4

Managing Windows Storage

In this chapter, we cover the following recipes:

* Managing physical disks and disk volumes
* Managing NTFS permissions
* Managing Storage Replica
* Managing Filestore quotas
* Using filesystem reporting

# Introduction

Windows Server 2019 provides a range of features that allows access to a wide variety of storage and storage devices. Windows supports spinning disks, USB memory sticks, and SSD devices (including MVMe SSD devices).

Before a disk can be used, you need to create partitions or volumes on the device, then format the volume. When you first initialize a disk, you need to define which partitioning method to use. You have two choices: Master Boot Record (MBR) or GUID Partition Table (GPT). For a good discussion of the differences between these two mechanisms, see: <https://www.howtogeek.com/193669/whats-the-difference-between-gpt-and-mbr-when-partitioning-a-drive/>.

With a volume created, you can then format the disk volume. Windows supports five key filesystems you can use: ReFS, NTFS, exFAT, UDF, and FAT32. For details of the latter four, see: <https://docs.microsoft.com//windows/desktop/fileio/filesystem-functionality-comparison>. The ReFS filesystem is newer and is based on NFTS but lacks some features a file server might need (it has no encrypted files). For a comparison between the ReFS and NTFS filesystems, see: <https://www.iperiusbackup.net/en/refs-vs-ntfs-differences-and-performance-comparison-when-to-use/>. You examine partitioning and formatting volumes in the Managing physical disks and disk volumes recipe.

NTFS (and ReFS) volumes allow you to create access control lists (ACLs) that control access to files and folders stored in Windows volumes. Managing ACLs is somewhat difficult and lacks rich PowerShell support. To manage ACLs on NTFS volumes, as you will see in the Managing NTFS permissions recipe, you can download and use a third-party module, NTFSecurity.

The Windows File Server Resource Manager (FSRM) feature in Windows Server 2019 helps you to manage filestore resources. With FSRM, you can set Filestore quotas. With soft quotas, users can exceed their Filestore quota and administrators are notified when this occurs. With hard quotas, users are prohibited from storing more than their allocated quotas. You'll work with quotas in the Managing Filestore quotas recipe.

FSRM also supports detailed file server reporting, which you examine in the Using filesystem reporting recipe. These reports, and the supporting XML files, can provide assistance in managing and controlling file servers.

This chapter, specifically the Managing Filestore quotas recipe, makes use of a free email account from [Sendgrid.com](http://Sendgrid.com). This enables you to set up the Windows SMTP server to forward mail to SendGrid for onward transmission. This could be a great way to test reporting or to avoid issues with internal SMTP servers.

# Managing physical disks and disk volumes

Windows Server 2019 requires a computer with at least one disk drive (that is, the C:\ drive). A disk drive can be connected via different bus types, such as IDE, SATA, SAS, or USB. Before you can utilize a disk in Windows, you need to initialize it and create volumes or partitions.

There are two partitioning schemes you can use: the older format of MBR, and the newer GPT. The MBR scheme, first introduced with the PC DOS 2 in 1983, had a number of restrictions. For example, the largest partition supported with MBR is just 2 TB. And creating more than four partitions required you to create an extended partition and create additional partitions inside the extended partition. The GPT scheme provides much larger drives (partition limits are OS-imposed), as well as up to 128 partitions per drive.

In this recipe, you add two new disk devices to a server, SRV1, and then create new volumes/partitions on those disks.

## Getting ready

You run this recipe on SRV1. To perform this recipe, SRV1 needs two additional disks. If you're using a Hyper-V VM to test this recipe, you can use the following script to add the necessary disks. Run this on your Hyper-V Host that runs the SRV1 VM:

# Create Virtual Disks to add to SRV1

New-VHD -Path D:\v6\SRV1\SRV1-F.vhdx -SizeBytes 20gb -Dynamic

New-VHD -Path D:\v6\SRV1\SRV1-G.vhdx -SizeBytes 20gb -Dynamic

# Add them to the VM

$HDHT1 = @{

VMName = 'SRV1'

Path = 'D:\v6\SRV1\SRV1-F.vhdx'

ControllerType = 'SCSI'

ControllerNumber = 0

}

Add-VMHardDiskDrive @HDHT1

$HDHT2 = @{

VMName = 'SRV1'

Path = 'D:\v6\SRV1\SRV1-G.vhdx'

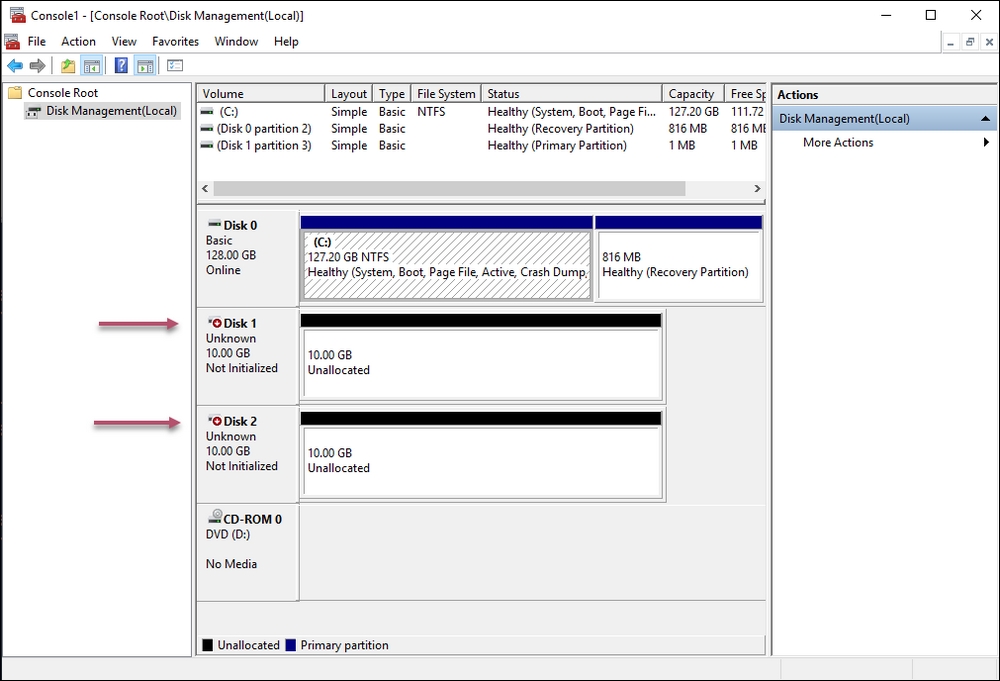
ControllerType = 'SCSI'

ControllerNumber = 0

}

Add-VMHardDiskDrive @HDHT2

The GitHub repository for this book contains a script, Add-DiskstoSrv1+2.ps1, which creates the disks used in this and other recipes in this chapter. Once you've added the (virtual) disks to the SRV1 server, you can use Disk Management, a Control Panel applet, to view the starting disk configuration for this recipe, like this:



## How to do it...

1. Get physical disks on this system:

Get-Disk |

Format-Table -AutoSize

1. Initialize the disks:

Get-Disk |

Where PartitionStyle -eq Raw |

Initialize-Disk -PartitionStyle GPT

1. Redisplay the disks on SRV1:

Get-Disk |

Format-Table -AutoSize

1. Create a volume in Disk 1:

$NVHT1 = @{

DiskNumber = 1

FriendlyName = 'Storage(F)'

FileSystem = 'NTFS'

DriveLetter = 'F'

}

New-Volume @NVHT1

1. Create two volumes in disk 2—first, create G:

New-Partition -DiskNumber 2 -DriveLetter G -Size 4gb

1. Create a second partition, H:

New-Partition -DiskNumber 2 -DriveLetter H -UseMaximumSize

1. Format the G: and H: drives:

$NVHT1 = @{

DriveLetter = 'G'

FileSystem = 'NTFS'

NewFileSystemLabel = 'Log'}

Format-Volume @NVHT1

$NVHT2 = @{

DriveLetter = 'H'

FileSystem = 'NTFS'

NewFileSystemLabel = 'GDShow'}

Format-Volume @NVHT2

1. Get partitions on this system:

Get-Partition |

Sort-Object -Property DriveLetter |

Format-Table -Property DriveLetter, Size, Type

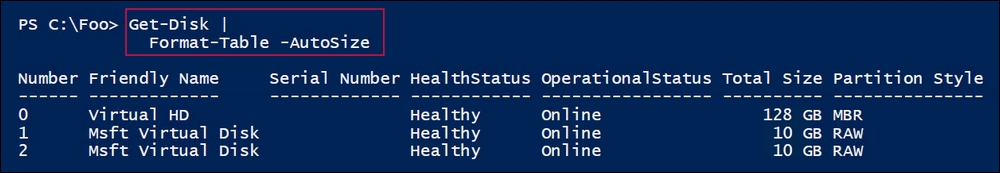
1. Get volumes on SRV1:

