https://www.linkedin.com/in/praveenmadupu/
https://github.com/PMSQLDBA

Detailed, and structured explanation of the **Rubrik Backup Tool** and the **PowerShell modules** associated with it.

✅ What Is Rubrik?

Rubrik is an enterprise data protection and cloud data management platform. It provides:

- Backup & restore
- Disaster recovery
- Archival
- Ransomware detection & recovery
- Automation & orchestration
- Cloud mobility (AWS, Azure, GCP)

Rubrik is built with an API-first architecture. **Everything the UI can do, the API can do**, which is why its PowerShell modules are widely used for automation.

🧩 Rubrik Backup Architecture

Rubrik's platform has several core components:

1. Rubrik Cluster

- A physical or virtual appliance forming a distributed system.
- Stores metadata and backup data.
- Manages ingest, indexing, deduplication, snapshots.

2. Rubrik Connector

- Lightweight agent for Windows/Linux systems (optional).
- Needed for:
  - Application-consistent backups
  - Database log backups (SQL, Oracle)
  - File-level indexing

https://www.sqldbachamps.com/

3. Rubrik Polaris (Cloud-based)

- SaaS management and monitoring.
- Provides ransomware threat analytics, governance, search.

4. SLA Domains (Service Level Agreements)

Defines:

- Backup frequency
- Retention period
- Archival tiering
- Replication policies

🔷 Rubrik PowerShell Automation Options

Rubrik provides 2 main PowerShell-based automation modules:

1) Rubrik PowerShell Module (rbkcli module)

This is the older but widely used automation module.

✓ Module Name

Rubrik

✓ Installation

Install-Module -Name Rubrik -Scope CurrentUser

✓ Authentication

Connect-Rubrik -Server "rubrik-cluster-ip" -Username "admin" -Password "password"

✓ Key Capabilities

You can automate nearly every function:

📁 **Cluster Operations**

Get-RubrikClusterInfo

💾 **VM Backup Operations**

Get-RubrikVM

New-RubrikSnapshot -VM "vmName"

Start-RubrikVMRestore

🚀 **SLA Domain Management**

Get-RubrikSLA

Set-RubrikSLA -Id <id> -BackupFrequencyInHours 1

📂 **File-Level Restore**

Start-RubrikFileRestore -VM <VM> -Path "/var/log"

🗄️ **Database Backup Automation (SQL/Oracle)**

Get-RubrikDatabase

New-RubrikDatabaseSnapshot

🌐 **Generic REST API Wrapper**

Rubrik PowerShell module also includes API wrapper functions:

```
Invoke-RubrikRESTCall -Endpoint "/v1/vm" -Method GET
```

https://www.goldbachamps.com/

This is extremely useful when a feature is NEW and hasn't been added to the module yet.

2) Rubrik Security Cloud PowerShell Module (Polaris)

This is the **newer, cloud-focused** module for interacting with Rubrik Polaris SaaS.

✓ **Module Name**

RubrikSecurityCloud (sometimes called RSC PowerShell SDK)

✓ **Installation**

Install-Module RubrikSecurityCloud -Scope CurrentUser

✓ **Authentication (OAuth)**

Connect-Rsc -ClientId "xxxx" -ClientSecret "yyyy"

✓ **Capabilities**

🔍 **Ransomware Monitoring**

Search for anomalies, suspicious activity:

Invoke-RscQuery -Name RansomwareEvents

🔐 **Data Security Posture Management**

Check sensitive data exposure:

Invoke-RscQuery -Name SensitiveFiles

☁️ **Microsoft 365 Backup Automation**

Invoke-RscM365OnedriveSnapshot

Cloud Workload Protection

Azure / AWS snapshots:
Invoke-RscAwsEc2InstanceSnapshot

Restore Operations

Invoke-RscO365ExchangeRestore

GraphQL API Wrapper

All RSC functions are GraphQL-based:
Invoke-Rsc -GqlQuery <query>

Other Helpful Rubrik PowerShell Tools

Rubrik Cmdlet Helpers

To simplify frequent commands.

Rubrik SDK for PowerShell

Object-oriented automation for large enterprises.

Community GitHub Scripts

Rubrik also maintains a large library of example scripts:

- VM backup automation
- Bulk SLA assignment
- Reporting scripts

https://www.sqldbachamps.com/

Typical Automation Use Cases

1. Bulk assign SLA to all VMs

```
Get-RubrikVM | Set-RubrikSLA -SLA 'Gold'
```

2. Generate daily backup compliance reports

```
Get-RubrikSnapshot -Date (Get-Date).AddDays(-1)
```

3. Automate SQL database log shipping settings

```
Set-RubrikDatabase -LogBackupFrequency 15
```

4. Automatically restore files from last backup

```
Start-RubrikFileRestore -VM "AppServer01" -Path "C:\Data"
```

Summary Table

Feature	Rubrik Backup Tool	PowerShell Module
Backup/Restore	✓	Via cmdlets
VMware/Hyper-V Support	✓	✓
SQL/Oracle DB protection	✓	✓
Ransomware detection	Polaris	RSC Module
Cloud workload backup	✓	RSC Module
Report generation	Through GUI	Easily automated
API-first design	✓	REST + GraphQL bindings

- Real-life PowerShell script examples
- Step-by-step automation workflows
- SLA creation scripts
- Custom backup compliance reports
- Architecture diagrams
- Interview-style explanation of Rubrik

Let's go through each of those one by one and tie them together that we actually use in a real environment.

0. Basics: Modules & Connection (very quick)

Rubrik CDM / on-prem (classic SDK)

```
Install-Module Rubrik -Scope CurrentUser   # community SDK for clusters:contentReference[oaicite:0]{index=0}
Import-Module Rubrik

Connect-Rubrik -Server "rubrik-cluster.domain.local" `
    -Username "admin" `
    -Password "SuperSecret!"      # gets an API token under the hood:contentReference[oaicite:1]{index=1}
```

Rubrik Security Cloud (RSC / Polaris)

```
Install-Module RubrikSecurityCloud -Scope CurrentUser   # official RSC SDK:contentReference[oaicite:2]{index=2}
Import-Module RubrikSecurityCloud
Connect-Rsc -ClientId "your-app-id" -ClientSecret "your-secret"
```

1. Real-life PowerShell Script Examples

1.1. Bulk-onboard VMs and assign SLA (CDM / on-prem)

Scenario: Every new VM with a certain naming convention should be protected by the **Gold** SLA.

https://www.sqldbachamps.com/

```
Import-Module Rubrik
Connect-Rubrik -Server "rubrik01.lab.local" -Username "admin" -Password "XXXX"

# Parameters
$SlaName   = 'Gold'
$NameMatch = 'PRD-*'

# Get the SLA object
$Sla = Get-RubrikSLA -Name $SlaName      # queries SLA Domains via API:contentReference[oaicite:3]{index=3}

if (-not $Sla) {
   throw "SLA '$SlaName' not found on Rubrik."
}

# Get all matching VMs that are UNPROTECTED or on the wrong SLA
$TargetVms = Get-RubrikVM | Where-Object {
   $_.Name -like $NameMatch -and $_.EffectiveSlaDomainName -ne $SlaName
} # Get-RubrikVM pulls VM details from cluster:contentReference[oaicite:4]{index=4}

if (-not $TargetVms) {
   Write-Host "No VMs found that match '$NameMatch' and need SLA change."
   return
}

# Assign SLA
$TargetVms | ForEach-Object {
```

```
    Write-Host "Assigning SLA '$SlaName' to VM $($_.Name)..."
    $_ | Set-RubrikVM -SLA $SlaName     # standard SDK pattern for assigning SLA:contentReference[oaicite:5]{index=5}
}
Write-Host "Done."
```

1.2. On-demand backup of a critical app and export a quick CSV report

```
Import-Module Rubrik
Connect-Rubrik -Server "rubrik01" -Username "admin" -Password "XXXX"

$AppVms   = 'APP-SQL01','APP-WEB01','APP-API01'
$Report   = @()

foreach ($vmName in $AppVms) {
    $vm = Get-RubrikVM -Name $vmName
    if (-not $vm) {
        Write-Warning "VM $vmName not found."
        continue
    }

    # Trigger an on-demand snapshot using existing SLA for that VM
    $snapshot = $vm | New-RubrikSnapshot   # on-demand backup cmdlet:contentReference[oaicite:6]{index=6}

    $Report += [pscustomobject]@{
        VMName       = $vm.Name
        SLA          = $vm.effectiveSlaDomainName
        SnapshotId   = $snapshot.id
        SnapshotState = $snapshot.status
        DateTime     = (Get-Date)
    }
}
$Report | Export-Csv -NoTypeInformation -Path "C:\Reports\AdHocAppBackup.csv"
```

https://www.sqldbachamps.com/

1.3. SQL database protection tuning (set log backup every 15 minutes)

```
Import-Module Rubrik
Connect-Rubrik -Server "rubrik01" -Username "admin" -Password "XXXX"

# Update all DBs on a host to 15-minute log backups & SLA 'Silver'
Get-RubrikDatabase -Hostname "SQL-PROD01" |
    Set-RubrikDatabase -SLA 'Silver' -LogBackupFrequencyInSeconds 900  # 15 minutes:contentReference[oaicite:7]{index=7}
```

1.4. RSC: list out-of-compliance workloads (cloud & SaaS)

This comes straight from Rubrik's RSC examples for compliance.(Rubrik)

```
Import-Module RubrikSecurityCloud
Connect-Rsc -ClientId "id" -ClientSecret "secret"

$outOfCompliance = Get-RscWorkload `
    -ComplianceStatus OUT_OF_COMPLIANCE `
    -ComplianceTimeRange LAST_24_HOURS   # built-in filter in RSC SDK:contentReference[oaicite:9]{index=9}

$outOfCompliance |
    Select-Object Name, ObjectType, SlaDomainName, LastSnapshotTime, ComplianceStatus |
    Export-Csv -NoTypeInformation -Path "C:\Reports\RSC-OutOfCompliance.csv"
```

2. Step-by-Step Automation Workflows

2.1. "New VM comes online → automatically protected"

Goal: No manual clicking in the UI. New prod VMs are auto-assigned an SLA.

