WHITE PAPER

Using FlashArray Snapshots with Microsoft SQL Server

How-to, technical details, best practices, and suggested use cases for volume snapshots with Microsoft SQL Server.

Contents

Executive Summary

As organizations experience rapid data growth, traditional data management approaches reliant on backups struggle to keep pace. These methods often fall short in terms of efficiency, cost-effectiveness, and speed, presenting challenges in maintaining data integrity and security amidst increasing threats like ransomware and data loss.

FlashArray™ snapshots offer a powerful solution to these challenges by providing a quick, reliable, and cost-effective way to manage data within Microsoft SQL Server environments. By utilizing snapshots at the storage layer, organizations can streamline data management, reduce storage sprawl, improve recovery times, and enhance data protection—all while mitigating the risks associated with data security.

This document explains how FlashArray snapshots integrate with SQL Server databases to address common data management needs, from copying and restoring databases to recovering from user errors, cloning databases, and seeding availability groups. Readers will gain insights into both foundational and advanced use cases, in addition to best practices for ensuring database consistency and troubleshooting snapshot issues. This paper serves as a practical guide for leveraging FlashArray snapshots to improve efficiency, security, and data control in SQL Server environments.

Solution Overview

This solution focuses on utilizing FlashArray block storage volumes with SQL Server databases. These hosts contain SQL Server user databases, including both data and log files, stored on FlashArray volumes. FlashArray snapshots can then be created for these volumes, providing point-in-time copies that capture the volume's data state.

Snapshots offer a versatile tool for managing SQL Server data: they can be used to recover data by restoring the original volume, to overwrite that data, to create copies on new volumes, to replicate asynchronously to another array, or to offload for long-term retention to object storage such as Amazon Simple Storage Service or Microsoft Azure Blob Storage. With these capabilities, FlashArray snapshots simplify data management, enhance recovery options, and support flexible, scalable storage practices for SQL Server environments.

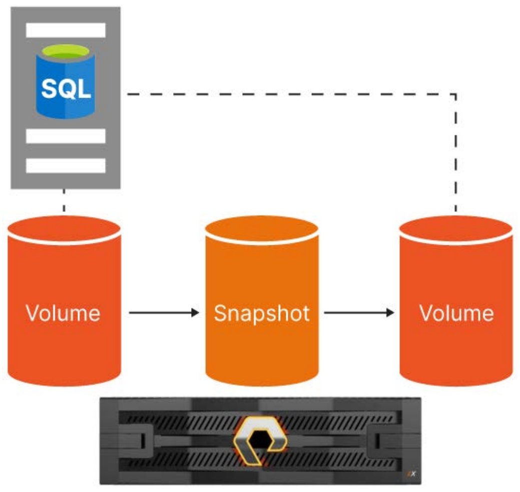


FIGURE 1 Using snapshots with SQL Server.

Solution Benefits

Using FlashArray volume snapshots with SQL Server provides the following benefits:

- **No additional licensing:** Snapshot capabilities are included with FlashArray at no extra cost, eliminating additional licensing fees.
- **Immutable, point-in-time images:** Snapshots create unchangeable, point-in-time images of data, ensuring data integrity and resilience against threats like ransomware. SafeMode enhances this protection with secure retention policies and recovery capabilities, preventing unauthorized modifications or deletions.
- **Space-efficiency**: Snapshots are thin-provisioned, deduplicated, and compressed, requiring only minimal storage space compared to full data copies, as they are metadata pointers to the data at a specific time.
- **Performance independent of volume of data:** FlashArray snapshots can be created and restored almost instantaneously, regardless of data volume, due to their design. This ensures that storage space requirements are separated from the time needed for snapshot creation and recovery, with no impact on performance, even as data grows.
- **Portability:** Snapshots can be easily transferred to another FlashArray, Pure Cloud Block Store™ instance, or third-party storage solution, such as Amazon Simple Storage Service or Microsoft Azure Blob Storage, enhancing flexibility and mobility.
- **Rapid recovery and flexibility:** Snapshots allow for near instantaneous restores, database cloning, and environment provisioning, reducing downtime and minimizing the effects of data loss or user errors.
- **Automated management and scheduling:** Snapshots support automation for regular backups and offloads, ensuring consistent protection without manual intervention.
- **Consistent protection across environments:** Snapshots maintain data-protection consistency across on-premises, cloud, and virtualized environments, simplifying hybrid strategies.
- **Data mobility for reporting, development and testing:** Snapshots can quickly generate copies of production data for development, testing, and analytics, accelerating application development cycles and supporting agile methodologies.

Technology Overview

The solution outlined in this white paper encompasses the following technology components:

- SQL Server instances running on Windows Server or a supported Linux operating system.
- Pure Storage® FlashArray as block storage for SQL Server user databases

The following section provides details of each technology used in this solution.

Microsoft SQL Server

SQL Server is a relational database management system that stores and retrieves data as requested by other software applications. Developed by Microsoft, SQL Server stores, retrieves, and manages large volumes of data for enterprise applications. SQL Server supports a wide range of workloads, from online transactional processing to complex online analytical processing. Data resiliency features, such as transparent data encryption and role-based security, help organizations keep their data safe both at rest and in transit. As a Microsoft product, SQL Server offers deep integration with other Microsoft enterprise-grade technologies, such as Active Directory and Azure.

Pure Storage FlashArray

Pure Storage FlashArray is a unified block- and file-storage solution designed to deliver a seamless and reliable user experience in SQL Server environments. Driven by software-defined technology, FlashArray incorporates high data reduction that does not compromise performance. FlashArray is ideal for organizations seeking to improve storage system sustainability.

The FlashArray product portfolio consists of:

- FlashArray//E™: Economical-at-scale storage for workloads that aren't latency-sensitive
- FlashArray//C™: Low-latency storage for capacity-oriented workloads
- FlashArray//X™: High-performance, high-capacity storage that is ideal for performance-oriented workloads
- FlashArray//XL™: High-performance storage at scale that helps reduce the number of arrays needed to run large applications

FlashArray Volumes, Snapshots, and Protection Groups

Snapshots are point-in-time images of a volume. A volume itself is a logical storage unit within a FlashArray. Both snapshots and volumes use metadata pointers to reference data blocks on the FlashArray. A protection group is designed to manage and automate the protection of multiple related storage objects, such as volumes. Snapshots can use protection groups to act on multiple volumes concurrently.

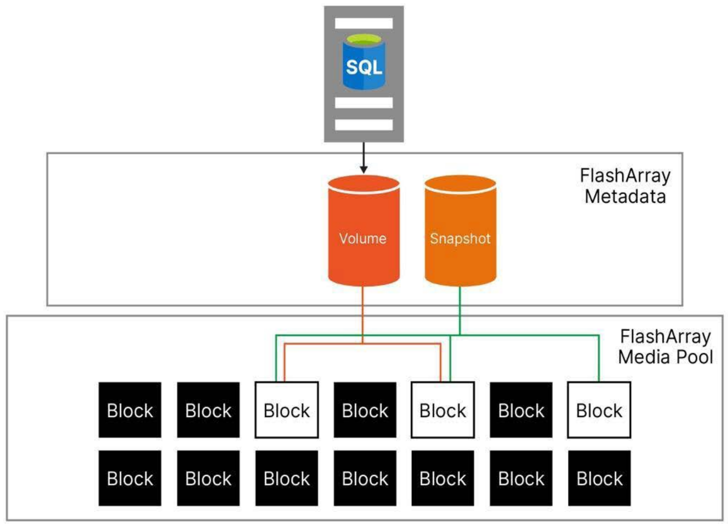


FIGURE 2 FlashArray volumes and snapshots use metadata pointers to reference data blocks on the FlashArray.

Volumes

A FlashArray volume is a block storage object that can be presented to a host, where it appears as a block device. Volumes store user data on a file system managed by the host's operating system. When a FlashArray volume is exported to a host, it allows applications like SQL Server to read and write data to this storage.

In FlashArray, the internal mapping of data for volumes differs from the way it is presented to the user. Rather than directly containing the user data, FlashArray volumes maintain metadata pointers that map to data blocks stored in the array's media pool. When a host writes data to a FlashArray volume, the data itself is stored in the media pool, and the volume's metadata is updated to reflect the location of each new data block. This separation means that the user sees a traditional block storage volume, but the actual storage is managed through metadata pointers, which optimizes storage efficiency and enables features like instant snapshots.

This unique architecture allows FlashArray to create snapshots that reference existing data blocks without duplicating them, making snapshots highly space-efficient.

Snapshots

Volume snapshots provide a fast, secure, and less-disruptive means to capture data at a particular point in time. When a volume snapshot is taken, the FlashArray makes a copy of a volume's metadata such that both the volume and snapshot are now pointing to the same data blocks, without copying the individual data blocks. The data blocks that are referenced by the volume and snapshot are read-only and cannot be changed, which makes the snapshot immutable.

If a SQL Server host requests a volume write that would change any data block that is referenced by a snapshot, that request is redirected and written as a new data block. The FlashArray updates the volume metadata to point to the new data block, while the previous data block is preserved and referenced by the snapshot.

