Instituto de Educación Superior Privado

KHIPU

Sesión

31

Gestión de Copias de Seguridad de Bases de Datos

Desarrollo de Sistemas de Información

Desarrollo de Bases de Datos

Docente: Rildo M. Tapia Pacheco

Introducción

La gestión de copias de seguridad es un pilar fundamental de la seguridad de la información y la continuidad del negocio. Su propósito principal es permitir la restauración de datos originales en caso de pérdida, daño o corrupción, ya sea por fallos de hardware, errores humanos, ciberataques o desastres naturales.

Conceptos Clave

- **Backup (Copia de Seguridad/Respaldo):** El proceso de duplicar datos de la base de datos y almacenarlos en un medio secundario (disco duro, cinta, nube) para su posterior recuperación.
- **Restauración (Restore/Recovery):** El proceso inverso, mediante el cual se recuperan los datos a partir de una copia de seguridad almacenada para volver al estado operativo.
- **RPO (Recovery Point Objective - Objetivo de Punto de Recuperación):** Define la cantidad máxima aceptable de pérdida de datos medida en tiempo. Por ejemplo, un RPO de 1 hora significa que no se puede permitir la pérdida de más de una hora de datos; esto determina la frecuencia con la que se deben realizar las copias de seguridad.
- **RTO (Recovery Time Objective - Objetivo de Tiempo de Recuperación):** Define el tiempo máximo aceptable para restaurar los sistemas y volver a la operación normal después de un incidente. Un RTO bajo implica la necesidad de mecanismos de recuperación más rápidos y eficientes.
- **Tipos de Backups:**
    - **Completo (Full):** Copia todos los datos de la base de datos. Es la base de cualquier estrategia.
    - **Diferencial (Differential):** Copia solo los datos que han cambiado desde el último backup completo. La restauración requiere el último backup completo y el último diferencial.
    - **Incremental (Incremental):** Copia solo los datos que han cambiado desde el último backup de cualquier tipo (completo, diferencial o incremental). La restauración requiere toda la cadena de backups desde el último completo.
    - **De Registro de Transacciones (Transaction Log):** En bases de datos que soportan un registro transaccional (como SQL Server u Oracle), este backup captura todas las transacciones realizadas desde el último backup del registro, permitiendo una recuperación granular a un punto específico en el tiempo (Point-in-Time Recovery).
- **Estrategia 3-2-1:** Una práctica recomendada que sugiere tener al menos **3** copias de los datos, almacenadas en **2** tipos de medios diferentes, y **1** copia fuera de sitio (off-site o en la nube).

Parte 2: Procedimientos de Planificación, Diseño, Ejecución y Control

La gestión eficaz de las copias de seguridad sigue un ciclo de vida estructurado:

1. Planificación (Planning)

Esta fase implica definir los requisitos y la estrategia general.

- **Evaluación de Necesidades:** Determinar qué datos deben respaldarse, su criticidad y el impacto de su pérdida.
- **Definición de RPO y RTO:** Establecer, junto a la dirección y los interesados del negocio, los objetivos de tiempo y pérdida de datos aceptables. Esto guía el diseño de la frecuencia y los métodos de backup.
- **Política de Retención:** Decidir por cuánto tiempo se deben conservar las copias de seguridad (por ejemplo, diariamente durante 1 semana, semanalmente durante 1 mes, mensualmente durante 1 año, etc.), considerando requisitos legales y de cumplimiento.
- **Asignación de Recursos:** Estimar los requisitos de almacenamiento, ancho de banda, software y personal.

2. Diseño (Design)

En esta fase se traduce el plan en una solución técnica específica.

- **Selección de Tipos de Backup:** Elegir la combinación adecuada de backups completos, diferenciales e incrementales o de registros, basándose en el RPO/RTO y el volumen de datos.
- **Selección de Medios y Ubicaciones de Almacenamiento:** Decidir dónde se almacenarán los backups (servidores locales, NAS, SAN, nube pública como AWS o Azure) y garantizar la implementación de la regla 3-2-1.
- **Diseño del Software/Herramientas:** Seleccionar las herramientas de software de backup (nativas del sistema de gestión de bases de datos, de terceros) y configurar sus ajustes, incluyendo compresión y, si es necesario, cifrado.
- **Documentación de Procedimientos:** Crear manuales claros y documentados para los procedimientos de backup y, crucialmente, de restauración.

3. Ejecución (Execution)

Esta es la fase operativa donde se implementan y realizan las copias de seguridad.

- **Implementación de Scripts/Tareas Programadas:** Configurar las tareas automatizadas (jobs) para ejecutar los backups según la frecuencia planificada, usualmente en horarios de bajo impacto para los usuarios.

