Sao lưu với PBS (Advance)

Trong phần trước mình đã tìm hiểu về cách tạo backup với PBS. Phần này mình muốn nâng cấp thêm một chút.

Vì thời gian cần backup không nhiều. Việc để PBS chạy lên tục có thể gây lãng phí tài nguyên nên trong phần này mình hướng tới tự động hóa quy trình **bật PBS > Backup > Verify > tắt PBS**

Để thực hiện việc này thì sẽ sử dụng script và systemd timer.

Trước khi đến bới script chính. Thì thêm một điều cần chú ý. Script này sẽ vẫn sử dụng script backup pve host đã viết ở bài Sao lưu với PBS (Simple)

Ok, Lets go!

Quy trình backup

1. Bật PBS
2. Kiểm tra kết nối đảm bảo PBS sẵn sàng
3. Backup PVE host
4. Verify PVE host backup
5. Backup các VM được khai báo
6. Backup các CT được khai báo
7. Verify datastore chứa cả VM và CT backup
8. Prune
9. GC
10. Tắt PBS

Các bước trên được thực hiện tuần tự. Xong bước trên mới thực hiện bước dưới. Dừng ngay khi có lỗi.

Script backup

PVE Web > Shell

- Tạo file thực thi backup `nano /usr/local/bin/backup.sh`

- Cấp quyền thực thi `chmod +x /usr/local/bin/backup.sh`
- Chạy thử `/usr/local/bin/backup.sh`
- Log kết thúc bằng dòng `===== BACKUP PIPELINE DONE =====` là thành công.

```
#!/usr/bin/env bash
set -euo pipefail

########################################
# C?U HÌNH CHUNG
########################################
PBS_VMID=104

PBS_HOST='192.168.1.104'  # ??a ch? IP c?a PBS
PBS_DATASTORE='DSM_NFS' # Datastore ?ã t?o trên PBS
PBS_USER_TOKEN='root@pam!PVE'   # API Token ID
PBS_TOKEN_SECRET='0218e32d-fefe-4bce-9042-c2dd89389856' # API Token Secret
PBS_FINGERPRINT='35:72:be:a5:08:5f:7a:d1:b1:50:0e:69:fa:6c:bf:78:c9:94:d1:9e:45:34:e6:a0:21:88:3d:ab:59:a
6:d8:2d' # Fingerprinter

# Prune and GC job
PRUNE_JOB_ID=${PRUNE_JOB_ID:-default-prune} # Tên Prune job
PRUNE_SCHEDULE=${PRUNE_SCHEDULE:-'Sun 00:00'} # L?ch prune # Không quan tr?ng vì job này ch?y b?ng script
ch? k ch?y t? ??ng
KEEP_BACKUPS=${KEEP_BACKUPS:-7} # s? b?n backup t?i ?ã s? gi?

export PBS_REPOSITORY="${PBS_USER_TOKEN}@${PBS_HOST}:${PBS_DATASTORE}"
export PBS_PASSWORD="${PBS_TOKEN_SECRET}"
export PBS_FINGERPRINT="${PBS_FINGERPRINT}"

# Script backup host pve
# ?ã vi?t ? ph?n tr??c
BACKUP_HOST_SCRIPT="/usr/local/bin/pve-self-backup.sh"

# Danh sách VM/CT c?n backup
# Không c?n d?u ph?y gi?a các s?
VM_LIST=(100 101 104)       # VM IDs
CT_LIST=(102 103)           # CT IDs

# Tu? ch?n vzdump dùng cho VM/CT
VZDUMP_STORAGE="VM_PBS"
VZDUMP_OPTS=(--mode snapshot --compress zstd --ionice 7 --bwlimit 0 --storage "${VZDUMP_STORAGE}" --
notes-template '{{cluster}} - {{guestname}} - {{node}} - {{vmid}}')

# Th?i gian ??i PBS online t?i ?a (giây)
WAIT_PBS_READY_SEC=$((10*60))  # 10 phút
# Th?i gian ??i PBS shutdown t?i ?a (giây)
WAIT_PBS_SHUT_SEC=$((5*60))    # 5 phút

########################################
# HÀM TI?N ÍCH
########################################
# L?y ngày gi?
# Th??ng dùng ?? in ra trong log
ts() { date '+%F %T'; }

# Ki?m tra xem l?nh có chính xác không
require_cmd() {
  command -v "$1" >/dev/null 2>&1 || { echo "[$(ts)] Missing command: $1"; exit 1; }
}

# Ki?m tra VM có ch?y không
qm_is_running() {
  local vmid=$1
```

```
  qm status "$vmid" 2>/dev/null | grep -q 'status: running'
}

# Ki?m tra PBS s?n sàng ch?a b?ng cách
# Th? TCP connect
# ??ng nh?p PBS
# Ki?m tra d?ch v? proxmox-backup-proxy
wait_for_pbs_ready() {
  local deadline=$(( $(date +%s) + WAIT_PBS_READY_SEC ))
  local ok_streak=0
  echo "[$(ts)] Waiting for PBS services (up to ${WAIT_PBS_READY_SEC}s)..."

  while :; do
    # 1) Port 8007 m??
