Technical Report

NetApp Data Fabric for Enterprise MongoDB

Solution Guide

Ranga Sankar, Aaron Carter, Jayakumar Chendamari, and David Bellizzi, NetApp
January 2018 | TR-4660

Abstract

This document details the NetApp® Data Fabric benefits for enterprise MongoDB installation on NetApp AFF. The benefits include:

- Database protection, backup, and rapid recovery using NetApp SnapCenter®
- Superior data protection with unified replication across the data fabric endpoints such as hybrid array and cloud
- Resilient and fault-tolerant Data Fabric enabling nondisruptive upgrades
- Secure tenant separation, enabling strict SLAs with secure multitenancy and QoS

TABLE OF CONTENTS

NetApp Data Fabric for Enterprise MongoDB Solution Guide 1

1 Introduction 4

2 Solution Overview 4

2.1 Use Case Summary 5

3 Configuration Details 5

4 Customer Use Case 1: MongoDB Management Using SnapCenter 6

4.1 Installation of MongoDB and SnapCenter on AFF8080 6

5 Customer Use Case 2: Database Protection and Rapid Recovery 7

5.1 Creating Backups Using SnapCenter 8

5.2 Restoring Backups Using SnapCenter 8

5.3 Creating Backups Using SnapCenter with Storage Takeover/Giveback 9

6 Customer Use Case 3: Unified Data Replication to Secondary Storage and Cloud 10

6.1 Creating Backups to Secondary Storage 10

7 Customer Use Case 4: Resilient Fault-Tolerant Data Fabric Enabling Simple, Nondisruptive Upgrades 13

7.1 Performing a Nondisruptive Upgrade 13

8 Customer Use Case 5: Secure Tenant Separation 14

8.1 Creating MongoDB Tenants and Setting QoS for Tenant Management 15

8.2 Enabling SLA Management in Data Fabric 18

8.3 Multitenancy and ONTAP Resiliency 20

9 Customer Use Case 6: Reduce TCO with FlexClone and Variable Compression Groups 22

9.1 FlexClone and MongoDB Space Savings 23

9.2 Clone from a Backup Copy 24

Conclusion 25

Where to Find Additional Information 25

Version History 26

LIST OF TABLES

Table 1) Requirements for test configuration 5

LIST OF FIGURES

Figure 1) NetApp Data Fabric components. 5
Figure 2) Supported applications and protocols. 6
Figure 3) Test configuration....15
Figure 4) Space saving with FlexClone volumes. 23

1 Introduction

MongoDB is an open-source NoSQL database used by companies of all sizes across all industries, and for a wide variety of applications. These applications include business-critical operational applications for which low latency, high throughput, and continuous availability are crucial.

MongoDB is an agile database that uses a flexible document data model so that schemas can change quickly as applications evolve.

This database combines the functionality that developers expect from traditional databases, such as secondary indexes, an expressive query language, and strong consistency, combined with the performance and agility of a NoSQL database.

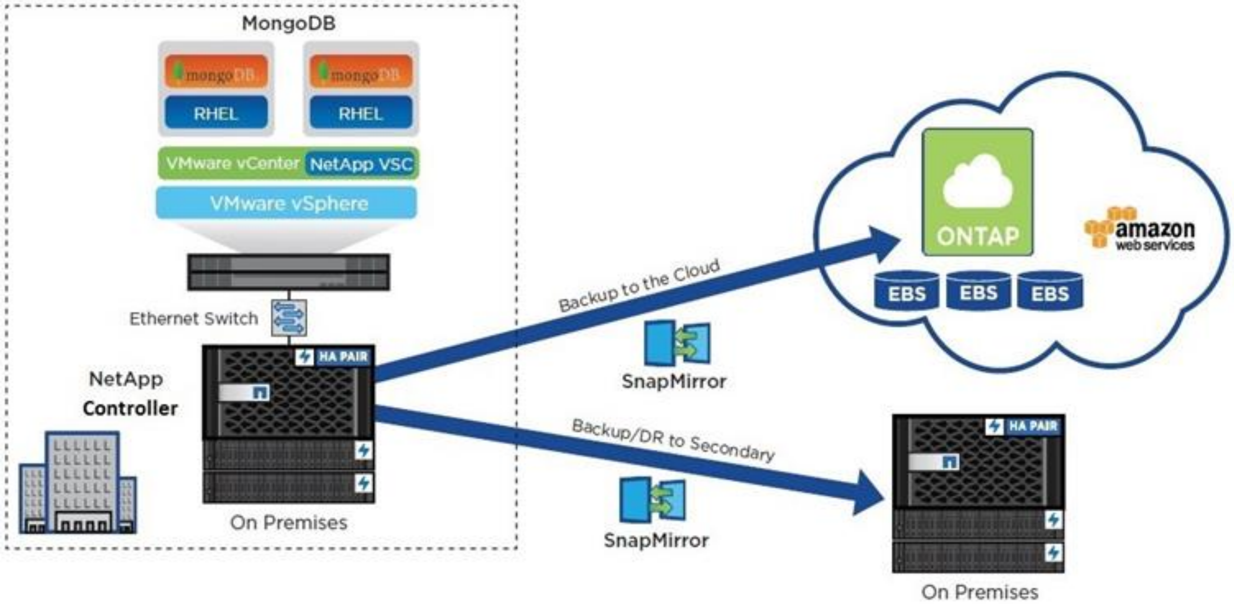
MongoDB is built for scalability, performance, and high availability, scaling from single-server deployments to large, complex, multisite architectures. By leveraging in-memory computing, MongoDB provides high performance for both reads and writes. MongoDB's native replication and automated failover enable enterprise-grade reliability and operational flexibility. By deploying MongoDB on NetApp technology, you get consistent microsecond response, backup, and recovery from the cloud, helping you maintain the highest levels of performance and uptime. With the advanced fault recovery features and easy in-service growth capabilities of NetApp solutions, you can meet ever-changing business requirements.

This document highlights customer use cases and workflows that leverage NetApp Data Fabric features for the management of MongoDB databases.

2 Solution Overview

In the end-to-end solution in Figure 1, virtualized MongoDB is hosted on NetApp AFF8080 storage and VMware vSphere ESXi-6.0. The NetApp AFF array provides low latency, inline deduplication, and compression to deliver high performance. NetApp SnapCenter enables consistent backup for the application and the corresponding database. It also enables users to create zero-cost clones of the entire MongoDB ecosystem. SnapCenter provides the capability to manage remote replication of the entire MongoDB environment to ONTAP® Cloud instances that run in the cloud or to hybrid arrays. The data that is replicated to the hybrid array or to the cloud can be used for disaster recovery. Refer to the technical report MongoDB on the NetApp Data Fabric for more details about the reference architecture of MongoDB on AFF. Figure 1 shows the technical components of the NetApp Data Fabric solution for MongoDB.

Figure 1) NetApp Data Fabric components.



2.1 Use Case Summary

This solution applies to the following use cases:

- Simple, scalable MongoDB management using NetApp SnapCenter
- Database protection and rapid recovery
- Unified data replication to secondary storage and to the cloud
- Resilient fault-tolerant Data Fabric enabling simple, nondisruptive upgrades
- Secure tenant separation, enabling strict SLAs with SMT and QoS
- Reduced total cost of ownership (TCO) by increasing space savings with NetApp FlexClone® and variable compression groups

3 Configuration Details

This section covers the technology components used in the test configuration (see Table 1). The components that are used in any particular implementation of the solution might vary based on your requirements.

Table 1) Requirements for test configuration.

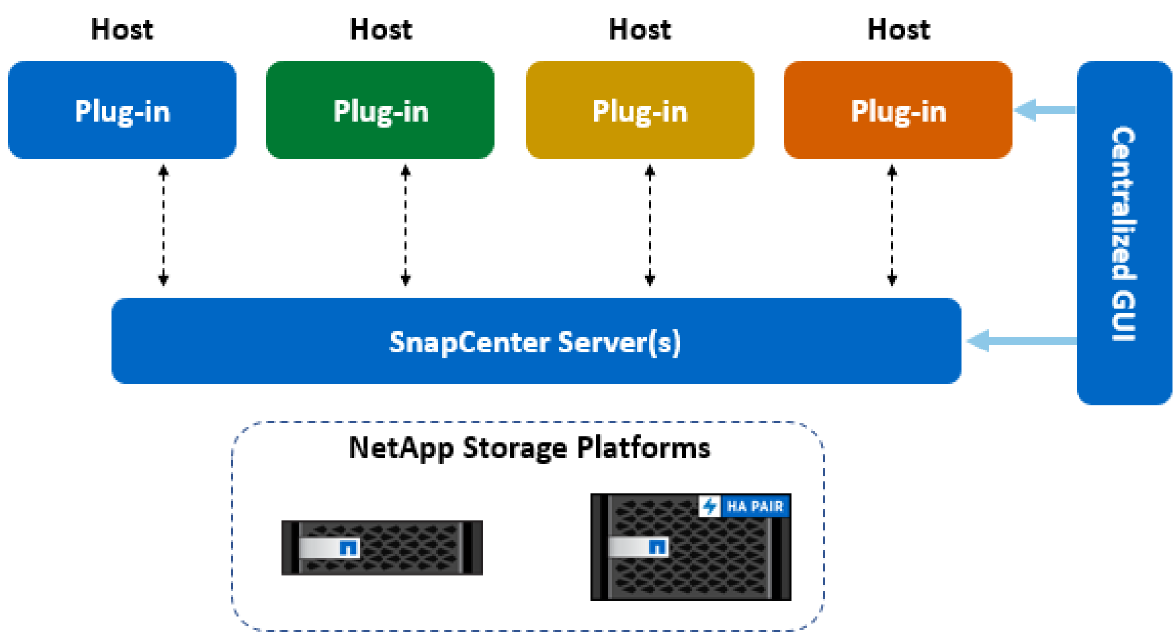
Component	Version
SnapCenter	3.0
Windows (SnapCenter)	Windows Server 2012 R2
ESX/VM hosts	VMware ESXi 6.0/RHEL 7.2
MongoDB	3.4.6 (2 shards of 3 VMs, 3 config servers, 3 app servers)
PyMongo	3.4.5

Component	Version
MongoDB plug-in	1.0
NetApp ONTAP version	9.3x4
Controller model	AFF8080
Processors type/cache	Intel/20/130GB

4 Customer Use Case 1: MongoDB Management Using SnapCenter

SnapCenter is a unified, scalable platform for application-consistent data protection. SnapCenter provides centralized control and oversight, while delegating the ability for users to manage application-specific backup, restore, and clone jobs. With SnapCenter, database and storage administrators learn a single tool to manage backup, restore, and cloning operations for a variety of applications and databases. SnapCenter manages data across endpoints in the NetApp Data Fabric. You can use SnapCenter to replicate data between on-premises environments; between on-premises environments and the cloud; and between private, hybrid, or public clouds. SnapCenter supports application-specific plug-ins to manage various operations. The MongoDB plug-in is bundled with SnapCenter to manage the MongoDB application. SnapCenter supports a variety of applications and protocols. Consult the NetApp Interoperability Matrix Tool (IMT) for details.

