Backup

Hier werden zwei unterschiedliche Konzepte diskutiert.
BorgBackup ist eine einfache OSS Lösung um zum Beispiel eine Remote-Storagebox (hier von Hetzner) als Speichermedium zu nutzen. Man muss sich jedoch auch bewusst sein, dass durch Borg jede Datei genau einmal gespeichert wird. Sollte eine Datei z.B. durch einen Festplattenfehler beschädigt sein, ist diese in allen Backups beschädigt. Deshalb gehört es zum Best Practice sehr wichtige Daten in mehr als einem Repository zu speichern!
Neben Borg wird ein mit eigenen Scripten erstelltes Backup vorgestellt, dass jedes einzeln durchgeführte Backup wieder herstellen kann, da es vorherige nicht überschreibt. Der Nachteil hierbei ist, dass der Speicherbedarf kontinuierlich anwächst[1)]. Hierbei ist also eine regelmäßige manuelle Wartung erforderlich.

BorgBackup

Quellen:

- Borg - Installation
- Borg - Quick Start
- Hetzner - BorgBackup

BorgBackup (kurz: Borg) ist ein deduplizierendes Backupprogramm. Optional wird Kompression und authentifizierte Verschlüsselung unterstützt.
Die Deduplizierung sorgt bei Borg Backups für einen sehr effizienten Speicherverbrauch und hohe Geschwindigkeit.

Bei diesem Beispiel wurde ein externe Storage-Box von Hetzner eingesetzt (your-storagebox.de). Der User u123456 muss entsprechend Vorgaben der Storage-Box angepasst werden.
Soll das Backup auf einem lokalen Laufwerk oder Verzeichnis gesichert werden, ist der Pfad:

```
ssh://u123456@u123456.your-storagebox.de:23/./backups/...
```

in allen u.a Komandos mit dem entsprechenden lokalen Pfad auszutauschen. Ein (externer) Key für den Zugriff wird dann natürlich nicht benötigt. Wird ein Backup lokal und auf einer verschlüsselten Festplatte durchgeführt, kann auch auf das Passwort/Passphrase verzichtet werden.

Installation

Installation Borg PC/Server

```
sudo apt install borgbackup
```

Key generieren (sofern nicht bereits vorhanden) und auf Storage-Box übertragen, um Zugang ohne Passwort zu gewährleisten

```
ssh-keygen -t rsa -b 4096
```

```
ssh-copy-id -p23 -s u123456@u123456.your-storagebox.de
```

Hetzner nutzt den Port 23 (Standard ist 22), daher muss er explizit angegeben werden (-p23)

Falls das Kopieren auf diese Weise nicht funktioniert, alternativ wie folgt:
Sollte das Verzeichnis auf der Storage-Box noch nicht existieren:

```
ssh -p23 u123456@u123456.your-storagebox.de mkdir .ssh
```

```
scp -P23 .ssh/id_rsa.pub u123456@u123456.your-
storagebox.de:.ssh/authorized_keys
```

Login auf Storage-Box (sollte jetzt ohne PW klappen).

```
ssh -p23 u123456@u123456.your-storagebox.de
```

Ordner (auf der Storage-Box) für Backups erstellen

```
ssh -p23 u123456@u123456.your-storagebox.de mkdir backups
```

Backup erstellen

Ordner (auf der Storage-Box) für dieses System erstellen

```
ssh -p23 u123456@u123456.your-storagebox.de mkdir -p backups/mctest
```

Backup-Ordner auf der Storage-Box initialisieren. Passwort wird abgefragt. Auf einer sicheren Storagebox wäre das doppelte Sicherheit, kann daher ggf. ohne Passwort erfolgen > dann einfach leerlassen. Muss individuell und fallabhängig entschieden werden. Für sicherheitsrelevante Daten ist es vermutlich angezeigt, ein Passwort zu vergeben.

```
borg init --encryption=repokey ssh://u123456@u123456.your-
storagebox.de:23/./backups/mctest
```

Erstes Backup erstellen (test0)

```
borg create ssh://u123456@u123456.your-
storagebox.de:23/./backups/mctest::test0 /home/me/Downloads
```

Hier Testhalber den Ordner Downloads aus dem Home-Verzeichnis des ausführenden Users.

