NetApp Docker Volume Plugin Documentation

Release

NetApp

Jan 25, 2018

Contents:

The NetApp Docker Volume Plugin (nDVP) provides direct integration with the Docker ecosystem for NetApp's ONTAP, SolidFire, and E-Series storage platforms. The nDVP package supports the provisioning and management of storage resources from the storage platform to Docker hosts, with a robust framework for adding additional platforms in the future.

Multiple instances of the nDVP can run concurrently on the same host. This allows simultaneous connections to multiple storage systems and storage types, with the ablity to customize the storage used for the Docker volume(s).

CHAPTER 1

Quick Start

This quick start is targeted at the Docker Managed Plugin method (Docker >= 1.13 / 17.03). If you're using an earlier version of Docker, please refer to the documentation: *Host Configuration*.

1. nDVP is supported on the following operating systems:
   - Debian
   - Ubuntu, 14.04+ if not using iSCSI multipathing, 15.10+ with iSCSI multipathing.
   - CentOS, 7.0+
   - RHEL, 7.0+
2. Verify your storage system meets the minimum requirements:
   - ONTAP: 8.3 or greater
   - SolidFire: ElementOS 7 or greater
   - E-Series: Web Services Proxy
3. Ensure you have Docker Engine 17.03 (nee 1.13) or above installed.

   ```
   docker --version
   ```

   If your version is out of date, follow the instructions for your distribution to install or update.
4. Verify that the protocol prerequesites are installed and configured on your host. See *Host Configuration*.
5. Create a configuration file. The default location is `/etc/netappdvp/config.json`. Be sure to use the correct options for your storage system.

   ```
   # create a location for the config files
   sudo mkdir -p /etc/netappdvp

   # create the configuration file, see below for more configuration examples
   cat << EOF > /etc/netappdvp/config.json
   {
       "version": 1,
   ```

```
    "storageDriverName": "ontap-nas",
    "managementLIF": "10.0.0.1",
    "dataLIF": "10.0.0.2",
    "svm": "svm_nfs",
    "username": "vsadmin",
    "password": "netapp123",
    "aggregate": "aggr1"
}
EOF
```

6. Start nDVP using the managed plugin system.

```
docker plugin install netapp/ndvp-plugin:17.07 --alias netapp --grant-all-
↪permissions
```

7. Begin using nDVP to consume storage from the configured system.

```
# create a volume named "firstVolume"
docker volume create -d netapp --name firstVolume

# create a default volume at container instantiation
docker run --rm -it --volume-driver netapp --volume secondVolume:/my_vol alpine␣
↪ash

# remove the volume "firstVolume"
docker volume rm firstVolume
```

CHAPTER 2

Installing nDVP

The first step to installing the NetApp Docker Volume Plugin is to ensure that your host is configured for the protocol you intend to use, NFS or iSCSI. Once that is complete you will want to create a configuration file which details the nDVP instance configuration, and finally instantiate the daemon on the host.

Follow the directions in the *Quick Start* for downloading the nDVP binary and installing it to your host.

2.1 Host Configuration

2.1.1 Prerequesites

- nDVP is supported on the following operating systems:
  - Debian
  - Ubuntu, 14.04+ if not using iSCSI multipathing, 15.10+ with iSCSI multipathing.
  - CentOS, 7.0+
  - RHEL, 7.0+
- Verify your storage system meets the minimum requirements:
  - ONTAP: 8.3 or greater
  - SolidFire: ElementOS 7 or greater
  - E-Series: Santricity

2.1.2 NFS

Install the following system packages:

- RHEL / CentOS

```
sudo yum install -y nfs-utils
```

- Ubuntu / Debian

```
sudo apt-get install -y nfs-common
```

2.1.3 iSCSI

- RHEL / CentOS

  1. Install the following system packages:

  ```
  sudo yum install -y lsscsi iscsi-initiator-utils sg3_utils device-mapper-
  ↪multipath
  ```

  2. Start the multipathing daemon:

  ```
  sudo mpathconf --enable --with_multipathd y
  ```

  3. Ensure that *iscsid* and *multipathd* are enabled and running:

  ```
  sudo systemctl enable iscsid multipathd
  sudo systemctl start iscsid multipathd
  ```

