Qumulo On-Premises Administrator Guide

Copyright © 2024 Qumulo, Inc.

Table of Contents

Getting Started

Installing the Qumulo Core Product Package 5
Creating and Managing Directory Quotas 12
Supported Configurations and Known Limits 13

System Upgrades

Feature Log 17
Upgrade Modes 40
Performing Upgrades 44
Replication Version Requirements and Upgrade Recommendations 50

Authentication

Installing VPN Keys 54
Configuring SAML Single Sign-On (SSO) 57
Configuring Search Trusted Domains 66
Configuring LDAP 68
Creating and Using Bearer Tokens 69

Connecting to External Services

Creating and Using Access Tokens 71
Connecting a Kubernetes Cluster 81

Authorization

Managing File Locks 89
Managing Role-Based Access Control 90
Managing Cross-Protocol Permissions 93
Using SMB Host Restrictions 95
Using Active Directory for POSIX Attributes 97

Network Configuration

Required Networking Ports 100
Configuring IPv6 102
Connecting to Multiple Virtual Networks 104
Configuring Round-Robin DNS on Windows Server 106

Network Multitenancy

Partitioning a Cluster into Tenants 108
Configuring Management Protocols 113
Configuring File System Protocols 116

Web UI

Setting the Web UI Login Banner 122
Setting the Web UI Inactivity Timeout 123

qq CLI

Getting Started with the qq CLI 124
Enabling Autocomplete for the qq CLI 126

Metadata

Managing User-Defined Metadata 129

Snapshots

How Snapshots Work 132
Managing Snapshots 135
Locking and Unlocking Snapshots 140
Recovering Files by Using Snapshots 143

Encryption and Data Security

Managing Encryption at Rest 145
Managing SMB3 Encryption in Transit 150
Generating and Storing ECDSA Keys 153
Managing Security Keys 159
Installing a Signed SSL Certificate 164

Node Addition and Replacement

How Drive and Node Failure Protection Works 166
Adding Nodes to an Existing Cluster 172
Increasing the Node Fault Tolerance Level during Node-Add Operations 174
Adding Nodes with Adaptive Data Protection 175
Replacing Nodes by Performing a Transparent Platform Refresh 178
Improving Performance by Migrating a Cluster to a Different License Class 184
Understanding Offline Nodes 185

Data Replication

Creating and Managing a Continuous Replication Relationship 186
Shift-To Amazon S3 191
Shift-From Amazon S3 203

File System Changes

How File System Change Notifications Work 215
Watching for Changes with SMB2 CHANGE_NOTIFY 219
Watching for Changes with REST 223

NFS

Creating and Managing an NFS Export 229
Enabling and Using NFSv4.1 231
Managing File Access Permissions with ACLs 238
Host Access Rules for NFS Exports 246

NFSv4.1 with Kerberos

How NFSv4.1 Works with Kerberos 252
Prerequisites for Joining to Active Directory 254
Configuring Active Directory 256
Performing Additional Cluster Configuration 261
Using Kerberos Permissions 264
Configuring a Linux Client 270
Configuring Cross-Domain Active Directory Trusts 280
Troubleshooting NFSv4.1 with Kerberos 282

SMB

Creating and Managing an SMB Share 285
Managing File Shares by Using the Shared Folders MMC Snap-In 287

S3 API

Configuring and Using the S3 API 290
Creating and Managing S3 Access Keys 295
Creating and Managing S3 Buckets 303
Managing Access to S3 Buckets 312
Managing Access Policies for S3 Buckets 319
Managing Multipart S3 Uploads 327
Managing S3 Bucket Versioning 333
Supported Functionality and Limits 335

Monitoring and Metrics

Enabling Cloud-Based Monitoring and Remote Support 343
Connecting to Cloud-Based Monitoring and S3 with a Custom Proxy 347
Restoring the Default Values for Cloud-Based and Nexus Monitoring 349
How Audit Logging Works in Qumulo Core 351
OpenMetrics API Specification 369

Getting Started

Installing the Qumulo Core Product Package

This section explains how to install the Qumulo Core Product Package on a user-managed, Linux-based host system from a `.deb` or `.rpm` package.

The Qumulo Core Product Package permits flexible deployment: For example, it can let your organization adhere to its security and compliance requirements, or use a specific Linux distribution that has become standardized across the entire system fleet.

> ☑ Tip
> The Qumulo Core Product Package is currently in Private Availability. To obtain a copy, contact the Qumulo Care team.

How is the Qumulo Core Product Package Different from the Qumulo USB Installer?

Whereas the Qumulo USB Installer is designed for specific models of third-party hardware bundled with Qumulo Core, the Qumulo Core Product Package is designed for installation on your own hardware.

Because Qumulo has no control over the host operating system (OS), the following are the main differences in functionality between the two.

- **Web UI:** The Qumulo Core Product Package has no kiosk mode. The Web UI runs directly on your node.
- **Well-Known `admin` User:** When you use the Qumulo Core Product Package, changing the `admin` user's password has no effect on the host OS. You must create your own users on the host OS.
- **Automatic SSH Configuration:** Any SSH configuration set by using `multitenancy` REST APIs have no effect on the host OS. You must configure SSH on the host OS.
- **System Partitions and Directories:** The Qumulo Core Product Package has no `/config` partition for storing logs and container images or `/history` partitions for storing configuration files.
  The Qumulo Core container stores logs and container images in the `/var/opt/qumulo/history` directory and configuration files in the `/etc/qumulo` directory. You can also configure your own mounts and partitions on the host OS.
- **Core Dump Handler:** You must configure the core dump handler on the host OS. For more information, see `core` Linux Manual Page.

For more information, see Creating a Qumulo Core USB Drive Installer in the Qumulo-Certified Hardware Servicing Guide.

Prerequisites

Before installing the Qumulo Core Product Package, you must ensure your system satisfies the minimum requirements and prepare and configure the host operating system.

Step 1: Ensure Minimum System Requirements

Ensure that your host system meets the minimum requirements.

Requirement	Description
Linux Distribution	A `systemd`-based Linux distribution< **Note** • We've performed comprehensive testing of the `.deb` package running on Ubuntu 20.04, on-premises and on AWS, as well as spot-checks of installation and clustering processes on Ubuntu 22.04, Ubuntu 24.04, and Debian 11. • We've also spot-checked the `.rpm` package by performing installation and clustering on the Rocky 9.3 distribution (compatible with RHEL in terms of performance and known issues).

Kernel	**Important** Kernel versions prior to 5.19 have a bug that can cause core dumps from Qumulo Core to be truncated. We've tested support for the following kernel versions: • Debian 6.1.0 (and higher) • Rocky or Red Hat Enterprise Linux (RHEL) 5.14 (and higher) • Ubuntu 5.15 (and higher) **Note** To ensure compatibility with other kernels, check whether the following features are supported. • `io_uring`: To check whether your kernel has this system call, run the `grep io_uring_setup /proc/kallsyms` command. • `tcp_bbr`: To check whether your kernel has this congestion control algorithm, run the `sudo modinfo tcp_bbr` command.
CPU	• x86-64-v3 Microarchitecture (or higher) • Advanced Encryption Standard (AES) New Instructions • Advanced Vector Extensions (AVX) • Carry-Less Multiplication (PCLMUL) **Tip** • To allow Qumulo Core to configure your CPU for optimal performance, install the `cpupower` and `x86_energy_perf_policy` utilities. • Utilities are built from specific versions of the kernel source tree. For information about the packages that you can use to install these utilities, see the documentation for your Linux distribution.

Memory	• Minimum: 3 GB • Recommended: 128 GB (or more) for hybrid and all-flash nodes • ECC DIMMs **Tip** • Individual CPU core and data drive caches, as well as the number of CPU cores and number and size of data drives, can affect the memory requirements. For information about recommendations for performance benchmarking, see Qumulo Supported Platforms on the Qumulo Terms Hub. • Because the Debian kernel in the official AMI is compiled with the `CONFIG_EDAC` kernel configuration option disabled, any ECC memory errors aren't present in `sysfs` or the OpenMetrics API specification (page 369). In addition, your system doesn't reboot automatically if an uncorrectable ECC error occurs.
Data Drives	• Hybrid nodes must have at least 2 SSDs and 4 HDDs **Note** The number of HDDs must divide evenly by the number of SSDs. • All-flash systems must have at least 2 SSDs. • All data drives must be unpartitioned.

Step 2: Prepare the Host Operating System

Before configuring your host operating system (OS) for installing the Qumulo Core Product Package, certain Linux distributions require some preliminary configuration.

Note
Ubuntu doesn't require any additional configuration. You can now configure the host OS (page 9).

To Prepare a Rocky or Red Hat Enterprise Linux (RHEL) 9.3 System

- To enable the `io_uring` kernel interface, use the `sysctl` command. For example: `sudo sysctl -w kernel.io_uring_disabled=0`

- Set SELinux to permissive mode. For more information, see Changing SELinux States and Modes in the Red Hat Documentation.

To Prepare a Debian 11 System

- To ensure that you have the features that the Qumulo Core Product Package requires, update the 5.10 kernel that Debian 11 ships with to a newer kernel, such as 6.1.0 (and higher). For more information, see How to Upgrade the Linux Kernel in the Debian documentation.
- Configure `systemd-networkd` and `systemd-resolved`. For more information, see Setting Up `systemd-networkd` and Using `systemd-resolved` for DNS resolution in the Debian documentation.

Step 3: Configure the Host Operating System

Before installing the Qumulo Core Product Package, you must configure your host operating system.

- To provide the `systemd-nspawn` tool for the Qumulo Core container, install the `systemd-container` package.
- To ensure that Qumulo Core has full control over network configuration, disable or remove the default network configuration tool (such as ENI, netplan.io, NetworkManager and so on).
- Ensure that `systemd-networkd` and `systemd-resolved` are installed and enabled. Qumulo Core uses these services, as well as the configuration files located in the `/etc/systemd/network` directory, to perform network configuration.

To Install the Qumulo Core Product Package

> **Note**
>
> - The Qumulo Core Product Package is currently in Private Availability. To obtain a copy, contact the Qumulo Care team.
> - Use the latest, official image available for your Linux distribution on the distribution's website or cloud marketplace.

1. To install Qumulo Core, use the correct installation package for your Linux distribution:

- For the `.deb` package, run the `dpkg` command. For example:

```
sudo dpkg -i qumulo-core.deb
```

- For the `.rpm` package, run the `rpm` command. For example:

```
sudo rpm -i qumulo-core.rpm
```

2. To verify that the Qumulo Core service has started successfully, run the `systemctl` command. For example:

```
sudo systemctl status qumulo-qcore.service
```

Upgrading the Qumulo Core Product Package

> ⚠ Important
>
> - Don't attempt to upgrade the Qumulo Core Produce Package by installing a newer `.deb` or `.rpm` package on your system.
> - The Qumulo Core Product Package doesn't have separate versions for on-premises and cloud platforms. You must use the on-premises upgrade image for upgrading both on-premises and cloud clusters.
> - You must upgrade all the nodes in your cluster at the same time, by using the Web UI, Qumulo REST API, or the `qq` CLI. The Qumulo Core Product Package doesn't support upgrading individual nodes and attempting to do so takes the cluster offline.

Qumulo Core Product Package upgrades follow the standard Qumulo Core upgrade process (page 44). All Qumulo Core Product Package upgrades are instant software upgrades (page 0) and never require a node to reboot (however, you can use the the Qumulo REST API or `qq reboot_start` command to perform a manual or rolling reboot).

Troubleshooting the Qumulo Core Product Package

Depending on the issue, you can troubleshoot the Qumulo Core Product Package from the host operating system or from within the Qumulo container. The following sections outline common scenarios and some potential troubleshooting approaches to them.

From the Host Operating System

Scenario	Troubleshooting Approach

Examine a container that starts successfully	To get shell access to the container, run the following command: `machinectl shell qcore`
Examine a container that doesn't start successfully	To check the logs for the container service, run the following command: `sudo journalctl -u qumulo-qcore.service`
View QFSD logs	To view the logs within the container, run the following command: `journalctl -M qcore -u qumulo-qfsd.service`
Use the `qsh` c tool	To be use the `qsh` tool, run the following command: `export PATH=$PATH:/opt/qumulo/bin/qsh` Alternatively, add the path to your shell configuration file.

From within the Container

Scenario	Troubleshooting Approach
Examine a container that is unable to create a cluster	To check the QFSD logs, run the following command: `journalctl -u qumulo-qfsd`

Creating and Managing Directory Quotas in Qumulo Core

This section explains how to create, modify, and delete directory quotas by using the Qumulo Core Web UI and how to use the Cluster Alerts for Qumulo script to manage cluster quota notifications.

To Create a Directory Quota

1. Log in to the Qumulo Core Web UI.
2. Click **Sharing** > **Quotas**.
3. On the right side of the **Storage Quotas** page, click **Create Quota**.
4. In the **Create Quota** dialog box:
    a. Enter the **Path** to the directory to which to add a quota.
    b. Enter the quota **Limit** and enter the units.
    c. Click **Save**.

To Modify a Directory Quota

1. Log in to the Qumulo Core Web UI.
2. Click **Sharing** > **Quotas**.
3. For a storage quota, in the **Actions** column, click .
4. In the **Edit Quota** dialog box, change the quota limit and click **Save**.

To Delete a Directory Quota

1. Log in to the Qumulo Core Web UI.
2. Click **Sharing** > **Quotas**.
3. For a storage quota, in the **Actions** column, click .
4. In the **Delete quota for path?** dialog box, click **Yes, Delete**.

Configuring Email Notifications for Cluster Quotas

For information about configuring email notifications for your cluster's quotas, see Cluster Alerts for Qumulo on GitHub.

For an example configuration, see `example_config.json`.

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 • FTPS • NFSv3 • NFSv4.1 (page 231) • S3 API (page 290) • SMB 2.002 • SMB 2.1 • SMB 3.0 • SMB 3.1 • SMB 3.1.1
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 20.04 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)

Supported Switches

Qumulo Core requires switches that meet the following criteria:

- Enterprise-grade
- Fully non-blocking
- Managed
- Supports IPv6

Known Limits

Limit Type	Maximum Value
On-Premises Cluster Size	265 nodes
Cloud Cluster Size	100 nodes
NFS Exports	64,000
SMB Shares	40,000
Access Control Entries (ACEs) in an Access Control List (ACL)	200

Limit Type	Maximum Value
NFS Groups	16, when not using LDAP or Active Directory for RFC 2307 attributes
Characters in Cluster Name	2-15, alphanumeric and hyphen (-)
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
Total Files	18 quintillion
Hard Links for Each File	1,024
LDAP Domains	1
Active Directory Domains	1
DNS Servers	3
Snapshots	40,000
Quotas	4.3 billion **Note** This approximate value of 2^{32} is equivalent to the possible maximum of directories or the entire inode space.
S3 Bucket Object Versions	Unlimited (4,294,967,296 theoretical)

Limit Type	Maximum Value
Total 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.
Usable Provisioned Capacity on Cluster	100%

System Upgrades

Qumulo Core Feature Log

This section lists the most important features from each release.

For information about upgrade types for each release, see Qumulo Core Upgrade Mode Reference (page 40).

Qumulo Core 7.2.1

Downloads and Release Notes

- Released the Qumulo Core Product Package in `.deb` and `.rpm` formats
- Allowed Object Lock to be enabled to for newly created S3 buckets
- Added support for resuming TLS sessions over FTP
- Made improvements in SMB session handling
- Made improvements in audit logging

Qumulo Core 7.2.0.1 (Quarterly)

Downloads and Release Notes

> ⚠ Important
> This release resolves an issue on HPE platforms by maximizing the reuse of existing REST client connections and by restarting the iLO monitor regularly. Previously, REST connections within the iLO monitor consumed all available memory after running for approximately longer than one week.

- The majority of functionality and stability improvements in this Qumulo Core release are internal.
- Resolved an issue with the **APIs & Tools** page in the Web UI not showing certain REST endpoints correctly

Qumulo Core 7.1.4

Downloads and Release Notes

- The majority of functionality and stability improvements in this Qumulo Core release are internal.
- Fixed issues with the default (empty) policy for S3 buckets
- Configured Qumulo Core to permit anonymous users to perform read-only API actions on

S3 buckets

Qumulo Core 7.1.3

Downloads and Release Notes

Extended S3 object versioning support to additional API actions

Qumulo Core 7.1.2

Downloads and Release Notes

- Added Amazon S3 object versioning
- Improved the prefetcher's performance
- Disabled automatic NIC firmware updates during platform upgrades

Qumulo Core 7.1.1

Downloads and Release Notes

- The majority of functionality and stability improvements in this Qumulo Core release are internal.
- Resolved two issues that could potentially cause a crash

Qumulo Core 7.1.0.1 (Quarterly)

Downloads and Release Notes

> ⚠ Important
> This release resolves an issue in Qumulo Core 7.1.0 that prevented Broadcom NIC firmware from being updated correctly.

- Improved NFSv4.1 performance for writes and reads
- Configured Qumulo Core to move a minimal number of IP addresses
- Configured S3 and HTTP servers to accept HTTP headers with empty values

Qumulo Core 7.0.2

Downloads and Release Notes

- Added the ability for the system to explain user access that an S3 bucket access policy allows
- Made changes to the REST API and `qq` CLI to support this functionality

Qumulo Core 7.0.1.1

Downloads and Release Notes

> ⚠ Important
> This release resolves a scenario in Qumulo Core 7.0.1 where a node in a Qumulo cluster could go offline if a tree delete operation was performed on a directory while an SMB client was watching it for changes.

- Configured rolling reboots to restart `N-1` nodes at a time, where `N` is the number of nodes configured for the cluster's fault tolerance
- Configured Shift-To S3 and Shift-From S3 to replicate file system user metadata.
- Added support for S3 bucket policies and the ability to configure, retrieve, and delete policies for S3 buckets
- Further improved NFSv3 random small-write performance
- Added support for Active Directory Global Catalog for look-ups
- Added support for `DeleteBucketPolicy` and `PutBucketPolicy` S3 API actions
- Configured RBAC privileges to override bucket policies which deny a user access to API actions through the S3 API
- Improved REST API and `qq` CLI operations for working with S3 buckets
- Fixed an issue with custom LDAP schemas

Qumulo Core 7.0.0.1 (Quarterly)

Downloads and Release Notes

> ⚠ Important
> To install Qumulo Core on HPE Alletra 4110 platforms, you must use the 7.0.0.1 release.

- Added support for NFSv4.1 cross-connection write combining
- Made significant improvements in SMB random small-write performance
- Added partial support for new S3 API actions
- Configured Qumulo Core to take daily snapshots at regular intervals throughout the day
- Made changes to the Qumulo Core REST API and the `qq` CLI to accommodate improvements in snapshot and at-rest encryption functionality
- Revised permissions for SMB `copychunk` requests

- Resolved a minor issue with AD authentication, an issue with `.snapshot` directories in Windows Command Prompt, and an issue with SMB copy operations and long share names and volume labels

Qumulo Core 6.3.2

Downloads and Release Notes

- Added support for additional custom metadata operations to the S3 API
- Configured Qumulo Core to replicate user metadata
- Added support for counting metadata updates for a file
- Configured OpenMetrics API metrics for protocol operations to track file system REST API operations
- Fixed an issue with cluster authentication for Kerberos client principals with names that contain the `@` character

Qumulo Core 6.3.1.1

Downloads and Release Notes

> ⚠ Important
> This release resolves an issue with memory utilization during secure credential handling.

- Added support for storing and retrieving custom metadata and tags by using the S3 API and Qumulo Core REST API and made supporting changes to the API and `qq` CLI
- Added the optional `data_revision` field for all REST resources that return file or stream attributes
- Resolved an issue with a third-party GSS authentication library

Qumulo Core 6.3.0.1 (Quarterly)

Downloads and Release Notes

> ⚠ Important
>
> - This release resolves an issue with memory utilization during secure credential handling.
> - This release resolves an issue in Qumulo 6.2.1.1 that interrupted the functioning of cross-realm Kerberos authentication when the system returns CNAMEs for PTR records while performing reverse DNS look-ups.

- Configured nodes to remain powered on after Transparent Platform Refresh operations
- Changed the behavior of static IP addresses when a cluster's networking configuration changes or nodes are replaced
- Improved the Qumulo Core Web UI by making accessibility improvements, renaming headings on the **Dashboard** page, and added physical drive bay locations to the **Node Details** page for HPE Alletra 4110 and Quiver 1U All-NVMe Gen1 node types
- Improved compatibility between Qumulo Core and the Auth0 SSO provider, increased the replication speed for large numbers of deleted files, updated the `smartpqi` driver for the HPE Apollo 4200 Gen10 and HPE Apollo 4200 Gen10 Plus node types, and resolved an issue with misreported temperatures for AMD CPUs

Qumulo Core 6.2.2.2

Downloads and Release Notes

> ⚠ Important
>
> - This release resolves an issue with memory utilization during secure credential handling.
> - This release resolves an issue in Qumulo 6.2.1.1 that interrupted the functioning of cross-realm Kerberos authentication when the system returns CNAMEs for PTR records while performing reverse DNS look-ups.
> - This release resolves an issue with caching NFS hostnames from the NFS allowed list.

- Added the ability to store master keys for at-rest encryption in an external Key Management Server (KMS) and updated the corresponding REST APIs and `qq` CLI commands
- Added the ability to dismiss errors and warnings on the **Cluster** > **Active Directory** page in the Qumulo Core Web UI by clicking **X**
- Configured the drive LEDs for the front (external) storage bays on Quiver 1U All-NVMe Gen 1 platforms to light up red if a drive fails
- Ensured that certain SMB `qq` CLI commands always output valid JSON

Qumulo Core 6.2.1.2

Downloads and Release Notes

⚠ Important

- This release resolves an issue in Qumulo 6.2.1.1 that interrupted the functioning of cross-realm Kerberos authentication when the system returns CNAMEs for PTR records while performing reverse DNS look-ups.
- This release resolves an issue in Qumulo Core 6.2.1 that affected the persistence of cluster network settings configured by using the Qumulo Core Web UI.
- This release removes the flag `--dns-config-id` that was added in Qumulo Core 6.2.1. This flag prevents the normal functioning of the `qq fs_list_locks` command.
- This release resolves an issue that we have identified in Qumulo Core 6.1.0 (and higher), where a potential session impersonation exploit was possible when a client accesses a Qumulo cluster by using the SMB protocol.

- Configured Qumulo Core 6.0.0 (and higher) to have replication compatibility with all quarterly and non-quarterly releases up to 8 quarters in the future (previously, compatibility was for 2 quarters)
- Improved the functionality of S3 `ListObjects`, `ListObjectsV2`, and `PutObject` API actions
- Added support for the SMB `ATTR_OFFLINE` extended file attribute
- Replaced deprecated REST API and `qq` CLI command pertaining to authentication
- Deprecated and replaced REST API command pertaining to the UID light and PSU information

Qumulo Core 6.2.0.1 (Quarterly)

Downloads and Release Notes

⚠ Important

- The removal of the deprecated `/v1/smb/shares` REST API endpoints in this release can affect certain third-party backup or migration workflows.
- This release resolves an issue with Qumulo Core's ability to route return packets back to clients when the following conditions are true for a cluster:
    - The cluster has a single tenant
    - VLAN networks are configured
    - Packets are destined for specific VLANs without a configured gateway

  After upgrading to Qumulo Core 6.2.0.1, you can continue to use the default gateway to route VLAN-specific packets.

- Added the ability to specify hostnames in host access rules for NFS exports
- Added the privilege `FILE_READ_ACCESS` to Qumulo Core for roles that require read-only access
- Improved the snapshot deletion process
- Added information about drive bays and disk serial numbers for Quiver 1U All-NVMe Gen1 and virtual machine nodes
- Disabled the automatic addition of DNS records to an AD domain when you join the domain

Qumulo Core 6.1.2.2

Downloads and Release Notes

⚠ Important

- This release resolves:
  - An issue that affects parsing Kerberos principal names that contain special characters in Active Directory Kerberos authentication in Qumulo Core 6.1.2. This issue doesn't prevent access to SMB shares by using NTLM-based authentication.
  - An issue that affects upgrading Qumulo Core on the Quiver 1U Hybrid Gen2 platform in Qumulo Core 6.1.1.
  - An issue that affects getting a useful core dump in Qumulo Core 6.1.1.
- We've identified a known issue that affects event processing in the Cloud-Based Monitoring (CBM) service Qumulo Core 6.1.0.3. This issue is resolved in Qumulo Core 6.2.0.1.

- Added Adaptive Data Protection to the Qumulo Core REST API and `qq` CLI and made performance improvements
- Added Transparent Platform Refresh to the Qumulo Core REST API and `qq` CLI
- Added S3 configuration settings to the Qumulo Core Web UI
- Improved hard link creation

Qumulo Core 6.1.1

Downloads and Release Notes

- Added support for new S3 API actions, snapshot locking for replication target relationships, and lock configuration to the Qumulo Core Web UI
- Enabled the Search Trusted Domains Active Directory configuration option by default
- Upgraded the host and container images
- Revised the Qumulo Core REST API for snapshots
- Added `qq` CLI commands for lock key functionality
- Added a `qq` CLI command for viewing blocked upgrade status
- Updated the OpenMetrics API Specification
- Made general improvements to Qumulo Core

Qumulo Core 6.1.0.3 (Quarterly)

Downloads and Release Notes

⚠ Important
This release resolves an issue where, under certain circumstances, while using Adaptive Data Protection in Qumulo Core 6.1.0 (and lower), it becomes impossible to add nodes to a cluster during normal cluster expansion or Transparent Platform Refresh operations, until you upgrade the cluster to version 6.1.0.3 (or higher).

- Configured Qumulo Core to use recursive change notifications by default
- Enabled SMB multichannel by default
- Added snapshot locking functionality by using the `qq` CLI
- Made usability changes to event logging and the Qumulo Core Web UI
- Removed beta multitenancy REST APIs and `qq` CLI commands

Qumulo Core 6.0.2

Downloads and Release Notes

⚠ Important
The removal of the deprecated `/v1/smb/shares` REST API endpoints in this release can affect certain third-party backup or migration workflows.

- Added support for streaming file change notifications
- Improved the S3 API
- Made improvements to the Qumulo Core allocation algorithm
- Added audit logging and REST API and `qq` CLI commands for streaming file change notifications
- Removed v2 Python bindings for NFS export and SMB share APIs
- Added new fields to the OpenMetrics API
- Fixed issues with creating SMB shares, transparent platform refresh, vSphere clusters, upgrades in the Qumulo Core Web UI, making SMB2 copies by using ODX, and creating snapshot policies

Qumulo Core 6.0.1.1

Downloads and Release Notes

- Added new SMB2 `CHANGE_NOTIFY` functionality
    - Full support for recursive watch-tree notifications

    - Alternate Data Streams
    - Numerous performance improvements
    - Elimination of missed events
- Added read combining for NFSv4.1
- Improved the S3 API
    - S3 API action traversal rights
    - S3 traffic in the Qumulo Core Web UI and in `qq` CLI output
    - Presigned URL functionality
    - HTTP identity transfer encoding for the `GetObject` API action
- Added operational information to audit logging in JSON format
- Made numerous updates to the multitenancy and S3 `qq` CLI commands and REST APIs
- Fixed an issue with Network Interface Card (NIC) addressing on 240TB and 480TB General Purpose models of the HPE Apollo 4200 Gen10 Plus node

Qumulo Core 6.0.0.2 (Quarterly)

Downloads and Release Notes

> **Important**
>
> - This release resolves an issue with transparent platform refresh that we discovered in Qumulo Core 6.0.0 and 6.0.0.1.
> - This release restores the status of the `/v1/smb/shares` REST API endpoints (previously removed in Qumulo Core 5.3.4) to deprecated. We strongly recommend changing over to `/v2/smb/shares` endpoints as soon as possible.

- Added Transparent Platform Refresh for node replacement
- Enabled concurrent node reboots for rolling upgrades
- Added write combining for NFSv4.1
- Introduced optional HTTP mode for the S3 API
- Made SSL certificate changes for the S3 API without a cluster restart
- Added audit syslog in JSON format
- Made corresponding changes and additions to the Qumulo Core REST API and `qq` CLI
- Fixed issues with the Qumulo Core Web UI, directory access by non-root users on Linux,

and changed caching for some OpenMetrics API metrics

Qumulo Core 5.3.4

Downloads and Release Notes

> **Important**
> The removal of the deprecated `/v1/smb/shares` REST API endpoints in this release can affect certain third-party backup or migration workflows.

- Added data protection configuration, multitenancy, tenant-aware SMB and NFS, and automatic clean-up of stale multipart S3 uploads
- Removed all REST API endpoints under `/v1/smb/shares`
- Improved and added REST APIs and `qq` CLI commands for restriper, multitenancy, and tenant-aware SMB and NFS operations
- Improved the OpenMetrics APIs
- Fixed an issue with SMB2 attributes for symlinks and a potential security issue with REST API session tokens

Qumulo Core 5.3.3.1

Downloads and Release Notes

> **Note**
> This release resolves an issue that we discovered in Qumulo Core 5.3.3.

- Put out the initial release of the S3 Protocol Server
- Added support for NFSv4.1 with KRB5i
- Added the ability to disable access tokens
- Removed `/v1/session/kv` REST API endpoints
- Fixed an issue with listing trustees for a role and an issue with LDAP connections

Qumulo Core 5.3.2

Downloads and Release Notes

- Added distributed multi-AZ clustering to Qumulo clusters on AWS
- Added SMB2 `CHANGE_NOTIFY` parity with Windows
- New Qumulo OpenMetrics API metrics

- Added expiration time for access tokens
- Improved the Qumulo Core REST API

Qumulo Core 5.3.1

Downloads and Release Notes

- Added support for watching for changes in file attributes and directory entries with SMB2 `CHANGE_NOTIFY` filters
- Updated all firmware for all variants of Mellanox CX5 and CX6 and Broadcom NICs
- Added metrics to the Qumulo OpenMetrics API
- Added two-factor authentication for qq-CLI-initiated SAML SSO workflows
- Improved REST API and `qq` CLI
- Added a missing library dependency that allows SAML SSO to work on cloud platforms

Qumulo Core 5.3.0 (Quarterly)

Downloads and Release Notes

- Added interactive SAML SSO login and SSO for Active Directory users who manage clusters
- Identified a known issue with SAML SSO on cloud platforms
- Improved SMB `CHANGE_NOTIFY` requests
- Released the Qumulo OpenMetrics API
- Released the Access Tokens API
- Improved the Qumulo Core Web UI
- Added REST APIs for node identification
- Improved `qq` CLI commands

Qumulo Core 5.2.5.1

Downloads and Release Notes

> **Note**
> This release resolves an issue that we discovered in Qumulo Core 5.2.5.

- Allowed clusters joined to Active Directory to use SAML 2.0 SSO
- Added support for displaying NFSv4.1 quotas

- Created a more balanced usage of free space on clusters after the addition of nodes
- Improved the Qumulo Core Web UI
- Added a flag to a `qq` CLI command
- Fixed issues related to quotas and Alternative Data Streams

Qumulo Core 5.2.4

Downloads and Release Notes

The majority of functionality and stability improvements in this Qumulo Core release are internal.

Qumulo Core 5.2.3

Downloads and Release Notes

The majority of functionality and stability improvements in this Qumulo Core release are internal.

Qumulo Core 5.2.2

Downloads and Release Notes

- Made improvements to the Qumulo Core Web UI on the NFS Exports and Role Management pages
- Adjusted the Qumulo Core REST API endpoint `/v1/smb/sessions` and the corresponding `qq` CLI command
- Fixed an issue that occurs during NFSv4.1 failover
- Fixed an issue that prevents a cluster from forming due to an incorrect HDD-SSD ratio.

Qumulo Core 5.2.1

Downloads and Release Notes

- Added over-the-wire encryption to NFSv4.1 with Kerberos
- Improved performance for SMB Find requests
- Added an optional Web UI login banner
- Removed `/v1/upgrade/` endpoints
- Fixed an issue on the NFS Exports page.

Qumulo Core 5.2.0.2 (Quarterly)

Downloads and Release Notes

- Enabled local trustees for NFSv4.1 with Kerberos

- Added support for Amazon EC2 M6i instances and Amazon EBS gp3 volumes
- Removed and replaced REST API endpoints related to shutting down and stopping clusters and nodes
- Fixed an SMB client issue
- Resolved a file system operations latency issue with continuous replication jobs and on clusters with high file creation load
- Resolved an issue that caused a node to enter a reboot loop under certain conditions.

Qumulo Core 5.1.5

Downloads and Release Notes

- Enabled creating mounts by using NFSv4.1 with Kerberos
- Improved NLM and NFSv4.1 locking
- Updated Mellanox NIC firmware
- Made improvements to the Qumulo Core Web UI on the **APIs & Tools**, **Add Nodes**, and **Active Directory** pages
- Added NFS fields to the Qumulo Core REST API and `qq` CLI
- Deprecated the `/v2/upgrade/status` REST API
- Updated security permissions for home directories
- Prohibited the use of the switch-user command
- Added support for user principal (UPN) names

Qumulo Core 5.1.4.1

Downloads and Release Notes

- Improved workload read performance for streaming
- Deprecated the `/v1/session/kv/` REST API endpoints
- Fixed an issue on the FTP Settings page
- Fixed an issue where unexpected behavior from an NFS client can cause issues with memory usage on a Qumulo cluster.

Qumulo Core 5.1.3

Downloads and Release Notes

- Added NFSv4.1 byte-range locking support for clusters that have NFSv4.1 enabled

- Added custom naming functionality for snapshots
- Made improvements to the Qumulo Core Web UI
- Improved node fault tolerance for node-adds
- Updated the time synchronization for cloud clusters
- Added a REST API to calculate node-add capacity
- Fixed issues related to adding S3 privileges, making `OPEN` requests over NFSv4.1, and reassigning floating IP addresses.

Qumulo Core 5.1.2

Downloads and Release Notes

- Improved user experience for NFSv4.1 ACL editor
- Made improvements to the Qumulo Core Web UI
- Updated time synchronization APIs for QaaS Azure clusters
- Fixed issues related to malformed `RPCBIND` requests and replication relationships.

Qumulo Core 5.1.1

Downloads and Release Notes

- Improved time management on QaaS Azure clusters
- Improved resiliency against data center accidents
- Made improvements to the Qumulo Core Web UI
- Updated the SDK to Python 3.8
- Disabled time management APIs on QaaS Azure clusters
- Fixed SMB and Web UI form issues

Qumulo Core 5.1.0.1 (Quarterly)

Downloads and Release Notes

- Improved write performance for workloads on clusters with high-churn, small-file datasets
- Added a new Active Directory capability
- Made a number of pages in the Qumulo Core Web UI more clear
- Improved SMB and NFS share, export management, and status APIs
- Added inactivity timeout to the Qumulo Core Web UI

- Identified a regression in 5.0.5 and fixed an issue that occurs for rare NFS client applications.

Qumulo Core 5.0.6

Downloads and Release Notes

- Improved read performance for previously read large files
- Improved write performance for certain platforms and workloads
- Improved cluster resilience on Azure
- Added support for HPE Broadcom NICs and updated the NIC and SKU reference
- Made improvements to the Qumulo Core Web UI
- Fixed issues related to snapshots of directories with SMB-illegal characters and localized Active Directory installations.

Qumulo Core 5.0.5

Downloads and Release Notes

- Increased the tolerance of uncertainty for NTP
- Mapped Isilon-style and Samba-style NFS SIDs to Qumulo SIDs
- Made changes to the Python SDK
- Made an XSS security fix

Qumulo Core 5.0.4

Downloads and Release Notes

- Enabled advanced data integrity for new clusters
- Added a check to confirm whether an upgrade can begin
- Added an initial release of the S3 protocol
- Deprecated reboot `qq` CLI commands
- Added access time (`atime`) API resources and `qq` CLI commands.

Qumulo Core 5.0.3

Downloads and Release Notes

- Added Rolling Reboot for platform upgrades (including new APIs and CLI commands)
- Updated the Qumulo Core Web UI for both software and platform upgrades
- Made improvements to write performance over NFSv3

- Improved the **Add Nodes** and **Snapshots** pages in the Qumulo Core Web UI

Qumulo Core 5.0.2

Downloads and Release Notes

- Changed all HPE firmware updates to be performed using the Qumulo installer
- Improved cluster resilience against boot drive failures
- Added the ability to set timestamps through NFSv4.1
- Added the `access_time` field to the attributes REST API resource.

Qumulo Core 5.0.1

Downloads and Release Notes

- Enforced the Qumulo-to-Qumulo replication compatibility guarantee to 3 consecutive quarterly releases

  > **Tip**
  > For example, 5.0.1 can't replicate with versions lower than 4.3.0 or higher than 5.2.0.

- Made improvements to Shift-to-S3
- Removed unconfigured node APIs

Qumulo Core 5.0.0.1 (Quarterly)

Downloads and Release Notes

- Changed the Qumulo-to-Qumulo replication compatibility guarantee from 2 to 3 consecutive quarterly releases
- Enforced Kerberos ticket expiration
- Enabled NLM persistence by default
- Added sorting and filtering to the Snapshot Policies page
- Improved speed of deleted snapshot clean-up
- Removed requirements for QaaS clusters
- Added support for Python 3.8
- Improved the API and CLI version and user-querying command
- Resolved an issue that causes an unpredictable node failure if two NFSv4.1 clients have the same hostname.

Qumulo Core 4.3.4

Downloads and Release Notes

> ⚠ Important
> This release of Qumulo Core resolves a critical security vulnerability. We strongly recommend upgrading your Qumulo cluster to 4.3.4 to resolve this issue.

- Added work estimation to Shift-from-S3 relationships
- Added a feature that improves cluster resiliency for unresponsive devices
- Changed the kiosk timeout window to 15 minutes

Qumulo Core 4.3.3

Downloads and Release Notes

- Prevented authentication requests with PAC buffers of unknown type from failing
- Improved Saved Snapshots page in the Qumulo Core Web UI
- Added a mechanism for monitoring status of front-end network on platforms with separate interfaces.

Qumulo Core 4.3.2

Downloads and Release Notes

Fixed an issue in Qumulo Core's handling of cross-protocol permissions

Qumulo Core 4.3.1

Downloads and Release Notes

- Improved write performance
- Improved speed and duration of terminating outstanding requests to AD Controllers
- Added file system permission checks for SMB shares and NFS exports
- Enabled mounting NFS over UDP with floating IP addresses

Qumulo Core 4.3.0 (Quarterly)

Downloads and Release Notes

- Added Network File System Version 4.1 (NFSv4.1)
- Improved performance for SMB offloaded data transfer (ODX) and SMB directory query
- Allowed AWS Sidecar to handle a new class of EBS failures

- Fixed issues in Shift-From-S3 and the Qumulo Core Web UI

Qumulo Core 4.2.6

Downloads and Release Notes

- Made an enhancement to Web UI for Qumulo on Azure
- Adjusted a REST resource
- Made an important change to export names that end with a slash (`/`) or are longer than 64 KiB

Qumulo Core 4.2.5

Downloads and Release Notes

Added Web UI management support for Qumulo Shift-from-S3 operations

Qumulo Core 4.2.4

Downloads and Release Notes

- Enhanced tree-delete operations
- Made performance improvement for C-72T, C-168T, C-192T, and C-432T platforms
- Enhanced Shift-to-S3
- Made performance improvements
- Fixed several issues

Qumulo Core 4.2.3

Downloads and Release Notes

- Added Qumulo Shift-from-S3
- Web UI Update for SMB Shares
- Enabled DNS configuration for cloud deployments in the Qumulo Core Web UI

Qumulo Core 4.2.2

Downloads and Release Notes

- Improved the appearance of spinners in the Qumulo Core Web UI
- Fixed a failover procedure issue

Qumulo Core 4.2.1

Downloads and Release Notes

- Made improvements for security identifiers

  > **Important**
  > We removed this feature for compatibility reasons. For more information, see the release notes on Qumulo Nexus.

- Upgraded the Linux kernel to 5.4
- Improved multistream throughput for 4U nodes
- Updated `qq` CLI commands for unconfigured node cluster creation

Qumulo Core 4.2.0 (Quarterly)

Downloads and Release Notes

- Enhanced snapshot performance
- Enhanced Active Directory connections
- Fixed an important issue

Qumulo Core 4.1.5

Downloads and Release Notes

- Added details to the **Node Add** page in the Qumulo Core Web UI
- Updated `qq` CLI upgrade commands for unconfigured nodes
- Made improvements to snapshot cleanup
- Fixed an important issue

Qumulo Core 4.1.4

Downloads and Release Notes

- Updated user privileges for the quota `qq` CLI commands
- Enhanced autocomplete functionality in the Qumulo Core Web UI
- Fixed several important issues

Qumulo Core 4.1.3

Downloads and Release Notes

The majority of functionality and stability improvements in this Qumulo Core release are internal.

