Qumulo Administrator Guide

September 09, 2022

Copyright © 2022 Qumulo, Inc.

Table of Contents

Working with File System Protocols

Enabling and Using NFSv4.1 on a Qumulo Cluster 2

Managing File Access Permissions by Using NFSv4.1 Access Control Lists (ACLs) 9

Using NFSv4.1 with Kerberos

How NFSv4.1 Works with Kerberos in Qumulo Core 18

Prerequisites for Joining a Qumulo Cluster to Active Directory 19

Configuring Active Directory for Use With Kerberos 21

Performing Additional Cluster Configuration after Joining Active Directory 26

Using Kerberos Permissions in the Qumulo Filesystem 29

Configuring a Linux Client for NFSv4.1 with Kerberos 36

Configuring Cross-Domain Active Directory Trusts 46

Troubleshooting NFSv4.1 with Kerberos 48

Working with Qumulo Core

Creating a Qumulo Core USB Drive Installer 51

Increasing the Node-Fault-Tolerance Level for Your Cluster During Node-Add Operations 53

Using Qumulo Shift-To to Copy Objects to Amazon S3 55

Using Qumulo Shift-From for Amazon S3 to Copy Objects 68

Managing Snapshots 81

Setting the Web UI Login Banner 83

Setting the Web UI Inactivity Timeout 84

Supported Configurations and Known Limits for Qumulo Core 85

Enabling and Using NFSv4.1 on a Qumulo Cluster

Qumulo Core 4.3.0 (and higher) supports Network File System version 4.1 (NFSv4.1). This section explains how you can configure your cluster for a supported export configuration and enable or disable NFSv4.1 on your cluster. It also provides detail about supported and unsupported features. For more information about NFSv4.1 and file access permissions, see Managing File Access Permissions by Using NFSv4.1 Access Control Lists (ACLs) (page 9).

⚠ Important

- Currently, Qumulo Core supports only NFSv4.1. Mounting with version 4.0 or 4.2 isn't supported.
- The NFSv4.1 protocol requires clients to provide the server with globally unique identifiers. By default, the NFSv4.1 client for Linux uses the machine's hostname as `co_ownerid`. Because the NFSv4.1 protocol requires a unique identifier for every client, an unpredictable failure can occur if two clients have the same hostname. To configure unique identification for your NFS clients, set the `nfs4_unique_id` value for them. For more information, see The nfs4_unique_id parameter in the *Linux kernel user's and administrator's guide*.

Configuring and Using Exports for NFSv4.1

Qumulo's NFS exports can present a view of your cluster over NFS that might differ from the contents of the underlying file system. You can mark NFS exports as read-only, restricted (to allow access only from certain IP adresses), or configure specific user mappings. For more information, see Create an NFS Export on Qumulo Care.

While NFSv3 and NFSv4.1 share each cluster's NFS export configuration, exports behave differently when you access them using NFSv4.1. This section explains these differences and the new requirements for export configurations with NFSv4.1.

Differences Between NFSv3 and NFSv4.1 Exports

In the following example, a Qumulo cluster has the following export configuration.

Export Name	File System Path	Read-Only
`/home`	`/home`	No
`/files`	`/home/admin/files`	No

Export Name	File System Path	Read-Only
`/read_only/home`	`/home`	Yes
`/read_only/files`	`/home/admin/files`	Yes

NFSv3 lets you mount one of these exports by specifying the full export name, for example:

```
mount -o nfsvers=3 cluster.qumulo.com:/read_only/home /mnt/cluster/home
```

This command gives read-only access to the `/home` directory on the cluster by using the path `/mnt/cluster/home`. However, the following command fails with the `No such file or directory` message.

```
mount -o nfsvers=3 cluster.qumulo.com:/read_only /mnt/cluster/read_only
```

NFSv4.1 still lets you mount exports by specifying the full export name. However, NFSv4.1 also supports navigating *above* exports, as if they are part of the file system. The following command succeeds.

```
mount -o nfsvers=4.1 cluster.qumulo.com:/read_only /mnt/cluster/read_only
```

At the mount, the exports under `/read_only` are visible: `/mnt/cluster/read_only` displays virtual directories named `files/` and `home/` with the contents of the corresponding directories in the file system, for example:

```
/mnt/cluster/read_only/
|--- files/<file system contents>
|--- home/
|------ admin/files/<file system contents>
|------ <other file system contents>
```

This presentation of exports lets you view existing exports by using the file system's own interface. It also lets you view new exports as soon as someone creates or modifies them without remounting.

Preparing Export Configurations for NFSv4.1

Qumulo's implementation of NFSv4.1 distinguishes between navigating *above* exports and *inside* an export. To avoid confusion between paths that refer to a virtual directory above an export or a real file system directory inside an export, no export name can be a prefix of another export name when NFSv4.1 is enabled.

In the following example, a Qumulo cluster has the following export configuration.

Export Name	File System Path
`/`	`/`
`/admin`	`/home/admin`

Because `/` is a prefix of `/admin`, you can't enable NFSv4.1 with this export configuration. This restriction prevents the situation where the path `/admin` can refer to both the export of `/home/admin` or the actual file system path `/admin`.

To prepare this configuration for NFSv4.1, you can do one of the following:

- Delete the `/` export and use NFSv4.1 presentation of exports when mounting `/`.
- Delete the `/admin` export.
- Give the `/` export a name that doesn't use other exports as a prefix, for example:

Export Name	File System Path
`/root`	`/`
`/admin`	`/home/admin`

Visibility of IP-Address-Restricted Exports

> **Note**
> The names of exports are public to all NFSv4.1 clients, regardless of IP address restrictions. You can't disable this behavior.

NFSv4.1 respects IP address restrictions on exports: Only clients with allowed IP addresses can access the contents of an export. However, clients without access to an export can still view the export as a directory when they traverse *above* exports. The restrictions apply only when a client attempts to access the contents of the export.

32-Bit Sanitization

- In NFSv3, you can configure specific exports to return 32-bit sanitized data for individual fields. NFSv3 converts any data larger than 32 bits in configured fields to 32-bit data and returns the data. For example, it can sanitize file size to 32-bit format. This truncates the field to `max_uint32` whenever the NFSv3 server returns the attribute.
- NFSv4.1 doesn't support 32-bit sanitization and ignores any sanitizations configured for an export.

Enabling NFSv4.1 on a Qumulo Cluster

> **Note**
> Currently, you can enable NFSv4.1 only by using the `qq` CLI.

You can enable NFSv4.1 on your Qumulo cluster by using a single cluster-wide configuration command, for example:

```
qq nfs_modify_settings --enable-v4
```

When you enable NFSv4.1, all NFS exports are accessible using NFSv3 and NFSv4.1.

Specifying the NFS Mount Option

Typically, NFS clients find and use the highest version of the protocol that both the client and server support. For example, the following command mounts using NFSv4.1 (if it is enabled) and using NFSv3 otherwise.

```
mount -t nfs your.qumulo.cluster:/mount_path /path/to/mountpoint
```

Because Qumulo's NFSv4.1 implementation currently doesn't have full feature parity with NFSv3, you must provide the `nfsvers=3` option for any mounts that require features (such as snapshot access) that only NFSv3 supports, for example:

```
mount -t nfs -o nfsvers=3 your.qumulo.cluster:/mount_path /path/to/mountpoint
```

> **Note**

We recommend specifying the `nfsvers=4` or `nfsvers=4.1` option for any mounts that use NFSv4.1.

Checking Whether NFSv4.1 is enabled

To check whether NFSv4.1 is enabled on your cluster, use the following `qq` CLI command:

```
qq nfs_get_settings
```

Disabling NFSv4.1 on a Qumulo Cluster

⚠ Important
Disabling NFSv4.1 makes any NFSv4.1 mounts unusable immediately. We recommend switching any NFSv4.1 mounts to NFSv3 before disabling NFSv4.1.

To disable NFSv4.1 on an entire Qumulo cluster, use the following `qq` CLI command:

```
qq nfs_modify_settings --disable-v4
```

Configuring Floating IPs for Nodes

Currently, each Qumulo node is limited to 1,000 clients connected using NFSv4.1 simultaneously. To account for nodes going down, we recommend balancing the number of client connections across your nodes by configuring a sufficient number of floating IP addresses per node. This prevents a node failover event from overloading the nodes to which the clients might fail over.

For example, if you configure only one IP address per node, on a cluster with 600 clients per node, a single node failure might overload one of the remaining nodes, preventing 200 clients from connecting. If you assign multiple floating IP addresses to each node, the clients' connections are distributed across multiple nodes.

Listing NFSv4.1 Byte-Range Locks

Rather than lock an entire file, byte-range locking lets you lock specific portions of a file or an entire file in use. This feature is available in Qumulo Core 5.1.3 (and higher). It doesn't require client mount configuration.

The NFSv4.1 implementation in Qumulo Core has a non-configurable lease of one minute. During each lease period, clients send a heartbeat to your Qumulo cluster. The cluster uses this heartbeat to detect lost client connections and to revoke the client leases. When the cluster revokes a lease, it releases any byte-range locks and makes them available to other clients.

Important

- NFSv4.1 byte-range locks are interoperable with NLM (NFSv3) byte-range locks. NFSv4.1 clients view and respect locks that NFSv3 clients hold (the opposite is also true).
- NFSv4.1 and NLM locks aren't interoperable with SMB locks.

To list NFSv4.1 byte-range locks in your cluster, use the following `qq` CLI command:

```
qq fs_list_locks --protocol nfs4 --lock-type byte-range
```

Note

- Currently, Qumulo Core doesn't support revoking NFSv4.1 byte-range locks by using the CLI.
- The time to acquire or release a lock scales linearly with the number of locks that the system already holds on a specific file. If a file has a very large number of locks, system performance can degrade.

Supported and Unsupported Features in Qumulo's Implementation of NFSv4.1

Qumulo's implementation of NFSv4.1 currently supports:

- Authentication with Kerberos (page 18)
- General file system access (reading, writing, and navigating files)
- Unstable writes
- Full use of the NFS exports configuration shared with NFSv3
- Navigation in the pseudo-file system above your exports
- NFSv3-style `AUTH_SYS` authentication (also known as `AUTH_UNIX`)
- Fine-grained control over file permissions using access control lists (ACLs)

- File locking (for example, by using the `fcntl` command)

Qumulo Core doesn't support the following NFSv3 features through NFSv4.1:

- Quota sizes don't appear through NFSv4.1 with certain commands, such as `df` (however, Qumulo Core respects directory quotas)
- You can't access snapshots through NFSv4.1

Qumulo Core doesn't currently support the following NFSv4.1 features:

- Delegations

Managing File Access Permissions by Using NFSv4.1 Access Control Lists (ACLs)

This section explains how you can use Qumulo Core's implementation of NFSv4.1 with access control lists (ACLs) to manage access permissions for files. The Qumulo Core implementation supports using `AUTH_SYS` credentials (also known as `AUTH_UNIX`), `AUTH_NONE` (which acts as `AUTH_SYS` but maps incoming UIDs and GIDs to `nobody`), and `AUTH_KRB5` or `AUTH_KRB5P` credentials. You can use the CLI tools in the `nfs-acl-tools` Linux package to allow or deny various operations.

For more information about NFSv4.1, see Enabling and Using NFSv4.1 on a Qumulo Cluster (page 2).

Using the NFSv4.1 CLI Commands to Manage ACLs

In most Linux distributions, the `nfs-acl-tools` package contains the NFSv4.1 commands that let you manage ACLs for files.

Showing the ACL of a File

To show the ACL of a file, use the `nfs4_getfacl` command. In the following example, we create the file `my-file` and then show the ACL for it.

```
$ touch /mnt/qumulo/my-file
$ nfs4_getfacl /mnt/qumulo/my-file
A::user1@domain.example.com:rwatTnNcy
A:g:group1@domain.example.com:rwatTnNcy
A::EVERYONE@:rtncy
```

The entries in the ACL have four parts separated by colons (`:`). For more information, see the nfs4_acl in the Linux documentation.

The ACL in this example corresponds to `664` mode: The owner (`user1`) and group (`group1`) of the file are allowed to read and write, while others (`EVERYONE@`) are allowed to only read. To check the current mode, use the `stat` command, for example:

```
$ stat -c %a /mnt/qumulo/my-file
664
```

Editing the ACL of a File

To edit the ACL of a file (by using the text editor specified in the `$EDITOR` environment variable), use the `nfs4_editfacl` (or `nfs4_setfacl -e`) command. For more information, see the nfs4_editfacl and nfs4_setfacl in the Linux documentation.

Setting the ACL of a File

To set the ACL of a file, you can use one of the following commands:

- Add a Single ACE: `nfs4_setfacl -a <ace>`
- Set an Entire ACL: `nfs4_setfacl -s <acl>`

Configuring Access Control Entries (ACEs) and Trustee Representation

> **Note**
> The following guidance applies to all `nfs4_acl` scenarios, including getting, editing, and setting the ACL.

There are four fields in the `nfs4_acl` syntax, separated by colons (`:`):

- The ACE type
- Additional ACE flags
- The trustee to which the ACE applies
- The access types to which the ACE applies

ACE Type

In the example of the file ACL (page 9), all three ACEs are set to `A` (allow).

> **Note**
> Qumulo Core supports only `A` and `D` ACEs.

- **A:** Allow
- **D:** Deny
- **U:** Audit
- **L:** Alarm

Additional ACE Flags

In the example of the file ACL (page 9), the second ACE has the flag `g` that shows that the ID in the following part represents a *group* (rather than a user).

> **Note**
> Qumulo Core doesn't support The S and F flags.

The Trustee to Which the ACE Applies

You can use the following trustee representation formats.

> **Important**
>
> - Be careful when you copy *local users and groups* across different Qumulo clusters manually. Aside from UIDs and GIDs, local users and groups are the only identity types in this table that aren't globally unique (because a user or group name represents them). If the destination cluster interprets the named user or group differently, the permissions you set might be unexpected.
> - This consideration doesn't apply to replication copies of local user or group trustees.

Trustee Representation	Example	Description
`<user>@<domain>`	`user1@domain.example.com`	A Kerberos principal that represents a user in the domain to which a Qumulo cluster is joined. You can use this format regardless of client mount security, but only when the cluster is joined to AD. For this trustee in the ACE, the system stores the corresponding AD SID for this user principal on disk. For more information about configuring your clients and Qumulo cluster for Kerberos, see the Using NFSv4.1 with Kerberos (page 18).

