Qumulo Integration Guide

Table of Contents

Integrating with Varonis

Integrating with Varonis

How Qumulo Core Integrates with Varonis

This section explains how Qumulo Core integrates with Varonis by using Qumulo Broker.

The Qumulo-Varonis integration monitors file and directory operations in Qumulo Core. When events take place in a Qumulo system, Qumulo Core adds the events to audit logs which track all actions that users take within a Qumulo namespace, including data access and modification, file system access, data sharing through new SMB shares or NFS exports, and system configuration changes. Qumulo Core uses the Qumulo Broker (page 3) to process and send audit logs to Varonis.

How the Qumulo-Varonis Integration Works

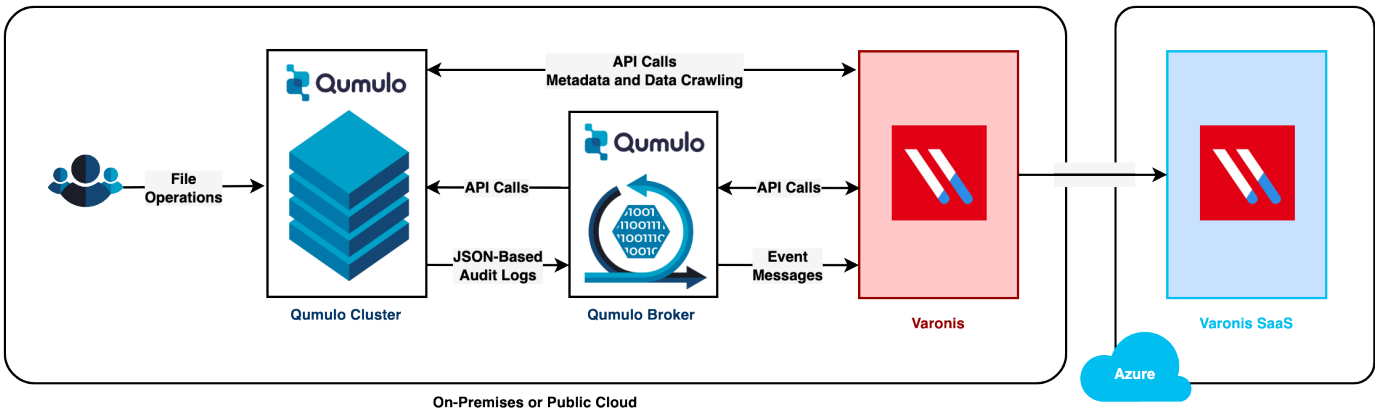
This section describes how the Qumulo-Varonis integration works. It provides an overview of the integration workflow; explains how Qumulo Broker gathers, processes, and emits Qumulo Core audit logs; and describes how Qumulo Broker uses rsyslog queues to ensure efficient data transfer.

How Qumulo Clusters Send Audit Log Data to Varonis

Qumulo Core sends audit logs for each supported file- and directory-level operation (page 5) in real time to Varonis for continuous monitoring. To detect anomalous behavior that system administrators can use to detect potential activity from a bad actor (for example, abnormal or high-frequency changes in file activity—such as file creation, deletion, and modification—or changes to access permissions), Varonis applies machine learning to Qumulo Core audit logs and issues alerts. In addition to common patterns, Varonis uses thread feeds and blacklists to identify known ransomware and attack patterns.

The following architecture diagram shows the workflow between Qumulo Broker and Qumulo Core.

> **Note**
> We recommend installing Qumulo Broker and Varonis in the same VLAN or VPC.



> **Note**
> Although Qumulo currently is certified only for the Varonis SaaS offering, you can configure and use the SaaS offering with an on-premises Qumulo cluster.

