Red Hat OpenShift AI Self-Managed 2.23

Working with data in an S3-compatible object store

Work with data stored in an S3-compatible object store from your workbench

Last Updated: 2025-08-19

Red Hat OpenShift AI Self-Managed 2.23 Working with data in an S3-compatible object store

Work with data stored in an S3-compatible object store from your workbench

Abstract

Learn how to work with data stored in an S3-compatible object store from your workbench.

Table of Contents

PREFACE

If you have data stored in an S3-compatible object store such as Ceph, MinIO, or IBM Cloud Object Storage, you can access the data from your workbench.

CHAPTER 1. PREREQUISITES

- You have created a workbench in OpenShift AI. For more information, see Creating a workbench and selecting an IDE.
- You have access to an S3-compatible object store.
- You have the credentials for your S3-compatible object storage account.
- You have files to work with in your object store.
- You have configured a connection for your workbench based on the credentials of your S3-compatible storage account. For more information, see Using connections.

CHAPTER 2. CREATING AN S3 CLIENT

To interact with data stored in an S3-compatible object store from a workbench, you must create a local client to handle requests to the AWS S3 service by using an AWS SDK such as Boto3.

Boto3 is an AWS SDK for Python that provides an API for creating and managing AWS services, such as AWS S3 or S3-compatible object storage.

After you have configured a Boto3 client for the S3 service from a workbench, you can connect and work with data in your S3-compatible object store.

Prerequisites

- You have access to an S3-compatible object store.
- You have stored files in a bucket on your object store.
- You have logged in to Red Hat OpenShift AI.
- You have created a data science project.
- You have added a workbench to the project using a workbench image.
- You have configured a connection for your workbench based on the credentials of your S3-compatible storage account.

Procedure

1. From the OpenShift AI dashboard, click **Data science projects**.
2. Click the name of the project that contains the workbench.
3. Click the **Workbenches** tab.
4. If the status of the workbench is **Running**, skip to the next step.
   If the status of the workbench is **Stopped**, in the **Status** column for the workbench, click **Start**.

   The **Status** column changes from **Stopped** to **Starting** when the workbench server is starting, and then to **Running** when the workbench has successfully started.
5. Click the open icon (↗) next to the workbench.
   Your Jupyter environment window opens.
6. On the toolbar, click the **Git Clone** icon and then select **Clone a Repository**.
7. In the **Clone a repo** dialog, enter the following URL **https://github.com/opendatahub-io/odh-doc-examples.git** and then click **Clone**.
8. In the file browser, select the newly-created **odh-doc-examples** folder.
9. Double-click the newly created **storage** folder.
   You see a Jupyter notebook named **s3client_examples.ipynb**.
10. Double-click the **s3client_examples.ipynb** file to launch the Jupyter notebook.
    The Jupyter notebook opens. You see code examples for the following tasks:

- Installing Boto3 and required Boto3 libraries
- Creating an S3 client session
- Creating an S3 client connection
- Listing files
- Creating a bucket
- Uploading a file to a bucket
- Downloading a file from a bucket
- Copying files between buckets
- Deleting an object from a bucket
- Deleting a bucket

11. In the Jupyter notebook, locate the following instructions to install Boto3 and its required libraries, and run the code cell:

```
#Upgrade pip to the latest version
!pip3 install --upgrade pip

#Install Boto3
!pip3 install boto3

#Install Boto3 libraries
import os
import boto3
from botocore.client import Config
from boto3 import session

#Check Boto3 version
!pip3 show boto3
```

    The instructions in the code cell update the Python Package Manager (pip) to the latest version, install Boto3 and its required libraries, and display the version of Boto3 installed.

12. Locate the following instructions to create an S3 client and session. Run the code cell.

```
#Creating an S3 client
#Define credentials
key_id = os.environ.get('AWS_ACCESS_KEY_ID')
secret_key = os.environ.get('AWS_SECRET_ACCESS_KEY')
endpoint = os.environ.get('AWS_S3_ENDPOINT')
region = os.environ.get('AWS_DEFAULT_REGION')

#Define client session
session = Boto3.session.Session(aws_access_key_id=key_id,
aws_secret_access_key=secret_key)

#Define client connection
s3_client = Boto3.client('s3', aws_access_key_id=key_id,
aws_secret_access_key=secret_key,aws_session_token=None,
    config=Boto3.session.Config(signature_version='s3v4'),
                    endpoint_url=endpoint,
                    region_name=region)
```

The instructions in the code cell configure an S3 client and establish a session to your S3-compatible object store.