FlashArray Volume Snapshots Advantages and Benefits

The ability to preserve data blocks referenced by a snapshot ensures that snapshots:

- Maintain their data integrity and immutability because referenced data blocks cannot be overwritten
- Are space-efficient because the FlashArray is maintaining metadata pointers to the data, not actual copies of the data
- Can be instantly provisioned by the FlashArray into new cloned volumes
- Are easily replicable by matching target blocks already present with those to be transferred and only transmitting the differences, which helps optimize bandwidth usage and helps significantly reduce replication time

These advantages are not just for data protection and recovery, but can aid in application development, testing, and replication. As a point-in-time reference to the volume data as it was when the snapshot occurred, snapshots have several practical uses in a SQL Server environment, such as:

- **Fast recovery of data in case of a ransomware attack**: If a volume's data becomes encrypted due to a ransomware attack, the volume can be instantly rolled back to a previous state using a snapshot. This helps prevent data loss and extended downtime.
- **Development or testing of applications where frequent rollback occurs**: Snapshots taken from a production volume can be attached to dev and test SQL Server hosts instead of using full backups to restore databases. These snapshots can be quickly refreshed in seconds, as opposed to hours, days, or longer with traditional backups.
- **Backup of the state of a database before a major upgrade:** Database and storage administrators can take a snapshot of the state of a volume where a user database data is stored and can instantly roll back a volume to the snapshot if problems occur during an update.
- **Offload CHECKDB operations without impacting performance:** Database maintenance tasks, such as CHECKDB, can be offloaded from a production SQL Server host by mounting a snapshot that contains the database data to another SQL Server host. The CHECKDB process can then be run on the snapshot on the other host, ensuring the integrity of the database without impacting the performance of the production host.
- **Efficient intra-instance extract, transform, and load and data movement**: Snapshots allow data to be duplicated within the instance for extract, transform, and load processes. By using snapshots for extract, transform, and load tasks, data transformations can occur independently of production databases, which reduces the risk of interference with production workloads.

Protection Groups

While a volume is a logical storage unit within a FlashArray, a protection group is designed to manage and automate the protection of multiple related storage objects, such as volumes and hosts.

Protection groups offer several advantages, including snapshot management and consistency. For example, protection groups provide for the scheduling of snapshots. When a snapshot is taken, all volumes within the protection group are snapped together to create a point-in-time consistency group across all included volumes.

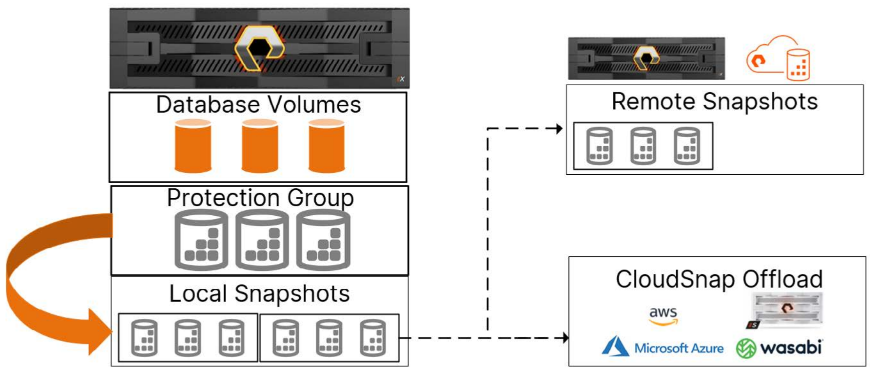


FIGURE 3 High-level overview of FlashArray protection groups, volumes, and snapshots.

Database and storage administrators can use protection groups and snapshots to create consolidated recovery points for SQL Server databases. If a SQL Server host uses multiple volumes to store user database data, a storage administrator can create a protection group that includes all of the user database volumes. Regular snapshots can be scheduled for the protection group, which creates instant recovery points for the user database data. If problems with a database occur, the storage administrator and database administrator can roll a volume back to a previous snapshot instantly, which reduces downtime when compared with restoring a full database backup.

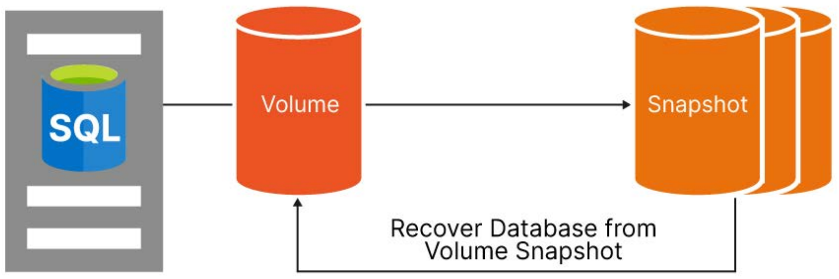


FIGURE 4 SQL Server database recovery via volume snapshots.

Traditional Backups Versus Crash-consistent and Application-consistent Snapshots

FlashArray supports crash-consistent and application-consistent snapshots for data protection and other data-management requirements with SQL Server. These types of snapshots differ from traditional SQL Server backups in significant ways, and the use cases will vary depending on how you are using them.

When to Use Application-consistent Snapshots	When to Use Crash-consistent Snapshots
• Recovering to a specific point-in-time that is facilitated by restoring log backups into a SQL Server database in restoring mode	• Restoring database objects • Refreshing non-production or reporting environments • As a substitute for offloaded data warehousing or reporting • Performing database consistency checks using DBCC CHECKDB

TABLE 1 When to use application-consistent snapshots or crash-consistent snapshots.

Traditional Backups

Traditional backups are a key component of the disaster recovery and data-protection management process and have several key benefits, as the backup:

- Contains a full copy of all database data
- Contains all active transaction log data up to the point of the backup
- Can be written out to a variety of archival media, such as tape, disk, and cloud storage

While these are important benefits, traditional backups also have certain drawbacks:

- Backing up or restoring large, multi-terabyte databases can take a considerable amount of time.
- Because the backup is a full copy of the database and active portion of the transaction log, it can require a large amount of storage space.
- As databases are constantly growing, so are the backups and the time required to take or restore those backups.
- Backups can consume CPU, memory, and network bandwidth resources on the host, which takes those resources away from application requests.

In contrast, FlashArray snapshots are storage-efficient, portable, and can be taken as near as instantaneous with no impact to the SQL Server host. Together, snapshots and traditional SQL Server backups can provide organizations with a comprehensive data-protection and data-portability strategy.

Crash-consistent Snapshots

Crash-consistent FlashArray snapshots are a point-in-time view of the database data at the time it was taken; therefore, the recovery point is the point at which the snapshot occurred. Crash-consistent snapshots can be used in any scenario for database recovery, assuming all database files are on volumes in a protection group that uses snapshots. Note that the term “crash-consistent” refers to a restore point being captured at the instant leading up to a server crashing or being powered off. In the case of FlashArray snapshots, this type of snapshot can be taken or scheduled as often as five minutes or taken manually at any time.

Crash-consistent snapshots are reliable because FlashArray avoids caching and does not reorder I/O operations. When a crash-consistent snapshot is taken, FlashArray respects the requirements of the SQL Server write-ahead logging protocol, ensuring the database remains recoverable. There is no impact to the SQL Server host, and the snapshot does not require that the database be quiesced, thereby avoiding delays that can occur when a database is quiesced. For more information about the SQL Server write-ahead logging protocol, see the section on Description of logging and data storage algorithms that extend data reliability in SQL Server.

Crash-consistent snapshots can provide reliable, rapid recovery of databases that only require a recovery point of when the snapshot is taken. For transactional-level recovery, application-consistent snapshots should be used. Other examples of the type of restore or recovery for crash-consistent snapshots include handling data loss due to user error, such as an accidental table deletion.

Application-consistent Snapshots

Application snapshots differ from crash-consistent snapshots in the following ways:

- The database is quiesced, which causes a brief pause to database input/output operations.
- Any application-consistent snapshot can be rolled forward to achieve a more granular point-in-time recovery with SQL transaction log backups, if they are available.

When transaction-level, point-in-time recovery is essential, an application-consistent snapshot can be taken of a database. Combined with SQL Server transaction log backups, the database can be restored to an exact point in time. Application-consistent snapshots ensure point-in-time recovery as long as SQL log backups are implemented as part of the restore.

SQL Server 2022 introduces T-SQL Snapshot Backup, a new feature that integrates with FlashArray and lets database administrators produce application-consistent snapshots transparently without the need of the Volume Shadow Copy Service.

T-SQL Snapshot Backup provides:

- **SQL Server-aware snapshots**: Because SQL Server controls the snapshot process, it records what's in the snapshots in its backup history.
- **Database quiescing with no external tools:** A single database, group of databases, or an entire SQL Server instance can be quiesced directly using T-SQL, which eliminates the need for external tools while maintaining consistency and recoverability.
- **A snapshot-based point for log restores**: Restoring to an exact recovery point can be done without having to read from a full backup. A snapshot can be restored, and then the log backups can be restored to an exact recovery point. This capability dramatically reduces recovery times while enabling exact recovery points.

For an example using T-SQL snapshots, see the Pure Storage script library: Point In Time Recovery - Using SQL Server 2022's T-SQL Snapshot Backup feature.

Alternatively, database administrators can use the Volume Shadow Copy Service provider for SQL Server 2019 and earlier. More information about the Volume Shadow Copy Service is available at Microsoft Learn.

Using FlashArray Volume Snapshots with SQL Server

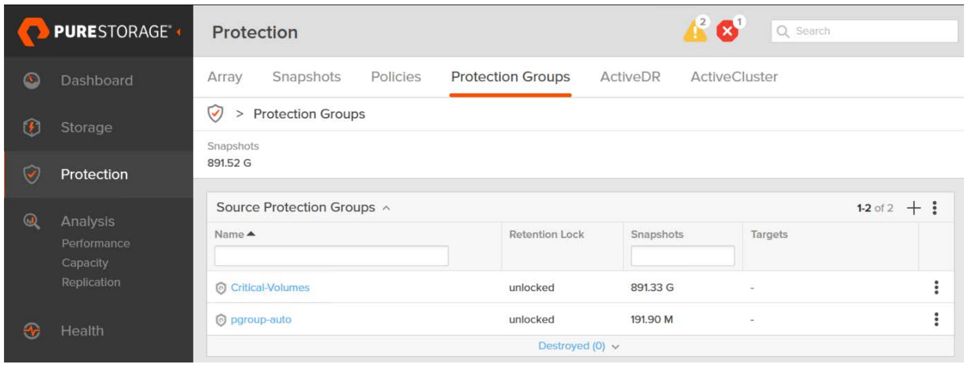
The following examples illustrate some of the more common tasks associated with data handling via FlashArray snapshots with SQL Server, including refreshing data, database-protection examples, restoring data, point-in-time recovery, cloning databases to new instances, and seeding availability groups.

