

Backup – eine tägliche, wöchentliche, monatliche Dienstpflicht für den Admin?!

Quelle: Artikel aus der Ix Oliver Frommel 09.04.2018



Die meisten ADMIN-Leser verwenden für Backup eigene Skripts. Bei kommerziellen Lösungen liegt ein relativer Newcomer vorn.

Laut einer Online-Umfrage von ADMIN verwenden knapp 20 Prozent der Leser für Backups eigene Skripts. Dies ist unter den den knapp 15 Vorschlägen der höchste Einzelwert. Die restlichen Lösungen und Produkte sind mit ein bis

Welche Backup-Lösung setzen Sie für Linux im professionellen Umfeld ein?

- keine (400) 12.40%
- eigene Scripts (633) 19.63%
- rsnapshot (222) 6.885%
- rdiff-backup (89) 2.760%
- Bacula (171) 5.303%
- Bareos (169) 5.241%
- Borg (155) 4.807%
- Duplicity (80) 2.481%
- Amanda (31) 0.961%
- Burp (22) 0.682%
- Clonezilla (131) 4.063%
- Acronis (130) 4.032%
- Arkeia (16) 0.496%
- SEP sesam (71) 2.202%
- Veeam (381) 11.81%
- Redo Backup (23) 0.713%
- Relax-and-Recover (28) 0.868%
- andere kommerzielle Software (231) 7.165%
- andere Open-Source-Software (241) 7.475%

sieben Prozent relativ gleichmäßig verteilt.

Leicht die Nase vorn hat unter den Open-Source-Lösungen dabei rsnapshot, das mit einem trickreichen Hardlink-Ansatz hilft Speicherplatz zu sparen. Dies und das ähnlich funktionierende rdiff-backup stellt der ADMIN-Artikel "[Backups mit rdiff-backup und rsnapshot](#)" näher vor. Empfohlen wird die Lektüre auch den gut zwölf Prozent der ADMIN-Leser, die komplett auf Backups verzichten. Mit je gut fünf Prozent kommen der Klassiker Bacula und sein Fork [Bareos](#) zusammen auf mehr als zehn Prozent. [Borg](#), das Verschlüsselung und Deduplizierung unterstützt, wird von gut vier Prozent der ADMIN-Leser verwendet.

Bei den namentlich gelisteten kommerziellen Programmen sind Acronis mit vier und SEP Sesam mit zwei Prozent vertreten. Heraus ragt hier die Lösung von Veeam, die auf das Backup virtueller Maschinen spezialisiert ist, und die von fast zwölf Prozent der Befragten eingesetzt wird.

An der Umfrage teilgenommen haben über 3200 Leser. Jetzt läuft eine [neue Umfrage zur verwendeten Container-Technologie](#).

Ein Beispiel für die Anwendung einer Backup-Software

Quelle: [zu ix.de](https://www.ix.de) Michael Plura, iX 5/2018, S. 142



Datensicherung u. Wiederherstellung mit Borg-Backup

BorgBackup ist eine einfach zu bedienende Backup-Software, die wenig Ressourcen benötigt. Es füllt die Lücke zwischen einfachen Tools wie rsync und Backup-Frameworks wie Bacula oder Bareos.

BorgBackup, kurz Borg, ist ein in Python 3 entwickeltes Backup-Programm, dessen zeitkritische Teile der Software in C/Cython geschrieben und kompiliert wurden. Es basiert ursprünglich auf Attic und wird von einem Team rund um den Schweden Jonas Borgström entwickelt – daher der Name. Zu sichernde Daten dedupliziert und komprimiert es, was die benötigte Bandbreite zum und den Platz im Backup-Storage

gering hält. Dank der Authentifizierung per HMAC-SHA256 und einer optionalen, clientseitigen Verschlüsselung mit 256-Bit-AES kann man Backups ohne allzu große Bedenken auf nicht vertrauenswürdigen Zielen wie der Cloud anlegen.

