Virtuozzo Infrastructure Platform 2.5

Administrator's Command Line Guide

December 22, 2018

Virtuozzo International GmbH

Vordergasse 59

8200 Schaffhausen

Switzerland

Tel: + 41 52 632 0411

Fax: + 41 52 672 2010

https://virtuozzo.com

Contents

CHAPTER 1

About This Guide

This guide complements the documentation on managing Virtuozzo Infrastructure Platform via the web-based admin panel.

It is recommended to manage Virtuozzo Infrastructure Platform via the admin panel. If you have it installed, consider the command-line tools secondary and use them with caution.

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

- set custom paths for Virtuozzo Infrastructure Platform 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.

CHAPTER 2

Accessing Storage Clusters via iSCSI

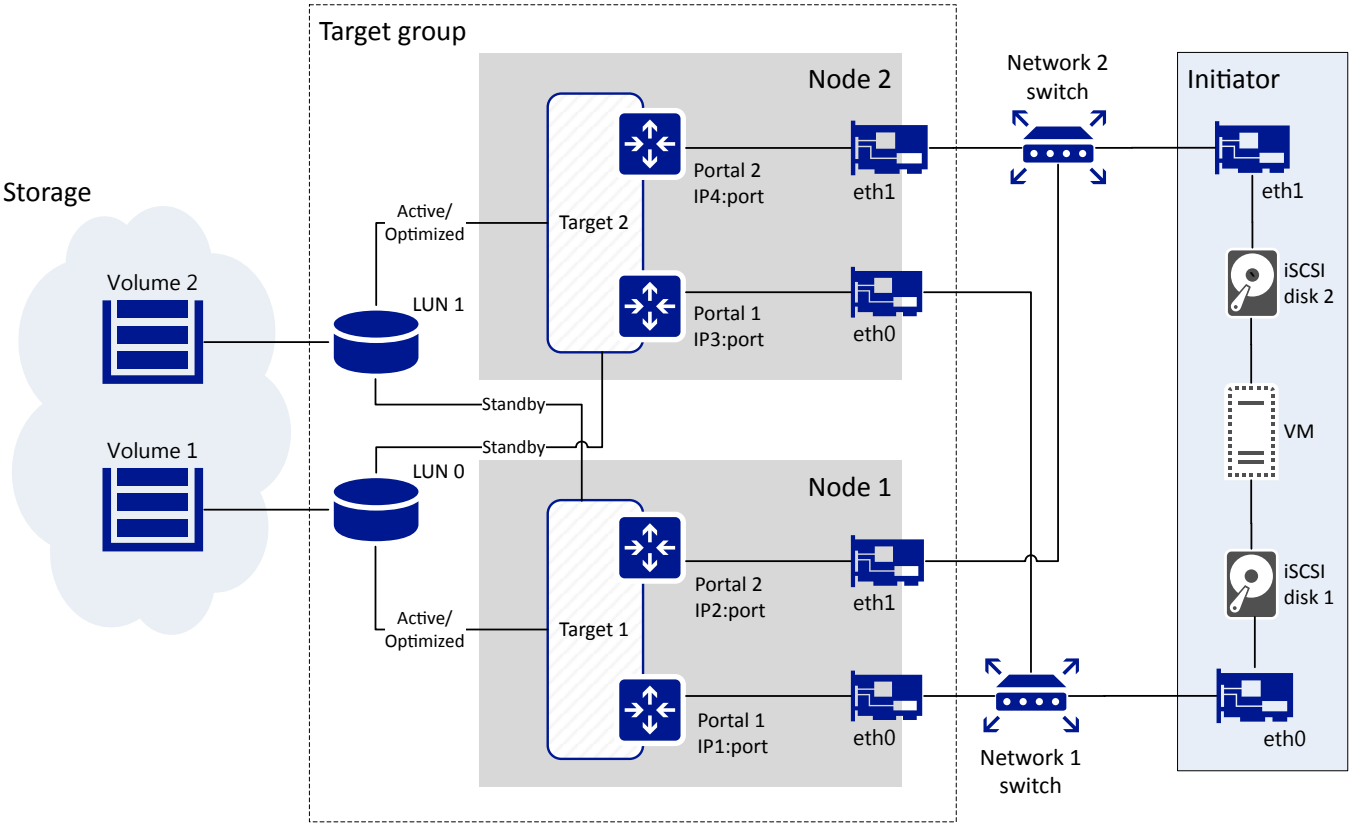
Virtuozzo Infrastructure Platform allows you to export cluster disk space to external operating systems and third-party virtualization solutions in the form of LUN block devices over iSCSI in a SAN-like manner.

In Virtuozzo Infrastructure Platform, you can create groups of redundant targets running on different storage nodes. To each target group you can attach multiple storage volumes with their own redundancy provided by the storage layer. These volumes are exported by targets as LUNs.

Each node in a target group can host a single target for that group if Ethernet is used or one target per FC port if Fibre Channel is used. If one of the nodes in a target group fails along with its target(s), healthy targets from the same group continue to provide access to the LUNs previously serviced by the failed target(s).

You can create multiple target groups on same nodes. A volume, however, may only be attached to one target group at any moment of time.

The figure below shows a typical setup for exporting Virtuozzo Infrastructure Platform disk space via iSCSI.



The figure shows two volumes located on redundant storage provided by Virtuozzo Infrastructure Platform. The volumes are attached as LUNs to a group of two targets running on Virtuozzo Infrastructure Platform nodes. Each target has two portals, one per network interface with the iSCSI traffic type, which makes a total of four discoverable endpoints with different IP addresses. Each target provides access to all LUNs attached to the group. Targets work in the ALUA mode, so one path to the volume is preferred and considered Active/Optimized while the other is Standby. Network interfaces `eth0` and `eth1` on each node are connected to different switches for redundancy. The initiator, e.g., VMware ESXi, is connected to both switches as well and provides volumes as iSCSI disks 1 and 2 to a VM via different network paths. If the Active/Optimized path becomes unavailable for some reason (e.g., the node with the target or network switch fails), the Standby path through the other target will be used instead to connect to the volume. When the Active/Optimized path is restored, it will be used again.

2.1 iSCSI Workflow Overview

The typical workflow of exporting volumes via iSCSI is as follows:

1. Assign the network with the traffic type **iSCSI** to a network interface on each node that you will add to a

target group. See “Managing Traffic Types and Networks” in the *CLI Reference*.

2. Create a target group on chosen nodes, providing details for target WWNs and portals. Targets will be created automatically and added to the group. Target portals will be created on specified network interfaces and ports. See *Creating Target Groups* (page 5).
3. Create volumes and attach them to the target group. See *Managing Volumes* (page 8).
4. Optionally, enable CHAP authorization for the target group, create CHAP accounts, and assign them to the target group. See *Managing CHAP Accounts* (page 14).
5. Optionally, enable ACL authorization for the target group, create a list of initiators that will be allowed to access only specific LUNs. Initiators not on the list will be able to access all LUNs in the target group. See *Managing LUN Views* (page 15).
6. Start the target group. See *Starting and Stopping Target Groups* (page 7).
7. Connect initiators to targets using standard tools of your operating system or product, e.g., `iscsiadm`. Use the `vstorage-target session-list` command to view iSCSI sessions active on a node in a target group.

2.2 Configuring CLI Tool

Before you can use the `vstorage-target` CLI tool to export volumes via iSCSI, set it up as described further. Perform these steps on each node where you plan to run iSCSI targets.

1. Create a configuration file `/etc/vstorage/iscsi/config.json` with at least these mandatory parameters:

```
{
  "ClusterName": "cluster1",
  "VolumesRoot": "/vols/iscsi/vols",
}
```

where `ClusterName` is the name of your storage cluster and `VolumesRoot` is the path to the directory for iSCSI volumes.

You can also set these optional parameters:

- `"PcsLogLevel"`, log level, ranges from 1 (log errors only) to 7 (log all, including debug messages),
- `"LogPath"`, path to log files, default is `"/var/log/vstorage"` (the log will be saved to `vstorage-target.log`),
- `"GetTimeout"`, the timeout for the initiator’s command to read target port group status, default is 3000

ms.

1. Enable the TCM monitor service:

```
# systemctl start vstorage-target-monitor.service
# systemctl enable vstorage-target-monitor.service
```

1. Create the iSCSI volume directory if it does not exist:

```
# mkdir -p /mnt/vstorage/vols/iscsi/
```

If you modify the configuration file later, restart the TCM monitor service to apply changes:

```
# systemctl restart vstorage-target-monitor.service
```

2.3 Managing Target Groups

This section explains how to create and manage groups of iSCSI targets.

2.3.1 Creating Target Groups

Before you create any target groups, assign the network with the iSCSI traffic type to a network interface on each node that you will add to a target group.

To create a target group, you will need a configuration file with a list of nodes to add to the group as well as target WWNs and portals. For example:

```
[
  {
    "NodeId": "01baeabee73e4a0d",
    "WWN": "iqn.2013-10.com.vstorage:test1",
    "Portals": [
      {
        "Addr": "192.168.10.11",
        "Port": 3025
      }
    ]
  },
  {
    "NodeId": "0d90158e9d2444e1",
    "WWN": "iqn.2013-10.com.vstorage:test2",
    "Portals": [
      {
        "Addr": "192.168.10.12",
        "Port": 3025
```

```
      }
    ]
  },
  {
    "NodeId": "a9eca47661a64031",
    "WWN": "iqn.2013-10.com.vstorage:test3",
    "Portals": [
      {
        "Addr": "192.168.10.13",
        "Port": 3025
      }
    ]
  }
]
```

In this configuration file:

- `NodeId` is a node identifier that you can obtain from `/etc/vstorage/host_id` on a node.
- `WWN` is a target world wide name:
  - an IQN if iSCSI protocol is used, e.g., `iqn.2013-10.com.vstorage:test1` (you can only customize the last part after the colon), or
  - a WWPN in NAA format if Fibre Channel protocol is used, e.g., `naa.21000024ff586d3b` (you can obtain the port number from `/sys/class/fc_host/host6/port_name`).
- `Portals` is one or more target portals, IP address and port combinations that the target will be accessible at. The IP address `Addr` is one that belongs to a public network interface on the node that handles the iSCSI traffic type. The port `Port` is optional and defaults to 3260 if omitted.

Once you have the configuration file, e.g., `tg1.json`, you can create the target group with the `vstorage-target tg-create` command. For example, to create an iSCSI target group, run:

```
# vstorage-target tg-create -name tg1 -targets tg1.json -type ISCSI
{
  "Id": "3d8364f5-b830-4211-85af-3a19d30ebac4"
}
```

When you run the command, targets are created on nodes specified in the configuration file and joined to the target group, target portals are created on specified network interfaces and ports.

2.3.2 Starting and Stopping Target Groups

When you create a target group, its targets are initially stopped. You can start them with the `vstorage-target tg-start` command. For example:

```
# vstorage-target tg-start -id 3d8364f5-b830-4211-85af-3a19d30ebac4
```

This command starts all targets in the group with the ID `3d8364f5-b830-4211-85af-3a19d30ebac4`.

All targets in a group can either be running or stopped. So if you add targets to a group of running targets, the new targets will be started automatically.

To stop a target group, use the `vstorage-target tg-stop` command. For example:

```
# vstorage-target tg-stop -id 3d8364f5-b830-4211-85af-3a19d30ebac4
```

2.3.3 Listing Target Groups

You can list target groups with the `vstorage-target tg-list` command that displays basic information about groups. For example:

```
# vstorage-target tg-list
[
  {
    "Id": "3d8364f5-b830-4211-85af-3a19d30ebac4",
    "Name": "tg1",
    "Type": "ISCSI",
    "Running": true,
    "ACL": false,
    "ChapAuth": false,
    "CHAP": {},
    "Mode": 0
  },
  {
    "Id": "78c3b51e-fd9a-485b-91ce-bc0a8171c89d",
    "Name": "tg2",
    "Type": "ISCSI",
    "Running": false,
    "ACL": false,
    "ChapAuth": false,
    "CHAP": {},
    "Mode": 0
  }
]
```

To print complete information about all target groups, use `vstorage-target tg-list -all`.

2.3.4 Printing Details of Target Groups

To print the details of a specific group, use the `vstorage-target tg-status` command. For example:

```
# vstorage-target tg-status -id faeacacd-eba6-416c-9a7f-b5ba9e372e16
```

This command prints the complete details of the target group with the ID `faeacacd-eba6-416c-9a7f-b5ba9e372e16`. One parameter to pay attention to is `NodeState`. It indicates whether a node is in sync with the target group, i.e. aware of its current configuration. The following states can be shown:

- `synced`, node is in sync with the target group,
- `syncing`, node is syncing with the target group,
- `failed`, node failed to sync with the target group (see the `Error` parameter for details),
- `offline`, node is offline,
- `disabled`, node is disabled and its target is offline.

