# Installieren Sie Trident auf einem Red Hat OpenShift-Cluster und erstellen Sie Speicherobjekte

NetApp virtualization solutions

NetApp
August 18, 2025

# Inhalt

# Installieren Sie Trident auf einem Red Hat OpenShift-Cluster und erstellen Sie Speicherobjekte

Installieren Sie Trident mit dem Red Hat Certified Trident Operator auf OpenShift-Clustern und bereiten Sie Worker-Knoten für den Blockzugriff vor. Erstellen Sie Trident -Backend- und Speicherklassenobjekte für ONTAP und FSxN-Speicher, um die dynamische Volumebereitstellung für Container und VMs zu ermöglichen.

Wenn Sie VMs in OpenShift Virtualization erstellen müssen, muss Trident installiert sein und die Back-End-Objekte und die Speicherklassenobjekte müssen im OpenShift-Cluster erstellt werden, bevor OpenShift Virtualization auf dem Cluster (vor Ort und ROSA) installiert wird. Die Standardspeicherklasse und die Standard-Volume-Snapshot-Klasse müssen auf den Trident -Speicher und die Snapshot-Klasse im Cluster eingestellt werden. Nur wenn dies konfiguriert ist, kann OpenShift Virtualization die Golden Images lokal für die VM-Erstellung mithilfe von Vorlagen verfügbar machen.

Wenn der OpenShift Virtualization-Operator vor der Installation von Trident installiert wird, können Sie den folgenden Befehl verwenden, um die mit einer anderen Speicherklasse erstellten Golden Images zu löschen und dann OpenShift Virtualization die Golden Images mit der Trident -Speicherklasse erstellen zu lassen, indem Sie sicherstellen, dass die Standardwerte für die Trident -Speicher- und Volume-Snapshot-Klasse festgelegt sind.

```
oc delete dv,VolumeSnapshot -n openshift-virtualization-os-images
--selector=cdi.kubevirt.io/dataImportCron
```

Um Beispiel-YAML-Dateien zum Erstellen von Trident-Objekten für FSxN-Speicher für ROSA-Cluster und eine Beispiel-YAML-Datei für die VolumeSnapshotClass zu erhalten, scrollen Sie auf dieser Seite nach unten.

- Trident installieren**

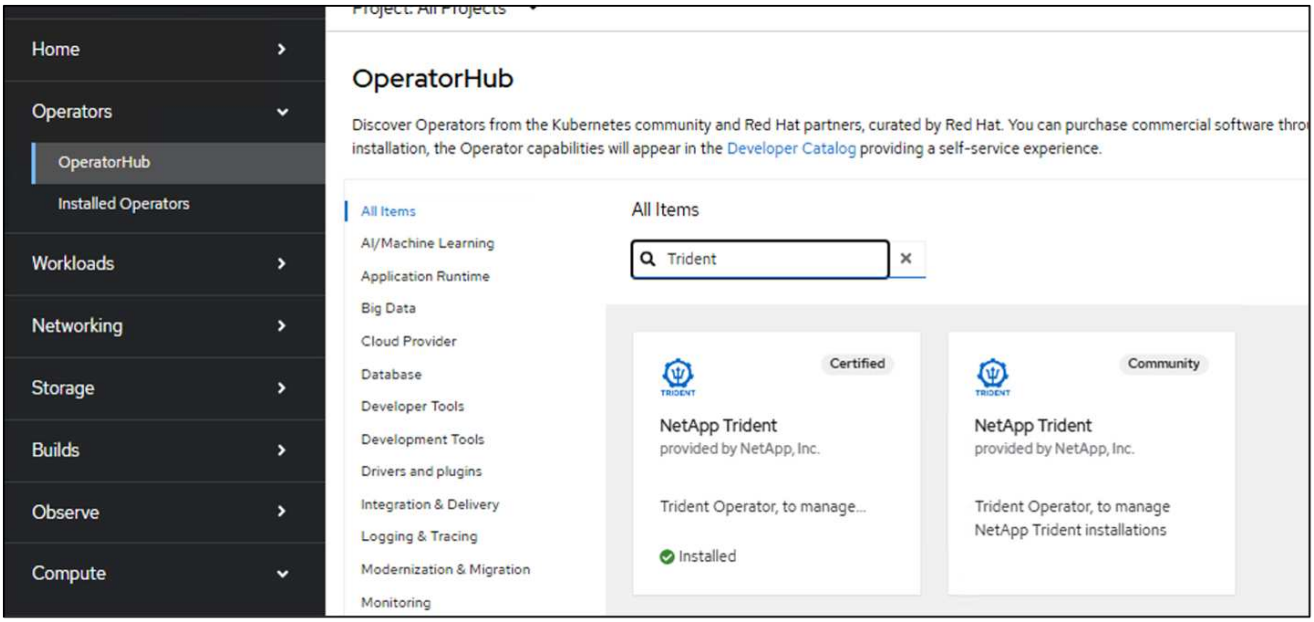
### Installieren von Trident mit dem Red Hat Certified Operator

