使用 PowerShell 建立、強化及驗證 ONTAP 網路資料保險箱

NetApp Solutions

NetApp
September 26, 2024

目錄

使用 PowerShell 建立、強化及驗證 ONTAP 網路資料保險箱

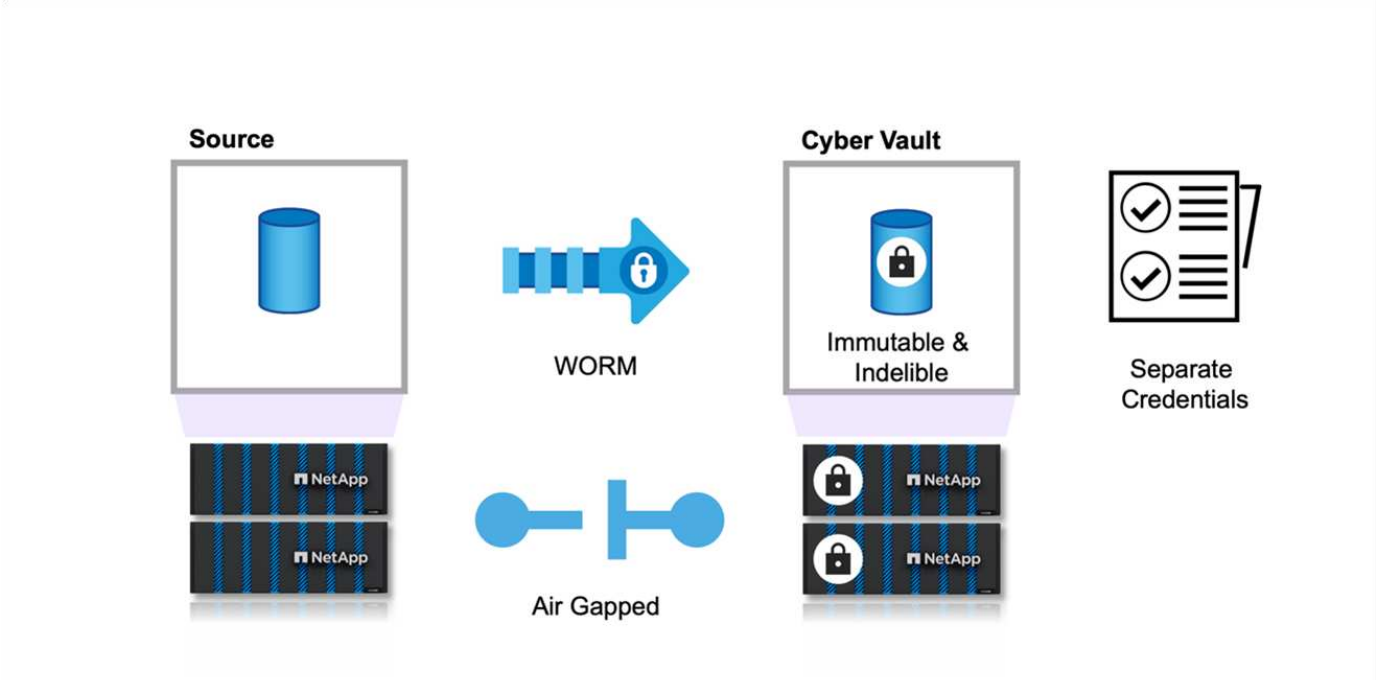
使用 PowerShell 的 ONTAP 網路資料保險箱總覽

在現今的數位環境中、保護組織的重要資料資產不只是最佳實務做法、更是企業的當務之急。網路威脅正以前所未有的速度進化、而傳統的資料保護措施已不再足以保護敏感資訊的安全。這就是網路資料保險箱的誕生所在。 NetApp 最先進的 ONTAP 型解決方案結合了先進的通風技術與強大的資料保護措施、可建立無法滲透的網路威脅屏障。網路資料保險箱利用安全強化技術隔離最寶貴的資料、將攻擊面降到最低、讓最關鍵的資料保持機密、完整無缺、並在需要時隨時可用。

網路資料保險箱是一種安全的儲存設備、包含多層保護、例如防火牆、網路和儲存設備。這些元件可保護關鍵業務營運所需的重要恢復資料。根據資料保險箱原則、網路資料保險箱的元件會定期與重要的正式作業資料同步、但否則將無法存取。這種隔離且中斷連線的設定可確保在網路攻擊危及正式作業環境的情況下、可從網路資料保險箱輕鬆進行可靠且最終的還原。

NetApp 可設定網路、停用生命體、更新防火牆規則、以及將系統與外部網路和網際網路隔離、輕鬆建立網路資料保險箱的空空缺。這種穩健的方法可有效中斷系統與外部網路和網際網路的連線、提供無與倫比的保護、防範遠端網路攻擊和未經授權的存取嘗試、使系統免受網路型威脅和入侵。

將此功能與 SnapLock Compliance 保護相結合、即使是 ONTAP 管理員或 NetApp 支援人員也無法修改或刪除資料。SnapLock 會定期根據 SEC 和 FINRA 法規進行稽核、確保資料恢復能力符合銀行業嚴格的 WORM 和資料保留法規。NetApp 是唯一經過 NSA CSfC 驗證、可用來儲存機密資料的企業儲存設備。

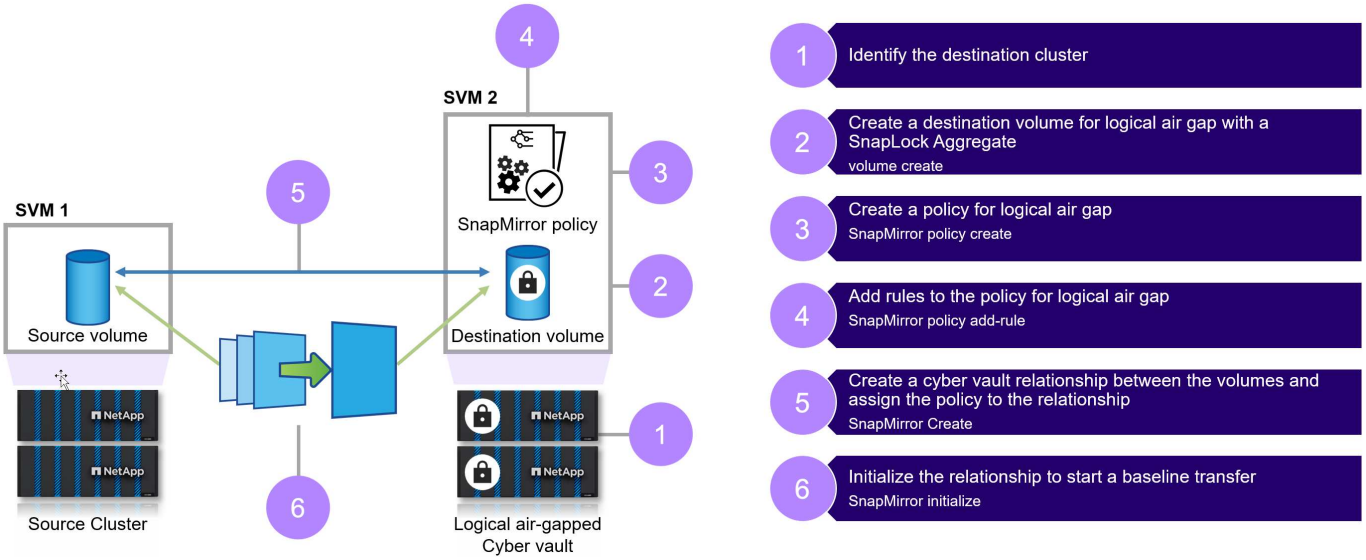


本文件說明 NetApp 的網路儲存庫在內部部署 ONTAP 儲存設備的自動組態、以及另一個指定的 ONTAP 儲存設備、其中包含不可變快照、可增加額外的保護層、避免網路攻擊不斷增加、以利快速恢復。作爲此架構的一部分，將根據 ONTAP 最佳實踐應用整個配置。最後一節說明如何在發生攻擊時執行恢復。

