Database services on AWS

AWS Israel Community

- Founded - Feb **2013**
- **85** meetups with **~6800** Members
- Monthly meetups
- No Marketing, No bullshit
- All AWS: AI, BigData, Serverless, Containers, etc

MEET THE TEAM

Shimon Tolts

Arthur Schmunk

Tal Hibner

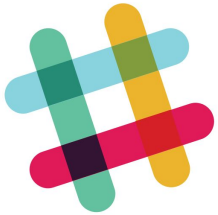
Niv Yungelson

Eitan Sela

Doron Rogov

Boaz Ziniman

Join the Community!



https://bit.ly/2zJJ3Fh

https://www.meetup.com/AWS-IL/

https://aws.org.il/

https://www.facebook.com/groups/IsraelAWSUserGroup/

Today’s Agenda

- **AWS Redshift Concurrency Scaling Explained**

  Adam Sharvit, Director of Engineering @ Innovid

- **Use ProxySQL to upgrade your Aurora Cluster without downtime**

  Michael Greenshtein, Devops Engineer @ Cloudinary

INNOVID

Thank you

Redshift Concurrency Scaling Explained

May 27th, 2019

Adam Sharvit, Director of Engineering
INNOVID

ADAM SHARVIT

- Proud father & husband
- B.Sc. CS Ext. TAU
- R&D Leader since 2011
- Joined Innovid @ 2016
- Passionate about backend
- Wish I could ride my bike more

INNOVID

AGENDA

1 Concurrency Scaling Explained

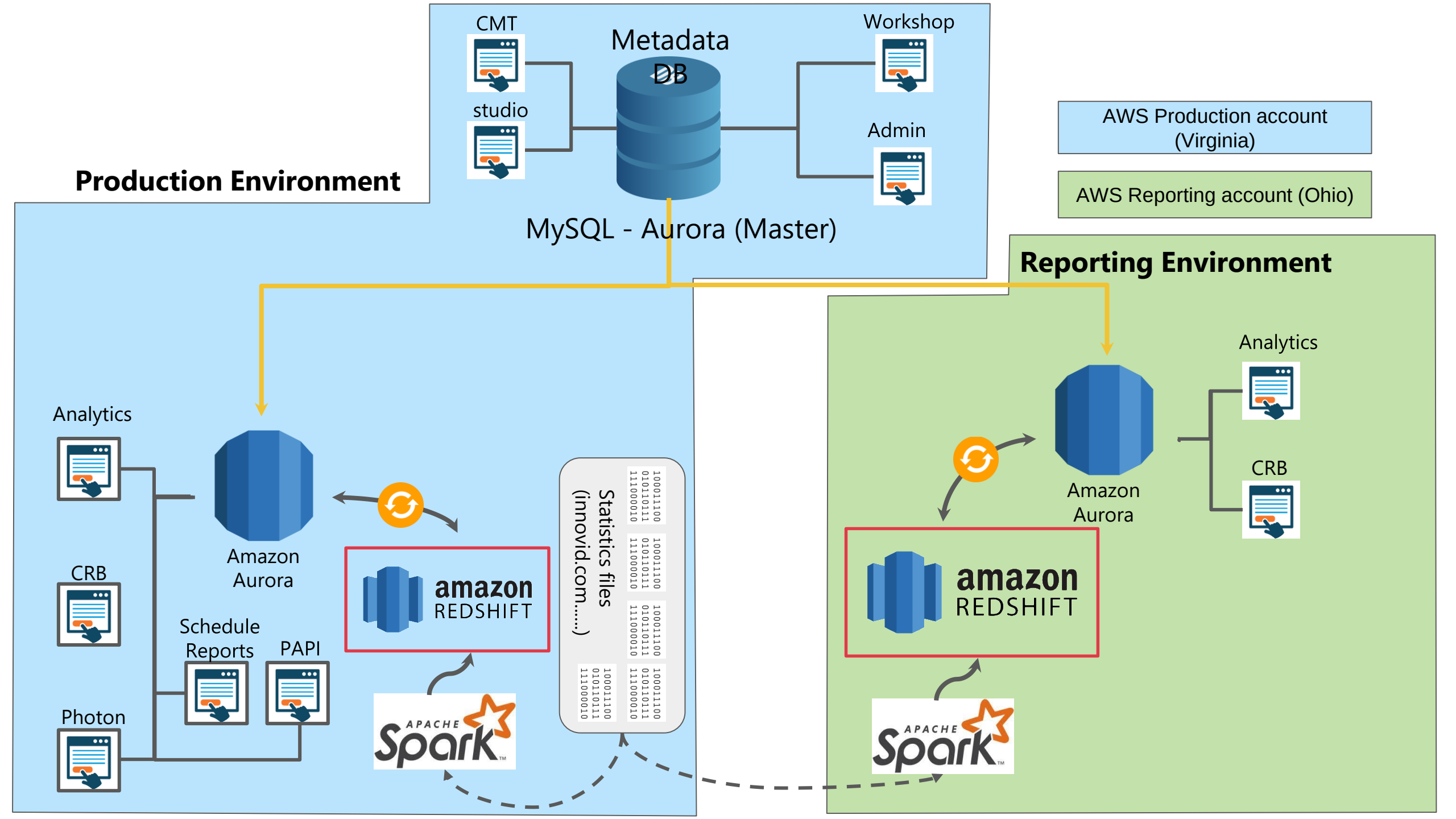
2 Testing

3 Pricing

4 Summary & Conclusions

INNOVID

- We process more than **5 billion** events per day.
- Basic processing using ***Spark EMR*** cluster
- Business logic layer is implemented on ***Amazon Redshift***, which aggregates the raw data and slices it into various forms of information.


CMT
studio
Metadata DB
Workshop
Admin
MySQL - Aurora (Master)
AWS Production account (Virginia)
AWS Reporting account (Ohio)
Production Environment
Reporting Environment
Analytics
CRB
Photon
Schedule Reports
PAPI
Amazon Aurora
amazon REDSHIFT
APACHE Spark
Statistics files (innovid.com.....)
Analytics
CRB
Amazon Aurora
amazon REDSHIFT
APACHE Spark

Concurrency Scaling Explained

INNOVID

WHY SCALING OUT?

BUT ON DEMAND, PLEASE

MISSION CRITICAL ETLs	MULTIPLE BI USERS	SHORT PEEK TIMES
Assure performance	Multiple BI users and dashboards	Tuned for short burst but idle most of the time

INNOVID

ALTERNATIVES

1 Outscale (over-provision)

2 Tune for average use case

3 Seek for replacements

4 **Concurrency Scaling**

INNOVID

WHAT IS CONCURRENCY SCALING?

"With Concurrency Scaling, you can support virtually unlimited concurrent users and concurrent queries, with consistently fast query performance. When Concurrency Scaling is enabled, Amazon Redshift automatically adds additional cluster capacity when you need it to process an increase in concurrent read queries. Write operations continue as normal on your main cluster. Users always see the most current data, whether the queries run on the main cluster or on a concurrency scaling cluster."

INNOVID

WHAT IS CONCURRENCY SCALING?

“

unlimited

consistently

automatically

normal

current

”

INNOVID

WHAT IS CONCURRENCY SCALING?

