Azure Native Qumulo Administrator Guide

Copyright © 2024 Qumulo, Inc.

Table of Contents

Getting Started

How Azure Native Qumulo Works 4
Virtual Networking Prerequisites 10
Deploying an Instance 13
Connecting to Microsoft Entra Domain Services 16
Creating and Managing Directory Quotas 18
Performance Characteristics and Default Limits 19
Supported Configurations and Known Limits 20
Replication Version Requirements 23

Authentication

Configuring SAML Single Sign-On (SSO) 26
Configuring Search Trusted Domains 35

Authorization

Managing Cross-Protocol Permissions 37

External Services

Using Access Tokens 39
Connecting a Kubernetes Cluster 49

Network Configuration

Required Networking Ports 57

Web UI

Setting the Web UI Login Banner 59
Setting the Web UI Inactivity Timeout 60

qq CLI

Enabling Autocomplete for the qq CLI 61

Metadata

Managing User-Defined Metadata 64

Snapshots

How Snapshots Work 67
Managing Snapshots 70
Locking and Unlocking Snapshots 75
Recovering Files by Using Snapshots 78

Encryption and Data Security

Generating and Storing ECDSA Keys 80
Managing Security Keys 86

Data Replication

Creating and Managing a Continuous Replication Relationship 91
Shift-To Amazon S3 96
Shift-From Amazon S3 107

File System Changes

How File System Change Notifications Work 119
Watching for Changes with SMB2 CHANGE_NOTIFY 123
Watching for Changes with REST 127

NFS

Creating and Managing an NFS Export 133
Enabling and Using NFSv4.1 135
Managing File Access Permissions with ACLs 142
Host Access Rules for NFS Exports 151

NFSv4.1 with Kerberos

How NFSv4.1 Works with Kerberos 157
Prerequisites for Joining to Active Directory 159
Configuring Active Directory 161
Performing Additional Cluster Configuration 165
Using Kerberos Permissions 168
Configuring a Linux Client 174
Configuring Cross-Domain Active Directory Trusts 184
Troubleshooting NFSv4.1 with Kerberos 186

SMB

Creating and Managing an SMB Share 189

S3 API

Configuring and Using the S3 API 191
Creating and Managing S3 Access Keys 196
Creating and Managing S3 Buckets 204
Managing Access to S3 Buckets 213
Managing Access Policies for S3 Buckets 219
Managing Multipart S3 Uploads 227
Supported Functionality and Limits 233

Monitoring and Metrics

OpenMetrics API Specification 240

Getting Started

How Azure Native Qumulo Works

This section explains the main functionality of Azure Native Qumulo (ANQ) and the differences between ANQ v2 and ANQ v1, provides a feature comparison between ANQ and Qumulo on other platforms and ANQ's known limitations and compliance posture, gives an overview of deploying the service in Azure, and lists the supported Azure Regions for the service.

For detailed instructions, see Deploying and Viewing Information about Your Azure Native Qumulo Instance (page 13).

What is Azure Native Qumulo?

ANQ is a fully managed service that provisions a Qumulo file system and creates a resource (for managing the file system) under your Azure subscription. ANQ provides the same multi-protocol support, interfaces, and functionality as Qumulo on premises.

ANQ makes it possible to configure file protocols, quotas, replication, and other features regardless of underlying infrastructure or storage and without tracking resource quotas or costs. The service receives the latest updates and features continuously and, if any issues occur, replaces compute and storage resources automatically.

Names and Versions

In this guide, we refer to the features and functionality of Qumulo Core as *Azure Native Qumulo (ANQ)* or *the service.*

Following ANQ's initial launch, we configured the Qumulo file system in Azure to have significant flexibility and performance improvements. This configuration appears in the Azure Portal as **ANQ v2**. For more information, see Feature Comparison with Qumulo on Other Platforms (page 4).

> **Note**
>
> - For a limited time, you can select the original **ANQ v1** configuration in the Azure Portal (after this time, **ANQ v2** remains the only available configuration).
> - For help with expanding ANQ v1 capacity, email Azure Native Qumulo Support.

Feature Comparison with Qumulo on Other Platforms

The following table compares the features of ANQ with those of Qumulo on other platforms.

Note

Because ANQ is a fully managed service, direct access to hosts with SSH is unavailable. To configure the service, you can use:

- `qq`CLI—from a remote machine
- Qumulo Core Web UI—by using any of the service's IP addresses

Feature	ANQ v2	ANQ v1	Qumulo on AWS as an AMI	Qumulo on Premises
Automatic deployment	✓	✓		
Automatic infrastructure replacement	✓	✓	✓	
Automatic updates	✓	✓		
Availability in Cloud Marketplace	✓	✓	✓	
Customer support	✓	✓	✓	✓
Integration with Azure Portal	✓			
Payment for pre-provisioned file system capacity		✓	✓	✓
Payment for used storage space only	✓			
Performance scales elastically at any capacity	✓			
Performance scales with provisioned capacity		✓	✓	✓
Qumulo Core features	✓	✓	✓	✓
Simple and fast deployment under 15 Minutes	✓			

Known Limitations

- **IPv6 Addresses:** Currently, Azure Networking features don't support IPv6 addresses.

- **Initial Authentication over SMB:** When you deploy the service initially, all users can use the SMB protocol. However, the `admin` user can authenticate over all protocols except over SMB.

  To allow the `admin` user to authenticate over the SMB protocol, change the `admin` user's password.

- **File Systems Larger than 1 PB:** This limitation applies only to ANQ v1.

  To deploy ANQ v1 instances larger than 1 PB by using the Azure Portal, email Azure Native Qumulo Support.

Qumulo Compliance Posture

For information about Qumulo's third-party attestations, including FIPS 140-2 Level 1, GDPR, HIPAA, and SOC 2 Type II, see Qumulo Trust Center.

Deploying Azure Native Qumulo

This section outlines the process of configuring and deploying ANQ. For detailed instructions, see Deploying and Viewing Information about Your Azure Native Qumulo Instance (page 13).

1. You specify the following configuration.

   - **Regional Settings:** The availability zone and region. For more information, see Supported Azure Regions (page 7)
   - **Networking Settings:** The virtual network in the same region. For more information, see Virtual Networking Prerequisites (page 10)
   - **Usable Capacity:** For ANQ v1 instances, the available file system capacity (in TB).

     > **Note**
     > Because ANQ v2 instances use the Azure Blob Store capacity limit (in EB), it isn't necessary to configure usable capacity in advance.

2. When Qumulo creates your ANQ instance, it deploys and configures the following Azure resources:

   - **Managed Resource Group:** This group contains the networking resources that the service deploys.

     When you create your service instance, you can specify an existing resource group or create a new one.

   - **Delegated Subnet:** The delegated subnet that the service uses to provision endpoints for your virtual network.

     When you create your service instance, you can specify an existing delegated subnet or create a new one..

- **Qumulo Service Resource:** The Azure resource that represents one instance of the service.

  You can use this resource to manage and view the service configuration.

- **Marketplace SaaS Resource:** The Qumulo Marketplace SaaS resource that you select.

  Azure uses this resource for billing purposes.

> ☑ Tip
> To automate the creation of ANQ instances for long-term use cases and for short-term components of automated storage workflows, use Azure Resource Manager.

Supported Azure Regions

The following table lists regions in US, Canada, Europe, and the UK that ANQ supports.

Geographical Location	Azure Region	ANQ v2	ANQ v1
US (Arizona)	West US 3	✓	✓
US (California)	West US		✓
US (Illinois)	North Central US		✓
US (Iowa)	Central US	✓	✓
US (Texas)	South Central US	✓	✓
US (Virginia)	East US	✓	✓
US (Virginia)	East US 2	✓	✓
US (Washington)	West US 2	✓	✓
Canada (Toronto)	Canada Central	✓	✓
Europe (Frankfurt)	Germany West Central	✓	✓
Europe (Gavle)	Sweden Central	✓	✓
Europe (Ireland)	North Europe	✓	✓
Europe (Netherlands)	West Europe	✓	✓

Geographical Location	Azure Region	ANQ v2	ANQ v1
Europe (Oslo)	Norway East	✓	✓
Europe (Paris)	France Central	✓	✓
Europe (Zurich)	Switzerland North	✓	✓
UK (London)	UK South	✓	✓
Australia (New South Wales)	Australia East	✓	
Brazil (São Paulo State)	Brazil South	✓	
India (Pune)	Central India	✓	
Japan (Tokyo, Saitama)	Japan East	✓	
Korea (Seoul)	Korea Central	✓	
UAE (Dubai)	UAE North	✓	

Usage Metering and Billing for Azure Native Qumulo

Once an hour, ANQ reports a metering event to Azure Marketplace for each deployed instance.

> **Note**
> ANQ v1 and ANQ v2 use different dimensions for metering.

ANQ v1 Metering for Total Available Capacity

Qumulo provisions ANQ v1 instances with a total available capacity (to increase this capacity, you can contact the Qumulo Care team).

Qumulo performs metering on a single dimension—the total available capacity that your instance reports.