  4. Discover the iSCSI targets:

  ```
  sudo iscsiadm -m discoverydb -t st -p <DATA_LIF_IP> --discover
  ```

  5. Login to the discovered iSCSI targets:

  ```
  sudo iscsiadm -m node -p <DATA_LIF_IP> --login
  ```

  6. Start and enable `iscsi`:

  ```
  sudo systemctl enable iscsi
  sudo systemctl start iscsi
  ```

- Ubuntu / Debian

  1. Install the following system packages:

  ```
  sudo apt-get install -y open-iscsi lsscsi sg3-utils multipath-tools scsitools
  ```

  2. Enable multipathing:

  ```
  sudo tee /etc/multipath.conf <<-'EOF'
  defaults {
      user_friendly_names yes
      find_multipaths yes
  }
  EOF

  sudo service multipath-tools restart
  ```

  3. Ensure that `iscsid` and `multipathd` are running:

```
sudo service open-iscsi start
sudo service multipath-tools start
```

4. Discover the iSCSI targets:

```
sudo iscsiadm -m discoverydb -t st -p <DATA_LIF_IP> --discover
```

5. Login to the discovered iSCSI targets:

```
sudo iscsiadm -m node -p <DATA_LIF_IP> --login
```

2.1.4 Traditional Install Method (Docker <= 1.12)

1. Ensure you have Docker version 1.10 or above

```
docker --version
```

If your version is out of date, update to the latest.

```
curl -fsSL https://get.docker.com/ | sh
```

Or, follow the instructions for your distribution.

2. After ensuring the correct version of Docker is installed, install and configure the NetApp Docker Volume Plugin. Note, you will need to ensure that NFS and/or iSCSI is configured for your system. See the installation instructions below for detailed information on how to do this.

```
# download and unpack the application
wget https://github.com/NetApp/netappdvp/releases/download/v17.07.0/netappdvp-17.
↪07.0.tar.gz
tar zxf netappdvp-17.07.0.tar.gz

# move to a location in the bin path
sudo mv netappdvp /usr/local/bin
sudo chown root:root /usr/local/bin/netappdvp
sudo chmod 755 /usr/local/bin/netappdvp

# create a location for the config files
sudo mkdir -p /etc/netappdvp

# create the configuration file, see below for more configuration examples
cat << EOF > /etc/netappdvp/ontap-nas.json
{
    "version": 1,
    "storageDriverName": "ontap-nas",
    "managementLIF": "10.0.0.1",
    "dataLIF": "10.0.0.2",
    "svm": "svm_nfs",
    "username": "vsadmin",
    "password": "netapp123",
    "aggregate": "aggr1"
}
EOF
```

3. After placing the binary and creating the configuration file(s), start the nDVP daemon using the desired configuration file.

Note: Unless specified, the default name for the volume driver will be "netapp".

```
sudo netappdvp --config=/etc/netappdvp/ontap-nas.json
```

4. Once the daemon is started, create and manage volumes using the Docker CLI interface.

```
docker volume create -d netapp --name ndvp_1
```

Provision Docker volume when starting a container:

```
docker run --rm -it --volume-driver netapp --volume ndvp_2:/my_vol alpine ash
```

Destroy docker volume:

```
docker volume rm ndvp_1
docker volume rm ndvp_2
```

Starting nDVP at System Startup

A sample unit file for systemd based systems can be found at `contrib/netappdvp.service.example` in the git repo. To use the file, with CentOS/RHEL:

```
# copy the file to the correct location.  you must use unique names for the
# unit files if you have more than one instance running
cp contrib/netappdvp.service.example /usr/lib/systemd/system/netappdvp.service

# edit the file, change the description (line 2) to match the driver name and the
# configuration file path (line 9) to reflect your envrionment.

# reload systemd for it to ingest changes
systemctl daemon-reload

# enable the service, note this name will change depending on what you named the
# file in the /usr/lib/systemd/system directory
systemctl enable netappdvp

# start the service, see note above about service name
systemctl start netappdvp

# view the status
systemctl status netappdvp
```

Note that anytime the unit file is modified you will need to issue the command `systemctl daemon-reload` for it to be aware of the changes.

