

[On this page](#)

Privileges

Based on [the inheritance model](#), when a [user](#) or [role](#) is granted privileges to an object with child objects, those privileges also apply to the child objects, regardless of an [object's ownership](#). This is also known as [scope](#), meaning the limits to which a privilege applies.

By default, all users are granted the [PUBLIC role](#). However, privileges must be manually applied to each object.

Object Hierarchy

Each object, such as a project, cloud, or dataset, exists within a hierarchy of “containers.” The uppermost container exists as the system administrator. All other objects are contained within sources or spaces and then organized into folders. The hierarchy of these objects is illustrated and described further on the [Architecture help page](#).

All Supported Privileges

Note:

Users may only see and use child objects if they've been given access to its parent objects. For example, a user cannot use [SELECT](#) or [ALTER](#) privileges on a dataset until they've been granted the [USAGE](#) privilege for the project housing those datasets.

If a user lacks the necessary privileges to edit or use an object, they will see a permissions error message indicating such.

Dataset Privileges

Warning:

The [VIEW SCHEMA](#) privilege was previously included in the following table. However, [VIEW SCHEMA](#) has not been implemented and that entry has been removed.

Privilege	Target Objects	Description
ALL	Project, Space, Source, Folder/Schema, Dataset/View	Grant the user all possible privileges for an object type, except MANAGE GRANTS and OWNERSHIP .

	dremio Documentation	Dremio Cloud ▾	
ALTER	Folder/Schema, Dataset/View	managing metadata, such as Metadata Refresh and Forget.	
ALTER REFLECTION	Project, Source, Space, Folder/Schema	Create, edit, and view reflections on all datasets in scope. Includes granting access to all interfaces, such as the Dataset Reflection pages, Administrator Reflection pages, and any REST API endpoints.	
DELETE	Project, Source, Space, Folder/Schema, Dataset/View	Execute the associated DML operation on all datasets in scope. This is only supported with Apache Iceberg datasets.	
DROP	Project, Source, Folder/Schema	Drop tables on any dataset in the scope.	
EXECUTE	Dataset/View	Execute the associated user-defined function (UDF) for the purposes of querying tables/view with row-access or column-masking filters applied. Only the owner of a UDF may view or edit the associated function. Users or roles must have this privilege in order to apply filtering and masking policies.	
INSERT	Project, Source, Space, Folder/Schema, Dataset/View	Execute the associated DML operation on all datasets in scope. This is only supported with Apache Iceberg datasets.	
MANAGE GRANTS	Organization, Project, Engine, Cloud, Source, Space, Folder/Schema, Dataset/View	Modifies the privileges of all objects in the set scope. Also changes the owner of all objects within the scope.	
OPTIMIZE	Iceberg tables	Rewrite data files using compaction. Logically combine small files into larger files or split large files to reduce metadata overhead and runtime file open costs. This functionality also enables you to repartition the data when a table's partition has changed. Dremio admins, the table's owner, and users with the SELECT and UPDATE privileges can use the optimize functionality.	



dremio
Documentation

Dremio Cloud ▾







dremio
Documentation

Dremio Cloud ▾







dremio
Documentation

Dremio Cloud ▾





dremio
Documentation

Dremio Cloud ▾





dremio
Documentation

Dremio Cloud ▾





dremio
Documentation

Dremio Cloud ▾






OWNERSHIP


Organization, Cloud,
Project, Engine, Space,
Source,
Folder/Schema,

Allows all actions on the object and objects within the object. Actions include modifying object settings, granting/revoking user and role access, and deleting the object.

	dremio Documentation	Dremio Cloud ▾	☰
ROLLBACK	Iceberg tables	Roll back your Iceberg tables to a previous state. Dremio admins, the table's owner, and users with the INSERT , UPDATE , and DELETE privileges can use the rollback functionality.	
SELECT	Project, Source, Space, Folder/Schema, Dataset/View	Gives the ability to execute SELECT queries in the scope.	
TRUNCATE	Project, Source, Space, Folder/Schema, Dataset/View	Execute the associated DML operation on all datasets in scope. This is only supported with Apache Iceberg datasets.	
UPDATE	Project, Source, Space, Folder/Schema, Dataset/View	Execute the associated DML operation on all datasets in scope. This is only supported with Apache Iceberg datasets.	
VIEW REFLECTION	Project, Source, Space, Folder/Schema	View Reflections on all datasets in the scope. Includes access to all Dremio interfaces, such as the Dataset Reflection pages, Administrator Reflection pages, and any REST API endpoints.	

