Linux Advanced

Lab Guide

Formatierung

Format	Beschreibung
Note	Wichtiger Hinweis
`command`	Shell-Ausgabe oder Kommando-Name

Gehostetes Labor

Für das Training können entsprechende Hosts in einer Cloud-Umgebung (*z.B. Hetzner*) bereitgestellt worden sein. Die Verbindungsinformationen werden vom/von der Trainer:in vorab verteilt.

Lokales Labor

Für ein lokales Labor bitte die folgenden Programme installieren:

- Oracle VirtualBox (https://www.virtualbox.org)
- Vagrant (https://www.vagrantup.com/)

Ablauf

1. Git-Repo klonen bzw. ZIP-Archiv herunterladen und entpacken
2. Kommandozeile öffnen und in den neuen Ordner wechseln
3. Wechseln in den `environment/vagrant`-Ordner

Eingeben des Befehls:

```
$ vagrant up --provision
```

Das Bereitstellen der VMs dauert einige Minuten!

`vagrant`-Kommandos

Kommando	Erklärung
`vagrant ssh ubuntu,almalinux`	Login auf der Konsole
`vagrant up`	Umgebung starten
`vagrant up --provision`	Bereiststellung erneut starten
`vagrant ssh [name]`	Login auf der Konsole
`vagrant halt`	Herunterfahren der Umgebung
`vagrant snapshot`	Verwalten von Snapshots
`vagrant destroy`	Löschen der Umgebung

Weitere Informationen: `vagrant --help`

Musterlösungen

Für einige Aufgaben der Schulung gibt es im Unterordner `labs` Musterlösungen. Auf den Labor-Systemen sind diese Dateien unterhalb des Ordners `/labs` erreichbar.

Die Lösungen bestehen aus einem Ordner, in welchem sich ein Skript befindet. Um die Lösung anzuwenden, genügt es in den Ordner zu wechseln und das Skript auszuführen.

```
$ /labs/01/node1.sh
```

Für manche Aufgaben gibt es auch Tests zur Überprüfung, ob die Aufgabe korrekt ausgeführt wurde. Diese werden über das `lab`-Kommando ausgeführt, beispielsweise:

```
Checking lab 01

[2/2] Checking extract... Success!
[1/2] Checking archives... Success!
All tasks correct!
```

Übersicht

Lab D01: Dienste verwalten

Ziel:

- Login auf node1
- Gestartete Dienste auflisten
- Status des Dienstes `crond.service` anzeigen
- Dienst stoppen und Status erneut anzeigen
- Dienst starten und Status erneut anzeigen
- Gestoppte Dienste anzeigen

Vorgehensweise

Auflisten gestarteter Dienste:

```
# systemctl --type=service
```

Status des Dienstes `crond.service` anzeigen:

```
# systemctl status crond.service
```

Dienst stoppen:

```
# systemctl stop crond.service
# systemctl status crond.service
```

Dienst starten:

```
# systemctl start crond.service
# systemctl status crond.service
```

Gestoppte Dienste anzeigen:

```
# systemctl --type=service --state=inactive
```

Lab D02: Systemumfang konfigurieren

Ziel:

- Login auf node1
- Aktuelles Target inkl. Abhängigkeiten anzeigen
- Verfügbare Targets auflisten
- Startzeit des Systems analysieren und auffälligen Dienst deaktivieren

Vorgehensweise

Aktuelles Target **anzeigen**:

```
# systemctl get-default
multi-user.target

# systemctl status default.target
● multi-user.target - Multi-User System
     Loaded: loaded (/usr/lib/systemd/system/multi-user.target; indirect;
preset: disabled)
     Active: active since Mon 2025-01-13 14:59:58 UTC; 16min ago
      Until: Mon 2025-01-13 14:59:58 UTC; 16min ago
       Docs: man:systemd.special(7)
```

Das aktuelle Target lautet `multi-user.target`.

Weitere Targets **anzeigen**:

```
# systemctl list-units --type target
  UNIT                   LOAD   ACTIVE SUB    DESCRIPTION
  basic.target           loaded active active Basic System
  cryptsetup.target      loaded active active Local Encrypted Volumes
  getty.target           loaded active active Login Prompts
...
```

Abhängigkeiten des aktuellen Targets anzeigen:

```
# systemctl list-dependencies multi-user.target
multi-user.target
● ├─auditd.service
● ├─chronyd.service
● ├─crond.service
...
```

Startzeiten aktivierter systemd-Units auflisten:

```
# systemd-analyze blame
22.006s slow-app.service
 2.162s dev-disk-by\x2dpath-pci\x2d0000:00:04.0.device
 2.162s dev-vdb.device
 2.162s dev-disk-by\x2ddiskseq-2.device
...
```

Auffällig ist, dass der Dienst `slow-app.service` bedeutend mehr Zeit zum Starten benötigt und so den **Bootvorgang** verlangsamt. Die Unit kann wie folgt deaktiviert werden:

```
# systemctl disable --now slow-app.service
```

Überprüfen, dass der Dienst deaktiviert wurde:

```
# systemctl status slow-app.service
○ slow-app.service - The infamous slow(TM) application
     Loaded: loaded (/etc/systemd/system/slow-app.service; disabled; preset:
disabled)
     Active: inactive (dead)
...
```

Lab D03: Units erstellen

Ziel:

- Login auf node1 oder node2
- Service-Unit `passwd-log.service` erstellen
    - Startet ein Skript, welches eine Nachricht in ein Protokoll schreibt, sobald ein neuer User angelegt wurde
- Path-Unit `passwd-mon.path` erstellen
    - Reagiert auf Änderungen in der Datei `/etc/passwd`
    - Löst die Unit `passwd-log.service` aus
- Mount-Unit `ramdisk.mount` erstellen
    - Hängt eine kleine RAM-Disk (200 MB) nach `/ramdisk` ein

Vorgehensweise

Datei `/etc/systemd/system/passwd-log.service` erstellen:

```
[Unit]
Description="Write alarm to passwd log"

[Service]
ExecStart=/usr/local/bin/passwd-alert.sh
```

Datei `/usr/local/bin/passwd-alert.sh` erstellen und ausführbar machen:

```
#!/bin/sh
echo "ALARM ALARM" >> /var/log/passwd.alert
echo "New user created" >> /var/log/passwd.alert
```

```
# chmod +x /usr/local/bin/passwd-alert.sh
```

Path-Unit `/etc/systemd/system/passwd.path` erstellen:

```
[Unit]
Description="Monitor /etc/passwd for changes"

[Path]
PathModified=/etc/passwd
Unit=passwd-log.service

[Install]
WantedBy=multi-user.target
```

Änderungen einlesen:

```
# systemctl daemon-reload
# systemctl enable --now passwd.path
```

User anlegen und Log überprüfen:

```
# useradd dummy
# cat /var/log/passwd.alert
ALARM ALARM
New user created
```

Erstellen der Datei `/etc/systemd/system/ramdisk.mount`:

```
[Unit]
Description=Mount smol ramdisk

[Mount]
What=none
Where=/ramdisk
Type=tmpfs
Options=size=200M

[Install]
WantedBy=multi-user.target
```

Änderungen einlesen und Mount einhängen:

```
# systemctl daemon-reload
# systemctl start ramdisk.mount
# df -h /ramdisk
```

Lab D04: Timer verwalten

Ziel:

- Login auf node2
- Sichten aktueller Timer und deren Logs
- Anlegen einer Unit `ping.service`
    - startet ein Skript `/usr/local/bin/ping.sh`
        - schreibt pong in die Datei `/var/log/ping`
- Anlegen eines Timers `ping.timer`
    - startet minütlich die Unit `ping.service`
- Starten und Überprüfen des Timers

