IBM Storage Ceph 7.1

# *NVMe-oF Gateway Support Guide*
*2024-05-16*

# Important

**DRAFT BETA PROVIDED WITH NO SUPPORT:** Do not upgrade a production installation to a beta release or from a beta release. Upgrade support is not provided between beta releases. No automated upgrade path is provided from a beta release to the final GA release. The cluster will need to be rebuilt for production use.

**Attention:** When copying and pasting commands from the beta PDF, hyphens at the end of the lines get removed and the content no longer pastes as expected, creating broken commands and links. This specifically occurs in longer commands, such as the **`curl https://public.dhe.ibm.com/ibmdl/export/pub/storage/ceph/beta/ibm-storage-ceph-7-beta-rhel-9.repo`** command. Where the `ibm-storage` copies and pastes as `ibmstorage`.

Be sure that when copying and pasting commands, they paste as is written in this document, and add in any necessary hyphens.

## Known issues

**Host may not be able to see namespaces, due to multiple entries for the same NVMe-oF gateway host**

Multiple entries for the same NVMe-oF gateway host are found in the NVMe-oF gateway monitor. This can occur if the gateway deployment fails when downloading the gateway docker image.

When this occurs, the namespaces can be left unassigned to any of the optimized ANA groups, resulting in the host not being able to see the namespaces.

If this occurs, contact development to clean up the monitor database.

(22793520

**Host may not be able to see namespaces, due to incomplete cleanup of the NVMe-oF gateway monitor database**

The NVMe-oF gateway monitor database does not get properly cleaned after removing the NVMe-oF service.

When this occurs, the namespaces can be left unassigned to any of the optimized ANA groups, resulting in the host not being able to see the namespaces.

If this occurs, contact development to clean up the monitor database.

(2279862

**In some cases, NVMe-oF gateway stays out of quorum during a failback scenario, when using high availability**

When using NVMe-oF gateway high availability, if a gateway is in the process of failback and starts another failback process shortly after, it goes down again.

When this occurs, the gateway might stay out of quorum.

If this occurs, contact development for recovery help.

(2280332)

Last updated: 2024-05-16

# Contents

# Figures

# Tables

# NVMe over Fabrics

Ceph Block Device now offers NVMe-oF gateway support. Non-Volatile Memory express (NVMe) and NVMe over Fabrics (NVMe-oF) protocols are designed specifically to enable the full capabilities of parallelism and performance with all-flash storage systems.

NVMe-oF enables the performance of direct attached all-flash storage along with the flexibility and total cost of ownership (TCO) savings of shared storage. Implementing NVMe-oF support in enterprise storage arrays allows the combination of the NVMe protocol performance with the rich feature set of modern storage arrays. Using NVMe protocol enables modern storage arrays to meet growing customer demands.

NVMe-oF uses the TCP protocol as a ubiquitous transport that does not require special network configuration, by using existing Ethernet connectivity. Ethernet speeds currently support up to 400 Gib per second and the use of Ethernet technology is significant in data centers.

For more information about Ceph Block Device NVMe-oF gateway support, see Ceph NVMe-oF gateway.

# Ceph NVMe-oF gateway

Storage administrators can install and configure an NVMe over Fabrics (NVMe-oF) gateway for an IBM Storage Ceph cluster. With the Ceph NVMe-oF gateway, you can effectively run a fully integrated block storage infrastructure with all features and benefits of a conventional Storage Area Network (SAN).

**Note:** The NVMe gateway supports VMware vSphere APIs (VAAI), which includes support for vMotion, compare and write, unmap, and write zero.

Block-level access to a Ceph storage cluster used to be limited to QEMU and `librbd`. Block-level access to the Ceph storage cluster can now take advantage of the NVMe-oF standard to provide data storage. In IBM Storage Ceph 7.1, significant enhancements were made to the Ceph NVMe-oF gateway, specifically to the CLI commands and the addition of High Availability.

The NVMe-oF gateway integrates IBM Storage Ceph with the NVMe over TCP (NVMe/TCP) protocol to provide an NVMe/TCP target that exports RADOS Block Device (RBD) images. The NVMe/TCP protocol allows clients, which are known as *initiators*, to send NVMe-oF commands to storage devices, which are known as *targets*, over an Internet Protocol network. Initiators can be Linux clients, VMWare clients, or both. For VMWare clients, the NVMe/TCP volumes are shown as VMFS Datastore and for Linux clients, the NVMe/TCP volumes are shown as block devices.

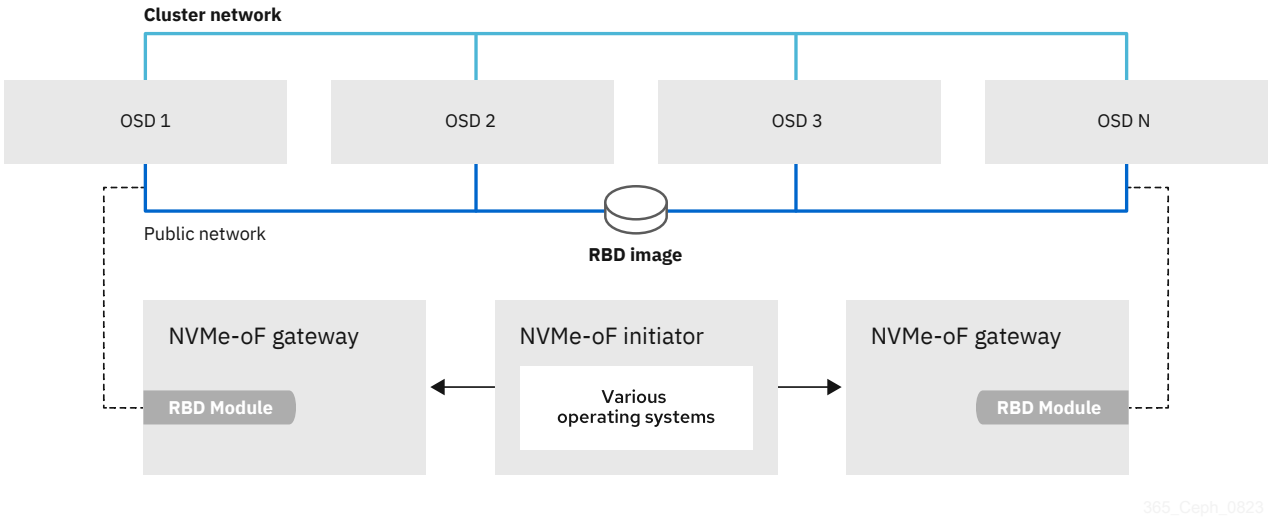


*Figure 1. Ceph NVMe-oF gateway*

For more information about the NVMe over Fabrics (NVMe-oF) protocol, see NVMe over Fabrics.

### High Availability with NVMe-oF gateway

*High Availability* (HA) provides I/O and control path redundancies for the host *initiators*. High Availability is also sometimes referred to as *failover and failback support*. The redundancy that HA creates is critical to protect against one or more gateway failures. With HA, the host can continue the I/O with only the possibility of performance latency until the failed gateways are back and functioning correctly.