Figure 2) Supported applications and protocols.



4.1 Installation of MongoDB and SnapCenter on AFF8080

The following steps provide an overview of the installation and configuration of MongoDB and SnapCenter on AFF8080.

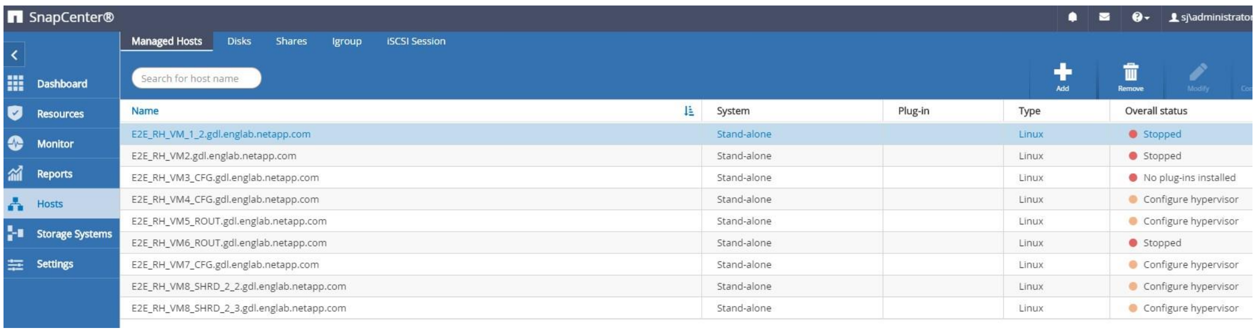
1. Install RHEL 7.2 on the 12 hosts or virtual machines (VMs).
2. Install the PyMongo 3.4.5 and MongoDB 3.4.6 packages on the hosts or VMs.

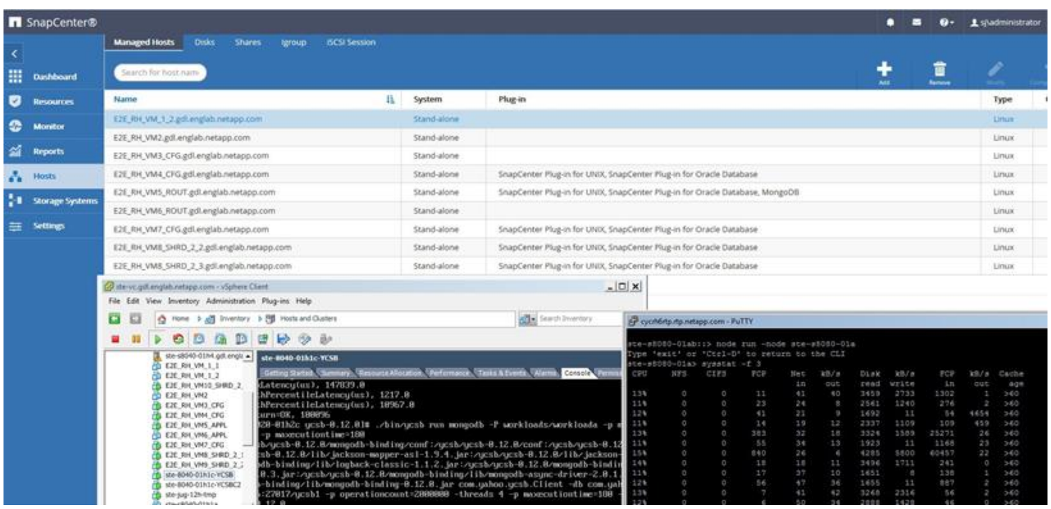
3. Refer to the configuration section guidelines to set up the hosts and controller.
4. Start the `mongods` (shards and config servers) and `mongos` servers.
5. Install SnapCenter in a host or VM.
6. Install and configure the MongoDB plug-in from SnapCenter.

Test Results

To verify that the hosts and controllers are listed in SnapCenter, complete the following steps:

1. Verify successful installation of all components.
2. From the SnapCenter dashboard, add the MongoDB hosts and the controllers.
3. Verify that the hosts and the controllers are listed.





5 Customer Use Case 2: Database Protection and Rapid Recovery

SnapCenter provides a simple, centralized, scalable, end-to-end data protection and in-place copy data management solution for enterprise data on the premises or in a hybrid cloud. SnapCenter leverages NetApp Snapshot™, SnapRestore®, FlexClone, SnapMirror®, and SnapVault® technologies to provide fast, space-efficient, application-consistent, disk-based backups; rapid, granular restore and application-consistent recovery; and quick, space-efficient cloning.

SnapCenter provides fast, efficient replication from any ONTAP primary storage to secondary NetApp storage, including:

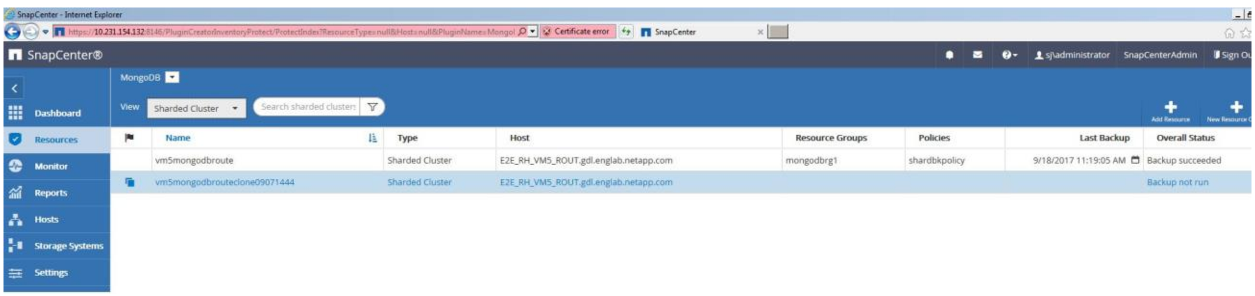
- AFF A-Series all-flash storage
- FAS hybrid flash storage
- NetApp AltaVault™ cloud-integrated storage (physical/virtual)
- NetApp private storage for cloud collocated storage
- NetApp ONTAP Select software-defined storage
- NetApp Cloud ONTAP in a public cloud

5.1 Creating Backups Using SnapCenter

1. Using the `ycsb` tool, create a MongoDB database of about 50GB in size.
2. Using `ycsb`, create a workload that is a mix of 50/50 reads and writes (workload A). Allow this traffic to run for about 120 seconds.
3. Using `sysstat`, verify the progress of the I/O workload.
4. Using the SnapCenter GUI, create a backup to a local volume.

Test Results

1. Verify that the point-in-time backup is created without any failures.

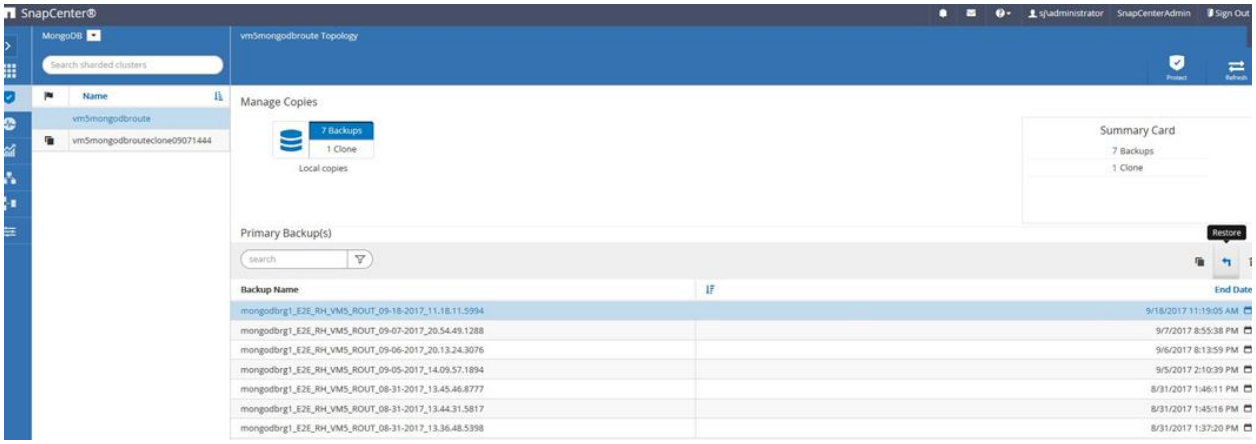


5.2 Restoring Backups Using SnapCenter

1. Using the `ycsb` tool, create a MongoDB database of about 50GB in size.
2. Using the `ycsb` tool, create a workload that is a mix of 50/50 reads and writes (workload A). Allow this traffic to run for about 120 seconds.
3. Using `sysstat`, verify the progress of the I/O workload.
4. Using the SnapCenter GUI, create a backup to a local volume.
5. Using the SnapCenter GUI, perform a restore from the point-in-time backup.
6. Repeat steps 1 through 5 three times and verify that I/O restarts, backup completes, and `ycsb` does not display errors.