Up-to-date data

Unlimited concurrent users

Throughput scales linearly

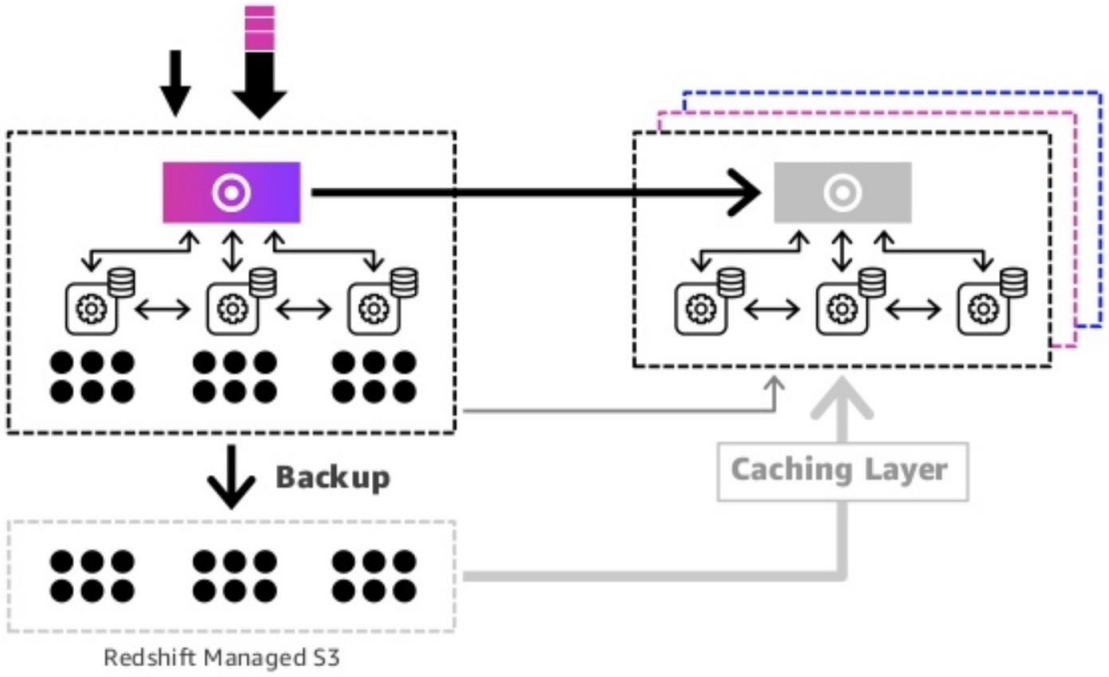
Auto scaling

Writes on master

INNOVID

HOW IS CONCURRENCY SCALING ACHIEVED?

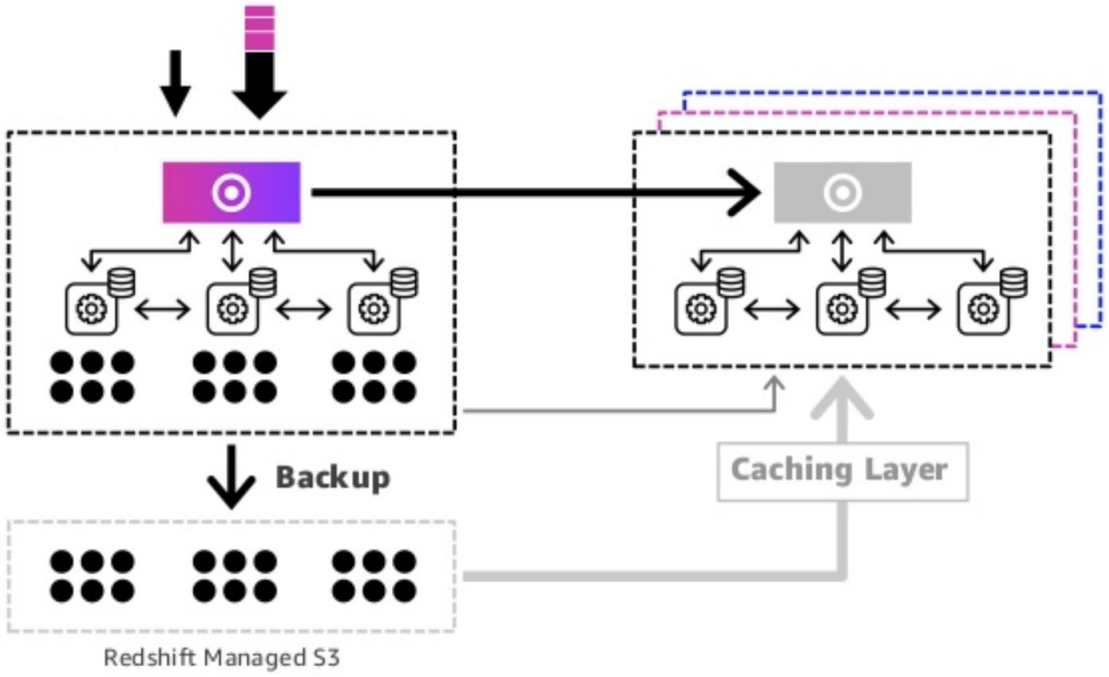
- Redshift can spin up additional ***transient clusters*** when queueing is detected in the WLM queue on the main cluster
- Users connects to the main cluster only, and RS directs the requests to the new clusters transparently



INNOVID

HOW IS CONCURRENCY SCALING ACHIEVED?

- When the queue clears, the transient clusters are released
- **No hydration** time needed. The transient clusters read the data from the Redshift snapshots combined with up to date data from the main cluster



INNOVID

CONCURRENCY SCALING PROPERTIES

ACID

ACID is maintained throughout the entire scaling process

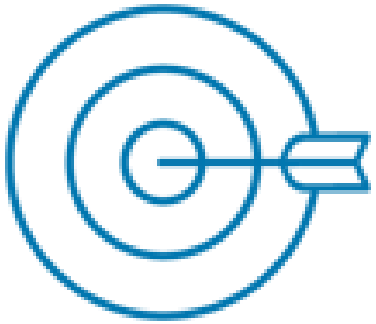
Controlled in the WLM Queue Level

Concurrency Scaling for queues is optional

Automatic Statements Detection

DDL statements will go to the main cluster automatically

Short queries to SQA queue



INNOVID

ENABLING CONCURRENCY SCALING IN THE CONSOLE

- Redshift dashboard
- Clusters
- Query editor New
- Saved queries
- Snapshots
- Security
- Parameter groups
- **Workload management**
- Reserved nodes
- Advisor
- Events
- Connect client
- What's new

Parameters | WLM

Enable **Short Query Acceleration** for queries whose maximum runtime is dynamic Learn more

Max Concurrency Scaling clusters: 1 Edit

Queue 1

Memory (%)	Concurrency on main	Concurrency Scaling mode	Timeout (ms)	User groups	Query groups
40	1	auto	0		rnf

Query Monitoring Rules (0)

No rules have been defined.

Queue 2

Memory (%)	Concurrency on main	Concurrency Scaling mode	Timeout (ms)	User groups	Query groups
10	5	auto	0	manual_queue	manual_queue_group

Query Monitoring Rules (0)

No rules have been defined.

