SOLUTION BRIEF

# PostgreSQL Database as a Service (DBaaS) with Astra

Simplify day 2 operations

NetApp

### Key benefits

- Simplified application data lifecycle management
- Facilitate application-consistent snapshots and backup
- Recover the entire PostgreSQL application and Kubernetes resources in a disaster scenario
- Migrate applications to another Kubernetes cluster

## Introduction

Kubernetes has become the standard IT infrastructure for  businesses of all sizes. Production applications are being deployed on or migrated to Kubernetes. Deploying a stateful application like PostgreSQL requires lots of planning, understanding the challenges, and identifying the right solutions. Do-it-yourself need you to take the entire responsibility for building the database, setting the backup and disaster recovery strategies. More importantly to identify the way to do entire application portability.
The journey of implementing DBaaS for PostgreSQL requires the following actions on day 1, whether it's a managed Kubernetes service or vanilla Kubernetes:
The journey of implementing DBaaS for MySQL requires the following actions on day 1, whether it's a managed Kubernetes service or vanilla Kubernetes:

1. Identify or build your own registry.
2. Identify the right storage and the Container Storage Interface (CSI) provisioner,
3. Find the performance requirements and define appropriate storage classes.
4. Create your own manifest or identify a helm chart that meets your  requirements.
5. Deploy the MySQL application.

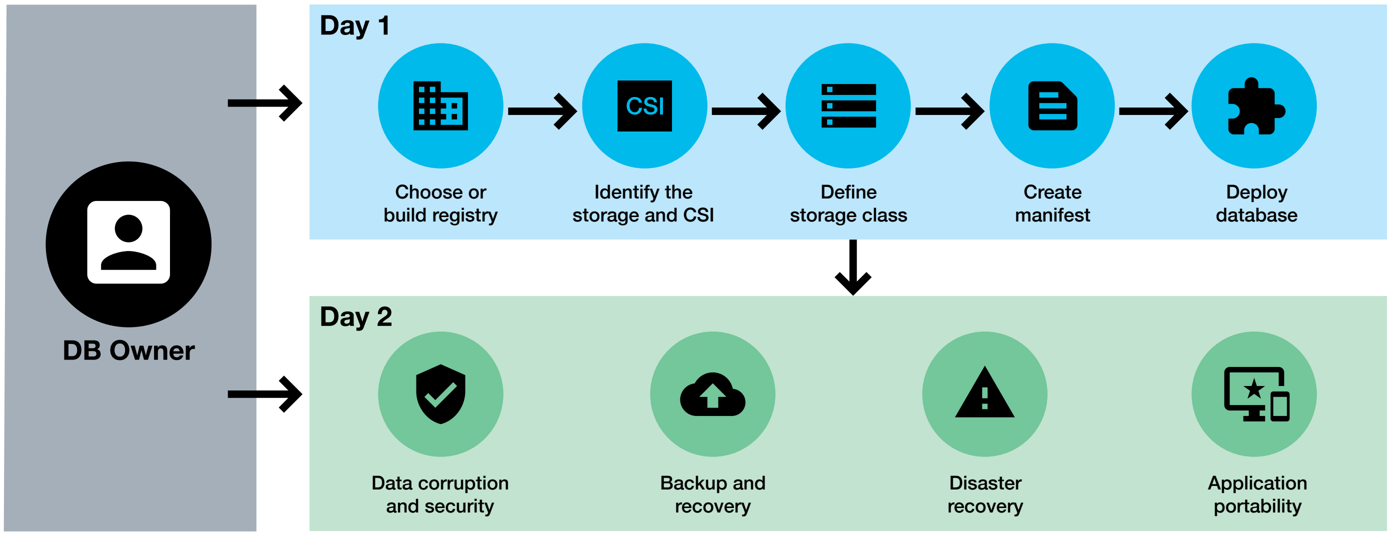


Figure 1) Build your own DBaaS.

Kubernetes offers solutions for all of the day 1 requirements. When it comes to day 2 operations, you need a strategy and solution for:

1. Data corruption and security
2. Backup and recovery
3. Disaster recovery
4. Application portability

Kubernetes natively doesn't have any solutions to address the day 2 challenges. Astra simplifies and automates the day 1 operations by simply registering the Kubernetes cluster. The day 1 operations are simplified to

1. Identify or build your own registry.
2. Create your own manifest or identify a helm chart that meets your requirements.
3. Register the Kubernetes cluster with Astra.
4. Deploy the PostgreSQL application

Project Astra managing your application addresses all day 2 challenges.

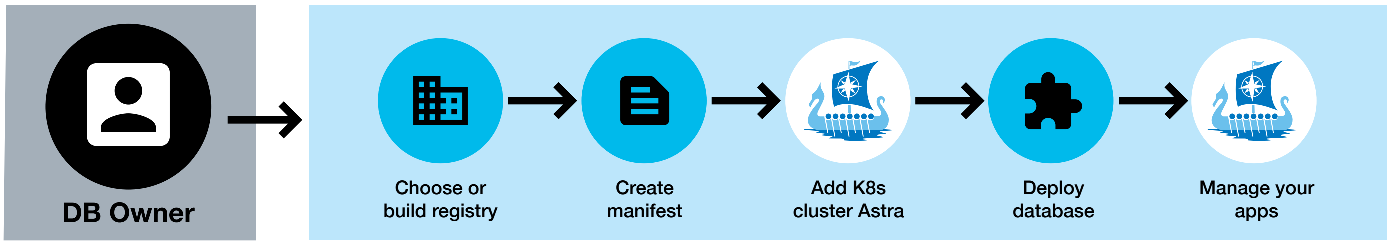


Figure 2) Day 1 operations with Astra.

## Project Astra overview

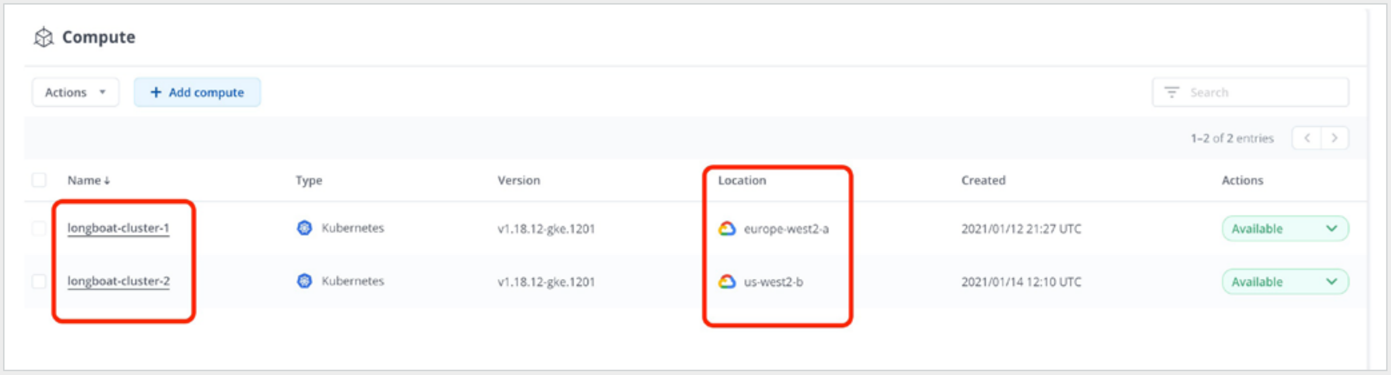
Astra is a fully managed service that makes it easier for our customers to manage, protect, and move their data-rich containerized workloads running on Kubernetes within and across public clouds and on -premises. Astra provides persistent container storage that leverages NetApp's proven and expansive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

## Managing PostgreSQL with Astra

Simply register your Google Kubernetes Engine (GKE) clusters with Astra. Upon registration, Astra:

- Installs NetApp® Trident, NetApp's open source Kubernetes storage orchestrator.
- Creates a bucket on the cloud object store for saving application backups.
- Creates a service account on your cluster for itself.