Das quelloffene Borg unterliegt der BSD-Lizenz und ist in der Regel über das Paketmanagement von Linux, FreeBSD, OpenBSD, NetBSD oder Mac OS X (brew) als *borg* oder *borgbackup* zu installieren. Für WSL (Windows Subsystem for Linux) oder Cygwin gibt es eine experimentelle Variante. Alternativ lässt sich die Software über den Python-Installer *pip* einrichten:

```
pip install borgbackup
```

Zusätzlich stehen auf der GitHub-Seite des Projekts Binaries für Linux, FreeBSD und Mac OS X zur Verfügung. Benutzer rufen diese, falls sie im *\$PATH* liegen, per *borg* auf. Eine Übersicht über alle Befehle und zusätzliche Infos zu einem bestimmten Befehl erhält man per

```
borg --help
```

```
borg create --help
```

Die Entwickler stellen eine gute Onlinedokumentation unter borgbackup.readthedocs.io bereit.

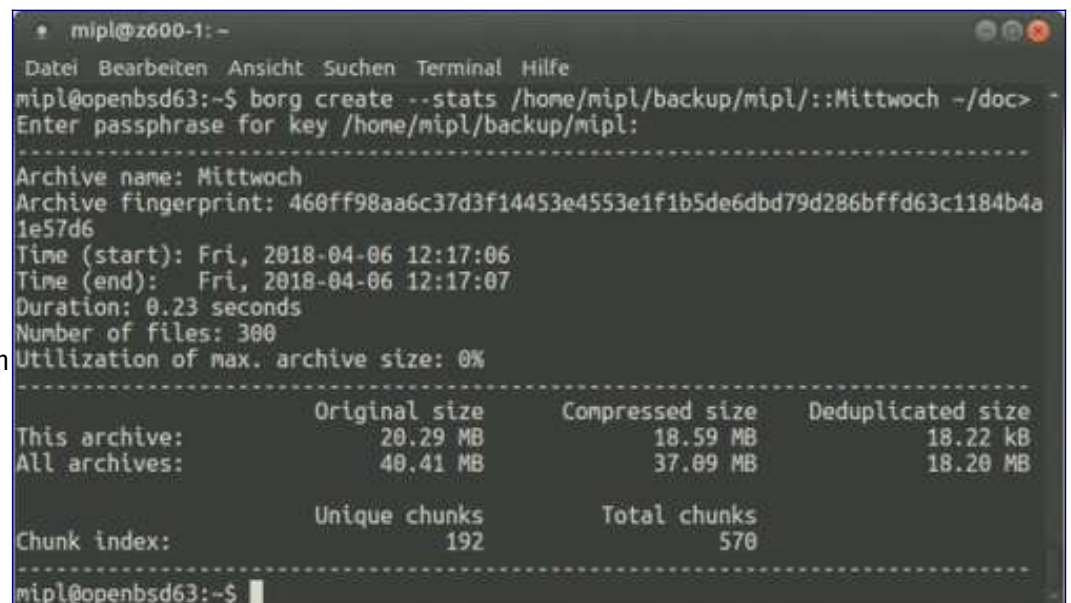
Borg organisiert das Backup-Storage in Form eines Repository (ein Verzeichnis), in dem die einzelnen Backups als Info- und Datenblöcke abgelegt werden.

Im Beispiel sichert und überprüft der Nutzer *mipl* seine Dokumente und Quellen verschlüsselt auf einem per NFS eingebundenen Server:

```
borg init --encryption=repokey /<path>/mipl
```

Diese Befehle legt das Repository unter */<path>p/mipl* an und fragt den Benutzer nach einem Passwort, das für jeden weiteren Zugriff erforderlich ist. Das Passwort kann als Umgebungsvariable *export BORG_PASSPHRASE="KeineGuteld666"* für einen automatischen Betrieb gesetzt werden.