Folge-Backup erstellen (test1)

```
borg create --stats ssh://u123456@u123456.your-
storagebox.de:23/./backups/mctest::test1 /home/me/Downloads
```

Beim Folgebackup werden nur die seit letztem Backup geänderten Daten gesichert. Bei der

Wiederherstellung dieser Sicherung, wird auf alle Dateien zugegriffen, es ist der Gesamt-Stand zu diesem Zeitpunkt.

Backup Inhalt auflisten

Backups für einen PC/Server zeigen → hier mctest

```
borg list ssh://u123456@u123456.your-storagebox.de:23/./backups/mctest
```

Liste gesicherte Dateien aus einem Backup auf Konsole

```
borg list ssh://u123456@u123456.your-
storagebox.de:23/./backups/mctest::test1
```

Liste eines Teils der Backupdateien → hier Verzeichnis /home

```
borg list ssh://u123456@u123456.your-
storagebox.de:23/./backups/mctest::test1 /home
```

Liste Dateien eines Backups in eine Text-Datei schreiben

```
borg list ssh://u123456@u123456.your-
storagebox.de:23/./backups/mctest::test1 > contents.txt
```

Daten wiederherstellen

Inhalt zurückholen

```
cd <restore-verzeichnis>
```

Inhalte werden in das aktuelle, in der Konsole gewählte Verzeichnis zurückgeschrieben. Der komplette Baum wird dort wieder aufgebaut.

```
borg extract ssh://u123456@u123456.your-
storagebox.de:23/./backups/mctest::test1
```

Eine bestimmte Datei oder ein Verzeichnis zurückholen (kompletten Originalpfad angeben). Wird wieder mit gesamtem Pfad in das aktuelle Verzeichnis aufgelöst.

```
borg extract ssh://u123456@u123456.your-
storagebox.de:23/./backups/mctest::test1 /home/me/Downloads/test.txt
```

Backups löschen

Backups für diesen PC (hier mctest) auflisten

```
borg list ssh://u123456@u123456.your-storagebox.de:23/./backups/mctest
```

Einzelnes Backup löschen

```
borg delete ssh://u123456@u123456.your-
storagebox.de:23/./backups/mctest::test1
```

Alle Backups zu einem Server/PC löschen (hier mctest)

```
borg delete ssh://u123456@u123456.your-storagebox.de:23/./backups/mctest
```

Backup automatisieren

Legen Sie ein Verzeichnis für die Logdatei an.

```
sudo mkdir -p /var/log/borg
```

Zunächst muss ein Skript erstellt werden, welches die Backups ausführt. Dies könnte wie folgt aussehen und unter /usr/local/bin/backup.sh liegen.

```
sudo nano /usr/local/bin/backup.sh
```

```
#!/usr/bin/env bash
## Setzten von Umgebungsvariablen
## Pfad zu Ihrem private Key, falls nicht der Standard SSH Key verwendet
wird
export BORG_RSH='ssh -i /home/me/.ssh/id_rsa'
## Passwort in Umgebungsvariable setzen, sofern Repository damit geschützt.
## Andernfalls würde es abgefragt.
export BORG_PASSPHRASE="Repository_Password"
## This has to be set when the repository has been created by user and the
script is called by cron
export BORG_UNKNOWN_UNENCRYPTED_REPO_ACCESS_IS_OK=yes

## Setzten von Variablen
LOG='/var/log/borg/backup_'$(date +%Y-%m-%d_%H%M)'.log'
export BACKUP_USER='u123456'
export REPOSITORY_DIR='mctest'
## Hier Struktur für den Hetzner-Server
export REPOSITORY="ssh://${BACKUP_USER}@${BACKUP_USER}.your-
storagebox.de:23/./backups/${REPOSITORY_DIR}"

## Ausgabe in Logdatei schreiben
exec > >(tee -i ${LOG})
exec 2>&1

echo "###### Backup gestartet: $(date) ######"

## An dieser Stelle können verschiedene Aufgaben vor der Übertragung der
```

```
Dateien ausgeführt werden,
## wie z.B.
## - Liste der installierten Software erstellen
## - Datenbank Dump erstellen

## Dateien ins Repository übertragen
## Gesichert werden hier beispielsweise die Ordner root, etc, var/www und
home
## Ausserdem finden Sie hier gleich noch eine Liste Excludes,die in kein
Backup sollten und
## somit per default ausgeschlossen werden.

echo "Übertrage Dateien ..."
borg create -v --stats                    \
  $REPOSITORY::'{now:%Y-%m-%d_%H:%M}'  \
  /root                                 \
  /etc                                  \
  /var/www                              \
  /home                                 \
  --exclude /dev                        \
  --exclude /proc                       \
  --exclude /sys                        \
  --exclude /var/run                    \
  --exclude /run                        \
  --exclude /lost+found                 \
  --exclude /mnt                        \
  --exclude /var/lib/lxcfs

echo "###### Backup beendet: $(date) ######"
```