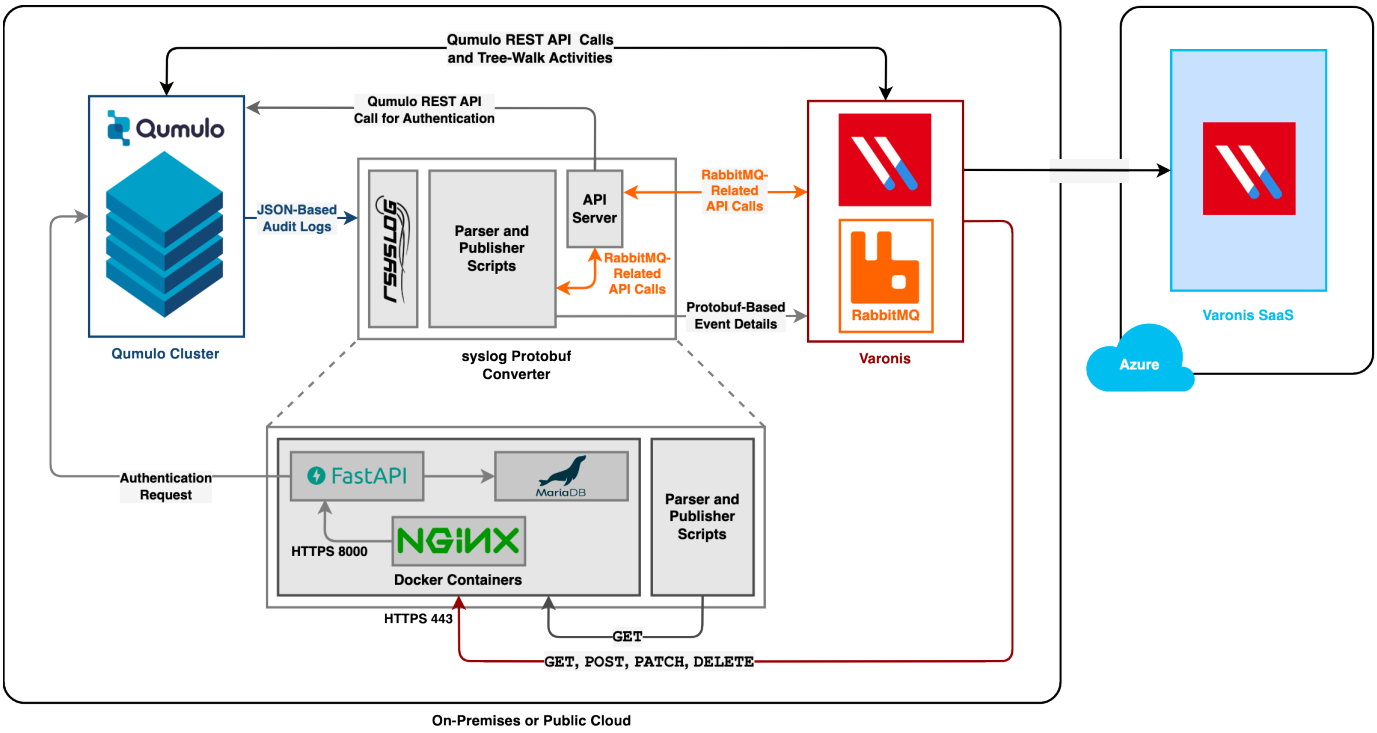
How Qumulo Broker Gathers, Processes, and Emits Data

In Qumulo Core, each audit log has a specific logging requirement (for example, certain log types include only specific fields). Although normally Qumulo Core outputs audit logs in CSV format, it can output these additional fields in JSON format. For more information, see Configure Qumulo Audit Logging by Using the qq CLI (page 8).

Typically, Qumulo Core sends the audit logs to a single remote syslog instance. In the Qumulo-Varonis integration, Qumulo Broker receives the audit logs from multiple Qumulo clusters, converts them to various formats, and then sends them to Varonis.

> **Note**
> Qumulo Core can send audit logs to only one target syslog instance. For information about sending your Qumulo audit logs to different target systems in addition to Varonis, see Configuring rsyslog to Communicate with Multiple Clusters (page 12).

