Tutorials

Cloud Manager

NetApp
March 18, 2022

Table of Contents

Tutorials

Copying ACLs from SMB shares

Cloud Sync can copy access control lists (ACLs) between a source SMB share and a target SMB share, or from a source SMB share to object storage (except for ONTAP S3). If needed, you also have the option to manually preserve ACLs between SMB shares by using robocopy.

Cloud Sync doesn't support copying ACLs back from object storage to SMB shares.

Choices

- Set up Cloud Sync to automatically copy ACLs
- Manually copy the ACLs between SMB shares

Setting up Cloud Sync to copy ACLs from an SMB server

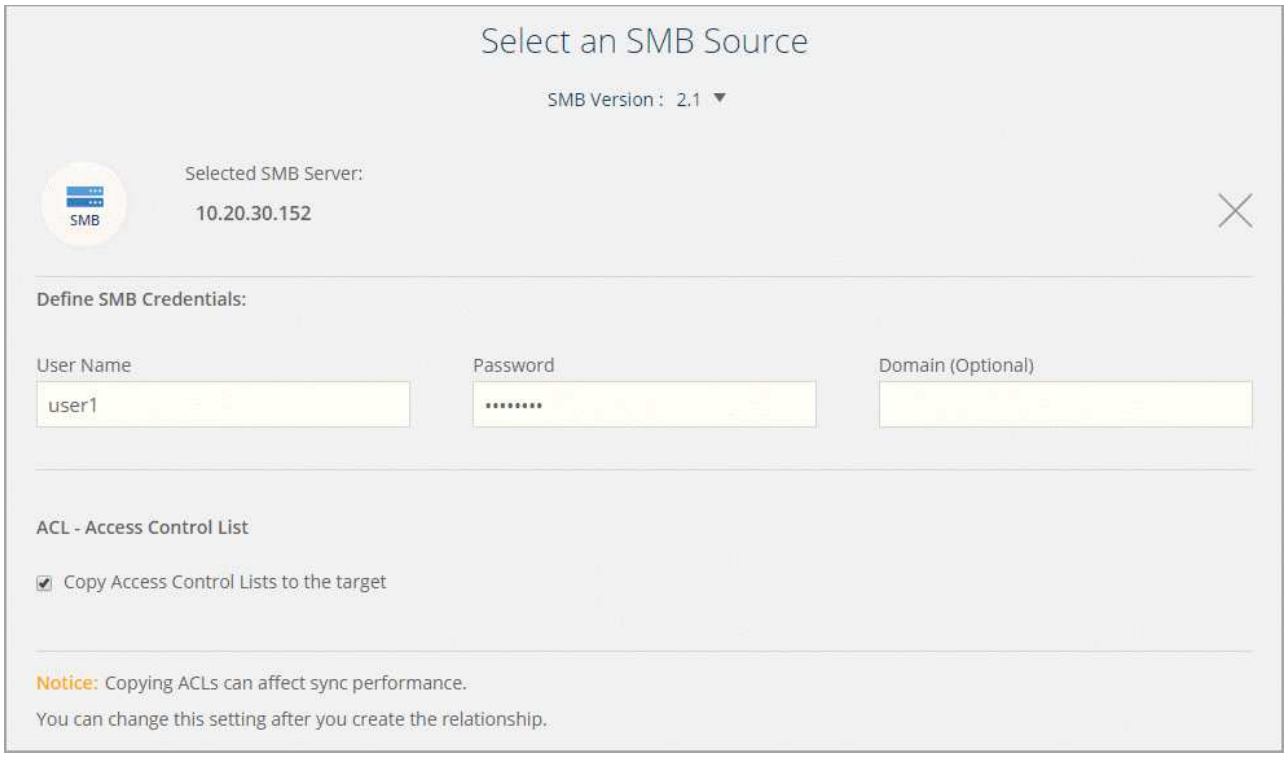
Copy ACLs from an SMB server by enabling a setting when you create a relationship or after you create a relationship.

What you'll need

This feature works with *any* type of data broker: the AWS, Azure, Google Cloud Platform, or on-prem data broker. The on-prem data broker can run any supported operating system.

