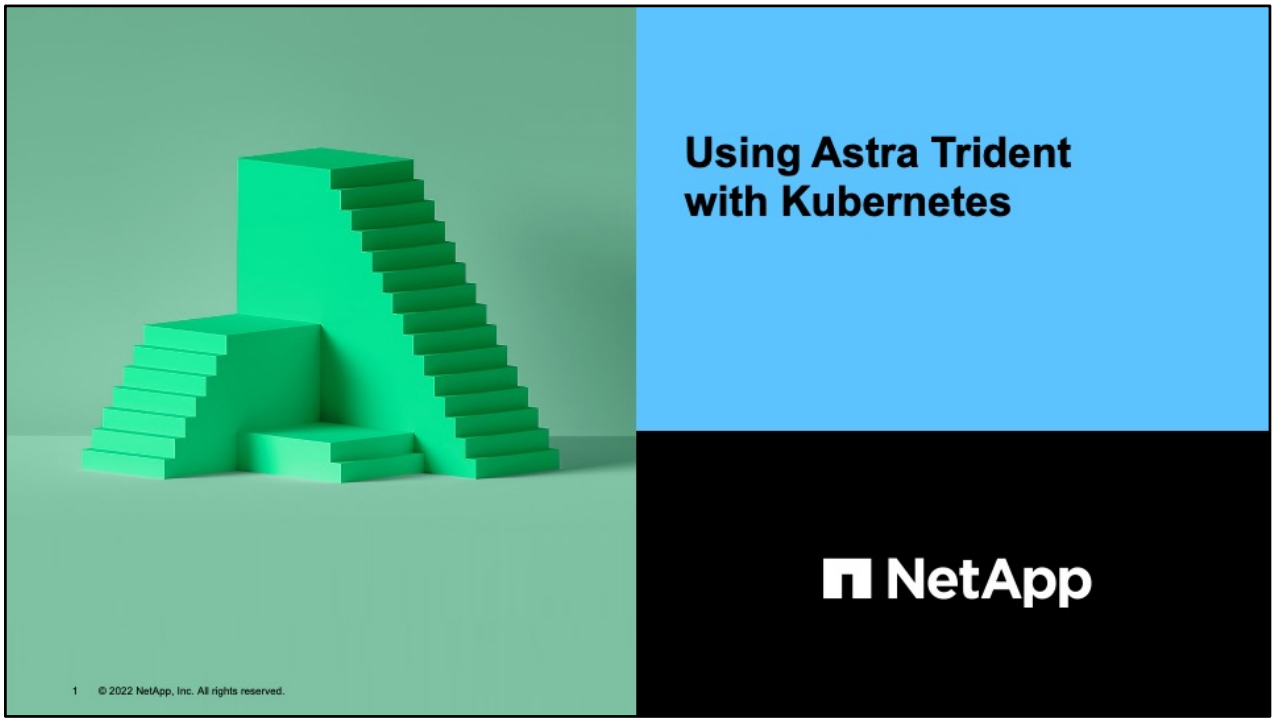
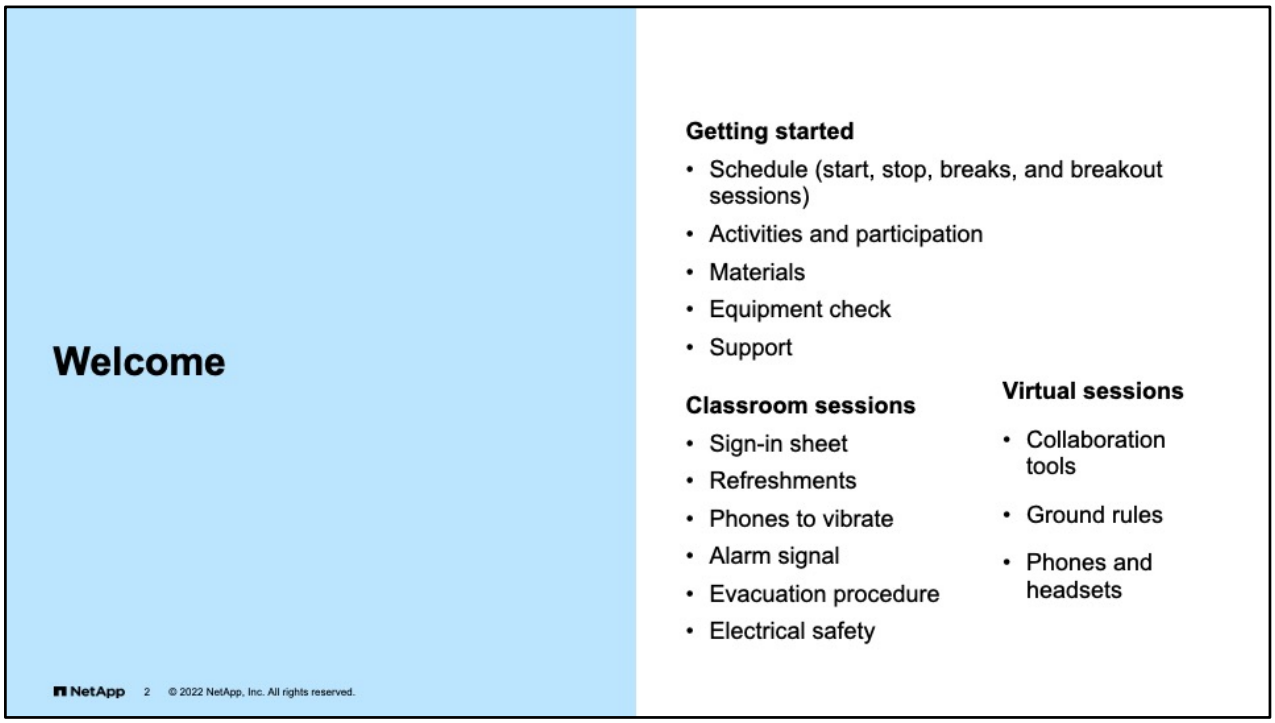


STRSW-ILT-UATWK
Kubernetes Version 1.23.1
Astra Trident 22.01.0
ONTAP 9.9.1



Instructor Notes

Review the schedule, class structure, and logistics details.

If students have not retrieved their courseware, direct them to the download site.

Use the Getting Started exercise in the module to do an equipment check of connectivity to the lab environment.

For a course delivered virtually, be sure to discuss the following topics.

- Collaboration tools: Explain the collaboration tools that are used during the class. If necessary, refer the students to a demonstration video or demonstrate the tool.
- Remind students to turn off distractions, be prepared to be called on, raise hand or use chat to ask questions, be patient waiting for responses to chat, and use the coffee cup icon to indicate temporary absence.
- Suggest appropriate phones and headset. Non-battery telephones have best results. Voice over IP (VoIP) might be spotty.
- Explain how breakout sessions work, if applicable.

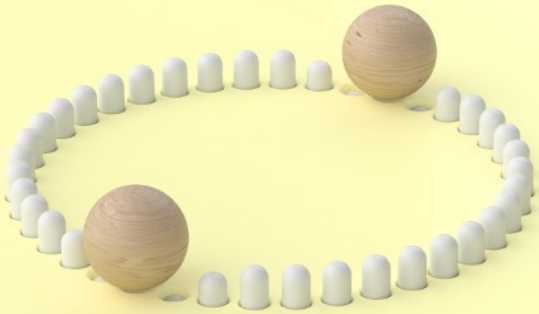
Student Notes

Set your phone to vibrate to prevent disturbing your fellow students. NetApp realizes that work does not always stop when you are in training. If you need to take a call, step outside the classroom.

Instructions

Tell everyone the following:

- Your name
- The company you work for
- Your Docker experience
- Your cloud experience (AWS, Google Cloud, Azure)
- Your Kubernetes experience
- Your NetApp Astra family experience



Instructor Notes

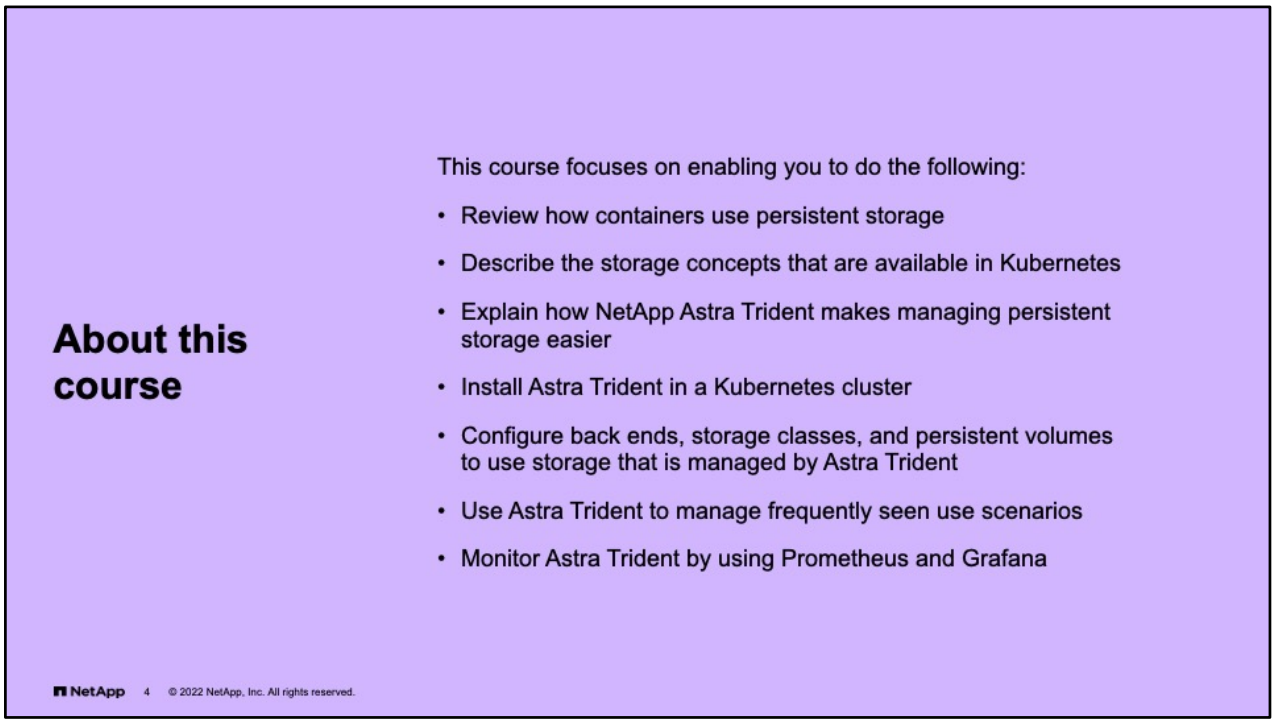
Use this slide to introduce yourself and to prompt students to state their names and their professions.

In VILT classes, you can have students introduce themselves through the chat window or through the conference connection. Remind students to unmute and mute their lines when speaking.

In VILT classes, you can insert the optional Map Your Location slide, from the Interactive Formats section, to personalize introductions and give students the opportunity to practice with the whiteboard tools.

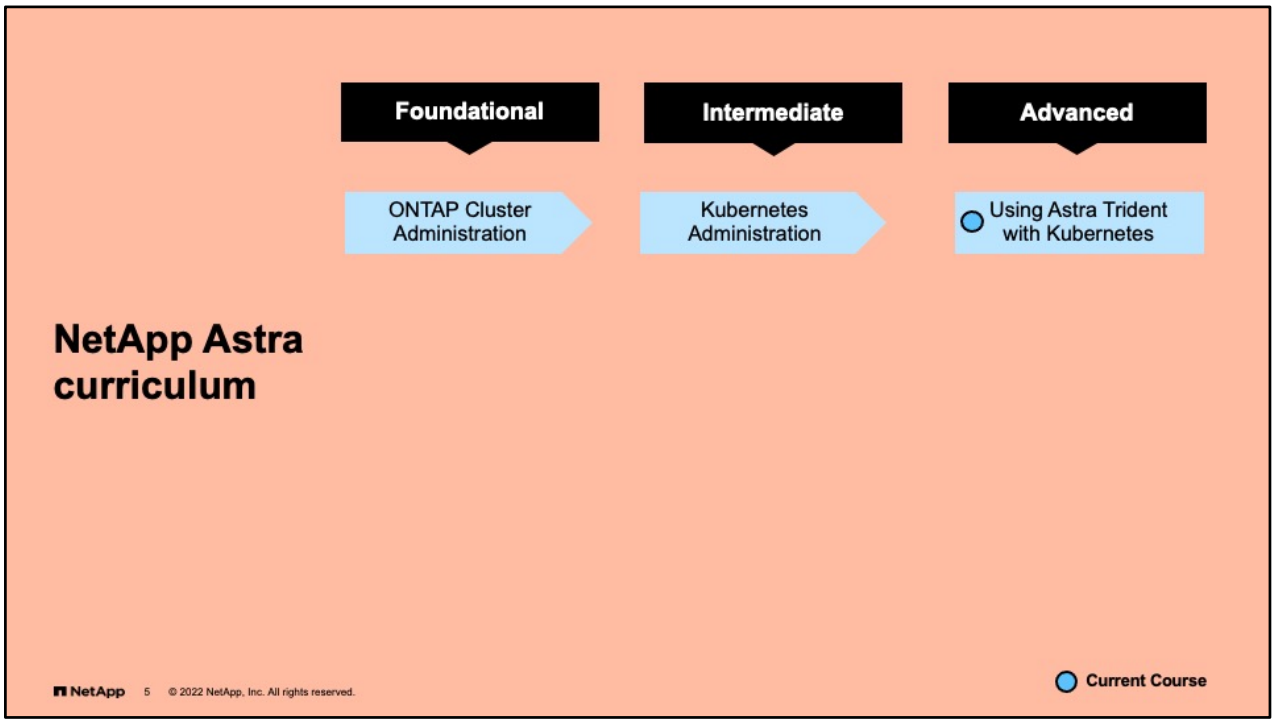
Student Notes

Take time to meet and get to know one another. If you are participating in a NetApp Virtual Live class, your instructor asks you to use the chat window or a conference connection to speak. If you are using a conference connection, unmute your line to speak and be sure to mute again after you speak.



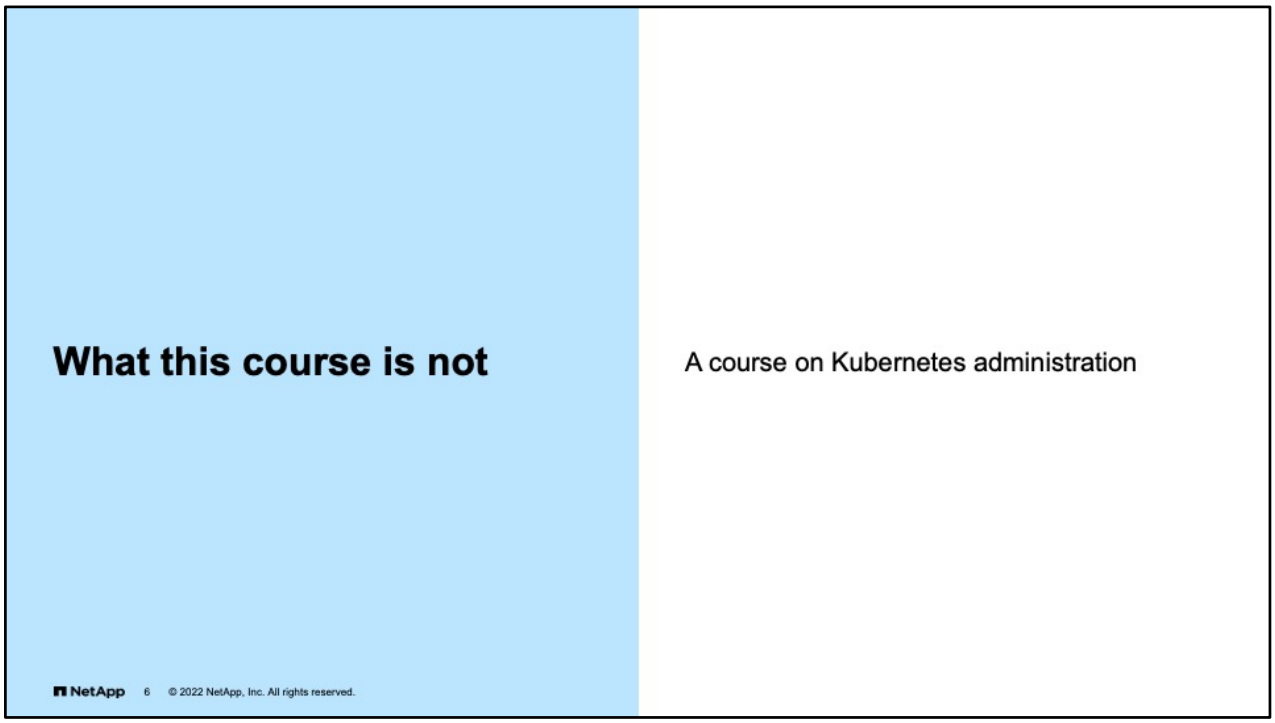
Instructor Notes

Student Notes



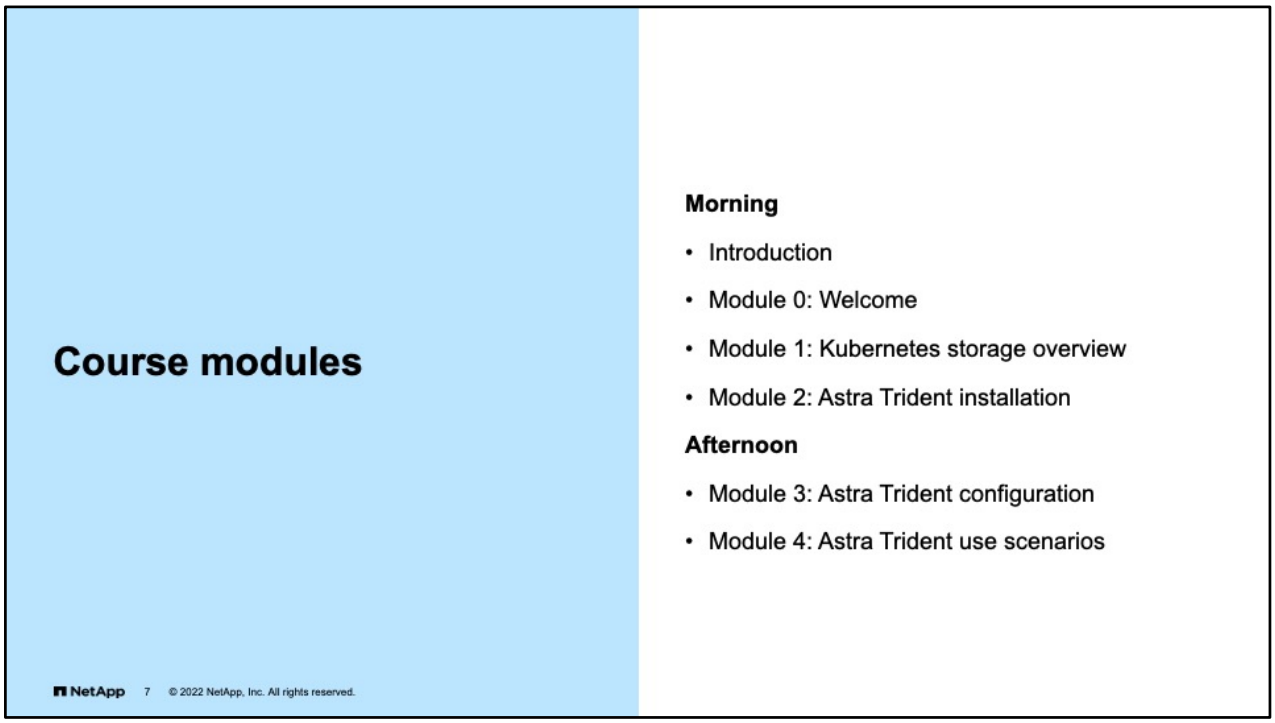
Instructor Notes

Student Notes



Instructor Notes

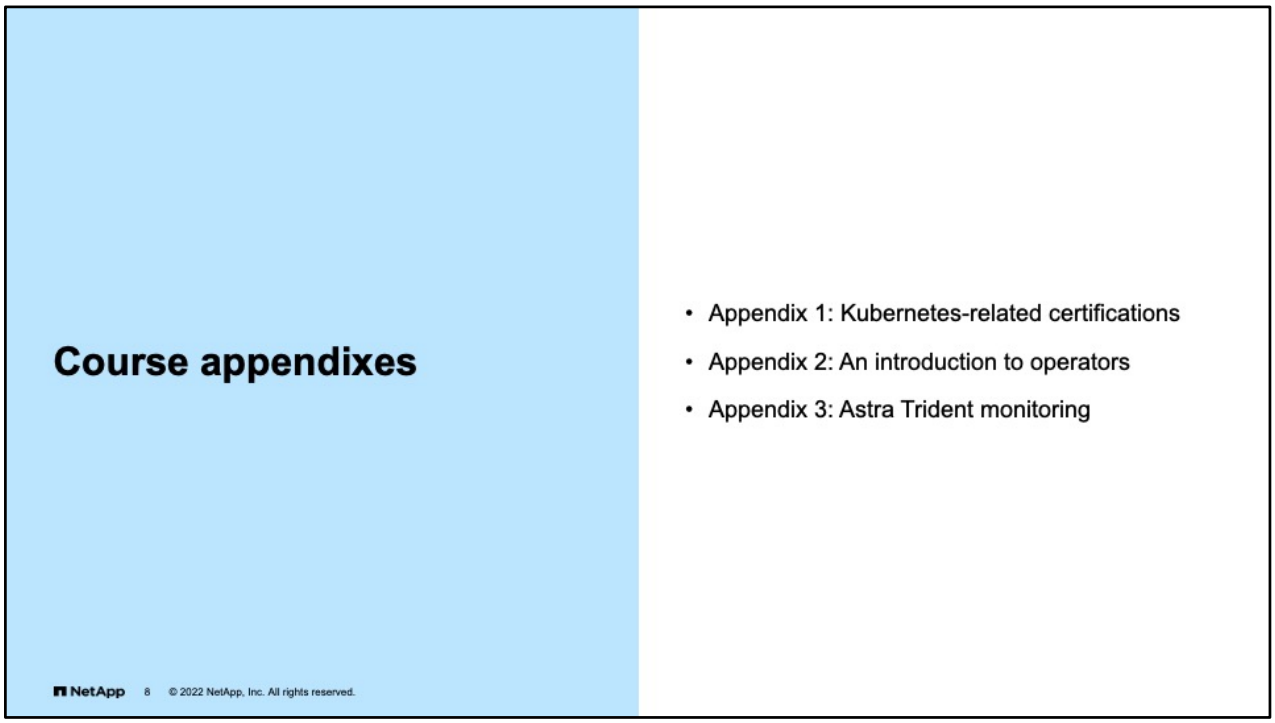
Student Notes



Instructor Notes

Student Notes

This schedule is based on average completion times for modules. Each class includes learners with different backgrounds and experience levels. This situation means that some modules might take more or less time to complete. Your instructor will adjust the schedule accordingly for breaks, meals, and the start time of each module.

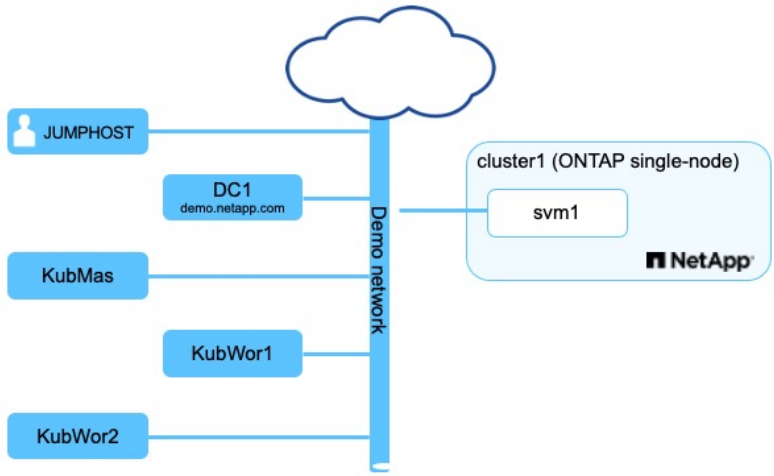


Instructor Notes

Student Notes

Exercise equipment: Basic architecture

Location	User name	Password
Windows	Administrator	Netapp1!
KubMas	root	Netapp1!
KubWor1	root	Netapp1!
KubWor2	root	Netapp1!
ONTAP	admin	Netapp1!



Content: https://github.com/NetApp-Learning-Services/STRSW-ILT-UATWK

Instructor Notes

For equipment-based offerings, provide an equipment kit and connection information to each participant. Explain each server in the exercise kit.
Verify that license codes are installed. Refer to the hidden Courseware Licenses slide at the end of this module.

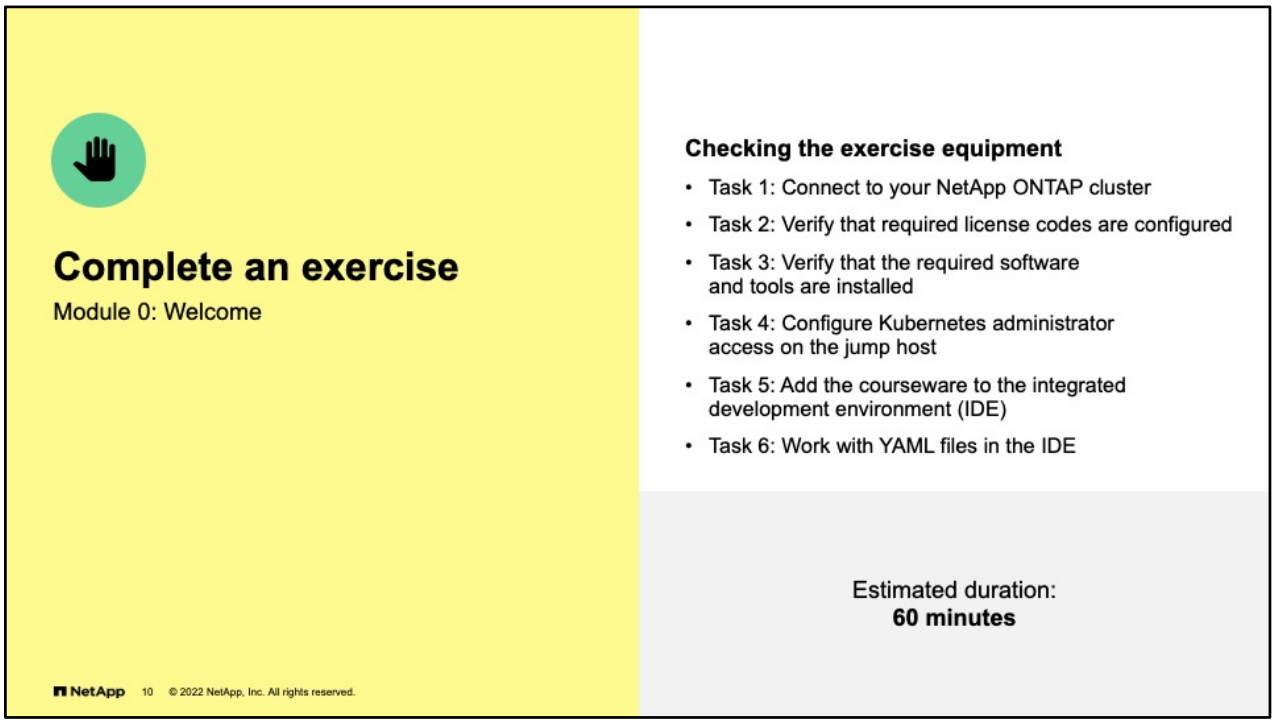
Contact information for courseware licenses:

- Courses offered directly through NetApp University: ng-nau-labsupport@netapp.com
- Courses provided by Authorized Learning Partners: Contact your direct manager or supervisor or NetApp University ALP Program Manager, ng-netappu-nci@netapp.com.

Student Notes

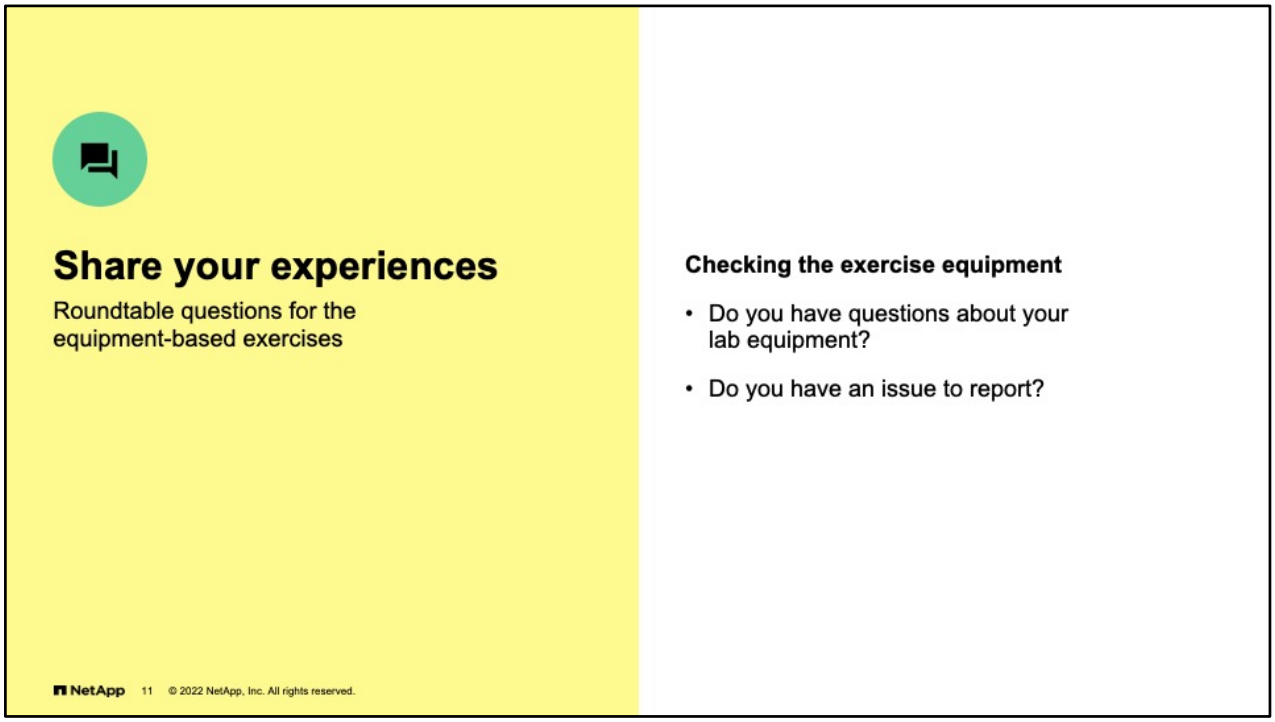
Launch your exercise equipment kit from your laptop or from the classroom desktop. To connect to your exercise equipment, use Remote Desktop Connection or the NetApp Learning Services portal.

