Backup

BorgBackup

BorgBackup (kurz: Borg) free and open source software, 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. Man muss sich jedoch auch bewusst sein, dass dadurch 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!

Quellen & Tutorials:

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

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

```
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
##
## falls nicht der Standard SSH Key verwendet wird können
## Sie hier den Pfad zu Ihrem private Key angeben
export BORG_RSH='ssh -i /home/me/.ssh/id_rsa'
## Damit das Passwort vom Repository nicht eingegeben werden muss
## kann es in der Umgepungsvariable gesetzt werden
export BORG_PASSPHRASE="sehr_sicheres_passwort"
##
## Setzten von Variablen
##
LOG='/var/log/borg/backup_'$(date +%Y-%m-%d_%H%M)'.log'
export BACKUP_USER='u123456'
export REPOSITORY_DIR='mctest'

## Hinweis: Für die Verwendung mit einem Backup-Account muss
## 'your-storagebox.de' in 'your-backup.de' geändert werden.

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) ######"
```

Ausführbar machen

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

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

Selbst programmiertes, einfaches Backup

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

Das vorgestellte Konzept: Dateien und Datenbanken auf einem entfernten (remote) Server

automatisch und regelmäßig auf diesem sichern. Automatisch und regelmäßig die Daten auf einen lokalen Rechner/Server kopieren.

Scripte Remote-System

Dateien sichern

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

```
#!/bin/bash
SOURCE="/etc/apache2/sites-available/ "
BACKUP_DIR="/<PFAD>/daten"
mkdir -p ${BACKUP_DIR}
cp -p -u -r ${SOURCE} ${BACKUP_DIR}
cp -p -u <ONE-FILE> ${BACKUP_DIR}
exit
```

Ersetzen: SOURCE, <PFAD>, <ONE-FILE>
Leerzeichen am Ende der SOURCE!
Verzeichnis wird angelegt, sofern nicht existent.
So können einzelne Dateien <ONE-FILE> oder ganze Verzeichnisse SOURCE gesichert werden.

Parameter für cp (copy)
-p oder - -preserve ⇒ Attribute der Originaldatei werden beim Kopieren vererbt
- -preserve=timestamp ⇒ Zeitstempel der Originaldatei wird beim Kopieren vererbt
-r oder -R oder - -recurcive ⇒ Verzeichnisse inkl. Unterverzeichnisse
-u oder - -update ⇒ Kopiert die Datei nur, wenn die Zieldatei älter als das Original ist
-v oder - -verbose ⇒ Gibt nach dem Kopiervorgang eine Meldung aus
Werden (nur) einzelne Dateien kopiert, müssen die Zielverzeichnisse bereits existieren!

Siehe auch UbuntuUsers

Systemdateien können ggf. nur durch root gesichert werden (z.B let's-Encrypt-Zertifikate). Wird die Sicherung wie unten beschrieben automatisiert durchgeführt, erfolgt der Zugriff durch root und sollte problemlos klappen. Beim externen Abholen als „normaler User“ könnte es Zugriffsprobleme geben (Prüfen, o User Leserechte auf die gesicherten Dateien hat). Durch Packen der Daten ins Home-Verzeichnis des <USERS> kann man das Problem umgehen, da dann Zugriff auf die gepackten Daten besteht.

Dateien packen und sichern

```
sudo nano /<PFAD>/backup_gepackt.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!
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.

Datenbanken sichern

```
sudo nano /<PFAD>/backup_dbase.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.

Legt bis zu 31 Sicherungen an, jeweils mit Tagesdatum.
Keine Angabe von Monat oder Jahr.
Bei demselben Datum wird überschrieben.

(Zurück)Importieren über Konsole:

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

Die Datenbank muss bereits existieren 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.

Backup ausführen

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

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

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

Die Ausführung wird hier eingeschränkt auf Datei-Eigentümer (u+x) und Gruppe (g+x)

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_dbase.sh
0 3 1 * *   root    /PFAD/backup.sh
0 3 2 * *   root    /PFAD/backup_gepackt.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 → Datenbank(en)
- jeden 1. Tag im Monat um 3:00 Uhr → Dateien
- jeden 2. Tag im Monat um 3:00 Uhr → Dateien packen

Daten auf lokalen PC kopieren

Von einem lokalen Debian-/Linux-System kann das Backup des Remote-Systems abgeholt/kopiert werden. Der Login auf den Remote-Server muss mit einer SSH-Zertifikatsdatei ohne Passwort für den im Cronjob stehenden User realisiert sein. Siehe SSH-Verbindungen.

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

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

Ersetzen: <PFADEXTERN>, <PFADLOKAL>, <USERNAME>, <REMOTESERVER> (IP-Adresse)
Sind lokaler User und Remoteuser bei der manuellen Ausführung identisch, kann „<USERNAME>@“ entfallen.
scp = Kopieren über gesicherte ssh-Verbindung

-r = Kopieren inkl. aller Unterverzeichnisse

Die (angepasste) Zeile „scp -r ….“ kann zur manuellen Ausführung auch direkt als Befehl auf der Konsole eingegeben werden

Automatisiert über Cronjob

Voraussetzung: der ausführende PC muss zu der Zeit in Betrieb sein. Der Job wird nicht bei einem späteren Einschalten nachgeholt!
Läuft der PC nicht dauerhaft, kann dies über **anacron** geregelt werden, siehe Ubuntuusers.de -> Cron. Anacron führt die Jobs regelmäßig aus (täglich, wöchentlich oder monatlich), auch wenn der PC nicht immer an ist. Die ausführbaren Dateien müssen dann in den entsprechenden Ordnern liegen (/etc/cron.daily, /etc/cron.weekly, …). Anacron holt Jobs beim starten des Systems nach, wenn ihre Ausführung noch offen ist.

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

```
# /etc/crontab: system-wide crontab
0 4 * * 7               <USERNAME>      /<PFAD>/backup_externer_server.sh
```

Ausführung:

- jeden 7. Tag der Woche (Sonntag) um 4:00 Uhr

Achtung: Es kann nur kopiert werden, wenn auch Zugriff vom einloggenden <USERNAME> auf die externen Dateien besteht. Sollen z.B. die letsencrypt-Zertifikate kopiert werden, dann müssen hierfür am externen System Zugriffsrechte vorliegen. Soll Z.B. das Backup auf dem entfernten System (als root über cronjob) ins Home-Verzeichnis schreiben, aber trotzdem keine Zugriffsrechte darauf bestehen (Prüfen) Dateien vorher packen, siehe oben.