2.1.5 Docker Managed Plugin Method (Docker >= 1.13 / 17.03)

Note: If you have used nDVP pre-1.13/17.03 in the traditional daemon method, please ensure that you stop the nDVP process and restart your Docker daemon before using the managed plugin method.

```
# stop all running instances
pkill /usr/local/bin/netappdvp

# restart docker
systemctl restart docker
```

nDVP Specific Plugin Startup Options

- `config` - Specify the configuration file the plugin will use. Only the file name should be specified, e.g. `gold.json`, the location must be `/etc/netappdvp` on the host system. The default is `config.json`.
- `log-level` - Specify the logging level (`debug`, `info`, `warn`, `error`, `fatal`). The default is `info`.
- `debug` - Specify whether debug logging is enabled. Default is false. Overrides log-level if true.

Installing the Managed Plugin

1. Ensure you have Docker Engine 17.03 (nee 1.13) or above installed.

   ```
   docker --version
   ```

   If your version is out of date, follow the instructions for your distribution to install or update.

2. Create a configuration file. The config file must be located in the `/etc/netappdvp` directory. The default filename is `config.json`, however you can use any name you choose by specifying the `config` option with the file name. Be sure to use the correct options for your storage system.

   ```
   # create a location for the config files
   sudo mkdir -p /etc/netappdvp

   # create the configuration file, see below for more configuration examples
   cat << EOF > /etc/netappdvp/config.json
   {
       "version": 1,
       "storageDriverName": "ontap-nas",
       "managementLIF": "10.0.0.1",
       "dataLIF": "10.0.0.2",
       "svm": "svm_nfs",
       "username": "vsadmin",
       "password": "netapp123",
       "aggregate": "aggr1"
   }
   EOF
   ```

3. Start nDVP using the managed plugin system.

   ```
   docker plugin install --grant-all-permissions --alias netapp netapp/ndvp-
   ↪plugin:17.07 config=myConfigFile.json
   ```

4. Begin using nDVP to consume storage from the configured system.

   ```
   # create a volume named "firstVolume"
   docker volume create -d netapp --name firstVolume

   # create a default volume at container instantiation
   docker run --rm -it --volume-driver netapp --volume secondVolume:/my_vol alpine␣
   ↪ash

   # remove the volume "firstVolume"
   docker volume rm firstVolume
   ```

2.2 Global Configuration

These configuration variables apply to all nDVP configurations, regardless of the storage platform being used.

Option	Description	Example
`version`	Config file version number	1
`storageDriverName`	`ontap-nas`, `ontap-nas-economy`, `ontap-san`, `eseries-iscsi`, or `solidfire-san`	ontap-nas
`storagePrefix`	Optional prefix for volume names. Default: "netappdvp_"	netappdvp_

Also, default option settings are available to avoid having to specify them on every volume create. The `size` option is available for all controller types. See the ONTAP config section for an example of how to set the default volume size.

Defaults Option	Description	Example
`size`	Optional default size for new volumes. Default: "1G"	10G

Storage Prefix

A new config file variable has been added in v1.2 called "storagePrefix" that allows you to modify the prefix applied to volume names by the plugin. By default, when you run *docker volume create*, the volume name supplied is prepended with "netappdvp_" *("netappdvp-" for SolidFire)*.

If you wish to use a different prefix, you can specify it with this directive. Alternatively, you can use *pre-existing* volumes with the volume plugin by setting `storagePrefix` to an empty string, "".

SolidFire specific recommendation do not use a storagePrefix (including the default). By default the SolidFire driver will ignore this setting and not use a prefix. We recommend using either a specific tenantID for docker volume mapping or using the attribute data which is populated with the docker version, driver info and raw name from docker in cases where any name munging may have been used.

A note of caution: *docker volume rm* will *delete* these volumes just as it does volumes created by the plugin using the default prefix. Be very careful when using pre-existing volumes!

2.2.1 Known Issues and Limitations

1. Volume names must be a minimum of 2 characters in length

   This is a Docker client limitation. The client will interpret a single character name as being a Windows path. See bug 25773.

#. Because Docker Swarm does not orchestrate volume creation across multiple nodes, only the ontap-nas and ontap-san drivers will work in Swarm.