The Windows Server is your jump host to access the lab environment.



Instructor Notes

Student Notes
See the instructions in your Exercise Guide.

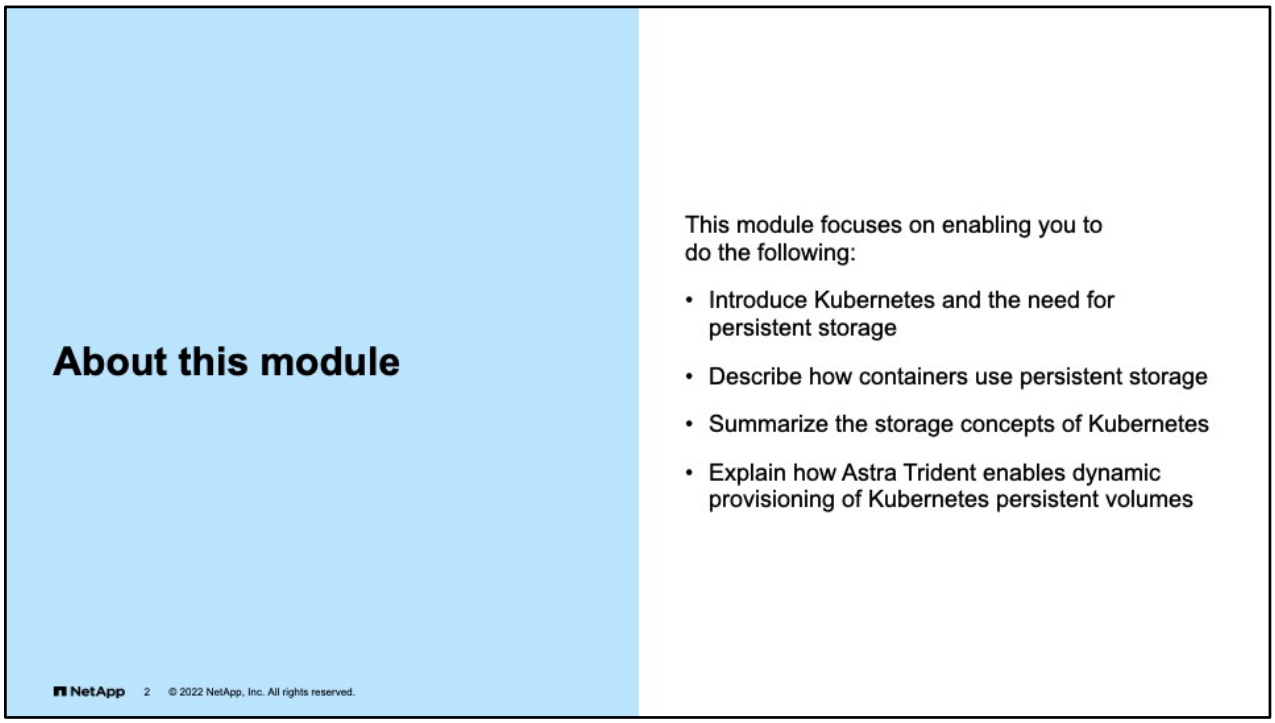


Instructor Notes
Display while students are completing the exercise(s)

Student Notes

Instructor Notes

Student Notes



Instructor Notes

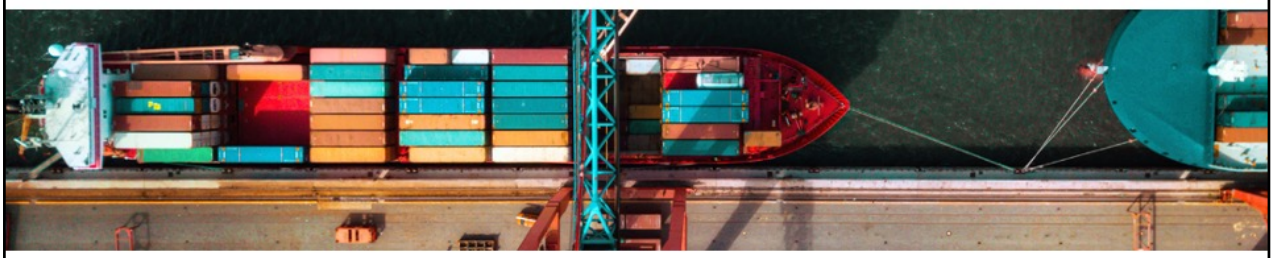
Student Notes

Instructor Notes

Student Notes

Modern IT is adopting Kubernetes

Benefits of a containerized environment



Agility
- Build and deploy apps faster
- Automate almost everything

Hybrid and reusable
- Reuse existing pipeline in the public, private, and hybrid cloud
- Build once, deploy multiple times anywhere

Cost optimization
- Pay for what you use, not what you own
- Smaller apps > Fewer resources > Lower cost

Stability
- Help developers deliver business value above all
- Focus on business outcomes, not technology outcomes

NetApp

Instructor Notes

Student Notes

The missing piece

Data persistence

- Containers: ephemeral
- Applications: not ephemeral
- Need for persistence:
  - Data protection
  - Availability
  - Scalability
  - Performance

Instructor Notes

Student Notes

Top challenges when deploying containers

Data persistence, management, compliance, and disaster recovery are top challenges

Source: IDC's Enterprise Infrastructure Market Pulse, Q2 2019, IDC, April/May 2019

Instructor Notes

Student Notes

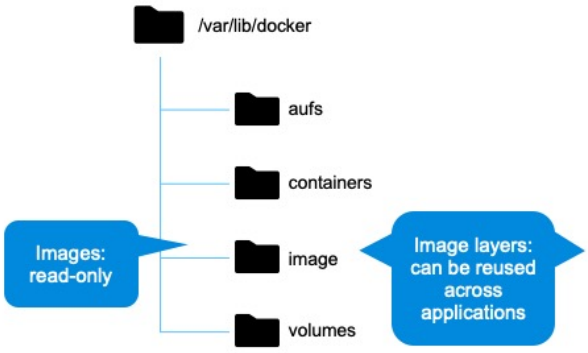
Instructor Notes

Student Notes

Container storage

Layers are generated from the container build file

- Containers store files in a file structure:



- Following is the layered architecture:

Container build file (dockerfile):

```
FROM Ubuntu
RUN apt-get update && apt-get -y install python
RUN pip install flask flask-mysql
COPY . /opt/code
ENTRYPOINT FLASK_APP=/opt/code/app.py flask run
```

- Build the container:

```
docker build dockerfile -t myapp
```

Layer 1: Base Ubuntu Layer	120 MB
Layer 2: APT update and python	306 MB
Layer 3: PIP packages installed	6.3 MB
Layer 4: Add custom app source	229 B
Layer 5: Change entrypoint	0 B

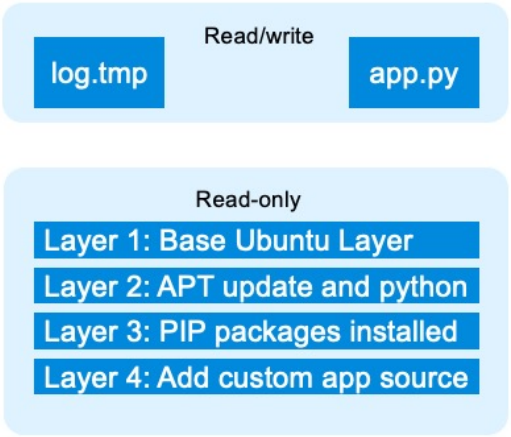
Instructor Notes

Student Notes

Container storage layers

Read-only layers with a read/write container cache

- Images layers:
  - Are predefined and read-only
  - Write new files to a read/write "container" layer during the container's execution
  - Write modified files from the read-only layers (for example, app.py) to the "container" layer (an operation called copy-on-write)
- Read/write (container) layer:
  - Persists during the life of the container
  - Is deleted when the container is stopped



Instructor Notes

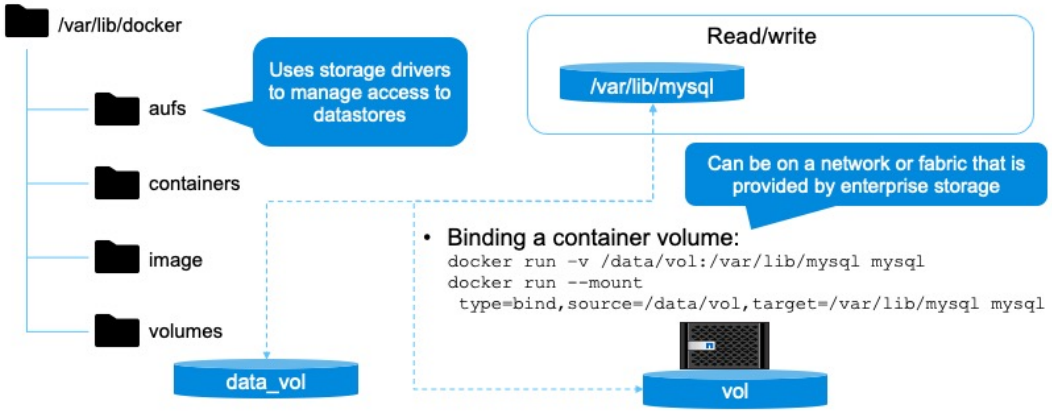
Student Notes

Container volumes

Mounting and binding

- Mounting a container volume:

```
docker run -v data_vol:/var/lib/mysql mysql
```



- Binding a container volume:

```
docker run -v /data/vol:/var/lib/mysql mysql
docker run --mount
 type=bind,source=/data/vol,target=/var/lib/mysql mysql
```


Instructor Notes

Student Notes

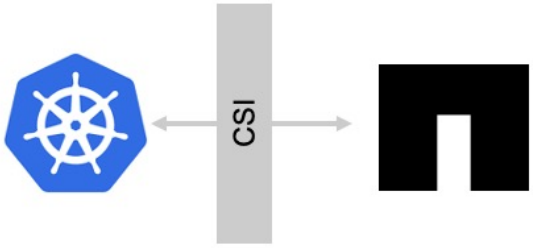
Instructor Notes

Student Notes

Kubernetes uses and extends container storage

Managing volumes with legacy drivers and the new CSI standard

- Provides "legacy" storage volumes:
  - Local
  - Azure File Storage
  - ConfigMap
  - GCE Docker
  - VMware vSphere Storage
  - iSCSI
  - NFS
  - FC
  - Amazon Elastic Block Store
  - Host Path
- Adopted new Container Storage Interface (CSI) standard:
  - Determines how to create, manage, and delete volumes
  - Has been adopted by NetApp



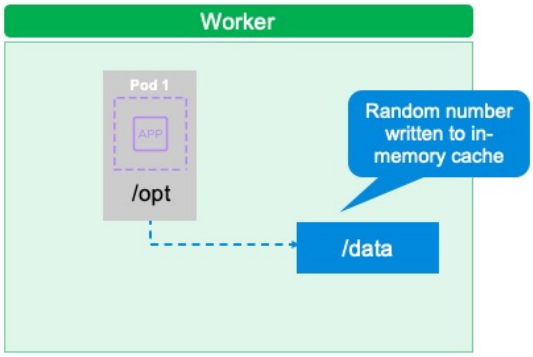
Instructor Notes

Student Notes

For more information about the legacy storage volumes, see https://kubernetes.io/docs/concepts/storage/volumes/.
For more information about the CSI standard, see https://github.com/container-storage-interface

Volumes in Kubernetes

Simple volume with emptyDir



Scratch space volume example:

```
apiVersion: v1
kind: Pod
metadata:
  name: Random-number
spec:
  containers:
  - name: alpine
    image: alpine
    command: ['/bin/sh', '-c']
    arg: ['shuf -i 0-99 n1 >> /opt/number.out;']
    volumeMounts:
    - mountPath: /opt
      name: data_vol
  volumes:
  - name: data_vol
    emptyDir: {}
```

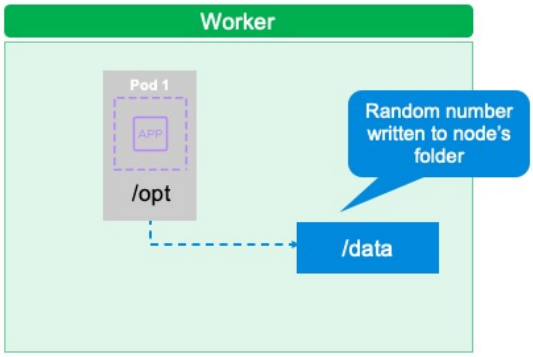
Data not persisted when another instance of the pod is created

Instructor Notes

Student Notes

Volumes in Kubernetes

Simple volume with HostPath



A node's directory volume example:

```
apiVersion: v1
kind: Pod
metadata:
  name: Random-number
spec:
  containers:
  - name: alpine
    image: alpine
    command: ['/bin/sh', '-c']
    arg: ['shuf -i 0-99 n1 >> /opt/number.out;']
    volumeMounts:
    - mountPath: /opt
      name: data_vol
  volumes:
  - name: data_vol
    hostPath:
      path: /data
      type: Directory
```

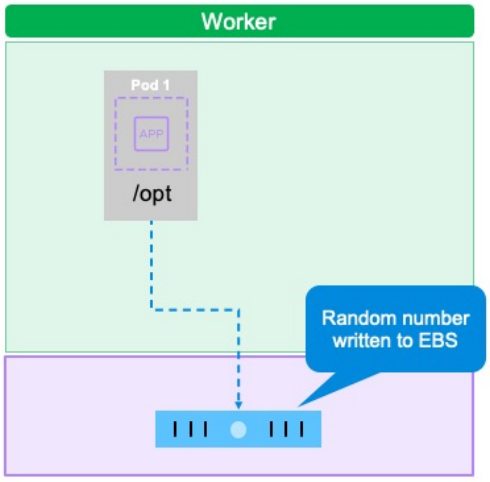
Data not available to all nodes in a multinode cluster

Instructor Notes

Student Notes

Volumes in Kubernetes

Simple volume with Amazon Elastic Block Store



Amazon Elastic Block Store (Amazon EBS) example:

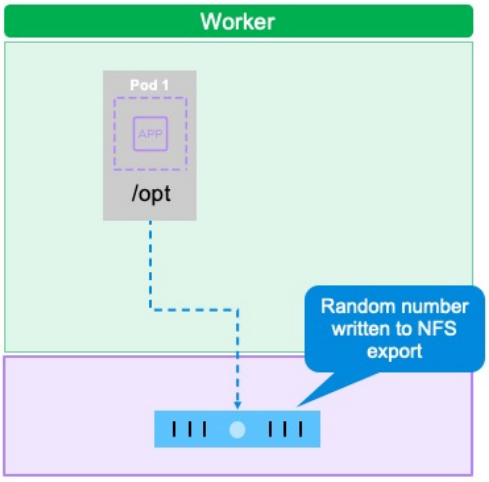
```
apiVersion: v1
kind: Pod
metadata:
  name: Random-number
spec:
  containers:
  - name: alpine
    image: alpine
    command: ['/bin/sh', '-c']
    arg: ['shuf -i 0-99 n1 >> /opt/number.out;']
    volumeMounts:
    - mountPath: /opt
      name: data_vol
  volumes:
  - name: data_vol
    awsElasticBlockStore:
      volumeId: ….
      fsType: ext4
```


Instructor Notes

Student Notes

Volumes in Kubernetes

Simple volume with NFS



NFS example:

```
apiVersion: v1
kind: Pod
metadata:
  name: Random-number
spec:
  containers:
  - name: alpine
    image: alpine
    command: ['/bin/sh', '-c']
    arg: ['shuf -i 0-99 n1 >> /opt/number.out;']
    volumeMounts:
    - mountPath: /opt
      name: data_vol
  volumes:
  - name: data_vol
    nfs:
      server: 192.168.0.31
      path: /nfs
      readOnly: false
```

Instructor Notes

Student Notes

Volumes versus persistent volumes

Overview

Kubernetes volumes

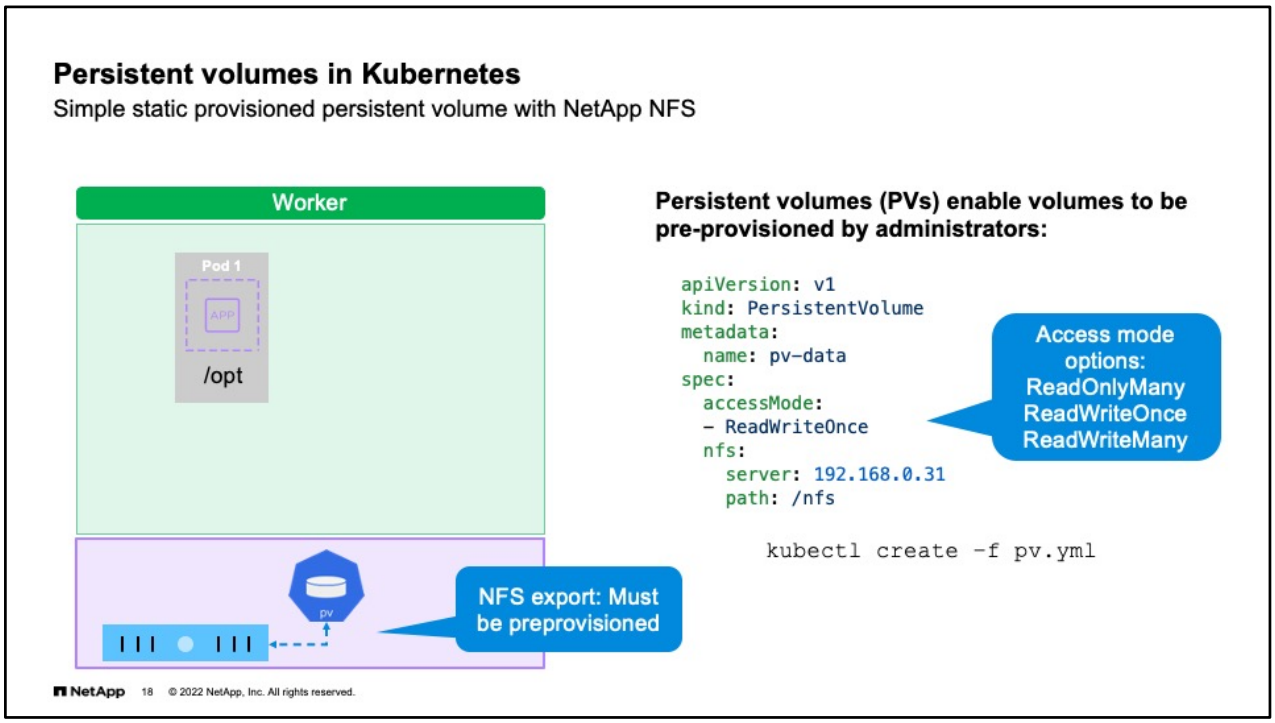
- Are tightly coupled with a pod definition
- Out live any container that is running in a pod (preserved across container restarts)
- Live for the lifetime of the pod
- Have the following persistence:
  - Some volume types, like emptyDir volumes, deleted when a pod is removed
  - Some volume types, like Amazon EBS volumes and NFS, persist after the pod removal

Persistent volumes (PV)

- Are loosely coupled with a pod definition
- Enable storage administrators to preprovision the storage for pods
- Are used by pods through a persistent volume claim (PVC)

Instructor Notes

Student Notes



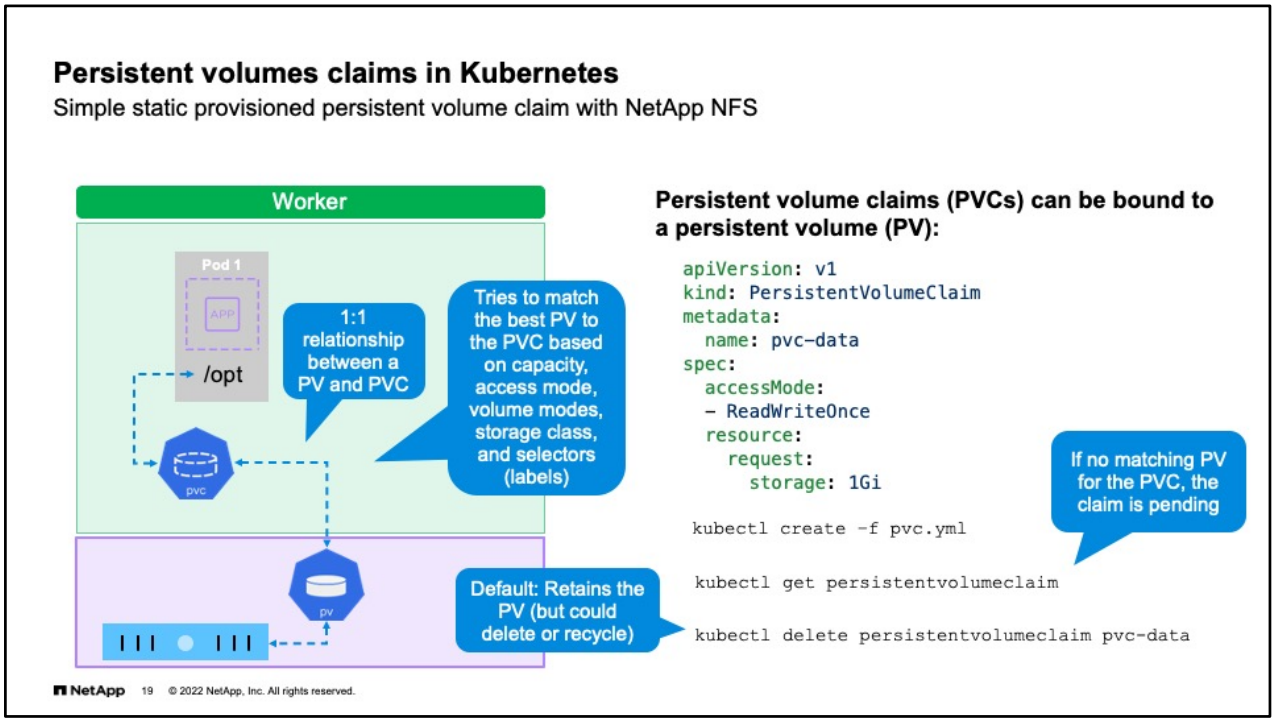
Instructor Notes

Student Notes

The access modes:

- ReadWriteOnce: The volume can be mounted as read/write by a single node.
- ReadOnlyMany: The volume can be mounted read-only by many pods on different nodes.
- ReadWriteMany: The volume can be mounted as read/write by many pods on different nodes.

For more information, see https://kubernetes.io/docs/concepts/storage/persistent-volumes/.



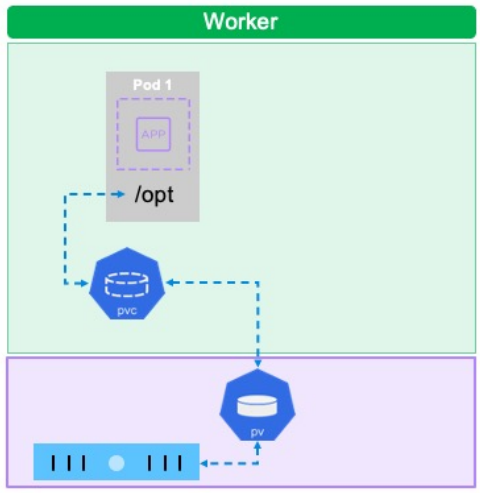
Instructor Notes

Student Notes

If you delete a persistent volume claim, you have three choices what to do with the persistent volume. Use the `persistentVolumeReclaimPolicy` to retain, delete, or recycle (delete the data and recreate) the persistent volume.

Persistent volumes claims in Kubernetes

Using a simple static provisioned persistent volume claim with NetApp NFS



A pod can be assigned a persistent volume claim:

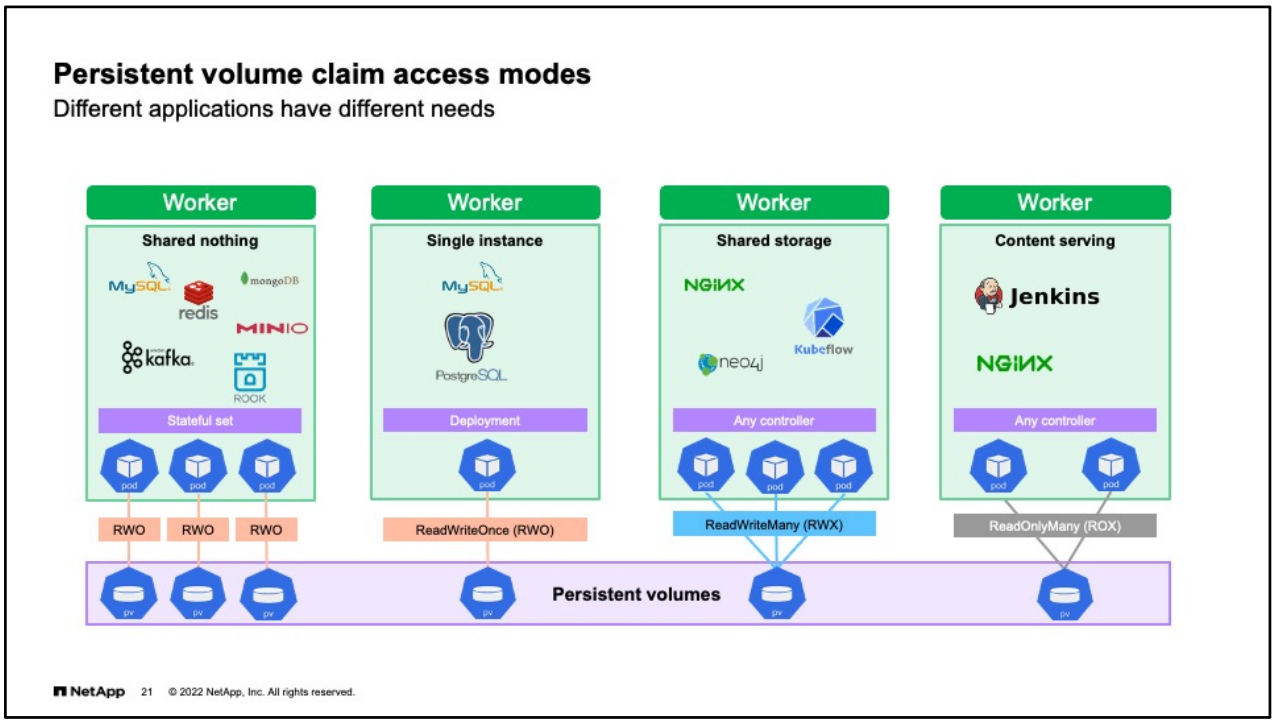
```
apiVersion: v1
kind: Pod
metadata:
  name: Random-number
spec:
  containers:
  - name: alpine
    image: alpine
    command: ['/bin/sh', '-c']
    arg: ['shuf -i 0-99 n1 >> /opt/number.out;']
    volumeMounts:
    - mountPath: /opt
      name: data_vol
  volumes:
  - name: data_vol
    persistentVolumeClaim:
      claimName: pvc-data
```

If accessMode is "many," multiple pods can use the claim

Can add claims to pod templates of ReplicaSets or deployments

Instructor Notes

Student Notes



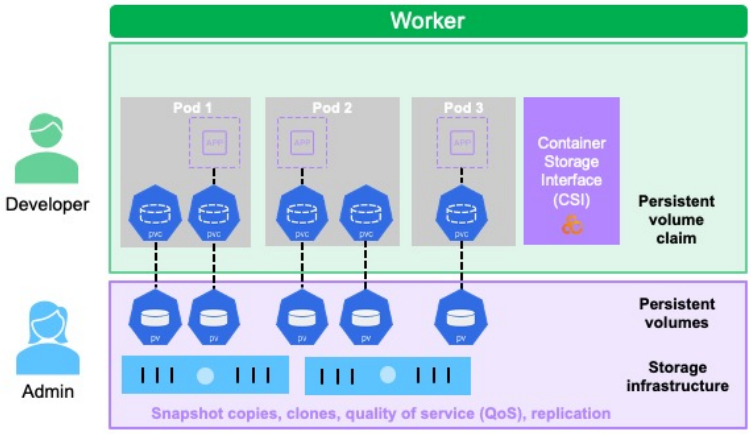
Instructor Notes

Student Notes

A volume can only be mounted by using one access mode at a time, even if it supports many. For example, a GCEPersistentDisk can be mounted as ReadWriteOnce by a single node or ReadOnlyMany by many nodes, but not both access modes at the same time.