同樣的解決方案也適用於使用適用於 ONTAP 的 FSX 在 AWS 中建立指定的網路資料保險箱。

建立 ONTAP 網路資料保險箱的高階步驟

- 建立對等關係
  - 使用 ONTAP 儲存設備的正式作業站台會與指定的網路資料保險箱 ONTAP 儲存設備進行對等連接
- 建立 SnapLock Compliance Volume
- 設定 SnapMirror 關係和規則以設定標籤
  - 已設定 SnapMirror 關係和適當的排程
- 在初始化 SnapMirror （資料保險箱）傳輸之前設定保留
  - 保留鎖定會套用到複製的資料、進一步防止資料遭到內部人員或資料故障的影響。使用此選項時、資料將無法在保留期間到期之前刪除
  - 組織可以根據需求、將這些資料保留數週 / 月
- 根據標籤初始化 SnapMirror 關係
  - 根據 SnapMirror 排程、進行初始植入和永久遞增傳輸
  - SnapLock Compliance 可保護資料（不可變且無法磨滅）、而且資料可用於恢復
- 實作嚴格的資料傳輸控制
  - 網路資料保險箱會在有限的一段期間內解除鎖定、並會與資料保險箱中的資料同步。傳輸完成後、連線會中斷連線、關閉並再次鎖定
- 快速恢復
  - 如果主要資料在正式作業站台中受到影響、則網路資料保險箱中的資料會安全地還原至原始正式作業環境或其他選擇的環境



解決方案元件

在來源叢集和目的地叢集上執行 9.15.1 的 NetApp ONTAP 。

ONTAP One ： NetApp ONTAP 的 All-In-One 授權。

ONTAP One 授權所使用的功能：

- 符合法規SnapLock
- SnapMirror
- 多管理員驗證
- ONTAP 公開的所有強化功能
- 為網路資料保險箱提供獨立的 RBAC 認證

所有 ONTAP 統一化實體陣列都可用於網路資料保險箱、但 AFF C 系列容量型 Flash 系統和 FAS 混合式 Flash 系統是最具成本效益的理想平台。請參閱"ONTAP 網路資料保險箱規模調整"以取得尺寸指南。

使用 PowerShell 建立 ONTAP 網路資料保險箱

使用傳統方法的氣載備份需要創造空間、並將主要媒體和次要媒體實體分開。透過將媒體移出站台及 / 或中斷連線、不良使用者將無法存取資料。這樣可以保護資料、但可能會導致恢復時間變慢。使用 SnapLock Compliance 時、不需要實體隔離。SnapLock Compliance 可保護資料保險箱快照時間點的唯讀複本、使資料能夠快速存取、不受刪除或無法刪除的影響、而且不會遭到修改或不可變的影響。

先決條件

在開始執行本文件下一節中的步驟之前、請確定符合下列先決條件：

- 來源叢集必須執行 ONTAP 9 或更新版本。
- 來源與目的地集合體必須為64位元。
- 來源叢集和目的地叢集必須執行對等關係。
- 必須對來源和目的地 SVM 進行對等處理。
- 確保已啟用叢集對等加密。

設定資料傳輸至 ONTAP 網路資料保險箱需要幾個步驟。在主要磁碟區上、設定快照原則、指定要建立的複本、以及使用適當排程建立複本的時間、並指派標籤以指定 SnapVault 應傳輸哪些複本。在次要系統上、必須建立 SnapMirror 原則、指定要傳輸的 Snapshot 複本標籤、以及網路資料保險箱中應保留多少這些複本。設定這些原則之後、請建立 SnapVault 關係並建立傳輸排程。

本文件假設主要儲存設備和指定的 ONTAP 網路保存庫已設定及設定完成。

網路資料保險箱叢集可以與來源資料位於相同或不同的資料中心。

建立 ONTAP 網路資料保險箱的步驟

1. 使用 ONTAP CLI 或系統管理員來初始化規範時鐘。

2. 在啟用 SnapLock Compliance 的情況下建立資料保護磁碟區。
3. 使用 SnapMirror create 命令建立 SnapVault 資料保護關係。
4. 設定目的地 Volume 的預設 SnapLock Compliance 保留期間。

> 預設保留為「設為最小」。做為保存目的地的流通量會指派預設保留期間給它。SnapLock此期間的值最初設定為 SnapLock Compliance 磁碟區的最少 0 年和最長 30 年。每個NetApp Snapshot複本一開始就會提交此預設保留期間。如果需要、可在稍後延長保留期間、但不可縮短。

以上包含手動步驟。安全專家建議將程序自動化、以避免手動管理造成重大錯誤。以下程式碼片段可完全自動化 SnapLock Compliance 的先決條件和組態、並可初始化時鐘。

以下是初始化 ONTAP 規範時鐘的 PowerShell 程式碼範例。

```
function initializeSnapLockComplianceClock {
    try {
        $nodes = Get-NcNode

        $isInitialized = $false
        logMessage -message "Cheking if snaplock compliance clock is
initialized"
        foreach($node in $nodes) {
            $check = Get-NcSnaplockComplianceClock -Node $node.Node
            if ($check.SnaplockComplianceClockSpecified -eq "True") {
                $isInitialized = $true
            }
        }

        if ($isInitialized) {
            logMessage -message "SnapLock Compliance clock already
initialized" -type "SUCCESS"
        } else {
            logMessage -message "Initializing SnapLock compliance clock"
            foreach($node in $nodes) {
                Set-NcSnaplockComplianceClock -Node $node.Node
            }
            logMessage -message "Successfully initialized SnapLock
Compliance clock" -type "SUCCESS"
        }
    } catch {
        handleError -errorMessage $_.Exception.Message
    }
}
```