INNOVID

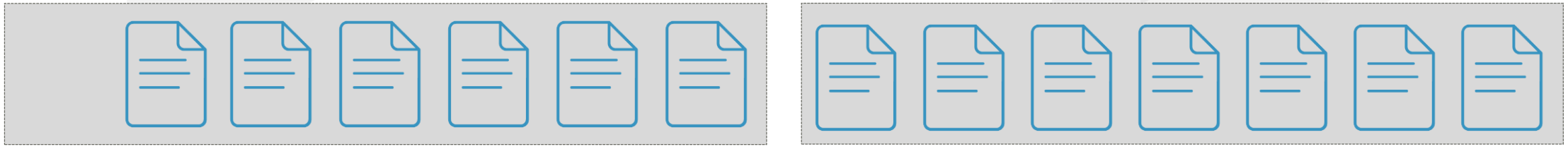
CONCURRENCY SCALING OFF

DEFAULT QUEUE

SQA QUEUE

INNOVID

CONCURRENCY SCALING ON



DEFAULT QUEUE

SQA QUEUE

INNOVID

CONCURRENCY SCALING ON

REPORTS QUEUE

ETL QUEUE

SQA QUEUE

INNOVID

ELASTIC RESIZE VS. CONCURRENCY SCALING

	Elastic Resize	Concurrency Scaling
Time		
Enablement		
Resize Option		
Granularity		
Hydration		
Queries Constraints		

INNOVID

Testing

INNOVID

TESTING VARIANTS

1. **Concurrency Scaling On/Off**
2. **Queue Concurrency**
3. **SQA**
4. **Dedicated Queues**

INNOVID

TESTING INVARIANTS

1. **Query Types**
2. **Test Machine (Client)**
3. **Dedicated Cluster**
4. **Data**

INNOVID

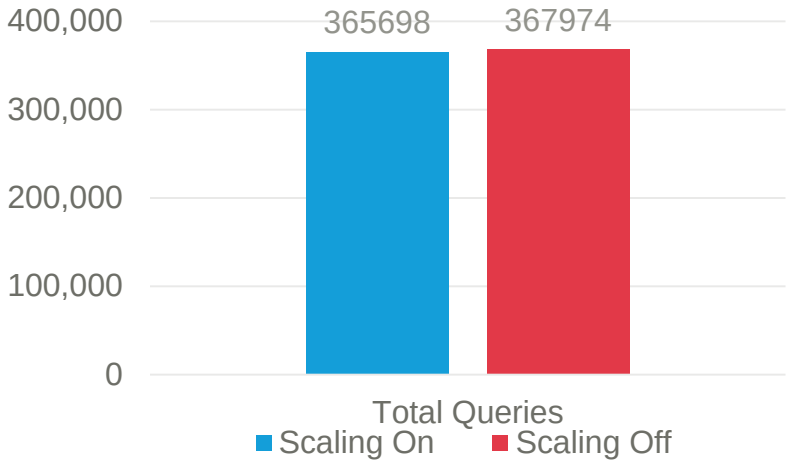
SHORT QUERIES TEST

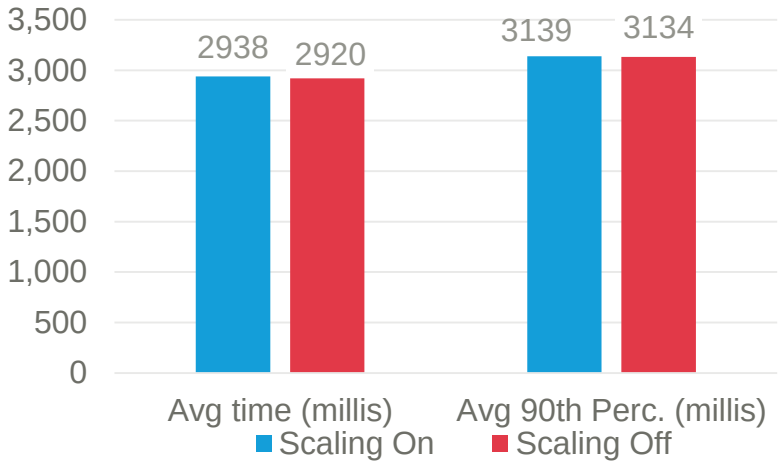
TEST SETUP:

- 300 concurrent queries
- SQA = Dynamic
- Queue Concurrency = 15

TEST HIGHLIGHTS:

- Metrics shows virtually the same throughput regardless of Concurrency Scaling
- Different ***SQA / Queue Concurrency*** values do not help to increase the throughput
- **Short queries will always go to SQA queue**





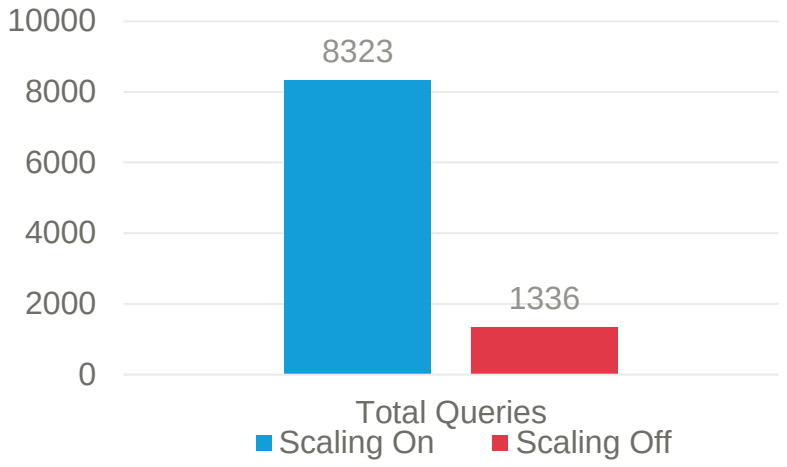
MEDIUM QUERIES TEST

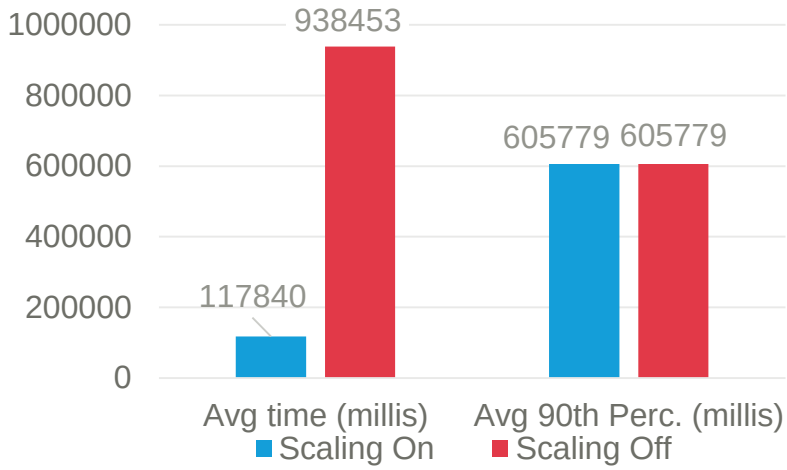
TEST SETUP:

- 300 concurrent queries
- SQA = Off
- Queue Concurrency = 35

TEST HIGHLIGHTS:

- **622% improvement in throughput!**
- **796% improvement in average running time**