Qumulo Core 4.1.2

Downloads and Release Notes

- Enhanced security of `qq` CLI commands
- Fixed several important issues

Qumulo Core 4.1.1

Downloads and Release Notes

- Added the ability to repeat completed jobs for Shift
- Enhanced the security flag for SMB connections
- Improved directory-caching performance
- Updated the HPE Field Verification Tool

Qumulo Core 4.1.0.1 (Quarterly)

Downloads and Release Notes

- Enhanced the upgrade experience
- Improved the Qumulo Sidecar for AWS
- Fixed several important issues

Qumulo Core 4.0.6

Downloads and Release Notes

Made a platform upgrade for cloud clusters

Qumulo Core 4.0.5

Downloads and Release Notes

- Updated the SMB API and CLI
- Fixed several important issues

Qumulo Core 4.0.4

Downloads and Release Notes

- Updated the firmware for C-72T, C-168T, C-192T, C-432T, K-144T, K-168T, and K-432T platforms
- Enhanced `qq` CLI commands
- Fixed an important issue related to mixed-node clusters

Qumulo Core 4.0.3

Downloads and Release Notes

- Enhanced the web server security
- Fixed an important issue related to mixed-node clusters

Qumulo Core 4.0.2

Downloads and Release Notes

- Improved the performance of SSD clusters
- Made minor performance enhancements
- Fixed an important issue in the Qumulo Core Web UI

Qumulo Core 4.0.1.1

Downloads and Release Notes

Fixed important issues in:

- Cloud-Based Monitoring
- SMB file enumeration
- Compound API and `qq` CLI request errors

Qumulo Core 4.0.0.2 (Quarterly)

Downloads and Release Notes

- Deprecated the Python 2.7 SDK
- Fixed several important issues

Qumulo Core 3.3.5

Downloads and Release Notes

- Updated mixed-node compatibility
- Made a performance improvement for Qumulo Shift
- Improved the Field Verification Tool for HPE Apollo 4200 Gen10 clusters
- Adjusted audit logging functionality
- Fixed several important issues

Qumulo Core 3.3.4

Downloads and Release Notes

- Added support for the HPE ProLiant DL325 Gen10 Plus platform
- Made general performance enhancements for hybrid clusters

Qumulo Core 3.3.3

Downloads and Release Notes

- Added Instant Software Upgrade
- Made Qumulo Shift performance improvements and Web UI enhancements
- Updated the HPE Service Pack Pro (SPP) Update for HPE Apollo 4200 Gen10 clusters

Qumulo Core Upgrade Mode Reference

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

- An *instant software upgrade* (page 44) requires restarting only the container on your nodes and has a downtime of less than 30 seconds without disruption to the operation of the cluster.
- A *platform upgrade* (page 45) 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 and cloud upgrades 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.

For downloads, release notes, and upgrade paths for on-premises and cloud releases of Qumulo Core, see Qumulo Nexus 🔒.

Qumulo Core Upgrade Modes

For information about the most important features from each release, click the Qumulo Core version.

Version	On-Premises Upgrade Type	Cloud Upgrade Type
7.2.1 (page 17)	Instant	
7.2.0.1 (Quarterly) (page 17)	Instant	
7.1.4 (page 17)	Instant	
7.1.3 (page 18)	Instant	
7.1.2 (page 18)	Instant	
7.1.1 (page 18)	Instant	
7.1.0.1 (Quarterly) (page 18)	Platform	

Version	On-Premises Upgrade Type	Cloud Upgrade Type
7.0.2 (page 18)	Instant	
7.0.1.1 (page 19)	Instant	
7.0.0.1 (Quarterly) (page 19)	Instant	
6.3.2 (page 20)	Instant	
6.3.1.1 (page 20)	Instant	
6.3.0.1 (Quarterly) (page 20)	Platform	
6.2.2.2 (page 21)	Instant	
6.2.1.2 (page 21)	Instant	
6.2.0.1 (Quarterly) (page 22)	Instant	
6.1.2.2 (page 23)	Platform	
6.1.1 (page 24)	Platform	
6.1.0.3 (Quarterly) (page 24)	Instant	
6.0.2 (page 25)	Instant	
6.0.1.1 (page 25)	Platform	
6.0.0.2 (Quarterly) (page 26)	Instant	
5.3.4 (page 27)	Instant	
5.3.3.1 (page 27)	Instant	
5.3.2 (page 27)	Instant	
5.3.1 (page 28)	Platform	Instant
5.3.0 (Quarterly) (page 28)	Instant	
5.2.5.1 (page 28)	Instant	
5.2.4 (page 29)	Instant	
5.2.3 (page 29)	Instant	

Version	On-Premises Upgrade Type	Cloud Upgrade Type
5.2.2 (page 29)	Instant	
5.2.1 (page 29)	Instant	
5.2.0.2 (Quarterly) (page 29)	Instant	
5.1.5 (page 30)	Platform	
5.1.4.1 (page 30)	Instant	
5.1.3 (page 30)	Instant	
5.1.2 (page 31)	Instant	
5.1.1 (page 31)	Platform	
5.1.0.1 (Quarterly) (page 31)	Instant	
5.0.6 (page 32)	Instant	
5.0.5 (page 32)	Instant	
5.0.4 (page 32)	Instant	
5.0.3 (page 32)	Instant	
5.0.2 (page 33)	Instant	
5.0.1 (page 33)	Instant	
5.0.0.1 (Quarterly) (page 33)	Instant	
4.3.4 (page 34)	Instant	
4.3.3 (page 34)	Instant	
4.3.2 (page 34)	Instant	
4.3.1 (page 34)	Instant	
4.3.0 (Quarterly) (page 34)	Instant	
4.2.6 (page 35)	Instant	
4.2.5 (page 35)	Instant	

Version	On-Premises Upgrade Type	Cloud Upgrade Type
4.2.4 (page 35)	Platform	
4.2.3 (page 35)	Instant	
4.2.2 (page 35)	Instant	
4.2.1 (page 35)	Platform	
4.2.0 (Quarterly) (page 36)	Instant	
4.1.5 (page 36)	Instant	
4.1.4 (page 36)	Instant	
4.1.3 (page 36)	Instant	
4.1.2 (page 37)	Instant	
4.1.1 (page 37)	Instant	
4.1.0.1 (Quarterly) (page 37)	Instant	
4.0.6 (page 37)	Instant	Platform
4.0.5 (page 37)	Instant	
4.0.4 (page 37)	Instant	
4.0.3 (page 38)	Instant	
4.0.2 (page 38)	Instant	
4.0.1.1 (page 38)	Instant	
4.0.0.2 (Quarterly) (page 38)	Instant	
3.3.5 (page 38)	Instant	
3.3.4 (page 38)	Instant	
3.3.3 (page 39)	Instant	

Performing Qumulo Core Upgrades

This section explains the difference between Qumulo Core Instant Software Upgrades and Platform Upgrades and how to perform upgrades.

> ⚠ 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 the Qumulo Care team for guidance.
> - If you don't see a rolling reboot option for a platform upgrade, refresh the page in your browser.

For downloads, release notes, and upgrade paths for on-premises and cloud releases of Qumulo Core, see Qumulo Nexus 🔒.

Understanding the Differences Between Upgrade Modes

For information about which upgrade modes different Qumulo Core releases use, see Qumulo Core Upgrade Mode Reference (page 40).

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 without disruption to the operation of the cluster.

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 with SMB and NFS client interruption) or—in Qumulo Core 5.0.3 (and higher)—a *rolling reboot* (rebooting the nodes in your cluster one at a time with impact only to SMB clients).

The reboot process differentiates platform upgrades from instant software upgrades, where your nodes maintain the Linux operating system (and certain services that Qumulo Core relies on) during the upgrade process and where 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 the Qumulo Care team 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 view your cluster's upgrade phase and status, run the `qq upgrade_status` command.

The following is example output.

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

Upgrading Your Qumulo Cluster

⚠ Important

- Before beginning the upgrade process, make sure that you have the correct cloud upgrade image🔒 or on-premises upgrade image🔒.
- To allow certain background processes to run, multiple sequential Qumulo Core upgrades might require a waiting period between specific releases. Before installing multiple Qumulo Core releases within an extended maintenance window, contact the Qumulo Care team.
- When the node to which you are connected reboots, the Qumulo Core Web UI might redirect you to the **Cluster is booting...** page. To view the reboot status, you can connect to a different node or wait until the current node goes online and then click **Support** > **Software Upgrade**.
- During rolling reboots, the system reboots one or more nodes (depending on the configured protection level) in the cluster in succession. If your cluster is under heavy load due to write or delete operations, this process can take a long time.

To Upgrade Your Qumulo Cluster by Using the Qumulo Core Web UI

1. Upload the `qumulo_core_x.x.x.qimg` upgrade file to any directory on your cluster by using a client protocol such as NFS or SMB.
2. Log in to the Qumulo Core Web UI.
3. Click **Support** > **Software Upgrade**.
4. On the **Software Upgrade** page, enter the file system path for the upgrade file without the leading slash, for example: `upgrade/qumulo_core_7.1.0.1.qimg`.
5. Click **Upgrade...**
6. Depending on the upgrade mode for your release (page 40), do one of the following:

- **Instant Software Upgrade:** In the **Ready to upgrade?** dialog box, confirm the current and new versions of Qumulo Core and then click **Start Upgrade**.

  On the **Software Upgrade** page, the upgrade progress is displayed.

- **Platform Upgrade:** In the **Ready to upgrade?** dialog box, choose one of the following modes and then click **Start Upgrade**.

  - **Complete reboot (with SMB and NFS client interruption):** You can't access your cluster during the reboot process.
  - **Rolling reboot (with SMB client impact):** You can access your cluster during reboot, but SMB connectivity to specific nodes is impacted.

    Qumulo Core prepares the upgrade then installs Qumulo software on your cluster.

    When the upgrade is complete, the message **You have successfully upgraded from Qumulo Core x.x.x to y.y.y** is displayed.

To Upgrade Your Qumulo Cluster by Using the qq CLI

> **⚠ Important**
>
> - For two-phase upgrades, there is no client downtime during the *prepare* phase, only during the commit phase.
> - You can disable the CLI progress bar by adding the `--no-monitor` flag to your command. If you do this, it isn't possible to cancel the upgrade process after it begins.
> - Qumulo Core reboots a number of nodes (depending on the protection level configured on your cluster) in succession. This requires the restriper job to finish before Qumulo Core can continue to the next node in the cluster. If your cluster is under heavy load from write or delete operations, this process can take a long time.

1. Upload the `qumulo_core_x.x.x.qimg` upgrade file to any directory on your cluster by using a client protocol such as NFS or SMB.
2. Use SSH to connect to an IP address of a node in your cluster.
3. To become the root user, run the `sudo -s` command.
4. Depending on the current Qumulo Core version on your cluster, do one of the following:

- Qumulo Core 5.1.0.1 (and Higher):

  - Immediate Upgrade with Rolling Reboot: Run the `qq upgrade_cluster` command. Use the `--rolling` flag to specify rolling reboot, and the `--path` flag to specify the path to the upgrade file. For example:

```
qq upgrade_cluster \
  --rolling \
  --path /qumulo_upgrade_x.x.x.qimg
```

  - Two-Phase Upgrade with Rolling Reboot: Run the `qq upgrade_cluster` command first with the `prepare` subcommand and then with the `commit` subcommand. Use the `--rolling` flag to specify rolling reboot, and the `--path` flag to specify the path to the upgrade file. For example:

```
qq upgrade_cluster prepare \
  --rolling \
  --path /qumulo_upgrade_x.x.x.qimg
```

```
qq upgrade_cluster commit
```

- Qumulo Core 5.0.6 (and Lower):

  - Immediate Upgrade: Run the `qq upgrade_cluster` command and use the `--path` flag to specify the path to the upgrade file. For example:

```
qq upgrade_cluster \
  --path /qumulo_core_x.x.x.qimg
```

  - Two-Phase Upgrade: Run the `qq upgrade_cluster` command first with the `prepare` subcommand and then with the `commit` subcommand. Use the `--path` flag to specify the path to the upgrade file. For example:

```
qq upgrade_cluster prepare \
  --path /qumulo_core_x.x.x.qimg
```

```
qq upgrade_cluster commit
```

5. To view your cluster's upgrade phase and status, run the `qq upgrade_status` command.

   When the CLI progress bar shows that the upgrade is complete, Qumulo Core restarts your cluster.

   > **Note**
   > You can disable the CLI progress bar by adding the `--no-monitor` flag to your command. If you do this, it isn't possible to cancel the upgrade process after it begins.

Replication Version Requirements and Upgrade Recommendations for Qumulo Core

This section explains the relationship between the version of Qumulo Core that a cluster runs and data replication between it and other clusters. It also gives recommendations for upgrading Qumulo Core in relation to data replication tasks.

The replication process creates a consistent point-in-time copy of data in a directory on a source cluster when Qumulo Core transfers the data to a directory on a target cluster. Because two clusters are required for the replication process, there are specific requirements for version of Qumulo Core that the two clusters must run.

Replication for Qumulo Core 6.0.0.x (and Higher)

For Qumulo Core 6.0.0.x (and higher), clusters that run different versions can replicate *with all quarterly and non-quarterly versions, up to eight quarters in the future.*

The following example shows a replication compatibility matrix for quarterly and non-quarterly version of Qumulo Core.

	6.0.0.x (q)	6.0.1	6.1.0 (q)	...	8.0.0 (q)	8.0.1	...	10.0.0 (q)	10.0.1
6.0.0.x (q)	✅	✅	✅		✅				
6.0.1	✅	✅	✅		✅				
6.1.0 (q)	✅	✅	✅		✅	✅			
...									
8.0.0 (q)	✅	✅	✅		✅	✅		✅	
8.0.1			✅		✅	✅		✅	
...									
10.0.0 (q)					✅	✅		✅	✅
10.0.1								✅	✅

The following example shows replication options for a cluster running the quarterly (future) 8.0.0 version.

6.0.0.x (q) ◄ 6.1.0 (q) ◄ ... ◄ 7.2.0 (q) ◄ 7.3.0 (q) ◄ **8.0.0 (q)** ► 8.1.0 (q) ► 8.2.0 (q) ► ... ► 9.3.0 (q) ► 10.0.0 (q)

The following example shows replication options for a cluster running the non-quarterly (future) 8.0.1 version.

6.1.0 (q) ◄ 6.1.0 (q) ◄ ... ◄ 7.2.0 (q) ◄ 7.3.0 (q) ◄ **8.0.1** ► 8.1.0 (q) ► 8.2.0 (q) ► ... ► 9.3.0 (q) ► 10.0.0 (q)

Note
This schema doesn't impact replication compatibility for versions lower than 6.0.0 that are still only compatible with a maximum of two quarterly versions.

Replication for Qumulo Core 5.0.1 to 6.0.0

From Qumulo Core 5.0.1 to 5.3.4, clusters that run different versions can replicate *between the current version and up to two previous or future quarterly versions.*

The following example shows replication options for a cluster running the quarterly 5.1.0 version.

4.3.0 (q) ◄ 5.0.0 (q) ◄ **5.1.0 (q)** ► 5.2.0 (q) ► 5.3.0 (q)

The following example shows replication options for a cluster running the non-quarterly 5.1.1 version

5.0.0 (q) ◄ 5.1.0 (q) ◄ **5.1.1** ► 5.2.0 (q) ► 5.3.0 (q)

Note

- From version 5.0.1, Qumulo Core blocks replication between unsupported versions. For example, version 5.0.1 can't replicate with versions before 4.3.0 or after 5.2.0.
- In this scenario, version 5.2.0 is a hard limit. Versions 5.2.1 (and higher) can't replicate with versions 5.0.1 (or lower).

Replication for Qumulo Core 2.11.0 to 5.0.0

From Qumulo Core 2.11.0 to 5.0.0, clusters that run different versions can replicate *between at least two consecutive quarterly versions*. For example:

	4.1.5	4.2.0 (q)	4.2.1
4.1.5	✅	✅	
4.2.0 (q)	✅	✅	✅
4.2.1		✅	✅

Upgrade Recommendations

Because the target directory of an existing replication relationship is read-only to all client access, use the following recommendations to upgrade a cluster whose file system root directory is the target of a replication relationship.

> **Note**
> Regardless of version, upgrading Qumulo Core while a replication job is running interrupts the job in progress. When both clusters return to states in which replication is possible, the replication job resumes.

If the Source Cluster is Available

To avoid having to resynchronize your clusters after you reconnect the replication relationship, we recommend the following steps.

1. Copy the upgrade image to a directory on the source cluster. (It isn't possible to copy it to the read-only target cluster).
2. Wait for the upgrade image to replicate to the target cluster.
3. Continue with the normal upgrade process.

If the Source Cluster is Unavailable or Another Issue Exists

> **Important**
> The following steps interrupt the replication process.

On clusters that run Qumulo Core 2.12.5 (and lower), this process requires additional time to resynchronize the clusters by running a full-tree scan after you reconnect the replication relationship.

1. To temporarily make the target relationship writable for the duration of the upgrade, use the **Make Target Writable** action in the Qumulo Core Web UI.

2. Wait for the upgrade to complete.
3. Use the **Reconnect Relationship** action in the Qumulo Core Web UI.

Authentication

Installing VPN Keys on a Qumulo Cluster

This section explains how to install VPN keys obtained from the Qumulo Care team 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.

> **Caution**
> Follow these steps only if a member of the Qumulo Care team instructs you to do so. Performing these steps incorrectly can cause network performance, connectivity, and data integrity issues. It can also expose your cluster to unauthorized access. For help with this task, contact the Qumulo Care team.

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:
  - `ep1.qumulo.com`
  - `api.missionq.qumulo.com`
  - `monitor.qumulo.com`
  - `api.nexus.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 Getting Started with the qq CLI (page 124).
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, run the {% include qq.html command="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, run 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, run 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, run the {% include qq.html command="get_vpn_keys" %} command. For example:

```
sudo qq get_vpn_keys
```

To Register Cluster with Cloud-Based Monitoring

1. To retrieve your cluster ID, run the `qq node_state_get` command.
2. Send the output of the command to Qumulo Care.
3. Use the Qumulo Core 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.

- To join your cluster to an Active Directory (AD) domain, log in to the Qumulo Core Web UI and click **Cluster > Active Directory**.

  > **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 name 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 57) (IdP).
    a. The SSO administrator uses the cluster's fully qualified domain name (FQDN) format for the service provider (page 57) (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 the IdP sign both the assertion and the entire SAML response.

    e. To configure the IdP to use an algorithm based on SHA-256 (certain SSO providers use older algorithms, such as SHA-1, by default), follow the instructions in your SSO provider's documentation.

> **Tip**
> Commonly, a `signatureAlgorithm` key is set to `rsa-sha256` and the `digestAlgorithm` key is set to `sha256` in the configuration file.

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
```

> **Note**
> The IdP SSO URL often contains a unique identifier for the SAML integration. We don't recommend using the same identifier on several clusters simultaneously.

- 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 run 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 run 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 Qumulo Core 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 57) or SP (page 57) initiates
- A workflow that the `qq` CLI initiates

Note

- Members of the built-in Administrators role always have access to the Qumulo Core Web UI.
- To allow other users to access the Qumulo Core Web UI, you must assign the built-in Observers role to individual users or to groups.
- 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 Qumulo Core Web UI by connecting to a node physically, the login page doesn't show doesn't show **Continue to SSO login** on the Qumulo Core Web UI login page, even if SSO is configured.

IdP-Initiated SSO Workflow

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 Qumulo Core Web UI logs the user in. Otherwise, the Qumulo Core 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 Qumulo Core Web UI login page.

   the Qumulo Core 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` 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 run the `qq login` command with your AD account password. Instead, you must run the `qq sso_login` command</a>.
> - 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 (TLS) 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 run the `qq saml_modify_settings` 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 Qumulo Core Web UI, REST API, and `qq` CLI.

Known Issues and Limitations

- Local users (the built-in `admin` user and any additional users) can always use their passwords to authenticate to the Qumulo Core 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 cluster, it isn't possible to log in to the **Interactive API documentation** section of the **APIs & Tools** page in the Qumulo Core Web UI.
- Qumulo Core doesn't support:
  - **SAML Single Logout (SLO):** We recommend clicking **Sign out** in the Qumulo Core 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 61).

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 Qumulo Core 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 254).

Configuring the Search Trusted Domains Option in Active Directory for a Qumulo Cluster

This section explains how to restrict the scope of LDAP queries by using the Search Trusted Domains configuration option for a Qumulo cluster joined to an Active Directory (AD) domain.

During normal AD domain operations, a Qumulo cluster often encounters *LDAP referrals* that indicate to the cluster in what other locations within an AD domain it might locate requested information. Often, these referrals are hints to other trusted AD domains which a cluster accesses through a Domain Trust, such as a Parent Domain Trust or an external Domain Trust.

Reducing Latency by Disabling Search Trusted Domains

In Qumulo Core 6.1.0.3 (and lower), to permit Qumulo clusters to follow LDAP referrals, the **Search Trusted Domains** configuration option is enabled by default.

In Qumulo Core 6.1.1 (and higher), to reduce the potential latency of AD domain operations that might trigger and follow LDAP referrals unnecessarily (particularly for large, complex AD environments with multiple Domain Trusts), you can disable the **Search Trusted Domains** configuration option.

Disabling this option might benefit your system if you can determine that all relevant user and group accounts—which you might expect to use POSIX attributes, logins with SAML Single Sign-On (SSO), or logins with NFS4.1 and Kerberos—are located entirely in the current domain.

Limitations of Disabling Search Trusted Domains

This section explains the limitations of disabling the **Search Trusted Domains** configuration option.

Trusted Domains Specified in the Base DN

The **Base DN** (Distinguished Name) configuration option specifies the path that limits LDAP queries. When you set the Base DN to the top-level domain or base path of a domain, LDAP searches span the entire domain's LDAP structure, including LDAP referrals to other domains that have a Trust with the currently joined domain.

Often, the Base DN configuration ensures that the system searches all Organizational Units (OUs) in the domain, for example when the Administrator team might not have control over the OUs that contain the user accounts to be retrieved. (This is common in a dynamic environment that an external team manages.)

Qumulo Core lets you configure multiple Base DNs by providing their paths in a semicolon-separated list that includes the paths of other trusted domains. This configuration permits the trusted domains to use POSIX attributes and SAML SSO logins.

> ⚠ Important
> Disabling Search Trusted Domains disregards any trusted domains specified in the Base DN.

Ignoring LDAP Referrals and Qumulo Core Authentication Processes

To decide whether your system should ignore LDAP referrals, consider the Qumulo Core authentication processes that this might affect.

Authentication Processes that Trigger LDAP Queries

- Identity mapping from NTFS to POSIX (SMB to NFS) by using the **Use Active Directory for POSIX attributes** AD configuration option
- SAML single sign-on (SSO) (page 57)
- NFSv4.1 and Kerberos (page 252)
- REST API access tokens (page 71)
- S3 access keys (page 295)

Unaffected Authentication Processes

- Kerberos SMB SSO logins from Domain Local or Trusted Domain users
- NTLMv2 SMB logins (username and password) from Domain Local or Trusted Domain users
- Domain Local groups that contain users and groups from other Trusted Domains
- Users or groups added to SMB share permissions by using the Qumulo Core Web UI or `qq` CLI
- Security Identifiers (SIDs) resolved to usernames by using client dialog boxes, for example in macOS Finder or Windows File Explorer

Configuring LDAP on your Qumulo Cluster

This section explains how to configure LDAP on Qumulo Core 2.12.1 (and higher).

To Configure LDAP by Using the Qumulo Core Web UI

1. Log in to the Qumulo Core Web UI..
2. Click **Cluster** > **LDAP**.
3. On the **LDAP Configuration** page, click **Edit** and then do the following:
    a. For **Use LDAP features**, click **Yes**.
    b. Enter the **LDAP URI**.
    c. For **Base DN**, enter the Distinguished Name (DN) from which the LDAP server searches for users.
    d. For **Bind Username**, enter the username for logging in to LDAP services.

    > **Note**
    > This username must have the permission to search within the provided DN.

    e. For **Bind Password**, enter the password that corresponds to the username.
    f. (Optional) For **Encrypt Connection**, click **Yes**.

    > **Important**
    >
    > By default, Qumulo requires an encrypted connection to connect to LDAP (either LDAPS or StartTLS). If you disable the option to connect without TLS, you might expose credentials over your network.
    >
    > To use encrypted connections, you must install a valid certificate for your LDAP server (page 164).

    g. Click **Save**.

Creating and Using Bearer Tokens to Authenticate Qumulo Core REST API Calls

This section explains how to create bearer tokens—by using the Qumulo Core REST API or Web UI—to authenticate Qumulo Core REST API calls.

When you use the Qumulo Core REST API, you begin an authentication session by logging in to a Qumulo cluster. Different REST endpoints require different types of authentication: For example, certain REST API endpoints, such as `/v1/version`, don't require any authentication, while the `/v1/session/login` API endpoint requires a username and a password.

Calling the login API gives you a *bearer token* (or *access token*)—a temporary credential that Qumulo Core sends together with subsequent API calls as proof of authentication. A bearer token is valid for 10 hours. After a bearer token expires, you must create a new bearer token.

Creating a Bearer Token

To create a bearer token, you can use the Qumulo Core REST API or Qumulo Core Web UI.

> ⚠ Important
> Only administrative users (or users with `PRIVILEGE_ACCESS_TOKENS_WRITE`) can create bearer tokens.

To Create a Bearer Token by Using the REST API

Begin an authentication session by calling the `/v1/session/login` REST API endpoint and specify the username and password. For example:

```
curl -k -X POST https://203.0.113.0:8000/v1/session/login \
  -H "Content-Type: application/json" \
  -d '{ "username": "Alice", "password": "rbFAYMdtGrwTAV4TR0cZ"}'
```

The following is example output.

```
{ "bearer_token": "1:EXAMPLElSnp6MVZvUXhRQUViN2RCYUFVZy9zTElB..." }
```

To Create a Bearer Token by Using the Qumulo Core Web UI

1. Log in to the Qumulo Core Web UI.
2. Click **APIs & Tools**.

3. Under **Interactive API documentation**, enter your username and password and click **Apply credentials**.

   The **Authentication succeeded.** message appears.

4. Expand the **Access Tokens** section and then the **POST /v1/auth/access-tokens/** section.

5. On the upper-right side, click **Try it out**, ensure that the **Request body** is correct, and then click **Execute**.

   The following is example output.

```
{ "bearer_token": "1:EXAMPLElSnp6MVZvUXhRQUViN2RCYUFVZy9zTElB..." }
```

To Use a Bearer Token to Authenticate a Qumulo Core REST API Call

Place the bearer token in your request header. In the following example, the API call lists the nodes in a cluster.

```
curl -k -X GET https://203.0.113.0:8000/v1/cluster/nodes/ \
  -H "Authorization: Bearer 1:EXAMPLElSnp6MVZvUXhRQUViN2RCYUFVZy9zTElB..."
```

The following is example output.

```
{
  "id": 1,
  "node_status": "online",
  "node_name": "my-node-name",
  "uuid": "12345a6b-7c89-0d12-3456-78fe9012f345",
  "label": "a1:23:45:6b:70:80",
  "model_number": "Q0626",
  "capacity_in_bytes": "25605032656896",
  "serial_number": "1234567890",
  "mac_address": "00:00:1a:00:23:bc"
},
...
```

Connecting to External Services

Creating and Using Access Tokens to Authenticate External Services to Qumulo Core

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

> **Tip**
> It is possible to confuse the terms *access token* and *session token*. Unlike access tokens, session tokens are short-lived and require a password to refresh, for example, to authenticate by using the `qq login` command. Access tokens are the focus of this section.

In Qumulo Core 5.3.0 (and higher), you can use *access tokens* to let a user authenticate to the Qumulo Core REST API without having to complete repetitive login procedures.

Access tokens are long-lived. They provide an alternative to session-based authentication that the `qq login` command and the Qumulo Core Web UI use. They also support 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 78).

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

Run the `qq 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 `qq 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 run the `qq auth_create_access_token` 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 71) or update the expiration time for your existing token (page 77).

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 72), 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 Core 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://203.0.113.0:8000/v1/session/who-am-i -H 'Authorization: Bearer acces
s-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('203.0.113.0', 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

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

```
$ qq auth_get_access_token 123456789012345678901
```

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-1-1-12-12345678-2345678901-1234567890-500",
    "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-1-1-12-12345678-2345678901-1234567890-1002",
    "uid": null
  }
}
```


To Get Metadata for All Access Tokens

Run 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
======================  =====  =======  ==============================
1234567890123456789012  svc    admin    2022-10-27T15:18:09.725513764Z
0987654321098765432109  svc    admin    2022-10-27T15:18:24.997572918Z

expiration time       enabled
====================  =======
                      True
2023-01-01T00:00:00Z  False
```

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

Run 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 71) or update the expiration time for your existing token (page 77).

> **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, run the `qq auth_modify_access_token` command, specify the access token ID, and use the `-d` flag. For example:

```
$ qq auth_modify_access_token 123456789012345678901 -d
```

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

```
$ qq auth_modify_access_token 123456789012345678901 -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, run the `qq auth_delete_access_token` command and specify the access token ID. For example:

```
$ qq auth_delete_access_token 123456789012345678901
```

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 the Qumulo Core Web UI.
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 71) 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, run 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, run 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, run the `qq auth_delete_access_token` command.

Connecting Your Kubernetes Cluster to Your Qumulo Cluster by Using the Qumulo Container Storage Interface (CSI) Driver

This section introduces the Qumulo Container Storage Interface (CSI) driver and explains how you can connect your Kubernetes cluster to your Qumulo cluster by using the Qumulo CSI driver.

To automate container storage, enable dynamic volumes, and help you scale your application container images based on usage and workflows, Qumulo uses its CSI driver to connect the Kubernetes orchestrator to Qumulo persistent storage. (In comparison, for example, the NFS CSI Driver for Kubernetes requires unprivileged NFS access for dynamic volumes and doesn't support volume sizing and expansion.)

For general driver information, see the Container Storage Interface (CSI) Specification.

Supported Features

The Qumulo CSI Driver supports:

- Static and dynamic (expansion) provisioning over NFSv3
- The following Persistent Volume Claim access modes:
  - `ReadOnlyMany`
  - `ReadWriteMany`
  - `ReadWriteOnce`
  - `ReadWriteOncePod`
- NFSv4.1

> **Important**
> Even when you enable NFSv4.1 for your Qumulo cluster, you must explicitly configure NFSv4.1 to work with Kerberos (page 252).

Unsupported Features

- Volume cloning
- Volume snapshot and restore

Requirements

- A Qumulo cluster
- Kubernetes 1.19 (and higher)

Connecting Your Qumulo Cluster to Kubernetes

This section explains how you can configure, provision, and mount Qumulo storage for each *Pod* (a logical wrapper for a container) on Kubernetes by using dynamic provisioning. This gives you more control over persistent volume capacity.

Step 1: Install the Qumulo CSI Driver

1. Log in to a machine that has `kubectl` and can access your Kubernetes cluster.
2. Download the `.zip` file or use one of the following commands.

- S3:

```
aws s3 cp s3://csi-driver-qumulo/deploy_v1.1.0.zip ./
```

- HTTP:

```
wget https://csi-driver-qumulo.s3.us-west-2.amazonaws.com/deploy_v1.1.0.zip
```

3. Extract the contents of the `.zip` file.
4. Run the shell script and specify the current release version. For example:

- Linux:

```
cd deploy_v1.1.0
chmod +x install_driver.sh
./install-driver.sh
```

- Windows:

```
cd deploy_v1.1.0
install-driver.bat
```

The script configures Qumulo's prebuilt Elastic Container Registry (ECR) image (from `public.ecr.aws/qumulo/csi-driver-qumulo:v1.1.0`) and installs it on your Kubernetes system.

Step 2: Configure Volume and NFS Export Paths

To prepare your Qumulo cluster for connecting to your Kubernetes cluster, you must first configure your volume and NFS export paths on your Qumulo cluster by setting the following parameters for each storage class that you define.

> **Tip**
> Write down the paths for the following YAML keys for the `storageclass-qumulo.yaml` file that you use when you create a storage class in step 5 (page 85).

1. For `storeRealPath`, from the root of the Qumulo file system, create a directory for storing volumes on your Qumulo cluster, for example `/csi/volumes1`.

   > **Note**
   > Because the CSI driver doesn't create the directory listed in the `storeRealPath` key automatically, this directory must exist below the NFS export and must not be the NFS export itself.

2. For `storeExportPath`, create the NFS export for hosting the persistent volume.
3. If your cluster has more than one tenant, specify the tenant ID that contains your NFS export for the `tenantId` parameter. For more information, see Configure Multi-Tenancy with Qumulo on Qumulo Care.

   > **Note**
   >
   > f you have only one tenant, it isn't necessary to specify the `tenantId` parameter.
   >
   > ou must provide the value for `tenantId` as a string. For example: `"2"`.

Step 3: Configure Credentials

To connect your Kubernetes cluster to your Qumulo cluster, you must either use an existing account or create a new account for the CSI driver to communicate with the Qumulo API.

1. Configure a username and password for a user on your Qumulo cluster.
2. The configured username must have the following file permissions:

- Lookup on `storeRealPath`
- Create directories in `storeRealPath`
- Create and modify quotas:

◦ `PRIVILEGE_QUOTA_READ`

- `PRIVILEGE_QUOTA_WRITE`
  - Read NFS exports: `PRIVILEGE_NFS_EXPORT_READ`
  - Perform `TreeDelete` operations on volume directories: `PRIVILEGE_FS_DELETE_TREE_WRITE`

For more information, see Role-Based Access Control (RBAC) with Qumulo Core (page 90) on Qumulo Care.

Step 4: Create and Configure Secrets

To allow the CSI driver to operate with your Qumulo cluster, you must create and configure Secrets. You may use either Basic Authentication with a username and password, or an Access Token. Depending on configuration, Basic Authentication may be disallowed and using an Access Token will be required.

1. Configure one of the following authentication types.

- Basic Authentication:

```
kubectl create secret generic cluster1-login \
  --type="kubernetes.io/basic-auth" \
  --from-literal=username=bill \
  --from-literal=password=SuperSecret \
  --namespace=kube-system
```

- Access Token:

```
TOKEN='access-v1:zNTc5D0zWTdNi/KsZo620fu71TweGh47u+S/5NbV...'
kubectl create secret generic cluster1-login \
  --from-literal=access_token="$TOKEN" \
  --namespace=kube-system
```

2. Give the CSI driver access to the Secrets. For example:

```
kubectl create role access-secrets \
  --verb=get,list,watch \
  --resource=secrets \
  --namespace kube-system
kubectl create rolebinding \
  --role=access-secrets default-to-secrets \
  --serviceaccount=kube-system:csi-qumulo-controller-sa \
  --namespace kube-system
```

Step 5: Create a Storage Class

To link your Kubernetes cluster to your Qumulo cluster, you must create a storage class on your Kubernetes cluster.

1. Begin with the example Qumulo storage class configuration.

> **Note**
>
> n the following example, it is possible to use a fully qualified domain name (FQDN) for he `parameters: server:` entry.
>
> For such a configuration, all Kubernetes nodes in the cluster must be able to resolve FQDNs.

```
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cluster1
provisioner: qumulo.csi.k8s.io
parameters:
  server: 203.0.113.0
  storeRealPath: "/regions/4234/volumes"
  storeExportPath: "/some/export"
  csi.storage.k8s.io/provisioner-secret-name: cluster1-login
  csi.storage.k8s.io/provisioner-secret-namespace: kube-system
  csi.storage.k8s.io/controller-expand-secret-name: cluster1-login
  csi.storage.k8s.io/controller-expand-secret-namespace: kube-system
reclaimPolicy: Delete
volumeBindingMode: Immediate
mountOptions:
  - nolock
  - proto=tcp
  - vers=3
allowVolumeExpansion: true
```

2. Edit the configuration for your Qumulo cluster.

    a. Name your storage class.

    b. Specify server and `storeRealPath`.

    c. Specify `storeExportPath`.

    d. (Optional) Specify `tenantId`.

    > **Note**
    > You must provide the value for `tenantId` as a string. For example: `"2"`.

    e. Configure the following parameters to point to the Secrets that you have created and configured (page 84) in the namespace in which you installed the CSI driver:

- `controller-expand-secret-name`
- `controller-expand-secret-namespace`
- `provisioner-secret-name`
- `provisioner-secret-namespace`

    f. Specify the NFS `mountOptions`. For example:

```
mountOptions:
  - nolock
  - proto=tcp
  - vers=3
```

g. To create the class, apply the configuration. For example:

```
kubectl create -f storageclass-qumulo.yaml
```

Step 6: Create a Persistent Volume Claim (PVC) and Apply it to a Pod

To apply a PVC claim to a Pod dynamically, you must first configure and create it.

1. Begin with the example PVC configuration.

```
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: claim1
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: cluster1
  resources:
    requests:
      storage: 1Gi
```

2. Edit the configuration for your PVC claim.
   a. Name your claim.
   b. Change `storeClassName` to the name of your claim.
   c. Specify the capacity in `spec.resources.requests.storage`. This parameter lets you create a quota on your Qumulo cluster.
   d. To create the claim, apply the configuration. For example:

```
kubectl apply -f dynamic-pvc.yaml
```

3. Use the claim in a Pod or a Deployment. For example:

```
---
apiVersion: v1
kind: Pod
metadata:
  name: claim1-pod
spec:
  volumes:
    - name: cluster1
      persistentVolumeClaim:
        claimName: claim1
  containers:
    - name: claim1-container
      image: ...
      volumeMounts:
        - mountPath: "/cluster1"
          name: cluster1
```

> ⚠ Important
> When the PVC is released, a tree-delete is initiated on the Qumulo cluster for the directory that the PVC indicates. To prevent this behavior, set `reclaimPolicy` to `Retain`.

4. You can launch and use your container image.

Authorization

Managing File Locks in Qumulo Core

This section explains how file locks, legal holds, and retention periods work in Qumulo Core. For information about working with S3 Object Lock, see Enabling Object Lock for S3 Buckets (page 308).

How File Locks Works in Qumulo Core

A *file lock* can prevent a regular file from being modified. There are two file lock types, both of which require the `FS_FILE_LOCK_WRITE` privilege.

- **Legal Hold:** A file lock that you can enable for an indefinite time period. This file lock type is best suited for temporary scenarios.
- **Retention Period:** A file lock that you can enable for a specific period of time by configuring a *retention period* which can be extended but never shortened or removed. This file lock type is best suited to locking files for a known time period.

For more information and examples of configuring file locks, see the following resources in the Qumulo `qq` CLI Command Guide:

- `qq fs_file_modify_lock`
- `qq fs_file_get_attr`

How You Can Use File Locks to Implement the Write Once Read Many (WORM) Model

To lock any newly created file once the client finishes writing to it, you can watch for file attribute and directory changes by using REST (page 223). For example, you can use the `child_data_written` or `child_size_change` events. For more information, see SSE Payload Notification Types (page 224).

To implement the WORM model and maintain the file's *write-once* integrity, use file locks to prevent any further modifications after the file becomes immutable following the initial write operation.

Managing Role-Based Access Control (RBAC) for Users and Groups in Qumulo Core

This section explains how Role-Based Access Control (RBAC) for users and groups works in Qumulo Core, explains the role types, and shows how to manage them by using the Qumulo Core Web UI.

To share management responsibilities with others, you can grant specific privileges to a user or group—locally or through Active Directory—by using RBAC.

⚠ Important

- For changes to take effect, a user account with newly assigned roles must log out of Qumulo Core and then log back in (or its sessions must time out).
- Because certain privileges (such as replication-write privileges) can overwrite or move data to a location where a user has greater (or total) permissions, use special care when you grant privileges to roles and users.

Qumulo Core Role Types

This section explains the Administrators, Data-Administrators, Observers, and Custom role types in Qumulo Core.

Administrators

ℹ Note
Only the default administrator account can access a Qumulo cluster by using SSH.

This role is suitable for system administrators. Users with this role have full access to, and control of, the cluster, including:

- Configuration and management of general cluster settings for audit logging, snapshots, replication, quotas, and so on by using the Qumulo Core Web UI, REST API, or `qq` CLI
- Creation of files and directories in any current and future directories
- Reading of any files and file attributes and listing of any directories in any current and future directories
- Deletion or renaming of any files and directories in any current and future directories
- Changing of ownership and permissions for any files and directories in any current and future directories

Data-Administrators

This role is suitable for Qumulo Core REST API and `qq` CLI users who don't have access to the Qumulo Core Web UI but have the same file privileges as those of the Administrators role, including:

- Read and write permissions for all NFS, SMB, quota, and snapshot APIs
- Read-only permissions for local API users
- Access to analytics and file system

Observers

This role is suitable for users or groups who can access the Qumulo Core Web UI and read-only APIs (with the exception of debug APIs and authentication settings).

> **Note**
>
> - Clusters that run Qumulo Core 3.0.5 (and higher) don't assign the Observers role automatically and non-administrative users don't have access to the Qumulo Core Web UI or read-only APIs (unless you explicitly assign the necessary role to specific usernames).
> - It is possible to assign both Data-Administrators and Observers roles to a single user. This can give the user the ability to manage data on your Qumulo cluster by using the Qumulo Core Web UI without full administrative access.

Custom

For information about managing RBAC and creating custom roles by using the `qq` CLI, see the following sections in the Qumulo `qq` CLI Command Guide:

- `qq auth_assign_role`
- `qq auth_create_role`
- `qq auth_list_privileges`
- `qq auth_modify_role`
- `qq auth_unassign_role`

Managing Roles by Using the Qumulo Core Web UI

This section explains how to add a member to, and remove a member from, an existing Qumulo Core role and how to create and edit a custom role.

To Add a Member to an Existing Qumulo Core Role

1. Log in to the Qumulo Core Web UI.

2. Click **Cluster** > **Role Management**.
3. On the **Role Management** page, next to the role to assign, click **Add Member**.
4. In the **Add Member to <Role Type>** dialog box, enter the **Trustee** and then click **Yes, Add Member**.

   > **Tip**
   > For examples of valid trustees, click .

5. Click **Yes, Assign Role**.

To Remove a Member from an Existing Qumulo Core Role

1. Log in to the Qumulo Core Web UI.
2. Click **Cluster** > **Role Management**.
3. On the **Role Management** page, next to the user or group to remove from a role, click .

To Create a Custom Qumulo Core Role

1. Log in to the Qumulo Core Web UI.
2. Click **Cluster** > **Role Management**.
3. On the **Role Management** page, on the the upper-right side, click **Create Role**.
4. On the **Create Role** page:
   a. Enter a **Name** and **Description**.
   b. Select the privileges to add to the role and click **Save**.

To Edit a Custom Qumulo Core Role

1. Log in to the Qumulo Core Web UI.
2. Click **Cluster** > **Role Management**.
3. On the **Cluster Management** page, next to the role to edit, click .
4. On the **Edit <Role Name>** page, select the privileges to include in the role and click **Save**.

Managing Cross-Protocol Permissions (XPP) in Qumulo Core

This section explains how Cross-Protocol Permissions (XPP) work in Qumulo Core and how to enable, disable, and check the status of XPP by using the `qq` CLI.

