Ein vollständiges Backup

- LIT 2016
  - rsync, SSH, LUKS im Team
  - Ergebnis:
    - ‚rsnapshot' ist gut für ein Backup
    - Historie über Hardlinks
    - Schnell, platzsparend,
    - schwer wartbar, komplex
- LIT 2018
  - Sicheres privates Netz in unsicheren Zeiten
  - SSH
    - universelle sichere Verbindung
- LIT 2023
  - Warum sollten wir BTRFS verwenden?
  - Unterbau für ein Backup
- heute
  - ein komplettes Backup
- Demo am Ende des Vortrages

Physiker in Halle
Fernstudium Theologie
seit 1988 in München
Physiker am Max-Planck-Institut
IT Berater und Entwickler
Senior in Görlitz
Vorträge in
Augsburg, Graz, Dornbirn,
Wien, Kiel, Chemnitz, ...

einfaches Backup (privates Foto)

Backup-Programme

- eine Schicht mehr
  - weniger Kontrolle, auf Hersteller angewiesen
  - Updates, Datenformat, Historie?
  - Kosten, Wartung?
  - Verschlüsselung, SSH?
  - remote Entkopplung?
  - 3-2-1 Lösung?
  - Cloud?

- wir haben Linux, wir brauchen das nicht
  - rsync, diff, SSH, LUKS, gawk, rsnapshot, tail, cat, vim, ...

- was nicht in einem Backup-Programm sein sollte
  - Verschlüsselung
    - bessere Kontrolle
  - Kompression
    - Aufgabe des Filesystems
  - Deduplizierung
    - Aufgabe des Filesystems
  - Cloud
    - steht im Notfall nicht zur Verfügung

Linux-Kenntnisse nötig

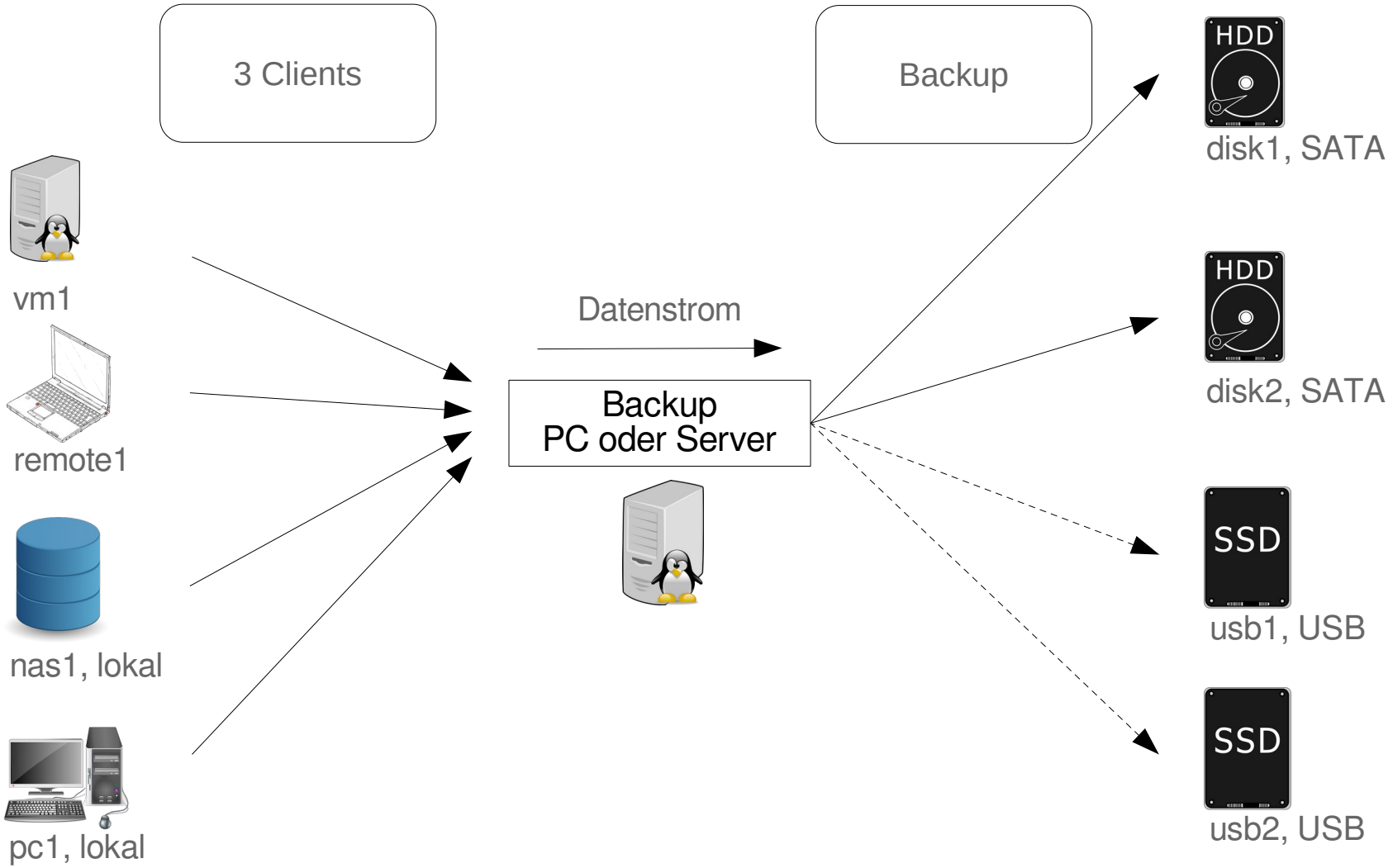
- Warum?
  - zur Navigation im Filesystem
  - zum Testen
  - für Fehlersuche
  - zum Verifizieren
  - Lösungen finden