Organization, Project, Cloud, Engine, Source, & Space Privileges

Privilege	Target Objects	Description
CREATE BILLING ACCOUNT	Organization	Creates a new billing account, which is used to handle usage invoices if you are using Dremio Enterprise edition . The account creator is the default owner.
CREATE CLOUD	Organization	Create a new cloud. Users may not add clouds without this privilege. The cloud creator is the default owner for the cloud.
CREATE PROJECT	Organization	Create a new project. The project creator is the default owner of the project.
CREATE ROLE	Organization	Create a role. The user responsible for its creation automatically becomes its owner.

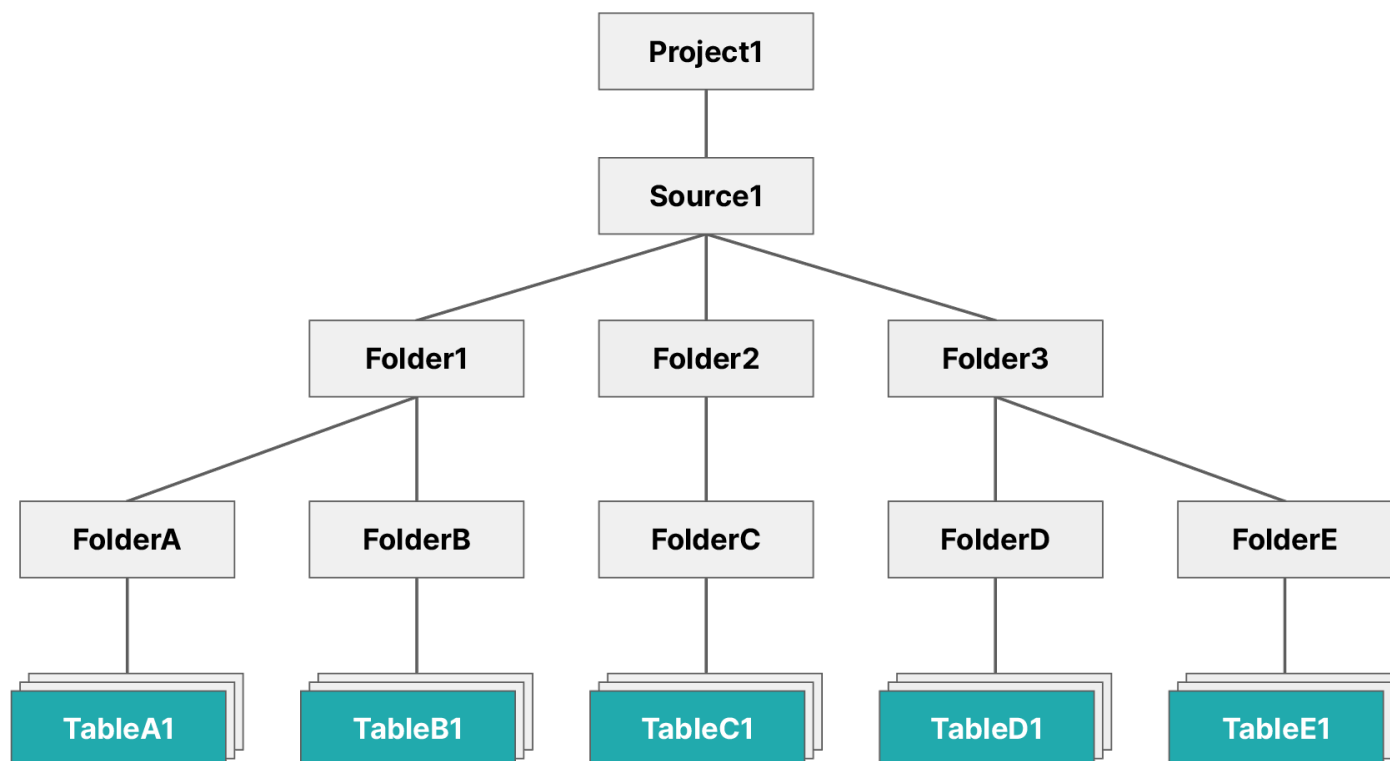
 dremio Documentation Dremio Cloud ☰		
CREATE USER	Organization	Create a user. The user responsible for its creation automatically becomes its owner.
EXTERNAL QUERY	Project, Source	Run the external_query table function on the source.
MODIFY	Project, Cloud, Engine, Source, Space	<p>Edit and delete an object. The following conditions apply:</p> <ul style="list-style-type: none"> • If Space or Source, edit the object's settings. • If Project, edit workload management settings and change support key settings.
MONITOR	Project, Cloud, Engine	Allow the user to read all current object settings.
OPERATE	Project, Engine	Give the ability to start and stop an engine, as well as blacklist any node in the scope. If this is set at the Project level, it applies to all Engines.
USAGE	Project, Engine	<p>Give the ability to use the object. Without this privilege, users cannot access datasets contained in a project or run queries using the available engines.</p> <ul style="list-style-type: none"> • Projects: This privilege is required for a user to access the project and its datasets. • Engines: This privilege is required for a user to run queries against the engine. By default, engines grant all users the USAGE privilege, but this can be revoked or otherwise changed.
VIEW JOB HISTORY	Project	Give the ability to view the job history of all users from the Jobs page.