2.3.5 Deleting Target Groups

To delete a target group, use the `vstorage-target tg-delete` command. For example:

```
# vstorage-target tg-delete -id 3d8364f5-b830-4211-85af-3a19d30ebac4
```

This command deletes the target group with the ID `3d8364f5-b830-4211-85af-3a19d30ebac4`.

2.4 Managing Volumes

This section describes how to create and manage volumes to be exported via iSCSI.

2.4.1 Creating Volumes

To create a volume, use the `vstorage-target vol-create` command. For example:

```
# vstorage-target vol-create -name vol1 -size 1G \
-vstorage-attr "replicas=3:2 failure-domain=host tier=0"
{
```

```
  "Id": "3277153b-5296-49c5-9b66-4c200ddb343d"
}
```

This command creates a 1 GB volume named `vol1` on storage tier 0 with 3:2 replication and host as failure domain.

2.4.2 Listing and Printing Details of Volumes

To list volumes, use the `vstorage-target vol-list` command. For example:

```
# vstorage-target vol-list
[
  "3277153b-5296-49c5-9b66-4c200ddb343d",
  "a12110d5-cbbc-498a-acdd-a8567286f927",
  "d5cc3c13-cfb4-4890-a20d-fb80e2a56278"
]
```

Add `-all` to print details of all volumes. To print details of a specific volume, run `vstorage-target vol-stat -id <vol_ID>`.

2.4.3 Attaching Volumes to Target Groups

To attach a volume to a target group, use the `vstorage-target tg-attach` command. A volume cannot be attached to multiple target groups at the same time. For example:

```
# vstorage-target tg-attach -id 3d8364f5-b830-4211-85af-3a19d30ebac4 \
-volume 3277153b-5296-49c5-9b66-4c200ddb343d -lun 0
```

This command attaches the volume with the ID `3277153b-5296-49c5-9b66-4c200ddb343d` to a target group with the ID `3d8364f5-b830-4211-85af-3a19d30ebac4` as LUN 0. LUN ID numbering must start with 0.

2.4.4 Viewing and Setting Volume Parameters

To view and set volume parameters, e.g. redundancy mode, failure domain, or tier, use the commands `vstorage-target vol-attr get` and `vstorage-target vol-attr set`, respectively. For example:

```
# vstorage-target vol-attr get -id d5cc3c13-cfb4-4890-a20d-fb80e2a56278
{
  "chunk-size": "268435456",
  "client-ssd-cache": "1",
  "failure-domain": "host",
```

```
  "replicas": "3:2",
  "tier": "0"
}
# vstorage-target vol-attr set -id d5cc3c13-cfb4-4890-a20d-fb80e2a56278 \
-vstorage-attr "replicas=2:1 tier=1"
```

2.4.5 Increasing Volume Size

To increase the size of a volume, use the `vstorage-target vol-grow` command. For example:

```
# vstorage-target vol-grow -id d5cc3c13-cfb4-4890-a20d-fb80e2a56278 -size 2G
```

2.4.6 Setting Volume Limits

To set read/write limits for a volume, use the `vstorage-target vol-limits` command. For example:

```
# vstorage-target vol-limits -id d5cc3c13-cfb4-4890-a20d-fb80e2a56278 -read-bps 10485760 \
-write-bps 10485760
```

This command sets read/write speed for the volume with the ID `d5cc3c13-cfb4-4890-a20d-fb80e2a56278` to 10485760 bytes per second.

2.4.7 Detaching Volumes from Target Groups

To detach a volume from a target group, use the `vstorage-target tg-detach` command. LUN 0 must be detached last. For example:

```
# vstorage-target tg-detach -id 3d8364f5-b830-4211-85af-3a19d30ebac4 \
-volume d5cc3c13-cfb4-4890-a20d-fb80e2a56278
```

This command detaches the volume with the ID `d5cc3c13-cfb4-4890-a20d-fb80e2a56278` from the target group with the ID `3d8364f5-b830-4211-85af-3a19d30ebac4`.

2.4.8 Deleting Volumes

To delete a volume, use the `vstorage-target vol-delete` command. You cannot delete volumes attached to target groups. For example:

```
# vstorage-target vol-delete -id d5cc3c13-cfb4-4890-a20d-fb80e2a56278
```

This command deletes the volume with the ID `d5cc3c13-cfb4-4890-a20d-fb80e2a56278`.

2.5 Managing Nodes

This section describes how to manage nodes in relation to target groups.

2.5.1 Adding Nodes to Target Groups

To add a node to a target group, create a configuration file with details about target WWN and portal. The target will be created automatically on the added node. One node can be added to multiple target groups and the same network interfaces on it can be used simultaneously by multiple targets from different groups.

For example:

```
# vstorage-target node-add -node bbfd0e7a26b1406d -tg 3d8364f5-b830-4211-85af-3a19d30ebac4 \
-targets target.json
```

This command adds the node with the ID `bbfd0e7a26b1406d` to the target group with the ID `3d8364f5-b830-4211-85af-3a19d30ebac4` and creates a target on it according to the `target.json` configuration file that looks as follows:

```
[
  {
    "NodeId": "bbfd0e7a26b1406d",
    "WWN": "iqn.2013-10.com.vstorage:test2",
    "Portals": [
      {
        "Addr": "10.94.104.89",
        "Port": 3260
      }
    ]
  }
]
```

2.5.2 Setting Node Status

To enable or disable a node in a specific target group or all target groups at once, use the `vstorage-target node-set` command. Enabling a node starts its targets while disabling a node stops its targets and moves the `PREFERRED` bit to another node.

For example, to enable a node with the ID `bbfd0e7a26b1406d` in all target groups it belongs to, run

```
# vstorage-target node-set -tg any -node bbfd0e7a26b1406d -enable
```

To disable a node with the ID `bbfd0e7a26b1406d` in the target group with the ID `3d8364f5-b830-4211-85af-3a19d30ebac4`, allowing 60 seconds to move the preferred path to another node (i.e. target), run

```
# vstorage-target node-set -tg 3d8364f5-b830-4211-85af-3a19d30ebac4 -node bbfd0e7a26b1406d \
-disable -release-timeout 60
```

The `release-timeout` parameter sets time in seconds that the initiator has to move the preferred (Active/Optimized) path following the `PREFERRED` bit. If the initiator fails to do so within the given time, the disable operation is cancelled and the node remains enabled. The `PREFERRED` bit, however, is still moved to another node.

The `-force` parameter stops the target(s) on the node at once without moving the `PREFERRED` bit.

2.5.3 Deleting Nodes from Target Groups

To delete a node from a target group, use the `vstorage-target node-del` command. You can delete nodes only after disabling them in specified target groups. Deleting a node also deletes the targets on that node. For example:

```
# vstorage-target node-del -tg 3d8364f5-b830-4211-85af-3a19d30ebac4 -node bbfd0e7a26b1406d
```

This command deletes the node with the ID `bbfd0e7a26b1406d` from the target group with the ID `3d8364f5-b830-4211-85af-3a19d30ebac4`. The node is already disabled in the target group (see *Setting Node Status* (page 12)).

2.6 Managing Targets and Portals

This section describes how to create and manage targets.

The optimal way is to create a single target per node if you use the iSCSI protocol and one target per FC port if you use the FC protocol.

2.6.1 Creating Targets

Typically, targets are created automatically when you create target groups or add nodes to them. However, as you can delete target(s) from a node without removing the node from a target group, you can also create target(s) on such a node again. Use the `vstorage-target target-create` command. For example:

```
# vstorage-target target-create -tg 3d8364f5-b830-4211-85af-3a19d30ebac4 -json target.json
```

This command creates a target based on the `target.json` configuration file in the target group with the ID `3d8364f5-b830-4211-85af-3a19d30ebac4`. The configuration file lists target details like the node to create the target on, WWN, and portal. For example:

```
{
  "NodeId": "bbfd0e7a26b1406d",
  "WWN": "iqn.2013-10.com.vstorage:test22",
  "Portals": [
    {
      "Addr": "10.94.104.90",
      "Port": 3260
    }
  ]
}
```

2.6.2 Adding and Removing Target Portals

To add a portal to a target, use the `vstorage-target target-portal add` command. For example:

```
# vstorage-target target-portal add -wwn iqn.2013-10.com.vstorage:test2 -addr 10.94.104.90 \
-tg 3d8364f5-b830-4211-85af-3a19d30ebac4
```

This command adds a portal with the IP address 10.94.104.90 and default port 3260 to the target with the IQN `iqn.2013-10.com.vstorage:test2` in the target group with the ID `3d8364f5-b830-4211-85af-3a19d30ebac4`.

To delete a portal from a target, use the `vstorage-target target-portal del` command. For example:

```
# vstorage-target target-portal del -wwn iqn.2013-10.com.vstorage:test2 -addr 10.94.104.90 \
-tg 3d8364f5-b830-4211-85af-3a19d30ebac4
```

This command deletes the portal created before.

2.6.3 Deleting Targets

To delete a target from a target group (as well as the node it is on), use the `vstorage-target target-delete` command. For example:

```
# vstorage-target target-delete -tg 3d8364f5-b830-4211-85af-3a19d30ebac4 \
-wwn iqn.2013-10.com.vstorage:test22
```

This command deletes the target with the IQN `iqn.2013-10.com.vstorage:test22` from the target group with the ID `3d8364f5-b830-4211-85af-3a19d30ebac4` and from the node it is located on.

Nodes that have no targets left on them remain in target groups.

2.7 Managing CHAP Accounts

The Challenge-Handshake Authentication Protocol (CHAP) provides a way to restrict access to targets and their LUNs by requiring a user name and a password from the initiator. CHAP accounts apply to entire target groups. Fibre Channel target groups do not use CHAP. .. see also in admin guide.

To use CHAP, enable it for the target group:

```
# vstorage-target tg-auth -enable-chap -id <tg_ID>
```

2.7.1 Creating and Listing CHAP Accounts

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

```
# vstorage-target account-create -user user1 -desc "User for TG1"
Enter Password:
```

The password must be 12 to 16 characters long.

To list existing CHAP accounts and their details, use the `vstorage-target account-list` command.

2.7.2 Changing CHAP Account Details

To change the password or description of a CHAP account, use the `vstorage-target account-set` command. For example:

```
# vstorage-target account-set description -user user1 -desc "A new description"
# vstorage-target account-set password -user user1
Enter Password:
```

2.7.3 Assigning CHAP Accounts to Target Groups

To assign a CHAP account to a target group, use the `vstorage-target tg-chap` command. For example:

```
# vstorage-target tg-chap set -id faeacacd-eba6-416c-9a7f-b5ba9e372e16 -user user1
```

To remove an assignment, run

```
# vstorage-target tg-chap del -id faeacacd-eba6-416c-9a7f-b5ba9e372e16 -user user1
```

2.7.4 Deleting CHAP Accounts

To delete an unused CHAP account, use the `vstorage-target account-delete` command. For example:

```
# vstorage-target account-delete -user user1
```

2.8 Managing LUN Views

LUN views provide a way to create and manage an access control list (ACL) that limits access to chosen LUNs for specific initiators. Initiators not on the list have access to all LUNs in iSCSI target groups. Volumes exported via Fibre Channel target groups, however, can only be accessed by initiators that are added to group ACL. .. see also in admin guide.

To use ACL-based authorization, enable it for the target group:

```
# vstorage-target tg-auth -enable-acl -id <tg_ID>
```

2.8.1 Creating LUN Views

To create a LUN view for an initiator, use the commands `vstorage-target tg-initiator add` or `vstorage-target view-add`. The former command adds an initiator to target group's ACL and creates a view for it. The latter command is used to add views to initiators that are already on the ACL.

For example:

```
# vstorage-target tg-initiator add -alias initiator2 -luns 0,1 \
-tg ee764519-80e3-406e-b637-8d63712badf1 -wwn iqn.1994-05.com.redhat:1535946874d
```

This command adds the initiator with the IQN `iqn.1994-05.com.redhat:1535946874d` to the ACL of the target group with the ID `ee764519-80e3-406e-b637-8d63712badf1` and creates a view allowing it to access the LUNs with the IDs `0` and `1`.

Another example:

```
# vstorage-target view-add -tg faeacacd-eba6-416c-9a7f-b5ba9e372e16 -lun 2 -map 2 \
-wwn iqn.1994-05.com.redhat:1535946874d
```

This command adds a view for the same initiator allowing it to access LUN 2 as well.

2.8.2 Listing LUN Views

To list LUN views for an initiator, use the `vstorage-target view-list` command. For example:

```
# vstorage-target view-list -tg ee764519-80e3-406e-b637-8d63712badf1 \
-wwn iqn.1994-05.com.redhat:1535946874d
```

This command lists views for the initiator with the IQN `iqn.1994-05.com.redhat:1535946874d`.

