Abstract

This quick start guides administrators and developers through how to configure, and connect to an external MongoDB Atlas, Crunchy Data Bridge, or CockroachDB database instance using OpenShift Database Access. Red Hat is committed to replacing problematic language in our code, documentation, and web properties. We are beginning with these four terms: master, slave, blacklist, and whitelist. Because of the enormity of this endeavor, these changes will be implemented gradually over several upcoming releases. For more details, see our CTO Chris Wright's message.
# Table of Contents

PREFACE ................................................................................................................. 3  
CHAPTER 1. INSTALLING THE RED HAT OPENSSHIFT DATABASE ACCESS ADD-ON .......... 4  
CHAPTER 2. ACCESSING THE DATABASE ACCESS MENU FOR CONFIGURING AND MONITORING ..... 5  
CHAPTER 3. ACCESSING THE DEVELOPER WORKSPACE AND ADDING A DATABASE INSTANCE ... 7  
CHAPTER 4. CONNECTING AN APPLICATION TO A DATABASE INSTANCE USING THE TOPOLOGY VIEW ... 9  
APPENDIX A. FIND YOUR MONGODB ATLAS ACCOUNT CREDENTIALS .................. 10  
APPENDIX B. FIND YOUR CRUNCHY DATA BRIDGE ACCOUNT CREDENTIALS ............... 12  
APPENDIX C. FIND YOUR COCKROACHDB ACCOUNT CREDENTIALS ...................... 13  
APPENDIX D. FIND YOUR AMAZON RDS ACCOUNT CREDENTIALS ......................... 16  
APPENDIX E. DELETING A DATABASE PROVIDER ACCOUNT .................................... 17  
APPENDIX F. SERVICE BINDING LIBRARIES ..................................................... 18  
APPENDIX G. OPENSHEET DATABASE ACCESS PROVIDER ACCOUNT POLICIES AND USER PERSONAS 19  
APPENDIX H. MANUAL INSTALLATION OF OPENSHEET DATABASE ACCESS ON AZURE RED HAT  
OPENSHEET ........................................................................................................... 22  
H.1. PREREQUISITES ......................................................................................... 22  
H.2. CREATING AN AZURE RED HAT OPENSHEET CLUSTER ............................. 22  
H.3. REGISTERING AN AZURE RED HAT OPENSHEET CLUSTER WITH THE RED HAT HYBRID CLOUD  
CONSOLE .......................................................................................................... 26  
H.4. INSTALLING OPENSHEET DATABASE ACCESS USING THE OPERATOR LIFECYCLE MANAGER 27
The Red Hat OpenShift Database Access is a Service Preview release add-on that enables:

- Easy consumption of database-as-a-service (DBaaS) offerings from partners including MongoDB Atlas, Crunchy Bridge, CockroachDB, and Amazon RDS directly from managed OpenShift clusters.
- Easy management, monitoring and control by administrators of cloud-hosted DBaaS including consumption, usage, and status.

Red Hat OpenShift Database Access (OpenShift Database Access) is a Service Preview release. A Service Preview release contains features that are early in development. Service Preview releases are not production ready and might have features and functionality that are not fully tested. You are advised not to use OpenShift Database Access for production or business-critical workloads.

To provide feedback or inform our engineering team of any technical issues with OpenShift Database Access, please use dbaas-alpha-support@redhat.com.
CHAPTER 1. INSTALLING THE RED HAT OPENSHEET
DATABASE ACCESS ADD-ON

The Red Hat OpenShift Database Access add-on allows you to configure a connection to cloud-database providers, create new database instances, and connect database instances to applications for developers to use.

Procedure

1. Log into the Red Hat Hybrid Cloud Console with your credentials.

2. Click OpenShift from the navigation menu.

3. Click Clusters to display a list of your clusters. Select a cluster name from the list to add database access to.

4. Click Add-ons, and select the Red Hat OpenShift Database Access tile.

5. Click Install.

6. Wait for the installation process to finish. Once the add-on installation completes successfully, a green checkmark appears on the tile.
CHAPTER 2. ACCESSING THE DATABASE ACCESS MENU FOR CONFIGURING AND MONITORING

From the OpenShift console, you can access the OpenShift Database Access navigation menu. On the Database Access page, you can select the correct project namespace for importing a cloud-database provider account.