Cronjob einrichten

Datei Ausführbar machen

```
sudo chmod u+x /usr/local/bin/backup.sh
```

ggf. Testen

```
sudo /usr/local/bin/backup.sh
```

Einfügen in crontab

```
sudo nano /etc/crontab
```

```
0 0 * * *    root    /usr/local/bin/backup.sh
```

Hier Start täglich um 0:00 Uhr.
ACHTUNG: Wenn das Script durch root ausgeführt werden soll, für die Sicherung von Systemdateien vermutlich erforderlich, muss auch root Zugriff auf den, ggf. entfernten, Backup-Rechner besitzen.

Voraussetzung: das ausführende System muss zu der angegebenen Zeit in Betrieb sein. Der Job wird nicht bei einem späteren Einschalten nachgeholt!

Anacronjob

Läuft ein System nicht dauerhaft, kann dies über **anacron** geregelt werden. Das System prüft regelmäßig (z.B. beim Systemstart) ob Programme/Scripte, die in einem bestimmten Ordner abgelegt sind, ausgeführt wurden und holt dies nach, falls noch nicht geschehen.
Zur Verfügung stehen die Zyklen:

- täglich » Ordner /etc/cron.daily
- wöchentlich » Ordner /etc/cron.weekly
- monatlich » Ordner /etc/cron.monthly

Die ausführbahren Scripte müssen in den entsprechenden Ordnern liegen.

Selbst programmiertes, einfaches Backup

Quellen:

- https://www.ionos.de/digitalguide/server/tools/backup-mit-tar-so-erstellen-sie-archive-unter-linux/

Die vorgestellten Scripte sichern regelmäßig und automatisiert ausgewählte Dateien in gepackten Datein.

- Sichern von MySQL-Datenbanken
- Sichern von Dateien und Verzeichnissen
- Komplexes aber einfaches Backup-System
- Daten eines entfernten Rechners (Remote-System) lokal sichern

Scripte

Datenbanken sichern

Datenbanken eines MySQL-Servers sichern.
Legt bis zu 31 Sicherungen an, jeweils mit Tagesdatum - keine Angabe von Monat oder Jahr.
Bei demselben Datum wird überschrieben.

```
sudo nano /<PFAD>/backup_sql.sh
```

```
#!/bin/bash
BACKUP_DIR="/<PFAD>/dbase"
DAY=$(date +%d)
USER="<BENUTZER>"
```

```
PW="<PASSWORT>"
mkdir -p ${BACKUP_DIR}
DBASE="<DATENBANK>"
mysqldump -u${USER} -p${PW} ${DBASE} > ${BACKUP_DIR}/${DBASE}-${DAY}.sql
exit
```

Ersetzen: <PFAD>, <BENUTZER>, <PASSWORT>, <DATENBANK>
Hier Beispielhaft für eine Datenbank. Die beiden Zeilen „DBASE= ….“ und „mysqldump ….“ einfach vervielfältigen um weitere Datenbanken zu sichern. Bei <BENUTZER> und <PASSWORD> handelt es sich um einen User mit Rechten für die MySQL-Datenbank.

