Script Migration SQL édition Standard vers Web Partie 2 :

```
# Configuration
$instanceName = "SQLSPI2017"
$productKey = ""
$backupPath = "D:\Usr\Data.Net\BaseSqlBackup"
$restorePath = "D:\Usr\Data.Net\BaseSqlBackup\Log"  # Dossier de restauration des fichiers .mdf et .ldf
$sqlSetupPath = "\\sr9v-ftpspi\c$\Data\FtpSepteoPI\FtpPrive\SPIsuite\SQLServer2017Standard\SQL2017_Standard\setup.exe"
$saPassword = "**********"
$tempSaPassword = "**********"  # Mot de passe temporaire complexe pour l'installation

# Création du dossier de restauration si nécessaire
if (-not (Test-Path $restorePath)) {
    Write-Host "Création du dossier de restauration: $restorePath" -ForegroundColor Yellow
    New-Item -Path $restorePath -ItemType Directory -Force | Out-Null
    Write-Host "  -> Dossier créé" -ForegroundColor Green
}

Write-Host "=== INSTALLATION SQL SERVER 2017 SANS MODULE SQLPS ===" -ForegroundColor Cyan

# Installation SQL Server 2017 Web Edition
Write-Host "`n=== Étape 5: Installation SQL Server 2017 Web Edition ===" -ForegroundColor Cyan
Write-Host "Installation en cours... (Cela peut prendre 15-30 minutes)" -ForegroundColor Yellow
Write-Host "Utilisation d'un mot de passe SA temporaire complexe pour l'installation" -ForegroundColor Yellow

$setupArgs = @(
    "/ACTION=Install",
    "/QUIET",
    "/IACCEPTSQLSERVERLICENSETERMS",
    "/FEATURES=SQLENGINE",
    "/INSTANCENAME=$instanceName",
    "/SQLSVCACCOUNT=`"NT AUTHORITY\NETWORK SERVICE`"",
    "/SQLSYSADMINACCOUNTS=`"BUILTIN\Administrators`"",
    "/SECURITYMODE=SQL",
    "/SAPWD=`"$tempSaPassword`"",
    "/TCPENABLED=1",
    "/PID=$productKey"
)

if (Test-Path $sqlSetupPath) {
```

```
    Write-Host "Chemin setup validé: $sqlSetupPath" -ForegroundColor Gray
    Write-Host "Lancement de l'installation..." -ForegroundColor Yellow

    $setupProcess = Start-Process -FilePath $sqlSetupPath -ArgumentList $setupArgs -Wait
-PassThru -NoNewWindow

    if ($setupProcess.ExitCode -eq 0) {
        Write-Host "  -> Installation terminée avec succès" -ForegroundColor Green
    } elseif ($setupProcess.ExitCode -eq 3010) {
        Write-Host "  -> Installation terminée avec succès (redémarrage requis)"
-ForegroundColor Green
    } else {
        Write-Host "  -> AVERTISSEMENT: Code de sortie $($setupProcess.ExitCode)"
-ForegroundColor Yellow
        Write-Host "     Vérifiez les logs dans C:\Program Files\Microsoft SQL Server\140\Setup
Bootstrap\Log\" -ForegroundColor Gray
    }
} else {
    Write-Host "ERREUR: Fichier setup.exe introuvable à: $sqlSetupPath" -ForegroundColor
Red
    Write-Host "Veuillez mettre à jour la variable `$sqlSetupPath dans le script"
-ForegroundColor Red
    exit 1
}

# Attente du démarrage du service
Write-Host "`nAttente du démarrage du service SQL..." -ForegroundColor Yellow
Start-Sleep -Seconds 30

$sqlService = Get-Service -Name "MSSQL`$$instanceName" -ErrorAction SilentlyContinue
if ($sqlService -and $sqlService.Status -eq 'Running') {
    Write-Host "  -> Service SQL démarré" -ForegroundColor Green
} else {
    Write-Host "AVERTISSEMENT: Le service SQL n'est pas démarré automatiquement"
-ForegroundColor Red
    Write-Host "Tentative de démarrage manuel..." -ForegroundColor Yellow

    try {
        Start-Service -Name "MSSQL`$$instanceName" -ErrorAction Stop
        Start-Sleep -Seconds 10
        Write-Host "  -> Service démarré manuellement" -ForegroundColor Green
    } catch {
        Write-Host "ERREUR: Impossible de démarrer le service" -ForegroundColor Red
        Write-Host "Démarrez-le manuellement avant de continuer" -ForegroundColor Yellow
        Read-Host "Appuyez sur Entrée une fois le service démarré"
    }
}
```

```
# Fonction pour exécuter des commandes SQL via sqlcmd
function Invoke-SqlCommand {
    param(
        [string]$ServerInstance,
        [string]$Query,
        [string]$Username = "sa",
        [string]$Password,
        [string]$Database = "master"
    )

    $sqlcmdPath = "sqlcmd.exe"

    # Construction des arguments
    $args = @(
        "-S", $ServerInstance,
        "-U", $Username,
        "-P", $Password,
        "-d", $Database,
        "-Q", $Query,
        "-b"  # Stop batch job if there is an error
    )

    try {
        $output = & $sqlcmdPath $args 2>&1
        if ($LASTEXITCODE -ne 0) {
            throw "Erreur SQL: $output"
        }
        return $output
    } catch {
        throw $_
    }
}

# Changement du mot de passe SA
Write-Host "`n=== Étape 5b: Changement du mot de passe SA ===" -ForegroundColor Cyan
try {
    $serverInstance = ".\$instanceName"
    Write-Host "Connexion avec le mot de passe temporaire..." -ForegroundColor Yellow

    # Test de connexion
    $testQuery = "SELECT @@VERSION"
    Invoke-SqlCommand -ServerInstance $serverInstance -Query $testQuery -Password
$tempSaPassword | Out-Null
    Write-Host "  -> Connexion réussie" -ForegroundColor Green

    Write-Host "Changement du mot de passe SA vers le mot de passe souhaité..."
