



Storage

Administrator's Command Line Guide

January 22, 2025

Virtuozzo International GmbH

Vordergasse 59

8200 Schaffhausen

Switzerland

Tel: + 41 52 632 0411

Fax: + 41 52 672 2010

<https://virtuozzo.com>

Copyright ©2016-2025 Virtuozzo International GmbH. All rights reserved.

This product is protected by United States and international copyright laws. The product's underlying technology, patents, and trademarks are listed at <https://www.virtuozzo.com/legal.html>.

Microsoft, Windows, Windows Server, Windows NT, Windows Vista, and MS-DOS are registered trademarks of Microsoft Corporation.

Apple, Mac, the Mac logo, Mac OS, iPad, iPhone, iPod touch, FaceTime HD camera and iSight are trademarks of Apple Inc., registered in the US and other countries.

Linux is a registered trademark of Linus Torvalds. All other marks and names mentioned herein may be trademarks of their respective owners.

Contents

1. Introduction	1
1.1 About This Guide	1
1.2 About Storage	2
2. Setting Up a Storage Cluster	3
2.1 Setup Overview	3
2.2 Configuring Cluster Discovery	4
2.2.1 Using DNS Records	5
2.2.1.1 Announcing the Information About MDS Servers	5
2.2.1.2 Discovering Cluster Names	6
2.2.2 Setting Up Zeroconf	7
2.2.3 Specifying MDS Servers Manually	7
2.3 Preparing Disks for Storage	8
2.3.1 Preparing Disks for Chunk Servers	8
2.3.2 Preparing SSDs for Write Journaling or Caching	9
2.4 Cache Configuration	10
2.4.1 Supported Device Types	10
2.4.2 Choosing a Cache Device	11
2.4.3 Provisioning Cache Devices	12
2.4.4 Journal Sizing	12
2.4.5 Cache Sizing	13
2.5 Setting Up the First Metadata Server	14
2.5.1 Stage 1: Preparing to Create the First MDS Server	14
2.5.2 Stage 2: Creating the First MDS Server	15
2.6 Setting Up Chunk Servers	16
2.6.1 Stage 1: Preparing to Create a Chunk Server	16
2.6.2 Stage 2: Creating a Chunk Server	17

2.6.2.1	Creating Host UUIDs for Chunk Servers	18
2.7	Setting Up Clients	18
2.7.1	Stage 1: Preparing to Mount the Cluster	19
2.7.2	Stage 2: Mounting the Cluster	19
2.7.3	Stage 3: Configuring Virtual Machines and Containers	20
3.	Configuring Storage Clusters	22
3.1	Configuring MDS Servers	22
3.1.1	Adding MDS Servers	23
3.1.2	Removing MDS Servers	24
3.2	Configuring Chunk Servers	25
3.2.1	Adding New Chunk Servers to Increase Disk Space	25
3.2.2	Removing Chunk Servers	25
3.3	Configuring Clients	27
3.3.1	Adding Clients	27
3.3.2	Updating Clients	27
3.3.3	Removing Clients	27
3.4	Configuring High Availability	28
3.5	Managing Cluster Parameters	28
3.5.1	Cluster Parameters Overview	28
3.5.2	Configuring Replication Parameters	29
3.5.3	Configuring Encoding Parameters	30
3.5.4	Configuring Failure Domains	32
3.5.4.1	Failure Domain Topology	32
3.5.4.2	Defining Failure Domains	33
3.5.4.3	Recommendations on Failure Domains	33
3.5.5	Using Storage Tiers	34
3.5.5.1	What Are Storage Tiers	34
3.5.5.2	Configuring Storage Tiers	34
3.5.5.3	Monitoring Storage Tiers	35
3.5.6	Changing Storage Cluster Network	36
3.5.7	Enabling Online Compacting of Virtual Machine Disks	37
3.6	Managing Storage Licenses	38
3.6.1	Obtaining the Hardware Node ID	38
3.6.2	Installing the License	38
3.6.3	Updating the License	39

3.6.4	Viewing the License Contents	39
3.6.5	Checking the License Status	41
3.7	Shutting Down and Starting Up Cluster Nodes	41
4.	Exporting Storage Cluster Data	44
4.1	Accessing Storage Clusters via NFS	44
4.1.1	Preparing the Ploop	45
4.1.2	Setting Up the NFS Share	45
4.2	Accessing Storage Clusters via iSCSI	46
4.2.1	Preparing to Work with iSCSI Targets	47
4.2.2	Creating and Running iSCSI Targets	48
4.2.3	Listing iSCSI Targets	49
4.2.4	Transferring iSCSI Targets Between Nodes	50
4.2.5	Stopping iSCSI Targets	51
4.2.6	Deleting iSCSI Targets	51
4.2.7	Accessing iSCSI Targets from Operating Systems and Third-Party Virtualization Solutions	52
4.2.7.1	Accessing iSCSI Targets from CentOS 6.5	52
4.2.7.2	Accessing iSCSI Targets from Microsoft Windows Server 2012 R2	52
4.2.7.3	Accessing iSCSI Targets from VMware ESXi	59
4.2.7.4	Accessing iSCSI Targets from Citrix XenServer 6.2	59
4.2.7.5	Accessing iSCSI Targets from Microsoft Hyper-V	60
4.2.8	Configuring Multipath I/O for iSCSI Targets	61
4.2.9	Managing CHAP Accounts for iSCSI Targets	62
4.2.9.1	Creating CHAP Accounts for iSCSI Targets	62
4.2.9.2	Creating iSCSI Targets Bound to CHAP Accounts	62
4.2.9.3	Changing the CHAP Account Password	63
4.2.9.4	Listing CHAP Accounts and iSCSI Targets Assigned to Them	63
4.2.10	Managing LUN Snapshots	63
4.2.10.1	Creating LUN Snapshots	63
4.2.10.2	Listing LUN Snapshots	64
4.2.10.3	Switching Between LUN Snapshots	64
4.2.10.4	Viewing LUN Snapshot Information	64
4.2.10.5	Deleting LUN Snapshots	65
4.3	Accessing Storage Clusters via S3-like Object Storage	65
4.3.1	About Object Storage	65
4.3.1.1	Object Storage Infrastructure	66

4.3.1.2	Object Storage Overview	68
4.3.1.3	Object Storage Components	68
4.3.1.4	Data Interchange	71
4.3.1.5	Operations on Objects	72
4.3.2	Deploying Object Storage	74
4.3.2.1	Manually Binding Services to Nodes	80
4.3.3	Managing S3 Users	81
4.3.3.1	Creating S3 Users	82
4.3.3.2	Listing S3 Users	82
4.3.3.3	Querying S3 User Information	83
4.3.3.4	Disabling S3 Users	83
4.3.3.5	Deleting S3 Users	83
4.3.3.6	Generating S3 User Access Key Pairs	83
4.3.3.7	Revoking S3 User Access Key Pairs	84
4.3.4	Managing Object Storage Buckets	84
4.3.4.1	Managing Buckets with CyberDuck	85
4.3.4.2	Managing Buckets from Command Line	88
4.3.5	Best Practices for Using Object Storage	89
4.3.5.1	Bucket and Key Naming Policies	89
4.3.5.2	Improving Performance of PUT Operations	90
4.3.6	Appendices	90
4.3.6.1	Appendix A: Supported Amazon S3 REST Operations	90
4.3.6.2	Appendix B: Supported Amazon Request Headers	91
4.3.6.3	Appendix C: Supported Authentication Schemes	92
4.3.6.4	Appendix D: Amazon S3 Features Supported by Bucket Policies	92
5.	Monitoring Storage Clusters	97
5.1	Monitoring General Cluster Parameters	97
5.2	Monitoring Metadata Servers	100
5.3	Monitoring Chunk Servers	101
5.3.1	Understanding Disk Space Usage	103
5.3.1.1	Understanding Allocatable Disk Space	104
5.3.1.2	Viewing Space Occupied by Data Chunks	105
5.3.2	Exploring Chunk States	106
5.4	Monitoring Clients	107
5.5	Monitoring Physical Disks	109

5.6	Monitoring Event Logs	111
5.6.1	Exploring Basic Events	112
5.7	Monitoring Replication Parameters	114
6.	Managing Cluster Security	116
6.1	Security Considerations	116
6.2	Securing Server Communication in the Cluster	117
6.3	Cluster Discovery Methods	118
6.4	Storage Ports	119
6.5	Password-based Authentication	122
6.6	Installations via PXE Servers	122
7.	Maximizing Cluster Performance	123
7.1	Checking Data Flushing	123
7.2	Exploring Possible Disk Drive Configurations	125
7.3	Carrying Out Performance Benchmarking	127
7.4	Using 1 GbE and 10 GbE Networks	127
7.5	Setting Up Network Bonding	128
7.6	Using SSD Drives	129
7.6.1	Calculating Write Journal Size	130
7.6.2	Setting Up Chunk Servers with a Journal on SSD	131
7.6.3	Adding, Destroying, and Configuring CS Journals in Live Storage Clusters	133
7.6.3.1	Adding CS Journals	133
7.6.3.2	Destroying CS Journals	133
7.6.3.3	Moving CS Journals	133
7.6.3.4	Resizing CS Journals	134
7.6.4	Disabling Checksumming	134
7.6.5	Configuring Data Scrubbing	134
7.7	Improving High-Capacity HDD Performance	134
7.7.1	Improving Virtual Disk Performance	136
7.7.1.1	Preventing Partition Misalignment in Legacy Operating Systems	136
7.8	Disabling Inter-Tier Data Allocation	136
7.9	Managing Fast Path	136
7.9.1	Disabling Fast Path in Virtuozzo Hybrid Server 7.5 Update 4 and Earlier Versions	137
8.	Appendices	139
8.1	Appendix A - Troubleshooting	139

8.1.1	Submitting Problem Reports to Technical Support	139
8.1.2	Out of Disk Space	140
8.1.2.1	Symptoms	140
8.1.2.2	Solutions	141
8.1.3	Poor Write Performance	141
8.1.4	Poor Disk I/O Performance	141
8.1.5	Hardware RAID Controller and Disk Write Caches	142
8.1.6	SSD Drives Ignore Disk Flushes	142
8.1.7	Cluster Cannot Create Enough Replicas	143
8.1.8	Failed Chunk Servers	143
8.1.8.1	Replacing Disks Used as Chunk Servers	144
8.1.9	Failed Write Journaling SSDs	145
8.1.10	Failed MDS Servers	145
8.2	Appendix B - Frequently Asked Questions	145
8.2.1	General	145
8.2.2	Scalability and Performance	146
8.2.3	Availability	147
8.2.4	Cluster Operation	148
8.2.5	Shaman Service	150

CHAPTER 1

Introduction

This chapter provides basic information about this guide and Storage.

1.1 About This Guide

This guide serves two purposes:

- If you manage Storage via the web-based management panel, this guide complements the [Storage Administrator's Guide](#). Read it to find out how to use the command-line tools to perform tasks that are safe for Storage with the management panel. However, skip the chapters [Setting Up a Storage Cluster](#) on page 3 and [Configuring Storage Clusters](#) on page 22 as those console commands are unsafe for Storage with the management panel.
- If you manage Storage via the command line (i.e. the management panel is not installed), this guide is the primary documentation on Storage. Read it to find out how to perform every task related to managing Storage clusters, from initial setup to monitoring, optimization, and troubleshooting.

In general, it is recommended to manage Storage via the management panel. If you have it installed, consider the command-line tools secondary and use them with caution.

If you have the management panel installed, do not do the following via the command-line tools:

- Set custom paths for Storage services, in particular:
 - Create S3 clusters only in `/mnt/vstorage/vols/s3`.
 - Create iSCSI targets only in `/mnt/vstorage/vols/iscsi`.
- Mount clusters or change cluster mount options.

- Configure firewall with `firewall-cmd`.
- Rename network connections.
- Manage MDS/CS.
- Manage partitions, LVMs, or software RAID.
- Modify files in `/mnt/vstorage/vols` and `/mnt/vstorage/webcp/backup` directories.
- Set encoding or replication of cluster root.

1.2 About Storage

Storage is a solution allowing you to quickly and easily transform low-cost commodity storage hardware and network equipment into a protected enterprise-level storage, like SAN (Storage Area Network) and NAS (Network Attached Storage).

Storage is optimized for storing large amounts of data and provides replication, high-availability, and self-healing features for your data. Using Storage, you can safely store and run virtual machines and containers, migrate them with zero downtime across physical hosts, provide high availability for your Virtuozzo installations, and much more.

CHAPTER 2

Setting Up a Storage Cluster

This chapter provides information on setting up a Storage cluster. It starts with an overview of the Storage setup and then describes each setup step in detail.

Warning: Do not run console commands described in this chapter if you have the Storage management panel installed. Doing so may break your Storage installation. Instead, perform the same tasks via the management panel as explained in the [Storage Administrator's Guide](#).

2.1 Setup Overview

Setting up a Storage cluster includes these steps:

1. Configuring Storage cluster discovery. In this step, you define the way of detecting a cluster name and resolving the detected name into the IP addresses of MDS servers. For more on this step, see [Configuring Cluster Discovery](#) on page 4.
2. Checking data flushing. In this step, you check that all storage devices (hard disk drives, solid disk drives, RAID, etc.) you plan to include in your cluster can successfully flush data to disk when the server power goes off unexpectedly.
3. If required, prepare the additional (second, third, etc.) disks as described in [Preparing Disks for Storage](#) on page 8.
4. Setting up metadata servers. In this step, you create and configure metadata servers for storing metadata about chunk servers and data chunks. For more on this step, see [Setting Up the First Metadata Server](#) on page 14.

5. Setting up chunk servers. In this step, you create and configure chunk servers for storing the actual content of virtual machines and containers in data chunks. For more on this step, see [Setting Up Chunk Servers](#) on page 16.
6. Setting up clients. In this step, you create and configure clients from where you will access the Storage cluster and run virtual machines and containers. For more on this step, see [Setting Up Clients](#) on page 18.

Note: You can also set up a Storage using the Virtuozzo Hybrid Server 7 installer. It automatically configures your system as a metadata server, chunk server, or client. For detailed information on this, see the [Virtuozzo Hybrid Server 7 Installation Guide](#).

2.2 Configuring Cluster Discovery

Storage discovery is the process of:

1. Detecting all available cluster names on the network. Each Storage cluster is identified by a unique name. All cluster tools use this name when performing specific operations on the cluster or monitoring its health and status.
2. Resolving the detected cluster names into the network addresses of MDS servers. MDS servers are the central components of any cluster, so all cluster tools must be able to discover their IP addresses.

To set up cluster discovery in your network, you can use one of the following techniques:

- (Recommended) DNS records (see [Using DNS Records](#) on page 5),
- Zeroconf (see [Setting Up Zeroconf](#) on page 7).

You can also manually specify the information about metadata servers when setting up or configuring the cluster (see [Specifying MDS Servers Manually](#) on page 7).

Note: To verify that the Hardware Node can discover the cluster, use the `vstorage discover` command.

2.2.1 Using DNS Records

The recommended way of configuring cluster discovery is to use special DNS records. The process of setting up this type of cluster discovery includes two steps:

1. Announcing the information about running MDS servers in the cluster so that chunk servers, clients, and new MDS servers can automatically obtain this information when necessary. You do this using DNS SRV records.
2. Defining DNS TXT records or enabling DNS zone transfers so that you can discover the unique names of available clusters when necessary.

2.2.1.1 Announcing the Information About MDS Servers

You can use SRV records to announce the information about running MDS servers in the cluster. The service field of an SRV record pointing to an MDS server must have the following format:

```
_pstorage._tcp.CLUSTER_NAME
```

Where:

- `_pstorage` is the symbolic name reserved for Storage.
- `_tcp` denotes that Storage uses the TCP protocol for communication in the cluster.
- `CLUSTER_NAME` is the name of the Storage cluster described by the record.

The following example shows a DNS zone file that contains records for three MDS servers listening on the default port 2510 and configured for the `stor1` cluster:

```
$ORIGIN stor.test.
$TTL 1H
@ IN SOA ns rname.invalid. (1995032001 5H 10M 1D 3H)
  NS @
  A 192.168.100.1
s1 A 192.168.100.1
s2 A 192.168.100.2
s3 A 192.168.100.3
; SERVICE SECTION
; MDS for the 'stor1' cluster runs on s1.stor.test and listens on port 2510
_pstorage._tcp.stor1 SRV 0 1 2510 s1
; MDS for the 'stor1' cluster runs on s2.stor.test and listens on port 2510
_pstorage._tcp.stor1 SRV 0 1 2510 s2
; MDS for the 'stor1' cluster runs on s3.stor.test and listens on port 2510
_pstorage._tcp.stor1 SRV 0 1 2510 s3
```

```
; eof
```

Once you configure DNS SRV records for the `stor1` cluster, you can list them by issuing the following SRV query:

```
# host -t SRV _pstorage._tcp.stor1
_pstorage._tcp.stor1.stor.test has SRV record 0 1 2510 s1.stor.test.
_pstorage._tcp.stor1.stor.test has SRV record 0 1 2510 s2.stor.test.
_pstorage._tcp.stor1.stor.test has SRV record 0 1 2510 s3.stor.test.
```

2.2.1.2 Discovering Cluster Names

The easiest and most efficient way of discovering the names of clusters in your network is to specify all cluster names in `pstorage_clusters` TXT records of DNS zone files. The following example provides a sample of valid TXT record formats for Storage clusters:

```
pstorage_clusters 300 IN TXT "cluster1,cluster2" "cluster3,cluster4"
pstorage_clusters 300 IN TXT "cluster5"
pstorage_clusters 300 IN TXT "cluster6" "cluster7"
```

Another way of discovering cluster names in your network is to use DNS zone transfers. Once DNS zone transfers are enabled, cluster tools will be able to retrieve all DNS SRV records from DNS zone files and extract cluster names from these records.

After you set up cluster discovery via DNS TXT records or DNS zone transfers, you can run the `vstorage discover` command on any of the cluster servers to discover the names of all clusters in your network:

```
# vstorage discover
02-10-12 13:16:46.233 Discovering using DNS TXT records: OK
02-10-12 13:16:46.308 Discovering using DNS zone transfer: FAIL
stor1
stor2
stor3
```

The example `vstorage` output shows that:

- Clusters names are discovered via the DNS TXT records.
- Three clusters with the names of `stor1`, `stor2`, and `stor3` are currently set up on the network.

2.2.2 Setting Up Zeroconf

Warning: Zeroconf discovery does not work if services are running in containers or virtual machines.

Zeroconf is another method of discovering cluster names and resolving the discovered names into the IP addresses of running MDS servers. This method does not require any special configuration efforts on your part, except ensuring that multicasts are supported and enabled on your network.

Note: To verify that the Hardware Node can discover the cluster, use the `vstorage discover` command.

2.2.3 Specifying MDS Servers Manually

If you cannot configure the DNS records in your network, you need to manually specify the IP addresses of all running MDS servers in the cluster each time you do any of the following:

- Set up a new MDS server in a cluster (except for the first MDS server). For details, see [Adding MDS Servers](#) on page 23.
- Set up a new chunk server to a cluster. For details, see [Setting Up Chunk Servers](#) on page 16.
- Set up a new client for a cluster. For details, see [Setting Up Clients](#) on page 18.

To specify the IP address of an MDS server manually, create the `bs.list` file in the `/etc/vstorage/clusters/Cluster_Name` directory (make this directory if it does not exist) on the server you are configuring for the cluster and specify in it the IP address and port to use for connecting to the MDS server. For example:

```
# echo "10.30.100.101:2510" >> /etc/vstorage/clusters/stor1/bs.list
# echo "10.30.100.102:2510" >> /etc/vstorage/clusters/stor1/bs.list
```

This command:

1. Assumes that you are configuring discovery for the `stor1` cluster (thus, the directory name of `/etc/vstorage/clusters/stor1`).
2. Creates the `/etc/vstorage/clusters/stor1/bs.list` file on the server, if it did not exist before.
3. Adds the information on two MDS servers with IP addresses 10.30.100.101 and 10.30.100.102 to the

bs.list file.

2.3 Preparing Disks for Storage

Each chunk server is a service that handles a single physical disk in the cluster. Although the disk should be used solely by the CS service, technically, you can use it for multiple purposes. E.g., create a small partition for the operating system and leave the rest of disk space for Storage. If the disk is already partitioned, skip this section and proceed to creating a chunk server. Otherwise follow the instructions in this section to prepare the disk for use in Storage.

New disks attached to and recognized by the Hardware Node need to be prepared for use in the Storage cluster by means of the `/usr/libexec/vstorage/prepare_vstorage_drive` tool. The tool does the following:

1. Removes existing partitions from the disk.
2. Creates and formats the required partition(s).

After that, manually add the new partition to `/etc/vstorage/fstab` and mount it.

Note the following:

- If you do not need the disk to be bootable, run the tool with the `--noboot` option to skip GRUB bootloader installation.
- For SSD drives, use the `--ssd` option.
- To have the tool proceed without confirmation prompts, use the `-y` option.

2.3.1 Preparing Disks for Chunk Servers

1. To prepare an HDD or SSD for use as a chunk server, run the tool with the physical drive name as the option. For example:

```
# /usr/libexec/vstorage/prepare_vstorage_drive /dev/sdb --noboot
ALL data on /dev/sdb will be completely destroyed. Are you sure to continue? [y]
y
Zeroing out beginning and end of /dev/sdb...
Partitioning /dev/sdb...
Waiting for kernel...
Formatting /dev/sdb1 partition...
Done!
```

Note: The tool does not accept partition name as the option.

2. Find out partition UUID:

```
# ls -al /dev/disk/by-uuid/ | grep sdb1
lrwxrwxrwx 1 root root 10 Jun 19 02:41 f3fbcbb8-4224-4a6a-89ed-3c55bbc073e0 -> ../../sdb1
```

3. Add the new partition to `/etc/fstab` by UUID and specify the `noatime`, `nofail`, and `lazytime` mount options. For example:

```
UUID=f3fbcbb8-4224-4a6a-89ed-3c55bbc073e0 /vstorage/stor1-cs1 ext4 \
defaults,nofail,lazytime,noatime 0 0
```

Note: For some installations of Storage, you can find the `fstab` file at `/etc/vstorage/fstab`. Thus, add the partition details to `/etc/vstorage/fstab`.

4. Mount the partition to `/vstorage/<cluster>-cs<N>`, where `<cluster>` is cluster name and `<N>` is the first unused CS index number.

Note: If `/vstorage/<cluster>-cs<N>` does not exist, create it.

2.3.2 Preparing SSDs for Write Journaling or Caching

1. To prepare an SSD for write journaling or caching, run the tool with two options: `--ssd` and physical drive name. For example:

```
# /usr/libexec/vstorage/prepare_vstorage_drive /dev/sdb --ssd
ALL data on /dev/sdb will be completely destroyed. Are you sure to continue? [y]
y
Zeroing out beginning and end of /dev/sdb...
Partitioning /dev/sdb...
Waiting for kernel...
Formatting /dev/sdb1 partition...
Done!
```

Note: The tool does not accept partition name as the option.

2. Find out partition UUID:

```
# ls -al /dev/disk/by-uuid/ | grep sdb1
lrwxrwxrwx 1 root root 10 Jun 19 02:41 f3fbcbb8-4224-4a6a-89ed-3c55bbc073e0 -> ../../sdb1
```

3. Add the new partition to `/etc/fstab` by UUID and specify the `noatime`, `nofail`, and `lazytime` mount options. For example:

```
UUID=f3fbcbb8-4224-4a6a-89ed-3c55bbc073e0 /vstorage/stor1-ssd1 ext4 \
defaults,nofail,lazytime,noatime 0 0
```

4. Mount the partition to `/vstorage/<cluster>-ssd<N>`, where `<cluster>` is cluster name and `<N>` is the first unused SSD index number.

Note: If `/vstorage/<cluster>-ssd<N>` does not exist, create it.

2.4 Cache Configuration

2.4.1 Supported Device Types

Currently supported drives include HDD, SSD, and NVMe devices. Their characteristics are described in the table below.

Type	Cost	Performance	Interface and form-factor
Hard disk drives (HDD)	Low	Up to 200 MB/s Tens/Hundreds IOPS	SAS or SATA
Solid-state drives (SSD)	Average	Up to 600 MB/s Tens of thousands IOPS	SAS or SATA
Non-volatile memory express (NVMe)	High	From 1 to 10 GB/s Hundreds of thousands IOPS	2.5" U.2, PCIe Add-In-Card (AIC), or M.2

Note: PMem or NVRAM devices are not officially supported.

The amount and type of cache devices supported in your cluster should be checked per cluster node. In

order to be of any use, devices that provide acceleration must be faster than the underlying devices.

Note: Cache devices configured in RAID1 mirroring are not officially supported.

It is recommended that all capacity devices in the same storage tier should be identical in terms of technology and size. Otherwise, there may be unpredictable performance and behavior in case of a hardware failure. Moreover, all cluster nodes should offer the same amount of storage. If this requirement is not met, the storage space in the cluster will be limited by the smallest node.

A similar recommendation applies to cache devices. As the writing speed is constrained by the slowest device in the cluster, we strongly recommend using cache devices of the same technology and size.

2.4.2 Choosing a Cache Device

As all the data ingested in the system goes through cache devices, the choice of a cache device should be based not only on speed, but also on device endurance. Device endurance is measured in two ways:

- Drive Writes per Day (DWPD) measures the number of times the device can be completely overwritten each day, to reach the expected device end-of-life (usually five years).
- Terabytes Written (TBW) measures the expected amount of data that can be written before the device fails.

Both parameters are equivalent and should be carefully evaluated. For example, you have a 1-TB flash drive with 1 DPWD, that means you can write 1 TB into it every day over its lifetime. If its warranty period is five years, that works out to 1 TB per day * 365 days/year * 5 years = 1825 TB of cumulative writes, after which the drive usually will have to be replaced. Thus, the drive's TBW will be 1825.

The DWPD of a typical consumer-grade SSD drive can be as low as 0.1, while a high-end datacenter-grade flash drive can have up to 60 DWPD. For a cache device, the recommended minimum is 10 DWPD.

Another parameter to consider is power loss protection of the device. Some consumer-grade flash drives are known to silently ignore data flushing requests, which may lead to data loss in case of a power outage. Examples of such drives include OCZ Vertex 3, Intel 520, Intel X25-E, and Intel X-25-M G2. We recommend avoiding these drives (or test them with the `vstorage-hwflush-check` tool), and using enterprise-grade or datacenter-grade devices instead.

2.4.3 Provisioning Cache Devices

The minimum number of cache devices per node is one. However, note that in this case, if caching is used for all capacity devices, the cache device becomes a single point of failure, which may make the entire node unavailable. In order to avoid this, at least three cache devices per node are recommended.

Using multiple cache devices also provides the following improvements:

- More capacity. This can be helpful if data is written in long bursts or if the cache fails in offloading to the underlying device.
- Performance boost. If there is enough parallelism on the client side, the workload can be split among several cache devices, thus increasing the overall throughput.
- High availability. With fewer capacity devices per cache device or with RAID mirroring, you can lower the probability of a downtime or its impact.

We recommend provisioning one cache device to every 4-12 capacity devices. Keep in mind that the speed of a cache device should be at least twice as high as that of the underlying capacity devices combined.

Otherwise, the cache device may be a performance bottleneck. In this case, however, using cache can still improve latency and even performance in systems with lower parallelism.

2.4.4 Journal Sizing

Regardless of a cache device size, its journal size can be different, depending on the available space and number of chunk services that share the cache device. There are scenarios when using a journal smaller than the available capacity leads to performance improvements.

On one hand, if the size of all journals is less than the amount of available RAM, then the journal will only be used to store metadata. This will allow the system to keep the journal in RAM, avoiding all reads from the journal and resulting in fewer I/O operations. Ultimately, this will reduce the load on the cache devices and may improve the overall performance.

On the other hand, if the size of all journals is more than the amount of available RAM, then the journal will also be used to store temporary data and will serve as a read and write cache. This will boost the performance of both read and write requests. However, in this case, the cache device should be at least twice as fast as all of the underlying capacity devices combined, to be beneficial to the overall performance. If this is not the case, it is preferable to have a smaller journal. As speed is also largely dependent on the workload, this might not be obvious.

2.4.5 Cache Sizing

To decide on a cache device size, consider the endurance factor of a particular device and its journal size.

If you use cache for user data, then the cache device should be able to withstand sustained high throughput for as long as needed without filling up. The cache must offload its contents to the underlying device periodically, and this process depends on the speed of the underlying device. If the cache device becomes full, the system performance will degrade to the speed of the underlying devices, thus negating the caching benefits. Therefore, if the expected workload comes in bursts of a certain duration (for example, during office hours), the cache should be able to store at least the amount of data written during that period of time.

Risks and Possible Failures

Though cache devices may significantly improve the cluster performance, you need to consider their possible failures. Flash devices generally have a shorter lifespan and their use in this context exposes them to greater wear, when compared to capacity devices.

Also, keep in mind that as one cache device can be used to store multiple journals, all capacity devices associated with a cache device will become unavailable if this cache device fails.

Consider the following possible issues when using cache devices:

- **Data loss.** A cache device failure may lead to data loss if the data has no replicas or RAID mirroring is not configured.
- **Performance degradation.** If a cache device fails, the system will use other devices for storing data, which may result in a performance bottleneck or trigger the data rebalancing process to restore the data redundancy. This, in turn, will lead to increased disk and network usage and reduce the cluster performance.
- **Low availability.** With a failed cache device, data redundancy may be degraded, which may result in a read-only or unreadable cluster in severe cases.
- **Less capacity.** If a cache device fails, several capacity devices may become unavailable, leading to a lack of disk space available for writing new data.

To prevent these issues, use optimal redundancy policies and multiple cache devices in your system. Additionally, you can consider the possibility of using local replication (for example, RAID1) on top of distributed replication, especially in systems with low replication factors (1 replica or 1+0 encoding).

2.5 Setting Up the First Metadata Server

Setting up the first MDS server is the first step in creating a Storage cluster. You can add more MDS servers later to provide better availability for your cluster, as described in *Configuring MDS Servers* on page 22.

The process of setting up the first MDS server (called *master MDS server*) and, thus, creating a cluster includes two stages:

1. Preparing to create the MDS server.
2. Creating the MDS server.

2.5.1 Stage 1: Preparing to Create the First MDS Server

To prepare for making the first MDS server, do the following:

1. Choose a name for the cluster that will uniquely identify it among other clusters in your network. A name may contain the characters a-z, A-Z, 0-9, dash (-), and underscore (_). The examples used throughout this guide assume that the cluster name is stor1.

Note: When choosing a name for the cluster, make sure it is unique on your network. Also, do not use names that were once assigned to other clusters in your network, even if these clusters do not exist any more. This will help you avoid possible problems with services from previous cluster setups that might still be running and trying to operate on the new cluster. Though such operations will not succeed, they can make your work as a cluster administrator more difficult.

2. Log in to the computer you want to configure as a metadata server as root or as a user with root privileges.
3. Download and install the following RPM packages on the computer: `vstorage-ctl`, `vstorage-libs-shared`, and `vstorage-metadata-server`. The packages are available in the Virtuozzo remote repository (this repository is automatically configured for your system when you install Virtuozzo) and can be installed with this command:

```
# yum install vstorage-metadata-server
```

4. Make sure that cluster discovery is configured in your network. For details, see *Configuring Cluster Discovery* on page 4.

After you complete the steps above, you are ready to create the MDS server.

2.5.2 Stage 2: Creating the First MDS Server

To create the first MDS server, you use the `vstorage make-mds` command, for example:

```
# vstorage -c stor1 make-mds -I -a 10.30.100.101 -r /vstorage/stor1-mds -p
```

This command does the following:

1. Asks you for a password to use for password-based authentication in your cluster. Password-based authentication enhances security, requiring each server to be authenticated before it can be included in the cluster. The password you specify is encrypted and saved into the `/etc/vstorage/clusters/stor1/auth_digest.key` file on the MDS server.
2. Creates a Storage cluster with the name of `stor1` (the `-I` option tells `vstorage` to create a new cluster).
3. Creates a metadata server and configures the IP address of `10.30.100.101` for communication with this server. By default, Storage uses ports `2510` and `2511` to communicate with MDS servers. If necessary, you can replace the default ports with your own ones by reserving two unoccupied consecutive ports and specifying the first one after the IP address of your MDS server (e.g., `-a 10.30.100.101:4110` if your custom ports are `4110` and `4111`). Replace `10.30.100.101` in the example above with the IP address of your own MDS server. The specified IP address must be (1) static (or in the case of using DHCP, mapped to the MAC address of the MDS server) and (2) chosen from the range of IP addresses on the BackNet network dedicated to your Storage cluster.
4. Creates a journal in the `/vstorage/stor1-mds` directory on the MDS server and adds the information about the `stor1` cluster to it. When choosing a directory for storing the journal, make sure that the partition where the directory is located has at least 10 GB of free disk space.

After you have created the MDS server, start the MDS management service (`vstorage-mdsd`) and configure it to start automatically when the server boots:

```
# systemctl start vstorage-mdsd.target
# systemctl enable vstorage-mdsd.target
```

For information on including additional MDS servers in a Storage cluster, see [Configuring MDS Servers](#) on page 22.

2.6 Setting Up Chunk Servers

A chunk server stores actual data of virtual machines and containers and services requests to it. All data is split into chunks and can be stored in a Storage cluster in multiple copies called replicas.

Initially, any cluster is configured to have only one replica per each data chunk, which is sufficient to evaluate the Storage functionality using one server only. In production, however, to provide high availability for your data, you need to configure the cluster for each data chunk to have at least three replicas. This requires at least three chunk servers to be set up in the cluster. You can modify the default replication parameter using the `vstorage` utility. For details, see [Configuring Replication Parameters](#) on page 29.

Note the following:

- Using shared JBOD arrays across multiple nodes running CS services may introduce a single point of failure and make the cluster unavailable if all data replicas happen to be allocated and stored on the failed JBOD. For more information, see [Configuring Failure Domains](#) on page 32.
- Do not place chunk servers on disks already used in other I/O workloads, e.g., system or swap. Sharing disks between CS and other sources of I/O will result in severe performance loss and high I/O latencies.

The process of setting up a chunk server includes two stages:

1. Preparing to create a chunk server.
2. Creating the chunk server.

2.6.1 Stage 1: Preparing to Create a Chunk Server

To prepare for creating a chunk server, do the following:

1. Log in to the computer you want to act as a chunk server as root or as a user with root privileges.
2. Open the firewall for storage traffic. For a list of ports to open, consult [Opened Ports on Servers in Virtuozzo Storage Clusters](#).
3. Download and install the following RPM packages: `vstorage-ctl`, `vstorage-libs-shared`, and `vstorage-chunk-server`. These packages are available in the Virtuozzo remote repository (this repository is automatically configured for your system when you install Virtuozzo) and can be installed with this command:

```
# yum install vstorage-chunk-server
```


4. Make sure that cluster discovery is configured for the server. For details, see [Configuring Cluster Discovery](#) on page 4.
5. Authenticate the server in the cluster. This step is required only if the server where you are setting up the chunk server has never been authenticated in the cluster before. For example, you can skip this step if this is the same server where you set up the first MDS server. Otherwise, run the following command to authenticate the server in the cluster:

```
# vstorage -c stor1 auth-node
Please enter password for cluster:
```

During its execution, the command asks you for the password to validate the server. Type the password you specified when setting up the first MDS server and press Enter. `vstorage` then compares the provided password with the one stored on the MDS server, and if the passwords match, successfully authenticates the server.

6. If the disk the CS will be created on has not been prepared for Storage, do so as described in [Preparing Disks for Storage](#) on page 8.
7. Mount the prepared disk.

2.6.2 Stage 2: Creating a Chunk Server

Take note of the following:

- For large clusters (according to [Recommended Configuration](#)), it is critically important to configure proper failure domains to improve data availability. For more information, see [Configuring Failure Domains](#) on page 32.
- For details on creating chunk servers with journals on SSD, see [Setting Up Chunk Servers with a Journal on SSD](#) on page 131.

To create the chunk server, use the `vstorage make-cs` command, for example:

```
# vstorage -c stor1 make-cs -r /vstorage/stor1-cs
```

This command:

1. Creates the `/vstorage/stor1-cs` directory if it does not exist and configures it for storing data chunks.
2. Configures your disk as a chunk server and joins it to the `stor1` cluster.
3. Assigns the chunk server to the default storage tier. Storage tiers allow you to keep different kinds of

data on different chunk servers. To assign the CS to a specific tier, add `-t <tier>` parameter to the command, e.g., `-t 1` to assign the CS to tier 1. For details, see *Configuring Storage Tiers* on page 34.

After you have created the chunk server, start the chunk management service (`vstorage-csd`) and configure it to start automatically when the chunk server boots:

```
# systemctl start vstorage-csd.target
# systemctl enable vstorage-csd.target
```

Once you set up the first chunk server, proceed with creating other chunk servers.

