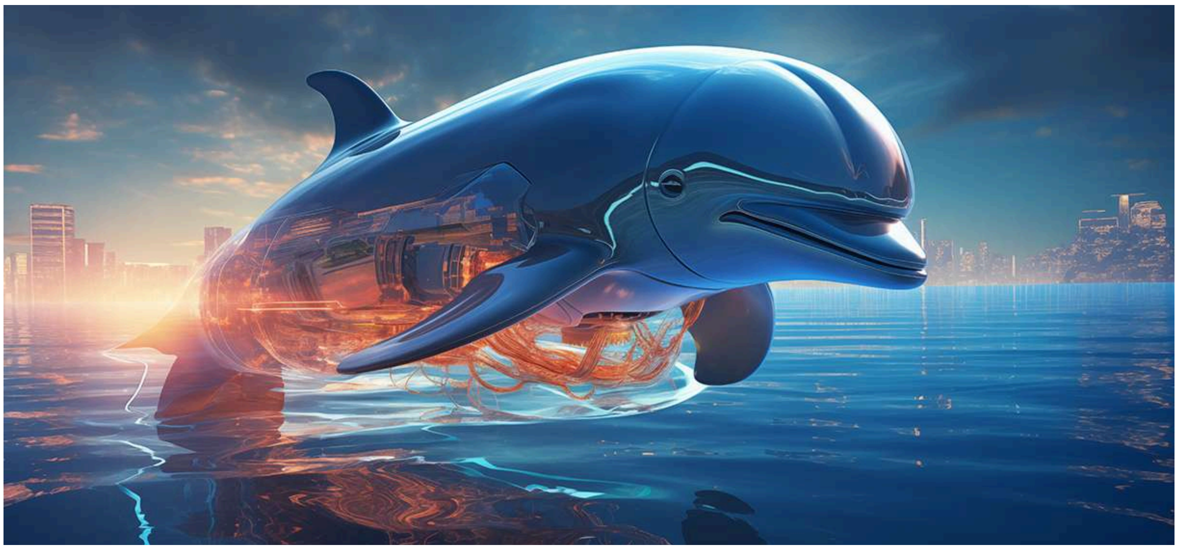


Insight for DBAs MySQL Subscribe to RSS Feed

Using Blue/Green Deployment For (near) Zero-Downtime Primary Key Updates in RDS MySQL

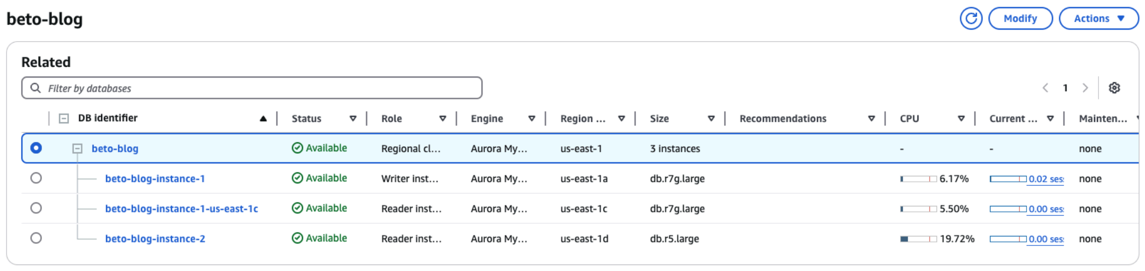
January 22, 2025 Roberto De Bem

Large tables can pose challenges for many operations when working with a database. Occasionally, we may need to modify the table definition. Since RDS replication does not use asynchronous for its replication, the typical switchover procedure is not feasible. However, the Blue/Green feature of RDS utilizes asynchronous replication, which allows us to update the table definition of large tables using the switchover procedure.

The Blue/Green deployment strategy involves creating two separate but identical environments for an application. The current environment where the application runs is referred to as “Blue,” while the environment that contains the modifications or new version is called “Green.” This continuous deployment method allows the database to be updated with minimal downtime.

In this blog post, I will demonstrate how to alter a table whose primary key is nearing its maximum value using the Blue/Green environment.

For the following blog, please assume the following environment:

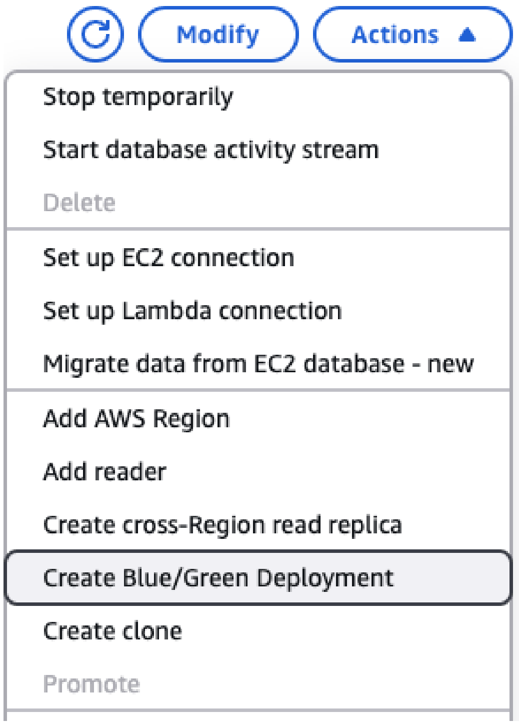

beto-blog

Related

	DB identifier	Status	Role	Engine	Region ...	Size	Recommendations	CPU	Current ...	Mainten...
●	beto-blog	Available	Regional cl...	Aurora My...	us-east-1	3 instances		-	-	none
○	beto-blog-instance-1	Available	Writer inst...	Aurora My...	us-east-1a	db.r7g.large		6.17%	0.02 ses	none
○	beto-blog-instance-1-us-east-1c	Available	Reader inst...	Aurora My...	us-east-1c	db.r7g.large		5.50%	0.00 ses	none
○	beto-blog-instance-2	Available	Reader inst...	Aurora My...	us-east-1d	db.r5.large		19.72%	0.00 ses	none

We can create a Blue/Green environment with the aforementioned environment in mind. This process will establish a new environment that mirrors the structure of the production environment. Please note that to create a Blue/Green environment, the cluster parameter must have `binlog_format` set to `row`.

To create the Blue/Green environment, use the dropdown menu for actions and select the “Create Blue/Green Deployment” button.


Modify
Actions
Stop temporarily
Start database activity stream
Delete
Set up EC2 connection
Set up Lambda connection
Migrate data from EC2 database - new
Add AWS Region
Add reader
Create cross-Region read replica
Create Blue/Green Deployment
Create clone
Promote

Set the identifier of the deployment:

Before creating the Blue/Green environment, please review the configuration carefully. Remember that you can change the Aurora version during the creation process. This option allows for validation checks before the upgrade since all data will be replicated in the green environment.

After creation, please wait until all nodes in the green environment are available.

Having the nodes available allows us to begin the modification process. However, changing the column data type in a replica environment may lead to replication issues. To avoid this, since we are increasing the column type from `int` to `bigint`, we need to set the value of `replica_type_conversions` in the cluster parameter group. We will set this variable to `ALL_NON_LOSSY` (https://dev.mysql.com/doc/mysql-replication-excerpt/8.0/en/replication-features-different-data-types.html).

It's important to take note of the following bug report: https://bugs.mysql.com/bug.php?id=82599, which could cause issues when using unsigned data. Once we have completed the necessary steps, we can proceed with all the required alterations.

Stopping the replica:

```
MySQL
MySQL - Green > CALL mysql.rds_stop_replication;
+-----------------------------+
| Message                     |
+-----------------------------+
| Replica is down or disabled |
+-----------------------------+
1 row in set (1.02 sec)
Query OK, 0 rows affected (1.02 sec)
```

Consider the following table definitions for the changes:

```
MySQL - Green > SHOW CREATE TABLE joinitG
*************************** 1. row ***************************
       Table: joinit
Create Table: CREATE TABLE `joinit` (
  `i` int NOT NULL AUTO_INCREMENT,
  `s` varchar(64) DEFAULT NULL,
  `t` time NOT NULL,
  `g` int NOT NULL,
  PRIMARY KEY (`i`),
  KEY `g_fk` (`g`),
  CONSTRAINT `joinit_ibfk_1` FOREIGN KEY (`g`) REFERENCES `joinit_fk` (`i`) ON DELETE
) ENGINE=InnoDB AUTO_INCREMENT=4653058 DEFAULT CHARSET=latin1
1 row in set (0.02 sec)

MySQL - Green > SHOW CREATE TABLE joinit_fkG
*************************** 1. row ***************************
       Table: joinit_fk
Create Table: CREATE TABLE `joinit_fk` (
  `i` int NOT NULL,
  PRIMARY KEY (`i`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
1 row in set (0.01 sec)
```

Doing the alter change:

```mysql
MySQL - Green > ALTER TABLE joinit DROP FOREIGN KEY joinit_ibfk_1;
Query OK, 0 rows affected (0.03 sec)
Records: 0 Duplicates: 0 Warnings: 0

MySQL - Green > ALTER TABLE joinit MODIFY COLUMN g bigint NOT NULL;
Query OK, 4194391 rows affected (1 min 39.09 sec)
Records: 4194391 Duplicates: 0 Warnings: 0

MySQL - Green > ALTER TABLE joinit_fk MODIFY COLUMN i bigint NOT NULL;
Query OK, 60 rows affected (0.09 sec)
Records: 60 Duplicates: 0 Warnings: 0

MySQL - Green > ALTER TABLE joinit ADD CONSTRAINT joinit_ibfk_1 FOREIGN KEY (`g`) REFE
Query OK, 4194391 rows affected (1 min 45.66 sec)
Records: 4194391 Duplicates: 0 Warnings: 0
```