以下是設定 ONTAP 網路資料保險箱的 PowerShell 程式碼範例。

```
function configureCyberVault {
    for($i = 0; $i -lt $DESTINATION_VOLUME_NAMES.Length; $i++) {
        try {
            # checking if the volume already exists and is of type
snaplock compliance
            logMessage -message "Checking if SnapLock Compliance volume
$($DESTINATION_VOLUME_NAMES[$i]) already exists in vServer
$DESTINATION_VSERVER"
            $volume = Get-NcVol -Vserver $DESTINATION_VSERVER -Volume
$DESTINATION_VOLUME_NAMES[$i] | Select-Object -Property Name, State,
TotalSize, Aggregate, Vserver, Snaplock | Where-Object { $_.Snaplock.Type
-eq "compliance" }
            if($volume) {
                $volume
                logMessage -message "SnapLock Compliance volume
$($DESTINATION_VOLUME_NAMES[$i]) already exists in vServer
$DESTINATION_VSERVER" -type "SUCCESS"
            } else {
                # Create SnapLock Compliance volume
                logMessage -message "Creating SnapLock Compliance volume:
$($DESTINATION_VOLUME_NAMES[$i])"
                New-NcVol -Name $DESTINATION_VOLUME_NAMES[$i] -Aggregate
$DESTINATION_AGGREGATE_NAMES[$i] -SnaplockType Compliance -Type DP -Size
$DESTINATION_VOLUME_SIZES[$i] -ErrorAction Stop | Select-Object -Property
Name, State, TotalSize, Aggregate, Vserver
                logMessage -message "Volume $($DESTINATION_VOLUME_NAMES[
$i]) created successfully" -type "SUCCESS"
            }

            # Set SnapLock volume attributes
            logMessage -message "Setting SnapLock volume attributes for
volume: $($DESTINATION_VOLUME_NAMES[$i])"
            Set-NcSnaplockVolAttr -Volume $DESTINATION_VOLUME_NAMES[$i]
-MinimumRetentionPeriod $SNAPLOCK_MIN_RETENTION -MaximumRetentionPeriod
$SNAPLOCK_MAX_RETENTION -ErrorAction Stop | Select-Object -Property Type,
MinimumRetentionPeriod, MaximumRetentionPeriod
            logMessage -message "SnapLock volume attributes set
successfully for volume: $($DESTINATION_VOLUME_NAMES[$i])" -type "SUCCESS"

            # checking snapmirror relationship
            logMessage -message "Checking if SnapMirror relationship
exists between source volume $($SOURCE_VOLUME_NAMES[$i]) and destination
SnapLock Compliance volume $($DESTINATION_VOLUME_NAMES[$i])"
            $snapmirror = Get-NcSnapmirror | Select-Object SourceCluster,
SourceLocation, DestinationCluster, DestinationLocation, Status,
MirrorState | Where-Object { $_.SourceCluster -eq
```

```
$SOURCE_ONTAP_CLUSTER_NAME -and $_.SourceLocation -eq "$($SOURCE_VSERVER)
:$($SOURCE_VOLUME_NAMES[$i])" -and $_.DestinationCluster -eq
$DESTINATION_ONTAP_CLUSTER_NAME -and $_.DestinationLocation -eq "
$($DESTINATION_VSERVER):$($DESTINATION_VOLUME_NAMES[$i])" -and ($_.Status
-eq "snapmirrored" -or $_.Status -eq "uninitialized") }
            if($snapmirror) {
                $snapmirror
                logMessage -message "SnapMirror relationship already
exists for volume: $($DESTINATION_VOLUME_NAMES[$i])" -type "SUCCESS"
            } else {
                # Create SnapMirror relationship
                logMessage -message "Creating SnapMirror relationship for
volume: $($DESTINATION_VOLUME_NAMES[$i])"
                New-NcSnapmirror -SourceCluster $SOURCE_ONTAP_CLUSTER_NAME
-SourceVserver $SOURCE_VSERVER -SourceVolume $SOURCE_VOLUME_NAMES[$i]
-DestinationCluster $DESTINATION_ONTAP_CLUSTER_NAME -DestinationVserver
$DESTINATION_VSERVER -DestinationVolume $DESTINATION_VOLUME_NAMES[$i]
-Policy $SNAPMIRROR_PROTECTION_POLICY -Schedule $SNAPMIRROR_SCHEDULE
-ErrorAction Stop | Select-Object -Property SourceCluster, SourceLocation,
DestinationCluster, DestinationLocation, Status, Policy, Schedule
                logMessage -message "SnapMirror relationship created
successfully for volume: $($DESTINATION_VOLUME_NAMES[$i])" -type "SUCCESS"
            }

        } catch {
            handleError -errorMessage $_.Exception.Message
        }
    }
}
```

1. 完成上述步驟後、即可使用 SnapLock Compliance 和 SnapVault 的無線網路保存庫即已就緒。

將快照資料傳輸至網路資料保險箱之前、必須先初始化 SnapVault 關係。不過、在此之前、您必須執行安全性強化、以保護資料保險箱的安全。

使用 PowerShell 強化 ONTAP 網路資料保險箱

相較於傳統解決方案、 ONTAP 網路資料保險箱可提供更好的網路攻擊恢復能力。設計架構以強化安全性時、必須考慮採取措施來減少攻擊面。這可以透過各種方法達成、例如實作強化的密碼原則、啟用 RBAC 、鎖定預設使用者帳戶、設定防火牆、以及利用核准流程來變更資料保險箱系統。此外、限制特定 IP 位址的網路存取通訊協定有助於限制潛在的弱點。

ONTAP 提供一組控制項、可強化 ONTAP 儲存設備。使用"ONTAP 的指引與組態設定"協助組織達成資訊系統機密性、完整性和可用度等規定的安全目標。

強化最佳實務做法

手動步驟

1. 建立具有預先定義及自訂管理角色的指定使用者。
2. 建立新的 IPspace 來隔離網路流量。
3. 在新的 IPspace 中建立新的 SVM 。
4. 確保正確設定防火牆路由原則、並視需要定期稽核及更新所有規則。

ONTAP CLI 或透過自動化指令碼

1. 使用多重管理驗證（ MFA ）來保護管理
2. 啟用叢集間標準資料「在線中」的加密。
3. 使用強式加密密碼來保護 SSH 安全、並強制執行安全密碼。
4. 啟用全域 FIPS 。
5. 應停用 Telnet 和遠端 Shell （ RSH ）。
6. 鎖定預設管理帳戶。
7. 停用資料生命和安全的遠端存取點。
8. 停用及移除未使用或無關的通訊協定和服務。
9. 加密網路流量。
10. 設定超級使用者和管理角色時、請使用最低權限原則。
11. 使用允許的 IP 選項、限制 HTTPS 和 SSH 來自特定 IP 位址。
12. 根據傳輸排程來關閉及恢復複寫。

項目符號 1-4 需要手動介入、例如指定隔離的網路、隔離 IPspace 等、而且必須事先執行。如需設定強化的詳細資訊"ONTAP 安全強化指南"，請參閱。其餘的可輕鬆自動化、以便輕鬆部署和監控。這種協調方法的目標是提供一種機制來自動化強化步驟、以供未來驗證資料保險箱控制器。網路資料保險箱的空缺開放時間範圍越短越好。SnapVault 運用遞增的 Forever 技術、只會將上次更新後的變更移至網路資料保險箱、因此可將網路資料保險箱必須保持開啟的時間減至最低。為了進一步最佳化工作流程、網路資料保險箱的開啟會與複寫排程協調、以確保最小的連線時間。