2.8.3 Changing LUN View Details

To change LUN views for an initiator, use the `vstorage-target view-set` command. For example:

```
# vstorage-target view-set -luns 1 -tg ee764519-80e3-406e-b637-8d63712badf1 \
-wwn iqn.1994-05.com.redhat:1535946874d
```

This command allows the initiator with the IQN `iqn.1994-05.com.redhat:1535946874d` to access only LUN 1. Essentially, it deletes all LUN views for it excluding specified.

2.8.4 Deleting LUN Views

To delete a LUN view for an initiator, use the `vstorage-target view-del` command.

```
# vstorage-target view-del -lun 1 -tg ee764519-80e3-406e-b637-8d63712badf1 \
-wwn iqn.1994-05.com.redhat:1535946874d
```

This command deletes the view for LUN 1 for the initiator with the IQN `iqn.1994-05.com.redhat:1535946874d`.

CHAPTER 3

Accessing Storage Clusters via S3 Protocol

Virtuozzo Infrastructure Platform can export data via an Amazon S3-like API, enabling service providers to:

- run S3-based services in their Virtuozzo Infrastructure Platform deployments,
- sell S3-based storage-as-a-service to customers along with Virtuozzo Infrastructure Platform.

The support for S3 expands the functionality of Virtuozzo Infrastructure Platform and requires a working storage cluster.

3.1 About Object Storage

Object storage is a storage architecture that enables managing data as objects (like in a key-value storage) as opposed to files in file systems or blocks in a block storage. Except for the data, each object has metadata that describes it as well as a unique identifier that allows finding the object in the storage. Object storage is optimized for storing billions of objects, in particular for application 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.

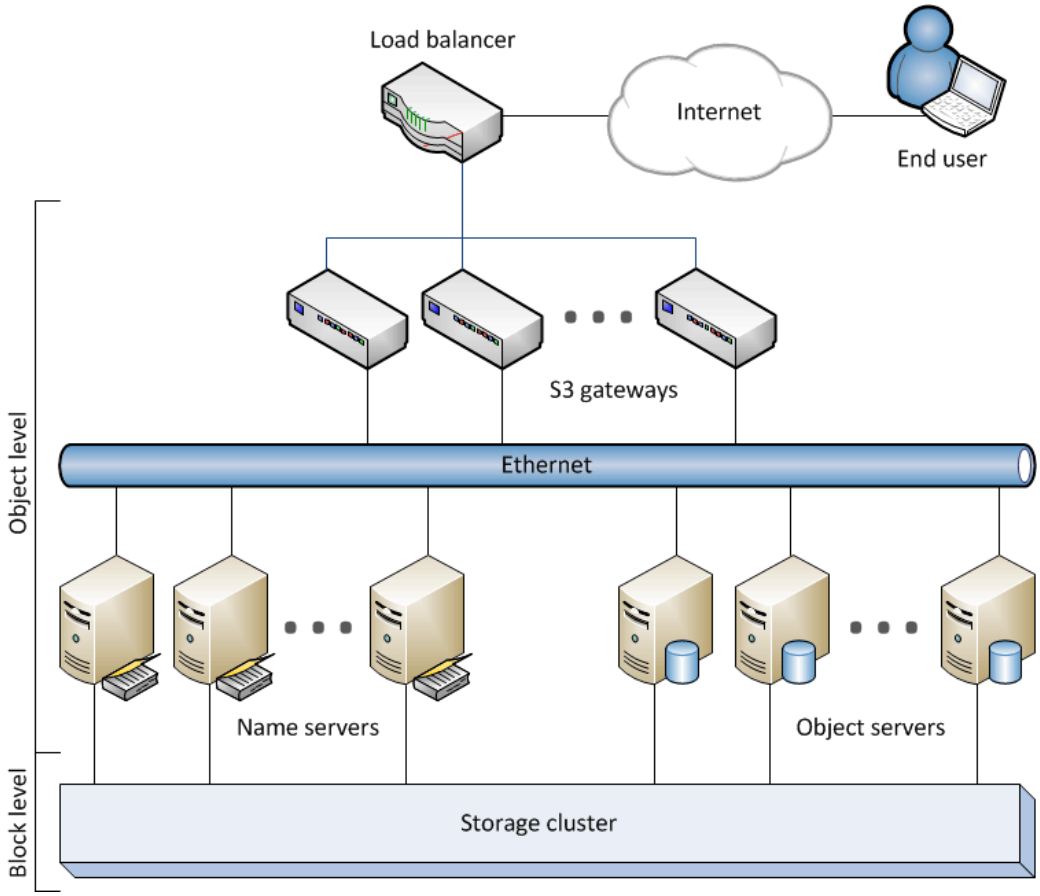
Compared to other types of storage, the key difference of object storage is that parts of an object cannot be modified, so if the object changes a new version of it is spawned instead. This approach is extremely important for maintaining data availability and consistency. First of all, changing an object 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.

3.1.1 Object Storage Infrastructure Overview

The object storage infrastructure consists of the following entities: object servers (OS), name servers (NS), S3 gateways (GW), and the block-level backend.

These entities run as services on the Virtuozzo Infrastructure Platform nodes. Each service should be deployed on multiple Virtuozzo Infrastructure Platform nodes for high availability.



- An object server stores actual object data received from S3 gateway. The data is packed into special containers to achieve high performance. The containers are redundant, you can specify the redundancy mode while configuring object storage. An object server also stores its own data in block storage with built-in high availability.
- A 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.
- An S3 gateway is a data proxy between object storage services and end users. It receives and handles Amazon S3 protocol requests and S3 user authentication and ACL checks. The S3 gateway uses the NGINX web server for external connections and has no data of its own (i.e. is stateless).
- The block-level backend is block storage with high availability of services and data. Since all object storage services run on hosts, no virtual environments (and hence licenses) are required for object storage.

For more information, see *Object Storage Components* (page 21).

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.

3.1.2.1 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.

3.1.2.2 Object Storage Interaction with a Storage Cluster

An S3 storage cluster requires a working storage cluster on each of S3 cluster nodes. Virtuozzo Infrastructure Platform 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* (page 22)) that the Virtuozzo Infrastructure Platform file system layer (vstorage-mount) does not interpret in any way.

3.1.3 Object Storage Components

This section offers more detail on S3 storage components: gateways, object servers, and name servers; and describes S3 management tools and service buckets.

3.1.3.1 Gateway

A 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).

3.1.3.2 Name Server

A 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. Virtuozzo Infrastructure Platform).

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.

3.1.3.3 Object Server

An 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.

3.1.3.4 S3 Management Tools

Object storage has two tools:

- `ostor` for configuring storage components, and
- `s3-ostor-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).

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 `s3-ostor-admin` tool). The information corresponds to the following names in the object storage:

- Names with a `/u/` prefix. Correspond to user data (user identifier, e-mail, access key ID, and secret access key).
- Names with an `/m/` prefix. Correspond to temporary information on current multipart uploads and their parts.
- Names with a `/tmp/` prefix. Correspond to information on operations that consist of several atomic alterations of objects in the storage. These names are necessary in case the operation fails.

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.

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.

3.1.5 Operations on Objects

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

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.

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.

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.

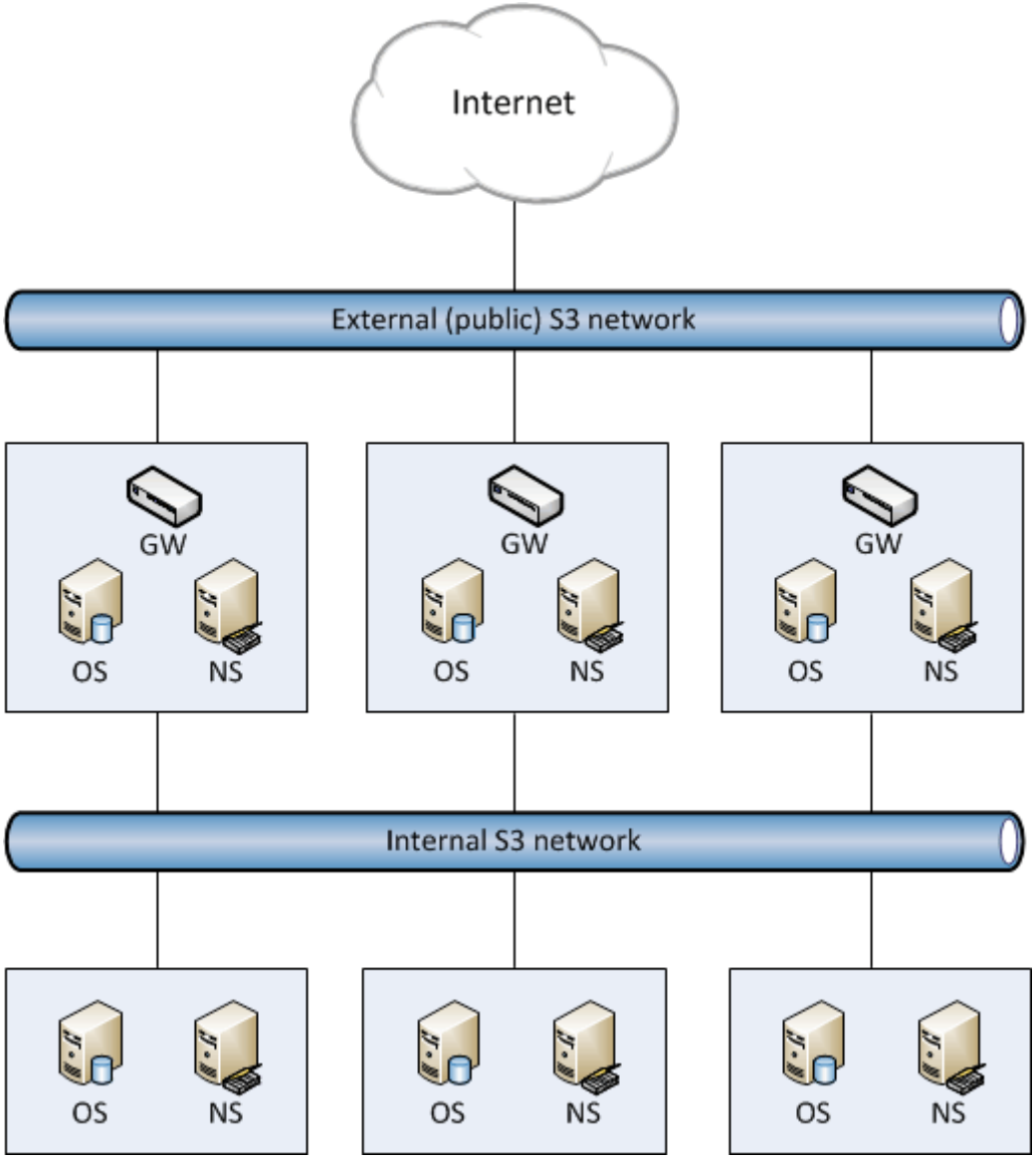
To fulfill the request, object server transmits fixed-size messages with the object's data to the gateway. The last message contains an end-of-data flag.

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.

3.2 Deploying Object Storage

This chapter describes how to deploy object storage on top of a ready storage cluster and create a setup like the one shown on the figure below. 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 S3 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 Virtuozzo Infrastructure Platform. If, however, you expect that object storage traffic will compete with Virtuozzo Infrastructure Platform 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 Virtuozzo Infrastructure Platform (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 -n <IP_addr> \
-c "vstorage://<cluster_name>/<ostor_dir> <NS_num> <OS_num>"
```

where

- <IP_addr> is the node's IP address that object storage will go through,
- <cluster_name> is the name of your storage cluster,
- <ostor_dir> is the directory in the cluster with object storage service files,
- <NS_num>, <OS_num> are the numbers of NS and OS, respectively.

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.

5. Launch the configuration service.

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

6. Initialize 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.

7. 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* (page 38).

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/* 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.

8. Add nodes where object storage services will run to the configuration. To do this run the `ostor-ctl add-host` command on every such node:

```
# ostor-ctl add-host -H <internal_IP_address> -r <cluster_mount_point>/<ostor_dir>
```

This command will automatically detect and use the node's hostname and have the object storage agent service listen on an internal IP address. You will need to provide the object storage password set on step 3.

9. Create S3 gateway instances on chosen nodes with Internet access and external IP addresses.

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>
```

where `<internal_IP_address>` is the internal IP address of the node with the gateway.

Note: Port number is mandatory.

---

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

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

11. 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* (page 32).