- feststellen, was in das Backup soll und was nicht
  - Filetree kennen
  - Rechte kennen
  - lokal und remote (SSH)
  - Snapshots mit Btrfs, z.B. bei laufendem System
  - zusätzliche Scripte vorher/nachher
    - z.B. zur Aufbereitung von Daten
  - Vorarbeit kostet viel Zeit
  - Testdaten anlegen, ca. 1GB

- ein **‚one click‘** und alles ist erledigt gibt es nicht

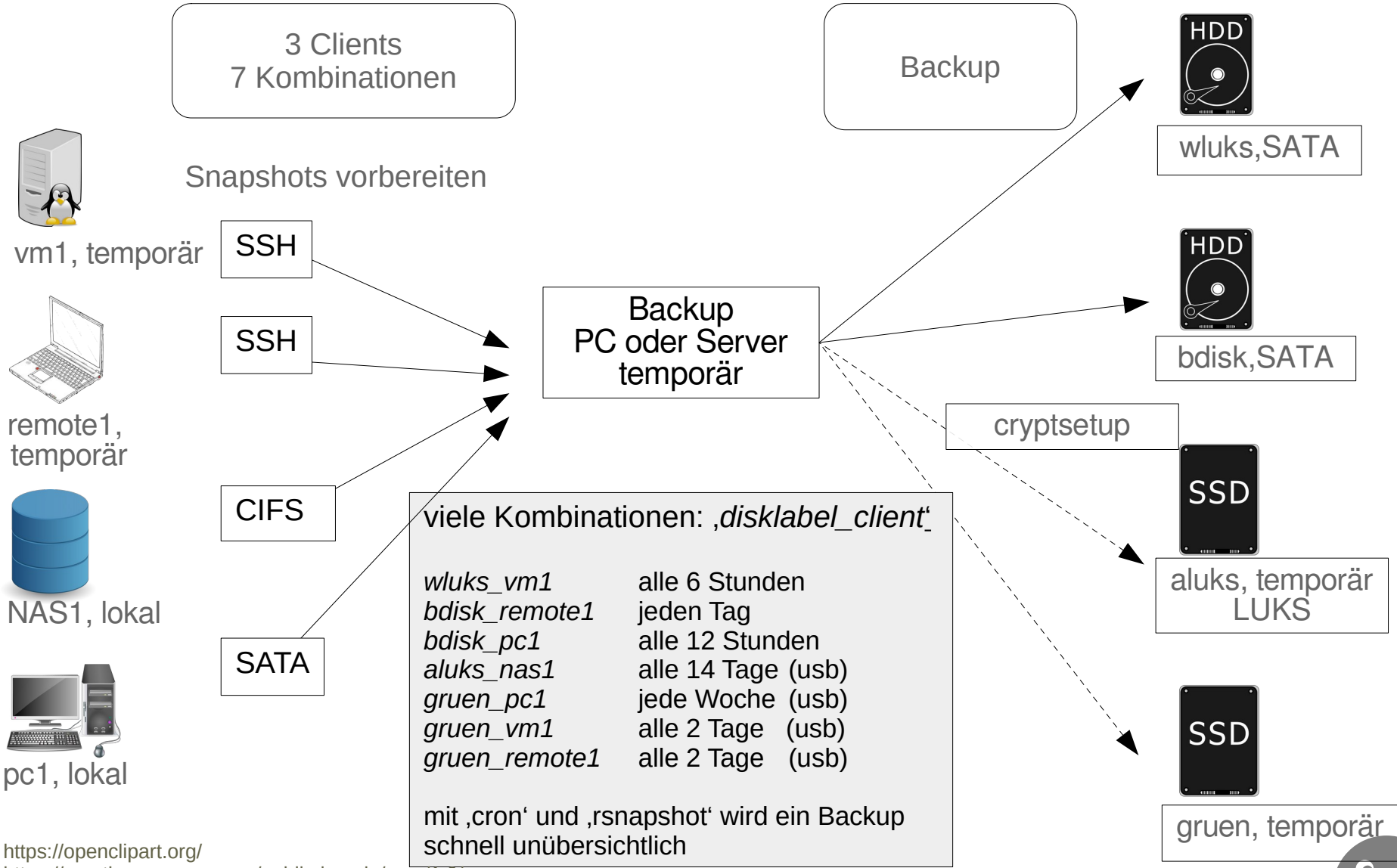
  - Was ist, wenn es nicht ‚erledigt’ ist?
  - Wie kann man das feststellen?

Backup ist ganz einfach …



https://openclipart.org/
https://creativecommons.org/publicdomain/zero/1.0/

oder doch nicht ...



https://openclipart.org/
https://creativecommons.org/publicdomain/zero/1.0/

zur Erinnerung: Layout ‚rsnapshot'

Jedes Verzeichnis auf der Backup-Festplatte enthält das vollständige Backup eines oder mehrerer Verzeichnisbäume der Festplatte eines Clients.

