

Backups mit Linux

ITS-Net-Lin

Sebastian Meisel

16. Januar 2025

1 Einführung

Backups sind ein essenzieller Bestandteil der IT-Sicherheit und Datenverwaltung. Sie gewährleisten, dass Daten nach einem Hardwareausfall, versehentlichem Löschen oder einer Malware-Infektion wiederhergestellt werden können. In dieser Einführung betrachten wir die verschiedenen Arten von Backups und wie diese mit dem Tool `rsync` in Linux umgesetzt werden können.

1.1 Funktionsweise von `rsync`

Das Tool `rsync` ist ein Programm zur Synchronisation und Sicherung von Dateien, das standardmäßig auf Linux-Systemen installiert ist. Es arbeitet effizient, indem es nur die geänderten Daten zwischen Quelle und Ziel überträgt. `rsync` nutzt dazu sogenannte delta-Transfers, bei denen nur die tatsächlich veränderten Teile einer Datei kopiert werden.

`rsync` muss unter Umständen zunächst installiert werden:

```
1 sudo apt update                # wenn letzter Aufruf länger als 24h
   zurückliegt
2 sudo apt install rsync
```

1.2 Unterschiedliche Backup-Strategien

Es gibt drei grundlegende Arten von Backups, die sich hinsichtlich Speicherbedarf, Zeitaufwand und Wiederherstellungszeit unterscheiden:

1.2.1 Vollbackup

Ein Vollbackup umfasst alle Daten in einem definierten Verzeichnis oder auf einem gesamten Datenträger. Es ist die vollständigste Form des Backups, benötigt aber am meisten Speicherplatz und Zeit.

- **Vorteil:** Einfaches Wiederherstellen, da alle Daten in einem einzigen Backup enthalten sind.
- **Nachteil:** Hoher Speicher- und Zeitaufwand.

Beispiel mit `rsync`:

```
1 sudo mkdir -p -m 777 /backup/vollbackup
2 rsync -av --progress /quelle/ /backup/vollbackup/ziel
```

`mkdir` Verzeichnis erstellen:

`-p` inklusive Elternverzeichnis (*parent*).

-m 777 setze die Nutzerrechte: `rwX = 111 = 7`

rsync Verzeichnisse synchronisieren:

- a (archive)** Aktiviert den Archivmodus, wodurch Dateien rekursiv kopiert und die wichtigsten Attribute (z. B. Rechte, Besitzer, Zeitstempel) beibehalten werden.
- v (verbose)** Gibt detaillierte Informationen über den Fortschritt aus.
- progress** Zeigt den Fortschritt für jede Datei an.

1.2.2 Inkrementelles Backup

Ein inkrementelles Backup speichert nur die Änderungen, die seit dem letzten Backup (egal welcher Art) vorgenommen wurden. Es benötigt weniger Speicherplatz und Zeit als ein Vollbackup.

- **Vorteil:** Spart Speicherplatz und Zeit.
- **Nachteil:** Wiederherstellung ist komplexer, da alle inkrementellen Backups seit dem letzten Vollbackup benötigt werden.

Beispiel mit `rsync` und Nutzung eines Zeitstempels:

Um ein inkrementelles Backup zu erstellen, müssen wir stets wissen, welche das letzte Backup war - unabhängig davon, ob es ein Voll-, ein inkrementelles oder differentiell Backup war.

Um das zu erreichen setzen wir zunächst einen Softlink auf das Vollbackup:

```
1 mkdir -p -m 777 /backup/latest/  
2 sudo ln -snf /backup/vollbackup/ziel /backup/latest
```

-ln Link erstellen:

- s** Softlink erstellen (keinen Hardlink).
- n** Stelle sicher, dass ein Link auf das Verzeichnis (nicht in dem Verzeichnis) erstellt wird.
- f** Falls bereits ein Link besteht, ersetze ihn.