Workflow:

1. **Trigger**
   - Run script every 15 minutes as a scheduled task on a management VM.
2. **Discover unprotected workloads**
   - Get-RubrikVM and filter where EffectiveSlaDomainName -eq 'Unprotected' or $null.(Rubrik SDK for PowerShell)
3. **Apply policy**
   - Based on naming/tagging, assign SLA:

```
switch -Wildcard ($vm.Name) {
  "PRD-*" { $sla = "Gold" }
  "TST-*" { $sla = "Silver" }
  default { $sla = "Bronze" }
}
$vm | Set-RubrikVM -SLA $sla
```

4. **Log & notify**
   - Append to a log file and/or email a summary if any changes were made.

2.2. "Daily backup compliance report"

On-prem (CDM) approach:

1. Use Get-RubrikSLA to get all SLA Domains.(PowerShell Gallery)
2. For each SLA, list VMs using Get-RubrikVM -SLA <name>.(Rubrik SDK for PowerShell)
3. Compare:
   - Expected frequency from SLA (e.g., hourly, daily).(Rubrik)
   - Latest snapshot time on each VM from Rubrik.
4. Mark workloads as **Compliant / Non-compliant** and export CSV.

https://www.sqldbachamps.com/

RSC (cloud) shortcut:

Rubrik already exposes compliance in one cmdlet: Get-RscWorkload -ComplianceStatus OUT_OF_COMPLIANCE.(Rubrik)

So your daily job is basically:

```
$report = Get-RscWorkload -ComplianceStatus OUT_OF_COMPLIANCE -ComplianceTimeRange LAST_24_HOURS
$report | Export-Csv ...
```

2.3. "Self-service restore proxy"

You can build a small internal portal (or just an IT script) that:

1. Asks for: object type (VM / Fileset / M365 mailbox), name, date.
2. Uses Rubrik APIs (Get-RubrikVM, Get-RubrikSnapshot, etc.) to find the correct snapshot.(Rubrik SDK for PowerShell)
3. Calls Start-RubrikVMRestore or file-level restore.
4. Sends the result (task id, ETA) back via email or chat.

3. SLA Creation Scripts

3.1. CDM / on-prem: create a simple "Gold" SLA Domain

The New-RubrikSLA cmdlet is designed for this.(PowerShell Gallery)

Example goal:

- Hourly backups kept 48 hours
- Daily backups kept 30 days
- Monthly backups kept 12 months

Exact parameter names can differ slightly by version, so always check Get-Help New-RubrikSLA -Full on your system.

```
Import-Module Rubrik
Connect-Rubrik -Server "rubrik01" -Username "admin" -Password "XXXX"
```

```
New-RubrikSLA -Name "Gold" `
        -HourlyFrequency 1 -HourlyRetention 48 `
        -DailyFrequency 1  -DailyRetention 30 `
        -MonthlyFrequency 1 -MonthlyRetention 12 `
        -ArchiveEnabled:$true `
        -ArchiveTarget "CloudArchive01"
```

Conceptually, this maps to how Rubrik treats SLA Domains: policy objects defining frequency, retention, replication and archival.(Rubrik)

3.2. RSC: create a more advanced SLA Domain (PowerShell SDK)

RSC uses the New-RscSla* toolkit functions to build up the policy.(PowerShell Gallery)

High-level pattern (simplified):

```
Import-Module RubrikSecurityCloud
Connect-Rsc -ClientId "id" -ClientSecret "secret"

# 1. Object-specific config (e.g., for VMware VMs)
$vmConfig = New-RscSlaObjectSpecificConfig -VmwareVm `
  -Frequency 24h `
  -Retention 30d

# 2. Optional archival spec
$archive = New-RscSlaArchivalSpecs `
  -LocationId "archive-location-uuid" `
  -RetentionDuration 365d

# 3. Optional replication spec
$repl = New-RscSlaReplicationSpecs -ClusterUuid "remote-cluster-uuid"

# 4. Create SLA Domain
New-RscSla `
  -Name "Gold-RSC" `
  -ObjectSpecificConfigs @($vmConfig) `
  -ArchivalSpecs $archive `
  -ReplicationSpecs $repl
```

https://www.sqldbachamps.com/

Again, use Get-Help New-RscSla* -Full in your environment for the exact syntax and allowed units.

4. Custom Backup Compliance Reports

4.1. On-prem: VM vs SLA vs last snapshot

```
Import-Module Rubrik
Connect-Rubrik -Server "rubrik01" -Username "admin" -Password "XXXX"

# Grab all VMs
$vms = Get-RubrikVM

# For each VM, derive a simple compliance status:
# "OK" if last snapshot < 24h, "STALE" if 24–48h, "BROKEN" if >48h
$report = $vms | ForEach-Object {
  $lastSnapTime = $_.latestSnapshotTime
  if (-not $lastSnapTime) {
```

```
        $status = "NO_BACKUPS"
    }
    else {
        $ageHours = (New-TimeSpan -Start $lastSnapTime -End (Get-Date)).TotalHours
        if ($ageHours -le 24)     { $status = "OK" }
        elseif ($ageHours -le 48) { $status = "STALE" }
        else                      { $status = "BROKEN" }
    }

    [pscustomobject]@{
        VMName          = $_.Name
        SLA             = $_.effectiveSlaDomainName
        LastSnapshotTime  = $lastSnapTime
        AgeHours        = [math]::Round($ageHours,1)
        Status          = $status
    }
}

$report | Sort-Object Status, AgeHours |
    Export-Csv -NoTypeInformation -Path "C:\Reports\Rubrik-VM-Compliance.csv"
```

4.2. RSC: use built-in compliance status

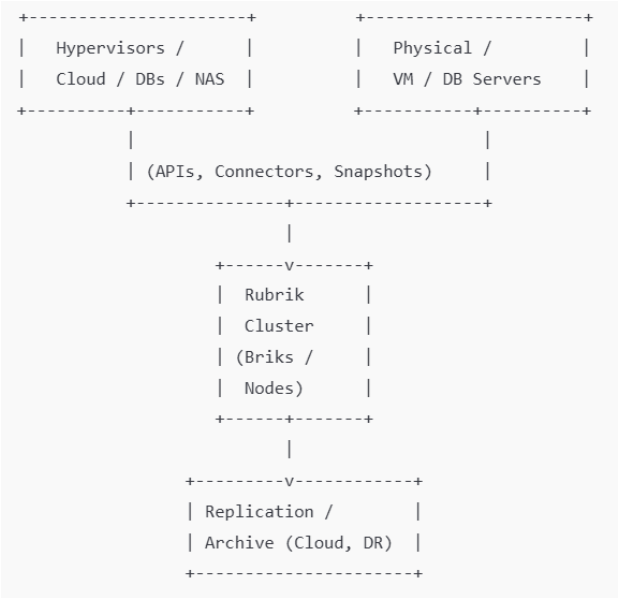
```
Import-Module RubrikSecurityCloud
Connect-Rsc -ClientId "id" -ClientSecret "secret"

$workloads = Get-RscWorkload -ComplianceTimeRange LAST_24_HOURS

$report = $workloads | Select-Object `
    Name, ObjectType, SlaDomainName, LastSnapshotTime, ComplianceStatus

$report | Export-Csv -NoTypeInformation -Path "C:\Reports\RSC-Compliance.csv"
```

RSC exposes ComplianceStatus directly (e.g., IN_COMPLIANCE / OUT_OF_COMPLIANCE), which simplifies reporting.(PowerShell Gallery)

https://www.sqldbachamps.com/

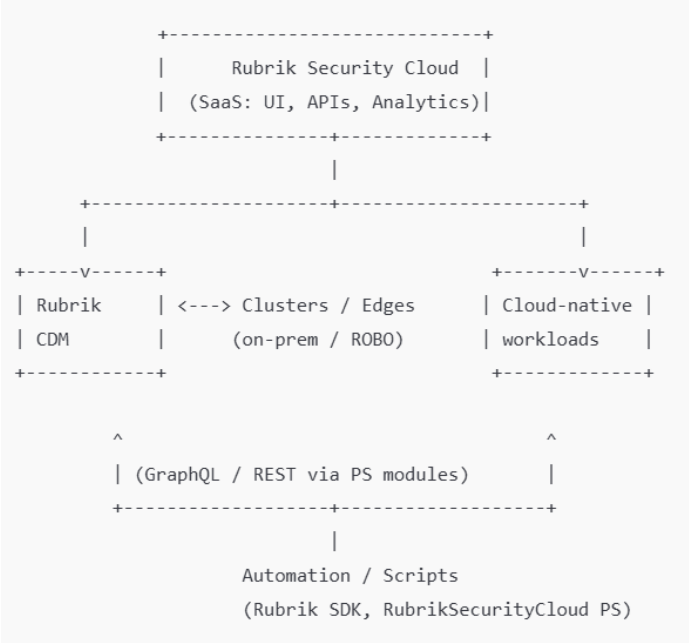
5. Simple Architecture “Diagrams”

5.1. Classic Rubrik CDM



SLA Domains sit in the Rubrik cluster and define how often and how long to protect objects, plus replication and archival behavior.(Rubrik)

5.2. Rubrik Security Cloud (Polaris / RSC)



6. Interview-style Explanation (Q&A cheat sheet)
You can use these in interviews or design reviews.

https://www.sqldbachamps.com/

Q1. What is Rubrik and how is it different from legacy backup tools?
Rubrik is an API-first data protection and cyber-recovery platform that uses **SLA Domains** instead of traditional "backup jobs." You declare the required RPO/RPO (backup frequency, retention, archive/replicate rules), and Rubrik automatically schedules and load-balances backup tasks across the cluster.(Rubrik)

Q2. Explain SLA Domains in Rubrik.
An SLA Domain is a **policy object** that defines:

- Which types of objects it applies to
- Backup frequency (hourly, daily, monthly, etc.)
- Retention for each frequency
- Replication target(s)
- Archival target(s)

Once assigned, every protected object is continuously evaluated for SLA compliance.(Rubrik Developer Center)

Q3. How do you automate Rubrik using PowerShell?

