w

Lab 1:

Creating a NAS

and a Linux SAN   
with  
 Redundancy   
& Flexibility

Datacenter Virtualization

2024-2025

© Daan Pareit

CONTENT

[1 Introduction 3](#_Toc178861898)

[1.1 Learning goals 3](#_Toc178861899)

[1.1.1 Knowledge 3](#_Toc178861900)

[1.1.2 Skills 3](#_Toc178861901)

[1.2 Installing VMware virtualization software 3](#_Toc178861902)

[2 Exploring RAID and iSCSI in a Synology NAS 5](#_Toc178861903)

[2.1 Emulating a Synology NAS 5](#_Toc178861904)

[2.2 Creating a RAID5 array 6](#_Toc178861905)

[2.3 Use your NAS to store data 6](#_Toc178861906)

[2.4 RAID redundancy 7](#_Toc178861907)

[2.4.1 Emulating disk failure 7](#_Toc178861908)

[2.4.2 Restoring redundancy 8](#_Toc178861909)

[2.5 iSCSI 10](#_Toc178861910)

[3 RAID 13](#_Toc178861911)

[3.1 Preface 13](#_Toc178861912)

[3.2 Create a Virtual Machine 14](#_Toc178861913)

[3.3 Installing Linux with software RAID 14](#_Toc178861914)

[3.4 Verifying the RAID arrays 21](#_Toc178861915)

[3.5 Usage and impact of disk failure 22](#_Toc178861916)

[3.6 Recovery 25](#_Toc178861917)

[3.6.1 Fix booting from a failed RAID array with no redundancy 25](#_Toc178861918)

[3.6.2 Rebuild the RAID array with redundancy 26](#_Toc178861919)

[4 LVM 29](#_Toc178861920)

[4.1 Let’s go 29](#_Toc178861921)

[4.2 Expansion! 32](#_Toc178861922)

[5 iSCSI 36](#_Toc178861923)

# 

# Introduction

**RAID** (Redundant Array of Independent/Inexpensive Disks) can be done in software on both Linux and Windows. As the hardware requirements for a modern Linux system are still a lot lower compared to a similar Windows system, many organizations use software RAID on Linux to set up a low-cost backup server.  
Old hardware (containing many disks) can get a second life using this setup where the use of redundant disks provides extra levels of safety. Performance is also not too bad, and, in some cases, they can be equally fast as an entry-level hardware RAID solution.

A big downside of RAID is the lack of flexibility: expansion is not always that easy. However, Linux provides a different technology to allocate disk space on-the-fly or move it around. This is called **LVM** (Logical Volume Manager) and is built into most, if not all, Linux distributions. When combined with RAID, we get a very good combination of a safe and flexible storage server.

Storage is, of course, only useful when it can be accessed. Therefore, we will need to set it up on a NAS or SAN. In this lab, we’ll set it up in a NAS as well as a SAN configuration using **iSCSI**.

## Learning goals

### Knowledge

* Know what RAID is
* Know how RAID volumes are called in Linux
* Know what LVM is
* Know what iSCSI targets are

### Skills

* Be able to configure RAID volumes in Linux, during and after installation
* Be able to configure and extend LVM volumes in Linux
* Be able to configure iSCSI Targets in Linux

## Installing VMware virtualization software

We will use VMware to work with Virtual Machines. You need to install:

* VMware Workstation Pro (on Windows)
* or VMware Fusion Pro (on Mac).

Since mid-2024, Broadcom (which acquired VMware in 2023) has been offering this software for free.

First you need to register at: <https://profile.broadcom.com/web/registration>

Next, you can download the software for personal usage:

* Workstation Pro (Windows/Linux): <https://support.broadcom.com/group/ecx/productdownloads?subfamily=VMware+Workstation+Pro>
  + Note for Windows users: version 17.6.0 has a bug causing it only to run on Windows editions which have English as system language. In that case, install version 17.5.2.
* Fusion Pro (Mac): <https://support.broadcom.com/group/ecx/productdownloads?subfamily=VMware%20Fusion>

# Exploring RAID and iSCSI in a Synology NAS

Before delving deeper in setting up RAID and iSCSI manually, we’ll first explore these concepts via an easy-to-use interface of a commercial NAS brand ‘Synology’.

## Emulating a Synology NAS

We’ve have created a preconfigured virtual machine for you. This is an ‘ova’ file, which you can open with your VMware Workstation/Fusion.

The virtual machine emulates the Synology DS920+ NAS for educational testing purposes. (Based on <https://github.com/RROrg/rr> if you’re interested.)



