"คัมภีร์ป้องกันระบบล่ม" ยุทธศาสตร์ป้องกันเซิร์ฟเวอร์ล่มแบบยั่งยืน (The Unstoppable Infrastructure)

1. ปราการด่านแรก: Hardware Resilience (ความทนทาน)

คอนเซปต์: "อุปกรณ์พังได้ แต่ระบบต้องไม่หยุด"

- **RAID 10:** ทางเลือกที่ดีที่สุดสำหรับ Database/ERP เพราะได้ทั้งความเร็ว (Speed) และความทนทาน (Redundancy)
- **Hot Spare:** ต้องเสียบดิสก์สำรองทิ้งไว้ในเครื่อง 1 ก้อนเสมอ เพื่อให้ระบบทำการ Rebuild ข้อมูลทันทีที่ก้อนหลักพัง โดยไม่ต้องรอมนุษย์
- **Enterprise Grade Only:** ใช้ HDD/SSD สำหรับเซิร์ฟเวอร์เท่านั้น เพราะมีระบบ **S.M.A.R.T.** ที่แม่นยำและทนความร้อนได้สูงกว่า
- **RAID Controller Battery (BBU):** ป้องกันข้อมูลใน Cache เสียหายเมื่อไฟดับกะทันหัน

2. ปราการด่านที่สอง: Real-time Monitoring (การเฝ้าระวัง)

คอนเซปต์: "รู้ก่อนพัง แก้ก่อนล่ม"

ระดับความซับซ้อน	เครื่องมือที่แนะนำ	เหมาะสำหรับ
เริ่มต้น (Solo IT)	PowerShell + LINE Notify	ระบบเดียว, งบจำกัด, ติดตั้งไว
ระดับองค์กร (Team)	Zabbix / PRTG	มี Dashboard, เก็บกราฟย้อนหลัง, ดูหลายเครื่องพร้อมกัน
สาย Storage	NAS Built-in (Synology/QNAP)	เน้นเฝ้าระวังเรื่องดิสก์และการ Backup

สิ่งที่ต้องเฝ้าระวัง (Critical Metrics):

- **Disk Health:** แจ้งเตือนเมื่อเกิด *Reallocated Sectors* (สัญญาณก่อนดิสก์ลาโลก)
- **Service Status:** ตรวจสอบ SQL Server / ERP Service ทุก 5-10 นาที
- **Threshold Warning:** พื้นที่ดิสก์ < 10% หรือ RAM Usage > 90% ต้องแจ้งเตือนทันที

3. ปราการด่านสุดท้าย: Recovery & Process (การกู้คืน)

คอนเซปต์: "สำรองข้อมูลต้องกู้ได้จริง 100%"

- **กฎ 3-2-1:** ข้อมูล 3 ชุด, เก็บในสื่อ 2 ชนิด (เช่น Server + NAS), และ 1 ชุดต้องอยู่นอกสถานที่ (Offsite/Cloud)
- **Restore Test:** "มี Backup แต่กู้ไม่ได้ = ไม่มี Backup" ต้องซ้อมกู้คืนจริงอย่างน้อยเดือนละครั้ง
- **Health Check Cycle:** ตรวจสุขภาพระบบรายเดือน (เปรียบเสมือนการเช็คระยะรถยนต์) เพื่อดู Log และอัปเดต Patch ที่จำเป็น

สคริปต์เฝ้าระวังอัจฉริยะ (PowerShell) สคริปต์นี้จะช่วยคุณเฝ้าบ้านตลอด 24 ชม. หาก Service หยุดทำงาน มันจะพยายาม Restart ให้เอง และแจ้งเข้า LINE ทันที

```
# --- Config ---
$lineToken = "ใส่_TOKEN_ของคุณ"
$services = @("MSSQLSERVER", "YourERPServiceName")

# --- Function ---
function Send-Line ($msg) {
    $url = "https://notify-api.line.me/api/notify"
    $headers = @{ "Authorization" = "Bearer $lineToken" }
    Invoke-RestMethod -Uri $url -Method Post -Headers $headers -Body @{message=$msg}
}

# --- Logic ---
foreach ($svcName in $services) {
    $svc = Get-Service -Name $svcName -ErrorAction SilentlyContinue
    if ($svc.Status -ne "Running") {
        Send-Line "⚠ ALERT: $svcName หยุดทำงาน! กำลังพยายาม Restart..."
        Start-Service -Name $svcName
        Start-Sleep -Seconds 5
        if ((Get-Service -Name $svcName).Status -eq "Running") {
            Send-Line "☑ RECOVERED: $svcName กลับมาทำงานปกติแล้ว"
        } else {
            Send-Line "🚨 FAILED: $svcName เริ่มทำงานไม่ได้ โปรดตรวจสอบด่วน!"
        }
    }
}
```

