# Das Backup-Karussell dreht sich immer noch

# 20 Jahre Datensicherung mit rsync


Hans-Jürgen Beie
hjb@pollux.franken.de

# Motivation und Historie

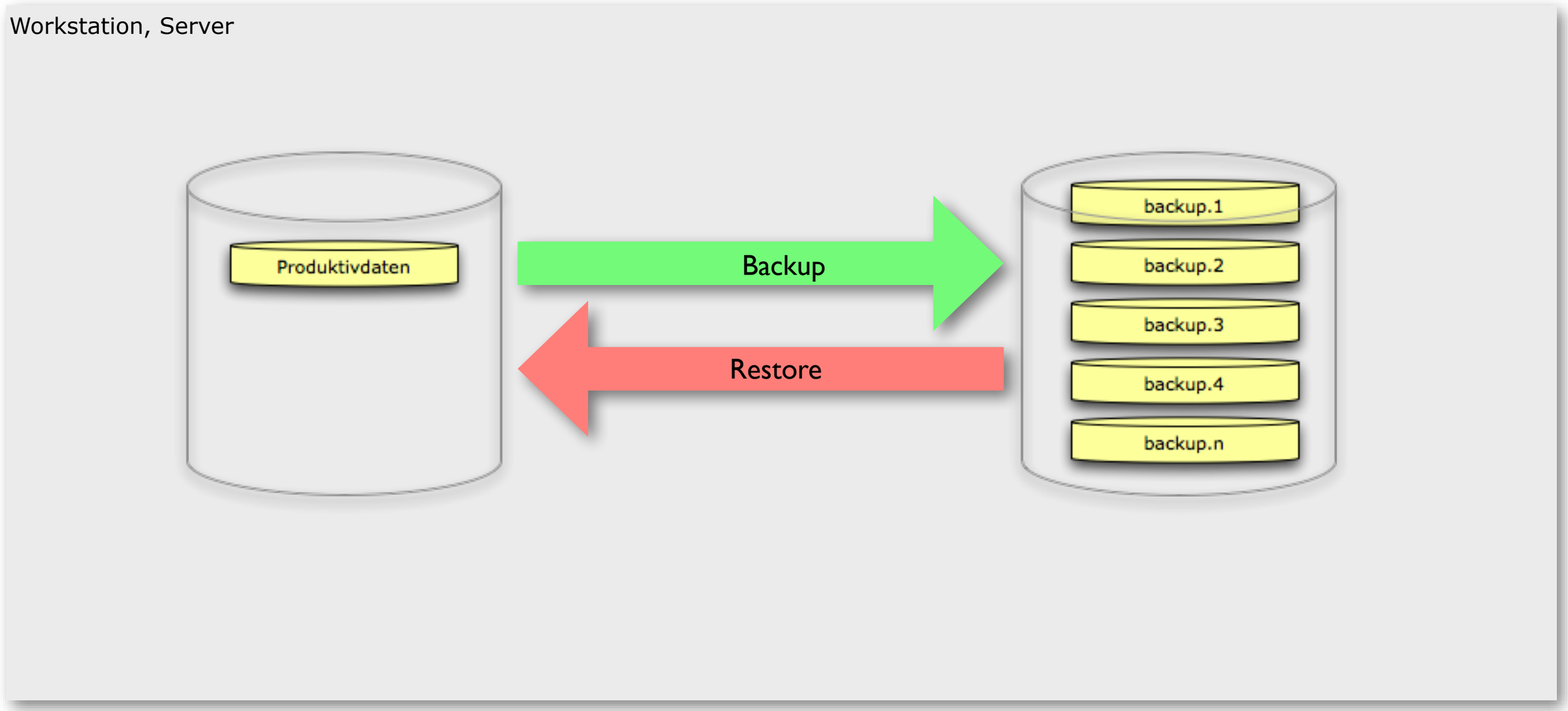
## Um die Jahrtausendwende - Wie alles anfing

- Ich wollte große Datenbestände (damals ein paar Hundert MB :-) von einem Linux-System auf einen anderen Server sichern.
- Das sollte schnell, platzsparend und **automatisch** geschehen.
- Die Backups sollten über mehrere Tage und Wochen zurück reichen.
- Ohne spezielle Software oder Hardware und einfach zu handhaben.
- Ich stieß auf den Ansatz von Mike Rubel
  „Easy Automated Snapshot-Style Backups with Linux and Rsync“
  http://www.mikerubel.org/computers/rsync_snapshots/
  und habe damit experimentiert. Das gefiel mir gut!