The following example shows two clusters, one located in the europe-west2-a GCP region (London) and one located in the us-west2-a (Los Angels) region.


Compute

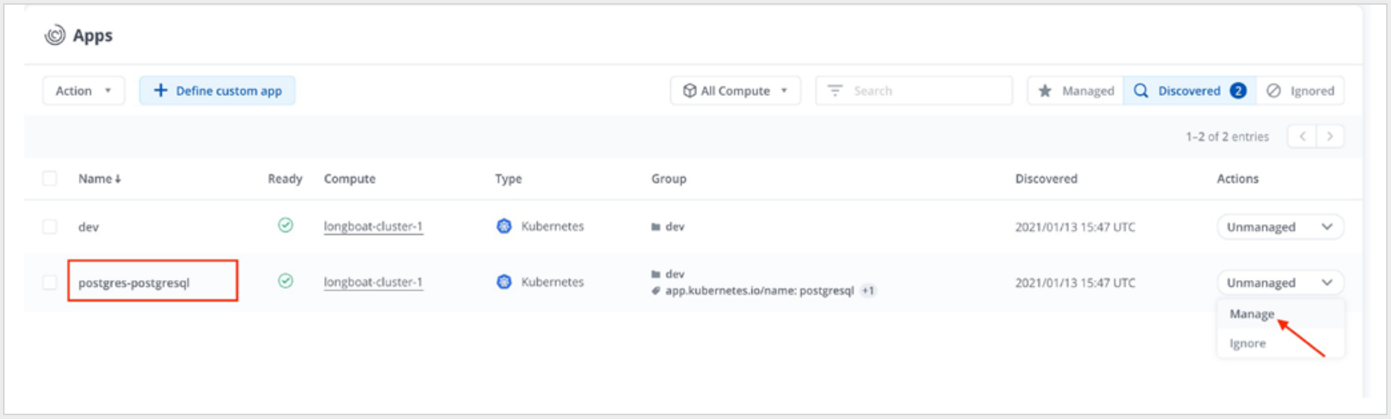
Actions | + Add compute | Search

1–2 of 2 entries

| Name | Type | Version | Location | Created | Actions |
|---|---|---|---|---|---|
| longboat-cluster-1 | Kubernetes | v1.18.12-gke.1201 | europe-west2-a | 2021/01/12 21:27 UTC | Available |
| longboat-cluster-2 | Kubernetes | v1.18.12-gke.1201 | us-west2-b | 2021/01/14 12:10 UTC | Available |

Figure 3) Registered clusters.

Install PostgreSQL on cluster longboat-cluster-1 using the current Bitnami Helm chart or a custom manifest. Trident automatically provisions the Kubernetes Persistent Volume Claims from NetApp Cloud Volumes Services for PostgreSQL.  Astra discovers the applications on your registered clusters, and you can easily  manage either just the application or all the resources in the entire namespace as one unit.


Apps

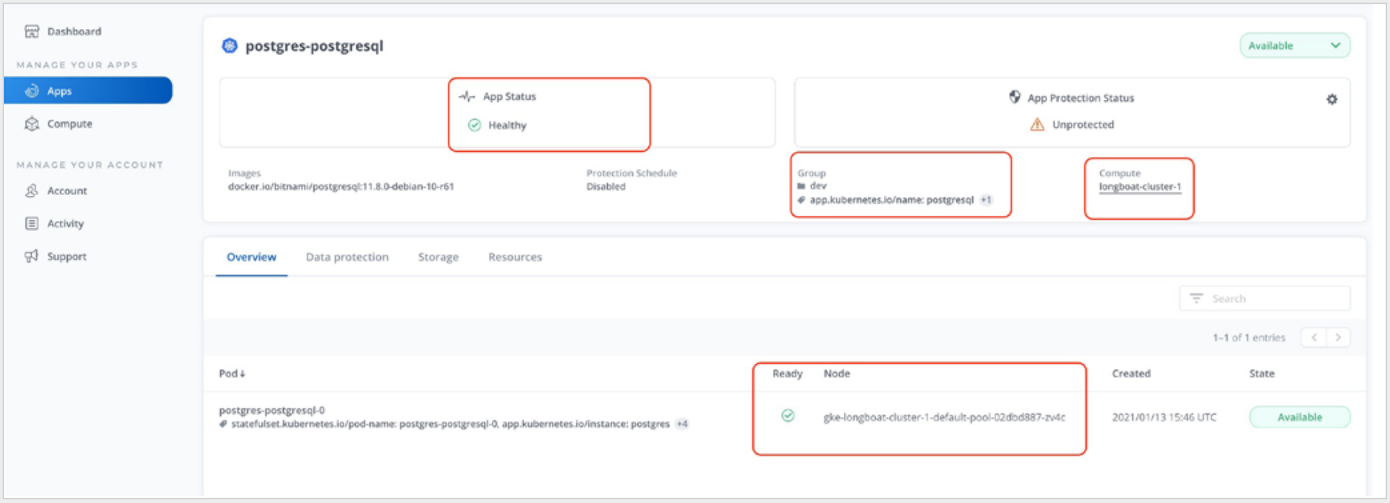
Action | + Define custom app | All Compute | Search | Managed | Discovered 2 | Ignored

1–2 of 2 entries

| Name | Ready | Compute | Type | Group | Discovered | Actions |
|---|---|---|---|---|---|---|
| dev | ✓ | longboat-cluster-1 | Kubernetes | dev | 2021/01/13 15:47 UTC | Unmanaged |
| postgres-postgresql | ✓ | longboat-cluster-1 | Kubernetes | dev<br>app.kubernetes.io/name: postgresql +1 | 2021/01/13 15:47 UTC | Unmanaged |

Figure 4) Managing the PostgreSQL application.

