Utilidades

Un paquete de utulidades para la administracion de sistemas, con foco en backups de diversos tipos y sitauaciones.

- Sistema de backup pull para Proxmox VE

Sistema de backup pull para Proxmox VE

Sistema de backup basado en snapshots LVM para maquinas virtuales en Proxmox VE. Utiliza una arquitectura **pull** donde el servidor de storage inicia las conexiones SSH hacia Proxmox y tira de los datos. El servidor Proxmox nunca tiene acceso de escritura al storage.

Motivacion

El modelo tradicional **push** (Proxmox envia al storage) tiene un problema de seguridad critico: si el servidor Proxmox es comprometido, el atacante tiene credenciales SSH con acceso de escritura al storage y puede destruir todos los backups.

Con el modelo **pull**:

- **Proxmox no conoce las credenciales del storage**
- **Proxmox no puede escribir ni borrar en el storage**
- **Solo el storage decide cuando y que se respalda**
- Un Proxmox comprometido no puede destruir los backups existentes

Arquitectura

A partir de v2.0.0 el flujo es **por VM** en vez de por disco individual. Para VMs con multiples discos, todos los snapshots se crean atomicamente antes de transferir cualquier dato, garantizando consistencia temporal. Desde v2.1.0, un lock distribuido permite que multiples storages respalden el mismo Proxmox sin colisiones.

```
Storage Server                          Proxmox Server

     |                                        |
     |-- ssh "list" ------------------------->|  (inventario de VMs)
     |<-- CSV: vmid,status,vg,lv,size -------|
     |                                        |
     |  Para cada VM:                         |
     |-- ssh "snapshot-vm <vmid> <pct>        |
     |        <caller_id> <lock_ttl>" ------>|  (lock distribuido + snapshots atomicos)
```

```
|<-- manifest: snap_name,vg,lv,size ----|
|                                       |
|  Para cada disco en manifest:         |
|-- ssh "stream-disk <vg> <snap>" ----->|  (stream comprimido dd|pigz)
|<------------ compressed stream -------|
|-- verifica PIPESTATUS + tamano        |
|                                       |
|  Si TODOS los discos OK:              |
|-- mv atomico incoming/ -> snapshots/  |
|-- prune copias antiguas               |
|                                       |
|  Si ALGUN disco falla:                |
|-- elimina TODO incoming/ de esta VM   |
|-- backup previo en snapshots/ intacto |
|                                       |
|  SIEMPRE:                             |
|-- ssh "cleanup-vm <vmid>              |
|        <caller_id>" ----------------->|  (elimina snapshots + libera lock)
```

Consistencia multi-disco (v2.0.0)

El problema que resuelve v2.0.0: en versiones anteriores, los snapshots se creaban uno a uno (snapshot disk0 → transfer disk0 → cleanup disk0 → snapshot disk1 → transfer disk1). Para VMs con multiples discos, los snapshots capturaban diferentes instantes, produciendo backups inconsistentes.

La solucion:

1. `snapshot-vm` crea TODOS los snapshots en sucesion rapida (milisegundos entre ellos)
2. Si algun snapshot falla, rollback automatico de los ya creados
3. Los streams se transfieren secuencialmente pero desde snapshots del mismo instante
4. `cleanup-vm` elimina todos los snapshots al final

Semantica all-or-nothing

- Un backup parcial de VM es inutil (disk0 de hoy y disk1 de ayer)
- `snapshots/` siempre contiene conjuntos completos
- Si ANY disco falla: se borra todo el incoming de esa VM, el backup anterior queda intacto
- Prune solo ejecuta tras exito total

Validacion de integridad (v2.1.2)

La compresion se realiza **on-the-fly** en el pipeline SSH: `dd if=/dev/vg/snap bs=16M | pigz -c -p4 -3`. Los errores del pipeline se detectan via **PIPESTATUS** inmediatamente. No se ejecuta verificacion post-transferencia (`pigz -t`) porque:

- Re-leer ficheros de 93GB+ desde disco mecanico (30-40 MB/s) tarda 40-50 minutos
- Es redundante: si el pipeline falla, PIPESTATUS lo captura
- La verificacion de integridad de backups pertenece a auditorias periodicas, no al flujo diario