Note: Some of the examples provide instructions that use the FlashArray user interface, while others use PowerShell commands. Providing tasks that use both methods gives you an overview of how to accomplish tasks using several different approaches. Additional Pure Storage test scripts for SQL Server are available from the Pure Storage OpenConnect SQL Server Scripts repository on GitHub. In the steps below, when creating a protection group for the volumes, it is imperative that all volumes that database files reside on are included in it.

Initial Steps: Create a Protection Group for Volumes

These initial steps demonstrate how to create a protection group for volumes on the array using the FlashArray user interface and T-SQL commands. This protection group will be used in examples for the following use cases.

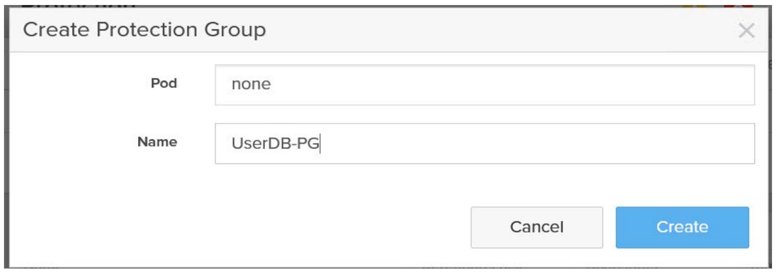
1. Create a protection group by clicking the **+** icon for **Source Protection Groups**.
   The **Create Protection Group** dialogue will appear.



Name	Retention Lock	Snapshots	Targets
Critical-Volumes	unlocked	891.33 G	-
pgroup-auto	unlocked	191.90 M	-

FIGURE 5 The Protection Groups tab in the FlashArray user interface.

2. Name the **Protection Group**, and then click **Create**.

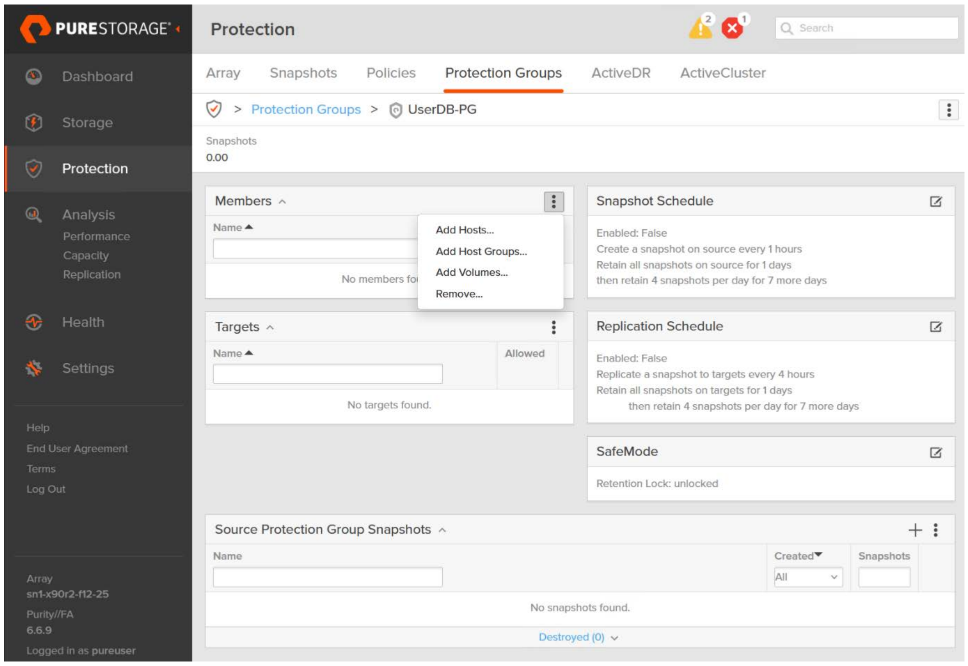


Pod: none

Name: UserDB-PG

FIGURE 6 The Create Protection Group dialogue.

3. To add volumes to the protection group, click the ellipsis icon for **Members**, and then select **Add Volumes**.



Snapshots
0.00

Snapshot Schedule
Enabled: False
Create a snapshot on source every 1 hours
Retain all snapshots on source for 1 days
then retain 4 snapshots per day for 7 more days

Replication Schedule
Enabled: False
Replicate a snapshot to targets every 4 hours
Retain all snapshots on targets for 1 days
then retain 4 snapshots per day for 7 more days

SafeMode
Retention Lock: unlocked

Array
sn1-x90r2-f12-25
Purity//FA
6.6.9
Logged in as pureuser

FIGURE 7 Adding volumes to a protection group.

4. Select all volumes relevant to the database. This should be done for both application- and crash-consistent snapshots.

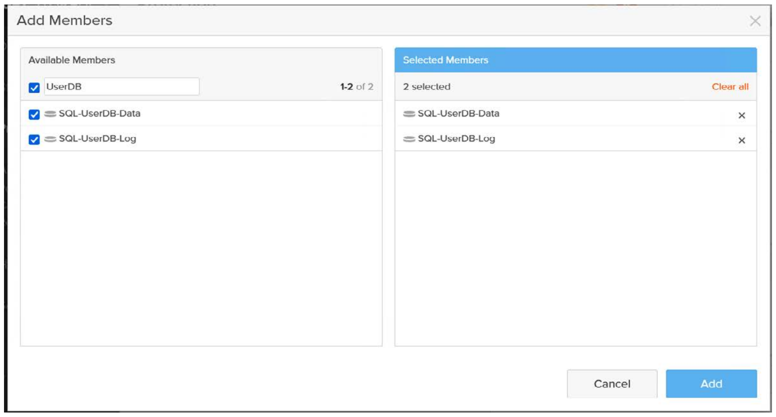
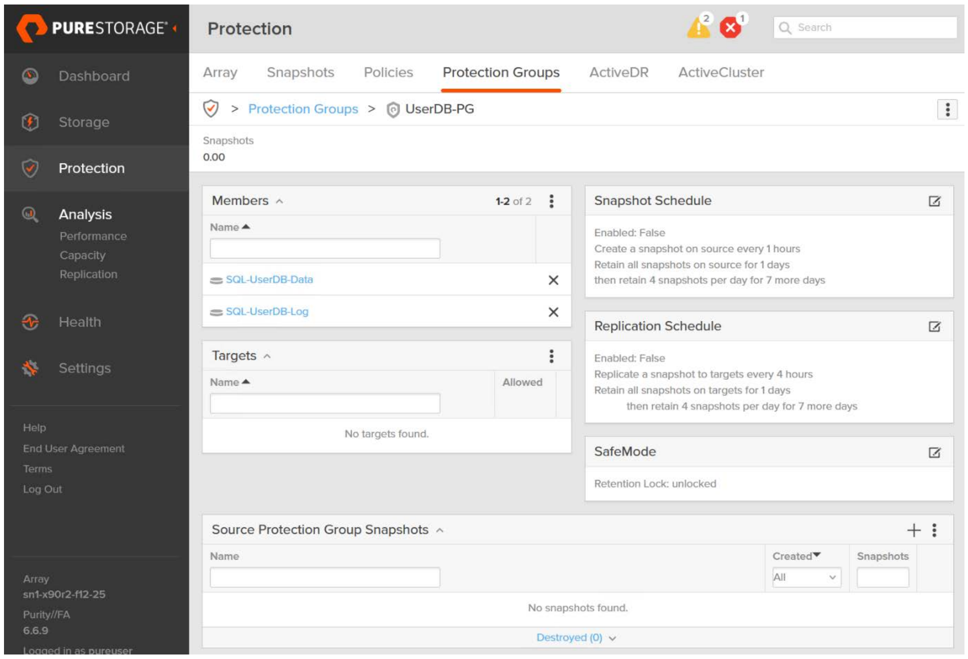


FIGURE 8 The Add Members dialogue.

5. After adding selected members to the protection group, use the example protection group “UserDB-PG” for all use case samples going forward.



Snapshot Schedule
Enabled: False
Create a snapshot on source every 1 hours
Retain all snapshots on source for 1 days
then retain 4 snapshots per day for 7 more days

Replication Schedule
Enabled: False
Replicate a snapshot to targets every 4 hours
Retain all snapshots on targets for 1 days
then retain 4 snapshots per day for 7 more days

SafeMode
Retention Lock: unlocked

FIGURE 9 The UserDB-PG protection group.

Crash-consistent Backup and In-Place Restore to the Same Host

Scenario: This use case demonstrates how to perform a crash-consistent backup and restore on the same host. The example shows how to use a crash-consistent snapshot to perform an in-place restore of a user database on the same SQL Server instance. This scenario recovers the protection group snapshot to the original volumes from where they were taken.

Prerequisites: A SQL Server instance running with a source database configured on a FlashArray protection group and volume. This example assumes the ability to manage snapshots directly within the same host environment.

Note: See the section on Initial Steps: Create a Protection Group for Volumes to create a protection group and add volume(s) to the protection group. PowerShell examples exist for automating this scenario on the SQL Server scripts GitHub repository.

Creating a Crash-consistent Snapshot

Using the previously created protection group, go to **Source Protection Group Snapshots**, click the ellipsis, and then select **Create** to create a snapshot.

