Qumulo Administrator Guide

January 17, 2023

Copyright © 2023 Qumulo, Inc.

Table of Contents

Getting Started with Qumulo Core

Creating a Qumulo Core USB Drive Installer 3
Installing VPN Keys on a Qumulo Cluster 6
Configuring SAML Single Sign-On (SSO) for Your Qumulo Cluster 9
Supported Configurations and Known Limits for Qumulo Core 17
Qumulo Compliance Posture 20

Upgrading Qumulo Core

Performing Instant Software Upgrades and Platform Upgrades 21
Qumulo Core Upgrade Mode Reference 24

Configuring Networking for Qumulo Core

Required Networking Ports for Qumulo Core 30

Connecting Qumulo Core to External Services

Using Qumulo Core Access Tokens 33

Working with the Qumulo Core Web UI

Setting the Web UI Login Banner 42
Setting the Web UI Inactivity Timeout 43

Working with the qq CLI

Enabling Autocomplete for the qq CLI 44

Protecting Your Data

Increasing the Node-Fault-Tolerance Level for Your Cluster During Node-Add Operations 47
Managing Snapshots 49

Moving Your Data

Using Qumulo Shift-To to Copy Objects to Amazon S3 51
Using Qumulo Shift-From for Amazon S3 to Copy Objects 63

Working with File System Protocols

Enabling and Using NFSv4.1 on a Qumulo Cluster 76
Managing File Access Permissions by Using NFSv4.1 Access Control Lists (ACLs) 83
Watching for File Attribute and Directory Changes by Using SMB2 CHANGE_NOTIFY 92

Using the S3 API with Qumulo Core

Getting Started with the S3 API 98
Creating and Managing S3 Access Keys 103
Creating and Managing S3 Buckets 111
Supported S3 Functionality and Known Limits for Qumulo Core 124

Using NFSv4.1 with Kerberos in Qumulo Core

How NFSv4.1 Works with Kerberos in Qumulo Core 130
Prerequisites for Joining a Qumulo Cluster to Active Directory 132
Configuring Active Directory for Use With Kerberos 134
Performing Additional Cluster Configuration after Joining Active Directory 138
Using Kerberos Permissions in the Qumulo Filesystem 141
Configuring a Linux Client for NFSv4.1 with Kerberos 147
Configuring Cross-Domain Active Directory Trusts 157
Troubleshooting NFSv4.1 with Kerberos 159

Configuring and Collecting Metrics in Qumulo Core

Qumulo OpenMetrics API Specification 162

Getting Started with Qumulo Core

Creating a Qumulo Core USB Drive Installer

This section explains how to 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 by 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**.

Installing VPN Keys on a Qumulo Cluster

This section explains how to install VPN keys on your Qumulo cluster over a network. You can install the VPN keys by using the `qq` CLI from a machine on the same network as your cluster or from one of your nodes.

Prerequisites

Before you begin, make sure that you have done the following.

- Obtain a `.zip` file with VPN keys from Qumulo Care
- Whitelist the following domains in your firewall rules:
  - `ep.qumulo.com`
  - `missionq.qumulo.com`
  - `monitor.qumulo.com`
- Permit outbound HTTPS traffic on port 443

> **Note**
> If your firewall performs stateful packet inspection (also known as *SPI* or *deep-packet inspection*), you must allow OpenVPN (SSL VPN) explicitly, rather than only open port 443.

To Install VPN Keys from a Networked Machine

1. Copy the `.zip` file from Qumulo Care to a computer on the same network as your cluster, and decompress the file.
2. Install the `qq` CLI on the same computer. For more information, see QQ CLI: Get Started on Qumulo Care.
3. To log in to your cluster, use the `qq` CLI and specify your cluster's IP address. For example:

   ```
   qq --host 203.0.113.0 login
   ```

   > **Note**
   > Your user must have `PRIVILEGE_SUPPORT_WRITE` and `PRIVILEGE_SUPPORT_READ`.

4. To install the VPN keys on your cluster, specify your cluster's IP address and the path to the directory that contains the VPN keys. For example:

```
qq --host 203.0.113.0 install_vpn_keys /my/path
```

5. To verify that the VPN keys installed correctly, use the `get_vpn_keys` command. For example:

```
qq --host 203.0.113.0 get_vpn_keys
```

6. Remove any local copies of the VPN key files.

To Install VPN Keys from a Node

> **Note**
> On macOS and Linux, you can use the `scp` and `ssh` tools. On Windows Server 2022, Windows Server 2019, and Windows 10 (build 1809 and higher), we recommend installing OpenSSH.

1. Copy the `.zip` file from Qumulo Care to a computer on the same network as your cluster, and decompress the file.
2. To copy the VPN key files to one of your nodes, use the `scp` command. For example:

```
scp /my-path/* admin@203.0.113.0:~/
```

3. To connect to the node to which you copied the VPN key files, use the `ssh` command. For example:

```
ssh admin@203.0.113.0
```

   The `qq` CLI is available to the admin user. For example:

```
qq version
```

4. To install the VPN keys on your cluster, specify the path to the directory that contains the VPN keys. For example:

```
sudo qq install_vpn_keys /my/path/
```

5. To verify that the VPN keys installed correctly, use the `get_vpn_keys` command. For example:

   ```
   sudo qq get_vpn_keys
   ```

To Register Cluster with Cloud-Based Monitoring

1. To retrieve your cluster ID, use the `node_state_get` command.
2. Send the output of the command to Qumulo Care.
3. Use the Web UI to enable Qumulo Care Remote Support.
4. Notify Qumulo Care when this process is complete.

   Qumulo Care verifies your VPN functionality and then adds your cluster to Cloud-Based Monitoring.

Configuring SAML Single Sign-On (SSO) for Your Qumulo Cluster

This section explains how to integrate your Qumulo cluster with your organization's single sign-on (SSO) service by configuring Security Assertion Markup Language (SAML) 2.0 for Qumulo Core 5.2.5.1 (and higher).

For more information about the SAML standard for exchanging authentication information, see SAML 2.0.

Prerequisites

Before you begin, make sure that you have done the following.

- Join your cluster to an Active Directory (AD) domain. For more information, see Join Your Qumulo Cluster to Active Directory on Qumulo Care.

  > **Note**
  > Qumulo Core supports SAML authentication only for AD users.

- To allow the cluster to find group memberships for SAML-authenticated users, configure the Base DN in your AD configuration, even if you don't use POSIX attributes.
- Ensure that your SAML Identity Provider (IdP) is linked to the same AD. An *identity provider* (such as Azure AD, Duo, or Okta) is a system that authenticates users (for example, by using passwords and additional factors).

  Typically, an IT department manages an IdP centrally and the IdP is linked with AD. Before you can enable SSO, your IT department must register a new Service Provider (SP) in your IdP. A *service provider* is the server which users access, in this case a Qumulo cluster.

  > **Note**
  > You can use trusts, as long as the Base DN covers all users that might require access to your cluster.

- Configure your IdP to return AD User Principal Names (UPNs, for example `alice@example.com`) or an email address as a NameID. A *NameID* is an identifier for an authenticated user. Typically, a NameID uses the format of an email address.

To Configure SAML SSO for Your Qumulo Cluster

This process requires coordination between the cluster administrator and SSO administrator.

1. The cluster administrator contacts the SSO administrator and asks the SSO administrator to create a SAML integration for the Qumulo cluster.
2. The SSO administrator creates a SAML integration with your organization's SSO identity provider (page 9) (IdP).

   a. The SSO administrator uses the cluster's fully qualified domain name (FQDN) format for the service provider (page 9) (SP) endpoint (also known as the *assertion consumer service URL*), in the following format:

   ```
   https://<my-cluster>.<my-org>.com/saml
   ```

   > **Note**
   > Because the user's browser performs DNS resolution (for example, in a VPN-only scenario), it isn't necessary for an external DNS server to be able to resolve the cluster's FQDN.

   b. If prompted, the SSO administrator enters the HTTP POST binding for the SP endpoint. Typically, this binding is specified by default.

   c. If prompted for **SP Entity ID** (alternatively named **Application Identifier** or **Audience**), the SSO administrator enters `https://<my-cluster>.<my-org>.com/saml`.

   d. If **SAML Signing** (depending on the SSO service, this option is named differently) configuration is available, the SSO administrator sets it to **Sign SAML response and assertion**.

   > **Note**
   > Qumulo Core requires that IdP sign both the assertion and the entire SAML response.

3. After creating the SAML integration, the SSO administrator provides the following information to the cluster administrator.

   - The certificate (public key) of the identity provider, in a `.pem` file.

     This certificate lets the cluster verify the authenticity of the messages from the IdP.

   - The IdP SSO URL—to which the Qumulo cluster can send authentication requests—in the following format:

     ```
     https://<my-org>.<sso-provider>.com/foo
     ```

- The IdP issuer or `EntityId`.

  > **Note**
  > Don't confuse `EntityId` with SP Entity ID.

  For example:

  ```
  http://www.<sso-provider>.com/abc12de34fgAB5CDh6i7
  ```

- The FQDN of the cluster, in the following format:

  ```
  <qumulo-cluster>.<my-org>.com
  ```

4. To configure and enable SAML login to the Qumulo cluster, the cluster administrator runs the `qq saml_modify_settings` command. For example:

```
qq saml_modify_settings --enable \
  --idp-certificate-file ~/certificate.pem \
  --cluster-dns-name <qumulo-cluster>.<my-org>.com \
  --idp-entity-id http://www.<sso-provider>.com/abc12de34fgAB5CDh6i7 \
  --idp-sso-url https://<my-org>.<sso-provider>.com/abc12de34fgAB5CDh6i7/saml
\
```

> **Note**
>
> - To view the current SAML configuration, the cluster administrator can use the `qq saml_get_settings` command.
> - To allow specific changes (for example, correct a typo, update a DNS name or an expired certificate, or temporarily disable SAML SSO without losing any of the other settings), the cluster administrator can use the `qq saml_modify_settings` command to change individual SAML settings independently.
> - For first-time SAML configurations, the cluster administrator must provide all of the required settings.
> - Aside from a basic check of the IdP certificate, Qumulo Core doesn't verify the configuration parameters. It is the cluster administrator's responsibility to ensure that IdP-initiated SAML login works correctly. (This login type initiates when the user clicks **Continue to SSO login** in the Web UI or selects the Qumulo cluster on the SSO portal.)

Supported SAML SSO Workflows

Qumulo Core supports three SAML SSO workflows:

- Standard SAML workflows that the IdP (page 9) or SP (page 9) initiates
- A workflow that the `qq` CLI initiates

> **Note**
>
> - Members of the built-in Administrators role always have access to the Web UI.
> - To allow other users to access the Web UI, you must assign the built-in Observers role to the users.
> - Depending on policy, additional verification might be necessary for users. For example, the SSO administrator can enforce mandatory two-factor authentication (2FA) for certain clusters.
> - If the user accesses the Web UI by connecting to a node physically, the login page doesn't show doesn't show **Continue to SSO login** on the Web UI login page, even if SSO is configured.

IdP-Initiated SSO Worfklow

1. A user authenticates to her organization's SSO portal and then selects the Qumulo cluster on the SSO portal.
2. The SSO portal redirects the user to the cluster's endpoint.

   If the user has sufficient privileges, the Web UI logs the user in. Otherwise, the Web UI displays an error message.

SP-Initiated SSO Workflow

1. A user navigates to the Qumulo cluster's Web UI endpoint in a browser.
2. If the Qumulo cluster has SAML SSO configured, the user can click **Continue to SSO login** on the Web UI login page.

   The Web UI redirects the user to the configured SSO portal. Because the authentication request uses HTTP-Redirect Binding, the login URL appears.

   ```
   https://<my-org>.<sso-provider>.com/abc12de34fgAB5CDh6i7/saml?SAMLRequest=abcd
   efgh1234567890...
   ```

3. The user clicks the login link and the SSO portal authenticates the user.
4. The SSO portal redirects the user to the cluster's endpoint.

qq-CLI-Initiated SSO Workflow

In Qumulo Core 5.3.0 (and higher), a user can authenticate a `qq` CLI session by using SSO.

1. A user uses the `qq sso_login` CLI command. For example:

   ```
   qq --host 203.0.113.0 sso_login
   ```

   The login URL and a prompt appear. The following is an example URL.

   ```
   https://<my-cluster>.<my-org>.com/saml-login?login-id=12345678-1234-1234-123
   4-123456789012
   ```

   > **Note**
   > The user must complete the following step within 5 minutes, while the `qq` CLI pauses for authentication.

2. When the user opens the login URL in a browser, the URL redirects the user to a configured SSO portal and one of the following two scenarios takes place:

    - If authentication succeeds, the browser shows a message that contains an eight-character verification code and asks the user to return to the CLI session.

      The user copies the verification code and enters it into the waiting prompt of the `sso_login` command.

        - If the verification code is correct, the command recognizes that authentication is complete and shows the authenticated username.
        - If the verification code is incorrect, the user must retry the workflow.

    - If authentication doesn't succeed, the browser displays an error message.

      The user must retry the workflow.

Requiring SSO Authentication for Cluster Management

> ⚠ Important
>
> - If you use the `--require-sso` flag, you can no longer use the `qq login` command with your AD account password. Instead, you must use the `qq sso_login` command (page 0).
> - This setting doesn't restrict access through file protocols such as SMB.
> - Because the FTP protocol sends passwords in plaintext, it is inherently insecure. In addition, many FTP clients don't support Transport Layer Security (TSL) or fall back quietly to the plaintext protocol. For this reason, all Qumulo clusters have FTP disabled by default.

In Qumulo Core 5.3.0 (and higher), you can use the `qq saml_modify_settings` CLI command to require AD users to use SSO authentication for managing your cluster. For example:

```
qq saml_modify_settings --require-sso true
```

When the cluster requires SSO authentication, your cluster rejects password-based authentication from AD users in the Web UI, the `qq` CLI, and the REST API.