Get-Volume |

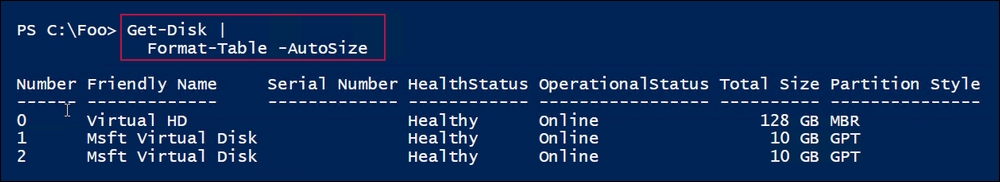
Sort-Object -Property riveLetter

## How it works…

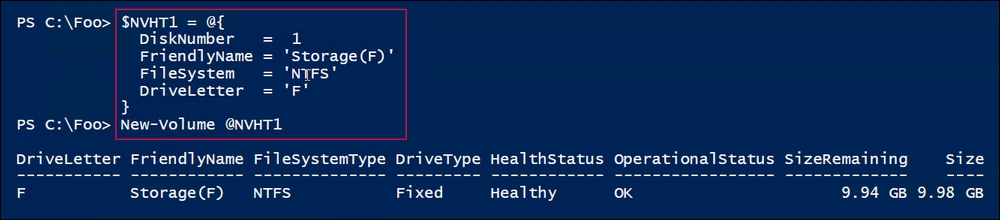
In step 1, you look at the disks available on SRV1, noting the two new disks:



In step 2, you initialize the two new drives—this produces no output. In step 3, you redisplay the disks, which looks like this:



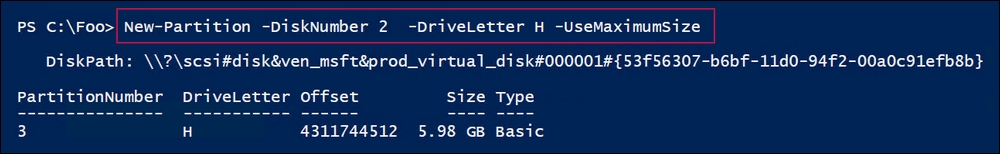
In step 4, you use the New-Volume cmdlet to partition and format a disk volume (F:) in the first added disk, which looks like this:



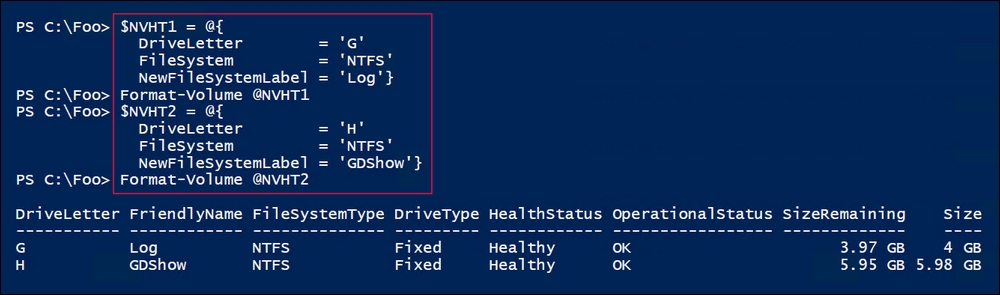
With step 5, you create the first of two new partitions on the second drive added to SRV1, which looks like this:



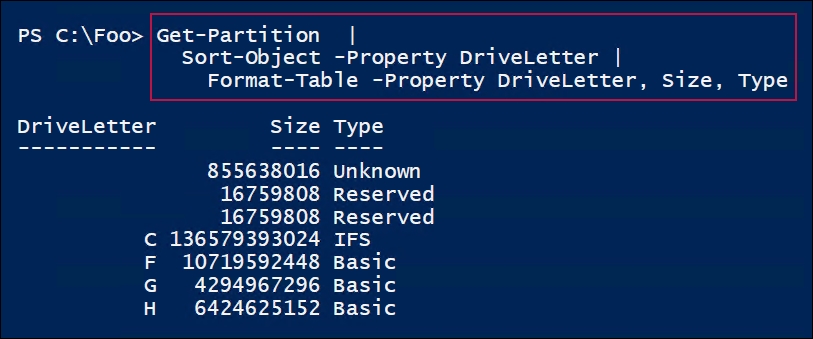
In step 6, you create a second partition on the second disk, the G: drive. That looks like this:



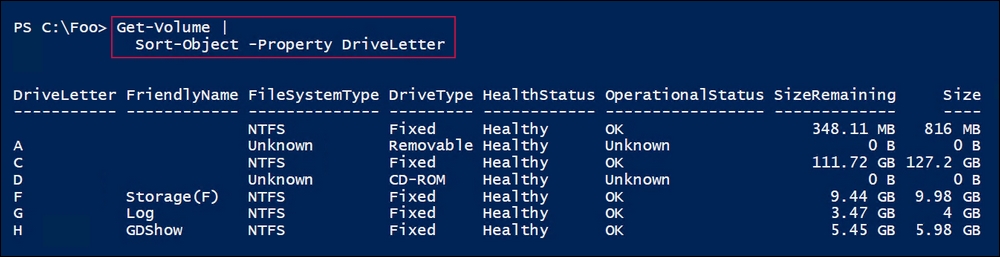
In step 7, you format the two partitions you just created, which looks like this:



In step 8, you use the Get-Partition cmdlet to return the partitions on the SRV1 server, which looks like this:



In step 9, you use the Get-Volume cmdlet to return the volumes (also known as the partitions) on the SRV1 server, which looks like this:



## There's more...

In the Getting ready section of this recipe, you add two new disks to SRV1. These new disks are uninitialized, thus the first thing to do, in step 1, is to initialize the disks.

In step 4, you create a new volume on Disk 1. This creates the partition and then formats the drive. In step 5 and step 6, you create two new partitions on disk 2 which, in step 7, you format. This shows two ways of creating drives within a disk.

In step 8 and step 9, you use different cmdlets to return what's essentially the same set of objects—the volumes/partitions on SRV1.

# Managing NTFS permissions

Every file and folder in an NTFS filesystem has an Access Control List (ACL). The ACL contains a set of Access Control Entries (ACEs). Each ACE defines a permission to a file or folder for an account. For example, the Sales AD global group could be given full control of a file.

Permissions can also be inherited from parent folders. If you create a new folder and then create a file within that folder, the new file inherits permissions from the parent folder and from any further parent folder(s) by default. You can manage the ACL list to add or remove permissions and you can modify inheritance.

There's limited PowerShell support for managing NTFS permissions. PowerShell does have the Get-ACL and Set-ACL cmdlets, but creating the individual ACEs and managing inheritance requires the use of the .NET Framework (by default). A simpler approach is to use a third-party module, NTFSSecurity, which makes managing ACEs and ACLs, including dealing with inheritance, much simpler.

## Getting ready

This recipe uses SRV1, a general-purpose domain-joined server that runs Windows Server 2019. You also need to have a domain controller (such as DC1) up and running. The Sales group should exist in Active Directory (AD), but if not, this recipe creates the group. You should use an account that's a member of the Domain Admins group and run this recipe from an elevated console. Additionally, you should have the AD RSAT tools loaded onto SRV1.

## How to do it...

1. Download and install the NTFSSecurity module from the PowerShell Gallery:

Install-Module NTFSSecurity -Force

1. Use the following Get commands in the module:

Get-Command -Module NTFSSecurity

1. Create a new folder and a new file in the following folder:

New-Item -Path C:\Secure1 -ItemType Directory

"Secure" | Out-File -FilePath C:\Secure1\Secure.Txt

1. View the ACL of the folder:

Get-NTFSAccess -Path C:\Secure1 |

Format-Table -AutoSize

1. View the ACL of the file:

Get-NTFSAccess C:\Secure1\Secure.Txt |

Format-Table -AutoSize

1. Create the Sales group, if it doesn't already exist:

try {

Get-ADGroup -Identity 'Sales' -ErrorAction Stop

}

catch {

New-ADGroup -Name Sales -GroupScope Global

}

1. Display the group:

Get-ADGroup -Identity Sales

1. Give the domain administrators full control of the folder:

$AHT1 = @{

Path = 'C:\Secure1'

Account = 'Reskit\Domain Admins'

AccessRights = 'FullControl'

}

Add-NTFSAccess @AHT1

1. Remove the Builtin\Users access from the Secure.Txt file:

$AHT2 = @{

Path = 'C:\Secure1\Secure.Txt'

Account = 'Builtin\Users'

AccessRights = 'FullControl'

}

Remove-NTFSAccess @AHT2

1. Remove the inherited rights for the folder:

$IRHT1 = @{

Path = 'C:\Secure1'

RemoveInheritedAccessRules = $True

}

Disable-NTFSAccessInheritance @IRHT1

1. Add Sales group access to the folder:

$AHT3 = @{

Path = 'C:\Secure1\'

