Violin MEMORY

Volume Driver V4.0 Installation and Configuration Guide

For OpenStack Cinder Kilo Release

Document Number: 535-0258-00 Rev 01
March 2016

LEGAL NOTICE

Violin Memory, Inc.
4555 Great America Parkway
Santa Clara, CA 95054
USA

Contents

Preface

This preface outlines the organization of this book, describes document conventions, and provides information about additional resources.

Intended Audience

This guide is intended for experienced systems administrators. Violin Memory assumes that you are experienced in installing and servicing high-performance storage systems.

Contact Violin Memory Customer Support for any assistance with installing and servicing this system. See Contacting Violin Memory on page 4 for contact information.

Reference Documents

In addition to this guide, the following Violin Memory documents comprise the documentation suite that will assist you with setting up, using and servicing the Violin 7000 Series Flash Storage Platforms. These guides are available for download from the Violin Memory Support site at: http://www.violin-memory.com/support/

This document...	Provides this information...
Release Notes	This document describes the new features, resolved issues, known limitations and software upgrade instructions for the current release.
7000 Series Flash Storage Platform Installation Guide	This guide provides instructions for installing the Violin 7000 Series Flash Storage Platforms in an equipment rack and completing the system setup and configuration.
7000 Series Flash Storage Platform User's Guide	This guide provides instructions for managing, monitoring, and maintaining a 7700 or 7000 Series FSP.
7700 Flash Storage Platform Installation Guide	This guide provides instructions for installing and configuring the devices in a 7700 Flash Storage Platform configuration.
7300 Flash Storage Platform Best Practices Guide	This guide provides instructions and best practices recommendations for connecting, configuring and optimizing a 7000 Series Flash Storage Platform in a Fibre Channel storage network.
Violin Symphony (online help)	If the Violin Symphony management tool is installed on your system, you can also use Violin Symphony to monitor Violin Memory Flash Arrays from a Web browser. See http://www.violin-memory.com/ for more information.

Reference Documents

Document Conventions

Violin Memory documentation follows the conventions outlined in this section.

Important Information

The following table summarizes the notations used to call out important information, such as warning, caution, and note using example text.

Important Notations

Notation and Sample Text
WARNING! Only authorized, qualified, and trained personnel should attempt to work on this equipment.
Caution: Follow the listed safety precautions when working on the Flash Storage Platform.
Note: Read through this entire chapter and plan your installation according to your location before installing the equipment. The following procedures and the order in which they appear are general installation guidelines only.

Typographical Conventions

The following typographic conventions are used in this guide:

Format	Meaning
Bold	User Interface text.
Italic	Provides emphasis and identifies variables and document titles.
`Courier`	Command names, examples, and output.
`Courier bold`	Input you must type exactly as shown.
`<Courier italic>`	Information for which you must supply a value.
`[ ]`	Optional command parameters are enclosed within square brackets.
|	Separates a set of command choices from which only one may be chosen.
`{ }`	Required command parameters that must be specified are enclosed within curly brackets.

Typographical Conventions

Security

Violin Memory, Inc., cannot be responsible for unauthorized use of equipment and will not make allowance or credit for unauthorized use or access.

Contacting Violin Memory

To obtain additional information or technical support for Violin Memory products, contact us at:

Phone: 1-855-VIOLIN-5 (1-855-846-5465)

International: +1 650-396-1500 Extension 3

Web site: http://www.violin-memory.com

When contacting Violin Memory Customer Support, please have the following information available:

- Model and serial number of the system for which you are requesting support.
- Software version.
- A brief description of the problem.

CHAPTER 1 Volume Driver Installation and Configuration

This chapter demonstrates how to install and configure the Violin OpenStack driver, and covers the following topics:

Supported Configurations

This release supports the following configurations:

- OpenStack Kilo release deployments
- 7000 Series Flash Storage Platform (FSP) or 7700 FSP controller running Concerto OS 7.5.7 or later
- Fibre Channel HBAs
- iSCSI HBAs

Note: The Volume Driver is not supported on FSPs running Concerto software releases prior to 7.5.7.

Software Availability and Components

The released software is available as an installable tarball and a RHEL7.0 RPM. Software and support for existing Violin Memory customers is available from the Violin Memory Support portal at: http://www.violin-memory.com/support

The Violin OpenStack driver has two components:

- The Violin Cinder Driver:

  This driver implements the Cinder API calls.
- The vmemclient library:

  This is the Violin Array Communications library to the Flash Storage Platform through a REST-like interface.

