# Dockerfile Avanzado

**Arturo Silvelo**

Try New Roads

# ¿Qué es un backup?

Un backup (o copia de seguridad) es una copia de los datos originales que se realiza para poder recuperarlos en caso de pérdida, corrupción, borrado accidental o desastre. En el contexto de Docker, los backups suelen centrarse en los volúmenes y bases de datos, ya que los contenedores son efímeros y fácilmente reemplazables. Un buen sistema de backups permite restaurar la información y minimizar el impacto de cualquier incidente.

# 1. Backups incrementales, diferenciales y completos

- **Un backup completo** copia todos los datos cada vez.
- **Un backup incremental** solo copia los cambios desde el último backup (completo o incremental).
- **Un backup diferencial** copia los cambios desde el último backup completo.

**Ventajas:**

- **Ahorro de espacio y tiempo:** Los backups incrementales y diferenciales solo copian los archivos que han cambiado desde el último backup relevante. Esto reduce el espacio ocupado y acelera el proceso, ya que no se duplican datos innecesarios.
- **Restauración a diferentes puntos en el tiempo:** Al tener una secuencia de backups (completo + incrementales o diferenciales), puedes restaurar el sistema no solo al último estado, sino también a cualquier punto intermedio para el que tengas un backup. Esto es útil si necesitas recuperar datos de un día específico antes de un error o corrupción.

**Herramientas:**

- `rsync`: ideal para backups incrementales, ya que solo copia los archivos nuevos o modificados desde el último backup. Muy útil para sincronizar directorios de forma eficiente.
- `restic`, `borg`, `duplicity`: soluciones avanzadas que permiten realizar backups completos, incrementales y diferenciales, con deduplicación, cifrado y gestión de versiones. Son recomendadas para estrategias de backup más complejas y seguras.

## Ejemplo: Backup completo

Copia todo el contenido del volumen, sin importar si ha cambiado o no.

Ideal para hacer un backup completo semanal.

```
docker run --rm \
  -v datos_app:/data \
  -v $(pwd):/backup \
  alpine \
  tar czvf /backup/backup-completo-$(date +%F).tar.gz -C /data .
```

## Ejemplo: Backup incremental con rsync

Solo copia los archivos nuevos o modificados desde el último backup.

```
docker run --rm \
  -v datos_app:/data \
  -v $(pwd)/backup:/backup \
  alpine \
  sh -c "apk add --no-cache rsync && rsync -av --delete /data/ /backup/"
```

## Ejemplo: Backup diferencial con tar

Copia solo los archivos modificados desde el último backup completo.

```
docker run --rm \
  -v datos_app:/data \
  -v $(pwd):/backup \
  alpine \
  sh -c 'find /data -type f -newer /backup/backup-completo.tar.gz -print0 | xargs -0 tar rvf /backup/backup-diferencial.tar'
```

**Consideraciones:**

- Los backups incrementales requieren todos los backups anteriores para una restauración completa.
- Los diferenciales requieren solo el último completo y el último diferencial.
- Es importante planificar la rotación y limpieza de backups antiguos.

# 2. Backups consistentes de bases de datos

**¿Por qué no es suficiente copiar archivos de bases de datos en caliente?**

- Las bases de datos suelen mantener archivos abiertos y estructuras en memoria.
- Copiar los archivos directamente mientras la base está en uso puede provocar corrupción o backups incompletos.
- Es fundamental garantizar la consistencia de los datos en el momento del backup.

**Uso de herramientas nativas:**

- Cada motor de base de datos tiene su propia herramienta para realizar backups consistentes:
  - `mysqldump` para MySQL/MariaDB
  - `pg_dump` para PostgreSQL
  - `mongodump` para MongoDB
  - Otras: `sqlite3 .dump`, `redis-cli --rdb`, etc.

## Ejemplo: Backup y restauración de PostgreSQL desde un contenedor

*Backup:*

```
docker exec mi_postgres pg_dump -U postgres mi_db > backup.sql
```

*Restauración:*

```
cat backup.sql | docker exec -i mi_postgres psql -U postgres mi_db
```

# 3. Automatización de backups

## Ventajas de automatizar

- Reduce errores humanos y olvidos.
- Permite mantener una política de backups regular y predecible.
- Facilita la integración con sistemas de monitorización y alertas.

## Uso de cron en un contenedor dedicado

- Puedes crear un contenedor que ejecute backups periódicamente usando cron.
- Así, el proceso de backup queda aislado y es fácilmente portable entre entornos.

## Ejemplo de Dockerfile para backup automatizado:

```
FROM alpine
RUN apk add --no-cache bash tar curl
COPY backup.sh /backup.sh
RUN chmod +x /backup.sh
CMD ["/backup.sh"]
```

## Ejemplo de script backup.sh:

```
#!/bin/sh
tar czvf /backup/backup-$(date +%F).tar.gz -C /data .
```

- Monta el volumen de datos en `/data` y el de backups en `/backup` al ejecutar el contenedor.

# Ejecución periódica con supercronic

- Usa un contenedor tipo **supercronic** para ejecutar el script de backup periódicamente con cron.

```
services:
  backup:
    image: alpine
    command: supercronic /etc/crontab
    volumes:
      - datos_app:/data
      - ./backups:/backup
      - ./backup.sh:/backup.sh
      - ./crontab:/etc/crontab
```

Archivo `crontab` de ejemplo:

```
0 2 * * * /backup.sh >> /backup/backup.log 2>&1
```

## Ejemplo de logs y alertas ante fallo de backup

- Redirige la salida del script a un archivo de log para auditoría.
- Puedes añadir notificación por email o webhook en caso de error:

```
#!/bin/sh
set -e
LOGFILE="/backup/backup.log"
if tar czvf /backup/backup-$(date +%F).tar.gz -C /data . >> $LOGFILE 2>&1; then
  echo "Backup OK: $(date)" >> $LOGFILE
else
  echo "Backup FAILED: $(date)" | tee -a $LOGFILE
  # Ejemplo de alerta por webhook (adaptar a tu sistema)
  curl -X POST -H 'Content-type: application/json' \
    --data '{"text":"Backup fallido en '$(date)'"}' \
    https://hooks.slack.com/services/TU_WEBHOOK
fi
```

# 4. Verificación y restauración de backups

## ¿Por qué verificar los backups?

- Un backup no verificado puede estar corrupto o incompleto.
- La única forma de asegurar que un backup sirve es restaurándolo y comprobando los datos.

## Estrategias de verificación

- **Verificación de integridad:** Usa checksums (`sha256sum`, `md5sum`) para comprobar que los archivos no se han corrompido.
- **Pruebas de restauración:** Restaura backups en un entorno de pruebas y valida que los servicios y datos funcionan correctamente.
- **Automatización:** Programa verificaciones periódicas y registra los resultados.

## Ejemplo: Verificar integridad de un backup

```
sha256sum backup-completo-2025-07-29.tar.gz > backup-completo-2025-07-29.tar.gz.sha256
sha256sum -c backup-completo-2025-07-29.tar.gz.sha256
```

## Ejemplo: Restaurar un volumen Docker desde un backup

```
docker volume create datos_app_restaurado
docker run --rm \
  -v datos_app_restaurado:/data \
  -v $(pwd):/backup \
  alpine \
  tar xzvf /backup/backup-completo-2025-07-29.tar.gz -C /data
```

## Ejemplo: Restaurar una base de datos PostgreSQL

```
cat pg_backup-2025-07-29.sql | docker exec -i mi_postgres psql -U postgres mi_db
```