2.3 ONTAP Configuration

2.3.1 User Permissions

nDVP does not need full permissions on the ONTAP cluster and should not be used with the cluster-level admin account. Below are the ONTAP CLI comands to create a dedicated user for nDVP with specific permissions.

```
# create a new nDVP role
security login role create -vserver [VSERVER] -role ndvp_role -cmddirname DEFAULT -
↪access none

# grant common nDVP permissions
```

```
security login role create -vserver [VSERVER] -role ndvp_role -cmddirname "event␣
↪generate-autosupport-log" -access all
security login role create -vserver [VSERVER] -role ndvp_role -cmddirname "network␣
↪interface" -access readonly
security login role create -vserver [VSERVER] -role ndvp_role -cmddirname "version" -
↪access readonly
security login role create -vserver [VSERVER] -role ndvp_role -cmddirname "vserver" -
↪access readonly
security login role create -vserver [VSERVER] -role ndvp_role -cmddirname "vserver␣
↪nfs show" -access readonly
security login role create -vserver [VSERVER] -role ndvp_role -cmddirname "volume" -
↪access all
security login role create -vserver [VSERVER] -role ndvp_role -cmddirname "snapmirror
↪" -access all

# grant ontap-san nDVP permissions
security login role create -vserver [VSERVER] -role ndvp_role -cmddirname "vserver␣
↪iscsi show" -access readonly
security login role create -vserver [VSERVER] -role ndvp_role -cmddirname "lun" -
↪access all

# grant ontap-nas-economy nDVP permissions
security login role create -vserver [VSERVER] -role ndvp_role -cmddirname "vserver␣
↪export-policy create" -access all
security login role create -vserver [VSERVER] -role ndvp_role -cmddirname "vserver␣
↪export-policy rule create" -access all

# create a new nDVP user with nDVP role
security login create -vserver [VSERVER] -username ndvp_user -role ndvp_role -
↪application ontapi -authmethod password
```

2.3.2 Configuration File Options

In addition to the global configuration values above, when using ONTAP these top level options are available.

Option	Description	Example
`managementLIF`	IP address of ONTAP management LIF	10.0.0.1
`dataLIF`	IP address of protocol lif; will be derived if not specified	10.0.0.2
`svm`	Storage virtual machine to use (req, if management LIF is a cluster LIF)	svm_nfs
`username`	Username to connect to the storage device	vsadmin
`password`	Password to connect to the storage device	netapp123
`aggregate`	Aggregate to use for provisioning; it must be assigned to the SVM	aggr1

For the ontap-nas and ontap-nas-economy drivers, an additional top level option is available. For NFS host configuration, see also: http://www.netapp.com/us/media/tr-4067.pdf

Option	Description	Example
`nfsMountOptions`	Fine grained control of NFS mount options; defaults to “-o nfsvers=3”	-o nfsvers=4

Also, when using ONTAP, these default option settings are available to avoid having to specify them on every volume create.

Defaults Option	Description	Example
`spaceReserve`	Space reservation mode; “none” (thin provisioned) or “volume” (thick)	none
`snapshotPolicy`	Snapshot policy to use, default is “none”	none
`splitOnClone`	Split a clone from its parent upon creation, defaults to “false”	false
`encryption`	Enable NetApp Volume Encryption, defaults to “false”	true
`unixPermissions`	NAS option for provisioned NFS volumes, defaults to “777”	777
`snapshotDir`	NAS option for access to the .snapshot directory, defaults to “false”	false
`exportPolicy`	NAS option for the NFS export policy to use, defaults to “default”	default
`securityStyle`	NAS option for access to the provisioned NFS volume, defaults to “unix”	mixed
`fileSystemType`	SAN option to select the file system type, defaults to “ext4”	xfs

2.3.3 Scaling Options

The ontap-nas and ontap-san drivers create an ONTAP FlexVol for each Docker volume. ONTAP supports up to 1000 FlexVols per cluster node with a cluster maximum of 12,000 FlexVols. If your Docker volume requirements fit within that limitation, the ontap-nas driver is the preferred NAS solution due to the additional features offered by FlexVols such as Docker-volume-granular snapshots and cloning.

If you need more Docker volumes than may be accommodated by the FlexVol limits, choose the ontap-nas-economy driver, which creates Docker volumes as ONTAP Qtrees within a pool of automatically managed FlexVols. Qtrees offer far greater scaling, up to 100,000 per cluster node and 2,400,000 per cluster, at the expense of some features. The ontap-nas-economy driver does not support Docker-volume-granular snapshots or cloning. The ontap-nas-economy driver is not currently supported in Docker Swarm, as Swarm does not orchestrate volume creation across multiple nodes.

To get advanced features and huge scale in the same environment, you can run multiple instances of the Docker Volume Plugin, with one using ontap-nas and another using ontap-nas-economy.

