

# **NetApp Storage Integrations Overview**

**NetApp Solutions** 

NetApp August 13, 2023

This PDF was generated from https://docs.netapp.com/us-en/netapp-solutions/containers/rh-os-n\_astra\_register.html on August 13, 2023. Always check docs.netapp.com for the latest.

# **Table of Contents**

| NetApp Storage Integration Overview  | <br> |  | <br> | <br> | <br>. 1 |
|--------------------------------------|------|------|------|------|------|------|------|------|------|--|------|------|---------|
| NetApp Astra Control Center overview | <br> |  | <br> | <br> | <br>. 2 |
| Astra Trident Overview               | <br> |  | <br> | <br> | <br>30  |

## **NetApp Storage Integration Overview**

NetApp provides a number of products to help you with orchestrating and managing persistent data in container based environments, such as Red Hat OpenShift.



NetApp Astra Control offers a rich set of storage and application-aware data management services for stateful Kubernetes workloads, powered by NetApp data protection technology. The Astra Control Service is available to support stateful workloads in cloud-native Kubernetes deployments. The Astra Control Center is available to support stateful workloads in on-premises deployments, like Red Hat OpenShift. For more information visit the NetApp Astra Control website here.

NetApp Astra Trident is an open-source and fully-supported storage orchestrator for containers and Kubernetes distributions, including Red Hat OpenShift. For more information, visit the Astra Trident website here.

The following pages have additional information about the NetApp products that have been validated for application and persistent storage management in the Red Hat OpenShift with NetApp solution:

- NetApp Astra Control Center
- NetApp Astra Trident

Next: NetApp Astra Control Center Overview

## NetApp Astra Control Center overview

NetApp Astra Control Center offers a rich set of storage and application-aware data management services for stateful Kubernetes workloads deployed in an on-premises environment and powered by NetApp data protection technology.



NetApp Astra Control Center can be installed on a Red Hat OpenShift cluster that has the Astra Trident storage orchestrator deployed and configured with storage classes and storage backends to NetApp ONTAP storage systems.

For the installation and configuration of Astra Trident to support Astra Control Center, see this document here.

In a cloud-connected environment, Astra Control Center uses Cloud Insights to provide advanced monitoring and telemetry. In the absence of a Cloud Insights connection, limited monitoring and telemetry (7-days worth of metrics) is available and exported to Kubernetes native monitoring tools (Prometheus and Grafana) through open metrics endpoints.

Astra Control Center is fully integrated into the NetApp AutoSupport and Active IQ ecosystem to provide support for users, provide assistance with troubleshooting, and display usage statistics.

In addition to the paid version of Astra Control Center, a 90-day evaluation license is available. The evaluation

version is supported through the email and community (Slack channel). Customers have access to these and other knowledge-base articles and the documentation available from the in-product support dashboard.

To get started with NetApp Astra Control Center, visit the Astra website.

## **Astra Control Center installation prerequisites**

- 1. One or more Red Hat OpenShift clusters. Versions 4.6 EUS and 4.7 are currently supported.
- 2. Astra Trident must already be installed and configured on each Red Hat OpenShift cluster.
- 3. One or more NetApp ONTAP storage systems running ONTAP 9.5 or greater.



It's best practice for each OpenShift install at a site to have a dedicated SVM for persistent storage. Multi-site deployments require additional storage systems.

- 4. A Trident storage backend must be configured on each OpenShift cluster with an SVM backed by an ONTAP cluster.
- 5. A default StorageClass configured on each OpenShift cluster with Astra Trident as the storage provisioner.
- 6. A load balancer must be installed and configured on each OpenShift cluster for load balancing and exposing OpenShift Services.



See the link here for information about load balancers that have been validated for this purpose.

- 7. A private image registry must be configured to host the NetApp Astra Control Center images.

See the link here to install and configure an OpenShift private registry for this purpose.

- 8. You must have Cluster Admin access to the Red Hat OpenShift cluster.
- 9. You must have Admin access to NetApp ONTAP clusters.
- 10. An admin workstation with docker or podman, tridentctl, and oc or kubectl tools installed and added to your \$PATH.



Docker installations must have docker version greater than 20.10 and Podman installations must have podman version greater than 3.0.

## **Install Astra Control Center**

## **Using OperatorHub**

1. Log into the NetApp Support Site and download the latest version of NetApp Astra Control Center. To do so requires a license attached to your NetApp account. After you download the tarball, transfer it to the admin workstation.



To get started with a trial license for Astra Control, visit the Astra registration site.

2. Unpack the tar ball and change the working directory to the resulting folder.

```
[netapp-user@rhe17 ~]$ tar -vxzf astra-control-center-
21.12.60.tar.gz
[netapp-user@rhe17 ~]$ cd astra-control-center-21.12.60
```

3. Before starting the installation, push the Astra Control Center images to an image registry. You can choose to do this with either Docker or Podman, instructions for both are provided in this step.

#### **Podman**

a. Export the registry FQDN with the organization/namespace/project name as a environment variable 'registry'.

```
[netapp-user@rhe17 ~]$ export REGISTRY=astra-
registry.apps.ocp-vmw.cie.netapp.com/netapp-astra
```

b. Log into the registry.

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



If you are using kubeadmin user to log into the private registry, then use token instead of password - podman login -u ocp-user -p token --tls-verify=false astra-registry.apps.ocp-vmw.cie.netapp.com.



Alternatively, you can create a service account, assign registry-editor and/or registry-viewer role (based on whether you require push/pull access) and log into the registry using service account's token.

c. Create a shell script file and paste the following content in it.

```
[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
```



If you are using untrusted certificates for your registry, edit the shell script and use --tls-verify=false for the podman push command podman push \$REGISTRY/\$ (echo \$astraImage | sed 's/[\/]\+\///') --tls-verify=false.

d. Make the file executable.

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

e. Execute the shell script.

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

#### Docker

a. Export the registry FQDN with the organization/namespace/project name as a environment variable 'registry'.

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

b. Log into the registry.

