Manage SnapMirror volume replication

ONTAP 9

NetApp
January 13, 2022

Table of Contents

Manage SnapMirror volume replication

SnapMirror replication workflow

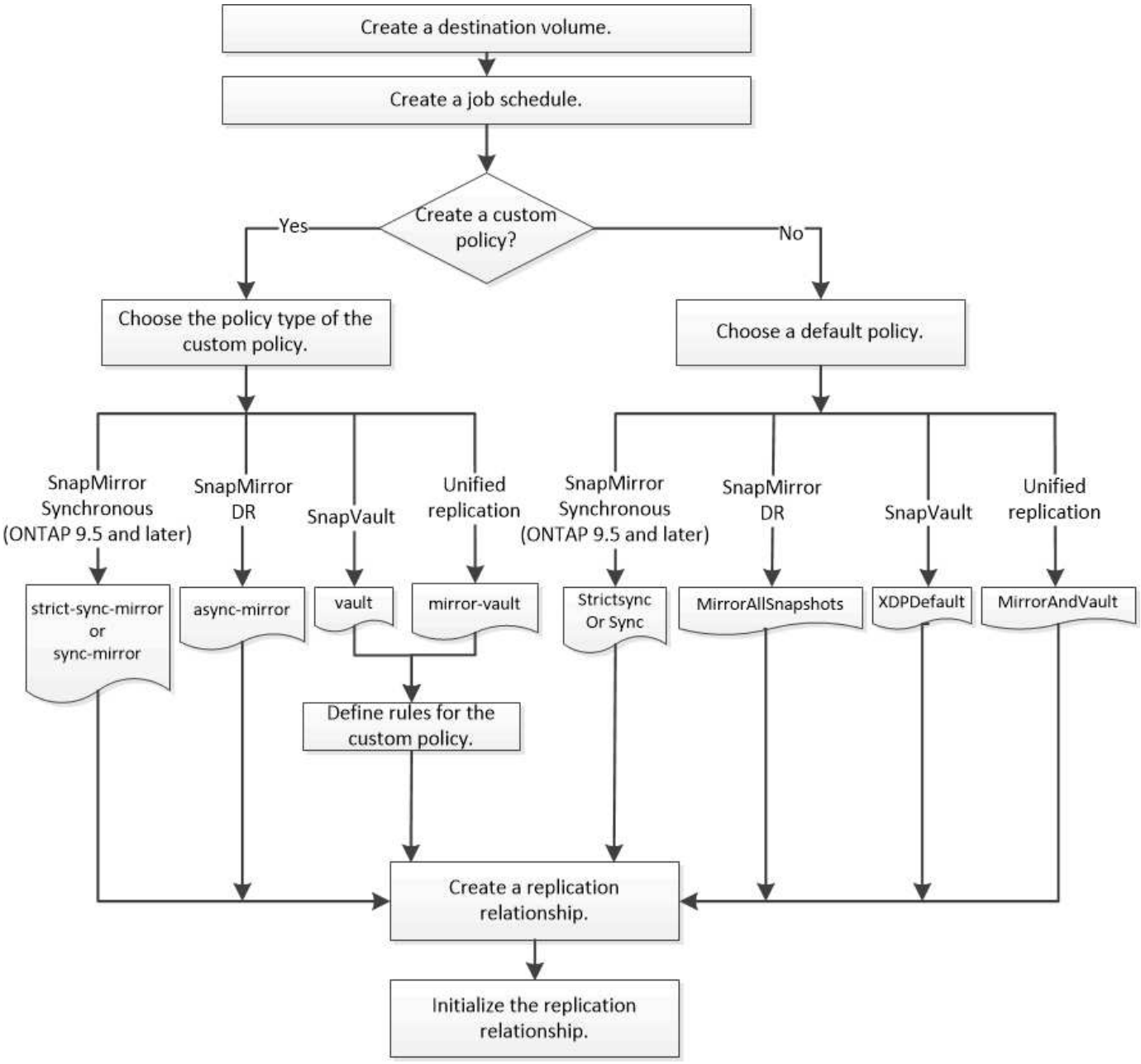
SnapMirror offers three types of data protection relationship: SnapMirror DR, archive (previously known as SnapVault), and unified replication. You can follow the same basic workflow to configure each type of relationship.

Beginning with general availability in ONTAP 9.9.1, SnapMirror Business Continuity (SM-BC) provides Zero Recovery Time Objective (Zero RTO) or Transparent Application Failover (TAF) to enable automatic failover of business-critical applications in SAN environments. SM-BC is supported in a configuration of either two AFF clusters or two All SAN Array (ASA) clusters.

NetApp Documentation: SnapMirror Business Continuity

For each type of SnapMirror data protection relationship, the workflow is the same: create a destination volume, create a job schedule, specify a policy, create and initialize the relationship.

Beginning with ONTAP 9.3, you can use the `snapmirror protect` command to configure a data protection relationship in a single step. Even if you use `snapmirror protect`, you need to understand each step in the workflow.



Configure a replication relationship in one step

Beginning with ONTAP 9.3, you can use the `snapmirror protect` command to configure a data protection relationship in a single step. You specify a list of volumes to be replicated, an SVM on the destination cluster, a job schedule, and a SnapMirror policy. `snapmirror protect` does the rest.

What you'll need

- The source and destination clusters and SVMs must be peered.

  Cluster and SVM peering

- The language on the destination volume must be the same as the language on the source volume.

About this task

The `snapmirror protect` command chooses an aggregate associated with the specified SVM. If no aggregate is associated with the SVM, it chooses from all the aggregates in the cluster. The choice of aggregate is based on the amount of free space and the number of volumes on the aggregate.

The `snapmirror protect` command then performs the following steps:

- Creates a destination volume with an appropriate type and amount of reserved space for each volume in the list of volumes to be replicated.
- Configures a replication relationship appropriate for the policy you specify.
- Initializes the relationship.

The name of the destination volume is of the form *`source_volume_name_dst`*. In case of a conflict with an existing name, the command appends a number to the volume name. You can specify a prefix and/or suffix in the command options. The suffix replaces the system-supplied `dst` suffix.

In ONTAP 9.3 and earlier, a destination volume can contain up to 251 Snapshot copies. In ONTAP 9.4 and later, a destination volume can contain up to 1019 Snapshot copies.

Initialization can be time-consuming. `snapmirror protect` does not wait for initialization to complete before the job finishes. For this reason, you should use the `snapmirror show` command rather than the `job show` command to determine when initialization is complete.

Beginning with ONTAP 9.5, SnapMirror Synchronous relationships can be created by using the `snapmirror protect` command.

Step

1. Create and initialize a replication relationship in one step:

   `snapmirror protect -path-list` *`SVM:volume|cluster://SVM/volume`*`, … -destination -vserver` *`destination_SVM`* `-policy` *`policy`* `-schedule` *`schedule`* `-auto-initialize true|false -destination-volume-prefix` *`prefix`* `-destination-volume-suffix` *`suffix`*

   You must run this command from the destination SVM or the destination cluster. The `-auto -initialize` option defaults to “true”.

   The following example creates and initializes a SnapMirror DR relationship using the default `MirrorAllSnapshots` policy:

   ```
   cluster_dst::> snapmirror protect -path-list svm1:volA, svm1:volB
   -destination-vserver svm_backup -policy MirrorAllSnapshots -schedule
   replication_daily
   ```

   You can use a custom policy if you prefer. For more information, see Creating a custom replication policy.

   The following example creates and initializes a SnapVault relationship using the default `XDPDefault` policy:

```
cluster_dst::> snapmirror protect -path-list svm1:volA, svm1:volB
-destination-vserver svm_backup -policy XDPDefault -schedule
replication_daily
```

The following example creates and initializes a unified replication relationship using the default `MirrorAndVault` policy:

```
cluster_dst::> snapmirror protect -path-list svm1:volA, svm1:volB
-destination-vserver svm_backup -policy MirrorAndVault
```

The following example creates and initializes a SnapMirror Synchronous relationship using the default `Sync` policy:

```
cluster_dst::> snapmirror protect -path-list svm1:volA, svm1:volB
-destination-vserver svm_sync -policy Sync
```

For SnapVault and unified replication policies, you might find it useful to define a schedule for creating a copy of the last transferred Snapshot copy on the destination. For more information, see Defining a schedule for creating a local copy on the destination.