Account = 'Reskit\Sales'

AccessRights = 'FullControl'

}

Add-NTFSAccess @AHT3

1. Get the updated ACL for the folder:

Get-NTFSAccess -Path C:\Secure1 |

Format-Table -AutoSize

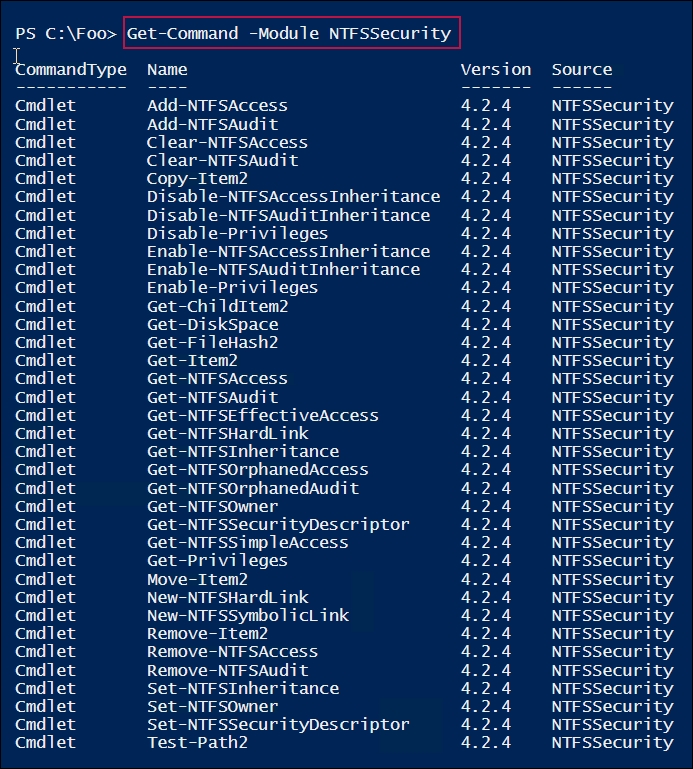
1. View the updated ACL on the file:

Get-NTFSAccess -Path C:\Secure1\Secure.Txt |

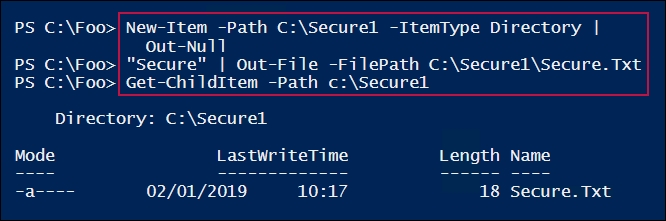
Format-Table -AutoSize

## How it works…

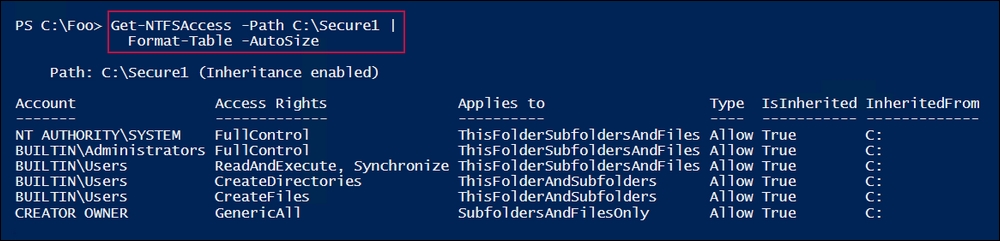
In step 1, you download and install the NTFSSecurity module from the PowerShell Gallery on the internet, which produces no output. In step 2, you use the Get-Command cmdlet to view the commands inside the NTFSSecurity module, which looks like this:



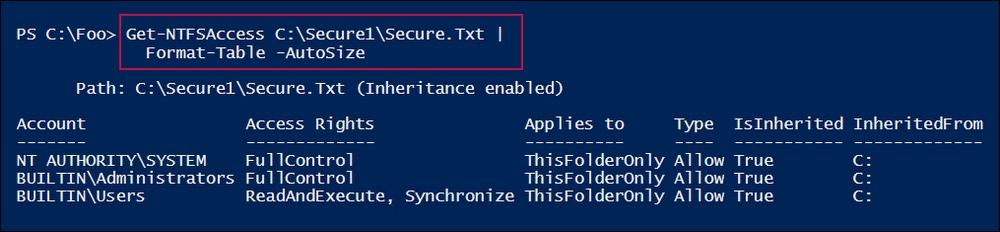
In step 3, you create a new folder, C:\Secure1, and a file within that folder (C:\Secure1\Secure.Txt) that looks like this:



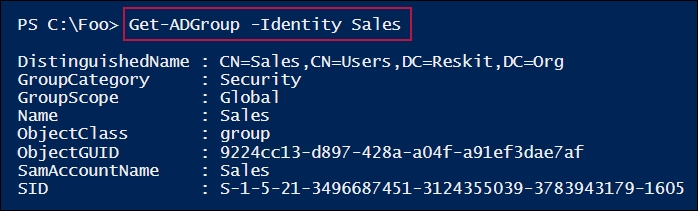
In step 4, you use the Get-NTFSAccess cmdlet to view the ACL for the C:\Secure1 folder, which looks like this:



In step 5, you use the Get-NTFSAccess cmdlet to view the ACL on the C:\Secure1\Secure.txt file, which looks like this:

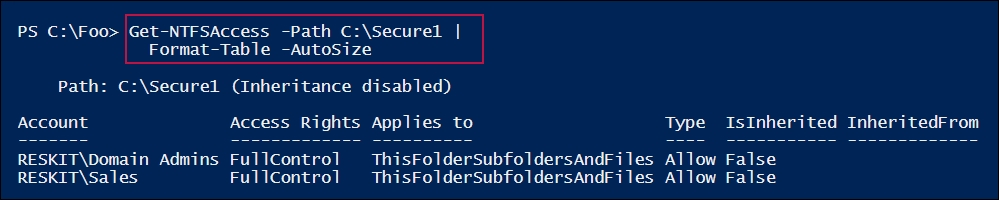


In step 6, you create a global group (Sales) in the Reskit.Org domain (if the group doesn't already exist). Whether or not the global group exists, this step produces no output. In step 7, you view the Sales global group, which looks like this:

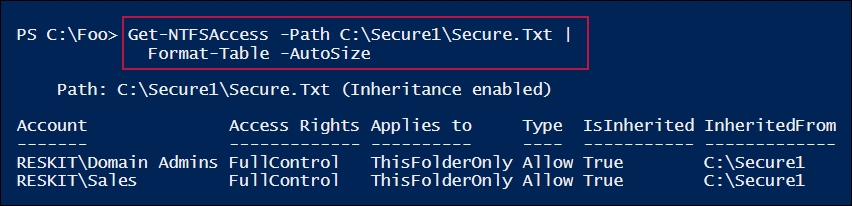


In step 8, you add an explicit ACE that gives the members of the Domain Admins group full control. In step 9, you remove the access ACE for Builtin\Users from the file's ACL. And in step 10, you remove the ACE entries on the C:\Secure1\Secure.txt file that were initially inherited from the C:\Secure1 folder. In step 11, you add an ACL that gives full control to members of the Reskit\Sales global group. These four steps produce no output.

In step 12, you use the Get-NTFSAccess cmdlet to review the ACL for the C:\Secure1 folder, which looks like this:



In step 13, you view the ACL on the C:\Secure1\Secure.Txt file, which looks like this:



## There's more...

In step 3, you create a folder (C:\Secure1) and a file (C:\Secure1\Secure.Txt). In later steps, you adjust the ACLs on the file and folder to support the organization's security policy.

In step 6, you ensure that a global group, Sales, exists in the AD that you use to set ACLs. If the global group already exists (for example, if you've used other recipes in this book that make use of that group) then this step does nothing.

In step 8, you add a full control ACE to the folder for the Reskit\Domain Admins group. This means that, since you are logged on as a member of that group, when you remove the default ACE for Builtin\Users in step 9, you still have access to the folder and folder contents to continue the recipe.

After you adjust the ACL to the folder, you can see, in step 12 and step 13, that the C:\Secure1 folder no longer inherits ACE entries from the parent folder. Additionally, you can see that the file's ACL includes an ACE inherited from the folder.

In a production environment, it might be appropriate to remove the permissions for the Domain Admins account once the users are able to access and use the folder successfully. If a Domain Administrator does need to change the ACL, they could just take ownership of the folder, give themselves full control of the folder, and then perform any needed management (and removing any temporary access once this maintenance is complete).

## See also

This recipe shows you how to use the NTFSSecurity module to manage aspects of file and folder ACLs. The module was first announced in 2014 and has been improved since. See <https://blogs.technet.microsoft.com/fieldcoding/2014/12/05/ntfssecurity-tutorial-1-getting-adding-and-removing-permissions/> for more details on the module.