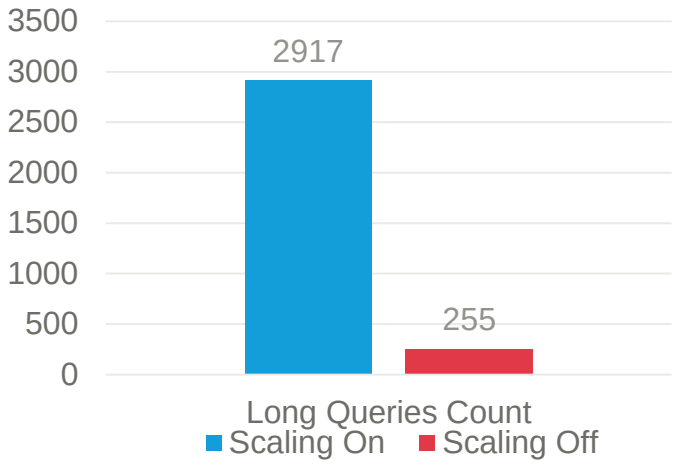
MIX OF LONG & SHORT QUERIES TEST

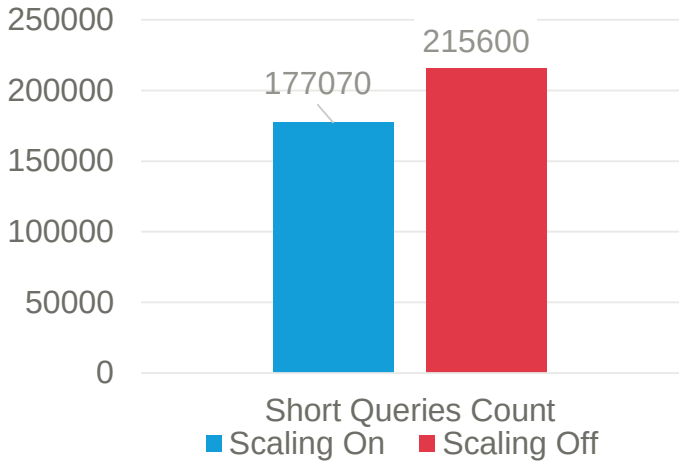
TEST SETUP:

- 300 concurrent queries
  - 200 long
  - 100 short
- SQA = Dynamic
- Queue Concurrency = 5

TEST HIGHLIGHTS:

- **1143% improvements in throughput for long queries!**
- **18% degradation of performance in short queries**
- **Short queries throughput still high**



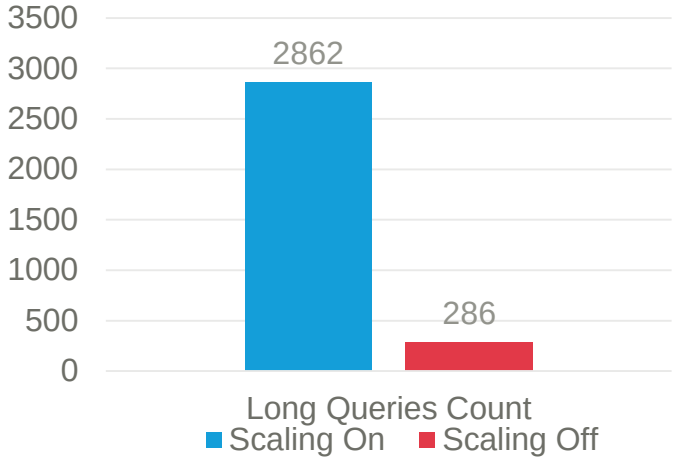


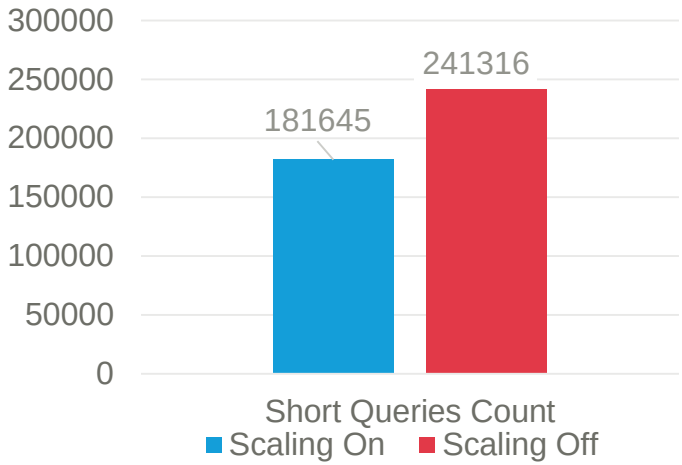
MIX OF LONG & SHORT QUERIES

WITH A DEDICATED QUEUE FOR SHORT QUERIES AND LONG QUERIES

TEST SETUP:

- Same parameters as previous
- Dedicated queue for short queries with Concurrency = Off
- Long queries queue concurrency = 15





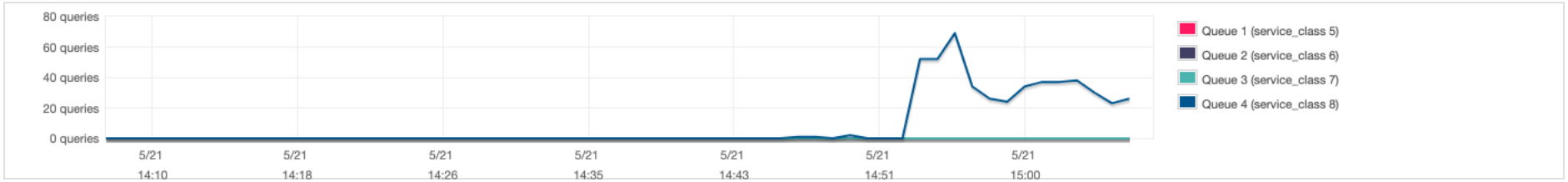
TEST HIGHLIGHTS:

- **X10** improvement for long queries is maintained
- Dedicated queue for short queries does not help to increase the short queries performance

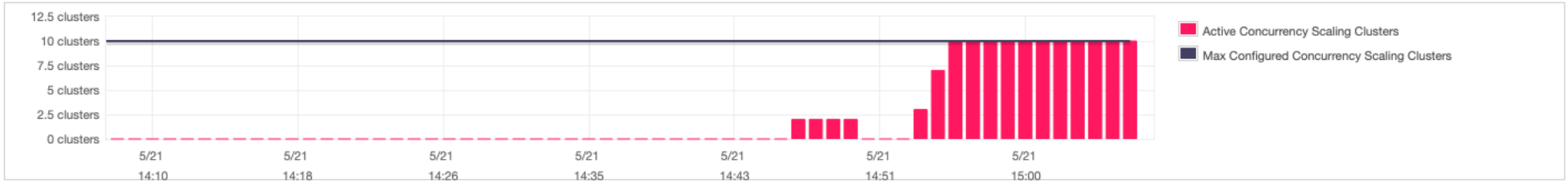
INNOVID

DATABASE PERFORMACE METRICS

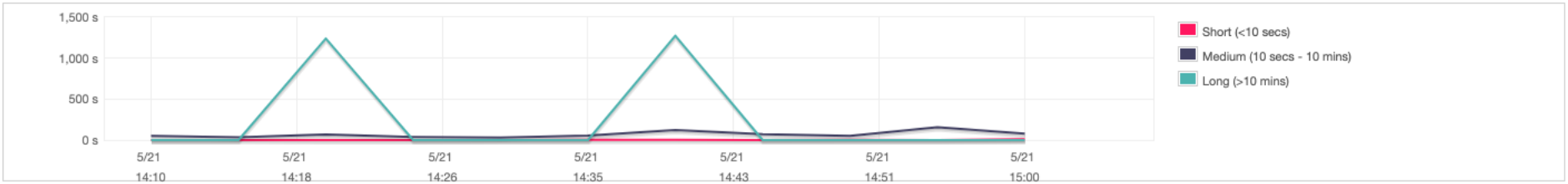
Queue Length by WLM Queues



Concurrency Scaling Activity



Query Duration



Short, medium, and long time ranges are based on the total time it takes to process a query.