After you finish

Use the `snapmirror show` command to verify that the SnapMirror relationship was created. For complete command syntax, see the man page.

Configure a replication relationship one step at a time

Create a destination volume

You can use the `volume create` command on the destination to create a destination volume. The destination volume should be the same or greater in size than the source volume.

Step

1. Create a destination volume:

   `volume create -vserver` *`SVM`* `-volume volume -aggregate` *`aggregate`* `-type DP -size` *`size`*

   For complete command syntax, see the man page.

   The following example creates a 2-GB destination volume named `volA_dst`:

```
cluster_dst::> volume create -vserver SVM_backup -volume volA_dst
-aggregate node01_aggr -type DP -size 2GB
```

Create a replication job schedule

You can use the `job schedule cron create` command to create a replication job schedule. The job schedule determines when SnapMirror automatically updates the data protection relationship to which the schedule is assigned.

About this task

You assign a job schedule when you create a data protection relationship. If you do not assign a job schedule, you must update the relationship manually.

Step

1. Create a job schedule:

   `job schedule cron create -name` *`job_name`* `-month` *`month`* `-dayofweek` *`day_of_week`* `-day` *`day_of_month`* `-hour` *`hour`* `-minute` *`minute`*

   For `-month`, `-dayofweek`, and `-hour`, you can specify `all` to run the job every month, day of the week, and hour, respectively.

   Beginning with ONTAP 9.10.1, you can include the Vserver for your job schedule:

   `job schedule cron create -name` *`job_name`* `-vserver` *`Vserver_name`* `-month` *`month`* `-dayofweek` *`day_of_week`* `-day` *`day_of_month`* `-hour` *`hour`* `-minute` *`minute`*

   The following example creates a job schedule named `my_weekly` that runs on Saturdays at 3:00 a.m.:

```
cluster_dst::> job schedule cron create -name my_weekly -dayofweek
"Saturday" -hour 3 -minute 0
```

Customize a replication policy

Create a custom replication policy

You can create a custom replication policy if the default policy for a relationship is not suitable. You might want to compress data in a network transfer, for example, or modify the number of attempts SnapMirror makes to transfer Snapshot copies.

You can use a default or custom policy when you create a replication relationship. For a custom archive (formerly SnapVault) or unified replication policy, you must define one or more *rules* that determine which Snapshot copies are transferred during initialization and update.You might also want to define a schedule for creating local Snapshot copies on the destination.

The *policy type* of the replication policy determines the type of relationship it supports. The table below shows the available policy types.

Policy type	Relationship type
async-mirror	SnapMirror DR
vault	SnapVault
mirror-vault	Unified replication
strict-sync-mirror	SnapMirror Synchronous in the StrictSync mode (supported beginning with ONTAP 9.5)
sync-mirror	SnapMirror Synchronous in the Sync mode (supported beginning with ONTAP 9.5)

When you create a custom replication policy, it is a good idea to model the policy after a default policy.

Step

1. Create a custom replication policy:

   `snapmirror policy create -vserver` *`SVM`* `-policy` *`policy`* `-type async-mirror|vault|mirror-vault|strict-sync-mirror|sync-mirror -comment` *`comment`* `-tries` *`transfer_tries`* `-transfer-priority low|normal -is-network-compression -enabled true|false`

   For complete command syntax, see the man page.

   Beginning with ONTAP 9.5, you can specify the schedule for creating a common Snapshot copy schedule for SnapMirror Synchronous relationships by using the `-common-snapshot-schedule` parameter. By default, the common Snapshot copy schedule for SnapMirror Synchronous relationships is one hour. You can specify a value from 30 minutes to two hours for the Snapshot copy schedule for SnapMirror Synchronous relationships.

   The following example creates a custom replication policy for SnapMirror DR that enables network compression for data transfers:

   ```
   cluster_dst::> snapmirror policy create -vserver svm1 -policy
   DR_compressed -type async-mirror -comment "DR with network compression
   enabled" -is-network-compression-enabled true
   ```

   The following example creates a custom replication policy for SnapVault:

   ```
   cluster_dst::> snapmirror policy create -vserver svm1 -policy
   my_snapvault -type vault
   ```

   The following example creates a custom replication policy for unified replication:

```
cluster_dst::> snapmirror policy create -vserver svm1 -policy my_unified
-type mirror-vault
```

The following example creates a custom replication policy for SnapMirror Synchronous relationship in the StrictSync mode:

```
cluster_dst::> snapmirror policy create -vserver svm1 -policy
my_strictsync -type strict-sync-mirror -common-snapshot-schedule
my_sync_schedule
```

After you finish

For “vault” and “mirror-vault” policy types, you must define rules that determine which Snapshot copies are transferred during initialization and update.

Use the `snapmirror policy show` command to verify that the SnapMirror policy was created. For complete command syntax, see the man page.

Define a rule for a policy

For custom policies with the “vault” or “mirror-vault” policy type, you must define at least one rule that determines which Snapshot copies are transferred during initialization and update. You can also define rules for default policies with the “vault” or"`mirror-vault`" policy type.

About this task

Every policy with the “vault” or “mirror-vault” policy type must have a rule that specifies which Snapshot copies to replicate. The rule “bi-monthly”, for example, indicates that only Snapshot copies assigned the SnapMirror label “bi-monthly” should be replicated. You specify the SnapMirror label when you configure the Snapshot policy on the source.

Each policy type is associated with one or more system-defined rules. These rules are automatically assigned to a policy when you specify its policy type. The table below shows the system-defined rules.

System-defined rule	Used in policy types	Result
sm_created	async-mirror, mirror-vault, Sync, StrictSync	A Snapshot copy created by SnapMirror is transferred on initialization and update.
all_source_snapshots	async-mirror	New Snapshot copies on the source are transferred on initialization and update.
daily	vault,mirror-vault	New Snapshot copies on the source with the SnapMirror label “daily” are transferred on initialization and update.

weekly	vault,mirror-vault	New Snapshot copies on the source with the SnapMirror label “weekly” are transferred on initialization and update.
monthly	mirror-vault	New Snapshot copies on the source with the SnapMirror label “monthly” are transferred on initialization and update.
app_consistent	Sync, StrictSync	Snapshot copies with the SnapMirror label “app_consistent” on source are synchronously replicated to the destination. Supported Beginning with ONTAP 9.7.

Except for the “async-mirror” policy type, you can specify additional rules as needed, for default or custom policies. For example:

- For the default `MirrorAndVault` policy, you might create a rule called “bi-monthly” to match Snapshot copies on the source with the “bi-monthly” SnapMirror label.
- For a custom policy with the “mirror-vault” policy type, you might create a rule called “bi-weekly” to match Snapshot copies on the source with the “bi-weekly” SnapMirror label.

Step

1. Define a rule for a policy:

   `snapmirror policy add-rule -vserver` *`SVM`* `-policy` *`policy_for_rule`* `-snapmirror -label` *`snapmirror-label`* `-keep` *`retention_count`*

   For complete command syntax, see the man page.

   The following example adds a rule with the SnapMirror label `bi-monthly` to the default `MirrorAndVault` policy:

   ```
   cluster_dst::> snapmirror policy add-rule -vserver svm1 -policy
   MirrorAndVault -snapmirror-label bi-monthly -keep 6
   ```

   The following example adds a rule with the SnapMirror label `bi-weekly` to the custom `my_snapvault` policy:

   ```
   cluster_dst::> snapmirror policy add-rule -vserver svm1 -policy
   my_snapvault -snapmirror-label bi-weekly -keep 26
   ```

   The following example adds a rule with the SnapMirror label `app_consistent` to the custom `Sync` policy:

```
cluster_dst::> snapmirror policy add-rule -vserver svm1 -policy Sync
-snapmirror-label app_consistent -keep 1
```

You can then replicate Snapshot copies from the source cluster that match this SnapMirror label:

```
cluster_src::> snapshot create -vserver vs1 -volume vol1 -snapshot
snapshot1 -snapmirror-label app_consistent
```

Define a schedule for creating a local copy on the destination

For SnapVault and unified replication relationships, you can protect against the possibility that an updated Snapshot copy is corrupted by creating a copy of the last transferred Snapshot copy on the destination. This “local copy” is retained regardless of the retention rules on the source, so that even if the Snapshot originally transferred by SnapMirror is no longer available on the source, a copy of it will be available on the destination.

