facebook

Massive Schema Changes in Facebook

Junyi Lu / 卢钧轶
Production Engineer

Agenda

1. How OnlineSchemaChange works
2. How OSC Scheduler works

Online Schema Change

https://github.com/facebookincubator/OnlineSchemaChange

Online Schema Change

Why

Why not MySQL's native online DDL?

- no resource control
- causing replication lag on slaves
- cannot cover all our use cases.
  - ie: ALTER TABLE WHERE COL1 = 2

Online Schema Change

Existing approach

- pt-online-schema-change 2011
- Large Hadron Migrator(LHM) 2011
- Github's Online Schema Change (gh-ost)? 2016

Online Schema Change

OSC.php (2009-2016)

Core logic is written in PHP

First open sourced in Sep 2010

A wrapper is required

Hard to test

https://www.facebook.com/notes/mysql-at-facebook/online-schema-change-for-mysql/430801045932/

Online Schema Change

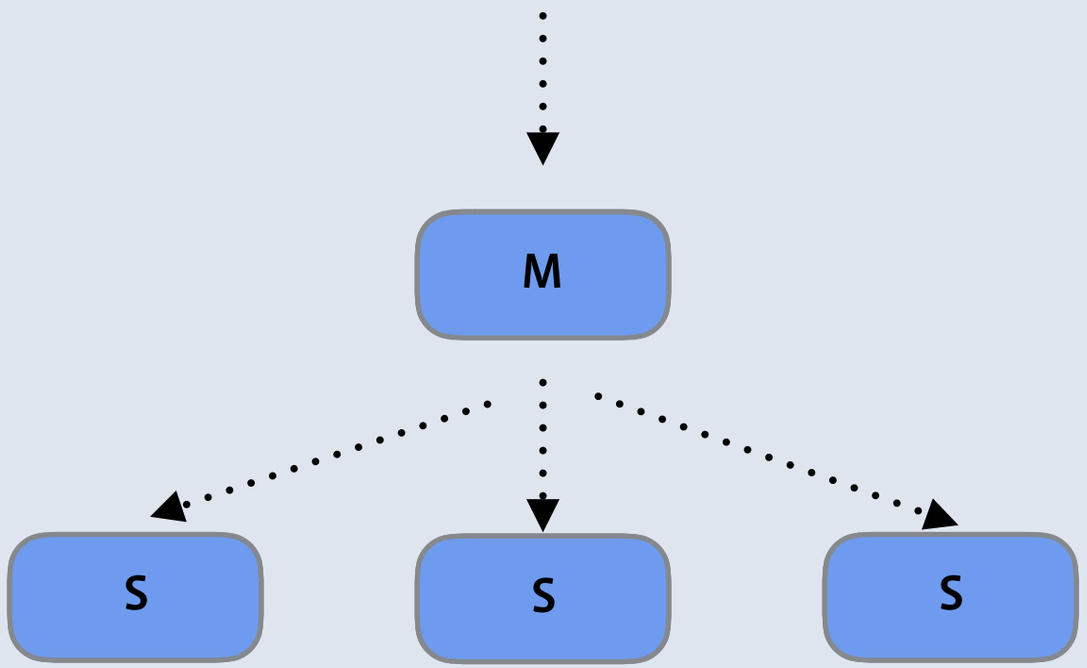
OSC.py (2016 - ?)

- Well tested (unittest/integration test) & bug fix
- Faster catchup
- Checksum
- Hook system
- Unicode CREATE TABLE `(ノ≧∇≦)ノミ┻━━┻` ...

Online Schema Change

What is OSC?

- Out of replication
- Async dump & load
- Trigger based



Online Schema Change

How to use

Cats		
ID	int	PRIMARY
Name	varchar(10)	

Cats		
ID	int	PRIMARY
Name	varchar(10)	
PictureURL	**varchar(30)**	

Online Schema Change

How to use

```
ALTER TABLE Cats ADD COLUMN
PictureURL VARCHAR(30)
```

```
CREATE TABLE Cats (
    ID  INT NOT NULL,
    Name VARCHAR(10),
    PictureURL VARCHAR(30),
    PRIMARY (ID)
) ENGINE=INNODB
```

>> osc --database cats_db --ddl-file-list /path/to/a.sql

OnlineSchemaChange

OnlineSchemaChange

Sanity check:

- Primary key/Unique key (required on new schema)
- Foreign key (not supported)
- Table existence
- Accidentally drop column
- Primary key coverage on new schema

OnlineSchemaChange

Cats		
CatID	int	PRIMARY
Name	varchar(10)	

__osc_new_Cats		
CatID	int	PRIMARY
Name	varchar(10)	
PictureURL	varchar(20)	

__osc_chg_Cats		
ID	int	PRIMARY
DML_TYPE	ENUM	
CatID	int	
Name	varchar(10)	

OnlineSchemaChange

OnlineSchemaChange

INSERT INTO Cats VALUES (1, 'foo')

ID	DML_TYPE	CatID	Name
1	DML_TYPE_INSERT	1	foo

OnlineSchemaChange

DELETE FROM Cats

WHERE CatID = 1

ID	DML_TYPE	CatID	Name
2	DML_TYPE_DELETE	1	

OnlineSchemaChange

UPDATE Cats set Name = 'bar'

WHERE CatID = 2

ID	DML_TYPE	CatID	Name
3	DML_TYPE_UPDATE	2	bar

OnlineSchemaChange

UPDATE Cats set CatID = 4, Name = 'foobar'

WHERE CatID = 3

ID	DML_TYPE	CatID	Name
4	DML_TYPE_DELETE	3	
5	DML_TYPE_INSERT	4	foobar

OnlineSchemaChange

INSERT INTO Cats VALUES (1, 'foo')

DELETE FROM Cats WHERE CatID = 1

UPDATE Cats set Name = 'new_foo' WHERE CatID = 3

UPDATE Cats set CatID = 4, Name = 'foobar'
WHERE CatID = 3

ID	CHANGE_TYPE	CatID	Name
1	DML_TYPE_INSERT	1	foo
2	DML_TYPE_DELETE	1	
3	DML_TYPE_UPDATE	3	bar
4	DML_TYPE_DELETE	3	
5	DML_TYPE_INSERT	4	foobar

OnlineSchemaChange

_osc_chg_Cats

start: chg_id = 5

START TRANSACTION

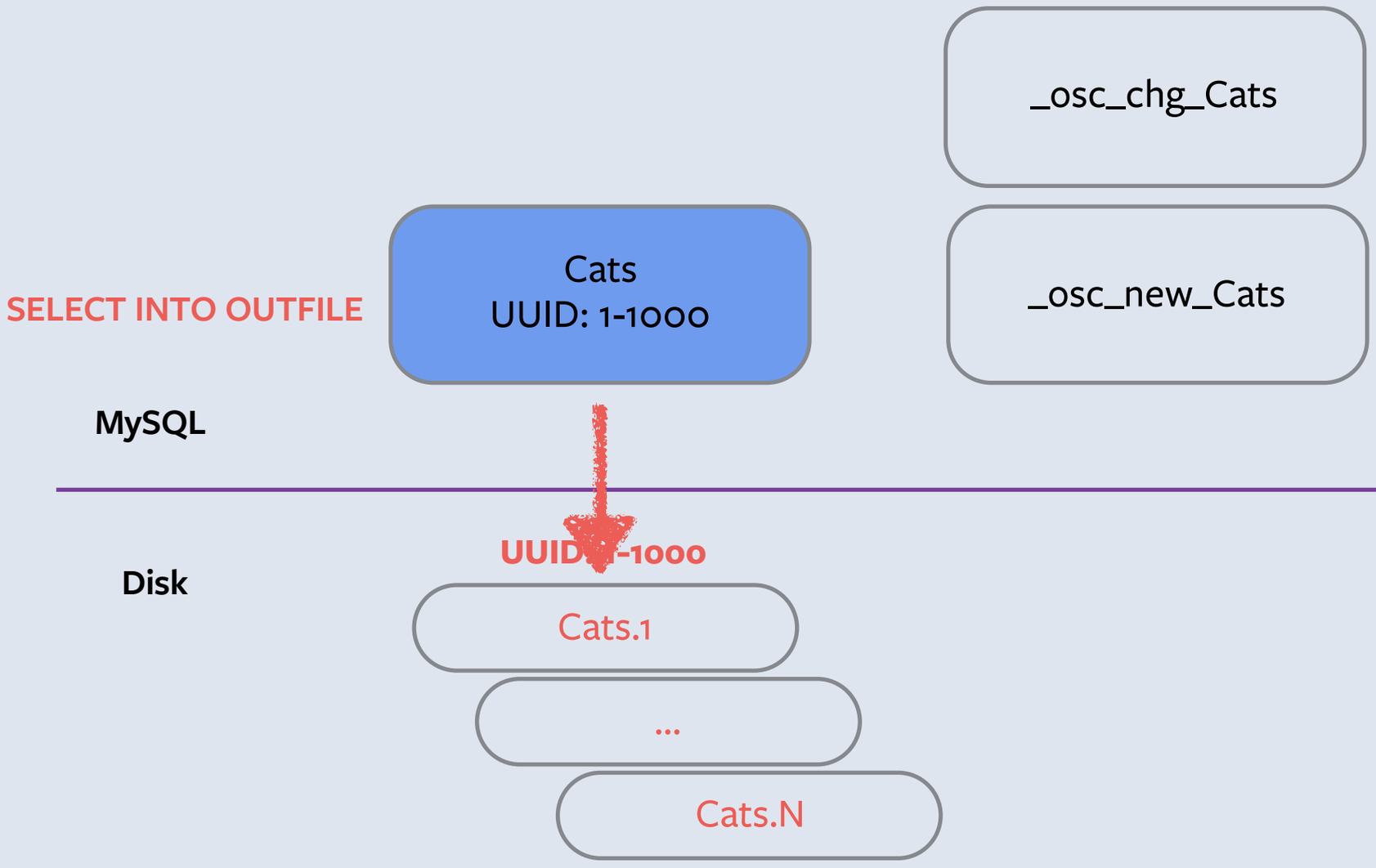
Cats
UUID: 1-1000

_osc_new_Cats

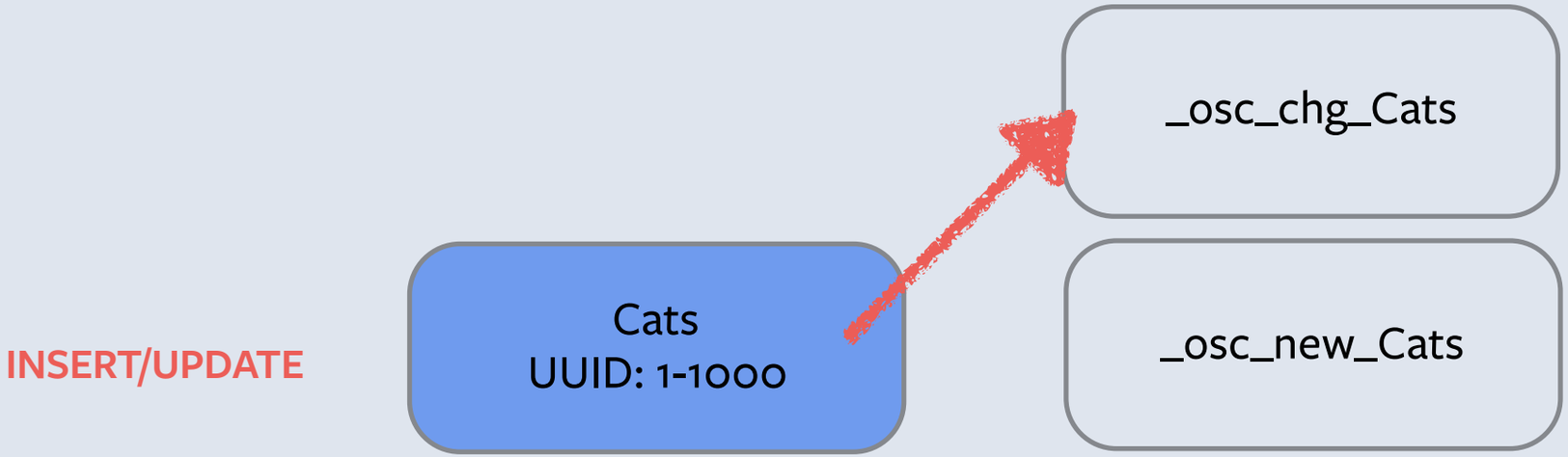
MySQL

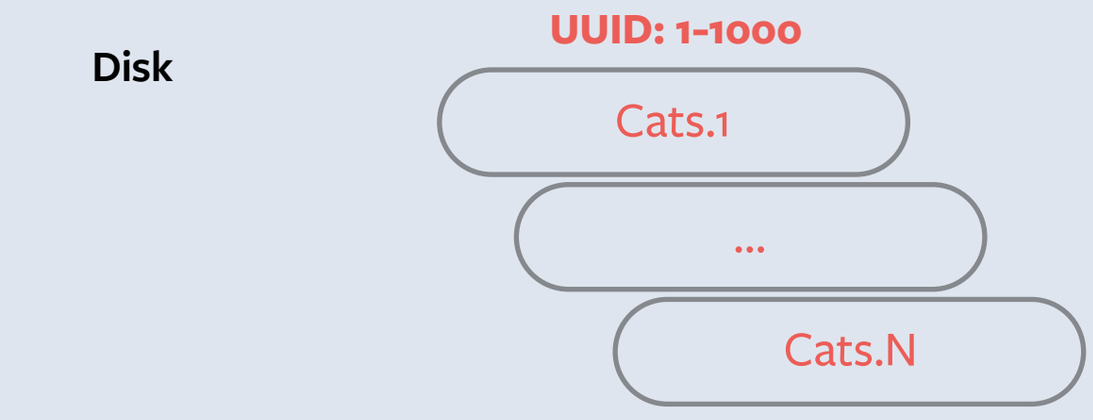
Disk

OnlineSchemaChange

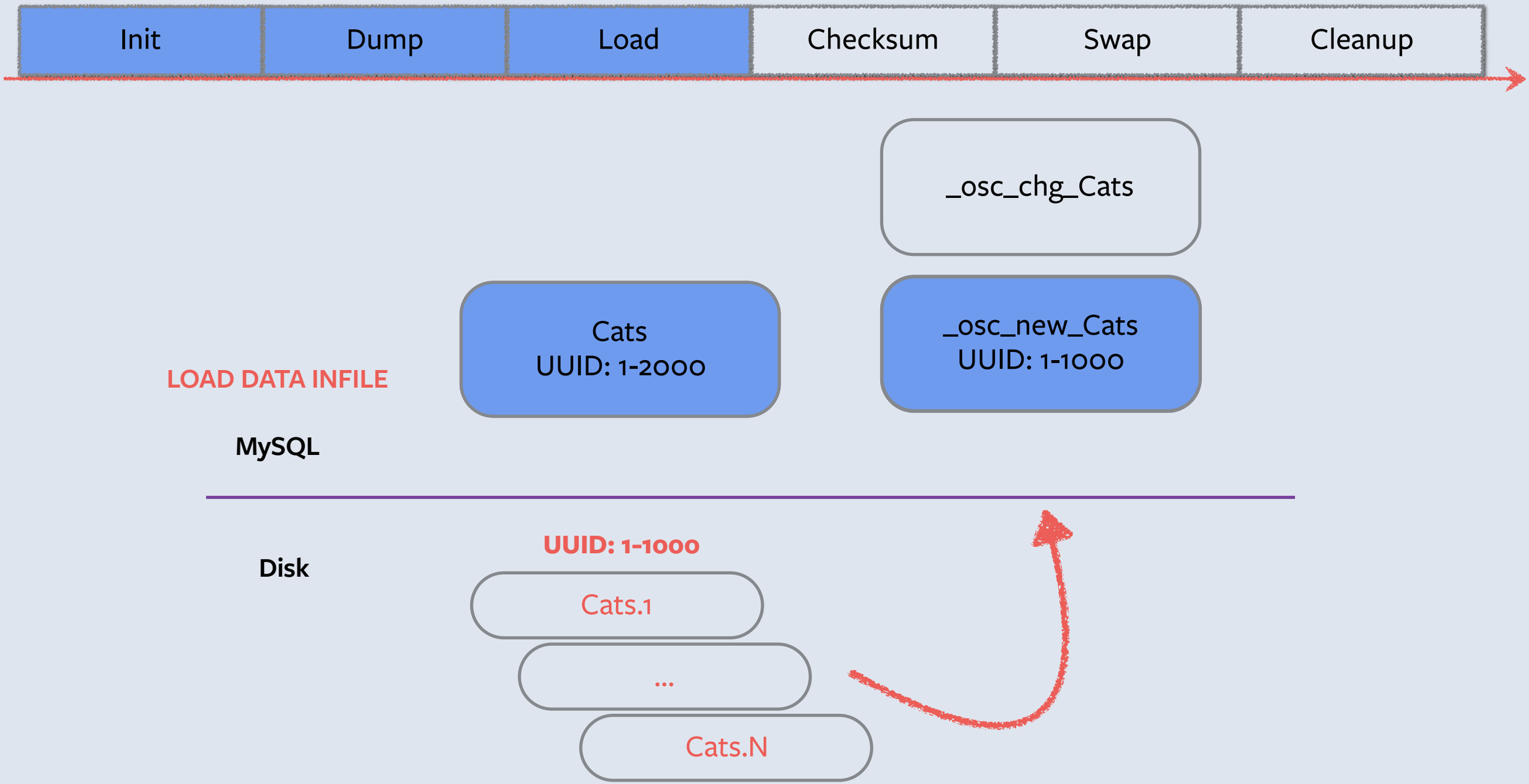


OnlineSchemaChange



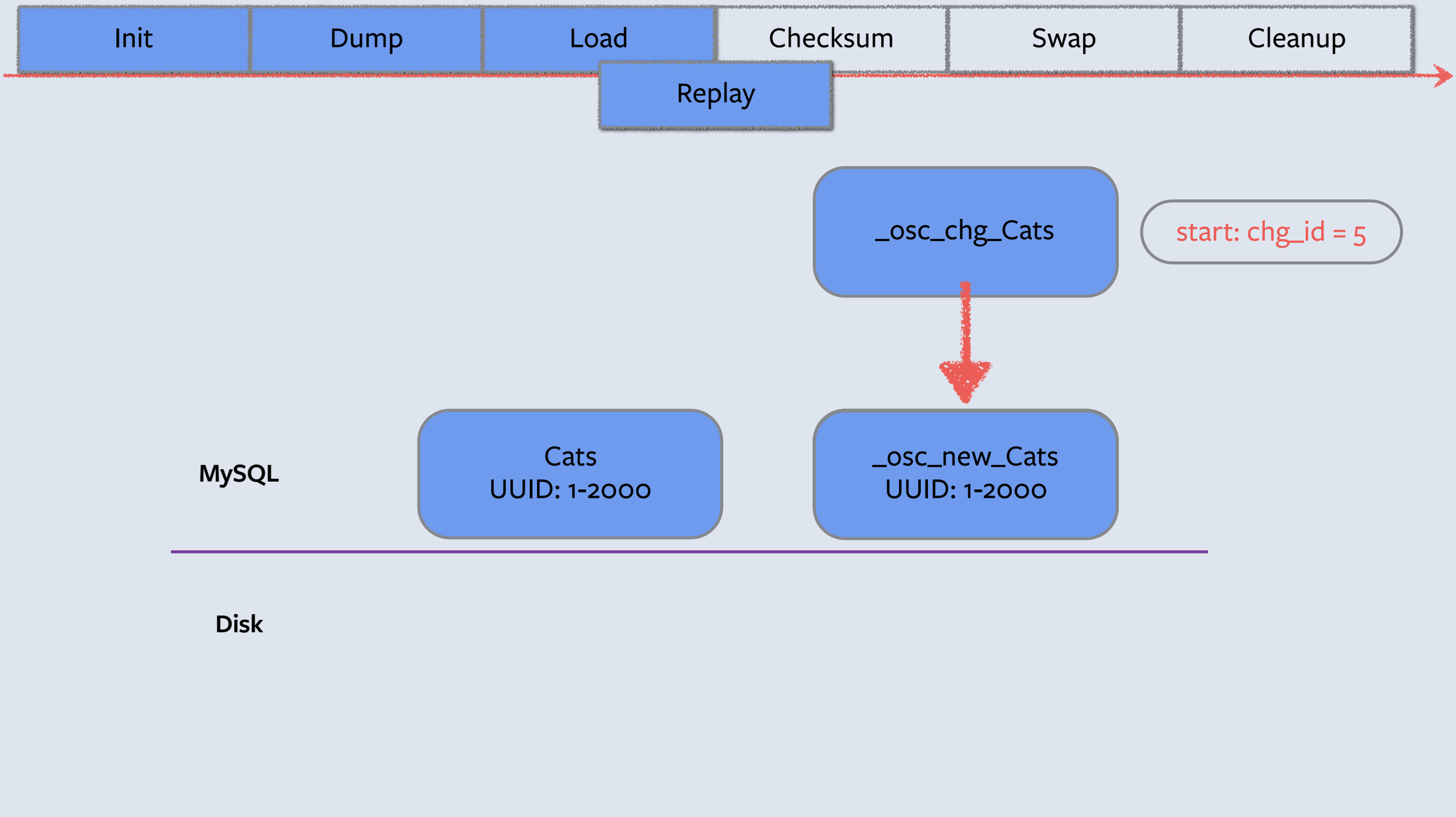


OnlineSchemaChange

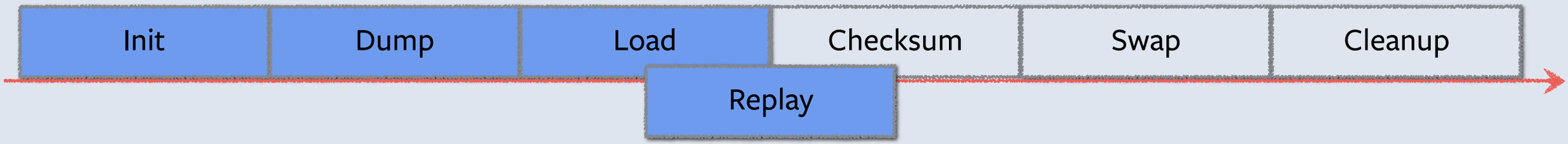


Replay

OnlineSchemaChange



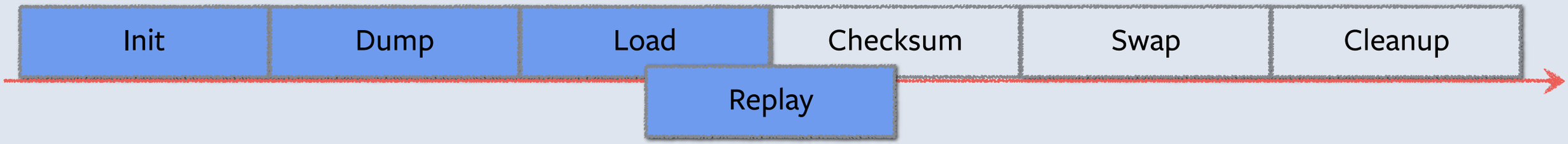
OnlineSchemaChange



ID	DML_TYPE	CatID(PK)	Name
1	DML_TYPE_INSERT	1	foo

```
INSERT INTO __osc_new_Cats (CatID, Name)
SELECT CatID, Name
FROM __osc_chg_Cats WHERE ID = 1;
```

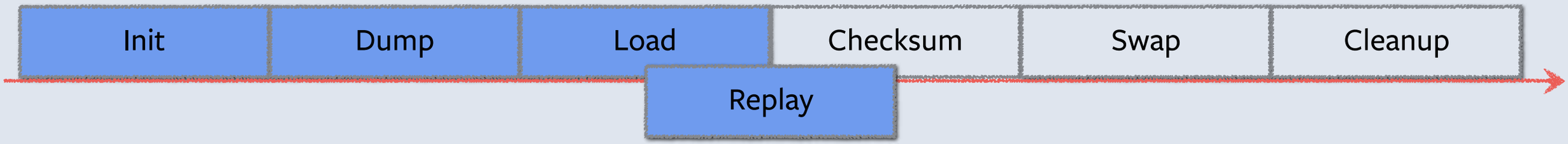
OnlineSchemaChange



ID	DML_TYPE	CatID(PRIMARY)	Name
2	DML_TYPE_DELETE	2	

```
DELETE __osc_new_Cats
FROM __osc_chg_Cats, __osc_new_Cats
WHERE
__osc_chg_Cats.CatID = __osc_new_Cats.CatID
AND ID = 2
```

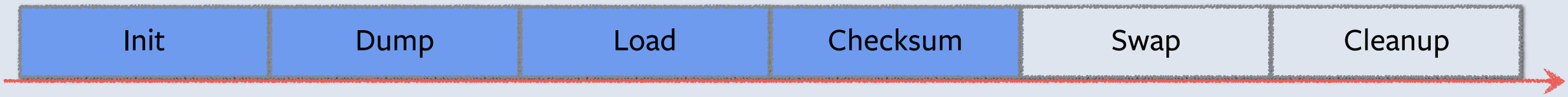
OnlineSchemaChange



ID	DML_TYPE	CatID(PRIMARY)	Name
3	DML_TYPE_UPDATE	3	bar

```
UPDATE __osc_new_Cats, __osc_chg_Cats
SET  __osc_new_Cats.Name = __osc_chg_Cats.Name
WHERE __osc_new_Cats.CatID = __osc_chg_Cats.CatID
AND ID=3
```

