jetbackup-remote - Manual de uso

Orquestador remoto que serializa trabajos de backup JetBackup5 de múltiples servidores para evitar la saturación del controlador de almacenamiento.

El problema

Cuando varios servidores JetBackup5 lanzan backups simultáneamente hacia un NAS conectado por USB (controlador JMicron JMS567 RAID en Raspberry Pi), el controlador se satura y el rendimiento cae a ~1KB/s. Backups que deberían tardar 1-2 horas pasan a tardar días.

La solución

`jetbackup-remote` ejecuta en la Raspberry Pi y dispara los trabajos de backup uno a uno (cola FIFO), esperando a que termine cada uno antes de lanzar el siguiente. El NAS solo maneja un flujo de escritura a la vez, manteniendo el rendimiento a velocidad máxima.

Características

- **Cola FIFO**: los trabajos se ejecutan uno a uno en todos los servidores
- **SSH + jetbackup5api**: dispara y monitoriza trabajos remotamente vía la CLI de JetBackup5
- **SSH con restricción de comando**: un script gate limita el acceso de la Pi a solo funciones autorizadas de la API
- **Timeouts no abortivos**: los trabajos que exceden el timeout se reportan, nunca se matan (evita corrupción de backup)
- **Notificaciones por email**: alertas en caso de fallo, timeout o finalización
- **Apagado limpio**: gestión de señales SIGTERM/SIGINT
- **Fichero de bloqueo**: impide ejecuciones simultáneas
- **Cero dependencias**: Python stdlib puro, sin paquetes pip

Requisitos

- **Orquestador (Raspberry Pi):** Python 3.9+, acceso SSH a los servidores
- **Servidores JetBackup:** JetBackup5 con `jetbackup5api` CLI disponible

Instalación

1. Clonar el repositorio en la Raspberry Pi

```
git clone https://github.com/AichaDigital/jetbackup-remote.git
cd jetbackup-remote
```

2. Instalar el ejecutable

En sistemas Debian 12+ (Bookworm) con Python gestionado externamente, crear un wrapper:

```
# Copiar el código
sudo cp -r . /opt/jetbackup-remote

# Crear wrapper en /usr/local/bin
sudo tee /usr/local/bin/jetbackup-remote << 'WRAPPER'
#!/bin/bash
exec env PYTHONPATH=/opt/jetbackup-remote/src python3 -m jetbackup_remote "$@"
WRAPPER
sudo chmod +x /usr/local/bin/jetbackup-remote

# Verificar
jetbackup-remote --help
```

Alternativa con pip (si el sistema lo permite):

```
pip3 install .
```

3. Configurar

```
# Crear directorio de configuración
sudo mkdir -p /etc/jetbackup-remote

# Copiar y editar config
sudo cp config.example.json /etc/jetbackup-remote/config.json
sudo nano /etc/jetbackup-remote/config.json
```

Campos a personalizar:

- `ssh_key`: ruta a la clave privada SSH (ej: `/root/.ssh/id_ed25519`)
- `servers`: hostnames y puertos de los servidores JetBackup
- `jobs`: IDs de los backup jobs — obtener con:

  ```
  ssh -p PUERTO root@SERVIDOR "jetbackup5api -F listBackupJobs -O json"
  ```

- `notification`: configuración SMTP si se desean alertas por email

4. Configurar SSH en los servidores

En **cada servidor JetBackup**, instalar el filtro SSH que restringe lo que la Pi puede hacer:

```
# Copiar el gate script al servidor
scp -P 51514 server/jetbackup-ssh-gate.sh root@servidor:/usr/local/bin/
ssh -p 51514 root@servidor "chmod +x /usr/local/bin/jetbackup-ssh-gate.sh"
```

Añadir la clave pública de la Pi al `authorized_keys` del servidor con restricción de comando:

```
# En el servidor, editar /root/.ssh/authorized_keys y añadir:
command="/usr/local/bin/jetbackup-ssh-gate.sh",no-port-forwarding,no-X11-forwarding,no-agent-forwarding,no-pty ssh-ed25519 AAAA... root@raspberrypinas
```

Esto asegura que aunque la clave sea comprometida, solo puede ejecutar funciones autorizadas de la API JetBackup.

