NetApp Astra Trident 概述

NetApp Solutions

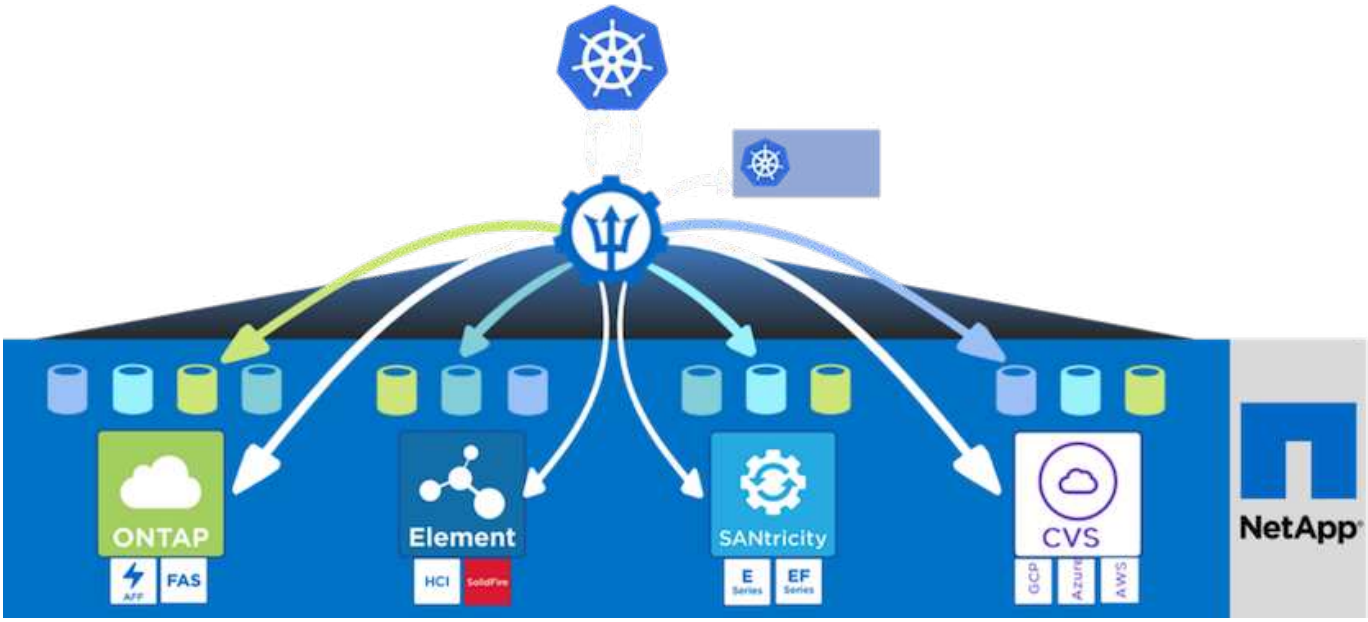
NetApp
April 12, 2024

目录

Astra Trident 概述

Astra Trident 是一款开源且完全受支持的存储编排程序，适用于容器和 Kubernetes 分发版，包括 Red Hat OpenShift 。Trident 可与包括 NetApp ONTAP 和 Element 存储系统在内的整个 NetApp 存储产品组合配合使用，并且还支持 NFS 和 iSCSI 连接。Trident 允许最终用户从其 NetApp 存储系统配置和管理存储，而无需存储管理员干预，从而加快了 DevOps 工作流的速度。

管理员可以根据项目需求和存储系统型号配置多个存储后端，以实现高级存储功能，包括数据压缩，特定磁盘类型或 QoS 级别，以保证一定水平的性能。定义后，开发人员可以在其项目中使用这些后端创建永久性卷声明（PVC），并按需将永久性存储附加到容器。



Astra Trident 具有快速的开发周期，就像 Kubernetes 一样，每年发布四次。

最新版 Astra Trident 于 2022 年 1 月发布。已测试的 Trident 版本的支持列表，可在该支持列表中找到 Kubernetes 分发版本 "此处"。

从 20.04 版开始， Trident 设置由 Trident 操作员执行。操作员可以简化大规模部署，并为在 Trident 安装过程中部署的 Pod 提供额外的支持，包括自我修复。

