RSTOR

Transporter

User Guide

December 2020

Prepared for: RSTOR Customers

Prepared by: RSTOR Support

Table of Contents

12930 Saratoga Ave., Saratoga, CA 95070

12930 Saratoga Ave., Saratoga, CA 95070

1 Introduction: Terminology

Bucket

A cloud storage resource; each bucket can be identified by tuple
id = (***bucketName, providerType, cloudserviceprovider_endpoint, region, accesskey, secretkey***)

Migration

A data transfer from one bucket to another

Migration States

A migration can have the following states:

1. Started
2. Running
3. Completed
4. Terminated
5. Waiting
6. Paused
7. Failed
8. CompletedWithErrors

12930 Saratoga Ave., Saratoga, CA 95070

2 Transporter Migration Guide

2.1 Start a Migration

1. Log in to the RSTOR Portal and navigate to the buckets screen.
2. On the top right, click on the round network icon, labeled RSTOR Transporter, to the left of the “bucket” icon.
3. To create a new migration, click on the “plus” icon in the upper right corner.
4. Select the migration source type.
5. Enter the migration source information. See the section Retrieve Credentials from different Cloud Storage Providers to see how to identify the credentials.
6. Click “Confirm” and you will be brought back to the Transporter configuration screen.
7. Repeat the steps 4 and 5 for the migration destination.
8. Once all the necessary information has been entered, click on “START MIGRATION” icon on the right.
9. Here, you will have the option to track the migration with the email your account is under. You may also enable incremental sync which migrates only the objects that are not present in your migration destination. You will also have the option to store an additional copy in RSTOR Space and/or delete the source after the transfer has been completed.

   To enable any of these, check the box next to it. If the words and box are greyed out, that means that specific feature is unavailable for the current migration.
10. Review your chosen migration then click “CONFIRM” in the popup.
11. As the migration completes, you will be able to click on the Status message of Completed to view the logs, estimated time till completion, and if needed, to abort or resume migration.

2.2 Find All Created Migrations

1. Log in to the RSTOR Portal.
2. On the top right, click on the round network icon, labeled RSTOR Transporter, to the left of the “bucket” icon.
3. There will be a list of all the migrations. If there are no migrations a message will display that there are no migrations to show.

2.3 Examine a Migration Process

1. Log in to the RSTOR Portal and navigate to the buckets screen.
2. On the top right click on the round network icon, labeled RSTOR Transporter, to the left of the "bucket" icon.
3. There will be a list of all the migrations.
4. The "Status" and "Progress" column will list the status of each migration. The migration state can be:

   Aborted: the migration was cancelled by a user

   Completed with errors: the migration was completed, but some object transfers failed

   Failed: the migration failed with an error that could not be handled; usually it is due to a user error during the input of the migration credentials

   Finished: the migration completed successfully

   In Progress: the system started working on the migration

   Paused: the migration was paused by a user

   Queued: not yet processed by the system

5. For more information about an individual migration, click on the status. If clicked more details will be shown. Here you can examine the speed, estimated time of completion, what has been queued, and what has been completed. Also, if desired, you can also pause or abort the ongoing migration.

**Note that if your migration remains in a queued state, click the "play" icon to automatically reload the migrations.*

2.4 Abort a Migration

1. View the list of all the migrations.
2. On the far right of the migration, click on the status.
3. Press the red rectangular "ABORT" icon to abort the ongoing migration.

2.5 Retry a Migration

1. If one of your migrations fail some time during the process, you will be able to retry the process. This will restart the migration and try everything again.

   You will also have the option to restart the migration in incremental mode. In incremental mode, only the objects missing at the destination will be transferred and no object will be overwritten.

3 Transporter Scripting Guide

3.1 AWSCurl

This guide will show the ways to automate and script Transporter from a bash shell using `awscurl` utility. **AWSCurl** is a Curl like tool with AWS Signature Version 4 request signing. Python 3 is required for using `awscurl`.

3.2 AWSCurl Installation

```
$ pip install awscurl
```

3.3 Installation from Source (Bleeding Edge)

```
$ pip install git+https://github.com/okigan/awscurl
```

3.4 Installation from Homebrew for MacOS

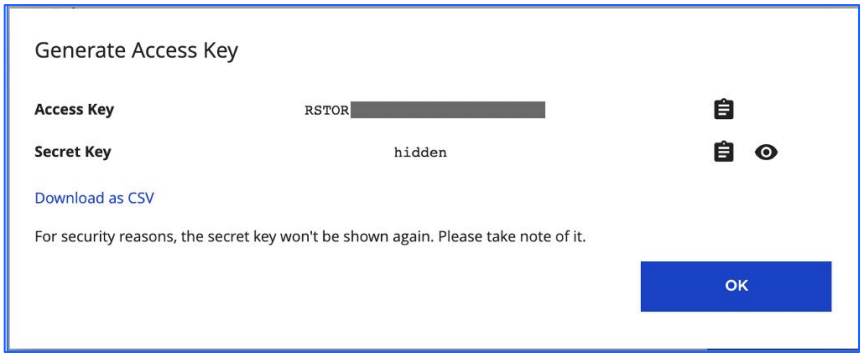
```
$ brew install awscurl
```

3.5 AWSCurl and Credentials

In order to access Transporter and perform operations you must have “Admin” privilege and use your RSTOR API access key and secret key. If you do not have one already:

1. Go to the RSTOR Portal and click on your name in the upper right corner. From there choose “MY ACCOUNT”.
2. A screen will appear with a button called “+ GENERATE KEY”. Click on “+ GENERATE KEY”.

