Data sheet
Public

Deploy NetApp Trident CSI Plug-in on Cisco Container Platform with FlexPod

Contents

Executive summary

This document provides step-by-step procedures for deploying the NetApp Trident Container Storage Interface (CSI) plug-in on a Cisco® Container Platform Kubernetes tenant cluster in a FlexPod solution.

Introduction

A recent survey, summarized in Figure 1, showed that from 2018 to 2019, the use of containers in development, testing, and production environments grew rapidly. Most notably, the use of containers in production environments increased significantly. In 2019, 84 percent of respondents were using containers in production environments: an impressive jump from 73 percent in 2018, and from 23 percent in 2016. This growth is a result of organizations' increased trust in containers and use of them in user-facing applications. Another 14 percent of survey respondents have future plans to use containers in their production environments.

The proof-of-concept (PoC) environment is the only area in which the use of containers has shown a gradual decline over the past few years: an indication that containers are seen not as just an idea, but instead are being adopted in production deployments in the real world. Only slightly more than 2 percent of respondents reported no plans to use containers in 2019.

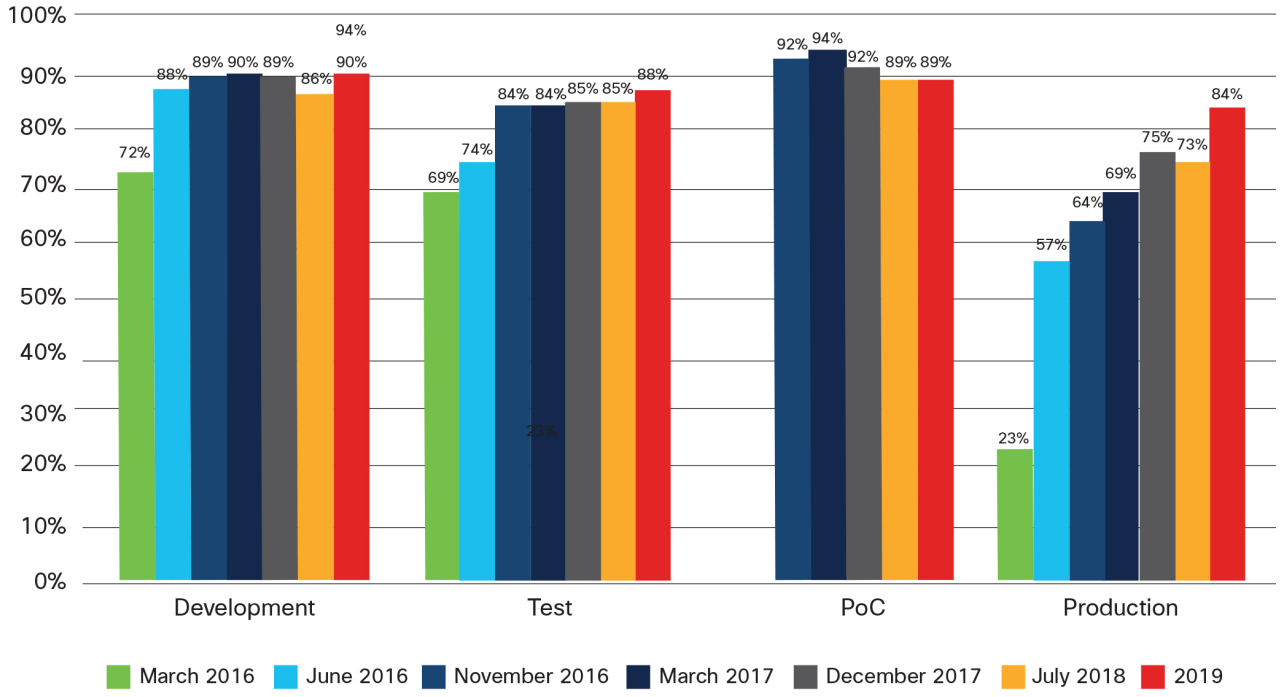


Figure 1.
Use of containers since 2016

In addition, container-based usage is growing rapidly at scale.

As organizations are trusting their production workloads to containers, they also are using more containers. The number of respondents using 249 or fewer containers decreased by 26 percent between 2018 and 2019. Conversely, the number of respondents using 250 or more containers increased by 28 percent, to more than half. The most significant change was in the number of organizations using fewer than 50 containers, which fell by 43 percent (Figure 2).

