UTTec


# PORTAFOLIO DE EVIDENCIAS
# SEGUNDO PARCIAL

## ADMINISTRACIÓN DE BASES DE DATOS

## 8IDS1

*PRESENTA*

**MARTINEZ DE LA CRUZ JOSE JULIAN**

INGENIERÍA EN DESARROLLO Y GESTIÓN DE SOFTWARE

NOMBRE DEL PROFESOR: MTI. JESÚS ESTANISLAO ROMERO MORENO

CUATRIMESTRE: MAYO – AGOSTO 2025

Universidad Tecnológica de Tecámac

Organismo Público Descentralizado del Gobierno del Estado de México


# Automatización de la Administración de Máquinas Virtuales con VBoxManage

**MTI JESÚS E. ROMERO MORENO**

**DIVISIÓN DE TECNOLOGÍAS DE LA INFORMACIÓN Y COMUNICACIÓN**

# Automatización de la Administración de Máquinas Virtuales con VBoxManage

Hoy exploraremos cómo mejorar la **automación en la administración de máquinas virtuales** utilizando **VBoxManage** y archivos por lotes (**.bat**). La administración eficiente y automatizada es esencial para garantizar la **continuidad operativa** de los sistemas y prevenir posibles **desastres informáticos**. Por lo tanto, en esta sesión nos enfocaremos en proporcionarte las herramientas necesarias para que puedas automatizar procesos y asegurar que tu infraestructura virtual esté preparada para cualquier eventualidad.

**Objetivo de la Sesión**

El objetivo principal de hoy es que, al finalizar, seas capaz de **modificar y mejorar un archivo .bat** que automatiza varias tareas de administración de máquinas virtuales. Con esto, no solo optimizarás tu flujo de trabajo, sino que también garantizarás que los sistemas virtuales funcionen de manera eficiente y estén preparados para afrontar problemas técnicos antes de que se conviertan en fallos críticos.

**Recursos Proporcionados**

1. **Archivo .bat**: Te proporcionaremos un script por lotes que ya realiza algunas operaciones comunes, como la creación y configuración de máquinas virtuales. Este archivo es tu punto de partida.
2. **Listado de Comandos de VBoxManage**: Además, tendrás acceso a un **listado detallado de comandos**, que te permitirá entender mejor las funcionalidades que puedes agregar o modificar dentro del archivo .bat.
3. **Tarea de Investigación**: Es fundamental que, además de trabajar con estos recursos, realices una investigación adicional sobre cómo puedes optimizar el script y agregar funciones avanzadas, como la gestión de snapshots o la ejecución en modo headless.

**Estructura de la Sesión**

1. **Análisis del Archivo .bat**: Comenzaremos analizando el archivo proporcionado, explicando cada sección y los comandos de **VBoxManage** que utiliza.
2. **Modificación y Mejora del Script**: Tu misión será **modificar y mejorar**este archivo, agregando nuevos comandos que automaticen tareas adicionales, como:
    - Toma de snapshots automáticos (VBoxManage snapshot),
    - Inicio y detención de máquinas virtuales en distintos modos (VBoxManage startvm),

- Gestión de almacenamiento y discos duros virtuales (VBoxManage storageattach).

3. **Aplicación de los Conocimientos**: Utilizarás el listado de comandos y tu investigación para hacer que el archivo .bat sea más robusto y funcional.

**Misión Final**

El propósito es que, al finalizar la sesión, hayas creado un script automatizado que no solo sea capaz de **crear y gestionar máquinas virtuales**, sino que también sea capaz de **responder a eventos críticos**. Esto incluye la capacidad de restaurar snapshots y apagar máquinas de manera controlada para evitar desastres informáticos.

**¿Qué esperamos de ti?**

- **Explorar** el archivo .bat proporcionado y comprender cómo interactúa con **VBoxManage**.
- **Investigar** cómo puedes mejorar su funcionalidad mediante comandos adicionales.
- **Modificar** el script para adaptarlo a escenarios más avanzados, como la gestión de snapshots, la programación de tareas y la creación de entornos virtuales más complejos.

Este ejercicio te permitirá dominar la automatización en entornos virtuales, lo que es esencial para mantener la estabilidad de los sistemas y la recuperación rápida en caso de fallos.

**Tabla de Comandos de VirtualBox**

A continuación se presenta una tabla que detalla los comandos más comunes de VirtualBox junto con sus parámetros, descripciones, ejemplos y resultados esperados. Esta tabla es una referencia útil para usuarios que deseen gestionar máquinas virtuales a través de la interfaz de línea de comandos (CLI) de VirtualBox, **VBoxManage**.

| Comando | Parámetros | Descripción | Ejemplo | Resultado Esperado |
|---|---|---|---|---|
| **VBoxManage createvm** | **--name <vmname> --register** | Crea una nueva máquina virtual y la registra en VirtualBox. | **VBoxManage createvm --name "UbuntuVM" --register** | Crea una nueva máquina virtual llamada "UbuntuVM" y la registra en VirtualBox. |
| **VBoxManage modifyvm** | **<vmname> --memory <size> --cpus <count>** | Modifica las propiedades de una máquina virtual existente. | **VBoxManage modifyvm "UbuntuVM" --memory 2048 --cpus 2** | Modifica la VM "UbuntuVM" para que use 2048 MB de memoria y 2 CPUs. |
| **VBoxManage createhd** | **--filename <path> --size <size>** | Crea un nuevo disco duro virtual. | **VBoxManage createhd --filename "C:\vms\UbuntuVM.vdi" --size 20000** | Crea un nuevo disco duro virtual de 20 GB en la ruta especificada. |
| **VBoxManage storagectl** | **<vmname> --name <name> --add <type> --controller <controller>** | Agrega un controlador de almacenamiento a una VM. | **VBoxManage storagectl "UbuntuVM" --name "SATA Controller" --add sata --controller IntelAhci** | Agrega un controlador SATA llamado "SATA Controller" a la VM "UbuntuVM". |
| **VBoxManage storageattach** | **<vmname> --storagectl <name> --port <port> --device <device> --type <type> --medium <path>** | Adjunta un disco duro o un medio de CD/DVD a una VM. | **VBoxManage storageattach "UbuntuVM" --storagectl "SATA Controller" --port 0 --device 0 --type hdd --medium "C:\vms\UbuntuVM.vdi"** | Adjunta el disco duro especificado a la VM "UbuntuVM" en el puerto 0 del controlador "SATA Controller". |
| **VBoxManage startvm** | **<vmname> --type <type>** | Inicia una máquina virtual. | **VBoxManage startvm "UbuntuVM" --type headless** | Inicia la VM "UbuntuVM" en modo headless (sin GUI). |
| **VBoxManage controlvm** | **<vmname> poweroff** | Apaga una máquina virtual. | **VBoxManage controlvm "UbuntuVM" poweroff** | Apaga inmediatamente la VM "UbuntuVM". |
| **VBoxManage snapshot** | **<vmname> take <snapshotname> --description <description>** | Toma una instantánea (snapshot) de la VM especificada. | **VBoxManage snapshot "UbuntuVM" take "Snapshot1" --description "Estado inicial"** | Crea una instantánea llamada "Snapshot1" de la VM "UbuntuVM" con la descripción "Estado inicial". |