- **Monitorización Activa:** Supervisar que las tareas de backup se ejecuten correctamente. Es vital verificar los registros y alertas para detectar fallos inmediatamente.
- **Almacenamiento Seguro:** Mover las copias de seguridad a las ubicaciones designadas (off-site, cinta, etc.) de forma segura, garantizando la integridad de los datos.

4. Control y Verificación (Control and Verification)

La fase más crítica, a menudo subestimada: verificar que los backups funcionen cuando se necesiten.

- **Pruebas de Restauración (Backup Certification):** Realizar restauraciones de prueba periódicas en un entorno de *staging* o pruebas, NUNCA en producción. Esto valida que los datos se puedan recuperar con éxito y que el RTO planificado sea alcanzable.
- **Auditoría y Cumplimiento:** Revisar periódicamente las políticas y procedimientos para asegurar que cumplen con las normativas internas y externas (HIPAA, GDPR, SOX, etc.).
- **Mantenimiento y Revisión:** Actualizar los planes de backup a medida que la base de datos crece, cambian los requisitos del negocio o la infraestructura tecnológica evoluciona.

Backups en MariaDB

MariaDB (un fork popular de MySQL) ofrece varias herramientas y enfoques para implementar las estrategias de backup que hemos discutido.

Consolidemos la teoría con la práctica, enfocándonos en las **estrategias de backup para MariaDB/MySQL**.

Estrategias de Backup para MariaDB

MariaDB soporta dos tipos principales de métodos de backup, que se adaptan a diferentes necesidades de RPO/RTO:

1. Backups Lógicos (Logical Backups)

Extraen los datos en un formato de texto plano (sentencias SQL INSERT), que es universalmente compatible y fácil de restaurar en casi cualquier versión o plataforma de base de datos.

- **Herramienta Principal:** mysqldump o mariadb-dump.
- **Pros:**
    - Fáciles de usar y entender.
    - Portátiles (archivos .sql comprimidos).
    - Permiten la restauración granular (elegir qué tablas restaurar).

- **Contras:**
    - **Lentos** para bases de datos muy grandes (cientos de GB o TB) porque procesan fila por fila y luego deben reconstruir los índices durante la restauración.
    - Generan un bloqueo (lock) en las tablas durante parte del proceso (aunque se puede minimizar con opciones como --single-transaction para motores InnoDB).

Estrategia con Backups Lógicos:

Se suelen usar para backups diarios o semanales de bases de datos de tamaño moderado, o para backups de configuración/esquemas.

bash

 # Ejemplo de uso básico de mysqldump para un backup completo:

mysqldump -u [usuario] -p[contraseña] --all-databases --single-transaction | gzip > /ruta/a/backup_completo_$(date +\%F).sql.gz

Usa el código con precaución.

2. Backups Físicos (Physical Backups)

Copian directamente los archivos de datos subyacentes del sistema de archivos (los archivos .frm, .ibd, etc.).

- **Herramientas Principales:** mariabackup (recomendado para MariaDB), xtrabackup (de Percona, compatible con MariaDB/MySQL).
- **Pros:**
    - **Muy rápidos** para hacer el backup y **extremadamente rápidos** para la restauración, ya que solo es copiar archivos de vuelta al directorio de datos.
    - Permiten backups incrementales eficientes.
    - Permiten la recuperación a un punto en el tiempo (Point-in-Time Recovery - PITR) cuando se combinan con los binlog.
- **Contras:**
    - Menos portátiles (solo funcionan en la misma versión y arquitectura de MariaDB/MySQL).
    - Más complejos de configurar y gestionar que mysqldump.

Estrategia con Backups Físicos:

Son ideales para backups de bases de datos grandes y críticas donde el RTO es muy bajo (se necesita recuperar la base de datos rápidamente). Se usan para backups completos semanales/diarios y backups incrementales frecuentes.

 # Ejemplo conceptual de uso de mariabackup (requiere configuración y permisos específicos):

mariabackup --backup --target-dir=/data/backups/full_backup/

 # Luego se prepara el backup para ser restaurado

mariabackup --prepare --target-dir=/data/backups/full_backup/

Usa el código con precaución.

3. La Estrategia Combinada y los Binlogs (Registros Binarios)

La estrategia más robusta combina ambos métodos y utiliza los **Binary Logs (Binlogs)** de MariaDB. Los binlogs registran cada cambio de datos ejecutado en la base de datos en tiempo real.

- **El RPO Cero o Casi Cero:** Si un backup físico (que puede tener unas horas de antigüedad) se daña, puedes restaurar el backup físico y luego "reproducir" los binlogs desde el momento en que se hizo el backup hasta el instante exacto antes del fallo. Esto logra una recuperación a un punto en el tiempo (PITR) y un RPO mínimo.

