Vollständige Anleitung: Automatisches Backup mit Borg (Linux)

Deutsch + Englisch – allgemein formuliert, ohne Projekt-/DB-Namen.

Ziel: Ein Produktionsserver sichert Dateien und Datenbanken automatisch auf einen getrennten Backup-Server. Borg speichert verschlüsselt, nutzt Deduplizierung (gleiche Daten werden nicht doppelt abgelegt) und hält eine definierte Historie.

A. Übersicht & Voraussetzungen

Du hast zwei Server: • **PROD** = Produktionsserver (läuft dauerhaft) • **BACKUP** = Backup-Server (nimmt nur Backups an) Wir sichern: 1) **Dateien** (z. B. /home, /etc, wichtige Konfigurationen) 2) **Datenbanken** (z. B. MariaDB/MySQL) – täglich eine kleinere/„Haupt“-DB und wöchentlich eine sehr große DB. Wichtig: Cron-Jobs laufen ohne interaktives Passwort. Deshalb wird Borg per **BORG_PASSCOMMAND** automatisiert entsperrt.

Platzplanung in einfach

• Borg spart Platz durch Deduplizierung – trotzdem wächst ein Backup-Repository mit der Zeit. • Bei großen Datenbanken gilt: Wenn sich täglich viel ändert, wächst auch das Backup merklich. • Mit 1 TB Backup-Platte ist eine konservative Historie sinnvoll (weniger Wochen/Monate).

B. Backup-Server vorbereiten

Alle Befehle in diesem Abschnitt laufen auf dem **BACKUP**-Server.

```
# 1) Borg installieren
apt-get update
apt-get install -y borgbackup

# 2) eigenen Benutzer für das Repository anlegen (ohne Shell-Login ist ok)
adduser --disabled-password --gecos "" borg

# 3) Repository-Verzeichnis anlegen
mkdir -p /srv/borg
chown borg:borg /srv/borg
chmod 700 /srv/borg
```

Repository initialisieren (als Benutzer borg):

```
su - borg
borg init --encryption=repokey /srv/borg
# Passphrase sicher notieren (ohne sie kommst du später NICHT an die Backups)
```

Optional: Key exportieren (sehr empfohlen):

```
su - borg
borg key export /srv/borg /home/borg/borg-key-export.txt
chmod 600 /home/borg/borg-key-export.txt
```

C. Produktionsserver vorbereiten

Alle Befehle in diesem Abschnitt laufen auf dem **PROD**-Server.

```
# Borg installieren
apt-get update
apt-get install -y borgbackup

# Datenbank-Tools (für MariaDB/MySQL)
apt-get install -y mariadb-client gzip
# (mysqldump ist meist schon dabei, sonst: apt-get install -y mariadb-client)

# SSH-Key für root (oder einen dedizierten Backup-User) erzeugen
ssh-keygen -t ed25519 -C "prod-to-backup"
# ENTER drücken für Standardpfad, dann optional Passphrase (hier: besser ohne, weil Cron)
```

SSH-Key auf BACKUP-Server übertragen:

```
ssh-copy-id borg@BACKUP_IP
# Beim ersten Mal Host-Key bestätigen (yes) und borg-Passwort eingeben
```

Test: Login ohne Passwort muss gehen:

```
ssh borg@BACKUP_IP "echo OK"
```

Passphrase-Automation für Borg (PROD)

Cron hat oft eine „leere“ Umgebung. Deshalb hinterlegen wir ein kleines, ausführbares Passphrase-Kommando. Es liest die Passphrase aus einer Datei mit restriktiven Rechten.

```
# 1) Passphrase-Datei anlegen (Inhalt = exakt die Borg-Passphrase, nur eine Zeile)
mkdir -p /root/.config/borg
chmod 700 /root/.config/borg
nano /root/.config/borg/passphrase
chmod 600 /root/.config/borg/passphrase

# 2) Passcommand-Skript anlegen
cat >/etc/borg-passcommand <<'EOF'
#!/bin/sh
cat /root/.config/borg/passphrase
EOF
chmod 700 /etc/borg-passcommand
```

