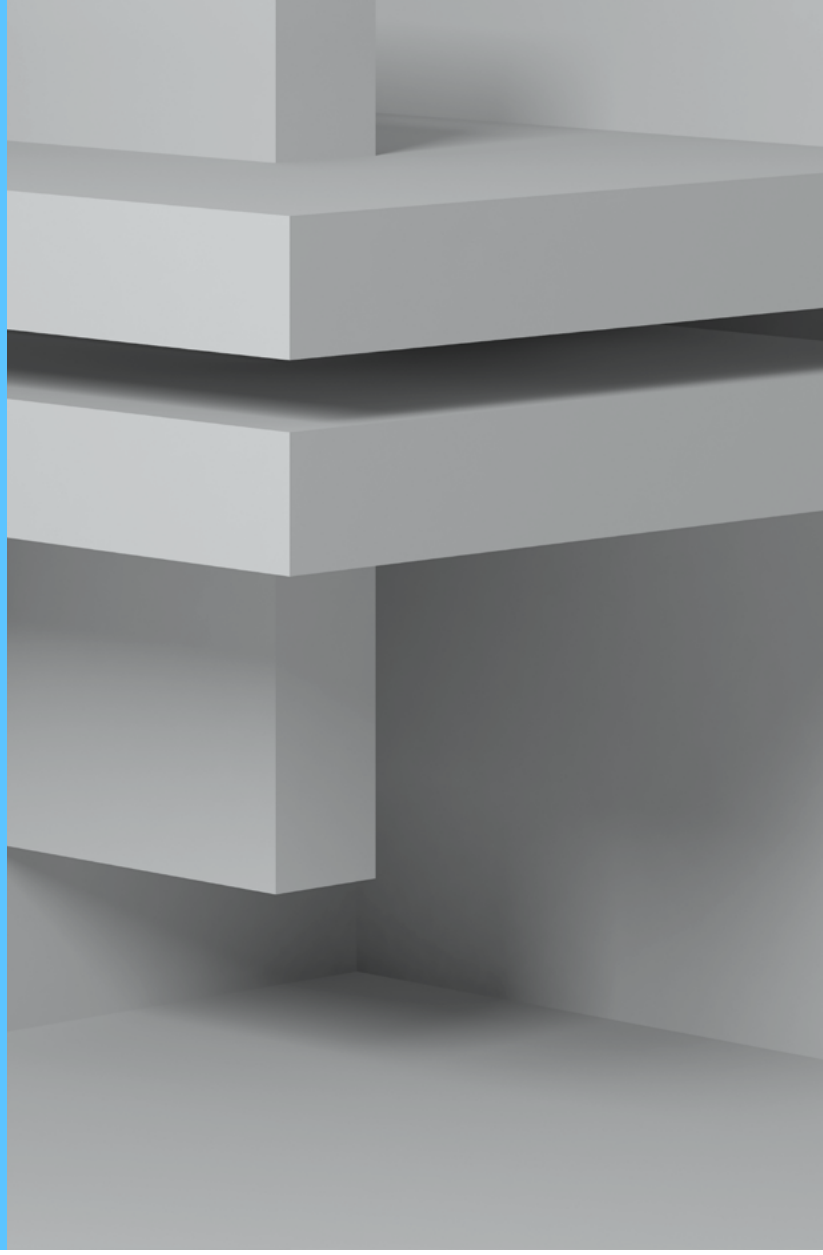
SOLUTION BRIEF

Application-aware data management for DataStax Enterprise with NetApp Astra

Backup / restore, cloning, and DR of Cassandra clusters in a Kubernetes environment.



NetApp | DataStax

Key benefits

- Realize faster time to value with DataStax Enterprise and Apache Cassandra with NetApp Astra when delivering business applications
- Clone DataStax Enterprise and Apache Cassandra clusters locally, and even migrate to different geographical locations, allowing for improved application unit and system testing
- Rapid recovery from a disaster, or going back to a point-in-time copy of a DataStax Enterprise or Apache Cassandra cluster

Executive Summary

This document details the business advantages of data management solutions (such as backup and recovery, business continuity, and active database cluster cloning) when using NetApp Astra for containerized DataStax Enterprise and Apache Cassandra database server clusters on Google Kubernetes Engine (GKE) clusters in Google Cloud.

NetApp's Cloud Volumes Service as persistent storage for containerized DataStax Enterprise and Apache Cassandra performs well. It also extends the data management benefits for which NetApp enterprise storage is known.

Use Cases

Simplified Day One operations: Automatic storage provisioning and storage class set-up

NetApp Astra simplifies and automates Day One operations for the DataStax Enterprise and Apache Cassandra clusters by easily registering the GKE cluster.

Dispelling the network-attached storage (NAS) myth

Network-attached storage performs at the same level as block storage but offers more features.

Immediate bootstrapping (recovery) of failed data nodes

Data nodes fail; operate a database service long enough and it will happen. With NetApp Cloud Volume Service in GCP and the Kubernetes platform, recovery from these events is nearly instantaneous, with zero or low impact on the production application.

DataStax Enterprise and Apache Cassandra Cluster Data Management with NetApp Astra

The ability to clone DataStax Enterprise and Apache Cassandra clusters locally, and even migrate to a different geographical location

To improve the effectiveness of application development unit and system tests, we need to test with realistically-sized and realistically-complex data. DataStax Enterprise and Apache Cassandra cluster cloning is a much-needed testing feature when running atop Kubernetes to accurately master and then replicate golden copies of an entire database server's contents. This capability aids in improved application system unit and system testing.

Database cluster portability is essential in such a situation to meet the business continuity requirements, whether it's on demand (a special analytics project or need) or in a disaster recovery scenario (an entire site or zone fails).

Rapid recovery from a disaster, or going back to a point-in-time copy of a DataStax Enterprise or Apache Cassandra cluster

Disaster can happen even when running DataStax Enterprise or Apache Cassandra clusters atop Kubernetes. This can be caused by a datacenter failure or human error. Business must continue to run regardless of the situation. NetApp Astra enables DataStax Enterprise or Apache Cassandra to recover quickly in these cases by using NetApp Astra's application-aware backups.

The NetApp Astra solution for DataStax Enterprise and Apache Cassandra offers the following key benefits:

- Automatic storage provisioning and storage class setup.
- Rich set of data management services, including data protection, business continuity and disaster recovery, active cloning, activity log, and more.
- Consistent data management UI.
- Clear visualization of data protection status.
- Simple data protection management.
- Seamless portability and migration.
- Simple and fully-managed.

About DataStax

DataStax is the leader in scale-out data and the company behind Apache Cassandra. DataStax is committed to Kubernetes as the cloud-native deployment and orchestration technology of choice for modern enterprises. DataStax Enterprise and Apache Cassandra end users benefit from rich data APIs, zero-downtime, and global scale with (Cassandra). When combined with Kubernetes and NetApp storage technology, enterprises are able to achieve transformational outcomes with the best-of-breed technologies for cloud-native operations. These benefits span across digital transformation initiatives of all types including new application development and application/service portfolio modernization.

DataStax also creates the open source DataStax Kubernetes Operator for Apache Cassandra, referenced in this article. This operator, a Kubernetes CRD, actively configures and manages DataStax Enterprise and Apache Cassandra clusters.

About NetApp Astra

NetApp Astra is a fully managed service, which makes it easier for enterprises to manage, protect, and move their data-rich containerized workloads running on Kubernetes within and across public clouds and on-premises. NetApp Astra provides persistent container storage which leverages NetApp's proven and expansive storage portfolio in the public cloud and on-premises deployments. It also offers a rich set of advanced application-aware data management functionality (including snapshot revert, backup and restore, activity log, and active cloning) for data protection, disaster recovery, data audit, and migration use-cases for modern apps.

About GKE

Google Kubernetes Engine (GKE) provides a managed environment for deploying, managing, and scaling your containerized applications using Google infrastructure. The GKE environment consists of multiple machines (specifically, Compute Engine instances) grouped together to form a cluster.

Detailed Solution Overview

This example uses two GKE clusters with four worker nodes each.

The first cluster (cloudwater-cluster-1) is provisioned in the europe-west2-a (London) region. The second cluster (pliny-cluster-1) is provisioned in the us-east4-a (Ashburn, Virginia) region.

Worker node machine type: N2-standard-16

Worker node image version: Ubuntu

Three-node Apache Cassandra Cluster version 3.11.7 using Helm to install the DataStax Kubernetes Operator for Apache Cassandra (available from https://github.com/datastax/cass-operator). The operator is installed in a namespace called cass-operator. The operator is Kubernetes-cluster-scoped.

The Apache Cassandra cluster nodes (pods) use Persistent Volume Claims (PVC) from NetApp Cloud Volumes Service in GCP.

NetApp Trident automatically provisions the Kubernetes persistent volume claims from NetApp Cloud Volumes Services for Apache Cassandra.

Simplified Day One operations: Automatic storage provisioning and storage class setup

NetApp Astra simplifies and automates Day One operations for the DataStax Enterprise and Apache Cassandra clusters by registering the GKE cluster:

- Automated installation of NetApp® Trident, NetApp's open-source Kubernetes storage provisioner and orchestrator.
- Backend configuration for Cloud Volume Service in GCP.
- Creation of storage classes for Cloud Volumes Service.
- Creation of a bucket in the cloud object store for saving application backups.
- Creation of a service account on the cluster for itself.

Dispelling the network attached storage (NAS) myth

The current version of the NetApp Astra software defaults to using NFS v3 for the storage classes it creates. For this test we created an additional storage class netapp-cvs-extreme-nfsv4 with mount options for NFSv4.1 as follows:

```
parameters:
  backendType: gcp-cvs
  selector:
 serviceLevel=extreme;storageClass=hardware
provisioner: csi.trident.netapp.io
reclaimPolicy: Retain
volumeBindingMode: WaitForFirstConsumer
mountOptions:
    - nfsvers=4.1
```

You will be able to select access-protocol-version (NFSv3 or NFSv4.1) for a storage class from the NetApp Astra UI in a future product update.

We used the cassandra-stress tool to measure latency and IOPS in this setup. When we ran a test plan with 800 threads of 1 million writes and reads, we observed the following results:

Write workload Results:

```
Op rate: 78,381 op/s  [WRITE: 78,381 op/s]
Partition rate: 78,381 pk/s  [WRITE: 78,381 pk/s]
Row rate: 78,381 row/s [WRITE: 78,381 row/s]
Latency mean: 9.8 ms [WRITE: 9.8 ms]
Latency median: 5.8 ms [WRITE: 5.8 ms]
Latency 95th percentile: 18.6 ms [WRITE: 18.6 ms]
Latency 99th percentile: 138.7 ms [WRITE: 138.7 ms]
Latency 99.9th percentile: 213.8 ms [WRITE: 213.8 ms]
Latency max: 237.5 ms [WRITE: 237.5 ms]
Total partitions: 1,000,000 [WRITE: 1,000,000]
Total errors: 0 [WRITE: 0]
Total GC count: 0
Total GC memory: 0.000 KiB
Total GC time: 0.0 seconds
Avg GC time: NaN ms
StdDev GC time: 0.0 ms
Total operation time: 00:00:12
```

Read workload Results:

```
op/s] Op rate: 82,861 op/s  [READ: 82,861
Partition rate: 82,861 pk/s  [READ: 82,861 pk/s]
Row rate: 82,861 row/s [READ: 82,861 row/s]
Latency mean: 8.9 ms [READ: 8.9 ms]
Latency median: 5.7 ms [READ: 5.7 ms]
Latency 95th percentile: 9.8 ms [READ: 9.8 ms]
Latency 99th percentile: 125.3 ms [READ: 125.3 ms]
Latency 99.9th percentile: 163.6 ms [READ: 163.6 ms]
Latency max: 177.1 ms [READ: 177.1 ms]
Total partitions: 1,000,000 [READ: 1,000,000]
Total errors: 0 [READ: 0]
Total GC count: 0
Total GC memory: 0.000 KiB
Total GC time: 0.0 seconds
Avg GC time: NaN ms
StdDev GC time: 0.0 ms
Total operation time: 00:00:12
```

Day 1 operations with Astra to deploy Apache Cassandra Cluster.

