

# Lektion 08

```
defaultProps = {
  'default',
  deAvatar: false,
}

UserDetailsCardOnHover = showOnHover(UserDetailsCard);

UserLink = ({  
  ...  
  secondaryLink,  
  children,  
  deAvatar,  
  href,  
  ...  
} in className={styles.container}>  
  
  includeAvatar && (  
    <UserDetailsCardOnHover  
      user={user}  
      delay={CARD_HOVER_DELAY}  
      wrapperClassName={styles.avatarContainer}>  
      <Avatar user={user} />  
    </UserDetailsCardOnHover>  
  )  
  
  div  
    className={classNames(  
      styles.linkContainer,  
      inline && styles.inlineContainer  
    )}  
  
    <UserDetailsCardOnHover user={user} delay={CARD_HOVER_DELAY}>  
      <Link  
        to={({ pathname: buildUserUrl(user) })}  
        className={classNames(styles.name, {  
          [styles.alt]: type === 'alt',  
          [styles.centerName]: !secondaryLink,  
          [styles.inlineLink]: inline,  
        })}  
      >  
      {children || user.name}  
    </Link>  
  
    <secondaryLink  
      ? null  
      : <a  
        href={secondaryLink.href}  
        className={classNames(styles.name, {  
          [styles.alt]: type === 'alt',  
          [styles.secondaryLink]: secondaryLink,  
        })}  
      >  
      {secondaryLink.label}  
    </a>  
  </UserDetailsCardOnHover>  
  </div>  
  span>  
  
Link.propTypes = propTypes;  
Link.defaultProps = defaultProps;  
  
  default UserLink;
```

Linux

# Backups

- Ein sehr wichtigstes Sicherheitskonzept ist die Anfertigung von Backups. Das betrifft den PC zuhause genau wie den Server und die Clients im Unternehmen.
- Manche Daten sind sehr wertvoll, andere haben ideellen Wert
- Leider werden Backups trotzdem am meisten vernachlässigt
- Firmen sind schon wegen verloren gegangener Daten in den Ruin geschlittert
- Dabei ist das Anfertigen von Backups sehr einfach, vor allem, weil man alle Schritte automatisieren kann

# Nutzen von Backups

- **Wofür sind Backups nützlich (aus Sicht eines Unternehmens)?**
  - Durch defekte Datenträger sind wichtige Kundendaten und Firmendaten zerstört. Das ganze System kann ausfallen (wenn kein RAID-System eingesetzt wird, oder das RAID defekt ist)
  - Durch falsche Bedienung wurde ein Großteil der Kundendaten gelöscht (hier hilft kein RAID)
- **Wofür sind Backups nützlich (für private Anwender)?**
  - Auch auf den privaten Rechnern befinden sich wertvolle Daten. Das muss nicht in Geld-Wert sein. Oft sind es ideelle Wert (Fotos oder Filme der Kinder, usw.)
  - Backups sind auch hier wichtige Sicherungselemente
  - Raid-System kommen nur ganz selten zum Einsatz
- **Datenträger gehen irgendwann kaputt**

# **Reicht nicht ein RAID-System?**

- RAID-Systeme (Redundant Array of Independent Discs) sind heute sehr verbreitet, weil ein RAID mit relativ geringem finanziellem Aufwand realisierbar ist
- Die Redundanz gewährleistet gute Sicherheit beim Ausfall eines Datenträgers (außer RAID 0)
- Wenn mehrere Datenträger ausfallen, versagt ein RAID ebenfalls
- Datenverlust durch Fehlbedienung kann ein RAID-System auch nicht verhindern, weil die Redundanz den Datenverlust ebenfalls „spiegelt“
- Für diese Fälle helfen nur Backups

# Backup-Speicher 1

## ■ Externe Festplatten

- Wegen der niedrigen Preise wird diese Variante immer häufiger verwendet. Schnelle USB-Schnittstellen und hohe Speicherkapazitäten begünstigen dies ebenfalls
- Die Ausfallsicherheit ist aber vergleichsweise gering und viele Sicherungen auf einem Datenträger begünstigen den *Single-Point-of-Failure*

## ■ Beschreibbare DVD bzw. Blu-Ray

- Durch die vergleichsweise geringe Speicherkapazität ist ein häufiger Medienwechsel nötig.
- Für sehr große Datenmengen sind DVD und Blu-Ray weniger geeignet.
- Bei entsprechender Lagerung sind sie aber recht robust

# **Backup-Speicher 2**

## **■ Bandstreamer**

- Bandstreamer erfreuen sich ebenfalls wieder hoher Beliebtheit. Speicherkapazitäten bis in den TeraByte-Bereich begünstigen den Trend. Sie sind stabil, zuverlässig und gut lagerfähig. Band-Automaten sorgen für eine effiziente Lagerung und Bereitstellung der Bänder unter hermetischem Abschluss. Nachteil: Durch die serielle Speicherung benötigen sie relativ viel Zeit zum Speichern und Auslesen

## **■ NAS- und SAN-Systeme**

- In großen Netzwerken, aber auch im privaten Bereich findet man zunehmend NAS-Systeme für Backupaufgaben
- Im NAS kann ebenfalls ein RAID-System für Ausfallsicherheit sorgen. In großen Umgebungen benutzt man dafür SAN-Systeme (Storage Area Network)

## **■ Auch Kombinationen von Backup-Speichern sind möglich und üblich**

# Generationen-Prinzip

- Auch *Großvater - Vater - Sohn - Prinzip* genannt
  - Angenommen, Sie entdecken erst nach ein paar Tagen, dass Ihre Kundendaten gelöscht sind und Sie überschreiben jeden Tag die Sicherung vom Vortag. Dann hat sich das Backup nicht gelohnt
  - Um dies zu vermeiden werden Backups in verschiedenen zeitlichen Abstufungen erstellt. So erhält man die Datenzustände zu verschiedenen Zeitpunkten
  - Im klassischen GVS-Prinzip setzt man 21 Bandkassetten ein. 12 Bänder für die monatliche Sicherung (Großväter), 5 für die Wochensicherungen in den Monaten (Väter, werden z.B. freitags gesichert) und die letzten 4 Bänder (Söhne) werden täglich von Mo. bis Do. verwendet
  - Die 21 Kassetten sind ein Gedankenspiel, denn oft reicht ein Band für eine Sicherung nicht aus
  - Aus dieser Strategie heraus lassen sich viele Variationen bilden
- Ziel muss es sein, die Daten für einen bestimmten Zeitpunkt schnell wieder herstellen zu können

# Arten der Sicherung

- **Vollsicherung**
  - Hier werden sämtliche Daten gesichert – oft ein langer Prozess
- **Differentielle Sicherung**
  - Diese Sicherung enthält nur die Daten, die sich seit der letzten Vollsicherung geändert haben
- **Inkrementelle Sicherung**
  - Hierbei werden alle Daten gesichert, die sich seit der letzten Sicherung geändert haben, egal, ob es eine Vollsicherung, eine differentielle oder eine vorherige inkrementelle Sicherung war
- **Einige Strategien benutzen nur Voll- und inkrementelle Sicherungen oder nur Voll- und differentielle Sicherungen**
- **In Linux sind auch Mischformen denkbar. Archivbits werden nicht genutzt.**

# Faktoren einer Strategie

- Sicherungsstrategien sind sehr stark an das jeweilige Unternehmen gebunden.  
Die Strategie eines Unternehmens kann für ein anderes Unternehmen  
vollkommen unbrauchbar sein. Hier die Hauptfaktoren aller Strategien:
  - Zeitraum der Sicherungen
  - Größe des Datenbestands
  - Gewünschte Wiederherstellungszeit
  - Finanzielle Mittel

# Zu sichernde Daten in Linux

- Wichtig für Datensicherungen ist, dass man vor allen die veränderlichen Daten sichert. Die zugehörigen (unveränderlichen) Programme lassen sich schnell installieren, müssen also nicht unbedingt gesichert werden
- Die veränderlichen Daten befinden sich in den Verzeichnissen:
  - /etc
  - /home
  - /root
  - /var
  - und ggf. /opt

# Sicherungsprogramme

- In den verschiedenen Distributionen existiert eine Vielzahl von Sicherungssoftware. Man kann Sicherungsprogramme aus der grafischen Oberfläche (z.B. Deja Dub), wenn vorhanden, ausführen oder Sicherungswerkzeuge aus der Konsole (z.B. tar) nutzen, die für eine eigene Sicherungsstrategie besser geeignet sind
- tar ist die Abkürzung für tape archiver (also Band Archivierer) und fasst alle zu sichernden Dateien zu einer Archivdatei zusammen, die lt. Konvention den Suffix .tar erhält
  - Wie es der Name schon sagt, ist dieses Verfahren ursprünglich für die Bandspeicherung gedacht
  - Es lässt sich aber für jedes beliebige Speichermedium einsetzen

# Komprimierung

- Mittels gzip lassen sich tar-Sicherungen komprimieren, was erheblichen Speicherplatz sparen kann. Die Dateiendung lautet dann .tar.gz oder kurz .tgz
- gz steht für gzip. Auch andere Komprimierungsverfahren werden genutzt, z.B. bzip (bz)
- Ob man die Komprimierung einsetzen sollte, muss der Administrator für sich entscheiden, hängt aber auch von den Faktoren der Strategie ab
- Im Kommando tar unterscheidet man zwischen Aktionen und Optionen
  - Aktionen schieben dabei grundlegende Vorgänge an (z.B. sichern oder wiederherstellen)
  - Optionen begleiten die Aktionen (z.B. Anzeigen der Aktionen)

# Liste der Aktionen

Aktion	Beschreibung
-c	Erstellt ein neues Archiv (c – create)
-t	Zeigt das Inhaltsverzeichnis des Archivs an (t – table)
-r	Fügt eine Datei an das vorhandene Archiv an (r – recreate)
-d	Vergleicht Dateien, um Unterschiede aufzuzeigen (d – difference)
-x	Extrahiert die Dateien aus dem angegebenen Archiv in das aktuelle Verzeichnis oder bei der Erstellung mit der Option –P an den Originalspeicherplatz. Dabei werden die Dateien im Archiv nicht gelöscht

# Liste der Optionen

Option	Beschreibung
-f <Archivname>.tar	Legt den gewünschten Dateinamen des Archives fest, was natürlich immer benutzt wird (f – filename)
-v	Zeigt die Aktionen während des Archivierens an (v – verbose)
-z	Komprimiert das angegebene Archiv mit gzip
-C <Verzeichnis>	Mit –C geben Sie das Verzeichnis an, unter dem die Dateien wieder hergestellt werden
-P	Achiviert die Dateien mit absoluter Pfadangabe, so dass beim Wiederherstellen der ursprüngliche Standort der Dateien gewählt wird

# Archiv erstellen

- Sie wollen Ihr eigenes home-Verzeichnis in das vorher erstellte Verzeichnis /backup mittels tar sichern (Achtung root-Rechte sind nötig)
  - # tar -cvf <Archivname>.tar <verzeichnis/der zu archivierenden Dateien>
  - # tar -cvf /backup/helmut-home-sicherung.tar /home/helmut
    - Sie benutzen natürlich Ihren Namen
    - -c erstelle ein Archiv (create)
    - -v zeige die Aktionen an (verbose)
    - -f verwende den angegebenen Archivnamen (filename)
    - /backup - Verzeichnis in dem das Archiv *helmut-home-Sicherung.tar* gespeichert wird
- Die folgenden Übungen werden hier auf dem Server „deb-s1“ beschrieben, sind aber auch auf dem Client-Rechner „debian“ sinnvoll

# Beispiel 1: einfaches Archiv

```
root@deb-s1:~# tar -cvf /backup/helmut-home-sicherung.tar /home/helmut
tar: Entferne führende „/“ von Elementnamen
/home/helmut/
/home/helmut/.gnupg/
/home/helmut/.gnupg/private-keys-v1.d/
/home/helmut/.local/
/home/helmut/.local/share/
/home/helmut/.local/share/nano/
/home/helmut/.bash_history
/home/helmut/.bashrc
/home/helmut/.ssh/
/home/helmut/.ssh/authorized_keys
/home/helmut/.bash_logout
/home/helmut/.profile
```

```
root@deb-s1:~# ls -l /backup
insgesamt 20
-rw-r--r-- 1 root root 20480 Mai  9 18:48 helmut-home-sicherung.tar
root@deb-s1:~#
```

# Komprimiertes Archiv

- **Wir sichern wieder unsere eigenes home-Verzeichnis in das Verzeichnis /backup**
- `# tar -cvzf <Archivname>.tar.gz <verzeichnis/der zu archivierenden Dateien>`
- `# tar -cvzf /backup/helmut-home-sicherung.tar.gz /home/helmut`
  - Sie benutzen natürlich Ihren Namen
  - -c erstelle ein Archiv (create)
  - -v zeige die Aktionen an (verbose)
  - -z benutze gzip zum komprimieren des Archivs (Komprimierung)
  - -f verwende den angegebenen Archivnamen (filename)
  - /backup – Verzeichnis in dem das Archiv *helmut-home-Sicherung.tar.gz* gespeichert wird

## Beispiel 2: Archiv komprimiert

```
root@deb-s1:~# tar -cvzf /backup/helmut-home-sicherung.tar.gz /home/helmut
tar: Entferne führende "/" von Elementnamen
/home/helmut/
/home/helmut/.gnupg/ Komprimierung mit gzip
/home/helmut/.gnupg/private-keys-v1.d/
/home/helmut/.local/
/home/helmut/.local/share/
/home/helmut/.local/share/nano/
/home/helmut/.bash_history
/home/helmut/.bashrc
/home/helmut/.ssh/
/home/helmut/.ssh/authorized_keys
/home/helmut/.bash_logout
/home/helmut/.profile

-rw-r--r-- 1 root root 3103 Mai  9 18:57 helmut-home-sicherung.tar.gz
-rw-r--r-- 1 root root 20480 Mai  9 18:48 helmut-home-sicherung.tar
root@deb-s1:~#
```

# Fazit

- Die Kompression mit gzip verkleinert die Archivdatei um einige Faktoren.
- Allerdings hängt die Kompression stark von den Inhalten ab.
- Bereits komprimierte Daten von Bildern, Videos und anderen Inhalten, lassen sich natürlich nicht noch weiter komprimieren oder nur in sehr geringem Umfang
- Mit tar kann oder sollte man Archivdateien erstellen, die sich an die Pfadstruktur halten
  - separates Archiv von /etc
  - separates Archiv von /home
  - usw.