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            ADDRS
S3GW     8000000000000009    ACTIVE             527  fcbf5602197245da   127.0.0.1:9090
S3GW     8000000000000008    ACTIVE             536  4f0038db65274507   127.0.0.1:9090
S3GW     8000000000000007    ACTIVE             572  958e982fcc794e58   127.0.0.1:9090
OS       1000000000000005    ACTIVE             452  4f0038db65274507   10.30.29.124:39746
OS       1000000000000004    ACTIVE             647  fcbf5602197245da   10.30.27.69:56363
OS       1000000000000003    ACTIVE             452  4f0038db65274507   10.30.29.124:52831
NS       0800000000000002    ACTIVE             647  fcbf5602197245da   10.30.27.69:56463
NS       0800000000000001    ACTIVE             452  4f0038db65274507   10.30.29.124:53044
NS       0800000000000000    ACTIVE             647  fcbf5602197245da   10.30.27.69:37876
```

12. Install one NGINX web server per each S3 endpoint you need. On nodes where you install `nginx`, replace the contents of its configuration file `/etc/nginx/conf.d/nginx.conf` with the following (replace the IP addresses as required):

```
upstream s3 {
        server 127.0.0.1:9000; #S3 gateway 1 internal IP address
        server 127.0.0.2:9000; #S3 gateway 2 internal IP address
        server 127.0.0.3:9000; #S3 gateway 3 internal IP address
# Optional load balancing parameters (see
# http://nginx.org/en/docs/http/load_balancing.html)
}
server {
    listen       80;
    server_name  172.0.0.1; #S3 endpoint. If you have DNS configured,
    #replace the IP address with the corresponding hostname.
    client_max_body_size 5g;
    #charset koi8-r;
    #access_log  /var/log/nginx/log/host.access.log main;
    location / {
        fastcgi_pass_header Connection-close;
        fastcgi_pass s3;
        fastcgi_no_cache 1;
        include fastcgi_params;
        fastcgi_request_buffering off;
        fastcgi_max_temp_file_size 0;
}
```

13. Launch NGINX:

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

The object storage is deployed. Now you can add S3 users with the `ostor-s3-admin` tool. For example:

```
# ostor-s3-admin create-user -e user@email.com
Created user: email=user@email.com,user id=81d406fa613ad6c1
Key pair[0]: access key id=81d406fa613ad6c1S8HL,
secret access key=ya8iq3yrEYEhpErCkSmui6ifBghDDLdN2vso3sJn
```

The access key ID and secret access key pair, along with S3 endpoint, are required to connect to object

storage from a client application.

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
```

3.2.1 Manually Binding Services to Nodes

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 `800000000000001`, `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
0800000000000001     NS  vstorage://stor1/s3-data/services/0800000000000001
0800000000000002     NS  vstorage://stor2/s3-data/services/0800000000000002
1000000000000003     OS  vstorage://stor1/s3-data/services/1000000000000003
1000000000000004     OS  vstorage://stor2/s3-data/services/1000000000000004
1000000000000005     OS  vstorage://stor1/s3-data/services/1000000000000005
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
```

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). The Amazon S3 protocol uses a permission 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 the entity owner assigned to every object and bucket at creation. The entity owner has extra rights compared to other users. For example, the bucket owner is the only one who can delete that bucket.

User model and access policies implemented in Virtuozzo Infrastructure Platform comply with the Amazon S3 user model and access policies.

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

You can manage users with the `ostor-s3-admin` tool.

To do it via CLI, you will need to know the ID of the volume that they 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
...
```

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

3.3.1 Adding 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:HSDu2DA00JNGjnRcAhLKfhrvlymzOVdLPsCK2dcq
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.

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><Ema
il>user@email.com</Email><Keys><OwnerId>0000000000000000</OwnerId><KeyPair><S3AccessKeyId>a49e12a2
26bd760fGHQ7</S3AccessKeyId><S3SecretAccessKey>HSDu2DA00JNGjnRcAhLKfhrvlymzOVdLPsCK2dcq</S3SecretA
ccessKey></KeyPair></Keys></User><User><Id>d7c53fc1f931661f</Id><Email>user@email.com</Email><Keys
><OwnerId>0000000000000000</OwnerId><KeyPair><S3AccessKeyId>d7c53fc1f931661fZLIV</S3AccessKeyId><S
3SecretAccessKey>JL7gt1OH873zR0Fzv8Oh9ZuA6JtCVnkgV7lET6ET</S3SecretAccessKey></KeyPair></Keys></Us
er></Users></QueryUsersResult>
```

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=83tTsNAuuRyoBBqhxMFqHAC60dhKHtTCCkQe54zu
```

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
```

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
```

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=83tTsNAuuRyoBBqhxMFqHAC60dhKHtTCCkQe54zu
```

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

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
```

3.4 Managing S3 Buckets

All objects in Amazon S3-like storage are stored in containers called “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.

To do it via CLI, you will need to know the ID of the volume that they 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
...
```

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

3.4.1 Listing S3 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` or `mybucket.mys3storage.example.com` (depending on DNS configuration).

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

3.4.2 Listing S3 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><Own
er>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</Ve
rsioning><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
```

3.4.3 Querying S3 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
```

3.4.4 Changing S3 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
```

3.4.5 Deleting S3 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
Deleted bucket bucket1
```

3.5 Best Practices for Using Object Storage

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

3.5.1 Bucket and Key Naming Policies

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

- can be from 3 to 63 characters long,
- must start and end with a lowercase letter or number,
- can contain 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.

3.5.2 Improving Performance of PUT Operations

Object storage supports uploading objects as large as 5 GB per single PUT request (5 TB via multipart upload). Upload performance can be improved by splitting large objects into pieces and uploading them concurrently (thus dividing the load between multiple OS services) with multipart upload API.

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

3.6 Supported Amazon S3 Features

This section lists Amazon S3 operations, headers, and authentication schemes supported by the Virtuozzo Infrastructure Platform implementation of the Amazon S3 protocol.

3.6.1 Supported Amazon S3 REST Operations

The following Amazon S3 REST operations are currently supported by the Virtuozzo Infrastructure Platform implementation of the Amazon S3 protocol:

Supported service operations: GET Service.

Supported bucket operations:

- DELETE/HEAD/PUT Bucket
- GET Bucket (List Objects; only version 1)
- GET/PUT Bucket acl
- GET Bucket location (returns `US East`)
- GET Bucket Object versions
- GET/PUT Bucket versioning
- List Multipart Uploads

Supported object operations:

- DELETE/GET/HEAD/POST/PUT Object
- Delete Multiple Objects

- PUT Object - Copy
- GET/PUT Object acl
- Delete Multiple Objects
- Abort Multipart Upload
- Complete Multipart Upload
- Initiate Multipart Upload
- List Parts
- Upload Part

Note: For more information on Amazon S3 REST operations, see Amazon S3 REST API documentation.

3.6.2 Supported Amazon Request Headers

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

- Authorization
- Content-Length
- Content-Type
- Content-MD5
- Date
- Host
- x-amz-content-sha256
- x-amz-date
- x-amz-security-token

The following Amazon S3 REST request headers are ignored:

- Expect

- x-amz-security-token

Note: For more information on Amazon S3 REST request headers, see the Amazon S3 REST API documentation.

3.6.3 Supported Amazon Response Headers

The following Amazon S3 REST response headers are currently supported by the Virtuozzo Infrastructure Platform implementation of the Amazon S3 protocol:

- Content-Length
- Content-Type
- Connection
- Date
- ETag
- x-amz-delete-marker
- x-amz-request-id
- x-amz-version-id

The following Amazon S3 REST response headers are not used:

- Server
- x-amz-id-2

Note: For more information on Amazon S3 REST response headers, see the Amazon S3 REST API documentation.

3.6.4 Supported Amazon Error Response Headers

The following Amazon S3 REST error response headers are currently supported by the Virtuozzo Infrastructure Platform implementation of the Amazon S3 protocol:

- Code
- Error
- Message
- RequestId
- Resource

The following Amazon S3 REST error response headers are not supported:

- RequestId (not used)
- Resource

Note: For more information on Amazon S3 REST response headers, see the Amazon S3 REST API documentation.

3.6.5 Supported Authentication Scheme and Methods

The following authentication scheme is supported by the Virtuozzo Infrastructure Platform implementation of the Amazon S3 protocol:

- Signature Version 2.
- Signature Version 4.

The following authentication methods is supported by the Virtuozzo Infrastructure Platform implementation of the Amazon S3 protocol:

- HTTP Authorization header.
- Query string parameters.

CHAPTER 4

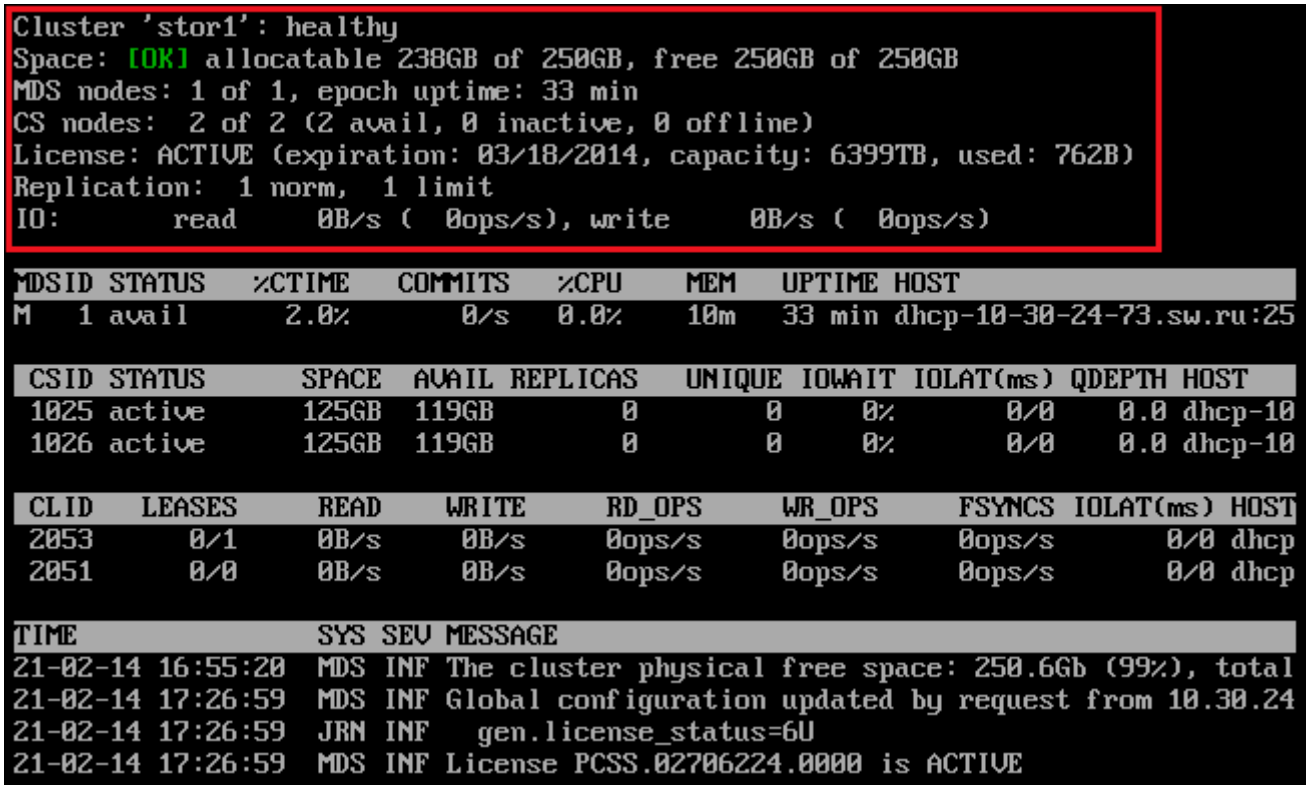
Monitoring Storage Cluster

Monitoring the 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.

The main command for monitoring is `vstorage -c <cluster_name> top`. It invokes a text user interface that you can control with keys (press **h** for help).

4.1 Monitoring General Storage Cluster Parameters

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



The command above shows detailed information about the `stor1` cluster. The general parameters (highlighted in red) are as follows.

Cluster Overall status of the cluster:

- **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 *Monitoring Physical Disks* (page 54).

Space Amount of disk space in the cluster:

- **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.

Note: For more information on monitoring and understanding disk space usage in clusters, see *Understanding Disk Space Usage* (page 48).

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 Number of active chunk servers as compared to the total number of chunk servers configured for the cluster.

The information in parentheses informs you about the number of these chunk servers:

- 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.

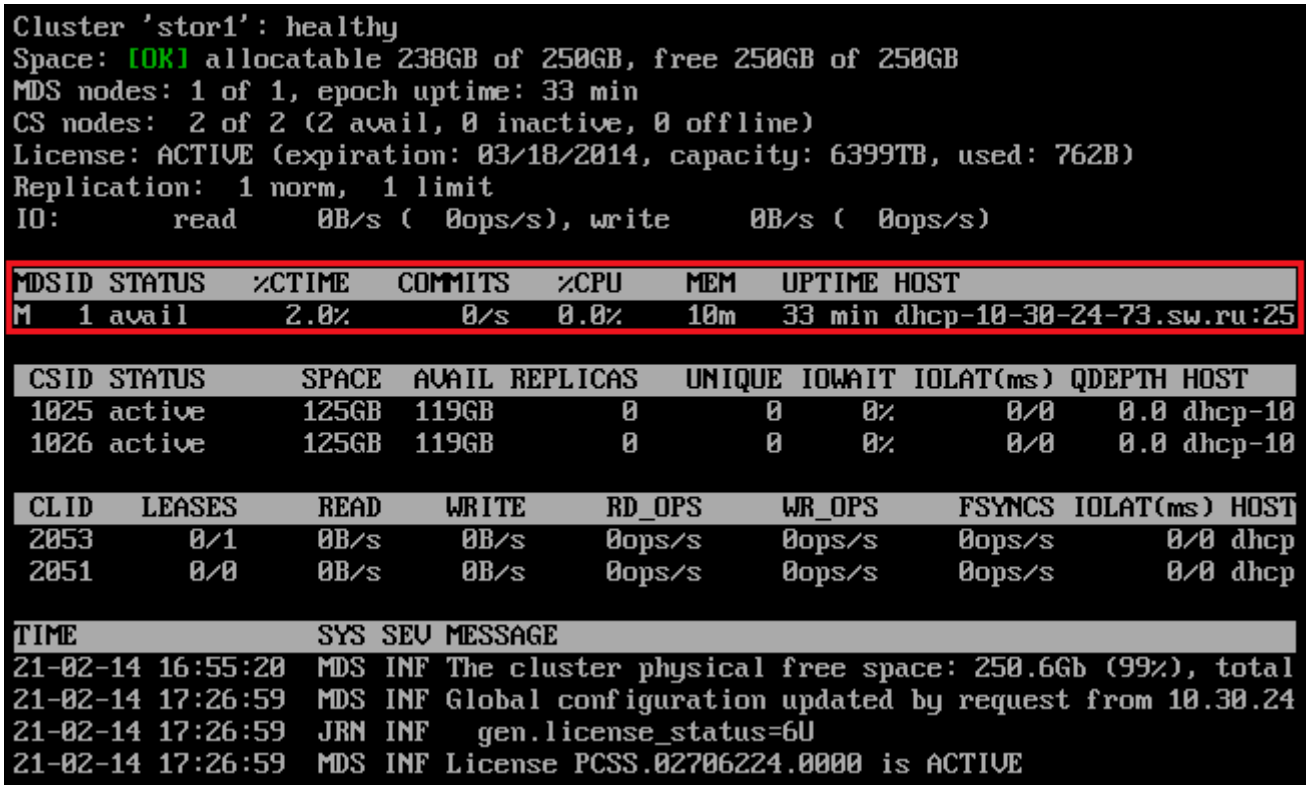
Replication Replication settings. The normal number of chunk replicas and the limit after which a chunk gets blocked until recovered.

IO Disks IO activity in the cluster:

- Speed of read and write I/O operations, in bytes per second.
- Number of read and write I/O operations per second.

4.2 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:



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

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.

COMMITS Local journal commit rate.

%CPU MDS server activity time.

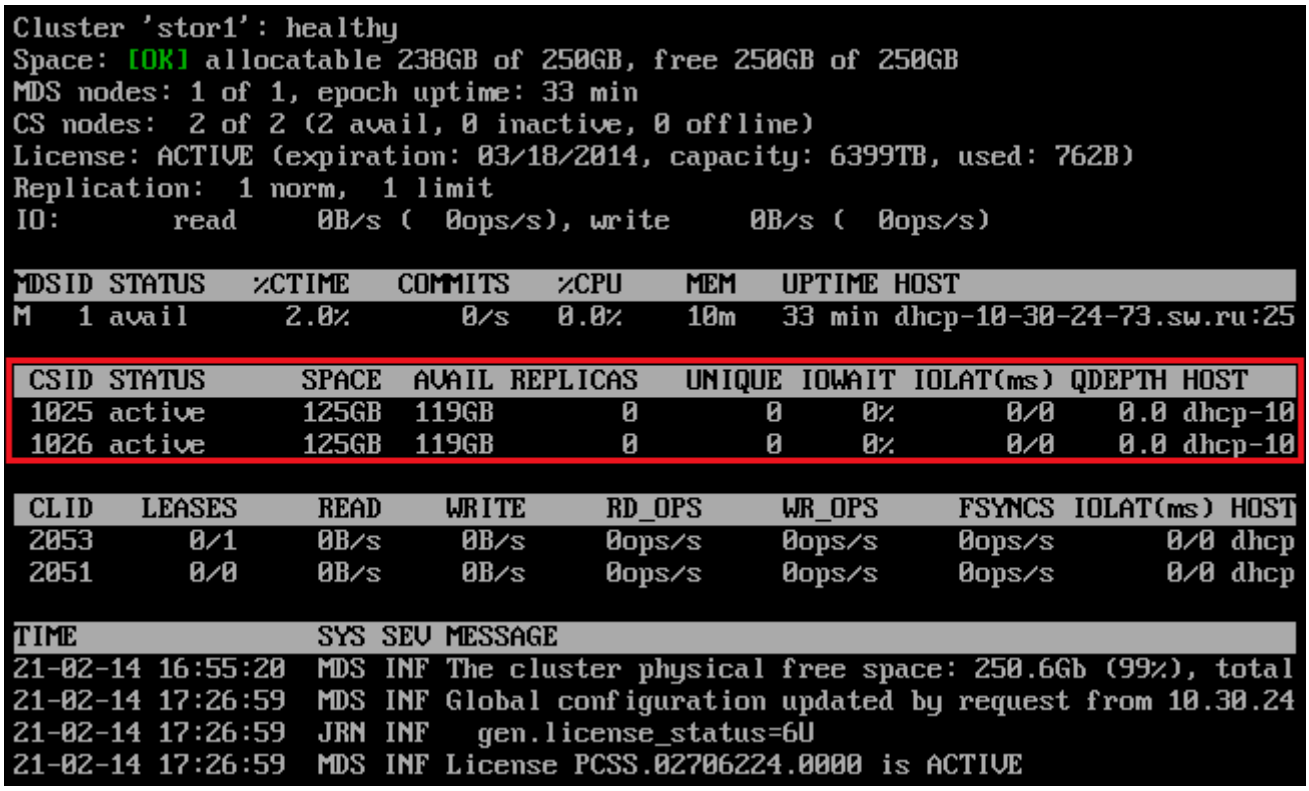
MEM Amount of physical memory the MDS server uses.

UPTIME Time elapsed since the last MDS server start.

HOST MDS server hostname or IP address.

4.3 Monitoring Chunk Servers

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



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

CSID Chunk server identifier (ID).

STATUS Chunk server status:

- **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 serve 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 The following flags may be shown for active chunk servers:

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.

4.3.1 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
...
```