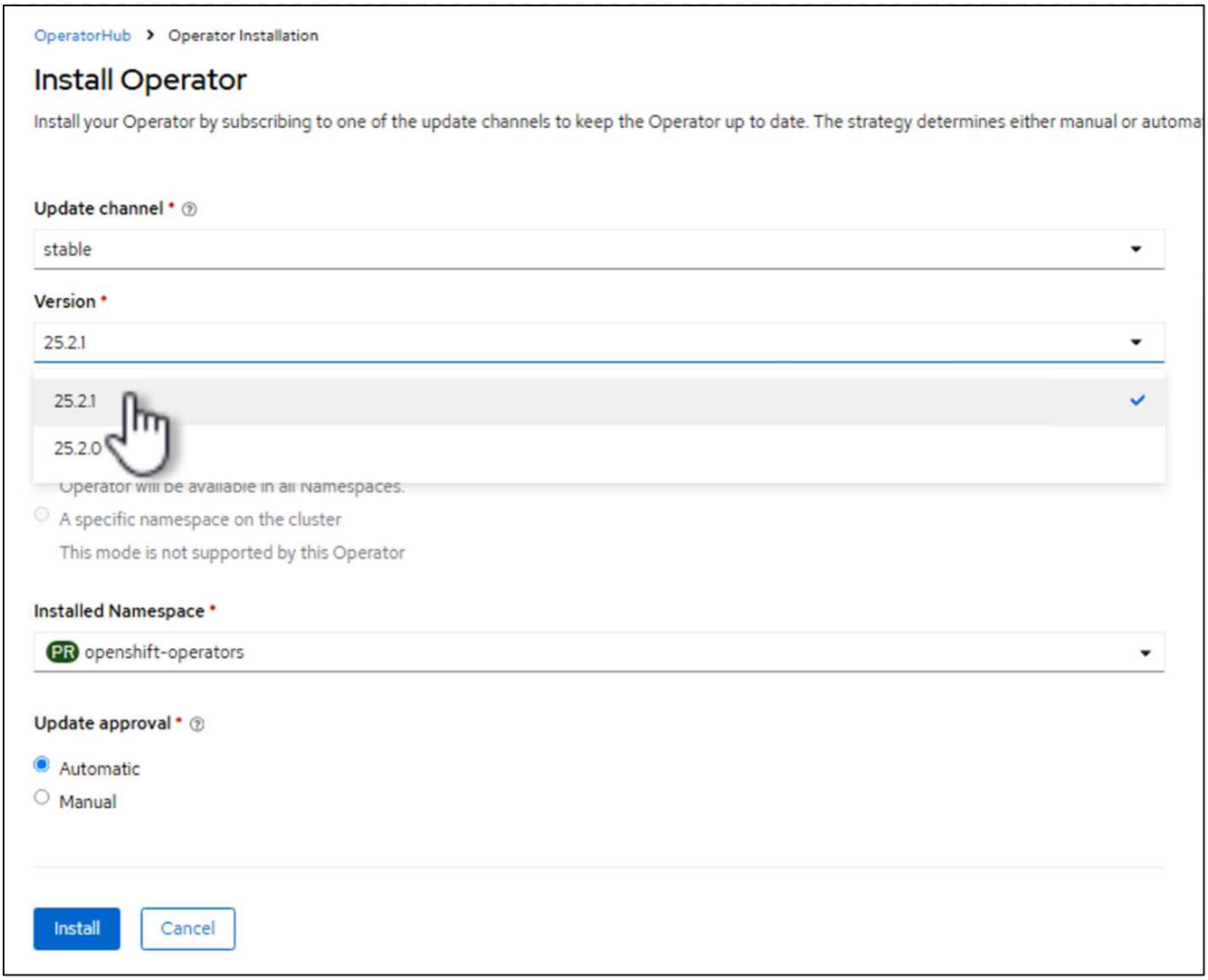
In diesem Abschnitt werden Details zur Installation von Trident mit dem Red Hat Certified Trident Operator bereitgestellt."Siehe die Trident -Dokumentation" für andere Möglichkeiten zur Installation von Trident. Mit der Veröffentlichung von Trident 25.02 können Benutzer von Trident in Red Hat OpenShift vor Ort und in der Cloud sowie verwaltete Dienste wie Red Hat OpenShift Service auf AWS Trident jetzt mit dem Trident Certified Operator vom Operator Hub installieren. Dies ist für die OpenShift-Benutzergemeinschaft von Bedeutung, da Trident zuvor nur als Community-Betreiber verfügbar war.