Das freie BorgBackup sichert Daten schnell, einfach und platzsparend.



```
mipl@z600-1: ~$ borg create --stats /home/mipl/backup/mipl::Mittwoch ~/doc
Enter passphrase for key /home/mipl/backup/mipl:
-----
Archive name: Mittwoch
Archive fingerprint: 460ff98aa6c37d3f14453e4553e1f1b5de6dbd79d286bffd63c1184b4a1e57d6
Time (start): Fri, 2018-04-06 12:17:06
Time (end):   Fri, 2018-04-06 12:17:07
Duration: 0.23 seconds
Number of files: 300
Utilization of max. archive size: 0%
-----
Original size  Compressed size  Deduplicated size
This archive:  20.29 MB      18.59 MB      18.22 kB
All archives:  40.41 MB      37.09 MB      18.20 MB
-----
Unique chunks  Total chunks
Chunk index:   192          570
-----
mipl@openbsd63:~$
```

Ein Backup erzeugt man mit dem Argument *create* und einem Namen, im Beispiel „Montag“. Gesichert werden die Verzeichnisse *doc* und *code*:

```
borg create --stats /<path>/mipl::Montag ~/doc ~/code
```

```
borg check -v --verify-data /<path>/ mipl::Montag
```

Derselbe Befehl mit anderen Labels legt weitere, deduplizierte Backups an. Eine Übersicht liefert *borg list*. Nicht mehr benötigte Sicherungen löscht der Befehl

```
borg delete /<path>/mipl::Montag
```

Dabei berücksichtigt Borg die Deduplizierung und hält immer einen kompletten Backup-Satz vor. Eine vollständige Datensicherung stellt man mit

```
borg extract --list /<path>/mipl::Freitag
```

wieder her. Ein Repository oder einzelnes Backup lässt sich über FUSE in das Dateisystem einbinden, um so mit einem Dateimanager einzelne Dateien zu restaurieren:

```
mkdir tmp
```

```
borg mount /<path>/mipl::Freitag ~/tmp
```

```
[...]
```

```
borg unmount ~/tmp
```

Normalerweise komprimiert Borg die Daten mit dem schnellen *lz4*. Besser und dabei anpassbar ist *zstd*, *N*, wobei *N* zwischen *N*=1 (schnell) und *N*=22 (hochkomprimiert) gewählt wird. Zum Testen (und Lernen) lassen sich die meisten Borg-Befehle mit den Parametern *-v* – *-list* – *-dry-run* starten. Sie zeigen dann ausführlich, was passiert, nehmen aber keinerlei Änderungen vor. (jab@ix.de)

Zur Übung nun noch ein paar Aufgaben zu „BORG“-Backup

1. Wie effektiv und mit welcher Methode wird der verfügbare Backup-Speicher genutzt?

Deduplication based on content-defined chunking is used to reduce the number of bytes stored: each file is split into a number of variable length chunks and only chunks that have never been seen before are added to the repository.

A chunk is considered duplicate if its *id_hash* value is identical. A cryptographically strong hash or MAC function is used as *id_hash*, e.g. (*hmac*-)*sha256*.

To deduplicate, all the chunks in the same repository are considered, no matter whether they come from different machines, from previous backups, from the same backup or even from the same single file.

Compared to other deduplication approaches, this method does NOT depend on:

- file/directory names staying the same: So you can move your stuff around without killing the deduplication, even between machines sharing a repo.
- complete files or time stamps staying the same: If a big file changes a little, only a few new chunks need to be stored - this is great for VMs or raw disks.
- The absolute position of a data chunk inside a file: Stuff may get shifted and will still be found by the deduplication algorithm.

2. Wie wird eine hohe backup- und restore-Geschwindigkeit erreicht?

- performance-critical code (chunking, compression, encryption) is implemented in C/Python
- local caching of files/chunks index data
- quick detection of unmodified files

3. Wird das Backup zur Sicherheit verschlüsselt – und wo wird es verschlüsselt (server-/clientseitig)?

All data can be protected using 256-bit AES encryption, data integrity and authenticity is verified using HMAC-SHA256. Data is encrypted clientside.

4. Werden die Backups komprimiert und mit welchen Algorithmus bzw. welcher Technik?

All data can be optionally compressed:

- *lz4* (super fast, low compression)
- *zstd* (wide range from high speed and low compression to high compression and lower speed)
- *zlib* (medium speed and compression)
- *lzma* (low speed, high compression)

5. Ist das Backup nur auf dem lokalen Host/PC oder auch remote auf einem entfernten Server / einer entfernten Cloud möglich?

