How to move the MySQL Database from the 'system-mysql' pod to an external one using the default 'amp.yml' template?

SOLUTION VERIFIED - Updated Friday at 4:02 PM - English

Environment

- Red Hat 3scale API Management
  - 2.X On-premises

Issue

We realized that it's better to externalize our MySQL Database for `High Availability (HA)`. We aren't using the `Tech Preview` templates but the default `amp.yml` one. How to proceed?

Resolution

Disclaimer: *This KCS Article has been released as **Private (Red Hat Internal)** and should not be available to the customers, since it details a procedure that **has not yet been officially validated or supported.** That being said, it could be helpful on certain situations where the customer is facing several infrastructure (e.g. network and/or filesystem) issues using the default `system-mysql` pod. The Red Hat Associate reading this Article should use his/her best judgement to decide whether the benefits of applying an unsupported procedure are higher than the risks of leaving the `MySQL` Database not externalized.*

NOTE: The difference between this approach and the one described in High Availability and Evaluation templates is that it provides a way for externalizing a `MySQL` Database in case Red Hat 3scale API Management was initially deployed using the default `amp.yml` template.

Limitations

There are a few current limitations with this process listed below:

Red Hat 3scale On-premises versions

It has only been tested and verified on the `2.5 On-premises` and `2.6 On-premises` versions from Red Hat 3scale API Management. It's possible that it could work on previous versions, however this has not been tested yet.

MySQL Database User

Although there is a `mysql2://` formatted `URL` that would supposedly allow for any combination of `username` and `password`, the user `'root'@'%'` should always be used otherwise it will fail. The reason behind it is that some sections from Red Hat 3scale are still hardcoded to use the `'root'@'%'` user and this issue still needs to be addressed in the future.

MySQL Host

Always use the `IP Address` from the external `MySQL` Database instead of the `hostname`, otherwise the latter might not be resolved. For instance, use `1.1.1.1` instead of `mysql.mydomain.com`.

System Database

The remote `MySQL` Server must not have a currently existing Database named `system`.

Instructions

Follow the steps below to fully externalize the `MySQL` Database. Please notice that they will cause a full `downtime` in the environment while the process is executed:

1. Login to the OpenShift Node where Red Hat 3scale `2.X On-premises` is hosted and change to its project:

```
$ oc login -u <USER> <URL>

$ oc project <PROJECT>
```

   NOTE: Replace `<USER>`, `<URL>` and `<PROJECT>` accordingly.

2. Use the instructions provided on the **Stop Red Hat 3scale On-premises** section from the Article How to restart Red Hat 3scale On-premises? to scale down all the pods using the correct order to avoid any loss of data.

3. The following command should now return `No resources found`:

```
$ oc get pod
```

4. Scale up the Database level pods again:

```
$ oc scale dc/{backend-redis,system-memcache,system-mysql,system-redis,zync-
database} --replicas=1
```

5. Ensure that you are able to login to the external `MySQL` Database through using the `system-mysql` pod before proceeding with the next steps:

```
$ oc rsh system-mysql-<XYZ>

$ mysql -u root -p -h <HOST>
```

NOTE: Replace `<XYZ>` with the proper `id` of the `system-mysql` pod and `<HOST>` with the IP Address from the external `MySQL` Database. The user should be always `root` (more information on this topic in the **Limitations** section above).

6. The expected result from the step above is a `mysql>` console displaying. Type `exit` twice and go back to the OpenShift Node console.

7. Perform a full `MySQL dump`

```
$ oc rsh system-mysql-<XYZ> /bin/bash -c "mysqldump -u root --single-transaction --
routines --triggers --all-databases" > system-mysql-dump.sql
```

NOTE: Replace `<XYZ>` with the proper `id` from the `system-mysql` pod and validate that the file `system-mysql-dump.sql` contains a valid `MySQL` level dump generated data as in the following example:

```
$ head -n 10 system-mysql-dump.sql
-- MySQL dump 10.13  Distrib 5.7.24, for Linux (x86_64)
--
-- Host: localhost    Database:
-- ------------------------------------------------------
-- Server version   5.7.24

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8 */;
```

8. Scale down the `system-mysql` pod and leave it to `0 replicas`:

```
$ oc scale dc/system-mysql --replicas=0
```

9. Find the `base64` equivalent of the URL `mysql2://root:<PASSWORD>@<HOST>/system`, replacing `<PASSWORD>` and `<HOST>` accordingly:

```
$ echo "mysql2://root:<PASSWORD>@<HOST>/system" | base64
```

10. Create a default `'user'@'%'` on the remote `MySQL` Database (it only needs to have `SELECT` privileges) and also find its `base64` equivalents:

```
$ echo "user" | base64

$ echo "<PASSWORD>" | base64
```

NOTE: Replace `<PASSWORD>` with the proper `password` for `'user'@'%'`.

11. Perform a backup and edit the OpenShift secret `system-database`:

```
$ oc get secret system-database -o yaml > system-database-orig.bkp.yml

$ oc edit secret system-database
```

`URL`: Replace it with the value from the **Step 9**.
`DB_USER` and `DB_PASSWORD`: Use the values from the previous step for both.

12. Send the file `system-mysql-dump.sql` to the remote database Server and import the dump into it. Following below is an **example** shell command to import it:

```
$ mysql -u root -p < system-mysql-dump.sql
```

13. Ensure that a new Database called `system` was created:

```
$ mysql -u root -p -se "SHOW DATABASES"
```

14. Follow the instructions provided on the **Start Red Hat 3scale On-premises** section from the Article How to restart Red Hat 3scale On-premises? until the `system-app` pod is scaled up (don't scale up any pods after `system-app` yet).

15. The expected scenario is the `system-app` pod up and running without any issues. After it has been validated, continue scaling back up the other pods mentioned on the Article above.

16. After everything has been properly validated, the `system-mysql` DeploymentConfig object should be backed up and deleted after a few days to avoid any confusion on the future.

Rollback

A `rollback` procedure should be performed if on the **Step 14** from the **Instructions** above the `system-app` pod is not fully back online and the root cause for it could not be determined or addressed:

1. Edit the secret `system-database` again, using the original values from the file `system-database-orig.bkp.yml` generated on the **Step 11** from **Instructions**:

```
$ oc edit secret system-database
```

NOTE: Replace `URL`, `DB_USER` and `DB_PASSWORD` with their original values.

2. Scale down all the pods and then up back again, including the `system-mysql` one.
   The `system-app` pod and the others to be started after it should be up and running again.

SBR API Mgmt **Product(s)** Red Hat 3scale API Management **Category** Customize or extend

Tags api-mgmt database high_availability mysql on-premises

This solution is part of Red Hat's fast-track publication program, providing a huge library of solutions that Red Hat engineers have created while supporting our customers. To give you the knowledge you need the instant it becomes available, these articles may be presented in a raw and unedited form.