Trustee Representation	Example	Description
`<group>@<domain>`	`group1@domain.example.com`	A Kerberos principal that represents a group in the domain to which that a Qumulo cluster is joined. You can use this format regardless of client mount security, but only when the cluster is joined to AD. The group flag isn't necessary to show that this is a group. For this trustee in this ACE, the system stores the corresponding AD SID for this group principal on disk. For more information about configuring your clients and Qumulo cluster for Kerberos, see Using NFSv4.1 with Kerberos (page 18).
`<S-R-X-Y1-Y2-Yn-1-Yn>`	`S-1-5-32-544`	A raw SID. For more information, see Security Identifiers in the Microsoft documentation. To store a SID on disk for this trustee, you can use this format in place of a Kerberos principal. An AD SID must be a user or a group, but can't be both. However, the group flag isn't necessary for showing whether the SID represents a user or group. This can be useful if you have SIDs in a foreign domain (that is, a domain that the cluster isn't joined to). You can use this representation when the cluster isn't joined to a domain at all. When you retrieve an ACL by using `nfs4_getfacl`, the presentation for joined domain SIDs is `<group>@<domain>` and the presentation for foreign SIDs is `<S-R-X-Y1-Y2-Yn-1-Yn>`.
`<numeric_uid>`	`1234`	A numerical UID for an `AUTH_SYS` user. For this trustee in the ACE, the system stores this UID on disk.

Trustee Representation	Example	Description
`<numeric_gid>`	`5678`	A numerical GID for an `AUTH_SYS` user. To avoid having the group interpreted as a user, you must specify the group flag (page 10). For this trustee in the ACE, the system stores the GID on disk.
`qumulo_local/<username>`	`qumulo_local/localuser1`	A user local to a Qumulo cluster (that is, a user that created by using Qumulo Web UI or the `qq` CLI. For the trustee in this ACE, the system stores this user as a local user.
`qumulo_local/<groupname>`	`qumulo_local/localgroup1`	A group local to a Qumulo cluster (that is, a group created by using the Qumulo Web UI or the `qq` CLI. Because local Qumulo users and groups can't share a name, the group flag isn't necessary to show this is a group. For the trustee in this ACE, the system stores this group as a local group, on disk.
`EVERYONE@`	—	Any user of the file system.
`GROUP@`	—	The group owner of a file.
`OWNER@`	—	The owner of a file.

You you can use all trusteee representations interchangeably, even within a single ACL. For example, the following ACL is possible for a file:

```
$ nfs4_getfacl /mnt/qumulo/my-file
A::user1@domain.example.com:rwatTnNcy
A:g:group1@domain.example.com:rwatTnNcy
A::1234:rwatTnNcy
A:g:5678:rwatTnNcy
A::S-1-5-8-9:rwatTnNcy
A:g:S-1-5-32-544:rwatTnNcy
A::qumulo_local/localuser1:rwatTnNcy
A:g:qumulo_local/localgroup1:rwatTnNcy
A::EVERYONE@:rtncy</code>
```

The Access Types to Which the ACE Applies

For example:

- `r`: Read
- `t`: Read attributes
- `w`: Write

The `nfs4_setfacl` command also lets you use the following shorthand:

- `R`: Generic read
- `W`: Generic write
- `X`: Execute permissions

Managing NFSv4.1 Permissions with ACLs and POSIX-Style Modes

You can manage NFSv4.1 access permissions by using ACLs, POSIX-style modes, or a combination of both.

- If you set an ACL on a file and then also set a mode on it, the restrictions that the mode expresses also apply to the ACL. These restrictions change or remove ACEs that apply to the owner, group, or other users.
- If you use the `OWNER@` or `GROUP@` identifiers in an ACL that allows read, write, or execute permissions, the identifiers appear in the `owner` or `group` bits of the mode when you read the file's mode.

> **Note**
> Because the `EVERYONE@` identifier includes the owner and group of a file and the `other` bits of a mode don't apply to the owner or group, the permissions you grant to the `EVERYONE@`

identifier are more broad than a mode's `other` bits.

Using NFSv4.1 ACLs with SMB Access Control

NFSv4.1 ACLs are interoperable with SMB access controls. You can write and read by using both protocols. When you edit over NFS, the system represents SMB SIDs Kerberos principals.

Changing File Owners

When you change the owner of a file, the ACEs that refer to the owner change to the new owner, for example:

```
$ nfs4_getfacl /mnt/qumulo/my-file
A::user1@domain.example.com:rwatTnNcy
A:g:group1@domain.example.com:rwatTnNcy
A::EVERYONE@:rtncy

$ sudo chown user2 /mnt/qumulo/my_file

$ nfs4_getfacl /mnt/qumulo/my-file
A::user2@domain.example.com:rwatTnNcy
A:g:group1@domain.example.com:rwatTnNcy
A::EVERYONE@:rtncy
```

Using Equivalent NFSv4.1 and Qumulo ACL Commands

The syntax for the `nfs4_setfacl` command is `<type>:<flags>:<principal>:<permissions>`, for example `A:fd:GROUP@:rwaDdxtTnNcCoy`. You can use equivalent NFS (`nfs4_setfacl`) and Qumulo (`qq fs_modify_acl`) CLI commands to set ACL permissions.

The following tables compare elements of NFS and Qumulo ACL permissions.

NFSv4.1 ACL Type	Qumulo ACL Type
`A`	`Allowed`
`D`	`Denied`

NFSv4.1 ACL Flag	Qumulo ACL Flag
`d`	`Container inherit`

NFSv4.1 ACL Flag	Qumulo ACL Flag
`f`	`Object inherit`

NFSv4.1 Rights	Qumulo Rights
`a`	`Extend file`
`c`	`Read ACL`
`C`	`Write ACL`
`d`	`Delete`
`n`	`Read EA`
`o`	`Take Ownership`
`r`	`Read contents`
`R`	`Read`, `Synchronize`
`t`	`Read attr`
`T`	`Write attr`
`w`	`Write data`
`W`	`Read ACL`, `Read attr`, `Synchronize`, `Write ACL`, `Write file`
`x`	`Execute/Traverse`
`X`	`Execute/Traverse`, `Read ACL`, `Read attr`, `Synchronize`
`y`	`Synchronize`

The following table gives examples of permissions and equivalent NFS and Qumulo CLI commands.

Permissions	NFSv4.1 Command	Qumulo Command
Add Read Permission to File	`nfs4_setfacl -a "A::OWNER@:R" myfile.ext`	`qq fs_modify_acl --path /myfile.ext add_entry -y Allowed -t "File Owner" -r Read`

Permissions	NFSv4.1 Command	Qumulo Command
Add Read and Execute Permissions to File	`nfs4_setfacl -a "A::EVERYONE@:rtRX" myfile.ext`	`qq fs_modify_acl --path /myfile.ext add_entry -y Allowed -t "EVERYONE" -r Execute/Traverse, Read`
Add Read, Write, and Execute Permissions to File	`nfs4_setfacl -a "A::GROUP@:rtwRWX" myfile.ext`	`qq fs_modify_acl --path /myfile.ext add_entry -y Allowed -t "File Group Owner" -r Execute/Traverse, Read, Write ACL, Write file`
Add Full Access to File	`nfs4_setfacl -a "A::GROUP@:rtwRWX" myfile.ext`	`qq fs_modify_acl --path /myfile.ext add_entry -y Allowed -t "File Group Owner" -r Execute/Traverse, Read, Write ACL, Write file`
Remove Write and Execute Permission to File	`nfs4_setfacl -a "D::OWNER@:wx" myfile.ext`	`qq fs_modify_acl --path /myfile.ext add_entry -y Denied -t "File Owner" -r Execute/Traverse, Write data`
Add Full Access to Group File and Directory Inheritances to Directory	`nfs4_setfacl -a "A:fd:GROUP@:rwaDdxtTnNcCoy" mydirectory`	`qq fs_modify_acl --path /mydirectory add_entry -y Allowed -t "File Group Owner" -r All -f 'Container inherit' 'Object inherit'`

How NFSv4.1 Works with Kerberos in Qumulo Core

This section provides an overview of how NFSv4.1 works with Kerberos in Qumulo Core.

Kerberos is a network authentication protocol that works by using a three-way trust between a key distribution center (KDC), a service server (for example, NFSv4.1 on Qumulo Core), and a client system (for example, a Linux system). This section of the Qumulo Administrator Guide explains how you can configure and use the three entities involved in the trust and provides troubleshooting directions. For more information, see Kerberos on Wikipedia and the MIT Kerberos documentation .

Active Directory (AD) simplifies Kerberos requirements by providing a globally unique security identifier for every user and group (SID) and a KDC implementation with a ticket-granting service (TGS) and an authentication service (AS) .

Configuring Kerberos for Qumulo Core

Qumulo Core 5.1.5 (and higher) supports Kerberos for authenticating AD users over NFSv4.1. The following is an overview of the Kerberos configuration process following the configuration of your AD domain.

1. Join your Qumulo cluster to your AD domain.
2. Join Linux systems to your AD domain.
3. Log in to a Linux system and mount the Qumulo cluster by using the `-o sec=krb5` mount option.

Known Kerberos Limitations for Qumulo Core

Qumulo Core supports only the following features:

- NFSv4.1
- The `krb5` mount flavor

  > **Note**
  > Currently, Qumulo Core doesn't support `krb5i` (integrity, or signing).

- Linux clients
- AES-128 and AES-256 encryption algorithms—for more information, see Network security: Configure encryption types allowed for Kerberos in the Microsoft documentation
- Microsoft Windows Active Directory (Windows Server 2008 and higher)

Prerequisites for Joining a Qumulo Cluster to Active Directory

This section describes the prerequisites for joining a Qumulo Cluster to Active Directory for using NFSv4.1 with Kerberos. For more information, see Join Your Qumulo Cluster to Active Directory on Qumulo Care.

Using Active Directory (AD) for POSIX Attributes (RFC2307)

While using AD for POSIX attributes is optional, it helps avoid issues with Linux ID mapping. We recommend enabling RFC 2307 to match your client's functionality.

- Enabling RFC 2307 might simplify `AUTH_SYS`-based Linux clients that access the cluster by using known UIDs and GIDs. In this way, the cluster can map the UIDs and GIDs to the user or group objects on the AD server and enforce the appropriate permissions.
- If you configure `sssd` on Kerberos-mounted Linux clients for mapping by SID, disabling RFC 2307 can help avoid ascribing special meaning to randomly assigned Linux UIDs and GIDs.

Specifying the Base Distinguished Name (Base DN)

Qumulo uses LDAP to query the AD domain for users and groups. For this functionality, a Base DN must cover any identities intended for use with Kerberos. For example, if multiple organizational units (OUs) contain users, you must include them all in the Base DN (separated with semicolons).

Alternatively, a parent container can hold all nested containers of interest. It is possible to set a top-level domain (TLD) as the Base DN (however, this can cause queries to perform poorly in certain scenarios). We recommend using as specific a Base DN as possible. If you don't configure the Base DN correctly, Linux clients might present permissions such as `nobody` or `65534`.

In the following example, there is an OU with the AD domain `my.example.com`. The TLD Base DN for this domain is as follows.

```
DC=my,DC=example,DC=com
```

If a `Users` container holds users and a `Computers` container holds machine accounts, you can set the Base DN as follows.

```
CN=Users,DC=my,DC=example,DC=com;CN=Computers,DC=stuff,DC=example,DC=com
```

> **Note**
> This example is a very common configuration for user and computer objects in AD.

Using the Active Directory Domain Controller as the NTP Server

Kerberos is very sensitive to clock skew. It is important for all systems involved in a Kerberos relationship—the KDC, your Qumulo cluster, and any Linux clients—to have as little clock skew as possible. We recommend using the same NTP server for all three components.

- You can use your AD domain controller as an NTP server. In the Web UI, on the **Active Directory** page, for **Use Active Directory as your primary time server**, click **Yes**.
- To configure any other NTP server in the Web UI, click **Cluster** > **Date & Time**.

Configuring Active Directory for Use With Kerberos

This section describes the Active Directory Domain Controller (DC) configuration changes necessary for enabling NFSv4.1 with Kerberos.

Configuring DNS in Active Directory

Kerberos relies on DNS to identify machines involved in authentication. NFS clients and servers require DNS `A` records for forward-DNS lookups and `PTR` records for reverse-DNS lookups.

You can use a variety of DNS implementations with Kerberos. In some cases, for example, it might be convenient to use the DNS server that the AD DC provides. For this reason, this section discusses DNS configuration in general terms.

Modifying the Default DNS Configuration

By default, the Qumulo domain-join operation creates a machine account on the domain in the organizational unit (OU)—that you specify during the join process—automatically. This machine account represents all nodes in the cluster, not a single machine.

By default, this machine account has a single, automatically created DNS `A` record that refers to the node on which the system performs the domain-join operation. This DNS record exists on the AD DC used for the domain-join operation and the record refers to a single, public IP address for the node.

The default DNS configuration is generally not useful without additional modifications because:

- **It applies to the DNS server for the DC:** If the environment doesn't use this DNS server, you must create the entry on the DNS server manually.
- **It creates only a DNS `A` (forward) record:** You must create the `PTR` record (a reverse record that maps an IP address to a hostname) manually. This can require creating a reverse zone for the subnet and then adding the specific `PTR` record to the zone.
- **We don't recommend assigning a single IP address to an entire cluster:** In such a configuration, any client that mounts the cluster points at the same node.

Configuring DNS for Distributing Workflows Across Nodes

The Qumulo distributed file system works best when you spread the workload evenly across multiple nodes. We recommend configuring DNS round robin in Active Directory .

This approach provides a list of IP addresses which refer to different nodes in the cluster. Successive DNS queries for the single cluster hostname return different IP addresses. From the perspective of Kerberos, all nodes that comprise a Qumulo cluster act as one host and have the same Kerberos key table. In this way, the Kerberos experience is the same regardless of the selected node.

Unless you need direct access to a specific node through a DNS fully qualified domain name (FQDN), it isn't necessary to use individual DNS `A` records for each node in the cluster (for example, `qumulo1.example.com`, `qumulo2.example.com`, `qumulo3.example.com`, and so on). Instead, we recommend creating a DNS `A` record for the cluster and then duplicating this `A` record for each IP address in the cluster (for example, `qumulo.example.com` → `203.0.113.1`, `qumulo.example.com` → `203.0.113.2`, and so on).