Der Vorteil des Red Hat Certified Trident -Operators besteht darin, dass die Grundlage für den Operator und seine Container bei Verwendung mit OpenShift (ob vor Ort, in der Cloud oder als Managed Service mit ROSA) vollständig von NetApp unterstützt wird. Darüber hinaus ist NetApp Trident für den Kunden kostenlos. Sie müssen es also lediglich mit dem zertifizierten Operator installieren, der nachweislich nahtlos mit Red Hat OpenShift funktioniert und für ein einfaches Lebenszyklusmanagement verpackt ist.

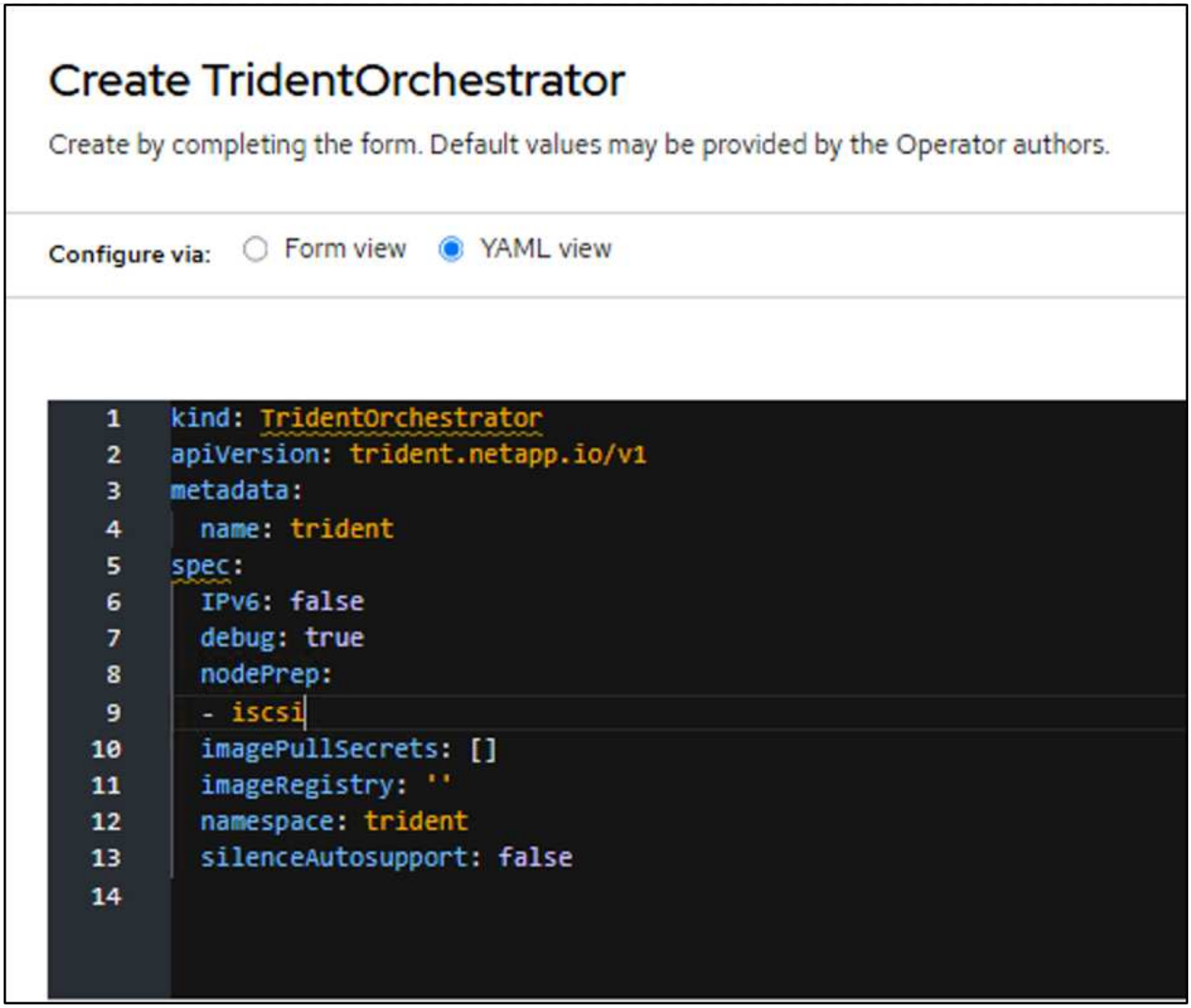
Darüber hinaus bietet der Trident 25.02-Operator (und zukünftige Versionen) den optionalen Vorteil, die Worker-Knoten für iSCSI vorzubereiten. Dies ist besonders vorteilhaft, wenn Sie Ihre Workloads auf ROSA-Clustern bereitstellen und das iSCSI-Protokoll mit FSxN verwenden möchten, insbesondere für OpenShift Virtualization VM-Workloads. Die Herausforderung der Vorbereitung von Worker-Knoten für iSCSI auf ROSA-Clustern mit FSxN wurde durch diese Funktion bei der Installation von Trident auf dem Cluster gemildert.