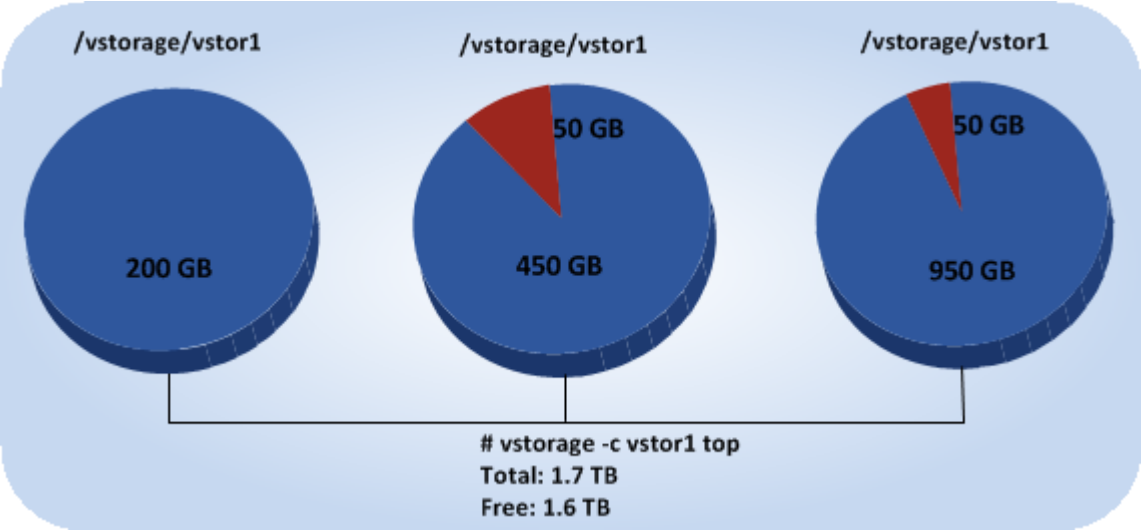
In this command output:

- 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 used for storing data chunks. See **Understanding allocatable disk space** below for details.

4.3.1.1 Understanding Allocatable Disk Space

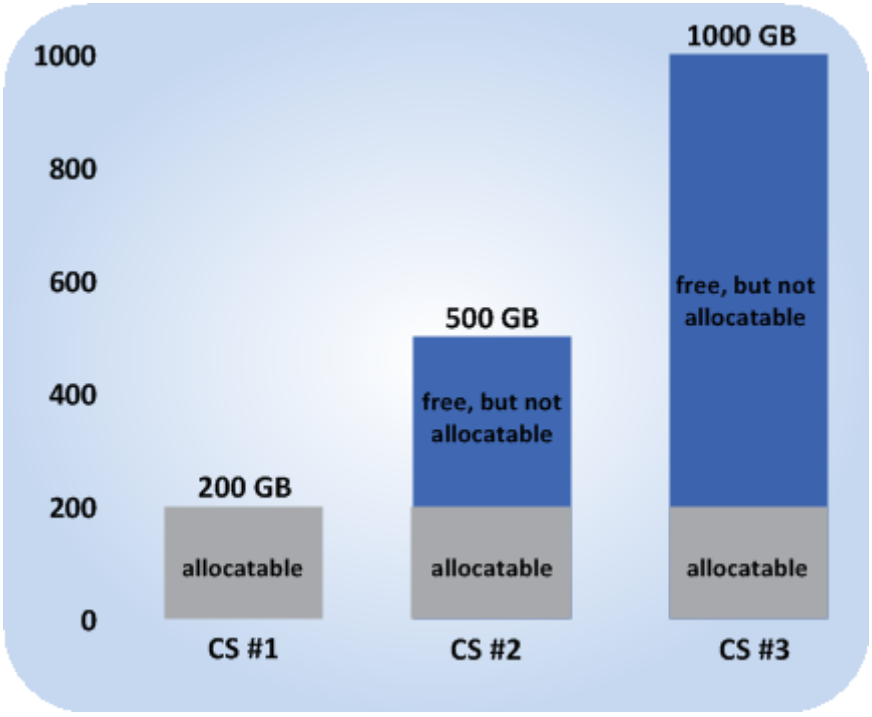
When monitoring disk space information in the cluster, you also need to pay attention to the space reported by the `vstorage top` utility as *allocatable*. Allocatable space is the amount of disk space that is free and can be used for storing user data. Once this space runs out, no data can be written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

Calculation of allocatable disk space is illustrated on 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 is 200 GB, which equals 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
...
```

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 the replication factor changes 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).

Allocatable disk space may also be limited by license.

4.3.1.2 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
...
```

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

4.3.2 Exploring Chunk States

The following is a list of all possible states a chunk can be in.

healthy Percentage of chunks that have enough active replicas. The normal state of chunks.

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.

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.

replicating Percentage of chunks which are being replicated. Write operations on such chunks are frozen until replication ends.

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.

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.

deleting Percentage of chunks queued for deletion.

unique Percentage of chunks that do not have replicas.

4.4 Monitoring Clients

By monitoring clients, you can check the status and health of servers that you use to access virtual machines. 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   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 clients (highlighted in red) are as follows.

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.