2.6.2.1 Creating Host UUIDs for Chunk Servers

Storage distinguishes hosts the CS services run on by unique host UUIDs generated during installation. If you plan to set up new hosts by deploying a golden image with an OS and preinstalled Storage packages, you will need to generate new host UUIDs instead of the one inherited from the golden image.

To create a CS on a copy of the host, do the following:

1. Make sure the golden image does not contain any metadata servers, chunk servers, or clients.
2. Deploy the golden image on a clean host.
3. Generate a new UUID for the host to replace the one inherited from the golden image:

```
# /usr/bin/uuidgen -r | tr '-' ' ' | awk '{print $1$2$3}' > /etc/vstorage/host_id
```

Note: For more information on the `uuidgen` utility, see its man page.

4. Create a CS on the host.

2.7 Setting Up Clients

The process of setting up a client includes three stages:

1. Preparing to mount the Storage cluster to the client.
2. Mounting the cluster.
3. Configuring virtual machines and containers to be stored in the cluster.

2.7.1 Stage 1: Preparing to Mount the Cluster

To prepare for mounting the Storage cluster to the client:

1. Log in to the server you want to act as a client as root or as a user with root privileges.
2. Download and install the `vstorage-libs-shared` and `vstorage-client` RPM packages. These packages are available in the Virtuozzo remote repository (this repository is automatically configured for your system when you install Virtuozzo) and can be installed with this command:

```
# yum install vstorage-client
```

3. Create the directory to mount the Storage cluster to, for example:

```
# mkdir -p /vstorage/stor1
```

4. Make sure that cluster discovery is configured in your network. For details, see [Configuring Cluster Discovery](#) on page 4.
5. Authenticate the server in the cluster. This step is required only if the server where you are setting up the client has never been authenticated in the cluster before. For example, you can skip this step if this is the same server where you set up the first MDS server or some of the chunk servers. Otherwise, run the following command to authenticate the server in the cluster:

```
# vstorage -c stor1 auth-node  
Please enter password for cluster:
```

During its execution, the command asks you for the password to validate the server. Type the password you specified when setting up the first MDS server and press Enter. `vstorage` then compares the provided password with the one stored on the MDS server, and if the passwords match, successfully authenticates the server.

2.7.2 Stage 2: Mounting the Cluster

Next, you need to mount the cluster to make it accessible to the client. You can do this with the `vstorage-mount` command. For example, if your Storage cluster has the name of `stor1`, you can run this command to mount it to the `/vstorage/stor1` directory on the client:

```
# vstorage-mount -c stor1 /vstorage/stor1
```

You can also configure the client to automatically mount the cluster to the `/vstorage/stor1` directory when the client boots. To do this, add a line like the following to the `/etc/fstab` file:

```
vstorage://stor1 /vstorage/stor1 fuse.vstorage rw,nosuid,nodev,_netdev 0 0
```

Note: If the cluster is not used for virtualization, you can mount it with the `--fail-on-nospace` option. In this case an `ERR_NO_SPACE` error will be returned if the cluster runs out of free space.

2.7.3 Stage 3: Configuring Virtual Machines and Containers

To configure a server with Virtuozzo to store its virtual machines and containers in the cluster, do the following:

1. Log in to the server as root.
2. Configure containers for use in the cluster:
 - 2.1. Check the path to the container private area in the `/etc/vz/vz.conf` file. By default, the path is set to the following:

```
VE_PRIVATE=/vz/private/$VEID
```

- 2.2. Make a symbolic link from the container private area to the directory in the Storage cluster that will store containers. Assuming that this directory is `/vstorage/stor1/private`, create this directory and run the following command:

```
# ln -s /vstorage/stor1/private/ /vz/private
```

Note: If the `/vz/private` directory already exists on your server, remove it before running the `ln -s` command.

3. Configure virtual machines for use in the cluster:
 - 3.1. Check the default location of virtual machine files:

```
# prlsrvctl info | grep "VM home"  
VM home: /vz/vmprivate
```

- 3.2. Make a symbolic link from the default location to the directory in the Storage cluster that will store virtual machines. For example, to make a link to the `/vstorage/stor1/vmprivate` directory, create this directory and execute the following command:

```
# ln -s /vstorage/stor1/vmprivate/ /vz/vmprivate
```

Note: If the /vz/vmprivate directory already exists on your server, remove it before running the ln -s command.

CHAPTER 3

Configuring Storage Clusters

This chapter describes the ways to configure a Storage cluster once you create it.

Warning: Do not run console commands described in this chapter if you have the Storage management panel installed. Doing so may break your Storage installation. Instead, perform the same tasks via the management panel as explained in the [Storage Administrator's Guide](#).

3.1 Configuring MDS Servers

For a Storage cluster to function, the majority of MDS servers must be up and running in the cluster. So to ensure high availability of a cluster, you need to set up at least three MDS servers for it. This will allow you to survive the loss of one MDS server. By configuring five MDS servers for a cluster, you can ensure that your cluster will continue operating even if two MDS servers go offline.

Note the following:

- When adding and removing MDS servers, make sure that the running MDS servers in the cluster always have a majority.
- Remove non-functioning MDS servers from the cluster as soon as possible (e.g., right after you replace a broken server with a new one) to ensure that all MDS servers are up and running and the majority is not lost if one more MDS server fails. Let us assume that 3 MDS servers are running in your cluster. 1 MDS server fails so you add a new MDS server to the cluster. Now the total number of MDS servers in the cluster is 4, with one server offline. If one more MDS server fails, the cluster will have only 2 running MDS servers and become unavailable because the majority (3 running MDS servers) cannot be achieved.

any more.

This section explains how to:

- Add new MDS servers to a cluster.
- Remove existing MDS servers from a cluster.

3.1.1 Adding MDS Servers

The procedure of setting up the first MDS server is described in *Setting Up the First Metadata Server* on page 14. To configure a second and all subsequent MDS servers for a cluster, follow the steps below:

1. Make sure that you remember the exact name of the Storage cluster where you want to add an MDS server. The example below uses `stor1` as the cluster name.
2. Log in to the computer you want to configure as an MDS server and add to the cluster as root or as a user with root privileges.
3. Download and install the following RPM packages on the computer: `vstorage-ctl`, `vstorage-libs-shared`, and `vstorage-metadata-server`. These packages can be installed with this command:

```
# yum install vstorage-metadata-server
```

4. Make sure that cluster discovery is configured for the server. For details, see *Configuring Cluster Discovery* on page 4.
5. Authenticate the server in the cluster. This step is required only if the physical server where you are setting up the MDS server has never been authenticated in the cluster before. For example, you can skip this step if you have already configured the server as a chunk server or a client. Otherwise, run the following command to authenticate the server in the cluster:

```
# vstorage -c stor1 auth-node  
Please enter password for cluster:
```

During its execution, the command asks you for the password to validate the server. Enter the password you specified when setting up the first MDS server and press Enter. `vstorage` then compares the provided password with the one stored on the MDS server, and if the passwords match, successfully authenticates the server.

6. Create the MDS server and add it to the cluster:

```
# vstorage -c stor1 make-mds -a 10.30.100.102 -r /vstorage/stor1-mds
```

In the command above:

- `stor1` is the name of the cluster you are adding the MDS server to.
- `10.30.100.102` is the IP address of the new MDS server. Replace `10.30.100.102` in the example above with the IP address of your own MDS server. The specified IP address must be (1) static (or in the case of using DHCP, mapped to the MAC address of the MDS server) and (2) chosen from the range of IP addresses on the BackNet network dedicated to your Storage cluster.
- `/vstorage/stor1-mds` is the path to a journal that will store the information about the `stor1` cluster. When choosing a directory for storing the journal, make sure that the partition where the directory is located has at least 10 GB of free disk space.

If the DNS records or Zeroconf discovery is not configured in your network, you need to additionally use the `-b` option and specify the IP addresses of the first MDS server (and all other MDS servers, if you have more than one in your cluster) when running the command:

```
# vstorage -c stor1 make-mds -a 10.30.100.102:2510 -r /vstorage/stor1-mds -b 10.30.100.101
```

7. Start the MDS management service (`vstorage-mdsd`) and configure it to start automatically on the MDS server boot:

```
# systemctl start vstorage-mdsd.target
```

For instructions on how to check that the MDS server has been successfully configured for your cluster, see [Monitoring Storage Clusters](#) on page 97.

3.1.2 Removing MDS Servers

Sometimes, you may need to remove an MDS server from a Storage cluster, for example, to upgrade the server or to perform some maintenance tasks on it. To do this:

1. Configure a new MDS server to replace the one you plan to remove from the cluster. For instructions, see [Adding MDS Servers](#) on page 23.
2. Find out the index number of the MDS server to remove by running the following command on some of your MDS servers or clients:

```
# vstorage -c stor1 top
```

This will display detailed information about the cluster. Locate the section with the information about

MDS servers, for example:

```

...
MDSID STATUS  %CTIME  COMMITS %CPU  MEM  UPTIME  HOST
M   1 avail    0.0%    0/s    0.0%  64m  17d  6h  10.30.17.38
   2 avail    0.0%    0/s    0.0%  50m  12d  3h  10.30.45.12
   3 avail    0.0%    0/s    0.0%  57m  7d   1h  10.30.10.15
...

```

The index number is displayed in the **MDSID** column. In the output above, three MDS servers are configured for the stor1 cluster. They have index numbers of 1, 2, and 3.

3. Remove the MDS server from the cluster. For example, to remove the MDS server with index number 3, run this command:

```
# vstorage -c stor1 rm-mds 3
```

3.2 Configuring Chunk Servers

This section explains how to complete the following tasks:

- Increase disk space in a cluster by adding new chunk servers to it.
- Remove chunk servers from a cluster for performing maintenance tasks on them.

3.2.1 Adding New Chunk Servers to Increase Disk Space

You can increase the amount of disk space in a Storage cluster simply by adding new chunk servers to it. For details, see *Setting Up Chunk Servers* on page 16.

Note: Storage can scale to support at least 8 PB of effective available disk space, which means up to 24 PB of physical disk space in the case of mirroring with 3 copies.

3.2.2 Removing Chunk Servers

If you need to remove a chunk server from the cluster, for example, to perform some maintenance tasks on it, do the following:

1. Make sure that:

- The number of chunk servers configured for your cluster is enough to store the required number of data chunks (that is, equals or exceeds the current replication value).
 - The chunk servers have enough disk space to store the chunks. For instructions on obtaining this information, see [Monitoring Chunk Servers](#) on page 101.
2. Find out the index number of the chunk server you want to remove by running the following command on some of your cluster servers:

```
# vstorage -c stor1 top
```

This will display detailed information about the stor1 cluster. Locate the section with the information about chunk servers, for example:

```
...
CSID  STATUS    SPACE   FREE  REPLICAS  IOWAIT  IOLAT(ms)  QDEPTH  HOST
1025  active    105GB   88GB   40         0%      0/0        0.0     10.30.17.38
1026  active    105GB   88GB   40         0%      0/0        0.0     10.30.18.40
1027  active    105GB   99GB   40         0%      0/0        0.0     10.30.21.30
1028  active    107GB   101GB  40         0%      0/0        0.0     10.30.16.38
...
```

The **CSID** column displays the index number of chunk servers. In the output above, four chunk servers are configured for the stor1 cluster. They have index numbers of 1025, 1026, 1027, and 1028.

3. Remove the chunk server from the cluster. For example, to delete the chunk server with index number 1028 from the stor1 cluster, run this command:

```
# vstorage -c stor1 rm-cs --wait 1028
```

Once you initiate the delete operation, the cluster starts replicating data chunks that were stored on the removed server and placing them on the remaining chunk servers in the cluster. The `--wait` option, when specified, tells the command to wait until the operation is complete (which may take a long time).

Note the following:

- If the CS's disk has disappeared from the OS and the CS has no access to its repository, data chunks which need to be replicated will not be accessible and CS removal will not complete. You will need to forcibly remove the CS from the cluster with the `-f` option. **Warning:** Do so only if the CS is irreversibly lost. Never remove active chunk servers with the `-f` option.
- When deletion is in progress, you can cancel it with the command `vstorage -c stor1 rm-cs --cancel 1028`. This might be useful, for example, if you specified a wrong ID of the chunk server to remove.

To add a removed chunk server back to the cluster, set it up from scratch by following the steps in [Setting Up Chunk Servers](#) on page 16.

3.3 Configuring Clients

This section explains how to complete the following tasks:

- Add new clients to clusters.
- Update clients.
- Remove clients from clusters.

3.3.1 Adding Clients

The process of including additional clients in a Storage cluster does not differ from that of setting up the first client and is described in *Setting Up Clients* on page 18 in detail.

Once you configure the client, you can run different commands on it to administer the cluster. For example, you can monitor the cluster health and status using the `vstorage top` command. For more details on monitoring Storage clusters, see *Monitoring Storage Clusters* on page 97.

3.3.2 Updating Clients

The process of updating software on clients that participate in clusters does not differ from that of updating software on standalone servers, except for updating the `vstorage-client` package. When updating this package, pay attention to the following:

- If no cluster is mounted to the client, the client starts using the updated package right away.
- If at least one cluster is mounted to the client, the updated package is installed, but the client starts using it only after you remount the cluster or reboot the client.

3.3.3 Removing Clients

Removing a client from a Storage cluster simply means unmounting the directory under which the cluster is mounted on the client. Assuming that the cluster is mounted under `/vstorage/stor1`, you can unmount it as follows:

1. Make sure that all virtual machines and containers in the cluster are stopped.

2. Unmount the cluster:

```
# umount /vstorage/stor1
```

3. If your cluster is configured to be automatically mounted when the client boots, comment out the cluster entry in the `/etc/fstab` file on the client:

```
# vstorage://stor1 /vstorage/stor1 fuse.vstorage rw,nosuid,nodev 0 0
```

3.4 Configuring High Availability

High availability keeps virtual machines, containers, and iSCSI targets operational even if the hardware node they are hosted on fails. Three modes are available:

- DRS (default). In this mode, virtual machines and containers which were running on a failed node are relocated to healthy nodes based on available RAM and license capacity. This mode can be used for nodes on which the `pdrs` service is running.
- Round-robin (default fallback). In this mode, virtual machines, containers, and iSCSI targets from a failed node are relocated to healthy nodes in the round-robin manner.
- Spare. In this mode, virtual machines and containers from a failed node are relocated to a spare node—an empty node (that has no virtual machines, containers, iSCSI targets, and S3 clusters stored on it) with enough resources and a license to host all virtual environments from any given node in the cluster. Such a node is required for high availability to work in this mode.

For information on how to configure high availability and switch between its modes, consult [Managing High Availability Clusters](#).

3.5 Managing Cluster Parameters

This section explains what cluster parameters are and how you can configure them with the `vstorage` utility.

3.5.1 Cluster Parameters Overview

The cluster parameters control creating, locating, and managing replicas for data chunks in a Storage cluster. All parameters can be divided into three main groups of parameters: replication, encoding, location.

The table below briefly describes some of the cluster parameters. For more information on the parameters and how to configure them, see the following sections.

Parameter	Description
Replication Parameters	
Normal Replicas	The number of replicas to create for a data chunk, from 1 to 15. Recommended: 3.
Minimum Replicas	The minimum number of replicas for a data chunk, from 1 to 15. Recommended: 2.
Location Parameters	
Failure Domain	A placement policy for replicas, can be host (default) or disk (CS).
Tier	Storage tiers, from 0 to 3 (0 by default).

3.5.2 Configuring Replication Parameters

The cluster replication parameters define the following:

- The normal number of replicas of a data chunk. When a new data chunk is created, Storage automatically replicates it until the normal number of replicas is reached.
- The minimum number of replicas of a data chunk (optional). During the life cycle of a data chunk, the number of its replicas may vary. If a lot of chunk servers go down it may fall below the defined minimum. In such a case, all write operations to the affected replicas are suspended until their number reaches the minimum value.

To check the current replication parameters applied to a cluster, you can use the `vstorage get-attr` command. For example, if your cluster is mounted to the `/vstorage/stor1` directory, you can run the following command:

```
# vstorage get-attr /vstorage/stor1
connected to MDS#1
File: '/vstorage/stor1'
Attributes:
<...>
replicas=1:1
<...>
```

As you can see, the normal and minimum numbers of chunk replicas are set to 1.

Initially, any cluster is configured to have only 1 replica per each data chunk, which is sufficient to evaluate

the Storage functionality using one server only. In production, however, to provide high availability for your data, you are recommended to

- Configure each data chunk to have at least 3 replicas.
- Set the minimum number of replicas to 2.

Such a configuration requires at least 3 chunk servers to be set up in the cluster.

To configure the current replication parameters so that they apply to all virtual machines and containers in your cluster, you can run the `vstorage set-attr` command on the directory to which the cluster is mounted. For example, to set the recommended replication values to the `stor1` cluster mounted to `/vstorage/stor1`, set the normal number of replicas for the cluster to 3:

```
# vstorage set-attr -R /vstorage/stor1 replicas=3
```

The minimum number of replicas will be automatically set to 2 by default.

Note: For information on how the minimum number of replicas is calculated, see the `vstorage-set-attr` man page.

Along with applying replication parameters to the entire contents of your cluster, you can also configure them for specific directories and files. For example:

```
# vstorage set-attr -R /vstorage/stor1/private/MyCT replicas=3
```

3.5.3 Configuring Encoding Parameters

As a better alternative to replication, Storage can provide data redundancy by means of erasure coding. With it, Storage breaks the incoming data stream into fragments of certain size, then splits each fragment into a certain number (M) of 1-megabyte pieces and creates a certain number (N) of parity pieces for redundancy. All pieces are distributed among M+N storage nodes, that is, one piece per node. On storage nodes, pieces are stored in regular chunks but such chunks are not replicated as redundancy is already achieved. The cluster can survive failure of any N storage nodes without data loss.

The values of M and N are indicated in the names of erasure coding redundancy modes. For example, in the 5+2 mode, the incoming data is broken into 5MB fragments, each fragment is split into five 1MB pieces and two more 1MB parity pieces are added for redundancy. In addition, if N is 2, the data is encoded using the RAID6 scheme, and if N is greater than 2, Reed-Solomon erasure codes are used.

It is recommended to use the following erasure coding redundancy modes (M+N):

- 1+0
- 3+2
- 5+2
- 7+2
- 17+3

Encoding is configured for directories. For example:

```
# vstorage set-attr -R /vstorage/stor1 encoding=5+2
```

After encoding is enabled, the redundancy mode cannot be changed back to replication. However, you can switch between different encoding modes for the same directory.

The general recommendation is to always have at least one more node in a cluster than required by the chosen redundancy scheme. For example, a cluster using replication with 3 replicas should have four nodes, and a cluster that works in the 7+2 erasure coding mode should have ten nodes. Such a cluster configuration has the following advantages:

- The cluster will not be exposed to additional failures when in the degraded state. With one node down, the cluster may not survive another even single-disk failure without data loss.
- You will be able to perform maintenance on cluster nodes that may be needed to recover a failed node (for example, for installing software updates).
- In most cases, the cluster will have enough nodes to rebuild itself. In a cluster without a spare node, each replica of user data is distributed to each cluster node for redundancy. If one or two nodes go down, the user data will not be lost, but the cluster will become degraded and will only start self-healing after the failed nodes are back online. During its rebuilding process, the cluster may be exposed to additional failures until all of its nodes are healthy again.
- You can replace and upgrade a cluster node without adding a new node to the cluster. A graceful release of a storage node is only possible if the remaining nodes in the cluster can comply with the configured redundancy scheme. You can, however, release a node forcibly without data migration, but it will make the cluster degraded and trigger the cluster self-healing.

Note: After upgrading a node in a mixed cluster, you cannot migrate VEs (virtual machines and containers) created in datastores with encoding EC 3+2, 5+2, 7+2, or 17+3 from VHS 7.5 Update 4 to VHS 7.5 Update 3.

However, the migration of VEs created in local datastores and datastores with a 3-replica and 2-replica data redundancy mode is available. A mixed cluster is not supported and exists during the upgrade only.

3.5.4 Configuring Failure Domains

A failure domain is a set of services which can fail in a correlated manner. Due to correlated failures it is very critical to scatter data replicas across different failure domains for data availability. Failure domain examples include:

- A single disk (the smallest possible failure domain). For this reason, Storage never places more than 1 data replica per disk or chunk server (CS).
- A single host running multiple CS services. When such a host fails (e.g., due to a power outage or network disconnect), all CS services on it become unavailable at once. For this reason, Storage is configured by default to make sure that a single host never stores more than 1 chunk replica (see *Defining Failure Domains* on page 33 below).

3.5.4.1 Failure Domain Topology

Every Storage service component has topology information assigned to it. Topology paths define a logical tree of components' physical locations consisting of identifiers `host_ID.cs_ID` that are generated automatically:

- `host_ID` is a unique, randomly generated host identifier created during installation and located at `/etc/vstorage/host_id`.
- `cs_ID` is a unique service identifier generated at CS creation.

Note: To view the current services topology and disk space available per location, run the `vstorage top` command and press **w**.

3.5.4.2 Defining Failure Domains

Based on the levels of hierarchy described above, you can use the `vstorage set-attr` command to define failure domains for proper file replica allocation:

```
# vstorage -c <cluster_name> set-attr -R -p /failure-domain=<disk|host>
```

Where:

- `disk` means that only 1 replica is allowed per disk or chunk server.
- `host` means that only 1 replica is allowed per host (default).

You should use the same configuration for all cluster files as it simplifies the analysis and is less error-prone.

3.5.4.3 Recommendations on Failure Domains

Important: Do not use failure domain `disk` simultaneously with journaling SSDs. In this case, multiple replicas may happen to be located on disks serviced by the same journaling SSD. If that SSD fails, all replicas that depend on journals located on it will be lost. As a result, your data may be lost.

- For the flexibility of Storage allocator and rebalancing mechanisms, it is always recommended to have at least 5 failure domains configured in a production setup. Reserve enough disk space on each failure domain so if a domain fails it can be recovered to healthy ones.
- At least 3 replicas are recommended.
- If a huge failure domain fails and goes offline, Storage will not perform data recovery by default, because replicating a huge amount of data may take longer than domain repairs. This behavior managed by the global parameter `mds.wd.max_offline_cs_hosts` (configured with `vstorage-config`) which controls the number of failed hosts to be considered as a normal disaster worth recovering in the automatic mode
- Depending on the global parameter `mds.alloc.strict_failure_domain` (configured with `vstorage-config`), the domain policy can be strict (default) or advisory. Tuning this parameter is highly not recommended unless you are absolutely sure of what you are doing.

3.5.5 Using Storage Tiers

This section describes storage tiers used in Storage clusters and provides information of how to configure and monitor them.

3.5.5.1 What Are Storage Tiers

Storage tiers represent a way to organize storage space. You can use them to keep different categories of data on different chunk servers. For example, you can use high-speed solid-state drives to store performance-critical data instead of caching cluster operations.

3.5.5.2 Configuring Storage Tiers

To assign disk space to a storage tier, do the following:

1. Assign all chunk servers with SSD drives to the same tier. You can do this when setting up a chunk server (see *Stage 2: Creating a Chunk Server* on page 17 for details).

Note: For information on recommended SSD drives, see *Using SSD Drives* on page 129.

2. Assign the frequently accessed directories and files to tier 1 with the `vstorage set-attr` command. For example:

```
# vstorage set-attr -R /vstorage/stor1/private/MyCT tier=1
```

This command recursively assigns the directory `/vstorage/stor1/private/MyCT` and its contents to tier 1.

When assigning storage to tiers, have in mind that faster storage drives should be assigned to higher tiers. For example, you can use tier 0 for backups and other cold data (CS without SSD journals), tier 1 for virtual environments—a lot of cold data but fast random writes (CS with SSD journals), tier 2 for hot data (CS on SSD), journals, caches, specific virtual machine disks, and such.

This recommendation is related to how Storage works with storage space. If a storage tier runs out of free space, Storage will attempt to temporarily use a lower tier. If you add more storage to the original tier later, the data, temporarily stored elsewhere, will be moved to the tier where it should have been stored originally.

For example, if you try to write data to the tier 2 and it is full, Storage will attempt to write that data to tier 1, then to tier 0. If you add more storage to tier 2 later, the aforementioned data, now stored on the tier 1 or

0, will be moved back to the tier 2 where it was meant to be stored originally.

Automatic Data Balancing

To maximize the I/O performance of chunk servers in a cluster, Storage automatically balances CS load by moving hot data chunks from hot chunk servers to colder ones.

A chunk server is considered hot if its request queue depth exceeds the cluster-average value by 40% or more (see example below). With data chunks, “hot” means “most requested”.

The hotness (i.e. request queue depth) of chunk servers is indicated by the QDEPTH parameter shown in the output of `vstorage top` and `vstorage stat` commands. For example:

```
...
IO QDEPTH: 0.1 aver, 1.0 max; 1 out of 1 hot CS balanced    46 sec ago
...
CSID STATUS      SPACE AVAIL  REPLICAS  UNIQUE IOWAIT IOLAT(ms) QDEPTH HOST          BUILD_VERSION
1025 active      1007.3 156.8G   7142      0      10%    1/117    0.3 10.31.240.167 6.0.11-10
1026 active      1007.3 156.8G   7267      0      11%    0/225    0.1 10.31.240.167 6.0.11-10
1027 active      1007.3 156.8G   7151      0      2%     0/10     0.1 10.31.240.167 6.0.11-10
1028 active      1007.3 156.8G   7285      0      13%    1/141    1.0 10.31.240.167 6.0.11-10
...
```

In the output, the IO QDEPTH line shows the average and maximum request queue depth values in the entire cluster for the last 60 seconds. The QDEPTH column shows average request queue depth values for each CS for the last 5 seconds.

Each 60 seconds, the hottest data chunk is moved from a hot CS to one with a shorter request queue.

3.5.5.3 Monitoring Storage Tiers

You can monitor disk space assigned to each storage tier with the `top` utility in the verbose mode (enabled by pressing `v`). Typical output may look like this:

```

Cluster 'tiers': healthy
Space: [OK] allocatable 3.3TB of 3.5TB, 3.5TB total, 3.5TB free
tier 0: allocatable 1.6TB of 1.7TB, 1.7TB total, 1.7TB free
tier 1: allocatable 866GB of 912GB, 912GB total, 912GB free
tier 3: allocatable 866GB of 912GB, 912GB total, 912GB free
MDS nodes: 1 of 1, epoch uptime: 3 min
CS nodes: 4 of 4 (4 avail, 0 inactive, 0 offline)
Replication: 1 norm, 1 limit
Chunks: [OK] 0 healthy, 0 standby, 0 degraded, 0 urgent,
          0 blocked, 0 pending, 0 offline, 0 replicating,
          0 overcommitted, 0 deleting, 0 void
FS: 0B in 1 files, 0 inodes, 0 file maps, 0 chunks, 0 chunk
IO:      read      0B/s ( 0ops/s), write      0B/s ( 0ops/s)
IO total: read      0B ( 0ops), write      0B ( 0ops)
Repl IO: read      0B/s, write:      0B/s
Sync rate: 0ops/s, datasync rate: 0ops/s

```

3.5.6 Changing Storage Cluster Network

Before moving your cluster to a new network, consider the following:

- Changing the cluster network results in a brief downtime for the period when more than half of the MDS servers are unavailable.
- It is highly recommended to back up all MDS repositories before changing the cluster network.

To change the Storage cluster network, do the following on each node in the cluster where an MDS service is running:

1. Stop the MDS service:

```
# systemctl stop vstorage-mdsd.target
```

2. Specify new IP addresses for all metadata servers in the cluster with the command `vstorage configure-mds -r <MDS_repo> -n <MDS_ID@new_IP_address>[:port] [-n ...]`, where:

- `<MDS_repo>` is the repository path of the MDS on the current node.
- Each `<MDS_ID@new_IP_address>` pair is an MDS identifier and a corresponding new IP address. For example, for a cluster with 5 metadata servers:

```
# vstorage -c stor1 configure-mds -r /vstorage/stor1-cs1/mds/data -n 1@10.10.20.1 \
-n 2@10.10.20.2 -n 3@10.10.20.3 -n 4@10.10.20.4 -n 5@10.10.20.5
```

Note the following:

- You can obtain the identifier and repository path for the current MDS with the `vstorage list-services -M` command.
- If you omit the port, the default port 2510 will be used.

3. Start the MDS service:

```
# systemctl start vstorage-mdsd.target
```

3.5.7 Enabling Online Compacting of Virtual Machine Disks

Online compacting of virtual machine disks allows reclaiming disk space no longer occupied by data. It is based on triggering the TRIM command inside the VM once a week.

Online compacting is enabled by default for disks of Windows VMs. For disks of Linux VMs it is enabled when the guest tools are installed.

Virtual machine disks for which online compacting is enabled have the `discard='unmap'` flag set. Removing the flag from the virtual disk configuration disables online compacting for it.

Note: Online compacting may not work for virtio-blk devices even if the `discard='unmap'` flag is set.

For online compacting to work for VM disks located on Storage (in the replication mode), punch hole support also needs to be enabled for the Storage cluster file system. Do the following:

1. Make sure that all cluster nodes are updated to Virtuozzo Hybrid Server 7 Update 5 or newer. See [Updating Nodes in Storage Clusters](#).
2. Set the `FALLOC_FL_PUNCH_HOLE` flag by running the following command on any cluster node:

```
# vstorage set-config "gen.do_punch_hole=1"
```

Warning: Running the command before updating all of the chunk servers will result in data corruption!

To reclaim unused space accumulated before online compacting was enabled (e.g., from VMs created on Virtuozzo Hybrid Server 7 Update 4 and older), create a file inside the VM with the size comparable to that of the unused space, then remove it. The space will be reclaimed as soon as the TRIM command is triggered.

3.6 Managing Storage Licenses

This section describes the process of managing Storage licenses. You will learn to do the following:

- Install a new license for a Storage cluster.
- Update the installed license.
- View the installed license contents.
- Check the current license status.

3.6.1 Obtaining the Hardware Node ID

The Hardware Node ID (HWID) is required to purchase a Storage license. You can obtain the HWID with the `vstorage stat --license-hwid` command. For example:

```
# vstorage -c stor1 stat --license-hwid
...
3F96.DFF2.EAF6.CE86.DD49.786C.DC01.3D53
```

Note: A Storage Hardware Node ID is not the same as a Virtuozzo Hardware Node ID shown by the `vzlicview` command.

3.6.2 Installing the License

Along with installing Virtuozzo licenses on all clients in a cluster, you need to install a separate license to start using the Storage functionality. One license is required per cluster. You can install the license from any server participating in the cluster: an MDS server, a chunk server, or a client.

To install the license, use the `vstorage load-license` command:

```
# vstorage -c stor1 load-license -p XXXXXX-XXXXXX-XXXXXX-XXXXXX-XXXXXX
```

If you have obtained the license in the form of a file, you can install it by using the `-f` option instead of `-p` and specifying the full path to the license file:

```
# vstorage -c stor1 load-license -f /etc/storlicense
```

3.6.3 Updating the License

In Virtuozzo, you can use the `vstorage update-license` command to update the license currently installed on your server. When executed, the utility connects to the Key Authentication (KA) server, retrieves a new license, downloads it to the server, and installs it there.

To update a license, do the following:

1. Make sure that the server where you want to update the license is connected to the Internet.
2. Run the `vstorage update-license` command to update the license.

For example, to update a license installed in the `pcs1` cluster, execute this command:

```
# vstorage -c pcs1 update-license
```

By default, `vstorage` obtains a new license from the default KA server. However, you can explicitly specify what KA server to use by passing the `--server` option to the command:

```
# vstorage -c pcs1 update-license --server ka.server.com
```

Once you run the command, `vstorage` will connect to the KA server with the hostname of `ka.server.com`, download a new license, and install it on your server.

3.6.4 Viewing the License Contents

You can use the `vstorage view-license` command to view the information on the license currently installed in your cluster. When executed, the utility processes the license and shows its contents on the screen. A sample output of `vstorage view-license` is given below:

```
# vstorage -c stor1 view-license
HWID: XXXX.XXXX.XXXX.XXXX.XXXX.XXXX.XXXX.XXXX
PCSSTOR
  status="ACTIVE"
  version=1.0
  expiration="08/24/2012 19:59:59"
  graceperiod=3600
  key_number="PCSS.XXXXXXXXXX.XXXX"
```

```
platform="Linux"
product="PCSS"
gracecapacity=5
autorecovery=0
autorebalance=0
snapshots=1
capacity=500
replicas=5
```

The main license parameters are explained in the table below.

Name	Description
HWID	Cluster ID.
status	License status. For details, see Checking the License Status (p. 46).
version	Version of Storage for which the license was issued.
expiration	License expiration date and time.
graceperiod	Period, in seconds, during which Storage continues functioning after the license has expired.
key_number	Key number under which the license is registered on the Key Authentication server.
platform	Operating system with which the license is compatible.
product	Product for which the license has been issued.
gracecapacity	Amount of disk space that data chunks may occupy in the cluster, in per cent to the capacity limit value. For example, if the capacity limit is set to 1 TB, and the grace capacity is 5%, data chunks may use 50 GB above the capacity limit.
capacity	Total amount of disk space, in GB, data chunks may occupy in the cluster. To view the disk space currently used by chunks, run the <code>vstorage top</code> command, press the V key on your keyboard, and check the FS field. For details, see Understanding Disk Space Usage on page 103.
replicas	Maximum number of replicas a data chunk may have.
autorecovery	Denotes whether the auto-recovery feature is enabled (1) or disabled (0).
autorebalance	Denotes whether the auto-rebalance feature is enabled (1) or disabled (0).
snapshots	Denotes whether the snapshots feature is enabled (1) or disabled (0).

3.6.5 Checking the License Status

You can check the status of your license in one of the following ways:

- Using the `vstorage view-license`, for example:

```
# vstorage -c stor1 view-license | grep status
status="ACTIVE"
```

- Using the `vstorage stat` or `vstorage top` command, for example:

```
# vstorage -c stor1 stat | grep License
connected to MDS#1
License: PCSS.XXXXXXXXXX.XXXX is ACTIVE
```

The table below lists all statuses a license can have.

Status	Description
ACTIVE	License is valid and active.
VALID	License is valid and can be installed in the cluster.
EXPIRED	License has expired.
GRACED	License is currently on the grace period or data chunks in the cluster use disk space from the grace capacity.
INVALID	License is invalid (for example, because its start date is in the future).

3.7 Shutting Down and Starting Up Cluster Nodes

To shut down a Storage cluster completely, do the following:

1. Stop and disable the following services: `shaman`, `pdrs`, `vstorage-iscsi`, and `va-agent` (in case Virtuozzo Automator is installed):

```
# systemctl stop shaman pdrs vstorage-iscsid va-agent
# systemctl disable shaman pdrs vstorage-iscsid va-agent
```

2. Check what containers and virtual machines are running in the cluster with the `prlctl list` command and shut them down using `prlctl stop`. For example:

```
# prlctl list
```

```

UUID                                STATUS    IP_ADDR    T  NAME
{8330ff27-5e84-4d13-a69c-63f3f8a3c516}  running  -          VM  MyVM
# prlctl stop MyVM
Stopping the VM...
The VM has been successfully stopped.

```

3. Stop and disable the `prl-disp` and `vz` services:

```

# systemctl stop prl-disp vz
# systemctl disable prl-disp vz

```

4. If you use Storage GUI, also stop and disable the `vstorage-ui-agent` service:

```

# systemctl stop vstorage-ui-agent
# systemctl disable vstorage-ui-agent

```

5. Stop all clients in the cluster. To do this, on each client:

- 5.1. Make sure no storage files are opened using the `lsof` utility. For example, for the `stor1` cluster, run

```
# lsof +D /vstorage/stor1
```

- 5.2. Unmount the cluster file system using the `umount` command. For example, if the cluster is mounted to the `/vstorage/stor1` directory on a client, you can unmount it as follows:

```
# umount /vstorage/stor1
```

- 5.3. Disable the automatic mounting of the cluster by removing the cluster entry from the `/etc/fstab` file.

6. Stop and disable all CS and MDS servers:

```

# systemctl stop vstorage-csd.target vstorage-mdsd.target
# systemctl disable vstorage-csd.target vstorage-mdsd.target

```

7. Shut down cluster servers.

To start up a Storage cluster again, do the following:

1. Power on cluster servers.
2. Start and enable all MDS and chunk servers:

```

# systemctl start vstorage-mdsd.target vstorage-csd.target
# systemctl enable vstorage-mdsd.target vstorage-csd.target

```

3. Enable the automatic mounting of the cluster by adding the cluster entry to the `/etc/fstab` file and mount the cluster file system using the `mount` command.

4. If you use Storage GUI, also start and enable the `vstorage-ui-agent` service:

```
# systemctl start vstorage-ui-agent
# systemctl enable vstorage-ui-agent
```

5. Start and enable the following services: `prl-disp`, `vz`, `shaman`, `pdrs`, `vstorage-iscsid`, and `va-agent` (in case Virtuozzo Automator is installed):

```
# systemctl start prl-disp vz shaman pdrs vstorage-iscsid va-agent
# systemctl enable prl-disp vz shaman pdrs vstorage-iscsid va-agent
```

6. Start containers and virtual machines with the `prlctl start` command. For example:

```
# prlctl start MyVM
Starting the VM...
The VM has been successfully started.
```

CHAPTER 4

Exporting Storage Cluster Data

Usually, you access virtual machines and containers natively from clients running Storage. To do this, you use standard command-line utilities like `pr1ct1`. Another way to remotely access data located in Storage clusters is to export it as:

- NFS (running on ploops)
- Images over iSCSI
- S3-like object storage.

