Cloud Service Credentials in Unity Catalog: Private Preview Guide

Jun 5, 2024

⚠️ **Databricks Confidential:** All information in this document is strictly confidential and not intended for being shared outside of the Unity Catalog cloud credentials private preview. The private preview may change or terminate at any time, and the product may change or may never be released.

Feature Overview

The Cloud Service Credentials in Unity Catalog feature provides the ability to:

- **Create and manage cloud credentials using Unity Catalog.** Bring in cloud credentials (such as an IAM role on AWS) as a new securable object governed by Unity Catalog (UC). This includes to create one or more cloud credential objects under a UC metastore, and manage permissions who can use those credentials to access cloud services.
- **Use cloud credentials to authenticate with cloud services.** If granted access to a UC cloud credential, users can use those to authenticate with cloud services from their REPL. This can be done by retrieving credentials directly, or by using the built-in integration for native cloud SDKs (such as boto3 for Python on AWS).
- **Extra security using short-lived credentials.** When a user requests to access a cloud credential, Unity Catalog will talk to the underlying cloud resource to generate a short-lived credential and only pass that credential to the user (code). When using the official cloud SDKs, Unity Catalog also takes care of automatic credential rotation and will generate and use a new credential when the existing one is about to expire.

Feature Limitations

👉 **Feedback appreciated:** Please let us know what you need and why.

- **AWS only:** In this private preview, we only support Amazon Web Services (AWS) only. This is a temporary limitation, and we will add support for Azure and GCP as well.
- **Python only:** The private preview supports Python only. This is a temporary limitation and will be lifted in the future.a

- **No SQL warehouses:** We only support Unity Catalog-enabled clusters (in single user or shared access mode) and DLT pipelines. SQL warehouses are not supported. We plan to add support for cloud service credentials for Python UDFs on SQL warehouses in the future.
- **Default credential via environment variable only.** In this private preview, the only way to specify a default credential for workloads is by setting a specific environment variable. This is an intermediate design and might change in the future.
- **API only for creating & managing UC cloud credentials.** UC cloud credentials can only be created and managed (e.g., granted permissions) by using the API or notebook mentioned in this guide. This is a temporary limitation, and we plan to add the ability to do the same using the UI and SQL API.
- **Preliminary API.** The APIs provided as part of this private preview are preliminary in nature, and might change in the future. We strongly recommend not to create any hard dependencies based on the syntax or semantics of the APIs as provided in this private preview.

Known Bugs

- Audit events for actions performed on service credentials might not show up on the `system.access.audit` table. Audit information about who created/deleted/updated/read/listed/used a service credential will be available.

Requirements

In order to participate in the private preview, you must meet the following requirements:

- **Unity Catalog (UC):** Service credentials are created and governed in UC, and hence, Unity catalog must be enabled for the workspace used for the private preview. For an introduction to Unity Catalog, see this site.
- **UC-enabled clusters only:** Cloud credentials are accessed through Unity Catalog, which is only available on UC-enabled compute, such as single user or shared access mode clusters in a UC-enabled workspace. No user isolation clusters, credential passthrough clusters or any other legacy cluster created via APIs are not supported.
- **Terms and conditions**: In order to participate, you need to acknowledge the terms and conditions of the private preview. If you have not been provided or have not accepted the terms, you cannot participate in this private preview. Please reach out to your account team.

Feedback & Support

For any feedback or questions concerning the private preview, please contact your Databricks representative or reach out to uc-cloud-credentials-preview@databricks.com.

User Guide: Cloud Credentials in Unity Catalog

High level steps to get started with UC cloud credentials private preview:

(A) Prepare your workspace and clusters to use UC cloud credentials
(B) Create a cloud credential in Unity Catalog
(C) Grant users permission to use a cloud credential
(D) List/Update/Delete credentials
(E) Use a UC cloud credential to access a cloud service (using Python and boto3)
(F) Use a UC service credential to access a cloud service (using Scala and AWS SDK 1.x)
(G) Set a default UC cloud credential for a cluster

Please follow the detailed walkthrough in the below sections:

(A) Prepare your workspace and clusters

⚠️ **Before you start:** Please verify you fulfill all the requirements mentioned above and that your workspace has already been activated for the private preview. If not sure, please contact your Databricks representative.

⚠️ **Access to Google Drive** is restricted to the email addresses used to sign up for the private preview. If you need to access the files from another account, please request access via Google or reach out to uc-cloud-credentials-private-preview@databricks.com.

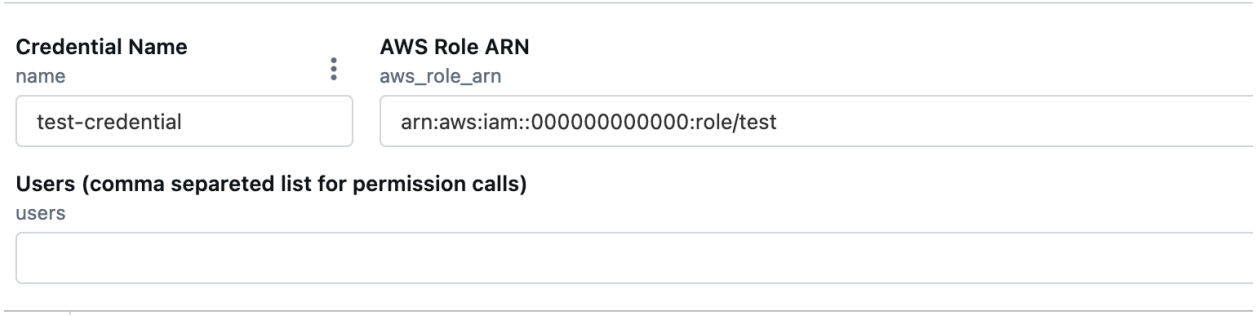
1. Log into the workspace that the UC service credentials preview was activated for.
2. Import the credential management notebook into the workspace
3. Import the service credentials Python library into the workspace filesystem or UC volume.