2.3.4 Example ONTAP Config Files

NFS Example for ontap-nas driver

```
{
    "version": 1,
    "storageDriverName": "ontap-nas",
    "managementLIF": "10.0.0.1",
    "dataLIF": "10.0.0.2",
    "svm": "svm_nfs",
    "username": "vsadmin",
    "password": "netapp123",
    "aggregate": "aggr1",
    "defaults": {
      "size": "10G",
      "spaceReserve": "none",
      "exportPolicy": "default"
    }
}
```

NFS Example for ontap-nas-economy driver

```
{
    "version": 1,
    "storageDriverName": "ontap-nas-economy",
    "managementLIF": "10.0.0.1",
    "dataLIF": "10.0.0.2",
    "svm": "svm_nfs",
    "username": "vsadmin",
    "password": "netapp123",
    "aggregate": "aggr1",
}
```

iSCSI Example for ontap-san driver

```
{
    "version": 1,
    "storageDriverName": "ontap-san",
    "managementLIF": "10.0.0.1",
    "dataLIF": "10.0.0.3",
    "svm": "svm_iscsi",
    "username": "vsadmin",
    "password": "netapp123",
    "aggregate": "aggr1"
}
```

2.4 SolidFire Configuration

In addition to the global configuration values above, when using SolidFire, these options are available.

Option	Description	Example
`Endpoint`	Ex. `https://<login>:<password>@<mvip>/json-rpc/<element-version>`	
`SVIP`	iSCSI IP address and port	10.0.0.7:3260
`TenantName`	SF Tenant to use (created if not found)	“docker”
`InitiatorIFace`	Specify interface when restricting iSCSI traffic to non-default interface	“default”
`Types`	QoS specifications	See below
`LegacyNamePrefix`	Prefix for upgraded nDVP installs	“netappdvp-“

The SolidFire driver does not support Docker Swarm.

LegacyNamePrefix If you used a version of nDVP prior to 1.3.2 and perform an upgrade with existing volumes, you’ll need to set this value in order to access your old volumes that were mapped via the `volume-name` method.

2.4.1 Example Solidfire Config File

```
{
    "version": 1,
    "storageDriverName": "solidfire-san",
    "Endpoint": "https://admin:admin@192.168.160.3/json-rpc/7.0",
    "SVIP": "10.0.0.7:3260",
    "TenantName": "docker",
    "InitiatorIFace": "default",
    "Types": [
        {
            "Type": "Bronze",
            "Qos": {
                "minIOPS": 1000,
                "maxIOPS": 2000,
                "burstIOPS": 4000
            }
        },
        {
            "Type": "Silver",
            "Qos": {
                "minIOPS": 4000,
                "maxIOPS": 6000,
                "burstIOPS": 8000
            }
        },
        {
            "Type": "Gold",
            "Qos": {
                "minIOPS": 6000,
                "maxIOPS": 8000,
                "burstIOPS": 10000
            }
        }
    ]
}
```

2.5 E-Series Configuration

In addition to the global configuration values above, when using E-Series, these options are available.

Option	Description	Example
`webProxyHostname`	Hostname or IP address of Web Services Proxy	localhost
`webProxyPort`	Port number of the Web Services Proxy (optional)	8443
`webProxyUseHTTP`	Use HTTP instead of HTTPS for Web Services Proxy (default = false)	true
`webProxyVerifyTLS`	Verify server's certificate chain and hostname (default = false)	true
`username`	Username for Web Services Proxy	rw
`password`	Password for Web Services Proxy	rw
`controllerA`	IP address of controller A	10.0.0.5
`controllerB`	IP address of controller B	10.0.0.6
`passwordArray`	Password for storage array if set	blank/empty
`hostDataIP`	Host iSCSI IP address (if multipathing just choose either one)	10.0.0.101
`poolNameSearchPattern`	Regular expression for matching storage pools available for nDVP volumes (default = .+)	docker.*
`hostType`	Type of E-series Host created by nDVP (default = linux_dm_mp)	linux_dm_mp
`accessGroupName`	Name of E-series Host Group to contain Hosts defined by nDVP (default = netappdvp)	DockerHosts

2.5.1 Example E-Series Config File

Example for eseries-iscsi driver

```
{
  "version": 1,
  "storageDriverName": "eseries-iscsi",
  "webProxyHostname": "localhost",
  "webProxyPort": "8443",
  "webProxyUseHTTP": false,
  "webProxyVerifyTLS": true,
  "username": "rw",
  "password": "rw",
  "controllerA": "10.0.0.5",
  "controllerB": "10.0.0.6",
  "passwordArray": "",
  "hostDataIP": "10.0.0.101"
}
```

2.5.2 E-Series Array Setup Notes

The E-Series Docker driver can provision Docker volumes in any storage pool on the array, including volume groups and DDP pools. To limit the Docker driver to a subset of the storage pools, set the `poolNameSearchPattern` in the configuration file to a regular expression that matches the desired pools.