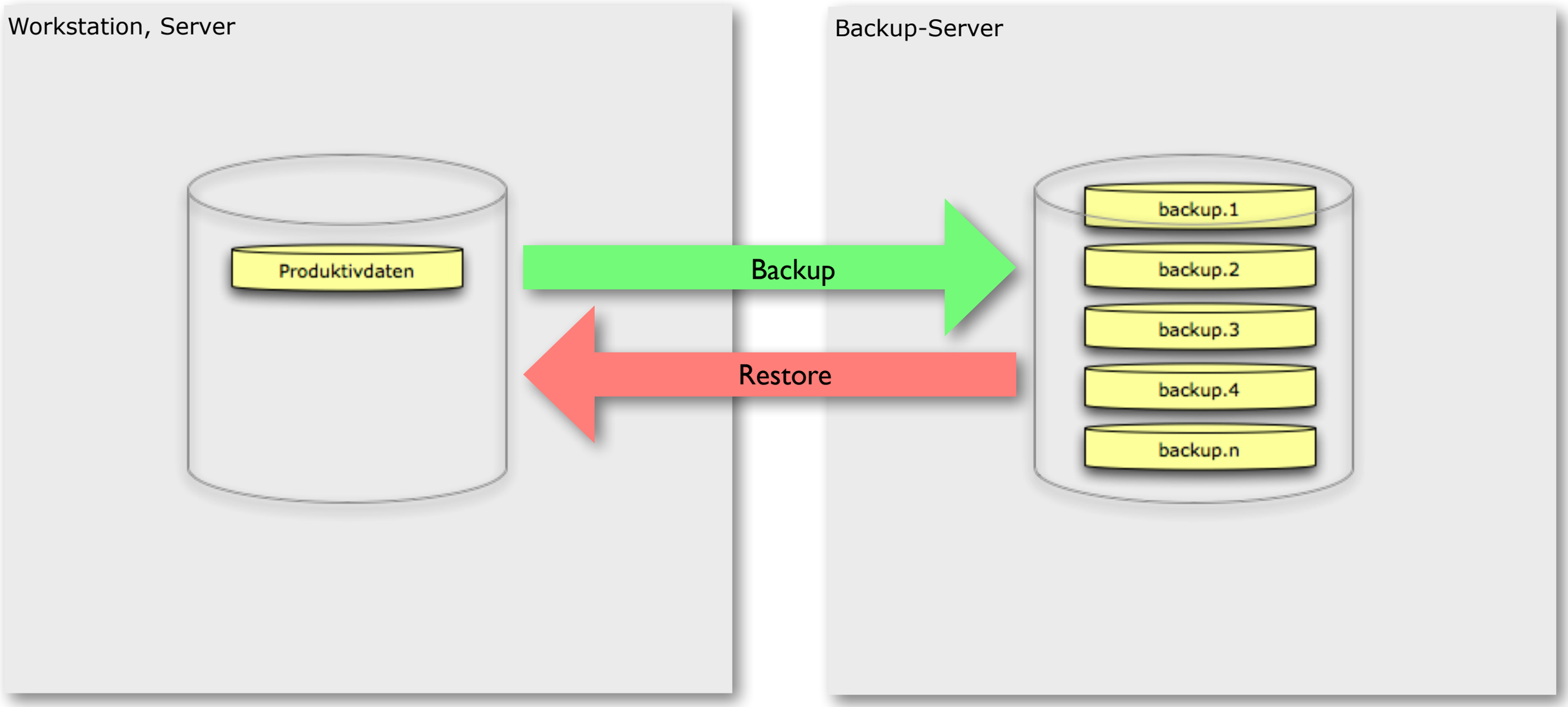
## Das Backup-Karussell - rsback

- Nicht lange vorher hatte ich begonnen, mit Perl zu programmieren.
  Jetzt hatte ich ein sinnvolles Projekt zum Üben :-)
- Daraus ist **`rsback`** entstanden.
- Das Backup-Karussell dreht sich nach mehr als 20 Jahren immer noch, inzwischen mit ein paar Hundert GB pro Installation (auch beim KNF).

