ARCHITECTURAL WHITE PAPER

CLONING SQL SERVER LINUX DATABASES

WITH PURE STORAGE FLASHARRAY

PURESTORAGE®

TABLE OF CONTENTS

INTRODUCTION

The need to move data has increased. From real time analytics to DevOps, the need to clone SQL Server databases has become an essential task for database administrators. Pure Storage® FlashArray™ helps reduce the time required for SQL Server database operations with efficient snapshots.

GOALS AND OBJECTIVES

- Define crash-consistent backup with regard to data consistency.
- Provide example snapshot creation steps using the Pure Storage REST Client for a given test configuration.
- Provide example snapshot deletion steps using the Pure Storage REST Client for a given test configuration.

AUDIENCE

This document is for database, system, and storage administrators responsible for maintaining SQL Server storage environments.

PURE STORAGE INTRODUCTION

FlashArray Product Offerings

The Pure Storage FlashArray family delivers software-defined all-flash power and reliability for every need and every budget, from the entry-level FlashArray//M10 to the new FlashArray//X – the first mainstream, 100% NVMe, enterprise-class all-flash array.

With our Purity Operating Environment software, every FlashArray model enables organizations to achieve the highest levels of business continuity with ActiveCluster while enjoying proven 99.9999% availability, completely non-disruptive operations, and support that's above and beyond.

Whether it's accelerating a single database, powering VMs and desktops, or the foundation of an all-flash cloud, the rich data services and effortless operations of FlashArray will make your enterprise storage something you simply don't worry about anymore.

FIGURE 1. FlashArray//M and FlashArray//X

Solution Summary

PROBLEM

SQL Server database copies are needed to accommodate a variety of consumers. This can be time-consuming using traditional data copy methods.

SOLUTION

FlashArray's architecture is flexible, utilizing snapshot technology with flash-based storage to increase performance and create efficient SQL Server database cloning in enterprise environments.

RESULTS

Using FlashArray snapshot technology with SQL Server results in a fast and efficient method to clone databases.

Purity Protect

Purity Protect combines Purity ActiveCluster with space-saving snapshots, replication, and protection policies into an end-to-end data protection and recovery solution that protects data against loss locally and globally. All Purity Protect services are fully-integrated in FlashArray and leverage native data reduction capabilities.

FIGURE 2. Purity is the software-defined heart of FlashArray

Zero-Compromise Data Services

Consolidating on FlashArray is simple – there's nothing else to buy or install, and all data services are built-in and included. Our industry-leading 5:1 average data reduction (across the entire FlashArray install base) is inline and always-on, which means you'll save on storage, power, cooling, and space. And you can keep that space savings with data reduction-aware snapshots and replication.

Enjoy the protection of always-on encryption and our zero-configuration, Always-On QoS while maintaining consistent mixed workload performance even through component failures and upgrades.

Accelerate

- Latency sensitive applications
- Databases
- Virtual machines and desktops

Pure Storage REST Client

The Pure Storage REST Client provides integration with the Purity Operating Environment and FlashArray. It provides functionalities of Purity's REST API as python requests HTTP library.

SNAPSHOT USE CASES

Development Databases

Database administrators can quickly copy databases with FlashArray snapshots and present data stores to a host for development group usage.

Reporting Databases

Different organization groups often require their own data stores outside of production database environments. FlashArray snapshots can quickly copy databases for these purposes.

Alternate Database Recovery

SQL Server native database backups, and the testing of these backups, are always recommended first for organizations to meet their data restore strategy requirements (amount of acceptable downtime, and data loss). In addition to this recommendation, FlashArray snapshots can also be implemented as an additional means to help mitigate outages when quick access to data is required.

SNAPSHOT ARCHITECTURE

- FlashArray snapshots are crash-consistent snapshots. Data in memory is lost, only the data residing on disk is preserved.
- FlashArray snapshots rely on copying the metadata mapping of data location information managed by Purity Operating Environment. These snapshots are immutable; there are no parent/child relationships.
- Any subsequent writes that occur on the source volume (Volume 1) in the example below are redirected for efficiency.
- A new volume (Volume 2) can be activated with no hierarchical relationship to the snapshot or parent volume, offering flexible deployment.