Die Installationsschritte mit dem Operator sind dieselben, unabhängig davon, ob Sie ihn auf einem lokalen Cluster oder auf ROSA installieren. Um Trident mit dem Operator zu installieren, klicken Sie auf den Operator-Hub und wählen Sie „Certified NetApp Trident“ aus. Auf der Installationsseite ist standardmäßig die neueste Version ausgewählt. Klicken Sie auf Installieren.






Sobald der Operator installiert ist, klicken Sie auf „Operator anzeigen“ und erstellen Sie dann eine Instanz des Trident Orchestrator. Wenn Sie die Worker-Knoten für den iSCSI-Speicherzugriff vorbereiten möchten, gehen Sie zur YAML-Ansicht und ändern Sie den Parameter „nodePrep“, indem Sie „iscsi“ hinzufügen.




Jetzt sollten alle Trident-Pods in Ihrem Cluster ausgeführt werden.

```
[root@localhost ~]# oc get pods -n trident
NAME                                READY   STATUS    RESTARTS   AGE
trident-controller-84cb9bff89-lkx6k   6/6     Running   0          16h
trident-node-linux-d88b9            2/2     Running   0          16h
trident-node-linux-ld4b8            2/2     Running   0          16h
trident-node-linux-mj5r8            2/2     Running   0          16h
trident-node-linux-mkmmp            2/2     Running   0          16h
trident-node-linux-qhgr7            2/2     Running   0          16h
trident-node-linux-vt9tp            2/2     Running   0          16h
[root@localhost ~]#
```

Um zu überprüfen, ob iSCSI-Tools auf den Worker-Knoten des OpenShift-Clusters aktiviert wurden, melden Sie sich bei den Worker-Knoten an und überprüfen Sie, ob die iscsid, multipathd aktiv und die Einträge in der Datei multipath.conf wie gezeigt angezeigt werden.

```
sh-5.1# systemctl status iscsid
● iscsid.service - Open-iSCSI
     Loaded: loaded (/usr/lib/systemd/system/iscsid.service; enabled; preset: disabled)
     Active: active (running) since Fri 2025-04-25 00:23:49 UTC; 3 days ago
TriggeredBy: ● iscsid.socket
       Docs: man:iscsid(8)
             man:iscsiuio(8)
             man:iscsiadm(8)
   Main PID: 74787 (iscsid)
     Status: "Ready to process requests"
      Tasks: 1 (limit: 410912)
     Memory: 1.8M
        CPU: 6ms
     CGroup: /system.slice/iscsid.service
             └─74787 /usr/sbin/iscsid -f

Apr 25 00:23:49 ocp11-worker1 systemd[1]: Starting Open-iSCSI...
Apr 25 00:23:49 ocp11-worker1 systemd[1]: Started Open-iSCSI.
sh-5.1#
```

```
sh-5.1# systemctl status multipathd
● multipathd.service - Device-Mapper Multipath Device Controller
     Loaded: loaded (/usr/lib/systemd/system/multipathd.service; enabled; preset: enabled)
     Active: active (running) since Fri 2025-04-25 00:23:50 UTC; 3 days ago
TriggeredBy: ● multipathd.socket
    Process: 74905 ExecStartPre=/sbin/modprobe -a scsi_dh_alua scsi_dh_emc scsi_dh_rdac dm-multipath (code=exited, status=0/SUCCESS)
    Process: 74906 ExecStartPre=/sbin/multipath -A (code=exited, status=0/SUCCESS)
   Main PID: 74907 (multipathd)
     Status: "up"
      Tasks: 7
     Memory: 18.3M
        CPU: 23.008s
     CGroup: /system.slice/multipathd.service
             └─74907 /sbin/multipathd -d -s

Apr 25 00:23:50 ocp11-worker1 systemd[1]: Starting Device-Mapper Multipath Device Controller...
Apr 25 00:23:50 ocp11-worker1 multipathd[74907]: --------start up--------
Apr 25 00:23:50 ocp11-worker1 multipathd[74907]: read /etc/multipath.conf
Apr 25 00:23:50 ocp11-worker1 multipathd[74907]: path checkers start up
Apr 25 00:23:50 ocp11-worker1 systemd[1]: Started Device-Mapper Multipath Device Controller.
sh-5.1#
```

```
sh-5.1# cat /etc/multipath.conf
defaults {
    find_multipaths no
}
blacklist {
    device {
        product .*
        vendor  .*
    }
}
blacklist_exceptions {
    device {
        product LUN
        vendor  NETAPP
    }
}
sh-5.1#
```