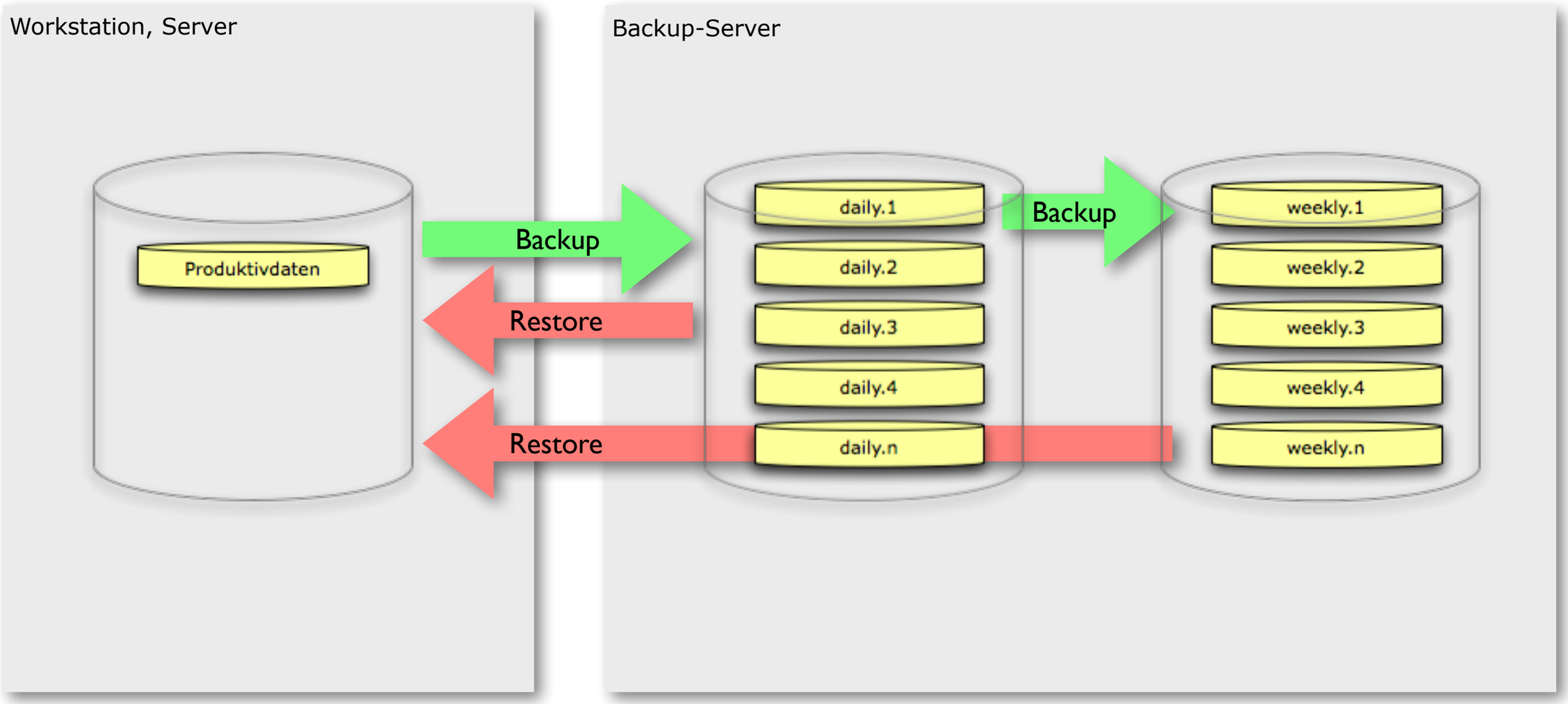
# Szenario 1 — Rechner mit zweiter Backup-Platte / -Partition



# Szenario 2 — Produktivsystem und Backup-Server



# Szenario 3 — Produktivsystem und Backup-Server mit zwei oder mehr Archiv-Linien



# Grundlagen — hard links

## Was ist ein „hard link“?

- Eine Datei wird im File-System durch einen Block an Informationen - den so genannten **inode** (index node) - repräsentiert.
- Ein **hard link** ist ein Zeiger auf eine Datei. Ein hard link zeigt auf einen inode und gilt nur innerhalb ein und desselben File-Systems (Partition).
- Jeder Dateiname ist ein **hard link**!