Lock distribuido (v2.1.0)

Cuando multiples storages respaldan el mismo Proxmox, un lock distribuido previene colisiones:

- Ficheros lock en `/run/pull-backup/vm-{vmid}.lock` en Proxmox
- Adquisicion atomica via `ln` (hard link, falla con EEXIST si ya existe)
- Contenido: `caller_id`, timestamp, TTL
- VM bloqueada por otro storage: skip limpio (exit 80, `VM_LOCKED`)
- Locks expirados (mas antiguos que `lock_ttl`) se limpian automaticamente
- Solo el caller que creo el lock puede liberarlo

Configuracion: `caller_id` (unico por storage) y `lock_ttl` (segundos) en `pull-backup.cfg`. Sin `caller_id`, el lock se desactiva (modo single-storage).

Componentes

Componente	Version	Ubicacion	Funcion
`proxmox-backup-helper.sh`	2.1.1	Proxmox: `~/utilidades/proxmox-backup/pull/` (repo clonado)	Manejador SSH restringido con 13 comandos, lock distribuido
`proxmox-pull-backup.sh`	2.1.2	Storage: `~/utilidades/proxmox-backup/pull/` (repo clonado)	Orquestador de backups (flujo por VM, lock distribuido)
`proxmox-pull-restore.py`	1.0.0	Storage: mismo directorio	Herramienta de restauracion (Python 3.6+)
`backup-access-shell.sh`	1.0.0	Storage: mismo directorio	Shell de solo lectura para restore
`install-pull-backup.sh`	1.0.0	Storage: mismo directorio	Instalador automatizado

IMPORTANTE: Todos los scripts se ejecutan **desde el repo clonado** (`~/utilidades/proxmox-backup/pull/`). Nunca copiar a `/usr/local/bin/` ni a otra ruta del sistema. Un `git pull` actualiza todo.

Requisitos previos

- **Proxmox**: Version 7.x, 8.x o 9.x con almacenamiento LVM
- **Storage**: Servidor Linux con espacio suficiente
- **Ambos servidores**: `pigz` instalado (`apt-get install pigz`)
- **Storage**: Python 3.6+ (solo stdlib, sin dependencias externas)
- **Red**: Acceso SSH desde storage hacia Proxmox

Instalacion

Paso 1: Clonar el repositorio en ambos servidores

En Proxmox (como root):

```
cd ~
git clone <url-del-repo> utilidades
```

En storage (como usuario de backup, NO root):

```
cd ~
git clone <url-del-repo> utilidades
```

Paso 2: Instalar pigz en ambos servidores

```
apt-get install -y pigz
```

Paso 3: Generar clave SSH y configurar acceso

En el storage:

```
cd ~/utilidades/proxmox-backup/pull
mkdir -p keys
ssh-keygen -t ed25519 -f keys/id_pull_backup -N "" -C "pull-backup@$(hostname)"
```

Copiar la clave publica al Proxmox con restriccion `command=`:

```
# En /root/.ssh/authorized_keys del Proxmox:
command="/root/utilidades/proxmox-backup/pull/proxmox-backup-helper.sh",no-port-forwarding,no-X11-forwarding,no-agent-forwarding,no-pty ssh-ed25519 AAAA... pull-backup@mystor01
```

Paso 4: Crear la configuracion

```
cd ~/utilidades/proxmox-backup/pull
cp config/pull-backup.cfg.example config/pull-backup.cfg
cp config/proxmox_hosts.cfg.example hosts/proxmox_hosts.cfg
```

Editar `config/pull-backup.cfg`:

```
# Directorios
incoming_dir="/srv/storage/backups_kvm/incoming"
final_dir="/srv/storage/backups_kvm/snapshots"

# Retencion
numcopias=3

# Snapshots
snapshot_pct=15          # Porcentaje del LV para snapshot
default_vm_scope="all"   # "all" = descubre todas las VMs; "none" = requiere filtro

# Rendimiento
min_speed_mbs=150    # MB/s minimos esperados (para timeout dinamico)
min_free_gb=500      # GB libres requeridos (ademas del tamano estimado)
ssh_timeout=30       # Timeout SSH en segundos

# Lock distribuido (v2.1.0+, obligatorio si hay multiples storages)
caller_id="mystor01"    # Identificador unico de este storage
lock_ttl=7200            # TTL del lock en segundos (2h)

# Permisos de ficheros
file_owner="backupuser:backupuser"   # chown tras escribir
```

```
# Notificaciones (solo en errores)
correo="admin@example.com"
```

