Dremio Software

Migrate Catalog Objects Between Environments

Introduction

Dremio customers frequently adopt modern CI/CD practices for moving catalog objects from development to testing and production environments. This process requires careful planning, execution, and expertise to ensure a smooth transition. This comprehensive guide will explore the best practices, strategies, and considerations involved in successfully migrating semantic layer catalog objects like datasets, sources, and role-based access controls (RBAC) between Dremio environments.

CI/CD with Dremio

Dremio provides REST APIs that can be combined with Python or other scripting languages to automate much of the migration process. In this doc, we'll cover some options for tooling and architecture of different approaches to leverage the REST API into customers' existing CI/CD processes.

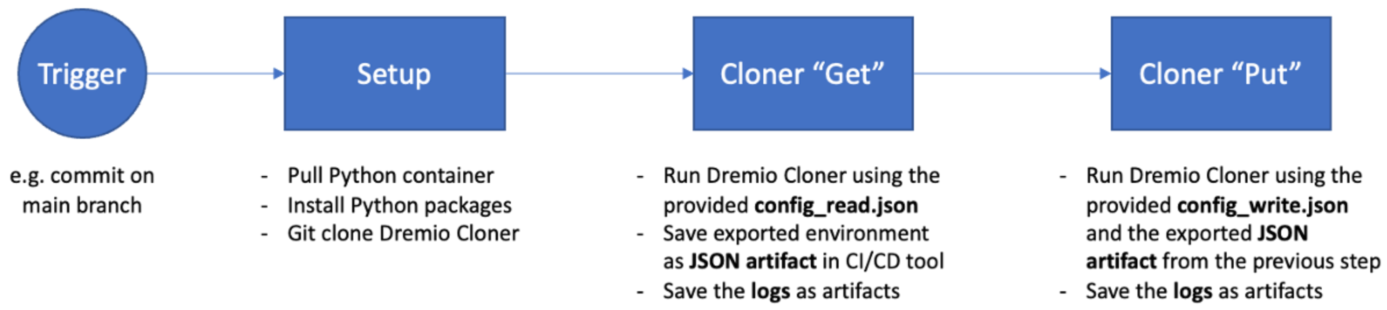
Each customer can develop their own scripts using the Dremio APIs. The Dremio Professional Services team has also developed a helpful script named Dremio Cloner that can optionally be used. Using Dremio Cloner allows you to get a fast start on scripting but is not required.

Introducing Dremio Cloner

Dremio Cloner is an open-source Python-based utility built by the professional services team at Dremio that uses Dremio's REST API to automate the extraction of objects and creation in another environment. It can be used for various purposes such as migrating entire Dremio environments, CI/CD processes, disaster recovery scenarios, partial backup/restore, security audit reporting, and reflection reporting.

Dremio Cloner is executed with a command that includes a configuration file in JSON format. The configuration file defines the Dremio environment and configuration. Below is a simple workflow of how to execute Dremio Cloner to facilitate a migration.

Simple Dremio Cloner CI/CD Pipeline



e.g. commit on main branch

- Pull Python container
- Install Python packages
- Git clone Dremio Cloner

- Run Dremio Cloner using the provided **config_read.json**
- Save exported environment as **JSON artifact** in CI/CD tool
- Save the **logs** as artifacts

- Run Dremio Cloner using the provided **config_write.json** and the exported **JSON artifact** from the previous step
- Save the **logs** as artifacts

Cloner Prerequisites

The prerequisites to running Dremio Cloner are as follows:

1. Any client machine that can connect to your Dremio environments via REST API endpoints (typically on port 9047). If you are following best practices, you would want this client machine to have connectivity to your separate environments if you have SSL enabled.

   a. Alternatively, if SSL is not enabled and you can reach your separate environments from each of the coordinators, you can install the libraries and cloner on the coordinator of your source system.
   b. Make sure that wherever you are looking to install Dremio cloner you can reach your source and destination coordinator nodes.