## Platz sparen mit hard links

- Mit dem Link-Kommando

```
ln backup.2/kongress23.knf backup.1/kongress23.knf
```

  wird der Datei `backup.1/kongress23.knf` ein neuer (zusätzlicher) Bezeichner `backup.2/kongress23.knf`, also ein weiterer **hard link**, zugewiesen.

- Das ist physikalisch immer noch **eine** Datei, nur ist sie mit dem gleichen Namen in zwei unterschiedlichen Verzeichnissen verfügbar.

## Kopieren mit hard links

- Das `cp`-Kommando (die GNU-Variante!) kann man mit der Option `--link` „virtuelle Duplikate“ von ganzen Verzeichnisbäumen anlegen:

```
cp --link /arc/backup.1 /arc/backup.0
```

**Wozu ist rsync gut?**

- Daten in einem Zielverzeichnis rekursiv mit den Daten aus einem Quellverzeichnis synchronisieren:

```
rsync --archive --delete source_dir destination_dir/
```

  Dabei werden nur die Dateien kopiert, die im Zielverzeichnis nicht vorhanden sind oder sich von denen im Zielverzeichnis unterscheiden. Dateien, die im Zielverzeichnis vorhanden sind aber keine Gegenstücke auf Seiten der Quelle haben, werden gelöscht (`--delete`).

- Das Ganze funktioniert auch zwischen zwei Rechnern. So lässt sich ein lokales Verzeichnis mit dem Inhalt eines Verzeichnisses auf einem anderen Rechner synchronisieren:

```
rsync --archive --delete remote.host:/source /backup/
```

**rsync und hard links**

**rsync** kann Verknüpfungen mit **hard links** auf der Zielseite selbst erzeugen (`--link-dest`):

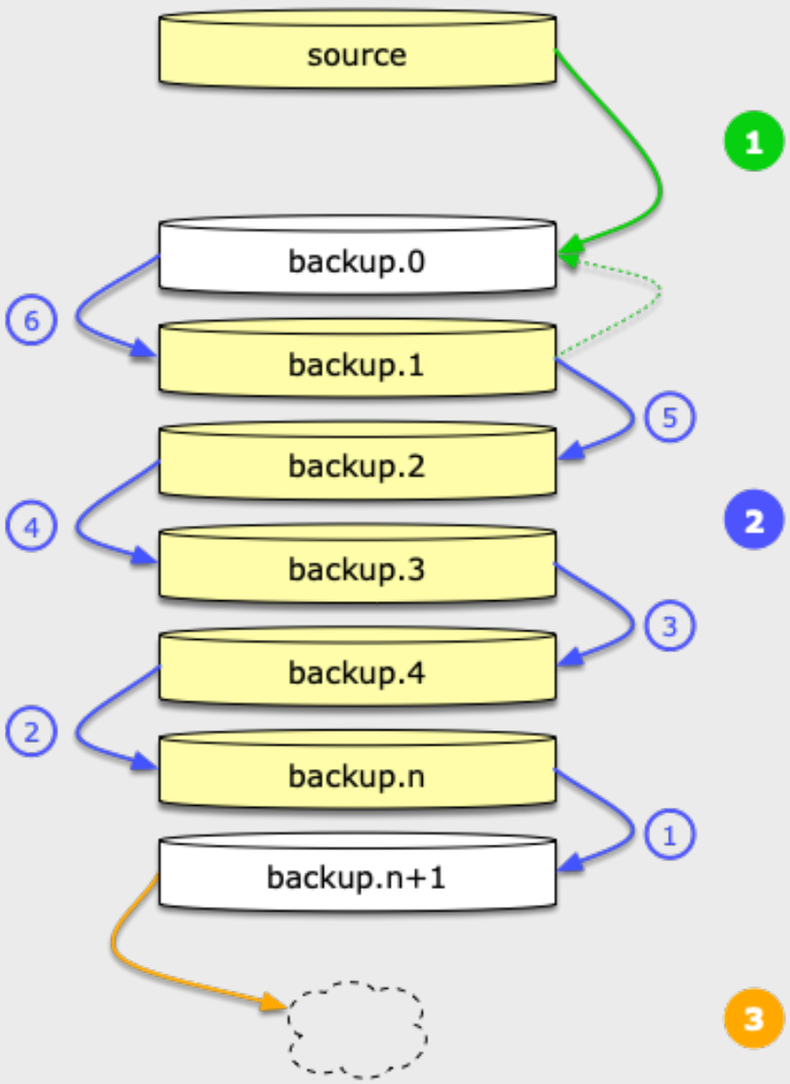
- Daten im Zielverzeichnis `backup.0` mir denen im Quellverzeichnis **`source`** auf einem anderen Rechner synchronisieren und dabei automatisch hard links zu gleichen (unveränderten) Dateien in einem zweiten Referenzverzeichnis `backup.1` anlegen:

```
rsync --archive --delete --link-dest=/arc/backup.1 remote.host:/source /arc/backup.0/
```