To Configure DNS Round Robin

1. Join your Qumulo cluster to AD (page 19).
2. Find the DNS entry for the cluster on the DNS server.

   Unless you renamed the cluster after joining it to AD, this entry is generally the cluster's name. To find the machine account name in the Web UI, click **Cluster** > **Active Directory** and note the name under **Machine Account**.
3. Update the list of IP addresses for this host record. Include the IP addresses for all nodes.

   To find the IP addresses in the Web UI, click **Cluster** > **Network Configuration**.
4. Configure the DNS resolver to point to the DNS server.

   To find the IP addresses, look up the hostname for the DC. For example:

   ```
   nslookup stuff.example.com
   ```
5. Confirm that successive `ping <cluster_name>` requests connect to a different IP address every time.

Configuring the Service Principal Name (SPN) for NFS

The SPN is a string that identifies the Kerberos services that a particular host provides. We recommend configuring the Qumulo cluster to provide the NFS service. When you configure the SPN, clients can enumerate the cluster and the NFS service as part of a service-ticket-granting request.

To Configure the SPN for NFS by Using the Windows Server Attribute Editor

> **Note**
> To maximize compatibility with Linux, we recommend formatting SPN entries in lowercase.

1. Use RDP to log in to the DC for your AD domain.
2. Open **Active Directory Users and Computers**.
3. Find the machine account for your Qumulo cluster.

   To find the machine account name in the Web UI, click **Cluster** > **Active Directory** and note the name under **Machine Account**.
4. Right-click the account and then click **Properties** > **Attribute Editor**.
5. On the **Attribute Editor** tab, find the `servicePrincipalName` attribute and edit its value to include a new SPN in the `nfs/<machine_account>.<domain_fqdn>` format, for example:

   ```
   nfs/qumulo-cluster.ad.eng.example.com
   ```

   > **Tip**
   > You can use the other, automatically generated entries as syntax examples.

To Configure the SPN for NFS by Using the Windows Server Command Prompt

> **Note**
> - To maximize compatibility with Linux, we recommend formatting SPN entries in lowercase.
> - The SPN formatting in the following example is generally sufficient for Linux service ticket requests. However, depending on your environment and client configuration, additional entries might be necessary.

1. Open a command prompt with administrative privileges.
2. Use RDP or SSH to connect to your AD domain.

3. Run the `setspn` command with the machine account (in this example, `qumulo-cluster`) followed by a period (`.`) and the FQDN (in this example, `ad.eng.example.com`). For example:

```
setspn -s nfs/qumulo-cluster.ad.eng.example.com
```

4. Confirm the configuration by using the `setspn` command with the machine account name. For example:

```
setspn qumulo-cluster
```

To Troubleshoot Your SPN Configuration

If your SPN is configured incorrectly, a client is likely to display the following error:

```
mount.nfs: access denied by server while mounting qumulo-cluster.ad.eng.qumulo.com:/
```

1. Take a client-side packet capture and find the logs for the client and AD Kerberos.
2. Search the logs for the `S_PRINCIPAL_UNKNOWN` error.
3. Add the required client parameters to the SPN configuration.

Configuring SPN with DNS

For Kerberos authentication to work correctly, SPN entries must correspond to DNS `A` records exactly. Although the machine account is sometimes the same as the DNS `A` record created during the domain-join process, depending on your the DNS environment, this might not always be true.

In the following example, a Qumulo cluster has a machine account with the SPN `nfs/qumulo.example.com` and two DNS `A` records that point to the same Qumulo cluster IP, `203.0.113.0`:

- `qumulo.example.com`
- `storage.example.com`

Because the `storage.example.com` doesn't have a corresponding SPN, you can perform Kerberos authentication by using the `qumulo.example.com` record. However, if you add the second SPN (`nfs/storage.example.com`) to the machine account account SPN list, the account can authenticate by using either of the two hostnames.

`CNAME` (alias) records are an exception to this arrangement. `CNAME` records that point to a correctly-configured `A` record, and which have a corresponding SPN entry in the machine account, don't require the `CNAME` host to be added to the SPN. For example, the `CNAME` record `storage-alias.example.com` that points to `storage.example.com` requires the SPN list to contain only `nfs/storage.example.com` to authenticate against `storage-alias.example.com`.

Performing Additional Cluster Configuration after Joining Active Directory

This section describes additional Qumulo cluster configuration that can affect the behavior of NFSv4.1 with Kerberos.

When your Qumulo cluster is joined to AD (page 19), you must configure the NFSv4.1 server (page 2) and NFSv4.1 security settings.

To Configure Security Settings by Using the qq CLI

Qumulo provides configuration for the permitted NFSv4.1 authentication flavors in the `qq` CLI or directly through the REST API.

1. Use the `qq` CLI to get the current settings:

   ```
   $ qq nfs_get_settings
   {
     "auth_sys_enabled": true,
     "krb5_enabled": true,
     "krb5p_enabled": true,
     "v4_enabled": false
   }
   ```

   This is the default configuration:

   - NFSv4.1 is disabled by default.
   - `AUTH_SYS`, `AUTH_KRB5`, and `AUTH_KRB5P` are enabled by default (however, the `AUTH_KRB5` configuration has no effect on NFSv3 because Qumulo Core doesn't support Kerberos with NFSv3).

2. To harden security, configure your cluster to use only Kerberos by disabling `AUTH_SYS` (without changing `AUTH_KRB5`). For example:

   > ⚠ Important
   > Because it uses authentication based on a simple UID and GID passed over the wire in plain text, RPC `AUTH_SYS` is inherently insecure. In a trusted environment, `AUTH_SYS` might be sufficient for enforcing basic permissions and preventing good-faith actors from making mistakes. In all other cases, you must treat `AUTH_SYS` as if it provides *no security whatsoever*.

```
$ qq nfs_modify_settings --disable-auth-sys
{
  "v4_enabled": false,
  "auth_sys_enabled": false,
  "auth_krb5_enabled": true
  "auth_krb5p_enabled": true
}
```

3. (Optional) You can also use the following commands.

Command	Description
`qq nfs_modify_settings --enable-auth-sys`	Enables `AUTH_SYS` without changing `AUTH_KRB5`
`qq nfs_modify_settings --enable-krb5`	Enables `AUTH_KRB5` without changing `AUTH_SYS`
`qq nfs_modify_settings --enable-krb5p`	Enables `AUTH_KRB5P` without changing `AUTH_SYS`
`qq nfs_modify_settings --enable-v4`	Enables NFSv4.1
`qq nfs_modify_settings --disable-v4`	Disables NFSv4.1
`qq nfs_modify_settings --disable-krb5`	Disables `AUTH_KRB5` without changing `AUTH_SYS`
`qq nfs_modify_settings --disable-krb5p`	Disables `AUTH_KRB5P` without changing `AUTH_SYS`

Note

- Security configuration options apply to *all* versions of NFS (NFSv3 and NFSv4.1). Thus, disabling `AUTH_SYS` also disables NFSv3, because `AUTH_SYS` is the only authentication flavor that NFSv3 supports by design.
- In a secure environment, where Kerberos is required, `AUTH_SYS` NFSv3 connections aren't allowed.
- These configuration options apply cluster-wide to all NFS exports and files.

Configuring Export Configuration

You can use NFSv4.1 exports (page 2) to configure access to the Qumulo file system.

> **Note**
> The user-mapping portion of the export configuration has no effect on Kerberos configuration. Specifying `root` or `any` user mapping for a particular export applies only to `AUTH_SYS` mounts that access this export.

In all other ways, exports behave the same for `AUTH_KRB5` or `AUTH_KRB5P` as they do for `AUTH_SYS`. IP address restrictions that you specify in an export work as expected.

Using Kerberos Permissions in the Qumulo Filesystem

This section describes how NFSv4.1 interacts with the secure file permissions that Kerberos enables for the Qumulo Core file system. For more information, see Qumulo File Permissions Overview on Qumulo Care.

Listing Permissions for Files

> **Note**
>
> - This section uses the Kerberos term *trustee* and Qumulo term *identity* (or `auth_id`) interchangeably.
> - The term *file* in the Qumulo file system can refer to:
>   - A file
>   - A directory
>   - A symbolic link
>   - A special block device

All files in the Qumulo file system have the following fields associated with them:

- Owner
- Group owner
- Access control list (ACL)—a list of access control entries (ACEs)

These fields, stored in the metadata for a file or directory, determine the access permissions that a trustee or identity has to files.

For any file operation, the system checks the authenticated user against file permissions to determine whether the operation should be allowed. When you create a new file, the authenticated user becomes the owner of the new file.

In the following example, we create a file in a mount over NFS.

> **Note**
>
> - Because this example uses an `AUTH_SYS` mount, it has UID and GID identity values set to `1000`.

- We recommend becoming familiar with the following commands to better understand the various elements for permissions types that the system stores on disk.

```
touch /mnt/mount_point/filename
```

To view the exact permissions metadata for this file, use the `qq fs_file_get_attr` command. For example:

```
$ qq fs_file_get_attr --path /filename
{
  "group_details": {
    "id_type": "NFS_GID",
    "id_value": "1000"
  },
  "owner_details": {
    "id_type": "NFS_UID",
    "id_value": "1000"
  },
  ...
}
```

To view the permissions configured in an ACL, use the `qq fs_get_acl` command. For example:

```
$ qq fs_get_acl --path /filename
Control: Present
Posix Special Permissions: None

Permissions:
Position  Trustee   Type     Flags  Rights
========  ========  =======  =====  ==============================
1         uid:1000  Allowed         Delete child, Read, Write file
2         gid:1000  Allowed         Delete child, Read, Write file
3         Everyone  Allowed         Read
```

Listing Security Identifiers (SIDs)

The SID is a globally unique identifier for a user or group object in a domain. For more information, see Security identifiers in the Microsoft documentation.

Because Qumulo's Kerberos implementation requires AD, every user is also an Active Directory user. The domain controller (DC) has an equivalent mapping for AD users and SIDs. Qumulo uses LDAP to determine the AD-user ↔ SID mapping. For this reason, it is important to configure the Base DN for your cluster correctly.

Qumulo's Kerberos implementation stores SIDs on disk for files that have Kerberos identities in the user, group, or ACL. When a user authenticates by using Kerberos and creates a file, Qumulo Core configures the user, group, and ACL automatically.

To set the identity for an AD user, you can modify the permissions for an existing file by using the `chown` or `nfs4_setfacl` command.

In the following example, the Kerberos-authenticated AD domain user `AD\myusername` creates a file over NFSv4.1 and the system gives an ACL response from the REST API. The response contains an ACE entry for the owner and group owner of the user `AD\myusername`, with corresponding SIDs for both.

```
$ qq fs_get_acl --path /filename --json
{
  "aces": [{
    "trustee": {
      "name": "AD\\myusername",
      "sid": "S-1-5-21-4202559609-EXAMPLE158-3224923410-13507",
      ...
    },
    ...
  }, {
    "trustee": {
      "name": "AD\\Domain Users",
      "sid": "S-1-5-21-4202559609-EXAMPLE158-3224923410-513",
      ...
    },
    ...
  }]
}
```

Using Kerberos Principals

Although Qumulo stores SIDs on disk, SIDs appear rarely when you use NFSv4.1 on Linux systems. Instead, the system represents Kerberos identities as Kerberos principals. A *Kerberos principal*, a string in the `<user@domain>` or `<group@domain>` format, is easier to read.

> **Note**
> There is an equivalent mapping between AD users, SIDs and Kerberos principals. Each of these representations is unique (a primary key to the AD identity database).

Qumulo's implementation of the SID ↔ Kerberos principal mapping uses the `sAMAccountName` field, which is always present and unique for all AD users and groups. The system forms the Kerberos principal by concatenating the name and domain in the `<sAMAccountName>@<domain>` format.

AD has fields with similar content but without the guarantee of uniqueness (such as the `name`, `distinguishedName`, `CN`, and `servicePrincipalName`). However, AD permits setting these fields to unrelated values. For this reason, it is unlikely but possible that certain environments use special values in these fields. Qumulo's Kerberos implementation ignores these fields and uses only the value in the `sAMAccountName` field.

> **Note**
> The fields can diverge significantly if an administrator edits them.

The following example shows how the system represents the SIDs from the previous example as Kerberos principals.

```
$ nfs4_getfacl filename
A::test2@ad.eng.qumulo.com:rwatTnNcy
A:g:Domain Users@ad.eng.qumulo.com:rtncy
A::EVERYONE@:rtncy
```

Although the system stores raw SIDs on disk, the `nfs_getfacl` command displays users and groups as Kerberos principals. This format is valid for setting identities on a file by using commands such as `nfs4_setfacl`, `chown`, and so on.

Understanding Kerberos Principal Caveats

This section explains some of the caveats of working with Kerberos principals.

Machine Account Object Names

When you work with machine accounts, AD stores the `sAMAccountName` as the object name and appends `$` to it. If a client named `myclient` is joined to the domain `stuff.example.com`, the name of the machine account object in **Active Directory Users or Computers** appears as `myclient` while the Kerberos principal representation over NFS appears as `myclient$@stuff.example.com`.

This functionality is different from other account types in AD, where the object name usually matches the `sAMAccountName` exactly.

ID Mapping on Linux systems

Linux systems perform their own ID mapping separately from the Qumulo cluster ID mapping. Linux systems also use `sAMAccountName` as the AD user primary key when joined to an AD domain. However, Linux systems use `CN` when looking up groups. Thus, in groups where the `sAMAccountName` and `CN` don't match (possibly due to edits by an administrator), a Linux system and Qumulo Core might understand differently the group that the Kerberos principal refers to.

Ensure the two fields are in sync to prevent the following possible scenarios:

- An error appears when you configure the group.
- Group configuration succeeds but the configured group is incorrect.

Unicode Characters in Kerberos Principals

For most standard Linux tools, Qumulo Core supports all arbitrary Unicode characters in Kerberos principals. However, we don't recommend using the period (`.`) character in principals, except in the domain name.

Using the chown Tool With Kerberos

`chown` is a Linux tool that changes the owner or group owner for a file. You can generally use `chown` with Kerberos principals. On most Linux systems, `chown` requires the root user (`sudo chown`).