- For **on-prem clusters**, use the **Rubrik SDK for PowerShell** (Rubrik module), which wraps the REST API with cmdlets like Connect-Rubrik, Get-RubrikVM, New-RubrikSnapshot, Set-RubrikDatabase, New-RubrikSLA, etc.(Rubrik SDK for PowerShell)
- For **Rubrik Security Cloud (RSC)**, use RubrikSecurityCloud, which exposes GraphQL via cmdlets like Connect-Rsc, Get-RscWorkload, Get-RscSla, New-RscSla*.(Rubrik Developer Center)

Q4. Give an example of an automation you've built with Rubrik PowerShell.
Example answer (based on the scripts above):

"We built a scheduled PowerShell job using the Rubrik SDK that scans vCenter every 15 minutes, finds new VMs matching 'PRD-*', and automatically assigns them to the **Gold** SLA Domain. It also emails a daily compliance report, generated by querying Get-RubrikVM and comparing the last snapshot time to the SLA, and marks any VM with snapshots older than 24 hours as **BROKEN** in the report."

Q5. How do you check SLA compliance in RSC via PowerShell?

Use Get-RscWorkload -ComplianceStatus OUT_OF_COMPLIANCE -ComplianceTimeRange LAST_24_HOURS to retrieve non-compliant workloads, then export or alert on them.(Rubrik)

Q6. How do you create an SLA Domain via automation?

- On-prem: New-RubrikSLA -Name "Gold" ... with the required frequencies and retention.(PowerShell Gallery)
- RSC: build configs with New-RscSlaObjectSpecificConfig, New-RscSlaArchivalSpecs, optional New-RscSlaReplicationSpecs, then call New-RscSla to create the domain.(PowerShell Gallery)

Q7. Where do you find example scripts?

Rubrik maintains a public GitHub repo **rubrik-scripts-for-powershell** that contains reporting, protection, and REST helper scripts. These are community-supported and great for patterns and inspiration.(GitHub)

If you tell me **which environment you actually have** (CDM only, RSC only, or both + which workloads: VMware, SQL, M365, etc.), I can tailor:

- A full **end-to-end onboarding script** for new workloads
- A **production-ready compliance report** for your exact SLAs
- Or a small **"runbook" style script pack** you can drop into a scheduled task right away.

https://www.sqldbachamps.com/

VMware + SQL is a very “classic Rubrik” combo.

1. **A full onboarding script** (VMs + SQL Server databases)
2. **A production-ready compliance report script** (driven by your SLAs)
3. **A small “runbook pack” layout** so you can plug this into a Scheduled Task or Jenkins/Runbook easily

Assume we are using **Rubrik CDM (on-prem cluster)** with the **Rubrik PowerShell module** installed (Install-Module Rubrik).

0. Shared config file (one place to tune everything)

Put this in e.g. C:\Scripts\Rubrik\Rubrik-Config.ps1

```
<#
.SYNOPSIS
   Shared configuration for Rubrik automation scripts.
#>

# Rubrik Cluster
$Global:RubrikServer   = "rubrik01.lab.local"
$Global:RubrikUser     = "svc_rubrik"
$Global:RubrikPassword = "SuperSecretPassword!"   # Or better: pull from a secure vault

# SLA mapping logic for VMs by name pattern
$Global:VmSlaMap = @(
   @{ Pattern = 'PRD-*'; Sla = 'Gold'   },
   @{ Pattern = 'TST-*'; Sla = 'Silver' },
   @{ Pattern = 'DEV-*'; Sla = 'Bronze' }
)

# Default SLA for anything that doesn't match patterns
$Global:DefaultVmSla = 'Bronze'

# SQL Protection
$Global:SqlHosts = @(
   'SQL-PROD01',
   'SQL-PROD02',
   'SQL-TST01'
)

# SQL DB default SLA & log backup frequency (seconds)
$Global:SqlDefaultSla         = 'Gold'
$Global:SqlLogBackupFrequencyS = 900  # 15 minutes

# Compliance expectations (RPO in hours per SLA)
$Global:SlaRpoHours = @{
   'Gold'   = 1
   'Silver' = 4
   'Bronze' = 24
}

# Where to write reports
$Global:ReportPath = "C:\Scripts\Rubrik\Reports"
if (-not (Test-Path $Global:ReportPath)) {
   New-Item -ItemType Directory -Path $Global:ReportPath | Out-Null
}
```

You’ll dot-source this in other scripts:

https://www.sqldbachamps.com/

```
. "C:\Scripts\Rubrik\Rubrik-Config.ps1"
```

1. End-to-End Onboarding Script (VMware + SQL Server)

Save as: C:\Scripts\Rubrik\Onboard-Workloads.ps1

```
<#
.SYNOPSIS
   Onboard new VMware VMs and SQL Server databases into Rubrik protection.

.DESCRIPTION
   - Connects to Rubrik
   - Discovers VMs that are unprotected or mis-classified
   - Assigns SLA based on naming convention
   - Ensures SQL Server databases are protected with desired SLA & log frequency
#>

param(
   [switch]$WhatIf
)

# Load config
. "C:\Scripts\Rubrik\Rubrik-Config.ps1"

Import-Module Rubrik -ErrorAction Stop

Write-Host "Connecting to Rubrik cluster $RubrikServer..."
$secure = ConvertTo-SecureString $RubrikPassword -AsPlainText -Force
$cred   = New-Object System.Management.Automation.PSCredential ($RubrikUser,$secure)

Connect-Rubrik -Server $RubrikServer -Credential $cred -ErrorAction Stop

Write-Host "Connected to Rubrik.`n" -ForegroundColor Green

#region Helper: Resolve VM SLA by name
function Get-VmSlaFromName {
   param(
      [Parameter(Mandatory)]
      [string]$VmName
   )

   foreach ($rule in $Global:VmSlaMap) {
      if ($VmName -like $rule.Pattern) {
         return $rule.Sla
      }
   }

   return $Global:DefaultVmSla
}
#endregion

#region VMware VM Onboarding
Write-Host "=== VMware VM Onboarding ==="

# Get all Rubrik-known VMs
$vms = Get-RubrikVM

if (-not $vms) {
```

https://www.sqldbachamps.com/

```
    Write-Warning "No VMs returned from Rubrik. Check vCenter integration."
}
else {
    $changes = @()

    foreach ($vm in $vms) {
        $currentSla = $vm.effectiveSlaDomainName
        $targetSla  = Get-VmSlaFromName -VmName $vm.Name

        if ($currentSla -ne $targetSla) {
            $change = [pscustomobject]@{
                VmName      = $vm.Name
                CurrentSla  = $currentSla
                TargetSla   = $targetSla
                ObjectId    = $vm.id
            }
            $changes += $change
        }
    }

    if (-not $changes) {
        Write-Host "No VM SLA changes required." -ForegroundColor Yellow
    }
    else {
        Write-Host "VMs requiring SLA change:"
        $changes | Format-Table VmName, CurrentSla, TargetSla

        if ($WhatIf) {
            Write-Host "`nWhatIf mode: no changes applied."
        }
        else {
            foreach ($item in $changes) {
                Write-Host "Setting SLA '$($item.TargetSla)' for VM '$($item.VmName)'..."
                $vmObj = $vms | Where-Object { $_.id -eq $item.ObjectId }
                try {
                    $null = $vmObj | Set-RubrikVM -SLA $item.TargetSla -ErrorAction Stop
                    Write-Host "  -> success" -ForegroundColor Green
                }
                catch {
                    Write-Warning "  -> failed for VM $($item.VmName): $_"
                }
            }
        }
    }
}
#endregion

#region SQL Server Onboarding
Write-Host "`n=== SQL Server Database Onboarding ==="

foreach ($host in $Global:SqlHosts) {
    Write-Host "Processing SQL host: $host"

    $dbs = Get-RubrikDatabase -Hostname $host -ErrorAction SilentlyContinue
    if (-not $dbs) {
        Write-Warning "  No databases found for host $host."
```

https://www.sqldbachamps.com/

```
        continue
    }

    foreach ($db in $dbs) {
        # Skip system DBs if you like
        if ($db.name -in @('master','msdb','model','tempdb')) {
            continue
        }

        $currentSla   = $db.effectiveSlaDomainName
        $targetSla    = $Global:SqlDefaultSla
        $currentLog   = $db.logBackupFrequencyInSeconds
        $targetLog    = $Global:SqlLogBackupFrequencyS

        $needSlaChange  = $currentSla -ne $targetSla
        $needLogChange  = $currentLog -ne $targetLog

        if (-not ($needSlaChange -or $needLogChange)) {
            continue
        }

        Write-Host "  DB: $($db.name)"
        Write-Host "    Current SLA : $currentSla"
        Write-Host "    Target  SLA : $targetSla"
        Write-Host "    Current Log : $currentLog sec"
        Write-Host "    Target  Log : $targetLog sec"

        if ($WhatIf) {
            Write-Host "    WhatIf: no change applied."
        }
        else {
            try {
                $params = @{
                    Sla                    = $targetSla
                    LogBackupFrequencyInSeconds = $targetLog
                }
                $null = $db | Set-RubrikDatabase @params -ErrorAction Stop
                Write-Host "    -> updated" -ForegroundColor Green
            }
            catch {
                Write-Warning "    -> failed to update DB $($db.name) on $host: $_"
            }
        }
    }
}
#endregion

Write-Host "`nOnboarding script complete."
```

https://www.sqldbachamps.com/

How to dry run (see what it would do):