OnlineSchemaChange



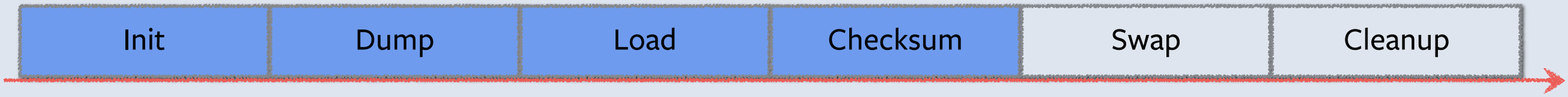
START TRANSACTION

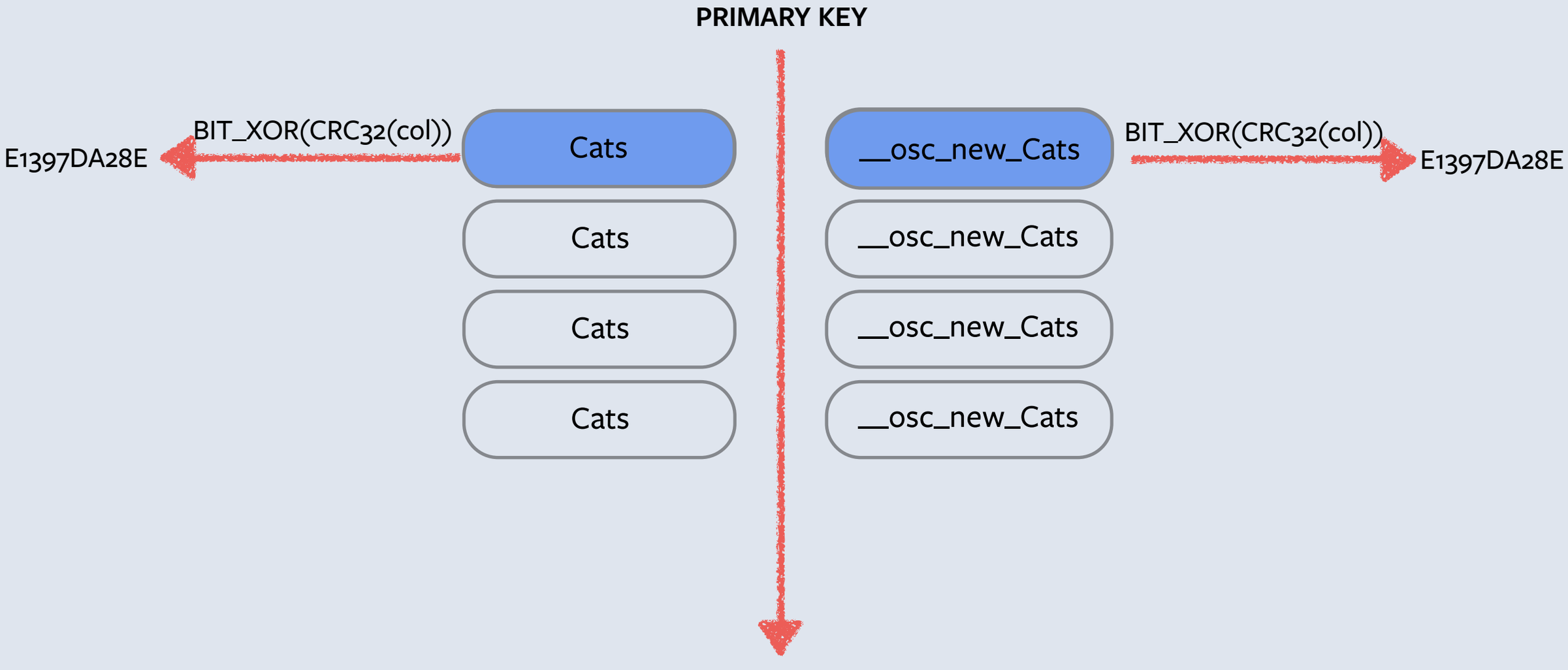
Cats
UUID: 1-2500

__osc_chg_Cats

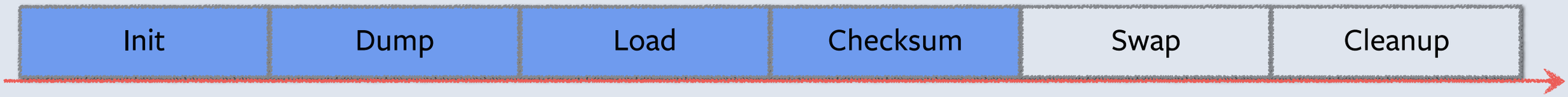
_osc_new_Cats
UUID: 1-2500

OnlineSchemaChange





OnlineSchemaChange

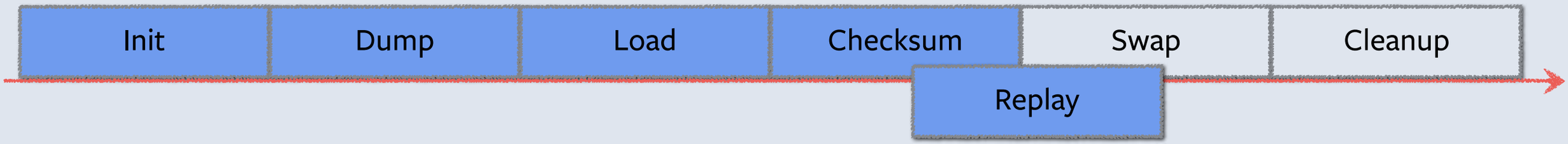


Q: Is checksum useful?

A: Absolutely!!! Any code written by human can have bugs:

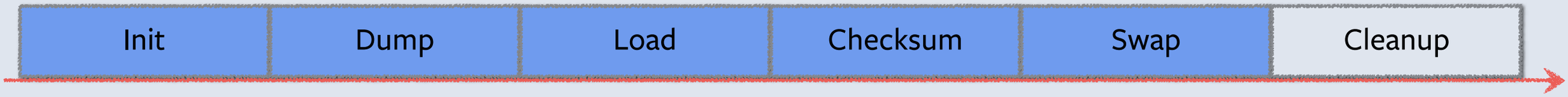
- ID recorded by triggers are not always monotonic with commit order
- transaction visibility when using unique_checks=0 on MyRocks

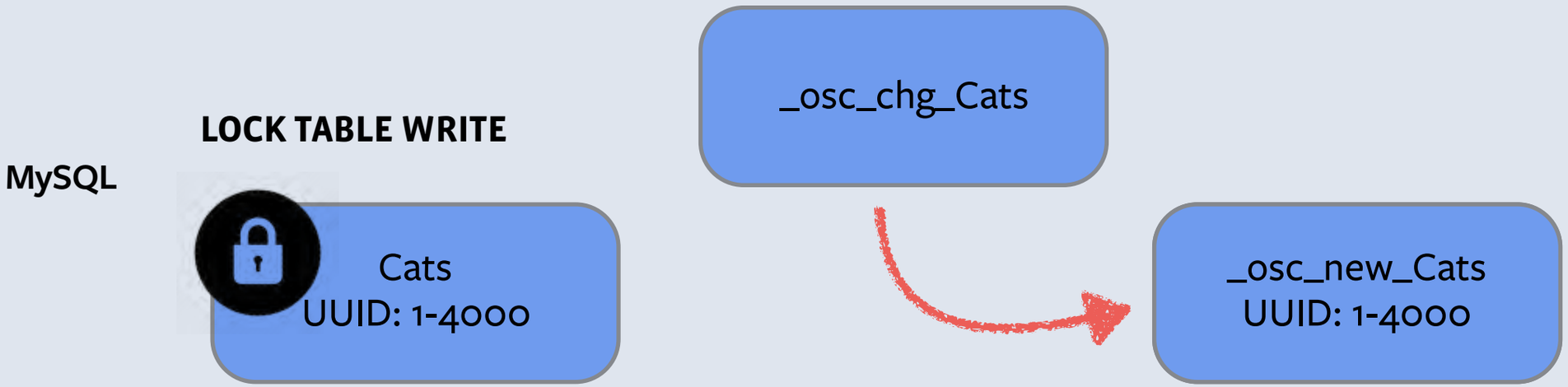
OnlineSchemaChange



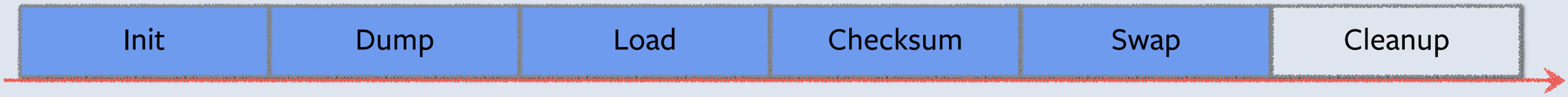
Stage	Changes Stacked	Time Spent in Replay
Dump	10000 rows	
Load	20000 rows	
Replay	5000 rows	500 Seconds
Checksum	10000 rows	
Replay Round 1	2000 rows	200 Seconds
Replay Round 2	1000 rows	50 Seconds
Replay Round 3	100 rows	5 Seconds
Replay Round 4	1 rows	0.1 Seconds

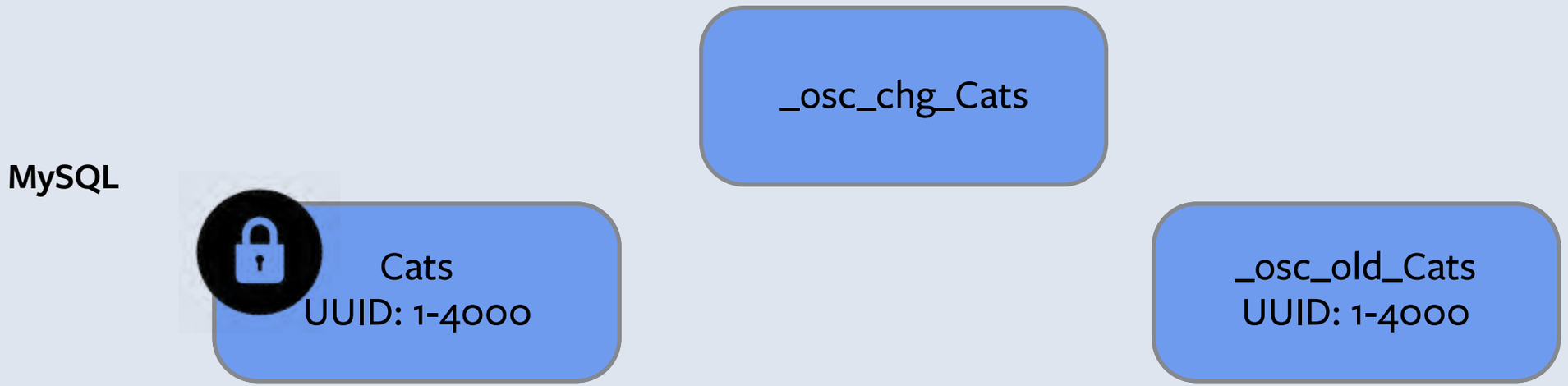
OnlineSchemaChange



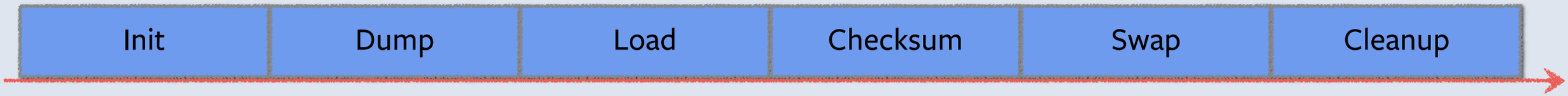


OnlineSchemaChange





OnlineSchemaChange



MySQL

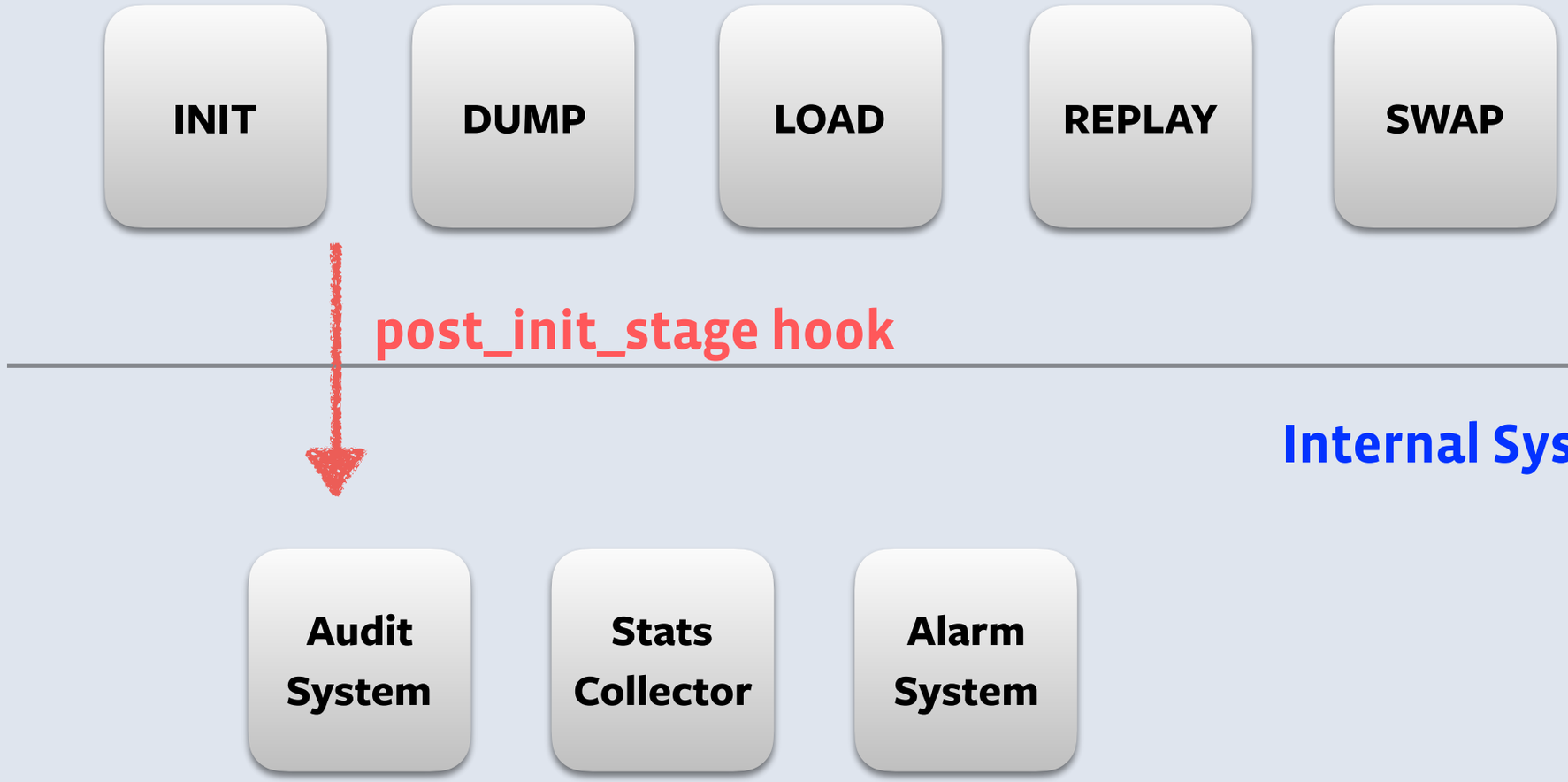
Cats

ID	int	PRIMARY
Name	varchar(10)	
PictureURL	varchar(30)	

Hooks

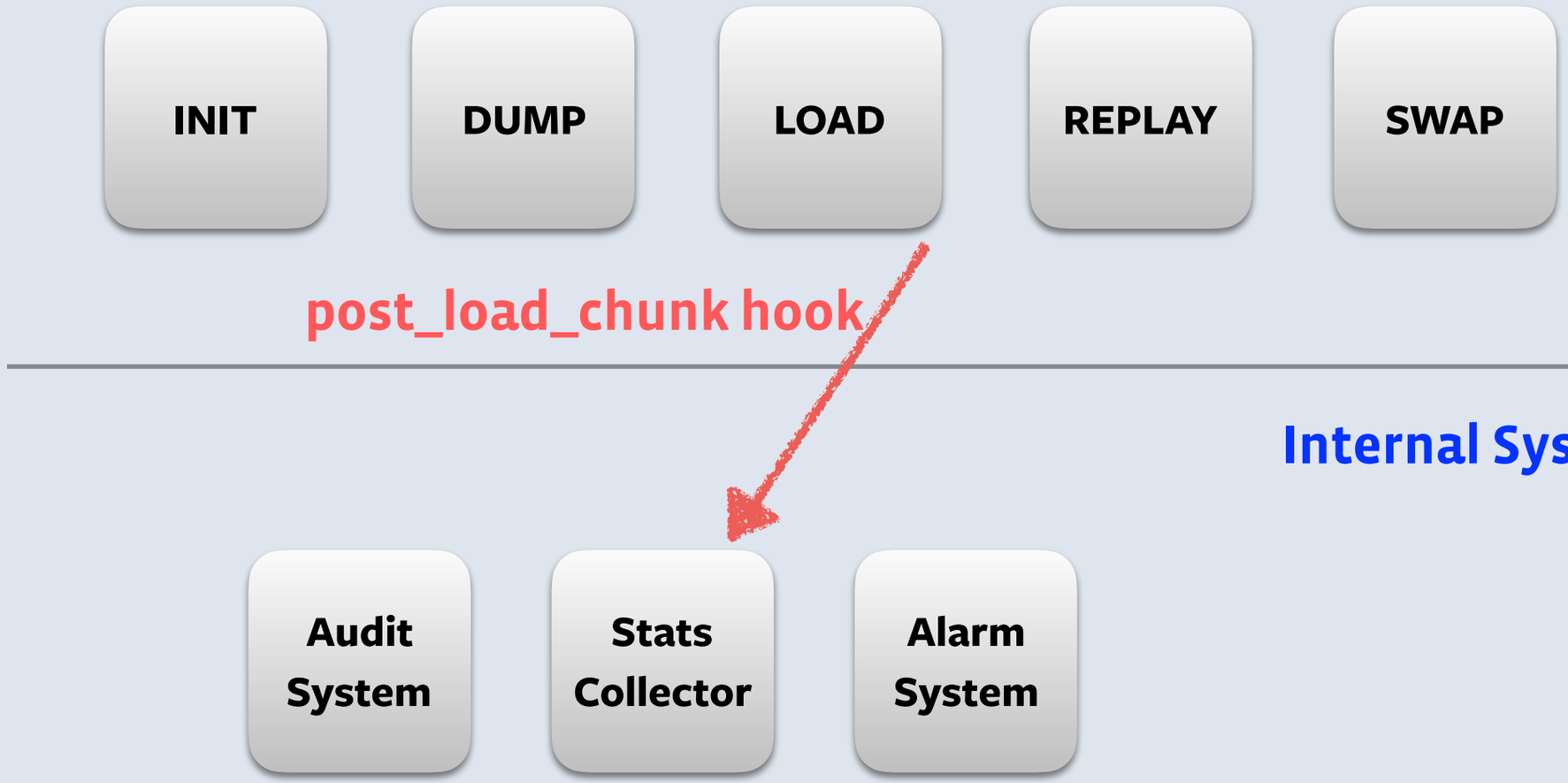
OnlineSchemaChange

Hook



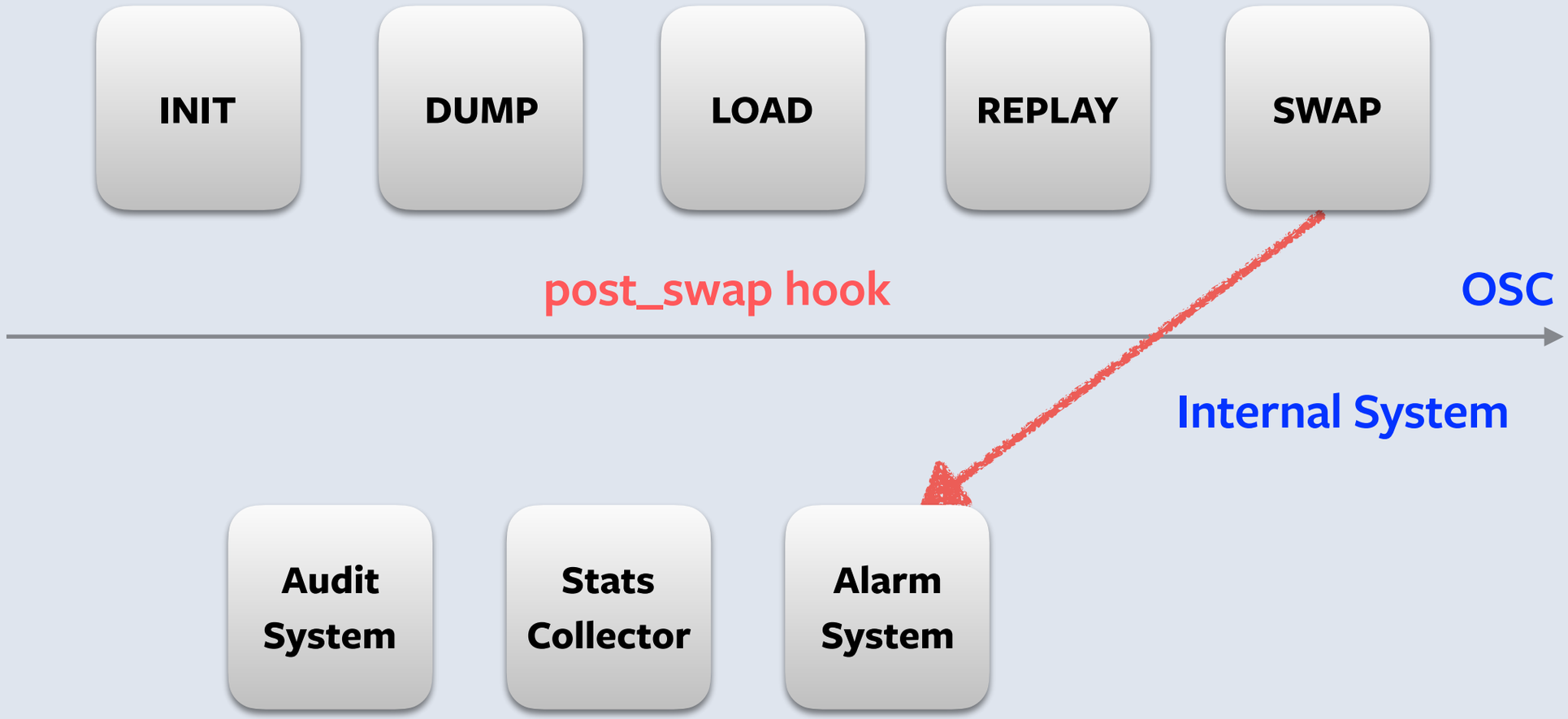
OnlineSchemaChange

Hook



OnlineSchemaChange

Hook



OnlineSchemaChange

What's the difference

	fb-osc	pt-osc	gh-ost
resource control	NO	static threshold	interactive threshold
consistency check	YES	NO	NO
replay speed	Fast	ON THE FLY	Slow
binlog_format	SBR & RBR + FBMYSQL	RBR MASTER + SBR	1+ RBR
MyRocks ready	YES	NO	NO
charset support	=MySQL	=MySQL	=Go