NVMe-oF gateways are virtually grouped into *gateway groups* and the HA domain sits within the gateway group. An NVMe-oF gateway group supports four gateways. Each NVMe-oF gateway in the gateway group can be used as a path to any of the subsystems or namespaces that are defined in that gateway group. HA is effective with two or more gateways in a gateway group.

It is important to create redundancy between the host and the gateways. To create a fully redundant network connectivity, be sure that the host has two Ethernet ports that are connected to the gateways over a network with redundancy (for example, two network switches).

The HA feature uses the *Active/Standby* approach for each namespace. Using Active/Standby means that at any point in time, only one of the NVMe-oF gateways serve I/O from the host to a specific namespace. To properly use all NVMe-oF gateways, each namespace is assigned to a different load-balancing group. The number of load-balancing groups is equal to the number of NVMe-oF gateways in the gateway group.

With HA, if an NVMe-oF gateway fails, the initiator continues trying to connect. The amount of time that it tries to connect for depends on what is defined for the initiator. For more information about defining the reconnect time for the initiator and general configuration instructions, see “Configuring the NVMe-oF gateway initiator” on page 25.

### Scaling-out with NVMe-oF gateway

The NVMe-oF gateway supports scale-out. NVMe-oF gateway scale-out supports:

- One NVMe-oF gateway group.
- Up to four NVMe-oF gateways in a gateway group.
- Up to 16 NVMe-oF subsystems within a Ceph cluster.
- Up to 32 hosts per NVMe-oF subsystem.
- 400 namespaces per Ceph cluster.

### NVMe Discovery

The IBM Storage Ceph NVMe-oF gateway also supports *NVMe Discovery*. Each NVMe-oF gateway that runs in the Ceph cluster also runs a Discovery Controller. The Discovery Controller reports all IP addresses of each of the gateways in the group that are defined with listeners.

For configuring information, see “Configuring the NVMe-oF gateway initiator” on page 25.

## Networking requirements

NVMe over Fabrics with the TCP protocol requires sizing of network bandwidth and latency.

*Table 1. Networking requirements for NVMe-oF gateway*

| Requirement type | Description |
|---|---|
| *Minimum* - a single 10 Gb Ethernet link in the Ceph public network | The public network is used for client traffic (NVMe/TCP) and for communication with the Ceph Monitors. |

| *Table 1. Networking requirements for NVMe-oF gateway (continued)* | |
|---|---|
| **Requirement type** | **Description** |
| 25 or 100 Gb links in the public network | Use for environments where high amounts of IOPS and/or low latency is required.<br>**Note:** By default, the NVMe/TCP traffic is on the same network as the `librbd` traffic to the OSDs. For best performance, have the link over a dedicated client network for NVMe initiator to target gateway traffic. |

# Installing the NVMe-oF gateway

Before you can utilize the benefits of the Ceph NVMe-oF gateway, you must install the required software packages. You can install the Ceph NVMe-oF gateway by using the command-line interface.

Each NVMe-oF gateway runs the Storage Performance Development Kit (SPDK) to provide NVMe-oF protocol support. SPDK utilizes a user-space implementation of the NVMe-oF protocol to interact with the Ceph `librbd` library to expose RBD images to NVMe-oF clients. With the Ceph NVMe-oF gateway you can effectively run a fully integrated block storage infrastructure with all features and benefits of a conventional Storage Area Network (SAN).

## Installing the Ceph NVMe-oF gateway with the command-line interface

The Ceph NVMe-oF gateway is the NVMe-oF target node and also a Ceph client node. The Ceph NVMe-oF gateway can be a stand-alone node or be collocated on a Ceph Object Storage Daemon (OSD) node.

### Before you begin

Installing the NVMe-oF gateway requires a storage system that is running IBM Storage Ceph 7.1 or later cluster.

### Procedure

Complete the following steps to install the Ceph NVMe-oF gateway.

1. Create a pool in which the gateway group configuration is to be managed.

   ```
   ceph osd pool create NVME-OF_POOL_NAME
   ```

   For example,

   ```
   [root@host01 ~]# ceph osd pool create nvmeof_pool01
   ```

2. Enable the RADOS Block Device (RBD) application on the NVMe-oF pool.

   ```
   rbd pool init NVME-OF_POOL_NAME
   ```

   For example,

   ```
   [root@host01 ~]# rbd pool init nvmeof_pool01
   ```

3. Set the `mgr` with the NVMe-oF container image.

   ```
   podman pull icr.io/ibm-ceph-beta/nvmeof-rhel9:latest
   ceph config set mgr mgr/cephadm/NVME_OF_CONTAINER_IMAGE icr.io/ibm-ceph-beta/nvmeof-
   rhel9:latest
   ```

   For example,

```
[root@host01 ~]# podman pull icr.io/ibm-ceph-beta/nvmeof-rhel9:latest

[root@host01 ~]#ceph config set mgr mgr/cephadm/container_image_nvmeof icr.io/ibm-ceph-beta/
nvmeof-rhel9:latest
```

4. Deploy the `nvmeof` daemons that use placement specification on a specific set of nodes.

   **Important:** Deploy a minimum of two gateways to be able to use High Availability.

   ```
   ceph orch apply nvmeof NVME-OF_POOL_NAME --placement="NODE1,NODE2,..."
   ```

   For example,

   ```
   [ceph: root@host01 /]# ceph orch apply nvmeof nvmeof_pool01 --placement="host01,host02"
   ```

   **Note:** To remove a node from a daemon, redeploy the `nvmeof` daemon on the wanted nodes.

5. Install NVMe daemons by using the service specification file with the `cpumask` option.

   Use this option to increase the number of cores, improving the NVMe performance.

   a) Create a specification file with the following details.

   ```
   service_type: nvmeof
   service_id: NVME-OF_POOL_NAME
   placement:
     hosts:
       - HOST_NAME
   spec:
     pool: NVME-OF_POOL_NAME
     tgt_cmd_extra_args: --cpumask=0xFF
   ```

   FF is a hexadecimal bit mask that is based on the number of cores that are used in the cluster.

   In the following example, 8 cores are used, so the value of `--cpumask` is `0xFF`. For 6 cores, the value is `0x3F`.

   ```
   [root@host01 ~]# cat file.yaml
   service_type: nvmeof
   service_id: nvmeof_pool01
   placement:
     hosts:
       - host01
       - host02
   spec:
     pool: nvmeof_pool01
     tgt_cmd_extra_args: --cpumask=0xFF
   ```

   b) Deploy the NVMe daemons on the admin node.

   ```
   ceph orch apply -i NVME_FILE
   ```

## Configuring mTLS authentication

Configure mutual TLS (mTLS) to ensure secure connections between the command-line interface gRPC client and the Ceph NVMe-oF gateway gRPC server.

