Things Fall Apart: Navigating Amazon RDS and Aurora for Over a Decade as a Non-DBA

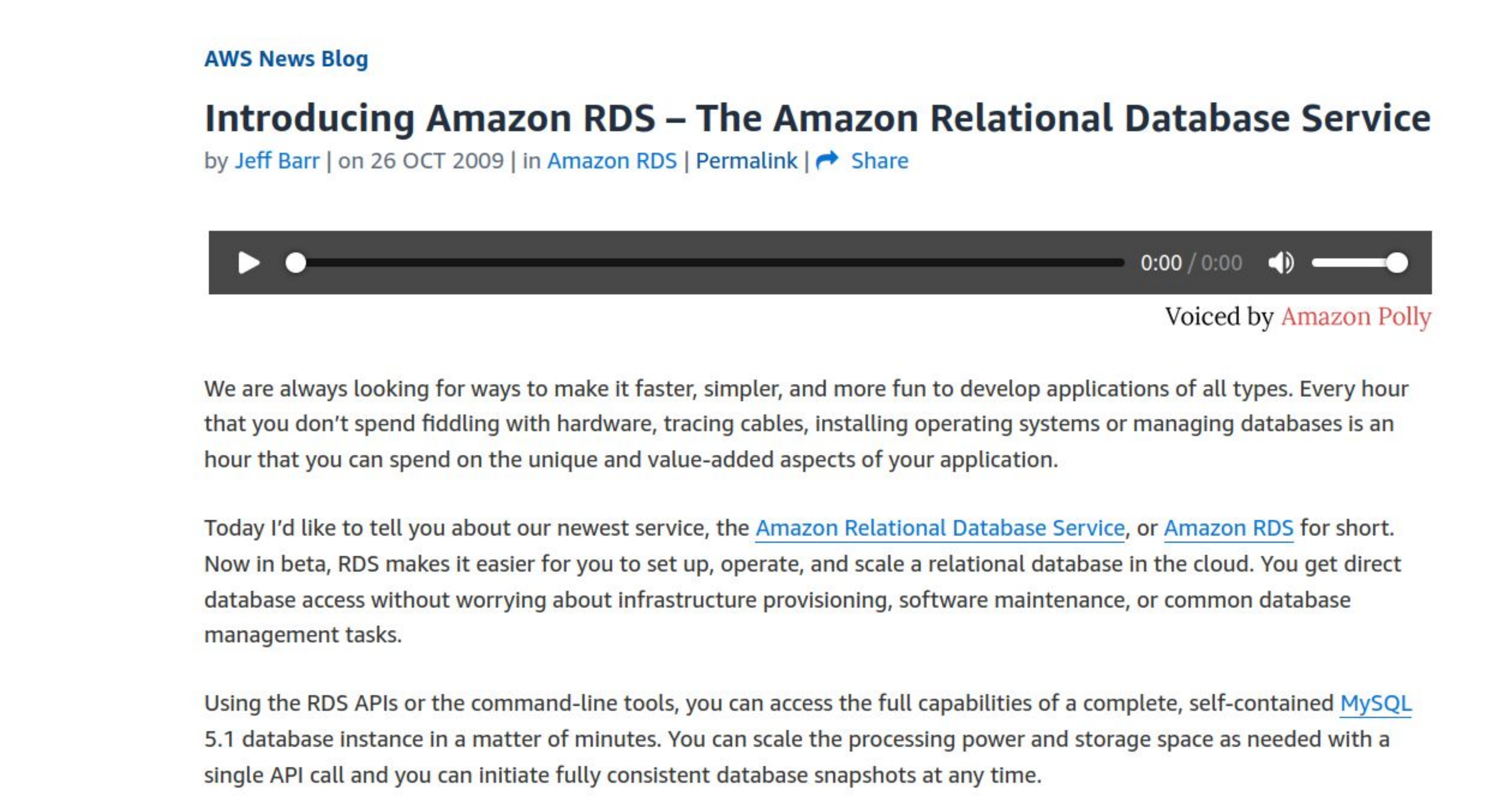
•••

Renato Losio

Agenda

1. My RDS journey so far
2. Some interesting CASEs
3. Challenges and solutions
4. Super simple experiments
5. Lessons learned

My RDS journey so far



AWS News Blog

Introducing Amazon RDS – The Amazon Relational Database Service

by Jeff Barr | on 26 OCT 2009 | in Amazon RDS | Permalink | Share

Voiced by Amazon Polly

We are always looking for ways to make it faster, simpler, and more fun to develop applications of all types. Every hour that you don't spend fiddling with hardware, tracing cables, installing operating systems or managing databases is an hour that you can spend on the unique and value-added aspects of your application.

Today I'd like to tell you about our newest service, the Amazon Relational Database Service, or Amazon RDS for short. Now in beta, RDS makes it easier for you to set up, operate, and scale a relational database in the cloud. You get direct database access without worrying about infrastructure provisioning, software maintenance, or common database management tasks.

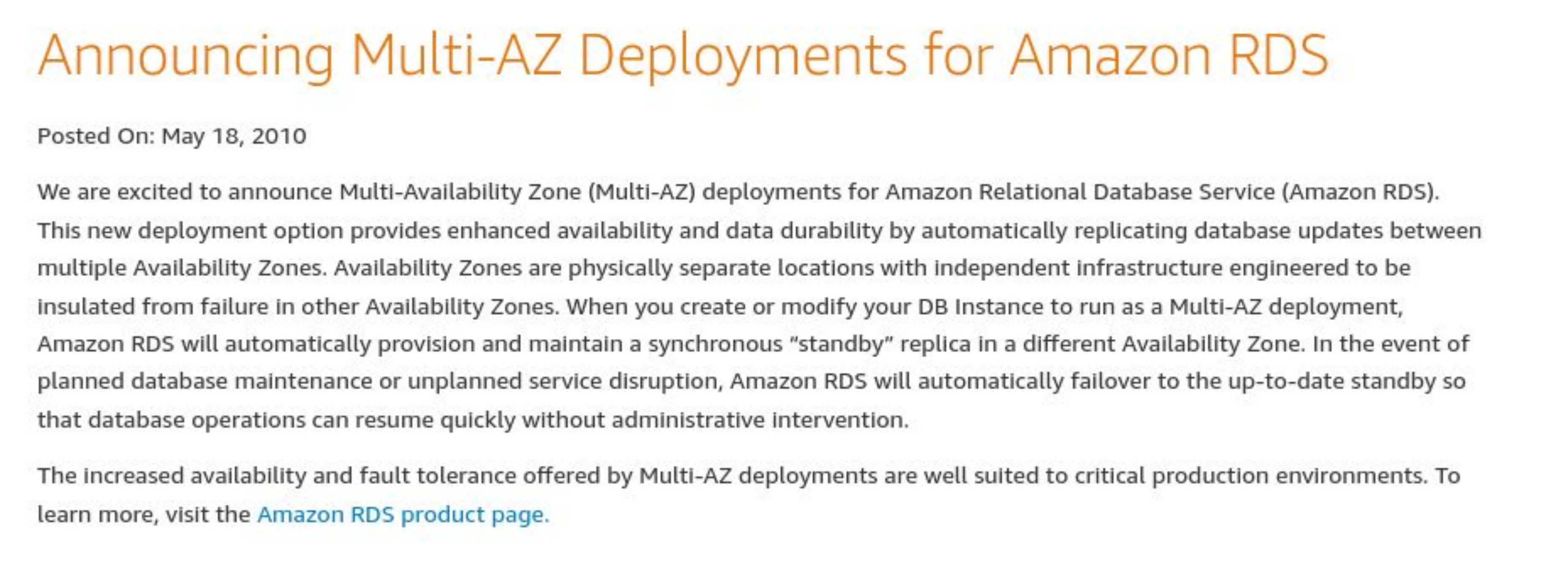
Using the RDS APIs or the command-line tools, you can access the full capabilities of a complete, self-contained MySQL 5.1 database instance in a matter of minutes. You can scale the processing power and storage space as needed with a single API call and you can initiate fully consistent database snapshots at any time.

Back in 2011

Simplicity

Constraint

More expensive



Announcing Multi-AZ Deployments for Amazon RDS

Posted On: May 18, 2010

We are excited to announce Multi-Availability Zone (Multi-AZ) deployments for Amazon Relational Database Service (Amazon RDS). This new deployment option provides enhanced availability and data durability by automatically replicating database updates between multiple Availability Zones. Availability Zones are physically separate locations with independent infrastructure engineered to be insulated from failure in other Availability Zones. When you create or modify your DB Instance to run as a Multi-AZ deployment, Amazon RDS will automatically provision and maintain a synchronous "standby" replica in a different Availability Zone. In the event of planned database maintenance or unplanned service disruption, Amazon RDS will automatically failover to the up-to-date standby so that database operations can resume quickly without administrative intervention.

The increased availability and fault tolerance offered by Multi-AZ deployments are well suited to critical production environments. To learn more, visit the Amazon RDS product page.

m1.xlarge	0.465
m2.xlarge	0.330
m3.xlarge	0.370
m4.xlarge	0.350
m5.xlarge	0.342
m6i.xlarge	**0.342**
m6g.xlarge	**0.304**
m7g.xlarge	**0.337**
t3.xlarge	**0.272**
t4g.xlarge	**0.258**

GP2 (or GP3)

MariaDB

Oracle

Provisioned
IOPS

SQLServer

RDS without

PostgreSQL

Amazon
Aurora

PerformanceInsights

Iterate, and keep up to date

Some interesting CASEs

Learn through AWS Support

[CASE *****1] Multi AZ replica broken after scale up from m7g.2xlarge to m7g.4xlarge

[CASE *****2] Aurora MySQL minor roadmap versus RDS

[CASE *****3] Enable Amazon RDS Optimized Writes using RDS Blue/Green Deployments

[CASE *****4] RDS stuck in Storage-optimization 99% for almost 24 hours

[CASE *****5] System snapshot creation time exploded without significant (...)

[CASE *****6] Support for innodb_redo_log_capacity?