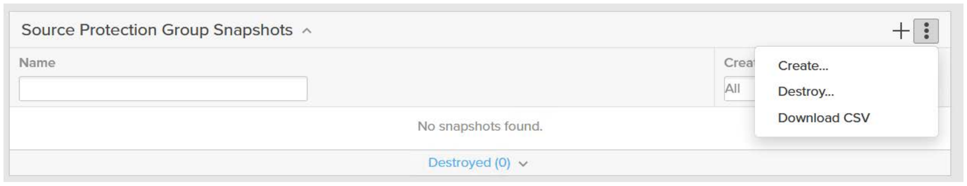
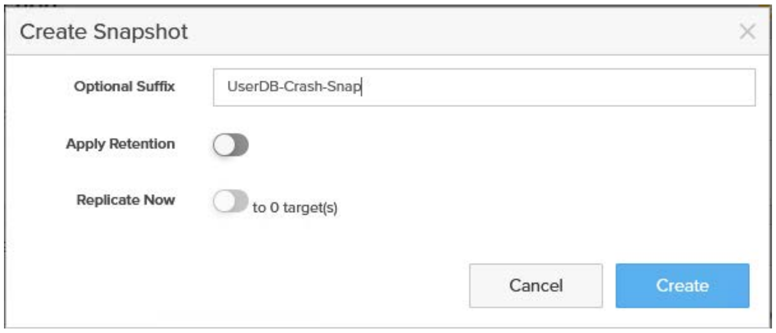


FIGURE 10 Creating a snapshot.

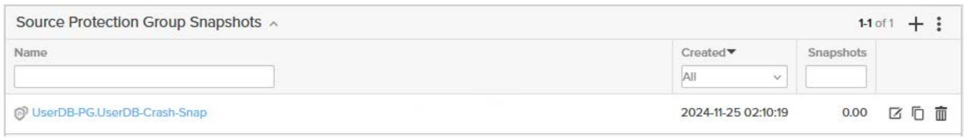
1. Enter **Optional Suffix** information for the snapshot's name (the snapshot name will default to "volume name.snapshot-number" without a suffix being set), and then click **Create**.



Optional Suffix: UserDB-Crash-Snap

FIGURE 11 Naming a snapshot in the **Create Snapshot** dialogue.

2. The snapshot name (with optional suffix) is displayed for Source Protection Group Snapshots.



Source Protection Group Snapshots 1-1 of 1

Name	Created	Snapshots
UserDB-PG.UserDB-Crash-Snap	2024-11-25 02:10:19	0.00

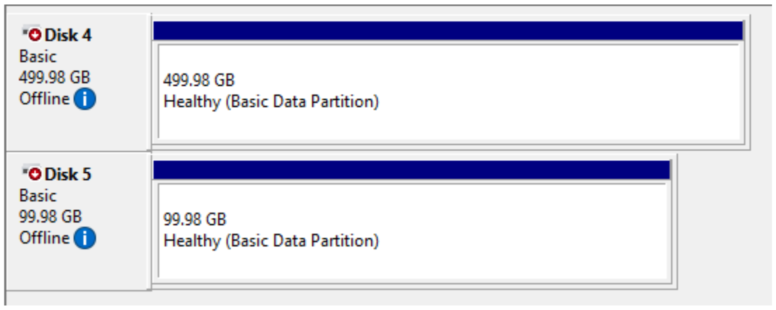
FIGURE 12 The snapshot created for the source protection group.

Recovering a Crash-consistent Snapshot to the Source Host

1. If the database being restored is available, online, and usable, ensure that it is set offline using the following command:

```
ALTER DATABASE [UserDB] OFFLINE WITH ROLLBACK IMMEDIATE;
```

2. The Windows disks that the volumes correspond to also need to be offline prior to restoring the volume snapshot. To offline the disks, in the Disk Management application on the host, right-click on the disk, and then select **Offline.**


Disk 4
Basic
499.98 GB
Offline

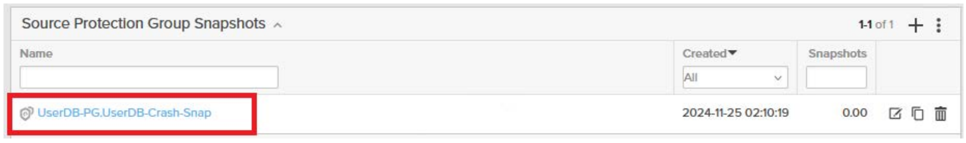
499.98 GB
Healthy (Basic Data Partition)

Disk 5
Basic
99.98 GB
Offline

99.98 GB
Healthy (Basic Data Partition)

FIGURE 13 Volumes attached to the SQL Server host as offline drives.

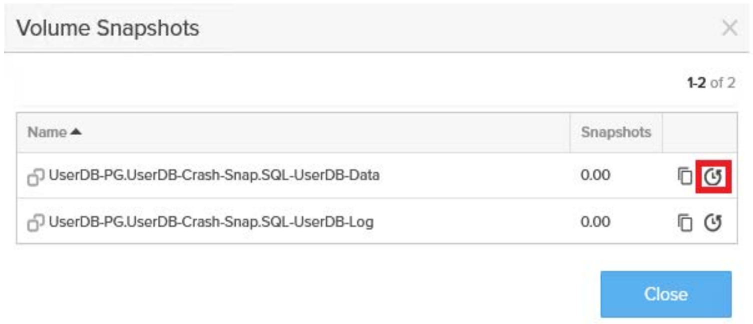
3. In the FlashArray user interface, select the protection group snapshot from the previously created snapshot, and then select it to open the detail view for the volume snapshots contained within it.


Source Protection Group Snapshots — 1-1 of 1

Name	Created	Snapshots
UserDB-PG.UserDB-Crash-Snap	2024-11-25 02:10:19	0.00

FIGURE 14 Selecting the protection group snapshot.

4. To restore the snapshots to the original volumes, click the overwrite icon.

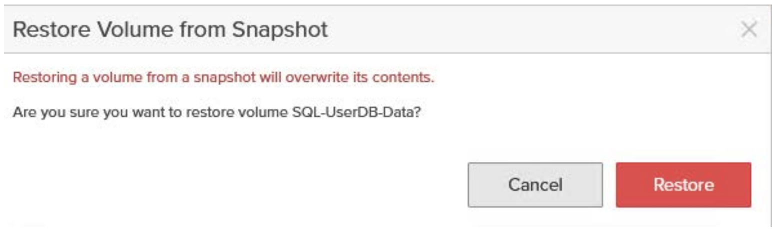

Volume Snapshots

1-2 of 2

Name	Snapshots
UserDB-PG.UserDB-Crash-Snap.SQL-UserDB-Data	0.00
UserDB-PG.UserDB-Crash-Snap.SQL-UserDB-Log	0.00

FIGURE 15 Restoring a snapshot using the overwrite icon.

5. In the **Restore Volume from Snapshot** prompt, review the volume name, and then select **Restore** to confirm.

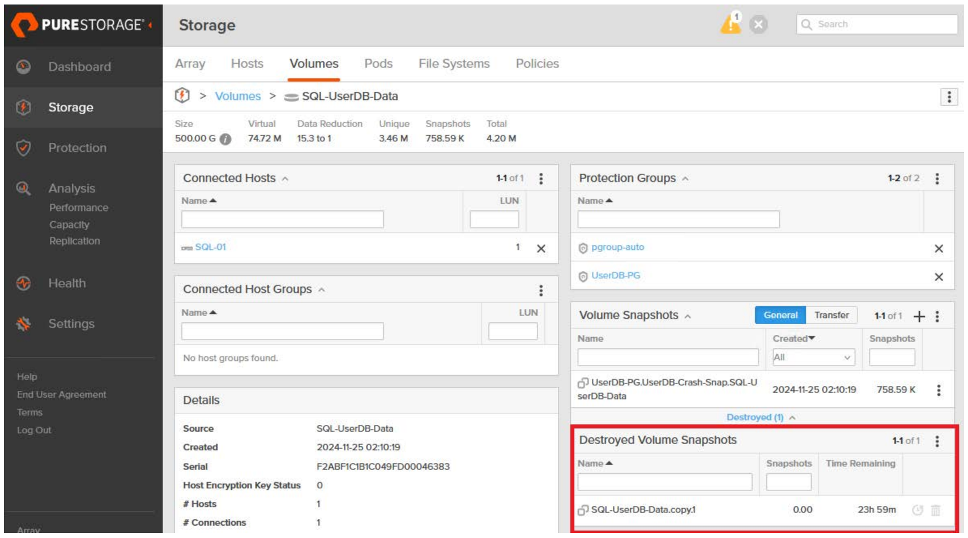

Restore Volume from Snapshot

Restoring a volume from a snapshot will overwrite its contents.

Are you sure you want to restore volume SQL-UserDB-Data?

FIGURE 16 The Restore Volume from Snapshot dialogue.

6. Repeat steps 4–5 for each volume in the protection group snapshot.
7. Overwriting volumes with snapshots does not immediately destroy the state of a volume when it is overwritten. Instead, a snapshot of the state of the volume is taken and then placed in the Destroyed Volume Snapshots section, which is then retained for a set period prior to being eradicated.



Size	Virtual	Data Reduction	Unique	Snapshots	Total
500.00 G	74.72 M	15.3 to 1	3.46 M	758.59 K	4.20 M

Connected Hosts 1-1 of 1

Name	LUN
SQL-01	1

Connected Host Groups

Name	LUN

No host groups found.

Details

Source	SQL-UserDB-Data
Created	2024-11-25 02:10:19
Serial	F2ABF1C1B1C049FD00046383
Host Encryption Key Status	0
# Hosts	1
# Connections	1

Protection Groups 1-2 of 2

Name
pgroup-auto
UserDB-PG

Volume Snapshots General Transfer 1-1 of 1

Name	Created	Snapshots
UserDB-PG.UserDB-Crash-Snap.SQL-UserDB-Data	2024-11-25 02:10:19	758.59 K

Destroyed (1)

Destroyed Volume Snapshots 1-1 of 1

Name	Snapshots	Time Remaining
SQL-UserDB-Data.copy1	0.00	23h 59m

FIGURE 17 The Destroyed Volume Snapshots section of the Volumes tab.

8. If the volumes are in an offline state, they need to be set online. In **Disk Management**, right-click each offline disk, and then select **Online**.