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



If you are using kubeadmin user to log into the private registry, then use token instead of password - docker login -u ocp-user -p token astra-registry.apps.ocp-vmw.cie.netapp.com.



Alternatively, you can create a service account, assign registry-editor and/or registry-viewer role (based on whether you require push/pull access) and log into the registry using service account's token.

c. Create a shell script file and paste the following content in it.

```
[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. Make the file executable.

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

e. Execute the shell script.

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

4. When using private image registries that are not publicly trusted, upload the image registry TLS certificates to the OpenShift nodes. To do so, create a configmap in the openshift-config namespace using the TLS certificates and patch it to the cluster image config to make the certificate trusted.

```
[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
```



If you are using an OpenShift internal registry with default TLS certificates from the ingress operator with a route, you still need to follow the previous step to patch the certificates to the route hostname. To extract the certificates from ingress operator, you can use the command oc extract secret/router-ca --keys=tls.crt -n openshift-ingress-operator.

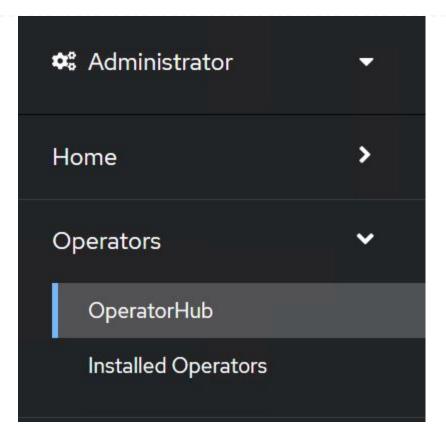
5. Create a namespace netapp-acc-operator for Astra Control Center.

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

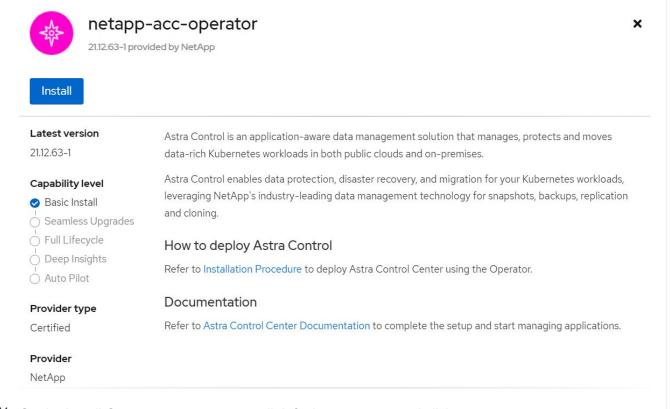
6. Create a secret with credentials to log into the image registry in netapp-acc-operator namespace.