Understanding Inheritance, Scope, and Ownership

Inheritance



user/role the same privilege access to any objects in the project that exist currently or are created in the future, whether it is a dataset or source. Without privileges based at the project level, even basic users cannot view, query, or alter the datasets it houses.



For example, giving a user privileges to [ALL DATASETS](#) will only grant the user access to all existing datasets in that project, not the folders that contain the datasets—which means they cannot access the dataset without first having the same privilege applied to the parent folder, and then the source, and then the project.

The following rules apply:

- The object to which a user's privileges are applied affects what is [known as the scope](#), and, like inheritance, follows a parent-child relationship with regard to inheritance.
- Even if a user only requires access to a single dataset, they must have access to the parent object(s) that contain it.

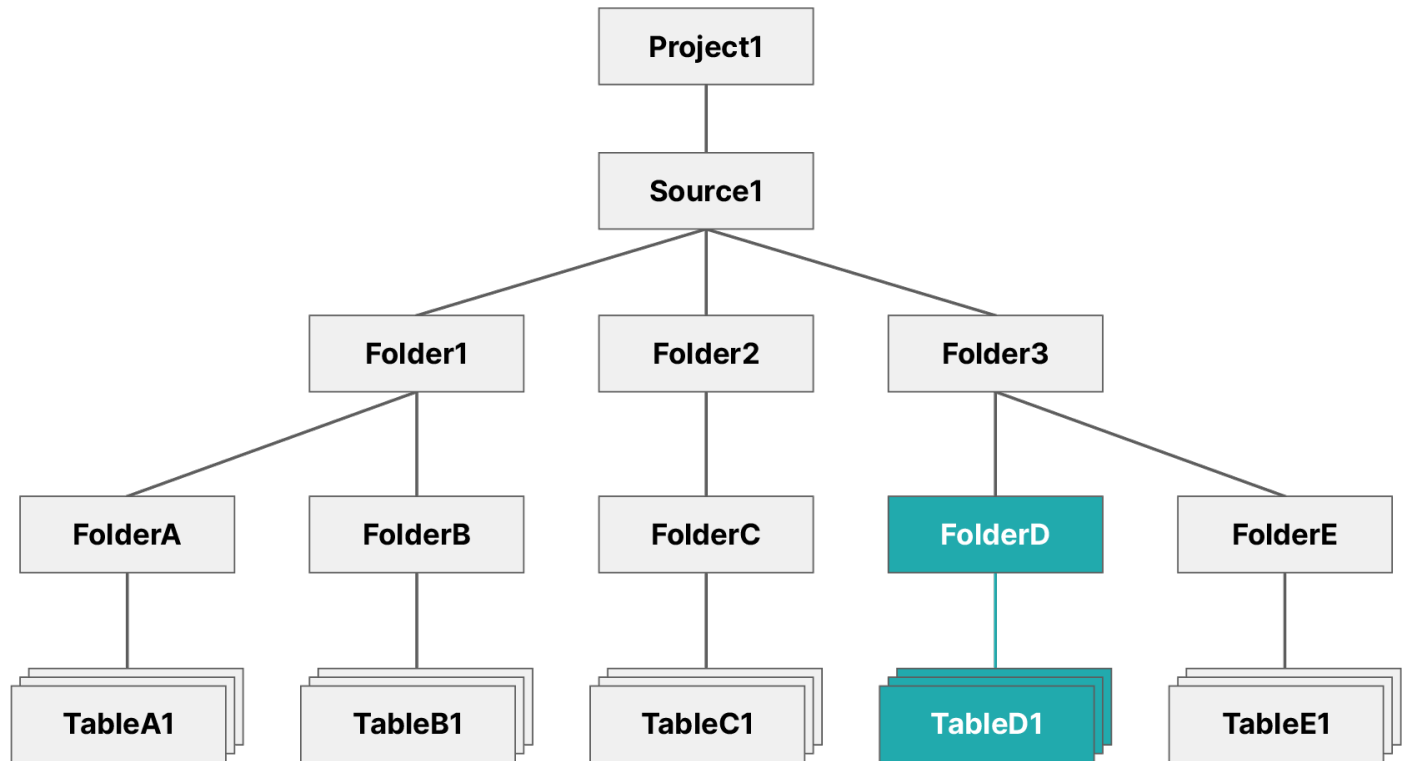
Note:

If a user has sufficient privileges for a single table, they may create a view based on that dataset, but with the user now having [ALTER](#) and [MANAGE GRANTS](#) privileges for any view. However, the user still retains the same privileges they held before with the original table, meaning the extended access granted to the view will not extend to the table it is based upon. For more information, see [View Delegation](#) below.

Scope



for any given object.



The following rules apply:

- As described by [the inheritance model](#) any objects contained within that folder will likewise apply the privileges a user has for a parent object, such as [FolderA](#).
- The user's access also extends to all existing and future datasets contained in that folder.

For example, [user1](#) is granted the [SELECT](#) privilege for [FolderD](#). This object contains multiple datasets, which the user may now query (but not edit, as [ALTER](#) was not granted).

