Implementación de Alta Disponibilidad DFS

Gestión de Almacenamiento Distribuido (DCSUME.LOCAL)

Infraestructura: NAS Synology DS1522+ (Principal) y NAS Synology DS223 (Respaldo)

1. Resumen del Sistema

El sistema utiliza **DFS (Distributed File System)** para gestionar conexiones hacia dos destinos físicos, permitiendo la conmutación rápida entre ellos (se ejecuta en Powershell con permisos de administrador).

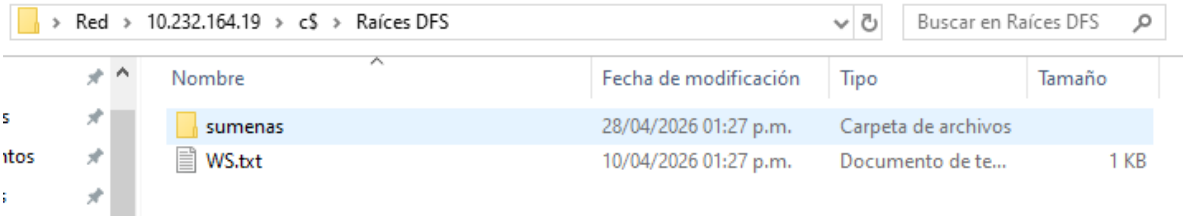
2. Flujo de Implementación (Orden de Ejecución)

Paso 1: Creación de la Estructura de Carpetas

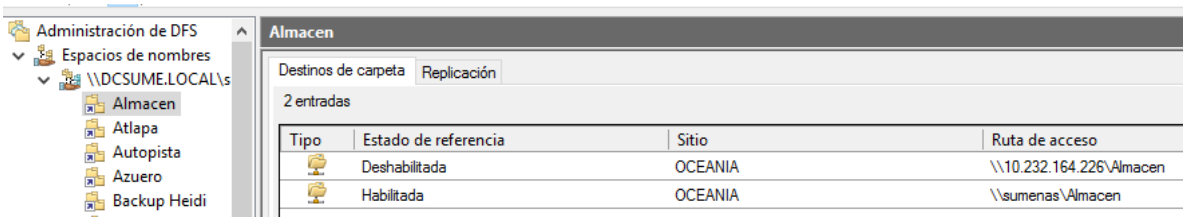
Script: Carpetas-DFS.ps1

Este script inicializa el entorno leyendo una lista de carpetas desde un archivo de texto (ya creado).

- **Función:** Crea los enlaces en el Namespace y asocia tanto el NAS principal como el de respaldo.



- **Configuración Crítica:** Por seguridad, el destino de respaldo se configura inicialmente como **Offline** para evitar escrituras simultáneas que causen conflictos de versiones, (Duración de la memoria en cache de las referencias **60 segundos**).



Script:

Carpetas-DFS.ps1*

```powershell
# --- CONFIGURACIÓN ---
$Namespace = "\\DCSUME.LOCAL\sumenas"  # Nombres DFS actual
$NAS_Principal = "\\sumenas"           # Nombre o IP del DS1522+ NAS Principal
$NAS_Respaldo = "\\10.232.164.226"      # IP del DS223 del NAS de Respaldo
$ArchivoRutas = "C:\Raíces DFS\WS.txt"

# --- PROCESO ---
$Nombres = Get-Content $ArchivoRutas

foreach ($Nombre in $Nombres) {
    # Aqui elimino espacios en blanco si existen
    $Nombre = $Nombre.Trim()
    if ([string]::IsNullOrWhiteSpace($Nombre)) { continue }

    Write-Host "Configurando: $Nombre" -ForegroundColor Cyan

    # 1. Creo la carpeta en el Namespace DFS apuntando al principal
    # Uso -TargetPath para el primer destino
    New-DfsnFolder -Path "$Namespace\$Nombre" -TargetPath "$NAS_Principal\$Nombre"

    # 2. Agrego el segundo destino (NAS Respaldo)
    # ERROR ANTERIOR: El comando correcto es New-DfsnFolderTarget
    New-DfsnFolderTarget -Path "$Namespace\$Nombre" -TargetPath "$NAS_Respaldo\$Nombre"

    # 3. aqui Deshabilita el destino del respaldo si no esta disponible
    # ERROR ANTERIOR: En DFS de Windows, el estado es 'Offline' (no Disabled)
    Set-DfsnFolderTarget -Path "$Namespace\$Nombre" -TargetPath "$NAS_Respaldo\$Nombre" -State Offline
}

Write-Host "¡Proceso completado con éxito!" -ForegroundColor Green
```

Paso 2: Actualización de Directivas de Grupo (GPO)

Script: actualizarGPO-DFS.ps1

Es fundamental para la migración de usuarios antiguos hacia la nueva infraestructura con los cambios mencionados (se ejecuta en Powershell con permisos de administrador).