```
[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
```

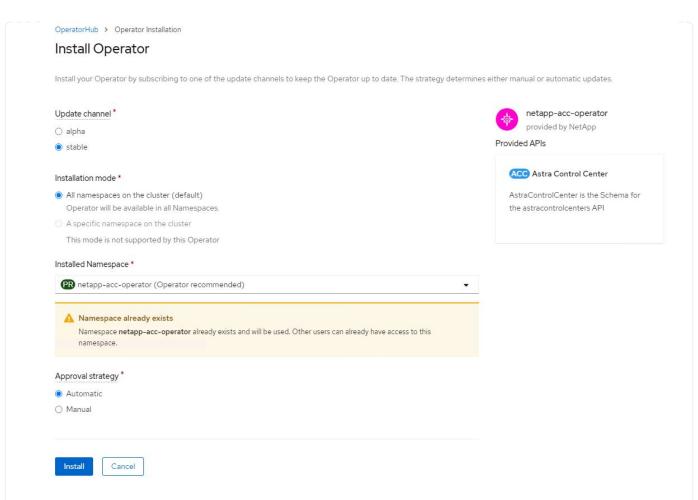
- 7. Log into the Red Hat OpenShift GUI console with cluster-admin access.
- 8. Select Administrator from the Perspective drop down.
- 9. Navigate to Operators > OperatorHub and search for Astra.



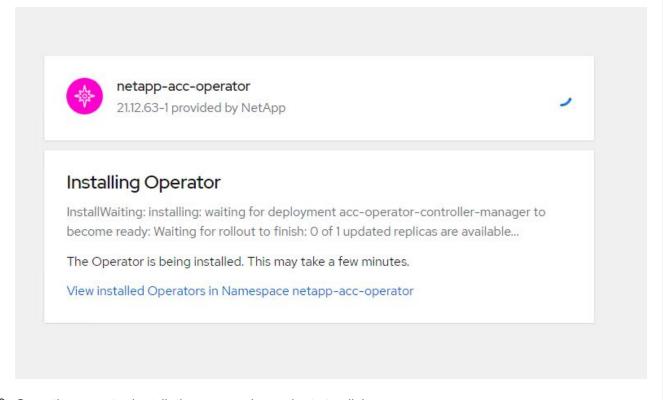
10. Select netapp-acc-operator tile and click Install.



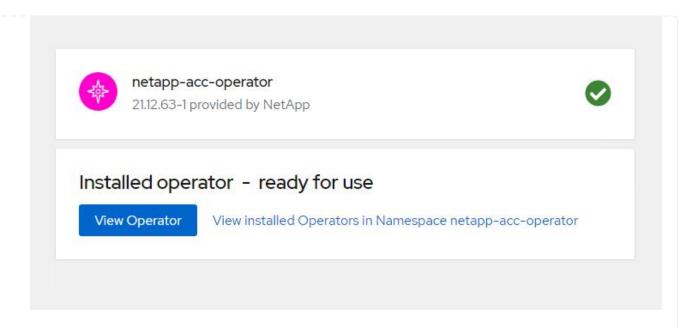
11. On the Install Operator screen, accept all default parameters and click Install.



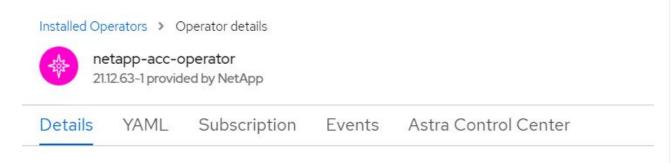
12. Wait for the operator installation to complete.



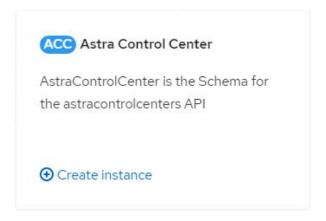
13. Once the operator installation succeeds, navigate to click on View Operator.



14. Then click on Create Instance in Astra Control Center tile in the operator.



## Provided APIs



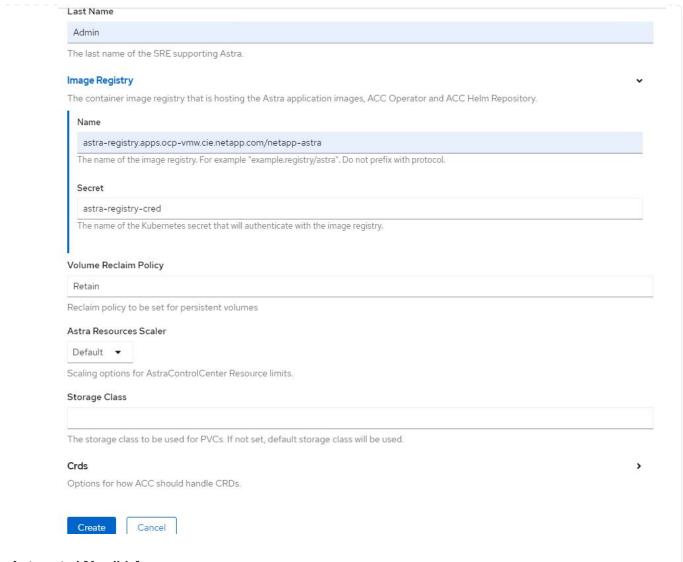
- 15. Fill the Create AstraControlCenter form fields and click Create.
  - a. Optionally edit the Astra Control Center instance name.
  - b. Optionally enable or disable Auto Support. Retaining Auto Support functionality is recommended.
  - c. Enter the FQDN for Astra Control Center.
  - d. Enter the Astra Control Center version; the latest is displayed by default.
  - e. Enter an account name for Astra Control Center and admin details like first name, last name and

email address.

- f. Enter the volume reclaim policy, default is Retain.
- g. In Image Registry, enter the FQDN for your registry along with the organization name as it was given while pushing the images to the registry (in this example, astra-registry.apps.ocp-vmw.cie.netapp.com/netapp-astra)
- h. If you use a registry that requires authentication, enter the secret name in Image Registry section.
- i. Configure scaling options for Astra Control Center resource limits.
- j. Enter the storage class name if you want to place PVCs on a non-default storage class.
- k. Define CRD handling preferences.

The first name of the SRE supporting Astra.

Name *	
Name *	
astra	
Labels	
app=frontend	
Account Name *	
HCG Solutions Engineering	
Astra Control Center account n	ame
Astra Address *	
astra-control-center.cie.netap	p.com
Astra Control Center. Example - Center	a will be found in the data center. This IP address and/or DNS A record must be created prior to provisioning - "astra.example.com" The A record and its IP address must be allocated prior to provisioning Astra Control
Astra Version *	
21.12.60	
	to deploy. You are provided a Helm repository with a corresponding version. Example - 1.5.2, 1.4.2-patch
Email *	
solutions_tme@netapp.com	
EmailAddress will be notified by	Astra as events warrant.
Auto Support *	•
	ess to participate in NetApp's proactive support application, NetApp Active IQ. The default election is true and ent to NetApp. An empty or blank election is the same as a default election. Air gapped installations should ent
Talse.	
First Name	



#### Automated [Ansible]

- 1. To use Ansible playbooks to deploy Astra Control Center, you need an Ubuntu/RHEL machine with Ansible installed. Follow the procedures here for Ubuntu and RHEL.
- 2. Clone the GitHub repository that hosts the Ansible content.

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

3. Log into the NetApp Support site and download the latest version of NetApp Astra Control Center. To do so requires a license attached to your NetApp account. After you download the tarball, transfer it to the workstation.



To get started with a trial license for Astra Control, visit the Astra registration site.

- 4. Create or obtain the kubeconfig file with admin access to the OpenShift cluster on which Astra Control Center is to be installed.
- 5. Change the directory to the na astra control suite.

cd na astra control suite

6. Edit the vars/vars.yml file, and fill in the variables with the required information.

```
#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
#"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. Run the playbook to deploy Astra Control Center. The playbook requires root privileges for certain configurations.

If the user running the playbook is root or has passwordless sudo configured, then run the following command to run the playbook.

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

If the user has password-based sudo access configured, run the following command to run the playbook, and then enter the sudo password.

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

#### **Post Install Steps**

1. It might take several minutes for the installation to complete. Verify that all the pods and services in the netapp-astra-cc namespace are up and running.

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

2. Check the acc-operator-controller-manager logs to ensure that the installation is completed.

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



The following message indicates the successful installation of Astra Control Center.

```
{"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]"}
```

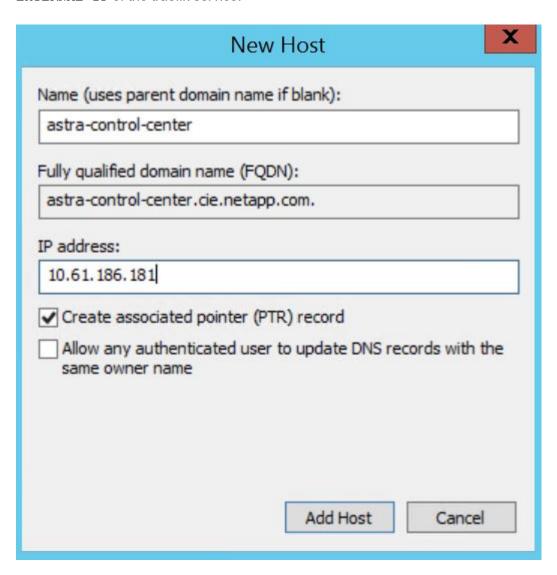
3. The username for logging into Astra Control Center is the email address of the administrator provided in the CRD file and the password is a string ACC- appended to the Astra Control Center UUID. Run the following command:



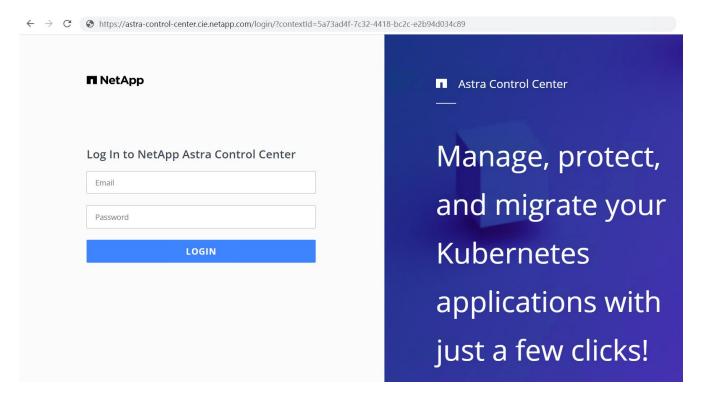
In this example, the password is ACC-345c55a5-bf2e-21f0-84b8-b6f2bce5e95f.

4. Get the traefik service load balancer IP.

5. Add an entry in the DNS server pointing the FQDN provided in the Astra Control Center CRD file to the



6. Log into the Astra Control Center GUI by browsing its FQDN.



When you log into Astra Control Center GUI for the first time using the admin email address provided in CRD, you need to change the password.



8. If you wish to add a user to Astra Control Center, navigate to Account > Users, click Add, enter the details of the user, and click Add.



9. Astra Control Center requires a license for all of it's functionalities to work. To add a license, navigate to Account > License, click Add License, and upload the license file.





If you encounter issues with the install or configuration of NetApp Astra Control Center, the knowledge base of known issues is available here.

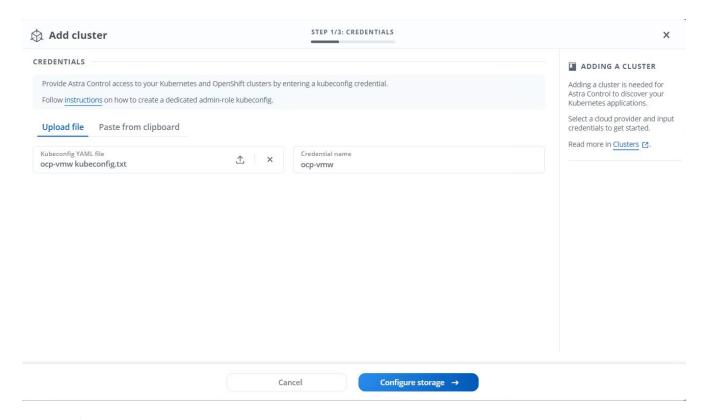
Next: Register your Red Hat OpenShift Clusters: Red Hat OpenShift with NetApp.

## Register your Red Hat OpenShift Clusters with the Astra Control Center

To enable the Astra Control Center to manage your workloads, you must first register your Red Hat OpenShift cluster.

#### Register Red Hat OpenShift clusters

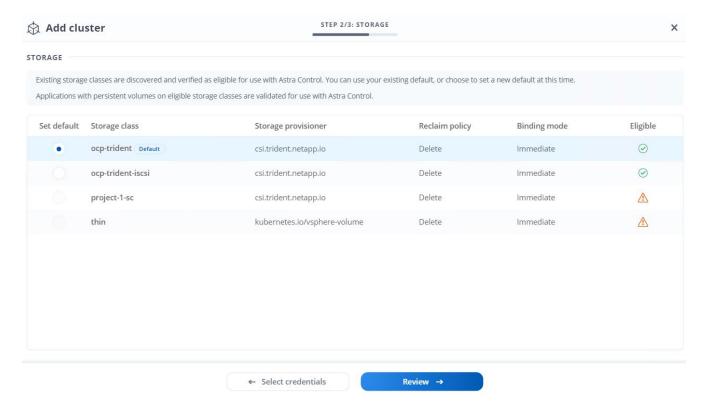
1. The first step is to add the OpenShift clusters to the Astra Control Center and manage them. Go to Clusters and click Add a Cluster, upload the kubeconfig file for the OpenShift cluster, and click Select Storage.





The kubeconfig file can be generated to authenticate with a username and password or a token. Tokens expire after a limited amount of time and might leave the registered cluster unreachable. NetApp recommends using a kubeconfig file with a username and password to register your OpenShift clusters to Astra Control Center.

Astra Control Center detects the eligible storage classes. Now select the way that storageclass provisions volumes using Trident backed by an SVM on NetApp ONTAP and click Review. In the next pane, verify the details and click Add Cluster.



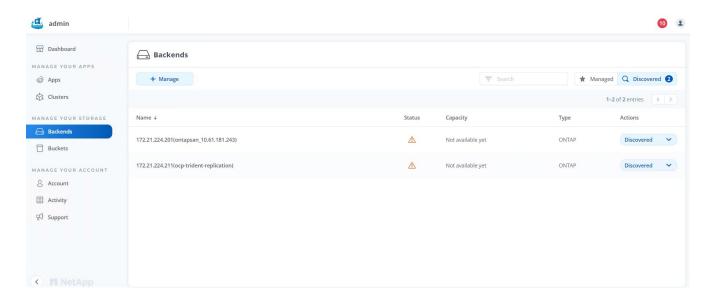
3. Register both OpenShift clusters as described in step 1. When added, the clusters move to the Discovering status while Astra Control Center inspects them and installs the necessary agents. Cluster status changes to Running after they are successfully registered.





All Red Hat OpenShift clusters to be managed by Astra Control Center should have access to the image registry that was used for its installation as the agents installed on the managed clusters pull the images from that registry.