   The system will show the following dialog:



3. Without closing the above dialog, open the file `/home/<youruser>/.aws/credentials` and add a section at the end, like the example below:

```
[rramp]
aws_access_key_id = <accesskey>
aws_secret_access_key = <secretkey>
```

Copy and paste the access key from the dialog using the "Copy to clipboard" icons on the right. Repeat the process for the secret key. This creates an aws cli profile called rramp. We will use that profile during all of these document examples. For creating a new migration, we need the right credentials (***AccessKey***, ***SecretKey***, ***Endpoint***) from different cloud storage providers. See section Retrieve Credentials From Different Cloud Storage Providers.

3.6 Create a New Migration

Below we have the **migration interoperability table** that specifies the kind of the migration (source, destination) we are currently supporting.

Source	Destination	Status
AWS / S3 compatible	RSTOR	SUPPORTED
RSTOR	AWS / S3 compatible	SUPPORTED
RSTOR	Azure	SUPPORTED
RSTOR	RSTOR	SUPPORTED
Azure	RSTOR	SUPPORTED
Azure	AWS / S3 compatible	SUPPORTED

Azure	Azure	SUPPORTED
AWS / S3 compatible	Azure	SUPPORTED
AWS / S3 compatible	AWS	SUPPORTED
Google Cloud Storage	RSTOR	SUPPORTED
Google Cloud Storage	Azure	SUPPORTED
Google Cloud Storage	AWS / S3 compatible	SUPPORTED
Any provider	GCP	WILL BE SUPPORTED SOON

For Amazon AWS we support only S3 signature version 4. The name of the migration should be unique.

3.6.1 Create a Migration from S3 With Sig V4 to S3 With Sig V4

This method is valid for the following part of the **migration interoperability table**.

Source	Destination
RSTOR	AWS / S3 compatible
RSTOR	RSTOR
AWS / S3 compatible	RSTOR
AWS / S3 compatible	AWS / S3 compatible

We make the following assumptions:

1. Use the **rramp** profile created before.
2. The migration name is **DOCAWSTOAWS02**.
3. **BUCKETSRC** is the name of the source bucket. **ACCESSKEY1** is the access key of the source bucket. **SECRETKEY1** is the secret key of the source bucket. **REGIONSRC** is the AWS region of the source bucket.
4. **BUCKETDST** is the name of the destination bucket. **ACCESSKEY2** is the access key of the destination bucket. **SECRETKEY2** is the secret key of the destination bucket. **REGIONDST** is the AWS region of the destination bucket.

5. The source endpoint is https://s3.eu-west-1.amazonaws.com.

6. The destination endpoint is http://s3.us-west-1.amazonaws.com.

```
awscurl –service rramp
https://rramp.rstorcloud.io/migrations/DOCAWSTOAWS02 \
 -X POST \
 --data ‘{“source”:{“type”:”s3v4”,
 “accessKey”:”ACCESSKEY1”,
 “secretKey”:”SECRETKEY1”,
 “bucket”:”BUCKETSRC”,
 “endpoint”:”https://s3.eu-west-1.amazonaws.com”,
 “region”:”eu-west-1”},
 “destination”:{“type”:”s3v4”,
 “accessKey”:”ACCESSKEY2”,
 “secretKey”:”SECRETKEY2”,
 “bucket”:”BUCKETDST”,
 “endpoint”:”https://s3.us-west-1.amazonaws.com”,
 “region”:”us-west-1”}}’ \
 --profile rramp
```

3.6.2 Create a Migration from GCP to S3 With Sig V4

This method is valid for the following part of the **migration interoperability table**.

Source	Destination
GCP	AWS / S3 compatible
GCP	RSTOR

In this case we make the following assumptions:

1. Use the **rramp** profile created before.

2. The migration name is **GCPTOAWS01**.

3. **BUCKETSRC** is the name of the source bucket. GACCESSKEY is the **AccessKey** and GSECRETKEY is the **SecretKey** to read from the source bucket

4. **BUCKETDST** is the name of the destination bucket. **ACCESSKEY2** and **SECRETKEY2** are the access key and secret key to write to the destination bucket. **REGIONDST** is the AWS region of the destination bucket.

5. The source endpoint is https://storage.googleapis.com.

6. The destination endpoint is http://s3.us-west-1.amazonaws.com.

```
awscurl –service rramp https://rramp.rstorcloud.io/migrations/GCPTOAWS01
\
 -X POST \
 --data ‘{“source”:{“type”:”gcp”,
 “accessKey”:”GACCESSKEY”,
 “secretKey”:”GSECRETKEY”,
 “bucket”:”BUCKETSRC”,
 “endpoint”:”https://storage.googleapis.com”,
 “region”:”us-west-1”},
 “destination”:{“type”:”s3v4”,
 “accessKey”:”ACCESSKEY2”,
 “secretKey”:”SECRETKEY2”,
 “bucket”:”BUCKETDST”,
 “endpoint”:” https://s3.us-west-1.amazonaws.com”,
 “region”:”us-west-1”}}’ \
 --profile rramp
```

3.6.3 Create a Migration from GCP to S3 with Sig V4 for a Specific Prefix

This method is valid for the following part of the **migration interoperability table**.