在 21.01 版中，我们提供了一个 Helm 图表，用于简化 Trident 操作员的安装。

下载 Astra Trident

要在已部署的用户集群上安装 Trident 并配置永久性卷，请完成以下步骤：

1. 将安装归档下载到管理工作站并提取内容。Trident 的当前版本为 22.01 ，可以下载 "此处"。

```
[netapp-user@rhel7 ~]$ wget
https://github.com/NetApp/trident/releases/download/v22.01.0/trident-
installer-22.01.0.tar.gz
--2021-05-06 15:17:30--
```

```
https://github.com/NetApp/trident/releases/download/v22.01.0/trident-
installer-22.01.0.tar.gz
Resolving github.com (github.com)... 140.82.114.3
Connecting to github.com (github.com)|140.82.114.3|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://github-
releases.githubusercontent.com/77179634/a4fa9f00-a9f2-11eb-9053-
98e8e573d4ae?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-
Credential=AKIAIWNJYAX4CSVEH53A%2F20210506%2Fus-east-
1%2Fs3%2Faws4_request&X-Amz-Date=20210506T191643Z&X-Amz-Expires=300&X-
Amz-
Signature=8a49a2a1e08c147d1ddd8149ce45a5714f9853fee19bb1c507989b9543eb36
30&X-Amz-
SignedHeaders=host&actor_id=0&key_id=0&repo_id=77179634&response-
content-disposition=attachment%3B%20filename%3Dtrident-installer-
22.01.0.tar.gz&response-content-type=application%2Foctet-stream
[following]
--2021-05-06 15:17:30--  https://github-
releases.githubusercontent.com/77179634/a4fa9f00-a9f2-11eb-9053-
98e8e573d4ae?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-
Credential=AKIAIWNJYAX4CSVEH53A%2F20210506%2Fus-east-
1%2Fs3%2Faws4_request&X-Amz-Date=20210506T191643Z&X-Amz-Expires=300&X-
Amz-
Signature=8a49a2a1e08c147d1ddd8149ce45a5714f9853fee19bb1c507989b9543eb36
30&X-Amz-
SignedHeaders=host&actor_id=0&key_id=0&repo_id=77179634&response-
content-disposition=attachment%3B%20filename%3Dtrident-installer-
22.01.0.tar.gz&response-content-type=application%2Foctet-stream
Resolving github-releases.githubusercontent.com (github-
releases.githubusercontent.com)... 185.199.108.154, 185.199.109.154,
185.199.110.154, ...
Connecting to github-releases.githubusercontent.com (github-
releases.githubusercontent.com)|185.199.108.154|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 38349341 (37M) [application/octet-stream]
Saving to: 'trident-installer-22.01.0.tar.gz'

100%[=====================================================================
==============================================>] 38,349,341  88.5MB/s
in 0.4s

2021-05-06 15:17:30 (88.5 MB/s) - 'trident-installer-22.01.0.tar.gz'
saved [38349341/38349341]
```

2. 从下载的软件包中提取 Trident 安装。

```
[netapp-user@rhel7 ~]$ tar -xzf trident-installer-22.01.0.tar.gz
[netapp-user@rhel7 ~]$ cd trident-installer/
[netapp-user@rhel7 trident-installer]$
```

使用 Helm 安装 Trident 操作员

1. 首先将用户集群的 `kubeconfig` 文件的位置设置为环境变量，以便您不必引用该文件，因为 Trident 没有传递此文件的选项。

```
[netapp-user@rhel7 trident-installer]$ export KUBECONFIG=~/ocp-
install/auth/kubeconfig
```

2. 在用户集群中创建 Trident 命名空间时，运行 Helm 命令从 Helm 目录中的 tarball 安装 Trident 操作员。

```
[netapp-user@rhel7 trident-installer]$ helm install trident
helm/trident-operator-22.01.0.tgz --create-namespace --namespace trident
NAME: trident
LAST DEPLOYED: Fri May  7 12:54:25 2021
NAMESPACE: trident
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
Thank you for installing trident-operator, which will deploy and manage
NetApp's Trident CSI
storage provisioner for Kubernetes.

Your release is named 'trident' and is installed into the 'trident'
namespace.
Please note that there must be only one instance of Trident (and
trident-operator) in a Kubernetes cluster.

To configure Trident to manage storage resources, you will need a copy
of tridentctl, which is
available in pre-packaged Trident releases.  You may find all Trident
releases and source code
online at https://github.com/NetApp/trident.

To learn more about the release, try:

  $ helm status trident
  $ helm get all trident
```