以下是強化 ONTAP 控制器的 PowerShell 程式碼範例。

```
function removeSvmDataProtocols {
    try {

        # checking NFS service is disabled
        logMessage -message "Checking if NFS service is disabled on
vServer $DESTINATION_VSERVER"
        $nfsService = Get-NcNfsService
        if($nfsService) {
            # Remove NFS
            logMessage -message "Removing NFS protocol on vServer :
```

```
$DESTINATION_VSERVER"
            Remove-NcNfsService -VserverContext $DESTINATION_VSERVER
-Confirm:$false
            logMessage -message "NFS protocol removed on vServer :
$DESTINATION_VSERVER" -type "SUCCESS"
        } else {
            logMessage -message "NFS service is disabled on vServer
$DESTINATION_VSERVER" -type "SUCCESS"
        }

        # checking CIFS/SMB server is disabled
        logMessage -message "Checking if CIFS/SMB server is disabled on
vServer $DESTINATION_VSERVER"
        $cifsServer = Get-NcCifsServer
        if($cifsServer) {
            # Remove SMB/CIFS
            logMessage -message "Removing SMB/CIFS protocol on vServer :
$DESTINATION_VSERVER"
            $domainAdministratorUsername = Read-Host -Prompt "Enter Domain
administrator username"
            $domainAdministratorPassword = Read-Host -Prompt "Enter Domain
administrator password" -AsSecureString
            $plainPassword = [Runtime.InteropServices.Marshal
]::PtrToStringAuto([Runtime.InteropServices.Marshal]::SecureStringToBSTR($
domainAdministratorPassword))
            Remove-NcCifsServer -VserverContext $DESTINATION_VSERVER
-AdminUsername $domainAdministratorUsername -AdminPassword $plainPassword
-Confirm:$false -ErrorAction Stop
            logMessage -message "SMB/CIFS protocol removed on vServer :
$DESTINATION_VSERVER" -type "SUCCESS"
        } else {
            logMessage -message "CIFS/SMB server is disabled on vServer
$DESTINATION_VSERVER" -type "SUCCESS"
        }

        # checking iSCSI service is disabled
        logMessage -message "Checking if iSCSI service is disabled on
vServer $DESTINATION_VSERVER"
        $iscsiService = Get-NcIscsiService
        if($iscsiService) {
            # Remove iSCSI
            logMessage -message "Removing iSCSI protocol on vServer :
$DESTINATION_VSERVER"
            Remove-NcIscsiService -VserverContext $DESTINATION_VSERVER
-Confirm:$false
            logMessage -message "iSCSI protocol removed on vServer :
```

```
$DESTINATION_VSERVER" -type "SUCCESS"
        } else {
            logMessage -message "iSCSI service is disabled on vServer
$DESTINATION_VSERVER" -type "SUCCESS"
        }

        # checking FCP service is disabled
        logMessage -message "Checking if FCP service is disabled on
vServer $DESTINATION_VSERVER"
        $fcpService = Get-NcFcpService
        if($fcpService) {
            # Remove FCP
            logMessage -message "Removing FC protocol on vServer :
$DESTINATION_VSERVER"
            Remove-NcFcpService -VserverContext $DESTINATION_VSERVER
-Confirm:$false
            logMessage -message "FC protocol removed on vServer :
$DESTINATION_VSERVER" -type "SUCCESS"
        } else {
            logMessage -message "FCP service is disabled on vServer
$DESTINATION_VSERVER" -type "SUCCESS"
        }

    } catch {
        handleError -errorMessage $_.Exception.Message
    }
}

function disableSvmDataLifs {
    try {
        logMessage -message "Finding all data lifs on vServer :
$DESTINATION_VSERVER"
        $dataLifs = Get-NcNetInterface -Vserver $DESTINATION_VSERVER |
Where-Object { $_.Role -contains "data_core" }
        $dataLifs | Select-Object -Property InterfaceName, OpStatus,
DataProtocols, Vserver, Address

        logMessage -message "Disabling all data lifs on vServer :
$DESTINATION_VSERVER"
        # Disable the filtered data LIFs
        foreach ($lif in $dataLifs) {
            $disableLif = Set-NcNetInterface -Vserver $DESTINATION_VSERVER
-Name $lif.InterfaceName -AdministrativeStatus down -ErrorAction Stop
            $disableLif | Select-Object -Property InterfaceName, OpStatus,
DataProtocols, Vserver, Address
        }
```

```
        logMessage -message "Disabled all data lifs on vServer :
$DESTINATION_VSERVER" -type "SUCCESS"

    } catch {
        handleError -errorMessage $_.Exception.Message
    }
}

function configureMultiAdminApproval {
    try {

        # check if multi admin verification is enabled
        logMessage -message "Checking if multi-admin verification is
enabled"
        $maaConfig = Invoke-NcSsh -Name $DESTINATION_ONTAP_CLUSTER_MGMT_IP
-Credential $DESTINATION_ONTAP_CREDS -Command "set -privilege advanced;
security multi-admin-verify show"
        if ($maaConfig.Value -match "Enabled" -and $maaConfig.Value -match
"true") {
            $maaConfig
            logMessage -message "Multi-admin verification is configured
and enabled" -type "SUCCESS"
        } else {
            logMessage -message "Setting Multi-admin verification rules"
            # Define the commands to be restricted
            $rules = @(
                "cluster peer delete",
                "vserver peer delete",
                "volume snapshot policy modify",
                "volume snapshot rename",
                "vserver audit modify",
                "vserver audit delete",
                "vserver audit disable"
            )
            foreach($rule in $rules) {
                Invoke-NcSsh -Name $DESTINATION_ONTAP_CLUSTER_MGMT_IP