Source	Destination
GCP	AWS / S3 compatible
GCP	RSTOR

In this case we make the following assumptions:

1. Use the **rramp** profile created before.
2. The migration name is **GCPTOAWS01**.
3. **BUCKETSRC** is the name of the source bucket. GACCESSKEY is the GCP **AccessKey** and GSECRETKEY is the **SecretKey** to read from the source bucket.
4. **PREFIXSRC** is the name of the prefix of the objects in the source bucket (e.g. “archive-202008/”) which should be copied to destination. Only objects whose key starts with PREFIXSRC will be copied over.

5. **BUCKETDST** is the name of the destination bucket. **ACCESSKEY2** and **SECRETKEY2** are the access key and secret key to write to the destination bucket. **REGIONDST** is the AWS region of the destination bucket.

6. The source endpoint is https://storage.googleapis.com.

7. The destination endpoint is http://s3.us-west-1.amazonaws.com.

```
awscurl –service rramp https://rramp.rstorcloud.io/migrations/GCPTOAWS01
\
 -X POST \
 --data ‘{“source”:{“type”:”gcp”,
 “accessKey”:”GACCESSKEY”,
 “secretKey”:”GSECRETKEY”,
 “bucket”:”BUCKETSRC”,
 “prefix”:”PREFIXSRC/”,
 “endpoint”:”https://storage.googleapis.com”,
 “region”:”us-west-1”},
 “destination”:{“type”:”s3v4”,
 “accessKey”:”ACCESSKEY2”,
 “secretKey”:”SECRETKEY2”,
 “bucket”:”BUCKETDST”,
 “endpoint”:” https://s3.us-west-1.amazonaws.com”,
 “region”:”us-west-1”}}’ \
 --profile rramp
```

3.6.4 Create a Migration from GCP to Azure

This method is valid for the following part of the **migration interoperability table**.

Source	Destination
GCP	Azure

In this case we make the following assumptions:

1. We use the **rramp** profile created before.

2. We call the migration **GCPTOAZURE01**.

3. **BUCKETSRC** is the name of the source bucket. GACCESSKEY is the GCP **AccessKey** and GSECRETKEY is the **SecretKey** to read from the source bucket.

4. (optional) **PREFIXSRC** is the name of the prefix of the objects in the source bucket (e.g. “archive-202008/”) which should be copied to destination. Only objects whose key starts with PREFIXSRC will be copied over.

5. **BUCKETDST** is the name of the destination bucket. **rstormigrate** (name of the container) is the access key of the destination bucket. **SECRETKEY2** is the secret key of the destination bucket.

6. The source endpoint is https://storage.googleapis.com.

7. The destination endpoint is https://rstormigrate.blob.core.windows.net.

```
awscurl –service rramp
https://rramp.rstorcloud.io/migrations/GCPTOAZURE01 \
 -X POST \
 --data ‘{“source”:{“type”:”gcp”,
 “accessKey”:”GACCESSKEY”,
 “secretKey”:”GSECRETKEY”,
 “bucket”:”BUCKETSRC”,
 “prefix”:”PREFIXSRC/”,
 “endpoint”:”https://storage.googleapis.com},
 “destination”:{“type”:”azure”,
 “accessKey”:”rstormigrate”,
 “secretKey”:”SECRETKEY2”,
 “bucket”:”BUCKETDST”,
“endpoint”:”https://rstormigrate.blob.core.windows.net”}}’\
 --profile rramp
```

3.6.5 Create a Migration from Azure to S3 With Sig V4

This method is valid for the following part of the **migration interoperability table**.

Source	Destination
Azure	AWS / S3 compatible
Azure	RSTOR

In this case we make the following assumptions:

1. We use the **rramp** profile created before.

2. We call the migration **AZUREAWS01**.

3. **BUCKETSRC** is the name of the destination bucket. **rstormigrate** (name of the container) is the access key of the source bucket. **SECRETKEY1** is the secret key of the destination bucket.

4. **BUCKETDST** is the name of the destination bucket. **ACCESSKEY2** is the access key of the destination bucket. **SECRETKEY2** is the secret key of the destination bucket. **REGIONDST** is the AWS region of the destination bucket.

