Technical Report

Accelerate SQL Server 2012 AlwaysOn Availability Groups Deployment on NetApp Storage

Pat Sinthusan, NetApp
October 2012 | TR-4106

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 Introduction

The demand for higher availability and disaster recovery for database systems is creating a need to have geographically dispersed databases. This means that it is imperative not only to understand where all the critical and sensitive information resides, but also to make sure it is backed up consistently and securely for a timely recovery.

The higher demand for data availability has driven Microsoft to create solution features such as AlwaysOn Availability Groups. The business demand for high uptime requires that solution builders and administrators remove unplanned data outages. A solid data protection strategy makes sure of accessibility and availability of that data whenever and wherever it is needed to keep business running with as little interruption as possible.

2 Audience

This white paper is intended for NetApp® employees, partners, and customers, including IT planners, storage architects, SQL Server database administrators, and NetApp field personnel who are tasked with deploying such a solution in a customer environment. It is assumed that the reader is familiar with the various components of the solution.

3 AlwaysOn Availability Groups Overview

Microsoft® SQL Server® 2012 AlwaysOn Availability Groups are a data replication feature that provides a high-availability solution for both partial and complete site failures. AlwaysOn Availability Groups protect against data loss by replicating data changes from a source database instance, called the primary, to replica instances, called standby instances.

A partial site failure can be caused by hardware, network, or software failure, and without AlwaysOn Availability Groups, the database management system (DBMS) server or the machine where the database resides has to be rebooted. The length of time it takes to bring a database back online after a reboot can be unpredictable. It can take several minutes before the database is brought back to a consistent state and made available. AlwaysOn Availability Groups allow the standby database to take over database operations within seconds. The SQL Server native access client (SNAC) does not have to change since it can utilize the availability group listener, so users and applications don't need to reconnect to the database.

AlwaysOn Availability Groups also allow you to configure up to four additional availability replicas and allow read-only queries to be run on these secondary replicas. A secondary replica with read-only connections provides read-only access to secondary databases within the context of the availability group. The data on secondary replicas is updated in near real time, depending on replication method and the network bandwidth between SQL Server instances.

Table 1) Summarized benefits of AlwaysOn Availability Groups and NetApp storage.

Flexible	Integrated	Efficient	NetApp
• Multidatabase failover • Multiple secondaries • Total of 4 secondaries • 2 synchronous secondaries	• Application failover using virtual name • Configuration wizard • Dashboard • System center integration	• Active secondary • Readable secondary • Backup from secondary • Improves primary server performance by	• Accelerate set up of AlwaysOn Availability Groups • Quick and efficient backups of databases in AlwaysOn

Flexible	Integrated	Efficient	NetApp
• 1 automatic failover pair • Synchronous and asynchronous data movement • Built-in compression and encryption • Auto-page repair • Automatic and manual failover (new design) • Flexible failover policy	• Rich diagnostic infrastructure • File-stream replication • Replication publisher failover	offloading work to secondary • Monitoring and Troubleshooting enhanced • Automation using Windows PowerShell™	Availability Groups • Improved provisioning of secondary databases • Improved deployment of remote secondary databases • Create space-efficient clones of databases in AlwaysOn Availability Groups • Automation using NetApp Data ONTAP® PowerShell Toolkit

4 Terminology

Table 2 lists terminology and examples that have been used to develop this document.

Table 2) Terminology and examples.

Term	Definition	Example in This Document
Server	Windows® host server where SQL Server is installed	SMBInstall.sea-tm.netapp.com
Controller	NetApp storage controller	Eos and Aura
WFC	Windows failover cluster	SQL12Cluster
AG	Availability groups	Adventureworks_AG

5 Software Requirements

The software listed in Table 3 was installed and utilized in the making of this document:

Table 3) Installed software.

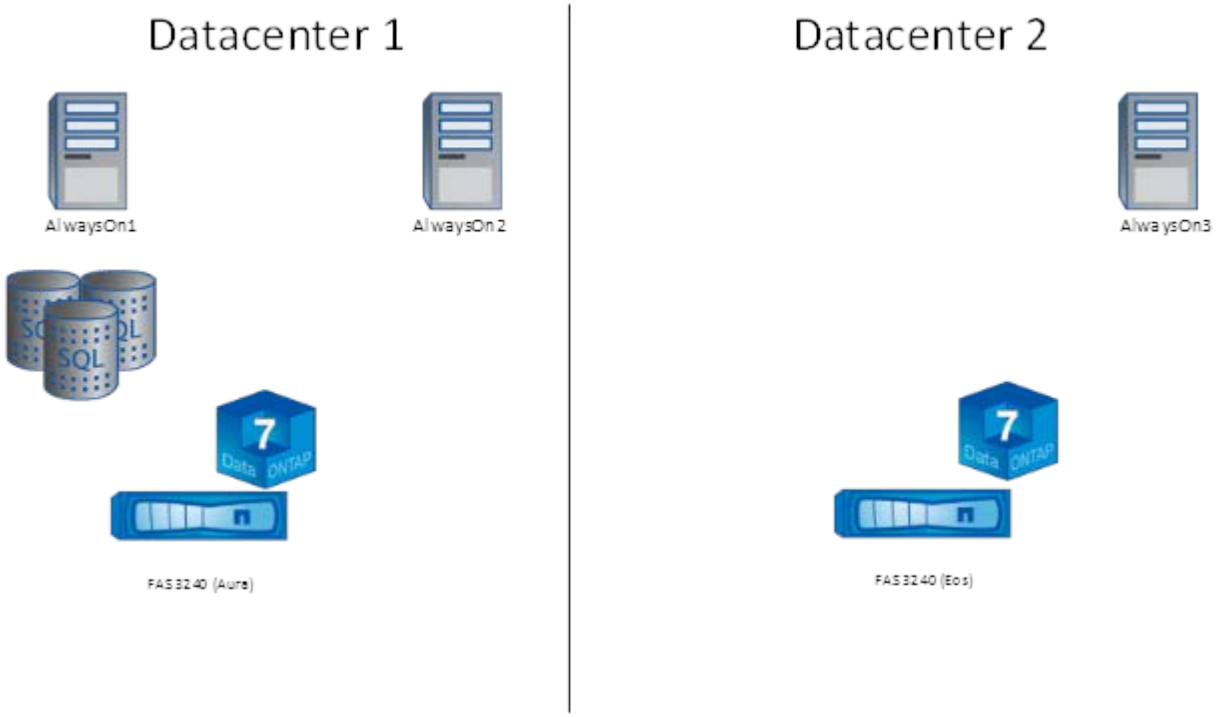
Hardware	Software
Server	• Microsoft Windows Server® 2008 R2 SP1 • Microsoft SQL Server 2012 x64 Enterprise Edition • NetApp Windows Host Utilities Kit 5.3 x64 • NetApp Data ONTAP DSM for Windows MPIO • NetApp SnapDrive® 6.4.1 • NetApp SnapManager® 6.0 for SQL Server x64* • NetApp Data ONTAP PowerShell Toolkit 2.0 • NetApp OnCommand® System Manager
NetApp Controller	• NetApp Data ONTAP 8.1 RC2 (running in 7-Mode)

*SnapManager for SQL Server is in prerelease at the time of publication of this technical report.

6 Topology

Demonstrating the creation of SQL Server 2012 AlwaysOn Availability Groups, this solution used three servers using Windows Server 2008 R2 servers (AlwaysOn1, AlwaysOn2, and AlwaysOn3). Two NetApp FAS3240 controllers (Aura and Eos) have been set up for database storage.

Figure 1) Topology used to produce this document.



The following are the steps required to deploy AlwaysOn Availability Groups.

6.1 Create Windows Failover Cluster

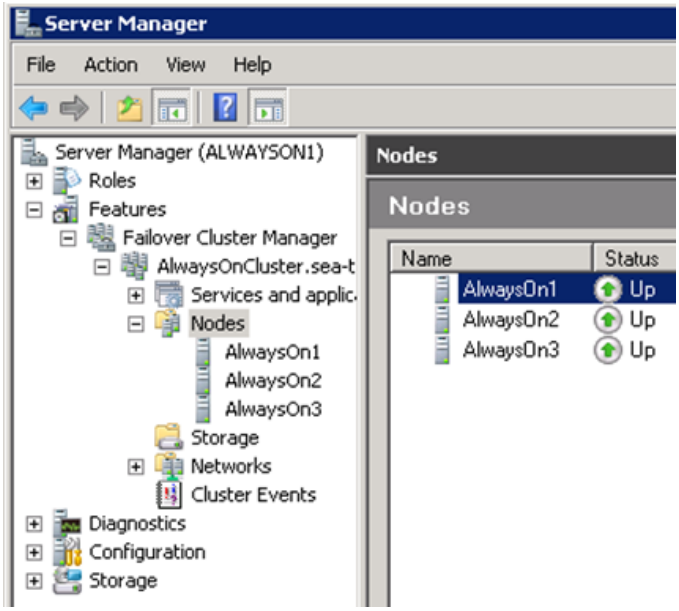
Because AlwaysOn Availability Groups utilize Windows failover cluster (WFC), all participating servers must have the failover cluster feature added. This process can be accomplished by running the following Windows PowerShell script on all the servers:

```
if ((Get-Module| select -exp name) -notcontains “ServerManager”) {Import-Module ServerManager}
Add-WindowsFeature Failover-Clustering
```

The Windows failover cluster can now be created by adding all server nodes (AlwaysOn1, AlwaysOn2, and AlwaysOn3) to the cluster. This can be automated with the following Windows PowerShell script:

```
if ((Get-Module| select -exp name) -notcontains “FailoverClusters”) { Import-Module
FailoverClusters}
$Clustername = "AlwaysOnCluster"
$Node1 = "AlwaysOn1"
$Node2 = "AlwaysOn2"
$Node3 = "AlwaysOn3"

#For testing cluster
$Result = Test-Cluster -Cluster $Clustername -Node $Node1, $Node2, $Node3
#Display the result of cluster verification
Invoke-Item $Result.versioninfo.filename
New-Cluster $Clustername -Node $Node1, $Node2, $Node3 –NoStorage
Set-ClusterQuorum –Cluster $ClusterName -NodeMajority
```

Figure 2) Windows failover cluster has been created, and all three servers have been added as nodes.