-Credential $DESTINATION_ONTAP_CREDS -Command "security multi-admin-verify
rule create -operation `"$rule`""
            }

            logMessage -message "Creating multi admin verification group
for ONTAP Cluster $DESTINATION_ONTAP_CLUSTER_MGMT_IP, Group name :
$MULTI_ADMIN_APPROVAL_GROUP_NAME, Users : $MULTI_ADMIN_APPROVAL_USERS,
Email : $MULTI_ADMIN_APPROVAL_EMAIL"
            Invoke-NcSsh -Name $DESTINATION_ONTAP_CLUSTER_MGMT_IP
-Credential $DESTINATION_ONTAP_CREDS -Command "security multi-admin-verify
```

```
approval-group create -name $MULTI_ADMIN_APPROVAL_GROUP_NAME -approvers
$MULTI_ADMIN_APPROVAL_USERS -email `"$MULTI_ADMIN_APPROVAL_EMAIL`""
            logMessage -message "Created multi admin verification group
for ONTAP Cluster $DESTINATION_ONTAP_CLUSTER_MGMT_IP, Group name :
$MULTI_ADMIN_APPROVAL_GROUP_NAME, Users : $MULTI_ADMIN_APPROVAL_USERS,
Email : $MULTI_ADMIN_APPROVAL_EMAIL" -type "SUCCESS"

            logMessage -message "Enabling multi admin verification group
$MULTI_ADMIN_APPROVAL_GROUP_NAME"
            Invoke-NcSsh -Name $DESTINATION_ONTAP_CLUSTER_MGMT_IP
-Credential $DESTINATION_ONTAP_CREDS -Command "security multi-admin-verify
modify -approval-groups $MULTI_ADMIN_APPROVAL_GROUP_NAME -required
-approvers 1 -enabled true"
            logMessage -message "Enabled multi admin verification group
$MULTI_ADMIN_APPROVAL_GROUP_NAME" -type "SUCCESS"

            logMessage -message "Enabling multi admin verification for
ONTAP Cluster $DESTINATION_ONTAP_CLUSTER_MGMT_IP"
            Invoke-NcSsh -Name $DESTINATION_ONTAP_CLUSTER_MGMT_IP
-Credential $DESTINATION_ONTAP_CREDS -Command "security multi-admin-verify
modify -enabled true"
            logMessage -message "Successfully enabled multi admin
verification for ONTAP Cluster $DESTINATION_ONTAP_CLUSTER_MGMT_IP" -type
"SUCCESS"

            logMessage -message "Enabling multi admin verification for
ONTAP Cluster $DESTINATION_ONTAP_CLUSTER_MGMT_IP"
            Invoke-NcSsh -Name $DESTINATION_ONTAP_CLUSTER_MGMT_IP
-Credential $DESTINATION_ONTAP_CREDS -Command "security multi-admin-verify
modify -enabled true"
            logMessage -message "Successfully enabled multi admin
verification for ONTAP Cluster $DESTINATION_ONTAP_CLUSTER_MGMT_IP" -type
"SUCCESS"
        }

    } catch {
        handleError -errorMessage $_.Exception.Message
    }
}

function additionalSecurityHardening {
    try {
        $command = "set -privilege advanced -confirmations off;security
protocol modify -application telnet -enabled false;"
        logMessage -message "Disabling Telnet"
        Invoke-NcSsh -Name $DESTINATION_ONTAP_CLUSTER_MGMT_IP -Credential
```

```
$DESTINATION_ONTAP_CREDS -Command $command
        logMessage -message "Disabled Telnet" -type "SUCCESS"

        #$command = "set -privilege advanced -confirmations off;security
config modify -interface SSL -is-fips-enabled true;"
        #logMessage -message "Enabling Global FIPS"
        ##Invoke-SSHCommand -SessionId $sshSession.SessionId -Command
$command -ErrorAction Stop
        #logMessage -message "Enabled Global FIPS" -type "SUCCESS"

        $command = "set -privilege advanced -confirmations off;network
interface service-policy modify-service -vserver cluster2 -policy default-
management -service management-https -allowed-addresses $ALLOWED_IPS;"
        logMessage -message "Restricting IP addresses $ALLOWED_IPS for
Cluster management HTTPS"
        Invoke-NcSsh -Name $DESTINATION_ONTAP_CLUSTER_MGMT_IP -Credential
$DESTINATION_ONTAP_CREDS -Command $command
        logMessage -message "Successfully restricted IP addresses
$ALLOWED_IPS for Cluster management HTTPS" -type "SUCCESS"

        #logMessage -message "Checking if audit logs volume audit_logs
exists"
        #$volume = Get-NcVol -Vserver $DESTINATION_VSERVER -Name
audit_logs -ErrorAction Stop

        #if($volume) {
        #    logMessage -message "Volume audit_logs already exists!
Skipping creation"
        #} else {
        #    # Create audit logs volume
        #    logMessage -message "Creating audit logs volume : audit_logs"
        #    New-NcVol -Name audit_logs -Aggregate
$DESTINATION_AGGREGATE_NAME -Size 5g -ErrorAction Stop | Select-Object
-Property Name, State, TotalSize, Aggregate, Vserver
        #    logMessage -message "Volume audit_logs created successfully"
-type "SUCCESS"
        #}

        ## Mount audit logs volume to path /vol/audit_logs
        #logMessage -message "Creating junction path for volume audit_logs
at path /vol/audit_logs for vServer $DESTINATION_VSERVER"
        #Mount-NcVol -VserverContext $DESTINATION_VSERVER -Name audit_logs
-JunctionPath /audit_logs | Select-Object -Property Name, -JunctionPath
        #logMessage -message "Created junction path for volume audit_logs
at path /vol/audit_logs for vServer $DESTINATION_VSERVER" -type "SUCCESS"
```

```
        #logMessage -message "Enabling audit logging for vServer
$DESTINATION_VSERVER at path /vol/audit_logs"
        #$command = "set -privilege advanced -confirmations off;vserver
audit create -vserver $DESTINATION_VSERVER -destination /audit_logs
-format xml;"
        #Invoke-SSHCommand -SessionI  $sshSession.SessionId -Command
$command -ErrorAction Stop
        #logMessage -message "Successfully enabled audit logging for
vServer $DESTINATION_VSERVER at path /vol/audit_logs"

    } catch {
        handleError -errorMessage $_.Exception.Message
    }
}
```