Vorgehensweise

Sichten aktueller Timer:

```
# systemctl list-timers
```

Sichten der Logs eines Timers:

```
# journalctl -u logrotate.timer
```

Datei `/etc/systemd/system/ping.service` erstellen:

```
[Unit]
Description="Write ping to log"

[Service]
ExecStart=/usr/local/bin/ping.sh
```

Datei `/usr/local/bin/ping.sh` erstellen und ausführbar machen:

```
#!/bin/sh
echo "pong" >> /var/log/ping
```

```
# chmod +x /usr/local/bin/ping.sh
```

Anlegen des Timers `/etc/systemd/system/ping.timer:

```
[Unit]
Description=Ping every minute

[Timer]
OnCalendar=*:0/1
Unit=ping.service

[Install]
WantedBy=timers.target
```

Konfiguration übernehmen und Timer aktivieren:

```
# systemctl daemon-reload
# systemctl enable --now ping.timer
```

Status des Timers anzeigen und Log ausgeben:

```
# systemctl list-timers ping.timer
# cat /var/log/ping
pong
```

Lab D05: Cronjobs verwenden

Ziel:

- Login auf node1
- Sichten aktueller Cronjobs und der dazugehörigen Konfigurationen
- Erstellen eines systemweiten Cronjobs
    - Protokollieren der Ausgabe des `uptime`-Kommandos
    - alle 5 Minuten, in das Log `/var/log/5min-load`
- Erstellen eines User-Cronjobs
    - minütliches Protokollieren der Auslastung des Heimatverzeichnisses (`~/disk_usage.log`)

Vorgehensweise

Überprüfen, dass `crond.service` ausgeführt und automatisch beim Boot gestartet wird:

```
# systemctl is-active crond.service
active

# systemctl is-enabled crond.service
enabled
```

Betrachten der letzten Logs:

```
# journalctl -u crond.service
...
Jan 14 09:00:41 localhost.localdomain systemd[1]: Started Command Scheduler.
...
Jan 14 09:01:01 0-node1.sva.de anacron[4543]: Will run job `cron.daily' in 6
min.
Jan 14 09:01:01 0-node1.sva.de anacron[4543]: Will run job `cron.weekly' in 26
min.
Jan 14 09:01:01 0-node1.sva.de anacron[4543]: Will run job `cron.monthly' in 46
min.
Jan 14 09:01:01 0-node1.sva.de anacron[4543]: Jobs will be executed
sequentially
Jan 14 09:01:01 0-node1.sva.de run-parts[4545]: (/etc/cron.hourly) finished
0anacron
Jan 14 09:07:01 0-node1.sva.de anacron[4543]: Job `cron.daily' started
Jan 14 09:07:01 0-node1.sva.de anacron[4543]: Job `cron.daily' terminated
Jan 14 09:27:01 0-node1.sva.de anacron[4543]: Job `cron.weekly' started
Jan 14 09:27:01 0-node1.sva.de anacron[4543]: Job `cron.weekly' terminated
Jan 14 09:47:01 0-node1.sva.de anacron[4543]: Job `cron.monthly' started
Jan 14 09:47:01 0-node1.sva.de anacron[4543]: Job `cron.monthly' terminated
```

Drei Jobs (`cron.daily`, `cron.weekly` und `cron.monthly`) wurden gestartet und gestoppt - vermutlich, weil sie leer sind. Eine Konfigurationsdatei (`/etc/cron.hourly`) wurde ausgelesen.

Anzeigen der systemweiten Konfigurationsdatei `/etc/crontab` des klassischen Cron-Dienstes:

```
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root

# For details see man 4 crontabs

# 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
```

Die Konfigurationsdatei ist leer und erläutert lediglich den Syntax. Für konfigurierte Jobs werden einige Umgebungsvariablen definiert (`SHELL`, `PATH`, `MAILTO`).

Auflisten des dazugehörigen Ordners `/etc/cron.d`:

```
# ls /etc/cron.d/
0hourly

# cat /etc/cron.d/0hourly
# Run the hourly jobs
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
01 * * * * root run-parts /etc/cron.hourly

# ls /etc/cron.hourly
0anacron
```

Der Ordner enthält eine Datei, die wiederrum verschiedene Jobs aus einem weiteren Ordner (`/etc/cron.hourly`) ausführt.

Während der klassische Cron für die Ausführung von Jobs zu **bestimmten Zeiten** gedacht ist, unterstützt `anacron` bei der Ausführung **periodisch wiederkehrender** Aufgaben (z.B. täglich, wöchentlich, etc.).

Anzeigen der systemweiten Konfigurationsdatei `/etc/anacrontab` für `anacron`:

```
SHELL=/bin/sh
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
# the maximal random delay added to the base delay of the jobs
RANDOM_DELAY=45
# the jobs will be started during the following hours only
START_HOURS_RANGE=3-22

#period in days   delay in minutes   job-identifier   command
1       5       cron.daily              nice run-parts /etc/cron.daily
7       25      cron.weekly             nice run-parts /etc/cron.weekly
@monthly 45     cron.monthly            nice run-parts /etc/cron.monthly
```

- es werden einige Umgebungsvariablen für die auszuführenden Jobs definiert (`SHELL`, `PATH`, `MAILTO`)
- Jobs werden nur zwischen 03:00 und 22:00 gestartet (`START_HOURS_RANGE`)
- je nach Systemlast können die Jobs um maximal 45 Minuten verschoben werden (`RANDOM_DELAY`)
- es wurden drei Jobs definiert
    - `cron.daily` - führt täglich Jobs aus der Datei `/etc/cron.daily` aus
    - `cron.weekly` - führt wöchentlich Jobs aus der Datei `/etc/cron.weekly` aus
    - `cron.monthly` - führt monatlich Jobs aus der Datei `/etc/cron.monthly` aus

Anzeigen der erwähnten Jobs:

```
# ls /etc/cron.{daily,weekly,monthly}
/etc/cron.daily:

/etc/cron.monthly:

/etc/cron.weekly:
```

Derzeit sind noch keine Jobs definiert, Definitionen können aber in den Ordnern erfolgen.

Erstellen eines **systemweiten Cronjobs** in der Datei `/etc/cron.d/5min-load`:

```
# Check load every 5 minutes
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
*/5 * * * * root uptime >> /var/log/5min-load
```

`crond` erkennt die Änderung automatisch:

```
# systemctl status crond.service
...
Jan 14 12:26:01 0-node1.sva.de crond[677]: (*system*) RELOAD (/etc/cron.d/5min-
load)
```

Falls nicht, hilft ein **erneutes Einlesen** der Konfiguration:

```
# systemctl reload crond.service
```

Die Ausführung des Jobs wird im Protokoll des Dienstes sowie in der Datei `/var/log/5-min-load` vermerkt:

```
# journalctl -fu crond.service
...
Jan 14 12:35:01 0-node1.sva.de CROND[19926]: (root) CMD (uptime >>
/var/log/5min-load)
Jan 14 12:35:01 0-node1.sva.de CROND[19925]: (root) CMDEND (uptime >>
/var/log/5min-load)
```

```
 12:35:01 up  3:34,  1 user,  load average: 0.00, 0.00, 0.00
```

Zur Erstellung eines **User-Cronjobs** die Root-Sitzung beenden:

```
# exit
$ whoami
user
```

Crontab-Editor mit `crontab -e` starten und Job definieren:

```
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
*/1 * * * * echo "Aktuelle /home-Auslastung: $(du -hsx ~)" > ~/disk_usage.log
```

Beim Speichern des Jobs wird dieser direkt aktiviert:

```
no crontab for user - using an empty one
crontab: installing new crontab
```

Nach einer Minute sollte die Datei `disk_usage.log` befüllt sein:

```
$ cat disk_usage.log
Aktuelle /home-Auslastung: 28K  /home/user
```

Lab L01: Protokolle überwachen

Ziel:

- Erkunden des Ordners `/var/log` auf beiden Lernsystemen
- Erkunden des Journals
    - alle Einträge anzeigen
    - Einträge der `rsyslog.service`-Unit anzeigen
- Anzeigen der Kernel-Ausgaben

Vorgehensweise

Verbindungen zu beiden Lernsystemen herstellen und den Ordner `/var/log` erkunden.