Resumen de un Plan de Backup Completo para MariaDB

Aplicando las fases que vimos anteriormente, un plan podría ser:

Fase	Actividad	Herramientas/Métodos en MariaDB
Planificación	Definir RPO/RTO (e.g., RPO 15 min, RTO 1 hr)	Decisión de usar mariabackup + Binlogs
Diseño	Estrategia 3-2-1. Backups completos semanales, incrementales diarios, Binlogs continuos.	Servidor local, NAS, Nube (S3). Cifrado.
Ejecución	Tareas programadas (CRON jobs)	mariabackup scripts, rotación de logs
Control	Pruebas de restauración mensuales	Entorno de *staging* para validar la recuperación

ejemplos de scripts para MariaDB y, posteriormente, en las pruebas de restauración.

Ejemplos de Scripts

Parte 1: Ejemplos de Scripts para Backups de MariaDB

Aquí tienes ejemplos prácticos para automatizar los backups usando mariadb-dump (lógico) y mariabackup (físico).

A. Script para Backup Lógico Diario (usando mariadb-dump / mysqldump)

Este script es ideal para bases de datos pequeñas/medianas y se puede programar fácilmente con cron. Utiliza gzip para comprimir el resultado.

Nombre del archivo: backup_logico_diario.sh

bash

```
#!/bin/bash

# --- Configuración ---
DB_USER="backup_user"
DB_PASS="TuSuperContraseñaSegura"
BACKUP_DIR="/var/backups/mariadb/logical"
DATE_FORMAT=$(date +"%Y%m%d_%H%M%S")
BACKUP_FILE="$BACKUP_DIR/full_backup_$DATE_FORMAT.sql.gz"
# --- Fin Configuración ---

mkdir -p "$BACKUP_DIR"

echo "Iniciando backup lógico completo de MariaDB..."

# El comando principal:
# --single-transaction asegura la consistencia para InnoDB sin bloquear la BD
mariadb-dump -u "$DB_USER" -p"$DB_PASS" --all-databases --single-transaction | gzip > "$BACKUP_FILE"
```

```
if [ $? -eq 0 ]; then
    echo "Backup completado exitosamente: $BACKUP_FILE"
    # Opcional: Eliminar backups más antiguos de 7 días
    find "$BACKUP_DIR" -type f -name '*.sql.gz' -mtime +7 -exec rm {} \;
    echo "Archivos antiguos eliminados."
else
    echo "¡ERROR al realizar el backup!"
fi
```

Usa el código con precaución.

B. Script para Backup Físico Completo (usando mariabackup)

mariabackup es más complejo de configurar, requiere permisos específicos y acceso a los archivos del sistema, pero es mucho más rápido.

Nombre del archivo: backup_fisico_completo.sh

bash

```
#!/bin/bash

# --- Configuración ---
BACKUP_DIR="/var/backups/mariadb/physical"
DATE_FORMAT=$(date +"%Y%m%d_%H%M%S")
TARGET_DIR="$BACKUP_DIR/full_$DATE_FORMAT"
# Usuario de MariaDB con privilegios RELOAD, PROCESS, LOCK TABLES, entre otros
DB_USER="backup_user_physical"
DB_PASS="OtraContraseñaSegura"
# --- Fin Configuración ---
```

```
mkdir -p "$TARGET_DIR"

echo "Iniciando backup físico completo con mariabackup en $TARGET_DIR..."

# El comando principal de backup
mariabackup --backup --target-dir="$TARGET_DIR" --user="$DB_USER" --password="$DB_PASS"

if [ $? -eq 0 ]; then
    echo "Backup físico completado exitosamente."

    # MUY IMPORTANTE: Se debe "preparar" el backup antes de que pueda ser restaurado
    echo "Preparando el backup para la restauración (commit de transacciones pendientes)..."
    mariabackup --prepare --target-dir="$TARGET_DIR"

    if [ $? -eq 0 ]; then
        echo "Backup preparado y listo para restaurar."
    else
        echo "¡ERROR al preparar el backup!"
    fi
else
    echo "¡ERROR al realizar el backup físico!"
fi
```

Usa el código con precaución.

Parte 2: La Fase Crítica de las Pruebas de Restauración

Como mencionamos en la teoría, un backup que no ha sido probado es inútil. La fase de control y verificación es donde validas tu RTO y RPO.

El Procedimiento de Prueba de Restauración

El objetivo es simular un desastre sin afectar el entorno de producción.

1. **Entorno Aislado (Staging/QA):** Nunca restaures sobre tu base de datos de producción. Debes tener un servidor de pruebas o una máquina virtual idéntica (en lo posible) a la de producción.
2. **Automatización de Pruebas:** Idealmente, este proceso también debería ser automatizado. Un script de prueba semanal podría:

1. Copiar el último archivo de backup al entorno de pruebas.
2. Detener el servicio de MariaDB en pruebas.
3. Limpiar el directorio de datos existente en pruebas.
4. Restaurar el backup.
5. Iniciar el servicio de MariaDB.
6. Ejecutar un script de validación (por ejemplo, contar el número de filas en tablas clave o ejecutar consultas de prueba).