Steps for a new relationship

1. From Cloud Sync, click **Create New Sync**.
2. Drag and drop **SMB Server** to the source, choose an SMB server or object storage as the target, and click **Continue**.
3. On the **SMB Server** page:
    a. Enter a new SMB server or select an existing server and click **Continue**.
    b. Enter credentials for the SMB server.
    c. Select **Copy Access Control Lists to the target** and click **Continue**.



4. Follow the remaining prompts to create the sync relationship.

   When you copy ACLs from SMB to object storage, you can choose to copy the ACLs to the object's tags or on the object's metadata, depending on the target. For Azure and Google Cloud Storage, only the metadata option is available.

   The following screenshot shows an example of the step where you can make this choice.

Steps for an existing relationship

1. Hover over the sync relationship and click the action menu.
2. Click **Settings**.
3. Select **Copy Access Control Lists to the target**.
4. Click **Save Settings**.

Result

When syncing data, Cloud Sync preserves the ACLs between the source and target SMB shares, or from a source SMB share to object storage.

Manually copying ACLs between SMB shares

You can manually preserve ACLs between SMB shares by using the Windows robocopy command.

Steps

1. Identify a Windows host that has full access to both SMB shares.
2. If either of the endpoints require authentication, use the **net use** command to connect to the endpoints from the Windows host.

   You must perform this step before you use robocopy.

3. From Cloud Sync, create a new relationship between the source and target SMB shares or sync an existing relationship.
4. After the data sync is complete, run the following command from the Windows host to sync the ACLs and ownership:

   ```
   robocopy /E /COPY:SOU /secfix [source] [target] /w:0 /r:0 /XD ~snapshots
   /UNILOG:”[logfilepath]
   ```

   Both *source* and *target* should be specified using the UNC format. For example: \\<server>\<share>\<path>

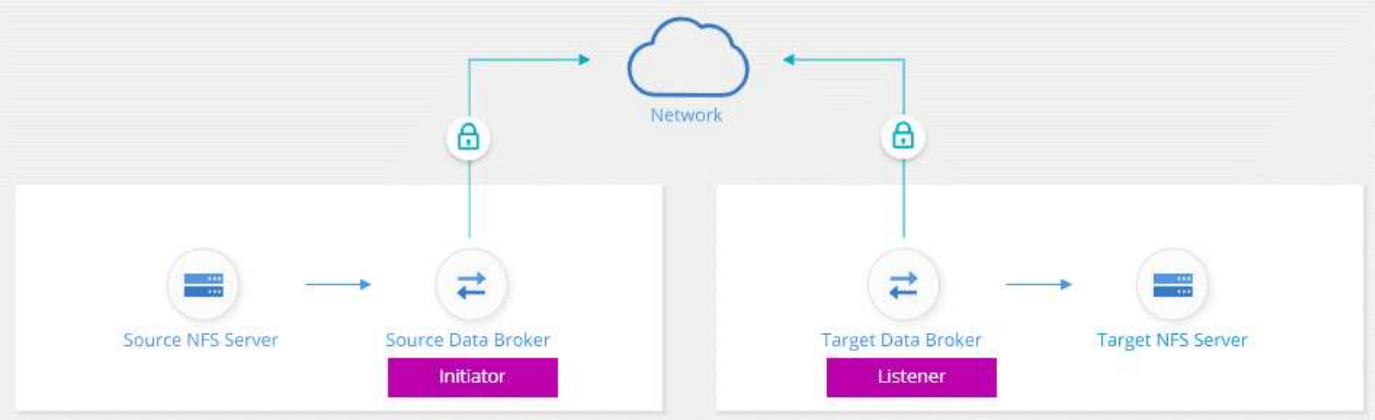
Syncing NFS data using data-in-flight encryption

If your business has strict security policies, you can sync NFS data using data-in-flight encryption. This feature is supported from an NFS server to another NFS server and from Azure NetApp Files to Azure NetApp Files.

For example, you might want to sync data between two NFS servers that are in different networks. Or you might need to securely transfer data on Azure NetApp Files across subnets or regions.