3. 您可以通过检查命名空间中运行的 Pod 或使用 tridentctl 二进制文件检查已安装的版本来验证 Trident 是否已成功安装。

```
[netapp-user@rhel7 trident-installer]$ oc get pods -n trident
NAME                                READY   STATUS    RESTARTS   AGE
trident-csi-5z45l                   1/2     Running   2          30s
trident-csi-696b685cf8-htdb2        6/6     Running   0          30s
trident-csi-b74p2                   2/2     Running   0          30s
trident-csi-lrw4n                   2/2     Running   0          30s
trident-operator-7c748d957-gr2gw    1/1     Running   0          36s

[netapp-user@rhel7 trident-installer]$ ./tridentctl -n trident version
+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 22.01.0          | 22.01.0          |
+----------------+----------------+
```

在某些情况下，客户环境可能需要自定义 Trident 部署。在这种情况下，还可以手动安装 Trident 操作员并更新所包含的清单以自定义部署。

手动安装 Trident 操作员

1. 首先，将用户集群的 `kubeconfig` 文件的位置设置为环境变量，以便您不必引用该文件，因为 Trident 没有传递此文件的选项。

```
[netapp-user@rhel7 trident-installer]$ export KUBECONFIG=~/ocp-
install/auth/kubeconfig
```

2. `trident` **安装程序** 目录包含用于定义所有所需资源的清单。使用适当的清单创建 `TridentOrchestrator` 自定义资源定义。

```
[netapp-user@rhel7 trident-installer]$ oc create -f
deploy/crds/trident.netapp.io_tridentorchestrators_crd_post1.16.yaml
customresourcedefinition.apiextensions.k8s.io/tridentorchestrators.tride
nt.netapp.io created
```

3. 如果不存在 Trident 命名空间，请使用提供的清单在集群中创建一个 Trident 命名空间。

```
[netapp-user@rhel7 trident-installer]$ oc apply -f deploy/namespace.yaml
namespace/trident created
```

4. 为 Trident 操作员部署创建所需的资源，例如为操作员创建 `ServiceAccount` ，为 `SClusterRole` 和 `ClusterRoleBinding` ，为` erviceAccount ，专用 PodSecurityPolicy `或操作员本身创建。

```
[netapp-user@rhel7 trident-installer]$ oc create -f deploy/bundle.yaml
serviceaccount/trident-operator created
clusterrole.rbac.authorization.k8s.io/trident-operator created
clusterrolebinding.rbac.authorization.k8s.io/trident-operator created
deployment.apps/trident-operator created
podsecuritypolicy.policy/tridentoperatorpods created
```

5. 您可以使用以下命令在操作员部署后检查其状态：

```
[netapp-user@rhel7 trident-installer]$ oc get deployment -n trident
NAME                READY   UP-TO-DATE   AVAILABLE   AGE
trident-operator    1/1     1            1           23s
[netapp-user@rhel7 trident-installer]$ oc get pods -n trident
NAME                                 READY   STATUS    RESTARTS   AGE
trident-operator-66f48895cc-lzczk    1/1     Running   0          41s
```

