Lab 9 - FILES II

friscdom | lederluc | wehrltob

19. Mai 2023

Task 1 – Uncompressed and compressed btrfs, defragmentation

> In this task you will create two volumes in the ned.cloudlab.zhaw.ch OpenStack environment and attach them to an existing instance. Then you will partition the volumes and format the partitions with btrfs and mount them uncompressed and compressed. At the end you will defragment and simultaneously compress the uncompressed file system.

Create two volumes of 8 GB each in the ned.cloudlab.zhaw.ch OpenStack environment. To do this, click on the button “Create Volume” in the Volume section.

- open Openstack
- click Volumes
- click Volumes
- Create Volume (twice)
    - pick name
    - Size: 8GB
    - click Create Volume

Attach the two volumes to an existing running instance. Select therefore the action “Manage Attachments”.

- For both volumes
    - Manage Attachments
    - Pick Instance
    - Attach Volume

Verify that volumes are attached.

- Table now shows "Attached To" `/dev/vdX on [Instance Name]`
- `lsblk` on VM now shows two new volumes

```
ubuntu@bsylab:~$ lsblk -e7
NAME    MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
vda     252:0    0   40G  0 disk
├─vda1  252:1    0 39.9G  0 part /
├─vda14 252:14   0    4M  0 part
└─vda15 252:15   0  106M  0 part /boot/efi
vdb     252:16   0    8G  0 disk
vdc     252:32   0    8G  0 disk
```

  - -e7 to exclude loop devices

Create three directories /data, /data1 and /data2

`mkdir /data /data1 /data2`

and a primary partition on the first volume (/dev/vdx) and on the second volume (/dev/vdy). The partitions should cover 25% of the block devices.

- `fdisk /dev/vdb`
  - Type `p` to print the partition table and verify that there are no existing partitions on the device.
  - Type `n` to create a new partition.
  - Type `p` to create a primary partition
  - When prompted for the partition number, type `1`
  - When prompted for the first sector, press `Enter` to use the default value.
  - When prompted for the last sector, enter `+2G`
  - Type `p` to print the partition table and verify the new Partition. You can also type `F` to view unpartitioned space
  - Type `w` to write the changes to disk and exit.

How can you be sure that a volume is not already partitioned?

- `lsblk` shows no partitions
- typing `p` in `fdisk` shows no partitions

Format the first partition of the first volume /dev/vdx1 with btrfs and mount it on /data1.

- `mkfs.btrfs /dev/vdb1`
- `mount /dev/vdb1 /data1`
- `mkfs.btrfs /dev/vdc1`
- `mount -o compress=zstd /dev/vdc1 /data2/`

Verify the disk usage in /dev/vdx1 and /dev/vdy1 with btrfs and Linux commands

```
root@bsylab:~# df -h -x squashfs -x tmpfs -x devtmpfs
Filesystem      Size  Used Avail Use% Mounted on
/dev/vda1        39G  3.6G   36G  10% /
/dev/vda15      105M  6.1M   99M   6% /boot/efi
/dev/vdb1       2.0G  3.6M  1.8G   1% /data1
/dev/vdc1       2.0G  3.6M  1.8G   1% /data2
```

`-x squashfs -x tmpfs -x devtmpfs` to exclude loop, tmpfs and udev devices

```
root@bsylab:~# du -h /data1 /data2
16K     /data1
16K     /data2
```

Is the disk usage equal ?

- yep

List the compression algorithms/methods you can choose from.