Installing vmemclient

The version of vmemclient recommended for VMEM OpenStack 4.0 driver is vmemclient 1.1.6. You can download the driver from the python web site: https://pypi.python.org/pypi/vmemclient/

To install vmemclient driver, do the following:

1. Download, unzip, and untar the vmemclient-1.1.6.tar.gz file.
2. Login as root and navigate to the vmemclient-1.1.6 directory.
3. Run the following command:

```
python setup.py install
```

Installing the Violin Cinder Driver Software

This section covers the following topics:

Downloading the Software

To install the Violin Cinder driver, do the following:

1. Go to http://www.violin-memory.com/support/
2. Log in to Customer Support using your Violin Memory Customer Portal user name and password. (Contact Customer Support if you do not have an account.)
3. Click the **Software Downloads** tab.
4. From the OpenStack Cinder folder, download the cinder driver rpm or the tarball file to a client computer (laptop).

Follow the instructions below, depending on your environment.

Note: The RPM is only for use on Red Hat Enterprise Linux 7.0 systems.

RPM Installation

To perform an RPM installation, do the following:

1. Install the RPM on all Cinder volume hosts. For example:

```
rpm -Uvh openstack-cinder-vmemdriver-x.x.x.noarch.rpm
```

2. Ensure that Fibre Channel is enabled and the HBAs are configured on the Flash Storage Platform. See the *7000 Series Flash Storage Platform Installation Guide* for more information.
3. Configure Cinder to use one of the Violin drivers. Also see Configuration Options on page 8 for more details on configuring the Violin Cinder driver V4.0 and to configure the driver accordingly.
4. Restart cinder-volume.

Tarball Installation

To perform a tarball installation, do the following:

1. Unzip the tarball on the machine(s) running Cinder's volume service (cinder-volume). For example:

```
tar -zxvf openstack-cinder-vmemdriver-x.x.x.tar.gz
```

2. Recursively copy the Cinder directory to the same directory as your Cinder code installation.

- Devstack example:

```
cp -r cinder /opt/stack/cinder
```

- Ubuntu 12.04 example:

```
cp -r cinder /usr/local/lib/python2.7/dist-packages/cinder
```

3. Ensure that Fibre Channel or iSCSI is enabled and the HBAs are configured on the FSP. See the *7000 Series Flash Storage Platform Installation Guide* for more information.
4. Configure Cinder to use one of the Violin drivers. Also see Configuration Options on page 8 for more details on configuring the Violin Cinder driver V4.0 and to configure the driver accordingly.
5. Restart cinder-volume.

Configuration Options

This section covers information on the following configuration options:

Basic Configuration

The Cinder configuration parameters are defined in `/etc/cinder/cinder.conf`. To use the Violin Cinder Driver, this is the only configuration that needs to be changed.

The following table lists the basic options you have to be set prior to running cinder.

IP address or hostname of the Violin 7000 Series Memory Gateway (string value – Can be # ipaddress or hostname) san_ip=
Log-in credentials for the Violin 7000 Series Memory Gateway or 7700 FSP controller (string value) san_login= san_password=
The path to the Violin Cinder FCP driver volume_driver=
IP address of this host- the host where Cinder is running (string value) my_ip=
prefix for iscsi volumes (string value) iscsi_target_prefix=iqn.2004-02.com.vmem:
The port that the iSCSI daemon is listening on (integer # value) #iscsi_port=3260
iscsi target user-land tool to use (string value) #iscsi_helper=tgtadm

The following table lists optional parameters you can set at your preference.

Use thin provisioning for SAN volumes – default is True? # (boolean value, enter **true** for thin provision and **false** if not) san_thin_provision=
Name of the Volume backend (string value) volume_backend_name=
User defined capabilities, a JSON formatted string # specifying key/value pairs. (string value) # The ones particularly supported are dedup and thin. Only these two capabilities are # listed here in cinder.conf, will this backend be selected for creating luns which have # volume type associated with them that have 'dedup' or 'thin' extra_specs specified extra_capabilities={}

Additional Configuration for a 7700 FSP Controller

The 7700 FSP controller, also known as an external head setup, requires the following additional configuration.

Note: The following table contains configuration parameters that are required for the 7700 FSP controller to come up. Although optional, it is recommended that these parameters also be set for 7000 Series FSPs.