6. 部署操作员后，我们现在可以使用它来安装 Trident 。这需要创建 `TridentOrchestrator` 。

```
[netapp-user@rhel7 trident-installer]$ oc create -f
deploy/crds/tridentorchestrator_cr.yaml
tridentorchestrator.trident.netapp.io/trident created
[netapp-user@rhel7 trident-installer]$ oc describe torc trident
Name:         trident
Namespace:
Labels:       <none>
Annotations:  <none>
API Version:  trident.netapp.io/v1
Kind:         TridentOrchestrator
Metadata:
  Creation Timestamp:  2021-05-07T17:00:28Z
  Generation:          1
  Managed Fields:
    API Version:  trident.netapp.io/v1
    Fields Type:  FieldsV1
    fieldsV1:
      f:spec:
        .:
        f:debug:
        f:namespace:
    Manager:      kubectl-create
    Operation:    Update
```

```
    Time:         2021-05-07T17:00:28Z
    API Version:  trident.netapp.io/v1
    Fields Type:  FieldsV1
    fieldsV1:
      f:status:
        .:
        f:currentInstallationParams:
          .:
          f:IPv6:
          f:autosupportHostname:
          f:autosupportImage:
          f:autosupportProxy:
          f:autosupportSerialNumber:
          f:debug:
          f:enableNodePrep:
          f:imagePullSecrets:
          f:imageRegistry:
          f:k8sTimeout:
          f:kubeletDir:
          f:logFormat:
          f:silenceAutosupport:
          f:tridentImage:
        f:message:
        f:namespace:
        f:status:
        f:version:
    Manager:         trident-operator
    Operation:       Update
    Time:            2021-05-07T17:00:28Z
  Resource Version:  931421
  Self Link:
/apis/trident.netapp.io/v1/tridentorchestrators/trident
  UID:               8a26a7a6-dde8-4d55-9b66-a7126754d81f
Spec:
  Debug:      true
  Namespace:  trident
Status:
  Current Installation Params:
    IPv6:                       false
    Autosupport Hostname:
    Autosupport Image:          netapp/trident-autosupport:21.01
    Autosupport Proxy:
    Autosupport Serial Number:
    Debug:                      true
    Enable Node Prep:           false
    Image Pull Secrets:
```

```
    Image Registry:
    k8sTimeout:           30
    Kubelet Dir:          /var/lib/kubelet
    Log Format:           text
    Silence Autosupport:  false
    Trident Image:        netapp/trident:22.01.0
  Message:                Trident installed
  Namespace:              trident
  Status:                 Installed
  Version:                v22.01.0
Events:
  Type    Reason      Age   From                        Message
  ----    ------      ----  ----                        -------
  Normal  Installing  80s   trident-operator.netapp.io  Installing
Trident
  Normal  Installed   68s   trident-operator.netapp.io  Trident
installed
```

7. 您可以通过检查命名空间中运行的 Pod 或使用 tridentctl 二进制文件检查已安装的版本来验证 Trident 是否已成功安装。

```
[netapp-user@rhel7 trident-installer]$ oc get pods -n trident
NAME                                READY   STATUS    RESTARTS   AGE
trident-csi-bb64c6cb4-lmd6h         6/6     Running   0          82s
trident-csi-gn59q                   2/2     Running   0          82s
trident-csi-m4szj                   2/2     Running   0          82s
trident-csi-sb9k9                   2/2     Running   0          82s
trident-operator-66f48895cc-lzczk   1/1     Running   0          2m39s

[netapp-user@rhel7 trident-installer]$ ./tridentctl -n trident version
+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 22.01.0        | 22.01.0        |
+----------------+----------------+
```

准备工作节点以进行存储

NFS

大多数 Kubernetes 分发软件包和实用程序都会随附用于挂载默认安装的 NFS 后端的软件包和实用程序，包括 Red Hat OpenShift 。

但是，对于 NFSv3 ，客户端和服务器之间没有协商并发的机制。因此，客户端的最大 SUNRPC 插槽表条目数必须与服务器上支持的值手动同步，以确保 NFS 连接的最佳性能，而服务器不必减小连接的窗口大小。

对于 ONTAP ，支持的最大 SUNRPC 插槽表条目数为 128 ，即 ONTAP 一次可以处理 128 个并发 NFS 请求。但是，默认情况下，每个连接的 Red Hat CoreOS/Red Hat Enterprise Linux 最多包含 65 ， 536 个 SUNRPC 插槽表条目。我们需要将此值设置为 128 ，可以在 OpenShift 中使用计算机配置操作员（ Machine Config Operator ， MCO ）来完成此操作。

要修改 OpenShift 工作节点中的最大 SUNRPC 插槽表条目，请完成以下步骤：

1. 登录到 OCP Web 控制台并导航到 Compute > Machine Configs 。单击 Create Machine Config 。复制并粘贴 YAML 文件，然后单击创建。