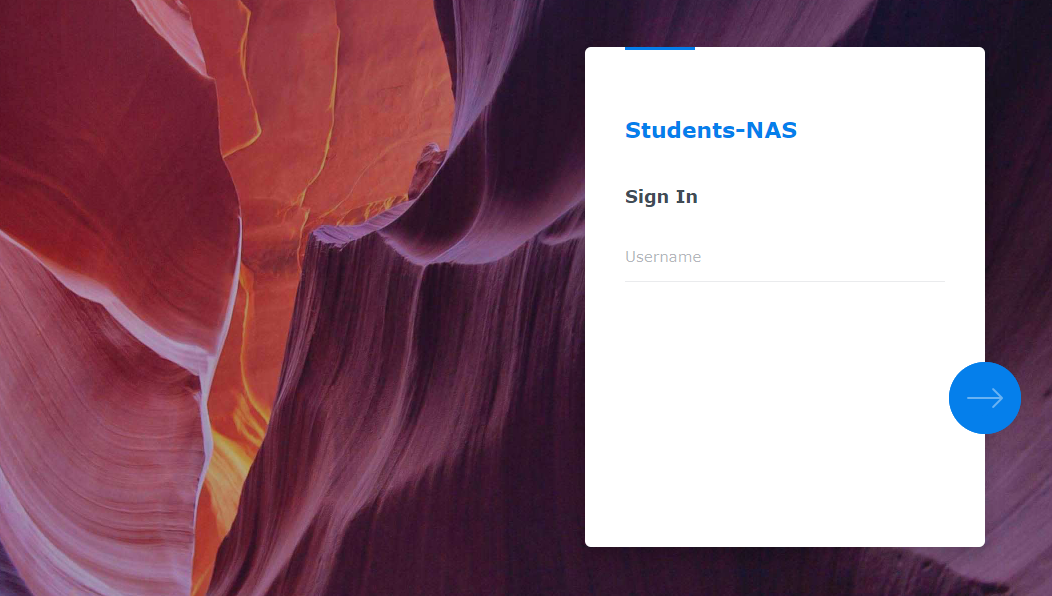
Figure 1 Synology DS920+

🡪 The VM contains a specific bootloader for emulating the Synology NAS. When booting, you’ll get a screen like the one below. It mentions a url http://<ip address>:5000 to connect to. You’ll have to give the system a few minutes to boot (you could check if it’s up and running by continuously pinging the IP address) and then surf to that URL.

A screenshot of a computer

Description automatically generated

🡪 The emulated NAS uses the DiskStation Manager (DSM) Linux-based operation system. When surfing to the URL, you’ll be presented with a login screen.



Use the following credentials:

* Username: Students
* Password: Howest42

## Creating a RAID5 array

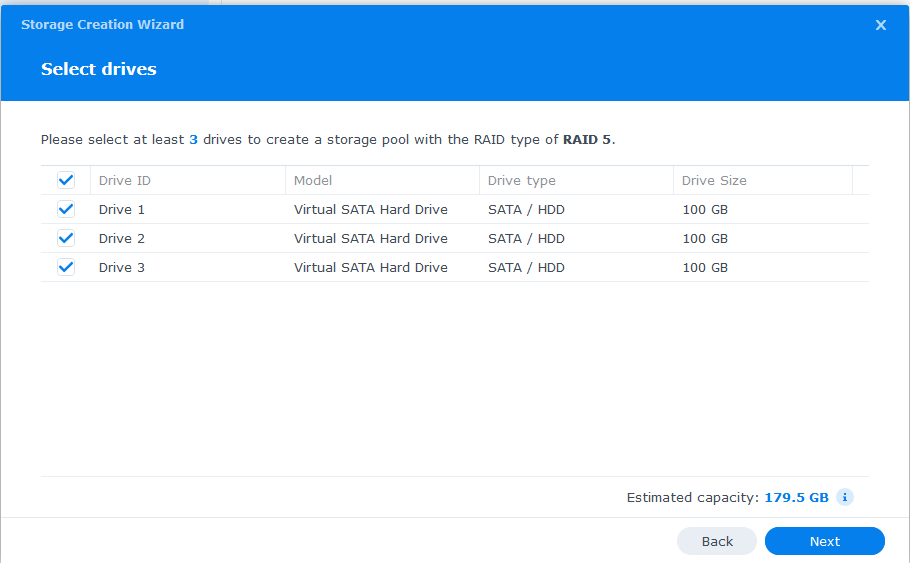
The NAS has the DSM Operating System installed and contains a few SATA disks which aren’t used yet.

🡪 In the DSM web interface, go to ‘Storage Manager’ (Dutch: ‘Opslagbeheer’).

🡪 Create a RAID5 array with maximum capacity.

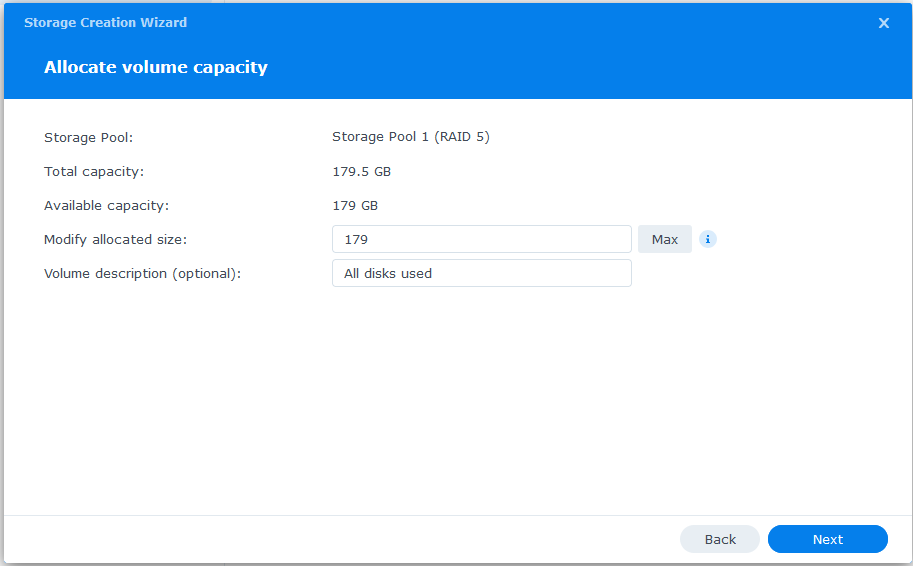
**Q: How many disks do you need to use? What is the capacity of all disks together?**