5. The source endpoint is https://rstormigrate.blob.core.windows.net.

6. The destination endpoint is http://s3.us-west-1.amazonaws.com.

```
awscurl --service rramp
https://rramp.rstorcloud.io/migrations/AZURETOAWS01 \
-X POST \
--data '{"source":{"type":"azure",
"accessKey":"rstormigrate",
"secretKey":"SECRETKEY1",
"bucket":"BUCKETSRC",
"endpoint":"https://rstormigrate.blob.core.windows.net",
"destination":{"type":"s3v4",
"accessKey":"ACCESSKEY2",
"secretKey":"SECRETKEY2",
"bucket":"BUCKETDST",
"region":"eu-west-1",
"endpoint":"https://s3.eu-west-1.amazonaws.com"}}' \
--profile rramp
```

3.7 Monitor the State of a Migration

Each migration has a well-defined id (i.e. **AZUREAWS01**) and if we want to check the state of a migration, we can do it with `awscurl`. As usual, we have created a **rramp** profile before.

```
awscurl --service rramp https://rramp.rstorcloud.io/state/AZUREAWS01 -X
GET --profile rramp
```

3.8 List All the Migrations

Using `awscurl` we provide an endpoint for listing all present migrations.

```
awscurl --service rramp https://rramp.rstorcloud.io/migrations -X GET --
profile rramp
```

3.9 Abort a Migration

Each migration has a well-defined id (i.e. **AZUREAWS01**) and if we abort a migration, we use that id with `awscurl`. We make the assumption that we have created a **rramp** profile before.

```
awscurl --service rramp https://rramp.rstorcloud.io/abort/AZUREAWS01 -X
POST --profile rramp
```

3.10 Retry a Migration

Each migration has a well-defined id (i.e. **AZUREAWS01**) and if we repeat an already finished migration, we use that id with `awscurl`. We make the assumption that we have created a **rramp** profile before.

```
awscurl --service rramp https://rramp.rstorcloud.io/retry/AZUREAWS01 -X
POST --profile rramp
```

4 Transporter Python Client

4.1 Requirements

```
pip install -U aws_requests_auth
```

4.2 Example

```
import requests
from aws_requests_auth.aws_auth import AWSRequestsAuth
import json

RRAMP_HOST="rramp.rstorcloud.io"
BASE_URL="https://rramp.rstorcloud.io"
MIGRATIONS_URL=BASE_URL+"/migrations"
STATE_URL= lambda x : BASE_URL+"/state/"+x
"""
aws_access_key_id and aws_secret_access_key are in ~/.aws/credentials
"""
aws_access_key_id="your key in credentials"
aws_secret_access_key="your secret in credentials"
auth = AWSRequestsAuth(aws_access_key=aws_access_key_id,
                       aws_secret_access_key=aws_secret_access_key,
                       aws_host=RRAMP_HOST,
                       aws_region='',
                       aws_service='rramp')
rrampStates = {-1: "queue",
            0: "started",
           1: "running",
           2: "completed",
           3: "terminated",
```


12930 Saratoga Ave., Saratoga, CA 95070

```
            6: "failed",
            7: "completed with errors"}

def state_to_string(s):
    if s in rrampStates.keys():
       return rrampStates[s]
    raise Exception("Invalid RRamp State")

def get_migrations():
    response = requests.get(MIGRATIONS_URL, auth=auth)
    if response.status_code != 200:
        print(response.text)
        exit()

    return json.loads(response.content)

def get_full_state(id):
    response = requests.get(STATE_URL(id), auth=auth)
    if response.status_code != 200:
        if response.status_code == 404:
            return {"state":-1}
        else:
            print(response.text)

    return json.loads(response.content)

def create_migration(migrationName):
    migration = {}
```

```
    # fill the json with the correct credentials
    migration ["source"] = {
        "type":"gcp",
        "accessKey":"GACCESSKEY",
        "secretKey":"GSECRETKEY",
        "bucket":"BUCKETSRC",
        "region": "
        "endpoint":"https://storage.googleapis.com",
    }

    migration["destination"]={
        "type":"azure",
        "accessKey":"rstormigrate",
        "secretKey":"SECRETKEY2",
        "bucket":"BUCKETDST",
        "endpoint":"https://rstormigrate.blob.core.windows.net",
    }
    # join is faster
    endpoint = ''.join([MIGRATIONS_URL,"/",migrationName])
    response = requests.post(endpoint, auth=auth, json=migration)

    if response.status_code != 201:
        print(response.text)
        exit()

if __name__ == "__main__":
    create_migration("TestMigration15")
    migrations = get_migrations()
    print("Migration states:")
```

```
    for migration in migrations:
        id = migration["name"]
        print("\tID:\t", id)
        state = get_full_state(id)["state"]
print("\tState: \t", state_to_string(state))
```

5 Transporter .NET Core Client

5.1 Create a Migration

In order to create a migration in .NET Core the S3 V4 or V2 signer logic is required in a class. Signer examples in C# at http://docs.aws.amazon.com/AmazonS3/latest/API/samples/AmazonS3SigV4_Samples_CSharp.zip.

Unzipping the *.zip will create two folders:

1. **Signers**, that contains all the logic for signing a S3V4, in particular **AWS4SignerForAuthorizationHeader** and **AWS4SigningForPost**.

2. **Util** contains utilities for signing.