IMPORTANT

If using MongoDB Atlas as a cloud-database provider, then you must add the IP address of the application pod to MongoDB Atlas’ IP Access List. If the IP address is not in the IP Access List, then a 504 gateway timeout error occurs. Visit the MongoDB Atlas website for more details on adding an IP address to your database project.

Prerequisites

- A service account with either the MongoDB Atlas, or Crunchy Data Bridge, or CockroachDB, or Amazon RDS cloud-hosted database provider.

Procedure

1. Log into the OpenShift console.

2. To select the correct project namespace follow these sub-steps.

   a. Select the Administrator perspective.

   b. Expand the Data Services navigation menu, and click Database Access.

   NOTE

You might need to scroll down the navigation menu.
c. Click the **Project** dropdown menu and then enable the **Show default projects** switch.

d. Type **dbaas** in the search field.

e. Select **redhat-dbaas-operator** or **openshift-dbaas-operator** project namespace.

From the database inventory page you can monitor the database environment, import cloud-hosted database provider accounts, or create new database instances.
CHAPTER 3. ACCESSING THE DEVELOPER WORKSPACE AND ADDING A DATABASE INSTANCE

You can access the developer workspace in the OpenShift console to manage connectivity between database instances and applications.

Prerequisites

- Installation of the OpenShift Database Access add-on.
- Import at least one cloud-database provider account.

Procedure

1. Log into the OpenShift console.

2. Access the developer workspace, and select or create your project, then select a cloud-hosted database provider to add to your project:

   a. Select the Developer perspective 1.

   b. Click +Add 2.

   c. Click the Project dropdown menu 3.

   d. Create a new project or search for your application’s project 4.

   e. Select the Cloud-Hosted Databases tile to connect to a cloud-database provider 5.
3. Select your cloud-hosted database provider tile.

4. Click Add to Topology.

5. Select a previously configured Provider Account for this database instance from the dropdown menu.

6. Select the database instance ID you want to use, and then click Add to Topology.

7. Click Continue. Upon a successful connection, you are taken to the Topology page.
CHAPTER 4. CONNECTING AN APPLICATION TO A DATABASE INSTANCE USING THE TOPOLOGY VIEW

You can add a database to an application by making a connection to the database instance from the cloud-database provider. On the Topology page, you see the application, along with the new database instance.

Procedure

1. When hovering the cursor over the deployment node, you can drag the arrow from the application to the new database instance to create a binding connector. You can also right-click on the deployment node, and click Create Service Binding to create a binding connector.

![Diagram of connecting an application to a database instance](image)

2. On the pop-up dialog, click Create. Once the binding is created, the application pod restarts. After the application pod restarts, your application now has database connectivity.

![Diagram of application and database connectivity](image)

This binding visually represents the injection of database connection information and credentials into the application pod.

3. Use a service binding library based on your application’s framework to consume the service binding information and credentials.

Additional Resources

- See Appendix F, Service binding libraries for more details on service bindings, and working application examples using service binding libraries.
APPENDIX A. FIND YOUR MONGODB ATLAS ACCOUNT CREDENTIALS

You need the Organization ID, the Organization Public Key, and the Organization Private Key to create a provider account resource for MongoDB Atlas.

IMPORTANT

If using MongoDB Atlas as a cloud–database provider, then you must add the IP address of the application pod to MongoDB Atlas’ IP Access List. If the IP address is not in the IP Access List, then a 504 gateway timeout error occurs. Visit the MongoDB Atlas website for more details on adding an IP address to your database project.

Procedure

1. From the MongoDB Atlas home page, Sign In to your account.

2. From your account home page:

   a. Select Organization from the dropdown menu.

   b. Click Settings from the Organization navigation menu.

   c. Copy the Organization ID value.

   

   NOTE

   In some cases your organization ID may be hidden by default.

3. Next, from the account home page:
a. Click Access Manager from the Organization navigation menu.

b. Click API Keys.

c. If you have existing API keys, you can find them listed here. Copy the API public and private keys for the import provider account fields. Also, verify that your API keys have the Organization Owner and Organization Member permissions.

4. If you need new API keys, click Create API Key, and proceed to the next step.

5. On the Create API Key page, enter a Description, and under the Organization Permissions dropdown box select the Organization Owner and Organization Member permissions. Click Next.