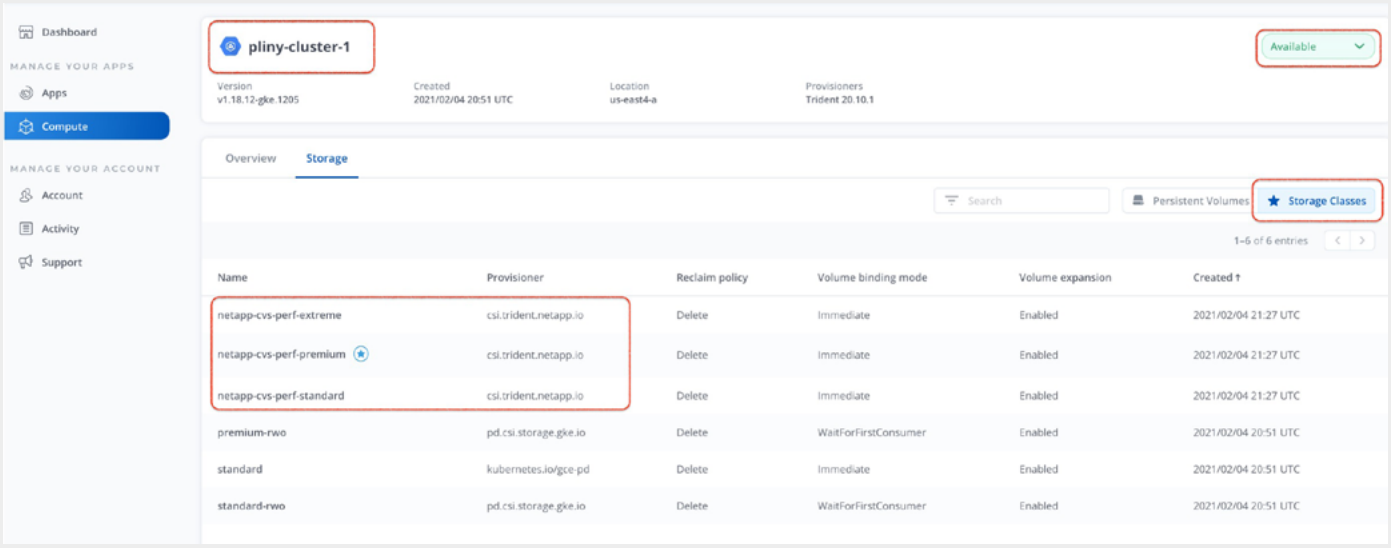


Fig. 1 Kubernetes cluster view.

```
bash-3.2$ kubectl get sc
NAME                          PROVISIONER             RECLAIMPOLICY   VOLUMEBINDINGMODE      ALLOWVOLUMEEXPANSION   AGE
netapp-cvs-extreme-nfsv4      csi.trident.netapp.io   Retain          WaitForFirstConsumer   true                   16h
netapp-cvs-perf-extreme (default)   csi.trident.netapp.io   Delete          Immediate              true                   16h
netapp-cvs-perf-premium       csi.trident.netapp.io   Delete          Immediate              true                   16h
netapp-cvs-perf-standard      csi.trident.netapp.io   Delete          Immediate              true                   16h
premium-rwo                   pd.csi.storage.gke.io   Delete          WaitForFirstConsumer   true                   16h
standard                      kubernetes.io/gce-pd    Delete          Immediate              true                   16h
standard-rwo                  pd.csi.storage.gke.io   Delete          WaitForFirstConsumer   true                   16h
bash-3.2$
bash-3.2$
bash-3.2$ helm install cass-operator ./C8_CassOperator --namespace=ns-cass-operator
NAME: cass-operator
```

Fig. 2 Deploying the DataStax Kubernetes Operator for Apache Cassandra.

```
bash-3.2$
bash-3.2$ kubectl apply -f D4_D1With3Nodes.yaml
cassandradatacenter.cassandra.datastax.com/dc1 created
bash-3.2$
bash-3.2$
bash-3.2$ kubectl get pods -n ns-cass-clus1
NAME                          READY   STATUS     RESTARTS   AGE
cluster1-dc1-default-sts-0    0/2     Init:0/1   0          28s
cluster1-dc1-default-sts-1    0/2     Pending    0          28s
cluster1-dc1-default-sts-2    0/2     Pending    0          28s
bash-3.2$
bash-3.2$ kubectl get pvc -n ns-cass-clus1
NAME                                  STATUS   VOLUME                                     CAPACITY   ACCESS MODES   STORAGEC
LASS             AGE
server-data-cluster1-dc1-default-sts-0   Bound    pvc-363d73ea-4d83-49b9-9ee4-e82c060aac55   1Ti        RWO            netapp-c
vs-perf-extreme   52s
server-data-cluster1-dc1-default-sts-1   Bound    pvc-adae9b17-a14a-411c-9360-a331f24fef39   1Ti        RWO            netapp-c
vs-perf-extreme   52s
server-data-cluster1-dc1-default-sts-2   Bound    pvc-08af9da6-d051-43c7-816a-c47b2f56fa1f   1Ti        RWO            netapp-c
vs-perf-extreme   52s
bash-3.2$
bash-3.2$
```

Fig. 3 Deploying the Apache Cassandra cluster.

Immediate bootstrapping (recovery) of failed data nodes

Anyone with experience operating database services knows that nodes can and will eventually fail. With NetApp Cloud Volume Service in GCP, recovery from these events is nearly instantaneous, with zero or low impact to the production application.

We killed (deleted the node pool) one of the worker nodes in availability zone us-east4-c (The Apache Cassandra nodes are running in GKE worker nodes from three different availability zones) to simulate failure of an Apache Cassandra cluster node. The GKE scheduler detects the pod failure and automatically schedules the downed Apache Cassandra pod to another worker node. In this case, there is no worker node available with enough CPU resources. A new node pool is created on the GCP availability zone us-east4-c with the same node type as the deleted pool. The Apache Cassandra failed node is brought up on the new worker node and completes the initialization. The new pod (Apache Cassandra node) is bound to the original Kubernetes persistent volume claim. Data created during the event will be resynced from whichever Apache Cassandra nodes in the cluster have the most up-to-date data.

```
bash-3.2$ gcloud container node-pools list --cluster=pliny-cluster-1  --region=us-east4
NAME          MACHINE_TYPE    DISK_SIZE_GB  NODE_VERSION
default-pool  n2-standard-16  100           1.18.12-gke.1210
pool-1        n2-standard-16  100           1.18.12-gke.1210
pool-2        n2-standard-16  100           1.18.12-gke.1210
pool-3        e2-medium       100           1.18.12-gke.1210
bash-3.2$
bash-3.2$ gcloud container node-pools delete  pool-2 --cluster=pliny-cluster-1 --region=us-east4
The following node pool will be deleted.
[pool-2] in cluster [pliny-cluster-1] in [us-east4]

Do you want to continue (Y/n)?  y

Deleting node pool pool-2...done.
Deleted [https://container.googleapis.com/v1/projects/astra-tme-sandbox/zones/us-east4/clusters/pliny-cluster-1/nodePools/pool-2].
bash-3.2$
bash-3.2$
bash-3.2$ gcloud container node-pools list --cluster=pliny-cluster-1  --region=us-east4
NAME          MACHINE_TYPE    DISK_SIZE_GB  NODE_VERSION
default-pool  n2-standard-16  100           1.18.12-gke.1210
pool-1        n2-standard-16  100           1.18.12-gke.1210
pool-3        e2-medium       100           1.18.12-gke.1210
bash-3.2$
bash-3.2$
bash-3.2$
bash-3.2$ kubectl get sts -n ns-cass-clus1
NAME                      READY   AGE
cluster1-dc1-default-sts  2/3     96m
```

Fig. 4 Deleting the GKE cluster node pool to simulate the Apache Cassandra Cluster node failure.

```
bash-3.2$ kubectl get pods -n ns-cass-clus1
NAME                        READY   STATUS    RESTARTS   AGE
cluster1-dc1-default-sts-0  2/2     Running   0          11m
cluster1-dc1-default-sts-1  2/2     Running   0          100m
cluster1-dc1-default-sts-2  2/2     Running   0          100m
bash-3.2$
bash-3.2$ kubectl -n ns-cass-clus1 exec -it -c cassandra cluster1-dc1-default-sts-1 -- nodetool status
Datacenter: dc1
===============
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address     Load       Tokens       Owns (effective)  Host ID                               Rack
UN  10.72.1.9   6.81 GiB   1            71.2%             16ad2d16-08e3-431e-9988-e6fdc11f97a3  default
UN  10.72.2.2   889.74 MiB  1            9.1%              eee56941-cf7a-4500-8fce-352bce3a1e81  default
UN  10.72.0.4   1.87 GiB   1            19.7%             174232e0-ff79-4911-bbfd-997101c49bec  default

bash-3.2$ kubectl -n ns-cass-clus1 exec -it -c cassandra cluster1-dc1-default-sts-0 -- nodetool status
Datacenter: dc1
===============
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address     Load       Tokens       Owns (effective)  Host ID                               Rack
UN  10.72.1.9   6.81 GiB   1            71.2%             16ad2d16-08e3-431e-9988-e6fdc11f97a3  default
UN  10.72.2.2   890.08 MiB  1            9.1%              eee56941-cf7a-4500-8fce-352bce3a1e81  default
UN  10.72.0.4   1.87 GiB   1            19.7%             174232e0-ff79-4911-bbfd-997101c49bec  default

bash-3.2$ kubectl get pods -n ns-cass-clus1 -o wide
NAME                        READY   STATUS    RESTARTS   AGE    IP          NODE                                                NOMINATED NODE
SS GATES
cluster1-dc1-default-sts-0  2/2     Running   0          89m    10.72.2.2   gke-pliny-cluster-1-newpool-d5df855d-cf61            <none>
cluster1-dc1-default-sts-1  2/2     Running   0          179m   10.72.1.9   gke-pliny-cluster-1-default-pool-4d9ca9e8-9g64   <none>
cluster1-dc1-default-sts-2  2/2     Running   0          179m   10.72.0.4   gke-pliny-cluster-1-pool-1-dce04b1e-qrhw         <none>
```

Fig. 5 Apache Cassandra cluster node is up after the GKE worker node is available.

DataStax Enterprise and Apache Cassandra Data Management with NetApp Astra

We create a new database table and insert sample data into the Apache Cassandra database to demonstrate the data management capabilities of NetApp Astra.