These methods are described further in this chapter.

4.1 Accessing Storage Clusters via NFS

To access a Storage Cluster via NFS, you need to:

1. Create and mount a ploop with the `ext4` file system.
2. Set up an NFS share using either the standard `exportfs` command or the `/etc/exports` file.
3. Access the NFS share on the remote computer.

The following sections describe these steps in detail.

4.1.1 Preparing the Ploop

Loopback block devices (ploops) allow you to attach any Storage file as a block device and format it to a conventional file system like ext4. Since Storage is not optimized for small files and does not use a POSIX-compliant file system, you can use ploops with ext4 when you need the aforementioned functionality.

To prepare the ploop, do the following:

1. Load the required modules:

```
# modprobe ploop pfmt_ploop1 pio_kaio
```

2. Create the ploop:

```
# mkdir /vstorage/ploop0
# ploop init -s 1g -t ext4 /vstorage/ploop0/img0
```

This command creates a 1 GB ploop with the ext4 file system.

3. Mount the ploop:

```
# mkdir /mnt/ploop0
# ploop mount -m /mnt/ploop0 /vstorage/ploop0/DiskDescriptor.xml
```

4. (Optional) Set up a permanent ploop mount using /etc/fstab:

```
# cat >> /etc/fstab <<EOF
/vstorage/ploop0/DiskDescriptor.xml          /mnt/ploop0      ploop           defaults 0 0
EOF
```

4.1.2 Setting Up the NFS Share

For the purpose of this example, let us assume that:

1. The source computer IP address is 192.168.0.1 and the destination computer IP address is 192.168.0.2.
2. The `exportfs` command is used to set up the NFS share
3. On the destination computer, the NFS share is mounted to the `/nfsshare` directory.

To share the created file system via NFS, do the following:

1. On the source computer, make sure the `nfsd` service is running.
2. On the source computer, run the `exportfs` command as follows:

```
# exportfs 192.168.0.2:/mnt/ploop0
```

3. On the remote computer, mount the shared path as follows:

```
# mount 192.168.0.1:/mnt/ploop0 /nfsshare
```

Now you can access the contents of the shared ext4 file system on the remote computer.

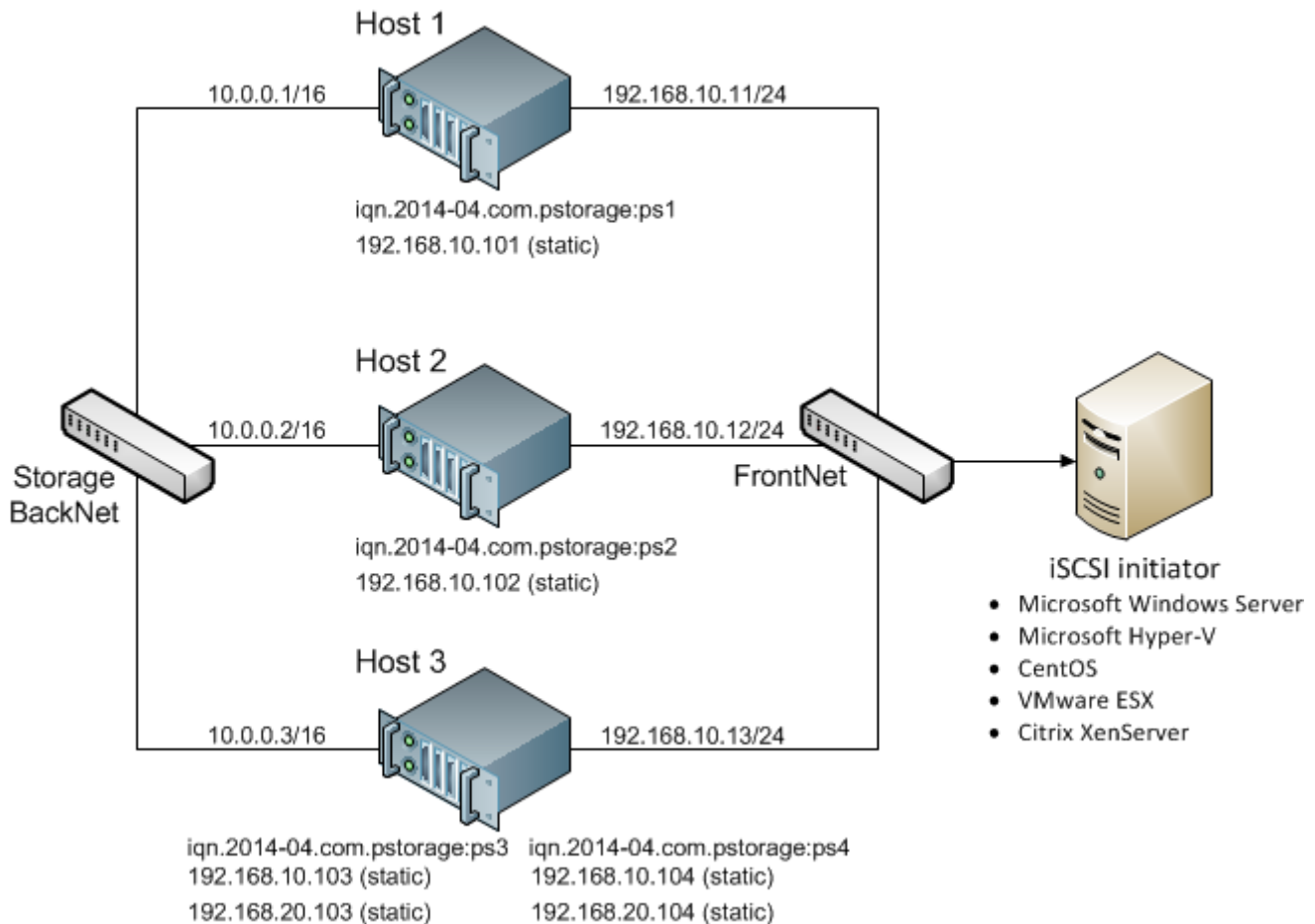
Note: More details on setting up NFS shares are provided in the [Red Hat Enterprise Linux Storage Administration Guide](#).

4.2 Accessing Storage Clusters via iSCSI

Storage allows you to export cluster disk space outside Storage bounds to operating systems and third-party virtualization solutions. Using dedicated `vstorage-iscsi` tools, you can export Storage disk space as LUN block devices over iSCSI in a SAN-like manner.

In Storage, you can create and run multiple iSCSI targets per Storage cluster Node. In turn, each iSCSI target can have multiple LUNs (virtual disks). At any given moment, each iSCSI target runs on a single Hardware Node. Thanks to High Availability, if a Node fails, iSCSI targets hosted on it are moved to and relaunched on a healthy Node (for more details on High Availability, see [Managing High Availability Clusters](#)).

The figure below shows a typical network configured for exporting Storage disk space over iSCSI.



In this example are three Virtuozzo Hardware Nodes working in a Storage cluster. Two Nodes host one iSCSI target each while the third hosts two iSCSI targets. Each Node has a static or dynamic IP address assigned from the Storage BackNet (created along with the Storage cluster) and the FrontNet. Each iSCSI target has a static IP address assigned from the FrontNet.

4.2.1 Preparing to Work with iSCSI Targets

On each Virtuozzo Hardware Node, where you need to create and run iSCSI targets, do the following:

1. Make sure the `vstorage-iscsi` and `pstorage-scsi-target-utils` packages are installed on the Hardware Node.
2. Make sure that the Hardware Node has access to the Storage cluster as client and has an entry in `/etc/fstab`. For more information, see [Setting Up Clients](#) on page 18.
3. Create a directory in the Storage cluster where you will store iSCSI targets and their configuration. For

example, `/vstorage/stor1/iscsi`.

4. Set the `ISCSI_ROOT` variable in `/etc/vstorage/iscsi/config` to the directory from the previous step. For example: `ISCSI_ROOT=/vstorage/stor1/iscsi`
5. Enable High Availability support for the hardware node. See [Enabling and Disabling High Availability for Nodes](#).

You are now ready to create and run iSCSI targets in your Storage cluster.

4.2.2 Creating and Running iSCSI Targets

Prerequisites:

- Each iSCSI target must be assigned at least one unique IP address from frontnet's static pool.
- The name of each iSCSI target must be unique in the Storage cluster.
- Storage iSCSI targets support persistent reservations to allow iSCSI initiators obtain exclusive access to the specified target's LUNs.

To create and start a target `test1` with the size of 100 GB, the LUN of 1, and the IP address of 192.168.10.100, execute the following commands:

```
# vstorage-iscsi create -n test1 -a 192.168.10.100
IQN: iqn.2014-04.com.vstorage:test1
# vstorage-iscsi lun-add -t iqn.2014-04.com.vstorage:test1 -l 1 -s 100G
# vstorage-iscsi start -t iqn.2014-04.com.vstorage:test1
```

If you need to change target's IP address, stop the target as described in [Stopping iSCSI Targets](#) on page 51, then run the command `vstorage-iscsi set -t <target_name> -a <new_IP_address>`.

If you need to increase the size of a LUN, stop the target as described in [Stopping iSCSI Targets](#) on page 51, then run the command `vstorage-iscsi lun-grow -t <target_name> -l <lun_ID> -s <new_size>`.

To check that the target is up, run the `vstorage-iscsi list` command with the target's name as the option. For example:

```
# vstorage-iscsi list -t iqn.2014-04.com.vstorage:test1
Target iqn.2014-04.com.vstorage:test1:
Portals:      192.168.10.100
Status:       running
Registered:   yes
Host:         fefacc38a2f140ca
LUN: 1, Size: 102400M, Used: 1M, Online: Yes
```


For information about the command output, see *Listing iSCSI Targets* on page 49.

iSCSI initiators can now access the target `iqn.2014-04.com.vstorage:test1` via the portal `192.168.10.100`.

Performance Tips

- Spread iSCSI targets evenly across Hardware Nodes in the cluster. For example, 10 Hardware Nodes with 1 iSCSI target per each will perform better than a single Hardware Node with 10 iSCSI targets on it.
- More LUNs per fewer iSCSI targets will perform better than fewer LUNs per more iSCSI targets.

4.2.3 Listing iSCSI Targets

Using the `vstorage-iscsi list` command, you can list all iSCSI targets registered on a Storage Node or display detailed information about a specific iSCSI target on a Storage Node.

To list all iSCSI targets registered on a Storage Node, run the command as follows:

```
# vstorage-iscsi list
IQN                STATUS  LUNs  HOST                PORTAL(s)
iqn.2014-04.com.vstorage:test1  running 1    fefacc38a2f140ca  192.168.10.100
iqn.2014-04.com.vstorage:test2  running 1    fefacc38a2f140ca  192.168.10.101
iqn.2014-04.com.vstorage:test3  stopped 1    fefacc38a2f140ca  192.168.10.102
iqn.2014-04.com.vstorage:test4  stopped 0    fefacc38a2f140ca  192.168.10.103
```

To display detailed information about an iSCSI target registered on a Storage Node, run the `vstorage-iscsi list` command with the target's name as the option. For example:

```
# vstorage-iscsi list -t iqn.2014-04.com.vstorage:test1
Target iqn.2014-04.com.vstorage:test1:
Portals:    192.168.10.100
Status:     running
Registered: yes
Host:       fefacc38a2f140ca
LUN:    1, Size: 102400M, Used:    1M, Online: Yes
```

The command outputs above show the following data:

Item	Description
Target	Unique alphanumeric name of the iSCSI target.
Portals	Target's IP address(es).

Continued on next page

Table 4.2.3.1 -- continued from previous page

Item	Description
Status	Target's current state. <ul style="list-style-type: none"> • running: target is running and ready for use (for local targets). • stopped: target is stopped (for local targets). • service failed: the iSCSI service is down (for local targets). • remote: target is registered on a different node. • unregistered: target is not registered on any node in the Storage cluster.
Registered	Whether or not the target is registered on the host which ID is shown in the Host entry.
Host	Storage hardware node ID.
LUN	Virtual disk's integer number within the target.
Size	Virtual disk's logical size (16 TB maximum).
Used	Virtual disk's physical size. The physical size can be smaller than logical due to the expanding format of the virtual disk (for more information, see Virtual Hard Disks).
Online	<ul style="list-style-type: none"> • Yes: the LUN is visible to and can be mounted by iSCSI initiators. • No: the LUN is invisible to and cannot be mounted by iSCSI initiators.

4.2.4 Transferring iSCSI Targets Between Nodes

You can transfer stopped iSCSI targets between Storage nodes. After the transfer, you will be able to start and manage the iSCSI target on the destination Node. On the source Node, you will only be able to delete the transferred target with the `--force` option (for more details, see [Deleting iSCSI Targets](#) on page 51).

To transfer an iSCSI target, do the following:

1. Make sure the target is stopped. For more details, see [Stopping iSCSI Targets](#) on page 51.
2. Unregister the target on its current Node with the `vstorage-iscsi unregister` command. For example:

```
# vstorage-iscsi unregister -t iqn.2014-04.com.vstorage:test1
```

3. Register the target on the new Node with the `vstorage-iscsi register` command. For example:

```
# vstorage-iscsi register -t iqn.2014-04.com.vstorage:test1
```

4.2.5 Stopping iSCSI Targets

To stop a Storage iSCSI target to which no initiators are connected, use the `vstorage-iscsi stop` command. For example, for the target `iqn.2014-04.com.vstorage:test1`:

```
# vstorage-iscsi stop -t iqn.2014-04.com.vstorage:test1
```

If one or more iSCSI initiators are still connected to the target, you will be informed as follows:

```
# vstorage-iscsi stop -t iqn.2014-04.com.vstorage:test1
initiators still connected
Initiator: iqn.1994-05.com.redhat:c678b9f6f0 (192.168.30.100)
Unable stop target iqn.2014-04.com.vstorage:test1
```

In this case, disconnect the iSCSI initiator according to the product manual and run the `vstorage-iscsi stop` command again.

To forcibly stop a target to which one or more initiators are still connected, add the `-f` option to the command above. For example:

```
# vstorage-iscsi stop -t iqn.2014-04.com.vstorage:test1 -f
```

Breaking the iSCSI connection in such a way may result in I/O errors on the iSCSI initiator's side.

4.2.6 Deleting iSCSI Targets

You can delete Storage iSCSI targets with the `vstorage-iscsi delete` command. Deleting a Storage iSCSI target, you will also delete all the LUNs within it.

To delete a Storage iSCSI target, do the following:

1. Make sure the target is stopped (for more details, see [Stopping iSCSI Targets](#) on page 51).
2. Run the `vstorage-iscsi delete` command with the target name as the option. For example:

```
# vstorage-iscsi delete -t iqn.2014-04.com.vstorage:test1
```

To delete a stopped iSCSI target registered on a different host, add the `--force` option to the `vstorage-iscsi delete` command. For example:

```
# vstorage-iscsi delete -t iqn.2014-04.com.vstorage:test1 --force
```

4.2.7 Accessing iSCSI Targets from Operating Systems and Third-Party Virtualization Solutions

This section describes ways to attach Storage iSCSI targets to a number of operating systems and third-party virtualization solutions.

4.2.7.1 Accessing iSCSI Targets from CentOS 6.5

1. Make sure that the `iscsi-initiator-utils` package is installed.
2. Discover the required target by its IP address. For example:

```
# iscsiadm --mode discovery --type sendtargets --portal 192.168.10.100
```

3. Restart the `iscsid` service to rescan for newly added drives:

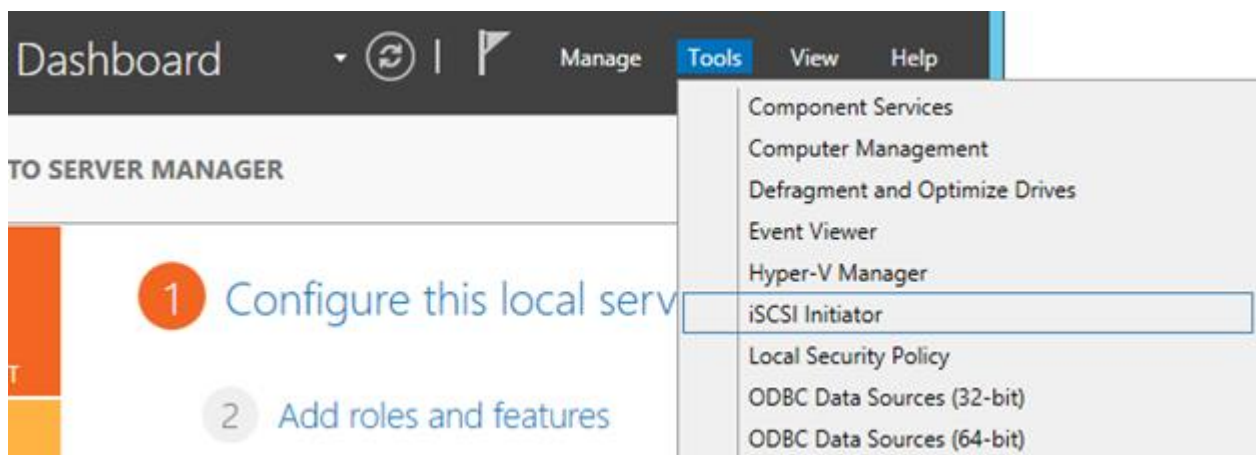
```
# service iscsi restart
```

To check that the new drive has appeared in the system, use `fdisk`, `parted` or similar tools.

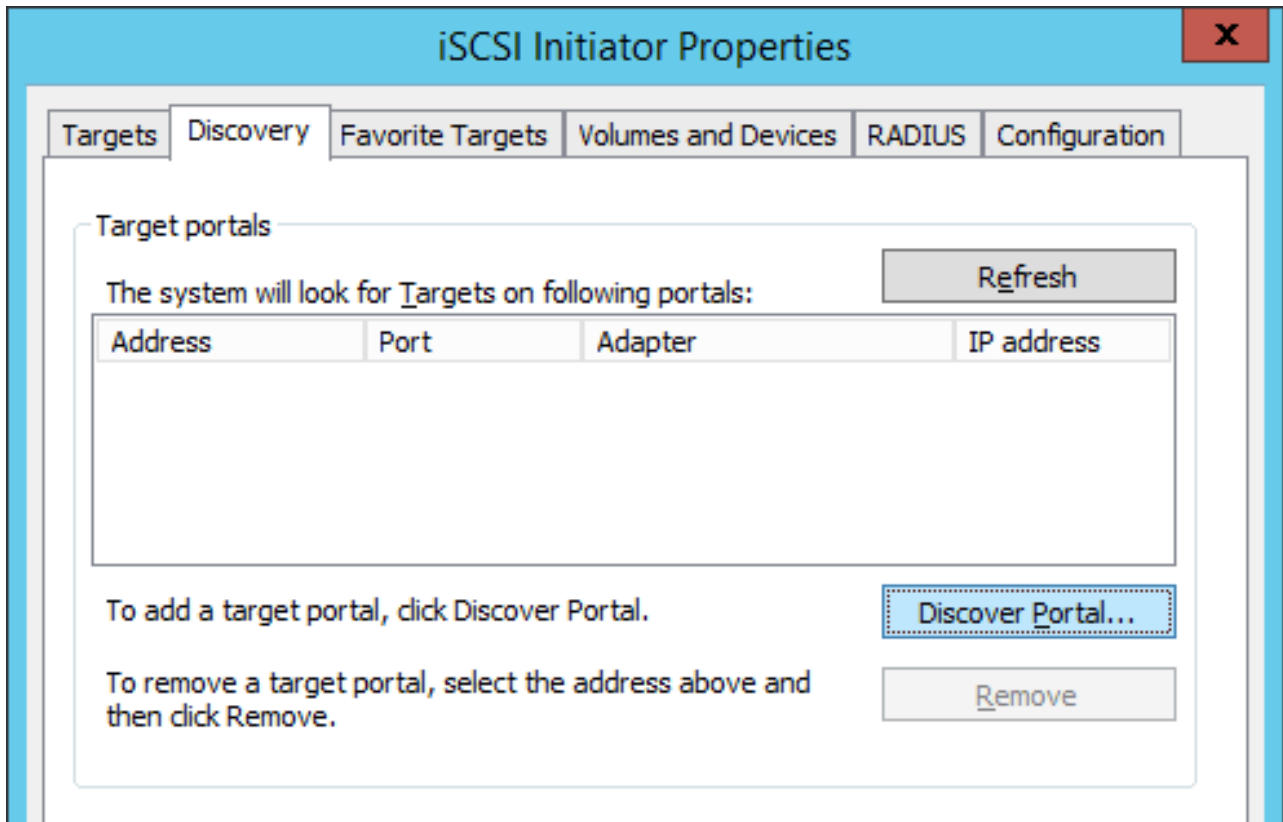
For more information, see the [Red Hat Enterprise Linux Storage Administration Guide](#).

4.2.7.2 Accessing iSCSI Targets from Microsoft Windows Server 2012 R2

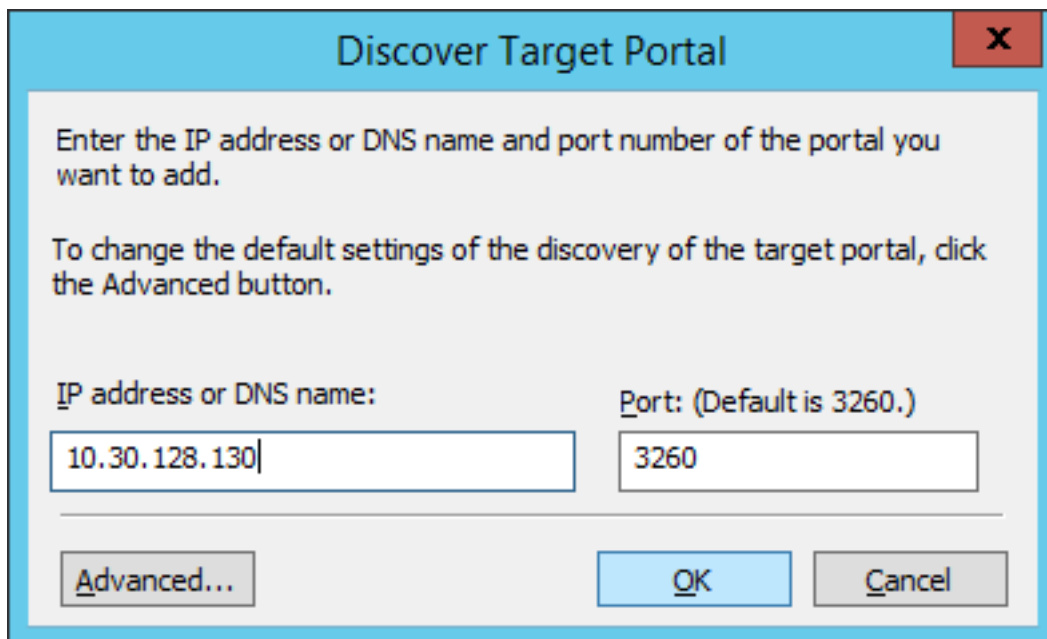
1. In the **Server Manager Dashboard**, click the **Tools** menu in the toolbar and select **iSCSI Initiator**.



2. In the **iSCSI Initiator Properties**, switch to the **Discovery** tab and click **Discover Portal...**

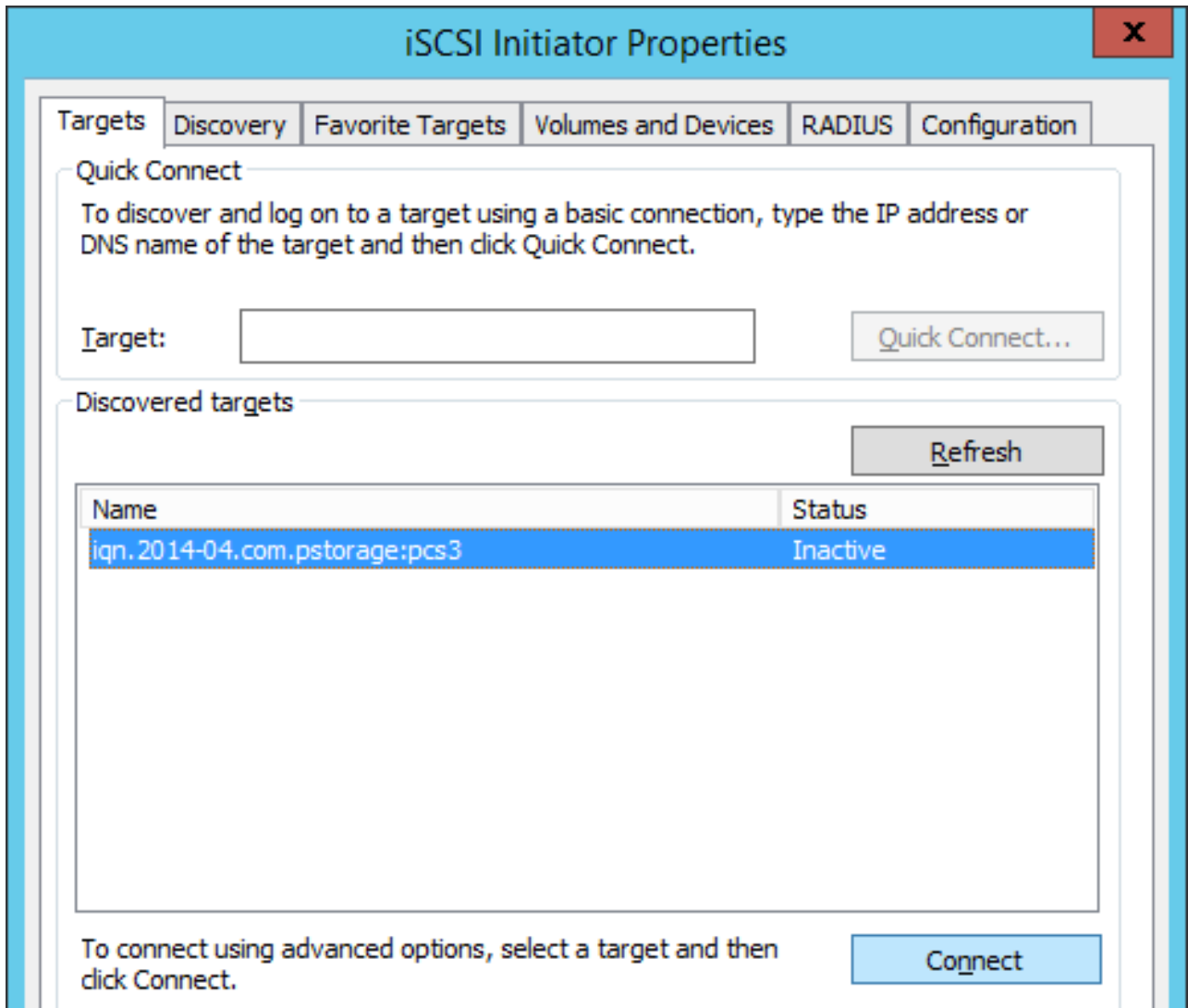


3. In the **Discover Target Portal** window, enter the portal IP address and click **OK**.

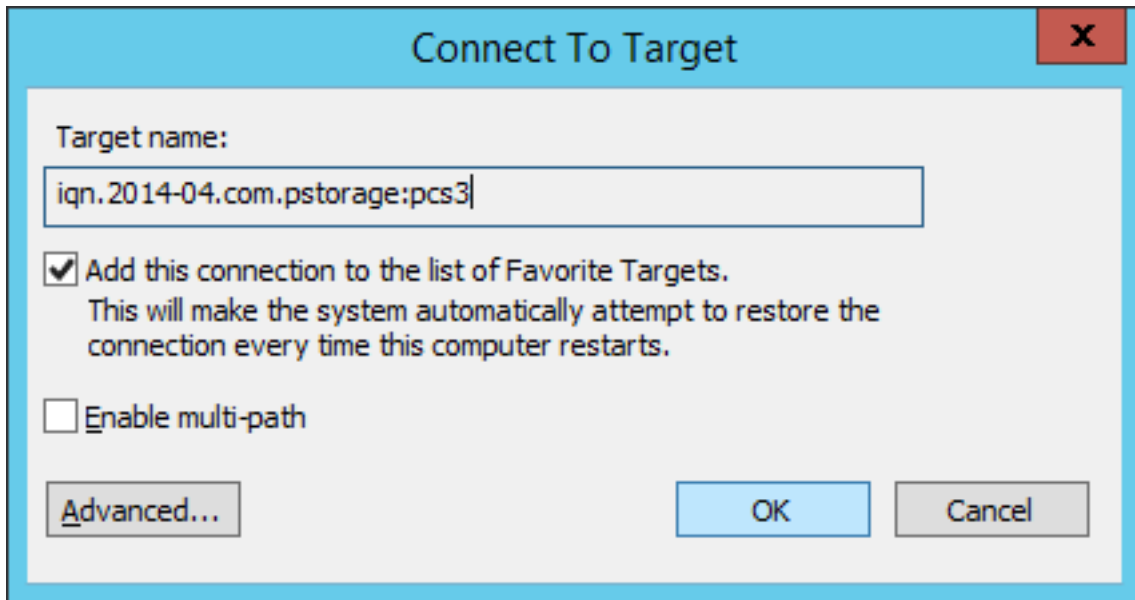


The newly added portal will appear in the **Target portals** section.

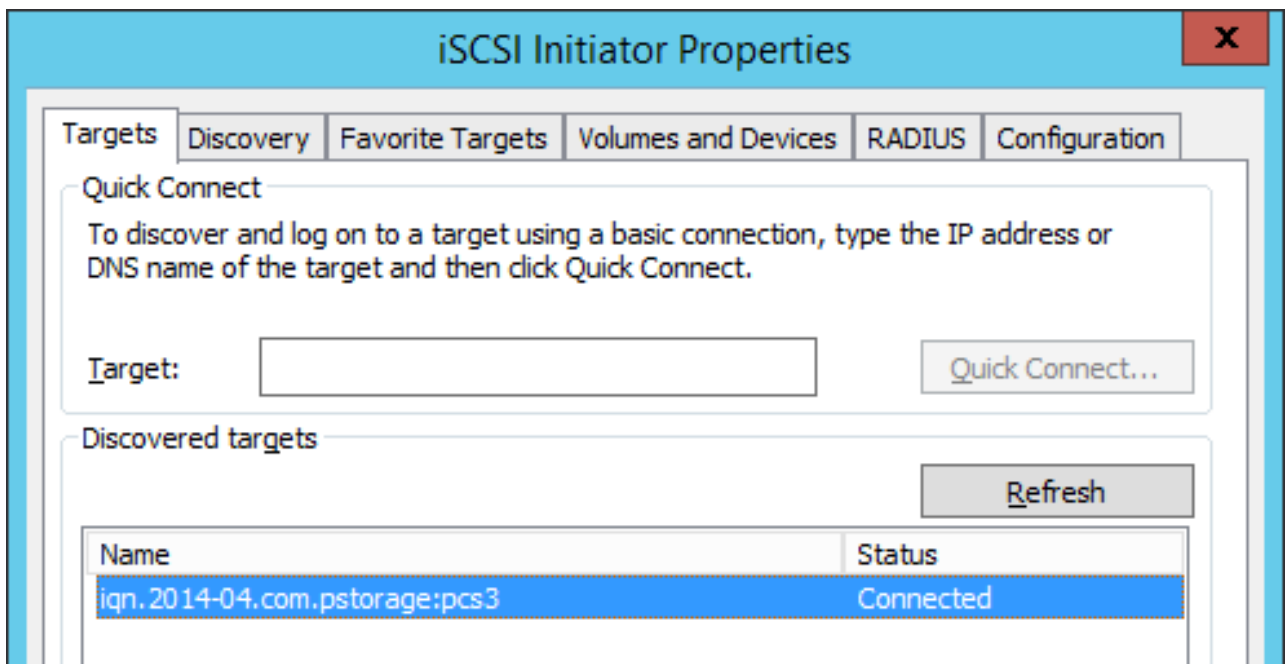
4. On the **iSCSI Initiator Properties** > **Targets** tab, select the new target in the **Discovered targets** section and click **Connect**.



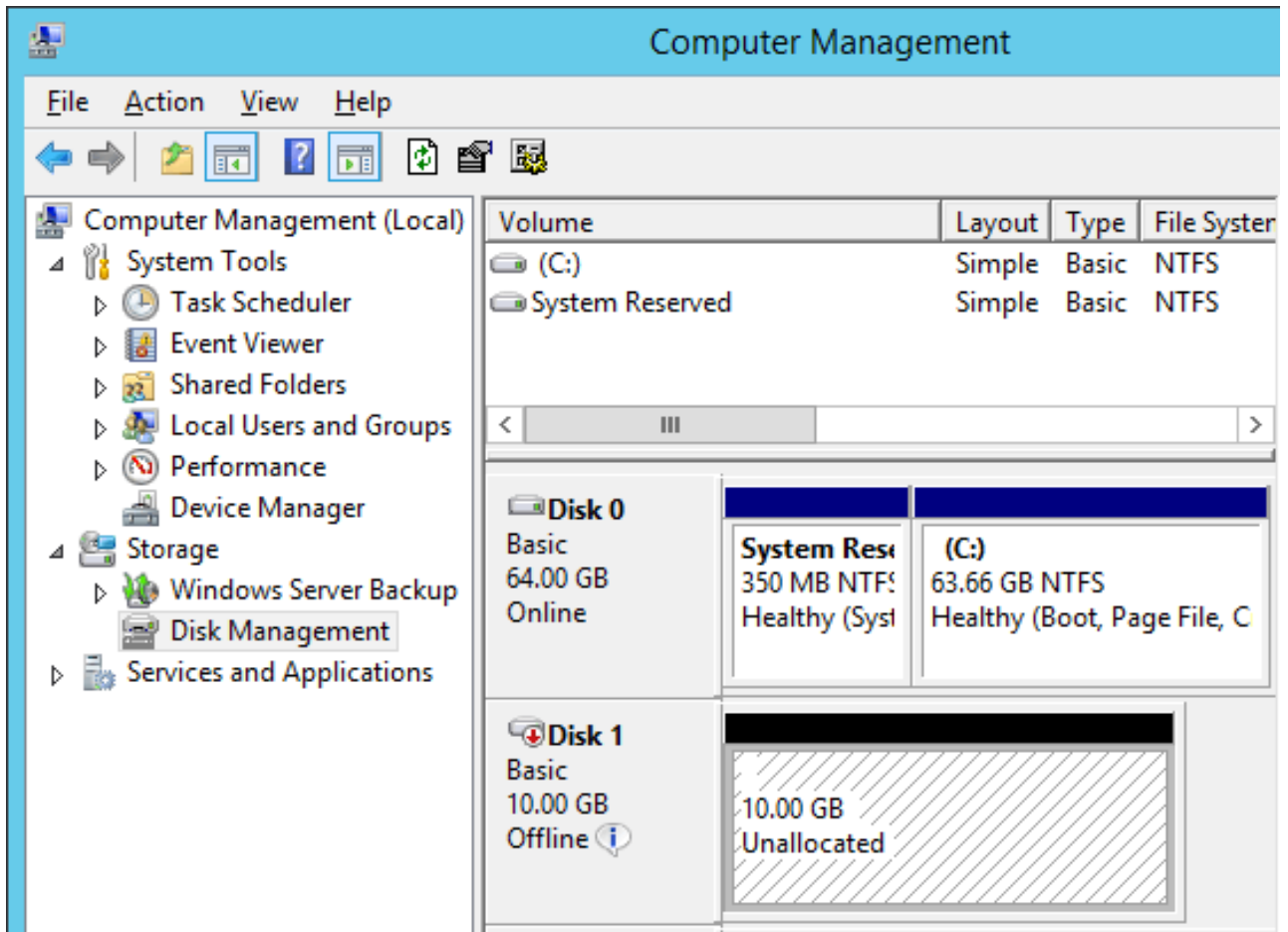
5. In the **Connect to Target** window, click **OK**.



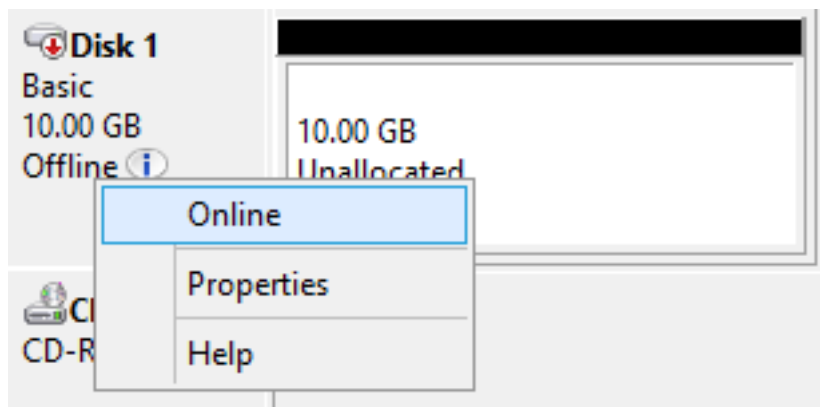
Target's **Inactive** status will change to **Connected**.