About this task

You specify the schedule for creating a local copy in the `-schedule` option of the `snapmirror policy add-rule` command.

Step

1. Define a schedule for creating a local copy on the destination:

   `snapmirror policy add-rule -vserver` *`SVM`* `-policy` *`policy_for_rule`* `-snapmirror -label` *`snapmirror-label`* `-schedule` *`schedule`*

   For complete command syntax, see the man page. For an example of how to create a job schedule, see Creating a replication job schedule.

   The following example adds a schedule for creating a local copy to the default `MirrorAndVault` policy:

   ```
   cluster_dst::> snapmirror policy add-rule -vserver svm1 -policy
   MirrorAndVault -snapmirror-label my_monthly -schedule my_monthly
   ```

   The following example adds a schedule for creating a local copy to the custom `my_unified` policy:

   ```
   cluster_dst::> snapmirror policy add-rule -vserver svm1 -policy
   my_unified -snapmirror-label my_monthly -schedule my_monthly
   ```

Create a replication relationship

The relationship between the source volume in primary storage and the destination volume in secondary storage is called a *data protection relationship.* You can use the

`snapmirror create` command to create SnapMirror DR, SnapVault, or unified replication data protection relationships.

What you'll need

- The source and destination clusters and SVMs must be peered.

  Cluster and SVM peering

- The language on the destination volume must be the same as the language on the source volume.

About this task

Until ONTAP 9.3, SnapMirror invoked in DP mode and SnapMirror invoked in XDP mode used different replication engines, with different approaches to version-dependence:

- SnapMirror invoked in DP mode used a *version-dependent* replication engine in which the ONTAP version was required to be the same on primary and secondary storage:

```
cluster_dst::>  snapmirror create -type DP -source-path ... -destination
-path ...
```

- SnapMirror invoked in XDP mode used a *version-flexible* replication engine that supported different ONTAP versions on primary and secondary storage:

```
cluster_dst::>  snapmirror create -type XDP -source-path ...
-destination-path ...
```

With improvements in performance, the significant benefits of version-flexible SnapMirror outweigh the slight advantage in replication throughput obtained with version-dependent mode. For this reason, beginning with ONTAP 9.3, XDP mode has been made the new default, and any invocations of DP mode on the command line or in new or existing scripts are automatically converted to XDP mode.

Existing relationships are not affected. If a relationship is already of type DP, it will continue to be of type DP. The table below shows the behavior you can expect.

If you specify…	The type is…	The default policy (if you do not specify a policy) is…
DP	XDP	MirrorAllSnapshots (SnapMirror DR)
Nothing	XDP	MirrorAllSnapshots (SnapMirror DR)
XDP	XDP	XDPDefault (SnapVault)

See also the examples in the procedure below.

The only exceptions to conversion are as follows:

- SVM data protection relationships continue to default to DP mode.

Specify XDP explicitly to obtain XDP mode with the default `MirrorAllSnapshots` policy.

- Load-sharing data protection relationships continue to default to DP mode.
- SnapLock data protection relationships continue to default to DP mode.
- Explicit invocations of DP continue to default to DP mode if you set the following cluster-wide option:

```
options replication.create_data_protection_rels.enable on
```

This option is ignored if you do not explicitly invoke DP.

In ONTAP 9.3 and earlier, a destination volume can contain up to 251 Snapshot copies. In ONTAP 9.4 and later, a destination volume can contain up to 1019 Snapshot copies.

Beginning with ONTAP 9.5, SnapMirror Synchronous relationships are supported.

Step

1. From the destination cluster, create a replication relationship:

   `snapmirror create -source-path` *`SVM:volume|cluster://SVM/volume`*`, … -destination -path` *`SVM:volume|cluster://SVM/volume`*`, … -type DP|XDP -schedule` *`schedule`* `-policy` *`policy`*

   For complete command syntax, see the man page.

   The `schedule` parameter is not applicable when creating SnapMirror Synchronous relationships.

   The following example creates a SnapMirror DR relationship using the default `MirrorLatest` policy:

   ```
   cluster_dst::> snapmirror create -source-path svm1:volA -destination
   -path svm_backup:volA_dst -type XDP -schedule my_daily -policy
   MirrorLatest
   ```

   The following example creates a SnapVault relationship using the default `XDPDefault` policy:

   ```
   cluster_dst::> snapmirror create -source-path svm1:volA -destination
   -path svm_backup:volA_dst -type XDP -schedule my_daily -policy
   XDPDefault
   ```

   The following example creates a unified replication relationship using the default `MirrorAndVault` policy:

   ```
   cluster_dst:> snapmirror create -source-path svm1:volA -destination-path
   svm_backup:volA_dst -type XDP -schedule my_daily -policy MirrorAndVault
   ```

The following example creates a unified replication relationship using the custom `my_unified` policy:

```
cluster_dst::> snapmirror create -source-path svm1:volA -destination
-path svm_backup:volA_dst -type XDP -schedule my_daily -policy
my_unified
```

The following example creates a SnapMirror Synchronous relationship using the default `Sync` policy:

```
cluster_dst::> snapmirror create -source-path svm1:volA -destination
-path svm_backup:volA_dst -type XDP -policy Sync
```

The following example creates a SnapMirror Synchronous relationship using the default `StrictSync` policy:

```
cluster_dst::> snapmirror create -source-path svm1:volA -destination
-path svm_backup:volA_dst -type XDP -policy StrictSync
```

The following example creates a SnapMirror DR relationship. With the DP type automatically converted to XDP and with no policy specified, the policy defaults to the `MirrorAllSnapshots` policy:

```
cluster_dst::> snapmirror create -source-path svm1:volA -destination
-path svm_backup:volA_dst -type DP -schedule my_daily
```

The following example creates a SnapMirror DR relationship. With no type or policy specified, the policy defaults to the `MirrorAllSnapshots` policy:

```
cluster_dst::> snapmirror create -source-path svm1:volA -destination
-path svm_backup:volA_dst -schedule my_daily
```

The following example creates a SnapMirror DR relationship. With no policy specified, the policy defaults to the `XDPDefault` policy:

```
cluster_dst::> snapmirror create -source-path svm1:volA -destination
-path svm_backup:volA_dst -type XDP -schedule my_daily
```

The following example creates a SnapMirror Synchronous relationship with the predefined policy `SnapCenterSync`:

```
cluster_dst::> snapmirror create -source-path svm1:volA -destination
-path svm_backup:volA_dst -type XDP -policy SnapCenterSync
```

The predefined policy `SnapCenterSync` is of type `Sync`. This policy replicates any Snapshot copy that is created with the `snapmirror-label` of "app_consistent".

After you finish

Use the `snapmirror show` command to verify that the SnapMirror relationship was created. For complete command syntax, see the man page.

Initialize a replication relationship

For all relationship types, initialization performs a *baseline transfer*: it makes a Snapshot copy of the source volume, then transfers that copy and all the data blocks it references to the destination volume. Otherwise, the contents of the transfer depend on the policy.

What you'll need

The source and destination clusters and SVMs must be peered.

Cluster and SVM peering

About this task

Initialization can be time-consuming. You might want to run the baseline transfer in off-peak hours.

Beginning with ONTAP 9.5, SnapMirror Synchronous relationships are supported.

Step

1. Initialize a replication relationship:

   `snapmirror initialize -source-path` *`SVM:volume|cluster://SVM/volume`*`, …`
   `-destination-path` *`SVM:volume|cluster://SVM/volume`*`, …`

   For complete command syntax, see the man page.