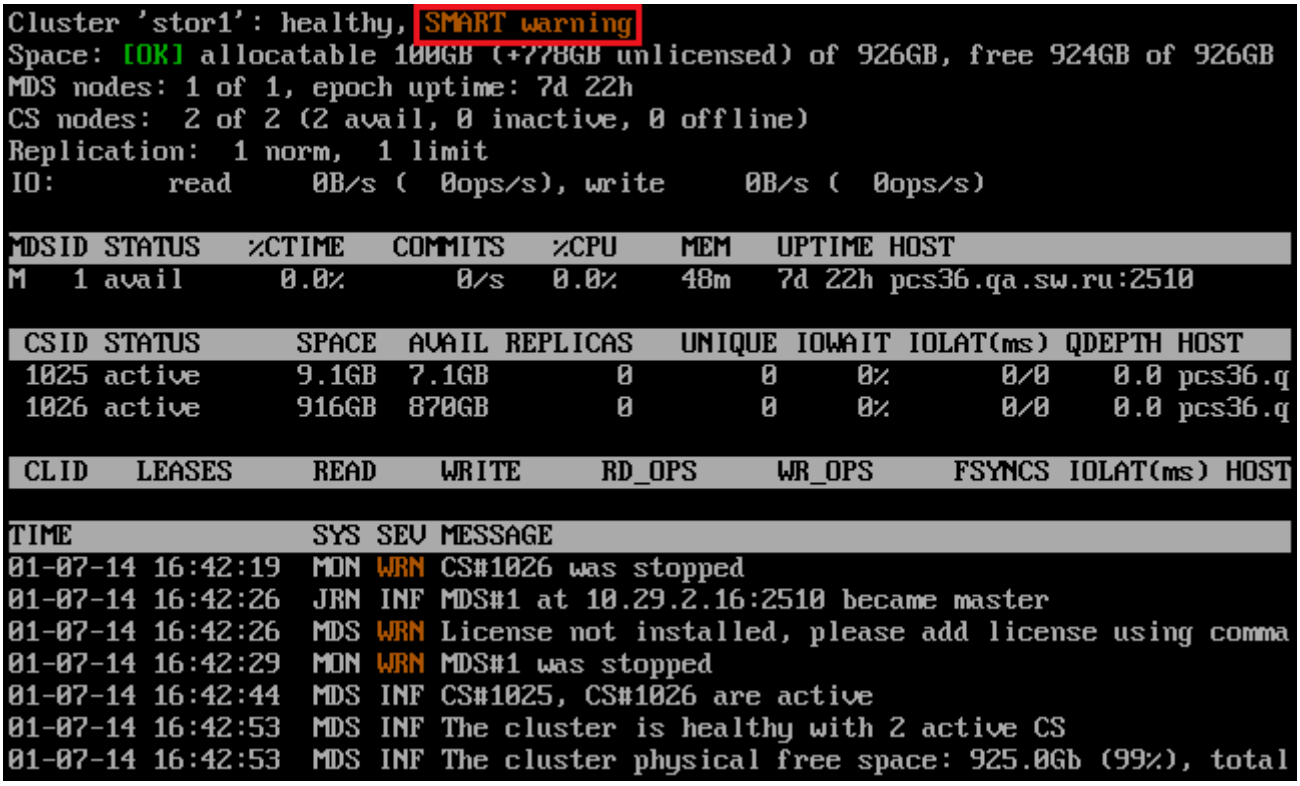
IOLAT Average/maximum time, in milliseconds, the client needed to complete a single IO operation, for the last 20 seconds.

HOST Client hostname or IP address.

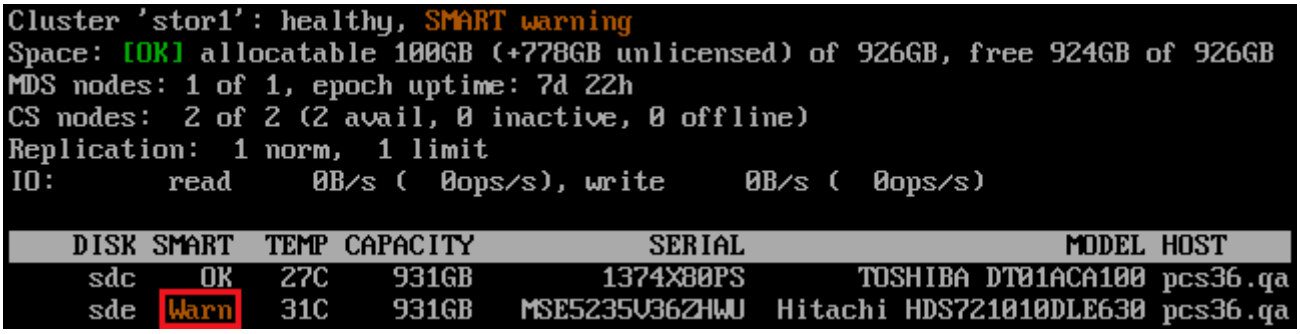
4.5 Monitoring Physical Disks

The S.M.A.R.T. status of physical disks is monitored by the `smartctl` tool installed along with Virtuozzo Infrastructure Platform. For it to work, S.M.A.R.T. functionality must be enabled in node's BIOS. The tool is run every 10 minutes as a `cron` job also added during installation. The `smartctl` tool polls all physical disks attached to nodes in the cluster, including caching and journaling SSDs, and reports the results to the MDS server.

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



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:



The disks table shows the following parameters:

DISK Disk name assigned by operating system.

SMART Disk's S.M.A.R.T. status:

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:

- Reallocated Sector Count
- Reallocated Event Count
- Current Pending Sector Count
- Offline Uncorrectable

TEMP Disk temperature in Celsius.

CAPACITY Disk capacity.

SERIAL Disk serial number.

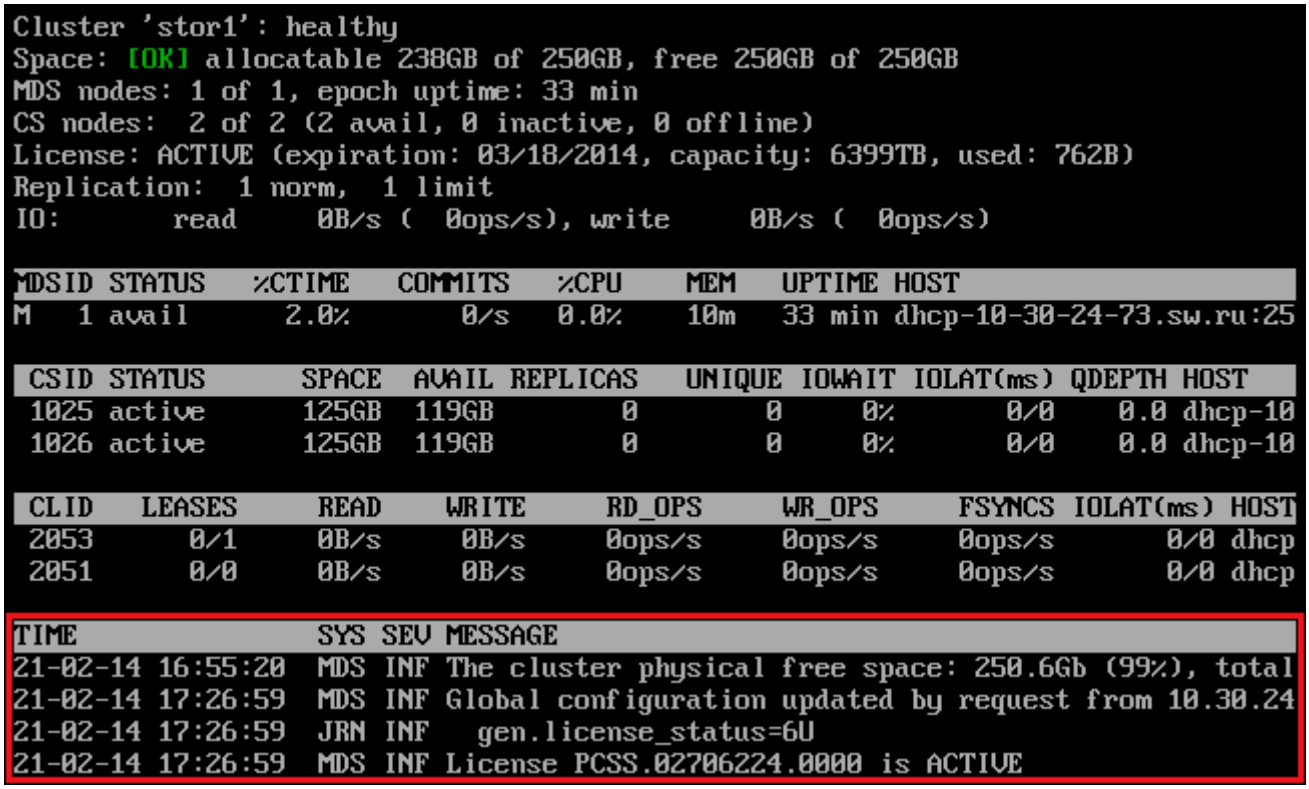
MODEL Disk model.

HOST Disk's host address.

To disable S.M.A.R.T. disk monitoring, delete the corresponding `cron` job.

4.6 Monitoring Event Logs

You can use the `vstorage -c <cluster_name> top` utility to monitor significant events happening in the storage cluster. For example:



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

TIME Time of event.

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.

The following table lists basic events displayed when you run the `vstorage top` utility.

Table 4.1: Basic events

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.

Continued on next page

Table 4.1 – continued from previous page

Event	Severity	Description
MDS#<N> (<addr>:<port>) didn't accept commits for *M* sec	JRN err	Generated by the MDS master server if MDS#<N> did not accept commits for *M* seconds. MDS#<N> gets marked as stale. This message may indicate that the MDS service on MDS#<N> is experiencing a problem. The problem may be critical and should be resolved 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 stale. 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 and should be resolved 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 *N* active CS	MDS info	Generated when the cluster status changes to healthy 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 4.1 – continued from previous page

Event	Severity	Description
The cluster is degraded with *N* active, *M* inactive, *K* offline CS	MDS warn	Generated when the cluster status changes to degraded or when a new MDS master server is elected. This message indicates that some chunk servers in the cluster are • inactive, i.e. do not send any registration messages, or • offline, i.e. have been inactive for longer than `mds.wd.offline_tout`, which is 5min by default.
The cluster failed with *N* active, *M* inactive, *K* offline CS (mds.wd.max_offline_cs=<n>)	MDS err	Generated when the cluster status changes to failed or when a new MDS master server is elected. This message indicates that the number of offline chunk servers exceeds `mds.wd.max_offline_cs`, which is 2 by default. When the cluster fails, the automatic replication is not scheduled any more. So the cluster administrator must take action to either repair failed chunk servers or increase `mds.wd.max_offline_cs`. Setting this value to 0 disables the failed mode completely.
The cluster is filled up to <N>%	MDS info/warn	Shows the current space usage in the cluster. A warning is generated if the disk space consumption equals or exceeds 80%. It is important to have spare disk space for data replicas if one of the chunk servers fails.
Replication started, N chunks are queued	MDS info	Generated when the cluster starts automatic data replication to recover the missing replicas.
Replication completed	MDS info	Generated when the cluster finishes automatic data replication.

Continued on next page

Table 4.1 – continued from previous page

Event	Severity	Description
CS#<N> has reported hard error on *path*	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 *T* 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 automatic replication of data to restore the replicas that were stored on the offline chunk server.
Failed to allocate *N* replicas for '*path*' by request from <addr>:<port> - *K* out of *M* chunks servers are available	MDS warn	Generated when the cluster cannot allocate chunk replicas, for example, when it runs out of disk space.
Failed to allocate *N* replicas for '*path*' by request from <addr>:<port> since only *K* chunk servers are registered	MDS warn	Generated when the cluster cannot allocate chunk replicas because not enough chunk servers are registered in the cluster.

4.7 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. Typical command output may look like this:

```
# 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%.

   For more information on available chunk statuses, see *Exploring Chunk States* (page 51).

CHAPTER 5

Maximizing Storage Cluster Performance

This chapter describes recommendations for maximizing the performance of your storage cluster.

Note: Also consider updating storage cluster nodes.

5.1 Carrying Out Performance Benchmarking

When testing the performance of a storage cluster and comparing it with third-party 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, perform benchmarks from inside virtual machines.
- Keep in mind that the data replication factor affects the storage cluster performance: clusters with two replicas are slightly faster than those with three replicas.

5.2 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.

Virtuozzo Infrastructure Platform ships 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 a different Virtuozzo Infrastructure Platform server than the one with the storage device to check, install the `vstorage-hwflush-check` tool, which is a part of the `vstorage-ctl` package:

   ```
   # yum install vstorage-ctl
   ```

2. Run the `vstorage-hwflush-check` server:

   ```
   # vstorage-hwflush-check -l
   ```

On the client part:

1. On the Virtuozzo Infrastructure Platform server with the 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 press the **Power** button or disconnect the power cord to switch off the computer.

---

4. 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 and a “cache error detected” message is returned, 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.

5.3 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 node. Although if you have 1 or 2 HDDs on a node, two bonded network adapters are still recommended for better reliability (see *Setting Up Network Bonding* (page 65)).
- A 10 Gbit/s link per node for the maximum performance.

The following table illustrates how these recommendations may apply to a node with 1 to 6 HDDs:

Table 5.1: 1 GbE and 10 GbE link examples

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)

Take note of the following:

- For the maximum sequential I/O performance, it is recommended to use one 1 Gbit/s link per each hard drive, or one 10 Gbit/s link per 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.

5.4 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 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=no
   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="no"
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: For more information on network bonding, see the *Red Hat Enterprise Linux Deployment Guide* and *Linux Ethernet Bonding Driver HOWTO.*

5.5 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 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 Virtuozzo Infrastructure Platform 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.

You can check for unaligned write operations in the storage cluster 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.

5.6 Disabling Inter-Tier Data Allocation

If a storage tier runs out of free space, Virtuozzo Infrastructure Platform will attempt to temporarily use the space of the lower tiers down to the lowest. If the lowest tier also becomes full, Virtuozzo Infrastructure Platform 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
```

CHAPTER 6

Advanced Tasks

This chapter describes miscellaneous configuration and management tasks that you may need to perform.