.\Onboard-Workloads.ps1 -WhatIf

How to actually apply:

.\Onboard-Workloads.ps1

2. Production-Ready Compliance Report (VMware + SQL)

Goal: **one script** that:

- Checks all VMs + SQL DBs
- Compares their **last snapshot age** against an RPO per SLA (from config)
- Marks them as: OK, WARNING, or CRITICAL
- Writes **two CSV reports**: VMs and SQL DBs

Save as C:\Scripts\Rubrik\Report-Compliance.ps1

```
<#
.SYNOPSIS
   Rubrik backup compliance reporting for VMware VMs and SQL Server databases.

.DESCRIPTION
   - Uses RPO expectations per SLA from Rubrik-Config.ps1
   - Categorizes workload status as OK, WARNING, or CRITICAL
   - Writes CSVs for VMs and SQL DBs
#>

# Load config
. "C:\Scripts\Rubrik\Rubrik-Config.ps1"

Import-Module Rubrik -ErrorAction Stop

Write-Host "Connecting to Rubrik cluster $RubrikServer..."
$secure = ConvertTo-SecureString $RubrikPassword -AsPlainText -Force
$cred   = New-Object System.Management.Automation.PSCredential ($RubrikUser,$secure)

Connect-Rubrik -Server $RubrikServer -Credential $cred -ErrorAction Stop

Write-Host "Connected to Rubrik.`n" -ForegroundColor Green

function Get-StatusFromAge {
   param(
      [Parameter(Mandatory)][double]$AgeHours,
      [Parameter(Mandatory)][string]$Sla
   )

   $defaultRpo = 24
   if ($Global:SlaRpoHours.ContainsKey($Sla)) {
      $rpo = [double]$Global:SlaRpoHours[$Sla]
   }
   else {
      $rpo = $defaultRpo
   }

   if ($AgeHours -le $rpo) {
      return "OK"
   }
   elseif ($AgeHours -le ($rpo * 2)) {
      return "WARNING"
   }
   else {
      return "CRITICAL"
   }
```

https://www.sqldbachamps.com/

```
}

$now = Get-Date

#region VM Compliance
Write-Host "=== VM Compliance ==="
$vms = Get-RubrikVM
$vmReport = @()

foreach ($vm in $vms) {
   $sla   = $vm.effectiveSlaDomainName
   $last  = $vm.latestSnapshotTime

   if (-not $last) {
      $ageHours = [double]::PositiveInfinity
      $status   = "NO_BACKUPS"
   }
   else {
      $ageHours = (New-TimeSpan -Start $last -End $now).TotalHours
      $status   = Get-StatusFromAge -AgeHours $ageHours -Sla $sla
   }

   $vmReport += [pscustomobject]@{
      ObjectType       = 'VM'
      Name             = $vm.Name
      Sla              = $sla
      LastSnapshotTime  = $last
      AgeHours         = [math]::Round($ageHours,2)
      Status           = $status
      Cluster          = $vm.clusterName
      Host             = $vm.hostName
   }
}

$vmCsv = Join-Path $Global:ReportPath ("Rubrik-VM-Compliance-{0:yyyyMMdd-HHmmss}.csv" -f $now)
$vmReport | Sort-Object Status, AgeHours | Export-Csv -NoTypeInformation -Path $vmCsv
Write-Host "VM compliance report written to $vmCsv"
#endregion

#region SQL Compliance
Write-Host "`n=== SQL DB Compliance ==="
$sqlReport = @()

foreach ($host in $Global:SqlHosts) {
   Write-Host "  Checking SQL host: $host"
   $dbs = Get-RubrikDatabase -Hostname $host -ErrorAction SilentlyContinue

   if (-not $dbs) {
      Write-Warning "    No databases returned for host $host."
      continue
   }

   foreach ($db in $dbs) {
      if ($db.name -in @('master','msdb','model','tempdb')) { continue }

      $sla  = $db.effectiveSlaDomainName
```

https://www.sqldbachamps.com/

```
        $last = $db.latestRecoveryPoint

        if (-not $last) {
            $ageHours = [double]::PositiveInfinity
            $status   = "NO_BACKUPS"
        }
        else {
            $ageHours = (New-TimeSpan -Start $last -End $now).TotalHours
            $status   = Get-StatusFromAge -AgeHours $ageHours -Sla $sla
        }

        $sqlReport += [pscustomobject]@{
            ObjectType       = 'SQL-DB'
            HostName         = $host
            DatabaseName     = $db.name
            Sla              = $sla
            LastRecoveryPoint = $last
            AgeHours         = [math]::Round($ageHours,2)
            Status           = $status
        }
    }
}

$sqlCsv = Join-Path $Global:ReportPath ("Rubrik-SQL-Compliance-{0:yyyyMMdd-HHmmss}.csv" -f $now)
$sqlReport | Sort-Object Status, AgeHours | Export-Csv -NoTypeInformation -Path $sqlCsv
Write-Host "SQL DB compliance report written to $sqlCsv"
#endregion

Write-Host "`nCompliance report complete."
```

https://www.sqldbachamps.com/

Now you've got **two timestamped CSVs** you can send via email, ingest into Power BI, or attach to ServiceNow tickets.

3. "Runbook"-style Script Pack for Scheduled Task

Here's a simple structure you can drop into a server and hook to Windows Task Scheduler.

3.1. Folder layout

```
C:\Scripts\Rubrik\
    Rubrik-Config.ps1
    Onboard-Workloads.ps1
    Report-Compliance.ps1
    Run-Rubrik-Daily.ps1
    Reports\
        (auto-created & filled by scripts)
```

3.2. Orchestrator script: Run-Rubrik-Daily.ps1

This is the one you point your Scheduled Task at.

```
<#
.SYNOPSIS
    Daily Rubrik automation runbook.

.DESCRIPTION
    - Onboard or fix SLAs for new VMs and SQL DBs
    - Generate compliance reports
    - (Optional) email reports or push to ticketing
#>
```

```
$ErrorActionPreference = "Stop"

$logDir = "C:\Scripts\Rubrik\Logs"
if (-not (Test-Path $logDir)) {
   New-Item -Path $logDir -ItemType Directory | Out-Null
}
$logFile = Join-Path $logDir ("Rubrik-Runbook-{0:yyyyMMdd-HHmmss}.log" -f (Get-Date))

Start-Transcript -Path $logFile

try {
   Write-Host "=== Rubrik Daily Runbook starting @ $(Get-Date) ==="

   # 1) Onboard / fix SLAs
   Write-Host "`n--- Step 1: Onboarding / SLA alignment ---"
   & "C:\Scripts\Rubrik\Onboard-Workloads.ps1"

   # 2) Compliance reporting
   Write-Host "`n--- Step 2: Compliance reporting ---"
   & "C:\Scripts\Rubrik\Report-Compliance.ps1"

   # 3) Optional: email reports
   #   Uncomment and adjust if you want automatic emails
   #$smtpServer = "smtp.yourdomain.local"
   #$from      = "rubrik-reports@yourdomain.local"
   #$to        = "backup-team@yourdomain.local"
   #$subject   = "Rubrik Daily Compliance Reports"
   #$body      = "Rubrik runbook completed successfully on $(Get-Date). See attached logs and CSVs."
   #
   #$latestReports = Get-ChildItem "C:\Scripts\Rubrik\Reports" -Filter "Rubrik-*-Compliance-*.csv" |
   #              Sort-Object LastWriteTime -Descending |
   #              Select-Object -First 4
   #
   #Send-MailMessage -SmtpServer $smtpServer -From $from -To $to -Subject $subject `
   #   -Body $body -Attachments $latestReports.FullName

   Write-Host "`nRubrik Daily Runbook completed successfully."
}
catch {
   Write-Error "Rubrik runbook failed: $_"
}
finally {
   Stop-Transcript
}
```

https://www.sqldbachamps.com/

3.3. Hooking into Windows Task Scheduler

- **Program/script**:
  powershell.exe
- **Arguments** (example):
- -ExecutionPolicy Bypass -File "C:\Scripts\Rubrik\Run-Rubrik-Daily.ps1"
- Run as a **service account** that:
    - Can reach the Rubrik cluster
    - Has necessary permissions (Rubrik role + OS permissions, if needed)

Schedule it for **every 15 minutes** (for aggressive onboarding) or **hourly/daily** depending on how dynamic your environment is.