How data-in-flight encryption works

Data-in-flight encryption encrypts NFS data when it’s sent over the network between two data brokers. The following image shows a relationship between two NFS servers and two data brokers:



One data broker functions as the *initiator*. When it's time to sync data, it sends a connection request to the other data broker, which is the *listener*. That data broker listens for requests on port 443. You can use a different port, if needed, but be sure to check that the port is not in use by another service.

For example, if you sync data from an on-premises NFS server to a cloud-based NFS server, you can choose which data broker listens for the connection requests and which sends them.

Here's how in-flight encryption works:

1. After you create the sync relationship, the initiator starts an encrypted connection with the other data broker.
2. The source data broker encrypts data from the source using TLS 1.3.
3. It then sends the data over the network to the target data broker.
4. The target data broker decrypts the data before sending it to the target.
5. After the initial copy, the service syncs any changed data every 24 hours. If there is data to sync, the process starts with the initiator opening an encrypted connection with the other data broker.

   If you prefer to sync data more frequently, you can change the schedule after you create the relationship.

Supported NFS versions

- For NFS servers, data-in-flight encryption is supported with NFS versions 3, 4.0, 4.1, and 4.2.
- For Azure NetApp Files, data-in-flight encryption is supported with NFS versions 3 and 4.1.

Proxy server limitation

If you create an encrypted sync relationship, the encrypted data is sent over HTTPS and isn't routable through a proxy server.

What you'll need to get started

Be sure to have the following:

- Two NFS servers that meet source and target requirements or Azure NetApp Files in two subnets or regions.
- The IP addresses or fully qualified domain names of the servers.
- Network locations for two data brokers.

You can select an existing data broker but it must function as the initiator. The listener data broker must be a *new* data broker.

If you want to use an existing data broker group, the group must have only one data broker. Multiple data brokers in a group aren't supported with encrypted sync relationships.

If you have not yet deployed a data broker, review the data broker requirements. Because you have strict security policies, be sure to review the networking requirements, which includes outbound traffic from port 443 and the internet endpoints that the data broker contacts.

- Review AWS installation
- Review Azure installation
- Review GCP installation
- Review Linux host installation

Syncing NFS data using data-in-flight encryption

Create a new sync relationship between two NFS servers or between Azure NetApp Files, enable the in-flight encryption option, and follow the prompts.

Steps

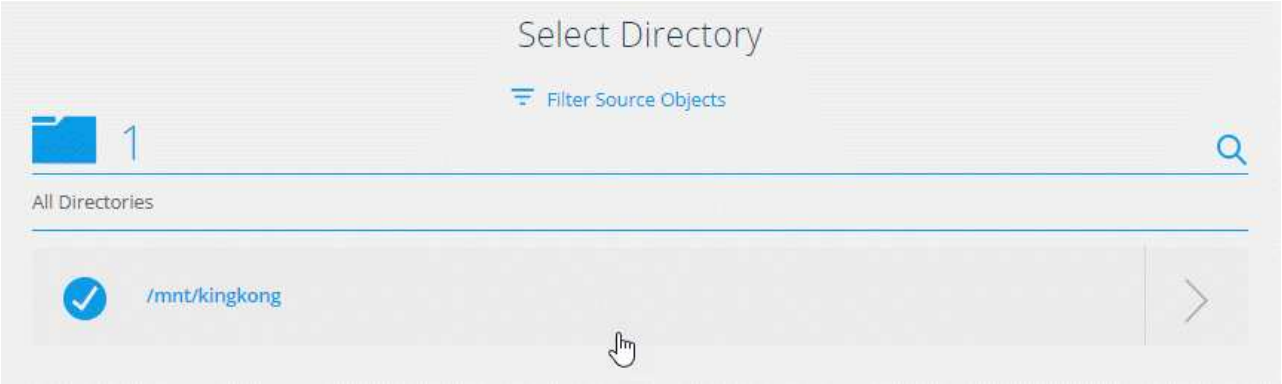
1. Click **Create New Sync**.
2. Drag and drop **NFS Server** to the source and target locations or **Azure NetApp Files** to the source and target locations and select **Yes** to enable data-in-flight encryption.
3. Follow the prompts to create the relationship:
   a. **NFS Server**/**Azure NetApp Files**: Choose the NFS version and then specify a new NFS source or select an existing server.
   b. **Define Data Broker Functionality**: Define which data broker *listens* for connection requests on a port and which one *initiates* the connection. Make your choice based on your networking requirements.
   c. **Data Broker**: Follow the prompts to add a new source data broker or select an existing data broker.