### Before you begin

Before configuring mTLS, ensure that the following are on each Ceph node that the NVMe-oF gateway will run on.

- Server certificate and key files.
- Client certificate and key files.
- A running IBM Storage Ceph cluster.
- Ceph NVMe-oF gateway installed. For more information, see “Installing the Ceph NVMe-oF gateway with the command-line interface” on page 14.

### About this task

Configuring mTLS authentication is not mandatory but provides secure connectivity.

Use the following procedures to configure mTLS authentication:

1. “Configuring mTLS on the NVMe-oF server” on page 16
2. “Configuring mTLS on the NVMe-oF client” on page 18

The mTLS configuration can be disabled at any time. To disable, see “Disabling the mTLS secure connection” on page 19.

## Configuring mTLS on the NVMe-oF server

### About this task

Start the mTLS configuration setup from the NVMe-oF server side.

### Procedure

1. Export the NVMe-oF gateway specification configuration to a YAML file.

   ```
   ceph orch ls nvmeof --export > gw-conf.yaml
   ```

   For example,

   ```
   [ceph: root@host01 /]# ceph orch ls nvmeof --export > gw-conf.yaml
   ```

   The following is an example for an NVMe-oF gateway specification configuration file.

```
service_type: nvmeof
    service_id: nvmeof-pool
    service_name: nvmeof.nvmeof-pool
    placement:
    hosts:
    - a-rhel9-node2
    spec:
    bdevs_per_cluster: 32
    client_cert: ./client.crt
    client_key: ./client.key
    conn_retries: 10
    discovery_port: 8009
    enable_prometheus_exporter: true
    log_directory: /var/log/ceph/
    log_files_enabled: true
    log_files_rotation_enabled: true
    log_level: INFO
    max_log_directory_backups: 10
    max_log_file_size_in_mb: 10
    max_log_files_count: 20
    monitor_timeout: 1.0
    omap_file_lock_duration: 60
    omap_file_lock_retries: 15
    omap_file_lock_retry_sleep_interval: 5
    omap_file_update_reloads: 10
    pool: nvmeof-pool
    port: 5500
    rpc_socket_dir: /var/tmp/
    rpc_socket_name: spdk.sock
    server_cert: ./server.crt
    server_key: ./server.key
    spdk_log_level: WARNING
    spdk_path: /usr/local/bin/nvmf_tgt
    spdk_timeout: 60.0
    state_update_interval_sec: 5
    state_update_notify: true
    tgt_path: /usr/local/bin/nvmf_tgt
    transport_tcp_options:
        in_capsule_data_size: 8192
        max_io_qpairs_per_ctrlr: 7
    transports: tcp
    verbose_log_messages: true
    verify_nqns: true
```

2. Get the Ceph cluster ID by using the **ceph -s** command.
   For example,

```
[ceph: root@host01 /]# ceph -s
        cluster:
            id:     cd3bfe3e-f11e-11ee-ac41-005056b4ea89
            health: HEALTH_OK

        services:
            mon: 3 daemons, quorum a-rhel9-node1,a-rhel9-node2,a-rhel9-node3 (age 73m)
            mgr: a-rhel9-node1.ejdcun(active, since 42h), standbys: a-rhel9-node2.nhveiq
            osd: 9 osds: 9 up (since 42h), 9 in (since 42h)

        data:
            pools:   2 pools, 33 pgs
            objects: 4 objects, 449 KiB
            usage:   243 MiB used, 576 GiB / 576 GiB avail
            pgs:     33 active+clean

        io:
            client:   682 B/s rd, 0 op/s rd, 0 op/s wr
```

3. Create an `mtls` folder on each Ceph node that the NVMe-oF gateway is to be placed on.

```
mkdir /var/lib/ceph/CLUSTER_ID/mtls
```

   For example,

```
[ceph: root@host01 /]# mkdir /var/lib/ceph/cd3bfe3e-f11e-11ee-ac41-005056b4ea89/mtls
```

4. Copy the server certificate, client certificate, and server key files to the new `mtls` folder.

```
cp SERVER_CERTIFICATE.CRT /var/lib/ceph/CLUSTER_ID/mtls/
cp SERVER_KEY.key /var/lib/ceph/CLUSTER_ID/mtls/
cp CLIENT_CERTIFICATE.crt /var/lib/ceph/CLUSTER_ID/mtls/
```

**Note:** For details on obtaining the certificate information, see **Creating and managing TLS keys and certificates** within *Securing networks* in the Red Hat Enterprise Linux documentation.

5. Copy the NVMe-oF exported specification file to a new mTLS file.

```
cp gw-conf.yaml mTLS_SPECIFICATION_FILE.yaml
```

For example,

```
[ceph: root@host01 /]# cp gw-conf.yaml gw-conf-with-mtls.yaml
```

6. Edit the NVMe-oF mTLS specification file.

Add the following values into the spec section of the specification file.

```
enable_auth: true
client_cert: /src/mtls/CLIENT_CERTIFICATE.crt
server_cert: /src/mtls/SERVER_CERTIFICATE.crt
server_key: /src/mtls/SERVER_KEY.key
```

For example,

```
vi gw-conf-with-mtls.yaml

service_type: nvmeof
   service_id: nvmeof-pool
   service_name: nvmeof.nvmeof-pool
   placement:
     hosts:
     - a-rhel9-node2
   spec:
     pool: nvmeof-pool
     enable_auth: true
     client_cert: /src/mtls/client.crt
     server_cert: /src/mtls/server.crt
     server_key: /src/mtls/server.key
```

7. Apply the specification file changes.

```
ceph orch apply -i mTLS_SPECIFICATION_FILE.yaml
```

For example,

```
[ceph: root@host01 /]# ceph orch apply -i gw-conf-with-mtls.yaml
Scheduled nvmeof.nvmeof-pool update...
```

8. Reconfigure the NVMe-oF service.

```
ceph orch reconfig nvmeof.POOL_NAME
```

For example,

```
[ceph: root@host01 /]# ceph orch reconfig nvmeof.nvmeof-pool
Scheduled to reconfig nvmeof.nvmeof-pool.a-rhel9-node2.blhrht on host 'a-rhel9-node2'
```

## Configuring mTLS on the NVMe-oF client

### About this task

After the NVMe-oF server is configured, configure mTLS on the NVMe-oF client side.

To configure the NVMe-oF client for mTLS, use the following command, as either **docker run** or **podman run**.

```
podman run --add-host=HOST_NAME -v
PATH_TO_LOCAL_SERVER_CERTIFICATE:MOUNTED_LOCATION_OF_SERVER_CERTIFICATE_FILE:z \
-v PATH_TO_LOCAL_CLIENT_CERTIFICATE:MOUNTED_LOCATION_OF_CLIENT_CERTIFICATE_FILE:z \
-v PATH_TO_LOCAL_CLIENT_KEY:MOUNTED_LOCATION_OF_CLIENT_KEY:z -it \
--rm NVME-OF_CLI_CONTAINER_IMAGE:NVME-OF_CLI_CONTAINER_IMAGE_TAG --server-address NVME-
OF_SERVER_HOSTNAME \
--client-key MOUNTED_LOCATION_IN_CONTAINER_OF_CLIENT_KEY --client-cert
MOUNTED_LOCATION_IN_CONTAINER_OF_CLIENT_CERTIFICATE \
--server-cert MOUNTED_LOCATION_IN_CONTAINER_OF_SERVER_CERTIFICATE gw info
```

- The **--add-host** parameter is only needed if you need to map an IP to a hostname.
- The **--server-address** parameter is either the NVMe-oF server hostname or the server IP address.