# Archivinhalt ansehen

- Nachdem ein Archiv erstellt ist möchte man nicht selten sehen, welche Dateien sich im Archiv befinden
  - Für nicht komprimierte Archive nutzt man dazu die Option -tf
  - # tar -tf </Pfad/zur/Archivdatei>.tar
  - # tar -tf /backup/helmut-home-sicherung.tar
- Der Inhalt lässt sich auch aus komprimierten Dateien auslesen
  - Dafür nutzt man die Option -tzf
  - # tar -tzf </Pfad/zur/Archivdatei>.tar.gz
  - # tar -tzf /backup/helmut-home-sicherung1.tar.gz

## Beispiel 3: Archivinhalte

```
root@deb-s1:~# tar -tf /backup/helmut-home-sicherung.tar
home/helmut/
home/helmut/.gnupg/
home/helmut/.gnupg/private-keys-v1.d/
home/helmut/.local/
home/helmut/.local/share/
home/helmut/.local/share/nano/
home/helmut/.bash_history
home/helmut/.bashrc
home/helmut/.ssh/
home/helmut/.ssh/authorized_keys
home/helmut/.bash_logout
home/helmut/.profile
root@deb-s1:~# ■
```

```
root@deb-s1:~# tar tfz /backup/helmut-home-sicherung.tar.gz
home/helmut/
home/helmut/.gnupg/
home/helmut/.gnupg/private-keys-v1.d/
home/helmut/.local/
home/helmut/.local/share/
home/helmut/.local/share/nano/
home/helmut/.bash_history
home/helmut/.bashrc
home/helmut/.ssh/
home/helmut/.ssh/authorized_keys
home/helmut/.bash_logout
home/helmut/.profile
root@deb-s1:~# ■
```

# Archiv zurückspielen

- Archive wiederherstellen geschieht mit der Option x
- Das Wiederherstellen geschieht hier im Ordner, in dem man sich befindet!
  - # tar -xf <Pfad/zur/Archivdatei>.tar
  - # tar -xf /backup/helmut-home-sicherung.tar
- Wurde beim Archivieren die Option -P gesetzt wird das Archiv in die Originalordner zurückgespielt.
- Gehen Sie vorsichtig mit dieser Option um, da neuere Dateien durch ältere Archivdateien überschrieben werden

## Beispiel 4: Archiv zurückspielen

```
root@deb-s1:~# mkdir /replace
root@deb-s1:~# cd /replace
root@deb-s1:/replace# tar -xf /backup/helmut-home-sicherung.tar
root@deb-s1:/replace# ls
home
root@deb-s1:/replace# █
```

## Beispiel 4: Ordner bestimmen

```
root@deb-s1:/# mkdir /zurueck
root@deb-s1:/# cd
root@deb-s1:~/# tar -xf /backup/helmut-home-sicherung.tar -C /zurueck/
root@deb-s1:~/# cd /zurueck/
root@deb-s1:/zurueck# ls
home
root@deb-s1:/zurueck# █
```

```
root@deb-s1:/# mkdir /zurueck1
root@deb-s1:/# cd
root@deb-s1:~/# tar -xzf /backup/helmut-home-sicherung1.tar.gz -C /zurueck1
root@deb-s1:~/# cd /zurueck1
root@deb-s1:/zurueck1# ls
home
root@deb-s1:/zurueck1# █
```

# Zusammenfassung und Ausblick

- Mit all den bisher gezeigten Methoden von tar ist es Ihnen nun möglich eine Vollsicherung zu erstellen und diese an einen gewünschten Ort zurückzuspielen.
- Allerdings lassen sich damit keine inkrementellen oder differentiellen Backups erstellen.
- Anders als in Windows werden keine Archivbits gesetzt, die anzeigen, welche Dateien und Verzeichnisse sich seit der letzten Vollsicherung geändert haben.
- Dafür bietet Linux einen flexibleren Umgang mit neuen und geänderten Dateien bzw. Verzeichnissen an

# Das Kommando `find`

- Mit dem Kommando `find` lassen sich beliebige Dateien und Verzeichnisse in Ihrem Dateisystem suchen und finden
- Durch unterschiedliche Suchoptionen, die durchaus schwierig erscheinen können, lassen sich präzise Muster erstellen, die gefilterte Ergebnisse liefern
- Für die Nutzung von `find` sind root-Rechte notwendig
- Die allgemeine Syntax des `find`-Kommandos lautet:
  - `# find <Startverzeichnis der Suche> <Suchoption(en)> -Option`
- Die nachfolgende Liste gibt einen Überblick über die vielfältigen Möglichkeiten
- Am Ende werden wir `find` in `tar` einbauen (Kommandosubstitution)

# Liste der Suchoptionen

Suchoption	Wirkung
-atime [+/-]<Tage>	Letzter Zugriff (access) in Tagen der älter (+) oder jünger (-) als die angegebenen Tage ist
-mtime [+/-]<Tage>	Letzte Änderung (modification) in Tagen die älter (+) oder jünger (-) als die angegebenen Tage ist
-user <Name>	Ergebnisse gehören zum Benutzer
-group <Name>	Ergebnisse gehören zur Gruppe
-type <Dateityp>	f = Datei (file) d = Verzeichnis (directory) l = Verweis (link)
-name <Datei oder Verzeichnis>	Suche nach dem angegebenen Datei- oder Verzeichnisnamen
-size [+/-]<Größe>k	Dateigröße größer (+) oder kleiner (-) als die angegebene Dateigröße in Kilobyte
-maxdepth <Level>	Suchtiefe in der Verzeichnisstruktur gerechnet ab dem Startverzeichnis, wobei Level eine Ziffer ist

# Einige Beispiele mit find (1)

- Sie suchen beginnend vom / (Wurzel-Verzeichnis) den Datei- oder Verzeichnisname korn auf dem Server deb-s1 (Verbindung über ssh hergestellt)

```
root@deb-s1:~# find / -name korn  
/home/korn  
root@deb-s1:~#
```

- Das Durchsuchen aller Verzeichnisse benötigt einige Sekunden

## Einige Beispiele mit find (2)

- Sie suchen beginnend vom /home-Verzeichnis alle Verzeichnisse, die zur Gruppe „einkauf“ gehören

```
root@deb-s1:~# find /home -group einkauf -type d
/home/verteiler/einkauf
/home/meier
/home/korn
root@deb-s1:~# █
```

- Sie suchen beginnend vom /home-Verzeichnis nach allen Dateien des Benutzers helmut, die sich in den letzten 3 Tagen verändert haben

```
root@deb-s1:~# find /home -user helmut -mtime -3
/home/helmut
/home/helmut/.local
/home/helmut/.local/share
/home/helmut/.local/share/nano
/home/helmut/.bash_history
/home/helmut/.bashrc
root@deb-s1:~# █
```

# Einige Beispiele mit find (3)

- Sie suchen beginnend vom /home-Verzeichnis alle Dateien auf die der Benutzer helmut in den letzten 3 Tagen Zugriff hatte (Vertrauensstellung des Administrators)

```
root@deb-s1:~# find /home -user helmut -atime -3
/home/helmut
/home/helmut/.gnupg
/home/helmut/.gnupg/private-keys-v1.d
/home/helmut/.local
/home/helmut/.local/share
/home/helmut/.local/share/nano
/home/helmut/.bash_history
/home/helmut/.bashrc
/home/helmut/.ssh
/home/helmut/.ssh/authorized_keys
/home/helmut/.bash_logout
/home/helmut/.profile
root@deb-s1:~# ■
```

## Einige Beispiele mit find (4)

- Sie suchen beginnend von /home nach allen Dateien des Benutzers meier die größer als 5000 Kilobyte sind, um „Speicherfresser“ zu finden

```
root@deb-s1:~# find /home -user meier -size +5000k  
root@deb-s1:~# ■
```

- Es wurden keine Dateien gefunden, die in das Suchmuster passen

# Differenzielles Konzept

- **Die Sicherungen beziehen sich im Beispiel immer auf die Vollsicherung**
  - # tar -cvzf /backup/home-voll-fr.tar.gz /home
- **Einbinden von find in die tar-Funktion zu Erstellung eines Backups vom /home-Verzeichnis, nur mit den veränderten Daten nach der Vollsicherung am Freitag**
  - # tar -czf /backup/home-diff-mo.tar.gz \$(find /home -mtime -3)
  - # tar -czf /backup/home-diff-di.tar.gz \$(find /home -mtime -4)
  - # tar -czf /backup/home-diff-mi.tar.gz \$(find /home -mtime -5)
  - # tar -czf /backup/home-diff-do.tar.gz \$(find /home -mtime -6)
- **So kann man eine Wochenstrategie erstellen, die in vielen Unternehmen in gleicher oder ähnlicher Form praktiziert wird**

# **Der Schadensfall**

- Werden am Dienstag alle Kundendaten gelöscht, kann man im schlimmsten Fall die Vollsicherung vom letzten Freitag und die Teilsicherung vom Montag zurückspielen
- Geschäftsvorfälle vom Dienstag müssen allerdings händisch recherchiert werden
- Der Aufwand und der Schaden sind aber viel geringer, als keine Kundendaten mehr zu haben

# Zeitliche Steuerungen

- Sicherungen werden normalerweise erstellt, wenn geringer oder am besten kein Datenverkehr im Unternehmen stattfindet – also meistens nachts
- Nun wäre es ziemlich mühselig, wenn der Administrator jede Nacht ins Unternehmen fahren müsste, um eine Datensicherung zu erstellen
- Alle Linux-Distributionen haben dafür eine Lösung – cron
- Bei cron handelt es sich um einen Dienst, der zeitlich gesteuert bestimmte Aufgaben übernehmen kann
- Im Hintergrund läuft wie üblich ein cron-Dämon, in Debian in /usr/sbin

# Der Dämon cron

- Was macht dieser Dämon?
- Normal schläft er die ganze Zeit – aber der Schlaf ist kurz.
- Jede Minute wacht er auf, schaut in seinen Terminkalender, crontab genannt, und wenn nichts eingetragen ist, dreht er sich wieder um und schläft wieder eine Minute
- Die Datei crontab befindet sich im Verzeichnis /etc
- Schauen wie einmal hinein: # nano /etc/crontab

# Die crontab

GNU nano 3.2

/etc/crontab

```
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab'
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# Example of job definition:
# .----- minute (0 - 59)
# | .----- hour (0 - 23)
# | | .----- day of month (1 - 31)
# | | | .---- month (1 - 12) OR jan,feb,mar,apr ...
# | | | | .--- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
# | | | |
# * * * * * user-name command to be executed
17 * * * * root    cd / && run-parts --report /etc/cron.hourly
25 6 * * * root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )
47 6 * * 7 root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
52 6 1 * * root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly )
```

# Die Funktion der crontab

- Interessant ist vor allem der untere Bereich in der crontab
- Dort ist ein Kalender hinterlegt, der keine Wünsche offen lässt
  - Die vorderen 5 Spalten dienen der Zeiteinstellung nach folgendem Schema:  
(ist ein Stern eingetragen, ist jeder Zeitpunkt der jeweiligen Spalte aktiv)

*	*	*	*	*
Minuten	Stunden	Tag	Monat	Wochentag
0 - 59	0 - 23	1 – 31	1 - 12	0 – 7 (0; 7 = So)

- Dahinter folgt der Benutzer mit dessen Rechten der Task (Aufgabe) aufgerufen wird
- Als letztes folgt dann der Befehl oder das auszuführende Skript für den vorn genannten Zeitpunkt

# Die erste Zeitzeile der crontab

```
17 * * * * root cd / && run-parts --report /etc/cron.hourly
```

- Sie sagt aus, dass zur 17. Minute, jeder Stunde (\*), an jedem Tag (\*), in jedem Monat (\*) und an jedem Wochentag (\*)
- mit root-Rechten
- nach einem erfolgreichen Wechsel ins Wurzelverzeichnis, ein Programm mit dem Namen run-parts mit der Option --report und dem Parameter /etc/cron.hourly, ausgeführt wird

# Zeitübungen

- 25 6 \* \* \* - jeden Tag um 6 Uhr 25
- 45 5 \* \* 5 - jeden Freitag um 5 Uhr 45
- 30 9 1 \* \* - an jedem 1. eines Monats um 9 Uhr 30
- 00 12 21 6 \* - jedes Jahr am 21.06. um 12 Uhr
- 00 13,17 \* \* \* - jeden Tag um 13 Uhr und um 17 Uhr (ein Komma erlaubt 2 Zeitangaben)
- 00 8-16 \* \* \* - jeden Tag, jeweils zur vollen Stunde zwischen 8 Uhr und 16 Uhr
- \*/30 9-12 \* \* \* - alle halbe Stunde von 9 Uhr bis 12 Uhr
- **Dieses sehr flexible Zeitsystem benötigt ein wenig Eingewöhnung, ist aber für fast alle Eventualitäten nutzbar**
- **Achtung: Der nächste gleiche Aufruf in der crontab überschreibt das vorherige Ergebnis**

# anacron

- In den Zeilen darunter finden wir den Eintrag `/usr/bin/anacron`
- Dies ist ein Kommando, der für Rechner genutzt wird, die nicht rund um die Uhr genutzt werden
- Normalerweise wird ein Cronjob nicht ausgeführt, wenn der Rechner zu diesem Zeitpunkt ausgeschaltet ist
- anachron sorgt angeblich dafür, dass das Ereignis nachgeholt wird, sobald der Rechner wieder in Betrieb ist
- **Setzen Sie anacron nicht auf Ihren Servern ein. Es kann zu schweren Fehlern in Ihren Backups führen (Überschneidungen in der Ausführung)**