Because of the established scope, [user1](#) may not access [FolderC](#), because they were only granted access to [FolderD](#) and its child objects.

Current vs. Future Objects

Based on the selected scope, you may restrict a user's access to future and existing datasets. For example, if you set privileges for a single table as the scope of a user's privilege, then that user may only perform the associated action to that dataset. However, if the scope of a privilege is instead set at the folder level, then the user may perform the granted privilege to all tables and views contained in that folder.

Privilege Delegation

an owner, which is automatically granted to the user that initially created the object. For example, if a system administrator creates an S3 data source, Dremio automatically assigns them ownership.

The implication of ownership is that only an object's owner retains all privileges for that object. As a result, the owner is able to grant or revoke user/role access to that object and any child objects associated, modify the object's settings, and also drop or delete the object as desired.

The following behaviors or limitations apply to ownership:

- Each object may only have one (1) owner
- An object's creator is automatically granted ownership
- Object ownership may be assigned or modified via [GRANT TO USER](#) or [GRANT TO ROLE](#) commands
- If an owner is deleted or removed, access to the object may not work
- Object owners may be identified by querying [sys.project."tables"](#) or [sys.project.views](#) for views
 - If an object has no owner, the `owner_id` will display as `$unowned`

View Delegation

Tables with restricted access may be used with greater access through the creation of a view. When a user with [SELECT](#) access to a table creates a view, then that user automatically becomes owner of the new object. They now have the ability to query and alter settings or data as desired. However, changes made to the view will not affect the original table.