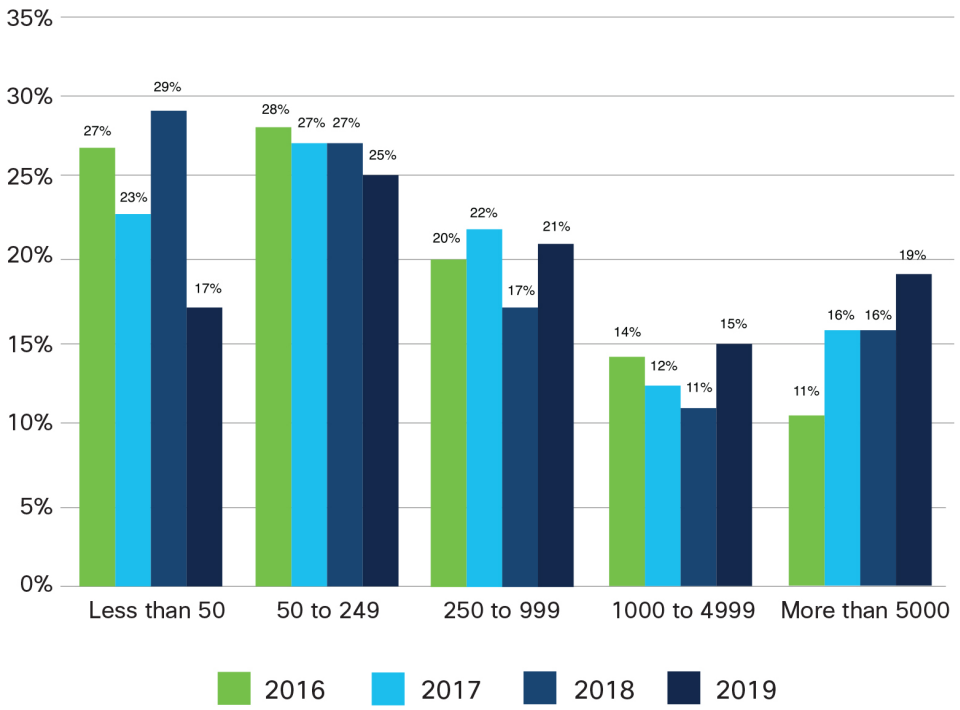


Figure 2.
Number of containers in production environments

Source: CNCF 2019 Survey, March 20: https://www.cncf.io/wp-content/uploads/2020/03/CNCF_Survey_Report.pdf

However, as Kubernetes has matured as a viable platform during this period of rapid adoption, the complexity of installing it and operating it over time at the enterprise level has increased. The container space is crowded with numerous products and services that all aim to provide Kubernetes at various levels of complexity and capability. Such a fragmented environment has led to a number of challenges for both IT operations and cloud administrators as well as for developers, including the following:

- IT operations and cloud administrators lack common tools to manage and deploy Kubernetes in heterogeneous environments.
- Developers lack common development experience with containers (best practices), slowing down development.

A guiding goal in the creation and continued rapid evolution of the Cisco Container Platform has been to help customers overcome these challenges and deliver what they want: the capability to install and operate a lightweight containers-as-a-service (CaaS) on-premises platform that is built on pure upstream Kubernetes. With Cisco Container Platform, customers can do the following:

- Quickly (within hours) and seamlessly deploy a production-class Kubernetes environment.
- Rapidly deploy containerized application clusters on-premises and in the public cloud of choice.
- Seamlessly manage applications throughout the lifecycle and dynamically scale them (with Cisco CloudCenter™ Suite, a cloud management platform).
- Quickly set up and accelerate artificial intelligence (AI) and machine-learning (ML) workloads using multiple graphics processing units (GPUs) and Kubeflow (automation framework).

- Eliminate shadow IT and development team silos by delivering a native Kubernetes experience through an easy-to-operate self-service portal, allowing developers to focus on delivering good software.
- Build once, run anywhere. Either build applications on-premises and seamlessly deploy them in the public cloud, or the reverse, through a consistent and secure environment.
- Deploy applications in cloud-native Kubernetes environments, getting the most from cloud investments while maintaining compliance and security through a single-pane deployment model.
- Help ensure that corporate policies (legal, finance, security, and privacy) are enforced by providing a common environment on which IT operations, development, and security teams can operate.
- Give developers access to the best platform and tools while enabling IT operations and security teams to maintain visibility and control over application and Kubernetes resource utilization across the premises and in public clouds.

Cisco Container Platform is a ready-to-use, lightweight, multicluster container management software platform for deploying production-class upstream Kubernetes environments and managing their lifecycle across on-premises and public cloud environments. Cisco Container Platform automates the installation of 100 percent upstream Kubernetes clusters with self-service and centralized automation and management capabilities (Figure 3).