    if ! timeout 2 bash -c "echo > /dev/tcp/${PBS_HOST}/8007" 2>/dev/null; then
      ok_streak=0; goto_sleep; continue
    fi
    # 2) Login repo thành công?
    if ! proxmox-backup-client login --repository "${PBS_REPOSITORY}" >/dev/null 2>&1; then
      ok_streak=0; goto_sleep; continue
    fi
    # 3) D?ch v? proxy active bên trong PBS?
    if ! ssh -o BatchMode=yes -o ConnectTimeout=3 root@"${PBS_HOST}" \
         'systemctl is-active --quiet proxmox-backup-proxy'; then
      ok_streak=0; goto_sleep; continue
    fi

    ok_streak=$((ok_streak+1))
    if (( ok_streak >= 3 )); then
      echo "[$(ts)] PBS ready (proxy up, port 8007 OK, login OK)."
      sleep 5  # grace thêm vài giây
      return 0
    fi

    goto_sleep
  done

  echo "[$(ts)] ERROR: PBS not ready within ${WAIT_PBS_READY_SEC}s"; return 1
}

# ng? 5s
goto_sleep() {
  if (( $(date +%s) > deadline )); then return; fi
  sleep 5
}

# Hàm không còn ???c s? d?ng n?a
# Th? ??ng nh?p PBS
# L?p l?i ??n khi thành công
wait_for_pbs_login() {
  local deadline=$(( $(date +%s) + WAIT_PBS_READY_SEC ))
  echo "[$(ts)] Waiting for PBS to be reachable (up to ${WAIT_PBS_READY_SEC}s)..."
  while :; do
    if proxmox-backup-client login --repository "${PBS_REPOSITORY}" >/dev/null 2>&1; then
      echo "[$(ts)] PBS login OK."
      return 0
    fi
    if (( $(date +%s) > deadline )); then
      echo "[$(ts)] ERROR: PBS not ready within ${WAIT_PBS_READY_SEC}s"
      return 1
    fi
    sleep 5
  done
}
```

```
# Verify data
# B? qua nh?ng phiên b?n ?ã verify
verify_datastore() {
  echo "[$(ts)] Trigger verify on PBS datastore '${PBS_DATASTORE}'..."
  ssh -o BatchMode=yes root@"${PBS_HOST}" \
    "proxmox-backup-manager verify '${PBS_DATASTORE}' --ignore-verified true" \
    && echo "[$(ts)] Verify finished." \
    || { echo "[$(ts)] ERROR: Verify failed"; return 1; }
}

# Sao l?u PVE host s? d?ng pve-self-backup.sh
backup_host() {
  echo "[$(ts)] === BACKUP HOST PVE ==="
  local tries=0
  local max=5
  local rc
  while ((tries<max)); do
    if DRY_RUN=0 "${BACKUP_HOST_SCRIPT}"; then
      echo "[$(ts)] Host backup DONE."; return 0
    fi
    rc=$?
    # N?u l?i k?t n?i thì retry, còn l?i khác thì fail ngay
    if journalctl -u run-backup-pipeline.service -n 50 --no-pager 2>/dev/null | grep -qE 'No route to
host|client error \(Connect\)'; then
      tries=$((tries+1))
      echo "[$(ts)] Connect error, retry ${tries}/${max}..."
      sleep $((5*tries))
    else
      return $rc
    fi
  done
  echo "[$(ts)] ERROR: Host backup failed after retries."; return 1
}

# Backup VMs
backup_vms() {
  echo "[$(ts)] === BACKUP VMs ==="
  for id in "${VM_LIST[@]}"; do
    echo "[$(ts)] vzdump VM ${id} ..."
    vzdump "${id}" "${VZDUMP_OPTS[@]}"
    echo "[$(ts)] VM ${id} backup DONE."
  done
}

# Backup CTs
backup_cts() {
  echo "[$(ts)] === BACKUP CTs ==="
  for id in "${CT_LIST[@]}"; do
    echo "[$(ts)] vzdump CT ${id} ..."
    vzdump "${id}" "${VZDUMP_OPTS[@]}"
    echo "[$(ts)] CT ${id} backup DONE."
  done
}

# Prune
prune_datastore() {
  local job="$PRUNE_JOB_ID"
  echo "[$(ts)] Ensure PRUNE job '${job}' exists (store='${PBS_DATASTORE}', keep-
last=${KEEP_BACKUPS})..."

  # N?u job ?ã t?n t?i -> update (gi? nguyên l?ch ho?c ??t l?i), disable ?? không t? ch?y
  if ssh -o BatchMode=yes root@"${PBS_HOST}" "proxmox-backup-manager prune-job show '${job}' >/dev/null
2>&1"; then
    ssh -o BatchMode=yes root@"${PBS_HOST}" \
      "proxmox-backup-manager prune-job update '${job}' \
```

```