Verification

- To use the S3 client to connect to your object store and list the available buckets, locate the following instructions to list buckets and run the code cell:

  ```
  s3_client.list_buckets()
  ```

  A successful response includes a **HTTPStatusCode** of **200** and a list of buckets similar to the following output:

  ```
  'HTTPStatusCode': 200,
  'Buckets': [{'Name': 'aqs086-image-registry',
  'CreationDate': datetime.datetime(2024, 1, 16, 20, 21, 36, 244000, tzinfo=tzlocal  ())}]
  ```

CHAPTER 3. LISTING AVAILABLE BUCKETS IN YOUR OBJECT STORE

To list buckets that are available in your object store, use the **list_bucket()** method.

Prerequisites

- You have cloned the **odh-doc-examples** repository to your workbench.
- You have opened the **s3client_examples.ipynb** file in your workbench.
- You have installed Boto3 and configured the S3 client.

Procedure

1. In the Jupyter notebook, locate the following instructions that lists available buckets and then run the code cell.

   ```
   #List available buckets
   s3_client.list_buckets()
   ```

   A successful response includes an HTTP request status code of **200** and a list of buckets, similar to the following output:

   ```
   'HTTPStatusCode': 200,
   'Buckets': [{'Name': 'aqs086-image-registry',
   'CreationDate': datetime.datetime(2024, 1, 16, 20, 21, 36, 244000, tzinfo=tzlocal( ))},
   ```

2. Locate the instructions that prints only the names of available buckets and execute the code cell.

   ```
   #Print only names of available buckets
   for bucket in s3_client.list_buckets()[‘Buckets’]:
       print(bucket[‘Name’])
   ```

   The output displays the names of the buckets, similar to the following example.

   ```
   aqs086-image-registry
   aqs087-image-registry
   aqs135-image-registry
   aqs246-image-registry
   ```

CHAPTER 4. CREATING A BUCKET IN YOUR OBJECT STORE

To create a bucket in your object store from your workbench, use the **create_bucket()** method.

Prerequisites

- You have cloned the **odh-doc-examples** repository to your workbench.
- You have opened the **s3client_examples.ipynb** file in your workbench.
- You have installed Boto3 and configured an S3 client.

Procedure

1. In the notebook file, locate the following instructions to create a bucket:

```
#Create bucket                                        `
s3_client.create_bucket(Bucket='<bucket_name>')
```

2. Replace **<name_of_the_bucket>** with the name of the bucket that you want to create, as shown in the example, and then run the code cell.

```
#Create bucket
s3_client.create_bucket(Bucket='aqs43-image-registry')
```

The output displays an HTTP response status code of **200**, indicating a successful request.

Verification

- Locate the instructions to list buckets and execute the code cell.

```
for bucket in s3_client.list_bucket()['Buckets']:
    print(bucket['Name'])
```

The bucket that you created appears in the output.

CHAPTER 5. LISTING FILES IN YOUR BUCKET

To list files in a specific bucket, use the **list_bucket_v2()** method.

Prerequisites

- You have cloned the **odh-doc-examples** repository to your workbench.
- You have opened the **s3client_examples.ipynb** file in your workbench.
- You have installed Boto3 and configured an S3 client.

Procedure

1. In the Jupyter notebook, locate the following code for listing files.

```
#List files
#Replace <bucket_name> with the name of the bucket.
bucket_name = ‘<bucket_name>’
s3_client.list_objects_v2(Bucket=bucket_name)
```

2. Replace **<bucket_name>** with the name of your own bucket, as shown in the example, and then run the code cell.

```
#List files
#Replace <bucket_name> with the name of the bucket.
bucket_name = ‘aqs27-registry’
s3_client.list_objects_v2(Bucket=bucket_name)
```

The output displays information about the files that are available in the specified bucket.

3. Locate the code cell that lists only the names of the files.

```
#Print only names of files
bucket_name = ‘<bucket_name>’
for key in s3_client.list_objects_v2(Bucket=bucket_name)[‘Contents’]:
 print(key[‘Key’])
```

4. Replace ***<bucket_name>*** with the name of your bucket, as shown in the example, and run the code cell:

```
#Print only names of files
bucket_name = ‘aqs27-registry’
for key in s3_client.list_objects_v2(Bucket=bucket_name)[‘Contents’]:
print(key[‘Key’])
```

The output displays a list of file names that are available in the specified bucket.