How Cross-Protocol Permissions (XPP) Work in Qumulo Core

Qumulo Core works with clients that use multiple protocols, such as SMB (page 0) and NFS (page 0). While SMB and NFS permission models are interoperable at a basic level, SMB offers a complex permission definition which isn't fully compatible with NFS. For this reason, it is necessary to "translate" between the two protocols when clients access the same files and directories over SMB and NFS.

XPP enables mixed SMB and NFS protocol workflows by preserving SMB access control lists, by maintaining permission inheritance, and by reducing application permission incompatibility.

When there are no cross-protocol interactions, Qumulo Core operates according to precise protocol specifications. When protocol conflicts arise, XPP minimizes the possibility of application incompatibility.

> ⚠ Important
>
> - XPP doesn't break compatibility with previous Qumulo Core releases.
> - Enabling XPP doesn't change the rights on *existing* files in your file system. Changes take place only *after* you enable XPP.

For more information, see the following resources:

- Qumulo Core Permission Modes
- Cross-Protocol Permissions (XPP) in Common Scenarios
- Cross-Protocol Permissions Test Drive Website.

Common Workflow Scenarios for Working with Cross-Protocol Permissions (XPP)

This section gives examples of common workflow scenarios and explains how Qumulo Core functions when you enable XPP in these scenarios.

- **Single-Protocol Workflows (Only SMB or NFS):** Qumulo Core operates as expected, according to original protocol specifications.
- **Mixed-Protocol Workflows (Mostly Windows or SMB):** Qumulo Core operates as expected, with the following exceptions:

    - Because running the `chmod` command on a directory doesn't affect the ACL that the directory's children inherit, the command doesn't break the permission inheritance.
    - To preserve compatibility, the `chmod` command retains the ability to strip rights from privileged groups and to override the inherited rights for individual files.
- Mixed-Protocol Workflows (Mostly NFS) Qumulo core operates as expected, with one exception: To preserve compatibility, Qumulo Core permits SMB clients to add access control entries (ACEs) to files and directories

> **Note**
> XPP reveals permissions that Native Permissions Mode hides. This can trigger security checks from `ssh` and `sshd` commands. If you use `ssh` to access NFS home directories, see Using SSH with Cross-Protocol Permissions for more information.

To Manage Cross-Protocol Permissions (XPP)

Qumulo Core enables and disables XPP immediately, without scanning the directory tree. Existing file and directory permissions remain unaffected unless—or until—your workflow modifies them.

- To enable XPP, run the `qq fs_set_permissions_settings cross_protocol` command.

  > **Tip**
  > We recommend creating a snapshot before enabling XPP in a production environment.

- To disable XPP, run the `qq fs_set_permissions_settings native` command.
- To check the current permissions mode, run the `qq fs_get_permissions_settings` command.

Troubleshooting the Permissions for a File or Directory

Explain Permissions Tools is a suite of diagnostic utilities that examines a file or directory and explains the structure of permissions for the file or directory. For more information, see the following sections in the Qumulo `qq` CLI Command Guide:

- `qq fs_acl_explain_chmod`
- `qq fs_acl_explain_posix_mode`
- `qq fs_acl_explain_rights`

Using SMB Host Restrictions in Qumulo Core

This section explains how to use SMB host restrictions in Qumulo Core to provide fine-grained control of access to SMB shares, based on client IP address ranges.

Depending on the configuration of your Qumulo cluster, you can grant full access, read-only access, deny specific hosts, or deny all access. It is also possible to configure a Qumulo cluster to prevent shares which a client can't access from being enumerated.

> ⚠ Important
> Because host restrictions interact with user or group share permissions and file permissions on the basis of *least privilege*, in order for Qumulo Core to grant a privilege for a particular file, the file's permissions, the share's user permissions, and the share's host permissions must *all* permit the privilege.

Host restrictions apply in the order in which you write them, from top to bottom. For example, if you deny a privilege to a host at the beginning of the permission list, and a later entry allows the same privilege to the same host, Qumulo Core doesn't grant the privilege.

How SMB Host Restriction Precedence Works

When you create or modify an SMB share, you can use one of the following SMB host restrictions, listed here in order of precedence.

> ⚠ Important
>
> - If you don't specify any of the following restrictions, the SMB share gives all hosts full control
> - If you specify one of the following restrictions, the SMB share denies access to any hosts that you don't specify.
> - If you specify multiple addresses or ranges, separate them by using *spaces*.

1. Deny All Access
2. Deny Specific Hosts: IP address ranges to which Qumulo Core denies access to this share, regardless of other permissions
3. Permit Read-Only Access: IP address ranges to which Qumulo Core permits only read-only access
4. Full Access: IP address ranges to which Qumulo Core permits full access

> **Important**
> The file's permissions and the share's user permissions must also grant full access.

Managing SMB Host Restrictions by Using the qq CLI

For information about viewing, modifying, and removing host restrictions and hiding SMB shares from unauthorized hosts by using the `qq` CLI, see the following sections in the Qumulo `qq` CLI Command Guide:

- `qq smb_add_share`
- `qq smb_list_share`
- `qq smb_mod_share`
- `qq smb_modify_settings`

> **Tip**
> To hide an SMB share in the Web UI, append `$` to its name. To access the share, you must use the fully qualified domain name (FQDN). For example: `\\storage.example.com\MyShare$`.

Using Active Directory for POSIX Attributes in Qumulo Core

This section explains how to use Active Directory (AD) for POSIX attributes in Qumulo Core for clusters with multi-protocol access (with NFS and SMB) that manage POSIX and Windows identities from within Active Directory.

> **Important**
> For changes to take effect, any Qumulo clusters that are already joined to an Active Directory domain must leave the domain and then rejoin it.

How Full Credential Expansion Works in Qumulo Core

Because the SMB and NFS protocols have unique identifiers and exist in different identity domains, it becomes difficult to link the two protocols when they represent the same identity. In addition, storage devices can't determine the entity that attempts to access a file; as a result, a file that a Linux system writes can be inaccessible on a Windows machine.

One solution to this issue is *full credential expansion*, which involves mapping the two identities—Windows identities for SMB clients and POSIX identities for NFS clients—by using Active Directory as a central reference. For more information, see RFC 2307 . This approach ensures that, when you use Active Directory to maintain identity mappings from POSIX to Windows, Qumulo Core abides by the mappings.

After you enable Active Directory for POSIX attributes in Qumulo Core, you must enable user identity mapping from your Windows SID to your NFS UID. To do this, assign a *user object* (SID or `objectSid`) to every object in Windows and enter the NFS UID of the user as an object attribute. This configuration allows Qumulo Core to correlate an NFS UID (for example, `2053`) to a SID on Windows (for example, `S-1-5-21-...`).

Whenever this user identity is required (for example, to check permissions), Qumulo Core uses the established mapping to retrieve the entire identity for the user by referencing the NFS UIDs and GIDs, and all SIDs, including the group IDs of any relevant parent groups.

> **Note**
> The full credential expansion method lets your Qumulo Core cluster support more than 16 group memberships for your NFS users, as long as Active Directory manages the group memberships.

Enabling Active Directory for POSIX Attributes in Qumulo Core

This section explains how to enable Active Directory for POSIX attributes in Qumulo Core by using the Web UI and REST API.

To Enable Active Directory for POSIX Attributes by Using the Qumulo Core Web UI

1. Log in to the Qumulo Core Web UI.
2. Click **Cluster** > **Active Directory**.
3. On the **Active Directory** page:
    a. Enter the details for your Active Directory server.
    b. For **Use Active Directory for POSIX Attributes**, click **Yes**.
    c. (Optional) To limit part of the Active Directory tree that Qumulo Core can query, enter a **Base DN (Distinguished Name) for User and Group Accounts**.
    d. Click **Join**.

When your cluster joins Active Directory, all SMB sessions and NFS operations result in full credential expansion for every user. For example, when NFS UID 2053 attempts to access a file, the cluster first queries the AD server to:

- Determine the groups to which the user belongs
- Map the user and groups to all Windows SIDs
- Apply permissions based on the fully expanded credential set

This configuration allows Qumulo Core to correlate an NFS UID (for example, `2053`) to a SID on Windows (for example, `S-1-5-21-...`).

To Manage Active Directory for POSIX Attributes by Using the Qumulo Core REST API

To toggle Qumulo Core's ability to use Active Directory for POSIX attributes, use the fields `use_ad_posix_attributes` and `base_dn` for the following REST API endpoints.

- Get Configuration and Status: `/v1/ad/status`
- Get Operation Status: `/v1/ad/monitor`
- Join Active Directory: `/v1/ad/join`

To map identities from one domain to another, use the following REST API endpoints.

- GID to SIDs: `/v1/ad/uids/:gid:/sids`
- Local Username to All Related Identities: `/v1/auth/local-username/:username:/related-identities`
- POSIX GID to All Related Identities: `/v1/auth/posix-gids/:id:/related-identities`
- POSIX UID to All Related Identities: `/v1/auth/posix-uids/:id:/related-identities`

- SID to Expanded Group SIDs: `/v1/ad/uids/:gid:/sids`
- SID to GID: `/v1/ad/uids/:uid:/gid`
- SID to UID: `/v1/ad/uids/:sid:/uid`
- UID to SIDs: `/v1/ad/uids/:uid:/sids`
- Windows NT SID to All Related Identities: `/v1/auth/sids/:id:/related-identities`

> **Note**
> It is possible for one UID to be mapped to multiple SIDs.

To retrieve related identities, use the `/v1/auth/auth-ids/:id:/related-identities` REST API endpoint.

Network Configuration

Required Networking Ports for Qumulo Core

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

> **Note**
>
> - Communication *between* nodes in a cluster must be possible on any port.
> - Active Directory authentication services require their own network port range. For an authoritative list, see Active Directory and Active Directory Domain Service Port Requirements in the Windows Server 2008 R2 and Windows Server 2008 documentation.

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—that specify the TCP mount option or transfer data over NFS after mounting—don't use UDP.
3712	TCP	Replication

Port	Protocols	Use
8000	TCP	REST API
9000	TCP	S3 API, if you enable the S3 API for your Qumulo cluster (page 290)
32768-60999	TCP	FTP Passive Mode

Networking Ports for Outbound Connections

> **Note**
> For cluster formation and inter-node communication Qumulo Core requires unblocked IPv6 traffic in the local subnet—for more information, see Configuring IPv6 in Qumulo Core (page 102).

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 (by default)
443	TCP	Qumulo Shift for Amazon S3 (by default)
514	TCP	Audit with `Rsyslog` (by default)
3712	TCP	Replication (by default)

Configuring IPv6 in Qumulo Core

This section explains how to configure IPv6 in Qumulo Core by configuring the default gateway and maximum transmission unit (MTU).

> **Note**
>
> - The Qumulo Core Web UI doesn't show the default gateway for IPv6.
> - As for IPv4, you can configure one untagged IPv6 network by using VLAN 0 or multiple tagged networks by using VLAN 1-4094 statically.
> - It is possible to configure an IPv4 network alongside an IPv6 network on any VLAN.
> - Currently, Qumulo Core doesn't support VPN connections by using IPv6.

To Configure IPv6 by Using the qq CLI

1. To specify the default gateway for IPv4 and IPv6 traffic, run the `qq network_mod_interface` command. For example:

```
qq network_mod_interface --default-gateway 192.168.0.1 \
  --default-gateway-ipv6 2001:db8:60e0:7352:e9d2:e180:e7e8:cdb8 \
  --mtu 1500
```

2. To specify IPv6 addresses for your Qumulo cluster, run the `qq network_mod_network` command to modify the default network configuration. For example:

```
qq network_mod_network --network-id 1 \
  --assigned-by STATIC \
  --netmask 2001:db8::/64 \
  --ip-ranges 2001:db8:0:0:1::1-2001:db8:0:0:1::ffff \
  --floating-ip-range 2001:db8:0:0:2::1-2001:db8:0:0:2::ffff \
  --dns-servers 2001:db8:8560:26eb:19f2:fe28:c49f:7f4c \
    2001:db8:2ebf:276a:375:a593:5c3a:d4c4 \
  --dns-search-domains example.com
```

> **Note**
>
> For IPv6 networks, you can specify subnets by using CIDR notation (for example, `2001:DB8/32`) or by using a standard netmask (for example, `ffff:ffff:ffff:ffff::`).

3. To verify the configuration and confirm the assigned IPv6 addresses, uses the `qq network_poll` command.
4. (Optional) to check that the assigned IPv6 addresses are accessible, run the `ping` command with the IPv6 address of a node in your cluster.

Connecting to Multiple Virtual Networks in Qumulo Core

This section explains how to connect a Qumulo cluster to multiple virtual networks by using VLAN tagging.

Each node in a Qumulo cluster has a single NIC labeled `bond0` and creates a bond between two physical ports by using either *active backup* (a redundant configuration in which one port is active while the other is in standby mode) or *LACP* (the aggregation of multiple, parallel network connections).

When you create a Qumulo cluster, its configuration includes a network named `Default`. The configuration uses DHCP, the bond, and a single default gateway associated with the bond.

Step 1: Configure Default DHCP Settings

You can configure your Qumulo cluster from the default DHCP configuration. If you have already configured a single static network, skip this step and add another network (page 104).

1. To assign a default gateway, run the `qq network_mod_interface` command. For example:

```
qq network_mod_interface \
  --default-gateway 203.0.113.0
```

2. To configure the `Default` network to use static (rather than DHCP) addressing, run the `qq network_mod_network` command. For example:

```
qq network_mod_network \
  --network-id 1\
  --assigned-by STATIC \
  --ip-ranges 203.0.113.0-10 \
  --floating-ip-ranges 203.0.113.10-20 \
  --netmask 255.255.255.0 \
  --dns-servers 203.0.113.1 \
  --dns-search-domains example.com
```

3. Log in to the system by using one of the assigned IP addresses.

Step 2: Add More Networks

After you configure the first network for your Qumulo cluster, you can add more networks to it.

Note

- If you can add both static and floating IP addresses, they must be on the same network.
- You can use multiple `--floating-ip-ranges` or `--dns-servers` flags.
- Each VLAN can have different MTU values, as long as the MTU value of the `Default` network is equal or greater than the combined MTU values of all networks in your Qumulo cluster. To set the MTU value for the `Default` network, run the `qq network_mod_interface` command. Then, you can set an equal or smaller MTU value for the other networks.

1. To add a network, run the `qq network_add_network` command. For example:

```
qq network_add_network \
  --name MyNetworkName \
  --ip-ranges 192.168.0.1-10 \
  --floating-ip-ranges 192.168.0.11-21 \
  --netmask 255.255.255.0 \
  --dns-servers 8.8.8.8 \
  --dns-search-domains example.com \
  --mtu 1500 \
  --vlan-id 200
```

2. To view the details your networks, run the `qq network_list_networks` command.
3. To view the IP addresses assigned to the nodes in your cluster, run the `qq network_poll` command.

Configuring Round-Robin DNS on Windows Server for Qumulo Core

This section explains how to configure a single namespace on your Qumulo cluster to configure round-robin DNS on a domain controller running Windows Server 2008 R2 (or higher).

> **Note**
> To perform the following operations, you must be a member of the Domain Admins, Enterprise Admins, or DnsAdmins group.

Step 1: Confirm that Round-Robin DNS Support is Enabled

1. In the Microsoft Management Console (MMC), ensure that the DNS Manager snap-in is installed.
2. Click **Start** > **Administrative Tools** > **DNS**, right-click **DNS**, and then click **Properties**.
3. Click **Advanced** and ensure that round-robin DNS is enabled.

Step 2: Add DNS Entries for Each Remote Desktop Session Host Server

1. In the DNS Manager snap-in, expand your server name, then **Forward Lookup Zones**, and then your domain name.
2. Right-click a zone and then click **New Host (A or AAAA)**.
3. For **Name**, enter the hostname (virtual name) for clients that connect to your Qumulo cluster.

   > **Note**
   > Don't enter the hostname of another existing server.

4. Enter the first floating IP address for one of the nodes in your cluster.

   > **Note**
   > Use only floating IP addresses for round-robin DNS entries.

5. Repeat these steps for each floating IP address that belongs to a node in your cluster.

Step 3: Configure the Time to Live (TTL) for DNS Requests to Your Cluster

To ensure that client connections to your cluster are balanced evenly, you must provide a single namespace for your cluster. To do this, configure your DNS server to send a different IP address for each DNS request for your Qumulo cluster.

For example, you can set the TTL for each record to `0`, to allow each DNS lookup for your Qumulo cluster to yield one of the four configured IP addresses.

1. In the DNS Manager snap-in, click **View > Advanced** and open your records for editing.
2. Configure the TTL for each record.

Network Multitenancy

Partitioning a Qumulo Cluster into Tenants

This section explains how to enable, disable, and use network multitenancy in Qumulo Core.

In Qumulo Core 5.3.4 (and higher), *network multitenancy* lets you partition a single physical Qumulo cluster into multiple virtual *tenants.* You can define a tenant by using a name and one or more networks. When you assign a network to a tenant, Qumulo Core treats any client that connects from that network as part of that tenant. For more information, see Connecting to Multiple Virtual Networks in Qumulo Core.

For each tenant, you can specify individual management protocol access and configuration (page 113) and file system protocol access and configuration (page 116).

> **Note**
> All tenants share the cluster's underlying file system, identity providers, role-based access control (RBAC) configuration, and other global settings.

Prerequisites

> **Important**
> If your cluster runs a version of Qumulo Core lower than 6.1.0.3, you must use the `qq multitenancy_enable` command to enable multitenancy for your cluster.

To manage network multitenancy and tenants, your user must have membership in a Qumulo role with the following privileges.

- `PRIVILEGE_NETWORK_READ`: Viewing networks
- `PRIVILEGE_NETWORK_WRITE`: Assigning networks to tenants
- `PRIVILEGE_TENANT_READ`: Viewing tenants
- `PRIVILEGE_TENANT_WRITE`: Enabling network multitenancy and creating, modifying, and deleting tenants

Creating, Configuring, and Unassigning Tenants by Using the qq CLI.

This section explains the lifecycle of working with tenants, including creating tenants, assigning networks to new and existing tenants, viewing tenant configuration, and unassigning tenants.

> ⚠ Important
> Creating a tenant and assigning a network to it causes the removal of any static routes from all nodes in a cluster. After creating the new tenant, you must reapply the static routes to your nodes.

Step 1: Create a New Tenant

After you enable multitenancy for your cluster, you can create a new tenant.

Run the `qq multitenancy_create_tenant` command and specify a name for your tenant. For example:

```
$ qq multitenancy_create_tenant \
  --name my_tenant
```

Your cluster creates a new tenant with no networks assigned to it. By default, Qumulo Core disables access to the tenant through all protocols.

Step 2: Assign Networks to a Tenant

To allow a tenant to apply its configuration to clients that connect to the cluster from specific networks, you must associate the tenant with one or more networks. To do this, you can:

- Create a new tenant with networks assigned to it.
- Assign networks to, and unassign networks from, an existing tenant.
- Move networks between tenants.

Creating a New Tenant with Assigned Networks

Use the `qq multitenancy_create_tenant` and specify the tenant name and network ID.

> ⚠ Important
> The network must not be assigned to another tenant already.

In the following example, we specify a single network.

```
$ qq multitenancy_create_tenant \
  --name my_tenant_name \
  --network-id 2
```

Assigning a Single Network to an Existing Tenant

To assign a single network to a tenant, modify *the tenant that belongs to the network.* Run the `qq network_mod_network` command and specify the network and tenant ID.

```
$ qq network_mod_network \
  --network-id 3 \
  --tenant-id 2
```

Assigning Multiple Networks to an Existing Tenant

To assign multiple networks to a tenant, modify *the networks that belong to the tenant.* Run the `qq multitenancy_modify_tenant` command and specify the tenant and network ID.

> ⚠ Important
> Any existing networks not specified after the `--network-id` flag become unassigned.

In the following example, we specify three networks.

```
$ qq multitenancy_modify_tenant \
  --id 2 \
  --network-id 2 3 4
```

Unassigning a Single Network from a Tenant

To unassign a single network from a tenant, clear *the tenant that belongs to the network*. Run the `qq network_mod_network` command and specify the network and the `--clear-tenant-id` flag.

> ℹ Note
> After you unassign a network from a tenant, you can assign it to another tenant.

```
$ qq network_mod_network \
  --network-id 3 \
  --clear-tenant-id
```

Unassigning All Networks from a Tenant

To unassign all networks from a tenant, clear *the networks that belong to the tenant*. Run the `qq multitenancy_modify_tenant` command and specify the tenant and network ID.

> **Note**
>
> - After you unassign a network from a tenant, you can assign it to another tenant.
> - Don't specify any arguments for the `--network-id` flag.

```
$ qq multitenancy_modify_tenant \
  --id 2 \
  --network-id
```

Moving Networks between Tenants

Run the `qq network_mod_network` command and specify the network and target tenant.

```
$ qq network_mod_network \
  --network-id 2 \
  --tenant-id 1
```

Step 3: View Tenant Information

To determine a tenant's network assignments and enabled management and file system protocols, you can view the tenant information.

Viewing Information for a Single Tenant

To view the information for a single tenant, run the `qq multitenancy_get_tenant` command.

```
$ qq multitenancy_get_tenant \
  --id 1
```

Viewing Information for All Tenants

- Log in to the Qumulo Core Web UI.
- Click **Cluster** > **Network Multitenancy**.
- In the `qq` CLI, run the `qq multitenancy_list_tenants` command.

Determining the Tenant Assignment for Networks

- To view the information for a single network, run the `qq network_get_network` command.

```
$ qq network_get_network \
  --network-id 2
```

- To view the information for all networks, run the `qq network_list_networks` command.

Step 4: Delete a Tenant

> ⚠ Important
>
> - When you delete a tenant, Qumulo Core removes the tenant's entire configuration from your cluster, including NFS exports and SMB shares associated with the tenant.
> - It isn't possible to delete the last tenant.

To delete a tenant, run the `qq multitenancy_delete_tenant` command and specify the tenant ID.

```
$ qq multitenancy_delete_tenant \
  --id 2
```

Known Network Multitenancy Limitations in Qumulo Core

Currently, Qumulo Core doesn't support:

- Creating or modifying tenants on cloud-based clusters
- Using one VLAN on multiple tenants
- Using a separate Active Directory, standalone LDAP, or user-defined mapping configuration for each tenant
- Using a separate DNS configuration for each tenant
- Scoping RBAC privileges to each tenant

Configuring Management Protocols on a Tenant

This section explains how to configure management protocols for each tenant on a Qumulo cluster.

After you create tenants (page 108) on your Qumulo cluster, you can manage access for clients that connect to the cluster from the tenant's network by enabling or disabling the management protocols for each tenant.

> ⚠ Important
>
> - Access to a management protocol lets a client use the protocol to view and modify resources across the entire cluster, not only within the client's tenant.
> - If you disable a management protocol, you can still access your cluster by using a physical or remote console.

Prerequisites

To configure management protocols, your user must have membership in a Qumulo role with the following privileges.

- `PRIVILEGE_TENANT_READ`: Viewing tenants
- `PRIVILEGE_TENANT_WRITE`: Modifying tenants

Enabling and Disabling REST API Access

The Qumulo Core REST API lets you manage clusters by using the `qq` CLI, Python bindings, and REST API calls.

- To enable REST API access, run the `qq multitenancy_modify_tenant` command, specify the tenant ID, and use the `--enable-rest-api` flag.
- To disable access, use the `--disable-rest-api` flag.

In the following example, the cluster begins to accept REST API traffic on TCP port 8000 on all networks assigned to the tenant.

```
$ qq multitenancy_modify_tenant \
  --id 2
  --enable-rest-api
```

Enabling and Disabling Web UI Access

The Qumulo Core Web UI lets you manage clusters from a browser.

> **Note**
> Enabling Web UI access doesn't require enabling REST API access.

- To enable REST API access, run the `qq multitenancy_modify_tenant` command, specify the tenant ID, and use the `--enable-web-ui` flag.
- To disable access, use the `--disable-web-ui` flag.

In the following example, the cluster begins to serve the Qumulo Core Web UI on TCP ports 80 and 433 on all networks assigned to the tenant.

```
$ qq multitenancy_modify_tenant \
  --id 2
  --enable-web-ui
```

Enabling and Disabling SSH Access

SSH lets you view logs and use the `qq` CLI by using a client to connect to nodes in a cluster remotely.

> **Note**
> Enabling SSH access doesn't require enabling REST API access.

- To enable SSH access, run the `qq multitenancy_modify_tenant` command, specify the tenant ID, and use the `--enable-ssh` flag.
- To disable access, use the `--disable-ssh` flag.

In the following example, the cluster begins to accept SSH traffic on TCP port 22 on all networks assigned to the tenant.

```
$ qq multitenancy_modify_tenant \
  --id 2
  --enable-ssh
```

Enabling and Disabling Replication Access

Replication lets a cluster transfer data from a directory on one cluster to a directory on another cluster.

- To enable replication access, run the `qq multitenancy_modify_tenant` command, specify the tenant ID, and use the `--enable-replication` flag.
- To disable access, use the `--disable-replication` flag.

In the following example, the cluster begins to accept replication traffic on TCP port 3712 on all networks assigned to the tenant.

```
$ qq multitenancy_modify_tenant \
  --id 2 \
  --enable-replication
```

Configuring File System Protocols on a Tenant

This section describes how to configure file system protocols for each tenant on a Qumulo cluster.

After you create tenants (page 108) on your Qumulo cluster, you can manage access for clients that connect to the cluster from the tenant's network by enabling or disabling file system protocols, or configuring file system protocols specifically for each tenant.

Prerequisites

To configure file system protocols, your user must have membership in a Qumulo role with the following privileges.

- `PRIVILEGE_NFS_EXPORT_READ`: View NFS exports
- `PRIVILEGE_NFS_EXPORT_WRITE`: Create, modify, and delete NFS exports
- `PRIVILEGE_NFS_SETTINGS_READ`: View NFS settings
- `PRIVILEGE_NFS_SETTINGS_WRITE`: Modify NFS settings
- `PRIVILEGE_SMB_SHARE_READ`: View SMB shares and settings
- `PRIVILEGE_SMB_EXPORT_WRITE`: Create, modify, and delete SMB shares; modify SMB settings
- `PRIVILEGE_TENANT_READ`: View tenants
- `PRIVILEGE_TENANT_WRITE`: Modify tenants

Enabling NFS and Configuring Settings and Exports

This section explains how to enable the NFS protocol for a tenant, the difference between global settings and settings for each tenant, and how to configure NFS exports for a tenant.

Enabling the NFS Protocol for a Tenant

To let a tenant use NFS to access a specific set of exports, run the `qq multitenancy_modify_tenant` command, specify the tenant ID, and use the `--enable-nfs` flag.

> **Note**
> Initially, the tenant has no exports to mount.

In the following example, the cluster begins to accept NFS traffic on TCP and UDP ports 111 and 2019, on all networks assigned to the tenant.

```
$ qq multitenancy_modify_tenant \
  --id 2 \
  --enable-nfs
```

Configuring the NFS Protocol Globally or for a Tenant

You can use *global settings* that apply to all tenants. For example, to enable NFSv4.1 for all tenants, run the `qq nfs_modify_settings` command and use the `--enable-v4` flag. For more information about NFS settings see Enabling and Using NFSv4.1 on a Qumulo Cluster (page 231) and How NFSv4.1 works with Kerberos in Qumulo Core (page 252).

To override this global setting, run the `qq multitenancy_nfs_modify_settings` command to disable NFSv4.1 on a specific tenant, specify the tenant ID, and use the `--disable-v4` flag. In the following example, despite the global setting, the specified tenant no longer accepts NFSv4.1 traffic on port 2049, on all networks assigned to the tenant.

```
$ qq multitenancy_nfs_modify_settings \
  --tenant-id 2 \
  --disable-v4
```

To revert to the global setting for a specific tenant, run the `qq multitenancy_nfs_delete_settings` command and specify the tenant ID. For example:

```
$ qq multitenancy_nfs_delete_settings \
  --tenant-id 2
```

Configuring NFS Exports

To isolate clients within a network in specific directories in the file system, you can configure NFS exports for each tenant.

> **Note**
> Because each new tenant has no exports initially, to give clients within that tenant's network access to the cluster over NFS, you must add exports. For more information, see Create an NFS Export on Qumulo Care.

To create an NFS export for a tenant, run the `qq nfs_add_export` command and specify the tenant ID, the export path, and the file system path and use the `--no-restrictions` flag. In the following example, a cluster lets clients access directory `/my-directory` from the export `/my-export` when the clients are in the network assigned to the tenant.

```
$ qq nfs_add_export \
  --tenant-id 2 \
  --export-path /my-export \
  --fs-path /my-directory \
  --no-restrictions
```

> ⚠ Important
> When multiple tenants exist on a cluster, you must specify the tenant ID.

To move an NFS export from one tenant to another, run the `qq nfs_mod_export` command. In the following example, while the cluster denies access to clients in tenant 2, it lets clients in tenant 1 access the export `/my-export`.

```
$ qq nfs_mod_export \
  --tenant-id 2 \
  --export-path /my-export \
  --new-tenant-id 1
```

Enabling SMB and Configuring Settings and Shares

This section explains how to enable the SMB protocol for a tenant, the difference between global settings and settings for each tenant, and how to configure SMB shares for a tenant.

Enabling the SMB Protocol for a Tenant

To let a tenant use SMB to access a specific set of shares, run the `qq multitenancy_modify_tenant` command, specify the tenant ID, and use the `--enable-smb` flag.

> ℹ Note
> Initially, the tenant has no shares to mount.

In the following example, the cluster begins to accept SMB traffic on TCP port 445, on all networks assigned to the tenant.

```
$ qq multitenancy_modify_tenant \
  --id 2 \
  --enable-smb
```

Configuring the SMB Protocol Globally or for a Tenant

You can use *global settings* that apply to all tenants. For example, to require encryption for all tenants, run the `qq smb_modify_settings` command with the `--encryption-mode required` flag. For more information about SMB settings, see the following resources:

- Managing SMB3 Encryption in Transit in Qumulo Core (page 150)
- Using SMB Host Restrictions in Qumulo Core (page 95)
- Hiding an SMB Share (page 96)
- `qq smb_modify_settings` in the Qumulo `qq` CLI Command Guide (set the `--snapshot-directory-mode` flag to `visible`, `hidden`, or `disabled`)

To override this global setting, run the `qq multitenancy_smb_modify_settings` command to allow clients on only specific networks to require encryption. In the following example, despite the global setting, the specified tenant accepts unencrypted SMB requests on all networks assigned to the tenant.

```
$ qq multitenancy_smb_modify_settings \
  --tenant-id 2 \
  --encryption-mode none
```

To revert to the global setting for a specific tenant, run the `qq multitenancy_smb_delete_settings` command and specify the tenant ID. For example:

```
$ qq multitenancy_smb_delete_settings \
  --tenant-id 2
```

Configuring SMB Shares

To isolate clients within a network in specific directories in the file system, you can configure SMB shares for each tenant.

> **Note**
> Because each new tenant has no shares initially, to give clients within that tenant's network access to the cluster over SMB, you must add shares. For more information, see Create an SMB Share on Qumulo Care.

To create an SMB share for a tenant, run the `qq smb_add_share` command and specify the tenant ID, the share path, and the file system path and use the `--all-access` flag. In the following example, a cluster lets clients access access directory `/my-directory` from the share `/my-share` when the clients are in the network assigned to the tenant.

```
$ qq smb_add_share \
  --tenant-id 2 \
  --name /my-share \
  --fs-path /my-directory \
  --all-access
```

> ⚠ Important
> When multiple tenants exist on a cluster, you must specify the tenant ID.

To move an SMB share from one tenant to another, run the `qq smb_mod_share` command. In then following example, while the cluster denies access to clients in tenant 2, it lets clients in tenant 1 access the share `/my-share`.

```
$ qq smb_mod_share --tenant-id 2 \
  --name foo \
  --new-tenant-id 1
```

Using the Microsoft Management Console (MMC)

To create, modify, and delete SMB shares for each tenant, you can use the MMC Shared Folders snap-in. For more information, see Managing Qumulo Core File Shares by Using the Shared Folders MMC Snap-In (page 287).

> ⚠ Important
> Because folder paths always start at the root of the file system, using the MMC on a Windows client on a tenant's network causes the MMC to modify that tenant's shares.

Global Configuration of S3 and FTP Protocols

Qumulo Core doesn't permit configuring file system protocols such as S3 and FTP for each tenant. You must enable, disable, and configure these protocols globally.

> ⚠ Important
> Enabling these protocols makes them available on all networks.

For more information, see S3 API (page 290) in this guide and FTP in Qumulo Core on Qumulo Care.

Listing Client Connections

To determine tenant activity and client types connected to the tenant, run the `qq network_list_connections` command. For more information about balancing client connections to your Qumulo cluster, see `qq network_list_connections` in the Qumulo `qq` CLI Command Guide.

The following is example output.

```
[
  {
    "connections": [
      {
        "type": "CONNECTION_TYPE_NFS",
        "ip_address": "255.0.0.1",
          "tenant_id": 1
      },
      {
        "type": "CONNECTION_TYPE_SMB",
        "ip_address": "192.168.0.1",
        "tenant_id": 2
      }
    ],
    "id": 1
  },
  ...
]
```

Web UI

Setting the Qumulo Core 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 Qumulo Core Web UI.

To Set the Qumulo Core Web UI Login Banner

To set the login banner, run the `qq 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.html
```

To Clear the Qumulo Core Web UI Login Banner

To clear the login banner, run the `qq web_ui_modify_settings` command and use 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, run the `qq web_ui_get_settings` command and use the `--login-banner` flag.

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

Setting the Qumulo Core 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 Qumulo Core 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 Qumulo Core Web UI Inactivity Timeout

To set an inactivity timeout, run the `qq web_ui_modify_settings` command and use the `--inactivity-timeout` flag to specify the timeout in minutes. For example:

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

To Clear the Qumulo Core Web UI Inactivity Timeout

To clear an inactivity timeout, run the `qq web_ui_modify_settings` command and use 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, run the `qq web_ui_get_settings` command:

```
qq web_ui_get_settings
```

qq CLI

Getting Started with the qq CLI

This section explains how to download and get started with the `qq` CLI.

The `qq` CLI is a powerful tool that lets you configure, manage, and administer Qumulo clusters.

Prerequisites

The `qq` CLI works with Python 3.8 to 3.11.

> ☑ Tip
>
> - On Linux or macOS, run the `chmod +x qq` command to make the `qq` CLI executable.
> - On Windows, use the `python.exe` interpreter to run the `qq` CLI.

Downloading or Installing the qq CLI

This section explains how to download the `qq` CLI from your Qumulo cluster or how to install it by using the Python SDK.

To Download and Run the qq CLI from Your Qumulo Cluster

1. Log in to the Qumulo Core Web UI.
2. Click **APIs & Tools > Download qq Command-Line Tool for Python 3.8+**
3. Run the `qq` CLI.

- On Linux or macOS, run the `chmod +x qq` command to make the `qq` CLI executable.
- On Windows, use the `python.exe` interpreter to run the `qq` CLI.

To Install the qq CLI by Using the Python SDK

Run the `pip install qumulo_api` command.

Exploring Your Qumulo Cluster by Using the qq CLI

The best way to start exploring your Qumulo cluster by using the `qq` CLI is to learn about its most frequently used commands. For example:

- **`qq login`** : Connect to the IP address of one of the nodes in your cluster and log in.
- **`qq nodes_list`** : List information about the nodes in your cluster.
- **`qq version`** : Show the version of Qumulo Core running on your cluster.
- **`qq fs_read_dir`** : List the contents of a directory on your cluster.

- **`qq fs_write`** : Write a file to a directory on your cluster.

> **Tip**
> To get a complete list of `qq` CLI commands, run the `qq --help` command.

To Run qq CLI from a Remote Machine

Run the `qq login` command and specify the IP address of one of the nodes in your cluster and your credentials. For example:

```
qq --host 203.0.113.0 login \
  -u admin
  -p NW0bJbixtQcQzkq5q4sp
```

To Run qq CLI from a Node in Your Cluster

Use SSH to log in to one of the nodes in your cluster.

> **Note**
> It is possible to run `qq` CLI commands as the administrative Linux user. However, to do this, you must authenticate by using the `qq login` command.

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
   ```

   > **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 Core 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 doesn't 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.

Metadata

Managing User-Defined Metadata in Qumulo Core

This section explains how to create, retrieve, list, and delete user-defined metadata in Qumulo Core by using the `qq` CLI.

How User-Defined Metadata Works in Qumulo Core

Qumulo Core lets you add *user-defined metadata* to any file type stored in its file system. User-defined metadata comprises user-specified key-value pairs that have the following requirements:

- The key must be a Unicode string.
- The value must be a sequence of bytes.
- The total size of each key-value pair must be under 400 KB.

Keyspace Types and Functions

User-defined metadata in Qumulo Core is divided into `GENERIC` and `S3` *keyspaces*. Keyspaces work like *containers* for key-value pairs. The `S3` keyspace primarily supports the S3 API, which requires all files to have two sets of metadata in separate keyspaces.

Keyspaces can hold approximately 17 trillion key-value pairs and have the following requirements:

- All keys within a keyspace must be unique.
- The keyspace and key are required to create or access a user-defined metadata entry.

Managing User-Defined Metadata by Using the qq CLI

This section explains how to create, retrieve, list, and delete user-defined metadata by using the `qq` CLI.

> **Note**
>
> - All `qq` CLI commands default to using the `GENERIC` keyspace (page 129). For the `S3` keyspace, use the `--s3` flag.
> - In the following examples, you can specify the file path by using the `--path` flag or the file ID by using the `--id` flag.

Prerequisites

Managing user-defined metadata requires the following privileges:

- `READ_EA`: Read the user-defined metadata from a file
- `WRITE_EA`: Write to, or delete, the user-defined metadata of a file

To Create a Generic User-Defined Metadata Entry for a File by Using the qq CLI

Run the `fs_set_user_metadata` command and specify the path to the file, the key, and the value. For example:

```
qq fs_set_user_metadata \
  --path my-file \
  --key my-key \
  --value my-value
```

To specify a non-text value for the user-defined metadata, use the `--base64-value` or `--hex-value` flag.

For more information, see `qq fs_set_user_metadata` in the Qumulo `qq` CLI Command Guide.

To Retrieve a Generic User-Defined Metadata Entry for a File by Using the qq CLI

Use `fs_get_user_metadata` command and specify the path to the file and the key. For example:

```
qq fs_get_user_metadata \
  --path my-file \
  --key my-key
```

- To specify a non-text value for the user-defined metadata, use the `--base64-value` or `--hex-value` flag.
- To access the user-defined metadata within a file snapshot, use the `--snapshot` flag and specify the shapshot ID.

For more information, see `qq fs_get_user_metadata` in the Qumulo `qq` CLI Command Guide.

To List All Generic User-Defined Metadata Entries for a File by Using the qq CLI

Run the `fs_list_user_metadata` command and specify the path to the file. For example:

```
qq fs_list_user_metadata \
  --path my-file
```

- To specify a non-text value for the user-defined metadata, use the `--base64-value` or `--hex-value` flag.
- To access the user-defined metadata within a file snapshot, use the `--snapshot` flag and

specify the shapshot ID.

For more information, see `qq fs_list_user_metadata` in the Qumulo `qq` CLI Command Guide.

To Delete a Generic User-Defined Metadata Entry for a File by Using the qq CLI

Run the `fs_delete_user_metadata` command and specify the path to the file and the key. For example:

```
qq fs_delete_user_metadata \
  --path my-file \
  --key my-key
```

For more information, see `qq fs_delete_user_metadata` in the Qumulo `qq` CLI Command Guide.

Managing User-Defined Metadata by Using the S3 API

S3 categorizes metadata as:

- Metadata
  - Immutable metadata that remains for the life of the object.
  - Qumulo Core maps metadata to the `S3` keyspace (page 129).
- Tags
  - Mutable metadata that doesn't impact the object's entity tag.

    > **⚠ Important**
    > Tag values that can't be encoded by using UTF-8 aren't visible to S3.

  - Qumulo Core maps tags to the `GENERIC` keyspace (page 129).

In Qumulo Core 6.3.2 (and higher) the Qumulo S3 API (page 0) supports user-defined metadata fully. For more information about how to access metadata by using the S3 API, see the Amazon Simple Storage Service API Reference.

Snapshots

How Snapshots Work in Qumulo Core

This section explains snapshots, their storage usage, and their locking functionality in Qumulo Core.

How Snapshots Work

Qumulo Core 2.5.0 (and higher) can take instant snapshots of the file system. A *snapshot* is an entry for every version of file system elements such as files, directories, creation and modification timestamps, permissions, and so on. Each new entry points only to changed data and, to allow original and new entries to share data, Qumulo Core writes the entries alongside each other.

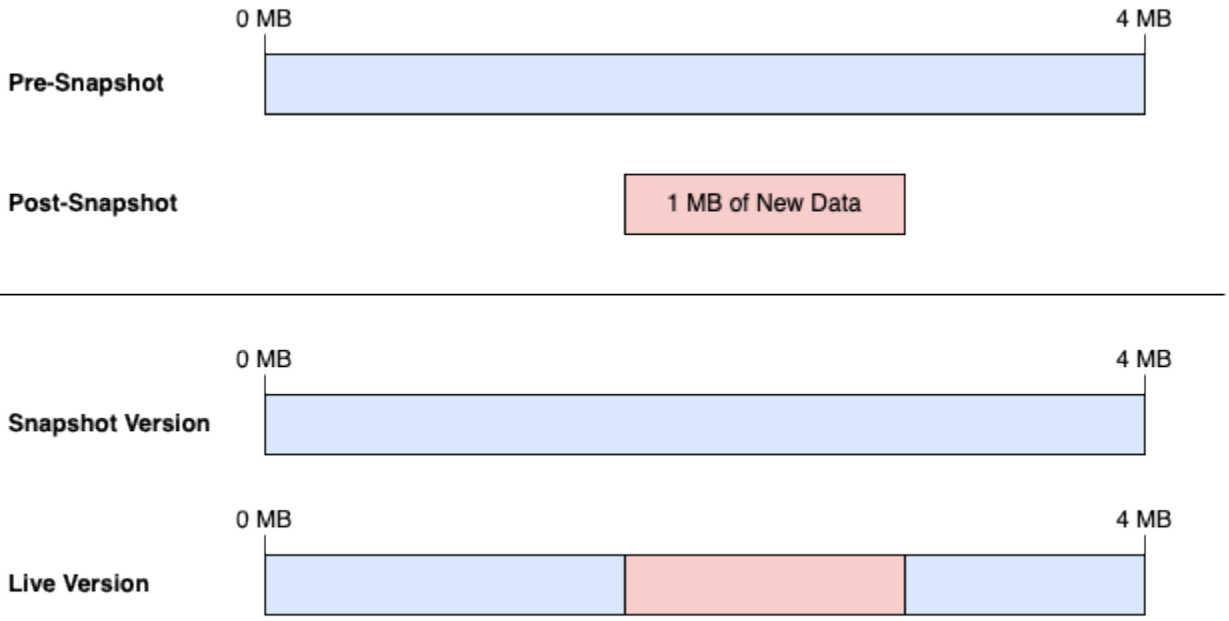
Taking a snapshot doesn't consume storage or incur a performance penalty. There is only a negligible performance penalty for reading and writing snapshotted file system data.