‚*disklabel_client*' Kombination dient der individuellen Konfiguration und Unterscheidung

dserver
- drei.0
- drei.1
- drei.2
- drei.3
- eins.0
- eins.1
- eins.2
- eins.3
- eins.4
- vier.0
- zwei.0
- zwei.1
- zwei.2

svrleo/
- drei.0
- drei.1
- drei.2
- drei.3
- drei.4
- eins.0
- eins.1
- eins.2
- eins.3
- eins.4
- vier.0
- vier.1
- vier.2
- vier.3
- vier.4
- zwei.0
- zwei.1
- zwei.2
- zwei.3

nc/
- drei.0
- drei.1
- drei.2
- drei.3
- drei.4
- eins.0
- eins.1
- eins.2
- eins.3
- eins.4
- vier.0
- zwei.0
- zwei.1
- zwei.2
- zwei.3
- zwei.4
- zwei.5
- zwei.6

/mnt/fdisk/rs/pserver/
- drei.0
- drei.1
- drei.2
- drei.3
- drei.4
- eins.0
- eins.1
- eins.2
- eins.3
- eins.4
- vier.0
- vier.1
- vier.2
- vier.3
- vier.4
- zwei.0
- zwei.1
- zwei.2
- zwei.3
- zwei.4

Was muss getan werden?