# Managing Storage Replica

Storage Replica (SR) is a feature of Windows Server 2019 that replicates storage volumes to other systems. SR is only available with the Windows Server 2019 Datacentre edition.

## Getting ready

This recipe makes use of additional disk volumes on SRV1 and SRV2. In the Managing physical disks and disk volumes recipe, you added two additional disks to SRV1. In this recipe, you need to have two additional disks added to SRV2 that form the basis for storage replication. This recipe requires two additional hard disks for SRV2—you can use the Add-DiskstoSrv1+2.ps1 script to create the disks used in this and other recipes in this chapter. You can find the script on this book's GitHub repository (<https://github.com/DoctorDNS/PowerShellCookBook2019/blob/master/Chapter%2004%20-%20Managing%20Storage/Add-DiskstoSRV1%2B2.ps1>).

Run this recipe on SRV1 with SRV2 and DC1 online:

1. Create content on the F: drive in SRV1:

1..100 | ForEach {

$NF = "F:\CoolFolder$\_"

New-Item -Path $NF -ItemType Directory | Out-Null

1..100 | ForEach {

$NF2 = "$NF\CoolFile$\_"

"Cool File" | Out-File -PSPath $NF2

}

}

1. Show what's on F: locally on SRV1:

Get-ChildItem -Path F:\ -Recurse | Measure-Object

1. Examine the same drives remotely on SRV2:

$SB = {

Get-ChildItem -Path F:\ -Recurse |

Measure-Object

}

Invoke-Command -ComputerName SRV2 -ScriptBlock $SB

1. Add a Storage Replica feature to SRV1:

Add-WindowsFeature -Name Storage-Replica

1. Restart SRV1 to finish the installation process:

Restart-Computer

1. Add a storage-replica feature to SRV2:

$SB= {

Add-WindowsFeature -Name Storage-Replica | Out-Null

}

Invoke-Command -ComputerName SRV2 -ScriptBlock $SB

1. Restart SRV2 and wait until the restart is complete:

$RSHT = @{

ComputerName = 'SRV2'

Force = $true

}

Restart-Computer @RSHT -Wait -For PowerShell

1. Create a Storage Replica by replicating from F: on SRV1 to F: on SRV2:

$SRHT = @{

SourceComputerName = 'SRV1'

SourceRGName = 'SRV1RG'

SourceVolumeName = 'F:'

SourceLogVolumeName = 'G:'

DestinationComputerName = 'SRV2'

DestinationRGName = 'SRV2RG'

DestinationVolumeName = 'F:'

DestinationLogVolumeName = 'G:'

LogSizeInBytes = 2gb

}

New-SRPartnership @SRHT -Verbose

1. View the storage replication partnership:

Get-SRPartnership

1. Examine the same drives remotely on SRV2:

$SB = {

Get-Volume |

Sort-Object -Property DriveLetter |

Format-Table

}

Invoke-Command -ComputerName SRV2 -ScriptBlock $SB

1. Reverse the replication:

$SRHT2 = @{

NewSourceComputerName = 'SRV2'

SourceRGName = 'SRV2RG'

DestinationComputerName = 'SRV1'

DestinationRGName = 'SRV1RG'

Confirm = $False

}

Set-SRPartnership @SRHT2

1. View the replication partnership after reversing:

Get-SRPartnership

1. Examine the same drives remotely on SRV2:

$SB = {

Get-ChildItem -Path F:\ -Recurse |

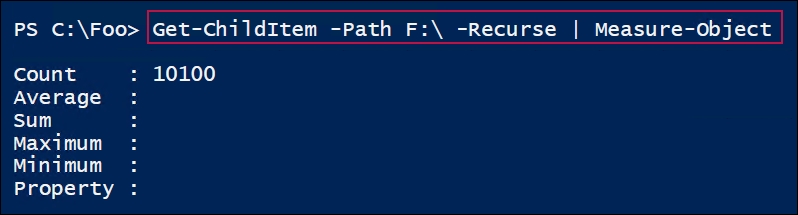
Measure-Object

}

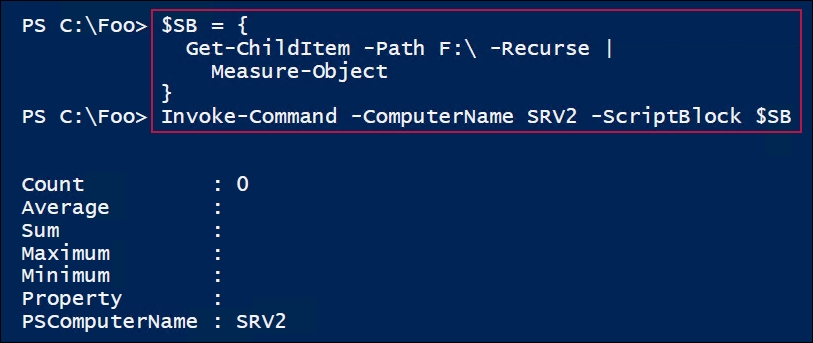
Invoke-Command -ComputerName SRV2 -ScriptBlock $SB

## How it works…

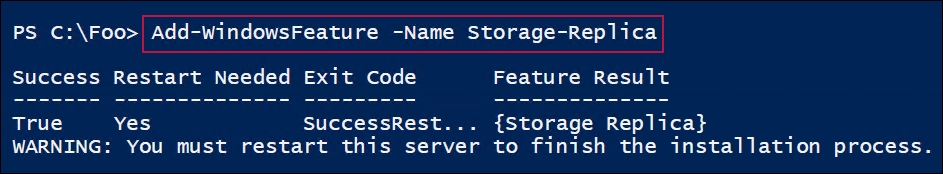
In step 1, you create 100 folders on the F: of SRV1. Inside each folder, you also create 100 files. Each file contains some content. This step produces no output. In step 2, you view what you just created, which looks like this:



In step 3, you view the F: drive on SRV2 (which contains no files yet). The output from step 3 looks like this:

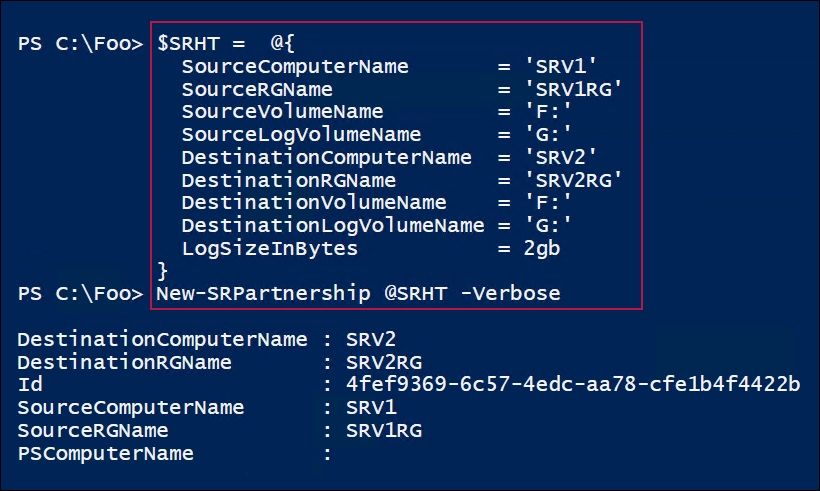


In step 4, you add the Storage Replica feature to SRV1, which looks like this:

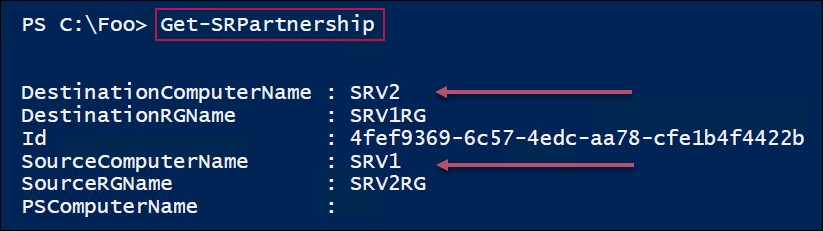


In step 5, you reboot SRV1. In step 6 and step 7, you add the Storage Replica feature to SRV2 then reboot SRV2. These three steps produce no output.