How Snapshots Grow Over Time

The following example shows how Qumulo Core allocates storage to data and links it to file metadata as file system data changes.

In this scenario:

1. A user creates a file with 4 MB of data.
2. Qumulo Core takes a snapshot of the file.
3. A user modifies 1 MB of data within the file.
4. Qumulo Core allocates a new 1 MB region to the modified data.



Now, the following is true:

- **5 MB:** The total storage that the file occupies
    - **3 MB:** Data shared between the original and new versions of the file
    - **1 MB:** Original data that exists only in the *saved* (snapshotted) version of the file
    - **1 MB:** New data that exists only in the *live* (latest) version of the file

Next, the following conditions take effect:

- If the user rewrites that particular 1 MB of data, the system overwrites the existing live data without allocating new space.
- If the user rewrites a different region of the file, Qumulo Core allocates additional storage.

Determining Snapshots' Storage Usage

When Qumulo Core tracks the difference between the *saved* (snapshotted) and *live* (latest) versions of a file, it creates a *lineage* of snapshots independent from each other. To determine the amount of data that a single snapshot references, run the `qq snapshot_get_capacity_used_per_snapshot` command and specify the snapshot ID. For example:

```
qq snapshot_get_capacity_used_per_snapshot \
  --id 1682119059
```

More than one snapshot can reference *covered data*. It isn't possible to release covered data until you delete all *covering snapshots* that reference it.

- To determine the total covered data, including data no longer present in the snapshot, run the `qq snapshot_get_capacity_used_per_snapshot` command and specify multiple, comma-separated snapshot IDs.
- To determine the total amount of data, including covered data that multiple snapshots reference, run the `qq snapshot_get_total_used_capacity` command.

When you delete a snapshot, Qumulo Core removes the data which that snapshot references but retains the data which any other snapshot references. This ensures a full file representation within the remaining snapshots. Qumulo Core uses a background process to recover the storage that the snapshot had consumed.

> **Note**
> When you delete a snapshot, the background process might take some time. To track the reclaimed storage, run the `qq snapshot_get_total_used_capacity` command.

Example: Tracking Covering Snapshots and Data Changes

For example, if you run the `qq snapshot_get_total_used_capacity` command, Qumulo Core shows that storage usage is 1,319,413,953,331 Bytes (1.2 TiB). This amount includes the total snapshot data and the covering snapshots.

If you add up the usage for all snapshots currently in the file system (by using the `qq snapshot_get_capacity_used_per_snapshot` command), Qumulo Core shows that total snapshot storage usage is 2,147,483,648 Bytes (2 GiB). This amount includes the data changes that each snapshot stores but doesn't include the unchanged file portions within each snapshot.

Example: Tracking File Snapshot Changes Over Time

For example, you have a 1 TiB file that you modify over time.

- **Snapshot 1:** This snapshot is 1,099,511,627,776 Bytes in size and contains the full 1 TiB file.
- **Snapshot 2:** This snapshot is 1,073,741,824 Bytes in size and contains 1 GiB of data changes.
- **Snapshot 3:** This snapshot is 1,073,741,824 Bytes in size and contains an additional 1 GiB of data changes.

If you delete snapshot 1, only 1,023 GiB of data (covered by snapshots 2 and 3) remain. Qumulo Core doesn't release this 1,023 GiB of data until you delete all snapshots that reference the original file.

> **Note**
> Without the data that snapshots 2 and 3 cover, no full file representation is possible.

Managing Snapshots in Qumulo Core

This section explains how to create on-demand snapshots and snapshot policies, view and search for existing snapshots, and delete snapshots by using the Qumulo Core Web UI. It also explains how to create snapshots on a schedule, create a snapshot with an expiration time, and modify a snapshot's expiration time.

Managing Snapshots by Using the Qumulo Core Web UI

This section explains how to create on-demand snapshots and snapshot policies, view and search for existing snapshots, and delete snapshots by using the Qumulo Core Web UI.

To Create an On-Demand Snapshot

1. Log in to the Qumulo Core Web UI.
2. Click **Cluster** > **Saved Snapshots**.
3. On the **Saved Snapshots** page, in the upper right, click **Take Snapshot**.
4. In the **On Demand Snapshot** dialog box, do the following.
    a. Enter the **Snapshot Name**.
    b. For **Apply to Directory**, enter the directory to snapshot.
    c. For **Delete Snapshot**, specify whether Qumulo Core should never delete the snapshot or delete it after a specified time period.
    d. Click **Save**.

To Create a Snapshot Policy

1. Log in to the Qumulo Core Web UI.
2. Click **Cluster** > **Policy**.
3. On the **Snapshot Policies** page, in the upper right, click **Create Policy**.
4. On the **Create Snapshot Policy** page:
    a. Enter the **Policy Name**.
    b. For **Apply to Directory**, enter the directory to snapshot.
    c. In the **Run Policy on the Following Schedule** section, specify the snapshot frequency and when to delete snapshots.
    d. Click **Enable policy upon creation**.
    e. Click **Create Policy**.

To View Existing Snapshots

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

1. Log in to the Qumulo Core Web UI.
2. Click **Cluster** > **Saved Snapshots**.
3. In Qumulo Core version 4.3.3 (and higher), if you have more than 50 snapshots, use 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 for each page.

To Find a Specific Snapshot

In Qumulo Core version 4.3.3 (and higher), you can search for a specific snapshot by name, creation time, and so on.

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

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

   The table rows match your query as you type.
5. (Optional) To turn off filtering, click **disable filters**.

To Delete a Single Snapshot

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

To Delete Multiple Snapshots

In Qumulo Core version 4.3.3 (and higher), you can delete multiple snapshots at once.

1. Log in to the Qumulo Core Web UI.
2. Click **Cluster** > **Saved Snapshots**.
3. On the left side of the table, select every snapshot to delete.

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

> **Note**
> Because all selection and deletion controls operate only on the current page, it isn't possible to delete a snapshot accidentally if it isn't listed on the current page.

Managing Snapshots by Using the qq CLI

This section explains how to create snapshots on a schedule, create a snapshot with an expiration time, and modify a snapshot's expiration time by using the `qq` CLI.

> **Important**
> Creating and modifying snapshot policies with an associated lock requires the `SNAPSHOT_LOCK` permission in addition to policy permissions.

Creating Snapshots on a Schedule by Using a Snapshot Policy

Run the `qq snapshot_create_policy` command to create a snapshot policy and specify the interval at which Qumulo Core takes and deletes snapshots.

In the following example, we create a policy named `every_day` that takes a snapshot every midnight in the Pacific time zone and retains the snapshot for two days. Every new snapshot that this policy creates is locked with a key named `my-key-name`. For more information, see Locking and Unlocking Snapshots in Qumulo Core (page 140).

> **Note**
> The `timezone` flag uses values from the tz database. If you don't specify a time zone, the snapshot policy uses UTC time.

```
qq snapshot_create_policy daily \
  --name every_day \
  --days-of-week all \
  --at 00:00 \
  --timezone America/Los_Angeles \
  --time-to-live 7days
  --lock-key my-key-name
```

In the following example, we change a previously created policy with ID `1` to a policy named `hourly` that takes a snapshot every hour, but only during business hours (Monday to Friday, 8am to 6pm in the Pacific time zone), and retains snapshots for two days. Every new snapshot that this policy creates is unlocked (previously created snapshots remain locked). For more information, see Locking and Unlocking Snapshots in Qumulo Core (page 140).

```
qq snapshot_modify_policy change_to_hourly_or_less \
  -i 1 \
  --name hourly \
  --period 1hours \
  --days-of-week MON,TUE,WED,THU,FRI \
  --start-time 08:00 \
  --end-time 18:00 \
  --timezone America/Los_Angeles \
  --time-to-live 2days
  --clear-lock-key
```

Creating an On-Demand Snapshot with an Expiration Time

Run the `qq_snapshot_create_snapshot` command to specify an expiration date or expiration time before Qumulo Core deletes the snapshot.

> **Note**
> If you don't specify an expiration date or expiration time before deletion, Qumulo Core never deletes the snapshot.

In the following example, the snapshot expires on December 31, 2030, at midnight, in UTC time.

```
qq snapshot_create_snapshot \
  --expiration 2030-12-31T00:00:00Z
```

In the following example, Qumulo Core deletes the snapshot in one year from the snapshot's creation time.

```
qq snapshot_create_snapshot \
  --time-to-live 12months
```

Modifying a Snapshot's Expiration Time

Run the `qq snapshot_modify_snapshot` command and specify the snapshot ID from the **Saved Snapshots** page in the Qumulo Core Web UI (for example, for the `1234567_replication_from_prod` snapshot, the ID is `1234567`).

In the following example, the snapshot with ID `1234567` never expires.

```
qq snapshot_modify_snapshot \
  -i 1234567
  -e ''
```

In the following example, the snapshot with ID `1234567` expires after one month.

```
qq snapshot_modify_snapshot \
  -i 1234567 \
  -t 1month
```

Locking and Unlocking Snapshots in Qumulo Core

This section explains how to lock or unlock a snapshot by using a key located in the Qumulo file system key store and the `qq` CLI. In addition, it explains how to lock policy-created snapshots for local policies and for policies that are part of a replication target relationship.

For more information, see Managing Security Keys in the Qumulo File System Key Store (page 159).

Locking and Unlocking Snapshots

> **Important**
>
> - Unlocking a snapshot requires a cryptographic signature. Before you lock a snapshot, make sure that you have access to your private keys and that you understand the unlocking procedure.
> - It isn't possible to delete or shorten the expiration time of a locked snapshot. However, you can extend the expiration time of a locked snapshot.
> - Qumulo Core removes both locked and unlocked snapshots at their expiration time automatically.

In Qumulo Core 6.1.0.3 (and higher), you can lock a snapshot by using a key located in the Qumulo file system key store (page 159). You can also ensure that a snapshot policy locks all new snapshots with a particular key (page 137) by associating the key with the snapshot policy.

In Qumulo Core 6.1.1 (and higher), you can ensure that a replication target relationship locks all new policy snapshots with a specific key (page 142) by associating the key with the replication target.

To Lock a Snapshot by Using the qq CLI

Run the `qq snapshot_lock_snapshot` command and specify the snapshot ID and either the key ID or key name. For example:

```
qq snapshot_lock_snapshot \
  --id 1682119059 \
  --lock-key my-key-name
```

To Unlock a Snapshot by Using the qq CLI

Unlocking a snapshot requires proving that you can sign a challenge by using the same key that locked the snapshot. You can do this by using either of the following methods.

If You Have Direct Access to the Private Key

> **Note**
> To use this method, you must install the Python `cryptography` library.

Run the `qq snapshot_unlock_snapshot` command and specify the snapshot ID and the path to the private key file. For example:

```
qq snapshot_unlock_snapshot \
  --id 1682119059 \
  --private-key-file /path/to-my-file.pem
```

If You Don't Have Direct Access to the Private Key

If you can use the private key only to sign data, take the following steps.

1. To receive the unlock challenge, run the `qq snapshot_get_unlock_challenge` command and specify the snapshot ID. For example:

   ```
   qq snapshot_get_unlock_challenge \
     --id 1682119059
   ```

   > **Important**
   > If you change a snapshot's expiration time while the snapshot is locked, Qumulo Core changes the unlock challenge for the snapshot.

2. To generate a verification signature, use the response from the challenge with your private key.

   For more information about creating a verification signature by using a private key or key management service, see Signing a Security Challenge by Using an ECDSA Private Key (page 156).

3. To unlock the snapshot, run the `qq snapshot_unlock_snapshot` command and specify the snapshot ID and the Base64-encoded unlock challenge that your private key signed. For example:

```
qq snapshot_unlock_snapshot \
  --id 1682119059 \
  --signature "VGhpcyBpcyBteSB1bmxvY2sgY2hhbGxlbmdlLg=="
```

Associating a Lock Key with a Replication Target Relationship

To lock all policy-created snapshots by using a lock key, you can associate the key with a replication target relationship. Consider the following system behavior:

- Qumulo Core locks only policy-created snapshots that have an expiration time.
- If you reverse the relationship by switching the source and target, the new target can't use the existing key and you must set a key for the new target. However, if you revert the relationship by returning the source and target to their original assignments, Qumulo Core lets you use the key from the original source-target relationship.
- If a target replication relationship uses a key, you can't disable or delete the key, unless you reverse the relationship.
- If you disable or delete a key while a target replication relationship is reversed and then return the source and target to their original assignments, you must set a new key to be able to lock future snapshots.

To Associate a Lock Key with a Replication Target Relationship

Run the `qq replication_set_target_relationship_lock` command and specify the relationship ID and key name or ID. For example:

```
qq replication_set_target_relationship_lock \
  --relationship-id 12345a6b-7c89-0d12-3456-78fe9012f345
  --lock-key my-key-name
```

To Disassociate a Lock Key from a Replication Target Relationship

Run the `qq replication_set_target_relationship_lock` command and specify the relationship ID and and `--clear-lock-key` flag. For example:

```
qq replication_set_target_relationship_lock \
  --relationship-id 12345a6b-7c89-0d12-3456-78fe9012f345
  --clear-lock-key
```

Recovering Files by Using Snapshots

This section explains how to use snapshots to recover files.

In Qumulo Core 2.5.0 (and higher), you can recover files by accessing the hidden `.snapshot` directory over SMB or NFS.

Inside the `.snapshot` directory, directories with snapshot IDs represent various snapshots. The *modified* timestamp of a directory is the time at which Qumulo Core took the snapshot.

> **Note**
>
> - When you use NFS on Linux and macOS (even if you configure your system to show hidden files), the `.snapshot` directory doesn't appear when you list a directory's contents. You must navigate to the `.snapshot` directory explicitly.
> - When you use SMB, the `.snapshot` directory appears only at the root of the share in Finder or File Explorer. In other directories, you must navigate to the `.snapshot` directory explicitly.

To Recover Files on Linux or macOS by Using the Command Line

1. Navigate to the `.snapshot` directory. For example:

   ```
   cd /Volumes/MyShareName/.snapshot
   ```

2. Locate the file or directory to recover and copy it to a new location.

> **Tip**
> To see the `.snapshot` directory at the root of the share, show hidden files by pressing ⌘ + Shift + .

To Recover Files on macOS by Using Finder

1. On the Finder menu, click **Go > Go to Folder...**.
2. In the dialog box, enter the path to the `.snapshot` directory. For example:

   ```
   /Volumes/MyShareName/.snapshot
   ```

> **Note**
> You must specify the `.snapshot` directory from the root of the share.

3. Locate the file or directory to recover and copy it to a new location.

To Recover Files on Windows by Using File Explorer

1. On Windows 7 (and higher), configure Windows Explorer (or File Explorer) as follows:
    a. Disable **Hide protected operating system files**.
    b. Enable **Show hidden files, folders, and drives**.
2. Navigate to the `.snapshot` directory.
3. Locate the file or directory to recover and copy it to a new location.

Encryption and Data Security

Managing Encryption at Rest in Qumulo Core

This section explains how encryption at rest works in Qumulo Core, how to rotate master keys, how to configure a Key Management Server (KMS), and how to ensure that the master keys across your cluster are secured correctly by using the `qq` CLI.

> ⚠ Important
>
> - Upgrading a Qumulo cluster from a version of Qumulo Core lower than 3.1.5 doesn't enable encryption automatically. You must rebuild your cluster to take advantage of this feature. When you create a new cluster, Qumulo Core enables encryption automatically and distributes the master key to all nodes in the cluster.
> - In case of replication processes, Qumulo Core maintains the encryption type after data transfers. Although source and target clusters don't require encryption for replication, we strongly recommend encrypting both source and target clusters.

How Encryption at Rest and Master Keys Work in Qumulo Core

In Qumulo Core 3.1.5 (and higher), in addition to encrypting data *in transit* (for example, to clients that use SMBv3.1), software-based encryption also secures data *at rest* for on-premises clusters. Qumulo Core encrypts all data and metadata in the file system. Removing or reinserting drives and replication doesn't affect encryption at rest. For more information, see Encryption Limitations (page 149).

Qumulo Core uses a *master key* to protect the *data key* that encrypts the data on the cluster. The master key is stored either locally—on the boot drive of every node, in a file that only the `root` user can access—or on an external Key Management Server (KMS)—from where the system retrieves the master key upon startup. Both approaches help protect your data from potential threats such as a malicious actor's access to stolen or decommissioned disks.

Retrieving Information about a Qumulo Cluster's Encryption Configuration

This section explains how to retrieve the status or detailed information about an active encryption configuration for a Qumulo cluster and gives examples for a system that uses a locally stored master key and a system that uses a Key Management Server (KMS).

> ℹ Note
>
> The `qq encryption_get_key_store` and `qq encryption_get_status` commands require the `PRIVILEGE_ENCRYPTION_READ` privilege.

To View the Status of an Active Encryption Configuration

Run the `qq encryption_get_status` command.

The following is example output. The master key is stored locally.

```
{
  "last_key_rotation_time": "2022-11-20T12:15:25.683207795Z",
  "status": "Encrypted",
  "type": "Local"
}
```

The following is example output. The master key is stored in a KMS.

```
{
  "ca_cert_expiry": "2027-04-18T19:55:17Z",
  "client_cert_expiry": "2027-04-18T19:55:17Z",
  "last_key_rotation_time": "2023-09-05T20:15:40.06864014Z",
  "last_status_update_time": "2023-09-05T20:28:58.108120131Z",
  "status": "KMS Available",
  "type": "KMS"
}
```

To View Detailed Information for an Active Encryption Configuration

Run the `qq encryption_get_key_store` command.

The following is example output. The master key is stored locally.

```
{
  "config_details": {
    "status": "Encrypted"
  },
  "config_type": "Local"
}
```

The following is example output. The master key is stored in a KMS.

```
{
  "config_details": {
    "config_creation_time": "2024-02-28T20:01:25.683207795Z",
    "hostname": "kms.example.com",
      "key_id": "abcd-1234-efgh-5678",
      "port": 5696
    },
  "config_type": "KMS"
}
```


Configuring Qumulo Core to Use a Master Key Stored Locally or in a Key Management Server (KMS)

This section explains how to configure Qumulo Core to use a master key stored locally or in a Key Management Server (KMS) by using the `qq` CLI.

> **Note**
>
> - The `qq` CLI command `qq encryption_set_key_store` requires the `PRIVILEGE_ENCRYPTION_WRITE` privilege.
> - To be able to configure an external KMS, the KMS must support Key Management Interoperability Protocol (KMIP) 1.0.

To Configure Qumulo Core to Use a Master Key Stored Locally

> **Important**
>
> - While the *master key* on your boot drive encrypts your *data keys*, the master key *itself* isn't encrypted.
> - The boot drive contains the disk image, the installed build of Qumulo Core, and configuration files. In the unlikely event that your boot drive fails and requires replacement, remove the encrypted data keys associated with the master key from the boot drive by rotating the master key (page 148). When you complete the key rotation process, you can dispose of the failed boot drive securely.
> - To avoid potential decryption, ensure that your data keys eventually *age out* by rotating the master key any time you replace a drive in your cluster.

1. To configure the system to use a local key store, run the `qq encryption_set_key_store` with the `local` subcommand.
2. To confirm that the system is configured correctly, `qq encryption_get_status` .

In the output, ensure that the `type` field is set to `Local`.

To Configure Qumulo Core to Use a Master Key Stored in a Key Management Server (KMS)

> **Caution**
>
> - If the master key is deleted from the KMS, and all nodes in the cluster are rebooted, all data on the cluster becomes permanently unrecoverable.
> - If you allow the certificates to expire, or the master key is deleted accidentally, you must create a new, valid configuration as soon as possible. To warn you of this scenario, the Qumulo Core Web UI indicates if any of your certificates are about to expire, or if the configured master key becomes unavailable.

1. To configure the system to use a KMS, use `qq encryption_set_key_store kms` command and specify the path to the client certificate, private key, the server CA certificate, the key ID, and the KMS server hostname. For example:

   ```
   qq encryption_set_key_store kms \
     --client-cert path/to/client_cert.pem \
     --client-private-key path/to/client_pk.pem \
     --server-ca-cert /path/to/server_cert.pem \
     --key-id abcd-1234-efgh-5678 \
     --host-name kms.example.com
   ```

2. To confirm that the system is configured correctly, run the `qq encryption_get_key_store` command.

   In the output, ensure that the `type` field is set to `KMS`.

Rotating the Master Key

This section explains how to rotate the master key and check the encryption status for your cluster by using the `qq` CLI and how to check the encryption status by using the Qumulo Core Web UI.

> **Caution**
>
> Qumulo Care team members can help you rotate your master keys (page 148). However, they don't have access to your encryption keys and can't retrieve them for you.

To Rotate Master Keys Stored Locally

1. Run the `qq rotate_encryption_keys` command.

   When the process is complete, the command shows the `Key rotation complete` message.

2. To view your cluster's encryption status and the last key rotation time, run `qq encryption_get_status` .

To Rotate Master Keys Stored in a Key Management Server (KMS)

1. Run the `qq rotate_encryption_keys` command and specify the key ID. For example:

   ```
   qq rotate_encryption_keys --key-id abcd-1234-efgh-5678
   ```

   > ☑ Tip
   > The key ID might be different from the key name.

2. To ensure that the system is using the new key, run `qq encryption_get_key_store` .

   In the output, ensure that the `key_id` field lists the new key ID.

To Check the Encryption Status of a Qumulo Cluster by Using the Qumulo Core Web UI

1. Log in to the Qumulo Core Web UI.
2. On the **Dashboard** page, in the **Cluster Overview** section, click **More details**.
3. If encryption is enabled for your cluster, the **Cluster** page shows the message **Data Encrypted**.

Encryption Limitations

- Qumulo Core doesn't encrypt host file system data on the node (such as system logs, core files, and so on).
- Qumulo Core doesn't support removing encryption from encrypted clusters.
- On encrypted systems, single-stream throughput and latency might experience up to 5-10% degradation for writes and up to 5% for reads.

Managing SMB3 Encryption in Transit in Qumulo Core

This section explains how to manage SMB3 encryption for individual shares or entire clusters in Qumulo Core 2.14 (and higher).

To confirm the settings for your cluster from the Qumulo Core Web UI, click **Sharing** > **SMB Shares** > **SMB Settings**. By default, Qumulo Core supports AES-128-GCM and AES-128-CCM encryption, sets cluster-level SMB encryption to **None** and share-level encryption to **Unencrypted**.

For all clusters created by using Qumulo Core 3.1.5 (and higher), Qumulo Core enables at-rest encryption automatically (page 0).

> **Note**
>
> - Clients that connect to your cluster can send encrypted or unencrypted packets when your cluster doesn't require encryption.
> - It isn't necessary to use signing as a share-level protection mechanism if you set **Require Encryption** for a specific SMB share or if you configure cluster-level SMB encryption.

How Cluster-Level and Share-Level Encryption Settings Interact in Qumulo Core

The following table explains the possible levels of encryption of clusters and shares and the relationships between them.

Cluster Encryption Level	Unencrypted Share	Encrypted Share
No Encryption	Clients can send unencrypted or encrypted packets	• Clients must send encrypted packets • Unencrypted clients are disconnected
Prefer Encryption	Client can send unencrypted or encrypted packets.	• Clients must send encrypted packets • Unencrypted clients are disconnected

Cluster Encryption Level	Unencrypted Share	Encrypted Share
Require Encryption	• Clients must send encrypted packets • Unencrypted clients are disconnected	• Clients must send encrypted packets • Unencrypted clients are disconnected

Configuring Cluster-Level and Share-Level Encryption

This section explains how to configure cluster-level encryption in Qumulo Core by using the Web UI and `qq` CLI and how to configure share-level encryption by using the `qq` CLI.

To Configure Cluster-Level Encryption by Using the Qumulo Core Web UI

1. Log in to the Qumulo Core Web UI.
2. Click **Cluster** > **SMB Settings**.
3. On the **SMB Settings** page, select an encryption level.

   The Web UI shows any unencrypted shares on your cluster.
4. Click **Configure SMB**.

To Configure Cluster-Level and Share-Level Encryption by Using the qq CLI

For information about configuring cluster-level and share-level encryption by using the `qq` CLI, see the following sections in the Qumulo `qq` CLI Command Guide.

- Cluster-Level Encryption: `qq smb_modify_settings`
- Share-Level Encryption: `qq smb_mod_share`

Disabling SMB3 Negotiation to Improve Workload Performance

Clients that connect to your cluster can send encrypted or unencrypted packets when your cluster doesn't require encryption. In certain scenarios, compared to unencrypted configurations, while workflows triggered by pipelines can experience a slight performance degradation, synchronized operations can experience a more significant drop in performance.

To avoid potential performance impact, you can prohibit Qumulo Core from advertising its encryption capabilities by turning off SMB3 negotiation.

For more information, see `qq smb_modify_settings` in the Qumulo `qq` CLI Command Guide.

Checking Encryption of SMB3 Session

To check whether an SMB3 client session is encrypted, run the `Get-SmbConnection` PowerShell command. For example:

```
Get-SmbConnection | Select-Object -property *
```

The following is example output.

```
SmbInstance : Default
ContinuouslyAvailable : False
Credential : SILENCE\jcage
Dialect : 3.0
Encrypted : False
NumOpens : 2
Redirected : False
ServerName : qq
ShareName : Files
Signed : True
UserName : SILENCE\jcage
PSComputerName :
CimClass : ROOT/Microsoft/Windows/SMB:MSFT_SmbConnection
CimInstanceProperties : {ContinuouslyAvailable, Credential, Dialect, Encrypted...}
CimSystemProperties : Microsoft.Management.Infrastructure.CimSystemProperties
```

Generating and Storing ECDSA Keys on a Qumulo Cluster

This section explains how to generate Elliptic Curve Digital Signature Algorithm (ECDSA) keys and ECDSA verification signatures that are compatible with the Qumulo file system key store.

In Qumulo Core 6.1.0 (and higher), you can store multiple ECDSA public keys in the Qumulo file system key store and use these keys to protect file system resources.

> ⚠ Important
>
> - Currently, Qumulo Core supports only 256-bit ECDSA keys in `.pem` and `.der` formats. Qumulo Core doesn't support storing ECDSA keys of other lengths and formats.
> - No KMS system shows the private key. To sign messages later, write down the key ID in the responses from key generation commands.

Generating an ECDSA Private Key

This section explains how to generate a 256-bit ECDSA private key by using Linux CLI tools and AWS, GCP, and Azure CLI or API.

To Generate a Private Key by Using Linux CLI Tools

To generate a key in the `.pem` format, run the `openssl` or `ssh-keygen` tools.

- Run the `openssl` tool and specify the path to the private key. For example:

```
openssl ecparam \
  -genkey \
  -name prime256v1 \
  -out /private-key-path
```

- Run the `ssh-keygen` tool and specify the path to the private key. For example:

```
ssh-keygen \
  -f /private-key-path \
  -t ecdsa \
  -m PEM
```

The following is an example private key.

```
-----BEGIN EC PRIVATE KEY-----
EXAMPLEabCDe8fghi8J28KlB8mO+no93N0pBqrs/TUvWXYza4BC0DefghiJklmNO
PQRsTUVWXyZAbc0DEFGhIJ////////////////////////////////7///kl
MNOPQRSTUVwXYZA5bcD++ef7gHIjKlMOnopQRst82u3OVWxY8oZaBcdEfEg62hij
k8LmNoP7/Q4RSTu9V7WXyzABCDeF0G/7HIJ4KlMN////////////////////opq
3OrsTUV7w9XyzAB2CDEFGHIjKLMNOPQ7r7S7t/uVwxY1zaBO9c23Sd1EF3G3hijk
L1mN1OPQRSTUve2+X06YZABcD/eFGhIJ/Kl5MNOPQrsTuEXAMPLE
-----END EC PRIVATE KEY-----
```

Generating a Key Pair by Using the AWS Key Management Service (KMS)

Use the AWS Management Console, AWS CLI, or AWS KMS API. For more information, see the following resources:

- Console: Creating asymmetric KMS keys
- AWS CLI: `create-key` in the AWS CLI Command Reference.
- AWS KMS API: `CreateKey` in the AWS Key Management Service API Reference

When you create a key pair, specify the following details:

- Key Type: Asymmetric key
- Usage: Sign and verify
- Key Specification: ECC_NIST_P256

Generating a Key Pair by Using the GCP Cloud Key Management Service (Cloud KMS)

Use the GCP Cloud Console, Cloud CLI, or Cloud API. For more information, see Create a key in the Cloud Key Management Service documentation.

When you create a key pair, specify the following details:

- Protection Level: software or HSM
- Purpose: Asymmetric sign
- Algorithm: Elliptic Curve P-256 - SHA256 Digest

Generating a Key Pair by Using the Azure Key Vault

Use the Azure Key Vault and the Azure CLI. For more information, see `az keyvault key create` in the Azure documentation.

When you create a key pair, specify the following details:

- Key Type: EC
- Curve: P-256
- Key Size: 256

Extracting the Public Key from an ECDSA Private Key

After you create a 256-bit ECDSA private key, you can extract a public key from it by using Linux CLI tools and AWS, GCP, and Azure CLI and API. You can store the public key in the Qumulo file system key store (page 159).

To Extract the Public Key by Using Linux CLI Tools

1. Run the `openssl` tool and specify the path to the private key and the path for saving the public key. For example:

```
openssl pkey \
  -in /private-key-path \
  -pubout > /public-key-path
```

2. If your private key is in OpenSSH format, export the public key into the `.pem` format. Run the `ssh-keygen` tool and specify the path to the private key and the path for saving the public key. For example:

```
ssh-keygen \
 -e \
 -f /private-key-path \
 -m PEM > /public-key-path
```

3. To convert your private key to `.pem` format, run the `ssh-keygen` tool and specify the path to the private key. For example:

```
ssh-keygen \
  -p \
  -f /private-key-path \
  -m pem
```

The following is an example public key in `.pem` format.

```
-----BEGIN PUBLIC KEY-----
EXAMPLEabCDef0GHIJKL4MNOPqRStUV5wXyz491abc1d2efGijklmNOP0qrsTUv
WXYza1BCdEfGHIjkOlMnOpqr1STUvW3XYzAB6c8DefghIJkEXAMPLE==
-----END PUBLIC KEY-----
```

Extracting the Public Key by Using the AWS Key Management Service (KMS)

Use the AWS Management Console, AWS CLI, or AWS KMS API. For more information, see the following resources:

- Console: Displaying KMS key details
- AWS CLI: `get-public-key`
- AWS API: `GetPublicKey`

Extracting the Public Key by Using the GCP Cloud Key Management Service (Cloud KMS)

Use the GCP Cloud Console, Cloud CLI, or Cloud API. For more information, see Retrieve the public key in the Cloud Key Management Service documentation.

Extracting the Public Key by Using the Azure Key Vault

Use the Azure Key Vault and the Azure CLI. For more information, see `az keyvault key download` in the Azure documentation.

Signing a Security Challenge by Using an ECDSA Private Key

When you perform actions such as adding a new key to the Qumulo file system key store, replacing an existing key in the key store, or unlocking a snapshot, you must verify that you have access to the private key by signing a security challenge.

You can use your private key to generate a verification signature and then provide this signature to Qumulo Core in Base64 encoding.

> **Note**
>
> - When you add a new key, the key name is the security challenge.
> - When you replace an existing key or unlock a snapshot, the `qq` CLI command provides the challenge.

To Sign a Security Challenge by Using Linux CLI Tools

1. Save the security challenge to be signed to a file. For example:

```
echo -n "This is my challenge." > /tmp/challenge.out
```

> **Note**
> The `-n` flag ensures that there are no newline characters following the challenge.

2. To sign the challenge, run the `openssl` tool and specify the path to the private key. For example:

```
openssl dgst \
  -sha256 \
  -r \
  -sign /path-to-private-key \
  -out /tmp/signature.sha256 /tmp/challenge.out
```

3. To encode the signature in Base64 format, run the `openssl` tool. For example:

```
openssl base64 \
  -in /tmp/signature.sha256 \
  -out /tmp/key_signature.b64
```

To Sign a Security Challenge by Using the AWS Key Management Service (KMS)

1. Sign a security challenge by using the AWS CLI or AWS KMS API. For more information, see the following resources:

- AWS CLI: `sign` in the AWS CLI Command Reference
- AWS KMS API: `Sign` in the AWS Key Management Service API Reference

2. Specify the `ECDSA_SHA_256` algorithm.

   The response returns a Base64-encoded verification signature.

To Sign a Security Challenge by Using the GCP Cloud Key Management Service (Cloud CMS)

1. Sign a security challenge by using the GCP Cloud Console, Cloud CLI, or Cloud API. For more information, see Creating a signature in the Cloud Key Management Service documentation.
2. Specify the `SHA256` digest algorithm.
3. If the signature in the response is comprised of raw bytes (not a Base64-encoded string) encode the signature file by using the `base64` CLI tool on the signature file that the Cloud CLI generates. For example:

```
base64 /gcp-output-path/signature.bytes
```

To Sign a Security Challenge by Using the Azure Key Vault

> ⚠ Important
> The Azure API requires a security challenge as a UrlBase64-encoded SHA-256 digest.

1. Sign a security challenge by using the Azure Key Vault and the Azure API. For more information, see `sign` in the Azure documentation.
2. Convert your plaintext challenge into the correct format. For example:

```
echo -n "This is my challenge." \
  | sha256sum \
  | cut -d' ' -f1 \
  | xxd -r -p \
  | base64 \
  | tr '/+' '_-'
```

3. Use the re-encoded challenge to call the Azure API.
4. Specify ES256 as the algorithm.

   The signature in the response is encoded in UrlBase64 format.
5. Encode the signature in Base64 format. For example:

```
echo $(echo -n VGhpcyBpcyBteSBzaWduYXR1cmUu | tr '_-' '/+')==
```

Managing Security Keys in the Qumulo File System Key Store

This section explains how to manage security keys in the Qumulo file system key store by using the `qq` CLI.

In Qumulo Core 6.1.0 (and higher), you can store multiple ECDSA public keys in the Qumulo file system key store and use these keys to protect file system resources.

> **Important**
>
> - Currently, Qumulo Core supports only 256-bit ECDSA keys in `.pem` and `.der` formats. Qumulo Core doesn't support storing ECDSA keys of other lengths and formats.
> - Qumulo Core retains only the public key. We strongly recommend storing the corresponding private key safely, outside of your Qumulo cluster and according to your organization's security policy.

For information about protecting your snapshots by using a key from the Qumulo file system key store, see Locking and Unlocking Snapshots (page 140).

Adding a Public Key

This section explains how to add a public key to the Qumulo file system key store. To store a public key in the key store, you must have a pair of asymmetric keys. For more information, see Generating an ECDSA Private Key (page 153).

If You Have Access to the Private Key

> **Note**
> To use this method, you must install the Python `cryptography` library.

Run the `qq fs_security_add_key` command and specify the key name, the path to the private key file, and an optional comment. For example:

```
qq fs_security_add_key \
  --name my-key-name \
  --private-key-file /path/to-my-file.pem \
  --comment "This is an optional comment."
```

If You Don't Have Direct Access to the Private Key

Run the `qq fs_security_add_key` command and specify the key name, the public key contents, the Base64-encoded verification signature (the key name signed with the private key), and an optional comment. For example:

```
qq fs_security_add_key \
  --name my-key-name \
  --public-key "VGhpcyBpcyBteSBwdWJsaWMga2V5IGNvbnRlbnRzLg==" \
  --verification-signature "VGhpcyBpcyBteSB1bmxvY2sgY2hhbGxlbmdlLg==" \
  --comment "This is an optional comment."
```

For more information, see Extracting the Public Key from an ECDSA Private Key (page 155) and Signing a Security Challenge by Using an ECDSA Private Key (page 156).

Retrieving Public Key Information

- To retrieve information for a single public key, run the `qq fs_security_get_key` command and specify the key identifier or name.
- To retrieve information for all public keys, run the `qq fs_security_list_keys` command.

  The output displays information in a table format. To view the output in JSON format, use the `--json` flag.

Retrieving Public Key Usage Information

Run the `qq fs_security_get_key_usage` command and specify the key identifier or name.

The output displays information in a table format. To view the output in JSON format, use the `--json` flag.

Modifying a Public Key

To modify the name or comment for a public key, run the `qq fs_security_modify_key` command and specify the key identifier or name and the flags for the fields to modify.

> **Note**
> The response of the `qq fs_security_add_key` command includes the *key identifier.* When you use `qq fs_security` commands, you can specify either the key identifier (by using the `--id` flag) or the key name (by using the `--name` flag).

Rotating a Public Key

This section explains how to rotate a public key in the Qumulo file system key store.

> **Note**
>
> - Key rotation doesn't affect the resources that the key protects or change the identifier or name of the public key.
> - When the key rotation is complete, only the replacement public key can unlock the protected resources.
> - The response of the `qq fs_security_add_key` command includes the *key identifier.* When you use `qq fs_security` commands, you can specify either the key identifier (by using the `--id` flag) or the key name (by using the `--name` flag).

If You Have Access to the Existing and Replacement Private Keys

> **Note**
> To use this method, you must install the Python `cryptography` library.

Run the `qq fs_security_replace_key` command and specify the key identifier or name, the path to the existing private key, and the path to the replacement private key. For example:

```
qq fs_security_replace_key \
  --name my-key-name \
  --old-private-key-file /path/to-existing-key.pem \
  --replacement-private-key-file /path/to-replacement-key.pem
```

If You Don’t Have Direct Access to the Existing and Replacement Private Keys

1. To receive the key replacement challenge, run the `qq fs_security_get_key_replace_challenge` command and specify the identifier or name of the key to replace.
2. To generate a verification signature, use the response from the challenge with the existing private key and another verification signature by using the challenge and the replacement private key.

   For more information, see Signing a Security Challenge by Using an ECDSA Private Key (page 156).

3. To rotate the key, run the `qq fs_security_replace_key` command and specify the key identifier or name, the replacement public key contents, the replacement key verification signature (Base64-encoded key replacement challenge signed with the replacement private key), and the existing key verification signature (Base64-encoded key replacement challenge signed with the existing private key). For example:

```
qq fs_security_replace_key \
  --name my-key-name \
  --replacement-public-key "VGhpcyBpcyBteSByZXBsYWNlbWVudCBwdWJsaWMga2V5Lg==" \
  --replacement-key-verification-signature "UmVwbGFjZW1lbnQga2V5IHZlcmlmaWNhdGlvbiBzaWduYXR1cmU=" \
  --old-key-verification-signature "RXhpc3Rpbmcga2V5IHZlcmlmaWNhdGlvbiBzaWduYXR1cmU="
```

   For more information, see Extracting the Public Key from an ECDSA Private Key (page 155).

   > **Note**
   > Because the key version is part of the challenge message, and this version changes when a user writes or modifies the key, any change to the key name or comment after you receive the challenge message makes the message stale.

Disabling a Public Key

When you add a key to the Qumulo file system key store, Qumulo Core enables it automatically.

- To disable a key, run the `qq fs_security_modify_key` command and specify the key identifier or name and the `--disable` flag.
- To re-enable a key, use the `--enable` flag.

Note

- It isn't possible to lock a Qumulo file system resource with a disabled key. However, you can still unlock resources that this key locked previously.
- It isn't possible to disable a key that Qumulo Core uses to create new resources. For example, you can't disable a key associated with a snapshot policy when the snapshot policy creates new snapshots by using the key. In this scenario, you must disassociate the key from the snapshot policy before you can disable it. For more information, see Retrieving Public Key Usage Information (page 160).
- The response of the `qq fs_security_add_key` command includes the *key identifier.* When you use `qq fs_security` commands, you can specify either the key identifier (by using the `--id` flag) or the key name (by using the `--name` flag).

Deleting a Public Key

Run the `qq fs_security_delete_key` command and specify the key identifier or name.

Note

- It isn't possible to delete a key that a Qumulo file system resource uses. For more information, see Retrieving Public Key Usage Information (page 160).
- The response of the `qq fs_security_add_key` command includes the *key identifier.* When you use `qq fs_security` commands, you can specify either the key identifier (by using the `--id` flag) or the key name (by using the `--name` flag).

Installing a Signed SSL Certificate on a Qumulo Cluster

This section explains how to install a signed SSL certificate from your certificate authority (CA) on your Qumulo cluster.

Requirements

- An SSL certificate based on your certificate signing request (CSR) file from your certificate authority (CA)
- A CA-bundle PEM chain in the following order:
    - Your certificate
    - The intermediate CA
    - The root CA

To Install a Signed SSL Certificate on a Qumulo Cluster from the Command Line

1. Verify that your certificate and the CA-bundle are in the PEM format by running the `file *` command.

   The following is example output.