After managing the application, Astra can take snapshots, backups, and clones of that application, its Kubernetes resources, and its associated Persistent Volumes.



postgres-postgresql

App Status: Healthy

App Protection Status: Unprotected

Images: docker.io/bitnami/postgresql:11.8.0-debian-10-r61

Protection Schedule: Disabled

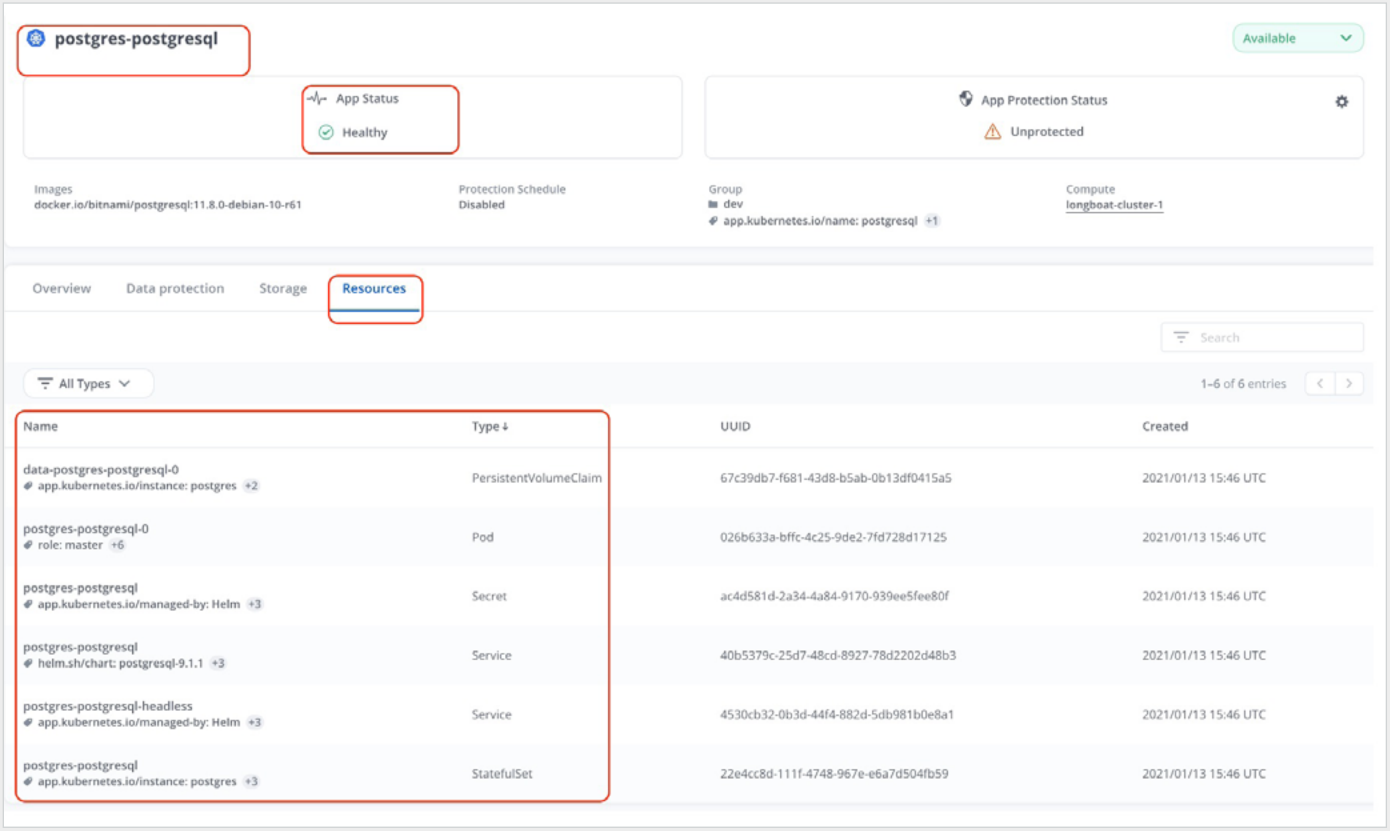
Group: dev, app.kubernetes.io/name: postgresql +1

Compute: longboat-cluster-1

1–1 of 1 entries

| Pod ↓ | Ready | Node | Created | State |
|---|---|---|---|---|
| postgres-postgresql-0<br>statefulset.kubernetes.io/pod-name: postgres-postgresql-0, app.kubernetes.io/instance: postgres +4 | ✓ | gke-longboat-cluster-1-default-pool-02dbd887-zv4c | 2021/01/13 15:46 UTC | Available |

Figure 5) Managed application overview.



postgres-postgresql

App Status: Healthy

App Protection Status: Unprotected

Images: docker.io/bitnami/postgresql:11.8.0-debian-10-r61

Protection Schedule: Disabled

Group: dev, app.kubernetes.io/name: postgresql +1

Compute: longboat-cluster-1

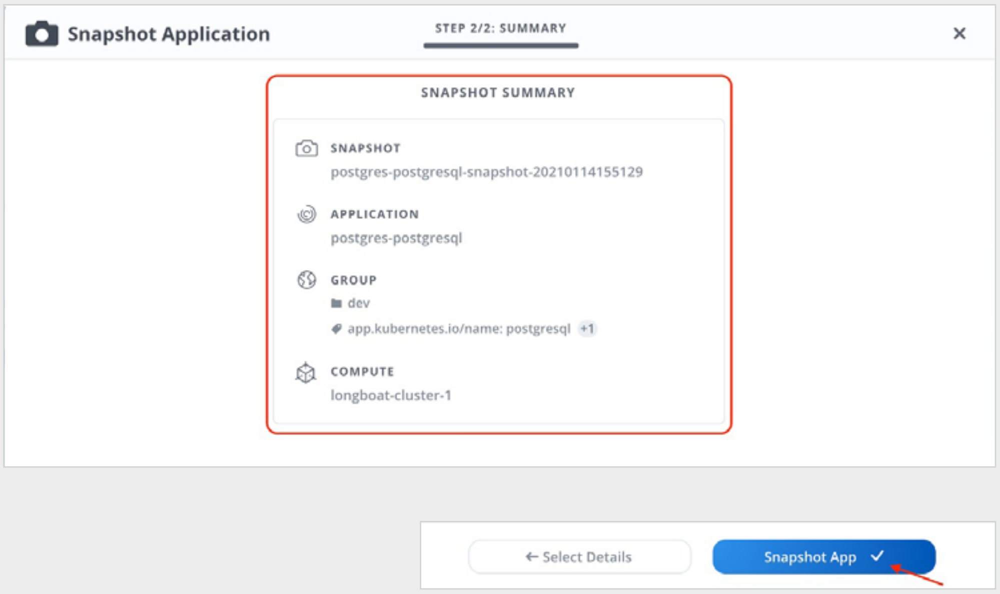
1–6 of 6 entries

| Name | Type ↓ | UUID | Created |
|---|---|---|---|
| data-postgres-postgresql-0<br>app.kubernetes.io/instance: postgres +2 | PersistentVolumeClaim | 67c39db7-f681-43d8-b5ab-0b13df0415a5 | 2021/01/13 15:46 UTC |
| postgres-postgresql-0<br>role: master +6 | Pod | 026b633a-bffc-4c25-9de2-7fd728d17125 | 2021/01/13 15:46 UTC |
| postgres-postgresql<br>app.kubernetes.io/managed-by: Helm +3 | Secret | ac4d581d-2a34-4a84-9170-939ee5fee80f | 2021/01/13 15:46 UTC |
| postgres-postgresql<br>helm.sh/chart: postgresql-9.1.1 +3 | Service | 40b5379c-25d7-48cd-8927-78d2202d48b3 | 2021/01/13 15:46 UTC |
| postgres-postgresql-headless<br>app.kubernetes.io/managed-by: Helm +3 | Service | 4530cb32-0b3d-44f4-882d-5db981b0e8a1 | 2021/01/13 15:46 UTC |
| postgres-postgresql<br>app.kubernetes.io/instance: postgres +3 | StatefulSet | 22e4cc8d-111f-4748-967e-e6a7d504fb59 | 2021/01/13 15:46 UTC |

Figure 6) Kubernetes resources for the application.