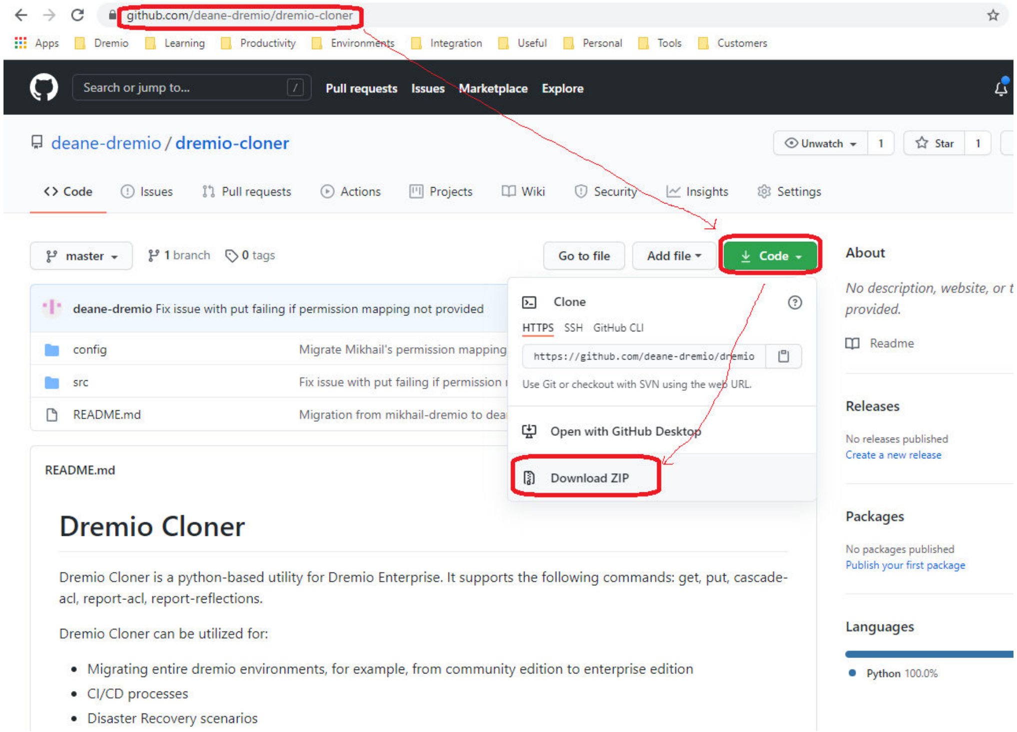
2. Python version 3.5 or later and required Python Libraries:

```Python
$ pip3 install mo-sql-parsing requests
```

3. SQL Parsing Library: If you are using Dremio Cloner, you also need to install the sqlparse library as a prerequisite. This library is used for parsing and reformatting SQL queries.

```Python
$ pip3 install sqlparse
```

4. Install Dremio Cloner: Once the dependencies are installed, you can download the Dremio Cloner code from the GitHub repository.



deane-dremio Fix issue with put failing if permission mapping not provided

config — Migrate Mikhail's permission mapping

src — Fix issue with put failing if permission

README.md — Migration from mikhail-dremio to dea

Use Git or checkout with SVN using the web URL.

No description, website, or t provided.

Dremio Cloner

Dremio Cloner is a python-based utility for Dremio Enterprise. It supports the following commands: get, put, cascade-acl, report-acl, report-reflections.

Dremio Cloner can be utilized for:

- Migrating entire dremio environments, for example, from community edition to enterprise edition
- CI/CD processes
- Disaster Recovery scenarios

- Unzip the package onto a machine that can successfully make calls to the REST API of your Dremio instance. You can run Dremio Cloner from either a Windows or Linux machine. In this document, we assume the tool is copied onto a Linux machine and unzipped into /opt/tools/dremio-cloner.

```
unzip dremio-cloner-master.zip -d /opt/tools
mv /opt/tools/dremio-cloner-master /opt/tools/dremio-cloner
```

- The tool also requires Python 3.5 or greater.

```
python -V
```

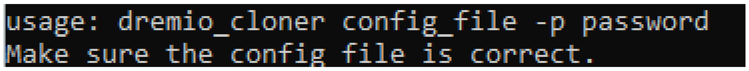
- If you are not presented with details of the currently installed Python version, then you will need to install Python. The following site provides an explanation if required:

  https://docs.python-guide.org/starting/installation/

- Navigate to the `dremio-cloner` directory and issue the following command:

```
python dremio_cloner.py
```

  You should be presented with the following output which indicates the tool is installed and is executable:



```
usage: dremio_cloner config_file -p password
Make sure the config file is correct.
```

  That's it! You have successfully installed Dremio Cloner.