6.2 Enable AlwaysOn High Availability

After each server has been added to the WFC, AlwaysOn High Availability must be enabled for all of the SQL Server instances. The following Windows PowerShell script enables AlwaysOn High Availability and restarts the SQL Server service for each instance:

```
powershell.exe -ExecutionPolicy Unrestricted -NoLogo -NonInteractive
if ((Get-Module| select -exp name) -notcontains 'SQLPS') {Import-Module SQLPS -
DisableNameChecking}
$nodes = "AlwaysOn1", "AlwaysOn2", "AlwaysOn3"
foreach ($node in $nodes){
        Write-Output "Enable AlwaysOn Service for $node instance"
        Enable-SqlAlwaysOn -Path SQLSERVER:\SQL\$node\Default –force
```

```
        Start-Sleep -Seconds 10
        Write-output "Restart SQL Server service for $node instance"
        Stop-Service -InputObject (get-Service -ComputerName $node -Name MSSQLServer) -force
        Start-Service -InputObject (get-Service -ComputerName $node -Name SQLServerAgent)
}
```

6.3 Create Initial Database Backup

In order to create an availability group, the database must be in full recovery mode, and at least one full backup must have been performed, for which we use SnapManager for SQL Server (SMSQL). SMSQL utilizes NetApp Snapshot™ technology to deliver near-instantaneous and space-efficient backups. After SMSQL has been installed, it needs to be configured to perform a full database and log backup. The following figures display the steps of configuring SMSQL.

Figure 3) Select database verification server in SnapManager for SQL Server Configuration wizard.

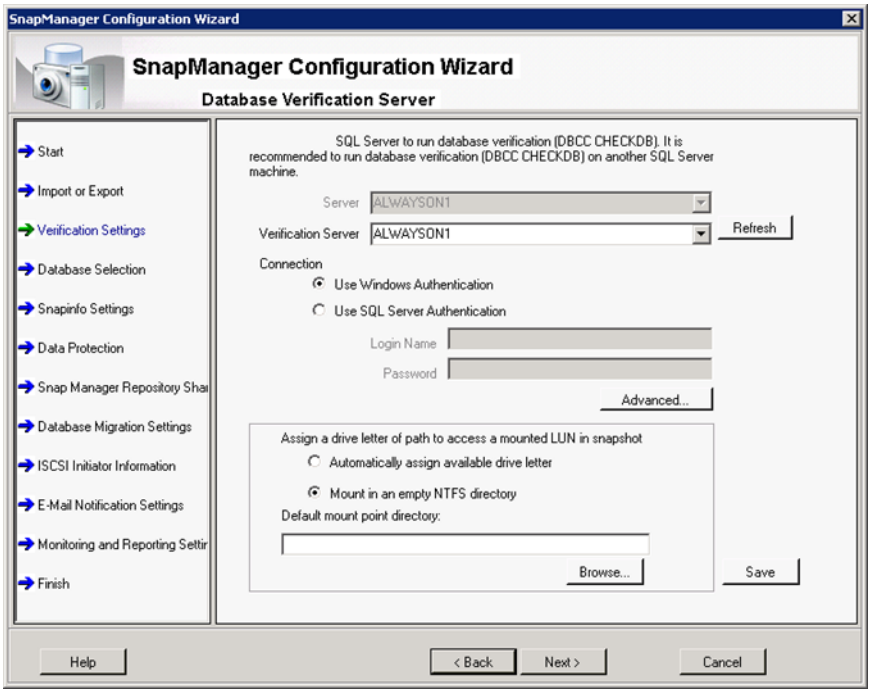


Figure 4) Select location of data and log files.

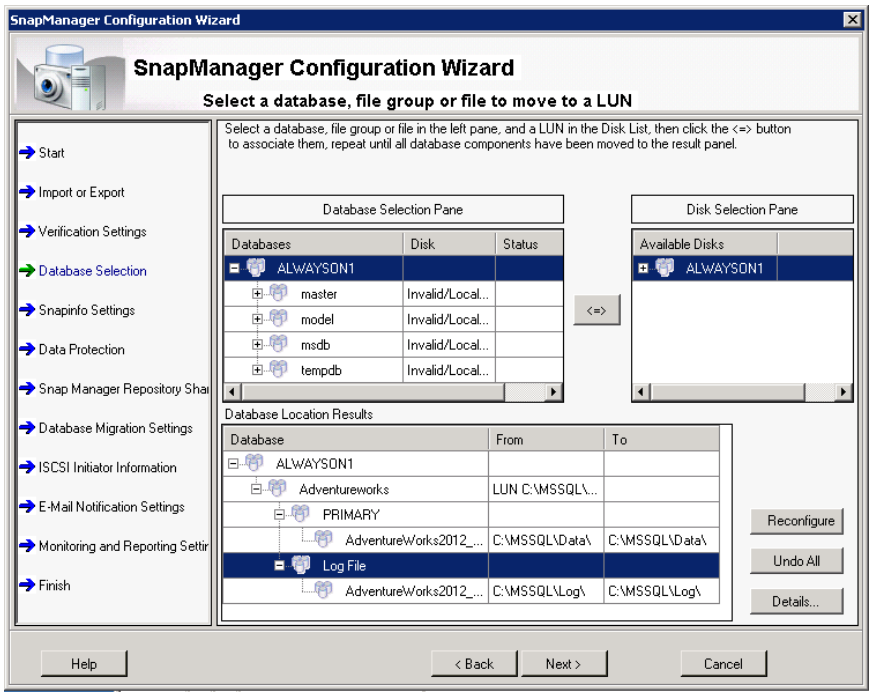


Figure 5) A single directory for SnapInfo metadata has been specified.

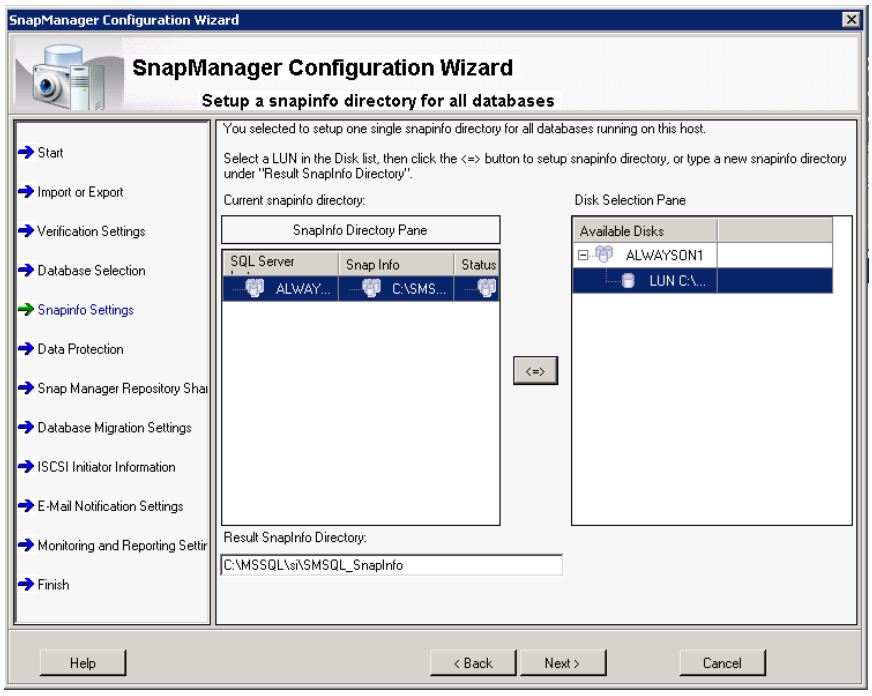
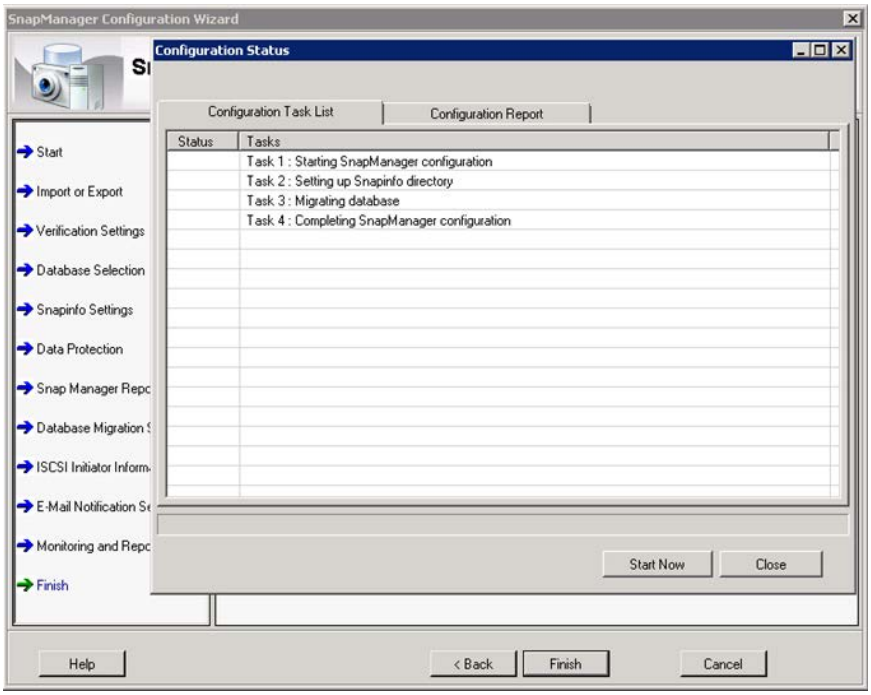


Figure 6) Start Configuration wizard.