6. Copy the API public and private keys for the import provider account fields.
APPENDIX B. FIND YOUR CRUNCHY DATA BRIDGE ACCOUNT CREDENTIALS

You need the Public API Key, and the Private API Secret to create a provider account resource for Crunchy Data Bridge.

Procedure

1. From the Crunch Data Bridge Log in page, sign in to your account.

2. From your personal account home page, click **Settings**, and then click **Settings** from the navigation menu.

3. Copy the **Application ID** and **Application Secret** values for the import provider account fields.
APPENDIX C. FIND YOUR COCKROACHDB ACCOUNT CREDENTIALS

You need the API Key to create a provider account resource for CockroachDB.

**IMPORTANT**

Currently, access to the Service Accounts tab on the Access Management page is enabled by invite only from CockroachDB. To expose the Service Accounts tab on the Access Management page, you can request that this feature be enabled. Contact CockroachDB support and ask for the Cloud API to be enabled in the CockroachDB Cloud Console for your user account.

Additionally, you can view this quick video tutorial from Cockroach Labs on creating an account.

**Procedure**

1. From the CockroachDB service account page, log in to your account.

2. From your service account home page, select Access from the navigation menu.

3. Click Service Accounts from the Access Management page.

4. Click Create Service Account

5. Enter an Account name, select the Permissions, and click Create.
Create service account

6. Enter an **API key name**, and click **Create**.

Create service account

(Optional) Enter a name for the API Key. The name should identify how the API Key will be used.

**API key name**

My API key

7. Copy the **Secret key** for the import provider account field, and click **Done**.
Create service account

The secret key contains the API Key ID and secret. It is used to authenticate the Service Account. The secret key should be stored in a secure location and never shared. Learn more about using the API in the Quickstart.

Secret key

CCDB1_hsP7gqkalO9wwiBsUXCHof_3XIHam2hZHFPgdi2rzRggI...

Copy

⚠️ Copy this secret key now and store it in a secure location. It will not be available to view after you leave this page.

Done
APPENDIX D. FIND YOUR AMAZON RDS ACCOUNT CREDENTIALS

You need an Amazon Web Services (AWS) Access key ID, an AWS Secret access key, and know which AWS Region you are using to import an Amazon Relational Database Service (RDS) provider account for OpenShift Database Access. If you lose your AWS Access key ID, and your AWS Secret access key, new ones must be created.

NOTE

Amazon only allows two secret access keys for each user. You might need to deactivate unused keys, or delete lost keys before you can create a new access key.

IMPORTANT

You are limited to one Amazon RDS provider account per OpenShift cluster. Using your AWS credentials on more than one OpenShift cluster breaks established connections on all OpenShift clusters, except for the last OpenShift cluster that established a connection.

IMPORTANT

OpenShift Database Access only supports RDS database instance deployments, and does not support database cluster deployments.

IMPORTANT

Database instances using a custom Oracle or custom SQL Server engine type are not supported.

Prerequisites

- An Amazon Web Services (AWS) account name.

Procedure

1. Sign in to Amazon’s Identity and Access Management (IAM) console with your AWS user account.

2. From the IAM console home page, expand the Access management menu, and click Users.

3. Select a user from the list.

4. On the user’s summary page, select the Security credentials tab, and click the Create access key button.

5. Copy the AWS Access key ID and the AWS Secret access key.
APPENDIX E. DELETING A DATABASE PROVIDER ACCOUNT

Instead of directly editing your cloud-hosted database provider account information, Red Hat recommends you delete the provider account, and recreate a new one.

Procedure

1. Log into the OpenShift console.
2. Select the Administrator perspective from the navigation menu.
3. Expand the Operators navigation menu, and click Installed Operators.
4. Click OpenShift Database Access Operator from the list of installed operators.
5. Select Provider Account
6. Click the vertical ellipsis for the database provider account you want to delete, and click Delete DBaaSInventory.
7. A dialog box appears to confirm the deletion, click Delete.
8. After deleting the database provider account, you can recreate the database provider account by clicking Create DBaaSInventory.
APPENDIX F. SERVICE BINDING LIBRARIES

The Kubernetes service binding feature was introduced to bring consistency to the way secrets are shared for connecting applications to external services, such as REST APIs, databases, and many other services. OpenShift Database Access leverages the service binding feature to bring a low-touch administrative experience to provisioning, and managing access to external database services. The service binding feature enables developers to connect their applications to database services with a consistent, and predictable experience. Specifically, a service binding creates a volume on the application pod, and organizes the information to make a connection to the database in a directory structure. The volume mount point is exposed as an environment variable. Developer frameworks, such as Quarkus, are service binding aware, and can automatically connect to a database using this exposed workload information without needing to embed database connection information in the application source code.