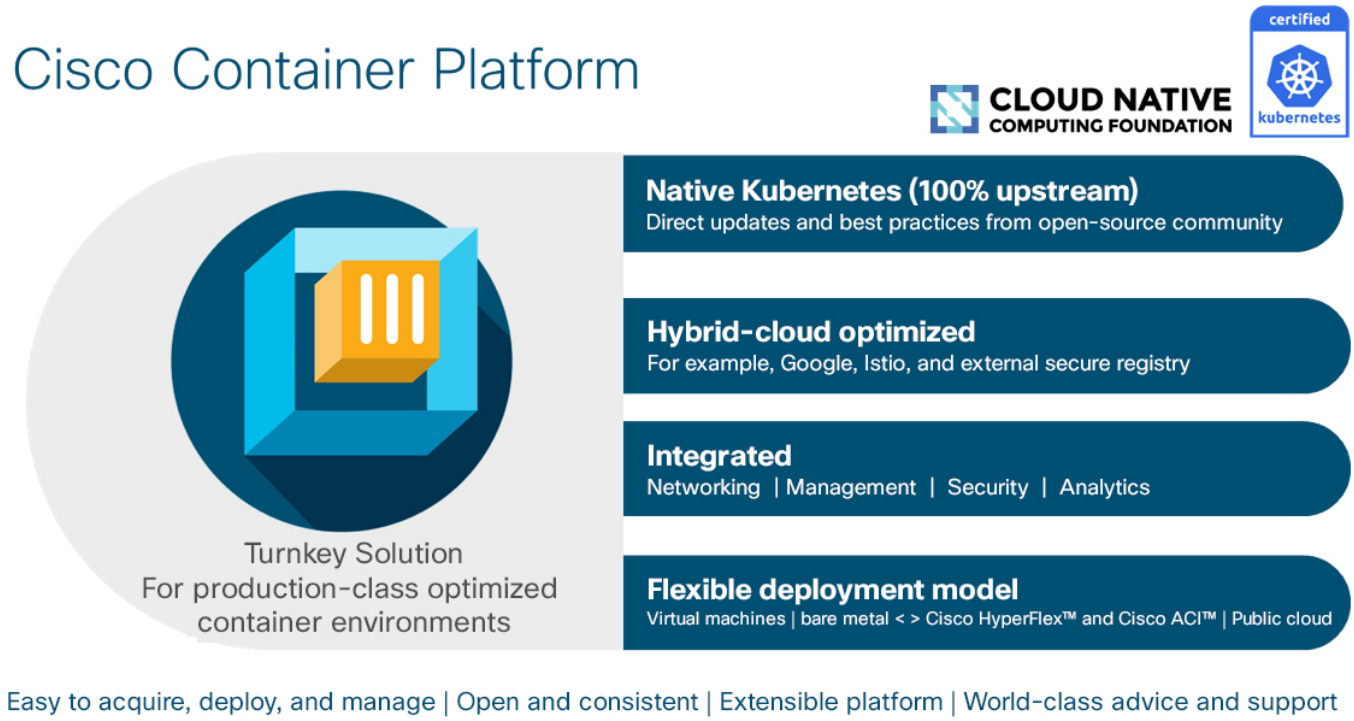


Figure 3.
Cisco Container Platform

When the Cisco Container Platform is installed in a FlexPod environment, the use of the NetApp Trident plug-in allows users to deploy Kubernetes persistent volume storage for use in a Kubernetes pod seamlessly. The NetApp Trident Container Storage Interface (CSI) plug-in is available for Network File System (NFS) and Small Computer System Interface over IP (iSCSI) back-end storage.

This document describes the procedures and references the necessary articles to perform the following tasks:

- Create a Cisco Container Platform Kubernetes tenant cluster.
- Install the NetApp Trident CSI plug-in on the created Kubernetes tenant cluster for both NFS back ends and iSCSI back ends.
- Deploy a Kubernetes persistent volume claim and pod and verify operations on Kubernetes and NetApp user interfaces.

Prerequisites

The following items need to be preconfigured before you begin the setup and configuration of a Cisco Container Platform tenant cluster on FlexPod:

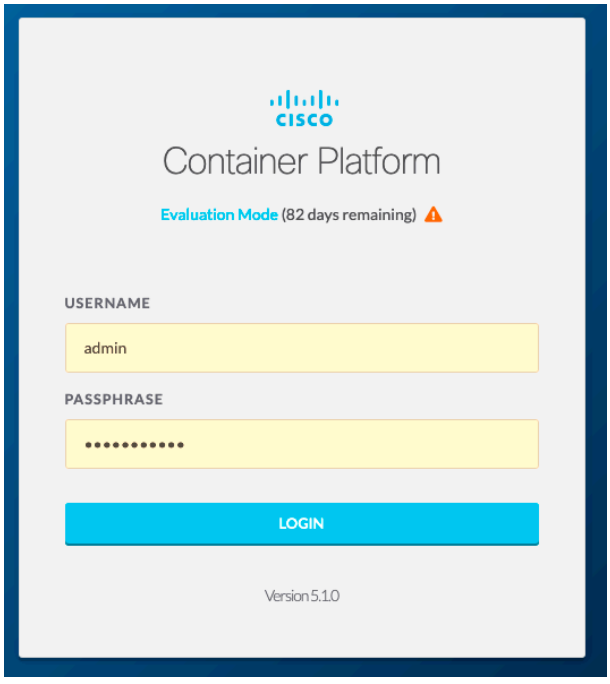
- A Linux host that has the kubectl client binary installed, that has access to the Internet to download the Trident CSI plug-in, and that can be routed to the created Cisco Container Platform Kubernetes tenant cluster
- A Cisco Container Platform control plane installed and configured to deploy Kubernetes tenant clusters
- A FlexPod environment with a storage virtual machine (SVM) configured to accept either NFS access or iSCSI initiator access

Create a Kubernetes cluster on the Cisco Container Platform control plane

Use the procedure described in this section to create a Kubernetes cluster on the Cisco Container Platform control plane.