**Example**

```
podman run --add-host=a-rhel9-node2:10.10.10.101 -v /tmp/server.crt:/root/server.crt:z \
-v /tmp/client.crt:/root/client.crt:z -v /tmp/client.key:/root/client.key:z -it --rm icr.io/ibm-
ceph-beta/nvmeof-cli-rhel9:1.1.0 \
--server-address a-rhel9-node2 --client-key /root/client.key --client-cert /root/client.crt --
server-cert /root/server.crt gw info

Enable server auth since both --client-key and --client-cert are provided
CLI's version: 1.1.0
Gateway's version: 1.1.0
Gateway's name: client.nvmeof.nvmeof-pool.a-rhel9-node2.jilauu
Gateway's host name: a-rhel9-node2
Gateway's load balancing group: 1
Gateway's address: 10.10.10.101
Gateway's port: 5500
SPDK version: 23.01.1
```

## Disabling the mTLS secure connection

### About this task

To disable the mTLS secure connection to the server, the NVMe-oF gateway specification file needs to be reconfigured.

### Procedure

1. Edit the NVMe-oF mTLS specification file.
   a) Change the **enable_auth** value to `false`.
   b) From the `spec` section of the file, remove the **client_cert**, **server_cert**, and **server_key** lines.

   For example,

```
vi gw-conf-with-mtls.yaml

service_type: nvmeof
   service_id: nvmeof-pool
   service_name: nvmeof.nvmeof-pool
   placement:
     hosts:
     - a-rhel9-node2
   spec:
     pool: nvmeof-pool
```

2. Apply the specification file changes.

```
ceph orch apply -i mTLS_SPECIFICATION_FILE.yaml
```

   For example,

```
[ceph: root@host01 /]# ceph orch apply -i gw-conf-with-mtls.yaml
Scheduled nvmeof.nvmeof-pool update...
```

3. Reconfigure the NVMe-oF service.

```
ceph orch reconfig nvmeof.POOL_NAME
```

For example,

```
[ceph: root@host01 /]# ceph orch reconfig nvmeof.nvmeof-pool
Scheduled to reconfig nvmeof.nvmeof-pool.a-rhel9-node2.blhrht on host 'a-rhel9-node2'
```

The mTLS configuration is removed and the connection is insecure. The NVMe-oF command-line interface can now be used without the certificate and keys.

# Configuring the NVMe-oF gateway target

Configure targets, LUNs, and clients, by using the Ceph gateway **nvmeof-cli** command-line utility.

Configuring the NVMe-oF gateway target requires a storage system that is running IBM Storage Ceph 7.1 or later cluster and a running Ceph NVMe-oF gateway.

## Defining an NVMe-oF subsystem

Define an NVMe-oF subsystem. Defining the NVMe subsystem includes creating an NVMe subsystem, configuring the IP port for communication, and enabling hosts to use the subsystem.

### About this task

Configure the Ceph NVMe-oF gateway by using the gateway `nvmeof-cli` utility.

### Before you begin

The gateway `nvmeof-cli` container automatically downloads during initial container use on the host. Alternatively, the `nvmeof-cli` container can be downloaded before first use.

To download the gateway `nvmeof-cli` container, use the following command:

```
podman pull icr.io/ibm-ceph-beta/nvmeof-cli-rhel9:latest
```

**Note:** These commands can be run on either `podman` or `docker`.

### Procedure

1. Optional: Define an `nvmeof-cli` alias.

   This step is recommended but optional. Creating an alias enables streamlined commands.

   **Important:** If an alias is not created, be sure to replace `podman run --rm <icr.io/ibm-ceph-beta/nvmeof-cli-rhel9:latest>` at the beginning of each command, instead of `nvmeof-cli`.

   a) From the Linux command line, create the alias.

   ```
   alias nvmeof-cli="podman run --rm icr.io/ibm-ceph-beta/nvmeof-cli-rhel9:latest"
   ```

   By default, the command line output goes to `stderr`. To change the output placement, use the **`--output`** parameter. For example, **`--output stdio`**.

   **Note:** Changing the output placement to `stdio`, enables CLI calls to integrate more seamlessly into scripts and automation.

   b) Verify that the alias is successfully created.

   ```
   nvmeof-cli
   ```

When created successfully, the `nvmeof-cli` help menu is displayed.

For example,

```
[root@host01 ~]# alias nvmeof-cli="icr.io/ibm-ceph-beta/nvmeof-cli-rhel9:latest"
[root@host01 ~]# nvmeof-cli
usage: python3 -m control.cli [-h] [--format {text,json,yaml,plain,python}] [--output
{log,stdio}]
                              [--server-address SERVER_ADDRESS] [--server-port SERVER_PORT]
                              [--client-key CLIENT_KEY] [--client-cert CLIENT_CERT]
                              [--server-cert SERVER_CERT]

{version,gw,log_level,subsystem,listener,host,connection,namespace,ns}
                              ...

CLI to manage NVMe gateways

optional arguments:
  -h, --help            show this help message and exit
  --format {text,json,yaml,plain,python}
                        CLI output format
  --output {log,stdio}  CLI output method
  --server-address SERVER_ADDRESS
                        Server address
  --server-port SERVER_PORT
                        Server port
  --client-key CLIENT_KEY
                        Path to the client key file
  --client-cert CLIENT_CERT
                        Path to the client certificate file
  --server-cert SERVER_CERT
                        Path to the server certificate file

Commands:
  {version,gw,log_level,subsystem,listener,host,connection,namespace,ns}
    version             Get CLI version
    gw                  Gateway commands
    log_level           Log level commands
    subsystem           Subsystem commands
    listener            Listener commands
    host                Host commands
    connection          Connection commands
    namespace (ns)      Namespace commands
[root@host01 ~]#
```

2. Create an NVMe subsystem.

   **Note:** Up to 16 subsystems are supported.

   ```
   nvmeof-cli --server-address GATEWAY_IP subsystem add --subsystem SUBSYSTEM_NQN [--serial-
   number SERIAL_NUMBER] [--max-namespaces MAX_NAMESPACES]
   ```

   The *SUBSYSTEM_NQN* is a user-defined string. In this example it is defined as `nqn.2016-06.io.spdk:cnode1`.