Nun kann das inkrementelle Backup auf dieser Grundlage erstellt werden:

```
1 sudo mkdir -p -m 777/backup/inkrementell/  
2 rsync -av --progress --link-dest=../../latest/ /quelle/ /backup/inkrementell/ziel-  
  $(date +%Y%m%d)/
```

--link-dest=<Pfad> Verwendet eine Referenz auf ein vorheriges Backup, um Hardlinks zu erstellen. Diese Methode spart Speicherplatz, da unveränderte Dateien nicht erneut kopiert werden. Hier sollte ein relativer Pfad verwendet werden, da dies auch bei Backups auf Remote-Servern funktioniert.

Abschließend muss ein neuer Link erstellt werden:

```
1 sudo ln -snf /backup/inkrementell/$(ls -ltr ziel* | tail -1) /backup/latest
```

\$(...) Füge die Ausgabe der Befehle in Klammern als String ein (Subshell).

ls Verzeichnisinhalt anzeigen:

- 1** Eine Datei pro Zeile.
- t** Sortiere nach Zeit der letzten Änderung.
- r** Umgekehrte Reihenfolge (neueste Datei zuletzt).

tail Letzte Zeilen ausgeben:

- 1** Nur eine (die letzte) Zeile ausgeben.

Das Ganze führt dazu, dass jeweils das inkrementelle verlinkt wird, dass zuletzt erstellt wurde.

1.2.3 Differenzielles Backup

Ein differenzielles Backup speichert alle Änderungen seit dem letzten Vollbackup. Es bietet eine Kompromisslösung zwischen Voll- und inkrementellem Backup.

- **Vorteil:** Schneller als ein Vollbackup, aber weniger aufwändig als inkrementelle Backups.
- **Nachteil:** Kann mit der Zeit speicherintensiv werden, da alle Änderungen seit dem letzten Vollbackup enthalten sind.

Beispiel mit rsync:

```
1 sudo mkdir -p -m 777/backup/differenziell/
2 rsync -av --progress --link-dest=../../vollbackup/ /quelle/ /backup/differenziell/
  ziel-$(date +%Y%m%d)/
```

1.3 Praktische Hinweise

- Vor einem Backup sollte geprüft werden, ob ausreichend Speicherplatz zur Verfügung steht.
- Automatisierungen mit cron oder systemd Timer können regelmäßige Backups sicherstellen.
- Es empfiehlt sich, die Backups regelmäßig zu testen, um sicherzustellen, dass die Wiederherstellung im Ernstfall funktioniert.

1.4 Beispiel für ein Backup-Schema mit systemd-Timer

Ein sinnvolles Backup-Schema könnte wie folgt aussehen:

- **Täglich:** Inkrementelle Backups.
- **Wöchentlich:** Differenzielle Backups.
- **Monatlich:** Vollbackups.

Dazu richten wir mit systemd einen Service und einen Timer ein.

1.4.1 Schritt 1: Backup-Skript erstellen

Erstellen Sie ein Skript, das den Backup-Vorgang ausführt:

```
1 #!/bin/bash
2 SUB_DIR="bros"
3 BACKUP_DIR="/backup"
4 # Für remote Backups, z.B. BACKUP_DIR="deb:/backup"
5 SOURCE_DIR="/home/${SUB_DIR}"
6
7 # Datum im Format YYYYMMDD
8 DATE=$(date +%Y%m%d)
9
10 FULLB="${BACKUP_DIR}/Vollbackup"
11 INCRB="${BACKUP_DIR}/Inkrementell/${DATE}"
12 DIFFB="${BACKUP_DIR}/Differenziell"
13 LASTB="${BACKUP_DIR}/Latest"
14
15 # Backup-Art je nach Argument
16 case "$1" in
17     full)
18         # Stelle sicher, dass Zielverzeichnis existiert
```

```

19 [[ -d "${FULLB}" ]] || mkdir -p -m 777 "${FULLB}"
20 # Synchronisiere
21 rsync -av "${SOURCE_DIR}" "${FULLB}"
22 # Link to Latest
23 ln -snf "${FULLB}/${SUB_DIR}" "${LASTB}"
24 ;;
25 incremental)
26 # Stelle sicher, dass Zielverzeichnis existiert
27 [[ -d "${INCRB}" ]] || mkdir -p -m 777 "${INCRB}"
28 # Synchronisiere
29 rsync -av --link-dest=../../Latest "${SOURCE_DIR}" "${INCRB}"
30 # Link to Latest
31 ln -snf "${INCRB}/${SUB_DIR}" "${LASTB}"
32 ;;
33 differential)
34 # Stelle sicher, dass Zielverzeichnis existiert
35 [[ -d "${DIFFB}" ]] || mkdir -p -m 777 "${DIFFB}"
36 # Synchronisiere
37 rsync -av --link-dest=../../Vollbackup "${SOURCE_DIR}" "${DIFFB}"
38 # Link to Latest
39 ln -snf "${DIFFB}/${SUB_DIR}" "${LASTB}"
40 ;;
41 *)
42 echo "Usage: _$0_ {full|differential|incremental}"
43 exit 1
44 ;;
45 esac