3. **Validación de Datos:** No basta con que la base de datos inicie; debes verificar la integridad de los datos.

1. ¿Faltan registros?
2. ¿Son los datos consistentes?
3. ¿Puedes acceder a las tablas críticas?

4. **Medición de Tiempos:** Mide cuánto tiempo tarda todo el proceso (copia, restauración, inicio, validación). Esto te dirá si tu **RTO** (Recovery Time Objective) es realista o si necesitas una estrategia más rápida (como cambiar de backups lógicos a físicos).
5. **Reporte y Alerta:** El resultado de la prueba debe generar un reporte automático. Si la prueba falla, debe enviar una alerta inmediata al equipo de operaciones.

Un backup no está verificado hasta que ha sido restaurado con éxito en un entorno controlado.

Ajustaremos los scripts para que puedas ejecutarlos en

Windows, utilizando archivos .bat o powershell, que son más comunes en ese entorno, aunque las herramientas subyacentes de MariaDB (mariadb-dump, mariabackup) siguen siendo las mismas en Windows que en Linux.

Parte 1: Ejemplos de Scripts para Backups de MariaDB en Windows

Asumo que tienes instalado MariaDB Server y que las herramientas de línea de comandos están accesibles a través del PATH del sistema (normalmente se instalan en C:\Program Files\MariaDB\Server X.X\bin).

A. Script para Backup Lógico Diario en Windows (Archivo .bat)

Este script usa mariadb-dump y las herramientas nativas de Windows para manejar fechas y compresión (requiere una herramienta como 7-Zip instalada y en el PATH para comprimir a .zip o .gz).

Nombre del archivo: backup_logico_diario.bat

batch

```
@echo off

REM --- Configuración ---

SET DB_USER=backup_user

SET DB_PASS=TuSuperContraseñaSegura

SET BACKUP_DIR=C:\Backups\MariaDB\Logical

REM Formato de fecha YYYYMMDD_HHMMSS

FOR /F "tokens=2 delims==" %%I IN ('WMIC OS GET LocalDateTime /VALUE') DO SET "dt=%%I"

SET DATE_FORMAT=%dt:~0,4%%dt:~4,2%%dt:~6,2%_%dt:~8,2%%dt:~10,2%%dt:~12,2%

SET BACKUP_FILE=%BACKUP_DIR%\full_backup_%DATE_FORMAT%.sql

REM Ruta a la herramienta de 7-Zip (ajusta si es necesario)

SET P7ZIP_CLI="C:\Program Files\7-Zip\7z.exe"

REM --- Fin Configuración ---

IF NOT EXIST "%BACKUP_DIR%" MKDIR "%BACKUP_DIR%"

ECHO Iniciando backup lógico completo de MariaDB...
```

```
REM El comando principal:
REM Redirige la salida estándar (STDOUT) a un archivo .sql
mariadb-dump -u %DB_USER% -p%DB_PASS% --all-databases --single-transaction > "%BACKUP_FILE%"

IF %ERRORLEVEL% EQU 0 (
    ECHO Backup SQL completado exitosamente. Comprimiendo...
    REM Comprimir el archivo SQL usando 7-Zip en formato gzip para eficiencia
    %P7ZIP_CLI% a "%BACKUP_FILE%.gz" "%BACKUP_FILE%" -tgzip

    IF %ERRORLEVEL% EQU 0 (
        ECHO Compresión completada.
        REM Eliminar el archivo .sql original después de comprimir
        DEL "%BACKUP_FILE%"
    ) ELSE (
        ECHO ¡ERROR en la compresión con 7-Zip!
    )

    REM Opcional: Eliminar backups .gz más antiguos de 7 días (esto es más complejo en .bat puro)
    REM Para una gestión avanzada de retención en Windows, Powershell es mejor.

) ELSE (
    ECHO ¡ERROR al realizar el backup con mariadb-dump!
)
ECHO Proceso finalizado.
```

Usa el código con precaución.

B. Script para Backup Físico Completo en Windows (Archivo .bat)

mariabackup funciona de manera muy similar en Windows. La principal diferencia es el manejo de rutas y permisos de usuario de Windows para acceder a los directorios de datos.