The newly attached disk will appear in **Server Manager Dashboard > Computer Management > Storage > Disk Management**.

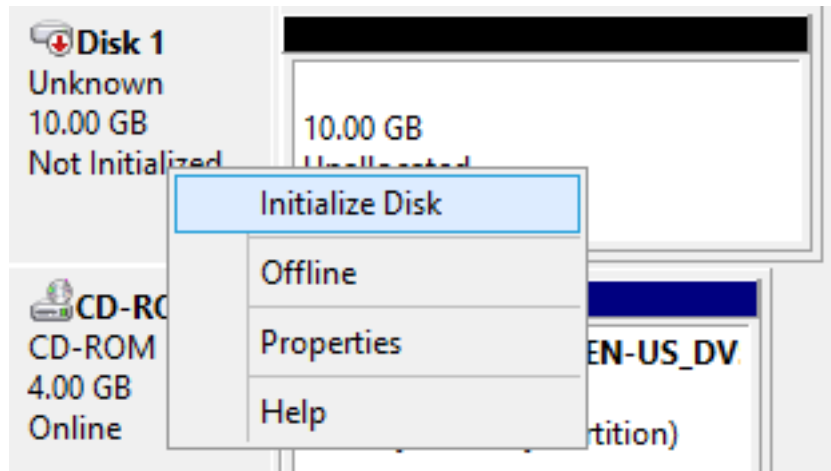


6. Right-click the disk information section and select **Online**.

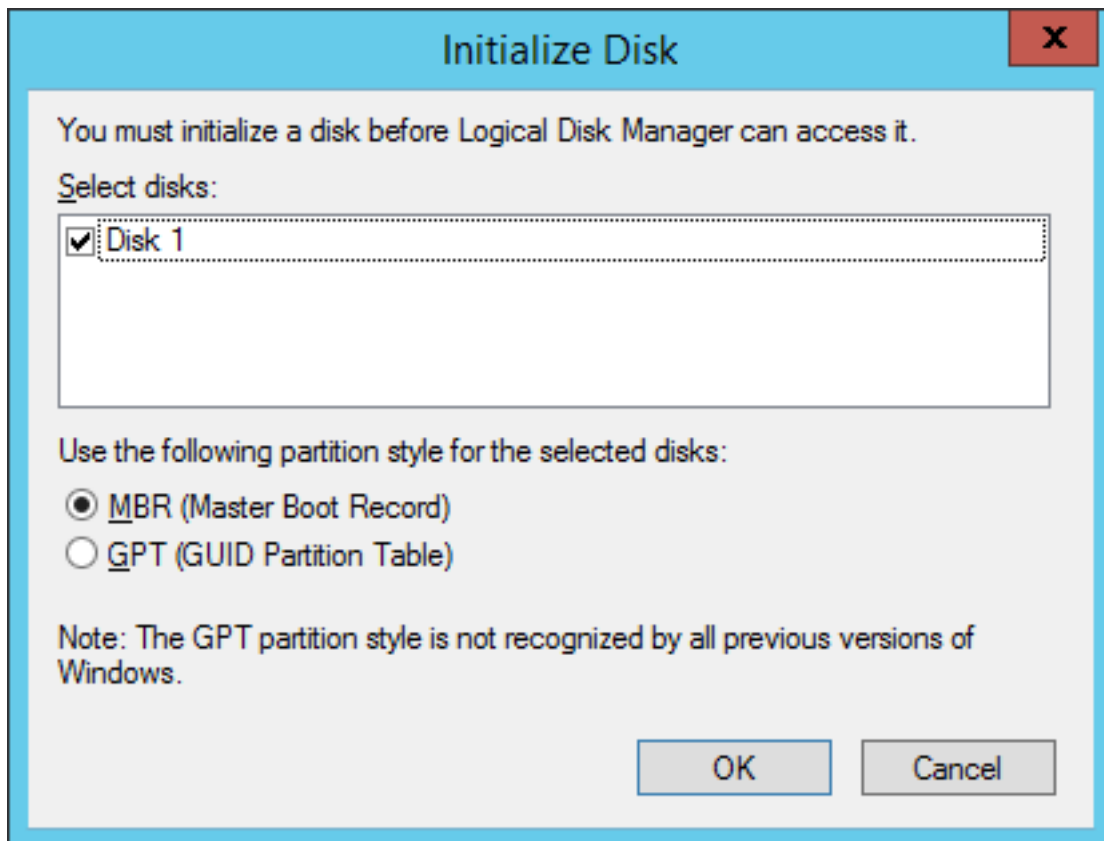


The disk status will change to **Online**.

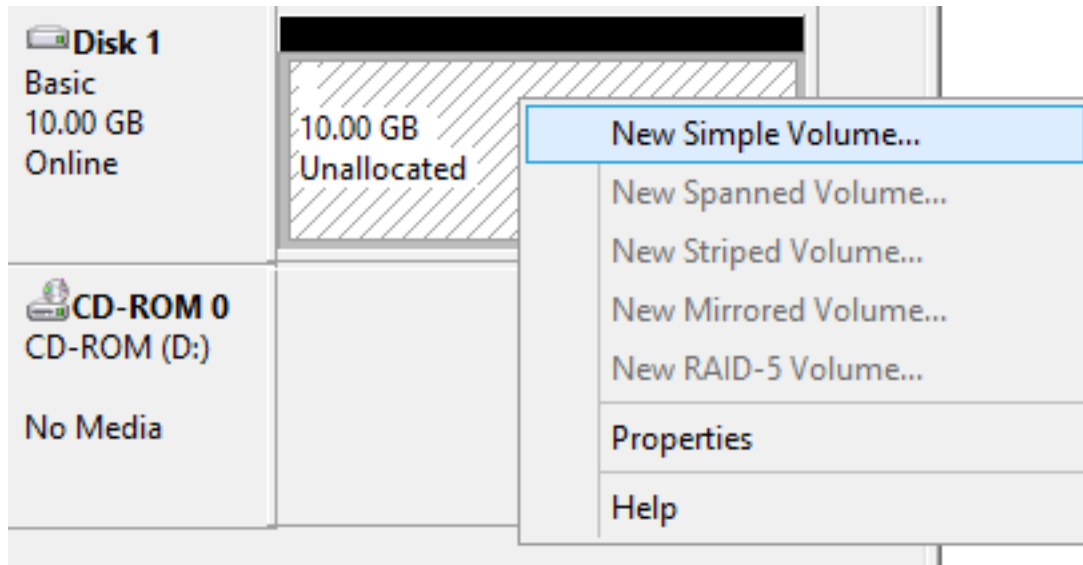
7. Right-click the disk information section and select **Initialize Disk**.



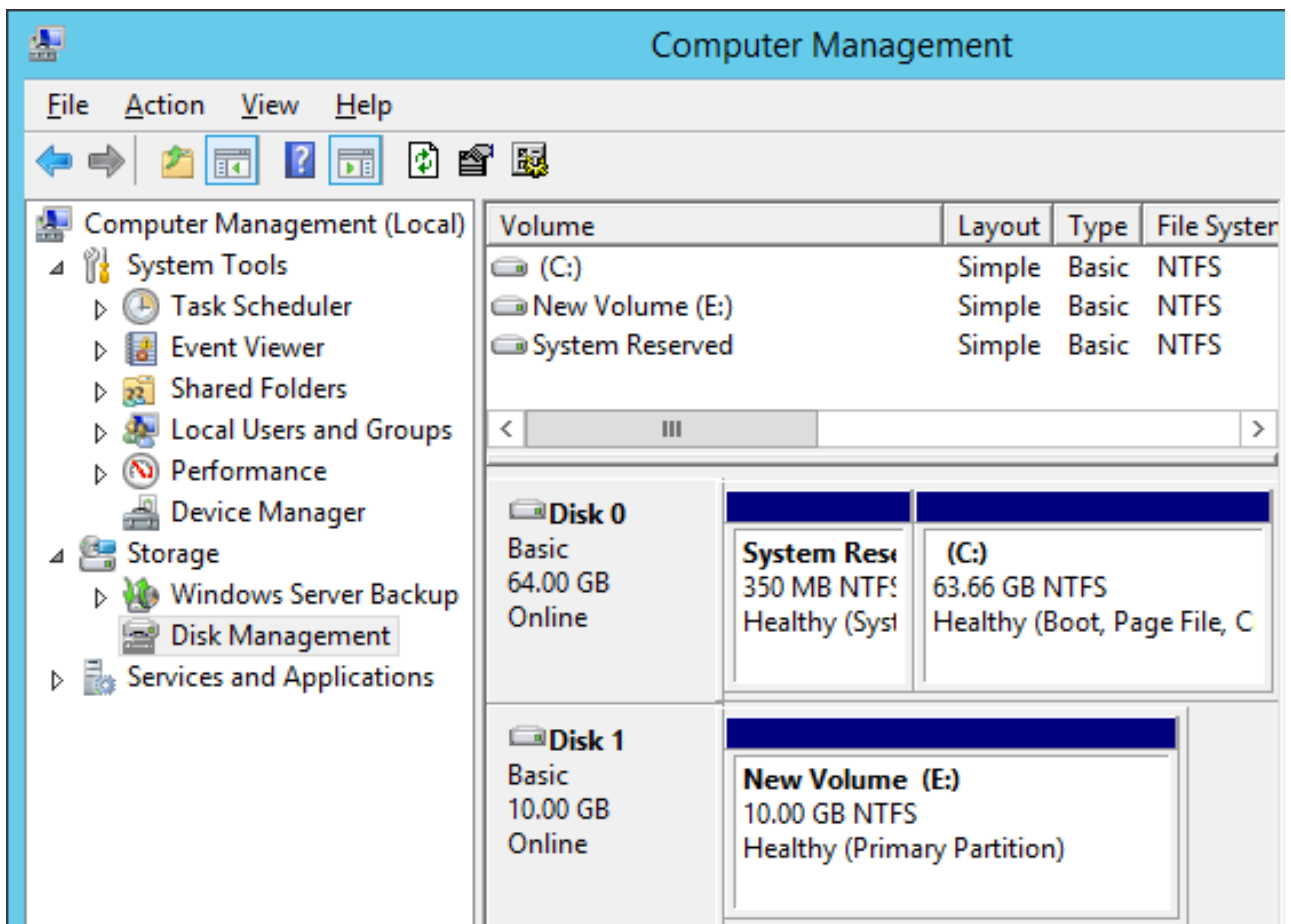
8. In the Initialize Disk window, click **OK**.



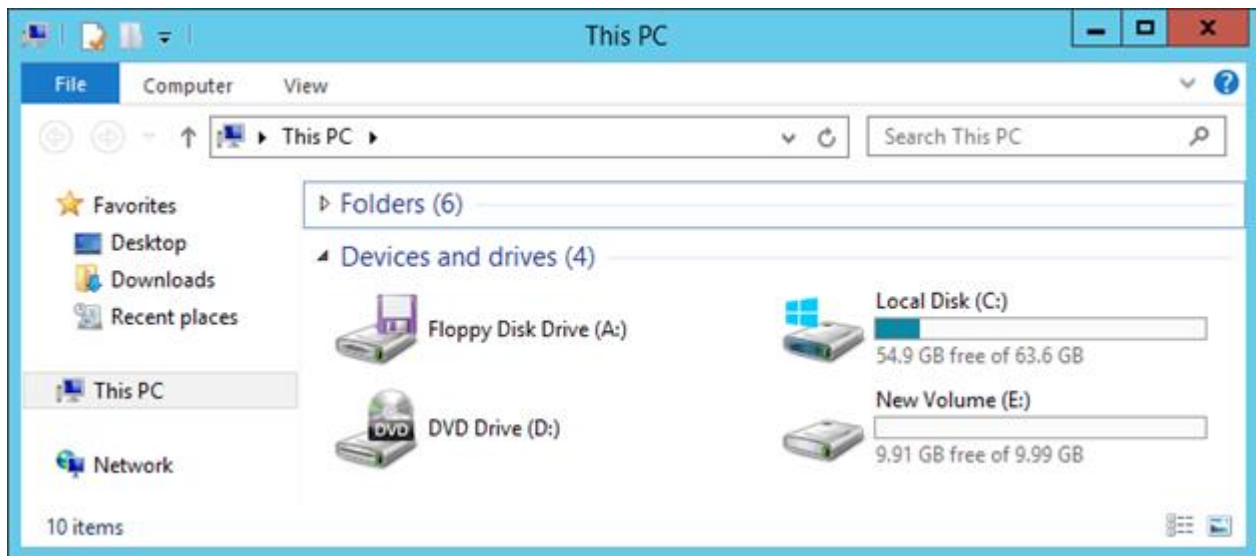
9. Right-click the disk space section, select **New Simple Volume...**, and follow the wizard's instruction to format the new disk to NTFS.



The disk state will change to **Healthy**.



The new disk will appear in Windows Explorer.



4.2.7.3 Accessing iSCSI Targets from VMware ESXi

1. In the **vSphere Client**, switch to the **Configuration** tab, and click **Storage Adapters** in the **Hardware** section.
2. If no software iSCSI adapters have been added, do so by right-clicking in the **Storage Adapters** section and selecting **Add Software iSCSI Adapter....**
3. Open the software iSCSI adapter's properties, switch to the **Static Discovery** tab and click **Add....**
4. In the **Add Static Target Server** window, enter the target's IP address and name.
5. Close the software iSCSI adapter's properties window and rescan the adapter as prompted.
6. The newly added iSCSI target will appear in the **Details** section of the software iSCSI adapter you have configured.
7. For more information, see the [VMware vSphere Storage Guide](#).

4.2.7.4 Accessing iSCSI Targets from Citrix XenServer 6.2

1. In XenCenter, switch to the **Storage** tab and click **New SR....**
2. In the **New Storage Repository** window:
 - 2.1. In the **Type** section, select the **Software iSCSI** option,
 - 2.2. In the **Name** section, provide a name or leave the default,

2.3. In the **Location** section, enter target's IP address in the **Target Host** field, click **Discover IQNs** and select the desired target, then click **Discover LUNs** and select the desired LUN.

3. Click **Finish** to format the disk.

The new storage repository will appear in XenCenter.

For more information, see the [XenCenter documentation](#).

4.2.7.5 Accessing iSCSI Targets from Microsoft Hyper-V

Note: Names of the targets to be mounted must not contain underscore characters.

1. Make sure that Microsoft iSCSI Initiator Service, MSiSCSI, is running.
2. Discover a new target portal. For example, for the portal 192.168.10.100, run:

```
PS C:\Users\Administrator>new-iscsitargetportal -targetportaladdress 192.168.10.100
Initiator Instance Name :
Initiator Portal Address :
IsDataDigest           : False
IsHeaderDigest         : False
TargetPortalAddress    : 192.168.10.100
TargetPortalPortNumber : 3260
PSComputerName        :
```

3. Connect to the desired target. For example, for the target iqn.2014-03.com.vstorage:test1

```
PS C:\Users\Administrator> connect-iscsitarget
cmdlet Connect-IscsiTarget at command pipeline position 1
Supply values for the following parameters:
NodeAddress: iqn.2014-04.com.vstorage:test1
AuthenticationType      : NONE
InitiatorInstanceName   : ROOT\ISCSIPRT\0000_0
InitiatorNodeAddress    : iqn.1991-05.com.microsoft:win-l2dj7g36n7e.sw.swsoft.com
InitiatorPortalAddress  : 0.0.0.0
InitiatorSideIdentifier : 400001370000
IsConnected             : True
IsDataDigest            : False
IsDiscovered            : True
IsHeaderDigest          : False
IsPersistent            : False
NumberOfConnections     : 1
SessionIdentifier       : fffffe00000b5e020-4000013700000005
TargetNodeAddress       : iqn.2014-04.com.vstorage:test1
TargetSideIdentifier    : 0001
```

```
PSComputerName      :
```

4. To check that the disk has been connected, run

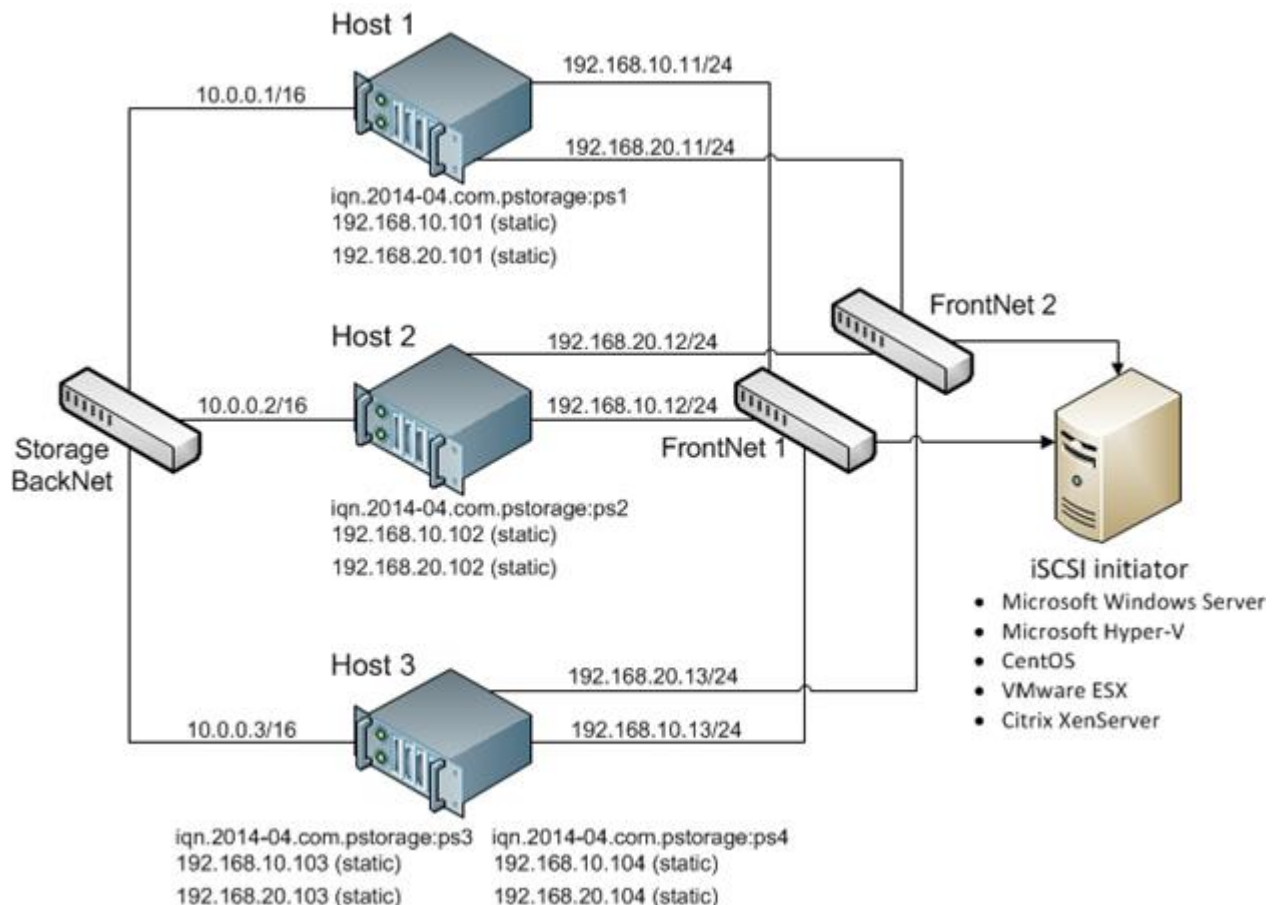
```
PS C:\Users\Administrator> get-disk
Number Friendly Name                               OperationalStatus Total Size Partition Style
-----
1      IET VIRTUAL-DISK SCSI Disk Device Offline          100 GB RAW
<...>
```

You can now initialise the newly mounted disk for use in Microsoft Hyper-V.

For more information, see [iSCSI Cmdlets in Windows PowerShell](#).

4.2.8 Configuring Multipath I/O for iSCSI Targets

Multipath I/O is a technique called to increase fault tolerance and performance by establishing multiple paths to the same iSCSI target. The figure below shows a typical multipath-enabled network configured for exporting Storage disk space over iSCSI.



In this example are three Virtuozzo Hardware Nodes working in a Storage cluster. Two Nodes host one iSCSI target each while the third hosts two iSCSI targets. Each Hardware Node is assigned a static or dynamic IP address from the FrontNet 1 and the same from the FrontNet 2. In turn, each iSCSI target is assigned a static IP address from the FrontNet 1 and a static IP address from the FrontNet 2. In case one of the frontnets fails, the iSCSI targets will still be accessible via the other one.

To enable multipath I/O for a Storage iSCSI target, assign to it multiple IP addresses from different networks using the `-a` option. For example, for a Node connected to two networks, 192.168.10.0/24 and 192.168.20.0/24, run the following command:

```
# vstorage-iscsi create -n ps1 -a 192.168.10.101 -a 192.168.20.101
```

4.2.9 Managing CHAP Accounts for iSCSI Targets

Storage allows you to restrict access to iSCSI targets by means of CHAP authentication.

To make use of CHAP authentication, you need to:

1. Create a CHAP account.
2. Create an iSCSI target bound to this CHAP account.

These actions are described in detail in the following subsections.

4.2.9.1 Creating CHAP Accounts for iSCSI Targets

To create a CHAP account, use the `vstorage-iscsi account-create` command. For example, to create the CHAP account `user1`:

```
# vstorage-iscsi account-create -u user1
Enter password:
Verify password:
```

4.2.9.2 Creating iSCSI Targets Bound to CHAP Accounts

To create a Storage iSCSI target bound to a CHAP account, use the `vstorage-iscsi create` command with the additional `-u` option. For example, create a target bound to the CHAP account `user1`:

```
# vstorage-iscsi create -n test1 -a 192.168.10.100 -u user1
IQN: iqn.2014-04.com.vstorage:test1
```

4.2.9.3 Changing the CHAP Account Password

To change the password of a CHAP account, use the `vstorage-iscsi account-set` command. For example, to change the password of the CHAP account `user1`:

```
# vstorage-iscsi account-set -u user1
Enter password:
Verify password:
```

The new password will become active after target reboot.

4.2.9.4 Listing CHAP Accounts and iSCSI Targets Assigned to Them

To list existing CHAP accounts, use the `vstorage-iscsi account-list` command. For example:

```
# vstorage-iscsi account-list
user1
```

To list Storage iSCSI targets assigned to a specific CHAP account, use the `vstorage-iscsi account-list` command with the `-u` option. For example, to list iSCSI targets assigned to the CHAP account `user1`:

```
# vstorage-iscsi account-list -u user1
iqn.2014-04.com.vstorage:test1
```

4.2.10 Managing LUN Snapshots

As with virtual machines, you can create and manage snapshots of LUNs. At that, to create a snapshot of the entire target, you will need to create snapshots of each LUN within it.

4.2.10.1 Creating LUN Snapshots

To create a snapshot of a LUN in an iSCSI target, use the `vstorage-iscsi snapshot-create` command. For example, for LUN 1 on target `iqn.2014-04.com.vstorage:test1`:

```
# vstorage-iscsi snapshot-create -t iqn.2014-04.com.vstorage:test1 -l 1
Snapshot a1f54314-bc06-40c6-a587-965feb9d85bb successfully created.
```

Note: To generate a UUID manually, use `uuidgen`.

4.2.10.2 Listing LUN Snapshots

To list snapshots for the specified LUN, use the `vstorage-iscsi snapshot-list` command. For example, for LUN 1 on target `iqn.2014-04.com.vstorage:test1`:

```
# vstorage-iscsi snapshot-list -t iqn.2014-04.com.vstorage:stor4 -l 1
CREATED          C UUID                                PARENT_UUID
2014-04-11 13:16:51 a1f54314-bc06-40c6-a587-{} 00000000-0000-0000-{}
2014-04-11 13:16:57 * 9c98b442-7482-4fd0-9c45-{} a1f54314-bc06-40c6-{}
```

In the output above, the asterisk in the column `C` indicates the current snapshot, while the column `PARENT_UUID` shows snapshot dependency or history.

4.2.10.3 Switching Between LUN Snapshots

To switch to the specified LUN snapshot, use the `vstorage-iscsi snapshot-switch` command. For example:

```
# vstorage-iscsi snapshot-switch -u a1f54314-bc06-40c6-a587-965feb9d85bb
```

After you switch to a snapshot, the current LUN image will be removed.

Note: You can only switch between snapshots, if the LUN is offline.

4.2.10.4 Viewing LUN Snapshot Information

To view information about the specified snapshot, use the `vstorage-iscsi snapshot-info` command. For example:

```
# vstorage-iscsi snapshot-info -u 9c98b442-7482-4fd0-9c45-9259374ca84e
Target: iqn.2014-04.com.vstorage:stor4
LUN: 1
Created: 2014-04-11 13:16:57
Parent: 00000000-0000-0000-0000-000000000000}
{a1f54314-bc06-40c6-a587-965feb9d85bb}
{9c98b442-7482-4fd0-9c45-9259374ca84e
Description: None
```


4.2.10.5 Deleting LUN Snapshots

To delete the specified LUN snapshot, use the `vstorage-iscsi snapshot-delete` command. For example:

```
# vstorage-iscsi snapshot-delete -u a1f54314-bc06-40c6-a587-965feb9d85bb
```

If the snapshot has no any children, it will be deleted. If the snapshot has a single child, it will be merged to that child.

Note the following:

- You can only delete offline snapshots.
- Deleting a snapshot that has multiple children is currently not supported.

4.3 Accessing Storage Clusters via S3-like Object Storage

Storage can export data via an Amazon S3-compatible API, enabling service providers to:

- Run S3-based services in their own Storage infrastructures.
- Sell S3-based storage-as-a-service to customers along with Storage.

S3 support expands the functionality of Storage and requires a working Storage cluster.

4.3.1 About Object Storage

Object storage is a storage architecture that enables managing data as objects (like in key-value storage) as opposed to files in file systems or blocks in block storage. Except data, each object has name (i.e. full path to object) that describes it and also a unique identifier that allows finding said object in the storage. Object storage is optimized for storing billions of objects, in particular for application back-end storage, static web content hosting, online storage services, big data, and backups. All of these uses are enabled by object storage thanks to a combination of very high scalability and data availability and consistency.

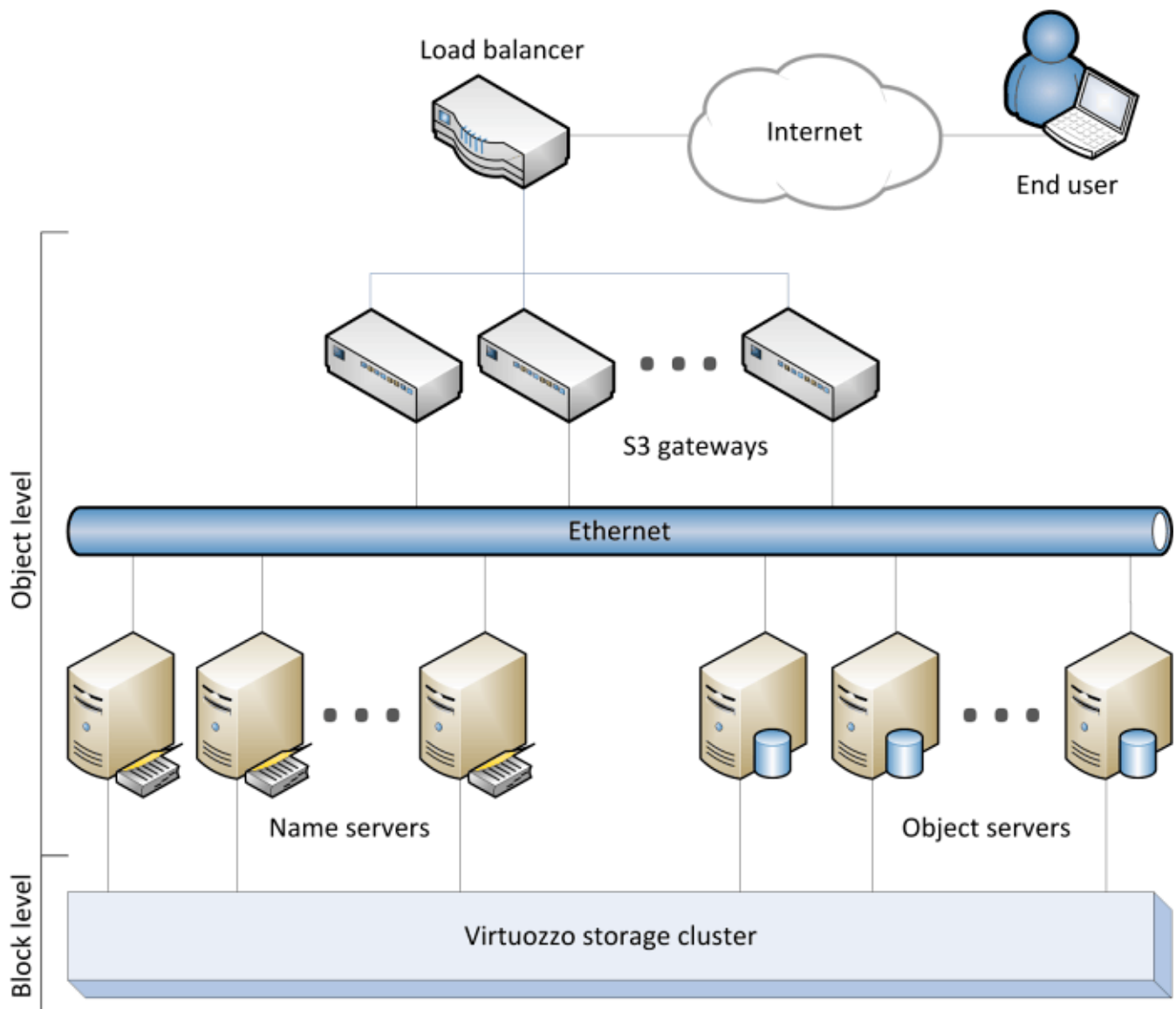
Compared to other types of storage, the key difference of S3 object storage is that parts of an object cannot be modified, so if the object changes a new version of it is spawned instead. This approach is extremely important for maintaining data availability and consistency. First of all, changing an object as a whole eliminates the issue of conflicts. That is, the object with the latest timestamp is considered to be the current

version and that is it. As a result, objects are always consistent, i.e. their state is relevant and appropriate.

Another feature of object storage is eventual consistency. Eventual consistency does not guarantee that reads are to return the new state after the write has been completed. Readers can observe the old state for an undefined period of time until the write is propagated to all the replicas (copies). This is very important for storage availability as geographically distant data centers may not be able to perform data update synchronously (e.g., due to network issues) and the update itself may also be slow as awaiting acknowledges from all the data replicas over long distances can take hundreds of milliseconds. So eventual consistency helps hide communication latencies on writes at the cost of the probable old state observed by readers. However, many use cases can easily tolerate it.

4.3.1.1 Object Storage Infrastructure

The infrastructure of Object Storage consists of the following entities: object servers, name servers, S3 gateways, and the block level backend.



- Object server (OS) stores actual object data (contents) received from S3 gateway. It stores its own data in block storage with built-in high availability.
- Name server stores object metadata received from S3 gateway. Metadata includes object name, size, ACL (access control list), location, owner, and such. Name server (NS) also stores its own data in block storage with built-in high availability.
- S3 gateway (GW) is a data proxy between object storage services and end users. It receives and handles Amazon S3 protocol requests and uses nginx Web server for external connections. S3 gateway handles S3 user authentication and ACL checks. It has no data of its own (i.e. is stateless).
- Block level backend is block storage with high availability of services and data. Since all object storage

services run on hosts, no virtual environments (or respective licenses) are required for object storage.

4.3.1.2 Object Storage Overview

In terms of S3 object storage, a file is an object. Object servers store each object loaded via the S3 API as a pair of entities:

- Object names and associated object metadata stored on an NS. An object name in the storage is determined based on request parameters and bucket properties in the following way:
 - If bucket versioning is disabled, an object name in the storage contains bucket name and object name taken from an S3 request.
 - If bucket versioning is enabled, an object name also contains a list of object versions.
- Object data stored on an OS. The directory part of an object name determines an NS to store it while the full object name determines an OS to store the object data.

4.3.1.2.1 Interaction between S3 Storage and the Storage Cluster

An S3 storage cluster requires a working Storage cluster on each of S3 cluster nodes. Storage provides content sharing, strong consistency, data availability, reasonable performance for random I/O operations, and high availability for storage services. In storage terms, S3 data is a set of files (see *Object Server* on page 70) that the Storage file system layer (vstorage-mount) does not interpret in any way.

4.3.1.2.2 Multipart Uploads

A name of a multipart upload is defined by a pattern similar to that of an object name but the object that corresponds to it contains a table instead of file contents. The table contains index numbers of parts and their offsets within the file. This allows to upload parts of a multi-part upload in parallel (recommended for large files). The maximum number of parts is 10,000.

4.3.1.3 Object Storage Components

This section familiarises you with S3 storage components—gateways, object servers, and name servers—and describes S3 management tools and service buckets.

4.3.1.3.1 Gateway

Gateway performs the following functions:

- Receives S3 requests from the web server (via nginx and FastCGI).
- Parses S3 packets and validates S3 requests (checks fields of a request and XML documents in its body).
- Authenticates S3 users.
- Validates access permissions to buckets and objects using ACL.
- Collects statistics on the number of various requests as well as the amount of the data received and transmitted.
- Determines paths to NS and OS storing the object's data.
- Inquires names and associated metadata from NS.
- Receives links to objects stored on Oses by requesting the name from NSes.
- Caches metadata and ACL of S3 objects received from NSes as well as the data necessary for user authentication also stored on the NSes.
- Acts as a proxy server when clients write and read object data to and from the Oses. Only the requested data is transferred during read and write operations. For example, if a user requests to read 10MB from a 1TB object, only said 10MB will be read from the OS.

S3 gateway consists of incoming requests parser, type-dependent asynchronous handlers of these requests, and an asynchronous handler of the interrupted requests that require completion (complex operations such as bucket creation or removal). Gateway does not store its state data in the long-term memory. Instead, it stores all the data needed for S3 storage in the object storage itself (on NS and OS).

4.3.1.3.2 Name Server

Name server performs the following functions:

- Stores object names and metadata.
- Provides the API for pasting, deleting, listing object names and changing object metadata.

Name server consists of data (i.e. object metadata), object change log, an asynchronous garbage collector, and asynchronous handlers of incoming requests from different system components.

The data is stored in a B-tree where to each object's name corresponds that object's metadata structure. S3 object metadata consists of three parts: information on object, user-defined headers (optional), and ACL for the object. Files are stored in the corresponding directory on base shared storage (i.e. Storage).

Name server is responsible for a subset of S3 cluster object namespace. Each NS instance is a userspace process that works in parallel with other processes and can utilize up to one CPU core. The optimal number of name servers are 4-10 per node. We recommend to start with creating 10 instances per node during cluster creation to simplify scalability later. If your node has CPU cores that are not utilized by other storage services, you can create more NSes to utilize these CPU cores.

4.3.1.3.3 Object Server

Object server performs the following functions:

- Stores object data in pools (data containers).
- Provides an API for creating, reading (including partial reads), writing to, and deleting objects.

Object server consists of the following:

- Information on object's blocks stored on this OS
- Containers that store object data
- Asynchronous garbage collector that frees container sections after object delete operations

Object data blocks are stored in pools. The storage uses 12 pools with blocks the size of the power of 2, ranging from 4 kilobytes to 8 megabytes. A pool is a regular file on block storage made of fixed-size blocks (regions). In other words, each pool is an extremely large file designed to hold objects of specific size: the first pool is for 4KB objects, the second pool is for 8KB objects, etc.

Each pool consists of a block with system information, and fixed-size data regions. Each region contains has a free/dirty bit mask. The region's data is stored in the same file with an object's B-tree. It provides atomicity during the block's allocation and deallocation. Every block in the region contains a header and object's data. The header stores the ID of an object to which the data belong. The ID is required for a pool-level defragmentation algorithm that does not have an access to the object's B-tree. A pool to store an object is chosen depending on object size.

For example, a 30KB object will be placed into the pool for 32KB objects and will occupy a single 32KB object. A 129KB object will be split into one 128KB part and one 1KB part. The former will be placed in the pool for 128KB objects while the latter will go to the pool for 4KB objects. The overhead may seem significant in case

of small objects as even a 1-byte object will occupy a 4KB block. In addition, about 4KB of metadata per object will be stored on NS. However, this approach allows achieving the maximum performance, eliminates free space fragmentation, and offers guaranteed object insert performance. Moreover, the larger the object, the less noticeable the overhead. Finally, when an object is deleted, its pool block is marked free and can be used to store new objects.

