



NetApp 存储集成概述

NetApp Solutions

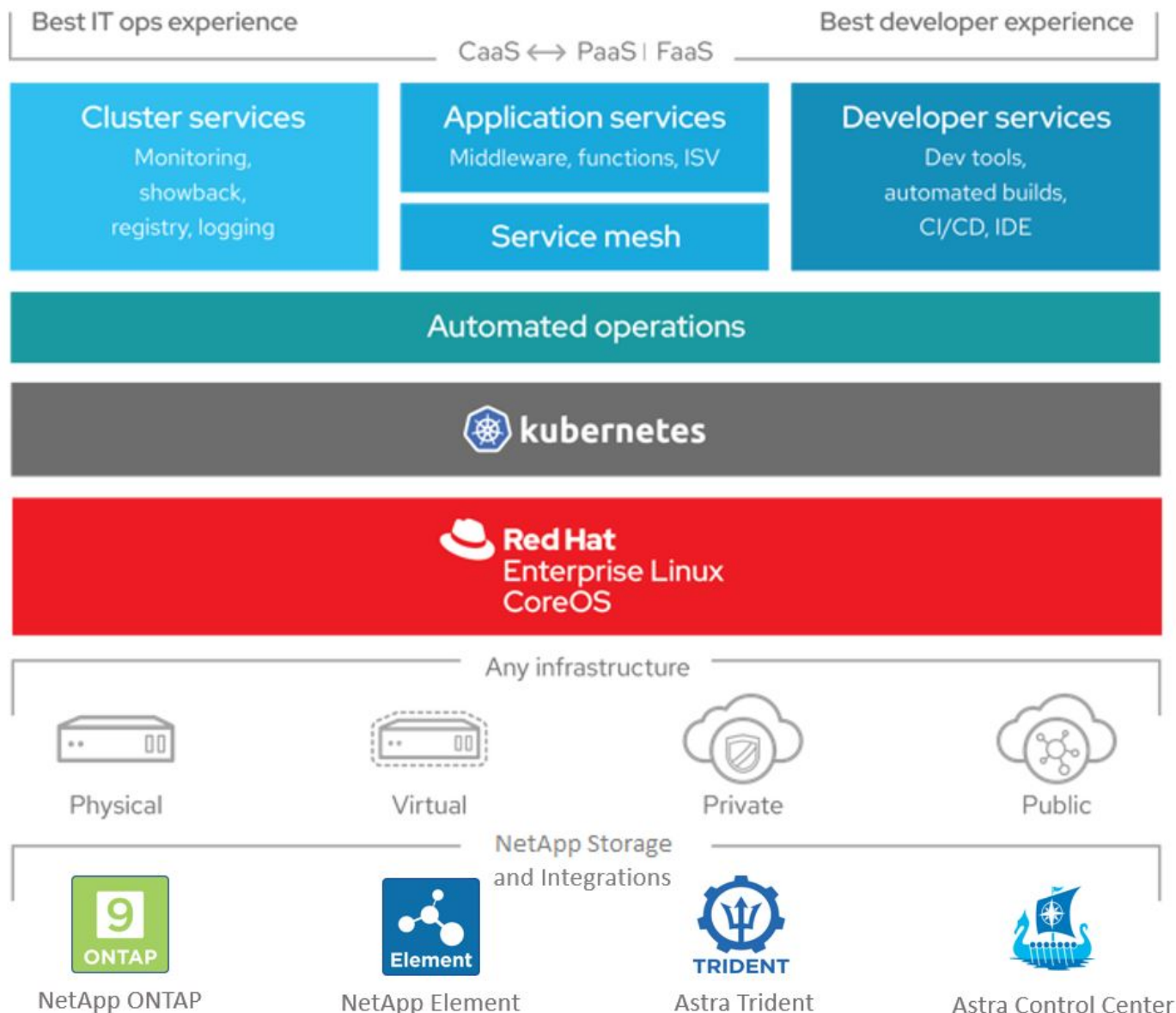
NetApp
April 12, 2024

目录

- NetApp 存储集成概述 1
 - NetApp Astra 控制中心概述 2
 - Astra Trident 概述 29

NetApp 存储集成概述

NetApp 提供了许多产品，可帮助您在基于容器的环境中编排和管理永久性数据，例如 Red Hat OpenShift。



NetApp Astra Control 采用 NetApp 数据保护技术，为有状态 Kubernetes 工作负载提供丰富的存储和应用程序感知型数据管理服务。Astra 控制服务可用于在云原生 Kubernetes 部署中支持有状态工作负载。Astra 控制中心可支持内部部署中的有状态工作负载，例如 Red Hat OpenShift。有关详细信息，请访问 NetApp Astra Control 网站 ["此处"](#)。

NetApp 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。有关详细信息，请访问 Astra Trident 网站 ["此处"](#)。

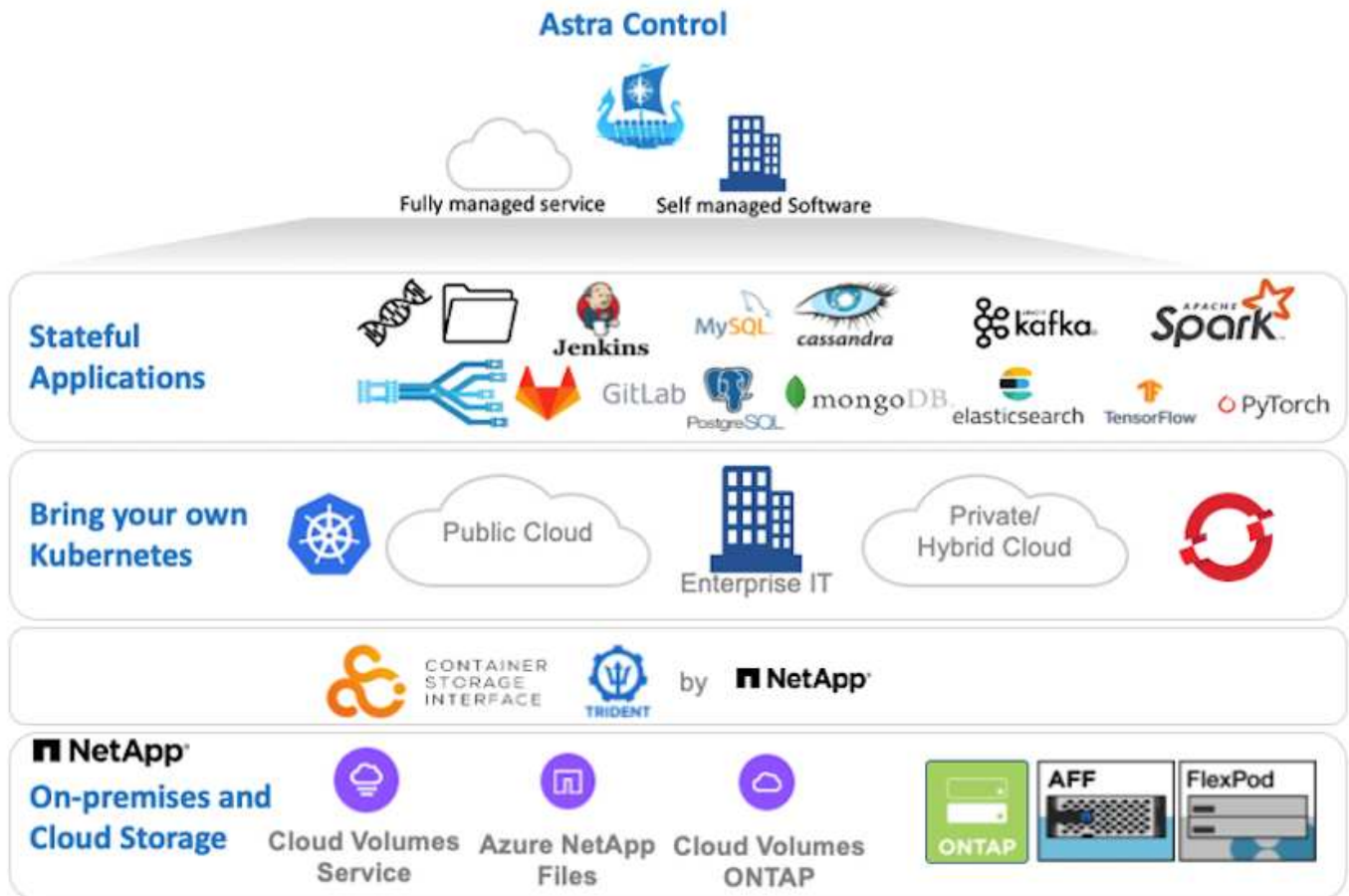
以下页面介绍了有关已在 Red Hat OpenShift with NetApp 解决方案中针对应用程序和永久性存储管理进行验证的 NetApp 产品的追加信息：

- ["NetApp Astra 控制中心"](#)

- "NetApp Astra Trident"

NetApp Astra 控制中心概述

NetApp Astra 控制中心为部署在内部环境中并采用 NetApp 数据保护技术的有状态 Kubernetes 工作负载提供丰富的存储和应用程序感知型数据管理服务。



NetApp Astra 控制中心可以安装在 Red Hat OpenShift 集群上，该集群已部署 Astra Trident 存储编排程序并为其配置存储类和存储后端到 NetApp ONTAP 存储系统。

有关安装和配置 Astra Trident 以支持 Astra 控制中心的信息，请参见 ["本文档在此处提供"](#)。

在云互联环境中，Astra 控制中心使用 Cloud Insights 提供高级监控和遥测功能。在没有 Cloud Insights 连接的情况下，可以使用有限的监控和遥测（7 天的指标），并通过开放式指标端点导出到 Kubernetes 原生监控工具（Prometheus 和 Grafana）。

Astra 控制中心完全集成到 NetApp AutoSupport 和 Active IQ 生态系统中，可为用户提供支持，协助进行故障排除以及显示使用情况统计信息。

除了已付费版本的 Astra 控制中心之外，还提供 90 天评估许可证。评估版可通过电子邮件和社区（Slack 通道）获得支持。客户可以访问这些以及其他知识库文章以及产品支持信息板上提供的文档。