Here are some application examples on how to use a service binding library:

- Mongo Quarkus application
- Crunchy Postgres Quarkus application
- CockroachDB Postgres Quarkus application
- CockroachDB Postgres Go application
- Crunchy Postgres Python test program
- CockroachDB Postgres Python test program

Additional resources

- See the Kubernetes GitHub project for more details on service bindings.
Policies

Red Hat OpenShift Database Access (RHODA) uses policies to manage access capabilities for the provider account inventories in a given namespace. Access capabilities are predefined by different user personas, and can be used to define multi-tenancy configurations that allow multiple organizations to share a single cluster. Additionally, policies allow you to create strict inventory policies to control access to provider account inventories. A policy’s default values can be overridden on a per-inventory basis.

After installing the OpenShift Database Access add-on, the RHODA operator creates a new Database-as-a-service (DBaaS) policy object in the operator’s installation namespace. By default this namespace is `redhat-dbaas-operator` or `openshift-dbaas-operator`.

The RHODA operator only allows one policy per namespace, and watches for inventory object changes as defined in the policy. The RHODA operator then configures the appropriate access requirements.

Below is an example of the `DBaaSPolicy` object, with the optional `spec` fields. In this example policy, “User1” shares the provider account inventories in their namespace, `user1-project`, with another namespace, `user1-project2`.

Example

```yaml
apiVersion: dbaas.redhat.com/v1alpha1
kind: DBaaSPolicy
metadata:
  name: user1-policy
  namespace: user1-project
spec:
  connectionNamespaces:  
    - user1-project2
  disableProvisions: false
```

1. A list of other namespaces that are allowed a connection to a policy’s inventories. Instead of listing namespaces, using an asterisk surrounded by single quotes (‘*’) allows a connection from all namespaces available in the OpenShift cluster.

2. A user needs at least, the `view` role to see the listed namespaces’ inventories.

3. Disables provisioning in the provider account inventory, defaults to `false`.

Personas

User personas define roles for OpenShift Database Access’ intended audience. Each role has key characteristics that a user can do as they interact with the service. OpenShift Database Access utilizes four personas: Cluster Administrator, Project Administrator, RHODA Service Administrator, and RHODA Developer. These roles start with the highest authority level, the Cluster Administrator, and move down to the lowest authority level, the RHODA Developer.

With the exception of the Cluster Administrator, all other personas are namespace-specific. A user might have different roles within each namespace that they are working in. For example, a user can be a Developer in one namespace, and also be a Project Administrator in another namespace.
Cluster Administrator

A Cluster Administrator is a user with the `cluster-admin` role, and has full access to all resources and namespaces in an OpenShift cluster. Cluster Administrators can do the following:

- Install and upgrade the RHODA operator.
- Assign other users or groups to be operator administrators.

**Command-line Syntax and Examples**

```bash
oc adm policy add-role-to-group admin GROUP_NAME\n   -n NAMESPACE_OF_OPERATOR_INSTALLATION
oc adm policy add-role-to-user admin USER_NAME\n   -n NAMESPACE_OF_OPERATOR_INSTALLATION

$ oc adm policy add-role-to-group admin rhoda-admins \n   -n redhat-dbaas-operator
$ oc adm policy add-role-to-user admin user01 \n   -n redhat-dbaas-operator
```

- Everything a Project Administrator can do.

Project Administrator

A Project Administrator is any user with administrative rights to a specific namespace, and has the `admin` role. Project Administrators can do the following:

- Assign users as additional Project Administrators, RHODA Service Administrators, and RHODA Developers to a specific namespace.

**Command-line Syntax and Examples**

```bash
oc adm policy add-role-to-user admin USER_NAME -n PROJECT_NAMESPACE
oc adm policy add-role-to-user edit USER_NAME -n PROJECT_NAMESPACE
oc adm policy add-role-to-user view USER_NAME -n PROJECT_NAMESPACE

$ oc adm policy add-role-to-user admin user02 -n example-project 1
$ oc adm policy add-role-to-user edit user03 -n example-project 2
$ oc adm policy add-role-to-user view user04 -n example-project 3
```

1. Assign users as additional Project Administrators.
2. Assign RHODA Service Administrators to a specific namespace.
3. Assign RHODA Developers to a specific namespace.