The following architecture diagram shows how Qumulo Broker gathers, processes, and emits data.



Qumulo Broker Specifications

This section describes the specifications for Qumulo Broker, including system requirements, prerequisites, firewall definitions, and supported operations. Deploy Qumulo Broker on a stand-alone machine (or virtual machine) so that it sits between your Qumulo cluster and Varonis. For more information, see the Qumulo-Varonis integration architecture diagram (page 2).

System Requirements

We recommend the following system requirements for Qumulo Broker.

- 8-core processor
- 16 GB memory
- 200 GB disk space

Prerequisites

Deploying Qumulo Broker requires:

- Qumulo Core 6.0.2 (and higher)
- Git
- Docker 23.0.1 (and higher)
- rsyslog 8.2001 (and higher)

Firewall Definitions

In addition to the Varonis firewall requirements, you must also define the following firewall rules for Qumulo Broker connections.

Port	Protocol	Source IP address	Destination IP address	Description
22	TCP	The system administrator's machine	Qumulo Broker	Qumulo Broker SSH connection
443	TCP	Qumulo Broker	GitHub and Docker Hub	Temporary GitHub and Docker Hub connections from Qumulo Broker
443	TCP	Varonis	Qumulo Broker	Qumulo Broker API calls
514	TCP	Qumulo Core (persistent and floating IP addresses)	Qumulo Broker	Qumulo Broker syslog connection
8000	TCP	Qumulo Broker IP address	Qumulo Core persistent and floating IP addresses	Qumulo Core API calls from Qumulo Broker

Supported Operations

Qumulo Broker supports the following file- and directory-level operations.

File-Level Operations	Directory-Level Operations
• Add file permissions • Add file protection • Change file owner • Create file • Delete file • Read file • Rename file • Remove file permissions • Remove file protection • Write to file	• Add directory permissions • Add directory protection • Change directory owner • Create directory • Delete directory • Rename directory • Remove directory permissions • Remove directory protection

Integrating Qumulo Core with Varonis

This section explains how to integrate Qumulo Core with Varonis by deploying Qumulo Broker.

Deploy Qumulo Broker on a stand-alone machine (or virtual machine) so that it sits between your Qumulo cluster and Varonis. For more information, see the Qumulo-Varonis integration architecture diagram (page 2).

Step 1: Prepare for Deploying Qumulo Broker

This section explains how to prepare your Qumulo Broker machine for deployment.

1. Clone the Qumulo Broker repository from GitHub into the `/opt/qumulo` directory on your Qumulo Broker machine.
2. To configure the system to use the MariaDB database, edit the `/opt/qumulo/QumuloBroker/api/.env` file and specify the values for the `MYSQL_ROOT_PASSWORD` and `MYSQL_PASSWORD` variables.

   > ⚠ Important
   > Leave the `MYSQL_DATABASE` and `MYSQL_USERNAME` variables unchanged.

Step 2: Deploy the Qumulo Broker API Server

This section explains how to deploy Qumulo Broker on a standalone machine or virtual machine.

1. Navigate to the `/opt/qumulo/QumuloBroker/api/` directory on your Qumulo Broker machine.
2. Ensure that Docker and Docker Compose are installed on your Qumulo Broker machine.

   ```
   docker version
   docker compose version
   ```

3. To start the Docker containers, run the `docker compose up -d` command.

   The command creates the network and containers. The following is example output.

```
⠿ Network api_qumulo-net  Created      0.1s
⠿ Container api-db-1      Started      1.2s
⠿ Container api-web-1     Started      1.3s
⠿ Container api-proxy-1   Started      1.6s
```