When creating a docker volume you can specify the volume size as well as the disk media type using the `-o` option and the tags `size` and `mediaType`. Valid values for media type are `hdd` and `ssd`. Note that these are optional; if unspecified, the defaults will be a *1 GB* volume allocated from an *HDD pool*. An example of using these tags to create a 2 GiB volume from an SSD-based pool:

```
docker volume create -d netapp --name my_vol -o size=2G -o mediaType=ssd
```

The E-series Docker driver will detect and use any preexisting Host definitions without modification, and the driver will automatically define Host and Host Group objects as needed. The host type for hosts created by the driver defaults to `linux_dm_mp`, the native DM-MPIO multipath driver in Linux.

The current E-series Docker driver only supports iSCSI.

2.6 Multiple Instances of nDVP

Multiple instances of nDVP are needed when you desire to have multiple storage configurations available simultaneously. The key to multiple instances is to give them different names using the `--alias` option with the containerized plugin, or `--volume-driver` option when instantiating the nDVP driver on the host.

Docker Managed Plugin (Docker >= 1.13 / 17.03)

1. Launch the first instance specifying an alias and configuration file

```
docker plugin install --grant-all-permissions --alias silver netapp/ndvp-
↪plugin:17.07 config=silver.json
```

2. Launch the second instance, specifying a different alias and configuration file

```
docker plugin install --grant-all-permissions --alias gold netapp/ndvp-plugin:17.
↪07 config=gold.json
```

3. Create volumes specifying the alias as the driver name

```
# gold volume
docker volume create -d gold --name ntapGold
```

```
# silver volume
docker volume create -d silver --name ntapSilver
```

Traditional (Docker <=1.12)

1. Launch the plugin with an NFS configuration using a custom driver ID:

   ```
   sudo netappdvp --volume-driver=netapp-nas --config=/path/to/config-nfs.
   ↪json
   ```

2. Launch the plugin with an iSCSI configuration using a custom driver ID:

   ```
   sudo netappdvp --volume-driver=netapp-san --config=/path/to/config-iscsi.
   ↪json
   ```

3. Provision Docker volumes each driver instance:

   - NFS

   ```
   docker volume create -d netapp-nas --name my_nfs_vol
   ```

   - iSCSI

   ```
   docker volume create -d netapp-san --name my_iscsi_vol
   ```

CHAPTER 3

Using nDVP

Creating and consuming storage from ONTAP, SolidFire, and/or E-Series systems is easy with nDVP. Simply use the standard `docker volume` commands with the nDVP driver name specified when needed.

3.1 Volume Driver CLI Options

Each storage driver has a different set of options which can be provided at volume creation time to customize the outcome. Refer to the documentation below for your configured storage system to determine which options apply.

3.1.1 ONTAP Volume Options

Volume create options for both NFS and iSCSI:

- `size` - the size of the volume, defaults to 1 GiB
- `spaceReserve` - thin or thick provision the volume, defaults to thin. Valid values are `none` (thin provisioned) and `volume` (thick provisioned).
- `snapshotPolicy` - this will set the snapshot policy to the desired value. The default is `none`, meaning no snapshots will automatically be created for the volume. Unless modified by your storage administrator, a policy named "default" exists on all ONTAP systems which creates and retains six hourly, two daily, and two weekly snapshots. The data preserved in a snapshot can be recovered by browsing to the .snapshot directory in any directory in the volume.
- `splitOnClone` - when cloning a volume, this will cause ONTAP to immediately split the clone from its parent. The default is `false`. Some use cases for cloning volumes are best served by splitting the clone from its parent immediately upon creation, since there is unlikely to be any opportunity for storage efficiencies. For example, cloning an empty database can offer large time savings but little storage savings, so it's best to split the clone immediately.
- `encryption` - this will enable NetApp Volume Encryption (NVE) on the new volume, defaults to `false`. NVE must be licensed and enabled on the cluster to use this option.