Perform initial login

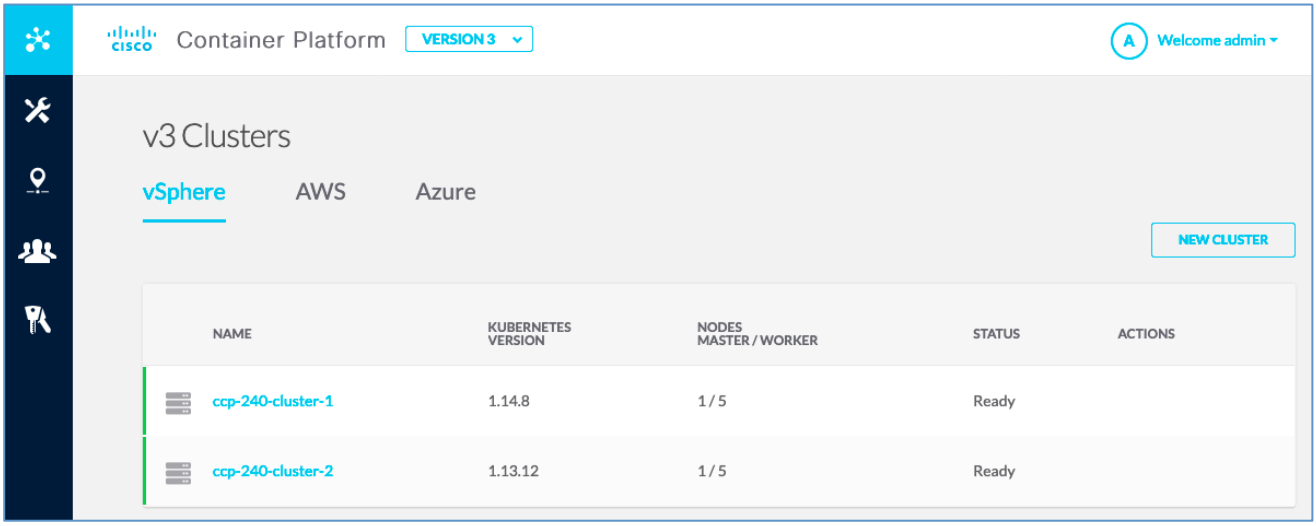
After installing the Cisco Container Platform management control plane (installation process), log in to the user interface with the necessary credentials.



Create a cluster

On the main login page, you should already be in the Cisco Container Platform Version 3 v3 Clusters section. If you are not, select Clusters on the menu at the left and make sure the drop-down Container Platform menu is set to Version 3.

Follow the online procedure for creating an on-premises cluster on VMware vSphere.