   **Important:** NQN and gateway name values must not contain an underscore character.

   For example,

   ```
   [root@host01 ~]# nvmeof-cli --server-address 10.172.19.21 subsystem add --subsystem subnqn
   nqn.2016-06.io.spdk:cnode1
   Adding subsystem nqn.2016-06.io.spdk:cnode1: Successful
   ```

   For extra input, the following parameters can be added to the command:

   **`--serial-number`**
   If the serial number is not specified, the number is randomly generated.

   Input type: String

   **`--max-namespaces`**
   The maximum namespaces per subsystem default is 256.

   Input type: Number

3. Define the IP port on the gateway that is to process NVMe/TCP commands and I/O operations.

   **Important:** This step must be repeated on all gateway nodes with the appropriate Gateway IP address for High Availability to work. Listeners must be created on all gateway nodes (1–4) and for all subsystems (1–16).

   ```
   nvmeof-cli --server-address GATEWAY_IP listener add --subsystem SUBSYSTEM_NQN --host-name
   HOST_NAME --traddr GATEWAY_NODE_IP
   ```

   **Important:** NQN and gateway name values must not contain an underscore character.

   For example,

   ```
   [root@host01 ~]# nvmeof-cli --server-address 10.172.19.01 listener add --subsystem
   nqn.2016-06.io.spdk:cnode1 --host-name host02 --traddr 10.172.19.01
   ```

   To define the IP port on the gateway, be sure to include the **--host-name** and **--traddr** parameters.

   For extra input, the following parameters can optionally be added to the command:

   - **--trsvcid**

     Input the port number. The default port is 4420.
   - **--ADRFAM**

     Input the *IP_TYPE*. The supported value is IPv4 and the input can be written as either `ipv4` or `IPV4`.
4. Get the host NVMe Qualified Name (NQN) for each host.

   **For Red Hat Enterprise Linux initiators**

   ```
   cat /etc/nvme/hostnqn
   ```

   For example,

   ```
   # cat /etc/nvme/hostnqn
   nqn.2014-08.org.nvmexpress:uuid:950ddadf-f995-47b7-9416-b9bb233f66e3
   ```

   In this example `950ddadf-f995-47b7-9416-b9bb233f66e3` is the UUID.

   **For VMware ESXi initiators**

   ```
   esxcli nvme info get
   ```

   For example,

   ```
   esxcli nvme info get
   Host NQN: nqn.2014-08.com.ibm.ceph:nvme:host01
   ```
5. Allow the NVMe initiator host to run NVMe/TCP commands to the newly created NVMe subsystem.

   **Note:** Specific hosts can be selected, as in the following example. To specify all hosts, use `--host "*"`.

   ```
   nvmeof-cli --server-address NODE_IP host add --subsystem SUBSYSTEM_NQN --host "HOST01_NQN,
   HOST02_NQN"
   ```

   **Important:** NQN and gateway name values must not contain an underscore character.

   For example,

   ```
   [root@host01 ~]# nvmeof-cli --server-address 10.172.19.01 host add --subsystem
   nqn.2016-06.io.spdk:cnode1 --host "nqn.2014-08.org.nvmexpress:uuid:950ddadf-f995-47b7-9416-
   b9bb233f66e3,nqn.2014-010.org.nvmexpress:uuid:960ddadf-f995-47b7-9416-b9bb233f66e3"
   ```
6. Verify by listing all NVMe-oF Gateway target entities configured on the gateway.

   a) List the subsystems.

```
nvmeof-cli --server-address GATEWAY_IP subsystem list
```

For example,

```
[root@host01 ~]# nvmeof-cli --server-address 10.172.19.01 subsystem list
Namespaces in subsystem nqn.2016-06.io.spdk:cnode1:
Subtype| NQN                         | HA State | Serial            | Controller IDs |
Namespace | Max
       |                             |          | Number            |                |
Count     | Namespaces
================================================================================================
=================
NVMe   | nqn.2016-06.10.spdk:cnode1 | enabled  | Ceph1230985676540 | 4081-6120      |
32        | 256
------------------------------------------------------------------------------------------------
-----------------
NVMe   | nqn.2016-06.10.spdk:cnode2 | enabled  | Ceph5400985676123 | 4081-6120      |
32        | 256
```

b) Verify the listeners defined on a subsystem.

**Note:** Run this command for each subsystem.

```
nvmeof-cli listener list -n SUBSYSTEM_NQN
```

The following is an example output for two gateways on a subsystem.

```
[root@host01 ~]# nvmeof-cli listener list -n nqn.2016-06.io.spdk:cnode1

Listeners for nqn.2016-06.io.spdk:cnode1:
Host   | Transport | Address | Address
       |           | Family  |
================================================
host03 | TCP       | IPv4    | 10.8.128.223:4420
------------------------------------------------
host04 | TCP       | IPv4    | 10.8.128.223:4420
```

# Defining block devices to use NVMe/TCP

Define storage block devices to be used with NVMe/TCP.

## About this task

**Note:** Run this procedure only one time per block device, even if multiple gateways exist.

## Procedure

1. Create an rbd pool.

```
ceph osd pool create POOL_NAME PG_NUM
ceph osd pool application enable POOL_NAME rbd
rbd pool init -p POOL_NAME
```

For example,

```
[root@rbd-client ~]# ceph osd pool create pool1
[root@rbd-client ~]# ceph osd pool application enable pool1 rbd
[root@rbd-client ~]# rbd pool init -p pool1
```

2. Optional: Create an image within any RADOS Block Device (RBD) application-enabled pool.

**Note:** An image can be created separately, by using this step, or while adding a namespace. To create an image directly during the name space creation, skip this step and go to step “3” on page 24. Be sure to add the **--rbd-create-image** and **--size** parameters.

```
rbd create IMAGE_NAME --size MEGABYTES --pool POOL_NAME
```

For example,

```
[root@host01 ~]# rbd create image1 --size 1024 --pool pool1
```

For more information about creating an image, see Creating images.

3. Add a namespace to each block device that was created in “Defining an NVMe-oF subsystem” on page 20.

   To create an image within an RBD application-enabled pool, use the **`--rbd-create-image`** and **`--size`** parameters.

   ```
   nvmeof-cli namespace add --subsystem SUBSYSTEM_NQN --server-address NODE_IP --rbd-pool
   RBD_POOL --rbd-image RBD_IMAGE
   ```

   **Note:** The image size can be expanded at any time. For more information see, “Expanding a block device image” on page 24.

   - Example of adding a namespace to a block device with a previously created image.

     ```
     [root@host01 ~]# nvmeof-cli namespace add --subsystem nqn.2016-06.io.spdk:cnode1 --server-
     address 10.172.19.01 --rbd-pool pool1 --rbd-image image1
     ```

   - Example of creating an image and adding a namespace within the same command.

     ```
     [root@host01 ~]# nvmeof-cli namespace add --subsystem nqn.2016-06.io.spdk:cnode1 --server-
     address 10.172.19.01 --rbd-pool pool1 --rbd-create-image --rbd-image image1 --size 1024
     ```

## What to do next

Verify the block device configuration.