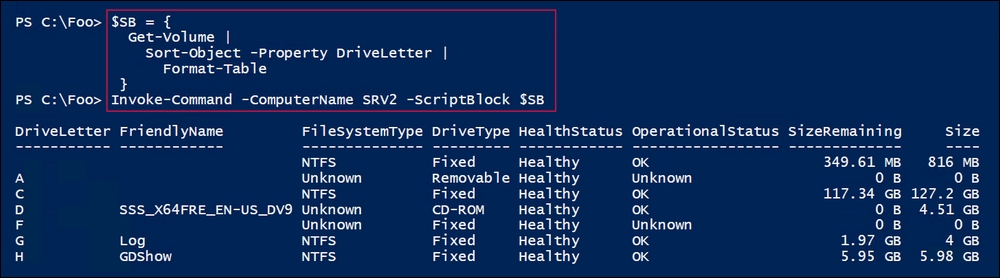
In step 8, you create a Storage Replica partnership, replicating the contents of F: on SRV1 to F: on SRV2 (with the G: drive on both servers serving as a log file folder for storage replication). The output from this step looks like this:



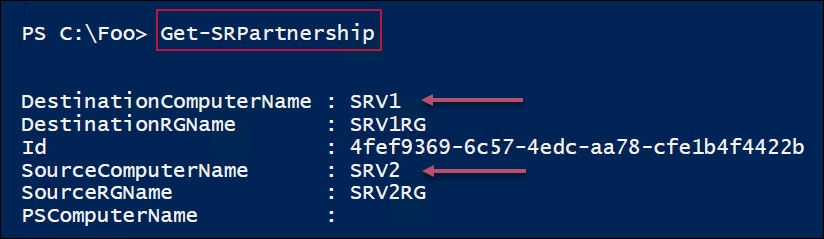
In step 9, you use the Get-SRPartnership cmdlet to view the now-reversed replication partnership, which looks like this:



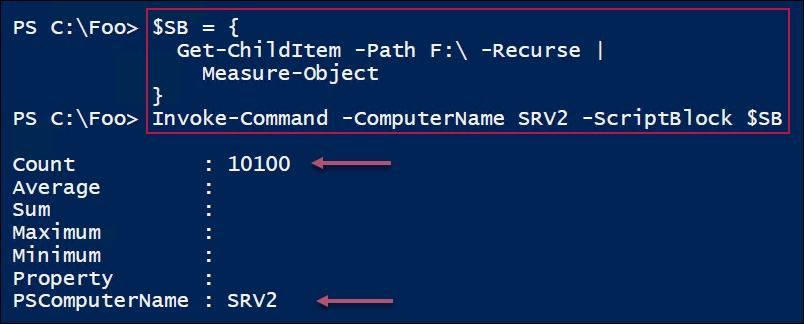
In step 10, you examine the volumes available on SRV2, which look like this:



Thus far, you're replicating from SRV1 to SRV2. In step 11, you reverse the replication, which generates no output. With step 12, you can see that the replication is now reversed, replicating from files on SRV2 to SRV1. The output from this step looks like this:



In the final step, step 13, you count the files now available on SRV2, which looks like this:



## There's more...

In this recipe, you create a set of 100 folders and 10,000 files on SRV1, which you replicate to SRV2. After the replication starts, you can't actually see anything useful on SRV2 since the volume is being used by SR itself. As you can see in step 10, the replicated data isn't viewable on SRV2—the partition into SR is replicating has no apparent filesystem. Once you reverse the replication, as you can see in step 11 and step 12, the files and folders on SRV2 are now viewable (and those on SRV1 wouldn't be viewable). This is a feature of SR that prevents accidentally storing data on the replicated disk.

# Managing Filestore quotas

The FSRM is a feature of the Windows server that assists you in managing file servers. FSRM has three key features:

* Quota management: With FSRM, you can set soft or hard quotas on volumes and folders. A soft quota allows a user to exceed an allowance, while hard quotas stop a user from exceeding an allowance. You can configure a quota with thresholds and threshold actions. If a user exceeds 65% of the quota allowance, FSRM can send an email, while at 90%, you log an event in the event log or run a program. You have different actions for different quota levels. This recipe shows how to use quotas.
* File screening: You can set up a file screen and stop a user from saving screened files. For example, you could screen for .MP3, or FLAC files—should a user then attempt to save a file (say, jg75-02-28D1T1.flac), the file screen rejects the request and doesn't allow the user to save the file.
* Reporting: FSRM enables you to create a wealth of storage reports that can be highly useful for management purposes.

In this recipe, you install FSRM, perform some general configuration, and then work with soft and hard quotas.

## Getting ready

This recipe makes use of an email server so that FSRM can send email to the admin. To test the email-related components of this recipe, you need have an SMTP server or an email-forwarder. The resultant emails generated by this recipe were sent to SRV1, then forwarded to a free email service at <https://www.sendgrid.com>. With a SendGrid account in place, you can add the SMTP service to a server in your environment and then configure it to forward mail to SendGrid to then send the emails onward.

## How to do it...

1. Install the FSRM feature:

$IHT = @{

Name = 'FS-Resource-Manager'

IncludeManagementTools = $True

}

Install-WindowsFeature @IHT

1. Set the SMTP settings in FSRM:

$MHT = @{

SmtpServer = 'SRV1.Reskit.Org' # Previously setup

FromEmailAddress = 'FSRM@Reskit.Org'

AdminEmailAddress = 'Doctordns@Gmail.Com'

}

Set-FsrmSetting @MHT

1. Send a test email to check the setup:

$MHT = @{

ToEmailAddress = 'DoctorDNS@gmail.com'

Confirm = $false

}

Send-FsrmTestEmail @MHT

1. Create a new FSRM quota template for a 10 MB quota:

$QHT1 = @{

Description = 'Quota of 10MB'

Name = 'TenMB Limit'

Size = 10MB

}

New-FsrmQuotaTemplate @QHT1

1. Create another quota template for a quota of 5 MB:

$QHT2 = @{

Name = 'Soft 5MB Limit'

Description = 'Soft Quota of 5MB'

Size = 5MB

SoftLimit = $True

}

New-FsrmQuotaTemplate @QHT2

1. View the available FSRM quota templates:

Get-FsrmQuotaTemplate |

Format-Table -Property Name, Description, Size, SoftLimit

1. Create two new folders on which to place quotas:

If (-Not (Test-Path C:\Quota)) {

New-Item -Path C:\Quota -ItemType Directory |

Out-Null

}

If (-Not (Test-Path C:\QuotaS)) {

New-Item -Path C:\QuotaS -ItemType Directory |

Out-Null

}

1. Create an FSRM action for when the threshold is exceeded:

$Body = @'

User [Source Io Owner] has exceeded the [Quota Threshold]% quota threshold for the quota on [Quota Path] on server [Server].

The quota limit is [Quota Limit MB] MB, and [Quota Used MB] MB

currently is in use ([Quota Used Percent]% of limit).

'@

$NAHT = @{

Type = 'Email'

MailTo = 'Doctordns@gmail.Com'

Subject = 'FSRM Over limit [Source Io Owner]'

Body = $Body

}

$Action1 = New-FsrmAction @NAHT

1. Create an FSRM action for when the soft threshold is exceeded:

$Thresh = New-FsrmQuotaThreshold -Percentage 85 -Action $Action1

1. Create a soft 10 MB quota on the C:\Quotas folder with a threshold:

$NQHT1 = @{

Path = 'C:\QuotaS'

Template = 'Soft 5MB Limit'

Threshold = $Thresh

}

New-FsrmQuota @NQHT1

1. Now test the 85% soft quota limit on C:\QuotaS:

Get-ChildItem c:\quotas -Recurse | Remove-Item -Force

$S = '42'

1..24 | foreach {$s = $s + $s}

$S | Out-File -FilePath C:\QuotaS\Demos.txt

Get-ChildItem -Path C:\QuotaS\Demos.txt

1. Check if you received a notification email via Outlook or another mail client.
2. Create a second threshold action to log to the application log:

$Action2 = New-FsrmAction -Type Event -EventType Error

$Action2.Body = $Body

1. Create two quota thresholds for a new quota:

$Thresh2 = New-FsrmQuotaThreshold -Percentage 65

$Thresh3 = New-FsrmQuotaThreshold -Percentage 85

$Thresh2.Action = $Action2

$Thresh3.Action = $Action2 # same action details

1. Create a hard quota, with two thresholds and related threshold actions, based on an FSRM quota template:

$NQHT = @{

Path = 'C:\Quota'

Template = 'TenMB Limit'

Threshold = ($Thresh2, $Thresh3)

Description = 'Hard Threshold with2 actions'

}

New-FsrmQuota @NQHT

1. Remove existing files, if any:

Get-ChildItem C:\Quota -Recurse | Remove-Item -Force

1. Test a hard limit on C:\Quota from a different user:

$URK = "ThomasL@Reskit.Org"

$PRK = ConvertTo-SecureString 'Pa$$w0rd' -AsPlainText -Force

$CredRK = New-Object system.management.automation.PSCredential $URK,$PRK

$SB = {

$S = '42'

1..27 | foreach {$s = $s + $s}

$S | Out-File -FilePath C:\Quota\Demos.Txt -Encoding ascii

$Len = (Get-ChildItem -Path C:\Quota\Demos.Txt).Length}

$ICMHT = @{

ComputerName = 'SRV1'