ANQ v2 Metering for Used Capacity and Throughput

Qumulo doesn't provision ANQ v2 instances with a total available capacity. Instead, it performs metering on two dimensions:

- Used capacity
- Used throughput

Note

Because the throughput for an instance can vary significantly within a given hour, Qumulo samples the used throughput *each minute*. It rounds the computed throughput value to 1 GBps and then multiplies it by the used capacity during the given minute. For more information about pay-as-you-go price estimates, see the Pricing and Performance Calculator.

Virtual Networking Prerequisites for Azure Native Qumulo

This section lists the prerequisites for Azure Native Qumulo (ANQ), describes the components of virtual networking for the service, explains how to configure them, and provides virtual networking best practices.

How Qumulo Manages Virtual Networking for Azure Native Qumulo

When you create an ANQ instance, Qumulo manages the underlying storage and compute resources for the service. These resources reside within Qumulo's Azure tenant.

The ANQ instance connects to your Azure subscription by using *VNet injection*, an Azure-specific networking technology that establishes an automatic, direct connection between your resources and service resources without complicated manual configuration or VNet peering.

VNet injection lets you:

- Apply routing and security policies to your ANQ service endpoints by using the Azure Portal, CLI, and API.
- Create endpoints that allow access to ANQ by inserting special network interfaces into your subnet. This process binds these network interfaces directly to the compute resources of your ANQ instance.

When you create your ANQ instance, the Azure Portal guides you to create an appropriate subnet configuration in your virtual network. Then, VNet injection delegates privileges to Qumulo by communicating with the subnet.

Prerequisites for Configuring Virtual Networking

This section explains the prerequisites for configuring virtual networking for ANQ, such as creating roles, configuring dedicated subnets, and load-balancing endpoints.

Creating Owner and Contributor Roles

The service requires an owner or contributor role with access to your Azure subscription.

> ⚠ Important
> A custom role must have write permissions to the resource groups in which you create your delegated subnet and service.

Creating A Dedicated Subnet

The service requires a dedicated subnet.

> **Note**
>
> - Your subnet address range should be at least `/24` (it should contain at least 256 IP addresses, including 251 free IP addresses and 5 IP addresses reserved for Azure.)
> - Your subnet must be in the same region as the ANQ file system.

To Create a Dedicated Subnet Automatically

We recommend using the Azure Portal's automatic subnet creation and configuration functionality.

1. Create your ANQ instance. For detailed instructions, see Deploying and Viewing Information about Your Azure Native Qumulo Instance (page 13).
2. In the Azure Portal, click **Manage Subnet Configuration**.
3. When prompted, enter an IP address range for your subnet.

The Azure Portal configures your subnet and the required delegation for VNet injection automatically.

To Create a Dedicated Subnet Manually

To apply a specific subnet configuration, you can first create a subnet and then select it when you create your ANQ instance.

1. Identify the region in which you want to subscribe to ANQ.
2. In the region, create a new virtual network or select an existing virtual network.
3. In your virtual network, create a new subnet.

   Use the default configuration or update the subnet network configuration based on your network policy.
4. Delegate the newly created subnet to `Qumulo.Storage/fileSystems`.

Load-Balancing ANQ Endpoints

Qumulo provisions multiple endpoints to allow access to ANQ. Every endpoint appears in the Azure Portal as a network interface with an IP address. Qumulo creates a managed resource group under your subscription for these endpoints.

> **Tip**
> To view links to your managed resource groups and network interfaces, use the **Portal** view of your `Qumulo.Storage/fileSystems` resource.

To avoid the bandwidth limits of individual endpoints, use round-robin DNS to distribute your workload traffic across your endpoints.

Configuring Virtual Networking

This section provides an overview of configuring virtual networking for ANQ, including configuration of network security groups, route tables, and back- and front-end networking.

> ⚠ Important
> To enforce network policies for traffic to and from the service, you can apply network security groups and route tables to a delegated subnet.

Configuring Network Security Groups

Network security groups let administrators enforce networking traffic rules. You can assign network security groups to individual network interfaces or to entire subnets.

> ☑ Tip
> Because it is possible to create or remove network interfaces from an ANQ instance, we recommend assigning security groups to a delegated subnet.

To ensure that your configuration doesn't block a specific protocol, follow the guidance in Required Networking Ports for Qumulo Core.

Configuring Route Tables

To configure explicit traffic routing to and from the service, you must attach an Azure route table to a delegated subnet, and then configure your route table.

Common configuration scenarios include routing service traffic:

- Through a firewall
- Through a gateway appliance
- Across multiple virtual network peering configurations

Configuring Back-End and Front-End Networking

The ANQ service uses a *split-networking configuration* in which different network interfaces handle back-end and front-end traffic.

Because it isn't possible to access the back-end network configuration or affect back-end traffic within your ANQ instance, you can configure firewalls and security groups within your virtual network without having to consider back-end connectivity requirements.

Deploying and Viewing Information about Your Azure Native Qumulo Instance

This section explains how to deploy Azure Native Qumulo (ANQ), view information about your service, and connect to the Qumulo Web UI.

For an introduction, see How Azure Native Qumulo Works (page 4).

To Deploy

This section explains how to deploy the ANQ service in Azure.

1. Log in to the Azure Portal and search for `Azure Native Qumulo`.
2. On the **Create a Qumulo resource in Azure** page, on the **Basics** tab, in the **Project details** section:
    a. Select a **Subscription** that you can access as an owner or contributor.
    b. Select a **Resource group** or click **Create new**.

    > **Note**
    > A *resource group* is a container that holds related Azure resources. We recommend creating a resource group exclusive to your Qumulo infrastructure.

3. In the **Azure resource details** section:
    a. Enter a **Resource name**.

    This is the name of your service.

    b. Select a **Region**.

    For more information, see Supported Azure Regions (page 7).

    c. Select an **Availability zone**.

    Azure pins the service resources in a region to this availability zone.

    > **Note**
    > By creating all your Qumulo resources within the same availability zone, Azure can reduce latency.

4. In the **Administrator account** section, enter a **Password** and then re-enter it.

5. In the **Qumulo file system details** section:
    a. Select the **Standard** or **Performance** storage type.
    b. Specify the size of the service to create in TB.
6. In the **Pricing plan** section, select a pricing plan.

    The pay-as-you-go plan is the default plan.

    - For more information about pay-as-you-go price estimates, see the Pricing and Performance Calculator.
    - For up-front pricing plans and free trials, email Azure Native Qumulo Support.
7. On the **Networking** tab, in the **Configure virtual network** section:
    a. Select the **Virtual network** for hosting your service. For more information, see Virtual Networking Prerequisites for ANQ (page 10).
    b. Do one of the following:
        - Select an existing delegated subnet to associate with your service.
        - To create a new delegated subnet, click **Manage subnet configuration**.

    > **Note**
    > You can associate only one delegated subnet with one service instance.
8. On the **Tags** tab, enter any custom tags as a name-value pair.
9. To create a service, click **Next: Review + Create >**.

Viewing Service Information and Connecting to the Web UI

When Azure finishes creating your service, you can view information about the service and start using the Qumulo Core Web UI.

Viewing the IP Addresses of Your Service

To view the IP addresses associated with your service, click **IP Addresses** on the sidebar.

> **Tip**
> We recommend using round-robin DNS to load balance (page 11) traffic across your service IP addresses.

To Log in to the Web UI

To log in to the Web UI, you must identify your service endpoint.

1. Click **Overview** and then copy the **Qumulo Core Web UI Login** URL. For example:

```
https://192.0.0.4/login
```

2. Enter the URL into a browser from a machine that runs, or is connected to, the virtual network where you deployed ANQ.

   > **Note**
   >
   > - If you connect from a machine that is in a different virtual network, establish virtual network peering between the two virtual networks.
   > - If you connect from an on-premises machine, ensure that you connect by using Azure VPN Gateway or Azure ExpressRoute.

3. When the page prompts you for a **Username**, enter `admin`.
4. When the page prompts you for a **Password**, enter the administrator password that you configured previously (page 13).

Connecting Azure Native Qumulo to Microsoft Entra Domain Services

This section explains how to connect Azure Native Qumulo (ANQ) to Microsoft Entra Domain Services (DS).

> ⚠ Important
> On October 1, 2023, Microsoft renamed Azure Active Directory Domain Services to Microsoft Entra Domain Services.

Microsoft Entra DS provides managed domain services such as Windows Domain Join, Group Policy, LDAP, and Kerberos authentication. You can connect your ANQ to standard Active Directory (on-premises AD or self-managed AD in the cloud) or to Microsoft Entra DS.

For information about joining ANQ to standard AD, see Join Your Qumulo Cluster to Active Directory on Qumulo Care.

For information about joining Microsoft Entra DS, see the following resources in the Microsoft Entra documentation.