1. Import those folders in the .NET Project and fix the namespaces. In Transporter, the data model for a migration is composed of:

```
namespace rramp.model {
public class CspProvider {
    public string type { get; set;}
    public string accessKey { get; set;}
    public string secretKey {get; set;}
    public string endpoint {get; set;}
    public string region { get; set;}
}
public class Migration {
    [JsonIgnore]
    public string id { get; set;}
    public CspProvider source { get; set;}
    public CspProvider destination { get;set; }
}
}
```

During the JSON serialization we should skip the id that instead will be used to fetch a migration state. We define a migration model and then we serialize it.

```
    var migration = new Migration() {
                id = "AZURETOAWS01",
                source = new CspProvider{
                    type= "azure",
                    endpoint =
"https://rstormigrate.blob.core.windows.net",
                    accessKey = "rstormigrate",
                    secretKey = "<secretkey>",
                    region = "us-west-1"},
                destination = new CspProvider{
```

```
                type= "s3v4",
                endpoint = "https://s3.us-west-1.amazonaws.com",
                accessKey = "<accesskey>",
                secretKey = "<secretkey>",
                region = "us-west-1"}};
    var body = JsonConvert.SerializeObject(migration);
```

2. This will be the body to sign with signer that we can find in the Signer folder for creating the migration.

```
  var migrationId = migration.id;
  var endpointUri = new
Uri($"https://rramp.rstorcloud.io/migrations/{migrationId}");
  var contentHash =
AWS4SignerBase.CanonicalRequestHashAlgorithm.ComputeHash(Encoding.UT
F8.GetBytes(body));
  var contentHashString = AWS4SignerBase.ToHexString(contentHash,
true);
  var signer = new AWS4SignerForPOST
           {
               EndpointUri = endpointUri,
               HttpMethod = "POST",
               Service = "rramp",
               Region = "us-west-1"
           };
```

3. Create the value to put in the ***Authorization Header***.

```
  var headers = new Dictionary<string, string>{
                  {AWS4SignerBase.X_Amz_Content_SHA256,
   contentHashString},
               {"content-length", body.Length.ToString()},
               {"content-type", "application/json"}
   };
   /*
    *  awsAccessKey is the RSTOR Transporter access key
    *  awsSecretKey is the RSTOR Transporter secret key
    *  Example define somewhere, something similar:
    *  readonly string awsAccessKey = "<accesskey>";
    *  readonly string awsSecretKey = "<secretkey>";
    */
  var authorization = signer.ComputeSignature(headers,
                                              "",   // no query
parameters
                                              "",
                                              awsAccessKey,
                                              awsSecretKey);
```

4. Now we are ready for using `HttpClient` and sending the creation request.

```
var stringContent = new HttpStringContent(body,
                            UnicodeEncoding.Utf8,
                            "application/json");
using(client = new HttpClient()) {
client.DefaultRequestHeaders.Authorization = new
AuthenticationHeaderValue(authorization);
client.DefaultRequestHeaders.Add("Content-Type",
"application/json");
client.DefaultRequestHeaders.Add(AWS4SignerBase.X_Amz_Content_SHA256
, contentHashString);
client.DefaultRequestHeaders.Add("Content-
Lenght",body.Length.ToString());
var response = await client.PostAsync(uri, stringContent);
if (reponse.StatusCode == HttpStatusCode.OK) {
   Console.Writeln("Migration created with success!");
}
}
```

5.2 List All Migrations

In this case as well, we need to prepare a signature for the Authorization header, but since it is a get we can use a different class for signing.

```
/*
*  awsAccessKey is the RSTOR Transporter access key
*  awsSecretKey is the RSTOR Transporter secret key
*  Example to define somewhere, something similar:
*  readonly string awsAccessKey = "<accesskey>";
*  readonly string awsSecretKey = "<secretkey>";
*/
var signer = new AWS4SignerForAuthorizationHeader
           {
               EndpointUri = "https://rramp.rstorcloud.io/migrations",
               HttpMethod = "GET",
               Service = "rramp",
               Region = "us-west-1"
           };
var authorization = signer.ComputeSignature(headers,
                                            "",   // no query parameters

AWS4SignerBase.EMPTY_BODY_SHA256,
                                           awsAccessKey,
                                           awsSecretKey);

using(client = new HttpClient()) {
client.DefaultRequestHeaders.Authorization = new
AuthenticationHeaderValue(authorization);
client.DefaultRequestHeaders.Add("Content-Type", "application/json");
```

```
client.DefaultRequestHeaders.Add(AWS4SignerBase.X_Amz_Content_SHA256,
AWS4SignerBase.EMPTY_BODY_SHA256);
var response = await client.GetAsync(signer.EndpointUri);
// here use the response.
}
```

5.3 Get Single Migration State

In order to retrieve the migration state, we need the id of the migration. Suppose the Id is **AWSAZURE01**. We can use the following code:

```
/*
*  awsAccessKey is the RSTOR Transporter access key
*  awsSecretKey is the RSTOR Transporter secret key
*  Example to define somewhere, something similar:
*  readonly string awsAccessKey = "<accesskey>";
*  readonly string awsSecretKey = "<secretkey>";
*/
var migrationId = "AWSAZURE01"
var endpointState = $"https://rramp.rstorcloud.io/state/{migrationId}";
var signer = new AWS4SignerForAuthorizationHeader
            {
                EndpointUri = "https://rramp.rstorcloud.io/migrations",
                HttpMethod = "GET",
                Service = "rramp",
                Region = "us-west-1"
            };
var authorization = signer.ComputeSignature(headers,
                                            "",   // no query parameters
                                           AWS4SignerBase.EMPTY_BODY_SHA
256,
                                           awsAccessKey,
                                           awsSecretKey);

using(client = new HttpClient()) {
client.DefaultRequestHeaders.Authorization = new
AuthenticationHeaderValue(authorization);
client.DefaultRequestHeaders.Add("Content-Type", "application/json");
client.DefaultRequestHeaders.Add(AWS4SignerBase.X_Amz_Content_SHA256,
AWS4SignerBase.EMPTY_BODY_SHA256);
var response = await client.GetAsync(signer.EndpointUri);
// here use the response.
```

6 Transporter CLI Readme

In this folder we have a sample Python application that works as CLI. We provide some samples of usage below.