Test (PROD):

```
BORG_PASSCOMMAND=/etc/borg-passcommand borg info borg@BACKUP_IP:/srv/borg >/dev/null && echo
```

D. Repository testen

Kleines Test-Backup (PROD → BACKUP):

```
# /etc testen
BORG_PASSCOMMAND=/etc/borg-passcommand borg create --stats --compression lz4 \
  borg@BACKUP_IP:/srv/borg::test-etc-$(date +%F) /etc
```

```
# Liste anzeigen (PROD)
BORG_PASSCOMMAND=/etc/borg-passcommand borg list borg@BACKUP_IP:/srv/borg | tail
```

E. Backup-Skripte

Wir nutzen 3 Skripte auf PROD: 1) **backup-home.sh** – tägliches Files-Backup (z. B. /home) 2) **backup-daily.sh** – täglicher Datenbank-Dump der „Haupt“-DB(s) 3) **backup-weekly.sh** – wöchentlich ein Dump einer sehr großen DB (Asset-/Media-/„Monster“-DB) Wichtig: Die Skripte schreiben in ein Logfile. Das ist später deine Hauptkontrolle.

E1) /root/backup-home.sh (PROD)

```
cat >/root/backup-home.sh <<'EOF'
#!/bin/bash
set -euo pipefail

export BORG_PASSCOMMAND="/etc/borg-passcommand"
export BORG_RSH="ssh"

DATE=$(date +%F)
REPO="borg@BACKUP_IP:/srv/borg"
ARCHIVE="home-daily-$DATE"
LOG="/var/log/backup-borg.log"

{
  echo "==== $(date -u) HOME gestartet: $ARCHIVE ===="

  borg create --lock-wait 600 --stats --compression lz4 \
    "$REPO::$ARCHIVE" \
    /home \
    --exclude '/home/*/.cache' \
    --exclude '/home/*/Downloads'

  echo "==== $(date -u) HOME fertig: $ARCHIVE ===="

  # Retention (konservativ)
  borg prune --lock-wait 600 -v --list "$REPO" \
    -a 'home-daily-*' \
    --keep-daily=14 --keep-weekly=8 --keep-monthly=6

  echo "==== $(date -u) HOME prune fertig ===="
} >> "$LOG" 2>&1
EOF

chmod +x /root/backup-home.sh
```

E2) /root/backup-daily.sh (PROD)

Täglicher Dump ausgewählter Datenbanken. Allgemein formuliert: ersetze DB_A und DB_B durch deine „täglichen“ Datenbanken.

```
cat >/root/backup-daily.sh <<'EOF'
#!/bin/bash
set -euo pipefail

export BORG_PASSCOMMAND="/etc/borg-passcommand"
export BORG_RSH="ssh"

DATE=$(date +%F)
REPO="borg@BACKUP_IP:/srv/borg"
ARCHIVE="db-daily-$DATE"
LOG="/var/log/backup-borg.log"

# Hier anpassen:
```

```
DBS=("DB_A" "DB_B")

{
  echo "==== $(date -u) DAILY gestartet: $ARCHIVE ===="

  # Dump -> gzip -> Borg via stdin
  mysqldump --databases "${DBS[@]}" \
    --single-transaction --quick --lock-tables=false \
    --routines --events --triggers \
  | gzip -1 \
  | borg create --lock-wait 600 --stats --compression lz4 \
      "$REPO::$ARCHIVE" - \
      --stdin-name "mysqldump-${DATE}.sql.gz"

  echo "==== $(date -u) DAILY fertig: $ARCHIVE ===="

  borg prune --lock-wait 600 -v --list "$REPO" \
    -a 'db-daily-*' \
    --keep-daily=14 --keep-weekly=8 --keep-monthly=6

  echo "==== $(date -u) DAILY prune fertig ===="
} >> "$LOG" 2>&1
EOF

chmod +x /root/backup-daily.sh
```