        --store '${PBS_DATASTORE}' \
        --keep-last ${KEEP_BACKUPS} \
        --schedule '${PRUNE_SCHEDULE}' \
        --disable true"
  else
    # T?o job m?i: c?n l?ch h?p l?, nh?ng disable ?? không t? ??ng ch?y
    ssh -o BatchMode=yes root@"${PBS_HOST}" \
      "proxmox-backup-manager prune-job create '${job}' \
        --store '${PBS_DATASTORE}' \
        --schedule '${PRUNE_SCHEDULE}' \
        --keep-last ${KEEP_BACKUPS} \
        --disable true"
  fi || { echo "[$(ts)] ERROR: create/update prune job failed"; return 1; }

  echo "[$(ts)] Running PRUNE job '${job}'..."
  ssh -o BatchMode=yes root@"${PBS_HOST}" \
    "proxmox-backup-manager prune-job run '${job}'" \
    && echo "[$(ts)] Prune finished." \
    || { echo "[$(ts)] ERROR: Prune failed"; return 1; }
}

# GC
gc_datastore() {
  echo "[$(ts)] Running GARBAGE COLLECTION on datastore '${PBS_DATASTORE}'..."
  ssh -o BatchMode=yes root@"${PBS_HOST}" \
    "proxmox-backup-manager garbage-collection start '${PBS_DATASTORE}'" \
    && echo "[$(ts)] GC finished." \
    || { echo "[$(ts)] ERROR: GC failed"; return 1; }
}

# B?t PBS
pbs_start() {
  echo "[$(ts)] Starting PBS VM ${PBS_VMID}..."
  if qm_is_running "${PBS_VMID}"; then
    echo "[$(ts)] PBS already running."
  else
    qm start "${PBS_VMID}"
  fi
  wait_for_pbs_ready
}

# T?t PBS
pbs_shutdown() {
  echo "[$(ts)] Shutting down PBS VM ${PBS_VMID}..."

  # 0) N?u ?ã t?t thì thoát s?m
  if ! qm_is_running "${PBS_VMID}"; then
    echo "[$(ts)] PBS already off."
    return 0
  fi

  # 1) ?u tiên t?t t? bên trong guest (an toàn nh?t)
  #    Yêu c?u b?n ?ã c?u hình SSH key PVE -> PBS
  #    Dùng systemd ?? poweroff không ch? (fork ra n?n)
  ssh -o BatchMode=yes -o ConnectTimeout=5 root@"${PBS_HOST}" \
    'nohup systemctl poweroff >/dev/null 2>&1 &' || true
  echo "[$(ts)] Sent 'systemctl poweroff' to PBS."

  # 2) Ch? PBS t?t h?n (polling qm status)
  SHUT_WAIT=${WAIT_PBS_SHUT_SEC:-300} # 5 phút m?c ??nh
  deadline=$(( $(date +%s) + SHUT_WAIT ))
  while qm_is_running "${PBS_VMID}"; do
    if (( $(date +%s) > deadline )); then
      echo "[$(ts)] Guest did not poweroff within ${SHUT_WAIT}s. Trying 'qm shutdown'..."
      break
```

```
    fi
    sleep 5
  done

  # 3) N?u v?n còn ch?y -> th? 'qm shutdown' v?i timeout dài h?n
  if qm_is_running "${PBS_VMID}"; then
    qm shutdown "${PBS_VMID}" --timeout "${SHUT_WAIT}" || true
  fi

  # 4) Ch? thêm chút n?a
  end2=$(( $(date +%s) + 60 ))
  while qm_is_running "${PBS_VMID}"; do
    if (( $(date +%s) > end2 )); then
      echo "[$(ts)] Still running after graceful attempts. Forcing 'qm stop'..."
      qm stop "${PBS_VMID}" || true
      break
    fi
    sleep 5
  done

  # 5) K?t lu?n: coi nh? thành công n?u th?c s? ?ã t?t
  if ! qm_is_running "${PBS_VMID}"; then
    echo "[$(ts)] PBS is off."
    return 0
  else
    echo "[$(ts)] WARNING: PBS looks stuck running. Please check 'qm monitor ${PBS_VMID}'."
    return 1
  fi
}


########################################
# MAIN
########################################
require_cmd qm
require_cmd proxmox-backup-client
require_cmd ssh
require_cmd vzdump

echo "[$(ts)] ===== BACKUP PIPELINE START ====="

# 1) Start PBS & wait
pbs_start

# 2) Backup host PVE
backup_host

# 3) Verify sau host backup
verify_datastore

# 4) Backup các VM/CT
backup_vms
backup_cts

# 5) Verify sau VM/CT backup
verify_datastore

# 6) PRUNE r?i GC
prune_datastore
gc_datastore

# 7) Shutdown PBS
pbs_shutdown

echo "[$(ts)] ===== BACKUP PIPELINE DONE ====="
```