The AUTH_SYS Root User

`AUTH_SYS` has the concept of the root user. Using `sudo` on a Linux NFS client fills in `0` for the UID and GID. As long as the mounted export doesn't *root squash*—maps a client's UID `0` (root) to `65534` (nobody) or to another non-root user—the Linux client receives root permissions on the Qumulo file system, where the client can perform `chown` operations.

The Kerberos Root User

Kerberos doesn't have the concept of the root user. However, you can still use it to run `chown` operations under the following conditions.

- The ACL for the file must grant the `CHANGE_OWNER` privilege to an authenticated user.
- The currently authenticated user must be a member of the destination group (if provided) or a member of the current group (if the group isn't being modified).

If both conditions are true, a `chown` operation on files performed as a Kerberos user over NFSv4.1 succeeds. For example:

```
$ chown user3:group4 filename
```

> **Note**
> Including `@<domain>` for the destination user and group is optional.

Viewing the Owner and Group

The following examples show how to display user and group membership by using the `ls -l` and `stat -c` commands.

```
$ ls -l filename
-rw-r--r--    1 user3    group4         0 Jun  9 23:18 filename
```

```
$ stat -c '%U, %G' filename
user3, group4
```

> **Note**
> The Kerberos restrictions for `chown` also apply to other Linux tools that use the `chown` system call, such as `cp` and `rsync`, when you run them in ownership-preserving modes.

Using the Linux ACL Editor

The Linux ACL Editor consists of the following tools:

- `nfs4_editfacl`
- `nfs4_getfacl`
- `nfs4_setfacl`

You can use the editor to read and write ACLs on a Qumulo cluster that uses NFSv4.1 with Kerberos. For more information, see Managing File Access Permissions by Using NFSv4.1 Access Control Lists (ACLs) (page 9).

Configuring a Linux Client for NFSv4.1 with Kerberos

This section describes how to configure a Linux client for using NFSv4.1 with Kerberos.

> **Note**
> Qumulo Core supports only Linux for using NFSv4.1 with Kerberos.

Linux systems implement Kerberos support as a series of loosely related packages and configuration files. For this reason, configuration depends on the Linux distribution and version. This section refers to tools, packages, dæmons, configuration files, and other elements in Ubuntu 18.04 LTS.

Joining a Linux Client to a Domain

There are two common ways of joining a Linux client to an Active Directory (AD) domain automatically, by using `samba` or `realmd`. Both methods require creating the /etc/krb5.conf configuration file and defining a default domain and the relationships between domains and realms.

Configuring the /etc/krb5.conf File

The following is an example configuration for joining a domain.

```
[libdefaults]
    default_realm = MY-DOMAIN.EXAMPLE.COM

[realms]
    MY-DOMAIN.EXAMPLE.COM = {
        kdc = my-domain.example.com:88
        admin_server = my-domain.example.com:749
    }

[domain_realm]
    my-domain.example.com = MY-DOMAIN.EXAMPLE.COM
    .my-domain.exmaple.com = MY-DOMAIN.EXAMPLE.COM
```

To Join a Linux Client to a Domain by using samba

samba is a suite of Linux tools that provides Windows-like functionality on Linux. The `net ads join` command creates a machine account on the domain.

1. To specify how the domain-join process behaves, edit the `/etc/samba/smb.conf` file. For example:

```
workgroup = my-domain
server role = member server
realm = my-domain.example.com
kerberos method = system keytab
```

2. To join the domain, run the `net ads join` command. For example:

```
$ net ads join my-domain.example.com -U Administrator
```

3. `samba` doesn't create configuration files. Configure the `sssd` and `idmapd` tools manually. For more information, see Mapping External Identities to Linux Identities (page 38).

To Join a Linux Client to a Domain by using realmd

realmd is a tool that allows managing realm-based authentication. It can be somewhat more difficult to use than `samba`. However, it creates a more complete configuration. For example, it configures the `sssd` tool during the domain-join process.

1. To join a domain, use the `realm join` command. For example:

```
$ realm join my-domain.example.com -U Administrator
```

2. Configure the `sssd` and `idmapd` tools manually. For more information, see Mapping External Identities to Linux Identities (page 38).

To Configure DNS and Service Principal Name (SPN)

Kerberos relies on DNS to identify machines involved in authentication. NFS clients and servers require DNS `A` records for forward-DNS lookups and `PTR` records for reverse-DNS lookups.

1. After you configure DNS, check DNS resolution from your client. For example:

```
$ nslookup my-client-machine.my-domain.example.com
```

2. In addition to DNS configuration, Linux clients require a standard host SPN on the machine account created while joining the domain. We recommend configuring the SPN by using the `setspn` command on the domain controller after the join procedure. For example:

   > **Note**
   > Running this command resets the SPN to the default value for your machine.

   ```
   setspn -r my-client_machine
   ```

Mapping External Identities to Linux Identities

During the *ID mapping* process, a Linux system converts external identities to Linux identities.

- For Qumulo Core, *external identities* are equivalent to *Kerberos principals*.
- For Linux, *identities* are simple integers: UIDs and GIDs.

> **Note**
> Because Linux can't use complex external identities in system calls, a Linux system must perform identity conversion before operating on files.

ID mapping is bidirectional. A system call, such as `chown`, that takes a UID or GID as input requires mapping the UID or GID be mapped to a domain user or group *before* passing it to your Qumulo cluster over NFS.

A system call, such as `stat`, that returns a UID or GID, requires that the domain user or group that returned from your Qumulo cluster over NFS be converted to a UID or GID before the system can present it to the user.

Configuring Active Directory Authentication by using sssd

sssd (System Security Services Daemon) is a tool responsible for managing authentication with external providers in Linux. To use NFSv4.1 with Kerberos, you must configure `sssd` with AD as the identity provider.

- If you join domains by using `samba`, you must create the /etc/sssd.conf file.
- If you join domains by using `realmd`, you might already have a `/etc/sssd.conf` file. For detailed configuration information, see sssd-ldap in the Linux documentation.

In the following example, the `sssd.conf` file configures basic ID mapping for AD.

```
[sssd]
domains = my-domain.example.com
config_file_version = 2
services = nss, pam

[domain/my-domain.example.com]
ad_domain = my-domain.example.com
krb5_realm = MY_DOMAIN.EXAMPLE.COM
cache_credentials = True
id_provider = ad
krb5_store_password_if_offline = True
default_shell = /bin/bash
ldap_id_mapping = False
use_fully_qualified_names = False
fallback_homedir = /home/%u@%d
access_provider = ad
```

Configuring LDAP Queries against the Domain Controller (DC) by using sssd

Like Qumulo clusters, Linux systems can resolve details about user and group objects by querying the DC over LDAP. In particular, a Linux system looks for an object with a matching `sAMAccountName` (user) or `CN` (group)

1. To toggle RFC 2307 for mappings in the `sssd.conf` file, configure the ldap_id_mapping field .
    - When you set the field to `False`, the client checks whether the RFC 2307 `uidNumber` or `gidNumber` are set on an object.
        - If the number is set, it becomes the Linux UID or GID for the operation.

          Important
          AD doesn't prevent duplicate UID or GID numbers from being added to RFC 2307 values. For this reason, incorrect configuration can lead or UID or GUID collisions. When a Linux system determines that a collision has occurred, it chooses the first UID or GID it finds.
        - Otherwise, the UID or GID becomes `nobody` or `nogroup` (`65534`).

          Note
          In most cases, an owner or group becomes 65534 as a result of incorrect user mapping configuration in the client. To understand

which LDAP queries run and why they have trouble finding the correct information, check your logs.

- When you set the field to `True`, the client assigns locally a new unique UID or GID to each `objectSID` that it finds on the DC.

  > **Note**
  > This is a more flexible approach than requiring RFC 2307. However, this also means that UIDs and GIDs aren't the same across different Linux systems within the same domain.

  In both cases, the client communicates with the DC by using its machine account.

2. To pick up changes to the `/etc/sssd.conf` file on a live system, restart the `sssd` service.

Configuring the Conversion of Local Identities to NFS Representations by Using idmapd

idmapd (or nfsidmap) , is a tool that lets you convert local identities to their on-the-wire NFS representations. Although `idmapd` works with `sssd`, it has additional configuration options.

In the following example, the `/etc/idmapd.conf` file configures a Linux client joined to AD:

```
[General]
Domain = my-domain.example.com
Verbosity = 0
Pipefs-Directory = /run/rpc_pipefs

[Mapping]
Nobody-User = nobody
Nobody-Group = nogroup
```

> **Note**
> Depending on your Linux distribution and configuration, you might have to add the `Domain` field to the default configuration file.

Authenticating as an AD User and Mounting Your Qumulo Cluster

Qumulo Core supports three methods of authenticating as an AD user and mounting your cluster over NFSv4.1 as the AD user. These methods, from least to most complex, and in an increasing order of utility, are:

- By using a machine account
- By using manual authentication with the `kinit` tool
- By using the `autofs` tool

To Authenticate as an AD User by Using a Machine Account and Mount Your Qumulo Cluster

Machine account authentication uses one AD user for each Linux system. This *machine account user* is the same as the *machine account* created on the domain during the domain-join operation. Any user on the Linux system who has access to the machine account mount point can operate as the machine account user on a Qumulo cluster.

Machine account authentication can be useful for simple scenarios in which trusted users on trusted Linux machines require a secure mechanism for communicating with a Qumulo cluster. Because this is also the easiest authentication method to configure, it can be a good starting point for administrators who configure NFSv4.1 with Kerberos for the first time.

> **Note**
> Both machine account authentication and `kinit` have limited usefulness because they limit the mount point to a single authenticated user. Between the two authentication options, `kinit` has an advantage because of the way it handles ID mapping.

1. Confirm that your /etc/nfs.conf file , contains the following flag.

   ```
   [gssd]
   use-machine-creds=true
   ```

   The `use-machine-creds` flag specifies whether authentication uses machine credentials when `sudo mount` is invoked for NFSv4.1 with Kerberos. When you set the flag to `true`, `gssd` authenticates as the machine account for the system on behalf of the NFS client. (It performs a `kinit` operation as the machine account). The credential cache that results from the `kinit` is usually located in `/tmp`. To search for the cache, use the `ls /tmp/*krb5*` command.

> **Note**
> In versions of Ubuntu lower than 22.04 (and possibly on other Linux distributions), you can't use the `/etc/nfs.conf` file to configure `gssd`. If this is the case for your system, we recommend starting the `rpc.gssd` service by using the `-n` flag.

2. Mount your cluster by using the `krb5` security mechanism. For example:

```
$ sudo mount -o vers=4.1,sec=krb5 my-cluster.my-domain.example.com:/ /mnt/poin
t
```

3. Use the Qumulo file system.

```
$ cd /mnt/point
$ touch filename
$ ls -l filename
-rw-r--r--    1 MY_MACHINE$    domain computers         0 Jun  9 23:18 filenam
e
```

> **Important**
> The machine account is the owner of any new files.

If the machine name isn't visible, make sure that the AD container holds this machine in the Qumulo cluster's Base DN configuration (typically, `CN=Computers,DC=...`). If the machine name is still not visible, configure the Linux client ID mapper to provide local mappings when no RFC 2307 mapping is available. It is uncommon for machine accounts to have RFC 2307 mappings.

To Authenticate as an AD User Manually by Using kinit and Mount Your Qumulo Cluster

kinit authentication is very similar to machine account authentication. The main difference is that you must create the credentials for the mount manually. You can use any user in the AD domain. However (this is also true for machine accounts), any local Linux user that can access the mount point can operate on the Qumulo cluster as this single user.

> **Note**
> Both machine account authentication and `kinit` have limited usefulness because they limit the mount point to a single authenticated user. Between the two authentication options, `kinit` has an advantage because of the way it handles ID mapping.

In environments where Linux systems map exactly to end users that have `kinit`-based Kerberos mounts on their Qumulo clusters, `kinit` might be sufficient.

1. Authenticate by using `kinit`. For example:

   ```
   $ sudo kinit my-user
   ```

2. When prompted for a password, use the AD domain password for the user.
3. To confirm the result of the authentication operation, use the `sudo klist` command.
4. Confirm that the /etc/nfs.conf file contains the following flag:

   ```
   [gssd]
   use-machine-creds=false
   ```

   The `use-machine-creds` flag specifies whether authentication uses machine credentials when `sudo mount` is invoked for NFSv4.1 with Kerberos. When you set the flag to `false`, `gssd` searches for an existing credential cache (which you created by running `kinit`) in `/tmp/krb5cc_0` for authenticating with the Qumulo cluster.

5. Mount your cluster by using the `krb5` security mechanism. For example:

   ```
   $ sudo mount -o vers=4.1,sec=krb5 my-cluster.my-domain.example.com:/ /mnt/poin
   t
   ```

6. Use the Qumulo file system.

   ```
   $ cd /mnt/point
   $ touch filename
   $ ls -l filename
   -rw-r--r--    1 my-user    domain users         0 Jun  9 23:18 filename
   ```

⚠ Important
The `kinit` user is the owner of any new files.

To Authenticate as an AD User Manually by Using autofs and Mount Your Qumulo Cluster

autofs is a dæmon that manages mount points for individual Linux users. For this reason, Linux users have different views of a mount point. `autofs` can authenticate an AD user through `ssh`, the Linux filesystem, or a Qumulo cluster mounted on a Linux system.

> ⚠ Important
> When you use `autofs`, the Linux system maps the root user to the machine account user for the Linux system on the Qumulo cluster. However, the machine account user doesn't have all the privileges of the root user, such as special permissions for the Qumulo cluster. You must specify all permissions in ACLs.

1. Log in to an AD domain and configure `sssd` to authenticate with this domain. For example:

   ```
   $ sudo login my-domain-user
   ```

   Alternatively, you can use the following command.

   ```
   $ ssh my-domain_user@my-linux-system
   ```

2. Configure the `autofs` mappings. For more information, see auto.master in the Linux documentation. The following is an example of a simple configuration that provides a single (direct) mount point which authenticates AD users automatically.

   a. To define a mount point and the path to its map file, add the following line to the `/etc/auto.master` file.

   ```
   /- /etc/auto.kerberos_nfs_mount_example --timeout 60
   ```

   For more information, see Autofs in the Ubuntu documentation.

   b. Add the following line to the `/etc/auto.kerberos_nfs_mount_example` map file.