Known Issues and Limitations

- Local users (the built-in `admin` user and any additional users) can always use their passwords to authenticate to the Web UI and the `qq` CLI.

> ⚠ Important
> We recommend setting a strong password for the built-in `admin` user and using this account only for emergencies.

- If SSO is required for a Qumulo Core cluster, it isn't possible to log in to the **Interactive API documentation** section of the **APIs & Tools** page in the Web UI.
- Qumulo Core doesn't support:
  - **SAML Single Logout (SLO):** We recommend clicking **Sign out** in the Web UI.
  - **Automatic Configuration from Metadata XML:** You must specify each parameter by using the `qq` CLI.
  - **Returning to Previous Web UI Page:** You can't return to a previous page after re-authenticating (for example, after a timeout).
  - **Azure AD SAML Toolkit:** Currently, due to a configuration deficiency in the toolkit, IdP-initiated SSO isn't operational for Qumulo as a Service. Use the SP-initiated SSO workflow (page 0).

Troubleshooting SAML SSO Authentication

This section explains troubleshooting common and uncommon SAML SSO authentication issues.

Common Issues

Typically, if SAML authentication fails, Qumulo Core's in-browser error message explains the reasons for failure and you can resolve the issue by setting the right configuration by using the `qq saml_modify_settings` command. Examples of this issue type include the following scenarios:

- SAML isn't enabled on the Qumulo cluster.
- There is clock skew between the IdP and the Qumulo cluster (the SSO service sets the clock skew tolerance, typically to 5 minutes).
- The `cluster-dns-name` or `idp-entity-id` on the Qumulo cluster aren't configured correctly.
- A user isn't a member of the Observers role that Qumulo Core requires for granting access to the Web UI.

Uncommon Issues

In more complex cases, the in-browser errors are less informative for security reasons. For example, if you configure an incorrect IdP certificate on your cluster, the **Signature validation failed. SAML Response rejected.** error appears.

Several AD configuration issues can cause a **User not found** error:

- The Qumulo cluster isn't joined to AD.

- The Qumulo cluster is joined to AD that isn't connected to the IdP.
- IdP sends usernames (NameID) in an unusual format.

  To verify that you can use a username, run the `qq auth_find_identity` command. For example:

  ```
  qq auth_find_identity --name MyUsername
  ```

- The Configured Base DN doesn't include all users.

  To find a security identifier (SID), run the `qq auth_find_identity` command. For example:

  ```
  qq auth_find_identity --name MyUsername
  ```

  To verify that a username is discoverable, run the `qq ad_sid_to_account` command. For example:

  ```
  qq ad_sid_to_account --sid S-1-5-32-544
  ```

  If an error occurs, contact your AD administrator and request the correct Base DN. For more information, see Specifying the Base Distinguished Name (Base DN) (page 132).

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 0) and Managing File Access Permissions by Using NFSv4.1 Access Control Lists (ACLs) (page 0).
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	265 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 (limited by protocol)
Characters in File Path Component (File or Directory)	255 (limited by protocol)
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.

Qumulo Compliance Posture

This section lists the attestations that Qumulo has achieved from third parties.

SOC 2 Type II (Safeguarding Customer Data)

Azure Native Qumulo Scalable File Service has achieved the SOC 2 Type II attestation. To receive a copy of our report, fill out and submit the form in the Qumulo Trust Center.

FIPS 140-2 Level 1 (Security Requirements for Cryptographic Modules)

The Qumulo Secure service has been validated against FIPS 140-2 Level 1. To verify our certification, see Qumulo Secure TLS KDF in the Cryptographic Algorithm Validation Program.

Upgrading Qumulo Core

Performing Instant Software Upgrades and Platform Upgrades

This section explains the difference between Instant Software Upgrades and Platform Upgrades. For more information, see Performing Qumulo Core Upgrades by Using the qq CLI on Qumulo Care.

> ⚠ Important
>
> - If you perform multiple upgrades back to back, you might encounter one or more platform upgrades in one of the incremental releases; you must install these upgrades before you continue. Before performing back to back upgrades, contact Qumulo Care for guidance.
> - If you don't see a rolling reboot option for a platform upgrade, refresh the page in your browser.

Understanding the Differences Between Upgrade Modes

For information about which upgrade modes different Qumulo Core releases use, see Qumulo Core Upgrade Mode Reference (page 24).

Instant Software Upgrade

The more common, faster instant software upgrade requires restarting only the container on your nodes and has a downtime of less than 30 seconds.

Because in Qumulo Core 3.3.2 (and higher), the Qumulo file and data protection systems are separate from the host in charge of running the operating system and the services specific to each hardware or cloud platform, and because these services run in a lightweight container (by using Ubuntu-native systemd-nspawn containerization) in the user space, it is possible to move quickly from one version of Qumulo Core to another by loading a new container and pointing the runtime environment at updated software.

Note

- Under certain conditions, an end-to-end instant software upgrade might take a little longer while Qumulo Core performs background tasks. This doesn't impact user experience.
- Instant software upgrades don't impact existing support for the `qq` CLI or REST API commands.
- A direct upgrade to Qumulo Core 3.3.3 *isn't* an instant software upgrade (it only establishes the framework for this functionality). Upgrading from Qumulo Core 3.3.2 to 3.3.3 is the first official, minimally disruptive instant software upgrade. Any subsequent upgrade, regardless of release, is an instant software upgrade unless we specify otherwise.

Platform Upgrade

The infrequent, somewhat slower platform upgrade requires either a *complete reboot* (rebooting all nodes in your cluster at the same time) or—in Qumulo Core 5.0.3 (and higher)—a *rolling reboot* (rebooting the nodes in your cluster one at a time).

In contrast with instant software upgrades, your nodes maintain the Linux operating system, and certain services that Qumulo Core relies on, through most upgrades because the underlying host changes less frequently than the container and the file and data protection systems.

Important

- If you perform multiple upgrades back to back, you might encounter one or more platform upgrades in one of the incremental releases; you must install these upgrades before you continue. Before performing back to back upgrades, contact Qumulo Care for guidance.
- If you don’t see a rolling reboot option for a platform upgrade, refresh the page in your browser.
- Upgrading past a platform upgrade still requires a node reboot, even if you don't install the exact build. For example, if your cloud cluster runs Qumulo Core 4.0.0, installing Qumulo Core 4.1.0 triggers a platform upgrade, because the installation includes all changes contained in Qumulo Core 4.0.6 that is a platform upgrade.

Understanding the Upgrade Phases

Every Qumulo Core upgrade has two phases, *preparation* and *commit*.

1. **Preparation:** Qumulo Core stages a new image in an alternate boot drive partition while the current image continues to run. This phase is responsible only for the background work (unpacking and writing the platform image and upgrade firmware, and so on). When the preparation phase is complete, we continue to the commit phase.
2. **Commit:** Qumulo Core does one of the following:
    - **Instant Software Upgrade:** Stops the existing container and starts a new one.
    - **Platform Upgrade:** Initiates a reboot and selectively upgrades the operating system image.

To determine what phase an upgrade is in, use the `qq upgrade_status` command while your cluster is performing and upgrade. For more information, see Performing Qumulo Core Upgrades by Using the qq CLI on Qumulo Care.

The following is example output from the command.

```
{
  "install_path": "/upgrade/qinstall.qimg",
  "state": "UPGRADE_PREPARING",
  "details": "",
  "error_message": "",
  "error_state": "UPGRADE_ERROR_NO_ERROR",
  "is_blocked": false,
  "blocked_reason": ""
}
```

Qumulo Core Upgrade Mode Reference

This section provides a reference for Qumulo Core upgrade modes from version 3.3.3 onwards.

- An *instant software upgrade* (page 21) requires restarting only the container on your nodes and has a downtime of less than 30 seconds.
- A *platform upgrade* (page 22) requires either a complete reboot (rebooting all nodes in your cluster at the same time) or a rolling reboot (rebooting the nodes in your cluster one at a time).
- A *quarterly upgrade* aggregates all improvements and fixes since the last quarterly upgrade. The version number of a quarterly upgrade ends in `.0`.

> ⚠ Important
> Although the *upgrade types* for on-premises upgrades (page 24) and cloud upgrades (page 26) are most often the same, they do occasionally diverge. For example, for Qumulo Core 5.3.1, a cloud deployment allows an instant upgrades, an on-premises deployment requires a platform upgrade.

On-Premises Upgrades

Version	Upgrade Type
5.3.3.1	Instant
5.3.2	Instant
5.3.1	Platform
5.3.0	Instant, Quarterly
5.2.5.1	Instant
5.2.4	Instant
5.2.3	Instant
5.2.2	Instant
5.2.1	Instant
5.2.0.2	Instant, Quarterly

Version	Upgrade Type
5.1.5	**Platform**
5.1.4.1	Instant
5.1.3	Instant
5.1.2	Instant
5.1.1	**Platform**
5.1.0.1	Instant, Quarterly
5.0.6	Instant
5.0.5	Instant
5.0.4	Instant
5.0.3	Instant
5.0.2	Instant
5.0.1	Instant
5.0.0.1	Instant, Quarterly
4.3.4	Instant
4.3.3	Instant
4.3.2	Instant
4.3.1	Instant
4.3.0	Instant, Quarterly
4.2.6	Instant
4.2.5	Instant
4.2.4	**Platform**
4.2.3	Instant
4.2.2	Instant

Version	Upgrade Type
4.2.1	Platform
4.2.0	Instant, Quarterly
4.1.5	Instant
4.1.4	Instant
4.1.3	Instant
4.1.2	Instant
4.1.1	Instant
4.1.0.1	Instant, Quarterly
4.0.6	Instant
4.0.5	Instant
4.0.4	Instant
4.0.3	Instant
4.0.2	Instant
4.0.1.1	Instant
4.0.0.2	Instant, Quarterly
3.3.5	Instant
3.3.4	Instant
3.3.3	Instant

Cloud Upgrades

Version	Upgrade Type
5.3.3.1	Instant
5.3.2	Instant

Version	Upgrade Type
5.3.1	Instant
5.3.0	Instant
5.2.5.1	Instant
5.2.4	Instant
5.2.3	Instant
5.2.2	Instant
5.2.1	Instant
5.2.0.2	Instant, Quarterly
5.1.5	Platform
5.1.4.1	Instant
5.1.3	Instant
5.1.2	Instant
5.1.1	Platform
5.1.0.1	Instant, Quarterly
5.0.6	Instant
5.0.5	Instant
5.0.4	Instant
5.0.3	Instant
5.0.2	Instant
5.0.1	Instant
5.0.0.1	Instant, Quarterly
4.3.4	Instant
4.3.3	Instant

Version	Upgrade Type
4.3.2	Instant
4.3.1	Instant
4.3.0	Instant, Quarterly
4.2.6	Instant
4.2.5	Instant
4.2.4	Platform
4.2.3	Instant
4.2.2	Instant
4.2.1	Platform
4.2.0	Instant, Quarterly
4.1.5	Instant
4.1.4	Instant
4.1.3	Instant
4.1.2	Instant
4.1.1	Instant
4.1.0.1	Instant, Quarterly
4.0.6	Platform
4.0.5	Instant
4.0.4	Instant
4.0.3	Instant
4.0.2	Instant
4.0.1.1	Instant
4.0.0.2	Instant, Quarterly

Version	Upgrade Type
3.3.5	Instant
3.3.4	Instant
3.3.3	Instant

Configuring Networking for Qumulo Core

Required Networking Ports for Qumulo Core

This section explains which inbound and outbound networking ports Qumulo Core requires.

> **Note**
> Active Directory authentication services require their own network port range. For an authoritative list, see Active Directory and Active Directory Domain Service Port Requirements

Networking Ports for Inbound Connections

Port	Protocols	Use
21	TCP	FTP
22	TCP	SSH
80	TCP	HTTP (Web UI)
111	TCP UDP	`rpcbind` or `portmapper` for NFSv3
443	TCP	HTTPS (Web UI)
445	TCP	SMB
2049	TCP UDP	NFS or MOUNT **Note** Qumulo Core supports UDP for the MOUNT protocol for older clients. However, any NFS clients&emdash;that specify the TCP mount option or transfer data over NFS after mounting&em-dash;don't use UDP.
3712	TCP	Replication
8000	TCP	REST API

Port	Protocols	Use
32768-60999	TCP	FTP Passive Mode

Networking Ports for Outbound Connections

> **Note**
>
> - In the following table, uses marked with an asterisk (*) are default configurations. You can reconfigure these ports.
> - For cluster formation and inter-node communication Qumulo Core requires the following:
>   - **Hardware Platforms:** Unblocked IPv6 traffic in the local subnet&emdash;for more information, see Configuring IPv6 in Qumulo Core.
>   - **Cloud Platforms:** Unblocked IPv4 traffic in the local subnet

Port	Protocols	Use
53	UDP	DNS
88	TCP	Kerberos
111	TCP	`rpcbind` or `portmapper` for NSM and NLM **Note** Depending on the client `portmapper` configuration, Qumulo Core might require additional ports.
123	UDP	Synchronization of product and network time, for authentication and time-stamping of artifacts such as audit logs, by using the Network Time Protocol (NTP).
135	TCP	DCERPC or Netlogon (Domain Controller Binding)
389 636	TCP	LDAP to Active Directory or to a standalone LDAP server*
443	TCP	Qumulo Shift for Amazon S3*

Port	Protocols	Use
514	TCP	Audit with `Rsyslog` *
3712	TCP	Replication*

Connecting Qumulo Core to External Services

Using Qumulo Core Access Tokens

This section explains how to create and use access tokens—by using the Qumulo REST API, Python SDK, and `qq` CLI—to authenticate external services to Qumulo Core.