要开始使用 NetApp Astra 控制中心，请访问 ["Astra 网站"](#)。

安装 Astra 控制中心的前提条件

1. 一个或多个 Red Hat OpenShift 集群。目前支持版本 4.6 EUS 和 4.7 。
2. 必须已在每个 Red Hat OpenShift 集群上安装和配置 Astra Trident 。
3. 运行 ONTAP 9.5 或更高版本的一个或多个 NetApp ONTAP 存储系统。



最佳做法是，在站点上安装的每个 OpenShift 都要有一个专用的 SVM 来用于永久性存储。多站点部署需要额外的存储系统。

4. 必须在每个 OpenShift 集群上配置一个 Trident 存储后端，其中包含一个由 ONTAP 集群提供支持的 SVM 。
5. 在每个 OpenShift 集群上配置的默认 StorageClass ， 其中使用 Astra Trident 作为存储配置程序。
6. 必须在每个 OpenShift 集群上安装和配置负载均衡器，以实现负载均衡并公开 OpenShift 服务。



请参见链接 "[此处](#)" 有关已为此目的验证的负载均衡器的信息。

7. 必须配置私有映像注册表以托管 NetApp Astra Control Center 映像。



请参见链接 "[此处](#)" 为此安装和配置 OpenShift 专用注册表。

8. 您必须对 Red Hat OpenShift 集群具有集群管理员访问权限。
9. 您必须对 NetApp ONTAP 集群具有管理员访问权限。
10. 一个管理工作站，其中安装了 Docker 或 podman ， tridentctl 以及 oc 或 kubectl 工具，并将其添加到 \$path 中。



Docker 安装的 Docker 版本必须大于 20.10 ， 而 Podman 安装的 Podman 版本必须大于 3.0 。

安装 Astra 控制中心

使用 OperatorHub

1. 登录到 NetApp 支持站点并下载最新版本的 NetApp Astra 控制中心。为此，您需要在 NetApp 帐户中附加许可证。下载完 tarball 后，将其传输到管理工作站。



要开始获取 Astra Control 的试用许可证，请访问 "[Astra 注册站点](#)"。

2. 打开 tar ball 的包装并将工作目录更改为生成的文件夹。

```
[netapp-user@rhel7 ~]$ tar -vxzf astra-control-center-21.12.60.tar.gz
[netapp-user@rhel7 ~]$ cd astra-control-center-21.12.60
```

3. 开始安装之前，请将 Astra Control Center 映像推送到映像注册表。您可以选择使用 Docker 或 Podman 执行此操作，此步骤将提供这两者的说明。

Podman

- a. 将 're' 名称为组织 / 命名空间 / 项目的注册表 FQDN 导出为环境变量 "registry"。

```
[netapp-user@rhel7 ~]$ export REGISTRY=astra-registry.apps.ocp-vmw.cie.netapp.com/netapp-astra
```

- b. 登录到注册表。

```
[netapp-user@rhel7 ~]$ podman login -u ocp-user -p password --tls-verify=false astra-registry.apps.ocp-vmw.cie.netapp.com
```



如果使用 `kubeadmin user` 登录到专用注册表，请使用 `token` 代替 `password` - `podman login -u Ocp-user -p token -tls-verify=false astra-registry.apps.ocp-vmw.cie.netapp.com`。



或者，您也可以创建服务帐户，分配注册表编辑器和 / 或注册表查看器角色（取决于您是否需要推 / 拉访问），并使用服务帐户的令牌登录到注册表。

- c. 创建 Shell 脚本文件并将以下内容粘贴到其中。

```
[netapp-user@rhel7 ~]$ vi push-images-to-registry.sh

for astraImageFile in $(ls images/*.tar) ; do
    # Load to local cache. And store the name of the loaded
    image trimming the 'Loaded images: '
    astraImage=$(podman load --input ${astraImageFile} | sed
's/Loaded image(s): //' )
    astraImage=$(echo ${astraImage} | sed 's!localhost/!!')
    # Tag with local image repo.
    podman tag ${astraImage} ${REGISTRY}/${astraImage}
    # Push to the local repo.
    podman push ${REGISTRY}/${astraImage}
done
```



如果您的注册表使用的是不可信的证书，请编辑 shell 脚本并对 podman 推送命令 `podman 推送 $registry/$ (echo $astraImage ` s/^^``
.....
.....

- d. 使文件可执行

```
[netapp-user@rhel7 ~]$ chmod +x push-images-to-registry.sh
```

e. 执行 shell 脚本。

```
[netapp-user@rhel7 ~]$ ./push-images-to-registry.sh
```


Docker

- a. 将 're名称为组织 / 命名空间 / 项目的注册表 FQDN 导出为环境变量 "gregistry"。

```
[netapp-user@rhel7 ~]$ export REGISTRY=astra-registry.apps.ocp-vmw.cie.netapp.com/netapp-astra
```

- b. 登录到注册表。

```
[netapp-user@rhel7 ~]$ docker login -u ocp-user -p password astra-registry.apps.ocp-vmw.cie.netapp.com
```



如果使用 `kubeadmin user` 登录到专用注册表，请使用 `token` 代替 `password` - `docker login -u Ocp-user -p token astra-registry.apps.ocp-vmw.cie.netapp.com`。



或者，您也可以创建服务帐户，分配注册表编辑器和 / 或注册表查看器角色（取决于您是否需要推 / 拉访问），并使用服务帐户的令牌登录到注册表。

- c. 创建 Shell 脚本文件并将以下内容粘贴到其中。

```
[netapp-user@rhel7 ~]$ vi push-images-to-registry.sh

for astraImageFile in $(ls images/*.tar) ; do
    # Load to local cache. And store the name of the loaded
    image trimming the 'Loaded images: '
    astraImage=$(docker load --input ${astraImageFile} | sed
's/Loaded image: //' )
    astraImage=$(echo ${astraImage} | sed 's!localhost/!!')
    # Tag with local image repo.
    docker tag ${astraImage} ${REGISTRY}/${astraImage}
    # Push to the local repo.
    docker push ${REGISTRY}/${astraImage}
done
```

- d. 使文件可执行

```
[netapp-user@rhel7 ~]$ chmod +x push-images-to-registry.sh
```

- e. 执行 shell 脚本。

```
[netapp-user@rhel7 ~]$ ./push-images-to-registry.sh
```

4. 使用非公共信任的私有映像注册表时，请将映像注册表 TLS 证书上传到 OpenShift 节点。为此，请使用 TLS 证书在 OpenShift-config 命名空间中创建一个配置映射，并将其修补到集群映像配置中以使此证书可信。

```
[netapp-user@rhel7 ~]$ oc create configmap default-ingress-ca -n  
openshift-config --from-file=astra-registry.apps.ocp  
-vmw.cie.netapp.com=tls.crt  
  
[netapp-user@rhel7 ~]$ oc patch image.config.openshift.io/cluster  
--patch '{"spec":{"additionalTrustedCA":{"name":"default-ingress-  
ca"}}}' --type=merge
```



如果您使用的是包含传入操作员的默认 TLS 证书的 OpenShift 内部注册表和路由，则仍需要按照上一步将这些证书修补到路由主机名。要从运算符提取证书，您可以使用命令 `oc extract secret/router -ca -keys=tls.crt -n OpenShift-Inuse-operator`。

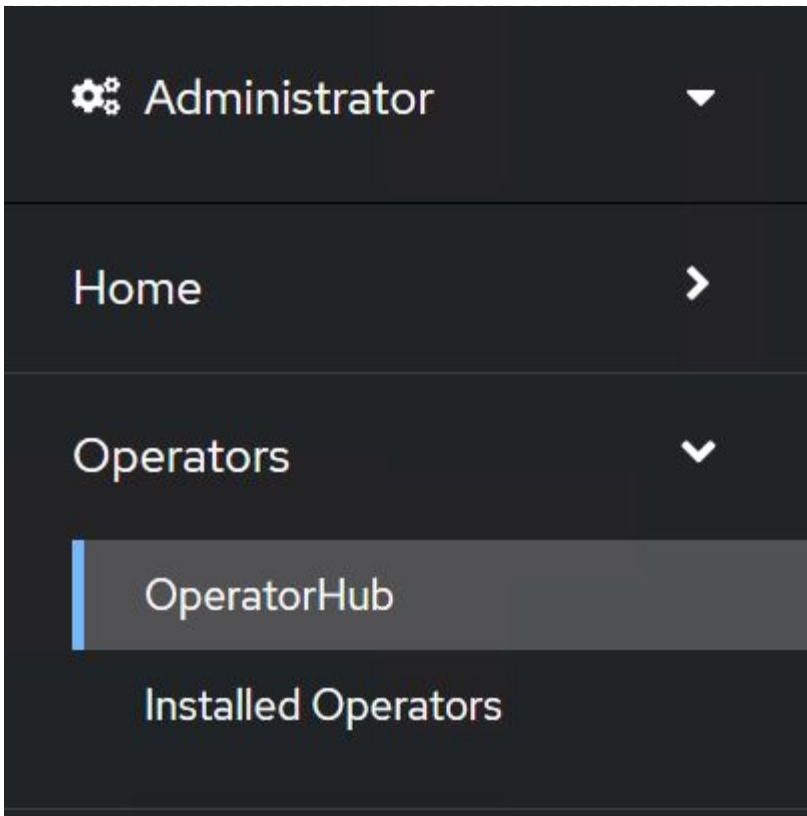
5. 为 Astra 控制中心创建命名空间 NetApp-Acc-operator。

```
[netapp-user@rhel7 ~]$ oc create ns netapp-acc-operator  
  
namespace/netapp-acc-operator created
```


6. 使用凭据创建一个密钥，以登录到 NetApp-Acc-operator 命名空间中的映像注册表。

```
[netapp-user@rhel7 ~]$ oc create secret docker-registry astra-  
registry-cred --docker-server=astra-registry.apps.ocp  
-vmw.cie.netapp.com --docker-username=ocp-user --docker  
-password=password -n netapp-acc-operator  
  
secret/astra-registry-cred created
```

7. 使用 cluster-admin 访问权限登录到 Red Hat OpenShift GUI 控制台。
8. 从 "Perspective" 下拉列表中选择 "Administrator"。
9. 导航到 Operators > OperatorHub 并搜索 Astra。



10. 选择 NetApp-Acc-operator Tile ，然后单击 Install 。

**netapp-acc-operator**21.12.63-1 provided by NetApp✕

Install

Latest version
21.12.63-1

Capability level
☒ Basic Install
☐ Seamless Upgrades
☐ Full Lifecycle
☐ Deep Insights
☐ Auto Pilot

Provider type
Certified

Provider
NetApp

Astra Control is an application-aware data management solution that manages, protects and moves data-rich Kubernetes workloads in both public clouds and on-premises.