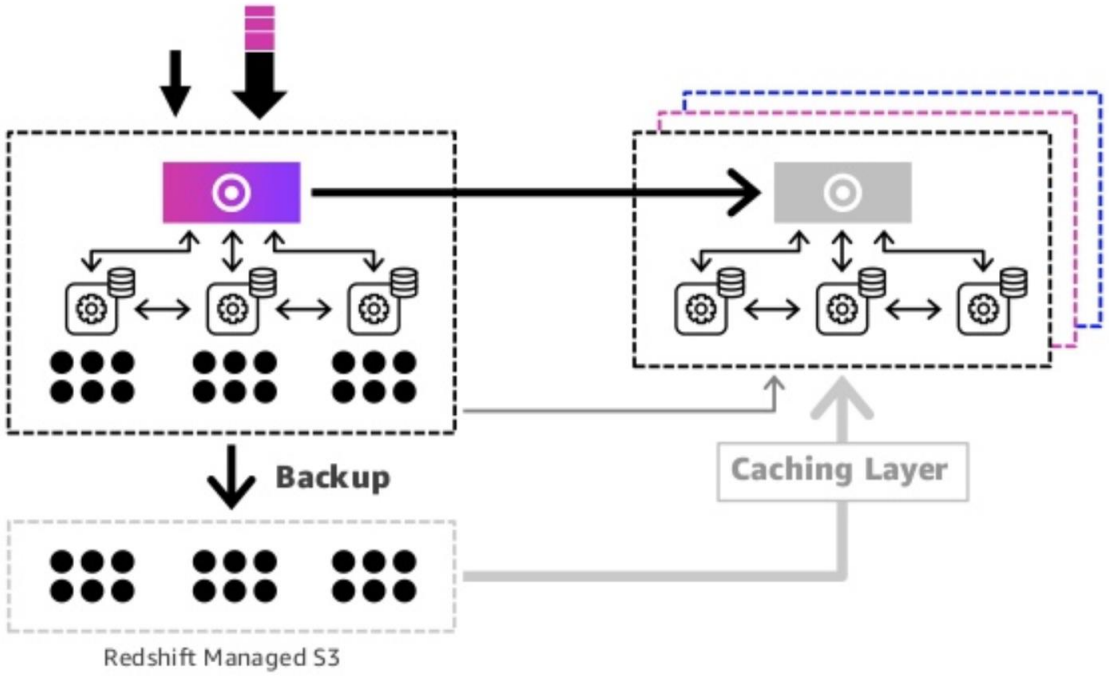
INNOVID

Pricing

INNOVID

CONCURRENCY SCALING PRICING

- Every 24 hours of up-time, the cluster gets credited for 1 free hour of Concurrency Scaling credit
- Hence, Concurrency Scaling should be free for 97% of Redshift users (based on usage analysis)
- Can accumulate up to 30 hours of credit per cluster



INNOVID

Summary & Conclusions

SUMMARY & CONCLUSIONS

- Great addition to Redshift
- WLM queue level
- SQA is on regardless of checkbox
- Works best for medium/long queries
- Free for 97% of Redshift clients
- Experiment to see if it works for YOU!

INNOVID

THANK YOU

adam.sharvit@innovid.com
www.linkedin.com/in/adam-sharvit

INNOVID

Cloudinary

Use ProxySQL to upgrade your Aurora cluster without downtime

Michael Greenshtein

DEVOPS

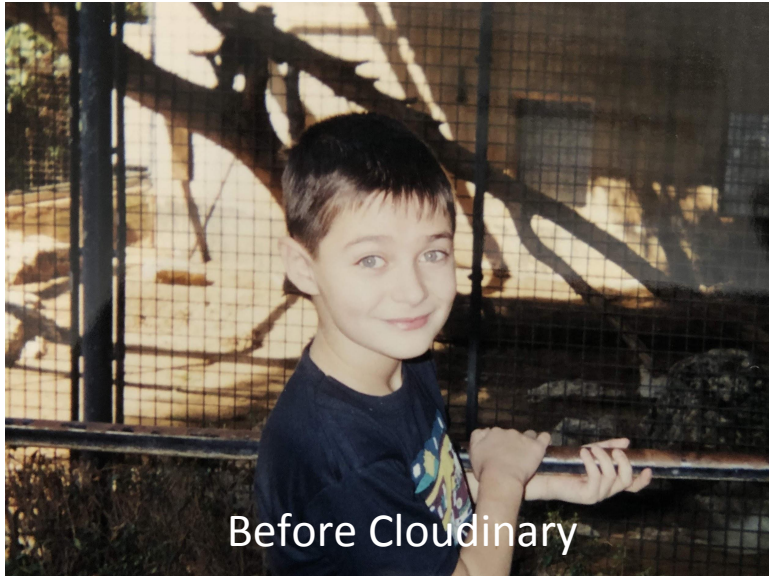
Michael Greenshtein

Almost 2 years at Cloudinary.

Fullstack and Backend developer in the past.

Getting married next week.

https://www.linkedin.com/in/michael-greenshtein-19ba9856/



Cloudinary

The Media Full Stack

Heavy-duty image & video platform

What is AWS Aurora

Faster than MySQL

Cross Region Read Replicas

Automatic backups

Failover mechanism

Cloudwatch integration

Cloudinary

Why upgrade Aurora?

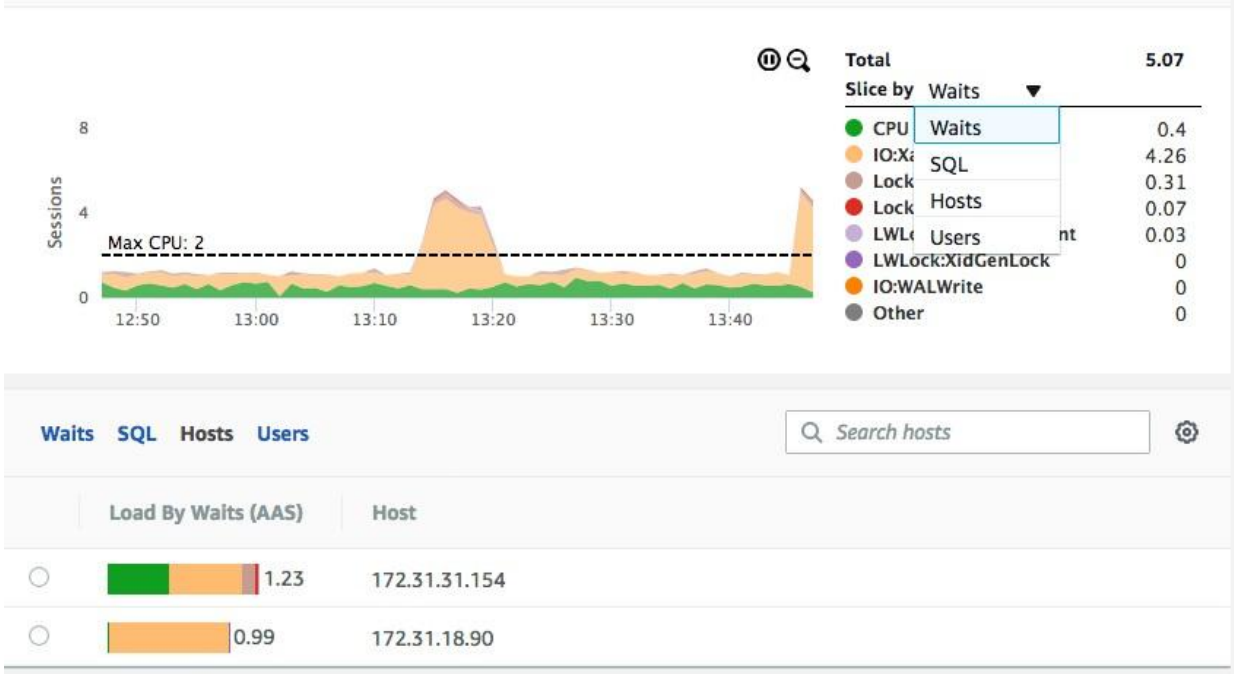
Security patches

Bug fixes

New instance types support

New features

Performance insights



Cloudinary

Upgrade = Downtime

Updates are applied to all instances in a DB cluster at the same time. An **update requires a database restart on all instances** in a DB cluster, so you experience 20 to 30 seconds of downtime, after which you can resume using your DB cluster or clusters. You can view or change your maintenance window settings from the AWS Management Console.