Credential = $CredRK

ScriptBlock = $SB}

Invoke-Command @ICMHT

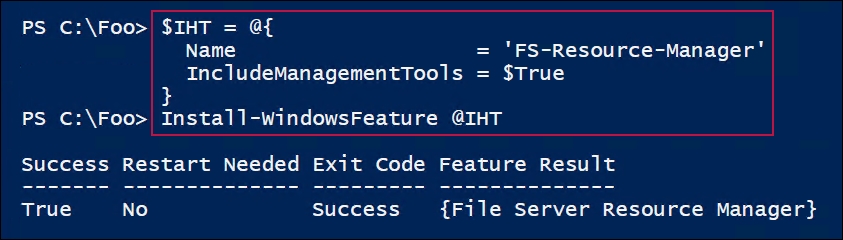
1. View the event log entries related to the overuse of the quota:

Get-EventLog -LogName Application -Source SRMSVC |

Format-Table -AutoSize -Wrap

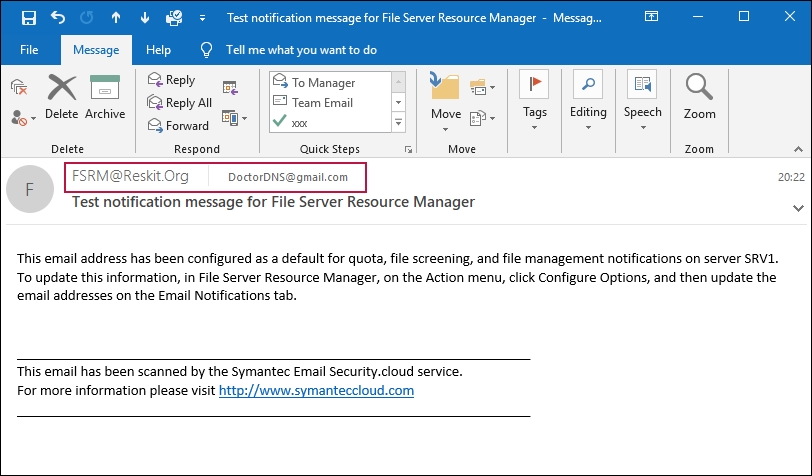
## How it works…

In step 1, you install the FSRM feature on SRV1, which looks like this:

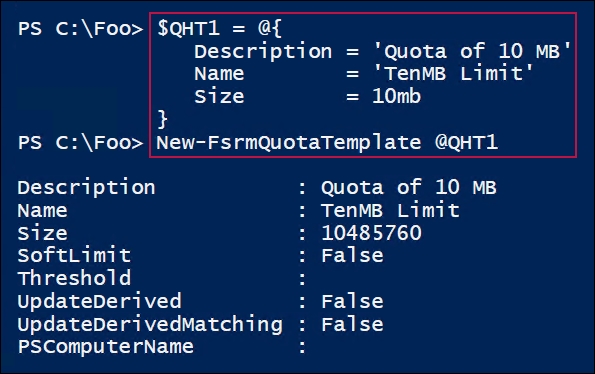


In step 2, you set SMTP server settings for FSRM, which generates no output. The assumption is that you've configured SRV1 to be an email forwarder, forwarding mail to SendMail.Com for onward transmission. This step produces no output.

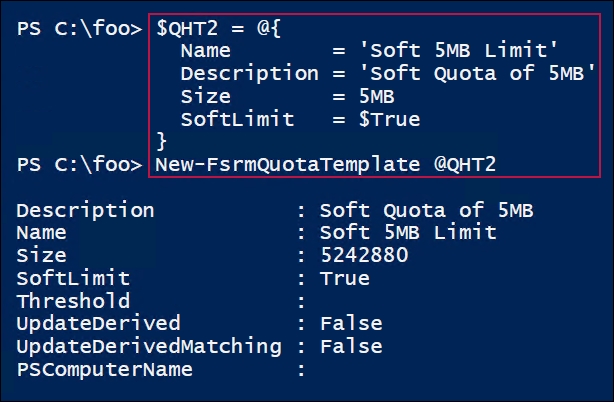
In step 3, you test the SMTP service by using the Send-FsrmTestEmail cmdlet. There's no output as such from this step, but the resultant email looks like this:



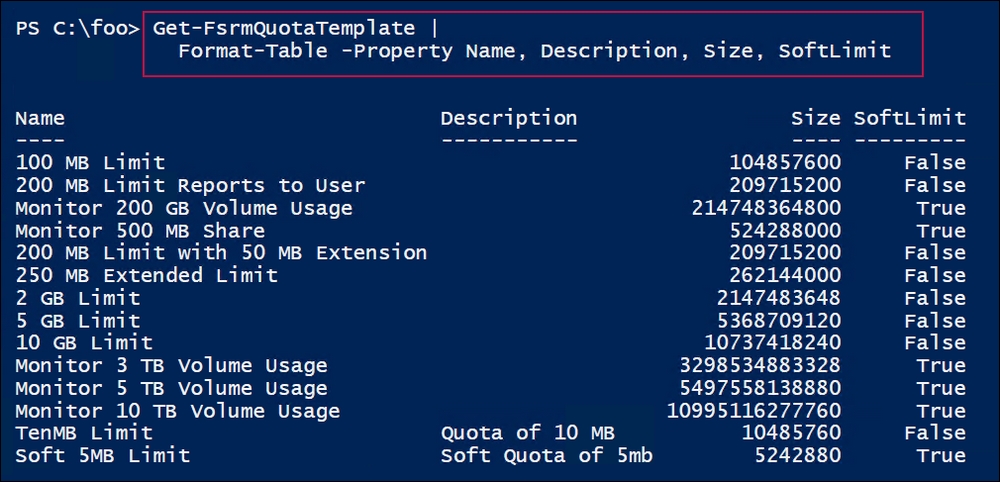
With step 4, you create an FSRM quota template, which looks like this:



In step 5, you create an additional FSRM quota template, this time for 5 MB, which looks like this:

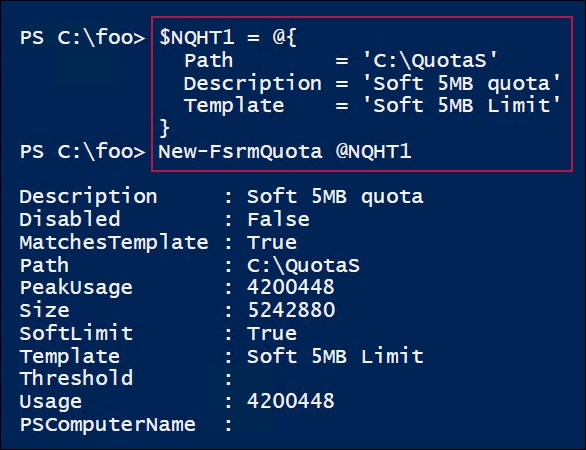


In step 6, you review the FSRM templates available on SRV1. This includes templates added when you installed the FSRM feature, plus the ones you created in step 4 and step 5. The available templates look like this:

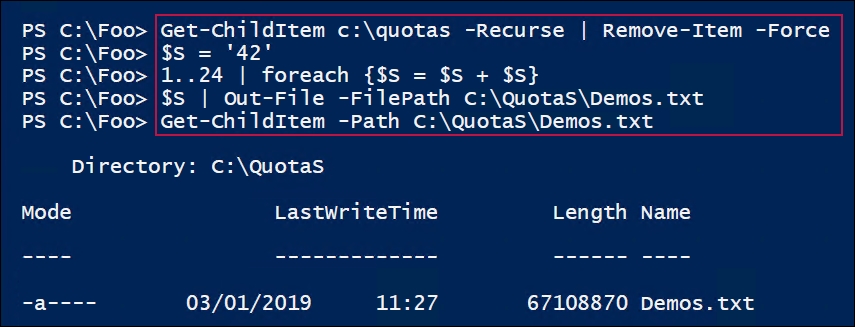


In step 7, you create two new folders on SRV1 to assist in testing soft and hard Filestore quotas. In step 8 and step 9, you create two new FSRM quota-exceeded actions. These three steps produce no output.

