VAST Driver for Cinder

22 April 2025

Always check support.vastdata.com for the latest

Contents

VAST Driver for Cinder

VAST Driver for Cinder enables you to use VAST NVMe storage as a backend for data volumes managed through the OpenStack Block Storage service (Cinder).

The VAST driver for Cinder supports volume operations shown in Supported Operations and Usage Examples.

To deploy the VAST driver for Cinder, complete these steps:

- Ensure that prerequisites are met.
- Set up the OpenStack host.
- Configure the VAST cluster so that Cinder can use it as the backend.
- Configure Cinder to use the VAST cluster as the backend.

If you encounter any issues, refer to troubleshooting guidelines in Troubleshooting.

Prerequisites

Support of block storage (NVMe over TCP) has been introduced in VAST release 5.3. Your VAST cluster must be running VAST release 5.3 or later so that it allows configuring VAST Element Store views that support the block access protocol.

Set up the OpenStack Host

- **Networking**: Ensure that your OpenStack has access to the access network (e.g. the Web UI address) and the data network (e.g. protocol virtual IP pool address) of the VAST cluster.
- **Permissions**:
  - root or sudo permissions on the OpenStack host
  - admin permission to the OpenStack Web UI (Horizon)
- Install **NVMe CLI**:

  ```
  sudo yum install nvme-cli
  ```

- Load kernel modules required to enable NVMe over Fabrics:

  ```
  sudo modprobe nvme
  sudo modprobe nvme-fabrics
  ```

Configure VAST Cluster for Cinder

Note

The VAST cluster must be running release 5.3 or later to support block access.

Complete the following steps to make your VAST cluster ready for integration with Kubernetes:

1. **Create a tenant** for Cinder block volumes.

   VAST strongly recommends creating a new tenant dedicated to storing Cinder volumes. Since the VAST driver requires a VMS manager account to automate provisioning via API calls, it is highly recommended to isolate this account from being able to impact data and configuration that might otherwise exist on the same tenant. Even if every other use case on the cluster is using the default tenant, creating a Cinder-dedicated tenant is strongly recommended.

2. Under the tenant created in step 1, **create a view** that will expose the NVMe subsystem for Cinder.

   The view must have the block storage protocol enabled (Protocols = *Block*) and an NVMe subsystem defined.

   To view and manage views in VAST Web UI, log in and choose Element Store -> Views in the main navigation menu. For more information about VAST Cluster views, see *VAST Cluster Administrator's Guide*.

3. Ensure that the cluster provides a **virtual IP pool** to be used by the VAST driver.

   You can create a new virtual IP pool or use an existing one. The virtual IP pool must be explicitly associated with the same tenant (created in step 1) as the NVMe subsystem (block view) created in step 2.

   Since you'll have to specify the virtual IP pool by its name, ensure that the VAST cluster has DNS configured, and the virtual IP pool has *Virtual IP Pool Domain Name* defined in its settings.

   To view and manage virtual IP pools in VAST Web UI, log in and choose Network Access -> Virtual IP Pools in the main navigation menu. For more information about VAST Cluster virtual IP pools, see *VAST Cluster Administrator's Guide*.

4. **Create a role** to be assigned to a service account (VMS manager user that you'll create in step 5) the VAST driver will use to communicate with VMS. The role must be created in the tenant dedicated to storing Cinder volumes. The role must grant the Create, View, Edit and Delete permissions in the Logical realm.

   To view and manage VMS manager roles in VAST Web UI, log in and choose Administrators -> Roles in the main navigation menu.

5. **Create a VMS manager user** to be used by the VAST driver to communicate with the VAST Management Service (VMS) via VAST REST API. You'll need to supply, in the Cinder configuration file, the VMS user credentials (username and password) or VMS API token generated for that user.

   While the VAST driver can use the `admin` account, it is strongly recommended to create a dedicated VMS manager user account for the driver. This account must be assigned a role associated with the tenant that will store the Cinder volumes. **Assign the role** created in step 4 to the VMS manager user.

   To view and manage VMS manager users in VAST Web UI, log in and choose Administrators -> Managers in the main navigation menu.

   To generate and manage VMS API tokens for a VMS manager user, run the VAST CLI's `apitoken *` commands.

   For more information about managing VMS manager users, see *VAST Cluster Administrator's Guide*.

Configure Cinder to Use VAST Backend

1. Edit the Cinder configuration file (`/etc/cinder/cinder.conf`) as follows:

   - In the `[DEFAULT]` section, specify a name for the new backend on the `enabled_backends` parameter. Below the name `vast` is specified:

     ```
     [DEFAULT]
     enabled_backends = vast
     ```

If the parameter lists multiple backends, separate the values with commas.

