Storage (for more cmdlets run 'gcm –module Storage')

Physical Disks	List all physical disks	**Get-PhysicalDisk**
	List all physical disks sorted from large to small	**Get-PhysicalDisk** | Sort Size -Descending
	List all physical disks that are currently associate with a pool	Get-StoragePool -IsPrimordial $false**| Get-PhysicalDisk**
Storage Pool	List all storage pools	**Get-StoragePool**
	Create a new storage pool named "Pool1" with 2 disks	$pDisks= Get-PhysicalDisk $s=Get-StorageSubSystem **New-StoragePool** -FriendlyName "Pool1" -PhysicalDisks $pDisks[0] , $pDisks[1] -StorageSubSystemFriendlyName $s.FriendlyName
Virtual Disk	Create a 100GB virtual disk using the Mirror resiliency type and thin provisioning	**New-VirtualDisk** -FriendlyName "VD1" -StoragePoolFriendlyName "Pool1" -Size 100GB -ResiliencySettingName "Mirror" -ProvisioningType Thin
	Create the biggest possible virtual disk using the Parity resiliency type and fixed provisioning. Simple (no resiliency) is another resiliency choice.	**New-VirtualDisk** -FriendlyName "VD1" -StoragePoolFriendlyName "Pool1" -ResiliencySettingName "Parity" -UseMaximumSize
	Remove (delete) a virtual disk named "VD1"	**Remove-VirtualDisk** -FriendlyName "VD1"
Disk	List all disks visible to Windows	**Get-Disk**
Volume	Initialize a disk with a GPT partition on the new virtual disk	Get-VirtualDisk -FriendlyName "VD1" | get-disk | **Initialize-Disk** -PartitionStyle "MBR"
	Create a 50GB Partition and assign the drive letter P	Get-VirtualDisk -FriendlyName "VD1" | get-disk | **New-Partition** -Size 50GB -DriveLetter P
	Format drive P with the NTFS file system and the label "Data"	**Format-Volume** -DriveLetter P -FileSystem NTFS -NewFileSystemLabel "Data"
	Format an ReFS volume and skip confirmation	**Format-Volume** -DriveLetter P -FileSystem ReFS -Confirm:$false

Shares (for more cmdlets run 'gcm –module SMBShare' and 'gcm –module NFS')

SMB Shares	Get All SMB shares	**Get-smbshare**
	Create a new SMB share named "SmbShare1" on path "C:\Shares\SmbShare1" with full access granted to "Contoso\User1"	**New-SmbShare -Name** "SmbShare1" **-**Path "c:\Shares\SmbShare1" -FullAccess "Contoso\User1"
	Remove the SMB share "SmbShare1" from the server	**Remove-SmbShare -Name** "SmbShare1"
SMB Server	Get SMB server configurations	**Get-SmbClientConfiguration**
NFS Shares	Get All NFS shares	**Get-NFSShare**
	Create a new NFS Share named NFSShare1 with "Krb5" authentication and Root access denied	**New-NfsShare** -Name "NfsShare1" -Path "c:\NFS1" -Authentication krb5 -AllowRootAccess $false
	Remove the share "NfsShare1" from the machine	**Remove-NfsShare** -Name "NfsShare1"
NFS Server	Get NFS server configuration	**Get-NfsServerConfig**
	Create a new client group "ClientGrp1" and add "localhost" as member of the client group	**New-NfsClientgroup** -ClientGroupName "ClientGrp1" -AddMember "localhost"

File Server Resource Manager (for more cmdlets run 'gcm –module FileServerResourceManager)

File Management Tasks	Create a file management task that moves old (expired) HR files to a designated directory	$action = N**ew-FsrmFmjAction** -ExpirationFolder "c:\Expired" -Type "Expiration" $sched = **New-FsrmScheduledTask** -Time "21:59" -Weekly "Sunday" $condition = **New-FsrmFmjCondition** -Condition "LessThan" -DateOffset "-360" -Property File.DateCreated -Value Date.Now **New-FsrmFileManagementJob** -Name "Expiration11" -Action $action -Schedule $sched -Condition $condition -Namespace "c:\Shares\HR"
Quotas	Get the specified quota template	$QuotaTemplate = **Get-FsrmQuotaTemplate** -Name "Monitor 500 MB Share"
	Create a quota that applies the specified quota template	**New-FsrmQuota** -Path "c:\Shares\HR" -Template $QuotaTemplate.Name
Storage Reports	Create an interactive storage report on large files and list the contents	$Report = **New-FsrmStorageReport** -Name "LargeFilesReport" -LargeFileMinimum 100MB -ReportFormat "Text" -Namespace "c:\Shares\HR " -ReportType "LargeFiles" -interactive | Wait-FsrmStorageReport **Get-ChildItem** -Path $Report.LastReportPath -Recurse | where { $_.Name -like "Large*.txt"} | Get-Content
Access-Denied Assistance	Enables customized access denied messages for shares on this server and allows access requests to be sent to the server admin and share owner	**Set-FsrmAdrSetting** -Event "AccessDenied" -Enabled:$true -DisplayMessage "Access to the share is restricted" -AllowRequests:$true -EmailMessage "Please grant me access to the share." -MailTo JB@contoso.com -MailToOwner:$true