After the SMSQL Configuration wizard has completed, the first full and log backup need to be performed in order to allow the database to join availability groups. This process can be completed with SMSQL using the following Windows PowerShell script:

```
powershell.exe -ExecutionPolicy Unrestricted -NoLogo -NonInteractive
if ((Get-Module| select -exp name) -notcontains 'SQLPS') {Import-Module SQLPS -
DisableNameChecking}
Add-PSSnapin NetApp.SnapManager.SQL.PS.Admin -ErrorAction silentlycontinue
new-backup  –svr 'AlwaysOn1' -d 'AlwaysOn1', '1', 'Adventureworks' -lb –mgmt daily
```

6.4 Create Endpoints and Availability Group Replica Objects

To create availability groups, all servers in a Windows failover cluster must be able to communicate. This communication can be set up using endpoints. The following script creates an endpoint object and availability group replica object for each server:

```
$node1 = “AlwaysOn1”
$node2 = “AlwaysOn2”
$node3 = “AlwaysOn3”
$database = “Adventureworks”
$agname = “AlwaysOn_AG”

$instance1 = Get-Item "SQLSERVER:\SQL\$node1\Default"
$instance2 = Get-Item "SQLSERVER:\SQL\$node2\Default"
$instance3 = Get-Item "SQLSERVER:\SQL\$node3\Default"

$endpoint1 = "TCP://" + $node1 + ":5022"
$endpoint2 = "TCP://" + $node2 + ":5022"
$endpoint3 = "TCP://" + $node3 + ":5022"

$replica1 = New-SqlAvailabilityReplica `
-Name $node1 `
-EndpointUrl $endpoint1 `
-FailoverMode "Automatic" `
       -AvailabilityMode "SynchronousCommit" `
       -AsTemplate `
```

```
        -Version ($instance1.Version)

$replica2 = New-SqlAvailabilityReplica `
        -Name $node2 `
        -EndpointUrl $endpoint2 `
        -FailoverMode "Automatic" `
        -AvailabilityMode "SynchronousCommit" `
        -AsTemplate `
        -Version ($instance2.Version)

# Note that there can only be 2 node in FailoverMode
# node3 need to be set to manual
$replica3 = New-SqlAvailabilityReplica `
        -Name $node3 `
        -EndpointUrl $endpoint3 `
        -FailoverMode "manual" `
        -AvailabilityMode "AsynchronousCommit" `
        -AsTemplate `
        -Version ($instance3.Version)
```

6.5 Create Availability Groups and Add Database to Availability Groups

After the endpoint objects have been established, the availability group can be created, and the principal database can be added with the following Windows PowerShell script:

```
# Create the Availability Group.
New-SqlAvailabilityGroup -InputObject $instance1 -Name $agname -AvailabilityReplica ($replica1,
$replica2, $replica3) -Database $database
```

The second and third replicas also can be added to the availability group using Windows PowerShell:

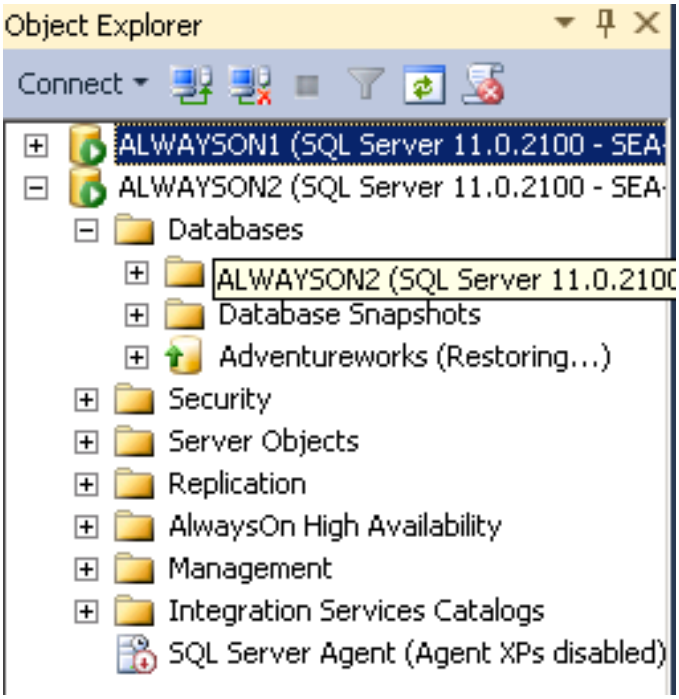
```
#Join the node2 replica to the availability group.
Join-SqlAvailabilityGroup -Path "SQLSERVER:\SQL\$node2\Default" -Name $agname

# Join the node3 replica to the availability group.
Join-SqlAvailabilityGroup -Path "SQLSERVER:\SQL\$node3\Default" -Name $agname
```

6.6 Back Up and Restore Database to Replica Instance in Same Data Center

At this point an availability group has been created. However, only the database in the primary instance has been added to it. In order to add the database from the replica to the availability group, the database on the primary (AlwaysOn1) needs to be backed up and restored with norecovery mode on the replica (AlwaysOn2). The backup and restore process for large databases can take a long time, and the backup file takes up a large amount of disk space. SMSQL provides a faster backup and restore process by using NetApp Snapshot and FlexClone® technologies. Furthermore, this process with SMSQL does not involve a backup file requiring storage space. The following Windows PowerShell script provides the backup and restores using the SMSQL clone command:

```
$node1 = “AlwaysOn1”
$node2 = “AlwaysOn2”
$database = “Adventureworks”
Add-PSSnapin NetApp.SnapManager.SQL.PS.Admin -ErrorAction silentlycontinue
$targetmountpoint = "C:\MSSQL\"
clone-database -Server $node1 -ServerInstance $node1 -Database $database `
-TargetServerInstance $node2 -TargetDatabase $database `
-TargetServerMountPointDir $targetmountpoint -RecoverDatabase $false –lb
```

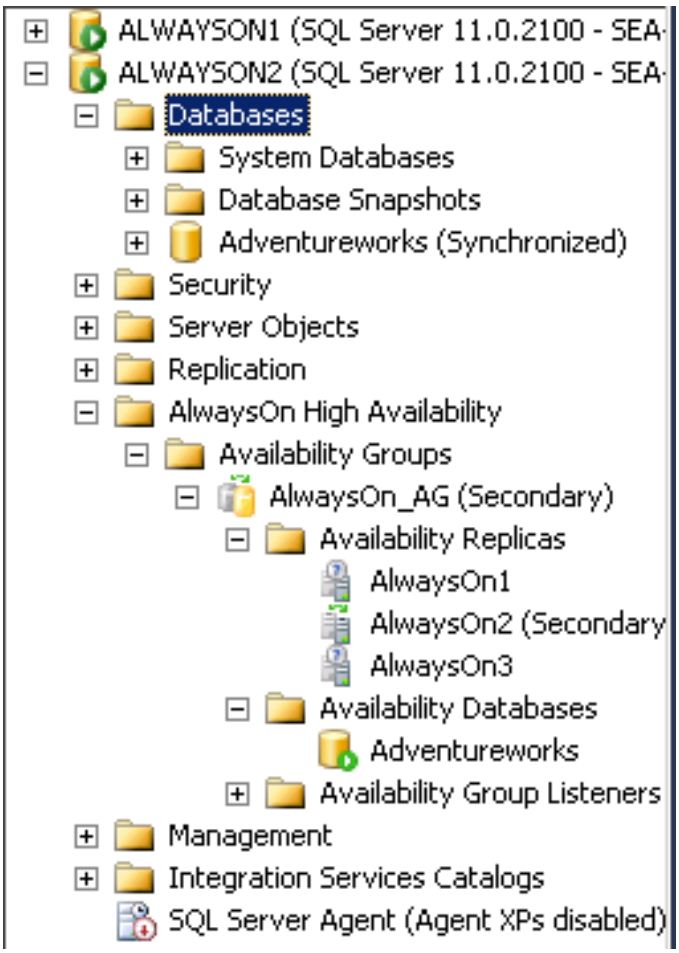
Figure 7) After restore completes, database in replica will be in norecovery mode.



The database in second replica is now ready to be added into the availability group. This task can be accomplished by the following Windows PowerShell script:

```
$availabilitygroup = get-item SQLSERVER:\SQL\$node2\DEFAULT\AvailabilityGroups\$agname
Add-SqlAvailabilityDatabase -InputObject $availabilitygroup -Database $database
```

Figure 8) Adventureworks database has been added to availability groups.