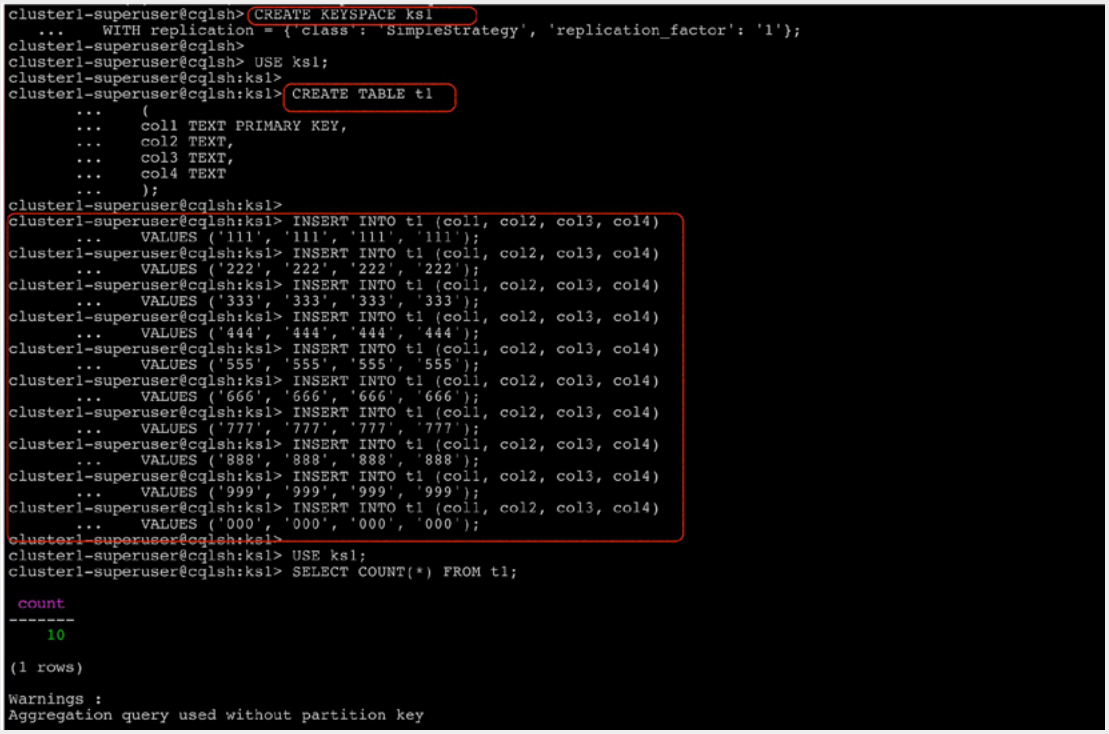


Fig. 6 Create the database and table, then insert data.

Apache Cassandra clusters can be installed in any custom namespace. You can have multiple Apache Cassandra clusters within the Kubernetes cluster, but only one DataStax Kubernetes Operator for Apache Cassandra is required. NetApp Astra discovers the operator, any Apache Cassandra clusters, and their associated namespaces. As each Apache Cassandra cluster can have multiple pods, (Apache Cassandra cluster nodes), the recommended way to manage the application in NetApp Astra is by choosing the namespace hosting Apache Cassandra cluster as a management unit. Using NetApp Astra to manage your application addresses critical Day Two challenges like application-aware snapshots, backups, and disaster recovery.

We performed the following steps to leverage the application-aware data management features for Apache Cassandra Cluster on Kubernetes with NetApp Astra:

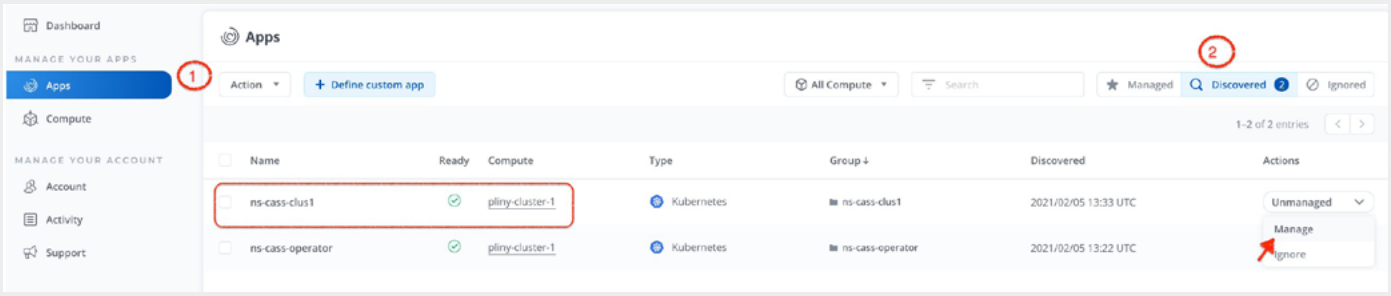


Fig. 7 Managing the Apache Cassandra application.

After the Apache Cassandra application is registered as a managed application with NetApp Astra, NetApp Astra can take application consistent snapshots, backups, and clones of that application, its Kubernetes resources, and its associated Persistent Volumes.

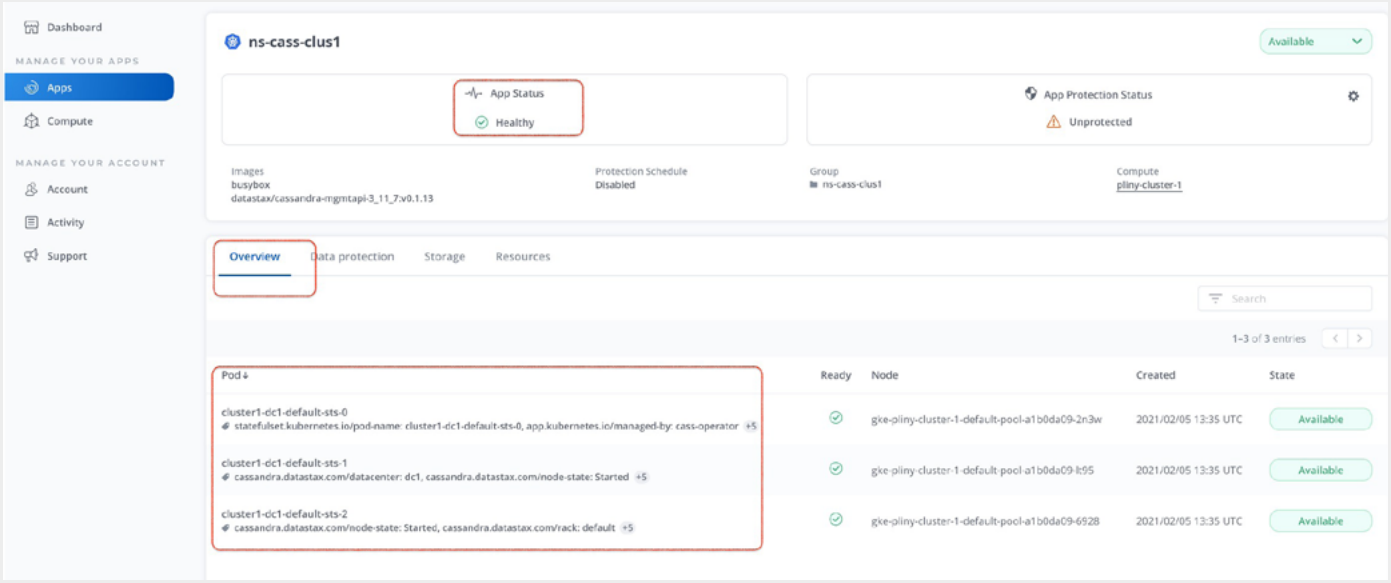


Fig. 8 Managed application overview.

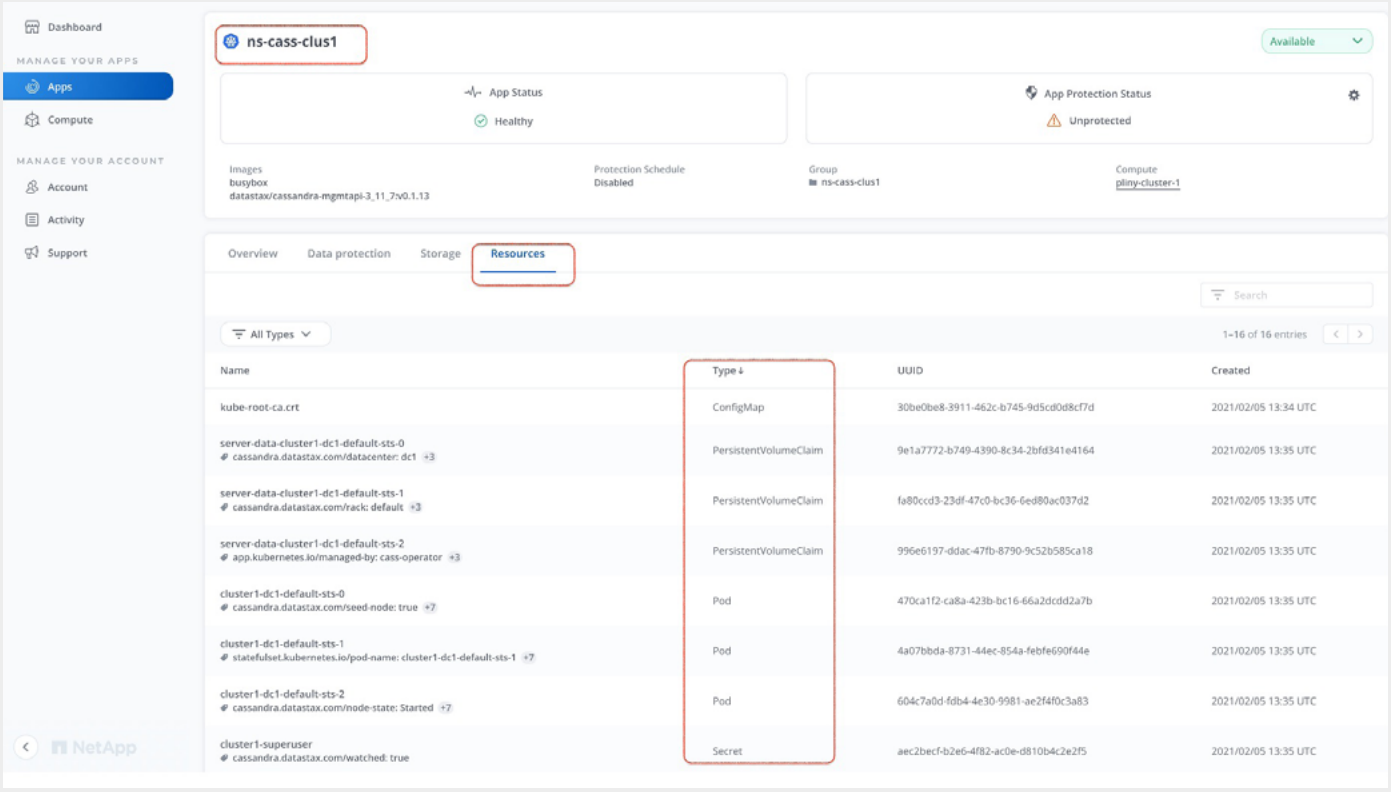


Fig. 9 Kubernetes resources of the Apache Cassandra cluster.

All the data generated by the DataStax Enterprise and Apache Cassandra database nodes can be automatically protected with snapshots and backups. NetApp Astra snapshots and backups preserve the application state, its Kubernetes resources, and its volumes in one easily-manageable unit. All application backups are stored in an object store.

NetApp Astra supports both on-demand and scheduled snapshots and backups. When taking on-demand backups, you have the option to choose any existing snapshot. Otherwise, the backup will be created from the application's current state.