   ```
   certbundle.pem: PEM certificate
   private.key: PEM RSA private key
   ```

2. (Optional) If your file isn't an RSA key, run the `openssl rsa` command to convert your key. For example:

   ```
   openssl rsa \
     -in original.key \
     -out private.key
   ```

3. Run the `qq login` command to log in to your Qumulo cluster as an administrator. For example:

   ```
   qq login \
     -u admin \
     --host 203.0.113.0
   ```

4. Run the `qq ssl_modify_certificate` command to install your certificate. For example:

```
qq ssl_modify_certificate \
  --host 203.0.113.0 \
  -c certbundle.pem \
  -k private.key.insecure
```

To Import a Certificate Authority (CA) Certificate on macOS

1. Press ⌘ + Space, enter `Keychain Access`, and press Enter.
2. When prompted, click Open Keychain Access.
3. In the Keychain Access window, on the left panel, under Default Keychains, click login.
4. On the right, click Certificates.
5. Copy your CA certificate file to the list of certificates.
6. Right-click your certificate and then click Get Info.
7. On the window with the certificate information, expand the Trust section and When using this certificate: select Always Trust.

> **Note**
> To ensure your certificate is installed correctly, restart your browser.

Node Addition and Replacement

How Drive and Node Failure Protection Works in Qumulo Core

This section provides an overview of how Qumulo clusters ensure continued operation in the event of a drive or node failure.

How Qumulo Core Ensures Fault Tolerance

Qumulo Core protects your cluster with a `6,4` erasure coding (2 concurrent drive failures or 1 node failure), at minimum. When a drive fails, Qumulo Core begins to rebuild the data that was previously stored on the failed drive.

> **Note**
> When Qumulo Core finishes reprotecting the drive, it resets the fault tolerance for the cluster, regardless of whether you have replaced the failed drive.

As a cluster increases in size, Qumulo Core makes additional fault tolerance options available during the cluster creation process. After creating a cluster, you can use Adaptive Data Protection (page 175) to include the cluster's fault tolerance during node-add procedures.

> **Note**
> Depending on a cluster's size constraints, certain configurations (such as 1 concurrent drive failure or 4 node failures) might not be possible.

To view the fault tolerance of your Qumulo cluster:

- **Web UI:** Navigate to the **Cluster Overview** page
- `qq` **CLI:** Run the `qq protection_status_get` command
- **REST API:** Call the `/v1/cluster/protection/status` endpoint

Read-Only Mode Scenario for Hybrid Nodes

When a hybrid node goes offline for a substantial period of time, there is a risk of the cluster entering read-only mode because Qumulo Core writes all inbound operations only to the node's SSDs.

The length of time before this scenario takes place depends on the number of drives in a node and the rate of incoming writes, deletes, and changes. For more information, see Understanding Offline Nodes and Checking for Free Space (page 185). If you encounter this scenario, contact the Qumulo Care team.

The following sections describe various drive and node failure protection configurations and how they correspond to failure scenarios and data protection states.

2-Drive, 1-Node Protection (2,1)

This is the default system configuration. This configuration requires a minimum of 4 nodes.

Failure Scenario	Severity	Data Protection State
1 drive failure	⬇️ Low	The data is protected. You can replace a failed drive at any time.
2 drive failures	⚠️ Medium	The data is protected. You can replace a failed drive at any time.
1 node failure	🚩 High	The data is protected. The cluster is at risk of going into read-only mode.
3 (or more) drive failures or multiple node failures	🚩 High	The data is unavailable but intact.

3-Drive, 1-Node Protection (3,1)

This configuration requires a minimum of 5 nodes.

Failure Scenario	Severity	Data Protection State
1 drive failure	⬇️ Low	The data is protected. You can replace a failed drive at any time.
2 drive failures	⚠️ Medium	The data is protected. You can replace a failed drive at any time.
3 drive failures	⚠️ Medium	The data is protected. You can replace a failed drive at any time.
1 node failure	🚩 High	The data is protected. The cluster is at risk of going into read-only mode.
4 (or more) drive failures or multiple node failures	🚩 High	The data is unavailable but intact.

3-Drive, 2-Node Protection (3,2)

This configuration requires a minimum of 11 nodes.

Failure Scenario	Severity	Data Protection State
1 drive failure	⬇️ Low	The data is protected. You can replace a failed drive at any time.
2 drive failure	⬇️ Low	The data is protected. You can replace a failed drive at any time.
3 drive failures	⚠️ Medium	The data is protected. You can replace a failed drive at any time.
1 node failure	🚩 High	The data is protected. The cluster is at risk of going into read-only mode.
2 node failures	🚩 High	The data is protected. The cluster is at risk of going into read-only mode.
4 (or more) drive failures or more than 2 node failures	🚩 High	The data is unavailable but intact.

3-Drive, 3-Node Protection (3,3)

This configuration requires a minimum of 11 nodes.

Failure Scenario	Severity	Data Protection State
1 drive failure	⬇️ Low	The data is protected. You can replace a failed drive at any time.
2 drive failures	⬇️ Low	The data is protected. You can replace a failed drive at any time.
3 drive failures	⚠️ Medium	The data is protected. You can replace a failed drive at any time.
1 node failure	🚩 High	The data is protected. The cluster is at risk of going into read-only mode.
2 node failures	🚩 High	The data is protected. The cluster is at risk of going into read-only mode.
3 node failures	🚩 High	The data is protected. The cluster is at risk of going into read-only mode.

Failure Scenario	Severity	Data Protection State
4 (or more) drive failure or more than 3 node failures	🚩 High	The data is unavailable but intact.

4-Drive, 2-Node Protection (4,2)

This configuration requires a minimum of 12 nodes.

Failure Scenario	Severity	Data Protection State
1 drive failure	⬇️ Low	The data is protected. You can replace a failed drive at any time.
2 drive failures	⬇️ Low	Data is protected. You may replace a failed drive at any time.
3 drive failures	⚠️ Medium	Data is protected. You may replace a failed drive at any time.
4 drive failures	⚠️ Medium	Data is protected. You may replace a failed drive at any time.
1 node failure	🚩 High	The data is protected. The cluster is at risk of going into read-only mode.
2 node failures	🚩 High	The data is protected. The cluster is at risk of going into read-only mode.
5 (or more) drive failures or more than 2 node failures	🚩 High	The data is unavailable but intact.

4-Drive, 3-Node Protection (4,3)

This configuration requires a minimum of 24 nodes.

Failure Scenario	Severity	Data Protection State
1 drive failure	⬇️ Low	The data is protected. You can replace a failed drive at any time.
2 drive failures	⬇️ Low	The data is protected. You can replace a failed drive at any time.

Failure Scenario	Severity	Data Protection State
3 drive failures	⚠️ Medium	The data is protected. You can replace a failed drive at any time.
4 drive failures	⚠️ Medium	The data is protected. You can replace a failed drive at any time.
1 node failure	🚩 High	The data is protected. The cluster is at risk of going into read-only mode.
2 node failures	🚩 High	The data is protected. The cluster is at risk of going into read-only mode.
3 node failures	🚩 High	The data is protected. The cluster is at risk of going into read-only mode.
5 (or more) drive failure or more than 3 node failure	🚩 High	The data is unavailable but intact.

4-Drive, 4-Node Protection (4,4)

This configuration requires a minimum of 24 nodes.

Failure Scenario	Severity	Data Protection State
1 drive failure	⬇️ Low	The data is protected. You can replace a failed drive at any time.
2 drive failures	⬇️ Low	The data is protected. You can replace a failed drive at any time.
3 drive failures	⚠️ Medium	The data is protected. You can replace a failed drive at any time.
4 drive failures	⚠️ Medium	The data is protected. You can replace a failed drive at any time.
1 node failure	🚩 High	The data is protected. The cluster is at risk of going into read-only mode.
2 node failures	🚩 High	The data is protected. The cluster is at risk of going into read-only mode.

Failure Scenario	Severity	Data Protection State
3 node failures	🚩 High	The data is protected. The cluster is at risk of going into read-only mode.
4 node failures	🚩 High	The data is protected. The cluster is at risk of going into read-only mode.
5 (or more) drive failures or more than 4 node failures	🚩 High	The data is unavailable but intact.

Adding Nodes to an Existing Qumulo Cluster

This section explains how to add new HPE, Supermicro, or Quiver nodes to an existing cluster.

After you connect and power on your new nodes, Qumulo Core discovers any unconfigured nodes automatically and prompts you to add nodes in the Qumulo Core Web UI.

If Qumulo Core doesn't discover any unconfigured nodes, it displays the message **No unconfigured nodes found**. If you expect to see nodes, contact the Qumulo Care team.

> **Note**
>
> - Qumulo Core requires a short time to update the total available storage.
> - Existing nodes retain their numbering.

Prerequisites

- **Sufficient Static IP Addresses:** The number of static IP addresses must be equal to or greater than the number of nodes in your cluster. For more information, see IP Failover with Qumulo Core on Qumulo Care.
- **Same Qumulo Core Version on All Nodes:** For information about upgrading Qumulo Core, see Performing a Clean Installation of Qumulo Core and Performing Qumulo Core Upgrades.

Step 1: Resolve Drive Compatibility Issues

> **Important**
> If the version of Qumulo Core on your existing nodes predates the Qumulo-certified drives that you received with your new nodes, you can't install a lower version of Qumulo Core on your new node and Qumulo Core displays the message **Installation failed**. Use the cluster logs to identify any incompatible drives.

To receive support for new, Qumulo-certified drives, do one of the following:

- Upgrade your existing cluster to the latest version of Qumulo Core and then install the same version of Qumulo Core on your new node. For more information, see Upgrading Your Qumulo Cluster (page 44).
- Update the Supported Drive List on your new node. For more information, contact the Qumulo Care team.

Step 2: Add Your New Nodes to an Existing Qumulo Cluster

1. Log in to the Qumulo Core Web UI.
2. Click **Cluster** > **Add Nodes**.
3. On the **Add Nodes** page, select unconfigured nodes to add to your cluster.
4. Click **Add Selected Nodes to Cluster**.
5. In the **Add <N> nodes to cluster <name>?** dialog box, click **Yes**.

   If you add one or more node model types, a message reminds you about Qumulo Core adding a new model type to your cluster.

Qumulo Core configures your new nodes and adds them to your cluster.

On the **Cluster** page, the Qumulo Core Web UI shows the banner **Successfully added <N> nodes to the cluster** and the total available storage.

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

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

- In Qumulo Core 5.1.3 (and higher) you can increase the node fault tolerance level for an existing cluster during the cluster expansion process.
- In Qumulo Core 6.1.0.3 (and higher), you can manage the drive and node fault tolerance levels during the cluster expansion process by changing the data protection configuration (page 175).

> **Important**
> We strongly recommend contacting the Qumulo Care team before proceeding with cluster expansion.

To Reconfigure Your Cluster's Node Fault Tolerance Level

1. Follow the instructions in Adding Nodes to an Existing Qumulo Cluster (page 172).

   > **Note**
   >
   > When a trade-off between increasing node fault tolerance and maximizing usable capacity is available for your cluster, the **Cluster > Add Nodes** page shows the **Accept a trade-off in the increase of usable capacity** checkbox.
   >
   > When a trade-off isn't available, the page shows an message which indicates that it is possible to increase the node fault tolerance level.

2. Before you click **Yes** in the **Add <N> nodes to cluster <MyCluster>?** dialog box, check that the projected capacity matches the expected capacity.
3. After the cluster expansion process finishes, Qumulo Core begins data protection reconfiguration automatically.
4. 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.
5. When the restriper completes the data protection reconfiguration, the **Data Protected** section of the **Cluster** page shows the increased node fault tolerance level.

Adding Nodes with Adaptive Data Protection to a Qumulo Cluster

This section explains how to configure Adaptive Data Protection for your Qumulo cluster during node-add operations.

- Qumulo Core 6.1.0.1 and 6.1.1 support Adaptive Data Protection by letting you reconfigure your cluster's fault tolerance level and storage efficiency only when you add nodes to your cluster.

  Depending on your cluster, Qumulo Core shows configuration options that offer better fault tolerance levels, better storage efficiency, or both benefits. To enable Adaptive Data Protection for your cluster, you must contact the Qumulo Care team.

- In Qumulo Core 6.1.2 (and higher), you can change your cluster's data protection configuration when you add or replace nodes by using the `qq` CLI.

> **Note**
> Your cluster's usable capacity doesn't increase until the data protection reconfiguration is complete. Because data protection reconfiguration is a long process (that can take days or weeks), we don't recommend beginning this process when your cluster is nearly running out of usable capacity. In this scenario, contact the Qumulo Care team.

To Add a Node with Adaptive Data Protection to a Qumulo Cluster

A cluster's data protection configuration includes the stripe configuration (for example, `10.8`) and the node fault tolerance level.

> **Important**
> Ensure that your configuration and node order are correct. Unless you add more nodes, it isn't possible to revert this operation or reorder nodes after adding them to a cluster.

1. To select the new data protection configuration for your nodes, contact the Qumulo Care team.
2. Rack and wire your new nodes and then power them on.
3. To determine the UUIDs of the nodes to add to your cluster, run the `qq unconfigured_nodes_list` command.
4. Write down the UUIDs of the nodes that you want to add to the cluster, in the order that you want to add them.

5. To add your nodes to the cluster, run the `qq add_nodes` command and specify the node UUIDs (their ordering in the command determines their ordering in the cluster), the configuration type, and the node fault tolerance level.

   In the following example, we change the stripe configuration to 10.8 and the node fault tolerance level to 2 nodes.

   ```
   qq add_nodes \
     --node-uuids 12345a6b-7c89-0d12-3456-78fe9012f345 \
       abcde1f2-g3hi-j4kl-mnop-qr56stuv7wxy \
     --target-stripe-config 10 8 \
     --target-max-node-failures 2
   ```

   The following is example output.

   ```
   Current cluster:
       Usable capacity: 600 TB
       Node fault tolerance level: 1 node
   With the selected node-add operation and data protection reconfiguration:
       Usable capacity: 800 TB
       Node fault tolerance level: 2 nodes
   ```

6. To confirm the configuration with the selected node-add and data protection configuration operations, enter `yes`.

Monitoring the Data Protection Reconfiguration Process

To view the progress of the three stages of the data protection reconfiguration process, log in to the Qumulo Core Web UI and click **Cluster**.

1. Qumulo Core begins to move data to new nodes in the cluster and the Qumulo Core Web UI displays the message **Rebalancing for data protection reconfiguration**.
2. Qumulo Core reencodes all data on your cluster and the Qumulo Core Web UI displays the message **Reconfiguring data protection**.

   > **Note**
   > In certain scenarios, this stage might appear to pause while the system performs preparatory work on the cluster.

   When this stage is complete, your data is protected according to the cluster's new configuration and the system begins to use the new drive and node fault tolerance levels.

3. Qumulo Core adds new capacity to your cluster and the Qumulo Core Web UI displays the message **Rebalancing**.

   If you initiated the reconfiguration process as part of a node replacement step, the system migrates data from the existing nodes in the cluster.

Cluster Availability During the Reconfiguration Process

Your cluster remains available throughout the data protection reconfiguration process.

- You can upgrade Qumulo Core.
- Your cluster maintains the ability to recover from node and drive failure automatically.

  During the reconfiguration process, drive and node fault tolerance levels remain at the minimums that the existing and new configurations specify. For example, if your existing cluster has 2-node and 2-drive fault tolerance, and you initiate reconfiguration where the new configuration has 1-node and 3-drive fault tolerance, your cluster has 1-node and 2-drive fault tolerance during the reconfiguration process.

> **Note**
>
> - To avoid impact to front-end workloads, Qumulo Core slows down the reconfiguration process automatically.
> - When Qumulo Core finds missing nodes or drives, it pauses the reconfiguration process. When you replace or bring the nodes or drives online, the reconfiguration process continues.
> - It isn't possible to add or replace nodes during the reconfiguration process.

Replacing Nodes in a Qumulo Cluster by Performing a Transparent Platform Refresh

This section explains how to replace nodes that have reached retirement or end of life by performing a two-stage transparent platform refresh on clusters that run Qumulo Core 6.1.0.3 (and higher).

> **Note**
>
> - Qumulo Core doesn't support replacing nodes in clusters with more than 100 nodes.
> - In Qumulo Core 6.1.2.2 (and higher), you can use the `qq` CLI to replace nodes. To replace nodes on a lower version of Qumulo Core, contact the Qumulo Care team.

How Transparent Platform Refresh Works

Transparent platform refresh comprises two stages. For help with your node replacement plan, contact the Qumulo Care team.

Stage 1: Register a Node Replacement Plan

In this stage, you register a node replacement plan (page 179) with your cluster. The plan includes information about the nodes to replace and the data protection configuration.

In following example, we use a four-node cluster and:

- Replace nodes 1-4 with five new nodes in a single step
- Change the data protection configuration to the `8.6` stripe configuration with 1-node fault tolerance

Stage 2: Execute the Node Replacement Plan Steps

In this stage, you execute the node replacement plan's steps (page 180) and Qumulo Core performs data protection reconfiguration.

> **Note**
> It isn't possible to add nodes or begin another node replacement step while a node replacement step is already in progress.

There are two node replacement plan types:

- **Single-Step Node Replacement:** Qumulo Core adds all new nodes and removes all nodes marked for replacement in a single step. Use this approach when the node replacement speed is a priority.

- **Multi-Step Node Replacement:** Each step of the plan adds some the new nodes and removes some nodes marked for replacement. Use this approach when rack space or switch port capacity in your data center is limited.

Cluster Properties During Node Replacement

- When a replacement step begins, Qumulo Core distributes floating IP addresses among the nodes in the combined cluster (page 179). After Qumulo Core removes nodes marked for replacement, it redistributes any client connections that use floating IP addresses among the nodes that remain in the cluster.
- While a node replacement step is in progress, both new nodes and nodes marked for replacement appear on the **Cluster** page of the Qumulo Core Web UI and clients can connect to any of the nodes in the *combined cluster* while the step is in progress.
- When a node replacement step is complete, the reassignment of static IP addresses differs between versions of Qumulo Core:
    - In Qumulo Core 6.3.0.1 (and higher), the static IP addresses assigned to nodes remain unchanged and Qumulo Core removes only the static IP addresses for nodes removed from the cluster.
    - In Qumulo Core versions lower than 6.3.0.1, Qumulo Core reassigns static IP addresses to different nodes. To view the reassigned IP addresses in the Qumulo Core Web UI, click **Cluster** > **Network Configuration**.
- When Qumulo Core adds nodes to a cluster, it assigns node IDs sequentially, without reusing or changing IDs.

  For example, if you have a four-node cluster with node IDs 1-4, and you replace node IDs 2 and 3 with two new nodes, after node replacement the cluster contains node IDs 1, 4, 5, and 6. If you add another node, it has the ID 7.
- A cluster's usable capacity doesn't increase until:
    - Any data protection reconfiguration is complete
    - The last step of the node replacement plan is in progress

  For example, if you replace nodes in a single step without data protection reconfiguration, usable capacity increases as soon as Qumulo Core begins the step.

Prerequisites

Ensure that the number of static and floating IP addresses is equal to or greater than the number of nodes in the combined cluster (page 179).

Step 1: Register a Node Replacement Plan by Using the qq CLI

1. Run the `qq replace_nodes register_plan` command and the `--nodes-to-be-replaced` flag to specify the nodes to replace and the `--target-stripe-config` flag to specify the stripe configuration. For example:

   ```
   qq replace_nodes register_plan \
     --nodes-to-be-replaced 1 2 3 4 \
     --target-stripe-config 8 6
   ```

   Qumulo Core stores the node replacement plan on your cluster.

   > **Note**
   >
   > f your plan includes data protection reconfiguration, Qumulo Core records only the tripe configuration. You specify the node fault tolerance when you execute the plan teps.
   >
   > f your plan doesn't include data protection reconfiguration, you can omit the `--target-tripe-config` flag.
   >
   > o replace all nodes in the cluster, use the `--replace-all` flag instead of the `--nodes-to-e-replaced` flag.

2. Rack and wire your new nodes and then power them on.
3. To determine the UUIDs of the nodes to add to your cluster, run the `qq unconfigured_nodes_list` command.
4. Write down the UUIDs of the nodes that you want to add to the cluster, in the order that you want to add them.

Step 2: Execute the Node Replacement Plan Steps by Using the qq CLI

1. Run the `qq replace_nodes add_nodes_and_replace` command to initiate each step, the `--nodes-being-replaced` flag to specify the nodes to replace, and the `--node-uuids` flag to specify the nodes to add during the current step.

   > **Important**
   > Qumulo Core adds nodes to the cluster in the order in which you list their UUIDs after the `--node-uuids` flag. When you begin the node replacement step, it isn't possible to revert this operation or reorder nodes after adding them to a cluster.

If your plan includes data protection reconfiguration, use the `--reconfigure-data-protection` and `--target-max-node-failures` flags to initiate the reconfiguration during the current step. For example:

```
qq replace_nodes add_nodes_and_replace \
  --nodes-being-replaced 1 2 3 4 \
  --node-uuids 12345a6b-7c89-0d12-3456-78fe9012f345 abcde1f2-g3hi-j4kl-mnop-qr
56stuv7wxy \
  --reconfigure-data-protection \
  --target-max-node-failures 1
```

The following is example output.

```
Current cluster:
    Usable capacity: 200 TB
    Node fault tolerance level: 1 node
With the selected node replacement step:
    Usable capacity: 220 TB
    Node fault tolerance level: 1 node
```

> **Note**
> To replace all nodes in the cluster, use the `--replace-all` flag instead of the `--nodes-being-replaced` flag.

2. To confirm the reconfiguration with the selected node-replace and data protection configuration operations, enter `yes`.

   For more information, see Monitoring the Data Protection Reconfiguration Process (page 182).

3. Wait for the node replacement step to complete.

   After each node replacement step, Qumulo Core begins to migrate data from existing nodes in the background.

   > **Note**
   > This is a long process (that can take days or weeks). When the data migration is complete, Qumulo Core removes the nodes marked for replacement from the cluster. These nodes no longer appear on the **Cluster** page of the Qumulo Core Web UI.

4. Unrack the removed nodes from your data center.

5. Initiate the next node replacement step.

Viewing, Editing, and Canceling the Node Replacement Plan

- To view the current node replacement plan, run the `qq replace_nodes` command with the `get_plan` subcommand.

  If a node replacement step is in progress, the command shows the list of nodes in process of being replaced during the current step.

- To edit the node replacement plan after you register it with your cluster, run the `qq replace_nodes` with the `register_plan` subcommand and a new node replacement plan (page 179).

- To cancel the current node replacement plan, run the `qq replace_nodes` command with the `cancel_plan` subcommand.

  > **Important**
  > Canceling a node replacement plan after executing one or more steps might make it impossible to reregister and complete the plan.

Monitoring the Data Protection Reconfiguration Process

To view the progress of the three stages of the data protection reconfiguration process, log in to the Qumulo Core Web UI and click **Cluster**.

1. Qumulo Core begins to move data to new nodes in the cluster and the Qumulo Core Web UI displays the message **Rebalancing for data protection reconfiguration**.
2. Qumulo Core reencodes all data on your cluster and the Qumulo Core Web UI displays the message **Reconfiguring data protection**.

   > **Note**
   > In certain scenarios, this stage might appear to pause while the system performs preparatory work on the cluster.

   When this stage is complete, your data is protected according to the cluster's new configuration and the system begins to use the new drive and node fault tolerance levels.
3. Qumulo Core adds new capacity to your cluster and the Qumulo Core Web UI displays the message **Rebalancing**.

   If you initiated the reconfiguration process as part of a node replacement step, the system migrates data from the existing nodes in the cluster.

Cluster Availability During the Reconfiguration Process

Your cluster remains available throughout the data protection reconfiguration process.

- You can upgrade Qumulo Core.
- Your cluster maintains the ability to recover from node and drive failure automatically.

  During the reconfiguration process, drive and node fault tolerance levels remain at the minimums that the existing and new configurations specify. For example, if your existing cluster has 2-node and 2-drive fault tolerance, and you initiate reconfiguration where the new configuration has 1-node and 3-drive fault tolerance, your cluster has 1-node and 2-drive fault tolerance during the reconfiguration process.

> **Note**
>
> - To avoid impact to front-end workloads, Qumulo Core slows down the reconfiguration process automatically.
> - When Qumulo Core finds missing nodes or drives, it pauses the reconfiguration process. When you replace or bring the nodes or drives online, the reconfiguration process continues.
> - It isn't possible to add or replace nodes during the reconfiguration process.

Improving Performance by Migrating a Qumulo Cluster to a Different License Class

This section explains how you can improve the performance of a cluster that runs Qumulo Core 6.1.0.3 (or higher) by migrating it a different license class (for example, from hybrid to all-NVMe nodes).

> ⚠ Important
> Because it isn't possible to have nodes with different license classes in the same cluster, you must migrate all nodes in your cluster to the same license class.

For help with migrating your cluster to a different license class, contact the Qumulo Care team.

Understanding Offline Qumulo Core Nodes

This section explains what happens when a Qumulo Core node goes offline.

What Happens When a Qumulo Core Node Goes Offline

Qumulo Core uses *erasure coding* to let multiple drives or nodes to go offline but continue to serve data. For more information, see Qumulo Drive Failure Protection on Qumulo Care.

Qumulo Core attempts to maintain cluster functionality for as long as possible. When a hybrid node (that has both HDDs and SSDs) goes offline, the cluster evicts existing data promoted to SSDs on the remaining nodes and makes all writes to the free space on the SSDs. When the node comes online, Qumulo Core begins to push writes to the HDDs that back the SSDs.

> **Important**
> Writes, deletes, and changes count towards SSD space.

A cluster can operate with an offline node until its SSD space fills up. When a cluster has no more free space, the cluster goes into read-only (`ENOSPC`) state until the node comes back online and all nodes rejoin the quorum.

> **Note**
> The amount of time that users have before the cluster enters the `ENOSPC` state depends on the rate at which data is being added to, or modified within, the cluster.

Data Replication

Creating and Managing a Continuous Replication Relationship in Qumulo Core

This section explains how to create, authorize, modify, and delete a replication relationship by using the Qumulo Core Web UI.

How Continuous Replication Works

> **Important**
> Qumulo Core supports replication between different versions only if either the source or target cluster is running Qumulo Core 2.12.0 (or higher). For more information, see Replication Version Requirements and Upgrade Recommendations (page 50).

Continuous replication takes a snapshot of the data in a directory on the *source cluster* and transfers it to a directory on the *target cluster*. While continuous replication runs, Qumulo Core scans modified files for any changed regions and transfer only these changes to the target cluster.

Continuous replication uses snapshots to generate a consistent point-in-time copy of the source directory on the target cluster. To ensure that a directory contains only the most recent snapshot, Qumulo Core deletes previous snapshots automatically. Administrators can view the snapshots used for replication and any other policy-based snapshots.

Prerequisites

The following privileges are required for continuous replication.

> **Note**
>
> - We don't recommend granting the following privileges to specific users because they grant administrative access to your cluster.
> - The following privileges grant user access to Qumulo Core functionality beyond replication relationship management:
>   - `PRIVILEGE_REPLICATION_SOURCE_WRITE`: Grants the permission to access any data on a cluster, regardless of file or directory permissions
>   - `PRIVILEGE_REPLICATION_TARGET_WRITE`: Grants the permission to authorize replication relationships to any target directory on a cluster

- Creating a replication relationship

- `SOURCE_WRITE`: For the user on the source cluster to initiate the creation of the relationship
- `TARGET_WRITE`: For the user on the target cluster to authorize the relationship

- Viewing the replication relationship status
  - `PRIVILEGE_REPLICATION_SOURCE_READ`
  - `PRIVILEGE_REPLICATION_TARGET_READ`

To Create a Replication Relationship

> **Important**
> A replication job doesn't begin until you authorize the relationship on the target cluster.

1. Log in to the Qumulo Core Web UI on the source cluster.
2. Click **Cluster** > **Replication**.
3. On the right side of the **Replication Relationships** page, click **Create Relationship**.
4. On the **Create Replication Relationship** page:
   a. For **Source Directory Path**, enter the existing directory from which to replicate data.
   b. For **Target Directory Path**, enter the existing directory to which to replicate data.
   c. For **Target Address**, enter one of the IP addresses from a node on the target cluster.

      > **Tip**
      > We recommend using a floating IP address.

   d. For **Port Number**, click **Default (3712)** or enter a custom port.

      > **Note**
      > Your organization's firewall might require a custom port.

5. Click **Add Blackout Window** and then select the days of the week and time when replication suspends.
6. (Optional) To add another blackout window, click **Add Blackout Window**.

> **Note**
> You can add up to ten blackout windows. For more information, see Replication: Blackout Windows on Qumulo Care.

7. To replicate files by using locally-owned NFS IDs, under **Map Local User/Group IDs to Associated NFS IDs**, click Enabled.

   For more information, see Replication: NFS ID Mapping on Qumulo Care.

8. Under **Enable Replication**, click **Enabled**.
9. Click **Save Relationship**.

To Authorize a Replication Relationship

> **Note**
> If your cluster is currently in a blackout window or if continuous replication for the replication relationship is disabled, replication doesn't begin. For more information, see Replication: Blackout Windows on Qumulo Care.

1. Log in to the Qumulo Core Web UI on the target cluster.

   A notification banner informing you of a new relationship authorization request appears.

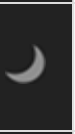
2. On the banner, click **See details**.
3. On the **Replication Relationships** page, click **Accept**.

To View the Status of a Replication Relationship

1. Log in to the Qumulo Core Web UI on the source cluster.
2. Click **Cluster** > **Replication**.

   The **Replication Relationships** page shows a list of source and target clusters. The following table explains the icons that appear between the **Source** and **Target** columns.

Icon	Description
	The replication is running or is ready to run.
	The replication is awaiting authorization from the target cluster's administrator.

	The replication relationship is disconnected and the target directory is writable.
	The replication relationship is in a blackout window.
	Continuous replication is disabled.
	The target directory is reverting to the last recovery point before becoming writable.
	The replication job is incomplete and will retry soon.

A progress bar in the **Status** column indicates the replication process.

> **Note**
> The replication process percentage considers the number of files to be replicated *and* the amount of data to be transferred.

3. To review the throughput, run time, and data statistics for the replication job in progress, click **Details**.

To Modify a Replication Relationship

> **Note**
> It isn't possible to edit the source and target directory paths. To make these changes, you must create a new replication relationship.

1. Log in to the Qumulo Core Web UI.
2. Click **Cluster** > **Replication**.
3. On the **Replication Relationships** page, next to the relationship to modify, click > **Edit Relationship...**
4. Make changes to your replication relationship (for more information, see To Create a Replication Relationship (page 187)) and then click **Save Relationship**.

To Delete a Replication Relationship

1. Log in to the Qumulo Core Web UI.
2. Click **Cluster** > **Replication**.
3. On the **Replication Relationships** page, next to the relationship to delete, click > **Delete Relationship...**
4. In the **Delete relationship** dialog box, review the source and target clusters and then click **Yes, Delete**.

Known Continuous Replication Limitations in Qumulo Core

- **Continuous Replication:** Depending on applications in use while a replication job runs, continuous replication increases the load on the cluster and can cause latency delays.
- **Local Users and Groups:** Continuous replication doesn't support replicating local user or group information and fails when it encounters a file associated with local users or groups.
- **Target Directory Permissions** When you create a replication relationship, Qumulo Core updates these permissions from read-write to read-only. When you delete the relationship, the permissions revert to read-write automatically.
- **Root Directory:** Including this directory makes the entire target cluster read-only and might prohibit administrative functions such as upgrades. For more information, see Replication Version Requirements and Upgrade Recommendations (page 50)
- **Upgrades During Replication:** Before you perform any Qumulo Core upgrades, you must wait until the replication job is complete. Upgrading Qumulo Core while a replication job is in progress causes the job to fail. When both clusters return to a state where replication is possible (the versions are the same), continuous replication can continue.

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 0).

Prerequisites

- A Qumulo cluster with:
  - Qumulo Core 3.2.1 (and higher) for the CLI and 3.2.5 (and higher) for the Qumulo Core 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:PutObjectTagging`
    - `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",
        "s3:PutObjectTagging"
      ]
      "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.

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 for each 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.

How Entities in the Qumulo File System are Represented in an S3 Bucket

This section explains which entity types Qumulo Core doesn't copy to an S3 bucket and how an S3 bucket represents the entities that Qumulo Core copies to an S3 bucket.

Entity Types that Qumulo Core Doesn't Copy

- Access control list (ACL)
- Alternate data stream
- Directory

  > **Note**
  > For objects created for files, the system preserves the directory structure in the object key.

- Hard link to a non-regular file
- SMB extended file attribute
- Symbolic link
- Timestamp (`mtime`, `ctime`, `atime`, `btime`)
- UNIX device file

Entity Types that Qumulo Core Copies

Entity in the Qumulo File System	Representation in an Amazon S3 Bucket
Hard link to a regular file	Copy of the S3 object
Generic user metadata	S3 tags

Entity in the Qumulo File System	Representation in an Amazon S3 Bucket
Hole in sparse files	Zero **Note** The system expands any holes.
Regular file	S3 object **Note** The object key is the file system path and the object value is the metadata.
S3 Metadata	Object metadata

Using the Qumulo Core Web UI to Copy Files and Manage Relationships

This section describes how to use the Qumulo Core 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 the Qumulo Core Web UI.
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 the Qumulo Core Web UI.
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 the Qumulo Core Web UI.
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 the Qumulo Core Web UI.
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 the Qumulo Core Web UI.
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, run the `qq 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, run the `qq replication_list_object_relationships` command.
- To view configuration details for a specific relationship, run the `qq 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-a4f5f6ceff78
  ```

- To view the status of a specific relationship, run the `qq replication_get_object_relationship_status` command followed by the `--id` and the Shift relationship ID.
- To view the status of all relationships, run the `qq 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, run the `qq replication_abort_object_replication` command and use the `--id` flag to specify the Shift relationship ID.

Repeating a Completed Copy Job

To repeat a completed copy job, run the `qq replication_start_object_relationship` command and use the `--id` flag to specify 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 and use the `--id` flag to specify 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 for Shift-to-S3

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).
- **User Metadata Limits:** Amazon S3's limits on object metadata (up to 2 kB across key bytes and value bytes) and tagging (10 entries with a key size of 128 bytes and a value size of 256 bytes) are more restrictive than those of Qumulo Core. When a metadata entry exceeds one of these limits, Qumulo Core omits the entry from a replication job. For more information, see User Defined Object Metadata and Categorizing your storage using tags in the Amazon Simple Storage Service User Guide.

Shift-to-S3 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 default 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 0) 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:GetObjectTagging`
  - `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 0).)

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": "s3:ListBucket",
      "Effect": "Allow",
      "Resource": "arn:aws:s3:::my-bucket"
    },
    {
      "Action": [
        "s3:GetObject",
        "s3:GetObjectTagging"
      ],
      "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 cluster 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 for each 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 Core Web UI to Copy Files and Manage Relationships

This section describes how to use the Qumulo Core 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 the Qumulo Core Web UI.
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 Qumulo Core 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 for every 5,000 objects (or 200 times for every 1 million objects).

To View Configuration Details and Status of Shift Relationships

1. Log in to the Qumulo Core Web UI.
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 the Qumulo Core Web UI.
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 the Qumulo Core Web UI.
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 the Qumulo Core Web UI.
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 to Amazon S3

To copy files, run the `qq 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, run the `qq replication_list_object_relationships` command.
- To view configuration details for a specific relationship, run the `qq 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, run the `qq replication_get_object_relationship_status` command followed by the `--id` and the Shift relationship ID.
- To view the status of all relationships, run the `qq 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, run the `qq replication_abort_object_replication` command and use the `--id` flag to specify the Shift relationship ID.

Repeating a Completed Copy Job

To repeat a completed copy job, run the `qq replication_start_object_relationship` command and use the `--id` flag to specify 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 and use the `--id` flag to specify 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 for Shift-from-S3

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

- **Inheritable Permissions:** Because the system user creates the files that Shift-From for S3 copies, the system owns these files. By default, everyone is granted read permissions and administrators always have full access to the files.

  > **Note**
  > To ensure that the copied files and subdirectories have the correct permissions, you must assign the necessary inheritable permissions to the root directory of the relationship *before* you create a Shift-From S3 relationship. To edit directory permissions, you can use the Windows Security Dialog or the `qq fs_modify_acl` command. For more information, see Qumulo File Permissions Overview on Qumulo Care.

- **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).

Shift-from-S3 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.

File System Changes

How File System Change Notifications Work in Qumulo Core

This section describes how file system change notifications work in Qumulo Core and explains request filtering, recursion, and the three configuration modes for notification requests.

Qumulo Core can stream file system change notifications to a client whenever someone modifies a file or directory. The client can specify for which directories in the file system to receive notifications and what notification types Qumulo Core sends for these directories.

Qumulo Core supports two protocols for streaming file system change notifications. Both protocols provide roughly the same functionality.

- SMB: For more information, see Watching for File Attribute and Directory Changes by Using SMB2 CHANGE_NOTIFY (page 219)
- REST: For more information, see Watching for File Attribute and Directory Changes by Using REST (page 223).

The Qumulo Core notification system guarantees that:

- The system never misses a notification.
- The system sends notifications in real time.
- A client connected to any node in a cluster receives a notification which might originate from any node in the cluster.
- The system sends notifications in accurate chronological order. (For example, the system doesn't send a `child_file_added` event after a `child_file_removed` event.)
- In case a client can't keep up with the amount of events that the system emits, the cluster stops queuing events and produces an error the next time the client attempts to contact the cluster.

How Request Filtering Works

A client can request Qumulo Core to filter notifications. Although available filters differ between protocols, they work in a similar way.

> **Tip**
> We recommend using a filter whenever you have an idea of the kind of events for which you want to receive notifications. Filtering notifications reduces back-end and front-end load and helps your client keep up with the data the cluster streams to it.

For example, you want to configure a client to receive notifications only about files being created but not deleted. In this scenario, you can make a notification request with the `child_file_added` filter for your protocol.

How Recursion Works

> ⚠ Important
>
> - Because recursive notification lets you monitor large regions of the file system tree without having to "walk" through these regions to look for changes, recursive notification is a powerful feature. Use this feature carefully: Watching too large a file tree can lead to the system sending too many notifications.
> - In case a client can't keep up with the amount of events that the system emits, the cluster stops queuing events and produces an error the next time the client attempts to contact the cluster.
> - When Qumulo Core sends too many messages, there can also be a slight performance impact for your cluster. For example, thousands of recursive watches at the file system root can have a measurable performance impact on a write-heavy IOPS workload.
> - In the Windows implementation, recursive notifications over SMB require permissions only for the directory that the system watches. The system doesn't check permissions anywhere below the directory. Before you enable recursive notification, consider whether this policy is appropriate for your organization.

When you don’t use recursion, the system sends notifications for changes that occur immediately within a target directory (changes to files that are children of the watched directory).

Consider the following example with recursion disabled:

1. You watch the `/a/b` directory.
2. You create the `/a/b/f` file.
3. You receive a notification that a user created the `/a/b/f` file.
4. You create the `/a/b/c/f` file.

In this example, you receive no additional notifications.

When you use recursion, the system sends notifications for changes that occur in the sub-tree *below* the target directory. Both protocols let you use recursion.

If you repeat the previous example scenario with recursion enabled, you receive two notifications:

- You receive a notification that a user created the `/a/b/f` file.

- You receive a notification that a user created the `/a/b/c/f` file.

Configuration Modes for Notification Requests

Qumulo Core has three global configuration modes that affect all recursive notification requests for both protocols.

- `DISABLED_ERROR`: Recursive change notification requests return errors immediately.
- `DISABLED_IGNORE`: The system accepts recursive change notification requests but sends notifications only for the top directory that it watches. (The system behaves as if the user doesn't specify the recursive flag.)

  Use this mode to improve compatibility with applications that request recursive behavior but don't depend on it.

  > **Important**
  > For scenarios that require recursive behavior, this mode can cause an application to become unresponsive or exhibit other unexpected behavior.

- `ENABLED`: This is the default mode. This mode provides full support for recursive change notification requests. The system pushes notifications for all descendants of the watched directory to the watcher.

  > **Important**
  > This mode can affect system performance. For example, thousands of recursive watches at the file system root can have a measurable performance impact on a write-heavy IOPS workload.

To select the configuration mode, use the `/v1/file-system/settings/notify` REST API or the `qq fs_set_notify_settings` command.

Supported Functionality

Functionality	Supported from Qumulo Core Version
Fully featured REST API for streaming file system notifications	6.0.2

Functionality	Supported from Qumulo Core Version
• Full support for SMB `CHANGE_NOTIFY` • Parity with Windows Server • Full support for recursion • No limit on maximum directory size • Configuration options for controlling recursive behavior	6.0.1
• Improved compatibility with Windows applications • Configuration options for controlling behavior of unsupported features	5.3.3
• Support for all possible SMB `CHANGE_NOTIFY` types except for Alternate Data Streams (ADS)	5.3.1
SMB2 `CHANGE_NOTIFY` support for adding and removing files and directories	5.3.0

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.

> **Note**
> Qumulo provides REST access to the same change notification system (page 215) that backs SMB2 `CHANGE_NOTIFY`. Because the notification interface and types are easier to work with compared to SMB2, we recommend programming by using REST rather than SMB `CHANGE_NOTIFY`.

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 file system—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.

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 the system requests only the `FILE_NOTIFY_CHANGE_EA` filter, no events propagate.

Completion Filters and Corresponding Actions

The following table shows the requested completion filters (grouped by the number of inode reads required to support them), the changes they watch for, and the actions that correspond 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 `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 the system requests only the `FILE_NOTIFY_CHANGE_EA` filter, 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.

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`
	• `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]`

> **Note**
> If you don't supply a filter, the SMB server accepts the request but doesn't send any notifications.

Re-Enumerating Changes after the STATUS_ENUM_DIR Error

If the SMB client can't keep up with the notification stream from the server, the server returns the `STATUS_ENUM_DIR` error code to the client and stops sending notifications.

In this scenario, the client must re-enumerate any changes that concern it directly, by opening and inspecting files, rather than relying on notifications. This scenario can happen because the request is too broad, for example, a recursive watch on the file system root.

Configuring Full Recursion for the WATCH_TREE Flag

By default, when a client supplies the `WATCH_TREE` flag, the server sends an error to the client immediately. In this scenario, you can configure your Qumulo cluster to support full recursion (page 215).

Watching for File Attribute and Directory Changes by Using REST

This section describes how to configure Qumulo Core and watch for file attribute and directory changes by using REST.

> **Note**
> Qumulo provides REST access to the same change notification system (page 215) that backs SMB2 `CHANGE_NOTIFY`. Because the notification interface and types are easier to work with compared to SMB2, we recommend programming by using REST rather than SMB `CHANGE_NOTIFY`.

Qumulo Core streams notifications to the client by using HTML server-sent events (SSE). For more information about the SSE syntax, see Server-sent events in the HTML Living Standard documentation.

- The comment syntax—any line that begins with a colon (`:`)— shows that the call has registered successfully for notifications and periodic keep-alive connections.
- The data syntax (`data: payload`) shows the event content.

Qumulo Core continues to stream events until the client closes the connection.

> **Important**
>
> - Standard file system permissions apply to API requests for non-recursive watching: The system compares the authenticated user that makes the API request with the defined access control list (ACL) permissions for the file and grants or denies access access. The authenticated user must have permission to read a directory in order to request notifications for its changes.
> - Because of the complexity of representing and enforcing a permissions model for an arbitrary subtree of the file system, recursive notification requests require an authenticated API user to have the `DATA_ADMINISTRATOR` privilege. This requirement remains true even if you configure your Qumulo cluster to ignore the recursive notification mode.

How SSE Event Payloads are Structured for Recursive Notification Requests

An SSE event payload is a JSON-encoded list of notification objects. The following is a format example of the SSE event payload.