Nombre del archivo: backup_fisico_completo.bat

batch

```
@echo off

REM --- Configuración ---

SET BACKUP_DIR=C:\Backups\MariaDB\Physical

FOR /F "tokens=2 delims==" %%I IN ('WMIC OS GET LocalDateTime /VALUE') DO SET "dt=%%I"

SET DATE_FORMAT=%dt:~0,4%%dt:~4,2%%dt:~6,2%_%dt:~8,2%%dt:~10,2%%dt:~12,2%

SET TARGET_DIR=%BACKUP_DIR%\full_%DATE_FORMAT%

SET DB_USER=backup_user_physical

SET DB_PASS=OtraContraseñaSegura

REM --- Fin Configuración ---


IF NOT EXIST "%TARGET_DIR%" MKDIR "%TARGET_DIR%"


ECHO Iniciando backup físico completo con mariabackup en %TARGET_DIR%...


mariabackup --backup --target-dir="%TARGET_DIR%" --user="%DB_USER%" --password="%DB_PASS%"


IF %ERRORLEVEL% EQU 0 (
    ECHO Backup físico completado exitosamente.
    ECHO Preparando el backup para la restauración (commit de transacciones pendientes)...


    mariabackup --prepare --target-dir="%TARGET_DIR%"
```

```
    IF %ERRORLEVEL% EQU 0 (
        ECHO Backup preparado y listo para restaurar.
    ) ELSE (
        ECHO ¡ERROR al preparar el backup!
    )
) ELSE (
    ECHO ¡ERROR al realizar el backup físico!
)

ECHO Proceso finalizado.
```

Usa el código con precaución.

Parte 2: La Restauración y Verificación en Entorno Windows

Para la fase de control, el procedimiento es el mismo, pero los comandos se ejecutan en el **Command Prompt (CMD)** o **PowerShell** en tu entorno de pruebas de Windows.

Procedimiento de Restauración Física (Ejemplo con mariabackup)

Este es el proceso para verificar si el backup físico que acabas de crear funciona. **Recuerda hacerlo en un servidor de pruebas aislado.**

1. **Detener el Servicio de MariaDB en el entorno de pruebas:**

batch

```
net stop MariaDB
```

Usa el código con precaución.

2. **Limpiar el Directorio de Datos Antiguo:**
   (Ten mucho cuidado con este comando, asegúrate de estar en el entorno de pruebas)

batch

```
REM Ajusta la ruta a tu directorio de datos real de MariaDB
DEL /F /S /Q "C:\Program Files\MariaDB\Server X.X\data\*"
```

Usa el código con precaución.

3. **Restaurar los Archivos del Backup Preparado:**

batch

REM Ajusta la ruta a tu directorio de backup que contiene el backup "preparado"

SET BACKUP_PREPARADO=C:\Backups\MariaDB\Physical\full_20260108_160000

mariabackup --copy-back --target-dir="%BACKUP_PREPARADO%"

Usa el código con precaución.

4. **Ajustar Permisos (Crítico en Windows):**
   A veces, la cuenta de servicio de MariaDB pierde permisos sobre los archivos copiados. Debes asegurarte de que la cuenta de servicio (típicamente NT Service\mysql o similar) tenga control total sobre el directorio de datos.

5. **Iniciar el Servicio de MariaDB:**

batch

net start MariaDB

Usa el código con precaución.

6. **Validación:**
   Intenta conectarte con un cliente (mariadb.exe) y ejecuta una consulta simple:

batch

mariadb -u root -p -e "SELECT count(*) FROM nombre_de_tu_base_de_datos.nombre_de_tu_tabla_critica;"

Usa el código con precaución.

Automatización de Pruebas de Restauración en Windows

Automatizar esto requiere un entorno de prueba dedicado (staging) que idealmente esté en una máquina separada de la producción para garantizar un aislamiento total. Usaremos **PowerShell** para esta automatización, ya que es más potente que los archivos .bat para tareas complejas como la gestión de servicios y la validación de archivos.

Requisitos Previos para la Automatización

1. **Entorno de Staging:** Un segundo servidor Windows con MariaDB instalado, que no sea de producción.
2. **Rutas Consistentes:** El script asume rutas de backup y rutas de datos consistentes.
3. **Usuario de Servicio/Cuenta con Permisos:** La cuenta que ejecuta la tarea programada de PowerShell debe tener permisos para detener/iniciar el servicio de MariaDB y escribir/borrar en el directorio de datos.

Script de Prueba de Restauración Automatizada (PowerShell)

Este script:

1. Busca el último backup físico disponible.
2. Detiene el servicio de MariaDB de prueba.
3. Borra los datos de prueba existentes.
4. Restaura el backup físico.
5. Inicia el servicio.
6. Realiza una validación simple.