| | | | | |
|---|---|---|---|---|
| **VBoxManage list vms** | | Lista todas las máquinas virtuales registradas en VirtualBox. | **VBoxManage list vms** | Muestra una lista de todas las máquinas virtuales registradas en VirtualBox. |
| **VBoxManage list runningvms** | | Lista todas las máquinas virtuales en ejecución. | **VBoxManage list runningvms** | Muestra una lista de todas las máquinas virtuales que están actualmente en ejecución. |
| **VBoxManage unregistervm** | **<vmname> --delete** | Desregistra y elimina una máquina virtual. | **VBoxManage unregistervm "UbuntuVM" --delete** | Desregistra y elimina la VM "UbuntuVM" junto con todos sus archivos asociados. |
| **VBoxManage showvminfo** | **<vmname>** | Muestra información detallada sobre una máquina virtual específica. | **VBoxManage showvminfo "UbuntuVM"** | Muestra información detallada sobre la VM "UbuntuVM". |
| **VBoxManage list hdds** | | Lista todos los discos duros virtuales registrados en VirtualBox. | **VBoxManage list hdds** | Muestra una lista de todos los discos duros virtuales registrados en VirtualBox. |
| **VBoxManage list snapshots** | **<vmname>** | Lista todas las instantáneas (snapshots) de una máquina virtual específica. | **VBoxManage list snapshots "UbuntuVM"** | Muestra una lista de todas las instantáneas de la VM "UbuntuVM". |
| **VBoxManage snapshot restore** | **<vmname> <snapshotname>** | Restaura una instantánea específica de una máquina virtual. | **VBoxManage snapshot "UbuntuVM" restore "Snapshot1"** | Restaura la VM "UbuntuVM" al estado guardado en la instantánea "Snapshot1". |
| **VBoxManage snapshot delete** | **<vmname> <snapshotname>** | Elimina una instantánea específica de una máquina virtual. | **VBoxManage snapshot "UbuntuVM" delete "Snapshot1"** | Elimina la instantánea "Snapshot1" de la VM "UbuntuVM". |

**CIERRE DE TEMA**

# AUTOMATIZACIÓN DE LA ADMINISTRACIÓN DE MAQUINAS VIRTUALES CON VBOXMANAGE

Por Martinez de la Cruz Jose Julian

Fecha: 10/06/2025

Palabras clave: VirtualBox, VBoxManage, Automatización, Máquina Virtual (VM), Interfaz de Línea de Comandos (CLI), Script, .bat, Oracle.

## 1. Objetivo de la práctica

El propósito de esta práctica es conocer el funcionamiento de VBoxManage para lograr la automatización de varias tareas relacionadas con la administración de máquinas virtuales a través de la modificación y mejora de un archivo .bat para optimizar el flujo de trabajo.

## 2. Resumen

VBoxManage es la interfaz de línea de comandos (CLI) de Oracle VirtualBox, la cual permite gestionar y modificar la configuración tanto de VirtualBox como la de cada máquina virtual (VM), sin depender de la interfaz gráfica. Esto permite una administración remota para realizar tareas de manera más rápida y automática.

Para lograr una automatización de las máquinas virtuales se realizaron modificaciones en el archivo .bat proporcionado por el profesor. Las modificaciones añadidas incluyen:

- **Listar las máquinas virtuales registradas en VirtualBox**
  - Permite visualizar todas las máquinas virtuales que han sido registradas en el sistema, junto con su nombre y su Identificador Único Universal (UUID).
  - Ejecuta internamente el comando VBoxManage list vms
- **Listar todas las máquinas virtuales en ejecución**
  - Permite visualizar únicamente las VMs que están activas lo que permite supervisar el estado del sistema.
  - Ejecuta internamente el comando VBoxManage list runningvms
- **Desregistrar y eliminar una máquina virtual**
  - Permite eliminar completamente una VM tanto del registro como del disco para liberar recursos del sistema.
  - Ejecuta internamente el comando VBoxManage unregistervm " vname " --delete
- **Mostrar la información detallada sobre una máquina virtual específica**
  - Muestra todos los datos técnicos completos sobreuna VM, desde el hardware asignado hasta las instantáneas creadas.
  - Ejecuta internamente el comando VBoxManage showvminfo "vname"

De esta manera se puede asegurar que el script funcione en entornos controlados, pruebas, administración o automatización convirtiéndolo en una herramienta completa eficiente e incluso vendible.

**3. Conclusión**

Implementar VBoxManage para la automatización de tareas es una solución más rápida para administrar todas las máquinas virtuales sin necesidad de interactuar con la interfaz gráfica de VBox y sobre todo fácil de migrar (en caso de que se tenga que implementar en otros S.O).

De esta manera se vuelve un proceso menos complejo y sin intervención manual, además de que se vuelve una herramienta más fácil de usar para las personas que no tienen tanta experiencia con la interfaz gráfica, ya que al tener un menú intuitivo se minimizan los errores humanos de esta manera se reducen bastante los tiempos para gestionar una máquina virtual.