NFS has two additional options that aren't relevant when using iSCSI:

- `unixPermissions` - this controls the permission set for the volume itself. By default the permissions will be set to `---rwxr-xr-x`, or in numerical notation `0755`, and root will be the owner. Either the text or numerical format will work.
- `snapshotDir` - setting this to `true` will make the .snapshot directory visible to clients accessing the volume. The default value is `false`, meaning that access to snapshot data is disabled by default. Some images, for example the official MySQL image, don't function as expected when the .snapshot directory is visible.
- `exportPolicy` - sets the export policy to be used for the volume. The default is `default`.
- `securityStyle` - sets the security style to be used for access to the volume. The default is `unix`. Valid values are `unix` and `mixed`.

iSCSI has an additional option that isn't relevant when using NFS:

- `fileSystemType` - sets the file system used to format iSCSI volumes. The default is `ext4`. Valid values are `ext3`, `ext4`, and `xfs`.

Using these options during the docker volume create operation is super simple, just provide the option and the value using the `-o` operator during the CLI operation. These override any equivalent vales from the JSON configuration file.

```
# create a 10GiB volume
docker volume create -d netapp --name demo -o size=10G -o encryption=true

# create a 100GiB volume with snapshots
docker volume create -d netapp --name demo -o size=100G -o snapshotPolicy=default

# create a volume which has the setUID bit enabled
docker volume create -d netapp --name demo -o unixPermissions=4755
```

The minimum volume size is 20MiB.

3.1.2 SolidFire Volume Options

The SolidFire driver options expose the size and quality of service (QoS) policies associated with the volume. When the volume is created, the QoS policy associated with it is specified using the `-o type=service_level` nomenclature.

The first step to defining a QoS service level with the SolidFire driver is to create at least one type and specify the minimum, maximum, and burst IOPS associated with a name in the configuration file.

Example SolidFire Configuration File with QoS Definitions

```
{
    "...": "..."
    "Types": [
        {
            "Type": "Bronze",
            "Qos": {
                "minIOPS": 1000,
                "maxIOPS": 2000,
                "burstIOPS": 4000
            }
        },
        {
            "Type": "Silver",
            "Qos": {
                "minIOPS": 4000,
                "maxIOPS": 6000,
                "burstIOPS": 8000
            }
        },
        {
            "Type": "Gold",
            "Qos": {
                "minIOPS": 6000,
                "maxIOPS": 8000,
                "burstIOPS": 10000
            }
        }
    ]
}
```

In the above configuration we have three policy definitions: *Bronze*, *Silver*, and *Gold*. These names are well known and fairly common, but we could have just as easily chosen: *pig*, *horse*, and *cow*, the names are arbitrary.

```
# create a 10GiB Gold volume
docker volume create -d solidfire --name sfGold -o type=Gold -o size=10G

# create a 100GiB Bronze volume
docker volume create -d solidfire --name sfBronze -o type=Bronze -o size=100G
```

3.1.3 Other SolidFire Create Options

Volume create options for SolidFire:

- `size` - the size of the volume, defaults to 1GiB or config entry `... "defaults": {"size": "5G"}`
- `blocksize` - use either `512` or `4096`, defaults to 512 or config entry `DefaultBockSize`

3.1.4 E-Series Volume Options

Media Type

The E-Series driver offers the ability to specify the type of disk which will be used to back the volume and, like the other drivers, the ability to set the size of the volume at creation time.

Currently only two values for `mediaType` are supported: `ssd` and `hdd`.

```
# create a 10GiB SSD backed volume
docker volume create -d eseries --name eseriesSsd -o mediaType=ssd -o size=10G

# create a 100GiB HDD backed volume
docker volume create -d eseries --name eseriesHdd -o mediaType=hdd -o size=100G
```

Pool

The user can specify the pool name to use for creating the volume.

`poolName` is optional, if no pool is specified, then the default is to use all pools available.

```
# create a volume using the "testme" pool
docker volume create -d eseries --name testmePoolVolume -o poolName=testme -o␣
↪size=100G
```

```
```

File System Type

The user can specify the file system type to use to format the volume. The default for `fileSystemType` is `ext4`. Valid values are `ext3`, `ext4`, and `xfs`.

```
# create a volume using xfs
docker volume create -d eseries --name xfsVolume -o fileSystemType=xfs
```

3.2 Create a Volume

```
# create a volume with an nDVP driver using the default name
docker volume create -d netapp --name firstVolume

# create a volume with a specific nDVP instance
docker volume create -d ntap_bronze --name bronzeVolume
```

If no options are specified, the defaults for the driver are used. The defaults are documented on the page for the storage driver you're using below.