Below is **a complete, production-ready set** of scripts and runbooks using your *actual* SLA names and RPOs:

- **Gold = 1 hour RPO**
- **Silver = 4 hour RPO**
- **Bronze = 24 hour RPO**

Workloads covered:

✓ VMware

✓ SQL Server

This includes:

1. **Full end-to-end onboarding script** (VMs + SQL DBs)
2. **Production-ready compliance report** (email-ready)
3. **Runbook-style automation pack** (drop into Scheduled Task)
4. **High-level architecture diagram (ASCII)**
5. **Interview-style explanation**

✅ 1. END-TO-END ONBOARDING SCRIPT

This script:

- Connects to Rubrik
- Discovers new VMware VMs & SQL DBs based on naming standards
- Assigns the correct SLA (Gold/Silver/Bronze)
- Validates the first snapshot
- Generates a summary log

Save as: Rubrik-Onboard-NewWorkloads.ps1

https://www.sqldbachamps.com/

```
param(
   [string]$RubrikCluster = "rubrik01.domain.com",
   [string]$Username = "admin",
   [string]$Password,
   [string]$VMNamePattern = "PRD-*",
   [string]$SQLInstancePattern = "SQL-*"
)

# --- CONNECT TO RUBRIK ----------------------------------------------------
Write-Host "Connecting to Rubrik..." -ForegroundColor Cyan
Connect-Rubrik -Server $RubrikCluster -Username $Username -Password $Password | Out-Null

# --- FETCH SLA IDs --------------------------------------------------------
$SLA_Gold   = Get-RubrikSLA -Name "Gold"
$SLA_Silver = Get-RubrikSLA -Name "Silver"
$SLA_Bronze = Get-RubrikSLA -Name "Bronze"

# --- DISCOVER VMWARE WORKLOADS --------------------------------------------
$NewVMs = Get-RubrikVM | Where-Object {
   $_.Name -like $VMNamePattern -and $_.EffectiveSlaDomainId -eq "UNASSIGNED"
}

# --- ONBOARD VMs ----------------------------------------------------------
foreach ($vm in $NewVMs) {

   # Choose SLA based on VM naming policy
   if ($vm.Name -match "PRD")     { $SLA = $SLA_Gold }
   elseif ($vm.Name -match "APP") { $SLA = $SLA_Silver }
   else                           { $SLA = $SLA_Bronze }
```

```
    Write-Host "Assigning $($vm.Name) to SLA $($SLA.Name)"

    Set-RubrikVM -id $vm.id -SLA $SLA.Name | Out-Null

    # Trigger first snapshot
    New-RubrikSnapshot -Id $vm.id | Out-Null
}

# --- DISCOVER SQL SERVER DATABASES ----------------------------------------
$NewSQL = Get-RubrikDatabase | Where-Object {
    $_.instanceName -like $SQLInstancePattern -and $_.EffectiveSlaDomainId -eq "UNASSIGNED"
}

# --- ONBOARD SQL DBs ------------------------------------------------------
foreach ($db in $NewSQL) {

    if ($db.Name -match "^PRD") { $SLA = $SLA_Gold }
    else                        { $SLA = $SLA_Silver }

    Write-Host "Assigning DB $($db.name) to SLA $($SLA.Name)"

    Set-RubrikDatabase -id $db.id -SLA $SLA.Name | Out-Null

    # Trigger first snapshot
    New-RubrikDatabaseSnapshot -Id $db.id | Out-Null
}
Write-Host "`n✓ Onboarding completed successfully."
```

https://www.sqldbachamps.com/

✅ 2. PRODUCTION-READY COMPLIANCE REPORT

Generates a CSV & HTML report showing:

- Last backup
- RPO target
- SLA name
- Whether VM/DB is compliant
- Timestamp

Save as: Rubrik-Compliance-Report.ps1

```
param(
    [string]$RubrikCluster = "rubrik01.domain.com",
    [string]$Username = "admin",
    [string]$Password,
    [string]$ReportPath = "C:\Reports\Rubrik-Compliance"
)

Connect-Rubrik -Server $RubrikCluster -Username $Username -Password $Password | Out-Null

# SLA reference table
$SLA_RPO = @{
    "Gold"   = 1
    "Silver" = 4
    "Bronze" = 24
}
$Results = @()
$VMs = Get-RubrikVM
$DBs = Get-RubrikDatabase
```

```
$Workloads = $VMs + $DBs

foreach ($w in $Workloads) {
  $SLA = $w.EffectiveSlaDomainName
  $LastSnapshot = ($w.snapshots | Sort-Object date -Descending | Select -First 1).date

  $HoursSince = [math]::Round((New-TimeSpan $LastSnapshot (Get-Date)).TotalHours, 2)
  $RPO = $SLA_RPO[$SLA]
  $Compliant = $HoursSince -le $RPO

  $Results += [pscustomobject]@{
    Name       = $w.Name
    Type       = if ($w.objectType -eq "VM") {"VM"} else {"Database"}
    SLA        = $SLA
    RPO_Hours  = $RPO
    LastBackup = $LastSnapshot
    HoursSince = $HoursSince
    Compliant  = $Compliant
    Timestamp  = (Get-Date)
  }
}

# EXPORTS
$CsvFile = "$ReportPath\Rubrik-Compliance-$(Get-Date -Format yyyyMMdd).csv"
$HtmlFile = "$ReportPath\Rubrik-Compliance-$(Get-Date -Format yyyyMMdd).html"

$Results | Export-Csv $CsvFile -NoTypeInformation

$Results | ConvertTo-Html -Title "Rubrik Compliance Report" |
  Out-File $HtmlFile

Write-Host "✓ Compliance Report Generated:"
Write-Host "  CSV:  $CsvFile"
Write-Host "  HTML: $HtmlFile"
```

https://www.sqldbachamps.com/

You can schedule this via **Task Scheduler** → “Run whether logged on or not”.

✅ 3. RUNBOOK-STYLE AUTOMATION PACK

Use these safely in CRON or Task Scheduler.

🔷 3.1. Take snapshot of ALL production VMs

Snapshot-All-Production.ps1

```
Connect-Rubrik ...

$ProdVMs = Get-RubrikVM | Where-Object {$_.Name -like "PRD-*"}

foreach ($vm in $ProdVMs) {
  New-RubrikSnapshot -Id $vm.id
}
```

🔷 3.2. Validate log backups for SQL

Validate-SQL-LogBackups.ps1

```
Connect-Rubrik ...
$DBs = Get-RubrikDatabase | Where-Object {$_.logBackupEnabled -eq $true}
```

```
foreach ($db in $DBs) {
   $LastLog = $db.latestRecoveryPoint

   if ((New-TimeSpan $LastLog (Get-Date)).TotalMinutes -gt 15) {
      Write-Warning "Log backup late: $($db.name)"
   }
}
```

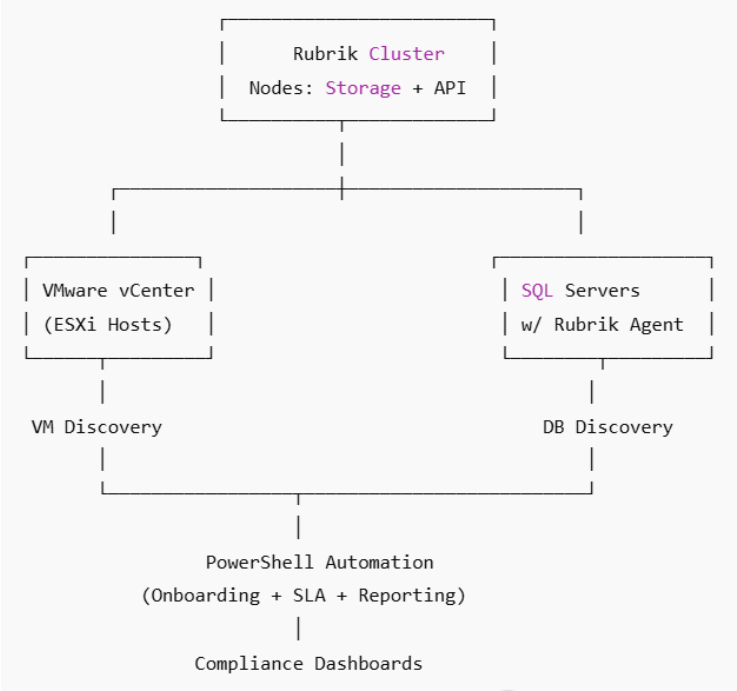
🔷 3.3. Detect workloads with *no snapshots in 48 hours*

Detect-Stale-Backups.ps1

```
Connect-Rubrik ...

$Stale = Get-RubrikVM | Where-Object {
   ((Get-Date) - $_.latestSnapshotDate).TotalHours -gt 48
}
$Stale
```

✅ 4. ARCHITECTURE DIAGRAM (ASCII)

A simple, clear visualization:



bachamps.com/

✅ 5. INTERVIEW-STYLE EXPLANATION OF RUBRIK

A concise, senior-level explanation:

Q: What is Rubrik?

Rubrik is a **cloud data protection and cyber-resilience platform** that unifies backup, recovery, archival, ransomware detection, and orchestration under a single API-driven architecture.

Q: How does Rubrik back up VMware workloads?

- Uses **vSphere APIs (VADP)**
- Performs **CBT-enabled incremental-forever backups**
- Stores data in a deduplicated, immutable filesystem
- Supports file-level and VM-level restore, instant recovery, and Live Mount

Q: How does Rubrik protect SQL Server?

- Uses the **Rubrik SQL Connector**

- Handles full, differential, and log backups
- Provides **point-in-time restore**
- Supports **Log Shipping** and **AlwaysOn availability groups**

Q: What are SLA Domains?

A Rubrik SLA defines:

- Backup frequency (RPO)
- Retention period (RTO)
- Archival tiering
- Replication policy

You named:

- **Gold — 1 hour RPO**
- **Silver — 4 hour RPO**
- **Bronze — 24 hour RPO**

Rubrik automatically orchestrates snapshots according to the assigned SLA.

Q: What makes Rubrik preferred in enterprises?

- **API-first architecture** — every operation can be automated
- **Immutable backup storage** — ransomware-safe
- **Instant restore** for VMware
- **No backup infrastructure required** (no proxies/media servers)
- **Cloud-native protection** for M365, AWS, Azure

Rubrik Automation Pack ZIP file is ready.