Test Result

1. From the SnapCenter GUI, list and view the backups.

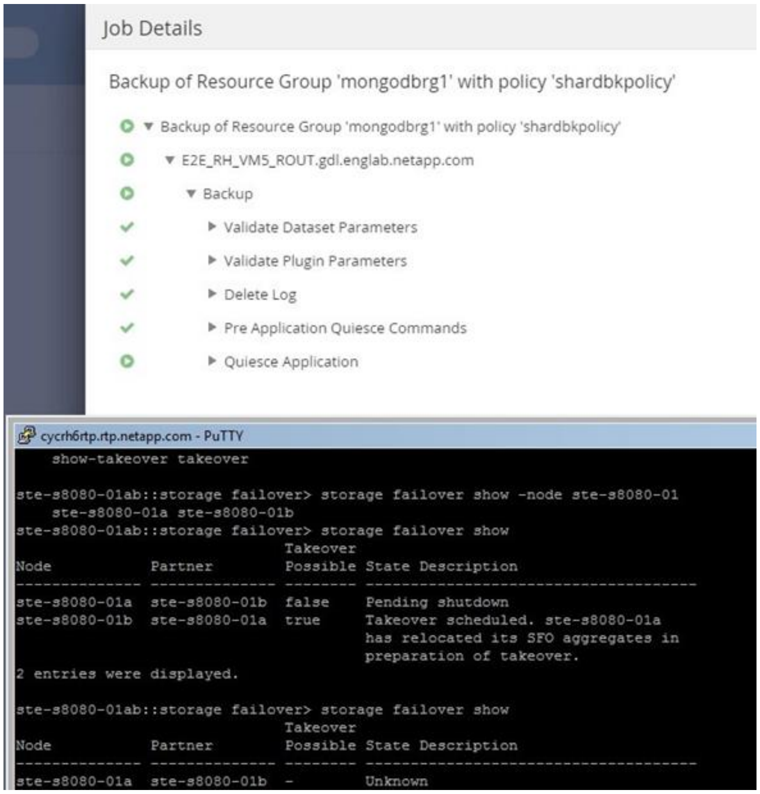


5.3 Creating Backups Using SnapCenter with Storage Takeover/Giveback

1. Using `ycsb`, create a MongoDB database of about 50GB in size.
2. Using `ycsb`, create a workload that is a mix of 50/50 reads and writes (workload A). Allow this traffic to run for about 120 seconds.
3. Using `sysstat`, verify the progress of the I/O workload.
4. Using the SnapCenter GUI, create a backup to a local volume.
5. Perform a storage takeover while the backup is occurring.
6. After about 30 seconds, verify that the backup completes successfully and I/O restarts without any issues.
7. Wait five minutes for the I/O to stabilize.
8. Using the SnapCenter GUI, create a backup to a local volume.
9. Perform a giveback and verify successful completion.

Test Result

1. Verify the MongoDB traffic continues without any failure while the storage takeover/giveback operation completes successfully. The following screenshot displays the backup operation with the storage takeover running in the background.



6 Customer Use Case 3: Unified Data Replication to Secondary Storage and Cloud

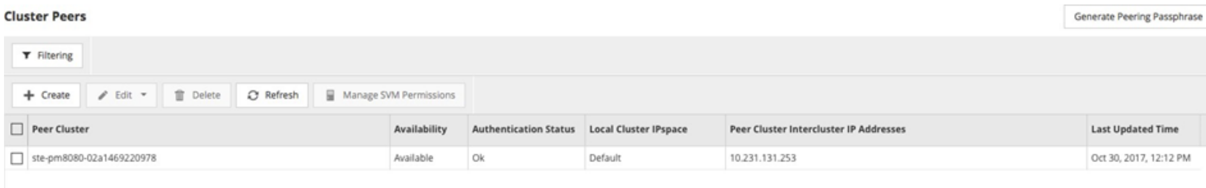
SnapCenter simplifies data protection management across the Data Fabric with cross-platform replication from flash to disk to cloud. A Data Fabric enables transporting data seamlessly between NetApp storage systems, enabling efficient backup and disaster recovery.

6.1 Creating Backups to Secondary Storage

This section describes the backup and restore using SnapCenter to a mirrored volume on the secondary storage on the hybrid array created by using SnapMirror.

Backing Up to a SnapMirror Volume

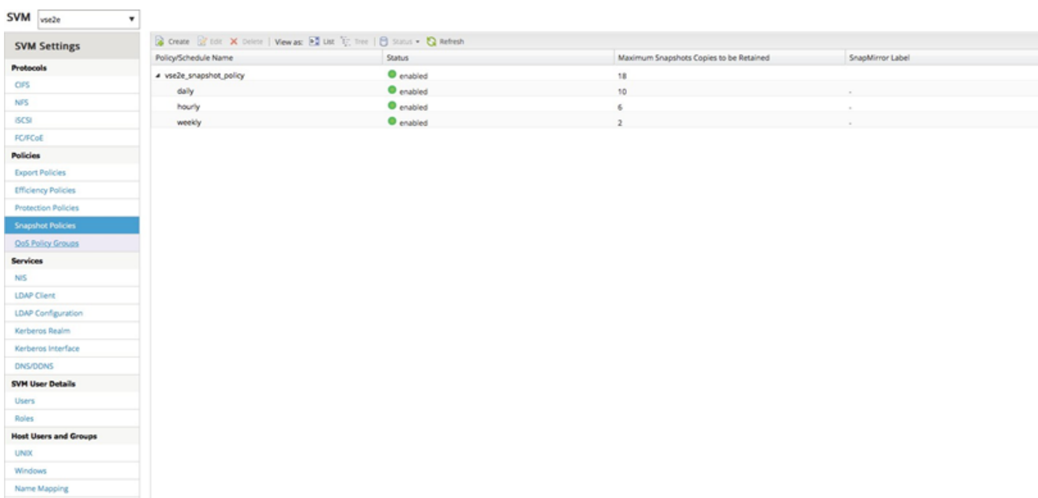
1. Using `ycsb`, create a MongoDB database about 50GB in size.
2. Using `ycsb`, create a workload that is a mix of 50/50 reads and writes (workload A). Allow this traffic to run for about 120 seconds.
3. Using `sysstat`, verify the progress of the I/O workload.
4. Create a SnapMirror relationship:
    a. Create a cluster peer between the source and destination using System Manager. Cluster peering is important for establishing the SnapMirror relationship between two clusters.




```
ste-pm8080-02a1469220978::*> cluster peer show
Peer Cluster Name         Cluster Serial Number Availability   Authentication
------------------------- --------------------- -------------- --------------
ste-s8080-01ab            1-80-000011           Available      ok
```

b. Configure the Snapshot policy on the source. These Snapshot copies are transferred using SnapMirror between the two clusters. It is important to set a schedule that is appropriate to your MongoDB database. If your database does a lot of write transactions, you might need an aggressive schedule to make sure that, if you must perform a restore, the Snapshot copy contains the most recent data.

Note: More Snapshot copies take up more space; therefore, it is important to size the volumes appropriately.




```
ste-s8080-01ab::volume snapshot policy> create -vserver vse2e -policy vse2e_snapshot_policy -enabled true -schedule1 daily -count1 10 -schedule2 hourly -count2 6 -schedule3 weekly -count3 2

ste-s8080-01ab::volume snapshot policy> show -vserver vse2e
Vserver: vse2e
                         Number of Is
Policy Name              Schedules Enabled Comment
------------------------ --------- ------- ----------------------------------
vse2e_snapshot_policy            3 true    -
    Schedule               Count     Prefix                 SnapMirror Label
    ---------------------- -----     ---------------------- -------------------
    daily                     10     daily                  -
    hourly                     6     hourly                 -
    weekly                     2     weekly                 -
```

c. Create the SnapMirror relationships between destination and source. The destination is on the remote cluster as backup for the source volumes.