4. Import ONTAP clusters as storage resources to be managed as backends by Astra Control Center. When OpenShift clusters are added to Astra and a storageclass is configured, it automatically discovers and inspects the ONTAP cluster backing the storageclass but does not import it into the Astra Control Center to be managed.



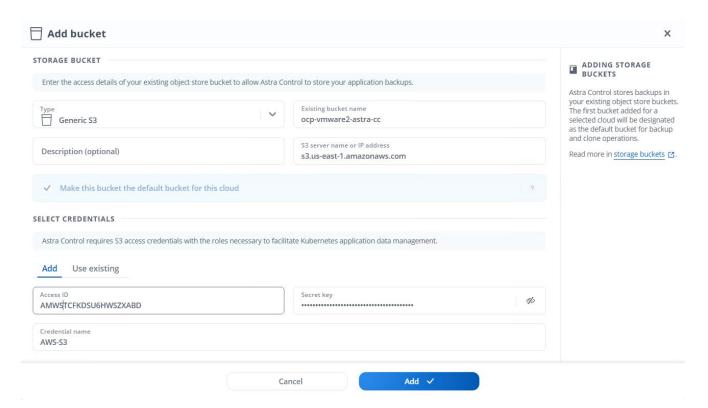
 To import the ONTAP clusters, go to Backends, click the dropdown, and select Manage next to the ONTAP cluster to be managed. Enter the ONTAP cluster credentials, click Review Information, and then click Import Storage Backend.



After the backends are added, the status changes to Available. These backends now have the information about the persistent volumes in the OpenShift cluster and the corresponding volumes on the ONTAP system.



7. For backup and restore across OpenShift clusters using Astra Control Center, you must provision an object storage bucket that supports the S3 protocol. Currently supported options are ONTAP S3, StorageGRID, and AWS S3. For the purpose of this installation, we are going to configure an AWS S3 bucket. Go to Buckets, click Add bucket, and select Generic S3. Enter the details about the S3 bucket and credentials to access it, click the checkbox "Make this bucket the default bucket for the cloud," and then click Add.



Next: Choose the Applications To Protect.

## Choose the applications to protect

After you have registered your Red Hat OpenShift clusters, you can discover the applications that are deployed and manage them via the Astra Control Center.

## Manage applications

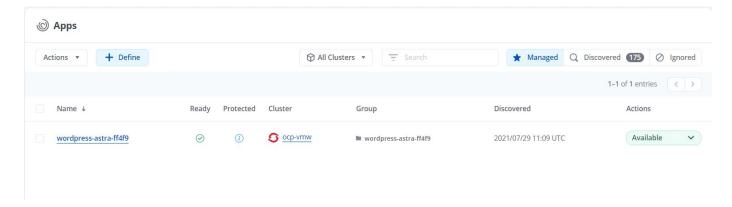
1. After the OpenShift clusters and ONTAP backends are registered with the Astra Control Center, the control center automatically starts discovering the applications in all the namespaces that are using the storageclass configured with the specified ONTAP backend.



2. Navigate to Apps > Discovered and click the dropdown menu next to the application you would like to manage using Astra. Then click Manage.



1. The application enters the Available state and can be viewed under the Managed tab in the Apps section.



Next: Protect Your applications.

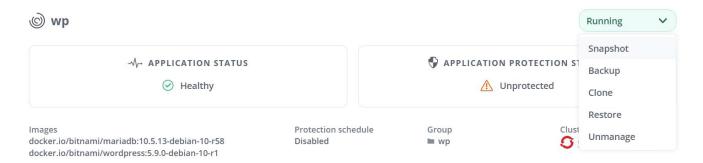
## **Protect your applications**

After application workloads are managed by Astra Control Center, you can configure the protection settings for those workloads.

#### Creating an application snapshot

A snapshot of an application creates an ONTAP Snapshot copy that can be used to restore or clone the application to a specific point in time based on that Snapshot copy.

1. To take a snapshot of the application, navigate to the Apps > Managed tab and click the application you would like to make a Snapshot copy of. Click the dropdown menu next to the application name and click Snapshot.



2. Enter the snapshot details, click Next, and then click Snapshot. It takes about a minute to create the snapshot, and the status becomes Available after the snapshot is successfully created.



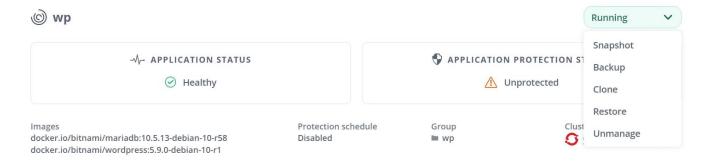
#### Creating an application backup

A backup of an application captures the active state of the application and the configuration of it's resources, coverts them into files, and stores them in a remote object storage bucket.