4. To view the status of running containers, run the `docker ps` command.

   The following is example output.

```
CONTAINER ID   IMAGE          COMMAND                  CREATED
1a234567b089   nginx:latest   "/docker-entrypoint.…"   6 seconds ago
a1234567bcde   api-web        "app/main.py"            6 seconds ago
123ab45678cd   mariadb        "docker-entrypoint.s…"   7 seconds ago

STATUS         PORTS                                              NAMES
Up 4 seconds   80/tcp, 0.0.0.0:443->443/tcp, :::443->443/tcp      api-proxy-1
Up 5 seconds   0.0.0.0:8000->8000/tcp, :::8000->8000/tcp          api-web-1
Up 5 seconds   0.0.0.0:3306->3306/tcp, :::3306->3306/tcp          api-db-1
```

5. To view the logs of a specific container, run the `docker logs <container-id>` command.

You can now configure Varonis to communicate with your Qumulo cluster.

Step 3: Configure Qumulo Audit Logging by Using the qq CLI

This section explains how to configure audit logging on your Qumulo cluster.

1. To configure audit logging on your Qumulo cluster, run the `qq audit_set_syslog_config` command with the `--enable` flag, use the `--json` flag to request logging in JSON format, and specify the IP address or hostname and port number for your Qumulo Broker machine. For example:

```
qq audit_set_syslog_config \
  --enable \
  --json \
  --server-address 203.0.113.1 \
  --server-port 514
```

   Qumulo Core enables audit logging for your cluster.

2. To confirm the audit logging configuration for your cluster, run the `qq audit_get_syslog_config` command.

In the following example output, audit logging is enabled in JSON format.

```
{
  "enabled": true,
  "format": "json",
  "local_enabled": false,
  "server_address": "203.0.113.1",
  "server_port": 514
}
```

3. To confirm the connection between the Qumulo Broker and the rsyslog instance, run the `qq audit_get_syslog_status` command.

   The command returns one of three possible values for the `connection_status` field:

- `AUDIT_LOG_CONNECTED`: The rsyslog instance is connected to your Qumulo Broker machine and all audit log messages are being transferred correctly.
- `AUDIT_LOG_DISCONNECTED`: The rsyslog instance is disconnected from your Qumulo Broker. Your Qumulo cluster is configured to buffer all outgoing audit log messages until it fills its buffer. When the rsyslog instance reconnects to your Qumulo Broker the cluster attempts to send all buffered messages.

  > **Caution**
  >
  > When the message buffer fills up, Qumulo Core discards all new messages. To change the buffer size, configure rsyslog parameters.
  >
  > If a power outage or cluster reboot occurs while Qumulo Core is waiting to send its messages, all unsent messages are lost.

- `AUDIT_LOG_DISABLED`: Audit logging has been disabled explicitly for this Qumulo cluster.

Step 4: Configure rsyslog to Communicate with Qumulo Broker

This section explains how to configure rsyslog on the Qumulo Broker machine.

> **Important**
> Before you restart the rsyslog service to apply a new configuration, you must always ensure that Qumulo Broker is deployed (page 7).

1. Change the file permission of the Qumulo Broker binary file.

```
chmod a+x /opt/qumulo/QumuloBroker/events/Broker
```

Configuring rsyslog to Communicate with a Single Cluster

Configure the following rsyslog parameters in the `/etc/rsyslog.d/10-qumulo.conf` file.

