MariaDB Enterprise & MariaDB Enterprise Cluster

MariaDB Enterprise Cluster in Azure - Test Drive

Contents

About the Test Drive **2**

What is MariaDB? **2**

About MariaDB Enterprise Cluster in Azure **2**

Architecture 2

Objective 2

Getting Started **3**

Configure WordPress 3

Administering MariaDB Enterprise Cluster 3

Connecting to MaxScale 3

Scenario 1: Taking one MaxScale node offline 4

Scenario 2: Taking one MariaDB Cluster data node offline 5

About the Test Drive

Welcome to the MariaDB Enterprise Cluster + MariaDB MaxScale test drive in Microsoft Azure!

In this 2 hour test drive, we will dive in and see what MariaDB Enterprise Cluster and MariaDB MaxScale can do. This guide walks you through the steps on how the High Availability (HA) properties of the MariaDB Enterprise Cluster + MariaDB MaxScale solution work in practice.

What is MariaDB?

MariaDB is the fastest growing Open Source database with more than 12 million users worldwide. It is the database of choice to power applications at companies like booking.com, HP, Virgin Mobile and Wikipedia. MariaDB is secure at the enterprise-level, highly reliable and trusted by the world's leading brands. Its extensible, modern architecture at every layer in the database allows for flexible configuration that supports both traditional and emerging enterprise use cases.

About MariaDB Enterprise Cluster in Azure

MariaDB Enterprise Cluster extends MariaDB, the widely adopted, MySQL-compatible open source database with Galera clustering technology. MariaDB MaxScale offers connection- and statement-based load balancing.

The MariaDB Enterprise Cluster + MariaDB MaxScale solution for Azure consists of a 3-node MariaDB Enterprise Cluster and dual MariaDB MaxScale nodes in a Highly Available (HA) configuration.

Architecture

- Developers and DBAs can now provision a 3-node production-ready MariaDB Enterprise Cluster with MariaDB MaxScale®.
- MariaDB Enterprise Cluster with Galera technology is a multi-master cluster that achieves high availability and scalability through features such as synchronous replication, read and write to any cluster nodes without slave lag, automatic membership control of new nodes joining the cluster and failed nodes dropping from the cluster.
- MariaDB MaxScale is a database gateway that insulates client applications from the complexities of a database cluster.
- In addition to load balancing, MariaDB MaxScale also provides logging, filtering, and monitoring mechanisms.
- An Azure load balancer is placed in front of the MaxScale VMs to achieve high availability.
- The VMs can be accessed through the load balancer's public IP or DNS.

Objective

The MariaDB Enterprise Cluster + MariaDB MaxScale offering in Microsoft Azure deploys 3 data/Galera nodes and 2 MariaDB MaxScale nodes, all running on CentOS 7. MariaDB MaxScale in this solution is automatically configured to provide three services: RW Split Router (Read/Write Split) on port 4006, Write Connection Router (to the "Master" node) on port 4007, and Read Connection Router (to the "Slave" nodes) on port 4008.

A Client VM is also part of this deployment. WordPress has been installed and configured to talk to the cluster deployed as part of the test drive.

The 2 MaxScale nodes are set up behind an Azure Load Balancer, configured to automatically fail over if one instance of MaxScale becomes unavailable.

This test drive gives you an opportunity to see MariaDB Enterprise Cluster in action and see first-hand some of its High Availability (HA) features.

Getting Started

- Once you have signed in to Microsoft Test Drive Portal you will see the launch page. Now you are ready to launch the test drive.
- It takes from 2 minutes to 15 minutes to deploy the test drive.
- Once the test drive is Active, DNS names, URLs, and Access credentials will be shown on the web page under "Access information" as well as sent by email.
  - The time remaining in the test drive is shown on the test drive launch page.

Configure WordPress

- Open the WordPress URL found in Access Information in a web browser
- Fill in the details and click "Install WordPress"
- Log in to your WordPress account with your WordPress Credentials
- Create some posts, view them, etc., to ensure that WordPress is set up and functioning

Administering MariaDB Enterprise Cluster

To demonstrate the High Availability of MariaDB, we have designed two scenarios. Scenario 1: Taking one MaxScale node offline Scenario 2: Taking one MariaDB Cluster data node offline

Before working through those scenarios, you'll need to follow the steps in Connecting to the cluster, below.