- Das ist äquivalent zu:

```
cp -al /arc/backup.1 /arc/backup.0
rsync --archive --delete remote.host:/source /arc/backup.0/
```

# Datensicherung mit rsync und hard links — Funktionsschema




## Sicherung von Produktivdaten in einem „Karussell“ von n rotierenden Backups

1 Aus der Quelle durch Verlinken mit dem letzten **`backup.1`** ein neues **`backup.0`** anlegen:

```
rsync -a --delete --link-dest=backup.1 source backup.0/
```

2 Backup-Verzeichnisse durch Umbenennen rotieren:

```
mv backup.n backup.n+1
...
mv backup.1 backup.2
mv backup.0 backup.1
```

3 Überzähliges Verzeichnis entfernen:

```
rm -rf backup.n+1
```

# Systemvoraussetzungen und Werkzeugkasten

**Was braucht man dazu?**

- Backup-Server: Betriebssystem auf Unix-Basis, z.B. Linux, xBSD, macOS
- GNU File Utilities: Bei BSD-basierten Systemen extra installieren!
  *http://www.gnu.org/software/fileutils/fileutils.html*
- rsync: Bei Unixen meistens schon im Bordwerkzeug
  *http://rsync.samba.org*
- Windows (alt) als Produktivsystem: cygwin, cwRsync
  *http://cygwin.com*, *http://www.itefix.no/cwrsync*
- Root/Admin-Rechte für Quelle und Ziel, wenn alle Dateiattribute restaurierbar sein sollen.

**Damit sich das Karusell auf möglichst von alleine dreht:**

- Der Ausgangspunkt von vielen Lösungen: Shell-Skripte von Mike Rubel (proof of concept)
  *http://www.mikerubel.org/computers/rsync_snapshots/*
- rsnapshot (Perl)
  *https://rsnapshot.org*
- Snapshot-Backups mit Rsync
  *https://www.linux-magazin.de/ausgaben/2004/09/unwetterzentrale/*
- rsback (Perl)
  *https://www.pollux.franken.de/hjb/rsback/*

! *Unvollständige Auswahl*

**Warum?**

Aufteilen von Backups in kleinere „Happen“ ist ratsam, denn...

- Rsync ist CPU- und speicherhungrig
- der Zugriff auf das „backup repository“ muss u.U. mit unterschiedlichen Rechten geregelt werden
- bei Backup-Fehlern soll der „Schaden“ begrenzt werden
- bei (remote) Backups sind u.U. mehrere Anläufe notwendig

Aufteilen bedeutet aber: viele Skripte erstellen und pflegen. Deshalb ...

