Disk Capacity Forecasting (Power BI + PowerShell)

Weekly disk capacity capture from Windows servers + a Power BI model to forecast runout at the drive level.

Table of Contents

Overview

This repo contains:

- A **PowerShell script** that queries a list of remote Windows servers for fixed drives and outputs weekly CSV snapshots.
- A **Power BI model** setup (DAX provided) that reads those CSVs, tracks Used/Free GB over time, and forecasts **weeks to full** per drive.
- An optional script to generate **sample CSVs** for testing.

Works with the following CSV schema (one row per `ServerName` + `Drive Letters` per capture date):

```
Capture Date, Capture Day, ServerName, Drive Letters, TotalSizeDrive,
FreeSpaceDrive, FreeSpacePercentage
```

Prerequisites

- **PowerShell 5.1+** (Windows) or **PowerShell 7+**.
- Remote servers are Windows and expose **CIM/WMI** (
- WinRM service running (HTTP 5985 / HTTPS 5986) or DCOM for WMI; default script uses **CIM over WinRM**).
- The running account is **Local Administrator** on targets **or** has WMI permissions to `root\cimv2`.
- Firewall allows WinRM.
- A text file with server names (one per line). Duplicates are fine; the script de-duplicates.
- Power BI Desktop (latest) for report authoring.

> Tip: If you need alternate credentials, the script supports `-Credential (Get-Credential)`.

Folder Structure (suggested)

```
.
├─ README.md
├─ scripts
│  ├─ Generate-SampleCSVs.ps1
│  └─ Collect-DriveCapacity.ps1
├─ data
│  ├─ servers.txt
│  └─ output\  (CSV snapshots land here)
└─ powerbi
   └─ notes.md  (DAX, visuals, setup steps)
```

Data Schema

Column	Type	Notes
Capture Date	Date	e.g., `2025-07-28`
Capture Day	Text	e.g., `Monday`
ServerName	Text	From `servers.txt`
Drive Letters	Text	e.g., `D:\\`, `E:\\`, `F:\\`
TotalSizeDrive	Decimal	GB
FreeSpaceDrive	Decimal	GB
FreeSpacePercentage	Decimal	percent (0–100), recomputable in DAX

File naming used by scripts: `StorageCapacity-MMMM-dd-yyyy.csv` (example: `StorageCapacity-July-28-2025.csv`).

Scripts

1) Generate Sample CSVs (Jan–Jul 2025 Mondays)

Use this to create realistic sample data for testing Power BI. Total size per server/drive is fixed (200–800 GB); weekly free space varies and is always less than total.

```
# scripts/Generate-SampleCSVs.ps1
# --------------------------------
# Creates one CSV per Monday from Jan–Jul 2025 under C:
\Output\DriveCapacitySamples

$OutputFolder = 'C:\Output\DriveCapacitySamples'
New-Item -Path $OutputFolder -ItemType Directory -Force | Out-Null

$Servers = @(
  'SQLDB-Prod-01','SQLDB-Prod-02','SQLDB-Dev-01','SQLDB-Dev-02',
  'SQLDB-Test-01','SQLDB-Test-02','SQLDB-Staging-01','SQLDB-Staging-02',
  'SQLDB-Archive-01','SQLDB-Backup-01'
)
$Drives = @('D:\\','E:\\','F:\\')

$start = Get-Date '2025-01-01'
$end   = Get-Date '2025-07-31'
while ($start.DayOfWeek -ne 'Monday') { $start = $start.AddDays(1) }

$rand = [System.Random]::new()
$totalSizeMap = @{}
foreach ($s in $Servers) {
  foreach ($d in $Drives) {
    $size = [Math]::Round(200 + ($rand.NextDouble() * (800 - 200)), 2)
    $totalSizeMap["$s|$d"] = $size
  }
}

function Get-FreeSpaceGB([double]$total, [System.Random]$r) {
  $min = [Math]::Max(1, $total * 0.05); $max = $total * 0.90
  [Math]::Round($min + ($r.NextDouble() * ($max - $min)), 2)
}

for ($dte = $start; $dte -le $end; $dte = $dte.AddDays(7)) {
  $rows = [System.Collections.Generic.List[object]]::new()
```

```
  foreach ($s in $Servers) {
    foreach ($drv in $Drives) {
      $total = $totalSizeMap["$s|$drv"]
      $free  = Get-FreeSpaceGB -total $total -r $rand
      $rows.Add([pscustomobject]@{
        'Capture Date'        = $dte.ToString('yyyy-MM-dd')
        'Capture Day'         = $dte.DayOfWeek.ToString()
        'ServerName'          = $s
        'Drive Letters'       = $drv
        'TotalSizeDrive'      = $total
        'FreeSpaceDrive'      = $free
        'FreeSpacePercentage' = [Math]::Round(($free / $total) * 100, 2)
      }) | Out-Null
    }
  }
  $fileName = "StorageCapacity-{0}.csv" -f $dte.ToString('MMMM-dd-yyyy')
  $rows | Export-Csv (Join-Path $OutputFolder $fileName) -NoTypeInformation -
Encoding UTF8
}
Write-Host "Sample files saved to $OutputFolder"
```