   You must run this command from the destination SVM or the destination cluster.

   The following example initializes the relationship between the source volume `volA` on `svm1` and the destination volume `volA_dst` on `svm_backup`:

   ```
   cluster_dst::> snapmirror initialize -source-path svm1:volA -destination
   -path svm_backup:volA_dst
   ```

Example: Configure a vault-vault cascade

An example will show in concrete terms how you can configure replication relationships one step at a time. You can use the vault-vault cascade deployment configured in the example to retain more than 251 Snapshot copies labeled “my-weekly”.

What you'll need

- The source and destination clusters and SVMs must be peered.

- You must be running ONTAP 9.2 or later. Vault-vault cascades are not supported in earlier ONTAP releases.

About this task

The example assumes the following:

- You have configured Snapshot copies on the source cluster with the SnapMirror labels “my-daily”, “my-weekly”, and “my-monthly”.
- You have configured destination volumes named “volA” on the secondary and tertiary destination clusters.
- You have configured replication job schedules named “my_snapvault” on the secondary and tertiary destination clusters.

The example shows how to create replication relationships based on two custom policies:

- The “snapvault_secondary” policy retains 7 daily, 52 weekly, and 180 monthly Snapshot copies on the secondary destination cluster.
- The “snapvault_tertiary policy” retains 250 weekly Snapshot copies on the tertiary destination cluster.

Steps

1. On the secondary destination cluster, create the “snapvault_secondary” policy:

   ```
   cluster_secondary::> snapmirror policy create -policy snapvault_secondary
   -type vault -comment “Policy on secondary for vault to vault cascade” -vserver
   svm_secondary
   ```

2. On the secondary destination cluster, define the “my-daily” rule for the policy:

   ```
   cluster_secondary::> snapmirror policy add-rule -policy snapvault_secondary
   -snapmirror-label my-daily -keep 7 -vserver svm_secondary
   ```

3. On the secondary destination cluster, define the “my-weekly” rule for the policy:

   ```
   cluster_secondary::> snapmirror policy add-rule -policy snapvault_secondary
   -snapmirror-label my-weekly -keep 52 -vserver svm_secondary
   ```

4. On the secondary destination cluster, define the “my-monthly” rule for the policy:

   ```
   cluster_secondary::> snapmirror policy add-rule -policy snapvault_secondary
   -snapmirror-label my-monthly -keep 180 -vserver svm_secondary
   ```

5. On the secondary destination cluster, verify the policy:

   ```
   cluster_secondary::> snapmirror policy show snapvault_secondary -instance
   ```

```
                    Vserver: svm_secondary
      SnapMirror Policy Name: snapvault_secondary
      SnapMirror Policy Type: vault
                Policy Owner: cluster-admin
                 Tries Limit: 8
           Transfer Priority: normal
   Ignore accesstime Enabled: false
     Transfer Restartability: always
 Network Compression Enabled: false
             Create Snapshot: false
                     Comment: Policy on secondary for vault to vault
cascade
       Total Number of Rules: 3
                  Total Keep: 239
                       Rules: SnapMirror Label     Keep  Preserve Warn
Schedule Prefix
                              ----------------     ----  -------- ----
-------- ------
                              my-daily                7  false       0 -
-
                              my-weekly              52  false       0 -
-
                              my-monthly            180  false       0 -
-
```

6. On the secondary destination cluster, create the relationship with the source cluster:

```
cluster_secondary::> snapmirror create -source-path svm_primary:volA
-destination-path svm_secondary:volA -type XDP -schedule my_snapvault -policy
snapvault_secondary
```

7. On the secondary destination cluster, initialize the relationship with the source cluster:

```
cluster_secondary::> snapmirror initialize -source-path svm_primary:volA
-destination-path svm_secondary:volA
```

8. On the tertiary destination cluster, create the “snapvault_tertiary” policy:

```
cluster_tertiary::> snapmirror policy create -policy snapvault_tertiary -type
vault -comment "Policy on tertiary for vault to vault cascade" -vserver
svm_tertiary
```

9. On the tertiary destination cluster, define the “my-weekly” rule for the policy:

```
cluster_tertiary::> snapmirror policy add-rule -policy snapvault_tertiary
-snapmirror-label my-weekly -keep 250 -vserver svm_tertiary
```

10. On the tertiary destination cluster, verify the policy:

```
cluster_tertiary::> snapmirror policy show snapvault_tertiary -instance
```

```
                    Vserver: svm_tertiary
      SnapMirror Policy Name: snapvault_tertiary
      SnapMirror Policy Type: vault
                Policy Owner: cluster-admin
                 Tries Limit: 8
           Transfer Priority: normal
   Ignore accesstime Enabled: false
     Transfer Restartability: always
 Network Compression Enabled: false
             Create Snapshot: false
                     Comment: Policy on tertiary for vault to vault
cascade
       Total Number of Rules: 1
                  Total Keep: 250
                       Rules: SnapMirror Label     Keep  Preserve Warn
Schedule Prefix
                              ----------------     ----  -------- ----
-------- ------
                              my-weekly             250  false       0 -
-
```

11. On the tertiary destination cluster, create the relationship with the secondary cluster:

```
cluster_tertiary::> snapmirror create -source-path svm_secondary:volA
-destination-path svm_tertiary:volA -type XDP -schedule my_snapvault -policy
snapvault_tertiary
```

12. On the tertiary destination cluster, initialize the relationship with the secondary cluster:

```
cluster_tertiary::> snapmirror initialize -source-path svm_secondary:volA
-destination-path svm_tertiary:volA
```

Convert an existing DP-type relationship to XDP

You can easily convert an existing DP-type relationship to XDP to take advantage of version-flexible SnapMirror.

About this task

SnapMirror does not automatically convert existing DP-type relationships to XDP. To convert the relationship, you need to break and delete the existing relationship, create a new XDP relationship, and resync the relationship. For background information, see XDP replaces DP as the SnapMirror default.

After you convert a SnapMirror relationship type from DP to XDP, space-related settings, such as autosize and space guarantee are no longer replicated to the destination.

Steps

1. Quiesce the existing DP-type relationship:

   `snapmirror quiesce -source-path` *`SVM:volume|cluster://SVM/volume`*`, …`
   `-destination-path` *`SVM:volume|cluster://SVM/volume`*`, …`

   For complete command syntax, see the man page.

   You must run this command from the destination SVM or the destination cluster.

   The following example quiesces the relationship between the source volume `volA` on `svm1` and the destination volume `volA_dst` on `svm_backup`:

   ```
   cluster_dst::> snapmirror quiesce -source-path svm1:volA -destination
   -path svm_backup:volA_dst
   ```

2. Break the existing DP-type relationship:

   `snapmirror break -source-path` *`SVM:volume|cluster://SVM/volume`*`, … -destination`
   `-path` *`SVM:volume|cluster://SVM/volume`*`, …`

   For complete command syntax, see the man page.

   You must run this command from the destination SVM or the destination cluster.

   The following example breaks the relationship between the source volume `volA` on `svm1` and the destination volume `volA_dst` on `svm_backup`:

   ```
   cluster_dst::> snapmirror break -source-path svm1:volA -destination-path
   svm_backup:volA_dst
   ```

3. Disable automatic deletion of Snapshot copies on the destination volume:

   `volume snapshot autodelete modify -vserver` *`SVM`* `-volume` *`volume`* `-enabled false`

   The following example disables Snapshot copy autodelete on the destination volume `volA_dst`:

   ```
   cluster_dst::> volume snapshot autodelete modify -vserver svm_backup
   -volume volA_dst -enabled false
   ```

4. Delete the existing DP-type relationship:

   `snapmirror delete -source-path` *`SVM:volume|cluster://SVM/volume`*`, … -destination`
   `-path` *`SVM:volume|cluster://SVM/volume`*`, …`

   For complete command syntax, see the man page.

You must run this command from the destination SVM or the destination cluster.