E3) /root/backup-weekly.sh (PROD)

Wöchentliches Backup einer **sehr großen** Datenbank. Allgemein: BIG_DB ist die große Asset-/Media-Datenbank. Retention konservativ wegen 1 TB: nur wenige Wochen/Monate.

```
cat >/root/backup-weekly.sh <<'EOF'
#!/bin/bash
set -euo pipefail

export BORG_PASSCOMMAND="/etc/borg-passcommand"
export BORG_RSH="ssh"

DATE=$(date +%F)
REPO="borg@BACKUP_IP:/srv/borg"
ARCHIVE="bigdb-weekly-$DATE"
LOG="/var/log/backup-borg.log"

# Hier anpassen:
BIG_DB="BIG_DB"

{
  echo "==== $(date -u) WEEKLY gestartet: $ARCHIVE ===="

  mysqldump --databases "$BIG_DB" \
    --single-transaction --quick --lock-tables=false \
    --routines --events --triggers \
  | gzip -1 \
  | borg create --lock-wait 600 --stats --compression lz4 \
      "$REPO::$ARCHIVE" - \
      --stdin-name "mysqldump-${BIG_DB}-${DATE}.sql.gz"

  echo "==== $(date -u) WEEKLY fertig: $ARCHIVE ===="

  # Weekly Timeline (sehr konservativ wegen 1TB)
  borg prune --lock-wait 600 -v --list "$REPO" \
    -a 'bigdb-weekly-*' \
    --keep-weekly=4 --keep-monthly=3

  echo "==== $(date -u) WEEKLY prune fertig ===="
} >> "$LOG" 2>&1
EOF
```

```
chmod +x /root/backup-weekly.sh
```

F. Zeitplan (Cron)

Cron läuft auf PROD. Beispiel: /home 03:30, daily DB 04:30, weekly big DB Sonntag 05:00.

```
crontab -e

# Daily /home (03:30)
30 3 * * * /root/backup-home.sh

# Daily DB(s) (04:30)
30 4 * * * /root/backup-daily.sh

# Weekly big DB (Sonntag 05:00)
0 5 * * 0 /root/backup-weekly.sh
```

G. Aufbewahrung (Retention/Prune)

Retention bedeutet: Borg löscht alte Archive automatisch nach Regeln. Das spart Platz und verhindert, dass der Backup-Server „voll läuft“. Die Regeln stehen bereits in den Skripten (prune).

H. Kontrolle: Hat das Backup geklappt?

1) Cron hat den Job gestartet?

```
# Debian ohne /var/log/syslog -> journalctl
journalctl -u cron --since "today" | egrep "backup-home|backup-daily|backup-weekly"
```

2) Logfile prüfen (PROD):

```
tail -n 80 /var/log/backup-borg.log
```

3) Archive im Repository anzeigen (PROD):

```
BORG_PASSCOMMAND=/etc/borg-passcommand borg list borg@BACKUP_IP:/srv/borg | tail -n 20
```

4) Optional: Repository prüfen (BACKUP, lokal):

```
# auf BACKUP (als root oder borg, Passphrase nötig)
borg list /srv/borg | tail -n 20
```

I. Restore – Wiederherstellung

I1) Einzelne Datei aus einem /home-Archiv zurückholen (PROD oder Recovery-Server):

```
# Archiv wählen
BORG_PASSCOMMAND=/etc/borg-passcommand borg list borg@BACKUP_IP:/srv/borg | grep home-daily |

# Datei extrahieren (Beispielpfad)
BORG_PASSCOMMAND=/etc/borg-passcommand borg extract \
  borg@BACKUP_IP:/srv/borg::home-daily-YYYY-MM-DD \
  home/USERNAME/datei.txt
```