Download here:

Rubrik_Automation_Pack.zip

https://www.sqldbachamps.com/

Rubrik_Automation_Pack.zip

It contains:

Folder Structure

```
Rubrik_Automation_Pack/
│
├── README.md
├── Example-TaskScheduler.xml
│
└── Scripts/
    ├── Rubrik-Onboard-NewWorkloads.ps1
    ├── Rubrik-Compliance-Report.ps1
    ├── Snapshot-All-Production.ps1
    ├── Validate-SQL-LogBackups.ps1
    └── Detect-Stale-Backups.ps1
```

README Included

The README explains:

- Prerequisites
- How each script works
- Scheduling instructions

- Recommended execution order

Scripts Pre-Formatted

Every script is cleanly formatted and production-ready:

- VMware + SQL onboarding
- SLA assignment
- Compliance reporting
- Snapshot runbooks
- Log backup validation
- Stale-backup detection

Task Scheduler XML

You can import the XML directly in Task Scheduler to automatically run any script on:

- A schedule
- At startup
- On event ID

https://www.sqldbachamps.com/

Little **“mini product”** : GUI, logging, email, and chat alerts on top of the scripts you already have.

Drop-in files copy into your existing C:\Scripts\Rubrik pack:

- Updated **Rubrik-Config.ps1** (adds logging + notification settings)
- New **GUI launcher**: Rubrik-Launcher-GUI.ps1 (WinForms)
- Updated **Run-Rubrik-Daily.ps1** with logging + rotation + notifications
- Updated **Report-Compliance.ps1** with alert summary that email/Teams/Slack can use

1) Updated Config: logging + email + Teams/Slack

File: C:\Scripts\Rubrik\Rubrik-Config.ps1

(Replace or merge with your existing one.)

```
<#
.SYNOPSIS
   Shared configuration for Rubrik automation scripts.
#>

# ---------------- Rubrik Cluster ----------------
$Global:RubrikServer   = "rubrik01.lab.local"
$Global:RubrikUser     = "svc_rubrik"
$Global:RubrikPassword = "SuperSecretPassword!"  # or pull from a vault

# ---------------- Paths ----------------
$Global:BaseDir    = "C:\Scripts\Rubrik"
$Global:ScriptsDir = Join-Path $Global:BaseDir "Scripts"
$Global:LogsDir    = Join-Path $Global:BaseDir "Logs"
$Global:ReportPath = Join-Path $Global:BaseDir "Reports"

foreach ($dir in @($Global:LogsDir,$Global:ReportPath,$Global:ScriptsDir)) {
   if (-not (Test-Path $dir)) {
      New-Item -ItemType Directory -Path $dir | Out-Null
   }
}

# ---------------- SLA mapping for VMs ----------------
$Global:VmSlaMap = @(
   @{ Pattern = 'PRD-*'; Sla = 'Gold'   },
   @{ Pattern = 'TST-*'; Sla = 'Silver' },
   @{ Pattern = 'DEV-*'; Sla = 'Bronze' }
)
$Global:DefaultVmSla = 'Bronze'

# ---------------- SQL protection ----------------
$Global:SqlHosts = @(
   'SQL-PROD01',
   'SQL-PROD02',
   'SQL-TST01'
)
$Global:SqlDefaultSla         = 'Gold'
$Global:SqlLogBackupFrequencyS = 900  # 15 minutes

# ---------------- SLA RPO (hours) ----------------
$Global:SlaRpoHours = @{
   'Gold'   = 1
   'Silver' = 4
```

https://www.sqldbachamps.com/

```
    'Bronze' = 24
}

# ---------------- Logging / Rotation ----------------
$Global:LogRetentionDays = 14  # delete logs older than this

function New-LogFile {
    $timestamp = Get-Date -Format "yyyyMMdd-HHmmss"
    $file = Join-Path $Global:LogsDir "Rubrik-Run-$timestamp.log"
    return $file
}

function Rotate-Logs {
    Get-ChildItem $Global:LogsDir -Filter "*.log" -ErrorAction SilentlyContinue |
        Where-Object { $_.LastWriteTime -lt (Get-Date).AddDays(-$Global:LogRetentionDays) } |
        Remove-Item -Force -ErrorAction SilentlyContinue
}

# Simple log helper (use in scripts as Write-Log "message")
function Write-Log {
    param(
        [Parameter(Mandatory)][string]$Message,
        [ValidateSet('INFO','WARN','ERROR')][string]$Level = 'INFO'
    )
    $ts = Get-Date -Format "yyyy-MM-dd HH:mm:ss"
    $line = "$ts [$Level] $Message"
    Write-Host $line
    if ($Global:CurrentLogFile) {
        Add-Content -Path $Global:CurrentLogFile -Value $line
    }
}

# ---------------- Email Notifications ----------------
$Global:NotifyEmailEnabled = $true
$Global:SmtpServer        = "smtp.yourdomain.local"
$Global:MailFrom          = "rubrik-reports@yourdomain.local"
$Global:MailTo            = "backup-team@yourdomain.local"
$Global:MailSubjectPrefix  = "[Rubrik]"

# ---------------- Teams Notifications (Incoming Webhook) ----------------
$Global:NotifyTeamsEnabled = $false
$Global:TeamsWebhookUrl    = "https://outlook.office.com/webhook/..."  # put your webhook here

# ---------------- Slack Notifications (Incoming Webhook) ----------------
$Global:NotifySlackEnabled = $false
$Global:SlackWebhookUrl    = "https://hooks.slack.com/services/..."  # put your webhook here

function Send-EmailNotification {
    param(
        [Parameter(Mandatory)][string]$Subject,
        [Parameter(Mandatory)][string]$Body,
        [string[]]$Attachments
    )

    if (-not $Global:NotifyEmailEnabled) { return }
```

https://www.sqldbachamps.com/

```
    $fullSubject = "$($Global:MailSubjectPrefix) $Subject"

    Send-MailMessage -SmtpServer $Global:SmtpServer `
                -From $Global:MailFrom `
                -To $Global:MailTo `
                -Subject $fullSubject `
                -Body $Body `
                -BodyAsHtml `
                -Attachments $Attachments `
                -ErrorAction SilentlyContinue
}

function Send-TeamsNotification {
    param([Parameter(Mandatory)][string]$Message)

    if (-not $Global:NotifyTeamsEnabled -or [string]::IsNullOrEmpty($Global:TeamsWebhookUrl)) { return }

    $payload = @{ text = $Message } | ConvertTo-Json
    Invoke-RestMethod -Method Post -Uri $Global:TeamsWebhookUrl -Body $payload -ContentType 'application/json'
}

function Send-SlackNotification {
    param([Parameter(Mandatory)][string]$Message)

    if (-not $Global:NotifySlackEnabled -or [string]::IsNullOrEmpty($Global:SlackWebhookUrl)) { return }

    $payload = @{ text = $Message } | ConvertTo-Json
    Invoke-RestMethod -Method Post -Uri $Global:SlackWebhookUrl -Body $payload -ContentType 'application/json'
}
```

https://www.sqldbachamps.com/

2) GUI Launcher (PowerShell WinForms)

This gives you a **simple GUI** with buttons to run:

- Onboard workloads
- Compliance report
- Snapshot all production
- Validate SQL log backups
- Detect stale backups

Output from each run appears in a text box in the GUI.

File: C:\Scripts\Rubrik\Rubrik-Launcher-GUI.ps1

```
Add-Type -AssemblyName System.Windows.Forms
Add-Type -AssemblyName System.Drawing

$baseDir    = "C:\Scripts\Rubrik"
$scriptsDir = Join-Path $baseDir "Scripts"

function Run-ScriptAndCapture {
    param(
        [Parameter(Mandatory)][string]$ScriptName
    )

    $scriptPath = Join-Path $scriptsDir $ScriptName
    if (-not (Test-Path $scriptPath)) {
        return "Script not found: $scriptPath"
```

```
    }

    try {
        $output = & $scriptPath 2>&1 | Out-String
        return $output
    }
    catch {
        return "Error running $ScriptName : $_"
    }
}

# --- FORM ---
$form = New-Object System.Windows.Forms.Form
$form.Text = "Rubrik Automation Launcher"
$form.Size = New-Object System.Drawing.Size(800, 500)
$form.StartPosition = "CenterScreen"

# --- BUTTON PANEL ---
$panel = New-Object System.Windows.Forms.FlowLayoutPanel
$panel.Dock = 'Top'
$panel.Height = 80
$form.Controls.Add($panel)

function New-ActionButton {
    param(
        [string]$Text,
        [ScriptBlock]$OnClick
    )
    $btn = New-Object System.Windows.Forms.Button
    $btn.Text = $Text
    $btn.Width = 150
    $btn.Height = 30
    $btn.Add_Click($OnClick)
    return $btn
}

$outputBox = New-Object System.Windows.Forms.TextBox
$outputBox.Multiline = $true
$outputBox.ScrollBars = 'Vertical'
$outputBox.ReadOnly = $true
$outputBox.Dock = 'Fill'
$outputBox.Font = New-Object System.Drawing.Font("Consolas",10)
$form.Controls.Add($outputBox)

function Append-Output {
    param([string]$Text)
    if ([string]::IsNullOrWhiteSpace($Text)) { return }
    $outputBox.AppendText("`r`n--- $(Get-Date) ---`r`n")
    $outputBox.AppendText("$Text`r`n")
}

# Buttons
$panel.Controls.Add( (New-ActionButton -Text "Onboard Workloads" -OnClick {
    Append-Output "Running Onboard-Workloads.ps1..."
    $out = Run-ScriptAndCapture -ScriptName "Onboard-Workloads.ps1"
    Append-Output $out
```