Relationships

Create | Edit | Delete | Operations | Refresh

Source Storage Virtu...	Source Volume	Destination Volume	Destination Storage ...	Is Healthy	Relationship State	Transfer Status	Relationship Type	Lag Time	Policy Name	Policy Type
vse2e	mdbcfg_p_01	mdbcfg_p_01_mirror	e2e_snapmirror_dest	Yes	Snapmirrored	Idle	Version-Flexible Mirror	30 min(s)	test_policy_1	Asynchronous Mirror
vse2e	mdbcfg_p_02	mdbcfg_p_02_mirror	e2e_snapmirror_dest	Yes	Snapmirrored	Idle	Version-Flexible Mirror	30 min(s)	test_policy_1	Asynchronous Mirror
vse2e	mdbdata_p_01	mdbdata_p_01_mirror	e2e_snapmirror_dest	Yes	Snapmirrored	Idle	Version-Flexible Mirror	30 min(s)	test_policy_1	Asynchronous Mirror
vse2e	mdbdata_p_011003172...	mdbdata_p_011003172...	e2e_snapmirror_dest	Yes	Snapmirrored	Idle	Version-Flexible Mirror	30 min(s)	test_policy_1	Asynchronous Mirror
vse2e	mdbdata_p_02	mdbdata_p_02_mirror	e2e_snapmirror_dest	Yes	Snapmirrored	Idle	Version-Flexible Mirror	30 min(s)	test_policy_1	Asynchronous Mirror
vse2e	mdbdata_p_021003172...	mdbdata_p_021003172...	e2e_snapmirror_dest	Yes	Snapmirrored	Idle	Version-Flexible Mirror	30 min(s)	test_policy_1	Asynchronous Mirror
vse2e	mdbdata_s_01	mdbdata_s_01_mirror	e2e_snapmirror_dest	Yes	Snapmirrored	Idle	Version-Flexible Mirror	30 min(s)	test_policy_1	Asynchronous Mirror
vse2e	mdbdata_s_02	mdbdata_s_02_mirror	e2e_snapmirror_dest	Yes	Snapmirrored	Idle	Version-Flexible Mirror	30 min(s)	test_policy_1	Asynchronous Mirror
vse2e	mdbdata_s_03	mdbdata_s_03_mirror	e2e_snapmirror_dest	Yes	Snapmirrored	Idle	Version-Flexible Mirror	30 min(s)	test_policy_1	Asynchronous Mirror

```
ste-pm8080-02a1469220978::snapmirror*> show -vserver e2e_snapmirror_dest
                                                                       Progress
Source            Destination Mirror  Relationship   Total             Last
Path        Type  Path        State   Status         Progress  Healthy Updated
----------- ---- ------------ ------- -------------- --------- ------- --------
vse2e:mdbcfg_p_01
            XDP  e2e_snapmirror_dest:mdbcfg_p_01_mirror
                              Snapmirrored
                                      Idle           -         true    -
vse2e:mdbcfg_p_02
            XDP  e2e_snapmirror_dest:mdbcfg_p_02_mirror
                              Snapmirrored
                                      Idle           -         true    -
vse2e:mdbdata_p_01
            XDP  e2e_snapmirror_dest:mdbdata_p_01_mirror
                              Snapmirrored
                                      Idle           -         true    -
vse2e:mdbdata_p_011003172251445690
            XDP  e2e_snapmirror_dest:mdbdata_p_011003172251445690_mirror
                              Snapmirrored
                                      Idle           -         true    -
vse2e:mdbdata_p_02
            XDP  e2e_snapmirror_dest:mdbdata_p_02_mirror
                              Snapmirrored
                                      Idle           -         true    -

                                                                       Progress
Source            Destination Mirror  Relationship   Total             Last
Path        Type  Path        State   Status         Progress  Healthy Updated
----------- ---- ------------ ------- -------------- --------- ------- --------
vse2e:mdbdata_p_021003172251459901
            XDP  e2e_snapmirror_dest:mdbdata_p_021003172251459901_mirror
                              Snapmirrored
                                      Idle           -         true    -
vse2e:mdbdata_s_01
            XDP  e2e_snapmirror_dest:mdbdata_s_01_mirror
                              Snapmirrored
                                      Idle           -         true    -
vse2e:mdbdata_s_02
            XDP  e2e_snapmirror_dest:mdbdata_s_02_mirror
                              Snapmirrored
                                      Idle           -         true    -
vse2e:mdbdata_s_03
            XDP  e2e_snapmirror_dest:mdbdata_s_03_mirror
                              Snapmirrored
                                      Idle           -         true    -
9 entries were displayed.
```

d. Create the SnapMirror policy. The SnapMirror policy determines the number of Snapshot copies to be transferred using SnapMirror.

Snapmirror-Label	Retention Count	Preserve older Snapshot copies if retention count is exceeded
sm_created	1	No
all_source_snapshots	1	No

```
ste-pm8080-02a1469220978::snapmirror*> modify -vserver e2e_snapmirror_dest -policy test_policy_1 -destination-path e2e_snapmirror_dest:* -source-path vse2e:*
Operation succeeded: snapmirror modify for the relationship with destination "e2e_snapmirror_dest:mdbcfg_p_01_mirror".
Operation succeeded: snapmirror modify for the relationship with destination "e2e_snapmirror_dest:mdbcfg_p_02_mirror".
Operation succeeded: snapmirror modify for the relationship with destination "e2e_snapmirror_dest:mdbdata_p_01_mirror".
Operation succeeded: snapmirror modify for the relationship with destination "e2e_snapmirror_dest:mdbdata_p_011003172251445690_mirror".
Operation succeeded: snapmirror modify for the relationship with destination "e2e_snapmirror_dest:mdbdata_p_02_mirror".
Operation succeeded: snapmirror modify for the relationship with destination "e2e_snapmirror_dest:mdbdata_p_021003172251459901_mirror".
Operation succeeded: snapmirror modify for the relationship with destination "e2e_snapmirror_dest:mdbdata_s_01_mirror".
Operation succeeded: snapmirror modify for the relationship with destination "e2e_snapmirror_dest:mdbdata_s_02_mirror".
Operation succeeded: snapmirror modify for the relationship with destination "e2e_snapmirror_dest:mdbdata_s_03_mirror".
9 entries were acted on.
```

e. Add the SnapMirror rule. This rule is part of the SnapMirror policy and determines the type of mirror. In the following example, asynchronous SnapMirror mode was selected to enable frequent backups.

Snapmirror-Label	Retention Count	Preserve older Snapshot copies if retention count is exceeded
sm_created	1	No
all_source_snapshots	1	No

```
ste-pm8080-02a1469220978::snapmirror policy*> add-rule -vserver e2e_snapmirror_dest -policy test_policy_1 -snapmirror-label all_source_snapshots -keep 1

ste-pm8080-02a1469220978::snapmirror policy*> show
Vserver Policy             Policy Number         Transfer
Name    Name               Type   Of Rules Tries Priority Comment
------- ------------------ ------ -------- ----- -------- ----------
e2e_snapmirror_dest
        test_policy_1      async-mirror  2     8  normal  -
  SnapMirror Label: sm_created                     Keep:       1
                    all_source_snapshots                       1
                                             Total Keep:       2
```

f. Create a backup to a SnapMirror volume on a hybrid array from the destination cluster.

Restoring from a SnapMirror Volume

1. Wait for about five minutes and then perform a restore from the SnapMirror volume.
2. Select the Snapshot copy that contains the data that you want to restore.
3. Verify that the point-in-time restore from a SnapMirror volume occurs without failure.

7 Customer Use Case 4: Resilient Fault-Tolerant Data Fabric Enabling Simple, Nondisruptive Upgrades

A nondisruptive upgrade (NDU) is an update to software or hardware that does not interrupt access to data or system services. Because new features are introduced in each release of ONTAP, it is important to understand these features and their upgrade requirements, if any. NetApp recommends using the Upgrade Advisor for planning platform and ONTAP upgrades. The NetApp Data Fabric provides the ability to nondisruptively upgrade the controller nodes to a newer version in a cluster. This ability is critical for big data, database, and other applications because there is no downtime involved with the upgrades.

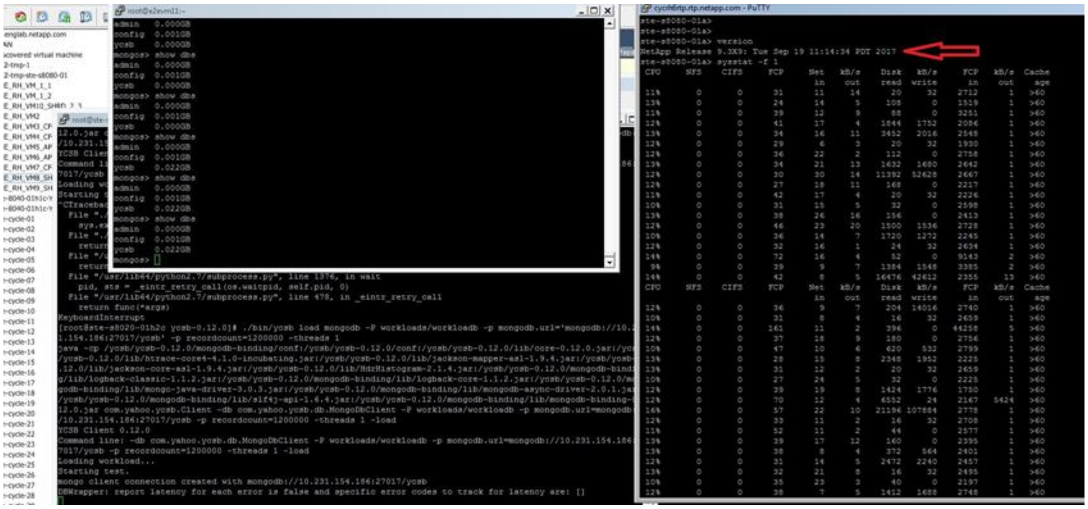
7.1 Performing a Nondisruptive Upgrade

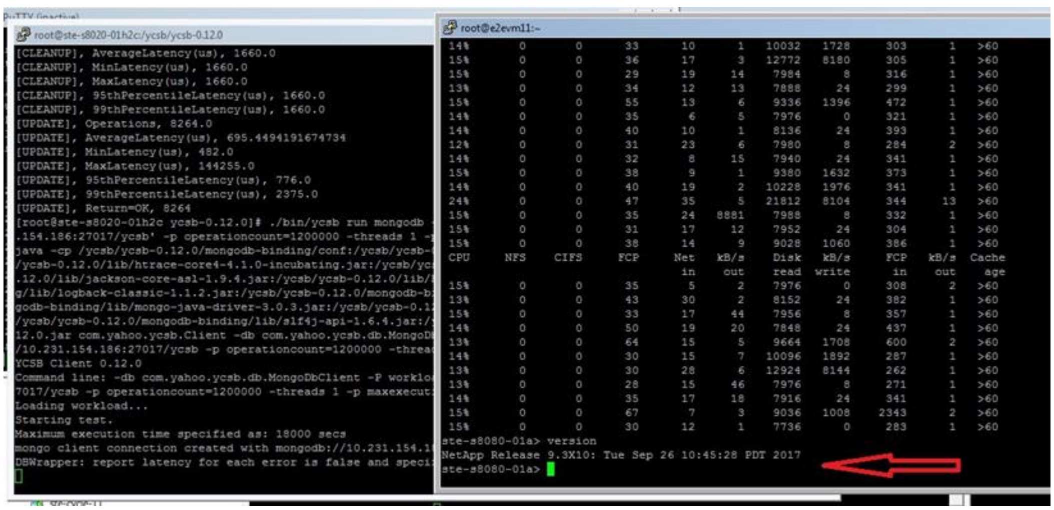
The following steps describe how to perform a nondisruptive upgrade and verify that the application I/Os in progress are not affected by the upgrade process:

1. Using the `ycsb` tool, create a MongoDB database about 50GB in size.
2. Using `ycsb`, create a workload that is a mix of 50/50 reads and writes (workload A). Allow this traffic to run for the duration of the upgrade.
3. Using `sysstat`, verify the I/O workload in progress.
4. Upgrade the controller from ONTAP version 9.3x9 to 9.3x11.

