Red Hat OpenStack Platform 9 Database-as-a-Service Guide

Database-as-a-Service for Red Hat OpenStack Platform

OpenStack Team

Red Hat OpenStack Platform 9 Database-as-a-Service Guide

Database-as-a-Service for Red Hat OpenStack Platform

OpenStack Team
rhos-docs@redhat.com

Legal Notice

Abstract

This guide describes how to deploy, configure, and use Database-as-a-Service, trove, with Red Hat OpenStack Platform.

Table of Contents

PREFACE

Database-as-a-Service (DBaaS) enables users to conveniently utilize a relational or non-relational database in their OpenStack environment without having to worry about the administrative tasks such as deployment, configuration, or backups. This functionality is provided by the *trove* component, which provisions and manages guest database images.

> **Warning**
>
> **DEPRECATION NOTICE**: Beginning in Red Hat OpenStack Platform 10, the OpenStack Trove service will no longer be included in the Red Hat OpenStack Platform distribution. We are working with a trusted partner to provide our customers with a production ready DBaaS service. Please contact your sales account manager to learn more about this option.

DBaaS comprises several services:

- Guest Agent service, running on guest instances and managing the datastore; this includes bringing a new datastore online and performing database operations.
- Task Manager service, which provisions, manages, and performs operations on instances.
- API service, which provides a RESTful API and forwards API requests to the guest agent or task manager. Command-line interface to the DBaaS API is provided by the **`trove`** client, which is part of the **`python-troveclient`** package.
- Conductor service, running on the host and listening for messages from guest instances to pass to the host; for example, status updates.

The following database platforms are supported:

- MySQL 5.5 and 5.6
- MariaDB 5.5 and 10

> **Warning**
>
> The OpenStack Database-as-a-Service is available in this release as a **Technology Preview**, and therefore is not fully supported by Red Hat. It should only be used for testing, and should not be deployed in a production environment. For more information about Technology Preview features, see Scope of Coverage Details.

CHAPTER 1. ADMINISTERING DATABASE-AS-A-SERVICE

1.1. INSTALL AND CONFIGURE THE DATABASE-AS-A-SERVICE CONTROLLER

For detailed instructions on how to install the necessary packages and set up the environment on the DBaaS controller, see the Manual Installation Procedures.

1.2. SECURE DATABASE-AS-A-SERVICE

In earlier versions of Red Hat OpenStack Platform, the DBaaS guest agent ran with credentials to the RabbitMQ message bus which was used by the entire management controller for all its traffic. Consequently, the DBaaS instance could view all the data sent over the message bus, including sensitive data, which could allow the instance to misuse or attack other services sharing the bus. With Red Hat OpenStack Platform 9, DBaaS is placed into an isolated tenant which does not have access to the rest of the management controller. However, to further increase the security of your DBaaS, you can set up the DBaaS instance on a separate host and a separate RabbitMQ message bus. To do so, perform the following steps.

1. On the management controller:

    1. Set the RabbitMQ user and password to custom values which will not be shared with the DBaaS tenant. For example, you can use the **`guest`** account and the **`/`** default virtual host with a changed password:

        ```
        # rabbitmqctl change_password guest password
        ```

        The password will be changed in all the configuration files where services are set to connect to the message bus.

    2. Update the DBaaS database endpoint to make service catalog lookups use DBaaS on the separate host. First determine the current endpoint UUID, then delete it, and finally create a new endpoint with the IP address of the separate DBaaS host:

        ```
        # keystone endpoint-list
        # keystone endpoint-delete current_DBaaS_endpoint_uuid
        # keystone endpoint-create --service trove --publicurl
        http://IP:8779/v1.0/\$\(tenant_id\)s --region RegionOne
        ```

2. On the DBaaS controller:

    1. Point the authentication token filter in the WSGI configuration at the remote management controller host. Use the following options and values in the **`/etc/trove/api-paste.ini`** file, replacing *IP* with the IP address of the management controller host:

        ```
        [filter:authtoken]
        paste.filter_factory =
        keystonemiddleware.auth_token:filter_factory
        service_protocol = http
        service_host = IP
        ```

```
service_port = 5000
auth_host = IP
auth_port = 35357
auth_protocol = http
auth_uri = http://IP:35357/v2.0/
signing_dir = /tmp/keystone-signing-trove
admin_tenant_name = admin
admin_user = admin
admin_password = admin
```