Persistent storage integration

Two different users and roles



Persistent volume claim (PVC): a request for storage by a developer

Persistent volume (PV): storage in a cluster that is provisioned by an administrator or automatically by code

Instructor Notes

Student Notes

Static provisioning summary

Preconfigured storage method

```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-data
spec:
  accessMode:
  - ReadWriteOnce
  nfs:
    server: 192.168.0.31
    path: /nfs
```

```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-data
spec:
  accessMode:
  - ReadWriteOnce
  resource:
    request:
      storage: 1Gi
```

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: Random-number
spec:
  containers:
  - name: alpine
    image: alpine
    volumeMounts:
    - mountPath: /opt
      name: data_vol
  volumes:
  - name: data_vol
    persistentVolumeClaim:
      claimName: pvc-data
```

Instructor Notes

Student Notes

Instructor Notes

Student Notes

Storage classes

Dynamically provisioning storage when requested

Dynamic provisioning:

- Uses a storage class to have the storage created when a persistent volume claim is requested
- No need to pre-create a PV
- PV created when a PVC is requested

Astra Trident is the provisioner for NetApp:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: sc-trident
provisioner: csi.trident.netapp.io
parameters:
  <Additional Trident Parameters>
```


Instructor Notes

Student Notes

Dynamic provisioning basics

Provisioning storage by using a storage class

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: sc-trident
provisioner: csi.trident.netapp.io
parameters:
  <Additional Trident Parameters>
```

Replace the PV with a storage class

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-data
spec:
  accessMode:
  - ReadWriteOnce
  storageClassName: sc-trident
  resource:
    request:
      storage: 1Gi
```

Specify the storage class to be used

```
apiVersion: v1
kind: Pod
metadata:
  name: Random-number
spec:
  containers:
  - name: alpine
    image: alpine
    volumeMounts:
    - mountPath: /opt
      name: data_vol
  volumes:
  - name: data_vol
    persistentVolumeClaim:
      claimName: pvc-data
```

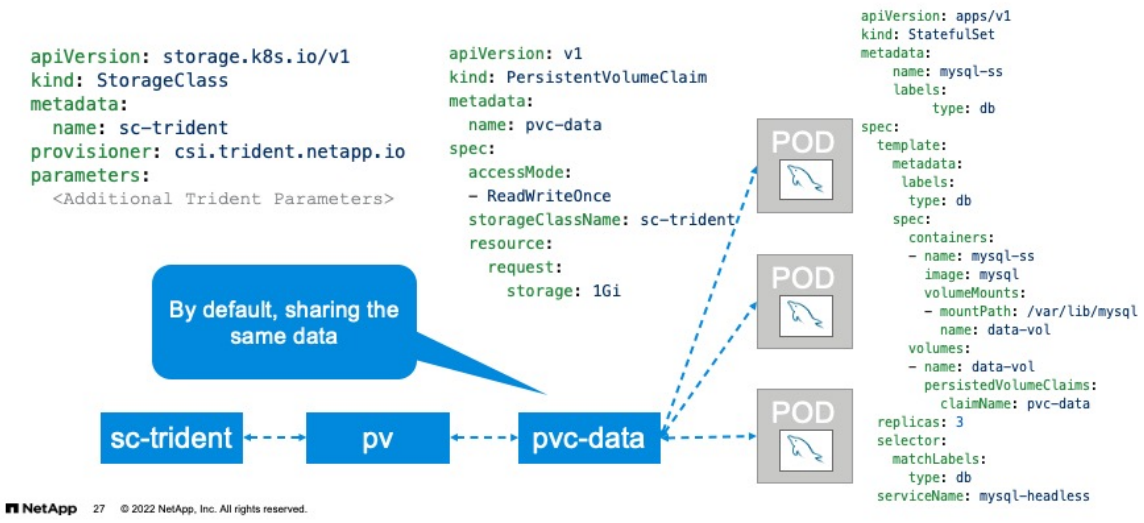
When requesting a PVC, the storage class creates a PV

Instructor Notes

Student Notes

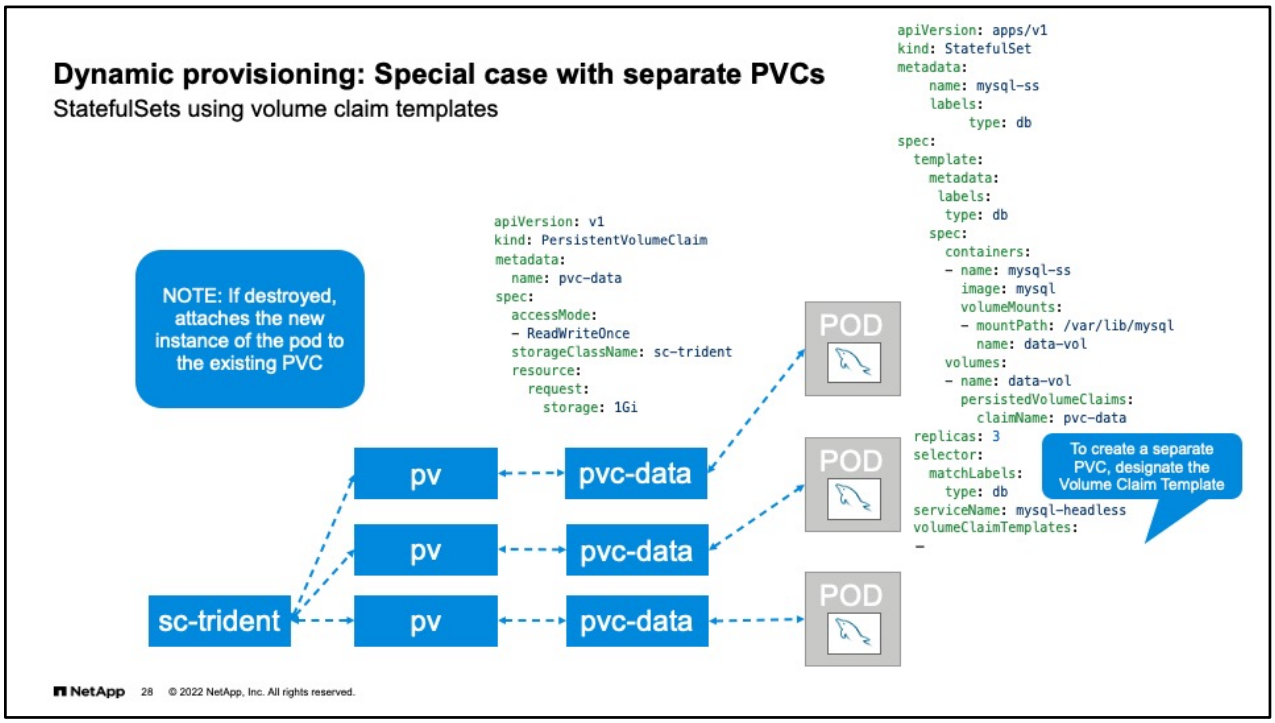
Dynamic provisioning: Special case

StatefulSets with shared storage



Instructor Notes

Student Notes



Instructor Notes

Student Notes

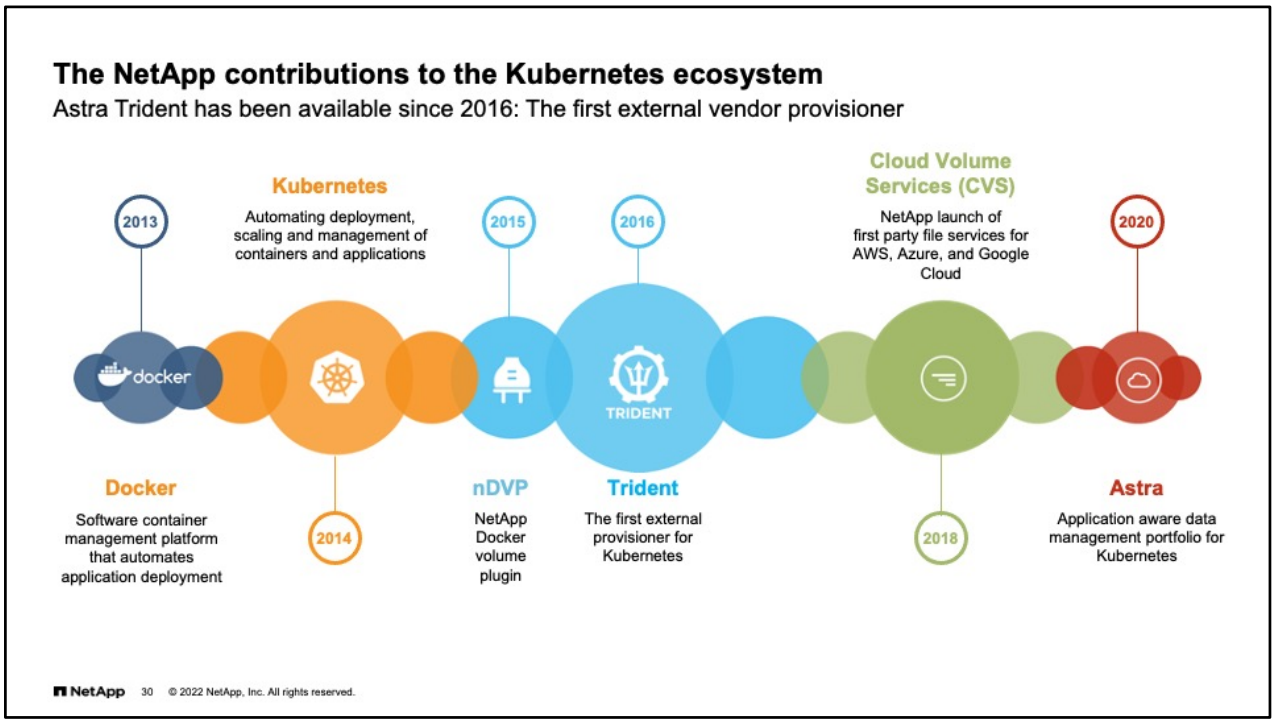
Astra Trident

The NetApp container storage provisioner

- Is a dynamic, automated storage provisioner
- Can abstract storage into pools of capabilities
  - Ability to differentiate storage
  - Ability for administrator to specify IOPS, compression, disk type, and so on
- Can map storage requests to storage pools
- Is open-source and fully supported:
  - https://github.com/netapp/trident
  - https://docs.netapp.com/us-en/trident/index.html

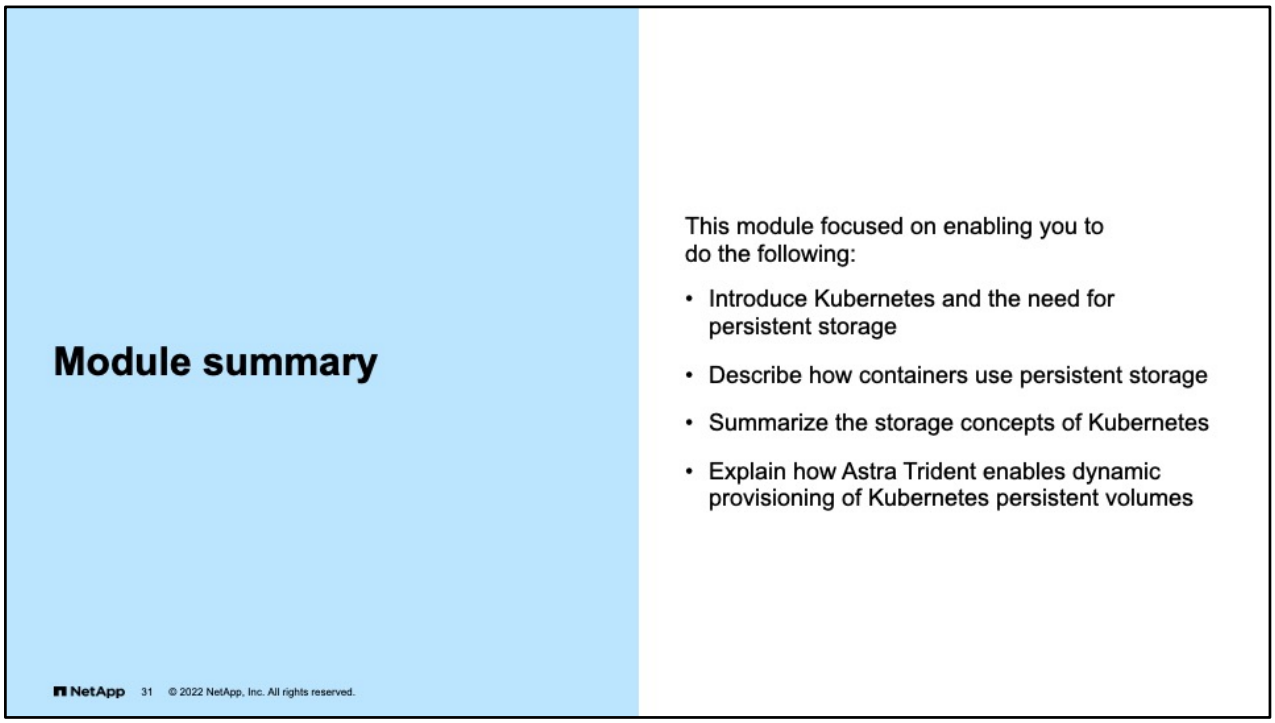
Instructor Notes

Student Notes



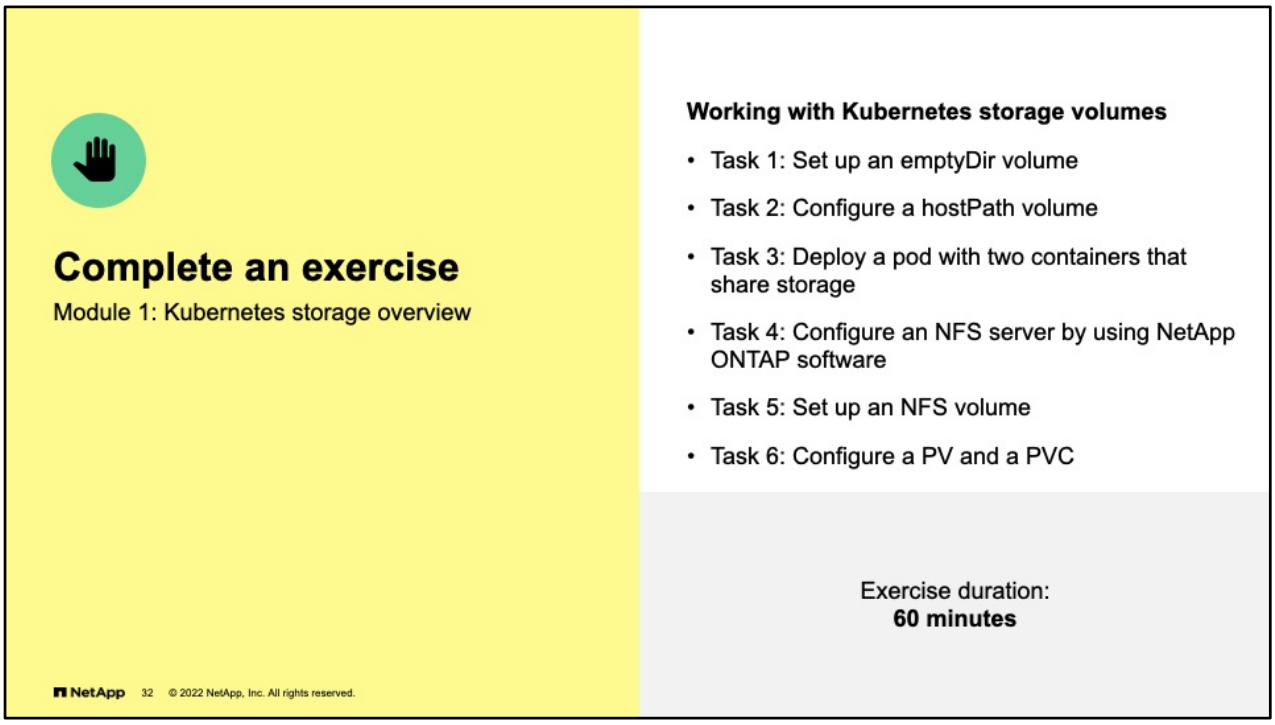
Instructor Notes

Student Notes



Instructor Notes

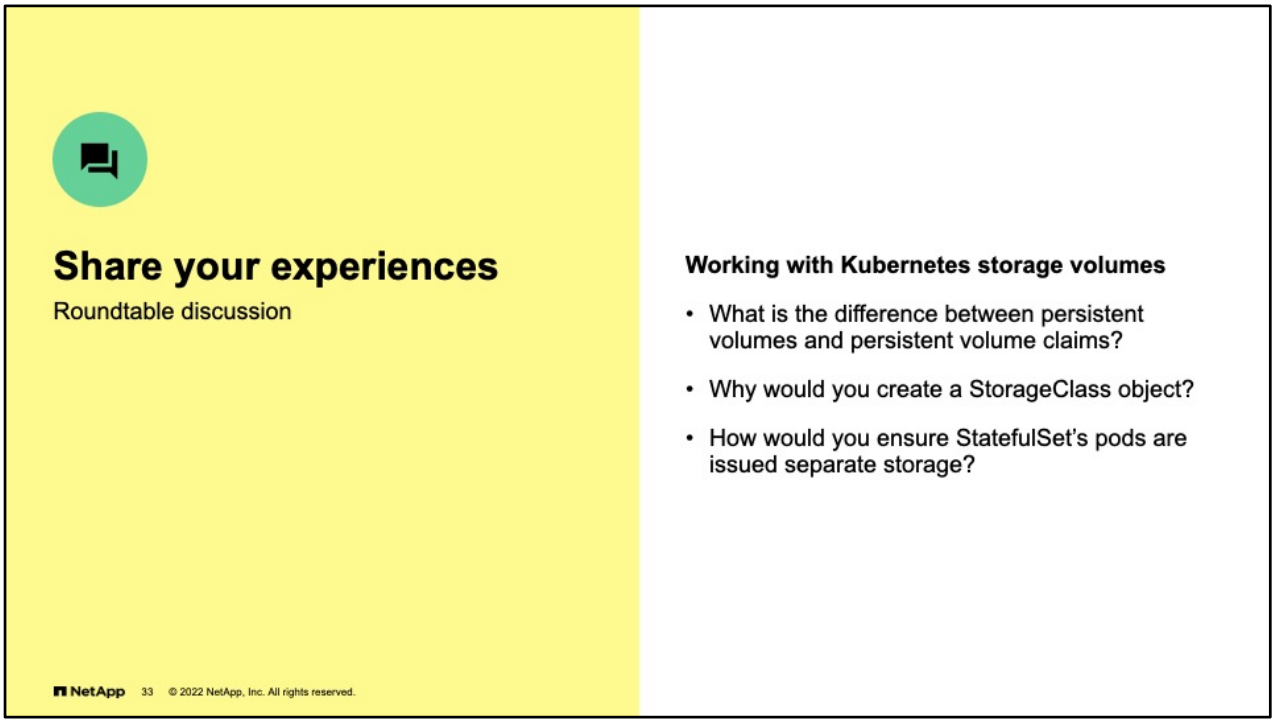
Student Notes



Instructor Notes

Student Notes

See the instructions in your Exercise Guide.



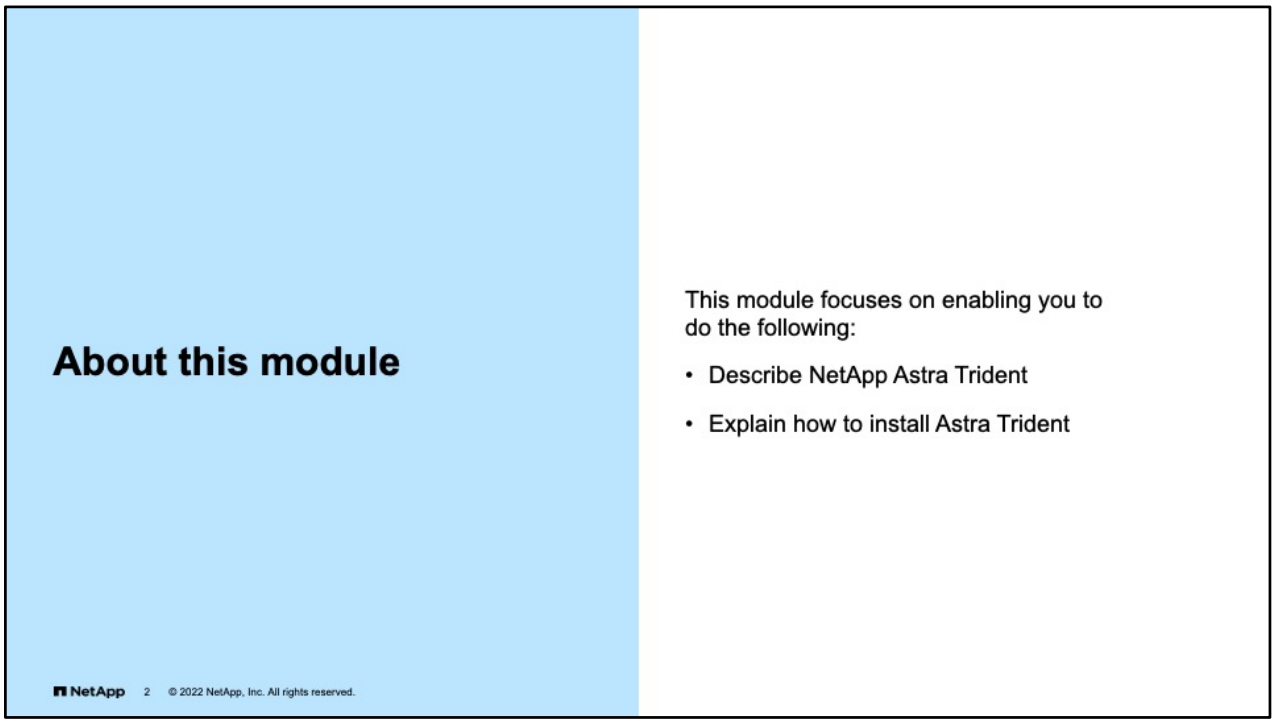
Instructor Notes
Display while students are completing the exercise(s)

Student Notes
Have a roundtable discussion with the class to answer these questions. Add any comments about experiences or “lessons learned” during the exercises that others might find helpful.

If you encounter an issue, notify your instructor immediately so that it can be resolved promptly.

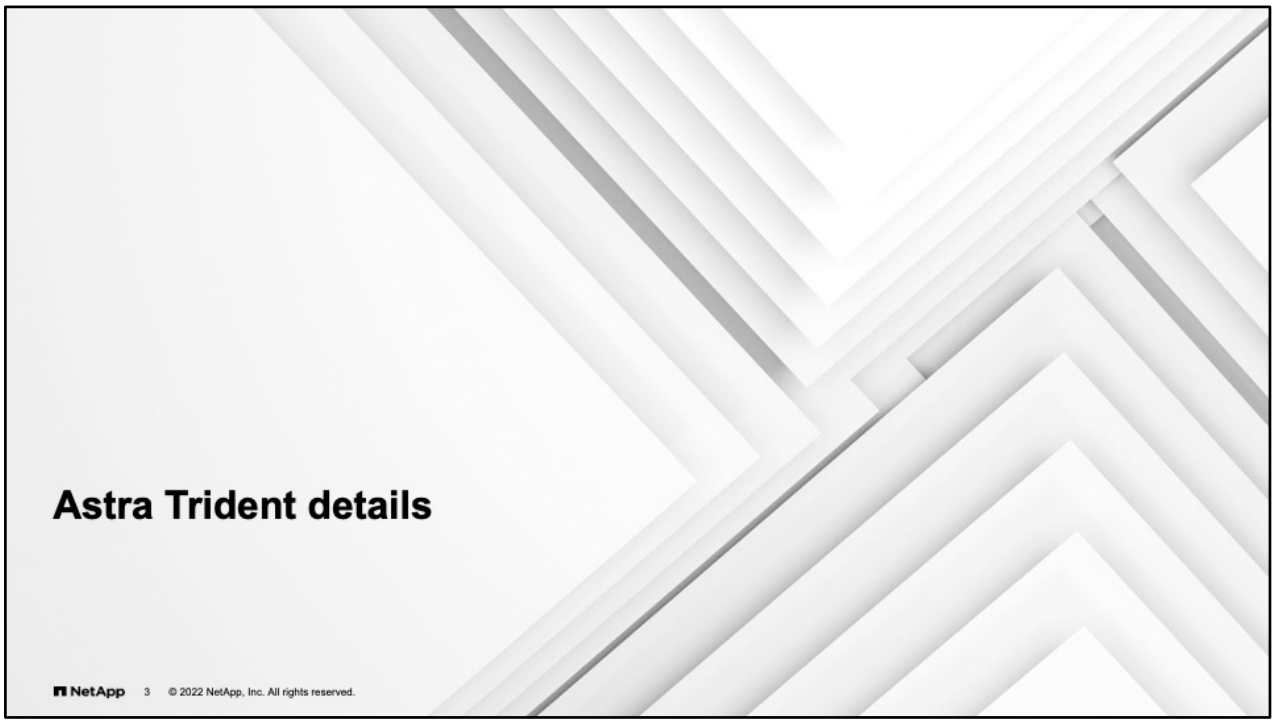
Instructor Notes

Student Notes



Instructor Notes

Student Notes

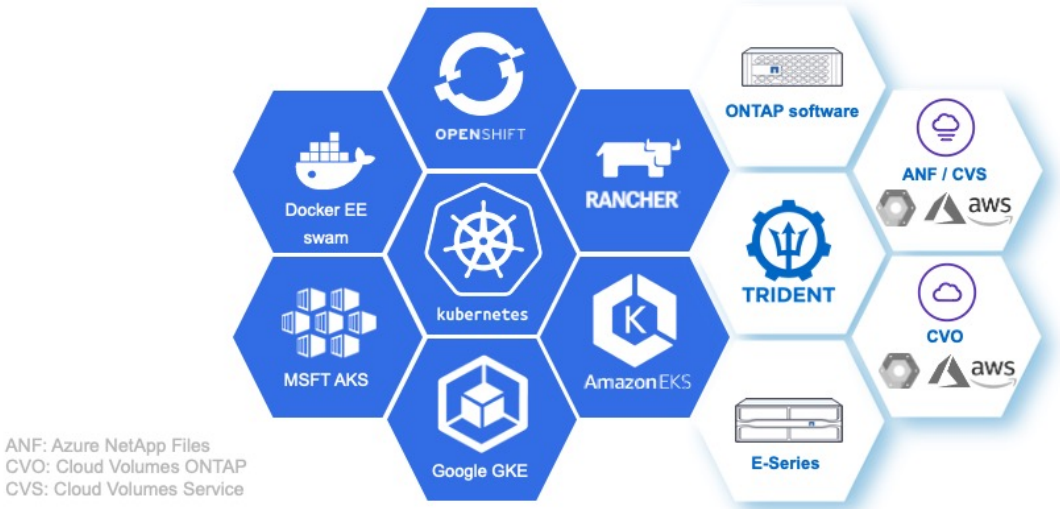


Instructor Notes

Student Notes

Astra Trident

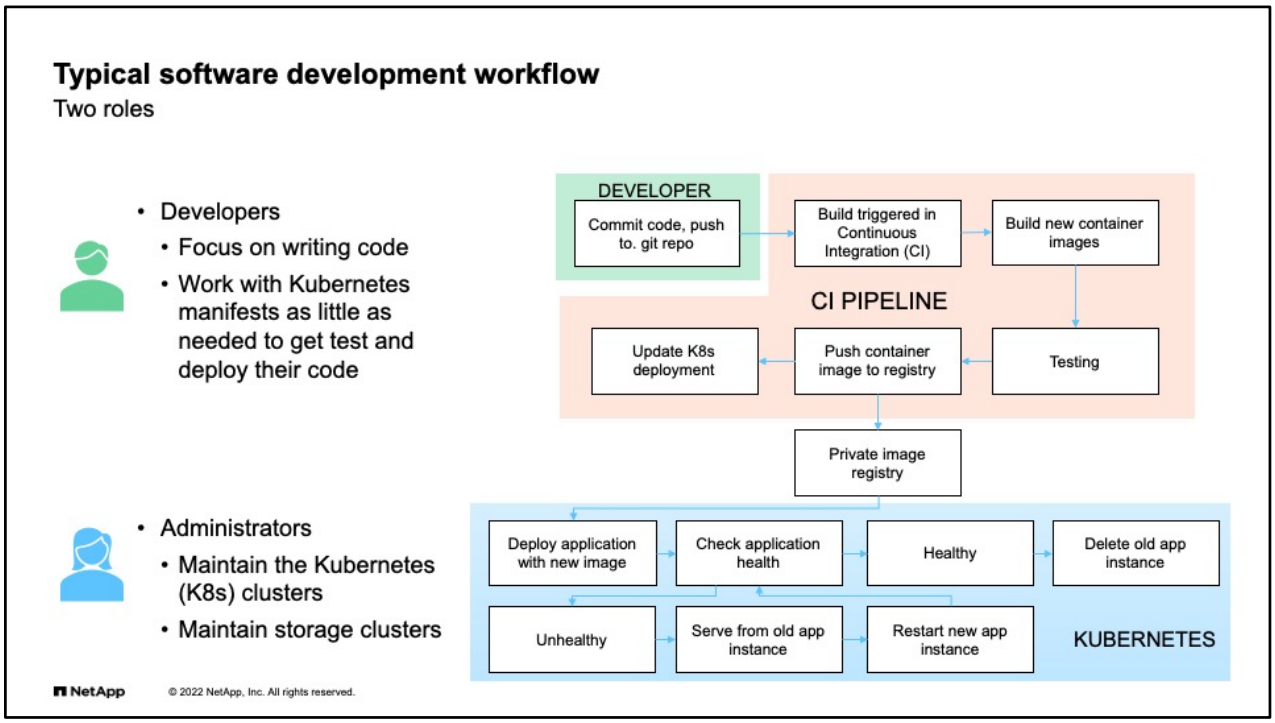
A cloud-native tool for NetApp storage



ANF: Azure NetApp Files
CVO: Cloud Volumes ONTAP
CVS: Cloud Volumes Service

Instructor Notes

Student Notes

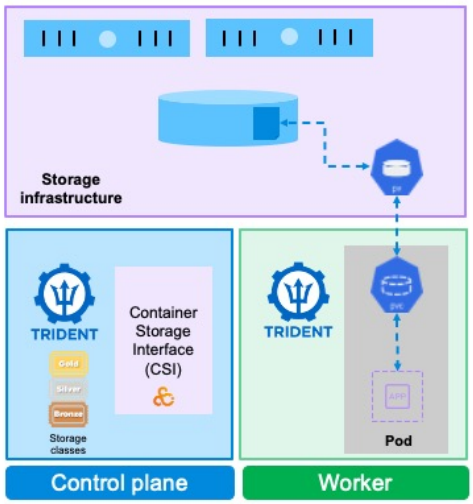


Instructor Notes

Student Notes

Astra Trident workflow

Creation of storage



Admin

1. Configures storage infrastructure
2. Installs Astra Trident in Kubernetes
3. Defines one or more back ends with virtual storage pool
4. Defines storage classes based on back ends

Developer

5. Creates a persistent volume claim (PVC) for a pod by using a storage class

TRIDENT

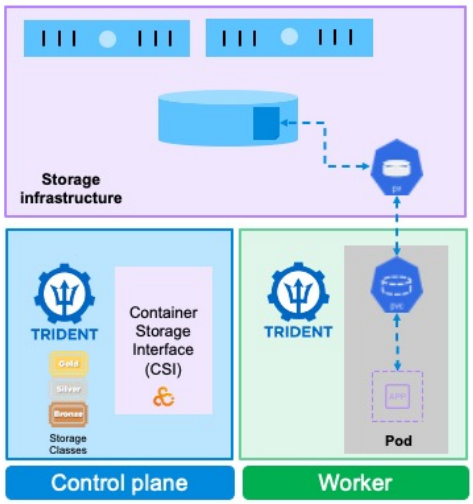
6. Detects creation of the PVC
7. Finds storage pool that satisfies the storage class
8. Creates a volume in an appropriate storage pool
9. Creates a persistent volume (PV) and binds the PVC with the PV
10. Mounts the volume with the container

Instructor Notes

Student Notes

Astra Trident workflow

Deletion of storage



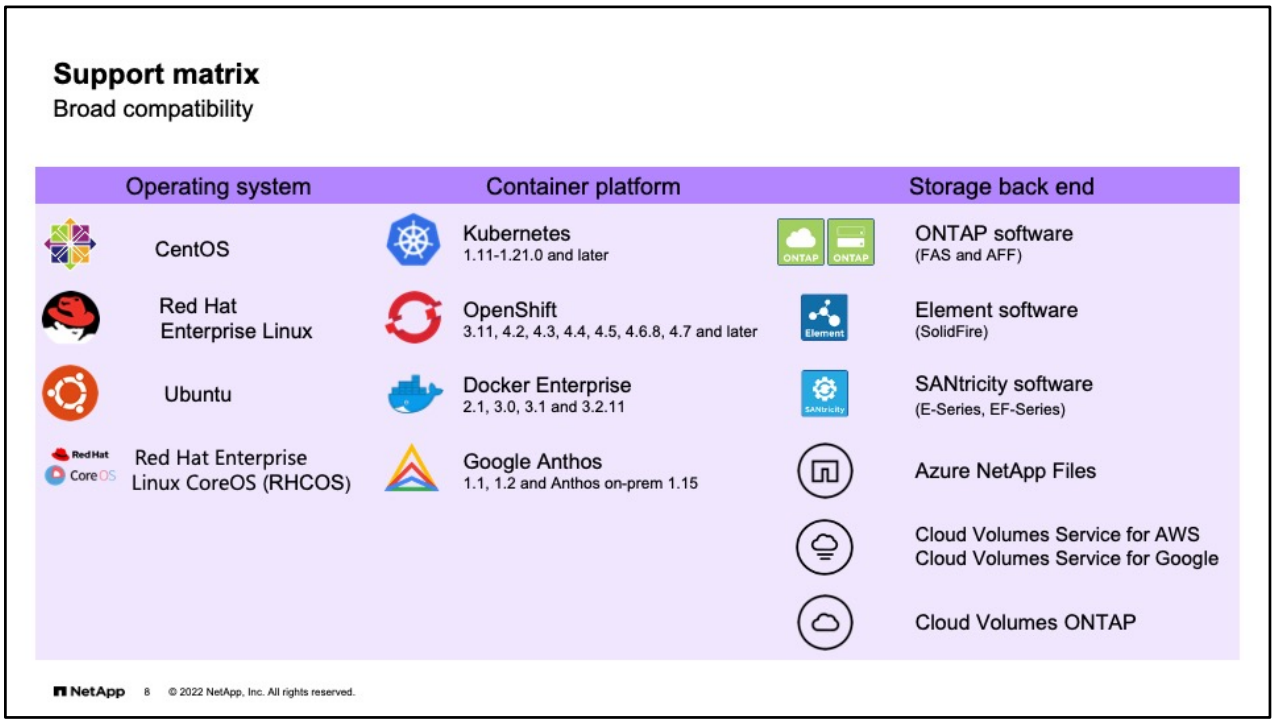

Admin


Developer

1. Deletes the PVC
2. Detects the deleted PVC
3. Finds and validates the matching PV
4. Determines the PV reclaim policy
5. Deletes the volume and the PV if the reclaim policy is `delete`

Instructor Notes

Student Notes



Instructor Notes

Student Notes

NOTE: Astra Trident should work with any distribution of Docker or Kubernetes that uses one of the supported versions as a base, such as Rancher and Tectonic.