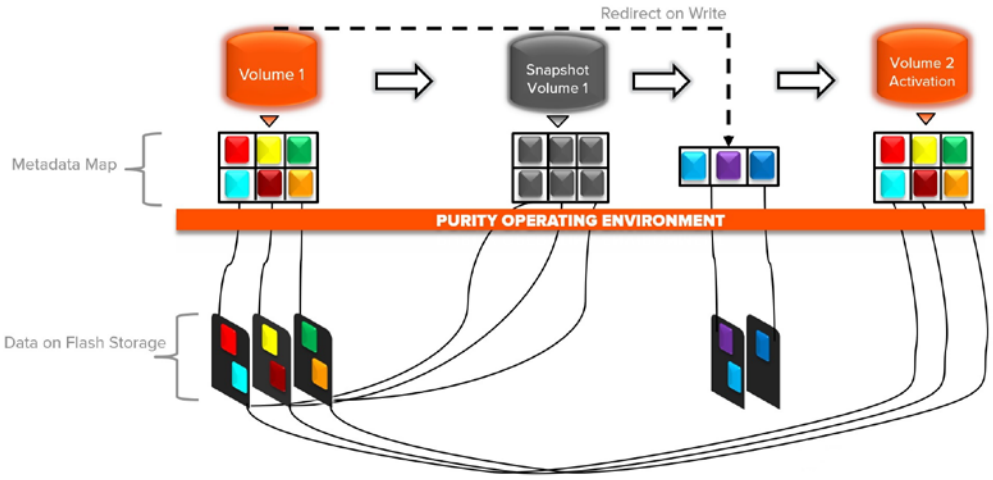


FIGURE 3. New volume activation from existing snapshot

- FlashArray snaphots are immutable. The source volume can be deleted and eradicated without affecting any subsequently created snapshots or activated volumes.
- In the example below, blocks that were associated with the deleted source volume are now marked for reuse.

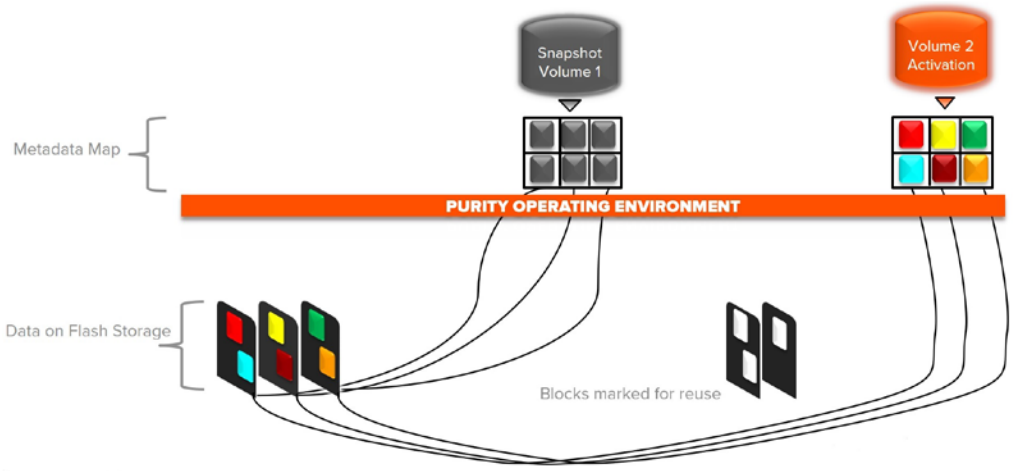


FIGURE 4. Deleting initial volume post snapshot

TEST CONFIGURATION

NOTE: To ensure configuration, performance, and compatibility are optimal, please reference and apply the following Pure Storage and Microsoft Best Practice Guides:

(Pure Storage Documentation) SQL Server Best Practices Guide

(Pure Storage Documentation) Linux Recommended Settings

(Pure Storage Documentation) FlashArray User Guide

(Pure Storage REST Client) Using Python requests HTTP library

OPERATING SYSTEM	INFORMATION/PARAMETER
RED HAT ENTERPRISE LINUX	RELEASE 7.6
DATABASE SOURCE STORAGE	**INFORMATION/PARAMETER**
MODEL	//X70
OPERATING ENVIRONMENT	PURITY 5.1.2
SOFTWARE	**VERSION**
PYTHON	2.7.5
SOFTWARE	**VERSION**
PURE STORAGE REST CLIENT	1.16
RELEASE COMPATIBILITY REQUIREMENTS	• PYTHON 2.6 OR HIGHER. • THIS LIBRARY REQUIRES THE USE OF PYTHON 2.6 OR LATER ANDTHE THIRD-PARTY LIBRARY "REQUESTS". • ADDITIONALLY, THIS LIBRARY CAN ONLY BE USED COMMUNICATE WITH FLASH ARRAYS THAT SUPPORT ONE OR MORE REST API VERSIONS BETWEEN 1.0 AND 1.16; CURRENTLY, THIS INCLUDES ANY FLASH ARRAY RUNNING PURITY 3.4.0 OR LATER.
DATABASE APPLICATION	**INFORMATION/PARAMETER**
MICROSOFT SQL SERVER ENTERPRISE	2017 14.0.3048.4 (X64)
DATABASE SIZE	402GB
DATA SIZE (REALISTIC DATA)	382

TABLE 1. Test configuration details

IMPLEMENTATION STEPS

NOTE: To ensure configuration, performance, and compatibility are optimal, please reference and apply the following Pure Storage and Microsoft Best Practice Guides:

(Pure Storage Documentation) SQL Server Best Practices Guide
(Pure Storage Documentation) Linux Recommended Settings
(Pure Storage Documentation) FlashArray User Guide
(Pure Storage REST Client) Using Python requests HTTP library