# Benutzer-cronjobs

- Jeder Benutzer kann eigene cronjobs erstellen
- Man muss allerdings beachten, dass er nur Aufgaben ausführen kann, die seine Benutzerrechte nicht überschreiten
- Eine Aufgabe wäre z.B. die Sicherung seiner Musiksammlung auf seinem Client-Rechner

```
helmut@debian:~$ mkdir musiksicherung
helmut@debian:~$ ls
Bilder Dokumente Musik          Öffentlich Vorlagen
Desktop Downloads musiksicherung Videos
helmut@debian:~$ crontab -e
```

# Wahl des Editors und cronjob

- Beim ersten Aufruf von crontab -e erhält man eine Auswahl von möglichen Editoren angeboten. Ich habe mich für nano entschieden.
- Scrollen Sie in der kommentierten crontab ganz nach unten und geben Sie zur gewünschten Zeit Ihren Sicherungsauftrag ein z.B.

```
#  
# m h  dom mon dow   command  
20 23  * * * tar -zcf /home/helmut/musiksicherung/musik.tar.gz /home/helmut/Musik/
```

```
helmut@debian:~/musiksicherung$ ls  
musik.tar.gz  
helmut@debian:~/musiksicherung$
```

# Welche cronjobs existieren

- Jeder Benutzer kann sich seine cronjobs anzeigen lassen
- Dazu gibt er das Kommando `$ crontab -l` (`l` - list) ein. Der Inhalt seiner cron-Tabelle wird angezeigt, ganz unten seine eingerichteten cronjobs
- Dies kann nach längerer Zeit durchaus nützlich sein

```
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow   command
20 23 * * * tar -zcf /home/helmut/musiksicherung/musik.tar.gz /home/helmut/Musik/
```

# Root sucht cronjobs

- Auch mit root-rechten versehen, kann man die cronjobs der anderen Benutzer kontrollieren
- Das ist z.B. dann sinnvoll, wenn der Benutzer mit seinem zugewiesenen Speicherplatz nicht auskommt

```
helmut@debian:~$ su -  
Passwort:  
root@debian:~# crontab -u helmut -l
```

- Nach der Option -u gibt man den Benutzernamen ein und mit der Option -l lässt man sich die Auflistung anzeigen

# Sicherungen mit root-Rechten

- Die Sicherung im eigenen home-Bereich zu speichern ist natürlich nicht sehr geschickt, aber man kann die entstandene komprimierte Datei auf einen anderen Datenträger kopieren oder verschieben
- Mit root-Rechten hat man natürlich weitaus mehr Möglichkeiten, da man sich im gesamten Datensystem bewegen und eine Sicherungsfestplatte fest einbinden kann
- Gleichzeitige Komplett-Sicherungen der home-Verzeichnisse aller Benutzer sind so genauso möglich, wie die Sicherung aller anderen relevanten Verzeichnisse, auf die der einzelne Benutzer keine Rechte hat

# Systemweite cronjobs

- Wenn man root-Rechte hat, lassen sich systemweite cronjobs ausführen
  - Diese trägt root direkt in die Datei /etc/crontab mit seinem Editor (nano) ein
  - Hier die umgesetzte wöchentliche Sicherungsstrategie:

```
# Wochenstrategie für das komplette home-Verzeichnis
```

```
15 20 * * 5    root    tar -czf /backup/home-voll-fr.tar.gz /home
15 20 * * 1    root    tar -czf /backup/home-mo.tar.gz $(find /home -mtime -3)
15 20 * * 2    root    tar -czf /backup/home-di.tar.gz $(find /home -mtime -4)
15 20 * * 3    root    tar -czf /backup/home-mi.tar.gz $(find /home -mtime -5)
15 20 * * 4    root    tar -czf /backup/home-do.tar.gz $(find /home -mtime -6)
```

- Die Sicherung erfolgt jeweils 20:15 Uhr, die Vollsicherung freitags (5), die erste Teilsicherung montags (1) und die weiteren Teilsicherungen dienstags (2) bis zum Donnerstag (4)
- Der Rechner muss zum Sicherungszeitpunkt laufen, was im Serverbetrieb normal ist!

# Nachteil der Strategie

- Die Wochenstrategie lässt sich genau für diesen Zweck sehr gut einsetzen
- In der nächsten Woche werden die Sicherungen allerdings komplett überschrieben
- Will man Sicherungen über einen langen Zeitraum behalten, so kann man die Sicherungen mit einem eingefügten Datum im Dateinamen ergänzen
- Diese Methode ist aber nur über die Skripterstellung möglich
- Die Skripte werden dann in die /etc/crontab eingetragen

# Eine weitere Festplatte

Vermeidung des Single Point of Failure

Linux

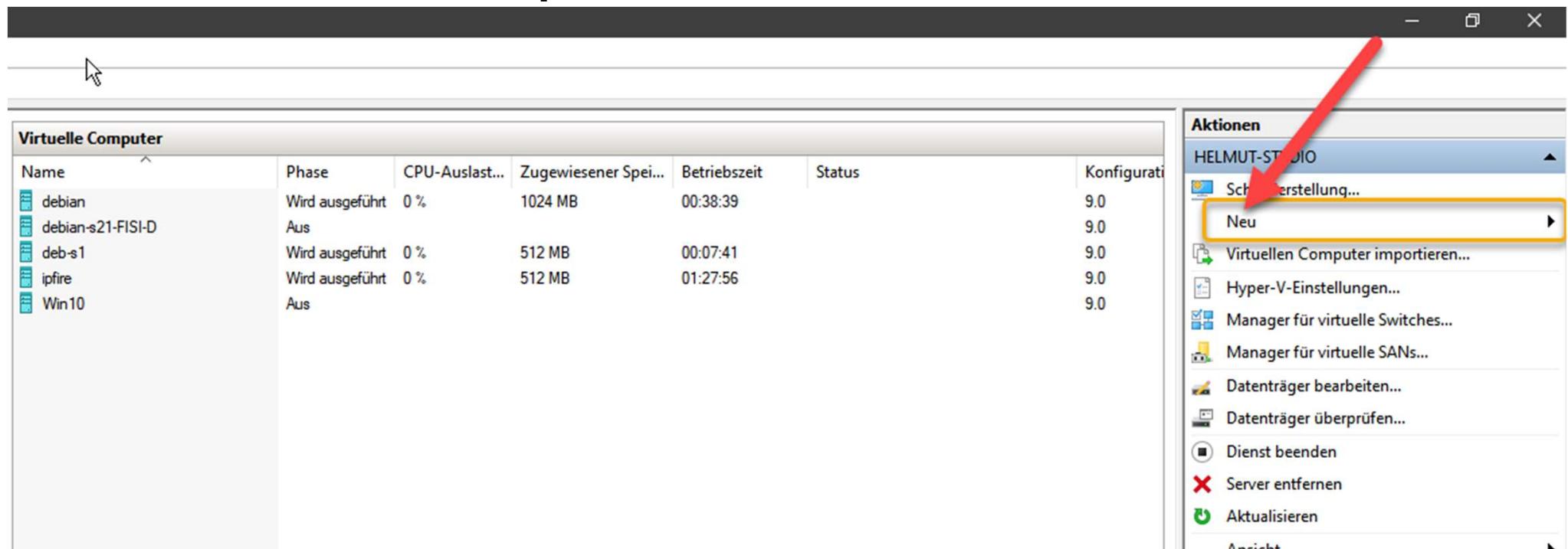


# Zusätzliche Datenträger

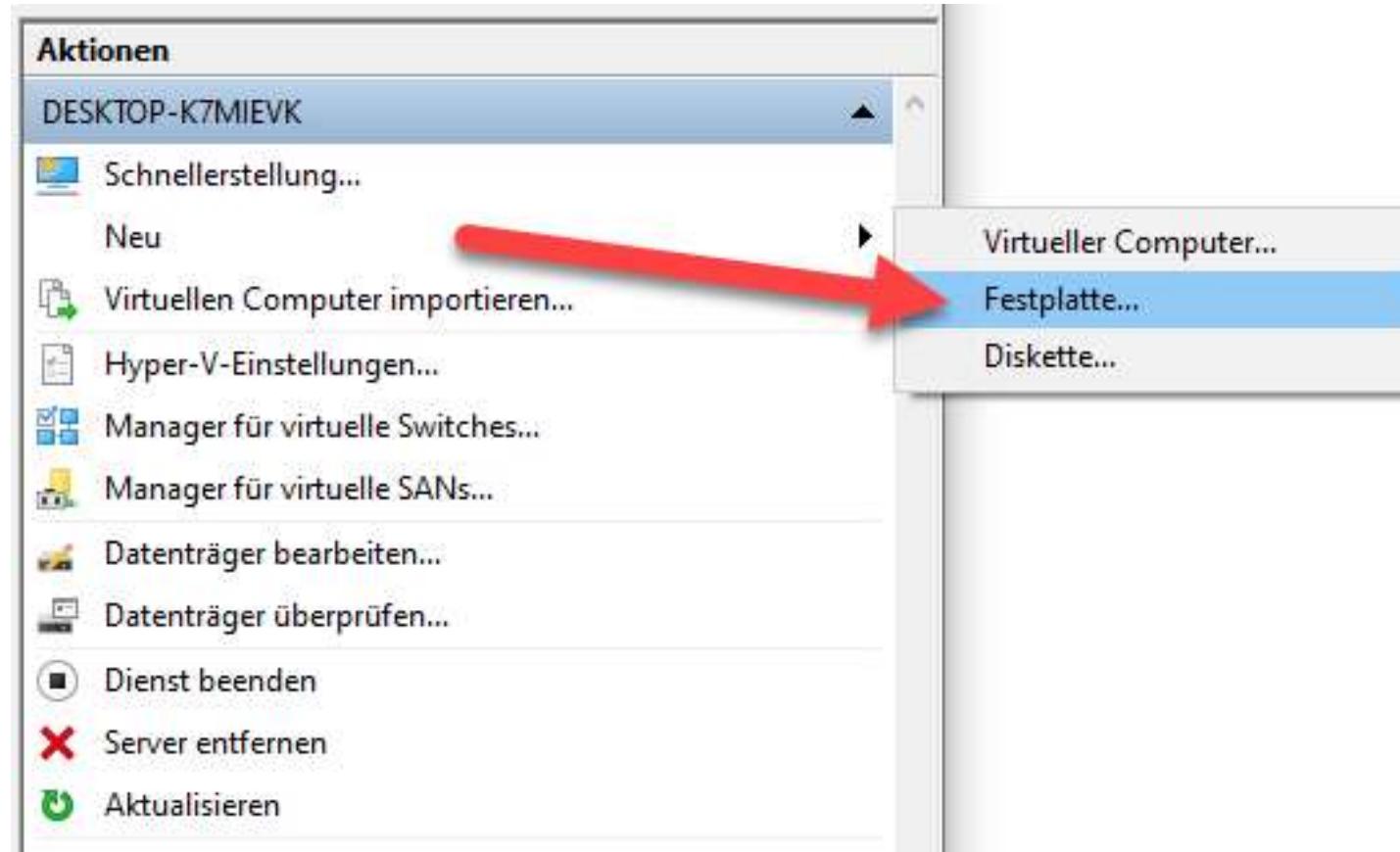
- Zusätzliche Datenträger (z.B. Festplatten) können an beliebiger, aber geeigneter Stelle ins Dateisystem eingefügt (gemountet) werden.
- Wir erstellen eine zusätzliche Festplatte in Hyper-V mit einer Größe von 10 GByte und mounten diese auf den Ordner /backup auf dem Server deb-s1, wodurch die Sicherungen auf die Festplatte geschrieben werden.
- Diese Festplatte werden wir bei Bedarf mounten, also so, als verbinden wir die Festplatte per USB mit dem PC
- Was hier mit einer kleinen Partition ausgeführt wird, kann auch mit einer richtigen großen Festplattenpartition eingerichtet werden.