[CASE *****7] No way to delete old automatic snapshot outside backup period

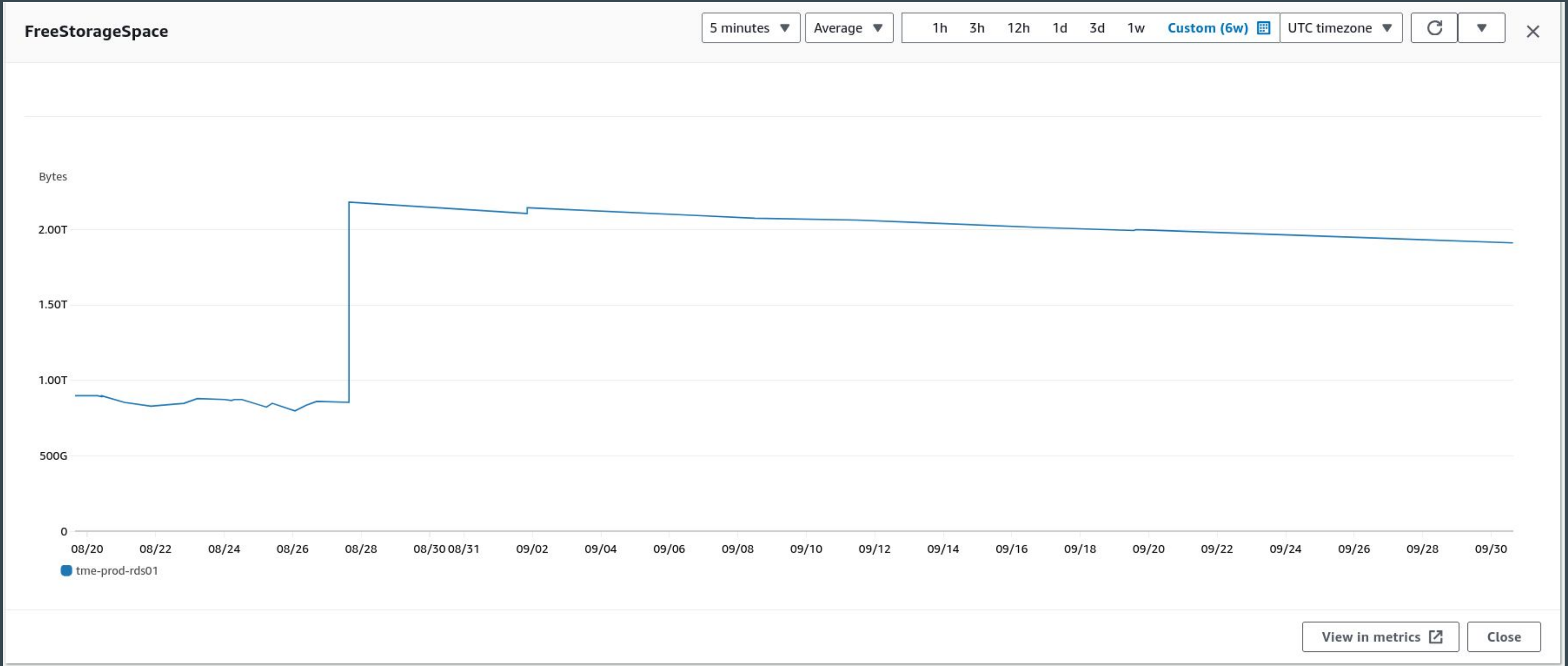
[CASE *****8] binlog_transaction_compression + binlog_transaction_compression_level_zstd

[CASE *****9] RDS, Blue/Green and backup retention

[CASE *****0] mysqld crash with optimize table statement

Challenges and solutions

How much free storage you need on RDS?



1) Storage autoscaling

```
OPTIMIZE TABLE MyLargeTable;

ALTER TABLE MyLargeTable MODIFY ENUM myEnum(U,A,B,D);
```

```
EVENTS  2024-08-25T15:48:40.477000+00:00       Applying autoscaling-initiated
modification to allocated storage.
arn:aws:rds:eu-west-1:053652639***:db:***-rds01-read-replica
tme-prod-rds01-read-replica  db-instance
```

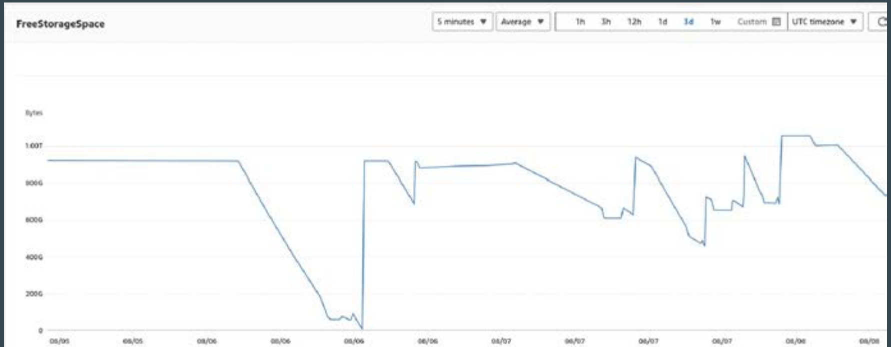
How to ALTER/OPTIMIZE an InnoDB table?

How?

- Native ALTER TABLE
- Third-party tool (pt-online-schema-change, etc.)

Where?

- Primary node
- Replica node (stop + swap)
- Amazon RDS Blue/Green



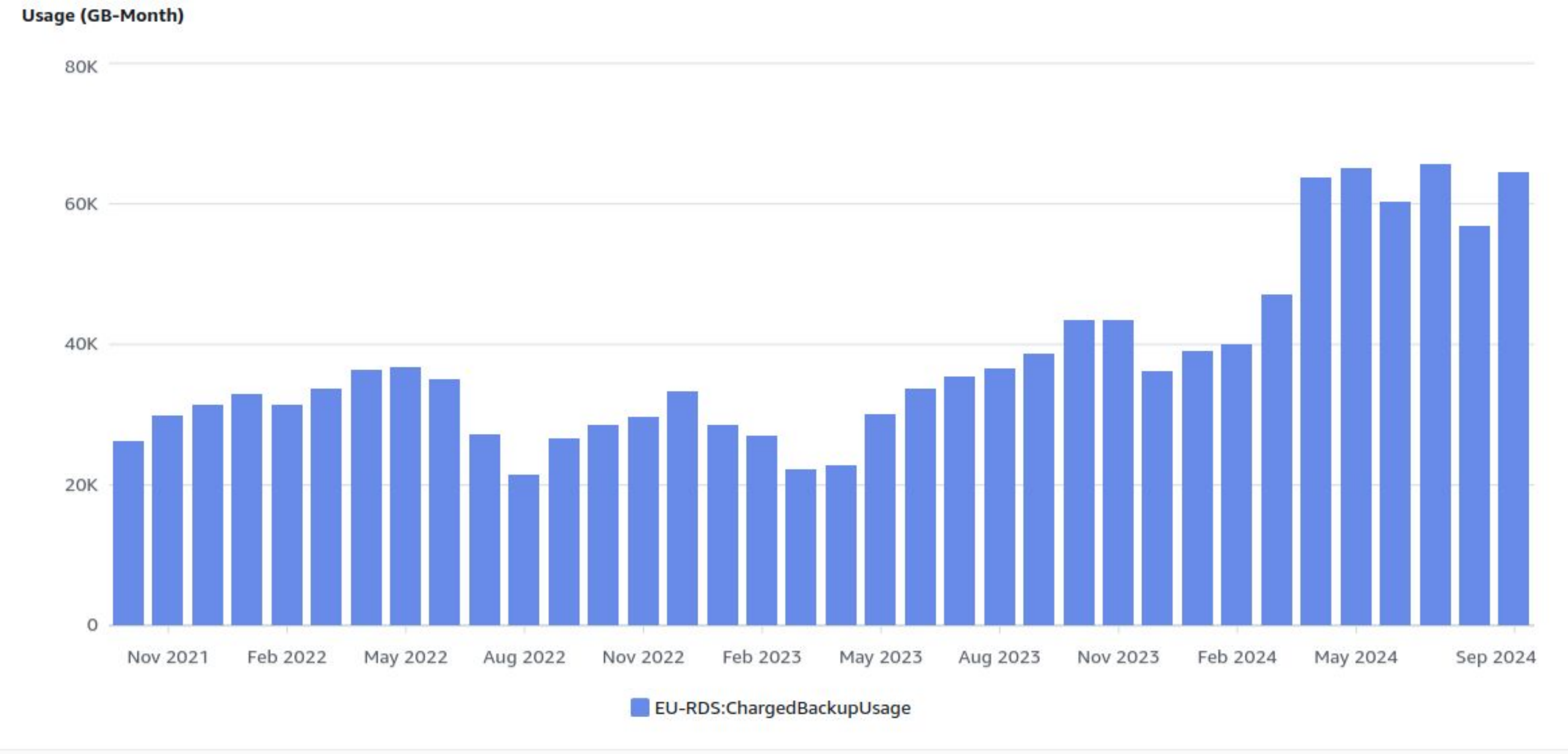
There is no additional charge for backup storage, up to 100% of your total database storage for a region. (Based upon our experience as database administrators, the vast majority of databases require less raw storage for a backup than for the primary dataset, meaning that most customers will never pay for backup storage.)