Upon creating the view, the same rules of ownership [as described above apply](#). The owner or delegation identity does not change when a view is edited or queried, but must be manually changed via the [GRANT TO USER](#) or [GRANT TO ROLE](#) commands. To identify the owner of a view, query the `sys.views` table.

Note:

The shared view still selects from the underlying dataset using the view owner's permissions at the time of the view's last modification, even if the end user querying the view lacks privileges to modify the underlying table. This applies to each table on the data graph and chain of datasets.

Scenarios for Assigning Privileges

When assigning privileges to users via SQL commands, you may utilize these methods: [granting to a single dataset](#), [granting to ALL DATASETS](#), and [granting to a scope](#). Examples may be found

**Note:**

These examples assume users already have the correct privileges assigned at the project level to access child folders and datasets.

Each of these examples includes an SQL command.

1. Granting to a Single Dataset

When you have a user that needs access to only one table and no other objects, then you would simply assign them privileges for that dataset.

You should use this method if you want to restrict a user's access from any other existing or future datasets.

Note:

If you're granting the user access to a table, then remember that they'll be able to create views based on that table, which that user can then grant access to other users.

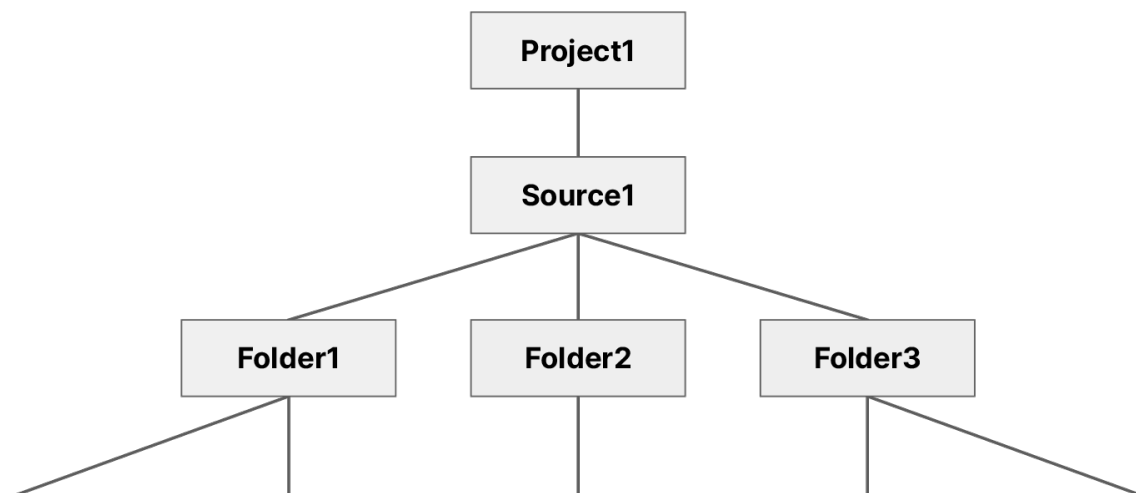
Example: Single dataset

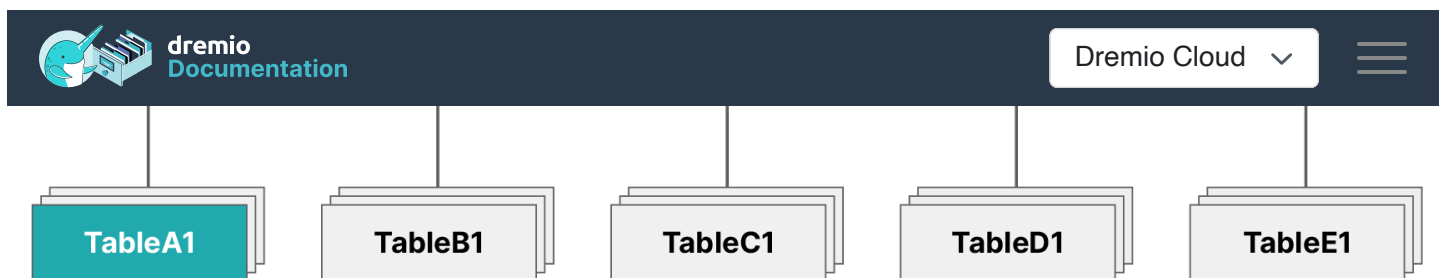
You have a user that you only want to give access to an individual table. You would need to navigate to the *Privileges* screen from that dataset's settings and grant the user the [SELECT](#) privilege, or perform the following command from the SQL Editor:

Example command for single dataset

```
GRANT SELECT ON TABLE TableA1 TO USER user1
```

The image below illustrates the objects [user1](#) now has access to.





This restricts `user1` so that they may only access the `TableA1` table, not any other datasets contained in the same folder. However, `user1` may still create views based on `TableA1`.

2. Granting to ALL DATASETS

When you have a user that needs access to all existing datasets, then you would use the SQL syntax `ON ALL DATASETS` (see the example scenario outlined below). This gives the user access to all existing datasets. The user would not, however, automatically receive access to any future datasets created by other users.

You should use this method of privilege assignment if you want to restrict a user's access from parent objects, but still wish for them to have access to all existing datasets.