Borg can store data on any remote host accessible over SSH. If Borg is installed IoT the remote host, big performance gains can be achieved compared to using a network filesystem (sshfs, nfs, ...).

6. Lassen sich die Backups zum Auffinden einzelner Verzeichnisse/Dateien etc. in ein anderes Dateisystem einbinden, durchsuchen und einzeln wieder herstellen?

Backup archives are mountable as userspace filesystems for easy interactive backup examination and restores (e.g. by using a regular file manager).

7. Für welche Betriebssysteme und Anwendungen steht die Software zur Verfügung und wie einfach lässt sie sich installieren und handhaben?

We offer single-file binaries that do not require installing anything - you can just run them on these platforms:

- Linux
- Mac OS X
- FreeBSD
- OpenBSD and NetBSD (no xattrs/ACLs support or binaries yet)
- Cygwin (experimental, no binaries yet)
- Linux Subsystem of Windows 10 (experimental)

8. Ist die Software Free, Open Source oder proprietär und lizenspflichtig?

- security and functionality can be audited independently
- licensed under the BSD (3-clause) license, see [License](#) for the complete license

Befehlsübersicht Borg	
Befehl <i>mit optionalen Ergänzungen u. Konkretisierungen</i>	Beschreibung
<i>borg break-lock</i> <i>[OPTIONEN]</i> REPO_VERZEICHNISPFAD	Hebt den Lockmechanismus auf. Nur benutzen, wenn kein Prozess auf das Repository zugreift!
<i>borg change-passphrase</i> <i>[OPTIONEN]</i> REPO_VERZEICHNISPFAD	Ändert die Passphrase
<i>borg check</i> <i>[OPTIONEN]</i> REPO_VERZEICHNISPFAD <i>[::ARCHIVNAME]</i>	Überprüfe die Konsistenz des gesamten Repositorys oder von einzelnen Archiven
<i>borg delete</i> <i>[OPTIONEN]</i> REPO_VERZEICHNISPFAD <i>[::ARCHIVNAME]</i>	Löscht einzelne Archive oder das gesamte Repository
<i>borg info</i> <i>[OPTIONEN]</i> REPO_VERZEICHNISPFAD <i>::ARCHIVNAME</i>	Ausgabe von Informationen über ein Archiv
<i>borg list</i> <i>[OPTIONEN]</i> REPO_VERZEICHNISPFAD <i>[::ARCHIVNAME]</i>	Gibt den Inhalt eines Repositorys oder eines Archivs aus
<i>borg rename</i> <i>[OPTIONEN]</i> REPO_VERZEICHNISPFAD <i>::ARCHIVNAME</i> NEUER_ARCHIVNAME	Umbenennung eines Archivs

9. Erläutere nachfolgende Befehle und das Skript auf der nächsten Seite:

```
borg prune --dry-run -v --list --keep-within=1d --keep-daily=7 --keep-weekly=4 --keep-monthly=12 /media/peter/HD_Backup/borgbackups
```

```
export BORG_RSH='ssh -i /PFAD/SCHLÜSSELDATEI'
```