Multi-part objects are stored as parts (each part being itself an object) that may be stored on different object servers.

4.3.1.3.4 S3 Management Tools

Object storage has two tools:

- `ostor-ct1` for configuring storage components
- `ostor-s3-admin` for user management, an application that allows to create, edit, and delete S3 user accounts as well as manage account access keys (create and delete paired S3 access key IDs and S3 secret access keys)

4.3.1.3.5 Service Bucket

The service bucket stores service and temporary information necessary for the S3 storage. This bucket is only accessible by the S3 admin (while the system admin would need access keys created with the `ostor-s3-admin` tool).

4.3.1.4 Data Interchange

In object storage, every service has a 64-bit unique identifier. At the same time, every object has a unique name. The directory part of an object's name determines a name server to store it, and the full object's name—an object server to store the object's data. Name and object server lists are stored in a `vstorage` cluster directory intended for object storage data and available to anyone with a cluster access. This directory includes subdirectories that correspond to services hosted on name and object servers. The names of subdirectories match hexadecimal representations of the service's ID. In each service's subdirectory, there is a file containing an ID of a host that runs the service. Thus, with the help of a gateway, a system component with a cluster access can discover an ID of a service, detect its host, and send a request to it.

S3 gateway handles data interchange with the following components:

- Clients via a web server. Gateway receives S3 requests from users and responds to them.
- Name servers. Gateway creates, deletes, changes the names that correspond to S3 buckets or objects, checks their existence, and requests name sets of bucket lists.
- Object servers in the storage. Gateway sends data altering requests to object and name servers.

4.3.1.4.1 Data Caching

To enable efficient data use in object storage, all gateways, name servers, and object servers cache the data they store. Name and object servers both cache B-trees.

Gateways store and cache the following data received from name services:

- Lists of paired user IDs and e-mails.
- Data necessary for user authentication: access key IDs and secret access keys. For more information on their semantics, consult the Amazon S3 documentation.
- Metadata and bucket's ACLs. The metadata contains its epoch, current version identifier and transmits it to NS to check if the gateway has the latest version of the metadata.

4.3.1.5 Operations on Objects

This section familiarizes you with operations S3 storage processes: operations requests; create, read, and delete operations.

4.3.1.5.1 Operation Requests

To create, delete, read an object or alter its data, S3 object storage must first request one if these operations and then perform it. The overall process of requesting and performing an operation consists of the following:

1. Requesting user authentication data. It will be stored on a name server in a specific format (see Service Buckets). To receive data (identifier, e-mail, access keys), a request with a lookup operation code is sent to an appropriate name server.
2. Authenticating the user.
3. Requesting bucket's and object's metadata. To receive it, another request with a lookup operation code is sent to the name server that stores names of objects and buckets.

4. Checking user's access permissions to buckets and objects.
5. Performing the requested object operation: creating, editing or reading data or deleting the object.

4.3.1.5.2 Create Operation

To create an object, gateway sends the following requests:

1. Request with a guard operation code to a name server. It creates a guard with a timer which will check after a fixed time period if an object with the data was indeed created. If it was not, the create operation will fail and the guard will request the object server to delete the object's data if some were written. After that the guard is deleted.
2. Request with a create operation code to an object server followed by fixed-size messages containing the object's data. The last message includes an end-of-data flag.
3. Another request with a create operation code to the name server. The server checks if the corresponding guard exists and, if it does not, the operation fails. Otherwise, the server creates a name and sends a confirmation of successful creation to the gateway.

4.3.1.5.3 Read Operation

To fulfill an S3 read request, gateway determines an appropriate name server's identifier based on the name of a directory and corresponding object server's identifier based on the object's full name. To perform a read operation, gateway sends the following requests:

1. Request with a read operation code to an appropriate name server. A response to it contains a link to an object.
2. Request to an appropriate object server with a read operation code and a link to an object received from the name server.

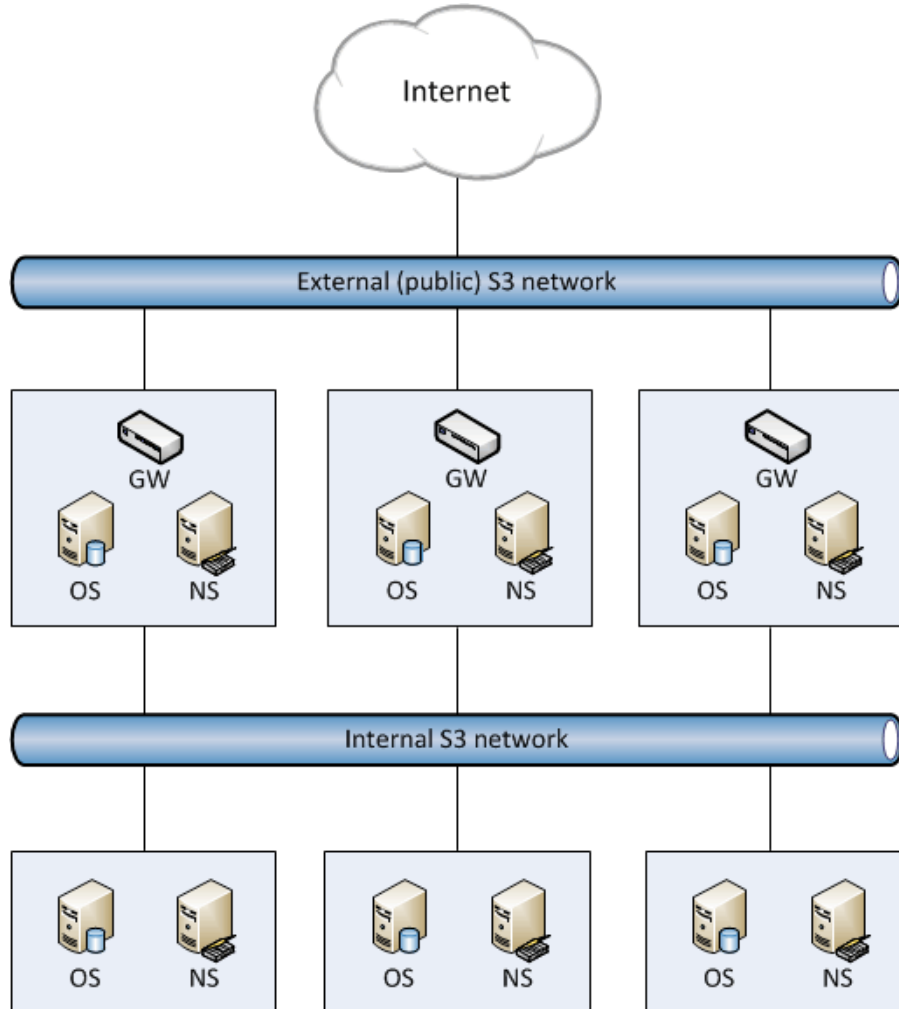
To fulfill the request, object server transmits fixed-size messages with the object's data to the gateway. The last message contains an end-of-data flag.

4.3.1.5.4 Delete Operation

To delete an object (and its name) from the storage, gateway determines a name server's identifier based on the directory's part of a name and sends a request with a delete operation code to the server. In turn, the name server removes the name from its structures and sends the response. After some time, the garbage collector removes the corresponding object from the storage.

4.3.2 Deploying Object Storage

This chapter describes deploying object storage on top of a ready Storage cluster. As a result you will create a setup like shown on the figure. Note that not all cluster nodes have to run object storage services. The choice should be based on workload and hardware configurations.



To set up object storage services, do the following:

1. Plan the S3 network. Like a Storage cluster, an object storage cluster needs two networks:
 - An internal network in which NS, OS, and GW will interact. These services will generate traffic similar in amount to the total (incoming and outgoing) S3 user traffic. If this is not going to be much, it is reasonable to use the same internal network for both object storage and Storage. If, however, you expect that object storage traffic will compete with Storage traffic, it is reasonable to have S3 traffic go through the user data network (i.e. datacenter network). Once you choose a network for S3 traffic, you determine which IP addresses can be used while adding cluster nodes.
 - An external (public) network through which end users will access the S3 storage. Standard HTTP and HTTPS ports must be open in this network.

An object storage cluster is almost completely independent on base block storage (like all access points, including virtual environments and iSCSI). Object and name servers keep their data in the Storage cluster in the same way as virtual environments, iSCSI, and other services do. So the OS and NS services depend on `vstorage-mount` (client) and can only work when the cluster is mounted. Unlike them, gateway is a stateless service that has no data. It is thus independent on `vstorage-mount` and can theoretically be run even on nodes where the Storage cluster is not mounted. However, for simplicity, we recommend creating gateways on nodes with object and name servers.

Object and name servers also utilize the standard high availability means of Storage (i.e. the shaman service). Like virtual environments and iSCSI, OS and NS are subscribed to HA cluster events. However, unlike other services, S3 cluster components cannot be managed (tracked and relocated between nodes) by shaman. Instead, this is done by the S3 configuration service that is subscribed to HA cluster events and notified by shaman whether nodes are healthy and can run services. For this reason, S3 cluster components are not shown in `shaman top` output.

Gateway services which are stateless are never relocated and their high availability is not managed by the Storage cluster. Instead, a new gateway service is created when necessary.

2. Make sure that each node that will run OS and NS services is in the high availability cluster. You can add nodes to HA cluster with the `shaman join` command.
3. Install the `vstorage-ostor` package on each cluster node.

```
# yum install vstorage-ostor
```

4. Create a cluster configuration on one of the cluster nodes where object storage services will run. It is recommended to create 10 NS and 10 OS services per each node. For example, if you are going to use five nodes, you will need 50 NS and 50 OS. Run this command on the first cluster node.

```
# ostor-ctl create -r /var/lib/ostor/configuration -n <IP_addr>
```

Where <IP_addr> is the node's IP address (that belongs to the internal S3 network) that the configuration service will listen on.

You will be asked to enter and confirm a password for the new object storage (it can be the same as your Storage cluster password). You will need this password to add new nodes.

The configuration service will store the cluster configuration locally in `/var/lib/ostor/configuration`. In addition, <IP_addr> will be stored in `/<storage_mount>/<ostor_dir>/control/name` (<ostor_dir> is the directory in the cluster with object storage service files). If the first configuration service fails (and the `ostor-ctl get-config` command stops working), replace the IP address in `/<storage_mount>/<ostor_dir>/control/name` with that of a node running a healthy configuration service (created on step 6).

5. Launch the configuration service.

```
# systemctl start ostor-cfgd.service
# systemctl enable ostor-cfgd.service
```

6. Add at least two more configuration services for redundancy (to have at least three in total). A configuration service is only required for adding and removing nodes to and from the S3 cluster and does not affect operation of S3 services and their high availability. So a failure of a configuration service is not critical for the S3 cluster. However, it is still undesirable and we recommend creating several configuration services so at least one is always up.

To add one more configuration service, run the following commands on a node where object storage services will run. Repeat to create the required number of configuration services.

```
# ostor-ctl join -n <remote_IP_addr> -a <local_IP_addr>
# systemctl start ostor-cfgd.service
# systemctl enable ostor-cfgd.service
```

Where <remote_IP_addr> is <IP_addr> from step 4.

Each added configuration service will store the cluster configuration locally in `/var/lib/ostor/configuration`.

7. Initialize the new object storage on the first node. The <ostor_dir> directory will be created in the root of your cluster.

```
# ostor-ctl init-storage -n <IP_addr> -s <cluster_mount_point>
```

You will need to provide the IP address and object storage password specified on step 3.

8. Add to the DNS public IP addresses of nodes that will run GW services. You can configure the DNS to enable access to your object storage via a hostname, and to have the S3 endpoint receive virtual hosted-style REST API requests with URIs like <http://bucketname.s3.example.com/objectname>.

After configuring DNS, make sure that DNS resolver for your S3 access point works from client machines.

Note: Only buckets with DNS-compatible names can be accessed with virtual hosted-style requests. For more details, see *Bucket and Key Naming Policies* on page 89.

Below is an example of a DNS zones configuration file for the BIND DNS server:

```
;$Id$
$TTL 1h @ IN SOA ns.example.com. s3.example.com. (
    2013052112 ; serial
    1h ; refresh
    30m ; retry
    7d ; expiration
    1h ) ; minimum
    NS ns.example.com.
$ORIGIN s3.example.com
h1 IN A 10.29.1.95
    A 10.29.0.142
    A 10.29.0.137
* IN CNAME @
```

This configuration instructs the DNS to redirect all requests with URI <http://s3.example.com> and its subdomains (http://*.s3.example.com/) to one of the endpoints listed in resource record h1 (10.29.1.95, 10.29.0.142 or 10.29.0.137) in a cyclic (round-robin) manner.

9. Add nodes where object storage services will run to the configuration.

Note: Adding nodes to existing clusters is done in the similar way by performing steps 8-12.

To do this, run the `ostor-ctl add-host` command on every such node:

```
# ostor-ctl add-host -r /var/lib/ostor/configuration --hostname <name> --roles OBJ
```

You will need to provide the object storage password set on step 3.

Note: If you want the object storage agent service to listen on an internal IP address, add the option `-H`

<internal_IP_address> to the command above.

10. Create a new S3 volume with the desired number of NS and OS:

```
# ostor-ctl add-vol --type OBJ -s <cluster_mount_point> --os-count <OS_num> \
--ns-count <NS_num> --vstorage-attr "failure-domain=host,tier=0,replicas=3"
```

Where:

- <NS_num> and <OS_num> are the numbers of NS and OS
- failure-domain=host, tier=0, replicas=3 parameters set volume's failure domain, tier, and redundancy mode (for more details, see *Cluster Parameters Overview* on page 28).

The command will return the ID for the created volume. You will need it on the next step.

11. Create S3 gateway instances on chosen nodes with Internet access and external IP addresses. It is recommended to create 4 GWs per node.

Note: For security reasons, make sure that only `nginx` can access the external network and that S3 gateways only listen on internal IP addresses.

```
# ostor-ctl add-s3gw -a <internal_IP_address>:<port> -V <volume_ID>
```

Where:

- <internal_IP_address> is the internal IP address of the node with the gateway
- <port> (mandatory) is an unused port unique for each GW instance on the node
- <volume_ID> is the ID of the volume you created on the previous step (it can also be obtained from `ostor-ctl get-config`)

For example:

```
# ostor-ctl add-s3gw -a 127.0.0.1:9001 -V 0100000000000001
# ostor-ctl add-s3gw -a 127.0.0.1:9002 -V 0100000000000001
# ostor-ctl add-s3gw -a 127.0.0.1:9003 -V 0100000000000001
# ostor-ctl add-s3gw -a 127.0.0.1:9004 -V 0100000000000001
```

12. Launch object storage agent on each cluster node added to the object storage configuration.

```
# systemctl start ostor-agentd.service
# systemctl enable ostor-agentd.service
```

13. Make sure NS and OS services are bound to the nodes.

By default agents will try to assign NS and OS services to the nodes automatically in a round-robin manner. However, manual assignment is required if a new host has been added to the configuration, or if the current configuration is not optimized (for details, see *Manually Binding Services to Nodes* on page 80).

You can check the current binding configuration with the `ostor-ctl agent-status` command. For example:

```
# ostor-ctl agent-status
TYPE      SVC_ID          STATUS      UPTIME  HOST_ID          ADDR
S3GW      800000000000009 ACTIVE      527     fcbf5602197245da 127.0.0.1:9090
S3GW      800000000000008 ACTIVE      536     4f0038db65274507 127.0.0.1:9090
S3GW      800000000000007 ACTIVE      572     958e982fcc794e58 127.0.0.1:9090
OS        100000000000005 ACTIVE      452     4f0038db65274507 10.30.29.124:39746
OS        100000000000004 ACTIVE      647     fcbf5602197245da 10.30.27.69:56363
OS        100000000000003 ACTIVE      452     4f0038db65274507 10.30.29.124:52831
NS        080000000000002 ACTIVE      647     fcbf5602197245da 10.30.27.69:56463
NS        080000000000001 ACTIVE      452     4f0038db65274507 10.30.29.124:53044
NS        080000000000000 ACTIVE      647     fcbf5602197245da 10.30.27.69:37876
```

14. On each node with GW instances, install `nginx` that will serve S3 requests from end users:

```
# yum install nginx
```

Use the `ostor-configure-nginx` tool to configure `nginx` for S3. There are two configuration options:

- create that creates a new configuration
- update that updates the existing upstream configuration; use it after changing GW configuration.

For the initial `nginx` configuration, use `create`. For example, for HTTP:

```
# ostor-configure-nginx create -D s3.mydomain.com -p 80
```

Where `s3.mydomain.com` is the S3 endpoint domain and `80` is the port for `nginx` to listen to.

To configure HTTP with an SSL certificate for the S3 endpoint domain and its subdomains, specify the certificate and key. For example:

```
# ostor-configure-nginx create -D s3.mydomain.com -p 443 **ssl **ssl-cert file.cert \
**ssl-key file.key
```

The configuration file will be created at `/etc/nginx/conf.d/ostor-s3.conf`. It will handle FastCGI redirection to local GW instances.

15. Launch `nginx`:

```
# systemctl start nginx.service
# systemctl enable nginx.service
```

16. Add nodes that are in the HA cluster but run no S3 services to the S3 cluster. That is, make sure that all nodes in the HA cluster are also in the S3 cluster. This is required for high availability to work correctly.

```
# ostor-ctl add-host -n <IP_addr>
# systemctl start ostor-agentd.service
# systemctl enable ostor-agentd.service
```

The object storage is deployed. Now you can add S3 users with the `ostor-s3-admin` tool as described in [Creating S3 Users](#) on page 82.

To check that installation has been successful or just monitor object storage status, use the `ostor-ctl get-config` command. For example:

```
# ostor-ctl get-config
07-08-15 11:58:45.470 Use configuration service 'ostor'
SVC_ID          TYPE  URI
8000000000000006  S3GW  svc://1039c0dc90d64607/?address=127.0.0.1:9000
0800000000000000  NS    vstorage://cluster1/ostor/services/0800000000000000
1000000000000001  OS    vstorage://cluster1/ostor/services/1000000000000001
1000000000000002  OS    vstorage://cluster1/ostor/services/1000000000000002
1000000000000003  OS    vstorage://cluster1/ostor/services/1000000000000003
1000000000000004  OS    vstorage://cluster1/ostor/services/1000000000000004
8000000000000009  S3GW  svc://7a1789d20d9f4490/?address=127.0.0.1:9000
800000000000000c  S3GW  svc://7a1789d20d9f4490/?address=127.0.0.1:9090
```

4.3.2.1 Manually Binding Services to Nodes

When deploying object storage, you can manually bind services to nodes with the `ostor-ctl bind` command. You will need to specify the target node ID and one or more service IDs to bind to it. For example, the command:

```
# ostor-ctl bind -H 4f0038db65274507 -S 0800000000000001 -S 1000000000000003 \
-S 1000000000000005
```

binds services with IDs `8000000000000001`, `1000000000000003`, and `1000000000000005` to a host with ID `4f0038db65274507`.

A service can only be bound to a host that is connected to the shared storage which stores that service's data. That is, the cluster name in service URI must match the cluster name in host URI.

For example, in a configuration with two shared storages `stor1` and `stor2` (see below) services with URIs starting with `vstorage://stor1` can only be bound to hosts `host510` and `host511` while services with URIs

starting with `vstorage://stor2` can only be bound to hosts `host512` and `host513`.

```
# ostor-ctl get-config
SVC_ID      TYPE  URI
0800000000000000    NS  vstorage://stor1/s3-data/services/0800000000000000
080000000000000001    NS  vstorage://stor1/s3-data/services/08000000000000001
080000000000000002    NS  vstorage://stor2/s3-data/services/08000000000000002
100000000000000003    OS  vstorage://stor1/s3-data/services/10000000000000003
100000000000000004    OS  vstorage://stor2/s3-data/services/10000000000000004
100000000000000005    OS  vstorage://stor1/s3-data/services/10000000000000005
HOST_ID     HOSTNAME  URI
0fcbf5602197245da  host510:2530  vstorage://stor1/s3-data
4f0038db65274507  host511:2530  vstorage://stor1/s3-data
958e982fcc794e58  host512:2530  vstorage://stor2/s3-data
953e976abc773451  host513:2530  vstorage://stor2/s3-data
```

4.3.3 Managing S3 Users

The concept of S3 user is one of the base concepts of object storage along with those of object and bucket (container for storing objects). Amazon S3 protocol uses permissions model based on access control lists (ACLs) where each bucket and each object is assigned an ACL that lists all users with access to the given resource and the type of this access (read, write, read ACL, write ACL). The list of users includes entity owner assigned to every object and bucket at creation. Entity owner has extra rights compared to other users, for example, bucket owner is the only one who can delete that bucket.

User model and access policies implemented in Object Storage comply with the Amazon S3 user model and access policies.

User management scenarios in Object Storage are largely based on the Amazon Web Services user management and include the following operations: create, query, delete users as well as generate, revoke user access key pairs.

You can manage users with the `ostor-s3-admin` tool. To do this, you will need to know the ID of the volume that the users are in. You can obtain it with the `ostor-ctl get-config` command. For example:

```
# ostor-ctl get-config -n 10.94.97.195
VOL_ID      TYPE  STATE
010000000000000002  OBJ  READY
...
```

Note: As `ostor-s3-admin` commands are assumed to be issued by object storage administrators, they do not include any authentication or authorization checks.

4.3.3.1 Creating S3 Users

You can generate a unique random S3 user ID and an access key pair (S3 Access Key ID, S3 Secret Access Key) using the `ostor-s3-admin create-user` command. You need to specify a user email. For example:

```
# ostor-s3-admin create-user -e user@email.com -V 0100000000000002
UserEmail:user@email.com
UserId:a49e12a226bd760f
KeyPair[0]:S3AccessKeyId:a49e12a226bd760fGHQ7
KeyPair[0]:S3SecretAccessKey:HSDu2DA00JNGjnRcAhLKfhrvlymz0VdLPsCK2dcq
Flags:none
```

S3 user ID is a 16-digit hexadecimal string. The generated access key pair is used to sign requests to the S3 object storage according to the Amazon S3 Signature Version 2 authentication scheme.

4.3.3.2 Listing S3 Users

You can list all object storage users with the `ostor-s3-admin query-users` command. Information for each user can take one or more sequential rows in the table. Additional rows are used to lists S3 access key pairs associated with the user. If the user does not have any active key pairs, minus signs are shown in the corresponding table cells. For example:

```
# ostor-s3-admin query-users -V 0100000000000002
      S3 USER ID      S3 ACCESS KEY ID      S3 SECRET ACCESS KEY  S3 USER EMAIL
bf0b3b15eb7c9019  bf0b3b15eb7c9019I36Y      ***                    user2@abc.com
d866d9d114cc3d20  d866d9d114cc3d20G456      ***                    user1@abc.com
                   d866d9d114cc3d20D8EW      ***
e86d1c19e616455      -                          -                      user3@abc.com
```

To output the list in XML, use the `-x` option; to output secret keys, use the `-a` option. For example:

```
# ostor-s3-admin query-users -V 0100000000000002 -a -X
<?xml version="1.0" encoding="UTF-8"?><QueryUsersResult><Users><User><Id>a49e12a226bd760f</Id><Email>user@email.com</Email><Keys><OwnerId>0000000000000000</OwnerId><KeyPair><S3AccessKeyId>a49e12a226bd760fGHQ7</S3AccessKeyId><S3SecretAccessKey>HSDu2DA00JNGjnRcAhLKfhrvlymz0VdLPsCK2dcq</S3SecretAccessKey></KeyPair></Keys></User><User><Id>d7c53fc1f931661f</Id><Email>user@email.com</Email><Keys><OwnerId>0000000000000000</OwnerId><KeyPair><S3AccessKeyId>d7c53fc1f931661fZLIV</S3AccessKeyId><S3SecretAccessKey>JL7gt10H873zR0Fzv80h9ZuA6JtCVnkgV71ET6ET</S3SecretAccessKey></KeyPair></Keys></User></Users></QueryUsersResult>
```

4.3.3.3 Querying S3 User Information

To display information about the specified user, use the `ostor-s3-admin query-user-info` command. You need to specify either the user email (`-e`) or S3 ID (`-i`). For example:

```
# ostor-s3-admin query-user-info -e user@email.com -V 0100000000000002
Query user: user id=d866d9d114cc3d20, user email=user@email.com
Key pair[0]: access key id=d866d9d114cc3d20G456,
secret access key=5EAne6PLL1jxprouRqq8hmfONMfgrJcOwbowCoTt
Key pair[1]: access key id=d866d9d114cc3d20D8EW,
secret access key=83tTsNAuuRyoBBqhxFqHAC60dhKHtTCCkQe54zu
```

4.3.3.4 Disabling S3 Users

You can disable a user with the `ostor-s3-admin disable-user` command. You need to specify either the user email (`-e`) or S3 ID (`-i`). For example:

```
# ostor-s3-admin disable-user -e user@email.com -V 0100000000000002
```

4.3.3.5 Deleting S3 Users

You can delete existing object storage users with the `ostor-s3-admin delete-user` command. Users who own any buckets cannot be deleted, so delete user's buckets first. You need to specify either the user email (`-e`) or S3 ID (`-i`). For example:

```
# ostor-s3-admin delete-user -i bf0b3b15eb7c9019 -V 0100000000000002
Deleted user: user id=bf0b3b15eb7c9019
```

4.3.3.6 Generating S3 User Access Key Pairs

You can generate a new access key pair for the specified user with the `ostor-s3-admin gen-access-key` command. The maximum of 2 active access key pairs are allowed per user (same as with the Amazon Web Services). You need to specify either the user email (`-e`) or S3 ID (`-i`). For example:

```
# ostor-s3-admin gen-access-key -e user@email.com -V 0100000000000002
Generate access key: user id=d866d9d114cc3d20, access key id=d866d9d114cc3d20D8EW,
secret access key=83tTsNAuuRyoBBqhxFqHAC60dhKHtTCCkQe54zu
```

Note: It is recommended to periodically revoke old and generate new access key pairs.

4.3.3.7 Revoking S3 User Access Key Pairs

You can revoke the specified access key pair of the specified user with the `ostor-s3-admin revoke-access-key` command. You need to specify the access key in the key pair you want to delete as well as the user email or S3 ID. For example:

```
# ostor-s3-admin revoke-access-key -e user@email.com -k de86d1c19e616455YIPU -V 0100000000000002
Revoke access key: user id=de86d1c19e616455, access key id=de86d1c19e616455YIPU
```

Note: It is recommended to periodically revoke old and generate new access key pairs.

4.3.4 Managing Object Storage Buckets

All objects in Amazon S3-like storage are stored in containers named *buckets*. Buckets are addressed by names that are unique in the given object storage, so an S3 user of that object storage cannot create a bucket that has the same name as a different bucket in the same object storage. Buckets are used to:

- Group and isolate objects from those in other buckets.
- Provide ACL management mechanisms for objects in them.
- Set per-bucket access policies, for example, versioning in the bucket.

You can manage buckets with the `ostor-s3-admin` tool as well as S3 API third-party S3 browsers like CyberDuck or DragonDisk. The `ostor-s3-admin` tool is to be used by object storage administrators, so these commands do not include any authentication or authorization checks. It is recommended to use standard Amazon S3 API commands first.

To manage buckets via CLI, you will need to know the ID of the volume that the buckets are in. You can obtain it with the `ostor-ctl get-config` command. For example:

```
# ostor-ctl get-config -n 10.94.97.195
VOL_ID          TYPE          STATE
0100000000000002  OBJ          READY
<...>
```

Important: The change and delete bucket operations are forced on the object storage. These commands are not part of the standard S3 API and may break integration with external billing and accounting systems.

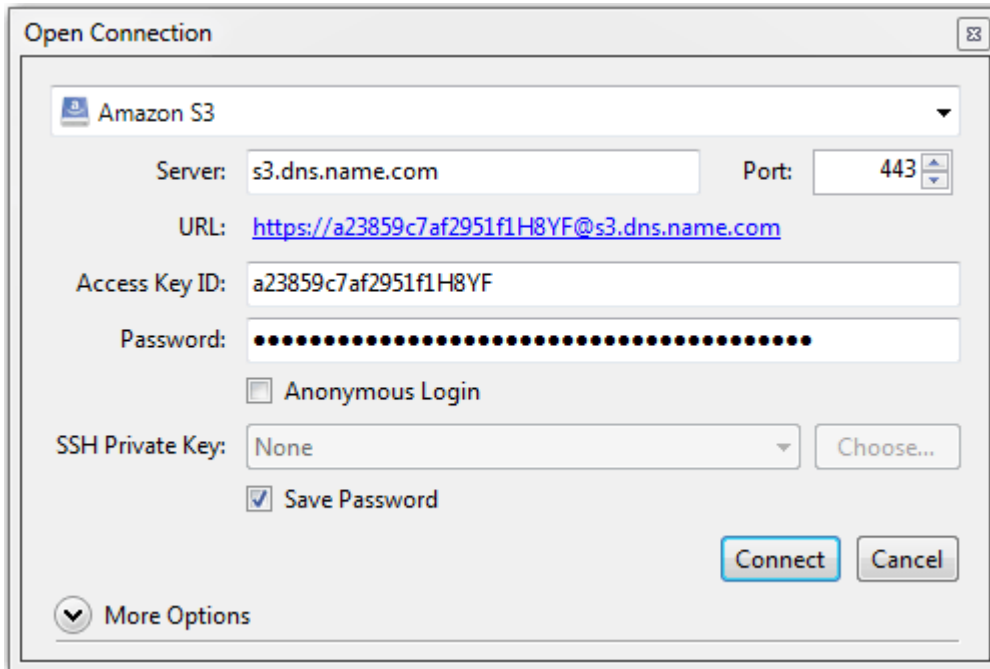
Use them only for a good reason and when you know what you are doing.

4.3.4.1 Managing Buckets with CyberDuck

4.3.4.1.1 Creating Buckets

To create a new S3 bucket with CyberDuck, do the following:

1. Click **Open Connection**.
2. Specify the following parameters:
 - The external DNS name for the S3 endpoint that you specified when creating the S3 cluster.
 - The Access Key ID and the Secret Access Key of an object storage user (see *Creating S3 Users* on page 82).

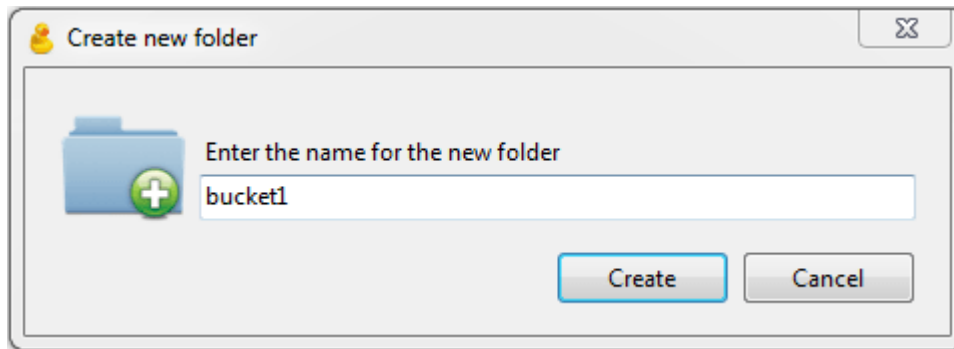


The screenshot shows the 'Open Connection' dialog box in CyberDuck. The dialog is titled 'Open Connection' and has a close button in the top right corner. It contains the following fields and options:

- A dropdown menu showing 'Amazon S3'.
- 'Server': s3.dns.name.com
- 'Port': 443
- 'URL': https://a23859c7af2951f1H8YF@s3.dns.name.com
- 'Access Key ID': a23859c7af2951f1H8YF
- 'Password': masked with dots
- Anonymous Login
- 'SSH Private Key': None, with a 'Choose...' button
- Save Password
- 'Connect' and 'Cancel' buttons
- 'More Options' dropdown at the bottom left

By default, the connection is established over HTTPS. To use CyberDuck over HTTP, you must install a special S3 profile.

3. Once the connection is established, click **File > New Folder**.



4. Specify a name for the new bucket, and then click **Create**.

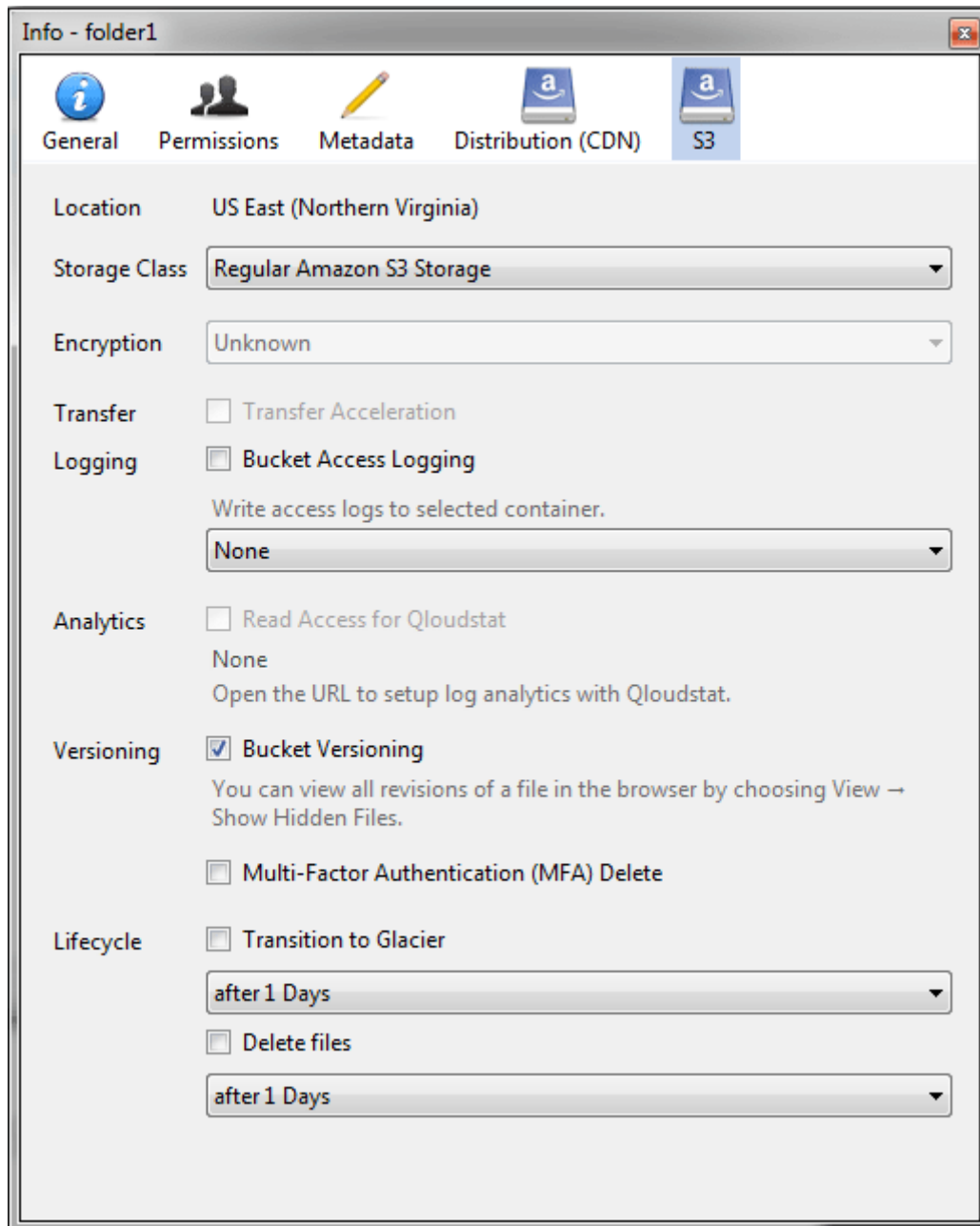
Note: It is recommended to use bucket names that comply with DNS naming conventions. For more information on bucket naming, see *Bucket and Key Naming Policies* on page 89.

The new bucket will appear in CyberDuck and you can manage it and upload files into it.

4.3.4.1.2 Managing Bucket Versions

Versioning is a way of keeping multiple variants of an object in the same bucket. You can use versioning to preserve, retrieve, and restore every version of every object stored in your S3 bucket. With versioning, you can easily recover from both unintended user actions and application failures. For more information about bucket versioning, refer to the [Amazon documentation](#).

Bucket versioning is turned by default. You can turn it on from a third-party S3 browser by selecting a checkbox in the bucket property. For example:



4.3.4.1.3 Listing Bucket Contents

You can list bucket contents with a web browser. To do this, visit the URL that consists of the external DNS name for the S3 endpoint that you specified when creating the S3 cluster and the bucket name. For example, `mys3storage.example.com/mybucket`.

Note: You can also copy the link to bucket contents by right-clicking it in CyberDuck, and then selecting Copy

URL.