**4. Referencias**


- Presentación: *Tabla de Comandos de VirtualBox*, MTI Jesús E. Romero, UTTEC.
- ORACLE (2024). *User Guide for Release 7.0*


**5. Reflexiones Finales**

1. **Ahorro de Tiempo:** La automatización con VBoxManage permite ahorrar tiempo ya que al tener todo en un script con un menú intuitivo, se puede ejecutar cualquier acción con solo un comando de manera más rápida.
2. **Facilidad de Uso:** Al tener un menú intuitivo se evita la necesidad de recordar los comandos o incluso de perderse entre todas las opciones con las que cuenta la interfaz gráfica reduciendo los errores humanos.
3. **Herramienta Vendible:** Si se realiza un script completo con muchísimos más comandos y además de bien hecho, puede ser vendido a las empresas como una solución lista para usar. Al ser portable, fácil de usar y adaptable a distintos S.O, se convierte en una herramienta bastante útil.

Universidad Tecnológica de Tecámac

Organismo Público Descentralizado del Gobierno del Estado de México


# Automatización de Tareas en Windows y Linux para la Administración de Bases de Datos y Manejo de Incidentes

**MTI JESÚS E. ROMERO MORENO**

**DIVISIÓN DE TECNOLOGÍAS DE LA INFORMACIÓN Y COMUNICACIÓN**

**Automatización de Tareas en Windows y Linux para la Administración de Bases de Datos y Manejo de Incidentes**

La automatización de tareas es fundamental en la administración de sistemas y bases de datos, especialmente cuando se requiere una gestión eficiente, rápida y sin intervención manual constante. Herramientas como **schtasks** en Windows y **cron** en Linux permiten programar la ejecución automática de tareas, lo que optimiza el rendimiento y reduce la posibilidad de errores humanos. Estos sistemas de programación son esenciales para implementar soluciones de administración de bases de datos y respuesta a incidentes, ya que pueden ejecutarse sin necesidad de intervención constante, permitiendo una gestión más eficiente y confiable.

**Automatización en Windows con schtasks**

En **Windows**, la herramienta **schtasks** es una forma robusta de automatizar la ejecución de scripts, aplicaciones o tareas del sistema. Al configurarla, es posible programar tareas que se ejecuten en intervalos específicos, como la creación de copias de seguridad de bases de datos, la ejecución de scripts de mantenimiento o el monitoreo del rendimiento del sistema.

Por ejemplo, podemos crear una tarea programada que ejecute un script de respaldo de una base de datos a las 2:00 AM todos los días:

**cmd**

**schtasks /create /tn "BackupDB" /tr "C:\Scripts\backup_db.bat" /sc daily /st 02:00**

Esto no solo garantiza que la base de datos esté respaldada regularmente, sino que también libera a los administradores de la necesidad de ejecutar manualmente estas tareas. Además, en el contexto de manejo de incidentes, se pueden automatizar procesos como la monitorización del sistema, alertas de errores, o la ejecución de procedimientos de recuperación ante fallos.

**Comando básico para crear una tarea programada:**

cmd

schtasks /create /tn "MiTarea" /tr "C:\Ruta\al\script.bat" /sc daily /st 09:00

- /tn "MiTarea": Especifica el nombre de la tarea.
- /tr "C:\Ruta\al\script.bat": Define el programa o script que quieres ejecutar.
- /sc daily: Especifica que la tarea se ejecutará todos los días.
- /st 09:00: Indica que la tarea debe ejecutarse a las 9:00 AM.

**Otros parámetros útiles:**

- /sc weekly para ejecutarse cada semana.
- /sc monthly para ejecutarse cada mes.

- /ru para definir el usuario bajo el cual se ejecutará la tarea.
- /f para forzar la creación de la tarea (si ya existe).

**Automatización en Linux con cron**

En **Linux**, el **cron** es la herramienta estándar para la programación de tareas repetitivas. A través de su archivo de configuración crontab, se pueden definir tareas programadas que se ejecuten en función del tiempo, facilitando la automatización de tareas relacionadas con la administración de bases de datos, como la limpieza de registros antiguos, la ejecución de actualizaciones, o la verificación de la integridad de la base de datos.

Por ejemplo, podemos crear una tarea cron para verificar la integridad de la base de datos y enviar un informe por correo electrónico todos los lunes a las 3:00 AM:

**bash**

**0 3 * * 1 /ruta/al/script_integridad_db.sh**

Este tipo de automatización asegura que se detecten problemas potenciales con la base de datos antes de que afecten el rendimiento o la disponibilidad del sistema. Además, en situaciones de incidentes, cron puede ser utilizado para ejecutar procesos de remediación, como el reinicio de servicios de base de datos, la recopilación de registros o la ejecución de diagnósticos de sistema.

**Sintaxis del archivo crontab:**

```
lua

* * * * * /ruta/al/script

- - - - -

| | | | |

| | | | +--- Día de la semana (0 - 7) (domingo = 0 o 7)

| | | +----- Mes (1 - 12)

| | +------- Día del mes (1 - 31)

| +--------- Hora (0 - 23)

+----------- Minuto (0 - 59)
```

**Ejemplo de tarea cron:**

```
Para ejecutar un script todos los días a las 3 AM:

bash

0 3 * * * /ruta/al/script.sh
```

**Otros ejemplos comunes de cron:**

- **Cada hora:**

bash

```
0 * * * * /ruta/al/script.sh
```

- **Cada 15 minutos:**

bash

```
*/15 * * * * /ruta/al/script.sh
```

Puedes verificar tus cron jobs con el comando crontab -l y eliminar un cron job con crontab -r.

**Importancia de la Automatización para la Administración de Bases de Datos**

La administración de bases de datos involucra tareas repetitivas y críticas que requieren ser gestionadas sin demora para mantener la integridad, disponibilidad y seguridad de los datos. Las herramientas **schtasks** y **cron** permiten a los administradores de bases de datos programar y automatizar tareas esenciales como:

- **Copia de seguridad regular**: Crear backups automáticos previene la pérdida de datos en caso de un desastre o incidente.
- **Mantenimiento periódico**: La ejecución de scripts de mantenimiento como la optimización de índices o la limpieza de registros antiguos ayuda a mantener el rendimiento del sistema.
- **Monitoreo constante**: Las tareas programadas permiten la supervisión continua de los sistemas, alertando sobre problemas antes de que se conviertan en incidentes críticos.