Nombre del archivo: Test-RestoreMariaDB.ps1

powershell

```
# --- Configuración ---
$BackupSourceDir = "C:\Backups\MariaDB\Physical"
$MariaDBDataDir = "C:\Program Files\MariaDB\Server X.X\data" # Ajusta tu versión de Server X.X
$MariaDBServiceName = "MariaDB" # Nombre exacto del servicio de Windows
$ValidationDatabase = "nombre_de_tu_base_de_datos_critica"
$ValidationQuery = "SELECT COUNT(*) FROM tabla_critica_para_validar"
$LogFile = "C:\Backups\MariaDB\restore_test_log.txt"
# --- Fin Configuración ---

Add-Content -Path $LogFile -Value "--- $(Get-Date) - Iniciando prueba de restauración ---"

# 1. Encontrar el backup más reciente
```

```
$LatestBackup = Get-ChildItem -Path $BackupSourceDir | Where-Object { $_.PSIsContainer -and $_.Name -like 'full_*' } | Sort-Object CreationTime -Descending | Select-Object -First 1

if ($LatestBackup -eq $null) {
    Add-Content -Path $LogFile -Value "ERROR: No se encontró ningún backup reciente en $BackupSourceDir. Falló la prueba."
    exit 1
}

Add-Content -Path $LogFile -Value "Usando el backup más reciente: $($LatestBackup.FullName)"

# 2. Detener el servicio de MariaDB de prueba
Add-Content -Path $LogFile -Value "Deteniendo el servicio de MariaDB..."
Stop-Service -Name $MariaDBServiceName -Force

# 3. Limpiar el directorio de datos de prueba
Add-Content -Path $LogFile -Value "Limpiando directorio de datos: $MariaDBDataDir"
Remove-Item -Path "$MariaDBDataDir\*" -Force -Recurse -ErrorAction SilentlyContinue

# 4. Restaurar los archivos (usando --copy-back)
Add-Content -Path $LogFile -Value "Iniciando restauración (--copy-back)..."
mariabackup --copy-back --target-dir="$($LatestBackup.FullName)" 2>&1 | Out-File -Append $LogFile
if ($LASTEXITCODE -ne 0) {
    Add-Content -Path $LogFile -Value "ERROR: mariabackup --copy-back falló. Falló la prueba."
    exit 1
}
```

```
# 5. Iniciar el servicio
Add-Content -Path $LogFile -Value "Iniciando el servicio de MariaDB..."
Start-Service -Name $MariaDBServiceName

# Esperar un momento a que la BD esté operativa
Start-Sleep -Seconds 30

# 6. Validar integridad de datos (requiere mariadb.exe en el PATH)
Add-Content -Path $LogFile -Value "Validando integridad de datos con consulta de prueba..."
$ValidationResult = mariadb -u root -pTuPasswordDeRootDePruebas -D $ValidationDatabase -e $ValidationQuery
$Count = ($ValidationResult | Select-Object -Last 1).Trim()

if ($Count -match "^\d+$") {
    Add-Content -Path $LogFile -Value "VALIDACIÓN EXITOSA. Conteo de filas en tabla crítica: $Count"
    Add-Content -Path $LogFile -Value "--- $(Get-Date) - Prueba de restauración COMPLETADA EXITOSAMENTE ---"
} else {
    Add-Content -Path $LogFile -Value "ERROR DE VALIDACIÓN: La consulta no devolvió un conteo válido. Revise la BD restaurada."
    Add-Content -Path $LogFile -Value "--- $(Get-Date) - Prueba de restauración FALLIDA ---"
    exit 1
}
```

Usa el código con precaución.

Programar la Ejecución en Windows

Puedes programar la ejecución de este script semanalmente usando el **Programador de Tareas de Windows (Task Scheduler)**:

1. Abre el Programador de Tareas.
2. Crea una nueva Tarea Básica.
3. Configura el disparador (Trigger) para que sea semanal, por ejemplo, cada domingo a las 2 AM.
4. En la acción, selecciona "Iniciar un programa".
5. Programa/script: powershell.exe
6. Añade argumentos: -ExecutionPolicy Bypass -File C:\Ruta\Al\Script\Test-RestoreMariaDB.ps1
7. Asegúrate de configurar la tarea para que se ejecute con una cuenta de usuario que tenga los permisos necesarios (marcar "Ejecutar con los privilegios más elevados" y "Ejecutar si el usuario ha iniciado sesión o no").

Una estrategia de gestión de copias de seguridad completa debe incluir un sistema robusto de notificación cuando las cosas van mal (o bien). Abordaremos la monitorización y las alertas para

Windows y **Linux**.

Monitorización y Alerta de Fallos

El objetivo principal es saber si los scripts de backup y los scripts de prueba de restauración se ejecutaron con éxito o fallaron, sin tener que iniciar sesión manualmente en los servidores todos los días.

Enfoque General: Códigos de Salida (Exit Codes)

Todos los scripts que hemos creado (.bat, .ps1, .sh) deben devolver un código de salida: 0 para éxito y cualquier otro valor (usualmente 1) para fallo. Las herramientas de automatización como cron (Linux) o el Task Scheduler (Windows) pueden leer este código y actuar en consecuencia.