# Neue Festplatte erstellen

- Klicken Sie im Bereich „Aktionen“ (rechts, oben in der Hyper-V) auf den Eintrag „Neu“ und wählen Sie „Festplatte“ aus (nächste Folie)

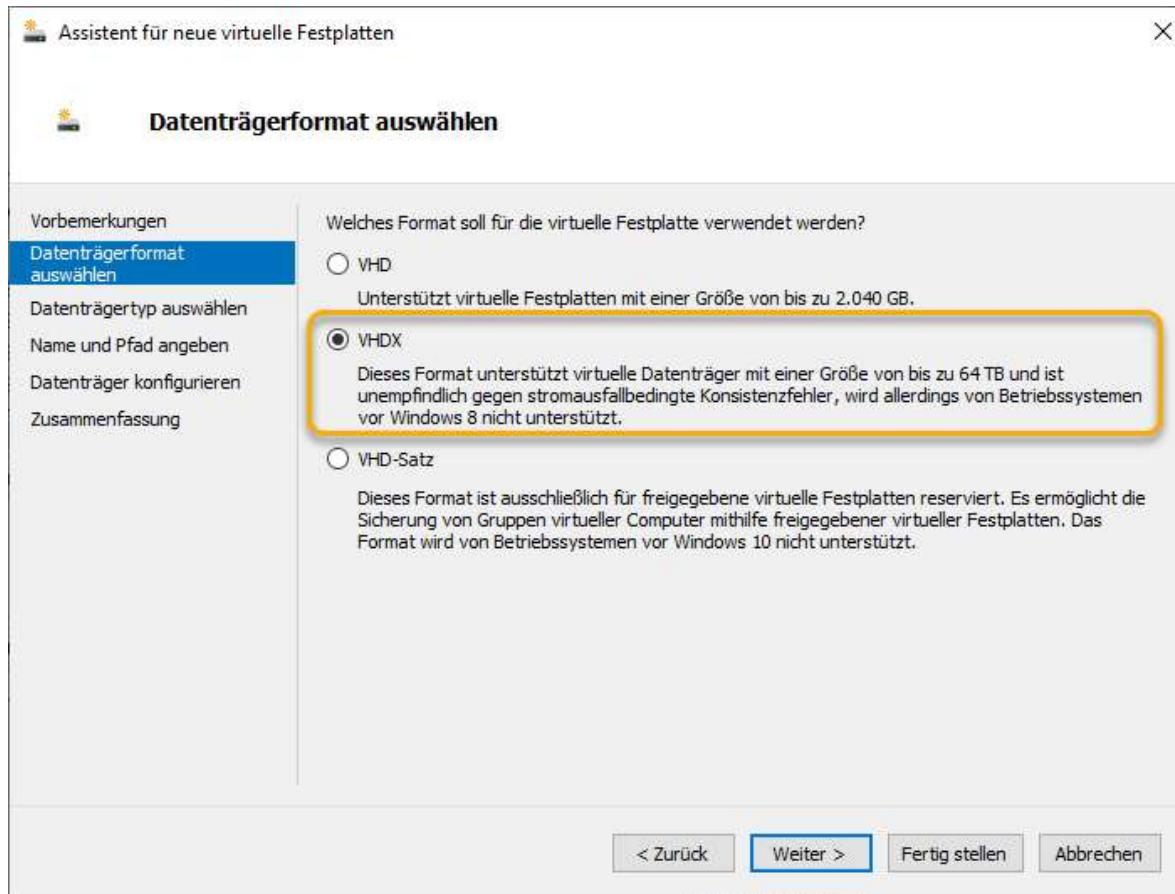


# Neue Festplatte erstellen



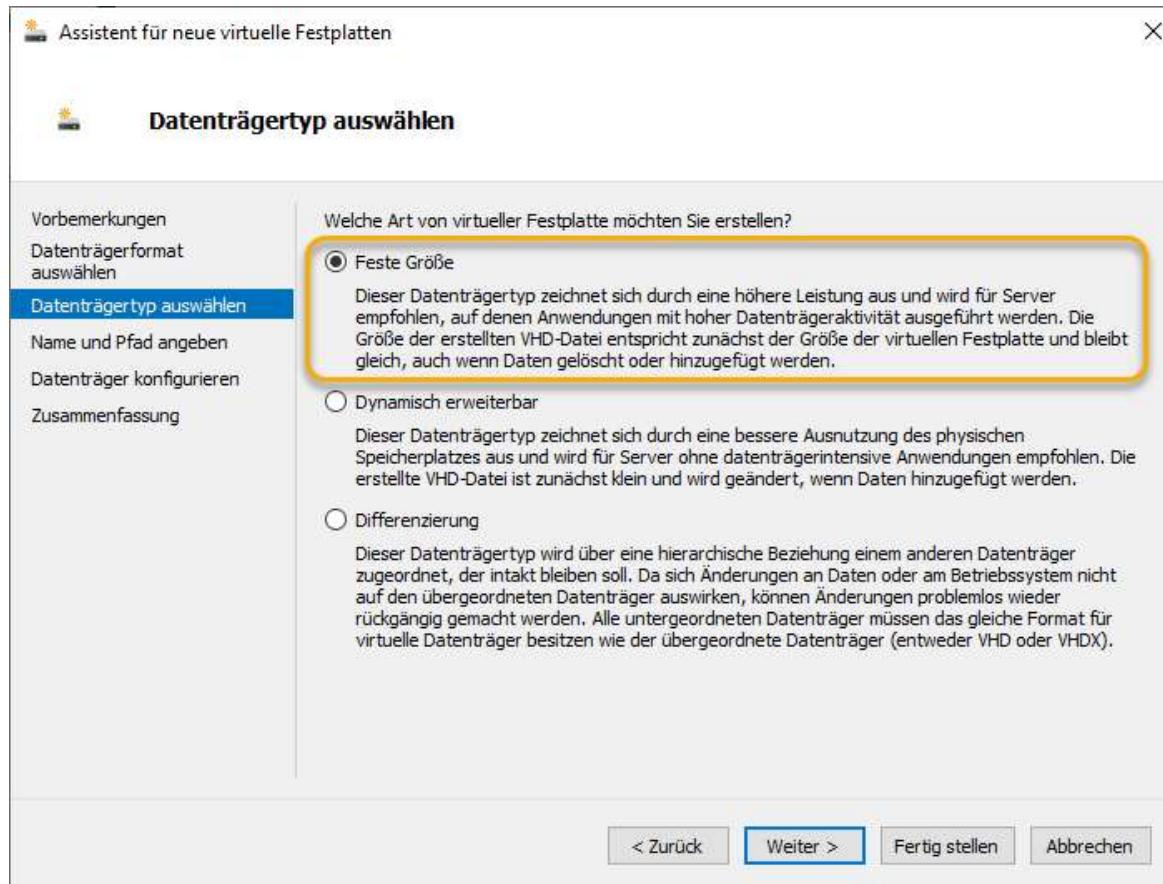
Linux

# Datenträgerformat



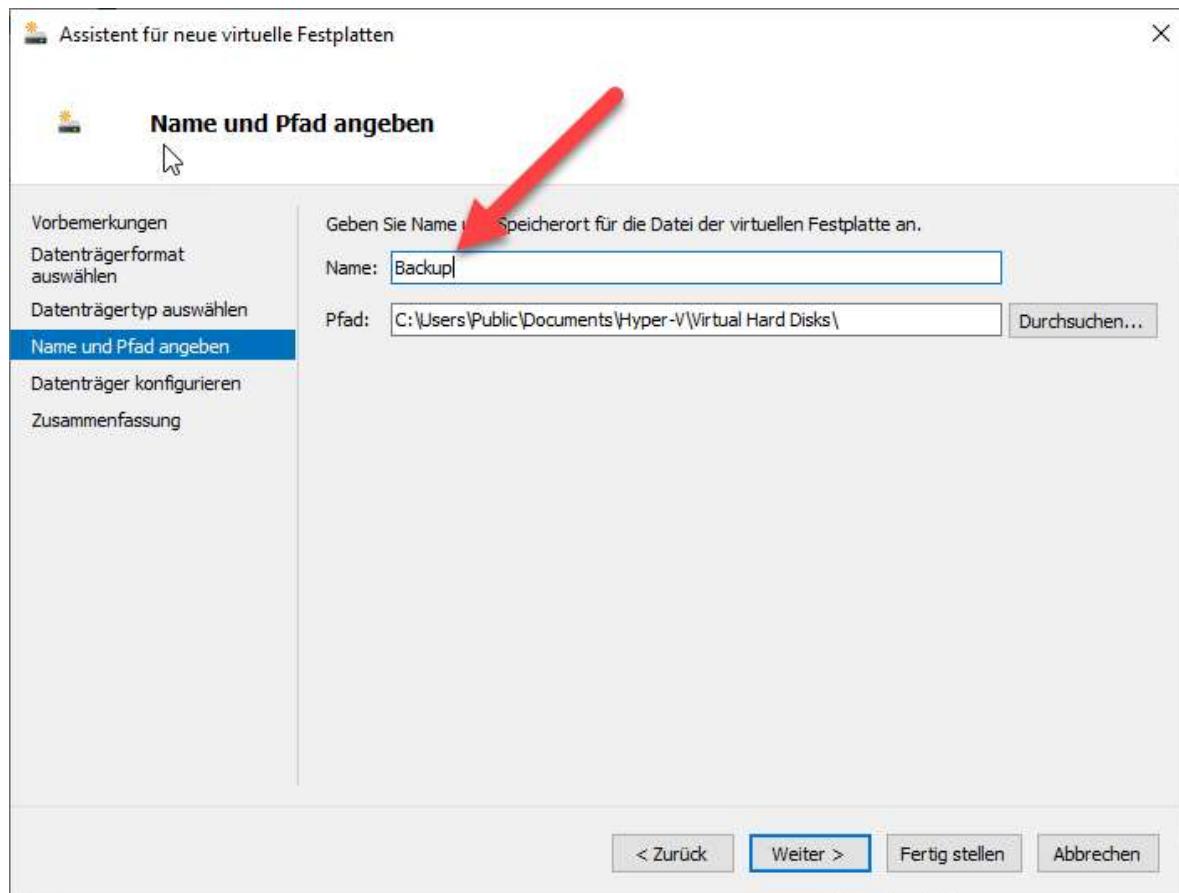
Linux

# Feste Größe



Linux

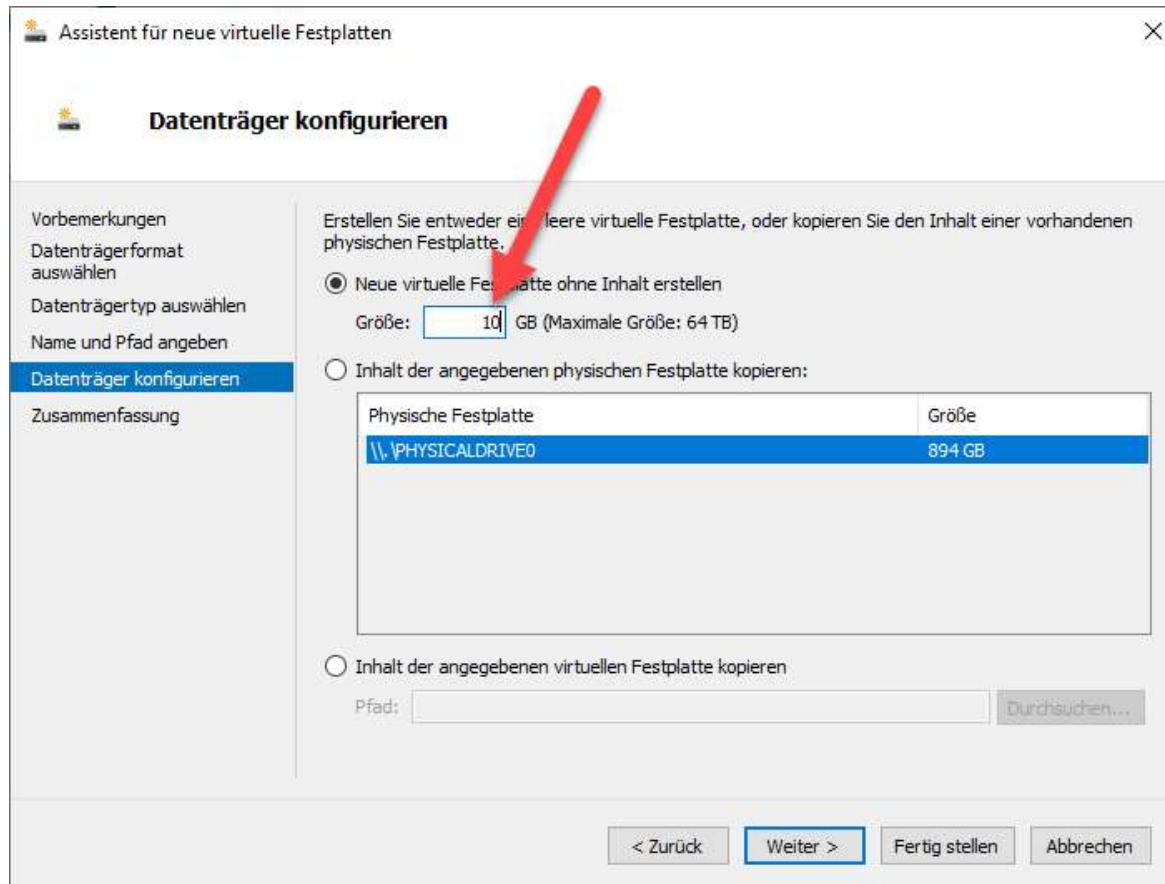
# Festplattenname



- Geben Sie der neuen Festplatte den Namen „Backup“
- Die Pfadangaben lassen Sie unberührt und klicken auf die Schaltfläche <Weiter>

Linux

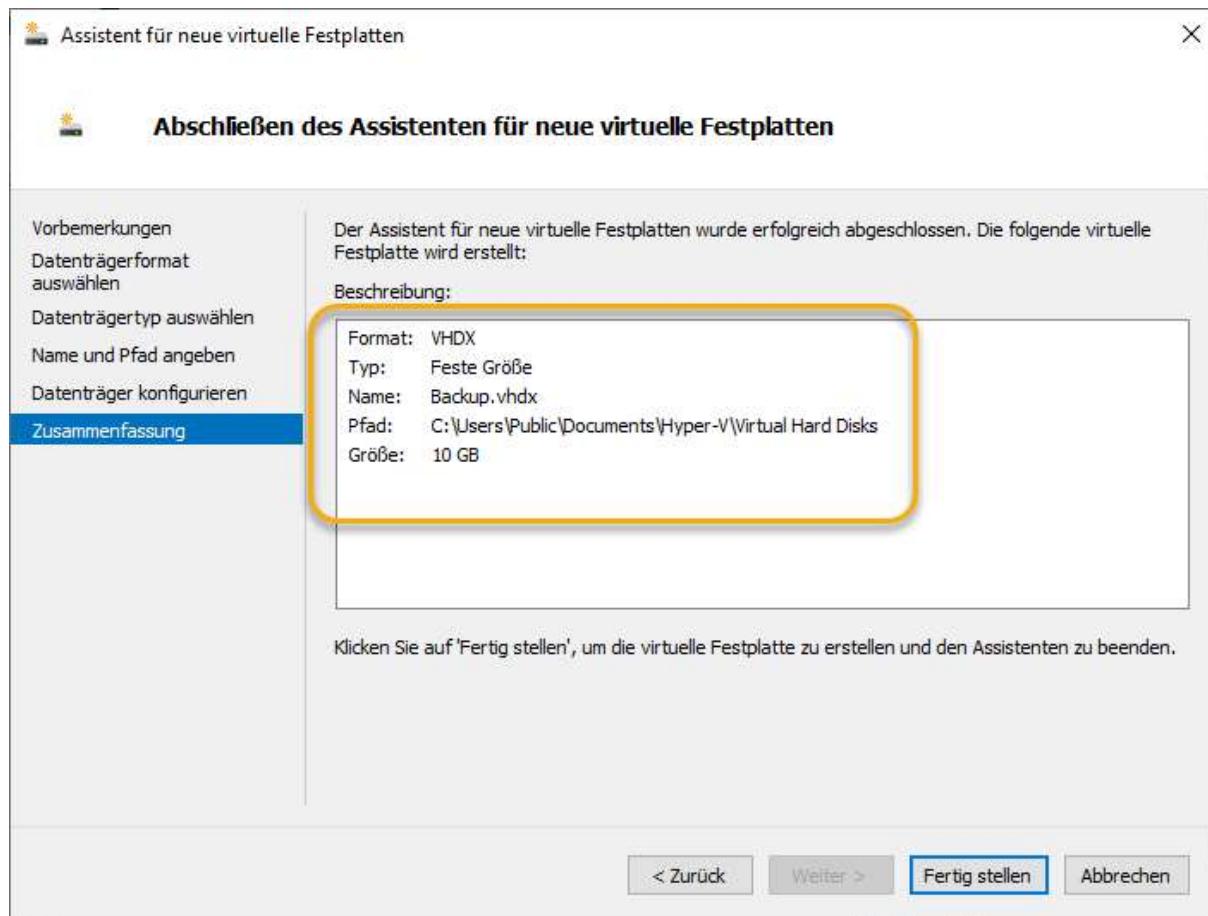
# Festplattengröße



- Sie übernehmen die Voreinstellung „**Neue virtuelle Festplatte ohne Inhalt erstellen**“
- Die Festplatte erhält eine Größe von **10 GB**