Test Results

1. Verify I/O generation on the controller using sysstat. The I/Os should not fail during the upgrade and should continue upon upgrade.
2. Verify that the upgrade completes successfully. The first screenshot displays the operation before an upgrade with I/O in progress. The second screenshot displays MongoDB I/O in progress and after an upgrade is complete.





8 Customer Use Case 5: Secure Tenant Separation

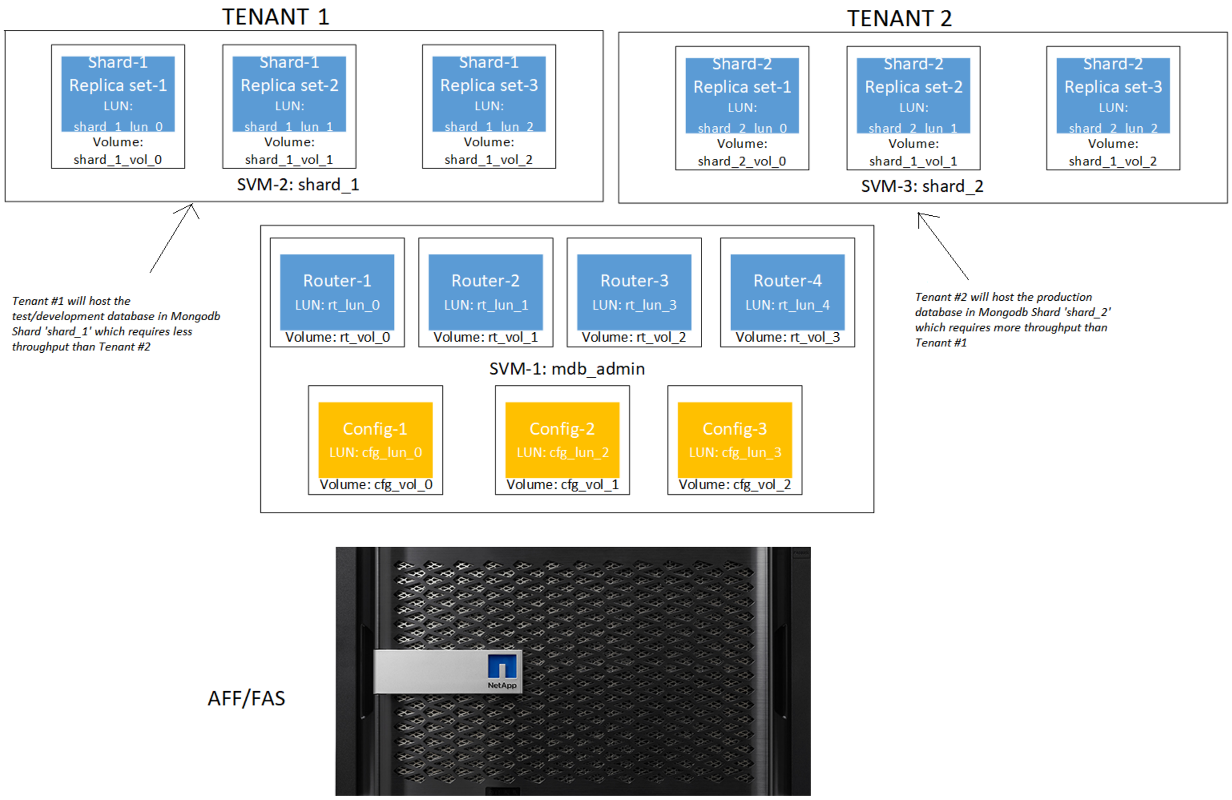
Organizations must understand how their service providers segment and isolate each customer in their cloud and enterprise infrastructure and how that manifests itself across the Data Fabric. It is also imperative that organizations know how their data is being secured. With secure multitenancy, multiple tenants can share infrastructure. Administrators can set the resource limits and quality of service (QoS) levels for each tenant. Tenants have administrative control over their provisioned environment, while remaining isolated from others.

Figure 3 shows the high-level details of the configuration used for the test. The MongoDB cluster uses three different tenants (or storage virtual machines [SVMs]) in the ONTAP cluster. The SVMs have the following configuration:

- mdb_admin. This SVM hosts the MongoDB routers and config/metadata servers. It includes four router LUNs and three config server LUNs, which are accessed by iSCSI protocol.
- shard_1. This SVM hosts the MongoDB shard `shard_1` with three replica set volumes. This shard hosts the development/test databases, which require less throughput than the production database in tenant 2.

- shard_2. This SVM hosts the MongoDB shard `shard_2` with three set volumes. This shard hosts the production databases, which require more throughput than the dev/test databases in tenant 1.

Figure 3) Test configuration.



Note: This configuration uses the traditional QoS policy instead of adaptive QoS policy because the `ycsb` load used fixed `recordcounts`, and therefore the volume usage did not increase. If the volume usage varies over time, you can use an adaptive QoS policy group to set the throughput ceiling or floor to volume size, maintaining the ratio of IOPS to TBps or GBps.

8.1 Creating MongoDB Tenants and Setting QoS for Tenant Management

The following steps describe how to set up MongoDB for separate production and development (MongoDB) tenants (multitenancy) and QoS for dynamic tenant management.

1. Create two tenants: a production tenant and a dev/test tenant. Multitenancy allows both tenants to be separate.
2. Determine the maximum throughput of the configuration.
3. Using QoS knobs, allocate 65K IOPS as the minimum throughput to the production tenant `shard_2` and 25K IOPS as the maximum throughput to the dev/test tenant `shard_1`. Applying this QoS setting triggers ONTAP to provide more resources to `shard_2` for more throughput than `shard_1`.
4. Using `ycsb`, create a workload that is a mix of 50/50 reads and writes (workload A). Because ONTAP supports multiprotocol, add NFS traffic in addition to iSCSI traffic created by `ycsb`. Allow this traffic to be run as a production tenant.
5. Using the `ycsb` client, create workload and NFS traffic on the dev/test tenant.
6. Slowly increase the workload to the dev/test tenant.

The QoS setting shows the throughput details for each QoS policy on a volume.

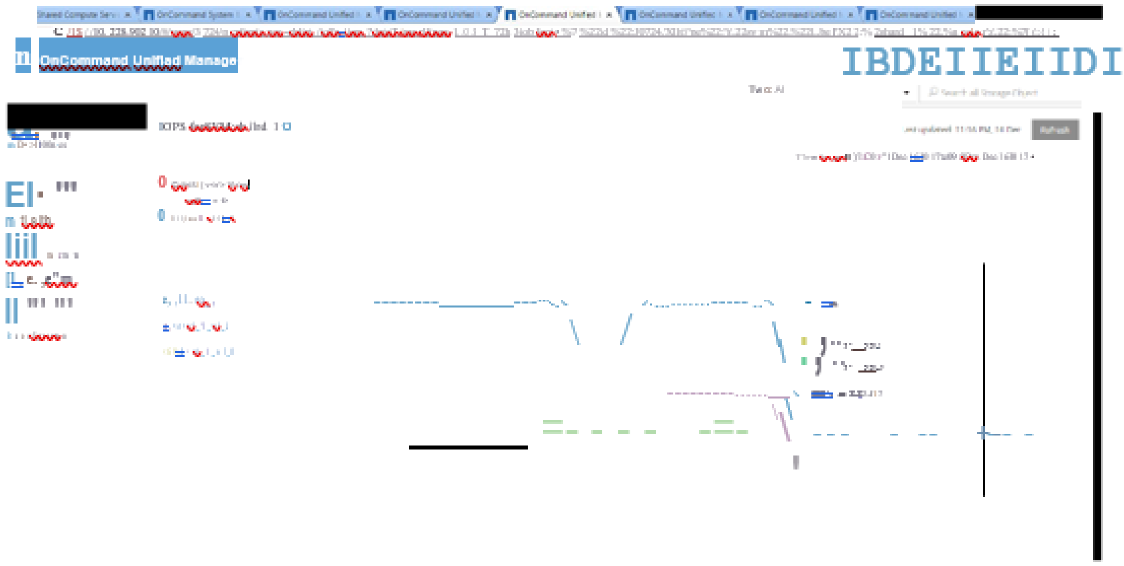
```
ste-pm8080-01::*> qos policy-group show -policy-group shard_*
Name             Vserver     Class        Wklds Throughput
---------------- ----------- ------------ ----- ------------
shard_1_vol_0    shard_1     user-defined 1     0-25000IOPS
shard_1_vol_1    shard_1     user-defined 1     0-25000IOPS
shard_1_vol_2    shard_1     user-defined 1     0-25000IOPS
shard_2_vol_0    shard_2     user-defined 1     65000IOPS-INF
shard_2_vol_1    shard_2     user-defined 1     65000IOPS-INF
shard_2_vol_2    shard_2     user-defined 1     65000IOPS-INF
6 entries were displayed.
```