(B) Create a service credential in Unity Catalog

In the management notebook:

1. Fill in a name of your service credential (it will be used later to reference the credential)

2. Fill in the role ARN



3. As a metastore admin, run cell 2 ("Create Service Credential")

OR

1. Use the API endpoint by running the cURL command:
```
curl -X POST -H "Authorization: Bearer $DATABRICKS_TOKEN" -d '{"name":
"test-credential", "purpose": "SERVICE", "aws_iam_role": {"role_arn":
"arn:aws:iam::000000000000:role/test"}}'
"$DATABRICKS_HOST/api/2.1/unity-catalog/credentials"
```

(C) Grant users permission to use a cloud credential

By default the metastore admin and the credential creator will have access to use it. To enable other users, you can grant the "ACCESS" privilege using the management notebook:

1. Fill in the credential name and a comma separated list of user names (note you can specify "account users" for a group of all workspace accounts)
2. As a metastore admin or service credential owner run, cell 10 ("Give access permission to service credentials")

(D) List/Update/Delete credentials

- ### Listing the credentials

  Use the API endpoint by running the cURL command:
  ```
  curl -X GET -H "Authorization: Bearer $DATABRICKS_TOKEN"
  "$DATABRICKS_HOST/api/2.1/unity-catalog/credentials”
  ```

  Returns a list of all credentials (storage and service) visible to the user. The ability to filter by credential type will be added to this endpoint in the future.

- ### Updating a credential

  Use the API endpoint by running the cURL command:

  ```
  curl -X PATCH -H "Authorization: Bearer $DATABRICKS_TOKEN" -d '{"new_name": "test-credential", "aws_iam_role": {"role_arn": "arn:aws:iam::000000000000:role/test"}}' "$DATABRICKS_HOST/api/2.1/unity-catalog/credentials/{name}”
  ```

  This will update the credential with name by setting the name to “new_name” and the associated IAM role to “aws_iam_role”. Both fields are optional - only the fields passed will be updated (as is the case for this endpoint

- ### Deleting a credential

  Use the API endpoint by running the cURL command:

  ```
  curl -X DELETE -H "Authorization: Bearer $DATABRICKS_TOKEN" "$DATABRICKS_HOST/api/2.1/unity-catalog/credentials/{name}”
  ```

  This will delete the credential “name” in the meta-store.

(E) Use a UC cloud credential to access a cloud service (using Python and boto3)

Using the service credentials in a notebook requires a dedicated Python library set up in the runtime environment (see here). The library can be installed as a notebook-scoped library or a cluster library.

To install it in a notebook scope the following command can be used:

```
%pip install /<Workspace or volume path>/service_credentials-1.0.1-py3-none-any.whl
```

To use service credentials in a notebook the library has to be explicitly included in the first cell run:

```python
import service_credentials
```

To configure boto3 session to use a specific service credential the following code should be used

```python
import service_credentials
import boto3
boto3_session =
boto3.Session(botocore_session=dbutils.credentials.getServiceCredentialsProvider('test-credential')) # for a service credential named 'test-credential'
```

```
s3 = boto3_session.client('s3')
```

Alternatively, if the `DATABRICKS_DEFAULT_SERVICE_CREDENTIAL_NAME` environment variable is set, this credential name will be used as default when creating a boto3 session:

```python
import service_credentials
import boto3
s3 = boto3.client('s3')
```

(F) Use a UC service credential to access a cloud service (using Scala and AWS SDK 1.x)

⚠️ **Scala on UC Shared Clusters** currently comes with a limited set of libraries compared to other cluster types, which does not include AWS SDK. We plan to lift this restriction for GA, but until then, you need to install any missing libraries by yourself.

1. Download the service credentials client library jar and upload it to a UC volume location.
2. In case of a UC cluster in shared access mode, add the UC volume containing the library (and the AWS SDK, if not already added) to the list of allowed libraries. If not added, the libraries cannot be installed on this cluster type.
3. Install the service credentials jar (and AWS SDK, if not already done) on the cluster (see here)
4. Use the library in a Scala notebook by passing a `DatabricksServiceCredentialsProvider` object as the credentials parameter to AWS client builders. The `DatabricksServiceCredentialsProvider` constructor takes a single parameter being the service credential name.

   Example (UC service credential specific code highlighted):

```java
import com.databricks.servicecredentials.DatabricksServiceCredentialsProvider
import com.amazonaws.services.secretsmanager.AWSSecretsManagerClientBuilder

val region = "us-east-1"
val client = AWSSecretsManagerClientBuilder.standard()
```

```
 .withCredentials(new
DatabricksServiceCredentialsProvider("service-credential-name"))
 .withRegion(region)
 .build()
```

5. (Optional) If the credential name is specified in the designated environment variable `DATABRICKS_DEFAULT_SERVICE_CREDENTIAL_NAME`, the credential name can be omitted. See here how to add this environment variable to your cluster configuration

(G) Set a default UC cloud credential for a cluster

1. Open cluster edit page
2. Open "Advanced options" section at the bottom
3. Select "Spark" tab
4. Add the following entry in the "Environment variables" input: `"DATABRICKS_DEFAULT_SERVICE_CREDENTIAL_NAME=test-credential"`