Das Backupscript:

```
1  #!/bin/bash
2
3  # Skriptvorlage BorgBackup
4  # https://wiki.ubuntuusers.de/BorgBackup/
5  # https://borgbackup.readthedocs.io/en/stable/
6
7  # Hier Pfad zum Sicherungsmedium angeben.
8  # z.B. zielpfad="/media/peter/HD_Backup"
9  zielpfad=""
10
11 # Hier Namen des Repositorys angeben.
12 # z.B. repository="borgbackups"
13 repository=""
14
15 # Hier eine Liste mit den zu sichernden Verzeichnissen angeben
16 # z.B. sicherung="/home/peter/Bilder /home/peter/Videos --exclude *.tmp"
17 sicherung=""
18
19 # Hier die Art der Verschlüsselung angeben
20 # z.B. verschluesselung="none"
21 verschluesselung="repokey"
22 # Hier die Art der Kompression angeben
23 # z.B. kompression="none"
24 kompression="lz4"
25 # Hier angeben, ob vor der Ausführung von BorgBackup auf vorhandene
26 # Root-Rechte geprüft werden soll z.B. rootuser="ja"
27 rootuser="nein"
28
29 # Hier angeben nach welchem Schema alte Archive gelöscht werden sollen.
30 # Die Vorgabe behält alle Sicherungen des aktuellen Tages. Zusätzlich das
31 # aktuellste Archiv der
32 # letzten 7 Sicherungstage, der letzten 4 Wochen sowie der letzten 12 Monate.
33 pruning="--keep-within=1d --keep-daily=7 --keep-weekly=4 --keep-monthly=12"
34 #####
35 repopfad="$zielpfad"/"$repository"
36 # check for root
37 if [ $(id -u) -ne 0 ] && [ "$rootuser" == "ja" ]; then
38     echo "Sicherung muss als Root-User ausgeführt werden."
39     exit 1
40 fi
41 # Init borg-repo if absent
42 if [ ! -d $repopfad ]; then
43     borg init --encryption=$verschluesselung $repopfad
44     echo "Borg-Repository erzeugt unter $repopfad"
45 fi
46 # backup data
47 SECONDS=0
48 echo "Start der Sicherung $(date)."
49 borg create --compression $kompression --exclude-caches --one-file-system -v \
50     --stats --progress \
51     $repopfad::'{hostname}-{now:%Y-%m-%d-%H%M%S}' $sicherung
52 echo "Ende der Sicherung $(date). Dauer: $SECONDS Sekunden"
53 # prune archives
54 borg prune -v --list $repopfad --prefix '{hostname}-' $pruning
```


Das zeitgesteuerte Backup

10. Beschreiben Sie die Unterschiede eines inkrementellen, eines differentiellen und eines Vollbackups!
11. Erstellen Sie ein zeitgesteuertes Backup-Skript, das Vollbackups und inkrementelle Backups ermöglicht an Hand nachfolgend beschriebener einfacher Beispiele!

Zuerst das Backup

Das folgende Script zeigt, wie man den Backup eines Verzeichnisses automatisieren kann. Normalerweise wird ein inkrementeller Backup erzeugt, d. h. es werden nur die Dateien gesichert, die nach dem letzten Backup geändert oder neu erstellt wurden. Will man ein volles Backup, muß man den Parameter '-a' (für 'all') angeben. Um festzustellen, welche Dateien neu sind, wird im entsprechenden Verzeichnis eine leere Datei namens '.lastbackup' angelegt bzw. deren Zugriffsdatum aktualisiert. Nach deren Änderungsdatum richtet sich die Auswahl der zu sichernden Dateien. Beim allerersten Backup muß der Parameter '-a' angegeben werden. Die Angabe des Backup-Devices (/dev/tape) muß eventuell an die lokalen Gegebenheiten angepasst werden. Die Angabe '-depth' beim find-Kommando sorgt dafür, dass die Dateien "von unten" her bearbeitet werden (nötig für das cpio-Kommando).

```
#!/bin/sh
# Tägliches Backup, als Parameter wird ein
# Verzeichnis angegeben
if [ $# -eq 0 ] ; then
    echo "Aufruf: $0 [-a] <directory>"
    echo "-a Alles sichern (sonst inkrementell)"
    exit
fi
echo "\nBand einlegen und [RETURN] druecken!"
read DUMMY
if [ "$1" = "-a" -o "$1" = "-A" ] ; then
    if [ -d "$2" ] ; then
        echo "Komplett-Backup von $2 ..."
        MARKER=$2/.lastbackup
        find $2 -depth -print | cpio -ovc >/dev/tape
        touch $MARKER
        echo "Fertig!"
    else
        echo "$2 ist kein Verzeichnis!"
        exit
    fi
else
    if [ -d "$1" ] ; then
        echo "Inkrementeller Backup von $1 ..."
        MARKER=$1/.lastbackup
        find $1 -newer $MARKER -print | cpio -ovc >/dev/tape
        touch $MARKER
        echo "Fertig!"
    else
        echo "$2 ist kein Verzeichnis!"
        exit
    fi
fi
echo "\nBand herausnehmen\n"
```