   ```
   /mnt/qumulo_mount_point -vers=4.1,sec=krb5 my-qumulo-cluster.my-domain.exampl
   e.com:/
   ```

3. Restart `autofs`.

```
$ sudo systemctl restart autofs
```

`autofs` creates the `/mnt/qumulo_mount_point` directory and mounts it as necessary for any user. For example:

```
$ ssh domain_user_1@my-linux-system touch /mnt/qumulo_mount_point/user1_file
$ ssh domain_user_2@my-linux-system touch /mnt/qumulo_mount_point/user2_file
$ ssh domain_user_3@my-linux-system ls -l /mnt/qumulo_mount_point
-rw-r--r--    1 user1    domain users         0 Jun  9 23:18 user1_file
-rw-r--r--    1 user2    domain users         0 Jun  9 23:18 user2_file
```

Important
The user you logged in to the AD domain with is the owner of any new files.

Network Time Protocol (NTP) Server

Kerberos is very sensitive to clock skew. It is important for all systems involved in a Kerberos relationship—the KDC, your Qumulo cluster, and any Linux clients—to have as little clock skew as possible. We recommend using the same NTP server for all three components.

- You can use your AD domain controller as an NTP server. In the Web UI, on the **Active Directory** page, for **Use Active Directory as your primary time server**, click **Yes**.
- To configure any other NTP server in the Web UI, click **Cluster** > **Date & Time**.

There are many NTP dæmons for Linux. For example, Ubuntu uses the NTP functionality in systemd (`timedatectl` and `timesyncd`).

Configuring Cross-Domain Active Directory Trusts

This section describes how the configuration of cross-domain Active Directory (AD) trusts supports NFSv4.1 with Kerberos.

Trusts are relationships between different AD domains. For more information, see Trust Technologies in the Microsoft documentation.

NFSv4.1 with Kerberos and the general AD configuration in Qumulo Core support the same forms of trust relationships.

- Child or parent trusts can:
    - Authenticate as a user from the child domain against the parent domain's AD domain controller (DC).
    - Authenticate as a user from the parent domain against the child domain's AD DC.
- Transitive trusts can authenticate as a user from any of the domains in the transitive trust, against any of the other trusted domains' AD DC.

Configuring the Base DN

For identity mapping to work, you must configure LDAP Base DNs correctly on your Qumulo cluster and on your client. This helps avoid `nobody` or `66534` identity responses that occur when you inspect files that contain trusted users (stored as identities) from other domains. For more information about configuring the Base DN, see Using Active Directory for POSIX Attributes on Qumulo Care.

The following example has trust between between `parent.example.com` and `child.example.com`. In order for both domains' identities to authenticate against a Qumulo cluster, you must configure the cluster and your client with the following Base DN.

```
CN=Users,DC=parent,DC=example,DC=com;CN=Users,DC=child,DC=parent,DC=example,DC=com
```

> **Note**
> AD doesn't prevent duplicate UID or GID numbers from being added to RFC 2307 values. Such improper configuration can cause UID and GID collisions across trusted domains. On Linux, if any collisions occur, the system chooses the first UID or GID that it finds.

Enabling More Secure Trust Encryption Types

While Linux systems disallow deprecated encryption types for Kerberos, Windows prefers RC4 for cross-domain traffic (which Linux systems consider to be deprecated).

For certain trust configurations, you must enable a more secure encryption type for trusted traffic. To enable AES-128 (or SHA1) and AES-256 (or SHA1) for a particular trust, use the `ksetup` command in a Windows Administrator console. For example:

```
$ ksetup /getenctypeattr <domain>
$ ksetup /setenctypeattr <domain> RC4-HMAC-MD5 AES128-CTS-HMAC-SHA1-96 AES256-CTS-HM
AC-SHA1-96
```

Note
This example doesn't disable RC4. Instead, it enables new encryption types *in addition* to RC4. When working with Windows systems, we recommend making additive changes whenever possible. We also recommend staging changes in a safe environment before applying them to a production environment.

Troubleshooting NFSv4.1 with Kerberos

This section describes common troubleshooting procedures when NFSv4.1 doesn't work with Kerberos.

Following General Debugging Techniques

This section lists common debugging techniques.

To Turn Up Logging Levels for Client-Side Tools

1. In the `/etc/sssd.conf` file, set `debug_level = 9`.
2. In the `/etc/idmapd.conf` file, set `Verbosity = 9`.
3. In the `[gssd]` section of the `/etc/nfs.conf` file, set `verbosity=9` and `rpc-verbosity=9`.

   > **Note**
   > In versions of Ubuntu lower than 22.04 (and possibly on other Linux distributions), you can't use the `/etc/nfs.conf` file to configure `gssd`. If this is the case for your system, we recommend starting the `rpc.gssd` service by using the `-n` flag.

4. Turn on `rpcdebug`, for example:

   ```
   rpcdebug -m nfs -s all && rpcdebug -m rpc -s all
   ```

Taking a Client-Side Packet Capture

Normally, there should be:

- Kerberos and LDAP traffic between the client and the domain controller
- DNS traffic between the client and DNS server
- RPC or NFS traffic between the client and the Qumulo cluster

Because a Kerberos mount requires the client to perform a series of steps, in most cases, the last traffic that the client issues indicates the source of failure. To view encrupted Kerberos traffic, use Wireshark with a Kerberos keytab file. For more information, see https://wiki.wireshark.org/Kerberos.md (page 0) in the Wireshark documentation.

For help with interpreting logging and metrics from your Qumulo cluster and for insights from the telemetry of our Kerberos implementation, contact Qumulo Care .

Resolving Incorrect Display of Users or Groups

Under certain conditions, users or groups display as `nobody` when you run the `ls -l` or `stat` command.

Differentiating Client and Cluster Issues

To resolve this issue, determine whether it is with the client or with the cluster by running the `nfs4_getfacl` command on a file. If the presentation in the ACL editor appears correct, the issue is with the client. Otherwise, the issue is with the cluster.

> **Note**
> The ACL editor doesn't perform any ID mapping. It only passes ACE trustees through, in plaintext.

Resolving Client-Side Issues

If the issue is with the client, it is most often an ID mapping issue. Confirm that your mappings are configured correctly. For more information, see User-Defined Identity Mappings on Qumulo Care.

If the issue persists, investigate logging and packet captures.

Resolving Cluster-Side Issues

If the issue is with the cluster, confirm that your cluster's Active Directory settings include the Base DNs that contain the expected users. For more information, see Prerequisites for Joining a Qumulo Cluster to Active Directory (page 19).

Diagnosing Mount-Failed Errors

Under certain conditions, you might receive mount-failed errors from `mount.nfs`. To diagnose this type of error, you can try the following procedures.

1. Confirm that the `rpc.gssd` service is running.
2. Confirm that the cluster and client both resolve from the client. It should be possible to reach the cluster and client through a fully qualified domain name (FQDN), such as `my-machine.my-domain.example.com`.
3. Confirm that reverse DNS works for the IP addresses on both the client and the cluster.
4. Confirm that the client has a `host` service principal name (SPN) and that the cluster has an `nfs` SPN that matches the DNS records.
5. Do one of the following:

- If you use a machine account or `kinit` authentication, confirm that the credentials are correct. You can use the keytab `ktutil` command or the credential cache `klist` command to list the encryption methods.
- Confirm that Kerberos tickets use AES-128 or AES-256 for service encryption by examining a packet capture or your Active Directory Kerberos settings.

6. If you use domain trusts, confirm that trust has AES-128 or AES-256 enabled.
7. Confirm that the clocks on the client, cluster, and domain controller are synchronized to the same time.
8. Confirm that the mount uses the `krb5` option.

> **Note**
> Qumulo Core doesn't support the `krb5i` option.

9. Inspect logs and packet captures.

Creating a Qumulo Core USB Drive Installer

This section explains how you can create a Qumulo Core USB Drive Installer on macOS or Windows.

Prerequisites

- USB drive (4 GB minimum)
- Qumulo Core USB installer image from the Qumulo Care Team

To Create a USB Drive Installer on macOS

1. Open Terminal and log in as `root` by using the `sudo -s` command.
2. Insert your USB drive and then find its disk label by using the `diskutil list` command.

   In the following example, the USB drive's device label is `disk2`.

   ```
   /dev/disk2 (external, physical):
      #:                       TYPE NAME                    SIZE       IDENTIFIER
      0:             Windows_FAT_32 MY_USB_DRIVE             *32.0 GB    disk2
   ```

3. To unmount the USB drive, use your USB drive's device label. For example:

   ```
   diskutil unmountDisk /dev/disk2
   ```

4. To write the Qumulo Core USB installer image to your USB drive, specify the path to your image file and the USB drive's device label. For example:

   ```
   dd if=/path-to-image-file/ of=/dev/rdisk2 bs=2m
   ```

   Note
   If you encounter an **Operation not permitted** error in macOS, do the following.

   a. Navigate to **System Preferences > Security & Privacy**.
   b. On the **Privacy** tab, grant **Full Disk Access** to Terminal.
   c. Restart Terminal and try the command again.

d. When finished, remove **Full Disk Access** from Terminal.

5. Eject your Qumulo Core USB Drive Installer. For example:

```
diskutil eject disk2
```

To Create a USB Drive Installer on Windows

To create a USB Drive Installer on Windows, you must use a third-party application such as Rufus . We recommend Rufus because it can detect many USB storage devices (rather than only Windows-compatible ones).

⚠ Important

- We don't recommend using other tools (such as Win32 Disk Imager) because they might encounter errors when unable to recognize the USB drive after writing data to it.
- When the operation concludes, you might not be able to view the contents of the USB drive on Windows because the drive will be formatted using a different file system.

1. Insert your USB drive and run Rufus.
2. Under **Drive Properties**, select a device and the path to the Qumulo Core USB installer image.
3. For **Partition scheme**, select **MBR** and for **Target System**, select **BIOS or UEFI**.
4. Under **Format Options**, ensure that the **File system** is set to **FAT32 (Default)** and **Cluster size** is set to **4096 bytes (Default)**.
5. Click **Start**.
6. If prompted to download a new version of `GRUB` or `vesamenu.c32`, click **No**.
7. When the **ISOHybrid image detected** dialog box appears, click **Write in DD Image mode** and then click **OK**.
8. To confirm the operation, destroy all data on the USB drive, and image the drive click **OK**.

Increasing the Node-Fault-Tolerance Level for Your Cluster During Node-Add Operations

This section explains how you can increase the node-fault-tolerance level for your cluster during node-add operations.

Reconfiguring Your Cluster's Node-Fault-Tolerance Level

- In Qumulo Core 5.1.2 (and lower), you must configure the node-fault-tolerance level for your cluster when you create the cluster. You can't modify this setting afterwards.
- In Qumulo Core 5.1.3 (and higher), you can reconfigure data protection to increase the node-fault-tolerance level for an existing cluster during the *cluster expansion* process.

> ⚠ Important
>
> - We strongly recommend contacting Qumulo Care before proceeding with cluster expansion.
> - In the following scenarios, Qumulo Core maximizes the usable capacity by default but offers the option to increase the node-fault-tolerance level during the node-add operation by means of a trade-off in the increase of usable capacity.
>   - Your cluster is already heterogeneous.
>   - Your cluster becomes heterogeneous after a node-add operation.

Adding Nodes to Your Cluster

The following sections describe node-add scenarios for various cluster configurations. Identify the scenario that applies to the cluster expansion option that you selected during the purchasing process.

Your Cluster Won't Support an Increased Node-Fault-Tolerance Level

1. Follow the instructions in Add a New Node to an Existing Qumulo Cluster on Qumulo Care.
2. Before you click **Yes** in the **Add <N> nodes to cluster <MyCluster>?** dialog box, check that the projected capacity matches the expected capacity.

To monitor this process, click **Cluster** > **Overview**. On the **Cluster** page, in the protection status section, you can view the rebalance phase status and the estimated time to completion.

When the restriper completes the provisioning of additional usable capacity, the **Data Protected** section shows the same node-fault-tolerance level as before node-add.

Your Cluster Will Support an Increased Node-Fault-Tolerance Level without a Trade-Off in the Increase of Usable Capacity

1. Follow the instructions in Add a New Node to an Existing Qumulo Cluster on Qumulo Care.
2. Before you click **Yes** in the **Add <N> nodes to cluster <MyCluster>?** dialog box, check that the projected capacity matches the expected capacity.

After the cluster expansion process finishes, Qumulo Core begins data protection reconfiguration automatically.

To monitor this process, click **Cluster** > **Overview**. On the **Cluster** page, in the protection status section, you can view the rebalance phase status and the estimated time to completion.

When the restriper completes the provisioning of additional usable capacity and data protection reconfiguration, the **Data Protected** section shows the increased node-fault-tolerance level.

Your Cluster Will Support an Increased Node-Fault-Tolerance Level with a Trade-Off in the Increase of Usable Capacity

This scenario lets you choose one of the following options.

Maintain the Current Node-Fault-Tolerance Level

1. Follow the instructions in Add a New Node to an Existing Qumulo Cluster on Qumulo Care.
2. Before you click **Yes** in the **Add <N> nodes to cluster <MyCluster>?** dialog box, check that the projected capacity matches the expected capacity.

To monitor this process, click **Cluster** > **Overview**. On the **Cluster** page, in the protection status section, you can view the rebalance phase status and the estimated time to completion.

When the restriper completes the provisioning of additional usable capacity, the **Data Protected** section shows the same node-fault-tolerance level as before node-add.

Increase the Node-Fault-Tolerance Level

To begin the node-add operation, contact Qumulo Care .

After the cluster expansion process finishes, Qumulo Core begins data protection reconfiguration automatically.

To monitor this process, click **Cluster** > **Overview**. On the **Cluster** page, in the protection status section, you can view the rebalance phase status and the estimated time to completion.

When the restriper completes the provisioning of additional usable capacity and data protection reconfiguration, the **Data Protected** section shows the increased node-fault-tolerance level.

Using Qumulo Shift-To to Copy Objects to Amazon S3

This section explains how you can use Qumulo Shift-To to copy objects from a directory in a Qumulo cluster to a folder in an Amazon Simple Storage Service (Amazon S3) bucket (cloud object store). For more information about copying objects from S3 to Qumulo, see Using Qumulo Shift-From for Amazon S3 to Copy Objects (page 68).