Storage pools that support Dedup luns only (list value) dedup_only_pools=
Storage pools that are capable of supporting Dedup in addition to other lun types (list value) dedup_capable_pools=
The method of choosing a suitable storage pool for creating a lun. The recognized # methods are random, largest and smallest. When none specified, random method # is used. pool_allocation_method=

For a complete list of Block storage options configurable through Cinder, refer to:
http://docs.openstack.org/kilo/config-reference/content/section_cinder.conf.html

Multi-Backend Configuration

Cinder can be configured to use multiple backend storage devices that are all controlled by a single Cinder Volume node. When used in this mode, each backend storage must have its own section in `cinder.conf`.

Each section identifies the name of the backend. Multiple backends can share the same name. In such a case, cinder scheduler takes care of selecting the most appropriate backend for the current request based on the user-specified volume-type. In addition to a name, each backend configuration should minimally have the following in its configuration section:

A configuration block name that uniquely identifies the backend [Block-name]
Name of the Volume backend (string value) volume_backend_name=
Path to the Violin FCP cinder driver volume_driver=
Log in credentials to the backend storage device san_ip= san_password=
Should luns be setup as thin luns by default (Boolean value) san_thin_provision=
Do we attach/detach volumes in cinder using multipath for # volume to image and image to volume transfers? (boolean # value) – Note: We recommend setting this to true use_multipath_for_image_xfer=true
prefix for iscsi volumes (string value) iscsi_target_prefix=iqn.2004-02.com.vmem:
The port that the iSCSI daemon is listening on (integer # value) #iscsi_port=3260
iscsi target user-land tool to use (string value) #iscsi_helper=tgtadm
#ip addresses of iscsi targets Iscsi_target_ips = 10.1.1.1, 10.1.1.2

For further information on multibackend configuration and support in Cinder, refer to:
http://docs.openstack.org/admin-guide-cloud/content/multi_backend.html

CHAPTER 2 Configuration Settings and Operational Overview

This chapter provides an overview of the most commonly used cinder configuration settings, as well as how to create and extend LUNs and manage storage pools.

For information on the basic configuration parameters, see “Volume Driver Installation and Configuration” on page 5.

This chapter covers the following topics:

Single Node Cinder Configuration

Here is an example configuration for a simple single node Cinder connected to a single backend storage device.

```
[DEFAULT]

volume_backend_name=VMEM_MGA

san_ip = violinfsp-mga

san_login = username

san_password = password

use_multipath_for_image_xfer=true

log_file=cinder.log

log_dir=/opt/stack/logs

#extra_capabilities={"dedup":"True","thin":"True"}

#dedup_only_pools = StoragePoolMga-1, PoolA, PoolB

#dedup_capable_pools = PoolC, PoolD

#pool_allocation_method = "smallest"
```

Multi-backend Configuration

In a typical OpenStack cloud setup with Violin Flash Storage Platforms deployed as the storage platform of choice, there are likely to be many backend devices connected to a Cinder instance. Also, when a 7700 FSP controller platform is used, to get the HA support, both of the external heads operating in an active-active mode should be configured as multiple backends to the Cinder that is using them as the storage of choice.

The following table shows an example of how to configure such a system. There must be a Default section that identifies the backends and the ones enabled in addition to some configuration that is common to all backends like the logging options. An additional section for each of the backend lists the configuration parameters specific to each backend.

```
[DEFAULT]

volume_driver=cinder.volume.drivers.violin.v7000_fcp.V7000FCPDriver

log_file=cinder.log

log_dir=/opt/stack/logs

san_thin_provision=False

use_multipath_for_image_xfer=true

enabled_backends= vmem-violinfsp,vmem-violinfsp

use_multibackend=True

[violinfsp]

volume_driver=cinder.volume.drivers.violin.v7000_fcp.V7000FCPDriver

volume_backend_name=violin-7700-3477

extra_capabilities={"dedup":"True","thin":"True"}

use_multipath_for_image_xfer=true

san_ip=vmem-violinfsp

san_login=user

san_password="Password"

san_thin_provision=False

dedup_only_pools = dedup-pool-3477

[vmem-violinfsp]

volume_driver=cinder.volume.drivers.violin.v7000_fcp.V7000FCPDriver

volume_backend_name=violin-7700-3478
```

```
extra_capabilities={"dedup":"True","thin":"True"}

use_multipath_for_image_xfer=true

san_ip=vmem-violinfsp

san_login=user

san_password="Password"

san_thin_provision=False

dedup_only_pools = dedup-pool-3478
```

Creating LUNs

Thin LUNs can be created in one of two ways. They can either be configured to be the default LUN type (in which case a LUN created with no specific volume type will be created as a thin LUN) or a volume_type that specifies thin can be created and associated with the LUN.