Editar `hosts/proxmox_hosts.cfg`:

```
# name,host,port,key_file,disks_file
myprox01,10.0.0.1,22,keys/id_pull_backup,
```

El campo `disks_file` es opcional desde v2.0.0. Con `default_vm_scope="all"`, el orquestador descubre automaticamente todas las VMs activas.

Paso 5: Crear directorios de datos

```
mkdir -p /srv/storage/backups_kvm/incoming
mkdir -p /srv/storage/backups_kvm/snapshots
mkdir -p ~/utilidades/proxmox-backup/pull/logs
```

Paso 6: Verificar conectividad

```
cd ~/utilidades/proxmox-backup/pull
./proxmox-pull-backup.sh --host myprox01 --verbose --dry-run
```

Paso 7: Configurar cron

```
# Backup diario a las 02:00 UTC
crontab -e
0 2 * * * ~/utilidades/proxmox-backup/pull/proxmox-pull-backup.sh >> ~/utilidades/proxmox-backup/pull/logs/cron.log 2>&1
```

IMPORTANTE con multiples storages: escalonar los crons con al menos 15 minutos de diferencia. Aunque el lock distribuido previene colisiones, la separacion temporal reduce contention.

Actualizacion

```
cd ~/utilidades
git pull
```

No requiere reinicio. Los scripts se invocan frescos en cada ejecucion.

Uso diario

Ejecutar backup manual

```
cd ~/utilidades/proxmox-backup/pull

# Dry-run (no ejecuta nada, muestra lo que haria)
./proxmox-pull-backup.sh --host myprox01 --verbose --dry-run

# Backup real de todas las VMs de un host
./proxmox-pull-backup.sh --host myprox01 --verbose

# Backup de una VM especifica
./proxmox-pull-backup.sh --host myprox01 --vm 104 --verbose

# Backup de varias VMs (separadas por coma o repetido)
./proxmox-pull-backup.sh --host myprox01 --vm 104,203 --verbose

# Backup de todos los hosts configurados
./proxmox-pull-backup.sh --verbose
```

Listar backups disponibles

```
python3 proxmox-pull-restore.py --config config/pull-backup.cfg list

# Filtrar por host
python3 proxmox-pull-restore.py --config config/pull-backup.cfg list --host myprox01
```

```
# Filtrar por VM
python3 proxmox-pull-restore.py --config config/pull-backup.cfg list --vm 316
```

Restaurar una VM

Escenario A: la VM existe y esta parada (sobreescribe los discos):

```
python3 proxmox-pull-restore.py --config config/pull-backup.cfg \
  restore --host myprox01 --vm 316 --data-only
```

Escenario B: la VM no existe (crea LV, config y restaura datos):

```
python3 proxmox-pull-restore.py --config config/pull-backup.cfg \
  restore --host myprox01 --vm 316 --from-scratch
```

Restaurar desde una fecha concreta:

```
python3 proxmox-pull-restore.py --config config/pull-backup.cfg \
  restore --host myprox01 --vm 316 --date 2026-02-08
```

Restaurar varias VMs en paralelo:

```
python3 proxmox-pull-restore.py --config config/pull-backup.cfg \
  restore --host myprox01 --vm 316 900 --workers 2
```

IMPORTANTE: el argumento `--config` va ANTES del subcomando (`list`, `restore`).

Modelo de seguridad

Lado Proxmox

El helper (`proxmox-backup-helper.sh`) se ejecuta exclusivamente via la restriccion `command=` en `authorized_keys`. Esto significa que **cualquier** conexion SSH con esa clave ejecuta el helper, ignorando el comando solicitado por el cliente.