5. Refine the previous query to specify a file path, by locating the following code cell:

```
bucket_name = ‘<bucket_name>’
for key in s3_client.list_objects_v2(Bucket=bucket_name,Prefix=’<start_of_file_path’)
[‘Contents’]:
print(key[‘Key’])
```

6. Replace ***<bucket_name>*** and ***<start_of_file_path>*** with your own values and run the code cell.

CHAPTER 6. DOWNLOADING FILES FROM YOUR BUCKET

To download a file from your bucket to your workbench, use the **download_file()** method.

Prerequisites

- You have cloned the **odh-doc-examples** repository to your workbench.
- You have opened the **s3client_examples.ipynb** file in your workbench.
- You have installed Boto3 and configured an S3 client.

Procedure

1. In the Jupyter notebook, locate the following instructions to download files from a bucket:

   ```
   #Download file from bucket
   #Replace the following values with your own:
   #<bucket_name>: The name of the bucket.
   #<object_name>: The name of the file to download. Must include full path to the file on the
   bucket.
   #<file_name>: The name of the file when downloaded.

   s3_client.download_file('<bucket_name>','<object_name>','<file_name>')
   ```

2. Modify the code sample:
   a. Replace **<bucket_name>** with the name of the bucket that the file is located in... Replace **<object_name>** with the name of the file that you want to download.
   b. Replace **<file_name>** with the name and path that you want the file to be downloaded to, as shown in the example.

   ```
   s3_client.download_file('aqs086-image-registry',
                           'series35-image36-086.csv',
                           '\tmp\series35-image36-086.csv_old')
   ```

3. Run the code cell.

Verification

- The file that you downloaded appears in the path that you specified on your workbench.

CHAPTER 7. UPLOADING FILES TO YOUR BUCKET

To upload files to your bucket from your workbench, use the **upload_file()** method.

Prerequisites

- You have cloned the **odh-doc-examples** repository to your workbench.
- You have opened the **s3client_examples.ipynb** file in your workbench.
- You have installed Boto3 and configured an S3 client.
- You have imported the files that you want to upload to your object store to your workbench.

Procedure

1. In the Jupyter notebook, locate the instructions to upload files to a bucket.

```
#Upload file to bucket
#Replace <file_name>, <bucket_name>, and <object_name> with your values.
#<file_name>: Name of the file to upload. This value must include the full local path to the file
on your workbench.
#<bucket_name>: The name of the bucket to upload the file to.
#<object_name>: The full key to use to save the file to the bucket.

s3_client.upload_file('<file_name>', '<bucket_name>', '<object_name>')
```

2. Replace ***<file_name>***, ***<bucket_name>*** and ***<object_name>*** with your own values, as shown in the example, and then run the code cell.

```
s3_client.upload_file('image-973-series123.csv', 'aqs973-image-registry', '/tmp/image-973-
series124.csv')
```

Verification

- Locate the following instructions to list files in a bucket:

```
#Upload Verification
for key in s3_client.list_objects_v2(Bucket='<bucket_name>')['Contents']:
print(key['Key'])
```

- Replace **<bucket_name>** with the name of your bucket, as shown in the example, and then run the code cell.

```
#Upload Verification
for key in s3_client.list_objects_v2(Bucket='aqs973-image-registry')['Contents']:
print(key['Key'])
```

The file that you uploaded is displayed in the output.

CHAPTER 8. COPYING FILES BETWEEN BUCKETS

To copy files between buckets in your object store from your workbench, use the **copy()** method.

Prerequisites

- You have cloned the **odh-doc-examples** repository to your workbench.
- You have opened the **s3client_examples.ipynb** file in your workbench.
- You have installed Boto3 and configured an S3 client.
- You know the key of the source file that you want to copy, and the bucket that the file is stored in.