All the data generated by PostgreSQL database clients can be automatically protected by using snapshots and backups. Astra snapshots and backups preserve the application state, its Kubernetes resources, and its volumes in one easily manageable unit. Project Astra understands the PostgreSQL application and is quiesced before a snapshot or backup so that an application-consistent snapshot or backup can be taken. Quiesce operations take no longer than 60 seconds. All application backups are stored in an object store.

Both on-demand and scheduled snapshots and backups are supported. When taking on-demand backup, there is an option to choose any existing snapshot. Otherwise, the backup will be from the current state of the application.



postgres-postgresql-snapshot-20210114155129

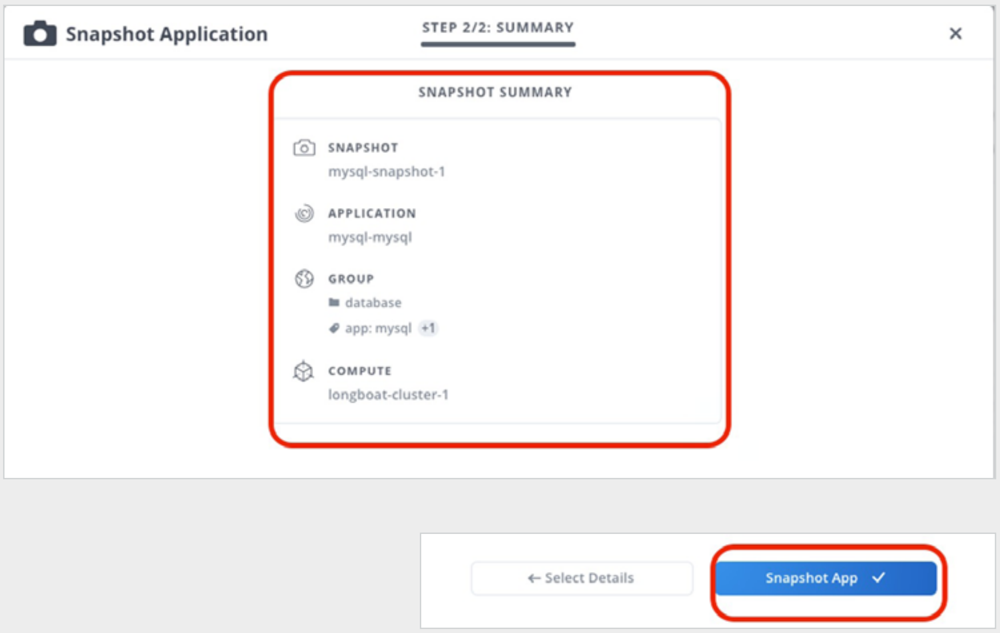
postgres-postgresql

dev

app.kubernetes.io/name: postgresql +1

longboat-cluster-1

Figure 7a) On-demand application snapshot.



mysql-snapshot-1

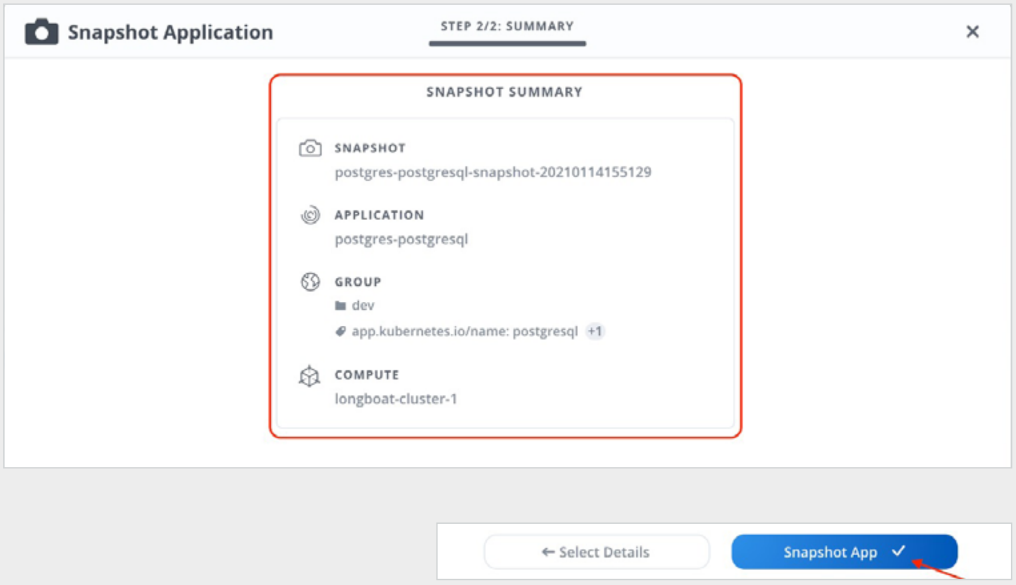
mysql-mysql

database

app: mysql +1

longboat-cluster-1

Figure 7b) On-demand application snapshot.



SNAPSHOT SUMMARY

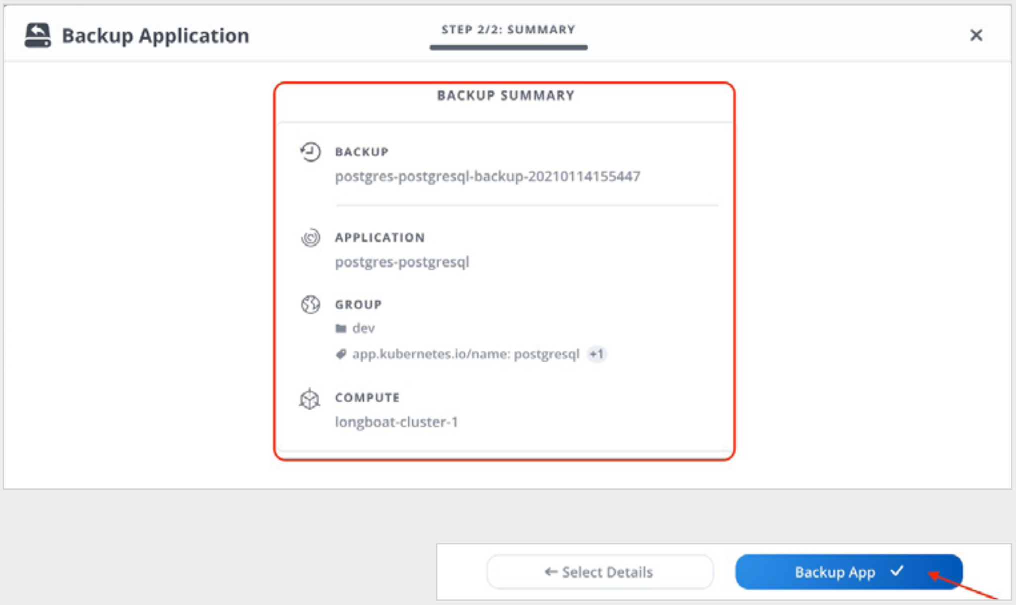
SNAPSHOT
postgres-postgresql-snapshot-20210114155129