El helper valida cada argumento con expresiones regulares estrictas:

- **Volume groups**: solo alfanumericos, guiones y guiones bajos
- **Logical volumes**: solo patron `vm-\d+-disk-\d+`
- **Snapshot names**: solo patron `snap-\d+-disk\d+`
- **VMIDs**: solo numeros entre 100 y 999999
- **Snapshot porcentaje**: solo numeros entre 1 y 100
- **caller_id**: solo alfanumericos, guiones y guiones bajos (v2.1.0)

Comandos no reconocidos se rechazan con error. Todas las operaciones se registran en syslog.

Lado Storage

El storage inicia todas las conexiones. Proxmox nunca se conecta al storage para los backups. El unico acceso inverso (Proxmox hacia storage) es para restore, y se canaliza a traves de `backup-access-shell.sh` que solo permite operaciones de lectura (pigz -dc, ls, test, cat, du) y valida rutas con `readlink -f` contra symlinks maliciosos.

Estructura de ficheros

En el servidor de storage

```
~/utilidades/proxmox-backup/pull/
  proxmox-pull-backup.sh        # Orquestador v2.1.2
  proxmox-pull-restore.py       # Herramienta de restore
  backup-access-shell.sh        # Shell read-only
  install-pull-backup.sh        # Instalador
  config/
    pull-backup.cfg              # Configuracion principal (gitignored)
    pull-backup.cfg.example      # Template (versionado)
  hosts/
    proxmox_hosts.cfg            # Inventario de hosts (gitignored)
    disks_myprox01.txt           # (Opcional) filtro de discos (gitignored)
  keys/
    id_pull_backup               # Clave privada SSH (gitignored)
    id_pull_backup.pub           # Clave publica SSH (gitignored)
  logs/
    pull-backup-YYYY-MM-DD.log   # Log diario (gitignored)
    cron.log                     # Log de cron (gitignored)
```

En el servidor Proxmox

```
~/utilidades/proxmox-backup/pull/
  proxmox-backup-helper.sh      # Helper restringido v2.1.1

/root/.ssh/authorized_keys      # Clave con command=
/run/pull-backup/               # Lock files distribuidos (runtime)
```

Directorio de backups

```
/srv/storage/backups_kvm/
  incoming/                               # Transferencias en curso
  snapshots/
    kvm-104-disk0.2026-02-26-10-00-12.gz   # Disco 0 comprimido
    kvm-104-disk1.2026-02-26-10-00-12.gz   # Disco 1 (mismo timestamp)
    kvm-104.conf.2026-02-26-10-00-12       # Config de la VM
```

Nomenclatura de ficheros

```
kvm-{vmid}-disk{N}.{YYYY-MM-DD-HH-MM-SS}.gz    # Discos
kvm-{vmid}.conf.{YYYY-MM-DD-HH-MM-SS}           # Configuraciones
```

Comandos del helper

El helper v2.1.1 acepta 13 comandos, todos validados con regex:

Comando	Argumentos	Funcion
`list`	(ninguno)	Lista VMs activas con discos LVM
`list-all`	(ninguno)	Lista TODAS las VMs (incluidas paradas)

Comando	Argumentos	Funcion
`list-snapshots`	(ninguno)	Lista snapshots activos (deteccion de huerfanos)
`snapshot-vm`	`vmid porcentaje [force] [caller_id] [lock_ttl]`	Adquiere lock distribuido (si caller_id), crea snapshots atomicos de TODOS los discos. Devuelve manifest
`stream-disk`	`vg snap_name`	Envia stream comprimido (`dd | pigz`) de un snapshot
`cleanup-vm`	`vmid [caller_id]`	Elimina TODOS los snapshots de una VM y libera lock
`check-lock`	`vmid`	Verifica estado del lock distribuido de una VM (v2.1.0)
`snapshot`	`vg lv porcentaje [force]`	(Deprecated) Crea snapshot individual y envia stream
`cleanup`	`vg snap_name`	Elimina un snapshot LVM individual
`config`	`vmid`	Devuelve el fichero de configuracion de la VM
`restore`	`vg lv`	Recibe stream comprimido y lo escribe al disco
`create-disk`	`vg nombre size_gb`	Crea un nuevo volumen logico
`restore-config`	`vmid`	Recibe y escribe la config de la VM