使用 PowerShell 進行 ONTAP 網路資料保險箱驗證

強大的網路資料保險箱應能抵禦複雜的攻擊、即使攻擊者擁有憑證、也能透過提高權限的 Privileges 存取環境。

一旦規則就緒、嘗試（假設攻擊者能夠進入）刪除資料保險箱端的快照將會失敗。所有強化設定也同樣適用、只要設定必要的限制、並保護系統。

PowerShell 程式碼範例、可依排程驗證組態。

```
function analyze {

    for($i = 0; $i -lt $DESTINATION_VOLUME_NAMES.Length; $i++) {
        try {
            # checking if volume is of type SnapLock Compliance
            logMessage -message "Checking if SnapLock Compliance volume
$($DESTINATION_VOLUME_NAMES[$i]) exists in vServer $DESTINATION_VSERVER"
            $volume = Get-NcVol -Vserver $DESTINATION_VSERVER -Volume
$DESTINATION_VOLUME_NAMES[$i] | Select-Object -Property Name, State,
TotalSize, Aggregate, Vserver, Snaplock | Where-Object { $_.Snaplock.Type
-eq "compliance" }
            if($volume) {
                $volume
                logMessage -message "SnapLock Compliance volume
$($DESTINATION_VOLUME_NAMES[$i]) exists in vServer $DESTINATION_VSERVER"
-type "SUCCESS"
            } else {
                handleError -errorMessage "SnapLock Compliance volume
$($DESTINATION_VOLUME_NAMES[$i]) does not exist in vServer
$DESTINATION_VSERVER. Recommendation: Run the script with SCRIPT_MODE
```

```
`"configure`" to create and configure the cyber vault SnapLock Compliance
volume"
            }

            # checking SnapMirror relationship
            logMessage -message "Checking if SnapMirror relationship
exists between source volume $($SOURCE_VOLUME_NAMES[$i]) and destination
SnapLock Compliance volume $($DESTINATION_VOLUME_NAMES[$i])"
            $snapmirror = Get-NcSnapmirror | Select-Object SourceCluster,
SourceLocation, DestinationCluster, DestinationLocation, Status,
MirrorState | Where-Object { $_.SourceCluster -eq
$SOURCE_ONTAP_CLUSTER_NAME -and $_.SourceLocation -eq "$($SOURCE_VSERVER)
:$($SOURCE_VOLUME_NAMES[$i])" -and $_.DestinationCluster -eq
$DESTINATION_ONTAP_CLUSTER_NAME -and $_.DestinationLocation -eq "
$($DESTINATION_VSERVER):$($DESTINATION_VOLUME_NAMES[$i])" -and $_.Status
-eq "snapmirrored" }
            if($snapmirror) {
                $snapmirror
                logMessage -message "SnapMirror relationship successfully
configured and in healthy state" -type "SUCCESS"
            } else {
                handleError -errorMessage "SnapMirror relationship does
not exist between the source volume $($SOURCE_VOLUME_NAMES[$i]) and
destination SnapLock Compliance volume $($DESTINATION_VOLUME_NAMES[$i])
(or) SnapMirror status uninitialized/unhealthy. Recommendation: Run the
script with SCRIPT_MODE `"configure`" to create and configure the cyber
vault SnapLock Compliance volume and configure the SnapMirror
relationship"
            }
        }
        catch {
            handleError -errorMessage $_.Exception.Message
        }
    }

    try {

        # checking NFS service is disabled
        logMessage -message "Checking if NFS service is disabled on
vServer $DESTINATION_VSERVER"
        $nfsService = Get-NcNfsService
        if($nfsService) {
            handleError -errorMessage "NFS service running on vServer
$DESTINATION_VSERVER. Recommendation: Run the script with SCRIPT_MODE
`"configure`" to disable NFS on vServer $DESTINATION_VSERVER"
        } else {
```

```
            logMessage -message "NFS service is disabled on vServer
$DESTINATION_VSERVER" -type "SUCCESS"
        }

        # checking CIFS/SMB server is disabled
        logMessage -message "Checking if CIFS/SMB server is disabled on
vServer $DESTINATION_VSERVER"
        $cifsServer = Get-NcCifsServer
        if($cifsServer) {
            handleError -errorMessage "CIFS/SMB server running on vServer
$DESTINATION_VSERVER. Recommendation: Run the script with SCRIPT_MODE
`"configure`" to disable CIFS/SMB on vServer $DESTINATION_VSERVER"
        } else {
            logMessage -message "CIFS/SMB server is disabled on vServer
$DESTINATION_VSERVER" -type "SUCCESS"
        }

        # checking iSCSI service is disabled
        logMessage -message "Checking if iSCSI service is disabled on
vServer $DESTINATION_VSERVER"
        $iscsiService = Get-NcIscsiService
        if($iscsiService) {
            handleError -errorMessage "iSCSI service running on vServer
$DESTINATION_VSERVER. Recommendation: Run the script with SCRIPT_MODE
`"configure`" to disable iSCSI on vServer $DESTINATION_VSERVER"
        } else {
            logMessage -message "iSCSI service is disabled on vServer
$DESTINATION_VSERVER" -type "SUCCESS"
        }

        # checking FCP service is disabled
        logMessage -message "Checking if FCP service is disabled on
vServer $DESTINATION_VSERVER"
        $fcpService = Get-NcFcpService
        if($fcpService) {
            handleError -errorMessage "FCP service running on vServer
$DESTINATION_VSERVER. Recommendation: Run the script with SCRIPT_MODE
`"configure`" to disable FCP on vServer $DESTINATION_VSERVER"
        } else {
            logMessage -message "FCP service is disabled on vServer
$DESTINATION_VSERVER" -type "SUCCESS"
        }

        # checking if all data lifs are disabled on vServer
        logMessage -message "Finding all data lifs on vServer :
$DESTINATION_VSERVER"
```

```
        $dataLifs = Get-NcNetInterface -Vserver $DESTINATION_VSERVER |
Where-Object { $_.Role -contains "data_core" }
        $dataLifs | Select-Object -Property InterfaceName, OpStatus,
DataProtocols, Vserver, Address

        logMessage -message "Checking if all data lifs are disabled for
vServer : $DESTINATION_VSERVER"
        # Disable the filtered data LIFs
        foreach ($lif in $dataLifs) {
            $checkLif = Get-NcNetInterface -Vserver $DESTINATION_VSERVER
-Name $lif.InterfaceName | Where-Object { $_.OpStatus -eq "down" }
            if($checkLif) {
                logMessage -message "Data lif $($lif.InterfaceName)
disabled for vServer $DESTINATION_VSERVER" -type "SUCCESS"
            } else {
                handleError -errorMessage "Data lif $($lif.InterfaceName)
is enabled. Recommendation: Run the script with SCRIPT_MODE `"configure`"
to disable Data lifs for vServer $DESTINATION_VSERVER"
            }
        }
        logMessage -message "All data lifs are disabled for vServer :
$DESTINATION_VSERVER" -type "SUCCESS"

        # check if multi-admin verification is enabled
        logMessage -message "Checking if multi-admin verification is
enabled"
        $maaConfig = Invoke-NcSsh -Name $DESTINATION_ONTAP_CLUSTER_MGMT_IP
-Credential $DESTINATION_ONTAP_CREDS -Command "set -privilege advanced;
security multi-admin-verify show"
        if ($maaConfig.Value -match "Enabled" -and $maaConfig.Value -match
"true") {
            $maaConfig
            logMessage -message "Multi-admin verification is configured
and enabled" -type "SUCCESS"
        } else {
            handleError -errorMessage "Multi-admin verification is not
configured or not enabled. Recommendation: Run the script with SCRIPT_MODE
`"configure`" to enable and configure Multi-admin verification"
        }

        # check if telnet is disabled
        logMessage -message "Checking if telnet is disabled"
        $telnetConfig = Invoke-NcSsh -Name
$DESTINATION_ONTAP_CLUSTER_MGMT_IP -Credential $DESTINATION_ONTAP_CREDS
-Command "set -privilege advanced; security protocol show -application
telnet"
```

```
        if ($telnetConfig.Value -match "enabled" -and $telnetConfig.Value
-match "false") {
            logMessage -message "Telnet is disabled" -type "SUCCESS"
        } else {
            handleError -errorMessage "Telnet is enabled. Recommendation:
Run the script with SCRIPT_MODE `"configure`" to disable telnet"
        }

        # check if network https is restricted to allowed IP addresses
        logMessage -message "Checking if HTTPS is restricted to allowed IP
addresses $ALLOWED_IPS"
        $networkServicePolicy = Invoke-NcSsh -Name
$DESTINATION_ONTAP_CLUSTER_MGMT_IP -Credential $DESTINATION_ONTAP_CREDS
-Command "set -privilege advanced; network interface service-policy show"
        if ($networkServicePolicy.Value -match "management-https:
$($ALLOWED_IPS)") {
            logMessage -message "HTTPS is restricted to allowed IP
addresses $ALLOWED_IPS" -type "SUCCESS"
        } else {
            handleError -errorMessage "HTTPS is not restricted to allowed
IP addresses $ALLOWED_IPS. Recommendation: Run the script with SCRIPT_MODE
`"configure`" to restrict allowed IP addresses for HTTPS management"
        }
    }
    catch {
        handleError -errorMessage $_.Exception.Message
    }
}
```