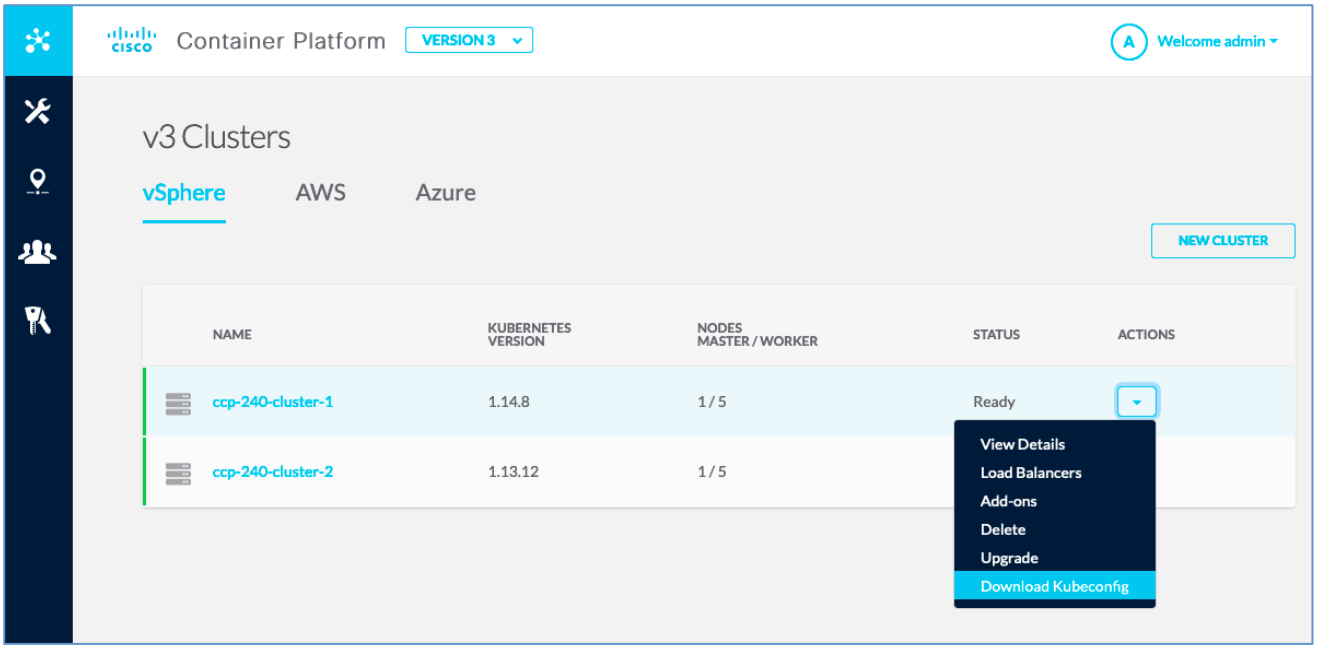


Download the kubeconfig file for access to the cluster using kubectl

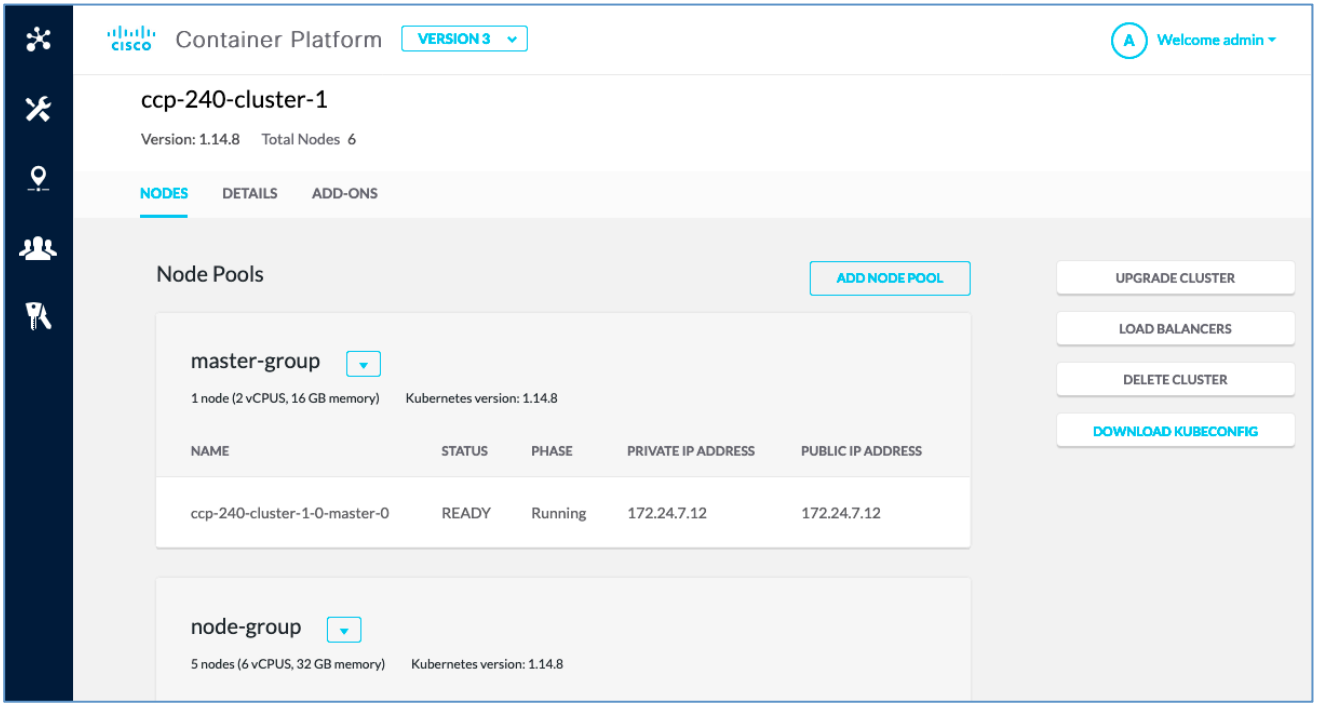
To install the NetApp Trident plug-in, a Linux host with the **kubectl** application must be installed. The kubeconfig file will allow the kubectl application to send commands to the Kubernetes cluster.

You can download the kubeconfig file using either of two methods:

- From the main cluster menu, open the drop-down menu in the Actions columns of the intended cluster and choose Download Kubeconfig.



- Select the cluster and then click the Download Kubeconfig button.



Copy the downloaded KUBECONFIG file to your Linux admin workstation and then export it as the **KUBECONFIG** system variable. To verify successful connection, run the **kubectl get nodes** command.

```
ubuntu@Ubuntu-jump:~$ export KUBECONFIG=~/Downloads/ccp-240-cluster1.yaml
ubuntu@Ubuntu-jump:~$ kubectl get nodes
NAME                              STATUS   ROLES    AGE    VERSION
ccp-240-cluster-1-0-master-0      Ready    master   5d2h   v1.14.8
ccp-240-cluster-1-1-node-gr-0     Ready    <none>   3d8h   v1.14.8
ccp-240-cluster-1-1-node-gr-1     Ready    <none>   3d8h   v1.14.8
ccp-240-cluster-1-1-node-gr-2     Ready    <none>   3d8h   v1.14.8
ccp-240-cluster-1-1-node-gr-3     Ready    <none>   3d8h   v1.14.8
ccp-240-cluster-1-1-node-gr-4     Ready    <none>   3d8h   v1.14.8
```

Collect Kubernetes node IP addresses and iSCSI initiators

Using the complementary private Secure Shell (SSH) key on the Linux host, run the following script to obtain the IP addresses of the nodes and iSCSI initiator names. This information can be used in NetApp ONTAP System Manager to configure intended NFS or iSCSI access for the dynamically created persistent volume claims created by the Kubernetes cluster.

```
ubuntu@Ubuntu-jump:~$ while read ip; do echo -n "IPaddress=${ip} - "; ssh $ip cat
/etc/iscsi/initiatorname.iscsi </dev/null; done < <(kubectl get no -o
jsonpath='{range.items[*].status.addresses[?(@.type=="InternalIP")]}{.address}{"\n"}{end}')
IPaddress=192.168.92.184 - InitiatorName=iqn.2005-03.org.open-iscsi:e89ed91cfdf
IPaddress=192.168.92.185 - InitiatorName=iqn.2005-03.org.open-iscsi:11868af880
IPaddress=192.168.92.186 - InitiatorName=iqn.2005-03.org.open-iscsi:c78f63f2923
IPaddress=192.168.92.187 - InitiatorName=iqn.2005-03.org.open-iscsi:6191668fc1
IPaddress=192.168.92.188 - InitiatorName=iqn.2005-03.org.open-iscsi:adf51eec7c4d
```

```
IPaddress=192.168.92.189 - InitiatorName=iqn.2005-03.org.open-iscsi:1a4b9383b059
IPaddress=192.168.92.191 - InitiatorName=iqn.2005-03.org.open-iscsi:a3e351201f8a
IPaddress=192.168.92.190 - InitiatorName=iqn.2005-03.org.open-iscsi:b67d8c1b46b
```

Install the NetApp Trident CSI plug-in

This section presents the installation procedure for the NetApp Trident CSI plug-in as described in the NetApp Trident documentation.