6.1 Updating Kernel with ReadyKernel

ReadyKernel is a kpatch-based service shipped with Virtuozzo Infrastructure Platform and available out-of-the-box on physical servers with active licenses. ReadyKernel offers a more convenient, rebootless alternative to updating the kernel the usual way and allows you not to wait for scheduled server downtime to apply critical security updates. ReadyKernel enables you to receive cumulative kernel patches that fix critical security issues and apply these patches without having to reboot the server. ReadyKernel updates are released for kernels younger than 18 months. When a kernel becomes older that 18 months, you need to switch to a newer kernel to keep receiving ReadyKernel updates.

Upon installation, the patches are loaded into server RAM and immediately applied to the kernel. If the server reboots, these patches are reapplied to the kernel on boot. You can check the details of the applied ReadyKernel patch at any time with `readykernel info`.

If later you install a new kernel or a major kernel update that requires a reboot, the downloaded patches will remain on the server but will not be applied.

In Virtuozzo Infrastructure Platform, ReadyKernel is set to automatically download and apply updates. Checks for new patches are added to each `yum` transaction that takes place on any node in the infrastructure.

Even though ReadyKernel requires no user interaction by default, you can read the following subsections to understand how this tool works and manage it if needed.

6.1.1 Installing ReadyKernel Patches Automatically

ReadyKernel is enabled by default and checks for new patches daily at 12:00 server time by means of a `cron.d` script. If a patch is available, ReadyKernel will download, install, and load it for the current kernel.

To disable automatic updating, run

```
# readykernel autoupdate disable
```

You can re-enable automatic updating later with the following command:

```
# readykernel autoupdate enable <hour>
```

The service will check for patches daily at the specified `<hour>` (set in 24-hour format, server time).

6.1.2 Managing ReadyKernel Patches Manually

6.1.2.1 Dowloading, Installing, and Loading ReadyKernel Patches

To download, install, and instantly load the latest ReadyKernel patch for the current kernel, do the following:

1. Check for new ReadyKernel patches:

   ```
   # readykernel check-update
   ```

2. If a new patch is available, download, install, and instantly load it for the current kernel by running:

   ```
   # readykernel update
   ```

   Note: You can also do this with `yum update`.

ReadyKernel patches are cumulative, i.e. the latest patch includes all the previous ones. To keep the kernel secure, you only need to install and load the latest patch.

6.1.2.2 Loading and Unloading ReadyKernel Patches

To manually load the latest installed ReadyKernel patch to the kernel, do one of the following:

- If an older patch is already loaded, unload it first, then load the latest patch by running:

```
# readykernel load-replace
```

- If no older patches are loaded, load the latest patch by running:

```
# readykernel load
```

To unload the patch from the current kernel, run

```
# readykernel unload
```

6.1.2.3 Installing and Removing ReadyKernel Patches for Specific Kernels

If multiple kernels are installed on the server, you can install a ReadyKernel patch for a specific kernel:

```
# yum install readykernel-patch-<kernel_version>
```

To remove a specific ReadyKernel patch from the server, run

```
# yum remove readykernel-patch-<kernel_version>
```

6.1.2.4 Downgrading ReadyKernel Patches

If you experience problems with the latest ReadyKernel patch, you can downgrade it to an older version if one is available.

To downgrade a patch for the current kernel to the previous version, run

```
# yum downgrade readykernel-patch-$(uname -r)
```

To downgrade a patch for a specific kernel to the previous version, run

```
# yum downgrade readykernel-patch-<kernel_version>
```

You can run these commands multiple times to downgrade to the patch version you need. Alternatively, you can downgrade a patch to a specific version by specifying the desired patch version. For example:

```
# yum downgrade readykernel-patch-12.7-0.4-17.vl7
```

6.1.2.5 Disabling Loading of ReadyKernel Patches on Boot

If for some reason you do not want ReadyKernel patches to be applied at boot time, run the following command:

```
# readykernel autoload disable
```

To re-enable automatic loading of ReadyKernel patches on boot, run

```
# readykernel autoload enable
```

6.1.2.6 Managing ReadyKernel Logs

ReadyKernel logs event information in `/var/log/messages` and `/var/log/kpatch.log`. You can specify logging parameters for the latter in the configuration file `/etc/logrotate.d/kpatch`. For more information on parameters you can use, see the `logrotate` man page.

6.2 Managing Guest Tools

This section explains how to install and uninstall guest tools in virtual machines. This functionality is required for *Running Commands in Virtual Machines without Network Connectivity* (page 75).

Note: To be able to use the `vinfra` CLI tool mentioned in this section, you may need to perform steps listed in “Providing Credentials” in the *CLI Reference*.

6.2.1 Installing Guest Tools in Linux Virtual Machines

To install guest tools in an existing Linux virtual machine, do the following on the controller node of your compute cluster:

1. Create an image from a guest tools ISO image. There are two guest tools ISO images: `vz-guest-tools-lin.iso` for Linux and `vz-guest-tools-win.iso` for Windows, which are located in `/usr/share/vz-guest-tools/`. For example:

   ```
   # vinfra service compute image create vz-guest-tools-lin \
   --file /usr/share/vz-guest-tools/vz-guest-tools-lin.iso --os-distro linux
   ```

```
Uploading image to server [Elapsed Time: 0:00:12] ...
```

2. Create a volume from the image. You will need the image ID that you can obtain with `vinfra service compute image list`. For example:

```
# vinfra service compute volume create vz-guest-tools-lin-vol --storage-policy default \
--size 1 --image d9a01520-6634-48d5-9803-34e5c38d43d2
```

3. Attach the guest tools volume to the virtual machine. You will need the IDs of the VM and the guest tools volume. You can obtain them with `vinfra service compute server list` and `vinfra service compute volume list`, respectively.

```
# vinfra service compute server volume attach \
--server 1d45a54b-0e20-4d5e-8f11-12c8b4f300db 1a40012a-7976-47a1-81f1-ff498cba90af
+--------+--------------------------------------+
| Field  | Value                                |
+--------+--------------------------------------+
| device | /dev/sdb                             |
| id     | 1a40012a-7976-47a1-81f1-ff498cba90af |
+--------+--------------------------------------+
```

4. Run guest tools installer inside the VM.

Note: Guest tools rely on QEMU guest agent which is installed alongside the tools. The agent daemon/service `qemu-ga` must be running for the tools to work.

Log in to the virtual machine, create a mount point for the optical drive with the guest tools image and run the installer:

```
# mkdir /mnt/cdrom
# mount /dev/sdb /mnt/cdrom
# bash /mnt/cdrom/install
```

6.2.2 Installing Guest Tools in Windows Virtual Machines

Currently you can only attach volumes but not images (i.e. virtual HDDs but not CD-ROMs) to existing VMs. This makes it impossible to attach a guest tools ISO image to an existing Windows VM. Following are two workarounds that allow you to install guest tools in a Windows VM. Perform commands on the controller node of the compute cluster.

Note: Round up the size of volumes to be created from ISO images. E.g., if the OS distribution image is 2.6

GB, use `size=3`.

6.2.2.1 Method #1

Attach the guest tools ISO image alongside the OS distribution ISO image when creating a new Windows VM. For example:

```
# vinfra service compute server create newvm --network id=private --flavor medium \
--volume source=blank,size=64,boot-index=0,type=disk \
--volume source=image,id=e6548c13-8479-4460-ab85-5ed60ff0e930,size=3,boot-index=1,type=cdrom \
--volume source=image,id=2e06406a-4ca0-43bd-8bd4-3a3e8372f997,size=1,boot-index=2,type=cdrom
```

In this example, the first volume is the blank virtual HDD, the second volume is the OS distribution ISO image, and the third volume is the guest tools ISO image. Make sure to specify the correct boot order by means of the `boot-index` parameter.

6.2.2.2 Method #2

Power off the Windows VM, convert its system volume to a template image, and create a new Windows VM from the template, attaching the guest tools ISO image to it during creation. For example:

```
# vinfra service compute server stop 60247b5b-2eef-4fb3-9d0d-a87322511489
# vinfra service compute volume upload-to-image 7116d747-a1e1-4200-bd4a-25cc51ef006c | grep id
| id               | 79da5239-b2bb-4779-ada2-46cb8da8ba0e                              |
# vinfra service compute server create newvm --network id=private --flavor medium \
--volume source=image,id=79da5239-b2bb-4779-ada2-46cb8da8ba0e,size=64,boot-index=0,type=disk \
--volume source=image,id=2e06406a-4ca0-43bd-8bd4-3a3e8372f997,size=1,boot-index=1,type=cdrom
```

In this example, the first volume is the template of the original VM's system disk and the second volume is the guest tools ISO image. Make sure to specify the correct boot order by means of the `boot-index` parameter.

Once the ISO is mounted inside the Windows VM, launch the installer in the AutoPlay window if autorun is enabled. Otherwise open the optical drive in Explorer and run `setup.exe`.

6.2.3 Uninstalling Guest Tools

The steps you need to perform to remove guest tools depend on the guest OS and are described in the following sections.

6.2.3.1 Uninstalling Guest Tools from Linux Virtual Machines

To uninstall guest tools from a Linux guest, log in to the virtual machine and do as follows:

1. Remove the packages:

   1.1. On RPM-based systems (CentOS and other):

   ```
   # yum remove dkms-vzvirtio_balloon prl_nettool qemu-guest-agent-vz vz-guest-udev
   ```

   1.2. On DEB-based systems (Debian and Ubuntu):

   ```
   # apt-get remove vzvirtio-balloon-dkms prl-nettool qemu-guest-agent-vz vz-guest-udev
   ```

   If any of the packages listed above are not installed on your system, the command will fail. In this case, exclude these packages from the command and run it again.

2. Remove the files:

   ```
   # rm -f /usr/bin/prl_backup /usr/share/qemu-ga/VERSION /usr/bin/install-tools \
   /etc/udev/rules.d/90-guest_iso.rules /usr/local/bin/fstrim-static /etc/cron.weekly/fstrim
   ```

3. Reload the udev rules:

   ```
   # udevadm control --reload
   ```

After removing guest tools, restart the virtual machine.

6.2.3.2 Uninstalling Guest Tools from Windows Virtual Machines

To uninstall guest tools for Windows, log in to the virtual machine and do as follows:

1. Remove QEMU device drivers from the device manager.

   Important: Do not remove the VirtIO/SCSI hard disk driver and NetKVM network driver. Without the former, the VM will not boot; without the latter, the VM will lose network connectivity.

2. Uninstall QEMU guest agent and guest tools from the list of installed applications.

3. Stop and delete Guest Tools Monitor:

   ```
   > sc stop VzGuestToolsMonitor
   > sc delete VzGuestToolsMonitor
   ```

4. Unregister Guest Tools Monitor from Event Log:

```
> reg delete HKLM\SYSTEM\CurrentControlSet\services\eventlog\Application\VzGuestToolsMonitor
```

5. Delete the autorun registry key for RebootNotifier:

```
> reg delete HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run /v VzRebootNotifier
```

6. Delete the `C:\Program Files\Qemu-ga\` directory.

   If `VzGuestToolsMonitor.exe` is locked, close all the Event Viewer windows. If it remains locked, restart the `eventlog` service:

```
> sc stop eventlog
> sc start eventlog
```

After removing the guest tools, restart the virtual machine.

6.3 Running Commands in Virtual Machines without Network Connectivity

If a VM cannot access a network for some reason, you can still run commands in it from the node the VM resides on. The VM in question must have the guest tools installed in it (see *Managing Guest Tools* (page 71)).

You will need the VM ID that you can obtain with `vinfra service compute server list`. You can also use a virsh domain name that you can get using `virsh list`.