- Clients
  - verfügbar?
- Backup-Festplatten
  - mount/umount, luks, usb, persistent oder umount
- Backup läuft durch
  - start/stop, wie?
  - alle Projekte werden sequentiell abgearbeitet
- Verwaltung der Zustände von ‚rsnapshot‘
  - Aufrufe der Intervalle
- Transparenz
  - leicht konfigurierbar
  - deutliche Fehlermeldungen
- Logfiles
  - umfangreich
  - mit Linux-Kenntnissen analysierbar

rsnapshot als cronjob
1 Szenario, umständlich

/etc/cron.d/rsnapshot

```
0 */4 * * *       root  /usr/bin/rsnapshot alpha
30 3      * * *         root  /usr/bin/rsnapshot beta
0  3      * * 1         root  /usr/bin/rsnapshot gamma
30 2      1 * *         root  /usr/bin/rsnapshot delta
```

Wirklichkeit:
7 Szenarien
mit unterschiedlichen Zeiten und Erreichbarkeiten

wluks_vm1	alle 6 Stunden
bdisk_remote1	jeden Tag
bdisk_pc1	alle 12 Stunden
aluks_nas1	alle 14 Tage (usb, temporär)
gruen_pc1	jede Woche (usb)
gruen_vm1	alle 2 Tage (usb)
gruen_remote1	alle 2 Tage (usb)

Design, so einfach, wie möglich

- Zitat
  - „Man muß die Dinge so einfach wie möglich machen. Aber nicht einfacher.“
    - Albert Einstein Quelle

- bash, weil überall vorhanden
  - keine ‚export‘ Variable
  - Namespaces erfinden
  - alle Linuxtools verfügbar

- alles in einem Verzeichnis
  - einfach zu kopieren
  - keine anderen Tools

- Konfiguration
  - bash-Syntax, mit ‚source‘ einbinden
  - Verwaltung der Zeiten, wann und wann nicht
  - ‚mount‘/‘umount’ einstellen
  - ‚rsnapshot’, original Konfiguration
    - angepasst, Struktur nicht geändert
- Ablauf
  - sequentiell
  - ‚*disklabel_client*‘ Kombination zur Steuerung
  - jede Stunde Zustand prüfen
  - Backup nach Zeitplan ausführen

```
# prefixes of variables in backup:
# bv_*  - global vars, all files
# lv_*  - local vars, global in file
# lc_*  - local constants, global in file
# _*    - local in functions or loops
# BK_*  - exitcodes, upper case, BK_
# cfg_*  - set in cfg.* file_
```


Lösung (pseudo code)

```
start

    some checks

    while true
    do
        begin script

        for each disk
        do
            lookup for dirty projects
                end for loop, if nothing to do

            check and mountHD,

            disk begin script

            for each dirty project at disk
            do
                project begin script
                    do rsnapshot
                project end script
            done

            disk end script

            umount HD
         done

     success message
     end script

     release_lock
     check stop  → end
   done
```


Tests und Logs

- Backup läuft wochenlang durch
- Kontrolle mit Linuxmitteln, tail, cat, grep
  - allgemeiner Logfile, mit ‚./tt.sh‘ (ist intern ‚tail-f‘)
  - rsnapshot-Logfile, mit ‚tail -f‘
  - rsync-Logfile, mit ‚tail -f
- Fehlersuche in den Logs
  - Hinweise in den Logfiless selbst
  - Linuxkenntnisse nötig
- Erreichbarkeit der Clients testen
  - über SSH und Keys
- Betriebssicherheit
  - Festplattenzustand
  - Füllstand der Festplatten, < 70%
  - Uptime
- Angriffssicherheit
  - SSH, nur von Backup zu Client
  - Rechte