After the database in the SQL Server instances running on AlwaysOn2 has been added to the availability group (under Availability Databases), the FlexClone copy of the database volume can be split from the Snapshot copy. This allows the volume to be independent from original volume (where the primary database resides) and its Snapshot copy. This task can be performed using NetApp OnCommand System Manager.

Figure 9) The sdw_cl_alwayson1_data_0 has been identified as a clone volume of alwayson1_data.

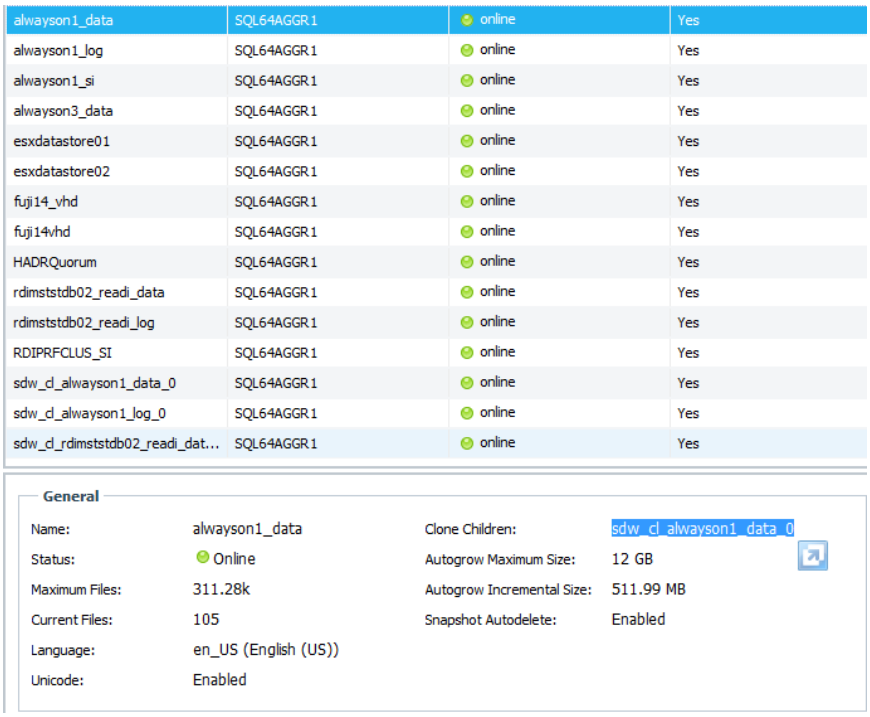
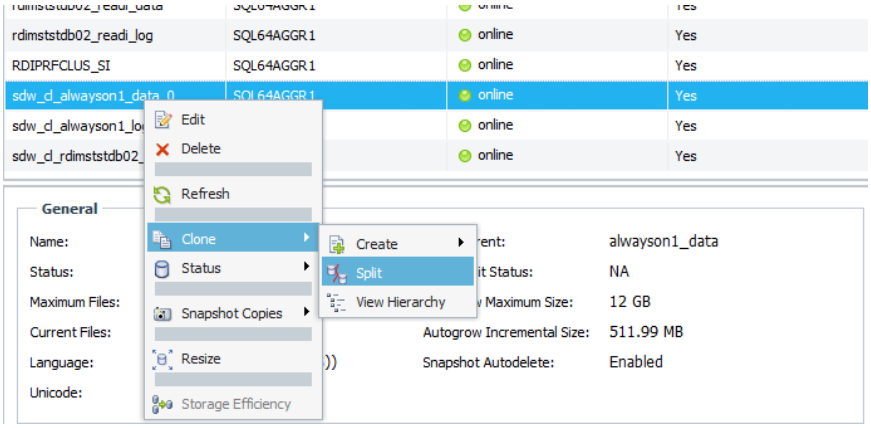
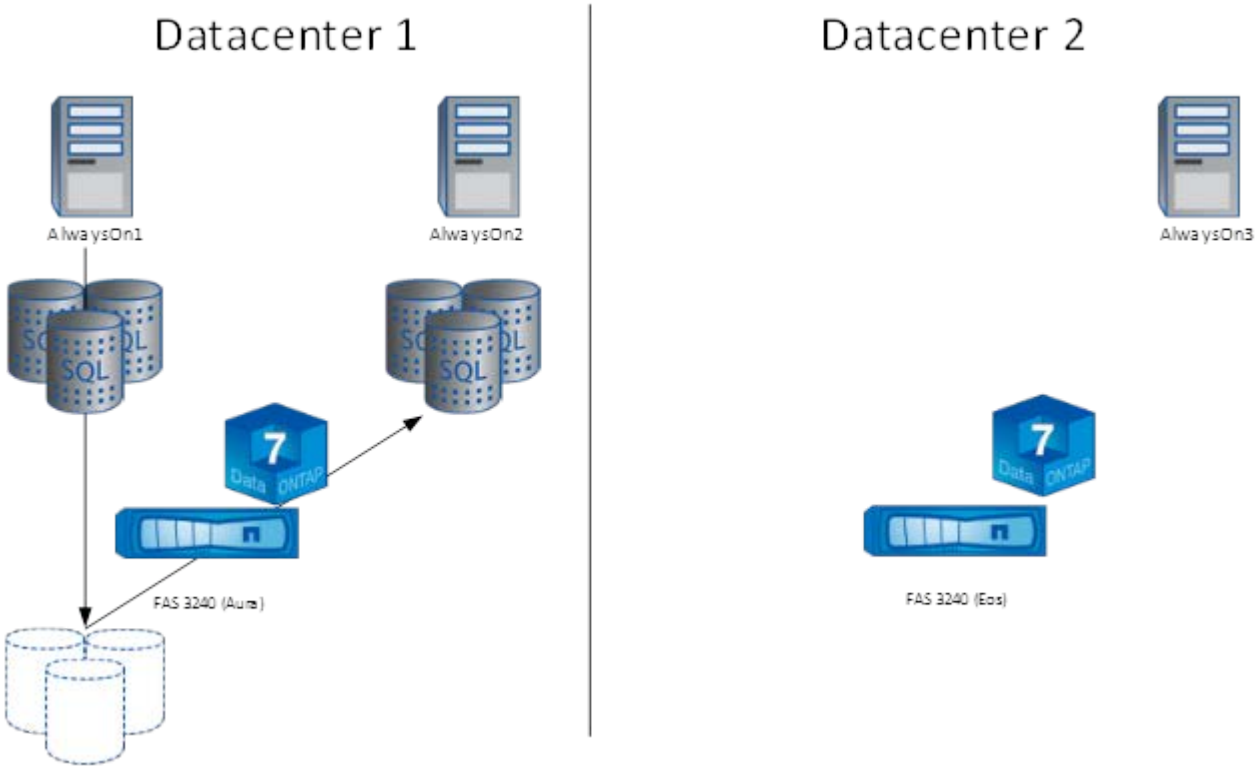


Figure 10) The sdw_cl_alwayson1_data_0 has been selected to be split from Snapshot copy.



The log volume can be split from the Snapshot copy using the same process.

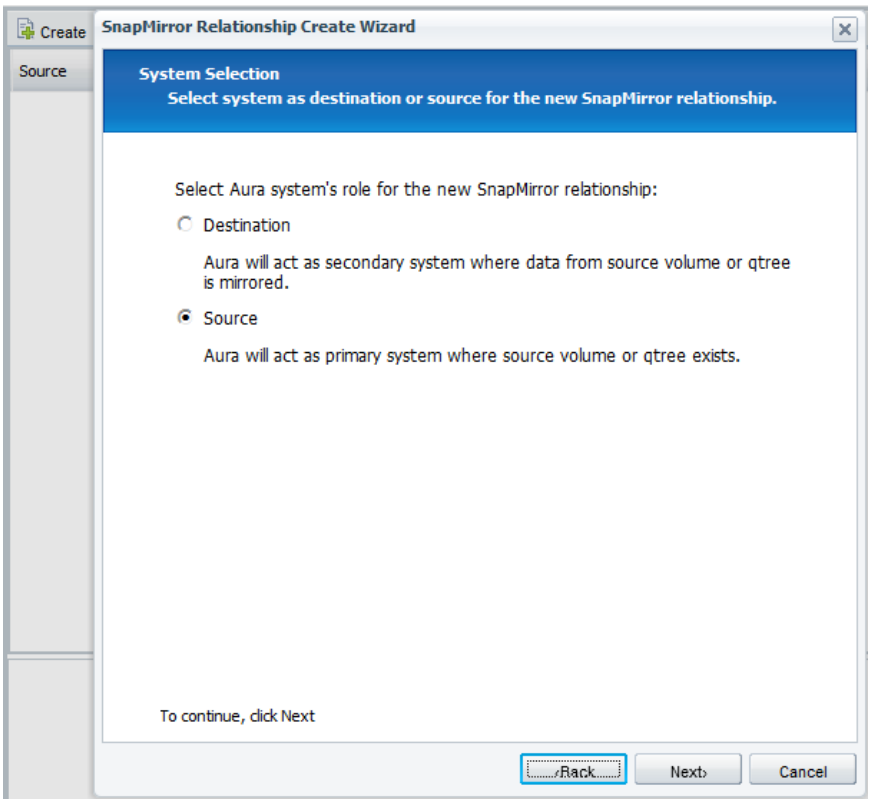
Figure 11) Database on AlwaysOn2 instance has been cloned, split, and added to availability group.