---

2) Collect Drive Capacity from Servers (live)

Queries each server via CIM (Win32_LogicalDisk) and writes a single CSV per run.

```
# scripts/Collect-DriveCapacity.ps1
# --------------------------------
param(
    [string]$ServerListPath = "C:\\temp\\servers.txt",
    [switch]$IncludeAllDrives,
    [string[]]$DriveLetters = @('D:\\','E:\\','F:\\'),
    [string]$OutputFolder = "C:\\Output\\DriveCapacityLive",
    [datetime]$DateOverride,
    [System.Management.Automation.PSCredential]$Credential
)

$null = New-Item -Path $OutputFolder -ItemType Directory -Force -ErrorAction
SilentlyContinue
$AsOf = if ($PSBoundParameters.ContainsKey('DateOverride')) {
$DateOverride.Date } else { (Get-Date).Date }
$targetDeviceIds = $DriveLetters | ForEach-Object { (($_ -replace '\
\','').TrimEnd(':')) + ':' }

if (-not (Test-Path $ServerListPath)) { throw "Server list not found:
$ServerListPath" }
```

```
$Servers = Get-Content $ServerListPath | Where-Object { $_ } | ForEach-Object {
$_.Trim() } | Select-Object -Unique
if (-not $Servers) { throw "No servers found in $ServerListPath" }

$rows = [System.Collections.Generic.List[object]]::new()

function Add-DriveRow {
    param([string]$Server,[string]$DeviceId,[double]$SizeBytes,[double]
$FreeBytes)
    $totalGB = if ($SizeBytes -gt 0) { [math]::Round($SizeBytes/1GB,2) } else {
0 }
    $freeGB  = if ($FreeBytes -ge 0) { [math]::Round($FreeBytes/1GB,2) } else {
0 }
    $pct     = if ($SizeBytes -gt 0) { [math]::Round(($FreeBytes/
$SizeBytes)*100,2) } else { $null }
    $rows.Add([pscustomobject]@{
        'Capture Date'        = $AsOf.ToString('yyyy-MM-dd')
        'Capture Day'         = $AsOf.ToString('dddd')
        'ServerName'          = $Server
        'Drive Letters'       = ($DeviceId.TrimEnd(':') + ':\\')
        'TotalSizeDrive'      = $totalGB
        'FreeSpaceDrive'      = $freeGB
        'FreeSpacePercentage' = $pct
    }) | Out-Null
}

foreach ($s in $Servers) {
    try {
        $sess = if ($Credential) { New-CimSession -ComputerName $s -Credential
$Credential -ErrorAction Stop } else { New-CimSession -ComputerName $s -
ErrorAction Stop }
        $disks = Get-CimInstance -ClassName Win32_LogicalDisk -Filter
"DriveType=3" -CimSession $sess -ErrorAction Stop
        if (-not $IncludeAllDrives) { $disks = $disks | Where-Object {
$targetDeviceIds -contains $_.DeviceID } }
        if ($disks) { foreach ($d in $disks) { Add-DriveRow -Server $s -DeviceId
$d.DeviceID -SizeBytes $d.Size -FreeBytes $d.FreeSpace } } else { Write-Warning
"No matching fixed drives on $s" }
    } catch { Write-Warning "Failed to query $s : $($_.Exception.Message)" }
finally { if ($sess) { $sess | Remove-CimSession } }
}

$fileName = "StorageCapacity-{0}.csv" -f $AsOf.ToString('MMMM-dd-yyyy')
$rows | Export-Csv (Join-Path $OutputFolder $fileName) -NoTypeInformation -
Encoding UTF8
Write-Host "Saved: " (Join-Path $OutputFolder $fileName)
```