**rsback**

- bietet für typische, immer wiederkehrende Aufgaben eine einfach zu konfigurierende Verwaltung
- hilft beim Aufteilen von umfangreichen Backups in handliche Pakete (→ tasks)
- startet und überwacht rsync bei den verschiedenen Aufgaben
- dreht das „Karussell“
- sorgt bei Problemen für die Schadensbegrenzung
- liefert kompakte Logs für vielbeschäftigte Admins

# rsback

**besteht aus ...**

- einem Perl-Skript
- einer Konfigurationsdatei (oder auch mehreren)
- optionalen „exclude lists“ für rsync, um Verzeichnisse oder Dateien vom Backup auszuschließen

**wird gestartet ...**

- über die Konsole

```
usage: rsback [options] backup-tasks(s)
Make a backup of one or more backup tree(s). The backup tasks itself
are described in the configuration file
   /etc/rsback/rsback.conf

options
-h             Display this help
-v             Be verbose
-d             Debug mode (more verbose, runs rsync with option --dry-run)
-i             Init backup repositories only (deprecated)
-c conf-file   Configuration file other than /etc/rsback/rsback.conf
```

- oder über cron jobs

```
11 01 * * 1-7 /usr/local/sbin/rsback -v work-daily  >>/var/log/rsback/work-daily.log
11 05 * * 7   /usr/local/sbin/rsback -v work-weekly >>/var/log/rsback/work-weekly.log
```

**Gobale Parameter für rsback**

In der Sektion **[global]** werden globale Parameter festgelegt, hier die wichtigsten:

Wo sind welche Programme zu finden?

Welche Aufgaben (tasks) gibt es?

Verzeichnis für Lock-Files

Optionen für rsync

Ausschlussliste

Soll rsync selbst hard links anlegen?

Welche „Fehler", die rsync meldet, sind weniger dramatisch?

! *Es gibt noch einige andere Parameter mehr.*

im Stil von Windows INI-Dateien oder rsyncd.conf

```
# rsback.conf
# ----------------------------------
[global]
rsync_cmd = /usr/bin/rsync
cp_cmd = /bin/cp
mv_cmd = /bin/mv
rm_cmd = /bin/rm
mkdir_cmd = /bin/mkdir

tasks = work-daily work-weekly

lock_dir = /var/lock

rsync_options = -a --delete \
   --delete-excluded --numeric-ids

exclude_file = /etc/rsback/rsback.exclude

use_link_dest = yes

ignore_rsync_errors = 24

# ----------------------------------
[work-daily]

...
```

## Die tägliche Arbeit: task im rsync-Modus

Für jede Aufgabe (task) muss in einer Sektion festlegt werden, was zu tun ist.

Was soll gesichert werden?

In welches Verzeichnis soll gesichert werden?

rsync soll verwendet werden

Wir wollen 14 Archive, die mit **`daily.1`** bis **`daily.14`** benannt sein sollen.

Ausschlussliste

Optionen für rsync
Hier etwas üppiger bestückt, weil wir per ssh über eine DSL-Verbindung mit Bandbreiten-begrenzung transferieren wollen.

Damit uns dieser Job nicht in die Quere kommt, sperren wir ihn.

! *Parameter in **`[<task>]`** können*
*Parameter in **`[global]`** überschreiben.*

```
...
# ----------------------------------
[work-daily]

source = rsback@remote.host:/var/work/

destination = /BACKUP/remote-host


mode = rsync

rotate = daily 14


exclude_file = /etc/rsback/work.exclude

rsync_options = -a --safe-links --delete\
    --delete-excluded --stats \
    --numeric-ids --bwlimit=48 \
    -e "ssh -i rsback.id_rsa" \
    --rsync-path='sudo rsync'

lock = work-weekly

# ----------------------------------
[work-weekly]
```

**Einmal pro Woche: task im link-Modus zum Fixieren der letzten Tagessicherung**

Für jede Aufgabe (task) muss in einer Sektion festlegt werden, was zu tun ist.

Wir „duplizieren“ die letzte Tagessicherung ...

... ins gleiche Verzeichnis. (dort wird daraus **`weekly.1`**)

Diesmal einfach mit hard links, dazu brauchen wir rsync nicht.