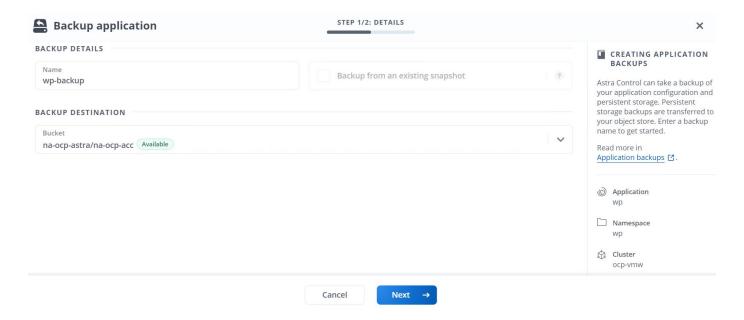
For the backup and restore of managed applications in the Astra Control Center, you must configure superuser settings for the backing ONTAP systems as a prerequisite. To do so, enter the following commands.

```
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. To create a backup of the managed application in the Astra Control Center, navigate to the Apps > Managed tab and click the application that you want to take a backup of. Click the dropdown menu next to the application name and click Backup.



Enter the backup details, select the object storage bucket to hold the backup files, click Next, and, after reviewing the details, click Backup. Depending on the size of the application and data, the backup can take several minutes, and the status of the backup becomes Available after the backup is completed successfully.



## Restoring an application

At the push of a button, you can restore an application to the originating namespace in the same cluster or to a remote cluster for application protection and disaster recovery purposes.

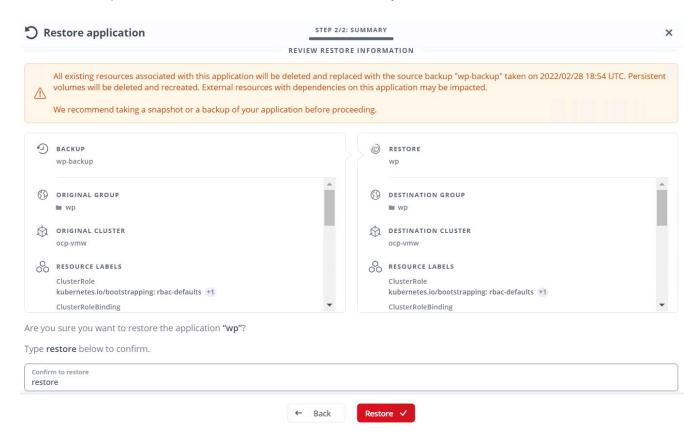
1. To restore an application, navigate to Apps > Managed tab and click the app in question. Click the dropdown menu next to the application name and click Restore.



2. Enter the name of the restore namespace, select the cluster you want to restore it to, and choose if you want to restore it from an existing snapshot or from a backup of the application. Click Next.

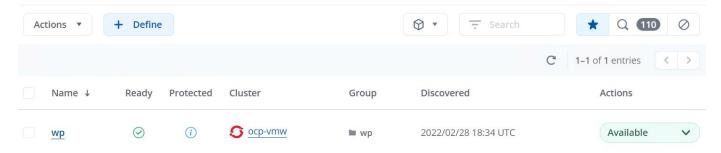


3. On the review pane, enter restore and click Restore after you have reviewed the details.



4. The new application goes to the Restoring state while Astra Control Center restores the application on the selected cluster. After all the resources of the application are installed and detected by Astra, the application goes to the Available state.

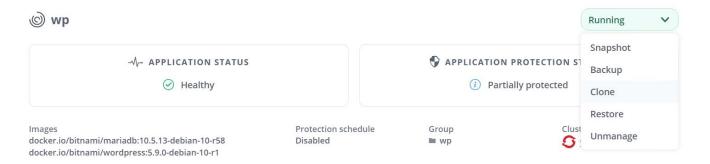
## Applications



#### Cloning an application

You can clone an application to the originating cluster or to a remote cluster for dev/test or application protection and disaster recovery purposes. Cloning an application within the same cluster on the same storage backend uses NetApp FlexClone technology, which clones the PVCs instantly and saves storage space.

1. To clone an application, navigate to the Apps > Managed tab and click the app in question. Click the dropdown menu next to the application name and click Clone.

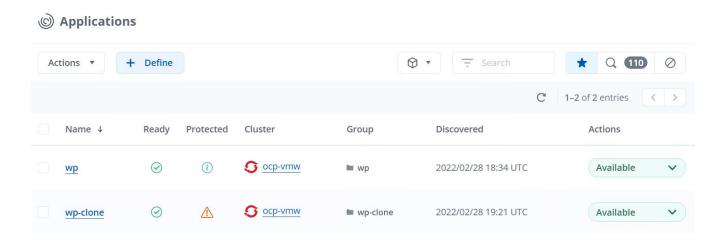


2. Enter the details of the new namespace, select the cluster you want to clone it to, and choose if you want to clone it from an existing snapshot or a backup or the current state of the application. Then click Next and click Clone on review pane once you have reviewed the details.



3. The new application goes to the Discovering state while Astra Control Center creates the application on the

selected cluster. After all the resources of the application are installed and detected by Astra, the application goes to the Available state.



Next: Solution Validation/Use Cases.

## **Astra Trident Overview**

Astra Trident is an open-source and fully supported storage orchestrator for containers and Kubernetes distributions, including Red Hat OpenShift. Trident works with the entire NetApp storage portfolio, including the NetApp ONTAP and Element storage systems, and it also supports NFS and iSCSI connections. Trident accelerates the DevOps workflow by allowing end users to provision and manage storage from their NetApp storage systems without requiring intervention from a storage administrator.

An administrator can configure a number of storage backends based on project needs and storage system models that enable advanced storage features, including compression, specific disk types, or QoS levels that guarantee a certain level of performance. After they are defined, these backends can be used by developers in their projects to create persistent volume claims (PVCs) and to attach persistent storage to their containers on demand.



Astra Trident has a rapid development cycle, and just like Kubernetes, is released four times a year.

The latest version of Astra Trident is 22.01 released in January 2022. A support matrix for what version of Trident has been tested with which Kubernetes distribution can be found here.

Starting with the 20.04 release, Trident setup is performed by the Trident operator. The operator makes large scale deployments easier and provides additional support including self healing for pods that are deployed as a part of the Trident install.

With the 21.01 release, a Helm chart was made available to ease the installation of the Trident Operator.

## **Download Astra Trident**

To install Trident on the deployed user cluster and provision a persistent volume, complete the following steps:

1. Download the installation archive to the admin workstation and extract the contents. The current version of Trident is 22.01, which can be downloaded here.

```
[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.qz
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
```

2. Extract the Trident install from the downloaded bundle.

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

## **Install the Trident Operator with Helm**

1. First set the location of the user cluster's kubeconfig file as an environment variable so that you don't have to reference it, because Trident has no option to pass this file.

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

2. Run the Helm command to install the Trident operator from the tarball in the helm directory while creating the trident namespace in your user cluster.