OnlineSchemaChange

What's coming

- No more triggers by using RBR
- Real time resource control
- Parallel mode
- Heuristic decision
- Support more versions

https://github.com/github/gh-ost/blob/master/doc/why-triggerless.md

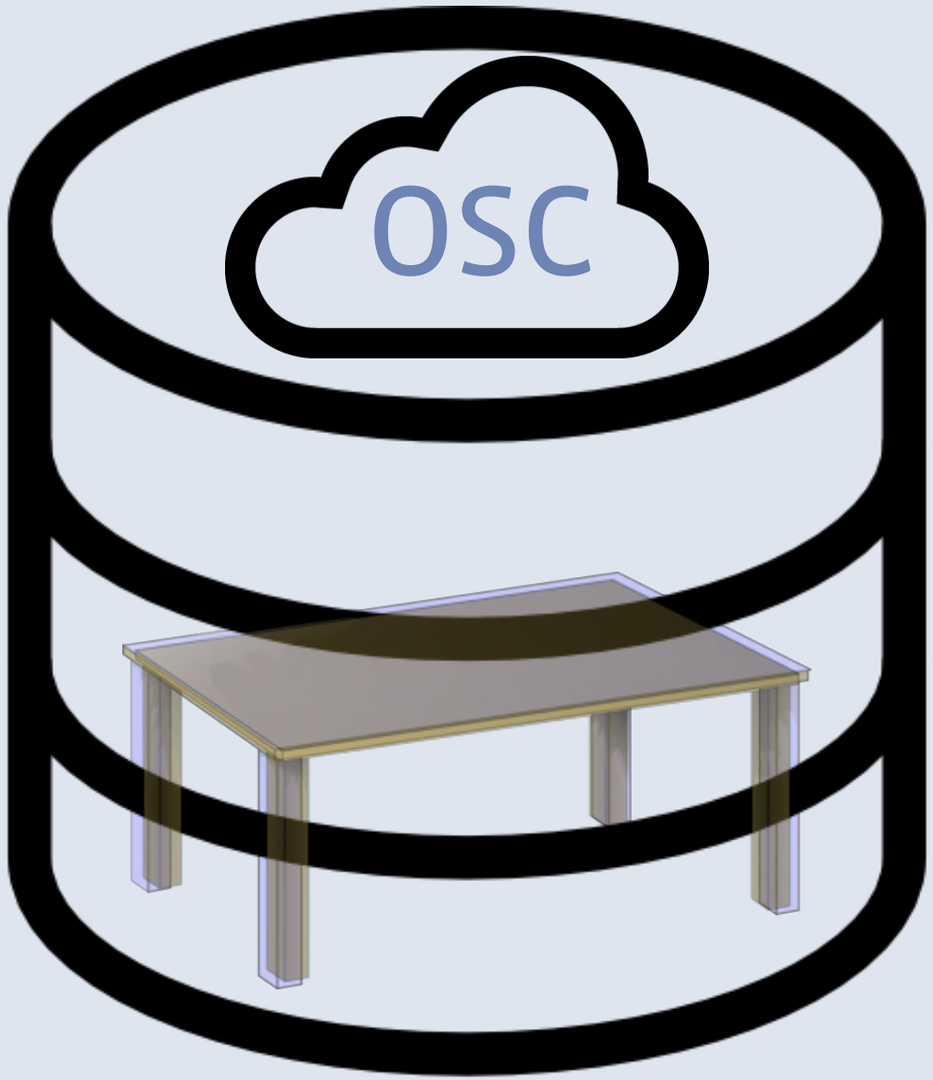
Automatic Online Schema Change

Schema Changes at Scale

Problem

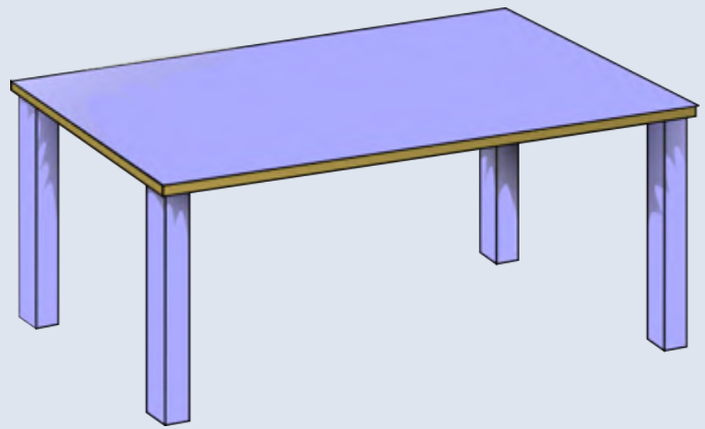
Schema Changes at Scale

Problem



Schema Changes at Scale

Problem

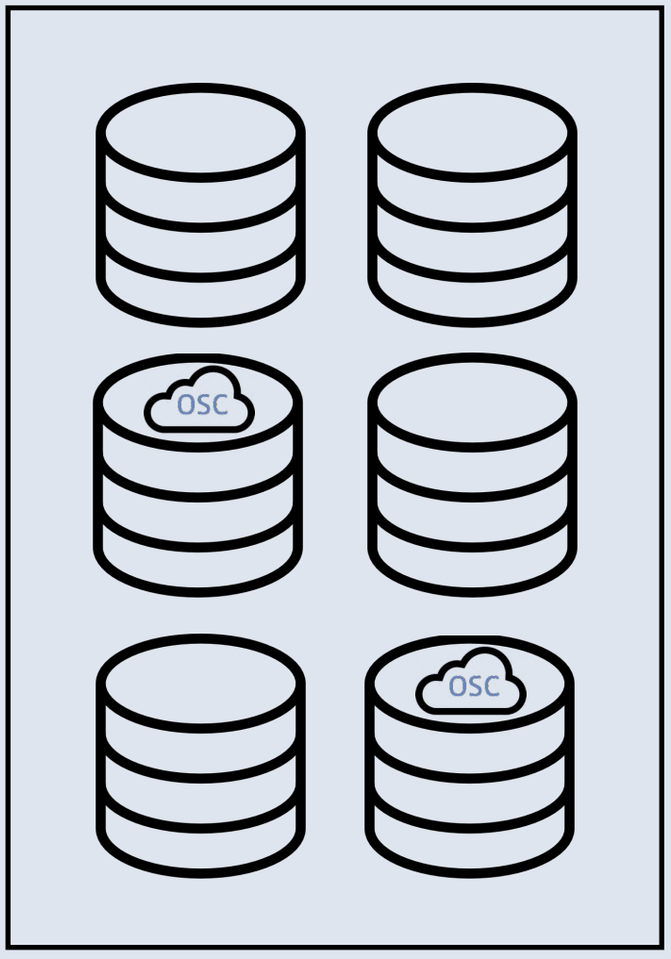


Schema Changes at Scale

Problem

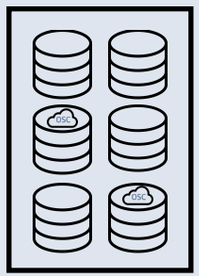
Schema Changes at Scale

Problem



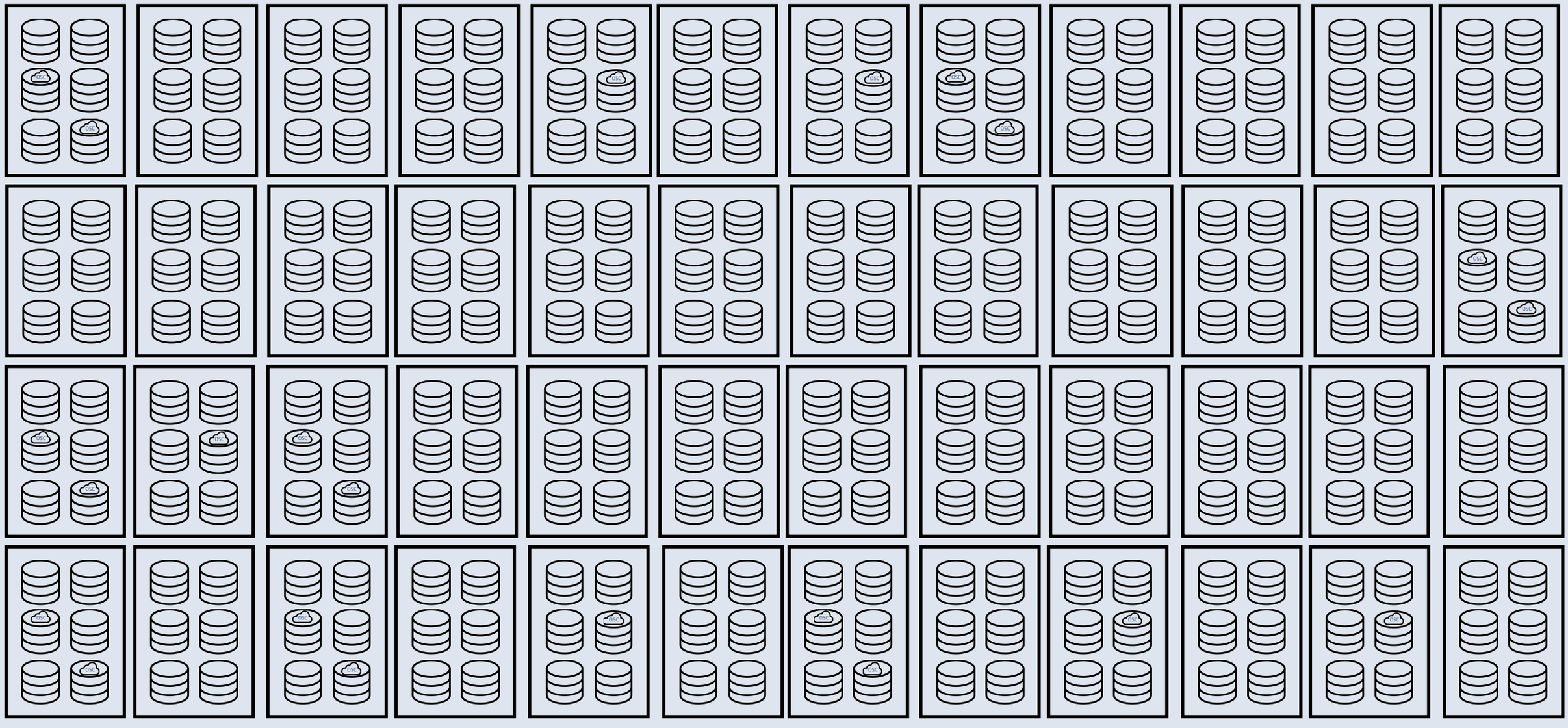
Schema Changes at Scale

Problem



Schema Changes at Scale

Problem

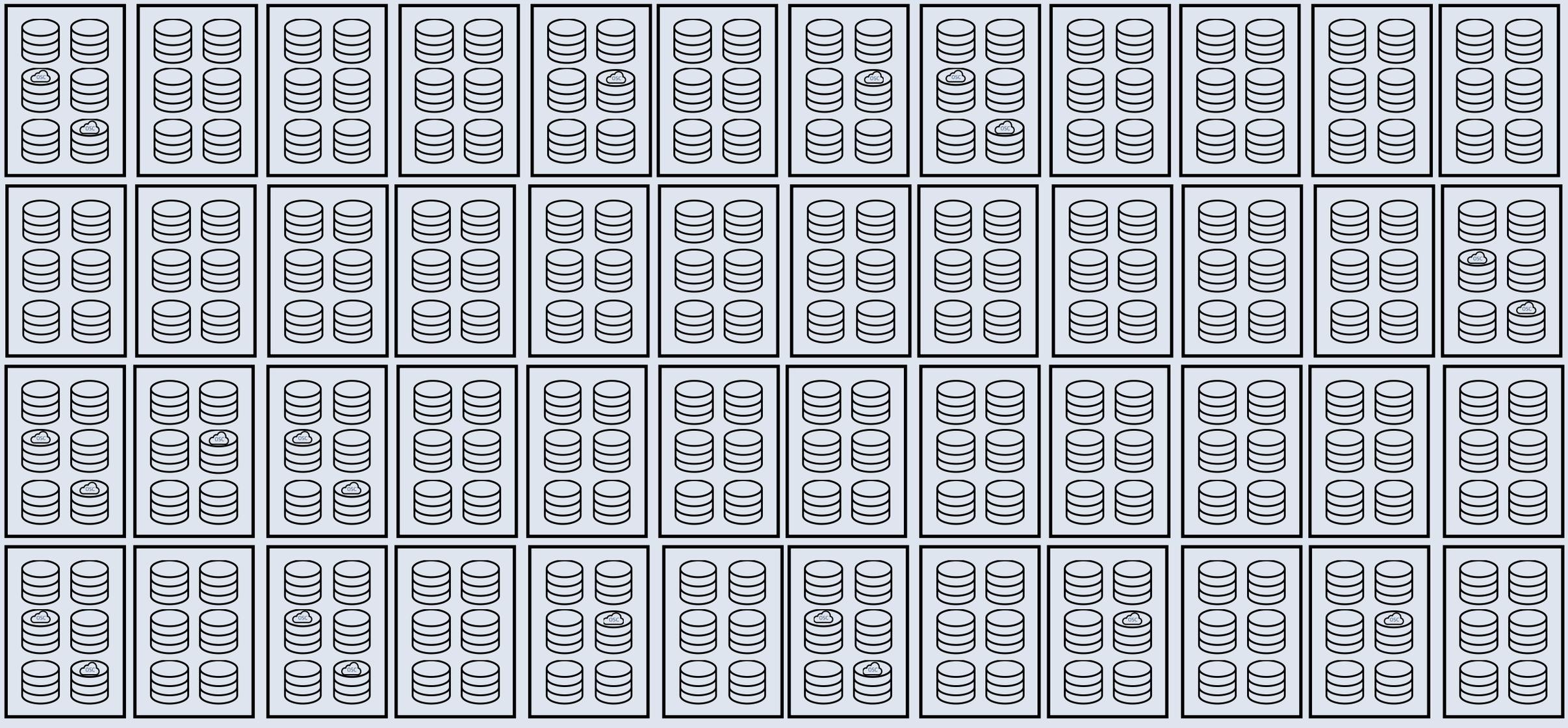


Schema Changes at Scale

Problem

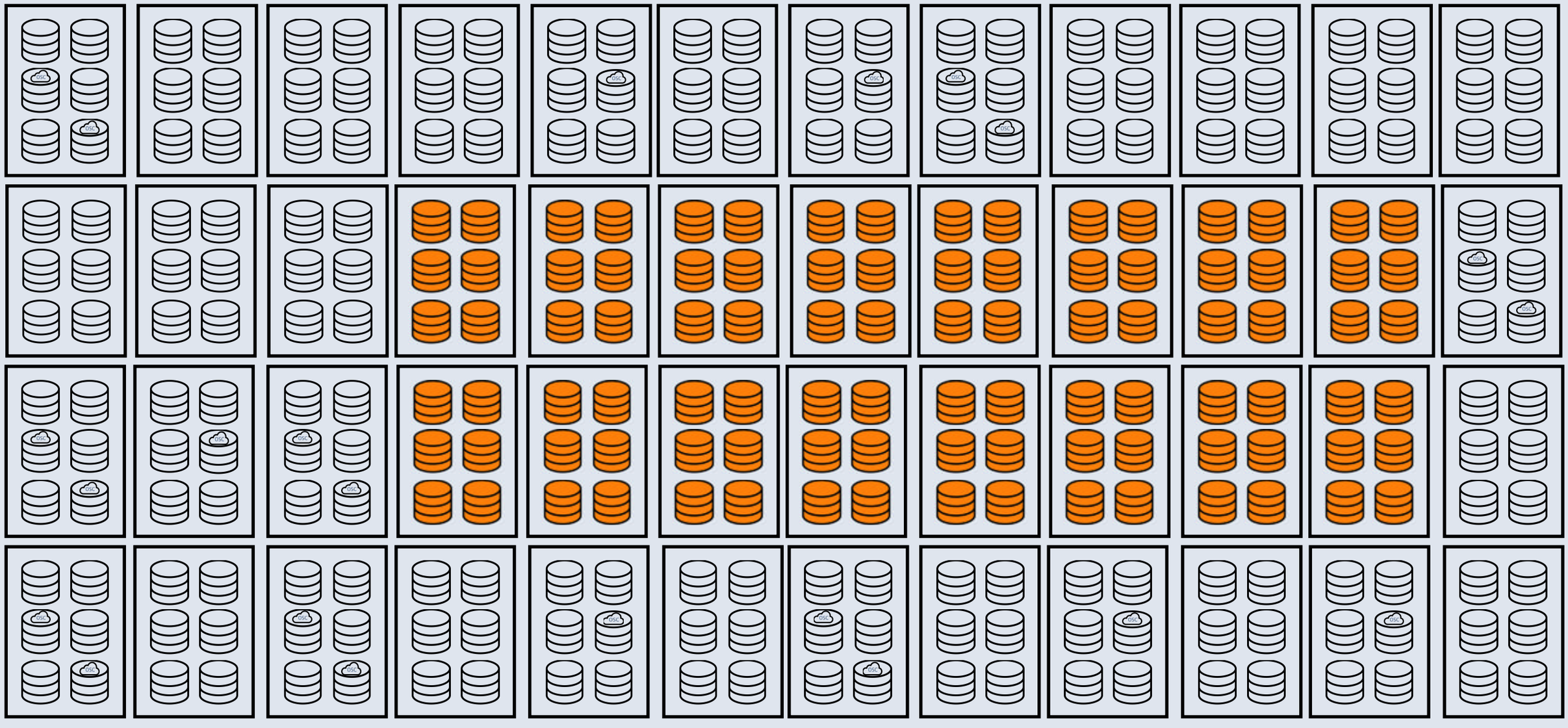
Schema Changes at Scale

Problem



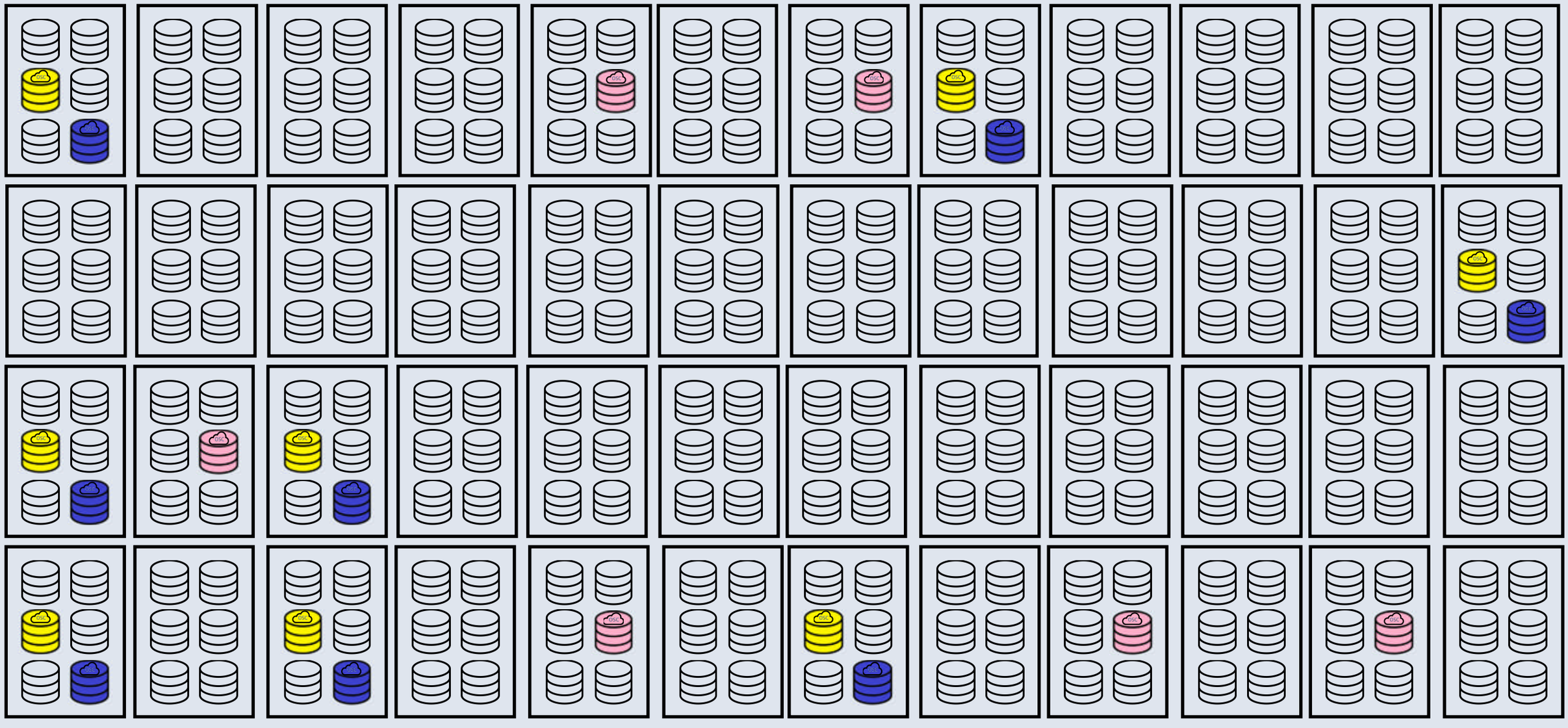
Schema Changes at Scale

Problem



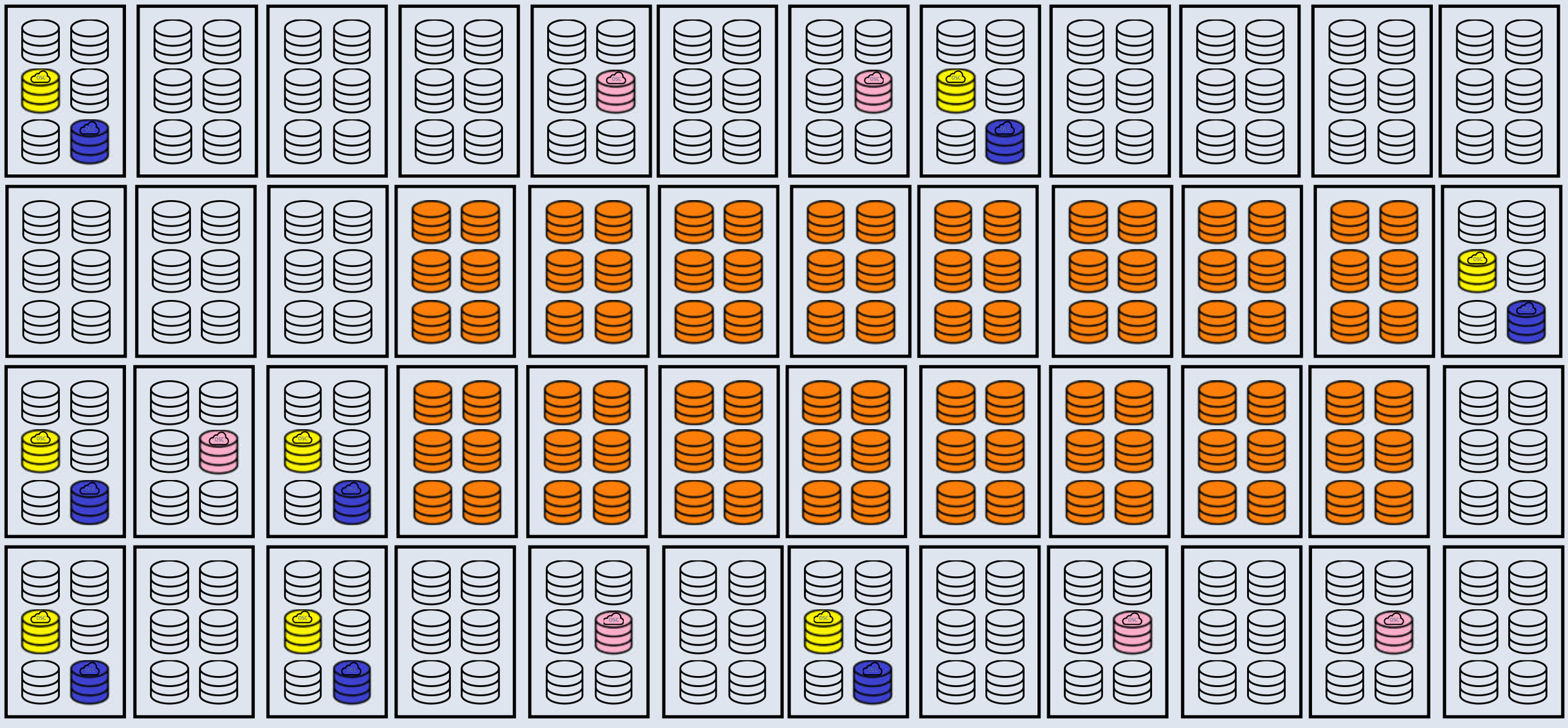
Schema Changes at Scale

Problem



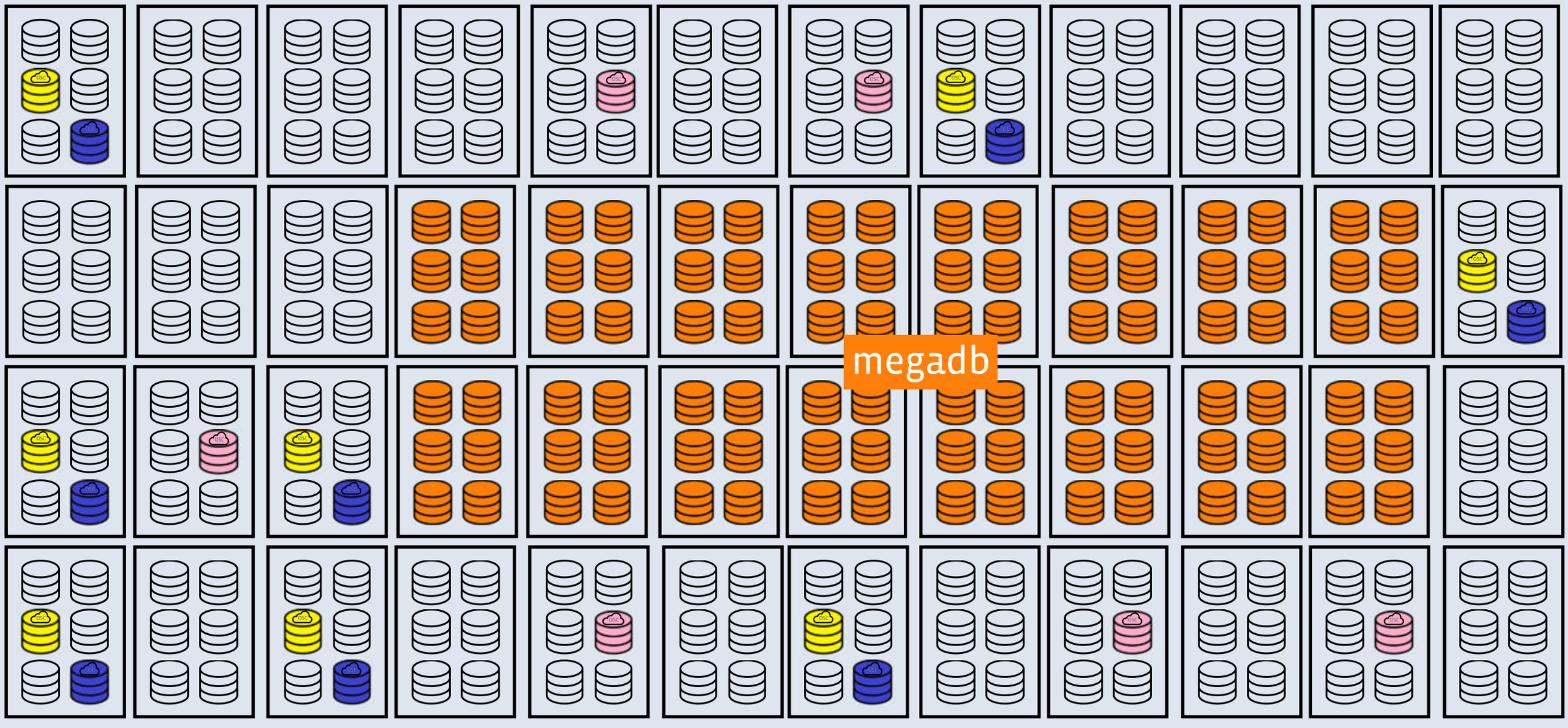
Schema Changes at Scale

Problem



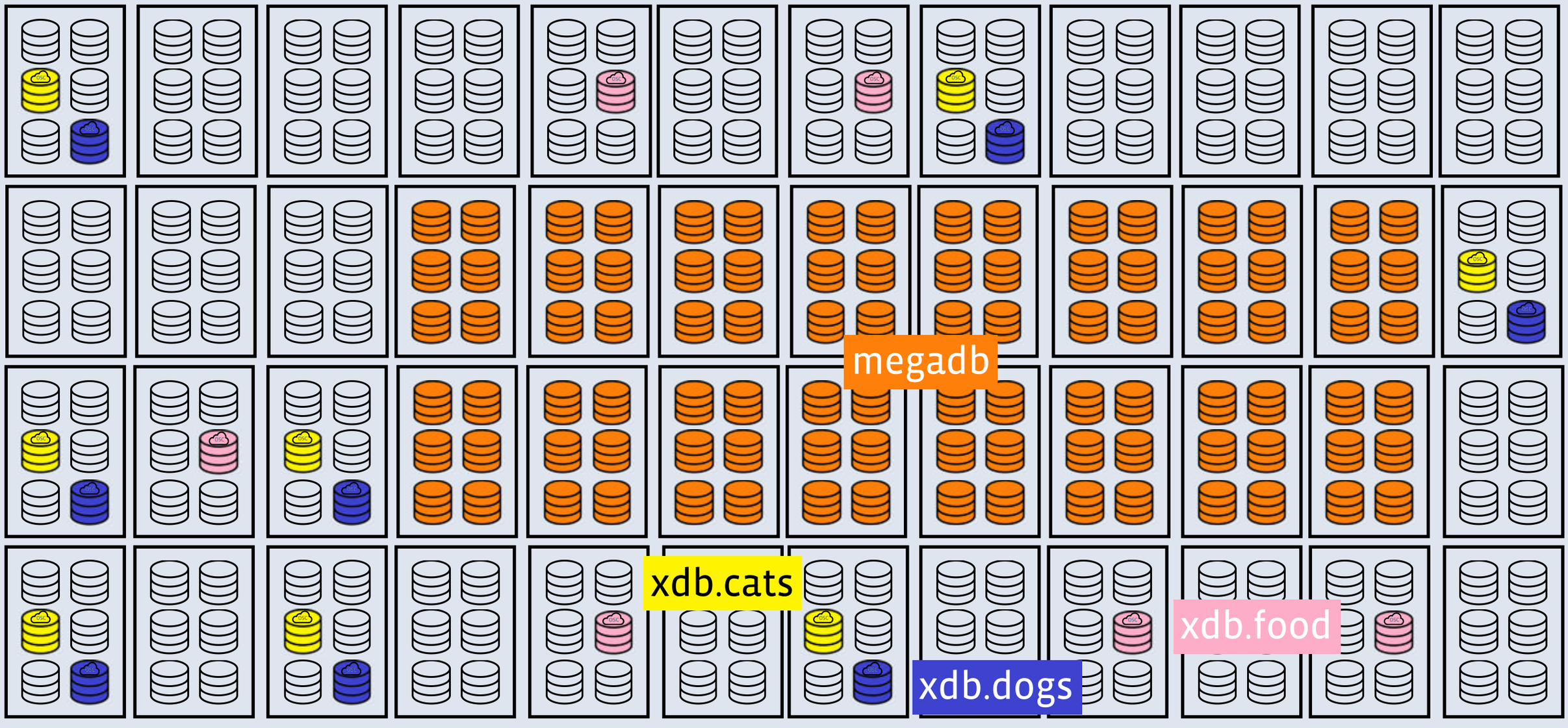
Schema Changes at Scale

Problem



Schema Changes at Scale

Problem



Schema Changes at Scale

Goal

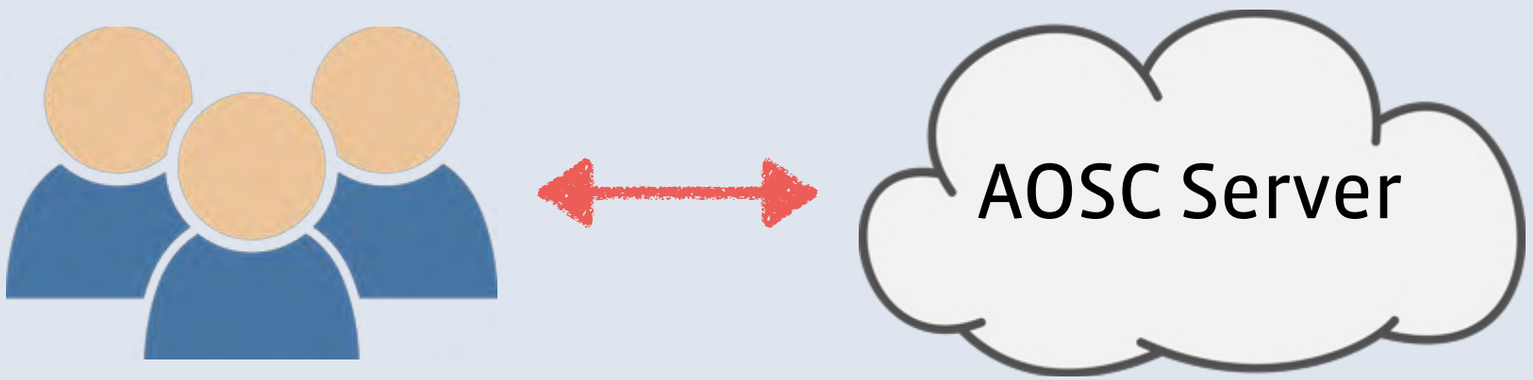
- A given table on a given tier should have an eventually consistent schema across all relevant databases
- Self-service
- Fast
- Safe

Automatic OSC

Solution

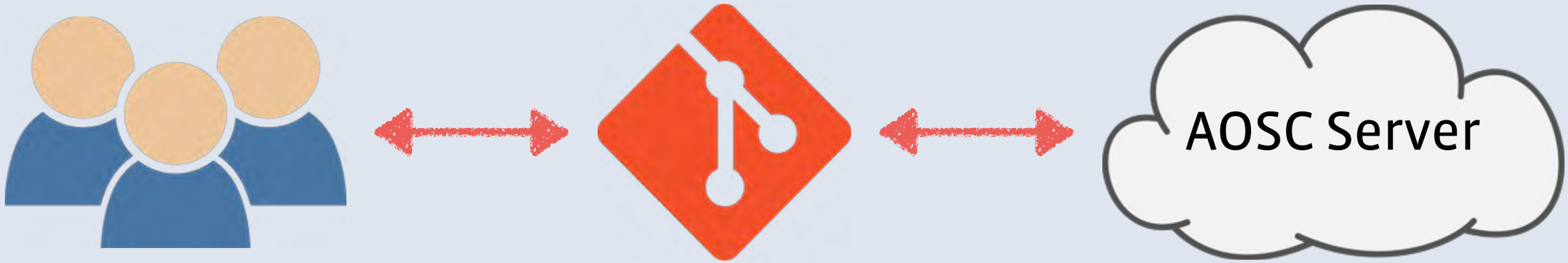
Automatic OSC

Solution



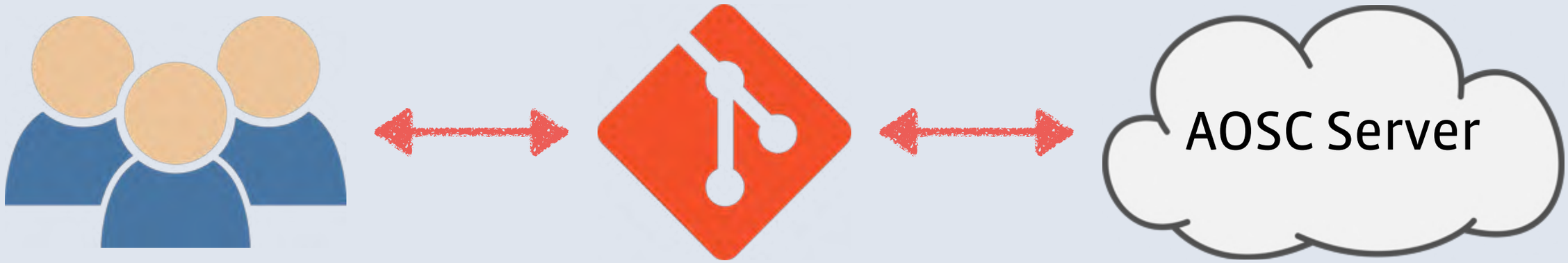
Automatic OSC

Git Repository



Automatic OSC

Git Repository

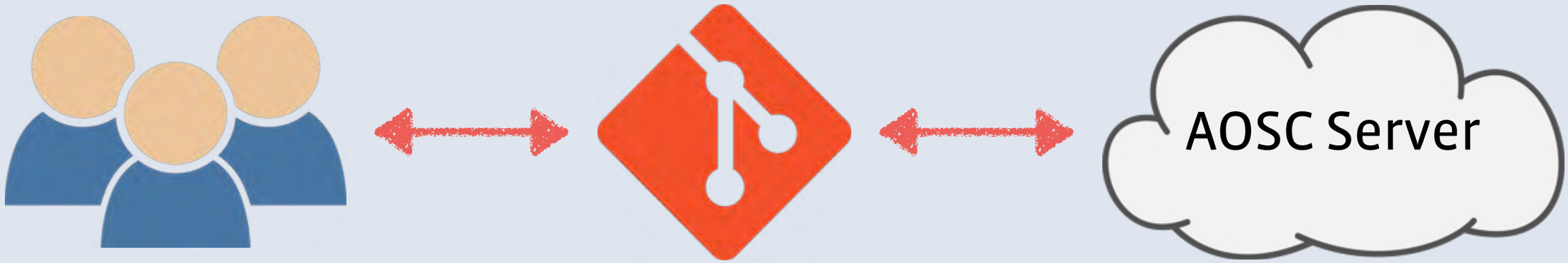



```
~/mysql-schemas $ ls
megadb
xdb.cats
xdb.dogs
xdb.food

~/mysql-schemas $ ls xdb.food
apple
banana
pear
```

```
~/mysql-schemas $ cat xdb.food/apple
CREATE TABLE `apple` (
  `id` int(11) AUTO_INCREMENT,
  `name` varchar(32),
  `quantity` int(11),
  PRIMARY KEY (`id`)
) ENGINE=RocksDB