2. Add an admin user to ensure that the DBaaS guest agent has different secrets than those used in the management controller:

```
# rabbitmqctl add_user isolated isolated
# rabbitmqctl set_permissions isolated ".*" ".*" ".*"
```

3. Configure the DBaaS control plane to connect to and authenticate against the local RabbitMQ instance by placing the following options and values into the **/etc/trove/trove.conf** file:

```
# AMQP Connection info
rabbit_userid=isolated
rabbit_password=isolated
rabbit_host=DBaaS_IP

# single instance tenant
nova_proxy_admin_user = user
nova_proxy_admin_pass = password
nova_proxy_admin_tenant_name = tenant
trove_auth_url = http://MGMT_IP:5000/v2.0
nova_compute_service_type = compute
cinder_service_type = volumev2
os_region_name = RegionOne
nova_compute_url = http://MGMT_IP:8774/v3

remote_nova_client =
trove_ext.cloudos.remote.nova_client_trove_admin
remote_cinder_client =
trove_ext.cloudos.remote.cinder_client_trove_admin
remote_neutron_client =
trove_ext.cloudos.remote.neutron_client_trove_admin
```

3. On the management controller, edit the **/etc/trove/trove-guestagent.conf** file as follows:

```
# AMQP Connection info
rabbit_userid=isolated
rabbit_password=isolated
rabbit_host=DBaaS_IP

# single tenant config
nova_proxy_admin_user = user
```

```
nova_proxy_admin_pass = password
nova_proxy_admin_tenant_name = tenant
trove_auth_url = http://MGMT_IP:5000/v2.0
swift_url = http://MGMT_IP:8080/v1/AUTH_
```

1.2.1. ACL Policies

Alternatively, you can improve the security by taking advantage of the ACL policies provided by RabbitMQ virtual hosts. In this case, you can assign separate users to each virtual host, including separate permissions. For example, you can define a virtual host called **/isolated** and assign the **isolated** user appropriate permissions for the virtual host. To do so, perform the following steps:

1. Change the **guest** account password for the default **/** virtual host:

```
# rabbitmqctl change_password guest password
```

2. Add the new user and virtual host and set the permissions appropriately:

```
# rabbitmqctl add_user isolated isolated
# rabbitmqctl add_vhost /isolated
# rabbitmqctl -p /isolated set_permissions isolated ".*" ".*"
".*"
```

3. Edit the DBaaS configuration. There are two files to edit: **/etc/trove/trove-guestmanager.conf**, which is generated by *packstack* from *puppet-trove* and is to be injected into the guest instance, and **/etc/trove/trove-conductor.conf**, which configures the DBaaS control plane service for asynchronous status updates from the DBaaS instance. The RabbitMQ settings are identical for both files:

```
[oslo_messaging_rabbit]
rabbit_host=IP
rabbit_virtual_host=/isolated
rabbit_userid=isolated
rabbit_password=isolated
rabbit_port=5672
rabbit_ha_queues=False
rabbit_hosts=IP:5672
rabbit_use_ssl=False
```

The advantage of this approach is its simplicity in comparison with the scenario where separate hosts and message queues are used. This can be useful if provisioning is performed in an automated manner; for example, using Red Hat OpenStack Platform director.

CHAPTER 2. USING DATABASE-AS-A-SERVICE

2.1. AUTOMATE THE CREATION AND CONFIGURATION OF DATABASE-AS-A-SERVICE GUEST IMAGES

Red Hat recommends using the **`trove-image-create`** tool for generating automated DBaaS-compatible images for supported datastores.

To get the **`trove-image-create`** tool, install the **`openstack-trove-images`** package:

```
# yum install openstack-trove-images
```

The following basic options are available:

Option	Description, parameters
`-i`, **`--image`**	The base image that you are going to use. QEMU images (qcow2) are supported. Specify the image file name (and, optionally, its path) as a pamaterer.
`-r`, **`--release`**	The OpenStack version you want to use. Specify **`kilo`**, **`liberty`**, or **`mitaka`** as a parameter.
`-s`, **`--datastore`**	The datastore you want to deploy. Supported datastores are listed in the Preface, and potential parameters are: - **`mysql`** — whatever in the distribution provides **`mysql`**. In the case of RHEL 7, MariaDB 5.5 will be used. - **`mysql55`** — MySQL 5.5 from mysql.com - **`mysql56`** — MySQL 5.6 from mysql.com - **`mariadb10`** — MariaDB 10.0 from mariadb.org

Example 2.1. Customize an Image

For example, you can use the tool this way:

```
# trove-image-create -s mysql -r mitaka -i myimage.qcow2
```

This will customize the image stored in the **`myimage.qcow2`** file in the current working directory by adding MariaDB 5.5 and Trove from Red Hat OpenStack Platform 9 (Mitaka) to it.