The following example deletes the relationship between the source volume `volA` on `svm1` and the destination volume `volA_dst` on `svm_backup`:

```
cluster_dst::> snapmirror delete -source-path svm1:volA -destination
-path svm_backup:volA_dst
```

5. Create the new XDP-type relationship:

   `snapmirror create -source-path` *`SVM:volume|cluster://SVM/volume`*`, … -destination -path` *`SVM:volume|cluster://SVM/volume`*`, … -type XDP -schedule` *`schedule`* `-policy` *`policy`*

   The new relationship must use the same source and destination volume. For complete command syntax, see the man page.

   You must run this command from the destination SVM or the destination cluster.

   The following example creates a SnapMirror DR relationship between the source volume `volA` on `svm1` and the destination volume `volA_dst` on `svm_backup` using the default `MirrorAllSnapshots` policy:

   ```
   cluster_dst::> snapmirror create -source-path svm1:volA -destination
   -path svm_backup:volA_dst
   -type XDP -schedule my_daily -policy MirrorAllSnapshots
   ```

6. Resync the source and destination volumes:

   `snapmirror resync -source-path` *`SVM:volume|cluster://SVM/volume`*`, … -destination -path` *`SVM:volume|cluster://SVM/volume`*`, …`

   For complete command syntax, see the man page.

   You must run this command from the destination SVM or the destination cluster. Although resync does not require a baseline transfer, it can be time-consuming. You might want to run the resync in off-peak hours.

   The following example resyncs the relationship between the source volume `volA` on `svm1` and the destination volume `volA_dst` on `svm_backup`:

   ```
   cluster_dst::> snapmirror resync -source-path svm1:volA -destination
   -path svm_backup:volA_dst
   ```

After you finish

Use the `snapmirror show` command to verify that the SnapMirror relationship was created. For complete

command syntax, see the man page.

Convert the type of a SnapMirror relationship

Beginning with ONTAP 9.5, SnapMirror Synchronous is supported. You can convert an asynchronous SnapMirror relationship to a SnapMirror Synchronous relationship or vice versa without performing a baseline transfer.

About this task

You cannot convert an asynchronous SnapMirror relationship to a SnapMirror Synchronous relationship or vice versa by changing the SnapMirror policy

Steps

- **Converting an asynchronous SnapMirror relationship to a SnapMirror Synchronous relationship**
  a. From the destination cluster, delete the asynchronous SnapMirror relationship:

     `snapmirror delete -destination-path` *`SVM:volume`*

     ```
     cluster2::>snapmirror delete -destination-path vs1_dr:vol1
     ```

  b. From the source cluster, release the SnapMirror relationship without deleting the common Snapshot copies:

     `snapmirror release -relationship-info-only true -destination-path` *`dest_SVM:dest_volume`*

     ```
     cluster1::>snapmirror release -relationship-info-only true
     -destination-path vs1_dr:vol1
     ```

  c. From the destination cluster, create a SnapMirror Synchronous relationship:

     `snapmirror create -source-path` *`src_SVM:src_volume`* `-destination-path` *`dest_SVM:dest_volume`* `-policy sync-mirror`

     ```
     cluster2::>snapmirror create -source-path vs1:vol1 -destination-path
     vs1_dr:vol1 -policy sync
     ```

  d. Resynchronize the SnapMirror Synchronous relationship:

     `snapmirror resync -destination-path` *`dest_SVM:dest_volume`*

     ```
     cluster2::>snapmirror resync -destination-path vs1_dr:vol1
     ```

- **Converting a SnapMirror Synchronous relationship to an asynchronous SnapMirror relationship**

a. From the destination cluster, quiesce the existing SnapMirror Synchronous relationship:

`snapmirror quiesce -destination-path dest_SVM:dest_volume`

```
cluster2::> snapmirror quiesce -destination-path vs1_dr:vol1
```

b. From the destination cluster, delete the asynchronous SnapMirror relationship:

`snapmirror delete -destination-path` *`SVM:volume`*

```
cluster2::>snapmirror delete -destination-path vs1_dr:vol1
```

c. From the source cluster, release the SnapMirror relationship without deleting the common Snapshot copies:

`snapmirror release -relationship-info-only true -destination-path` *`dest_SVM:dest_volume`*

```
cluster1::>snapmirror release -relationship-info-only true
-destination-path vs1_dr:vol1
```

d. From the destination cluster, create an asynchronous SnapMirror relationship:

`snapmirror create -source-path` *`src_SVM:src_volume`* `-destination-path` *`dest_SVM:dest_volume`* `-policy MirrorAllSnapshots`

```
cluster2::>snapmirror create -source-path vs1:vol1 -destination-path
vs1_dr:vol1 -policy sync
```

e. Resynchronize the SnapMirror Synchronous relationship:

`snapmirror resync -destination-path` *`dest_SVM:dest_volume`*

```
cluster2::>snapmirror resync -destination-path vs1_dr:vol1
```

Convert the mode of a SnapMirror Synchronous relationship

Beginning with ONTAP 9.5, SnapMirror Synchronous relationships are supported. You can convert the mode of a SnapMirror Synchronous relationship from StrictSync to Sync or vice versa.

About this task

You cannot modify the policy of a Snapmirror Synchronous relationship to convert its mode.

Steps

1. From the destination cluster, quiesce the existing SnapMirror Synchronous relationship:

   `snapmirror quiesce -destination-path dest_SVM:dest_volume`

   ```
   cluster2::> snapmirror quiesce -destination-path vs1_dr:vol1
   ```

2. From the destination cluster, delete the existing SnapMirror Synchronous relationship:

   `snapmirror delete -destination-path dest_SVM:dest_volume`

   ```
   cluster2::> snapmirror delete -destination-path vs1_dr:vol1
   ```

3. From the source cluster, release the SnapMirror relationship without deleting the common Snapshot copies:

   `snapmirror release -relationship-info-only true -destination-path dest_SVM:dest_volume`

   ```
   cluster1::> snapmirror release -relationship-info-only true -destination
   -path vs1_dr:vol1
   ```

4. From the destination cluster, create a SnapMirror Synchronous relationship by specifying the mode to which you want to convert the SnapMirror Synchronous relationship:

   `snapmirror create -source-path vs1:vol1 -destination-path dest_SVM:dest_volume -policy Sync|StrictSync`

   ```
   cluster2::> snapmirror create -source-path vs1:vol1 -destination-path
   vs1_dr:vol1 -policy Sync
   ```

5. From the destination cluster, resynchronize the SnapMirror relationship:

   `snapmirror resync -destination-path dest_SVM:dest_volume`

   ```
   cluster2::> snapmirror resync -destination-path vs1_dr:vol1
   ```

Serve data from a SnapMirror DR destination volume

Make the destination volume writeable

You need to make the destination volume writeable before you can serve data from the volume to clients. You can use the `snapmirror quiesce` command to stop scheduled transfers to the destination, the `snapmirror abort` command to stop ongoing transfers, and the `snapmirror break` command to make the destination writeable.

About this task

You must perform this task from the destination SVM or the destination cluster.

Steps

1. Stop scheduled transfers to the destination:

   `snapmirror quiesce -source-path` *`SVM:volume|cluster://SVM/volume`*`, …`
   `-destination-path` *`SVM:volume|cluster://SVM/volume`*`, …`

   For complete command syntax, see the man page.

   The following example stops scheduled transfers between the source volume `volA` on `svm1` and the destination volume `volA_dst` on `svm_backup`:

   ```
   cluster_dst::> snapmirror quiesce -source-path svm1:volA -destination
   -path svm_backup:volA_dst
   ```

2. Stop ongoing transfers to the destination:

   `snapmirror abort -source-path` *`SVM:volume|cluster://SVM/volume`*`, … -destination`
   `-path` *`SVM:volume|cluster://SVM/volume`*`, …`

   For complete command syntax, see the man page.

   > This step is not required for SnapMirror Synchronous relationships (supported beginning with ONTAP 9.5).