Astra Control enables data protection, disaster recovery, and migration for your Kubernetes workloads, leveraging NetApp's industry-leading data management technology for snapshots, backups, replication and cloning.

How to deploy Astra Control

Refer to [Installation Procedure](#) to deploy Astra Control Center using the Operator.

Documentation

Refer to [Astra Control Center Documentation](#) to complete the setup and start managing applications.

11. 在 Install Operator 屏幕上，接受所有默认参数，然后单击 Install 。

Install Operator

Install your Operator by subscribing to one of the update channels to keep the Operator up to date. The strategy determines either manual or automatic updates.

Update channel *

- ☐ alpha
- ☒ stable

Installation mode *

- ☒ All namespaces on the cluster (default)
Operator will be available in all Namespaces.
- ☐ A specific namespace on the cluster
This mode is not supported by this Operator

Installed Namespace *

PR netapp-acc-operator (Operator recommended)

⚠ Namespace already exists

Namespace **netapp-acc-operator** already exists and will be used. Other users can already have access to this namespace.

Approval strategy *

- ☒ Automatic
- ☐ Manual

Install

Cancel

 **netapp-acc-operator**
provided by NetApp

Provided APIs

 **Astra Control Center**

AstraControlCenter is the Schema for the astracontrolcenters API

12. 等待操作员安装完成。



netapp-acc-operator
21.12.63-1 provided by NetApp



Installing Operator

InstallWaiting: installing: waiting for deployment acc-operator-controller-manager to become ready: Waiting for rollout to finish: 0 of 1 updated replicas are available...

The Operator is being installed. This may take a few minutes.

[View installed Operators in Namespace netapp-acc-operator](#)

13. 操作员安装成功后，导航到单击 View Operator。



netapp-acc-operator
21.12.63-1 provided by NetApp



Installed operator - ready for use

[View Operator](#)

[View installed Operators in Namespace netapp-acc-operator](#)

14. 然后在运算符中单击 Astra Control Center 图块中的 Create Instance。

[Installed Operators](#) > [Operator details](#)



netapp-acc-operator
21.12.63-1 provided by NetApp

[Details](#)

[YAML](#)

[Subscription](#)

[Events](#)

[Astra Control Center](#)

Provided APIs

ACC Astra Control Center

AstraControlCenter is the Schema for the astracontrolcenters API

[+ Create instance](#)

15. 填写 Create AstraControlCenter Form 字段，然后单击 Create。

- 也可以编辑 Astra Control Center 实例名称。
- 也可以启用或禁用自动支持。建议保留自动支持功能。
- 输入 Astra 控制中心的 FQDN。
- 输入 Astra 控制中心版本；默认情况下会显示最新版本。

- e. 输入 Astra 控制中心的帐户名称和管理员详细信息，例如名字，姓氏和电子邮件地址。
- f. 输入卷回收策略，默认值为 Retain 。
- g. 在映像注册表中，输入注册表的 FQDN 以及在将映像推送到注册表时提供的组织名称（在此示例中为 `astra-registry.apps.ocp-vmw.cie.netapp.com/netapp-astra`）
- h. 如果您使用的注册表需要进行身份验证，请在映像注册表部分输入机密名称。
- i. 为 Astra 控制中心资源限制配置扩展选项。
- j. 如果要将 PVC 放置在非默认存储类上，请输入存储类名称。
- k. 定义 CRD 处理首选项。

Project: netapp-acc-operator ▼

Name *

Labels

Account Name *

Astra Control Center account name

Astra Address *

AstraAddress defines how Astra will be found in the data center. This IP address and/or DNS A record must be created prior to provisioning Astra Control Center. Example - "astra.example.com" The A record and its IP address must be allocated prior to provisioning Astra Control Center

Astra Version *

Version of AstraControlCenter to deploy. You are provided a Helm repository with a corresponding version. Example - 1.5.2, 1.4.2-patch

Email *

EmailAddress will be notified by Astra as events warrant.

Auto Support * >

AutoSupport indicates willingness to participate in NetApp's proactive support application, NetApp Active IQ. The default election is true and indicates support data will be sent to NetApp. An empty or blank election is the same as a default election. Air gapped installations should enter false.

First Name

The first name of the SRE supporting Astra.

Last Name

Admin

The last name of the SRE supporting Astra.

Image Registry

The container image registry that is hosting the Astra application images, ACC Operator and ACC Helm Repository.

Name

astra-registry.apps.ocp-vmw.cie.netapp.com/netapp-astra

The name of the image registry. For example "example.registry/astra". Do not prefix with protocol.

Secret

astra-registry-cred

The name of the Kubernetes secret that will authenticate with the image registry.

Volume Reclaim Policy

Retain

Reclaim policy to be set for persistent volumes

Astra Resources Scaler

Default

Scaling options for AstraControlCenter Resource limits.

Storage Class

The storage class to be used for PVCs. If not set, default storage class will be used.

Crds

Options for how ACC should handle CRDs.

Create

Cancel

自动化的〔可逆〕

1. 要使用Ansible攻略手册部署Astra控制中心、您需要安装安装有Ansible的Ubuntu或RHEL计算机。按照步骤进行操作 ["此处"](#) 适用于Ubuntu和RHEL。
2. 克隆托管 Ansible 内容的 GitHub 存储库。

```
git clone https://github.com/NetApp-
Automation/na_astra_control_suite.git
```

3. 登录到NetApp支持站点并下载最新版本的NetApp Astra控制中心。为此，您需要在 NetApp 帐户中附加许可证。下载完 tarball 后，将其传输到工作站。



要开始获取 Astra Control 的试用许可证，请访问 ["Astra 注册站点"](#)。

4. 创建或获取对要安装Astra控制中心的OpenShift集群具有管理员访问权限的kubeconfig文件。
5. 将目录更改为 na_astera_control_suite 。

```
cd na_astra_control_suite
```

6. 编辑`vars/vars.yml`文件、并使用所需信息填充变量。

```
#Define whether or not to push the Astra Control Center images to
your private registry [Allowed values: yes, no]
push_images: yes

#The directory hosting the Astra Control Center installer
installer_directory: /home/admin/

#Specify the ingress type. Allowed values - "AccTraefik" or
"Generic"
#"AccTraefik" if you want the installer to create a LoadBalancer
type service to access ACC, requires MetallB or similar.
#"Generic" if you want to create or configure ingress controller
yourself, installer just creates a ClusterIP service for traefik.
ingress_type: "AccTraefik"

#Name of the Astra Control Center installer (Do not include the
extension, just the name)
astra_tar_ball_name: astra-control-center-22.04.0

#The complete path to the kubeconfig file of the
kubernetes/openshift cluster Astra Control Center needs to be
installed to.
hosting_k8s_cluster_kubeconfig_path: /home/admin/cluster-
kubeconfig.yml

#Namespace in which Astra Control Center is to be installed
astra_namespace: netapp-astra-cc

#Astra Control Center Resources Scaler. Leave it blank if you want
to accept the Default setting.
astra_resources_scaler: Default

#Storageclass to be used for Astra Control Center PVCs, it must be
created before running the playbook [Leave it blank if you want the
PVCs to use default storageclass]
astra_trident_storageclass: basic

#Reclaim Policy for Astra Control Center Persistent Volumes [Allowed
values: Retain, Delete]
storageclass_reclaim_policy: Retain
```



```

#Private Registry Details
astra_registry_name: "docker.io"

#Whether the private registry requires credentials [Allowed values:
yes, no]
require_reg_creds: yes

#If require_reg_creds is yes, then define the container image
registry credentials
#Usually, the registry namespace and usernames are same for
individual users
astra_registry_namespace: "registry-user"
astra_registry_username: "registry-user"
astra_registry_password: "password"

#Kuberenets/OpenShift secret name for Astra Control Center
#This name will be assigned to the K8s secret created by the
playbook
astra_registry_secret_name: "astra-registry-credentials"

#Astra Control Center FQDN
acc_fqdn_address: astra-control-center.cie.netapp.com

#Name of the Astra Control Center instance
acc_account_name: ACC Account Name

#Administrator details for Astra Control Center
admin_email_address: admin@example.com
admin_first_name: Admin
admin_last_name: Admin

```

7. 运行攻略手册以部署 Astra 控制中心。对于某些配置、此攻略手册需要root特权。

如果运行该攻略手册的用户为root或配置了无密码sudo、请运行以下命令运行该攻略手册。

```
ansible-playbook install_acc_playbook.yml
```

如果用户配置了基于密码的sudo访问权限、请运行以下命令以运行攻略手册、然后输入sudo密码。

```
ansible-playbook install_acc_playbook.yml -K
```

安装后步骤

1. 完成安装可能需要几分钟时间。验证 NetApp-Astra-cc 命名空间中的所有 Pod 和服务是否均已启动且正在运行。

```
[netapp-user@rhel7 ~]$ oc get all -n netapp-astra-cc
```

2. 检查 Acc-operator-controller-manager 日志以确保安装已完成。

```
[netapp-user@rhel7 ~]$ oc logs deploy/acc-operator-controller-manager -n  
netapp-acc-operator -c manager -f
```



以下消息指示 Astra 控制中心已成功安装。

```
{"level":"info","ts":1624054318.029971,"logger":"controllers.AstraControlCenter","msg":"Successfully Reconciled AstraControlCenter in [seconds]s","AstraControlCenter":"netapp-astra-cc/astra","ae.Version":"[21.12.60]"}
```

3. 用于登录到 Astra 控制中心的用户名是 CRD 文件中提供的管理员电子邮件地址，密码是附加到 Astra 控制中心 UUID 的字符串 Acc-。运行以下命令：

```
[netapp-user@rhel7 ~]$ oc get astracontrolcenters -n netapp-astra-cc  
NAME      UUID  
astra     345c55a5-bf2e-21f0-84b8-b6f2bce5e95f
```



在此示例中，密码为 Acc-345c55a5-bf2e-21f0-84b8-b6f2bce5e95f。

4. 获取 traefik 服务负载均衡器 IP。

```
[netapp-user@rhel7 ~]$ oc get svc -n netapp-astra-cc | egrep  
'EXTERNAL|traefik'
```

NAME	TYPE	CLUSTER-IP
EXTERNAL-IP	PORT(S)	
AGE		
traefik	LoadBalancer	172.30.99.142
10.61.186.181	80:30343/TCP, 443:30060/TCP	
16m		

5. 在 DNS 服务器中添加一个条目，将 Astra 控制中心 CRD 文件中提供的 FQDN 指向 traefik 服务的

external-IP。

New Host

Name (uses parent domain name if blank):

astra-control-center

Fully qualified domain name (FQDN):

astra-control-center.cie.netapp.com.