Formato del manifest (snapshot-vm)

```
# snap_name,vg,lv,size_gb
snap-104-disk0,pve,vm-104-disk-0,150
snap-104-disk1,pve,vm-104-disk-1,125
```

Opciones del orquestador

```
proxmox-pull-backup.sh [--config <path>] [--host <name>] [--vm <vmid>[,<vmid>...]] [--dry-run]
[--force] [--verbose]

  --config <path>           Fichero de configuracion (default: ./config/pull-backup.cfg)
  --host <name>             Backup solo este host (default: todos los hosts)
  --vm <vmid>[,<vmid>...]   Backup solo estas VMs (separadas por coma, repetible)
```

```
--dry-run               Muestra lo que haria sin ejecutar
--force                 Auto-limpieza de snapshots huerfanos en Proxmox
--verbose               Salida detallada
```

Parametros de configuracion

Parametro	Default	Descripcion
`incoming_dir`	(obligatorio)	Directorio temporal para transferencias en curso
`final_dir`	(obligatorio)	Directorio final de backups
`numcopias`	3	Copias a retener por disco
`snapshot_pct`	15	Porcentaje del LV para snapshot
`default_vm_scope`	"all"	"all" = descubre todas las VMs; "none" = requiere filtro
`min_speed_mbs`	150	MB/s minimos esperados (para timeout dinamico)
`min_free_gb`	1024	GB libres minimos en storage (ademas del tamano estimado)
`force_orphan_cleanup`	0	Auto-limpiar snapshots huerfanos (1 = activado)
`caller_id`	""	Identificador de este storage (obligatorio para multi-storage)
`lock_ttl`	7200	TTL del lock distribuido en segundos
`file_owner`	""	`user:group` para chown tras escritura (ej: `backupuser:backupuser`)
`correo`	""	Email para notificaciones de error (nunca en exito)
`ssh_timeout`	30	Timeout de conexion SSH en segundos

Funcionalidades avanzadas

Descubrimiento dinamico (v2.0.0)

Con `default_vm_scope="all"` en la config, el orquestador descubre automaticamente todas las VMs activas usando el comando `list` del helper. Los ficheros `disks_*.txt` pasan a ser opcionales (filtro de exclusion, no inventario obligatorio).

Timeout dinamico

El timeout de cada transferencia se calcula automaticamente segun el tamano del disco:

```
timeout = (tamano_disco_gb * 1024) / min_speed_mbs
```

Con un minimo de 300 segundos. Para un disco de 150GB a 150 MB/s el timeout es ~17 minutos.

Deteccion de snapshots huerfanos

Antes de procesar cada host, el orquestador ejecuta `list-snapshots` y compara con los snapshots esperados. Los huerfanos (de ejecuciones previas fallidas) se limpian automaticamente si `force_orphan_cleanup=1`.

Estimacion de tamano

Si existe un backup previo del mismo disco, se usa su tamano como estimacion. Si no, se estima el 50% del tamano raw del disco. Esto se usa para la verificacion de espacio libre.

Reintentos

Si un backup falla, se reintenta una vez tras 10 minutos de espera. Solo se reporta error si el segundo intento tambien falla. Si falla el reintento, la VM completa se marca como fallida (all-or-nothing).

Pruning automatico

Tras completar los backups, se mantienen solo las ultimas `numcopias` copias de cada disco. Las configs asociadas a backups eliminados tambien se limpian.

Rendimiento validado

Produccion (marzo 2026, v2.1.2)

VM	Discos	Comprimido	Tiempo total	Velocidad	Ruta
KVM 104	2 (150+125 GB)	93 GB	14.5 min	106-111 MB/s	avanza01 → stor01avanzait

VM	Discos	Comprimido	Tiempo total	Velocidad	Ruta
KVM 203	2 (85+50 GB)	39 GB	~12 min	56-76 MB/s	avanza01 → stor01avanzait
KVM 100	1 (620 GB)	415 GB	30 min	233 MB/s	indivaprox0 → localstor (local)
KVM 101	1 (260 GB)	114 GB	10 min	187 MB/s	indivaprox0 → localstor (local)

Mejora v2.1.2: La eliminacion de la verificacion `pigz -t` post-transferencia redujo el tiempo total de KVM 104 de ~61 min (v2.0.0) a ~14.5 min — aceleracion de 4x. La integridad de la compresion se valida on-the-fly via PIPESTATUS del pipeline `dd | pigz`.