```

Automatic OSC

Git Repository

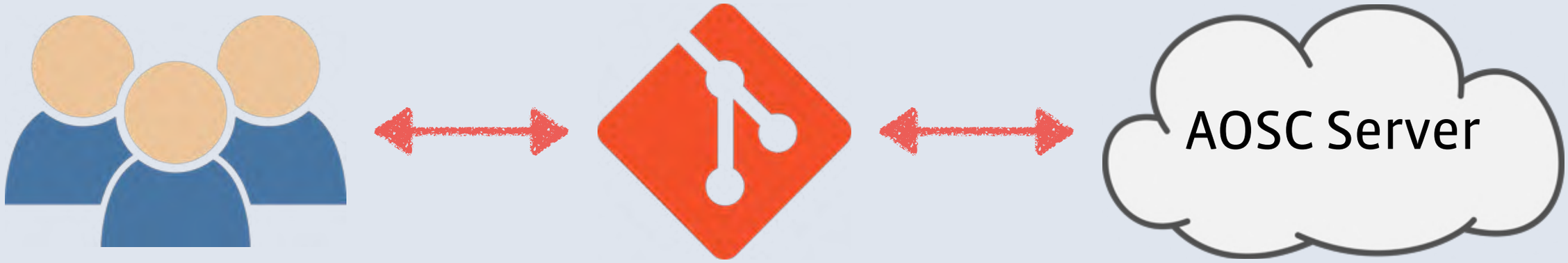



```
~/mysql-schemas $ cat xdb.food/apple
CREATE TABLE `apple` (
  `id` int(11) AUTO_INCREMENT,
  `name` varchar(32),
  `quantity` int(11),
  PRIMARY KEY (`id`)
) ENGINE=RocksDB
```

```
~/mysql-schemas $ ...

~/mysql-schemas $ git diff
CREATE TABLE `apple` (
  `id` int(11) AUTO_INCREMENT,
  `name` varchar(32),
  `quantity` int(11),
+ `color` enum('red','green'),
  PRIMARY KEY (`id`)
) ENGINE=RocksDB
```

Automatic OSC

Git Repository

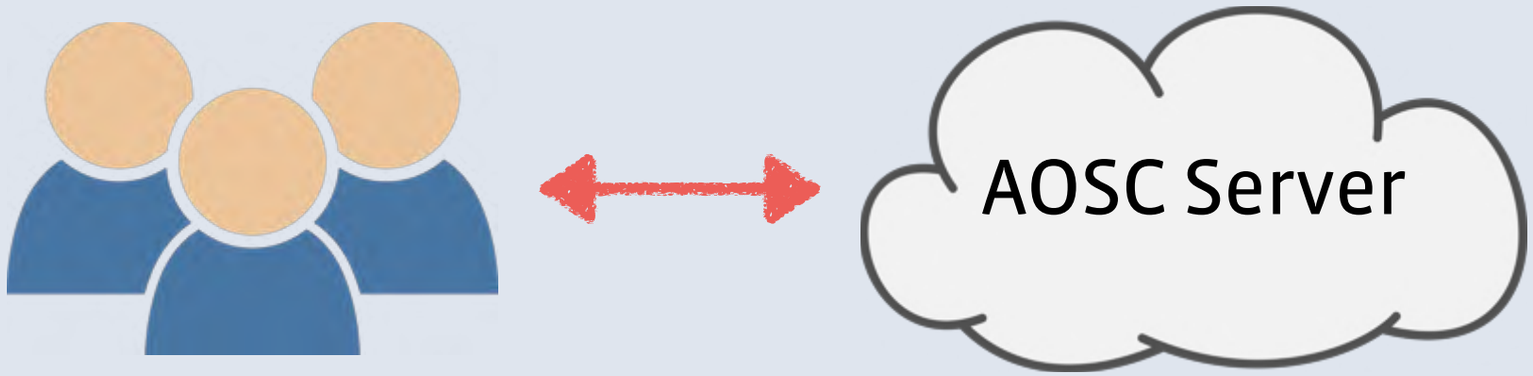



```
~/mysql-schemas $ ...

~/mysql-schemas $ git diff
CREATE TABLE `apple` (
  `id` int(11) AUTO_INCREMENT,
  `name` varchar(32),
  `quantity` int(11),
+ `color` enum('red','green'),
  PRIMARY KEY (`id`)
) ENGINE=RocksDB
```

- Versioning
- Code review /Auto linter
- Commit hook

Automatic OSC

User Interface




```
~/mysql-schemas $ ...

~/mysql-schemas $ git diff
CREATE TABLE `apple` (
  `id` int(11) AUTO_INCREMENT,
  `name` varchar(32),
  `quantity` int(11),
+ `color` enum('red','green'),
  PRIMARY KEY (`id`)
) ENGINE=RocksDB
```

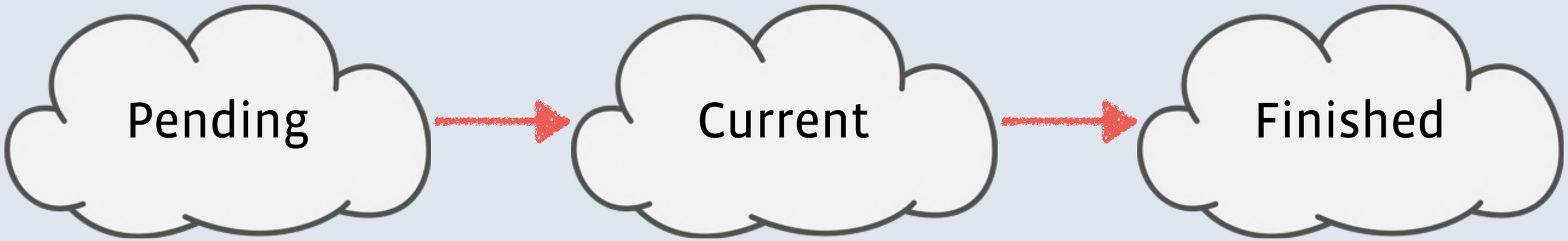
```
~ $ aosc allow xdb.food apple
...

~ $ aosc cancel xdb.food apple
...

~ $ aosc hold xdb.food apple
...

~ $ aosc continue xdb.food apple
...
```

Automatic OSC

Global Coordination

Automatic OSC

Alter-Level Coordination

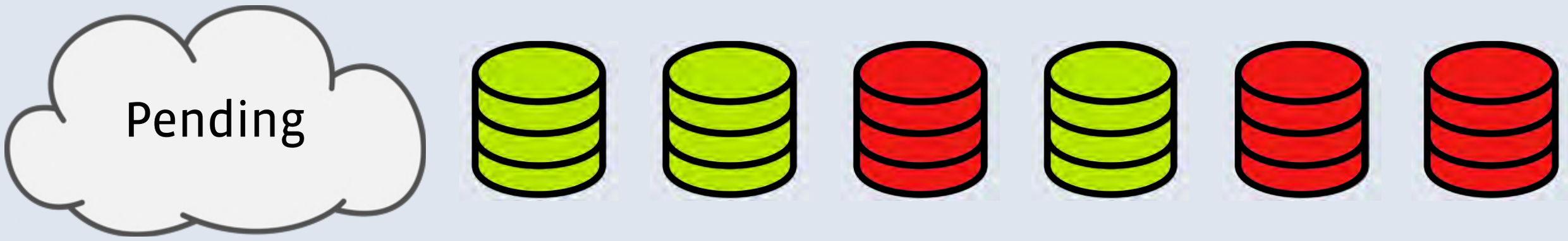


Automatic OSC

Alter-Level Coordination: Pending

Automatic OSC

Alter-Level Coordination: Pending




```
~/mysql-schemas $ git diff
CREATE TABLE `apple` (
  `id` int(11) AUTO_INCREMENT,
  `name` varchar(32),
  `quantity` int(11),
+ `color` enum('red','green'),
  PRIMARY KEY (`id`)
) ENGINE=RocksDB
```

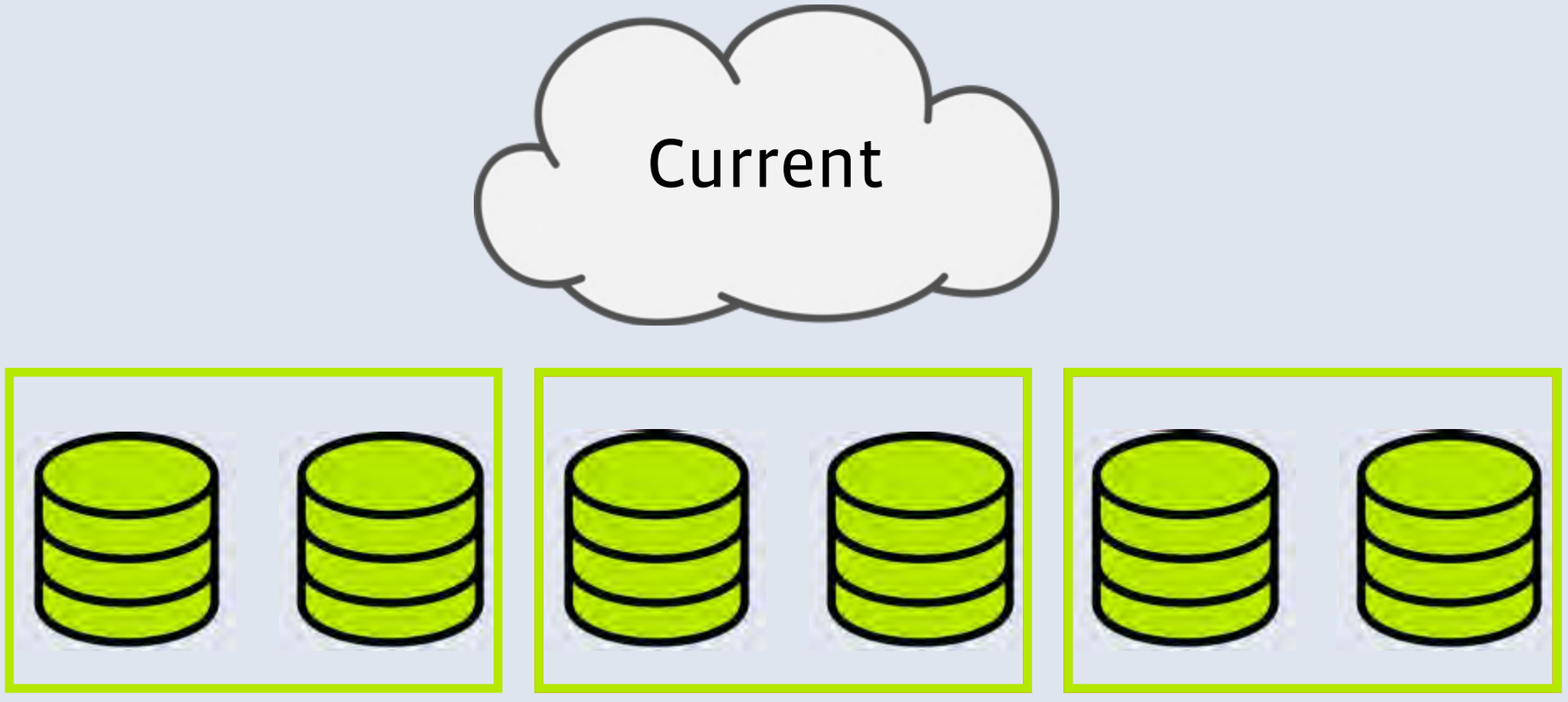
```
db3 > SHOW CREATE TABLE `apple`;
CREATE TABLE `apple` (
  `id` int(11) AUTO_INCREMENT,
  `name` varchar(32),
  `quantity` int(11),
  PRIMARY KEY (`id`)
) ENGINE=RocksDB
```