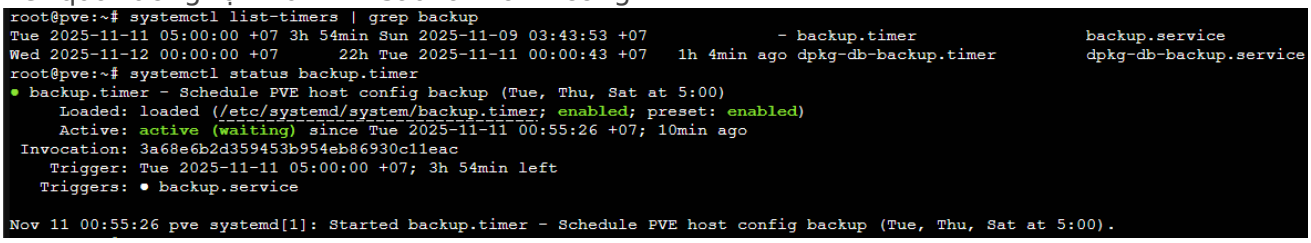
Tự động hóa

Thực hiện trên console hoặc ssh của PVE

- Tạo file `.service`. Đây là hành vi sẽ được thực hiện bởi `.timer`
  - `nano /etc/systemd/system/backup.service`
  - Sử dụng nội dung sau

    ```
    [Unit]
    Description=Backup PVE host configuration to PBS
    After=network-online.target
    [Service]
    Type=oneshot
    ExecStart=/usr/local/bin/backup.sh
    ```
- Tạo file `.timer`. Nó sẽ thực thi `.service` theo thời gian được chỉ định.
  - `nano /etc/systemd/system/backup.timer`
  - Sử dụng nội dung sau

    ```
    [Unit]
    Description=Schedule PVE host config backup (Tue, Thu, Sat at 2:00)
    [Timer]
    OnCalendar=Tue,Thu,Sat 2:00
    Persistent=true
    AccuracySec=1min
    [Install]
    WantedBy=timers.target
    ```
- Kích hoạt timer
  - Reload systemd `systemctl daemon-reload`
  - Kích hoạt timer `systemctl enable --now backup.timer`
- Kiểm tra
  - `systemctl list-timers | grep backup`
  - `systemctl status pve-host-backup.timer`
  - Kết quả tương tự như hình sau là thành công



Revision #2
Created 2025-11-09 09:41:08 UTC by ThanhDV
Updated 2025-11-21 10:12:28 UTC by ThanhDV