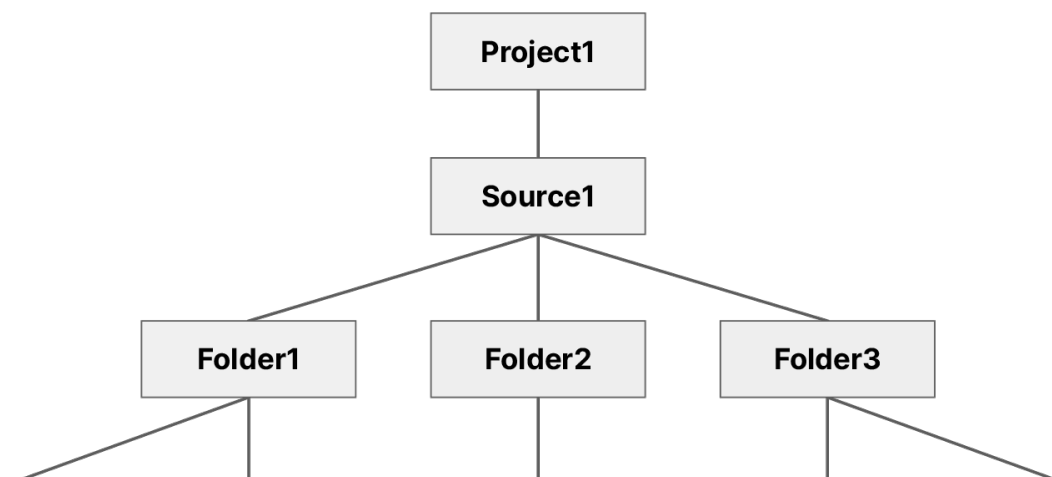
Example: ALL DATASETS

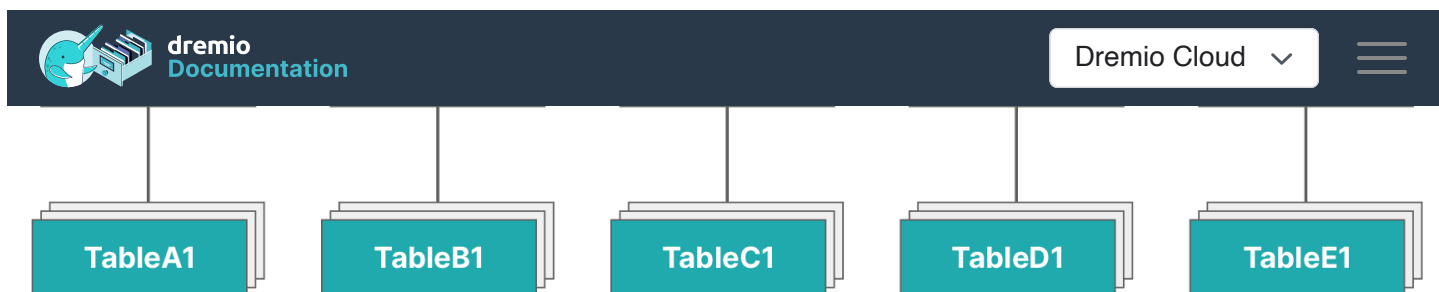
You have a specific user that needs access to all datasets in a specific folder, but they do not require privileges for the folders containing these tables. You would then execute the following command from the SQL Editor:

Example command for all datasets

```
GRANT SELECT ON ALL DATASETS IN PROJECT project1 TO USER user1
```

The image below illustrates the objects `user1` now has access to.





This command restricts the scope of `user1` to all datasets presently found in `source1`, such as `TableC1` and `TableD1`. Should additional datasets be created in the future, `user1` will not have access to them.

3. Granting to a Scope

When you want to grant a user access to a parent object, such as a folder, this will also grant the user access to any datasets contained (see the example scenario outlined below).

You should use this method of privilege management if you wanted to grant a user access to all existing and future datasets contained under a parent object.