There are additional options that must be used if you are working with a RHEL 7 image:

Option	Recognized parameters and their syntax	Description
--sm-register	*USER*:**password**:*PASSWORD* *USER*:**file**:*FILE_CONTAINING_PASSWORD*	Register with Subscription Manager using your Red Hat credentials.
--sm-pool	**pool**:*POOL_ID* **file**:*FILE_CONTAINING_POOL_ID* **auto**	Attach a specified or automatically determined subscription pool to the system.

Example 2.2. Customize a RHEL 7 Image

For example:

```
# trove-image-create -s mysql -r mitaka -i ../../images/rhel-
mariadb55.qcow2 --sm-register admin@example.com:password:123456 --
sm-pool auto
```

This will customize the image in a manner similar to the previous example but also register the system using Red Hat login name *admin@example.com*, password *123456*, and the best-matching subscription.

2.1.1. Loading the Image to Database-as-a-Service Management

When the customization of the image is complete, perform the following steps:

1. Upload the image to the Image service. To do so, run a command similar to this one:

```
# openstack image create rhel7-mariadb55 --disk-format qcow2 --
container-format bare --public < myimage.qcow2
```

2. Get the ID of the uploaded image from the output of the previous command, which should look like the following:

```
+------------------+----------------------------------------
-------------+
| Field            | Value
|
+------------------+----------------------------------------
-------------+
| checksum         | dec3f16054739459d03984b7a552cd9c
|
| container_format | bare
|
| created_at       | 2016-01-27T20:10:36Z
|
```

```
| disk_format      | qcow2
|
| file             | /v2/images/c637391b-e00f-47fb-adb5-
e8dfc4e224d4/file |
| id               | c637391b-e00f-47fb-adb5-e8dfc4e224d4
|
| min_disk         | 0
|
| min_ram          | 0
|
| name             | rhel7-mariadb55
|
| owner            | 483cae7de00c4f029e19eef5983c67a9
|
| protected        | False
|
| schema           | /v2/schemas/image
|
| size             | 1910767616
|
| status           | active
|
| updated_at       | 2016-01-27T20:10:46Z
|
| virtual_size     | None
|
| visibility       | public
|
+------------------+-----------------------------------------
-------------+
```

The ID would be *c637391b-e00f-47fb-adb5-e8dfc4e224d4* in this case.

3. Update the DBaaS management datastore to have the record of the new image that will be used to launch instances for the datastore and version you want:

```
# export DATASTORE=mariadb
# export DATASTORE_VERSION=5.5
# export IMAGE_ID=c637391b-e00f-47fb-adb5-e8dfc4e224d4
# export PACKAGES=mariadb-server
#
# trove-manage datastore_update ${DATASTORE} ""
# trove-manage datastore_version_update ${DATASTORE}
${DATASTORE_VERSION} ${DATASTORE} ${IMAGE_ID} ${PACKAGES} 1
# trove-manage datastore_update ${DATASTORE} ${DATASTORE_VERSION}
```

Important

The value of the **PACKAGES** variable depends on the datastore you are using. For MySQL (any version), use **`mysql-community-server`**. For MariaDB 10.0, use **`MariaDB-server`**.

2.1.2. Troubleshooting

In the event of an instance failing on launch, an image with an SSH key can be created and used for troubleshooting. The **`trove-image-create`** tool has the **`--root-ssh-key`** option for this purpose. This option takes the path to a public key as a parameter, and injects the key into the image. For example:

```
# trove-image-create -i myimage.qcow2 -r liberty -s mysql --root-ssh-
key ~/.ssh/id_rsa.pub
```

In order to be able to access an instance based on this image, follow these steps:

1. Edit the security group associated with the instance.
2. Open the SSH port.
3. Consider allowing ICMP as well.
4. If the instance is in a private network, you will need to add a floating IP to the instance.

After completing these steps, you should be able to run the following command to log in to the instance:

```
# ssh root@INSTANCE_IP
```