---

1. Monitorización y Alerta en Linux

En Linux, las herramientas estándar como cron manejan las alertas por correo electrónico de forma nativa.

A. Alertas Básicas por Correo Electrónico (Cron y mailutils)

Cuando un trabajo de cron se ejecuta, si produce alguna salida (stdout o stderr), cron intenta enviar esa salida por correo electrónico al usuario propietario del *crontab* (si el sistema está configurado para enviar correos externos).

Configuración:

1. Asegúrate de tener un agente de transferencia de correo (MTA) como postfix o sendmail instalado y configurado en el servidor para enviar correos salientes.
2. Añade la variable MAILTO en tu *crontab*:

bash

 # Crontab entry para ejecutar el backup diario a las 2 AM

MAILTO="admin.email@example.com"

0 2 * * * /ruta/a/tuscript/backup_fisico_completo.sh

Usa el código con precaución.

Si el script se ejecuta sin errores (código de salida 0) y no produce salida estándar, no recibirás correo. Si falla (código de salida != 0) o produce errores, recibirás un correo con la salida del script.

B. Soluciones de Monitorización Centralizada

Para entornos de producción reales, las alertas por correo electrónico son limitadas. Es mejor integrar los scripts con herramientas de monitorización profesionales:

- **Prometheus/Grafana:** Puedes modificar tus scripts para que expongan métricas personalizadas (ej. mariadb_backup_status{type="full"} 0/1) que un agente de Prometheus recoja y Grafana visualice y alerte.
- **Servicios de Log Centralizado (ELK Stack/Splunk):** Los scripts escriben sus resultados en el syslog, y un agente reenvía estos logs a un servidor centralizado que dispara alertas basadas en palabras clave como "ERROR" o "FALLO".

---

2. Monitorización y Alerta en Windows

En Windows, dependemos principalmente del registro de eventos de Windows y de PowerShell para enviar notificaciones.

A. Alertas Básicas por Correo Electrónico (PowerShell)

El script de PowerShell que ejecuta el backup o la prueba de restauración debe incluir lógica para enviar un correo electrónico si detecta un fallo (if ($LASTEXITCODE -ne 0)).

Añade una función como esta a tus scripts de PowerShell:

powershell

```
function Send-AlertEmail {
    param (
        [string]$Subject,
        [string]$Body,
        [string]$SMTPServer = "smtp.tuservidor.com",
        [string]$From = "backup-service@example.com",
        [string]$To = "admin.email@example.com"
    )
    Send-MailMessage -To $To -From $From -Subject $Subject -Body $Body -SmtpServer $SMTPServer -BodyAsHtml
}


# Ejemplo de uso en caso de fallo dentro del script:
# ...
if ($LASTEXITCODE -ne 0) {
    $ErrorBody = Get-Content -Path $LogFile | Out-String
    Send-AlertEmail -Subject "FALLO: Backup MariaDB Producción" -Body "Ha ocurrido un error en el backup. Detalles del log:<br><pre>$ErrorBody</pre>"
    exit 1
}
# ...
```

Usa el código con precaución.

B. Monitorización Integrada con el Programador de Tareas

El **Programador de Tareas de Windows** puede generar eventos en el Visor de Eventos cuando una tarea falla o termina con un código de error.

1. Configura la tarea para que, en caso de fallo, escriba en el registro de eventos.
2. Utiliza el "Visor de Eventos" para crear una **tarea adjunta** a ese evento específico (ID 101, por ejemplo) que, a su vez, ejecute un script de alerta o envíe un correo electrónico (esto último es menos fiable en versiones modernas de Windows).

C. Soluciones de Monitorización Centralizada (Windows)

Al igual que en Linux, para producción se usan herramientas como:

- **Microsoft SCOM o Azure Monitor:** Los scripts pueden escribir eventos personalizados en el Visor de Eventos de Windows, y el agente de monitorización de Azure/SCOM los recoge y dispara alertas.
- **Integración con Slack o Teams:** En lugar de enviar un correo, los scripts de PowerShell pueden usar webhooks para enviar notificaciones directamente a un canal de chat de operaciones.

Parte 1: Script de Creación de la Base de Datos y Tablas

Este script crea una base de datos de ejemplo llamada gestion_inventario y dos tablas: productos y ordenes_compra.