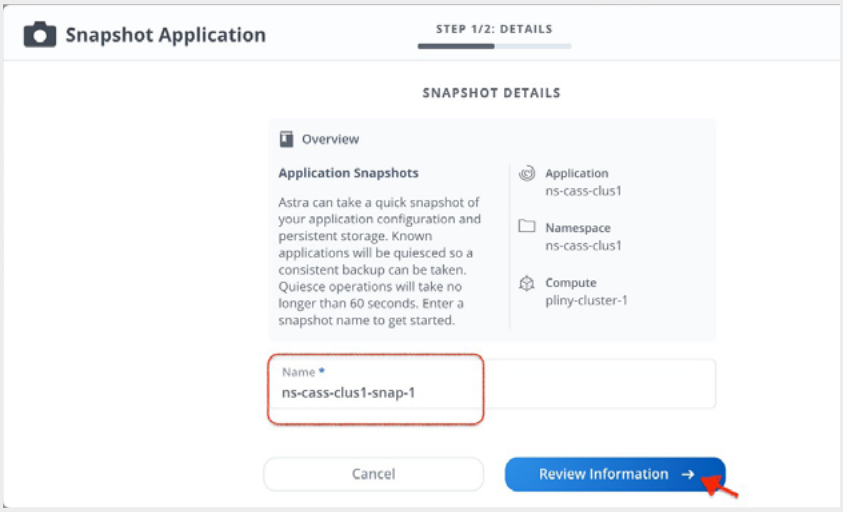
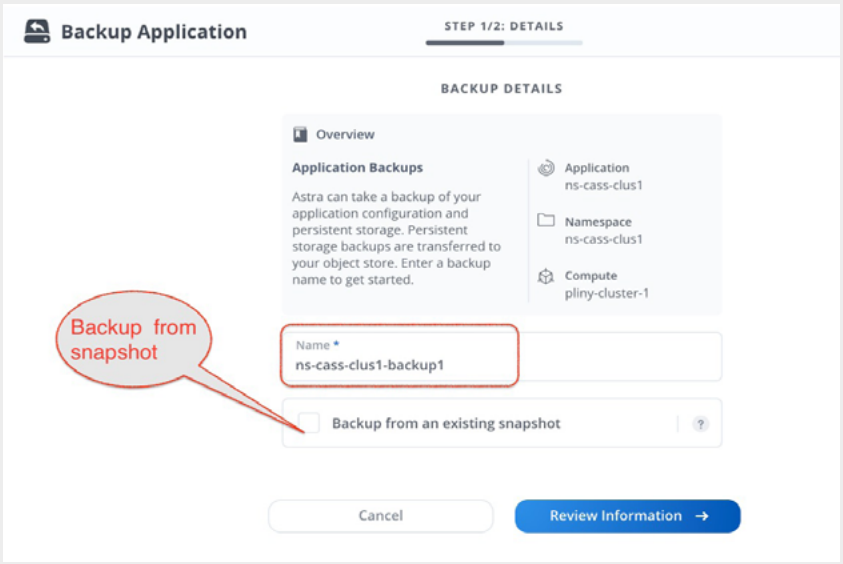


Fig. 10 On-demand application snapshot.



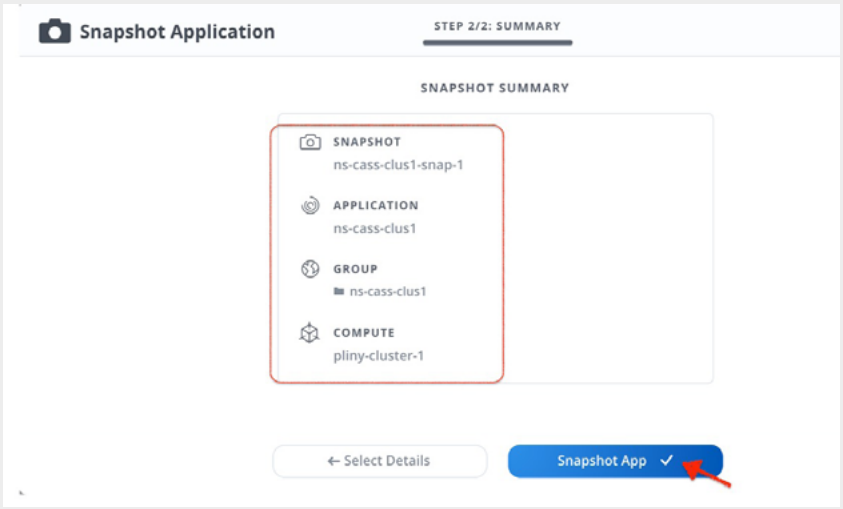
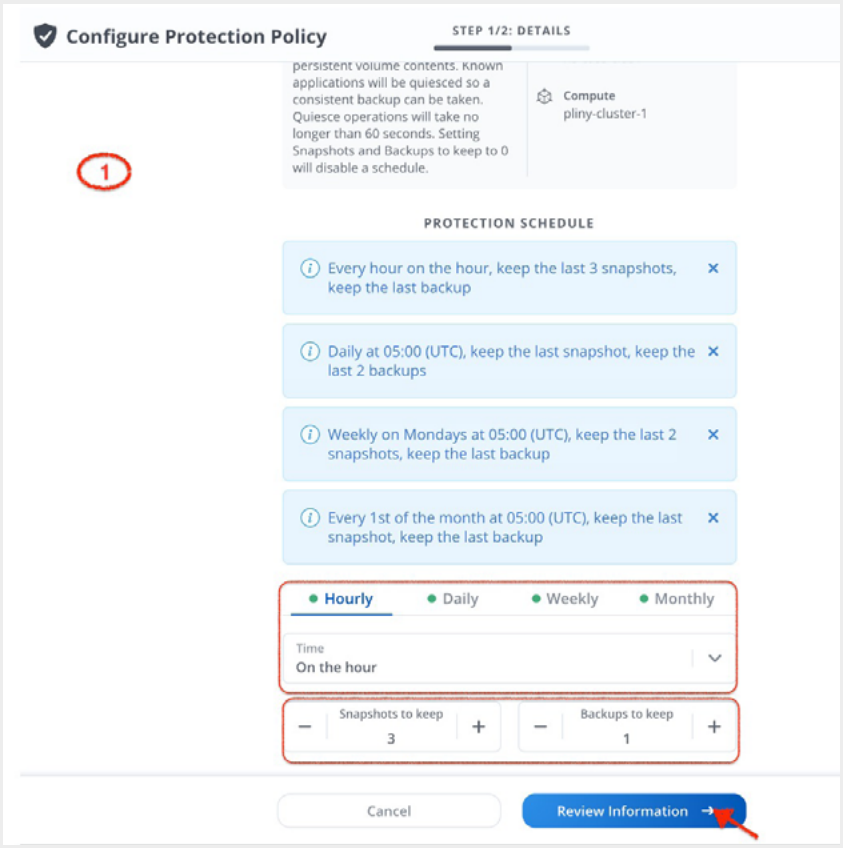


Fig. 11 and 12 On-demand application backup.

Set up a snapshot and backup schedule for the volume and all of its associated Kubernetes objects.



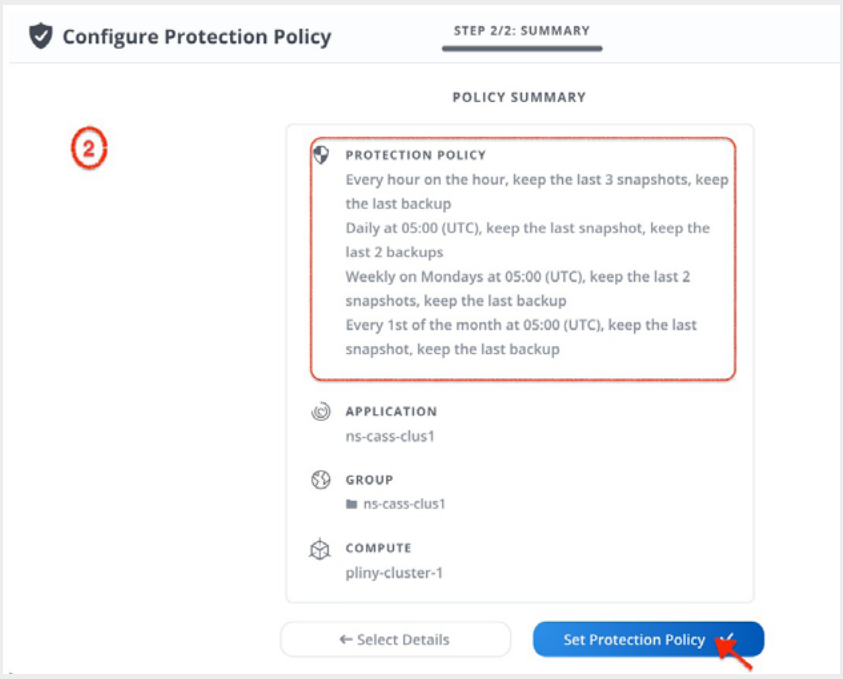


Fig. 13 and 14 Configure protection policy.

After reviewing the information, we set the protection policy. NetApp Astra automatically takes snapshots and backups based on the schedule and follows the defined retention policy.

Migrating Apache Cassandra to another Kubernetes namespace within the same Kubernetes cluster

After a successful backup, the Apache Cassandra database cluster is protected against disasters like losing the Kubernetes cluster, or a human error like deleting a namespace or corrupting a database with a bad SQL/CQL (Apache Cassandra) operation. You can use the clone option to redeploy Apache Cassandra to a new namespace, either within the same cluster or in a new cluster.

For example, suppose that your team needs a way to test the production database for a new use case without interrupting the production Apache Cassandra cluster. NetApp Astra can clone the Apache Cassandra database cluster to another namespace within the same Kubernetes cluster.

In our setup, Apache Cassandra is running on the pliny-cluster-1 cluster in the us-east4 (Virginia) region. You can use the clone option from the NetApp Astra UI and either provide the details for new namespace or application name or use the automatic namespace and application names created by NetApp Astra. You can also select an existing snapshot or backup to go back to a point- in-time copy of the Apache Cassandra application.

```
Connected to cluster1 at 127.0.0.1:9042.
[cqlsh 5.0.1 | Cassandra 3.11.7 | CQL spec 3.4.4 | Native protocol v4]
Use HELP for help.
cluster1-superuser@cqlsh> desc keyspaces;

ks1  system_schema  system_auth  system  system_distributed  system_traces

cluster1-superuser@cqlsh> use ks1
   ... ;
cluster1-superuser@cqlsh:ks1> select * from t1;

 col1 | col2 | col3 | col4
------+------+------+------
  888 |  888 |  888 |  888
  000 |  000 |  000 |  000
  444 |  444 |  444 |  444
  999 |  999 |  999 |  999
  777 |  777 |  777 |  777
  111 |  111 |  111 |  111
  666 |  666 |  666 |  666
  333 |  333 |  333 |  333
  555 |  555 |  555 |  555
  222 |  222 |  222 |  222

(10 rows)
```

Fig. 15 Current state of the Apache Cassandra application.

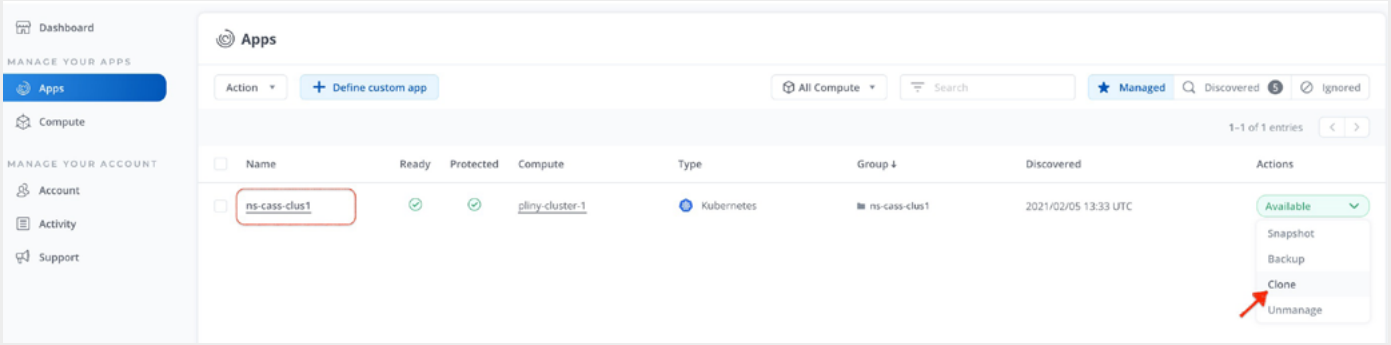


Fig. 16 Migrating Apache Cassandra from the current state.