In Qumulo Core 5.3.0 (and higher), *access tokens* let a user authenticate to the Qumulo REST API without having to complete repetitive login procedures. Access tokens provide an alternative to session-based authentication that the `qq login` command and the Web UI use.

Unlike *session bearer tokens* (that have a short expiration time and require a password to refresh, for example for authentication by using the `qq login` command), access tokens are long-lived. Access tokens support authentication for services, long-lived automation processes, and programmatic REST API access that doesn't require user input.

> ⚠ Important
>
> - An attacker can use an access token to authenticate as the token's user to Qumulo Core REST API (through HTTP, the Python SDK, or the `qq` CLI) and gain all of the user's privileges. Treat access tokens, and the bearer tokens they generate, like passwords. Store your tokens securely, rotate your tokens often, and create a token revocation policy for your organization.
> - Because a token allows indefinite authentication to the associated user's account, we strongly recommend against creating tokens for individual Qumulo Core REST API users. For more information, see Best Practices for Using Access Tokens (page 0).

Prerequisites

- `PRIVILEGE_ACCESS_TOKEN_WRITE` is required for creating, disabling, and deleting access tokens for all users in the system.
- `PRIVILEGE_ACCESS_TOKEN_READ` is required for listing access tokens.

Creating and Using Access Tokens

`PRIVILEGE_ACCESS_TOKEN_WRITE` is required for creating, disabling, and deleting access tokens for all users in the system. This section explains how to create access tokens without or with an expiration time by using the `qq` CLI.

To Create an Access Token without an Expiration Time

Use the `auth_create_access_token` command and specify the user. For example:

```
$ qq auth_create_access_token jane
```

You can:

- Specify the user as a name
- Qualify the user by using a domain prefix, for example:
  - `ad:jane`
  - `AD\jane`
  - `local:jane`
- Specify ID types, for example:
  - `auth_id:1234`
  - `SID:S-1-1-0`

> **Note**
>
> - Although you can create groups for users, you can't create access tokens for groups.
> - To use an access token in the `qq` CLI, you must use the `--file` flag—to specify a path for saving your credentials file in a format that the `qq` CLI can use—when you create the access token.

The `auth_create_access_token` command returns a JSON response that contains the bearer token body and the access token ID, which you can use to manage the access token.

```
{
  "bearer_token": "access-v1:abAcde...==",
  "id": "12345678901234567890123"
}
```

Important

- As soon as you receive your bearer token, record it in a safe place. If you misplace the bearer token, you can't retrieve it at a later time. You must create a new access token.
- Any user can have a maximum of two access tokens. If a user already has two access tokens, creating new tokens fails until you remove at least one token from the user. We strongly recommend creating a single access token for each user and using the second access token to perform secret rotation.
- Treat access tokens, and the bearer tokens they generate, like passwords. Store your tokens securely, rotate your tokens often, and create a token revocation policy for your organization.
- To decrease the risk of giving an attacker full administrative access—including access to cluster data—avoid generating tokens for accounts with administrative privileges.

To Create an Access Token with an Expiration Time

In Qumulo Core 5.3.2 (and higher), you can use the `auth_create_access_token --expiration-time` command and specify the expiration time. You can specify the expiration time in different formats. For example:

```
$ qq auth_create_access_token jane --expiration-time 'Jan 01 2023'
```

```
$ qq auth_create_access_token jane --expiration-time '01/01/2023 00:00'
```

When an access token's expiration time elapses, it isn't possible to use the token for authentication. Any attempt to use the token results in an authentication error. To continue the authentication process, you must either create a new access token (page 0) or update the expiration time for your existing token (page 0).

Note
The `--expiration-time` flag interprets arguments as timestamps in the UTC time zone.

Using Bearer Tokens for Authentication

A Qumulo Core access token returns a *bearer token* (page 34), an item in the `Authorization` HTTP header which acts as the authentication mechanism for the Qumulo Core REST API.

REST API

When you use the Qumulo REST API, add the bearer token to the `Authorization` HTTP header. For example:

```
Authorization: Bearer access-v1:abAcde...==
```

You can also add the bearer token to a `curl` command. For example:

```
$ curl https://<qumulo-cluster>:8000/v1/session/who-am-i -H 'Authorization: Bearer access-v1:abAcde...=='
```

Python SDK

When you use the Qumulo Python SDK, add the bearer token to a `RestClient` object. For example:

```
from qumulo.rest_client import RestClient
from qumulo.lib.auth import Credentials
client = RestClient('<qumulo-cluster>', 8000, Credentials('access-v1:abAcde...=='))
```

For more information, see the Qumulo Core Python SDK.

qq CLI

To use an access token in the `qq` CLI, you must use the `--file` flag—to specify a path for saving your credentials file in a format that the `qq` CLI can use—when you create the access token. For example:

```
$ qq auth_create_access_token jane --file ./qumulo_credentials
```

To use the credentials file, specify its location by using the `--credentials-store` flag. For example:

```
$ qq --credentials-store ./qumulo_credentials who_am_i
```

Getting Metadata for Access Tokens

`PRIVILEGE_ACCESS_TOKEN_READ` is required for listing access tokens. This section explains how to get metadata for a specific access token or all access tokens by using the `qq` CLI.

To Get Metadata for a Specific Access Token

Use the `auth_get_access_token` command and specify the access token ID. For example:

```
$ qq auth_get_access_token 1234567890123456789012
```

This command returns a JSON object that lists:

- The access token ID
- The user that the access token represents
- The access token's creator
- The access token's creation time
- The access token's expiration time
- Whether the access token is enabled

For example:


```
{
  "creation_time": "2022-12-06T01:14:39.56621474Z",
  "creator": {
    "auth_id": "500",
    "domain": "LOCAL",
    "gid": null,
    "name": "admin",
    "sid": "S-1-1-12-12345678-1234567890-1234567890-500",
    "uid": null
  },
  "enabled": true,
  "expiration_time": "2023-01-01T00:00:00Z",
  "id": "12345678901234567890123",
  "user": {
    "auth_id": "1002",
    "domain": "LOCAL",
    "gid": null,
    "name": "svc",
    "sid": "S-1-1-12-12345678-1234567890-1234567890-1002",
    "uid": null
  }
}
```


To Get Metadata for All Access Tokens

Use the `qq auth_list_access_tokens` command.

> ⚠ Important
> Listing access tokens *doesn't* return the bearer token required for authentication. If you misplace the bearer token, you can't retrieve it at a later time. You must create a new access token.

The `auth_list_access_tokens` command returns:

- The access token ID
- The user that the access token represents
- The access token's creator
- The access token's creation time
- The access token's expiration time
- Whether the access token is enabled

For example:

```
id                      user   creator  creation time                  expiration t
ime       enabled
======================  =====  =======  ==============================
====================  =======
1234567890123456789012  svc    admin    2022-10-27T15:18:09.725513764
Z                        True
0987654321098765432109  svc    admin    2022-10-27T15:18:24.997572918Z  2023-01-01T0
0:00:00Z  False
```

To filter the command's output by user, use the `--user` flag and use the same format for the name as for the `auth_create_access_token` (page 34) command.

Modifying the Expiration Time for an Access Token

`PRIVILEGE_ACCESS_TOKEN_WRITE` is required for creating, disabling, and deleting access tokens for all users in the system. This section explains how to modify access tokens by using the `qq` CLI.

Use the `auth_modify_access_token` command and specify the access token ID and the expiration time. For example:

```
$ qq auth_modify_access_token 1234567890123456789012 --expiration-time 'Jan 01 2023'
```

When an access token's expiration time elapses, it isn't possible to use the token for authentication. Any attempt to use the token results in an authentication error. To continue the authentication process, you must either create a new access token (page 0) or update the expiration time for your existing token (page 0).

> **Note**
> The `--expiration-time` flag interprets arguments as timestamps in the UTC time zone.

Disabling an Access Token

To help you check your system's security posture, Qumulo Core lets you disable an access token without deleting it. This is a good way to check for dependencies on the access token before you delete the token permanently.

`PRIVILEGE_ACCESS_TOKEN_WRITE` is required for creating, disabling, and deleting access tokens for all users in the system. This section explains how to disable an access token by using the `qq` CLI.

> **Important**
> After you disable an access token, you can no longer use any bearer tokens associated with the access token to authenticate to Qumulo Core.

To disable an access token, use the `auth_modify_access_token` command, specify the access token ID, and use the `-d` flag. For example:

```
$ qq auth_modify_access_token 123456789012345678 -d
```

To enable an access token, use the `auth_modify_access_token` command, specify the access token ID, and use the `-e` flag. For example:

```
$ qq auth_modify_access_token 123456789012345678 -e
```

Deleting Access Tokens

`PRIVILEGE_ACCESS_TOKEN_WRITE` is required for creating, disabling, and deleting access tokens for all users in the system. This section explains how to delete an access token by using the `qq` CLI.

> **Important**
> After you delete an access token, you can no longer use any bearer tokens associated with the access token to authenticate to Qumulo Core.

To delete an access token, use the `auth_delete_access_token` command and specify the access token ID. For example:

```
$ qq auth_delete_access_token 1234567890123456789012
```

Best Practices for Using Qumulo Core Access Tokens

This section lists the best practices for limiting the exposure to lost credentials and working with Qumulo Core access tokens securely.

Avoiding Creation of Tokens for Administrative Accounts

An attacker can use an access token to authenticate as the token's user to Qumulo Core REST API (through HTTP, the Python SDK, or the `qq` CLI) and gain all of the user's privileges. To decrease the risk of giving an attacker full administrative access—including access to cluster data—avoid generating tokens for accounts with administrative privileges.

Generating Tokens for Service Accounts

When you connect external services to the Qumulo Core REST API, we recommend creating a service account with limited privileges for each individual service and generating an access token for each service account.

To Create a New Service Account

1. Log in to Qumulo Core.
2. Create a service account.

   a. Click **Cluster** > **Local Users & Groups**.

   b. In the **Users** section, click **Create**.

   c. In the **Create user** dialog box, enter a **User name** and **Password**, re-enter the password, and then click **Create**.
3. Create a role with privileges.

   a. Click **Cluster** > **Role Management**.

   b. In the **Role Management** section, click **Create Role**.

   c. On the **Create Role** page, enter a **Name** and **Description**, click the **Privileges** for the user, and then click **Save**.
4. Assign the service user to the role.

   a. On the **Role Management** page, find the name of the role you created and then click **Add Member**.

   b. In the **Add Member to <MyRoleName>** dialog box, for **Trustee**, enter the name of the user you created and then click **Yes, Add Member**.

5. Create access tokens (page 0) for your service account.

Rotating Access Tokens

We strongly recommend rotating access tokens for a service account at a regular interval.

To Rotate an Access Token for a Service Account

1. To ensure that there is only one access token for each service account, use the `qq auth_list_access_tokens` command.

   If multiple access tokens exist, delete any unused access tokens.

2. To create a new access token for the service account, use the `qq auth_create_access_token` command.
3. In the credential store of your service, replace the old access token with the new one.
4. Test that your service account can access the Qumulo Core REST API.
5. Confirm that there is nothing else relying on the old access token by disabling it first. If this causes any disruptions then you can re-enable it while you resolve the issue.
6. To delete the old access token, use the `qq auth_delete_access_token` command.

Working with the Qumulo Core Web UI

Setting the Web UI Login Banner

This section explains how to 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

This section explains how to set an inactivity timeout for the Qumulo Core Web UI.

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
```

Working with the qq CLI

Enabling Autocomplete for the qq CLI

This section explains how to enable automatic command completion for the qq CLI and for command aliases.

The `qq` CLI supports Python argparse completion that helps you use the CLI more effectively. This section explains how to enable automatic command completion for the `qq` CLI and for command aliases.

> ⚠ Important
> The following procedures apply to running the `qq` CLI on Linux, macOS, and Windows Subsystem for Linux. Don't run these commands on Qumulo nodes

To Enable Autocomplete for the qq CLI

1. Install the `argcomplete` Python package.

   ```
   pip install argcomplete
   ```

   > ℹ Note
   > Qumulo Core supports `argcomplete` 2.0.0 and higher.

2. Activate the `argcomplete` package.

   ```
   sudo activate-global-python-argcomplete
   ```

3. Search for any conflicting `qq` entries.

   ```
   complete | grep qq
   ```

   If conflicting entries exist, remove them by specifying the entry name or path. For example:

   ```
   complete -r /my/path
   ```

4. To enable autocompletion for the `qq` CLI, add the following line to the end of your shell profile (`.bashrc`, `.bash_profile`, and so on).

```
eval "$(register-python-argcomplete qq)"
```

5. Reload your shell profile.

```
source ~/.bashrc
```

You can now use the **Tab** key to autocomplete `qq` CLI commands. The `qq` CLI supports autocomplete for all CLI arguments and Qumulo REST API command arguments.

Enabling Autocomplete for qq CLI Command Aliases

To eliminate the need to repeatedly enter `qq` CLI flags (such as `--host` or `--credentials-store`), for example when dealing with multiple Qumulo clusters, you can add aliases for `qq` CLI commands to your shell profile. In the following example, we alias a complex `qq` CLI command to the simple alias `qqcreds`.

```
alias qqcreds='qq --host my.qumulo.com --credentials-store ~/.my_creds'
```

When you reload your profile, you can append a parameter to the complex command by appending it to the alias. For example:

```
qqcreds my_credentials
```

To ensure that your `argcomplete` configuration works with `qq` CLI command aliases, you must perform additional configuration and add a third-party helper script to your system.

> ⚠ Important
> Before you begin, review the source code of the complete-alias helper script. Qumulo does not contribute to, maintain, or take responsibility for this script.

To Enable Autocomplete for qq CLI Command Aliases

1. Add a `qq` CLI command alias and the `COMPAL_AUTO_UNMASK` configuration parameter to your shell profile (`.bashrc`, `.bash_profile`, and so on). For example:

```
#qq CLI Autocomplete
eval "$(register-python-argcomplete qq)"
COMPAL_AUTO_UNMASK=1
source ~/.bash_completion.d/complete_alias
```

> ☑ Tip
> Don't reload your shell profile yet.

2. Create a directory for the `complete-alias` daemon and download the script to it.