Features

See https://github.com/NetApp/trident/releases for the latest

Feature	Astra Trident version	Kubernetes version	Feature gate required?
CSI Trident	19.07+	1.13+	Yes for 1.13
CSI Snapshots	20.01+	1.17+	No
PVC from CSI Snapshots	20.01+	1.17+	No
iSCSI PVC Resize	19.10+	1.16+	No
ONTAP bidirectional CHAP	20.04+	1.11+	No
Dynamic Export Policy	20.04+	1.13+	Requires CSI Astra Trident
Trident Operator	20.04+	1.14+	No
CSI Topology	20.10+	1.17	No
Auto-Worker Node Preparation (Beta)	20.10+	1.13+	Requires CSI Astra Trident

Instructor Notes

Student Notes

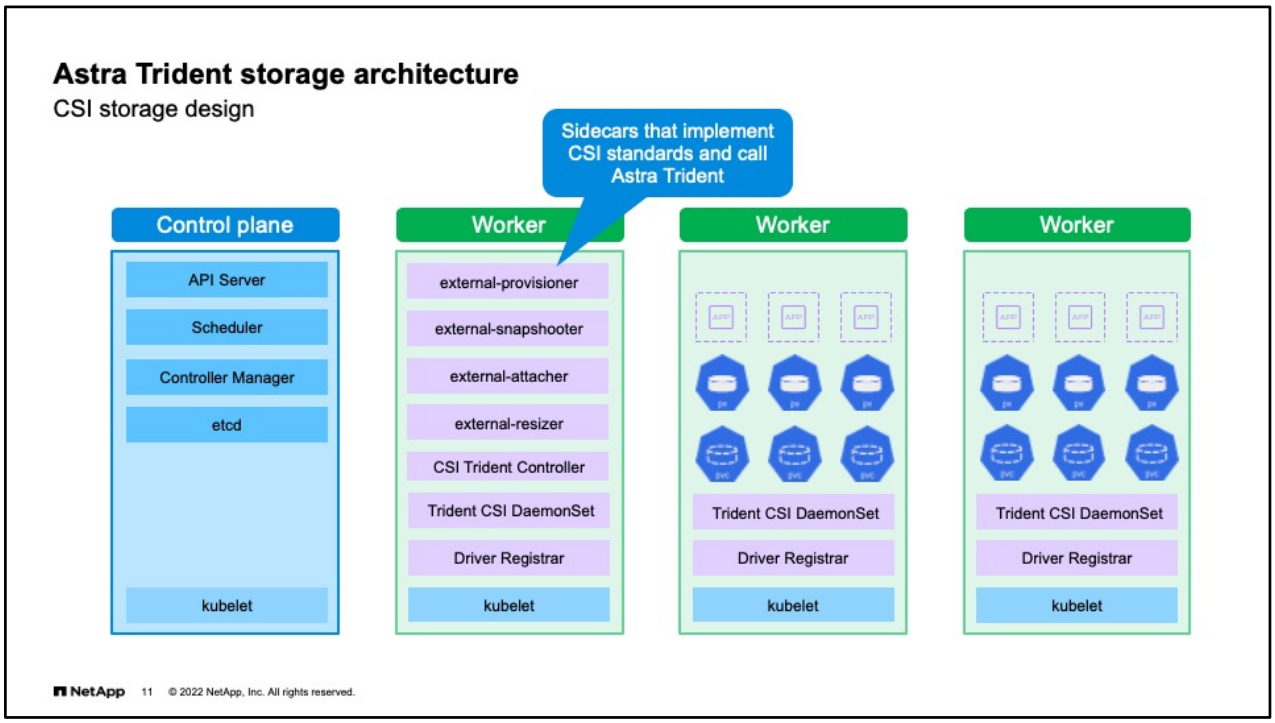
Astra Trident and ONTAP software

Protocols and access modes

Driver	Protocol	Volume mode	Access mode	File systems supported
ontap-nas	NFS	Filesystem	RWO,RWX,ROX	“ ”, nfs
ontap-nas-economy	NFS	Filesystem	RWO,RWX,ROX	“ ”, nfs
ontap-nas-flexgroup	NFS	Filesystem	RWO,RWX,ROX	“ ”, nfs
ontap-san	iSCSI	Block	RWO,ROX,RWX	No filesystem, raw block device
ontap-san-economy	iSCSI	Block	RWO,ROX,RWX	No filesystem, raw block device
ontap-san	iSCSI	Filesystem	RWO,ROX	xfs, ext3, ext4
ontap-san-economy	iSCSI	Filesystem	RWO,ROX	xfs, ext3, ext4

Instructor Notes

Student Notes



Instructor Notes

Student Notes

The CSI storage standard that Astra Trident implements has various containers that are used to help to implement the standard. Some of these containers include the following:

- external-provisioner is a sidecar container that watches the Kubernetes API server for PersistentVolumeClaim objects. It requests CreateVolume to the CSI Astra Trident Controller to provision a new volume in the back-end storage. For more information, see https://kubernetes-csi.github.io/docs/external-provisioner.html.
- external-snapshooter is a sidecar container that watches the Kubernetes API server for VolumeSnapshot and VolumeSnapshotContent custom resources. It requests CreateSnapshot, DeleteSnapshot, and ListSnapshot to the CSI Astra Trident Controller to manage Snapshot copies in the back-end storage. For more information, see https://kubernetes-csi.github.io/docs/external-snapshotter.html.
- external-attacher is a sidecar container that watches the Kubernetes API server for VolumeAttachment objects and requests CSI Astra Trident Controller to run various volume operations. For more information, see https://kubernetes-csi.github.io/docs/external-attacher.html.
- external-resizer is a sidecar container that watches the Kubernetes API server for PersistentVolumeClaims object edits and requests CSI Astra Trident Controller to run ControllerExpandVolume operations in the storage back end. For more information, see https://kubernetes-csi.github.io/docs/external-resizer.html.

Instructor Notes

Student Notes

Astra Trident installation

Binary location

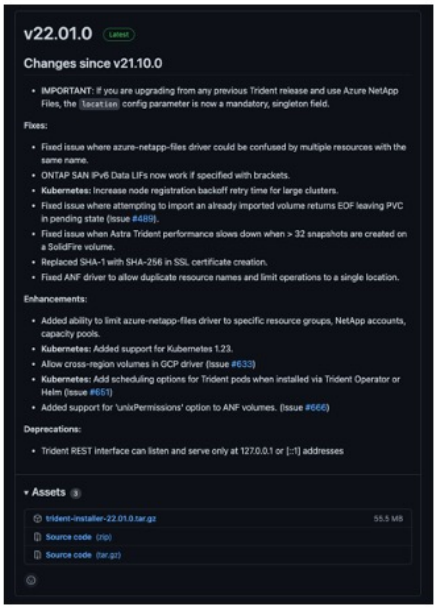
- Available at https://github.com/NetApp/trident/releases
  - Executable and sample YAML files
  - Source code
- Documented installation process at https://docs.netapp.com/us-en/trident/index.html
- Methods to install:
  - Use Binary (`tridentctl`)
  - Use Operator

    Introduced with Astra Trident 20.04 (approach used in the lab exercises)
  - Use a Helm chart

    Introduced with Astra Trident 21.01
  - Use GitOps

    Introduced with Astra Trident 21.04

Operator, Helm, and GitOps methods customizable



Instructor Notes

Student Notes

Astra Trident installation

Binary method overview

1. Create a namespace for Trident (`trident`)
2. Prepare worker nodes for NFS / iSCSI
3. Install trident: `./tridentctl install -n trident`
4. Verify `trident-csi-<generated id>` pod is running
5. Verify `./tridentctl -n trident version`

Instructor Notes

Student Notes

Astra Trident installation

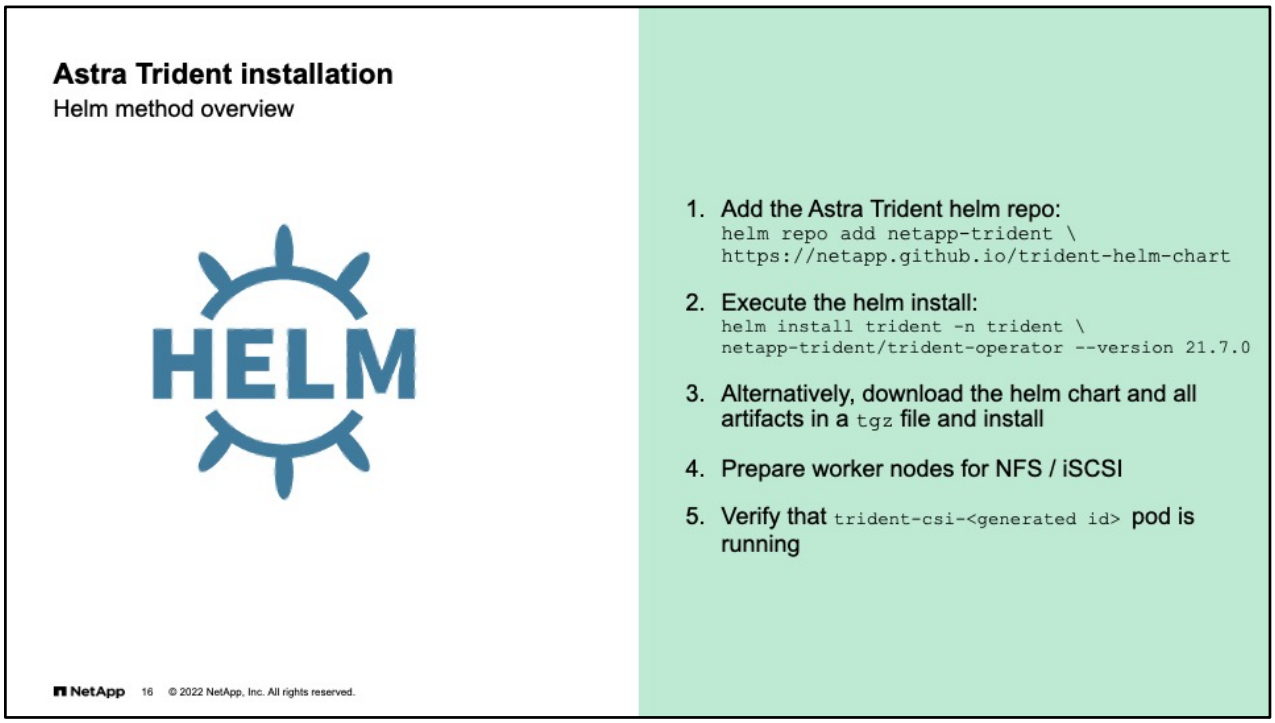
Operator method overview

1. Create the Custom Resource Definitions
2. Apply `deploy/bundle.yaml` from the tar.gz file
   - Creates namespace
   - Creates a service account and security bindings
   - Installs the operator
   - Sets up a pod security policy
3. Create a `TridentOrchestrator` custom resource

   Optionally, limit the nodes to deploy Astra Trident by using taints and tolerations or nodeSelectors
4. Prepare worker nodes for NFS / iSCSI
5. Verify that `trident-csi-<generated id>` pod is running

Instructor Notes

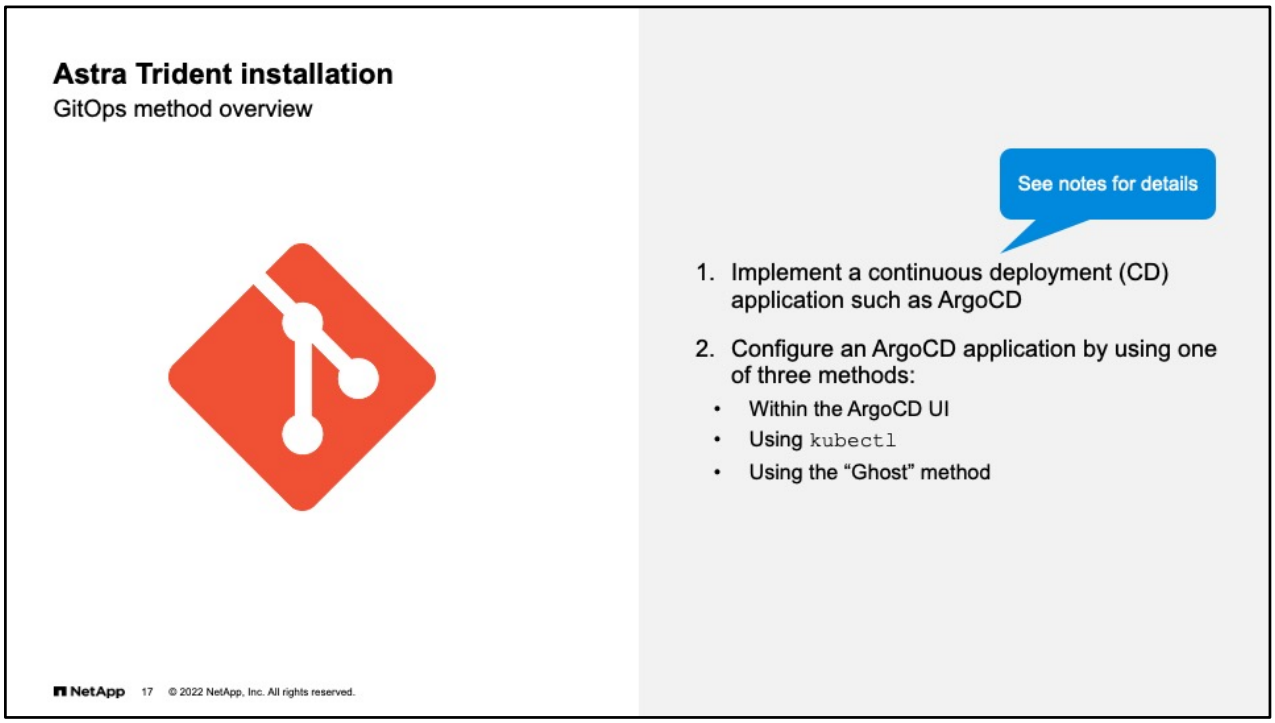
Student Notes



Instructor Notes

Student Notes

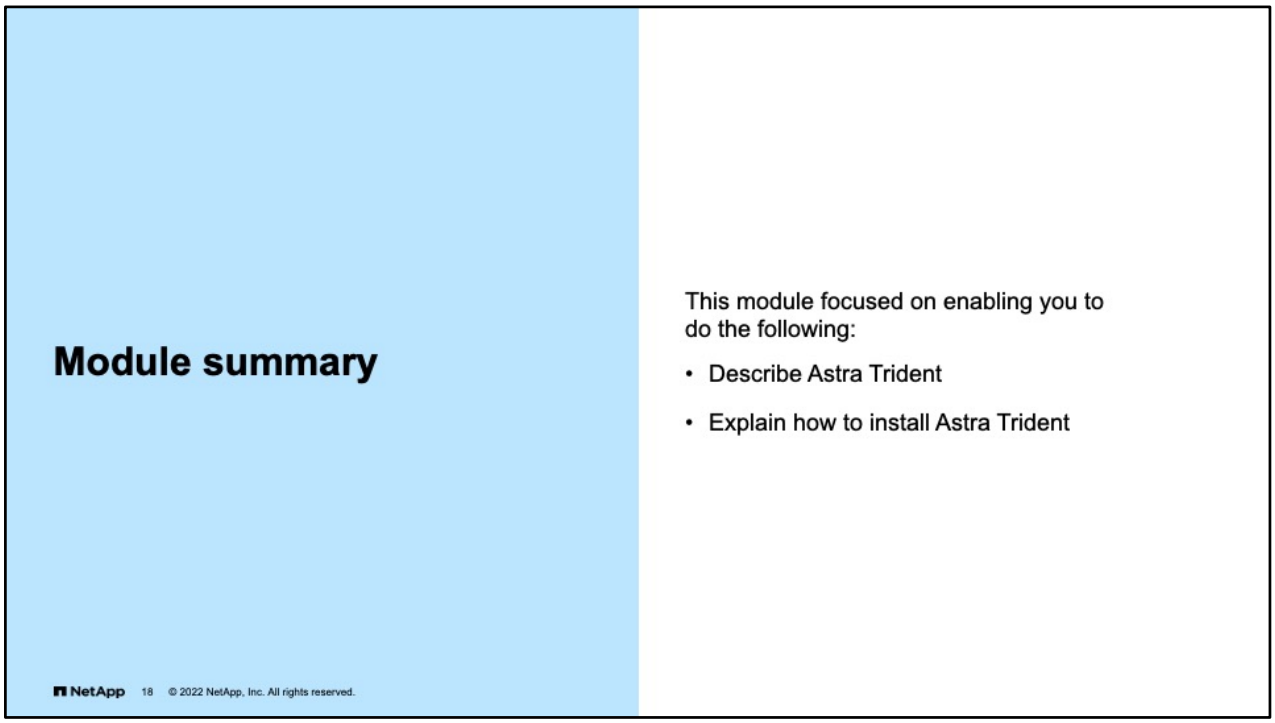
The Trident Helm chart can be installed directly from its public repository if the Kubernetes cluster has access to internet. Otherwise, the chart is also included in the Trident package that you can download from the NetApp GitHub page.



Instructor Notes

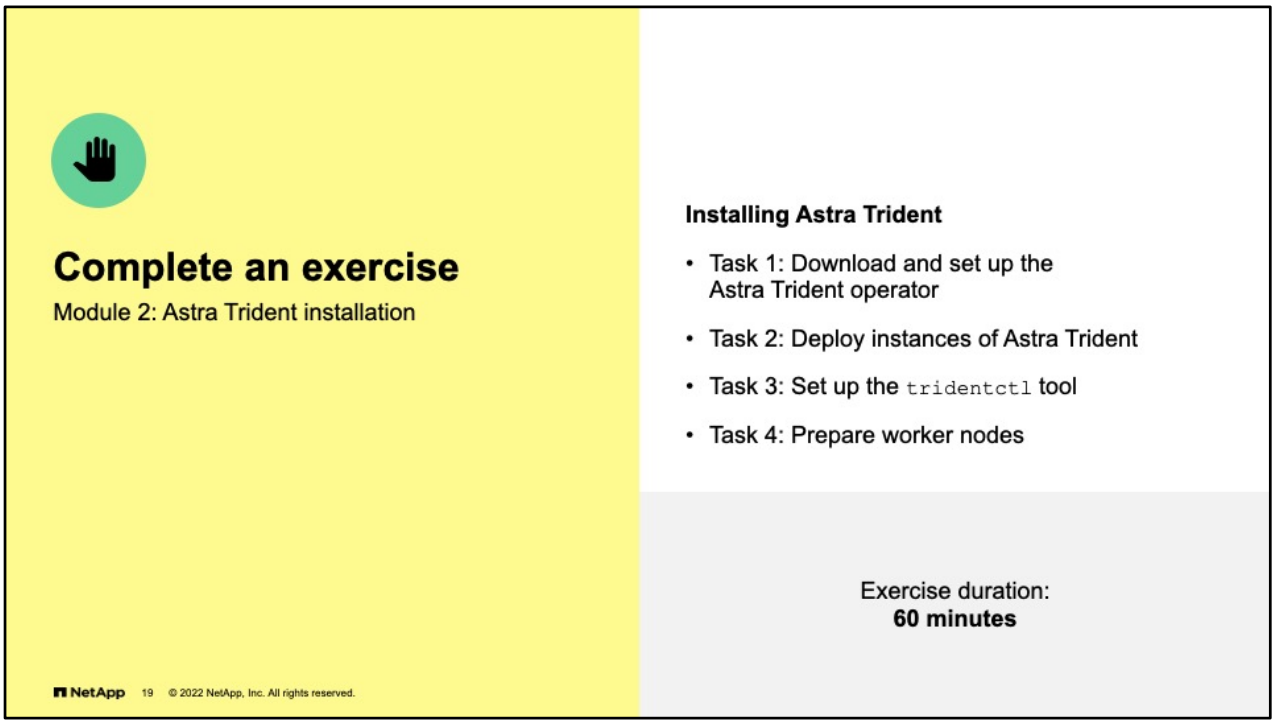
Student Notes

With Trident 21.04, the Kubernetes administrator does not need to manage Astra Trident outside of Kubernetes. Now that Trident back ends can be created with kubectl, the whole Astra Trident lifecycle can be managed through continuous deployment (CD) tools, such as ArgoCD. For more information, see https://github.com/YvosOnTheHub/LabNetApp/tree/master/Kubernetes_v4/Scenarios/Scenario18.



Instructor Notes

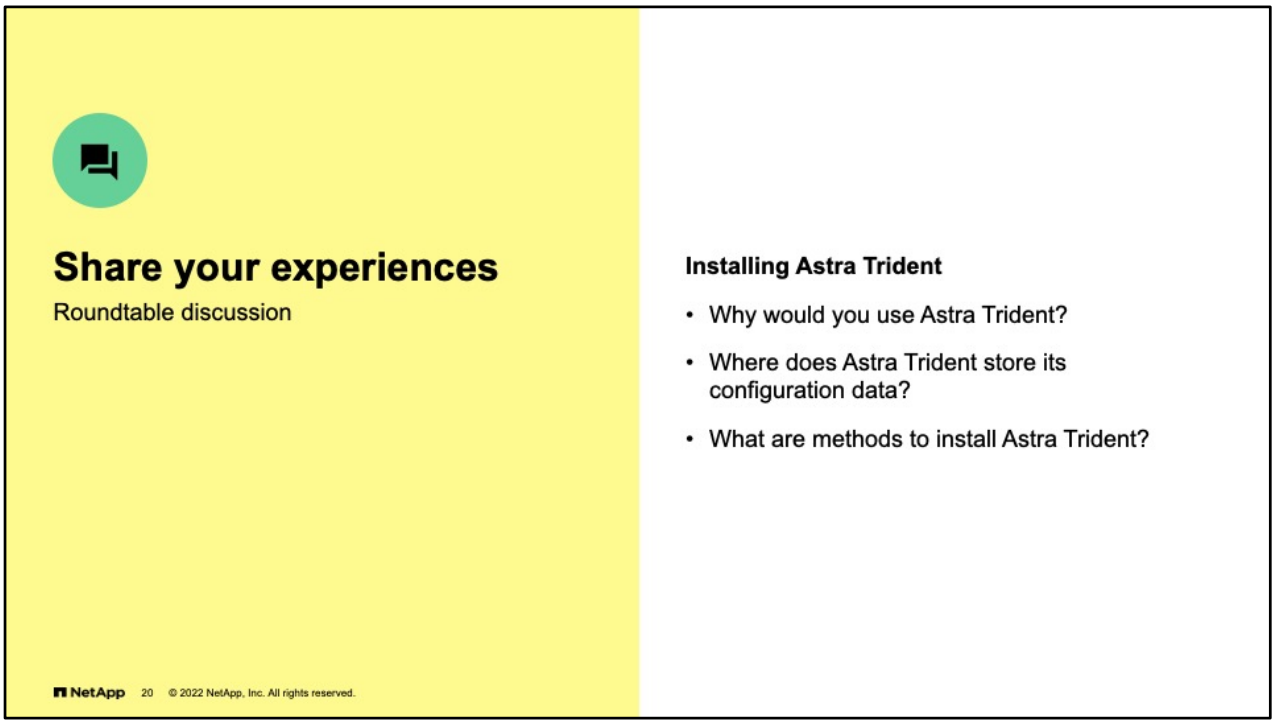
Student Notes



Instructor Notes

Student Notes

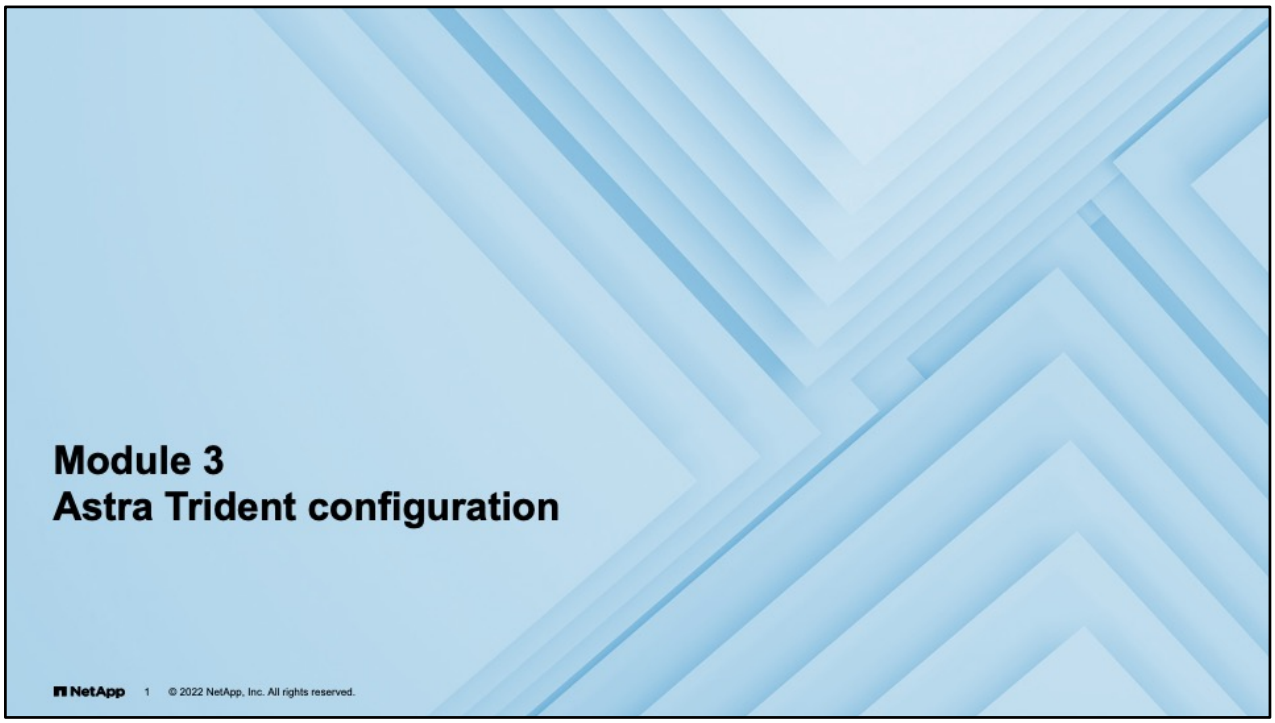
See the instructions in your Exercise Guide.



Instructor Notes
Display while students are completing the exercise(s)

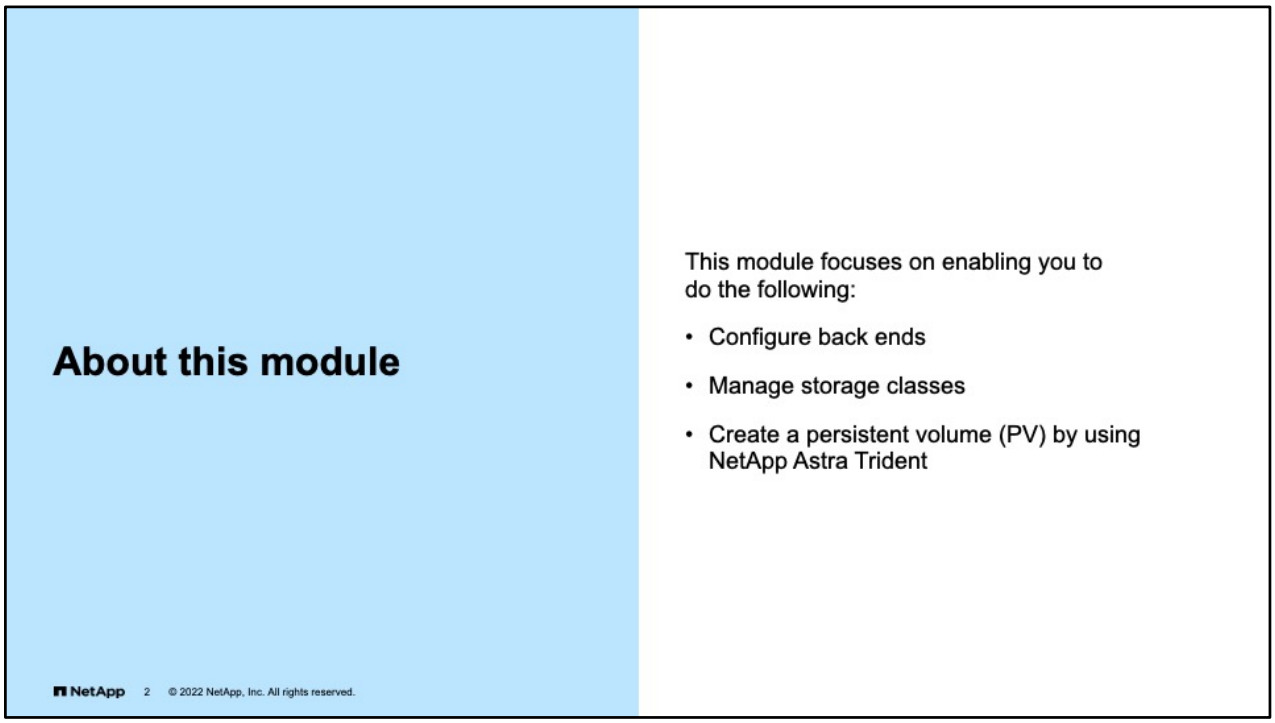
Student Notes
Have a roundtable discussion with the class to answer these questions. Add any comments about experiences or “lessons learned” during the exercises that others might find helpful.