Snapshot Creation Example Overview

1. A snapshot (**SQL-LINUX-BENCH-DATA.snap**), and new volume copy (**SQL-LINUX-BENCH-DATA-COPY**) will be created.
2. The new copy volume (SQL-LINUX-BENCH-DATA-COPY) will be connected to a host (**sn1-r720-f06-23**).
3. A new directory will be created (**/var/opt/mssql/sqldatalog_copy**).
4. The volume copy (SQL-LINUX-BENCH-DATA-COPY) wwid that is generated will be referenced.
5. The new RHEL 7.6 wwid based on the volume copy (SQL-LINUX-BENCH-DATA-COPY) will be mounted to the new created directory (/var/opt/mssql/sqldatalog_copy).
6. The SQL Server database clone will be created, named (**tpcc-copy**), and attached to a SQL Server instance.

STEP 1. INSTALL PURE STORAGE REST CLIENT

1. Root privileges are required to check the Python version installed, login with su.

```
[mp@sn1-r720-f06-23 ~]$ su
Password:
```

2. Ensure python modules and the correct version (2.6 or greater) is installed using the following command for verification.

```
[root@sn1-r720-f06-23 mp]# yum list installed python\*
```

```
yum list installed python\*
```

3. If Python is not installed run the following command:

```
[root@sn1-r720-f06-23 mp]# yum install python
```

```
yum install python
```

4. Install the Pure Storage REST Client module per the instructions listed on the site.
   (Pure Storage REST Client) Using Python requests HTTP library

STEP 2. FLASHARRAY CONNECTIVITY

1. Import the Pure Storage REST Client module.
2. Connect to the FlashArray, with api token referenced from the FlashArray.
3. Check FlashArray version.
4. Python commands used.

```
import purestorage
import requests
import sh
import os

#  Disable connection warnings
requests.packages.urllib3.disable_warnings()

#  Establish FlashArray connection
print "Connecting to FlashArray"
array = purestorage.FlashArray("10.21.229.25", api_token="857847a8-1102-0637-4ee4-
8fd062128742")

#  Check FlashArray version.
from purestorage import FlashArray
array = purestorage.FlashArray("10.21.229.25", api_token="857847a8-1102-0637-4ee4-
8fd062128742")
array_info = array.get()
print "FlashArray {} (version {}) REST session established!".format(
        array_info['array_name'], array_info['version'])
```

STEP 3. CREATE SNAPSHOT FROM SOURCE VOLUME

```
#  Create snapshot from source
print "Creating SQL-LINUX-BENCH-DATA snapshot"
array.create_snapshots(["SQL-LINUX-BENCH-DATA"], suffix="snap")
```

STEP 4. CREATE VOLUME COPY FROM SNAPSHOT

```
#  Create volume copy from snapshot
print "Creating SQL-LINUX-BENCH-DATA-COPY volume copy from snapshot"
array.copy_volume(source="SQL-LINUX-BENCH-DATA.snap", dest="SQL-LINUX-BENCH-DATA-COPY")
```

STEP 5. CONNECT VOLUME COPY TO HOST

```
#  Connect volume copy to host
print "Connecting SQL-LINUX-BENCH-DATA-COPY volume copy to host"
array.connect_host("sn1-r720-f06-23", "SQL-LINUX-BENCH-DATA-COPY")
```

STEP 6. RESCAN THE BUS

```
#  Rescan bus
command = os.popen('sudo rescan-scsi-bus.sh -m')
print(command.read())
print(command.close())
```

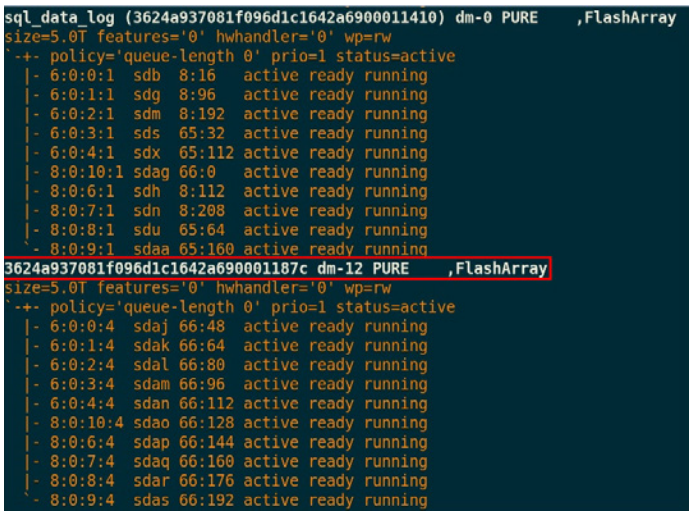
STEP 7. MAKE A NEW DIRECTORY FOR THE VOLUME COPY

```
command = os.popen('mkdir /var/opt/mssql/sqldatalog_copy')
print(command.read())
print(command.close())
```