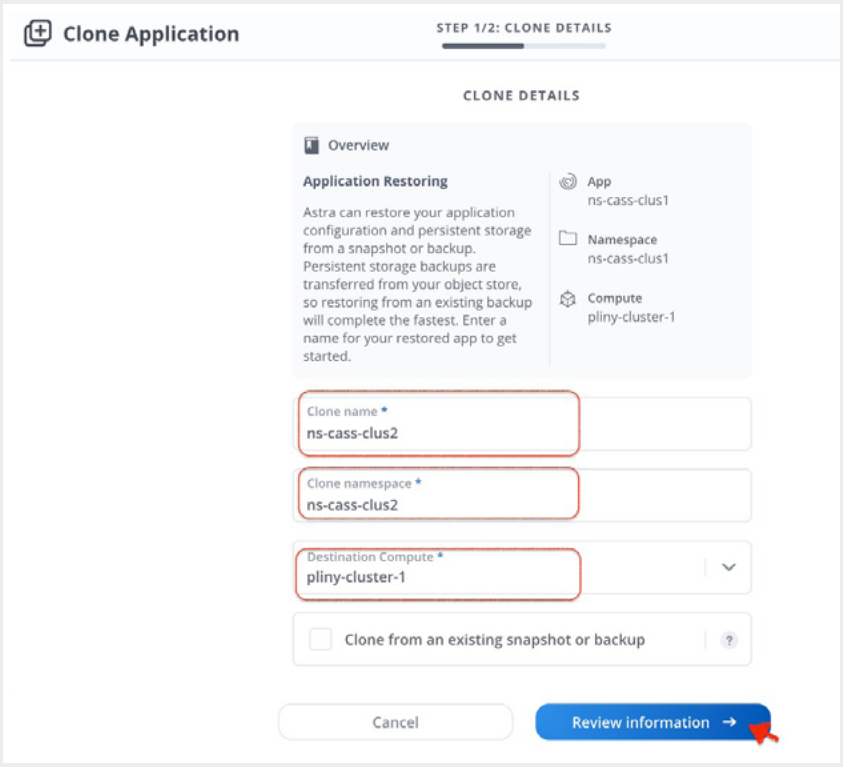


Fig. 17 Migrating Apache Cassandra from the current state.

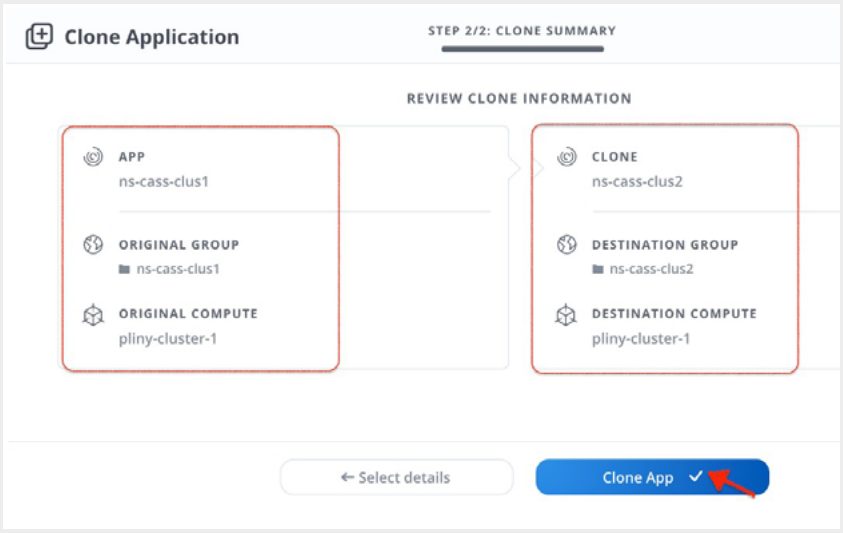


Fig. 18 Migrating Apache Cassandra from the current state.

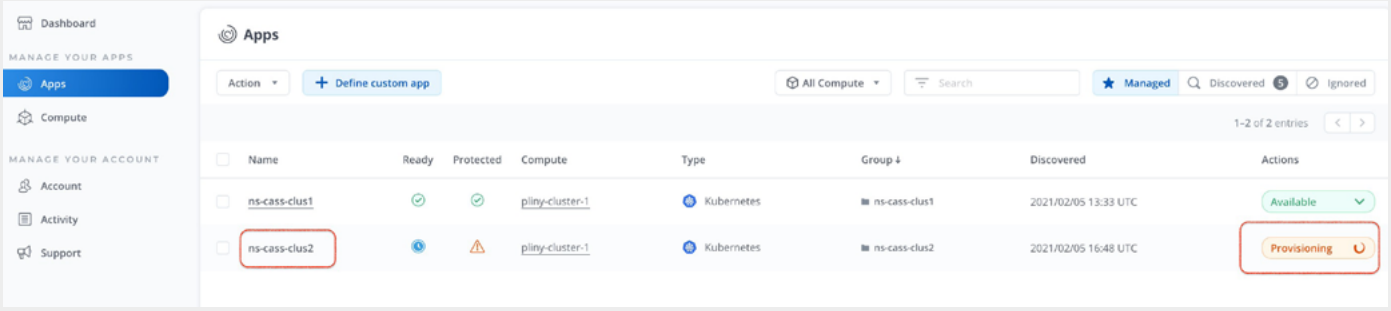


Fig. 19 Provisioning Apache Cassandra cluster in destination namespace.

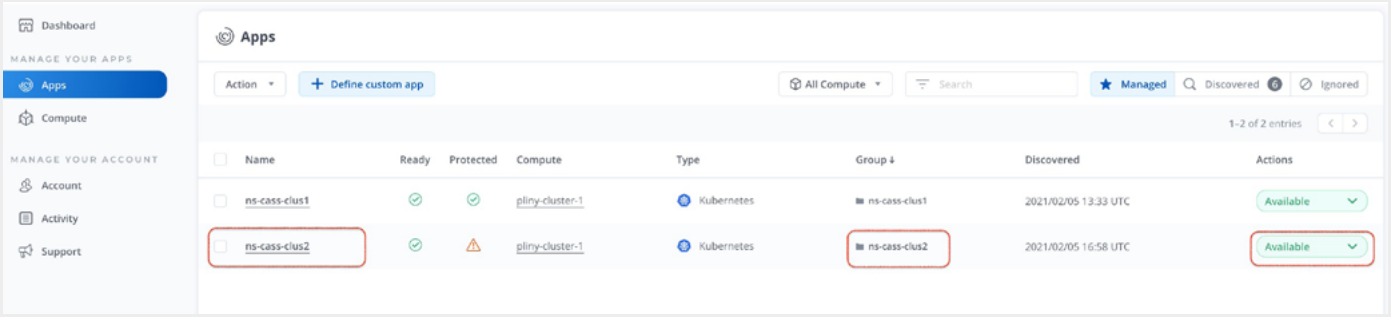


Fig. 20 Cloned Apache Cassandra cluster in destination namespace.

```
cqlsh'ing into: ns-cass-clus2/cluster1-dc1-default-sts-0
cluster1-dc1-default-sts-1
cluster1-dc1-default-sts-2
   Username: cluster1-superuser
   Password: E62ejU8GKAEAYJNjtHgXmZRWXINR7pORp4dj2mgFfkIQHC3v_FVMFQ


Connected to cluster1 at 127.0.0.1:9042.
[cqlsh 5.0.1 | Cassandra 3.11.7 | CQL spec 3.4.4 | Native protocol v4]
Use HELP for help.
cluster1-superuser@cqlsh> desc keyspaces;

ks1  system_schema  system_auth  system  system_distributed  system_traces

cluster1-superuser@cqlsh> use ks1;
cluster1-superuser@cqlsh:ks1>
cluster1-superuser@cqlsh:ks1> select * from t1;

 col1 | col2 | col3 | col4
------+------+------+------
  888 |  888 |  888 |  888
  000 |  000 |  000 |  000
  444 |  444 |  444 |  444
  999 |  999 |  999 |  999
  777 |  777 |  777 |  777
  111 |  111 |  111 |  111
  666 |  666 |  666 |  666
  333 |  333 |  333 |  333
  555 |  555 |  555 |  555
  222 |  222 |  222 |  222

(10 rows)
cluster1-superuser@cqlsh:ks1>
```

Fig. 21 Validate Cassandra cluster database in destination namespace.

The clone operation is initiated from the current state of the Apache Casandra cluster. NetApp Astra initiates an application-aware snapshot for the Apache Cassandra cluster. When the snapshot creation is successfully completed, NetApp Astra restores the snapshot into the new namespace provided. The DataStax Kubernetes Operator for Apache Cassandra running in the Kubernetes cluster, will identify the new Apache Cassandra cluster clone[1], and cluster pods can now talk to the operator.

Apache Cassandra clusters need to be backed up and restored with the CassandraDatacenter object within Kubernetes. After migrating the Apache Cassandra cluster, the DataStax Kubernetes Operator for Apache Cassandra will perform the actions necessary to verify that the Apache Cassandra database cluster is brought to a full online and operational state.

1 An issue in the version of NetApp Astra we used for this solution guide prevented the server-system-logger container from coming online after the clone operation. As a workaround, we first deleted the Apache Cassandra stateful set created in the clone destination. We then deployed the Apache Cassandra cluster in the same namespace created for the clone operation.

Migrating Apache Cassandra cluster to a remote Kubernetes cluster.

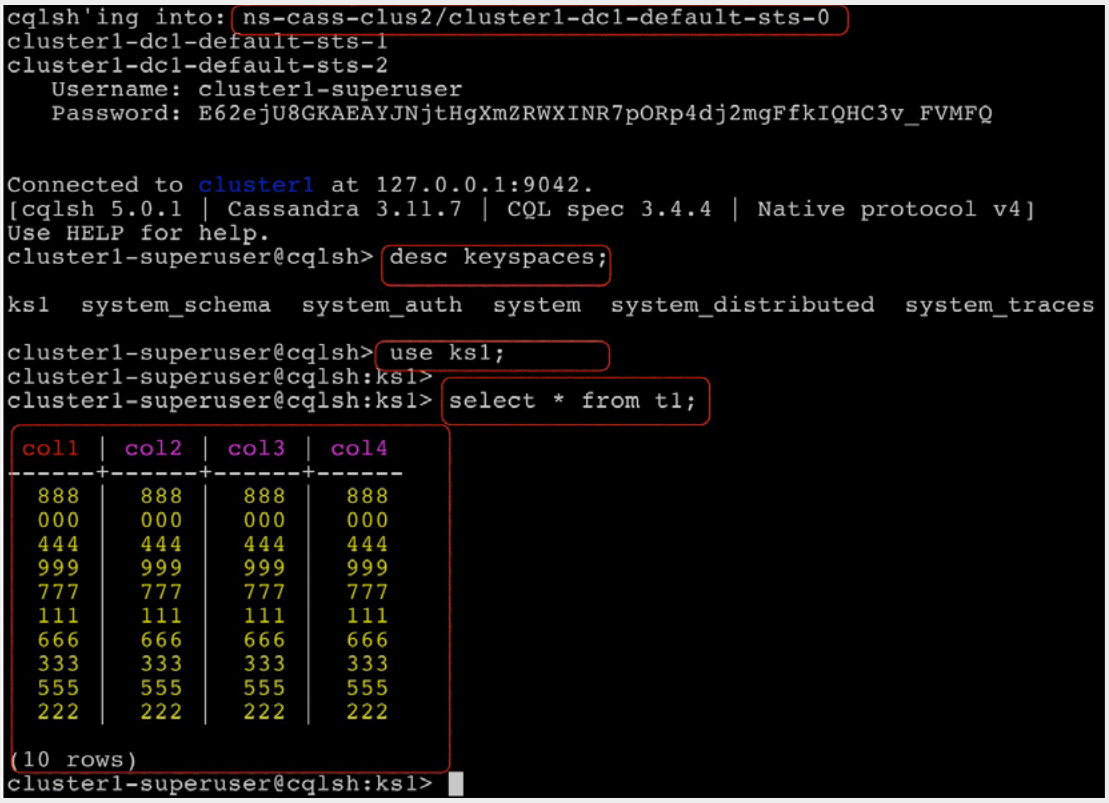


Fig. 22 Current state of the Apache Cassandra application.