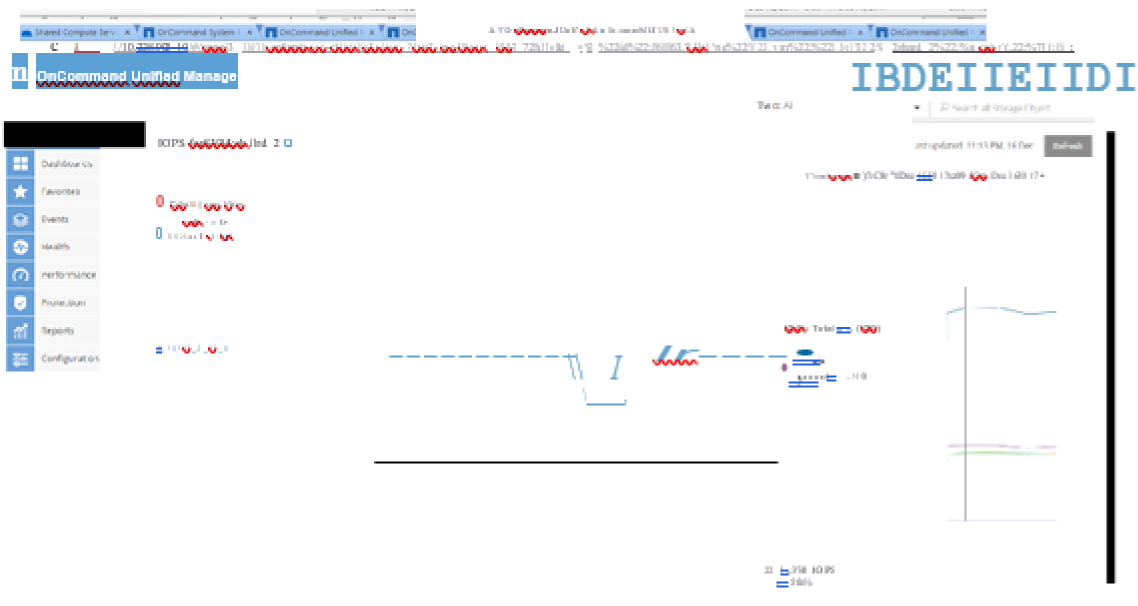
```
ste-pm8080-01::*> vol show -vserver shard_* -fields qos-policy-group
vserver volume            qos-policy-group
------- ----------------- ----------------
shard_1 mdb_2_shard_1_root -
shard_1 shard_1_vol_0      shard_1_vol_0
shard_1 shard_1_vol_1      shard_1_vol_1
shard_1 shard_1_vol_2      shard_1_vol_2
shard_2 shard_2_root       -
shard_2 shard_2_vol_0      shard_2_vol_0
shard_2 shard_2_vol_1      shard_2_vol_1
shard_2 shard_2_vol_2      shard_2_vol_2
8 entries were displayed.
```

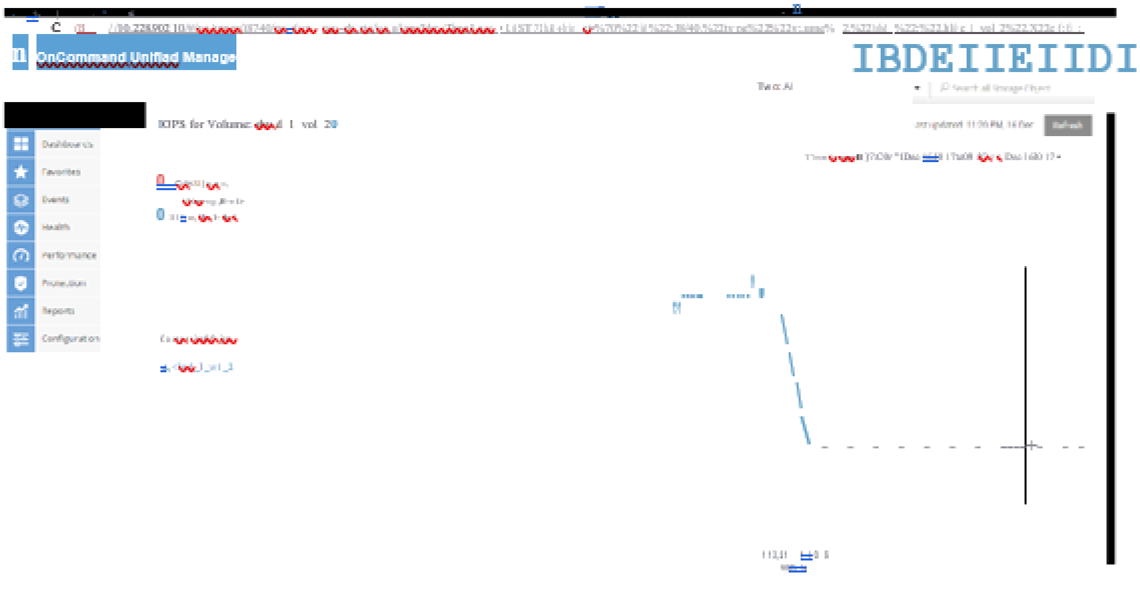
Test Results

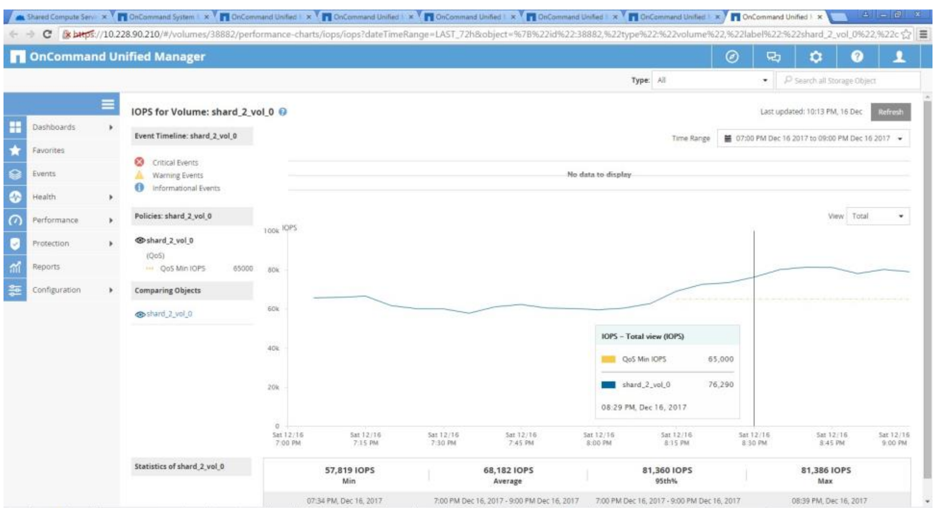
Verify that QoS is honored. The maximum throughput for the dev/test tenant should not exceed 25K IOPS throughput.

As expected, throughput was reduced below the 25K IOPS (ceiling) on dev/test tenants, and throughput increased above 65K IOPS (floor) on production tenants. This result is displayed in the performance graphs from OnCommand® Unified Manager 7.3, where the `shard_1` is the dev/test tenant, and `shard_2` is the production tenant.


OnCommand Unified Manage


OnCommand Unified Manage
Dashboards
Favorites
Events
Health
Performance
Protection
Reports
Configuration





Similarly, the following table shows the result of applying the QoS policy throughput values.

QOS Status	Dev/Test tenant (Shard_1) IOPS *(Ceiling qos_policy: max-throughput=25000IOPS)*			Production tenant (Shard_2) IOPS *(Floor qos policy: min-throughput=65000IOPS)*		
	shard_1_vol_0	**shard_1_vol_1**	**shard_1_vol_2**	**shard_2_vol_0**	**shard_2_vol_1**	**shard_2_vol_2**
Before applying QOS policy	54,332	59,023	114,222	60,110	61,614	60,715
After applying QOS policy	23,261	20,241	24,995	81,386	73,085	73,439

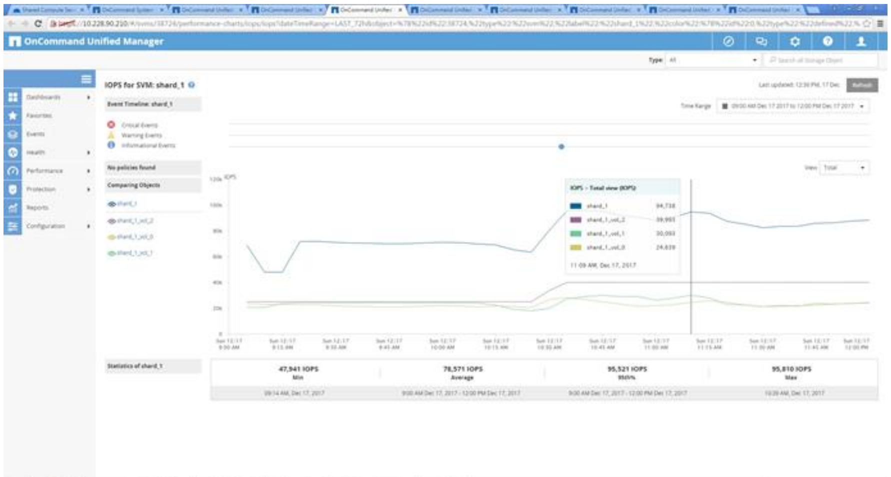
8.2 Enabling SLA Management in Data Fabric

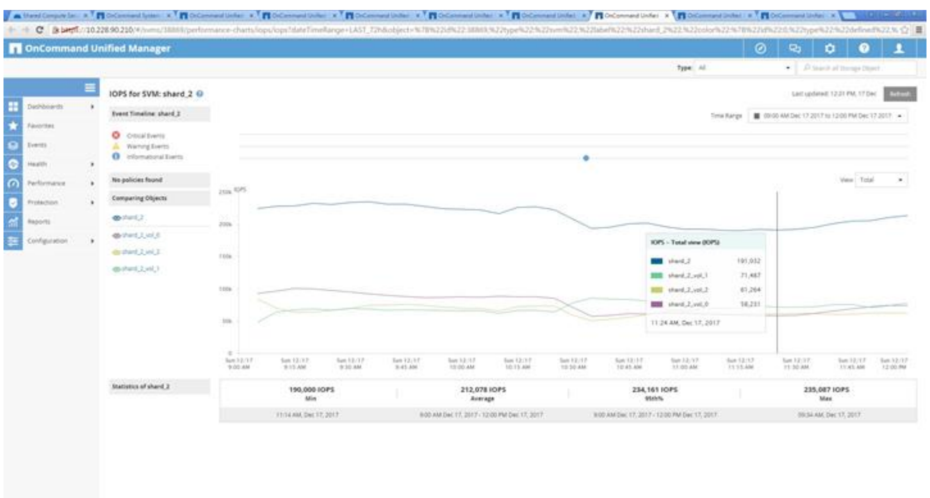
1. Use QoS to change the allocation of minimum throughput from 65K to 45K IOPS for the production tenant and maximum throughput from 25K to 40K IOPS for the dev/test tenant.
2. Add traffic by increasing the `ycsb` client and NFS workloads to the dev/test tenant.