```
[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. You can verify that Trident is successfully installed by checking the pods that are running in the namespace or by using the tridentctl binary to check the installed version.

```
[netapp-user@rhel7 trident-installer] $ oc get pods -n trident
NAME
                             READY
                                    STATUS
                                            RESTARTS
                                                     AGE
trident-csi-5z451
                             1/2
                                    Running
                                                     30s
                             6/6
trident-csi-696b685cf8-htdb2
                                    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
+----+
```



In some cases, customer environments might require the customization of the Trident deployment. In these cases, it is also possible to manually install the Trident operator and update the included manifests to customize the deployment.

## **Manually install the Trident Operator**

1. First, set the location of the user cluster's kubeconfig file as an environment variable so that you don't have to reference it, because Trident has no option to pass this file.

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

2. The trident-installer directory contains manifests for defining all the required resources. Using the appropriate manifests, create the TridentOrchestrator custom resource definition.

```
[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. If one does not exist, create a Trident namespace in your cluster using the provided manifest.

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

4. Create the resources required for the Trident operator deployment, such as a ServiceAccount for the operator, a ClusterRole and ClusterRoleBinding to the ServiceAccount, a dedicated

PodSecurityPolicy, or the operator itself.

```
[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. You can check the status of the operator after it's deployed with the following commands:

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

6. With the operator deployed, we can now use it to install Trident. This requires creating a 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:
  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
                    2021-05-07T17:00:28Z
    Time:
 Resource Version: 931421
  Self Link:
/apis/trident.netapp.io/v1/tridentorchestrators/trident
                     8a26a7a6-dde8-4d55-9b66-a7126754d81f
 UID:
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
                        Trident installed
 Message:
                        trident
 Namespace:
 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. You can verify that Trident is successfully installed by checking the pods that are running in the namespace or by using the tridentctl binary to check the installed version.

```
[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
                             2/2
trident-csi-m4szj
                                   Running 0
                                                    82s
trident-csi-sb9k9
                                                    82s
                             2/2
                                   Running 0
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
1 22.01.0
+----+
```

# Prepare worker nodes for storage

#### **NFS**

Most Kubernetes distributions come with the packages and utilities to mount NFS backends installed by default, including Red Hat OpenShift.

However, for NFSv3, there is no mechanism to negotiate concurrency between the client and the server. Hence the maximum number of client-side sunrpc slot table entries must be manually synced with supported value on the server to ensure the best performance for the NFS connection without the server having to

decrease the window size of the connection.

For ONTAP, the supported maximum number of sunrpc slot table entries is 128 i.e. ONTAP can serve 128 concurrent NFS requests at a time. However, by default, Red Hat CoreOS/Red Hat Enterprise Linux has maximum of 65,536 sunrpc slot table entries per connection. We need to set this value to 128 and this can be done using Machine Config Operator (MCO) in OpenShift.

To modify the maximum sunrpc slot table entries in OpenShift worker nodes, complete the following steps:

1. Log into the OCP web console and navigate to Compute > Machine Configs. Click Create Machine Config. Copy and paste the YAML file and click Create.

```
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, b3B0aW9ucyBzdW5ycGMqdGNwX21heF9zbG90X3RhYmx1X2VudHJpZXM9MTI4Cq=
          filesystem: root
          mode: 420
          path: /etc/modprobe.d/sunrpc.conf
```

2. After the MCO is created, the configuration needs to be applied on all worker nodes and rebooted one by one. The whole process takes approximately 20 to 30 minutes. Verify whether the machine config is applied by using oc get mcp and make sure that the machine config pool for workers is updated.

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

#### iSCSI

To prepare worker nodes to allow for the mapping of block storage volumes through the iSCSI protocol, you must install the necessary packages to support that functionality.

In Red Hat OpenShift, this is handled by applying an MCO (Machine Config Operator) to your cluster after it is deployed.

To configure the worker nodes to run iSCSI services, complete the following steps:

1. Log into the OCP web console and navigate to Compute > Machine Configs. Click Create Machine Config. Copy and paste the YAML file and click Create.

When not using multipathing:

```
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: ""
```

When using multipathing:

```
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
qIHByb3BlcnR5ICIoU0NTSV9JREVOVF98SURfV1dOKSIKfQoKYmxhY2tsaXN0IHsKfQoK
          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. After the configuration is created, it takes approximately 20 to 30 minutes to apply the configuration to the worker nodes and reload them. Verify whether the machine config is applied by using oc get mcp and make sure that the machine config pool for workers is updated. You can also log into the worker nodes to confirm that the isosid service is running (and the multipathd service is running if using multipathing).

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



It is also possible to confirm that the MachineConfig has been successfully applied and services have been started as expected by running the oc debug command with the appropriate flags.

### Create storage-system backends

After completing the Astra Trident Operator install, you must configure the backend for the specific NetApp storage platform you are using. Follow the links below in order to continue the setup and configuration of Astra Trident.

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

Next: Solution Validation/Use Cases: Red Hat OpenShift with NetApp.

## **NetApp ONTAP NFS configuration**

To enable Trident integration with the NetApp ONTAP storage system, you must create a backend that enables communication with the storage system.

1. There are sample backend files available in the downloaded installation archive in the sample-input folder hierarchy. For NetApp ONTAP systems serving NFS, copy the backend-ontap-nas.json file to your working directory and edit the file.

```
[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. Edit the backendName, managementLIF, dataLIF, svm, username, and password values in this file.

```
"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"
}
```



It is a best practice to define the custom backendName value as a combination of the storageDriverName and the dataLIF that is serving NFS for easy identification.

3. With this backend file in place, run the following command to create your first backend.

4. With the backend created, you must next create a storage class. Just as with the backend, there is a sample storage class file that can be edited for the environment available in the sample-inputs folder. Copy it to the working directory and make necessary edits to reflect the backend created.

```
[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. The only edit that must be made to this file is to define the backendType value to the name of the storage driver from the newly created backend. Also note the name-field value, which must be referenced in a later step.

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



There is an optional field called fsType that is defined in this file. This line can be deleted in NFS backends.

6. Run the oc command to create the storage class.

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

7. With the storage class created, you must then create the first persistent volume claim (PVC). There is a sample pvc-basic.yaml file that can be used to perform this action located in sample-inputs as well.

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

8. The only edit that must be made to this file is ensuring that the storageClassName field matches the one just created. The PVC definition can be further customized as required by the workload to be provisioned.

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

9. Create the PVC by issuing the oc command. Creation can take some time depending on the size of the backing volume being created, so you can watch the process as it completes.