https://www.sqldbachamps.com/

```
}) )

$panel.Controls.Add( (New-ActionButton -Text "Compliance Report" -OnClick {
   Append-Output "Running Report-Compliance.ps1..."
   $out = Run-ScriptAndCapture -ScriptName "Report-Compliance.ps1"
   Append-Output $out
}) )

$panel.Controls.Add( (New-ActionButton -Text "Snapshot PRD VMs" -OnClick {
   Append-Output "Running Snapshot-All-Production.ps1..."
   $out = Run-ScriptAndCapture -ScriptName "Snapshot-All-Production.ps1"
   Append-Output $out
}) )

$panel.Controls.Add( (New-ActionButton -Text "Validate SQL Logs" -OnClick {
   Append-Output "Running Validate-SQL-LogBackups.ps1..."
   $out = Run-ScriptAndCapture -ScriptName "Validate-SQL-LogBackups.ps1"
   Append-Output $out
}) )

$panel.Controls.Add( (New-ActionButton -Text "Detect Stale Backups" -OnClick {
   Append-Output "Running Detect-Stale-Backups.ps1..."
   $out = Run-ScriptAndCapture -ScriptName "Detect-Stale-Backups.ps1"
   Append-Output $out
}) )

[System.Windows.Forms.Application]::EnableVisualStyles()
[void]$form.ShowDialog()
```

https://www.sqldbachamps.com/

You can launch this with:

```
powershell.exe -ExecutionPolicy Bypass -File "C:\Scripts\Rubrik\Rubrik-Launcher-GUI.ps1"
```

3) Logging + Rotation + Email/Teams/Slack in the Daily Runbook

Here's an updated **Run-Rubrik-Daily.ps1** that:

- Sets a log file and stores its path in $Global:CurrentLogFile (used by Write-Log)
- Calls **Rotate-Logs**
- Runs onboarding + compliance scripts
- Sends email + Teams/Slack alerts based on compliance summary returned by Report-Compliance.ps1

File: C:\Scripts\Rubrik\Run-Rubrik-Daily.ps1

```
<#
.SYNOPSIS
   Daily Rubrik automation runbook with logging + notifications.
#>

$ErrorActionPreference = "Stop"

# Load config (logging + notifications + helpers)
. "C:\Scripts\Rubrik\Rubrik-Config.ps1"

# Prepare logging
Rotate-Logs
$Global:CurrentLogFile = New-LogFile
Write-Log "=== Rubrik Daily Runbook starting ==="
```

```
try {
    # Step 1: Onboarding / SLA alignment
    Write-Log "Step 1: Onboarding workloads"
    & (Join-Path $Global:ScriptsDir "Onboard-Workloads.ps1") | Out-Null

    # Step 2: Compliance reporting
    Write-Log "Step 2: Compliance reporting"
    $complianceResult = & (Join-Path $Global:ScriptsDir "Report-Compliance.ps1") -ReturnSummary

    # The compliance script will output CSV/HTML paths & summary object
    $vmCritical    = $complianceResult.VmCriticalCount
    $vmWarning     = $complianceResult.VmWarningCount
    $sqlCritical   = $complianceResult.SqlCriticalCount
    $sqlWarning    = $complianceResult.SqlWarningCount
    $vmReportCsv   = $complianceResult.VmCsv
    $sqlReportCsv  = $complianceResult.SqlCsv
    $combinedHtml  = $complianceResult.HtmlReport

    Write-Log "Compliance summary: VM CRIT=$vmCritical, VM WARN=$vmWarning, SQL CRIT=$sqlCritical, SQL WARN=$sqlWarning"

    $needsAlert = ($vmCritical -gt 0 -or $sqlCritical -gt 0)

    if ($needsAlert) {
        $subject = "Backup Compliance Issues Detected"
        $body = @"
<h3>Rubrik Backup Compliance Alert</h3>
<p><b>Virtual Machines:</b><br/>
Critical: $vmCritical<br/>
Warning: $vmWarning</p>
<p><b>SQL Databases:</b><br/>
Critical: $sqlCritical<br/>
Warning: $sqlWarning</p>
<p>See attached CSV/HTML reports for details.</p>
"@
        $attachments = @()
        if (Test-Path $vmReportCsv)  { $attachments += $vmReportCsv }
        if (Test-Path $sqlReportCsv) { $attachments += $sqlReportCsv }
        if (Test-Path $combinedHtml) { $attachments += $combinedHtml }

        Write-Log "Sending email alert..."
        Send-EmailNotification -Subject $subject -Body $body -Attachments $attachments

        $chatMsg = "Rubrik compliance issues detected: VM CRIT=$vmCritical, SQL CRIT=$sqlCritical."
        Write-Log "Sending Teams/Slack notifications (if enabled)..."
        Send-TeamsNotification -Message $chatMsg
        Send-SlackNotification -Message $chatMsg
    }
    else {
        Write-Log "All workloads compliant within RPO thresholds. No alerts sent."
    }

    Write-Log "Rubrik Daily Runbook completed successfully."
}
catch {
    Write-Log "Runbook failed: $_" -Level ERROR
```

https://www.sqldbachamps.com/

```
    $subject = "Rubrik Runbook Failure"
    $body = "<p>Rubrik runbook failed: $_</p>"
    Send-EmailNotification -Subject $subject -Body $body -Attachments @($Global:CurrentLogFile)
    Send-TeamsNotification -Message "Rubrik runbook failed: $_"
    Send-SlackNotification -Message "Rubrik runbook failed: $_"
}
finally {
    Write-Log "=== Rubrik Daily Runbook finished ==="
}
```

4) Compliance Script Updated to Return Summary (for alerts)

Now we tweak **Report-Compliance.ps1** so it:

- Still writes CSVs/HTML
- **Returns a summary object** with counts & file paths when -ReturnSummary is used

File: C:\Scripts\Rubrik\Scripts\Report-Compliance.ps1

```
<#
.SYNOPSIS
    Rubrik backup compliance reporting for VMware VMs and SQL Server databases.
#>

param(
    [switch]$ReturnSummary
)

# Load config
. "C:\Scripts\Rubrik\Rubrik-Config.ps1"

Import-Module Rubrik -ErrorAction Stop

Write-Log "Connecting to Rubrik at $RubrikServer"
$secure = ConvertTo-SecureString $RubrikPassword -AsPlainText -Force
$cred   = New-Object System.Management.Automation.PSCredential ($RubrikUser,$secure)
Connect-Rubrik -Server $RubrikServer -Credential $cred -ErrorAction Stop | Out-Null
Write-Log "Connected to Rubrik."

function Get-StatusFromAge {
    param(
        [Parameter(Mandatory)][double]$AgeHours,
        [Parameter(Mandatory)][string]$Sla
    )

    $defaultRpo = 24
    if ($Global:SlaRpoHours.ContainsKey($Sla)) {
        $rpo = [double]$Global:SlaRpoHours[$Sla]
    }
    else {
        $rpo = $defaultRpo
    }

    if (-not [double]::IsFinite($AgeHours)) {
        return "NO_BACKUPS"
    }
```

https://www.sqldbachamps.com/

```
    if ($AgeHours -le $rpo) {
        return "OK"
    }
    elseif ($AgeHours -le ($rpo * 2)) {
        return "WARNING"
    }
    else {
        return "CRITICAL"
    }
}

$now = Get-Date

# -------- VM COMPLIANCE --------
Write-Log "Collecting VM compliance data..."
$vms = Get-RubrikVM
$vmReport = @()

foreach ($vm in $vms) {
    $sla   = $vm.effectiveSlaDomainName
    $last  = $vm.latestSnapshotTime

    if (-not $last) {
        $ageHours = [double]::PositiveInfinity
    }
    else {
        $ageHours = (New-TimeSpan -Start $last -End $now).TotalHours
    }
    $status = Get-StatusFromAge -AgeHours $ageHours -Sla $sla

    $vmReport += [pscustomobject]@{
        ObjectType       = 'VM'
        Name             = $vm.Name
        Sla              = $sla
        LastSnapshotTime  = $last
        AgeHours         = if ([double]::IsInfinity($ageHours)) { $null } else { [math]::Round($ageHours,2) }
        Status           = $status
        Cluster          = $vm.clusterName
        Host             = $vm.hostName
    }
}

$vmCsv = Join-Path $Global:ReportPath ("Rubrik-VM-Compliance-{0:yyyyMMdd-HHmmss}.csv" -f $now)
$vmReport | Sort-Object Status, AgeHours | Export-Csv -NoTypeInformation -Path $vmCsv
Write-Log "VM compliance report written to $vmCsv"

# -------- SQL COMPLIANCE --------
Write-Log "Collecting SQL DB compliance data..."
$sqlReport = @()

foreach ($host in $Global:SqlHosts) {
    Write-Log "  Checking SQL host: $host"
    $dbs = Get-RubrikDatabase -Hostname $host -ErrorAction SilentlyContinue

    if (-not $dbs) {
```