1. If an image was created when adding the namespace, verify the image create on the node that has IBM Storage Ceph CLI, such as the admin node or the client.

   ```
   rbd ls -p pool1
   ```

2. List all namespaces configured on a subsystem.

   ```
   nvmeof-cli namespace list -n SUBSYSTEM_NQN --server-address NODE_IP
   ```

   For example,

```
[root@host01 ~]# nvmeof-cli namespace list -n nqn.2016-06.io.spdk:cnode1 --server-address
10.172.19.01
Namespaces in subsystem nqn.2016-06.io.spdk:cnode1:
NSID| Bdev                     | RBD    | RBD     | Image     | Block | UUID                 |
Load            | R/W IOs     | R/W MB      | Read MBs    | Write MBs
    | Name                     | Pool   | Image   | Size      | Size  |                      |
Balancing Group | per second | per second | per second | per second
================================================================================================
================================================================
1   | bdev-32c94dd8-8754-      | pool1 | image1 | 1024 MiB | 512 B | 32c94dd8-8754-442f- |
2               | unlimited  | unlimited  | unlimited  | unlimited
    | 442f-a12c-123abc456789  |        |         |           |       | a12c-123abc456789
|                 |            |           |           |
------------------------------------------------------------------------------------------------
----------------------------------------------------------------
2   | bdev-45c94ee8-ab56-      | pool1 | image2 | 1024 MiB | 512 B | 45c94ee8-ab56-332f- |
1               | unlimited  | unlimited  | unlimited  | unlimited
    | 332f-a24c-abc123ae7890  |        |         |           |       | a24c-abc123ae7890
|                 |            |           |           |
```

# Expanding a block device image

A block device image can be expanded at any time. Use this information to expand a namespace.

### About this task

**Important:** A block device image can be expanded but not reduced.

### Procedure

1. List all namespaces configured on a subsystem.

   Use the NSID or the UUID output to specify the subsystem to be resized, in step “2” on page 25.

   ```
   nvmeof-cli namespace list -n SUBSYSTEM_NQN
   ```

   For example,

   ```
   [root@host01 ~]# nvmeof-cli namespace list -n nqn.2016-06.io.spdk:cnode1
   Namespaces in subsystem nqn.2016-06.io.spdk:cnode1:
   NSID| Bdev                  | RBD   | RBD    | Image    | Block | UUID                |
   Load            | R/W IOs    | R/W MB     | Read MBs   | Write MBs
       | Name                  | Pool  | Image  | Size     | Size  |                     |
   Balancing Group | per second | per second | per second | per second
   ================================================================================================
   ================================================================
   1   | bdev-32c94dd8-8754-     | pool1 | image1 | 1024 MiB | 512 B | 32c94dd8-8754-442f- |
   2               | unlimited  | unlimited  | unlimited  | unlimited
       | 442f-a12c-123abc456789  |       |        |          |       | a12c-123abc456789
   |                 |           |           |            |
   ------------------------------------------------------------------------------------------------
   ----------------------------------------------------------------
   2   | bdev-45c94ee8-ab56-     | pool1 | image2 | 1024 MiB | 512 B | 45c94ee8-ab56-332f- |
   1               | unlimited  | unlimited  | unlimited  | unlimited
       | 332f-a24c-abc123ae7890  |       |        |          |       | a24c-abc123ae7890
   |                 |           |           |            |
   ```

2. Expand the block device image.

   To use the **`namespace resize`** command, include either the **`--uuid`** or **`--nsid`** (namespace ID) parameter indicate the new **`--size`**.

   The size is indicated in bytes, by default. To indicate a different size type, specify the unit. The supported size units are: KB, KiB, MB, MiB, GB, GiB, TB, and TiB.

   ```
   nvmeof-cli namespace resize --subsystem SUBSYSTEM [--uuid UUID | --nsid NSID] --size SIZE
   ```

   For example,

   ```
   [root@host01 ~]# nvmeof-cli namespace resize --subsystem nqn.2016-06.io.spdk:cnode1 --nsid 2
   --size 1536
   ```

   In this example, the size of NSID 2 on the `nqn.2016-06.io.spdk:cnode1` subsystem is being increased to 1536 bytes.

# Configuring the NVMe-oF gateway initiator

Configure the initiator to allow the NVMe/TCP protocol to send NVMe-oF commands to targets over an Internet Protocol network.

The NVMe-oF gateway initiator can be configured on either of the following platform version:

- Red Hat Enterprise Linux 9.2 or later
- VMware vSphere Hypervisor (ESXi) 7.0U3 or later

## Configuring the NVMe-oF initiator for Red Hat Enterprise Linux

Configure the NVMe-oF initiator for Red Hat Enterprise Linux.

## Before you begin

- Red Hat Enterprise Linux 9.2 or later

## Procedure

1. Install the `nvme-cli`.

   ```
   yum install nvme-cli
   ```

2. Install the necessary NVMe packages.

   ```
   modprobe nvme-fabrics
   ```

3. Verify that the target is reachable from the initiator.

   ```
   nvme discover -t tcp -a GATEWAY_IP -s 8009
   ```

   The output provides the IP address of the NVMe-oF subsystems that can be connected from this initiator.

   For example,

   ```
   [root@host01 ~]# nvme discover -t tcp -a 10.172.19.01 -s 8009
   =====Discovery Log Entry 0======
   trtype:  tcp
   adrfam:  ipv4
   subtype: nvme subsystem
   treq:    not required
   portid:  0
   trsvcid: 4420
   subnqn:  nqn.2016-06.io.spdk:cnode1
   traddr:  9.147.168.14
   eflags:  none
   sectype: none
   =====Discovery Log Entry 1======
   trtype:  tcp
   adrfam:  ipv4
   subtype: nvme subsystem
   treq:    not required
   portid:  1
   trsvcid: 4420
   subnqn:  nqn.2016-06.io.spdk:cnode1
   traddr:  9.147.168.32
   eflags:  none
   sectype: none
   ```

4. Connect to the NVMe-oF target.

   Use the *SUBSYSTEM_NQN* that was defined in “Defining an NVMe-oF subsystem” on page 20.

   Using the **`connect-all`** command connects to all gateways in the group, establishing multipath connections.

   **Note:** Multipath connections are mandatory for High Availability with NVMe-oF gateway.

   Adding the `-l` flag enables the initiator to retry connecting to the gateway in cases where the gateway becomes temporarily unavailable.

   ```
   nvme connect-all -traddr=GATEWAY_IP --transport=tcp -l 1800
   ```

   For example,

   ```
   [root@host01 ~]# nvme connect-all -traddr=10.172.19.01 --transport=tcp -l 1800
   ```

   In this example, the `-l` flag is set to retry after 1800 seconds, if the first connection does not succeed.

## What to do next

Verify that the initiator is set up correctly.