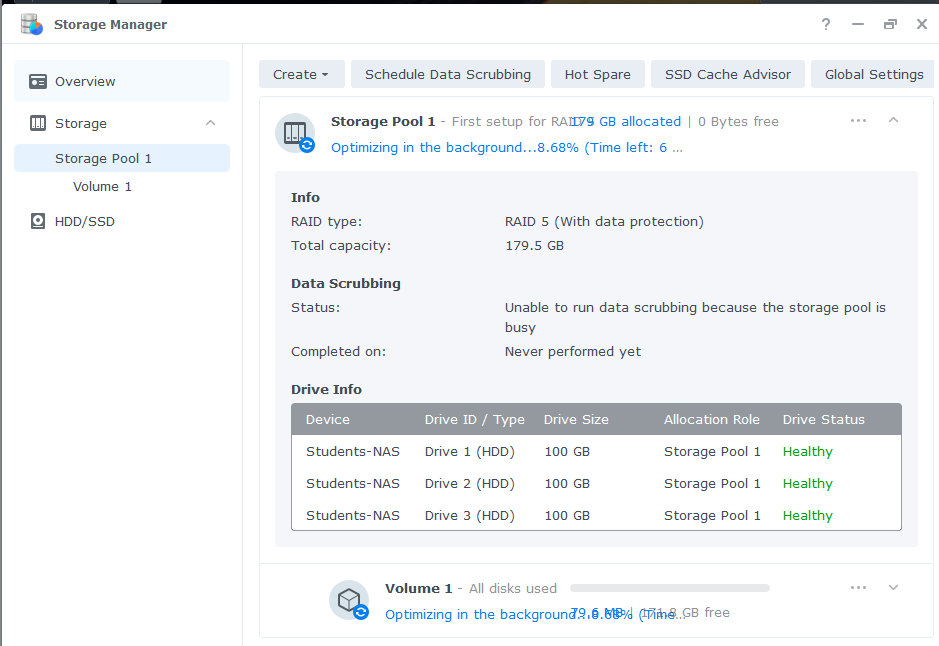
**I am going to use 3 disks. 300 GB**

****

**Q: What is the net capacity of your RAID5 array when created? Note that DSM itself already consumes about 10 GB of a disk. Does the net capacity then make sense?**

**179 GB, and yes it seems like it makes sense**

****

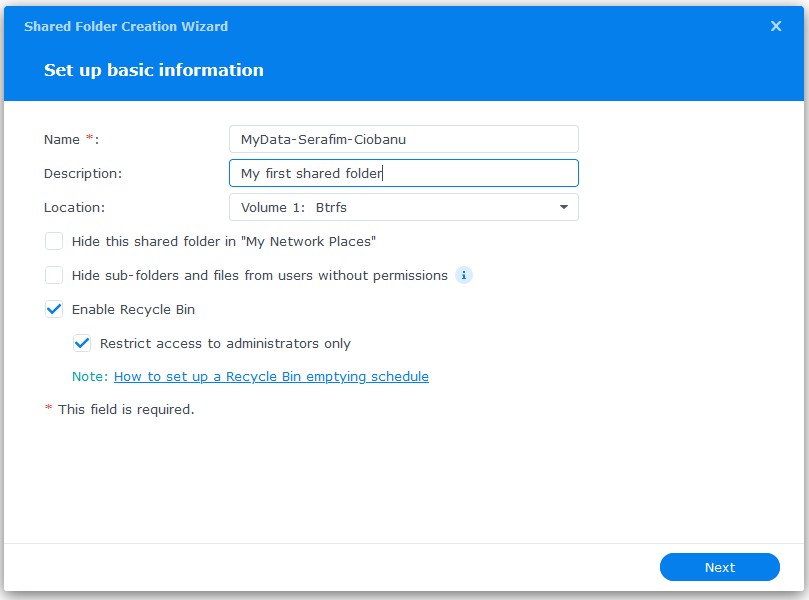
****

## Use your NAS to store data

We’ve turned the disks into a useful RAID array, now create a way to access the NAS remotely

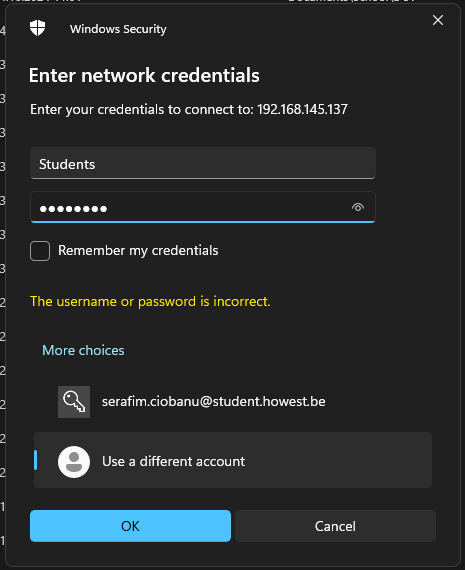
🡪 In the Control Panel, created a shared folder called ‘MyData-<firstname>-<lastname>’ (and keep all default settings).

Control Panel > Shared Folder > Create > Create Shared folder



🡪 On your host PC, now connect to that shared folder (\\<ip-address>) with the NAS credentials.

**Q: Take a screenshot of your host being connected to that shared folder (including IP address and folder name).**

****

**A screenshot of a computer

Description automatically generated**

🡪 On Leho, you’ll find some sample data: ‘bart\_simpson.zip’. Save that precious file in that shared folder on your NAS and unzip it there.

**Q: When putting files on that shared folder, are you doing ‘file IO’ or ‘block IO’ over the network?**

**I believe it is file IO, since I am only sending files and not doing anything specific with the block devices**

## RAID redundancy

Now, let’s test RAID’s biggest advantage here: its redundancy in case of a disk failure.

### Emulating disk failure

🡪 To emulate a disk failure, go to the settings of your Synology VM and remove ‘Hard Disk 4’ while your VM is still running. On your DSM web interface, keep the ‘Storage Manager’ view open and look what happens.