In this section, we clone a DataStax Enterprise or Apache Cassandra database server cluster from an existing Kubernetes cluster cloudwater-cluster-1 located in GCP region Europe-west2 (London) from the current state. We could also clone from an existing backup or snapshot.

When cloning from the current state, NetApp Astra creates an application-aware backup and restores the Apache Cassandra cluster to the destination cluster. As part of the backup, NetApp Astra first creates an application-aware snapshot, then copies that snapshot into a Google Object store. When the copy operation completes, NetApp Astra uses that backup to migrate to the destination cluster. This brings up a new instance of Apache Cassandra running as the same state in the source cluster.

NetApp Astra simplifies the operation by using the same user interface and functionality for local clone and a remote migration.

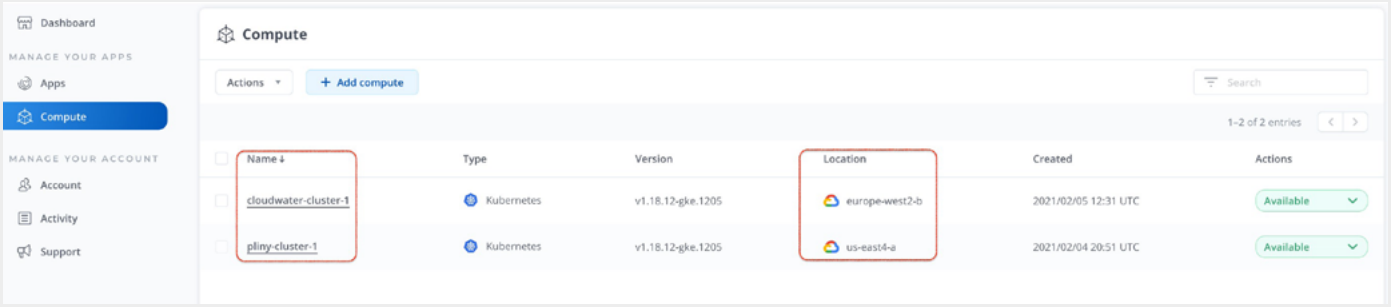


Fig. 23 Available Kubernetes clusters.

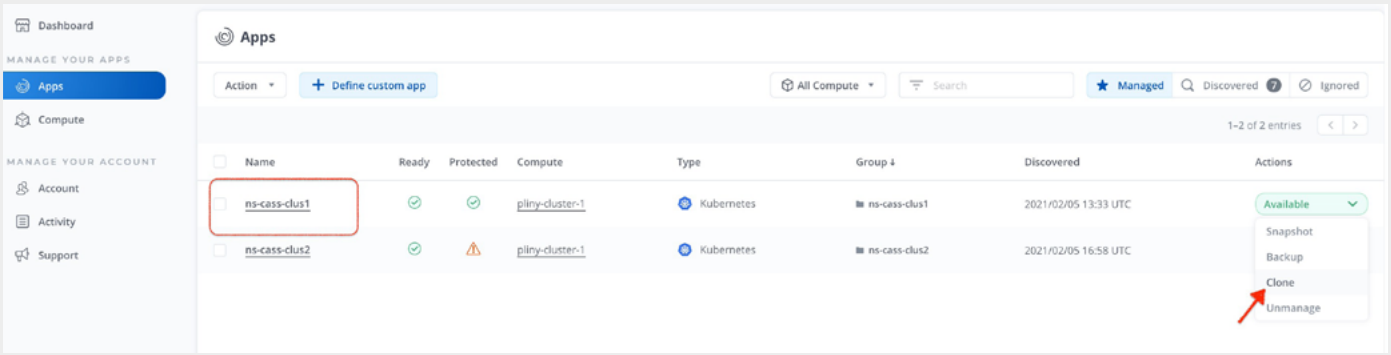


Fig. 24 Migrating Apache Cassandra from the current state.

Astra provisions a new Apache Cassandra clone in the destination cluster and automatically manages the application.

```
bash-3.2$ kubectl get pods -n ns-cass-remote -w
NAME                          READY   STATUS     RESTARTS   AGE
cluster1-dc1-default-sts-0    0/2     Init:0/1   0          24s
cluster1-dc1-default-sts-1    0/2     Pending    0          24s
cluster1-dc1-default-sts-2    0/2     Pending    0          24s
cluster1-dc1-default-sts-1    0/2     Pending    0          24s
cluster1-dc1-default-sts-1    0/2     Init:0/1   0          24s
cluster1-dc1-default-sts-0    0/2     Init:0/1   0          36s
cluster1-dc1-default-sts-0    0/2     PodInitializing   0          38s
cluster1-dc1-default-sts-2            Pending           0          38s
cluster1-dc1-default-sts-2    0/2     Init:0/1          0          38s
cluster1-dc1-default-sts-0    1/2     Running           0          39s
cluster1-dc1-default-sts-1    0/2     Init:0/1          0          41s
^Cbash-3.2$
bash-3.2$
bash-3.2$ kubectl get pods -n ns-cass-remote -w
NAME                          READY   STATUS    RESTARTS   AGE
cluster1-dc1-default-sts-0    1/2     Running   0          55s
cluster1-dc1-default-sts-1    1/2     Running   0          55s
cluster1-dc1-default-sts-2    1/2     Running   0          55s
cluster1-dc1-default-sts-0    2/2     Running   0          80s
cluster1-dc1-default-sts-0    2/2     Running   0          80s
cluster1-dc1-default-sts-2    1/2     Running   0          81s

^Cbash-3.2$ kubectl get pods -n ns-cass-remote -w
NAME                          READY   STATUS    RESTARTS   AGE
cluster1-dc1-default-sts-0    2/2     Running   0          111s
cluster1-dc1-default-sts-1    1/2     Running   0          111s
cluster1-dc1-default-sts-2    1/2     Running   0          111s
cluster1-dc1-default-sts-2    2/2     Running   0          2m22s
cluster1-dc1-default-sts-2    2/2     Running   0          2m22s
cluster1-dc1-default-sts-2    2/2     Running   0          2m22s
cluster1-dc1-default-sts-1    1/2     Running   0          2m22s
```

Fig. 25 Provisioning a new Apache Cassandra instance in the destination cluster.

After the migration[2], the Apache Cassandra application has the same Kubernetes resources and data as the source cluster.

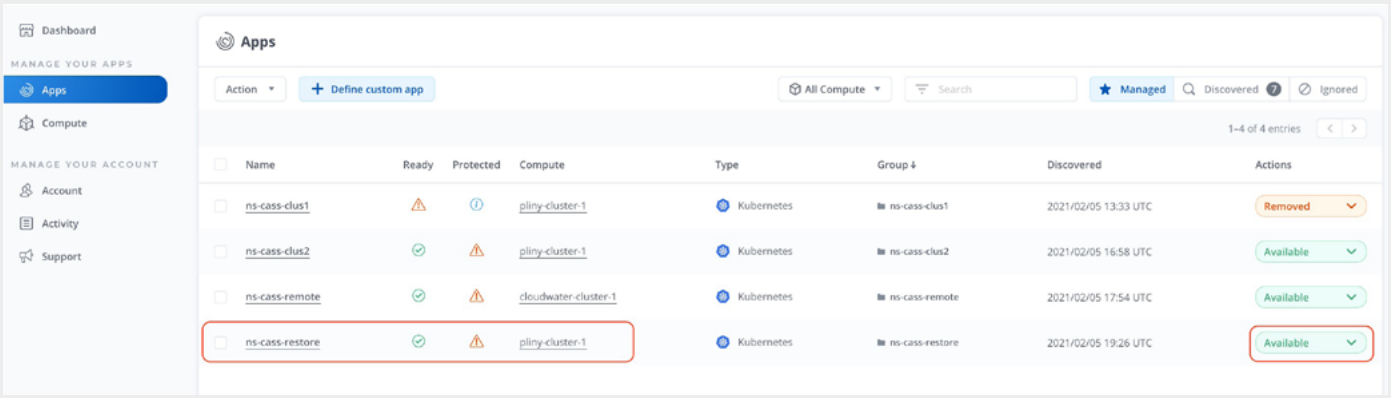


Fig. 26 Apache Cassandra application after migration.

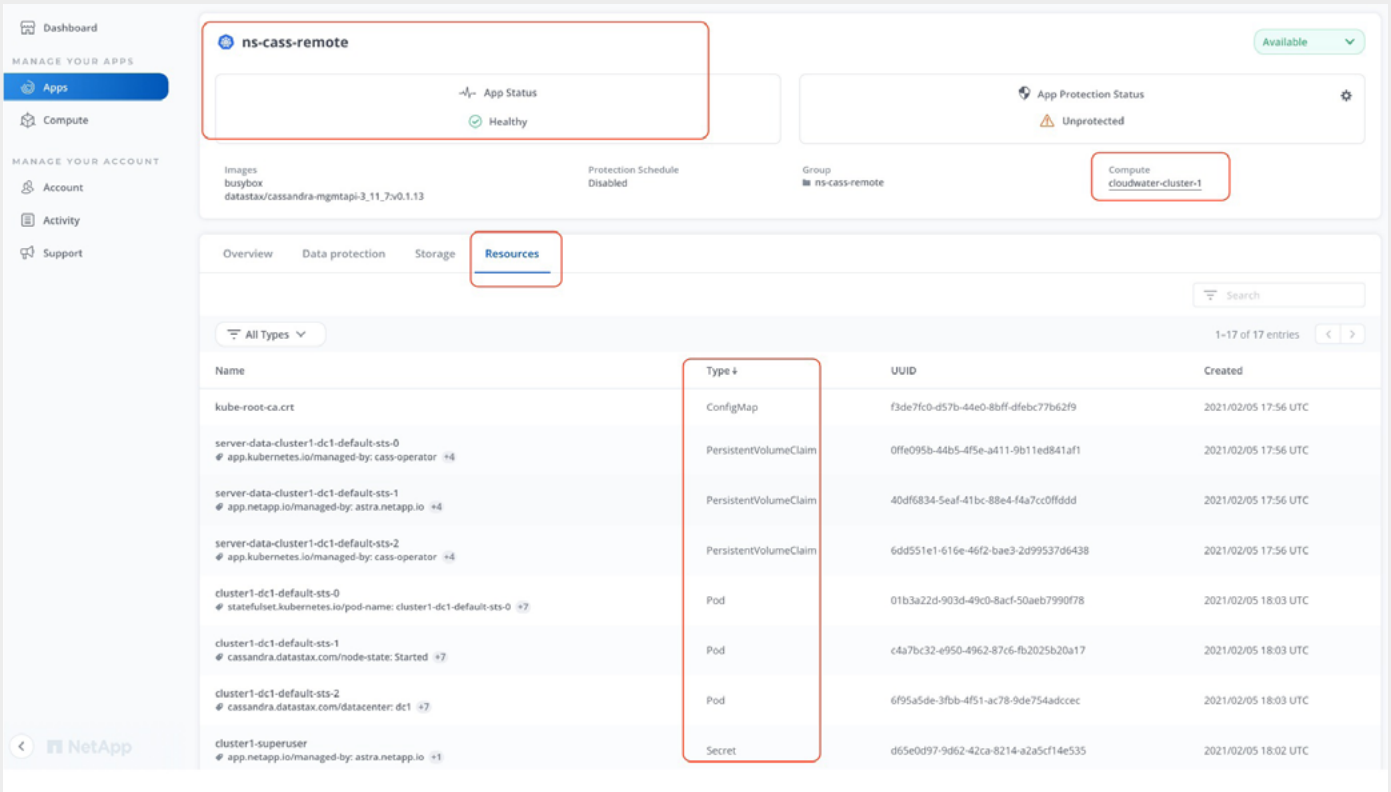


Fig. 27 Apache Cassandra application after migration.