4.3.4.2 Managing Buckets from Command Line

4.3.4.2.1 Listing Object Storage Buckets

You can list all buckets in the S3 object storage with the `ostor-s3-admin list-all-buckets` command. For each bucket, the command shows owner, creation data, versioning status, and total size (the size of all objects stored in the bucket plus the size of all unfinished multipart uploads for this bucket). For example:

```
# ostor-s3-admin list-all-buckets -V 0100000000000002
Total 3 buckets
BUCKET          OWNER          CREATION_DATE  VERSIONING      TOTAL_SIZE, BYTES
bucket1         968d1a79968d1a79  2015-08-18T09:32:35.000Z  none           1024
bucket2         968d1a79968d1a79  2015-08-18T09:18:20.000Z  enabled        0
bucket3         968d1a79968d1a79  2015-08-18T09:22:15.000Z  suspended     1024000
```

To output the list in XML, use the `-x` option. For example:

```
# ostor-s3-admin list-all-buckets -X
<?xml version="1.0" encoding="UTF-8"?><ListBucketsResult><Buckets><Bucket><Name>bucker2</Name><Owner>d7c53fc1f931661f</Owner><CreationDate>2017-04-03T17:11:44.000Z</CreationDate><Versioning>none</Versioning><Notary>off</Notary><TotalSize>0</TotalSize></Bucket><Bucket><Name>bucket1</Name><Owner>d7c53fc1f931661f</Owner><CreationDate>2017-04-03T17:11:33.000Z</CreationDate><Versioning>none</Versioning><Notary>off</Notary><TotalSize>0</TotalSize></Bucket></Buckets></ListBucketsResult>
```

To filter buckets by user who owns them, use the `-i` option. For example:

```
# ostor-s3-admin list-all-buckets -i d7c53fc1f931661f
BUCKET  OWNER          CREATION_DATE  VERSIONING  TOTAL_SIZE  NOTARY  NOTARY_PROVIDER
bucker2 d7c53fc1f931661f  2017-04-03T17:11:44.000Z  none       0          off     0
```

4.3.4.2.2 Querying Object Storage Bucket Information

You can query bucket metadata information and ACL with the `ostor-s3-admin query-bucket-info` command. For example, for bucket1:

```
# ostor-s3-admin query-bucket-info -b bucket1 -V 0100000000000002
BUCKET  OWNER          CREATION_DATE  VERSIONING  TOTAL_SIZE
bucket1 d339edcf885eeafc  2017-12-21T12:42:46.000Z  none       0

ACL: d339edcf885eeafc: FULL_CONTROL
```


4.3.4.2.3 Changing Object Storage Bucket Owners

You can pass ownership of a bucket to the specified user with the `ostor-s3-admin change-bucket-owner` command. For example, to make user with ID `bf0b3b15eb7c9019` the owner of `bucket1`:

```
# ostor-s3-admin change-bucket-owner -b bucket1 -i bf0b3b15eb7c9019 -V 0100000000000002
Changed owner of the bucket bucket1. New owner bf0b3b15eb7c9019
```

4.3.4.2.4 Deleting Object Storage Buckets

You can delete the specified bucket with the `ostor-s3-admin delete-bucket` command. Deleting a bucket will delete all objects in it (including their old versions) as well as all unfinished multipart uploads for this bucket. For example:

```
# ostor-s3-admin delete-bucket -b bucket1 -V 0100000000000002
```

4.3.5 Best Practices for Using Object Storage

This chapter describes recommendations on using various features of Object Storage. These recommendations are called to help you enable additional functionality or improve convenience or performance of Object Storage.

4.3.5.1 Bucket and Key Naming Policies

It is recommended to use bucket names that comply with DNS naming conventions:

- 3 to 63 characters long.
- Start and end with a lowercase letter or number.
- Contain only lowercase letters, numbers, periods (`.`), hyphens (`-`), and underscores (`_`).
- Can be a series of valid name parts (described previously) separated by periods.

An object key can be a string of any UTF-8 encoded characters up to 1024 bytes long.

4.3.5.2 Improving Performance of PUT Operations

Object storage supports uploading of objects as large as 5 GB in size with a single PUT request. Upload performance can be improved, however, by splitting large objects into pieces and uploading them concurrently with multipart upload API. This approach will divide the load between multiple OS services.

It is recommended to use multipart uploads for objects larger than 5 MB.

4.3.6 Appendices

This section provides reference information related to Object Storage.

4.3.6.1 Appendix A: Supported Amazon S3 REST Operations

The following Amazon S3 REST operations are currently supported by the Storage implementation of the Amazon S3 protocol:

Service operations:

- GET Service

Bucket operations:

- DELETE Bucket
- GET Bucket (List Objects)
- GET Bucket acl
- GET Bucket location
- GET Bucket Object versions
- GET Bucket versioning
- HEAD Bucket
- List Multipart Uploads
- PUT Bucket
- PUT Bucket acl
- PUT Bucket versioning

Object operations:

- DELETE Object
- DELETE Multiple Objects
- GET Object
- GET Object ACL
- HEAD Object
- POST Object
- PUT Object
- PUT Object - Copy
- PUT Object acl
- Initiate Multipart Upload
- Upload Part
- Complete Multipart Upload
- Abort Multipart Upload
- List Parts
- Upload Part Copy

Note: For a complete list of Amazon S3 REST operations, see [Amazon S3 REST API documentation](#).

4.3.6.2 Appendix B: Supported Amazon Request Headers

The following Amazon S3 REST request headers are currently supported by the Storage implementation of the Amazon S3 protocol:

- x-amz-acl
- x-amz-delete-marker
- x-amz-grant-full-control

- x-amz-grant-read-acp
- x-amz-grant-read
- x-amz-grant-write
- x-amz-grant-write-acp
- x-amz-meta-**
- x-amz-version-id
- x-amz-copy-source
- x-amz-metadata-directive
- x-amz-copy-source-version-id

4.3.6.3 Appendix C: Supported Authentication Schemes

The following authentication scheme is supported by the Storage implementation of the Amazon S3 protocol:

- [Signature Version 2.](#)
- [Signature Version 4.](#)

4.3.6.4 Appendix D: Amazon S3 Features Supported by Bucket Policies

The Storage implementation of the Amazon S3 bucket policies supports the following S3 actions, condition keys, and condition comparators:

Object actions:

- s3:AbortMultipartUpload
- s3:DeleteObject
- s3:DeleteObjectTagging
- s3:DeleteObjectVersion
- s3:DeleteObjectVersionTagging
- s3:GetObject
- s3:GetObject

- s3:GetObjectAcl
- s3:GetObjectTagging
- s3:GetObjectTorrent
- s3:GetObjectVersion
- s3:GetObjectVersionAcl
- s3:GetObjectVersionTagging
- s3:ListMultipartUploadParts
- s3:PutObject
- s3:PutObjectAcl
- s3:PutObjectTagging
- s3:PutObjectVersionAcl
- s3:PutObjectVersionTagging
- s3:RestoreObject

Bucket actions:

- s3:CreateBucket
- s3>DeleteBucket
- s3:ListBucket
- s3:ListBucketMultipartUploads
- s3:ListBucketVersions

Bucket subresource actions:

- s3>DeleteBucketPolicy
- s3>DeleteBucketWebsite
- s3:GetBucketAcl
- s3:GetBucketCORS
- s3:GetBucketLocation

- s3:GetBucketLogging
- s3:GetBucketNotification
- s3:GetBucketPolicy
- s3:GetBucketRequestPayment
- s3:GetBucketTagging
- s3:GetBucketVersioning
- s3:GetBucketWebsite
- s3:GetLifecycleConfiguration
- s3:GetReplicationConfiguration
- s3:PutBucketAcl
- s3:PutBucketCORS
- s3:PutBucketLogging
- s3:PutBucketNotification
- s3:PutBucketPolicy
- s3:PutBucketRequestPayment
- s3:PutBucketTagging
- s3:PutBucketVersioning
- s3:PutBucketWebsite
- s3:PutLifecycleConfiguration
- s3:PutReplicationConfiguration

Condition keys:

- s3:x-amz-storage-class
- s3:x-amz-acl
- s3:x-amz-grant-full-control
- s3:x-amz-grant-read

- s3:x-amz-grant-read-acp
- s3:x-amz-grant-write
- s3:x-amz-grant-write-acp
- s3:x-amz-copy-source
- s3:TlsVersion
- s3:x-amz-content-sha256
- s3:signatureversion
- s3:signatureAge
- s3:authType
- s3:x-amz-website-redirect-location
- s3:object-lock-mode
- s3:object-lock-retain-until-date
- s3:object-lock-legal-hold
- s3:object-lock-remaining-retention-days
- s3:prefix
- s3:versionid
- s3:max-keys
- s3:locationconstraint
- aws:SourceIp

Condition comparators:

- StringNotEquals
- StringEqualsIgnoreCase
- StringNotEqualsIgnoreCase
- StringLike
- StringNotLike

- NumericEquals
- NumericNotEquals
- NumericLessThan
- NumericLessThanEquals
- NumericGreaterThan
- NumericGreaterThanEquals
- DateEquals
- DateNotEquals
- DateLessThan
- DateLessThanEquals
- DateGreaterThan
- DateGreaterThanEquals
- BinaryEquals
- IpAddress
- NotIpAddress

CHAPTER 5

Monitoring Storage Clusters

Monitoring a Storage cluster is very important because it allows you to check the status and health of all computers in the cluster and react as necessary. This chapter explains how to monitor your Storage cluster.

5.1 Monitoring General Cluster Parameters

By monitoring general parameters, you can get detailed information about all components of a Storage cluster, its overall status and health. To display this information, use the `vstorage -c <cluster_name> top` command, for example:

```

Cluster 'stor1': healthy
Space: [OK] allocatable 238GB of 250GB, free 250GB of 250GB
MDS nodes: 1 of 1, epoch uptime: 33 min
CS nodes: 2 of 2 (2 avail, 0 inactive, 0 offline)
License: ACTIVE (expiration: 03/18/2014, capacity: 6399TB, used: 762B)
Replication: 1 norm, 1 limit
IO:      read      0B/s ( 0ops/s), write      0B/s ( 0ops/s)

```

MDSID	STATUS	%CTIME	COMMITTS	%CPU	MEM	UPTIME	HOST
M 1	avail	2.0%	0/s	0.0%	10m	33 min	dhcp-10-30-24-73.sw.ru:25

CSID	STATUS	SPACE	AVAIL	REPLICAS	UNIQUE	IOWAIT	IOLAT(ms)	QDEPTH	HOST
1025	active	125GB	119GB	0	0	0%	0/0	0.0	dhcp-10
1026	active	125GB	119GB	0	0	0%	0/0	0.0	dhcp-10

CLID	LEASES	READ	WRITE	RD_OPS	WR_OPS	FSYNCS	IOLAT(ms)	HOST
2053	0/1	0B/s	0B/s	0ops/s	0ops/s	0ops/s	0/0	dhcp
2051	0/0	0B/s	0B/s	0ops/s	0ops/s	0ops/s	0/0	dhcp

TIME	SYS SEV	MESSAGE
21-02-14 16:55:20	MDS INF	The cluster physical free space: 250.6Gb (99%), total
21-02-14 17:26:59	MDS INF	Global configuration updated by request from 10.30.24
21-02-14 17:26:59	JRN INF	gen.license_status=60
21-02-14 17:26:59	MDS INF	License PCSS.02706224.0000 is ACTIVE

The command above shows detailed information about the stor1 cluster. The general parameters (highlighted in red) are explained in the table below.

Parameter	Description
Cluster	<p>Overall status of the cluster:</p> <ul style="list-style-type: none"> • healthy. All chunk servers in the cluster are active. • unknown. There is not enough information about the cluster state (e.g., because the master MDS server was elected a while ago). • degraded. Some of the chunk servers in the cluster are inactive. • failure. The cluster has too many inactive chunk servers; the automatic replication is disabled. • SMART warning. One or more physical disks attached to cluster nodes are in pre-failure condition. For details, see <i>Monitoring Physical Disks</i> on page 109.

Continued on next page

Table 5.1.1 -- continued from previous page

Parameter	Description
Space	<p>Amount of disk space in the cluster:</p> <ul style="list-style-type: none"> • free. Free physical disk space in the cluster. • allocatable. Amount of logical disk space available to clients. Allocatable disk space is calculated on the basis of the current replication parameters and free disk space on chunk servers. It may also be limited by license. <hr/> <p>Note: For more information on monitoring and understanding disk space usage in clusters, see <i>Understanding Disk Space Usage</i> on page 103.</p> <hr/>
MDS nodes	Number of active MDS servers as compared to the total number of MDS servers configured for the cluster.
epoch time	Time elapsed since the MDS master server election.
CS nodes	<p>Number of active chunk servers as compared to the total number of chunk servers configured for the cluster.</p> <p>The information in parentheses informs you of the number of</p> <ul style="list-style-type: none"> • Active chunk servers (avail.) that are currently up and running in the cluster. • Inactive chunk servers (inactive) that are temporarily unavailable. A chunk server is marked as inactive during its first 5 minutes of inactivity. • Offline chunk servers (offline) that have been inactive for more than 5 minutes. A chunk server changes its state to offline after 5 minutes of inactivity. Once the state is changed to offline, the cluster starts replicating data to restore the chunks that were stored on the offline chunk server.
License	Key number under which the license is registered on the Key Authentication server and license state.
Replication	Replication settings. The normal number of chunk replicas and the limit after which a chunk gets blocked until recovered.
IO	<p>Disks IO activity in the cluster:</p> <ul style="list-style-type: none"> • Speed of read and write I/O operations, in bytes per second. • Number of read and write I/O operations per second.

5.2 Monitoring Metadata Servers

MDS servers are a critical component of any Storage cluster, and monitoring the health and state of MDS servers is a very critical task. To monitor MDS servers, use the `vstorage -c <cluster_name> top` command, for example:

```
Cluster 'stor1': healthy
Space: [OK] allocatable 238GB of 250GB, free 250GB of 250GB
MDS nodes: 1 of 1, epoch uptime: 33 min
CS nodes: 2 of 2 (2 avail, 0 inactive, 0 offline)
License: ACTIVE (expiration: 03/18/2014, capacity: 6399TB, used: 762B)
Replication: 1 norm, 1 limit
IO:      read      0B/s ( 0ops/s), write      0B/s ( 0ops/s)
```

MDSID	STATUS	%CTIME	COMMITTS	%CPU	MEM	UPTIME	HOST
M 1	avail	2.0%	0/s	0.0%	10m	33 min	dhcp-10-30-24-73.sw.ru:25

CSID	STATUS	SPACE	AVAIL	REPLICAS	UNIQUE	IOWAIT	IOLAT(ms)	QDEPTH	HOST
1025	active	125GB	119GB	0	0	0%	0/0	0.0	dhcp-10
1026	active	125GB	119GB	0	0	0%	0/0	0.0	dhcp-10

CLID	LEASES	READ	WRITE	RD_OPS	WR_OPS	FSYNCS	IOLAT(ms)	HOST
2053	0/1	0B/s	0B/s	0ops/s	0ops/s	0ops/s	0/0	dhcp
2051	0/0	0B/s	0B/s	0ops/s	0ops/s	0ops/s	0/0	dhcp

TIME	SYS SEV	MESSAGE
21-02-14 16:55:20	MDS INF	The cluster physical free space: 250.6Gb (99%), total
21-02-14 17:26:59	MDS INF	Global configuration updated by request from 10.30.24
21-02-14 17:26:59	JRN INF	gen.license_status=6U
21-02-14 17:26:59	MDS INF	License PCSS.02706224.0000 is ACTIVE

The command above shows detailed information about the `stor1` cluster. The monitoring parameters for MDS servers (highlighted in red) are explained in the table below:

Parameter	Description
MDSID	MDS server identifier (ID). The letter "M" before ID, if present, means that the given server is the master MDS server.
STATUS	MDS server status.
%CTIME	Total time the MDS server spent writing to the local journal.
COMMITTS	Local journal commit rate.
%CPU	MDS server activity time.

Continued on next page

Table 5.2.1 -- continued from previous page

Parameter	Description
MEM	Amount of physical memory the MDS server uses.
UPTIME	Time elapsed since the last MDS server start.
HOST	MDS server hostname or IP address.

5.3 Monitoring Chunk Servers

By monitoring chunk servers, you can keep track of the disk space available in a Storage cluster. To monitor chunk servers, use the `vstorage -c <cluster_name> top` command, for example:

```
Cluster 'stor1': healthy
Space: [OK] allocatable 238GB of 250GB, free 250GB of 250GB
MDS nodes: 1 of 1, epoch uptime: 33 min
CS nodes: 2 of 2 (2 avail, 0 inactive, 0 offline)
License: ACTIVE (expiration: 03/18/2014, capacity: 6399TB, used: 762B)
Replication: 1 norm, 1 limit
IO:      read      0B/s ( 0ops/s), write      0B/s ( 0ops/s)

MDSID STATUS  %CTIME  COMMITS  %CPU  MEM  UPTIME HOST
M   1 avail    2.0%    0/s    0.0%  10m  33 min dhcp-10-30-24-73.sw.ru:25

CSID STATUS  SPACE  AVAIL  REPLICAS  UNIQUE  IOWAIT  IOLAT(ms)  QDEPTH  HOST
1025 active  125GB  119GB    0         0       0%       0/0       0.0  dhcp-10
1026 active  125GB  119GB    0         0       0%       0/0       0.0  dhcp-10

CLID  LEASES  READ  WRITE  RD_OPS  WR_OPS  FSYNCS  IOLAT(ms)  HOST
2053   0/1    0B/s  0B/s   0ops/s  0ops/s  0ops/s  0/0  dhcp
2051   0/0    0B/s  0B/s   0ops/s  0ops/s  0ops/s  0/0  dhcp

TIME          SYS SEV MESSAGE
21-02-14 16:55:20 MDS INF The cluster physical free space: 250.6Gb (99%), total
21-02-14 17:26:59 MDS INF Global configuration updated by request from 10.30.24
21-02-14 17:26:59 JRN INF gen.license_status=6U
21-02-14 17:26:59 MDS INF License PCSS.02706224.0000 is ACTIVE
```

The command above shows detailed information about the `stor1` cluster. The monitoring parameters for chunk servers (highlighted in red) are explained in the table below:

Parameter	Description
CSID	Chunk server identifier (ID).

Continued on next page

Table 5.3.1 -- continued from previous page

Parameter	Description
STATUS	<p>Chunk server status:</p> <ul style="list-style-type: none"> • active. The chunk server is up and running. • inactive. The chunk server is temporarily unavailable. A chunk server is marked as inactive during its first 5 minutes of inactivity. • offline. The chunk server is inactive for more than 5 minutes. After the chunk server goes offline, the cluster starts replicating data to restore the chunks that were stored on the affected chunk server. • dropped. The chunk server was removed by the administrator.
SPACE	Total amount of disk space on the chunk server.
FREE	Free disk space on the chunk server.
REPLICAS	Number of replicas stored on the chunk server.
IOWAIT	Percentage of time spent waiting for I/O operations being served.
IOLAT	Average/maximum time, in milliseconds, the client needed to complete a single IO operation during the last 20 seconds.
QDEPTH	Average chunk server I/O queue depth.
HOST	Chunk server hostname or IP address.
FLAGS	<p>The following flags may be shown for active chunk servers:</p> <ul style="list-style-type: none"> • J: The CS uses a write journal. • C: Checksumming is enabled for the CS. Checksumming lets you know when a third party changes the data on the disk. • D: Direct I/O, the normal state for a CS without a write journal. • c: The chunk server's write journal is clean, there is nothing to commit from the write journaling SSD to the HDD where the CS is located. <hr/> <p>Note: Flags which may be shown for failed chunk servers are described in <i>Failed Chunk Servers</i> on page 143.</p> <hr/>

5.3.1 Understanding Disk Space Usage

Usually, you get the information on how disk space is used in your cluster with the `vstorage top` command. This command displays the following disk-related information: total space, free space, and allocatable space. For example:

```
# vstorage -c stor1 top
connected to MDS#1
Cluster 'stor1': healthy
Space: [OK] allocatable 180GB of 200GB, free 1.6TB of 1.7TB
<...>
```

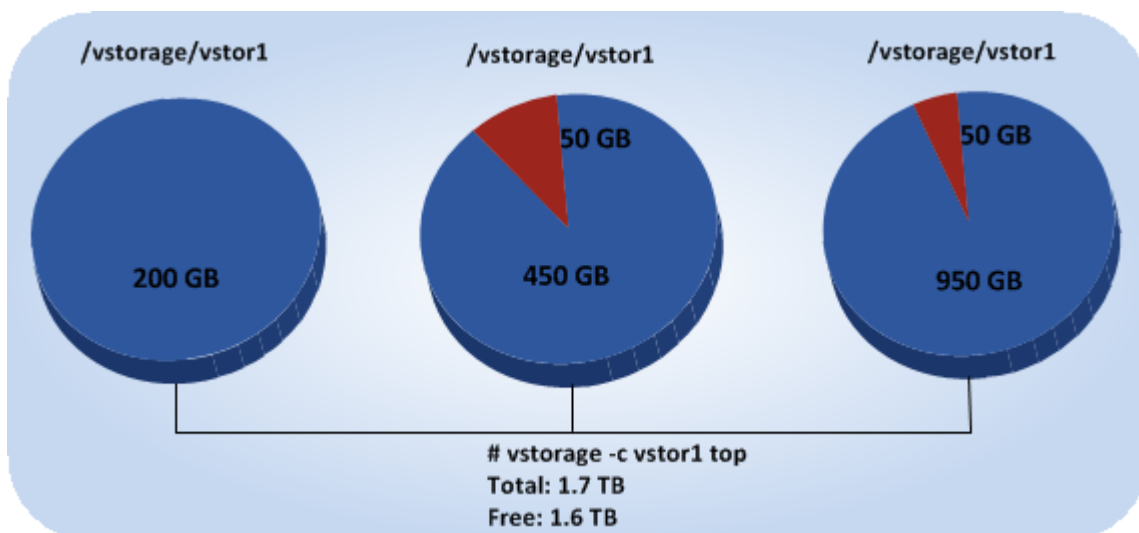
In this command output:

- 1.7TB is the total disk space in the `stor1` cluster. The total disk space is calculated on the basis of used and free disk space on all partitions in the cluster. Used disk space includes the space occupied by all data chunks and their replicas plus the space occupied by any other files stored on the cluster partitions.

Let us assume that you have a 100 GB partition and 20 GB on this partition are occupied by some files. Now if you set up a chunk server on this partition, this will add 100 GB to the total disk space of the cluster, though only 80 GB of this disk space will be free and available for storing data chunks.

- 1.6TB is the free disk space in the `stor1` cluster. Free disk space is calculated by subtracting the disk space occupied by data chunks and any other files on the cluster partitions from the total disk space.

For example, if the amount of free disk space is 1.6 TB and the total disk space is 1.7 TB, this means that about 100 GB on the cluster partitions are already occupied by some files.



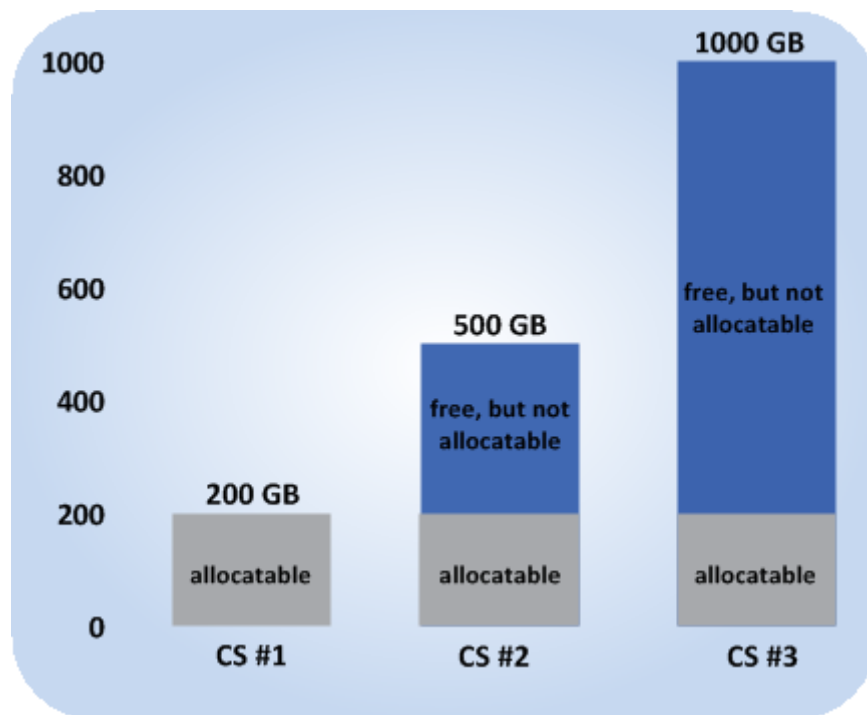
- `allocatable 180GB of 200GB` is the amount of free disk space that can be used for storing data chunks. See **Understanding allocatable disk space** below for details.

5.3.1.1 Understanding Allocatable Disk Space

When monitoring disk space information in the cluster, you also need to pay attention to the space reported by the `vstorage top` utility as *allocatable*. Allocatable space is the amount of disk space that is free and can be used for storing user data. Once this space runs out, no data can be written to the cluster.

To better understand how allocatable disk space is calculated, let us consider the following example:

- The cluster has 3 chunk servers. The first chunk server has 200 GB of disk space, the second one — 500 GB, and the third one — 1 TB.
- The default replication factor of 3 is used in the cluster, meaning that each data chunk must have 3 replicas stored on three different chunk servers.



In this example, the available disk space will equal 200 GB, that is, set to the amount of disk space on the smallest chunk server:

```
# vstorage -c stor1 top
connected to MDS#1
Cluster 'stor1': healthy
Space: [OK] allocatable 180GB of 200GB, free 1.6TB of 1.7TB
```



```
<...>
```

This is explained by the fact that in this cluster configuration each server is set to store one replica for each data chunk. So once the disk space on the smallest chunk server (200 GB) runs out, no more chunks in the cluster can be created until a new chunk server is added or the replication factor is decreased.

If you now change the replication factor to 2, the `vstorage top` command will report the available disk space as 700 GB:

```
# vstorage set-attr -R /vstorage/stor1 replicas=2:1
# vstorage -c stor1 top
connected to MDS#1
Cluster 'stor1': healthy
Space: [OK] allocatable 680GB of 700GB, free 1.6TB of 1.7TB
<...>
```

The available disk space has increased because now only 2 replicas are created for each data chunk and new chunks can be made even if the smallest chunk server runs out of space (in this case, replicas will be stored on a bigger chunk server).

Note: Allocatable disk space may also be limited by license.

5.3.1.2 Viewing Space Occupied by Data Chunks

To view the total amount of disk space occupied by all user data in the cluster, run the `vstorage top` command and press the `V` key on your keyboard. Once you do this, your command output should look like the following:

```
# vstorage -c stor1 top
Cluster 'stor1': healthy
Space: [OK] allocatable 180GB of 200GB, free 1.6TB of 1.7TB
MDS nodes: 1 of 1, epoch uptime: 2d 4h
CS nodes: 3 of 3 (3 avail, 0 inactive, 0 offline)
Replication: 2 norm, 1 limit, 4 max
Chunks: [OK] 38 (100%) healthy, 0 (0%) degraded, 0 (0%) urgent,
         0 (0%) blocked, 0 (0%) offline, 0 (0%) replicating,
         0 (0%) overcommitted, 0 (0%) deleting, 0 (0%) void
FS: 1GB in 51 files, 51 inodes, 23 file maps, 38 chunks, 76 chunk replicas
<...>
```

Note: The **FS** field shows the size of all user data in the cluster without consideration for replicas.

5.3.2 Exploring Chunk States

The table below lists all possible states a chunk can have.

Status	Description
healthy	Percentage of chunks that have enough active replicas. The normal state of chunks.
replicating	Percentage of chunks which are being replicated. Write operations on such chunks are frozen until replication ends.
offline	Percentage of chunks all replicas of which are offline. Such chunks are completely inaccessible for the cluster and cannot be replicated, read from or written to. All requests to an offline chunk are frozen until a CS that stores that chunk's replica goes online. Get offline chunk servers back online as fast as possible to avoid losing data.
void	Percentage of chunks that have been allocated but never used yet. Such chunks contain no data. It is normal to have some void chunks in the cluster.
pending	Percentage of chunks that must be replicated immediately. For a write request from client to a chunk to complete, the chunk must have at least the set minimum amount of replicas. If it does not, the chunk is blocked and the write request cannot be completed. As blocked chunks must be replicated as soon as possible, the cluster places them in a special high-priority replication queue and reports them as pending.
blocked	Percentage of chunks which have fewer active replicas than the set minimum amount. Write requests to a blocked chunk are frozen until it has at least the set minimum amount of replicas. Read requests to blocked chunks are allowed, however, as they still have some active replicas left. Blocked chunks have higher replication priority than degraded chunks. Having blocked chunks in the cluster increases the risk of losing data, so postpone any maintenance on working cluster nodes and get offline chunk servers back online as fast as possible.
degraded	Percentage of chunks with the number of active replicas lower than normal but equal to or higher than the set minimum. Such chunks can be read from and written to. However, in the latter case a degraded chunk becomes urgent.

Continued on next page

Table 5.3.2.1 -- continued from previous page

Status	Description
urgent	Percentage of chunks which are degraded and have non-identical replicas. Replicas of a degraded chunk may become non-identical if some of them are not accessible during a write operation. As a result, some replicas happen to have the new data while some still have the old data. The latter are dropped by the cluster as fast as possible. Urgent chunks do not affect information integrity as the actual data is stored in at least the set minimum amount of replicas.
standby	Percentage of chunks that have one or more replicas in the standby state. A replica is marked standby if it has been inactive for no more than 5 minutes.
overcommitted	Percentage of chunks that have more replicas than normal. Usually these chunks appear after the normal number of replicas has been lowered or a lot of data has been deleted. Extra replicas are eventually dropped, however, this process may slow down during replication.

5.4 Monitoring Clients

By monitoring clients, you can check the status and health of servers that you use to access virtual machines and containers. To monitor clients, use the `vstorage -c <cluster_name> top` command, for example:

```

Cluster 'stor1': healthy
Space: [OK] allocatable 238GB of 250GB, free 250GB of 250GB
MDS nodes: 1 of 1, epoch uptime: 33 min
CS nodes: 2 of 2 (2 avail, 0 inactive, 0 offline)
License: ACTIVE (expiration: 03/18/2014, capacity: 6399TB, used: 762B)
Replication: 1 norm, 1 limit
IO:      read      0B/s ( 0ops/s), write      0B/s ( 0ops/s)

```

MDSID	STATUS	%CTIME	COMMITTS	%CPU	MEM	UPTIME	HOST
M 1	avail	2.0%	0/s	0.0%	10m	33 min	dhcp-10-30-24-73.sw.ru:25

CSID	STATUS	SPACE	AVAIL	REPLICAS	UNIQUE	IOWAIT	IOLAT(ms)	QDEPTH	HOST
1025	active	125GB	119GB	0	0	0%	0/0	0.0	dhcp-10
1026	active	125GB	119GB	0	0	0%	0/0	0.0	dhcp-10

CLID	LEASES	READ	WRITE	RD_OPS	WR_OPS	FSYNCS	IOLAT(ms)	HOST
2053	0/1	0B/s	0B/s	0ops/s	0ops/s	0ops/s	0/0	dhcp
2051	0/0	0B/s	0B/s	0ops/s	0ops/s	0ops/s	0/0	dhcp

TIME	SYS SEV MESSAGE
21-02-14 16:55:20	MDS INF The cluster physical free space: 250.6Gb (99%), total
21-02-14 17:26:59	MDS INF Global configuration updated by request from 10.30.24
21-02-14 17:26:59	JRN INF gen.license_status=6U
21-02-14 17:26:59	MDS INF License PCSS.02706224.0000 is ACTIVE

The command above shows detailed information about the stor1 cluster. The monitoring parameters for clients (highlighted in red) are explained in the table below:

Parameter	Description
CLID	Client identifier (ID).
LEASES	Average number of files opened for reading/writing by the client and not yet closed, for the last 20 seconds.
READ	Average rate, in bytes per second, at which the client reads data, for the last 20 seconds.
WRITE	Average rate, in bytes per second, at which the client writes data, for the last 20 seconds.
RD_OPS	Average number of read operations per second the client made, for the last 20 seconds.
WR_OPS	Average number of write operations per second the client made, for the last 20 seconds.
FSYNCS	Average number of sync operations per second the client made, for the last 20 seconds.

Continued on next page

Table 5.4.1 -- continued from previous page

Parameter	Description
IOLAT	Average/maximum time, in milliseconds, the client needed to complete a single IO operation, for the last 20 seconds.
HOST	Client hostname or IP address.

5.5 Monitoring Physical Disks

The S.M.A.R.T. status of physical disks is monitored by the `smartctl` tool installed along with `Virtuozzo`. The tool is run every 5 minutes. The `smartctl` tool polls all physical disks attached to Hardware Nodes in the cluster, including caching and journaling SSDs, and reports the results to the MDS server.

Note: For the tool to work, enable the S.M.A.R.T. functionality in Node's BIOS.

You can view disk poll results for the last 10 minutes in the output of the `vstorage top` command. For example:

```
Cluster 'stor1': healthy, SMART warning
Space: [OK] allocatable 100GB (+778GB unlicensed) of 926GB, free 924GB of 926GB
MDS nodes: 1 of 1, epoch uptime: 7d 22h
CS nodes: 2 of 2 (2 avail, 0 inactive, 0 offline)
Replication: 1 norm, 1 limit
IO:      read      0B/s ( 0ops/s), write      0B/s ( 0ops/s)

MDSID STATUS  %CTIME  COMMITS  %CPU  MEM  UPTIME HOST
M  1 avail    0.0%    0/s    0.0%  48m  7d 22h pcs36.qa.sw.ru:2510

CSID STATUS  SPACE  AVAIL  REPLICAS  UNIQUE  IOWAIT  IOLAT(ms)  QDEPTH  HOST
1025 active  9.1GB  7.1GB      0         0      0%      0/0      0.0  pcs36.q
1026 active  916GB  870GB      0         0      0%      0/0      0.0  pcs36.q

CLID  LEASES  READ  WRITE  RD_OPS  WR_OPS  FSYNCS  IOLAT(ms)  HOST

TIME  SYS SEU MESSAGE
01-07-14 16:42:19  MON WRN CS#1026 was stopped
01-07-14 16:42:26  JRN INF MDS#1 at 10.29.2.16:2510 became master
01-07-14 16:42:26  MDS WRN License not installed, please add license using comma
01-07-14 16:42:29  MON WRN MDS#1 was stopped
01-07-14 16:42:44  MDS INF CS#1025, CS#1026 are active
01-07-14 16:42:53  MDS INF The cluster is healthy with 2 active CS
01-07-14 16:42:53  MDS INF The cluster physical free space: 925.0Gb (99%), total
```

If the **SMART warning** message is shown in the main table, one of the physical disks is in pre-failure condition according to S.M.A.R.T. Press **d** to switch to the disks table to see more details. For example:

```
Cluster 'stor1': healthy, SMART warning
Space: [OK] allocatable 100GB (+778GB unlicensed) of 926GB, free 924GB of 926GB
MDS nodes: 1 of 1, epoch uptime: 7d 22h
CS nodes: 2 of 2 (2 avail, 0 inactive, 0 offline)
Replication: 1 norm, 1 limit
IO:      read      0B/s ( 0ops/s), write      0B/s ( 0ops/s)
```

DISK	SMART	TEMP	CAPACITY	SERIAL	MODEL	HOST
sdc	OK	27C	931GB	1374X80PS	TOSHIBA DT01ACA100	pcs36.qa
sde	Warn	31C	931GB	MSE5235U36ZHWJ	Hitachi HDS721010DLE630	pcs36.qa

The disks table shows the following parameters:

Parameter	Description
DISK	Disk name assigned by operating system.
SMART	Disk's S.M.A.R.T. status: <ul style="list-style-type: none"> • OK: The disk is healthy. • Warn: The disk is in pre-failure condition. Pre-failure condition means that at least one of these S.M.A.R.T. counters is nonzero: <ul style="list-style-type: none"> • Reallocated Sector Count • Reallocated Event Count • Current Pending Sector Count • Offline Uncorrectable
TEMP	Disk temperature in Celsius.
CAPACITY	Disk capacity.
SERIAL	Disk serial number.
MODEL	Disk model.
HOST	Disk's host address.

To disable S.M.A.R.T. disk monitoring, stop and disable the corresponding systemd units: `vstorage-send-diskinfo.service` and `vstorage-send-diskinfo.timer`.