Note: The minimum size for dedup and thin LUNs is 10GB.

The configuration option “san_thin_provision” is by default set to true. Not specifying it in cinder.conf will result in all LUNs created without a volume_type to be thin LUNs.

To create a volume_type that creates a thin volume, do the following:

1. The extra_capabilities configuration in cinder.conf should identify the backend to be thin-LUN capable. This is done as follows:

   ```
   extra_capabilities={"dedup":"True","thin":"True"}
   ```

   Note: The `extra_capabilities` value must match the `extra_specs` value in step 3.

(The line above shows the backend to be capable of “dedup” in addition to “thin”.)

2. A volume_type has to be created to be associated with a thin LUN:

```
stack@qa-dl380g8-n2:~$ cinder type-list
+--------------------------------------+-------+
|                  ID                  |  Name |
+--------------------------------------+-------+
| 53dd6932-e3dd-4f4d-ae17-e938f31f15a0 |  thin |
+--------------------------------------+-------+
stack@qa-dl380g8-n2:~$
```

3. The extra-specs for the volume_type created above should specify the preferred backend and the lun-type:

Note: The `extra_specs` value must match the `extra_capabilities` value in step 1.

```
stack@qa-dl380g8-n2:~$ cinder extra-specs-list
+--------------------------------------+-------+----------------------+
| ID                                   |  Name |          extra_specs |
+--------------------------------------+-------+----------------------+
| 53dd6932-e3dd-4f4d-ae17-e938f31f15a0 |  thin | {u'capabilities:thin': |
| u'True', u'volume_backend_name': u'ViolinMGA'}                       |
+--------------------------------------+-------+----------------------+
stack@qa-dl380g8-n2:~$
```

4. A thin LUN can now be created as follows:

```
stack@qa-dl380g8-n2:~$ cinder create --display-name test-thin-lun --
volume-type thin 10
```

Note: The minimum size for a thin LUN is 10GB.

Creating Dedup LUNs

Dedup LUNs can be created by using volume_type with extra-specs that identify the LUN type to be dedup.

In order to create a volume_type that creates a dedup volume, the following steps have to be performed.

1. The extra_capabilities configuration in cinder.conf should identify the backend to be dedup-LUN capable. This is done as follows:

```
extra_capabilities={"dedup":"True","thin":"True"}
```

2. A volume_type has to be created to be associated with the dedup LUN:

```
stack@qa-dl380g8-n2:~$ cinder type-list
+--------------------------------------+-------+
|                  ID                  |  Name |
+--------------------------------------+-------+
| f9388e2a-5c0f-466f-bca8-1f9730bad7e3 | dedup |
+--------------------------------------+-------+
stack@qa-dl380g8-n2:~$
```

3. The extra-specs for the volume_type created above should specify the preferred backend and the LUN-type:

```
stack@qa-dl380g8-n2:~$ cinder extra-specs-list
+--------------------------------------+-------+----------------------+
| ID                                   |  Name | extra_specs          |
+--------------------------------------+-------+----------------------+
| f9388e2a-5c0f-466f-bca8-1f9730bad7e3 | dedup | {u'volume_backend_name':
| u'ViolinMGA', u'capabilities:dedup': u'True'}                        |
+--------------------------------------+-------+----------------------+
stack@qa-dl380g8-n2:~$
```

4. A dedup LUN can now be created as follows:

```
stack@qa-dl380g8-n2:~$ cinder create --display-name test-dedup-lun --
volume-type dedup 10
```

Note: The minimum size for dedup LUN is 10GB.

Creating Thick LUNs

Thick LUNs can be created by explicitly setting the “san_thin_provision” configuration parameter in cinder.conf to false. With that set, all LUNs created without a volume-type will be thick LUNs. There is no extra-spec defined to specifically create thick LUNs.

Extending a LUN

The Violin Cinder driver 4.0 allows you to increase the size of a LUN using the cinder extend command. However, the LUN size cannot be reduced. Also note that the size of a dedup LUN cannot be extended after it has been created. Therefore, you must estimate the size requirement of a dedup LUN at the time of creation.