-ForegroundColor Yellow
```

```
    # Changer le mot de passe
    $changePasswordQuery = @"
ALTER LOGIN [sa] WITH CHECK_POLICY = OFF;
ALTER LOGIN [sa] WITH PASSWORD = '$saPassword';
"@

    Invoke-SqlCommand -ServerInstance $serverInstance -Query $changePasswordQuery
-Password $tempSaPassword
    Write-Host "  -> Mot de passe SA changé avec succès" -ForegroundColor Green

} catch {
    Write-Host "  -> ERREUR lors du changement de mot de passe:
$($_.Exception.Message)" -ForegroundColor Red
    Write-Host "  -> Vous devrez changer le mot de passe SA manuellement après la
migration" -ForegroundColor Yellow
    Write-Host "  -> Le mot de passe temporaire est: $tempSaPassword" -ForegroundColor
Yellow
}

# Restauration des bases de données
Write-Host "`n=== Étape 6: Restauration des bases de données ===" -ForegroundColor
Cyan
try {
    $serverInstance = ".\$instanceName"
    Write-Host "Connexion à la nouvelle instance: $serverInstance" -ForegroundColor Yellow

    # Test de connexion et récupération des informations
    $versionQuery = "SELECT @@VERSION AS Version, SERVERPROPERTY('Edition') AS
Edition"
    $versionInfo = Invoke-SqlCommand -ServerInstance $serverInstance -Query
$versionQuery -Password $saPassword
    Write-Host "  -> Connexion réussie" -ForegroundColor Green
    Write-Host "  -> Informations serveur:" -ForegroundColor Gray
    Write-Host $versionInfo -ForegroundColor Gray

    # Utilisation du chemin de restauration configuré
    Write-Host "  -> Chemin de restauration: $restorePath" -ForegroundColor Gray

    # Vérification que le dossier existe
    if (-not (Test-Path $restorePath)) {
        Write-Host "Création du dossier de restauration..." -ForegroundColor Yellow
        New-Item -Path $restorePath -ItemType Directory -Force | Out-Null
        Write-Host "  -> Dossier créé" -ForegroundColor Green
    }

    $backupFiles = Get-ChildItem -Path $backupPath -Filter "*.bak" -ErrorAction Stop

    if ($backupFiles.Count -eq 0) {
```

```
        Write-Host "Aucun fichier de sauvegarde trouvé dans: $backupPath" -ForegroundColor
Yellow
    } else {
        Write-Host "`nFichiers de sauvegarde trouvés: $($backupFiles.Count)"
-ForegroundColor White

        $restoreSuccess = 0
        $restoreError = 0

        foreach ($backupFile in $backupFiles) {
            $dbName = $backupFile.BaseName
            Write-Host "`n[Restauration: $dbName]" -ForegroundColor Yellow
            Write-Host "  Fichier: $($backupFile.Name)" -ForegroundColor Gray
            Write-Host "  Taille: $([math]::Round($backupFile.Length / 1MB, 2)) MB"
-ForegroundColor Gray

            try {
                # Récupération des fichiers logiques du backup
                Write-Host "  -> Lecture des informations du backup..." -ForegroundColor Cyan
                $fileListQuery = "RESTORE FILELISTONLY FROM DISK =
N'$($backupFile.FullName)'"
                $fileListResult = Invoke-SqlCommand -ServerInstance $serverInstance -Query
$fileListQuery -Password $saPassword

                # Construction de la commande RESTORE avec MOVE
                $restoreQuery = "RESTORE DATABASE [$dbName] FROM DISK =
N'$($backupFile.FullName)' WITH REPLACE, RECOVERY"

                # Ajout des clauses MOVE vers le dossier de restauration configuré
                $dataFile = "$restorePath\$dbName.mdf"
                $logFile = "$restorePath\${dbName}_log.ldf"

                # Note: Cette approche simplifiée suppose que le backup contient des fichiers
standards
                # Pour une solution plus robuste, il faudrait parser le résultat de FILELISTONLY
                $restoreQuery += ", MOVE N'$dbName' TO N'$dataFile'"