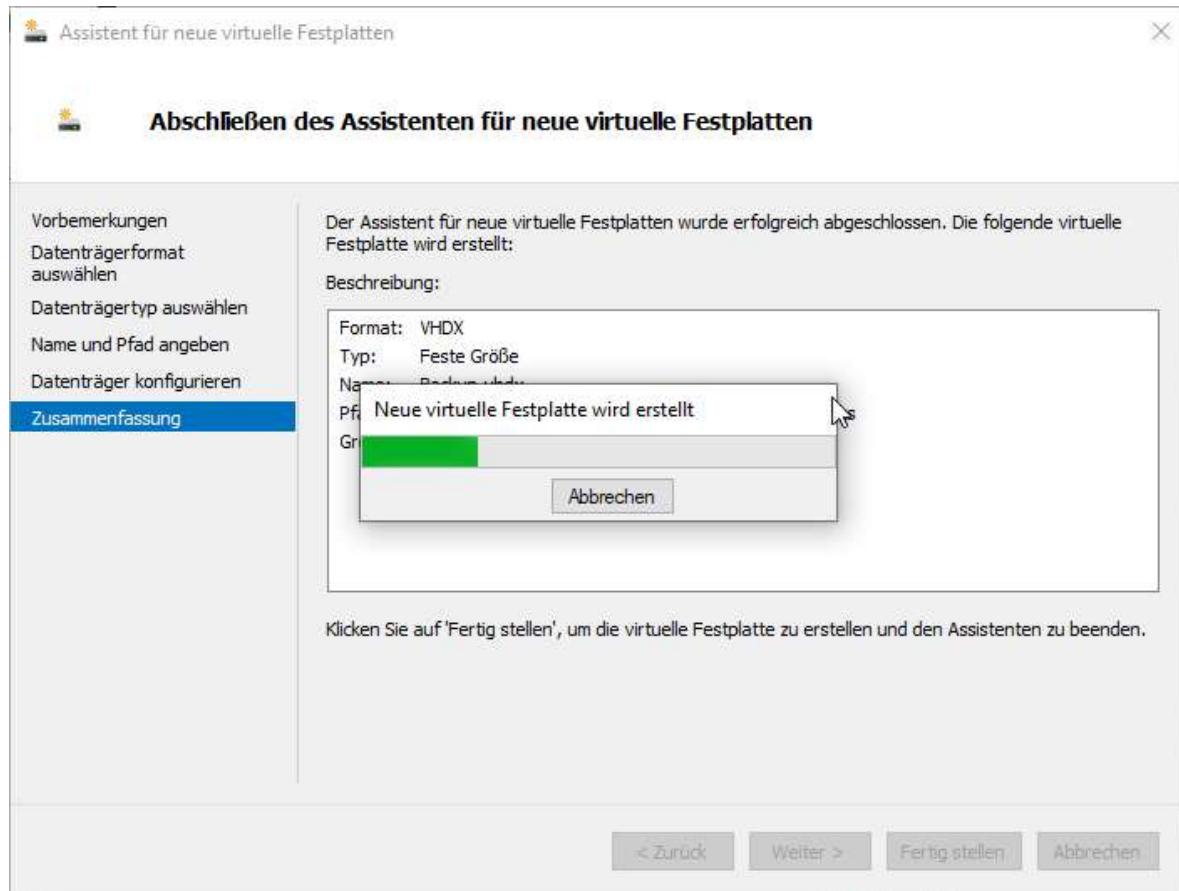
Linux

# Zusammenfassung



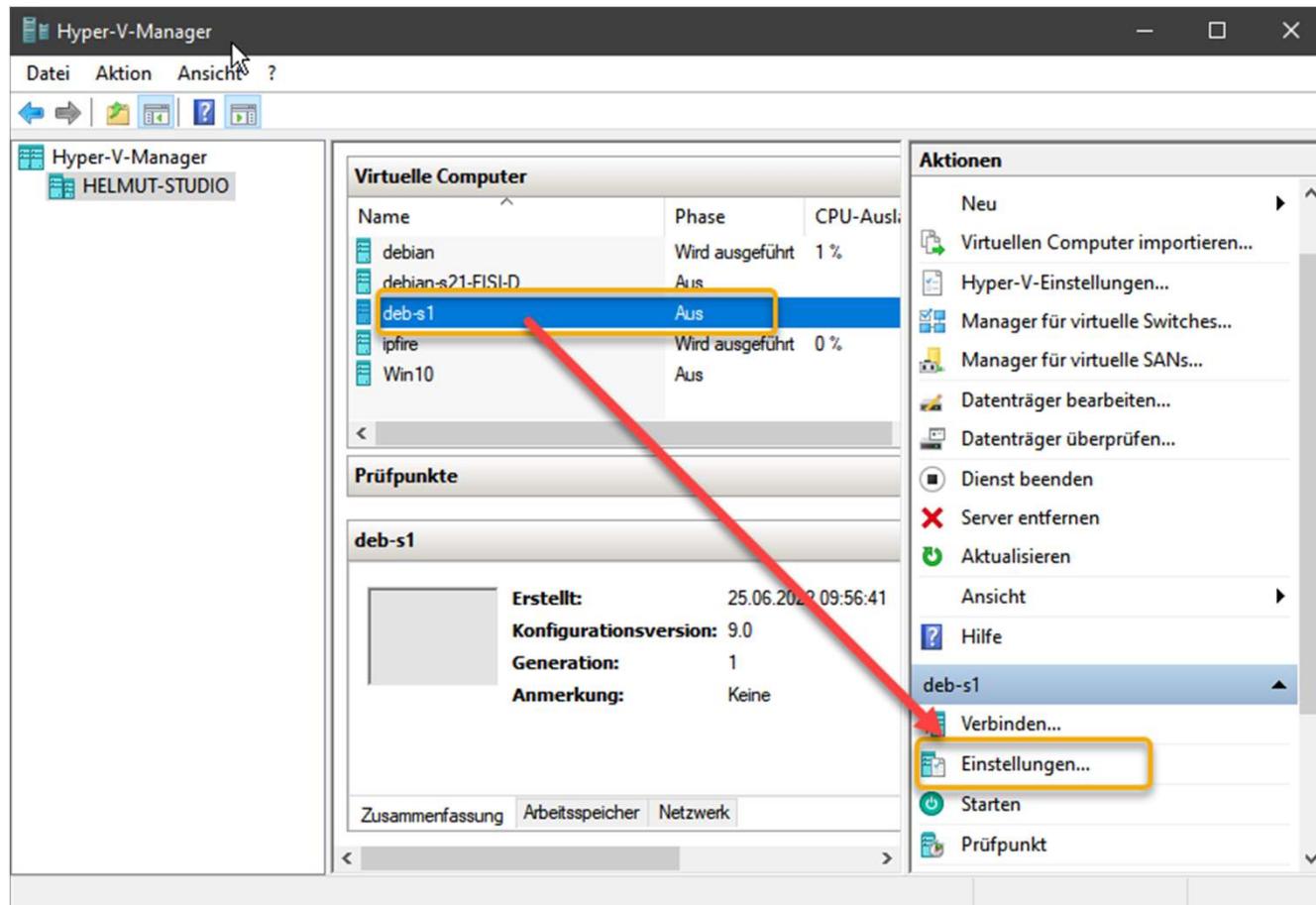
Linux

# Erstellungsvorgang



- **Die Erstellung benötigt ein wenig Zeit**
- **Wenn die Festplatte fertig erstellt ist, fahren Sie „deb-s1“ herunter, falls die VM bisher eingeschaltet war.**

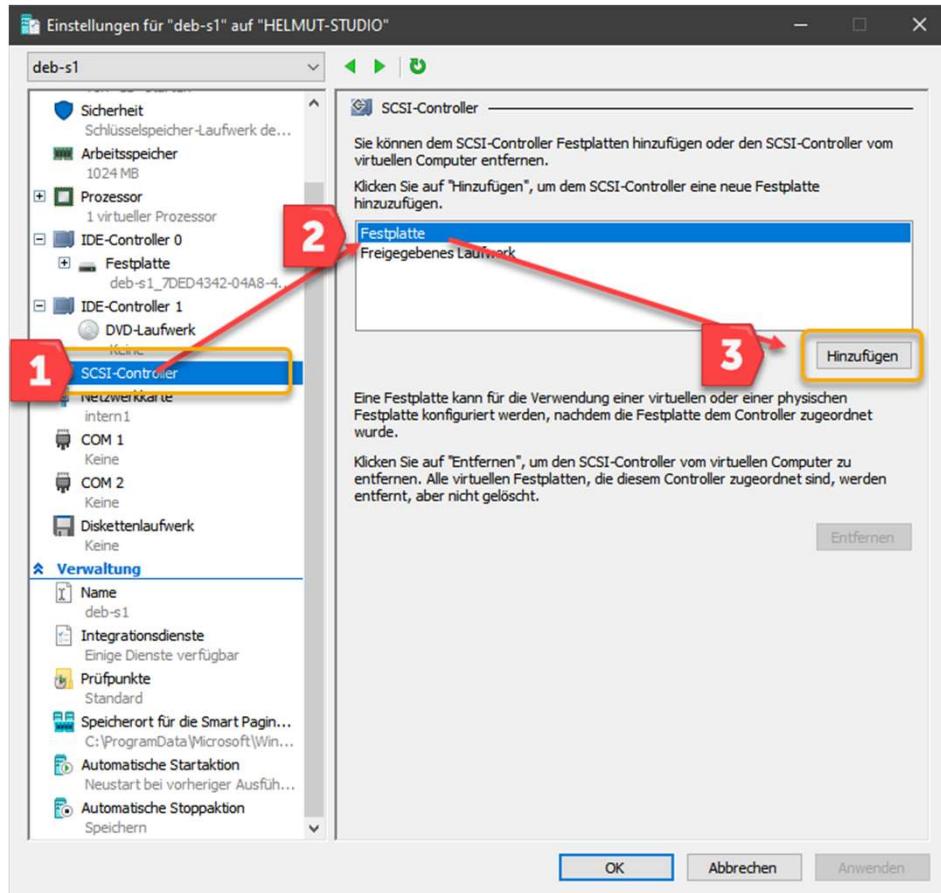
# VM auswählen



■ Wählen Sie „deb-s1“ aus und öffnen Sie die Einstellungen dieser VM.