APPLICATION
postgres-postgresql

GROUP
dev
app.kubernetes.io/name: postgresql +1

COMPUTE
longboat-cluster-1

Figure 8a) On-demand application backup.



BACKUP SUMMARY

BACKUP
postgres-postgresql-backup-20210114155447

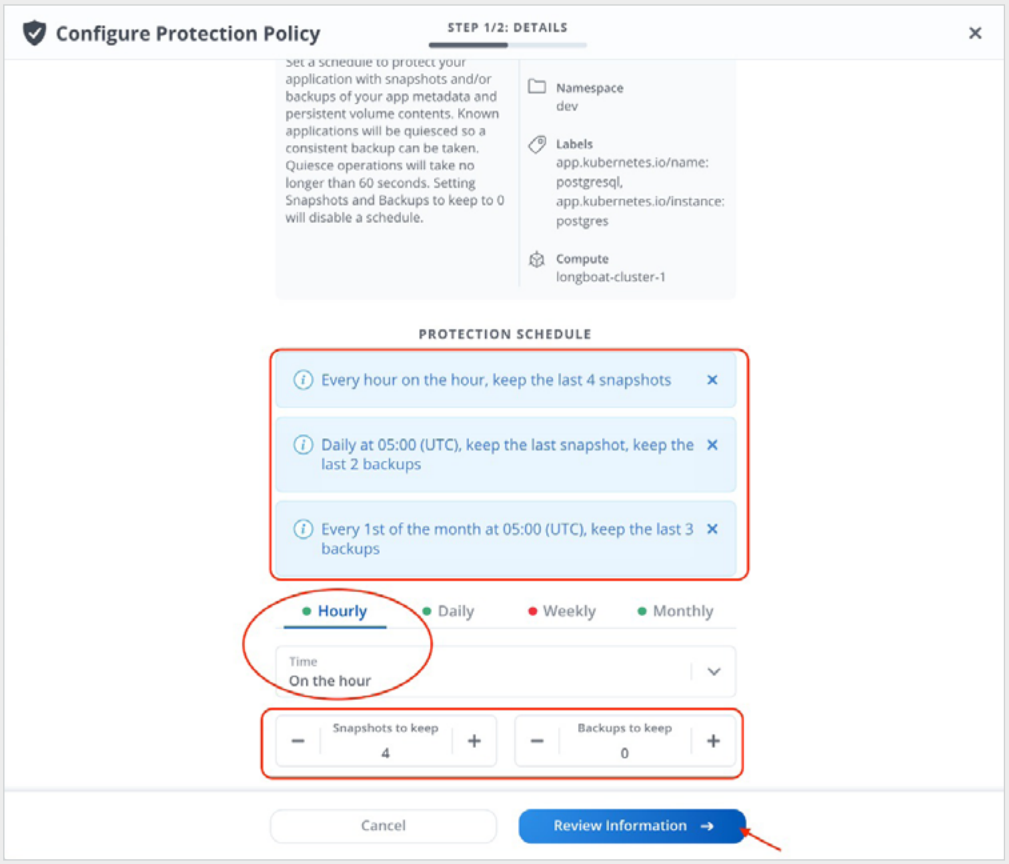
APPLICATION
postgres-postgresql

GROUP
dev
app.kubernetes.io/name: postgresql +1

COMPUTE
longboat-cluster-1

Figure 8b) On-demand application backup.

Set up a snapshot and backup schedule for the volume and all the Kubernetes objects that are associated with it.


Configure Protection Policy

STEP 1/2: DETAILS

Set a schedule to protect your application with snapshots and/or backups of your app metadata and persistent volume contents. Known applications will be quiesced so a consistent backup can be taken. Quiesce operations will take no longer than 60 seconds. Setting Snapshots and Backups to keep to 0 will disable a schedule.

Namespace
dev

Labels
app.kubernetes.io/name: postgresql, app.kubernetes.io/instance: postgres

Compute
longboat-cluster-1

PROTECTION SCHEDULE

- Every hour on the hour, keep the last 4 snapshots
- Daily at 05:00 (UTC), keep the last snapshot, keep the last 2 backups
- Every 1st of the month at 05:00 (UTC), keep the last 3 backups

[✓ Hourly] Daily Weekly Monthly

Time: On the hour

Snapshots to keep: 4

Backups to keep: 0

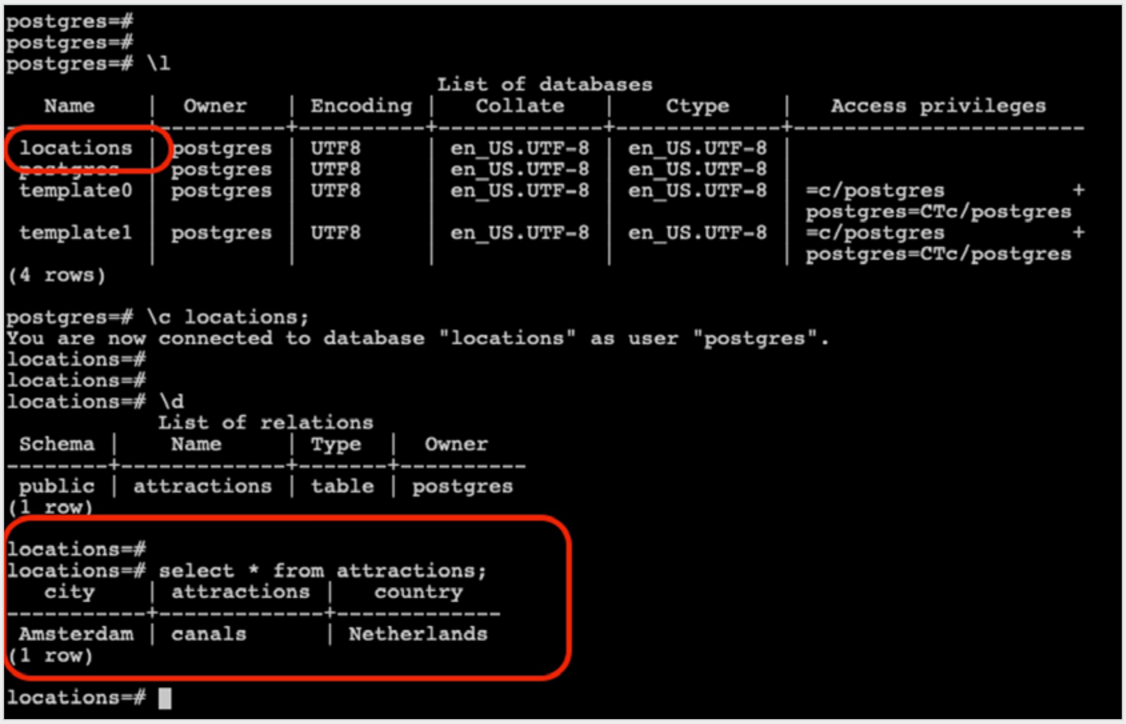
Figure 9) Configure protection policy.