此螢幕擷取畫面顯示資料保險箱控制器上沒有連線。

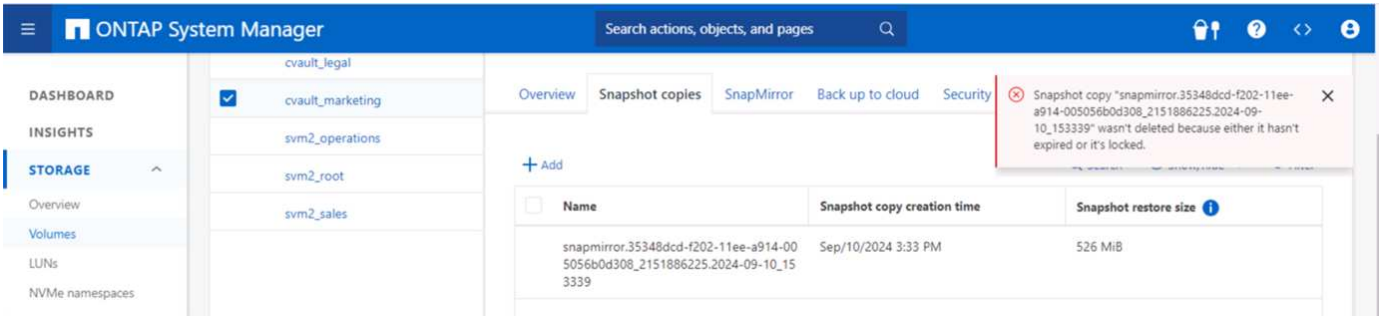
```
cluster2::> network connections  listening show
This table is currently empty.

cluster2::> network connections active show-services
This table is currently empty.

cluster2::> network connections active show-protocols
This table is currently empty.

cluster2::>
```

此螢幕擷取畫面顯示無法竄改快照。



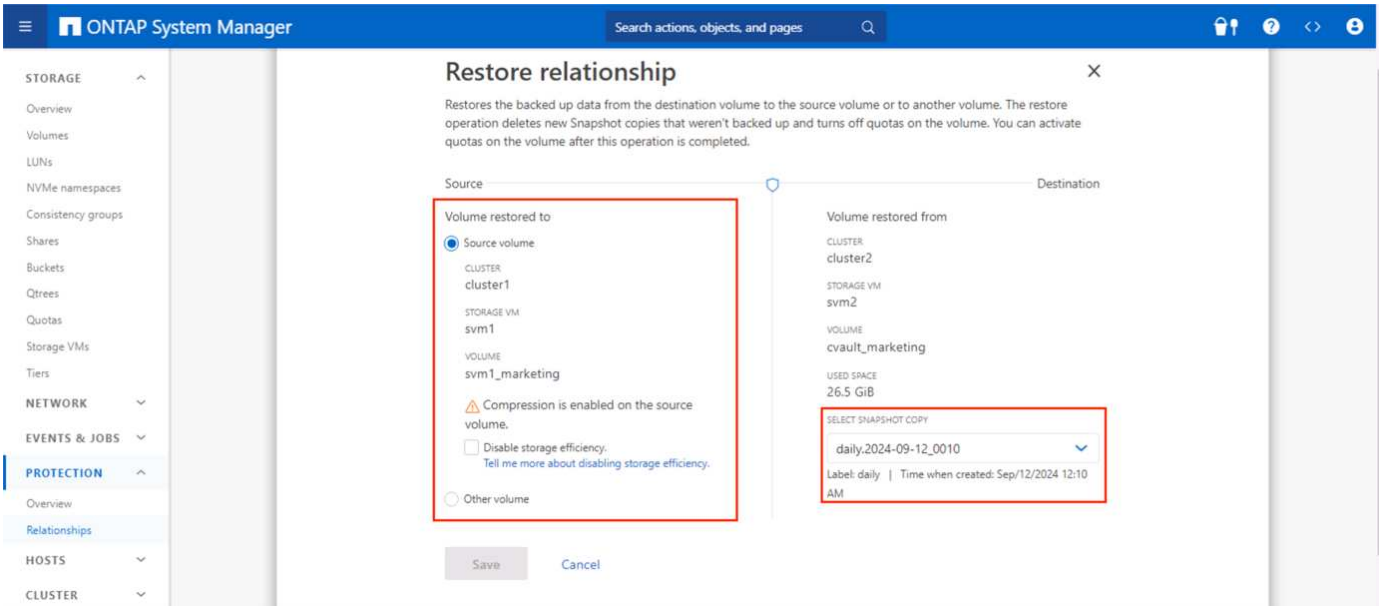
若要驗證並確認氣墊功能、請遵循下列步驟：

- 測試網路隔離功能、以及在資料未傳輸時停止連線的能力。
- 驗證除了允許的 IP 位址之外、無法從任何實體存取管理介面。
- 驗證多重管理驗證是否已就緒、以提供額外的核准層級。
- 驗證透過 CLI 和 REST API 存取的能力
- 從來源觸發資料保險箱的傳輸作業、並確保無法修改資料保險箱複本。
- 嘗試刪除傳輸至資料保險箱的不可變快照複本。
- 嘗試竄改系統時鐘來修改保留期間。

ONTAP 網路資料保險箱資料恢復

如果資料在正式作業資料中心遭到銷毀、則可將網路資料保險箱中的資料安全地還原至所選環境。與實體空拍解決方案不同的是、空中綁定的 ONTAP 網路資料保險箱是使用原生 ONTAP 功能（例如 SnapLock Compliance 和 SnapMirror ）打造而成。如此一來、恢復程序既快速又容易執行。

萬一發生勒索軟體攻擊且需要從網路資料保險箱中恢復、則只要使用網路資料保險箱中的快照複本來還原加密資料、就能輕鬆又簡單地進行還原。



如果要求提供更快速的方法、以便在必要時將資料恢復上線、以便快速驗證、隔離及分析資料以進行還原。使用 FlexClone 時、若將 SnapLock 類型選項設為非 SnapLock 類型、即可輕鬆達成此目標。

從 ONTAP 9 。 13.1 開始、只要建立 FlexClone 、並將 SnapLock 類型選項設為「 SnapLock 」、即可立即還原 SnapLock 資料保險箱關係目的地 SnapLock 磁碟區上的鎖定 Snapshot 複本。執行 Volume Clone 建立作業時、請將 Snapshot 複本指定為「父快照」。有關使用 SnapLock 類型建立 FlexClone Volume 的更多資訊"請按這裡。"