**Q: What is your Storage Manager telling you?**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**Q: On your host PC, can you still access the data on your shared folder? Can you add new data?**

Yes, I can still access it.

A cartoon of a person in a room

Description automatically generated

I did a copy of a file, and looks like I still can add it.

### Restoring redundancy

To restore redundancy, we need to replace the failed disk with a replacement disk.

🡪 Therefore, add a new **SATA** **100 GB** disk in the NAS via the VM settings. It’s hot pluggable, so you can do this while the VM is still running. You don’t want any downtime of your data availability.

VM > Settings > Add > Hard Disk > SATA > ADVANCED + VIRTUAL DEVICE NODE SATA 0:4 (OTHERWISE IT WOULD BREAK)

A screenshot of a computer

Description automatically generated

That new drive should now be detected by your NAS. You’ll see a message appearing and the drive will appear in your Storage Manager overview.

🡪 Choose to repair your RAID array now.

**Q: What kind of action do you see DSM is doing to restore the RAID array?**

**It is performing data scrubbing!**

Voila, after a few minutes your data is protected from disk failure again!

## iSCSI

Now, let’s turn our focus on another concept: iSCSI. The key idea here is to provide remote storage to your PC as if it were a local disk rather than via file shares.

🡪 Open the ‘SAN Manager’ on your DSM web interface. Neglect its proposal to install specific desktop software on your host PC.

🡪 Create a 1 GB ‘LUN’, using Thin Provisioning. (Pick that option for keeping the footprint of your VM small. In real life, for performance reasons ‘Thick’ would be better.) Keep default settings about the created iSCSI ‘Target’ and the access permissions.

LUN > Create > Space allocation - Thin provisioning

🡪 Now, find the IQN name of the iSCSI Target you’ve created.

Edit > Mapping > IQN

iqn.2000-01.com.synology:Students-NAS.default-target.757d919845b

🡪 Windows has a built-in ‘iSCSI Initiator’ tool. Use the Windows iSCSI Initiator of your Windows host OS (or within a Windows VM) to connect to this iSCSI Target.

Target - IP address

Then the disk is already connected.

🡪 Then, open your Windows ‘Disk Manager’. The LUN is made available as if it were a local disk!

A screenshot of a computer

Description automatically generated

🡪 Format that disk and put some data on it. You’re now using that ‘disk’ over iSCSI seamlessly integrated in your Windows.

A screenshot of a computer

Description automatically generated

**Q: When putting files on that drive, are you doing ‘file IO’ or ‘block IO’ over the network?**

**Block IO, since we are talking directly to the devices themselves, and not a file system first.**

# RAID

## Preface

In Linux, software RAID arrays always have the device name **md[x]** (where ‘md’ stands for ‘Multiple Device’). Once created correctly, we can use these as any other block device within Linux.

First, we need to prepare the disks involved. In Linux software RAID both disks and partitions on these disks can be members within a RAID array. Of course, it is recommended that all parts in a RAID array have the same size.

All management tasks in RAID arrays can to be using the **mdadm** tool. Start by opening the [man page of this tool](https://manpages.debian.org/current/mdadm/md.4.en.html) and reading it carefully.

Once the arrays are made (CREATE), they must be activated (ASSEMBLE). Activation needs to be done every time the system reboots or if the array was deactivated for any reason.

Once activated, the RAID arrays are ready to be used. We can monitor the state of all different RAID arrays on a system be reading the **/proc/mdstat** file. E.g. using the command “cat /proc/mdstat”, we can confirm that the arrays are online.

Before we can use the created and activated array, we first need to format it with a file system. This can be any file system of choice.

The easiest way to create and configure RAID is by doing it during the system installation, which you’ll do in the next step. In this part of the lab we will create two sorts of software RAID: RAID0 and RAID1.

**Q: Find the other RAID types (numbers only) which can be created using Linux software RAID.**

## Create a Virtual Machine

🡪 Download a Linux installation ISO file. We suggest the most recent Debian AMD64 netinst ISO from <https://cdimage.debian.org/debian-cd/current/amd64/iso-cd/>

🡪 Create a Virtual Machine (VM) suitable for the latest Debian x64 in VMware Workstation/Fusion. Customize the hardware as follows:

* 500GB hard disk (store virtual disk as a single file)
* Verify the network adapter is set to connect to the NAT network (Internet connectivity)
* The installation ISO is configured as your virtual CD/DVD drive and “*Connect at power on*” is checked

🡪 Before booting, now customize the hardware to add a *second* VMDK drive of 600 GB (do not allocate all disk space).

A screenshot of a computer

Description automatically generated

🡪 If you just created this VM, all disks are, of course, empty. Now launch the VM and it should start from the ISO file.

## Installing Linux with software RAID

🡪 Select “Install” and **not** Graphical Install. This way it looks the same on Ubuntu, Debian etc.

🡪 Select “**English**” as the language, the location & keymap can be Belgian if you have AZERTY keyboard. Otherwise even your command line environment might be entirely different.   
Also: for locale en\_US.UTF-8 is just fine.

🡪 Choose **debian-<firstname>-<lastname>** as hostname.  **(This important to verify your individual assignment for grading!)**

🡪Domain name, users and passwords are entirely your choice. But please remember these (and take notes).

🡪 Watch out when you arrive at the disk configuration (“Partition disks”). Select the **Manual** **partitioning method**. Read the full description below before finishing the partitioning!

A screenshot of a computer

Description automatically generated

We will create 2 RAID volumes, since we want to experiment with 2 different kinds of RAID arrays: RAID0 and RAID1. We will create several partitions to be part of each array.