5.6 Monitoring Event Logs

You can use the `vstorage -c <cluster_name> top` utility to monitor significant events happening in a Storage cluster, for example:

```
Cluster 'stor1': healthy
Space: [OK] allocatable 238GB of 250GB, free 250GB of 250GB
MDS nodes: 1 of 1, epoch uptime: 33 min
CS nodes: 2 of 2 (2 avail, 0 inactive, 0 offline)
License: ACTIVE (expiration: 03/18/2014, capacity: 6399TB, used: 762B)
Replication: 1 norm, 1 limit
IO:      read      0B/s ( 0ops/s), write      0B/s ( 0ops/s)

MDSID STATUS  %CTIME  COMMITS  %CPU  MEM  UPTIME HOST
M   1 avail    2.0%    0/s    0.0%  10m  33 min dhcp-10-30-24-73.sw.ru:25

CSID STATUS  SPACE  AVAIL  REPLICAS  UNIQUE  IOWAIT  IOLAT(ms)  QDEPTH  HOST
1025 active  125GB  119GB      0         0      0%      0/0      0.0  dhcp-10
1026 active  125GB  119GB      0         0      0%      0/0      0.0  dhcp-10

CLID  LEASES  READ  WRITE  RD_OPS  WR_OPS  FSYNCS  IOLAT(ms)  HOST
2053   0/1   0B/s  0B/s   0ops/s  0ops/s  0ops/s  0/0  dhcp
2051   0/0   0B/s  0B/s   0ops/s  0ops/s  0ops/s  0/0  dhcp

TIME          SYS SEV MESSAGE
21-02-14 16:55:20 MDS INF The cluster physical free space: 250.6Gb (99%), total
21-02-14 17:26:59 MDS INF Global configuration updated by request from 10.30.24
21-02-14 17:26:59 JRN INF gen.license_status=6U
21-02-14 17:26:59 MDS INF License PCSS.02706224.0000 is ACTIVE
```

The command above shows the latest events in the `stor1` cluster. The information on events (highlighted in red) is given in the table with the following columns:

Column	Description
TIME	Time when the event happened.
SYS	Component of the cluster where the event happened (e.g., MDS for an MDS server or JRN for local journal).
SEV	Event severity.
MESSAGE	Event description.

5.6.1 Exploring Basic Events

The table below describes the basic events displayed when you run the `vstorage top` utility.

Event	Severity	Description
MDS#<N> (<addr>:<port>) lags behind for more than 1000 rounds	JRN err	Generated by the MDS master server when it detects that MDS#<N> is stale. This message may indicate that some MDS server is very slow and lags behind.
MDS#<N> (<addr>:<port>) didn't accept commits for <i>M</i> sec	JRN err	Generated by the MDS master server if MDS#<N> did not accept commits for <i>M</i> seconds. MDS#<N> gets marked as <i>stale</i> . This message may indicate that the MDS service on MDS#<N> is experiencing a problem. The problem may be critical, resolve it as soon as possible.
MDS#<N> (<addr>:<port>) state is outdated and will do a full resync	JRN err	Generated by the MDS master server when MDS#<N> will do a full resync. MDS#<N> gets marked as <i>stale</i> . This message may indicate that some MDS server was too slow or disconnected for such a long time that it is not really managing the state of metadata and has to be resynchronized. The problem may be critical, resolve it as soon as possible.
MDS#<N> at <addr>:<port> became master	JRN info	Generated every time a new MDS master server is elected in the cluster. Frequent changes of MDS masters may indicate poor network connectivity and may affect the cluster operation.
The cluster is healthy with <i>N</i> active CS	MDS info	Generated when the cluster status changes to <i>healthy</i> or when a new MDS master server is elected. This message indicates that all chunk servers in the cluster are active and the number of replicas meets the set cluster requirements.

Continued on next page

Table 5.6.1.1 -- continued from previous page

Event	Severity	Description
The cluster is degraded with N active, M inactive, K offline CS	MDS warn	<p>Generated when the cluster status changes to <i>degraded</i> or when a new MDS master server is elected.</p> <p>This message indicates that some chunk servers in the cluster are</p> <ul style="list-style-type: none"> • inactive (do not send any registration messages) or • offline (are inactive for a period longer than <code>mds.wd.offline_tout = 5 min</code> (by default)).
The cluster failed with N active, M inactive, K offline CS (<code>mds.wd.max_offline_cs=<n></code>)	MDS err	<p>Generated when the cluster status changes to <i>failed</i> or when a new MDS master server is elected.</p> <p>This message indicates that the number of offline chunk servers exceeds <code>mds.wd.max_offline_cs</code> (2 by default). When the cluster fails, the automatic replication is not scheduled any more. So the cluster administrator must take some actions to either repair failed chunk servers or increase the <code>mds.wd.max_offline_cs</code> parameter. Setting the value of this parameter to 0 disables the failed mode completely.</p>
The cluster is filled up to $<N>$ %	MDS info/warn	<p>Shows the current space usage in the cluster. A warning is generated if the disk space consumption equals or exceeds 80%.</p> <p>It is important to have spare disk space for data replicas if one of the chunk servers fails.</p>
Replication started, N chunks are queued	MDS info	Generated when the cluster starts the automatic data replication to recover the missing replicas.
Replication completed	MDS info	Generated when the cluster finishes the automatic data replication.

Continued on next page

Table 5.6.1.1 -- continued from previous page

Event	Severity	Description
CS#<N> has reported hard error on <i>path</i>	MDS warn	Generated when the chunk server CS#<N> detects disk data corruption. You are recommended to replace chunk servers with corrupted disks as soon as possible with new ones and to check the hardware for errors.
CS#<N> has not registered during the last <i>T</i> sec and is marked as inactive/offline	MDS warn	Generated when the chunk server CS#<N> has been unavailable for a while. In this case, the chunk server first gets marked as inactive. After 5 minutes, the state is changed to offline, which starts the automatic replication of data to restore the replicas that were stored on the offline chunk server. You are recommended to replace chunk servers with corrupted disks as soon as possible with new ones and to check the hardware for errors.
Failed to allocate <i>N</i> replicas for ' <i>path</i> ' by request from «addr>:<port>» - <i>K</i> out of <i>M</i> chunks servers are available	MDS warn	Generated when the cluster cannot allocate chunk replicas, for example, when it runs out of disk space. You are recommended to replace chunk servers with corrupted disks as soon as possible with new ones and to check the hardware for errors.
Failed to allocate <i>N</i> replicas for ' <i>path</i> ' by request from «addr>:<port>» since only <i>K</i> chunk servers are registered	MDS warn	Generated when the cluster cannot allocate chunk replicas because not enough chunk servers are registered in the cluster. You are recommended to replace chunk servers with corrupted disks as soon as possible with new ones and to check the hardware for errors.

5.7 Monitoring Replication Parameters

When you configure replication parameters, keep in mind that the new settings do not come into effect immediately. For example, increasing the default replication parameter for data chunks may take some time to complete, depending on the new value of this parameter and the number of data chunks in the cluster.

To check that the new replication parameters have been successfully applied to your cluster:

1. Run the `vstorage -c <cluster_name> top` command.
2. Press **V** to display additional information about the cluster. Your command output should look similar to the following:

```
# vstorage -c stor1 top
connected to MDS#1
Cluster 'stor1': healthy
Space: [OK] allocatable 200GB of 211GB, free 211GB of 211GB
MDS nodes: 1 of 1, epoch uptime: 2h 21m
CS nodes: 2 of 2 (2 avail, 0 inactive, 0 offline)
License: PCSS.02444715.0000 is ACTIVE, 6399TB capacity
Replication: 3 norm, 2 limit
Chunks: [OK] 431 (100%) healthy, 0 (0%) degraded, 0 (0%) urgent,
        0 (0%) blocked, 0 (0%) offline, 0 (0%) replicating,
        0 (0%) overcommitted, 0 (0%) deleting, 0 (0%) void
<...>
```

3. Check the **Chunks** field for the following:
 - When decreasing the replication parameters, look for chunks that are in the **overcommitted** or **deleting** state. If the replication process is complete, no chunks with these states should be present in the output.
 - When increasing the replication parameters, look for chunks that are in the blocked or urgent state. If the replication process is complete, no chunks with these states should be present in the output. Besides, when the process is still in progress, the value of the healthy parameter is less than 100%.

Note: For more information on available chunk statuses, see *Exploring Chunk States* on page 106.

CHAPTER 6

Managing Cluster Security

This chapter describes some situations that may affect your cluster security.

6.1 Security Considerations

This section describes the security limitations you should keep in mind when deploying a Storage cluster.

Traffic sniffing

Storage does not protect you from traffic sniffing. Anyone who has access to your network can capture and analyze the data being sent and received through your network.

To learn how to keep your data secure, see *Securing Server Communication in the Cluster* on page 117.

Absence of users and groups

Storage does not use the concept of users and groups, providing specific users and groups with access to specific parts of a cluster. Anyone authorized to access a cluster can access all its data.

Non-encrypted data on disks

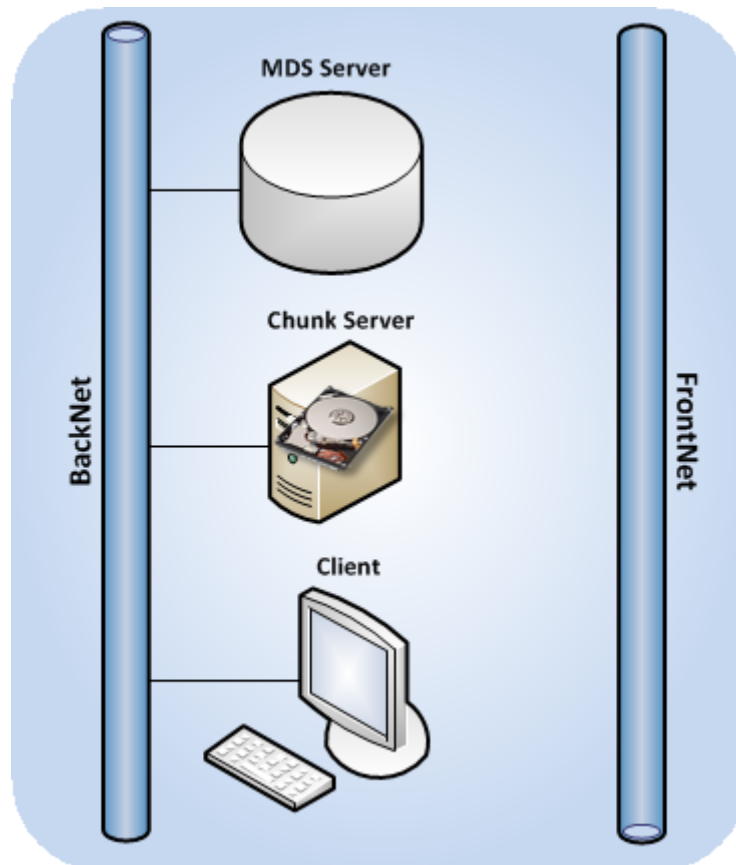
Storage does not encrypt data stored in a cluster. Attackers can immediately see the data once they gain access to a physical disk drive.

6.2 Securing Server Communication in the Cluster

A Storage cluster can contain three types of servers:

- MDS servers
- Chunk servers
- Clients

During cluster operation, the servers communicate with each other. To secure their communication, you should keep all servers on an isolated private network—BackNet. The figure below shows an example cluster configuration where all servers are set up on the BackNet.



The process of deploying such a configuration can be described as follows:

1. Create the cluster by making the MDS server and specifying one of its IP addresses:

```
# vstorage -c Cluster-Name make-mds -I -a MDS-IP-Address -r Journal-Directory -p
```

The specified address will then be used for MDS interconnection and intercommunication with the other servers in the cluster.

2. Set up a chunk server:

```
# vstorage -c Cluster-Name make-cs -r CS-Directory
```

Once it is created, the chunk server connects to the MDS server and binds to the IP address it uses to establish the connection. If the chunk server has several network cards, you can explicitly assign the chunk server to the IP address of a specific network card so that all communication between the chunk and MDS servers is carried out via this IP address.

To bind a chunk server to a custom IP address, you pass the `-a` option to the `vstorage make-cs` command when you create the chunk server:

```
# vstorage make-cs -r CS-Directory -a Custom-IP-Address
```

Note: A custom IP address must belong to the BackNet not to compromise your cluster security.

3. Mount the cluster on the client:

```
# vstorage-mount -c Cluster-Name Mount-Directory
```

Once the cluster is mounted, the client connects to the MDS and chunk server IP addresses.

This example configuration provides a high level of security for server communication because the MDS server, the chunk server, and the client are located on the isolated BackNet and cannot be compromised.

6.3 Cluster Discovery Methods

When choosing a cluster discovery method for your cluster, pay attention to the following:

- The recommended way of configuring auto-discovery for a Storage cluster is to use DNS records. For more details on this discovery method, see *Using DNS Records* on page 5.
- Zeroconf autodiscovery can be used for testing and evaluating the Storage functionality. This discovery method is not recommended for use in production for security reasons. Malicious users can potentially perform DoS attacks on the Zeroconf service even if you use security certificates for authentication in

your network.

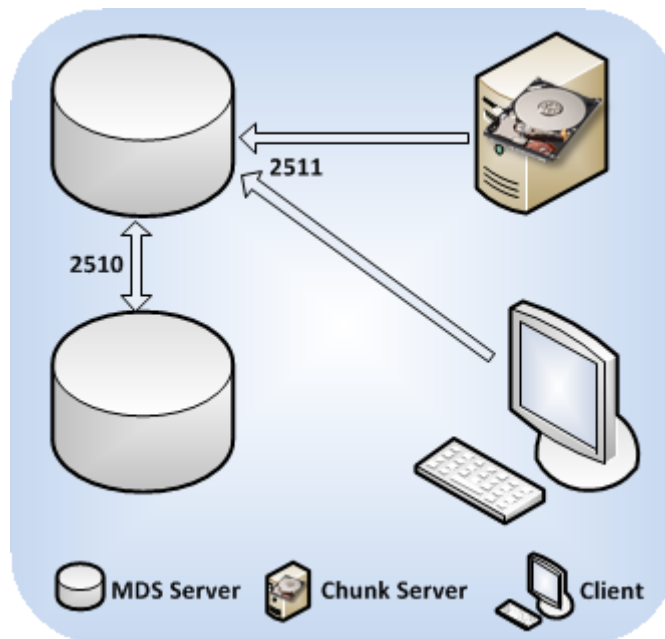
6.4 Storage Ports

This section lists the ports that must be open on servers that participate in a Storage cluster, in addition to the ports used by Virtuozzo.

MDS Servers

The following ports must be open on an MDS server:

- **Listening ports:** 2510 for incoming connections from other MDS servers and 2511 for incoming connections from chunk servers and clients
- **Outbound ports:** 2510 for outgoing connections to other MDS servers



By default, Storage uses port 2510 for communication between MDS servers and port 2511 for incoming connections from both chunk servers and clients. You can override the default ports when creating MDS servers:

1. Reserve two unoccupied consecutive ports.

The ports must be the same on all MDS servers you plan to set up in the cluster.

2. Execute the `vstorage make-mds` command to create the MDS server and specify the lower port after the

server IP address.

For example, to set up ports 4110 and 4111 for MDS communication in the `stor1` cluster, you can run this command:

```
# vstorage -c stor1 make-mds -I -a 10.30.100.101:4110 -r /vstorage/stor1-mds -p
```

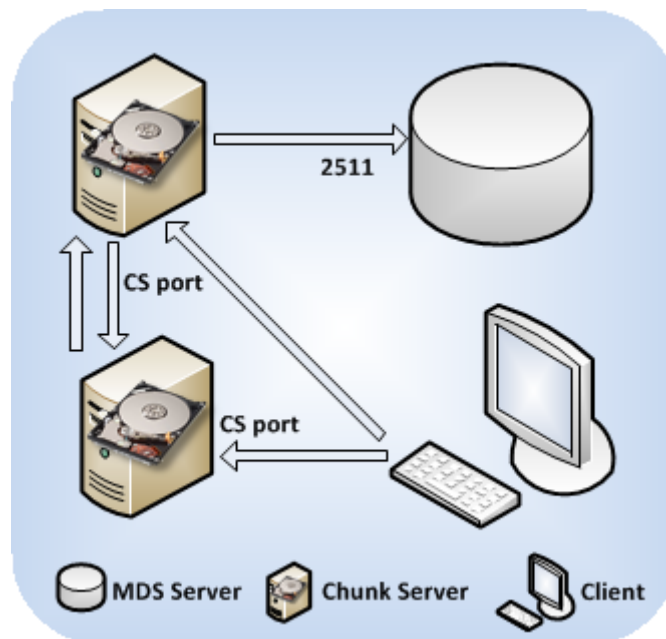
If you choose to use custom ports 4110 and 4111, do the following:

- On each MDS server with custom ports, open ports 4110 and 4111 for inbound traffic and port 4110 for outbound traffic.
- On all chunk servers and clients in the cluster, open port 4111 for outbound traffic.

Chunk Servers

The following ports must be open on a chunk server:

- **Listening ports:** a randomly chosen port for incoming connections from clients and other chunk servers
- **Outbound ports:** 2511 for outgoing connections to MDS servers and a randomly chosen port for outgoing connections to other chunk servers



The chunk server management service requires

- A port to communicate with MDS servers (either the default port 2511 or your custom port).
- A port to communicate with chunk servers and clients.

By default, the service binds to any available port. You can manually redefine the port by passing the `-a` option to the `vstorage make-cs` command when creating a chunk server. For example, to configure the management service to use port 3000, run this command:

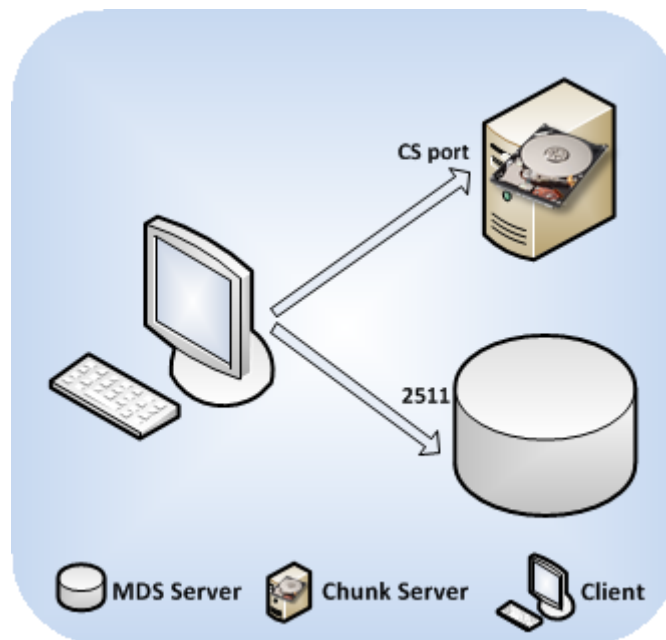
```
# vstorage make-cs -r /vstorage/stor1-cs -a 132.132.1.101:3000
```

Once you set a custom port, open this port for outbound traffic on all clients and other chunk servers in the cluster.

Clients

The following ports must be open on a client:

- **Listening ports:** no
- **Outbound ports:** 2511 for outgoing connections to MDS servers and a port configured as the listening one on chunk servers



By default, Storage automatically opens on a client the following outbound ports:

- For communication with MDS servers, the default port 2511.
- For communication with chunk servers, the port that was configured as the listening one on your chunk servers.

If you specify custom ports for MDS and chunk server communication, open these ports on the client for outgoing traffic. For example, if you configure port 4111 on MDS servers and port 3000 on chunk servers for communication with clients, open outbound ports 2511 and 3000 on the client.

6.5 Password-based Authentication

Storage uses password-based authentication to enhance security in clusters. Password authentication is mandatory meaning that you have to pass the authentication phase before you can add a new server to the cluster.

Password-based authentication works as follows:

1. You set the authentication password when you create the first MDS server in the cluster. The password you specify is encrypted and saved into the `/etc/vstorage/clusters/stor1/auth_digest.key` file on the server.
2. You add new MDS servers, chunk servers, or clients to the cluster and use the `vstorage auth-node` command to authenticate them. During authentication, you use the password you set when creating the first MDS server.
3. Storage compares the provided password with the one stored on the first MDS server, and if the passwords match, successfully authenticates the server.

For each physical server, authentication is a one-time process. Once a server is authenticated in the cluster (for example, when you configure it as an MDS server), the `/etc/vstorage/clusters/stor1/auth_digest.key` file is created on the authenticated server. When you set up this server as another cluster component (e.g., as a chunk server), the cluster checks that the `auth_digest.key` file is present and does not require you to authenticate the server again.

6.6 Installations via PXE Servers

Kickstart files you use to perform an unattended installation of Virtuozzo Hybrid Server 7 and Storage over a network contain a cluster authentication password in plain text. Attackers may intercept the password and gain access to your cluster. To secure your system, do one of the following:

- Physically isolate the network where the PXE server is located from all other (potentially insecure) networks.
- Install Virtuozzo Hybrid Server 7 via the PXE server but do not set up a Storage cluster. Once the installation is complete, manually deploy the cluster in your network using the Storage password-based authentication mechanism.

CHAPTER 7

Maximizing Cluster Performance

This chapter describes recommended configurations for Storage clusters and ways to tune them for maximum performance.

Please also see [Appendix A - Troubleshooting](#) on page 139 for common issues that might affect your cluster performance.

You may also consider updating hardware nodes in the cluster as described in [Keeping Your System Up To Date](#).

7.1 Checking Data Flushing

Before creating the cluster, you are recommended to check that all storage devices (hard disk drives, solid disk drives, RAIDs, etc.) you plan to include in your cluster can successfully flush data to disk when the server power goes off unexpectedly. Doing so will help you detect possible problems with devices that may lose data stored in their cache in the event of a power failure.

Storage ships with a special tool, `vstorage-hwflush-check`, for checking how a storage device flushes data to disk in an emergency case such as power outage. The tool is implemented as a client/server utility:

- **Client.** The client continuously writes blocks of data to the storage device. When a data block is written, the client increases a special counter and sends it to the server that keeps it.
- **Server.** The server keeps track of the incoming counters from the client so that it always knows the counter number the client will send next. If the server receives the counter that is less than the one

already stored on the server (e.g., because the power was turned off and the storage device did not flush the cached data to disk), the server reports an error.

To check that a storage device can successfully flush data to disk when the power fails, follow the procedure below:

On the server part:

1. On some computer running Virtuozzo Hybrid Server 7, install the `vstorage-hwflush-check` tool. This tool is part of the `vstorage-ctl` package and can be installed with this command:

```
# yum install vstorage-ctl
```

2. Run the `vstorage-hwflush-check` server:

```
# vstorage-hwflush-check -l
```

On the client part:

1. On the computer hosting a storage device you want to check, install the `vstorage-hwflush-check` tool:

```
# yum install vstorage-ctl
```

2. Run the `vstorage-hwflush-check` client, for example:

```
# vstorage-hwflush-check -s vstorage1.example.com -d /vstorage/stor1-ssd/test -t 50
```

Where

- `-s vstorage1.example.com` is the hostname of the computer where the `vstorage-hwflush-check` server is running.
 - `-d /vstorage/stor1-ssd/test` defines the directory to use for testing data flushing. During its execution, the client creates a file in this directory and writes data blocks to it.
 - `-t 50` sets the number of threads for the client to write data to disk. Each thread has its own file and counter. You can increase the number of threads (max. 200) to test your system in more stressful conditions. You can also specify other options when running the client. For more information on available options, see the `vstorage-hwflush-check` man page.
3. Wait for 10-15 seconds or more and power off the computer where the client is running, and then turn it on again.

Note: The **Reset** button does not turn off the power so you need to press the **Power** button or pull

out the power cord to switch off the computer.

- Restart the client by executing the same command you used to run it for the first time:

```
# vstorage-hwflush-check -s vstorage1.example.com -d /vstorage/stor1-ssd/test -t 50
```

Once launched, the client reads all written data, determines the version of data on the disk, and then restarts the test from the last valid counter. It then sends this valid counter to the server, and the server compares it with the latest counter it has. You may see output like:

```
id<N>:<counter_on_disk> -> <counter_on_server>
```

which means one of the following:

- If the counter on disk is lower than the counter on server, it means that the storage device has failed to flush the data to disk. Avoid using this storage device in production—especially for CS or journals—as you risk losing data.
- If the counter on disk is higher than the counter on server, it means that the storage device has flushed the data to disk but the client has failed to report it to the server. The network may be too slow or the storage device may be too fast for the set number of load threads so you may consider increasing it. This storage device can be used in production.
- If both counters are equal, it means the storage device has flushed the data to disk and the client has reported it to the server. This storage device can be used in production.

To be on the safe side, repeat the procedure several times. Once you check your first storage device, continue with all remaining devices you plan to use in the cluster. You need to test all devices you plan to use in the cluster: SSD disks used for client caching and CS journaling, disks used for MDS journals, disks used for chunk servers.

7.2 Exploring Possible Disk Drive Configurations

If the servers you plan to include in a Storage cluster have several disk drives, you can choose one of the following configurations for the cluster:

1. (Recommended with SSDs) No local RAIDs

A chunk server is set up per each hard drive and each data chunk has two or more replicas. This

configuration provides independence of disk failures, and the cluster replication adds reliability close to RAID 1 (mirroring without parity or striping). Solid-state drives are highly recommended for CS journaling, improving write commit latency (e.g., for databases).

2. **(Recommended) Passthrough individual drives attached to a hardware RAID controller (without RAID levels 0/1/10/5/6), with or without SSDs**

This highly recommended configuration is similar to the previous but is much faster. It features individual disks attached to a hardware RAID controller with a memory cache and a backup battery unit (BBU). RAID's write-back cache greatly improves random I/O write operations as well as database performance. The use of solid-state drives will further optimize random I/O, especially read operations.

3. **(Not recommended) Local striping RAID 0, data chunks with two or more replicas, with or without SSDs**

This configuration offers lower cluster reliability, because a single drive failure will cause the entire RAID 0 to fail, forcing the cluster to replicate more data each time such a failure occurs. Yet this can be considered a minor issue as Storage clusters perform parallel recovery from multiple servers, which is much faster than rebuilding a traditional RAID 1.

The use of solid-state drives for caching and journaling will further improve overall cluster performance and provide data checksumming and scrubbing to improve cluster reliability.

4. **(Not recommended) Local mirror RAID 1, data chunks with one or more replicas**

When deployed with one replica per each data chunk, this configuration does not provide high availability for your cluster if some of your servers fail. Besides, such configurations do not bring any disk space savings as they are equivalent to cluster mirroring, and a local RAID just doubles the data replication factor and saves cluster network traffic.

5. **(Highly not recommended) Local RAID 1, 5, or 6, data chunks with two or more replicas**

Avoid running Storage on redundant types of RAID like 1, 5, or 6 over local storage. In this case, a single write operation may affect a significant number of HDDs resulting in very poor performance. For example, for 3 Storage replicas and RAID5 on servers with 5 HDDs each, a single write operation may result in 15 I/O operations.

7.3 Carrying Out Performance Benchmarking

When testing the performance of a Storage cluster and comparing it with non-Virtuozzo Storage setups:

- Compare configurations with similar redundancy levels. For example, it is incorrect to compare the performance of a cluster with two or three replicas per data chunk with a standalone server that does not use any data redundancy, like RAID 1, 10, or 5.
- Take into account the usage of file system interfaces. Keep in mind that mounting a Storage cluster using the FUSE interface provides a convenient view into the cluster but is not optimal for performance. Therefore, do benchmarks from inside containers and virtual machines.
- Keep in mind that the data replication factor affects the cluster performance: clusters with two replicas are slightly faster than those with three replicas.

7.4 Using 1 GbE and 10 GbE Networks

1 Gbit/s Ethernet networks can deliver 110-120 MB/s, which is close to a single drive performance on sequential I/O. Since several drives on a single server can deliver higher throughput than a single 1 Gbit/s Ethernet link, networking may become a bottleneck.

However, in real-life applications and virtualized environments, sequential I/O is not common (backups mainly) and most of the I/O operations are random. Thus, typical HDD throughput is usually much lower, close to 10-20 MB/s, according to statistics accumulated from hundreds of servers by a number of major hosting companies.

Based on these two observations, we recommend to use one of the following network configurations (or better):

- A 1 Gbit/s link per each 2 HDDs on the Hardware Node. Although if you have 1 or 2 HDDs on a Hardware Node, two bonded network adapters are still recommended for better reliability (see [Setting Up Network Bonding](#) on page 128).
- A 10 Gbit/s link per Hardware Node for the maximum performance.

The table below illustrates how these recommendations may apply to a Hardware Node with 1 to 6 HDDs:

HDDs	1 GbE Links	10 GbE Links
1	1 (2 for HA)	1 (2 for HA)
2	1 (2 for HA)	1 (2 for HA)
3	2	1 (2 for HA)
4	2	1 (2 for HA)
5	3	1 (2 for HA)
6	3	1 (2 for HA)

Note the following:

- For the maximum sequential I/O performance, we recommend to use one 1Gbit/s link per each hard drive, or one 10Gbit/s link per Hardware Node.
- It is not recommended to configure 1 Gbit/s network adapters to use non-default MTUs (e.g., 9000-byte jumbo frames). Such settings require switch configuration and often lead to human errors. 10 Gbit/s network adapters, on the other hand, need to be configured to use jumbo frames to achieve full performance.
- For maximum efficiency, use the `balance-xor` bonding mode with the `layer3+4` hash policy. If you want to use the `802.3ad` bonding mode, also configure your switch to use the `layer3+4` hash policy.

7.5 Setting Up Network Bonding

Bonding multiple network interfaces together provides the following benefits:

1. High network availability. If one of the interfaces fails, the traffic will be automatically routed to the working interface(s).
2. Higher network performance. For example, two Gigabit interfaces bonded together will deliver about 1.7 Gbit/s or 200 MB/s throughput. The required number of bonded storage network interfaces may depend on how many storage drives are on the Hardware Node. For example, a rotational HDD can deliver up to 1 Gbit/s throughput.

To configure a bonding interface, do the following:

1. Create the `/etc/modprobe.d/bonding.conf` file containing the following line:

```
alias bond0 bonding
```

2. Create the `/etc/sysconfig/network-scripts/ifcfg-bond0` file containing the following lines:


```

DEVICE=bond0
ONBOOT=yes
BOOTPROTO=none
IPV6INIT=no
USERCTL=no
BONDING_OPTS="mode=balance-xor xmit_hash_policy=layer3+4 miimon=300 downdelay=300 \
updelay=300"
NAME="Storage net0"
NM_CONTROLLED=yes
IPADDR=xxx.xxx.xxx.xxx
PREFIX=24

```

Make sure to enter the correct values in the IPADDR and PREFIX lines.

The `balance-xor` mode is recommended, because it offers both fault tolerance and better performance. For more details, see the documents listed below.

3. Make sure the configuration file of each Ethernet interface you want to bond (e.g., `/etc/sysconfig/network-scripts/ifcfg-eth0`) contains the lines shown in this example:

```

DEVICE="eth0"
BOOTPROTO=none
NM_CONTROLLED="yes"
ONBOOT="yes"
TYPE="Ethernet"
HWADDR=xx:xx:xx:xx:xx:xx
MASTER=bond0
SLAVE=yes
USERCTL=no

```

4. Bring up the `bond0` interface:

```
# ifup bond0
```

5. Use `dmesg` output to verify that `bond0` and its slave Ethernet interfaces are up and links are ready.

Note: More information on network bonding is provided in the [Red Hat Enterprise Linux documentation](#) and [Linux Ethernet Bonding Driver HOWTO](#).

7.6 Using SSD Drives

Storage supports SSD drives formatted to the ext4 file system and optionally mounted with TRIM support enabled.

Note: Storage SSD usage scenario does not generate TRIM commands. Also, modern drives like Intel SSD DC S3700 do not need TRIM at all.

Along with all-flash setups where SSD drives are used for storing data chunks, Storage also supports hybrid clusters where SSDs are used for write journaling. You can attach an SSD drive to an HDD-based chunk server and configure the drive to store a write journal. By doing so, you can boost the performance of write operations in the cluster by up to two and more times.

The following sections provide detailed information on configuring SSD drives for write journaling and data caching.

Take note of the following:

- Not all solid-state drives obey flush semantics and commit data in accordance with the protocol. This may result in arbitrary data loss or corruption in case of a power failure. Always check your SSDs with the `vstorage-hwflush-check` tool (for more information, see [Checking Data Flushing](#) on page 123).
- It is recommended to use Intel SSD DC S3700 drives. However, you can also use Samsung SM1625, Intel SSD 710, Kingston SSDNow E or any other SSD drive with support for data protection on power loss. Some of the names of this technology are: Enhanced Power Loss Data Protection (Intel), Cache Power Protection (Samsung), Power-Failure Support (Kingston), Complete Power Fail Protection (OCZ). For more information, see [SSD Drives Ignore Disk Flushes](#) on page 142.

7.6.1 Calculating Write Journal Size

Using SSD drives for write journaling can help you reduce write latencies, thus improving the overall cluster performance. If you have multiple chunk servers on a single host, create a separate SSD journal for each CS.

To determine the size of each CS journal on the SSD, follow these guidelines:

1. Find out how many HDDs the SSD can service based on [Hardware Requirements](#). You can also use this formula:

$$\text{SSD_SSWS} * 0.8 / \text{HDD_SSWS}$$

Where:

- SSD_SSWS is the sustained sequential write speed of the SSD.
- HDD_SSWS is the sustained sequential write speed of an HDD (provided that identical HDD models

are used as recommended).

- 0.8 is the approximate percentage of error.

Note: The sustained sequential write speed is the average sequential write speed measured over a period of 60 seconds.

2. Reserve 20% of the SSD volume for normal operation, checksum storage, and metadata if needed. Typically, reserve 1GB of SSD space for checksums per each 1TB of HDD space. The rest of these 20% is reserved for emergency needs and also to prevent the SSD from filling up completely because its performance would then degrade.
3. Divide the remaining 80% of the SSD volume by the allowed number of HDDs.

For example, a 512 GB SSD rated at 1500 MB/s SSWS will be able to service $1500 * 0.8 / 150 = 8$ HDDs rated at 150 MB/s SSWS. And the journal size for each CS (i.e. HDD) will be $(512 - 20\%) / 8 = 51$ GB.

Note: Make sure your hardware follows the [Hardware Recommendations](#)

The following table gives a few examples of SSD models and the amount of HDDs they can service:

SSD type	Number of SSDs
Intel SSD 320 Series, Intel SSD 710 Series, Kingston SSDNow E enterprise series, or other SATA 3Gbps SSD models providing 150-200MB/s of sequential write of random data.	1 SSD per up to 3 HDDs
Intel SSD DC S3700 Series, Samsung SM1625 enterprise series, or other SATA 6Gbps SSD models providing at least 300MB/sec of sequential write of random data.	1 SSD per up to 5-6 HDDs

7.6.2 Setting Up Chunk Servers with a Journal on SSD

To set up a CS that stores a journal on an SSD drive, do the following:

1. Log in to the Node you want to act as a chunk server as root or as a user with root privileges. The Node must have at least one hard disk drive (HDD) and one solid state drive (SSD).

2. Download and install the following RPM packages: `vstorage-ctl`, `vstorage-libs-shared`, and `vstorage-chunk-server`.

These packages are available in the Virtuozzo remote repository (this repository is automatically configured for your system when you install Virtuozzo) and can be installed with this command:

```
# yum install vstorage-chunk-server
```

3. Make sure that cluster discovery is configured for the server. For details, see [Configuring Cluster Discovery](#) on page 4.

4. Authenticate the server in the cluster, if it is not yet authenticated:

```
# vstorage -c stor1 auth-node
```

5. If required, prepare the SSD as described in [Preparing Disks for Storage](#) on page 8.

6. Create the chunk server configuration, repository, and the journal, for example:

```
# vstorage -c stor1 make-cs -r /vstorage/stor1-cs -j /ssd/stor1/cs1 -s 30720 -T ssd
```

This command:

- 6.1. Makes the `/vstorage/stor1-cs` directory on your computer's hard disk drive and configures it for storing data chunks.
- 6.2. Configures your computer as a chunk server and joins it to the `stor1` Storage cluster.
- 6.3. Creates the journal in the `/ssd/stor1/cs1` directory on the SSD drive and allocates 30 GB of disk space to this journal.
- 6.4. Tunes the journal for improved write and read performance on SSD, with best results noticeable on random writes on all-flash clusters.

Note: When choosing a directory for the journal and deciding on its size, allocate the required space for the journal and make sure there is 1GB of SSD space per each 1TB of HDD space for checksums.

7. Start the chunk server management service `vstorage-csd` and configure it to start automatically on the chunk server boot:

```
# systemctl start vstorage-csd.target  
# systemctl enable vstorage-csd.target
```

7.6.3 Adding, Destroying, and Configuring CS Journals in Live Storage Clusters

Note: To obtain CS repository paths, use the `vstorage -c <cluster> list-services -C` command.