**Implementación de Acciones en Caso de Detectar un Incidente**

Ambas herramientas, **schtasks** y **cron**, no solo facilitan la automatización de tareas de administración, sino que también son cruciales para implementar medidas de respuesta ante incidentes. Algunos ejemplos incluyen:

- **Notificación de incidentes**: Al detectar un fallo o comportamiento anómalo en la base de datos, se puede programar una tarea que envíe un correo electrónico o mensaje a los administradores para tomar acción inmediata.
- **Automatización de recuperación**: En caso de que un servicio de base de datos falle, se pueden configurar tareas programadas para reiniciar el servicio o ejecutar procedimientos de recuperación sin intervención manual.

- **Monitoreo proactivo**: Con herramientas como cron o schtasks, es posible ejecutar scripts que monitoricen constantemente los logs del sistema y la base de datos, generando alertas cuando se detectan errores específicos, como bloqueos de base de datos o caídas del servidor.

**Conclusión**

La automatización de tareas mediante **schtasks** en Windows y **cron** en Linux representa una herramienta poderosa para la administración eficiente de bases de datos y la implementación de acciones automáticas en respuesta a incidentes. La capacidad de programar tareas críticas, como copias de seguridad, mantenimiento, o alertas de errores, no solo mejora la eficiencia operativa, sino que también incrementa la resiliencia y disponibilidad de los sistemas de base de datos. Al automatizar estos procesos, los administradores pueden centrarse en tareas estratégicas, dejando las tareas repetitivas y de monitoreo en manos de estas herramientas, lo que reduce riesgos y mejora la respuesta ante incidentes.

# Solicitud de Creación de Automatizaciones para Administración de Bases de Datos y Virtualización

**Objetivo:**
El propósito de esta actividad es que cada estudiante desarrolle tres automatizaciones utilizando las opciones del shell que se han explicado en clase, con el objetivo de garantizar la ejecución de actividades sin la intervención directa del administrador del sistema. Las automatizaciones deben estar orientadas a dos áreas clave: **Administración de Bases de Datos** y **Virtualización**.

**Requisitos Generales:**

1. **Automatización de Bases de Datos:** El estudiante deberá crear al menos una automatización relacionada con la administración de bases de datos. Esta podría incluir tareas como la creación de copias de seguridad, la optimización de índices o el monitoreo de la integridad de los datos.
2. **Automatización de Virtualización:** El estudiante deberá crear al menos una automatización relacionada con la gestión de entornos de virtualización, como la creación, destrucción o gestión de máquinas virtuales, así como la gestión de recursos.
3. **Permisos y Privilegios de Usuario:** Es esencial que, al programar estas actividades, se tomen en cuenta los permisos y privilegios de usuario necesarios para garantizar que las tareas se ejecuten correctamente sin comprometer la seguridad del sistema. Asegúrese de que las tareas automatizadas se ejecuten con el usuario adecuado o los permisos necesarios para evitar errores de ejecución o problemas de seguridad.
4. **Investigación y Pruebas:** Cada automatización debe ser investigada, probada y demostrada en un entorno real o simulado, validando su funcionalidad antes de presentarla. El estudiante debe documentar cómo realizó la investigación y las pruebas, explicando cómo verificó que la automatización funciona de manera correcta y confiable.
5. **Plataforma de Ejecución:** Los estudiantes pueden optar por utilizar **Windows** o **Linux** para realizar las automatizaciones, según sus preferencias y el entorno que estén utilizando.
6. **Trabajo Individual:** Esta actividad debe ser realizada de manera **individual**. No se permite la colaboración con otros compañeros del salón, y la solución entregada debe ser única para cada estudiante. Las automatizaciones no deben ser repetidas entre estudiantes.

**Entrega:**

1. **Documentación Técnica:** Incluya una breve descripción de cada automatización creada, explicando qué tarea automatiza, cómo se ha configurado y qué resultado se espera de su ejecución.
2. **Código Fuente:** Adjunte los scripts o comandos utilizados para configurar cada una de las automatizaciones, con los comentarios necesarios para explicar el funcionamiento de cada línea.
3. **Demostración de Funcionalidad:** Proporcione evidencias de que las automatizaciones funcionan correctamente, ya sea mediante capturas de pantalla, registros de salida o videos cortos mostrando su ejecución.

**Evaluación:**

- **Investigación y Pruebas:** Se evaluará la calidad de la investigación realizada para elegir la solución adecuada, así como la exhaustividad de las pruebas realizadas.
- **Funcionalidad:** Se evaluará si las automatizaciones cumplen su propósito de manera eficiente y sin requerir intervención del administrador.
- **Seguridad y Permisos:** Se tendrá en cuenta que las tareas automatizadas respeten las políticas de seguridad del sistema, con los permisos y privilegios adecuados.
- **Documentación y Presentación:** La claridad y calidad de la documentación será parte de la evaluación, así como la estructura de los scripts presentados.

**CIERRE DE TEMA**

# AUTOMATIZACIÓN DE TAREAS EN WINDOWS PARA LA ADMINISTRACIÓN DE BASES DE DATOS Y MANEJO DE INCIDENTES

Por Martínez de la Cruz José Julián

Fecha: 12/06/2025

Palabras clave: Schtask, CLI, VirtualBox, VBoxManage, Automatización, Programar tareas, Programador de Tareas, Windows, Snapshots, Backups, Shell, VM, Máquina Virtual, mkdir.

## 1. Objetivo del Tema

El propósito de esta práctica es lograr la implementación de tres automatizaciones de tareas enfocadas a la virtualización utilizando las opciones de la línea de comandos de VBoxManage y Shell de Windows, con el objetivo de garantizar la ejecución de actividades de respaldo y monitoreo de máquinas virtuales sin la intervención directa del administrador del sistema.