Qualify the Kubernetes cluster

Verify the version, permissions, and network connectivity for the NetApp Trident plug-in.

```
ubuntu@Ubuntu-jump:~$ kubectl version
Client Version: version.Info{Major:"1", Minor:"14", GitVersion:"v1.14.8",
GitCommit:"211047e9a1922595eaa3a1127ed365e9299a6c23", GitTreeState:"clean", BuildDate:"2019-
10-15T12:11:03Z", GoVersion:"go1.12.10", Compiler:"gc", Platform:"linux/amd64"}
Server Version: version.Info{Major:"1", Minor:"14", GitVersion:"v1.14.8",
GitCommit:"211047e9a1922595eaa3a1127ed365e9299a6c23", GitTreeState:"clean", BuildDate:"2019-
10-15T12:02:12Z", GoVersion:"go1.12.10", Compiler:"gc", Platform:"linux/amd64"}
ubuntu@Ubuntu-jump:~$
ubuntu@Ubuntu-jump:~$ # Are you a Kubernetes cluster administrator?
ubuntu@Ubuntu-jump:~$ kubectl auth can-i '*' '*' --all-namespaces
yes
ubuntu@Ubuntu-jump:~$
ubuntu@Ubuntu-jump:~$ # Can you launch a pod that uses an image from Docker Hub and can
reach your
ubuntu@Ubuntu-jump:~$ # storage system over the pod network?
ubuntu@Ubuntu-jump:~$ kubectl run -i --tty ping --image=busybox --restart=Never --rm -- ping
192.168.92.10
If you don't see a command prompt, try pressing enter.
64 bytes from 192.168.92.10: seq=1 ttl=63 time=0.145 ms
64 bytes from 192.168.92.10: seq=2 ttl=63 time=0.144 ms
64 bytes from 192.168.92.10: seq=3 ttl=63 time=0.185 ms
64 bytes from 192.168.92.10: seq=4 ttl=63 time=0.142 ms
^C
--- 192.168.92.10 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max = 0.142/0.211/0.441 ms
pod "ping" deleted
```

Download the Trident CSI plug-in

Download the latest version of the Trident installer bundle from the Downloads section and extract the files.

The version used for this document is Release 20.01.0.

```
ubuntu@Ubuntu-jump:~/netapp$ wget -q
https://github.com/NetApp/trident/releases/download/v20.01.0/trident-installer-
20.01.0.tar.gz
ubuntu@Ubuntu-jump:~/netapp$ tar -xf trident-installer-20.01.0.tar.gz
ubuntu@Ubuntu-jump:~/netapp$ cd trident-installer
/home/ubuntu/trident-installer
```

Install Trident CSI on Kubernetes

Run the trident **install** command and verify that the trident pods are running and that the version is correct.

```
ubuntu@Ubuntu-jump:~/netapp/trident-installer$ ./tridentctl install -n trident
INFO Starting Trident installation.                namespace=trident
INFO Created namespace.                            namespace=trident
INFO Created service account.
INFO Created cluster role.
INFO Created cluster role binding.
INFO Created custom resource definitions.          namespace=trident
INFO Added finalizers to custom resource definitions.
INFO Created Trident pod security policy.
INFO Created Trident service.
INFO Created Trident secret.
INFO Created Trident deployment.
INFO Created Trident daemonset.
INFO Waiting for Trident pod to start.
INFO Trident pod started.                          namespace=trident pod=trident-csi-
6bbd889f9f-bszg9
INFO Waiting for Trident REST interface.
INFO Trident REST interface is up.                 version=20.01.0
INFO Trident installation succeeded.


ubuntu@Ubuntu-jump:~/netapp/trident-installer$ kubectl get pod -n trident
NAME                           READY   STATUS    RESTARTS   AGE
trident-csi-4bvx9              2/2     Running   0          47s
trident-csi-6bbd889f9f-bszg9   3/3     Running   0          47s
trident-csi-9qph7              2/2     Running   0          47s
trident-csi-f7cjv              2/2     Running   0          47s
trident-csi-hkjdd              2/2     Running   0          47s
trident-csi-nzdrl              2/2     Running   0          47s
trident-csi-vp82h              2/2     Running   0          47s
trident-csi-wwc28              2/2     Running   0          47s
```

```
trident-csi-xbngs               2/2     Running   0          47s
ubuntu@Ubuntu-jump:~/netapp/trident-installer$ ./tridentctl -n trident version
+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 20.01.0        | 20.01.0        |
+----------------+----------------+
```

The Trident CSI plug-in is now running. You can configure either NFS or iSCSI, or both protocols, to connect the NetApp storage.

Configure NetApp NFS and NetApp iSCSI back ends for Trident CSI

Edit and apply the back-end JavaScript Object Notation (JSON) templates for back-end protocol you want to use.