5. Verificar conectividad

```
# Test SSH + JetBackup API en todos los servidores
jetbackup-remote test

# Solo un servidor
jetbackup-remote test --server server4

# Validar configuración
jetbackup-remote validate

# Simulación sin ejecutar backups
jetbackup-remote run --dry-run
```

Uso de la CLI

La configuración por defecto se lee de `/etc/jetbackup-remote/config.json`. Se puede especificar otra ruta con `-c`:

```
jetbackup-remote -c /ruta/a/config.json COMANDO
```

run — Ejecutar backups

Ejecuta los trabajos de la cola FIFO, uno a uno:

```
# Todos los trabajos de todos los servidores
jetbackup-remote run

# Solo los trabajos de un servidor
jetbackup-remote run --server server1

# Solo un trabajo específico (por ID)
jetbackup-remote run --job aaaaaaaaaaaaaaaaaaaaaaa
```

```
# Simulación (muestra la cola sin ejecutar nada)
jetbackup-remote run --dry-run
```

status — Ver estado

Consulta el estado de los trabajos de backup en cada servidor:

```
# Todos los servidores
jetbackup-remote status

# Solo un servidor
jetbackup-remote status --server server4

# Salida JSON
jetbackup-remote status --json
```

test — Verificar conectividad

Prueba SSH y la API JetBackup en cada servidor:

```
jetbackup-remote test
jetbackup-remote test --server server3
```

list — Listar trabajos

Lista los trabajos configurados y los disponibles en cada servidor:

```
jetbackup-remote list
jetbackup-remote list --json
```

stop — Detener un trabajo

Detiene un grupo de cola en ejecución:

```
jetbackup-remote stop --server server4 QUEUE_GROUP_ID
```

validate — Validar configuración

Verifica que el fichero de configuración es correcto:

```
jetbackup-remote validate
```

Ejecución manual en tmux

Para ejecutar un ciclo completo (todos los servidores, todos los trabajos) en segundo plano:

```
# Crear sesión tmux
tmux new-session -d -s backup 'jetbackup-remote run'

# Conectar a la sesión para ver progreso
tmux attach -t backup

# Desconectarse sin parar (Ctrl+B, luego D)
```

Ejecución automática con systemd

Copiar los ficheros de servicio:

```
sudo cp systemd/jetbackup-remote.service /etc/systemd/system/
sudo cp systemd/jetbackup-remote.timer /etc/systemd/system/

# Editar la hora si es necesario (por defecto: 02:00 diario)
sudo nano /etc/systemd/system/jetbackup-remote.timer

# Activar
sudo systemctl daemon-reload
sudo systemctl enable --now jetbackup-remote.timer
```

```
# Verificar
systemctl list-timers | grep jetbackup
```

El timer incluye un delay aleatorio de hasta 5 minutos para evitar colisiones.

Flujo de ejecución

```
1. Cargar configuración

2. Adquirir lock file (fcntl.flock) → impedir ejecuciones simultáneas

3. Construir cola FIFO (servidor1/job1, servidor1/job2, servidor2/job1...)

4. Por cada trabajo:
   a. Verificar conectividad SSH (echo test)
   b. Consultar estado del job (getBackupJob)
   c. Si ya está ejecutándose, esperar; si no, disparar (runBackupJobManually)
   d. Sondear cada 30s: comprobar que running=False
   e. Si excede timeout (4h): notificar y pasar al siguiente (NO abortar)
   f. Registrar resultado

5. Enviar resumen por email (si está configurado)

6. Liberar lock file
```

Configuración de ejemplo

```
{
    "ssh_key": "/root/.ssh/id_ed25519",

    "servers": {
        "server1": {
            "host": "server1.example.com",
```

```
        "port": 51514,
        "user": "root"
    },
    "server4": {
        "host": "server4.example.com",
        "port": 51514,
        "user": "root"
    }
},

"jobs": [
    {
        "job_id": "aaaaaaaaaaaaaaaaaaaaaaaa",
        "server": "server1",
        "label": "Accounts Backup",
        "type": "accounts",
        "priority": 0
    },
    {
        "job_id": "22222222222222222222222",
        "server": "server4",
        "label": "Database Backup",
        "type": "database",
        "priority": 0
    }
],

"orchestrator": {
    "poll_interval": 30,
    "job_timeout": 14400,
    "lock_file": "/tmp/jetbackup-remote.lock",
    "log_file": "/var/log/jetbackup-remote.log",
    "log_max_bytes": 10485760,
    "log_backup_count": 5
},

"notification": {
    "enabled": false,
    "smtp_host": "localhost",
```

```
        "smtp_port": 25,
        "from_address": "jetbackup-remote@raspberrypinas",
        "to_addresses": ["admin@example.com"],
        "on_failure": true,
        "on_timeout": true,
        "on_complete": false
    }
}
```