```
[
  {
    "type": "<type>",
    "path": "<path>",
    "spine": ["<file_id_1>", "<file_id_2>", ...],
    "stream_name": "<optional_stream_name>"
  },
  {
  ...
  }
]
```

- `type`: One of the possible notification types (page 224).
- `path`: The path to the file for which the notification occurred.

  This path is relative to the watched directory.
- `spine`: A representation of the file path that uses Qumulo file IDs (rather than path components).
  - The first file ID in the spine is the oldest ancestor in the path.
  - The last file ID in the spine is the file for which the system sends the notification.
- `stream_name`: The name of an alternate data stream (ADS) for the file.

  When this value is `null`, the notification is for the file's default stream. Otherwise, the notification is for the listed stream.

SSE Payload Notification Types

The following is a list of available notification types in SSE event payloads (page 223).

- The `type` field shows a single notification type.
- The `filter` field shows multiple notification types in comma-separated format.

Notification Type	Description
`child_acl_changed`	The ACL for the listed or directory has been modified.

Notification Type	Description
`child_atime_changed`	The `atime` (access time) of the listed file or directory has been modified. **Note** • When a client modifies the `atime` field for a file directly, Qumulo Cores sends `atime` notifications for the file. • If you have enabled `atime` monitoring on your Qumulo Cluster, Qumulo Core sends `atime` notifications automatically. To configure `atime` monitoring, use the `/v1/file-system/settings/atime` REST API or run the `qq fs_set_atime_settings` command.
`child_btime_changed`	The `btime` (creation time) of the listed file or directory has been modified.
`child_mtime_changed`	The `mtime` (modification time) of the listed file or directory has been modified.
`child_data_written`	Data has been written to the listed file.
`child_dir_added`	The listed directory has been created.
`child_dir_removed`	The listed directory has been removed.
`child_dir_moved_from`	A directory has been moved from the listed location. **Note** The combination of the `*_moved_to` and `*_moved_from` notification type constitutes the renaming of the listed directory.

Notification Type	Description
`child_dir_moved_to`	A directory has been moved to the listed location. **Note** The combination of the `*_moved_to` and `*_moved_from` notification type constitutes the renaming of the listed directory.
`child_extra_attrs_changed`	The additional attributes for the listed file or directory have been modified. **Note** The additional attributes are Windows-specific *extra file attributes* which include `HIDDEN`, `READ_ONLY`, and so on. For more information, see File Attributes in the Microsoft Open Attributes documentation.
`child_file_added`	The listed file has been created.
`child_file_removed`	The listed file has been removed.
`child_file_moved_from`	A file has been moved from the listed location. **Note** The combination of the `*_moved_to` and `*_moved_from` notification type constitutes the renaming of the listed file.
`child_file_moved_to`	A file has been moved from the listed location. **Note** The combination of the `*_moved_to` and `*_moved_from` notification type constitutes the renaming of the listed file.
`child_group_changed`	The group for the listed file or directory has been changed.

Notification Type	Description
`child_owner_changed`	The owner for the listed file or directory has been changed.
`child_size_changed`	The size of the listed file has been changed.
`child_stream_added`	The listed alternate data stream (ADS) has been added to the listed file or directory.
`child_stream_data_written`	Data has been written to the listed ADS.
`child_stream_moved_from`	The listed ADS has been moved to the listed file or directory.
`child_stream_moved_to`	The listed ADS has been moved from the listed file or directory.
`child_stream_removed`	The listed ADS has been removed from the listed file or directory.
`child_stream_size_changed`	The size of the listed ADS for the listed file or directory has been changed.
`self_removed`	The directory from which then system streams notifications has been removed from the file system. **Note** No notifications follow a `self_removed` notification.

Streaming Change Notifications by Using the qq CLI

Run the `qq fs_notify` command and specify the path to a directory. For example:

```
qq fs_notify --path /my/directory
```

In this example, Qumulo Core streams all notification types (page 224) for files immediately under the `/my/directory` directory.

To terminate the stream, send a `SIGQUIT` signal.

Streaming Change Notifications by Using the Qumulo Core REST API

Make a `GET` request to the REST endpoint in the following format:

```
/v1/files/<ref>/notify&filter=<filter>&recursive=<recursive>
```

In the following example:

- `ref`: An absolute path or a numeric file ID for the directory to watch.
- `filter`: A comma-separated list of notification types (page 224).
- `recursive`: When set to `true`, enables recursive change notifications.

```
/v1/files/my/directory/notify&filter=child_file_added,child_dir_removed&recursive=tr
ue
```

NFS

Creating and Managing an NFS Export in Qumulo Core

This section explains how to create, modify, and delete an NFS export by using the Qumulo Core Web UI.

To Create an NFS Export

1. Log in to the Qumulo Core Web UI.
2. Click **Sharing** > **NFS Exports**.
3. On the right side of the **NFS Exports** page, click **Create Export**.
4. On the **Create NFS Export** page:
    a. Enter the **File system path** from the root of your file system.
    b. To create a new directory, click **Create new directory if it doesn't exist**.
    c. Enter the **Export path**.
    d. Enter the **Description** for the export.
    e. Under **Host Access Rules**, enter **Allowed Hosts** and specify:

- Whether the host has **Read-only access**
- The **User Mapping**

> **Note**
> Qumulo Core enforces host access rules in the order of appearance, top to bottom. We recommend adding rules specific to IP addresses and hosts to the top of the list and rules specific to subnets and host wildcards to the bottom. For more information see Configuring and Troubleshooting Host Access Rules (page 246).

To add a host, click **Add a Host Access Rule**.

To delete an existing host, click .

5. Click **Save**.

To Modify an NFS Export

1. Log in to the Qumulo Core Web UI.

2. Click **Sharing** > **NFS Exports**.
3. For an NFS Export, in the **Actions** column, click .
4. Make changes to your NFS Export (for more information, see To Create an NFS Export (page 229)) and then click **Save**.

To Delete an NFS Export

1. Log in to the Qumulo Core Web UI.
2. Click **Sharing** > **NFS Exports**.
3. For an NFS Export, in the **Actions** column, click .
4. In the **Delete Export** dialog box, click **Yes, Delete Export**.

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 238).

⚠ 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 addresses), 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.example.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.example.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.example.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 sanitization 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

> **Note**
> In Qumulo Core 7.0.0 (and higher), to greatly improve throughput, use the `nconnect=16` option to enable cross-connection write combining.

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 \
  -o nconnect=16 \
  your.example.cluster:/mount_path \
  /path/to/mount_point
```

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,nconnect=16 \
  your.example.cluster:/mount_path \
  /path/to/mount_point
```

> **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, run 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, run the following `qq` CLI command:

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

Configuring Floating IP Addresses 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 for each 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 for each node, on a cluster with 600 clients for each 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, run the following `qq` CLI command:

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

ℹ 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 252)
- 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 0).

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, run 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, run 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), run the nfs4_editfacl command (or the nfs4_setfacl command with the `-e` flag).

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 238), 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 238), 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 How NFSv4.1 Works with Kerberos in Qumulo Core (page 252).
`<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 How NFSv4.1 Works with Kerberos in Qumulo Core (page 252).

Trustee Representation	Example	Description
`<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.
`<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 239). 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 Core Web UI or the `qq` CLI. For the trustee in this ACE, the system stores this user as a local user.

Trustee Representation	Example	Description
`qumulo_local/<groupname>`	`qumulo_local/localgroup1`	A group local to a Qumulo cluster (that is, a group created by using the Qumulo Core 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 trustee 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`

NFSv4.1 Rights	Qumulo Rights
`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`
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'`

Configuring and Troubleshooting Host Access Rules for NFS Exports in Qumulo Core

This section explains how host access rules work in Qumulo Core and how to configure and troubleshoot them.

In Qumulo Core 6.2.0.1, you can add a host access rule to an NFS export to restrict the export by IP address or hostname.

The following examples show the elements that a host access rule can include.

- Hostnames
  - Without a wildcard (`name.example.com`)
  - With a wildcard (`*.example.com`)
- IP Addresses
  - Single IP addresses (`203.0.113.0`)
  - IP address range (`203.0.113.0-203.0.113.10` or `203.0.113.0-10`)
- Network Segment
  - Without a subnet mask (`203.0.113.0/24`)
  - With a subnet mask (`203.0.113.0/255.255.255.0`)
- Allowed Kerberos Security Flavors

  To restrict access to NFSv4.1 clients that use only specific Kerberos security flavors (page 252), add the following special strings to the list of host access rules. For example:
  - `KRB5P@` : Allow only encrypted access for the specified export.
  - `KRB5@` , `KRB5I@` , and `KRB5P@` : Allow any Kerberos-authenticated access, but not `AUTH_SYS` access.

> ⚠ Important
> If you don't specify a host access rule, Qumulo Core allows access to all IP addresses.

Prerequisites

To be able to use hostnames, you must:

- Enable and configure reverse look-ups on your DNS server.

- Use fully qualified domain names (FQDNs).
- Use wildcards carefully because they match only one hostname level. For example, `*.accounting.example.com` matches `user1.accounting.example.com` but not `machine.user1.accounting.example.com`.
- Optimize your system for reverse-dns look-ups. (page 250)

Adding a Host Access Rule to an Existing NFS Export

This section explains how you can add a host access rule to an existing NFS export by using the Qumulo Core Web UI or the `qq` CLI.

To Add a Host Access Rule by Using the Qumulo Core Web UI

1. Log in to the Qumulo Core Web UI.
2. Click **Sharing** > **NFS Exports**.
3. For an NFS export, in the **Actions** column, click .
4. On the **NFS Export** page, in the **Host Access Rules** section:
    a. For **Allowed Hosts**, enter a comma-separated host access rule. (page 246)
    b. (Optional) To ensure that the allowed hosts have limited access to the NFS export, click **Read-only**.
    c. (Optional) For **User mapping** select one of the following:
    - **No mapping:** Qumulo Core doesn't apply a user mapping when it accesses the NFS export and relies on default NFS protocol behavior.
    - **Map root to...:** Qumulo Core associates the root user that accesses the NFS export with a specific user in your Qumulo cluster.
    - **Map all to...:** Qumulo Core associates all users that access the NFS export with a specific user in your Qumulo cluster.

    d. To add a new rule, click **+ Add a Host Access Rule**.
    e. Click **Save**.

Qumulo Core applies the host access rule to the NFS export.

To Add a Host Access Rule by Using the qq CLI

1. Prepare a list of host access rules in JSON format. The following is an example of the contents of `root_restrictions.json`.

```
{
  "restrictions": [{
    "host_restrictions": [
      "user1.accounting.example.com",
      "*.eng.example.com",
      "203.0.113.0"
    ]
  }]
}
```

2. Run the `qq nfs_mod_export` command and specify the export path and the file with the host access rules. For example:

```
qq nfs_mod_export \
  --export-path / \
  --restrictions root_restrictions.json
```

The following is example output.

```
{
  "description": "",
  "export_path": "/",
  "fields_to_present_as_32_bit": [],
  "fs_path": "/",
  "id": "1",
  "restrictions": [{
    "host_restrictions": [
      "user1.accounting.example.com",
      "*.eng.example.com",
      "203.0.113.0"
    ],
    "read_only": false,
    "require_privileged_port": false,
    "user_mapping": "NFS_MAP_NONE"
  }],
  "tenant_id": 1
}
```

To Troubleshoot Host Access Rules for an NFS Export

This section describes the troubleshooting steps for a scenario in which an NFS client can't mount or access an NFS export.

> **Note**
> Currently, if you use multiple DNS servers, the `dns_resolve_hostnames` and `dns_resolve_ips` commands aren't tenant-aware and might not return the same results as the DNS resolution mechanism in NFS.

1. To view the NFS export's host access rules, run the `qq nfs_get_export` command and specify the export path. For example:

   ```
   qq nfs_get_export --export-path /
   ```

   The following is example output.

   ```
   ID:          1
   Export Path: /
   Tenant ID:   1
   FS Path:     /
   Description:
   32bit-mapped fields: None
   Host Access:
   ID  Hosts                          Access Options
   ==  =============================  ============================
   1   user1.accounting.example.com   rw, insecure, no_root_squash
   ```

   In this example, only the machine `user1.accounting.example.com` can access the NFS export at `/`.

2. To find the client's IP address, we recommend viewing your Qumulo cluster logs. For example:

   ```
   Client 203.0.113.2 is not authorized to use export ExportId(1)
   ```

3.

   To find the client's hostname, run the `qq dns_resolve_ips` command and specify the client's IP address. For example:

   ```
   qq dns_resolve_ips --ips 203.0.113.2
   ```

   The following is example output.

```
[{
  "hostname": "user2.accounting.example.com",
  "ip_address": "203.0.113.2",
  "result": "OK"
}]
```

In this example, the `203.0.113.2` IP address maps to `user2.accounting.example.com`.

4. To troubleshoot the NFS client, you can take one or more of the following steps:

- Ensure the NFS client configuration entry is correct.
- Run the `dns_resolve_ips` (page 249) command to verify that the IP address maps to the correct name.
- Update the host access rules for `user2.accounting.example.com`.
- Ensure that your Qumulo cluster's DNS cache isn't out of date, for example, if `203.0.113.2` should resolve to `user1.accounting.example.com`.

  To reset your Qumulo cluster's DNS cache, run the `qq dns_clear_cache` command.
- Run the `qq dns_resolve_hostnames` command and specify the hostname to perform a lookup for `user1.accounting.example.com`.

  The following is example output.

```
[{
  "hostname": "user2.accounting.example.com",
  "ip_addresses": ["203.0.113.1"],
  "result": "OK"
}]
```

- Run the `qq dns_resolve_ips` command to find the hostname for your client's IP address and:

◦ If the NFS client can't access a share, but should be able to, add the IP address to the NFS export's host access rules.

◦ If the NFS client can access a share, but shouldn't be able to, remove the IP address from the NFS export's host access rules.

Optimizing Your System for Reverse-DNS Look-Ups

Qumulo Core checks hostnames by performing a reverse-DNS lookup on the cluster. Because continuous reverse-DNS look-ups can affect system performance, Qumulo Core caches the results on the cluster. Because Qumulo Core's cache abides by the DNS TTL, a low TTL can cause cache entries to expire frequently, which might require a new query.

By increasing TTL, you can reduce the number of DNS requests that your cluster makes. However, this might cause your cluster to keep outdated results for a longer time. For the most optimal configuration, list your organization's DNS servers first in your DNS configuration.

To bypass DNS, you can set explicit IP-host mappings for your cluster by using the `qq dns_set_lookup_overrides` command. If Qumulo Core finds an override for an IP address or host, it uses the override instead of the DNS cache.

In the following JSON example, the IP address `203.0.113.2` binds to the host `user3.accounting.qumulo.com` explicitly.

```
{
  "lookup_overrides": [{
    "aliases": ["user3.accounting.example.com"],
    "ip_address": "203.0.113.2"
  }]
}
```

NFSv4.1 with Kerberos

How NFSv4.1 Works with Kerberos in Qumulo Core

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

Kerberos is a network authentication protocol that works by using a three-way trust between a key distribution center (KDC), a service server (for example, NFSv4.1 on Qumulo Core), and a client system (for example, a Linux system). This section 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).

Choosing a Kerberos Security Flavor

Qumulo Core supports three *flavors* of Kerberos security that NFSv4.1 clients can use by specifying the following mount options:

- `sec=krb5`: Provides user authentication only.
- `sec=krb5i`: Provides authentication and message integrity by performing message signing for protection against man-in-the-middle attacks and message tampering.
- `sec=krb5bp`: Provides privacy by encrypting all traffic between the client and server. This is the most secure mount option.

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 one of the available mount options (page 252).

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.

To join your cluster to Active Directory, log in to the Qumulo Core Web UI and click **Cluster** > **Active Directory**.

Using Active Directory (AD) for POSIX Attributes (RFC 2307)

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 Qumulo Core 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 Qumulo Core 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 look-ups and `PTR` records for reverse-DNS look-ups.

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 round-robin DNS 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 Round-Robin DNS

1. Join your Qumulo cluster to AD (page 254).
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 Qumulo Core Web UI, click **Cluster** > **Active Directory** and write down 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 Qumulo Core 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 Qumulo Core Web UI, click **Cluster** > **Active Directory** and write down 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 254), you must configure the NFSv4.1 server (page 231) 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 Qumulo Core REST API.

1. Run the `qq nfs_get_settings` command to get the current settings.

   The following is example output.

```
$ 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 run 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`	

Command	Description
`--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 Kerberos security flavor (page 252) 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 231) 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 Kerberos security flavors (page 252): `AUTH_SYS`, `AUTH_KRB5`, `AUTH_KRB5P`, and `AUTH_KRB5I`.

Using Kerberos Permissions in the Qumulo File System

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, run 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, run 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 Qumulo Core 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-1-5-21-1234567890-4567890123-9876543210-13507",
      ...
    },
    ...
  }, {
    "trustee": {
      "name": "AD\\Domain Users",
      "sid": "S-1-5-21-1234567890-4567890123-9876543210-513",
      ...
    },
    ...
  }]
}
```

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 238).

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 272).

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, run 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 272).

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 look-ups and `PTR` records for reverse-DNS look-ups.

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, run 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, run 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 file system, 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 run 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.example.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 Qumulo Core 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 Qumulo Core 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 in Qumulo Core.

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, run 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 the Qumulo Care team.

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 254).

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 run 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.

SMB

Creating and Managing an SMB Share in Qumulo Core

This section explains how to create, modify, and delete an SMB share by using the Qumulo Core Web UI.

To Create an SMB Share

1. Log in to the Qumulo Core Web UI.
2. Click **Sharing** > **SMB Shares**.
3. On the right side of the **SMB Shares** page, click **Create Share**.
4. On the **Create SMB Share** page:
    a. Enter the **File system path** from the root of your file system.
    b. To create a new directory, click **Create new directory if it doesn't exist**.
    c. Enter the **Share name** (for example, `\\203.0.113.0\my-share`).
    d. Enter the **Description** for the share.
    e. To display only the files and directories to which the user has read access, click **Enable access-based enumeration**.
    f. To force users to connect over SMB3 (and higher) by using encryption-enabled clients, click **Require encryption**.
    g. Under **Share Permissions**, enter trustees and specify their:

- **Permission type:** Click **Add allow** or **Add deny**.

> **Note**
> To ensure that Qumulo Core processes users to whom it explicitly denies access before processing users to whom it grants access, **Deny** entries appear at the top of the list and **Allow** entries at the bottom.

- **Permissions:** Click **Read**, **Write**, or **Change Permissions**.

    To delete a trustee, click .

    h. Under **Advanced Options**:
        a. Enter the **Default file create mode** (`0644` by default).
        b. Enter the **Default directory create mode** (`0755` by default).

5. Click **Create Share**.

To Modify an SMB Share

1. Log in to the Qumulo Core Web UI.
2. Click **Sharing** > **SMB Shares**.
3. For an SMB share, in the **Actions** column, click .
4. Make changes to your SMB share (for more information, see To Create an SMB Share (page 285)) and then click **Save**.

To Delete an SMB Share

1. Log in to the Qumulo Core Web UI.
2. Click **Sharing** > **SMB Shares**.
3. For an SMB share, in the **Actions** column, click .
4. In the **Delete Share** dialog box, click **Yes, Delete Share**.

Managing Qumulo Core File Shares by Using the Shared Folders MMC Snap-In

By connecting the Shared Folders Microsoft Management Console (MMC) Snap-In to your Qumulo cluster, you can manage file shares centrally.

> **Note**
> To perform the following operations, you must use a Qumulo Core account with the Administrators or Data Administrators role.

Prerequisites

Windows 7, Windows Server 2008 R2 (or higher)

To Connect a Windows Machine to the SMB Shares on Your Qumulo Cluster

1. In Command Prompt, run the `net use` command to connect to your Qumulo cluster. For example:

   ```
   net use \\mycluster.example.com /user:admin
   ```

2. When prompted, enter your credentials.
3. In MMC, click **File** > **Add/Remove Snap-in...**
4. In the **Add or Remove Snap-ins** dialog box, under **Available snap-ins**, click **Shared Folders**, and then click **Add**.
5. In the **Shared Folders** window, in the left pane, right-click **Shared Folders (<JSMITH-WORK-DESKTOP>)**, and then click **Connect to another computer...**
6. In the **Shared Folders** dialog box:
   a. Under **This snap-in will always manage**, click **Another computer** and enter your cluster's hostname, for example `mycluster.example.com`.

      > **Important**
      > Don't place a slash (/) after your cluster's hostname. Although the Shared Folders snap-in doesn't display an error message in case of failure, adding the slash after the hostname causes the operation to fail.

   b. Under **View**, click **All**.

c. Click **Finish**.

MMC adds the Shared Folders snap-in to its left pane, under **Console Root**.

7. (Optional) To view the existing shares on your Qumulo cluster, expand **Shared Folders (\MYCLUSTER.EXAMPLE.COM)** and then click **Shares**.

To Connect Your Qumulo Cluster to a Share on Your Windows Machine

> **Note**
> Shared folder permissions are unrelated to NTFS access control lists (ACLs).

1. In MMC, click **Shared Folders (<JSMITH-WORK-DESKTOP>)** and then on the menu bar click **Action > New Share...**
2. In the **Create a Shared Folder Wizard: Folder Path** dialog box, for **Folder path**, specify the path for the SMB share (for example `C:\Users\Jennifer\Reports`) and then click **Next**.

   > **Note**
   > If you click **Browse...**, you might see the **q$** parent item in the directory tree. The is the root (/) of the file system.

3. In the **Create a Shared Folder Wizard: Name, Description, and Settings** dialog box:
   a. Enter the **Share name**, for example `jennifer-reports`.
   b. Enter the **Share path**, for example `\\JSMITH-WORK-DESKTOP\jennifer-reports`.
   c. Click **Next**.
4. In the **Create a Shared Folder Wizard: Shared Folder Permissions** dialog box, select one of the following permissions for the shared folder:

- All users have read-only access
- Administrators have full access; other users have read-only access
- Administrators have full access; other users have no access
- Customize permissions

> **Tip**
> We recommend clicking **Customize permissions** and then, in the dialog box, giving **Full Control** to the group **Everyone**.

5. Click **Finish**.

6. To confirm that your SMB share is visible to your Qumulo cluster, log in to the Web UI and click **Sharing** > **SMB Shares**.

To Change the Configuration for an Existing Share on Your Windows Machine

1. In MMC, click **Shared Folders (<JSMITH-WORK-DESKTOP>)** and then in the right pane right-click an existing share and then click **Properties**.
2. In the **File Properties** dialog box, on the **General** tab, you can edit properties such as **Share name**, **Folder path**, **Description**, and **User Limit**.

   > **Note**
   >
   > The settings of the **User limit** and **Offline Settings...** properties have no effect on your Qumulo cluster.
   >
   > The path `C:\` represents the root (`/`) of your Qumulo file system.

3. On the **Share Permissions** tab, ensure that the permissions for various users and groups are correct.

   > **Note**
   >
   > It is possible to add local users and groups from your Qumulo cluster by prefixing them with the name of your cluster followed by a backslash (\). For example: `accounting\admin`
   >
   > Because, if you *don't* select **Full Control**, there is a minor mismatch between the rights that Windows Change Permissions grants and the Qumulo Core Web UI `Write` permissions. For this reason, share permissions configured by using the MMC appear with an asterisk (*) in the Qumulo Core Web UI. We recommend using Qumulo Core to configure share permissions.

4. On the **Security** tab, ensure that the file permissions for the directory subtree under the share are connected.
5. To save changes, click **Apply**.It is possible to add local users and groups from your Qumulo cluster by prefixing them with the name of your cluster followed by a backslash (`\`). For example: `accounting\admin`

   > **Important**
   > For a Qumulo cluster with default settings, applying changes results in an error for the `.snapshots` meta-directory. It is safe to click **Continue** each time this error occurs.

S3 API

Configuring and Using the S3 API in Qumulo Core

This section explains how to configure and get started working with the S3 API. This API lets clients and applications interact with the Qumulo file system natively, by using the Amazon S3 API.

Prerequisites

To use the S3 API, you must install the `aws` and `qq` CLI (page 124).

> ⚠ Important
> The following instructions are for Ubuntu 18.04 (and higher).

Step 1: Configure HTTPS

The Qumulo Core S3 API accepts only HTTPS requests by default. To enable HTTPS support for your Qumulo cluster, you must install a valid SSL certificate on it.

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 configuring HTTPS for your cluster, see Installing the Qumulo Core Web UI SSL Certificate on Qumulo Care.

Enabling and Disabling Plaintext HTTP Connections

> ⚠ Important
> If you configure the S3 API service to accept only plaintext HTTP connections, no requests made through the S3 API are encrypted.

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

Step 2: Enable the S3 API for Your Qumulo Cluster

To let your Qumulo cluster accept S3 traffic, you must enable the S3 API by using the `qq s3_modify_settings --enable` command.

After you run the command, all nodes in your cluster begin to accept S3 API traffic on TCP port 9000.

Step 3: Create an Access Key Pair

To create and manage S3 buckets you must have a valid S3 access key pair associated with a specific user in your Qumulo cluster or in a connected external identity provider (such as Active Directory). For more information, see Creating and Managing S3 Access Keys (page 295).

Use the `qq s3_create_access_key` command and specify the username. For example:

```
$ qq s3_create_access_key my-username
```

> **Note**
> After Qumulo Core initially creates your secret access keys, it never logs or displays them again. If you lose your secret access key, it isn't possible to recover it and you must create a new access key pair.

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

To create and manage S3 buckets, you must configure AWS CLI to work with your Qumulo cluster.

> **Note**
>
> - We recommend configuring a dedicated profile for Qumulo in your AWS CLI S3 Configuration.
> - Qumulo Core listens for S3 API traffic on TCP port 9000. It isn't possible to change this setting.
> - Currently, Qumulo Core supports only path-style bucket addressing. For more information, see Bucket Addressing Style (page 338).

1. Configure the AWS CLI to use path-style bucket addressing by using the `aws configure` command and specify your profile.

```
$ aws configure \
  --profile my-qumulo-profile set s3.addressing_style path
```

2. Use the access key pair that you have created earlier (page 291) and the `aws configure` command to:

    a. Specify your profile and access key ID (page 295). For example:

    ```
    $ aws configure
      --profile my-qumulo-profile set aws_access_key_id \
        000000000001fEXAMPLE
    ```

    b. Specify your profile and secret access key (page 295). For example:

    ```
    $ aws configure
      --profile my-qumulo-profile set aws_secret_access_key \
        TEIT4liMZ8A32iI7JXmqIiLWp5co/jmkjEXAMPLE
    ```

3. Because it isn't possible to specify your cluster's URI persistently, create a shell alias to specify your cluster's URI, in the following format:

```
$ alias aws="aws --endpoint-url https://<qumulo-cluster>:9000 --profile my-qum
ulo-profile"
```

> **Note**
> If you haven't installed an SSL certificate, append `--no-verify-ssl` to the end of the command.

4. (Optional) If you haven't configured your machine to trust the SSL certificate installed on your Qumulo cluster, to configure the path to the trusted SSL certificate bundle that you have created and installed earlier (page 290) manually, run the `aws configure` command. For example:

```
$ aws configure \
  --profile my-qumulo-profile set ca_bundle MySpecialCert.crt
```

5. To test your configuration, send an S3 API request to your Qumulo cluster by using the `aws s3api list-buckets` command.

   A successful response includes an empty JSON array named `Buckets`.

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

Step 5: Create an S3 Bucket

> **Note**
> Creating buckets requires the `PRIVILEGE_S3_BUCKETS_WRITE` role-based access control (RBAC) (page 90) privilege and permission to create a directory under the cluster's root directory.

Run the `aws s3api create-bucket` command and specify the bucket name. For example:

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

The S3 API creates the new directory `/my-bucket/`. All of the bucket's objects are located under this directory. For more information, see Creating and Working with S3 Buckets in Qumulo Core (page 303).

Step 6: Test Writing and Reading S3 Objects

1. To test writing data to your Qumulo cluster, perform a `PutObject` S3 API action by using the `aws s3api put-object` command. For example:

   ```
   $ aws s3api put-object \
     --bucket my-bucket \
     --key archives/my-remote-file.zip \
     --body my-local-file.zip
   ```

   The S3 API uploads the contents of `my-local-file.zip` into an object named `my-remote-file.zip`.

2. To test reading read data from and S3 bucket, perform a `GetObject` S3 API action by using the `aws s3api get-object` command. For example:

```
$ aws s3api get-object \
  --bucket my-bucket \
  --key archives/my-remote-file.zip local-file.zip
```

The S3 API downloads the contents of the `my-remote-file.zip` object into `local-file.zip` and returns the object metadata. 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

This section explains how to create and manage credentials that S3 API actions in Qumulo Core require to access file system resources, such as access key pairs that sign requests.

> **Note**
> You can configure an S3 bucket to allow read-only, anonymous access (page 313). This approach requires no credentials but limits users to non-modifying operations.

Prerequisites

Managing S3 access keys requires the following role-based access control (RBAC) (page 90) privileges:

- `PRIVILEGE_S3_BUCKETS_WRITE`: Create and delete S3 access keys
- `PRIVILEGE_S3_BUCKETS_READ`: List S3 access keys

How S3 Access Keys Work in Qumulo Core

An *identity* is a single principal from an identity provider (IdP). Examples of identities include SMB security identifiers (SIDs), Active Directory user principal names (UPNs), POSIX user identifiers (UIDs), and local users in a Qumulo cluster.

> **Important**
> It isn't possible to create access keys for UIDs in an Active Directory environment that has POSIX extensions enabled. However, it is possible to use Active Directory identity identifiers (SIDs, UPNs, and so on).

An *access key* (or *access key pair)* is comprised of an S3 access key ID and an S3 secret access key.

- The *access key ID* is the public component of an S3 access key pair. It identifies the user that performs an S3 request.
- The *secret access key* (or *secret key)* is the private component of an S3 access key pair. The client uses the secret access key to sign requests and the server uses the secret access key to validate request signatures.

⚠ Important

- Qumulo Core uses a cryptographically secure source, certified according to FIPS 140-2 requirements, to derive secret access keys.
- Because access keys are cluster-local, you can't use an access key for an identity in one Qumulo cluster on a different Qumulo cluster.

Qumulo Core creates an access key pair whenever an authorized user requests it. For more information, see Creating S3 Access Keys for a Qumulo Cluster (page 297).

The way in which Qumulo Core access keys let you access your Qumulo cluster makes the process similar to the way in which IAM Access Keys let you access Amazon S3 resources. For this reason, applications that access objects stored in a Qumulo cluster can use the Qumulo S3 API similarly to the native Amazon S3 API.

How S3 Access Keys work with Identities

An S3 access key doesn't grant any additional permissions. It associates an S3 API request with a specific identity (page 295) that the Qumulo cluster knows.

When Qumulo Core processes a request, it evaluates permissions by using the Qumulo ACL (QACL) mechanism that operates like the access control list (ACL) mechanism that all file system protocols use. When the QACL grants or denies permissions to an associated identity, it also grants or denies the same permissions to the request being processed.

For more information, see Managing Access to S3 Buckets in a Qumulo Cluster (page 312).

How Qumulo Core Stores S3 Access Keys

To authenticate S3 API requests, Qumulo Core retrieves existing access key pairs that it stores securely as configuration metadata in your Qumulo cluster. Qumulo Core encrypts secret access keys on disk and holds decrypted secret access keys in memory only while it processes a request.

⚠ Important
Because (unlike *secret access keys)* your *access key IDs* aren't a cryptographic secret, Qumulo Core *can* log and display access key IDs. After Qumulo Core initially creates your secret access keys, it never logs or displays them again. If you lose your secret access key, it isn't possible to recover it and you must create a new access key pair.

S3 Access Key Lifecycle in Qumulo Core

Qumulo Core doesn't limit how long you can use an access key pair after you create it. Your system administrators must take responsibility for using the Qumulo Core REST API or `qq` CLI to view the creation dates for access keys and revoke any pair at their discretion.

For more information, see Listing S3 Access Keys for a Qumulo Cluster (page 299).

Note

- To facilitate key rotation, each user identity (page 295) can have at most two S3 access key pairs associated with it. It is a good practice to delete a user's old access key after you create a new one and test that the new key works.
- If you revoke an access key pair, it isn't possible to restore it. Before you revoke an access key pair, ensure that no critical applications depend on it.

Creating S3 Access Keys for a Qumulo Cluster

To make S3 API requests to a Qumulo cluster as a specific user, you must create an S3 access key pair for that user identity (page 295) by using the Qumulo REST API or `qq` CLI.

To create S3 access keys, you must have an administrator account or have the `PRIVILEGE_S3_BUCKETS_WRITE` privilege.

To Create an Access Key by Using the qq CLI

To create an S3 access key for a particular user identity (page 295), run the `qq s3_create_access_key` command and specify an identity. For example:

```
$ qq s3_create_access_key my_identity
```

You can specify an identity by using:

- A name, optionally qualified with a domain prefix:
  - `ad:MY_NAME`
  - `AD\MY_NAME`
  - `local:MY_NAME`
  - `MY_NAME`
- An Active Directory Security Identifier. For example: `SID:S-1-1-0`
- A Qumulo *auth ID*, Qumulo Core's common representation for identities, in the form of a numeric identifier. For example: `auth_id:513`

Important
Currently, it isn't possible to associate an S3 access key with a POSIX group ID (GID).

The following is example output.

```
{
  "access_key_id": "AKIAIOSFODNN7EXAMPLE",
  "creation_time": "2022-12-12T21:37:53.553457928Z",
  "owner": {
    "auth_id": "501",
    "domain": "LOCAL",
    "gid": null,
    "name": "guest",
    "sid": "S-0-1-23-4567890123-456789012-345678901-234",
    "uid": null
  },
  "secret_access_key": "wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY"
}
```


In this example, the access key id is `000000000001fEXAMPLE` and the secret access key is `TEIT4liMZ8A32iI7JXmqIiLWp5co/jmkjEXAMPLE`.

> **Important**
> After Qumulo Core initially creates your secret access keys, it never logs or displays them again. If you lose your secret access key, it isn't possible to recover it and you must create a new access key pair.

To Create an S3 Access Key by Using the Qumulo Core REST API

Send a `POST` request to the `/v1/s3/access-keys/` endpoint with the following body. You must include at least one of the following keys:

- `auth_id`
- `sid`
- `uid`

For example:


```
{
  "user": {
    "sid": "S-0-1-23-4567890123-456789012-345678901-234"
  }
}
```


The following is example output.

```
{
  "access_key_id": "AKIAIOSFODNN7EXAMPLE",
  "creation_time": "2022-12-12T21:37:53.553457928Z",
  "owner": {
    "auth_id": "501",
    "domain": "LOCAL",
    "gid": null,
    "name": "guest",
    "sid": "S-0-1-23-4567890123-456789012-345678901-234",
    "uid": null
  },
  "secret_access_key": "wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY"
}
```


In this example, the access key id is `000000000001fEXAMPLE` and the secret access key is `TEIT4liMZ8A32iI7JXmqIiLWp5co/jmkjEXAMPLE`.

> **Important**
> After Qumulo Core initially creates your secret access keys, it never logs or displays them again. If you lose your secret access key, it isn't possible to recover it and you must create a new access key pair.

Listing S3 Access Keys for a Qumulo Cluster

You can list every S3 access key that your Qumulo cluster knows, along with the identities associated with the key and the key creation times, by using the Qumulo REST API or `qq` CLI.

To list S3 access keys, you must have the `PRIVILEGE_S3_BUCKETS_READ` privilege.

> **Note**
> Qumulo Core doesn't list access keys in any particular order. To sort keys according to fields such as `creation_time` or `owner` you must process or filter the response.

To List S3 Access Keys by Using the qq CLI

- To list the S3 access keys that your Qumulo cluster knows, run the `qq s3_list_access_keys` :

  The following is example output. All times are in the UTC time zone.

```
access_key_id         owner  creation_time
====================  =====  ==============================
000000000001fEXAMPLE  Guest  2022-12-12T21:37:53.553457928Z
```

- For JSON output, use the `--json` flag.

  The following is example output. The command returns a single JSON object that contains the combined responses from calls to the `/v1/s3/access-keys/` Qumulo Core REST API endpoint.

```
{
  "entries": [
    {
      "access_key_id": "AKIAIOSFODNN7EXAMPLE",
      "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 S3 Access Keys by Using the Qumulo Core REST API

To list the S3 access keys that your Qumulo cluster knows, send a `GET` request to the `/v1/s3/access-keys/` endpoint.

> **Note**
> To restrict the number of returned results, up to the maximum of 10,000 access keys (this is the default limit), include the optional `limit` query parameter in the request.

The following is example output. The `entries` list contains the access keys, limited to the first 10,000. The `paging.next` field contains the URI to which you can send a `GET` request to retrieve the next page of access keys. By making `GET` requests with all returned `paging.next` values, you can iterate over all of the access keys in the cluster.

```
{
  "entries": [
    {
      "access_key_id": "AKIAIOSFODNN7EXAMPLE",
      "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
  }
}
```

Revoking S3 Access Keys for a Qumulo Cluster

To revoke an S3 access key, you must delete the access key from your Qumulo cluster. You can delete an S3 access key by using the Qumulo REST API or `qq` CLI.

To revoke an access key, you must have the `PRIVILEGE_S3_BUCKETS_WRITE` privilege.

To Delete an S3 Access Key by Using the qq CLI

Run the `qq s3_delete_access_key` command and specify the access key ID. For example:

```
$ qq s3_delete_access_key \
  --id 000000000001fEXAMPLE
```

To Delete an S3 Access Key by Using the Qumulo Core REST API

Send a `DELETE` request to the `/v1/s3/access-keys/<access-key-id>` Qumulo Core REST API endpoint and specify the access key ID.

Configuring Active Directory (AD) for S3

> **Note**
> To be able to create access keys for a user in a joined AD domain, the user must exist within the domain's base DN.

For users that exist in an AD domain that has a trust relationship with the joined domain, you must append that domain's base DN to the base DN in your Qumulo cluster's AD configuration.

To append the trusted base DN to the base DN in use—with a semicolon (`;`) separating the two—use the Qumulo Core Web UI or the `qq ad_reconfigure` command. For example:

```
$ qq ad_reconfigure \
  --base-dn 'CN=Users,DC=joined_domain,DC=example,DC=com;CN=Users,DC=trusted_domai
n,DC=example,DC=com'
```

For more information, see Configuring Cross-Domain Active Directory Trusts (page 280)

Creating and Managing S3 Buckets in Qumulo Core

This section explains how to create and manage S3 buckets for a Qumulo cluster. These buckets expose a part of your Qumulo file system to applications that use the Amazon S3 API.

You can create and work with S3 buckets by using the Qumulo REST API or `qq` CLI. You can also use the S3 API directly.

Prerequisites

To create and manage S3 buckets by using the Qumulo REST API or `qq` CLI, you need the following role-based access control (RBAC) (page 90) privileges:

- `PRIVILEGE_S3_BUCKETS_WRITE`: Create and delete S3 buckets

  > **Note**
  > If you perform create and delete operations on directories by using the `qq` CLI, you also need this privilege.

- `PRIVILEGE_S3_BUCKETS_READ`: List S3 buckets

To create and manage S3 buckets by using the S3 API, you also need:

- A valid Qumulo S3 access key (page 295)
- A configured AWS CLI (page 291)

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.

The *bucket root directory* (or *bucket root)* is the directory to which you attach an S3 bucket. All files under the bucket root directory (and all of its subdirectories) are objects in the bucket. The presence of the slash (`/`) in objects' keys determines the directory hierarchy.

> **Important**
> Because S3 buckets can use any directory in the file system as a root directory, the same file can be an object in multiple buckets.

How the Qumulo File System Determines Object Keys

The *object key* in a Qumulo S3 bucket is its file system path, relative to the bucket's root directory. Only objects that are directories have a trailing slash (`/`) in their keys.

The following example shows the contents of a Qumulo file system.

```
/
├── website-data/
│   └── publish.dat
│
└── application-data/
    ├── develop.dat
    ├── processing/
    └── deployment/
        ├── data1.dat
        └── data2.dat
```

In this example, if you have the S3 bucket `bucket1` with its root directory at `/application-data/deployment/`, the bucket contains objects with the following keys:

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

However, if you have the S3 bucket `bucket2` with its root directory at `/application-data/`, the bucket contains objects with the following keys:

- `develop.dat`
- `processing/`
- `deployment/data1.dat`
- `deployment/data2.dat`

> **Note**
> In this example:
>
> - Both buckets contain `/application-data/deployment/data1.dat` and `/application-data/deployment/data2.dat` as objects.
> - The `processing/` object in `bucket2` has a trailing slash because it is a directory.