## Videodemonstration

Das folgende Video zeigt eine Demonstration der Installation von Trident mit Red Hat Certified Trident Operator

Installieren von Trident 25.02.1 mit dem zertifizierten Trident Operator in OpenShift

## Trident Konfiguration für lokalen OpenShift-Cluster

**Trident -Backend und Speicherklasse für NAS**

```
cat tbc-nas.yaml
apiVersion: v1
kind: Secret
metadata:
  name: tbc-nas-secret
type: Opaque
stringData:
  username: <cluster admin username>
  password: <cluster admin password>
---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: tbc-nas
spec:
  version: 1
  storageDriverName: ontap-nas
  managementLIF: <cluster management lif>
  backendName: tbc-nas
  svm: zoneb
  storagePrefix: testzoneb
  defaults:
    nameTemplate: "{{ .config.StoragePrefix }}_{{ .volume.Namespace
}}_{{ .volume.RequestName }}"
  credentials:
    name: tbc-nas-secret
```

```
cat sc-nas.yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: sc-nas
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas"
  media: "ssd"
  provisioningType: "thin"
  snapshots: "true"
allowVolumeExpansion: true
```

**Trident -Backend und Speicherklasse für iSCSI**

```
# cat tbc-iscsi.yaml
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-ontap-iscsi-secret
type: Opaque
stringData:
  username: <cluster admin username>
  password: <cluster admin password>
---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: ontap-iscsi
spec:
  version: 1
  storageDriverName: ontap-san
  managementLIF: <management LIF>
  backendName: ontap-iscsi
  svm: <SVM name>
  credentials:
    name: backend-tbc-ontap-iscsi-secret
```

```
# cat sc-iscsi.yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: sc-iscsi
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
  media: "ssd"
  provisioningType: "thin"
  fsType: ext4
  snapshots: "true"
allowVolumeExpansion: true
```

**Trident -Backend und Speicherklasse für NVMe/TCP**

```
# cat tbc-nvme.yaml
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-ontap-nvme-secret
type: Opaque
stringData:
  username: <cluster admin password>
  password: <cluster admin password>
---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-nvme
spec:
  version: 1
  storageDriverName: ontap-san
  managementLIF: <cluster management LIF>
  backendName: backend-tbc-ontap-nvme
  svm: <SVM name>
  credentials:
    name: backend-tbc-ontap-nvme-secret
```

```
# cat sc-nvme.yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: sc-nvme
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
  media: "ssd"
  provisioningType: "thin"
  fsType: ext4
  snapshots: "true"
allowVolumeExpansion: true
```