Usage

A) Prepare server list

Create `data/servers.txt` with one name per line. Duplicates are OK; script de-duplicates.

```
SQLDB-Prod-01
SQLDB-Prod-02
SQLDB-Dev-01
SQLDB-Dev-02
SQLDB-Test-01
SQLDB-Test-02
SQLDB-Staging-01
SQLDB-Staging-02
SQLDB-Archive-01
SQLDB-Backup-01
```

B) Run the collection script

```
# Default (D: E: F: only)
./scripts/Collect-DriveCapacity.ps1 `
  -ServerListPath ./data/servers.txt `
  -OutputFolder   ./data/output

# All fixed drives
./scripts/Collect-DriveCapacity.ps1 `
  -ServerListPath ./data/servers.txt `
  -OutputFolder   ./data/output `
  -IncludeAllDrives

# Use alternate credentials
./scripts/Collect-DriveCapacity.ps1 `
  -ServerListPath ./data/servers.txt `
  -OutputFolder   ./data/output `
  -Credential (Get-Credential)

# Stamp a specific Monday (for backfilling)
./scripts/Collect-DriveCapacity.ps1 `
  -ServerListPath ./data/servers.txt `
  -OutputFolder   ./data/output `
  -DateOverride "2025-07-28"
```

C) (Optional) Schedule weekly runs

Task Scheduler → Create Task:

- Trigger: Weekly, **Monday**, e.g., 08:00.
- Action: `powershell.exe`
- Arguments:

```
-ExecutionPolicy Bypass -File "C:\\path\\to\\scripts\\Collect-DriveCapacity.ps1" -ServerListPath "C:\\path\\to\\data\\servers.txt" -OutputFolder "C:\\path\\to\\data\\output"
```

- Run whether user is logged on or not. Use a service account with required permissions.

Power BI Setup

Import

1. **Get Data → Folder** → point to `./data/output`.
2. **Combine & Transform** (Power Query):
3. Types: `Capture Date` → Date; `TotalSizeDrive`, `FreeSpaceDrive`, `FreeSpacePercentage` → Decimal.
4. You can drop `FreeSpacePercentage` and compute a measure instead.
5. **Close & Apply**.

Date Table

Create a proper calendar and mark it as the date table.

```
Calendar =
VAR MinDate = COALESCE ( MIN ( 'Disk Capacity Forecasting'[Capture Date] ), DATE ( 2025,1,1 ) )
VAR MaxDate = COALESCE ( MAX ( 'Disk Capacity Forecasting'[Capture Date] ), DATE ( 2025,12,31 ) )
RETURN
ADDCOLUMNS (
    CALENDAR ( MinDate, MaxDate ),
    "Year", YEAR ( [Date] ),
    "Month", FORMAT ( [Date], "MMM" ),
    "YearMonth", FORMAT ( [Date], "YYYY-MM" )
)
```

Create relationship: `Calendar[Date]` → `'Disk Capacity Forecasting'[Capture Date]` and **Mark as date table**.

Measures

> Create each of the following as a **separate** measure on the `'Disk Capacity Forecasting'` table.

```
Total Size GB = SUM ( 'Disk Capacity Forecasting'[TotalSizeDrive] )
Free Space GB = SUM ( 'Disk Capacity Forecasting'[FreeSpaceDrive] )
Free % = DIVIDE ( [Free Space GB], [Total Size GB] )
Used GB = [Total Size GB] - [Free Space GB]
```