- ZLIB
  - slower, higher compression ratio
  - levels: 1 to 9, mapped directly, default level is 3
  - good backward compatibility
- LZO
  - faster compression and decompression than ZLIB, worse compression ratio, designed to be fast
  - no levels
  - good backward compatibility
- ZSTD
  - compression comparable to ZLIB with higher compression/decompression speeds and different ratio
  - levels: 1 to 15, mapped directly (higher levels are not available)

- since 4.14, levels since 5.1

Create a few big files in /data1 and /data2.

```
dd if=/dev/zero of=/data1/bigfile_zero bs=1M count=800;
dd if=/dev/urandom of=/data1/bigfile_urand bs=1M count=800;
dd if=/dev/zero of=/data2/bigfile_zero bs=1M count=800;
dd if=/dev/urandom of=/data2/bigfile_urand bs=1M count=800;
```

Verify with the df command, if the file has been compressed in /data2.

```
root@bsylab:~# df -m -x squashfs -x tmpfs -x devtmpfs
Filesystem     1M-blocks  Used Available Use% Mounted on
/dev/vda1          39511  3689     35807  10% /
/dev/vda15           105     7        99   6% /boot/efi
/dev/vdb1           2048  1608       226  88% /data1
/dev/vdc1           2048   833      1001  46% /data2
```

Create a big file in /data and copy it to /data1 and /data2

```
dd if=/dev/zero of=/data/bigfile_copy0 bs=1M count=100;
cp /data/bigfile_copy0 /data1/bigfile_copy0;
cp /data/bigfile_copy0 /data2/bigfile_copy0;
```

You can verify again with the df and du commands, if the file has been compressed in /data2.

```
root@bsylab:~# df -m -x squashfs -x tmpfs -x devtmpfs
Filesystem     1M-blocks  Used Available Use% Mounted on
/dev/vda1          39511  3789     35707  10% /
/dev/vda15           105     7        99   6% /boot/efi
/dev/vdb1           2048  1708       126  94% /data1
/dev/vdc1           2048   837       998  46% /data2
```

Look at $ man btrfs filesystem. Defragment /dev/vdx applying the compression algorithm zlib and compare the disk usage of /data1 and /data2.

- `defragment [options] <file>|<dir> [<file>|<dir>...]`
- `btrfs filesystem defragment -rv -czlib /data1/`

```
root@bsylab:~# df -m -x squashfs -x tmpfs -x devtmpfs
Filesystem     1M-blocks  Used Available Use% Mounted on
/dev/vda1          39511  3789     35707  10% /
/dev/vda15           105     7        99   6% /boot/efi
/dev/vdb1           2048  1045       790  57% /data1
/dev/vdc1           2048   837       998  46% /data2
```

Task 2 – Btrfs file encryption and automatic volume mount

> In this task you will create and automatically mount an encrypted storage device.

Does btrfs support native file encryption?

- No. https://unix.stackexchange.com/a/423127

> To encrypt files with the 3rd party tool dm-crypt install btrfs-progs and cryptsetup if needed.

- `apt install btrfs-progs cryptsetup`

Generate a new 64 bytes long encryption key and store it in the file /etc/cryptkey

> `sudo dd if=/dev/urandom of=/etc/cryptkey bs=64 count=1`

Change the mode of the cryptkey file to 600, so that only the root has r/w access to it

- `chmod 600 /etc/cryptkey`

Look also at $ man cryptsetup and encrypt the storage device

- `umount /dev/vdb1`
- > `sudo cryptsetup -v --type luks2 luksFormat /dev/vdb /etc/cryptkey`
  - https://unix.stackexchange.com/a/558489

- `--type <device-type>` Device type can be plain, luks (default), luks1, luks2, loopaes or tcrypt
- `luksFormat <device> [<key file>]`
  - > Initializes a LUKS partition and sets the initial passphrase
  - > You can only call luksFormat on a LUKS device that is not mapped.
  - > To use LUKS2, specify --type luks2.

Open the encrypted storage device vdx and map it to the computer as a data storage device

- `sudo cryptsetup open --key-file=/etc/cryptkey --type luks2 /dev/vdb data`
  - `open <device> <name> --type <device_type>`
  - > Opens (creates a mapping with) backed by device

> Now, the decrypted storage device will be available in the path /dev/mapper/data. The desired btrfs files system has to be created in the /dev/mapper/data device and then mounted to the /dev/mapper/data device instead of /dev/vdx from now on.