How to Name an S3 Bucket

When you create an S3 bucket, you name it. A bucket's name doesn't need to be related to its root directory.

Except for names that contain the period (`.`), Qumulo Core accepts all names that conform to the following Amazon S3 bucket naming rules.

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

How to Choose a Bucket Root

You specify the bucket root directory (page 303) depending on how you create your S3 bucket.

- When you create an S3 bucket by using the Qumulo REST API or `qq` CLI, you can choose a directory to use as the bucket root.
- When you create an S3 bucket by using the `CreateBucket` S3 API action, the API creates a new directory with the same name as the bucket under the default bucket directory prefix. For more information, see Configuring the Default Bucket Directory Prefix for S3 Buckets (page 305).
- If you don't specify a directory, the Qumulo Core REST API and `qq` CLI use the default bucket directory prefix (page 305).

The user that creates a new directory for a new bucket owns the directory. For more information, see Managing Access to S3 Buckets in a Qumulo Cluster (page 312).

Creating S3 Buckets

You can create an S3 bucket by using the Qumulo REST API or `qq` CLI. You can also use the S3 API directly.

While the Qumulo Core REST API and `qq` CLI let you use an existing directory as the new bucket root, the S3 API always creates a new directory for the bucket root.

> ⚠ Important
>
> - All S3 buckets in a Qumulo cluster share the same namespace: It isn't possible to create two buckets with the same name, even if they use different directories as their bucket root.
> - All S3 buckets must follow the bucket naming rules (page 304).

Configuring the Default Bucket Directory Prefix for S3 Buckets

The *default bucket directory prefix* is the directory under which Qumulo Core creates new bucket root directories when it creates S3 buckets by using the `CreateBucket` S3 API action or when you create an S3 bucket without specifying a directory by using the Qumulo REST API or `qq` CLI.

By default, the default bucket directory prefix for newly created buckets is the cluster's root directory (`/`). Thus, if you create a bucket named `my-bucket`, its root directory is `/my-bucket`.

- To view the current default bucket directory prefix by using the Qumulo REST API or `qq` CLI, you need the `PRIVILEGE_S3_BUCKETS_READ` privilege.
- To change the default bucket directory prefix, you need the `PRIVILEGE_S3_BUCKETS_WRITE` privilege.
- To enable or suspend S3 bucket versioning, use the `qq s3_modify_bucket` command.

To Configure the Default Bucket Directory Prefix by Using the qq CLI

1. To view the current default bucket directory prefix, run the `qq s3_get_settings` command.

   The following is example output.

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

2. To change the setting, run the `qq s3_modify_settings` command and specify the new default bucket directory prefix. In the following example, we specify `/buckets`.

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

Creating an S3 Bucket by Using the qq CLI

To create an S3 bucket by using the Qumulo REST API or `qq` CLI, you need the `PRIVILEGE_S3_BUCKETS_WRITE` privilege.

> ⚠ Important
> In Qumulo Core 6.0.1.1 (and higher), the `qq` CLI command changed from `s3_create_bucket` to `s3_add_bucket` and the flag for specifying the directory path has changed from `--path` to `--fs-path`.

When you use the `qq` CLI to create a bucket, you can use a new or existing directory as the bucket root.

> **Note**
> If an entry with the specified name or directory already exists, or if you don't have permission to create a directory, the command returns an error. For more information, see Configuring the Default Bucket Directory Prefix for S3 Buckets (page 305).

- To create a new, empty bucket from the default bucket directory prefix (page 305), run the `qq s3_add_bucket` command and specify the bucket name. For example:

  ```
  $ qq s3_add_bucket \
    --name my-bucket
  ```

  Qumulo Core creates a new directory named `my-bucket` under the default bucket directory prefix.

- To create a bucket from an existing directory, run the `qq s3_add_bucket` command and specify the bucket name and the directory path. For example:

  ```
  $ qq s3_add_bucket \
    --name my-bucket \
    --fs-path /products/web/appliances/
  ```

- To create a bucket for a path that doesn't exist yet, specify the name and path and add the `--create-fs-path` flag. For example:

  ```
  $ qq s3_add_bucket \
    --name my-bucket \
    --fs-path /products/web/appliances/ \
    --create-fs-path
  ```

Creating an S3 Bucket by Using the S3 API

Run the `aws s3api create-bucket` command and specify the bucket name. This command uses the `CreateBucket` S3 API action. For example:

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

Qumulo Core creates the bucket root directory under the default bucket directory prefix (page 305) and names it the same as the bucket. In this example, if the default bucket directory prefix is `/buckets/`, the new bucket root directory is `/buckets/my-bucket/`.

> **Note**
> When you use the `CreateBucket` S3 API action with the `LocationConstraint` parameter, the Qumulo S3 API supports only the `local` region.

Configuring S3 Buckets

You can view and modify the settings for individual buckets by using the Qumulo REST API or `qq` CLI.

You can configure global settings, such as the default bucket directory prefix (page 305) for all S3 buckets. For more information about configuring anonymous access for individual S3 buckets, see, see Enabling Anonymous Access for an S3 Bucket (page 313).

- To view the current bucket configuration by using the Qumulo REST API or `qq` CLI, you need the `PRIVILEGE_S3_BUCKETS_READ` privilege. For more information, see `qq s3_get_bucket` in the Qumulo `qq` CLI Command Guide.
- To change the bucket configuration, you need the `PRIVILEGE_S3_BUCKETS_WRITE` privilege. For more information, see `qq s3_modify_bucket` in the Qumulo `qq` CLI Command Guide.

Enabling Object Lock for S3 Buckets

An Object Lock can prevent an S3 object from being deleted for a limited time or indefinitely. For information about file locks in Qumulo Core, see Managing File Locks (page 89).

To Enable Object Lock for an S3 Bucket by Using the qq CLI

1. Use an account with the `S3_BUCKETS_WRITE` role-based access control (RBAC) role (page 90).
2. Run the `qq s3_modify_bucket` command, specify the bucket name, and use the `--enable-object-locking` flag. For example:

```
$ qq s3_modify_bucket \
  --name my-bucket \
  --enable-object-locking
```

To Enable Object Lock for an S3 Bucket by Using the S3 API

> **Note**
> Currently, because Qumulo Core doesn't support default retention periods, you can set only a compliance retention period on an individual object in an S3 bucket by using the `PutObjectRetention` API action.

1. To enable versioning for your S3 bucket, use the `PutBucketVersioning` API action.
2. To enable Object Lock for your S3 bucket, use the `PutObjectLockConfiguration` API action.

Listing S3 Buckets

You can list all S3 buckets in your Qumulo cluster by using the Qumulo REST API or `qq` CLI. You can also use the S3 API directly.

To List S3 Buckets by Using the qq CLI

To list your S3 buckets by using the Qumulo REST API or `qq` CLI, you need the `PRIVILEGE_S3_BUCKETS_READ` privilege.

- Run the `qq s3_list_buckets` command.

  The following is example output. All times are in the UTC time zone.

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

- For JSON output, use the `--json` flag.

  The following is example output. All times are in the UTC time zone. The JSON output contains an array named `Buckets` that contains the individual buckets as objects.

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

To List S3 Buckets by Using the S3 API

Run the `aws s3api list-buckets` command. This command uses the `ListBuckets` S3 API action.

The following is example output. All times are in the UTC time zone. The JSON output contains an array named `Buckets` that contains the individual buckets as objects.

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

Deleting S3 Buckets

You can delete an S3 bucket by using the Qumulo REST API or `qq` CLI. You can also use the S3 API directly.

While the Qumulo Core REST API and `qq` CLI let you choose whether to also delete the bucket root directory, the S3 API always deletes the bucket root directory.

> **Note**
> Before you delete your S3 bucket, you must either let all in-progress upload operations for the bucket (`UploadPart`, `PutObject`, or `CopyObject`) complete or you must abort the operations.

Deleting an S3 Bucket by Using the qq CLI

To delete an S3 bucket by using the Qumulo REST API or `qq` CLI, you need the `PRIVILEGE_S3_BUCKETS_WRITE` privilege.

When you use the `qq` CLI to delete a bucket, you can choose to also delete the bucket root directory.

- To delete an S3 bucket, but not its root directory, run the `qq s3_delete_bucket` command and specify the bucket name. For example:

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

  This command doesn't delete the bucket root directory. It deletes all metadata related to the bucket from your Qumulo cluster.

If any of the following conditions are true, the command returns an error:

- The specified bucket doesn't exist.
- You don't have the `PRIVILEGE_S3_BUCKETS_WRITE` privilege.
- The bucket has in-progress upload operations (`UploadPart` , `PutObject` , or `CopyObject`).

- To delete a bucket together with its root directory, use the `qq s3_delete_bucket` command, specify the bucket name, and use the `--delete-root-dir` flag. For example:

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

If any of the following conditions are true, the command returns an error:

- You don't have permission to delete the bucket root directory.
- The bucket root directory isn't empty.

Deleting an S3 Bucket by Using the S3 API

Run the `aws s3api delete-bucket` command and specify the bucket name. This command uses the `DeleteBucket` S3 API action. For example:

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

This command deletes the bucket root directory and all metadata related to the bucket from your Qumulo cluster.

If any of the following conditions are true, the command returns an error:

- The specified bucket doesn't exist.
- You don't have permission to delete the bucket root directory.
- The bucket root directory isn't empty.
- The bucket has in-progress upload operations (`UploadPart` , `PutObject` , or `CopyObject`).

Managing Access to S3 Buckets in a Qumulo Cluster

This section explains how to manage access to S3 buckets in a Qumulo cluster.

Managing user access to S3 buckets in a Qumulo cluster is very similar to managing access to SMB shares and NFS exports, with the following exceptions:

- To let a user access S3 buckets in the cluster, you must assign an S3 access key (page 295) to the user. Alternatively, you can create presigned URLs (page 312) or enable read-only, anonymous access (page 313) for the entire S3 bucket.
- Because a Qumulo cluster restricts S3 actions based on file access control lists (ACLs), an S3 bucket might work differently or have more restrictive permissions than expected.

  > **Note**
  > To configure an S3 bucket in Qumulo Core to work more like an Amazon S3 bucket, use inheritable access control entries (ACEs) to imitate bucket-level permissions (page 315).

How S3 Bucket Permissions Work in Qumulo Core

To process an S3 API request, Qumulo Core performs one or more file system operations. Qumulo Core processes these operations by checking the user's access against the access control lists (ACLs) for each file that is part of the request.

> **Note**
> To permit an action to be performed, the bucket policy (page 319) and the object's file system ACL must allow the action.

For authenticated requests signed with Amazon Signature Version 4, Qumulo Core maps the access key ID (page 295) in the request to its corresponding auth ID (page 297), and then processes the request as that user. Qumulo Core processes unsigned, anonymous requests as the `Guest` user.

While Qumulo Core processes an S3 request, the ownership of any newly created files and directories belongs to the user that makes the request. These files and directories inherit access control entries (ACEs) from their parents (this process is the same for all protocols).

Granting Access to S3 Buckets by Using Presigned URLs

To let trusted users perform S3 API actions—such as `GetObject` or `UploadPart`—as if using your user account, you can generate a *presigned URL* (also known as *query parameter authentication*), associate the URL with specific API actions, and then share it with trusted users. Every presigned URLs has a configurable expiration time that ensures that the URL stops working at the configured time.

For more information, see Authenticating Requests: Using Query Parameters (AWS Signature Version 4) in the Amazon Simple Storage Service API Reference.

> **Note**
> Qumulo Core accepts only presigned requests that use the `PUT`, `GET`, `HEAD`, and `DELETE` HTTP methods. Qumulo Core rejects presigned requests for `POST` requests, such as the following:
>
> - `AbortMultipartUpload`
> - `CompleteMultipartUpload`
> - `CreateMultipartUpload`
> - `DeleteObjects`

To create a presigned URL, run the AWS CLI `presign` command. In the following example, the presigned URL expires in 10 minutes (600 seconds).

```
$ aws2 s3 presign s3://my-bucket/my-file.txt \
  --endpoint-url https://203.0.113.0:9000 \
  --profile my-qumulo-profile \
  --expires-in 600
```

The following is example output. The `X-Amz-Expires` header is set to 10 minutes.

```
https://203.0.113.0:9000/my-bucket/my-file.txt?
X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=00000000000003e88527%2F20230217%2Fus-east-1%2Fs3%2Faws4_request
&X-Amz-Date=20230217T205559Z
&X-Amz-Expires=600
&X-Amz-SignedHeaders=host
&X-Amz-Signature=141fa5b10caaa8575ba9c065d2270a24ce14b2ff58bb2c2e98382c76297b21ee
```

Enabling Anonymous Access for an S3 Bucket

In certain cases, it might be more practical to allow anonymous (unauthenticated) requests to access the contents of S3 buckets, for example, if you want to let users access objects from the S3 bucket by using a web browser or if the number of users who need read access is very large. When you enable anonymous access to an S3 bucket, your users can perform read-only S3 operations without authenticating their requests.

> **Important**
> Anonymous requests can never perform modifying operations. Qumulo Core requires all modifying operations on an S3 bucket to be authenticated.

When you enable anonymous access for an S3 bucket, Qumulo Core performs all anonymous requests as 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 to the `Guest` user. For more information, see Imitating Bucket-Level Read-Only Access (page 317).

> **Note**
> If a file's ACL doesn't allow reads for the `Guest` user, an anonymous request can't read the file.

- To view the current bucket policy configuration by using the Qumulo REST API or `qq` CLI, you need the `PRIVILEGE_S3_BUCKETS_READ` privilege. For more information, see `qq s3_get_bucket_policy` in the Qumulo `qq` CLI Command Guide.
- To change the bucket policy configuration, you need the `PRIVILEGE_S3_BUCKETS_WRITE` privilege. For more information, see `qq s3_modify_bucket_policy` in the Qumulo `qq` CLI Command Guide.

  The following is an example policy that enables anonymous access:

```
{
  "Id": "Anonymous Access Enabled",
  "Statements": [{
    "Action": [
      "s3:*"
    ],
    "Effect": "Allow",
    "Principal": {
      "Qumulo": ["Authenticated Users"]
     },
    "Sid": "Authenticated Full Access"
  },{
    "Action": [
      "s3:GetObject",
      "s3:GetObjectAcl",
      "s3:GetObjectAttributes",
      "s3:GetObjectTagging",
      "s3:ListBucket"
    ],
    "Effect": "Allow",
    "Principal": {
      "Qumulo": ["local:guest"]
    },
    "Sid": "Read-only Guest Access"
  }],
  "Version": "2012-10-17"
}
```

Using Inheritable ACEs to Imitate Bucket-Level Permissions

To grant multiple users access to all paths in a bucket and ensure that newly created directories inherit the correct permissions, use inheritable access control entries (ACEs).

In Amazon S3, permission to read objects from —and write objects to— an S3 bucket applies to the entire bucket. In Qumulo Core, each object key (page 303) corresponds to a file path relative to a bucket's root directory. Qumulo Core grants permissions for individual files and directories.

When users create objects in an S3 bucket in a Qumulo cluster, they might also create new directories. The user that creates these directories owns them. However, without the correct access control entries (ACEs) in your bucket, these directories might have restrictive permissions that prevent other users from creating objects with the same prefix.

How Permissions with Inheritable ACEs Work

Access control entries (ACEs) control the permissions that users have for files and directories in a Qumulo cluster. When you add ACEs to a directory and mark them as *inheritable,* all new files and directories created in that directory inherit those ACEs and pass them on.

You can use inheritable ACEs to:

- Imitate bucket-level permissions by ensuring that any files and directories that your users create in an S3 bucket receive the same permissions.

  To make all paths in an S3 bucket inherit the same set of ACEs, add the ACEs to the bucket's root directory and mark them as inheritable.

- Configure default permissions for newly created buckets.

  To make a set of ACEs the default for buckets that your users create by using the S3 API, add the ACEs to the default bucket directory prefix.

To add ACEs to a directory, use the `qq` CLI or use the File Explorer on a Windows client with a mapped SMB share that contains the directory.

> **Note**
> Adding inheritable ACEs to a directory doesn't affect any files that already exist in that directory. For more information, see To Recursively Add a New ACL with Multithreading.

Imitating Bucket-Level Permissions by Using the qq CLI

The following sections show how to use the `qq` CLI to imitate bucket-level permissions by adding inheritable ACEs.

Imitating Bucket-Level Read-Write Access

Run the `qq fs_modify_acl` command. In the following example, we add the access control entry (ACE) to the bucket whose root directory is `/buckets/my-bucket` for the user group `MyWriters`.

```
$ qq fs_modify_acl \
  --path /buckets/my-bucket add_entry \
  --trustee MyWriters \
  --type Allowed \
  --flags 'Container inherit' 'Object inherit' \
  --rights 'Delete child' 'Execute/Traverse' 'Read' 'Write file'
```

The ACE imitates bucket-level read-write access for a user or group of users.

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

Imitating Bucket-Level Read-Only Access

Run the `qq fs_modify_acl` command. In the following example, we add the access control entry (ACE) to the bucket whose root directory is `/buckets/my-bucket` for the user group `MyReaders`:

```
$ qq fs_modify_acl
  --path /buckets/my-bucket add_entry \
  --trustee MyReaders \
  --type Allowed \
  --flags 'Container inherit' 'Object inherit' \
  --rights 'Execute/Traverse' 'Read'
```

The ACE imitates bucket-level read-only access for a user or group of users.

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

Imitating Bucket-Level List-Only Access

Run the `qq fs_modify_acl` command. In the following example, we add two access control entries (ACEs) to the bucket whose root directory is `/buckets/my-bucket` for the user group `MyListers`.

```
$ qq fs_modify_acl
  --path /buckets/my-bucket add_entry \
  --trustee MyListers \
  --type Allowed \
  --flags 'Container inherit' \
  --rights 'Execute/Traverse' 'Read'
```

```
$ qq fs_modify_acl
  --path /buckets/my-bucket add_entry \
  --trustee MyListers \
  --type Allowed \
  --flags 'Object inherit' \
  --rights 'Read attr'
```

The two ACEs imitate bucket-level list-only access for a user or group of users:

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

Managing Access Policies for S3 Buckets in a Qumulo Cluster

This section explains how to manage access policies for S3 buckets in a Qumulo cluster.

Access policies let you control specific sets of S3 API actions that each user or group can perform. They provide an *additional* layer of access management for S3 buckets by adding further restrictions to those of access keys, presigned URLs, and file system access control lists (page 315).

Managing access policies for S3 buckets in Qumulo clusters is similar to managing SMB share access, only with a larger set of items that you can specify in the `Actions` (page 323) field of the policy statement (page 320).

For information about working with access policies for S3 buckets and for `qq` CLI examples, see the following sections in the Qumulo `qq` CLI Command Guide:

- `qq s3_get_bucket_policy`
- `qq s3_set_bucket_policy`
- `qq s3_modify_bucket_policy`
- `qq s3_delete_bucket_policy`

Anonymous Access to S3 Buckets

By default, S3 buckets in a Qumulo cluster are in a *no policy* state, in which Qumulo Core disallows unsigned, anonymous requests and the `qq s3_get_bucket_policy` command returns `{}`.

In Qumulo Core, anonymous S3 connections use the system `Guest` account, which is restricted to read-only S3 API actions. To permit anonymous access in an S3 bucket policy, grant access to one of the following principals:

- The `Everyone` group
- The `Guest` account
- Any group that includes the `Guest` account as a member

> **Note**
> When you upgrade Qumulo Core version 7.1.1, the system replaces anonymous S3 bucket access configuration with a default S3 bucket policy that permits all S3 API actions to all principals.

Prerequisites

The following prerequisites let you manage the access policy for an S3 bucket effectively.

- Grant your users access to the S3 bucket by using S3 access keys (page 295) or presigned URLs (page 312), or enable read-only, anonymous access (page 313) to the S3 bucket.
- Configure inheritable file ACLs (page 315) by using the `qq` CLI, SMB, or NFSv4.1 access control lists (ACLs) (page 238).
- Ensure that you have the following required role-based access control (RBAC) (page 90) privileges.
    - `PRIVILEGE_S3_BUCKETS_READ`
    - `PRIVILEGE_S3_BUCKETS_WRITE`
- (Optional) To delegate the management of an access policy for an S3 bucket to another user, grant the `s3:PutBucketPolicy` and `s3:DeleteBucketPolicy` S3 API actions to that user in the `Actions` (page 323) field of a policy statement.

How Policy Statements for S3 Buckets are Structured

Policy statements for S3 buckets use the JSON format. For example:

```
{
  "Id": "Example overall access policy description",
  "Statements": [{
    "Action": [
      "s3:GetBucketPolicy",
      ...
    ],
    "Effect": "Allow",
    "Index": 1,
    "Principal": {
      "Qumulo": ["Everyone"]
     },
    "Sid": "Example policy statement description"
  },{
    ...
  }],
  "Version": "2012-10-17"
}
```

To retrieve an example policy file, run the `qq s3_get_bucket_policy --example` command.

The S3 bucket policy statement contains the following fields.

Field Name	Description
(Optional) `Id`	Describes the functionality of your overall policy

Field Name	Description
`Statements`	Contains a list of statements, and the following fields for each policy statement **Note** The order of the fields has no effect on the permissions that an access policy grants for an S3 bucket. - `Action` : Specifies a list of API actions supported in Qumulo clusters (page 323) to which the policy statement applies - `Effect` : Specifies either `Allow` or `Deny` **Note** ◦ Unless the policy statement has at least one matching `Allow` statement and no `Deny` statements for an action, the system outputs the `AccessDeniedByBucketPolicy` error. ◦ For the S3 API and Qumulo REST API, if a user has the role-based access control privilege (RBAC) to perform an API action, Qumulo Core ignores the access policy (page 326) and permits the API action. ◦ The `ListBuckets` S3 API action has no associated access policy permission in Qumulo Core. Instead, this S3 API action checks each S3 bucket's policy and includes the S3 bucket in the enumeration if *any* action is allowed for a user. - `Index` : The system ignores this field when you configure the access policy for an S3 bucket. **Note** ◦ To retrieve index for a policy statement, run the `qq s3_get_bucket_policy` command. ◦ You can target a specific policy statement by specifying its index for the `--index` flag with the `qq s3_modify_bucket_policy modify_statement` command.

Field Name	Description
	• `Principal` : Specifies a list of users or groups (in various formats (page 325)) to which the policy statement applies This field uses the same identity specification as the `identifier` field of the `qq auth_find_identity` command. • `Sid` : Describes the functionality of your policy statement
(Optional) `Version`	If you specify this field, enter `2012-10-17` , the latest policy version from Amazon. For more information, see IAM JSON Policy Elements: Version.

Actions Supported in Qumulo Core

The following table describes the subset of the Amazon S3 API Actions which Qumulo Core supports.

> **Note**
>
> - Certain permissions (such as `s3:AbortMultipartUpload`) grant permission to both S3 API and Qumulo Core REST API variants of an API call.
> - Certain permissions (such as `s3:GetBucketAcl`) grant permission to S3 APIs that are currently implemented partially within Qumulo Core.
> - `s3:*` matches all S3 API actions.

API Action	Description
`s3:AbortMultipartUpload`	Abort a multipart upload to the S3 bucket
`s3:DeleteBucket`	Delete the S3 bucket
`s3:DeleteBucketPolicy`	Remove the access policy from the S3 bucket
`s3:DeleteObject`	Delete any object from the S3 bucket
`s3:DeleteObjectTagging`	Delete all tags from any object in the S3 bucket
`s3:DeleteObjectVersion`	Delete any object version from the S3 bucket

API Action	Description
`s3:GetBucketAcl`	Retrieve the access control list (ACL) for the S3 bucket
`s3:GetBucketLocation`	Retrieve the region in which the S3 bucket is located **Note** Currently, because Qumulo Core doesn't use regions, the system always returns `local`.
`s3:GetBucketNotification`	Retrieve the notification configuration for the S3 bucket
`s3:GetBucketObjectLockConfiguration`	Retrieve the object lock configuration for the S3 bucket
`s3:GetBucketPolicy`	Retrieve the bucket policy for the S3 bucket
`s3:GetBucketReplication`	Retrieve the replication state for the S3 bucket
`s3:GetBucketVersioning`	Retrieve the versioning state for the S3 bucket
`s3:GetEncryptionConfiguration`	Retrieve the encryption state for the S3 bucket
`s3:GetLifecycleConfiguration`	Retrieve the lifecycle configuration for the S3 bucket
`s3:GetObject`	Download any object from the S3 bucket **Note** The file system permissions take precedence over this permission.
`s3:GetObjectAcl`	Download the access control list (ACL) for any object in the S3 bucket
`s3:GetObjectAttributes`	Retrieve the attributes for any object in the S3 bucket
`s3:GetObjectTagging`	Retrieve the tags for any object in the S3 bucket
`s3:ListBucket`	Enumerate all objects in the S3 bucket

API Action	Description
`s3:ListBucketMultipartUploads`	Enumerate all multipart uploads to the S3 bucket
`s3:ListMultipartUploadParts`	Enumerate all multipart upload parts in the S3 bucket
`s3:PutBucketPolicy`	Configure the access policy for the S3 bucket
`s3:PutBucketVersioning`	Configure bucket versioning for the S3 bucket
`s3:PutObject`	Write or overwrite any object in the S3 bucket
`s3:PutObjectTagging`	Configure tags for any object in the S3 bucket

Principals Supported in Qumulo Core

The following table describes examples of principals which Qumulo Core supports.

Identity Specification Example	Description
`Mary Lou`	A username or group name
`local:Jane`	A user or group created by using the Qumulo Core REST API in the `local` domain, prefixed by `local:`
`local:guest`	An anonymous connection
`world:Everyone`	Any user connected to Qumulo Core, including unauthenticated, anonymous connections
`Authenticated Users`	Any authenticated user, excluding guest or anonymous connections
`EXAMPLE_DOMAIN\Jose Ramirez`	A user or group in a specific Active Directory domain, prefixed by the domain name
`ad:Company Name`	A user or group in any connected Active Directory domain, prefixed by `ad:`
`uid:1234`	A POSIX UID that identifies users by their RFC 2307 , prefixed by `uid:`

Identity Specification Example	Description
`gid:1234`	A POSIX GID that identifies users by their RFC 2307 , prefixed by `gid:`
`auth_id:12345678`	The numeric `auth_id` of a user or group
• `S-1-5-1234` • `sid:S-1-5-5678`	A Windows-style security identifier (SID), optionally prefixed by `sid:`

Role-Based Access Control (RBAC) Overrides

For the S3 API and Qumulo REST API, if a user has the role-based access control privilege (RBAC) to perform an API action, Qumulo Core ignores the access policy (page 326) and permits the API action.

The following table describes the relationship between Qumulo Core privileges and the S3 API actions associated with them.

Qumulo Core Privilege	Associated S3 API Actions
`S3_BUCKETS_READ`	• `s3:GetBucketPolicy` • `s3:GetBucketVersioning`
`S3_BUCKETS_WRITE`	• `s3:DeleteBucket` • `s3:DeleteBucketPolicy` • `s3:PutBucketPolicy` • `s3:PutBucketVersioning`
`S3_UPLOADS_READ`	• `s3:ListMultipartUploadParts` • `s3:ListBucketMultipartUploads`
`S3_UPLOADS_WRITE`	`s3:AbortMultipartUpload`

Managing Multipart S3 Uploads in Qumulo Core

This section explains how multipart S3 uploads affect usable capacity on a Qumulo cluster and how to abort and clean up multipart uploads manually or automatically.

Qumulo Core supports the multipart upload functionality of the S3 API, which lets you upload objects to a bucket in parts and then, at a later time, combine these parts into a single object.

> **Note**
> For objects above a certain size (typically, larger than 100 MiB), applications often use the multipart S3 uploads, rather than the `PutObject` S3 API action. The limitation for the `PutObject` action is 5 GiB. For more information about how Qumulo handles this type of operation, see System-Initiated Multipart S3 Uploads (page 328).

Prerequisites

To manage multipart S3 uploads by using the `qq` CLI, you need the following role-based access control (RBAC) (page 90) privileges:

- `PRIVILEGE_S3_SETTINGS_WRITE`: Configure frequency of multipart upload cleanup
- `PRIVILEGE_S3_UPLOADS_READ`: List multipart uploads
- `PRIVILEGE_S3_UPLOADS_WRITE`: Abort multipart uploads

How Multipart S3 Uploads Affect Usable Capacity on a Qumulo Cluster

The following conditions are true for multipart S3 uploads in Qumulo Core.

- To let you resume large uploads in the event of an outage, Qumulo Core stores data on the cluster durably.
- Multipart upload data isn't visible in the Qumulo file system, and isn't included in file system snapshots, until you complete the upload successfully by making a call to the `CompleteMultipartUpload` S3 API.

  > **Note**
  > When you view the breakdown of a Qumulo cluster's capacity by using the Qumulo Core Web UI, REST API, or `qq` CLI, Qumulo Core doesn't distinguish between capacity that the file system and incomplete multipart uploads use.

- Qumulo Core doesn't delete multipart data unless it aborts and cleans up the multipart upload automatically (page 330) or you abort and clean up the multipart upload manually (page 331).

To check how much space incomplete multipart uploads use on your cluster, you can list the uploads by using the Qumulo Core REST API or `qq` CLI. For more information, see Listing Multipart Uploads (page 328).

How System-Initiated Multipart S3 Uploads Work

Occasionally, when you list your multipart uploads (page 328), you might see uploads that you didn't initiate. These are *system-initiated uploads* which Qumulo Core uses for `PutObject` and `CopyObject` S3 API actions for objects that exceed a certain size.

If Qumulo Core encounters an error while performing a system-initiated upload, it attempts to abort the upload and clean up the partial upload data immediately.

However, if Qumulo Core is unable to clean up the incomplete upload data immediately, it cleans up the incomplete upload data in the background, according to the expiry interval (page 330).

> **Note**
> The process for background clean-up after incomplete and user-initiated uploads is the same. For more information, see Aborting and Cleaning Up Multipart S3 Uploads Automatically (page 330).

Listing Incomplete Multipart S3 Uploads

You can list the incomplete multipart uploads for a single S3 bucket by using the Qumulo Core REST API or `qq` CLI.

> **Note**
>
> - If you use the `ListMultipartUploads` S3 API action, the system doesn't show system-initiated uploads (page 328) or how much space the uploads use on your cluster.
> - If you use the Qumulo Core REST API or `qq` CLI, Qumulo Core shows system-initiated uploads (page 328) and how much space each upload uses on your cluster.

- To list incomplete uploads by using the `qq` CLI, run the `qq s3_list_uploads` command and specify the bucket name. For example:

```
$ qq s3_list_uploads \
  --bucket my-bucket
```

- To list incomplete uploads by using the Qumulo Core REST API, send a `GET` request to the `/v1/s3/buckets/<bucket-name>/uploads/` endpoint and specify the bucket name.

The output from the `qq` CLI and REST API is the same. The following example output is a single JSON object that contains the list of objects for the specified bucket. The list shows information for each multipart S3 upload, including:

- When each upload was initiated
- Which identity initiated the upload
- When the upload received data last
- How much space the upload uses on the cluster—by data, by metadata, and in total—in units of blocks (4,096 bytes per block)

```
{
  "uploads": [
    {
      "bucket": "my-bucket",
      "completing": false,
      "datablocks": "16384",
      "id": "00000000example1",
      "initiated": "2023-03-02T19:01:00.446468848Z",
      "initiator": {
        "auth_id": "500",
        "domain": null,
        "gid": null,
        "name": null,
        "sid": null,
        "uid": null
      },
      "key": "deployment/data1.dat",
      "last_modified": "2023-03-02T19:03:37.209271702Z",
      "metablocks": "3",
      "system_initiated": false,
      "total_blocks": "16387"
    },
    {
      "bucket": "my-bucket",
      "completing": false,
      "datablocks": "24576",
      "id": "00000000example2",
      "initiated": "2023-03-02T19:09:04.530619255Z",
      "initiator": {
        "auth_id": "500",
        "domain": null,
        "gid": null,
        "name": null,
        "sid": null,
        "uid": null
      },
      "key": "release.dat",
      "last_modified": "2023-03-02T19:09:06.436699236Z",
      "metablocks": "4",
      "system_initiated": true,
      "total_blocks": "24580"
    }
  ]
}
```

Aborting and Cleaning Up Multipart S3 Uploads Automatically

Qumulo Core automatically aborts and cleans up an incomplete multipart S3 if the upload doesn't receive any data after the configured *expiry interval* (1 day by default).

When Qumulo Core removes a multipart upload, it frees up the space that the upload uses on the cluster. You can configure the expiry interval by using the Qumulo Core REST API or `qq` CLI.

To configure the expiry interval for all current and future multipart uploads by using the `qq` CLI, run the `qq s3_modify_settings` command and the `--multipart-upload-expiry-interval` flag and specify one of the following:

- The string `never`.
- A string in the format `<quantity><units>` (without a space), where `<quantity>` is a positive integer less than 100 and `<units>` is one of the following strings:
  - `days`
  - `hours`
  - `minutes`
  - `months`
  - `weeks`

In the following example, we instruct Qumulo Core to abort and clean up uploads that haven't received data in more than 30 days.

```
$ qq s3_modify_settings \
  --multipart-upload-expiry-interval 30days
```

In the following example, we disable automatic cleanup.

```
$ qq s3_modify_settings \
  --multipart-upload-expiry-interval never
```

Aborting or Cleaning Up Multipart S3 Uploads Manually

Use the Qumulo Core REST API or `qq` CLI to abort and clean up the upload. You need the bucket name and upload ID. For more information about looking up this information, see Listing Incomplete Multipart S3 Uploads (page 328).

> **Note**
> If you are an administrative user or the user who initiated the upload, you can use the `AbortMultipartUpload` S3 API action. In addition to the bucket name and upload ID, you also need the object key for the upload.

- To abort an upload by using the `qq` CLI, run the `qq s3_abort_upload` command and specify the upload ID. For example:

```
$ qq s3_abort_upload \
  --bucket my-bucket \
  --upload-id 000000000example
```

- To abort an upload by using the Qumulo Core REST API, send a `DELETE` request to the `/v1/s3/buckets/<bucket-name>/uploads/<upload-ID>` endpoint and specify the upload ID. For example:

```
DELETE /v1/s3/buckets/my-bucket/uploads/000000000example
```

There is no response body for both the `qq` CLI and REST API. Qumulo Core returns a `204 No Content` status code when the upload is aborted or the cleanup is complete.

Managing S3 Bucket Versioning in a Qumulo Cluster

This section explains how Amazon S3 Versioning works in Qumulo Core and how to configure S3 bucket versioning by using the Qumulo REST API or `qq` CLI or by using the S3 API directly.

You can configure S3 bucket versioning by using the Qumulo REST API or `qq` CLI. For more information, see `qq s3_modify_bucket` in the Qumulo `qq` CLI Command Guide. You can also use the S3 API directly.

For information about S3 bucket versioning limits, see Supported Functionality and Limits (page 339).

How Bucket Versioning works in Qumulo Core

Qumulo Core stores S3 object versions as independent files in the file system. Qumulo Core names these versions by using their version ID.

In the following example:

- The versioned files are stored in `/dir/.s3-versioning/obj`.

  > **Important**
  > Because `.s3-versioning~` is a normal, hidden directory, visible and accessible to other protocols. To ensure consistent S3 API behavior, we strongly recommend avoiding the use of other protocols to write to the directory. Any protocol can read from the directory safely.

- An object with the object store key `dir/obj` has two versions: `12345a6b-7c89-0d12-3456-78fe9012f345` and `abcde1f2-g3hi-j4kl-mnop-qr56stuv7wxy`.

  > **Note**
  > - To access a specific version of a file by using CLI or REST API, specify the version ID.
  > - To access the *head version* (current version) of a file, omit the omit the version ID.

- The system-created hard link points to the latest version of the object.
- The on-disk representation is as follows:

```
/
└── dir/
    ├── .s3-versioning~/
    |   └── obj/
    |       ├── 12345a6b-7c89-0d12-3456-78fe9012f345
    |       └── abcde1f2-g3hi-j4kl-mnop-qr56stuv7wxy
    └── obj <hardlink: /dir/.s3-versioning~/obj/12345a6b-7c89-0d12-3456-78fe90
12f345>
```

Supported Functionality and Known Limits for S3 in Qumulo Core

This section documents Qumulo Core support for S3 API functionality and S3 API limits.

Supported S3 API Actions

The following table lists the S3 API actions that Qumulo Core supports and the version from which support begins. For the full list of S3 API actions, see Actions in the Amazon Simple Storage Service API Reference.

> **Note**
>
> - The S3 API became generally available in Qumulo Core 5.3.3. This guide doesn't document enabling or using API actions that became available with preview functionality in versions of Qumulo Core lower than 5.3.3.
> - The Qumulo S3 protocol creates data that supports all file system functionality, such as quotas, snapshots, and replication.

API Action	Supported from Qumulo Core Version
`AbortMultipartUpload`	5.3.3
`CompleteMultipartUpload`	5.3.3
`CopyObject`	5.3.3
`CreateBucket`	5.2.3
`CreateMultipartUpload`	5.3.3
`DeleteBucket`	5.2.4
`DeleteBucketVersioning`	7.1.2
`DeleteBucketPolicy`	7.0.1.1
`DeleteObject`	5.2.1
`DeleteObjects`	5.2.2
`DeleteObjectTagging`	6.3.2

API Action	Supported from Qumulo Core Version
`GetBucketAcl`	6.1.1
`GetBucketLocation`	5.1.2
`GetBucketVersioning`	7.1.2
`GetBucketPolicy`	7.0.0.1
`GetBucketVersioning`	7.0.0.1
`GetObject`	5.0.4
`GetObjectLockConfiguration`	7.2.1
`GetObjectRetention`	7.2.1
`GetObjectTagging`	7.1.2
`GetObjectAcl`	6.1.1
`GetObjectLockConfiguration`	7.0.0.1
`HeadBucket`	5.1.2
`HeadObject`	5.0.4
`ListBuckets`	5.0.4
`ListMultipartUploads`	5.3.3
`ListObjects`	5.0.5
`ListObjectsV2`	5.0.4
`ListParts`	5.3.3
`PutBucketPolicy`	7.0.1.1
`PutBucketVersioning`	7.1.2
`PutObject`	5.2.1
`PutObjectLockConfiguration`	7.2.1
`PutObjectRetention`	7.2.1

API Action	Supported from Qumulo Core Version
`PutObjectTagging`	6.3.2
`UploadPart`	5.3.3
`UploadPartCopy`	6.0.2

Unsupported S3 Functionality

The following table lists some of the S3 API functionality that Qumulo Core doesn't support.

Unsupported Feature	Description
BitTorrent	—
Bucket ACLs	For comparable functionality, use inheritable access control entries (ACEs) (page 315).
Bucket lifecycle configurations	—
Bucket notifications	—
Control of server-side encryption	All Qumulo Core data is encrypted at rest. You can't control this functionality by using the S3 API.
Logging controls	—
Multi-chunk payload signing	Qumulo Core doesn't support the streaming version of Amazon Signature Version 4 (SigV4), only the single-chunk version.
Object locks	—
Signature Version 2	Qumulo Core supports only SigV4 signatures.
Storage classes	Qumulo Core doesn't use the storage class concept. All objects have the same storage class status.
Retention policies	—
Temporary access credentials	—

Unsupported Feature	Description
Virtual-hosted bucket addressing	Qumulo Core supports only path-style bucket addressing.
Web hosting configuration	—

S3 API Limitations

This section describes the most important S3 API limitations in Qumulo Core.

Bucket Addressing Style

Because Qumulo Core supports only path-style bucket addressing, you must configure your client applications to use path-style addressing to send S3 API requests to a Qumulo cluster. For more information, see Configuring the AWS CLI for Use with Qumulo Core (page 291).

ETags

RESTful APIs, such as the S3 API, use HTTP ETags to identify different resource versions.

- Qumulo Core uses a proprietary mechanism to generate an object's ETag.
- Amazon S3 uses the MD5 checksum of an object's contents as its ETag.

> **Important**
> Well-behaved applications shouldn't attempt to interpret the contents of an ETag. However, certain applications do assume that S3 object ETags contain the MD5 checksum of the object's contents. Such applications might not function properly with the Qumulo S3 API.

Listing Objects

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

Function	Qumulo Core	Amazon S3
Returning results	Consistent but non-alphabetical order	Alphabetical order, by object key
Arbitrary prefix	Partial support for `Prefix`, only if `Prefix` is a path to a file or directory under the bucket root directory (page 303)	`Prefix` limits results to object keys that begin with the prefix

Function	Qumulo Core	Amazon S3
Arbitrary delimiter	Only the slash (`/`) character can act as `Delimiter`	`Delimiter` groups results into common prefixes

> **Note**
> Although Qumulo Core supports `Prefix` and `Delimiter` partially, it supports the most common use case—listing the contents of S3 buckets as a hierarchical file tree—fully.)))

Request Authentication

Qumulo Core supports authenticating requests by using only Amazon Signature Version 4. Most S3 client applications support this authentication type.

If your application attempts to use a previous Amazon signature version, you receive a `400 Bad Request` response with the error code `AuthorizationHeaderMalformed`.

Versioning

- **Object Version Limits:** In Qumulo Core, S3 bucket versioning is consistent with that of Amazon S3, with the exception of individual object versions. Qumulo Core limits directories to approximately 4.3 billion child files. The approach that Qumulo Core takes to indexing files in a directory might cause object creation commands to output the `QumuloDirectoryEntryLimitReached` error when a directory gets close to its capacity. Because Qumulo Core gives object versions unique identifiers, it might be possible to retry the command successfully. However, if you begin to observe this error, we recommend removing previous object versions from your system.
- **Creating Empty Versioned Directories:** Qumulo Core doesn't support creating empty, versioned directories.
- **Deleting Versioned Objects:** If you don't specify an object version ID, the DeleteObject and DeleteObjects S3 API actions create a *deletion marker* for an object but don't delete any file system data. Because currently Qumulo Core doesn't support bucket lifecycle policies, the data remains accessible by using S3 API actions and the object version ID. To delete a specific object version permanently, specify its version ID when you use either of these API actions.