- Tutorial: Configure virtual networking for a Microsoft Entra Domain Services managed domain
- Tutorial: Join a Windows Server virtual machine to a Microsoft Entra Domain Services managed domain

To Configure Microsoft Entra Domain Services (Microsoft Entra DS)

1. Create an instance of Microsoft Entra DS by entering the following details.
    - **Name:** Your domain name.

      We recommend entering `$DOMAIN.onmicrosoft.com` that the system creates for you.

      You can also use your own custom domain name that acts as a routable or non-routable domain suffix.
    - **VNet:** A VNet and a resource group for your Microsoft Entra DS instance.
    - **SKU:** Standard
    - **Forest:** User

    After the system completes deploying your managed domain (this takes 1-2 hours), it creates the VNet that you specified.
2. Configure DNS for your managed domain.

a. Log in to the Azure portal and search for `microsoft entra domain services`.

b. Click your domain.

c. In the **Required configuration steps** section, under **Update DNS server settings for your virtual network**, write down the domain controllers (DNS servers) that the managed domain deployment created for you, and then click **Configure**.

   For more information, see Update DNS settings for the Azure virtual network in the Microsoft Entra Domain Services documentation.

3. (Optional) If the Microsoft Entra DS managed domain VNet is different from the VNet that you used for deploying ANQ, peer the two VNets.

   For more information, see Configure virtual network peering in the Microsoft Entra Domain Services documentation.

4. Configure the ANQ DNS servers to point to the servers that the managed domain provided for you.

   For more information, see Custom DNS Configuration on Qumulo Care.

5. To finish configuring your file system to work with Microsoft Entra DS, join your Qumulo cluster to AD.

   > **Note**
   > We recommend giving an administrative role to the user who joins the domain. For newly created users, the system requires a password reset when the user logs in to the Azure portal.

Next Steps

After you deploy your Microsoft Entra DS instance and connect ANQ to it, you can configure SAML Single Sign-On (SSO) for your ANQ instance (page 26).

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 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, do the following:
    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 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 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`.

Performance Characteristics and Default Service Limits of Azure Native Qumulo v2

This section describes the performance characteristics and default service limits of Azure Native Qumulo (ANQ) v2.

ANQ v2 introduces a file system architecture that offers improved performance and flexibility for file systems of any size. For this reason, we eliminated the concept of *total available capacity.* You pay only for the data you store and, separately, for the throughput you use.

> **Note**
>
> - We describe the performance of your ANQ v2 instance in terms of throughput (bytes per second), IOPS, and operation latency.
> - Because ANQ v2 is a distributed file system, it is very effective for servicing multi-stream workloads with multiple clients or threads. The throughput and IOPS for single-stream and low-concurrency workloads might be lower than the performance characteristics listed in this section.

Throughput Performance

ANQ v2 instances can perform at above 100 Gbps with high-concurrency, multi-stream workloads. However, when you provision an ANQ v2 instance, Qumulo sets a default service limit of about 4 GBps. If you have a workload that needs higher sustained or peak throughput peak, email Azure Native Qumulo Support to raise this service limit.

> **Note**
> The default service limit isn't a hard cap. In certain scenarios, it might be possible to reach a throughput higher than 4 GBps with default configuration.

IOPS Performance

By default, ANQ v2 is optimized for high-throughput (rather than high-IOPS) workloads.

For workloads with IOPS sensitivity, email Azure Native Qumulo Support for a technical consultation.

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 135) • S3 API (page 191) • 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)

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
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)

Limit Type	Maximum Value
Files in a Directory	4.3 billion
File Size	9 exabytes
Number of 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 maximum number of directories or the entire inode space.
Number of Replication Relationships	100 **Note** If a directory is more than 100 levels below the file system root directory, you can't use it as a replication source.

Replication Version Requirements 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.

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		✅	✅

Authentication

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

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

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

Prerequisites

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

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

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

- To allow the cluster to find group memberships for SAML-authenticated users, configure the Base DN in your AD configuration, even if you don't use POSIX attributes.

- Ensure that your SAML Identity Provider (IdP) is linked to the same AD. An *identity provider* (such as Azure AD, Duo, or Okta) is a system that authenticates users (for example, by using passwords and additional factors).

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

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

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

To Configure SAML SSO for Your Qumulo Cluster

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

1. The cluster administrator contacts the SSO administrator and asks the SSO administrator to create a SAML integration for the Qumulo cluster.
2. The SSO administrator creates a SAML integration with your organization's SSO identity provider (page 26) (IdP).
    a. The SSO administrator uses the cluster's fully qualified domain name (FQDN) format for the service provider (page 26) (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 use the `qq saml_get_settings` command.
- To allow specific changes (for example, correct a typo, update a DNS name or an expired certificate, or temporarily disable SAML SSO without losing any of the other settings), the cluster administrator can use the `qq saml_modify_settings` command to change individual SAML settings independently.
- For first-time SAML configurations, the cluster administrator must provide all of the required settings.
- Aside from a basic check of the IdP certificate, Qumulo Core doesn't verify the configuration parameters. It is the cluster administrator's responsibility to ensure that IdP-initiated SAML login works correctly. (This login type initiates when the user clicks **Continue to SSO login** in the Web UI or selects the Qumulo cluster on the SSO portal.)

Supported SAML SSO Workflows

Qumulo Core supports three SAML SSO workflows:

- Standard SAML workflows that the IdP (page 26) or SP (page 26) initiates
- A workflow that the `qq` CLI initiates

Note

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

IdP-Initiated SSO Worfklow

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

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

SP-Initiated SSO Workflow

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

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

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

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

qq-CLI-Initiated SSO Workflow

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

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

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

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

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

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

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

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

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

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

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

     The user must retry the workflow.

Requiring SSO Authentication for Cluster Management

> ⚠ Important
>
> - If you use the `--require-sso` flag, you can no longer use the `qq login` command with your AD account password. Instead, you must use the `qq sso_login` command (page 30).
> - This setting doesn't restrict access through file protocols such as SMB.
> - Because the FTP protocol sends passwords in plaintext, it is inherently insecure. In addition, many FTP clients don't support Transport Layer Security (TSL) or fall back quietly to the plaintext protocol. For this reason, all Qumulo clusters have FTP disabled by default.

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

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

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

Known Issues and Limitations

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

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

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

- **Azure AD SAML Toolkit:** Currently, due to a configuration deficiency in the toolkit, IdP-initiated SSO isn't operational for Qumulo as a Service. Use the SP-initiated SSO workflow (page 30).

Troubleshooting SAML SSO Authentication

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

Common Issues

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

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

Uncommon Issues

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

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

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

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

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

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

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

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

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

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

If an error occurs, contact your AD administrator and request the correct Base DN. For more information, see Specifying the Base Distinguished Name (Base DN) (page 159).

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 26)
- NFSv4.1 and Kerberos (page 157)
- REST API access tokens (page 39)
- S3 access keys (page 196)

Unaffected Authentication Processes

- Kerberos SMB SSO logins from Domain Local or Trusted Domain users
- NTMLv2 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

Authorization

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 how the permissions for the file or directory were devised. For more information, see Cross-Protocol (XPP) Explain Permissions Tools on Qumulo Care.

External Services

Using Qumulo Core Access Tokens

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

> **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 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 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 46).

Prerequisites

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

Creating and Using Access Tokens

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

To Create an Access Token without an Expiration Time

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

```
$ qq auth_create_access_token jane
```

You can:

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

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

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

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

⚠ Important

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

To Create an Access Token with an Expiration Time

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

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

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

When an access token's expiration time elapses, it isn't possible to use the token for authentication. Any attempt to use the token results in an authentication error. To continue the authentication process, you must either create a new access token (page 39) or update the expiration time for your existing token (page 45).

> **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 40), an item in the `Authorization` HTTP header which acts as the authentication mechanism for the Qumulo Core REST API.

REST API

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

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

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

```
$ curl https://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

Use 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-1-1-12-12345678-1234567890-1234567890-500",
    "uid": null
  },
  "enabled": true,
  "expiration_time": "2023-01-01T00:00:00Z",
  "id": "12345678901234567890123",
  "user": {
    "auth_id": "1002",
    "domain": "LOCAL",
    "gid": null,
    "name": "svc",
    "sid": "S-1-1-12-12345678-1234567890-1234567890-1002",
    "uid": null
  }
}
```


To Get Metadata for All Access Tokens

Use the `qq auth_list_access_tokens` command.

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

The `auth_list_access_tokens` command returns:

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

For example:

```
id                     user   creator  creation time
=====================  =====  =======  ============================
123456789012345678901  svc    admin    2022-10-27T15:18:09.725513764Z
098765432109876543210  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 `auth_create_access_token` (page 40) command.

Modifying the Expiration Time for an Access Token

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

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

```
$ qq auth_modify_access_token 12345678901234567890 --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 39) or update the expiration time for your existing token (page 45).