- **Función:** Escanea los archivos Shortcuts.xml dentro de **SYSVOL** y reemplaza las rutas antiguas (\\DCSUME\sumenas) por la ruta jerárquica correcta (\\DCSUME.LOCAL\sumenas).
- **Impacto:** Automatiza la corrección de mapeos en los terminales de los empleados sin intervención manual.

Script:

actualizarGPO-DFS.ps1

```powershell
# --- CONFIGURACIÓN ---
$RutaBaseGPO = "C:\Windows\SYSVOL\domain\Policies"
$ViejaRuta = "\\DCSUME\sumenas"          # Esta es la ruta que mantiene actualmente
$NuevaRuta = "\\DCSUME.LOCAL\sumenas"    # Esta es la ruta que se quiere cambiar

# --- CONTADORES ---
$ArchivosProcesados = 0
$RutasReemplazadas = 0

# --- PROCESO ---
Write-Host "Buscando archivos de Accesos Directos..." -ForegroundColor Cyan

# Aqui se Localizan todos los archivos Shortcuts.xml en todas las GPOs
$Archivos = Get-ChildItem -Path $RutaBaseGPO -Recurse -Filter "Shortcuts.xml"

foreach ($Archivo in $Archivos) {
    $ArchivosProcesados++
    $ContenidoOriginal = Get-Content $Archivo.FullName -Raw

    # Verifico si contiene la ruta vieja
    if ($ContenidoOriginal -like "*$ViejaRuta*") {
        Write-Host "? Actualizando rutas en: $($Archivo.FullName)" -ForegroundColor Yellow

        # Cuento cuántas veces aparece la ruta vieja en este archivo
        $Coincidencias = ($ContenidoOriginal -split [regex]::Escape($ViejaRuta)).Count - 1
        $RutasReemplazadas += $Coincidencias

        # Reemplazo de la cadena de texto
        $NuevoContenido = $ContenidoOriginal -replace [regex]::Escape($ViejaRuta), $NuevaRuta

        # Guardo los cambios
        Set-Content -Path $Archivo.FullName -Value $NuevoContenido -Encoding UTF8
        Write-Host "   ?? Reemplazadas $Coincidencias rutas en este archivo." -ForegroundColor Green
    }
}

# --- RESUMEN FINAL ---
Write-Host ""
Write-Host "? RESUMEN DEL PROCESO:" -ForegroundColor Green
Write-Host "   Archivos procesados: $ArchivosProcesados"
Write-Host "   Rutas reemplazadas:  $RutasReemplazadas"
Write-Host "   ¡Cambio completado con éxito!" -ForegroundColor Green
```

Paso 3: Configuración de Red y DNS

Script: Configurar-DNS-Failover.ps1

Asegura que los clientes siempre puedan resolver el nombre del dominio, aquí está el requisito indispensable para DFS (se ejecuta en Powershell con permisos de administrador).

- **Función:** Configura estáticamente los adaptadores de red para apuntar a los DCs 10.232.164.19 y 10.232.160.13.
- **Acción adicional:** Ejecuta un comando para limpiar la caché DNS local (flushdns) en caso de ser necesario.

Script:

Configurar-DNS-Failover.ps1

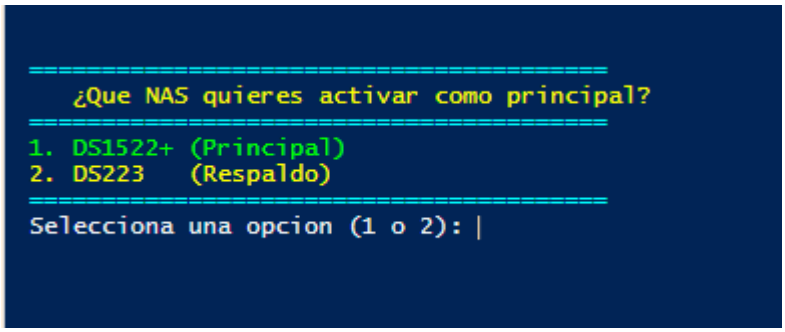
```powershell
# Script para configurar DNS con failover
# Servidores DNS: DCSUME (10.232.164.19) y DCSUME02 (10.232.160.13)

try {
    # Log de inicio
    $logFile = "C:\Windows\Temp\DNS-Config.log"
    $timestamp = Get-Date -Format "yyyy-MM-dd HH:mm:ss"
    Add-Content -Path $logFile -Value "$timestamp - Iniciando configuración DNS"

    # Obtener adaptadores de red activos (excluir virtuales y loopback)
    $adapters = Get-NetAdapter | Where-Object {
        $_.Status -eq "Up" -and
        $_.InterfaceDescription -notlike "*Hyper-V*" -and
        $_.InterfaceDescription -notlike "*VirtualBox*" -and
        $_.InterfaceDescription -notlike "*VMware*" -and
        $_.Name -notlike "*Loopback*"
    }

    foreach ($adapter in $adapters) {
        # Configurar DNS primario y secundario
        Set-DnsClientServerAddress -InterfaceAlias $adapter.Name -ServerAddresses ("10.232.164.19","10.232.160.13")

        Add-Content -Path $logFile -Value "$timestamp - DNS configurado en $($adapter.Name): 10.232.164.19, 10.232.160.13"
    }

    # Limpiar caché DNS
    ipconfig /flushdns | Out-Null

    Add-Content -Path $logFile -Value "$timestamp - Configuración DNS completada exitosamente"

} catch {
    Add-Content -Path $logFile -Value "$timestamp - ERROR: $($_.Exception.Message)"
}
```

Paso 4: Herramienta de Gestión de Failover

Script: Failover_DFS.ps1

Es el panel de control interactivo para el administrador en caso de contingencia, falla electica, de red oh cualquier otra que se dé (se ejecuta en Powershell con permisos de administrador).

- **Función:** Permite al administrador elegir qué NAS activar como principal mediante un menú.



- **Capacidades:** * Detecta automáticamente qué DC está disponible para ejecutar los cambios.
    - Muestra una previsualización de los cambios antes de aplicarlos.
    - Fuerza la replicación de Active Directory si se realiza desde el servidor de respaldo.

3. Componentes Técnicos

Componente	Identificador / Ruta
Namespace DFS	\\DCSUME.LOCAL\sumenas
NAS Principal	\\sumenas (DS1522+)
NAS Respaldo	\\10.232.164.226 (DS223)
DC Primario	DCSUME.LOCAL (10.232.164.19)
DC Secundario	10.232.160.13

Script:

Failover_DFS MOD.ps1*

```powershell
# --- CONFIGURACION DE LAS RUTAS ---
[cite_start]$Namespace     = "\\DCSUME.LOCAL\sumenas"   # Espacio de nombres DFS
[cite_start]$NAS_Principal = "\\sumenas"                # Nombre / IP del DS1522+
[cite_start]$NAS_Respaldo  = "\\10.232.164.226"         # IP del DS223
$LogFolder     = "C:\LogsDFS"

# --- DCs DISPONIBLES (orden de preferencia) ---
[cite_start]$DC_Principal = "DCSUME.LOCAL"             # DC primario (por nombre)
[cite_start]$DC_Respaldo  = "10.232.160.13"            # DC secundario (por IP directa)

# ============================================================
# --- DETECTAR QUE DC USAR ---
# ============================================================
Write-Host ""
Write-Host ">>> Verificando conectividad con DC Principal..." -ForegroundColor Cyan

$DCActivo = $null

if (Test-Connection -ComputerName $DC_Principal -Count 2 -Quiet -ErrorAction SilentlyContinue) {
    $DCActivo = $DC_Principal
    Write-Host "[OK] DC Principal disponible: $DC_Principal" -ForegroundColor Green
} elseif (Test-Connection -ComputerName $DC_Respaldo -Count 2 -Quiet -ErrorAction SilentlyContinue) {
    $DCActivo = $DC_Respaldo
    Write-Host "[!] DC Principal NO responde. Usando DC Secundario: $DC_Respaldo" -ForegroundColor Yellow
} else {
    Write-Host "[X] Ningun DC disponible. Verifica la conectividad de red." -ForegroundColor Red
    Start-Sleep -Seconds 5
    exit 1
}

# ============================================================
# --- CREAMOS CARPETA DE LOGS SI NO EXISTE ---
# ============================================================
if (-not (Test-Path $LogFolder)) {
    New-Item -ItemType Directory -Path $LogFolder -Force | Out-Null
    Write-Host "-> Carpeta de logs creada: $LogFolder" -ForegroundColor Green
}
```