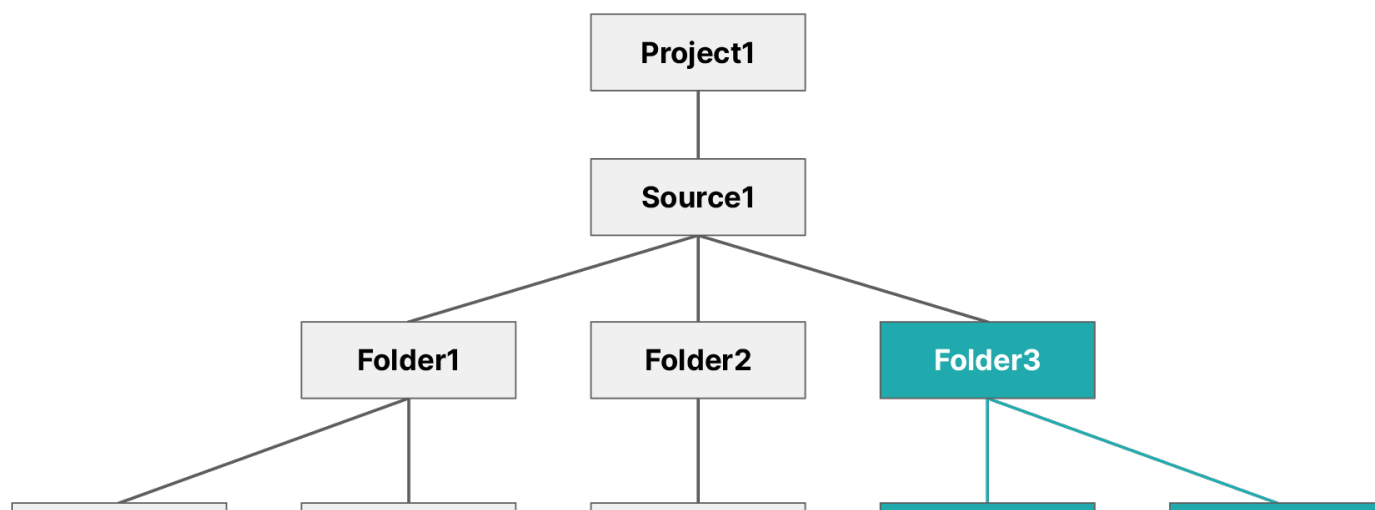
Example: Scope

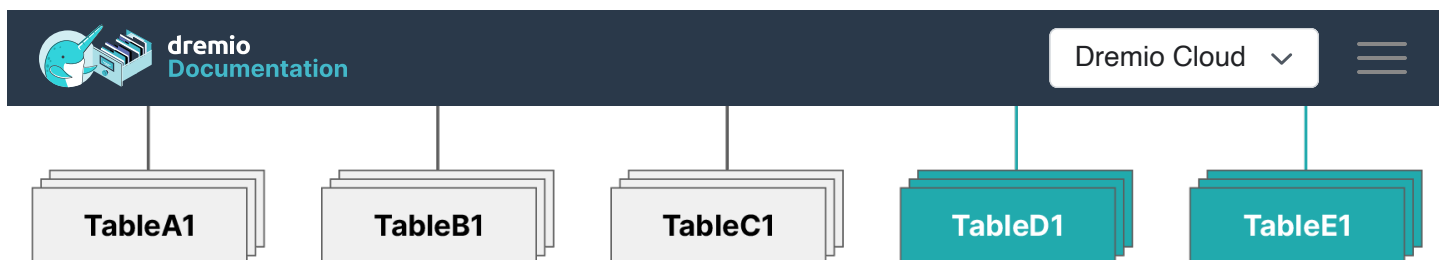
This method grants a user access to all existing and future datasets contained under a specified object. To accomplish this, you need to navigate to the *Privileges* screen from that folder's settings and grant the user the `SELECT` privilege, or execute the following command from the SQL Editor:

Example command for grant to a scope

```
GRANT SELECT ON FOLDER Folder3 TO USER user1
```

The image below illustrates the objects `user1` now has access to.





This grants [user1](#) the [SELECT](#) privilege on [Folder3](#), which means they now have access to all existing and future datasets contained in that folder and its subfolders.

Granting Privileges

By default, all users have all privileges granted to them for any objects without applicable permissions. Once a specific user has been granted access to an object, access is then restricted to only users granted access. All other users no longer have access.

The manual granting of privileges is accomplished using the SQL Editor, REST APIs, or the Privileges screen. The SQL Editor is accessible from any dataset and any [GRANT TO USER](#) command or [GRANT TO ROLE](#) command entered here will apply to the scope supplied with the command itself.

To assign privileges to a user for an object using the Dremio interface:

1. Navigate to the desired object (folder, dataset, source, etc.).
2. Under the *Action* column on the right-hand side of the screen, click the gear button for spaces and data sources or the ellipses (...) button for folders and datasets.
3. Click **Edit Details** or **Settings**, depending on the object type.
4. Select the **Privileges** tab. This is where all privileges may be manually assigned to users for an object.