Connecting to MaxScale

You can use SSH to connect directly to the MaxScale nodes from any host on the Internet. Because there are 2 MaxScale nodes, and both are accessed through the same public-facing DNS name, non-standard ports are used for SSH. To SSH to MaxScale VM1 (max1), use port 2201. To SSH to MaxScale VM 2 (max2), use port 2202.

First, you'll use SSH to connect to MaxScale VM 1.

For Windows:

You will need to have downloaded and installed the PuTTY tool. PuTTY is free and open source, and it can be obtained from http://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html.

You will need to download the **Windows SSH key** (.pek) file that will give you login access to the VM. The link to the key is provided in the "Access information" section on the test drive launch page after the test drive has become "Active". A link is also sent to you via email.

Copy and paste the key URL into a new web browser window, which will download the key. Save the key to your Desktop, which can be used during with PuTTY to SSH to the MaxScale VMs.

Launch putty.exe and connect to MaxScale VM 1 using the hostname provided in the test drive launch page and port 2201. Use the PuTTY SSH key (.ppk) file provided on test drive launch page.

For Linux or macOS:

Download the **OpenSSH key** (.pem) file from the link provided on the test drive launch page.

Open the terminal, `cd` to the location where you saved the .pem file, and execute this command:

```
chmod 400 mariadb_privatekey.pem
```

Execute this command, in the terminal, in the directory where the private key (.pem) file is located, using the hostname for MaxScale VM 1 provided on the test drive launch page.

```
ssh -i mariadb_privatekey.pem -p 2201 -l mdbe <hostname>
```

Scenario 1: Taking one MaxScale node offline

Check the status of MaxScale by executing `sudo maxadmin show servers`. You should see output like this:

```
[mdbe@mdbec-max1 ~]$ sudo maxadmin show servers
Server 0x228e640 (db1)
        Server:                             10.0.1.4
        Status:                             Master, Synced, Running
        Protocol:                           MySQLBackend
        Port:                               3306
        Server Version:                     10.1.19-MariaDB-enterprise
        Node Id:                            0
        Master Id:                          -1
        Slave Ids:
        Repl Depth:                         0
        Number of connections:              0
        Current no. of conns:               0
        Current no. of operations:          0
Server 0x228e160 (db2)
        Server:                             10.0.1.5
        Status:                             Slave, Synced, Running
        Protocol:                           MySQLBackend
        Port:                               3306
        Server Version:                     10.1.19-MariaDB-enterprise
        Node Id:                            1
        Master Id:                          -1
        Slave Ids:
        Repl Depth:                         0
        Number of connections:              0
        Current no. of conns:               0
        Current no. of operations:          0
Server 0x228dc80 (db3)
        Server:                             10.0.1.6
        Status:                             Slave, Synced, Running
        Protocol:                           MySQLBackend
        Port:                               3306
        Server Version:                     10.1.19-MariaDB-enterprise
        Node Id:                            2
        Master Id:                          -1
        Slave Ids:
        Repl Depth:                         0
        Number of connections:              0
        Current no. of conns:               0
        Current no. of operations:          0
```

Stop the MaxScale service by executing `sudo systemctl stop maxscale`.

To ensure that no process is listening on the ports after the service has been stopped, execute `sudo netstat -ntalp | grep -E '4006|4007|4008'`.

Even though one instance of MaxScale has gone offline, clients can still interact with the cluster via the Azure load balancer, which will automatically send traffic to the other MaxScale instance. You can see this by continuing to interact with WordPress even while this instance of MaxScale is offline.

You can also use the local MariaDB client program (`mysql`) to connect to the the cluster via the other MaxScale instance:

```
mysql -h mdbec-max2 -P 4006 -u myapp -pMariaDB123 mydb
```

You can start the service again by executing `sudo systemctl start maxscale`.