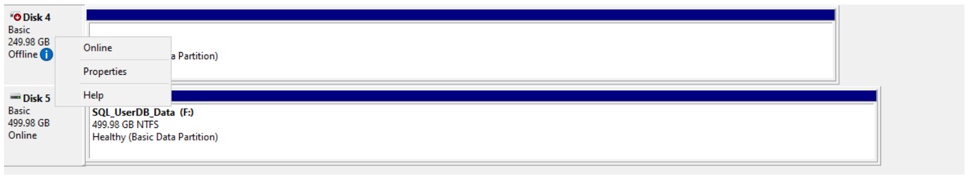


FIGURE 18 Bringing the disks online in the Windows Disk Manager.

9. If the disk does not have a drive letter assigned, right-click the disk, and then select **Change Drive Letter and Paths**.

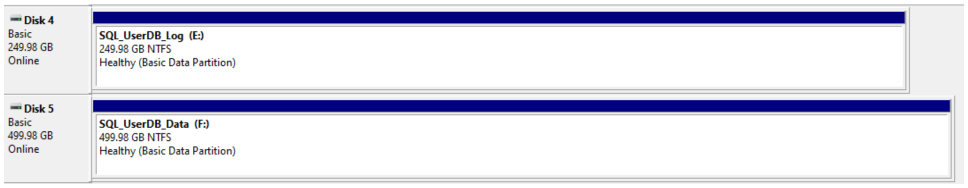


FIGURE 19 Volumes attached as drives on the SQL Server host.

10. If the disks come online in a read-only state, right-click each disk and set it to read/write.
11. If the database was set offline in step 1, and the database paths remain consistent, then set the database to online by using the following command:

```
ALTER DATABASE [UserDB] ONLINE;
```

12. If the database was deleted or is being recovered to a different set of paths, attach the database to the current instance by running the **CREATE DATABASE T-SQL** command with the **FOR ATTACH** property and the appropriate paths to each datafile.

```
CREATE DATABASE [UserDB_temp] ON
( FILENAME = 'E:\SQL\UserDB.mdf' ),
( FILENAME = 'F:\SQL\UserDB_log.ldf' )
 FOR ATTACH
```

Creating an Application-consistent Snapshot Backup

Scenario: This use case shows how to create an application-consistent snapshot for a protection group and volume(s) for a specific point in time. These steps capture the snapshot and a backup of the transaction log from SQL Server. This procedure should be used when a snapshot is needed for a FlashArray protection group and its volume(s). A SQL-native log backup is used to perform a point-in-time restore when the data is restored to the current location or a new location (for example, cloning). The benefit of using snapshots for backup in this way includes instantly recovering data, regardless of the size, without waiting for a full backup and restore.

Prerequisites: Running SQL Server 2022 or later with a database and FlashArray with a protection group and volume. For more information about T-SQL snapshot backups, see Create a Transact-SQL snapshot backup.

Usage notes: Each section of the example is meant to be run sequentially. Note that although this process uses the FlashArray user interface and T-SQL commands, the same result can be accomplished using PowerShell. Because this process freezes write input/output to the database, using PowerShell or other tools to automate this process is preferred so that database write input/output can resume as quickly as possible. See the Pure Storage OpenConnect SQL Server Scripts on GitHub for more information.

1. Prior to creating a Protection Group snapshot, the write operations on the database need to be suspended and the database prepared for a storage snapshot backup. The database will be frozen for write input with no impact to read output, until the **BACKUP DATABASE *[UserDB]* TO DISK…WITH METADATA_ONLY** command is run. This step should be automated to reduce the duration of the write input/output freeze. To do this, execute the following T-SQL command, substituting the database name for ***[UserDB]***:

```
ALTER DATABASE [UserDB] SET SUSPEND_FOR_SNAPSHOT_BACKUP = ON
```

2. Go to **Source Protection Group Snapshots**, click the ellipsis, and then select **Create** to create a snapshot.

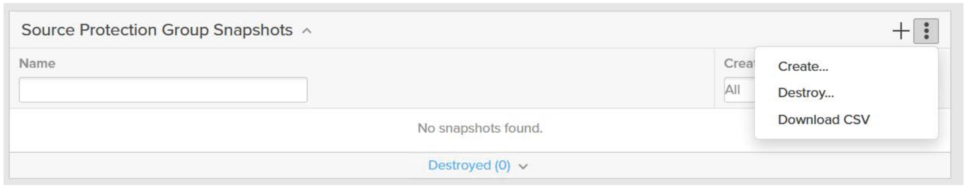


FIGURE 20 Creating a snapshot.

3. Enter **Optional Suffix** information for the snapshot's name, and then click **Create**. The snapshot name (with optional suffix) is displayed for Source Protection Group Snapshots. Take note of this snapshot name for reference in other examples in this document.

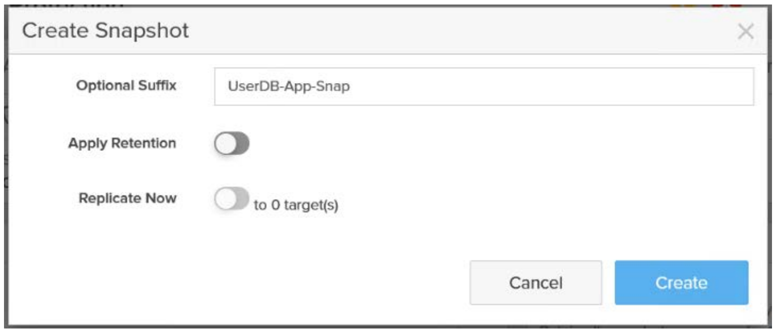
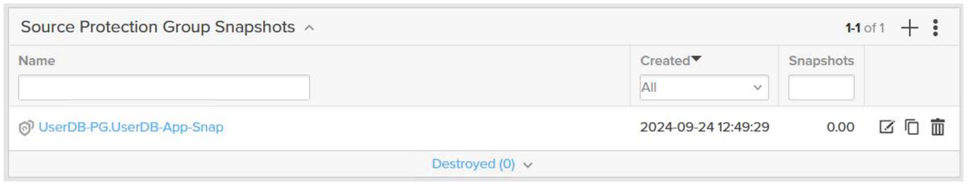


FIGURE 21 Naming a snapshot in the **Create Snapshot** dialogue.



Name	Created	Snapshots
UserDB-PG.UserDB-App-Snap	2024-09-24 12:49:29	0.00

FIGURE 22 The snapshot created for the source protection group.

4. After the protection group snapshot has been created, finish the snapshot backup process by executing the following command, substituting the database name and location for the metadata file (the backup file is required to put the database into restoring mode for point in time recovery):

```
BACKUP DATABASE [UserDB] TO DISK='C:\SQL\UserDB.bkm' WITH METADATA_ONLY
```

Creating Manual Log Backups

A log backup is only required if you are performing a point-in-time recovery. Log backups are scheduled separately from snapshot backups; however, if a manual log backup needs to be taken, execute the following command, substituting ***[UserDB]*** for the relevant database name:

```
BACKUP LOG [UserDB] FROM DISK = 'C:\SQL\UserDB-log.bkm'
```

Recovering An Application-consistent Snapshot Backup to the Same Host

Scenario: Snapshots are highly efficient and quick when data must be restored directly to the original database instance or host. These are the steps to recover data from an application-consistent snapshot to the original volume on the same array attached to the original host. This scenario uses T-SQL snapshot backups, introduced in SQL Server 2022.

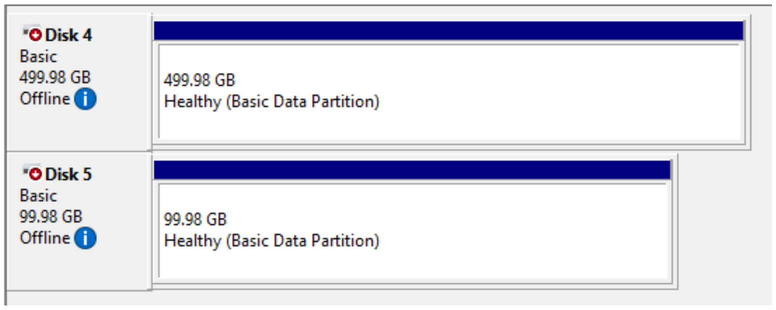
Prerequisites: Running SQL Server 2022 or later with a database and FlashArray with volumes assigned to a protection group.

Usage notes: Each section of this example is meant to be run sequentially.

1. If the database being restored is available, online, and usable, ensure that it is set offline using the following command:

```
ALTER DATABASE [UserDB] SET OFFLINE WITH ROLLBACK IMMEDIATE
```

2. The Windows disks that the volumes correspond to also need to be offline prior to the volume snapshot being restored. To offline the disks, in the **Disk Management** application on the host, right-click on the disk, and then select **Offline**.


Disk 4
Basic
499.98 GB
Offline

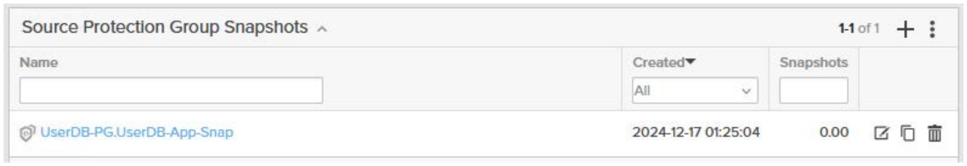
499.98 GB
Healthy (Basic Data Partition)

Disk 5
Basic
99.98 GB
Offline

99.98 GB
Healthy (Basic Data Partition)

FIGURE 23 Volumes attached to the SQL Server host as offline drives.

3. In the FlashArray user interface, select the protection group snapshot, and then select the snapshot link to open the detail view for the volume snapshots contained within the protection group.