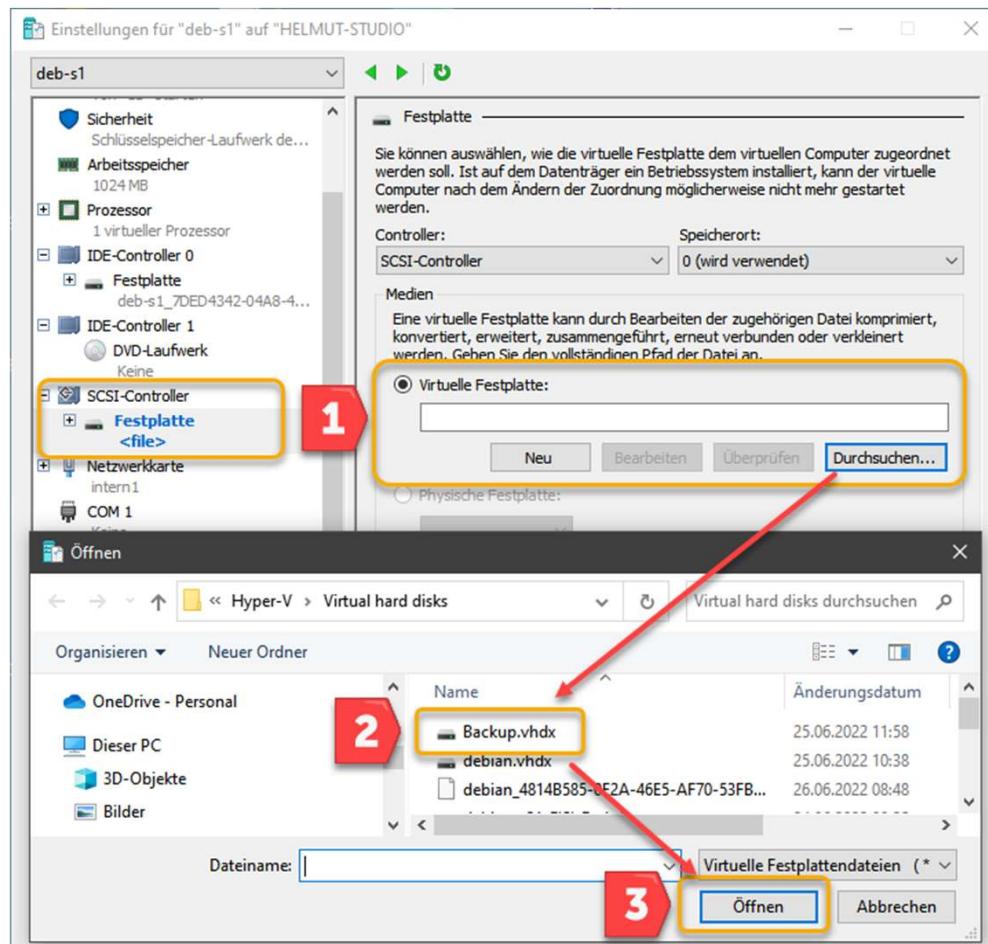
Linux

# SCSI-Controller



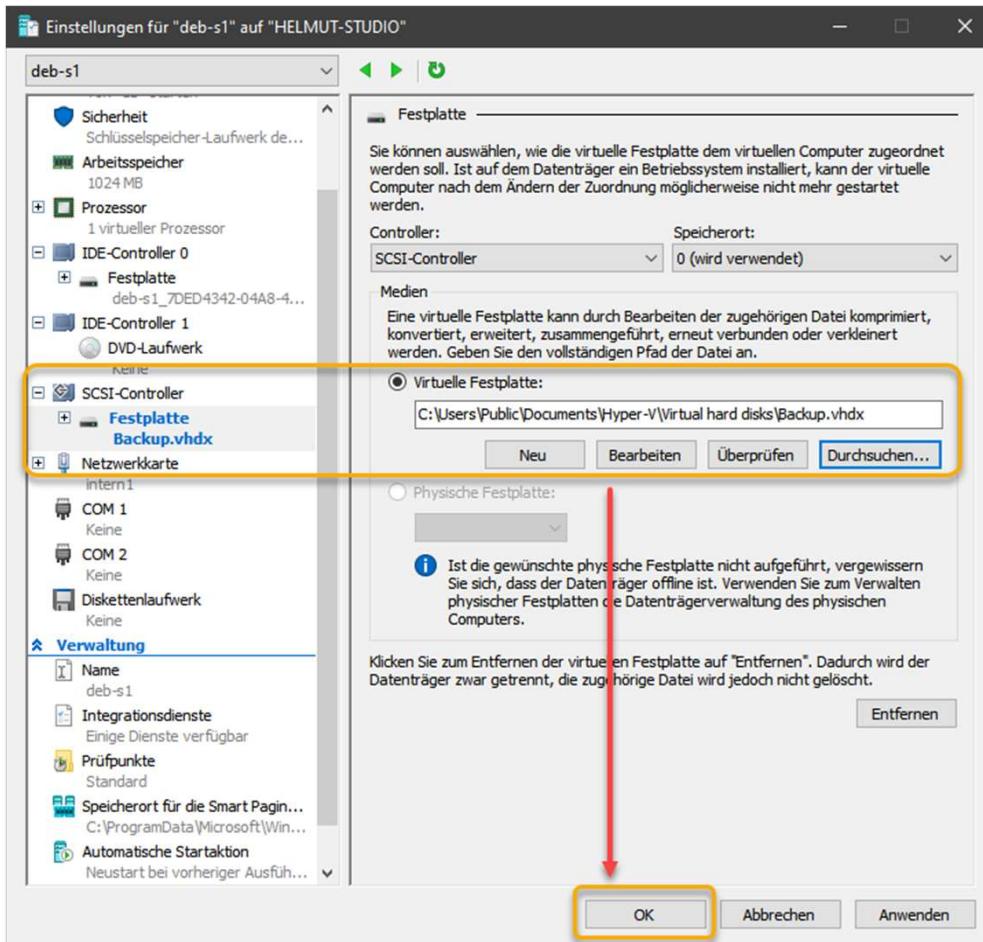
- Wählen Sie den SCSI-Controller aus. Egal, ob Sie für die vorhandene Festplatte einen IDE-Controller nutzen oder den SCSI-Controller
- Diesem können insgesamt 64 Datenträger zugeordnet werden
- Klicken Sie auf „Festplatte“ und danach auf die Schaltfläche <Hinzufügen>

# Virtuelle Festplatte suchen und einbinden



- Der Platzhalter für die Festplatte wird in der Hardwareliste angezeigt
- Durch Klick auf die Schaltfläche <Durchsuchen> landet man im Speicherort der vorhandenen virtuellen Festplatten
- Wählen Sie die Festplatte Backup.vhdx aus und klicken Sie auf <Öffnen>

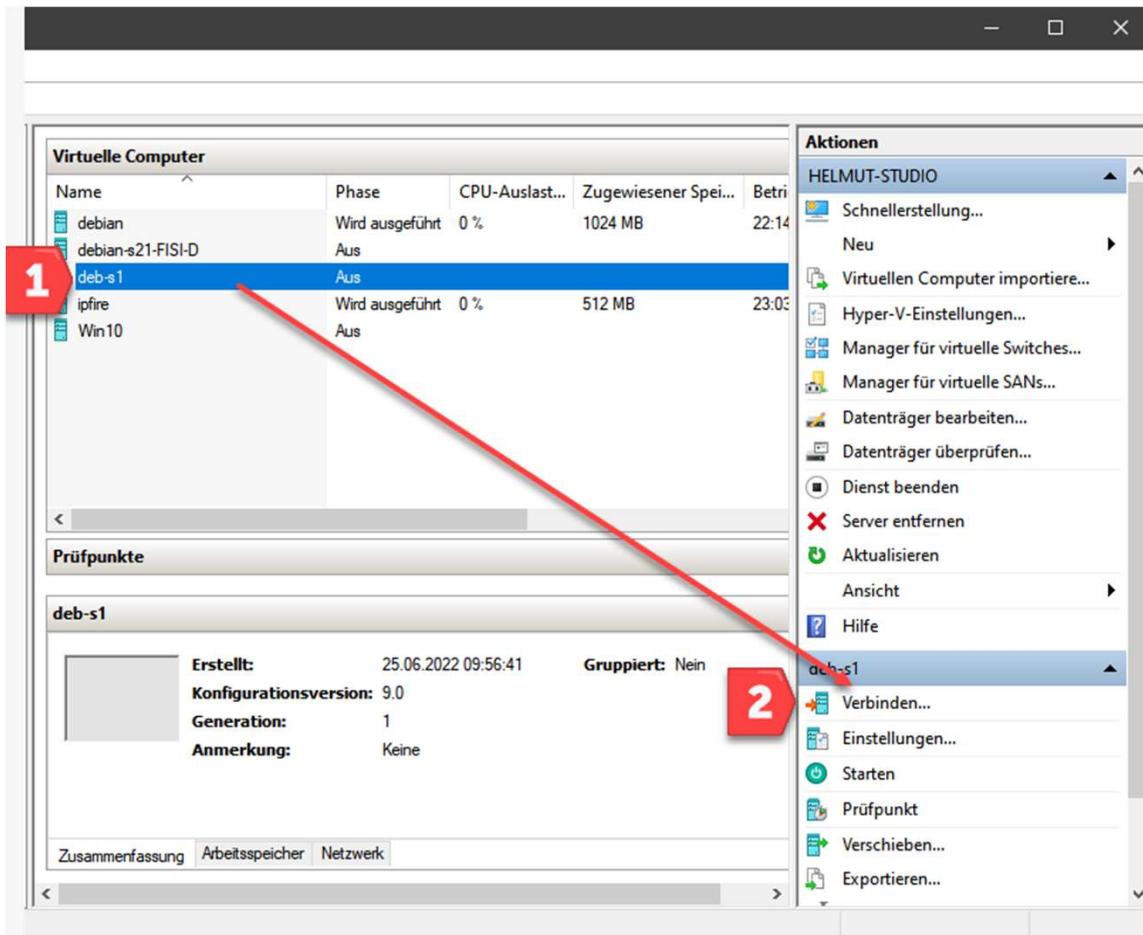
# Neue Festplatte bestätigen



- Kontrollieren Sie Ihre gewählten Einstellungen
- Bestätigen Sie Ihre Auswahl durch Klick auf <OK>

Linux

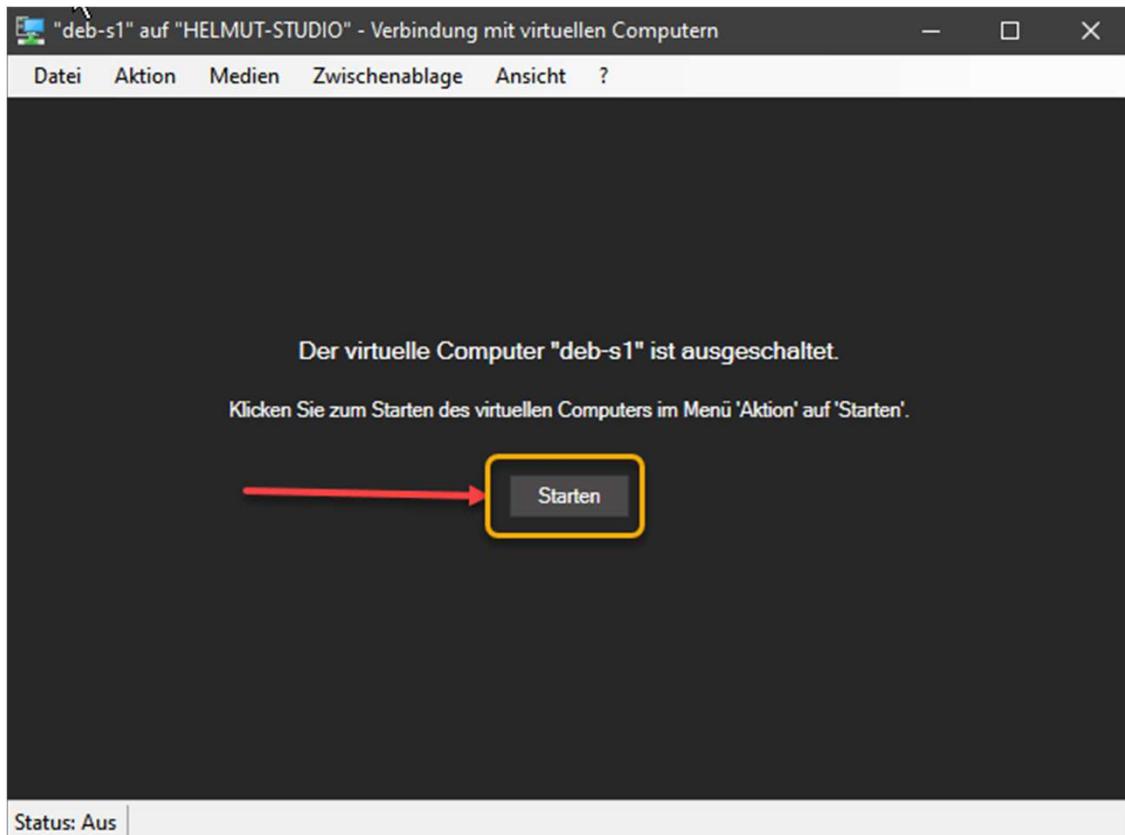
# VM verbinden



- Zeigen Sie Ihre VM „deb-s1“ wie gewohnt durch Klick auf „Verbinden“ an

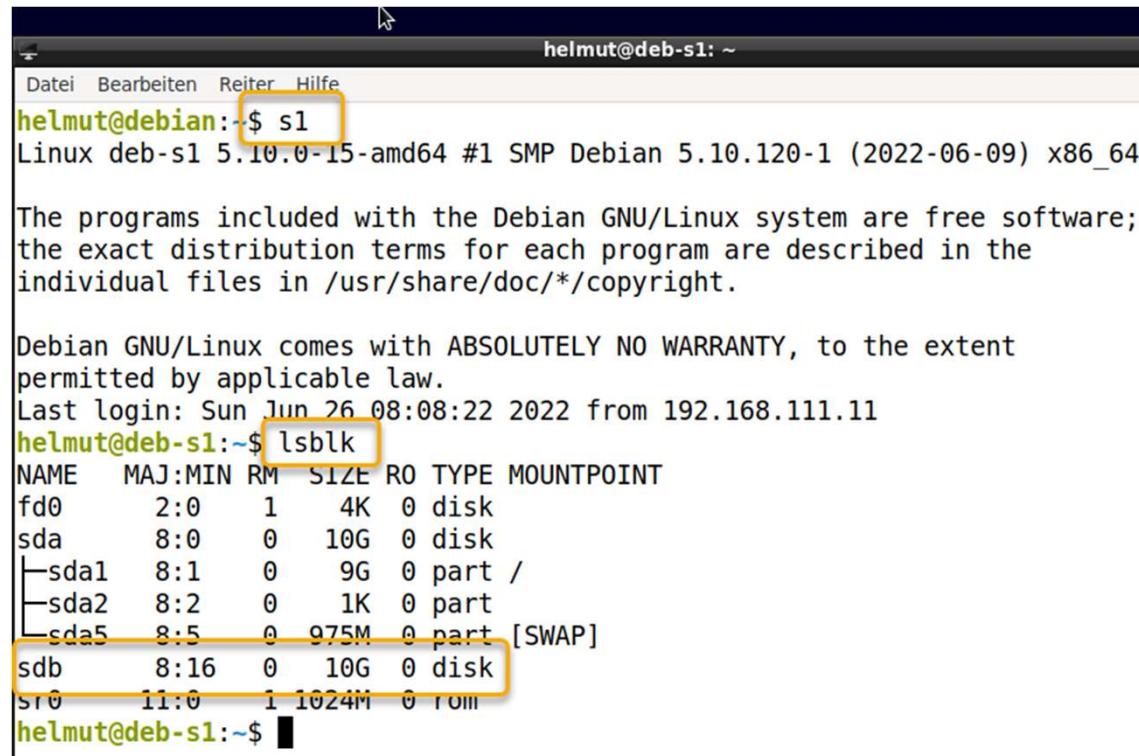
Linux

# VM starten



- **Starten Sie die VM durch Klick auf die gleichnamige Schaltfläche.**
- **Sie müssen sich nicht anmelden, weil die nachfolgenden Schritte über die ssh-Verbindung, vom Client aus, ausgeführt werden**