(AWS Documentation)

2) CPU is overestimated

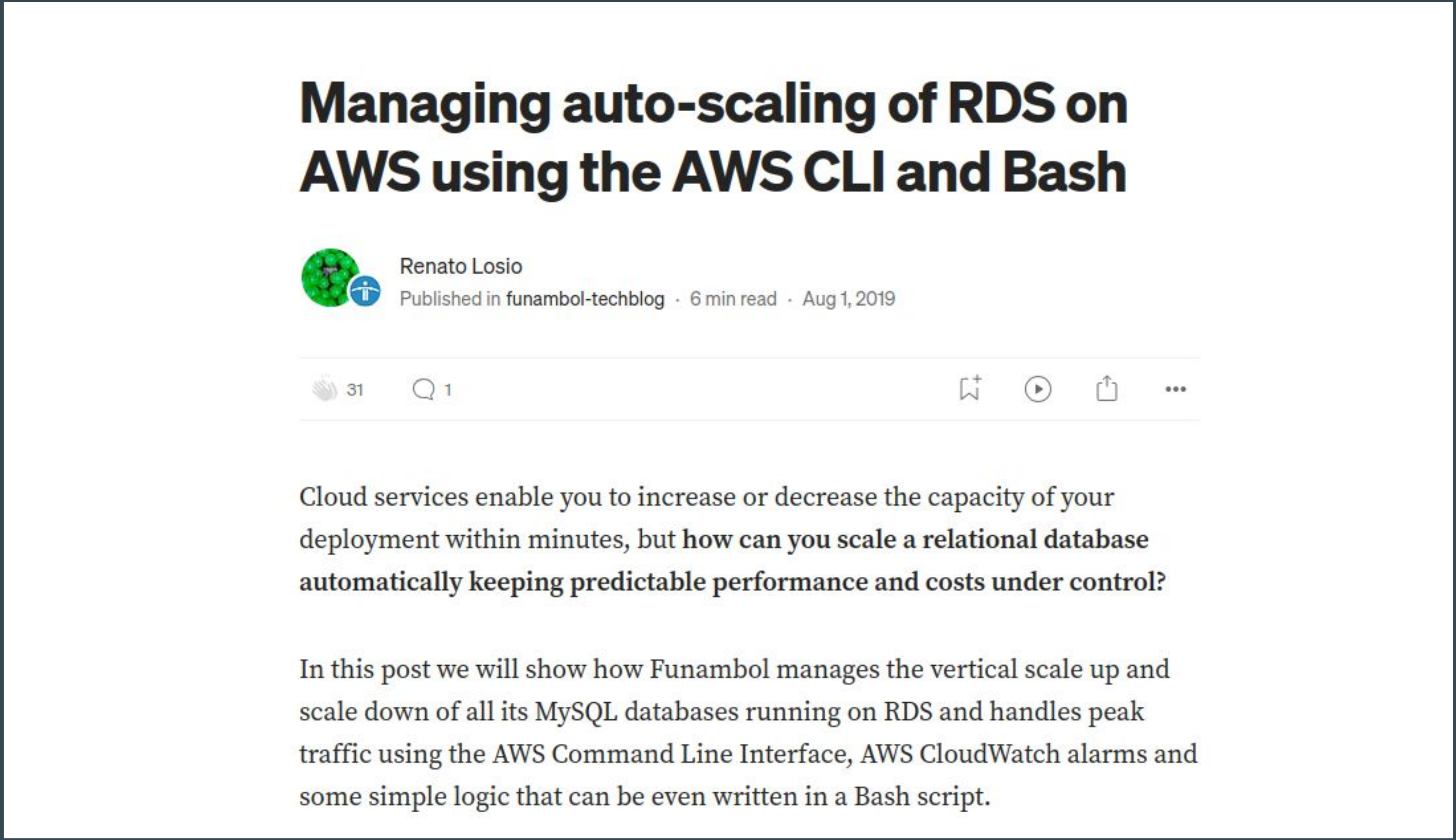
Monthly storage and backup storages grow over time

USD/Month	**2022**	**2023**	**2024**
CPU (RI)	3336	4682	6006
Storage (GP)	1960	2387	2994
Backup	2274	3300	5480


Usage (GB-Month)
80K
60K
40K
20K
0
Nov 2021
Feb 2022
May 2022
Aug 2022
Nov 2022
Feb 2023
May 2023
Aug 2023
Nov 2023
Feb 2024
May 2024
Sep 2024
EU-RDS:ChargedBackupUsage

Amazon RDS supports ARM or x86 processors? Good for you, you're a freaking database service. I couldn't possibly care less about that; just run my workload and stop bothering me with trivia.

Corey Quinn



Managing auto-scaling of RDS on AWS using the AWS CLI and Bash

Renato Losio
Published in funambol-techblog · 6 min read · Aug 1, 2019

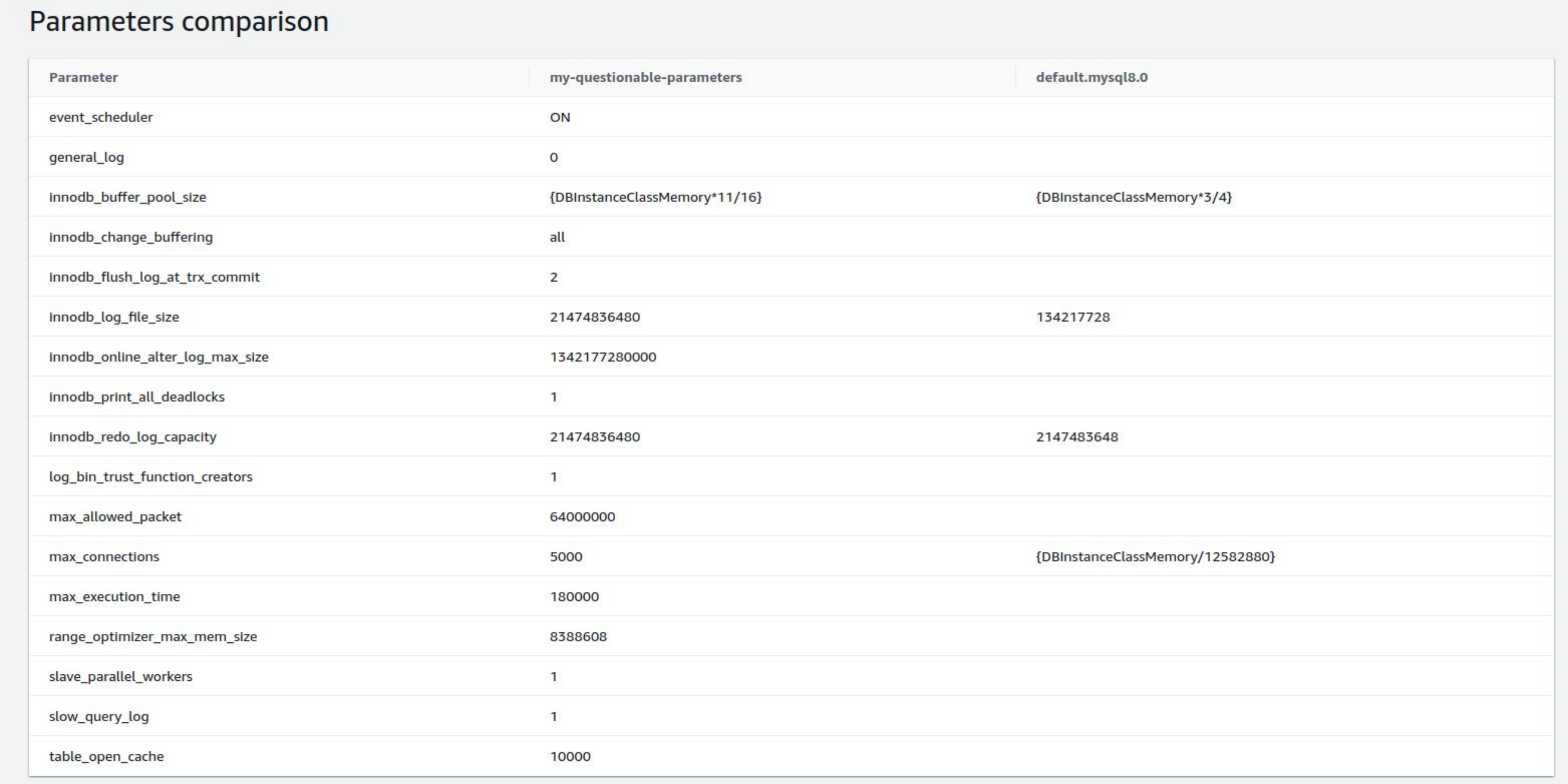
Cloud services enable you to increase or decrease the capacity of your deployment within minutes, but **how can you scale a relational database automatically keeping predictable performance and costs under control?**

In this post we will show how Funambol manages the vertical scale up and scale down of all its MySQL databases running on RDS and handles peak traffic using the AWS Command Line Interface, AWS CloudWatch alarms and some simple logic that can be even written in a Bash script.