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  name: 98-worker-nfs-rpc-slot-tables
  labels:
    machineconfiguration.openshift.io/role: worker
spec:
  config:
    ignition:
      version: 3.2.0
    storage:
      files:
        - contents:
            source: data:text/plain;charset=utf-
8; base64,b3B0aW9ucyBzdW5ycGMgdGNwX21heF9zbG90X3RhYmxlX2VudHJpZXM9MTI4Cg=
=
          filesystem: root
          mode: 420
          path: /etc/modprobe.d/sunrpc.conf
```

2. 创建 MCO 后，需要在所有工作节点上应用此配置并逐个重新启动。整个过程大约需要 20 到 30 分钟。使用 `oc get MCP` 验证是否应用了计算机配置，并确保已更新员工的计算机配置池。

```
[netapp-user@rhel7 openshift-deploy]$ oc get mcp
NAME     CONFIG                                     UPDATED   UPDATING
DEGRADED
master   rendered-master-a520ae930e1d135e0dee7168   True      False
False
worker   rendered-worker-de321b36eeba62df41feb7bc   True      False
False
```

iSCSI

要使工作节点做好准备，以便能够通过 iSCSI 协议映射块存储卷，您必须安装支持此功能所需的软件包。

在 Red Hat OpenShift 中，可通过在部署集群后将 MCO （计算机配置操作员）应用于集群来实现此目的。

要配置工作节点以运行 iSCSI 服务，请完成以下步骤：

1. 登录到 OCP Web 控制台并导航到 Compute > Machine Configs 。单击 Create Machine Config 。复制并粘贴 YAML 文件，然后单击创建。

   不使用多路径时：

   ```
   apiVersion: machineconfiguration.openshift.io/v1
   kind: MachineConfig
   metadata:
     labels:
       machineconfiguration.openshift.io/role: worker
     name: 99-worker-element-iscsi
   spec:
     config:
       ignition:
         version: 3.2.0
       systemd:
         units:
           - name: iscsid.service
             enabled: true
             state: started
     osImageURL: ""
   ```

   使用多路径时：

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  name: 99-worker-ontap-iscsi
  labels:
    machineconfiguration.openshift.io/role: worker
spec:
  config:
    ignition:
      version: 3.2.0
    storage:
      files:
      - contents:
          source: data:text/plain;charset=utf-
8; base64,ZGVmYXVsdHMgewogICAgICAgIHVzZXJfZnJpZW5kbHlfbmFtZXMgbm8KICAgICA
gICBmaW5kX211bHRpcGF0aHMgbm8KfQoKYmxhY2tsaXN0X2V4Y2VwdGlvbnMgewogICAgICA
gIHByb3BlcnR5ICIoU0NTSV9JREVOVF98SURfV1dOKSIKfQoKYmxhY2tsaXN0IHsKfQoK
          verification: {}
        filesystem: root
        mode: 400
        path: /etc/multipath.conf
    systemd:
      units:
        - name: iscsid.service
          enabled: true
          state: started
        - name: multipathd.service
          enabled: true
          state: started
  osImageURL: ""
```

2. 创建配置后，将此配置应用于工作节点并重新加载它们大约需要 20 到 30 分钟。使用 `oc get MCP` 验证是否应用了计算机配置，并确保已更新员工的计算机配置池。您还可以登录到工作节点，以确认 iscsid 服务正在运行（如果使用多路径，则 multipathd 服务正在运行）。

```
[netapp-user@rhel7 openshift-deploy]$ oc get mcp
NAME     CONFIG                                     UPDATED   UPDATING
DEGRADED
master   rendered-master-a520ae930e1d135e0dee7168   True      False
False
worker   rendered-worker-de321b36eeba62df41feb7bc   True      False
False

[netapp-user@rhel7 openshift-deploy]$ ssh core@10.61.181.22 sudo
systemctl status iscsid
● iscsid.service - Open-iSCSI
   Loaded: loaded (/usr/lib/systemd/system/iscsid.service; enabled;
vendor preset: disabled)
   Active: active (running) since Tue 2021-05-26 13:36:22 UTC; 3 min ago
     Docs: man:iscsid(8)
           man:iscsiadm(8)
 Main PID: 1242 (iscsid)
   Status: "Ready to process requests"
    Tasks: 1
   Memory: 4.9M
      CPU: 9ms
   CGroup: /system.slice/iscsid.service
           └─1242 /usr/sbin/iscsid -f

[netapp-user@rhel7 openshift-deploy]$ ssh core@10.61.181.22 sudo
systemctl status multipathd
 ● multipathd.service - Device-Mapper Multipath Device Controller
   Loaded: loaded (/usr/lib/systemd/system/multipathd.service; enabled;
vendor preset: enabled)
   Active: active (running) since Tue 2021-05-26 13:36:22 UTC; 3 min ago
  Main PID: 918 (multipathd)
    Status: "up"
    Tasks: 7
    Memory: 13.7M
    CPU: 57ms
    CGroup: /system.slice/multipathd.service
            └─918 /sbin/multipathd -d -s
```