6.1 Precondition

In order to use Transporter, you should have a Python ≥ 3.7 working environment and install the dependency:

```
$ pip3 install --user requests requests_aws4auth
```

6.2 Create a Migration

1. Prepare a migration descriptor in json, i.e. awstogcp.json
2. `user@transport-XPS-15-7590:~/data$ cat awstogcp.json`
3.
```
{"source":{"type":"s3v4","accessKey":"AWS_ACCESS_KEY",
             "secretKey":"AWS_SECRET_KEY","bucket":"rramp-
demo","region":"eu-west-1","endpoint":"https://s3.eu-west-
1.amazonaws.com"},"destination":
{"type":"gcp","accessKey":"GCP_ACCESS_KEY","secretKey":"GCP_SECRET_K
EY","bucket":"multipart02","region":"us-west-1","endpoint":"https://
storage.googleapis.com"}}
```
4. Launch transporter-cli.py with awstogcp.json as a parameter
5.
```
user@transport-XPS-15-7590:~/data$ transporter-cli.py -a ACCESS_KEY
-s
SECRET_KEY --host https://rramp.rstorcloud.io create awstogcp.json
```
6. `Creating migration from awstogcp.json`
7. `OK`

6.3 Create a Migration

We can list a migration with the following command:

```
user@transport-XPS-15-7590:~/data$ ./transporter-cli.py -a ACCESS_KEY -s
SECRET_KEY --host https://rramp.rstrocloud.io list --details
MIGRATION ID                          STATE    USER
CREATION
DATE             SPEED    OBJECTS FOUND  O. COMPLETED  O. FAILED  SOURCE
S.
BUCKET   DESTINATION  D. BUCKET
```

```
060aea83-3da5-4107-88a6-31fbbd1d3256  Running  user@rstor.io  2020-09-
25T16:32:31.413Z  1.8MB/s  6000           6000          0          s3v4
rramp-demo  gcp          multipart02
```

6.4 Abort a Migration

We can abort a migration with the following command:

```
user@transport-XPS-15-7590:~/data$ ./transporter-cli.py -a ACCESS_KEY -s
SECRET_KEY --host https://rramp.rstrocloud.io abort 060aea83-3da5-4107-
88a6-
31fbbd1d3256
Aborting migration 060aea83-3da5-4107-88a6-31fbbd1d3256
OK
```

6.5 Retry a Migration

We can retry a migration with the following command:

```
user@transport-XPS-15-7590:~/data$ ./transporter-cli.py -a ACCESS_KEY -s
SECRET_KEY --host https://rramp.rstor.pre-rstor.com retry 060aea83-3da5-
4107-
88a6-31fbbd1d3256
Retrying migration 060aea83-3da5-4107-88a6-31fbbd1d3256
OK
```

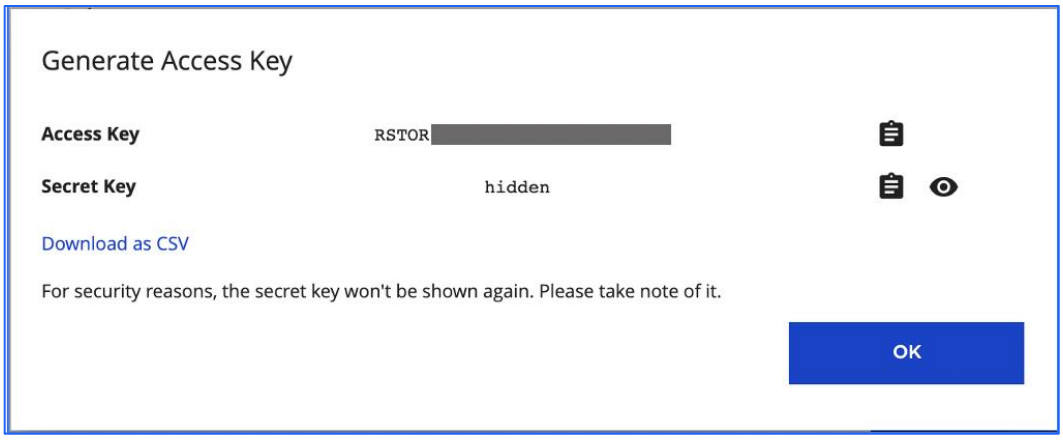
7 Retrieve Credentials from Different Cloud Storage Providers