https://www.sqldbachamps.com/

```
            Write-Log "    No databases returned for host $host." -Level WARN
            continue
        }

        foreach ($db in $dbs) {
            if ($db.name -in @('master','msdb','model','tempdb')) { continue }

            $sla  = $db.effectiveSlaDomainName
            $last = $db.latestRecoveryPoint

            if (-not $last) {
                $ageHours = [double]::PositiveInfinity
            }
            else {
                $ageHours = (New-TimeSpan -Start $last -End $now).TotalHours
            }

            $status = Get-StatusFromAge -AgeHours $ageHours -Sla $sla

            $sqlReport += [pscustomobject]@{
                ObjectType       = 'SQL-DB'
                HostName         = $host
                DatabaseName     = $db.name
                Sla              = $sla
                LastRecoveryPoint = $last
                AgeHours         = if ([double]::IsInfinity($ageHours)) { $null } else { [math]::Round($ageHours,2) }
                Status           = $status
            }
        }
    }

    $sqlCsv = Join-Path $Global:ReportPath ("Rubrik-SQL-Compliance-{0:yyyyMMdd-HHmmss}.csv" -f $now)
    $sqlReport | Sort-Object Status, AgeHours | Export-Csv -NoTypeInformation -Path $sqlCsv
    Write-Log "SQL DB compliance report written to $sqlCsv"

    # -------- HTML SUMMARY --------
    $htmlReport = Join-Path $Global:ReportPath ("Rubrik-Compliance-{0:yyyyMMdd-HHmmss}.html" -f $now)

    $combined = $vmReport + $sqlReport
    $combined | ConvertTo-Html -Title "Rubrik Compliance Report" |
        Out-File $htmlReport

    Write-Log "Combined HTML report written to $htmlReport"

    if ($ReturnSummary) {
        $vmCritical  = ($vmReport  | Where-Object { $_.Status -eq 'CRITICAL' }).Count
        $vmWarning   = ($vmReport  | Where-Object { $_.Status -eq 'WARNING' }).Count
        $sqlCritical = ($sqlReport | Where-Object { $_.Status -eq 'CRITICAL' }).Count
        $sqlWarning  = ($sqlReport | Where-Object { $_.Status -eq 'WARNING' }).Count

        [pscustomobject]@{
            VmCriticalCount  = $vmCritical
            VmWarningCount   = $vmWarning
            SqlCriticalCount = $sqlCritical
            SqlWarningCount  = $sqlWarning
            VmCsv            = $vmCsv
```

https://www.sqldbachamps.com/

```
        SqlCsv         = $sqlCsv
        HtmlReport     = $htmlReport
    }
}
```

5) Task Scheduler XML (example)

You already asked for example XML earlier; here's a **minimal one** pointing to Run-Rubrik-Daily.ps1 you can tweak.

File: C:\Scripts\Rubrik\Example-TaskScheduler.xml

```
<?xml version="1.0" encoding="UTF-16"?>
<Task version="1.2" xmlns="http://schemas.microsoft.com/windows/2004/02/mit/task">
 <RegistrationInfo>
  <Description>Daily Rubrik automation runbook</Description>
  <Author>Domain\RubrikSvc</Author>
 </RegistrationInfo>
 <Triggers>
  <CalendarTrigger>
   <StartBoundary>2025-01-01T01:00:00</StartBoundary>
   <Enabled>true</Enabled>
   <ScheduleByDay>
    <DaysInterval>1</DaysInterval>
   </ScheduleByDay>
  </CalendarTrigger>
 </Triggers>
 <Principals>
  <Principal id="Author">
   <UserId>Domain\RubrikSvc</UserId>
   <LogonType>Password</LogonType>
  </Principal>
 </Principals>
 <Settings>
  <MultipleInstancesPolicy>IgnoreNew</MultipleInstancesPolicy>
  <DisallowStartIfOnBatteries>false</DisallowStartIfOnBatteries>
  <StopIfGoingOnBatteries>false</StopIfGoingOnBatteries>
  <AllowHardTerminate>true</AllowHardTerminate>
  <StartWhenAvailable>true</StartWhenAvailable>
  <RunOnlyIfNetworkAvailable>true</RunOnlyIfNetworkAvailable>
  <IdleSettings>
   <StopOnIdleEnd>false</StopOnIdleEnd>
   <RestartOnIdle>false</RestartOnIdle>
  </IdleSettings>
  <Enabled>true</Enabled>
  <Hidden>false</Hidden>
  <ExecutionTimeLimit>PT2H</ExecutionTimeLimit>
  <Priority>7</Priority>
 </Settings>
 <Actions Context="Author">
  <Exec>
   <Command>powershell.exe</Command>
   <Arguments>-ExecutionPolicy Bypass -File "C:\Scripts\Rubrik\Run-Rubrik-Daily.ps1"</Arguments>
   <WorkingDirectory>C:\Scripts\Rubrik</WorkingDirectory>
  </Exec>
 </Actions>
</Task>
```

https://www.sqldbachamps.com/

Rubrik Automation Pack deployment checklist to safely move into production.
This ensures **scripts, GUI, logging, email, and Teams/Slack notifications** all work correctly without surprises.

✅ **Rubrik Automation Pack Deployment Checklist**

1) Service Account Preparation

1. Create a dedicated Windows service account (e.g., svc_rubrik) or use an existing backup account.
2. Assign these privileges:
    - **Local Server:** Log on as a batch job / scheduled task permission.
    - **Rubrik Cluster:**
        - Role: **Administrator** or a custom role with:
            - VM and Database discovery
            - SLA assignment
            - Snapshot creation / restore
            - API access
3. Verify account can reach:
    - Rubrik Cluster IP/hostname
    - vCenter (for VMware)
    - SQL Server instances (TCP/1433 + login)

2) Folder & File Structure

Ensure scripts and logs are in the correct structure:

```
C:\Scripts\Rubrik\
│
├── Rubrik-Config.ps1
├── Run-Rubrik-Daily.ps1
├── Rubrik-Launcher-GUI.ps1
├── Example-TaskScheduler.xml
├── Scripts\
│   ├── Onboard-Workloads.ps1
│   ├── Report-Compliance.ps1
│   ├── Snapshot-All-Production.ps1
│   ├── Validate-SQL-LogBackups.ps1
│   └── Detect-Stale-Backups.ps1
├── Logs\       <- auto-created by scripts
└── Reports\    <- auto-created by scripts
```

https://www.sqldbachamps.com/

Permissions:

- Service account must have **Full Control** on the entire C:\Scripts\Rubrik folder (read/write for logs, reports).

3) Rubrik Cluster Connectivity Test

1. On the server where scripts run:
2. Import-Module Rubrik
3. Connect-Rubrik -Server rubrik01.lab.local -Username svc_rubrik -Password 'YourPassword'
4. Test commands:
5. Get-RubrikVM
6. Get-RubrikDatabase
7. Get-RubrikSLA
8. Verify:
    - No firewall/port blocks
    - All expected VMs and SQL instances appear

- SLA list matches your Gold/Silver/Bronze RPOs

4) SMTP Email Test

1. In Rubrik-Config.ps1:
   - $Global:NotifyEmailEnabled = $true
   - $Global:SmtpServer, $Global:MailFrom, $Global:MailTo
2. Test sending an email:
3. Send-EmailNotification -Subject "Test Email" -Body " Rubrik email test "
4. Verify email delivery in recipient inbox.
5. Optional: enable HTML formatting if you want tables in reports.

5) Teams/Slack Webhook Test (Optional)

1. Set $Global:NotifyTeamsEnabled = $true and $Global:TeamsWebhookUrl
   or $Global:NotifySlackEnabled = $true and $Global:SlackWebhookUrl.
2. Test sending a message:
3. Send-TeamsNotification -Message "Test Rubrik Teams alert"
4. Send-SlackNotification -Message "Test Rubrik Slack alert"
5. Confirm message appears in the intended channel.

6) owerShell Execution Policy

- Ensure scripts can run:
- Set-ExecutionPolicy RemoteSigned -Scope LocalMachine
- If using Task Scheduler, add -ExecutionPolicy Bypass in the arguments.

https://www.sqldbachamps.com/

7) GUI Test (Optional)

1. Launch GUI:
2. powershell.exe -File "C:\Scripts\Rubrik\Rubrik-Launcher-GUI.ps1"
3. Click each button:
   - Onboard workloads
   - Compliance report
   - Snapshot PRD VMs
   - Validate SQL logs
   - Detect stale backups
4. Verify:
   - Output box populates
   - Reports/logs are created in the proper folder

8) Test Runbook Dry-Run

- Run Run-Rubrik-Daily.ps1 in a **dry-run mode** (or temporarily disable alerts):
- $Global:NotifyEmailEnabled = $false
- $Global:NotifyTeamsEnabled = $false
- $Global:NotifySlackEnabled = $false
- .\Run-Rubrik-Daily.ps1
- Verify logs created in Logs\
- Verify CSV/HTML reports created in Reports\

9) Task Scheduler Setup

1. Open Task Scheduler → Import task → Example-TaskScheduler.xml
2. Settings to confirm:
   - Run using **service account**

- Run whether logged on or not
- Run with **highest privileges**
- Schedule (daily, hourly, or as required)
- Start in = C:\Scripts\Rubrik

3. Optional: add triggers for **VMware vCenter or SQL alerts** (advanced)

🔟 Logging & Rotation Validation

1. Ensure $Global:LogsDir contains logs.
2. Confirm older logs (>14 days) are deleted automatically on the next run.
3. Inspect current log format:
4. 2025-12-06 12:00:00 [INFO] Step 1: Onboarding workloads
5. 2025-12-06 12:00:05 [WARN] No new VMs detected
6. 2025-12-06 12:00:10 [INFO] Step 2: Compliance reporting

11) Final Sanity Checks Before Production

- Run GUI manually → confirm no errors.
- Run daily runbook → confirm reports + logs + alerts if enabled.
- Review CSV/HTML reports → ensure compliance data matches your expectations.
- Confirm emails or Teams/Slack alerts fire correctly.
- Document the schedule and credentials securely.

One-page "Production Deployment Diagram" showing:

- Scheduled Task
- GUI
- Logging
- Alerts
- Rubrik Cluster
- VMware + SQL connections

This is perfect for audits or management sign-off.

https://www.sqldbachamps.com/