Configure NetApp NFS: Edit and apply back-end JSON template

From the sample-input directory found in the Trident installer package, copy the backend.json file up one directory, edit it with the credential information for your NetApp storage, and apply it to your Kubernetes cluster.

```
ubuntu@Ubuntu-jump:~/netapp/trident-installer$ cat backend.json
{
    "version": 1,
    "storageDriverName": "ontap-nas",
    "backendName": "aa14-a800one",
    "managementLIF": "192.168.92.10",
    "dataLIF": "192.168.92.51",
    "svm": "CCP-SVM",
    "username": "xxxxxx",
    "password": "PaxxWoxd"
}
ubuntu@Ubuntu-jump:~/netapp/trident-installer$ ./tridentctl -n trident create backend -f
backend.json
+-------------+----------------+--------------------------------------+--------+---------+
|     NAME    | STORAGE DRIVER |                 UUID                 | STATE  | VOLUMES |
+-------------+----------------+--------------------------------------+--------+---------+
| aa14-a800one | ontap-nas      | 2b2f70f6-a549-498f-81a0-e97beefa3a2d | online |       0 |
+-------------+----------------+--------------------------------------+--------+---------+
```

Configure NetApp iSCSI: Edit and apply back-end JSON template

From the sample-input directory found in the Trident installer package, copy the backend.json file up one directory, edit it with the credential information for your NetApp storage system, and apply it to your Kubernetes cluster.

```
ubuntu@Ubuntu-jump:~/netapp/trident-installer$ cat backend.json
{
    "version": 1,
    "storageDriverName": "ontap-san",
    "backendName": " aa14-a800iSCSI",
    "managementLIF": "192.168.92.10",
    "dataLIF": "192.168.92.54",
    "svm": "CCP-VMs",
    "username": "xxxxxx",
    "password": "PaxxWoxd"
}

ubuntu@Ubuntu-jump:~/netapp/trident-installer$ ./tridentctl -n trident create backend -f
backend.json
+----------------+----------------+--------------------------------------+--------+---------+
|      NAME      | STORAGE DRIVER |                 UUID                 | STATE  | VOLUMES
|
+----------------+----------------+--------------------------------------+--------+---------+
| aa14-a800iSCSI | ontap-san      | 9e031ed7-f179-45ba-9391-d2e67a42d66a | online |       0
|
+----------------+----------------+--------------------------------------+--------+---------+
```

Test NetApp Trident CSI persistent volumes

Use the procedures in this section to test the NetApp Trident CSI persistent volumes.

Create a Kubernetes storage class

From the sample-input directory found in the trident-installer package, copy the storage-class-csi.yaml.templ file up one directory as **storage-class-basic.yaml**. Edit the file and replace **__BACKEND_TYPE__** with the storage driver name. In the preceding examples, this name is either **ontap-nas** or **ontap-san**.

```
ubuntu@Ubuntu-jump:~/netapp/trident-installer$ cat storage-class-basic.yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: basic
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
```

```
ubuntu@Ubuntu-jump:~/netapp/trident-installer$ kubectl create -f storage-class-basic.yaml
storageclass.storage.k8s.io/basic created

ubuntu@Ubuntu-jump:~/netapp/trident-installer$ kubectl get sc
NAME                 PROVISIONER                    AGE
basic                csi.trident.netapp.io          4s
standard (default)   kubernetes.io/vsphere-volume   40m

ubuntu@Ubuntu-jump:~/netapp/trident-installer$ ./tridentctl -n trident get sc basic -o json
{
  "items": [
    {
      "Config": {
        "version": "1",
        "name": "basic",
        "attributes": {
          "backendType": "ontap-san"
        },
        "storagePools": null,
        "additionalStoragePools": null
      },
      "storage": {
        "aa14-a800iSCSI": [
          "aa14_a800_1_NVME_SSD_1",
          "aa14_a800_2_NVME_SSD_1"
        ]
      }
    }
  ]
}
```

Provision a dynamic persistent volume

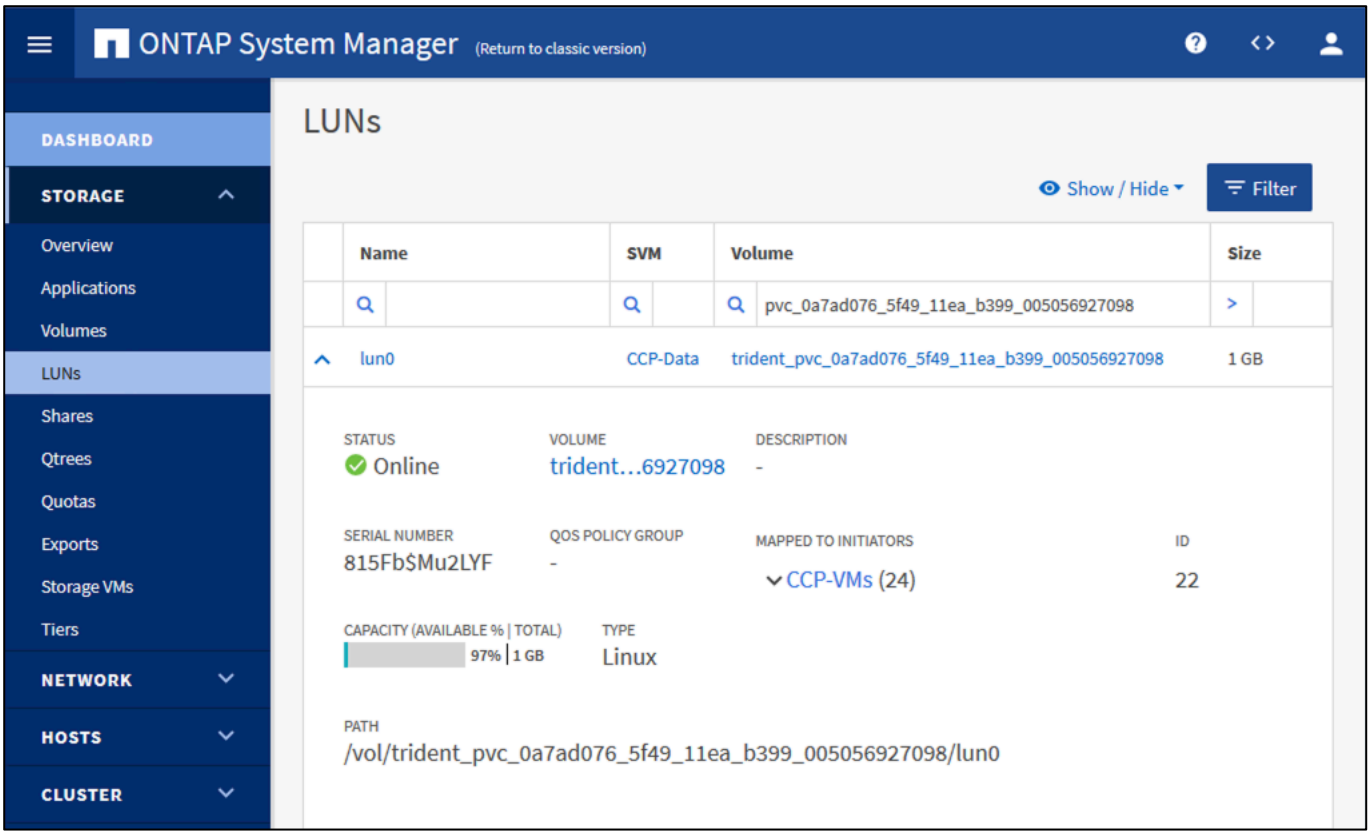
Create a PersistentVolumeClaim (PVC) claim by using the sample-input/pvc-basic.yaml file from the Trident installer package.

```
ubuntu@Ubuntu-jump:~/netapp/trident-installer$ kubectl create -f sample-input/pvc-basic.yaml
persistentvolumeclaim/basic created

ubuntu@Ubuntu-jump:~/netapp/trident-installer$ kubectl get pvc
NAME    STATUS   VOLUME                                     CAPACITY ACCESS STORAGECLASS   AGE
basic   Bound    pvc-0a7ad076-5f49-11ea-b399-005056927098 1Gi      RWO    basic          46s
```