1. List the NVMe-oF block devices.

```
nvme list
```

   For example,

```
[root@host01 ~]# nvme list
Node                  Generic           SN                   Model
Namespace Usage                      Format           FW Rev
--------------------  ----------------  -------------------  ----------------------
--------- -------------------------- ---------------- --------
/home/nvme01_node01     /home/ng1n1       SPDK00000000000001   SPDK bdev Controller
1          10,49  MB /  10,49  MB      4 KiB +  0 B   23.01
...
```

2. Verify that the initiator is connected to all NVMe-oF gateways and subsystems in the gateway group.

```
nvme list-subsys
```

   For example,

```
[root@init-nvme-vm5 ~]# nvme list-subsys
nvme-subsys5 - NQN=nqn.2016-06.io.spdk:cnode2
\
+- nvme5 tcp traddr=10.243.64.5,trsvcid=4420 live
+- nvme6 tcp traddr=10.243.64.10,trsvcid=4420 live
+- nvme7 tcp traddr=10.243.64.11,trsvcid=4420 live
+- nvme8 tcp traddr=10.243.64.12,trsvcid=4420 live
nvme-subsys1 - NQN=nqn.2016-06.io.spdk:cnode1
\
+- nvme1 tcp traddr=10.243.64.5,trsvcid=4420 live
```

3. Create a filesystem on the target of your choosing. Use the target path that was found in step “1” on page 27.

```
mkfs NVME_NODE_PATH
```

   For example,

```
[root@host01 ~]# mkfs /home/nvme01_node01
mke2fs 1.46.5 (20-Dec-2023)
Discarding device blocks: done
Creating filesystem with 2560 4k blocks and 2560 inodes

Allocating group tables: done
Writing inode tables: done
Writing superblocks and filesystem accounting information: done
```

4. Mount the NVMe node on the NVMe-oF directory.

   a. Mount NVMe-oF.

```
mkdir /mnt/nvmeof
```

      For example,

```
[root@host01 ~]# mkdir /mnt/nvmeof
```

   b. Mount the node on within the NVMe-oF directory.

```
mount NVME_NODE_PATH /mnt/nvmeof
```

      For example,

```
[root@host01 ~]# mount /home/nvme01_node01 /mnt/nvmeof
```

5. Using sudo commands, list mounted NVMe-oF files.

```
ls /mnt/nvmeof
```

For example,

```
$ ls /mnt/nvmeof
lost+found
```

6. Create a text file within the `mnt/nvmeof` directory.

   For example,

```
$ sudo bash -c "echo Hello NVMe-oF > /mnt/nvmeof/hello.txt"
```

7. Verify that the text file can now be reached.

   For example,

```
$ cat /mnt/nvmeof/hello.txt
Hello NVMe-oF
```

# Configuring the NVMe-oF initiator for VMware ESXi

Configure the NVMe-oF initiator for VMware vSphere Hypervisor (ESXi). You can set up a VMware ESXi host as a NVMe/TCP initiator.

## About this task

NVMe/TCP supports VMware vSphere Hypervisor (ESXi) 7.0U3 or later.

**Note:** The NVMe gateway supports VMware vSphere APIs (VAAI).

## Before you begin

- A VMware ESXi host that is running VMware vSphere Hypervisor (ESXi) 7.0U3 version or later.
- Ceph NVMe-oF gateway deployed.
- IBM Storage Ceph cluster and `ceph-nvmeof` configurations are ready and working correctly.
- A subsystem defined within the gateway. For more information, see "Defining an NVMe-oF subsystem" on page 20.
- NVMe/TCP adapter is configured.
  - Enabled NVMe/TCP on a physical network interface controller (NIC).

    ```
    esxcli nvme fabrics enable --protocol TCP --device vmnicN
    ```

    Replace *N* with the number of NIC.
  - Tag a VMkernel NIC to allow NVMe/TCP traffic.

    ```
    esxcli network ip interface tag add --interface-name vmkN --tagname NVMeTCP
    ```

    Replace *N* with the ID of the VMkernel.

## Procedure

Configuring the VMware ESXi host for NVMe/TCP transport includes discovering the NVMe/TCP targets and connecting to them.

1. List the NVMe-oF adapter.

```
esxcli nvme adapter list
```

For example,

```
[root@host01:~] esxcli nvme adapter list
Adapter  Adapter Qualified Name           Transport Type  Driver     Associated Devices
-------  -------------------------------- --------------  ---------  ------------------
vmhba64  aqn:nvmetcp:ac-1f-6b-0a-18-74-T  TCP             nvmetcp    vmnic0
```

2. Optional: Discover any NVMe-oF-gateway subsystems.

   **Note:** Use this command to discover a subsystem without connecting. To discover and connect, go to step “3” on page 29.

   ```
   esxcli nvme fabrics discover -a NVME_TCP_ADAPTER -i GATEWAY_IP -p 8009
   ```

   For example,

   ```
   [root@host01:~] esxcli nvme fabrics discover -a vmhba64 -i 10.0.211.196 -p 8009

   Transport Type Address Family Subsystem Type Controller ID Admin Queue Max Size Transport
   Address Transport Service ID Subsystem NQN              Connected
   -------------- -------------- -------------- ------------- -------------------
   ----------------- ------------------- ------------------------- ---------
   TCP            IPv4           NVM            65535         128
   10.0.211.196     8009                 nqn.2016-06.io.spdk:cnode1  false
   ```

3. Connect to the NVMe-oF gateway subsystem.

   This command discovers the NVMe-oF gateways in the gateway group and then connects to the gateways.

   ```
   esxcli nvme fabrics discover -a NVME_TCP_ADAPTER -i GATEWAY_IP -p 8009 -c
   ```

   **Note:** In cases that a specific connection is required without discovering all of the available gateways, run the **`nvme fabrics connect`** command.

   ```
   esxcli nvme fabrics connect -a NVME_TCP_ADAPTER -i GATEWAY_IP -s SUBSYSTEM_NQN -p 8009
   ```

   ```
   [root@host01:~] esxcli nvme fabrics connect -a vmhba64 -i 10.0.211.196 -s
   nqn.2016-06.io.spdk:cnode1 -p 8009
   ```

   To verify, use the **`nvme controller list`** command.

   ```
   esxcli nvme controller list |grep TCP
   ```

   Check that the new connection is listed and marked as `true`.