   The following example stops ongoing transfers between the source volume `volA` on `svm1` and the destination volume `volA_dst` on `svm_backup`:

   ```
   cluster_dst::> snapmirror abort -source-path svm1:volA -destination-path
   svm_backup:volA_dst
   ```

3. Break the SnapMirror DR relationship:

   `snapmirror break -source-path` *`SVM:volume|cluster://SVM/volume`*`, … -destination`
   `-path` *`SVM:volume|cluster://SVM/volume`*`, …`

   For complete command syntax, see the man page.

   The following example breaks the relationship between the source volume `volA` on `svm1` and the

destination volume `volA_dst` on `svm_backup` and the destination volume `volA_dst` on `svm_backup`:

```
cluster_dst::> snapmirror break -source-path svm1:volA -destination-path
svm_backup:volA_dst
```

Configure the destination volume for data access

After making the destination volume writeable, you must configure the volume for data access. NAS clients, NVMe subsystem, andSAN hosts can access the data from the destination volume until the source volume is reactivated.

NAS environment:

1. Mount the NAS volume to the namespace using the same junction path that the source volume was mounted to in the source SVM.
2. Apply the appropriate ACLs to the CIFS shares at the destination volume.
3. Assign the NFS export policies to the destination volume.
4. Apply the quota rules to the destination volume.
5. Redirect clients to the destination volume.
6. Remount the NFS and CIFS shares on the clients.

SAN environment:

1. Map the LUNs in the volume to the appropriate initiator group.
2. For iSCSI, create iSCSI sessions from the SAN host initiators to the SAN LIFs.
3. On the SAN client, perform a storage re-scan to detect the connected LUNs.

For information about NVMe environment, see SAN administration.

Reactivate the original source volume

You can reestablish the original data protection relationship between the source and destination volumes when you no longer need to serve data from the destination.

About this task

The procedure below assumes that the baseline in the original source volume is intact. If the baseline is not intact, you must create and initialize the relationship between the volume you are serving data from and the original source volume before performing the procedure.

Steps

1. Delete the original data protection relationship:

   `snapmirror delete -source-path` *`SVM:volume|cluster://SVM/volume`*`, … -destination -path` *`SVM:volume|cluster://SVM/volume`*`, …`

   For complete command syntax, see the man page.

You must run this command from the destination SVM or the destination cluster.

The following example deletes the relationship between the original source volume, `volA` on `svm1`, and the volume you are serving data from, `volA_dst` on `svm_backup`:

```
cluster_dst::> snapmirror delete -source-path svm1:volA -destination
-path svm_backup:volA_dst
```

2. Reverse the original data protection relationship:

   `snapmirror resync -source-path` *`SVM:volume|cluster://SVM/volume`*`, … -destination -path` *`SVM:volume|cluster://SVM/volume`*`, …`

   For complete command syntax, see the man page.

   You must run this command from the destination SVM or the destination cluster. Although resync does not require a baseline transfer, it can be time-consuming. You might want to run the resync in off-peak hours.

   The following example reverses the relationship between the original source volume, `volA` on `svm1`, and the volume you are serving data from, `volA_dst` on `svm_backup`:

   ```
   cluster_dst::> snapmirror resync -source-path svm_backup:volA_dst
   -destination-path svm1:volA
   ```

3. Stop the source SVM for the reversed relationship:

   `vserver stop -vserver` *`SVM`*

   For complete command syntax, see the man page.

   The following example stops the source SVM for the reversed relationship:

   ```
   cluster_src::> vserver stop svm_backup
   ```

4. Update the reversed relationship:

   `snapmirror update -source-path` *`SVM:volume|cluster://SVM/volume`*`, … -destination -path` *`SVM:volume|cluster://SVM/volume`*`, …`

   For complete command syntax, see the man page.

   > You must run this command from the destination SVM or the destination cluster.The command fails if a common Snapshot copy does not exist on the source and destination. Use `snapmirror initialize` to re-initialize the relationship.

   The following example updates the relationship between the volume you are serving data from, `volA_dst` on `svm_backup`, and the original source volume, `volA` on `svm1`:

```
cluster_dst::> snapmirror update -source-path svm_backup:volA_dst
-destination-path svm1:volA
```

5. Stop scheduled transfers for the reversed relationship:

   `snapmirror quiesce -source-path` *`SVM:volume|cluster://SVM/volume`*`, …`
   `-destination-path` *`SVM:volume|cluster://SVM/volume`*`, …`

   For complete command syntax, see the man page.

   > You must run this command from the destination SVM or the destination cluster.

   The following example stops scheduled transfers between the volume you are serving data from, `volA_dst` on `svm_backup`, and the original source volume, `volA` on `svm1`:

```
cluster_dst::> snapmirror quiesce -source-path svm_backup:volA_dst
-destination-path svm1:volA
```

6. Stop ongoing transfers for the reversed relationship:

   `snapmirror abort -source-path` *`SVM:volume|cluster://SVM/volume`*`, … -destination`
   `-path` *`SVM:volume|cluster://SVM/volume`*`, …`

   For complete command syntax, see the man page.

   > You must run this command from the destination SVM or the destination cluster.

   The following example stops ongoing transfers between the volume you are serving data from, `volA_dst` on `svm_backup`, and the original source volume, `volA` on `svm1`:

```
cluster_dst::> snapmirror abort -source-path svm_backup:volA_dst
-destination-path svm1:volA
```

7. Break the reversed relationship:

   `snapmirror break -source-path` *`SVM:volume|cluster://SVM/volume`*`, … -destination`
   `-path` *`SVM:volume|cluster://SVM/volume`*`, …`

   For complete command syntax, see the man page.

   > You must run this command from the destination SVM or the destination cluster.

   The following example breaks the relationship between the volume you are serving data from, `volA_dst` on `svm_backup`, and the original source volume, `volA` on `svm1`:

```
cluster_dst::> snapmirror break -source-path svm_backup:volA_dst
-destination-path svm1:volA
```

8. Start the original source SVM:

   `vserver start -vserver` *SVM*

   For complete command syntax, see the man page.

   The following example starts the original source SVM:

   ```
   cluster_dst::> vserver start svm1
   ```

9. Delete the reversed data protection relationship:

   `snapmirror delete -source-path` *SVM:volume|cluster://SVM/volume*`, … -destination -path` *SVM:volume|cluster://SVM/volume*`, …`

   For complete command syntax, see the man page.

   You must run this command from the source SVM or the source cluster for the reversed relationship.

   The following example deletes the reversed relationship between the original source volume, `volA` on `svm1`, and the volume you are serving data from, `volA_dst` on `svm_backup`:

   ```
   cluster_src::> snapmirror delete -source-path svm_backup:volA_dst
   -destination-path svm1:volA
   ```

10. Reestablish the original data protection relationship:

    `snapmirror resync -source-path` *SVM:volume|cluster://SVM/volume*`, … -destination -path` *SVM:volume|cluster://SVM/volume*`, …`

    For complete command syntax, see the man page.

    The following example reestablishes the relationship between the original source volume, `volA` on `svm1`, and the original destination volume, `volA_dst` on `svm_backup`:

    ```
    cluster_dst::> snapmirror resync -source-path svm1:volA -destination
    -path svm_backup:volA_dst
    ```

After you finish

Use the `snapmirror show` command to verify that the SnapMirror relationship was created. For complete command syntax, see the man page.

Restore files from a SnapMirror destination volume

Restore a single file, LUN, or NVMe namespace from a SnapMirror destination

You can restore a single file, LUN, a set of files or LUNs from a Snapshot copy, or an NVMe namespace from a SnapMirror destination volume. Beginning with ONTAP 9.7, you can also restore NVMe namespaces from a SnapMirror Synchronous destination. You can restore files to the original source volume or to a different volume.

What you'll need

To restore a file or LUN from a SnapMirror Synchronous destination (supported beginning with ONTAP 9.5), you must first delete and release the relationship.

About this task

The volume to which you are restoring files or LUNs (the destination volume) must be a read-write volume:

- SnapMirror performs an *incremental restore* if the source and destination volumes have a common Snapshot copy (as is typically the case when you are restoring to the original source volume).
- Otherwise, SnapMirror performs a *baseline restore*, in which the specified Snapshot copy and all the data blocks it references are transferred to the destination volume.