2 An issue in the version of NetApp Astra we used for this solution guide prevented the Apache Cassandra pods from initializing after a migration. As a workaround, we deployed the DataStax Kubernetes Operator for Apache.

Cassandra in the destination cluster manually and deployed the DataStax Enterprise or Apache Cassandra cluster in the same namespace that was created by NetApp Astra for migration.

```
cqlsh'ing into: ns-cass-remote/cluster1-dc1-default-sts-0
cluster1-dc1-default-sts-1
cluster1-dc1-default-sts-2
   Username: cluster1-superuser
   Password: E62ejU8GKAEAYJNjtHgXmZRWXINR7pORp4dj2mgFfkIQHC3v_FVMFQ


Connected to cluster1 at 127.0.0.1:9042.
[cqlsh 5.0.1 | Cassandra 3.11.7 | CQL spec 3.4.4 | Native protocol v3]
Use HELP for help.
cluster1-superuser@cqlsh> desc keyspaces;

ks1  system_schema  system_auth  system  system_distributed  system_traces

cluster1-superuser@cqlsh> use ks1;
cluster1-superuser@cqlsh:ks1>
cluster1-superuser@cqlsh:ks1>
cluster1-superuser@cqlsh:ks1> select * from t1;

 col1 | col2 | col3 | col4
------+------+------+------
  888 |  888 |  888 |  888
  000 |  000 |  000 |  000
  444 |  444 |  444 |  444
  999 |  999 |  999 |  999
  777 |  777 |  777 |  777
  111 |  111 |  111 |  111
  666 |  666 |  666 |  666
  333 |  333 |  333 |  333
  555 |  555 |  555 |  555
  222 |  222 |  222 |  222

(10 rows)
```

Fig. 28 Verify Apache Cassandra database after migration.

Rapid recovery from a disaster or going back to a point-in-time copy of DataStax Enterprise and Apache Cassandra Clusters

Accidently deleting the wrong Kubernetes namespace can bring down the production database cluster and delete all its Kubernetes resources. Having an application-consistent backup in this case is significant and using it to restore will bring the business back to life. NetApp Astra provides instantaneous restore of your DataStax Enterprise or Apache Cassandra cluster and application in a few clicks.

Earlier in this document we cloned an Apache Cassandra cluster across Kubernetes clusters. Here we are using a snapshot or backup within NetApp Astra to go back to a point-in-time, application-consistent state.

Restoring DataStax Enterprise or Apache Cassandra from a backup

With NetApp Astra, after a successful backup, the DataStax Enterprise or Apache Cassandra database server cluster is protected against disasters. Deleting a namespace will result in losing all the application data and Kubernetes resources (including persistent volume claims and snapshots). Use the restore option from the most recent backup taken by NetApp Astra to redeploy the DataStax Enterprise or Apache Cassandra cluster either to a new namespace within the same Kubernetes cluster or to a new Kubernetes cluster.

In this example we use the most recent backup (ns-cass-clus1-backup2) to restore Apache Cassandra to a different namespace in the same cluster (*pliny-cluster-1*). Backup ns-cass-clus1-backup2 is taken after adding keyspace ks2 and a table t2 with data.

```
Use HELP for help.
cluster1-superuser@cqlsh> DROP KEYSPACE IF EXISTS ks2;
cluster1-superuser@cqlsh>
cluster1-superuser@cqlsh> CREATE KEYSPACE ks2
   ...     WITH replication = {'class': 'SimpleStrategy', 'replication_factor': '1'};
cluster1-superuser@cqlsh>
cluster1-superuser@cqlsh> USE ks2;
cluster1-superuser@cqlsh:ks2>
cluster1-superuser@cqlsh:ks2> CREATE TABLE t2
       ...     (
       ...     col1 TEXT PRIMARY KEY,
       ...     col2 TEXT,
       ...     col3 TEXT,
       ...     col4 TEXT
       ...     );
cluster1-superuser@cqlsh:ks2>
cluster1-superuser@cqlsh:ks2> INSERT INTO t2 (col1, col2, col3, col4)
       ...     VALUES ('111', '111', '111', '111');
cluster1-superuser@cqlsh:ks2> INSERT INTO t2 (col1, col2, col3, col4)
       ...     VALUES ('222', '222', '222', '222');
cluster1-superuser@cqlsh:ks2> INSERT INTO t2 (col1, col2, col3, col4)
       ...     VALUES ('333', '333', '333', '333');
cluster1-superuser@cqlsh:ks2> INSERT INTO t2 (col1, col2, col3, col4)
       ...     VALUES ('444', '444', '444', '444');
cluster1-superuser@cqlsh:ks2> INSERT INTO t2 (col1, col2, col3, col4)
       ...     VALUES ('555', '555', '555', '555');
cluster1-superuser@cqlsh:ks2>
cluster1-superuser@cqlsh:ks2> USE ks2;
cluster1-superuser@cqlsh:ks2> SELECT COUNT(*) FROM t2;

 count
-------
     5

(1 rows)

Warnings :
Aggregation query used without partition key

cluster1-superuser@cqlsh:ks2>
```

Fig. 29 Add another database and table to DataStax Enterprise or Apache Cassandra, then insert some data.

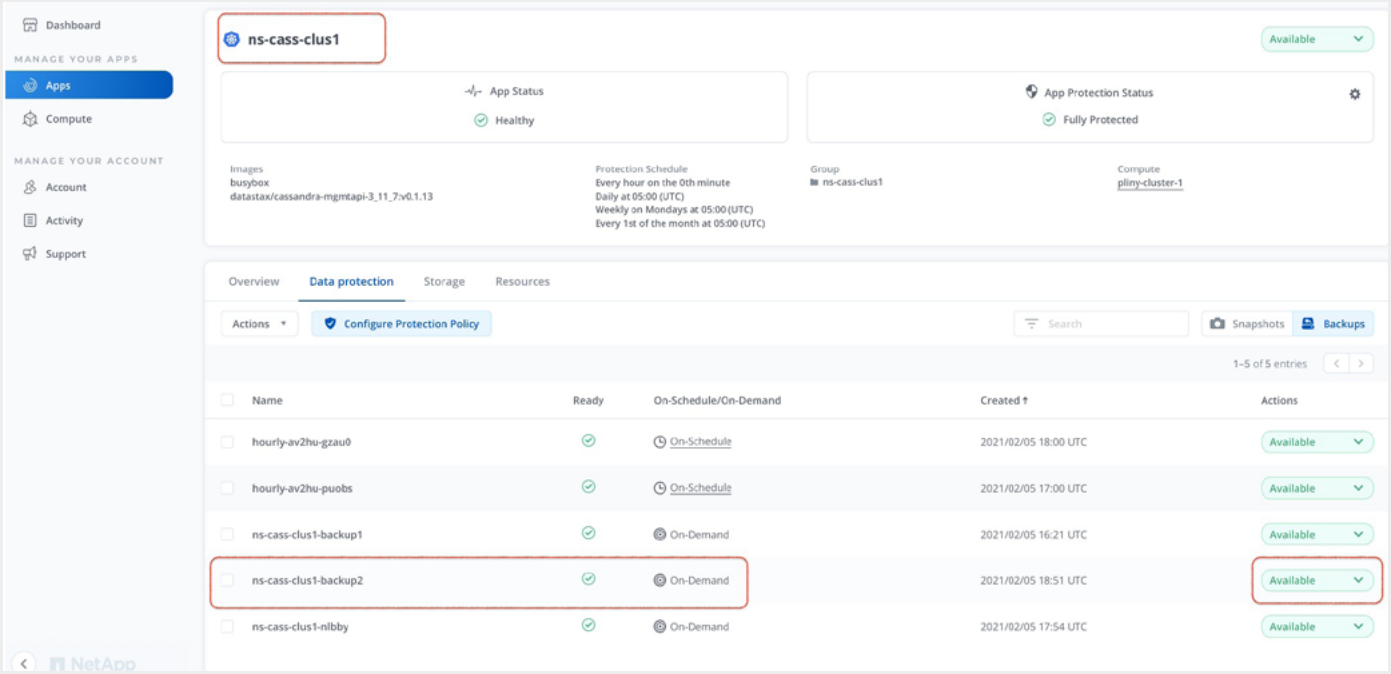


Fig. 30 Check the status of the on-demand backup.

```
jaimon-mac-0:DataStax jaimongeorge$ kubectl get pods -n ns-cass-clus1
NAME                          READY   STATUS    RESTARTS   AGE
cluster1-dc1-default-sts-0    2/2     Running   0          5h21m
cluster1-dc1-default-sts-1    2/2     Running   0          5h21m
cluster1-dc1-default-sts-2    2/2     Running   0          5h21m
jaimon-mac-0:DataStax jaimongeorge$
jaimon-mac-0:DataStax jaimongeorge$
jaimon-mac-0:DataStax jaimongeorge$
jaimon-mac-0:DataStax jaimongeorge$ kubectl delete ns ns-cass-clus1
namespace "ns-cass-clus1" deleted
```

Fig. 31 Simulating disaster by deleting the namespace where the Apache Cassandra cluster is running.

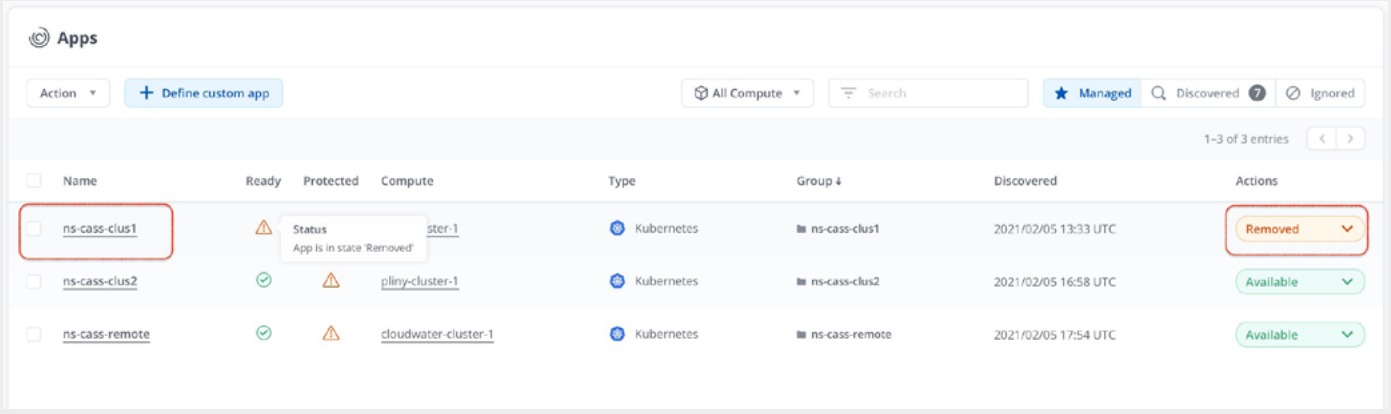


Fig. 32 NetApp Astra discovers that the namespace has been deleted.