Nombre del archivo: 01_create_database.sql

sql

```
-- 01_create_database.sql


-- Crear la base de datos si no existe
CREATE DATABASE IF NOT EXISTS gestion_inventario;


-- Usar la base de datos recién creada
USE gestion_inventario;
```

```sql
-- Crear la tabla de productos
CREATE TABLE IF NOT EXISTS productos (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    sku VARCHAR(50) NOT NULL UNIQUE,
    precio DECIMAL(10, 2) NOT NULL,
    stock INT NOT NULL DEFAULT 0,
    fecha_creacion TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Crear la tabla de órdenes de compra
CREATE TABLE IF NOT EXISTS ordenes_compra (
    id INT AUTO_INCREMENT PRIMARY KEY,
    producto_id INT NOT NULL,
    cantidad INT NOT NULL,
    fecha_orden TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (producto_id) REFERENCES productos(id)
);

-- Insertar algunos datos iniciales para empezar
INSERT INTO productos (nombre, sku, precio, stock) VALUES
('Laptop Gamer XYZ', 'LAPT-XYZ-01', 1250.00, 50),
('Monitor Curvo 27"', 'MON-CURV-27', 300.00, 150),
('Teclado Mecánico RGB', 'TEC-RGB-MX', 85.50, 300);
```

INSERT INTO ordenes_compra (producto_id, cantidad) VALUES

(1, 5),

(2, 10),

(3, 20),

(1, 2);

SELECT 'Base de datos y datos iniciales creados exitosamente.' AS Status;

Usa el código con precaución.

Cómo ejecutar este script:

Abre tu terminal (CMD, PowerShell o Bash en Linux) y ejecuta:

bash

mariadb -u root -p < 01_create_database.sql

Usa el código con precaución.

Ingresa tu contraseña de root cuando se te solicite.

Parte 2: Script para Generar Datos Aleatorios (Poblamiento)

Este script (.sql) insertará 1000 filas adicionales de datos ficticios en la tabla productos y ordenes_compra para hacer la base de datos un poco más grande y simular actividad.

Nombre del archivo: 02_generate_data.sql

sql

-- *02_generate_data.sql*

USE gestion_inventario;

-- *Desactivar temporalmente el autocommit y la verificación de FK para inserciones masivas*

SET autocommit=0;

SET foreign_key_checks=0;

```
-- Simulación de inserción masiva de productos ficticios (1000 filas)
-- Nota: En un entorno real usarías un generador de datos más sofisticado o LOAD DATA INFILE.
-- Esto es solo ilustrativo.
INSERT INTO productos (nombre, sku, precio, stock)
SELECT
   CONCAT('Producto Aleatorio ', FLOOR(RAND() * 100000)),
   CONCAT('SKU-', FLOOR(RAND() * 100000000)),
   ROUND(RAND() * 500 + 10, 2),
   FLOOR(RAND() * 100)
FROM information_schema.tables AS t1, information_schema.tables AS t2
LIMIT 1000;

-- Simulación de inserción de 2000 órdenes de compra aleatorias
INSERT INTO ordenes_compra (producto_id, cantidad)
SELECT
   FLOOR(RAND() * 1000) + 1, -- ID de producto aleatorio (asume que los IDs existen entre 1 y 1003)
   FLOOR(RAND() * 50) + 1
FROM information_schema.tables AS t1, information_schema.tables AS t2
LIMIT 2000;

COMMIT;

SET foreign_key_checks=1;

SELECT 'Datos de prueba adicionales generados exitosamente.' AS Status;
```

Usa el código con precaución.

Cómo ejecutar este script para poblar la BD:

bash

mariadb -u root -p < 02_generate_data.sql

Usa el código con precaución.

Parte 3: Aplicar los Scripts de Copia de Seguridad

Ahora que tienes la base de datos gestion_inventario con datos, puedes usar los scripts de backup que se presentaron anteriormente:

1. Ejecutar Backup Lógico (Windows .bat o Linux .sh)

Modifica la sección de configuración de tu script de backup lógico para respaldar **solo** esta nueva base de datos específica, en lugar de --all-databases.

Cambia esta línea:

bash

 # En Linux .sh:

mariadb-dump -u "$DB_USER" -p"$DB_PASS" --all-databases --single-transaction | gzip > "$BACKUP_FILE"

 # En Windows .bat:

mariadb-dump -u %DB_USER% -p%DB_PASS% --all-databases --single-transaction > "%BACKUP_FILE%"

Usa el código con precaución.

Por esta línea (especificando la BD):

bash

 # En Linux .sh:

mariadb-dump -u "$DB_USER" -p"$DB_PASS" gestion_inventario --single-transaction | gzip > "$BACKUP_FILE"

 # En Windows .bat:

mariadb-dump -u %DB_USER% -p%DB_PASS% gestion_inventario --single-transaction > "%BACKUP_FILE%"

Usa el código con precaución.

Ejecuta tu script de backup lógico (backup_logico_diario.bat o .sh).

2. Ejecutar Backup Físico (Windows .bat o Linux .sh)

Los scripts físicos de mariabackup siempre operan a nivel de instancia completa, pero capturarán los archivos de datos de gestion_inventario de manera eficiente.

Ejecuta tu script de backup físico (backup_fisico_completo.bat o .sh).