3) How I crashed my Graviton instance

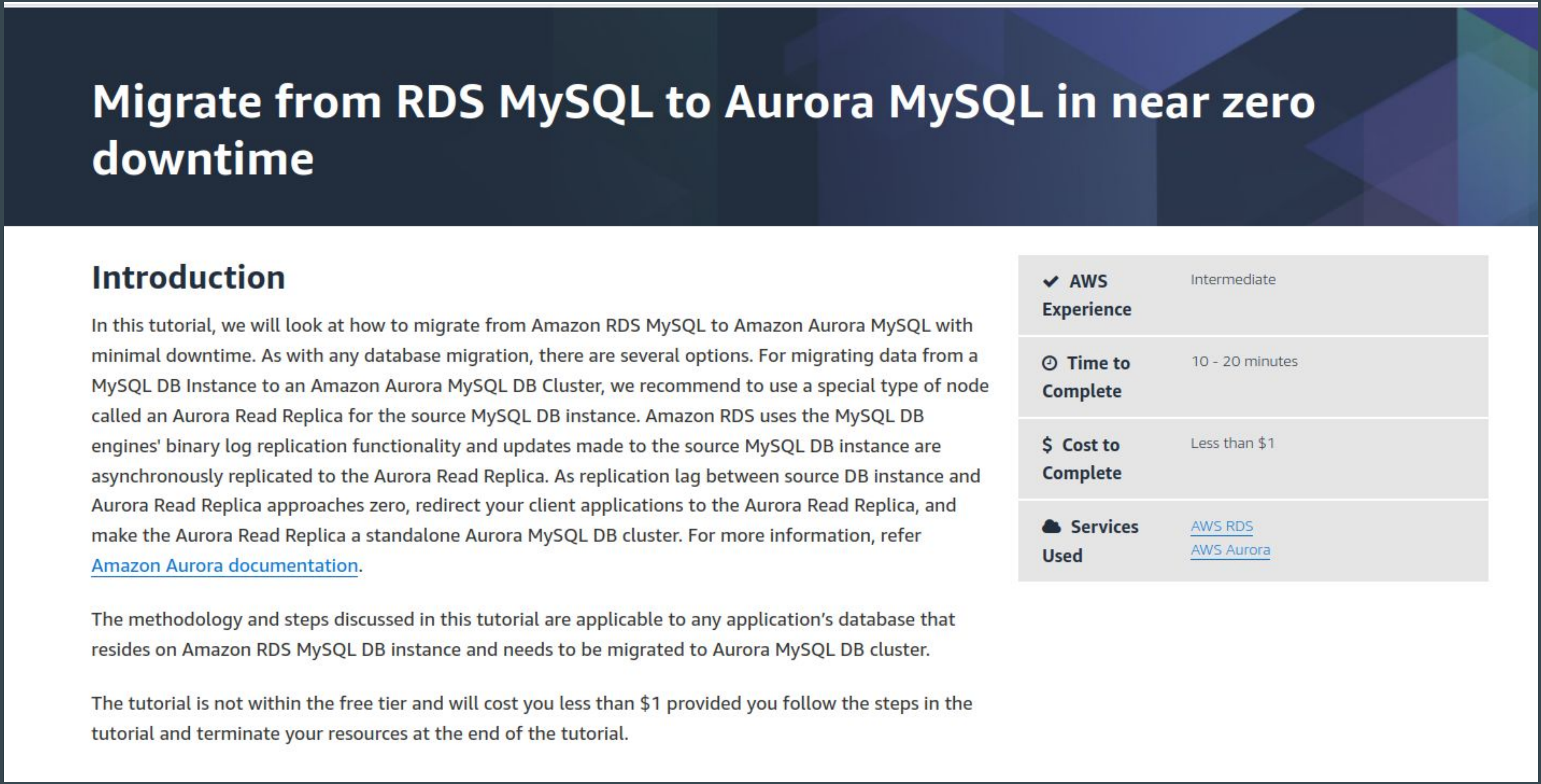
```
Most likely, you have hit a bug, but this error can also be caused by
malfunctioning hardware.
Thread pointer: 0x4017ad88f800
Attempting backtrace. You can use the following information to find out
where mysqld died. If you see no messages after this, something went
terribly wrong...
stack_bottom = 4017f3b1ee60 thread_stack 0x40000
/rdsdbbin/mysql/bin/mysqld(my_

print_stacktrace(unsigned char const*, unsigned long)+0x30) [0x1f7bb30]
/rdsdbbin/mysql/bin/mysqld(print_fatal_signal(int)+0x27c) [0xfaa93c]
/rdsdbbin/mysql/bin/mysqld(handle_fatal_signal+0xc4) [0xfaaa64]
linux-vdso.so.1(__kernel_rt_sigreturn+0) [0x400014b92860]
/rdsdbbin/mysql/bin/mysqld(trx_undo_read_v_cols(dict_table_t const,
```



Parameters comparison

Parameter	my-questionable-parameters	default.mysql8.0
event_scheduler	ON	
general_log	0	
innodb_buffer_pool_size	{DBInstanceClassMemory*11/16}	{DBInstanceClassMemory*3/4}
innodb_change_buffering	all	
innodb_flush_log_at_trx_commit	2	
innodb_log_file_size	21474836480	134217728
innodb_online_alter_log_max_size	1342177280000	
innodb_print_all_deadlocks	1	
innodb_redo_log_capacity	21474836480	2147483648
log_bin_trust_function_creators	1	
max_allowed_packet	64000000	
max_connections	5000	{DBInstanceClassMemory/12582880}
max_execution_time	180000	
range_optimizer_max_mem_size	8388608	
slave_parallel_workers	1	
slow_query_log	1	
table_open_cache	10000	



Migrate from RDS MySQL to Aurora MySQL in near zero downtime

Introduction

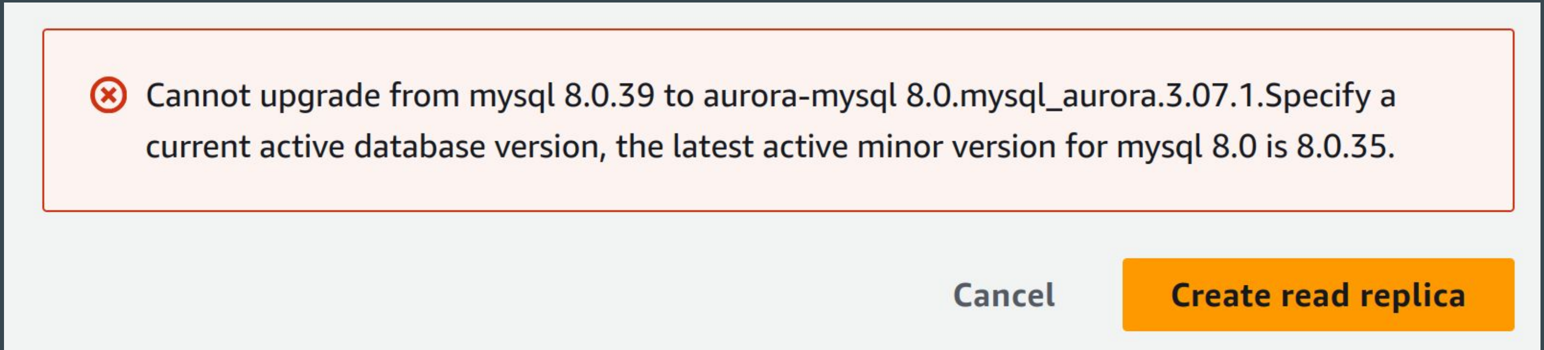
In this tutorial, we will look at how to migrate from Amazon RDS MySQL to Amazon Aurora MySQL with minimal downtime. As with any database migration, there are several options. For migrating data from a MySQL DB Instance to an Amazon Aurora MySQL DB Cluster, we recommend to use a special type of node called an Aurora Read Replica for the source MySQL DB instance. Amazon RDS uses the MySQL DB engines' binary log replication functionality and updates made to the source MySQL DB instance are asynchronously replicated to the Aurora Read Replica. As replication lag between source DB instance and Aurora Read Replica approaches zero, redirect your client applications to the Aurora Read Replica, and make the Aurora Read Replica a standalone Aurora MySQL DB cluster. For more information, refer Amazon Aurora documentation.

The methodology and steps discussed in this tutorial are applicable to any application's database that resides on Amazon RDS MySQL DB instance and needs to be migrated to Aurora MySQL DB cluster.

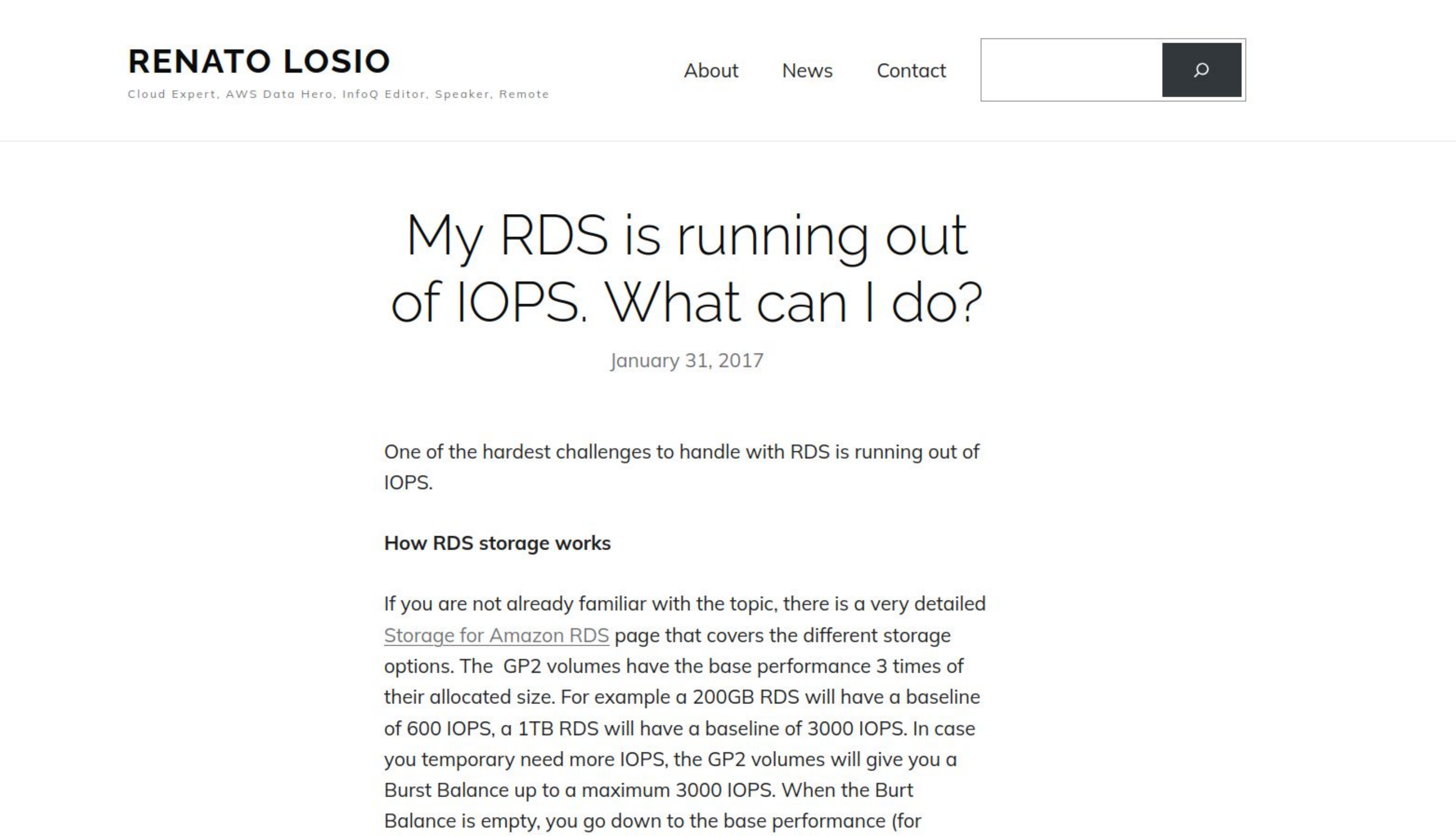
The tutorial is not within the free tier and will cost you less than $1 provided you follow the steps in the tutorial and terminate your resources at the end of the tutorial.