การตรวจสอบพื้นที่ดิสก์ (Disk Space) และ **การตรวจสอบไฟล์ Backup ล่าสุด** เข้ากับสคริปต์เดิมที่คุณมี เพื่อให้เป็นสคริปต์เดียวที่ดูแลครบทุกมิติ

PowerShell Script: Total System Monitor (Services + Disk + Backup)

สคริปต์นี้จะทำ 3 อย่างในครั้งเดียว:

1. **Service Check:** ตรวจ SQL/ERP ถ้าดับจะสั่ง Restart และแจ้ง LINE
2. **Disk Check:** ถ้าไดรฟ์ไหน (C, D, E) เหลือพื้นที่น้อยกว่า 10GB จะแจ้งเตือน
3. **Backup Check:** ตรวจว่าใน Folder สำรองข้อมูล มีไฟล์ที่อัปเดตล่าสุดภายใน 24 ชม. หรือไม่ (ป้องกันกรณี Backup Fail เงียบๆ)

```
# ========================================
# การตั้งค่า (Configuration)
# ========================================
$lineToken = "ใส่_LINE_TOKEN_ของคุณที่นี่"
$servicesToMonitor = @("MSSQLSERVER", "SQLBrowser", "YourERPServiceName") # ชื่อ Service
$minDiskGB = 10                                                    # พื้นที่ขั้นต่ำ (GB)
$backupPath = "D:\Backup\ERP"                                      # พาธที่เก็บไฟล์ Backup
$maxBackupAgeHours = 24                                            # ไฟล์ Backup ต้องไม่เก่าเกินกี่ ชม.

# --- ฟังก์ชันส่ง LINE Notify ---
function Send-LineNotify($message) {
    $url = "https://notify-api.line.me/api/notify"
    $headers = @{"Authorization" = "Bearer $lineToken"}
    $body = @{message=$message}
    Invoke-RestMethod -Uri $url -Method Post -Headers $headers -Body $body
}

# ========================================
# 1. ตรวจสอบ Services (SQL & ERP)
# ========================================
foreach ($svcName in $servicesToMonitor) {
    $svc = Get-Service -Name $svcName -ErrorAction SilentlyContinue
    if ($null -eq $svc) {
```

```
        Send-LineNotify "❌ ALERT: ไม่พบ Service [$svcName] ในระบบ!"
        continue
    }
    if ($svc.Status -ne "Running") {
        Send-LineNotify "⚠️ CRITICAL: Service [$svcName] หยุดทำงาน! กำลังพยายาม Restart..."
        Start-Service -Name $svcName
        Start-Sleep -Seconds 10
        if ((Get-Service -Name $svcName).Status -eq "Running") {
            Send-LineNotify "☑️ RECOVERED: Service [$svcName] กลับมาทำงานปกติแล้ว"
        } else {
            Send-LineNotify "🚨 FAILED: ไม่สามารถเริ่ม [$svcName] ได้ โปรดตรวจสอบด่วน!"
        }
    }
}

# ==========================================
# 2. ตรวจสอบพื้นที่ดิสก์ (Disk Space)
# ==========================================
Get-PSDrive -PSProvider FileSystem | Where-Object { $_.Free -ne $null } | ForEach-Object {
    $freeGB = [math]::Round($_.Free / 1GB, 2)
    if ($freeGB -lt $minDiskGB) {
        Send-LineNotify "💾 WARNING: ไดรฟ์ [$($_.Name)] พื้นที่เหลือน้อยมาก! เหลือเพียง $freeGB GB (ควรมีอย่างน้อย $minDiskGB GB)"
    }
}
```

```
# ========================================
# 3. ตรวจสอบการสำรองข้อมูล (Backup Health)
# ========================================
if (Test-Path $backupPath) {
    $latestFile = Get-ChildItem -Path $backupPath | Sort-Object LastWriteTime -Descending | Select-Object -First 1
    if ($null -eq $latestFile) {
        Send-LineNotify "📂 ALERT: ไม่พบไฟล์ Backup ในโฟลเดอร์ $backupPath !"
    } else {
        $age = (Get-Date) - $latestFile.LastWriteTime
        if ($age.TotalHours -gt $maxBackupAgeHours) {
            $lastTime = $latestFile.LastWriteTime.ToString("yyyy-MM-dd HH:mm")
            Send-LineNotify "🚨 ALERT: Backup ไม่อัปเดต! ไฟล์ล่าสุดคือเมื่อ $lastTime (เก่าเกิน $maxBackupAgeHours ชม.)"
        }
    }
} else {
    Send-LineNotify "❌ ERROR: ไม่พบพาธ Backup [$backupPath] โปรดตรวจสอบการเชื่อมต่อ NAS/Drive"
}
```

