DATAAVENUE TUTORIAL

DataAvenue tutorial for MTA Cloud users v1.0

MTA Cloud team
info@mta.cloud.hu

Table of contents

Overview

Data Avenue is a data storage management service that enables to access different types of storage resources (including S3, sftp, GridFTP, iRODS, SRM servers) using a uniform interface. The provided REST API allows of performing all the typical storage operations such as creating folders/buckets, renaming or deleting files/folders, uploading/downloading files, or copying/moving files/folders between different storage resources, respectively, even simply using 'curl' from command line. Data Avenue automatically translates users' REST commands to the appropriate storage protocols, and manages long-running data transfers in the background.

In this tutorial we establish a cluster with two nodes types. On the DataAvenue node the DataAvenue application will run, and on a predefined number of storage nodes an S3 storage will run, in order to be able to try DataAvenue file transfer software such as making buckets, download or copy files. We used Ceph and Docker components to build-up the cluster.

Features

- creating two types of nodes through contextualisation
- using the nova resource handler
- using parameters to scale up storage nodes

Prerequisites

- accessing an Occopus compatible interface
    - Occopus tool can launch virtual machines on MTA Cloud, and we recommend the use of nova interface (the url of the nova interface can be found under Compute/ Access & Security / API Access / Identity menu)
    - **Important**: **the use of Occopus tool currently works only on the SZTAKI branch of the MTA Cloud**
- target cloud contains an Ubuntu 16.04 image with cloud-init support
    - This image file can be found under Compute/Images/Public/Ubuntu 16.04 LTS image

Steps

Installation of Occopus

The deployment of the DataAvenue cluster will be established with the help of Occopus tool, therefore we need to install the Occopus tool first. You can install the Occopus orchestration tool with just one command. For more information about Occopus itself, and how to install it, visit the following link. We recommend launching an Ubuntu-based virtual machine in MTA Cloud to install the Occopus tool on it.

Download

Occopus works based on descriptors. We have prepared the descriptors for the installation of the DataAvenue cluster for the end-users. Based on these descriptors, Occopus will build the infrastructure in the target cloud. They can be downloaded from the following link: tutorial.examples.dataavenue-cluster .Install descriptors on a virtual machine which runs Occopus.

Note: In this tutorial it uses nova resources. However, feel free to use any Occopus-compatible cloud resource for the nodes, but we suggest to instantiate all nodes in the same cloud.

1. Fill the resource section

Open the file **nodes/node_definitions.yaml** and edit the resource section of the nodes labelled by node_def:.

- you must select an Occopus compatible resource plugin
  - this will be the nova resource plugin
- you can find and specify the relevant list of attributes for the plugin
- you may follow the help on collecting the values of the attributes for the plugin
- you may find a resource template for the plugin in the resource plugin tutorials

It is important that end-users should personalize the node definition file to the user before launching. In this file, we add the resource identifiers we will use, such as project ID, virtual machine size, and so on. We can not provide these identifiers for the user, but they can be easily collected from the MTA Cloud web interface. For detailed assistance, visit this link or the documentation below. The downloadable package in this example contains the nova plugin resource template (which is the same when using MTA Cloud).

2. Creating firewall rules

Components in the infrastructure connect to each other, therefore several port ranges must be opened for the VMs executing the components. Log in to the MTA Cloud OpenStack interface. Under "Compute / Access & Security" you can create a new firewall rule by clicking the "Create Security Group" button. After creation, you can edit the firewall rule by clicking the "Manage Rules / Add Rule" button. Add the following ports to the security group:

- `TCP 22 (SSH)`
- `TCP 80 (HTTP)`
- `TCP 8080`

3. Authentication

Make sure your authentication information is set correctly in your authentication file. You must set your email and password in the authentication file. Setting authentication information is described here.