## 2. Resumen

Para la realización y cumplimiento de esta práctica se decidió implementar un menú intuitivo que dependiendo de la opción seleccionada cree un nuevo archivo .bat mediante el comando mkdir, esto permite la creación automática de un script que realiza las siguientes automatizaciones enfocadas a la gestión de máquinas virtuales mediante el CLI (Línea de comandos) de VBoxManage y después crea una tarea programada usando el comando schtask en base al script generado previamente. De modo que cada una de estas tareas responda a una necesidad real de mantenimiento evitando los errores que pueda traer consigo la intervención manual.

1. **Programar una tarea para realizar Snapshots de una Máquina Virtual**

   La implementación de esta automatización permite crear puntos de restauración en caso de que ocurran fallos y de esta manera tenerlos controlados. Esto asegura que exista siempre una copia reciente de la máquina virtual.

2. **Programar una tarea que permita eliminar los Snapshots de una Máquina Virtual**

   Esta automatización permite eliminar Snapshots en un cierto periodo de tiempo para evitar los consumos innecesarios de recursos generados por Snapshots mal gestionadas evitando que el rendimiento de la VM se reduzca.

3. **Programar una tarea para realizar Backups periódicos de una Máquina Virtual**

   Esta tarea permite generar una copia externa completa de una VM de modo que cuando se requiera pueda ser restaurada en caso de que la Máquina Virtual original sufra daños.

4. **Programar una tarea que permita eliminar periódicamente los Backups de una Máquina Virtual**

   De la misma manera que con la eliminación de Snapshots, tener una tarea programada que permita la eliminación periódica de Backups permite preservar el espacio y evitar problemas relacionados con el rendimiento del sistema.

5. **Opción para eliminar una tarea schtask directamente desde el Shell de Windows**

   Esta acción permite a los usuarios que no están experimentados con la programación de tareas de Windows, una manera intuitiva de eliminar una tarea programada creadas por el usuario desde la consola de esta, de esta manera se reduce el riesgo de errores generados por usuarios sin experiencia.

La realización de estas automatizaciones presenta sin duda alguna una mejora significativa la manera en que una empresa realiza sus operaciones y les permite la creación de políticas de respaldo y monitoreo para la realización de un entorno de pruebas controlado.

## 3. Conclusión

La automatización de tareas permite a las empresas realizar mantenimiento de sus sistemas en un entorno controlado y mantener su continuidad operativa. En este sentido, la implementación de la herramienta schtask de Windows permite la programación de tareas periódicas para gestionar correctamente estos entornos virtuales sin depender totalmente de la intervención manual, lo que mejora totalmente estos procesos. Además, permite a los usuarios que no están experimentados en el área poder realizar estas tareas de manera intuitiva, de esta manera no se depende en su totalidad del Administrador de Base de Datos.

## 4. Referencias


- Presentación: *Automatización de Tareas en Windows y Linux para la Administración de Base de Datos y Manejo de Incidentes*, MTI Jesús E. Romero, UTTEC.
- Microsoft. (2023). *schtasks.* Microsoft Learn.
- Microsoft. (2023). *mkdir.* Microsoft Learn.


## 5. Reflexiones Finales

1. **Los archivos bat automatizados reducen tiempo y errores humanos:** La intervención manual constante en tareas repetitivas como Backups o Snapshots aumenta el margen de error y consume tiempo operativo. La automatización permite mantener entornos funcionales incluso en ausencia del personal técnico, garantizando la continuidad del servicio.

2. **Usar herramientas del SO sin instalar algo más:** Utilizar funcionalidades ya integradas en el sistema operativo como lo es schtask o mkdir reduce la complejidad del entorno. Esto es especialmente útil en organizaciones con recursos limitados que buscan soluciones efectivas sin agregar más aplicaciones a sus operaciones.

Universidad Tecnológica de Tecámac

Organismo Público Descentralizado del Gobierno del Estado de México


# Tabla de referencia de comandos

**MTI JESÚS E. ROMERO MORENO**

**DIVISIÓN DE TECNOLOGÍAS DE LA INFORMACIÓN Y COMUNICACIÓN**

# Tabla de referencia de comandos BK y Restauración

Estos comandos y opciones te permiten personalizar el respaldo de la base de datos **mi_financiera_demo** para adaptarse a diferentes necesidades y escenarios, como respaldos parciales, con diferentes niveles de detalle, compresión o formatos específicos.

Esta tabla es específica para la base de datos **mi_financiera_demo** en un entorno Windows, utilizando la ruta fija de respaldo en **C:\bkdb**.

**Tabla de Comandos de Respaldo para PostgreSQL (**pg_dump**)**

| Comando | Ejemplo | Descripción |
|---|---|---|
| pg_dump -U <usuario> <nombre_bd> | pg_dump-U postgres mi_financiera_demo > "C:\bkdb\respaldo.sql" | Realiza un respaldo completo de la base de datos especificada como el usuario **postgres**. |
| -f <archivo> | pg_dump-U postgres-f "C:\bkdb\respaldo_mi_financiera_demo.sql" mi_financiera_demo | Guarda el respaldo en un archivo específico, ejecutado como el usuario **postgres**. |
| -F <formato> | pg_dump-U postgres-F c mi_financiera_demo > "C:\bkdb\respaldo.dump" | Realiza el respaldo en un formato específico (p. ej., personalizado), como el usuario **postgres**. |
| -v | pg_dump-U postgres-v mi_financiera_demo > "C:\bkdb\respaldo.sql" | Genera un respaldo detallado, ejecutado como el usuario **postgres**. |
| -Z <nivel_compresión> | pg_dump-U postgres-Z 9 mi_financiera_demo > "C:\bkdb\respaldo.sql.gz" | Crea un respaldo comprimido, ejecutado como el usuario **postgres**. |
| -h <servidor> | pg_dump-U postgres-h localhost mi_financiera_demo > "C:\bkdb\respaldo.sql" | Especifica el servidor, ejecutado como el usuario **postgres**. |
| -p <puerto> | pg_dump-U postgres-p 5432 mi_financiera_demo > "C:\bkdb\respaldo.sql" | Conecta al puerto especificado, como el usuario **postgres**. |
| --no-owner | pg_dump-U postgres--no-owner mi_financiera_demo > "C:\bkdb\respaldo.sql" | Excluye información del propietario, ejecutado como el usuario **postgres**. |
| --no-acl | pg_dump-U postgres--no-acl mi_financiera_demo > "C:\bkdb\respaldo.sql" | Omite información de ACL, ejecutado como el usuario **postgres**. |
| -t <tabla> | pg_dump-U postgres-t clientes mi_financiera_demo > "C:\bkdb\respaldo_clientes.sql" | Respalda una tabla específica, como el usuario **postgres**. |
| --inserts | pg_dump-U postgres--inserts mi_financiera_demo > "C:\bkdb\respaldo.sql" | Utiliza comandos INSERT, ejecutado como el usuario **postgres**. |
| --column-inserts | pg_dump-U postgres--column-inserts mi_financiera_demo > "C:\bkdb\respaldo.sql" | Genera INSERT con nombres de columnas, como el usuario **postgres**. |
| --data-only | pg_dump-U postgres--data-only mi_financiera_demo > "C:\bkdb\solo_datos.sql" | Respalda solo los datos, ejecutado como el usuario **postgres**. |
| --schema-only | pg_dump-U postgres--schema-only mi_financiera_demo > "C:\bkdb\solo_esquema.sql" | Respalda solo el esquema, ejecutado como el usuario **postgres**. |