```
mkdir ~/.bash_completion.d
curl https://raw.githubusercontent.com/cykerway/complete-alias/master/complet
e_alias \
  > ~/.bash_completion.d/complete_alias
```

3. Add your alias to the `complete_alias` file.

```
echo "complete -F _complete_alias qqcreds" >> ~/.bash_completion.d/complete_al
ias
```

4. Search for any conflicting `complete` entries.

```
complete | grep complete
```

   If conflicting entries exist, remove them by specifying the entry name or path. For example:

```
complete -r /my/path
```

5. Reload your shell profile.

```
source ~/.bashrc
```

You can now use the **Tab** key to autocomplete `qq` CLI command aliases.

Protecting Your Data

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

This section explains how to 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.

Managing Snapshots

This section explains how to use the Qumulo Core Web UI to view and manage your saved snapshots in Qumulo Core 4.3.3 (and higher).

The **Snapshots** page lets you work with a 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

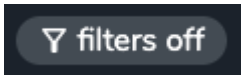
The **Snapshots** page lets you navigate a large number of snapshots.

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).

Moving Your Data

Using Qumulo Shift-To to Copy Objects to Amazon S3

This section explains how to use Shift-To to copy objects from a directory in a Qumulo cluster to a folder in an Amazon Simple Storage Service (Amazon S3) bucket and how to manage Shift relationships.

For more information about copying objects from S3 to Qumulo, see Using Qumulo Shift-From for Amazon S3 to Copy Objects (page 63).

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 by using the specified credentials, and contains downloadable objects.
2. Creates the Shift-To relationship.
3. Starts a job by 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

Entity in the Qumulo File System	Entity in an Amazon S3 Bucket
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 to 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)

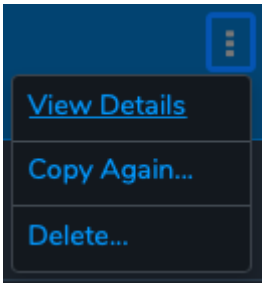
   f. Your AWS **Access Key ID** and **Secret Access Key**.
5. (Optional) For additional configuration, click **Advanced S3 Server Settings**.
6. Click **Create Copy**.
7. In the **Create Copy to S3?** dialog box, review the Shift relationship and then click **Yes, Create**.

   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**.

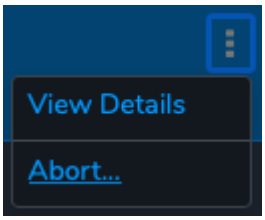


The **Copy to/from S3 Details** page displays the following information:

- **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**.

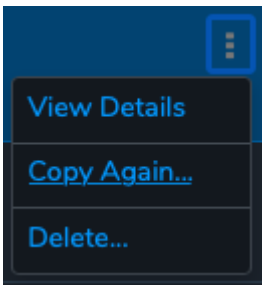


4. In the **Abort copy from?** dialog box, review the Shift relationship and then click **Yes, Abort**.

   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**.



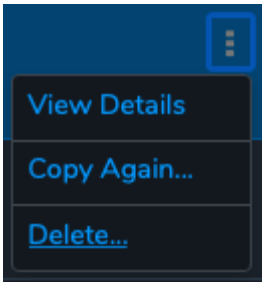
4. In the **Copy again?** dialog box, review the Shift relationship and then click **Yes, Copy Again**.

   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 to 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 to 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 by 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 by 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 by 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 to use 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 and how to manage Shift relationships.

For more information about copying objects from Qumulo to S3, see Using Qumulo Shift-To for Amazon S3 to Copy Objects (page 51) on Qumulo Care.

> **Note**
> 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 51).)

```
{
  "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 by using the specified credentials, and contains downloadable objects.
2. Creates the Shift-From relationship.
3. Starts a job by 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 to 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**.

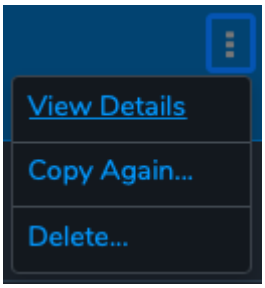
   The copy job begins and Qumulo Core estimates the work to be performed. When the estimation is complete, the Web UI displays a progress bar with a percentage for a relationship on the **Replication Relationships** page. The page also displays the estimated total work, the remaining bytes and files, and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

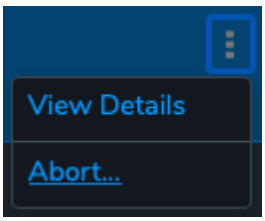


   The **Copy to/from S3 Details** page displays the following information:

   - **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**.

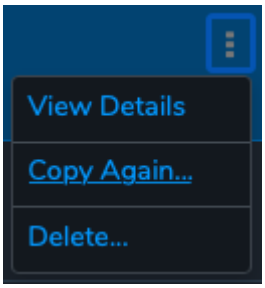


4. In the **Abort copy from?** dialog box, review the Shift relationship and then click **Yes, Abort**.

   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**.

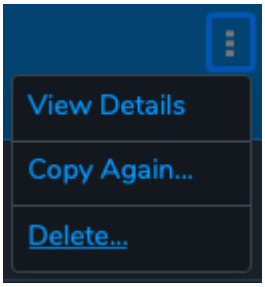


4. In the **Copy again?** dialog box, review the Shift relationship and then click **Yes, Copy Again**.

   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 to 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 to 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 by 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 by 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.

Working with File System Protocols

Enabling and Using NFSv4.1 on a Qumulo Cluster

This section explains how to configure your cluster for a supported export configuration and enable or disable NFSv4.1 on your cluster.

For more information about NFSv4.1 and file access permissions, see Managing File Access Permissions by Using NFSv4.1 Access Control Lists (ACLs) (page 83).

> ⚠ Important
>
> - Currently, Qumulo Core 4.3.0 (and higher) 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 by 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
`/read_only/home`	`/home`	Yes

Export Name	File System Path	Read-Only
`/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 through 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 by using NFSv4.1 (if it is enabled) or by 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 through 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 130)
- 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 by using access control lists (ACLs)
- File locking (for example, by using the `fcntl` command)
- Snapshots through NFSv4.1 (Qumulo Core 5.2.4 and higher)
- Quotas through NFSv4.1 (Qumulo Core 5.2.5.1 and higher)

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 to 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`, `AUTH_KRB5P`, or `AUTH_KRB5I` 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 76).

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 83), 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 83), 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 in Qumulo Core (page 130).

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 in Qumulo Core (page 130).
`<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 84). 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`
`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`
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`

Permissions	NFSv4.1 Command	Qumulo Command
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'`

Watching for File Attribute and Directory Changes by Using SMB2 CHANGE_NOTIFY

This section lists the completion filters that an SMB client can request and the corresponding actions that Qumulo Core returns for a matched change.

Qumulo Core can watch for changes in file attributes and directory entries with a combination of SMB2 `CHANGE_NOTIFY` filters. Depending on the requested filter—and activity in the filesystem—an SMB client or an application remains current by receiving a variety of notifications.

Commonly, these requests help limit the amount of traffic required to keep a current cache of entries for an open directory. The requests also help operating system applications such as Windows Explorer and macOS Finder update automatically when changes take place. It is also possible to make requests programmatically. For more information about language bindings, see the Windows Protocol documentation, such as ReadDirectoryChangesW function (winbase.h) for Win32 and FileSystemWatcher Class for .NET.

> **Note**
>
> - Certain events, such as `rename` and `delete` trigger multiple `NOTIFY_CHANGE` events.
> - For certain events, such as `setattr` and `write`, Qumulo Core aggregates changes into a single `MODIFIED` event.
> - Currently, Qumulo Core doesn't support watching `STREAM` attributes or the `WATCH_TREE` flag.

Completion Filter Types

Each request uses a *completion filter* to specify the events to watch for. When events occur, the system batches them into a `NOTIFY` response that contains a list of `FILE_ACTION` items, each tagged with the names of changed entries. As long as the handle for the watched directory remains open, events queue up on the server, so that no events are lost between `NOTIFY` requests.

- **Watching for Name Changes:** A name change can include four event types.
  - Renaming
  - Deleting
  - Moving into watched directory
  - Moving out of watched directory

The returned action specifies to your application whether an entry has been added, renamed, or removed.

- **Watching for Metadata Changes:** A metadata change can include six supported attribute types.
    - File attributes
    - File size
    - Last-write time
    - Last-access time
    - Creation time
    - Security (the permissions or access control list for the file or directory)

> **Note**
> Qumulo doesn't support mutating extended attributes (EA). If only the `FILE_NOTIFY_CHANGE_EA` filter is requested, no events propagate.

Completion Filters and Corresponding Actions

The following table show the requested completion filters (grouped by the number of inode reads required to support them), the changes they watch for, and the actions corresponding to them.

Completion Filters	Actions	Description
The following filters watch for name changes (`readdir-without-attrs`). • `FILE_NOTIFY_CHANGE_DIR_NAME` • `FILE_NOTIFY_CHANGE_FILE_NAME`	• `FILE_ACTION_ADDED` • `FILE_ACTION_MODIFIED` • `FILE_ACTION_REMOVED` • `FILE_ACTION_RENAMED_NEW_NAME` • `FILE_ACTION_RENAMED_OLD_NAME`	When Qumulo Core watches names, it notifies the client when there is an added, removed, or renamed file or directory in the watched directory. • A `delete` event sends both `REMOVED` and `MODIFIED` notifications. • A `rename` event sends separate, consecutive events for `OLD` and `NEW` names, for example: `[REMOVED, file_old_name], [ADDED, file_new_name]`

Completion Filters	Actions	Description
The following filters watch for metadata changes (`readdir-with-attrs`). • `FILE_NOTIFY_CHANGE_ATTRIBUTES` • `FILE_NOTIFY_CHANGE_CREATION` • `FILE_NOTIFY_CHANGE_SECURITY` • `FILE_NOTIFY_CHANGE_SIZE` • `FILE_NOTIFY_CHANGE_LAST_ACCESS` • `FILE_NOTIFY_CHANGE_LAST_WRITE` **Note** Qumulo doesn't support mutating extended attributes (EA). If only the `FILE_NOTIFY_CHANGE_EA` filter is requested, no events propagate.	`FILE_ACTION_MODIFIED`	When one of the watched attributes changes for an entry of the watched directory and the filter is requested, the client receives a `MODIFIED` event. **Note** In Microsoft terminology, attributes are *flags*. For more information, see File Attributes in the Open Specification documentation.

Completion Filters	Actions	Description

The following filters watch for alternative data stream (ADS) changes (`readdir-attrs-and-stream-names`).
• `FILE_NOTIFY_CHANGE_STREAM_NAME`
• `FILE_NOTIFY_CHANGE_STREAM_SIZE`
• `FILE_NOTIFY_CHANGE_STREAM_WRITE`
Note
Qumulo Core doesn't support watching ADS changes. The following explanation is only informational.
For more information, see Known CHANGE_NOTIFY Limitations for Qumulo Core (page 0).
	• `FILE_ACTION_ADDED_STREAM`
	• `FILE_ACTION_REMOVED_STREAM`
	• `FILE_ACTION_MODIFIED_STREAM`
		Consider the following example command.
		`echo "data" > watched_dir/file0:stream`
		This command generates the following event.
		`[ADDED_STREAM, file0:stream]`
		When a name change takes place, Qumulo Core returns the `STATUS_ENUM_DIR` message that indicates that the client should perform its own directory read.

Known CHANGE_NOTIFY Limitations for Qumulo Core

Qumulo Core doesn't support the following workflows.

Important
You must ensure that your application can handle the `STATUS_ENUM_DIR` response status that indicates to the SMB client that it needs to re-enumerate a directory manually. Depending on the library you use, this response might propagate as an error or an empty response.

- A `NOTIFY_CHANGE_STREAM` ADS completion filter is requested.
- The `SMB2_WATCH_TREE` flag is requested. Rather than watch a directory tree recursively, Qumulo Core handles the `SMB2_WATCH_TREE` flag the same way it handles the `CHANGE_STREAM` filters.

- The watched directory contains more than 5,000 entries.
- The response buffer size is exceeded. Most third-party libraries limit this size to 64 Kb. At the maximum SMB file name length of 255 UTF-16 characters, this limit corresponds to roughly 200 simultaneous file renames.
- The system reaches the heap usage quota for `CHANGE_NOTIFY`. In the worst scenario, this can correspond to more than 500 unique handles across all clients connected to a single Qumulo node.

Using the S3 API with Qumulo Core

Getting Started with the S3 API in Qumulo Core

This section explains how to get started using the S3 API with Qumulo Core.

Related Topics

- Creating and Managing S3 Access Keys (page 0)
- Creating and Managing S3 Buckets (page 0)
- Supported S3 Functionality and Known Limits for Qumulo Core (page 0)

The S3 API in Qumulo Core provides a native way for clients and applications to interact with the Qumulo file system using the Amazon S3 API.

This guide is intended for system administrators who are responsible for configuring and administering storage, as well as colleagues in your organization who wish to use the S3 API with Qumulo Core. It covers:

- How to configure and enable the S3 API server on a Qumulo cluster
- How to create S3 API access keys for use with a Qumulo cluster
- How to create S3 buckets and upload and retrieve objects

Prerequisites

To follow this guide, you should have:

- A basic understanding of Amazon S3
- The AWS Command Line Interface installed
- The Qumulo command line interface installed
- Network connectivity to one or more nodes in your Qumulo cluster

Step 1: Configuring HTTPS

The Qumulo Core S3 API accepts only HTTPS requests by default. For the S3 API service to run with HTTPS, you must install a valid SSL certificate on your Qumulo cluster. Every Qumulo cluster is preconfigured with a self-signed SSL certificate. However, because certain applications don't accept the default certificate, we recommend installing your own.