STEP 8. REFERENCE THE NEW VOLUME COPY WWID

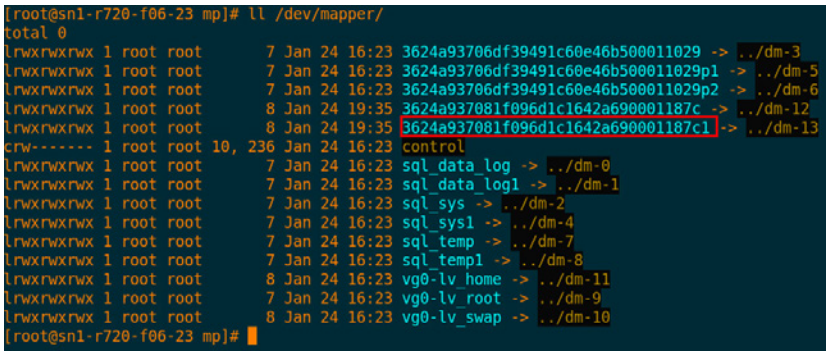
1. Retrieve current wwid of the volume copy using the following commands for reference.
2. **NOTE:** The wwid that is created for the volume copy is not persistent after a reboot: to rename a multipath device, reference the following Red Hat documentation.
   Red Hat Document: How to rename multipath (mpath) device names?
3. Also review the Red Hat documentation for making mount points persistent by editing the **/etc/fstab**.
4. Note the wwid that has not been mounted to a directory.

   ```
   multipath -ll
   ```



5. Note the wwid that ends with a 1 digit.

```
ll /dev/mapper/
```

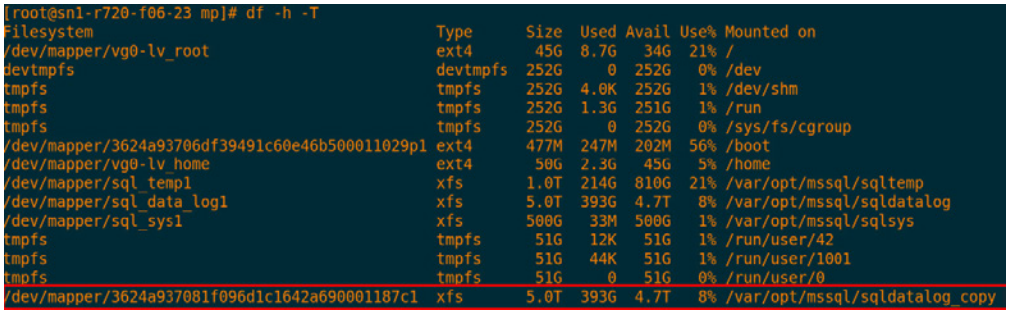


STEP 9. MOUNT THE VOLUME COPY TO THE DIRECTORY

```
#  Mount volume copy without UUID
command = os.popen('mount -o nouuid  /dev/mapper/3624a937081f096d1c1642a690001187c1  /var/opt/
mssql/sqldatalog _ copy')
print(command.read())
print(command.close()).
```

STEP 10. VERIFY AND LIST VOLUMES AND THEIR MOUNT POINTS

```
#  List volumes/mounting
command = os.popen('df -h -T')
print(command.read())
print(command.close()).
```



STEP 11. USING SQL SERVER MANAGEMENT STUDIO (SSMS) ATTACH DATABASE CLONE FILES

With the attach feature, locate the database clone files from the volume copy.

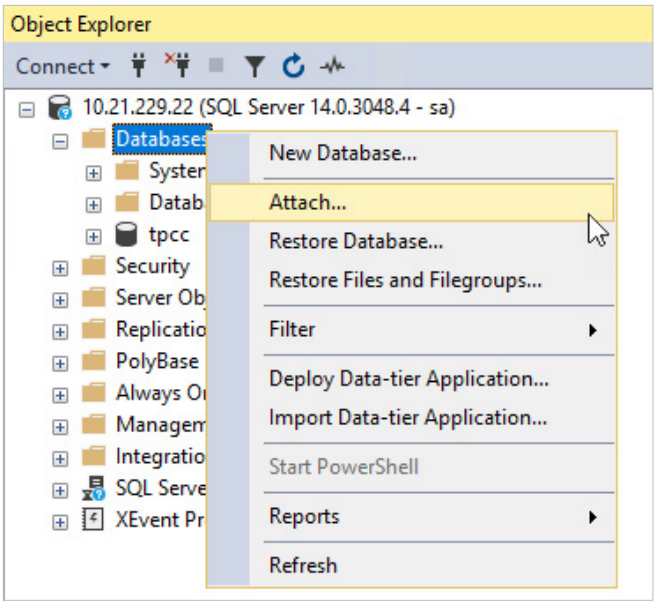


FIGURE 5. Attach feature

Select the corresponding database mdf file copy.