Bacup Target Destination

Shut down your destination Dremio environment and take a backup to ensure there are no background metadata refresh jobs or reflection refresh jobs in the process.

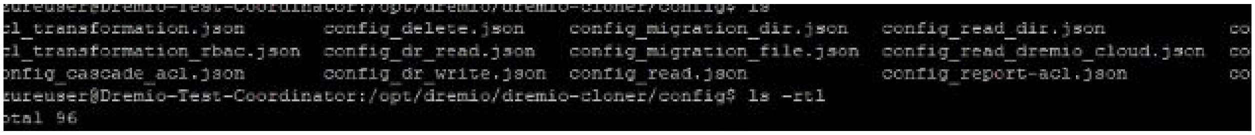
Generate PAT Tokens (Optional)

Create PAT Tokens and save generated tokens for Dremio Cloner to be able to authenticate to source and target Dremio environments

Configure Cloner for Reading/Get

To edit the "get" file in Dremio Cloner, you need to follow these steps:

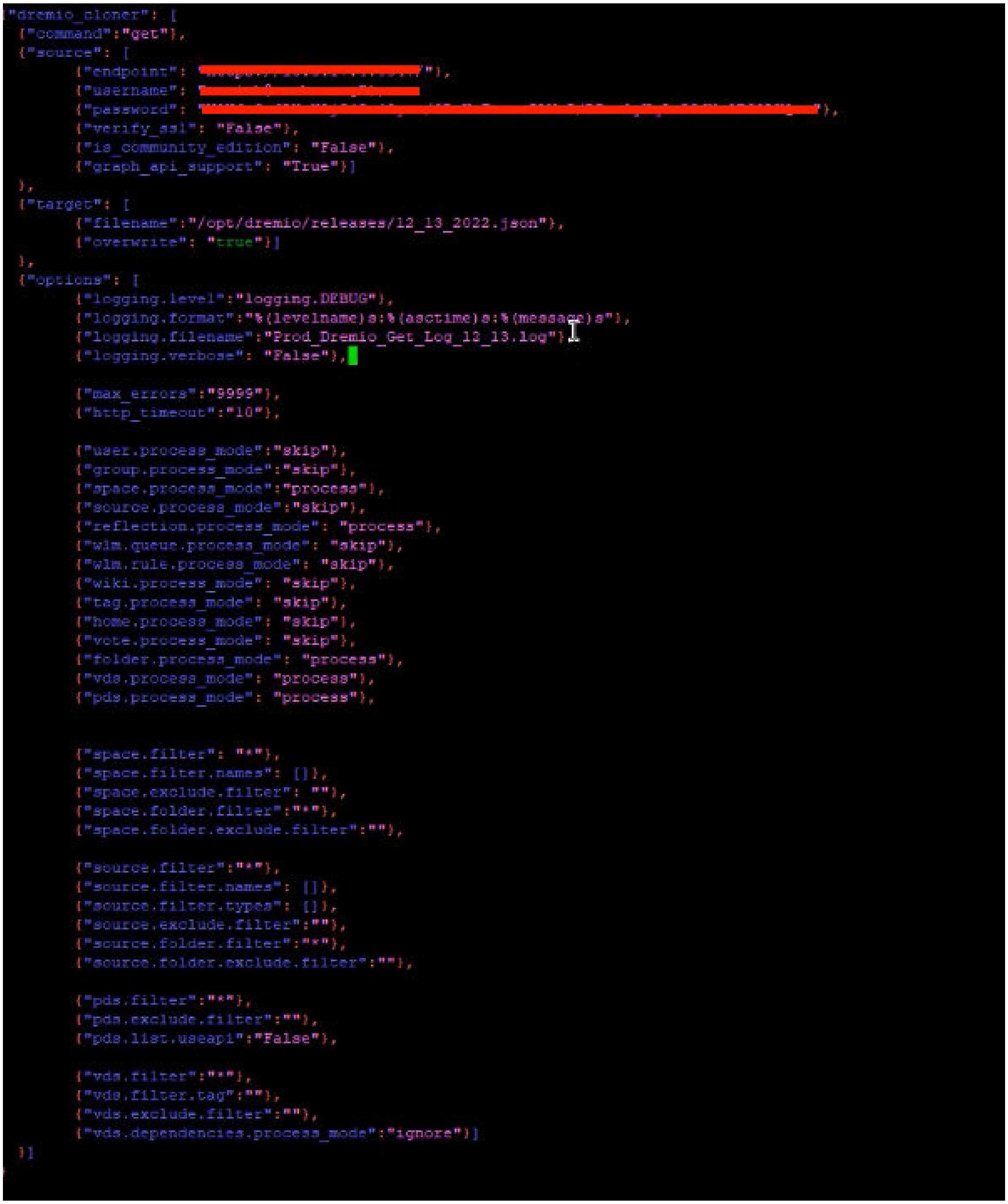
1. Locate the get file: The get file is a JSON file that contains the configuration for the objects you want to retrieve from a Dremio environment. By default, the get file is named "config_read.json" and is located in the /config directory of the Dremio Cloner project.



```
zureuser@Dremio-Test-Coordinator:/opt/dremio/dremio-cloner/config$ ls
cl_transformation.json       config_delete.json    config_migration_dir.json   config_read_dir.json           co
cl_transformation_rbac.json  config_dr_read.json   config_migration_file.json  config_read_dremio_cloud.json  co
onfig_cascade_acl.json       config_dr_write.json  config_read.json            config_report-acl.json         co
zureuser@Dremio-Test-Coordinator:/opt/dremio/dremio-cloner/config$ ls -rtl
otal 96
```

