Lenovo

Enabling Kubernetes on ThinkSystem DM Series and DE Series

Describes how to set up Kubernetes on DM Series using Trident CSI

Describes how to set up Kubernetes on DE Series without the use of Trident CSI

Lists the prerequisites for DM Series and DE Series

Provides additional instructions for setting up DE Series

Anthony Yu

LENOVO PRESS

Abstract

ThinkSystem DM Series and DE Series storage systems support containerized workloads that can be deployed with dynamic orchestration within a Kubernetes cluster. This document describes how set up a DM or DE Series storage system for storage provisioning and how to deploy the first container on a Kubernetes cluster.

The paper also describes how ThinkSystem DM supports the use of Trident. Trident is an open-source container storage interface (CSI) project maintained by NetApp that deploys in a Kubernetes cluster as a pod to provide dynamic storage orchestration within the cluster. The use of Trident streamlines the process for provisioning storage for pods.

Readers should be familiar with storage administration on DM Series or DE Series, Linux administration, and Kubernetes cluster management and deployment.

Contents

Introduction

This paper describes the procedures for getting started with a Kubernetes container using DM Series with Trident and DE Series without Trident. A quick containerized environment will be setup and can be expanded upon. Using Trident enables dynamic storage orchestration and quick provisioning of storage resources for Kubernetes clusters and is deployed in a Kubernetes cluster as a pod.

Note: Support of Trident with DE Series was deprecated in Trident v21.07, so it is not advised to use Trident with DE Series.

Setting up DM Series

This section describes the procedures to set up containers in Kubernetes clusters with DM Series storage. A server hosts the Kubernetes services and consumes DM storage. The DM Series controllers are not hosting or servicing Kubernetes services natively onboard the storage controllers.

It is assumed that a Kubernetes cluster has already been setup with master/worker nodes.

Components used in our lab environment:

- Ubuntu (21.04 LTS)
- Trident (v21.04; Server and Client)
- Kubernetes (v1.21.1; Server and Client)
- DM5000F (9.9.1X21)
- DM7000F (9.7P12)

Prerequisites

Nodes in the Kubernetes cluster should have the prerequisites:

1. Verify the following packages are installed (based on the desired protocol):

   Note: The Kubernetes cluster nodes should be rebooted after installing NFS or iSCSI tools

   – NFS Installed packages:

     - RHEL and CentOS: nfs-utils
     - Ubuntu: nfs-common

   – iSCSI Installed packages:

     - RHEL/CentOS: lsscsi, iscsi-initiator-utils, sg3_utils, device-mapper-multipath
     - Ubuntu: open-iscsi, lsscsi, sg3_utils, multipath-tools, scsitools

2. Enable multipathing (for iSCSI only):

   For RHEL and CentOS, use the following command:

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

Figure 1 RHEL and CentOS multipathing

For Ubuntu, use the following:

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


sudo systemctl enable multipath-tools.service
sudo service multipath-tools restart
```

Figure 2 Ubuntu multipathing

3. Verify the services are running and set them to autorun (for iSCSI only):

   For RHEL and CentOS:

```
sudo systemctl status iscsid multipathd iscsi
sudo systemctl enable iscsid multipathd iscsi
```

Figure 3 RHEL and CentOS services

   For Ubuntu:

```
sudo systemctl status multipath-tools open-iscsi
sudo systemctl enable open-iscsi
```

Figure 4 Ubuntu services

Setting up DM Series using Trident

The Prerequisites for Trident are as follows:

- Kubernetes cluster (v1.14 and above)
  - Installed packages: kubelet, kubeadm, kubectl, kubernetes-cni
- Master/Worker node requirements:
  - Swap is disabled
  - Multipathing is enabled (for iSCSI deployments)
- NFS or iSCSI SVM setup on DM

The steps to set up Trident are as follows:

1. Download and extract the Trident installer using wget:

```
wget https://github.com/NetApp/trident/releases/download/v21.04.0/trident-installer-21.04.0.tar.gz
tar -xvzf trident-installer-21.04.0.tar.gz
```

Figure 5 Download Trident

Note: In this example, v21.04.0 is used but the version used in the get and extract may change depending on the latest version. Check the latest release of Trident in Github:

https://github.com/NetApp/trident/releases

2. Install Trident as follows:

```
cd trident-installer
./tridentctl install -n trident
```

Figure 6 Installing Trident

3. Check if Trident pods have been set up using the following command:

```
kubectl get pod -n trident
```

Figure 7 Verifying Trident pods

4. Create and Add backend
   a. In the trident-installer directory, create a new directory named `setup`.
   b. In the setup directory, create a new file named `dm-backend.json`.
   c. Define the configuration for DM Series in the dm-backend file. For NFS using the following:

```
{
    "version": 1,
    "storageDriverName": "ontap-nas",
    "backendName": "DM5000F",
    "managementLIF": "<cluster_mgmt_lif_ip>",
    "dataLIF": "<nfs_svm_data_lif_ip>",
    "svm": "<nfs_svm>",
    "username": "<cluster_admin_username>",
    "password": "<cluster_admin_password",
    "aggregate": "<aggregate_to_use>"
}
```

Figure 8 Contents of dm-backend.json for NFS

   d. For iSCSI, use the following:

```
{
    "version": 1,
    "storageDriverName": "ontap-san",
    "backendName": "DM7000F",
    "managementLIF": "<cluster_mgmt_lif_ip>",
    "dataLIF": "<iscsi_svm_data_lif_ip>",
    "svm": "<iscsi_svm>",
    "username": "<cluster_admin_username>",
    "password": "<cluster_admin_password"
}
```

Figure 9 Contents of dm-backend.json for iSCSI

5. Add backend for the defined DM Series storage using the following command:

```
./tridentctl -n trident create backend -f setup/dm-backend.json
```

Figure 10 Add the backend

6. Create storage class by first creating a new yaml file named `storage-class-dm.yaml`

   For NFS, the file contains the following:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontapnastcp
provisioner: netapp.io/trident
mountOptions: ["rw", "nfsvers=3", "proto=tcp"]
parameters:
  backendType: "ontap-nas"
```

Figure 11 Contents of storage-class-dm.yaml for NFS

   For iSCSI, the file contains the following:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontapsan
provisioner: netapp.io/trident
parameters:
  backendType: "ontap-san"
```

Figure 12 Contents of storage-class-dm.yaml for iSCSI

7. Create storage class for the defined DM Series class

```
kubectl create -f storage-class-dm.yaml
```

Figure 13 Create the storage class

8. Check to see that the storage class was created using the following command:

```
kubectl get sc
```

Figure 14 Verify the storage class was created

9. Provision volume, first by creating a new yaml file named `pvc-ontap.yaml`. For NFS, the file contains the following:

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: ontapnastcp
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: ontapnastcp
```

Figure 15 Contents of pvc-ontap.yaml for NFS

For iSCSI, the file contains the following

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: ontapsan
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: ontapsan
```

Figure 16 Contents of pvc-ontap.yaml for iSCSI

10. Create the PersistentVolumeClaim for the defined DM Series PVC

```
kubectl create -f pvc-ontap.yaml
```

Figure 17 Create the PVC

11. Check to see that the volume was created:

```
kubectl get pvc
```

Figure 18 Verify that the PVC was created

> **Note:** The DM Series Storage Manager GUI should also display the newly created volume under **Storage** → **Volumes**.

12. Mount the volume in a pod for container, first by creating a new yaml file, `dm-pod.yaml`. For NFS, the file contains the following:

```
kind: Pod
apiVersion: v1
metadata:
    name: dm-pod
spec:
  volumes:
    - name: dm-storage
      persistentVolumeClaim:
        claimName: ontapnastcp
  containers:
    <...>
```

Figure 19 Contents of dm-pod.yaml for NFS

For iSCSI, the file contains the following:

```
kind: Pod
apiVersion: v1
metadata:
    name: dm-pod
spec:
  volumes:
    - name: dm-storage
      persistentVolumeClaim:
        claimName: ontapsan
  containers:
    <...>
```

Figure 20 Contents of dm-pod.yaml for iSCSI

> **Tip:** The claimName should match the name of the storage class created in step 6 on page 6. Use **`kubectl get pvc`** to check the name.

13. Create the pod