IP address:

10.61.186.181

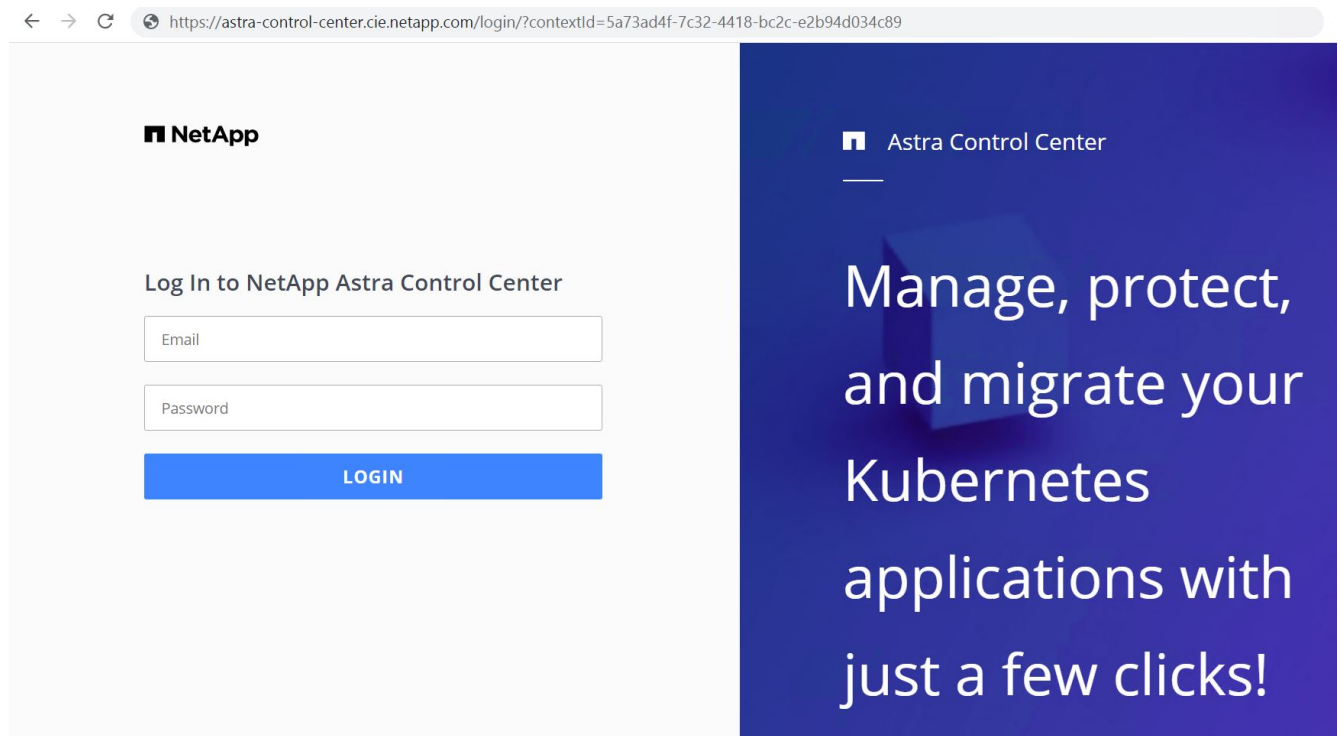
☒ Create associated pointer (PTR) record