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

Disabling an Access Token

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

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

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

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

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

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

```
$ qq auth_modify_access_token 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, use the `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 Qumulo Core.
2. Create a service account.
    a. Click **Cluster** > **Local Users & Groups**.
    b. In the **Users** section, click **Create**.
    c. In the **Create user** dialog box, enter a **User name** and **Password**, re-enter the password, and then click **Create**.
3. Create a role with privileges.
    a. Click **Cluster** > **Role Management**.
    b. In the **Role Management** section, click **Create Role**.
    c. On the **Create Role** page, enter a **Name** and **Description**, click the **Privileges** for the user, and then click **Save**.
4. Assign the service user to the role.
    a. On the **Role Management** page, find the name of the role you created and then click **Add Member**.
    b. In the **Add Member to <MyRoleName>** dialog box, for **Trustee**, enter the name of the user you created and then click **Yes, Add Member**.
5. Create access tokens (page 39) for your service account.

Rotating Access Tokens

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

To Rotate an Access Token for a Service Account

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

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

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

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 157).

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_v
     1.1.0.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 53).

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**
   > - If you have only one tenant, it isn't necessary to specify the `tenantId` parameter.
   > - You 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 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**
>
> - In the following example, it is possible to use a fully qualified domain name (FQDN) for the `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 52) 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.

Network Configuration

Required Networking Ports for Qumulo Core

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

> **Note**
> Active Directory authentication services require their own network port range. For an authoritative list, see Active Directory and Active Directory Domain Service Port Requirements in the Windows Server 2008 R2 and Windows Server 2008 documentation.

Networking Ports for Inbound Connections

Port	Protocols	Use
21	TCP	FTP
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
8000	TCP	REST API
9000	TCP	S3 API, if you enable the S3 API for your Qumulo cluster (page 191)
32768-60999	TCP	FTP Passive Mode

Networking Ports for Outbound Connections

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)

Web UI

Setting the Web UI Login Banner

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

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

To Set the Web UI Login Banner

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

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

To Clear the Web UI Login Banner

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

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

To View the Current Web UI Login Banner

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

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

Setting the Web UI Inactivity Timeout

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

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

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

To Set the Web UI Inactivity Timeout

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

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

To Clear the Web UI Inactivity Timeout

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

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

To View the Current Web UI Inactivity Timeout

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

```
qq web_ui_get_settings
```

qq CLI

Enabling Autocomplete for the qq CLI

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

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

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

To Enable Autocomplete for the qq CLI

1. Install the `argcomplete` Python package.

   ```
   pip install argcomplete
   ```

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

2. Activate the `argcomplete` package.

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

3. Search for any conflicting `qq` entries.

   ```
   complete | grep qq
   ```

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

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

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

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

5. Reload your shell profile.

```
source ~/.bashrc
```

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

Enabling Autocomplete for qq CLI Command Aliases

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

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

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

```
qqcreds my_credentials
```

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

> ⚠ Important
> Before you begin, review the source code of the complete-alias helper script. Qumulo 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 64). 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

Use 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

Use 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

Use 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 64).
- 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 64).

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.

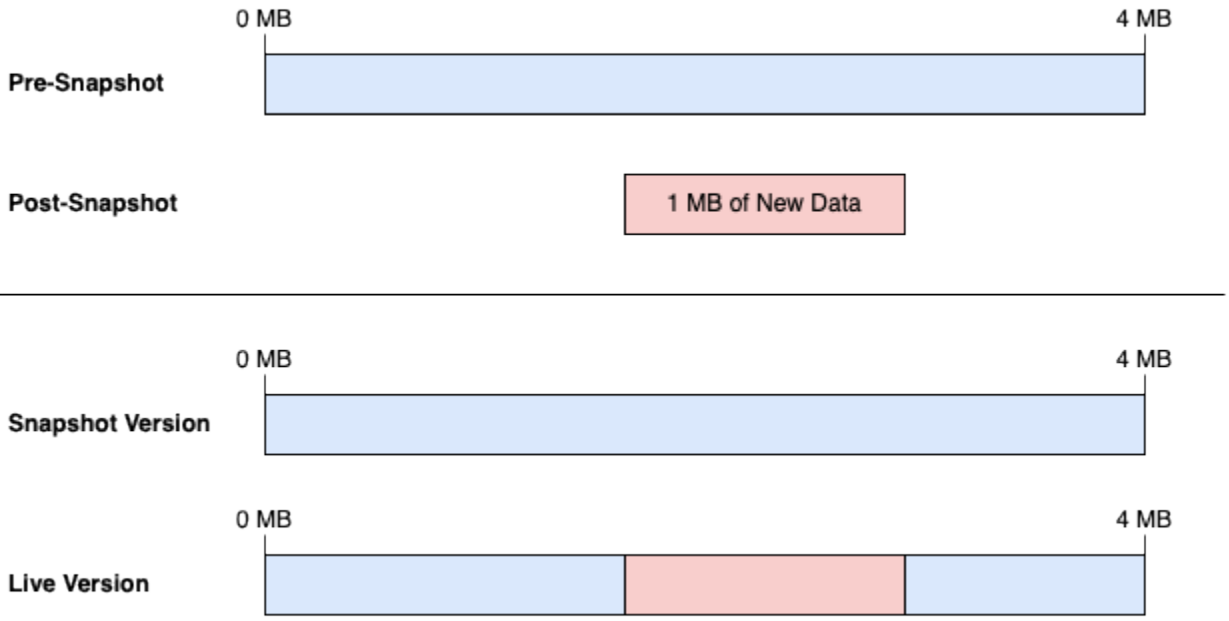
Taking a snapshot doesn't consume storage or incur a performance penalty. There is only a negligible performance penalty for reading and writing snapshotted file system data.

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, use 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, use 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, use 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 use 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 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 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 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, do the following:
    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

Use 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 75).

> **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 75).

```
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

Use 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

Use the `qq snapshot_modify_snapshot` command and specify the snapshot ID from the **Saved Snapshots** page in the 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 86).

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 86). You can also ensure that a snapshot policy locks all new snapshots with a particular key (page 72) 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 77) by associating the key with the replication target.

To Lock a Snapshot by Using the qq CLI

Use 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.

Use 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 83).

3. To unlock the snapshot, use 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

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 filesystem 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, use the `openssl` or `ssh-keygen` tools.

- Use the `openssl` command and specify the path to the private key. For example:

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

- Use the `ssh-keygen` command 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 86).

To Extract the Public Key by Using Linux CLI Tools

1. Use 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. Use 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, you can use 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, use 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, use 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 filesystem 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 75).

> **Tip**
> 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).

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 80).

If You Have Access to the Private Key

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

Use 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

Use 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 82) and Signing a Security Challenge by Using an ECDSA Private Key (page 83).

Retrieving Public Key Information

- To retrieve information for a single public key, use the `qq fs_security_key` command and specify the key identifier or name.
- To retrieve information for all public keys, use 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

Use the `qq fs_security_get_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, use the `qq fs_security_modify_key` command and specify the key identifier or name and the flags for the fields to modify.

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.

If You Have Access to the Existing and Replacement Private Keys

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

Use 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, use 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 83).
3. To rotate the key, use 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 82).

> **Important**
> 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, use 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 87).

Deleting a Public Key

Use the `qq fs_security_delete_key` 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 87).

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 (page 23)

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 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, do the following:
   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 Web UI on the target cluster.

   A notification banner informing you of a new relationship authorization request appears.

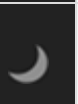
2. On the banner, click **See details**.
3. On the **Replication Relationships** page, click **Accept**.

To View the Status of a Replication Relationship

1. Log in to the 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 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 92)) and then click **Save Relationship**.

To Delete a Replication Relationship

1. Log in to the 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.

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 Web UI
  - HTTPS connectivity to `s3.<region>.amazonaws.com` though one of the following means:
    - Public Internet
    - VPC endpoint
    - AWS Direct Connect

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

  Note
  - For any changes to take effect, user accounts with newly assigned roles must log out and log back in (or their sessions must time out).
  - Use special care when granting privileges to roles and users because certain privileges (such as replication-write privileges) can use system privileges to overwrite or move data to a location where a user has greater permissions. This can give a user access to all directories and files in a cluster regardless of any specific file and directory settings.
- An existing bucket with contents in Amazon S3
- AWS credentials (access key ID and secret access key) with the following permissions:

- `s3:AbortMultipartUpload`
- `s3:GetObject`
- `s3:PutObject`
- `s3:ListBucket`

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

Example IAM Policy

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

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

How Shift-To Relationships Work

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

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

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

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

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

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

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

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

Entity in the Qumulo File System	Entity in an Amazon S3 Bucket
Access control list (ACL)	Not copied
Alternate data streams	Not copied
Directory	Not copied (directory structure is preserved in the object key for objects created for files)
Hard link to a non-regular file	Not copied

Entity in the Qumulo File System	Entity in an Amazon S3 Bucket
Hard link to a regular file	Copy of the S3 object
Holes in sparse files	Zeroes (holes are expanded)
Regular file	S3 object (the object key is the file system path and the metadata is the field data)
SMB extended file attributes	Not copied
Symbolic link	Not copied
Timestamps (`mtime` , `ctime` , `atime` , `btime`)	Not copied
UNIX device file	Not copied

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

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

8. Deletes the temporary snapshot.

Storing and Reusing Relationships

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

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 Qumulo Core.
2. Click **Cluster** > **Copy to/from S3**.
3. On the **Copy to/from S3** page, click **Create Copy**.
4. On the **Create Copy to/from S3** page, click **Local ⇨ Remote** and then enter the following:
    a. The **Directory Path** on your cluster (`/` by default)
    b. The S3 **Bucket Name**
    c. The **Folder** in your S3 bucket
    d. The **Region** for your S3 bucket
    e. Your AWS **Region** (`/` by default)
    f. Your AWS **Access Key ID** and **Secret Access Key**.
5. (Optional) For additional configuration, click **Advanced S3 Server Settings**.
6. Click **Create Copy**.
7. In the **Create Copy to S3?** dialog box, review the Shift relationship and then click **Yes, Create**.