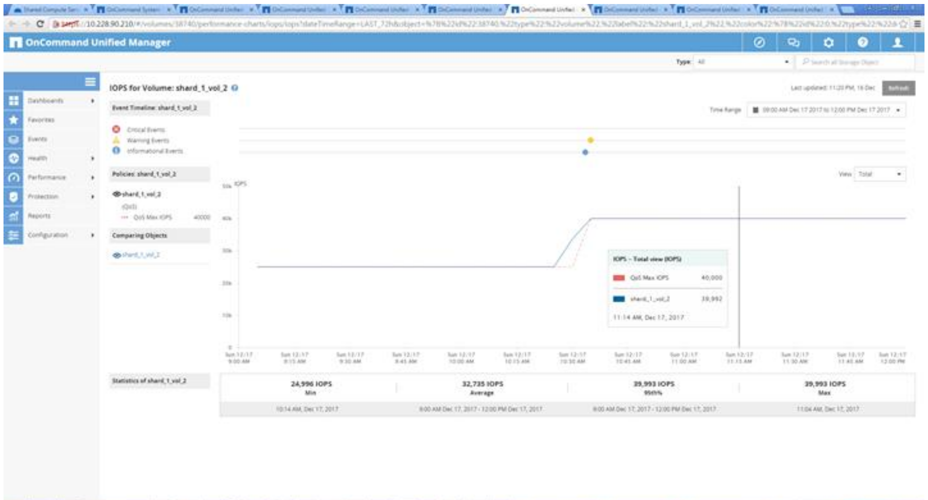
Test Results

Verify that the maximum throughput for the dev/test tenant should increase from 25K to 40K IOPS. At the same time, we can check that the minimum throughput on the production tenant drops from 65K to 45K IOPS.

As expected, throughput crossed 25K but stayed below 40K IOPS on the dev/test tenant when QoS maximum limit (ceiling) was increased from 25K to 40K IOPS. Similarly, throughput was reduced to 50K IOPS on the production tenant when QoS minimum limit (floor) was reduced from 65K to 45K IOPS. This result is displayed in the performance graphs from OnCommand Unified Manager 7.3 where the `shard_1` is the dev/test tenant and `shard_2` is the production tenant.







Similarly, the following table shows the result of modifying QoS policy throughput values.

QOS Status	Dev/Test tenant (Shard_1) IOPS *(Current qos_policy: max-throughput=25000IOPS)* *(new policy: max-throughput=40000IOPS)*			Production tenant (Shard_2) IOPS *(current qos policy: min-throughput=65000IOPS)* *(new policy: max-throughput=45000IOPS)*		
	shard_1_vol_0	**shard_1_vol_1**	**shard_1_vol_2**	**shard_2_vol_0**	**shard_2_vol_1**	**shard_2_vol_2**
Before modifying QOS policy	23,261	20,241	24,995	81,386	73,085	73,439
After modifying QOS policy	24,629	30,093	39,993	58,231	71,487	61,264

8.3 Multitenancy and ONTAP Resiliency

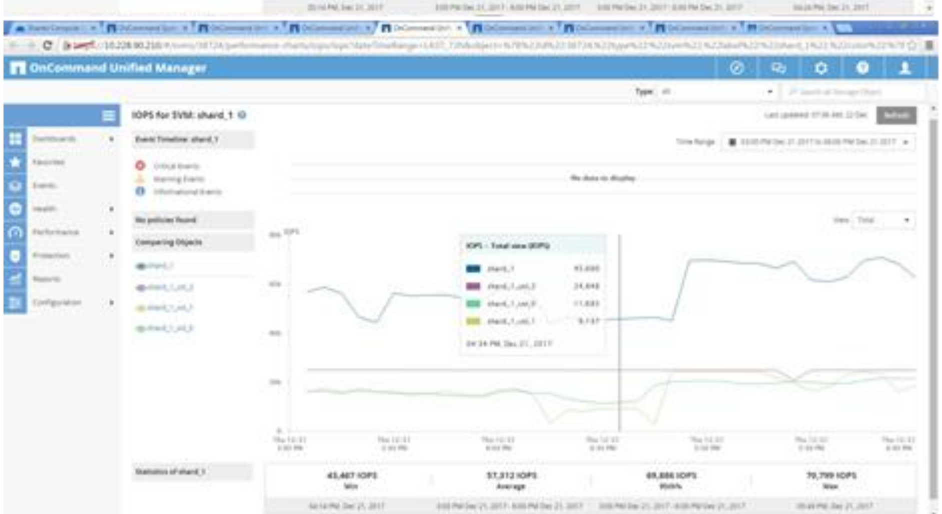
1. Create two tenants: a production tenant and a dev/test tenant; multitenancy allows both tenants to be separate.
2. Determine the maximum throughput of the configuration.
3. Using QoS knobs, allocate 45K IOPS as minimum throughput to the production tenant `shard_2` and 25K IOPS as maximum throughput to the dev/test tenant `shard_1`. Applying these QoS settings triggers ONTAP to provide more resources to `shard_2` for more throughput than `shard_1`.

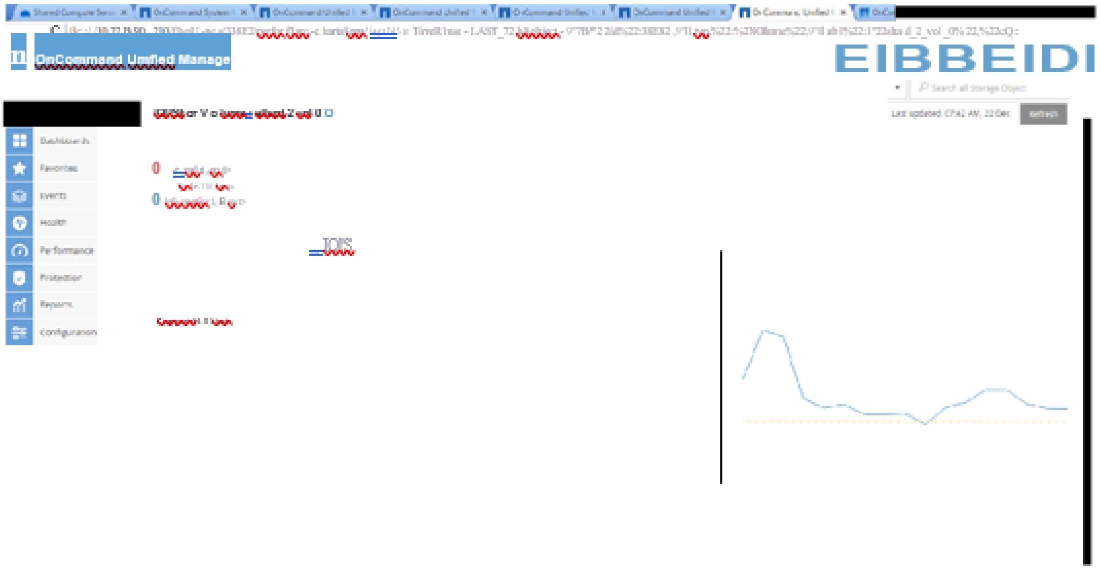
4. Using `ycsb`, create a workload that is a mix of 50/50 reads and writes (workload A). Because ONTAP supports multiple protocols, add NFS traffic in addition to iSCSI traffic created by `ycsb`. Allow this traffic to be run as a production tenant.
5. Using the `ycsb` client, create workload and NFS traffic on the dev/test tenant.
6. Perform a storage takeover. Wait for five minutes and then perform storage giveback.

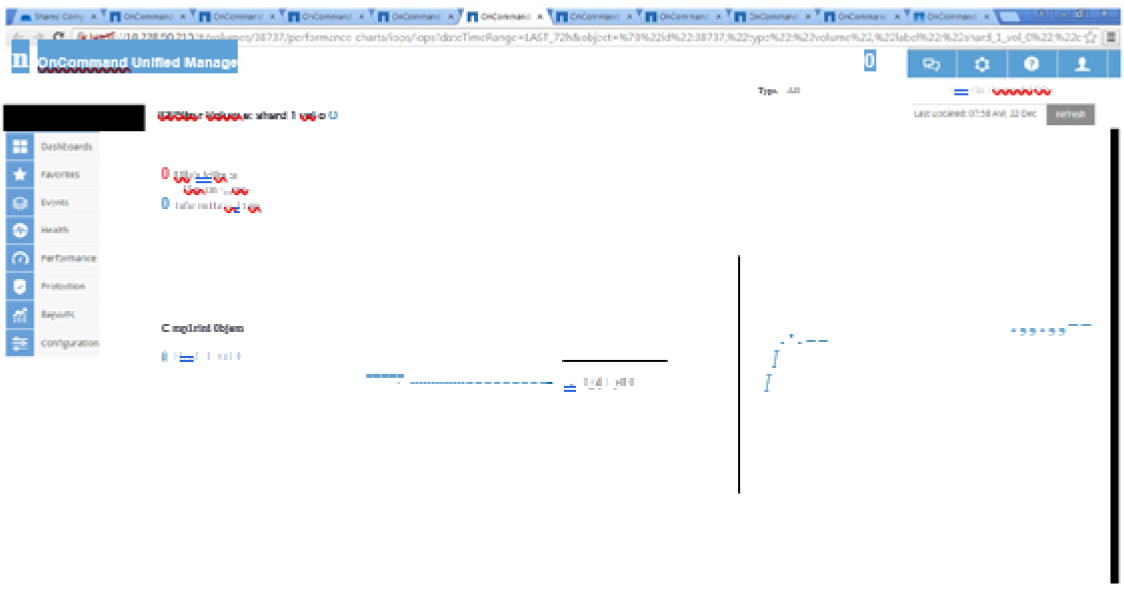
Test Results

Verify that there is no change to `ycsb` I/Os during storage takeover/giveback and the QoS policies continue to be honored.

As expected, the I/O on the volumes was not interrupted during takeover/giveback, and throughput stayed below 25K IOPS (ceiling) on dev/test tenants. Similarly, throughput stayed above 45K IOPS (floor) on the production tenants. This result is displayed in the performance graphs from OnCommand Unified Manager 7.3 where `shard_1` is the dev/test tenant and `shard_2` is the production tenant.