      Note the following:

      - If you want to use an existing data broker group, the group must have only one data broker. Multiple data brokers in a group aren't supported with encrypted sync relationships.
      - If the source data broker acts as the listener, then it must be a new data broker.
      - If you need a new data broker, Cloud Sync prompts you with the installation instructions. You can deploy the data broker in the cloud or download an installation script for your own Linux host.

   d. **Directories**: Choose the directories that you want to sync by selecting all directories, or by drilling down and selecting a subdirectory.

      Click **Filter Source Objects** to modify settings that define how source files and folders are synced and maintained in the target location.

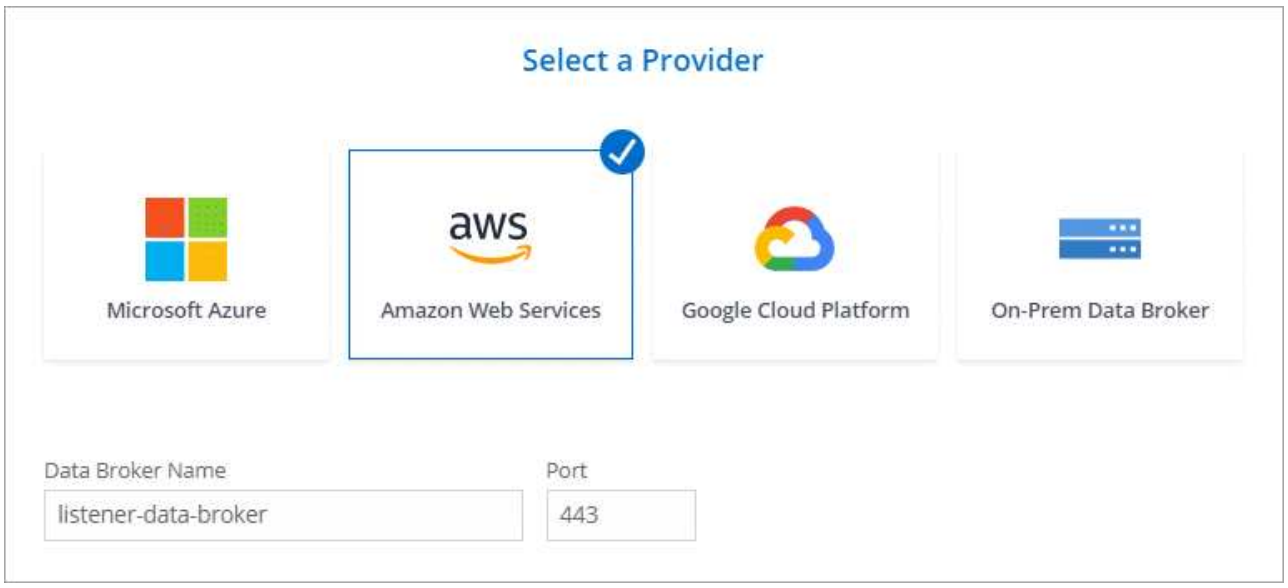


e. **Target NFS Server/Target Azure NetApp Files**: Choose the NFS version and then enter a new NFS target or select an existing server.

f. **Target Data Broker**: Follow the prompts to add a new source data broker or select an existing data broker.

   If the target data broker acts as the listener, then it must be a new data broker.