Scenario 2: Taking one MariaDB Cluster data node offline

The cluster's High Availability (HA) properties extend beyond the MaxScale nodes to the MariaDB Galera Cluster data nodes themselves. To show this, we'll stop one of the nodes in the running cluster and observe that the rest of the cluster stays online and available to the application.

To do this, you should first SSH to MaxScale VM 1 following the instructions above.

Check the status of the servers in the cluster by executing `sudo maxadmin show servers`. You should see that all 3 servers are Synced and Running, though your instance of MaxScale may have assigned the "Master" and "Slave" roles to different backend nodes than in this example:

```
[mdbe@mdbec-max1 ~]$ sudo maxadmin show servers
Server 0x1f7e640 (db1)
        Server:                             10.0.1.4
        Status:                             Master, Synced, Running
        Protocol:                           MySQLBackend
        Port:                               3306
        Server Version:                     10.1.19-MariaDB-enterprise
        Node Id:                            0
        Master Id:                          -1
        Slave Ids:
        Repl Depth:                         0
        Number of connections:              0
        Current no. of conns:               0
        Current no. of operations:          0
Server 0x1f7e160 (db2)
        Server:                             10.0.1.5
        Status:                             Slave, Synced, Running
        Protocol:                           MySQLBackend
        Port:                               3306
        Server Version:                     10.1.19-MariaDB-enterprise
        Node Id:                            1
        Master Id:                          -1
        Slave Ids:
        Repl Depth:                         0
        Number of connections:              0
        Current no. of conns:               0
        Current no. of operations:          0
Server 0x1f7dc80 (db3)
        Server:                             10.0.1.6
        Status:                             Slave, Synced, Running
        Protocol:                           MySQLBackend
```

```
        Port:                                3306
        Server Version:                      10.1.19-MariaDB-enterprise
        Node Id:                             2
        Master Id:                           -1
        Slave Ids:
        Repl Depth:                          0
        Number of connections:               0
        Current no. of conns:                0
        Current no. of operations:           0
```

Now, shut down one of the backend nodes. You can shut down any of the nodes, even the one marked "Master", by executing `mysqladmin -h mdbec-db1 shutdown`, replacing "db1" with either "db2" or "db3" if you wish to shut down another node. Note: the privilege to perform this type of remote shut down is put in place specially for this demo — it would not be advisable to create a scenario in production where this was possible in this way.

Now, execute `sudo maxadmin show servers` again and you should see that the node you shut down is marked as "Down":

```
[mdbe@mdbec-max1 ~]$ sudo maxadmin show servers
Server 0x1f7e640 (db1)
        Server:                              10.0.1.4
        Status:                              Down
        Protocol:                            MySQLBackend
        Port:                                3306
        Server Version:                      10.1.19-MariaDB-enterprise
        Node Id:                             -1
        Master Id:                           -1
        Slave Ids:
        Repl Depth:                          0
        Number of connections:               0
        Current no. of conns:                0
        Current no. of operations:           0
Server 0x1f7e160 (db2)
        Server:                              10.0.1.5
        Status:                              Master, Synced, Running
        Protocol:                            MySQLBackend
        Port:                                3306
        Server Version:                      10.1.19-MariaDB-enterprise
        Node Id:                             0
        Master Id:                           -1
        Slave Ids:
        Repl Depth:                          0
        Number of connections:               0
        Current no. of conns:                0
        Current no. of operations:           0
Server 0x1f7dc80 (db3)
        Server:                              10.0.1.6
        Status:                              Slave, Synced, Running
        Protocol:                            MySQLBackend
        Port:                                3306
        Server Version:                      10.1.19-MariaDB-enterprise
        Node Id:                             1
        Master Id:                           -1
        Slave Ids:
        Repl Depth:                          0
```

```
        Number of connections:              0
        Current no. of conns:               0
        Current no. of operations:          0
```

You can still use the MariaDB client (`mysql`) to connect to the cluster, even though one node is down:

```
mysql -P 4006 -u myapp -pMariaDB123 mydb
```

You can also confirm that the cluster is still online by interacting with WordPress.

Bringing the data node back online requires connecting to the backend node using SSH, which is beyond the scope of this tutorial.