此外，还可以通过使用适当的标志运行 `oc debug` 命令来确认 MachineConfig 已成功应用且服务已按预期启动。

创建存储系统后端

完成 Astra Trident 操作员安装后，您必须为所使用的特定 NetApp 存储平台配置后端。请访问以下链接继续设

置和配置 Astra Trident 。

- "NetApp ONTAP NFS"
- "NetApp ONTAP iSCSI"
- "NetApp Element iSCSI"

NetApp ONTAP NFS 配置

要启用 Trident 与 NetApp ONTAP 存储系统的集成，您必须创建一个后端，以便与存储系统进行通信。

1. 下载的安装归档中提供了 `sample-input` folder 层次结构中的示例后端文件。对于提供 NFS 的 NetApp ONTAP 系统，将 `backend-ontap-nas.json` 文件复制到您的工作目录并编辑该文件。

```
[netapp-user@rhel7 trident-installer]$ cp sample-input/backends-
samples/ontap-nas/backend-ontap-nas.json ./
[netapp-user@rhel7 trident-installer]$ vi backend-ontap-nas.json
```

2. 编辑 backendName ， managementLIF ， dataLIF ， SVM ，用户名， 和密码值。

```
{
    "version": 1,
    "storageDriverName": "ontap-nas",
    "backendName": "ontap-nas+10.61.181.221",
    "managementLIF": "172.21.224.201",
    "dataLIF": "10.61.181.221",
    "svm": "trident_svm",
    "username": "cluster-admin",
    "password": "password"
}
```

> 最佳做法是，将自定义 backendName 值定义为 storageDriverName 和为 NFS 提供服务的 dataLIF 的组合，以便于识别。

3. 安装此后端文件后，运行以下命令以创建第一个后端。

```
[netapp-user@rhel7 trident-installer]$ ./tridentctl -n trident create
backend -f backend-ontap-nas.json
+------------------------+----------------
+--------------------------------------+--------+---------+
|          NAME          | STORAGE DRIVER |                  UUID
| STATE  | VOLUMES |
+------------------------+----------------
+--------------------------------------+--------+---------+
| ontap-nas+10.61.181.221 | ontap-nas      | be7a619d-c81d-445c-b80c-
5c87a73c5b1e | online |       0 |
+------------------------+----------------
+--------------------------------------+--------+---------+
```

4. 创建后端后，您接下来必须创建一个存储类。与后端一样，可以在 sample-inputs 文件夹中为环境编辑一个示例存储类文件。将其复制到工作目录并进行必要的编辑，以反映所创建的后端。

```
[netapp-user@rhel7 trident-installer]$ cp sample-input/storage-class-
samples/storage-class-csi.yaml.templ ./storage-class-basic.yaml
[netapp-user@rhel7 trident-installer]$ vi storage-class-basic.yaml
```

5. 必须对此文件进行的唯一编辑是，为新创建的后端存储驱动程序的名称定义 `backendType` 值。另请注意 name-field 值，稍后必须引用该值。

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

> 此文件中定义了一个名为 `FSType` 的可选字段。可以在 NFS 后端删除此行。

6. 运行 `oc` 命令以创建存储类。

```
[netapp-user@rhel7 trident-installer]$ oc create -f storage-class-
basic.yaml
storageclass.storage.k8s.io/basic-csi created
```

7. 创建存储类后，您必须创建第一个永久性卷请求（ PVC ）。此外，还可以在 sample-inputs 中使用一个示例 `pva-basic 。 yaml` file 来执行此操作。

```
[netapp-user@rhel7 trident-installer]$ cp sample-input/pvc-samples/pvc-
basic.yaml ./
[netapp-user@rhel7 trident-installer]$ vi pvc-basic.yaml
```