從網路資料保險箱執行恢復程序、可確保建立適當的步驟、以連線至網路資料保險箱並擷取資料。在網路攻擊事件期間、規劃和測試程序對於任何恢復都是不可或缺的。

其他考量

設計和部署 ONTAP 型網路資料保險箱時、還有其他考量。

容量規模調整考量

ONTAP 網路資料保險箱目的地磁碟區所需的磁碟空間量取決於各種因素、其中最重要的因素是來源磁碟區資料的變更率。目的地磁碟區上的備份排程和 Snapshot 排程都會影響目的地磁碟區上的磁碟使用量、而且來源磁碟區上的變更率不太可能是固定的。建議您提供額外儲存容量的緩衝區、以滿足終端使用者或應用程式行為未來變更的需求。

在 ONTAP 中設定 1 個月的保留關係規模、需要根據多項因素來計算儲存需求、包括主要資料集的大小、資料變更率（每日變更率）、以及重複資料刪除和壓縮節省（如果適用）。

以下是逐步方法：

第一步是瞭解您使用網路資料保險箱保護的來源磁碟區大小。這是最初複寫至網路資料保險箱目的地的基本資料量。接下來、估計資料集的每日變更率。這是每天變更的資料百分比。瞭解資料動態的重要性。

例如：

- 主要資料集大小 = 5TB
- 每日變更率 = 5% （ 0.05 ）
- 重複資料刪除與壓縮效率 = 50% （ 0.50 ）

現在、讓我們逐一瞭解計算結果：

- 計算每日資料變更率：

```
Changed data per day = 5000 * 5% = 250GB
```

- 計算 30 天內變更的資料總計：

```
Total changed data in 30 days = 250 GB * 14 = 3.5TB
```

- 計算所需的總儲存容量：

```
TOTAL = 5TB + 3.5TB = 8.5TB
```

- 套用重複資料刪除與壓縮節省：

```
EFFECTIVE = 8.5TB * 50% = 4.25TB
```

- 儲存需求摘要 *
- 缺乏效率：需要 * 8.5TB* 來儲存 30 天的網路資料保險箱資料。
- 效率達 50% ：重複資料刪除和壓縮後需要使用 **4.25TB** 的儲存設備。

由於中繼資料、 Snapshot 複本可能會產生額外的負擔、但這通常是次要的。

如果每天執行多個備份、請根據每天執行的 Snapshot 複本數量來調整計算。

隨著時間的推移、資料成長的因素可確保規模調整符合未來需求。

效能對主要 / 來源的影響

由於資料傳輸是拉動作業、因此對主要儲存效能的影響可能會因工作負載、資料量和備份頻率而異。然而、整體效能對主要系統的影響通常中等且可管理、因為資料傳輸是為了將資料保護和備份工作卸載至網路資料保險箱儲存系統而設計。在初始關係設定和第一次完整備份期間、大量資料會從主要系統傳輸到網路資料保險箱系統（ SnapLock Compliance Volume ）。這可能會導致主要系統的網路流量增加和 I/O 負載增加。完成初始完整備份後、 ONTAP 只需追蹤和傳輸自上次備份以來變更的區塊。與初始複寫相比、這會造成更小的 I/O 負載。遞增更新效率極高、對主要儲存效能的影響也極微。資料保險箱程序會在背景執行、減少干擾主要系統正式作業工作負載的機會。

- 確保儲存系統擁有足夠的資源（ CPU 、記憶體和 IOP ）來處理額外的負載、可減輕效能影響。

設定、分析、 **cron** 指令碼

NetApp 已建立單一指令碼、可下載及用於設定、驗證及排程網路資料保險箱關係。

此指令碼的功能

- 叢集對等
- SVM 對等關係
- DP Volume 建立
- SnapMirror 關係與初始化
- 強化網路資料保險箱使用的 ONTAP 系統
- 根據傳輸排程來關閉及恢復關係
- 定期驗證安全性設定、並產生顯示任何異常的報告

如何使用此指令碼

下載指令碼並使用指令碼、只要遵循下列步驟即可：

- 以系統管理員身分啟動 Windows PowerShell 。
- 瀏覽至包含指令碼的目錄。
- 使用 `.\`語法和必要參數來執行指令碼

請確保輸入所有資訊。在第一次執行（設定模式）時、系統會要求提供正式作業和新網路資料保險箱系統的認證。之後、它會建立 SVM 對等關係（如果不存在）、磁碟區和系統之間的 SnapMirror 、並將其初始化。

cron 模式可用於排程資料傳輸的休眠和恢復。

操作模式

自動化指令碼提供 3 種執行模式： `configure`、 `analyze`和 `cron`。

```
if($SCRIPT_MODE -eq "configure") {
        configure
    } elseif ($SCRIPT_MODE -eq "analyze") {
        analyze
    } elseif ($SCRIPT_MODE -eq "cron") {
        runCron
    }
```

- 組態 - 執行驗證檢查、並將系統設定為無線。
- 分析 - 自動監控和報告功能、可將資訊傳送給監控群組、以確保組態不會發生異常和可疑活動。
- cron ：為了啟用中斷連線的基礎架構、 cron 模式會自動停用 LIF 並停止傳輸關係。

視系統效能和資料量而定、傳輸這些選定磁碟區中的資料需要時間。

```
./script.ps1 -SOURCE_ONTAP_CLUSTER_MGMT_IP "172.21.166.157"
-SOURCE_ONTAP_CLUSTER_NAME "NTAP915_Src" -SOURCE_VSERVER "svm_NFS"
-SOURCE_VOLUME_NAME "Src_RP_Vol01" -DESTINATION_ONTAP_CLUSTER_MGMT_IP
"172.21.166.159" -DESTINATION_ONTAP_CLUSTER_NAME "NTAP915_Destn"
-DESTINATION_VSERVER "svm_nim_nfs" -DESTINATION_AGGREGATE_NAME
"NTAP915_Destn_01_VM_DISK_1" -DESTINATION_VOLUME_NAME "Dst_RP_Vol01_Vault"
-DESTINATION_VOLUME_SIZE "5g" -SNAPLOCK_MIN_RETENTION "15minutes"
-SNAPLOCK_MAX_RETENTION "30minutes" -SNAPMIRROR_PROTECTION_POLICY
"XDPDefault" -SNAPMIRROR_SCHEDULE "5min" -DESTINATION_CLUSTER_USERNAME
"admin" -DESTINATION_CLUSTER_PASSWORD "PASSWORD123"
```

ONTAP 網路資料保險箱 PowerShell 解決方案結論

NetApp 運用 ONTAP 提供的強大強化方法來充分發揮氣力、讓您建立安全、隔離的儲存環

境、以因應不斷演變的網路威脅。所有這些都是在維持現有儲存基礎架構的敏捷度和效率的同時完成的。這項安全存取功能可讓公司以最少變更現有人員、程序和技術架構的方式、達成嚴格的安全和正常運作時間目標。

ONTAP 網路資料保險箱使用 ONTAP 的原生功能是一種簡易的方法、可提供額外的保護、以建立不可改變的資料複本。將 NetApp 的 ONTAP 型網路資料保險箱新增至整體安全狀態、將會：

- 建立與正式作業和備份網路分開和中斷連線的環境、並限制使用者存取。