The guide describes how a Shift-To relationship works and includes information about the prerequisites, IAM permissions, and CLI commands that you can use to copy files and manage Shift relationships.

Prerequisites

- A Qumulo cluster with:
  - Qumulo Core 3.2.1 (and higher) for the CLI and 3.2.5 (and higher) for the Web UI
  - HTTPS connectivity to `s3.<region>.amazonaws.com` though one of the following means:
    - Public Internet
    - VPC endpoint
    - AWS Direct Connect

    For more information, see AWS IP address ranges in the AWS General Reference.
- Membership in a Qumulo role with the following privileges:
  - `PRIVILEGE_REPLICATION_OBJECT_WRITE`: This privilege is required to create a Shift relationship.
  - `PRIVILEGE_REPLICATION_OBJECT_READ`: This privilege is required to view the status of a Shift relationship.

> **Note**
>
> - For any changes to take effect, user accounts with newly assigned roles must log out and log back in (or their sessions must time out).
> - Use special care when granting privileges to roles and users because certain privileges (such as replication-write privileges) can use system privileges to overwrite or move data to a location where a user has greater permissions. This can give a user access to all directories and files in a cluster regardless of

any specific file and directory settings.

- An existing bucket with contents in Amazon S3
- AWS credentials (access key ID and secret access key) with the following permissions:
  - `s3:AbortMultipartUpload`
  - `s3:GetObject`
  - `s3:PutObject`
  - `s3:ListBucket`

  For more information, see Understanding and getting your AWS credentials in the AWS General Reference

Example IAM Policy

In the following example, the IAM policy gives permission to read from and write to the `my-folder` folder in the `my-bucket`. This policy can give users the permissions required to run Shift-To jobs.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": "s3:ListBucket",
      "Effect": "Allow",
      "Resource": "arn:aws:s3:::my-bucket"
    },
    {
      "Action": [
        "s3:AbortMultipartUpload",
        "s3:GetObject",
        "s3:PutObject"
      ]
      "Effect": "Allow",
      "Resource": "arn:aws:s3:::my-bucket/my-folder/*"
    }
  ]
}
```

How Shift-To Relationships Work

Qumulo Core performs the following steps when it creates a Shift-To relationship.

1. Verifies that the directory exists on the Qumulo cluster and that the specified S3 bucket exists, is accessible using the specified credentials, and contains downloadable objects.
2. Creates the Shift-To relationship.
3. Starts a job using one of the nodes in the Qumulo cluster.

   > **Note**
   > If you perform multiple Shift operations, Qumulo Core uses multiple nodes.

4. To ensure that the copy is point-in-time consistent, takes a temporary snapshot of the directory (for example, named `replication_to_bucket_my_bucket`).
5. Recursively traverses the directories and files in the snapshots and copies each object to a corresponding object in S3.
6. Preserves the file paths in the local directory in the keys of replicated objects.

   For example, the file `/my-dir/my-project/file.text`, where `my-dir` is the directory on your Qumulo cluster, is uploaded to S3 as the following object, where `my-folder` is the specified S3 folder.

   ```
   https://my-bucket.s3.us-west-2.amazonaws.com/my-folder/my-project/file.txt
   ```

   > **Note**
   > This process doesn't encode or transform your data in any way. Shift-To replicates only the data in a regular file's primary stream, excluding alternate data streams and file system metadata such as access control lists (ACLs). To avoid transferring data across the public Internet, a server-side S3 copy operation also copies any hard links to files in the replication local directory to S3 as full copies of objects, with identical contents and metadata.

   The following table explains how entities in the Qumulo file system map to entities in an S3 bucket.

Entity in the Qumulo File System	Entity in an Amazon S3 Bucket
Access control list (ACL)	Not copied
Alternate data streams	Not copied
Directory	Not copied (directory structure is preserved in the object key for objects created for files)
Hard link to a non-regular file	Not copied
Hard link to a regular file	Copy of the S3 object
Holes in sparse files	Zeroes (holes are expanded)
Regular file	S3 object (the object key is the file system path and the metadata is the field data)
SMB extended file attributes	Not copied
Symbolic link	Not copied
Timestamps (`mtime` , `ctime` , `atime` , `btime`)	Not copied
UNIX device file	Not copied

7. Checks whether a file is already replicated. If the object exists in the remote S3 bucket, and neither the file nor the object are modified since the last successful replication, its data isn't retransferred to S3.

   > **Note**
   > Shift never deletes files in the remote S3 folder, even if the files are removed from the local directory since the last replication.

8. Deletes the temporary snapshot.

Storing and Reusing Relationships

The Shift-To relationship remains on the Qumulo cluster. You can monitor the completion status of a job, start new jobs for a relationship after the initial job finishes, and delete the relationship (when you no longer need the S3-folder-Qumulo-directory pair). To avoid reuploading objects that a previous copy job uploaded, relationships take up approximately 100 bytes per object. To free this storage, you can delete relationships that you no longer need.

If you repeatedly copy from the same Qumulo directory, you can speed up the upload process (and skip already uploaded files) by using the same relationship.

A new relationship for subsequent uploads doesn't share any tracking information with previous relationships associated with a directory and might recopy data that is already uploaded.

Using the Qumulo Web UI to Copy Files and Manage Relationships

This section describes how you can use the Qumulo Web UI 3.2.5 (and higher) to copy files from a Qumulo cluster to Amazon S3, review Shift relationship details, stop a running copy job, repeat a completed copy job, and delete a relationship.

To Copy Files to Amazon S3

1. Log in to Qumulo Core.
2. Click **Cluster** > **Copy to/from S3**.
3. On the **Copy to/from S3** page, click **Create Copy**.
4. On the **Create Copy to/from S3** page, click **Local ⇨ Remote** and then enter the following:

   a. The **Directory Path** on your cluster (`/` by default)

   b. The S3 **Bucket Name**

   c. The **Folder** in your S3 bucket

   d. The **Region** for your S3 bucket

   e. Your AWS **Region** (`/` by default)

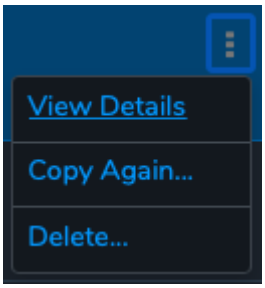
   f. Your AWS **Access Key ID** and **Secret Access Key**.
5. (Optional) For additional configuration, click **Advanced S3 Server Settings**.
6. Click **Create Copy**.
7. In the **Create Copy to S3?** dialog box, review the Shift relationship and then click **Yes, Create**.

   The copy job begins.

To View Configuration Details and Status of Shift Relationships

1. Log in to Qumulo Core.
2. Click **Cluster** > **Copy to/from S3**.

   The **Copy to/from S3** page lists all existing Shift relationships.
3. To get more information about a specific Shift relationship, click ⋮ > **View Details**.

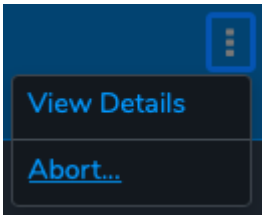


   The **Copy to/from S3 Details** page displays the following information:

   - **Throughput:** average
   - **Run Time**
   - **Data:** total, transferred, and unchanged
   - **Files:** total, transferred, and unchanged

To Stop a Copy Job in Progress

1. Log in to Qumulo Core.
2. Click **Cluster** > **Copy to/from S3**.
3. To stop a copy job for a specific relationship, click ⋮ > **Abort**.



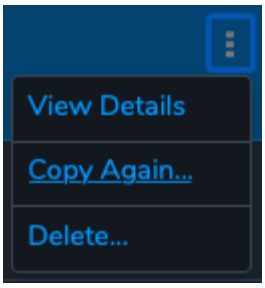
4. In the **Abort copy from?** dialog box, review the Shift relationship and then click **Yes, Abort**.

   The copy job stops.

To Repeat a Completed Copy Job

1. Log in to Qumulo Core.
2. Click **Cluster** > **Copy to/from S3**.

3. To stop a copy job for a specific relationship, click ⋮ > **Copy Again**.

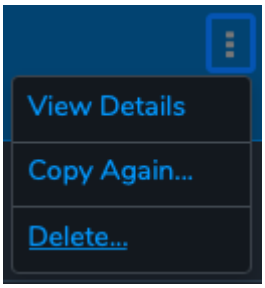


4. In the **Copy again?** dialog box, review the Shift relationship and then click **Yes, Copy Again**.

   The copy job repeats.

To Delete a Shift Relationship

1. Log in to Qumulo Core.
2. Click **Cluster** > **Copy to/from S3**.
3. To stop a copy job for a specific relationship, click ⋮ > **Delete**.



4. In the **Delete copy from?** dialog box, review the Shift relationship and then click **Yes, Delete**.

   The copy job is deleted.

Using the Qumulo CLI to Copy Files and Manage Relationships

This section describes how you can use the Qumulo CLI 3.2.5 (and higher) to copy files from a Qumulo cluster to Amazon S3, review Shift relationship details, stop a running copy job, repeat a completed copy job, and delete a relationship.

Copying Files from Amazon S3

To copy files, use the `replication_create_object_relationship` command and specify the following:

- Local directory path on Qumulo cluster
- Copy direction (copy-to)
- S3 object folder
- S3 bucket
- AWS region
- AWS access key ID
- AWS secret access key

The following example shows how you can create a relationship between the directory `/my-dir/` on a Qumulo cluster and the S3 bucket `my-bucket` and folder `/my-folder/` in the `us-west-2` AWS region. The secret access key is associated with the access key ID.

```
qq replication_create_object_relationship \
  --source-directory-path /my-dir/ \
  --direction COPY_TO_OBJECT \
  --object-folder /my-folder/ \
  --bucket my-bucket \
  --region us-west-2 \
  --access-key-id AKIAIOSFODNN7EXAMPLE \
  --secret-access-key wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
```

The CLI returns the details of the relationship in JSON format, for example:

```
{
  "access_key_id": "ABC",
  "bucket": "my-bucket",
  "object_store_address": "s3.us-west-2.amazonaws.com",
  "id": "1c23b4ed-5c67-8f90-1e23-a4f5f6ceff78",
  "object_folder": "my-folder/",
  "port": 443,
  "ca_certificate": null,
  "region": "us-west-2",
  "source_directory_id": "3",
  "direction": "COPY_TO_OBJECT",
}
```

Viewing Configuration Details and Status of Shift Relationships

- To view configuration details for all Shift relationships, use the `replication_list_object_relationships` command.

- To view configuration details for a specific relationship, use the `replication_get_object_relationship` command followed by the `--id` and the Shift relationship ID (GUID), for example:

```
 qq replication_get_object_relationship --id 1c23b4ed-5c67-8f90-1e23-a4f5f6cef
f78
```

- To view the status of a specific relationship, use the `replication_get_object_relationship_status` command followed by the `--id` and the Shift relationship ID.
- To view the status of all relationships, use the `replication_list_object_relationship_statuses` command.

  The CLI returns the details of all relationships in JSON format, for example:

```
[
  {
    "direction": "COPY_TO_OBJECT",
    "access_key_id": "AKIAIOSFODNN7EXAMPLE",
    "bucket": "my-bucket",
    "object_store_address": "s3.us-west-2.amazonaws.com",
    "id": "1c23b4ed-5c67-8f90-1e23-a4f5f6ceff78",
    "object_folder": "my-folder/",
    "port": 443,
    "ca_certificate": null,
    "region": "us-west-2",
    "source_directory_id": "3",
    "source_directory_path": "/my-dir/",
    "state": "REPLICATION_RUNNING",
    "current_job": {
      "start_time": "2020-04-06T17:56:29.659309904Z",
      "estimated_end_time": "2020-04-06T21:54:33.244095593Z",
      "job_progress": {
        "bytes_transferred": "178388608",
        "bytes_unchanged": "0",
        "bytes_remaining": "21660032",
        "bytes_total": "200048640",
        "files_transferred": "17",
        "files_unchanged": "0",
        "files_remaining": "4",
        "files_total": "21",
        "percent_complete": 89.0368314738253,
        "throughput_current": "12330689",
        "throughput_overall": "12330689"
      }
    },
    "last_job": null
  }
]
```


The `state` field shows the `REPLICATION_RUNNING` status and the `current_job` field shows the job's progress. When Qumulo Core copies files from S3, details for the most recently completed job become available in the `last_job` field, the `state` field changes to `REPLICATION_NOT_RUNNING`, and the `current_job` field reverts to `null`.

> **Note**
> If you already ran a job for a relationship, it is possible for both the `current_job` and `last_job` fields to be non-null while you run a new job.

The `bytes_total` and `files_total` fields represent the total amount of data and number of files to be transferred by a Shift job. The `bytes_remaining` and `files_remaining` fields show the amount of data and number of files not yet transferred. The values of these four fields don't stabilize until the work estimation for the job is complete.

The `percent_complete` field displays the overall job progress and the `estimated_end_time` field displays the time at which the job is estimated to be complete. The values of these two fields are populated when the work estimation for the job is complete.

Stopping a Copy Job in Progress

To stop a copy job already in progress, use the `replication_abort_object_relationship` command followed by the `--id` and the Shift relationship ID.

Repeating a Completed Copy Job

To repeat a completed copy job, use the `replication_start_object_relationship` command followed by the `--id` and the Shift relationship ID.

This command begins a new job for the existing relationship and downloads any content that changed in the S3 bucket or on the Qumulo cluster since the time the previous job ran.

Deleting a Shift Relationship

After your copy job is complete, you can delete your Shift relationship. To do this, run the `replication_delete_object_relationship` command followed by the `--id` and the Shift relationship ID.

> **Note**
> You can run this command only against a relationship that doesn't have any active jobs running.

This command removes the copy job's record, leaving locally stored objects unchanged. Any storage that the relationship used to track downloaded objects becomes available when you delete the relationship.

Troubleshooting Copy Job Issues

Any fatal errors that occur during a copy job cause the job to fail, leaving a partially copied set of files in the directory in your S3 bucket. However, to let you review the Shift relationship status any failure messages, the Shift relationship continues to exist. You can start a new job to complete the copying of objects to the S3 bucket—any successfully transferred files from the previous job aren't retransferred from your Qumulo cluster.

Whenever Qumulo Core doesn't complete an operation successfully and returns an error from the API or CLI, the `error` field within the `last_job` field (that the `replication_list_object_relationship_statuses` command returns) contains a detailed failure message. For more troubleshooting details, see `qumulo-replication.log` on your Qumulo cluster.

Best Practices

We recommend the following best practices for working with Qumulo Shift-To for Amazon S3.

- **Bucket Lifecycle Policy:** To abort any incomplete uploads older than several days and ensure the automatic clean-up of any storage that incomplete parts of large objects (left by failed or interrupted replication operations) use, configure a bucket lifecycle policy. For more information, see Uploading and copying objects using multipart upload in the *Amazon Simple Storage Service User Guide*.
- **VPC Endpoints:** For best performance when using a Qumulo cluster in AWS, configure a VPC endpoint to S3. For on-premises Qumulo clusters, we recommend AWS Direct Connect or another high-bandwidth, low-latency connection to S3.
- **Unique Artifacts:** To avoid collisions between different data sets, specify a unique object folder or unique bucket for each replication relationship from a Qumulo cluster to S3.
- **Object Versioning:** To protect against unintended overwrites, enable object versioning. For more information, see Using versioning in S3 buckets in the *Amazon Simple Storage Service User Guide*.
- **Completed Jobs:** If you don't plan to use a Shift relationship to download updates from S3, delete the relationship to free up any storage associated with it.
- **Concurrent Replication Relationships:** To increase parallelism, especially across distinct datasets, use concurrent replication relationships to S3. To avoid having a large number of concurrent operations impact client I/O to the Qumulo cluster, limit the number of concurrent replication relationships. While there is no hard limit, we don't recommend creating more than 100 concurrent replication relationships on a cluster (including both Shift and Qumulo local replication relationships).

Restrictions

- **Object-Locked Buckets:** You can't use buckets configured with S3 Object Lock and a default retention period for Shift-To. If possible, either remove the defauly retention period and set retention periods explicitly on objects uploaded outside of Shift or use a different S3 bucket without S3 Object Lock enabled. For more information, see How S3 Object Lock works in the *Amazon Simple Storage Service User Guide*.
- **File Size Limit:** The size of an individual file can't exceed 5 TiB (this is the maximum object size that S3 supports). There is no limit on the total size of all your files.

- **File Path Limit:** The length of a file path must be shorter than 1,024 characters, including the configured object folder prefix, excluding the local directory path.
- **Hard Links:** Qumulo Core 3.2.3 (and higher) supports hard links, up to the maximum object size that S3 supports.
- **Objects Under the Same Key:** Unless an object contains Qumulo-specific hash metadata that matches a file, any object that exists under the same key that a new relationship replicates *is overwritten*. To retain older versions of overwritten objects, enable versioning for your S3 bucket. For more information, see Using versioning in S3 buckets in the *Amazon Simple Storage Service User Guide*.
- **Object Checksums:** All files replicated using S3 server-side integrity verification (during upload) use a SHA256 checksum stored in the replicated object's metadata.
- **S3-Compatible Object Stores:** S3-compatible object stores aren't supported. Currently, Qumulo Shift-To supports replication only to Amazon S3.
- **HTTP:** HTTP isn't supported. All Qumulo connections are encrypted using HTTPS and verify the S3 server's SSL certificate.
- **Anonymous Access:** Anonymous access isn't supported. You must use valid AWS credentials.
- **Replication without Throttling:** Replication provides no throttling and might use all available bandwidth. If necessary, use Quality of Service rules on your network.
- **Amazon S3 Standard Storage Class:** Qumulo Shift-To supports uploading only objects stored in the Amazon S3 Standard storage class. You can't download objects stored in the Amazon S3 Glacier or Deep Archive storage classes and any buckets that contain such objects cause a copy job to fail.
- **Content-Type Metadata:** Because all objects are stored in S3 using the default `binary/octet-stream` content type, they might be interpreted as binary data if you download them using a browser. To attach content-type metadata to your objects, use the AWS Console.

Using Qumulo Shift-From to Copy Objects from Amazon S3

This section explains how you can use Qumulo Shift-From to copy objects from a folder in an Amazon Simple Storage Service (Amazon S3) bucket (cloud object store) to a directory in a Qumulo cluster. For more information about copying objects from Qumulo to S3, see Using Qumulo Shift-To for Amazon S3 to Copy Objects (page 55) on Qumulo Care.

The guide describes how a Shift-From relationship works and includes information about the prerequisites, IAM permissions, and CLI commands that you can use to copy files and manage Shift relationships. From Qumulo Core 4.3.4, Shift-From estimates the work that a copy job performs.

Prerequisites

- A Qumulo cluster with:
  - Qumulo Core 4.2.3 (or higher)
  - HTTPS connectivity to `s3.<region>.amazonaws.com` though one of the following means:
    - Public Internet
    - VPC endpoint
    - AWS Direct Connect

    For more information, see AWS IP address ranges in the AWS General Reference.
- Membership in a Qumulo role with the following privileges:
  - `PRIVILEGE_REPLICATION_OBJECT_WRITE`: This privilege is required to create a Shift relationship.
  - `PRIVILEGE_REPLICATION_OBJECT_READ`: This privilege is required to view the status of a Shift relationship.

> **Note**
>
> - For any changes to take effect, user accounts with newly assigned roles must log out and log back in (or their sessions must time out).
> - Use special care when granting privileges to roles and users because certain privileges (such as replication-write privileges) can use system privileges to overwrite or move data to a location where a user has greater permissions.

> This can give a user access to all directories and files in a cluster regardless of any specific file and directory settings.

- An existing bucket with contents in Amazon S3
- AWS credentials (access key ID and secret access key) with the following permissions:
  - `s3:GetObject`
  - `s3:ListBucket`

  For more information, see Understanding and getting your AWS credentials in the AWS General Reference

Example IAM Policy

In the following example, the IAM policy gives permission to read from and write to the `my-folder` folder in the `my-bucket`. This policy can give users the minimal set of permissions required to run Shift-From jobs. (Shift-To jobs require a less-restrictive policy. For more information and an example, see Using Qumulo Shift-To for Amazon S3 to Copy Objects (page 55).)

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": "s3:ListBucket",
      "Effect": "Allow",
      "Resource": "arn:aws:s3:::my-bucket"
    },
    {
      "Action": [
        "s3:GetObject"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:s3:::my-bucket/my-folder/*"
    }
  ]
}
```

How Shift-From Relationships Work

Qumulo Core performs the following steps when it creates a Shift-From relationship.

1. Verifies that the directory exists on the Qumulo cluster and that the specified S3 bucket exists, is accessible using the specified credentials, and contains downloadable objects.
2. Creates the Shift-From relationship.
3. Starts a job using one of the nodes in the Qumulo cluster.

   > **Note**
   > If you perform multiple Shift operations, Qumulo Core uses multiple nodes.

4. Lists the contents of the S3 folder and downloads the objects to the specified directory on your Qumulo cluster.
5. Forms the full path of the file on the Qumulo custer by appending the path of the object (relative to the S3 folder) to the directory path on the Qumulo cluster.

   For example, the following object is downloaded to `/my-dir/my-project/file.text`, where `my-folder` is the specified S3 folder and `my-dir` is the directory on your Qumulo cluster.

   ```
   https://my-bucket.s3.us-west-2.amazonaws.com/my-folder/my-project/file.txt
   ```

   > **Note**
   > This process doesn't encode or transform your data in any way. Shift-From attempts only to map every S3 object in the specified folder to a file on your Qumulo cluster.

6. Avoids redownloading an unchanged object in a subsequent job by tracking the information about an object and its replicated object.

   > **Note**
   > If you rename or move an object or local file between jobs, or if there are any metadata changes in S3 or Qumulo, the object is replicated again.

Storing and Reusing Relationships

The Shift-From relationship remains on the Qumulo cluster. You can monitor the completion status of a job, start new jobs for a relationship after the initial job finishes, and delete the relationship (when you no longer need the S3-folder-Qumulo-directory pair). To avoid redownloading objects that a previous copy job downloaded, relationships take up approximately 100 bytes per object. To free this storage, you can delete relationships that you no longer need.

If you repeatedly download from the same S3 folder, you can speed up the download process (and skip already downloaded files) by using the same relationship.

A new relationship for subsequent downloads doesn't share any tracking information with previous relationships associated with a directory and might recopy data that is already downloaded.

Using the Qumulo Web UI to Copy Files and Manage Relationships

This section describes how you can use the Qumulo Web UI 4.2.5 (and higher) to copy files from Amazon S3 to a Qumulo cluster, review Shift relationship details, stop a running copy job, repeat a completed copy job, and delete a relationship.

To Copy Files from Amazon S3

1. Log in to Qumulo Core.
2. Click **Cluster** > **Copy to/from S3**.
3. On the **Copy to/from S3** page, click **Create Copy**.
4. On the **Create Copy to/from S3** page, click **Local ⇦ Remote** and then enter the following:

   a. The **Directory Path** on your cluster (`/` by default)

   b. The S3 **Bucket Name**

   c. The **Folder** in your S3 bucket

   d. The **Region** for your S3 bucket

   e. Your AWS **Region** (`/` by default)

   f. Your AWS **Access Key ID** and **Secret Access Key**.
5. (Optional) For additional configuration, click **Advanced S3 Server Settings**.
6. Click **Create Copy**.
7. In the **Create Copy from S3?** dialog box, review the Shift relationship and then click **Yes, Create**.

   The copy job begins and Qumulo Core estimates the work to be performed. When the estimation is complete, the Web UI displays a progress bar with a percentage for a relationship on the **Replication Relationships** page. The page also displays the estimated total work, the remaining bytes and files, and the estimated time to completion for a running copy job.

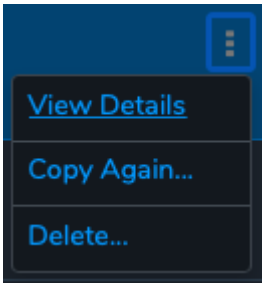
   > **Note**
   > For work estimates, Shift-From jobs calculate the total number of files and bytes in a job's bucket prefix. This requires the job to use the `ListObjectV2` S3 action once per

5,000 objects (or 200 times per 1 million objects).

To View Configuration Details and Status of Shift Relationships

1. Log in to Qumulo Core.
2. Click **Cluster** > **Copy to/from S3**.

   The **Copy to/from S3** page lists all existing Shift relationships.
3. To get more information about a specific Shift relationship, click ⋮ > **View Details**.

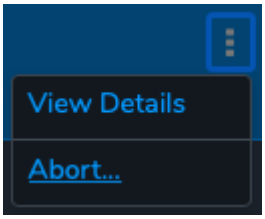


   The **Copy to/from S3 Details** page displays the following information:

   - **Throughput:** average
   - **Run Time**
   - **Data:** total, transferred, and unchanged
   - **Files:** total, transferred, and unchanged

To Stop a Copy Job in Progress

1. Log in to Qumulo Core.
2. Click **Cluster** > **Copy to/from S3**.
3. To stop a copy job for a specific relationship, click ⋮ > **Abort**.

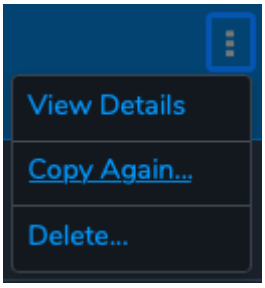


4. In the **Abort copy from?** dialog box, review the Shift relationship and then click **Yes, Abort**.

   The copy job stops.

To Repeat a Completed Copy Job

1. Log in to Qumulo Core.
2. Click **Cluster** > **Copy to/from S3**.
3. To stop a copy job for a specific relationship, click ⋮ > **Copy Again**.

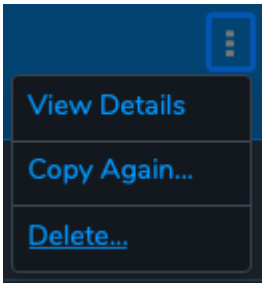


4. In the **Copy again?** dialog box, review the Shift relationship and then click **Yes, Copy Again**.

   The copy job repeats.

To Delete a Shift Relationship

1. Log in to Qumulo Core.
2. Click **Cluster** > **Copy to/from S3**.
3. To stop a copy job for a specific relationship, click ⋮ > **Delete**.



4. In the **Delete copy from?** dialog box, review the Shift relationship and then click **Yes, Delete**.

   The copy job is deleted.

Using the Qumulo CLI to Copy Files and Manage Relationships

This section describes how you can use the Qumulo CLI to copy files from Amazon S3 to a Qumulo cluster, review Shift relationship details, stop a running copy job, repeat a completed copy job, and delete a relationship.

Copying Files from Amazon S3

To copy files, use the `replication_create_object_relationship` command and specify the following:

- Local directory path on Qumulo cluster
- Copy direction (copy-from)
- S3 object folder
- S3 bucket
- AWS region
- AWS access key ID
- AWS secret access key

The following example shows how you can create a relationship between the directory `/my-dir/` on a Qumulo cluster and the S3 bucket `my-bucket` and folder `/my-folder/` in the `us-west-2` AWS region. The secret access key is associated with the access key ID.

```
qq replication_create_object_relationship \
  --local-directory-path /my-dir/ \
  --direction COPY_FROM_OBJECT \
  --object-folder /my-folder/ \
  --bucket my-bucket \
  --region us-west-2 \
  --access-key-id AKIAIOSFODNN7EXAMPLE \
  --secret-access-key wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
```

The CLI returns the details of the relationship in JSON format, for example:

```
{
  "access_key_id": "ABC",
  "bucket": "my-bucket",
  "object_store_address": "s3.us-west-2.amazonaws.com",
  "id": "1c23b4ed-5c67-8f90-1e23-a4f5f6ceff78",
  "object_folder": "my-folder/",
  "port": 443,
  "ca_certificate": null,
  "region": "us-west-2",
  "local_directory_id": "3",
  "direction": "COPY_FROM_OBJECT",
}
```

Viewing Configuration Details and Status of Shift Relationships

- To view configuration details for all Shift relationships, use the `replication_list_object_relationships` command.
- To view configuration details for a specific relationship, use the `replication_get_object_relationship` command followed by the `--id` and the Shift relationship ID (GUID), for example:

  ```
  qq replication_get_object_relationship --id 1c23b4ed-5c67-8f90-1e23-a4f5f6cef
  f78
  ```

- To view the status of a specific relationship, use the `replication_get_object_relationship_status` command followed by the `--id` and the Shift relationship ID.
- To view the status of all relationships, use the `replication_list_object_relationship_statuses` command.