(Zurück)Importieren über Konsole:

```
mysql -u <BENUTZER> -p <DATENBANK> < <SQL-FILE>.sql
```

Die Datenbank muss im MySQL-Server bereits angelegt sein und der bennante Benutzer Zugriffsrechte darauf besitzen.
Ersetzen: <BENUTZER>, <DATENBANK>, <SQL-FILE>
Die Zeichen „größer-als“ (>) und „kleiner als“ (<) entscheiden über die Schreibrichtung. Zu lesen wie ein Pfeil - hier also von der Datei in die Datenbank.

Dateien packen und sichern

```
sudo nano /<PFAD>/backup.sh
```

```
#!/bin/bash
SOURCE="/var/www/data/ "
BACKUP_DIR="/<PFAD>/daten"
mkdir -p ${BACKUP_DIR}
tar -cpzf ${BACKUP_DIR}/backup_data.tar.gz ${SOURCE}
exit
```

Ersetzen: SOURCE, <PFAD>
Leerzeichen am Ende der SOURCE!

Inhalt gepackter Dateien anzeigen

```
tar -tvf backup_gepackt.tar.gz
```

Inhaltsverzeichnis in Datei schreiben

```
tar -tvf backup_gepackt.tar.gz > content.txt
```

Dateien entpacken

Entpacken (in aktuelles Verzeichnis):

```
tar -xf backup_gepackt.tar.gz
```

Parameter für tar (tape archiver)
Siehe auch Ubunutuusers
-c Neues Archiv erzeugen.
-p Zugriffsrechte beim Extrahieren erhalten.
-z Archiv zusätzlich mit gzip (de-)komprimieren.
-x Dateien aus einem Archiv extrahieren.
-f Archiv in angegebene Datei schreiben / Daten aus angegebener Datei lesen. Diese Option muss die letzte sein, da die nachfolgende Zeichen als Datei interpretiert werden.

Komplexes System

Das System sichert regelmäßig[2] bestimmte Datenbereiche und jeder Zeitpunkt, an dem ein Backup durchgeführt wurde, kann wiederhersgestellt werden, solange die Backup-Dateien nicht gelöscht wurden.

Der Speicherbedarf wächst durch dieses System kontinuierlich[3].
Ältere Dateien müssen daher regelmäßig manuell gelöcht oder verschoben werden.
In keinem Fall sollte ein Backup auf der System-Partition gesichert werden, da dieses, sollte der Speicherplatz restlos verbraucht sein, ggf. unbrauchbar oder unausführbar werden könnte.
Eine eigene Partition oder ein eingebundenes NAS mit beschränktem bzw. reguliertem Speicher bietet sich an.
Aber auch das ist zu kontrollieren, da das Backup nicht mehr ausgeführt wird, wenn der Speicherplatz verbraucht ist.

1. Das Script verbindet das System mit einem Remote-NAS (mount) und trennt die Verbindung am Ende wieder.
2. Es werden nur die Daten gesichert, die seit dem letzten Backup geändert wurden - ohne dass der letzte Stand, das letzte Backup, überschrieben wird.
3. Nach einer festgelegten Anzahl Backups (hier 30) werden die Backups in ein Unterverzeichnis **rotate/<ZEITSTEMPEL>** verschoben und das System beginnt von vorne → zunächst ein komplett-Backup, dann wieder 30x nur geänderte Daten.
4. Es können definierte Unterverzeichnisse ausgeschlossen werden (exclude)