The following complete, annotated configuration file lets rsyslog on the Qumulo Broker machine communicate with a single Qumulo cluster.

```
# PARSE AND PUBLISH QUMULO AUDIT LOGS

# TCP connection for receiving audit logs
module(load="imtcp")
input(type="imtcp" port="514")

# To let rsyslog use standard input (to pass messages to an external
# script that parses and performs custom processing on audit log data),
# load the omprog syslog module.
module(load="omprog")

if ($app-name startswith "qumulo") then {
  # If the log show an issue related to audit log operations, uncomment
  # the following line and restart the resyslog service.
  # action(type="omfile" file="/var/log/qumulo_audit.log")
  action(
    # Invoke the omprog module
    type="omprog"
    name="QumuloLog"

    # The full path and any CLI parameters for the external script
    binary="/opt/qumulo/QumuloBroker/events/Broker"

    # The queue type to use
    queue.type="LinkedList"

    # The maximum queue size (100,000 messages)
    # Tip: To configure rsyslog to communicate with multiple Qumulo
    #      clusters, set this value to 200,000.
    queue.size="100000"

    # When enabled, the system saves data while shutting down
    queue.saveOnShutdown="on"

    # The maximum number of worker threads that can run in parallel
    # Tip: To configure rsyslog to communicate with multiple Qumulo
    #      clusters, set this value to 16.
    queue.workerThreads="8"

    # The number of messages that a worker thread processes before
    # rsyslog creates another worker thread. For example, if you set
    # queue.workerThreads to 200 and there are 201 messages in the
    # queue, rsyslog creates a second worker thread.
    # Note: The queue.workerThreads parameter limits the maximum
    #       value of the queue.size parameter.
    queue.workerThreadMinimumMessages="10000"
```

```
    # The interval after which the system retries the action, 30
    # seconds by default. If multiple retries fail, in order to prevent
    # the excessive resource use, the system extends the interval
    # automatically by using a specific formula.
    # Note: The suspension interval increases as the number of
    #       retries increases.
    action.resumeInterval="10"

    # The location where the system stores the output of the
    # publisher script for system troubleshooting.
    output="/var/log/varonis_publisher.log"
  )
  stop
} else
  action(type="omfile" file="/dev/null")
```

Configuring rsyslog to Communicate with Multiple Clusters

The `queue.size` and `queue.workerThreads` rsyslog parameters in the `/etc/rsyslog.d/10-qumulo.conf` file (page 10) let rsyslog on the Qumulo Broker machine communicate with multiple Qumulo clusters.

- `queue.size`: Set the maximum size of the queue to 200,000 messages
- `queue.workerThreads`: Set the maximum number of worker threads that can run in parallel to 16 threads

To restart the rsyslog service, run the `systemctl restart rsyslog` command.

Troubleshooting the Integration between Qumulo Core and Varonis

This section explains how to troubleshoot the integration between Qumulo Core and Varonis.

To Troubleshoot Qumulo Broker

1. Do one of the following:

- View Qumulo Broker operation logs in the `/var/log/qumulo_audit.log` file.
- View the logs for each container by using the `docker logs <container-id>` command.

2. If the logs show an issue related to audit log operations, uncomment the following line in the `/etc/rsyslog/10-qumulo.conf` file.

```
# action(type="omfile" file="/var/log/qumulo_audit.log")
```

3. To restart the rsyslog service, run the `systemctl restart rsyslog` command.
4. Get the input log that you suspect to cause an issue from the `/var/log/qumulo_audit.log` file.

```
Mar  3 14:08:51 q-varonis-1 qumulo
{
  "user_id": {
    "sid": "S-1-5-21-123456790-1234567890-1234567890-123",
    "auth_id": "500",
    "name": "admin"
  },
  "user_ip": "203.0.113.0,
  "protocol": "smb2",
  "operation": "fs_create_file",
  "status": "ok",
  "details": {
    "file_id": "1000003",
    "path": "/my-file.txt"
  }
}
```

5. Use the input log from the from the `/var/log/qumulo_audit.log` file to run the `/opt/qumulo/QumuloBroker/events/Broker` command manually.

> **Note**
> Change the timestamp definition in your input to ISO 8601 with milliseconds.

```
2023-03-03T14:08:51.058379Z q-varonis-1 qumulo
{
  "user_id": {
    "sid": "S-1-5-21-123456790-1234567890-1234567890-123",
    "auth_id": "500",
    "name": "admin"
  },
  "user_ip": "203.0.113.0",
  "protocol": "smb2",
  "operation": "fs_create_file",
  "status": "ok",
  "details": {
  "file_id": "1000003",
  "path": "/my-file.txt"
  }
}
```

6. For questions about any issues, contact the Qumulo Care team.