For information about installing a signed SSL certificate, see SSL: Install a Signed SSL Certificate.

Enabling and Disabling Plaintext HTTP Connections

> ⚠ Important
> If you configure the S3 API service to accept only plaintext HTTP connections, all requests made through the S3 API are not encrypted.

- To enable HTTP connections, use the `qq s3_modify_settings --insecure` command.
- To revert to encrypted HTTPS requests, use the `qq s3_modify_settings --secure` command.

Step 2: Enabling the S3 API on a Qumulo Cluster

The S3 API is not enabled by default; it must be enabled before a Qumulo cluster will accept S3 traffic.

To enable the S3 API via the `qq` CLI, use the `s3_modify_settings` command:

```
$ qq s3_modify_settings --enable
```

This will cause all nodes in the cluster to start accepting S3 API traffic on TCP port 9000.

Step 3: Creating an Access Key Pair

To use the S3 API with Qumulo, you need to have a valid S3 access key pair on the relevant Qumulo cluster. Every access key is associated with an existing user on the cluster or in a connected external identity service such as Active Directory.

> ⚠ Important
> Copy the secret key to a secure place as soon as you create it. It will only be displayed once.

To create an access key for a user via the `qq` CLI, use the `s3_create_access_key` command:

```
$ qq s3_create_access_key <username>
```

For a more detailed description, see Creating and Managing S3 Access Keys (page 103).

Step 4: Configuring the AWS CLI for Use with Qumulo Core

Configuring an application to use the S3 API with Qumulo Core requires a few additional steps. This section describes how to configure the AWS Command Line Interface, as that is what we'll use in later sections on this page. These configuration steps should be similar for other S3 applications that can connect to your Qumulo cluster.

> **Note**
>
> - Qumulo Core listens for S3 API traffic on TCP port 9000. This cannot be changed.
> - Only path-style bucket access is supported.

To Configure the AWS CLI

To use the AWS CLI with Qumulo Core, you must set the following configuration options in your AWS CLI S3 Configuration. We recommend using a dedicated profile in your AWS CLI config specifically .

```
$ aws configure --profile <your-profile> set s3.addressing_style path
$ aws configure --profile <your-profile> set s3.aws_access_key_id <access-key>
$ aws configure --profile <your-profile> set s3.aws_secret_access_key <secret-key>
```

This instructs the AWS CLI to use path-style bucket addressing, which is the only kind Qumulo Core supports. It also instructs it to use the access key pair you created in Step 3: Creating an Access Key Pair (page 0).

You will also need to point the CLI to the proper URI for your cluster. This cannot be configured persistently in the CLI. For the purposes of this guide, we recommend creating an alias for the AWS CLI:

```
$ alias aws="aws --endpoint-url https://<qumulo-cluster>:9000 --profile <your-profil
e>"
```

To Manually Specify an SSL Certificate

If your machine is not configured to trust the SSL certificate installed on the Qumulo cluster, you will also need to manually configure the path to the trusted SSL Certificate bundle:

```
$ aws configure --profile <your-profile> set ca_bundle <cert-bundle>
```

The `<cert-bundle>` is the SSL Certificate bundle created and installed during Step 1: Configuring HTTPS (page 0).

To Test Your Configuration

At this point you should be able to send a test S3 API request to your Qumulo cluster using the AWS CLI:

```
$ aws s3api list-buckets
```

A successful response should look like this, showing that no buckets have been created yet:

```
{
    "Buckets": []
}
```

Step 5: Creating an S3 Bucket

> **Note**
> Creating buckets requires the user to possess the `PRIVILEGE_S3_BUCKETS_WRITE` RBAC privilege and permission to create a directory under the cluster's root directory.

To create a bucket using the AWS CLI, run the following command:

```
$ aws s3api create-bucket --bucket <bucket-name>
```

This creates a new directory, `/<bucket-name>/`, under which all of the bucket's objects will live. See Creating and Managing S3 Buckets in Qumulo Core (page 111) for more details on how to manage S3 buckets and how they map to the Qumulo file system.

Step 6: Writing and Reading S3 Objects

Now that you have set everything up, you are ready to write and read data using the S3 API.

To Upload an Object to an S3 Bucket

To write data to the cluster over the S3 API, you can perform a `PutObject` action.

To perform a `PutObject` via the AWS CLI, use the `s3api put-object` subcommand:

```
$ aws s3api put-object --bucket <bucket-name> --key <object-name> --body <path-to-fi
le>
```

This will upload the contents of the local file at `<path-to-file>` into an object named `<object-name>` in the bucket.

To Download an Object from an S3 Bucket

To read back the uploaded object, you can perform a `GetObject` action.

To perform a `GetObject` via the AWS CLI, use the `s3api get-object` subcommand:

```
$ aws s3api get-object --bucket <bucket-name> --key <object-name> <path-to-file>
```

This will download the contents of the object into a local file at `<path-to-file>`. It also returns the metadata of the object, for example:

```
{
    "AcceptRanges": "bytes",
    "LastModified": "Wed, 14 Dec 2022 20:42:46 GMT",
    "ETag": "\"-gUAAAAAAAAAAwAAAAAAAAA\"",
    "ContentType": "binary/octet-stream",
    "Metadata": {}
}
```

Creating and Managing S3 Access Keys in Qumulo Core

S3 API actions in Qumulo Core require requests to be signed using an access key pair in order to be granted permissions to access file system resources. This section details how to create and manage such credentials.

> **Note**
> It is possible to configure a bucket to allow anonymous access, in which case no credentials are required, however this will limit the user to non-modifying operations. See Creating and Managing S3 Buckets in Qumulo Core (page 111).

Prerequisites

The following RBAC privileges are required for access key management:

- You must have the `PRIVILEGE_S3_CREDENTIALS_WRITE` privilege to create and delete S3 access keys.
- You must have the `PRIVILEGE_S3_CREDENTIALS_READ` privilege to list S3 access keys.

For more details see Role-Based Access Control (RBAC) with Qumulo Core.

Definitions

The following terms will be used:

- **Access key** or **access key pair**: A combination of an S3 access key ID and an S3 secret access key
- **Access key ID**: The public component of an S3 access key pair, which is used to identify the user performing an S3 request.
- **Auth ID**: Qumulo Core's common representation for identities, in the form of a numeric identifier.
- **Identity**: A single principal from an identity provider, e.g., a POSIX UID, an SMB SID, an Active Directory User Principal Name, or a local user on a Qumulo cluster.
- **Secret access key** or **secret key**: The private component of an S3 access key pair, which is used by the client to sign requests and by the server to validate request signatures.

How S3 Access Keys Work in Qumulo Core

> **Important**
> Qumulo Core secret access keys are derived from a FIPS 140-2 certified cryptographically secure source.

> **Note**
> Access keys are cluster local, thus an access key for a given identity in one Qumulo cluster will not be able to be used on a different Qumulo cluster.

Access key pairs are returned by Qumulo Core when an authorized user requests their creation, see Creating S3 Access Keys for a Qumulo Cluster (page 105). These are analogous to the IAM Access Keys used to access Amazon S3 resources, but specific to your Qumulo cluster. They should be used in a similar manner by applications when accessing objects stored in a Qumulo cluster via the S3 API.

Permissions Granted by an S3 Access Key

An S3 access key does not grant any additional permissions. A key simply associates an S3 API request with a specific identity known to the Qumulo cluster.

When a request is processed, permissions are evaluated using the same QACL mechanism that all other file system protocols use. Whichever permissions the QACL grants or denies to the associated identity will be granted or denied to the request being processed.

See “Managing Access to S3 Buckets” in Creating and Managing S3 Buckets in Qumulo Core (page 119) for more details on how to control access to S3 buckets on a Qumulo cluster.

How Qumulo Core Stores S3 Access Keys

> **Important**
> Secret access keys are not logged or displayed except on initial creation. If the secret access key is lost, it cannot be recovered and a new access key pair will need to be created.

In order to authenticate S3 API requests, Qumulo Core needs to store and retrieve the access key pairs that have been created.

Access key pairs are stored securely as configuration metadata in the Qumulo cluster. Secret access keys are encrypted on disk, and only held decrypted in memory while processing a request. However, access key IDs are not a cryptographic secret and can be logged and displayed.

S3 Access Key Lifecycle

There is no limit applied to how long an access key pair can be used after creation. Administrators can use the `qq` CLI or Qumulo REST API to see when they were created and revoke any pair at their own discretion. See Listing S3 Access Keys in a Qumulo Cluster (page 107).

⚠ Important

- A user identity may have at most two S3 access key pairs associated with it at any time.
- If an access key pair is revoked, it can not be restored. Ensure no critical applications depend on an access key pair before revoking it.

Each user identity can have up to two associated access key pairs to facilitate key rotation. After a new access key pair is created for a user, the user's old access key should be deleted. It is the administrator's responsibility to prescribe and enforce access key rotation policies within their organization.

Creating S3 Access Keys for a Qumulo Cluster

ℹ Note
Access keys must be created by an administrator or a user with the `PRIVILEGE_S3_CREDENTIALS_WRITE` privilege. Users without this privilege cannot create their own access keys.

In order to make S3 API requests against a Qumulo cluster as a given user, an S3 access key pair needs to be created for that user identity. You can create an S3 access key pair by using the `qq` CLI or by using the Qumulo REST API directly.

To Create an Access Key by Using the qq CLI

To create an S3 access key for a particular user identity, use the `s3_create_access_key` command:

```
$ qq s3_create_access_key IDENTITY
```

The `IDENTITY` argument can be specified in a number of different ways:

- A name, optionally qualified with a domain prefix:
  - `MY_NAME`
  - `local:MY_NAME`
  - `ad:MY_NAME`
  - `AD\MY_NAME`
- An Active Directory Security Identifier:
  - `SID:S-1-1-0`

- A Qumulo auth ID:
    - `auth_id:513`

Associating an S3 access key with a POSIX UID or GID is not currently supported.

The command output will have the following form:

```
{
    "access_key_id": "00000000000001f5b2dd",
    "creation_time": "2022-12-12T21:37:53.553457928Z",
    "owner": {
        "auth_id": "501",
        "domain": "LOCAL",
        "gid": null,
        "name": "guest",
        "sid": "S-1-5-21-1344365498-602363046-412545990-501",
        "uid": null
    },
    "secret_access_key": "TEIT4liMZ8A32iI7JXmqIiLWp5co/jmkjTSv3Rid"
}
```

In this example, the access key id is `00000000000001f5b2dd` and the secret access key is `TEIT4liMZ8A32iI7JXmqIiLWp5co/jmkjTSv3Rid`.

> **Important**
> This is the only time the secret access key will be returned or displayed. Make a copy of it and keep it in a safe place.

To Create an Access Key by Using the Qumulo REST API

To create an access key pair via the Qumulo REST API, send a `POST` request to `/v1/s3/access-keys/` with the following body (only one of `uid`, `sid`, or `auth_id` needs to be present):

```
{
  "user": {
    "uid": "<my_username>",
    "sid": "<my_sid>",
    "auth_id": "<my_auth_id>",
  }
}
```

The response will have the following form:

```
{
    "access_key_id": "00000000000001f5b2dd",
    "creation_time": "2022-12-12T21:37:53.553457928Z",
    "owner": {
        "auth_id": "501",
        "domain": "LOCAL",
        "gid": null,
        "name": "guest",
        "sid": "S-1-5-21-1344365498-602363046-412545990-501",
        "uid": null
    },
    "secret_access_key": "TEIT4liMZ8A32iI7JXmqIiLWp5co/jmkjTSv3Rid"
}
```


⚠ Important
This is the only time the secret access key will be returned or displayed. Make a copy of it and keep it in a safe place.

Listing S3 Access Keys in a Qumulo Cluster

You can list the S3 access keys known to the cluster, along with their associated identities and creation times, by using the `qq` CLI or by using the Qumulo REST API directly.

Note

- You must have the `PRIVILEGE_S3_CREDENTIALS_READ` privilege to list S3 access keys
- The access keys are listed in a consistent but unspecified order. You will need to do your own processing of the response to order keys by `creation_time`, `owner`, or to do other filtering

To List Access Keys by Using the qq CLI

To list all access keys known to the cluster, use the `s3_list_access_keys` command:

```
$ qq s3_list_access_keys
```

This outputs the list of access keys in the following form:

```
access_key_id         owner  creation_time
====================  =====  ==============================
00000000000001f5b2dd  Guest  2022-12-12T21:37:53.553457928Z
```

All times returned are in UTC.

To output results in JSON format

You can specify the `--json` flag if you would prefer this output to be JSON formatted:

```
$ qq s3_list_access_keys --json
```

This shows the JSON response bodies as returned by calls made to the `/v1/s3/access-keys/` endpoint of the Qumulo REST API. This API only returns up to 10,000 access keys per request, so if you have more access keys than this, you will see multiple response bodies. The `entries` field lists the access keys returned by the API call, and the `paging.next` field contains the URI used to request the next batch of access keys.


```
{
    "entries": [
        {
            "access_key_id": "00000000000001f5b2dd",
            "creation_time": "2022-12-12T21:37:53.553457928Z",
            "owner": {
                "auth_id": "501",
                "domain": null,
                "gid": null,
                "name": null,
                "sid": null,
                "uid": null
            }
        },
        ...
    ],
    "paging": {
        "next": null
    }
}
```


To List Access Keys by Using the Qumulo REST API

To list the S3 access keys known to the cluster, send a `GET` request to `/v1/s3/access-keys/`.

An optional `limit` query parameter can be included in the request, which will restrict the number of results returned, up to a maximum of 10,000 access keys. If not specified, the limit is 10,000.