✓ **AWS Experience**	Intermediate
Time to Complete	10 - 20 minutes
$ Cost to Complete	Less than $1
Services Used	AWS RDS AWS Aurora

4) Going too fast. And missing Aurora.



Cannot upgrade from mysql 8.0.39 to aurora-mysql 8.0.mysql_aurora.3.07.1.Specify a current active database version, the latest active minor version for mysql 8.0 is 8.0.35.



RENATO LOSIO

Cloud Expert, AWS Data Hero, InfoQ Editor, Speaker, Remote

About News Contact

My RDS is running out of IOPS. What can I do?

January 31, 2017

One of the hardest challenges to handle with RDS is running out of IOPS.

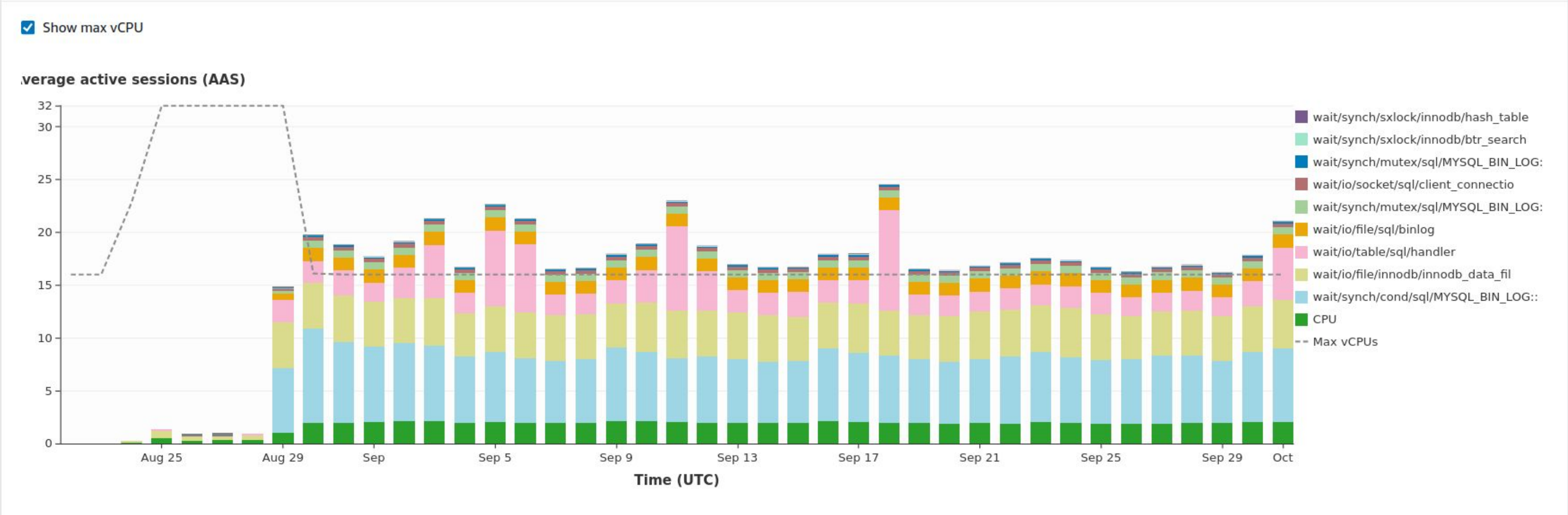
How RDS storage works

If you are not already familiar with the topic, there is a very detailed Storage for Amazon RDS page that covers the different storage options. The GP2 volumes have the base performance 3 times of their allocated size. For example a 200GB RDS will have a baseline of 600 IOPS, a 1TB RDS will have a baseline of 3000 IOPS. In case you temporary need more IOPS, the GP2 volumes will give you a Burst Balance up to a maximum 3000 IOPS. When the Burt Balance is empty, you go down to the base performance (for

5) Running out of IOPS

- You always underestimate IOPS
- The (newish) Total IOPS metric is useful
- **gp3** is usually better than **gp2**
- **gp3** is always more flexible than **gp2**
- do not use io1 on latest engines
- **io2** Express way better but often not needed
- watched out for copies, replicas, tests and staging instances

6) Blue, Green and Purple



☑ Show max vCPU

Because the green environment is a copy of the topology of the production environment, the green environment includes the features used by the DB instance. These features include the read replicas, the storage configuration, DB snapshots, automated backups, Performance Insights, and Enhanced Monitoring.

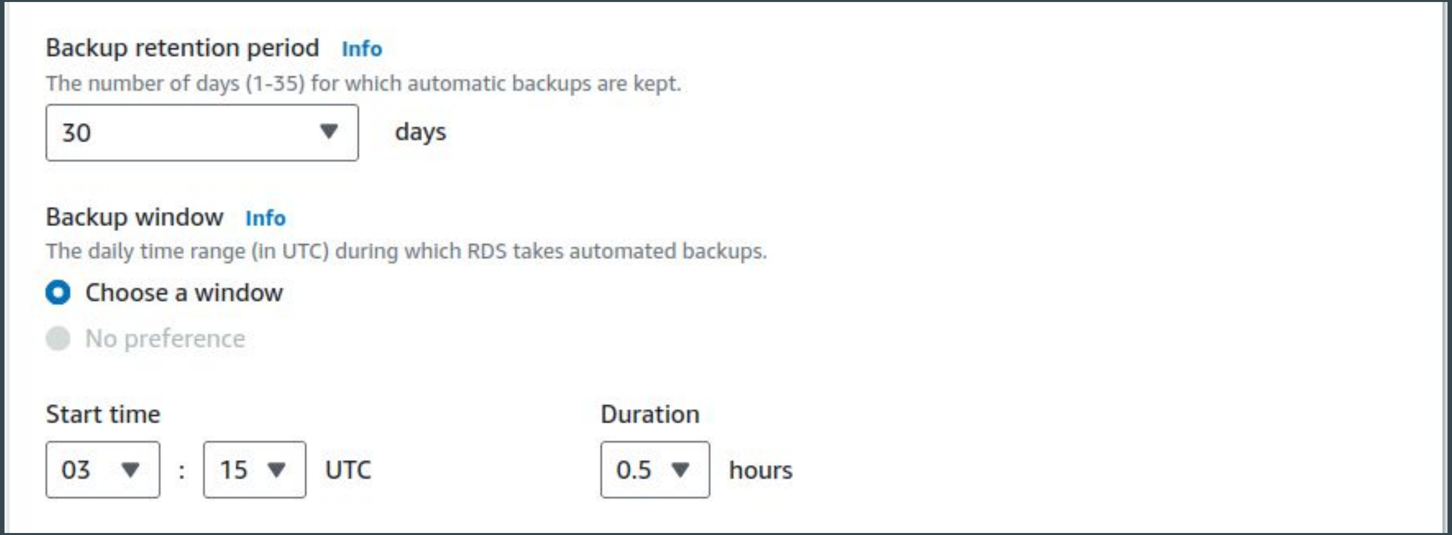
(AWS Documentation)

I was able to test this on my end and note that the **Performance Insights of the old blue instance will not be carried to the new blue instance after the blue-green deployment failover** . In addition, I have confirmed this from my internal as well.

(AWS Support)

7) The strange case of automated backups

30 days is 30 days. Or 60?



Backup retention period

The number of days (1-35) for which automatic backups are kept.

30 days

Backup window

The daily time range (in UTC) during which RDS takes automated backups.