If you encounter an issue, notify your instructor immediately so that it can be resolved promptly.



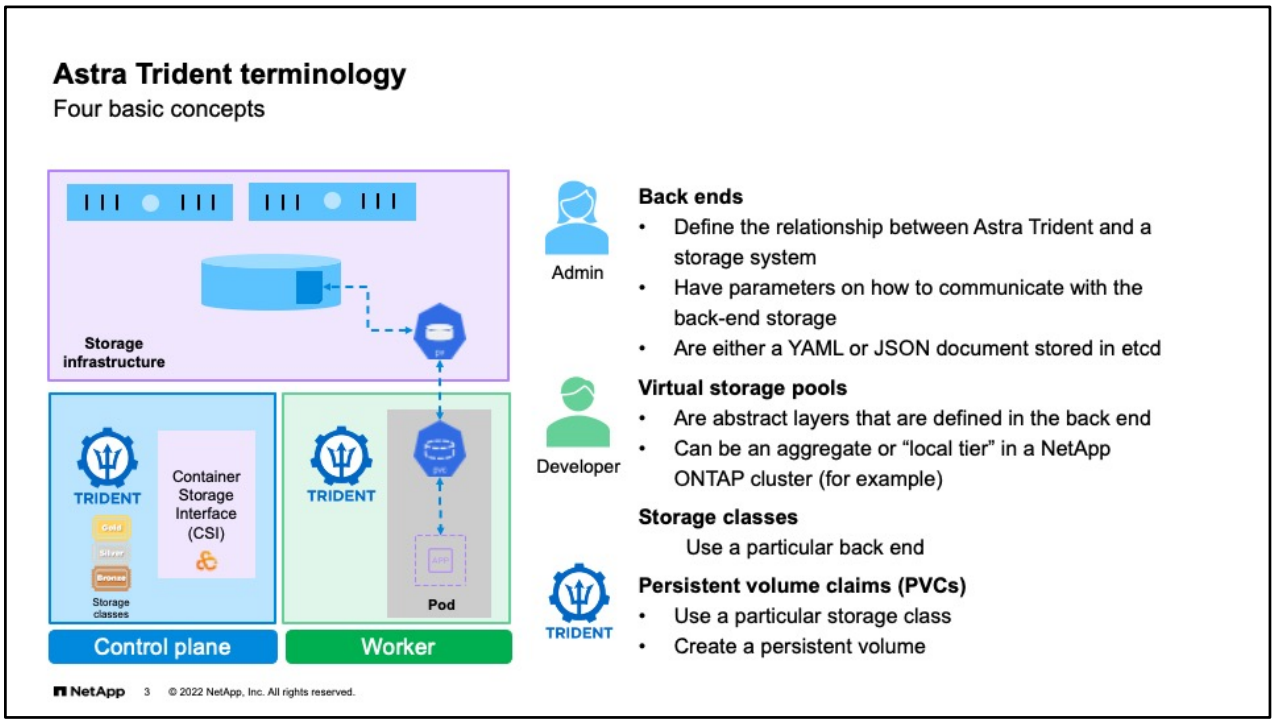
Instructor Notes

Student Notes



Instructor Notes

Student Notes



Instructor Notes

Student Notes

Back ends: Back ends represent the storage providers on top of which Astra Trident provisions volumes. A single Astra Trident instance can manage any number of back ends.

Storage pools: Storage pools represent the distinct locations available for provisioning on each back end. For NetApp ONTAP software, these storage pools correspond to aggregates in storage VMs (storage virtual machines, also known as SVMs). For NetApp SolidFire, these storage pools correspond to administrator-specified quality-of-service (QoS) bands. For NetApp Cloud Volumes Service, these storage pools correspond to cloud provider regions. Each storage pool has a set of distinct storage attributes, which define its performance characteristics and data protection characteristics.

Instructor Notes

Student Notes

Back ends

Overview

- Define the relationship between Astra Trident and a storage system
- Can be created by using one of the following:
  - A JSON and the `tridentctl` tool
  - A YAML definition file and the `kubectl` tool
- Are configured for various storage platforms, for example:
  - Azure NetApp Files
  - NetApp Cloud Volumes Service for AWS
  - NetApp Cloud Volumes Services for Google Cloud
  - Amazon FSx for NetApp ONTAP
  - NetApp Element software
  - NetApp ONTAP software or NetApp Cloud Volumes ONTAP
- Use only one storage driver at a time:
  - ontap-nas
  - ontap-nas-economy
  - ontap-nas-flexgroup
  - ontap-san
  - ontap-san-economy
  - ontap-san
  - ontap-san-economy
  - azure-netapp-files
  - aws-cvs
  - gcp-cvs
  - solidfire-san

Instructor Notes

Student Notes

Back-end creation

NFS example

NFS JSON example

```
{
    "version": 1,
    "storageDriverName": "ontap-nas",
    "backendName": "nas-backend",
    "managementLIF": "192.168.0.101",
    "dataLIF": "192.168.0.31",
    "svm": "svm0",
    "username": "admin",
    "password": "Netapp1!"
}
```

NFS YAML example

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: nas-backend
spec:
  version: 1
  backendName: ontap-nas-backend
  storageDriverName: ontap-nas
  managementLIF: 192.168.0.40
  dataLIF: 192.168.0.41
  svm: svm1
  credentials:
    name: nas-backend-secret
```

```
apiVersion: v1
kind: Secret
metadata:
  name: nas-backend-secret
type: kubernetes.io/basic-auth
stringData:
  username: vsadmin
  password: Netapp1!
```


Instructor Notes

Student Notes

Back-end creation

SAN example

SAN JSON example

```
{
    "version": 1,
    "storageDriverName": "ontap-san",
    "managementLIF": "10.0.0.1",
    "dataLIF": "10.0.0.2",
    "svm": "trident_svm",
    "username": "admin",
    "password": "password",
    "labels": {"k8scluster": "dev2", "backend": "dev2-sanbackend"},
    "storagePrefix": "alternate-trident",
    "igroupName": "custom",
    "debugTraceFlags": {"api":false, "method":true},
    "defaults": {
        "spaceReserve": "volume",
        "qosPolicy": "standard",
        "spaceAllocation": "false",
        "snapshotPolicy": "default",
        "snapshotReserve": "10"
    }
}
```

SAN YAML example

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: san-backend
spec:
  version: 1
  backendName: ontap-san-backend
  storageDriverName: ontap-san
  managementLIF: 192.168.0.40
  dataLIF: 192.168.0.41
  svm: svm1
  credentials:
    name: san-backend-secret
```


Instructor Notes

Student Notes

Back-end management

Examples

- Tridentctl approach
  - Create:
    ```
    tridentctl create backend -f <file> -n trident
    ```
  - Delete:
    ```
    tridentctl delete backend -f <file> -n trident
    ```
  - View:
    ```
    tridentctl get backend <backend> -o json -n trident
    ```
  - Update:
    ```
    tridentctl update backend <backend> -f <file> -n trident
    ```
  - Identify the storage class that uses a back end:
    ```
    tridentctl get backend -o json | jq '[.items[] | {backend:
    .name, storageClasses: [.storage[].storageClasses]|unique}]'
    ```

Requires JQuery tool

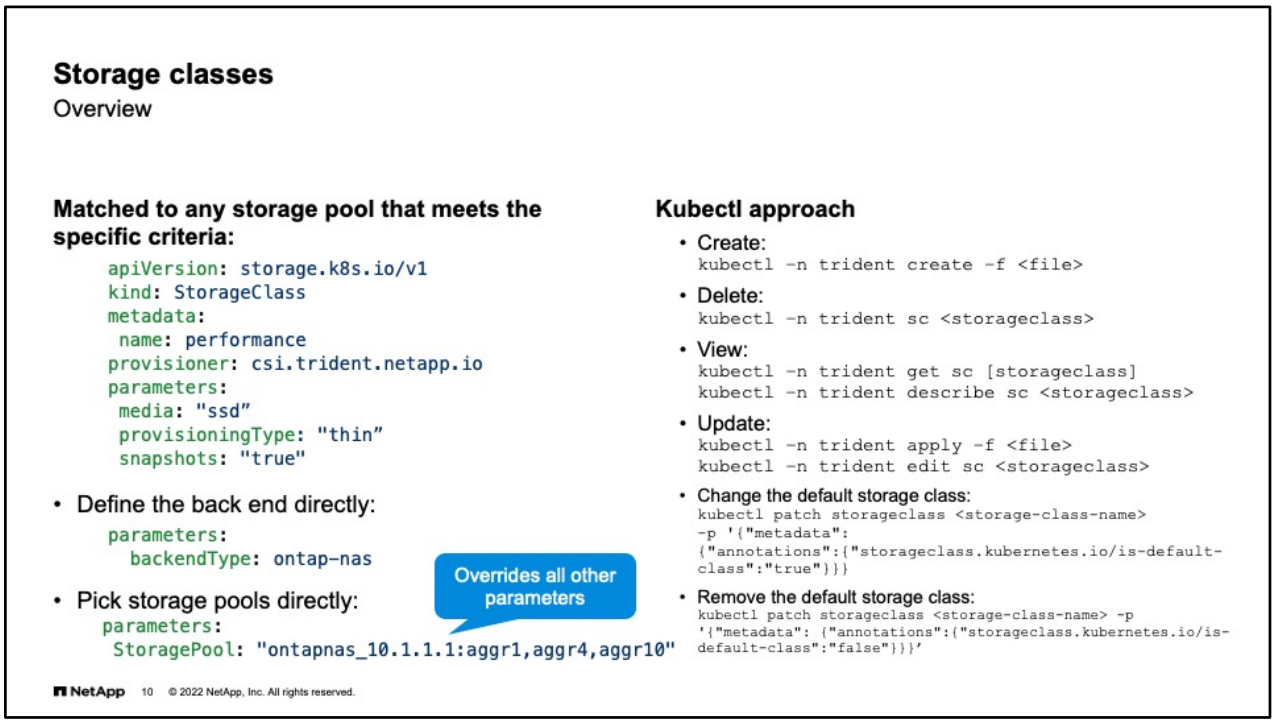
- Kubectl approach
  - Create:
    ```
    kubectl -n trident create -f <file>
    ```
  - Delete:
    ```
    kubectl -n trident tbc <backend>
    ```
  - View:
    ```
    kubectl -n trident get tbc [backend]
    kubectl -n trident describe tbc <backend>
    ```
  - Update:
    ```
    kubectl -n trident apply -f <file>
    kubectl -n trident edit tbc <backend>
    ```


Instructor Notes

Student Notes

Instructor Notes

Student Notes

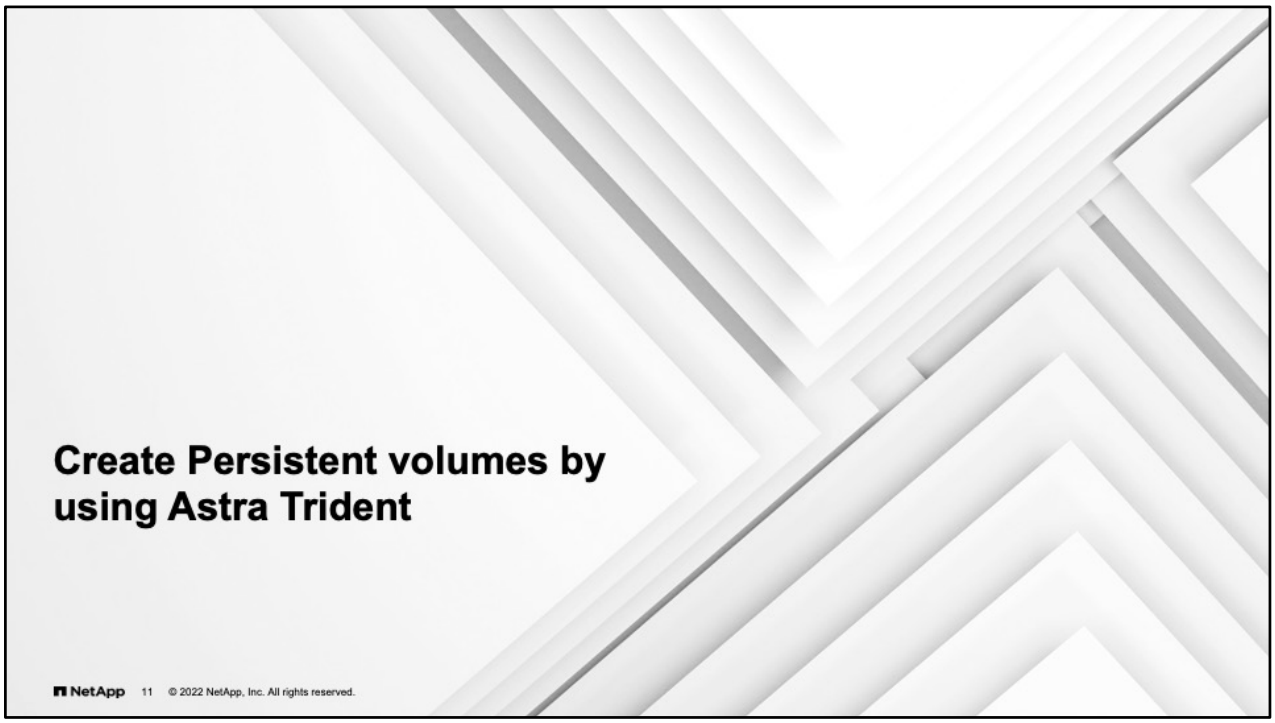


Instructor Notes

Student Notes

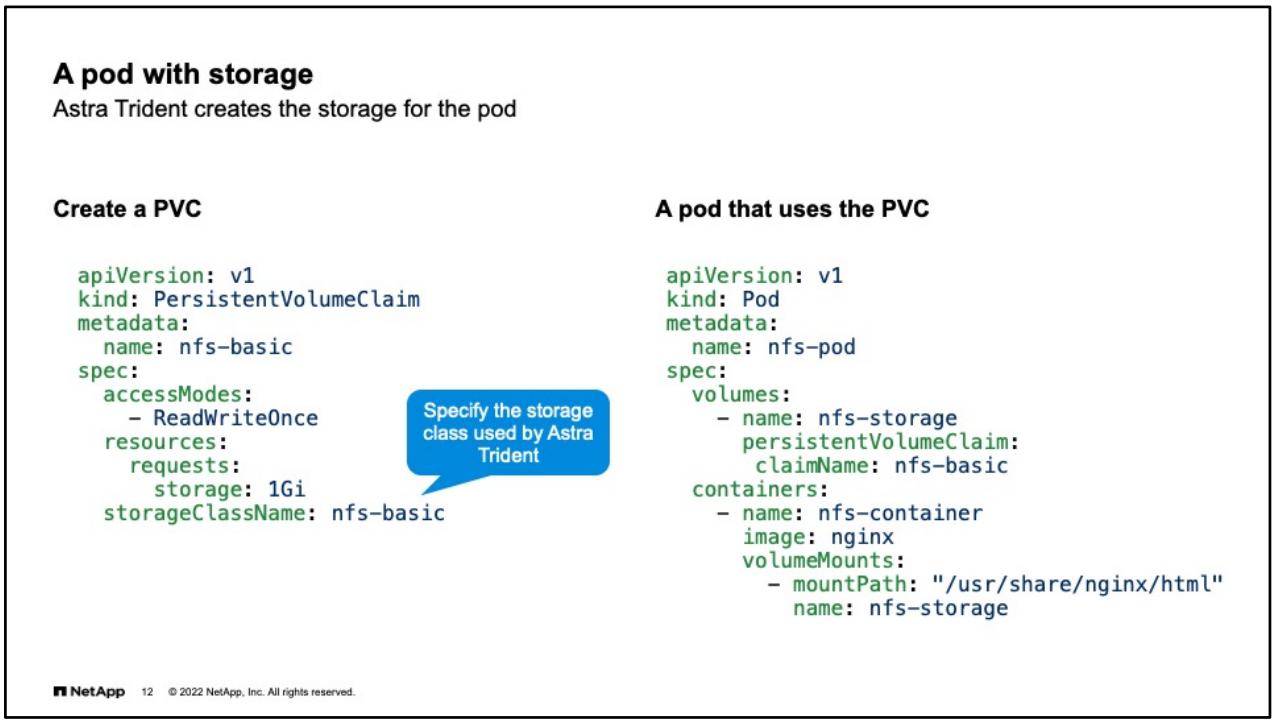
As of Kubernetes 1.16, you can specify a default storage class for all persistent volume classes. To make this selection, add the annotation storageclass.Kubernetes.io/is-default-class to true.

Astra Trident storage pools also offer the capability to select volume destinations based on properties such as back-end type, media type, provisioning type (thin versus thick), encryption, and so on. The supported choice capabilities vary according to the storage driver that is being used.



Instructor Notes

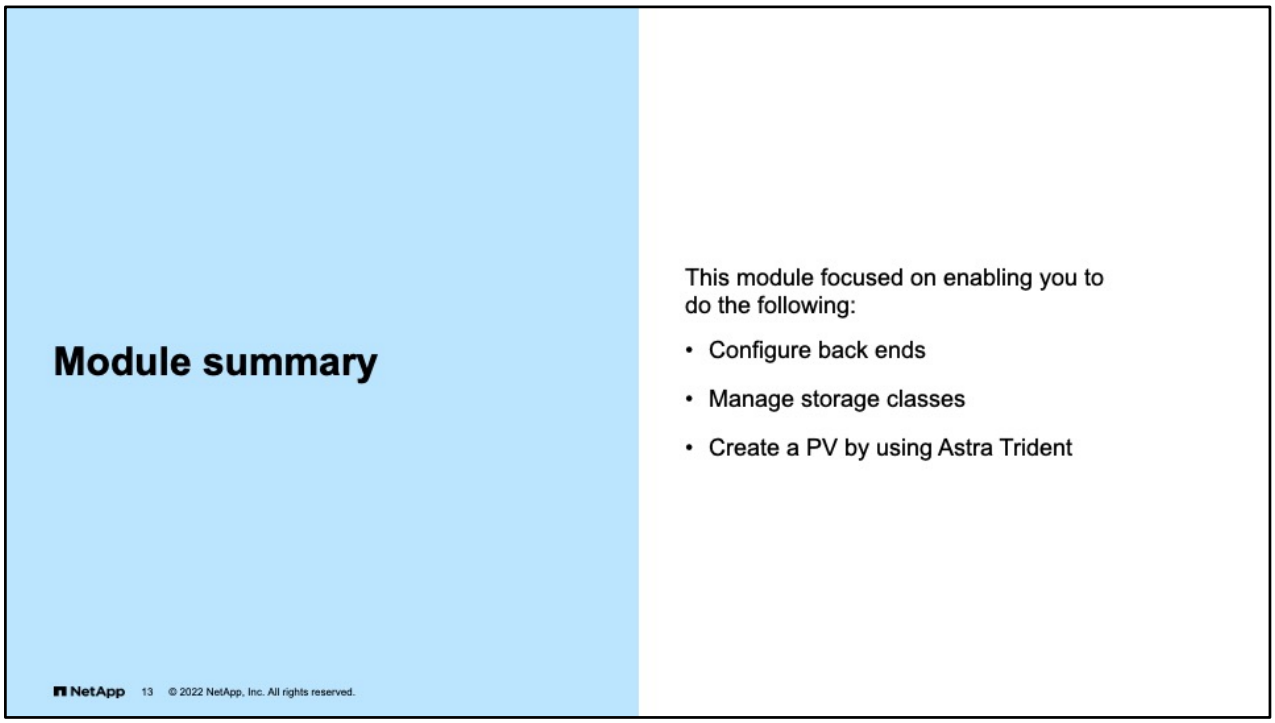
Student Notes



Instructor Notes

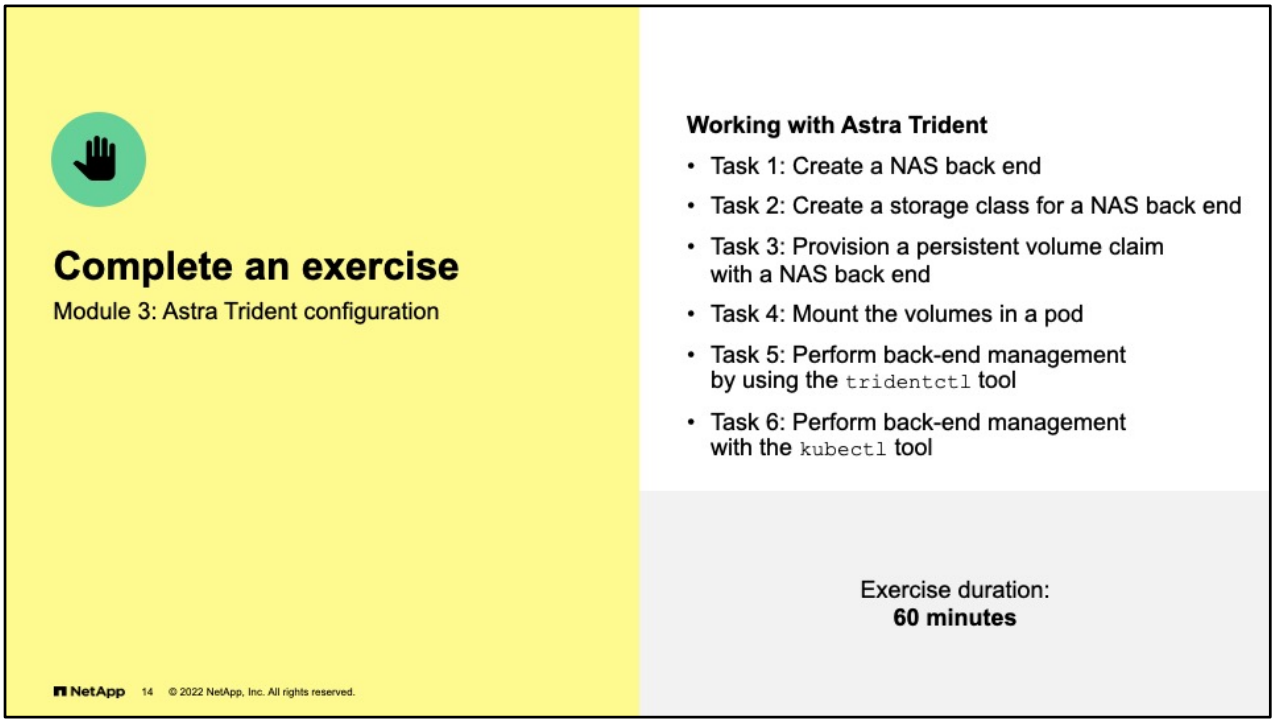
Student Notes

When Astra Trident provisions a persistent volume, it selects which back end to use based on the associated storage class's "backendType" parameter, which contains the name of a storage driver. For example, storage classes that specify the "ontap_nas" driver select a back end for ONTAP NFS storage. If you have multiple back ends of the desired type defined in your environment, Astra Trident randomly selects one on which to deploy the volume.



Instructor Notes

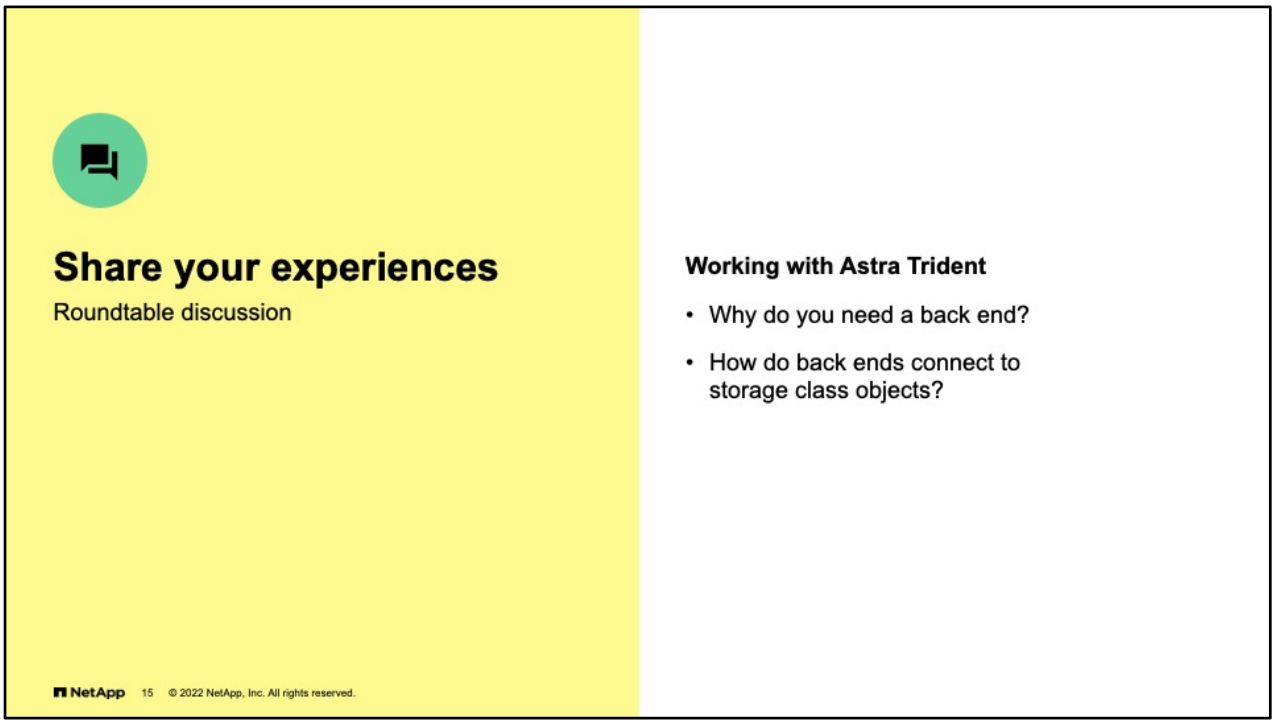
Student Notes



Instructor Notes

Student Notes

See the instructions in your Exercise Guide.



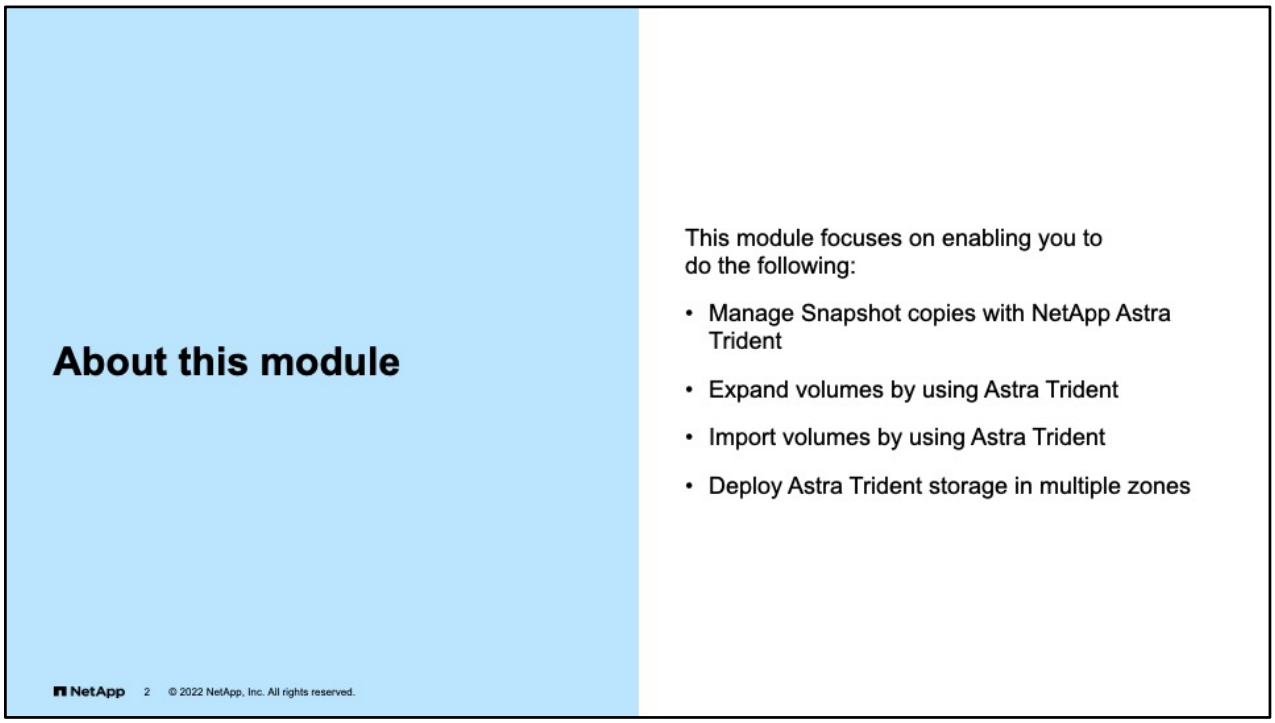
Instructor Notes
Display while students are completing the exercise(s)

Student Notes
Have a roundtable discussion with the class to answer these questions. Add any comments about experiences or “lessons learned” during the exercises that others might find helpful.

If you encounter an issue, notify your instructor immediately so that it can be resolved promptly.

Instructor Notes

Student Notes



Instructor Notes

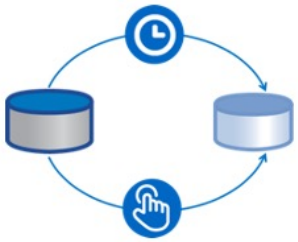
Student Notes

Instructor Notes

Student Notes

CSI Snapshots

Available in Astra Trident 20.01 or later



- Available from the community
- A point in time copy of data and the state of a volume
- A great framework to use as part of a backup job
- Scheduled or on-demand execution
- Standard method with Kubernetes v1.20 and later
- Container Storage Interface (CSI) Snapshots not *consistent* snapshots

Instructor Notes

Student Notes

For more information about the Container Storage Interface (CSI) specification, see https://github.com/container-storage-interface/spec/blob/master/spec.md. *Warning*: CSI Snapshots are not *consistent* snapshots. Application integration must be achieved with third-party solutions, such as NetApp Astra Control.