`node1`:

```
$ ls -1 /var/log
audit
cron
dnf.librepo.log
dnf.log
dnf.rpm.log
messages
secure
spooler
...
```

`node2`:

```
$ ls -1 /var/log
apt
auth.log
dmesg
dpkg.log
faillog
journal
kern.log
lastlog
syslog
ubuntu-advantage.log
...
```

Hinweis: Einige Dateien sind auf beiden Systemen zu finden, es gibt jedoch auch viel distributionsspezifische Protokolle.

Anzeigen aller Journal-Einträge auf einem der Systeme:

```
# journalctl
```

Hinweis: Es werden viele verschiedene Protokolle angezeigt.

Zeigen des Journals der Systemd-Unit `rsyslog.service`:

```
# journalctl -u rsyslog.service
```

Anzeigen der Kernel-Ausgaben:

```
# dmesg
```

Lab L02: Remote Logging konfigurieren

Ziel:

- node2 als Remote Logging-Ziel konfigurieren
- node1 nach node2 loggen lassen

Vorgehensweise

node2

Login auf node2 und erstellen der Konfigurationsdatei `/etc/rsyslog.d/99-receive-remote.conf`:

```
# TCP syslog reception
module(load="imtcp")
input(type="imtcp" port="514")

# allow messages from local network
$AllowedSender TCP, 127.0.0.1, 192.168.56.0/24, [::1]/128, *.sva.de

# store logs to dedicated directory
$template RemoteLogs, "/var/log/servers/%FROMHOST-IP%/%PROGRAMNAME%.log"
*.* ?RemoteLogs
& ~
```

Ordner `/var/log/servers` anlegen:

```
# mkdir /var/log/servers
# chown syslog:syslog /var/log/servers
```

Überprüfen der Konfiguration und Neustarten des Dienstes:

```
# rsyslogd -f /etc/rsyslog.conf -N1
rsyslogd: version 8.2112.0, config validation run (level 1), master config
/etc/rsyslog.conf
rsyslogd: End of config validation run. Bye.

# systemctl restart rsyslog
```

Überprüfen, dass auf Port 514 gelauscht wird:

```
# ss -tulpn|grep 514
tcp   LISTEN 0      25                   0.0.0.0:514        0.0.0.0:*    users:
(("rsyslogd",pid=10742,fd=6))
```

Firewall öffnen:

```
# ufw allow 514/tcp
```

node1

Datei `/etc/rsyslog.d/99-send-remote.conf` anlegen:

```
*.* action(type="omfwd"
queue.type="LinkedList"
action.resumeRetryCount="-1"
queue.size="10000"
queue.saveonshutdown="on"
target="192.168.56.30" Port="514" Protocol="tcp")
```

Da auf dem System SELinux zum Einsatz kommt, empfiehlt es sich, den Security-Kontext der Konfigurationsdatei falls benötigt anzupassen:

```
# restorecon -v /etc/rsyslog.d/99-send-remote.conf
```

Überprüfen der Konfiguration und Neustarten des Dienstes:

```
# rsyslogd -f /etc/rsyslog.conf -N1
rsyslogd: version 8.2310.0-4.el9, config validation run (level 1), master
config /etc/rsyslog.conf
rsyslogd: End of config validation run. Bye.

# systemctl restart rsyslog
```

Schreiben einer Testnachricht:

```
# logger -p user.info "Test message"
```

Diese sollte auf node2 in der Datei `/var/log/servers/192.168.56.20/root.log` zu sehen sein:

```
Feb 14 14:42:57 0-node1 root[28607]: Test message
```

Lab ST01: LVM konfigurieren

Ziel:

- Login auf node1
- Auf der Festplatte `/dev/sdb` (oder `/dev/sdb`) ein Physical Volume (**PV**) anlegen
- Die Volume Group (**VG**) `vg_training` anlegen
- Zwei Logical Volumes (**LV**) in der Volume Group anlegen
    - `lv_data1`, 2 GB

    - lv_data2, 2 GB
- Größen der Logical Volumes auf jeweils **4 GB** ändern

Vorgehensweise

Anlegen des **PVs**:

```
# pvcreate /dev/sdb
Physical volume "/dev/sdb" successfully created.
```

Vorhandene PVs **auflisten** und Details zum neu angelegten PV **anzeigen**:

```
# pvs
  PV         VG Fmt  Attr PSize  PFree
  /dev/sdb      lvm2 ---  10.00g 10.00g
```

```
# pvdisplay /dev/sdb
  "/dev/sdb" is a new physical volume of "10.00 GiB"
  --- NEW Physical volume ---
  PV Name               /dev/sdb
  VG Name
  PV Size               10.00 GiB
  Allocatable           NO
  PE Size               0
  Total PE              0
  Free PE               0
  Allocated PE          0
  PV UUID               Yw65R-jOM7-Crr4-URmd-z6ay-JhaN-j3g6eU
```

Die Ausgabe zeigt, dass das Volume 10 GB groß (`PV Size`) ist. Es wurde eine eindeutige Identifikationsnummer generiert (`PV UUID`). Das PV ist noch leer (`PE Size`, `Total PE`, `Free PE`, `Allocated PE`). Es kann noch nicht benutzt werden (`Allocatable`), da es noch keine VG angelegt wurde.

Anlegen der **VG**:

```
# vgcreate vg_training /dev/sdb
  Volume group "vg_training" successfully created
```

Vorhandene VGs **auflisten** und Details zur neu angelegten VG **anzeigen**:

```
# vgs
  VG          #PV #LV #SN Attr   VSize   VFree
  vg_training   1   0   0 wz--n- <10.00g <10.00g
```

```
# vgdisplay vg_display
  --- Volume group ---
  VG Name               vg_training
  System ID
  Format                lvm2
  Metadata Areas        1
  Metadata Sequence No  1
  VG Access             read/write
  VG Status             resizable
  MAX LV                0
  Cur LV                0
  Open LV               0
  Max PV                0
  Cur PV                1
  Act PV                1
  VG Size               <10.00 GiB
```

Die VG hat noch **keine LVs** (`Cur LV`), verwendet aber **ein PV** (`Cur PV`, `Act PV`) und ist **10 GB** groß (`VG Size`).