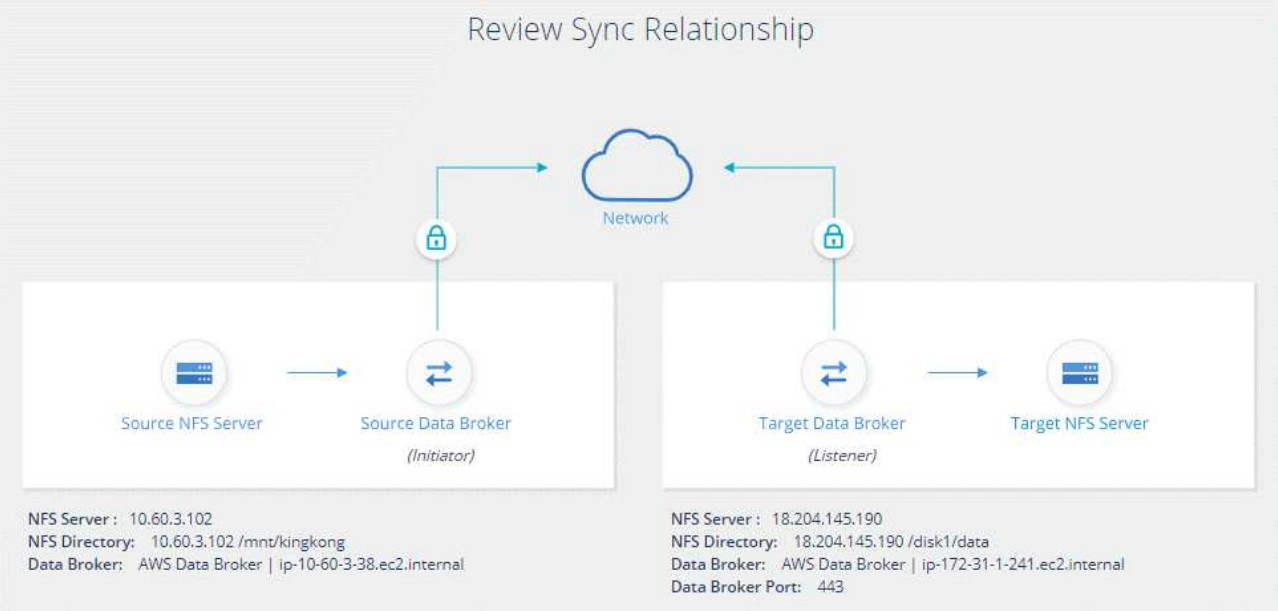
   Here's an example of the prompt when the target data broker functions as the listener. Notice the option to specify the port.



g. **Target Directories**: Select a top-level directory, or drill down to select an existing subdirectory or to create a new folder inside an export.

h. **Settings**: Define how source files and folders are synced and maintained in the target location.

i. **Review**: Review the details of the sync relationship and then click **Create Relationship**.



Result

Cloud Sync starts creating the new sync relationship. When it's done, click **View in Dashboard** to view details about the new relationship.

Setting up a data broker group to use an external HashiCorp Vault

When you create a sync relationship that requires Amazon S3, Azure, or Google Cloud credentials, you need to specify those credentials through the Cloud Sync user interface or API. An alternative is to set up the data broker group to access the credentials (or *secrets*) directly from an external HashiCorp Vault.

This feature is supported through the Cloud Sync API with sync relationships that require Amazon S3, Azure, or Google Cloud credentials.

1 Prepare the vault

Prepare the vault to supply credentials to the data broker group by setting up the URLs. The URLs to the secrets in the vault must end with *Creds*.

2 Prepare the data broker group

Prepare the data broker group to fetch credentials from the external vault by modifying the local config file for each data broker in the group.

3 Create a sync relationship using the API

Now that everything is set up, you can send an API call to create a sync relationship that uses your vault to get the secrets.

Preparing the vault

You'll need to provide Cloud Sync with the URL to the secrets in your vault. Prepare the vault by setting up those URLs. You need to set up URLs to the credentials for each source and target in the sync relationships that you plan to create.

The URL must be set up as follows:

```
/<path>/<requestid>/<endpoint-protocol>Creds
```

Path

The prefix path to the secret. This can be any value that's unique to you.

Request ID

A request ID that you need to generate. You'll need to provide the ID in one of the headers in the API POST request when you create the sync relationship.

Endpoint protocol

One of the following protocols, as defined in the post relationship v2 documentation: S3, AZURE, or GCP (each must be in uppercase).

Creds

The URL must end with *Creds*.

Examples

The following examples show URLs to secrets.