Resource definitions for Snapshots

Installed by using https://github.com/kubernetes-csi/external-snapshotter

- CRD: `VolumeSnapshotClass`
  - Defines the storage and storage parameters that are used in volume snapshots
  - Sets a deletion policy:
    - Delete: when deleting the `volumesnapshot`, the underlying data is also deleted
    - Retain: when deleting the `volumesnapshot`, the underlying data is not deleted
  - Is similar to `storageclass`
- CRD: `VolumeSnapshot`
  - Defines a request to create a snapshot of a volume
  - Is used by Astra Trident to create a NetApp Snapshot copy
  - Is associated with a single `volumesnapshotclass`
- CRD: `VolumeSnapshotContents`
  - Represents a snapshot taken from an already provisioned volume
  - Contains details about the unique snapshot
  - Has snapshotHandle attribute with the name of the persistent volume (PV) used

CRD: Custom Resource Definition

Instructor Notes

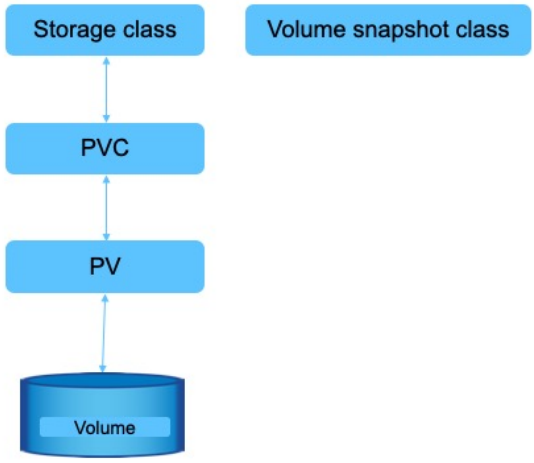
Student Notes

Instructor Notes

Student Notes

CSI Snapshot workflow

Volume snapshot class creation



Volume snapshot class

```
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshotClass
metadata:
 name: csi-snapclass
spec:
 driver: csi.trident.netapp.io
 deletionPolicy: Delete
```

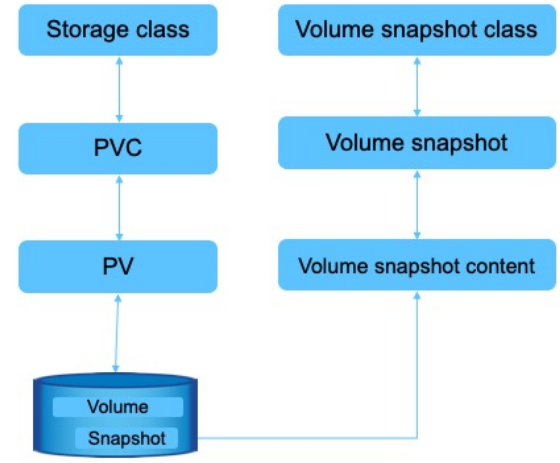
Can be "Delete" or "Retain"

Instructor Notes

Student Notes

CSI Snapshot workflow

Volume snapshot creation




```
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshot
metadata:
 name: volume-snapshot
spec:
 volumeSnapshotClassName: csi-snapclass
 source:
  persistentVolumeClaimName: pvc
```

Supported with the following drivers:

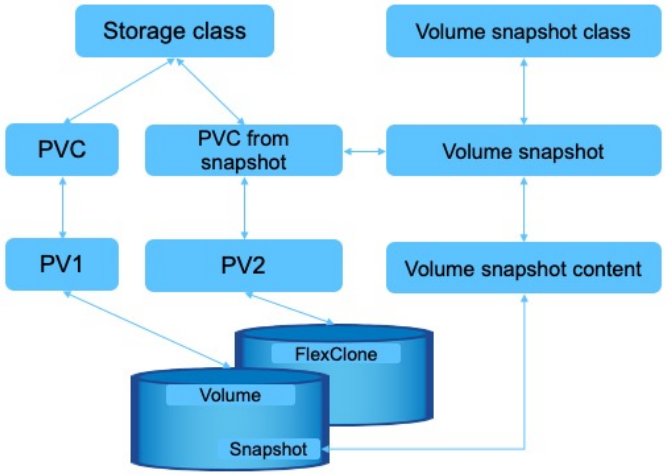
- ontap-nas
- ontap-san and ontap-san-economy
- solidfire-san
- aws-cvs
- gcp-cvs
- azure-netapp-files

Instructor Notes

Student Notes

CSI Snapshot workflow

Volume from snapshot creation



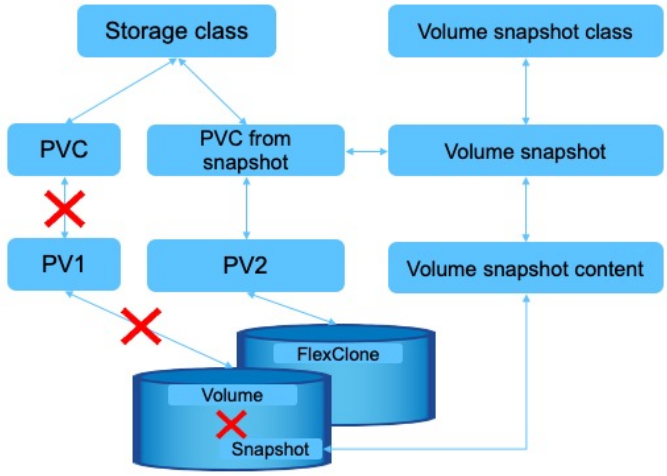

```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-from-snap
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: sc-nas
  dataSource:
    name: volume-snapshot
    kind: VolumeSnapshot
    apiGroup: snapshot.storage.k8s.io
```


Instructor Notes

Student Notes

CSI Snapshot workflow

Restore from a volume snapshot




```yaml
apiVersion: v1
kind: Pod
metadata:
  name: pod
spec:
  volumes:
    - name: storage
      persistentVolumeClaim:
       claimName: pvc-from-snap
  containers:
    - name: container
      image: nginx
      resources:
        requests:
          memory: "64Mi"
          cpu: "250m"
        limits:
          memory: "128Mi"
          cpu: "500m"
      ports:
        - containerPort: 80
          name: "http-server"
      volumeMounts:
        - mountPath: "/usr/share/nginx/html"
          name: storage
```

To reassociate an existing pod to the new restored snapshot PVC

Instructor Notes

Student Notes

Effect of SnapReserve

Using Astra Trident 21.07 or later

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
...
  defaults:
    snapshotDir: 'true'
    snapshotPolicy: default
    snapshotReserve: '40'


apiVersion: v1
kind: PersistentVolumeClaim
...
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 10Gi
...
```


```
# kubectl exec pod -- df -h /data
Filesystem                Size      Used  Available Use% Mounted on
192.168.0.31:/nfs_pvc…    1.0G    192.0K      11.0G   0% /data
```


Instructor Notes

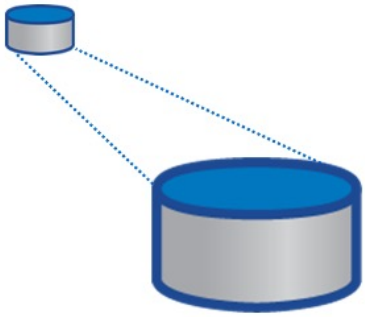
Student Notes

Astra Trident 21.07 sizes the available size of the FlexVol volume according to the PVC size. If a snapshot reserve is set, snapshot reserve is added to the available size of the FlexVol volume. The pod then sees the size requested in the PVC.

Instructor Notes

Student Notes

Volume resize
Available in Astra Trident 20.01 or later



- Can expand volume size as needs grow
- Has no need for administrator intervention
- Supports NFS and iSCSI

Instructor Notes

Student Notes

Resizing a volume

Using Astra Trident

To resize an existing PV:

- Change to the requested size in the PVC
- Astra Trident behavior:
  - If the PV is attached to a pod, expands the volume on the storage back end, rescans the device, and resizes the filesystem
  - When attempting to resize an unattached PV, expands the volume on only the storage back end

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 2Gi
  storageClassName: nfs-basic
```

To change the PVC requested size

Instructor Notes

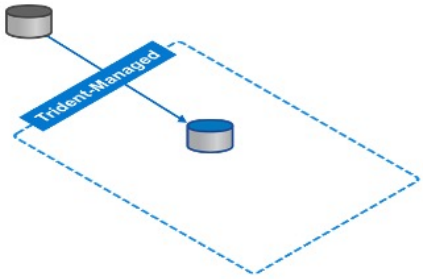
Student Notes

Instructor Notes

Student Notes

Volume import steps

Available in Astra Trident 20.01 or later



- Use cases include the following:
  - Inter-cluster data migration
  - Monolithic application migration to Kubernetes
  - Disaster recovery
- Supports NFS and iSCSI

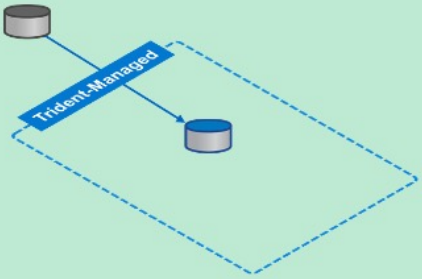
Instructor Notes

Student Notes

Volume import steps

Using Astra Trident

1. Verify the availability of a volume not managed by Astra Trident
2. Create a PVC that is appropriate for the volume that you want to import
3. Select a back end to manage the volume
4. Use `tridentctl` to import the volume:
   ```
   ./tridentctl import volume [backend] [volume name] -f [YAML file of PVC] -n [namespace]
   ```
5. Verify that the volume was imported and now appears as a PVC



Instructor Notes

Student Notes

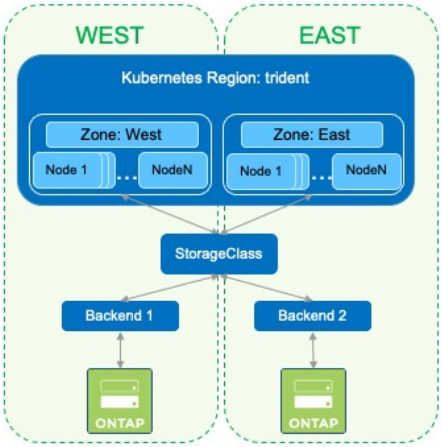
Instructor Notes

Student Notes

CSI topologies

A community-adopted standard

- Are used to support different storage across different segments
  - Racks
  - Regions
  - Zones (or availability zones)
  - Domains
  - Functionality (dev versus test versus prod)
- Are used to provide storage available to the different segments
- Ensure that the volume is created in the same segment as the pod
- Can reduce the number of storage classes needed

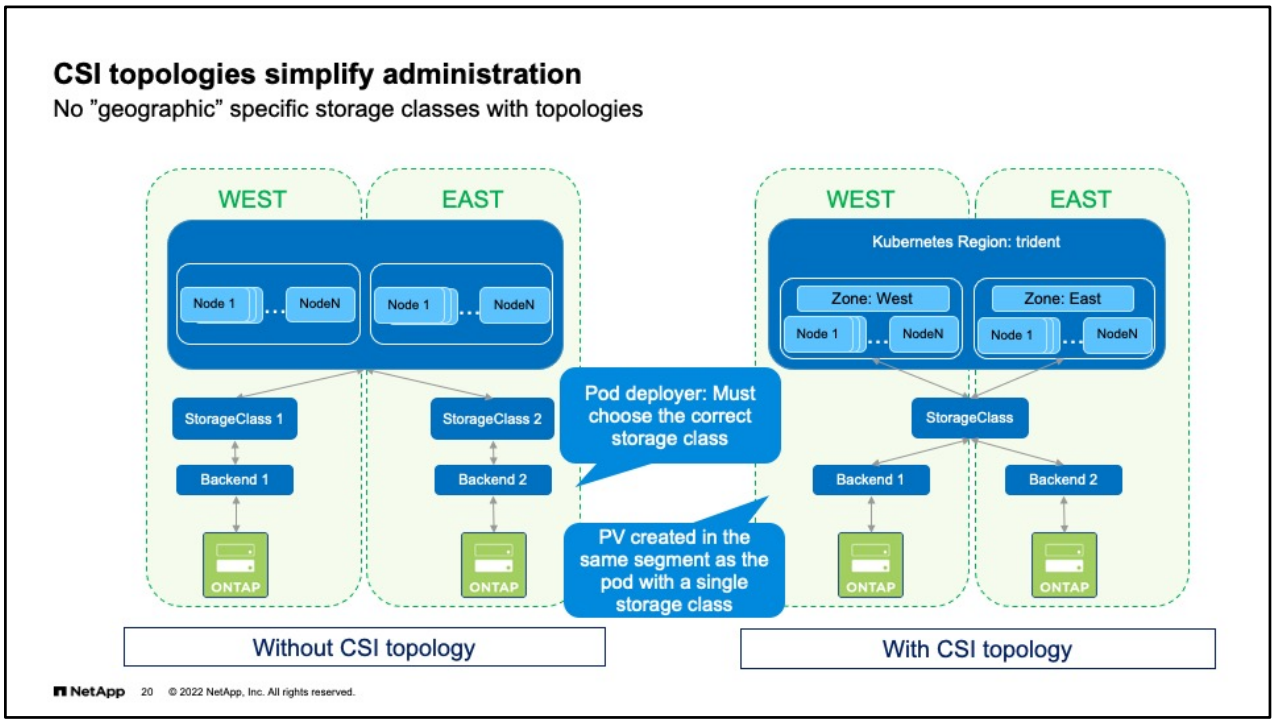


With CSI topology

Instructor Notes

Student Notes

For more information about Kubernetes CSI topologies, see https://kubernetes-csi.github.io/docs/topology.html.



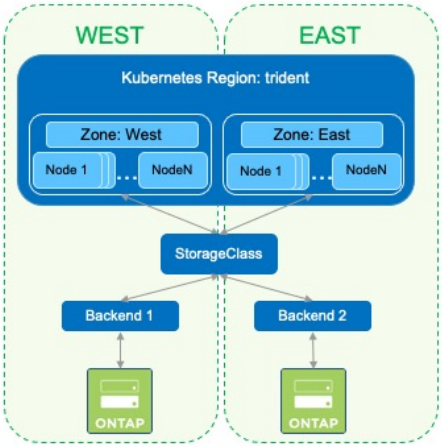
Instructor Notes

Student Notes

CSI topologies steps

Requires Kubernetes 1.17 or later and Astra Trident 20.10 or later

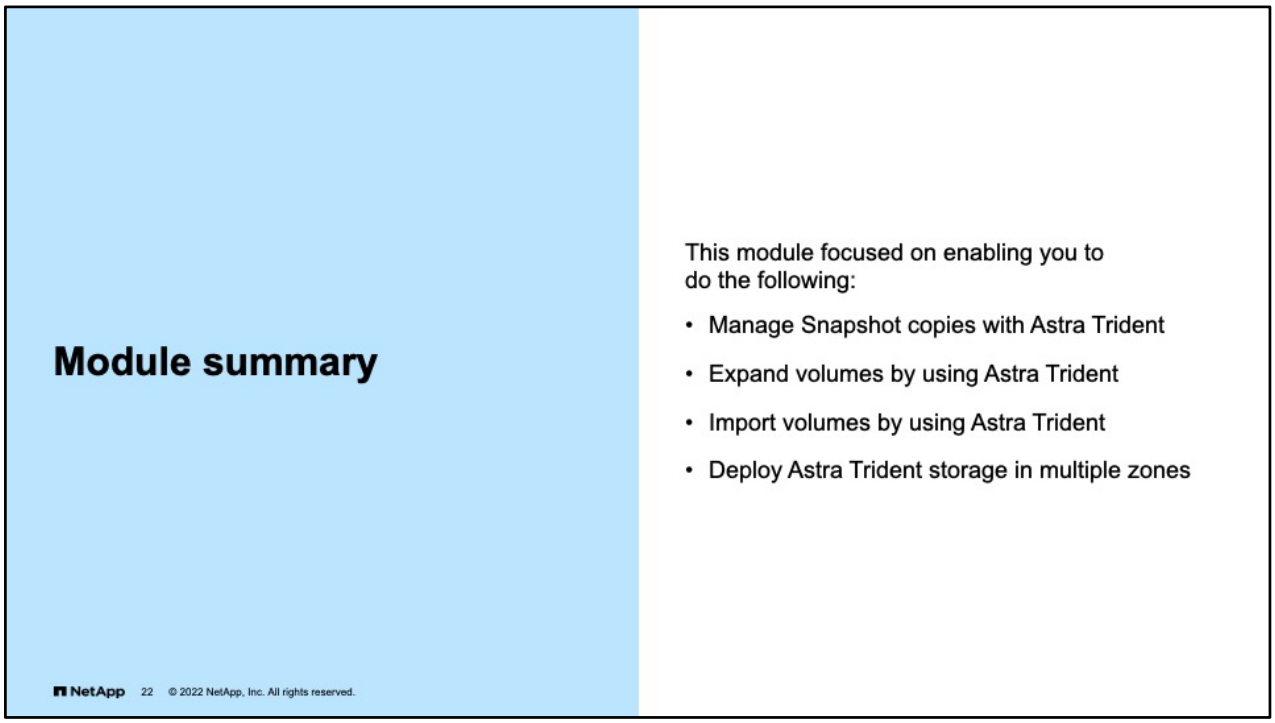
1. Label the Kubernetes nodes before installing Astra Trident
   - `topology.kubernetes.io/region`
   - `topology.kubernetes.io/zone`
2. Kubernetes storage classes optional parameters:
   - `allowedTopologies`
   - `volumeBindingMode: WaitForFirstConsumer`
3. Specify Astra Trident back-end parameter `supportedTopologies`
4. Define a pod by using the storage class created in Step 3



With CSI topology

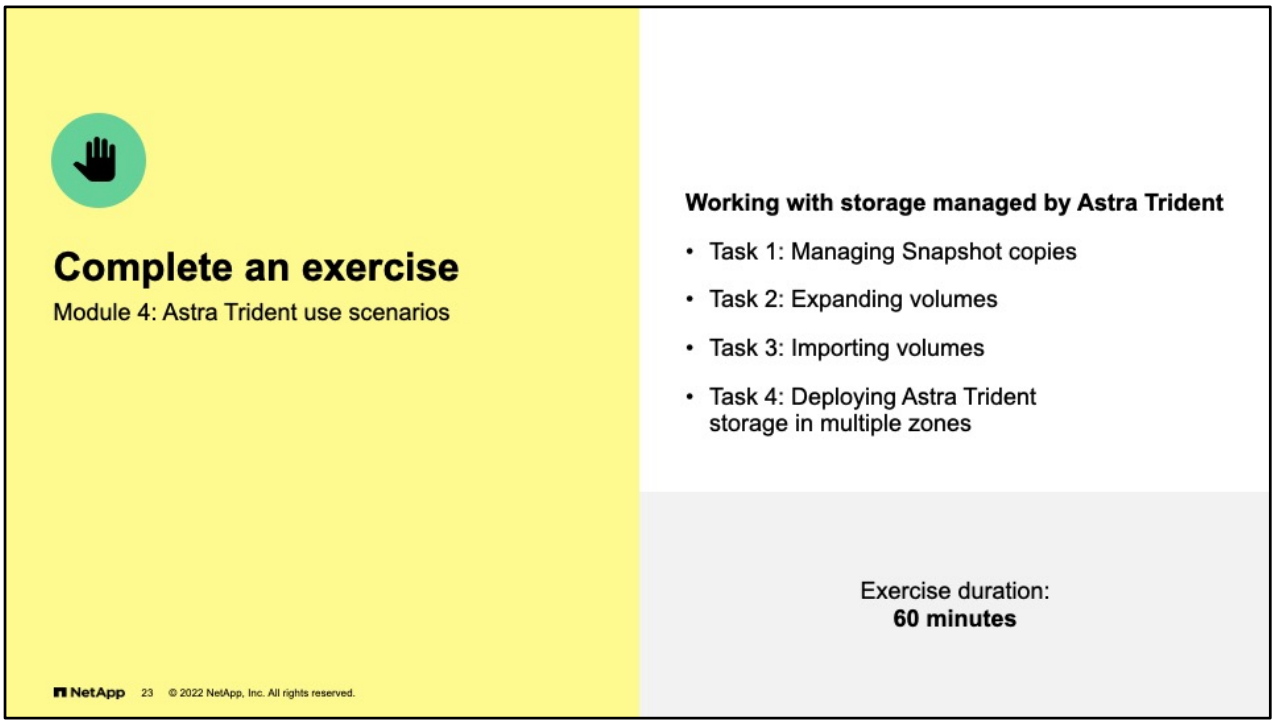
Instructor Notes

Student Notes



Instructor Notes

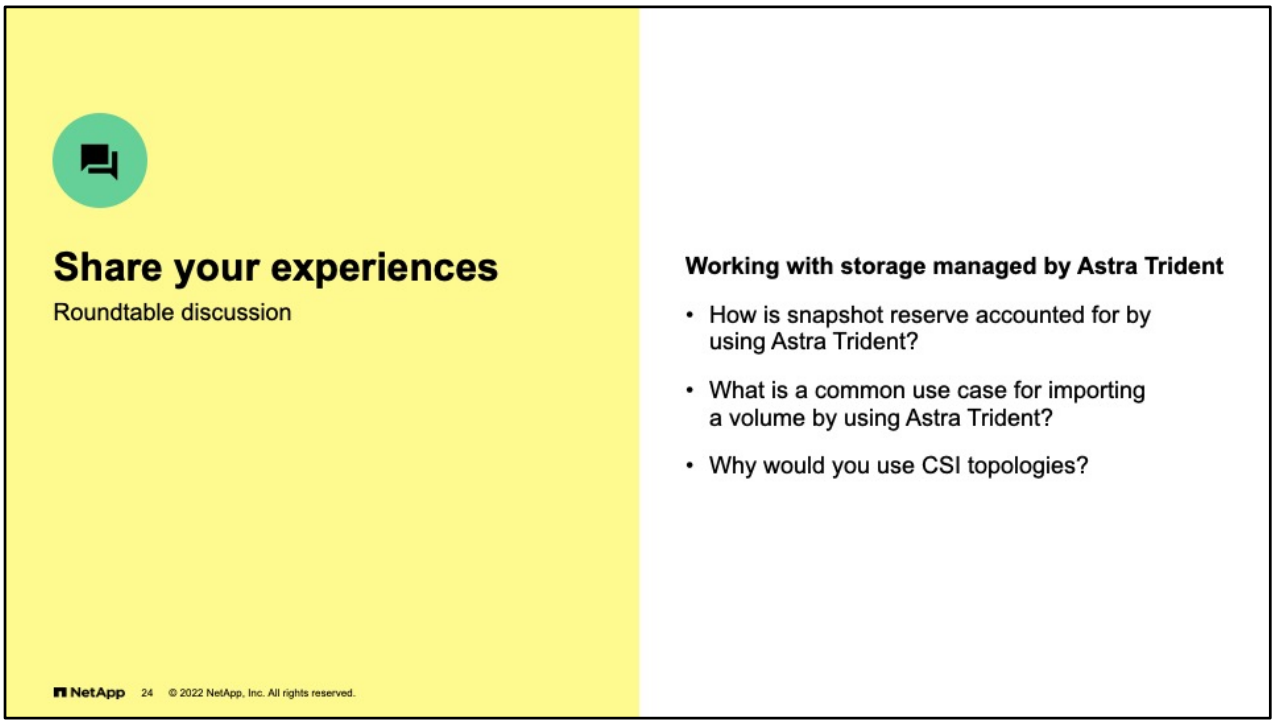
Student Notes



Instructor Notes

Student Notes

See the instructions in your Exercise Guide.



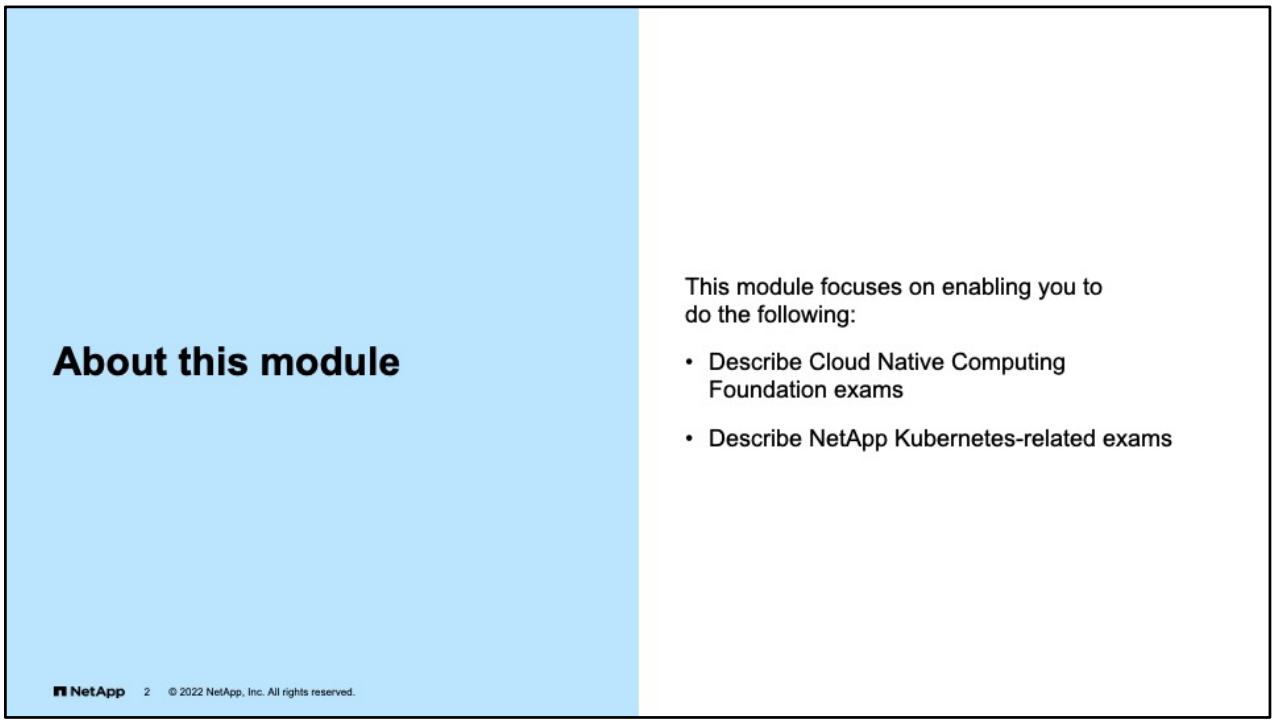
Instructor Notes
Display while students are completing the exercise(s)

Student Notes
Have a roundtable discussion with the class to answer these questions. Add any comments about experiences or “lessons learned” during the exercises that others might find helpful.

If you encounter an issue, notify your instructor immediately so that it can be resolved promptly.