2. Open the get file: You can use any text editor or IDE to open the get file. Make sure to preserve the JSON format while editing.
3. Modify the objects: Inside the get file, you will find an array of objects, each representing a Dremio object type (e.g., Source, Space, Folder, PDS, VDS, etc.). You can edit the properties of these objects according to your requirements.
4. Pay attention to the object structure: Each object in the get file has a specific structure with mandatory and optional properties. Make sure to understand the structure of the object you are editing and provide the necessary properties. Refer to the Dremio Cloner documentation for detailed information on the object structure.
5. Save the get file: Once you have finished editing the get file, save it with the changes. When editing the get file, the modifications you make will depend on your specific requirements. Here are some common modifications you might consider:
    a. Specify the objects you want to retrieve: In the get file, you can specify the objects you want to retrieve from the Dremio environment. For example, you can include specific Sources, Spaces, Folders, PDS, VDS, ACLs, Reflections, etc. by modifying the properties of each object in the get file.
    b. Filter objects based on criteria: You can add filters to the get file to retrieve objects that meet certain criteria. For example, you can filter Sources based on their type, filter Spaces based on their name, or filter datasets via Regular Expressions.
    c. Exclude certain objects: If there are specific object types that you do not want to retrieve, you can remove them from the get file or set their "enabled" property to false
    d. Customize the output format: By default, the get command saves the retrieved objects into a JSON file. It's important to note that the specific modifications you make to the get file will depend on your specific use case and requirements. Make sure to refer to the Dremio Cloner documentation for detailed information on the properties and options available for each object type.

Once Completed the get file should look something like this:



```
{"dremio_cloner": [
  {"command":"get"},
  {"source": [
        {"endpoint": [redacted]"},
        {"username": [redacted]
        {"password": [redacted]"},
        {"verify_ssl": "False"},
        {"is_community_edition": "False"},
        {"graph_api_support": "True"}]
  },
  {"target": [
        {"filename":"/opt/dremio/releases/12_13_2022.json"},
        {"overwrite": "true"}]
  },
  {"options": [
        {"logging.level":"logging.DEBUG"},
        {"logging.format":"%(levelname)s:%(asctime)s:%(message)s"},
        {"logging.filename":"Prod_Dremio_Get_Log_12_13.log"},
        {"logging.verbose": "False"},

        {"max_errors":"9999"},
        {"http_timeout":"10"},

        {"user.process_mode":"skip"},
        {"group.process_mode":"skip"},
        {"space.process_mode":"process"},
        {"source.process_mode":"skip"},
        {"reflection.process_mode": "process"},
        {"wlm.queue.process_mode": "skip"},
        {"wlm.rule.process_mode": "skip"},
        {"wiki.process_mode": "skip"},
        {"tag.process_mode": "skip"},
        {"home.process_mode": "skip"},
        {"vote.process_mode": "skip"},
        {"folder.process_mode": "process"},
        {"vds.process_mode": "process"},
        {"pds.process_mode": "process"},


        {"space.filter": "*"},
        {"space.filter.names": []},
        {"space.exclude.filter": ""},
        {"space.folder.filter":"*"},
        {"space.folder.exclude.filter":""},

        {"source.filter":"*"},
        {"source.filter.names": []},
        {"source.filter.types": []},
        {"source.exclude.filter":""},
        {"source.folder.filter":"*"},
        {"source.folder.exclude.filter":""},

        {"pds.filter":"*"},
        {"pds.exclude.filter":""},
        {"pds.list.useapi":"False"},

        {"vds.filter":"*"},
        {"vds.filter.tag":""},
        {"vds.exclude.filter":""},
        {"vds.dependencies.process_mode":"ignore"}]
  }]
}
```