Staging (febrero 2026)

Operacion	Disco	Tiempo	Velocidad	Resultado
Backup	5GB	11s	6 MB/s	73MB comprimido
Restore data-only	5GB	10s	7 MB/s	OK
Restore from-scratch	5GB	9s	7 MB/s	LV + config + datos
Deteccion huerfanos	--	instantaneo	--	Auto-limpieza
Pruning 4 a 3 copias	--	<1s	--	Retencion exacta

Despliegues actuales

Storage	Host Proxmox	VMs	Cron (UTC)	Version
localstor (stor00)	indivaprox0	KVM 100 (1d, 620GB), KVM 101 (1d, 260GB)	22:00 VM100, 23:15 VM101	v2.1.2
stor01	indivaprox0	todas (dynamic discovery)	02:00	v2.1.2
stor01avanzait	avanza01	KVM 104 (2d), KVM 203 (2d)	10:00	v2.1.2

Troubleshooting

El backup genera un fichero .gz invalido

Causa: Los comandos `lvcreate`/`lvremove` escriben mensajes a stdout que contaminan el stream gzip.

Solucion: Verificar que el helper redirige stdout de lvm a stderr (`>&2`). Actualizar el helper a v2.1.1+.

VM bloqueada por otro storage (exit 80)

Causa: Otro storage esta respaldando esa VM. El lock distribuido (v2.1.0) previene la colision.

Solucion: Es un comportamiento correcto. La VM se salta limpiamente y se respalda en la siguiente ejecucion. Si el lock es huerfano (proceso anterior murio), esperar a que expire el TTL (default 2h) o limpiar manualmente:

```
ssh -i keys/id_pull_backup proxmox "cleanup-vm <vmid> <caller_id>"
```

Snapshot/lock huerfano tras matar un proceso

Causa: Si se mata el proceso del orquestador (kill -9), pueden quedar snapshots y locks huerfanos en Proxmox.

Solucion: Limpiar manualmente:

```
ssh -i keys/id_pull_backup proxmox "cleanup-vm <vmid> <caller_id>"
```

Timeout en backups de discos grandes

Causa: El timeout dinamico depende de `min_speed_mbs` en la config.

Solucion: Reducir `min_speed_mbs` si la red es lenta. Por ejemplo, para 50 MB/s reales:

```
min_speed_mbs=40   # Dejar margen del 20%
```

Snapshot overflow (snap_percent=100)

Causa: El disco recibe mas escrituras de las esperadas durante la transferencia, llenando el snapshot COW.

Solucion: Aumentar `snapshot_pct` en la config (default: 15). Para VMs con alta actividad de I/O, usar 20-25.

El restore dice "pigz: unrecognized format"

Causa: El fichero .gz esta corrupto.

Solucion: Rehacer el backup con el helper v2.1.1+.

Repositorio

El codigo fuente esta en el repositorio `utilidades` bajo `proxmox-backup/pull/`.

Documentacion completa: `proxmox-backup/README.md` y `proxmox-backup/docs/SPEC-pull-backup-architecture.md`.

Historial de versiones

Version	Fecha	Cambio principal
2.1.2	2026-03-02	Eliminada verificacion `pigz -t` post-transferencia (aceleracion 4x en disco mecanico)
2.1.1	2026-03-01	9 bugs corregidos: race condition TOCTOU en lock, `set +e` en pipelines, globals stale, etc.
2.1.0	2026-03-01	Lock distribuido para prevencion de colisiones multi-storage
2.0.0	2026-02-26	Snapshots atomicos multi-disco, flujo per-VM, all-or-nothing, snapshot-vm/stream-disk/cleanup-vm
1.4.0	2026-02-09	Multi-VM filter, dynamic discovery
1.1.0	2026-02-08	Fix stdout pollution en helper, validacion staging
1.0.0	2026-02-08	Primera version funcional

Ultima actualizacion

- **Fecha**: 2026-03-02
- **Version**: 2.1.2
- **Autor**: Abkrim