After completing the process, we can start the replication and wait for it to catch up.

```mysql
MySQL - Green > CALL mysql.rds_start_replication;
+---------------------------+
| Message                   |
+---------------------------+
| Replica running normally. |
+---------------------------+
1 row in set (2.18 sec)
Query OK, 0 rows affected (2.18 sec)

MySQL - Green > SHOW REPLICA STATUSG
*************************** 1. row ***************************
             Replica_IO_State: Waiting for source to send event
                  Source_Host: 10.1.15.0
                  Source_User: rdsrepladmin
                  Source_Port: 3306
                Connect_Retry: 60
              Source_Log_File: mysql-bin-changelog.000004
          Read_Source_Log_Pos: 112774
               Relay_Log_File: relaylog.000002
                Relay_Log_Pos: 82470
        Relay_Source_Log_File: mysql-bin-changelog.000004
           Replica_IO_Running: Yes
          Replica_SQL_Running: Yes
[...]
        Seconds_Behind_Source: 0
[...]
1 row in set (0.01 sec)
```

Maintaining the replica in sync will enable the switchover; it's also highly recommended that you stop the application writes during the switchover procedure.

By selecting the deployment option from the action button in the top right, we can access the switchover button action:

After clicking, a panel will appear to switch over:

Now, we just need to wait for the switchover:

After the procedure is complete, we can see that the green nodes, which are the ones where we changed the columns, are now referred to as “new-blue.” These nodes already have the endpoint used by the application. The blue environment is now called “old-blue”:

Once completed, we can perform any needed validation before deleting the nodes. The application can now return to regular operation, and after the validations are done, the deployment and old blue versions can be deleted.

Conclusion

The switchover procedure is a common practice used for various purposes; however, as previously mentioned, it cannot be accomplished using standard RDS replication. This limitation arises because RDS replication does not use the default asynchronous replication method found with MySQL’s binary log. Therefore, adopting a Blue/Green deployment strategy is a viable solution.

For further reads regarding blue/green deployments, please refer to https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/blue-green-deployments.html
Also, please be familiar with the limitations of the blue/green deployments. You may see those on the following link:
https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/blue-green-deployments-considerations.html

FAQ

Is there any rollback?

Unfortunately, the Blue/Green Environment has no built-in rollback feature. However, it is possible to perform a manual rollback, as demonstrated in the following AWS blog post: https://aws.amazon.com/blogs/database/implement-a-rollback-strategy-after-an-amazon-aurora-mysql-blue-green-deployment-switchover/

Can I do MySQL upgrades with it?

Yes, using the Blue/Green for the MySQL upgrades is possible.

What happens if any error on the replica happens?

In case of a replica error, the switchover will not happen, and checking on the ***Log & events*** tab on the deployment, you'll be able to see a similar error message: "Switchover from DB cluster beto-blog to beto-blog-green-usht2d was canceled due to replication errors on beto-blog-green-usht2d. Correct the replication errors and then switch over."

Email

Subscribe

Related Blog Articles

RECOMMENDED ARTICLES

May 29, 2025

Arunjith
Aravinc

How to Safely Upgrade InnoDB Cluster From MySQL 8.0 to 8.4

Insight for DBAs

MySQL

May 28, 2025

Andrey

The Open Source Ripple Effect: How Valkey Is Redefining the Future of Caching, and Why

MOST POPULAR ARTICLES

June 20, 2023

Sergey P

Deploy Django on Kubernetes With Percona Operator for PostgreSQL

Cloud

Insight for Developers

Percona Software

PostgreSQL

December 28, 2012

Miguel A
Nieto

Auditing Login attempts in MySQL

MySQL

It Matters

Insight for DBAs

Open Source

Valkey

May 27, 2025 David (

Beyond Guesswork: Enterprise-Grade PostgreSQL Tuning with pg_stat_statements

Insight for DBAs

PostgreSQL

May 16, 2016 Muhamm Irfan

MySQL “Got an error reading communication packet”

MySQL

Ready to get started?

Subscribe to our newsletter for updates on enterprise-grade open source software and tools to keep your business running better.

Email

First Name

Last Name

Company Name

SUBMIT

Follow us for updates

Featured

MySQL 5.7 End of Life

Learn

Downloads

Resources

Products

MySQL

MongoDB

PostgreSQL

Cloud Native

Monitoring

Services

Support

Managed Services

Consulting

Policies

Training

Docs

Percona Toolkit

Use Cases

Emergency Support

High-Traffic Events

Performance Tuning

Migrations

Upgrades

Security

Discover

Blog

Community

Percona Community Forum

Events

About

About Percona

Partners

In The News

Careers

Terms of Use Privacy Copyright Legal Security Center