Learn

Astra

NetApp
April 15, 2021

Table of Contents

Learn

Storage classes and PV size for AKS clusters

Astra uses Azure NetApp Files as the backend storage for Azure Kubernetes Service (AKS) clusters. You should understand how choosing a storage class and persistent volume size can help you meet your performance objectives.

Service levels and storage classes

Azure NetApp Files supports three service levels: Ultra storage, Premium storage, and Standard storage. Each of these service levels are designed for different performance needs:

Ultra storage
: Provides up to 128 MiB/s of throughput per 1 TiB.

Premium storage
: Provides up to 64 MiB/s of throughput per 1 TiB.

Standard storage
: Provides up to 16 Mib/s of throughput per 1 TiB.

These service levels are an attribute of a capacity pool. You need to set up a capacity pool for each service level that you want to use with your Kubernetes clusters. Learn how to set up capacity pools.

Astra uses these service levels as storage classes for your persistent volumes. When you add Kubernetes compute to Astra, you're prompted to choose either Ultra, Premium, or Standard as the default storage class. The names of the storage classes are *netapp-anf-perf-ultra*, *netapp-anf-perf-premium*, and *netapp-anf-perf-standard*.

Learn more about these service levels in the Azure NetApp Files docs.

Persistent volume size and performance

As described above, the throughput for each service level is per 1 TiB of provisioned capacity. That means larger volumes provide better performance. So you should take both capacity and performance needs into consideration when provisioning volumes.

Minimum volume size

Astra provisions persistent volumes using a minimum volume size of 100 GiB, even if the PVC asks for a smaller volume size. For example, if the PVC in a Helm chart asks for 6 GiB, Astra automatically provisions a 100 GiB volume.

Service type, storage classes, and PV size for GKE clusters

Astra uses Cloud Volumes Service for Google Cloud as the backend storage for persistent volumes. You should understand how choosing a service type, storage class, and persistent volume size can help you meet your performance objectives.

Overview

Cloud Volumes Service for Google Cloud provides two service types: *CVS* and *CVS-Performance*. These service types are supported in specific Google Cloud regions. Go to NetApp Cloud Central's Global Regions Maps to identify the service type that's supported in the Google Cloud region where your clusters reside.

If your Kubernetes clusters must reside in a specific region, then you'll be using the service type supported in that region.

But if you have the flexibility to choose between Google Cloud regions, then we recommend the following based on your performance requirements:

- For K8s applications that have medium-to-high performance storage needs, choose a Google Cloud region that supports CVS-Performance and use the Premium or Extreme storage class. Such workloads include AI/ML pipelines, CI/CD pipelines, media processing, and databases including relational, noSQL, time series, etc.
- For K8s applications that have low-to-medium storage performance needs (web apps, general purpose file storage, etc.), choose a Google Cloud region that supports either CVS or CVS-Performance, with the Standard storage class.

The following table provides a quick comparison of the information described on this page.

Service type	Use case	Supported regions	Storage classes	Min volume size
CVS-Performance	Apps with medium-to-high storage performance needs	View supported Google Cloud regions	• netapp-cvs-standard • netapp-cvs-premium • netapp-cvs-extreme	100 GiB
CVS	Apps with low-to-medium storage performance needs	View supported Google Cloud regions	netapp-cvs-standard	100 GiB

CVS-Performance service type

Learn more about the CVS-Performance service type before you choose a storage class and create persistent volumes.

Storage classes

Three service levels are supported with the CVS-Performance service type: Standard, Premium, and Extreme. When you add compute to Astra, you're prompted to choose either Standard, Premium, or Extreme as the default storage class for persistent volumes. Each of these service levels are designed for different capacity and bandwidth needs.

The names of the storage classes are *netapp-cvs-standard*, *netapp-cvs-premium*, and *netapp-cvs-extreme*.

Learn more about these service levels in the Cloud Volumes Service for Google Cloud docs.

Persistent volume size and performance

As the Google Cloud docs explain, the allowed bandwidth for each service level is per GiB of provisioned capacity. That means larger volumes will provide better performance.

Be sure to read through the Google Cloud page linked to above. It includes cost comparisons and examples that can help you better understand how to couple a service level with volume size to meet your performance objectives.