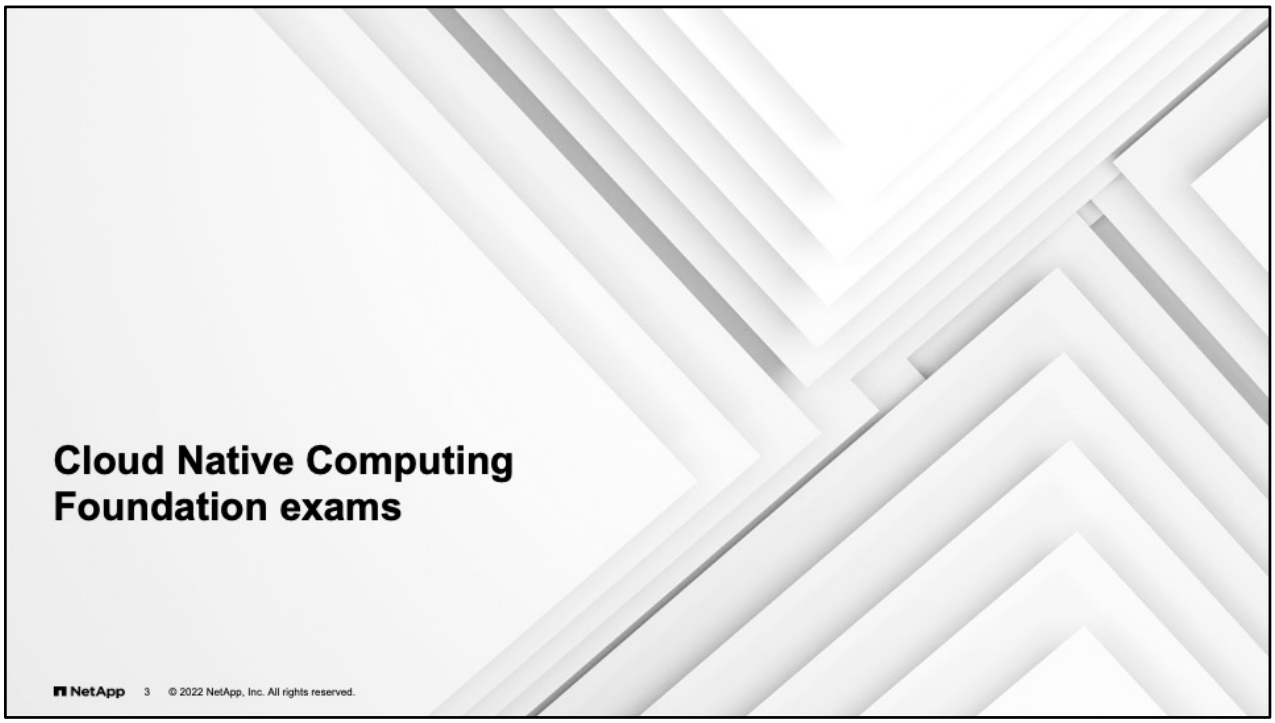
Instructor Notes

Student Notes



Instructor Notes

Student Notes

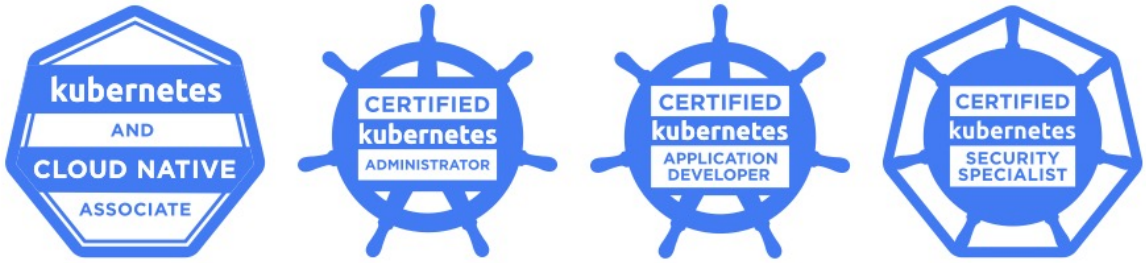


Instructor Notes

Student Notes

Cloud Native Computing Foundation (CNCF) and the Linux Foundation

General information—subject to change



Instructor Notes

Student Notes

Cloud Native Associate
General information—subject to change

- Cost: $250 with one free retake
- Time limit: 90 minutes
- Passing score: 75%
- Validity: 3 years
- Languages: English
- Proctored: with audio, video, and screen-sharing feeds
- Multiple choice exam
- FAQ: https://docs.linuxfoundation.org/tc-docs/certification/frequently-asked-questions-kcna

Instructor Notes

Student Notes
https://www.cncf.io/certification/kcna/

Certified Kubernetes Administrator
General information—subject to change

- Cost: $375 with one free retake
- Time limit: 2 hours
- Passing score: 66%
- Validity: 3 years
- Languages: English, Simplified Chinese, and Japanese
- Proctored: with audio, video, and screen-sharing feeds
- Practical, command-line exam (with no multiple choice), access to some websites
- Updated frequently: currently ectd v3.5 and Kubernetes v1.22
- FAQ: https://docs.linuxfoundation.org/tc-docs/certification/faq-cka-ckad-cks

Instructor Notes

Student Notes

https://www.cncf.io/certification/cka/

Certified Kubernetes Application Developer
General information—subject to change

- Cost: $375 with one free retake
- Time limit: 2 hours
- Passing score: 66%
- Validity: 3 years
- Languages: English, Simplified Chinese, and Japanese
- Proctored: with audio, video, and screen-sharing feeds
- Practical, command-line exam (with no multiple choice), access to some websites
- Updated frequently: currently ectd v3.5 and Kubernetes v1.22
- FAQ: https://docs.linuxfoundation.org/tc-docs/certification/faq-cka-ckad-cks

Instructor Notes

Student Notes

https://www.cncf.io/certification/ckad/

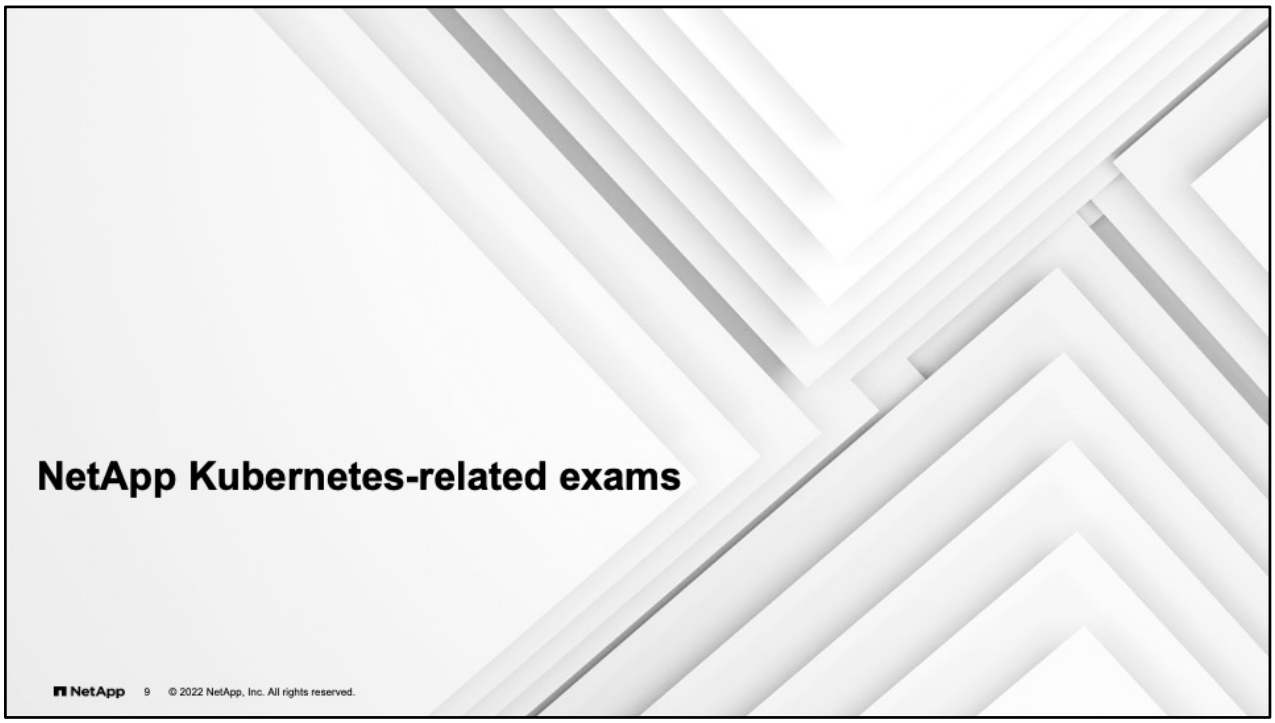
Certified Kubernetes Security Specialist
General information—subject to change

- Cost: $375 with one free retake
- Time limit: 2 hours
- Passing score: 67%
- Validity: 2 years
- Languages: English, Simplified Chinese, and Japanese
- Proctored: with audio, video, and screen-sharing feeds
- Practical, command-line exam (with no multiple choice), access to some websites
- Updated frequently: currently ectd v3.5 and Kubernetes v1.22
- FAQ: https://docs.linuxfoundation.org/tc-docs/certification/faq-cka-ckad-cks

Instructor Notes

Student Notes

https://www.cncf.io/certification/cks/



Instructor Notes

Student Notes

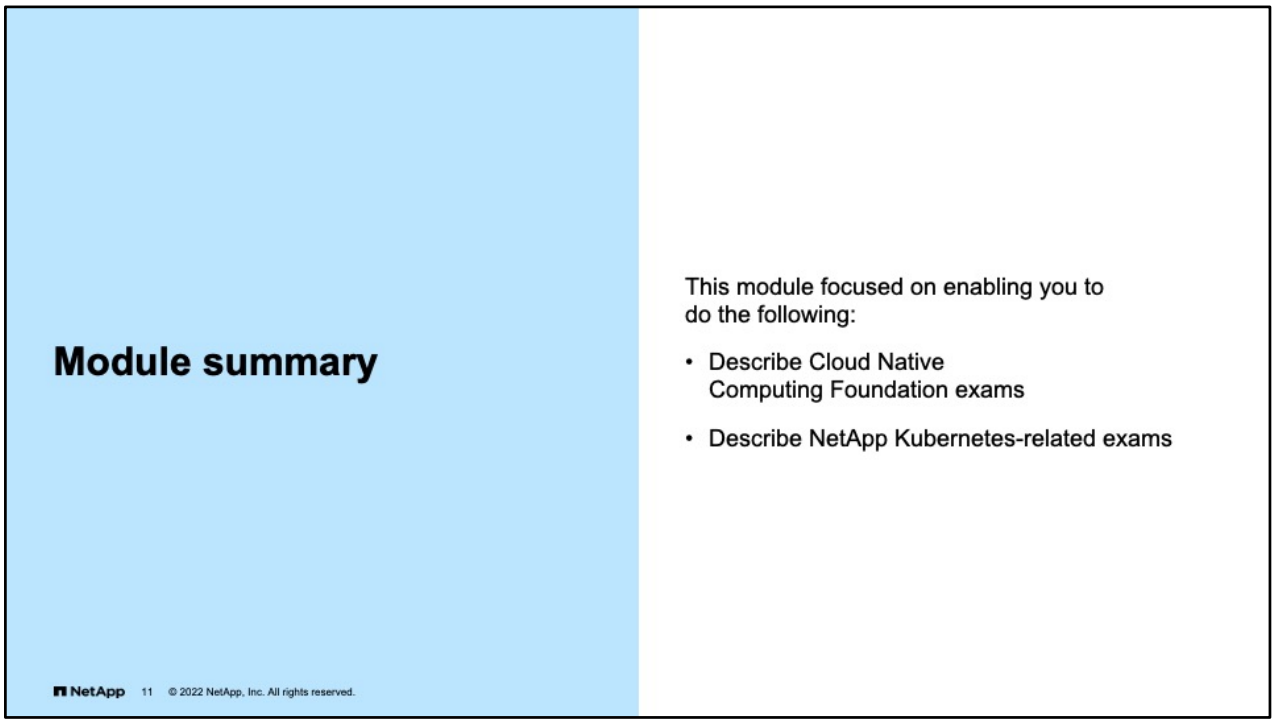
NetApp Certified Hybrid Cloud Implementation Engineer exam

Current version: NS0-403

- Number of test questions: **60**
- Time limit: **90 minutes** (additional 30 minutes allowed for non-English-speaking individuals)
- Passing score: **unpublished**
- Validity: **3 years**
- Languages: **English**
- Recommended field experience: **6–12 months**
- For more information: https://www.netapp.com/support-and-training/netapp-university-training-and-certification/certification/implementation-engineer/

Instructor Notes

Student Notes

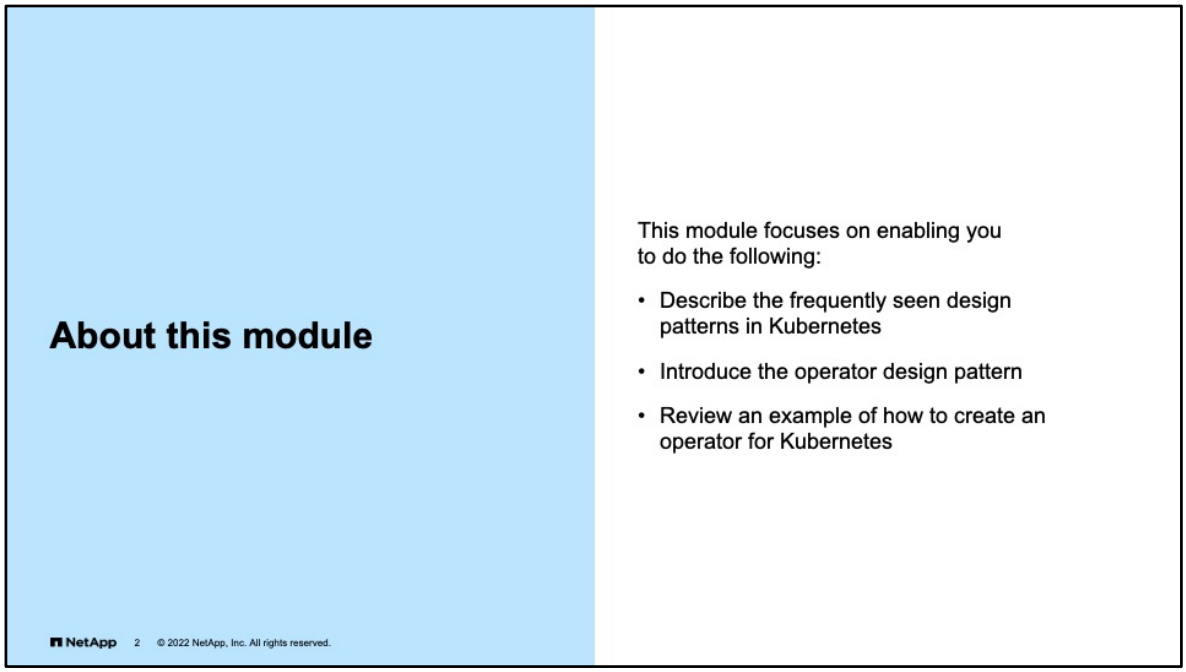


Instructor Notes

Student Notes

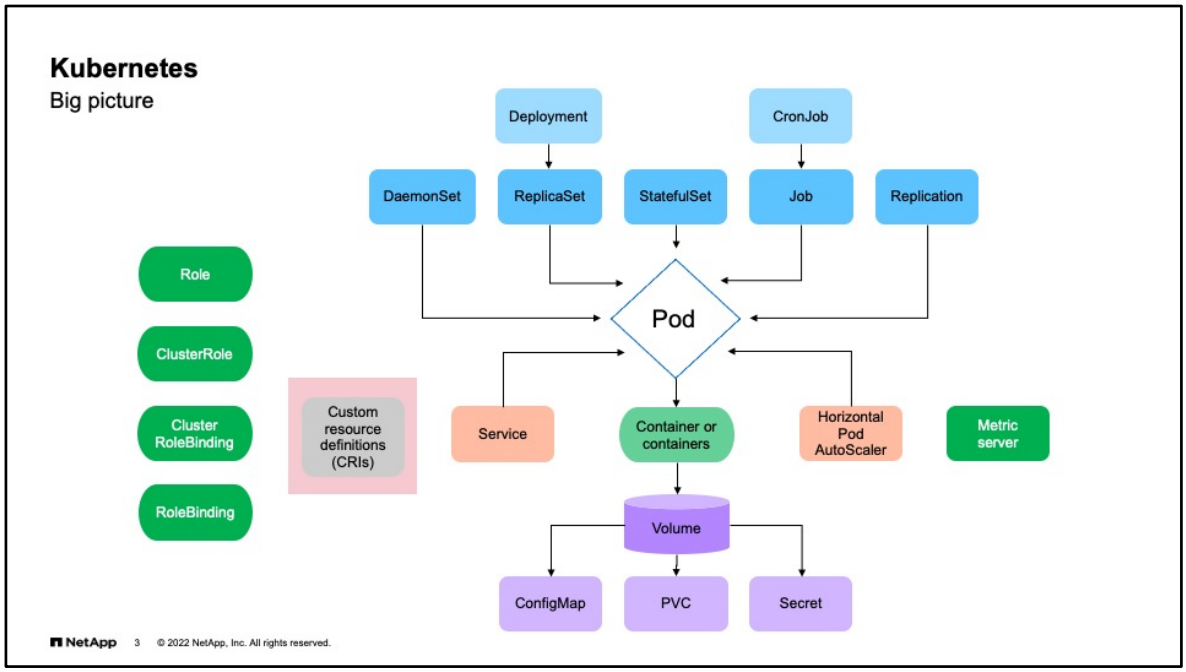
Instructor Notes

Student Notes



Instructor Notes

Student Notes



Instructor Notes

Student Notes

Instructor Notes

Student Notes

Design patterns classifications

Design patterns are classified in three ways

Category	Focus area	Examples
Creational patterns	Instantiation	Factory method: Defines a generic interface for object creation Object pools: Creates a pool of reusable objects
Structural patterns	Composition	Façade: Encapsulates interfaces of a complex system Proxy: Uses placeholder objects that represent other objects
Behavioral patterns	Communication	Iterator: Provides access to multiple elements of a complex object Observer: Notifies another interested object when a state changes

Instructor Notes

Student Notes

Creational patterns

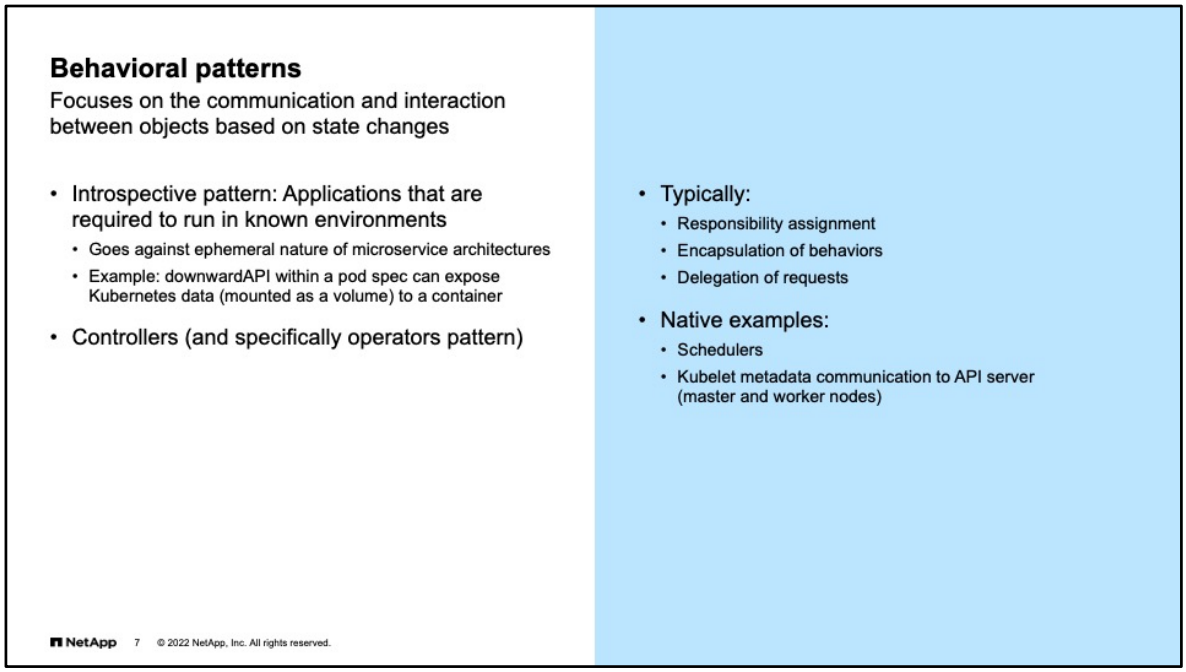
Instantiate objects within the environment based on various rules

- Sidecar pattern:
  - Is used to add monitoring, logging, configuration, testing, and networking
  - Adds support features to the main application (container)
  - Example: Istio sidecar
- Initialization pattern:
  - Is used during the startup of a pod
  - Provides “Separation of Concerns” within a pod
  - Example: initContainer—Executed for completion serially and expected to exit successfully
- Singleton pattern: Running only one instance (could be run as a StatefulSet with replica count of 1)

- Many examples natively:
  - Deployments / ReplicaSets
  - StatefulSets
  - DaemonSet
  - Job / Cronjob

Instructor Notes

Student Notes



Instructor Notes

Student Notes

For more information about downwardAPI, see https://kubernetes.io/docs/tasks/inject-data-application/downward-api-volume-expose-pod-information/.

Instructor Notes

Student Notes

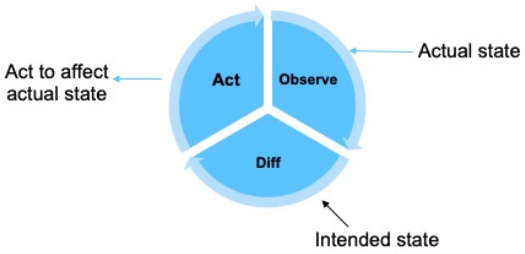
Recall: Kubernetes architecture

Controllers

Controllers

Have a control loop: observe, analyze, act

Example control loop ListWatcher: "Watches" for changes in etcd store and then remediates the situation



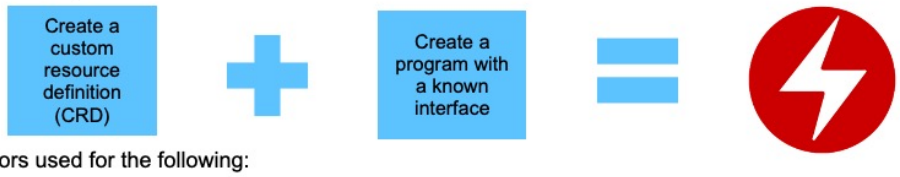
Instructor Notes

Student Notes

Extending Kubernetes

Overview of operators

- Definition: A custom controller watching a custom resource (CR) type and taking application-specific actions
- To create an operator:



- Operators used for the following:
  - Infrastructure applications: Storage, caching, databases, basic functionality (such as "PodSets")
  - Main applications: A method for deploying a specific application
- For a community of operators, see https://operatorhub.io/

Instructor Notes

Student Notes

Another list of awesome operators: https://github.com/operator-framework/awesome-operators

Instructor Notes

Student Notes

Example scenario

Sample workflow

- A slightly complex example (visitorapp):

- Manifests:
  - Front end = deployment and service (has a configurable title)
  - Back end = deployment and service (must know the database service name)
  - Database = deployment and service
  - Secret = credentials for the database

- The basic philosophy:
  - "SRE for every application."
  - "If a human operator needs to touch your system during normal operations, you have a bug."
  - "Operators are the bug-fixes."
- Use operators to do the following:
  - Run
  - Upgrade
  - Respond to errors
  - Slow performance

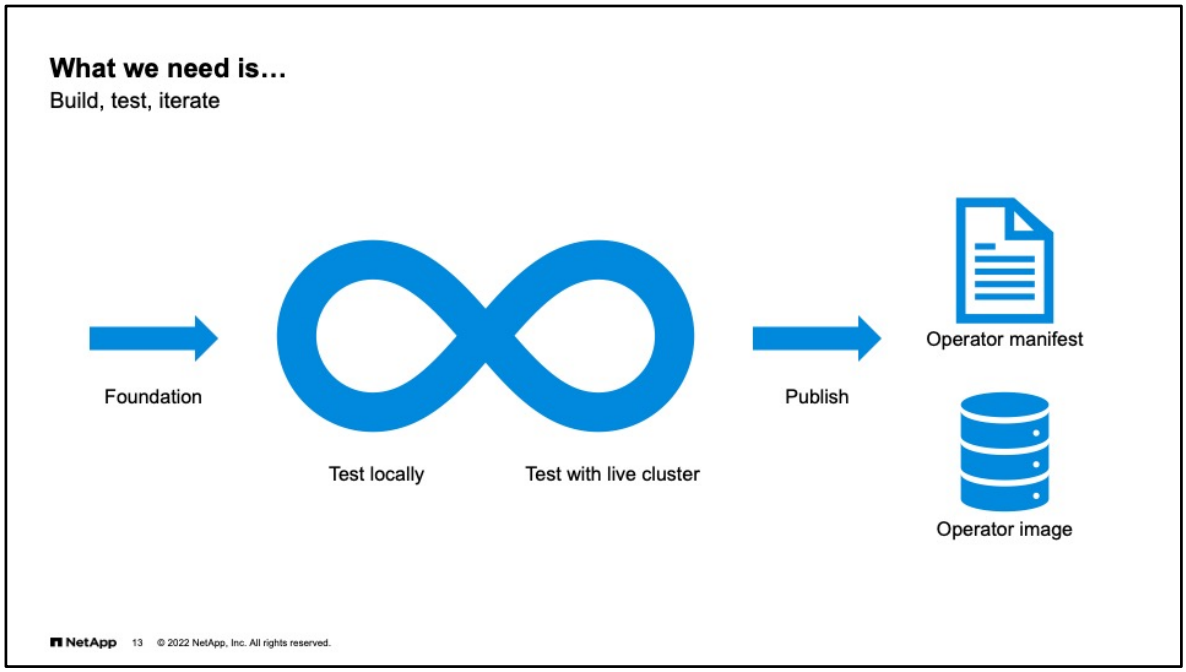
SRE: site reliability engineer

Instructor Notes

Student Notes

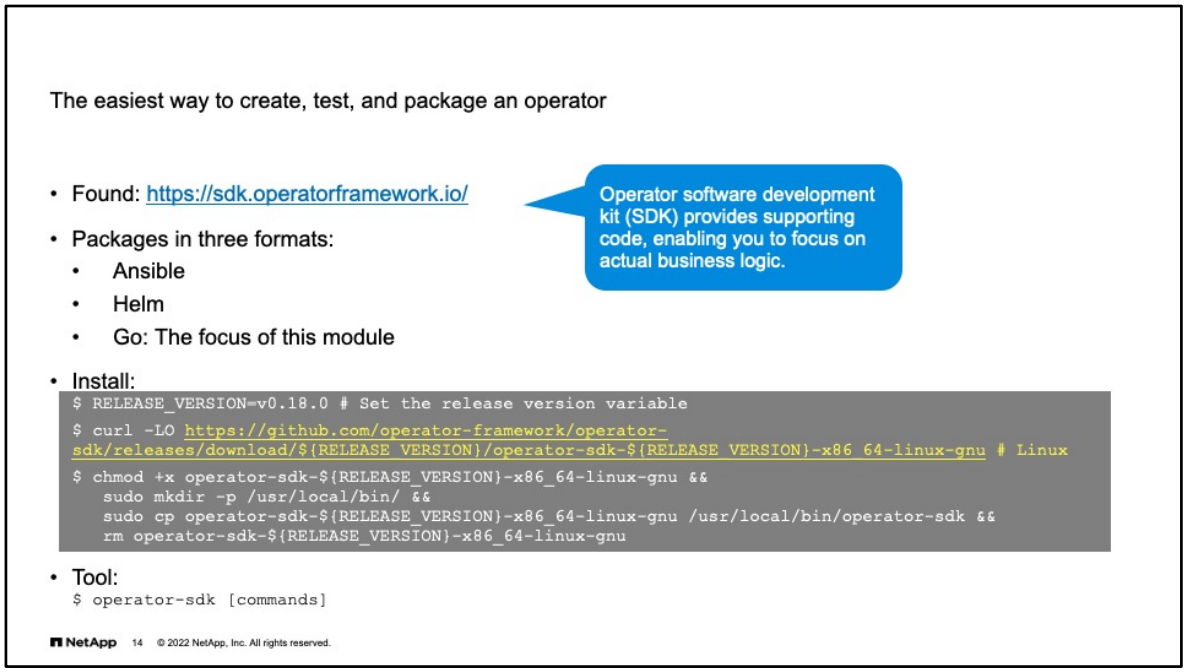
The example provided is from *Kubernetes Operators* (O'Reilly), by Jason Dobies and Joshua Wood.

The quotes are from *Site Reliability Engineering* (O'Reilly), by Betsy Beyer et al. (eds.).



Instructor Notes

Student Notes



Instructor Notes

Student Notes

For more information, see https://sdk.operatorframework.io/docs/.

Operator SDK project

Sample workflow

- Create an operator project:
  - `$ mkdir -p $HOME/projects`
  - `$ cd $HOME/projects`
  - `$ operator-sdk new visitorapp-operator`
  - `$ cd visitorapp-operator`
- Determine whether the operator is namespace-scoped or cluster-scoped
  - Defaults to namespace-scoped
  - To change it to a cluster-scoped:
    - Set the WATCH_NAMESPACE to "" in deploy/operator.yaml
    - Change the kind to ClusterRole in deploy/role.yaml
    - Change the kind to ClusterRoleBinding, under roleRef change the kind to ClusterRole, update the subjects with key namespace with the appropriate value in deploy/role_binding.yaml
    - Update the CRD spec to have scope of Cluster (see next step)
    - Update the _types.go file by adding // +genclient:nonNamespaced to the struct (see next step)

Instructor Notes

Student Notes

Custom resource definitions creation

Sample workflow

- Define the spec and status of the CR in pkg/apis/example/v1/visitorapp_types.go:

```
type visitorappSpec struct {
     Size  int32  `json:"size"`
     Title string `json:"title"`
}
type visitorappStatus struct {
     BackendImage  string `json:"backendImage"`
     FrontendImage string `json:"frontendImage"`
}
```

- Add a new CRD:

```
$ operator-sdk add api --api-
version=example.com/v1
                        --kind=VisitorApp
…
INFO[0000] Created
deploy/crds/example_v1_visitorapp_cr.yaml
INFO[0001] Created
deploy/crds/example_v1_visitorapp_crd.yaml …
```


Instructor Notes

Student Notes

Custom resource definitions verification

Need to have Kubernetes v1.16.x and later to run

- Verify and update the CRD file with the appropriate spec (and status) definitions:

```
apiVersion:
apiextensions.k8s.io/v1beta1
kind: CustomResourceDefinition
metadata:
  name: visitorapp.example.com
spec:
  group: example.com
  names:
    kind: VisitorApp
    listKind: VisitorAppList
    plural: visitorapps
    singular: visitorapp
…
```

```
    spec:
        type: object
        properties:
          size:
            type: integer
          title:
            type: string
        required:
        - size
      status:
        type: object
        properties:
          backendImage:
            type: string
          frontendImage:
            type: string
```

Removed title from "required"

Instructor Notes

Student Notes

Controller creation

Create a controller

- Add a new controller stub code:

```
$ operator-sdk add controller --api-version=example.com/v1 --kind=VisitorApp
```

- Modify pkg/controller/visitorapp/visitorapp_controller_go to perform the following logic:
  - Create a visitorapp Deployment if it does not exist
  - Ensure that the Deployment size is the same as specified by the visitorapp CR spec
  - Update the visitorapp CR status by using the status writer with the names of the visitorapp pods
- Review the watch for the primary resources as the add/update/delete events occur:

```
err := c.Watch(&source.Kind{Type: &examplev1.VisitorApp{}}, &handler.EnqueueRequestForObject{})
```

- Review the watch for the subresources as the add/update/delete events occur:

```
err := c.Watch(&source.Kind{Type: &appsv1.Deployment{}}, &handler.EnqueueRequestForOwner{
    IsController: true,
    OwnerType:    &examplev1.VisitorApp{},
})
...
```


Instructor Notes

Student Notes

Reconcile function

Core declarative implementation

- Verify the Reconcile() method that implements the reconcile loop:

```
func (r *ReconcileVisitorApp) Reconcile(request reconcile.Request) (reconcile.Result, error) {
  // Lookup the visitorapp instance for this reconcile request
  instance := &examplev1.VisitorApp{}
  err := r.client.Get(context.TODO(), request.NamespacedName, instance)
  ...
}
```

Makes sure that all the objects are created for a visitorapp

- Results of the function:
  - return reconcile.Result{}, nil //successful
  - return reconcile.Result{}, err //error - requeue
  - return reconcile.Result{Requeue: true}, nil //failed for another reason -requeue
  - return reconcile.Result{RequeueAfter: time.Second*5}, nil //requeue after 5 secs

Instructor Notes

Student Notes

Build and run the operator

Putting it all together

- Register the CRD:
  ```
  $ kubectl create -f deploy/crds/example.com_visitorapps_crd.yaml
  ```
- Build the operator:
  ```
  $ operator-sdk build [account]/visitorapp-operator:v0.0.1
  ```
- Push to Docker Hub:
  ```
  $ docker login
  $ docker push [account]/visitorapp-operator:v0.0.1
  ```
- Deploy the operator:
  ```
  $ kubectl create -f deploy/service_account.yaml
  $ kubectl create -f deploy/role.yaml
  $ kubectl create -f deploy/role_binding.yaml
  $ kubectl create -f deploy/operator.yaml
  ```
- Verify the running operator:
  ```
  $ kubectl get deployments
  NAME                 READY   UP-TO-DATE   AVAILABLE   AGE
  visitorapp-operator  1/1     1            1           1m
  ```

You can test your operator:
`operator-sdk run local --watch-namespace [ns]`

Optionally, use Operator Lifecycle Manager to deploy operators

Best practice: Modify role-based access control (RBAC) resources created by the operator SDK to grant only necessary permissions to run operator

Best practice: Aggregate all YAML files into a single YAML file

Instructor Notes

Student Notes

Test the operator

Create a VisitorApp type

- Create a definition file for an instance of the CR:

```
$ cat deploy/crds/example.com_v1_visitorapp_cr.yaml
 apiVersion: example.com/v1
 kind: visitorapp
 metadata:
   name: example-visitorapp
 spec:
   size: 1
   title: "My New Operator Dashboard"
```