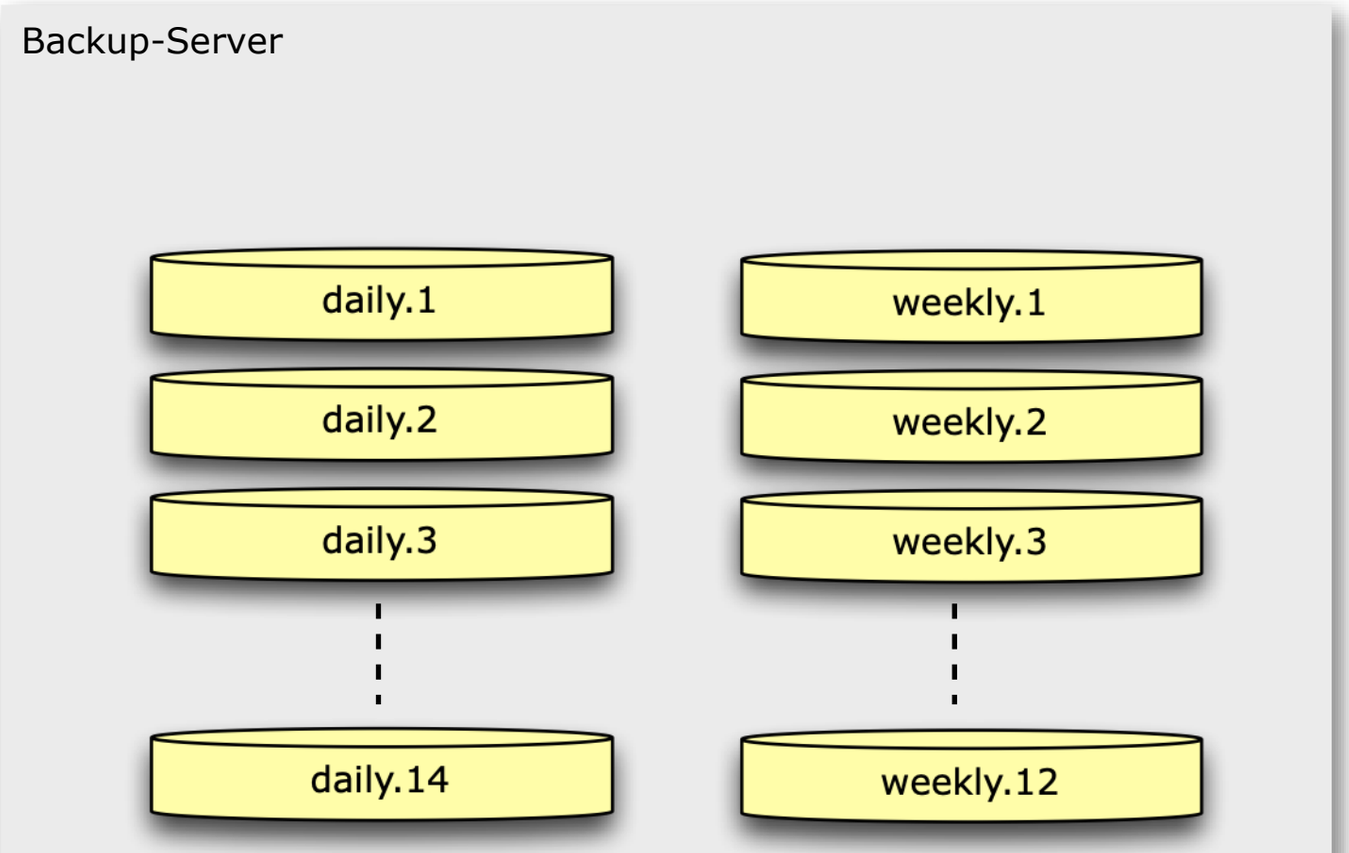
Wir heben 12 Archive auf, die mit **`weekly.1`** bis **`weekly.12`** benannt sind.

```
...
# ----------------------------------
[work-weekly]

source = /BACKUP/remote-host/daily.1/

destination = /BACKUP/remote-host

mode = link

rotate = weekly 12
```

! *Archive müssen nicht **`daily`** oder **`weekly`** heißen. Der Name ist frei wählbar.*