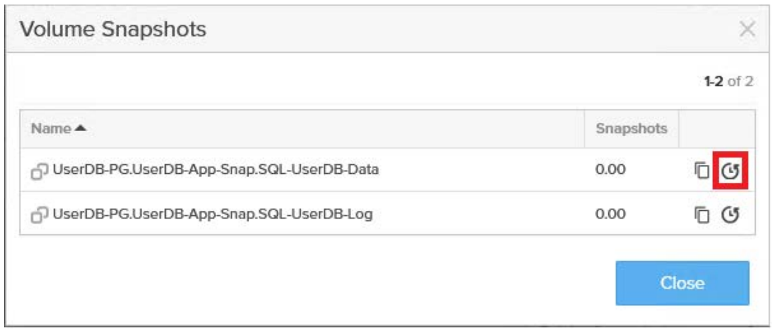

Source Protection Group Snapshots

1-1 of 1

Name	Created	Snapshots
UserDB-PG.UserDB-App-Snap	2024-12-17 01:25:04	0.00

FIGURE 24 The snapshot created for the source protection group.

4. To restore the snapshots to the original volumes, select the overwrite icon.

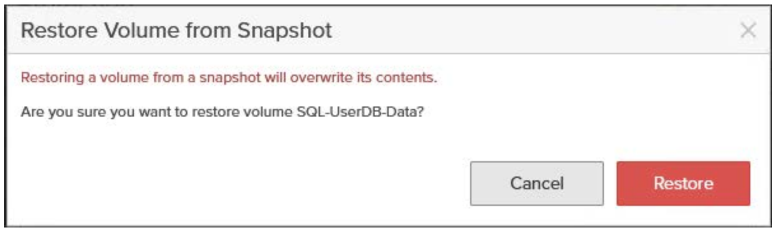

Volume Snapshots

1-2 of 2

Name	Snapshots
UserDB-PG.UserDB-App-Snap.SQL-UserDB-Data	0.00
UserDB-PG.UserDB-App-Snap.SQL-UserDB-Log	0.00

FIGURE 25 Restoring a snapshot using the overwrite icon.

5. At the **Restore Volume from Snapshot** prompt, review the volume name, and then select **Restore** to confirm.


Restore Volume from Snapshot

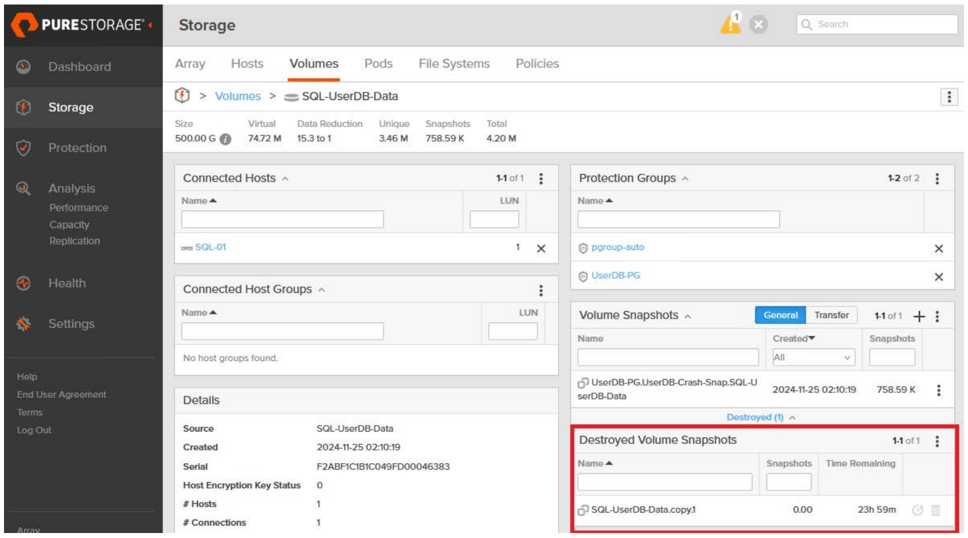
Restoring a volume from a snapshot will overwrite its contents.

Are you sure you want to restore volume SQL-UserDB-Data?

FIGURE 26 The Restore Volume from Snapshot dialogue.

6. Repeat steps 4–5 for each volume in the protection group snapshot.

7. Overwriting volumes with snapshots does not immediately destroy the state of the volume when it is overwritten. Instead, a snapshot of the state of the volume is taken and then placed in the Destroyed Volume Snapshots section, which is then retained for a set period prior to being eradicated.



> Volumes > SQL-UserDB-Data

Size	Virtual	Data Reduction	Unique	Snapshots	Total
500.00 G	74.72 M	15.3 to 1	3.46 M	758.59 K	4.20 M

Connected Hosts 1-1 of 1

Name	LUN
SQL-01	1

Connected Host Groups

Name	LUN
No host groups found.	

Details

Source	SQL-UserDB-Data
Created	2024-11-25 02:10:19
Serial	F2ABF1C1B1C049FD00046383
Host Encryption Key Status	0
# Hosts	1
# Connections	1

Protection Groups 1-2 of 2

Name
pgroup-auto
UserDB-PG

Volume Snapshots General Transfer 1-1 of 1

Name	Created	Snapshots
UserDB-PG.UserDB-Crash-Snap.SQL-UserDB-Data	2024-11-25 02:10:19	758.59 K

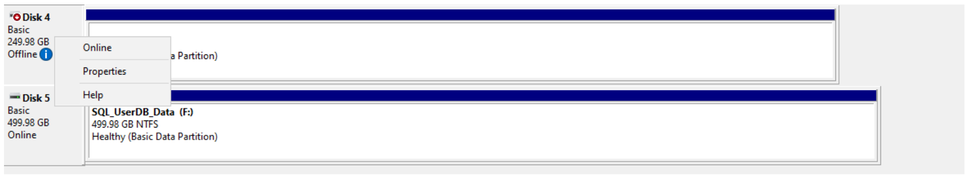
Destroyed (1)

Destroyed Volume Snapshots 1-1 of 1

Name	Snapshots	Time Remaining
SQL-UserDB-Data.copy.1	0.00	23h 59m

FIGURE 27 The Destroyed Volume Snapshots section of the Volumes tab.

8. If step 2 has previously set the volumes to offline, these volumes now need to be set online once more. In **Disk Management**, right-click each offline disk, and then select **Online**.

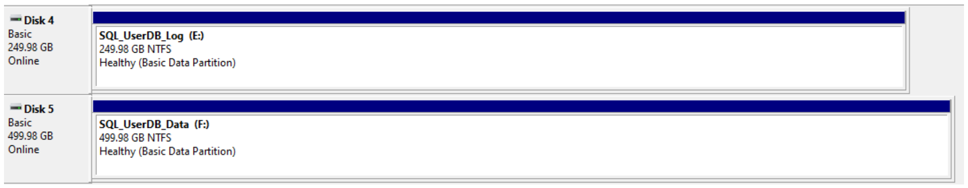

Disk 4
Basic
249.98 GB
Offline

Disk 5
Basic
499.98 GB
Online

SQL_UserDB_Data (F:)
499.98 GB NTFS
Healthy (Basic Data Partition)

FIGURE 28 Bringing the disks online in Windows Disk Manager.

9. If the disk does not have a drive letter assigned, right-click the disk, and then choose **Change Drive Letter and Paths**.


Disk 4
Basic
249.98 GB
Online

SQL_UserDB_Log (E:)
249.98 GB NTFS
Healthy (Basic Data Partition)

Disk 5
Basic
499.98 GB
Online

SQL_UserDB_Data (F:)
499.98 GB NTFS
Healthy (Basic Data Partition)

FIGURE 29 Volumes attached as drives on the SQL Server host.

10. If the disks come online in a read-only state, right-click each disk and set it to read/write.
11. Restore the database with METADATA only using the NORECOVERY option:

```
RESTORE DATABASE [UserDB] FROM DISK = 'E:\SQL\UserDB.mdf' with METADATA_ONLY, REPLACE, NORECOVERY;
```

12. (Optional) For point-in-time recovery, restore the transaction log to the database on the new SQL Server host by executing the following command on the new host. Replace the database name and transaction log backup location with the relevant values:

```
RESTORE LOG [UserDB] FROM DISK = 'C:\SQL\UserDB-log.bkm' WITH NORECOVERY;
```

13. Finish the database snapshot restore:

```
RESTORE DATABASE [UserDB] with RECOVERY;
```

Using Crash-consistent Snapshots to Clone to a New User Database

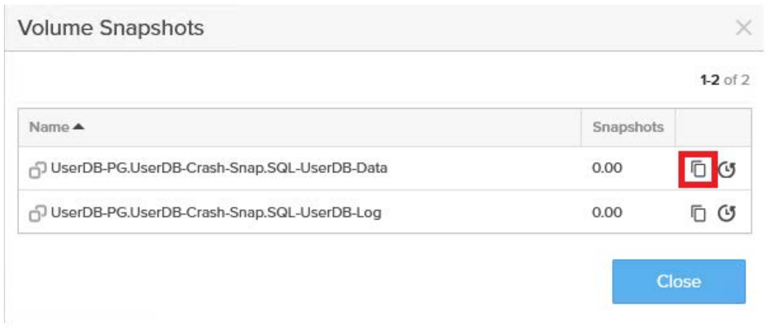
Scenario: FlashArray snapshots offer a fast and efficient method to clone SQL Server databases for a variety of use cases. As database sizes increase, administrators can use crash-consistent snapshots to reduce the complexity of copying large amounts of data and cloning that data for different consumers. For example, the volumes supporting one or more databases can be cloned to different SQL Server database instances for development, test, and reporting environments.

Prerequisites: Running SQL Server with a database and FlashArray with a protection group and volume. This procedure is based on a Pure Storage volume database refresh script that is available on GitHub.

Note: These examples clone snapshots to new volumes on the same array.

See the section on Initial Steps: Create a Protection Group for Volumes to create a protection group and add volume(s) to the protection group.

1. See the section on Creating a Crash-consistent Snapshot to create a crash-consistent snapshot.
2. To copy the volumes, select the protection group snapshot from the previously created snapshot, and then select the copy icon to copy to a new volume.