Validate the pod in the NetApp management console

Log in to NetApp ONTAP System Manager and verify the volume ID of the persistent volume claim. You need to replace dashes with underscores when finding the volume.



Mount the volume in a Kubernetes pod

Run the command shown here to start a Kubernetes pod that will mount the persistent volume claim.

```
ubuntu@Ubuntu-jump:~/netapp/trident-installer$ cat << EOF > task-pv-pod.yaml
kind: Pod
apiVersion: v1
metadata:
  name: task-pv-pod
spec:
  volumes:
    - name: task-pv-storage
      persistentVolumeClaim:
       claimName: basic
  containers:
    - name: task-pv-container
      image: nginx
      ports:
        - containerPort: 80
```

```
          name: "http-server"
      volumeMounts:
        - mountPath: "/usr/share/nginx/html"
          name: task-pv-storage
EOF
```

```
ubuntu@Ubuntu-jump:~/netapp/trident-installer$ kubectl create -f task-pv-pod.yaml
pod/task-pv-pod created
```

Verify that the pod is running.

```
ubuntu@Ubuntu-jump:~/netapp/trident-installer$ kubectl get pods
NAME          READY    STATUS    RESTARTS   AGE
task-pv-pod   1/1      Running   0          16s
```

Check the mount in the pod.

A network-attached storage (NAS)-backed pod will display an NFS mount path.

```
ubuntu@Ubuntu-jump:~/netapp/trident-installer$ kubectl exec -it task-pv-pod -- df -h
/usr/share/nginx/html
Filesystem      Size  Used Avail Use% Mounted on
192.168.92.51:/trident_pvc_acb37d38_47aa_11ea_83e6_00505692ecdf
                976M  3.6M  906M   1% /usr/share/nginx/html
```

An iSCSI-backed pod will show a block device path.

```
ubuntu@Ubuntu-jump:~/netapp/trident-installer$ kubectl exec -it task-pv-pod -- df -h
    /usr/share/nginx/html
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda1       976M  2.6M  907M   1% /usr/share/nginx/html
```

Perform cleanup operations

After the validation process is complete, run the following commands to delete both the Kubernetes pod and the persistent volume claim.

```
ubuntu@Ubuntu-jump:~/netapp/trident-installer$ kubectl delete pod task-pv-pod
pod "task-pv-pod" deleted
```

```
ubuntu@Ubuntu-jump:~/netapp/trident-installer$ kubectl delete pvc basic
persistentvolumeclaim "basic" deleted
```

```
ubuntu@Ubuntu-jump:~/netapp/trident-installer$ kubectl get pod,pvc
No resources found.
```

Conclusion

Cisco Container Platform is well suited for use with a NetApp storage device for persistent volumes in Kubernetes using either an NAS or iSCSI interface.

For more information

- https://netapp-trident.readthedocs.io
- https://www.cisco.com/c/en/us/products/cloud-systems-management/container-platform/index.html

Americas Headquarters
Cisco Systems, Inc.
San Jose, CA

Asia Pacific Headquarters
Cisco Systems (USA) Pte. Ltd.
Singapore

Europe Headquarters
Cisco Systems International BV Amsterdam,
The Netherlands

Cisco has more than 200 offices worldwide. Addresses, phone numbers, and fax numbers are listed on the Cisco Website at **https://www.cisco.com/go/offices**.

Printed in USA 220104.7 06/20