    The copy job begins.

To View Configuration Details and Status of Shift Relationships

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

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

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

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

To Stop a Copy Job in Progress

1. Log in to Qumulo Core.
2. Click **Cluster** > **Copy to/from S3**.
3. To stop a copy job for a specific relationship, click ⋮ > **Abort**.
4. In the **Abort copy from?** dialog box, review the Shift relationship and then click **Yes, Abort**.

   The copy job stops.

To Repeat a Completed Copy Job

1. Log in to Qumulo Core.
2. Click **Cluster** > **Copy to/from S3**.
3. To stop a copy job for a specific relationship, click ⋮ > **Copy Again**.
4. In the **Copy again?** dialog box, review the Shift relationship and then click **Yes, Copy Again**.

   The copy job repeats.

To Delete a Shift Relationship

1. Log in to Qumulo Core.
2. Click **Cluster** > **Copy to/from S3**.
3. To stop a copy job for a specific relationship, click ⋮ > **Delete**.
4. In the **Delete copy from?** dialog box, review the Shift relationship and then click **Yes, Delete**.

   The copy job is deleted.

Using the Qumulo CLI to Copy Files and Manage Relationships

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

Copying Files from Amazon S3

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

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

- AWS access key ID
- AWS secret access key

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

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

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

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

Viewing Configuration Details and Status of Shift Relationships

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

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

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

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


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

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

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

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

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

Stopping a Copy Job in Progress

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

Repeating a Completed Copy Job

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

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

Deleting a Shift Relationship

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

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

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

Troubleshooting Copy Job Issues

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

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

Best Practices

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

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

Restrictions

- **Object-Locked Buckets:** You can't use buckets configured with S3 Object Lock and a default retention period for Shift-To. If possible, either remove the 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: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"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:s3:::my-bucket/my-folder/*"
    }
  ]
}
```

How Shift-From Relationships Work

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

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

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

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

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

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

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

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

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

Storing and Reusing Relationships

The Shift-From relationship remains on the Qumulo cluster. You can monitor the completion status of a job, start new jobs for a relationship after the initial job finishes, and delete the relationship (when you no longer need the S3-folder-Qumulo-directory pair). To avoid

redownloading objects that a previous copy job downloaded, relationships take up approximately 100 bytes 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 Qumulo Core.
2. Click **Cluster** > **Copy to/from S3**.
3. On the **Copy to/from S3** page, click **Create Copy**.
4. On the **Create Copy to/from S3** page, click **Local ⇦ Remote** and then enter the following:
    a. The **Directory Path** on your cluster (`/` by default)
    b. The S3 **Bucket Name**
    c. The **Folder** in your S3 bucket
    d. The **Region** for your S3 bucket
    e. Your AWS **Region** (`/` by default)
    f. Your AWS **Access Key ID** and **Secret Access Key**.
5. (Optional) For additional configuration, click **Advanced S3 Server Settings**.
6. Click **Create Copy**.
7. In the **Create Copy from S3?** dialog box, review the Shift relationship and then click **Yes, Create**.

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

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

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

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

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

To Stop a Copy Job in Progress

1. Log in to Qumulo Core.
2. Click **Cluster** > **Copy to/from S3**.
3. To stop a copy job for a specific relationship, click ⋮ > **Abort**.
4. In the **Abort copy from?** dialog box, review the Shift relationship and then click **Yes, Abort**.

   The copy job stops.

To Repeat a Completed Copy Job

1. Log in to Qumulo Core.
2. Click **Cluster** > **Copy to/from S3**.
3. To stop a copy job for a specific relationship, click ⋮ > **Copy Again**.
4. In the **Copy again?** dialog box, review the Shift relationship and then click **Yes, Copy Again**.

   The copy job repeats.

To Delete a Shift Relationship

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

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

   The copy job is deleted.

Using the Qumulo CLI to Copy Files and Manage Relationships

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

Copying Files to Amazon S3

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

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

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

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

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

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


Viewing Configuration Details and Status of Shift Relationships

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

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

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

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

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


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

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

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

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

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

Stopping a Copy Job in Progress

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

Repeating a Completed Copy Job

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

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

Deleting a Shift Relationship

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

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

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

Troubleshooting Copy Job Issues

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

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

Best Practices

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

- **Inheritable Permissions:** Because the system user creates the files 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).

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 123)
- REST: For more information, see Watching for File Attribute and Directory Changes by Using REST (page 127).

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 119) 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 filesystem 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 119).

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 119) 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 128).
- `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 127).

- 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 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

Use 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 128) for files immediately under the `/my/directory` directory.

To terminate the stream, send a `SIGQUIT` signal.

Streaming Change Notifications by Using the 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 128).
- `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 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, do the following:
    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 151).

    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 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 133)) and then click **Save**.

To Delete an NFS Export

1. Log in to the 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 142).

⚠ Important

- Currently, Qumulo Core 4.3.0 (and higher) supports only NFSv4.1. Mounting with version 4.0 or 4.2 isn't supported.
- The NFSv4.1 protocol requires clients to provide the server with globally unique identifiers. By default, the NFSv4.1 client for Linux uses the machine's hostname as `co_ownerid`. Because the NFSv4.1 protocol requires a unique identifier for every client, an unpredictable failure can occur if two clients have the same hostname. To configure unique identification for your NFS clients, set the `nfs4_unique_id` value for them. For more information, see The nfs4_unique_id parameter in the *Linux kernel user's and administrator's guide*.

Configuring and Using Exports for NFSv4.1

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

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

Differences Between NFSv3 and NFSv4.1 Exports

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

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

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

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

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

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

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

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

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

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

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

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

Preparing Export Configurations for NFSv4.1

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

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

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

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

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

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

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

Visibility of IP-Address-Restricted Exports

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

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

32-Bit Sanitization

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

Enabling NFSv4.1 on a Qumulo Cluster

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

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

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

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

Specifying the NFS Mount Option

> **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.qumulo.cluster:/mount_path \
  /path/to/mountpoint
```

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

```
mount -t nfs \
  -o nfsvers=3,nconnect=16 \
  your.qumulo.cluster:/mount_path \
  /path/to/mountpoint
```

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

Checking Whether NFSv4.1 is enabled

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

```
qq nfs_get_settings
```

Disabling NFSv4.1 on a Qumulo Cluster

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

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

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

Configuring Floating IPs for Nodes

Currently, each Qumulo node is limited to 1,000 clients connected through NFSv4.1 simultaneously. To account for nodes going down, we recommend balancing the number of client connections across your nodes by configuring a sufficient number of floating IP addresses 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, use 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 157)
- 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, use the `nfs4_getfacl` command. In the following example, we create the file `my-file` and then show the ACL for it.

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

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

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

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

Editing the ACL of a File

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

Setting the ACL of a File

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

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

Configuring Access Control Entries (ACEs) and Trustee Representation

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

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

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

ACE Type

In the example of the file ACL (page 142), 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 142), 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 157).

Trustee Representation	Example	Description
`<group>@<domain>`	`group1@domain.example.com`	A Kerberos principal that represents a group in the domain to which that a Qumulo cluster is joined. You can use this format regardless of client mount security, but only when the cluster is joined to AD. The group flag isn't necessary to show that this is a group. For this trustee in this ACE, the system stores the corresponding AD SID for this group principal on disk. For more information about configuring your clients and Qumulo cluster for Kerberos, see How NFSv4.1 Works with Kerberos in Qumulo Core (page 157).
`<S-R-X-Y1-Y2-Yn-1-Yn>`	`S-1-5-32-544`	A raw SID. For more information, see Security Identifiers in the Microsoft documentation. To store a SID on disk for this trustee, you can use this format in place of a Kerberos principal. An AD SID must be a user or a group, but can't be both. However, the group flag isn't necessary for showing whether the SID represents a user or group. This can be useful if you have SIDs in a foreign domain (that is, a domain that the cluster isn't joined to). You can use this representation when the cluster isn't joined to a domain at all. When you retrieve an ACL by using `nfs4_getfacl`, the presentation for joined domain SIDs is `<group>@<domain>` and the presentation for foreign SIDs is `<S-R-X-Y1-Y2-Yn-1-Yn>`.
`<numeric_uid>`	`1234`	A numerical UID for an `AUTH_SYS` user. For this trustee in the ACE, the system stores this UID on disk.