Installation

- Copy von Github in das Arbeitsverzeichnis
  - z.B. nach ‚/usr//locl/bin/backup_data‘
- Konfiguration nach Beispiel
  - nach der Regel ’disklabel_client’
  - ‚cfg.projects’
  - ‚cfg.sshlogin‘
  - ‚cfg.successloglines’
- Backup-Festplattenverzeichnisse anlegen
  - Label vergeben, UUID eintragen
- ‚rlf_backup_data.rc‘ anpassen
- in „./conf‘
  - rsnapshot Konfigurationen ablegen
- in „./exclude‘
  - Excludes für rsync
- in „./pre‘
  - Verfügbarkeit der Clients testen

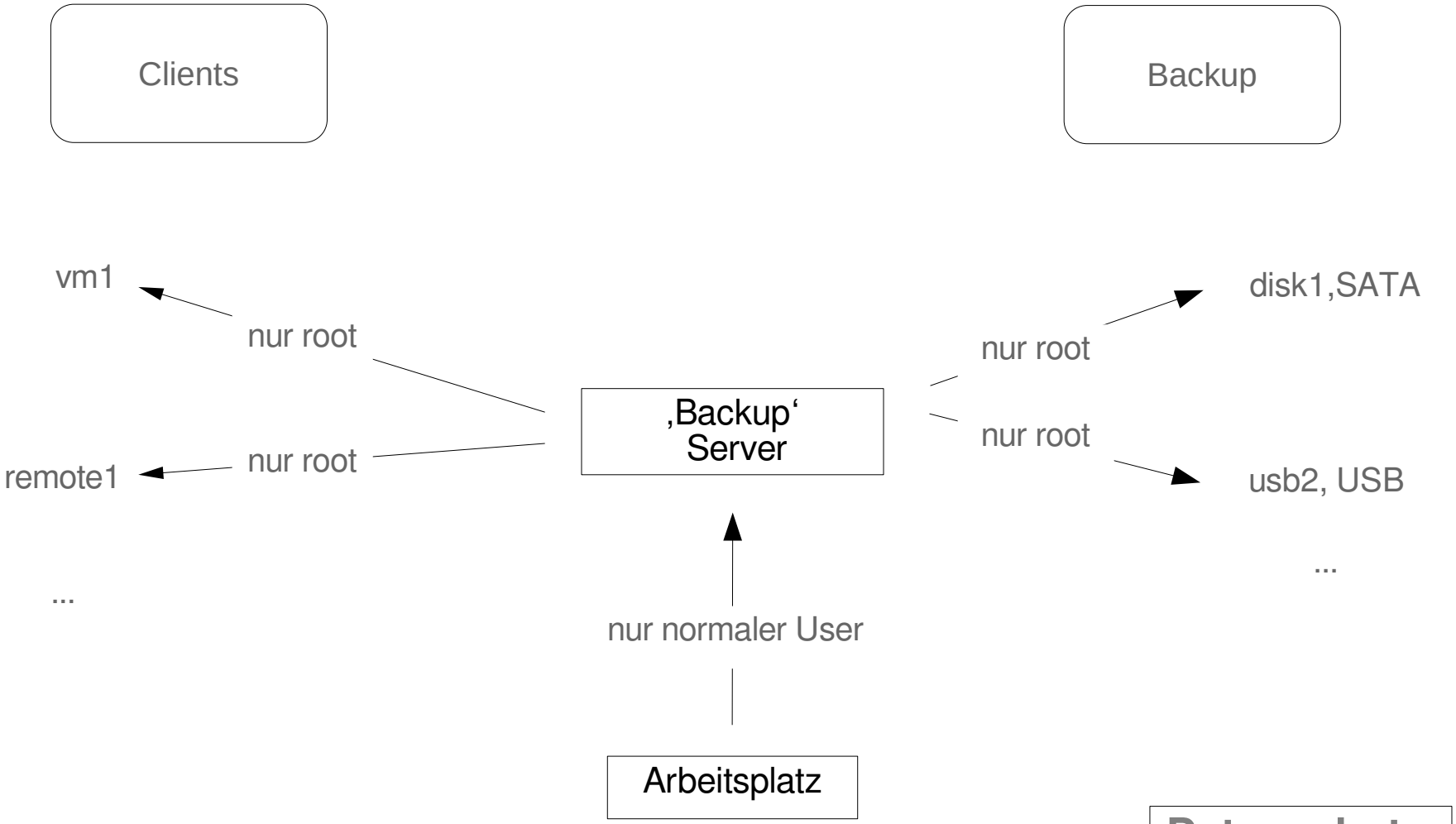
Beispiel: cfg.projects

```
DISKLIST="nbaluks gruen"

# sub projects per disk
a_projects['nbaluks']="nb snap"
a_projects['gruen']="nb snap"

# time intervals
a_interval['nbaluks_nb']=6:00
a_interval['nbaluks_snap']=1:00:00
a_interval['gruen_nb']=12:00
a_interval['gruen_snap']=7:00:00
```