☐ Allow any authenticated user to update DNS records with the same owner name

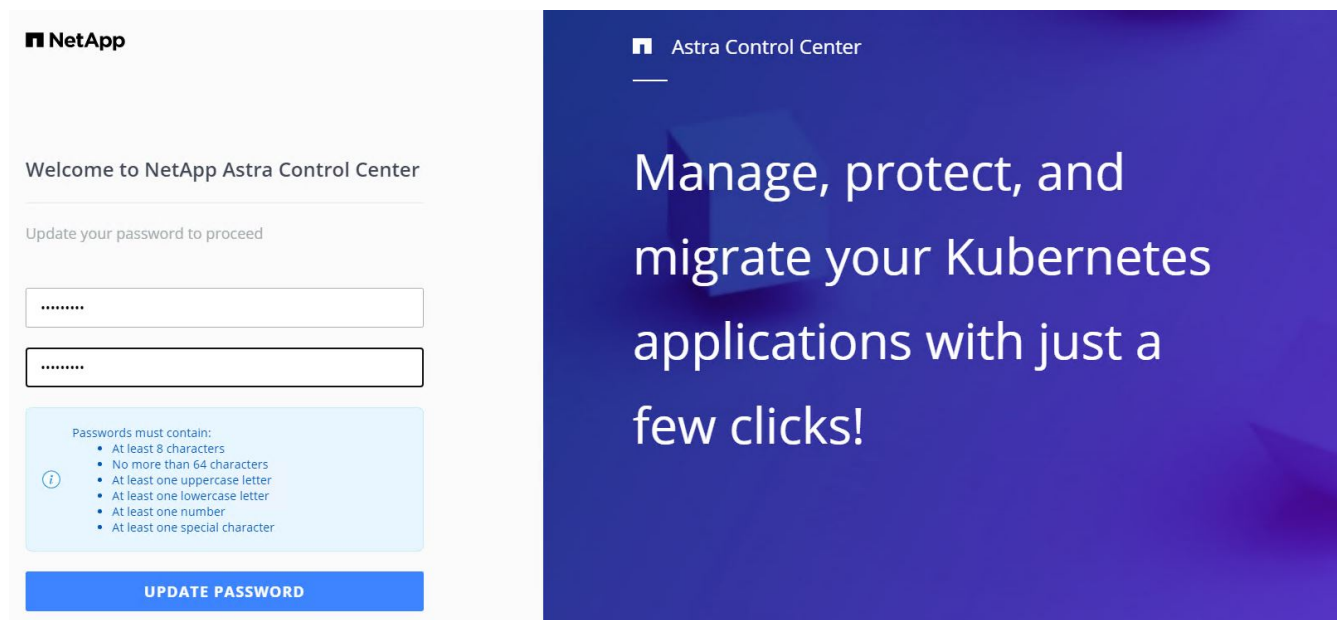
Add Host

Cancel

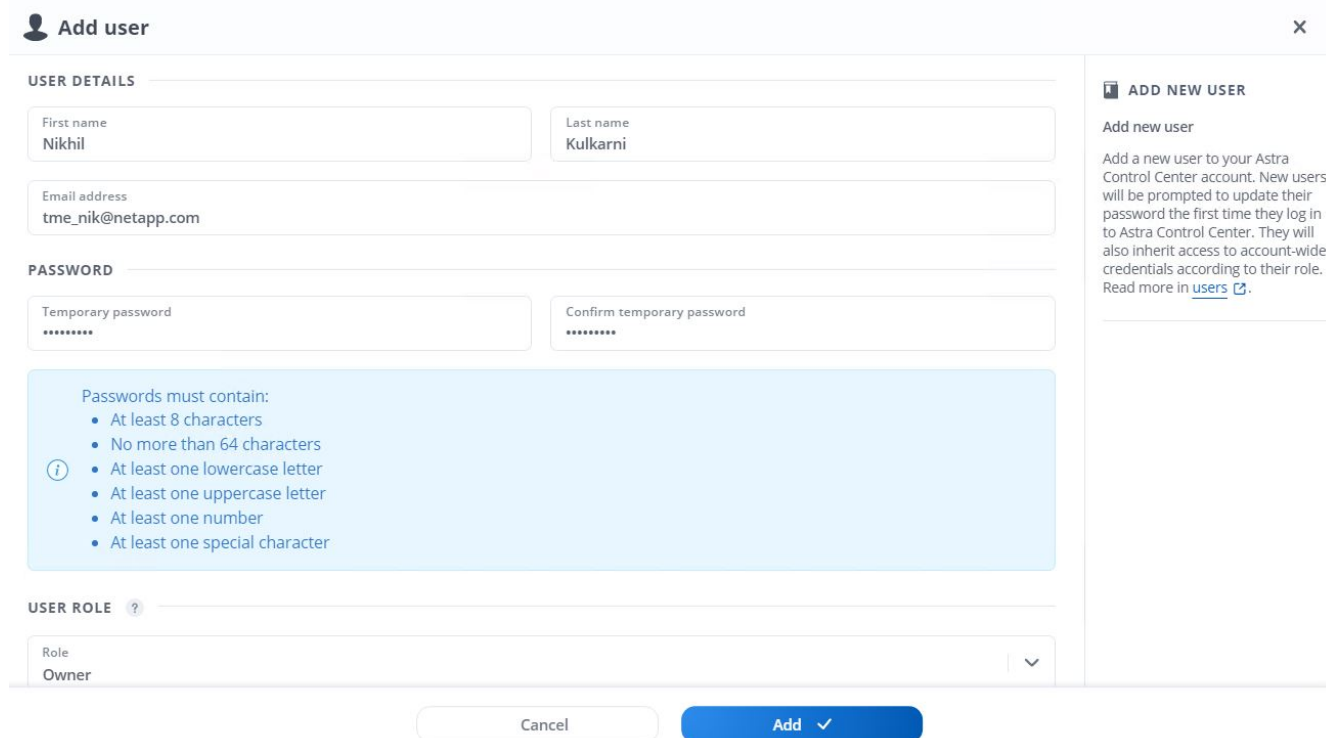
6. 通过浏览 Astra 控制中心的 FQDN 登录到该 GUI。



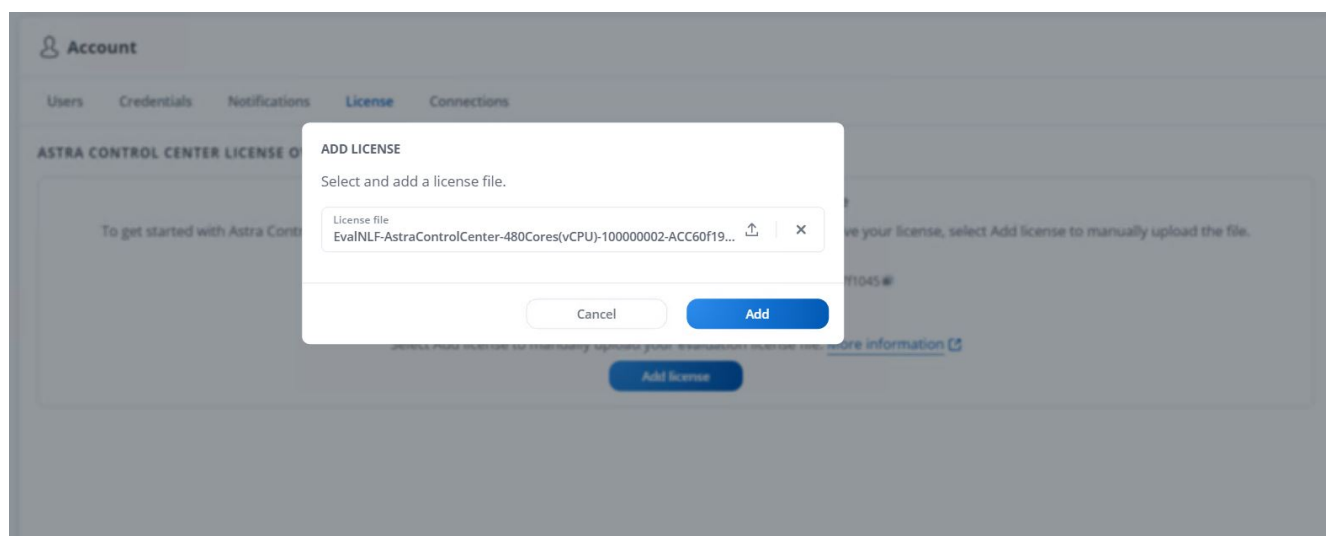
7. 首次使用 CRD 中提供的管理员电子邮件地址登录到 Astra 控制中心图形用户界面时，您需要更改密码。



8. 如果要添加用户到 Astra 控制中心，请导航到 Account > Users，单击 Add，输入用户的详细信息，然后单击 Add。



9. 要使 Astra 控制中心的所有功能正常运行，需要获得许可证。要添加许可证，请导航到 "帐户 "> "许可证 "，单击 "添加许可证 "，然后上传许可证文件。



如果您在安装或配置 NetApp Astra 控制中心时遇到问题，可以参考已知问题的知识库 ["此处"](#)。

将 Red Hat OpenShift 集群注册到 Astra 控制中心

要使 Astra 控制中心能够管理您的工作负载，您必须先注册 Red Hat OpenShift 集群。

注册 Red Hat OpenShift 集群

1. 第一步是将 OpenShift 集群添加到 Astra 控制中心并对其进行管理。转至集群并单击添加集群，上传

OpenShift 集群的 kubeconfig 文件，然后单击选择存储。

Add cluster

STEP 1/3: CREDENTIALS

X

CREDENTIALS

Provide Astra Control access to your Kubernetes and OpenShift clusters by entering a kubeconfig credential.

Follow [instructions](#) on how to create a dedicated admin-role kubeconfig.

Upload file | Paste from clipboard

Kubeconfig YAML file
ocp-vmw kubeconfig.txt

⬆ | ✕

Credential name
ocp-vmw

ADDING A CLUSTER

Adding a cluster is needed for Astra Control to discover your Kubernetes applications.

Select a cloud provider and input credentials to get started.

Read more in [Clusters](#).

Cancel

Configure storage →



可以生成 kubeconfig 文件，以便使用用户名和密码或令牌进行身份验证。令牌将在一段有限的时间后过期，并且可能会使注册的集群无法访问。NetApp 建议使用具有用户名和密码的 kubeconfig 文件将 OpenShift 集群注册到 Astra 控制中心。

2. Astra 控制中心会检测符合条件的存储类。现在，选择使用 NetApp ONTAP 上由 SVM 支持的 Trident 配置卷的 storageclass 方式，然后单击查看。在下一个窗格中，验证详细信息，然后单击 Add Cluster。

Add cluster

STEP 2/3: STORAGE

×

STORAGE

Existing storage classes are discovered and verified as eligible for use with Astra Control. You can use your existing default, or choose to set a new default at this time.

Applications with persistent volumes on eligible storage classes are validated for use with Astra Control.

Set default	Storage class	Storage provisioner	Reclaim policy	Binding mode	Eligible
<input checked="" type="radio"/>	ocp-trident <small>Default</small>	csi.trident.netapp.io	Delete	Immediate	
<input type="radio"/>	ocp-trident-iscsi	csi.trident.netapp.io	Delete	Immediate	
<input type="radio"/>	project-1-sc	csi.trident.netapp.io	Delete	Immediate	
<input type="radio"/>	thin	kubernetes.io/vsphere-volume	Delete	Immediate	

← Select credentials

Review →

- 按照步骤 1 中所述注册两个 OpenShift 集群。添加后，集群将变为 "正在发现" 状态，而 Astra 控制中心将对其进行检查并安装必要的代理。成功注册后，集群状态将更改为 "正在运行"。

admin

Dashboard

MANAGE YOUR APPS

Apps

Clusters

MANAGE YOUR STORAGE

Backends

Buckets

MANAGE YOUR ACCOUNT

Account

Activity

Support

Clusters

Actions

+ Add

Search

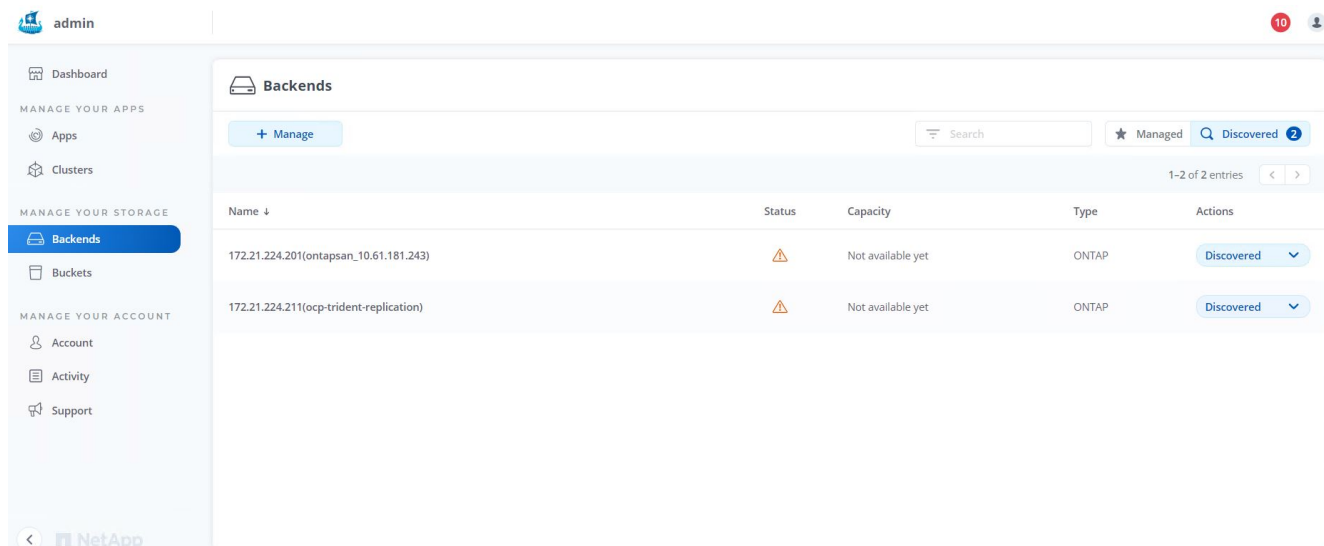
1-2 of 2 entries

<input type="checkbox"/>	Name	Ready	Type	Version	Actions
<input type="checkbox"/>	ocp-vmw		Red Hat OpenShift	v1.20.0+df9c838	Running
<input type="checkbox"/>	ocp-vmware2		Red Hat OpenShift	v1.20.0+c8905da	Running