7.6.3.1 Adding CS Journals

To add a new journal to a chunk server, use the `vstorage configure-cs -a -s` command. For example, to add a 2048MB journal to the chunk server CS#1, place it in a directory on an SSD drive mounted to `/ssd`, and tune the journal for SSDs, run:

```
# vstorage -c stor1 configure-cs -r <CS_repository_path> \  
-a /ssd/stor1-cs1-journal -s 2048 -T ssd
```

Restart the CS service to apply changes implied by the `-T` parameter:

```
# systemctl status vstorage-csd* | grep -m1 "stor1-cs1"  
vstorage-csd.stor1.1027.service - vstorage-csd(/vstorage/stor1-cs1)  
# systemctl restart vstorage-csd.stor1.1027.service
```

7.6.3.2 Destroying CS Journals

To destroy a chunk server journal, use the `vstorage configure-cs -d` command. For example:

```
# vstorage -c stor1 configure-cs -r <CS_repository_path> -d
```

7.6.3.3 Moving CS Journals

To change the chunk server journal directory, do the following using the commands above:

1. Destroy the existing journal
2. Add a new journal with the required size at the required location.

7.6.3.4 Resizing CS Journals

To resize a chunk server journal, use the `vstorage configure-cs -s` command. For example, to resize a CS journal to 4096MB:

```
# vstorage -c stor1 configure-cs -r <CS_repository_path> -s 4096
```

7.6.4 Disabling Checksumming

Using checksumming, you can provide better reliability and integrity of all data in the cluster. When checksumming is enabled, Storage generates checksums each time some data in the cluster is modified. When this data is then read, the checksum is computed once more and compared with the already existing value.

By default, data checksumming is automatically enabled for newly created chunk servers that use journaling. If necessary, you can disable this functionality using the `-s` option when you set up a chunk server, for example:

```
# vstorage -c stor1 make-cs -r /vstorage/stor1-cs -j /ssd/stor1/cs1 -s 30720 -S
```

7.6.5 Configuring Data Scrubbing

Data scrubbing is the process of checking data chunks for durability and verifying their contents for readability and correctness. By default, Storage is set to examine two data chunks per minute on each chunk server in the cluster. If necessary, you can configure this number using the `vstorage` utility, for example:

```
# vstorage -c stor1 set-config mds.wd.verify_chunks=3
```

This command sets the number of chunks to be examined on each chunk server in the `stor1` cluster to 3.

7.7 Improving High-Capacity HDD Performance

Unlike older hard disks with 512-byte sectors, many modern HDDs (3TB and more in capacity) use 4KB physical sectors. In certain cases, this can greatly reduce system performance (by 3-4 times) due to extra Read-Modify-Write (RMW) cycles required to align the source write request. Why this happens? When an

operating system issues an unaligned write request, the HDD has to align the beginning and end of that request to 4KB boundaries. To do this, the HDD reads the request's head and tail ranges to determine an even number of sectors to modify. For example, on a request to write a 4KB block at a 2KB offset, HDD will read the 0-2KB and 6-8KB ranges to modify the entire 0-8KB data range.

The typical reasons of poor performance with 4KB sector HDDs are:

1. Host OS file system unaligned on the 4KB boundary. The `make-cs` command of Storage tries to detect and report such issues to the administrator in advance, but be aware that the `fdisk` utility is not recommended for partitioning HDDs. You should use `parted` instead.
2. Unaligned writes (e.g., 1KB) performed by guest OS. Many legacy operating systems, like Microsoft Windows XP and Windows Server 2003 or Red Hat Enterprise Linux 5.x, have unaligned partitions by default and generate unaligned I/O patterns which are quite slow on both Storage and actual HDDs with 4KB sectors. If you plan running such legacy operating systems, consider the following:
 - Using smaller HDDs with 512-byte sectors, or use SSD journaling for CS services which mitigates the issue to some extent.
 - Aligning OS partitions properly as described in [Aligning Disks and Partitions in Virtual Machines](#).

You can check for unaligned write operations in the cluster by as follows:

1. Run the `vstorage top` or `stat` command. For example:

```
# vstorage -c stor1 top
```

2. Press `i` to display the **RMW** and **JRMW** columns in the CS part of the `top` output.
3. Check the **RMW** or **JRMW** counters, which are explained below.
 - When SSD journaling is used, the **RMW** counter shows the number of requests which lead to Read-Modify-Write cycles, while the **JRMW** counter shows the number of Read-Modify-Write cycles mitigated by the use of SSD journals.
 - When SSD journaling is not used, the **JRMW** counter shows the number of unaligned requests which potentially generate Read-Modify-Write cycles on the HDD in question.

7.7.1 Improving Virtual Disk Performance

This section lists ways of improving the performance of virtual disks.

7.7.1.1 Preventing Partition Misalignment in Legacy Operating Systems

Virtual disks in virtual machines running legacy guest operating systems such as Windows Server 2003, Windows XP, Windows 2000, CentOS 5, or RHEL 5 may work slower because of partition misalignment. For solutions to this issue, see [Aligning Disks and Partitions in Virtual Machines](#).

7.8 Disabling Inter-Tier Data Allocation

If a storage tier runs out of free space, Storage will attempt to temporarily use the space of the lower tiers down to the lowest. If the lowest tier also becomes full, Storage will attempt to use a higher one. If you add more storage to the original tier later, the data, temporarily stored elsewhere, will be moved to the tier where it should have been stored originally.

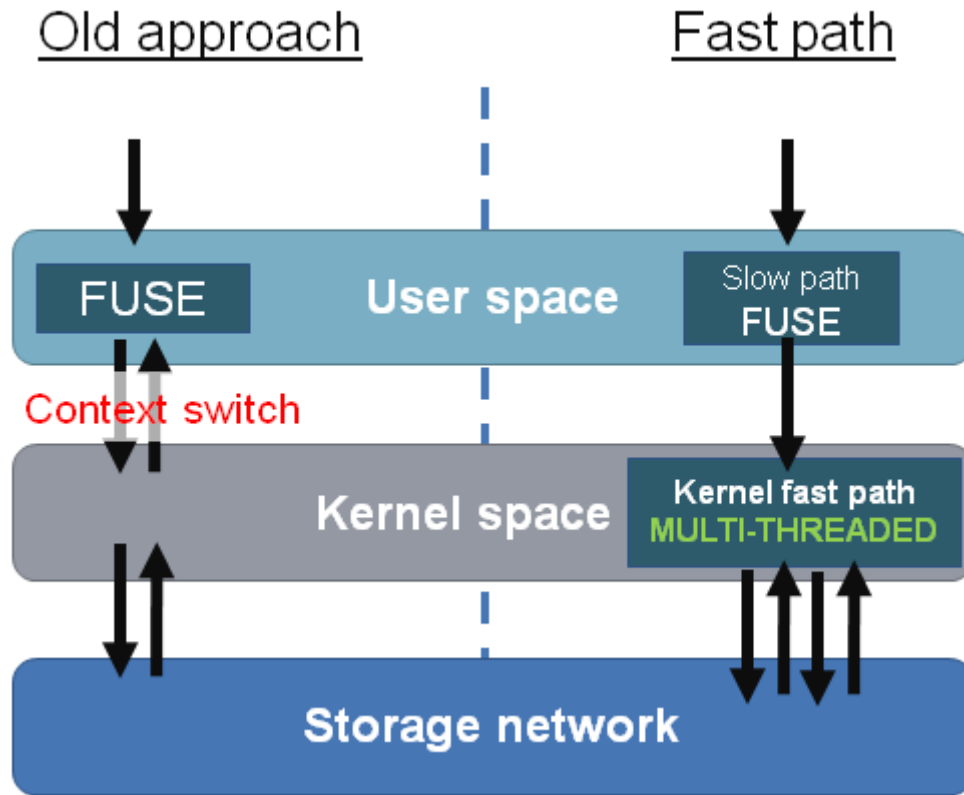
Mixing tier workloads is not recommended as it may decrease cluster performance. To prevent this, you can disable automatic data migration between tiers after making sure that each tier have enough free space. Execute the following command on any cluster node:

```
# vstorage -c cluster1 set-config mds.alloc.strict_tier=1
```

7.9 Managing Fast Path

Fast path is a feature that boosts Storage read performance if node I/O is a bottleneck. In previous versions of Storage, performance could be limited by node I/O being handled in a single thread in user space. In the current version of Storage, the fast path is enabled by default, and I/O is handled using multi-threading in kernel space, eliminating unnecessary context switches and improving performance.

Note: Starting from Virtuozzo Hybrid Server 7.5 Update 5, you cannot turn off the fast path option. The `fuse_kio_pcs` fast path module is a prerequisite for the `vstorage-mount` service, and it cannot be unloaded on the nodes where Storage is mounted.



If node I/O is a bottleneck, the fast path may increase maximum node read performance up to three times in some cases.

Note: Typically, node I/O may be a bottleneck in clusters with cache on SSD or all-SSD setups.

7.9.1 Disabling Fast Path in Virtuozzo Hybrid Server 7.5 Update 4 and Earlier Versions

Ways to disable the feature differ, depending on whether or not you use the GUI.

Important: We highly recommend not disabling the fast path for production clusters unless you should turn it off for debugging. In all other cases, ensure the fast path is enabled.

To disable the fast path on a GUI-enabled deployment, do the following on each node:

1. Set 'kdirect.enable' to 0 in '/etc/vstorage/vstorage-mount.conf'.
2. Stop all VEs on the node or migrate them to another node using `prlctl migrate` or Virtuozzo Automator.
3. Stop the `vstorage-ui-agent` service:

```
# systemctl stop vstorage-ui-agent.service
```

4. Unmount the storage location:

```
# umount /vstorage/<cluster_name>
```

5. Start the `vstorage-ui-agent` service again:

```
# systemctl start vstorage-ui-agent.service
```

The service will remount the storage location automatically.

6. Enable the [VE online compacting feature](#) that is turned off automatically when disabling the fast path:

```
# vstorage -c <cluster_name> set-config gen.do_punch_hole=1
```

7. Migrate all VEs back.

To disable the fast path on a deployment without GUI, do the following on each node:

1. Set 'kdirect.enable' to 0 in '/etc/vstorage/vstorage-mount.conf'.
2. Stop all VEs on the node or migrate them to another node using `prlctl migrate` or Virtuozzo Automator.
3. Unmount and remount the storage location:

```
# umount /vstorage/<cluster_name>
# mount -a /vstorage/<cluster_name>
```

4. Enable the [VE online compacting feature](#) that is turned off automatically when disabling the fast path:

```
# vstorage -c <cluster_name> set-config gen.do_punch_hole=1
```

5. Migrate all VEs back.

CHAPTER 8

Appendices

8.1 Appendix A - Troubleshooting

This chapter describes common issues you may encounter when working with Storage clusters and ways to resolve them. The main tool you use to solve cluster problems and detect hardware failures is `vstorage top`.

8.1.1 Submitting Problem Reports to Technical Support

If your cluster is experiencing a problem that you cannot solve on your own, you can use the `vstorage make-report` command to compile a detailed report about the cluster. You can then send the report to the support team who will closely examine your problem and make their best to solve it as quickly as possible.

To generate a report:

1. Configure passwordless SSH access for the `root` user from the server where you plan to run the `vstorage make-report` command to all servers that participate in the cluster.

The easiest way to do this is to create an SSH key with `ssh-keygen` and use `ssh-copy-id` to configure all servers to trust this key. For details, see the man pages for `ssh-keygen` and `ssh-copy-id`.

2. Run the `vstorage make-report` command to compile the report:

```
# vstorage -c stor1 make-report  
The report is created and saved to vstorage-report-20121023-90630.tgz
```

The command collects cluster-related information on all servers participating in the cluster and saves it to a file in your current working directory. You can find the exact file name by checking the `vstorage` output (`vstorage-report-20121023-90630.tgz` in the example above).

If necessary, you can save the report to a file with a custom name and put it to a custom location. To do this, pass the `-f` option to the command and specify the desired file name (with the `.tgz` extension) and location, for example:

```
# vstorage -c stor1 make-report -f /home/reportSTOR1.tgz
```

Once it is ready, [submit the report](#) to the support team.

Note: The report contains only information related to your cluster settings and configuration. It does not contain any private information.

8.1.2 Out of Disk Space

When very little free disk space remains in a Storage cluster, it is critically important to increase it as soon as possible by adding more chunk servers or removing some data. As soon as 95% of cluster disk space becomes occupied, the allocation of new data chunks is no longer possible and such requests are blocked until the cluster can satisfy the demand. As a result, user I/O becomes blocked as well, effectively freezing containers and virtual machines.

Note: It is highly recommended to keep at least 10% of disk space free for recovery in case of host machine failures. You should also monitor usage history, for example, using the `vstorage top` or `vstorage get-event` commands (for more information, see [Monitoring Storage Clusters](#) on page 97).

8.1.2.1 Symptoms

1. Stuck I/O or unresponsive mount point, `dmesg` messages about stuck I/O, frozen containers and virtual machines.
2. `vstorage top` and `vstorage get-event` show error messages like “Failed to allocate X replicas at tier Y since only Z chunk servers are available for allocation”.

8.1.2.2 Solutions

1. Remove any unnecessary data to free disk space.

Note: Additional effect which may surprise at first is that as soon as I/O queues in the kernel are full with the blocked I/O, a mount point on the client machine may stuck responding altogether and no longer be able to service the requests even such as file listing. In this case an additional mount point can be created to list, access and remove the unneeded data.

2. Add more Chunk Servers on unused disks (see *Setting Up Chunk Servers* on page 16).

If the solutions above are not possible, you can use one of the following *temporary* workarounds:

1. Lower the replication factor for some of the least critical user data (see *Configuring Replication Parameters* on page 29). Remember to revert the changes afterwards.
2. Reduce the allocation reserve. For example, for cluster stor1:

```
# vstorage -c stor1 set-config mds.alloc.fill_margin=2
```

Where `mds.alloc.fill_margin` is the percentage of reserved disk space for CS operation needs (the default value is 5). Remember to revert the changes afterwards.

8.1.3 Poor Write Performance

Some network adapters, like RTL8111/8168B, are known to fail to deliver full-bandwidth, full-duplex network traffic. This can result in poor write performance.

So before deploying a Storage cluster, you are highly recommended to test networks for full-duplex support. You can use the `netperf` utility to simultaneously generate in and out traffic. For example, in 1 GbE networks, it should constantly deliver about 2 Gbit/s of total traffic (1 Gbit/s for incoming and 1 Gbit/s for outgoing traffic).

8.1.4 Poor Disk I/O Performance

In most BIOS setups, AHCI mode is configured to work by default with the **Legacy** option enabled. With this option, your servers work with SATA devices via the legacy IDE mode, which may affect the cluster

performance, making it up to 2 times slower than expected. You can check that the option is currently enabled by running the `hdparm` command, for example:

```
# hdparm -i /dev/sda
...
PIO modes:  pio0 pio1 pio2 pio3 *pio4
DMA modes:  mdma0 mdma1 mdma2
UDMA modes: udma0 udma1 udma2 udma3 udma4 udma5 udma6
```

The asterisk before `pio4` in the **PIO modes** field indicates that your hard drive `/dev/sda` is currently operating in the legacy mode.

To solve this problem and maximize the cluster performance, always enable the **AHCI** option in your BIOS settings.

8.1.5 Hardware RAID Controller and Disk Write Caches

It is important that all hard disk drives obey the “flush” command and write their caches before the command completes. Unfortunately, not all hardware RAID controllers and drives do this, which may lead to data inconsistencies and file system corruptions on a power failure.

Some 3ware RAID controllers do not disable disk write caches and do not send “flush” commands to disk drives. As a result, the file system may sometimes become corrupted even though the RAID controller itself has a battery. To solve this problem, disable writes caches on all disk drives in the RAID.

Also, make sure that your configuration is thoroughly tested for consistency before deploying a Storage cluster. For information on how you can do this, see *SSD Drives Ignore Disk Flushes* on page 142.

8.1.6 SSD Drives Ignore Disk Flushes

A lot of desktop-grade SSD drives can ignore disk flushes and fool operating systems by reporting that data was written while it was actually not. Examples of such drives are OCZ Vertex 3 and Intel X25-E, X-25-M G2 that are known to be unsafe on data commits. Such disk drives should not be used with databases and may easily corrupt the file system on a power failure.

The 3rd generation Intel SSD drives (S3700 and 710 series) do not have these problems, featuring a capacitor to provide a battery backup for flushing the drive cache when the power goes out.

Use SSD drives with care and only when you are sure that drives are server-grade drives and obey “flush” rules. For more information on this issue, read [this article about PostgreSQL](#).

8.1.7 Cluster Cannot Create Enough Replicas

Sometimes, the cluster might not create the required number of data chunks even if enough chunk servers are present in the cluster.

This may be the case when you create new chunk servers by making copies of an existing chunk server (e.g., you set up a chunk server in a virtual machine and then clone this machine). In this case, all copied chunk servers have the same UUID — that is, the UUID of the original server. The cluster has information that all chunk servers are located on the original host and cannot allocate new data chunks.

To solve the problem, generate a new UUID for a cloned chunk server by running the following command on the destination host:

```
# /usr/bin/uuidgen -r | tr '-' ' ' | awk '{print $1$2$3}' > /etc/vstorage/host_id
```

For more information on the `uuidgen` utility, see its man page.

8.1.8 Failed Chunk Servers

If a chunk server in your Storage cluster fails, you need to identify the cause of failure and choose a correct way to solve the problem.

Do the following:

1. Run the `vstorage top` command. For example:

```
# vstorage -c stor1 top
```

2. Press `i` to cycle to the `FLAGS` column in the chunk server section and find the flags corresponding to the failed CS.
3. Find the shown flags in the table below to identify the cause of failure and the way to solve the problem.

Flag	Issue	What to do
H	An I/O error. The disk on which the chunk server runs is broken.	Check the disk for errors. If the disk is broken, replace it and recreate the CS as described in <i>Replacing Disks Used as Chunk Servers</i> on page 144. Otherwise, contact technical support.

Continued on next page

Table 8.1.8.1 -- continued from previous page

Flag	Issue	What to do
h	A chunk checksum mismatch. Either the chunk is corrupt or the disk where the chunk is stored is broken.	Check the disk for errors. If the disk is broken, replace it and recreate the CS as described in <i>Replacing Disks Used as Chunk Servers</i> on page 144. Otherwise, contact technical support.
S	The CS journal stored on a journaling SSD is not accessible. Either the journal is corrupt or the journaling SSD is broken.	Check the journaling SSD for errors. If the disk is broken, replace it as described in <i>Failed Write Journaling SSDs</i> on page 145.
R	The path to the chunk repository is invalid on CS start. The disk on which the chunk server runs is not attached or mounted.	Make sure the disk is attached and correctly mounted. Make sure the disk's entry in <code>/etc/fstab</code> is correct.
T	An I/O request timeout. The disk may only be inaccessible for some reason and not necessarily broken.	Make sure the disk is attached and check <code>dmesg</code> output for I/O request timeout messages to find out why the disk might be inaccessible.

8.1.8.1 Replacing Disks Used as Chunk Servers

To replace a broken HDD or SSD disk used as a chunk server, do the following:

1. Remove the failed CS from the cluster as described in *Removing Chunk Servers* on page 25.

Note: Do not detach the broken disk until you remove the CS.

2. Replace the broken disk with a new one.
3. Prepare the SSD as described in *Preparing Disks for Storage* on page 8.
4. Create a new CS as described in *Stage 2: Creating a Chunk Server* on page 17.

8.1.9 Failed Write Journaling SSDs

If the SSD used to store write journals breaks, all chunk servers which had journals on this SSD will fail. The cluster will continue to work and will create new replicas to make up for those which have been lost. If you need to set up same write journals on a replacement SSD, do the following:

1. Remove the failed chunk servers as described in *Removing Chunk Servers* on page 25.
2. Prepare the SSD as described in *Preparing Disks for Storage* on page 8.
3. Create new chunk servers which will keep write journals on the new SSD as described in *Using SSD Drives* on page 129.

8.1.10 Failed MDS Servers

If the disk hosting an MDS server fails, replace it as follows:

1. Delete the failed MDS server as described in *Removing MDS Servers* on page 24.
2. Create a new MDS server as described in *Adding MDS Servers* on page 23.

8.2 Appendix B - Frequently Asked Questions

This Appendix lists most frequently asked questions about Storage clusters.

8.2.1 General

- **Can /pstorage directory still be used on newer installations?**

Yes. In newer installations, /pstorage remains as a symlink to the new /vstorage directory for compatibility purposes.

- **Do I need to buy additional storage hardware for Storage?**

No. Storage eliminates the need for external storage devices typically used in SANs by converting locally attached storage from multiple nodes into a shared storage.

- **What are the hardware requirements for Storage?**

Storage does not require any special hardware and can run on commodity computers with traditional SATA drives and 1 GbE networks. Some hard drives and RAID controllers, however, ignore the FLUSH command to imitate better performance and must not be used in clusters as this may lead to file system or journal corruptions. This is especially true for RAID controllers and SSD drives. Please consult with your hard drive's manual to make sure you use reliable hardware.

For more information, see [Hardware Requirements](#).

- **How many servers do I need to run a Storage cluster?**

You need only one physical server to start using Storage. However, to provide high availability for your data, you are recommended to have at least 3 replicas per each data chunk. For this, you will need at least three nodes (and at least five in total) in the cluster. For details, see [|vstorage| Configurations](#) and [Configuring Replication Parameters](#) on page 29.

- **Can I join Hardware Nodes running different supported operating systems into a single Storage cluster?**

Yes. You can create Storage clusters of Hardware Nodes running any combination of supported operating systems. For example, you can have metadata servers on Hardware Nodes with Ubuntu 14.04, chunk servers on Hardware Nodes with Red Hat Enterprise Linux 7, and clients on computers with CentOS 7.

Note: The current standalone version of Storage does not support Virtuozzo.

8.2.2 Scalability and Performance

- **How many servers can I join to a Storage cluster?**

There is no strict limit on the number of servers you can include in a Storage cluster. However, you are recommended to limit the servers in the cluster to a single rack to avoid any possible performance degradation due to inter-rack communications.

- **How much disk space can a Storage cluster have?**

A Storage cluster can support up to 8 PB of effective available disk space, which totals to 24 PB of physical disk space when 3 replicas are kept for each data chunk.

- **Can I add nodes to an existing Storage cluster?**

Yes, you can dynamically add and remove nodes from a Storage cluster to increase its capacity or to take nodes offline for maintenance. For more information, see *Configuring Chunk Servers* on page 25.

- **What is the expected performance of a Storage cluster?**

The performance depends on the network speed and the hard disks used in the cluster. In general, the performance should be similar to locally attached storage or better. You can also use SSD caching to increase the performance of commodity hardware by adding SSD drives to the cluster for caching purposes. For more information, see *Using SSD Drives* on page 129.

- **What performance should I expect on 1-gigabit Ethernet?**

The maximum speed of a 1 GbE network is close to that of a single **rotational** drive. In most workloads, however, random I/O access is prevalent and the network is usually not a bottleneck. Research with large service providers has proved that average I/O performance rarely exceeds 20 MB/sec due to randomization. Virtualization itself introduces additional randomization as multiple independent environments perform I/O access simultaneously. Nevertheless, 10-gigabit Ethernet will often result in better performance and is recommended for use in production.

- **Will the overall cluster performance improve if I add new chunk servers to the cluster?**

Yes. Since data is distributed among all hard drives in the cluster, applications performing random I/O experience an increase in IOPS when more drives are added to the cluster. Even a single **client machine** may get noticeable benefits by increasing the number of chunk servers and achieve performance far beyond traditional, locally attached storage.

- **Does performance depend on the number of chunk replicas?**

Each additional replica degrades write performance by about 10%, but at the same time it may also improve read performance because the Storage cluster has more options to select a faster server.

8.2.3 Availability

- **How does Storage protect my data?**

Storage protects against data loss and temporary unavailability by creating data copies (replicas) and storing them on different servers. To provide additional reliability, you can configure Storage to maintain user data checksums and verify them when necessary.

- **What happens when a disk is lost or a server becomes unavailable?**

Storage automatically recovers from a degraded state to the specified redundancy level by replicating

data on live servers. Users can still access their data during the recovery process.

- **How fast does Storage recover from a degraded state?**

Since Storage recovers from a degraded state using all the available hard disks in the cluster, the recovery process is much faster than for traditional, locally attached RAID5. This makes the reliability of the storage system significantly better as the probability of losing the only remaining copy of data during the recovery period is very small.

- **Can I change redundancy settings on the fly?**

Yes, at any point you can change the number of data copies, and Storage will apply the new settings by creating new copies or removing unneeded ones. For more details on configuring replication parameters, see *Configuring Replication Parameters* on page 29.

- **Do I still need to use local RAID5?**

No, Storage provides the same built-in data redundancy as a mirror RAID1 array with multiple copies. However, for better sequential performance, you can use local striping RAID0 exported to your Storage cluster. For more information on using RAID5, see *Exploring Possible Disk Drive Configurations* on page 125.

- **Does Storage have redundancy levels similar to RAID5?**

No. To build a reliable software-based RAID5 system, you also need to use special hardware capabilities like backup power batteries. In the future, Storage may be enhanced to provide RAID5-level redundancy for read-only data such as backups.

- **What is the recommended number of data copies?**

It is recommended to configure Storage to maintain 2 or 3 copies, which allows your cluster to survive the simultaneous loss of 1 or 2 hard drives.

8.2.4 Cluster Operation

- **How do I know that the new replication parameters have been successfully applied to the cluster?**

To check whether the replication process is complete, run the `vstorage top` command, press the V key on your keyboard, and check information in the **Chunks** field:

- When decreasing the replication parameters, no chunks in the **overcommitted** or **deleting** state

should be present in the output.

- When increasing the replication parameters, no chunks in the **blocked** or **urgent** state should be present in the output. Besides, the overall cluster health should equal 100%.

For details, see *Monitoring Replication Parameters* on page 114.

- **How do I shut down a cluster?**

To shut down a Storage cluster:

1. Stop and disable all the required cluster services.
2. Stop all running containers and virtual machines.
3. Stop all clients.
4. Stop all chunk servers.
5. Stop all MDS servers.

For details, see *Shutting Down and Starting Up Cluster Nodes* on page 41.

- **What tool do I use to monitor the status and health of a cluster?**

You can monitor the status and health of your cluster using the `vstorage top` command. For details, see *Monitoring Storage Clusters* on page 97.

To view the total amount of disk space occupied by all user data in the cluster, run the `vstorage top` command, press the `V` key on your keyboard, and look for the **FS** field in the output. The **FS** field shows how much disk space is used by all user data in the cluster and in how many files these data are stored. For details, see *Understanding Disk Space Usage* on page 103.

- **How do I configure a Virtuozzo server for a cluster?**

To prepare a server with Virtuozzo for work in a cluster, you simply tell the server to store its containers and virtual machines in the cluster rather than on its local disk. For details, see *Stage 3: Configuring Virtual Machines and Containers* on page 20.

- **Why vmstat/top and Storage stat show different IO times?**

The `vstorage` and `vmstat/top` utilities use different methods to compute the percentage of CPU time spent waiting for disk IO (**wa%** in `top`, **wa** in `vmstat`, and **IOWAIT** in `vstorage`). The `vmstat` and `top` utilities mark an idle CPU as waiting only if an outstanding IO request is started on that CPU, while the `vstorage` utility marks all idle CPUs as waiting, regardless of the number of IO requests waiting for IO. As a result, `vstorage` can report much higher IO values. For example, on a system with 4 CPUs and one

thread doing IO, `vstorage` will report over 90% IOWAIT time, while `vmstat` and `top` will show no more than 25% IO time.

- **What effect tier numbering has on Storage operation?**

When assigning storage to tiers, have in mind that faster storage drives should be assigned to higher tiers. For example, you can use tier 0 for backups and other cold data (CS without SSD journals), tier 1 for virtual environments—a lot of cold data but fast random writes (CS with SSD journals), tier 2 for hot data (CS on SSD), journals, caches, specific virtual machine disks, and such.

This recommendation is related to how Storage works with storage space. If a storage tier runs out of free space, Storage will attempt to temporarily use a lower tier. If you add more storage to the original tier later, the data, temporarily stored elsewhere, will be moved to the tier where it should have been stored originally.

For example, if you try to write data to the tier 2 and it is full, Storage will attempt to write that data to tier 1, then to tier 0. If you add more storage to tier 2 later, the aforementioned data, now stored on the tier 1 or 0, will be moved back to the tier 2 where it was meant to be stored originally.

8.2.5 Shaman Service

- **What is the role of the shaman master?**

The master instance of the shaman service watches over nodes availability. When it detects that a node is not available, it revokes the node's access to the storage cluster and relocates resources from the unavailable node to other nodes in the cluster.

- **How is the shaman master elected? What ports are used for communication between cluster members to achieve this? What is the algorithm used for election and also for shaman workers to determine that the master is dead?**

Shaman uses the Storage file system for:

1. Master election
2. Node availability detection
3. Communication between the master and worker instances

It does not use networking directly. All communications go via the Storage file system. Storage itself uses the Paxos algorithm to reach a consensus between its members. By default, MDS servers use TCP ports 2510 and 2511 to exchange messages.

Following is a detailed description of how shaman utilizes the Storage file system for the above purposes.

A file on the Storage file system can be opened either in the exclusive (read-write) mode or the shared (read-only) mode. When a process opens a file, the master MDS instance grants the process a temporal lease for that file (exclusive or shared, respectively). The lease must be periodically refreshed by the process. The default timeout for a lease is 60 seconds. If the process that has obtained the lease fails to refresh it within 60 seconds, the master MDS instance revokes the lease from the process.

An exclusive lease granted for a file prevents other processes from obtaining a lease for that file. A process trying to open a file for which an exclusive lease has been granted receives an error. Even if the lease is granted to a process on another node.

Each shaman instance interacts with at least two files:

1. `.shaman/master`. Whoever takes an exclusive lease on this file becomes the master. Other instances periodically try to open this file. If a master instance becomes unavailable, then, after the lease timeout elapses, some other shaman instance may finally obtain the lease on this file and become a new master.
2. `.shaman/md.$host_id/lock`. On start, a shaman instance opens this file and refreshes its lease until the service is stopped. The master instance periodically tries opening the `.shaman/md.*/lock` files. If it succeeds, that means the shaman instance on the corresponding `$host_id` failed to refresh the lease for 60 seconds. The master instance treats this as a node crash event. It revokes the node access to the Storage file system with the `vstorage revoke` command. Then it schedules the relocation of resources registered on the crashed node to other nodes in the cluster by moving files from the `.shaman/md.$host_id/resources` directory to `.shaman/md.$host_id/pool` directories on other cluster nodes, according to the scheduler's decision. Each instance of the shaman service periodically checks the contents of its `.shaman/md.$host_id/pool` directory. If it finds new files there, it runs appropriate scripts from the `/usr/share/shaman` directory to relocate the resource to the node. After the relocation is done, the shaman instance moves the file from its pool directory into its resources directory on the Storage file system.

The default lease timeout can be changed with the `shaman set-config LOCK_TIMEOUT=seconds` command, as described in `man shaman.conf`.

- **What is the purpose of the shaman-monitor watchdog?**

The watchdog checks that its node has an access to the Storage file system. If the Storage file system is not available (e.g., due to network connectivity issues) for `WATCHDOG_TIMEOUT` seconds, the

WATCHDOG_ACTION is executed by the kernel.

The default action is **netfilter**. It blocks incoming and outgoing network packets on a node except for those required by SSH and the network file system.

This helps in situations when the following happens to a node:

- The internal network that handles Storage packets fails.
- The external network (e.g., one with public IP addresses) continues to work.
- The node has not been rebooted yet.
- Shaman has already relocated Storage resources from the node and resumed them on a healthy one.

In such cases, this failed node will continue to send outdated resource reports via its working external network. To avoid this, if node's internal network fails, shaman blocks all of node networks by means of firewall rules.

If the **netfilter** firewall rules cannot be loaded on shaman service start for some reason, the action falls back to **reboot**. The main and the fallback actions are configured with the command `shaman set-config WATCHDOG_ACTION=action[,action...]`, as described in `man shaman.conf`.

The **reboot**, **crash**, and **halt** actions are performed by writing the corresponding value into the `/proc/sysrq-trigger` special file. The **netfilter** action is configured by loading firewall rules from the `/usr/share/shaman/init_wdog` file, as described in `man shaman.conf` and `man shaman-scripts`.

So why is the watchdog needed at all?

When a node cannot access the Storage file system, the following outcomes are possible:

1. The node is isolated from the Storage network. The master shaman instance revoked the node access to Storage and relocated its resources to another node, even though the node itself is not aware of this yet. At the same time:
 - 1.1. The node may also be cut off from the public network that provides outside access to containers and VMs. The watchdog is not needed in this case.
 - 1.2. The node may still be connected to the public network. In this case, a container or VM instance relocated to and started on a healthy cluster node may get the same MAC or IP address as the original instance that may still be running on the failed node even though Storage is unavailable. The watchdog is needed in this case in order to prohibit access to containers and VMs that may be still running on the failed node.

2. The Storage file system is completely unavailable on all nodes. It can happen when the Storage network switch is down, or the majority of Storage MDS instances become unavailable at the same time. Watchdog is needed in this case to prohibit access from the outside to containers and VMs that cannot access the backing storage. If Storage connectivity returns on all nodes at once, the watchdog unfences the nodes and no relocations happen. The reason is that the shaman master is unavailable when the Storage file system is unavailable in the cluster.

So, by default, the watchdog simply fences nodes by enabling the corresponding firewall rules. The reason that nodes reboot often is that `LEASE_LOST_ACTION` and `CLUSTER_MOUNTPOINT_DEAD_ACTION` parameters are set to **reboot** by default. They can be set to **none**, but then one would need to manually clean up each node from which the master had relocated containers and VMs.

- **What does netfilter do? Why reboot hardware nodes instead of restarting shaman-monitor?**

When the **netfilter** action is chosen, the shaman service installs firewall rules with a custom matcher module on its start. The firewall rules are permissive, unless the watchdog is triggered by a timeout. After that, these rules become prohibitive. If the watchdog starts working again (e.g., if the Storage file system becomes accessible), the rules become permissive again.

A reboot is needed to clean up a node after the master shaman instance relocates containers or VMs from it. In this case, remnant ploop devices are left on the node. These ploop devices are created on top of Storage file system entries that are no longer available when the master revokes node's access to Storage. Mounting Storage once again on such a node will not help. One can manually kill all the remnant container processes, detach ploop devices, kill hanged QEMU instances (in VMs case), and so on. Yet some leftovers of container or VM instances that are already running on another cluster node may still remain. A node reboot, however, always results in a clean environment.

- **How does shaman notify the cluster about node's graceful shutdown or reboot?**

Currently, the shaman service does not notify the cluster when a graceful reboot is performed on a node. That is why all resources are left on the node when it is normally rebooted. But if node's shaman service is killed with `SIGKILL` just before the node is going to reboot, the master shaman instance will treat it as a node crash event and relocate resources from this node.

- **Does DRS use SNMP to track resource consumption on each hardware node?**

Yes, it does. An `rmond` plugin for `snmpd` is installed along with the `pdrs` service on each cluster node. The plugin exposes container and VM resource consumption metrics via the SNMP interface. The `snmpd` service provides these metrics along with other standard counters.

The `pdrs` service is started on each cluster node. One of the `pdrs` instances becomes the master in the

same way shaman does. The pdrs master is not coupled with the shaman master, however. These services may be running on different nodes.

Each pdrs instance registers itself in the cluster by creating an entry in the `.drs` directory on the Storage file system. When the master instance discovers a new entry in this directory, it sends an SNMP TRAP subscription request to the `snmpd` instance running on that node. The `snmpd` instance periodically sends SNMP TRAP messages on the UDP port 33333 (set in `.drs/config`) to the pdrs master's node. The master instance receives incoming messages on this port and this way collects statistics from all nodes in the cluster. The pdrs master instance also aggregates and stores the statistics in `.drs/memory.node_list` and `.drs/episode*` files on the Storage file system.

When a node crashes, the master shaman instance gets a list of resources located on the crashed node from the `.shaman/md.$host_id/resources` directory. It then invokes the scheduler defined in the `RESOURCE_RELOCATION_MODE` parameter of the global configuration. By default, this parameter has the `drs` value in the first item of the scheduler list. Shaman then invokes the `/usr/share/shaman/pdrs_schedule` script that connects to the pdrs instance running on the node. The pdrs instance reads the contents of `.drs/memory.node_list` and `.drs/episode*` files on the Storage file system and makes a scheduling decision based on this data. Note that the pdrs instance does not have to be the master to do this. After that, the master shaman instance receives the scheduling decision from the script called earlier and relocates the resources from the crashed node according to the decision.

- **Does DRS get the list of hardware nodes to query via SNMP from shaman?**

The pdrs service requires only one file from shaman, `/etc/shaman/shaman.conf`, that contains the storage cluster name. Shaman, in its turn, only needs to run the `/usr/share/shaman/pdrs_schedule` script if `RESOURCE_RELOCATION_MODE` contains `drs`.

- **Attempting to run scripts mentioned in [Managing Cluster Resources with Scripts](#) produces an error. Are these scripts the basis for correctly evacuating nodes for maintenance purposes? If not, how should that be done, given that graceful shutdown does not automatically relocate containers to a healthy node?**

Most shaman scripts are executed by the master shaman instance when it handles a node crash. For example, you can add a script that will send an alert that a node has crashed via an external web service. These scripts will not help with evacuating a node when preparing it for maintenance. For now, the best way to do it is to manually live-migrate all resources to other cluster nodes and only after that reboot the node.