# Die neue Festplatte ist vorhanden



```
helmut@deb-s1:~$ s1
Linux deb-s1 5.10.0-15-amd64 #1 SMP Debian 5.10.120-1 (2022-06-09) x86_64
The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sun Jun 26 08:08:22 2022 from 192.168.111.11
helmut@deb-s1:~$ lsblk
NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
fd0      2:0    1   4K  0 disk 
sda      8:0    0   10G  0 disk 
└─sda1   8:1    0   9G  0 part /
└─sda2   8:2    0   1K  0 part 
└─sda5   8:5    0  975M 0 part [SWAP]
sdb      8:16   0   10G  0 disk 
sr0     11:0    1 1024M 0 rom 

helmut@deb-s1:~$
```

- Stellen Sie von der Client-VM „debian“ durch Eingabe des Alias-Namens „s1“ im Terminal die Verbindung zur Server-VM „deb-s1“ her
- Kontrollieren Sie zunächst mit „lsblk“, ob unsere neue Festplatte korrekt eingebunden ist.
- Sie sollte als „sdb“ mit der Größe 10 GB angezeigt werden

# Festplatte partitionieren



The screenshot shows a terminal window with the following content:

```
helmut@deb-s1: ~
Datei Bearbeiten Reiter Hilfe
helmut@deb-s1:~$ su -
Passwort:
root@deb-s1:~# fdisk /dev/sdb

Welcome to fdisk (util-linux 2.36.1).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.

Device does not contain a recognized partition table.
Created a new DOS disklabel with disk identifier 0xdc42eb46.

Command (m for help):
```

The command `su -` and the command `fdisk /dev/sdb` are highlighted with yellow boxes.

- Zunächst müssen Sie sich als root, mit su - anmelden
- Zum Partitionieren benutzen wir hier den Befehl „fdisk“
- Der Speicherort der Festplatte(n) in Linux ist der Ordner /dev
  - `fdisk /dev/sdb`
- Die Taste „m“ zeigt uns die Buchstaben an, mit denen fdisk bedient wird

# GPT-Partitionstabelle erstellen

```
Generic
d  delete a partition
F  list free unpartitioned space
l  list known partition types
n  add a new partition
p  print the partition table
t  change a partition type
v  verify the partition table
i  print information about a partition

Misc
m  print this menu
u  change display/entry units
x  extra functionality (experts only)

Script
I  load disk layout from sfdisk script file
O  dump disk layout to sfdisk script file

Save & Exit
w  write table to disk and exit
q  quit without saving changes

Create a new label
g  create a new empty GPT partition table
G  create a new empty SGI (IRIX) partition table
o  create a new empty DOS partition table
```

- Hier der Auszug der für uns wichtigen Tasten
- Zunächst werden wir die Festplatte mit einer modernen Partition Table (GPT) einrichten
- Damit ist es möglich bis zu 128 Partitionen zu erstellen
- Drücken Sie dazu die Taste „g“

# Eine neue Partition erstellen

## Generic

```
d delete a partition  
F list free unpartitioned space  
l list known partition types  
n add a new partition  
p print the partition table  
t change a partition type  
v verify the partition table  
i print information about a partition
```

## Misc

```
m print this menu  
u change display/entry units  
x extra functionality (experts only)
```

## Script

```
I load disk layout from sfdisk script file  
O dump disk layout to sfdisk script file
```

## Save & Exit

```
w write table to disk and exit  
q quit without saving changes
```

## Create a new label

```
g create a new empty GPT partition table  
G create a new empty SGI (IRIX) partition table  
o create a new empty DOS partition table
```

- Die GPT-Partitionstabelle wird sofort erstellt
- Nun soll eine neue Partition erstellt werden. Drücken Sie dazu die Taste „n“

# Die neue Partition ist erstellt

```
Command (m for help): g
```

```
Created a new GPT disklabel (GUID: E274C8A4-0587-2B4F-9626-F069D0CF0D1B).
```

```
Command (m for help): n
```

```
Partition number (1-128, default 1): Enter
```

```
First sector (2048-20971486, default 2048): Enter
```

```
Last sector, +/-sectors or +/-size{K,M,G,T,P} (2048-20971486, default 20971486): Enter
```

```
Created a new partition 1 of type 'Linux filesystem' and of size 10 GiB.
```

```
Command (m for help): ■
```

- Das Erstellen der GPT-Partitionstabelle wird angezeigt
- Nun soll eine neue Partition erstellt werden. Drücken Sie dazu die Taste „n“
- Da wir nur eine einzige Partition erstellen wollen, bestätigen Sie alle Vorgaben mit der <Enter>-Taste

# Partition speichern und Kontrolle

```
Command (m for help): w
The partition table has been altered.
Calling ioctl() to re-read partition table.
Syncing disks.
```

```
root@deb-s1:~# lsblk
NAME   MAJ:MIN RM  SIZE R0 TYPE MOUNTPOINT
fd0      2:0    1    4K  0 disk
sda      8:0    0   10G  0 disk
└─sda1   8:1    0    9G  0 part /
└─sda2   8:2    0    1K  0 part
└─sda5   8:5    0  975M  0 part [SWAP]
sdb      8:16   0   10G  0 disk
└─sdb1   8:17   0   10G  0 part
sr0     11:0    1 1024M  0 rom
root@deb-s1:~#
```

- Durch Drücken der Taste „w“ speichern Sie Ihre Einstellungen, wodurch „fdisk“ beendet wird
- Führen Sie wiederum „lsblk“ aus, um die erstellte Partition „sdb1“ zu sehen

# ext4-Filesystem für die Partition

```
root@deb-s1:~# mkfs.ext4 /dev/sdb1
mke2fs 1.46.2 (28-Feb-2021)
Discarding device blocks: done
Creating filesystem with 2621179 4k blocks and 655360 inodes
Filesystem UUID: 2e8f4fca-db4a-4514-a286-af3756b4a331
Superblock backups stored on blocks:
    32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632
Allocating group tables: done
Writing inode tables: done
Creating journal (16384 blocks): done
Writing superblocks and filesystem accounting information: done

root@deb-s1:~# █
```

- Die erstellte Partition sdb1 benötigt noch ein Filesystem
- Die meisten Distributionen benutzen als Standard das Filesystem „ext4“
  - Mit `mkfs.ext4` und der Partitionsangabe richten Sie es ein, also:  
`mkfs.ext4 /dev/sdb1`
- Jetzt kann die Partition gemountet werden

# Die Partition sdb1 in /backup mounten

```
root@deb-s1:~# mkdir /backup
root@deb-s1:~# ls /
backup  bin  dev  home  initrd
bilder  boot  etc  homeverzeichnisse.tar  initrd
root@deb-s1:~# mount /dev/sdb1 /backup/
root@deb-s1:~# ls /backup/
lost+found
root@deb-s1:~# umount /backup/
root@deb-s1:~# ls /backup/
root@deb-s1:~# █
```

- Falls das Verzeichnis /backup noch nicht existiert, legen Sie es mit mkdir an, also: mkdir /backup
- Mit ls / können Sie es kontrollieren
- Jetzt können Sie Ihre neue Partition in den Ordner /backup einbinden, so dass alle Sicherungen in die Partition sdb1 geschrieben werden
- Mit umount /dev/sdb1 oder umount /backup kann man die Verbindung trennen

# sdb1 in /backup dauerhaft mounten

- Möchte man den Mountprozess dauerhaft und schon beim Systemstart ausführen, so trägt man den Mountbefehl in die Datei /etc/fstab ein
  - fstab – Filesystemtable
  - Allerdings sind zusätzliche Parameter notwendig
  - Der Aufruf geschieht mit nano /etc/fstab unterhalb der vorhandenen Einträge ggf. mit Kommentar

```
# /dev/sdb1 in /backup mounten
/dev/sdb1      /backup      ext4      defaults      0      0
```

- Die Abstände erreicht man mit Tabstopps
- Wie üblich wird im nano danach mit <Strg> + <O> gespeichert, danach mit <Enter> bestätigt und nano verlassen mit <Strg> + <X>
- Mit mount -a ( oder --all) wird der Mountprozess in /etc/fstab ausgeführt, ein Neustart ist nicht notwendig. Es wird sogar davon abgeraten, weil der Rechner im Fehlerfall ggf. nicht booten kann

# Die zusätzlichen Parameter

- Hinter dem Mountpoint gibt man das verwendete Dateisystem (ext4) an. Ist man sich nicht sicher, kann man stattdessen auch „auto“ eintragen. Dann wird das Dateisystem automatisch gesucht
- Der Eintrag „defaults“ verwendet allgemein gültige Voreinstellungen wie:
  - rw – Lese- und Schreibzugriff sind erlaubt
  - uid – Programme mit SUID- und SGID-Bit werden ausgeführt
  - dev – Gerätedateien auf diesem Dateisystem werden erkannt
  - exec – Binärdateien können problemlos ausgeführt werden
  - auto – Das Dateisystem wird automatisch beim Systemstart oder mit mount –a gemountet
  - nouser – Ausschließlich root darf das Dateisystem mounten
  - async – Schreibvorgänge können zum Zweck höherer Performance gepuffert werden
- Der Eintrag „dump“ (Wert = 0) ist heute bedeutungslos (früher Erstellen einer Sicherung)
- Der Eintrag „pass“ (Wert = 0) legt fest, ob die Partition in bestimmten Abständen geprüft wird ( 0 – nein)

```
# /dev/sdb1 in /backup mounten  
/dev/sdb1      /backup      ext4      defaults      0      0
```

# Verwenden der UUID

- Eine UUID (Universally Unique Identifier) ist eine unverwechselbare, sich nicht ändernde Nummerierung für blockorientierte Geräte
  - Selbst wenn die zugehörige blockorientiertes Gerät an einen anderen Controller angeschlossen wird, ändert sich ihre UUID nicht, die Gerätebezeichnung könnte sich ändern (z.B. aus sdb könnte sdc werden)
  - Mit dem Befehl blkid kann man sich alle UUID's der blockorientierten Geräte auflisten lassen

```
root@deb-s1:~# blkid  
/dev/sda1: UUID="CC61-F349" BLOCK_SIZE="512" TYPE="vfat" PARTUUID="c4c70f6f-4f31-4eb  
/dev/sda2: UUID="f4ea4518-8524-4e38-9e64-eccb6108173f" BLOCK_SIZE="4096" TYPE="ext4"  
/dev/sda3: UUTD="53f712f1-ae73-41a3-a625-e41309e105dd" TYPE="swap" PARTUUID="11367a9  
/dev/sdb1: UUID="694ce79f-1232-4807-b012-246853231b17" BLOCK_SIZE="4096" TYPE="ext4"
```

- Die zugehörige UUID wird anstelle des Ausdrucks /dev/sdb1 in die Datei /etc/fstab eingetragen

```
# /dev/sdb1 in /backup mounten  
UUID=694ce79f-1232-4807-b012-246853231b17      /backup      ext4      defaults      0      0
```

# **RAID 5 Projekt**

Software-RAID

Linux



# RAID 5-Projekt

- Im folgenden Abschnitt soll ein Software-RAID (RAID 5) erstellt werden
- Dazu verwenden wir die gleiche virtuelle Festplatte sdb
- Die vorhandene Partition sdb1 umfasst ja den gesamten Festplatten-Speicherbereich und muss daher gelöscht werden
- Den danach freien Speicherplatz der Festplatte teilen wir dann in 4 gleich große Partitionen auf (obwohl die vierte Partition ein wenig größer wird)
  - Die ersten drei Partitionen sdb1 bis sdb3 bilden das RAID 5 und die vierte Partition steht als Spare-Disk (Hot-Spare) als Ersatz für eine defekte Partition zur Verfügung
  - Was wie hier mit Partitionen einer Festplatte üben, wird man in der Praxis mit kompletten, gleich großen partitionierten Festplatten ausführen

# Die Partition sdb1 löschen

```
root@deb-s1:~# fdisk /dev/sdb
```

```
Welcome to fdisk (util-linux 2.36.1).  
Changes will remain in memory only, until you decide to write them.  
Be careful before using the write command.
```

```
Command (m for help): p  
Disk /dev/sdb: 10 GiB, 10737418240 bytes, 20971520 sectors  
Disk model: VBOX HARDDISK  
Units: sectors of 1 * 512 = 512 bytes  
Sector size (logical/physical): 512 bytes / 512 bytes  
I/O size (minimum/optimal): 512 bytes / 512 bytes  
Disklabel type: gpt  
Disk identifier: 72DE3AEB-F9E8-FC40-927B-34C5A556E738
```

Device	Start	End	Sectors	Size	Type
/dev/sdb1	2048	20971486	20969439	10G	Linux filesystem

```
Command (m for help): d  
Selected partition 1  
Partition 1 has been deleted.
```

```
Command (m for help): p  
Disk /dev/sdb: 10 GiB, 10737418240 bytes, 20971520 sectors  
Disk model: VBOX HARDDISK  
Units: sectors of 1 * 512 = 512 bytes  
Sector size (logical/physical): 512 bytes / 512 bytes  
I/O size (minimum/optimal): 512 bytes / 512 bytes  
Disklabel type: gpt  
Disk identifier: 72DE3AEB-F9E8-FC40-927B-34C5A556E738
```

```
Command (m for help): q
```

- Zum Löschen der Partition sdb1 kann man wiederum das Kommando fdisk nutzen (auch parted könnte genutzt werden)