Automatic OSC

Alter-Level Coordination: Pending

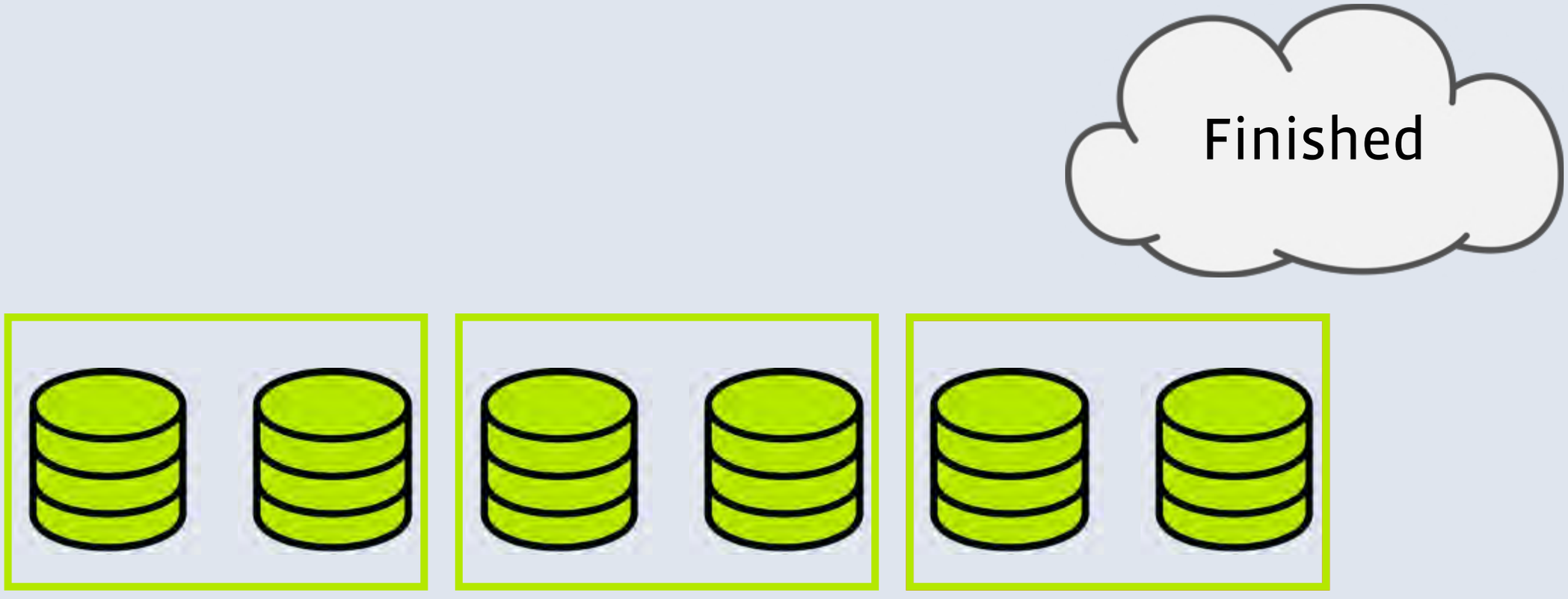
Automatic OSC

Alter-Level Coordination: Current



Automatic OSC

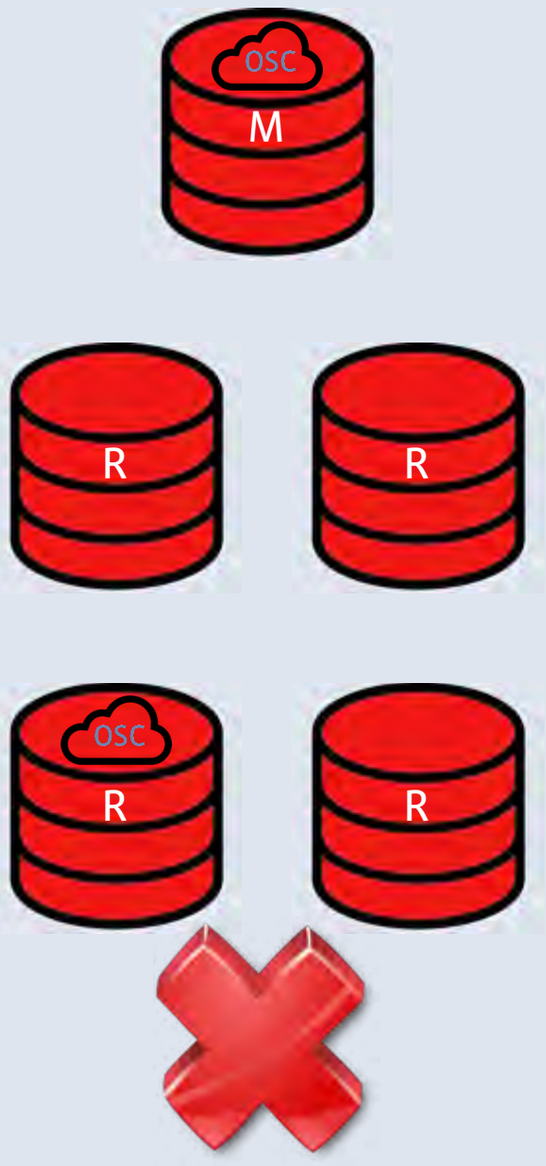
Alter-Level Coordination: Finished



Replicaset-Level Coordination

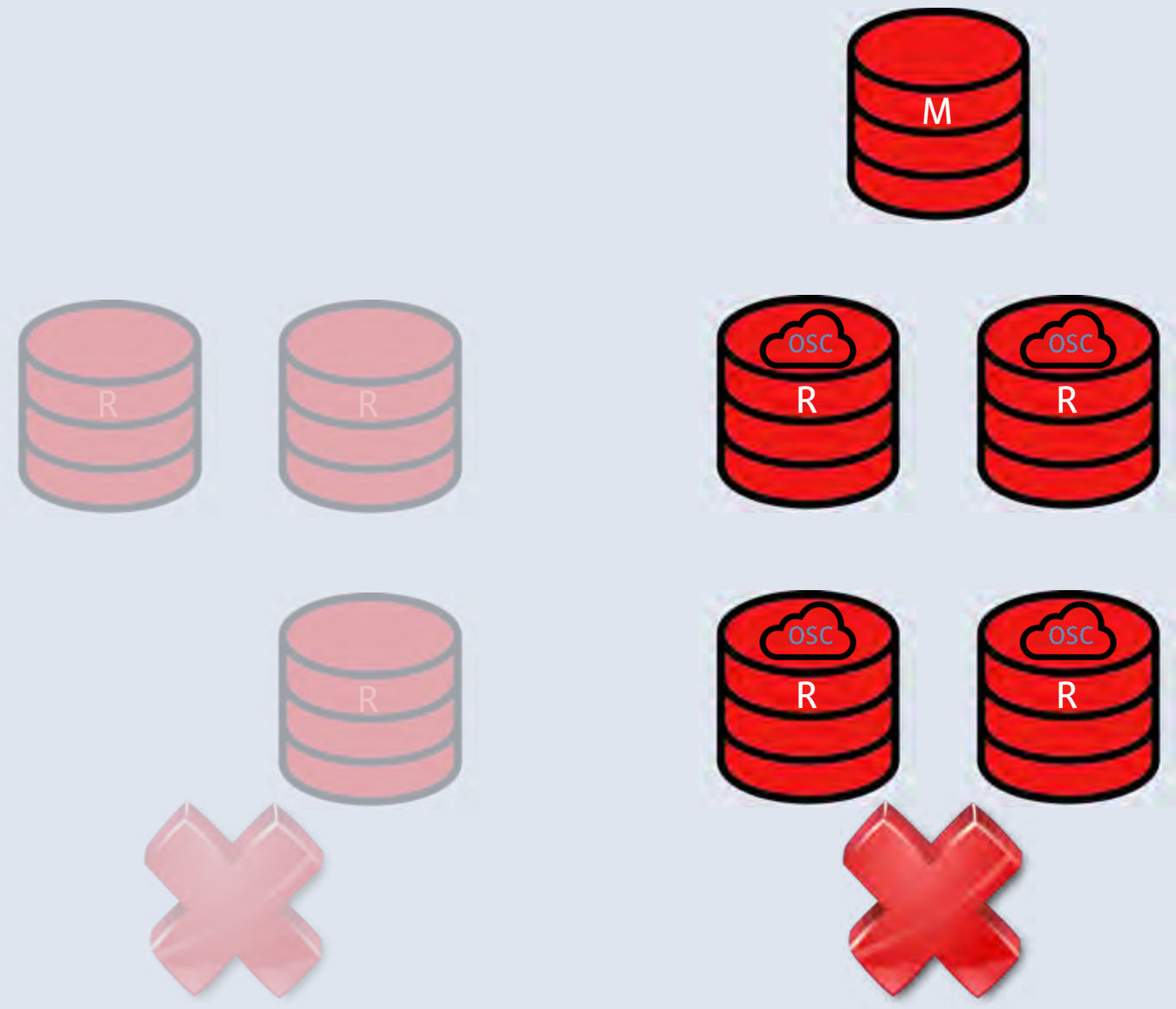
Automatic OSC

Replicaset-Level Coordination



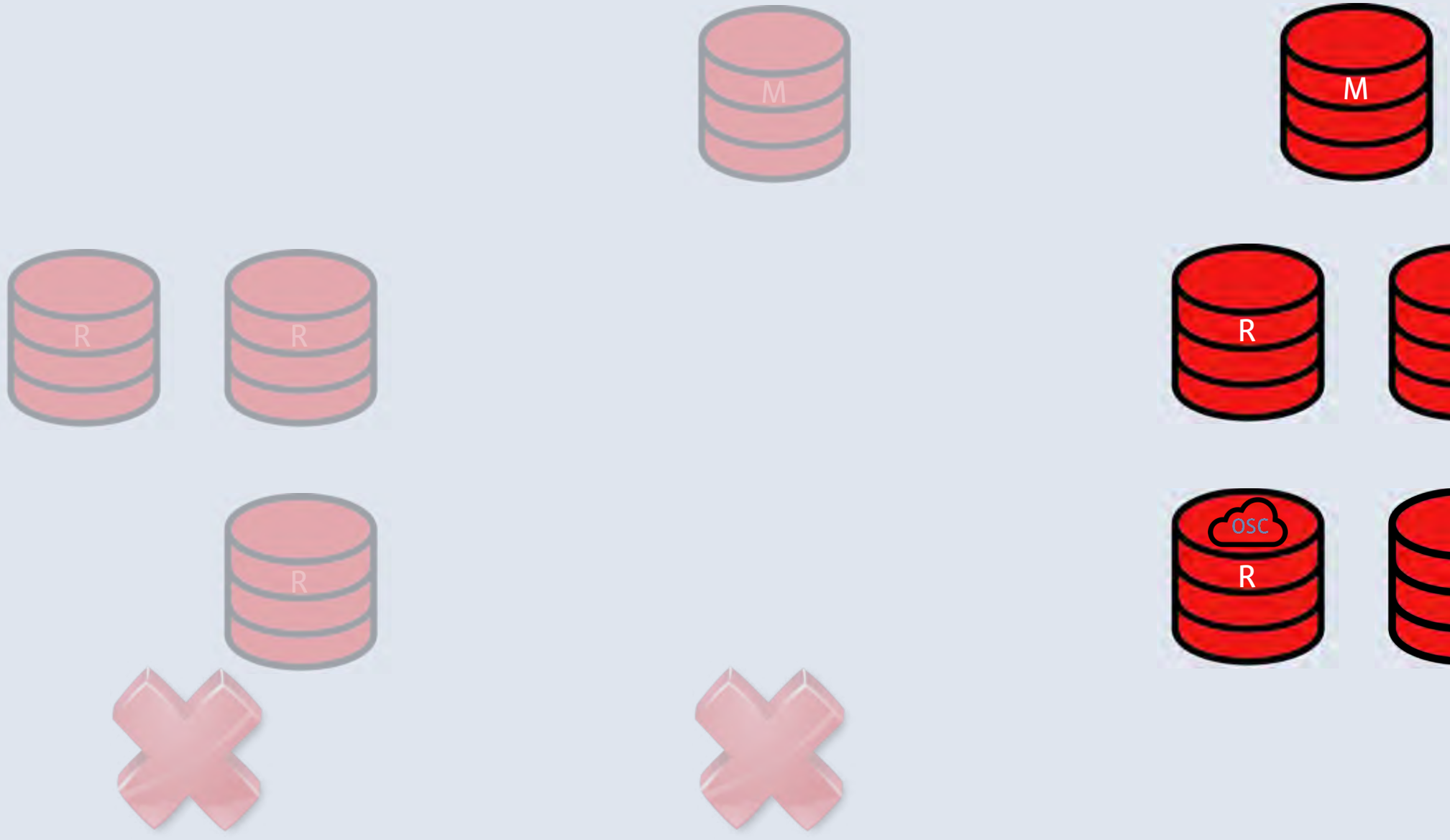
Automatic OSC

Replicaset-Level Coordination



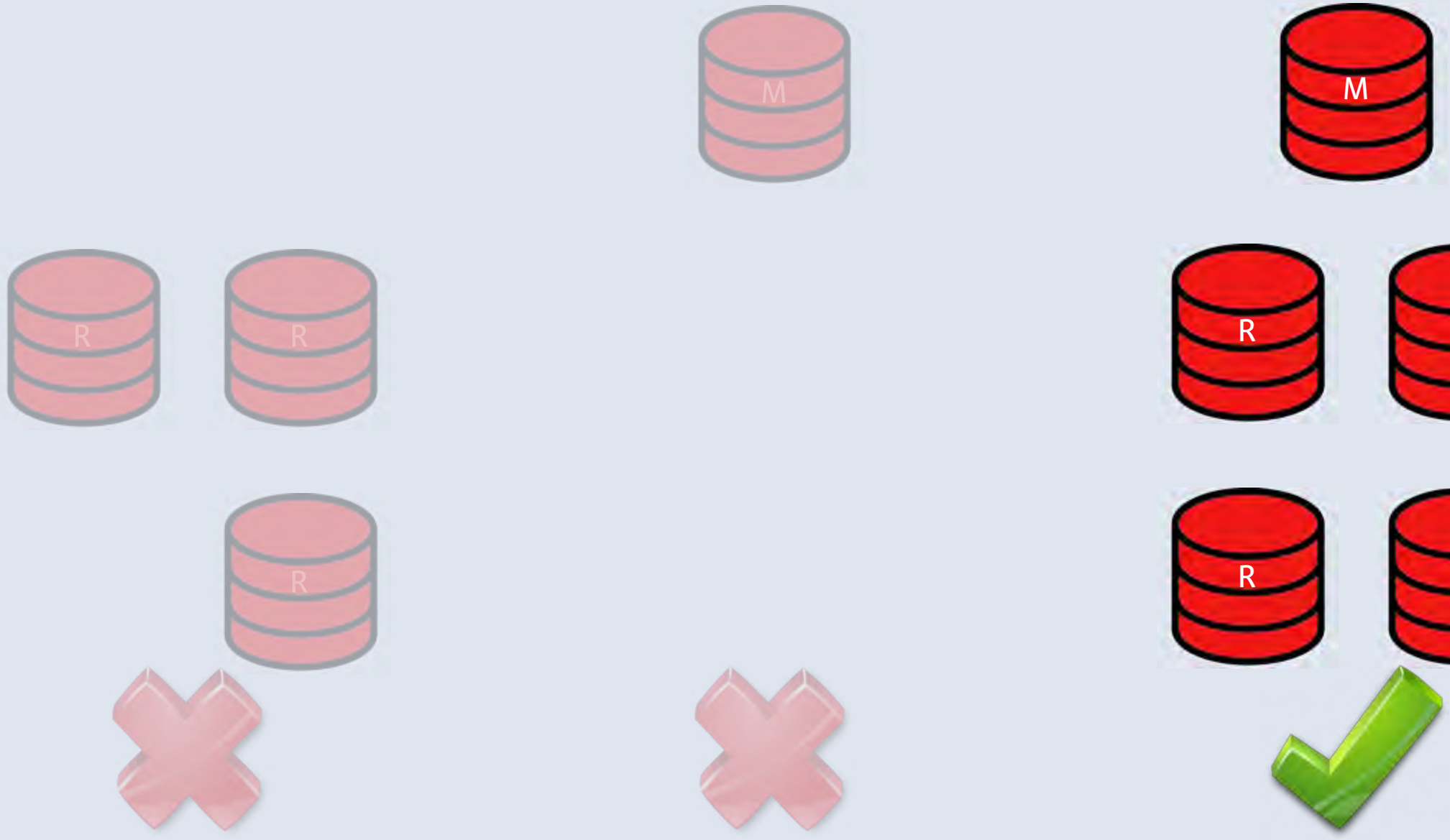
Automatic OSC

Replicaset-Level Coordination



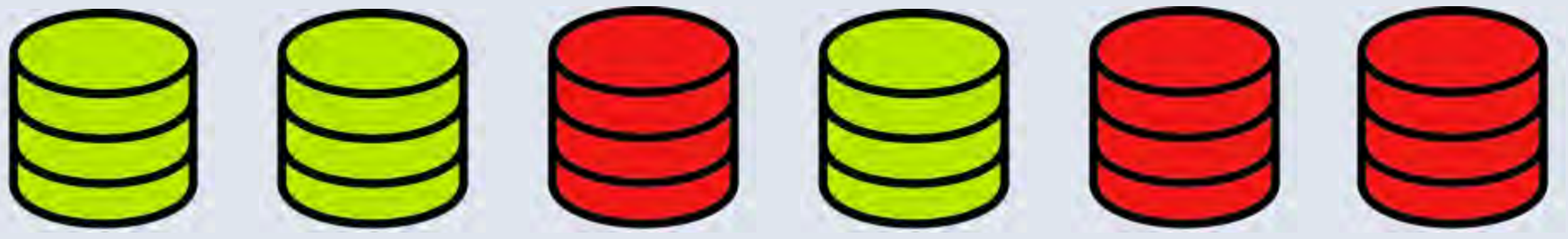
Automatic OSC

Replicaset-Level Coordination



Automatic OSC

Replicaset-Level Coordination

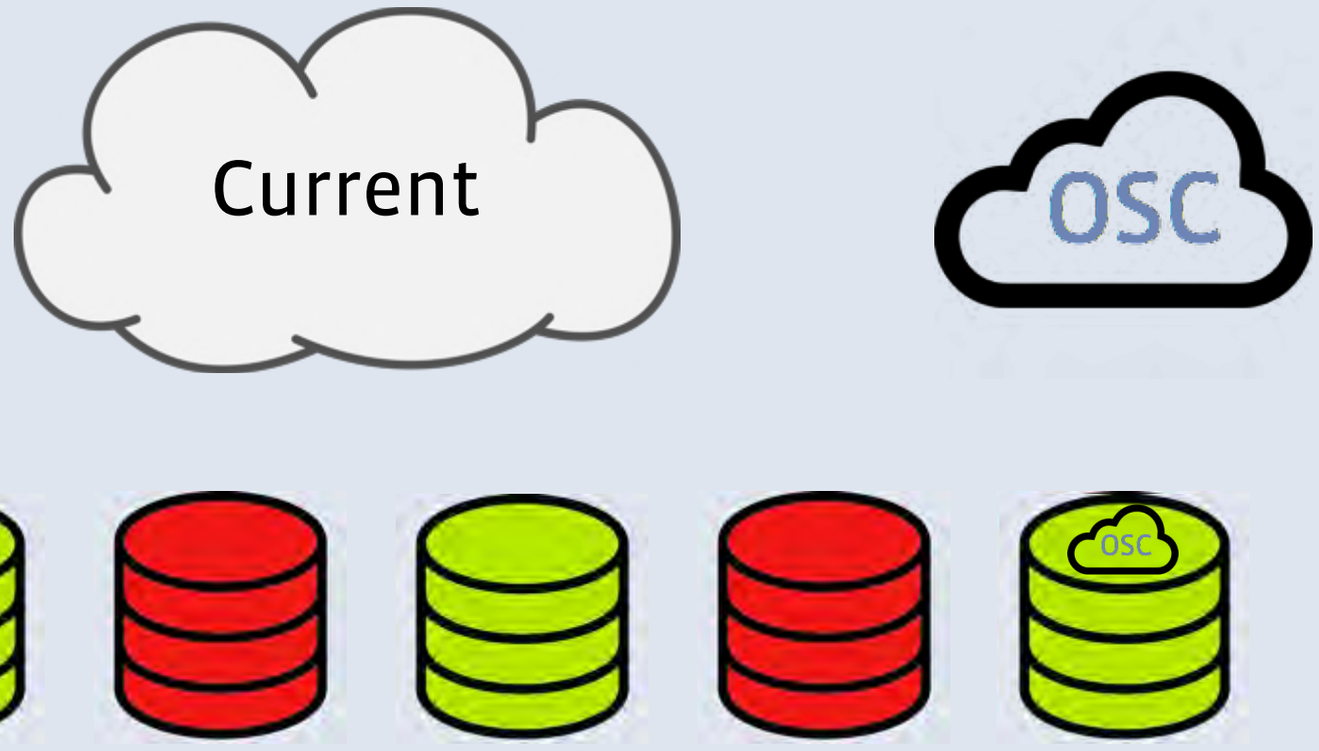


Automatic OSC

Replicaset-Level Coordination

Automatic OSC

Replicaset-Level Coordination



Automatic OSC

Replicaset-Level Coordination

Automatic OSC

Replicaset-Level Coordination

Automatic OSC

Replicaset-Level Coordination

Automatic OSC

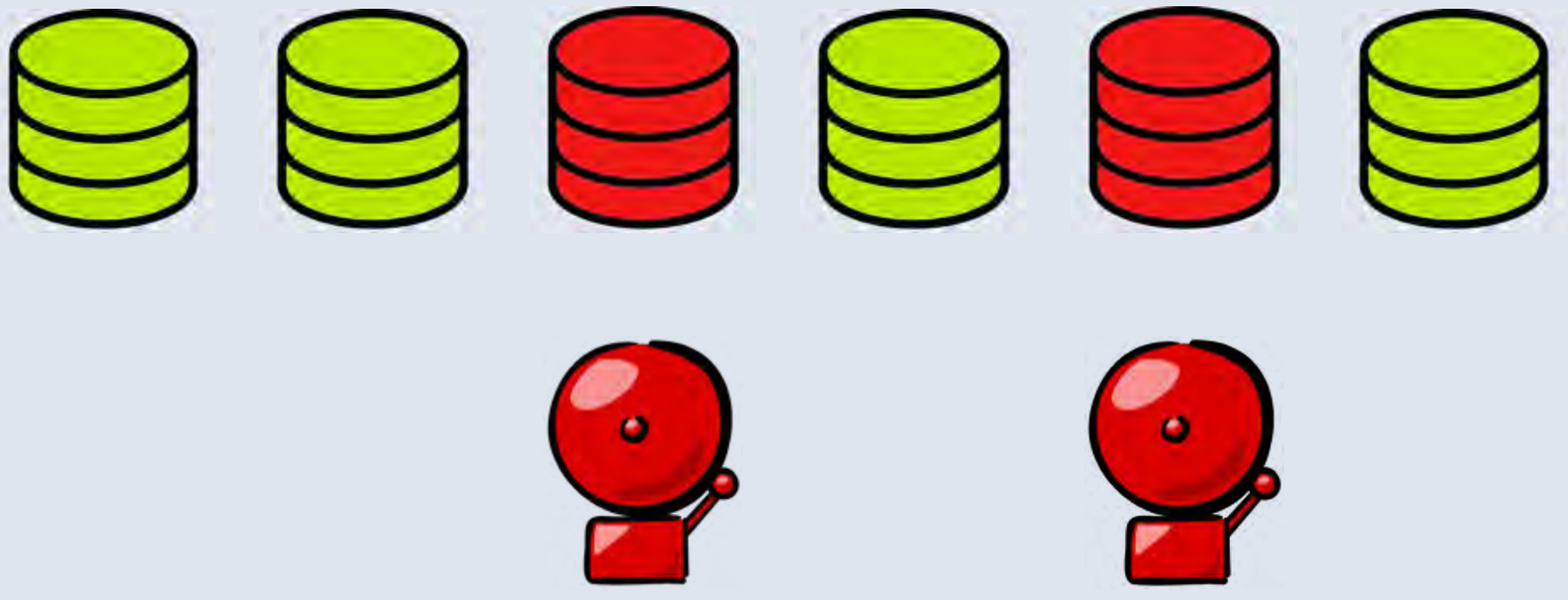
Replicaset-Level Coordination

Automatic OSC

Replicaset-Level Coordination

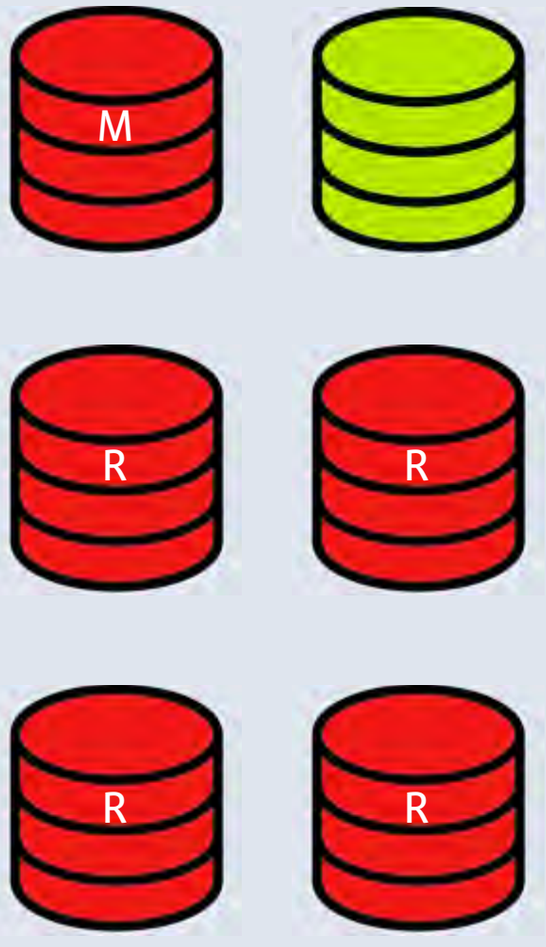
Automatic OSC