I2) Verzeichnis wiederherstellen (in ein Zielverzeichnis):

```
mkdir -p /restore
cd /restore
BORG_PASSCOMMAND=/etc/borg-passcommand borg extract \
  borg@BACKUP_IP:/srv/borg::home-daily-YYYY-MM-DD \
  home/USERNAME/
```

I3) Datenbank-Dump zurückspielen:

```
# 1) Dump aus Borg holen (auf PROD oder Recovery-Server)
mkdir -p /restore
cd /restore
BORG_PASSCOMMAND=/etc/borg-passcommand borg extract \
  borg@BACKUP_IP:/srv/borg::db-daily-YYYY-MM-DD \
  mysqldump-YYYY-MM-DD.sql.gz

# 2) Import
gunzip -c mysqldump-YYYY-MM-DD.sql.gz | mysql -u root
```

I4) Große DB zurückspielen (Weekly):

```
mkdir -p /restore
cd /restore
BORG_PASSCOMMAND=/etc/borg-passcommand borg extract \
  borg@BACKUP_IP:/srv/borg::bigdb-weekly-YYYY-MM-DD \
  mysqldump-BIG_DB-YYYY-MM-DD.sql.gz

gunzip -c mysqldump-BIG_DB-YYYY-MM-DD.sql.gz | mysql -u root
```

J. Disaster Recovery – kompletter Neuaufbau

Ziel: PROD ist kaputt (SSD, Hardware, Neuinstallation). Du willst alles wiederherstellen. Schritte in Kurzform:

```
1) Neuen Server installieren (Debian), Updates einspielen
2) Borg + MariaDB installieren
3) SSH-Key erzeugen, Zugriff auf BACKUP herstellen
4) BORG_PASSCOMMAND einrichten
5) /home aus einem passenden home-daily-* Archiv wiederherstellen
6) tägliche DB(s) aus db-daily-* importieren
7) große DB aus bigdb-weekly-* importieren
8) Dienste starten (Web, Apps, Games/Sim-Software etc.)
9) Funktionstest, dann Cron wieder aktivieren
```

Tipp:

Wenn du ganz sicher gehen willst, mach nach einem großen Restore einmalig ein „frisches“ Full-Initial-Backup. Danach wieder normal inkrementell weiter.

K. Typische Fehler & schnelle Lösungen

Problem: Cron fragt nach Passphrase / „BORG_PASSCOMMAND is not set“

```
# Ursache: Cron-Umgebung ist leer.
# Lösung: export BORG_PASSCOMMAND im Skript (steht oben) + Test:
env -i PATH=/usr/sbin:/usr/bin:/sbin:/bin BORG_PASSCOMMAND=/etc/borg-passcommand \
  borg info borg@BACKUP_IP:/srv/borg >/dev/null && echo OK
```

Problem: „Failed to create/acquire the lock … (timeout)“

```
# Ursache: Ein Borg-Job läuft noch oder hing.
# Lösung: warten oder alte Prozesse prüfen:
ps aux | grep "[b]org"
# Notfalls beenden und erneut starten.
```

Hinweis: „file changed while we backed it up“

Das ist bei Logfiles normal. Es bedeutet nur: die Datei hat sich während des Backups geändert. Für Logs ist das unkritisch.

L. Sicherheit & gute Praxis

• Backup-Server strikt halten: nur Borg-Repo, wenig zusätzliche Dienste. • Passphrase & Key separat sichern. • Regelmäßig testweise einen Restore durchführen (z. B. 1 Datei und 1 DB). • Cron-Jobs zeitlich staffeln. • Retention konservativ, wenn das Repository klein ist (z. B. 1 TB).

Full Guide: Automated Backups with Borg (Linux)

German + English – generic wording, no project- or database-specific names.

Goal: A production server automatically backs up files and databases to a separate backup server. Borg stores data encrypted, uses deduplication (identical blocks are stored only once) and keeps a defined history.