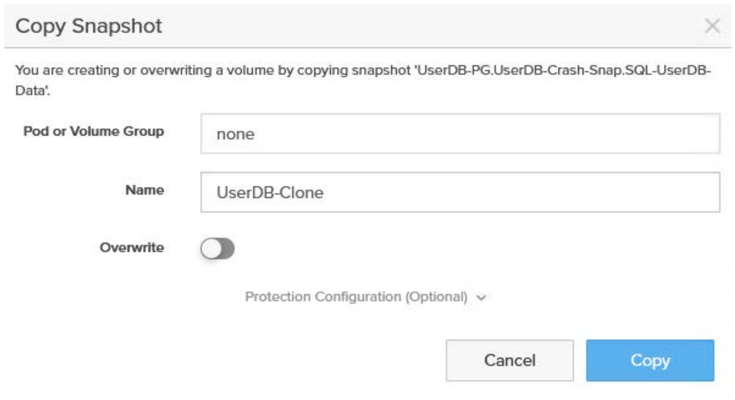

Volume Snapshots

1-2 of 2

Name ▲	Snapshots
UserDB-PG.UserDB-Crash-Snap.SQL-UserDB-Data	0.00
UserDB-PG.UserDB-Crash-Snap.SQL-UserDB-Log	0.00

FIGURE 30 Restoring a snapshot using the copy icon.

3. Copy the selected snapshot to a new volume by creating a name and then clicking **Copy**.


Copy Snapshot

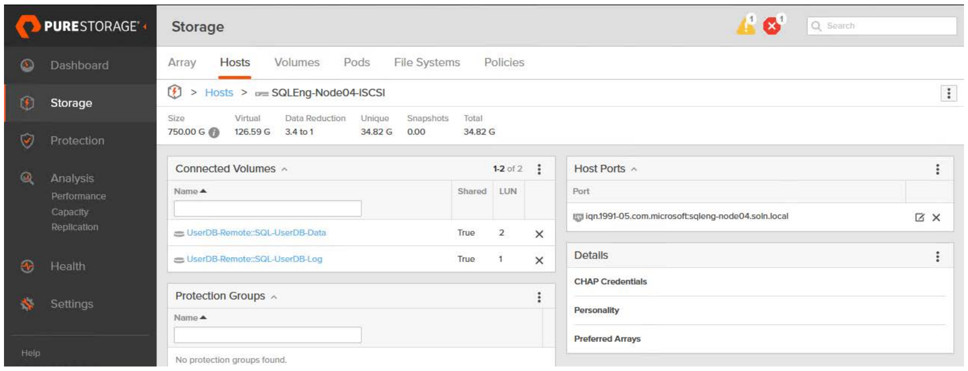
You are creating or overwriting a volume by copying snapshot 'UserDB-PG.UserDB-Crash-Snap.SQL-UserDB-Data'.

Pod or Volume Group	none
Name	UserDB-Clone
Overwrite	

FIGURE 31 The Copy Snapshot dialogue.

4. Repeat steps 3 and 4 for each volume in the protection group, providing a unique name for each volume.

5. Once the snapshot is cloned/copied (a data reduced copy), verify that a new host or hosts are available. These hosts can be part of a dev or test environment or a reporting server. In the FlashArray user interface, select **Storage** in the navigation pane, and then click the **Hosts** tab.



Hosts > SQLEng-Node04-ISCSI

Size	Virtual	Data Reduction	Unique	Snapshots	Total
750.00 G	126.59 G	3.4 to 1	34.82 G	0.00	34.82 G

Connected Volumes 1-2 of 2

Name	Shared	LUN
UserDB-Remote::SQL-UserDB-Data	True	2
UserDB-Remote::SQL-UserDB-Log	True	1

Protection Groups

Name

No protection groups found.

Host Ports

Port

iqn.1991-05.com.microsoft:sqleng-node04.soln.local

Details

CHAP Credentials

Personality

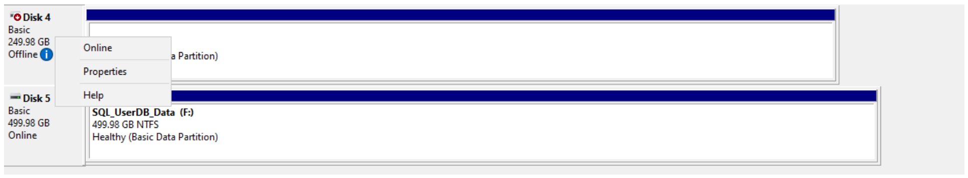
Preferred Arrays

FIGURE 32 The Hosts tab in the FlashArray user interface.

6. Verify that the new host appears in the **Hosts** group, and then attach the host from the **Hosts** tab.

7. On the new host, open the Disk Management application. The volumes appear as offline disks.

8. Right-click each offline disk, and then select **Online**.


Disk 4
Basic
249.98 GB
Offline

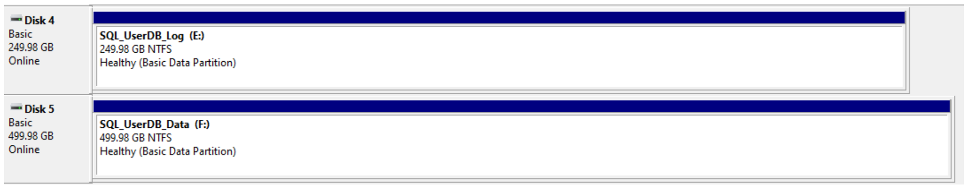
a Partition)

Disk 5
Basic
499.98 GB
Online

SQL_UserDB_Data (F:)
499.98 GB NTFS
Healthy (Basic Data Partition)

FIGURE 33 Bringing the disks online in the Windows Disk Manager.

9. If the disk does not have a drive letter assigned, right-click the disk, and then choose **Change Drive Letter and Paths**.


Disk 4
Basic
249.98 GB
Online

SQL_UserDB_Log (E:)
249.98 GB NTFS
Healthy (Basic Data Partition)

Disk 5
Basic
499.98 GB
Online

SQL_UserDB_Data (F:)
499.98 GB NTFS
Healthy (Basic Data Partition)

FIGURE 34 Volumes attached as drives on the SQL Server host.

10. If the disks come online in a read-only state, right-click each disk and set it to read/write.
11. Attach the database to the current instances by running the **CREATE DATABASE T-SQL** command with the **FOR ATTACH** property and the appropriate paths to each datafile.

```
CREATE DATABASE [UserDB] ON
( FILENAME = 'E:\SQL\UserDB.mdf' ),
( FILENAME = 'F:\SQL\UserDB_log.ldf' )
 FOR ATTACH
```

The database is now available on the new host. The snapshot can now be cloned to other dev, test, or reporting SQL Server hosts.

Seeding an Availability Group from a Snapshot

Scenario: SQL Server availability groups provide high availability and disaster recovery for SQL Server databases by enabling database administrators to replicate databases across multiple SQL Server instances and create a failover environment for a group of databases. However, the process of seeding replicas can be time-consuming. Even on fast networks, the sheer size of the data involved in the operation can result in seeding secondary replicas taking hours or even days. This challenge is further exacerbated if the primary replica contains very large databases or if seeding occurs across slow or wide area networks between data centers.

Pure Storage snapshots and the new T-SQL snapshot backup capability in SQL Server 2022 can help alleviate the difficulties in seeding secondary replicas.

Note: Several steps in this process use PowerShell commands. For more information, refer to the Pure Storage script library at Seeding an Availability Group - Using SQL Server 2022's T-SQL Snapshot Backup feature.

Prerequisites:

- **An existing SQL Server availability group:** This example requires at least two SQL Server 2022 replicas configured in an availability group, with one server acting as the primary replica, and the other acting as the secondary replica.
- **Volumes assigned to a protection group**: See the section on Initial Steps: Create a Protection Group for Volumes to create a protection group and add volumes to the protection group. These volumes are already attached to the hosts and assumed to be online.
- **Asynchronous snapshot replication between two FlashArray systems**: The FlashArray systems replicate the volumes in the protection group.
- **Log backups disabled**: Log backups must be disabled on the primary replica during the seeding process because the log backup must be restored on the secondary replica for the availability group to synchronize.
- **PowerShell modules:** This procedure was implemented using the Pure Storage PowerShell SDK2 and dbatools modules. The dbatools modules were used to open and hold a connection to the SQL Server instance.

Setting Up the Environment

The first step is to load the Pure Storage PowerShell SDK2 and dbatools modules, get the credentials that are used to connect to a FlashArray, and assign values to variables. Note that the variable values shown here are examples only; variable values will be different in your environment.

```
Import-Module dbatools
Import-Module PureStoragePowerShellSDK2

$Credential                     = Get-Credential                    # Get the user credentials
$PrimarySqlServer                       = 'SqlServer1.example.com'          # SQL Server primary replica
$SecondarySqlServer             = 'SqlServer2.example.com'          # SQL Server secondary replica
$AgName                         = 'ag1'                     # Name of the availability group
$DbName                         = 'UserDB'                  # Name of database to place in the
                                                            # availability group
$BackupShare                    = '\\FileServer1\SHARE\BACKUP'  # File location for metadata backup file.
$PrimaryArrayName                       = 'flasharray1.example.com'         # FlashArray containing the
volumes for
                                                            # the primary replica
$SecondaryArrayName             = 'flasharray2.example.com'         # FlashArray containing the volumes
for
                                                            # the secondary replica
$SourcePGroupName               = 'SqlServer1_Pg'                   # Name of the protection group on
                                                            # primary array
$TargetPGroupName               = 'flasharray1:SqlServer1_Pg'       # Name of the protection group repli-
cated
                                                            # from FlashArray1 to FlashArray2,
                                                    # formatted as
                                                            # ArrayName:ProtectionGroupName
$PrimaryFaDbVol                 = 'Fa1_Sql_Volume_1'                 # Volume name on FlashArray contain-
ing
                                                            # the primary replica database files
$SecondaryFaDbVol                        = 'Fa2_Sql_Volume_1'        # Volume name on FlashArray contain-
ing
                                                            # the primary replica database files
$TargetDisk                     = 'serialnum'               # The serial number of the Windows
                                                            # volume containing database files
```

Taking a Snapshot on the Primary Replica FlashArray Protection Group

The next step is to take a snapshot on the FlashArray that contains the primary replica's database file. This is done by freezing the write input/output on the database, taking a snapshot of the protection group that replicates to the secondary replica's FlashArray, and then creating a metadata backup that contains the contents of the backup.