**Trident -Backend und Speicherklasse für FC**

```
# cat tbc-fc.yaml
apiVersion: v1
kind: Secret
metadata:
  name: tbc-fc-secret
type: Opaque
stringData:
  username: <cluster admin password>
  password: <cluster admin password>
---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: tbc-fc
spec:
  version: 1
  storageDriverName: ontap-san
  managementLIF: <cluster mgmt lif>
  backendName: tbc-fc
  svm: openshift-fc
  sanType: fcp
  storagePrefix: demofc
  defaults:
    nameTemplate: "{{ .config.StoragePrefix }}_{{ .volume.Namespace
}}_{{ .volume.RequestName }}"
  credentials:
    name: tbc-fc-secret
```

```
# cat sc-fc.yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: sc-fc
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
  media: "ssd"
  provisioningType: "thin"
  fsType: ext4
  snapshots: "true"
allowVolumeExpansion: true
```

# Trident -Konfiguration für ROSA-Cluster mit FSxN-Speicher

**Trident -Backend und Speicherklasse für FSxN NAS**

```
#cat tbc-fsx-nas.yaml
apiVersion: v1
kind: Secret
metadata:
  name: backend-fsx-ontap-nas-secret
  namespace: trident
type: Opaque
stringData:
  username: <cluster admin lif>
  password: <cluster admin passwd>
---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-fsx-ontap-nas
  namespace: trident
spec:
  version: 1
  backendName: fsx-ontap
  storageDriverName: ontap-nas
  managementLIF: <Management DNS name>
  dataLIF: <NFS DNS name>
  svm: <SVM NAME>
  credentials:
    name: backend-fsx-ontap-nas-secret
```

```
# cat sc-fsx-nas.yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: trident-csi
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas"
  fsType: "ext4"
allowVolumeExpansion: True
reclaimPolicy: Retain
```

**Trident -Backend und Speicherklasse für FSxN iSCSI**

```
# cat tbc-fsx-iscsi.yaml
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-fsx-iscsi-secret
type: Opaque
stringData:
  username: <cluster admin username>
  password: <cluster admin password>
---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: fsx-iscsi
spec:
  version: 1
  storageDriverName: ontap-san
  managementLIF: <management LIF>
  backendName: fsx-iscsi
  svm: <SVM name>
  credentials:
    name: backend-tbc-ontap-iscsi-secret
```

```
# cat sc-fsx-iscsi.yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: sc-fsx-iscsi
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
  media: "ssd"
  provisioningType: "thin"
  fsType: ext4
  snapshots: "true"
allowVolumeExpansion: true
```

# Erstellen der Trident Volume Snapshot-Klasse

**Trident -Volume-Snapshot-Klasse**

```
# cat snapshot-class.yaml
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshotClass
metadata:
  name: trident-snapshotclass
driver: csi.trident.netapp.io
deletionPolicy: Retain
```

Sobald Sie die erforderlichen YAML-Dateien für die Backend-Konfiguration, die Speicherklassenkonfiguration und die Snapshot-Konfigurationen haben, können Sie das Trident-Backend, die Speicherklasse und die Snapshot-Klassenobjekte mit dem folgenden Befehl erstellen

```
oc create -f <backend-filename.yaml> -n trident
oc create -f < storageclass-filename.yaml>
oc create -f <snapshotclass-filename.yaml>
```

# Festlegen von Standardeinstellungen mit Trident Storage und Snapshot Class

### Festlegen von Standardeinstellungen mit Trident Storage und Snapshot Class

Sie können jetzt die erforderliche Trident-Speicherklasse und die Volume-Snapshot-Klasse als Standard im OpenShift-Cluster festlegen. Wie bereits erwähnt, ist das Festlegen der Standardspeicherklasse und der Volume-Snapshot-Klasse erforderlich, damit OpenShift Virtualization die Golden Image-Quelle zum Erstellen von VMs aus Standardvorlagen verfügbar machen kann.

Sie können die Trident Speicherklasse und die Snapshot-Klasse als Standard festlegen, indem Sie die Anmerkung über die Konsole bearbeiten oder über die Befehlszeile Folgendes patchen.

```
storageclass.kubernetes.io/is-default-class:true
or
kubectl patch storageclass standard -p '{"metadata": {"annotations":{
"storageclass.kubernetes.io/is-default-class":"true"}}}'

storageclass.kubevirt.io/is-default-virt-class: true
or
kubectl patch storageclass standard -p '{"metadata": {"annotations":{
"storageclass.kubevirt.io/is-default-virt-class": "true"}}}'
```

Sobald dies festgelegt ist, können Sie alle bereits vorhandenen dv- und VolumeSnapShot-Objekte mit dem folgenden Befehl löschen:

```
oc delete dv,VolumeSnapshot -n openshift-virtualization-os-images
--selector=cdi.kubevirt.io/dataImportCron
```