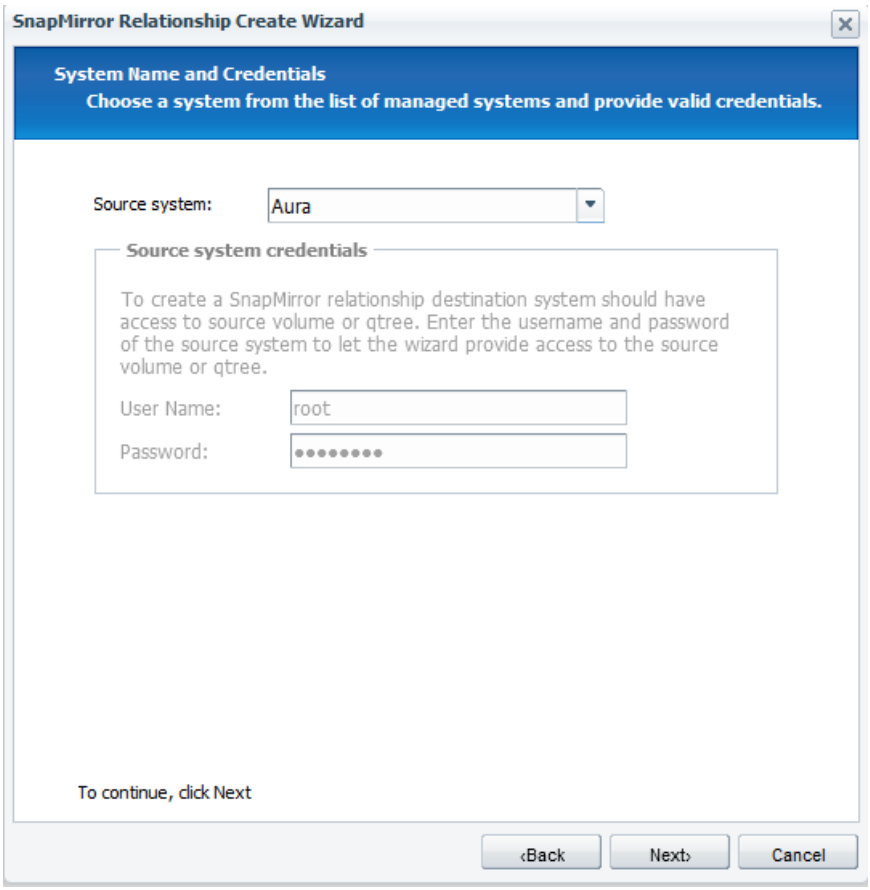


7 Back Up and Restore Database to Replica Instance in Different Data Centers

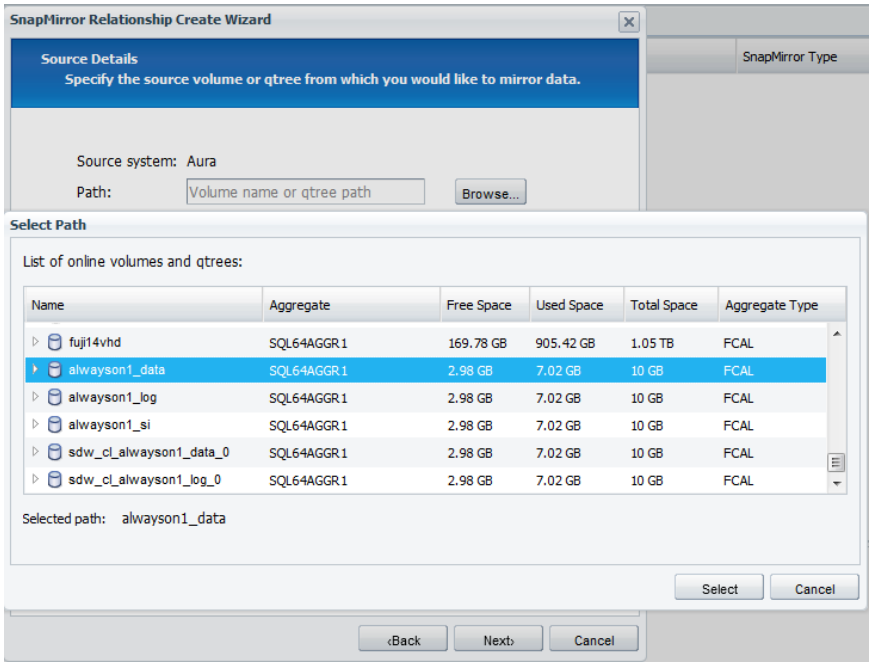
Network bandwidth limitations make it much more of a challenge to back up and restore databases with SQL Server between separate data centers. SMSQL and NetApp SnapMirror® make this task faster and simpler. In order to back up and restore databases in the remote locations, the SnapMirror relationship of the volumes that contain data and files must be created. This section describes how to create a SnapMirror relationship using NetApp OnCommand System Manager. In this example, a new relationship is created to mirror volume alwayson1_data from Aura to alwayson3_data volume in Eos controller.

1. In OnCommand System Manager, a new relationship can be created from either the source or destination controller. In this example, the source controller is selected. Click SnapMirror > and then Click Create.

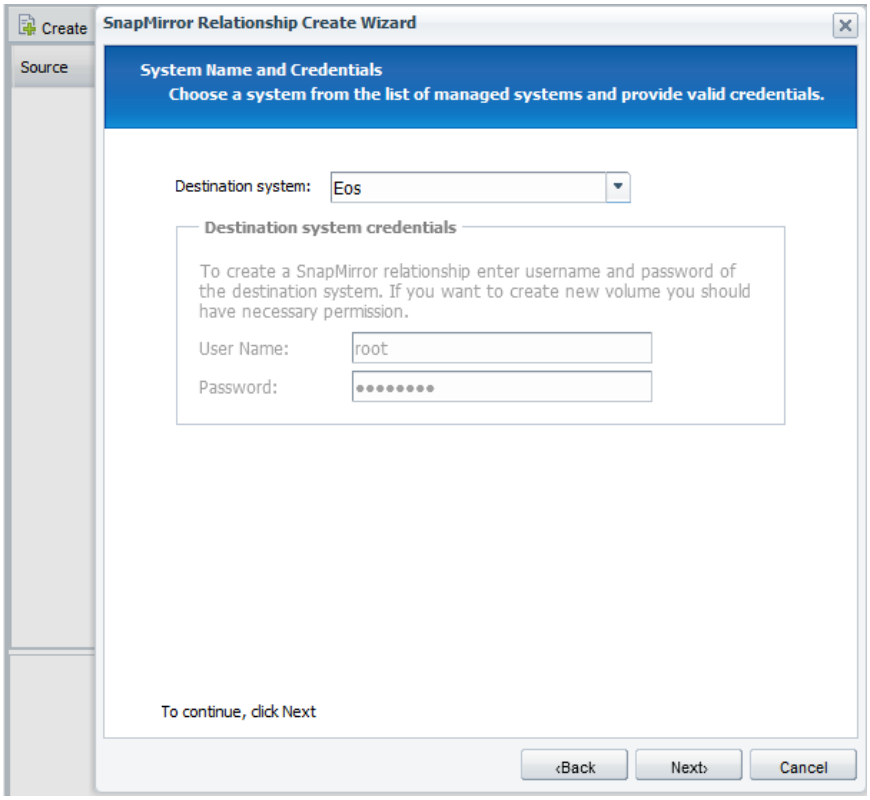

Create
Source
SnapMirror Relationship Create Wizard
System Selection
Select system as destination or source for the new SnapMirror relationship.
Select Aura system's role for the new SnapMirror relationship:
Destination
Aura will act as secondary system where data from source volume or qtree is mirrored.
Source
Aura will act as primary system where source volume or qtree exists.
To continue, click Next
‹Back
Next›
Cancel


SnapMirror Relationship Create Wizard
System Name and Credentials
Choose a system from the list of managed systems and provide valid credentials.
Source system:
Aura
Source system credentials
To create a SnapMirror relationship destination system should have access to source volume or qtree. Enter the username and password of the source system to let the wizard provide access to the source volume or qtree.
User Name:
root
Password:
To continue, click Next
‹Back
Next›
Cancel