🡺 Make sure to have following partitions (and file systems) for drive 1:

* /dev/sda1 - 1GB - primary - /boot - ext2 (**make sure it’s bootable**)

A screenshot of a computer

Description automatically generated

* /dev/sda2 - 90GB - primary - / (root filesystem) - ext4

A screenshot of a computer

Description automatically generated

* /dev/sda3 - 5GB - primary - SWAP Space

A screenshot of a computer screen

Description automatically generated

* /dev/sda5 - 100 GB - logical - N/A - RAID ## same size as on sdb

A screenshot of a computer screen

Description automatically generated

* /dev/sda6 - 100 GB - logical - N/A - RAID ## same size as on sdb

A screenshot of a computer screen

Description automatically generated

* /dev/sda7 - remainder - logical - N/A - LVM

A screenshot of a computer screen

Description automatically generated

A screenshot of a computer

Description automatically generated

**Where is /dev/sda4? Remember your ‘Forensic Analysis’ course: sda5, sda6 & sda7 are only possible when using extended partitions. This is because the partition table of the Master Boot Record can only have 4 entries. Thus sda1, sda2 and sda3 are ‘Primary’ partitions and sda5, sda6, sda7 are ‘Logical’ partitions within the implicit sda4 ‘Extended’ partition.**

🡺 Make sure to have following partitions (and file systems) for the second drive:

* /dev/sdb1 - 100 GB – N/A – RAID ## same size as on sda

A screenshot of a computer screen

Description automatically generated

* /dev/sdb2 - 100 GB – N/A – RAID ## same size as on sda

A screenshot of a computer screen

Description automatically generated

* /dev/sdb3 - 200 GB – N/A – LVM

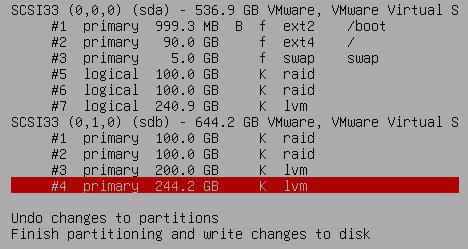
A screenshot of a computer screen

Description automatically generated

* /dev/sdb4 - remainder – N/A – LVM

A screenshot of a computer screen

Description automatically generated



sdb1, sdb2, sdb3 and sdb4 can all be primary partitions.

You have a certain liberty as to the sizes of the partitions, the sizes above are just examples. But do make sure that all RAID partitions are equal in size, at least if they are intended to be in the same RAID set together.

If you used the example sizes, it should now look like this:

A screenshot of a computer

Description automatically generated

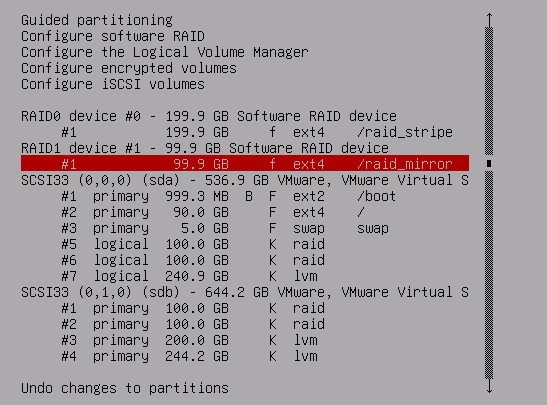
🡪 **Do NOT select “Finish partitioning and write changes to disk”** because the RAID devices are not yet configured. Go upwards and select “Configure software RAID” now.

🡪 Now configure **RAID0** to be on devices **sda5 and sdb1**, whereas **RAID1** should be configured with devices **sda6 and sdb2** (no spare devices).

**Q: Try to predict the sizes of the two RAID sets before actually configuring them. Why do they have a different size?**

**RAID 0 is working as a single device, hence it doubles the size, since the file is distributed between the 2 disks**

**RAID 1 is doing a backup on the other disk, so you only get on disk worth of space.**

****

🡪 After creating these, still in the installer, choose to use these with the ext4 file system. Next, give them a suitable mount point, you’ll have to enter it manually:

* For RAID0: /raid\_stripe – ext4
* For RAID1: /raid\_mirror – ext4

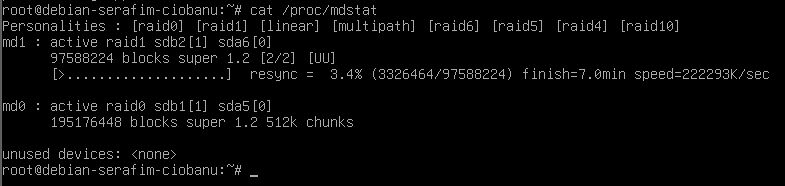
A screenshot of a computer program

Description automatically generated

🡪 Once you are satisfied with the partitioning work, **proceed with the installation** of Linux.

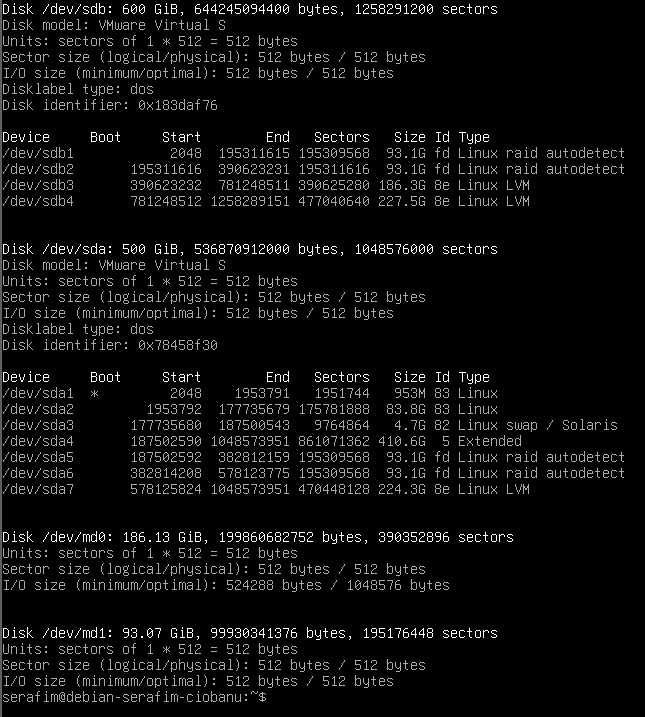
🡪 Make sure to **UNCHECK** installing a GUI (*desktop environment*), use **space bar** to deselect this option. For everything else, it is safe to use the defaults (or even better: only select system utilities and SSH).