- Everything that a RHODA Service Administrator can do.

RHODA Service Administrator

A RHODA Service Administrator’s rights are a subset of the Project Administrator, and has the `edit` role. A user can be both a Project Administrator, and a RHODA Service Administrator for a specific namespace, and for the cloud-hosted database providers they have credentials for. RHODA Service Administrators can do the following:

- Enable OpenShift Database Access in a namespace
Enable OpenShift Database Access in a namespace.

Set the policy for the namespace.

Import provider accounts for cloud-hosted database providers, and can generate secrets for those providers.

Create `DBaaSInventory`, `DBaaSConnections`, and `DBaaSInstances` objects in a namespace.

Everything that a RHODA Developer can do.

**RHODA Developer**

A RHODA Developer can connect to databases, but is limited by the cloud-hosted database provider accounts accessible to them. RHODA Developers have the view role, and can do the following:

- View specific inventories, and database instances available to them from provider accounts.
- Create their own namespace, where they become the Project Administrator for that new namespace.
- Create connections using `DBaaSConnections`, and `DBaaSInstances` custom resources (CRs) in allowed namespaces. These are namespaces that the user has at least edit rights to.
- Use the Topology View page to make service bindings between applications and databases in allowed namespaces.
- No access to stored secrets in an inventory’s namespace.
- No access to create any objects in an inventory’s namespace.

**Additional resources**

- Kubernetes user-facing role descriptions.
- See Appendix F, Service binding libraries for more information about using service bindings.
APPENDIX H. MANUAL INSTALLATION OF OPENSSHIFT DATABASE ACCESS ON AZURE RED HAT OPENSPLIT

As a systems administrator, you can manually install the OpenShift Database Access add-on to a Red Hat OpenShift cluster running on Microsoft Azure.

H.1. PREREQUISITES

- A Microsoft Azure organizational or user account with an active subscription.
- A Red Hat user account for accessing the Red Hat Hybrid Cloud Console.

H.2. CREATING AN AZURE RED HAT OPENSPLIT CLUSTER

You can manually create Red Hat OpenShift clusters running on Microsoft’s Azure cloud computing service. Once an Azure Red Hat OpenShift (ARO) cluster is running you can register the ARO cluster with the Red Hat Hybrid Cloud Console.

IMPORTANT
Currently, creating an Azure Red Hat OpenShift cluster is not supported using the Red Hat Hybrid Cloud Console.

Prerequisites

- Installation of the Azure client, version 2.6 or higher.
- A Microsoft Azure organizational or user account with an active subscription.
- Download a pull secret for authentication to an OpenShift cluster.

Procedure

1. Login to the Microsoft Azure portal from the command-line client:

   ```bash
   $ az login
   ```

   **NOTE**
   Running this command opens a web browser for you to finish the login process. You can close the browser after you successfully logged in.

2. Set the account subscription:

   **Syntax**
   ```bash
   az account set --subscription 'SUBSCRIPTION_NAME'
   ```

   **Example**
   ```bash
   $ az account set --subscription 'Example Sub'
   ```
3. Register the Azure resources:

Example

$ az provider register -n Microsoft.RedHatOpenShift --wait
$ az provider register -n Microsoft.Compute --wait
$ az provider register -n Microsoft.Storage --wait

4. Create a resource group:

Syntax

az group create --name RESOURCE_GROUP --location LOCATION

Example

$ az group create --name rhoda-aro-gr --location eastus

5. Create a virtual network:

Syntax

az network vnet create --resource-group RESOURCE_GROUP \ 
--name aro-vnet \ 
--address-prefixes IP_SUBNET/CIDR

Example

$ az network vnet create --resource-group rhoda-aro-gr \ 
--name aro-vnet \ 
--address-prefixes 10.0.0.0/22

6. Create a subnet for the main node:

Syntax

az network vnet subnet create --resource-group RESOURCE_GROUP \ 
--vnet-name aro-vnet \ 
--name main-subnet \ 
--address-prefixes IP_SUBNET/CIDR \ 
--service-endpoints Microsoft.ContainerRegistry