Managing Storage Pools

The *7300 Flash Storage Platform Best Practices Guide* makes some recommendations regarding storage pools for the 7000 Series FSP and the 7700 FSP controllers.

Automatic Pool Selection

Violin Cinder Driver 4.0 automatically selects the storage pool for creating a LUN. But to do so on a 7700 FSP controller, cinder.conf must list the storage pools configured on the backend storage devices. A storage pool configuration is not needed in cinder.conf for a 7000 Series FSP. However, it is recommended that storage pools be configured.

The needed storage pool configurations are:

- dedup_only_pools
- dedup_capable_pools

Example:

dedup_only_pools = PoolA, PoolB, PoolC

dedup_capable_pools = PoolD, PoolE

pool_allocation_method = "smallest" (This is optional)

User-specified Pool

You can specifically choose a storage pool for LUN creation instead of the Violin Cinder Driver choosing it automatically. This is done by associating a volume_type with the LUN that has extra_specs that specify a violin specific extra_spec known as storage_pool.

Note: Although this option allows you to select a particular pool to set up a LUN, you are strongly encouraged to use the Automatic Pool Selection to select a pool for LUN creation. See Automatic Pool Selection on page 16 for details.

The extra_spec is created as shown below:

```
stack@lab-srv3506:~$ cinder type-list
+--------------------------------------+-------------------+
|                  ID                  |        Name       |
+--------------------------------------+-------------------+
| dba897bf-e3a1-434d-b22c-ff3c009465d0 |      mga-pool     |
+--------------------------------------+-------------------+
stack@lab-srv3506:~$ cinder extra-specs-list
+--------------------------------------+-------------------+-----------+
| ID                                   |       Name        | extra_specs
+--------------------------------------+-------------------+-----------+
| dba897bf-e3a1-434d-b22c-ff3c009465d0 |      mga-pool     |
{u'violin:storage_pool': u'StoragePoolMga-1'}              |
+--------------------------------------+-------------------+-----------+
stack@lab-srv3506:~$
```

A LUN can then be created on this storage pool as show below:

```
stack@lab-srv3506:~$ cinder create --volume-type mga-pool --display-name
pool-lun
```

CHAPTER 3 Best Practices

Configuring Backend Targets with 7000 Series FSP (Standalone)

In a typical 7000 Series FSP standalone deployment, a single storage pool will be defined on Memory Gateway A (MG-B). Only a single backend target will need to be configured in this case with a single storage pool. In some cases, a storage pool may also be defined on MG-B. In this case, refer to the multiple backend configuration example in Multi-backend Configuration on page 12.

Note: It is not necessary to define a 7000 Series FSP as a backend target if it is being used as a storage shelf in a 7700 FSP deployment. Only the 7700 FSP controllers will need to be defined as targets for cinder in this case.

The following is an example configuration.

Example (cinder.conf):

```
san_thin_provision=false
volume_driver=cinder.volume.drivers.violin.v7000_fcp.V70
00FCPDriver
volume_backend_name=ViolinMGA
extra_capabilities={"dedup":"True","thin":"True"}
use_multipath_for_image_xfer=true
san_ip = <External IP of MG-A>
san_login = root
san_password= ViolinMEM1
```

Configuring Multiple Backend Targets with 7700 (HA Configuration)

In a typical 7700 deployment, there are two controllers in a HA pair. Multiple backend targets will need to be configured.

The following is an example configuration (cinder.conf) and example Cinder CLI commands to configure two backend 7700 FSP controllers.

```
enabled_backends= Controller_1, Controller_2
use_multibackend=True

[Controller_1]
volume_driver=cinder.volume.drivers.violin.v7000_fcp.V7000FCPDriver
volume_backend_name=violin-7700-Controller_1
extra_capabilities={"dedup":"True","thin":"True"}
use_multipath_for_image_xfer=true
san_ip=<External IP of 7700 Controller 1>
san_login=root
san_password=ViolinMEM1
san_thin_provision=False

[Controller_2]
volume_driver=cinder.volume.drivers.violin.v7000_fcp.V7000FCPDriver
volume_backend_name=violin-7700-Controller_2
extra_capabilities={"dedup":"True","thin":"True"}
use_multipath_for_image_xfer=true
san_ip=<External IP of 7700 Controller 2>
san_login=root
san_password=ViolinMEM1
san_thin_provision=False
```

Also see: https://wiki.openstack.org/wiki/Cinder-multi-backend

Multipath Recommendations with 7000 Series/ 7700

The following multipathing configuration is recommended for Linux based OpenStack compute nodes utilizing DM-Multipath with 7700 FSP controllers or 7000 Series FSPs.