After installation (including putting the boot loader on **/dev/sda**) and a successful reboot, the operating system should be ready for use. Add the ‘sudo’ package to your debian and your regular user to the sudo group.



apt install sudo

usermod -aG sudo serafim



🡪 Now log in and verify the disk space that is available for you, use tools like fdisk, df, du and mount to **verify** that what you expected is indeed configured correctly.

**Q: What is the device name for our two RAID arrays? Also, have a look at /proc/mdstat . (If done quickly after debian installation, you might see the initial resync). What does the “[2/2] [UU]” in the output stand for? Check** [**https://raid.wiki.kernel.org/index.php/Mdstat**](https://raid.wiki.kernel.org/index.php/Mdstat) **to find out.**

**The device names are /dev/md0 and /dev/md1 which stand for “multiple disks”**

**2/2 means there are 2 devices being used**

**UU both devices are running, or up.**

## Verifying the RAID arrays

Usually, when RAID arrays are configured manually (e.g. after installation), they are not available after a reboot. Since we just created them during installation, this is already taken care of. Let’s verify what the Linux installer has configured by looking at */etc/mdadm/mdadm.conf*.

A screenshot of a computer

Description automatically generated

It can also be helpful to look at the manpage for this file: *man mdadm.conf*  
This file can also be configured by using the mdadm tool.

**Q: For example, try this command and explain all output, please replace <device>:***# mdadm --detail <device>***What is the ‘Consistency policy’ for our RAID0 array and for our RAID1 array? What does it mean (check the man page)?**

**man mdadm, not man mdadm.conf**

**Specify how the array maintains consistency in the case of an unexpected shutdown. ONLY RELEVANT FOR RAID LEVELS WITH REDUNDANCY.**

**Resync - Full resync is performed and all redundancy is regenerated when the array is started after an unclean shutdown**

A screenshot of a computer

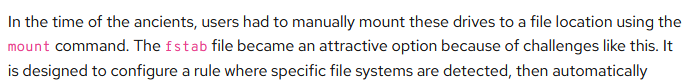
Description automatically generated

A screenshot of a computer

Description automatically generated

These devices are not only active, but also formatted AND mounted.   
**Q: which file in Linux is responsible for automatically mounting file systems?**

**nope, it is /etc/fstab**

****

## Usage and impact of disk failure

Now that our RAID arrays are ready for use, we can test these and emulate a disk failure.

🡪 First, we need to add some files on the mounted file systems. On Leho, we have some sample files: bart\_simpson.zip and lisa\_simpson.zip . Download these and upload these easily to your VM via the scp command in your laptop terminal (on your debian, install openssh-server first if necessary). Put bart\_simpson.zip in the folder where your RAID0 device is mounted and lisa\_simpson.zip in the one of your RAID1 device, respectively.

bart > /raid\_stripe

lisa > /raid\_mirror

🡪 Before we test redundancy, we want to make sure that these files are actually written on disk and that any file system buffers are cleared. Do this now by running the commands  
sync; echo 3 > /proc/sys/vm/drop\_caches

Now let’s test and compare the (lack of) redundancy of both arrays and emulate a disk failure!

🡪 **Without shutting it** down, open the VM settings and remove the **second** drive (the one of 600 GB) from the VM. Then, try to read and write from both mount points by unzipping the files (e.g. apt install unzip followed by unzip <zipfile> )

🡪 Which array is still working fine which array is throwing errors? It may take some minutes before an I/O error message appears (look for errors in the main console, not inside an SSH session)

A screenshot of a computer

Description automatically generated

RAID 0 throwing errors.

A screenshot of a computer

Description automatically generated

RAID 1 said that it now works on one device.

**Q: Which files are still available? Try copying them to your home folder.**

**RAID 1 allows the full directory to be copied**

**RAID 0 folder allows only the files that managed to be unzipped.**

**Q: Explain the contents of the /proc/mdstat file.**

**A screenshot of a computer

Description automatically generated**

**It means that there is only 1 device working out of 2, and one of the devices is up (U) and one is down (\_)**

**Q: Is it still possible to add a file to the mirror folder?**

**A screen shot of a computer

Description automatically generated**

**Yes, it is still possible.**

## Recovery

Now it’s time to start the recovery from the disk failure.

🡪 Let’s add a new hard drive to replace the failed disk. Shut down the VM, open the settings and add a new Hard Drive. Make it the same size or more than the original one (i.e. 600 GB or more).

### Fix booting from a failed RAID array with no redundancy

🡪 Upon booting the VM, some start jobs might fail, and we might drop into an emergency boot mode.   
Wait 90 seconds and then enter the root password (we told you to remember / take note of the password, didn’t we? 😉).

A computer screen with white text

Description automatically generated

**Q: The reason for being dropped in emergency boot mode is that one of the RAID arrays is completely useless and therefore cannot be mounted, which one?**

**RAID 0**

**🡪** While being in the emergency mode, edit the /etc/fstab file and remove (or comment out) the line with the faulty RAID array, then reboot.