สร้าง LINE Notify Token และนำ **PowerShell Script** ที่คุณมีไปตั้งใน **Task Scheduler** ของ Windows Server ให้ทำงานทุกๆ 1 ชั่วโมง การเขียนสคริปต์เพื่อให้เฝ้าระวัง (Monitor) เฉพาะเจาะจงที่ **SQL Server** และ **ERP Service** จะช่วยให้คุณรู้ตัวก่อนที่ User จะโทรมาแจ้งว่า "เข้าระบบไม่ได้" นี่คือสคริปต์ PowerShell แบบมืออาชีพที่ผมปรับปรุงให้มีความทนทาน (Robust) มีการดักจับข้อผิดพลาด และส่งเข้า **LINE Notify** ทันทีที่บริการหยุดทำงาน

PowerShell Script: Monitor ERP & SQL Services

สคริปต์นี้จะตรวจสอบ Service ที่คุณกำหนด ถ้าตัวไหน "Stopped" มันจะพยายาม **Restart ให้เอง 1 ครั้ง** และแจ้งเตือนเข้า LINE ครับ

PowerShell

```
# --- 1. ตั้งค่าการเชื่อมต่อ LINE ---
$lineToken = "ใส่_LINE_TOKEN_ของคุณที่นี่"

# --- 2. รายชื่อ Service ที่ต้องการเฝ้าระวัง (ใส่ชื่อ Service Name จริงๆ) ---
# ตัวอย่าง: MSSQLSERVER (SQL Standard), MSSQL$SQLEXPRESS (SQL Express), SAPServiceERP
$servicesToMonitor = @("MSSQLSERVER", "SQLBrowser", "YourERPServiceName")

# --- 3. ฟังก์ชันส่ง LINE Notify ---
function Send-LineNotify($message) {
    $url = "https://notify-api.line.me/api/notify"
    $headers = @{
        "Authorization" = "Bearer $lineToken"
        "Content-Type"  = "application/x-www-form-urlencoded"
    }
    $body = "message=$message"
    Invoke-RestMethod -Uri $url -Method Post -Headers $headers -Body $body
}

# --- 4. เริ่มต้นการตรวจสอบ ---
foreach ($svcName in $servicesToMonitor) {
    $service = Get-Service -Name $svcName -ErrorAction SilentlyContinue

    if ($null -eq $service) {
        Send-LineNotify "❌ ALERT: ไม่พบ Service [$svcName] ในระบบ! โปรดตรวจสอบ"
        continue
    }
```

```
    if ($service.Status -ne "Running") {
        $timestamp = Get-Date -Format "yyyy-MM-dd HH:mm:ss"
        Send-LineNotify " ⚠ CRITICAL ($timestamp): Service [$svcName] หยุดทำงาน! กำลังพยายาม
Restart..."

        # พยายาม Restart Service
        Start-Service -Name $svcName
        Start-Sleep -Seconds 5 # รอ 5 วินาทีเพื่อให้ระบบเริ่มทำงาน

        $checkAgain = Get-Service -Name $svcName
        if ($checkAgain.Status -eq "Running") {
            Send-LineNotify " ☑ RECOVERED: Service [$svcName] กลับมาทำงานปกติแล้ว"
        } else {
            Send-LineNotify " 🚨 FAILED: ไม่สามารถเริ่มการทำงาน [$svcName] ได้ด้วยสคริปต์ โปรดเข้าตรวจสอบ
ด่วน!"
        }
    }
}
```