5. Enter the user's name or email address under **Add User/Role**.
6. Click **Add to Privileges**. If the entry matches a user in Dremio, then a record will appear for them in the privileges table.
7. Select the desired privileges for that user.
8. When finished, click **Save** to preserve your changes.

Revoking Privileges

By default, all users have all privileges granted to them for any objects without applicable permissions. Once a specific user has been granted access to an object, access is then restricted to only users granted access. All other users no longer have access.

The manual revoking of privileges is accomplished using the SQL Editor, REST APIs, or the Privileges screen. The SQL Editor is accessible from any dataset and any [REVOKE SQL](#)



to revoke privileges from a user for an object using the Dremio interface.

1. Navigate to the desired object (folder, dataset, source, etc.).
2. Under the *Action* column on the right-hand side of the screen, click the gear button for spaces and data sources or the ellipses (...) button for folders and datasets.
3. Click **Edit Details** or **Settings**, depending on the object type.
4. Select the **Privileges** tab. This is where all currently-assigned privileges display.
5. Scroll down to the desired user record. If the user you're searching for does not display here, then they do not have privileges for the object, unless all privileges are granted to all users via the **PUBLIC** role.
6. Deselect the checkbox under the privilege columns you wish to revoke access.
7. When finished, click **Save** to preserve your changes.

Note:

If a user has been granted a specific privilege for an object by more than one group and that privilege is revoked for one group, the user will retain that privilege until it is revoked by all groups associated with the same objects.

Was this page helpful?