https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/AuroraMySQL.Updates.html#AuroraMySQL.Updates.Patching

Cloudinary

OUR SOLUTION

Use Clone to create new cluster with latest version

Manually enable 1 way replication between new and old cluster

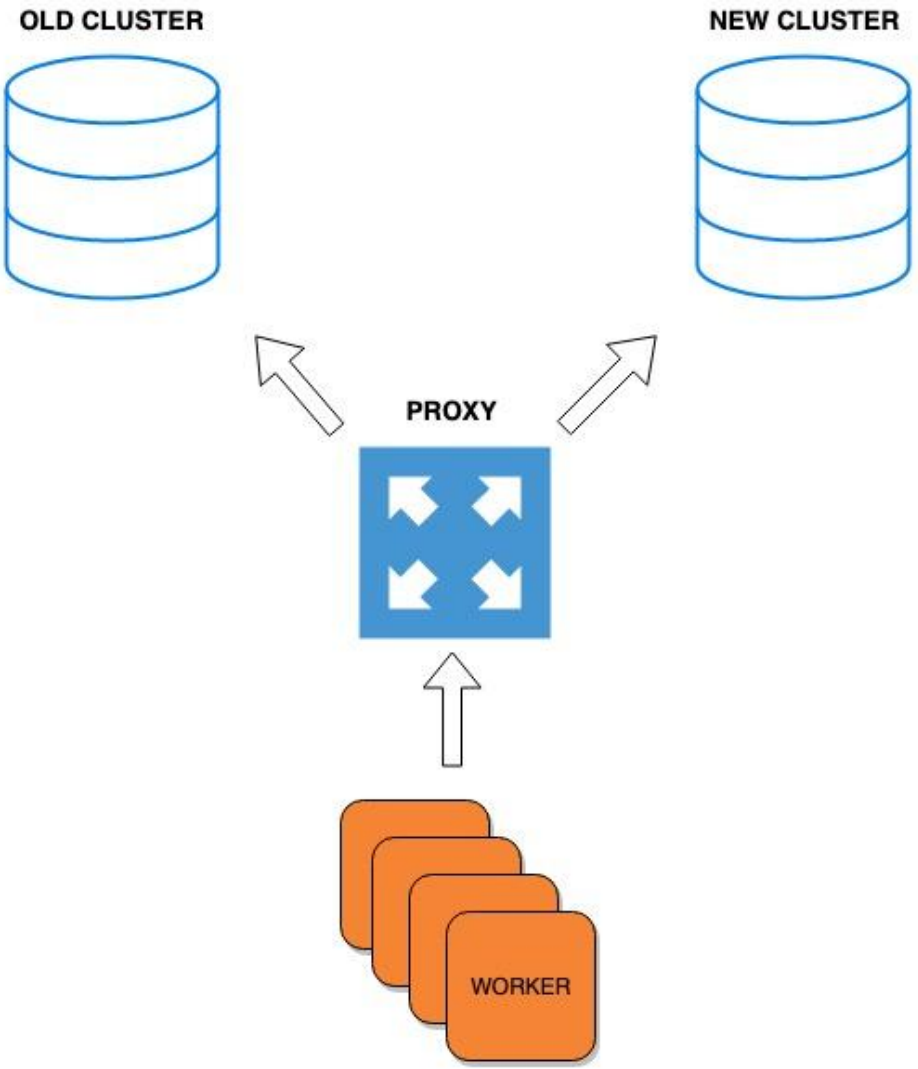
Redirect all the traffic to new cluster

The switch must be atomic to avoid database poisoning

Find Proxy to redirect traffic

No need to change backend code

More control



ProxySQL - High performance MySQL proxy tool.

Key Features

- Free software - GPL license (freedom to run, study, share and modify the software)
- Query caching
- Query Routing
- Supports failover
- Connection multiplexing

Some use cases:

- Read/Write Split
- Query rewrite

Cloudinary

ProxySQL - admin interface

Default user = admin

Default pass = admin

Default port 6032 (localhost only)

3 levels of configuration:

- Memory - operational database
- Runtime - actually running now
- Disk - will run after restart

```
mysql -u admin -padmin -h 127.0.0.1 -P6032

Admin> INSERT INTO mysql_servers(hostgroup_id,hostname,port) VALUES
(1,'127.0.0.1',21891);

Admin> LOAD MYSQL SERVERS TO RUNTIME;
Admin> SAVE MYSQL SERVERS TO DISK;
```