Replicaset-Level Coordination



Automatic OSC

Replicaset-Level Coordination

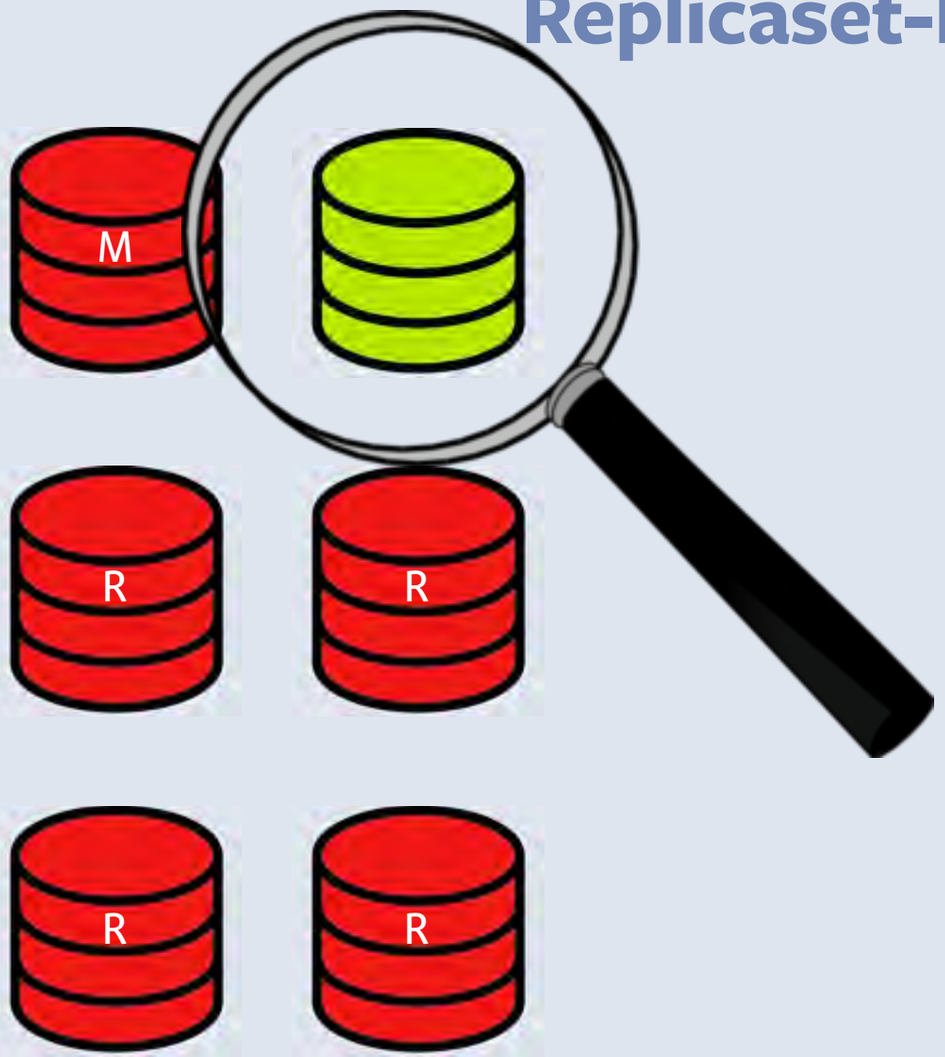


Automatic OSC

Replicaset-Level Coordination

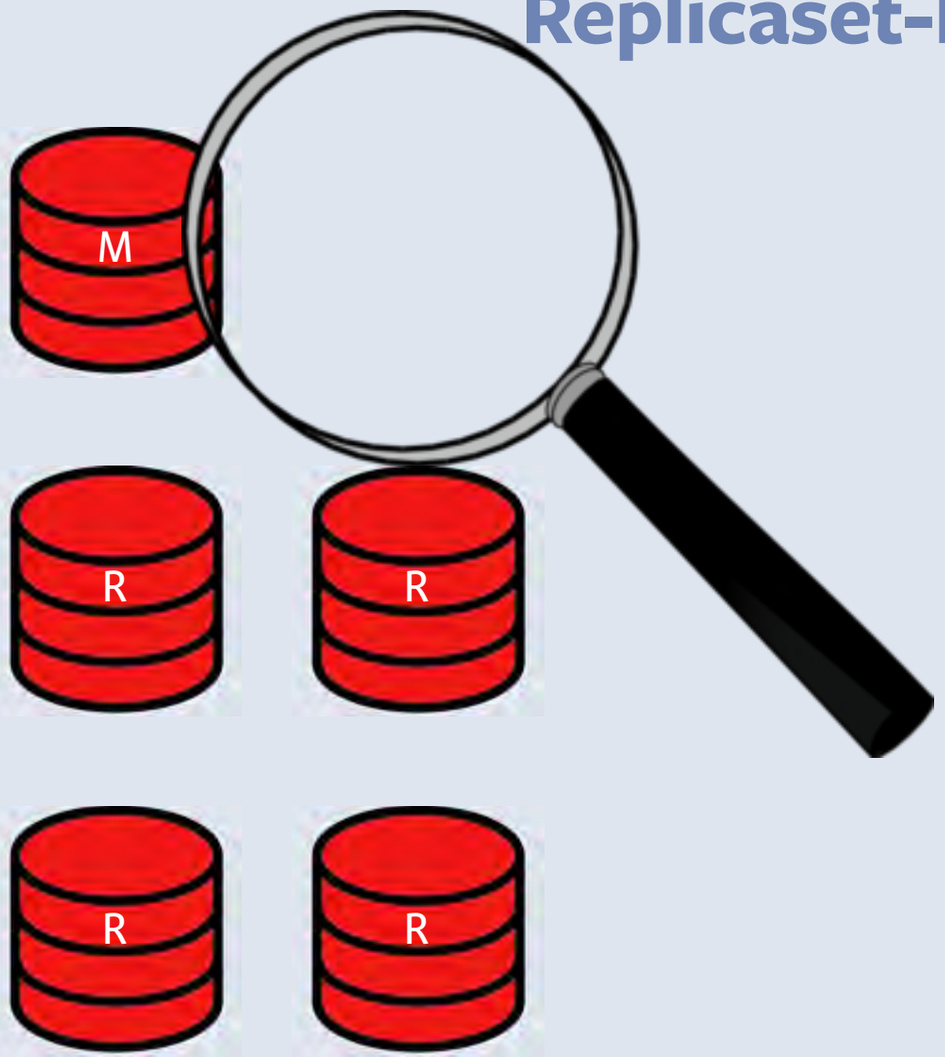
Automatic OSC

Replicaset-Level Coordination



Automatic OSC

Replicaset-Level Coordination

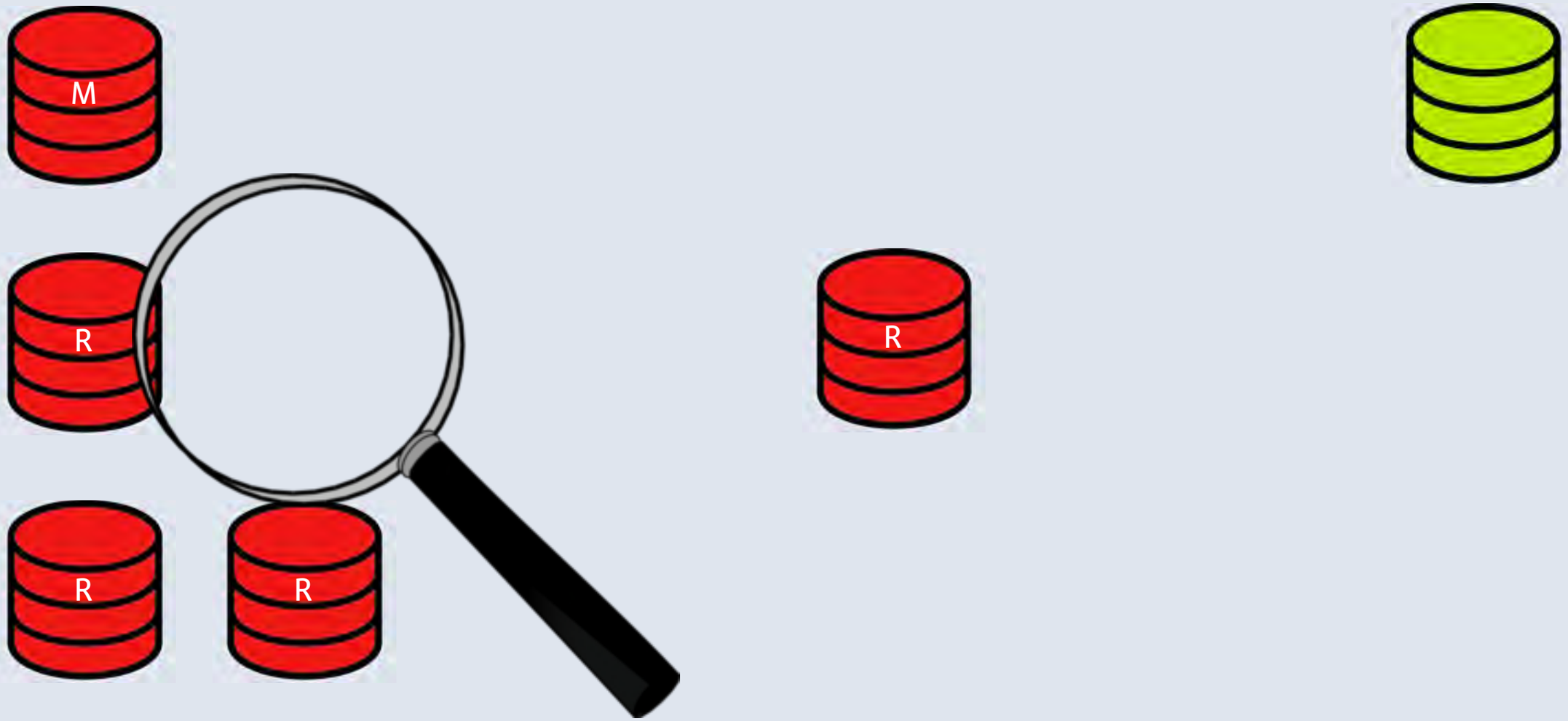


Automatic OSC

Replicaset-Level Coordination

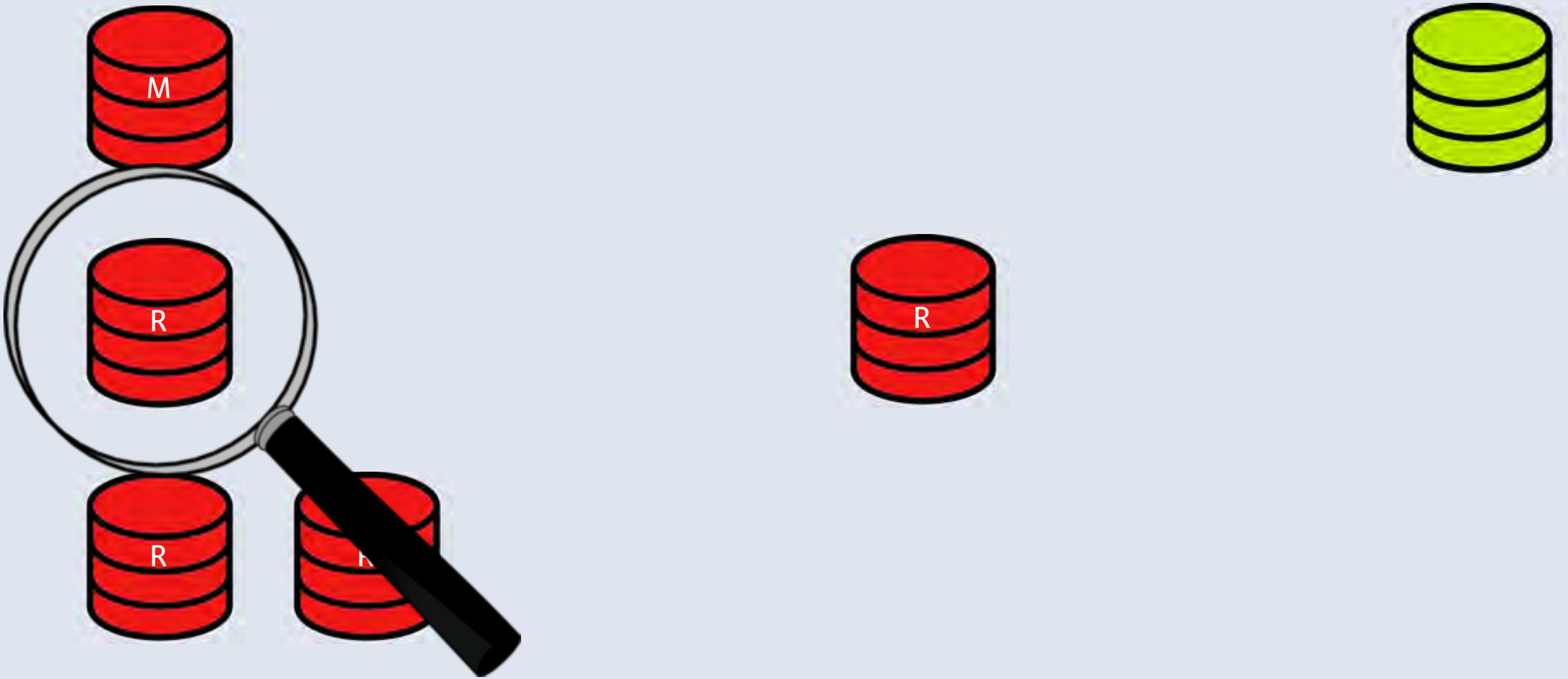
Automatic OSC

Replicaset-Level Coordination



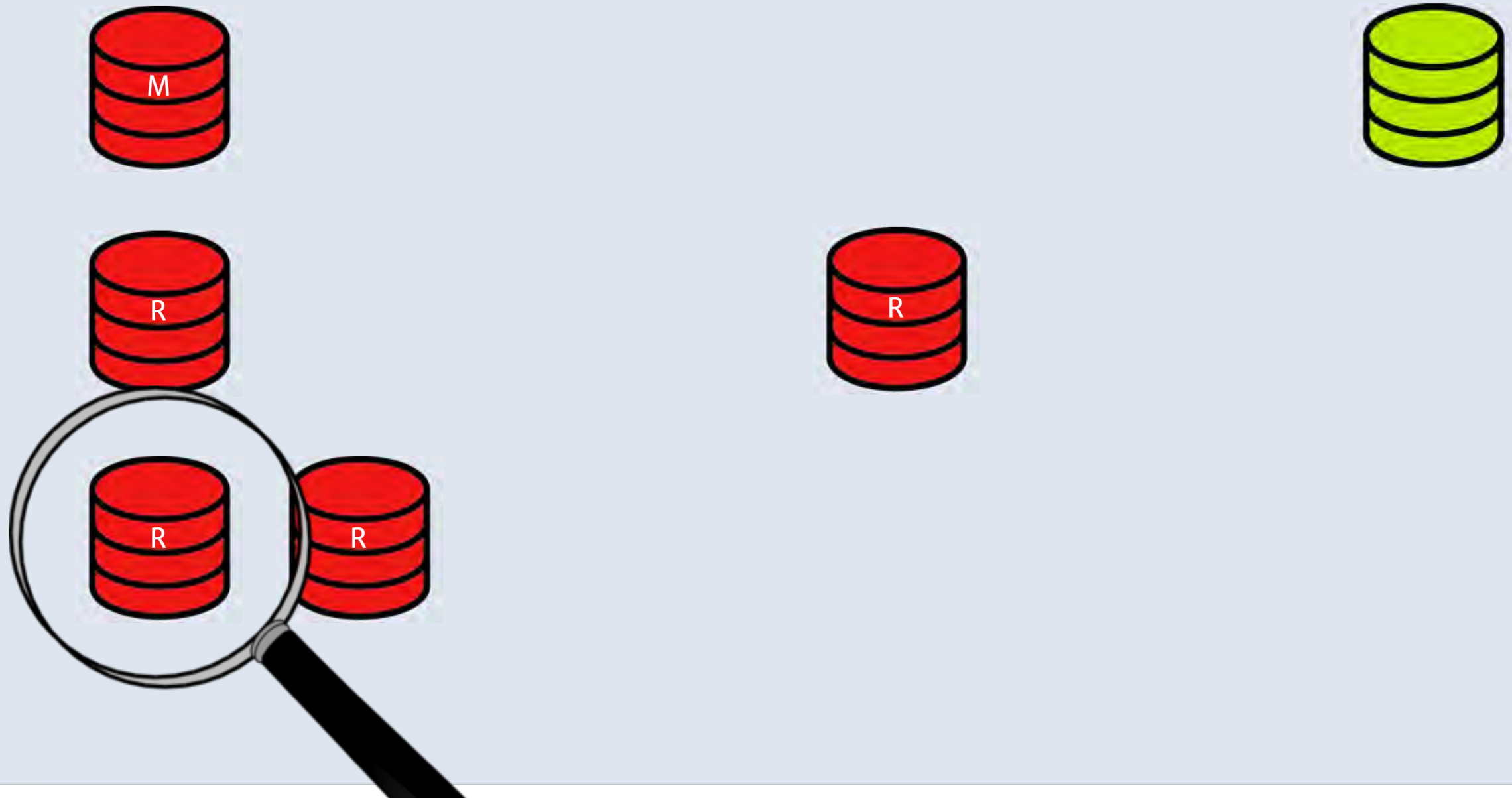
Automatic OSC

Replicaset-Level Coordination



Automatic OSC

Replicaset-Level Coordination



Automatic OSC

Replicaset-Level Coordination

Automatic OSC

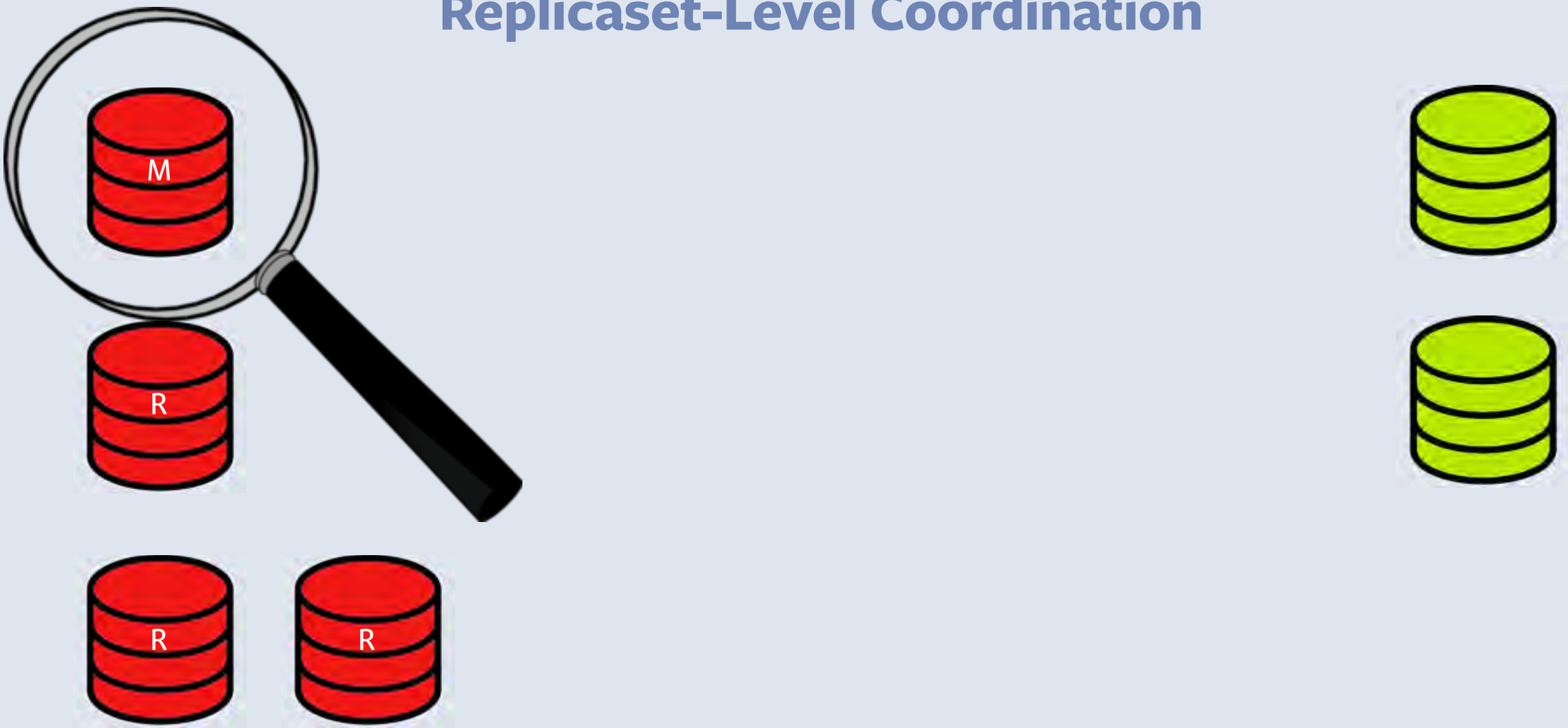
Replicaset-Level Coordination

Automatic OSC

Replicaset-Level Coordination

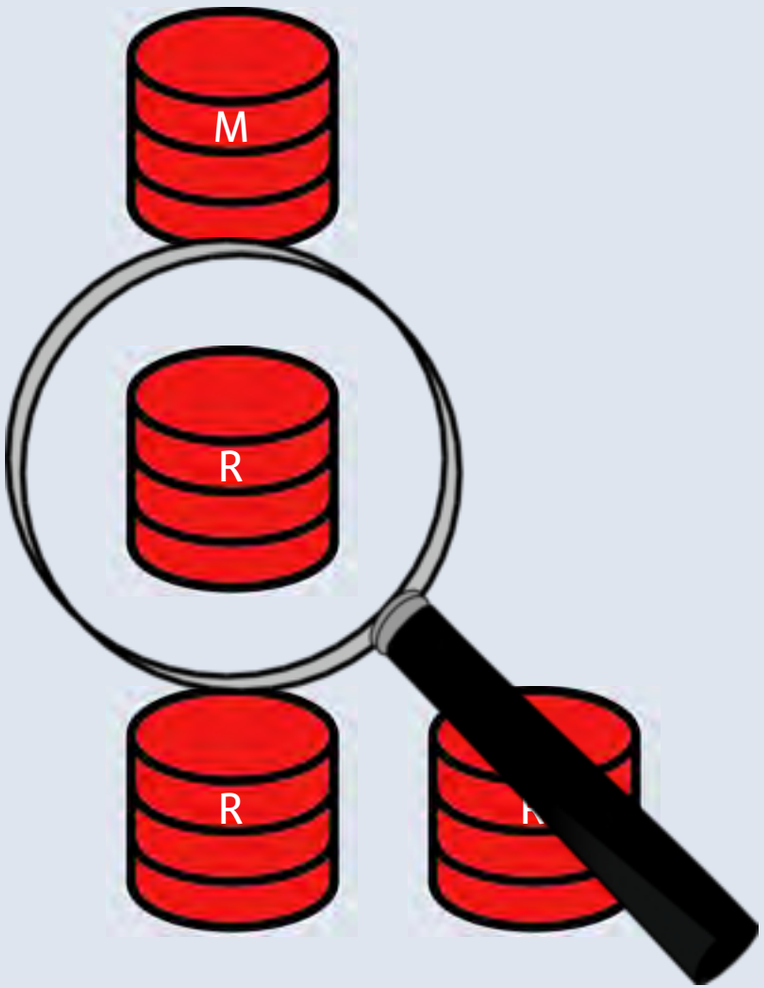
Automatic OSC

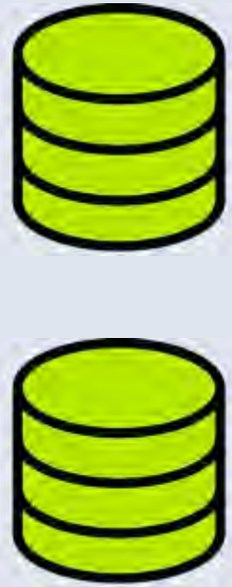
Replicaset-Level Coordination



Automatic OSC

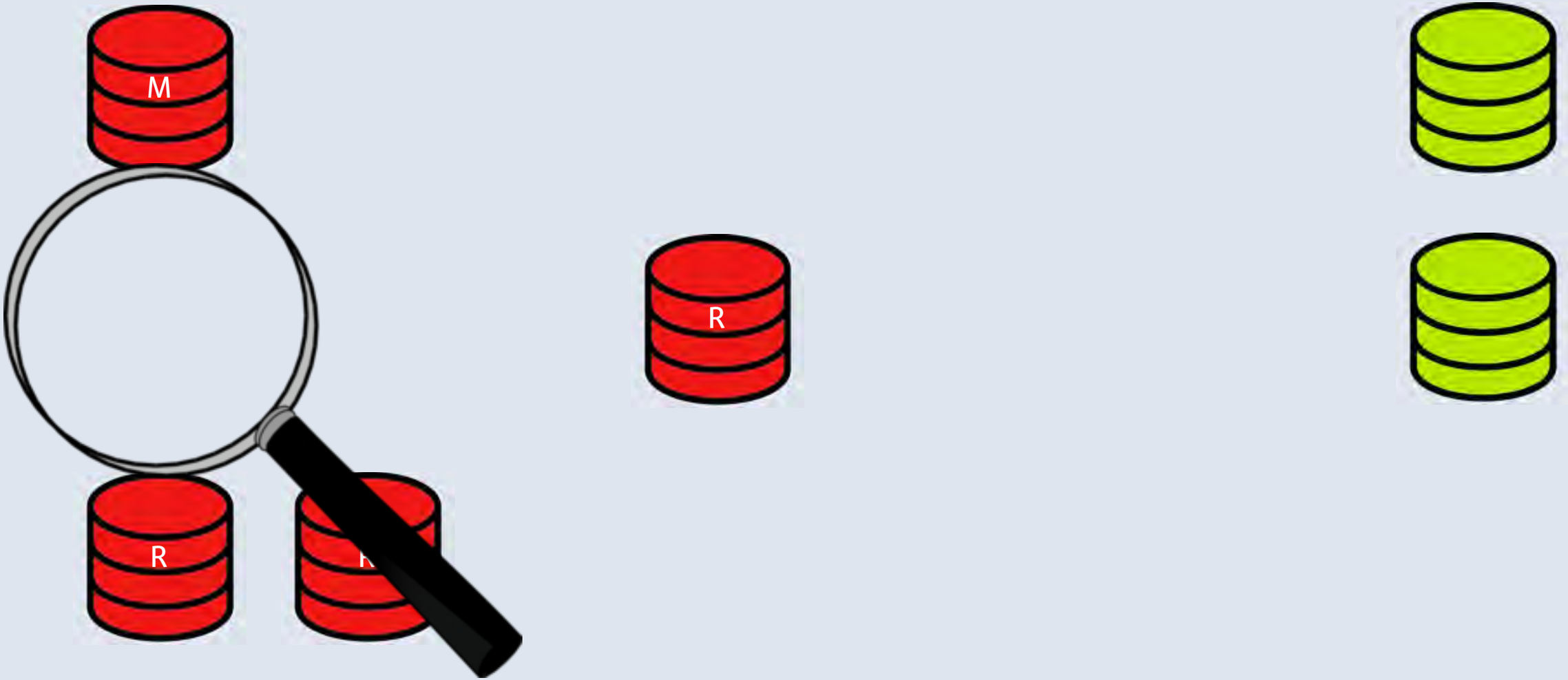
Replicaset-Level Coordination





Automatic OSC

Replicaset-Level Coordination



Automatic OSC

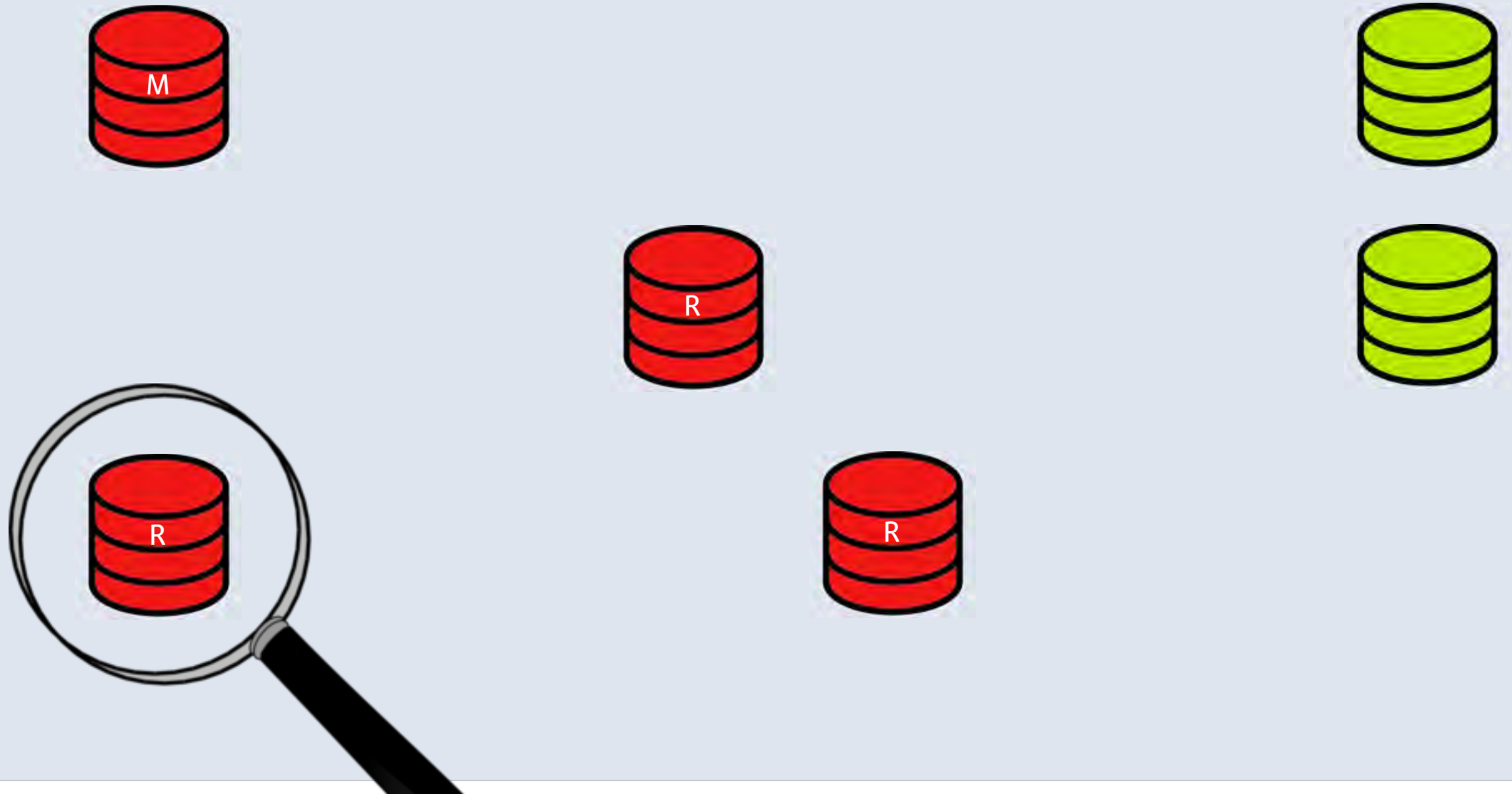
Replicaset-Level Coordination

Automatic OSC