  The CLI returns the details of all relationships in JSON format, for example:

```
[
  {
    "direction": "COPY_FROM_OBJECT",
    "access_key_id": "AKIAIOSFODNN7EXAMPLE",
    "bucket": "my-bucket",
    "object_store_address": "s3.us-west-2.amazonaws.com",
    "id": "1c23b4ed-5c67-8f90-1e23-a4f5f6ceff78",
    "object_folder": "my-folder/",
    "port": 443,
    "ca_certificate": null,
    "region": "us-west-2",
    "local_directory_id": "3",
    "local_directory_path": "/my-dir/",
    "state": "REPLICATION_RUNNING",
    "current_job": {
      "start_time": "2020-04-06T17:56:29.659309904Z",
      "estimated_end_time": "2020-04-06T21:54:33.244095593Z",
      "job_progress": {
        "bytes_transferred": "178388608",
        "bytes_unchanged": "0",
        "bytes_remaining": "21660032",
        "bytes_total": "200048640",
        "files_transferred": "17",
        "files_unchanged": "0",
        "files_remaining": "4",
        "files_total": "21",
        "percent_complete": 89.0368314738253,
        "throughput_current": "12330689",
        "throughput_overall": "12330689"
      }
    },
    "last_job": null
  }
]
```


The `state` field shows the `REPLICATION_RUNNING` status and the `current_job` field shows the job's progress. When Qumulo Core copies files from S3, details for the most recently completed job become available in the `last_job` field, the `state` field changes to `REPLICATION_NOT_RUNNING`, and the `current_job` field reverts to `null`.

> **Note**
> If you already ran a job for a relationship, it is possible for both the `current_job` and `last_job` fields to be non-null while you run a new job.

The `bytes_total` and `files_total` fields represent the total amount of data and number of files to be transferred by a Shift job. The `bytes_remaining` and `files_remaining` fields show the amount of data and number of files not yet transferred. The values of these four fields don't stabilize until the work estimation for the job is complete.

The `percent_complete` field displays the overall job progress and the `estimated_end_time` field displays the time at which the job is estimated to be complete. The values of these two fields are populated when the work estimation for the job is complete.

Shift-From performs a single task that estimates the amount of content to copy by listing all files and summing up their contents. Until this task is complete, the `percent_complete` field is set to `"None"` and the `estimated_end_time` field is set to `""`. To list the bucket prefix content in sets of 5,000 objects, this task uses the `ListObjectV2` S3 action .

Stopping a Copy Job in Progress

To stop a copy job already in progress, use the `replication_abort_object_relationship` command followed by the `--id` and the Shift relationship ID.

Repeating a Completed Copy Job

To repeat a completed copy job, use the `replication_start_object_relationship` command followed by the `--id` and the Shift relationship ID.

This command begins a new job for the existing relationship and downloads any content that changed in the S3 bucket or on the Qumulo cluster since the time the previous job ran.

Deleting a Shift Relationship

After your copy job is complete, you can delete your Shift relationship. To do this, run the `replication_delete_object_relationship` command followed by the `--id` and the Shift relationship ID.

> **Note**
> You can run this command only against a relationship that doesn't have any active jobs running.

This command removes the copy job's record, leaving locally stored objects unchanged. Any storage that the relationship used to track downloaded objects becomes available when you delete the relationship.

Troubleshooting Copy Job Issues

Any fatal errors that occur during a copy job cause the job to fail, leaving a partially copied set of files in the directory on your Qumulo cluster. However, to let you review the Shift relationship status any failure messages, the Shift relationship continues to exist. You can start a new job to complete the copying of objects from the S3 bucket—any successfully transferred files from the previous job aren't retransferred to your Qumulo cluster.

Whenever Qumulo Core doesn't complete an operation successfully and returns an error from the API or CLI, the `error` field within the `last_job` field (that the `replication_list_object_relationship_statuses` command returns) contains a detailed failure message. For more troubleshooting details, see `qumulo-replication.log` on your Qumulo cluster.

Best Practices

We recommend the following best practices for working with Qumulo Shift-From for Amazon S3.

- **Inheritable Permissions:** Because the system user creates the files copied using Shift-From for S3, the system owns these files. By default, Everyone will be granted Read permissions, and administrators always have full access to the files.

  To assign the necessary permissions to copied files, you must assign the necessary inheritable permissions to the root directory of the relationship **before** creating a Copy from S3 relationship. This ensures that the copied subdirectories and files inherit the permissions.

  Windows Security Dialog or `qq fs_modify_acl` can be used to edit permissions on a directory. See Qumulo-File-Permissions-Overview to learn more about file permissions.

- **VPC Endpoints:** For best performance when using a Qumulo cluster in AWS, configure a VPC endpoint to S3. For on-premises Qumulo clusters, we recommend AWS Direct Connect or another high-bandwidth, low-latency connection to S3.

- **Repeated Synchronization:** If you need to repeatedly synchronize an S3 folder with a Qumulo directory, we recommend reusing the same relationship. This lets you avoid repeated downloading of unchanged objects that already exist locally.

- **Completed Jobs:** If you don't plan to use a Shift relationship to download updates from S3, delete the relationship to free up any storage associated with it.

- **Concurrent Replication Relationships:** To increase parallelism, especially across distinct datasets, use concurrent replication relationships from S3. To avoid having a large number of concurrent operations impact client I/O to the Qumulo cluster, limit the number of concurrent replication relationships. While there is no hard limit, we don't recommend creating more than 100 concurrent replication relationships on a cluster (including both Shift and Qumulo local replication relationships).

Restrictions

- **S3-Compatible Object Stores:** S3-compatible object stores aren't supported. Currently, Qumulo Shift-From supports replication only from Amazon S3.
- **HTTP:** HTTP isn't supported. All Qumulo connections are encrypted using HTTPS and verify the S3 server's SSL certificate.
- **Anonymous Access:** Anonymous access isn't supported. You must use valid AWS credentials.
- **Replication without Throttling:** Replication provides no throttling and might use all available bandwidth. If necessary, use Quality of Service rules on your network.
- **Amazon S3 Standard Storage Class:** Qumulo Shift-From supports downloading only objects stored in the Amazon S3 Standard storage class. You can't download objects stored in the Amazon S3 Glacier or Deep Archive storage classes and any buckets that contain such objects cause a copy job to fail.
- **Disallowed Amazon S3 Paths in Qumulo Clusters:** Certain allowed Amazon S3 paths can't be copied to Qumulo clusters and cause a copy job to fail. Disallowed paths contain:
    - A trailing slash (`/`) character (with non-zero object content length)
    - Consecutive slash (`/`) characters
    - Single and double period (`.`, `..`) characters
    - The path component `.snapshot`
- **Disallowed Conflicting Types:** When content in an S3 bucket or Qumulo directory changes over time, a conflict related to type mismatches might arise, the Shift-from job fails, and an error message gives details about the conflict. For example, a conflict might occur when a remote object maps to a local file system directory entry which:
    - Is a regular file with two or more links
    - Isn't a regular file (for example, a directory or a special file)
- **Disallowed Amazon S3 Path Configurations:** Because of conflicting type requirements, Qumulo Core can't recreate certain allowed Amazon S3 path configurations on Qumulo clusters. For example, if an S3 bucket contains objects `a/b/c` and `a/b`, then path `a/b` must be both a file and directory on a Qumulo cluster. Because this isn't possible, this configuration causes a copy job to fail.
- **Directories in Multiple Relationships:** A directory on a Qumulo cluster for one Shift relationship can't overlap with a directory used for another Shift relationship, or with a remote directory for a Qumulo-to-Qumulo replication relationship. This causes the relationship creation to fail.

- Changes to S3 Folder During Copy Job: Currently, Shift-From assumes that the S3 folder remains unchanged throughout the copy job. Any changes (deleting, archiving, or modifying an object) during the copy job might cause a copy job to fail.
- Read-Only Local Directory: When the Shift-From copy job begins, the local directory on the Qumulo cluster becomes read-only. While no external clients can modify anything in the directory or its subdirectories, all content remains readable. When the copy job is complete, the directory reverts to its previous permissions.
- Partially Downloaded Files: If a copy job is interrupted or encounters a fatal error (that can't be resolved by retrying the operation), Qumulo Core attempts to delete partially downloaded files. Because this is a best-effort process, certain interruptions can prevent the cleanup of partially downloaded files.

Managing Snapshots

The **Snapshots** page in Qumulo Core 4.3.3 (and higher) lets you view and manage large numbers of saved snapshots without having to make API queries. This makes it possible to delegate snapshot management operations to a wide range of users.

To View Your Snapshots

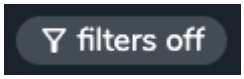
The **Snapshots** page lets you navigate a large number of snapshots.

1. Log in to Qumulo Core.
2. Click **Cluster** > **Saved Snapshots**.
3. If you have more than 50 snapshots, click to navigate the snapshot pages.

   You can also use the controls at the bottom of the table to navigate to a specific page or change the number of rows per page.

To Find a Specific Snapshot

The table on the **Snapshots** page has a filtering mode that lets you search for a specific snapshot by name, creation time, or any other column.

1. Log in to Qumulo Core.
2. Click **Cluster** > **Saved Snapshots**.
3. At the top of the table, click 

.

   The **Search...** field appears.
4. Enter a search query.

   The table rows filter to match your query as you type.
5. To toggle filtering off, click

.

To Delete a Single Snapshot

1. Log in to Qumulo Core.
2. Click **Cluster** > **Saved Snapshots**.
3. On the right-most side of a snapshot's row, click .

To Delete Multiple Snapshots

1. Log in to Qumulo Core.
2. Click **Cluster** > **Saved Snapshots**.
3. On the left-most side of the table, click the checkbox for every snapshot you want to delete.

   When you select more than one row, the **Delete** button appears.
4. When you finish selecting snapshots, click **Delete**.

> **Note**
> All selection and deletion controls modify only the current page. You can't delete a snapshot accidentally if it isn't listed on the current page (because it is on a different page or is filtered out).

Setting the Web UI Login Banner

This section explains how you can set a login banner for the Qumulo Core Web UI.

In Qumulo Core 5.2.1 (and higher), clusters have an optional login banner that users must acknowledge before being they can log in to the Web UI.

To Set the Web UI Login Banner

To set the login banner, use the `web_ui_modify_settings` command. To specify the Markdown file to use for the banner, use the `--login-banner` flag. For example:

```
qq web_ui_modify_settings --login-banner my-banner.md
```

To Clear the Web UI Login Banner

To clear the login banner, use the `web_ui_modify_settings` command with the `--disable-login-banner` flag.

```
qq web_ui_modify_settings --disable-login-banner
```

To View the Current Web UI Login Banner

To view the current login banner, use the `web_ui_get_settings` command with the `--login-banner` flag.

```
qq web_ui_get_settings --login-banner
```

Setting the Web UI Inactivity Timeout

In Qumulo Core 5.1.0 (and higher), clusters have an optional *inactivity timeout* that logs users out of the Web UI if they don't interact with it for a specified amount of time.

> **Note**
> During the final minute of the timeout period, the **Your Session is About to Expire** dialog box appears. The dialog box shows a countdown and lets the user renew the session or log out immediately. When deciding on the timeout length, take your users' needs into consideration.

To Set the Web UI Inactivity Timeout

To set an inactivity timeout, use the `web_ui_modify_settings` command. Specify the timeout in minutes by using the `--inactivity-timeout` flag. For example:

```
qq web_ui_modify_settings --inactivity-timeout 15
```

To Clear the Web UI Inactivity Timeout

To clear an inactivity timeout, use the `web_ui_modify_settings` command with the `--disable-inactivity-timeout` flag.

```
qq web_ui_modify_settings --disable-inactivity-timeout
```

To View the Current Web UI Inactivity Timeout

To view the current inactivity timeout, use the `web_ui_get_settings` command:

```
qq web_ui_get_settings
```

Supported Configurations and Known Limits for Qumulo Core

This section provides an overview of supported configurations and known limits for Qumulo Core.

Supported Configurations

Configuration Type	Supported Value
Protocols	• FTP • NFSv3 • NFSv4.1 • SMB 2.002 • SMB 2.1 • SMB 3.0 • SMB 3.1 For more information, see Enabling and Using NFSv4.1 on a Qumulo Cluster (page 2) and Managing File Access Permissions by Using NFSv4.1 Access Control Lists (ACLs) (page 9).
Browser	Google Chrome 80 (and higher)
Clients over SMB	• macOS 10.14 (and higher) • Windows 7 (and higher)
Clients over NFS	• macOS 10.14 (and higher) • Linux Kernel 2.6.X (and higher)
Linux Configuration	Qumulo Core is up to date with all Ubuntu 18.04 security updates.

Configuration Type	Supported Value
Domain-Functional Level	Microsoft Windows Server 2008 R2 (and higher) **Note** Qumulo Core doesn't support Samba Domain Controllers.
Kerberos V5 Encryption Types	• RC4-HMAC-MD5 • AES256-CTS-HMAC-SHA1 • AES128-CTS-HMAD-SHA1
LDAP Servers	OpenLDAP for Group Expansion
Python Version for `qq` CLI	3.8 (and higher)

Known Limits

Limit Type	Maximum Value
On-Premises Cluster Size	100 nodes
Cloud Cluster Size	100 nodes
Floating IP Addresses per Node	10
NFS Exports	64,000
SMB Shares	40,000
Access Control Entries (ACEs) in an Access Control List (ACL)	200
NFS Groups	16 without RFC 2307 with Kerberos
Combined Users and Groups	4 billion
Characters in Cluster Name	2-15, alphanumeric and hyphen (-)

Limit Type	Maximum Value
Characters in Full Path (Path Name)	32,760 (protocol-limited)
Characters in File Path Component (File or Directory)	255
Files in a Directory	4.3 billion
File Size	9 exabytes
Number of Files	18 quintillion
Hard Links per File	1,024
LDAP Domains	1
Active Directory Domains	1
DNS Servers	3
Snapshots	40,000
Quotas	Not specified
Number of Replication Relationships	100 **Note** If a directory is more than 100 levels below the file system root directory, you can't use it as a replication source.