**Notas sobre Formatos de Respaldo en pg_dump**

1. **Formato de Texto Plano (-F p o --format=p)**:
    - **Descripción**: El formato de texto plano produce un archivo SQL que contiene los comandos SQL necesarios para reconstruir la base de datos al estado en que estaba en el momento del respaldo.
    - **Uso Común**: Ideal para bases de datos más pequeñas o para casos donde se desea revisar o editar el archivo de respaldo antes de restaurarlo.
2. **Formato Personalizado (-F c o --format=c)**:
    - **Descripción**: Este formato produce un archivo de respaldo en un formato personalizado de PostgreSQL. No es legible por humanos, pero es altamente flexible y eficiente.
    - **Uso Común**: Recomendado para respaldos de bases de datos más grandes, ya que facilita respaldos incrementales y permite restaurar elementos específicos de la base de datos.
    - **Restauración**: Debe usarse **pg_restore** para restaurar desde este formato.
3. **Formato Directorio (-F d o --format=d)**:
    - **Descripción**: En este formato, pg_dump crea un directorio con un archivo por cada tabla y blob. Este formato es útil para bases de datos grandes ya que permite la restauración paralela y la manipulación de archivos individuales.
    - **Uso Común**: Ideal cuando se necesita realizar respaldos en paralelo o cuando se trabaja con bases de datos de gran tamaño.
    - **Restauración**: Debe usarse **pg_restore** para restaurar desde este formato.
4. **Formato Tar (-F t o --format=t)**:
    - **Descripción**: Genera un archivo de respaldo en formato tar. Es similar al formato personalizado, pero con algunas limitaciones, como un tamaño máximo de archivo.
    - **Uso Común**: Puede ser útil para bases de datos más pequeñas, donde se desea un equilibrio entre la legibilidad del archivo (como con los archivos de texto) y la eficiencia del formato personalizado.

**Nota sobre ACL en Respaldos de PostgreSQL**

- Cuando se utiliza pg_dump para crear un respaldo, el comando --no-acl es útil si se desea omitir estas listas de control de acceso. Esto puede ser deseable al mover datos entre diferentes entornos donde los usuarios o roles pueden no coincidir, o al simplificar la administración de permisos en el entorno de destino.

**Tabla de Comandos de Restauración para PostgreSQL (pg_restore)**

| Comando | Ejemplo | Descripción |
|---|---|---|
| **pg_restore -U <usuario> -d <nombre_bd> <archivo>** | **pg_restore -U postgres -d mi_financiera_demo "C:\bkdb\respaldo.dump"** | Restaura la base de datos **mi_financiera_demo** desde un archivo de respaldo. |
| **-c o --clean** | **pg_restore -U postgres -d mi_financiera_demo -c "C:\bkdb\respaldo.dump"** | Limpia (borra) los objetos de la base de datos antes de la restauración. |
| **-C o --create** | **pg_restore -U postgres -C -d mi_financiera_demo "C:\bkdb\respaldo.dump"** | Crea la base de datos antes de realizar la restauración. |
| **-F <formato>** | **pg_restore -U postgres -d mi_financiera_demo -F c "C:\bkdb\respaldo.dump"** | Especifica el formato del archivo de respaldo (c: personalizado, d: directorio, t: tar). |
| **-v o --verbose** | **pg_restore -U postgres -d mi_financiera_demo -v "C:\bkdb\respaldo.dump"** | Muestra información detallada durante la restauración. |
| **-l o --list** | **pg_restore -U postgres -l "C:\bkdb\respaldo.dump"** | Lista el contenido del archivo de respaldo sin restaurarlo. |
| **-P <nombre_funcion>(<argumentos>)** | **pg_restore -U postgres -d mi_financiera_demo -P mi_funcion(int) "C:\bkdb\respaldo.dump"** | Restaura solo la función especificada. |
| **-t <tabla> o --table=<tabla>** | **pg_restore -U postgres -d mi_financiera_demo -t clientes "C:\bkdb\respaldo.dump"** | Restaura solo la tabla especificada. |
| **-n <esquema> o --schema=<esquema>** | **pg_restore -U postgres -d mi_financiera_demo -n mi_esquema "C:\bkdb\respaldo.dump"** | Restaura solo el esquema especificado. |
| **--disable-triggers** | **pg_restore -U postgres -d mi_financiera_demo --disable-triggers "C:\bkdb\respaldo.dump"** | Desactiva los disparadores (triggers) durante la restauración. |
| **-a o --data-only** | **pg_restore -U postgres -d mi_financiera_demo -a "C:\bkdb\respaldo.dump"** | Restaura solo los datos, sin esquema. |
| **-s o --schema-only** | **pg_restore -U postgres -d mi_financiera_demo -s "C:\bkdb\respaldo.dump"** | Restaura solo el esquema, sin datos. |

**Notas sobre Restauración con pg_restore**

- **Usuario**: Es importante especificar el usuario (**-U**) que tiene permisos para realizar la restauración.