Cloudinary

ProxySQL - configuration files

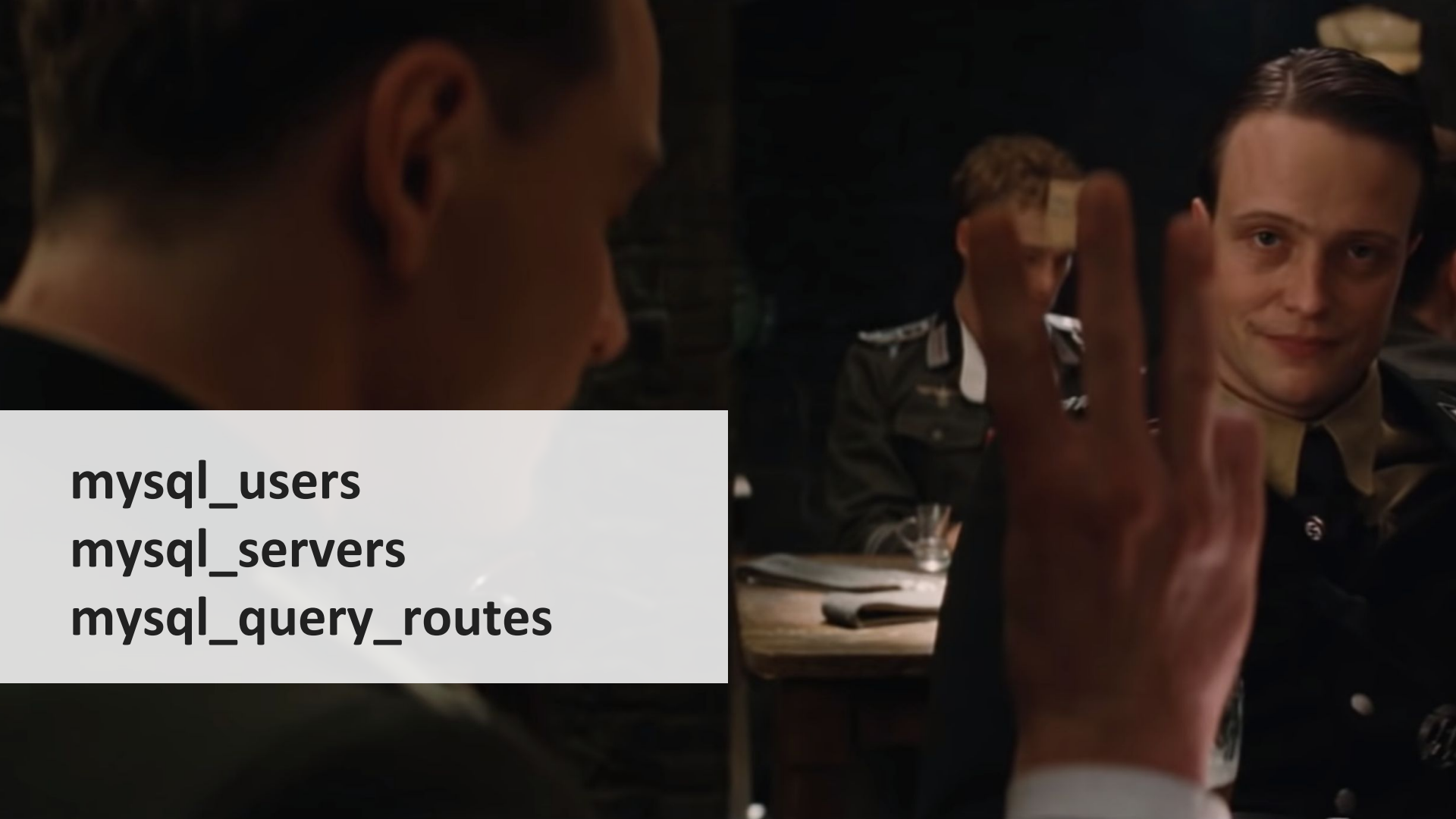
Format = .cfg / .cfn

Good for deployment and provision

```
mysql_group_replication_hostgroups=
(
        {
                writer_hostgroup=10
                reader_hostgroup=20
                backup_writer_hostgroup=11
                offline_hostgroup=0
                active=1
                max_writers=1
                writer_is_also_reader=0
                max_transactions_behind=0
                comment="repl group 1"
        }
)

mysql_users:
(
  {
    username = "root"
    password = "root"
    default_hostgroup = 0
    max_connections=1000
    default_schema="information_schema"
    active = 1
  }
)
```

Cloudinary



mysql_users
mysql_servers
mysql_query_routes

transaction_persistent (0, 1)

Once a transaction is started, it is possible that some queries are sent to a different hostgroup based on query rules. To prevent this to happen, it is possible to enable transaction_persistent.

status (ONLINE, OFFLINE_HARD, OFFLINE_SOFT)

To gracefully disable a backend server it is required to change its status to OFFLINE_SOFT. Active transactions and connections will still be used, but no new traffic will be send to the node

comment

we use comment column to set server name

mysql_users

```
mysql> select username, password, active, transaction_persistent from mysql_users;
+----------+----------+--------+-----------------------+
| username | password | active | transaction_persistent |
+----------+----------+--------+-----------------------+
| appuser  | 1234     | 1      | 1                     |
+----------+----------+--------+-----------------------+
```

mysql_query_rules

```
mysql> select rule_id,active,destination_hostgroup from mysql_query_rules;
+---------+--------+----------------------+
| rule_id | active | destination_hostgroup |
+---------+--------+----------------------+
| 1       | 1      | 1                    |
+---------+--------+----------------------+
```

mysql_servers

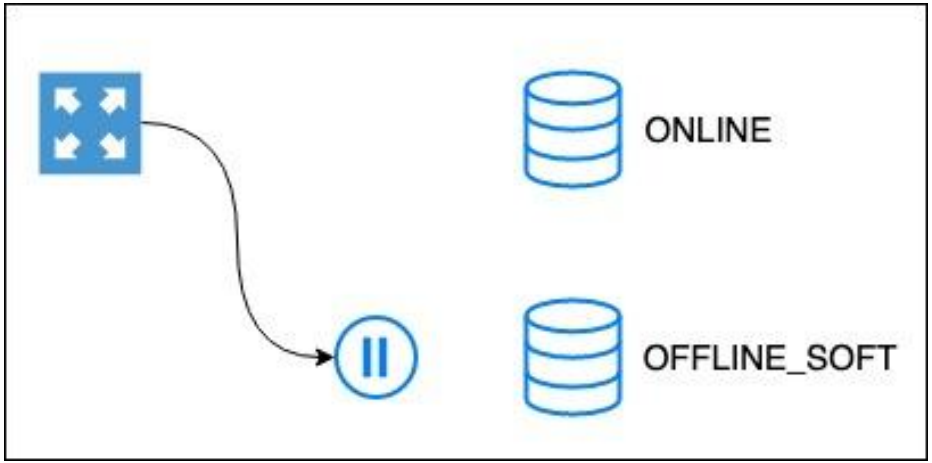
```
mysql> select hostgroup_id, hostname, status, max_connections, comment  from mysql_servers;
+--------------+-----------------------------------------------------+-------------+-----------------+-------------+
| hostgroup_id | hostname                                            | status      | max_connections | comment     |
+--------------+-----------------------------------------------------+-------------+-----------------+-------------+
| 1            | aurora1.cluster-xxxxxxx.us-east-1.rds.amazonaws.com | ONLINE      | 1000            | old cluster |
| 2            | aurora2.cluster-xxxxxxx.us-east-1.rds.amazonaws.com | OFFLINE_SOFT | 1000            | new cluster |
+--------------+-----------------------------------------------------+-------------+-----------------+-------------+
```

Solution

Redirect traffic to the new database with status=OFFLINE_SOFT

Old transactions will finish and new ones will hang in proxysql until you release it

Finally release the traffic by setting new cluster status=ONLINE



Cloudinary

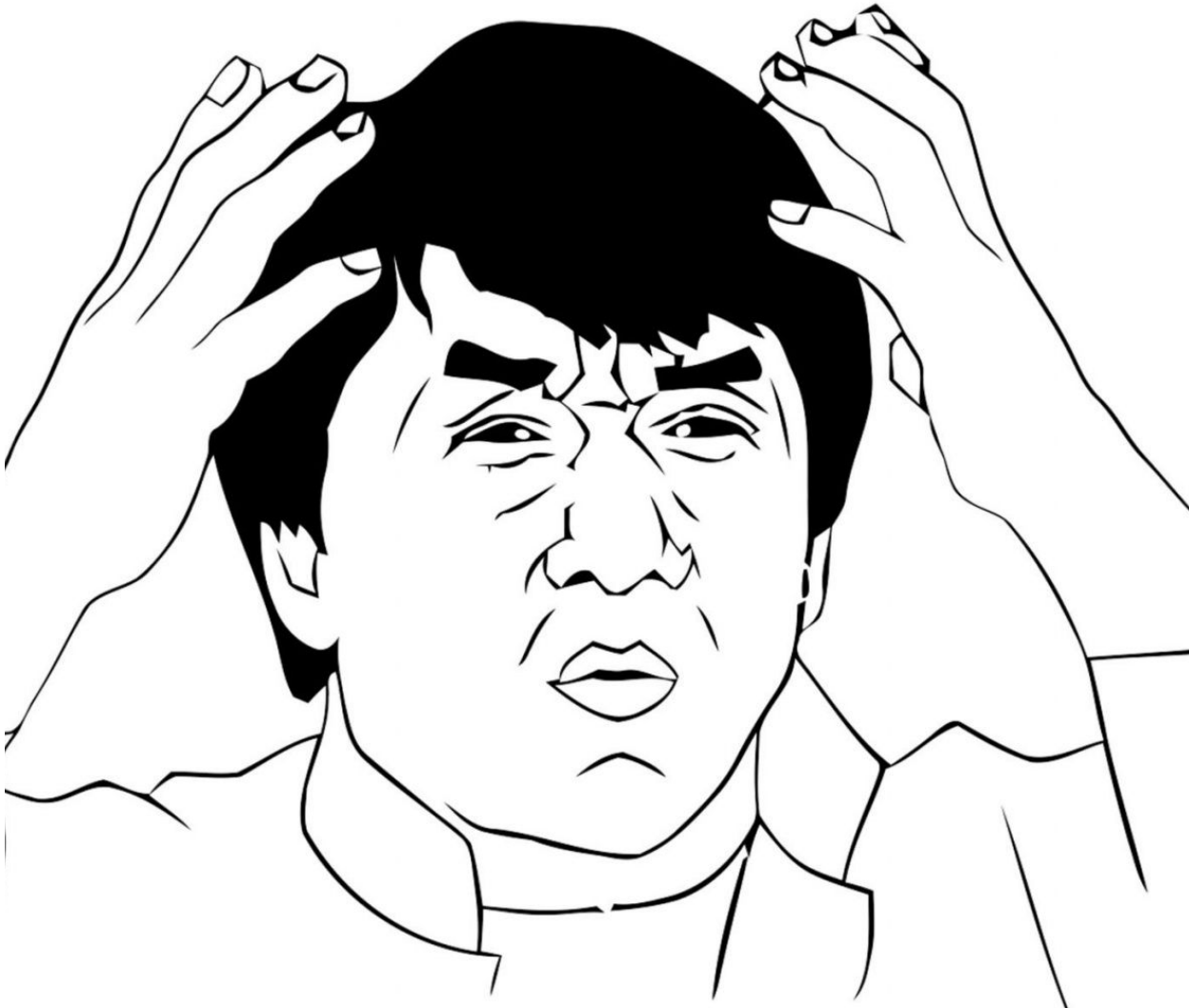
QUESTIONS?