Anlegen von der beiden **LVs**:

```
# lvcreate --name lv_data1 --size 2G vg_training
  Logical volume "lv_data1" created.

# lvcreate --name lv_data2 --size 2G vg_training
  Logical volume "lv_data2" created.
```

Vorhandene LVs **auflisten** und Details zu `lv_data1` in der VG `vg_training` **anzeigen**:

```
# lvs
  LV       VG          Attr       LSize Pool Origin Data%  Meta%  Move Log
Cpy%Sync Convert
  lv_data1 vg_training -wi-a----- 2.00g
  lv_data2 vg_training -wi-a----- 2.00g

```command
# lvdisplay vg_training/lv_data1
  --- Logical volume ---
  LV Path                /dev/vg_training/lv_data1
  LV Name                lv_data1
  VG Name                vg_training
  LV UUID                DQIG8w-tTvm-I5eC-T95h-bxAc-yHh5-AYL7oz
  LV Write Access        read/write
  LV Creation host, time 0-node1.sva.de, 2025-01-10 14:47:41 +0000
  LV Status              available
  # open                 0
  LV Size                2.00 GiB
  Current LE             512
  Segments               1
  Allocation             inherit
  Read ahead sectors     auto
  - currently set to     256
  Block device           253:0
```

Das LV hat ebenfalls eine eindeutige Kennzeichnung (`LV UUID`) und ist **2 GB** groß (`LV Size`).

Lab ST02: Dateisysteme anlegen

Diese Aufgabe erfordert, dass Lab ST01 bereits abgeschlossen wurde.

Ziel:

- Login auf node1
- Zwei Dateisysteme anlegen:
    - `/dev/mapper/vg_training-lv_data1`, **ext4**
    - `/dev/mapper/vg_training-lv_data2`, **XFS**
- Dateisysteme manuell einhängen
    - `/dev/mapper/vg_training-lv_data1` unter `/data1`
    - `/dev/mapper/vg_training-lv_data2` unter `/data2`
- jeweils eine **512 MB** große Datei anlegen
    - `dd if=/dev/random of=<Pfad>/file.img bs=1024k count=1024`

Vorgehensweise

Anlegen der beiden Dateisysteme:
```

```
# mkfs.ext4 /dev/mapper/vg_training-lv_data1
mke2fs 1.46.5 (30-Dec-2021)
Discarding device blocks: done
Creating filesystem with 524288 4k blocks and 131072 inodes
Filesystem UUID: 2ce68006-3029-4722-ba9c-41b6f1c77d75
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912

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

```
# mkfs.xfs /dev/mapper/vg_training-lv_data2
meta-data=/dev/mapper/vg_training-lv_data2 isize=512    agcount=4,
agsize=131072 blks
         =                       sectsz=512   attr=2, projid32bit=1
         =                       crc=1        finobt=1, sparse=1, rmapbt=0
         =                       reflink=1    bigtime=1 inobtcount=1 nrext64=0
data     =                       bsize=4096   blocks=524288, imaxpct=25
         =                       sunit=0      swidth=0 blks
naming   =version 2              bsize=4096   ascii-ci=0, ftype=1
log      =internal log           bsize=4096   blocks=16384, version=2
         =                       sectsz=512   sunit=0 blks, lazy-count=1
realtime =none                   extsz=4096   blocks=0, rtextents=0
Discarding blocks...Done.
```

Mountpoints anlegen und Dateisysteme **einhängen**:

```
# mkdir /data1 /data2
# mount /dev/mapper/vg_training-lv_data1 /data1
# mount /dev/mapper/vg_training-lv_data2 /data2
```

Zwei Dateien mit **512 MB**-Größe anlegen:

```
# dd if=/dev/random of=/data1/file.img bs=1024k count=512
# dd if=/dev/random of=/data2/file.img bs=1024k count=512
```

Lab ST03: LVs vergrößern und verkleinern

Diese Aufgabe erfordert, dass die Labs ST01 und ST02 bereits abgeschlossen wurden.

Ziel:

- Login auf node1
- LV-Größen ändern
    - `vg_training/lv_data2` von 2 GB auf 3 GB
    - `vg_training/lv_data1` von 2 GB auf 1 GB
- Dateisysteme vergrößern bzw. verkleinern

Vorgehensweise

In der Regel können alle Dateisysteme **online vergrößert** werden - d.h. ohne Aushängen. Beim Vergrößern muss erst das LV erweitert werden, bevor das Dateisystem vergrößert wird:

```
# lvresize --size +1G vg_training/lv_data2
# xfs_growfs /dev/mapper/vg_training-lv_data2
```

Die beiden Schritte lassen sich mit dem `--resize`-Parameter auch in einem Kommando kombinieren:

```
# lvresize --size +1G --resize <Pfad>
```

Nicht alle Dateisysteme können verkleinert werden, XFS unterstützt dies beispielsweise nicht. ext4 kann **offline verkleinert** werden. Hier muss das Dateisystem ausgehängt, geprüft und verkleinert werden, bevor das LV verkleinert wird:

```
# umount /data1
# fsck.ext4 -f /dev/mapper/vg_training-lv_data1

# resize2fs /dev/mapper/vg_training-lv_data1 1G
resize2fs 1.46.5 (30-Dec-2021)
Resizing the filesystem on /dev/mapper/vg_training-lv_data1 to 262144 (4k)
blocks.
The filesystem on /dev/mapper/vg_training-lv_data1 is now 262144 (4k) blocks
long.

# lvresize --size -1G vg_training/lv_data1
  File system ext4 found on vg_training/lv_data1.
  File system size (1.00 GiB) is equal to the requested size (1.00 GiB).
  File system reduce is not needed, skipping.
  Size of logical volume vg_training/lv_data1 changed from 2.00 GiB (512
extents) to 1.00 GiB (256 extents).
  Logical volume vg_training/lv_data1 successfully resized.
```

Dieser Vorgang kann **nicht** mit dem `--resize`-Parameter des `lvresize`-Kommandos vereinfacht werden.

Das Dateisystem kann nun wieder eingehängt werden:

```
# mount /dev/mapper/vg_training-lv_data1 /data1
```

Anschließend sollten sich die Dateisystemgrößen geändert haben:

```
# lvs
  LV       VG          Attr       LSize Pool Origin Data%  Meta%  Move Log
Cpy%Sync Convert
  lv_data1 vg_training -wi-ao---- 1.00g
  lv_data2 vg_training -wi-ao---- 3.00g

# df -h|grep /data
/dev/mapper/vg_training-lv_data2  3.0G  566M  2.4G  19% /data2
/dev/mapper/vg_training-lv_data1  939M  513M  360M  59% /data1
```

Lab ST04: LVM-Snapshots benutzen

Diese Aufgabe erfordert, dass die Labs ST01 bis ST03 bereits abgeschlossen wurden.

Ziel:

- Login auf node1
- Snapshot für LV `vg_training/lv_data2` anlegen
    - Name: `lv_data2_snap`
    - Größe: 1 GB
- Löschen der Datei `/data2/file.img`
- Wiederherstellen der Datei aus dem Snapshot
    - Einhängen des Snapshots
    - Kopieren der Datei

Vorgehensweise

Erstellen des Snapshots:

```
# lvcreate --size 1G --snapshot --name lv_data2_snap vg_training/lv_data2
  Logical volume "lv_data2_snap" created.

# lvs
  LV            VG          Attr       LSize Pool Origin   Data%  Meta%  Move
Log Cpy%Sync Convert
  lv_data1      vg_training -wi-ao---- 1.00g
  lv_data2      vg_training owi-aos--- 3.00g
  lv_data2_snap vg_training swi-a-s--- 1.00g      lv_data2 0.00
```

Löschen der Datei `/data2/file.img`:

```
$ rm /data2/file.img
```

Überprüfen, dass die Datei noch im Snapshot existiert:

```
# mkdir /data2_snapshot
# mount /dev/mapper/vg_training-lv_data2_snap /data2_snapshot
# ls /data2_snapshot
file.img
```

Aushängen beider Dateisysteme und Wiederherstellen des Snapshots:

```
# umount /data2_snapshot /data2
# lvconvert --merge /dev/mapper/vg_training-lv_data2_snap
```

Hinweis: Möglicherweise erscheint hierbei der folgende Fehler:

```
Command on LV vg_training/lv_data2_snap is invalid on LV with properties:
lv_is_merging_cow .
Command not permitted on LV vg_training/lv_data2_snap.
```

Hierbei hilft es das folgende Kommando auszuführen und den Vorgang zu wiederholen:

```
# lvchange --refresh vg_data
# lvconvert --merge /dev/mapper/vg_training-lv_data2_snap
```

Abschließend kann das ursprüngliche Dateisystem wieder eingehängt werden:

```
# mount dev/mapper/vg_training-lv_data2 /data2
# ls /data2
```

Hinweis: Die Datei ist wieder vorhanden.

Lab ST05: VG erweitern

VGs **auflisten**

```
# vgs
  VG          #PV #LV #SN Attr   VSize   VFree
  vg_training   1 1  0   0 wz--n- <10.00g <5.00g
```

Anlegen des neuen **PVs**:

```
# pvcreate /dev/sdc
Physical volume "/dev/sdc" successfully created.
```

Vorhandene PVs **auflisten** und Details zum neu angelegten PV **anzeigen**:

```
# pvs
  PV         VG          Fmt  Attr PSize   PFree
  /dev/sdb   vg_training lvm2 a--  <10.00g <5.00g
  /dev/sdc               lvm2 ---   10.00g 10.00g
```

```
# pvdisplay /dev/sdc
  "/dev/sdc" is a new physical volume of "10.00 GiB"
  --- NEW Physical volume ---
  PV Name               /dev/sdc
  VG Name
  PV Size               10.00 GiB
  Allocatable           NO
  PE Size               0
  Total PE              0
  Free PE               0
  Allocated PE          0
  PV UUID               h70ee0-dRRy-KwB0-HTWA-AGG1-pdHC-Pdnixn
```

Die Ausgabe zeigt, dass das Volume 10 GB groß (`PV Size`) ist. Es wurde eine eindeutige Identifikationsnummer generiert (`PV UUID`). Das PV ist noch leer (`PE Size`, `Total PE`, `Free PE`, `Allocated PE`). Es kann noch nicht benutzt werden (`Allocatable`), da es noch keine VG angelegt wurde.

VG um das neue PV erweitern:

```
# vgextend vg_training /dev/sdc
  Volume group "vg_training" successfully extended
```

Die VG umfasst nun **mehr Speicher**:

```
# vgs
  VG          #PV #LV #SN Attr   VSize  VFree
  vg_training   2   3   1 wz--n- 19.99g 14.99g
```

Es wurden noch keine Datenblöcke auf der neuen Disk hinterlegt:

```
# pvdisplay /dev/sdc
...
  Allocated PE          0
```

Lab ST06: LUKS konfigurieren

Diese Aufgabe erfordert, dass das Lab ST01 bereits abgeschlossen wurde.

Ziel:

- Login auf node1
- In der **VG** vg_training ein neues **LV** anlegen
    - Name: lv_luks
    - Größe: 1 GB
- Volume mit cryptsetup verschlüsseln
    - Passphrase LUKS1337
- LUKS-Volume einhängen und mit Dateisystem versehen
- Eine weitere Passphrase (L1nux1337) erstellen

Vorgehensweise

Neues **LV** anlegen:

```
# lvcreate --name lv_luks --size 1G vg_training
  Logical volume "lv_luks" created.
```

Volume mit cryptsetup verschlüsseln:

```
# cryptsetup luksFormat -y /dev/mapper/vg_training-lv_luks

WARNING!
========
This will overwrite data on /dev/mapper/vg_training-lv_luks irrevocably.

Are you sure? (Type 'yes' in capital letters): YES
Enter passphrase for /dev/mapper/vg_training-lv_luks: LUKS1337
Verify passphrase: LUKS1337
```

Öffnen des LUKS-Volumes unter Angabe der Passphrase:

```
# cryptsetup open /dev/mapper/vg_training-lv_luks secret_data
Enter passphrase for /dev/mapper/vg_training-lv_luks:
```

Details zum benutzten LUKS-Volume anzeigen:

```
# cryptsetup status /dev/mapper/secret_data
/dev/mapper/secret_data is active and is in use.
  type:    LUKS2
  cipher:  aes-xts-plain64
  keysize: 512 bits
  key location: keyring
  device:  /dev/mapper/vg_training-lv_luks
  sector size:  512
  offset:  32768 sectors
  size:    2064384 sectors
  mode:    read/write
```

Anlegen und Einhängen eines ext4-Dateisystems:

```
# mkfs.ext4 /dev/mapper/secret_data
# mkdir /secret_data
# mount /dev/mapper/secret_data /secret_data
```

Anlegen einer Datei sowie **Aushängen** des Dateisystems:

```
# echo "DM ist teurer als Rossmann" > /secret_data/secret.txt
# umount /secret_data
```

Hinzufügen einer weiteren Passphrase:

```
# cryptsetup luksAddKey /dev/mapper/vg_training-lv_luks
Enter any existing passphrase: LUKS1337
Enter new passphrase for key slot: L1nux1337
Verify passphrase: L1nux1337
```

Anzeigen von Header-Informationen des LUKS-Volumes:

```
# cryptsetup luksDump /dev/mapper/vg_training-lv_luks
LUKS header information
Version:        2
Epoch:          4
Metadata area:  16384 [bytes]
Keyslots area:  16744448 [bytes]
UUID:           ea15183b-ec73-4b75-9947-7648ad112c29
Label:          (no label)
Subsystem:      (no subsystem)
Flags:          (no flags)

Data segments:
  0: crypt
        offset: 16777216 [bytes]
        length: (whole device)
        cipher: aes-xts-plain64
        sector: 512 [bytes]

Keyslots:
  0: luks2
        Key:        512 bits
        ...
  1: luks2
        Key:        512 bits
        ...
...
```

Das LUKS-Volume entspricht der **Version 2** (`Version`) und umfasst die gesamte Festplatte (`length`). Es wurden zwei Passphrases (`Keyslots`) definiert (`0`, `1`).

Lab ST07: Software-RAID konfigurieren

Ziel:

- Login auf der node2
- Die Festplatten `/dev/sdb` (oder `/dev/sdb`) und `/dev/sdc` (oder `/dev/sdc`) **partitionieren**
    - Eine Partition über die gesamte Festplatte
    - RAID-Flag setzen
- **RAID-0**-Verbund erstellen und überprüfen
- Dateisystem erstellen und einhängen
    - ext4
    - `/raidvol`

Vorgehensweise

Erste Festplatte **partitionieren**:

```
# parted /dev/sdb mklabel gpt
# parted -a optimal -- /dev/sdb mkpart primary 0% 100%
# parted /dev/sdb set 1 raid on
```

Wiederholen der Schritte für die zweite Festplatte:

```
# parted /dev/sdc mklabel gpt
# parted -a optimal -- /dev/sdc mkpart primary 0% 100%
# parted /dev/sdc set 1 raid on
```

RAID 0-Volume erstellen:

```
# mdadm --create /dev/md0 --level=0 --raid-devices=2 /dev/sdb1 /dev/sdc1
mdadm: Defaulting to version 1.2 metadata
mdadm: array /dev/md0 started.
```

Volume **überprüfen**:

```
# cat /proc/mdstat
Personalities : [linear] [multipath] [raid0] [raid1] [raid6] [raid5] [raid4]
[raid10]
md0 : active raid0 sdc1[1] sdb1[0]
      20948992 blocks super 1.2 512k chunks

unused devices: <none>
```

Es wurde ein RAID 0-Volume `md0` mit zwei Festplatten (`sdb1`, `sdc1`) angelegt. Es gibt keine ungenutzten Festplatten.

Erstellen eines Dateisystems auf dem neuen Gerät `/dev/md0`:

```
# mkfs.ext4 /dev/md0
```

Einhängen des Dateisystems:

```
# mkdir /raidvol
# mount /dev/md0 /raidvol

# df -h | grep raidvol
/dev/md0                              20G   24K   19G   1% /raidvol
```

Lab ST08: Dateisysteme automatisch einhängen

Diese Aufgabe erfordert, dass die Labs ST01 und ST02 bereits abgeschlossen wurden.

Ziel:

- Login auf node1
- Automatisches Einhängen zweier Dateisysteme
    - `/dev/mapper/vg_training-lv_data1` unter `/data1` via `/etc/fstab`
    - `/dev/mapper/vg_training-lv_data2` unter `/data2` via systemd

Vorgehensweise

UUIDs der Dateisysteme auslesen:

```
# blkid /dev/mapper/vg_training-lv_data{1,2}
/dev/mapper/vg_training-lv_data1: UUID="2ce68006-3029-4722-ba9c-41b6f1c77d75"
TYPE="ext4"
/dev/mapper/vg_training-lv_data2: UUID="e7788fc7-c951-4781-8433-c9e48ca6a8d7"
TYPE="xfs"
```

Hinzufügen eines Eintrags in die Datei `/etc/fstab`:

```
UUID=2ce68006-3029-4722-ba9c-41b6f1c77d75 /data1  ext4  defaults  0 0
```

Einhängen aller automatisch einzuhängenden Speicher:

```
# mount -a
mount: (hint) your fstab has been modified, but systemd still uses
       the old version; use 'systemctl daemon-reload' to reload.
```

Moderne Linux-Distributionen übersetzen `/etc/fstab`-Einträge automatisch in **systemd** `.mount`-Units und speichern diese im `/run/systemd/generator`-Ordner:

```
# systemctl daemon-reload
# systemctl status data1.mount
● data1.mount - /data1
     Loaded: loaded (/etc/fstab; generated)
     Active: active (mounted) since Mon 2025-01-13 14:30:52 UTC; 1min 44s ago
      Until: Mon 2025-01-13 14:30:52 UTC; 1min 44s ago
      Where: /data1
       What: /dev/mapper/vg_training-lv_data1
       Docs: man:fstab(5)
             man:systemd-fstab-generator(8)

# ls -1 /run/systemd/generator/*.mount
/run/systemd/generator/-.mount
/run/systemd/generator/boot-efi.mount
/run/systemd/generator/boot.mount
/run/systemd/generator/data1.mount
```

Erstellen einer `.mount`-Unit unter `/etc/systemd/system/data2.mount`:

```
[Unit]
Before=local-fs.target

[Mount]
What=/dev/disk/by-uuid/e7788fc7-c951-4781-8433-c9e48ca6a8d7
Where=/data2
Type=xfs
```

systemd über Änderungen informieren und Dateisystem einhängen:

```
# systemctl daemon-reload
# systemctl start data2.mount

# df -h | grep data2
/dev/mapper/vg_training-lv_data2  3.0G  566M  2.4G  19% /data2
```

Die Konfiguration über systemd empfiehlt sich dann, wenn Dienste dynamisch Speicher einhängen sollen.

Lab M01: Prozesse kontrollieren

Ziel:

- **zwei** SSH-Verbindungen zu einem der Lernsysteme herstellen
    - in einer das Programm `/opt/train.sh` starten
- in der anderen Shell Prozessinformationen anzeigen
    - eigene Prozesse
    - alle Prozesse
    - Prozesse mit dem Namen `train.sh` sowie erweiterten Informationen
- Beenden des Programs
    - sauberes Beenden, falls nötig rabiates Beenden

Vorgehensweise

Zwei Verbindungen zu einem der Lernsysteme herstellen.

In einer Sitzung das Programm `/opt/train.sh` starten:

```
$ /opt/train.sh
```

In anderer Sitzung die eigenen Prozesse anzeigen:

```
$ ps -U user
    PID TTY          TIME CMD
    719 ?        00:00:00 systemd
   3153 ?        00:00:00 sshd
   3154 pts/0    00:00:00 bash
  13083 ?        00:00:00 sshd
  13084 pts/1    00:00:00 bash
  13112 pts/0    00:00:00 watch
  13263 pts/1    00:00:00 train.sh
  13288 pts/1    00:00:00 sl
  13289 pts/0    00:00:00 ps
...
```

Hinweis: Es werden verschiedene `bash`-, `sshd`-, `watch`- und `sl`-Prozesse angezeigt. Nummern und Vorkommen können variieren.

Anzeigen aller Prozesse:

```
$ ps -ef
UID          PID    PPID  C STIME TTY          TIME CMD
root           1       0  0 14:03 ?        00:00:01 /usr/lib/systemd/systemd --
system --deserialize 18
root           2       0  0 14:03 ?        00:00:00 [kthreadd]
root           3       2  0 14:03 ?        00:00:00 [rcu_gp]
...
```

Hinweis: Es werden signifikant mehr Prozesse angezeigt.

Anzeigen aller Prozesse mit erweiterten Informationen mit dem Namen `train.sh`:

```
$ ps -fC train.sh
UID        PID    PPID  C STIME TTY          TIME CMD
user     13263   13084  0 18:30 pts/1    00:00:00 /bin/sh /opt/train.sh
```

Hinweis: IDs und Zeiten können abweichen.

Versuchtes Beenden des Programms:

```
$ kill 13263
$ kill -2 13263
```

Hinweis: Das Skript scheint die Signale `SIGTERM` (*15*) und `SIGINT` (*2*) zu ignorieren.

Rabiates Beenden des Programms:

```
$ kill -9 13263
```

Das Programm ist nun gestoppt.

Betrachten aller Prozesse mit `top` und `htop`:

```
$ top
$ htop
```

Lab M02: Troubleshooting einer Anwendung

Ziel:

- Login auf node1
    - Dienst `crapp.service` starten
    - System auf Auffälligkeiten untersuchen
        - Ursachen analysieren
    - Dienst stoppen

Vorgehensweise

Starten des Dienstes:

```
# systemctl start crapp.service
```

Systemlast anzeigen:

```
# htop
```

Es scheint, als wäre eine CPU belastet, während die andere nicht belastet ist.

Detailliere Auslastung pro CPU anzeigen:

```
$ mpstat -P ALL 1 30
14:12:39     CPU    %usr   %nice    %sys %iowait    %irq   %soft  %steal
%guest  %gnice   %idle
14:12:40     all   50.25    0.00    0.00    0.00    0.00    0.00    0.00
0.00    0.00   49.75
14:12:40       0  100.00    0.00    0.00    0.00    0.00    0.00    0.00
0.00    0.00    0.00
14:12:40       1    0.00    0.00    0.00    0.00    0.00    0.00    0.00
0.00    0.00  100.00

14:12:40     CPU    %usr   %nice    %sys %iowait    %irq   %soft  %steal
%guest  %gnice   %idle
14:12:41     all   50.00    0.00    0.00    0.00    0.00    0.00    0.00
0.00    0.00   50.00
14:12:41       0  100.00    0.00    0.00    0.00    0.00    0.00    0.00
0.00    0.00    0.00
14:12:41       1    0.00    0.00    0.00    0.00    0.00    0.00    0.00
0.00    0.00  100.00
...
Average:     CPU    %usr   %nice    %sys %iowait    %irq   %soft  %steal
%guest  %gnice   %idle
Average:     all   50.00    0.00    0.00    0.00    0.00    0.00    0.00
0.00    0.00   50.00
Average:       0  100.00    0.00    0.00    0.00    0.00    0.00    0.00
0.00    0.00    0.00
Average:       1    0.00    0.00    0.00    0.00    0.00    0.00    0.00
0.00    0.00  100.00
```

Hinweis: Es ist wirklich nur eine der beiden CPUs ausgelastet.

Laut `htop` ist der Prozess `stress-ng` dafür verantwortlich. Von welchem Prozess wurde dieser gestartet?