A screen shot of a computer program

Description automatically generated

**🡪** After reboot: check the mdstat file again. Did something change?

A screenshot of a computer

Description automatically generated

Yes, now the RAID 0 array says inactive!

### Rebuild the RAID array with redundancy

**🡪** Now, use any partitioning tool (e.g. fdisk, …) to create the necessary partition on the replacement hard disk to rebuild the RAID array which is still operational. Only one partition is enough, just make sure it is equal or larger in size than the first one. (in fdisk, use ‘t’ to set partition type to raid)

A screenshot of a computer

Description automatically generated

A screenshot of a computer screen

Description automatically generated

**🡪** Finally, let’s rebuild our degraded (but still active) array by adding the newly created partition to the array. Only one mdadm command is necessary for this, make sure to note it down:

sudo mdadm --manage /dev/md1 --add /dev/sda1

**🡪** End by inspecting the mdstat file and by running mdadm *--detail /dev/md<x>* to show the rebuild status.

A screenshot of a computer

Description automatically generated

A computer screen with white text

Description automatically generated

**Q: Take a screenshot of the mdstat file while your Debian is recovering the RAID1 array.**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer program

Description automatically generated**

# LVM

The problem with RAID is the lack of flexibility: it is not easy to have one 100GB RAIDSET (which we have) and expand it to a 200GB RAIDSET/partition. This is where LVM comes in. When you’ve done all RAID exercises in the previous section, you currently have:

Hard drive 1 (the first original disk):

* /dev/sda1 - 1 GB - /boot - ext2 (make sure it’s bootable)
* /dev/sda2 - 90 GB - / (root filesystem) - ext4
* /dev/sda3 - 5 GB – SWAP Space
* /dev/sda5 - 100 GB (matching the sdb partition) – ext4 – broken RAID0
* /dev/sda6 - 100 GB – RAID1
* /dev/sda7 - remainder – N/A - LVM

Hard drive 2 (the replacement disk for the second disk which failed):

* /dev/sdb1 - 100 GB (matching the sda partition) – ext4 – RAID1 (rebuilt)

🡪 Note: The disk names ‘sda’ and ‘sdb’ could be reversed by removing and adding our disk, please verify before continuing!

A screenshot of a computer

Description automatically generated

In my case they are inverted, so not sure what to do about it.

The idea is to create a new LVM volume, copy the data from RAID1 to there and then expand that volume so that it uses the RAID1 array.

🡪 Remember the sequence: PV first, then VG and finally LV

**Q: What do PV, VG & LV mean?**

**Physical Volume (Partition itself I believe)**

**Volume Groups (Groups of partitions/PVs)**

**Logical Volumes (dividing VGs into other LV’s)**

🡪LVM is not installed by default on Debian, so install it by running *apt install lvm2*

## Let’s go

🡪Start by configuring sda7 as a PV, make sure to run *pvdisplay* and verify the creation.

sudo pvcreate /dev/sdb7 (since the partitions are inverted in my case, and that one is also a remainder)

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

🡪 Next: let’s create a VG within this PV, let’s set the name to “VG0”

sudo vgcreate VG0 /dev/sdb7

* Remember: *--help* is here to help!
* Verify by running *vgdisplay*

*A screenshot of a computer

Description automatically generated*

🡪Next, let’s create an actual useable LV within this VG. Make its size 100G

**sudo lvcreate -L 100G VG0**

**Q: If you don’t provide a name for this LV, what is the name then?**

**name can be specified with -n flag**

**lvol0**

**A black background with green and white text

Description automatically generated**

**A computer screen shot of a program

Description automatically generated**

Let’s format and use this device, let’s run a mkfs.ext4 on the LV, create a folder /lvmroot and mount it on that mountpoint:

* mkfs.ext4 <LV path>

A screenshot of a computer

Description automatically generated

* mkdir /lvmroot

A screenshot of a computer

Description automatically generated

* mount <LV path> /lvmroot

A screen shot of a computer

Description automatically generated

🡪 Copy all data from the still working RAID to this folder, just use the *cp* command for that. Verify by running *df -h*

*A computer screen shot of a program

Description automatically generated*

*A screenshot of a computer

Description automatically generated*

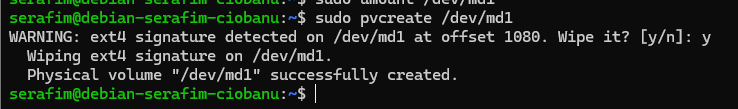
## Expansion!

🡪 To expand our LVM volume to the RAID1 array, we’ll first have to unmount the md-device.

sudo umount /dev/md1

🡪 If we now run a *pvcreate /dev/md*<x> command, we’ll be greeted with the message about an ext4 signature, confirm and run the command anyway.

sudo pvcreate /dev/md1



🡪 This means we have now formatted the md1 device to be used for LVM. So, all current data is lost. Luckily, we copied this to the already existing LVM volume (/lvmroot).