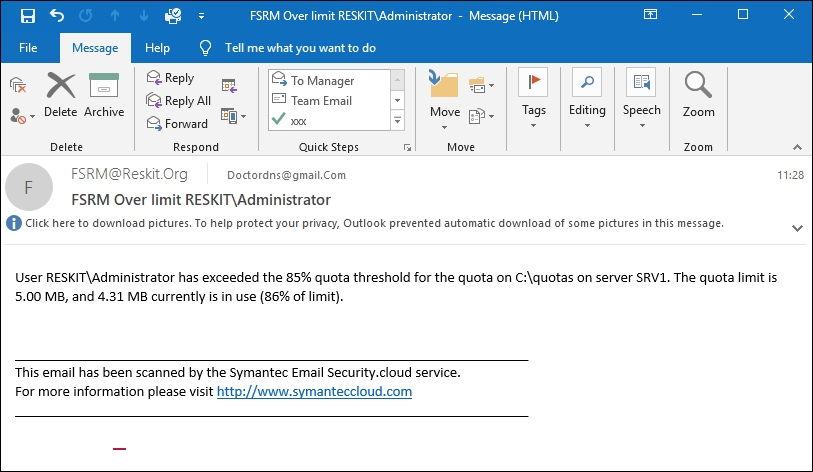
In step 10, you create a new soft quota on C:\QuotaS, which looks like this:



In step 11, you test the soft quota by building a large string and outputting the string to a file that exceeds the soft quota. The output from this step is as follows:

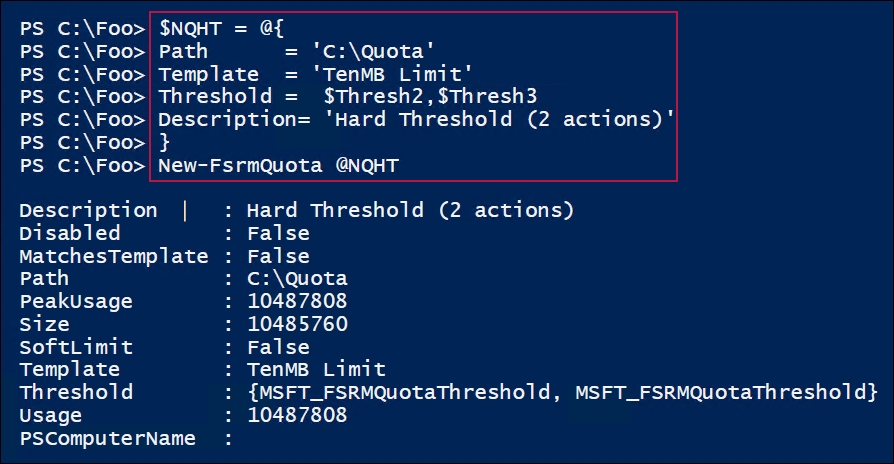


Exceeding the soft quota generates an email message, which looks like this:

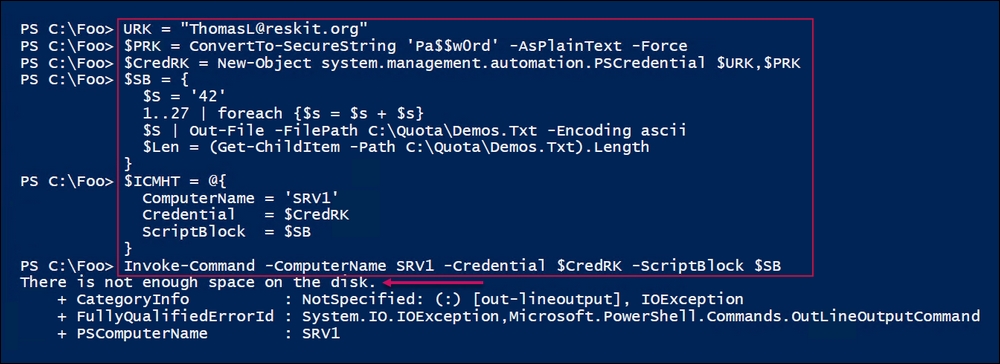


In step 13, you create a second threshold action to log to the application event log. In step 14, you create two new FSRM quota thresholds (for 65% of the quota exceeded, and 85% of the quota exceeded). These two steps produce no output.

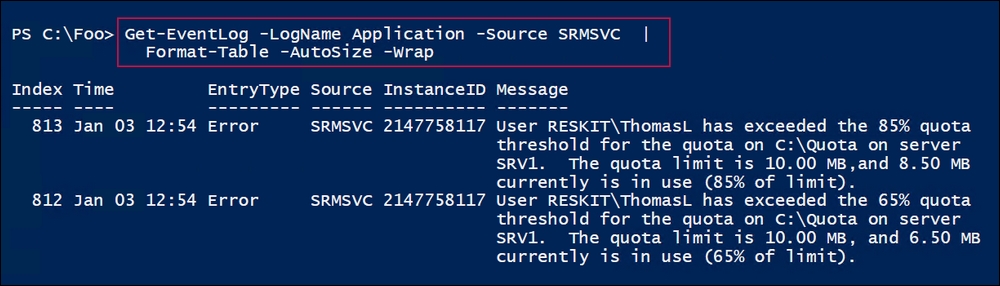
In step 15, you create a hard quota for C:\Quota, which has the two threshold actions you set in the two previous steps. The output from this step looks like this:



In step 16, you remove any existing files in C:\Quota, which produces no output. In step 17, you test the hard quota, which looks like this:



Finally, in step 18, you view the Application log events that FSRM logged when the two quota thresholds were exceeded (that is, when the quota threshold exceeded 65% and 85%). The output looks like this:



## There's more...

In this recipe, you set up and tested both a soft and a hard FSRM quota. With the soft quota, you configured FSRM to send an email to inform the recipient that a quota has been exceeded. With the hard quota, you logged two event-log messages (when the quota has been exceeded by 65% and 85%). While the soft quota means a user can exceed the quota, with a hard quota, the user can only save up to the quota limit. As you can see in step 17, the file saved in C:\Quota was limited to just 10 MB.

The quotas set in this recipe were extremely small and would probably not be of much use in production. But a simple change from, say, 10 MB to 10 GB, would be simple to make.

Also, for the soft quota, the quota exceeded the action results in the email being sent, while for the hard quota, FSRM just writes Application event-log entries. In production, you might want to send email to either or both an administrator and the user who has exceeded the quota thresholds.

In step 14, you create two quota thresholds (one invoked at 65%, and the second at 85%). For both thresholds, you apply the same text, which gets posted when either threshold is exceeded. You can see these two messages in step 18.

# Using filesystem reporting

A useful feature of the FSRM component is reporting. FSRM defines a number of basic report types that you can request. The reports can either be generated immediately (also known as interactive) or at a scheduled time. The latter causes FSRM to generate reports on a weekly or monthly basis.

## Getting ready

Run this recipe on SRV1, after installing the FSRM feature. You did this in the Managing Filestore quotas recipe. That recipe also created two largish files. If you haven't run that recipe, consider creating a few large files on SRV1 before running this recipe.

## How to do it...

1. Create a new interactive Storage Report for large files on C:\ on SRV1:

$NRHT = @{

Name = 'Large Files on SRV1'

NameSpace = 'C:\'

ReportType = 'LargeFiles'

LargeFileMinimum = 10MB

Interactive = $True

}

New-FsrmStorageReport @NRHT

1. Get the current FSRM reports:

Get-FsrmStorageReport \*

1. After the large file storage report is run, view the results in the filestore:

$Path = 'C:\StorageReports\Interactive'

Get-ChildItem -Path $Path

1. View the HTML report:

$Rep = Get-ChildItem -Path $path\\*.html

Invoke-item -Path $Rep

1. Extract key information from the XML:

$XF = Get-ChildItem -Path $Path\\*.xml # Find the XML file

$XML = [XML] (Get-Content -Path $XF) # Load file as XML

$Files = $XML.StorageReport.ReportData.Item # Get large files

$Files | Where-Object Path -NotMatch '^Windows|^Program|^Users'|

Format-Table -Property Name, Path,

@{Name ='Sizemb'

Expression = {(([int]$\_.size)/1mb).ToString('N2')}

},

DaysSinceLastAccessed -AutoSize

1. Create a monthly FSRM task in the task scheduler:

$Date = Get-Date

$NTHT = @{

Time = $Date

Monthly = 1

}

$Task = New-FsrmScheduledTask @NTHT

$NRHT = @{

Name = 'Monthly Files by files group report'

Namespace = 'C:\'

Schedule = $Task

ReportType = 'FilesbyFileGroup'

FileGroupINclude = 'text files'

LargeFileMinimum = 25MB

}

New-FsrmStorageReport @NRHT | Out-Null

1. Get the details of the task:

Get-ScheduledTask |

Where-Object Taskname -Match 'Monthly' |

Format-Table -AutoSize

1. Start the scheduled task:

Get-ScheduledTask -TaskName '\*Monthly\*' |

Start-ScheduledTask

Get-ScheduledTask -TaskName '\*Monthly\*'

1. View the report:

$Path = 'C:\StorageReports\Scheduled'