```
# ps -fC stress-ng
UID          PID    PPID  C STIME TTY          TIME CMD
root       21927   21926  0 14:10 ?        00:00:00 stress-ng --matrix 1
```

Hinweis: Der Prozess wurde vom Prozess **21926** gestartet (`PPID`) - welches Programm verbirgt sich dahinter?

```
# ps -fP 21926
UID          PID    PPID PSR  C STIME TTY      STAT   TIME CMD
root       21926       1   1  0 14:10 ?        Ss     0:00 /bin/sh
/usr/local/bin/crapp.sh
```

Das Skript `/usr/local/bin/crapp.sh` wurde von Systemd (PID 1) gestartet. Es muss sich also um einen Dienst handeln.

Überprüfen, ob der soeben gestartete Dienst das Skript startet:

```
# systemctl cat crapp.service
```

Stoppen des Dienstes:

```
# systemctl stop crapp.service
```

Lab M03: Netzwerkverkehr beobachten

Ziel:

- Login auf node1
    - Webserver starten
    - Lauschende Sockets überprüfen
    - Netzstatistiken überwachen
- Login auf node2
    - Zugriff auf Webserver
    - ICMP-Erreichbarkeit überprüfen

Vorgehensweise

node1

Login auf node1 und starten des Webservers:

```
# systemctl start httpd
# curl http://localhost
```

Überprüfen lauschender Sockets:

```
# ss -tulpen
Netid State  Recv-Q Send-Q Local Address:Port  Peer Address:PortProcess
...
tcp   LISTEN 0      511                *:80               *:*    users:
(("httpd",pid=794,fd=4),("httpd",pid=793,fd=4),("httpd",pid=792,fd=4),
("httpd",pid=717,fd=4)) ino:23653 sk:13 cgroup:/system.slice/httpd.service
v6only:0 <->
```

Netzstatistiken überwachen:

```
# iptraf-ng
```

- IP traffic monitor
    - Select Interface
        - eth1

node2

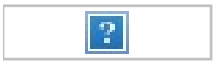
Login auf node2 und Zugreifen auf den Webserver:

```
$ curl http://node1
```

ICMP-Erreichbarkeit überprüfen:

```
$ ping node1
```

Hinweis: In `iptraf-ng` sollten nun erfolgte HTTP-Anfragen und ICMP-Echos zu sehen sein:



Lab S01: Apache installieren

Ziel:

- Login auf node1
    - Paket `httpd` installieren
    - Dienst `httpd` starten und für automatischen Start konfigurieren
    - Eigene Startseite unter `/var/www/html/index.html` definieren
    - Funktionsfähigkeit überprüfen
- Wiederholung auf node2
    - Paket und Dienst heißen hier apache2

Vorgehensweise

node1

Anmeldung auf node1 sowie Installieren des Pakets:

```
# dnf install httpd
```

Dienst starten und für automatischen Start konfigurieren:

```
# systemctl enable --now httpd
```

Startseite als `/var/www/html/index.html` anlegen:

```
Apache2-Testseite
Lorem ipsum doloret.
```

Funktionsfähigkeit überprüfen:

```
$ curl http://localhost
Apache2-Testseite
Lorem ipsum doloret.
```

Hinweis: Der Test kann auch unter Angabe der öffentlichen IP-Adresse mit einem Web-Browser vorgenommen werden.

node2

Anmeldung auf node2 sowie Installieren des Pakets:

```
# apt-get install apache2
```

Hinweis: Der Dienst wird automatisch nach der Installation gestartet und für den automatischen Start konfiguriert:

```
Created symlink /etc/systemd/system/multi-user.target.wants/apache2.service →
/lib/systemd/system/apache2.service.
Created symlink /etc/systemd/system/multi-user.target.wants/apache-
htcacheclean.service → /lib/systemd/system/apache-htcacheclean.service.
```

Vorherige Startseite sichern:

```
# mv /var/www/html/index.html /var/www/html/index.html.orig
```

Startseite als `/var/www/html/index.html` anlegen:

```
Apache2-Testseite
Lorem ipsum doloret.
```

Funktionsfähigkeit überprüfen:

```
$ curl http://localhost
Apache2-Testseite
Lorem ipsum doloret.
```

Hinweis: Der Test kann auch unter Angabe der öffentlichen IP-Adresse mit einem Web-Browser vorgenommen werden.

Lab S02: PHP-Anwendung installieren

Ziel:

- Login auf node1
    - PHP installieren
    - Setzen einer Apache-Konfigurationsdirektive
        - `AllowOverride All`
    - DokuWiki (https://dokuwiki.org) herunterladen und entpacken
    - Dateisystem-Berechtigungen gemäß Dokumentation (https://www.dokuwiki.org/install:centos) setzen
    - Anwendung installieren

Vorgehensweise

PHP installieren:

```
# dnf install php php-gd
```

Editieren der Datei `/etc/httpd/conf/httpd.conf:

```
<Directory "/var/www/html">
    ...
    AllowOverride All
    ...
    Require all granted
</Directory>
```

Hinweis Der Wert `AllowOverride` muss von `None` auf `All` gesetzt werden.

Neustarten des Webservers:

```
# systemctl restart httpd
```

DokuWiki herunterladen und nach `/var/www/html/` entpacken:

```
# curl -LO https://download.dokuwiki.org/src/dokuwiki/dokuwiki-stable.tgz
# tar xvfz dokuwiki-stable.tgz -C /var/www/html/
```

Entpackten Ordner in `dokuwiki` umbenennen:

```
# cd /var/www/html
# ls
dokuwiki-2024-02-06b/ index.html
# mv dokuwiki-2024-02-06b dokuwiki
```

Dateisystem-Berechtigungen [gemäß Dokumentation (https://www.dokuwiki.org/install:centos)](https://www.dokuwiki.org/install:centos) setzen:

```
# chown -R apache:apache /var/www/html/dokuwiki/data
# chown apache:apache /var/www/html/dokuwiki/conf
# semanage fcontext -a -t httpd_sys_rw_content_t
"/var/www/html/dokuwiki/conf(/.*)?"
# semanage fcontext -a -t httpd_sys_rw_content_t
"/var/www/html/dokuwiki/data(/.*)?"
# restorecon -R /var/www/html
```

Web-Browser öffnen und `/dokuwiki/install.php` aufrufen, um die Installation abzuschließen:

- Wählen Sie Ihre Sprache: de
- Wiki-Name: `Linux-Wiki`
- Benutzername des Administrators: `sgiertz`
- Voller Name: `Simone Giertz`
- E-Mail: `simone@gier.tz`
- Anfangseinstellungen der Zugangskontrolle (ACL): `Öffentliches Wiki (Lesen für alle, Schreiben und Hochladen nur für registrierte Benutzer)`
- Benutzer dürfen sich registrieren: **deaktivieren**
- Einmal monatlich anonymisierte Nutzungsdaten an das DokuWiki-Entwicklerteam senden: **deaktivieren**
- **Speichern** und **neues DokuWiki** klicken

Lab S03: MariaDB installieren

Ziel:

- Login auf node1
    - Paket `mariadb-server` installieren
    - Datenbank via `mariadb-secure-installation` härten
- Wiederholung auf node2

Vorgehensweise

node1

MariaDB installieren und starten:

```
# dnf install mariadb-server
# systemctl enable --now mariadb-server
```

Datenbank härten:

```
# mariadb-secure-installation

Enter current password for root (enter for none):

Switch to unix_socket authentication [Y/n] Y
Enabled successfully!
Reloading privilege tables..
 ... Success!