9 Customer Use Case 6: Reduce TCO with FlexClone and Variable Compression Groups

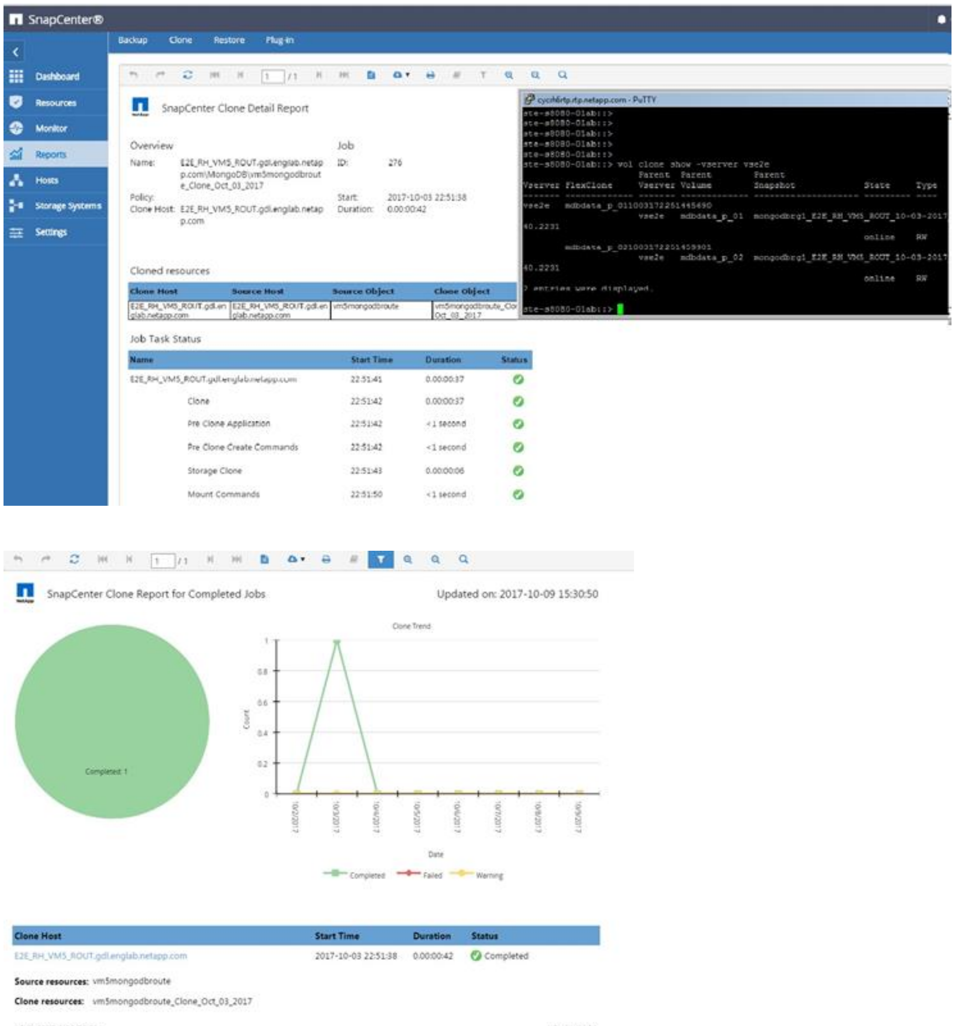
ONTAP provides a host of features targeting space savings, which is critical for reducing the TCO by reducing the storage footprint. You can run deduplication, data compression, and data compaction together or independently on a NetApp FlexVol® volume or an Infinite Volume to achieve optimal space savings. Deduplication eliminates duplicate data blocks, and data compression compresses the data blocks to reduce the amount of physical storage that is required. Cloning enables you to store more data in less space to increase storage efficiency.

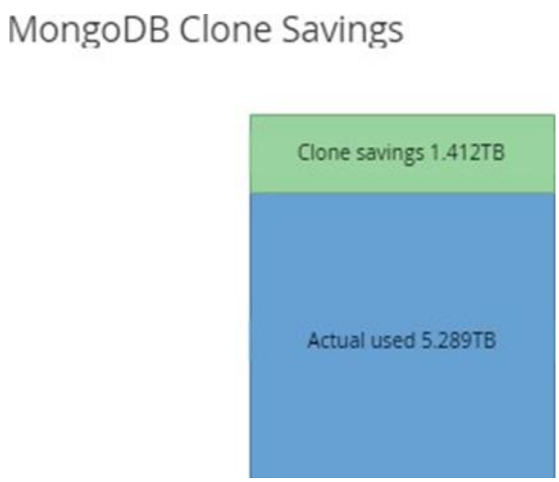
9.1 FlexClone and MongoDB Space Savings

NetApp FlexClone technology enables significant clone space savings by allowing you to create efficient copies of volumes, files, and LUNs. A FlexClone volume, FlexClone file, or FlexClone LUN is a writable, point-in-time image of the FlexVol volume or another FlexClone volume, file, or LUN. This technology enables you to store only data that changes between the parent and the clone.

FlexClone technology can be used to save space, power, and cost. Additionally, FlexClone has the same high performance as the parent volumes. Figure 4 illustrates the space savings of test and development storage with FlexClone volumes:

Figure 4) Space saving with FlexClone volumes.





9.2 Clone from a Backup Copy

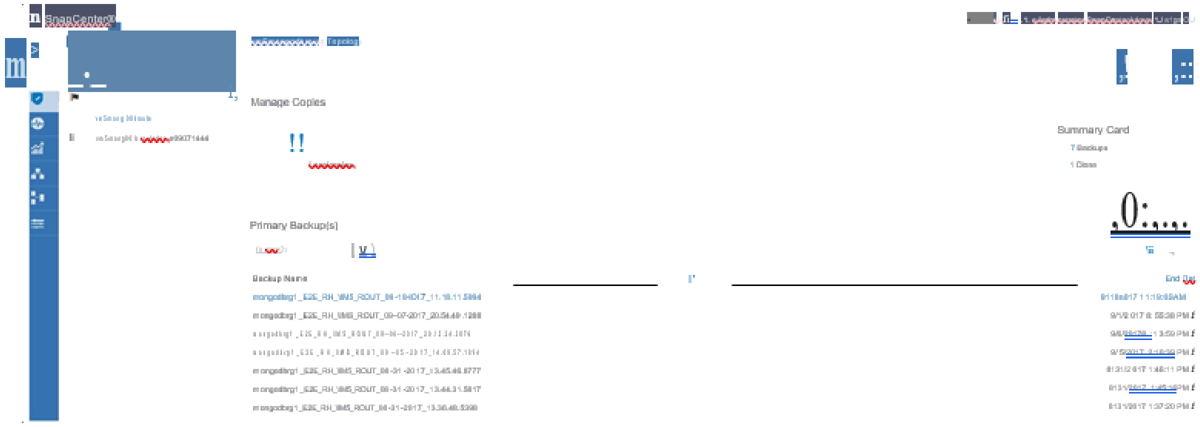
1. Using `ycsb`, create a MongoDB database about 50GB in size.
2. Using `ycsb`, create a workload that is a mix of 50/50 reads and writes (workload A). Allow this traffic to run for about 120 seconds.

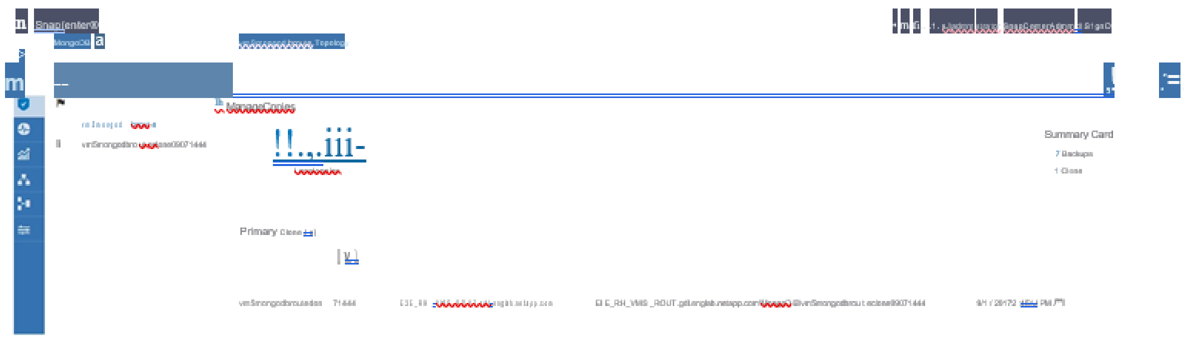
Using `sysstat`, verify the progress of the I/O workload.

3. Using the SnapCenter GUI, perform a backup to a local volume.
4. Verify the backup is complete and that a point-in-time backup copy is available.
5. Using the SnapCenter GUI, create a clone of the backup copy.
6. Verify the clone completes and that the cloned copy is mounted properly on a host.

Test Results

The clone is created and mounted properly on a host.





Conclusion

This document highlights specific ONTAP features and the advantages of a Data Fabric for a MongoDB ecosystem. The tests cover the common customer use cases in a MongoDB environment, and the results from the tests showcase the resiliency, high availability, and durability of a Data Fabric. The SnapCenter use cases highlight the SnapCenter centralized GUI management capability for monitoring, notification, logging, and reporting in the MongoDB ecosystem as well as its ability to enable backup, restore, and clone management of MongoDB applications.

Where to Find Additional Information

To learn more about the information described in this document, refer to the following documents and/or websites:

- TR-4492: MongoDB on the NetApp Data Fabric
- NetApp Interoperability Matrix Tool (IMT)
- NetApp Documentation Center

Version History

Version	Date	Document Version History
Version 1.0	January 2018	Initial release

Refer to the Interoperability Matrix Tool (IMT) on the NetApp Support site to validate that the exact product and feature versions described in this document are supported for your specific environment. The NetApp IMT defines the product components and versions that can be used to construct configurations that are supported by NetApp. Specific results depend on each customer's installation in accordance with published specifications.

TR-4660-0118

NetApp