The following commands connect to the primary replica FlashArray, pause the write input/output on the primary replica, and take a snapshot of the primary replica's protection group. The final command takes a metadata backup of the database and unfreezes the database if it is successful.

```
# Connect to the primary replica FlashArray
$FlashArrayPrimary = Connect-Pfa2Array –EndPoint $PrimaryArrayName -Credential $Credential -IgnoreCertifica-
teError

# Suspend write operations to the database on the primary replica
$Query = "ALTER DATABASE [$DbName] SET SUSPEND_FOR_SNAPSHOT_BACKUP = ON"
Invoke-DbaQuery -SqlInstance $SqlInstancePrimary -Query $Query -Verbose

# Create a snapshot of the primary replica's protection group
$SourceSnapshot = New-Pfa2ProtectionGroupSnapshot -Array $FlashArrayPrimary -SourceName $SourcePGroupName
-ForReplication $true -ReplicateNow $true

# Create a metadata backup, which will unfreeze the database if successful
$BackupFile = "$BackupShare\$DbName$(Get-Date -Format FileDateTime).bkm"
$Query = "BACKUP DATABASE $DbName
        TO DISK='$BackupFile'
        WITH METADATA_ONLY, MEDIADESCRIPTION='$($SourceSnapshot.Name)|$($FlashArrayPrimary.ArrayName)'"
Invoke-DbaQuery -SqlInstance $SqlInstancePrimary -Query $Query -Verbose
```

Replicating the Snapshot from the Primary Replica FlashArray to the Secondary Replica FlashArray

Once the protection group snapshot is created, the following commands connect to the secondary replica's FlashArray and verify that the snapshot has completed replication, once complete the process moves onto the next steps. Note that the exact protection group name from the primary replica's FlashArray must be used to extract the most recent snapshot name, and that the replication to the secondary replica's array must be completed prior to completing the next steps.

```
# Connect to the secondary replica FlashArray
$FlashArraySecondary = Connect-Pfa2Array –EndPoint $SecondaryArrayName -Credential $Credential -IgnoreCer-
tificateError
#
Write-Warning "Obtaining the most recent snapshot for the protection group..."
$TargetSnapshot = $null
do {
  Write-Warning "Waiting for snapshot to replicate to target array..."
  $TargetSnapshot = Get-Pfa2ProtectionGroupSnapshotTransfer -Array $FlashArraySecondary -Name $TargetP-
GroupName |
      Where-Object { $_.Name -eq "$TargetPGroupName.$($SourceSnapshot.Suffix)" }
  if ( $TargetSnapshot -and $TargetSnapshot.Progress -ne 1.0 ){
    Write-Warning "Snapshot $($TargetSnapshot.Name) found on Target Array...replication progress is $($Tar-
getSnapshot.Progress)"
    Start-Sleep 3
  }
} while ( [string]::IsNullOrEmpty($TargetSnapshot.Completed) -or ($TargetSnapshot.Progress -ne 1.0) )
Write-Warning "Snapshot $($TargetSnapshot.Name) replicated to Target Array. Completed at $($TargetSnapshot.
Completed)"
```

Cloning the Volumes on the Secondary Replica FlashArray

Once the snapshot is fully replicated, proceed to clone the volumes from the snapshot on the secondary replica FlashArray. To do this, offline the volumes that the secondary replica is currently using, clone the volumes from the snapshot, and then bring the volumes back online.

```
# Connect to the secondary replica FlashArray and take the volume offline
Invoke-Command -Session $SecondarySession -ScriptBlock { Get-Disk | Where-Object { $_.SerialNumber -eq $us-
ing:TargetDisk } | Set-Disk -IsOffline $True }

# Overwrite the volumes on the secondary replica FlashArray with the snapshot
New-Pfa2Volume -Array $FlashArraySecondary -Name $SecondaryFaDbVol -SourceName ($TargetSnapshot.Name +
".$PrimaryFaDbVol") -Overwrite $true

# Bring the volume online on the secondary replica FlashArray
Invoke-Command -Session $SecondarySession -ScriptBlock { Get-Disk | Where-Object { $_.SerialNumber -eq $us-
ing:TargetDisk } | Set-Disk -IsOffline $False }
```

Restoring the Database and Log Backup on the Secondary Replica

Now that clones have been created from the primary replica FlashArray, the database can be restored on the secondary replica. To do this, the database is restored from the cloned volume with the NORECOVERY option. This option is important because an availability group replica is being seeded. A log backup from the primary replica is also performed and then restored to the secondary replica.

```
# Restore the database with the NORECOVERY option
$Query = "RESTORE DATABASE [$DbName] FROM DISK = '$BackupFile' WITH METADATA_ONLY, REPLACE, NORECOVERY"
Invoke-DbaQuery -SqlInstance $SqlInstanceSecondary -Database master -Query $Query -Verbose
# Take a log backup from the primary replica
$Query = "BACKUP LOG [$DbName] TO DISK = '$BackupShare\$DbName-seed.trn' WITH FORMAT, INIT"
Invoke-DbaQuery -SqlInstance $SqlInstancePrimary -Database master -Query $Query -Verbose
# Restore the log backup to the secondary replica
$Query = "RESTORE LOG [$DbName] FROM DISK = '$BackupShare\$DbName-seed.trn' WITH NORECOVERY"
Invoke-DbaQuery -SqlInstance $SqlInstanceSecondary -Database master -Query $Query -Verbose
```

Start the Seeding on the Secondary Replica

With the database and log backup restored to the secondary replica, the availability group seeding process can now be started and the database added to the availability group. The status of the availability group is also checked to see if it is synchronized between the primary replica and secondary replica.

```
# Set the seeding mode on the secondary replica to manual
$Query = "ALTER AVAILABILITY GROUP [$AgName] MODIFY REPLICA ON N'$PrimarySqlServer' WITH (SEEDING_MODE =
MANUAL)"
Invoke-DbaQuery -SqlInstance $SqlInstancePrimary -Database master -Query $Query -Verbose

# Add the database to the availability group
$Query = "ALTER AVAILABILITY GROUP [$AgName] ADD DATABASE [$DbName];"
Invoke-DbaQuery -SqlInstance $SqlInstancePrimary -Database master -Query $Query -Verbose
# Start the availability group replication process
Invoke-DbaQuery -SqlInstance $SecondarySqlInstance -Database master -Query 'ALTER DATABASE [$DbName] SET
HADR AVAILABILITY GROUP = [ag1];'

# Check the status of the availability group to see if the primary and secondary replicas are synchronized
Get-DbaAgDatabase -SqlInstance $SqlInstancePrimary -AvailabilityGroup $AgName
```

Once the availability group status shows that the replicas are synchronized, the availability group is functioning.

Performing Database Consistency Checks on Snapshots

Performing database consistency checks on snapshots helps ensure the integrity and reliability of the databases on a SQL Server host while offloading the often-lengthy task onto a secondary SQL Server host. Key reasons why using a volume snapshot on a secondary SQL Server host is more efficient include:

- **Offloading the workload:** CHECKDB is a resource-intensive operation that can significantly impact CPU and memory resources, thereby degrading performance for end-user workloads on the primary SQL Server host. By offloading the consistency check onto a secondary SQL Server host, these critical resources remain available for handling application requests, ensuring smoother performance for end users.
- **Verifying the database integrity:** Consistency checks help prevent data corruption by identifying potential issues. If a consistency check is run on a secondary server and an issue is found, a consistency check can then be run on the primary SQL Server host. If no issues are found on the secondary SQL Server host, then no further action is needed on the primary host, conserving resources for application requests.
- **Finding resolutions to data corruption:** If an issue is found on the secondary SQL Server host, a database administrator can experiment with possible solutions to fix the corruption prior to applying the fix to the primary SQL Server host. This provides a better testing platform that avoids potential further damage to the primary host database.

To accomplish this task, a clone of a snapshot must be made, which is then attached to a secondary SQL Server host. These steps can be found in the section on Using Crash-consistent Snapshots to Clone a New SQL Server Instance.

When to Perform Database-consistency Checks on Snapshots

The DBCC CHECKDB command is recommended to be run on a volume snapshot under the following circumstances:

After Restoring the Snapshot to a Test Environment

To ensure the snapshot does not impact the production system, restore the volume snapshot to a test or non-production environment before running DBCC CHECKDB. This allows you to validate the logical and physical integrity of the database without risking disruptions.

As Part of a Regular Integrity Check Process

If you're using volume snapshots for backups or disaster recovery, include DBCC CHECKDB as a scheduled task in your maintenance plan. This ensures periodic validation of database integrity.

Using the DBCC CHECKDB Command

The DBCC CHECKDB command is used in SQL Server to check the logical and physical integrity of all the objects in a specified database. To check the integrity of the entire database:

```
DBCC CHECKDB ([DatabaseName]);
```

Conclusion

To manage data growth and accelerate development, SQL Server databases with FlashArray can benefit from the use of volume snapshots for data protection, copy data management, and managing applications at scale.

Learn more about Pure Storage solutions for SQL Server at purestorage.com/microsoft.

Test drive SQL Server primary storage in the FlashArray virtual lab at
https://www.purestorage.com/solutions/databases-applications/microsoft/sql/sql-test-drive.html