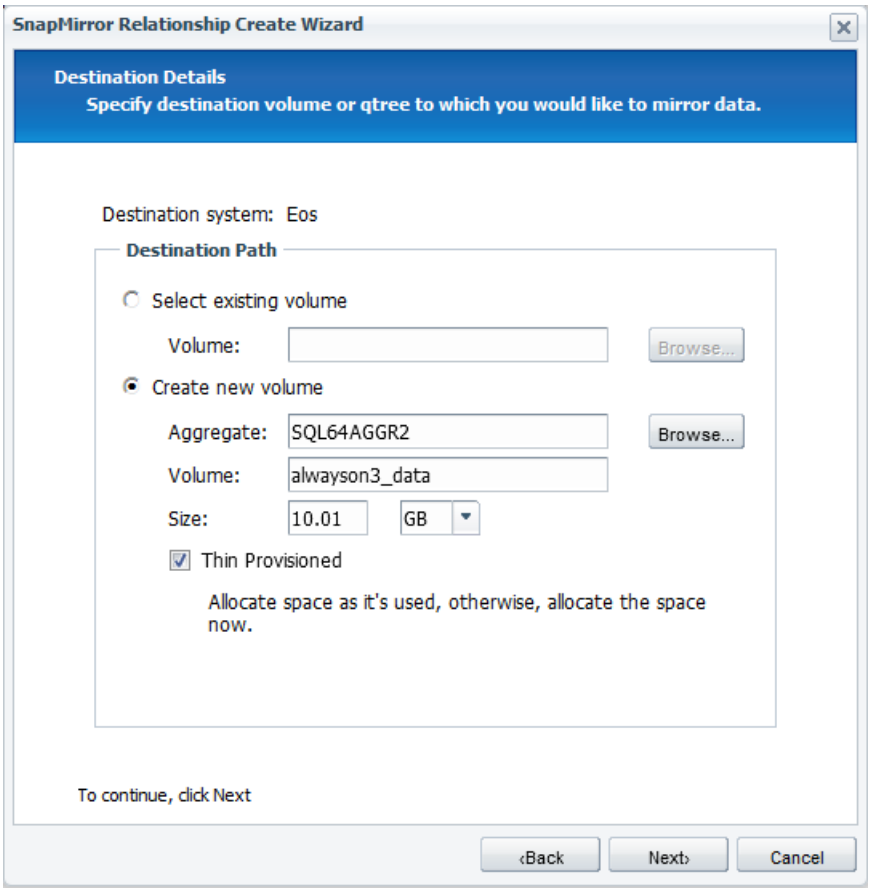
2. Select alwayson1_data volume as the source volume.



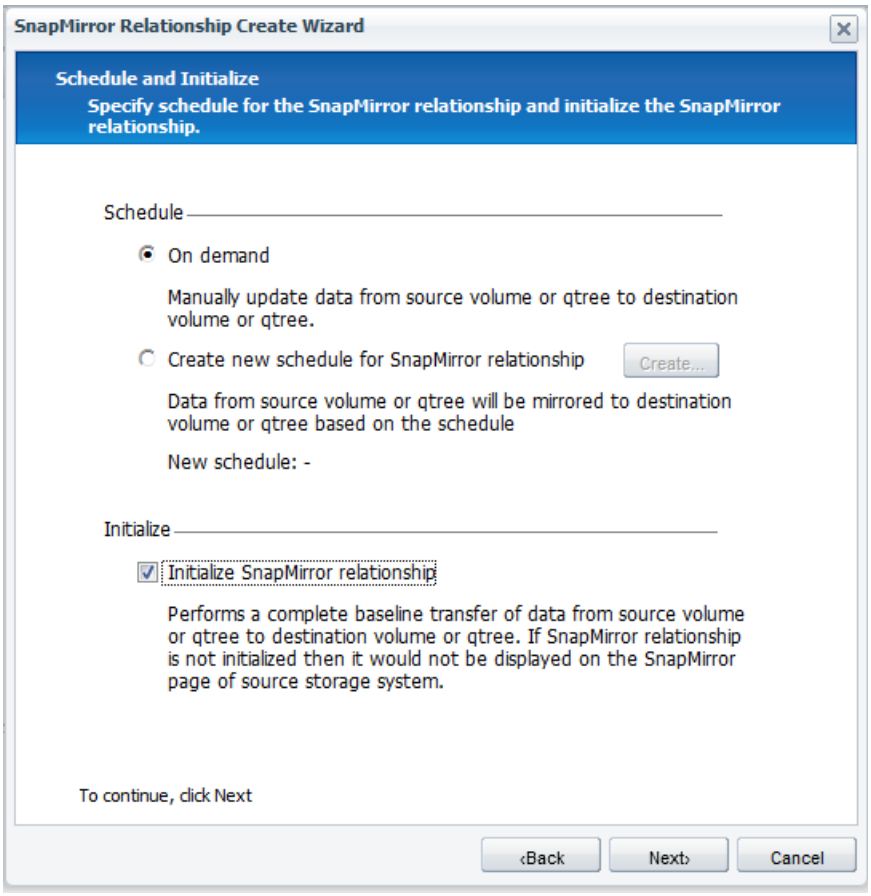
3. Select Eos controller as the destination system.



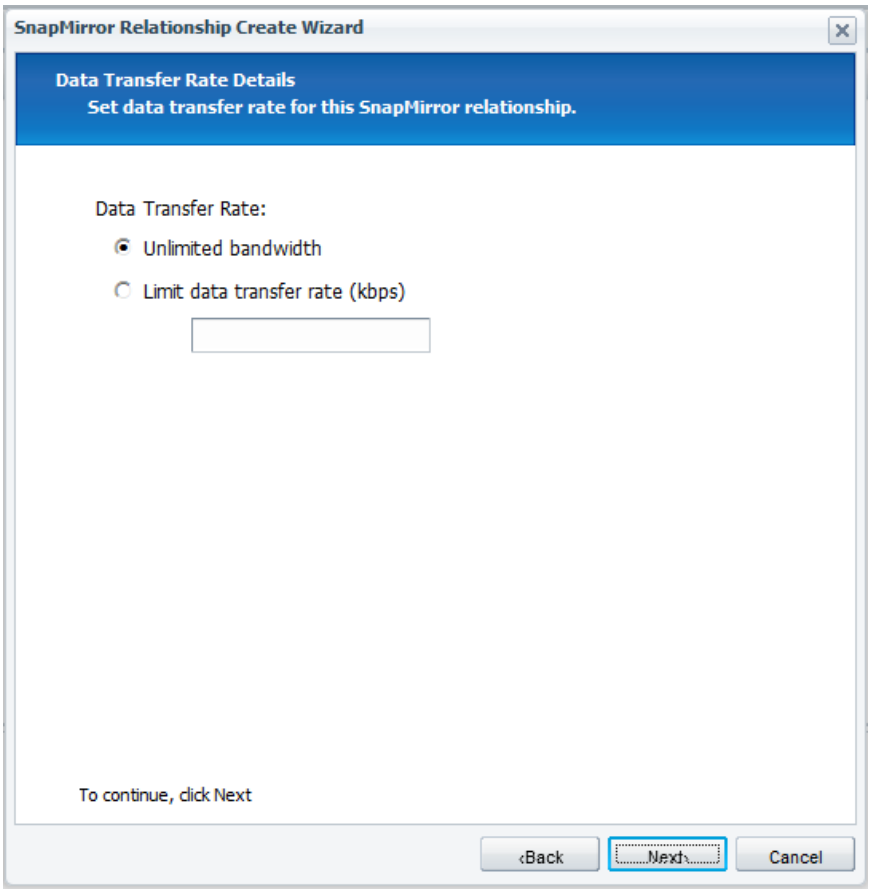
4. In the destination detail, select create new volume in the SQL64AGGR2 Aggregate and alwayson3_data volume.



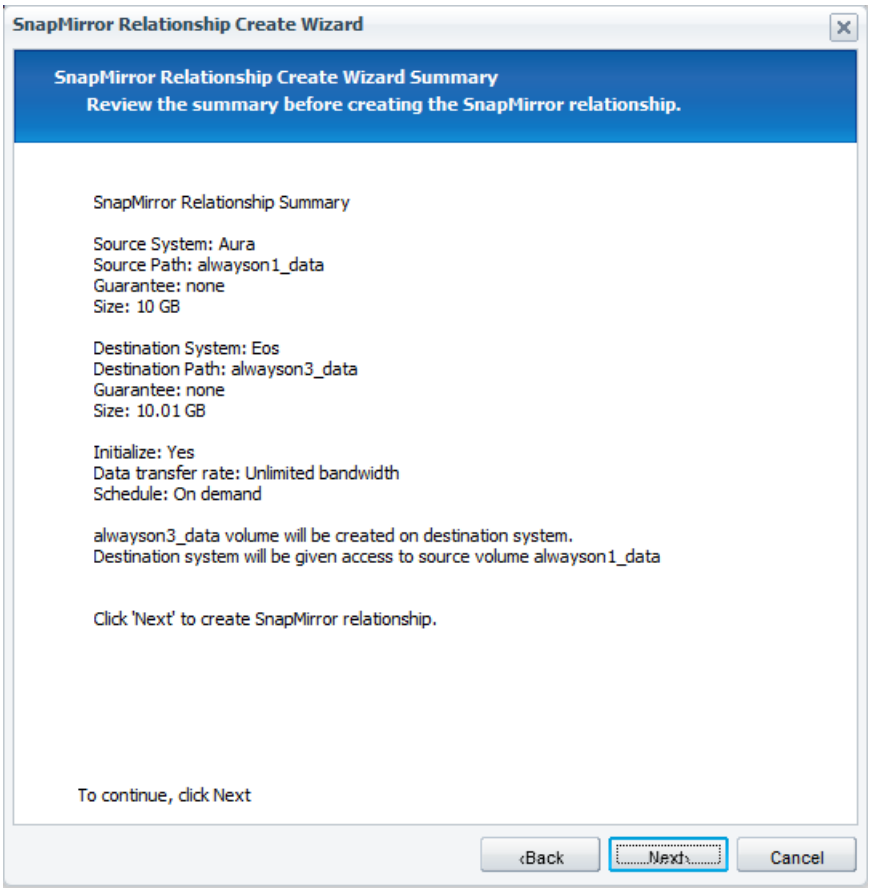
5. In Schedule and Initialize, select on demand since SMSQL will update SnapMirror after the backup has been created. Also select Initialize SnapMirror Relationship to start replication of the data from source to destination.