- **Base de Datos**: Se debe especificar la base de datos objetivo (**-d**) para la restauración. Si se utiliza la opción **-C**, la base de datos se creará si no existe.
- **Formato de Respaldo**: Es crucial conocer el formato del archivo de respaldo, ya que **pg_restore** puede manejar formatos como personalizado, directorio y tar.
- **Restauración Selectiva**: **pg_restore** permite restaurar elementos específicos de un respaldo (como tablas o esquemas individuales), lo que es útil para bases de datos grandes o para restaurar solo partes específicas de la base de datos.
- **Desactivación de Triggers**: La opción **--disable-triggers** puede ser útil al restaurar datos en una base de datos con triggers configurados que podrían interferir con la inserción de datos.

# Consideraciones para respaldos incrementales

| Categoría | Comando o Herramienta | Ejemplo de Uso | Descripción | Resultado Esperado | Entorno de Ejecución |
|---|---|---|---|---|---|
| **Respaldo Completo Inicial** | **pg_dump** | **pg_dump -U admin mi_financiera_demo > backup_full.sql** | Realiza un respaldo completo de la base de datos **mi_financiera_demo**. | Archivo **backup_full.sql** con el respaldo completo. | Windows CMD/Linux Shell |
| **Respaldo Incremental** | Herramienta de terceros (Barman) | Configuración de Barman para **mi_financiera_demo** | Configura Barman para realizar respaldos incrementales de la base de datos. | Respaldos incrementales periódicos de la base de datos. | Linux Shell |
| **Restauración Completa** | **psql** y Herramienta de terceros | Restauración usando **psql** y Barman | Restaura la base de datos a partir del último respaldo completo y los respaldos incrementales. | Base de datos **mi_financiera_demo** restaurada a su estado más reciente. | Windows CMD/Linux Shell |

**Pasos Adicionales y Consideraciones:**

- **Configuración Inicial**: Configura Barman o PgBackRest con las credenciales y parámetros adecuados para tu base de datos.
- **Programación de Respaldos Incrementales**: Configura una programación regular (por ejemplo, diaria) para los respaldos incrementales.
- **Monitorización y Verificación**: Monitoriza regularmente la integridad y el éxito de los respaldos.
- **Restauración de Pruebas**: Realiza restauraciones de prueba para asegurar que el proceso y los datos recuperados son válidos.



# Instrucciones Shell personalizado para Postgresql

**MTI JESÚS E. ROMERO MORENO**

**DIVISIÓN DE TECNOLOGÍAS DE LA INFORMACIÓN Y COMUNICACIÓN**

# Shell Personalizado para Administración de Base de Datos en PostgreSQL

**Introducción**

En la administración de bases de datos, la automatización es una herramienta esencial para garantizar eficiencia, consistencia y ahorro de tiempo. Hoy abordaremos una tarea de gran relevancia: la personalización y mejora de un shell diseñado para administrar bases de datos en **PostgreSQL**, con un enfoque especial en automatizar procesos críticos como **respaldo** y **restauración** de bases de datos. Para fines prácticos, trabajaremos con la base de datos **mi_financiera_demo**, que ya has utilizado en actividades anteriores. Esta práctica no solo te permitirá optimizar los procesos, sino también reforzar tus habilidades en **scripting**, lo que te hará más versátil en el manejo de entornos de administración de bases de datos.

**Objetivo de la Práctica**

Al finalizar esta actividad, se espera que hayas logrado los siguientes resultados:

- **Personalización del shell** para automatizar las tareas clave en la administración de bases de datos PostgreSQL.
- Implementación eficaz de comandos de **respaldo** y **restauración**, garantizando que la base de datos *mi_financiera_demo* esté bien gestionada.
- Aplicación de **mejoras adicionales** al shell, como la optimización de comandos, manejo eficiente de errores, y compresión de respaldos.
- **Documentación detallada** de los resultados obtenidos, con especial énfasis en los retos técnicos enfrentados y las soluciones aplicadas durante el proceso.

**Lista de Comandos Proporcionados**

A lo largo de la práctica, trabajarás con una lista de comandos clave que te servirán como base para personalizar el shell. Entre los más relevantes se incluyen **pg_dump** y **pg_restore**, fundamentales para el respaldo y la restauración. Además, se proporcionarán opciones adicionales, como la compresión de respaldos y el respaldo de tablas específicas, que deberás integrar en el script. Se espera que realices investigaciones complementarias para optimizar el shell de acuerdo con las necesidades de la base de datos que administras.

**Requisitos de la Práctica**

1. **Implementación en PostgreSQL**: El entorno principal para la práctica será PostgreSQL, trabajando específicamente con la base de datos **mi_financiera_demo**. Puedes realizar las tareas en una **máquina virtual (MV)** o en tu máquina de escritorio personal, pero debes asegurarte de que el shell funcione sin contratiempos en ambos

entornos y que los cambios realizados se guarden adecuadamente, **no olvidar configurar las variables de entorno, y path.**

2. **Mejoras y Personalización del Shell**: El enfoque central será la automatización de procesos cruciales como el **respaldo**, la **restauración** y gestión de usuarios. Es clave que optimices el script para que sea adaptable y eficiente en diferentes entornos, documentando cualquier reto que hayas superado durante el proceso de personalización.
3. **Investigaciones Adicionales**: Como parte de la práctica, se espera que lleves a cabo investigaciones para aplicar **mejoras al script y nuevos comandos**.

**Entregable: Video Explicativo**

El entregable principal de esta práctica será un video explicativo que deberás subir a YouTube como **video privado**. Este debe cumplir con los siguientes puntos:

- **Identificación personal** al inicio: Menciona tu nombre completo, grupo, asignatura, y el nombre del profesor encargado. Incluye una leyenda en el video indicando que el contenido es de **autoría propia**.
- **Explicación detallada del código**: No basta con mostrar el resultado final del shell; es necesario que expliques en detalle cada uno de los cambios realizados en el código y cómo estos contribuyen a la automatización de las tareas. Recuerda abordar los problemas que encontraste y cómo los resolviste.
- **Documentación de los resultados**: Proporciona ejemplos prácticos de cómo el shell mejoró en funcionalidad y automatización. Si enfrentaste algún desafío técnico, explícalo y detalla cómo lo superaste.