Example for the full URL and path for source credentials

http://example.vault.com:8200/my-path/all-secrets/hb312vdasr2/S3Creds

As you can see in the example, the prefix path is */my-path/all-secrets/*, the request ID is *hb312vdasr2* and the source endpoint is S3.

Example for the full URL and path for target credentials

http://example.vault.com:8200/my-path/all-secrets/n32hcbnejk2/AZURECreds

The prefix path is */my-path/all-secrets/*, the request ID is *n32hcbnejk2*, and the target endpoint is Azure.

Preparing the data broker group

Prepare the data broker group to fetch credentials from the external vault by modifying the local config file for each data broker in the group.

Steps

1. SSH to a data broker in the group.
2. Edit the local.json file that resides in /opt/netapp/databroker/config.
3. Set enable to **true** and set the config parameter fields under *external-integrations.hashicorp* as follows:

   enabled

   - Valid values: true/false

- Type: Boolean
- Default value: false
- True: The data broker gets secrets from your own external HashiCorp Vault
- False: The data broker stores credentials in its local vault

url

- Type: string
- Value: The URL to your external vault

path

- Type: string
- Value: Prefix path to the secret with your credentials

Reject-unauthorized

- Determines if you want the data broker to reject unauthorized external vault
- Type: Boolean
- Default: false

auth-method

- The authentication method that the data broker should use to access credentials from the external vault
- Type: string
- Valid values: “aws-iam” / “role-app” / "gcp-iam"

role-name

- Type: string
- Your role name (in case you use aws-iam or gcp-iam)

Secretid & rootid

- Type: string (in case you use app-role)

Namespace

- Type: string
- Your namespace (X-Vault-Namespace header if needed)

4. Repeat these steps for any other data brokers in the group.

Example for aws-role authentication

```
{
          “external-integrations”: {
                  “hashicorp”: {
                          “enabled”: true,
                          “url”: “https://example.vault.com:8200”,
                          “path”: ““my-path/all-secrets”,
                          “reject-unauthorized”: false,
                          “auth-method”: “aws-role”,
                          “aws-role”: {
                                “role-name”: “my-role”
                          }
                  }
        }
}
```

Example for gcp-iam authentication

```
{
"external-integrations": {
    "hashicorp": {
      "enabled": true,
      "url": http://ip-10-20-30-55.ec2.internal:8200,
      "path": "v1/secret",
      "namespace": "",
      "reject-unauthorized": true,
      "auth-method": "gcp-iam",
      "aws-iam": {
        "role-name": ""
      },
      "app-role": {
        "root_id": "",
        "secret_id": ""
      },
"gcp-iam": {
          "role-name": "my-iam-role"
      }
    }
  }
}
```

Setting up permissions when using gcp-iam authentication

If you’re using the *gcp-iam* authentication method, then the data broker must have the following GCP permission:

```
- iam.serviceAccounts.signJwt
```

Learn more about GCP permission requirements for the data broker.

Creating a new sync relationship using secrets from the vault

Now that everything is set up, you can send an API call to create a sync relationship that uses your vault to get the secrets.

Post the relationship using the Cloud Sync REST API.

```
Headers:
Authorization: Bearer <user-token>
Content-Type: application/json
x-account-id: <accountid>
x-netapp-external-request-id-src: request ID as part of path for source
credentials
x-netapp-external-request-id-trg: request ID as part of path for target
credentials
Body: post relationship v2 body
```

- To obtain a user token and your Cloud Central account ID, refer to this page in the documentation.
- To build a body for your post relationship, refer to the relationships-v2 API call.

Example

Example for the POST request:

```
url: https://api.cloudsync.netapp.com/api/relationships-v2
headers:
"x-account-id": "CS-SasdW"
"x-netapp-external-request-id-src": "hb312vdasr2"
"Content-Type": "application/json"
"Authorization": "Bearer eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6Ik…"
Body:
{
"dataBrokerId": "5e6e111d578dtyuu1555sa60",
"source": {
        "protocol": "s3",
        "s3": {
                "provider": "sgws",
                "host": "1.1.1.1",
                "port": "443",
                "bucket": "my-source"
     },
"target": {
        "protocol": "s3",
        "s3": {
                "bucket": "my-target-bucket"
        }
    }
}
```