7.1 RSTOR Cloud Portal

In the RSTOR example migration, the **endpoint URL** is https://s3.rstor.rstorcloud.io. For getting an AccessKey and a SecretKey use the following steps:

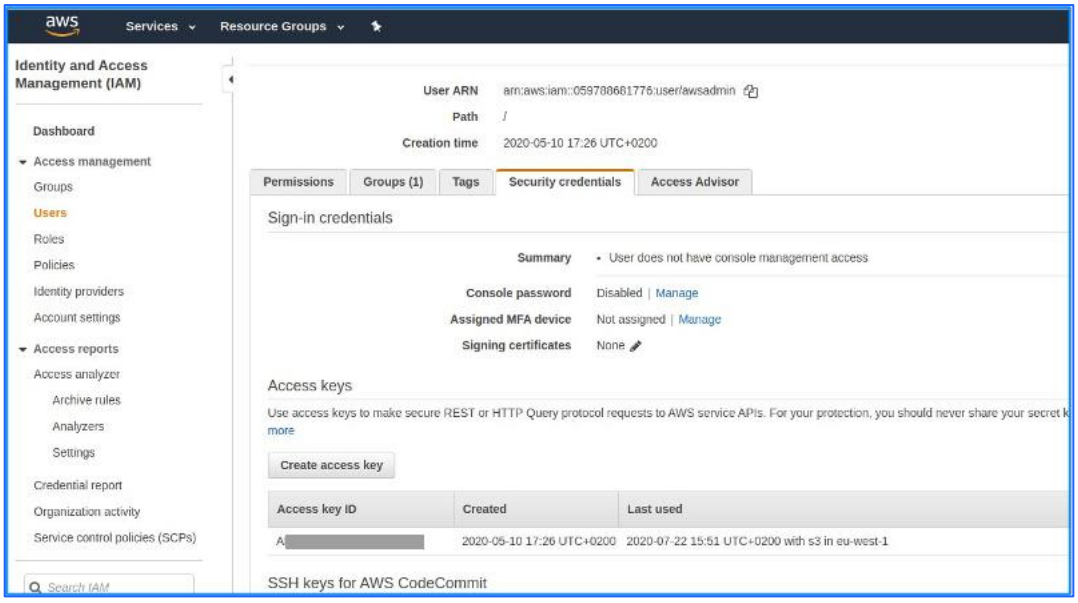
1. Go to the RSTOR Portal and click on your name in the upper right corner. From there choose “MY ACCOUNT”.
2. A screen will appear with a button called “+ GENERATE KEY” in the bottom right. Click on “+ GENERATE KEY”.

The system will show the following dialog:



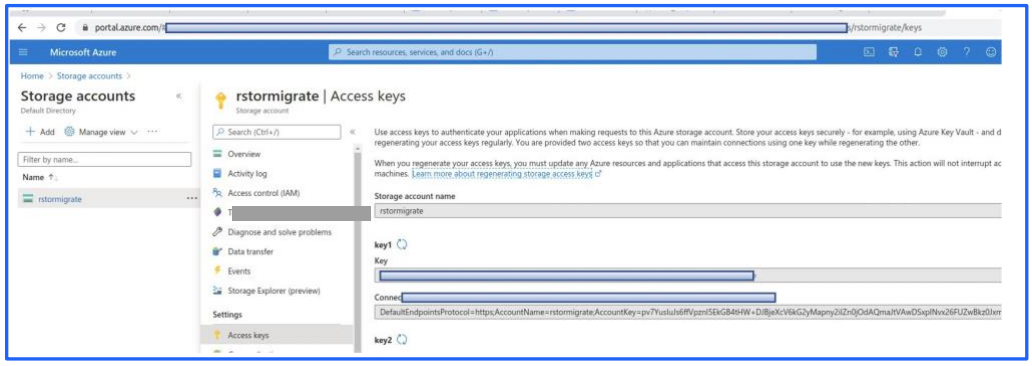
7.2 Amazon AWS S3

In Amazon AWS, the migration **endpoint URL** is https://s3.amazonregion.amazonaws.com. If we created the S3 bucket in us-east-1 we have: https://s3.us-east-1.amazonaws.com. For getting **AccessKey** and **SecretKey** in AWS go to IAM, go to Users, then click on “Create access key” near the bottom.