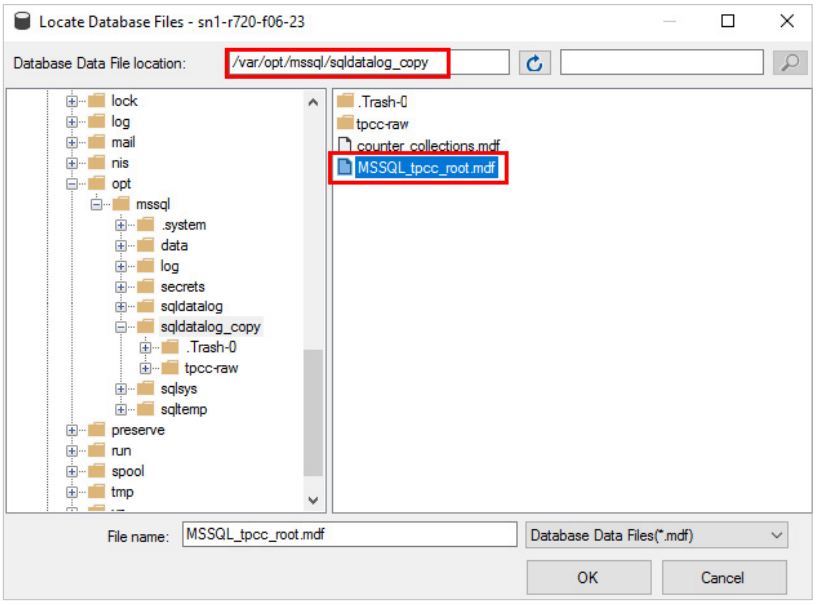


FIGURE 6. Selecting the corresponding mdf file copy

Attach the clone database to the SQL Server instance with a new name.