> `sudo mkfs.btrfs -L data /dev/mapper/data sudo mount /dev/mapper/data /data1`

> As you can see, the btrfs file system created on the encrypted storage device vdx is mounted in the /data1 directory.

Create the file HelloWorld.txt in /data1 with some readable content.

- `echo "some readable content" > /data1/HelloWorld.txt`

What happens, if you try to mount /dev/vdx directly to another directory?

```
root@bsylab:~# mount /dev/vdb /mnt/
mount: /mnt: unknown filesystem type 'crypto_LUKS'.
```

> To automatically mount the encrypted btrfs file system at boot time proceed as follows: Decrypt the storage device /dev/vdx at boot time using the encryption key in /etc/cryptkey Find therefore the UUID of /dev/vdx using the blkid command

```
root@bsylab:~# blkid | grep -v "loop"
/dev/vda1: LABEL="cloudimg-rootfs" UUID="6b18114a-e2da-4d1f-b897-
39b55cca2031" TYPE="ext4" PARTUUID="75d672b7-da2b-40f1-aa53-e88a21390460"
/dev/vda15: LABEL_FATBOOT="UEFI" LABEL="UEFI" UUID="C54E-14DF" TYPE="vfat"
PARTUUID="683364ed-ddbc-4c40-9773-6a2ca2b55d82"
/dev/vdc1: UUID="f69639a6-2b0c-4106-8fe0-710e45a58313" UUID_SUB="11961ae5-
3e79-4318-a1f5-7e0141a255c2" TYPE="btrfs" PARTUUID="890a9d23-01"
/dev/vdb: UUID="01b81a77-4d2c-4abd-b284-cad0ced400aa" TYPE="crypto_LUKS"
/dev/mapper/data: LABEL="data" UUID="7583d0c2-5a15-445a-8688-7f85bb19e0ff"
UUID_SUB="20227d22-3a66-410a-a1d4-f79d179cd0fc" TYPE="btrfs"
/dev/vda14: PARTUUID="23357df2-4bcc-424a-88da-28461cba69b2"
```

`01b81a77-4d2c-4abd-b284-cad0ced400aa`

Edit the file /etc/crypttab so, that the vdx storage device is automatically decrypted at boot time.

- `echo "data UUID=01b81a77-4d2c-4abd-b284-cad0ced400aa /etc/cryptkey luks,noearly" >> /etc/crypttab`

Mount the decrypted storage device /dev/mapper/data to the /data1 directory.

> Find therefore the UUID of the decrypted /dev/mapper/data storage device

- `blkid /dev/mapper/data`
- `UUID="7583d0c2-5a15-445a-8688-7f85bb19e0ff"`

Edit the /etc/fstab file so, that /dev/mapper/data is automatically mounted to /data1

- `echo "UUID=7583d0c2-5a15-445a-8688-7f85bb19e0ff /data1 btrfs defaults 0 0" >> /etc/fstab`

Reboot the system and verify that the encrypted storage device vdx is decrypted, mounted and the HelloWorld.txt file is still there.

```
root@bsylab:~# cat /data1/HelloWorld.txt
some readable content
root@bsylab:~# lsblk -e7
NAME    MAJ:MIN RM  SIZE RO TYPE  MOUNTPOINT
vda     252:0    0   40G  0 disk
├─vda1  252:1    0 39.9G  0 part  /
├─vda14 252:14   0    4M  0 part
```

```
└─vda15 252:15   0  106M  0 part  /boot/efi
vdb     252:16   0    8G  0 disk
└─data  253:0    0    8G  0 crypt /data1
vdc     252:32   0    8G  0 disk
└─vdc1  252:33   0    2G  0 part
root@bsylab:~# cat /data1/HelloWorld.txt
some readable content
```