Example

$ az network vnet subnet create --resource-group rhoda-aro-gr \ 
--vnet-name aro-vnet \ 
--name main-subnet \ 
--address-prefixes 10.0.0.0/23 \ 
--service-endpoints Microsoft.ContainerRegistry

7. Create a subnet for the worker node:

Syntax
Syntax

```
az network vnet subnet create --resource-group RESOURCE_GROUP
--vnet-name aro-vnet
--name worker-subnet
--address-prefixes IP_SUBNET/CIDR
--service-endpoints Microsoft.ContainerRegistry
```

Example

```
$ az network vnet subnet create --resource-group rhoda-aro-gr
--vnet-name aro-vnet
--name worker-subnet
--address-prefixes 10.0.2.0/23
--service-endpoints Microsoft.ContainerRegistry
```

8. Disable private endpoint policies for the main subnet:

Syntax

```
az network vnet subnet update --name main-subnet
--resource-group RESOURCE_GROUP
--vnet-name aro-vnet
--disable-private-link-service-network-policies true
```

Example

```
$ az network vnet subnet update --name main-subnet
--resource-group rhoda-aro-gr
--vnet-name aro-vnet
--disable-private-link-service-network-policies true
```

9. Create the ARO cluster:

Syntax

```
az aro create --resource-group RESOURCE_GROUP
--name CLUSTER_NAME
--vnet aro-vnet
--master-subnet main-subnet
--worker-subnet worker-subnet
--apiserver-visibility Public
--ingress-visibility Public
--pull-secret @DOWNLOAD_PULS_SECRET_FILE_PWD
```

Example

```
az aro create --resource-group rhoda-aro-gr
--name rhoda-aro-example
--vnet aro-vnet
--master-subnet main-subnet
--worker-subnet worker-subnet
```
NOTE

The cluster creation process can take up to an hour to complete.

Verification

1. Once the cluster creation process finishes, copy the **kubeadmin** credentials, and the OpenShift console URL.
   a. Find the **kubeadmin** credentials:
      
      Syntax
      
      ```bash
      az aro list-credentials --name CLUSTER_NAME \
      --resource-group RESOURCE_GROUP
      ```
      
      Example
      
      ```bash
      $ az aro list-credentials --name rhoda-aro-example \
      --resource-group rhoda-aro-gr
      ```
      
      ```json
      {}
      "kubeadminPassword": "AAFAA-Zk3aR-V46bu-A4F7D",
      "kubeadminUsername": "kubeadmin"
      }
      ```
   b. Get the OpenShift console URL:
      
      Syntax
      
      ```bash
      az aro show --name CLUSTER_NAME \
      --resource-group RESOURCE_GROUP \
      --query "consoleProfile.url" -o tsv
      ```
      
      Example
      
      ```bash
      $ az aro show --name rhoda-aro-example \
      --resource-group rhoda-aro-gr \
      --query "consoleProfile.url" -o tsv
      ```
      
      https://console-openshift-console.apps.b879bjix.eastus.example.com/

2. Use the **kubeadmin** credentials to login to the OpenShift console.

3. From the OpenShift console home page, select the **Administrator** perspective, expand **Home** on the navigation menu, and click **Overview**.

   Copy the **Cluster ID** value. This value is used to register the ARO cluster with the Red Hat Hybrid Cloud Console.
H.3. REGISTERING AN AZURE RED HAT OPENSHEET CLUSTER WITH THE RED HAT HYBRID CLOUD CONSOLE

You have to manually register Azure Red Hat OpenShift (ARO) clusters with the Red Hat Hybrid Cloud Console for them to start sending telemetry data about the ARO cluster.

Prerequisites

- A running ARO cluster.
- Download a pull secret for authentication to an OpenShift cluster, and the Red Hat Hybrid Cloud Console.
- A Red Hat user account for access to the Red Hat Hybrid Cloud Console.

Procedure

1. Log into OpenShift using the a command-line interface:

   Syntax

   ```
oc login --token=TOKEN --server=SERVER_URL_AND_PORT
   ```

   Example

   ```
   $ oc login \ --token=sha256~ZvFDBvoIYAbVECixS4-WmkN4RfnNd8Neh3y1WuiFPXC \ --server=https://example.com:6443
   ```

   NOTE

   You can find your command-line login token and URL from the OpenShift console. Once you log into the OpenShift console, click your user name, click Copy login command, login once again using your user name and password, then click Display Token to view the command.