Growth + runout:

```
Weekly Delta GB =
VAR PrevUsed = CALCULATE ( [Used GB], DATEADD ( 'Calendar'[Date], -7, DAY ) )
RETURN [Used GB] - PrevUsed

Avg Weekly Growth (8w) =
AVERAGEX (
    DATESINPERIOD ( 'Calendar'[Date], MAX ( 'Calendar'[Date] ), -56, DAY ),
    [Weekly Delta GB]
)

Weeks to Full =
VAR Growth = [Avg Weekly Growth (8w)]
RETURN IF ( Growth <= 0, BLANK(), DIVIDE ( [Free Space GB], Growth ) )

Projected Runout Date =
VAR Weeks = [Weeks to Full]
RETURN IF ( ISBLANK ( Weeks ), BLANK(), MAX ( 'Calendar'[Date] ) + ( Weeks *
7 ) )
```

Latest snapshot KPIs:

```
Latest Used GB = CALCULATE ( [Used GB], LASTDATE ( 'Calendar'[Date] ) )
Latest Free GB = CALCULATE ( [Free Space GB], LASTDATE ( 'Calendar'[Date] ) )
Latest Total Size GB = CALCULATE ( [Total Size GB], LASTDATE
( 'Calendar'[Date] ) )
Latest Free % = DIVIDE ( [Latest Free GB], [Latest Total Size GB] )
```

Useful helpers:

```
Is Latest Date =
VAR LastDate = MAXX ( ALL ( 'Calendar'[Date] ), 'Calendar'[Date] )
```

```
RETURN IF ( MAX ( 'Calendar'[Date] ) = LastDate, 1, 0 )

Total Size GB (ctx) =
IF (
  HASONEVALUE ( 'Disk Capacity Forecasting'[ServerName] ) &&
  HASONEVALUE ( 'Disk Capacity Forecasting'[Drive Letters] ),
  [Total Size GB]
)
```

Recommended Visuals

- **Slicers:** `ServerName`, `Drive Letters`, `Calendar[YearMonth]`.
- **Cards:** `Latest Free GB`, `Latest Free %`, `Weeks to Full`, `Projected Runout Date`.
- **Line chart (Used GB by Date):** X = `Calendar[Date]` (Continuous); Y = `[Used GB]`; add **Forecast** (8–12 weeks). Optionally add `Total Size GB (ctx)` as a capacity line.
- **Line chart (Free GB by Date)** for a second perspective or switch via field parameters.
- **Bar chart:** Top at-risk drives by `[Weeks to Full]` (ascending), with conditional colors (≤4 red, 4–8 amber, else green).
- **Matrix (latest only):** Filter by `Is Latest Date = 1`. Columns: `ServerName`, `Drive Letters`, `Latest Used GB`, `Latest Free GB`, `Avg Weekly Growth (8w)`, `Weeks to Full`, `Projected Runout Date`.

  Forecasts are most trustworthy when a **single server + single drive** is selected.

CI / Linting

- Workflow: `.github/workflows/powershell-lint.yml`
- Settings file: `.config/PSScriptAnalyzerSettings.psd1`
- The badge at the top reflects the status of this workflow. After pushing, replace `your-org/your-repo` in the badge URL with your repo path.

Run locally

```
Install-Module PSScriptAnalyzer -Scope CurrentUser -Force
Invoke-ScriptAnalyzer -Path ./scripts -Recurse -Settings ./.config/
PSScriptAnalyzerSettings.psd1 -Severity Error,Warning -ReportSummary
```

The settings file excludes `PSAvoidUsingWriteHost` and sets consistent indentation/whitespace rules. Adjust as needed.

Troubleshooting

- **Power BI: “The expression is not a valid table expression.”** You pasted a measure into **New table**. Create measures via **Modeling → New measure**. Only the Calendar goes into **New table**.
- **Power BI forecasting blank/odd:** Ensure X-axis is **Continuous**; at least \~10 data points; check that you’re filtered to a single drive for clarity.
- **DAX error around ``:** You likely pasted multiple measures into one. Create each measure separately.
- **Capture script can’t connect:** Check WinRM running, firewall open, and permissions (Admin or WMI access). Try `-Credential`.
- **No drives returned:** If you didn’t use `-IncludeAllDrives`, only `D: E: F:` are captured. Add the switch or change `-DriveLetters`.

Security Notes

- Prefer a dedicated service account with least-privilege WMI rights.
- If scheduling, store the task with **Run whether user is logged on or not** and limit logon rights.
- Avoid committing credential files to Git. Use `-Credential (Get-Credential)` or your enterprise secret store (CyberArk, etc.).

License

MIT (or your org’s standard). Update before publishing.