   For example,

   ```
   [root@host01:~] esxcli nvme fabrics discover -a vmhba64 -i 10.0.211.196 -p 8009 -c

   Transport Type Address Family Subsystem Type Controller ID Admin Queue Max Size Transport
   Address Transport Service ID Subsystem NQN              Connected
   -------------- -------------- -------------- ------------- -------------------
   ----------------- ------------------- ------------------------- ---------
   TCP            IPv4           NVM            65535         128
   10.0.211.196     8009                 nqn.2016-06.io.spdk:cnode1  true
   ```

4. List NVMe/TCP controller list.

   ```
   esxcli nvme controller list
   ```

```
[root@host01:~] esxcli nvme controller list
Name                                                                  Controller Number  Adapter
Transport Type  Is Online  Is VVOL
--------------------------------------------------------------------  -----------------  -------
-------------  ---------  -------
nqn.2014-08.org.nvmexpress.discovery#vmhba65#10.0.211.196:8009                  257  vmhba65
TCP                  true    false
nqn.2016-06.io.spdk:cnode1#vmhba65#10.0.211.196:4420                            258  vmhba65
TCP                  true    false                                                   301
vmhba64  TCP                  true
```

5. List NVMe-oF namespaces in the subsystem.

```
esxcli nvme namespace list
```

For example,

```
[root@host01:~] esxcli nvme namespace list
Name                                Controller Number  Namespace ID  Block Size  Capacity in MB
----------------------------------  -----------------  ------------  ----------  --------------
eui.0100000001000000e4d25c00001ae214                256             1         512
953869
eui.01abc123def456g7e4d25c00001ae214                301             1         512
500
eui.02abc123def456g7e4d25c00001ae215                301             2         512
500
eui.03abc123def456g7e4d25c00001ae216                301             3         512
500
```

### What to do next

Verify that the initiator is set up correctly.

1. From the vSphere Client, go to the ESXi host.
2. On the **Storage** page, go to the **Devices** tab.
3. Verify that the NVMe/TCP disks are listed in the table.

## NVMe-oF Gateway performance best practices

Use the NVMe-oF Gateway performance best practices to ensure that you are using the gateway to its fullest capacity.

For high IOPS requirements, use a dedicated host for the NVMe-oF Gateway.

1. Colocation with OSDs is possible, but the high CPU demand from the gateway might introduce CPU contention especially for write client workloads and when the node has multiple OSDs.
2. For dedicated gateways, the gateway needs a minimum of 16 cores. For higher performance requirements, use 30 or more cores.

### Rule of Thumb (RoT) sizing guide

The following RoT sizing is based on the mixed workload scenario at latencies less than or equal to 10 ms, with an environment based on 100 Gb networking, with 72 namespaces.

*Table 2. Rule of thumb sizing guide*

| **IOPS Goal** | **Number of reactor cores** | **Total CPU demand (cores)** | **`tgt_cmd_extra_rags` specification parameter** |
|---|---|---|---|
| Up to 100,000 | 1 | 16 | `--cpumask=0x1` |
| Up to 200,000 | 2 | 18 | `--cpumask=0x3` |
| Up to 250,000 | 4 | 22 | `--cpumask=0xF` (default) |

| *Table 2. Rule of thumb sizing guide (continued)* | | | |
|---|---|---|---|
| **IOPS Goal** | **Number of reactor cores** | **Total CPU demand (cores)** | **`tgt_cmd_extra_rags` specification parameter** |
| Up to 300,000 | 8 | 30 | `--cpumask=0xFF` |

## Modifying the cores

Use this procedure to modify the cores that are used by Storage Performance Development Kit (SPDK) to handle the I/O.

1. Fetch the specification file of the cluster.

   ```
   ceph orch ls --export > FILE.yaml
   ```

2. Modify the file to include or modify the **`tgt_cmd_extra_args`** parameter.

   For example, change from the default **`tgt_cmd_extra_args: --cpumask=0xF`** to **`tgt_cmd_extra_args: --cpumask=0xFF`**.

   ```
   service_type: nvmeof
   service_id: NVME-OF_POOL_NAME
   placement:
     hosts:
       - HOST_NAME
   spec:
     pool: NVME-OF_POOL_NAME
     tgt_cmd_extra_args: --cpumask=0xFF
   ```

   For example,

   ```
   [ceph: root@host01/ ]# cat file.yaml
   service_type: nvmeof
   service_id: nvmeof_pool01
   placement:
     hosts:
       - host01
       - host02
   spec:
     pool: nvmeof_pool01
     tgt_cmd_extra_args: --cpumask=0xFF
   ```

3. Redeploy the service.

   ```
   ceph orch apply -i FILE.yaml
   ```

4. Reconfigure the NVMe-oF gateway.

   ```
   ceph orch reconfig NVME-OF_SERVICE_ID
   ```

   For example,

   ```
   [ceph: root@host01/ ]# ceph orch reconfig nvmeof_pool01
   ```

## VMware ESX recommendations

1. Allocate more of the smaller datastores instead of a few large datastores. This approach does not carry capacity cost as Ceph has thin provisions by default and helps ensure that me Ceph I/O contexts are available for the gateway.
2. Consider isolating applications with high I/O demands to use a dedicated datastore to ensure adequate host adapter queue depth.

3. Use jumbo frames when possible for ESXi to Gateway network links. Using jumbo frames is beneficial for workloads that use larger I/O block sizes.

# Removing the NVMe-oF service

Remove the NVMe-oF service if no longer using the NVMe-oF gateway or before running a fresh installation.

## Before you begin

Verify that the volumes are stopped before the service is removed.

## About this task

The following are the three main steps to remove the NVMe-oF service:

1. Disconnect the NVMe-oF Gateway initiator.
2. Remove the NVMe-oF Gateway target.
3. Remove the NVMe-oF service.

## Procedure

1. Disconnect the NVMe-oF initiator from NVMe-oF target.

   **Red Hat Enterprise Linux**

   ```
   nvme disconnect-all
   ```

   **VMware ESX initiator**

   ```
   esxcli nvme disconnect -n CONTROLLER_NUMBER -a NVME_TCP_ADAPTER -s SUBSYSTEM_NQN
   ```

   Replace:

   ***CONTROLLER_NUMBER***
   With the number that is fetched from **`esxcli nvme namespace list`** command.

   ***NVME_TCP_ADAPTER* and *SUBSYSTEM_NQN***
   With values from the **`esxcli nvme fabrics connect`** command.

2. Remove the NVMe-oF Gateway target for each subsystem.

   **Note:** The `--force` flag is needed to delete subsystems that contain namespaces.

   ```
   nvmeof-cli --subsystem del SUBSYSTEM_NQN --force
   ```

   For example,

   ```
   [root@host01 ~]# nvmeof-cli --subsystem del nqn.2016-06.io.spdk:cnode1 --force
   ```

3. Remove the NVMe-oF service.

   a) On the Ceph admin node, remove the NVMe-oF service.

   ```
   ceph orch rm nvmeof.NVME-OF_POOL_NAME
   ```

   For example,

   ```
   [ceph: root@host01 /]# ceph orch rm nvmeof.nvmeof_pool01
   ```

   b) Check that the **`mon_allow_pool_delete`** parameter is set to `true`

   ```
   ceph config set mon mon_allow_pool_delete true
   ```

   c) On the Ceph admin node, remove the *NVME-OF_POOL* created to host NVMe-oF service.

```
ceph osd pool rm NVME-OF_POOL_NAME NVME-OF_POOL_NAME --yes-i-really-really-mean-it
```

**Note:** The pool name (*NVME-OF_POOL_NAME*) must be used twice in the command to remove the pool.

For example,

```
[ceph: root@host01 /]# ceph osd pool rm nvmeof_pool01 nvmeof_pool01 --yes-i-really-really-
mean-it
```