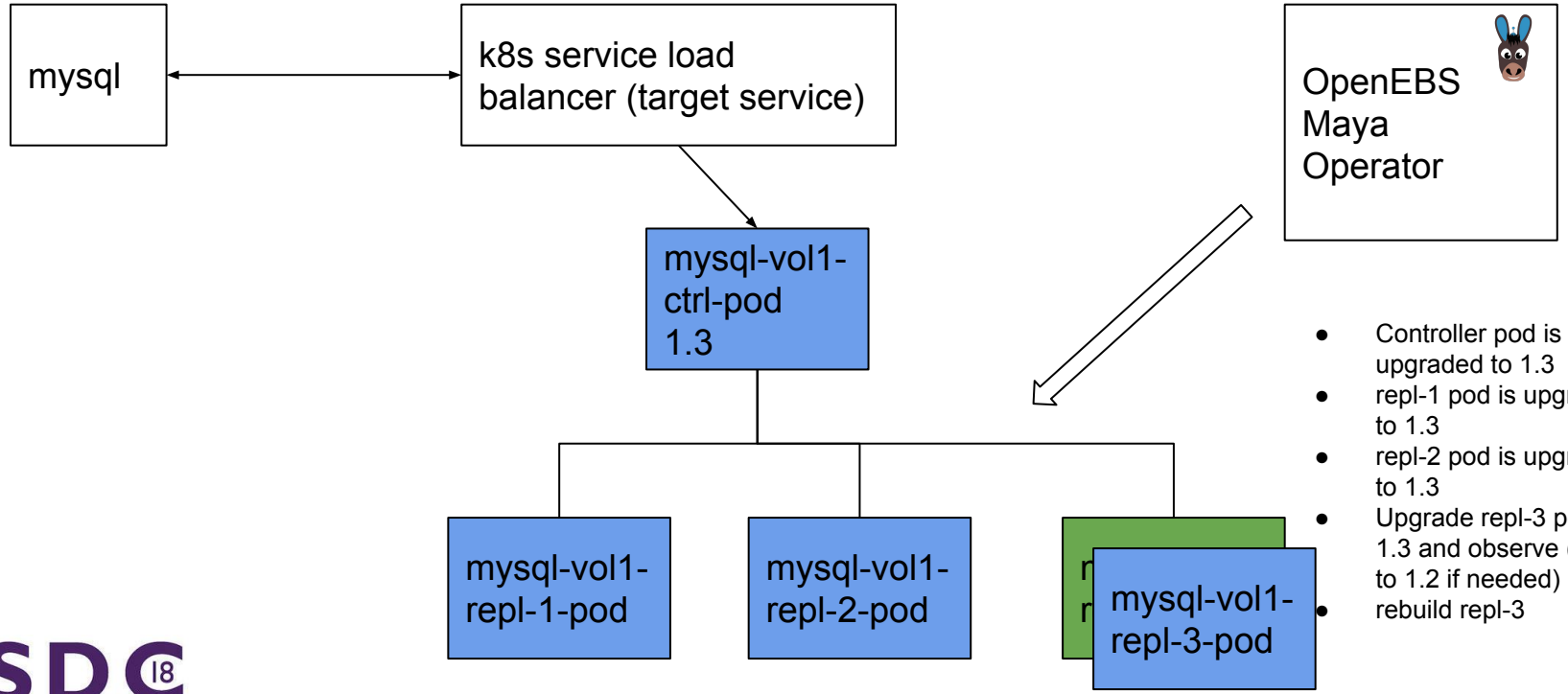
Blue Green deployment of mysql-vol1



- Controller pod is upgraded to 1.3
- repl-1 pod is upgraded to 1.3
- Upgrade repl-2 pod to 1.3 and observe (revert to 1.2 if needed)
- rebuild repl-2

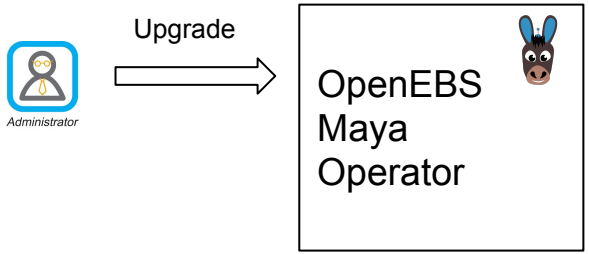
SDC 18 SNIA INDIA

Blue Green deployment of mysql-vol1



- Controller pod is upgraded to 1.3
- repl-1 pod is upgraded to 1.3
- repl-2 pod is upgraded to 1.3
- Upgrade repl-3 pod to 1.3 and observe (revert to 1.2 if needed)
- rebuild repl-3

Blue Green Deployment example with CAS



mysql-vol1 is upgraded to 1.3
Now upgrade postgre-vol1 to 1.3

#	volume name	ctrl ver	rep1 ver	rep2 ver	rep3 ver
1	mysql-vol1	1.3	1.3	1.3	1.3
2	postgre-vol1	1.1	1.1	1.1	1.1
3	postgre-vol2	1.2	1.2	1.2	1.2
4	mongoDB-vol1	1.2	1.2	1.2	1.2

Upgrade operations - B/G

- With CAS architecture, the upgrade process can be automated without human intervention. Something that is unheard of in the storage world.

What about speeds and feeds?

- The bottleneck is often now the Linux kernel
  - User space I/O and networking is the future
    - You already know about SPDK, FD.io, VPP
    - You might find interesting:
      - https://github.com/openebs/vhost-user
    - We are looking for collaboration in this area as well

CAS - References

Blog published by CNCF on CAS

https://www.cncf.io/blog/2018/04/19/container-attached-storage-a-primer/

Storage just **fades** away **as** a **concern**

Q&A

Thank you !!

slack.openebs.io