Data Deduplication (for more cmdlets run 'gcm –module Deduplication')

Volume	Get all Dedup volumes	**Get-DedupVolume**
	Enable deduplication on a volume	**Enable-DedupVolume** P:
	Set deduplication volume configuration	**Set-DedupVolume** P: -ExcludeFileType "txt", "log" -MinimumFileAgeDays 10
	Get volume status	**Get-DedupStatus** P:
Schedule	Get scheduled jobs	**Get-DedupSchedule**
	Set Optimization Schedule	**Set-DedupSchedule** ThroughputOptimization-2 -Type Optimization -Start "2:00 AM" -Days "Sunday","Saturday"
Jobs	Start an optimization job	**Start-DedupJob** P: -Type Optimization
	Get all running jobs	**Get-DedupJob**

iSCSI Target Server (for more cmdlets run 'gcm –module iSCSITarget')

iSCSI Virtual Disk	Create a new 400 MB iSCSI virtual disk	**New-IscsiVirtualDisk** -Path "c:\Iscsi\Data\SQL1.vhd" -size 400MB
	Get all iSCSI virtual disks	**Get-IscsiVirtualDisk**
	Remove an iSCSI virtual disk	**Remove-IscsiVirtualDisk** –Path "c:\Iscsi\Data\SQL1.vhd"
Server Target	Create a new iSCSI target	**New-IscsiServerTarget** -TargetName "SQLDisks"
	Unmask the target initiators (with IQN or DNS name list)	**Set-IscsiServerTarget** -TargetName "SQLDisks" -initiatorId "IQN: iqn.1991-05.com.contoso:SQL1.contoso.com", "DNSNAME:SQL2.contoso.com"
	Get all iSCSI targets	**Get-iscsiTarget**
	Remove a target	**Remove-IscsiServerTarget** –TargetName "SQLDisks"
Mapping	Map an iSCSI target to a VHD	**Add-IscsiVirtualDiskTargetMapping** -Path "c:\Iscsi\Data\SQL1.vhd" -TargetName SQLDisks
Sessions	Get all connected LUNs	**Get-IscsiServerTarget** | where {$_.Status -eq "Connected"} | fl LunMappings

iSCSI Initiator (for more cmdlets run 'gcm –module iSCSI')

Target	Discover the iSCSI Target Server	$Target=**New-iSCSITargetPortal** -TargetPortalAddress "10.20.30.40"
Portal	Get the portal object	$Portal **= Get-iSCSITargetPortal**
Connect	Connect to a target	**Connect-iSCSITarget** -NodeAddress $Target.NodeAddress
Session	Get the current active sessions	**Get-IscsiSession** | where IsConnected -eq $true
	Make the iSCSI connection persistent	**Register-IscsiSession** -SessionIdentifier (Get-IscsiSession)[1].SessionIdentifier

Failover Clustering (for more cmdlets run 'gcm –module FailoverClusters')

Create a new cluster	**New-Cluster –**Name "Clu1" **–**Node "N1","N2"
Assign cluster disks	Get-ClusterAvailableDisk | **Add-ClusterDisk**
Add a new File Server role	**Add-ClusterFileServerRole** –Name FSCluster –Storage "Cluster Disk 1"

DFS Namespaces (for more cmdlets run 'gcm –module DFSN')

Name spaces	Create a new stand-alone based namespace	**New-DFSNamespace** -NamespaceRootTarget \\FSServer -Type Standalone -NamespaceRoot \\FSServer\Share1
	Get server config	**Get-DFSNamespaceServerConfig** -NamespaceServer \\FS1