4. Set the size of the cluster

Update the number of storage nodes if necessary. For this, edit the **infra-dataavenue.yaml** file and modify the min and max parameter under the scaling keyword. Scaling is the interval in which the number of nodes can change (min, max). Currently, the minimum is set to 2 (which will be the initial number at startup).

```
- &S
    name: storage
    type: storage_node
    scaling:
        min: 2
```

Important: Keep in mind that Occopus has to start at least one node from each node type to work properly and scaling can be applied only for storage nodes in this example!

5. Edit variables

Optionally edit the "variables" section of the **infra-dataavenue.yaml** file. Set the following attributes:

- **storage_user_name** is the name of the S3 storage user
- **access_key** is the access key of the S3 storage user
- **secret_key** is the secret key of the S3 storage user

6. Import

Load the node definitions into the database:

```
occopus-import nodes/node_definitions.yaml
```

Make sure that the proper virtualenv is activated! If you have not done this before, use the following command to activate the Occopus virtual environment:

```
source occopus/bin/activate
```

Important: Occopus takes node definitions from its database when builds up the infrastructure, so importing is necessary whenever the node definition (file) changes!

7. Deployment

Start deploying the infrastructure.

```
occopus-build infra-dataavenue.yaml
```

8. The established infrastructure

After successful finish, the node with **ip address** and **node id** are listed at the end of the logging messages and the identifier of the newly built infrastructure is printed. You can store the identifier of the infrastructure to perform further operations on your infra or alternatively you can query the identifier using the **occopus-maintain** command.

```
List of nodes/ip addresses:
dataavenue:
  <ip-address> (dfa5f4f5-7d69-432e-87f9-a37cd6376f7a)
storage:
  <ip-address> (cae40ed8-c4f3-49cd-bc73-92a8c027ff2c)
  <ip-address> (8e255594-5d9a-4106-920c-62591aabd899)
77cb026b-2f81-46a5-87c5-2adf13e1b2d3
```

9. Save credentials

On the S3 storage nodes a user with predefined parameters will be created. The **access_key** will be the Username and the **secret_key** will be the Password, which are predefined in the **infra-dataavenue.yaml** file. Save user credentials into a file named **credentials** use the above command:

```
echo -e 'X-Key: 1a7e159a-ffd8-49c8-8b40-549870c70e73\nX-Username:
A8Q2WPCWAELW61RWDGO8\nX-Password:
FWd1mccBfnw6VHa2vod98NEQktRCYlCronxbO1aQ' > credentials
```

Note: This step will be useful to shorten the curl commands later when using DataAvenue!

10.Save ip adresses

Save the nodes' ip addresses in variables to simplify the use of commands.

```
export SOURCE_NODE_IP=[storage_a_ip]

export TARGET_NODE_IP=[storage_b_ip]

export DATAAVENUE_NODE_IP=[dataavenue_ip]
```

11.Make bucket

Make bucket on each S3 storage node:

```
curl -H "$(cat credentials)" -X POST -H "X-URI:
s3://$SOURCE_NODE_IP:80/sourcebucket/"
http://$DATAAVENUE_NODE_IP:8080/dataavenue/rest/directory


curl -H "$(cat credentials)" -X POST -H "X-URI:
s3://$TARGET_NODE_IP:80/targetbucket/"
http://$DATAAVENUE_NODE_IP:8080/dataavenue/rest/directory
```

Note: Bucket names should be at least three letter length. Now, the bucket on the source S3 storage node will be **sourcebucket**, and the bucket on the target S3 storage node will be **targetbucket**.