● Choose a window

○ No preference

Start time

03 : 15 UTC

Duration

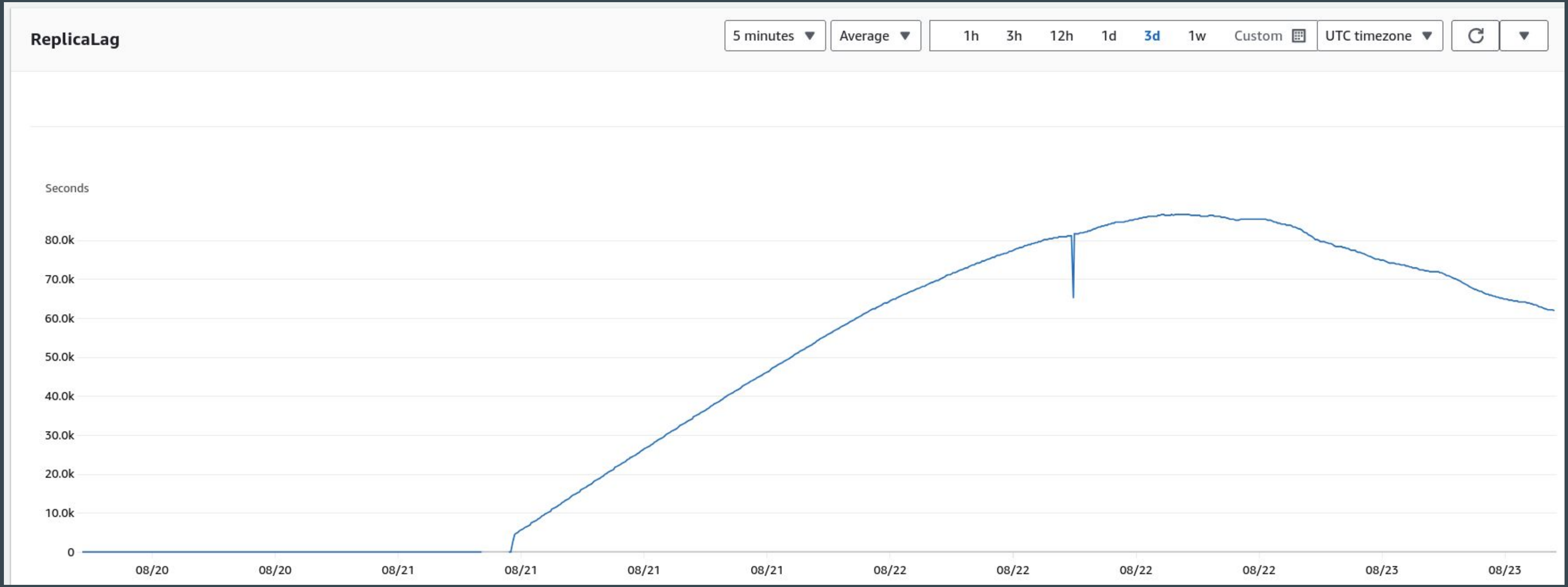
0.5 hours

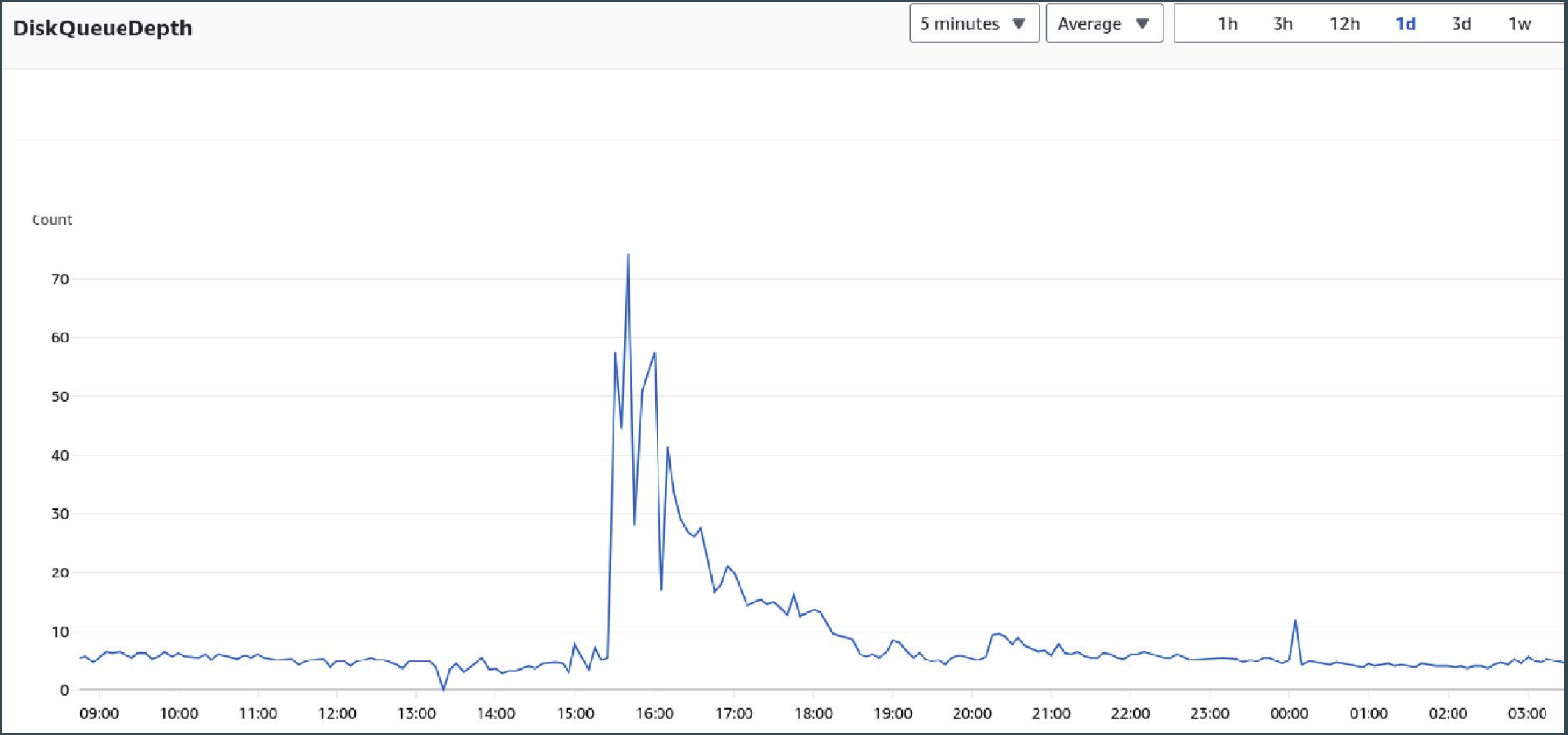
Retained automated backups are removed by the system after their last system snapshot expires. This occurs because the retention period is calculated by the running time of the database. As the instance is deleted, the only way for the system to ensure the backups were retained for the entire period specified is to set the deletion time for 30 days from the deletion of the database.

(AWS Support)

8) Underestimate time for a change

How long does an ALTER TABLE require promoting a replica?





DiskQueueDepth

5 minutes ▼ Average ▼

Super simple experiments

Benchmarking basics

- Aurora (MySQL) versus RDS for MySQL
- RDS for MySQL versus RDS for MariaDB
- io1 versus io2 Express

```
RDS setup: 400 GB + db.r7g.large
Aurora setup: db.r7g.large
```

Inserting one million records

- One at a time
- Single thread
- ACID

Developer

CPU

Engine

it depends...

Database

Table

Version

Latency

Storage

Memory

SQL

```
CREATE TABLE aws_community_day
(id bigint(20) NOT NULL AUTO_INCREMENT,
datetime TIMESTAMP NULL DEFAULT CURRENT_TIMESTAMP,
value integer DEFAULT NULL,
PRIMARY KEY (id));


DELIMITER $$
CREATE PROCEDURE load_demo()
BEGIN
  DECLARE i INT DEFAULT 0;
  WHILE i < 100000 DO
    INSERT INTO aws_community_day (value) VALUES (FLOOR(RAND()*1000));
    SET i = i + 1;
  END WHILE;
END$$
DELIMITER ;

CALL load_demo();
```

Minutes?

Seconds?

how long?

(db.m7g.large)

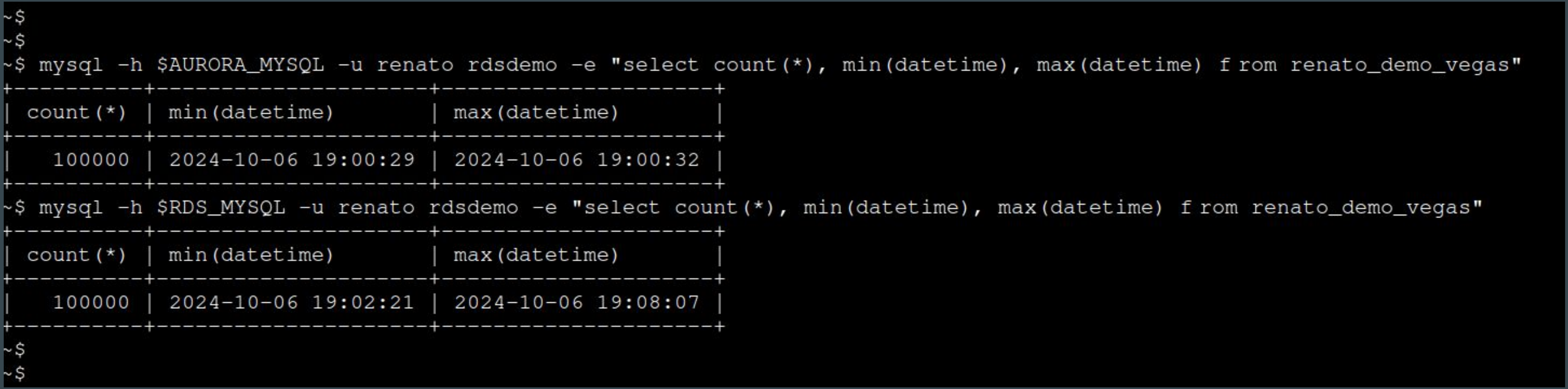
Milliseconds?

Hours?

<demo>

Aurora vs. Amazon RDS for MySQL

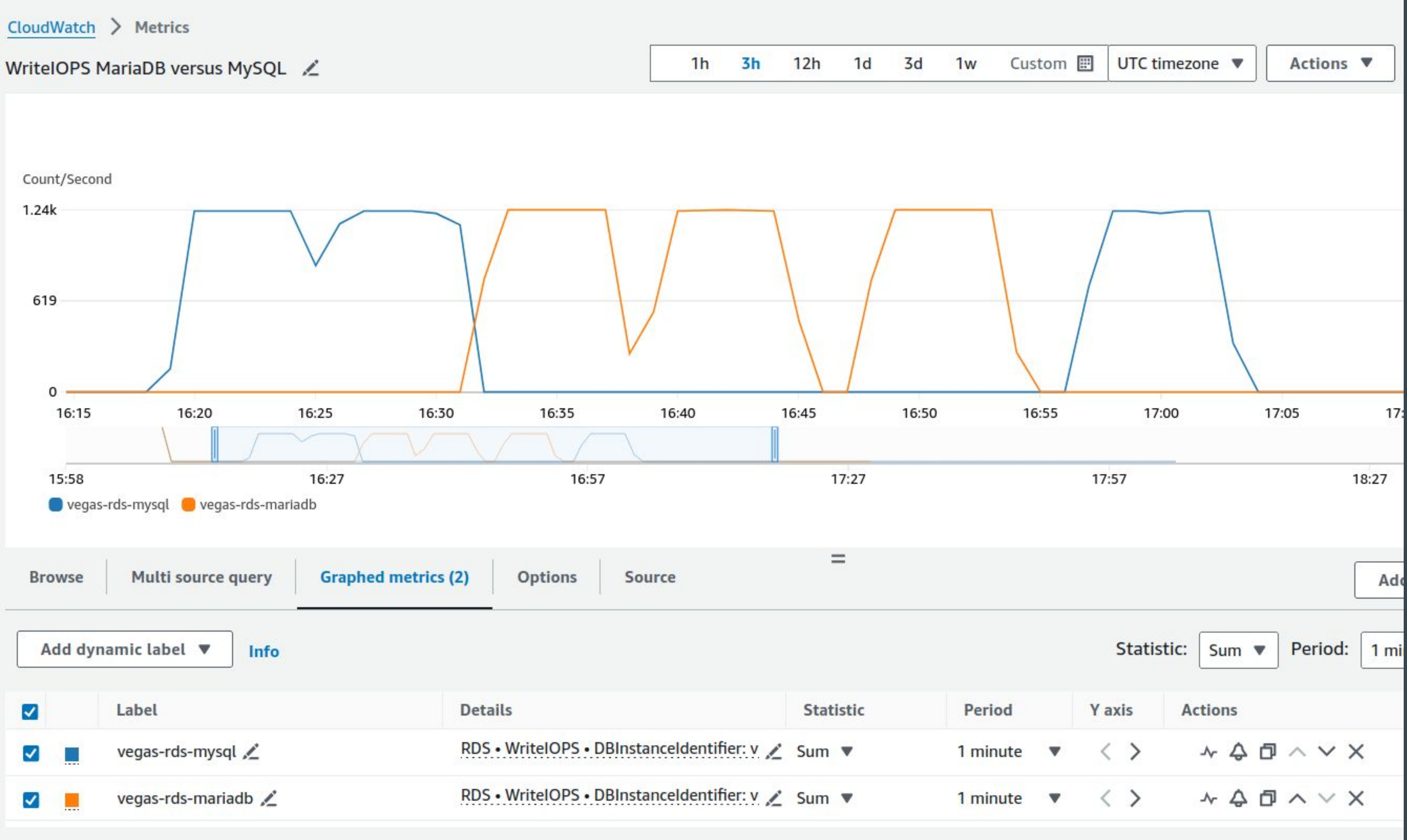
- Amazon RDS 8.0.39 (5m 49s)
- Aurora 3.05.2 (2.7s)



```
~$
~$
~$ mysql -h $AURORA_MYSQL -u renato rdsdemo -e "select count(*), min(datetime), max(datetime) from renato_demo_vegas"
+----------+---------------------+---------------------+
| count(*) | min(datetime)       | max(datetime)       |
+----------+---------------------+---------------------+
|   100000 | 2024-10-06 19:00:29 | 2024-10-06 19:00:32 |
+----------+---------------------+---------------------+
~$ mysql -h $RDS_MYSQL -u renato rdsdemo -e "select count(*), min(datetime), max(datetime) from renato_demo_vegas"
+----------+---------------------+---------------------+
| count(*) | min(datetime)       | max(datetime)       |
+----------+---------------------+---------------------+
|   100000 | 2024-10-06 19:02:21 | 2024-10-06 19:08:07 |
+----------+---------------------+---------------------+
~$
~$
```