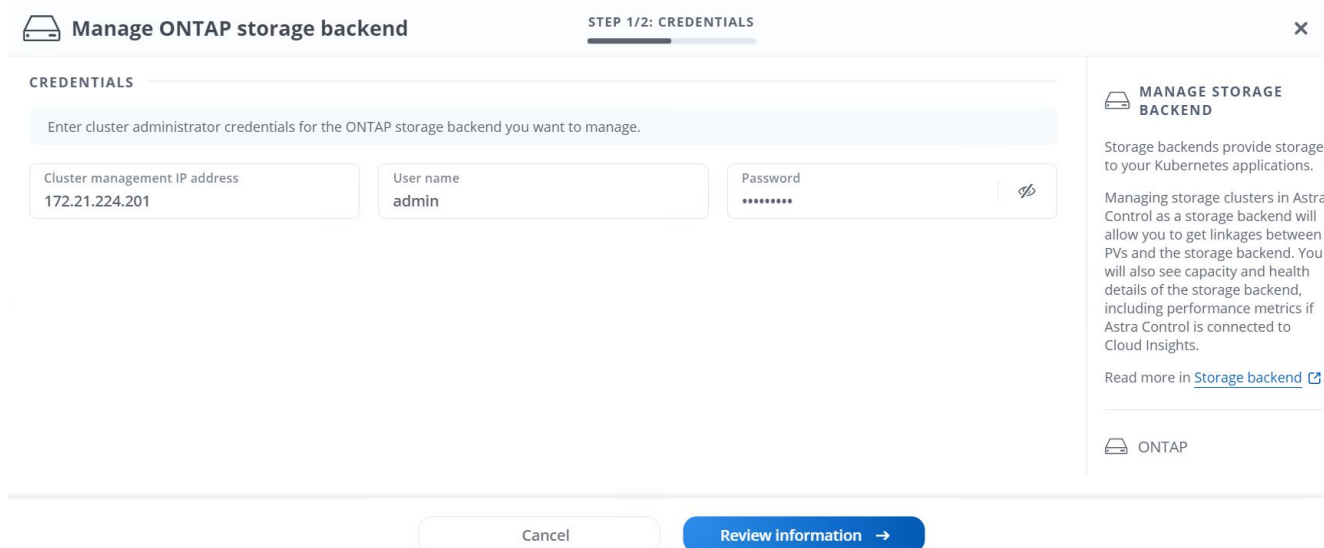


在受管集群上安装的代理从该注册表中提取映像时，由 Astra 控制中心管理的所有 Red Hat OpenShift 集群都应有权访问用于安装的映像注册表。

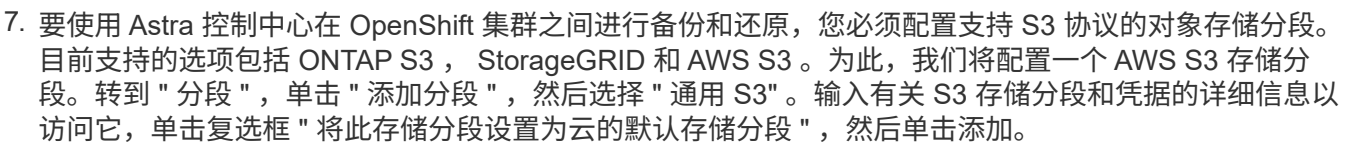
- 将 ONTAP 集群作为存储资源导入，以便由 Astra 控制中心作为后端进行管理。将 OpenShift 集群添加到 Astra 并配置了 storageclass 后，它会自动发现并检查支持该 storageclass 的 ONTAP 集群，但不会将其导入到要管理的 Astra 控制中心中。



5. 要导入 ONTAP 集群，请转到后端，单击下拉列表，然后选择要管理的 ONTAP 集群旁边的管理。输入 ONTAP 集群凭据，单击查看信息，然后单击导入存储后端。



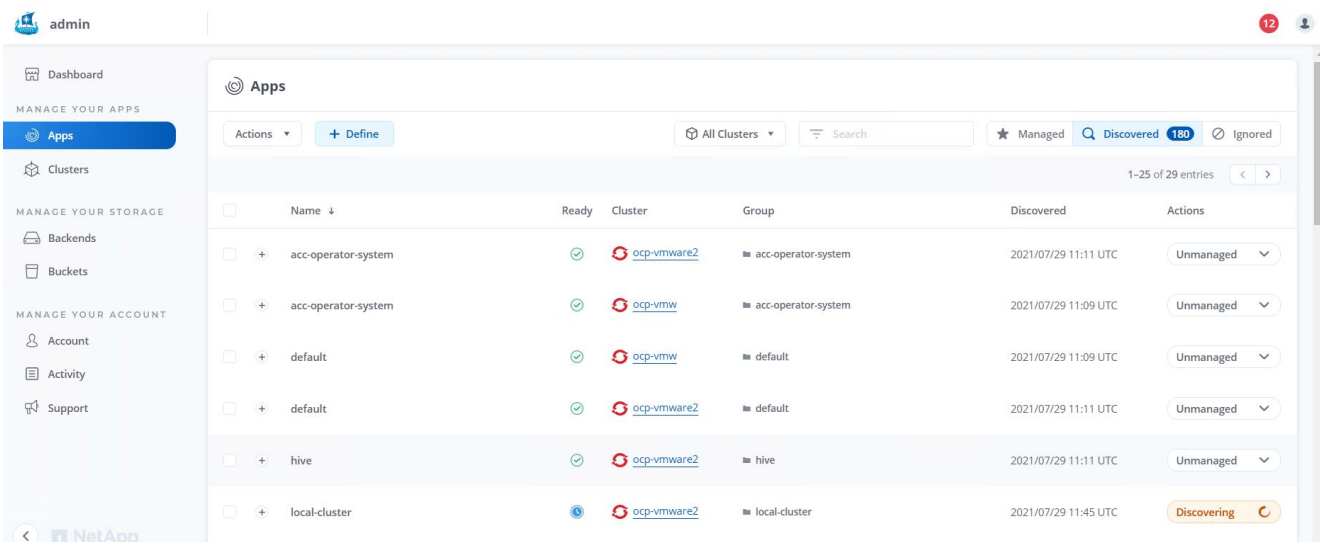
6. 添加后端后，状态将更改为 Available 。现在，这些后端可提供有关 OpenShift 集群中的永久性卷以及 ONTAP 系统上的相应卷的信息。