The 'user_friendly_names' parameter in multipath.conf must be set to 'no' for proper volume attachment/detachment to instances.

The following configuration file is an example for Ubuntu 14.04.

Example (multipath.conf):

```
defaults {
  user_friendly_names no
}

devices {
   device {
        vendor                  "VIOLIN"
        product                 "CONCERTO ARRAY"
        path_selector           "round-robin 0"
        path_grouping_policy    multibus
      getuid_callout        "/lib/udev/scsi_id -p 0x80 -g -u /dev/%n"
        prio                    alua
        path_checker            tur
        rr_min_io               100
        rr_weight               priorities
        failback                immediate
        features                "1 queue_if_no_path"
        no_path_retry           300
   }
}
```

Volume Types for Dedup and Thin Volumes

Note: The minimum size for dedup and thin LUNs is 10GB.

7000 Series (Standalone)

The recommended cinder configuration file documented in the installation guide will default to thick volumes for volume creation if no volume type is explicitly defined. If deduplication or thin volume types are needed, 'dedup' and 'thin' volume types should be added.

Example CLI commands:

```
cinder type-create dedup

cinder type-key dedup set volume_backend_name=ViolinMGA
capabilities:dedup="True"

cinder type-create thin

cinder type-key thin set volume_backend_name=ViolinMGA
capabilities:thin="True"
```

```
cinder type-list

+--------------------------------------+-------+
|                  ID                  |  Name |
+--------------------------------------+-------+
| 67d6ad6a-5a39-42d1-b6e4-d8bc46e965ff |  thin |
| 90125a24-ab18-404a-b423-aac4c20a7b56 | dedup |
+--------------------------------------+-------+

cinder extra-specs-list
+--------------------------------------------------------------------------+
| ID                          | Name |          extra_specs |
+--------------------------------------------------------------------------+
| 67d6ad6a-5a39-42d1-b6e4-d8bc46e965ff |  thin | {u'capabilities:thin': u'True',
u'volume_backend_name': u'ViolinMGA'}
| 90125a24-ab18-404a-b423-aac4c20a7b56 | dedup | {u'volume_backend_name':
u'ViolinMGA',   u'capabilities:dedup': u'True'}
+--------------------------------------------------------------------------+
```

7700 HA Configuration

For a 7700 deployment, the extra capability must be set to Thin to specify dedup volume type and a specific storage pool.

Example:

```
cinder type-key dedup set volume_backend_name=violin-7700-Controller_1
violin:storage_pool=dedup-pool_Controller_1 capabilities:thin=True

$ cinder extra-specs-list
ID     Name    extra_specs
+--------------------------------------------------------------------------+
xxxxx  dedup   | {u'violin:storage_pool': u'dedup-pool-Controller_1',
u'capabilities:thin': u'True', u'volume_backend_name': u'violin-7700-
Controller_1'} |
```

Default Storage Pool Selection

The following cinder.conf variables are mandatory if connecting to a 7700 FSP controller.

```
dedup_only_pools = dedup_pool1, dedup_pool2, dedup_pool3
dedup_capable_pools = mixed_pool1, mixed_pool2, mixed_pool3
```

The "dedup_only_pools" is for storage pools that only have dedup functionality. The "dedup_capable_pools" is for pools that support dedup, thick and thin volumes. Any pool that does not appear in either of these two variables is assumed to support only thick and thin volume types.

If connected to a 7000 Series FSP, these variables are ignored and the following behavior is assumed:

- All storage pools on MG-A are assumed to be "dedup_capable_pools"
- All storage pools on MG-B are treated as thick/thin only storage pools

Finally, the method for selecting a storage pool can also be configured, as another cinder.conf variable:

```
pool_allocation_method = random
```

The following methods are supported:

- random: Choose a random pool that meets the volume type/size criteria.
- largest: Choose the pool with the largest amount of available space that meets the volume type/size criteria.
- smallest: Choose the pool with the smallest amount of available space that meets the volume type/size criteria.

Index

W