6.3.1 Running Commands in Linux Virtual Machines

To run an arbitrary command inside a Linux VM and receive the output to your console, use the `virsh x-exec` command. For example:

```
# virsh x-exec 1d45a54b-0e20-4d5e-8f11-12c8b4f300db /usr/bin/bash -c 'lsblk'
NAME        MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
loop0         7:0    0 945.9M  1 loop
loop1         7:1    0     5G  1 loop
  live-rw   253:0    0     5G  0 dm   /
  live-base 253:1    0     5G  1 dm
loop2         7:2    0    32G  0 loop
  live-rw   253:0    0     5G  0 dm   /
sda           8:0    0    64G  0 disk
sdc           8:32   0     1G  1 disk
sr0          11:0    1     2G  0 rom  /run/initramfs/live
```

To copy a file to a Linux VM, use the `virsh x-exec` and `cat` commands. For example:

```
# virsh x-exec 1d45a54b-0e20-4d5e-8f11-12c8b4f300db \
--shell 'cat > test.file' < /home/test.file
```

To get a file from a Linux VM, use the `virsh x-exec` and `cat` commands as well. For example:

```
# virsh x-exec 1d45a54b-0e20-4d5e-8f11-12c8b4f300db \
--shell 'cat /home/test.file' > test.file
```

6.3.2 Running Commands in Windows Virtual Machines

To run an arbitrary command inside a Windows VM and receive the output to your console, use the `virsh x-exec` command. For example:

```
# virsh x-exec bbf4a6ec-865f-4e2c-ac21-8639d1bfb85c --shell dir c:\\
 Volume in drive C has no label.
 Volume Serial Number is D0BE-A8D1

 Directory of c:\

06/10/2009  01:42 PM                24 autoexec.bat
06/10/2009  01:42 PM                10 config.sys
07/13/2009  06:37 PM    <DIR>          PerfLogs
11/12/2018  07:45 AM    <DIR>          Program Files
11/12/2018  07:55 AM    <DIR>          test
11/12/2018  06:23 AM    <DIR>          Users
11/12/2018  07:53 AM    <DIR>          Windows
               2 File(s)             34 bytes
               5 Dir(s)  59,329,495,040 bytes free
```

To copy a file to a Windows VM, use the `virsh x-exec` and `prl_cat` commands. For example:

```
# virsh x-exec bbf4a6ec-865f-4e2c-ac21-8639d1bfb85c \
--shell '%programfiles%\\qemu-ga\\prl_cat c:\\test\\test.file' < /home/test.file
```

To get a file from a Windows VM, use the `virsh x-exec` and `type` commands. For example:

```
# virsh x-exec bbf4a6ec-865f-4e2c-ac21-8639d1bfb85c \
--shell 'type c:\\test\\test.file' > test.file
```

6.4 Setting Virtual Machines CPU Model

Virtual machines are created with the host CPU model by default. If nodes in the compute cluster have different CPUs, live migration of VMs between compute nodes may not work or applications inside VMs that depend on particular CPUs may not function properly. To avoid this, you can find out which CPU model offers

compatibility across all nodes in the compute cluster and manually set it as the compute cluster default.

Do the following:

1. Run `virsh capabilities` on each node to print an XML document with information on node's CPU. Join the `<cpu>` sections from all XML outputs to a single XML file, e.g., `cpu-compare.xml`.
2. Compare the CPU features using `virsh cpu-baseline`. For example:

   ```
   # virsh cpu-baseline cpu-compare.xml | grep model
     <model fallback='allow'>IvyBridge</model>
   ```

   The command will print the most compatible CPU model across all nodes.
3. Set this CPU model for the compute cluster. For example:

   ```
   # vinfra service compute cluster set --cpu-model IvyBridge
   ```

Take note of the following:

- For the list of supported CPU models, run `vinfra service compute cluster set --help`.
- Changing CPU model affects only new VMs (i.e. those created after the change).

See the *CLI Reference* for more details on the command and information on how to use the `vinfra` tool.

6.5 Creating Linux Templates

If you do not have a ready Linux template, you can build one with the `diskimage-builder` tool. The disk image is created with only the root user that has neither password nor SSH keys. You can use the `user data` and `cloud-init` methods to perform initial configuration tasks on VMs that will be deployed from the disk image, for example, create custom user accounts. For more options to customize a VM during boot, refer to the cloud-init documentation.

To create a template and deploy a VM from it, do as follows:

1. Install the `diskimage-builder` package:

   ```
   # yum install diskimage-builder
   ```
2. For the RHEL 7 guest OS, download the cloud image from the Red Hat Customer Portal (login required) and execute:

   ```
   # export DIB_LOCAL_IMAGE=<path_to_rhel7_image>
   ```

3. Execute the following command to build a disk image with installed `cloud-init` for the desired Linux guest. For example:

```
# disk-image-create vm centos7 -t qcow2 -o centos7
```

where

- `centos7` is the name of a guest OS. Can be one of the following: `centos6`, `centos7`, `debian`, `rhel7`, or `ubuntu`.

  By default, using the `ubuntu` element will create a disk image for Ubuntu 16.04. To build the Ubuntu 18.04 disk image, add the `DIB_RELEASE=bionic` to the command as follows: `DIB_RELEASE=bionic disk-image-create vm ubuntu -t qcow2 -o ubuntu18`.

- `-o` sets the name for the resulting disk image file.

4. Upload the created disk image using the `vinfra` tool to the compute cluster:

```
# vinfra service compute image create centos7-image --os-distro centos7 \
--disk-format qcow2 --file centos7.qcow2
```

where

- `centos7-image` is the name of a new image.
- `centos7` is the corresponding OS version. Can be one of the following: `centos6`, `centos7`, `debian9`, `rhel7`, `ubuntu16.04`, and `ubuntu18.04`.
- `centos7.qcow2` is the QCOW2-image created on step 3.

5. Create the `user-data` configuration file with a custom user account:

```
# cat <<EOF > user-data
#cloud-config

users:
  - name: user
    lock-passwd: false
    sudo: ALL=(ALL) NOPASSWD:ALL
    passwd: $6$CjI58V.q7gFvBlMU$9lvue5t8l/VD/tR9L8VTgaRnIksr7ZkSvjtxGxAelJaa.\
WWAmJwRKLjbJwjikOSO2P5DCEYSX/Uf.MuvvhrCE/
EOF
```

where `user` is the name of a custom user and `$6$CjI58V<...>` is a hashed password for the account.

6. Launch the deployment of a VM from the disk image using the configuration file as user data:

```
# vinfra service compute server create centos7-vm --flavor medium --network public \
--user-data user-data --volume source=image,id=centos7-image,size=10
```

where

- `centos7-vm` is the name of a new VM,
- `user-data` is the configuration file created in step 5,
- `centos7-image` is the image added to the compute cluster in step 4.

For more information on using the `vinfra` tool, see *Command Line Reference*.

6.6 Securing OpenStack API Traffic with SSL

By means of the **Compute API** traffic type, Virtuozzo Infrastructure Platform exposes a public endpoint that listens to OpenStack API requests. By default, it points to the IP address of the management node (or to its virtual IP address if high availability is enabled).

Traffic to and from the endpoint can be secured with an SSL certificate. However, as domain names are not used by default, the certificate will need a `subjectAltName` field containing the aforementioned management node IP address. If it does not have such a field, you will need to modify the public endpoint to use a domain name that you have a certificate for.

To secure public OpenStack API traffic with SSL, do the following:

1. Upload the certificate and then private key in the admin panel, on the **SETTINGS** > **Management node** > **SSL ACCESS** screen.
2. Place the CA certificate file to operating system's trusted bundle:

   ```
   # cp ca.pem /etc/pki/ca-trust/source/anchors/
   # update-ca-trust extract
   ```

   Alternatively, you can append the `--os-cacert ca.pem` option to each OpenStack client call.
3. If your certificate does not have the `subjectAltName` field, modify all public endpoints to use the domain name for which you have the certificate for. This domain name must resolve to the management node IP address (or to its virtual IP address if high availability is enabled). For example:

   ```
   # openstack --insecure endpoint list | grep public
   | 44aa0f53a40e4e52b1c7eeeb20c7811e | <...> | https://10.94.16.12:8774/v2.1/%(tenant_id)s |
   | 5a845b4b813047c292db73c42dad5efd | <...> | https://10.94.16.12:8780                    |
   ```

```
| 0b906e518b1041c8b94af7f410403369 | <...> | https://10.94.16.12:9696                    |
| d80af756adf1449f9237c3aeebc9206a | <...> | https://10.94.16.12:8004/v1/%(tenant_id)s   |
| d0e8c7da7d174e1f9aa4efbc6dff2113 | <...> | https://10.94.16.12:5000/v3                 |
| 0e6d3a39d6c44aa883984a35dde434bb | <...> | https://10.94.16.12:9292                    |
| 7d901686bca549f9b294e572f046f634 | <...> | https://10.94.16.12:8776/v2/%(tenant_id)s   |
| 1b68ac7c3f7949fbaeef4a815fe6f3b1 | <...> | https://10.94.16.12:8776/v3/%(tenant_id)s   |

# openstack --insecure endpoint set \
--url https://<DNS_name>:8774/v2.1/%(tenant_id)s 44aa0f53a40e4e52b1c7eeeb20c7811e
# openstack --insecure endpoint set \
--url https://<DNS_name>:8780 5a845b4b813047c292db73c42dad5efd
# openstack --insecure endpoint set \
--url https://<DNS_name>:9696 0b906e518b1041c8b94af7f410403369
# openstack --insecure endpoint set \
--url https://<DNS_name>:8004/v1/%(tenant_id)s d80af756adf1449f9237c3aeebc9206a
# openstack --insecure endpoint set \
--url https://<DNS_name>:5000/v3 d0e8c7da7d174e1f9aa4efbc6dff2113
# openstack --insecure endpoint set \
--url https://<DNS_name>:9292 0e6d3a39d6c44aa883984a35dde434bb
# openstack --insecure endpoint set \
--url https://<DNS_name>:8776/v2/%(tenant_id)s 7d901686bca549f9b294e572f046f634
# openstack --insecure endpoint set \
--url https://<DNS_name>:8776/v3/%(tenant_id)s 1b68ac7c3f7949fbaeef4a815fe6f3b1
```

4. In your OpenRC script, change `OS_AUTH_URL` to the same domain name and remove all parameters related to insecure access. For example:

```
export OS_PROJECT_DOMAIN_NAME=Default
export OS_USER_DOMAIN_NAME=Default
export OS_PROJECT_NAME=admin
export OS_USERNAME=admin
export OS_PASSWORD=<ADMIN_PASSWORD>
export OS_AUTH_URL=https://<DOMAIN_NAME>:5000/v3
export OS_IDENTITY_API_VERSION=3
```

Now you can run OpenStack commands without the `--insecure` option.

6.7 Enabling Backup Gateway Geo-Replication

Make sure the following prerequisites are met:

- Two or more storage clusters with ABGW are deployed.
- All storage clusters are updated to the latest version.
- All storage clusters can ping each other via IP addresses or domain names on port 44445.
- All storage clusters are registered in Acronis Backup Cloud.

To set up geo-replication between two storage clusters, a master and a slave, do the following:

1. Find out the `dc_uid` values from `/mnt/vstorage/vols/acronis-backup/conf.d/dc_uid` on both the master and the slave.
2. Copy the `/mnt/vstorage/vols/acronis-backup/certs/abgw.pem` files from both the master and the slave to a machine from which you will configure replication (i.e. run `vstorage-abgw-ctl`). For example, to `master_abgw.pem` and `slave_abgw.pem`, respectively.
3. Configure replication:

```
# vstorage-abgw-ctl replication \
--master-addr <master_DNS_name> --master-cert master_abgw.pem --master-uid <master_dc_uid> \
--slave-addr <slave_DNS_name> --slave-cert slave_abgw.pem --slave-uid <slave_dc_uid> \
--enable /<account-name>
```

This command will enable replication of existing files. If new files are added to an account, the command will need to be re-run. For more convenience, you can set up a cron job that will run the following script automatically on one of the storage nodes:

```
#!/bin/bash

cd /mnt/vstorage/vols/acronis-backup/storage
for dir in *; do
  vstorage-abgw-ctl replication \
    --master-addr <master_DNS_name> --master-cert master_abgw.pem --master-uid <master_dc_uid> \
    --slave-addr <slave_DNS_name> --slave-cert slave_abgw.pem --slave-uid <slave_dc_uid> \
    --enable /$dir
done
```

If PEM certificates expire, update them in the web interface and exchange them between clusters once again.

If a failover to the slave replica is needed, contact the technical support team. A tool for performing unassisted failovers will be added in a future update.