Hier Beispielhaft die Datensicherung des Verzeichnisses /home
Ausgenommen wird das darin enthaltene Verzeichnis /home/menot

```
sudo nano /<PFAD>/backup.sh
```

```
SOURCE="/home "
EXCLUDE="/home/menot/*"
mountPoint="/mnt/backup"
BACKUP_DIR="${mountPoint}/homes"
```

```
ROTATE_DIR="${BACKUP_DIR}/rotate"
TIMESTAMP="timestamp.dat"

mount "${mountPoint}"
if ! [ `mount |grep "${mountPoint}" | wc -l` -eq 1 ]; then
  exit
fi

DATE=$(date +%Y-%m-%d-%H%M%S)

cd /
#Verzeichnis anlegen, falls nicht existent
mkdir -p ${BACKUP_DIR}

set -- ${BACKUP_DIR}/backup-??.tar.gz
lastname=${!#}
backupnr=${lastname##*backup-}
backupnr=${backupnr%%.*}
backupnr=${backupnr//\?/0}
backupnr=$[10#${backupnr}]

if [ "$[backupnr++]" -ge 30 ]; then
mkdir -p ${ROTATE_DIR}/${DATE}
mv ${BACKUP_DIR}/b* ${ROTATE_DIR}/${DATE}
mv ${BACKUP_DIR}/t* ${ROTATE_DIR}/${DATE}
backupnr=1
fi

backupnr=0${backupnr}
backupnr=${backupnr: -2}
filename=backup-${backupnr}.tar.gz
tar -cpzf ${BACKUP_DIR}/${filename} -g ${BACKUP_DIR}/${TIMESTAMP} --
exclude="${EXCLUDE}" ${SOURCE}
umount "${mountPoint}"
exit
```

Backup ausführen

Scripte ausführbar machen (hier alle .sh-Dateien im angegebenen Verzeichnis):

```
sudo chmod +x /<PFAD>/*.sh
```

Script manuell ausführen:

```
sudo bash /<PFAD>/backup.sh
```

Automatisiert als root ausführen über Cronjob

```
sudo nano /etc/crontab
```

```
# /etc/crontab                 #system-wide crontab
0 4 * * *   root    /PFAD/backup_sql.sh
0 3 2 * *   root    /PFAD/backup.sh
```

Lesart der (einschränkenden) Zahlen:
Minute, Stunde, Tag im Monat, Monat, Wochentag.
Stern bedeutet jeder dieser Gruppe, unter Einschränkung der anderen Angaben.

Ausführung der o.a. Angaben:

- jeden Tag um 4:00 Uhr → SQL-Datenbank(en)
- jeden 2. Tag im Monat um 3:00 Uhr → Dateien packen

Voraussetzung: der ausführende PC muss zu der angegebenen Zeit in Betrieb sein. Der Job wird nicht bei einem späteren Einschalten nachgeholt!
Andernfalls kann das über einen Anacronjob geregelt werden.

Daten von Remote-System holen

Von einem lokalen System können Daten eines entfernten Systems (z.B. Backups) austomatisiert abgeholt/kopiert werden. Der Login auf den Remote-System muss mit einer SSH-Zertifikatsdatei ohne Passwort für den im Cronjob stehenden User realisiert sein. Siehe SSH-Verbindungen.

```
sudo nano /<PFAD>/get_remote_files.sh
```

```
#!/bin/bash
scp -r <USERNAME>@<REMOTESERVER>:/<PFADEXTERN>/ /<PFADLOKAL>/
exit
```

Ersetzen: <PFADEXTERN>, <PFADLOKAL>, <USERNAME>, <REMOTESERVER> (IP-Adresse)
scp = Kopieren über gesicherte ssh-Verbindung
-r = Kopieren inkl. aller Unterverzeichnisse
Achtung: Es kann nur kopiert werden, wenn auch Lese-Zugriff vom einloggenden <USERNAME> auf die Dateien im Remote-System besteht - z.B. bei Systemdateien. Ist dies nicht möglich, könnte am Remote-System ein regelmäßiges Backup die betroffenen Dateien in einen Bereich sichern, auf den der benannte User Zugriff hat.

Die (angepasste) Zeile „scp -r ….“ kann zur manuellen Ausführung auch direkt als Befehl auf der Konsole eingegeben werden.
Sind lokaler User und Remoteuser identisch, kann „<USERNAME>@“ bei der manuellen Ausführung entfallen.

Auch hier ist die Automation über einen Cronjob möglich, siehe Backup ausführen.

[1)] [3)]
in dem Rahmen, in dem es Änderungen am Datenbestand gab
[2)]
Zyklus definiert durch /etc/crontab