- Create a new backend section for the newly enabled backend (`vast`) and specify the following options:

```
[vast]
volume_driver = <driver path>
volume_backend_name = <backend name>
san_ip = <VMS IP>
[san_api_port = <VMS port>]
{
  san_login = <VMS user name>
  san_password = <VMS user password>
  |
  vast_api_token = <VMS API token>
  }
vast_vippool_name = <vip_pool>
vast_subsystem = <NVMe subsystem>
```

Where:

- `volume_driver` sets the driver path. Specify it as `cinder.volume.drivers.vastdata.driver.VASTVolumeDriver`.
- `volume_backend_name` sets the backend name (e.g. `vast`).
- `san_ip` sets the IP address of the VAST Management Service (VMS) running on the VAST cluster.
- `san_api_port` specifies the port where the VMS is listening. This parameter is optional. If not specified, port 443 is used.
- `san_login` is the user name of the VMS manager user that the driver will use to communicate with the VAST cluster's VMS. By default, the `admin` user is used. This parameter is only required if no VMS API token is supplied. If you have `vast_api_token` specified, this parameter is ignored.
- `san_password` is the VMS manager user password. This parameter is only required if no VMS API token is supplied. If you have `vast_api_token` specified, this parameter is ignored.
- `vast_api_token` specifies the VMS API token used to authenticate and authorize driver access to the VAST cluster. This parameter is required if no VMS manager credentials are supplied.
- `vast_vippool_name` specifies the name of the virtual IP pool configured for block storage on the VAST cluster.
- `vast_subsystem` is the name of the NVMe subsystem exposed by the VAST cluster.

For example:

```
[vast]
volume_driver = cinder.volume.drivers.vastdata.driver.VASTVolumeDriver
volume_backend_name = vast
san_ip = 10.27.200.136
san_login = admin
san_password = 123456
vast_vippool_name = vip-pool1
vast_subsystem = cinder-test
```

2. Restart the Cinder service to apply configuration changes:

```
systemctl restart openstack-cinder-*.service
```

Supported Operations and Usage Examples

The driver supports the following operations:

- Create, list, delete volumes
- Attach (map) and detach (unmap) volumes
- Create, list, delete volume snapshots
- Restore a volume from a snapshot
- Clone a volume
- Extend a volume

For example:

- To create a new volume type named `vast`:

  ```
  openstack volume type create vast
  ```

- To create a volume named `vast-vol` of type `vast`:

  ```
  openstack volume create --size 100 --type vast vast-vol
  ```

- To attach volume `1234abcd-5678-90ef-1234-567890abcdef` to instance `a1b2c3d4-e5f6-7890-abcd-ef1234567890`:

  ```
  openstack server add volume <server-id> <volume-id>
  ```

- To extend volume `9f8e7d6c-5b4a-3210-fedc-ba9876543210`:

  ```
  openstack volume set --size 200 9f8e7d6c-5b4a-3210-fedc-ba9876543210
  ```

- To view details of volume `1234abcd-5678-90ef-1234-567890abcdef`:

  ```
  openstack volume show 1234abcd-5678-90ef-1234-567890abcdef
  ```

- To create a snapshot named `test-snapshot` of volume `1234abcd-5678-90ef-1234-567890abcdef`:

  ```
  openstack volume snapshot create --volume 1234abcd-5678-90ef-1234-567890abcdef test-snapshot
  ```

- To delete snapshot `test-snapshot`:

  ```
  openstack volume snapshot delete test-snapshot
  ```

- To set volume `1234abcd-5678-90ef-1234-567890abcdef` to *Active* state:

  ```
  openstack volume set --state available abcdef12-3456-7890-abcd-ef1234567890
  ```

Note

Perform this operation only when the volume is in a bad state and no actions can be performed on it.

Troubleshooting

Cinder logs are saved under `/var/log/cinder/` but not all actions are logged under `cinder`. Disk attach/detach operations are performed by the OpenStack Compute Service (Nova).

it is recommended to start troubleshooting with `/var/log/cinder/volume.log`. When hitting volumes attachments issues, investigate `/var/log/nova/nova-compute.log`.

- **volume.log** captures information related to the core volume operations. It records events such as volume creation, deletion, attaching/detaching volumes to instances, and any errors encountered during these operations. This log is critical for troubleshooting volume-related issues, such as slow volume creation, failed volume operations, and issues with specific storage drivers.
- **api.log** captures information related to the Cinder API service. It records requests received by the API, responses sent to clients, and any exceptions or errors encountered during API processing. This log is crucial for debugging API-related issues, such as invalid requests, authentication problems, and authorization errors.
- **backup.log** specifically tracks activities related to volume backups. It records backup requests, their progress, successful completions, and any failures encountered during the backup process. This log is essential for troubleshooting backup issues, such as failed backups, slow backup performance, and issues with specific backup drivers.
- **cinder-manage.log** captures output from the `cinder-manage` command-line tool. The `cinder-manage` tool is used for various administrative tasks, such as database upgrades, volume migrations, and running checks on the Cinder service. This log provides valuable insights into the execution of these administrative tasks, including any errors or warnings encountered.
- scheduler.log records the activities of the Cinder scheduler. The scheduler is responsible for selecting the appropriate backend (e.g. storage driver) to host a newly created volume. This log tracks the scheduling decisions made by the scheduler, including the criteria used for selecting backends, any scheduling failures, and any conflicts encountered during the scheduling process.


VAST