Minimum volume size

Astra provisions persistent volumes using a minimum volume size of 100 GiB with the CVS-Performance service type, even if the PVC asks for a smaller volume size. For example, if the PVC in a Helm chart asks for 6 GiB, Astra automatically provisions a 100 GiB volume.

CVS service type

Learn more about the CVS service type before you choose a storage class and create persistent volumes.

Storage class

One service level is supported with the CVS service type: Standard. When you manage clusters in regions where the CVS service type is supported, Astra uses the Standard service level as the default storage class for persistent volumes. The storage class is named *netapp-cvs-standard*.

Learn more about the Standard service level in the Cloud Volumes Service for Google Cloud docs.

Persistent volume size and performance

The allowed bandwidth for the CVS service type is per GiB of provisioned capacity. That means larger volumes will provide better performance.

Minimum volume size

Astra provisions persistent volumes using a minimum volume size of 100 GiB with the CVS service type, even if the PVC asks for a smaller volume size.

Validated vs standard apps

There are two types of applications you can bring to Astra: Validated and Standard. Learn the difference between these two categories, and the potential impacts on your projects and strategy.

It's tempting to think of these two categories as "supported" and "unsupported." But as you will see, there is no such thing as an "unsupported" app in Astra. You can add any app to Astra, although validated apps have more infrastructure built around their Astra workflows compared to standard apps.

Validated Apps

Validated apps for Astra include the following:

- MySQL 0.3.22

- MariaDB 14.14
- PostgreSQL 11.7
- Jenkins 2.249.1 LTS

The short list of validated apps represents applications that Astra recognizes. The Astra QA team has analyzed and confirmed these apps to be fully tested to restore.

Validated apps have also been checked by the Astra Development team, which creates custom workflows to help ensure the safety and consistency of your data. For example, when Astra takes a backup of a PostgreSQL database, it first quiesces the database. After the backup is complete, Astra restores the database to normal operation.

No matter which type of app you use with Astra, always test the backup and restore workflow yourself to ensure that you can meet your disaster recovery requirements.

Let us know what apps you would like to see validated in the future. Contact us through the Feedback email address on the Support page.

Standard Apps

Any other app, including custom programs, is considered a standard app. You can add and manage standard apps through Astra. You can also create basic crash-consistent Snapshots and Backups of a standard app. However, these have not been QA-tested to restore the app to its original state.

Define a custom app

Creating a custom app lets you group elements of your Kubernetes cluster into a single app.

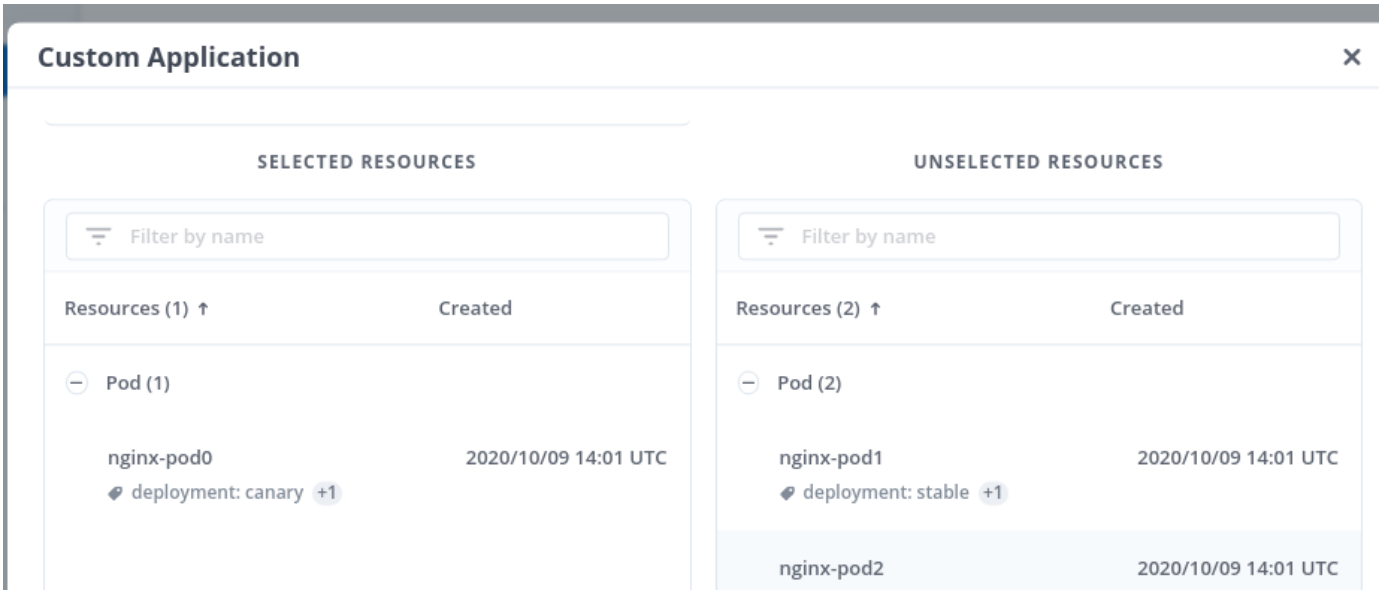
A custom app gives you more granular control over what to include in a Astra operation, including:

- Clone
- Snapshot
- Backup
- Protection Policy

In most cases you will want to use Astra's features on your entire app. However, you can also create a custom app to use these features by the labels you assign to Kubernetes objects in a namespace.

To create a custom app, go to the Apps page and click **+ Define custom app**.

As you make your selections, the Custom App window will show you which resources will be included or excluded from your custom app. This helps you make sure you are choosing the correct criteria for defining your custom app.



In the above example, one resource (the pod `nginx-pod0` labeled `deployment:canary`) will be included in the custom app. Two pods (`nginx-pod` and `nginx-pod2` both labeled `deployment:stable`) will be excluded.

Custom apps can only be created within a specified namespace on a single cluster. Astra does not support the ability for a custom app to span multiple namespaces or clusters.

A label is a key/value pair you can assign to Kubernetes objects for identification. Labels make it easier to sort, organize, and find your Kubernetes objects. To learn more about Kubernetes labels, see the official Kubernetes documentation.

Overlapping policies for the same resource under different names can cause data conflicts. If you create a custom app for a resource, be sure it's not being cloned or backed up under any other policies.

Example: Separate Protection Policy for canary release

In this example, the devops team is managing a canary release deployment. Their cluster has three pods running NginX. Two of the pods are dedicated to the stable release. The third pod is for the canary release.

The devops team's Kubernetes admin adds the label `deployment=stable` to the stable release pods. She adds the label `deployment=canary` to the canary release pod.

```
:~$ kubectl get pods --namespace=nginx-app --show-labels
NAME         READY   STATUS    RESTARTS   AGE   LABELS
nginx-pod0   1/1     Running   0          50s   deployment=canary,run=nginx-pod0
nginx-pod1   1/1     Running   0          45s   deployment=stable,run=nginx-pod1
nginx-pod2   1/1     Running   0          41s   deployment=stable,run=nginx-pod2
:~$
```

The team's stable release includes a requirement for hourly snapshots and daily backups. The canary release is more ephemeral, so they want to create a less aggressive, short-term Protection Policy for anything labeled `deployment=canary`.

In order to avoid possible data conflicts, the admin will create two custom apps: one for the canary release,

and one for the stable release. This keeps the backups, snapshots, and clone operations separate for the two groups of Kubernetes objects.

After she adds the cluster to Astra, her next step is to define a custom app. To do this, she clicks the **+ Define custom app** button on the Apps page.

In the pop-up window which appears, she sets `devops-canary-deployment` as the app name. She chooses the cluster in the **Compute** drop-down, then the app's namespace from the **Namespace** drop-down.

At this point, she can either type `deployment=canary` in the **Labels** field, or select that label from the resources listed below.

After defining the custom app for the canary deployment, she repeats the process for the stable deployment.

When she has finished creating the two custom apps, she can treat these resources as any other Astra application. She can clone them, create backups and snapshots, and create a custom Protection Policy for each group of resources based on her Kubernetes labels.