The default volume size may be overridden per volume as follows:

```
# create a 20GiB volume with an nDVP driver
docker volume create -d netapp --name my_vol --opt size=20G
```

Volume sizes are expressed as strings containing an integer value with optional units (e.g. "10G", "20GB", "3TiB"). If no units are specified, the default is 'G'. Size units may be expressed either as powers of 2 (B, KiB, MiB, GiB, TiB) or powers of 10 (B, KB, MB, GB, TB). Shorthand units use powers of 2 (G = GiB, T = TiB, . . .).

3.3 Destroy a Volume

```
# destroy the volume just like any other Docker volume
docker volume rm firstVolume
```

3.4 Volume Cloning

When using the ontap-nas, ontap-san, and solidfire-san storage drivers, the Docker Volume Plugin can clone volumes.

```
# inspect the volume to enumerate snapshots
docker volume inspect <volume_name>

# create a new volume from an existing volume.  this will result in a new snapshot␣
↪being created
docker volume create -d <driver_name> --name <new_name> -o from=<source_docker_volume>

# create a new volume from an existing snapshot on a volume.  this will not create a␣
↪new snapshot
docker volume create -d <driver_name> --name <new_name> -o from=<source_docker_volume>
↪ -o fromSnapshot=<source_snap_name>
```

Here is an example of that in action:

```
[me@host ~]$ docker volume inspect firstVolume

[
    {
        "Driver": "ontap-nas",
        "Labels": null,
        "Mountpoint": "/var/lib/docker-volumes/ontap-nas/netappdvp_firstVolume",
        "Name": "firstVolume",
        "Options": {},
        "Scope": "global",
        "Status": {
            "Snapshots": [
                {
                    "Created": "2017-02-10T19:05:00Z",
                    "Name": "hourly.2017-02-10_1505"
                }
            ]
        }
    }
]

[me@host ~]$ docker volume create -d ontap-nas --name clonedVolume -o from=firstVolume
clonedVolume

[me@host ~]$ docker volume rm clonedVolume
[me@host ~]$ docker volume create -d ontap-nas --name volFromSnap -o from=firstVolume␣
↪-o fromSnapshot=hourly.2017-02-10_1505
volFromSnap

[me@host ~]$ docker volume rm volFromSnap
```

3.5 Access Externally Created Volumes

Externally created block devices (or their clones) may be accessed by containers using nDVP only if they have no partitions and if their filesystem is supported by nDVP (example: an ext4-formatted /dev/sdc1 will not be accessible via nDVP).

CHAPTER 4

Support

4.1 Troubleshooting

The plugin has been built with a comprehensive logging capability that should help you diagnose most of the issues you are likely to come across. The method you use to access or tune those logs varies based on how you are running the plugin.

If you are running nDVP using the recommended managed plugin method (i.e., using `docker plugin` commands), the plugin is running in a container and the logs are available inside. Accessing those logs requires a little detective work because plugin containers are hidden from `docker ps` output:

```
# find the plugin container's abbreviated ID
docker plugin ls

# find the plugin container's full ID
docker-runc list | grep <abbreviated ID>

# view the logs in the container
docker-runc exec -t <full ID> cat /var/log/netappdvp/netapp.log
```

The standard logging level should allow you to diagnose most issues. If you find that's not enough, you can enable debug logging:

```
# install the plugin with debug logging enabled
docker plugin install netapp/ndvp-plugin:<version> --alias <alias> debug=true

# or, enable debug logging on one that's already installed
docker plugin disable <plugin>
docker plugin set <plugin> debug=true
docker plugin enable <plugin>
```

If you are not running as a managed plugin, the logs are available in the host's `/var/log/netappdvp` directory. If you need to enable debug logging, specify `-debug` when you run the plugin.

4.2 Getting Help

The nDVP is a supported NetApp project. See the find the support you need landing page on the Support site for options available to you. To open a support case, use the serial number of the backend storage system and select containers and nDVP as the category you want help in.

There is also a vibrant discussion community of container users and engineers on the #containers channel of Slack. This can be a great place to get answers and discuss with like-minded peers; highly recommended!