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

**Migrating PostgreSQL to another cluster**

After a successful backup, the PostgreSQL application is protected against disaster like losing the Kubernetes cluster or a human error like deleting the namespace. You can use the Clone option to redeploy PostgreSQL to a new namespace within the cluster or to new cluster. When choosing the option, you can also select an existing snapshot or backup to go back to a point- in- time copy of the PostgreSQL application.

For example, suppose that you have a new team in a different location that is going to take over the responsibility of managing the PostgreSQL database You want to migrate the PostgreSQL applications closer to the new team. PostgreSQL is currently running on the longboat-cluster-1 cluster in the us-cental1 (Iowa) region.



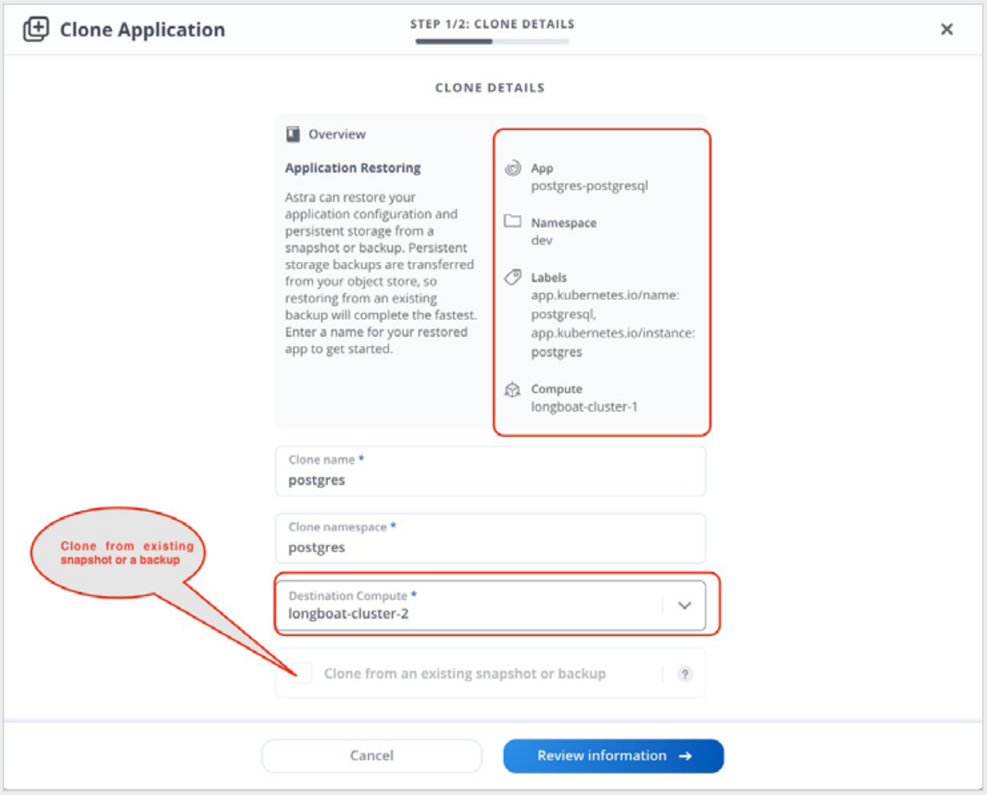
```
postgres=#
postgres=#
postgres=# \l
                                  List of databases
   Name    |  Owner   | Encoding |   Collate   |    Ctype    |   Access privileges
-----------+----------+----------+-------------+-------------+-----------------------
 locations | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
 postgres  | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
 template0 | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =c/postgres          +
           |          |          |             |             | postgres=CTc/postgres
 template1 | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =c/postgres          +
           |          |          |             |             | postgres=CTc/postgres
(4 rows)

postgres=# \c locations;
You are now connected to database "locations" as user "postgres".
locations=#
locations=#
locations=# \d
            List of relations
 Schema |    Name     | Type  |  Owner
--------+-------------+-------+----------
 public | attractions | table | postgres
(1 row)

locations=#
locations=# select * from attractions;
   city    | attractions |   country
-----------+-------------+-------------
 Amsterdam | canals      | Netherlands
(1 row)

locations=# █
```

**Headline**

Figure 10) Current state of the PostgreSQL application.

Body text

Clone PostgreSQL to a new cluster, longboat-cluster-2, in a different Google Cloud Platform region using its current state. You could also clone from an existing backup or snapshot. When cloning from the current state, Astra first creates a backup and then uses that backup for migrating to the destination cluster. This brings up a new instance of PostgreSQL, running at the same state as in the source cluster.


Clone Application

STEP 1/2: CLONE DETAILS

CLONE DETAILS

Overview

Application Restoring

Astra can restore your application configuration and persistent storage from a snapshot or backup. Persistent storage backups are transferred from your object store, so restoring from an existing backup will complete the fastest. Enter a name for your restored app to get started.

App: postgres-postgresql

Namespace: dev

Labels: app.kubernetes.io/name: postgresql, app.kubernetes.io/instance: postgres

Compute: longboat-cluster-1

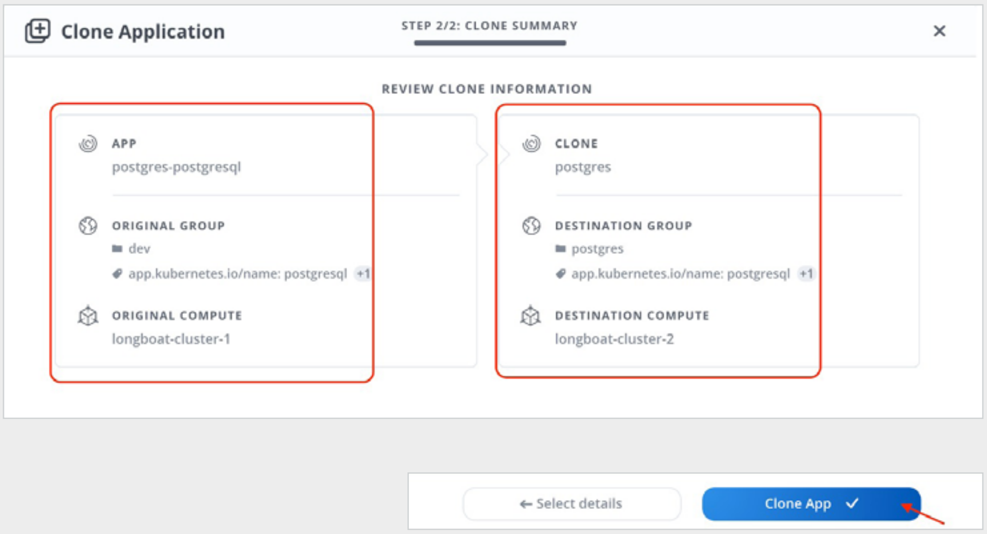
Clone name *: postgres

Clone namespace *: postgres

Destination Compute *: longboat-cluster-2

Clone from an existing snapshot or backup

Figure 11a) Migrating PostgreSQL from the current state.