Steps

1. List the Snapshot copies in the destination volume:

   `volume snapshot show -vserver` *`SVM`* `-volume volume`

   For complete command syntax, see the man page.

   The following example shows the Snapshot copies on the `vserverB:secondary1` destination:

```
cluster_dst::> volume snapshot show -vserver vserverB -volume secondary1

Vserver     Volume      Snapshot                State    Size  Total%
Used%
-------     ------      ---------- ----------- ------   -----  ------
-----
vserverB    secondary1  hourly.2013-01-25_0005  valid   224KB     0%
0%
                        daily.2013-01-25_0010   valid   92KB      0%
0%
                        hourly.2013-01-25_0105  valid   228KB     0%
0%
                        hourly.2013-01-25_0205  valid   236KB     0%
0%
                        hourly.2013-01-25_0305  valid   244KB     0%
0%
                        hourly.2013-01-25_0405  valid   244KB     0%
0%
                        hourly.2013-01-25_0505  valid   244KB     0%
0%

7 entries were displayed.
```

2. Restore a single file or LUN or a set of files or LUNs from a Snapshot copy in a SnapMirror destination volume:

   `snapmirror restore -source-path` *`SVM:volume|cluster://SVM/volume`*`, …`
   `-destination-path` *`SVM:volume|cluster://SVM/volume`*`, … -source-snapshot snapshot`
   `-file-list` *`source_file_path,@destination_file_path`*

   For complete command syntax, see the man page.

   > You must run this command from the destination SVM or the destination cluster.

   The following command restores the files `file1` and `file2` from the Snapshot copy `daily.2013-01-25_0010` in the original destination volume `secondary1`, to the same location in the active file system of the original source volume `primary1`:

```
cluster_dst::> snapmirror restore -source-path vserverB:secondary1
-destination-path vserverA:primary1 -source-snapshot daily.2013-01-
25_0010 -file-list /dir1/file1,/dir2/file2

[Job 3479] Job is queued: snapmirror restore for the relationship with
destination vserverA:primary1
```

The following command restores the files `file1` and `file2` from the Snapshot copy `daily.2013-01-25_0010` in the original destination volume `secondary1`, to a different location in the active file system of the original source volume `primary1`.

The destination file path begins with the @ symbol followed by the path of the file from the root of the original source volume. In this example, `file1` is restored to `/dir1/file1.new` and file2 is restored to `/dir2.new/file2` on `primary1`:

```
cluster_dst::> snapmirror restore -source-path vserverB:secondary1
-destination-path vserverA:primary1 -source-snapshot daily.2013-01-
25_0010 -file-list
/dir/file1,@/dir1/file1.new,/dir2/file2,@/dir2.new/file2

[Job 3479] Job is queued: snapmirror restore for the relationship with
destination vserverA:primary1
```

The following command restores the files `file1` and `file3` from the Snapshot copy `daily.2013-01-25_0010` in the original destination volume `secondary1`, to different locations in the active file system of the original source volume `primary1`, and restores `file2` from `snap1` to the same location in the active file system of `primary1`.

In this example, the file `file1` is restored to `/dir1/file1.new` and `file3` is restored to `/dir3.new/file3`:

```
cluster_dst::> snapmirror restore -source-path vserverB:secondary1
-destination-path vserverA:primary1 -source-snapshot daily.2013-01-
25_0010 -file-list
/dir/file1,@/dir1/file1.new,/dir2/file2,/dir3/file3,@/dir3.new/file3

[Job 3479] Job is queued: snapmirror restore for the relationship with
destination vserverA:primary1
```

Restore the contents of a volume from a SnapMirror destination

You can restore the contents of an entire volume from a Snapshot copy in a SnapMirror destination volume. You can restore the volume's contents to the original source volume or to a different volume.

About this task

The destination volume for the restore operation must be one of the following:

- A read-write volume, in which case SnapMirror performs an *incremental restore*, provided that the source and destination volumes have a common Snapshot copy (as is typically the case when you are restoring to the original source volume).

The command fails if there is not a common Snapshot copy. You cannot restore the contents of a volume to an empty read-write volume.

- An empty data protection volume, in which case SnapMirror performs a *baseline restore*, in which the specified Snapshot copy and all the data blocks it references are transferred to the source volume.

Restoring the contents of a volume is a disruptive operation. CIFS traffic must not be running on the SnapVault primary volume when a restore operation is running.

If the destination volume for the restore operation has compression enabled, and the source volume does not have compression enabled, disable compression on the destination volume. You need to re-enable compression after the restore operation is complete.

Any quota rules defined for the destination volume are deactivated before the restore is performed. You can use the `volume quota modify` command to reactivate quota rules after the restore operation is complete.

Steps

1. List the Snapshot copies in the destination volume:

   `volume snapshot show -vserver` *`SVM`* `-volume` *`volume`*

   For complete command syntax, see the man page.

   The following example shows the Snapshot copies on the `vserverB:secondary1` destination:

```
cluster_dst::> volume snapshot show -vserver vserverB -volume secondary1

Vserver     Volume      Snapshot                State    Size  Total%
Used%
-------     ------      ---------- ----------- ------   -----  ------
-----
vserverB    secondary1  hourly.2013-01-25_0005  valid   224KB     0%
0%
                        daily.2013-01-25_0010   valid   92KB      0%
0%
                        hourly.2013-01-25_0105  valid   228KB     0%
0%
                        hourly.2013-01-25_0205  valid   236KB     0%
0%
                        hourly.2013-01-25_0305  valid   244KB     0%
0%
                        hourly.2013-01-25_0405  valid   244KB     0%
0%
                        hourly.2013-01-25_0505  valid   244KB     0%
0%

7 entries were displayed.
```

2. Restore the contents of a volume from a Snapshot copy in a SnapMirror destination volume:

   `snapmirror restore -source-path` *`SVM:volume|cluster://SVM/volume`*`, …`
   `-destination-path` *`SVM:volume|cluster://SVM/volume`*`, … -source-snapshot` *`snapshot`*

   For complete command syntax, see the man page.

   > You must run this command from the destination SVM or the destination cluster.

   The following command restores the contents of the original source volume `primary1` from the Snapshot copy `daily.2013-01-25_0010` in the original destination volume `secondary1`:

```
cluster_dst::> snapmirror restore -source-path vserverB:secondary1
-destination-path vserverA:primary1 -source-snapshot daily.2013-01-
25_0010

Warning: All data newer than Snapshot copy daily.2013-01-25_0010 on
volume vserverA:primary1 will be deleted.

Do you want to continue? {y|n}: y

[Job 34] Job is queued: snapmirror restore from source
vserverB:secondary1 for the snapshot daily.2013-01-25_0010.
```

3. Remount the restored volume and restart all applications that use the volume.

Update a replication relationship manually

You might need to update a replication relationship manually if an update fails because the source volume has been moved.

About this task

SnapMirror aborts any transfers from a moved source volume until you update the replication relationship manually.

Beginning with ONTAP 9.5, SnapMirror Synchronous relationships are supported. Although the source and destination volumes are in sync at all times in these relationships, the view from the secondary cluster is synchronized with the primary only on an hourly basis. If you want to view the point-in-time data at the destination, you should perform a manual update by running the `snapmirror update` command.

Step

1. Update a replication relationship manually:

   `snapmirror update -source-path` *`SVM:volume|cluster://SVM/volume`*`, … -destination`
   `-path` *`SVM:volume|cluster://SVM/volume`*`, …`

   For complete command syntax, see the man page.

You must run this command from the destination SVM or the destination cluster. The command fails if a common Snapshot copy does not exist on the source and destination. Use `snapmirror initialize` to re-initialize the relationship.

The following example updates the relationship between the source volume `volA` on `svm1` and the destination volume `volA_dst` on `svm_backup`:

```
cluster_src::> snapmirror update -source-path svm1:volA -destination
-path svm_backup:volA_dst
```

Resynchronize a replication relationship

You need to resynchronize a replication relationship after you make a destination volume writeable, after an update fails because a common Snapshot copy does not exist on the source and destination volumes, or if you want to change the replication policy for the relationship.