- Create an instance of the CR:

```
$ kubectl apply -f deploy/crds/example.com_v1_visitorapp_cr.yaml
```

- Verify that the custom operator created the desired deployment:

```
$ kubectl get deployments
NAME                          READY   UP-TO-DATE   AVAILABLE   AGE
example-visitorapp-backend    1/1     1            1           5m
example-visitorapp-frontend   1/1     1            1           5m
example-visitorapp-mysql      1/1     1            1           5m
visitorapp-operator           1/1     1            1           5m
```


Instructor Notes

Student Notes

SRE for every app

Everything works

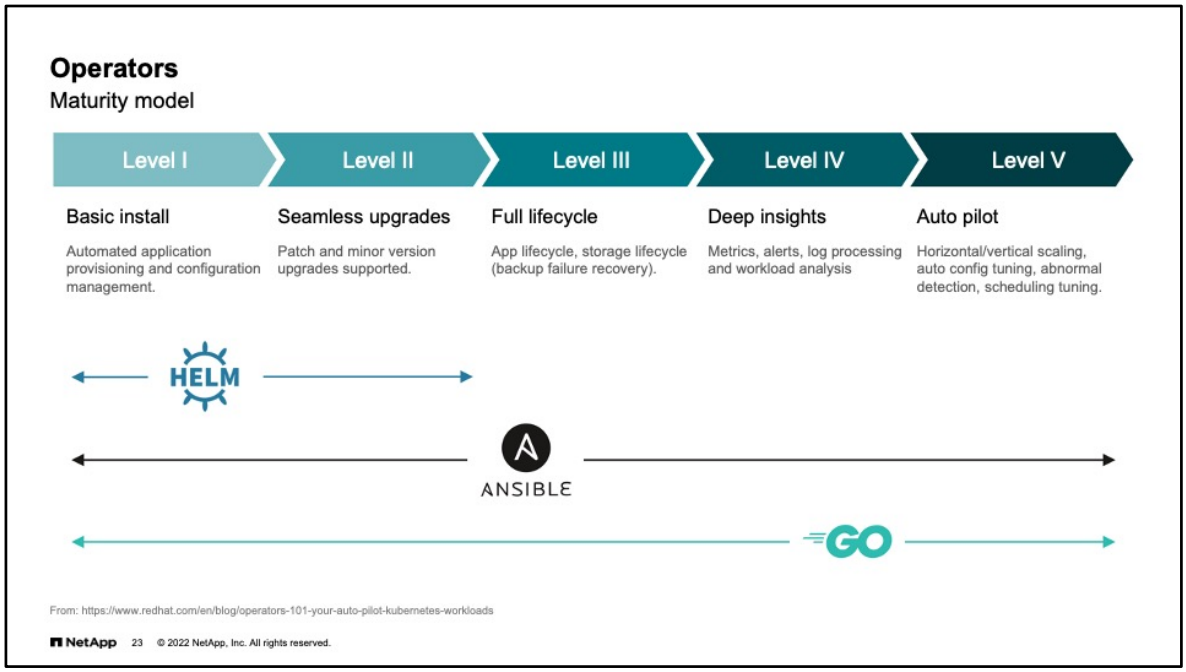
- Verify that the custom operator created the desired services:

```
$ kubectl get services
NAME                                TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)             AGE
example-visitorapp-backend-service  NodePort    10.104.164.255  <none>        8000:30685/TCP      68s
example-visitorapp-frontend-service NodePort    10.107.28.229   <none>        3000:30686/TCP      68s
example-visitorapp-mysql-service    ClusterIP   None            <none>        3306/TCP            70s
kubernetes                          ClusterIP   10.96.0.1       <none>        443/TCP             271d
visitorapp-operator-metrics         ClusterIP   10.103.18.72    <none>        8383/TCP,8686/TCP   11m
```

- Access the front-end image:

Instructor Notes

Student Notes



Instructor Notes

Student Notes

Four golden signals sent to software

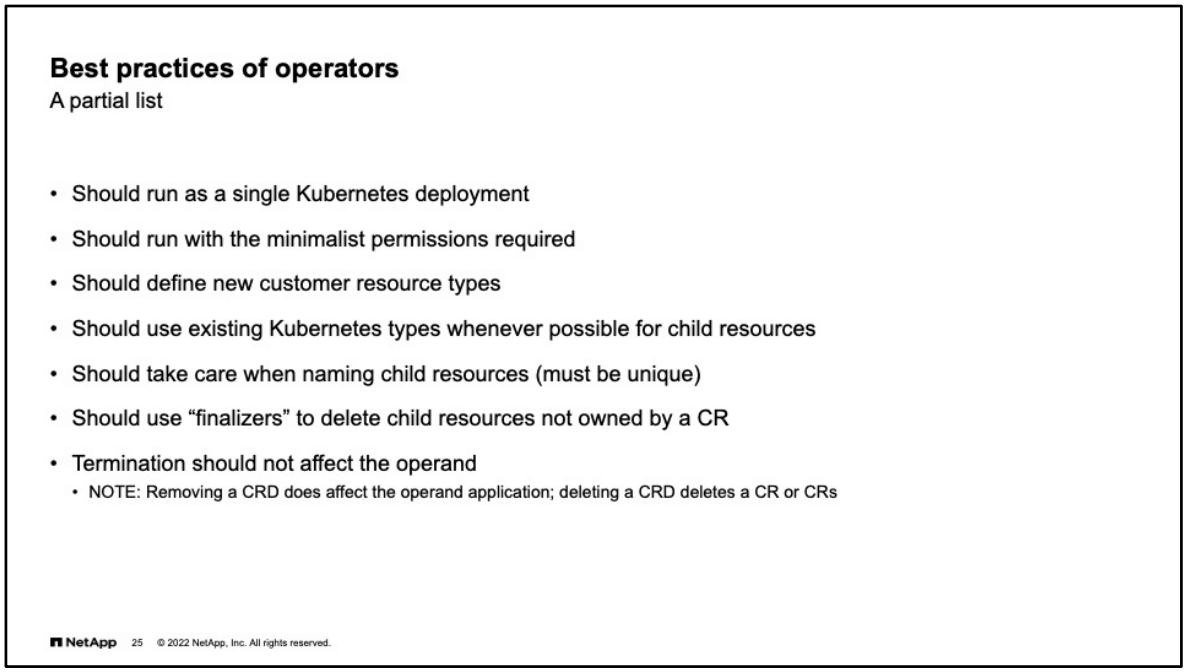
As the operator matures, verify that these metrics are captured

	Definition	Operator
Latency	How long it takes to do something	Measure application-specific latencies
Traffic	How frequently a service is requested	Measure transactions occurring with the application
Errors	The frequency and type of failed requests	Useful HTTP codes
Saturation	How much consumption of a limited resource	Could be CPU, memory, networking, or I/O

Instructor Notes

Student Notes

Four golden signals are taken from *Site Reliability Engineering* (O'Reilly), by Betsy Beyer et al. (eds.).



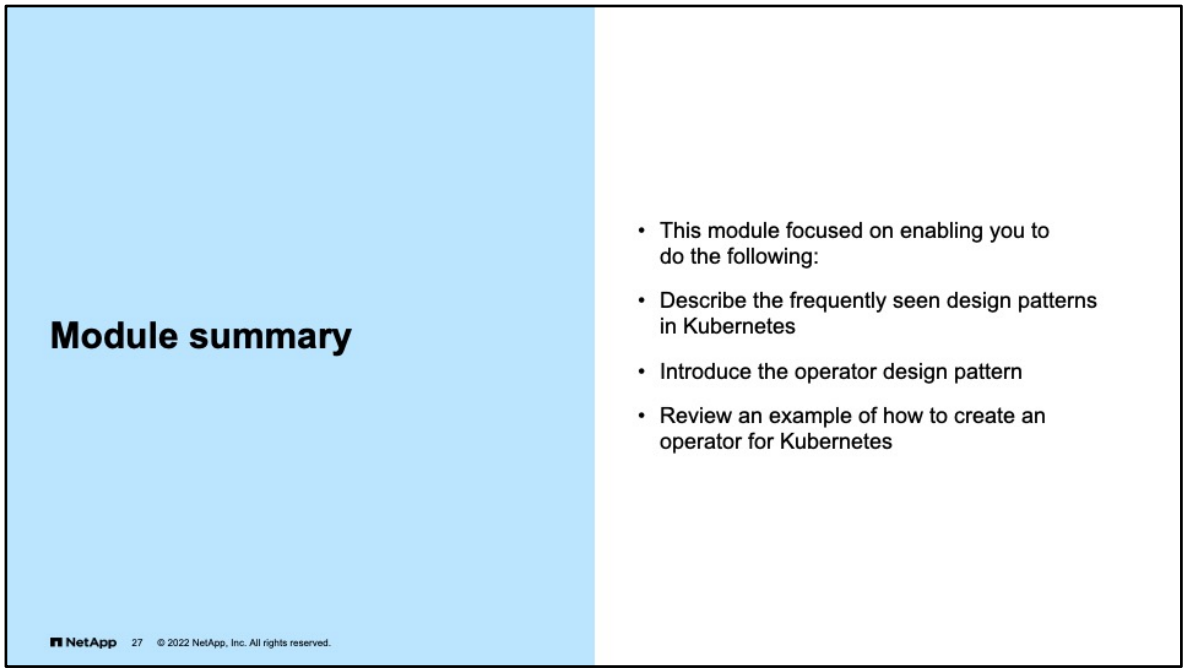
Instructor Notes

Student Notes

Finalizers: “Finalizers allow controllers to implement asynchronous pre-delete hooks. Let’s say you create an external resource (such as a storage bucket) for each object of your API type, and you want to delete the associated external resource on object’s deletion from Kubernetes, you can use a finalizer to do that.”

For more information, see https://book.kubebuilder.io/reference/using-finalizers.html.

The example does not use a finalizer because it did not create any “outside” resources that were not “owned” by the CR.

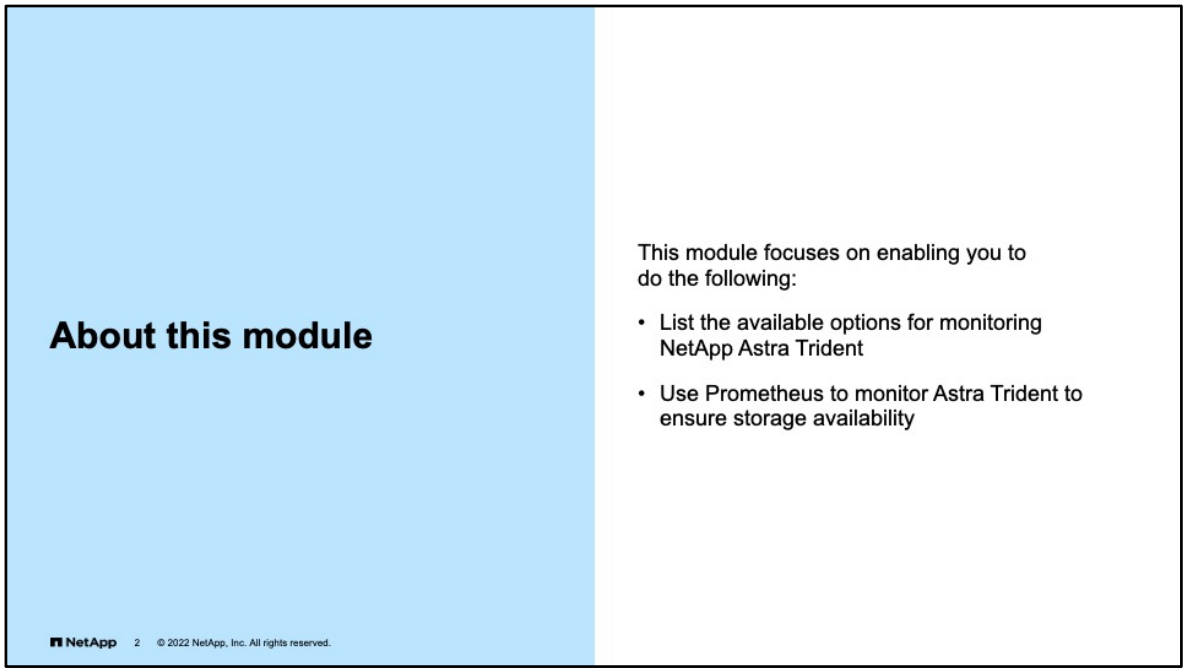


Instructor Notes

Student Notes

Instructor Notes

Student Notes

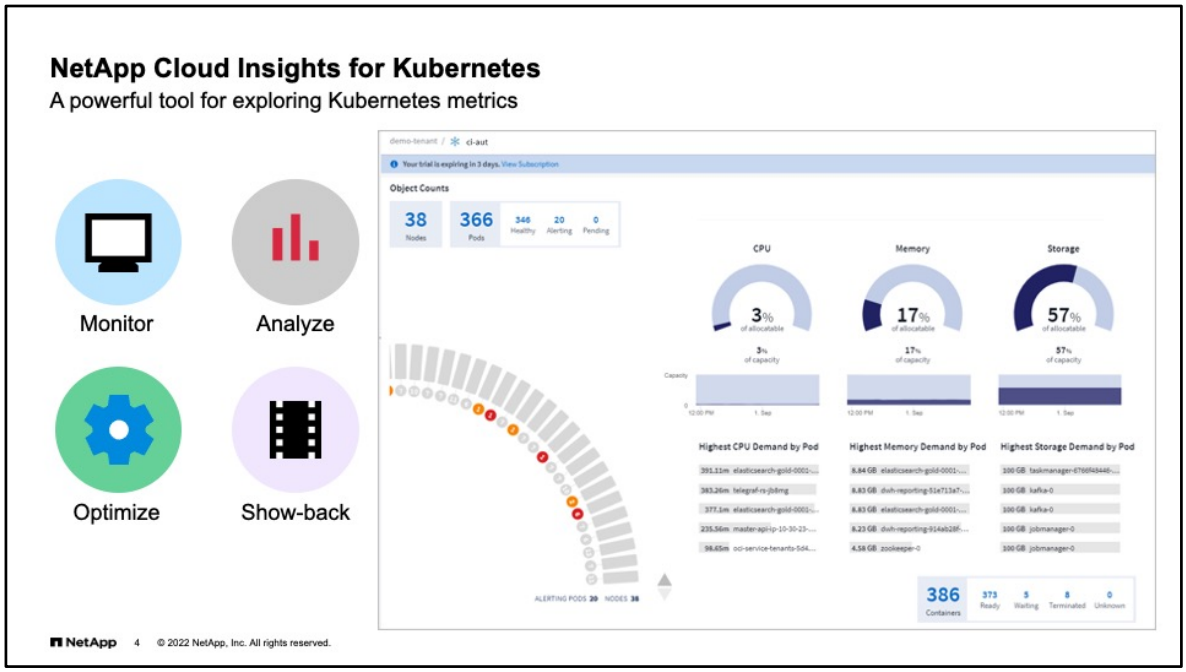


Instructor Notes

Student Notes

Instructor Notes

Student Notes

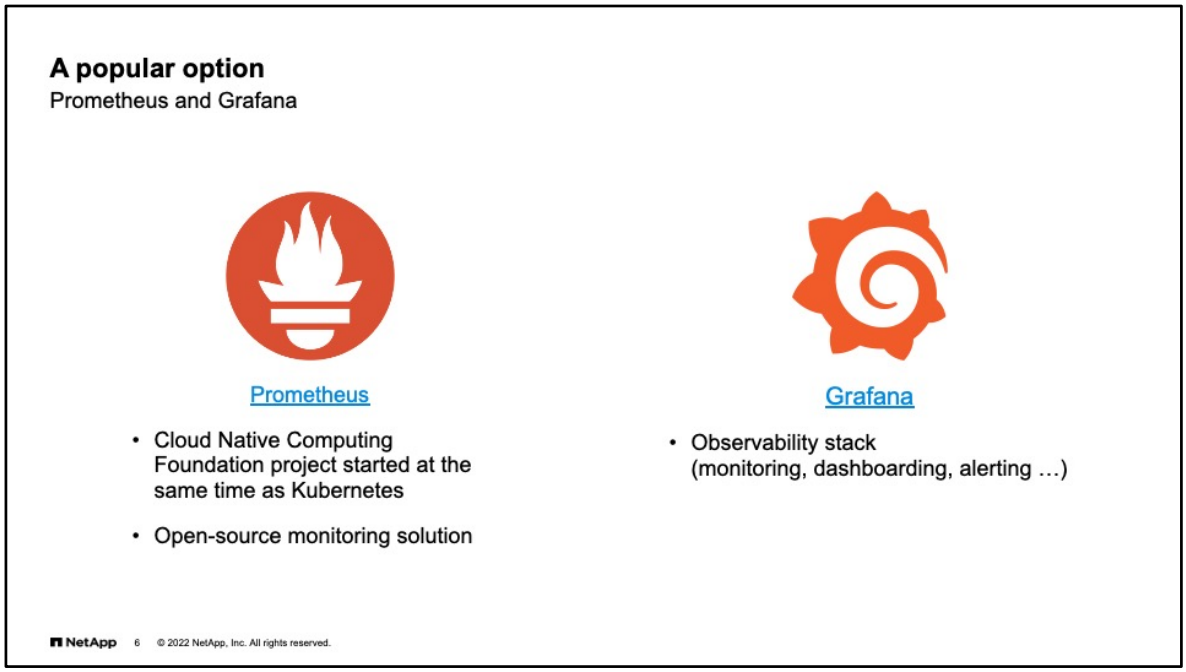


Instructor Notes

Student Notes

Instructor Notes

Student Notes



Instructor Notes

Student Notes

Harvest is not discussed in this course.

Prometheus setup overview

To install: https://github.com/prometheus-operator/kube-prometheus

- Create a monitoring namespace (default namespace for Prometheus)
- Install the custom resource definitions for Prometheus
- Apply the manifests to install Prometheus and related tools

Instructor Notes

Student Notes

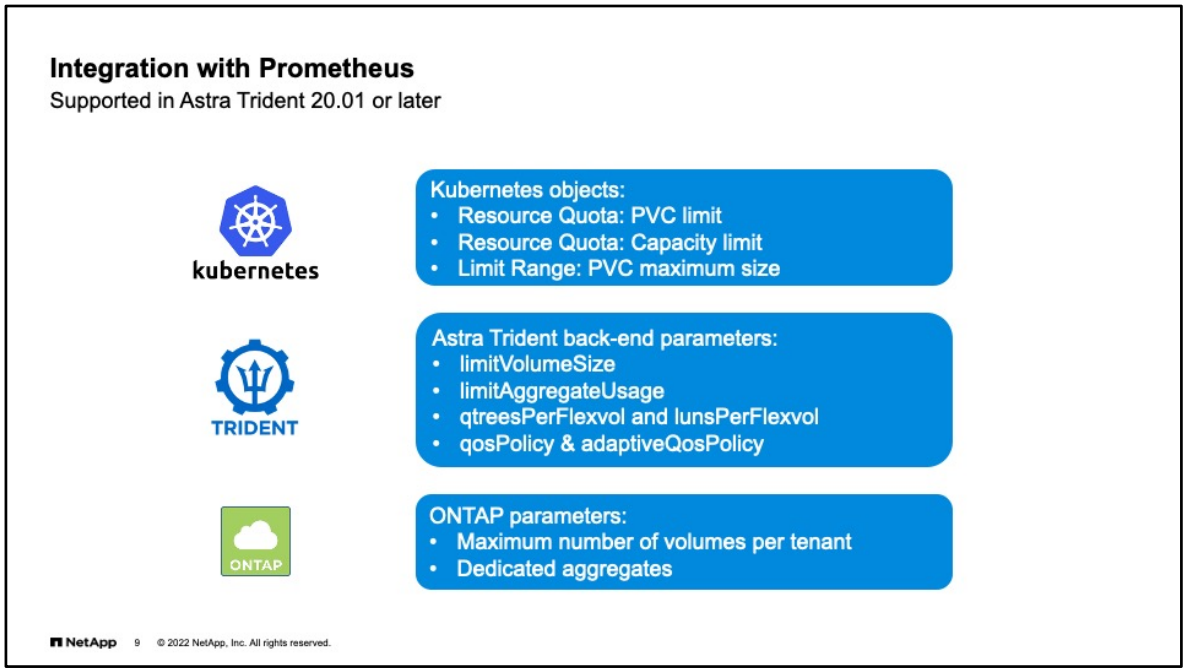
Prometheus tools

Usual pods deployed in a Prometheus setup

- Prometheus Operator (one pod): is used to manage all other Prometheus pods (deploys two Prometheus instances)
- Prometheus k8s (two pods): is used to aggregate metrics
- Alertmanager (three pods): is used to communicate Prometheus data with external services
- Kube State Metrics (one pod): is used to scrape telemetry data from your Kubernetes control plane services
- Node Explorer (one for each node in your cluster): is used to scrape telemetry data from your Kubernetes nodes
- Grafana (one pod): is used to provide a front-end UI for the Prometheus data
- Blackbox Exporter (one pod): enables black-box probing of endpoints over HTTP, HTTPS, DNS, TCP, and Internet ControlMessage Protocol (ICMP)
- Prometheus Adapter (two pods): contains an implementation of the Kubernetes resource metrics, custom metrics, and external metrics APIs to use with Horizontal Pod Autoscaler

Instructor Notes

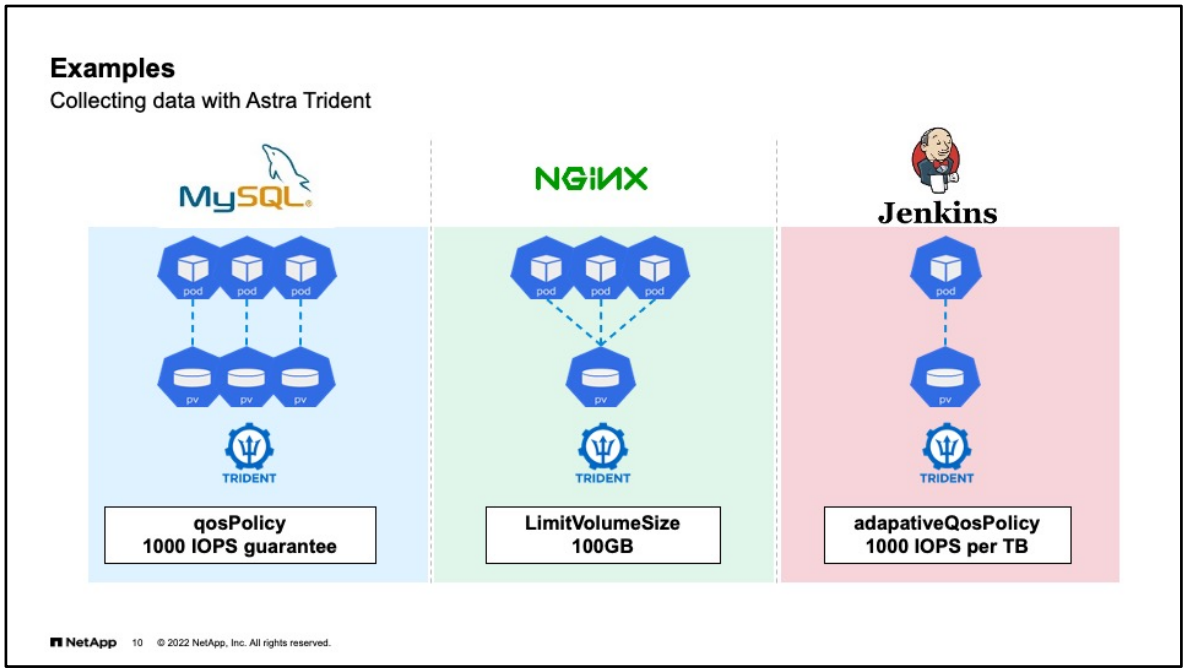
Student Notes



Instructor Notes

Student Notes

For more information,
see https://github.com/NetApp/trident/blob/master/core/metrics.go and https://netapp.io/2020/02/20/a-primer-on-prometheus-trident.



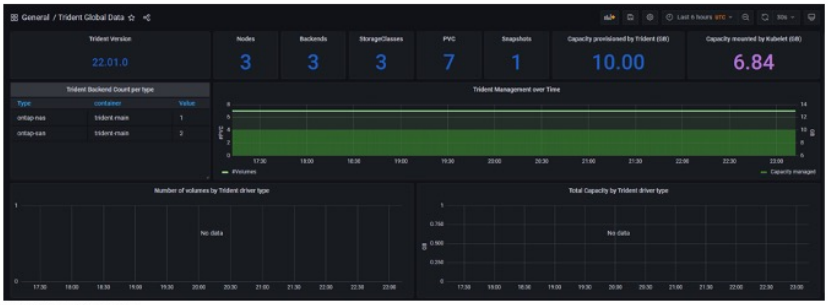
Instructor Notes

Student Notes

Monitoring Astra Trident as a Prometheus target

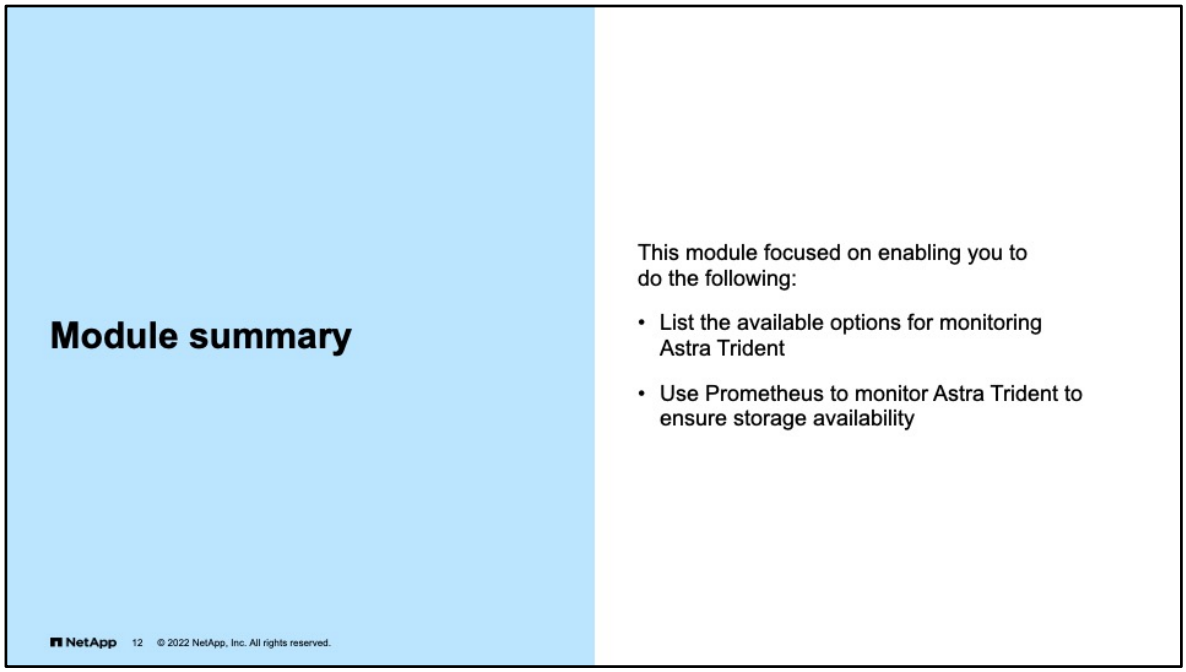
Using Prometheus and Grafana

1. Create a `role` and `rolebinding` to allow Prometheus to monitor the namespace used for Astra Trident
2. Create a Prometheus `servicemonitor` to point to the metrics port exposed by Astra Trident
3. Create a Grafana dashboard to review the Astra Trident data



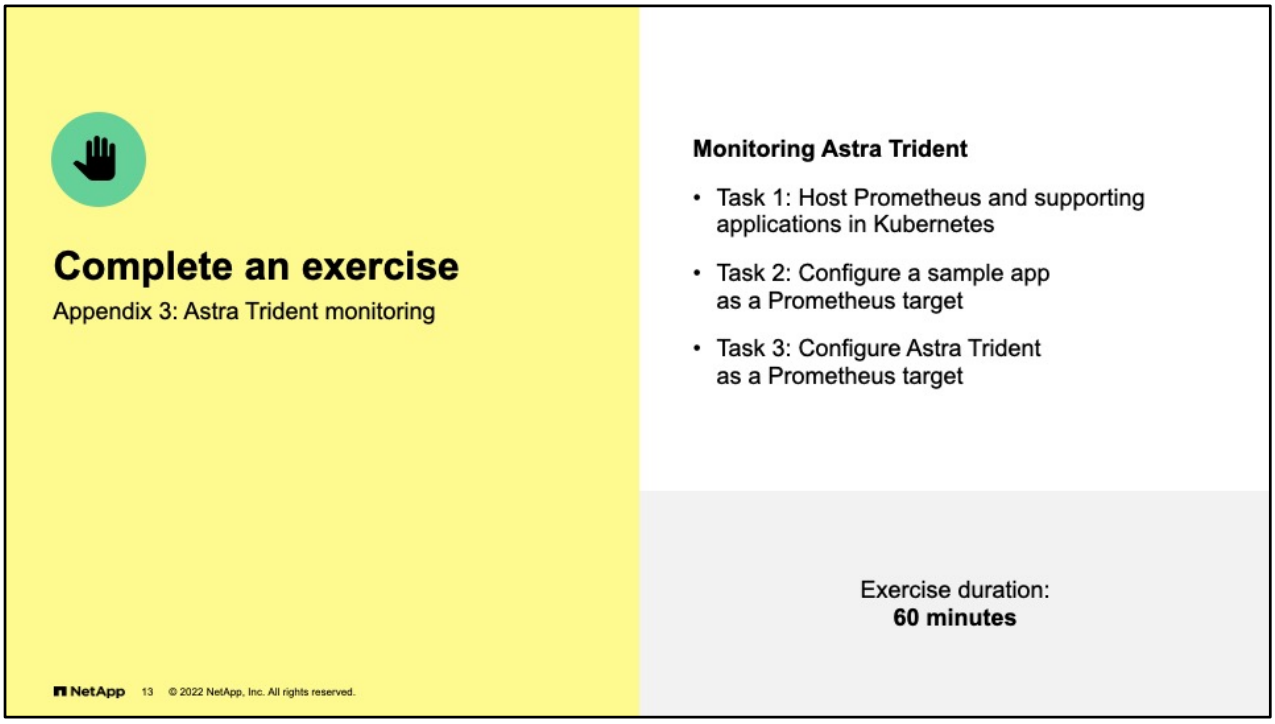
Instructor Notes

Student Notes



Instructor Notes

Student Notes



Instructor Notes

Student Notes

See the instructions in your Exercise Guide.