Das Zwischen-Backup

Manchmal ist es neben dem "normalen" Backup auf Band auch günstig, ein weiteres Backup auf der Platte anzulegen - z. B. für den Fall, dass jemand irrtümlich eine Datei löscht. Dieses kann dann mit ein paar Tastenbetätigungen wieder hervorgezaubert werden. Das folgende Skript sichert alle Dateien des WWW-Servers, die seit der letzten Sicherung hinzugekommen sind. Als Zeitmarkierung dient das Datei-Zugriffsdatum der Datei ".lastcheck". Bei jedem Backup wird das Datum der Datei per touch-Befehl aktualisiert. Man sieht auch schön, wie sich das 'date'-Kommando zum Erzeugen von eindeutigen Dateinamen verwenden lässt. Statt die Ausgabe zu unterdrücken, könnte man sie auch per

Mail an den WWW-Admin schicken (dann sollte man tar aber mittels Parameter etwas auskunftsfreudiger anwenden).

```
#!/bin/sh
#
# Inkrementelles Sichern aller Dateien des WWW-Servers
#
{
  TMPFILE="/tmp/check.$$"
  TIMESTAMP=`date +%y%m%d`
  DIRECTORY="/home/httpd/htdocs"
  WWWARCHIVE="/home/wwwarchive"
  cd $DIRECTORY
  find . -newer .lastcheck -print >$TMPFILE 2>/dev/null
  touch .lastcheck
  if [ `cat $TMPFILE | wc -l` -gt 0 ]
  then
    tar cf /$WWWARCHIVE/backup.$TIMESTAMP.tar $DIRECTORY
    gzip /$WWWARCHIVE/backup.$TIMESTAMP.tar
    chown wwwadm.staff /$WWWARCHIVE/backup.$TIMESTAMP.tar.gz
    chmod 660 /$WWWARCHIVE/backup.$TIMESTAMP.tar.gz
  fi
  rm $TMPFILE
} > /dev/null 2>&1
```

eval-Anwendung

Endlich eine Anwendung für das „eval“-Kommando, auf die jeder schon gewartet hat. Das folgende Fragment zeigt, wie man in der Bourne-Shell die Ausgabe des aktuellen Verzeichnisses im Prompt realisieren kann. Einziger Nachteil: Zum Logoff muss hier noch <Ctrl>+<C> und <Ctrl>+<D> gedrückt werden.

```
while true ; do
  echo "`pwd`: $PS1\c"
  read KDO
  eval $KDO
done
```

Ausgaben aus cron- und at-Jobs auf ein Terminal

Wie sendet man aus einem 'cron'- oder 'at'-Job etwas an den Benutzer, sofern er eingeloggt ist? Das Problem besteht darin, dass der Job nicht wissen kann, an welchem Terminal der User sitzt. Also muss zunächst per 'who'-Kommando ermittelt werden, ob der Adressat eingeloggt ist und an welchem Terminal er sitzt. Dann kann eine Nachricht nach folgendem Schema an den User geschickt werden.

```
#!/bin/sh
# Nachricht ($2-$nn) an User ($1) senden, sofern dieser
#eingeloggt ist
NAM="$1"
shift
MSG="$@"
if who | grep -q $NAM ; then          # User eingeloggt?
  write $NAM < $MSG
fi
```

Rundruf

Nach dem gleichen Schema kann man ein "wall für Arme" realisieren. Es werden jedoch im Gegensatz zum "echten" 'wall' nur die Benutzer erreicht, die ihren Mitteilungsempfang offen haben.

```
who | while read USR REST ; do      # für alle aktiven User
  banner "Teatime!" | write $USR
done
```

Datensicherung und -wiederherstellung mit BorgBackup

Mit „borg“ liegen Ihre Dateien „geborgen“ im sicheren und schnellen Backup!

- [!BorgBackup Projektseite](#)
- [BorgBackup Downloads](#)
- [BorgBackup Dokumentation](#)