7.3 Microsoft Azure

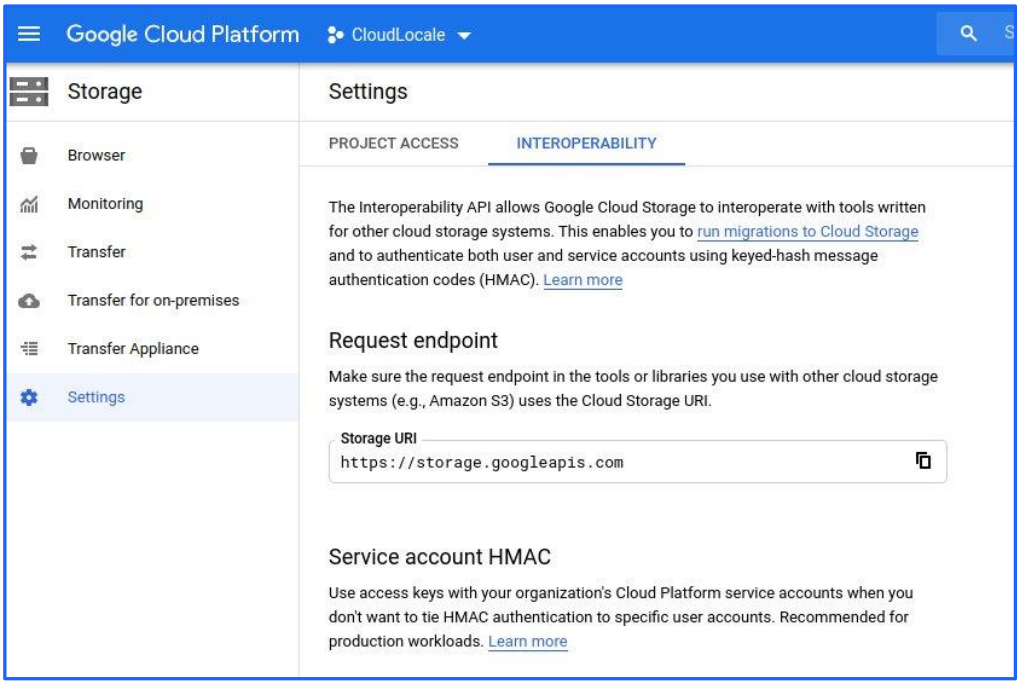
The storage provided by Microsoft Azure is called Azure Blob storage and it is based on a storage account that has one or more storage containers, in which we have different buckets. In this case, the name of the **AccessKey** is the name of the storage container and get the **SecretKey**.



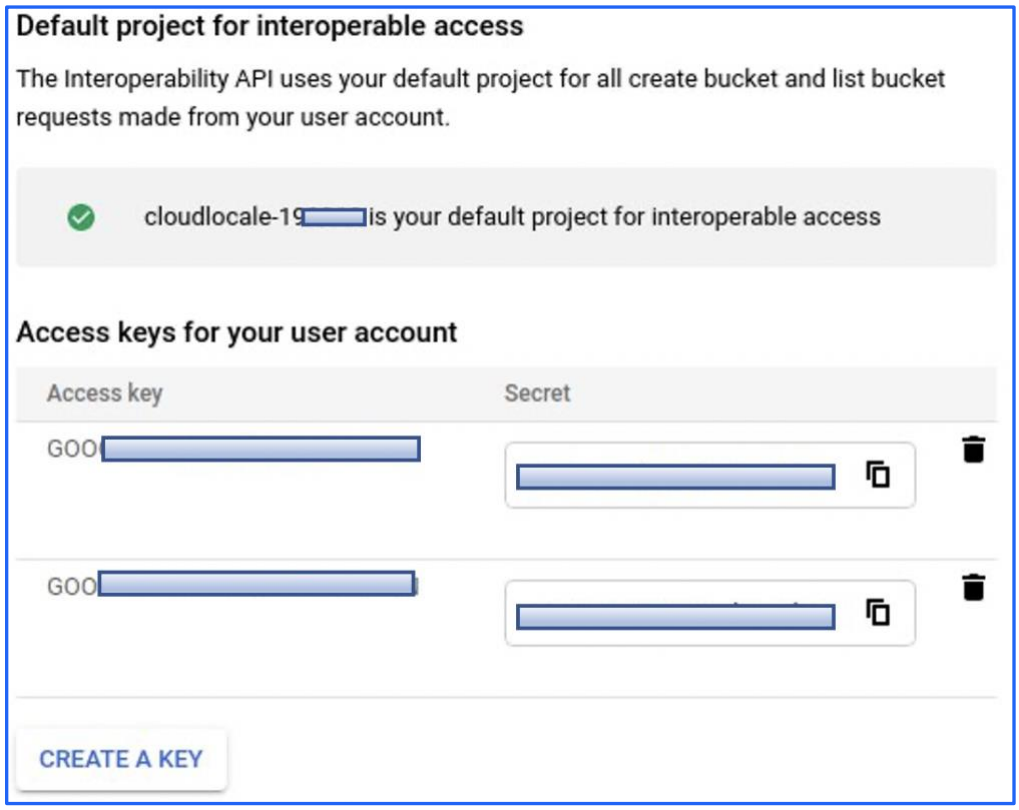
In the figure above we are inside the storage account for the storage container **rstormigrate** after having clicked Access Keys. In the case of Azure, the migration **Endpoint** is https://storagecontainername.blob.core.windows.net (in this case https://rstormigrate.blob.core.windows.net).

7.4 Google Cloud Platform

In the context of Google Cloud Platform, select the project, go to the Storage Browser, and then go to Settings.



Click "**INTEROPERABILITY**" for the StorageURL (https://storage.googleapis.com) to use the RSTOR Space Endpoint. Generate the AccessKey and Secret Key at the bottom of the page, needed for creating a new migration.


Default project for interoperable access
The Interoperability API uses your default project for all create bucket and list bucket requests made from your user account.
cloudlocale-19 is your default project for interoperable access
Access keys for your user account
Access key
Secret
GOO
GOO
CREATE A KEY