The response will have the following form:

```
{
    "entries": [
        {
            "access_key_id": "00000000000001f5b2dd",
            "creation_time": "2022-12-12T21:37:53.553457928Z",
            "owner": {
                "auth_id": "501",
                "domain": null,
                "gid": null,
                "name": null,
                "sid": null,
                "uid": null
            }
        },
        ...
    ],
    "paging": {
        "next": null
    }
}
```

The `entries` list will contain the access keys, limited to the first 10,000 in the system. The `paging.next` field will contain the URI to which you can send a subsequent `GET` request to retrieve the next page of access keys. By calling `GET` on all `paging.next` values returned, you can iterate all access keys in the cluster.

Revoking and Deleting S3 Access Keys from a Qumulo Cluster

S3 access keys are revoked by deleting them from the Qumulo cluster. You can delete S3 access keys by using the `qq` CLI or by using the Qumulo REST API directly.

> **Note**
> You must have the `PRIVILEGE_S3_CREDENTIALS_WRITE` privilege to delete an S3 access key

To Delete an Access Key by Using the qq CLI

To revoke and delete an S3 access key pair from a Qumulo cluster, use the `s3_delete_access_key` command, replacing `<access-key-id>` with the access key ID that corresponds to the key pair that you wish to delete:

```
$ qq s3_delete_access_key --id <access-key-id>
```

To Delete an Access Key by Using the Qumulo REST API

To revoke and delete an S3 access key pair via the Qumulo REST API, send a `DELETE` request to `/v1/s3/access-keys/<access-key-id>`, replacing `<access-key-id>` with the access key ID that corresponds to the key pair that you wish to delete.

Creating and Managing S3 Buckets in Qumulo Core

This section explains how to create and manage S3 buckets on a Qumulo cluster.

S3 buckets provide a way to expose a portion of your Qumulo file system to applications that use the Amazon S3 API.

You can create and manage S3 buckets by using the Qumulo REST API or `qq` CLI, or by using the S3 API directly. The Qumulo REST API and `qq` CLI provide more flexibility. This section explains how to do both.

Prerequisites

This section assumes that you understand the terms “bucket”, “object”, and “key” as they pertain to the Amazon S3 API.

In order to use the S3 API to create and manage buckets, you must have a valid Qumulo S3 access key. See Creating and Managing S3 Access Keys (page 103).

In addition, to use the AWS CLI, you must follow the instructions in Getting Started with the S3 API (page 0).

Required Permissions

- `PRIVILEGE_S3_BUCKETS_WRITE` is required when creating and deleting S3 buckets via the Qumulo REST API or `qq` CLI.
    - Additionally, permission to create or delete directories is required, unless you opt to skip those steps when using the `qq` CLI.
- `PRIVILEGE_S3_BUCKETS_READ` is required when listing S3 buckets via the Qumulo REST API or `qq` CLI.

How S3 Buckets Map to the Qumulo File System

An S3 bucket exposes a portion of your Qumulo file system to applications that use the Amazon S3 API.

> **Note**
> S3 buckets can be rooted at any directory in the file system. Therefore, the same file may be an object in multiple buckets.

The *bucket root directory* (or *bucket root*) is the directory that an S3 bucket is attached to. All files under the bucket root directory (and all of its subdirectories) are objects in the bucket. The directory hierarchy is reflected by the presence of slash (`/`) characters in the objects’ keys.

How Object Keys Are Determined

The key for every object in a Qumulo S3 bucket is its file system path relative to the bucket's root directory. Objects that are directories will have a trailing slash (`/`) character in their key, while objects that are files will have no trailing slash character.

For example, suppose your Qumulo file system has the following contents:

```
/
├── banana-data/
│   └── banana.dat
│
└── grumpquat-data/
    ├── quat.dat
    ├── fruity/
    └── grumpy/
        ├── data1.dat
        └── data2.dat
```

If you have a bucket named `bucket1` rooted at `/grumpquat-data/grumpy/`, it contains objects with the following keys:

- `data1.dat`
- `data2.dat`

And if you have another bucket named `bucket2` rooted at `/grumpquat-data/`, it contains objects with the following keys:

- `quat.dat`
- `fruity/`
- `grumpy/data1.dat`
- `grumpy/data2.dat`

Note how both buckets contain `/grumpquat-data/grumpy/data1.dat` and `/grumpquat-data/grumpy/data2.dat` as objects. Note also that the `fruity/` object in `bucket2` has a trailing slash because it corresponds to a directory.

How Bucket Names Are Determined

You choose a name for your bucket when you create it. A bucket's name does not need to have any relation to its root directory.

Qumulo Core accepts all bucket names that Amazon S3 accepts, except for those that contain any dot (`.`) characters.

Bucket naming rules

This is the full list of restrictions for bucket names in Qumulo S3:

- Bucket names must be between 3 (min) and 63 (max) characters long.
- Bucket names can consist only of lowercase ASCII letters, numbers, and hyphens (`-`).
- Bucket names must start with a letter or a number.

For comparison, see Amazon's bucket naming rules.

How Bucket Root Directories Are Determined

When you create an S3 bucket via the Qumulo REST API or `qq` CLI, you have the option of selecting which directory should be used as the bucket root.

When you create an S3 bucket via the S3 API's `CreateBucket` action, a new directory with the same name as the bucket will be created under the default bucket directory prefix. See Configuring the Default Directory Prefix (page 0). The Qumulo REST API and `qq` CLI provide this same behavior if you do not specify a directory.

When a new directory is created for a new bucket, it will be owned by the user that made the request. See Managing Access to S3 Buckets (page 0) for more information on how to manage bucket permissions.

Creating S3 Buckets

> **Note**
>
> - All S3 buckets in a Qumulo cluster share the same namespace. There cannot be two buckets with the same name, even if they are rooted at different directories.
> - All S3 buckets must follow the bucket naming rules (page 113).

You can create an S3 bucket by using the Qumulo REST API or `qq` CLI, or by using the S3 API directly. The Qumulo REST API and `qq` CLI allow you to specify an existing directory as the new bucket's root, while the S3 API always creates a new directory for the bucket root.

To Create an S3 Bucket by Using the qq CLI

`PRIVILEGE_S3_BUCKETS_WRITE` is required when creating and deleting S3 buckets via the Qumulo REST API or `qq` CLI.

When using the `qq` CLI to create a bucket, you have the option of creating a new directory or rooting the bucket at a pre-existing directory.

To Create a New, Empty Bucket

To create an empty S3 bucket named `my-bucket`, use the `s3_create_bucket` command with the `--name` argument:

```
$ qq s3_create_bucket --name my-bucket
```

A new directory named `my-bucket` will be created under the default bucket directory prefix. See Configuring the Default Directory Prefix (page 0) for more information.

If an entry with the name specified name already exists, or if you do not have permission to create the directory, this command will return an error.

To Create a Bucket on an Existing Directory

To create an S3 bucket that exposes the files under a pre-existing directory, specify the `--path` argument as well:

```
$ qq s3_create_bucket --name my-bucket --path /some/existing/directory
```

If the specified directory does not exist, or if you do not have permission to look up the directory, this command will return an error.

To Create a Bucket by Using the S3 API

> **Note**
> The Qumulo S3 API server supports only the `"local"` region when using `CreateBucket` with a location constraint.

You can use the `CreateBucket` S3 action to create a new bucket on a Qumulo cluster.

To create a bucket named `my-bucket` with the AWS CLI, use the `s3api create-bucket` command:

```
$ aws s3api create-bucket --bucket my-bucket
```

The bucket's root directory will be created under the default bucket directory prefix (see Configuring the Default Directory Prefix (page 0)), and it will have the same name as the bucket. The directory must not already exist, and you must have permission to create the directory.

For the above example, if the default bucket creation directory is `/buckets/`, then the new bucket's root directory is `/buckets/my-bucket/`.

Configuring the Default Directory Prefix for S3 Buckets

The *default bucket directory prefix* is the directory under which new bucket root directories will be created when creating S3 buckets via the S3 API `CreateBucket` command. It is also used when creating S3 buckets via the Qumulo REST API and `qq` CLI if no directory is specified.

By default, the bucket root of a newly created bucket will be created under the cluster's root directory, `/`. You can change this using the `qq` CLI.

To Configure the Default Directory Prefix by Using the qq CLI

`PRIVILEGE_S3_SETTINGS_WRITE` is required when configuring the Qumulo S3 server settings.

To change the default directory prefix to `/buckets/`, use the `s3_modify_settings` command:

```
$ qq s3_modify_settings --base-path /buckets
```

To Check the Default Directory Prefix by Using the qq CLI

`PRIVILEGE_S3_SETTINGS_READ` is required when checking the Qumulo S3 server settings.

To see the current setting for the default directory prefix, use the `s3_get_settings` command:

```
$ qq s3_get_settings
```

The output will look like:

```
{"enabled": true, "base_path": "/buckets/"}
```

Configuring S3 Buckets

You can view and modify per-bucket settings by using the Qumulo REST API or `qq` CLI.

The only per-bucket setting that can be configured in Qumulo Core is anonymous access (page 0).

To View the Settings for an S3 Bucket by Using the qq CLI

`PRIVILEGE_S3_BUCKETS_READ` is required when viewing the settings for an S3 bucket.

To view the settings for the S3 bucket named `my-bucket`, use the `s3_get_bucket` command:

```
$ qq s3_get_bucket --name my-bucket
```

The output will look like this:

```
{
    "anonymous_access_enabled": false,
    "creation_time": "2022-12-20T19:42:26.833076147Z",
    "name": "my-bucket",
    "path": "/buckets/my-bucket"
}
```

To Enable Or Disable Anonymous Access for a Bucket by Using the qq CLI

`PRIVILEGE_S3_BUCKETS_WRITE` is required when enabling or disabling anonymous access for an S3 bucket.

To enable or disable anonymous access (page 0) for the bucket named `my-bucket`, use the `s3_modify_bucket` command:

```
$ qq s3_modify_bucket --name my-bucket --enable-anonymous-access
{
    "anonymous_access_enabled": true,
    "creation_time": "2022-12-20T19:42:26.833076147Z",
    "name": "my-bucket",
    "path": "/buckets/my-bucket"
}
$ qq s3_modify_bucket --name my-bucket --disable-anonymous-access
{
    "anonymous_access_enabled": false,
    "creation_time": "2022-12-20T19:42:26.833076147Z",
    "name": "my-bucket",
    "path": "/buckets/my-bucket"
}
```

Listing S3 Buckets

You can list the S3 buckets on a Qumulo cluster by using the Qumulo REST API or `qq` CLI, or by using the S3 API directly.

To List S3 Buckets by Using the qq CLI

`PRIVILEGE_S3_BUCKETS_READ` is required when listing S3 buckets via the Qumulo REST API or `qq` CLI.

To list all of the S3 buckets on a Qumulo cluster, use the `s3_list_buckets` command:

```
$ qq s3_list_buckets
```

The output will look like this:

```
name       creation_time                   path
=========  ==============================  ==========
my-bucket  2022-12-13T22:18:01.406433425Z  /my-bucket
```

All times returned are in UTC.

If you prefer to show the output in JSON format, you can append the `--json` argument:

```
$ qq s3_list_buckets --json
```

This returns the JSON response as returned by the Qumulo REST API endpoint that the `qq` CLI invokes:


```
{
    "buckets": [
        {
            "anonymous_access_enabled": false,
            "creation_time": "2022-12-13T22:18:01.406433425Z",
            "name": "my-bucket",
            "path": "/my-bucket"
        }
    ]
}
```


To List S3 Buckets by Using the S3 API

You can use the `ListBuckets` S3 action to list all of the S3 buckets in a Qumulo cluster.

To use the AWS CLI to list all S3 buckets for which you have read access, use the `s3api list-buckets` command:

```
$ aws s3api list-buckets
```

The output will look like this:

```
{
    "Buckets": [
        {
            "Name": "my-bucket",
            "CreationDate": "2022-12-13T22:18:01.406Z"
        }
    ]
}
```

All times returned are in UTC.

Deleting S3 Buckets

> **Note**
> All in-progress multipart uploads for an S3 bucket must be completed or aborted before deleting the bucket.

You can delete an S3 bucket by using the Qumulo REST API or `qq` CLI, or by using the S3 API directly. The Qumulo REST API and `qq` CLI allow you to choose whether or not the bucket root directory should be deleted. The S3 API always deletes the bucket root.

To Delete an S3 Bucket by Using the qq CLI

`PRIVILEGE_S3_BUCKETS_WRITE` is required when creating and deleting S3 buckets via the Qumulo REST API or `qq` CLI.

When using the `qq` CLI to delete a bucket, you have the option of deleting the bucket root directory or leaving it untouched.

To Delete a Bucket But Not Its Root Directory

To delete the S3 bucket named `my-bucket` without deleting its root directory, use the `s3_delete_bucket` command with the `--name` argument:

```
$ qq s3_delete_bucket --name my-bucket
```

This will remove all metadata related to the bucket from the Qumulo cluster. The bucket root does not need to be empty.

This command will return an error if any of the following are true:

- The specified bucket does not exist.
- You do not have `PRIVILEGE_S3_BUCKETS_WRITE`.
- The bucket has in-progress multipart uploads.

To Delete a Bucket And Its Root Directory

To delete the bucket and its root directory, specify both the `--name` and `--delete-root-dir` arguments:

```
$ qq s3_delete_bucket --name my-bucket --delete-root-dir
```

This command will return an error in the same cases as listed above, as well as if:

- You do not have permission to delete the bucket root.
- The bucket root is not empty.

To Delete an S3 Bucket by Using the S3 API

You can use the `DeleteBucket` S3 action to delete an S3 bucket from a Qumulo cluster.