🡪 We will extend the VG, so let’s first verify the current size with *vgdisplay*

*A screenshot of a computer

Description automatically generated*

🡪 Now use *vgextend* to add the created PV to the current VG and verify again with *vgdisplay*

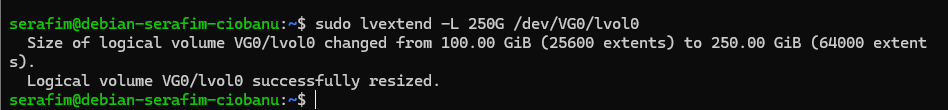
A black background with white text and green and blue text

Description automatically generated

A screenshot of a computer

Description automatically generatedsudo

🡪 Lastly, let’s try to expand the LV beyond the single PV size, e.g. make it a total of 250GB



**Q: What do you expect to see when you run *df -h* now? Which command can you use to ‘fix’ this?**

**A screenshot of a computer screen

Description automatically generated**

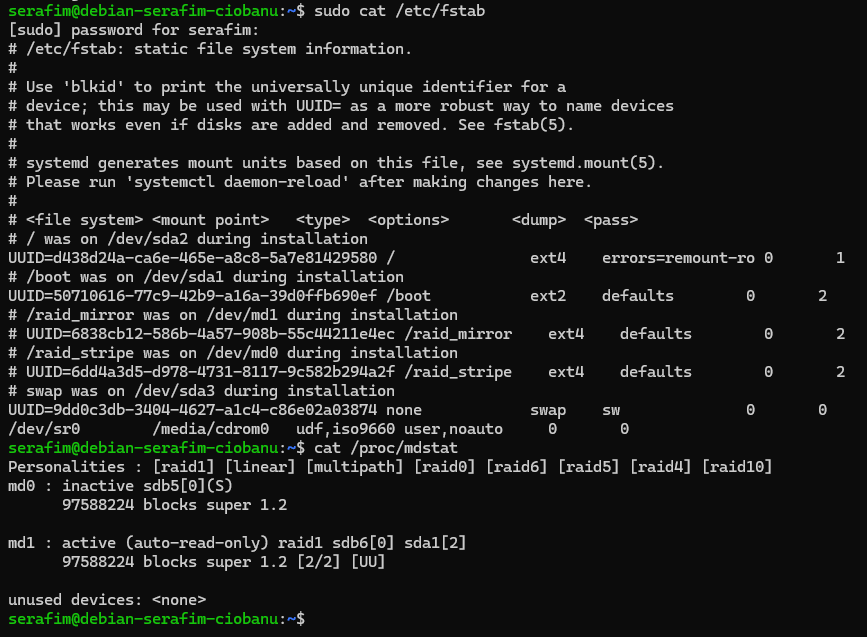
**A screenshot of a computer program

Description automatically generated**

**sudo resize2fs /dev/VG0/lvol0**

**This way I am most probably telling the mount point to check the real size of the PV and resize itself.**

**Wanted to shutdown to make a snapshot, but now the machine was getting into emergency mode, because it probably did not see the md1 anymore. I turned it off within the /etc/fstab because I do not know what to do with it. But the LVM can be still mounted.**

****

# iSCSI

At this point we have a Logical Volume that is mostly empty and with a specific mount point (e.g. /lvmroot), we will need this mount point for the next exercise

There are some software solutions for creating an iSCSI Target on Linux, an easy to use one is “Target CLI”.

A screenshot of a computer program

Description automatically generated*TargetCLI in action*

As can be seen in the screenshot above, the specific CLI (Command Line Interface) of targetcli uses a tree-like structure to group certain functionality and commands.

**Q: Can you detect the created target in the screenshot above? How many targets are configured? How many LUNs per target?**

**1 target, and 4 LUNs as far as I can understand.**

**A screenshot of a computer screen

Description automatically generated**

There is a frontend and backend. Backend is where the block devices reside (in our case the LV). And the frontend is where the portals, ACLs & LUNs reside.

🡪 Hint: Tab Completion works fine in this CLI and there is “help” (e.g. *help cd*)

1. First install TargetCLI by running *apt install -y targetcli\**
2. And then starting it by executing: *targetcli*   
   The first time you will get errors about non-existing config files. However, these files are being created instantly and you won’t get these errors next time.
3. Let’s import the backend, type *cd backstores/fileio*
   1. *create file\_or\_dev=<LV path>/iscsidev name=file0 size=50G*

*Should have specified to use the mounted point, which is /lvmroot. Otherwise it dose not work. So create file\_or\_dev=/lvmroot/iscsidev name=file0 size=50G*

*A screen shot of a computer

Description automatically generated*

* 1. You can always verify your actions by running ‘*ls*’ (or *ls /* to see everything)
  2. **Q: What does “write-back deactivated” mean?**

**It means that the data is first written to the cache, and then to the main storage.**

1. Now let’s create a target, first move in the frontend:
   1. Type *cd /iscsi* followed by the command *create* to make get a new IQN

*A screen shot of a computer

Description automatically generated*

* 1. Verify again by using *ls* to see the IQN details

*A screen shot of a computer

Description automatically generated*

1. Now let’s configure our LUNs: *cd <iqn>/tpg1/luns*
   1. *create storage\_object=/backstores/fileio/file0 lun=0*

*A screen shot of a computer

Description automatically generated*(It is extremely important that there is a LUN ID of 0)

1. And finally configure the access: *cd ../acls*
   1. Within the acl subnode add the IQN of the iSCSI Initiator that is allowed to access the LUN. Use the one from your Windows iSCSI Initiator. (hint: type ‘help’)

*A screenshot of a computer program

Description automatically generated*

*create <IQN From iSCSI initiator>*

* 1. **Q: what does ACL stand for?**

**Access Control List**

1. **Finally save the configuration: *cd /* followed by *saveconfig***

Eventually it should be possible to connect to the iSCSI target, format it and add some files to it.

A screenshot of a computer

Description automatically generated

Optionally:  
🡪 Create some files on the iSCSI target and then find a second (Windows) VM, boot it, add it’s IQN to the targetcli software and **also connect** to the same LUN. Move some files and see how NTFS behaves when multiple devices have block level access to the same device …

I am a bit exhausted by the lab so I will leave it as it is. But I am sure it is going to work just fine with more than one machine.