- **fdisk /dev/sdb**
- mit **p** kann man sich den jetzigen Status der Partition anzeigen lassen
- Mit **d** löscht man die vorhandene Partition
- Danach kann man sich wiederum mit **p** vom Erfolg überzeugen

# Vier neue Partitionen erstellen

- Wenn man 10 GB durch 4 teilt erhält man 4x 2,5 GB. Dieser Wert wird mit +2500 M in fdisk eingetragen
- Die vierte Partition wird durch Ungenauigkeiten etwas größer als 2500 MB, was aber kein Problem darstellt
- Nur kleiner darf die vierte Partition nicht sein
  - Durch Eingabe von **n** erstellt man neue Partitionen
  - Den Startsektor bestätigt man mit **Enter**
  - Die Größe der Partition gibt man mit **+2500 M** an
- Vergessen Sie nicht den Erstellungsprozess nach der 4. Partition mit **w** zu speichern

# Vier neue Partitionen erstellen

```
Command (m for help): n
Partition number (1-128, default 1): Enter
First sector (2048-20971486, default 2048): Enter
Last sector, +/-sectors or +/-size{K,M,G,T,P} (2048-20971486, default 20971486): +2500M

Created a new partition 1 of type 'Linux filesystem' and of size 2,4 GiB.
```

```
Command (m for help): n
Partition number (2-128, default 2): Enter
First sector (5122048-20971486, default 5122048): Enter
Last sector, +/-sectors or +/-size{K,M,G,T,P} (5122048-20971486, default 20971486): +2500M

Created a new partition 2 of type 'Linux filesystem' and of size 2,4 GiB.
```

```
Command (m for help): n
Partition number (3-128, default 3): Enter
First sector (10242048-20971486, default 10242048): Enter
Last sector, +/-sectors or +/-size{K,M,G,T,P} (10242048-20971486, default 20971486): +2500M

Created a new partition 3 of type 'Linux filesystem' and of size 2,4 GiB.
```

```
Command (m for help): n
Partition number (4-128, default 4): Enter
First sector (15362048-20971486, default 15362048): Enter
Last sector, +/-sectors or +/-size{K,M,G,T,P} (15362048-20971486, default 20971486): Enter
```

```
Created a new partition 4 of type 'Linux filesystem' and of size 2,7 GiB.
```

```
Command (m for help): w
```

# Kontrolle der Erstellung

```
root@deb-s1:~# lsblk
NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sda      8:0    0   10G  0 disk 
└─sda1   8:1    0   9G  0 part /
└─sda2   8:2    0   1K  0 part 
└─sda5   8:5    0 975M 0 part [SWAP]
sdb      8:16   0   10G  0 disk 
└─sdb1   8:17   0 2,4G  0 part 
└─sdb2   8:18   0 2,4G  0 part 
└─sdb3   8:19   0 2,4G  0 part 
└─sdb4   8:20   0 2,7G  0 part 
sr0     11:0    1 1024M 0 rom
root@deb-s1:~#
```

- Zur Kontrolle kann man das Kommando `lsblk` nutzen

# Installation des Pakets mdadm

- Um ein Software-RAID zu erstellen, benötigt man das Paket mdadm
- Dieses Paket muss mit  
apt install mdadm  
installiert werden.
- Bitte denken Sie daran ihr System vor der Installation mit  
apt update && apt upgrade -y  
ggf. auf den neuesten Stand zu bringen

# Installation des Pakets mdadm

```
root@deb-s1:~# apt update && apt upgrade -y
OK:1 http://security.debian.org/debian-security bullseye-security InRelease
OK:2 http://deb.debian.org/debian bullseye InRelease
OK:3 http://deb.debian.org/debian bullseye-updates InRelease
Paketlisten werden gelesen... Fertig
Abhängigkeitsbaum wird aufgebaut... Fertig
Statusinformationen werden eingelesen... Fertig
Alle Pakete sind aktuell.
Paketlisten werden gelesen... Fertig
Abhängigkeitsbaum wird aufgebaut... Fertig
Statusinformationen werden eingelesen... Fertig
Paketaktualisierung (Upgrade) wird berechnet... Fertig
0 aktualisiert, 0 neu installiert, 0 zu entfernen und 0 nicht aktualisiert.
root@deb-s1:~# apt install mdadm
Paketlisten werden gelesen... Fertig
Abhängigkeitsbaum wird aufgebaut... Fertig
Statusinformationen werden eingelesen... Fertig
Vorgeschlagene Pakete:
  dracut-core
Die folgenden NEUEN Pakete werden installiert:
  mdadm
```

# **Reboot ist notwendig**

- Hier tritt der seltene Fall ein, dass der Server nach der Installation von mdadm neu gestartet werden muss
- Das liegt daran, dass sich mdadm in den Kernel integriert und dieser den Dienst erst nach einem Neustart erkennt und ausführen kann
- Geben Sie als Benutzer root das Kommando  
reboot  
ein und verbinden sie sich nach angemessener Zeit wieder mit dem Server  
deb-s1 durch die Eingabe von s1
- Melden Sie sich danach wieder als root an (su -)

# Anzeige der Möglichkeiten von mdadm

- Nach dem Neustart wird mdadm auch im Ordner /proc erscheinen
- Mit  
`cat /proc/mdstat`  
kann man sich alle Möglichkeiten des Einsatzes von mdadm anzeigen lassen
- Natürlich ist auch der Verweis auf RAID 5 vorhanden

```
root@deb-s1:~# cat /proc/mdstat
Personalities : [linear] [multipath] [raid0] [raid1] [raid6] [raid5] [raid4] [raid10]
unused devices: <none>
root@deb-s1:~# █
```

# Einrichten des RAID 5

- Zu Einrichten unseres RAID 5 geben Sie folgende lange Befehlszeile ein:

```
root@deb-s1:~# mdadm --create --verbose /dev/md0 --level=raid5 --raid-devices=3 /dev/sdb1 /dev/sdb2 /dev/sdb3
mdadm: layout defaults to left-symmetric
mdadm: layout defaults to left-symmetric
mdadm: chunk size defaults to 512K
mdadm: size set to 2556928K
mdadm: Defaulting to version 1.2 metadata
mdadm: array /dev/md0 started.
root@deb-s1:~# █
```

- Würden wir mit 3 partitionierten Festplatten das RAID 5 erstellen, hießen die Devices am Zeilenende
  - /dev/sdb1 /dev/sdc1 und /dev/sdd1
- Das neue Gerät md0 ist jetzt im Ordner /dev erstellt worden
- Es beinhaltet unser RAID 5

# Formatierung von md0

- Unser RAID 5-System md0 muss natürlich noch formatiert werden
- Wir nutzen hier wieder das Linux-Format ext4

```
root@deb-s1:~# mkfs.ext4 /dev/md0
mke2fs 1.46.2 (28-Feb-2021)
Creating filesystem with 1277952 4k blocks and 320112 inodes
Filesystem UUID: 9818865c-9c55-4d66-98e2-04578d9aa92d
Superblock backups stored on blocks:
            32768, 98304, 163840, 229376, 294912, 819200, 884736

Allocating group tables: done
Writing inode tables: done
Creating journal (16384 blocks): done
Writing superblocks and filesystem accounting information: done

root@deb-s1:~# █
```

# Nutzung unseres RAID 5

- Unser RAID 5 kann jetzt temporär wieder in ein geeignetes Verzeichnis gemountet werden z. B. in /backup, wobei das Verzeichnis /backup schon existieren muss

```
root@deb-s1:~# mount /dev/md0 /backup
root@deb-s1:~# ls /backup
lost+found
root@deb-s1:~# █
```

- Mit umount /backup wird das Verzeichnis wieder freigegeben

```
root@deb-s1:~# umount /backup
root@deb-s1:~# ls /backup
helmut-home-Sicherung.tar  home-diff-mo.tgz
home-helmut-Sicherung.tar.gz
helmut-home-Sicherung.tgz  home-helmut-Sicherung.tar
home-mi.tgz
root@deb-s1:~# █
```

# Die 4. Partition

- Die noch übrige 4. Partition soll als Reserve für unser RAID 5 zur Verfügung gestellt werden. Dies geschieht mit dem Kommando:

```
root@deb-s1:~# mdadm --add /dev/md0 /dev/sdb4
mdadm: added /dev/sdb4
root@deb-s1:~# █
```

- Die Funktion als Spare-Disk bzw. Hot-Spare kann man wieder im Verzeichnis /proc ansehen:

```
root@deb-s1:~# cat /proc/mdstat
Personalities : [linear] [multipath] [raid0] [raid1] [raid6] [raid5] [raid4] [raid10]
md0 : active raid5 sdb4[4](S) sdb3[3] sdb2[1] sdb1[0]
      5113856 blocks super 1.2 level 5, 512k chunk, algorithm 2 [3/3] [UUU]

unused devices: <none>
root@deb-s1:~# █
```

# RAID 5 dauerhaft mounten

- Wenn man einen separaten Rechner im Netzwerk als Backupsystem eingerichtet hat (z. B. mit einem Raspberry-Pi und 4 Festplatten), ist es sinnvoll das RAID 5-System mit dem Rechnerstart des Raspberry-Pi dauerhaft zu mounten
- Hierfür ist die Datei fstab zuständig, die sich im Verzeichnis /etc befindet
- Anders als beim temporären mounten verweigert fstab die Arbeit mit /dev/md0
- Wir müssen die UUID von md0 in die Datei fstab eintragen
- Die UUIDs der Festplatten erhält man mit dem Kommando blkid
- Aus der Liste kopiert man sich die UUID von md0 und fügt sie anschließend in die Datei fstab ein (Die Einträge aus dem Kontextmenü sind dabei nutzbar)

# UUID von md0 anzeigen und kopieren

```
root@deb-s1:~# blkid  
/dev/md0: UUID="9818865c-9c55-4d66-98e2-04578d9aa92d" BLOCK_SIZE="4096" TYPE="ext4"  
/dev/sda1: UUID="2093155e-e605-4cd6-9306-a49dd581d578" BLOCK_SIZE="4096" TYPE="ext4" PARTUUID="ed27ec6f-05"  
/dev/sda5: UUID="46100afa-7e78-4e66-a456-07a48b38a002" TYPE="swap" PARTUUID="ed27ec6f-05"  
/dev/sdb1: UUID="ea88385f-ac1a-b50d-7495-67657f5b377b" UUID_SUB="ae07422d-def5-d993-defe-d33nux_raid_member" PARTUUID="0fbe76bc-b4fe-674a-a61a-a2698c21f8f2"  
/dev/sdb2: UUID="ea88385f-ac1a-b50d-7495-67657f5b377b" UUID_SUB="9201988f-8800-93d3-6e4a-188nux_raid_member" PARTUUID="af16f9cc-7f55-824c-9603-17bcfc021c2c"  
/dev/sdb3: UUID="ea88385f-ac1a-b50d-7495-67657f5b377b" UUID_SUB="01089dfa-f378-089b-8f87-5e5nux_raid_member" PARTUUID="ae6be650-9f93-4f4a-af32-b01c0ff649d2"  
/dev/sdb4: UUID="ea88385f-ac1a-b50d-7495-67657f5b377b" UUID_SUB="4621b8de-4b0c-2fa8-ffce-550nux_raid_member" PARTUUID="b8cfed5e-814f-4b4b-9b96-f15265c1e6fd"  
root@deb-s1:~# █
```

# UUID von md0 in fstab eintragen

- In der Datei fstab müssen neben der UUID vom md0 auch die übrigen Parameter eingetragen werden z. B. der Einhängepunkt (mount-point)
  - Im Beispiel wird wieder das Verzeichnis /backup als Einhängepunkt benutzt

```
Datei Bearbeiten Reiter Hilfe
GNU nano 5.4                               /etc/fstab *
# /etc/fstab: static file system information.
#
# Use 'blkid' to print the universally unique identifier for a
# device; this may be used with UUID= as a more robust way to name devices
# that works even if disks are added and removed. See fstab(5).
#
# systemd generates mount units based on this file, see systemd.mount(5).
# Please run 'systemctl daemon-reload' after making changes here.
#
# <file system> <mount point> <type> <options>           <dump>   <pass>
# / was on /dev/sda1 during installation
UUID=2093155e-e605-4cd6-9306-a49dd581d578 /           ext4    errors=remount-ro 0      1
# swap was on /dev/sda5 during installation
UUID=46100afa-7e78-4e66-a456-07a48b38a002 none        swap     sw          0      0
/dev/sr0       /media/cdrom0 udf,iso9660 user,noauto 0      0
# Einbinden des RAID5-Systems md0
UUID=9818865c-9c55-4d66-98e2-04578d9aa92d      /backup ext4    defaults      0      1
```

# Kontrolle

- Nach dem reboot des Servers deb-s1 muss dieser problemlos starten
- Danach kann man sich wieder mit dem Server über ssh verbinden (Eingabe: s1)
- Und mit lsblk sieht man alle 4 RAID-Partitionen und den Einhängepunkt /backup

```
helmut@deb-s1:~$ lsblk
NAME   MAJ:MIN RM  SIZE RO TYPE  MOUNTPOINT
sda      8:0    0   10G  0 disk
└─sda1   8:1    0    9G  0 part   /
└─sda2   8:2    0    1K  0 part
└─sda5   8:5    0  975M 0 part [SWAP]
sdb      8:16   0   10G  0 disk
└─sdb1   8:17   0  2,4G 0 part
  └─md127 9:127  0  4,9G 0 raid5 /backup
└─sdb2   8:18   0  2,4G 0 part
  └─md127 9:127  0  4,9G 0 raid5 /backup
└─sdb3   8:19   0  2,4G 0 part
  └─md127 9:127  0  4,9G 0 raid5 /backup
└─sdb4   8:20   0  2,7G 0 part
  └─md127 9:127  0  4,9G 0 raid5 /backup
sr0     11:0   1 1024M 0 rom
helmut@deb-s1:~$ █
```