To delete a bucket named `my-bucket` with the AWS CLI, use the `s3api delete-bucket` command:

```
$ aws s3api delete-bucket --bucket my-bucket
```

This will remove all metadata related to the bucket from the Qumulo cluster, and it will delete the bucket root directory.

This command will return an error if any of the following are true:

- The specified bucket does not exist.
- You do not have permission to delete the bucket root.
- The bucket root is not empty.
- The bucket has in-progress multipart uploads.

Managing Access to S3 Buckets

Managing users' access to S3 buckets on a Qumulo cluster is very similar to managing access to SMB shares or NFS exports. There are a few extra steps and important caveats, though.

First, users must be assigned S3 access keys in order to access any buckets on the cluster. See Creating and Managing S3 Access Keys (page 103). Alternatively, you can enable anonymous access on a bucket (page 120) to allow any user read-only access to the bucket. Access keys are still needed to perform modifying operations.

> **Note**
> There are no "bucket-level" permissions for S3 buckets in Qumulo, only per-file permissions. You can imitate bucket-level permissions by using inheritable ACEs (page 121).

Secondly, the behavior that users might expect from an S3 bucket may not align with how the bucket behaves without additional configuration. In Amazon's S3 service, access is provided to the entire bucket, but in Qumulo's S3 implementation, access is provided to individual files and directories. To configure a Qumulo S3 bucket to work more like an Amazon S3 bucket, see Using Inheritable ACEs to Imitate Bucket-Level Permissions (page 0).

How Permissions Work for S3 Buckets in Qumulo Core

Unlike Amazon S3, which lets you grant access at the bucket level, Qumulo Core only lets you grant access at the level of individual files and directories.

In Qumulo Core, every S3 API request is processed by performing one or more file system operations. These operations are processed the same way as other protocols such as SMB or NFS: by checking the user's access against the per-file access control lists (ACLs) of the file(s) involved.

For authenticated requests (those that are signed with Amazon Signature Version 4), Qumulo Core maps the access key ID in the request to its corresponding auth ID, and the request is processed as that user. For anonymous requests (those that are not signed), the request is processed as the `Guest` user, and all modifying operations are forbidden.

When processing an S3 request, any new files and directories that are created are owned by the user that made the request. They inherit access control entries (ACEs) from their parents just as they do when created through any other protocol. You can leverage the inheritable permissions to imitate bucket-level permissions similar to Amazon S3 (page 0).

For more information about S3 access keys and how they map to user identities, see Creating and Managing S3 Access Keys (page 103). For more information on anonymous access, see below.

Enabling Anonymous Access for a Bucket

Qumulo Core allows you to grant read-only access for anonymous S3 API requests on a per-bucket basis. This can be useful, for example, to allow for plain HTTPS access to a bucket's contents, e.g., via `wget` or a web browser.

If you want to grant applications unathenticated, read-only access to an S3 bucket, you must enable anonymous access on that bucket. Anonymous access is disabled by default for all newly created buckets and must be enabled as described in Configuring S3 Buckets (page 0).

> **Important**
> Anonymous S3 API requests are forbidden from making any modifications to a bucket, even when anonymous access is enabled.

When anonymous access is enabled for a bucket, applications may send anonymous requests to the bucket for non-modifying operations. These requests will be performed as the `Guest` user. Modifying operations must always be authenticated, even when anonymous access is enabled.

> **Note**
> Anonymous S3 API requests will not have read access to files if their ACLs do not allow reads for the `Guest` user.

The `Guest` user is a member of the `Everyone` group, but not of the `Users` group. To ensure that anonymous requests have permission to read files in a bucket, grant read permission to the `Everyone` group, or simply to the `Guest` user. See Imitating Bucket-Level Read-Only Access (page 0) below.

Using Inheritable ACEs to Imitate Bucket-Level Permissions

If you want to grant multiple users access to all paths within a bucket, you can use inheritable access control entries (ACEs) to ensure that newly created directories inherit the proper permissions.

> **Note**
> Adding inheritable ACEs to a directory does not affect any files that already exist under that directory. See this Qumulo Care article to see how to apply permissions recursively to existing files.

When you first create your bucket, you can add inheritable ACEs to that bucket's root directory. After doing so, all files and directories that are created under that bucket root will inherit those ACEs and pass them on to any files and directories that get created under them.

You can also put inheritable ACEs on the default bucket directory prefix so that all newly created buckets inherit the same set of ACEs.

You can add ACEs to a directory by using the `qq` CLI, which is described below. You can also use the File Explorer on a Windows client that has an SMB share mapped containing the directory.

How Permissions Work Without Inheritable ACEs

In Amazon S3, permission to read and write objects in a bucket is granted for the entire bucket. In Qumulo Core, each object key corresponds to a file path relative to some root directory, and permissions are granted for each individual file and directory.

When users create objects in an S3 bucket on a Qumulo cluster, they may end up creating new directories. These directories are owned by the user that created them. Without the proper access control entries in your bucket, these new directories may have restrictive permissions that prevent other users from creating objects with the same prefix.

Imitating Bucket-Level Read-Write Access with Inheritable ACEs

The following access control entry (ACE), when added to a bucket's root directory, will imitate bucket-level read-write access for some user or group of users:

```
Type     Flags                                Rights
=======  ===================================
================================================
Allowed  Object inherit, Container inherit  Delete child, Execute/Traverse, Read, Wr
ite file
```

To use the `qq` CLI to add this ACE to the bucket rooted at `/buckets/my-bucket` for the user group named `MyWriters`, run the following command:

```
$ qq fs_modify_acl --path /buckets/my-bucket add_entry -t MyWriters \
  -y Allowed \
  -f 'Container inherit' 'Object inherit' \
  -r 'Delete child' 'Execute/Traverse' 'Read' 'Write file'
```

Imitating Bucket-Level Read-Only Access with Inheritable ACEs

The following access control entry (ACE), when added to a bucket's root directory, will imitate bucket-level read-only access for some user or group of users:

```
Type     Flags                                Rights
=======  ===================================  ======================
Allowed  Object inherit, Container inherit  Execute/Traverse, Read
```

To use the `qq` CLI to add this ACE to the bucket rooted at `/buckets/my-bucket` for the user group named `MyReaders`, run the following command:

```
$ qq fs_modify_acl --path /buckets/my-bucket add_entry -t MyReaders \
  -y Allowed \
  -f 'Container inherit' 'Object inherit' \
  -r 'Execute/Traverse' 'Read'
```

Imitating Bucket-Level List-Only Access with Inheritable ACEs

The following two access control entries (ACEs), when added to a bucket's root directory, will imitate bucket-level list-only access for some user or group of users:

```
Type     Flags              Rights
=======  =================  ======================
Allowed  Container inherit  Execute/Traverse, Read
Allowed  Object inherit     Read attr
```

To use the `qq` CLI to add these ACEs to the bucket rooted at `/buckets/my-bucket` for the user group named `MyListers`, run the following commands:

```
$ qq fs_modify_acl --path /buckets/my-bucket add_entry -t MyListers \
  -y Allowed \
  -f 'Container inherit' \
  -r 'Execute/Traverse' 'Read'
$ qq fs_modify_acl --path /buckets/my-bucket add_entry -t MyListers \
  -y Allowed \
  -f 'Object inherit' \
  -r 'Read attr'
```

Supported S3 Functionality and Known Limits for Qumulo Core

This section documents the S3 API functionality that Qumulo Core does and does not support, along with various numeric limits for the S3 API.

Supported S3 API Actions

The following table lists all S3 API actions that Qumulo Core supports and the version in which support was added. For the full list of S3 API actions, see Amazon's documentation.

Action	Supported Since
`AbortMultipartUpload`	5.3.3
`CompleteMultipartUpload`	5.3.3
`CopyObject`	5.3.3
`CreateBucket`	5.2.3*
`CreateMultipartUpload`	5.3.3
`DeleteBucket`	5.2.4*
`DeleteObject`	5.2.1*
`DeleteObjects`	5.2.2*
`GetBucketLocation`	5.1.2*
`GetObject`	5.0.4*
`HeadBucket`	5.1.2*
`HeadObject`	5.0.4*
`ListBuckets`	5.0.4*
`ListMultipartUploads`	5.3.3
`ListObjects`	5.0.5*
`ListObjectsV2`	5.0.4*

Action	Supported Since
`ListParts`	5.3.3
`PutObject`	5.2.1*
`UploadPart`	5.3.3

* The S3 API became generally available in Qumulo Core version 5.3.3. Some S3 API actions were available in preview mode in prior versions. The functionality of these actions in versions prior to 5.3.3 — and the process required to enable the S3 API in those versions — is not documented.

Unsupported S3 Functionality

The following table contains a non-exhaustive list of S3 API functionality that is not supported in Qumulo Core.

Unsupported Feature	Comments
BitTorrent	
Bucket ACLs	These can be imitated by using inheritable ACEs (page 0).
Bucket lifecycle configurations	
Bucket notifications	
Control of server-side encryption	All data is encrypted at rest in Qumulo Core, but this cannot be controlled via the S3 API.
Logging controls	
Multi-chunk payload signing	The streaming version of Amazon Signature Version 4 is not supported. Only the single chunk version is supported.
Object locks	
Object tagging and custom object metadata	
Object versioning	Qumulo objects have only a single version. Qumulo Snapshots can be used to preserve the previous contents of objects.
Policies	

Unsupported Feature	Comments
Signature version 2	Only version 4 signatures are accepted by Qumulo Core.
Storage classes	Qumulo Core has no concept of storage classes. All objects have the same storage class status.
Retention policies	
Temporary access credentials	
Virtual-hosted bucket addressing	Only path-style bucket addressing is supported.
Web hosting configuration	

Key Differences

This subsection documents the most important limitations that are specific to Qumulo's implementation of the S3 API.

The limitations documented below can be summarized as follows:

- **Bucket addressing style**: Only path-style bucket addressing is supported.
- **ETags**: ETags returned by Qumulo Core are not MD5 checksums; they have no interpretable meaning.
- **Listing objects**:
  - Results are sorted in a consistent but unspecified order (not alphabetically).
  - The only supported delimiter is the slash (`"/"`) character.
  - Only prefixes that correspond exactly to objects in the bucket will function.
- **Request authentication**: Only Amazon Signature Version 4 is supported.

Bucket Addressing Style

Amazon S3 supports two styles of bucket addressing in requests: virtual-hosted and path-style. See Bucket Addressing Styles.

Qumulo Core supports only path-style bucket addressing. Therefore, you must configure your client applications to use path-style addressing in order to perform S3 API requests against a Qumulo cluster. We document how to do this for the AWS CLI in the Getting Started (page 0) section.

ETags

RESTful APIs like S3 use HTTP ETags to identify different versions of a resource.

Amazon's S3 service uses the MD5 checksum of an object's contents as its ETag. Qumulo Core uses a different, unspecified mechanism to generate an object's ETag.

Well-behaved applications should not attempt to interpret the contents of an ETag. However, there are some applications that assume that S3 object ETags contain the MD5 checksum of the object's contents. Such applications may not function properly against Qumulo's implementation of the S3 protocol.

Listing Objects

The S3 API supports listing objects in a bucket via the `ListObjects` and `ListObjectsV2` actions.

Amazon S3 returns results in alphabetical order by object name. Qumulo Core returns results in a consistent but unspecified order (not alphabetical).

Amazon S3 supports using an arbitrary `prefix` to limit results to object names that start with that prefix. Qumulo Core provides partial support for this; it only works if the prefix is a path to some file or directory under the bucket root.

Amazon S3 supports using an arbitrary `delimiter` to group results into common prefixes. Qumulo Core supports only the slash (`"/"`) character as a delimiter.

Although Qumulo Core's support for `delimiter` and `prefix` is partial, it handles the most common use case, which is to treat an S3 bucket as a hierarchical file tree.

Request Authentication

Qumulo Core supports authenticating requests using Amazon Signature Version 4 only. Most S3 client applications support this. If your application attempts to use an earlier Amazon signature scheme, you will see a `400 Bad Request` response with error code `AuthorizationHeaderMalformed`.

Known Limits

The following tables list the various numeric limits of the S3 API for Qumulo Core and how they compare to Amazon S3.

Limits for S3 Buckets

Limit	Qumulo Specification	Amazon Specification
Maximum number of buckets	16,000	1,000
Maximum number of objects in one bucket	Unlimited*	Unlimited
Minimum bucket name length	3 characters	3 characters

Limit	Qumulo Specification	Amazon Specification
Maximum bucket name length	63 characters	63 characters

* If all objects in a bucket fall under the same directory (that is, none of the object keys have the `/` character in them), the maximum number of objects in the bucket will be limited to the maximum number of files in a directory. See .

Limits for S3 Objects

Limit	Qumulo Specification	Amazon Specification
Minimum object size	0 bytes	0 bytes
Maximum object size (via `PutObject`)	5 GiB	5 GiB
Maximum object size (via multipart upload)	48.8 TiB (10,000 * 5 GiB)	5 TiB
Minimum object key length	1 character	1 character
Maximum object key length	1,530 characters if no slash (`/`) characters in the key	1,024 characters

Limits for S3 Multipart Uploads

Limit	Qumulo Specification	Amazon Specification
Minimum part ID	1	1
Maximum part ID	10,000	10,000
Minimum number of parts per upload	1	1
Maximum number of parts per upload	10,000	10,000
Minimum part size	5 MiB (except for the last part in an upload)	5 MiB (except for the last part in an upload)
Maximum part size	5 GiB	5 GiB

Limit	Qumulo Specification	Amazon Specification
Additional part size requirements	Must be a multiple of 4 KiB (4096 bytes), except for the last part in an upload	N/A

Limits for S3 API Requests

Limit	Qumulo Specification	Amazon Specification
Maximum number of object keys specified in `DeleteObjects`	Unlimited*	1,000
Maximum number of buckets returned from `ListBuckets`	16,000	1,000
Maximum number of objects returned from `ListObjects` or `ListObjectsV2`	1,000	1,000
Maximum number of parts returned from `ListParts`	Unlimited	1,000
Maximum number of uploads returned from `ListMultipartUploads`	1,000	1,000

There are also Qumulo-specific limits on the request payload size for certain S3 actions.