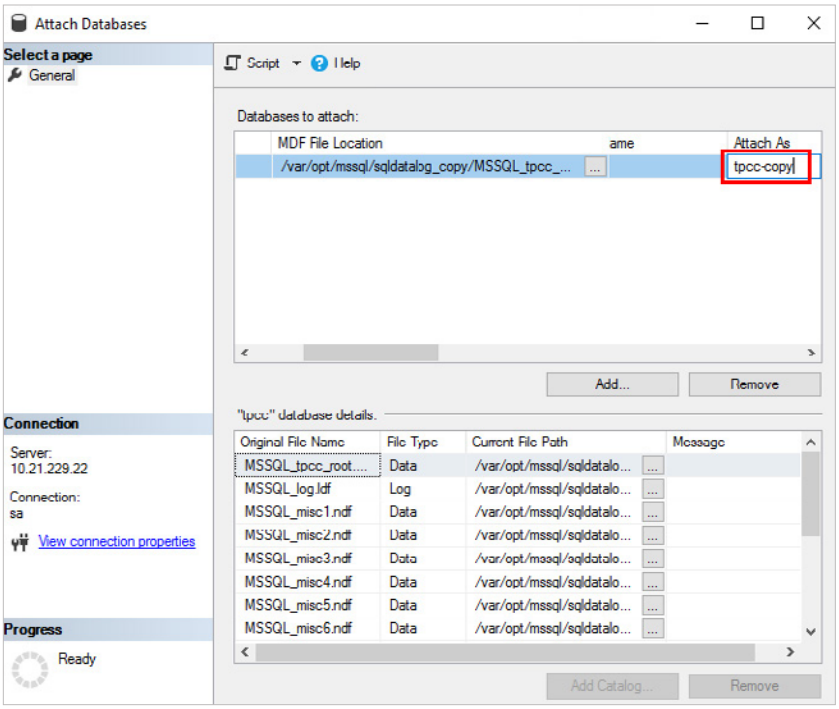


FIGURE 7. Ataching the clone database

SQL Server T-SQL example using the attach feature.

```
USE [master]
GO
CREATE DATABASE [tpcc-copy] ON
( FILENAME = N'/var/opt/mssql/sqldatalog_copy/MSSQL_tpcc_root.mdf'
( FILENAME = N'/var/opt/mssql/sqldatalog_copy/MSSQL_log.ldf' ),
( FILENAME = N'/var/opt/mssql/sqldatalog_copy/MSSQL_misc1.ndf' ),
( FILENAME = N'/var/opt/mssql/sqldatalog_copy/MSSQL_misc2.ndf' ),
( FILENAME = N'/var/opt/mssql/sqldatalog_copy/MSSQL_misc3.ndf' ),
( FILENAME = N'/var/opt/mssql/sqldatalog_copy/MSSQL_misc4.ndf' ),
( FILENAME = N'/var/opt/mssql/sqldatalog_copy/MSSQL_misc5.ndf' ),
( FILENAME = N'/var/opt/mssql/sqldatalog_copy/MSSQL_misc6.ndf' ),
( FILENAME = N'/var/opt/mssql/sqldatalog_copy/MSSQL_misc7.ndf' ),
( FILENAME = N'/var/opt/mssql/sqldatalog_copy/MSSQL_cs1.ndf' ),
( FILENAME = N'/var/opt/mssql/sqldatalog_copy/MSSQL_cs2.ndf' ),
( FILENAME = N'/var/opt/mssql/sqldatalog_copy/MSSQL_cs3.ndf' ),
( FILENAME = N'/var/opt/mssql/sqldatalog_copy/MSSQL_cs4.ndf' ),
( FILENAME = N'/var/opt/mssql/sqldatalog_copy/MSSQL_cs5.ndf' ),
( FILENAME = N'/var/opt/mssql/sqldatalog_copy/MSSQL_cs6.ndf' ),
( FILENAME = N'/var/opt/mssql/sqldatalog_copy/MSSQL_cs7.ndf' )
 FOR ATTACH
GO
```

```
USE [master]
GO
CREATE DATABASE [tpcc-copy] ON
( FILENAME = N'/var/opt/mssql/sqldatalog_copy/MSSQL_tpcc_root.mdf' ),
( FILENAME = N'/var/opt/mssql/sqldatalog_copy/MSSQL_log.ldf' ),
( FILENAME = N'/var/opt/mssql/sqldatalog_copy/MSSQL_misc1.ndf' ),
( FILENAME = N'/var/opt/mssql/sqldatalog_copy/MSSQL_misc2.ndf' ),
( FILENAME = N'/var/opt/mssql/sqldatalog_copy/MSSQL_misc3.ndf' ),
( FILENAME = N'/var/opt/mssql/sqldatalog_copy/MSSQL_misc4.ndf' ),
( FILENAME = N'/var/opt/mssql/sqldatalog_copy/MSSQL_misc5.ndf' ),
( FILENAME = N'/var/opt/mssql/sqldatalog_copy/MSSQL_misc6.ndf' ),
( FILENAME = N'/var/opt/mssql/sqldatalog_copy/MSSQL_misc7.ndf' ),
( FILENAME = N'/var/opt/mssql/sqldatalog_copy/MSSQL_cs1.ndf' ),
( FILENAME = N'/var/opt/mssql/sqldatalog_copy/MSSQL_cs2.ndf' ),
( FILENAME = N'/var/opt/mssql/sqldatalog_copy/MSSQL_cs3.ndf' ),
( FILENAME = N'/var/opt/mssql/sqldatalog_copy/MSSQL_cs4.ndf' ),
( FILENAME = N'/var/opt/mssql/sqldatalog_copy/MSSQL_cs5.ndf' ),
( FILENAME = N'/var/opt/mssql/sqldatalog_copy/MSSQL_cs6.ndf' ),
( FILENAME = N'/var/opt/mssql/sqldatalog_copy/MSSQL_cs7.ndf' )
 FOR ATTACH
GO
```

STEP 12. VERIFY SQL SERVER DATABASE COPY

SQL Server Management Studio example to verify the database copy exists.

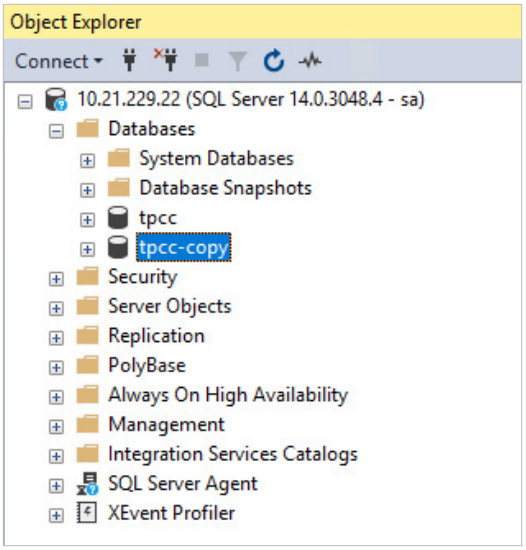


FIGURE 8. Verifying the database copy

Snapshot Deletion Example Overview

1. The SQL Server database clone (tpcc-copy) will be dropped from the instance.
2. The volume copy (SQL-LINUX-BENCH-DATA-COPY) wwid will be un-mounted from the directory (/var/opt/mssql/sqldatalog_copy).
3. The directory (/var/opt/mssql/sqldatalog_copy) will be removed.
4. The RHEL 7.6 volume copy (SQL-LINUX-BENCH-DATA-COPY) will be disconnected from the host.
5. The FlashArray volume ("SQL-LINUX-BENCH-DATA-COPY") will be removed and eradicated from the FlashArray. Eradication is optional.
6. The FlashArray snapshot ("SQL-LINUX-BENCH-DATA.snap") will be removed and eradicated from the FlashArray. Eradication is optional.