## 12 Wochen später ...

```
# tree /BACKUP/remote-host/work -L 1
/BACKUP/remote-host/work
|-- daily.1
|-- daily.10
|-- daily.11
|-- daily.12
|-- daily.13
|-- daily.14
|-- daily.2
|-- daily.3
|-- daily.4
|-- daily.5
|-- daily.6
|-- daily.7
|-- daily.8
|-- daily.9
|-- history.daily
|-- history.weekly
|-- weekly.1
|-- weekly.10
|-- weekly.11
|-- weekly.12
|-- weekly.2
|-- weekly.3
|-- weekly.4
|-- weekly.5
|-- weekly.6
|-- weekly.7
|-- weekly.8
`-- weekly.9
```



# rsback — Daten wieder restaurieren

Der Unfall:

```
hjb@pollux:/work$ ls knf/kongress-2023
ls: knf/kongress-2023: No such file or directory
```

Ich habe vor ungefähr zwei Wochen meine Daten zerstört!
Was nun?

Mit rsync vom Backup-Server wieder zurück holen:

```
# rsync -a hjb@backup.server/BACKUP/work/weekly.2/knf/kongress-2023 /work/knf/
```

Wenn die Archive lokal verfügbar sind, geht es mit einer einfachen Kopieraktion:

```
# cp -a /BACKUP/work/weekly.2/knf/kongress-2023 /work/knf/
```

Reloaded!

# rsback — Beispiele vom KNF-Backup-System

Log-Schnipsel eines Backups von cloud.franken.de (Volumen: 438 GB, inodes used: 861K)

```
### processing module 'cloud'
# running daily task
#    rsback: [2023-11-21 00:16:17] rsback-0.7.0 (hjb -- 2021-10-24)
#    rsback: processing 1 tasks: 'cloud3-daily'
#    rsback: [2023-11-21 00:16:17] task 'cloud3-daily'
#    rsback: [2023-11-21 00:16:17] processing backup task 'cloud3-daily'
#    rsback: [2023-11-21 00:22:19] backup task 'cloud3-daily' done, runtime: 00:06:02.
#    rsback: [2023-11-21 00:22:19] total runtime: 00:06:02
#    rsback: [2023-11-21 00:22:19] Thank you for making a simple program very happy.
```

Log-Schnipsel eines Backups von mail-n.franken.de (Volumen: 504 GB, inodes used: 8,2M)

```
### processing module 'mail-n'
# running daily task
#    rsback: [2023-11-21 00:22:19] rsback-0.7.0 (hjb -- 2021-10-24)
#    rsback: processing 1 tasks: 'mail-n-daily'
#    rsback: [2023-11-21 00:22:19] task 'mail-n-daily'
#    rsback: [2023-11-21 00:22:19] processing backup task 'mail-n-daily'
#    rsback: [2023-11-21 00:59:17] backup task 'mail-n-daily' done, runtime: 00:36:58.
#    rsback: [2023-11-21 00:59:17] total runtime: 00:36:58
#    rsback: [2023-11-21 00:59:17] Thank you for making a simple program very happy.
```

# rsback — Datensicherung übers Netz

**Probleme**

- Grundsätzlich muss man davon ausgehen, dass Viele „mitlesen“ können.
- Mindestens die brisanten Daten müssen verschlüsselt übetragen werden.
- Zur Sicherung/Restaurierung von Daten braucht man i.d.R. Administrationsrechte. Der Remote-Zugriff auf ein System mit administrativen Rechten birgt aber zusätzliche Gefahrenquellen.
- Bei der Übertragung übers Netz muss immer ein Kompromiss zwischen **Datensicherheit** und und Funktionalität der **Datensicherung** eingegangen werden.

**Auswege**

- rsync bietet die Möglichkeit Daten über ssh zu übertragen.
- Root-Login über ssh ist bei den meisten Admins nicht gerne gesehen.
- Einschränkung der Root-Rechte auf das rsync-Kommando ist möglich, aber nicht einfach zu überwachen (ssh wrapper).
- **rsyncd** (Server-Modus) **kombiniert mit einem Tunnel** (z.B. OpenVPN) ist relativ einfach zu konfigurieren und hält rsync-Kommandos übersichtlich.

  rsback ist für beide Möglichkeiten offen!

# Ende

Danke für Eure Geduld!