Action	Maximum Payload Size
`CompleteMultipartUpload`	10 MiB
`CreateBucket`	10 MiB
`DeleteObjects`	10 MiB

* `DeleteObjects` is subject to a 10 MiB request payload limit in Qumulo, which provides a practical upper limit on the number of object keys.

Using NFSv4.1 with Kerberos in Qumulo Core

How NFSv4.1 Works with Kerberos in Qumulo Core

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

Related Topics

- Prerequisites for Joining a Qumulo Cluster to Active Directory (page 0)
- Configuring Active Directory for Use With Kerberos (page 0)
- Performing Additional Cluster Configuration after Joining Active Directory (page 0)
- Using Kerberos Permissions in the Qumulo Filesystem (page 0)
- Configuring a Linux Client for NFSv4.1 with Kerberos (page 0)
- Configuring Cross-Domain Active Directory Trusts (page 0)
- Troubleshooting NFSv4.1 with Kerberos (page 0)

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 to 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
- 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` → `203.0.113.0`, `qumulo.example.com` → `203.0.113.1`, and so on).

To Configure DNS Round Robin

1. Join your Qumulo cluster to AD (page 132).
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.co
m:/
```

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 132), you must configure the NFSv4.1 server (page 76) 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,
     "krbi_enabled": true,
     "v4_enabled": false
   }
   ```

   This is the default configuration:

   - NFSv4.1 is disabled by default.
   - `AUTH_SYS`, `AUTH_KRB5`, `AUTH_KRB5P`, and `AUTH_KRB5I` are enabled by default (however, Qumulo Core doesn't support Kerberos configuration on 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,
  "auth_krb5i_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-krb5i`	Enables `AUTH_KRB5I` 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`
`qq nfs_modify_settings --disable-krb5i`	Disables `AUTH_KRB5I` 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 76) to configure access to the Qumulo file system.

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.

Otherwise, exports and IP address restrictions (that you specify in exports) behave identically for all authentication flavors: `AUTH_SYS`, `AUTH_KRB5`, `AUTH_KRB5P`, and `AUTH_KRB5I`.

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 83).

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 149).

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 149).

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.

ℹ 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 <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 for configuring NFSv4.1 to 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 encrypted Kerberos traffic, use Wireshark with a Kerberos keytab file. For more information, see Kerberos 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 132).

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. Inspect logs and packet captures.

Configuring and Collecting Metrics in Qumulo Core

Qumulo OpenMetrics API Specification

This section lists the names, types, labels, and descriptions for the metrics that Qumulo Core 5.3.0 (ans higher) emits in OpenMetrics API format.

The Qumulo OpenMetrics API has a single endpoint that provides a complete view of point-in-time telemetry from Qumulo Core to monitoring systems. These systems, such as Prometheus, can consume the OpenMetrics data format that the Qumulo REST API emits without custom code or a monitoring agent. For more information about data formats, see your monitoring system's documentation.

Accessing Qumulo Metrics

Qumulo metrics are available at the following endpoint.

```
https://<my-cluster-hostname>:8000/v2/metrics/endpoints/default/data
```

You can configure a monitoring system that supports the OpenMetrics Specification to use bearer token authentication (page 33) to access this endpoint.

Metric Types

All Qumulo metrics belong to one of the following OpenMetrics types.

Metric Type	Description
`counter`	An integer that increases monotonically from zero, stored in `<metric_name>_count` . **Note** During normal operation, the value of `counter` never decreases.

Metric Type	Description
`gauge`	A value that represents a single integer (similar to `counter`), stored in `<metric_name>`. **Note** During normal operation, the value of a `gauge` metric might increase or decrease.
`histogram`	A representation of a series of *buckets*, where each bucket tracks values within a specific range. A histogram has a `count` field and a `sum` field, stored in `<metric_name>_count` (the total number of samples) and `<metric_name>_sum` (the sum of all samples). Qumulo Core emits a single bucket that contains all samples. **Tip** You can use `histogram` metrics to keep track of averages by dividing the `sum` field by the `count` field.
`info`	Informational text about the system, stored in `<metric_name>_info`. An `info` metric always has a value of `1` and labels that contain detailed information.

For more information, see Metric Types in the OpenMetrics Specification.

Metric Labels

The OpenMetrics format allows for metric labeling for communicating additional information. To provide context for metrics, Qumulo Core emits metric-specific labels. For example, the `name` of a protocol operation or the `url` of a remote server. For more information, see Available Labels (page 172).

Available Metrics

The following table lists metric names, types, labels, and descriptions.

Note
For Qumulo as a Service, all metrics with a `node_id` label are unavailable because they refer to specific hardware.

Metric Name	Metric Type	Labels	Description
`qumulo`	`info` (page 163)	• `name` • `uuid` • `version`	Qumulo Co information cluding the cluster nam cluster UUI and the cu Qumulo Co version
`qumulo_ad_netlogon_request_errors`	`counter` (page 162)	• `server_url` (page 175)	The total nu ber Active D rectory (AD `NETLOGON` r quests that sulted in an ror
`qumulo_ad_netlogon_request_latency_seconds`	`histogram` (page 163)	• `server_url` (page 175)	The total la cy for AD `N OGON` reque
`qumulo_ad_netlogon_requests`	`counter` (page 162)	• `server_url` (page 175)	The total nu ber of comp ed AD `NETL GON` operat
`qumulo_cpu_max_temperature_celsius`	`gauge` (page 163)	• `cpu` (page 172) • `node_id` (page 174)	The maxim temperatu threshold f each physic CPU
`qumulo_disk_endurance_percent`	`gauge` (page 163)	• `disk_type` (page 173) • `drive_bay` (page 173) • `node_id` (page 174)	The remain disk endura value for ea disk in the ter, ranging `100` (no dis wear) to `0` is worn fully

Metric Name	Metric Type	Labels	Description
qumulo_disk_transport_errors	counter (page 162)	• disk_type (page 173) • drive_bay (page 173) • node_id (page 174)	The total nu ber of comm nication er between th specified d and its host
qumulo_disk_uncorrectable_media_errors	counter (page 162)	• disk_type (page 173) • drive_bay (page 173) • node_id (page 174)	The total nu ber of uncc rectable er on the spec fied drive's physical me
qumulo_cpu_temperature_celsius	gauge (page 163)	• cpu (page 172) • node_id (page 174)	The tempe ture for eac physical CP degrees Ce
qumulo_disk_is_unhealthy	gauge (page 163)	• disk_type (page 173) • drive_bay (page 173) • node_id (page 174)	The health each disk i cluster, ran from 0 (th disk is heal to 1 (the d unhealthy)

Metric Name	Metric Type	Labels	Description
qumulo_disk_operation_latency_seconds	histogram (page 163)	• disk_type (page 173) • drive_bay (page 173) • io_type (page 174) • node_id (page 174)	The total la cy for disk I operations
qumulo_fan_speed_rpm	gauge (page 163)	• fan (page 173) • node_id (page 174)	The fan spe in RPM
qumulo_fs_capacity_bytes	gauge (page 163)	—	The total cl space, in by
qumulo_fs_directory_tree_entries	gauge (page 163)	• entry_type (page 173) • path (page 174)	The numbe file system jects on the cluster, sort by object ty
qumulo_fs_directory_used_bytes	gauge (page 163)	• path (page 174) • usage_type (page 175)	The amoun space that ject types u in bytes
qumulo_fs_free_bytes	gauge (page 163)	—	The free sp on the clus in bytes
qumulo_fs_snapshots	gauge (page 163)	—	The numbe snapshots the cluster

Metric Name	Metric Type	Labels	Description
qumulo_ldap_lookup_request_errors	counter (page 162)	• server_url (page 175)	The total number of LDAP requests that resulted in an error
qumulo_ldap_lookup_request_latency_seconds	histogram (page 163)	• server_url (page 175)	The total latency of LDAP requests
qumulo_ldap_lookup_requests	counter (page 162)	• server_url (page 175)	The total number of completed LDAP requests
qumulo_ldap_operation_errors	counter (page 162)	—	The total number of LDAP operations that resulted in an error
qumulo_ldap_operation_latency_seconds	histogram (page 163)	—	The total latency for LDAP operations
qumulo_ldap_operations	counter (page 162)	—	The total number of completed LDAP operations
qumulo_memory_correctable_ecc_errors	counter (page 162)	• node_id (page 174)	The total number of memory errors that Qumulo Core corrected automatically

Metric Name	Metric Type	Labels	Description
qumulo_network_interface_is_down	gauge (page 163)	• name (page 172) • role (page 174) • interface (page 173) • node_id (page 174)	The interfac status, 0 (i face is up) (interface is down)
qumulo_network_interface_link_speed_bits_per_second	gauge (page 163)	• name (page 172) • role (page 174) • interface (page 173) • node_id (page 174)	The negotia link speed f the specifie interface
qumulo_network_interface_receive_errors	counter (page 162)	• name (page 172) • role (page 174) • interface (page 173) • node_id (page 174)	The total nu ber of recei errors on th specified in face

Metric Name	Metric Type	Labels	Description
qumulo_network_interface_received_bytes	counter (page 162)	• name (page 172) • role (page 174) • interface (page 173) • node_id (page 174)	The total by received or specified in face
qumulo_network_interface_received_packets	counter (page 162)	• name (page 172) • role (page 174) • interface (page 173) • node_id (page 174)	The total nu ber of pack received or specified in face
qumulo_network_interface_transmit_errors	counter (page 162)	• name (page 172) • role (page 174) • interface (page 173) • node_id (page 174)	The total nu ber of trans sion errors the specifie interface

Metric Name	Metric Type	Labels	Description
qumulo_network_interface_transmitted_bytes	counter (page 162)	• name (page 172) • role (page 174) • interface (page 173) • node_id (page 174)	The total nu ber of bytes transmitted the specifie interface
qumulo_network_interface_transmitted_packets	counter (page 162)	• name (page 172) • role (page 174) • interface (page 173) • node_id (page 174)	The total nu ber of pack transmitted the specifie interface
qumulo_power_supply_is_unhealthy	gauge (page 163)	• location (page 174) • node_id (page 174)	PSU health (healthy) or (unplugged moved, or u sponsive)
qumulo_protocol_client_connections	counter (page 162)	• protocol (page 174)	The total nu ber of clien that have c nected to t specified p col

Metric Name	Metric Type	Labels	Description
qumulo_protocol_client_disconnections	counter (page 162)	• protocol (page 174)	The total nu ber of clien that have d connected from the sp fied protoc
qumulo_protocol_operation_bytes	counter (page 162)	• data_type (page 173) • io_type (page 174) • op_name (page 174) • protocol (page 174)	The total by that protoc operations transferred
qumulo_protocol_operation_latency_seconds	histogram (page 163)	• data_type (page 173) • io_type (page 174) • op_name (page 174) • protocol (page 174)	The total la cy for proto operations
qumulo_protocol_operations	counter (page 162)	• data_type (page 173) • io_type (page 174) • op_name (page 174) • protocol (page 174)	The total nu ber of comp ed protocol erations

Metric Name	Metric Type	Labels	Description
`qumulo_quorum_node_is_offline`	`gauge` (page 163)	• `node_id` (page 174)	The online s tus for each node in the cluster, `0` (node onlin `1` (node of
`qumulo_time_is_not_synchronizing`	`gauge` (page 163)	• `node_id` (page 174)	The time sy chronizatio status for e node in the cluster, `0` (is synchron or `1` (time synchroniz

Available Labels

The following table lists metric label names, possible values, and descriptions.

Label Name	Possible Values	Description
`bond`	• `bond0` • `bond1`	The bond to which a network interface belongs
`cpu`	A non-negative integer	The CPU index in the node

Label Name	Possible Values	Description
`data_type`	• `data` : Read or write operations on the data of a file. • `metadata` : Operations (such as `lookup` , `stat` , or `getattr`) unrelated to a file's data • `none` : Operations that operate on neither the file data nor the metadata. **Note** The protocol often requires these operations for session negotiation and authentication.	The data type that an operation transfers
`disk_type`	• `hdd` : Hard Disk Drive • `ssd` : Solid-State Drive	The underlying storage type
`drive_bay`	A drive bay name. For example: `b3` , `1.1`	The physical drive bay in the chassis.
`entry_type`	• `alternate_data_stream` • `directory` • `file` • `other` • `symlink`	The file system object type
`fan`	A fan name, for example `system fan 1`	The fan name
`interface`	An interface name, for example `eth0`	The interface name

Label Name	Possible Values	Description
`io_type`	• `none` • `read` • `wait` : A blocking operation that takes an indeterminate amount of time • `write`	The I/O that an operation performs
`location`	A location on the chassis, for example `left` or `right`	The location on the chassis. **Note** For PSU, this location is relative to the back of the node.
`node_id`	A positive integer that represents a node ID in the cluster.	A value that differentiates between the different nodes in a cluster
`op_name`	Any operation name, including NFSv3, NFSv4.1, SMBv2, SMBv3 or FTP	The recorded operation
`path`	Slash (`/`)	The path to a directory in the file system
`protocol`	• `nfs` : NFSv3 or NFSv4.1 • `smb2` : SMBv2 or SMBv3 • `ftp`	The protocol of the recorded operation
`role`	• `frontend` • `backend`	The role of the interface **Note** `frontend` includes protocol, management, and replication traffic. `backend` includes all intra-node communications.

Label Name	Possible Values	Description
`server_url`	A hostname (for example, `ad.my-domain.com`) or an IP address	The URL of a remote server
`usage_type`	• `data` • `metadata` • `snapshot`	The data type that uses space