Clone Application

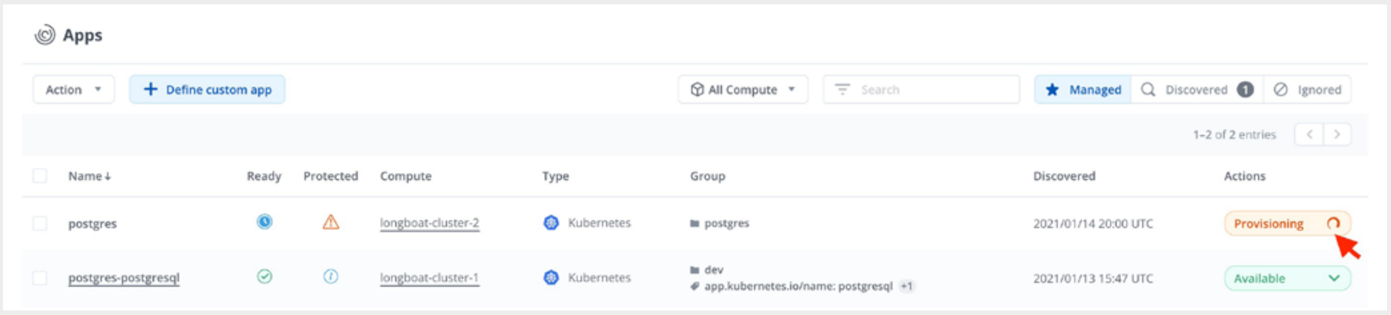
STEP 2/2: CLONE SUMMARY

REVIEW CLONE INFORMATION

| | |
|---|---|
| APP: postgres-postgresql | CLONE: postgres |
| ORIGINAL GROUP: dev; app.kubernetes.io/name: postgresql +1 | DESTINATION GROUP: postgres; app.kubernetes.io/name: postgresql +1 |
| ORIGINAL COMPUTE: longboat-cluster-1 | DESTINATION COMPUTE: longboat-cluster-2 |

Figure 11b) Migrating PostgreSQL from the current state.

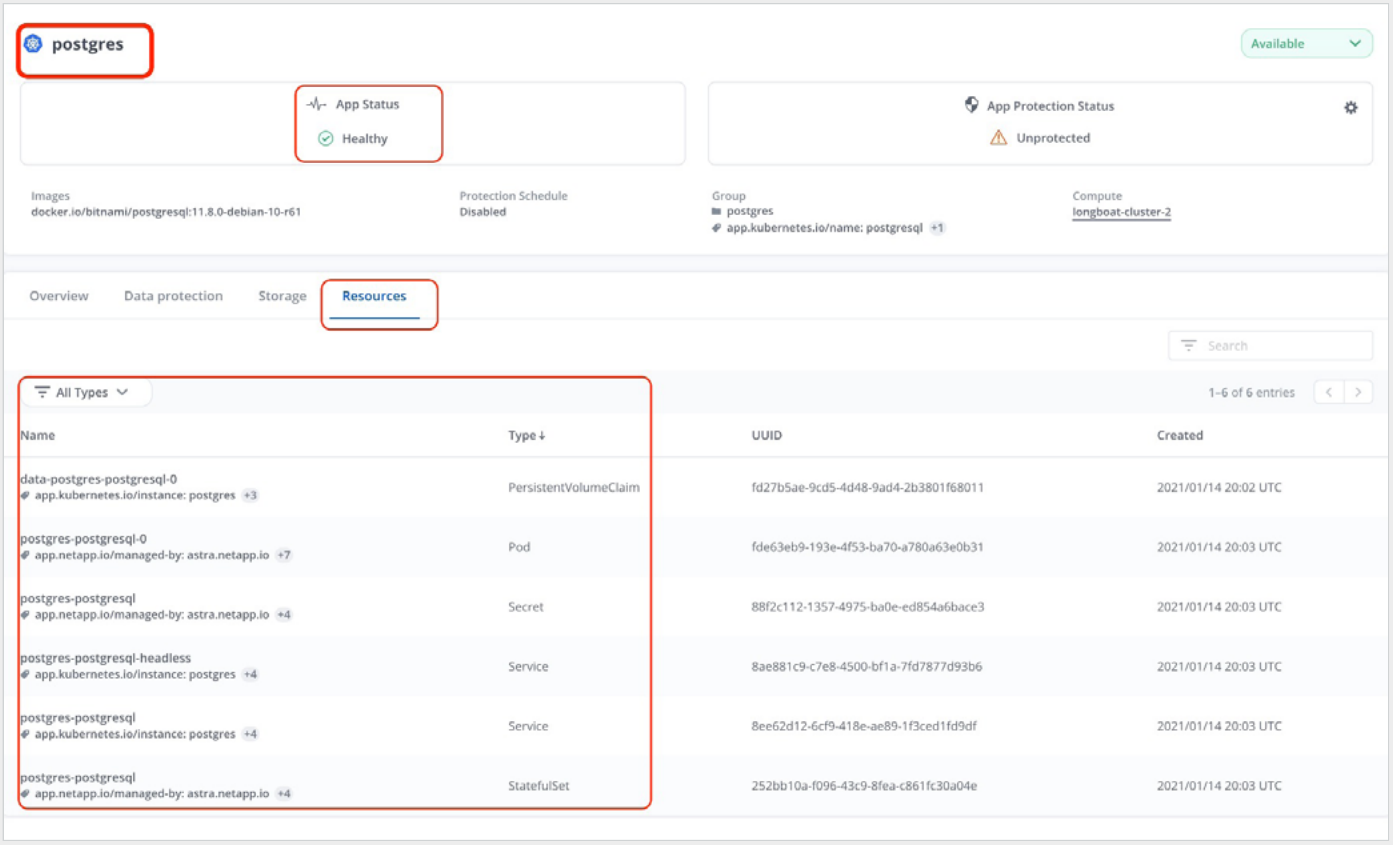
A new PostgreSQL clone is provisioned in the destination cluster and the application is automatically managed by Astra.



| Name | Ready | Protected | Compute | Type | Group | Discovered | Actions |
|---|---|---|---|---|---|---|---|
| postgres | | | longboat-cluster-2 | Kubernetes | postgres | 2021/01/14 20:00 UTC | Provisioning |
| postgres-postgresql | | | longboat-cluster-1 | Kubernetes | dev<br>app.kubernetes.io/name: postgresql +1 | 2021/01/13 15:47 UTC | Available |

Figure 12) Provisioning a new PostgreSQL instance in the destination cluster.