SOLUTION

PROXYSQL CONFIGURATION TYPES

PROXYSQL CONFIGURATION LEVELS

PROXYSQL CONFIGURATION TABLES



Cloudinary

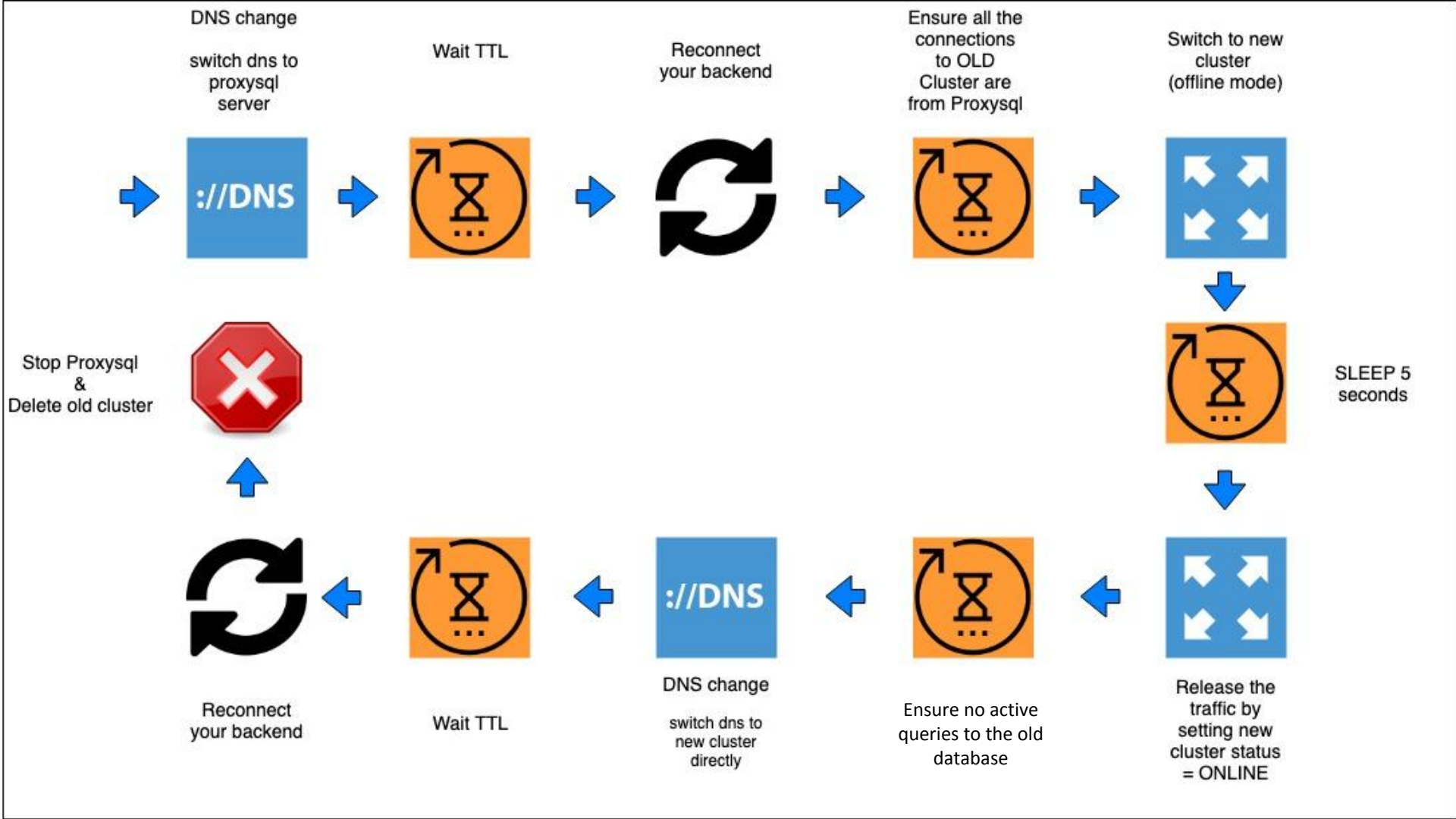
Upgrade process - prerequisites

Use Clone to create new Aurora cluster with recent version

Prepare proxysql instance(s) with all the settings. Use .cfg files for initial configuration

Manually enable 1 way replication between old and new cluster

Switch to DNS endpoints instead of Aurora cluster endpoints


DNS change
switch dns to proxysql server
Wait TTL
Reconnect your backend
Ensure all the connections to OLD Cluster are from Proxysql
Switch to new cluster (offline mode)
SLEEP 5 seconds
Release the traffic by setting new cluster status = ONLINE
Ensure no active queries to the old database
DNS change
switch dns to new cluster directly
Wait TTL
Reconnect your backend
Stop Proxysql & Delete old cluster
://DNS
://DNS

Cloudinary

LIVE DEMO

https://github.com/cloudinary-misha/proxysql-meetup

References

Github Repo of Live Demo

https://github.com/cloudinary-misha/proxysql-meetup

ProxySQL + Aurora

Seamless Planned Failover with ProxySQL
https://proxysql.com/blog/aurora-failover-without-losing-transactions

How to use ProxySQL with open source platforms to split SQL reads and writes on Amazon Aurora clusters
https://aws.amazon.com/blogs/database/how-to-use-proxysql-with-open-source-platforms-to-split-sql-reads-and-writes-on-amazon-aurora-clusters/

Cloudinary Tech Blog on Medium

https://medium.com/@cloudinary

Cloudinary

SUMMARY

PROXYSQL CONFIGURATION TYPES AND LAYERS

- SQL ADMIN INTERFACE
- CONFIGURATION FILES
- MEMORY, RUNTIME, DISK

PROXYSQL CONFIGURATION TABLES

- MYSQL_USERS
- MYSQL_QUERY_RULES
- MYSQL_SERVERS

UPGRADE PREREQUISITES

- DATABASE CLONE
- REPLICATION
- PROXYSQL SERVER CREATION
- DNS SWITCH

UPGRADE FLOW

- REDIRECT TRAFFIC TO DATABASE WITH STATUS=OFFLINE_SOFT

Cloudinary

FIN