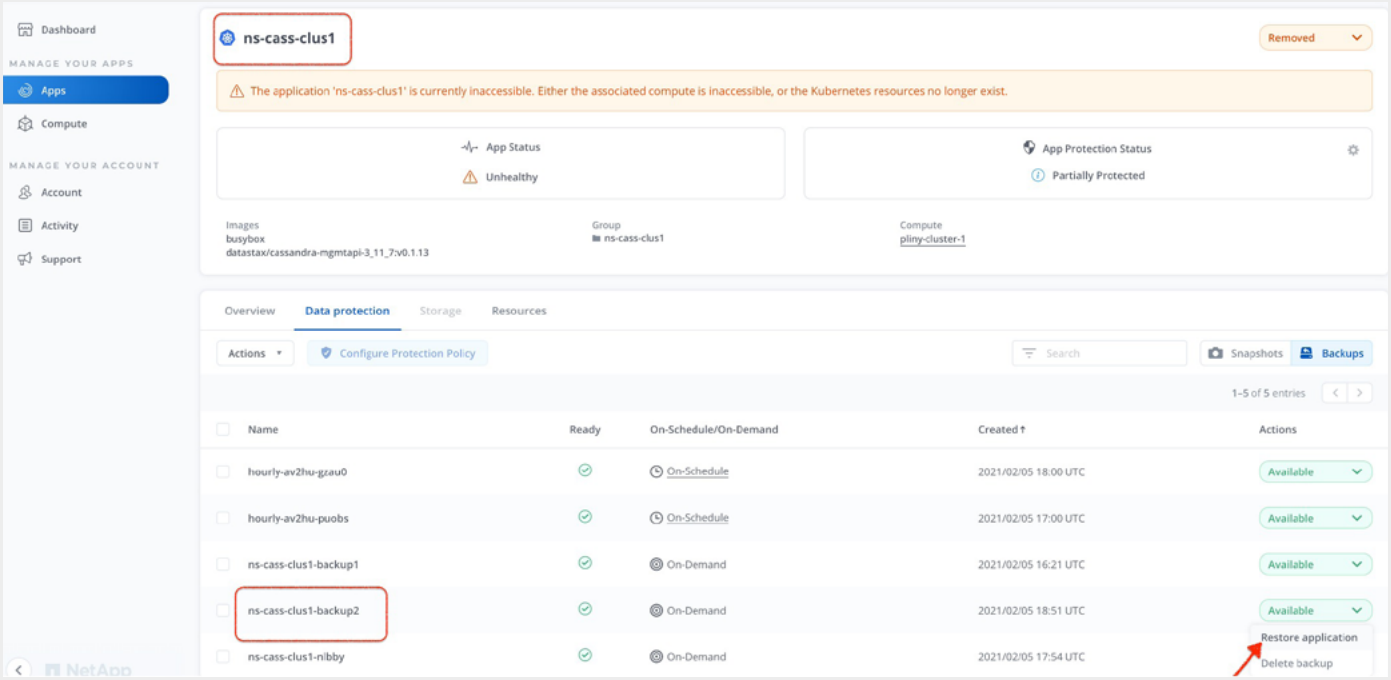


Fig. 33 Restore Apache Cassandra cluster from a backup.

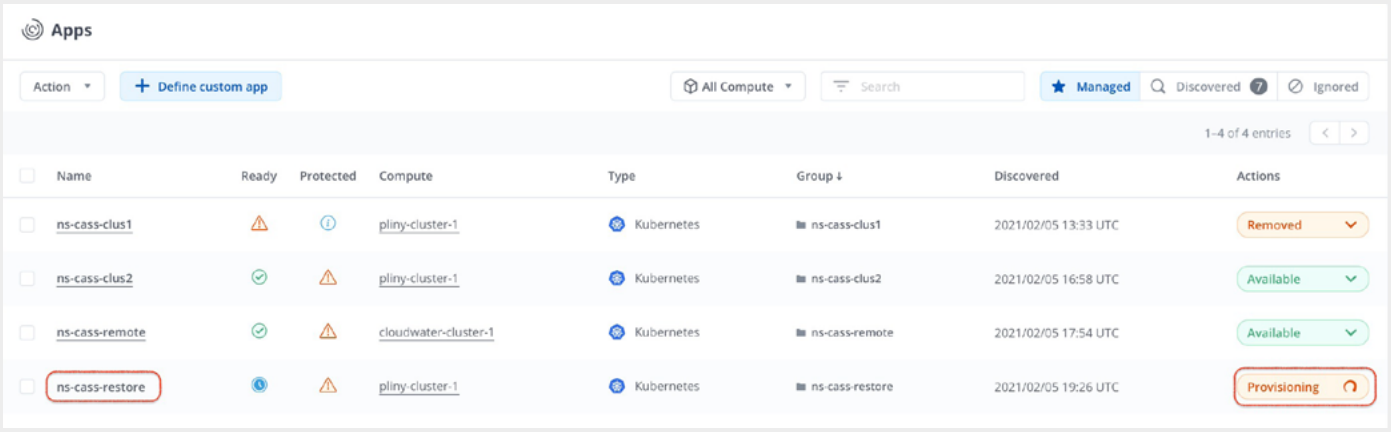


Fig. 34 NetApp Astra is restoring the Apache Cassandra cluster.

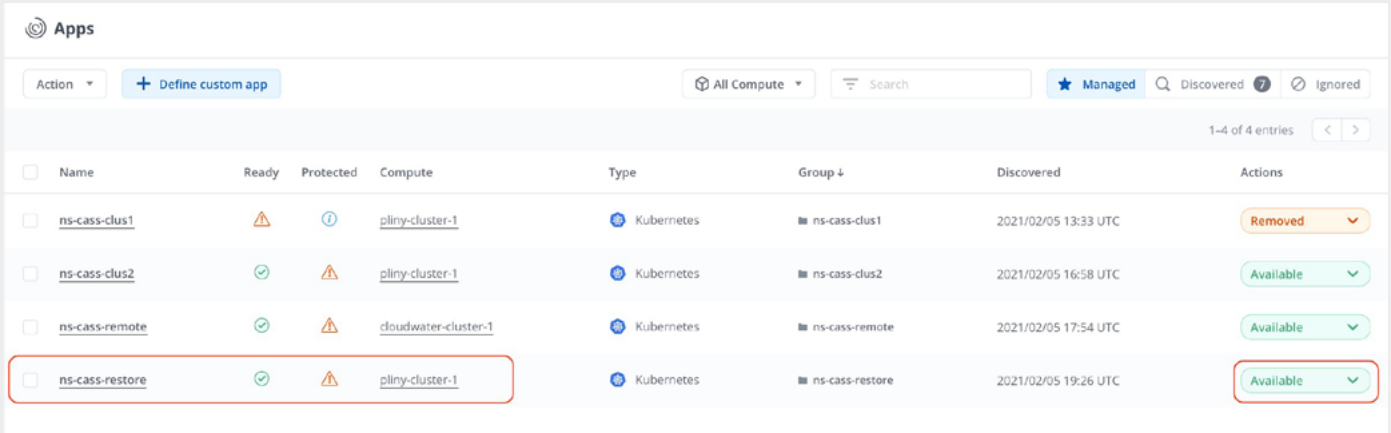


Fig. 35 After a successful restore of the Apache Cassandra cluster.

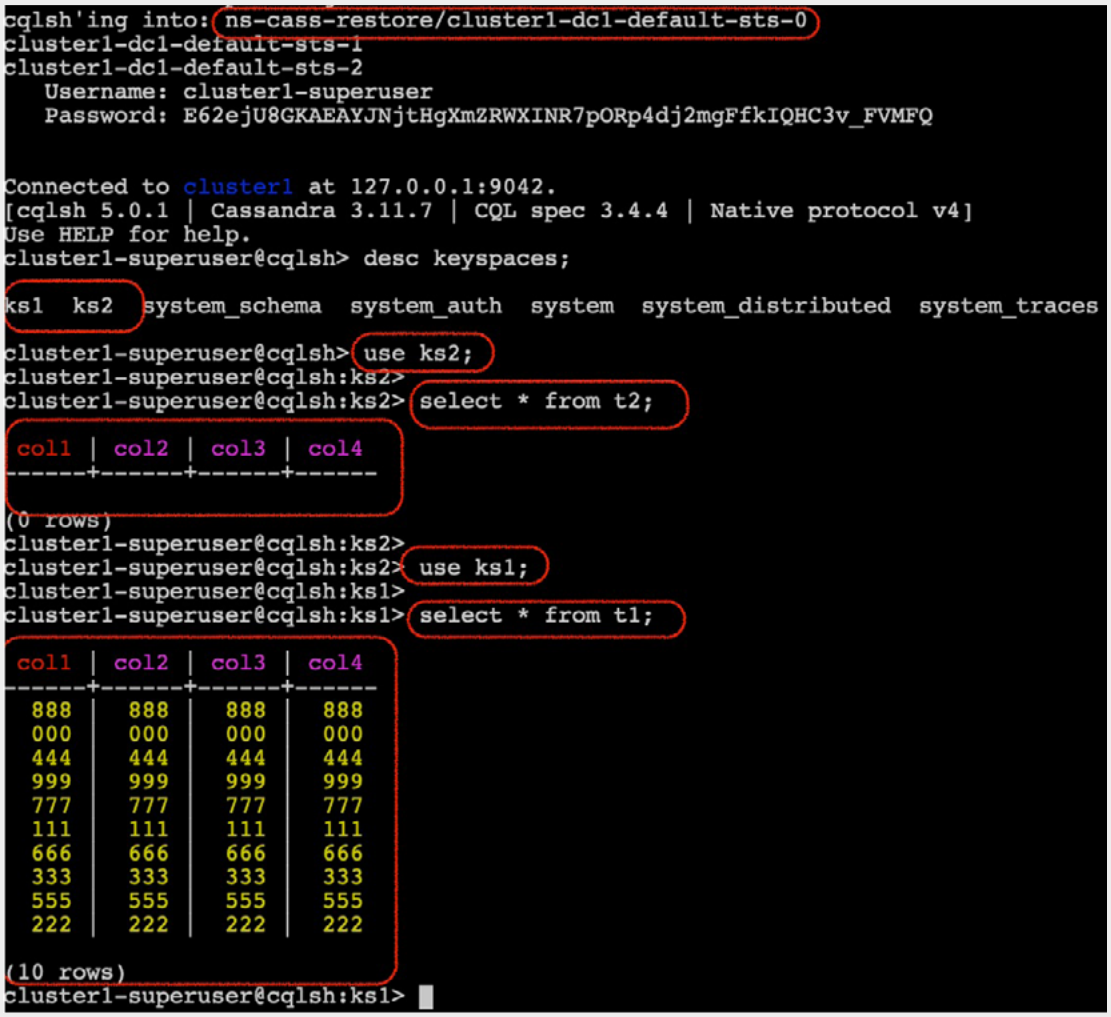


Fig. 36 Validate the database and its contents after restoring the Apache Cassandra cluster.

When restoring from the backup, NetApp Astra creates the namespace you specify. It then restores the application and its associated Kubernetes objects to that namespace.

Application-consistent snapshots and backup in NetApp Astra helps you to go back to a point-in-time copy of your DataStax Enterprise or Apache Cassandra cluster.

Summary

This solution guide provided a step-by-step guide for validating the following key benefits NetApp Astra provides to DataStax Enterprise and Apache Cassandra:

- Automatic storage provisioning and storage class set-up.
- Rich set of application-aware data management functionality (snapshot revert, backup and restore, activity log, and active cloning) for data protection, disaster recovery, data audit, and migration use-cases.
- Consistent data management UI.
- Clear visualization of data protection status.
- Simple data protection management.
- Seamless portability and migration.
- Simple and fully-managed.

Where can I learn more?

To learn more, visit the Astra website and the documentation on Project Astra.

About NetApp

In a world full of generalists, NetApp is a specialist. We're focused on one thing, helping your business get the most out of your data. NetApp brings the enterprise-grade data services you rely on into the cloud, and the simple flexibility of cloud into the data center. Our industry-leading solutions work across diverse customer environments and the world's biggest public clouds.

As a cloud-led, data-centric software company, only NetApp can help build your unique data fabric, simplify and connect your cloud, and securely deliver the right data, services and applications to the right people—anytime, anywhere. www.netapp.com

NetApp