Parámetros del orquestador

Parámetro	Valor por defecto	Descripción
`poll_interval`	30	Segundos entre sondeos del estado del trabajo
`job_timeout`	14400	Timeout por trabajo en segundos (4 horas)
`lock_file`	`/tmp/jetbackup-remote.lock`	Ruta del fichero de bloqueo
`log_file`	`/var/log/jetbackup-remote.log`	Ruta del log
`log_max_bytes`	10485760	Tamaño máximo del log antes de rotar (10 MB)
`log_backup_count`	5	Número de ficheros de log rotados a conservar

Tipos de trabajo

Tipo	Descripción
`accounts`	Backup de cuentas de hosting
`directories`	Backup de directorios específicos
`database`	Backup de bases de datos

Modelo de seguridad

Capas de protección

1. Clave SSH con restricción de comando

La clave SSH de la Pi está restringida en cada servidor mediante `command=` en `authorized_keys`. Incluso si un atacante obtiene la clave privada:

- No puede abrir shell en los servidores
- No puede ejecutar comandos arbitrarios
- No puede crear túneles ni reenviar puertos
- Solo puede ejecutar funciones autorizadas de la API JetBackup5

2. Gate script con whitelist

El script `jetbackup-ssh-gate.sh` valida `SSH_ORIGINAL_COMMAND` contra una lista blanca:

- `jetbackup5api -F getBackupJob` — consultar estado
- `jetbackup5api -F runBackupJobManually` — disparar backup
- `jetbackup5api -F listBackupJobs` — listar trabajos
- `jetbackup5api -F listQueueGroups` — consultar colas
- `jetbackup5api -F stopQueueGroup` — detener cola
- `echo` — test de conectividad

Todo lo demás es **DENIED** y registrado vía syslog.

3. Opciones SSH deshabilitadas

- `no-port-forwarding`: impide túneles SSH
- `no-X11-forwarding`: impide forwarding X11
- `no-agent-forwarding`: impide reutilizar el agente SSH
- `no-pty`: impide obtener terminal interactiva

4. Hardening systemd

El servicio systemd usa: `ProtectSystem=strict`, `PrivateTmp=true`, `NoNewPrivileges=true`, `ProtectHome=true`, `ProtectKernelModules=true`, `ProtectKernelTunables=true`.

Auditoría

```
# Intentar comando no autorizado desde la Pi
ssh -i /root/.ssh/id_ed25519 -p 51514 root@servidor "ls /"
# Esperado: "ERROR: Command not allowed: ls /"

# Verificar en syslog del servidor
journalctl -t jetbackup-ssh-gate | tail -5
```

```
# Verificar restricciones en authorized_keys
grep "jetbackup-ssh-gate" /root/.ssh/authorized_keys
```

Rotación de claves

Si se sospecha compromiso de la clave SSH:

```
# 1. En la Pi, generar nueva clave
ssh-keygen -t ed25519 -f /root/.ssh/id_ed25519_new -N ""

# 2. En cada servidor, reemplazar la clave pública vieja
#    (usar acceso alternativo, no la clave comprometida)

# 3. Verificar acceso con la nueva clave
jetbackup-remote test

# 4. Eliminar la clave vieja
rm /root/.ssh/id_ed25519_old*
```

Logs

```
# Log del orquestador
tail -f /var/log/jetbackup-remote.log

# Log del timer systemd
journalctl -u jetbackup-remote.service -f

# Logs del gate SSH (en cada servidor)
journalctl -t jetbackup-ssh-gate
```

Tests

```
# Ejecutar todos los tests
cd /ruta/a/jetbackup-remote
python3 -m unittest discover -s tests

# Test específico
python3 -m unittest tests.test_orchestrator
```

Licencia

GNU Affero General Public License v3.0 — ver fichero LICENSE.

Autor

Abdelkarim Mateos — abdelkarim@aichadigital.es

Revision #3
Created 2026-02-13 19:00:43 UTC by Abkrim
Updated 2026-02-13 19:24:41 UTC by Abkrim