Procedure

1. In the notebook, locate the following instructions to copy files between buckets:

```
#Copying files between buckets
#Replace the placeholder values with your own.
copy_source = {
    'Bucket': '<bucket_name>',
    'Key': '<key>'
}
s3_client.copy(copy_source, '<destination bucket>', '<destination_key>')
```

2. Within the **copy_source** block, replace ***<bucket_name>*** with the name of the source bucket and ***<key>*** with the key of the source file, as shown in the example.

```
copy_source = {
    'Bucket': 'aqs086-image-registry',
    'Key': 'series43-image12-086.csv'
}
```

3. Replace the **<destination_bucket>** with the name of the bucket to copy to, and **<destination_key>** with the name of the key to copy to, as shown in the example. Execute the code cell.

```
s3_client.copy(copy_source, 'aqs971-image-registry', '/tmp/series43-image12-086.csv')
```

Verification

- Locate the following instructions to list objects in a bucket.

```
#Copy Verification
bucket_name = '<bucket_name>'
for key in s3_client.list_objects_v2(Bucket=bucket_name)['Contents']:
        print(key['Key'])
```

- Replace **<bucket_name>** with the name of the destination bucket, as shown in the example, and run the code cell.

```
#Copy Verification
bucket_name = 'aqs971-image-registry'
for key in s3_client.list_objects_v2(Bucket=bucket_name)['Contents']:
print(key['Key']).
```

The file that you copied is displayed in the output.

CHAPTER 9. DELETING FILES FROM YOUR BUCKET

To delete files from your bucket from your workbench, use the **delete_file()** method.

Prerequisites

- You have cloned the **odh-doc-examples** repository to your workbench.
- You have opened the **s3client_examples.ipynb** file in your workbench.
- You have installed Boto3 and configured an S3 client.
- You know the key of the file you want to delete and the bucket that the file is located in.

Procedure

1. In the Jupyter notebook, locate the following instructions to delete files from a bucket:

```
#Delete files from bucket
s3_client.delete_object(Bucket='<bucket_name>', Key='<object_key>')
```

2. Replace **<bucket_name>** with the name of your bucket and **<key>** with the key of the file you want to delete, as shown in the example. Run the code cell.

```
#Delete object from bucket
s3_client.delete_object(Bucket='aqs971-image-registry', Key='/tmp/series43-image12-
086.csv')
```

The output displays a HTTP response status code of **204**, which indicates that the request was successful.

Verification

- Locate the following instructions to list files in a bucket:

```
#Delete Object Verification
bucket_name = '<bucket_name>'
for key in s3_client.list_objects_v2(Bucket=bucket_name)['Contents']:
print(key['Key'])
```

- Replace ***<bucket_name>*** with the name of your bucket, as shown in the example and run the code cell.

```
#Delete Object Verification
bucket_name = 'aqs971-image-registry'
for key in s3_client.list_objects_v2(Bucket=bucket_name)['Contents']:
print(key['Key'])
```

The deleted file does not appear in the output.

CHAPTER 10. DELETING A BUCKET FROM YOUR OBJECT STORE

To delete a bucket from your object store from your workbench, use the **delete_bucket()** method.

Prerequisites

- You have cloned the **odh-doc-examples** repository to your workbench.
- You have opened the **s3client_examples.ipynb** file in your workbench.
- You have installed Boto3 and configured an S3 client.
- You have ensured that the bucket that you want to delete is empty.

Procedure

1. In the Jupyter notebook, locate the following instructions to delete a bucket:

```
#Delete bucket
s3_client.delete_bucket(Bucket='<bucket_name>')
```

2. Replace **<bucket_name>** with the name of the bucket that you want to delete and run the code cell.

```
#Delete bucket
s3_client.delete_bucket(Bucket='aqs971-image-registry')
```

The output displays an HTTP response status code of **200**, which indicates that the request was successful.

Verification

- Locate the instructions to list buckets, and execute the code cell.

```
for bucket in s3_client.list_bucket()[‘Buckets’]:
    print(bucket[‘Name’])
```

The bucket that you deleted does not appear in the output.

CHAPTER 11. OVERVIEW OF OBJECT STORAGE ENDPOINTS

To ensure correct configuration of object storage in OpenShift AI, you must format endpoints correctly for the different types of object storage supported. These instructions are for formatting endpoints for Amazon S3, MinIO, or other S3-compatible storage solutions, minimizing configuration errors and ensuring compatibility.

IMPORTANT

Properly formatted endpoints enable connectivity and reduce the risk of misconfigurations. Use the appropriate endpoint format for your object storage type. Improper formatting might cause connection errors or restrict access to storage resources.

11.1. MINIO (ON-CLUSTER)

For on-cluster MinIO instances, use a local endpoint URL format. Ensure the following when configuring MinIO endpoints:

- Prefix the endpoint with **http://** or **https://** depending on your MinIO security setup.
- Include the cluster IP or hostname, followed by the port number if specified.
- Use a port number if your MinIO instance requires one (default is typically **9000**).