STEP 1. DROP SQL SERVER DATABASE CLONE

In SQL Server Management Studio, drop the database clone copy.

```
USE [master]
GO
DROP DATABASE [tpcc-copy]
```

```
USE [master]
GO
DROP DATABASE [tpcc-copy]
```

STEP 2. FLASHARRAY CONNECTIVITY

1. Import the Pure Storage REST Client module.
2. Connect to the FlashArray, with api token referenced from the FlashArray.
3. Check FlashArray version.
4. Python commands used.

```
import purestorage
import requests
import sh
import os

#  Disable connection warnings
requests.packages.urllib3.disable_warnings()

#  Establish FlashArray connection
print "Connecting to FlashArray"
array = purestorage.FlashArray("10.21.229.25", api_token="857847a8-1102-0637-4ee4-
8fd062128742")
```

```
#  Check FlashArray version.
from purestorage import FlashArray
array = purestorage.FlashArray("10.21.229.25", api_token="857847a8-1102-0637-4ee4-
8fd062128742")
array_info = array.get()
print "FlashArray {} (version {}) REST session established!".format(
        array_info['array_name'], array_info['version'])
```

STEP 3. UNMOUNT VOLUME COPY

```
# Unmount the volume copy
command = os.popen('umount /dev/mapper/3624a937081f096d1c1642a690001187c1  /var/opt/mssql/
sqldatalog_copy')
print(command.read())
print(command.close()))
```

STEP 4. REMOVE DIRECTORY

```
# Remove old directory
command = os.popen('rm -r /var/opt/mssql/sqldatalog_copy')
print(command.read())
print(command.close())
```

STEP 5. DISCONNECT VOLUME COPY FROM HOST

```
#  Disconnecting Volume Copy from host
print "Disconnecting SQL-LINUX-BENCH-DATA-COPY volume copy from host"
array.disconnect_host("sn1-r720-f06-23", "SQL-LINUX-BENCH-DATA-COPY")
```

STEP 6. DELETE AND ERADICATE VOLUME AND SNAPSHOTS

```
#  Delete and eradicate all objects
array.destroy_volume("SQL-LINUX-BENCH-DATA.snap")
array.eradicate_volume("SQL-LINUX-BENCH-DATA.snap")
array.destroy_volume("SQL-LINUX-BENCH-DATA-COPY")
array.eradicate_volume("SQL-LINUX-BENCH-DATA-COPY")
print "Objects  SQL-LINUX-BENCH-DATA.snap and SQL-LINUX-BENCH-DATA-COPY are destroyed and
eradicated."
```

STEP 7. LIST VOLUMES AND MOUNTING INFORMATION WITH VOLUME COPY REMOVED

```
#  List volumes/mounting
command = os.popen('df -h -T')
print(command.read())
print(command.close())
```

```
[root@sn1-r720-f06-23 mp]# df -h -T
Filesystem                                      Type      Size  Used Avail Use% Mounted on
/dev/mapper/vg0-lv_root                         ext4       45G  8.7G   34G  21% /
devtmpfs                                        devtmpfs  252G     0  252G   0% /dev
tmpfs                                           tmpfs     252G  4.0K  252G   1% /dev/shm
tmpfs                                           tmpfs     252G  1.4G  251G   1% /run
tmpfs                                           tmpfs     252G     0  252G   0% /sys/fs/cgroup
/dev/mapper/3624a93706df39491c60e46b500011029p1 ext4      477M  247M  202M  56% /boot
/dev/mapper/vg0-lv_home                         ext4       50G  2.3G   45G   5% /home
/dev/mapper/sql_temp1                           xfs       1.0T  214G  810G  21% /var/opt/mssql/sqltemp
/dev/mapper/sql_data_log1                       xfs       5.0T  393G  4.7T   8% /var/opt/mssql/sqldatalog
/dev/mapper/sql_sys1                            xfs       500G   33M  500G   1% /var/opt/mssql/sqlsys
tmpfs                                           tmpfs      51G   12K   51G   1% /run/user/42
tmpfs                                           tmpfs      51G   44K   51G   1% /run/user/1001
tmpfs                                           tmpfs      51G     0   51G   0% /run/user/0
```

> **NOTE:** If SQL Server database files are located across multiple FlashArray volumes a protection group can be created to snapshot multiple FlashArray volumes simultaneously (atomic snapshot). Protection groups can be implemented using the Pure Storage REST Client create_pgroup() method. See Pure Storage REST Client documentation for its usage.

SUMMARY

FlashArray snapshots offer a fast and efficient method of cloning SQL Server databases for a variety of use cases. As data store sizes increase, administrators now have a flexible tool to reduce the complexity of copying large amounts of data and representing it to different consumers.