Replicaset-Level Coordination

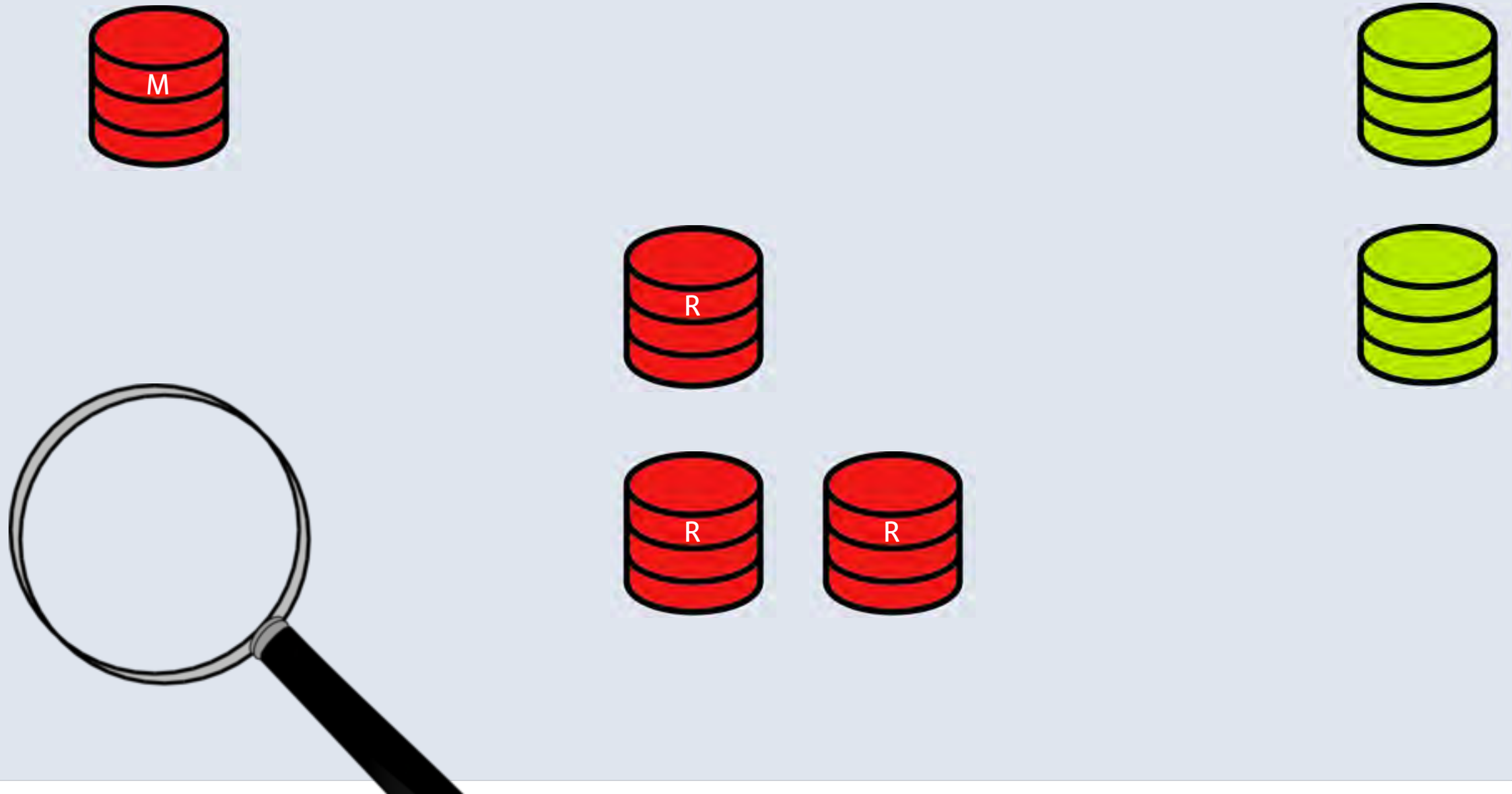
Automatic OSC

Replicaset-Level Coordination



Automatic OSC

Replicaset-Level Coordination



Automatic OSC

Replicaset-Level Coordination

Automatic OSC

Replicaset-Level Coordination

Automatic OSC

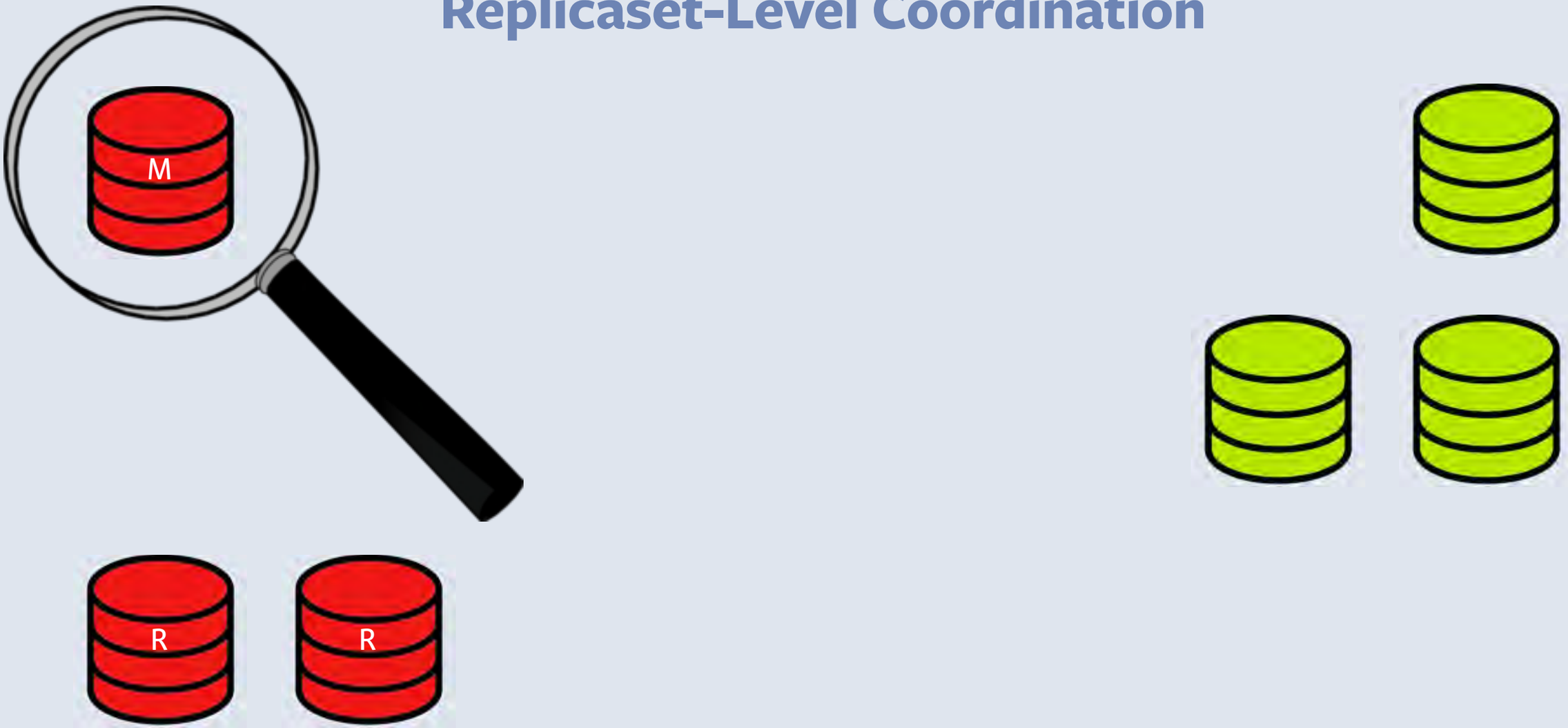
Replicaset-Level Coordination

Automatic OSC

Replicaset-Level Coordination

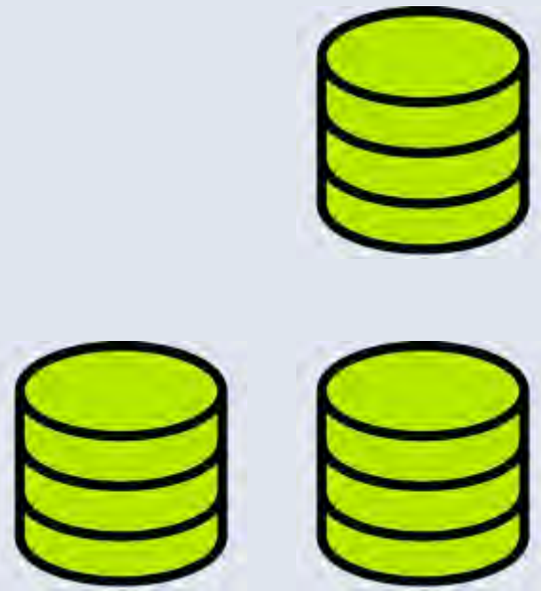
Automatic OSC

Replicaset-Level Coordination



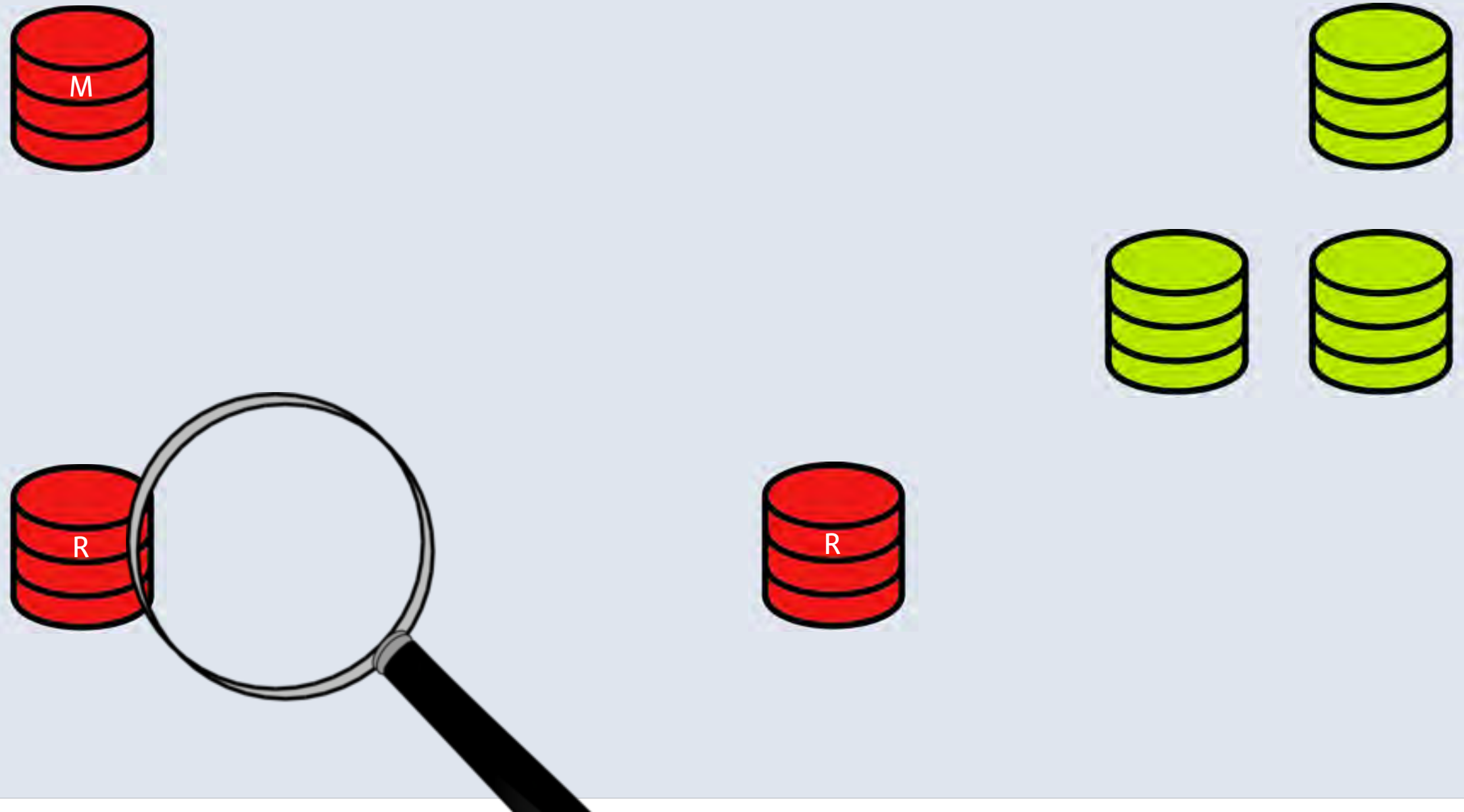
Automatic OSC

Replicaset-Level Coordination



Automatic OSC

Replicaset-Level Coordination



Automatic OSC

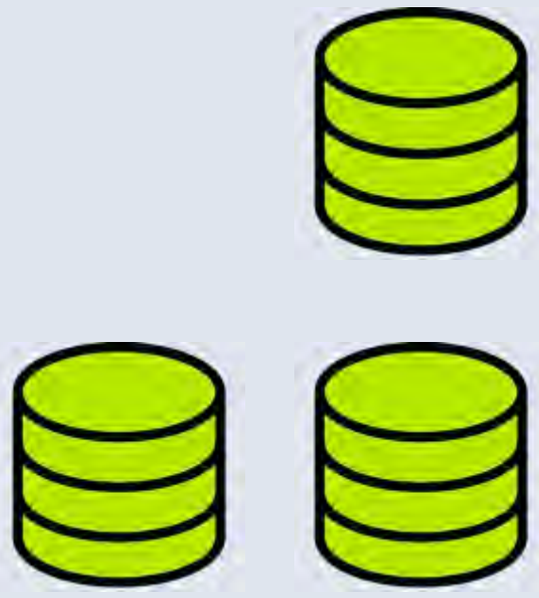
Replicaset-Level Coordination

Automatic OSC

Replicaset-Level Coordination

Automatic OSC

Replicaset-Level Coordination



Automatic OSC

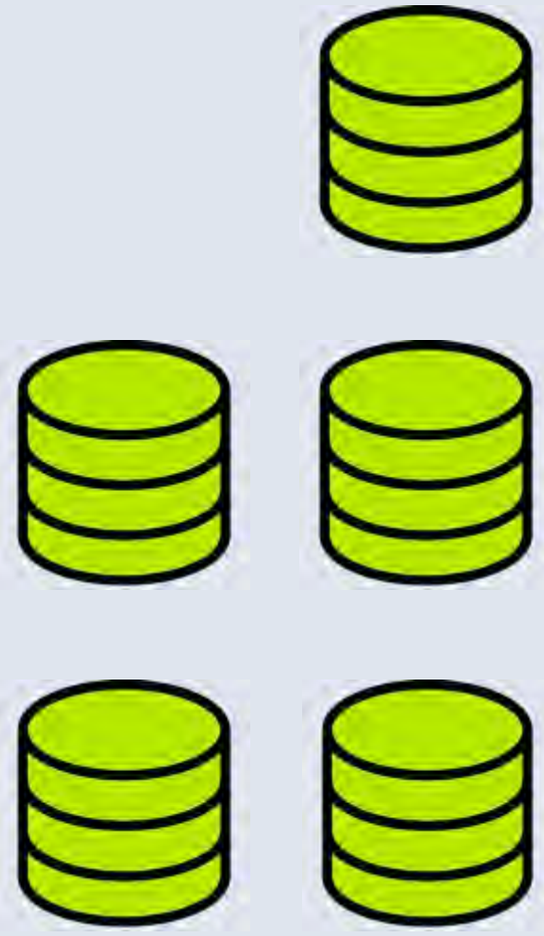
Host-Level Coordination

Automatic OSC

Host-Level Coordination

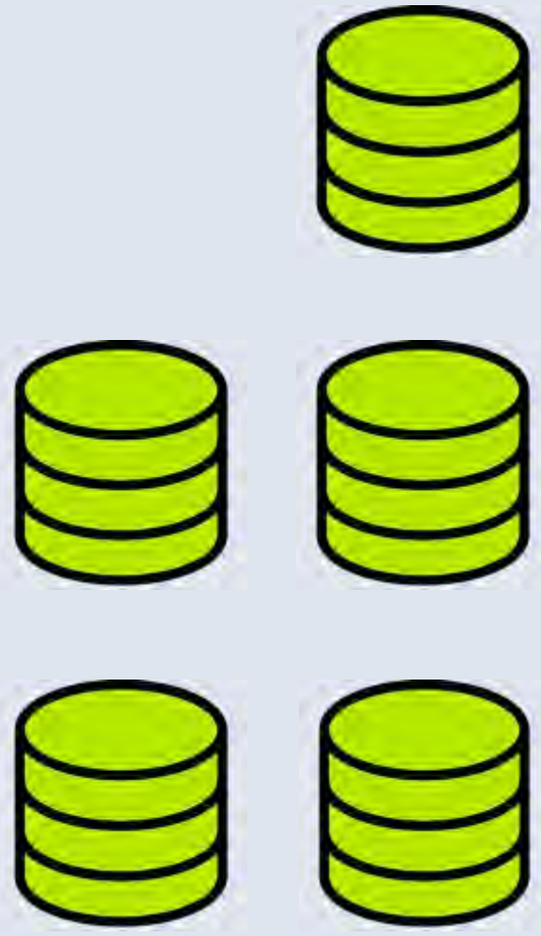
Automatic OSC

Host-Level Coordination



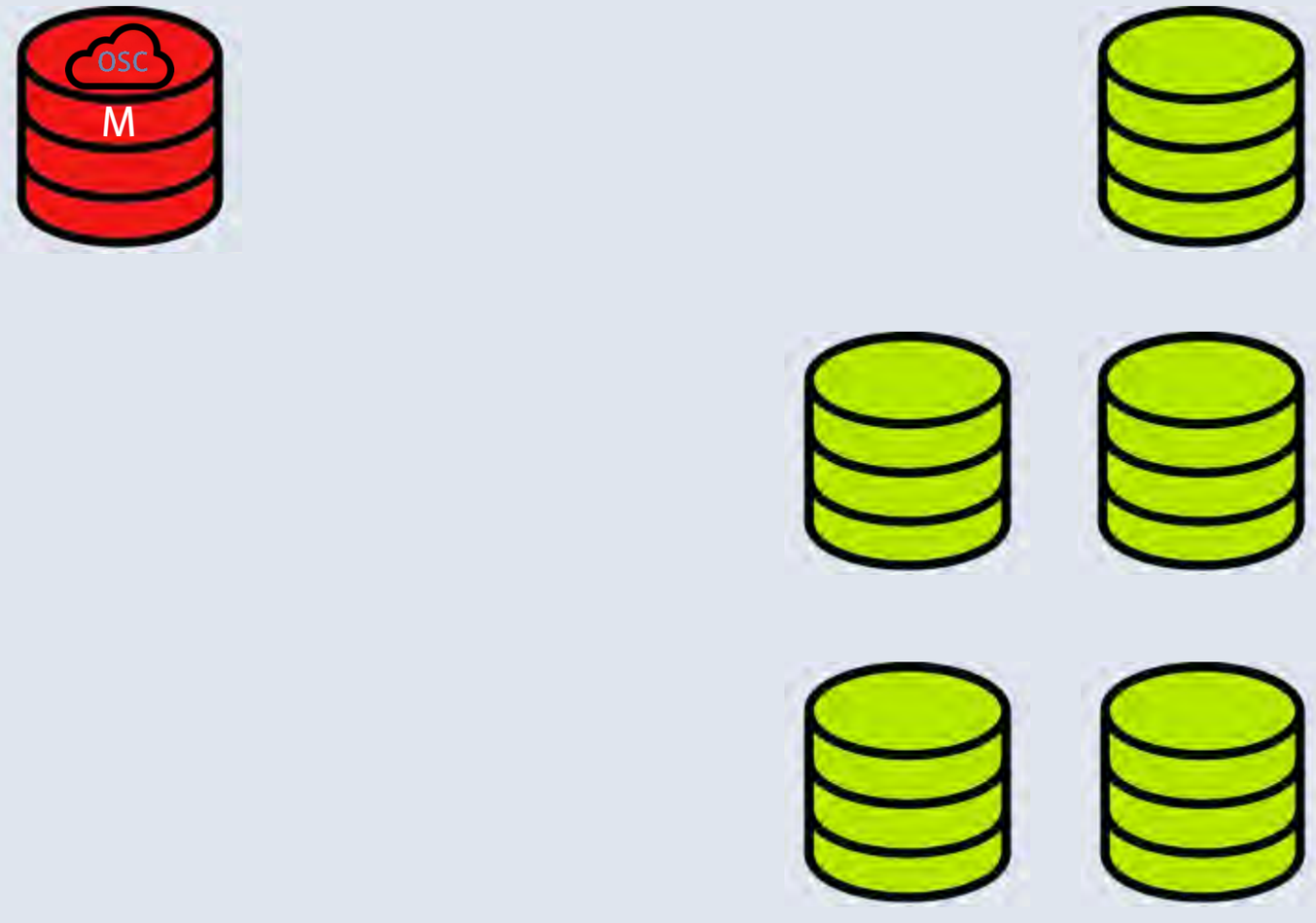
Automatic OSC

Host-Level Coordination



Automatic OSC

Host-Level Coordination