6. For this exercise, select unlimited bandwidth.

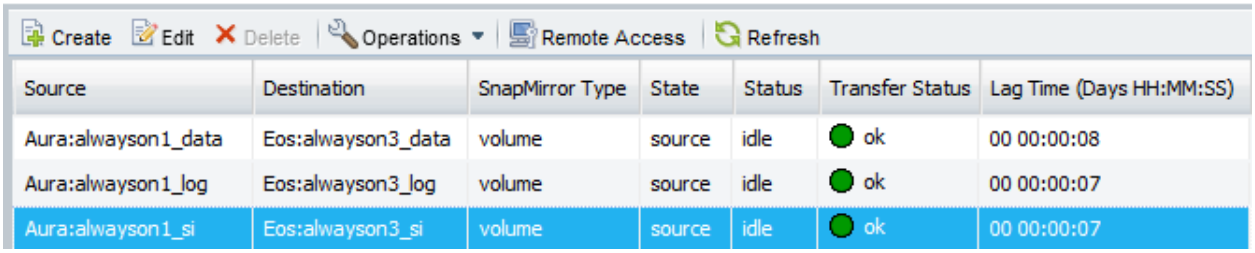


7. The SnapMirror relationship has now been created.



After the SnapMirror relationship for the database volume has been created, the SnapMirror relationship for the log and SnapInfo volumes needs to be created following the same processes outlined earlier. Further detail on SnapMirror configuration and best practices can be found in NetApp TR-4015.

Figure 12) SnapMirror relationship for all three volumes has been created, and all data has been transferred.



Source	Destination	SnapMirror Type	State	Status	Transfer Status	Lag Time (Days HH:MM:SS)
Aura:alwayson1_data	Eos:alwayson3_data	volume	source	idle	ok	00 00:00:08
Aura:alwayson1_log	Eos:alwayson3_log	volume	source	idle	ok	00 00:00:07
Aura:alwayson1_si	Eos:alwayson3_si	volume	source	idle	ok	00 00:00:07

In order to add the database to the AlwaysOn3 instance, the database must be backed up from the primary (AlwaysOn1) and restored to the remote replica (AlwaysOn3) instance with norecovery mode. SnapMirror also needs to be updated before the restore process. The backup and restore process can be performed with the following Windows PowerShell script:

```
powershell.exe -ExecutionPolicy Unrestricted -NoLogo -NonInteractive
if ((Get-Module| select -exp name) -notcontains 'SQLPS') {Import-Module SQLPS -
DisableNameChecking}
Add-PSSnapin NetApp.SnapManager.SQL.PS.Admin -ErrorAction silentlycontinue
#Note that we use –CloneOnMirrorDestination and –UpdateMirror switch
$node1 = “AlwaysOn1”
```

```
$node3 = “AlwaysOn3”
$database = “Adventureworks”

clone-database -Server $Node1 -ServerInstance $node1 -Database $database -TargetServerInstance
$node3 -TargetDatabase $Database -TargetServerMountPointDir $TargetMountPoint -RecoverDatabase
$false –lb –CloneOnMirrorDestination –UpdateMirror
```

The Adventureworks database in AlwaysOn3 is now ready to be added to the availability group by using the following Windows PowerShell script:

```
$agname = “AlwaysOn_AG”
$availabilityGroup = get-item SQLSERVER:\SQL\$node3\DEFAULT\AvailabilityGroups\$agName
Add-SqlAvailabilityDatabase -InputObject $availabilityGroup -Database $Database
```

Figure 13) SnapMirror has been created, and database in AlwaysOn3 instance has been added to availability groups.

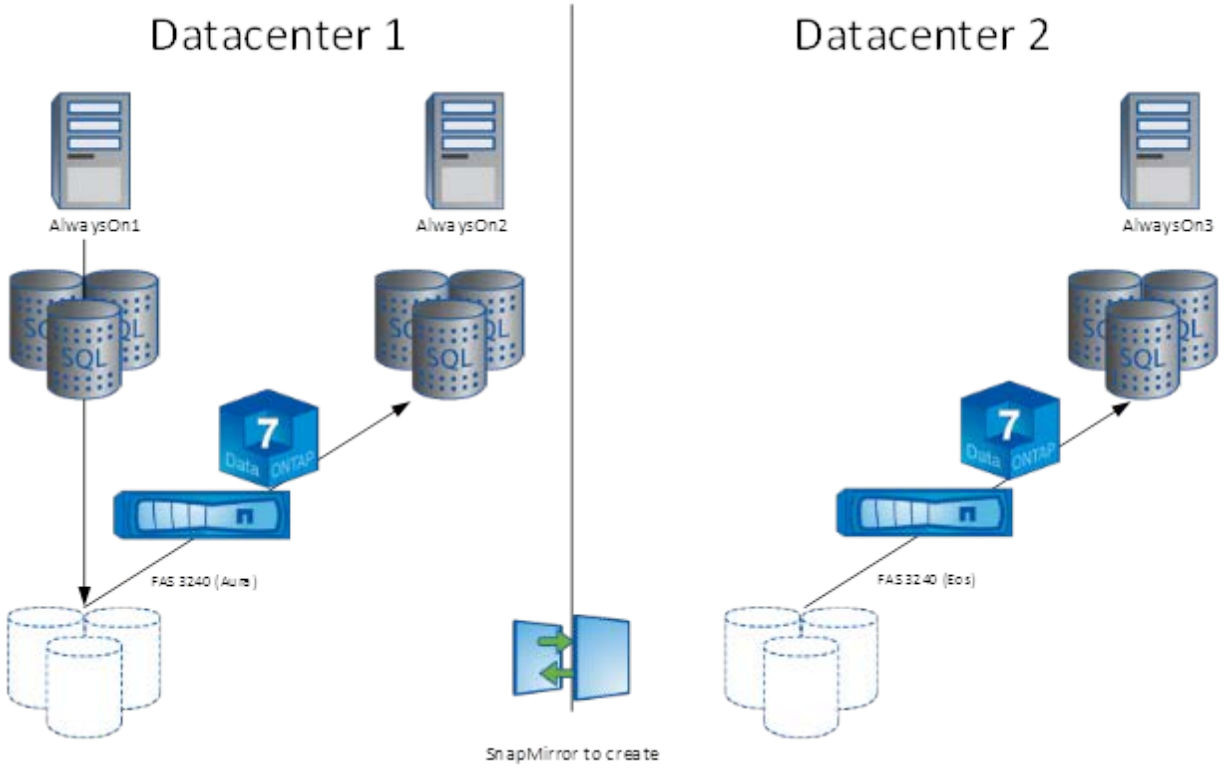
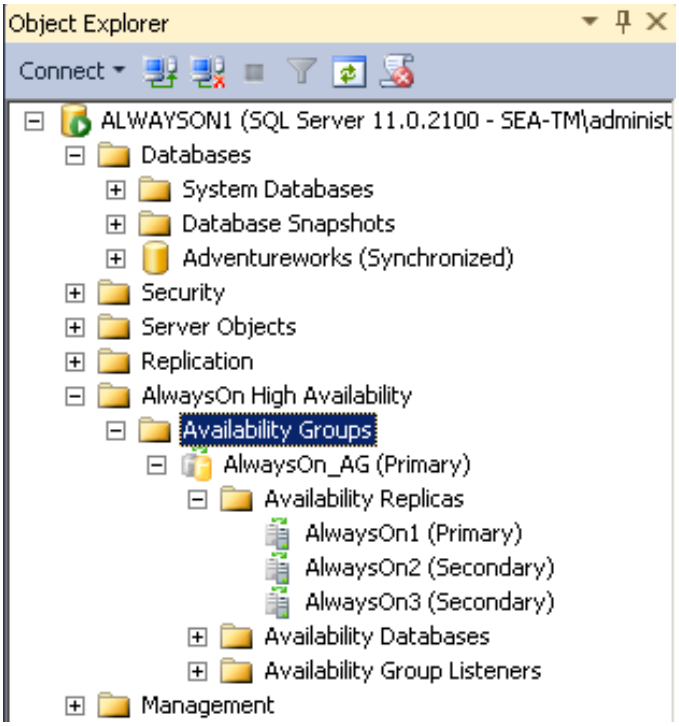


Figure 14) All nodes have been added to AlwaysOn_AG availability groups, and AlwaysOn1 is primary node.



After the database in AlwaysOn3 instance has been added to the availability group (under Availability Databases), the FlexClone copy of the database volume can now be split from the Snapshot copy. This will allow the volume to be independent from original volume and Snapshot copy.

Figure 15) The sdw_cl_alwayson3_data_0 has been identified as a clone of alwayson3_data volume.