About this task

Although resync does not require a baseline transfer, it can be time-consuming. You might want to run the resync in off-peak hours.

Step

1. Resync the source and destination volumes:

   ```
   snapmirror resync -source-path SVM:volume|cluster://SVM/volume, … -destination
   -path SVM:volume|cluster://SVM/volume, … -type DP|XDP -schedule schedule
   -policy policy
   ```

   For complete command syntax, see the man page.

   You must run this command from the destination SVM or the destination cluster.

   The following example resyncs the relationship between the source volume `volA` on `svm1` and the destination volume `volA_dst` on `svm_backup`:

   ```
   cluster_dst::> snapmirror resync -source-path svm1:volA -destination
   -path svm_backup:volA_dst
   ```

Delete a volume replication relationship

You can use the `snapmirror delete` and `snapmirror release` commands to delete a volume replication relationship. You can then delete unneeded destination volumes manually.

About this task

The `snapmirror release` command deletes any SnapMirror-created Snapshot copies from the source. You can use the `-relationship-info-only` option to preserve the Snapshot copies.

Steps

1. Quiesce the replication relationship:

   `snapmirror quiesce -destination-path` *`SVM:volume|cluster://SVM/volume`*

   ```
   cluster_dst::> snapmirror quiesce -destination-path svm_backup:volA_dst
   ```

2. Delete the replication relationship:

   `snapmirror delete -source-path` *`SVM:volume|cluster://SVM/volume`*`, … -destination -path` *`SVM:volume|cluster://SVM/volume`*`, …`

   For complete command syntax, see the man page.

   You must run this command from the destination cluster or destination SVM.

   The following example deletes the relationship between the source volume `volA` on `svm1` and the destination volume `volA_dst` on `svm_backup`:

   ```
   cluster_dst::> snapmirror delete -source-path svm1:volA -destination
   -path svm_backup:volA_dst
   ```

3. Release replication relationship information from the source SVM:

   `snapmirror release -source-path` *`SVM:volume|cluster://SVM/volume`*`, … -destination-path` *`SVM:volume|cluster://SVM/volume`*`, …`

   For complete command syntax, see the man page.

   You must run this command from the source cluster or source SVM.

   The following example releases information for the specified replication relationship from the source SVM `svm1`:

   ```
   cluster_src::> snapmirror release -source-path svm1:volA -destination
   -path svm_backup:volA_dst
   ```

Manage storage efficiency

SnapMirror preserves storage efficiency on the source and destination volumes, with one exception, when postprocess data compression is enabled on the destination. In that case, all storage efficiency is lost on the destination. To correct this issue, you need to

disable postprocess compression on the destination, update the relationship manually, and re-enable storage efficiency.

What you'll need

- The source and destination clusters and SVMs must be peered.

  Cluster and SVM peering

- You must disable postprocess compression on the destination.

About this task

You can use the `volume efficiency show` command to determine whether efficiency is enabled on a volume. For more information, see the man pages.

You can check if SnapMirror is maintaining storage efficiency by viewing the SnapMirror audit logs and locating the transfer description. If the transfer description displays `transfer_desc=Logical Transfer`, SnapMirror is not maintaining storage efficiency. If the transfer description displays `transfer_desc=Logical Transfer with Storage Efficiency`, SnapMirror is maintaining storage efficiency. For example:

```
Fri May 22 02:13:02 CDT 2020 ScheduledUpdate[May 22 02:12:00]:cc0fbc29-
b665-11e5-a626-00a09860c273 Operation-Uuid=39fbcf48-550a-4282-a906-
df35632c73a1 Group=none Operation-Cookie=0 action=End source=<sourcepath>
destination=<destpath> status=Success bytes_transferred=117080571
network_compression_ratio=1.0:1 transfer_desc=Logical Transfer - Optimized
Directory Mode
```

Logical Transfer with storage

Beginning with ONTAP 9.3, manual update is no longer required to re-enable storage efficiency. If SnapMirror detects that postprocess compression has been disabled, it automatically re-enables storage efficiency at the next scheduled update. Both the source and the destination must be running ONTAP 9.3.

Beginning with ONTAP 9.3, AFF systems manage storage efficiency settings differently from FAS systems after a destination volume is made writeable:

- After you make a destination volume writeable using the `snapmirror break` command, the caching policy on the volume is automatically set to "auto" (the default).

  This behavior is applicable to FlexVol volumes, only, and it does not apply to FlexGroup volumes.

- On resync, the caching policy is automatically set to "none", and deduplication and inline compression are automatically disabled, regardless of your original settings. You must modify the settings manually as needed.

Manual updates with storage efficiency enabled can be time-consuming. You might want to run the operation in off-peak hours.

Step

1. Update a replication relationship and re-enable storage efficiency:

   `snapmirror update -source-path SVM:volume|cluster://SVM/volume, … -destination -path SVM:volume|cluster://SVM/volume, … -enable-storage-efficiency true`

   For complete command syntax, see the man page.

   > You must run this command from the destination SVM or the destination cluster. The command fails if a common Snapshot copy does not exist on the source and destination. Use `snapmirror initialize` to re-initialize the relationship.

   The following example updates the relationship between the source volume `volA` on `svm1` and the destination volume `volA_dst` on `svm_backup`, and re-enables storage efficiency:

   ```
   cluster_dst::> snapmirror update -source-path svm1:volA -destination
   -path svm_backup:volA_dst -enable-storage-efficiency true
   ```

Use SnapMirror global throttling

Global network throttling is available for all SnapMirror and SnapVault transfers at a per-node level.

About this task

SnapMirror global throttling restricts the bandwidth used by incoming and/or outgoing SnapMirror and SnapVault transfers. The restriction is enforced cluster wide on all nodes in the cluster.

For example, if the outgoing throttle is set to 100 Mbps, each node in the cluster will have the outgoing bandwidth set to 100 Mbps. If global throttling is disabled, it is disabled on all nodes.

> The throttle has no effect on `volume move` transfers or load-sharing mirror transfers. Although data transfer rates are often expressed in bits per second (bps), the throttle values must be entered in kilobytes per second (KBps).

Global throttling works with the per-relationship throttle feature for SnapMirror and SnapVault transfers. The per-relationship throttle is enforced until the combined bandwidth of per-relationship transfers exceeds the value of the global throttle, after which the global throttle is enforced. A throttle value `0` implies that global throttling is disabled.

> SnapMirror global throttling has no effect on SnapMirror Synchronous relationships when they are In-Sync. However, the throttle does effect SnapMirror Synchronous relationships when they perform an asynchronous transfer phase such as an initialization operation or after an Out Of Sync event. For this reason, enabling global throttling with SnapMirror Synchronous relationships is not recommended.

Steps

1. Enable global throttling:

   `options -option-name replication.throttle.enable on|off`

The following example shows how to enable SnapMirror global throttling on `cluster_dst`:

```
cluster_dst::> options -option-name replication.throttle.enable on
```

2. Specify the maximum total bandwidth used by incoming transfers:

   `options -option-name replication.throttle.incoming.max_kbs` *KBps*

   The recommended minimum throttle bandwidth is 4 KBps and the maximum is up to 2 TBps. The default value for this option is `unlimited`, which means there is no limit on total bandwidth used.

   The following example shows how to set the maximum total bandwidth used by incoming transfers to 100 Mbps:

   ```
   cluster_dst::> options -option-name
   replication.throttle.incoming.max_kbs 12500
   ```

   100 Mbps = 12500 KBps

3. Specify the maximum total bandwidth used by outgoing transfers:

   `options -option-name replication.throttle.outgoing.max_kbs` *KBps*

   KBps is the maximum transfer rate in kilobytes per second. Valid transfer rate values are 1 to 125000. The default value for this option is `unlimited`, which means there is no limit on total bandwidth used.

   The following example shows how to set the maximum total bandwidth used by outgoing transfers to 100 Mbps:

   ```
   cluster_dst::> options -option-name
   replication.throttle.outgoing.max_kbs 12500
   ```