Trustee Representation	Example	Description
`<numeric_gid>`	`5678`	A numerical GID for an `AUTH_SYS` user. To avoid having the group interpreted as a user, you must specify the group flag (page 143). 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.
`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 trusteee representations interchangeably, even within a single ACL. For example, the following ACL is possible for a file:

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

The Access Types to Which the ACE Applies

For example:

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

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

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

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

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

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

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

Using NFSv4.1 ACLs with SMB Access Control

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

Changing File Owners

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

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

$ sudo chown user2 /mnt/qumulo/my_file

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

Using Equivalent NFSv4.1 and Qumulo ACL Commands

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

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

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

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

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

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

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

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

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 157), 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 lookups 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 lookups. (page 155)

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 Web UI or the `qq` CLI.

To Add a Host Access Rule by Using the Web UI

1. Log in to Qumulo Core.
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, do the following:
    a. For **Allowed Hosts**, enter a comma-separated host access rule. (page 151)
    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. Use 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 JSON output from the command.

```
{
  "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 from the command.

   ```
   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, use 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 JSON output from the command.

```
[{
  "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.
   - Use the `dns_resolve_ips` (page 154) 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, use the `qq dns_clear_lookup_cache` command.
   - Use the `qq dns_resolve_hostnames` command and specify the hostname to perform a lookup for `user1.accounting.example.com`.

     The following is example output from the command.

     ```
     [{
       "hostname": "user2.accounting.example.com",
       "ip_addresses": ["203.0.113.1"],
       "result": "OK"
     }]
     ```

   - Use 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 Lookups

Qumulo Core checks hostnames by performing a reverse-DNS lookup on the cluster. Because continuous reverse-DNS lookups 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 157).

Known Kerberos Limitations for Qumulo Core

Qumulo Core supports only the following features:

- NFSv4.1
- Linux clients

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

Prerequisites for Joining a Qumulo Cluster to Active Directory

This section describes the prerequisites for joining a Qumulo Cluster to Active Directory for using NFSv4.1 with Kerberos.

For more information, see Join Your Qumulo Cluster to Active Directory on Qumulo Care.

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

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

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

Specifying the Base Distinguished Name (Base DN)

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

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

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

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

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

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

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

Using the Active Directory Domain Controller as the NTP Server

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

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

Configuring Active Directory for Use With Kerberos

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

Configuring DNS in Active Directory

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

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

Modifying the Default DNS Configuration

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

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

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

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

Configuring DNS for Distributing Workflows Across Nodes

The Qumulo distributed file system works best when you spread the workload evenly across multiple nodes. We recommend configuring 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 159).
2. Find the DNS entry for the cluster on the DNS server.

   Unless you renamed the cluster after joining it to AD, this entry is generally the cluster's name. To find the machine account name in the Web UI, click **Cluster** > **Active Directory** and 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 Web UI, click **Cluster** > **Network Configuration**.
4. Configure the DNS resolver to point to the DNS server.

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

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

Configuring the Service Principal Name (SPN) for NFS

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

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

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

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

   To find the machine account name in the Web UI, click **Cluster** > **Active Directory** and 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 159), you must configure the NFSv4.1 server (page 135) and NFSv4.1 security settings.

To Configure Security Settings by Using the qq CLI

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

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

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

   This is the default configuration:

   - NFSv4.1 is disabled by default.
   - `AUTH_SYS`, `AUTH_KRB5`, `AUTH_KRB5P`, and `AUTH_KRB5I` are enabled by default (however, Qumulo Core doesn't support Kerberos configuration on NFSv3).

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

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

```
$ qq nfs_modify_settings --disable-auth-sys
{
  "v4_enabled": false,
  "auth_sys_enabled": false,
  "auth_krb5_enabled": true,
  "auth_krb5p_enabled": true,
  "auth_krb5i_enabled": true
}
```

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

Command	Description
`qq nfs_modify_settings --enable-auth-sys`	Enables `AUTH_SYS` without changing `AUTH_KRB5`
`qq nfs_modify_settings --enable-krb5`	Enables `AUTH_KRB5` without changing `AUTH_SYS`
`qq nfs_modify_settings --enable-krb5p`	Enables `AUTH_KRB5P` without changing `AUTH_SYS`
`qq nfs_modify_settings --enable-krb5i`	Enables `AUTH_KRB5I` without changing `AUTH_SYS`
`qq nfs_modify_settings --enable-v4`	Enables NFSv4.1
`qq nfs_modify_settings --disable-v4`	Disables NFSv4.1
`qq nfs_modify_settings --disable-krb5`	Disables `AUTH_KRB5` without changing `AUTH_SYS`
`qq nfs_modify_settings --disable-krb5p`	Disables `AUTH_KRB5P` without changing `AUTH_SYS`
`qq nfs_modify_settings --disable-krb5i`	Disables `AUTH_KRB5I` without changing `AUTH_SYS`

Note

- Security configuration options apply to *all* versions of NFS (NFSv3 and NFSv4.1). Thus, disabling `AUTH_SYS` also disables NFSv3, because `AUTH_SYS` is the only Kerberos security flavor (page 157) 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 135) 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 157): `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, use the `qq fs_file_get_attr` command. For example:

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

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

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

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

Listing Security Identifiers (SIDs)

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

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

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

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

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

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

Using Kerberos Principals

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

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

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

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

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

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

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

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

Understanding Kerberos Principal Caveats

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

Machine Account Object Names

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

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

ID Mapping on Linux systems

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

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

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

Unicode Characters in Kerberos Principals

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

Using the chown Tool With Kerberos

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

The AUTH_SYS Root User

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

The Kerberos Root User

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

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

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

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

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

Viewing the Owner and Group

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

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

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

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

Using the Linux ACL Editor

The Linux ACL Editor consists of the following tools:

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

You can use the editor to read and write ACLs on a Qumulo cluster that uses NFSv4.1 with Kerberos. For more information, see Managing File Access Permissions by Using NFSv4.1 Access Control Lists (ACLs) (page 142).

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 176).

To Join a Linux Client to a Domain by using realmd

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

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

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

2. Configure the `sssd` and `idmapd` tools manually. For more information, see Mapping External Identities to Linux Identities (page 176).

To Configure DNS and Service Principal Name (SPN)

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

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

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

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

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

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

Mapping External Identities to Linux Identities

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

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

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

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

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

Configuring Active Directory Authentication by using sssd

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

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

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

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

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

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

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

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

          Important
          AD doesn't prevent duplicate UID or GID numbers from being added to RFC 2307 values. For this reason, incorrect configuration can lead or UID or GUID collisions. When a Linux system determines that a collision has occurred, it chooses the first UID or GID it finds.

        - Otherwise, the UID or GID becomes `nobody` or `nogroup` (`65534`).

          Note
          In most cases, an owner or group becomes 65534 as a result of incorrect user mapping configuration in the client. To understand which LDAP queries run and why they have trouble finding the correct information, check your logs.

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

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

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

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

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

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

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

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

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

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

Authenticating as an AD User and Mounting Your Qumulo Cluster

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

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

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

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

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

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

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

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

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

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

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

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

3. Use the Qumulo file system.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

6. Use the Qumulo file system.

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

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

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

autofs is a dæmon that manages mount points for individual Linux users. For this reason, Linux users have different views of a mount point. `autofs` can authenticate an AD user through `ssh`, the Linux 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 use the following command.

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

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

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

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

    For more information, see Autofs in the Ubuntu documentation.

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

    ```
    /mnt/qumulo_mount_point -vers=4.1,sec=krb5 <qumulo-cluster>.my-domain.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 Web UI, on the **Active Directory** page, for **Use Active Directory as your primary time server**, click **Yes**.
- To configure any other NTP server in the Web UI, click **Cluster** > **Date & Time**.

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

Configuring Cross-Domain Active Directory Trusts

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

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

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

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

Configuring the Base DN

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

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

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

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

Enabling More Secure Trust Encryption Types

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

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

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

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

Troubleshooting NFSv4.1 with Kerberos

This section describes common troubleshooting procedures for configuring NFSv4.1 to work with Kerberos.

Following General Debugging Techniques

This section lists common debugging techniques.

To Turn Up Logging Levels for Client-Side Tools

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

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

4. Turn on `rpcdebug`, for example:

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

Taking a Client-Side Packet Capture

Normally, there should be:

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

Because a Kerberos mount requires the client to perform a series of steps, in most cases, the last traffic that the client issues indicates the source of failure. To view encrypted Kerberos traffic, use Wireshark with a Kerberos keytab file. For more information, see Kerberos in the Wireshark documentation.

For help with interpreting logging and metrics from your Qumulo cluster and for insights from the telemetry of our Kerberos implementation, contact 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 159).

Diagnosing Mount-Failed Errors

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

1. Confirm that the `rpc.gssd` service is running.
2. Confirm that the cluster and client both resolve from the client. It should be possible to reach the cluster and client through a fully qualified domain name (FQDN), such as `my-machine.my-domain.example.com`.
3. Confirm that reverse DNS works for the IP addresses on both the client and the cluster.
4. Confirm that the client has a `host` service principal name (SPN) and that the cluster has an `nfs` SPN that matches the DNS records.
5. Do one of the following:
    - If you use a machine account or `kinit` authentication, confirm that the credentials are correct. You can use the keytab `ktutil` command or the credential cache `klist` command to list the encryption methods.

- Confirm that Kerberos tickets use AES-128 or AES-256 for service encryption by examining a packet capture or your Active Directory Kerberos settings.

6. If you use domain trusts, confirm that trust has AES-128 or AES-256 enabled.
7. Confirm that the clocks on the client, cluster, and domain controller are synchronized to the same time.
8. Inspect logs and packet captures.

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 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, do the following:
    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**, do the following:
        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 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 189)) and then click **Save**.

To Delete an SMB Share

1. Log in to the 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**.

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 tools.

> ⚠ 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 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, use the `qq s3_modify_settings --insecure` command.
- To revert to encrypted HTTPS requests, use the `qq s3_modify_settings --secure` command.

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 196).

Use the `qq s3_create_access_key` 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 236).

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 192) and the `aws configure` command to:

    a. Specify your profile and access key ID (page 196). For example:

    ```
    $ aws configure
      --profile my-qumulo-profile set aws_access_key_id \
        000000000001fEXAMPLE
    ```

    b. Specify your profile and secret access key (page 196). 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-qumulo-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 191) manually, use 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) privilege and permission to create a directory under the cluster's root directory.

Use 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 204).

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 215). This approach requires no credentials but limits users to non-modifying operations. For more information, see To Enable Anonymous Access to S3 Buckets by Using the qq CLI (page 215).

Prerequisites

Managing S3 access keys requires the following role-based access control (RBAC) 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 198).

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 196) 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 213).

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 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 200).

Note

- To facilitate key rotation, each user identity (page 196) 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 196) by using the Qumulo REST API or `qq` CLI.

To create S3 access keys, you must have an administrator account or have .

To Create an Access Key by Using the qq CLI

To create an S3 access key for a particular user identity (page 196), use 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 from the command.