A. Overview & prerequisites

You have two servers: • **PROD** = production server (runs all the time) • **BACKUP** = backup server (receives backups) We back up: 1) **Files** (e.g., /home, /etc, important configs) 2) **Databases** (e.g., MariaDB/MySQL) – a smaller/primary DB daily and one very large DB weekly. Important: Cron must run without interactive prompts. We unlock Borg via **BORG_PASSCOMMAND**.

B. Prepare the backup server

Commands in this section run on the **BACKUP** server.

```
apt-get update
apt-get install -y borgbackup

adduser --disabled-password --gecos "" borg

mkdir -p /srv/borg
chown borg:borg /srv/borg
chmod 700 /srv/borg
```

Initialize the repository (as borg user):

```
su - borg
borg init --encryption=repokey /srv/borg
```

Optional but recommended: export the key:

```
su - borg
borg key export /srv/borg /home/borg/borg-key-export.txt
chmod 600 /home/borg/borg-key-export.txt
```

C. Prepare the production server

Commands in this section run on the **PROD** server.

```
apt-get update
apt-get install -y borgbackup mariadb-client gzip

ssh-keygen -t ed25519 -C "prod-to-backup"
ssh-copy-id borg@BACKUP_IP
ssh borg@BACKUP_IP "echo OK"
```

Passphrase automation (PROD)

```
mkdir -p /root/.config/borg
chmod 700 /root/.config/borg
nano /root/.config/borg/passphrase
chmod 600 /root/.config/borg/passphrase

cat >/etc/borg-passcommand <<'EOF'
#!/bin/sh
cat /root/.config/borg/passphrase
EOF
chmod 700 /etc/borg-passcommand

BORG_PASSCOMMAND=/etc/borg-passcommand borg info borg@BACKUP_IP:/srv/borg >/dev/null && echo
```

D. Repository test

```
BORG_PASSCOMMAND=/etc/borg-passcommand borg create --stats --compression lz4 \
  borg@BACKUP_IP:/srv/borg::test-etc-$(date +%F) /etc

BORG_PASSCOMMAND=/etc/borg-passcommand borg list borg@BACKUP_IP:/srv/borg | tail
```

E. Backup scripts

We use 3 scripts on PROD: 1) **backup-home.sh** – daily file backup (e.g., /home) 2) **backup-daily.sh** – daily dump of your primary DB(s) 3) **backup-weekly.sh** – weekly dump of one very large DB (asset/media/“monster” DB) All scripts append to the same log file.

E1) /root/backup-home.sh (PROD)

```
# Same script as in the German section, just replace BACKUP_IP
# (Already shown above.)
```

E2) /root/backup-daily.sh (PROD)

Replace DB_A and DB_B with your daily databases.

E3) /root/backup-weekly.sh (PROD)

Replace BIG_DB with your very large database.

F. Schedule (Cron)

```
crontab -e

30 3 * * * /root/backup-home.sh
30 4 * * * /root/backup-daily.sh
0  5 * * 0 /root/backup-weekly.sh
```

H. Verification

```
journalctl -u cron --since "today" | egrep "backup-home|backup-daily|backup-weekly"
tail -n 80 /var/log/backup-borg.log
BORG_PASSCOMMAND=/etc/borg-passcommand borg list borg@BACKUP_IP:/srv/borg | tail -n 20
```

I. Restore

```
# Example: restore a database dump from Borg and import
mkdir -p /restore && cd /restore
BORG_PASSCOMMAND=/etc/borg-passcommand borg extract \
  borg@BACKUP_IP:/srv/borg::db-daily-YYYY-MM-DD \
  mysqldump-YYYY-MM-DD.sql.gz

gunzip -c mysqldump-YYYY-MM-DD.sql.gz | mysql -u root
```

J. Disaster recovery (full rebuild)

```
1) Install a fresh Debian system
2) Install borg + MariaDB
3) Setup SSH access to BACKUP
4) Setup BORG_PASSCOMMAND
5) Restore /home from a home-daily-* archive
6) Import db-daily-* dumps
7) Import bigdb-weekly-* dump
8) Start services and verify
9) Re-enable Cron
```