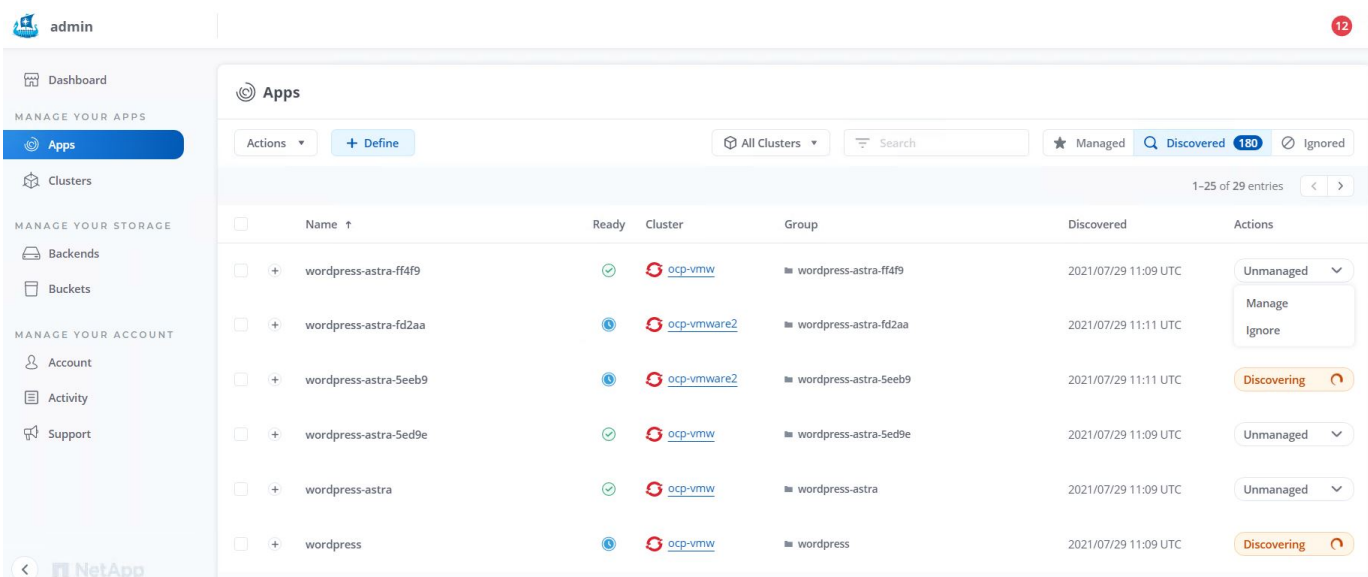
选择要保护的应用程序

管理应用程序

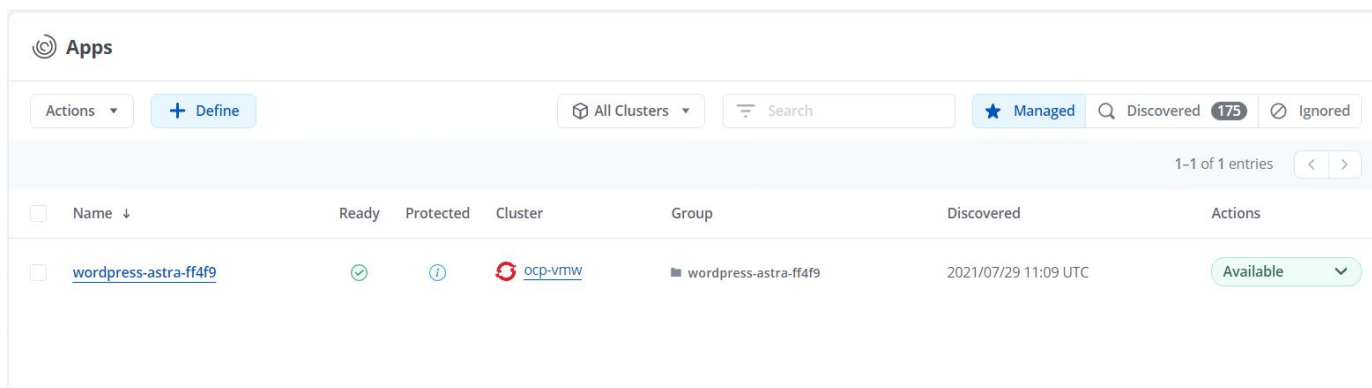
- 23



2. 导航到应用程序 > 已发现，然后单击要使用 Astra 管理的应用程序旁边的下拉菜单。然后单击管理。



1. 此应用程序将进入可用状态，并可在 "Apps" 部分的 "Managed " 选项卡下查看。



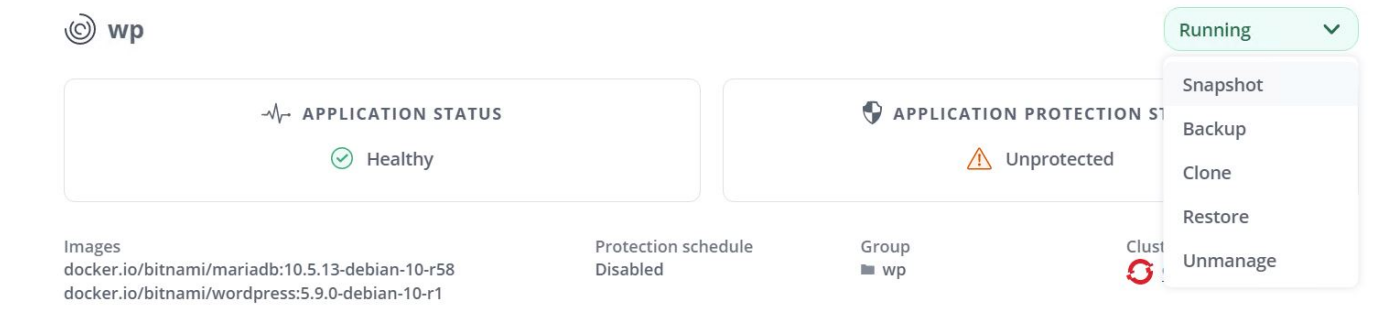
保护您的应用程序

在由 Astra 控制中心管理应用程序工作负载之后，您可以为这些工作负载配置保护设置。

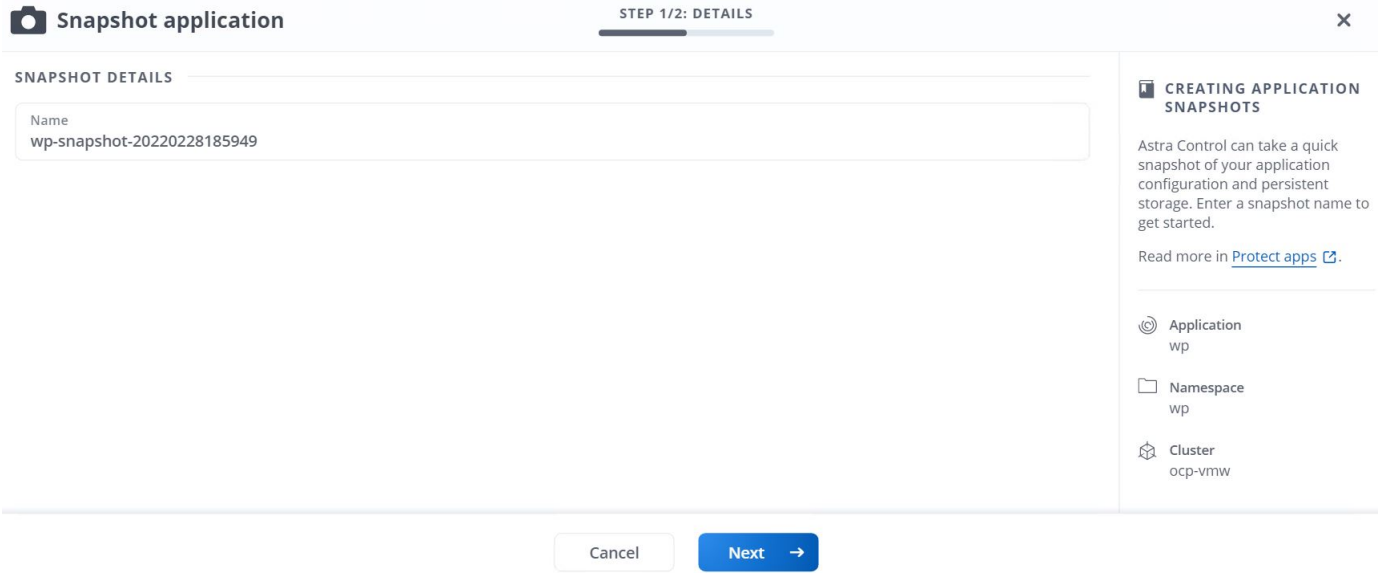
创建应用程序快照

应用程序的快照会创建一个 ONTAP Snapshot 副本，该副本可用于根据该 Snapshot 副本将应用程序还原或克隆到特定时间点。

- 1. 要为应用程序创建快照，请导航到 "Apps" > "Managed " 选项卡，然后单击要为其创建 Snapshot 副本的应用程序。单击应用程序名称旁边的下拉菜单，然后单击 Snapshot 。



- 2. 输入快照详细信息，单击下一步，然后单击 Snapshot 。创建快照大约需要一分钟，在成功创建快照后，状态将变为可用。



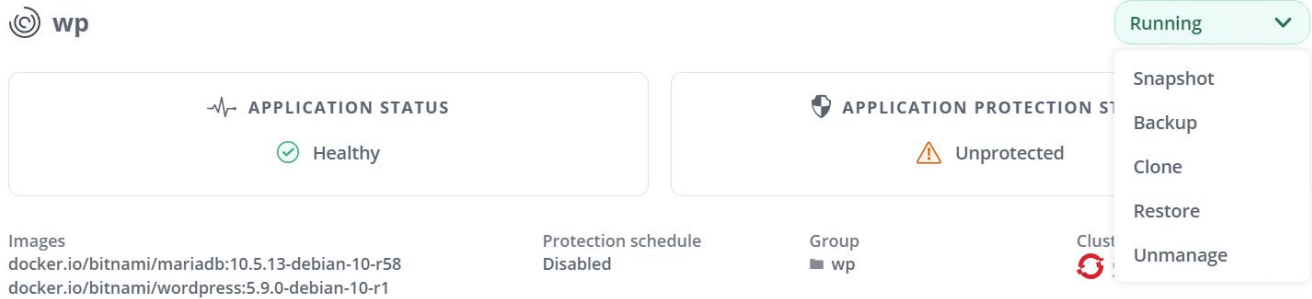
创建应用程序备份

应用程序的备份可捕获应用程序的活动状态及其资源的配置，将其覆盖到文件中，并将其存储在远程对象存储分段中。

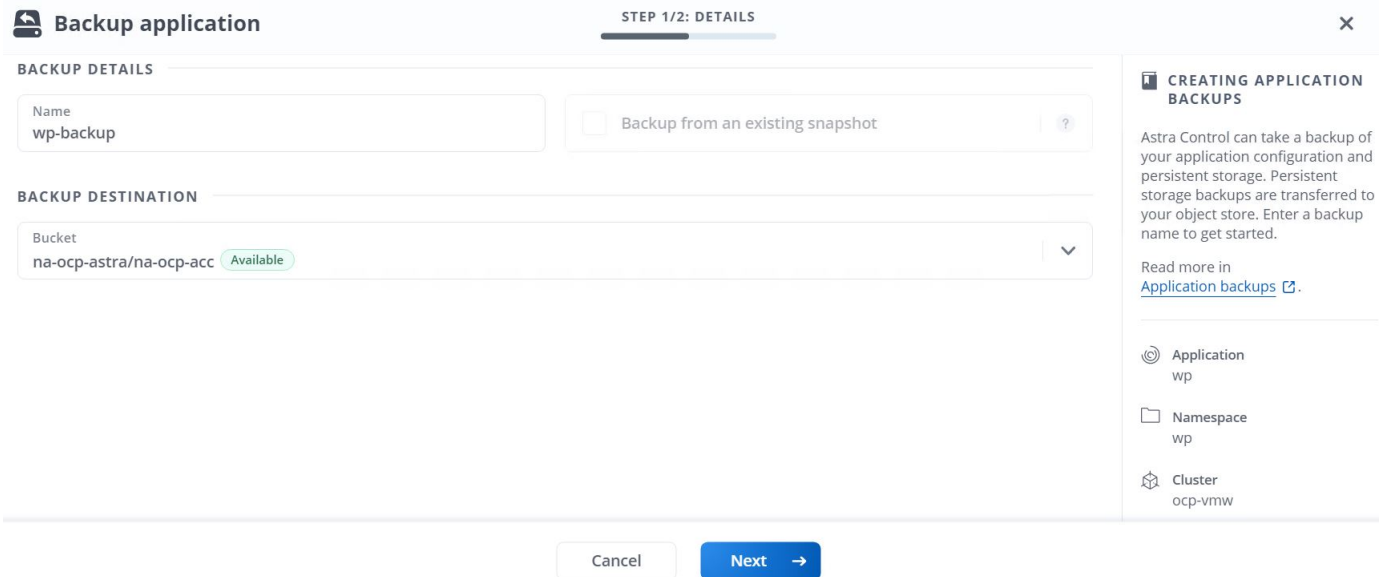
要在 Astra 控制中心备份和还原受管应用程序，必须先为支持的 ONTAP 系统配置超级用户设置。为此，请输入以下命令。

```
ONTAP::> export-policy rule modify -vserver ocp-trident -policyname
default -ruleindex 1 -superuser sys
ONTAP::> export-policy rule modify -policyname default -ruleindex 1 -anon
65534 -vserver ocp-trident
```

1. 要在 Astra 控制中心创建受管应用程序的备份，请导航到应用程序 > 受管选项卡，然后单击要备份的应用程序。单击应用程序名称旁边的下拉菜单，然后单击备份。



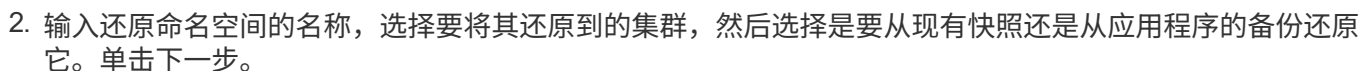
2. 输入备份详细信息，选择用于保存备份文件的对象存储分段，单击下一步，查看详细信息后，单击备份。根据应用程序和数据的大小，备份可能需要几分钟的时间，备份成功完成后，备份状态将变为可用。



还原应用程序

只需按一个按钮，即可将应用程序还原到同一集群中的原始命名空间或远程集群，以实现应用程序保护和灾难恢复。

1. 要还原应用程序，请导航到应用程序 > 受管选项卡，然后单击相关应用程序。单击应用程序名称旁边的下拉菜单，然后单击 Restore。



REVIEW RESTORE INFORMATION



All existing resources associated with this application will be deleted and replaced with the source backup "wp-backup" taken on 2022/02/28 18:54 UTC. Persistent volumes will be deleted and recreated. External resources with dependencies on this application may be impacted.