RDS for MySQL vs. RDS for MariaDB

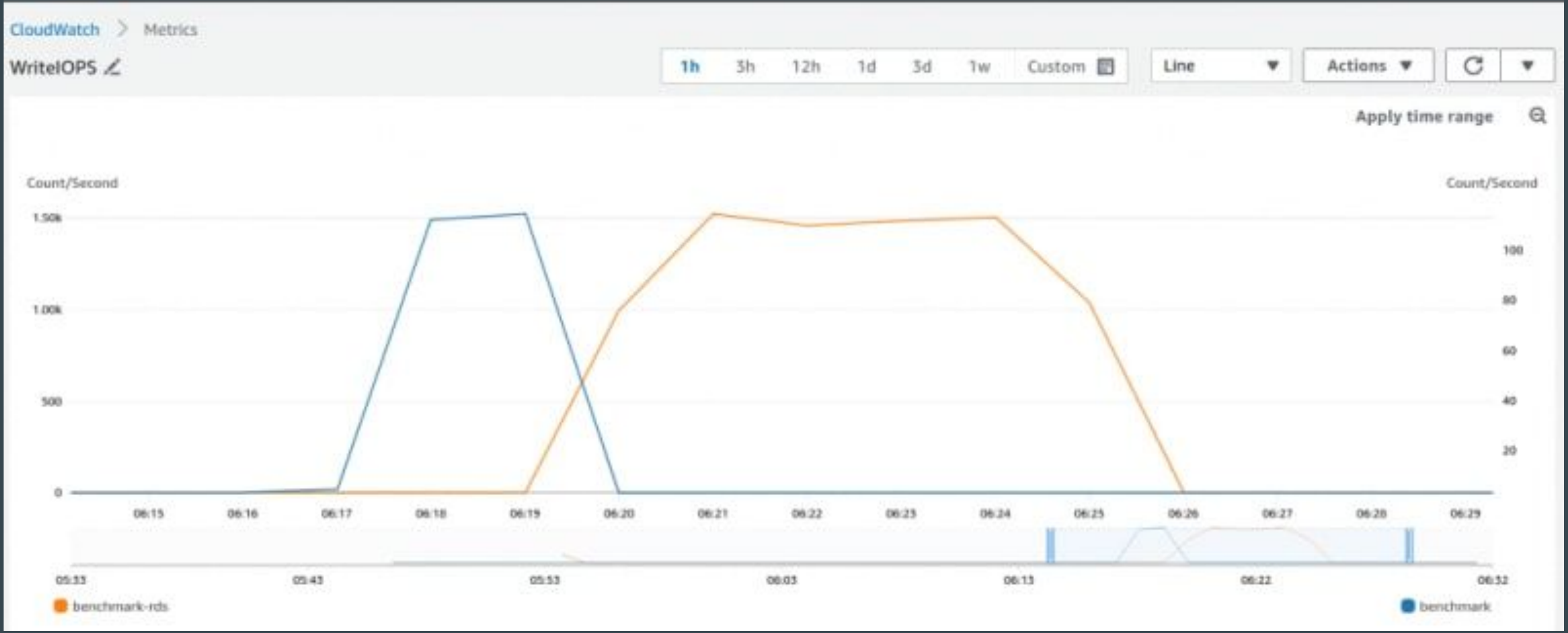
- MySQL 8.0.39 (5m 49s)
- MariaDB 10.11.8 (5m 48s)

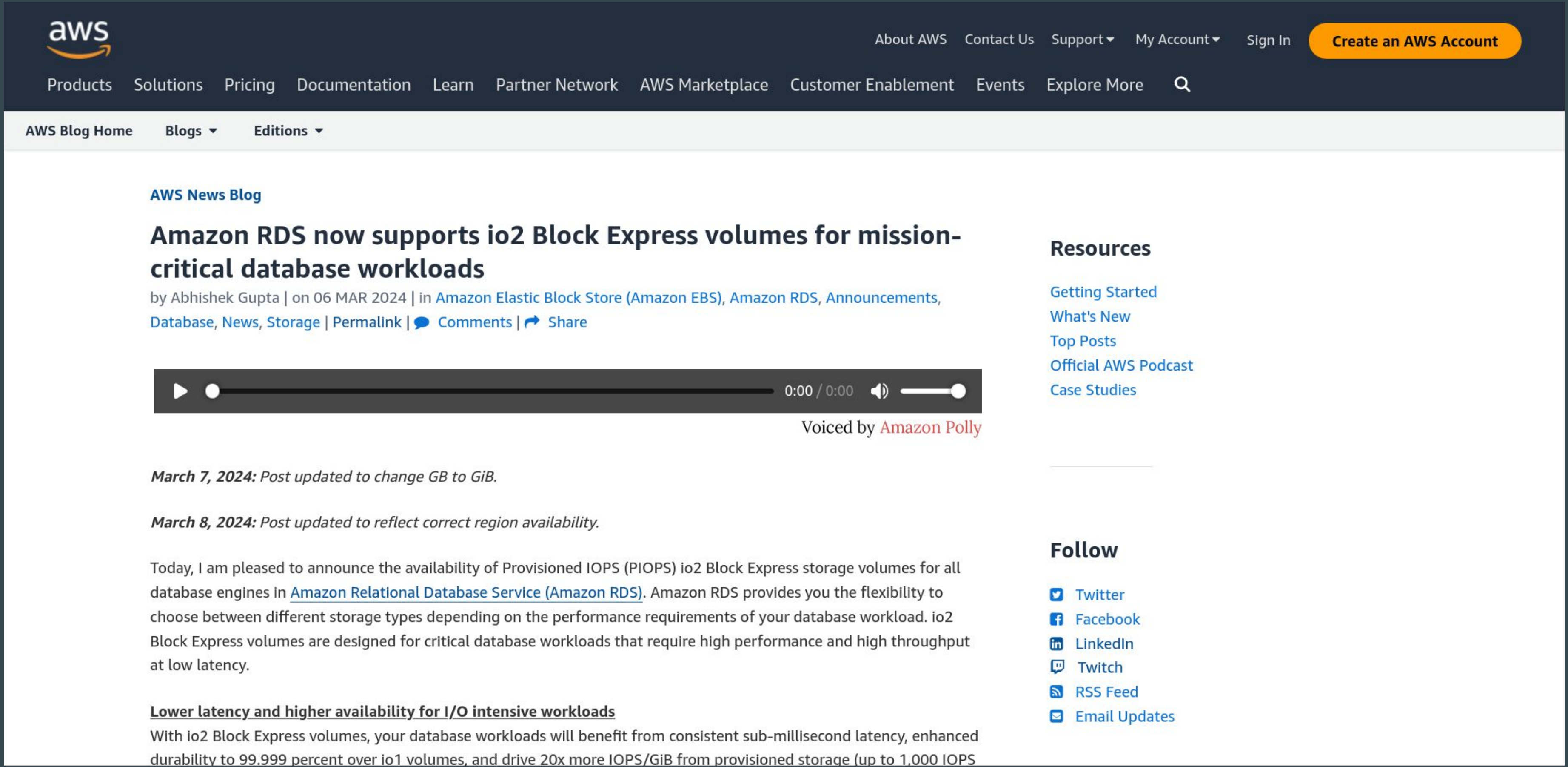


WriteIOPS MariaDB versus MySQL

Statistic: Sum ▼ Period: 1 mi

☑		Label	Details	Statistic	Period	Y axis	Actions
☑	🟦	vegas-rds-mysql	RDS • WriteIOPS • DBInstanceIdentifier: v	Sum ▼	1 minute ▼	< >	
☑	🟧	vegas-rds-mariadb	RDS • WriteIOPS • DBInstanceIdentifier: v	Sum ▼	1 minute ▼	< >	