```

Stellen Sie sicher, dass das Skript ausführbar ist:

```

1 chmod +x /usr/local/bin/backup.sh

```

1.4.2 Schritt 2: systemd-Service erstellen

Erstellen Sie eine Datei /etc/systemd/system/backup.service:

```

1 [Unit]
2 Description=Backup Service
3
4 [Service]
5 Type=oneshot
6 ExecStart=/usr/local/bin/backup.sh %i

```

1.4.3 Schritt 3: Timer für Backups erstellen

Erstellen Sie drei Timer-Dateien für die verschiedenen Backup-Typen.

1. Täglicher inkrementeller Timer: /etc/systemd/system/backup@incremental.timer

```

1 [Unit]
2 Description=Daily Incremental Backup Timer
3
4 [Timer]
5 OnCalendar=daily
6 Persistent=true
7
8 [Install]
9 WantedBy=timers.target

```

1. Wöchentlicher differenzieller Timer: /etc/systemd/system/backup@incremental.timer

```
1 [Unit]
2 Description=Weekly Differential Backup Timer
3
4 [Timer]
5 OnCalendar=weekly
6 Persistent=true
7
8 [Install]
9 WantedBy=timers.target
```

1. Monatlicher Vollbackup-Timer: /etc/systemd/system/backup@full.timer

```
1 [Unit]
2 Description=Monthly Full Backup Timer
3
4 [Timer]
5 OnCalendar=monthly
6 Persistent=true
7
8 [Install]
9 WantedBy=timers.target
```

1.4.4 Schritt 4: Timer aktivieren

Aktivieren Sie die Timer:

```
1 systemctl enable --now backup@incremental.timer
2 systemctl enable --now backup@incremental.timer
3 systemctl enable --now backup@full.timer
```

1.5 Moderne Backup-Tools unter Linux

Neben `rsync` gibt es eine Vielzahl moderner Tools, die speziell für Backups entwickelt wurden und viele zusätzliche Funktionen bieten. Einige Beispiele:

1.5.1 BorgBackup (borg)

- **Beschreibung:** Ein modernes deduplizierendes Backup-Tool, das effiziente und sichere Backups ermöglicht.
- **Funktionen:** Datenkomprimierung, Verschlüsselung und effiziente Speicherung durch Deduplizierung.
- **Installation:**

```
1 sudo apt install borgbackup
```

1.5.2 Restic

- **Beschreibung:** Ein sicheres, schnelles und benutzerfreundliches Backup-Tool, das auf vielen Plattformen läuft.
- **Funktionen:** Verschlüsselung, Unterstützung für mehrere Speichersysteme (lokal, Cloud), inkrementelle Backups.

- **Installation:**

```
1 sudo apt install restic
```

1.5.3 Duplicity

- **Beschreibung:** Ein Backup-Tool, das Verschlüsselung und inkrementelle Backups mit Unterstützung für viele Remote-Speicherarten (z. B. Amazon S3) bietet.

- **Funktionen:** Verwendet GPG zur Verschlüsselung, ideal für Cloud-Backups.

- **Installation:**

```
1 sudo apt install duplicity
```