                $restoreQuery += ", MOVE N'${dbName}_log' TO N'$logFile'"

                Write-Host "  -> Restauration en cours..." -ForegroundColor Cyan

                # Exécution de la restauration
                Invoke-SqlCommand -ServerInstance $serverInstance -Query $restoreQuery
-Password $saPassword | Out-Null

                Write-Host "  -> SUCCÈS: Base restaurée" -ForegroundColor Green
                $restoreSuccess++

            } catch {
```

```
            Write-Host "  -> ERREUR lors de la restauration:" -ForegroundColor Red
            Write-Host "     $($_.Exception.Message)" -ForegroundColor Red

            # Tentative de restauration sans MOVE (laisse SQL gérer les chemins)
            Write-Host "  -> Tentative de restauration automatique..." -ForegroundColor Yellow
            try {
                $simpleRestoreQuery = "RESTORE DATABASE [$dbName] FROM DISK =
N'$($backupFile.FullName)' WITH REPLACE, RECOVERY"
                Invoke-SqlCommand -ServerInstance $serverInstance -Query
$simpleRestoreQuery -Password $saPassword | Out-Null
                Write-Host "  -> SUCCÈS: Base restaurée (mode automatique)"
-ForegroundColor Green
                $restoreSuccess++
            } catch {
                Write-Host "  -> ÉCHEC définitif pour $dbName" -ForegroundColor Red
                Write-Host "     $($_.Exception.Message)" -ForegroundColor Red
                $restoreError++
            }
        }
    }

    # Résumé de la restauration
    Write-Host "`n=== Résumé de la restauration ===" -ForegroundColor Cyan
    Write-Host "Réussies: $restoreSuccess" -ForegroundColor Green
    Write-Host "Échouées: $restoreError" -ForegroundColor $(if ($restoreError -gt 0) {
"Red" } else { "Gray" })
  }

} catch {
  Write-Host "`nERREUR lors de la restauration: $_" -ForegroundColor Red
  Write-Host "Détails: $($_.Exception.Message)" -ForegroundColor Red
  exit 1
}

Start-Service -Name "LoginSpace - Service Dovadis Base" -ErrorAction Stop
$ServiceBase = Get-Service -Name "LoginSpace - Service Dovadis Base" -ErrorAction
SilentlyContinue
if ($ServiceBase -and $ServiceBase.Status -eq 'Running') {
  Write-Host "  -> Service Base démarré" -ForegroundColor Green
} else {
  Write-Host "AVERTISSEMENT: Le service Base n'est pas démarré automatiquement"
-ForegroundColor Red
}
# Résumé final
Write-Host "`n=====================================" -ForegroundColor Green
Write-Host "=== MIGRATION TERMINÉE ===" -ForegroundColor Green
Write-Host "=====================================" -ForegroundColor Green
Write-Host "`nInformations:" -ForegroundColor Cyan
```

```
Write-Host "  Instance: $instanceName" -ForegroundColor White
Write-Host "  Version: SQL Server 2017 Web Edition" -ForegroundColor White
Write-Host "  Sauvegardes: $backupPath" -ForegroundColor White
Write-Host "  Fichiers restaurés: $restorePath" -ForegroundColor White
Write-Host "  Mot de passe SA: $saPassword" -ForegroundColor White
Write-Host "`nÉtapes suivantes recommandées:" -ForegroundColor Yellow
Write-Host "  1. Vérifier la connectivité des applications" -ForegroundColor White
Write-Host "  2. Tester les bases de données restaurées" -ForegroundColor White
Write-Host "  3. Vérifier les logins et permissions" -ForegroundColor White
Write-Host "  4. Configurer les jobs SQL Agent si nécessaire" -ForegroundColor White
Write-Host "  5. Mettre à jour les configurations de backup régulier" -ForegroundColor White
Write-Host "  6. Conserver les sauvegardes dans $backupPath" -ForegroundColor White
Write-Host "`nMigration terminée avec succès!" -ForegroundColor Green
```