12.Check the result of making buckets

Check the bucket creation by listing the buckets on each storage node:

```
curl -H "$(cat credentials)" -H "X-URI: s3://$SOURCE_NODE_IP:80/"
http://$DATAAVENUE_NODE_IP:8080/dataavenue/rest/directory
```

The result should be: ["sourcebucket/"]

```
curl -H "$(cat credentials)" -H "X-URI: s3://$TARGET_NODE_IP:80"
http://$DATAAVENUE_NODE_IP:8080/dataavenue/rest/directory
```

The result should be: ["targetbucket/"]

13.Generate test file

To test the DataAvenue file transfer software you should make a file to be transferred. With this command you can create predefined sized file, now it will be 1 megabyte:

```
dd if=/dev/urandom of=1MB.dat bs=1M count=1
```

14.Upload file

Upload the generated **1MB.dat** file to the source storage node:

```
curl -H "$(cat credentials)" -X POST -H "X-URI:
s3://$SOURCE_NODE_IP:80/sourcebucket/1MB.dat" -H 'Content-Type:
```

```
application/octet-stream' --data-binary @1MB.dat
http://$DATAAVENUE_NODE_IP:8080/dataavenue/rest/file
```

15.Check the result of uploading file

Check the uploaded file by listing the **sourcebucket** bucket on the source node:

```
curl -H "$(cat credentials)" -H "X-URI:
s3://$SOURCE_NODE_IP:80/sourcebucket"
http://$DATAAVENUE_NODE_IP:8080/dataavenue/rest/directory
```

The result should be: ["1MB.dat"]

16.Save target node's credentials

Save the target node's credentials to a **target.json** file to shorten the copy command later:

```
echo
"{target:'s3://"$TARGET_NODE_IP":80/targetbucket/',overwrite:true,creden
tials:{Type:UserPass, UserID:"A8Q2WPCWAELW61RWDGO8",
UserPass:"FWd1mccBfnw6VHa2vod98NEQktRCYlCronxbO1aQ"}}" > target.json
```

17.Copy file

Copy the uploaded 1MB.dat file from the source node to the target node:

```
curl -H "$(cat credentials)"  -X POST -H "X-URI:
s3://$SOURCE_NODE_IP:80/sourcebucket/1MB.dat" -H "Content-type:
application/json" --data "$(cat target.json)"
http://$DATAAVENUE_NODE_IP:8080/dataavenue/rest/transfers > transferid
```

The result should be: [transfer_id]

18.Check the result of copying

Check the result of the copy command by querying the **transfer_id** returned by the copy command:

```
curl -H "$(cat credentials)"
http://$DATAAVENUE_NODE_IP:8080/dataavenue/rest/transfers/$(cat
transferid)
```

The following result means a successful copy transfer from the source node to the target node (see status: DONE):

```
"bytesTransferred":1048576,"source":"s3://[storage_a_ip]:80/sourcebucket
/1MB.dat","status":"DONE","serverTime":1507637326644,"target":"s3://[sto
rage_b_ip]:80/targetbucket/1MB.dat","ended":1507637273245,"started":1507
637271709,"size":1048576
```

19.List files in a bucket

You can list the files in the target node's bucket, to check the 1MB file:

```
curl -H "$(cat credentials)" -H "X-URI:
s3://$TARGET_NODE_IP:80/targetbucket"
http://$DATAAVENUE_NODE_IP:8080/dataavenue/rest/directory
```

The result should be: ["1MB.dat"].

20.Download file from a bucket

Also, you can download the copied file from the target node:

```
curl -H "$(cat credentials)" -H "X-URI:
s3://$TARGET_NODE_IP:80/targetbucket/1MB.dat" -o download.dat
http://$DATAAVENUE_NODE_IP:8080/dataavenue/rest/file
```

21.Delete the infrastructure

Finally, you may destroy the infrastructure using the infrastructure id returned by **occopus-build**

```
occopus-destroy -i 77cb026b-2f81-46a5-87c5-2adf13e1b2d3
```

Note: In this tutorial we used HTTP protocol only. DataAvenue also supports HTTPS on port 8443; storages could also be accessed over secure HTTP by deploying e.g. HAPROXY on their nodes

Report bugs or any other project related claims

Communication and support for MTA Cloud services are in the form of email. The common e-mail address is info@cloud.mta.hu. A notification form generated from this error will be generated by a designated member of the MTA Cloud team.