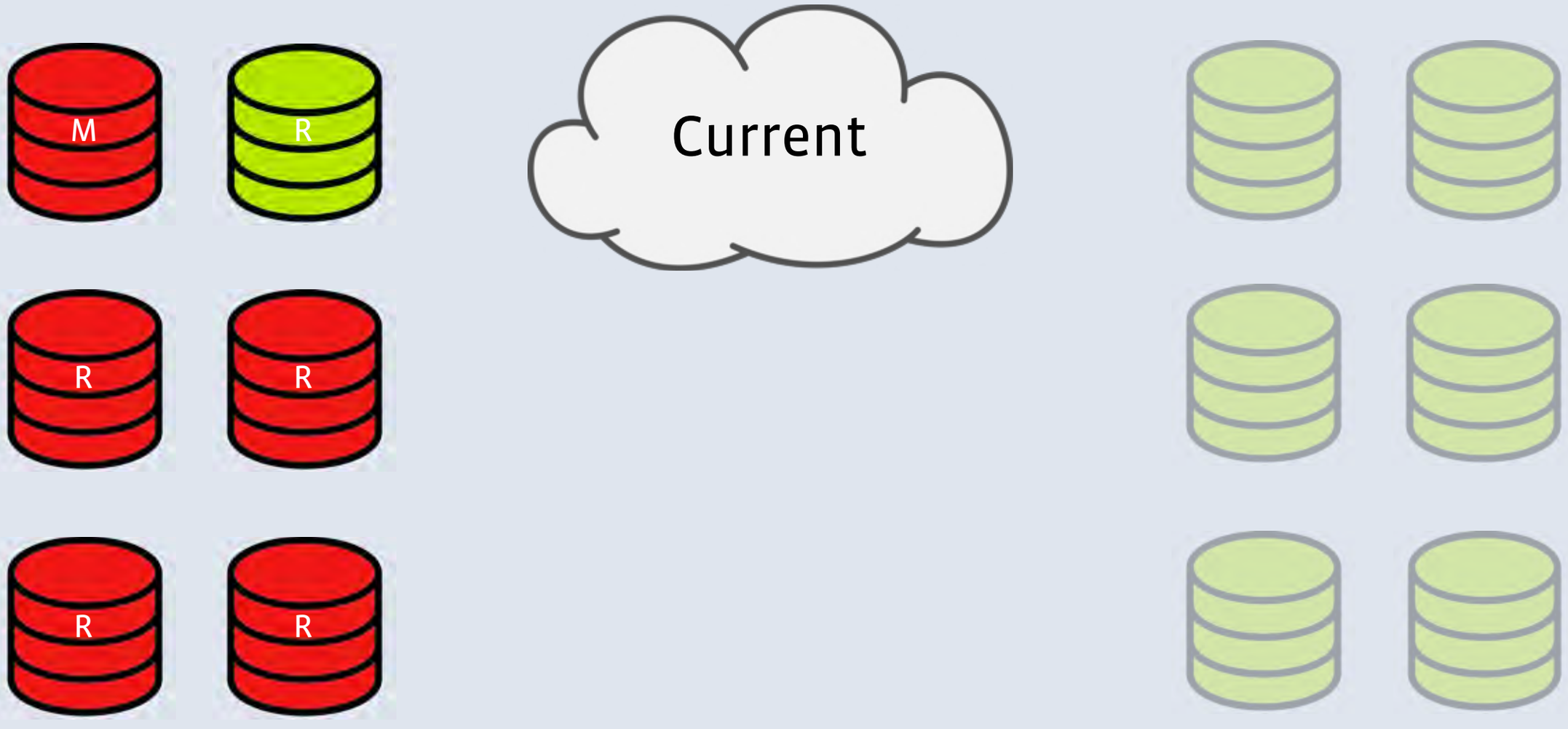


Automatic OSC

Host-Level Coordination

Automatic OSC

Host-Level Coordination



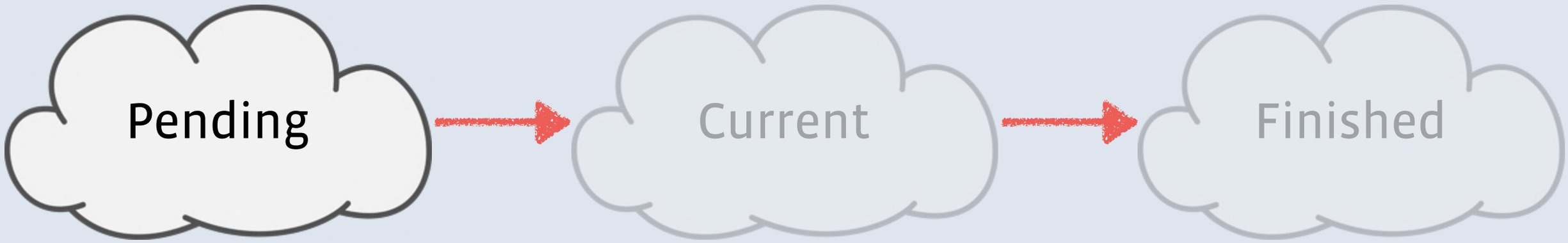
Automatic OSC

Host-Level Coordination

From Current to Finish

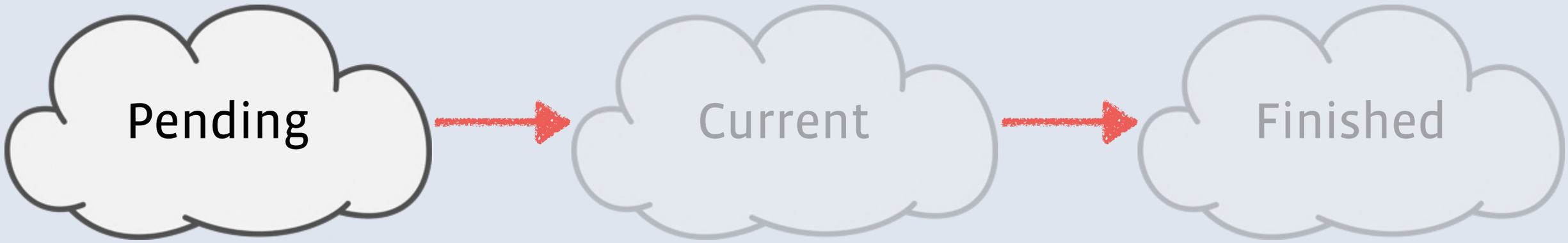
Automatic OSC

Alter-Level Coordination: Finished



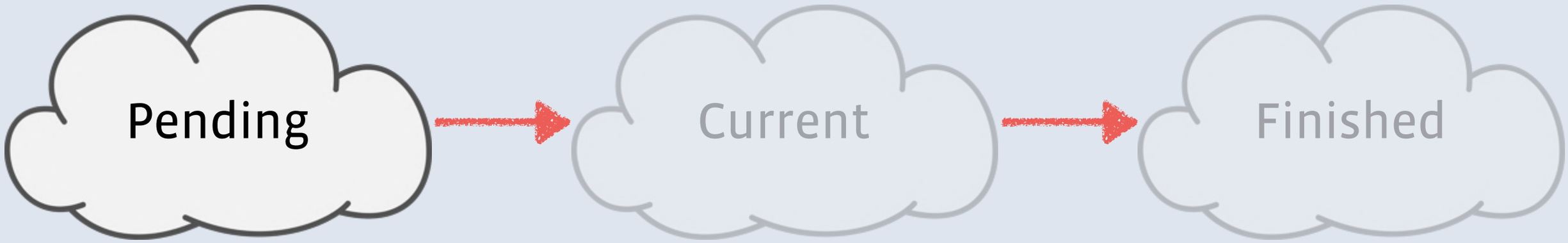
Automatic OSC

Alter-Level Coordination: Finished



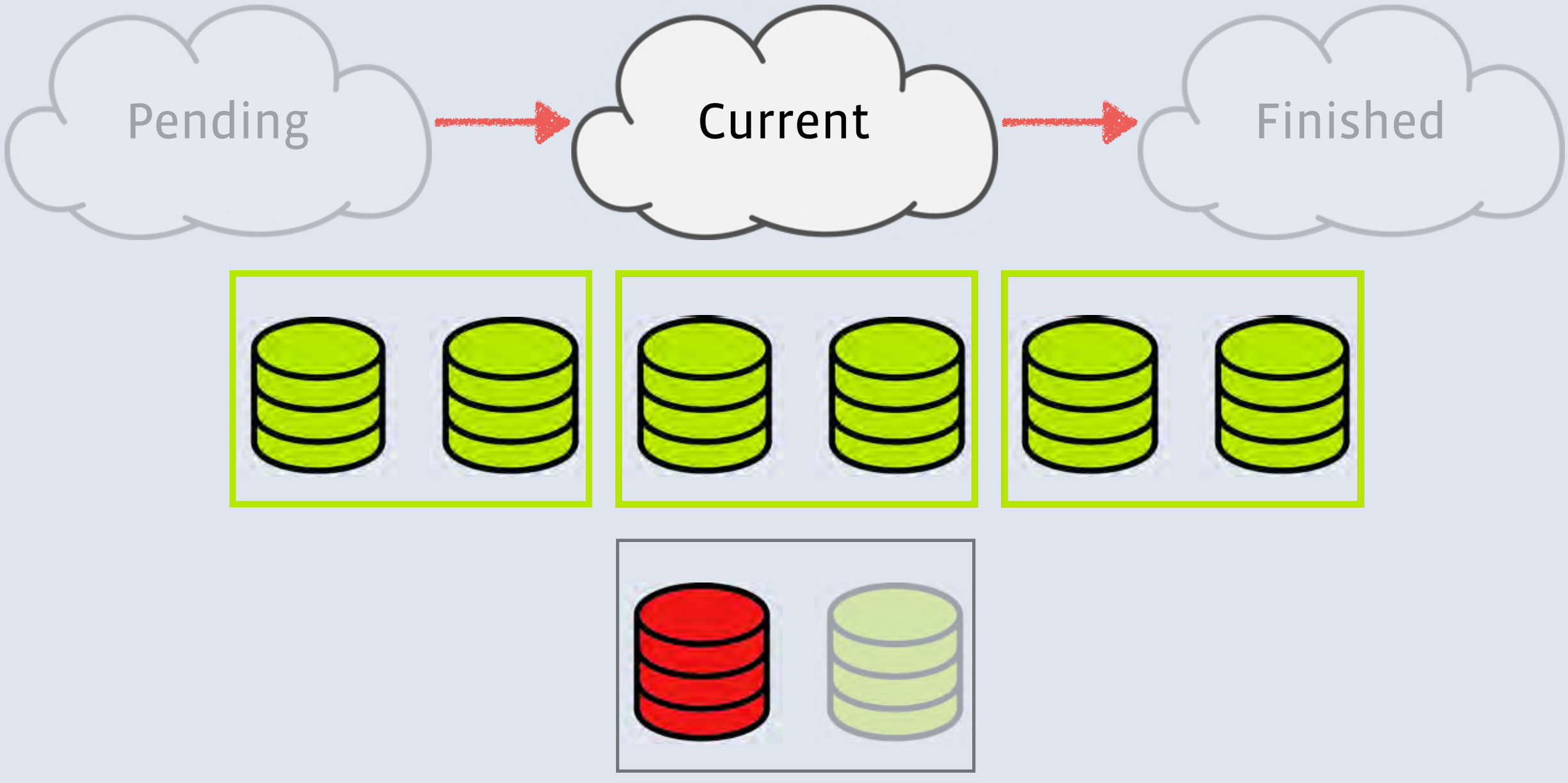
Automatic OSC

Alter-Level Coordination: Finished



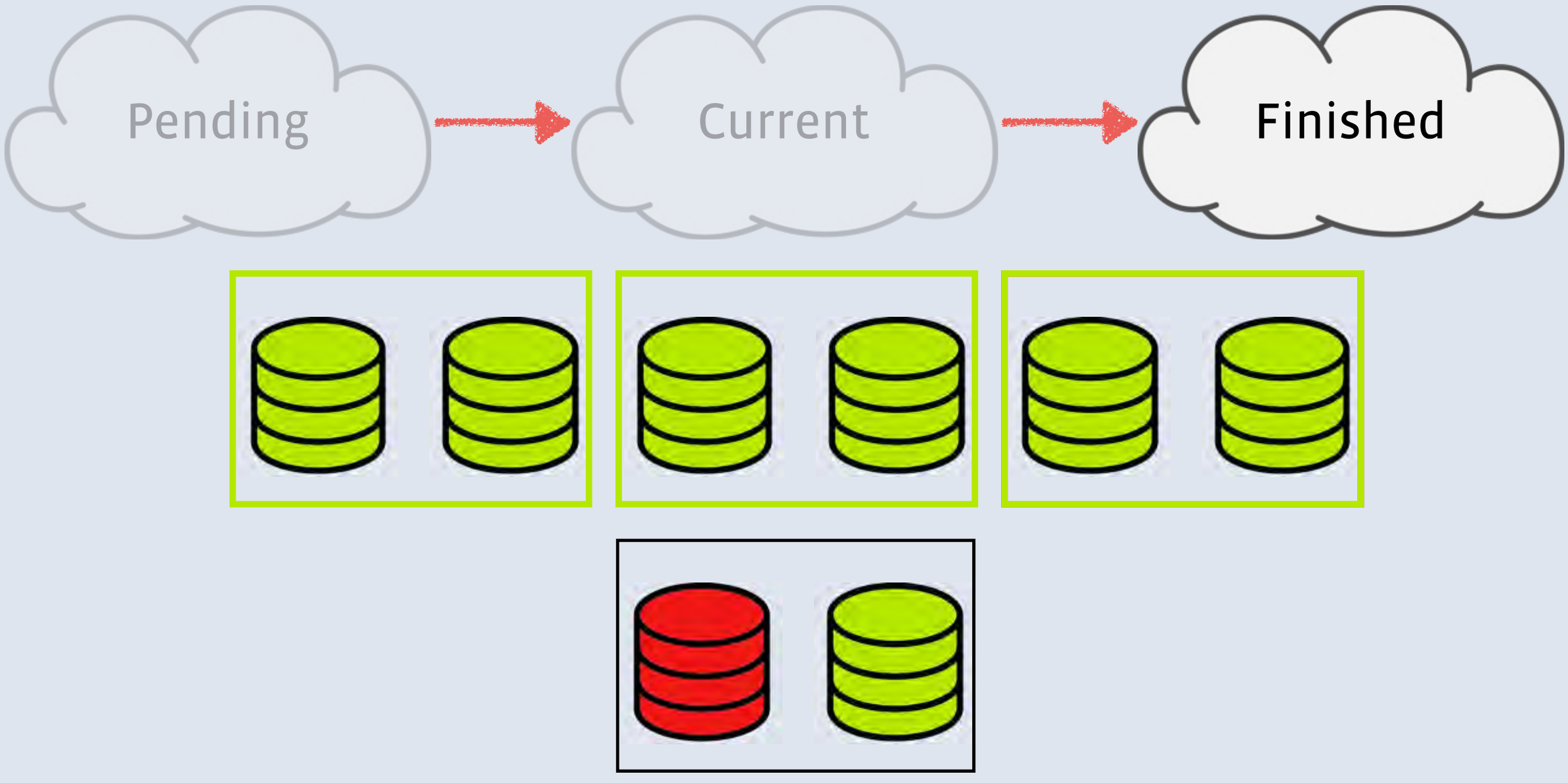
Automatic OSC

Alter-Level Coordination: Finished



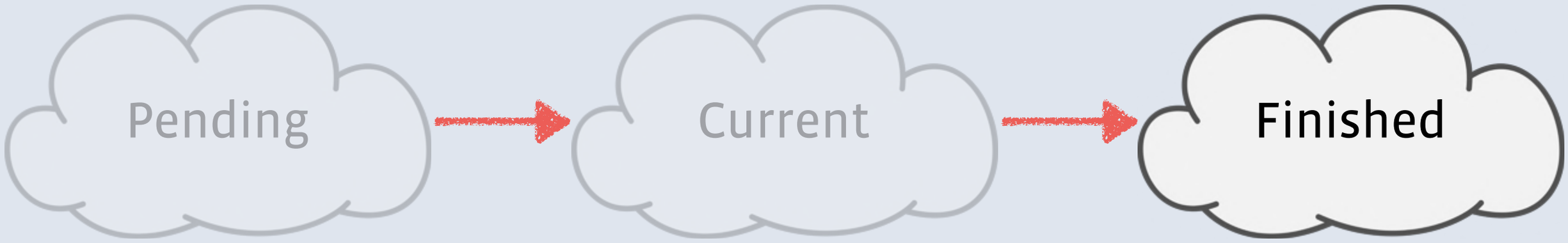
Automatic OSC

Alter-Level Coordination: Finished



Automatic OSC

Alter-Level Coordination: Finished

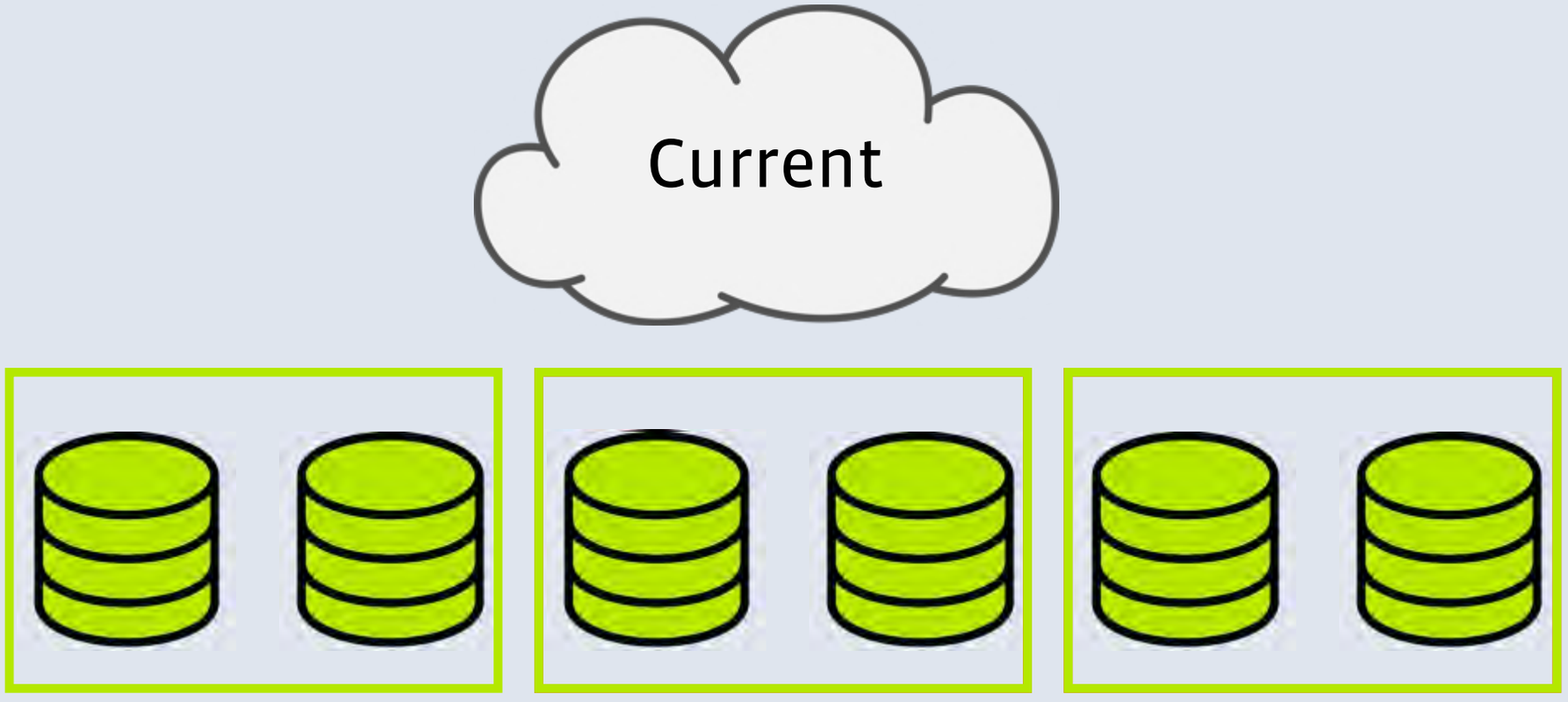


Automatic OSC

Alter-Level Coordination: Finished

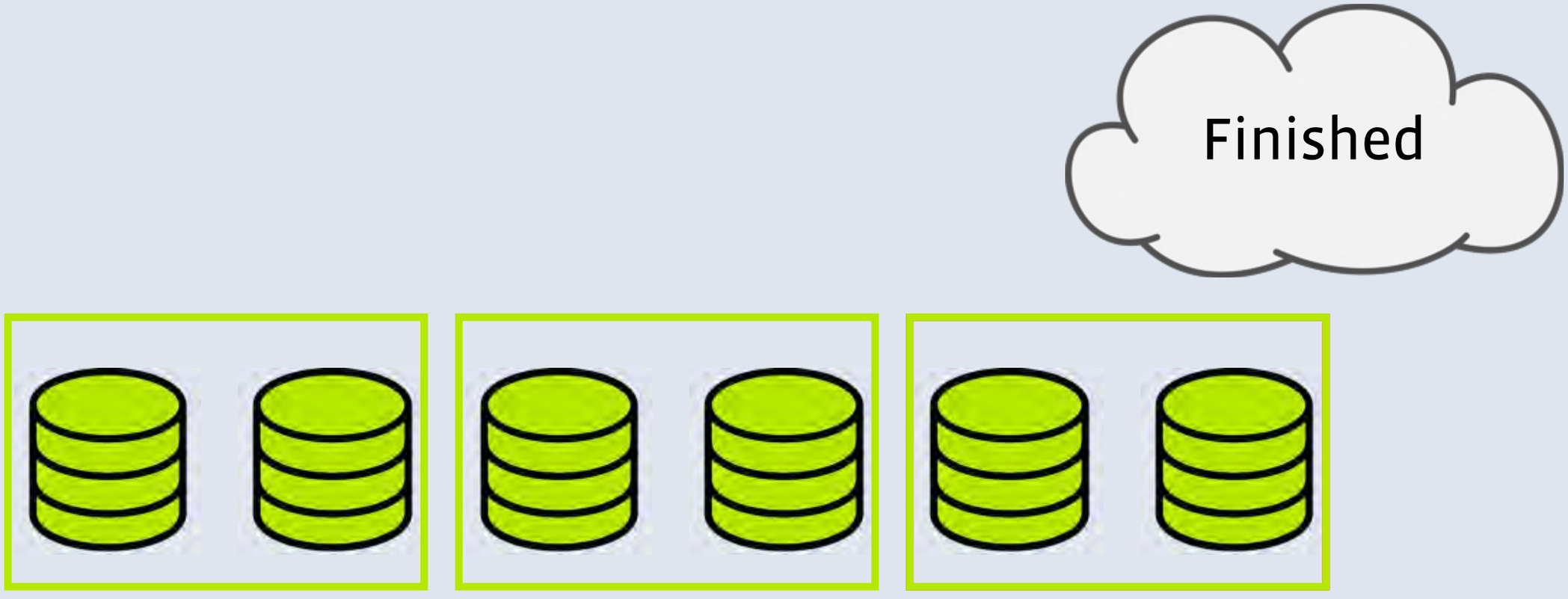
Automatic OSC

Alter-Level Coordination: Current



Automatic OSC

Alter-Level Coordination: Finished



Future Work

- Coordination between AOSC and code release system
- ETA based on regression formula

Q & A

https://github.com/facebookincubator/OnlineSchemaChange

facebook