```
{
  "access_key_id": "000000000001fEXAMPLE",
  "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": "TEIT4liMZ8A32iI7JXmqIiLWp5co/jmkjEXAMPLE"
}
```


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 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 from the response.

```
{
  "access_key_id": "000000000001fEXAMPLE",
  "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": "TEIT4liMZ8A32iI7JXmqIiLWp5co/jmkjEXAMPLE"
}
```


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, use the `qq s3_list_access_keys` command:

  The following is example output from the command. 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 from the command. The command returns a single JSON object that contains the combined responses from calls to the `/v1/s3/access-keys/` Qumulo REST API endpoint.


```
{
  "entries": [
    {
      "access_key_id": "000000000001fEXAMPLE",
      "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 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 from the response. 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": "000000000001fEXAMPLE",
      "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

Use 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 REST API

Send a `DELETE` request to the `/v1/s3/access-keys/<access-key-id>` Qumulo 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 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 184)

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) 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 196)
- A configured AWS CLI (page 192)

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 204) 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 206).
- If you don't specify a directory, the Qumulo REST API and `qq` CLI use the default bucket directory prefix (page 206).

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 213).

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 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 205).

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 Configure the Default Bucket Directory Prefix by Using the qq CLI

1. To view the current default bucket directory prefix, use the `qq s3_get_settings` command.

   The following is example output from the command.

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

2. To change the setting, use 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 206).

- To create a new, empty bucket from the default bucket directory prefix (page 206), use 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, use 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

Use 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 206) 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.

Although you can configure global settings, such as the default bucket directory prefix (page 206) for S3 buckets, the only individual S3 bucket setting that you can configure in Qumulo Core is anonymous access. For more information, see Enabling Anonymous Access for an S3 Bucket (page 214).

- To view the current bucket configuration by using the Qumulo REST API or `qq` CLI, you need the `PRIVILEGE_S3_BUCKETS_READ` privilege.
- To change the bucket configuration, you need the `PRIVILEGE_S3_BUCKETS_WRITE` privilege.

Viewing the Current S3 Bucket Configuration by Using the qq CLI

Use the `qq s3_get_bucket` command and specify the bucket name. For example:

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

The following is example output from the command. All times are in the UTC time zone..

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

Listing S3 Buckets

You can list 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.

- Use the `qq s3_list_buckets` command.

  The following is example output from the command. All times are in the UTC time zone.

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

- For JSON output, use the `--json` flag.

  The following is example output from the command. All times are in the UTC time zone. The JSON output contains an array named `Buckets` that contains the individual buckets as objects.

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

To List S3 Buckets by Using the S3 API

Use the `aws s3api list-buckets` command. This command uses the `ListBuckets` S3 API action.

The following is example output from the command. 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 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, use 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` , 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

Use 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 196) to the user. Alternatively, you can create presigned URLs (page 213) or enable read-only, anonymous access (page 215) 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 216).

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 219) 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 196) in the request to its corresponding auth ID (page 198), 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, use 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 from the command, with line breaks inserted for readability. 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 217).

> **ℹ 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 configuration by using the Qumulo REST API or `qq` CLI, you need the `PRIVILEGE_S3_BUCKETS_READ` privilege.
- To change the bucket configuration, you need the `PRIVILEGE_S3_BUCKETS_WRITE` privilege.

To Enable Anonymous Access to an S3 Bucket by Using the qq CLI

1. To view the current bucket configuration, use the `qq s3_get_bucket` command and specify the bucket name. For example:

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

   The following is example output from the command. All times are in the UTC time zone.

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

2. Anonymous access to S3 buckets is disabled by default. To enable anonymous access, use the `qq s3_modify_bucket` command, specify the bucket name, and use the `--enable-anonymous-access` flag.
3. To disable anonymous access, use the `qq s3_modify_bucket` command, specify the bucket name, and use the `--disable-anonymous-access` flag.

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 204) 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) on Qumulo Core.

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

Use 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

Use 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

Use 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, pre-signed URLs, and file system access control lists (page 216).

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 223) field of the policy statement (page 220).

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`

Default No-Policy State

By default, S3 buckets have *no policy* in a Qumulo cluster. In this state, there are no additional restrictions for authenticated users and any user with a valid access key and file system permissions can perform any S3 API action on the S3 bucket. However, the access control list (ACL) of the S3 bucket's root directory must grant a user some amount of access.

> **Note**
>
> - In the default non-policy state, Qumulo Core disallows unsigned, anonymous requests and the `qq s3_get_bucket_policy` command returns `{}`. To enable anonymous access, use the `qq s3_set_bucket_policy` command with an `Allow` statement that targets the `local:guest` account.
> - To remove an access policy from an S3 bucket, use the `qq s3_delete_bucket_policy` command.

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 196) or presigned URLs (page 213), or enable read-only, anonymous access (page 215) to the S3 bucket.
- Configure inheritable file ACLs (page 216) by using the `qq` CLI, SMB, or NFSv4.1 access control lists (ACLs) (page 142).
- Ensure that you have the following required role-based access control (RBAC) 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 223) 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 223) 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 226) 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, use 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 225)) 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 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:GetBucketAcl`	Retrieve the access control list (ACL) for the S3 bucket

API Action	Description
`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
`s3:ListBucketMultipartUploads`	Enumerate all multipart uploads to the S3 bucket

API Action	Description
`s3:ListMultipartUploadParts`	Enumerate all multipart upload parts in the S3 bucket
`s3:PutBucketPolicy`	Configure the access policy 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 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 details, prefixed by `uid:`
`gid:1234`	A Posix GID that identifies users by their RFC-2307 details, prefixed by `gid:`
`auth_id:12345678`	The numeric `auth_id` of a user or group

Identity Specification Example	Description
• `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 226) 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_BUCKETS_WRITE`	• `s3:DeleteBucket` • `s3:DeleteBucketPolicy` • `s3:PutBucketPolicy`
`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 228).

Prerequisites

To manage multipart S3 uploads by using the `qq` CLI, you need the following role-based access control (RBAC) 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 230) or you abort and clean up the multipart upload manually (page 231).

To check how much space incomplete multipart uploads use on your cluster, you can list the uploads by using the Qumulo REST API or `qq` CLI. For more information, see Listing Multipart Uploads (page 228).

How System-Initiated Multipart S3 Uploads Work

Occasionally, when you list your multipart uploads (page 228), 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 230).

> **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 230).

Listing Incomplete Multipart S3 Uploads

You can list the incomplete multipart uploads for a single S3 bucket by using the Qumulo REST API or `qq` CLI.

> **Note**
>
> - If you use the `ListMultipartUploads` S3 API action, the system doesn't show system-initiated uploads (page 228) or how much space the uploads use on your cluster.
> - If you use the Qumulo REST API or `qq` CLI, Qumulo Core shows system-initiated uploads (page 228) and how much space each upload uses on your cluster.

- To list incomplete uploads by using the `qq` CLI, use the `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 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 REST API or `qq` CLI.