Tests, Start/Stop

- erster Test, ‚rsync -n‘ in rsnapshot config eintragen
- ./test.sh
  - Test von ‚rsnapshot‘ Syntax
- ./show_config.sh
  - alle Einstellungen und Backup-Zeiten
- ./show disks,sh
  - Liste aller verfügbaren Backup-Festplatten
- ./show_times.sh
  - Liste der Fälligkeiten
- ./start_backup.sh oder ./cron_start_backup.sh
  - Start von Hand oder als cronjob
- ./stop.sh
  - ‚‘touch stop‘‘ stoppt das Backup, wenn möglich

Absicherung des Servers



Datenschutz!

Brandschutz, Wiederherstellung, Wartung

- doppelte Datenhaltung
  - einfach 2 Projekte mit identischen Quellen, aber verschiedenen Backends erstellen
- Wiederherstellung
  - automatisch nicht sinnvoll
  - schon kurz nach dem Backup sind Verzeichnisse nicht mehr gleich
  - unsere Linux-Kenntnisse helfen, s.o.
  - einfach die benötigten Files/Verzeichnisse aus dem Backup holen
    - diff, rsync, cat, nutzen
- Wiederherstellung System
  - wie oben, nicht sinnvoll
  - Systemausfall ist ohnehin ein größerer Schaden
  - Btrfs hilft mit Snapshots vom laufenden System
  - mit Linux-Kenntnissen kann man alles neu aufsetzen
- Wartung
  - Konfigurationen anpassen
  - start/stop wenn nötig
  - Logfiles kontrollieren
  - Füllstand der Festplatten beobachten

Demo

- 2 Platten
  - label ‚gruen‘ und ‚nbaluks‘
  - Projekte: ‚nb‘ und ‚snap‘
  - Disk_client Kombinationen
    - 'nbaluks_nb
    - 'nbaluks_snap'
    - 'gruen_nb'
    - 'gruen_snap'
- PC zu Hause mit vielen Projekten
- Backup-Programm kann mitgenommen werden
  - vieles steht in den Scripten
  - Konfiguration anschauen
  - Log anschauen
  - oder hier https://rleofield.de/backup/index.html
- Ist alles korrekt,
  - läuft das Backup im Hintergrund,
  - automatisch
  - für immer.

Danke

Take Home Message

Ein automatisches Backup mit Linux Bordmitteln ist mit **rsync, rsnapshot und LUKS** erreichbar.

Vielen Dank für Eure Aufmerksamkeit.

Richard Albrecht

LUG-Görlitz, LUG-Ottobrunn

Richard Albrecht

Linux in Görlitz