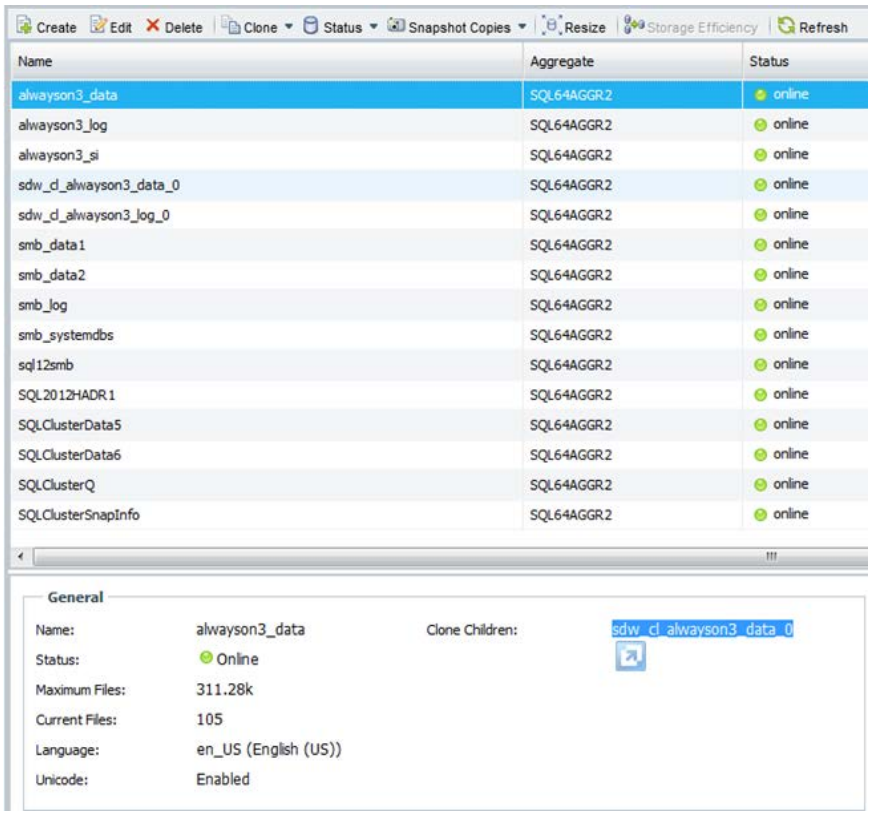
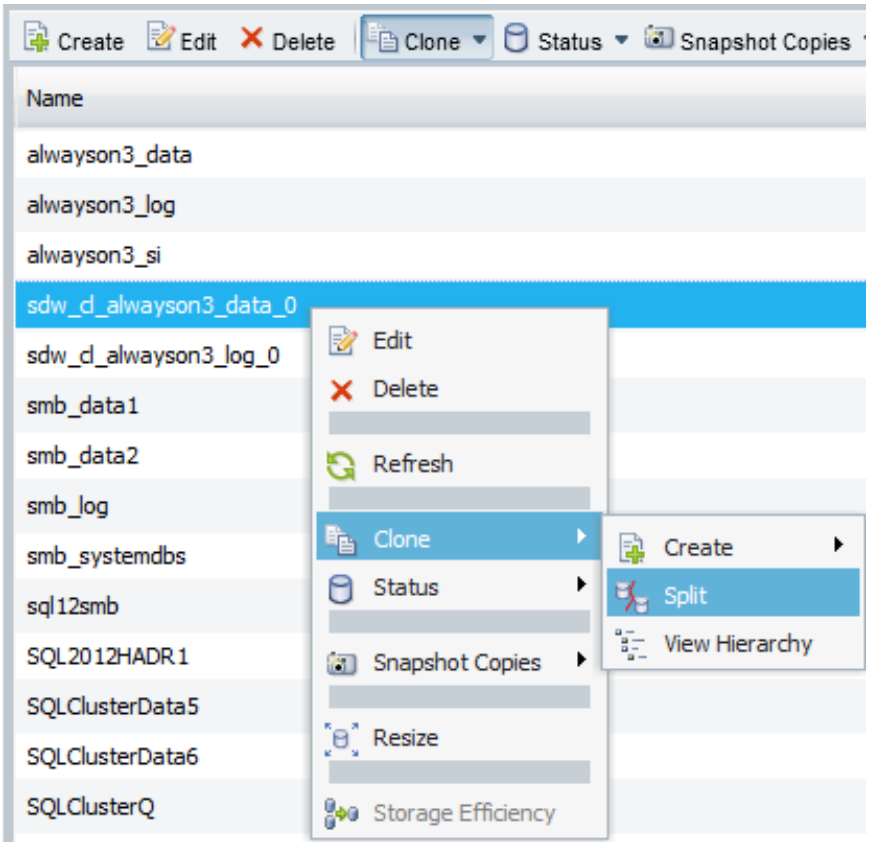


Figure 16) The sdw_cl_alwayon3_data_0 has been split through clone > split menu.



The FlexClone copy of the log volume can be split from the Snapshot copy using the same process as shown earlier.

8 Create Availability Groups Listeners

In order to have fast application failover access to SQL Server, availability group listeners must be created. The availability group listener must be created on the current primary node, and it can be done using the following Windows PowerShell command. The SQL Server access port and static IP address can be specified if needed.

```
$Listener = “SQL2012AlwaysOn”
$availabilityGroup = get-item SQLSERVER:\SQL\$node1\DEFAULT\AvailabilityGroups\$AgName
New-SqlAvailabilityGroupListener -Name $Listener -InputObject $availabilityGroup
```

9 Test Failover Cluster After Setup

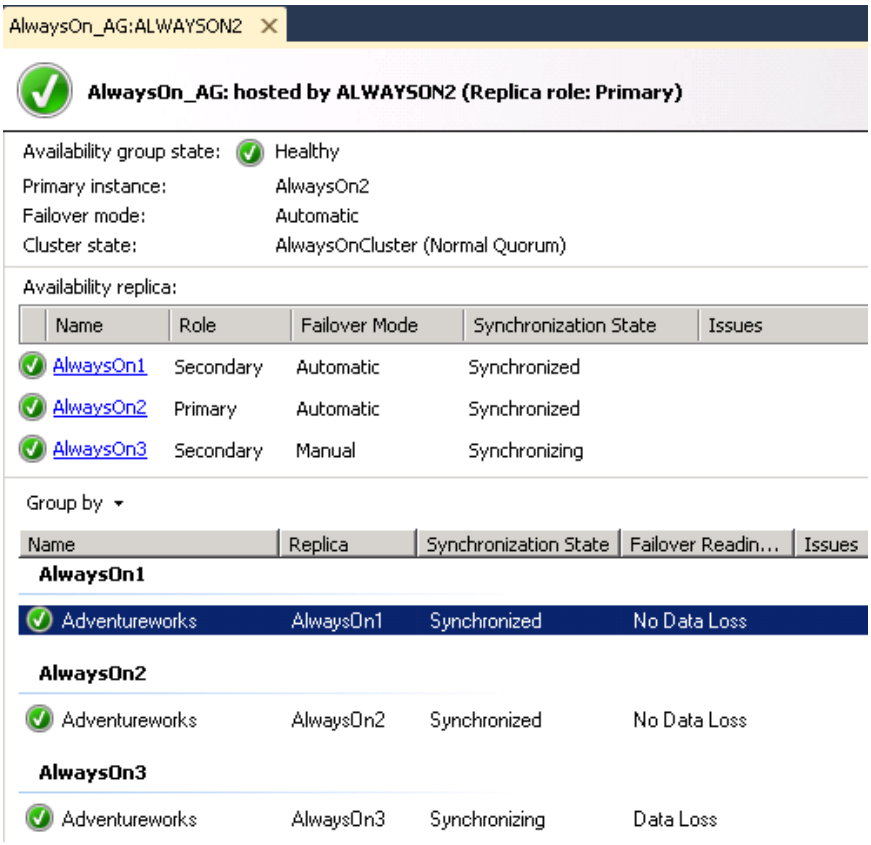
After the availability group has been set up, the failover can be tested using the following Windows PowerShell script:

```
powershell.exe -ExecutionPolicy Unrestricted -NoLogo -NonInteractive
if ((Get-Module| select -exp name) -notcontains 'SQLPS') {Import-Module SQLPS -
DisableNameChecking}
```

```
Set-location SQLServer:\SQL\AlwaysOn2\Default\AvailabilityGroups\AlwaysOn_AG
```

```
Switch-SqlAvailabilityGroup
```

Figure 17) Availability Groups dashboard display AlwaysOn2 has become primary role, and AlwaysOn1 and AlwaysOn3 are secondary roles.



10 Summary

The new SQL Server 2012 AlwaysOn Availability Groups feature allows businesses to maximize availability of critical user databases. However, setting up availability groups for large databases can be complicated and time consuming, especially if one of the nodes is in a remote data center or on a different subnet. NetApp storage simplifies this process compared to SQL Server built-in backup and restore capabilities by using SnapMirror and SMSQL instead. The processes of setting up AlwaysOn Availability can be accomplished by following these steps:

1. Enable Windows clustering feature on all nodes.
2. Create a failover cluster and add all nodes to the cluster.
3. Enable AlwaysOn High-Availability services on all nodes.
4. Restart SQL Server services for all instances.
5. Create an endpoint object for each replica.
6. Create initial full and log database backups using SMSQL.
7. Add the primary node and its database to the availability group.
8. Back up and restore database using SMSQL for the secondary replica, which is on the same storage controller.
9. Add the database on the secondary replica to the availability group.

10. Create a SnapMirror relationship between two controllers (Aura and Eos).
11. Back up and update SnapMirror and restore the database using SMSQL for the third replica on a separate storage controller.
12. Add the database on the third replica to the availability group.
13. Create availability group listeners.
14. Test failover.

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.

NetApp provides no representations or warranties regarding the accuracy, reliability, or serviceability of any information or recommendations provided in this publication, or with respect to any results that may be obtained by the use of the information or observance of any recommendations provided herein. The information in this document is distributed AS IS, and the use of this information or the implementation of any recommendations or techniques herein is a customer's responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. This document and the information contained herein may be used solely in connection with the NetApp products discussed in this document.

Go further, faster®

www.netapp.com