$Rep = Get-ChildItem -Path $path\\*.html

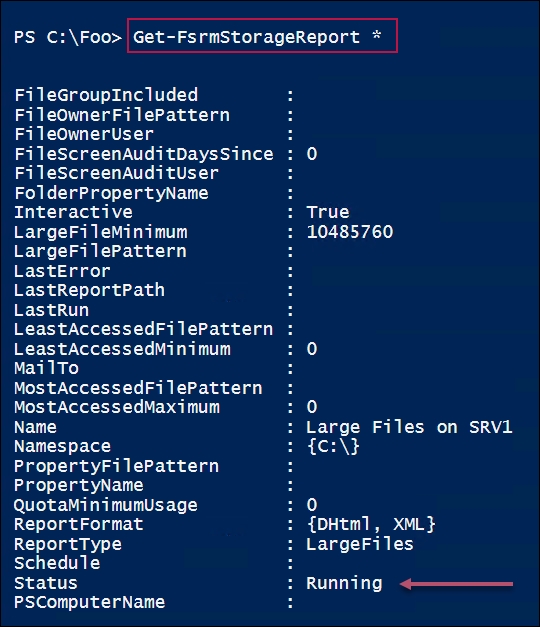
Invoke-item -Path $Rep

## How it works…

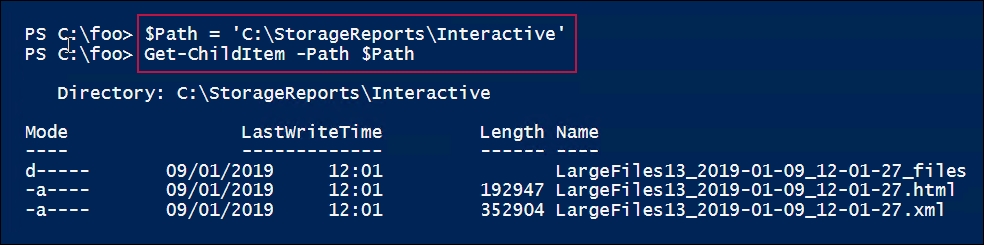
In step 1, you use the New-FsrmStorageReport to create a report to report any files over 10 MB on C:\, which generates the following output:



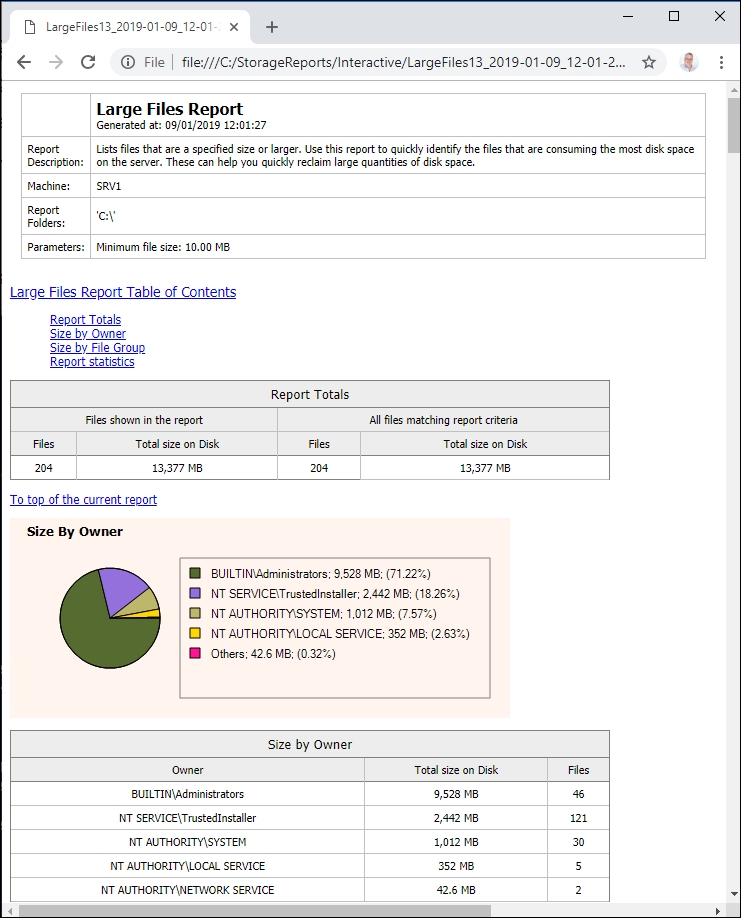
In step 2, you view the existing FSRM reports, which looks like this:



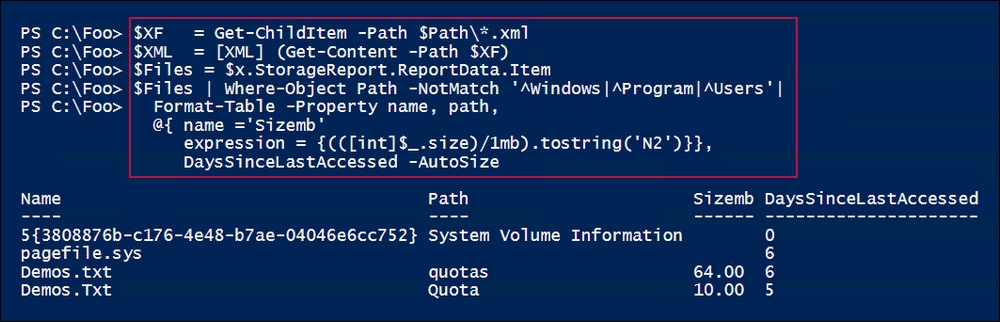
After the large file storage report is run, in step 3, you view the results in the filestore, which looks like this:



In step 4, you view the HTML report, which looks like this:



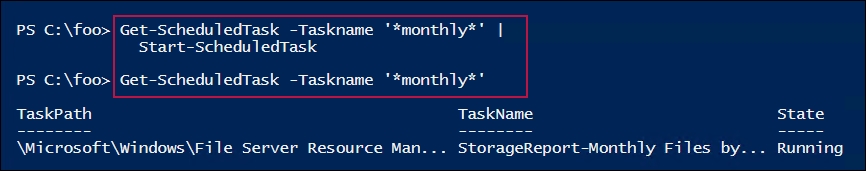
In step 5, you extract information from the XML version of the interactive large-file FSRM report, which looks like this:



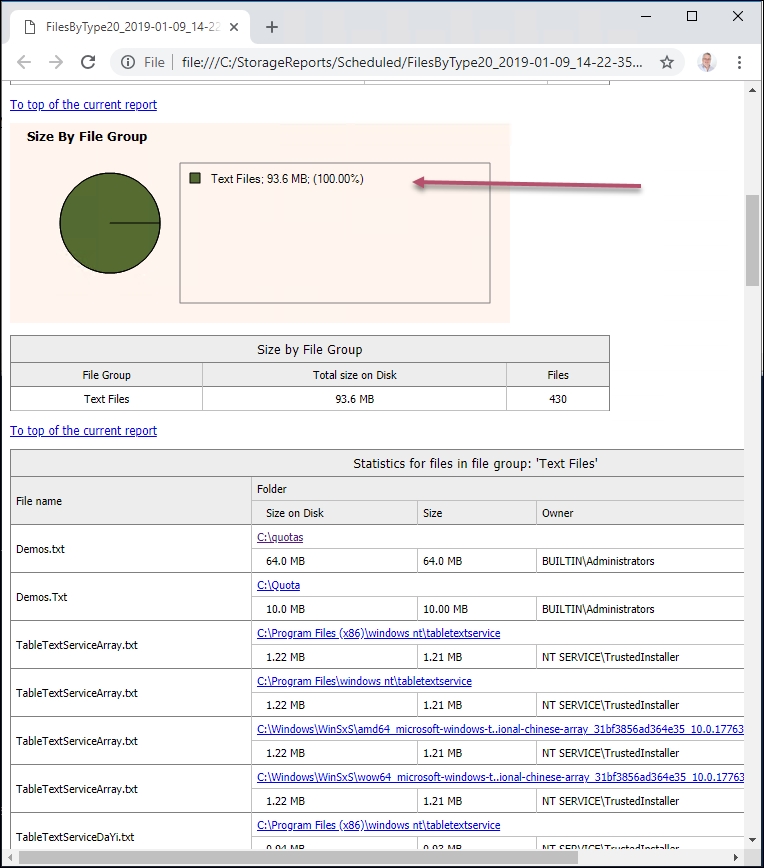
In step 6, you set up a scheduled report that produces no output. In step 7, you view the details of the report:



In step 8, you start the task you defined in step 7, and view that this report is running, which looks like this:



In step 9, you view the report of files by file groups, which looks like this:



## There's more...

In step 2, you view the current FSRM reports using the Get-FsrmStorageReport cmdlet. This shows all active FSRM reports (scheduled or interactive), although once FSRM completes an interactive report, the report is no longer visible using Get-FsrmStorageReport.

In step 3, you review the output FSRM generated for the interactive storage report. As you can see, there's both an HTML and an XML file. The HTML output looks good (as you see in step 4), but the format is fixed and can't be changed. However, the XML gives you all the information contained in the HTML report. You can use the XML to format the results exactly as you want. The XML file structure is pretty simple and you can pull out the basic information from the XML to create the details you might be interested in, as shown in step 5.

In step 9, you view the second report—a report on files by file group. In this case, the only file group reported is for text files. Were there other files belonging to other file groups, these would be reported.