8. 必须对此文件进行的唯一编辑是，确保 `storageClassName` 字段与刚刚创建的字段匹配。可以根据要配置的工作负载的需要进一步自定义 PVC 定义。

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

9. 发出 `oc` 命令创建 PVC 。根据所创建的后备卷的大小，创建可能需要一些时间，因此您可以在该过程完成后进行观察。

```
[netapp-user@rhel7 trident-installer]$ oc create -f pvc-basic.yaml
persistentvolumeclaim/basic created

[netapp-user@rhel7 trident-installer]$ oc get pvc
NAME    STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS   AGE
basic   Bound    pvc-b4370d37-0fa4-4c17-bd86-94f96c94b42d   1Gi
RWO            basic-csi      7s
```

NetApp ONTAP iSCSI 配置

要启用 Trident 与 NetApp ONTAP 存储系统的集成，您必须创建一个后端，以便与存储系统进行通信。

1. 下载的安装归档中提供了 `sample-input` folder 层次结构中的示例后端文件。对于提供 iSCSI 的 NetApp ONTAP 系统，将 `backend-ontap-san.json` 文件复制到您的工作目录并编辑该文件。

```
[netapp-user@rhel7 trident-installer]$ cp sample-input/backends-
samples/ontap-san/backend-ontap-san.json ./
[netapp-user@rhel7 trident-installer]$ vi backend-ontap-san.json
```

2. 编辑此文件中的 managementLIF ， dataLIF ， SVM ，用户名和密码值。

```
{
  "version": 1,
  "storageDriverName": "ontap-san",
  "managementLIF": "172.21.224.201",
  "dataLIF": "10.61.181.240",
  "svm": "trident_svm",
  "username": "admin",
  "password": "password"
}
```

3. 安装此后端文件后，运行以下命令以创建第一个后端。

```
[netapp-user@rhel7 trident-installer]$ ./tridentctl -n trident create
backend -f backend-ontap-san.json
+------------------------+----------------
+--------------------------------------+--------+---------+
|          NAME          | STORAGE DRIVER |                  UUID
| STATE  | VOLUMES |
+------------------------+----------------
+--------------------------------------+--------+---------+
| ontapsan_10.61.181.241 | ontap-san      | 6788533c-7fea-4a35-b797-
fb9bb3322b91 | online |       0 |
+------------------------+----------------
+--------------------------------------+--------+---------+
```

4. 创建后端后，您接下来必须创建一个存储类。与后端一样，可以在 sample-inputs 文件夹中为环境编辑一个示例存储类文件。将其复制到工作目录并进行必要的编辑，以反映所创建的后端。

```
[netapp-user@rhel7 trident-installer]$ cp sample-input/storage-class-
samples/storage-class-csi.yaml.templ ./storage-class-basic.yaml
[netapp-user@rhel7 trident-installer]$ vi storage-class-basic.yaml
```

5. 必须对此文件进行的唯一编辑是，为新创建的后端存储驱动程序的名称定义 `backendType` 值。另请注意 name-field 值，稍后必须引用该值。

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

> 此文件中定义了一个名为 `FSType` 的可选字段。在 iSCSI 后端，可以将此值设置为特定的 Linux 文件系统类型（ XFS ， ext4 等），也可以删除此值，以便 OpenShift 决定要使用的文件系统。

6. 运行 `oc` 命令以创建存储类。

```
[netapp-user@rhel7 trident-installer]$ oc create -f storage-class-basic.yaml
storageclass.storage.k8s.io/basic-csi created
```

7. 创建存储类后，您必须创建第一个永久性卷请求（ PVC ）。此外，还可以在 sample-inputs 中使用一个示例 `pva-basic 。 yaml` file 来执行此操作。

```
[netapp-user@rhel7 trident-installer]$ cp sample-input/pvc-samples/pvc-basic.yaml ./
[netapp-user@rhel7 trident-installer]$ vi pvc-basic.yaml
```

8. 必须对此文件进行的唯一编辑是，确保 `storageClassName` 字段与刚刚创建的字段匹配。可以根据要配置的工作负载的需要进一步自定义 PVC 定义。