We recommend taking a snapshot or a backup of your application before proceeding.



BACKUP
wp-backup



ORIGINAL GROUP
wp



ORIGINAL CLUSTER
ocp-vmw



RESOURCE LABELS
ClusterRole
kubernetes.io/bootstrapping: rbac-defaults +1
ClusterRoleBinding



RESTORE
wp



DESTINATION GROUP
wp



DESTINATION CLUSTER
ocp-vmw



RESOURCE LABELS
ClusterRole
kubernetes.io/bootstrapping: rbac-defaults +1
ClusterRoleBinding

Are you sure you want to restore the application "wp"?

Type **restore** below to confirm.

Confirm to restore
restore

← Back

Restore ✓

- 当 Astra 控制中心在选定集群上还原应用程序时，新应用程序将进入还原状态。在 Astra 安装并检测到应用程序的所有资源后，该应用程序将进入可用状态。

Applications

Actions ▾							
+ Define							
Search							
110							
1-1 of 1 entries							
<input type="checkbox"/>	Name ↓	Ready	Protected	Cluster	Group	Discovered	Actions
<input type="checkbox"/>	wp	✓	i	ocp-vmw	wp	2022/02/28 18:34 UTC	Available ▼

克隆应用程序

您可以将应用程序克隆到发起集群或远程集群，以进行开发 / 测试或应用程序保护和灾难恢复。在同一个存储后端的同一集群中克隆应用程序时，会使用 NetApp FlexClone 技术，从而可以即时克隆 PVC 并节省存储空间。

- 要克隆应用程序，请导航到应用程序 > 受管选项卡，然后单击相关应用程序。单击应用程序名称旁边的下拉菜单，然后单击克隆。

The screenshot shows the 'wp' application status. The 'APPLICATION STATUS' is 'Healthy'. The 'APPLICATION PROTECTION STATUS' is 'Partially protected'. Below this, there are three sections: 'Images' (listing docker.io/bitnami/mariadb:10.5.13-debian-10-r58 and docker.io/bitnami/wordpress:5.9.0-debian-10-r1), 'Protection schedule' (Disabled), and 'Group' (wp). A dropdown menu is open, showing options: Running, Snapshot, Backup, Clone, Restore, and Unmanage. The 'Clone' option is highlighted.

2. 输入新命名空间的详细信息，选择要将其克隆到的集群，然后选择是要从现有快照，备份还是应用程序的当前状态克隆该命名空间。查看详细信息后，单击下一步并单击审阅窗格时克隆。

The screenshot shows the 'Clone application' dialog box. The 'CLONE DETAILS' section includes fields for 'Clone name' (wp-clone), 'Clone namespace' (wp-clone), and 'Destination cluster' (ocp-vmw). There is a checkbox for 'Clone from an existing snapshot or backup'. The 'CLONING APPLICATIONS' section on the right provides information about cloning applications and includes a link to 'Read more in Clone applications'. At the bottom, there are 'Cancel' and 'Next' buttons.

3. 当 Astra 控制中心在选定集群上创建应用程序时，新应用程序将进入 "正在发现" 状态。在 Astra 安装并检测到应用程序的所有资源后，该应用程序将进入可用状态。

The screenshot shows the 'Applications' table. The table has columns: Name, Ready, Protected, Cluster, Group, Discovered, and Actions. There are two entries: 'wp' and 'wp-clone'. Both are in the 'Available' state.

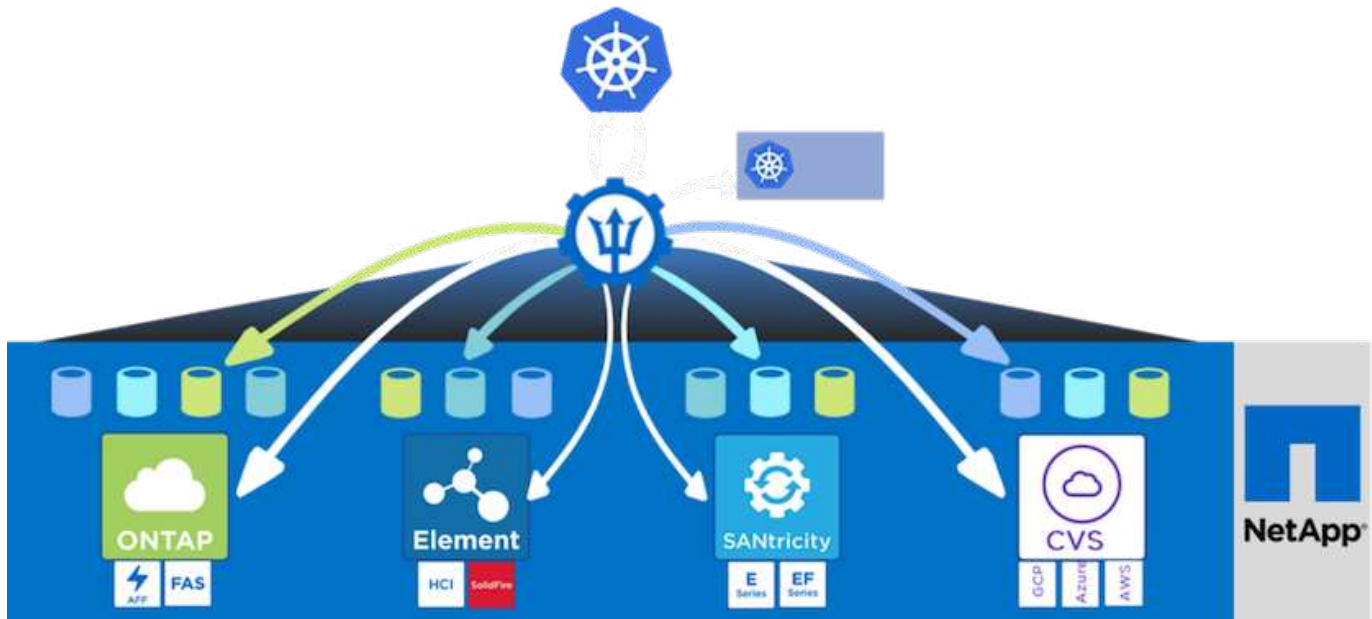
Name	Ready	Protected	Cluster	Group	Discovered	Actions
wp	✓	ⓘ	ocp-vmw	wp	2022/02/28 18:34 UTC	Available
wp-clone	✓	⚠	ocp-vmw	wp-clone	2022/02/28 19:21 UTC	Available

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-5z451	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.trident.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 `ServiceAccount`，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:             kubect1-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，客户端和服务端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端上支持的值手动同步，以确保 NFS 连接的最佳性能，而服务端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,ZGVmYXVsdHMgewogICAgICAgIHVzZXJfZnJpZW5kbHlfbmFtZXMgbm8KICAgICAgICBmaW5kX211bHRpcGF0aHMgbm8KfQoKYmxhY2tsaXN0X2V4Y2VwdGlvbnMgewogICAgICAgIHByb3BlcnR5ICIoU0NTSV9JREV0VF98SURfV1dOKSIKfQoKYmxhY2tsaXN0IHsKfQoK
          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@rhel17 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.tmpl ./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	STATE	VOLUMES	STORAGE DRIVER	UUID
solidfire_10.61.180.200	online	0	solidfire-san	b90783ee-e0c9-49af-8d26-3ea87ce2efdf

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 。 yml 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
```

版权信息

版权所有 © 2024 NetApp, Inc.。保留所有权利。中国印刷。未经版权所有者事先书面许可，本文档中受版权保护的任何部分不得以任何形式或通过任何手段（图片、电子或机械方式，包括影印、录音、录像或存储在电子检索系统中）进行复制。

从受版权保护的 NetApp 资料派生的软件受以下许可和免责声明的约束：

本软件由 NetApp 按“原样”提供，不含任何明示或暗示担保，包括但不限于适销性以及针对特定用途的适用性的隐含担保，特此声明不承担任何责任。在任何情况下，对于因使用本软件而以任何方式造成的任何直接性、间接性、偶然性、特殊性、惩罚性或后果性损失（包括但不限于购买替代商品或服务；使用、数据或利润方面的损失；或者业务中断），无论原因如何以及基于何种责任理论，无论出于合同、严格责任或侵权行为（包括疏忽或其他行为），NetApp 均不承担责任，即使已被告知存在上述损失的可能性。

NetApp 保留在不另行通知的情况下随时对本文档所述的任何产品进行更改的权利。除非 NetApp 以书面形式明确同意，否则 NetApp 不承担因使用本文档所述产品而产生的任何责任或义务。使用或购买本产品不表示获得 NetApp 的任何专利权、商标权或任何其他知识产权许可。

本手册中描述的产品可能受一项或多项美国专利、外国专利或正在申请的专利的保护。

有限权利说明：政府使用、复制或公开本文档受 DFARS 252.227-7013（2014 年 2 月）和 FAR 52.227-19（2007 年 12 月）中“技术数据权利 — 非商用”条款第 (b)(3) 条规定的限制条件的约束。

本文档中所含数据与商业产品和/或商业服务（定义见 FAR 2.101）相关，属于 NetApp, Inc. 的专有信息。根据本协议提供的所有 NetApp 技术数据和计算机软件具有商业性质，并完全由私人出资开发。美国政府对这些数据的使用权具有非排他性、全球性、受限且不可撤销的许可，该许可既不可转让，也不可再许可，但仅限在与交付数据所依据的美国政府合同有关且受合同支持的情况下使用。除本文档规定的情形外，未经 NetApp, Inc. 事先书面批准，不得使用、披露、复制、修改、操作或显示这些数据。美国政府对国防部的授权仅限于 DFARS 的第 252.227-7015(b)（2014 年 2 月）条款中明确的权利。

商标信息

NetApp、NetApp 标识和 <http://www.netapp.com/TM> 上所列的商标是 NetApp, Inc. 的商标。其他公司和产品名称可能是其各自所有者的商标。