CODE SAMPLES

Create Snapshots

```
import purestorage
import requests
import sh
import os

#  Disable connection warnings
requests.packages.urllib3.disable_warnings()

#  Establish FlashArray connection
print "Connecting to FlashArray"
array = purestorage.FlashArray("10.21.229.25", api_token="857847a8-1102-0637-4ee4-8fd062128742")

#  Check FlashArray version.
from purestorage import FlashArray
array_info = array.get()
print "FlashArray {} (version {}) REST session established!".format(
        array_info['array_name'], array_info['version'])

#  Create snapshot from source
print "Creating SQL-LINUX-BENCH-DATA snapshot"
array.create_snapshots(["SQL-LINUX-BENCH-DATA"], suffix="snap")

#  Create volume from snapshot
print "Creating SQL-LINUX-BENCH-DATA-COPY volume copy from snapshot"
array.copy_volume(source="SQL-LINUX-BENCH-DATA.snap", dest="SQL-LINUX-BENCH-DATA-COPY")

#  Connect volume copy to host
print "Connecting SQL-LINUX-BENCH-DATA-COPY volume copy to host"
array.connect_host("sn1-r720-f06-23", "SQL-LINUX-BENCH-DATA-COPY")

#  Rescan bus
command = os.popen('sudo rescan-scsi-bus.sh -m')
print(command.read())
print(command.close())

# Make new directory for volume copy
command = os.popen('mkdir /var/opt/mssql/sqldatalog_copy')
print(command.read())
```

```
print(command.close())

# Make note of the new volume copy ID wwid if host has been rebooted and input below.
# multipath -ll
# ll /dev/mapper/

#  Mount volume copy without UUID
command = os.popen('mount -o nouuid  /dev/mapper/3624a937081f096d1c1642a690001187c1  /var/opt/
mssql/sqldatalog_copy')
print(command.read())
print(command.close())

#  List volumes/mounting
command = os.popen('df -h -T')
print(command.read())
print(command.close())
```

Attach Database

```
USE [master]
GO
CREATE DATABASE [tpcc-copy] ON
( FILENAME = N'/var/opt/mssql/sqldatalog_copy/MSSQL_tpcc_root.mdf' ),
( FILENAME = N'/var/opt/mssql/sqldatalog_copy/MSSQL_log.ldf' ),
( FILENAME = N'/var/opt/mssql/sqldatalog_copy/MSSQL_misc1.ndf' ),
( FILENAME = N'/var/opt/mssql/sqldatalog_copy/MSSQL_misc2.ndf' ),
( FILENAME = N'/var/opt/mssql/sqldatalog_copy/MSSQL_misc3.ndf' ),
( FILENAME = N'/var/opt/mssql/sqldatalog_copy/MSSQL_misc4.ndf' ),
( FILENAME = N'/var/opt/mssql/sqldatalog_copy/MSSQL_misc5.ndf' ),
( FILENAME = N'/var/opt/mssql/sqldatalog_copy/MSSQL_misc6.ndf' ),
( FILENAME = N'/var/opt/mssql/sqldatalog_copy/MSSQL_misc7.ndf' ),
( FILENAME = N'/var/opt/mssql/sqldatalog_copy/MSSQL_cs1.ndf' ),
( FILENAME = N'/var/opt/mssql/sqldatalog_copy/MSSQL_cs2.ndf' ),
( FILENAME = N'/var/opt/mssql/sqldatalog_copy/MSSQL_cs3.ndf' ),
( FILENAME = N'/var/opt/mssql/sqldatalog_copy/MSSQL_cs4.ndf' ),
( FILENAME = N'/var/opt/mssql/sqldatalog_copy/MSSQL_cs5.ndf' ),
( FILENAME = N'/var/opt/mssql/sqldatalog_copy/MSSQL_cs6.ndf' ),
( FILENAME = N'/var/opt/mssql/sqldatalog_copy/MSSQL_cs7.ndf' )
FOR ATTACH
GO
```

Drop Database

```
USE [master]
GO
DROP DATABASE [tpcc-copy]
```

Delete Snapshot

```
import purestorage
import requests
import sh
import os

#  Disable connection warnings
requests.packages.urllib3.disable_warnings()

# Unmount the volume copy
command = os.popen('umount /dev/mapper/3624a937081f096d1c1642a690001187c1  /var/opt/mssql/
sqldatalog_copy')
print(command.read())
print(command.close())

# Remove old directory
command = os.popen('rm -r /var/opt/mssql/sqldatalog_copy')
print(command.read())
print(command.close())

print "Connecting to FlashArray"
array = purestorage.FlashArray("10.21.229.25", api_token="857847a8-1102-0637-4ee4-8fd062128742")

#  Check FlashArray version.
from purestorage import FlashArray
array_info = array.get()
print "FlashArray {} (version {}) REST session established!".format(
        array_info['array_name'], array_info['version'])

#  Cleanup
print "Cleanup of SQL-LINUX-BENCH-DATA.snap and SQL-LINUX-BENCH-DATA-COPY "

#  Disconnecting Volume Copy from host
print "Disconnecting SQL-LINUX-BENCH-DATA-COPY volume copy from host"
```

```
array.disconnect_host("sn1-r720-f06-23", "SQL-LINUX-BENCH-DATA-COPY")

#  Delete and eradicate all objects
array.destroy_volume("SQL-LINUX-BENCH-DATA.snap")
array.eradicate_volume("SQL-LINUX-BENCH-DATA.snap")
array.destroy_volume("SQL-LINUX-BENCH-DATA-COPY")
array.eradicate_volume("SQL-LINUX-BENCH-DATA-COPY")
print "Objects  SQL-LINUX-BENCH-DATA.snap and SQL-LINUX-BENCH-DATA-COPY are destroyed and
eradicated."

#  List volumes/mounting
command = os.popen('df -h -T')
print(command.read())
print(command.close())
```


ps_wp22p_cloning-sql-server-databases-with-flasharray_ltr_01