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

9. 发出 `oc` 命令创建 PVC 。根据所创建的后备卷的大小，创建可能需要一些时间，因此您可以在该过程完成后进行观察。

```
[netapp-user@rhel7 trident-installer]$ oc create -f pvc-basic.yaml
persistentvolumeclaim/basic created

[netapp-user@rhel7 trident-installer]$ oc get pvc
NAME    STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS   AGE
basic   Bound    pvc-7ceac1ba-0189-43c7-8f98-094719f7956c   1Gi
RWO            basic-csi      3s
```

NetApp Element iSCSI 配置

要启用 Trident 与 NetApp Element 存储系统的集成，您必须创建一个后端，以便使用 iSCSI 协议与存储系统进行通信。

1. 下载的安装归档中提供了 `sample-input` folder 层次结构中的示例后端文件。对于提供 iSCSI 服务的 NetApp Element 系统，将 `backend-solidfire.json` 文件复制到您的工作目录中，然后编辑该文件。

```
[netapp-user@rhel7 trident-installer]$ cp sample-input/backends-
samples/solidfire/backend-solidfire.json ./
[netapp-user@rhel7 trident-installer]$ vi ./backend-solidfire.json
```

   a. 编辑 `endpoint` 行上的用户，密码和 MVIP 值。
   b. 编辑 `SVIP` 值。

```
 {
    "version": 1,
    "storageDriverName": "solidfire-san",
    "Endpoint": "https://trident:password@172.21.224.150/json-
rpc/8.0",
    "SVIP": "10.61.180.200:3260",
    "TenantName": "trident",
    "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. 安装好此后端文件后，运行以下命令创建第一个后端。

```
[netapp-user@rhel7 trident-installer]$ ./tridentctl -n trident create
backend -f backend-solidfire.json
+------------------------+----------------
+--------------------------------------+--------+---------+
|          NAME          | STORAGE DRIVER |                  UUID
| STATE  | VOLUMES |
+------------------------+----------------
+--------------------------------------+--------+---------+
| solidfire_10.61.180.200 | solidfire-san  | b90783ee-e0c9-49af-8d26-
3ea87ce2efdf | online |       0 |
+------------------------+----------------
+--------------------------------------+--------+---------+
```

3. 创建后端后，您接下来必须创建一个存储类。与后端一样，可以在 sample-inputs 文件夹中为环境编辑一个示例存储类文件。将其复制到工作目录并进行必要的编辑，以反映所创建的后端。

```
[netapp-user@rhel7 trident-installer]$ cp sample-input/storage-class-
samples/storage-class-csi.yaml.templ ./storage-class-basic.yaml
[netapp-user@rhel7 trident-installer]$ vi storage-class-basic.yaml
```

4. 必须对此文件进行的唯一编辑是，为新创建的后端存储驱动程序的名称定义 `backendType` 值。另请注意 name-field 值，稍后必须引用该值。

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

> 此文件中定义了一个名为 `FSType` 的可选字段。在 iSCSI 后端，可以将此值设置为特定的 Linux 文件系统类型（ XFS， ext4 等），也可以将其删除以允许 OpenShift 决定要使用的文件系统。

5. 运行 `oc` 命令以创建存储类。

```
[netapp-user@rhel7 trident-installer]$ oc create -f storage-class-
basic.yaml
storageclass.storage.k8s.io/basic-csi created
```

6. 创建存储类后，您必须创建第一个永久性卷请求（ PVC ）。此外，还可以在 sample-inputs 中使用一个示

例 `pva-basic 。 yaml` file 来执行此操作。

```
[netapp-user@rhel7 trident-installer]$ cp sample-input/pvc-samples/pvc-
basic.yaml ./
[netapp-user@rhel7 trident-installer]$ vi pvc-basic.yaml
```

7. 必须对此文件进行的唯一编辑是，确保 `storageClassName` 字段与刚刚创建的字段匹配。可以根据要配置的工作负载的需要进一步自定义 PVC 定义。

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

8. 发出 `oc` 命令创建 PVC 。根据所创建的后备卷的大小，创建可能需要一些时间，因此您可以在该过程完成后进行观察。

```
[netapp-user@rhel7 trident-installer]$ oc create -f pvc-basic.yaml
persistentvolumeclaim/basic created

[netapp-user@rhel7 trident-installer]$ oc get pvc
NAME    STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS   AGE
basic   Bound    pvc-3445b5cc-df24-453d-a1e6-b484e874349d   1Gi
RWO            basic-csi      5s
```