Example:

```
http://minio-cluster.local:9000
```

NOTE

Verify that the MinIO instance is accessible within the cluster by checking your cluster DNS settings and network configurations.

11.2. AMAZON S3

When configuring endpoints for Amazon S3, use region-specific URLs. Amazon S3 endpoints generally follow this format:

- Prefix the endpoint with **https://**.
- Format as **<bucket-name>.s3.<region>.amazonaws.com**, where **<bucket-name>** is the name of your S3 bucket, and **<region>** is the AWS region code (for example, **us-west-1**, **eu-central-1**).

Example:

```
https://my-bucket.s3.us-west-2.amazonaws.com
```

NOTE

For improved security and compliance, ensure that your Amazon S3 bucket is in the correct region.

11.3. OTHER S3-COMPATIBLE OBJECT STORES

For S3-compatible storage solutions other than Amazon S3, follow the specific endpoint format required by your provider. Generally, these endpoints include the following items:

- The provider base URL, prefixed with **https:**//.
- The bucket name and region parameters as specified by the provider.
- Review the documentation from your S3-compatible provider to confirm required endpoint formats.
- Replace placeholder values like **<bucket-name>** and **<region>** with your specific configuration details.

WARNING

Incorrectly formatted endpoints for S3-compatible providers might lead to access denial. Always verify the format in your storage provider documentation to ensure compatibility.

11.4. VERIFICATION AND TROUBLESHOOTING

After configuring endpoints, verify connectivity by performing a test upload or accessing the object storage directly through the OpenShift AI dashboard. For troubleshooting, check the following items:

- **Network Accessibility**: Confirm that the endpoint is reachable from your OpenShift AI cluster.
- **Authentication**: Ensure correct access credentials for each storage type.
- **Endpoint Accuracy**: Double-check the endpoint URL format for any typos or missing components.

Additional resources

- Amazon S3 Region and Endpoint Documentation: AWS S3 Documentation

CHAPTER 12. ACCESSING S3-COMPATIBLE OBJECT STORAGE WITH SELF-SIGNED CERTIFICATES

To securely connect OpenShift AI components to object storage solutions or databases that are deployed within an OpenShift cluster that uses self-signed certificates, you must provide a certificate authority (CA) certificate. Each namespace includes a ConfigMap named **kube-root-ca.crt**, which contains the CA certificate of the internal API Server.

Prerequisites

- You have cluster administrator privileges for your OpenShift cluster.
- You have installed the OpenShift command-line interface (CLI). See Installing the OpenShift CLI.
- You have deployed an object storage solution or database in your OpenShift cluster.

Procedure

1. In a terminal window, log in to the OpenShift CLI as shown in the following example:

```
oc login api.<cluster_name>.<cluster_domain>:6443 --web
```

2. Retrieve the current OpenShift AI trusted CA configuration and store it in a new file:

```
oc get dscinitializations.dscinitialization.opendatahub.io default-dsci -o json | jq -r
'.spec.trustedCABundle.customCABundle' > /tmp/my-custom-ca-bundles.crt
```

3. Add the cluster's **kube-root-ca.crt** ConfigMap to the OpenShift AI trusted CA configuration:

```
oc get configmap kube-root-ca.crt -o jsonpath="{['data']['ca\.crt']}" >> /tmp/my-custom-ca-
bundles.crt
```

4. Update the OpenShift AI trusted CA configuration to trust certificates issued by the certificate authorities in **kube-root-ca.crt**:

```
oc patch dscinitialization default-dsci --type='json' -
p='[{"op":"replace","path":"/spec/trustedCABundle/customCABundle","value":"'"$(awk '{printf
"%s\\n", $0}' /tmp/my-custom-ca-bundles.crt)"'"}]'
```

Verification

- You can successfully deploy components that are configured to use object storage solutions or databases that are deployed in the OpenShift cluster. For example, a pipeline server that is configured to use a database deployed in the cluster starts successfully.

NOTE

You can verify your new certificate configuration by following the steps in the *OpenShift AI tutorial - Fraud Detection example*. Run the script to install local object storage buckets and create connections, and then enable data science pipelines.

For more information about running the script to install local object storage buckets, see Running a script to install local object storage buckets and create connections .

For more information about enabling data science pipelines, see Enabling data science pipelines.

CHAPTER 13. ADDITIONAL RESOURCES

- Red Hat OpenShift AI documentation
- Boto3 documentation
- Amazon Simple Storage Service documentation