```
kubectl create -f dm-pod.yaml
```

Figure 21 Create the pod

14. Check to see that the pod was created

```
kubectl get pod
```

Figure 22 Verify the pod

Setting up DE Series

The procedures described here can aid in setting up containers in Kubernetes clusters with DE Series storage. A server hosts the Kubernetes services and consumes DE storage. The DE Series controllers are not hosting or servicing Kubernetes services natively onboard the storage controllers.

It is assumed that a Kubernetes cluster has already been setup with master/worker nodes.

Components used:

- Ubuntu (21.04 LTS)
  - multipath-tools 0.8.5-lubuntu6
  - No VLANs. iSCSI traffic sent through dedicated Ethernet switch and management traffic sent through management switch
- Kubernetes (v1.21.1; Server and Client)
- DE4000H (11.60.2)
  - Tested configuration. DE2000/DE6000 also candidates

Prerequisites

Nodes in the Kubernetes cluster should have the prerequisites:

1. Verify the following iSCSI packages are installed:

   Note: The Kubernetes cluster nodes should be rebooted after installing the iSCSI tools

   - RHEL and CentOS: lsscsi, iscsi-initiator-utils, sg3_utils, device-mapper-multipath
   - Ubuntu: open-iscsi, lsscsi, sg3_utils, multipath-tools, scsitools

2. Verify that multipathing is enabled:

   RHEL and CentOS:

```
sudo tee /etc/multipath.conf <<-'EOF'
devices{
    device {
        vendor                  "LENOVO"
        product                 "DE_Series"
        product_blacklist       "Universal Xport"
        path_grouping_policy    "group_by_prio"
        path_checker            "alua"
        features                "2 pg_init_retries 50"
        hardware_handler        "1 alua"
        prio                    "alua"
        failback                immediate
        rr_weight               "uniform"
        no_path_retry           30
        retain_attached_hw_handler      yes
        detect_prio             yes
    }
}
EOF
sudo mpathconf --enable --with_multipathd y
```

Figure 23   RHEL and CentOS multipathing

Ubuntu:

```
sudo tee /etc/multipath.conf <<-'EOF'
devices{
    device {
        vendor                  "LENOVO"
        product                 "DE_Series"
        product_blacklist       "Universal Xport"
        path_grouping_policy    "group_by_prio"
        path_checker            "alua"
        features                "2 pg_init_retries 50"
        hardware_handler        "1 alua"
        prio                    "alua"
        failback                immediate
        rr_weight               "uniform"
        no_path_retry           30
        retain_attached_hw_handler      yes
        detect_prio             yes
    }
}
EOF

sudo systemctl enable multipath-tools.service
sudo service multipath-tools restart
```

Figure 24 Ubuntu multipathing

> **Note:** ALUA should be used if ALB is enabled. For configurations with ALB disabled, use RDAC instead. See "Auto Load Balancing" on page 14 for details about ALB.

3. Verify services are running and enable them to autorun:

   RHEL and CentOS:

```
sudo systemctl status iscsid multipathd iscsi
sudo systemctl enable iscsid multipathd iscsi
```

Figure 25 RHEL and CentOS services

Ubuntu:

```
sudo systemctl status multipath-tools open-iscsi
sudo systemctl enable open-iscsi
```

Figure 26 Ubuntu services

Setting up DE Series without Trident

Pre-requisites:

- Kubernetes cluster (v1.14 and above)
  - Installed packages: kubelet, kubeadm, kubectl, kubernetes-cni
- Master/Worker node requirements:
  - multipath is setup (see above in DE Series section for setup)
  - Swap is disabled
  - Network interfaces created to communicate with iSCSI target interfaces
- Host cluster created on DE System Manager with Kubernetes nodes added into host cluster (see "Creating Host Cluster on DE Series System Manager with Kubernetes nodes added" on page 13)
- iSCSI data network interfaces created on DE Series

> **Note:** If desired, refer to "Manual iSCSI Session Management" on page 14 on commands for validation of iSCSI target connectivity prior to container setup for validation of target connectivity. Ensure to logout of all iSCSI sessions prior to continuing with the procedure

The steps to set up Kubernetes on DE Series are as follows:

1. On the DE Series System Manager, create a new volume and assign it to the host cluster with the master/worker nodes

   See the following for additional information:

   - See "Creating volumes on System Manager" on page 13 for volume creation
   - See "Creating Host Cluster on DE Series System Manager with Kubernetes nodes added" on page 13 for host cluster/host creation
   - See "Assigning Volumes to Host Cluster on System Manager" on page 14 for associating volumes to host cluster

   > **Note:** Make note of the LUN ID for the volume. It will be used in later steps. See "Creating volumes on System Manager" on page 13 for more details on volume creation.

2. Mount volume in pod for container by first creating a new yaml file named iscsi.yaml with the following contents:

```
---
apiVersion: v1
kind: Pod
metadata:
  name: iscsi-test
spec:
  containers:
  - name: iscsi-test
    image: kubernetes/pause
    volumeMounts:
    - mountPath: "/mnt/iscsi-test"
      name: iscsi-test
  volumes:
  - name: iscsi-test
    iscsi:
      targetPortal: <de_iscsi_port_ip1>
      portals: ['<de_iscsi_port_ip2']
      iqn: <de_iscsi_iqn>
      lun: <lun_id>
      fsType: ext4
  readOnly: false
```

Figure 27 Contents of iscsi.yaml

> **Tip:** The `portals` parameter can be used to list all the iSCSI target network interfaces as necessary.

3. Create a Pod using the following command:

```
kubectl create -f iscsi.yaml
```

Figure 28 Creating a pod

4. Verify pod creation:

```
kubectl get pod
```

Figure 29 Verify that the pod was created

5. Verify multipath configuration for multiple paths to volume:

```
mpathe (XXXXXXXXXXXXXXXXXXX) dm-2 LENOVO,DE_Series
size=10G features='3 queue_if_no_path pg_init_retries 50' hwhandler='1 alua'
wp=rw
|-+- policy='service-time 0' prio=50 status=active
| `- 35:0:0:2 sde 8:64 active ready running
`-+- policy='service-time 0' prio=10 status=enabled
`- 36:0:0:2 sdg 8:96 active ready running
```

Figure 30 Verify multipath configuration

Appendix: Procedures for DE Series

Creating volumes on System Manager

1. Visit management IP address of DE Series system to access System Manager
2. Log in to System Manager
3. Create Volumes, go to **Storage** → **Volumes** and click **Create** → **Volume**
4. Complete the volume create wizard:
   a. On the host drop down menu, select the host cluster created with the Kubernetes nodes added
   b. Select an existing workload or create a new one
   c. On one of the volume groups, click **Add new volume**
   d. Enter a name for the volume and capacity for the volume
   e. Repeat for as many volumes as desired
   f. Click **Finish**

Creating Host Cluster on DE Series System Manager with Kubernetes nodes added

1. Visit management IP address of DE Series system to access System Manager
2. Log in to System Manager
3. Add Hosts
   a. Go to **Storage** → **Hosts**
   b. Click **Create** → **Host**
   c. Complete the information in the dialog box:
      - Enter name for the host
      - Select the operating system type for the host
      - Select iSCSI for host ports
      - Enter the IQN for the host
   d. Click **Create**
   e. Repeat steps c and d for all remaining nodes

4. Create Host Cluster
   a. Click **Create** → **Host Cluster**
   b. Complete the information in the dialog box:
      - Enter name for the host cluster
      - In the drop-down menu, select all the hosts (nodes) that were created in step 3
   c. Click **Create**

Assigning Volumes to Host Cluster on System Manager

1. Visit management IP address of DE Series system to access System Manager
2. Log in to System Manager
3. Assign volume(s) to host cluster
   a. Go to **Storage** → **Hosts**
   b. Select the row for the host cluster (Type listed as Cluster)
   c. Click **Assign Volumes**
   d. Select volumes to be assigned to host cluster
   e. Click **Assign**

Auto Load Balancing

To enable or disable Auto Load Balancing (ALB) on the DE Series system, perform the following:

1. Visit management IP address of DE Series system to access System Manager
2. Log in to System Manager
3. Enable or Disable ALB
   f. Go to **Settings** → **System**
   g. Scroll down to the Additional Settings Pane
   h. Under “Enable/Disable Automatic Load Balancing”, the status is shown as enabled or disabled
   i. Click on **Enable/Disable Automatic Load Balancing** to enable or disable this feature

Note that systems with ALB enabled should use alua for the multipath policy while systems with ALB disabled should use rdac as the multipath policy

Manual iSCSI Session Management

To discover an iSCSI target, connect and verify session connectivity, the following commands can be used:

Discover targets and IQN of targets:

```
sudo iscsiadm -m discovery -t sendtargets -p <target_iscsi_ip_address>
```

Figure 31 Discover targets and IQN of targets

Connect iSCSI session to discovered target:

```
sudo iscsiadm --mode node --targetname <target_iscsi_iqn> --portal
<target_iscsi_ip_address> --login
```

Figure 32 Connect iSCSI session to discovered target

Disconnect iSCSI session with a target:

```
sudo iscsiadm --mode node --targetname<target_iscsi_iqn> --portal
<target_iscsi_ip_address> --logout
```

Figure 33 Disconnect iSCSI session with a target

Verify session connectivity:

```
sudo iscsiadm -m session
```

Figure 34 Verify session connectivity

Note: Ensure at least one connection from each DE Series controller (target) is connected to for a total of two paths to the target. The target will need at least one iSCSI network interface created on each controller for connectivity

Resources

Resources for Trident:

- NetApp.io (Trident)

  https://netapp.io/persistent-storage-provisioner-for-kubernetes/
- Readthedocs (Trident)

  https://netapp-trident.readthedocs.io/
- Trident on GitHub

  https://github.com/netapp/trident

GitHub hosts the latest release that can be pulled using wget from the CLI of the Kubernetes host for installation. Users should be within a year of the latest Trident release. For more information, see the following page:

https://mysupport.netapp.com/site/info/trident-support

Authors

Anthony Yu is a Storage Development Engineer in the Lenovo Infrastructure Solutions Group. Current responsibilities and initiatives cover various storage technologies including NVMe, ONTAP, and block storage. Notable programs related to these technologies include ThinkAgile CP-SB and DM Series.

Notices

Lenovo may not offer the products, services, or features discussed in this document in all countries. Consult your local Lenovo representative for information on the products and services currently available in your area. Any reference to a Lenovo product, program, or service is not intended to state or imply that only that Lenovo product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any Lenovo intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any other product, program, or service.

Lenovo may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

Lenovo (United States), Inc.
1009 Think Place - Building One
Morrisville, NC 27560
U.S.A.
Attention: Lenovo Director of Licensing

LENOVO PROVIDES THIS PUBLICATION “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. Lenovo may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

The products described in this document are not intended for use in implantation or other life support applications where malfunction may result in injury or death to persons. The information contained in this document does not affect or change Lenovo product specifications or warranties. Nothing in this document shall operate as an express or implied license or indemnity under the intellectual property rights of Lenovo or third parties. All information contained in this document was obtained in specific environments and is presented as an illustration. The result obtained in other operating environments may vary.

Lenovo may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Any references in this publication to non-Lenovo Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this Lenovo product, and use of those Web sites is at your own risk.

Any performance data contained herein was determined in a controlled environment. Therefore, the result obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

This document was created or updated on April 25, 2022.

Send us your comments via the **Rate & Provide Feedback** form found at
http://lenovopress.com/lp1583

Trademarks

Lenovo and the Lenovo logo are trademarks or registered trademarks of Lenovo in the United States, other countries, or both. These and other Lenovo trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by Lenovo at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of Lenovo trademarks is available from https://www.lenovo.com/us/en/legal/copytrade/.

The following terms are trademarks of Lenovo in the United States, other countries, or both:

Lenovo®	Lenovo(logo)®	ThinkSystem™

The following terms are trademarks of other companies:

Other company, product, or service names may be trademarks or service marks of others.