To configure the expiry interval for all current and future multipart uploads by using the `qq` CLI, use the `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 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 228).

> **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, use the `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 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.

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.

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
`DeleteBucketPolicy`	7.0.1
`DeleteObject`	5.2.1
`DeleteObjectTagging`	6.3.2
`DeleteObjects`	5.2.2
`GetBucketAcl`	6.1.1
`GetBucketLocation`	5.1.2
`GetBucketPolicy`	7.0.0.1

API Action	Supported from Qumulo Core Version
`GetBucketVersioning`	7.0.0.1
`GetObject`	5.0.4
`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
`PutObject`	5.2.1
`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 216).

Unsupported Feature	Description
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	—
Object versioning	Qumulo objects have one object version. To preserve previous object contents in Qumulo Core, use snapshots.
Policies	—
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	—
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 192).

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 204)	`Prefix` limits results to object keys that begin with the prefix
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`.

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 20).

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

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

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 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 39) 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 248).

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 241)	• `max_drive_failures` • `max_node_failures` • `name` • `platform` • `service_model` • `uuid` • `version`	5.3.0	Qumulo Core information, including the cluster name, cluster UUID, and the current Qumulo Core version
`qumulo_node`	`info` (page 241)	• `node_id` (page 250) • `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 240)	• `domain_url` (page 249) • `server_url` (page 251)	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 241)	• `domain_url` (page 249) • `server_url` (page 251)	5.3.0	The total latency for AD `NETLOGON` requests
`qumulo_ad_netlogon_requests`	`counter` (page 240)	• `domain_url` (page 249) • `server_url` (page 251)	5.3.0	The total number of completed AD `NETLOGON` operations
`qumulo_cpu_max_temperature_celsius`	`gauge` (page 240)	• `cpu` (page 248) • `node_id` (page 250)	5.3.1	The maximum temperature threshold for each physical CPU

Metric Name	Metric Type	Labels	Supported from Qumulo Core Version	Description
`qumulo_cpu_temperature_celsius`	`gauge` (page 240)	• `cpu` (page 248) • `node_id` (page 250)	5.3.0	The temperature for each physical CPU, in degrees Celsius
`qumulo_disk_endurance_percent`	`gauge` (page 240)	• `disk_type` (page 249) • `drive_bay` (page 249) • `node_id` (page 250)	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 240)	• `disk_type` (page 249) • `drive_bay` (page 249) • `node_id` (page 250)	5.3.2	The total number of communication errors between the specified drive and its host.
`qumulo_disk_uncorrectable_media_errors`	`counter` (page 240)	• `disk_type` (page 249) • `drive_bay` (page 249) • `node_id` (page 250)	5.3.2	The total number of uncorrectable errors on the specified drive's physical media.
`qumulo_disk_is_unhealthy`	`gauge` (page 240)	• `disk_type` (page 249) • `drive_bay` (page 249) • `node_id` (page 250)	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 241)	• `disk_type` (page 249) • `drive_bay` (page 249) • `io_type` (page 250) • `node_id` (page 250)	5.3.0	The total latency for disk I/O operations

Metric Name	Metric Type	Labels	Supported from Qumulo Core Version	Description
qumulo_fan_speed_rpm	gauge (page 240)	• fan (page 249) • node_id (page 250)	5.3.0	The fan speed, in RPM
qumulo_fs_capacity_bytes	gauge (page 240)	—	5.3.0	The total cluster space, in bytes
qumulo_fs_directory_tree_entries	gauge (page 240)	• entry_type (page 249) • path (page 250)	5.3.0	The number of file system objects on the cluster, sorted by object type
qumulo_fs_directory_used_bytes	gauge (page 240)	• path (page 250) • usage_type (page 251)	5.3.0	The amount of space that object types use, in bytes
qumulo_fs_free_bytes	gauge (page 240)	—	5.3.0	The free space on the cluster, in bytes
qumulo_fs_snapshots	gauge (page 240)	—	5.3.0	The number of snapshots on the cluster
qumulo_ldap_lookup_request_errors	counter (page 240)	• domain_url (page 249) • server_url (page 251)	5.3.0	The total number of LDAP requests that resulted in an error
qumulo_ldap_lookup_request_latency_seconds	histogram (page 241)	• domain_url (page 249) • server_url (page 251)	5.3.0	The total latency of LDAP requests
qumulo_ldap_lookup_requests	counter (page 240)	• domain_url (page 249) • server_url (page 251)	5.3.0	The total number of completed LDAP requests

Metric Name	Metric Type	Labels	Supported from Qumulo Core Version	Description
`qumulo_ldap_operation_errors`	`counter` (page 240)	`domain_url` (page 249)	5.3.0	The total number of LDAP operations that resulted in an error
`qumulo_ldap_operation_latency_seconds`	`histogram` (page 241)	`domain_url` (page 249)	5.3.0	The total latency for LDAP operations
`qumulo_ldap_operations`	`counter` (page 240)	`domain_url` (page 249)	5.3.0	The total number of completed LDAP operations
`qumulo_memory_correctable_ecc_errors`	`counter` (page 240)	`node_id` (page 250)	5.3.0	The total number of memory errors that Qumulo Core corrected automatically
`qumulo_network_interface_is_down`	`gauge` (page 240)	• `bond` (page 248) • `interface` (page 249) • `role` (page 250) • `node_id` (page 250)	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 240)	• `bond` (page 248) • `interface` (page 249) • `role` (page 250) • `node_id` (page 250)	5.3.0	The negotiated link speed for the specified interface

Metric Name	Metric Type	Labels	Supported from Qumulo Core Version	Description
`qumulo_network_interface_receive_errors`	`counter` (page 240)	• `bond` (page 248) • `interface` (page 249) • `role` (page 250) • `node_id` (page 250)	5.3.0	The total number of receive errors on the specified interface
`qumulo_network_interface_received_bytes`	`counter` (page 240)	• `bond` (page 248) • `interface` (page 249) • `role` (page 250) • `node_id` (page 250)	5.3.0	The total bytes received on the specified interface
`qumulo_network_interface_received_packets`	`counter` (page 240)	• `bond` (page 248) • `interface` (page 249) • `role` (page 250) • `node_id` (page 250)	5.3.0	The total number of packets received on the specified interface
`qumulo_network_interface_transmit_errors`	`counter` (page 240)	• `bond` (page 248) • `interface` (page 249) • `role` (page 250) • `node_id` (page 250)	5.3.0	The total number of transmission errors on the specified interface
`qumulo_network_interface_transmitted_bytes`	`counter` (page 240)	• `bond` (page 248) • `interface` (page 249) • `role` (page 250) • `node_id` (page 250)	5.3.0	The total number of bytes transmitted on the specified interface

Metric Name	Metric Type	Labels	Supported from Qumulo Core Version	Description
`qumulo_network_interface_transmitted_packets`	`counter` (page 240)	• `bond` (page 248) • `interface` (page 249) • `role` (page 250) • `node_id` (page 250)	5.3.0	The total number of packets transmitted on the specified interface
`qumulo_power_supply_is_unhealthy`	`gauge` (page 240)	• `location` (page 250) • `node_id` (page 250)	5.3.0	PSU health, `0` (healthy) or `1` (unplugged, removed, or unresponsive)
`qumulo_protocol_client_connections`	`counter` (page 240)	`protocol` (page 250)	5.3.0	The total number of clients that have connected to the specified protocol
`qumulo_protocol_client_disconnections`	`counter` (page 240)	`protocol` (page 250)	5.3.0	The total number of clients that have disconnected from the specified protocol
`qumulo_protocol_operation_bytes`	`counter` (page 240)	• `data_type` (page 249) • `io_type` (page 250) • `op_name` (page 250) • `protocol` (page 250)	5.3.0	The total bytes that protocol operations have transferred
`qumulo_protocol_operation_latency_seconds`	`histogram` (page 241)	• `data_type` (page 249) • `io_type` (page 250) • `op_name` (page 250) • `protocol` (page 250)	5.3.0	The total latency for protocol operations

Metric Name	Metric Type	Labels	Supported from Qumulo Core Version	Description
`qumulo_protocol_operations`	`counter` (page 240)	• `data_type` (page 249) • `io_type` (page 250) • `op_name` (page 250) • `protocol` (page 250)	5.3.0	The total number of completed protocol operations
`qumulo_quorum_node_is_offline`	`gauge` (page 240)	`node_id` (page 250)	5.3.0	The online status for each node in the cluster, `0` (node online) or `1` (node offline)
`qumulo_time_is_not_synchronizing`	`gauge` (page 240)	`node_id` (page 250)	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

Label Name	Possible Values	Description
`data_type`	• `data` : Read or write operations on the data of a file. • `metadata` : Operations (such as `lookup` , `stat` , or `getattr`) unrelated to a file's data • `none` : Operations that operate on neither the file data nor the metadata. **Note** The protocol often requires these operations for session negotiation and authentication.	The data type that an operation transfers
`disk_type`	• `hdd` : Hard Disk Drive • `ssd` : Solid-State Drive	The underlying storage type
`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

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

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