Comparison of Known Limits between S3 in Qumulo and Amazon

This section compares the Qumulo Core S3 API limits with native Amazon S3 limits.

Limits for S3 Buckets

Limit	Qumulo Core	Amazon S3
Maximum number of buckets	16,000	1,000
Maximum number of objects in one bucket	Nominally unlimited	Unlimited
Minimum bucket name length	3 characters	
Maximum bucket name length	63 characters	

Note
If all objects in a bucket are under the same directory—none of the object keys have the slash (/) character in them—the maximum number of objects in the bucket is limited to the maximum number of files in a directory. For more information, see Supported Configurations and Known Limits for Qumulo Core (page 13).

Limits for S3 Objects

Limit	Qumulo Core	Amazon S3
Minimum object size	0 bytes	
Maximum object size (by using `PutObject`)	5 GiB	
Maximum object size (by using `MultipartUpload`)	48.8 TiB (10,000 * 5 GiB)	5 TiB
Minimum object key length	1 character	
Maximum object key length	1,530 characters, if there are no slash (`/`) characters in the key	1,024 characters
Maximum object versions	4,294,967,296 (theoretical)	Unlimited

Limits for S3 Multipart Uploads

Limit	Qumulo Core	Amazon S3
Minimum part ID	1	
Maximum part ID	10,000	
Minimum number of parts for each upload	1	
Maximum number of parts for each upload	10,000	
Minimum part size	5 MiB (except for the last part of an upload)	
Maximum part size	5 GiB	
Additional part size requirements	Must be a multiple of 4 KiB (4,096 bytes), except for the last part of an upload	—

Limits for S3 API Requests

Maximum Limit	Qumulo Core	Amazon S3
Object keys that `DeleteObjects` specifies	Nominally unlimited	1,000
Buckets that `ListBuckets` returns	16,000	1,000
Objects that `ListObjects` and `ListObjectsV2` return	1,000	
Parts that `ListParts` returns	Unlimited	1,000
Uploads that `ListMultipartUploads` returns	1,000	

Note
`DeleteObjects` is subject to a 10 MiB request payload limit in Qumulo Core. This provides a practical upper limit on the number of object keys that the API action can specify.

In addition, the following API actions have the Qumulo-specific maximum payload size limit of 10 MiB.

- `CompleteMultipartUpload`
- `CreateBucket`
- `DeleteObjects`

Monitoring and Metrics

Enabling Cloud-Based Monitoring and Remote Support

This section explains how to enable Cloud-Based Monitoring and Remote Support for your Qumulo cluster.

How Cloud-Based Monitoring Works

Enabling Cloud-Based Monitoring (page 345) lets the Qumulo Care team monitor your Qumulo cluster proactively.

> **⚠ Important**
> Cloud-Based Monitoring *doesn't* collect file names, path names, client IP addresses, or account credentials.

We use a proprietary application that aggregates diagnostic cluster data and alerts the Qumulo Care team if an issue arises. Depending on the issue severity and cluster state, a member of the Qumulo Care team reaches out. For more information, see:

- Qumulo Care Response Times
- Ways to Get Help

How Remote Support Works

Enabling Remote Support (page 346) lets the Qumulo Care team access your Qumulo cluster solely to assist you with a software update or perform diagnostics or troubleshooting on your cluster from the command line.

When you install VPN keys in the `/etc/openvpn` directory, an authorized member of the Qumulo Care team uses SSH to connect to the `ep1.qumulo.com` server and then uses SSH through a secure VPN connection to connect to your cluster (normally, this VPN connection is closed).

By default, the VPN tunnel remains open for four hours to allow members of the Qumulo Care team to perform operations such as uploading logs to `monitor.qumulo.com` or to a secured Amazon S3 bucket and sending diagnostic data to a private Amazon EC2 instance for analysis.

> **ℹ Note**
> Currently, Qumulo Core doesn't support VPN connections with IPv6.

You can configure the connection period and enable or disable Remote Support at any time.

What Data Gets Sent to Qumulo

Cloud-Based Monitoring and Remote Support let your cluster send the following detailed diagnostic data to Qumulo through an encrypted connection.

- Cluster name
- Number of nodes in cluster
- Hardware and software incidents
    - Drives
        - CRC errors
        - S.M.A.R.T. status alerts
        - Capacity triggers
    - Nodes
        - PSU failure
        - Fan failure
        - Recused node
        - Offline node
        - Unreachable cluster
    - Qumulo Core
        - New process core dump
- Configuration data (such as users, groups, SMB shares, and NFS exports)
- Logs, stack traces, and code dumps

Prerequisites

Before you can use Cloud-Based Monitoring and Remote Support:

- A member of the Qumulo Care Team must install VPN keys on your Qumulo cluster
- You must enable the following destination hostnames for TCP on port 443

Hostname	Description
`api.nexus.qumulo.com`	Nexus monitoring

Hostname	Description
`ep1.qumulo.com`	Remote Support **⚠ Important** If your organization has an intrusion detection device or a firewall that performs SSL or HTTPS deep-packet inspection, you must add an exception to the IP address that resolves to `ep1.qumulo.com`. To identify this IP address, log in to your Qumulo cluster and run the `nslookup ep1.qumulo.com` command.
`api.missionq.qumulo.com`	Cloud-Based Monitoring connectivity
`missionq-dumps.s3.amazon-aws.com`	Proxy forwarding
`monitor.qumulo.com`	Cloud-Based Monitoring log uploads

Enabling Cloud-Based Monitoring

You can enable Cloud-Based Monitoring by using the Qumulo Core Web UI or `qq` CLI.

To Enable Cloud-Based Monitoring by Using the Qumulo Core Web UI

1. Log in to the Qumulo Core Web UI.
2. Click **Support** > **Qumulo Care**.
3. On the **Qumulo Care** page:
    a. In the **Cloud-Based Monitoring** section, click **Edit**.
    b. Click **Yes, I want Qumulo Cloud-Based Monitoring** and then click **Save**.

If your configuration is valid, the Qumulo Core Web UI shows the status **Enabled | Connected**.

To Enable Cloud-Based Monitoring by Using the qq CLI

- To enable Cloud-Based Monitoring, run the `qq set_monitoring_conf --enabled` command.
- To disable Cloud-Based Monitoring, run the `qq set_monitoring_conf --disabled` command.

- To check the status of Cloud-Based Monitoring, run the `qq monitoring_conf` command.

Enabling Remote Support

You can enable Remote Support by using the Qumulo Core Web UI or `qq` CLI.

To Enable Remote Support by Using the Qumulo Core Web UI

1. Log in to the Qumulo Core Web UI.
2. Click **Support** > **Qumulo Care**.
3. On the **Qumulo Care** page:
    a. In the **Remote Support** section, click **Edit**.
    b. Under **Do you want to enable Qumulo Remote Support?**, click **Yes** and then click **Save**.

If your configuration is valid, the Qumulo Core Web UI shows the status **Enabled | Connected**.

To Enable Remote Support by Using the qq CLI

- To enable Remote Support, run the `qq set_monitoring_conf --vpn-enabled` command.
- To disable Remote Support, run the `qq set_monitoring_conf --vpn-disabled` command.
- To check the status of Remote Support, run the `qq set_monitoring_conf` command.

Connecting to Cloud-Based Monitoring and S3 by Using a Custom Proxy

This section explains how to connect to Cloud-Based Monitoring and S3 by using a custom proxy.

In Qumulo Core 2.6.4 (and higher), you can specify a custom proxy for both Cloud-Based Monitoring and S3 in different ways.

Configuring a Custom Proxy for Cloud-Based Monitoring and S3

The following examples show some common configurations for custom proxies by using the `qq set_monitoring_conf` command.

> ⚠ Important
> Remote VPN support isn't available when you connect to Cloud-Based Monitoring by using a custom proxy.

You can specify a custom proxy hostname and port *only* for Cloud-Based Monitoring. For example:

```
qq set_monitoring_conf
  --enabled
  --mq-proxy-host mq-proxy.example.com
  --mq-proxy-port 123
```

You can also specify a custom proxy hostname and port *only* for the S3 proxy (for the endpoint that Qumulo Core uses to store core dumps and logs). For example:

```
qq set_monitoring_conf
  --enabled
  --s3-proxy-host s3-proxy.example.com
  --s3-proxy-port 456
```

Finally, you can specify a custom proxy hostname and port for *both* Cloud-Based Monitoring and your S3 proxy. For example:

```
qq set_monitoring_conf
  --enabled
  --all-proxy-host mq-s3-proxy.example.com
  --all-proxy-port 789
```

Restoring the Default Values for Cloud-Based and Nexus Monitoring

This section explains how to set the default values for Cloud-Based Monitoring and Nexus Monitoring.

When you no longer connect to Cloud-Based Monitoring by using a custom proxy (page 347), you can use the `qq` CLI to restore the default values for Cloud-Based and Nexus Monitoring.

To Restore the Default Values for Cloud-Based and Nexus Monitoring by Using the qq CLI

1. Connect to your cluster by using SSH. For example:

   ```
   ssh admin@203.0.113.0
   ```

2. Log in to the Qumulo Core Web UI.by using the administrative account. For example:

   ```
   qq login -u admin -p HTEj0kGIOsNOAA0
   ```

3. To restore the default values for Cloud-Based and Nexus Monitoring, run the `qq set_monitoring_conf` command.

   In the following example, we:

- Enable Cloud-Based Monitoring (MQ)
- Enable Nexus monitoring
- Disable HTTPS for the S3 proxy
- Set the hostname and port for Cloud-Based Monitoring
- Disable the proxy for Cloud-Based Monitoring (by setting it to `0`)
- Set the monitoring polling interval to 60 seconds
- Set the hostname and port for the S3 proxy
- Specify the custom VPN for your organization, `example.qumulo.com`

```
qq set_monitoring_conf
  --enabled
  --nexus-enabled
  --s3-proxy-disable-https
  --mq-host api.missionq.qumulo.com
  --mq-port 443
  --mq-proxy-port 0
  --period 60
  --s3-proxy-host monitor.qumulo.com
  --s3-proxy-port 443
  --vpn-host example.qumulo.com
```

4. To confirm that Cloud-Based Monitoring is working correctly, log in to the Qumulo Core Web UI and then click **Support**.

   If your configuration is valid, the Qumulo Core Web UI shows the status **Enabled | Connected**.

How Audit Logging Works

This section explains the different formats of audit logs for file system operations that Qumulo Core generates whenever a connected client sends a request to a Qumulo cluster.

Qumulo Core creates a descriptive audit log message for every operation that a client attempts. Then, Qumulo Core sends the audit log messages to the remote syslog instance that the audit logging configuration specifies in compliance with RFC 5424 .

This section explains the differences between the levels of detail of audit logs in syslog CSV (page 351), syslog JSON (page 354), and CloudWatch JSON (page 356) formats. (In general, the syslog CSV and CloudWatch JSON formats contain audit logs contain an identical number of fields, some named differently, while the syslog JSON format has additional audit logging information.)

Note
Qumulo Core doesn't parse, analyze, index, or visualize the data. For more information, see the following articles on Qumulo Care:

- Sending Audit Logs for a Qumulo Cloud Cluster to Amazon CloudWatch
- Using Qumulo Core Audit Logging with Elasticsearch
- Using Splunk with Qumulo Core Audit Logging

Details Included in the Default syslog CSV Format

Note

- Because the user ID, path fields, and secondary path fields can contain characters that must be escaped (such as quotation marks and commas), you must enclose these fields in quotation marks.
- Qumulo Core system strips out the `\n` and `\r` newline characters from the user ID, file path, and secondary file path fields.
- Both syslog CSV and syslog JSON formats deduplicate repeated file reads. However, for metadata changes—such as modifications to an access-control list (ACL)—only the syslog CSV deduplicates repeated operations.
- Unlike the syslog JSON format (page 354), the syslog CSV format has only values (no keys) and the fields are empty when unused. The following table helps explain the fields and their possible values.

By default, Qumulo Core formats audit log messages in the syslog CSV format, prefaced by the date, time, and the name of the machine that issues the operation. The syslog CSV format includes the following fields in the following order within the log message body.

Field	Description	Possible Values
User IP address	The IP address of the user that performed the operation.	• IPv4 address • IPv6 address
User ID	The ID of the user who performed the operation. **Note** Qumulo Core specifies the Qumulo authentication ID if it can't resolve any of the other user ID types.	String in quotation marks: • Active Directory (AD) username • Qumulo local username • POSIX user ID (UID) • Windows security identifier (SID) • Qumulo authentication ID
Protocol	The protocol that the operation uses.	• `api` • `ftp` • `internal` • `nfsv3` • `nfsv4.1` • `s3` • `smb`

Operation Type	The operation that the user or the system attempts to perform.	• Connectivity operation (page 359), such as `ftp_login` • SMB-Specific operation (page 359), such as `smb_create_share` • Audit logging operation (page 359), such as `remote_syslog_startup` • File system operation (page 359), such as `fs_create` • REST API operation (page 360), such as `ad_join`

Operation Status	A success status or an error status.	• Success status message: `ok`
		• Error message:
		  ◦ Credential error message (page 365), such as `cred_invalid_sid_error`
		  ◦ File system operation error message (page 365), such as `fs_access_perm_not_owner_error`

File ID	The ID of the file on which the system performed an operation. **Note** For non-file entities, this field is empty.	Integer

File Path	The path to the file on which the system performed an operation. **Note** For files accessed by using a snapshot, the system prefixes the path with `/.snapshot`. (This is the same path prefix that the system uses to access snapshotted files through NFSv3 and SMB.)	String in quotation marks
Target File Path	The target path to the file on which the system performed a rename or move operation.	String in quotation marks

For example:

```
Jun 6 14:52:28 my-machine qumulo 203.0.113.0,"system",internal,remote_syslog_startu
p,ok,,"","",""
Jun 6 14:52:28 my-machine qumulo 203.0.113.0,"AD\alice",api,audit_modify_syslog_conf
ig,ok,,"","",""
Jun 6 14:52:40 my-machine qumulo 203.0.113.0,"AD\alice",api,rest_login,ok,,"","",""
Jun 6 14:53:22 my-machine qumulo 203.0.113.0,"AD\alice",api,fs_read_metadata,o
k,3,"/my_file","",""
Jun 6 14:53:22 my-machine qumulo 203.0.113.0,"AD\alice",api,fs_write_metadata,o
k,3,"/my_file","",""
Jun 6 14:53:22 my-machine qumulo 203.0.113.0,"AD\alice",api,fs_write_data,ok,3,"/m
y_file","",""
Jun 6 14:54:05 my-machine qumulo 203.0.113.0,"AD\alice",api,fs_rename,ok,3,"/my_fil
e","/another_file",""
Jun 6 14:55:24 my-machine qumulo 203.0.113.0,"AD\alice",api,begin_audit_modify_syslo
g_config,ok,,"","",""
Jun 6 14:55:24 my-machine qumulo 203.0.113.0,"system",internal,remote_syslog_shutdow
n,ok,,"","",""
```

Details Included in the syslog JSON Format

You can configure Qumulo Core to format audit log messages in the syslog JSON format. The fields in this format are similar to the fields that the syslog CSV format provides (page 351), with the following exceptions.

> **Note**
> The syslog JSON format isn't available in the Web UI.

Field	Description	Possible Values
`user_id` Object	In Qumulo Core 6.0.1 (and higher) the `user_id` object replaces the single user ID field in the syslog CSV format and contains the fields `sid`, `auth_id`, and `name`.	• `sid`: Security identifier • `auth_id`: Authentication ID • `name`: User role
`details` Object	• For most file system operations, the `details` object replaces the file path, secondary file path, and file ID fields in the syslog CSV format and contains the fields `path`, `target`, and `file_id`. • For `fs_write_*` and `fs_read_*` operations, the `details` object also includes the `offset` and `file_size` fields. • For operations that write metadata or change access-control lists (ACLs), the `details` object also includes the `after` and `before` objects that include fields for current and previous metadata.	• `details` object: ◦ `path`: File path ◦ `target`: Target file path ◦ `file_id`: File ID `fs_write_*` and `fs_read_*` operations only: ◦ `offset`: The starting position of the operation ◦ `file_size`: The size of the operation • `after` and `before` objects: ◦ `ctime`: Changed timestamp ◦ `mtime`: Modified timestamp ◦ `owner` ■ `sid`: Security identifier ■ `auth_id`: Authentication ID

For example:

```
Jun 6 14:52:28 my-machine qumulo {"user_id": {"auth_id": "1", "sid": "S-1-5-21-98765
43210-5678901234-1234567890-1", "name": "system"}, "user_ip": "203.0.113.0", "protoc
ol": "internal", "operation": "remote_syslog_startup", "status": "ok", "details":
{}}
Jun 6 14:52:28 my-machine qumulo {"user_id": {"sid": "S-1-5-21-9876543210-567890123
4-1234567890-500", "auth_id": "500", "name": "AD\alice"}, "user_ip": "203.0.113.0",
"protocol": "api", "operation": "audit_modify_syslog_config", "status": "ok", "detai
ls": {"second_extra_name": "", "extra_name": ""}}
Jun 6 14:52:40 my-machine qumulo {"user_id": {"auth_id": "500", "name": "AD\alice",
"sid": "S-1-5-21-9876543210-5678901234-1234567890-500"}, "user_ip": "203.0.113.0",
"protocol": "api", "operation": "rest_login", "status": "ok", "details": {"second_ex
tra_name": "", "extra_name": ""}}
Jun 6 14:53:22 my-machine qumulo {"user_id": {"sid": "S-1-5-21-9876543210-567890123
4-1234567890-500", "name": "AD\alice", "auth_id": "500"}, "user_ip": "203.0.113.0",
"protocol": "api", "operation": "fs_read_metadata", "status": "ok", "details": {"pat
h": "/my_file", "file_id": "4"}}
Jun 6 14:53:22 my-machine qumulo {"user_id": {"name": "AD\alice", "sid": "S-1-5-21-9
876543210-5678901234-1234567890-500", "auth_id": "500"}, "user_ip": "203.0.113.0",
"protocol": "api", "operation": "fs_write_metadata", "status": "ok", "details": {"fi
le_id": "4", "after": {"ctime": "2024-06-11T14:55:58.187394089Z", "mtime": "2024-0
6-11T14:55:58.187394089Z", "owner": {"sid": "S-1-5-21-9876543210-5678901234-12345678
90-500", "auth_id": "500"}}, "path": "/my_file", "before": {"ctime": "2024-06-11T1
4:55:43.616292461Z", "mtime": "2024-06-11T14:55:43.616292461Z", "owner": {"sid":
"S-1-5-21-9876543210-5678901234-1234567890-500", "auth_id": "500"}}}}
Jun 6 14:53:22 my-machine qumulo {"user_id": {"auth_id": "500", "sid": "S-1-5-21-987
6543210-5678901234-1234567890-500", "name": "AD\alice"}, "user_ip": "203.0.113.0",
"protocol": "api", "operation": "fs_write_data", "status": "ok", "details": {"pat
h": "/my_file", "size": 261456, "file_id": "4", "offset": 0, "file_size": 261456}}
Jun 6 14:54:05 my-machine qumulo {"user_id": {"name": "AD\alice", "auth_id": "500",
"sid": "S-1-5-21-9876543210-5678901234-1234567890-500"}, "user_ip": "203.0.113.0",
"protocol": "api", "operation": "fs_rename", "status": "fs_entry_exists_error", "det
ails": {"path": "/my_file", "target": "/another_file", "file_id": "4"}}
Jun 6 14:55:24 my-machine qumulo {"user_id": {"sid": "S-1-5-21-9876543210-567890123
4-1234567890-500", "auth_id": "500", "name": "AD\alice"}, "user_ip": "203.0.113.0",
"protocol": "api", "operation": "begin_audit_modify_syslog_config", "status": "ok",
"details": {"second_extra_name": "", "extra_name": ""}}
Jun 6 14:55:24 my-machine qumulo {"user_id": {"auth_id": "1", "sid": "S-1-5-21-98765
43210-5678901234-1234567890-1", "name": "system"}, "user_ip": "203.0.113.0", "protoc
ol": "internal", "operation": "remote_syslog_shutdown", "status": "ok", "details":
{}}
```


Details Included in the Amazon CloudWatch JSON Format

You can configure Qumulo Core to format audit log messages in the Amazon CloudWatch JSON format.

> ☑ Tip
> To download the audit log from the CloudWatch console, on the left navigation panel click **Logs** > **Log groups**, click a log group, and then on the **Log events** page click **Actions** > **Copy search results (ASCII)**.

Rather than preface each line of CSV or JSON with the date and time, CloudWatch creates an ASCII table, which contains Unix timestamps in its first column. The second column contains the fields that are similar to the fields that both the syslog CSV format (page 351) and the syslog JSON format (page 354) provide, with the following exceptions.

- The `result` field replaces the Operation Status or `status` field.
- The `object_id` field replaces the File ID `file_id` field.
- The `path_1` field replaces the File Path or `path` field.
- The `path_2` field replaces the Target File Path or `target` field.

For example:

```
------------------------------------------------------------------------------
------------------------------------------------------------------------------
------------------------------
|   timestamp
|
      messag
e
 |
|---------------
|------------------------------------------------------------------------------
------------------------------------------------------------------------------
-------------|
| 1717679548000 | {"ip_address": "203.0.113.0", "user": "system", "protocol": "inter
nal", "operation": "remote_syslog_startup", "result": "ok", "object_id": "", "pat
h_1": "", "path_2": ""}         |
| 1717679548000 | {"ip_address": "203.0.113.0", "user": "AD\alice", "protocol": "ap
i", "operation": "audit_modify_syslog_config", "result": "ok", "object_id": "", "pat
h_1": "", "path_2": ""}       |
| 1717679560000 | {"ip_address": "203.0.113.0", "user": "AD\alice", "protocol": "ap
i", "operation": "rest_login", "result": "ok", "object_id": "", "path_1": "", "pat
h_2": ""}                       |
| 1717679602000 | {"ip_address": "203.0.113.0", "user": "AD\alice", "protocol": "ap
i", "operation": "fs_read_metadata", "result": "ok", "object_id": "3", "path_1": "/m
y_file", "path_2": ""}        |
| 1717679602000 | {"ip_address": "203.0.113.0", "user": "AD\alice", "protocol": "ap
i", "operation": "fs_write_metadata", "result": "ok", "object_id": "3", "path_1":
"/my_file", "path_2": ""}       |
| 1717679602000 | {"ip_address": "203.0.113.0", "user": "AD\alice", "protocol": "ap
i", "operation": "fs_write_data", "result": "ok", "object_id": "3", "path_1": "/my_f
ile", "path_2": ""}           |
| 1717679645000 | {"ip_address": "203.0.113.0", "user": "AD\alice", "protocol": "ap
i", "operation": "fs_rename", "result": "ok", "object_id": "3", "path_1": "/my_fil
e", "path_2": "/another_file"}  |
| 1717679724000 | {"ip_address": "203.0.113.0", "user": "AD\alice", "protocol": "ap
i", "operation": "begin_audit_modify_syslog_config", "result": "ok", "object_id":
"", "path_1": "", "path_2": ""} |
| 1717679724000 | {"ip_address": "203.0.113.0", "user": "system", "protocol": "inter
nal", "operation": "remote_syslog_shutdown", "result": "ok", "object_id": "", "pat
h_1": "", "path_2": ""}        |
------------------------------------------------------------------------------
------------------------------------------------------------------------------
------------------------------
```

Operation Names in Audit Logging

This section lists the operation names in Qumulo Core audit logging.

Connectivity Operations

- `ftp_login`
- `nfs_mount`
- `rest_login`
- `share_connect`
- `smb_close_handle`
- `smb_close_session`
- `smb_login`

SMB-Specific Operations

- `smb_create_share`
- `smb_modify_share`
- `smb_delete_share`
- `smb2_notify_req`

Audit Logging Operations

- `remote_syslog_shutdown`
- `remote_syslog_startup`

File System Operations

- `fs_copy_chunk`
- `fs_create_directory`
- `fs_create_directories`
- `fs_create_file`
- `fs_create_hard_link`
- `fs_create_symlink`
- `fs_create_stream`
- `fs_create`

> **Note**
> This operation type applies to any but the previous file types.

- `fs_delete`
- `fs_fsstat`
- `fs_list_directory`
- `fs_open`
- `fs_read_data`
- `fs_read_link`
- `fs_read_metadata`
- `fs_read_user_metadata`
- `fs_rename`
- `fs_set_quota`
- `fs_tree_delete`
- `fs_write_data`
- `fs_write_metadata`
- `fs_write_file_lock`
- `fs_write_user_metadata`

REST API Operations

- `ad_cancel_operation`
- `ad_dismiss_error`
- `ad_join`
- `ad_kerberos_renew_ticket`
- `ad_leave`
- `ad_modify_settings`
- `ad_reconfigure`
- `audit_modify_cloudwatch_config`
- `audit_modify_syslog_config`

- `auth_add_group`
- `auth_add_member_to_group`
- `auth_assign_role`
- `auth_create_access_token`
- `auth_create_role`
- `auth_create_user`
- `auth_delete_access_token`
- `auth_delete_group`
- `auth_delete_identity_attributes`
- `auth_delete_role`
- `auth_delete_user`
- `auth_get_access_token`
- `auth_get_access_tokens`
- `auth_modify_access_token`
- `auth_modify_group`
- `auth_modify_identity_attributes`
- `auth_modify_role`
- `auth_modify_user`
- `auth_modify_user_password`
- `auth_remove_member_from_group`
- `auth_set_user_defined_mappings`
- `auth_unassign_role`
- `cluster_add_nodes`
- `cluster_calculate_node_add_capacity`
- `cluster_generate_vpn_private_key`
- `cluster_initiate_node_replacement_plan`
- `cluster_install_vpn_keys`
- `cluster_locate_disk_slot`
- `cluster_modify_config`

- `cluster_modify_monitoring_config`
- `cluster_modify_nodes`
- `cluster_modify_nodes_dry_run`
- `cluster_modify_upgrade_config`
- `cluster_nodes_resolve_indeterminate_modification`
- `dns_create`
- `dns_delete`
- `dns_modify`
- `dns_set_lookup_overrides`
- `encryption_create_key`
- `encryption_rotate_keys`
- `encryption_update_config`
- `fs_create_public_key`
- `fs_delete_public_key`
- `fs_modify_atime_settings`
- `fs_modify_notify_settings`
- `fs_modify_permissions_settings`
- `fs_modify_public_key`
- `fs_release_nlm_locks_by_client`
- `fs_release_nlm_locks_by_file`
- `fs_replace_public_key`
- `fs_sample`
- `ftp_modify_settings`
- `identity_create`
- `identity_delete`
- `identity_modify`
- `kerberos_delete_keytab`
- `kerberos_modify_settings`
- `kerberos_set_keytab`

- `ldap_modify_settings`
- `network_create_config`
- `network_delete_config`
- `network_modify_config`
- `network_modify_interface`
- `nfs_create_export`
- `nfs_delete_export`
- `nfs_delete_settings`
- `nfs_modify_export`
- `nfs_modify_settings`
- `node_halt`
- `node_restart`
- `node_set_identify_light`
- `reboot_pause`
- `reboot_resume`
- `reboot_start`
- `replication_abort`
- `replication_abort_object_relationship`
- `replication_authorize_target_relationship`
- `replication_create_object_relationship`
- `replication_create_source_relationship`
- `replication_delete_object_relationship`
- `replication_delete_source_relationship`
- `replication_delete_target_relationship`
- `replication_dismiss_source_relationship_error`
- `replication_dismiss_target_relationship_error`
- `replication_make_target_writable`
- `replication_modify_source_relationship`
- `replication_reconnect_target_relationship`

- `replication_release_queued_snapshot`
- `replication_replicate_object_relationship`
- `replication_reverse_target_relationship`
- `replication_start`
- `replication_target_relationship_lock`
- `s3_abort_upload`
- `s3_add_bucket`
- `s3_bucket_policy_explain_access`
- `s3_create_key`
- `s3_delete_bucket`
- `s3_delete_key`
- `s3_get_bucket_policy`
- `s3_modify_bucket`
- `s3_modify_settings`
- `s3_put_bucket_policy`
- `saml_modify_settings`
- `session_change_password`
- `shutdown_halt`
- `smb_delete_settings`
- `smb_modify_settings`
- `snapshot_create_policy`
- `snapshot_create_snapshot`
- `snapshot_delete_policy`
- `snapshot_delete_snapshot`
- `snapshot_lock_snapshot`
- `snapshot_modify_policy`
- `snapshot_modify_snapshot`
- `snapshot_unlock_snapshot`
- `snapshot_unlock_snapshot_challege`

- `ssl_delete_ca_certificate`
- `ssl_modify_ca_certificate`
- `ssl_modify_certificate`
- `tenant_create`
- `tenant_delete`
- `tenant_modify`
- `time_modify_config`
- `tree_delete_cancel`
- `tree_delete_start`
- `web_ui_modify_settings`

Error Status Messages

This section lists the error status messages in Qumulo Core audit logging.

Credential Error Messages

- `cred_error`
- `cred_identity_not_supported_error`
- `cred_invalid_local_user_error`
- `cred_invalid_sid_error`
- `cred_remote_resource_unavailable_error`

File System Operation Error Messages

- `fs_access_denied_error`
- `fs_access_error`
- `fs_access_perm_not_owner_error`
- `fs_already_read_only_error`
- `fs_default_stream_of_directory_error`
- `fs_directory_cycle_error`
- `fs_directory_error`
- `fs_directory_hash_exhaustion_error`

- `fs_directory_is_symlink_error`
- `fs_directory_not_empty_error`
- `fs_encountered_portal_error`
- `fs_entry_exists_error`
- `fs_entry_hardlink_ambiguity_error`
- `fs_entry_not_in_subtree_error`
- `fs_exceeded_maximum_stream_count_error`
- `fs_file_dirent_id_stale_error`
- `fs_file_is_named_stream_error`
- `fs_file_is_symlink_error`
- `fs_file_not_covered_by_snapshot_error`
- `fs_file_not_empty_error`
- `fs_file_path_invalid_error`
- `fs_file_size_error`
- `fs_file_too_many_links_error`
- `fs_invalid_acl_error`
- `fs_invalid_attrs_error`
- `fs_invalid_file_type_error`
- `fs_invalid_name_error`
- `fs_invalid_offset_error`
- `fs_invalid_size_error`
- `fs_name_size_error`
- `fs_no_such_entry_error`
- `fs_no_such_file_version_error`
- `fs_no_such_inode_error`
- `fs_no_such_path_error`
- `fs_not_a_directory_error`
- `fs_not_a_file_error`
- `fs_not_a_symlink_error`

- `fs_opened_file_stale_error`
- `fs_path_size_error`
- `fs_permissions_mode_operation_unsupported_error`
- `fs_portal_create_orphanage_error`
- `fs_portal_disconnected_error`
- `fs_portal_ended_error`
- `fs_portal_error`
- `fs_portal_eviction_error`
- `fs_portal_inconsistent_data_error`
- `fs_portal_incorrect_state_error`
- `fs_portal_invalid_lease_break_range_error`
- `fs_portal_not_covered_by_snapshot_error`
- `fs_portal_not_found_error`
- `fs_portal_out_of_order_lease_break_error`
- `fs_quota_limit_already_set_error`
- `fs_quota_limit_does_not_exist_error`
- `fs_quota_no_space_error`
- `fs_read_only_error`
- `fs_revision_mismatch_error`
- `fs_s3_versioning_full_error`
- `fs_same_inode_error`
- `fs_sharing_violation_error`
- `fs_smb_share_root_deleted_error`
- `fs_stream_is_default_error`
- `fs_symlink_loop_error`
- `fs_tag_mismatch_error`
- `fs_too_many_aces_error`
- `fs_user_metadata_continuation_token_type_mismatch`
- `fs_user_metadata_empty_insertion_error`

- `fs_user_metadata_hash_exhaustion_error`
- `fs_user_metadata_key_already_exists`
- `fs_user_metadata_key_not_found`
- `fs_user_metadata_too_large`
- `fs_watcher_missed_events_error`

Qumulo OpenMetrics API Specification

This section lists the names, types, labels, and descriptions for the metrics that Qumulo Core 5.3.0 (and 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 Core 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 71) 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.
`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.

Metric Type	Description
`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 378).

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	Supported from Qumulo Core Version	Description
`qumulo`	`info` (page 370)	• `max_drive_failures` • `max_node_failures` • `name` • `platform` • `service_model` • `uuid` • `version` **Tip** Don't confuse this label for the *Qumulo Core version* with the identically named label for the *kernel version* for the `qumulo_kernel` metric.	5.3.0	Qumulo Core information, including the cluster name, cluster UUID, and the current Qumulo Core version.
`qumulo_node`	`info` (page 370)	• `node_id` (page 380) • `node_model`	6.0.2	Information about the nodes in the cluster, including the node ID and the node model
`qumulo_ad_netlogon_request_errors`	`counter` (page 369)	• `domain_url` (page 379) • `server_url` (page 380)	5.3.0	The total number of Active Directory (AD) `NETLOGON` requests that resulted in an error
`qumulo_ad_netlogon_request_latency_seconds`	`histogram` (page 370)	• `domain_url` (page 379) • `server_url` (page 380)	5.3.0	The total latency for AD `NETLOGON` requests

Metric Name	Metric Type	Labels	Supported from Qumulo Core Version	Description
`qumulo_ad_netlogon_requests`	`counter` (page 369)	• `domain_url` (page 379) • `server_url` (page 380)	5.3.0	The total number of completed AD `NETLOGON` operations
`qumulo_cpu_crit_temperature_celsius`	`gauge` (page 369)	• `cpu` (page 378) • `node_id` (page 380)	7.2.0	The critical temperature threshold for each physical CPU
`qumulo_cpu_max_temperature_celsius`	`gauge` (page 369)	• `cpu` (page 378) • `node_id` (page 380)	5.3.1	The maximum temperature threshold for each physical CPU
`qumulo_cpu_temperature_celsius`	`gauge` (page 369)	• `cpu` (page 378) • `node_id` (page 380)	5.3.0	The temperature for each physical CPU, in degrees Celsius
`qumulo_disk_endurance_percent`	`gauge` (page 369)	• `disk_type` (page 379) • `drive_bay` (page 379) • `node_id` (page 380)	5.3.1	The remaining disk endurance value for each disk in the cluster, ranging `100` (no disk wear) to `0` (disk is worn fully)
`qumulo_disk_transport_errors`	`counter` (page 369)	• `disk_type` (page 379) • `drive_bay` (page 379) • `node_id` (page 380)	5.3.2	The total number of communication errors between the specified drive and its host.

Metric Name	Metric Type	Labels	Supported from Qumulo Core Version	Description
`qumulo_disk_uncorrectable_media_errors`	`counter` (page 369)	• `disk_type` (page 379) • `drive_bay` (page 379) • `node_id` (page 380)	5.3.2	The total number of uncorrectable errors on the specified drive's physical media.
`qumulo_disk_is_unhealthy`	`gauge` (page 369)	• `disk_type` (page 379) • `drive_bay` (page 379) • `node_id` (page 380)	5.3.0	The health of each disk in the cluster, ranging from `0` (the disk is healthy) to `1` (the disk is unhealthy)
`qumulo_disk_operation_latency_seconds`	`histogram` (page 370)	• `disk_type` (page 379) • `drive_bay` (page 379) • `io_type` (page 379) • `node_id` (page 380)	5.3.0	The total latency for disk I/O operations
`qumulo_fan_speed_rpm`	`gauge` (page 369)	• `fan` (page 379) • `node_id` (page 380)	5.3.0	The fan speed, in RPM
`qumulo_fs_capacity_bytes`	`gauge` (page 369)	—	5.3.0	The total cluster space, in bytes
`qumulo_fs_directory_tree_entries`	`gauge` (page 369)	• `entry_type` (page 379) • `path` (page 380)	5.3.0	The number of file system objects on the cluster, sorted by object type
`qumulo_fs_directory_used_bytes`	`gauge` (page 369)	• `path` (page 380) • `usage_type` (page 380)	5.3.0	The amount of space that object types use, in bytes

Metric Name	Metric Type	Labels	Supported from Qumulo Core Version	Description
`qumulo_fs_free_bytes`	`gauge` (page 369)	—	5.3.0	The free space on the cluster, in bytes
`qumulo_fs_snapshots`	`gauge` (page 369)	—	5.3.0	The number of snapshots on the cluster
`qumulo_ldap_lookup_request_errors`	`counter` (page 369)	• `domain_url` (page 379) • `server_url` (page 380)	5.3.0	The total number of LDAP requests that resulted in an error
`qumulo_ldap_lookup_request_latency_seconds`	`histogram` (page 370)	• `domain_url` (page 379) • `server_url` (page 380)	5.3.0	The total latency of LDAP requests
`qumulo_ldap_lookup_requests`	`counter` (page 369)	• `domain_url` (page 379) • `server_url` (page 380)	5.3.0	The total number of completed LDAP requests
`qumulo_ldap_operation_errors`	`counter` (page 369)	`domain_url` (page 379)	5.3.0	The total number of LDAP operations that resulted in an error
`qumulo_ldap_operation_latency_seconds`	`histogram` (page 370)	`domain_url` (page 379)	5.3.0	The total latency for LDAP operations
`qumulo_ldap_operations`	`counter` (page 369)	`domain_url` (page 379)	5.3.0	The total number of completed LDAP operations

Metric Name	Metric Type	Labels	Supported from Qumulo Core Version	Description
`qumulo_memory_correctable_ecc_errors`	`counter` (page 369)	`node_id` (page 380)	5.3.0	The total number of memory errors that Qumulo Core corrected automatically
`qumulo_network_interface_is_down`	`gauge` (page 369)	• `bond` (page 378) • `interface` (page 379) • `role` (page 380) • `node_id` (page 380)	5.3.0	The interface status, `0` (interface is up) or `1` (interface is down)
`qumulo_network_interface_link_speed_bits_per_second`	`gauge` (page 369)	• `bond` (page 378) • `interface` (page 379) • `role` (page 380) • `node_id` (page 380)	5.3.0	The negotiated link speed for the specified interface
`qumulo_network_interface_receive_errors`	`counter` (page 369)	• `bond` (page 378) • `interface` (page 379) • `role` (page 380) • `node_id` (page 380)	5.3.0	The total number of receive errors on the specified interface
`qumulo_network_interface_received_bytes`	`counter` (page 369)	• `bond` (page 378) • `interface` (page 379) • `role` (page 380) • `node_id` (page 380)	5.3.0	The total bytes received on the specified interface

Metric Name	Metric Type	Labels	Supported from Qumulo Core Version	Description
`qumulo_network_interface_received_packets`	`counter` (page 369)	• `bond` (page 378) • `interface` (page 379) • `role` (page 380) • `node_id` (page 380)	5.3.0	The total number of packets received on the specified interface
`qumulo_network_interface_transmit_errors`	`counter` (page 369)	• `bond` (page 378) • `interface` (page 379) • `role` (page 380) • `node_id` (page 380)	5.3.0	The total number of transmission errors on the specified interface
`qumulo_network_interface_transmitted_bytes`	`counter` (page 369)	• `bond` (page 378) • `interface` (page 379) • `role` (page 380) • `node_id` (page 380)	5.3.0	The total number of bytes transmitted on the specified interface
`qumulo_network_interface_transmitted_packets`	`counter` (page 369)	• `bond` (page 378) • `interface` (page 379) • `role` (page 380) • `node_id` (page 380)	5.3.0	The total number of packets transmitted on the specified interface
`qumulo_power_supply_is_unhealthy`	`gauge` (page 369)	• `location` (page 379) • `node_id` (page 380)	5.3.0	PSU health, `0` (healthy) or `1` (unplugged, removed, or unresponsive)
`qumulo_protocol_client_connections`	`counter` (page 369)	`protocol` (page 380)	5.3.0	The total number of clients that have connected to the specified protocol

Metric Name	Metric Type	Labels	Supported from Qumulo Core Version	Description
`qumulo_protocol_client_disconnections`	`counter` (page 369)	`protocol` (page 380)	5.3.0	The total number of clients that have disconnected from the specified protocol
`qumulo_protocol_operation_bytes`	`counter` (page 369)	• `data_type` (page 378) • `io_type` (page 379) • `op_name` (page 380) • `protocol` (page 380)	5.3.0	The total bytes that protocol operations have transferred
`qumulo_protocol_operation_latency_seconds`	`histogram` (page 370)	• `data_type` (page 378) • `io_type` (page 379) • `op_name` (page 380) • `protocol` (page 380)	5.3.0	The total latency for protocol operations
`qumulo_protocol_operations`	`counter` (page 369)	• `data_type` (page 378) • `io_type` (page 379) • `op_name` (page 380) • `protocol` (page 380)	5.3.0	The total number of completed protocol operations
`qumulo_quorum_node_is_offline`	`gauge` (page 369)	`node_id` (page 380)	5.3.0	The online status for each node in the cluster, `0` (node online) or `1` (node offline)

Metric Name	Metric Type	Labels	Supported from Qumulo Core Version	Description
`qumulo_time_is_not_synchronizing`	`gauge` (page 369)	`node_id` (page 380)	5.3.0	The time synchronization status for each node in the cluster, `0` (time is synchronized) or `1` (time isn't synchronized)

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
`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

Label Name	Possible Values	Description
`disk_type`	• `hdd` : Hard Disk Drive • `ssd` : Solid-State Drive	The underlying storage type
`domain_url`	An Active Directory domain (for example, `my-domain.com`) or an LDAP bind URI (for example, `ldap://my-server.my-domain.com`)	The URL of the domain
`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
`io_type`	• `composite` • `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.

Label Name	Possible Values	Description
`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, REST, S3, replication, or FTP	The recorded operation
`path`	Slash (`/`)	The path to a directory in the file system
`protocol`	• `nfs` : NFSv3 or NFSv4.1 • `smb` : SMBv2 or SMBv3 • `rest` • `s3` • `replication` • `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.
`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