CloudWatch > Metrics
WriteIOPS
1h
3h
12h
1d
3d
1w
Custom
Line ▼
Actions ▼
Apply time range
Count/Second
Count/Second
1.50k
1.00k
500
0
100
80
60
40
20
06:15
06:16
06:17
06:18
06:19
06:20
06:21
06:22
06:23
06:24
06:25
06:26
06:27
06:28
06:29
05:33
05:43
05:53
06:03
06:13
06:22
06:32
● benchmark-rds
● benchmark



AWS News Blog

Amazon RDS now supports io2 Block Express volumes for mission-critical database workloads

by Abhishek Gupta | on 06 MAR 2024 | in Amazon Elastic Block Store (Amazon EBS), Amazon RDS, Announcements, Database, News, Storage | Permalink | Comments | Share

Voiced by Amazon Polly

March 7, 2024: *Post updated to change GB to GiB.*

March 8, 2024: *Post updated to reflect correct region availability.*

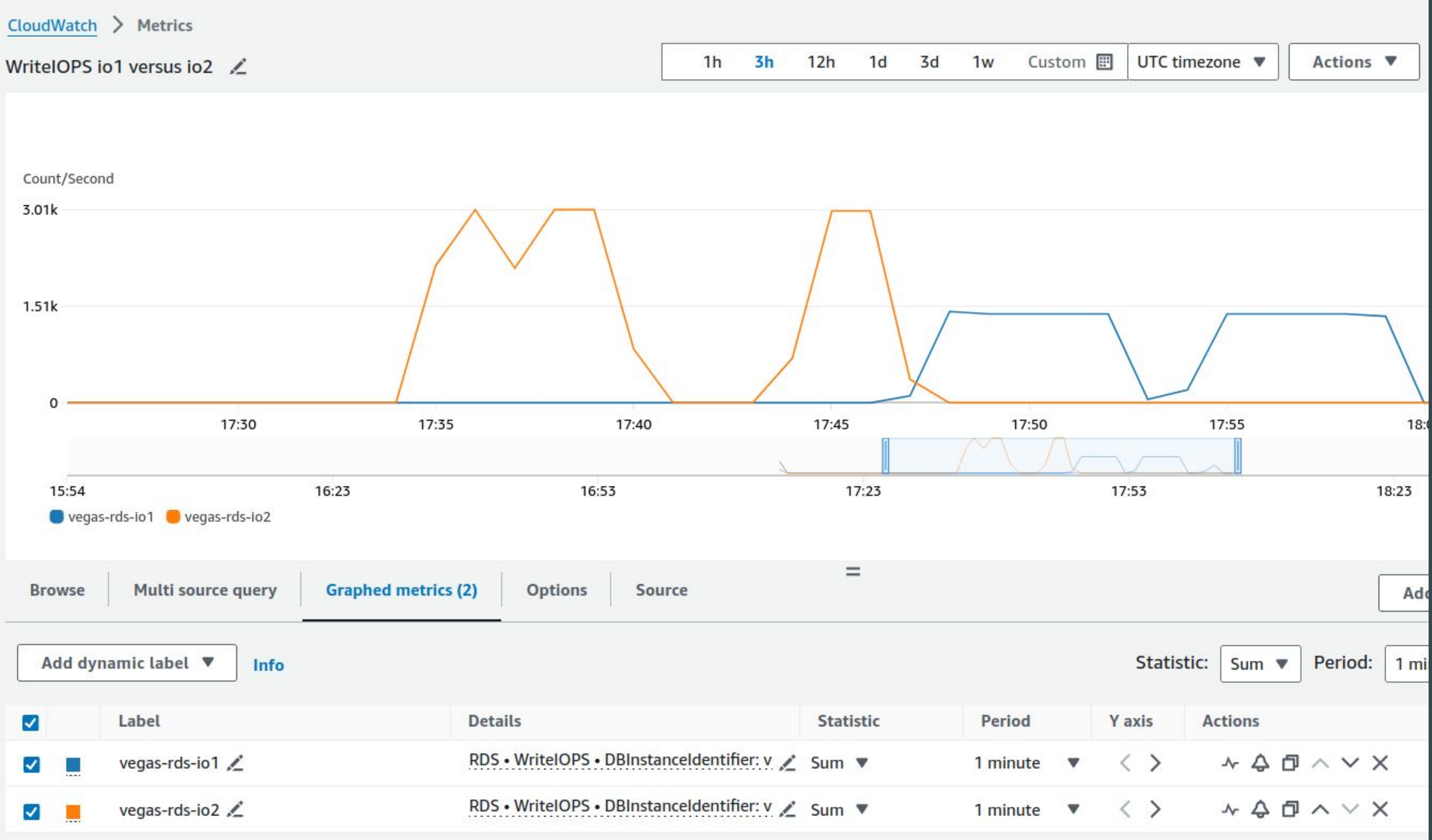
Today, I am pleased to announce the availability of Provisioned IOPS (PIOPS) io2 Block Express storage volumes for all database engines in Amazon Relational Database Service (Amazon RDS). Amazon RDS provides you the flexibility to choose between different storage types depending on the performance requirements of your database workload. io2 Block Express volumes are designed for critical database workloads that require high performance and high throughput at low latency.

Lower latency and higher availability for I/O intensive workloads

With io2 Block Express volumes, your database workloads will benefit from consistent sub-millisecond latency, enhanced durability to 99.999 percent over io1 volumes, and drive 20x more IOPS/GiB from provisioned storage (up to 1,000 IOPS

io1 vs. io2 Block Express

- io1 (5m 6s)
- io2 Block Express (2m 20s)



WriteIOPS io1 versus io2

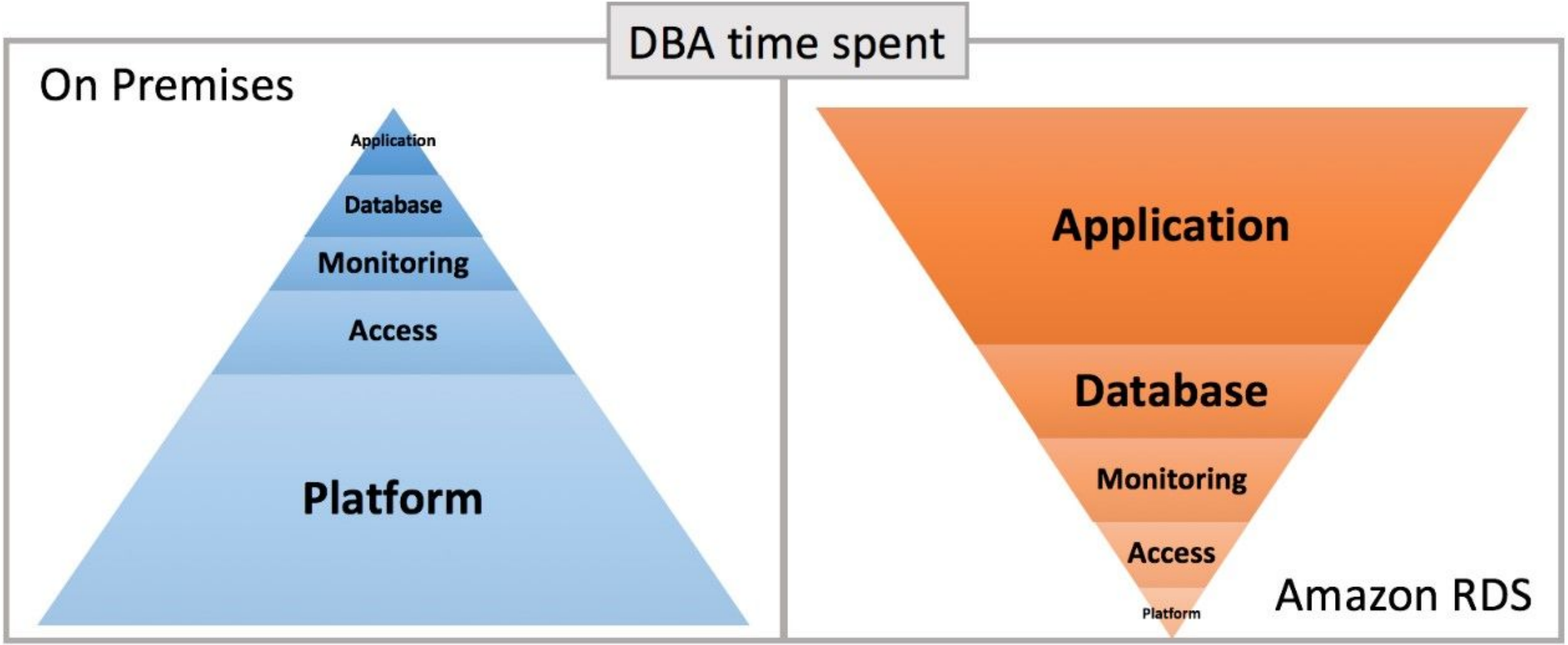
Statistic: Sum ▼ Period: 1 mi

☑		Label	Details	Statistic	Period	Y axis	Actions
☑	🟦	vegas-rds-io1	RDS • WriteIOPS • DBInstanceIdentifier: v	Sum ▼	1 minute ▼	< >	
☑	🟧	vegas-rds-io2	RDS • WriteIOPS • DBInstanceIdentifier: v	Sum ▼	1 minute ▼	< >	

With io2 Block Express volumes, your database workloads will benefit from consistent sub-millisecond latency, enhanced durability to 99.999 percent over io1 volumes, and drive 20x more IOPS/GiB from provisioned storage (up to 1,000 IOPS per GiB) at the same price as io1.

(AWS Documentation)

Are you really not a DBA?


DBA time spent
On Premises
Application
Database
Monitoring
Access
Platform
Application
Database
Monitoring
Access
Platform
Amazon RDS

Key takeaways (1/2)

- Changes to allocated RDS storage might take hours or days. Plan accordingly.
- RDS is a managed service but logs, performance schema, and parameters are all available. Look under the hood.
- Avoid changing parameter groups’ settings for short term gains. You might need to scale your instance later.

Key takeaways (2/2)

- Validate your findings and your plans with the AWS support team. They have access to your database.
- Warm up your database replica before directing traffic to it, the storage might still be cold.
- Cost optimization is vital but don’t focus only on the instance size, engine, or service.
- Approach benchmarks cautiously.

Leverage managed database, resist changes, but understand what runs under the hood.

cloudiamo.com