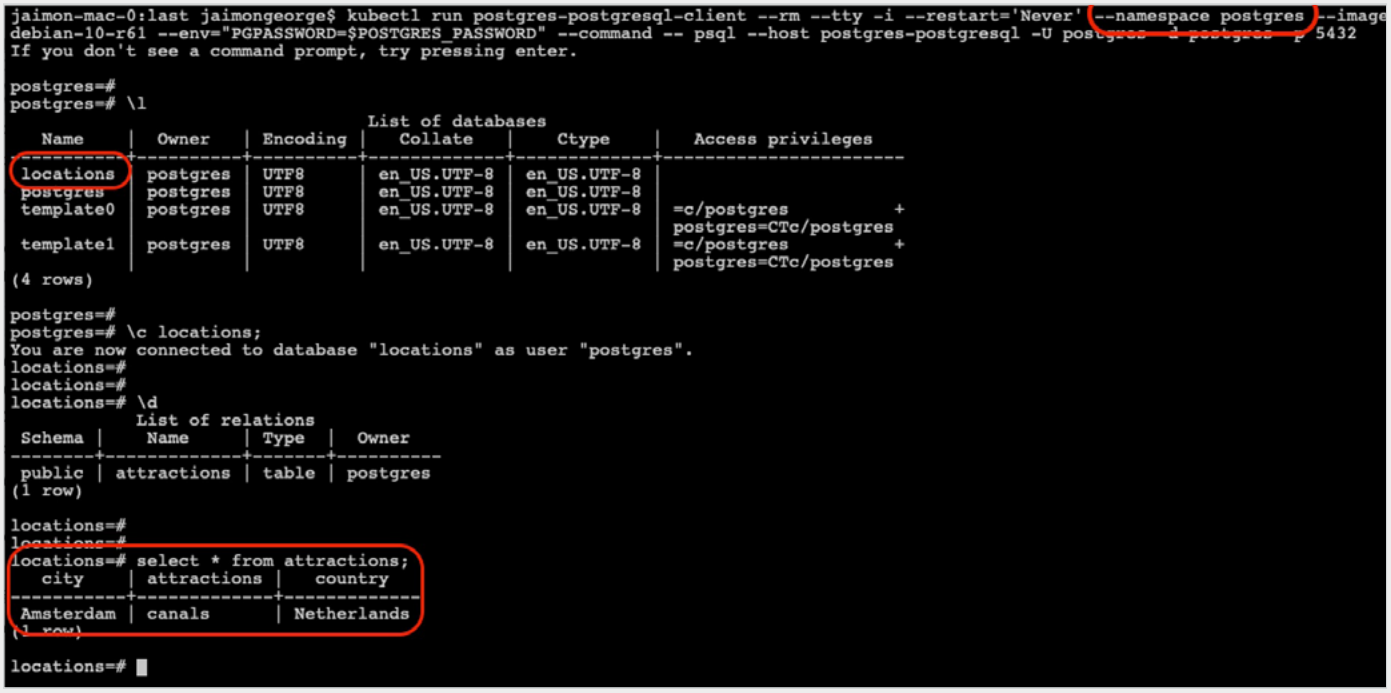
After the migration, the PostgreSQL application has the same Kubernetes resources and data as in the source cluster.



Images: docker.io/bitnami/postgresql:11.8.0-debian-10-r61
Protection Schedule: Disabled
Group: postgres; app.kubernetes.io/name: postgresql +1
Compute: longboat-cluster-2

| Name | Type | UUID | Created |
|---|---|---|---|
| data-postgres-postgresql-0<br>app.kubernetes.io/instance: postgres +3 | PersistentVolumeClaim | fd27b5ae-9cd5-4d48-9ad4-2b3801f68011 | 2021/01/14 20:02 UTC |
| postgres-postgresql-0<br>app.netapp.io/managed-by: astra.netapp.io +7 | Pod | fde63eb9-193e-4f53-ba70-a780a63e0b31 | 2021/01/14 20:03 UTC |
| postgres-postgresql<br>app.netapp.io/managed-by: astra.netapp.io +4 | Secret | 88f2c112-1357-4975-ba0e-ed854a6bace3 | 2021/01/14 20:03 UTC |
| postgres-postgresql-headless<br>app.kubernetes.io/instance: postgres +4 | Service | 8ae881c9-c7e8-4500-bf1a-7fd7877d93b6 | 2021/01/14 20:03 UTC |
| postgres-postgresql<br>app.kubernetes.io/instance: postgres +4 | Service | 8ee62d12-6cf9-418e-ae89-1f3ced1fd9df | 2021/01/14 20:03 UTC |
| postgres-postgresql<br>app.netapp.io/managed-by: astra.netapp.io +4 | StatefulSet | 252bb10a-f096-43c9-8fea-c861fc30a04e | 2021/01/14 20:03 UTC |

Figure 13a) PostgreSQL application after migration.



```
jaimon-mac-0:last jaimongeorge$ kubectl run postgres-postgresql-client --rm --tty -i --restart='Never' --namespace postgres --image
debian-10-r61 --env="PGPASSWORD=$POSTGRES_PASSWORD" --command -- psql --host postgres-postgresql -U postgres -d postgres -p 5432
If you don't see a command prompt, try pressing enter.

postgres=#
postgres=# \l
                              List of databases
   Name    |  Owner   | Encoding |   Collate   |    Ctype    |   Access privileges
-----------+----------+----------+-------------+-------------+-----------------------
 locations | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
 postgres  | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
 template0 | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =c/postgres          +
           |          |          |             |             | postgres=CTc/postgres
 template1 | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =c/postgres          +
           |          |          |             |             | postgres=CTc/postgres
(4 rows)

postgres=#
postgres=# \c locations;
You are now connected to database "locations" as user "postgres".
locations=#
locations=#
locations=# \d
            List of relations
 Schema |    Name     | Type  |  Owner
--------+-------------+-------+----------
 public | attractions | table | postgres
(1 row)

locations=#
locations=#
locations=# select * from attractions;
   city    | attractions |   country
-----------+-------------+-------------
 Amsterdam | canals      | Netherlands
(1 row)

locations=# █
```

Figure 13b) PostgreSQL application after migration.

## 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

© 2021 NetApp, Inc. All Rights Reserved. NETAPP, the NETAPP logo, and the marks listed at http://www.netapp.com/TM are trademarks of NetApp, Inc.
Other company and product names may be trademarks of their respective owners. SB-4122-0121