```
[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
```

Next: Solution validation/use cases.

# **NetApp ONTAP iSCSI configuration**

To enable Trident integration with the NetApp ONTAP storage system, you must create a backend that enables communication with the storage system.

1. There are sample backend files available in the downloaded installation archive in the sample-input folder hierarchy. For NetApp ONTAP systems serving iSCSI, copy the backend-ontap-san.json file to your working directory and edit the file.

```
[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. Edit the managementLIF, dataLIF, svm, username, and password values in this file.

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

3. With this backend file in place, run the following command to create your first backend.

4. With the backend created, you must next create a storage class. Just as with the backend, there is a sample storage class file that can be edited for the environment available in the sample-inputs folder. Copy it to the working directory and make necessary edits to reflect the backend created.

```
[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. The only edit that must be made to this file is to define the backendType value to the name of the storage driver from the newly created backend. Also note the name-field value, which must be referenced in a later step.

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



There is an optional field called fsType that is defined in this file. In iSCSI backends, this value can be set to a specific Linux filesystem type (XFS, ext4, etc) or can be deleted to allow OpenShift to decide what filesystem to use.

6. Run the oc command to create the storage class.

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

7. With the storage class created, you must then create the first persistent volume claim (PVC). There is a sample pvc-basic.yaml file that can be used to perform this action located in sample-inputs as well.

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

8. The only edit that must be made to this file is ensuring that the storageClassName field matches the one just created. The PVC definition can be further customized as required by the workload to be provisioned.

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

9. Create the PVC by issuing the oc command. Creation can take some time depending on the size of the backing volume being created, so you can watch the process as it completes.

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

Next: Solution validation/use cases.

### **NetApp Element iSCSI configuration**

To enable Trident integration with the NetApp Element storage system, you must create a backend that enables communication with the storage system using the iSCSI protocol.

1. There are sample backend files available in the downloaded installation archive in the sample-input folder hierarchy. For NetApp Element systems serving iSCSI, copy the backend-solidfire.json file to your working directory and edit the file.

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

- a. Edit the user, password, and MVIP value on the EndPoint line.
- b. Edit the SVIP value.

2. With this back-end file in place, run the following command to create your first backend.

3. With the backend created, you must next create a storage class. Just as with the backend, there is a sample storage class file that can be edited for the environment available in the sample-inputs folder. Copy it to the working directory and make necessary edits to reflect the backend created.

```
[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. The only edit that must be made to this file is to define the backendType value to the name of the storage driver from the newly created backend. Also note the name-field value, which must be referenced in a later step.

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



There is an optional field called fsType that is defined in this file. In iSCSI backends, this value can be set to a specific Linux filesystem type (XFS, ext4, and so on), or it can be deleted to allow OpenShift to decide what filesystem to use.

5. Run the oc command to create the storage class.

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

6. With the storage class created, you must then create the first persistent volume claim (PVC). There is a sample pvc-basic.yaml file that can be used to perform this action located in sample-inputs as well.

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

7. The only edit that must be made to this file is ensuring that the storageClassName field matches the one just created. The PVC definition can be further customized as required by the workload to be provisioned.

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

8. Create the PVC by issuing the oc command. Creation can take some time depending on the size of the backing volume being created, so you can watch the process as it completes.

```
[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
       Bound
basic
                pvc-3445b5cc-df24-453d-a1e6-b484e874349d
                                                           1Gi
              basic-csi
RWO
                             5s
```

Next: Solution validation/use cases.

#### Copyright information

Copyright © 2023 NetApp, Inc. All Rights Reserved. Printed in the U.S. No part of this document covered by copyright may be reproduced in any form or by any means—graphic, electronic, or mechanical, including photocopying, recording, taping, or storage in an electronic retrieval system—without prior written permission of the copyright owner.

Software derived from copyrighted NetApp material is subject to the following license and disclaimer:

THIS SOFTWARE IS PROVIDED BY NETAPP "AS IS" AND WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, WHICH ARE HEREBY DISCLAIMED. IN NO EVENT SHALL NETAPP BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

NetApp reserves the right to change any products described herein at any time, and without notice. NetApp assumes no responsibility or liability arising from the use of products described herein, except as expressly agreed to in writing by NetApp. The use or purchase of this product does not convey a license under any patent rights, trademark rights, or any other intellectual property rights of NetApp.

The product described in this manual may be protected by one or more U.S. patents, foreign patents, or pending applications.

LIMITED RIGHTS LEGEND: Use, duplication, or disclosure by the government is subject to restrictions as set forth in subparagraph (b)(3) of the Rights in Technical Data -Noncommercial Items at DFARS 252.227-7013 (FEB 2014) and FAR 52.227-19 (DEC 2007).

Data contained herein pertains to a commercial product and/or commercial service (as defined in FAR 2.101) and is proprietary to NetApp, Inc. All NetApp technical data and computer software provided under this Agreement is commercial in nature and developed solely at private expense. The U.S. Government has a non-exclusive, non-transferrable, nonsublicensable, worldwide, limited irrevocable license to use the Data only in connection with and in support of the U.S. Government contract under which the Data was delivered. Except as provided herein, the Data may not be used, disclosed, reproduced, modified, performed, or displayed without the prior written approval of NetApp, Inc. United States Government license rights for the Department of Defense are limited to those rights identified in DFARS clause 252.227-7015(b) (FEB 2014).

#### **Trademark information**

NETAPP, the NETAPP logo, and the marks listed at <a href="http://www.netapp.com/TM">http://www.netapp.com/TM</a> are trademarks of NetApp, Inc. Other company and product names may be trademarks of their respective owners.