2. From the OpenShift host, get and encode the pull secret file:

   Example

   ```
   $ oc get secrets pull-secret -n openshift-config \ -o template='{{index .data ".dockerconfigjson"}}' | \ base64 -d > pull-secret-post.json
   ```

3. Open the downloaded pull secret file, and copy the whole cloud.redhat.com authorization entry.

   Copy Example

   ```
   "cloud.openshift.com":{
   "auth":"<._REDACTED_>",
   "email":"user@example.com"
   }
   ```
Close the pull secret file.

4. Open the `pull-secret-post.json` file, and paste the copied `cloud.redhat.com` authorization entry. Close the pull secret file.

5. Validate the JSON file:

   **Example**
   
   ```
   $ cat pull-secret-post.json | jq
   ```

6. Update the ARO cluster with the new pull secret file:

   **Example**
   
   ```
   $ oc set data secret/pull-secret -n openshift-config \
   --from-file=.dockerconfigjson=./pull-secret-post.json
   ```

   After a few minutes the ARO cluster appears in the Red Hat Hybrid Cloud Console.

**Verification**

1. Login to the Red Hat Hybrid Cloud Console.

2. Click OpenShift, then click Cluster.

3. From the Cluster page, filter for the ARO cluster by its cluster ID. The new ARO cluster is now in the cluster list.

   **NOTE**
   
   The default service-level agreement (SLA) is set to Self-Support 60-day evaluation.

4. Optionally, you can change the default SLA:

   a. Click the cluster ID.

   b. Click the Edit subscription settings link within the warning message.

   c. Select the appropriate SLA for your subscription, and click Save.

**H.4. INSTALLING OPENSIGHT DATABASE ACCESS USING THE OPERATOR LIFECYCLE MANAGER**

For some OpenShift cluster types, such as Azure Red Hat OpenShift (ARO), you must install Red Hat OpenShift Database Access (RHODA) manually using the Operator Lifecycle Manager (OLM).

**Prerequisites**

- A running OpenShift Dedicate (OSD) or ARO cluster.

- Administrator privileges to the ARO cluster.
Procedure

1. Log into OpenShift using the a command-line interface:

**Syntax**

```
oc login --token=TOKEN --server=SERVER_URL_AND_PORT
```

**Example**

```
$ oc login --token=sha256~ZvFDBvOlYAbVECixS4-WmkN4RfnN8Neh3y1WuiFPXC --server=https://example.com:6443
```

**NOTE**

You can find your command-line login token and URL from the OpenShift console. Once you log into the OpenShift console, click your user name, click **Copy login command**, login once again using your user name and password, then click **Display Token** to view the command.

2. Create an OpenShift Database Access catalog source using the latest add-on image repository:

**Example**

```
$ cat <<EOF | oc apply -f -
apiVersion: operators.coreos.com/v1alpha1
type: CatalogSource
metadata:
  name: dbaas-operator
  namespace: openshift-marketplace
spec:
  sourceType: grpc
  image: quay.io/osd-addons/dbaas-operator-index@sha256:b699851c2a839ee85a98a8daf3b619c0b34716c081046b229c37e8ea2d2efa96
displayName: DBaaS Operator
EOF
```

3. Verify the catalog source is added, and in a ready state:

**Example**

```
$ oc get catalogsourcel dbaas-operator \
  -n openshift-marketplace \
  -o jsonpath='{.status.connectionState.lastObservedState} {"n"}'
```

**IMPORTANT**

Wait until the catalog source is in a READY state, before proceeding to the next step.

4. Login to the OpenShift console with a user that has administrative privileges.
5. In the Administrator perspective, expand the Operator navigation menu, and click OperatorHub.

6. In the filter field, type database access, and click the OpenShift Database Access Operator tile.

7. Click the Install button to show the operator details.

8. The default and recommended namespace for the OpenShift Database Access operator is openshift-dbaas-operator, click Install on the Install Operator page.

   **NOTE**

   All dependencies are automatically installed, this includes the provider account operators, and the quick-start guides.

**Verification**

1. Once the OpenShift Database Access operator successfully installs, a new navigation menu item is added, called Data Services. Expand the Data Services menu. This might take a few minutes to refresh the navigation menu.

2. Click Database Access.

3. On the Database Access home page you see an empty inventory table.