วิธีนำไปใช้งาน (Step-by-Step)

1. **หาชื่อ Service Name:** * กด Win + R พิมพ์ services.msc
   - คลิกขวาที่ Service (เช่น SQL Server) เลือก **Properties**
   - ก๊อปปี้ชื่อตรง **Service name:** มาใส่ในสคริปต์ (ตัวแปร $servicesToMonitor)
2. **บันทึกไฟล์:** เซฟเป็นชื่อ MonitorERP.ps1 ไว้ในไดรฟ์ C: หรือที่ปลอดภัย
3. **ตั้งค่า Task Scheduler:**
   - เปิด **Task Scheduler** > Create Task
   - **Triggers:** ตั้งให้ทำงานทุกๆ 5 หรือ 10 นาที (Repeat task every 5 minutes)
   - **Actions:** * Program/script: powershell.exe
     - Add arguments: -ExecutionPolicy Bypass -File "C:\Path\To\Your\MonitorERP.ps1"

- **Conditions:** เลือก "Run whether user is logged on or not" และ "Run with highest privileges" (เพื่อให้มีสิทธิ์ Restart Service)

ข้อแนะนำเพิ่มเติม

- **ตรวจเช็ค Error Log:** หาก SQL Server ล่มบ่อย สคริปต์จะช่วย Restart ให้ แต่คุณควรเช็ค Windows Event Viewer ควบคู่ไปด้วยว่ามีปัญหาเรื่อง RAM เต็มหรือ Disk Error หรือไม่
- **การ Monitor Disk:** นอกจากการเช็ค Service แล้ว อย่าลืมสคริปต์ตรวจเช็ค **Disk Free Space** (ที่คุณสรุปไว้ก่อนหน้า) เพราะถ้า Disk เต็ม SQL Server จะดับทันทีและ Restart ไม่ขึ้นครับ

วิธีติดตั้งและใช้งานแบบมืออาชีพ

1. **การตั้งสิทธิ์ (Permissions):**
   - เนื่องจากสคริปต์นี้มีการสั่ง Start-Service คุณต้องตั้งค่าใน **Task Scheduler** ให้รันด้วยสิทธิ์ **"Run with highest privileges"**
   - ในแท็บ General ของ Task ให้เลือก User เป็น SYSTEM หรือ Administrator เพื่อให้สคริปต์ทำงานได้แม้ไม่มีคน Login ค้างไว้
2. **ความถี่ในการทำงาน (Triggers):**
   - แนะนำให้ตั้งรันทุกๆ **15 นาที** หรือ **30 นาที** (Repeat task every: 15 minutes)
   - การตั้งถี่เกินไป (เช่น ทุก 1 นาที) อาจทำให้ LINE แจ้งเตือนรบกวนมากเกินไปในช่วงที่กำลังซ่อมระบบ
3. **การทดสอบ (Testing):**
   - ลองเปลี่ยนค่า $minDiskGB ให้สูงกว่าพื้นที่ปัจจุบันของคุณ แล้วสั่งรันสคริปต์ดูว่า LINE เด้งหรือไม่
   - ลองหยุด Service ตัวที่ไม่สำคัญ (เช่น Print Spooler - ถ้าใส่เพิ่มในลิสต์) เพื่อดูว่าสคริปต์สั่ง Start เองได้จริงไหม

การมีระบบ **Monitor + Auto Restart** จะช่วยให้ระบบ ERP ของคุณมี Uptime เกือบ 100% เพราะปัญหา 80% ของ SQL Server มักเกิดจาก Resource เต็มชั่วขณะหรือ Service ค้าง ซึ่งการ Restart มักจะแก้ได้ทันที