Configure Dremio for Writing/Put

Once you have generated the “get” JSON files used to produce the objects you want to recreate in your destination environment, there are some best practices to follow to migrate your objects over successfully.

Typically we would want to split the write into 3 separate steps depending on your environment configuration. These steps include using 3 separate JSON files to:

1. Write the source and connection parameters either via JSON file or Manually
2. Write PDS objects and promote them to the target destination
3. Write the Spaces, folders, VDSs, wiki, and tags into the target Dremio environment.

For source connections, using the get JSON is optional in the migration process and is only required if you choose to deploy sources using Dremio Cloner rather than create them manually directly. For this operation to succeed, it requires the JSON file containing the exported objects to be edited manually to supply passwords for all data sources directly in the file, which is often deemed a security risk, hence the recommendation to create the data sources manually in your target Dremio environment. Additionally, your connectivity may vary between environments.

Filtering Objects

If there are objects including spaces or folders that need to be omitted from migration, in the get and put files you have the option to configure these using the exclude.filter configuration parameter. For example:

For example, the configuration parameter "**source.filter**" is used to establish a filter that determines which Sources should be considered for processing. When setting it to "*", it will encompass all Sources. If the field is set to an empty string “”, it will exclude all Sources. The asterisk (wildcard) "*" can be utilized multiple times in the filter to define a matching pattern. It's important to note that folders must be separated using a backslash. This filter operates in conjunction with the "source.exclude.filter" parameter, utilizing a logical AND relationship.

Meanwhile the configuration parameter "**source.exclude.filter**" works in the opposite logical direction and is used to set up a filter that determines which Spaces should not be processed. When using "*", it will exclude all Spaces. Leaving the field as an empty string will include all sources. You can use the asterisk (wildcard) "*" multiple times in the filter to establish a pattern. It's important to note that folders need to be separated using a backslash. This filter operates in conjunction with the "source.filter" parameter, using a logical AND relationship.

Employing this combination of inclusion and exclusion filters for Source, Space, PDS, VDS, Reflection, User, and Group parameters provides you with the capability to precisely tailor the scope of desired objects to be moved from one environment to another.

Detailed information about these configurations can be found here:

https://github.com/deane-dremio/dremio-cloner/blob/master/README.md

Alternatives to Dremio Cloner

In addition to Dremio Cloner, there are a growing number of alternative tools that can be used to migrate Dremio. A tool that is gaining usage from our customer and partner community is dbt Core by dbt for which Dremio is maintaining an official connector. Dbt's primary focus, as an ETL and orchestration tool, is around versioning and writing SQL definitions to targets like Dremio. Dbt also uses the Dremio REST APIs to automate catalog object migration.

For references, check out our whitepaper here
https://www.dremio.com/wp-content/uploads/2022/12/Dremio_dbt_BetterTogether.pdf

Along with instructions to get started here:
https://docs.getdbt.com/docs/core/connect-data-platform/dremio-setup

Conclusion

Dremio provides the capability to integrate semantic layer content into your CI/CD process by exposing REST APIs. You can call those APIs directly or you can use pre-developed scripts such as Dremio Cloner. Dremio Cloner is open source and provides a lot of flexibility for configuration and extension to help you migrate objects and connections between your Dremio environments. You can also use the Cloner Python code as a starting point for your own scripts.