Change the root password? [Y/n] Y
New password:
Re-enter new password:
Password updated successfully!
Reloading privilege tables..
 ... Success!

Remove anonymous users? [Y/n] Y
 ... Success!

Disallow root login remotely? [Y/n] Y
 ... Success!

Remove test database and access to it? [Y/n] Y
 - Dropping test database...
 ... Success!
 - Removing privileges on test database...
 ... Success!

Reload privilege tables now? [Y/n] Y
 ... Success!
```

node2

MariaDB installieren:

```
# apt-get install -y mariadb-server
```

Datenbank härten:

```
# mariadb-secure-installation

Enter current password for root (enter for none):

Switch to unix_socket authentication [Y/n] Y
Enabled successfully!
Reloading privilege tables..
 ... Success!

Change the root password? [Y/n] Y
New password:
Re-enter new password:
Password updated successfully!
Reloading privilege tables..
 ... Success!

Remove anonymous users? [Y/n] Y
 ... Success!

Disallow root login remotely? [Y/n] Y
 ... Success!

Remove test database and access to it? [Y/n] Y
 - Dropping test database...
 ... Success!
 - Removing privileges on test database...
 ... Success!

Reload privilege tables now? [Y/n] Y
 ... Success!
```

Lab S04: MariaDB-Inhalte verwalten

Ziel:

- Login auf node1 oder node2
- User und Datenbank erstellen
    - User: `training`
    - Datenbank: `training`
    - Passwort: `geffjeerling`
- Tabelle `trainings` anlegen
    - Felder
        - `training_id`, `int`, `auto_increment`, **Primärschlüssel**

        - `training_name`, `tinytext`, `not null`,
        - `training_year`, `int`, `not null`

- Eintrag vornehmen
    - Training 'Linux Advanced' von 2025
- Backup der Datenbank anlegen
    - Tabelle löschen und wiederherstellen

Vorgehensweise

Verbindung zum MariaDB-Server herstellen, Benutzer und Datenbank anlegen und Berechtigungen vergeben:

```
# mysql -u root -p

MariaDB> CREATE USER 'training'@'localhost' IDENTIFIED BY 'geffjeerling';
MariaDB> GRANT ALL PRIVILEGES ON training.* TO 'training'@localhost;
MariaDB> CREATE DATABASE training;
MariaDB> FLUSH PRIVILEGES;
MariaDB> exit;
```

Mit neuem User und neuer Datenbank verbinden und die Tabelle anlegen:

```
# mysql -u training -p training
MariaDB> CREATE TABLE trainings(training_id int auto_increment, training_name
tinytext not null, training_year int not null, PRIMARY KEY(training_id));

MariaDB> INSERT INTO trainings (training_name, training_year) VALUES ('Linux
Advanced', 2025);

MariaDB> exit;
```

Backup der Datenbank anlegen:

```
# mariadb-dump training > training_backup.sql
```

Tabelle löschen und wiederherstellen:

```
# mysql -u training -p training
MariaDB> DROP TABLE trainings;
MariaDB> exit;

# mariadb -u training -p training < training_backup.sql
```

Überprüfen, dass die Daten wieder vorhanden sind:

```
# mariadb -u training -p training
MariaDB> SELECT * FROM trainings;
+-------------+---------------+---------------+
| training_id | training_name  | training_year |
+-------------+---------------+---------------+
|           1 | Linux Advanced |          2025 |
+-------------+---------------+---------------+
1 row in set (0.001 sec)
```

Lab S05: Samba-Server einrichten

Ziel:

- Login auf `node1`
- Installation des `samba`-Pakets
- Starten und automatisches Aktivieren des Dienstes
- Überprüfen der Konfiguration
- Anlegen eines Samba-Passworts für `user`
    - Passwort analog zum Linux-User
- Erstellen einer Freigabe
    - Name: `labs`
    - Pfad: `/labs`
    - Nur-lesen: `ja`
    - Durchsuchbar: `ja`

Vorgehensweise

Installation des Pakets sowie Aktivieren des Dienstes:

```
# dnf install samba
# systemctl enable --now smbd.service
```

Überprüfen der Konfiguration:

```
# testparm
Load smb config files from /etc/samba/smb.conf
Loaded services file OK.

Server role: ROLE_STANDALONE

Press enter to see a dump of your service definitions

# Global parameters
[global]
        printcap name = cups
        security = USER
        workgroup = SAMBA
        idmap config * : backend = tdb
        cups options = raw


[homes]
        browseable = No
        comment = Home Directories
        inherit acls = Yes
        read only = No
        valid users = %S %D%w%S


[printers]
        browseable = No
        comment = All Printers
        create mask = 0600
        path = /var/tmp
        printable = Yes
```

Anlegen eines Samba-Passworts für user:

```
# smbpasswd -a user
New SMB password:
Retype new SMB password:
Added user user.
```

Erstellen einer Freigabe durch Hinzufügen der folgenden Zeilen in die Datei /etc/samba/smb.conf:

```
[labs]
        comment = Lab Solutions
        path = /labs
        read only = Yes
        browseable = Yes
```

Überprüfen der Konfiguration und erneutes Einlesen der Konfiguration:

```
# testparm
# systemctl reload smbd.service
```

Lab S06: Samba-Client einrichten

Ziel:

- Login auf node2
- Installation des Pakets `smbclient`
- Auflisten aller verfügbaren Freigaben
    - Benutzername: `user`
- Einhängen der `labs`-Freigabe nach `/var/labs`
- Überprüfen, ob der Mount erfolgreich war

Vorgehensweise

Installation des Samba-Clients:

```
# apt-get install smbclient cifsutils
```

Auflisten verfügbarer Freigaben:

```
$ smbclient -U user -L //node1
Password for [WORKGROUP\user]:

        Sharename       Type      Comment
        ---------       ----      -------
        print$          Disk      Printer Drivers
        labs            Disk      Lab Solutions
        IPC$            IPC       IPC Service (Samba 4.20.2)
        user            Disk      Home Directories
SMB1 disabled -- no workgroup available
```

Einhängen der `labs`-Freigabe nach `/var/labs`:

```
# mkdir /var/labs
# mount.cifs //node1/labs -o user=user /var/labs
Password for user@//node1/labs:
```

Überprüfen, ob der Mount erfolgreich war:

```
# df -h /var/labs
Filesystem      Size  Used Avail Use% Mounted on
//node1/labs     19G  2.6G   16G  14% /var/labs

# ls /var/labs
functions.sh  tests  01  02  03
```

Lab S07: NFS-Server einrichten

Ziel:

- Login auf node1
- Paket `nfs-utils` installieren
- Erstellen einer Freigabe

- Rechner: node2
- Pfad: /labs
- nur lesen

Vorgehensweise

Installation des Pakets sowie Aktivieren des Dienstes:

```
# dnf install nfs-utils
# systemctl enable --now nfs-server.service
```

Anlegen einer Freigabe durch Anpassen der Datei /etc/exports:

```
/labs   node2(ro)
```

Erneutes Einlesen der Konfiguration:

```
# exportfs -ra
```

Lab S08: NFS-Client einrichten

Ziel:

- Login auf node2
- Installation des Pakets nfs-common
- Einhängen der /labs-Freigabe nach /import/labs

Vorgehensweise

Installation des NFS-Clients:

```
# apt-get install nfs-common
```

Einhängen der Freigabe:

```
# mkdir /import/labs
# mount.nfs4 node1:/labs /import/labs
```

Überprüfen, ob der Mount erfolgreich war:

```
# df -h /import/labs
Filesystem      Size  Used Avail Use% Mounted on
node1:/labs     19G  2.6G   16G  14% /import/labs

# ls /import/labs
functions.sh  tests  01  02  03
```