**Lo que se espera de ti**

Para cumplir con los objetivos de la práctica, deberás:

1. **Mejorar y automatizar el shell**: Personaliza el script proporcionado para que automatice las tareas clave de administración de bases de datos en PostgreSQL, como respaldo, restauración y compresión de datos.
2. **Documentar avances y retos**: Durante la explicación en video, menciona los cambios que realizaste en el código y cómo estos contribuyen a mejorar la eficiencia del shell. Asegúrate de explicar los desafíos enfrentados y las soluciones que implementaste.
3. **Asegurar la accesibilidad del video**: Sube tu video a YouTube como **privado** y comparte el enlace en Google Classroom para su evaluación.

**Herramientas: PostgreSQL y Shell**

En esta práctica, las principales herramientas serán PostgreSQL para gestionar la base de datos y un **shell script** para automatizar las tareas. Familiarízate con los comandos proporcionados como **pg_dump** y **pg_restore**, y realiza las investigaciones necesarias para optimizar tu código de acuerdo con el entorno en que lo estés ejecutando.

**Reflexión Final**

Este ejercicio es más que una simple tarea técnica. Es un paso hacia la consolidación de habilidades que te permitirán optimizar y automatizar procesos dentro de un entorno de bases de datos. A lo largo de esta práctica, no solo desarrollarás un shell más eficiente, sino que también demostrarás tu capacidad para enfrentar y superar retos técnicos, una habilidad crucial en el mundo profesional.

*"Quienes se dedican a administrar bases de datos son los arquitectos silenciosos del mundo digital; su trabajo invisible sostiene la estructura de la información, garantizando que todo fluya con precisión y seguridad."*

**CIERRE DE TEMA**

# AUTOMATIZACIÓN DE TAREAS MEDIANTE USO DELL SHELL DE POSTGRESQL

Por Martínez de la Cruz José Julián

Fecha: 28/06/2025

Palabras clave: Automatización, Shell PostgreSQL, Privilegios, Exportación de datos, Importación de datos, Automatización de tareas administrativas en PostgeSQL.

## 1. Objetivo del Tema

El propósito de esta práctica es lograr un Shell personalizado que permita la automatización de tareas clave en la administración de Bases de Datos de PostgreSQL mediante la implementación de scripts que ejecuten comandos tanto de respaldo, restauración, exportación, importación y registro de usuarios con privilegios para mantener la base de datos de Mi Financiera Demo perfectamente gestionada.

## 2. Resumen

Para mejorar la administración de la base de datos de Mi Financiera Demo, se realizó un Shell con scripts personalizados que permiten automatizar tareas clave para la misma. Todo esto se organiza y se ejecuta por medio de un menú principal el en el Shell de Windows que permite al usuario acceder fácilmente a las distintas funciones sin interactuar directamente con PostgreSQL con el objetivo de reducir errores manuales y facilitando la gestión.

Este menú general, llamado *main_menú*, actúa como un punto de control desde donde se puede mandar a llamar a cada uno de los diferentes scripts los cuales permiten:

1. **Registrar Usuarios con Privilegios:** Este script permite registrar nuevos usuarios en PostgreSQL y asignarles privilegios (crud) en una base de datos y tabla determinada según sus necesidades, lo que permite tener controlada y asegurada la base de datos.
2. **Modificar Privilegios:** Permite modificar los privilegios ya asignados a un usuario, ya sea para restringir o ampliar lo que puede hacer en la base de datos y tabla en específico. De este modo, se mantiene un control total sobre lo que se puede hacer y quién lo puede hacer dentro del sistema.
3. **Ver Usuarios:** Este script lista todos los usuarios registrados en PostgreSQL junto con sus respectivos privilegios, permitiendo una visualización rápida del estado actual de los accesos.
4. **Exportación de Datos:** En cuanto a la gestión de datos, se realizó un script que permita exportar todos los datos de una tabla en múltiples formatos: TEXT; CSV, JSON y XML. Estos formatos podrían cubrir bastantes necesidades como puede ser migrar de Modelo de relacional a no relacional o permitir realizar reportes.

5. **Importación de Datos:** Se integró otro script que permita importar los datos desde los archivos TEXT y CSV del sistema directamente hacia la base de datos, facilitando la carga de datos.

Finalmente, cada uno de estos scripts están diseñados para que se puedan operar de manera intuitiva, autónoma y facilitándole la interacción con los mismos al usuario mediante opciones numéricas. También se procura que se acceda a los scripts desde el menú general para permitir una operación más segura y organizada para administrar de manera eficiente las bases de datos de PostgreSQL.

**3. Conclusión**

La práctica permite cumplir correctamente el objetivo ya que con el uso de este Shell personalizado se puede administrar estructurada y organizadamente de forma más eficiente la base de datos Mi Financiera Demo. Al contar con un menú principal se simplifica demasiado el trabajo y reduce el riesgo de errores que pueden ser causados por personal que no cuenta con la suficiente experiencia pues todo está pensado para que el administrador no tenga que ejecutar comandos sueltos o recordar la sintaxis de los comandos de PostgreSQL, de esta manera de cumple todo lo esencial para mantener la base de gestionada correctamente.

**4. Referencias**


- Presentación: *Tabla de referencia de comandos*, MTI Jesús E. Romero, UTTEC.
- PostgreSQL (s.f). *SQL Syntax Copy.* PostgreSQL Documentation.


**5. Reflexiones Finales**

1. **Automatizar ahorra tiempo y evita errores:** Tener scripts listos para importar/exportar o dar permisos específicos a usuarios es mucho más eficiente que hacer todo a mano o mediante PgAdmin.
2. **El menú completo hace la diferencia:** Tener un solo punto de entrada que organiza todas las tareas da claridad, estructura y facilita al flujo de trabajo.
3. **Los formatos de exportación/importación amplían la utilidad:** Permitir trabajar con CSV, JSON, XML y TXT facilita migrar datos entre sistemas o documentar de mejor manera la información.