# Solution M1: Local Storage and Additional Techniques (Red Hat)

## Task 1

**Challenge:** Create a RAID10-based pool in ZFS out six devices each 5 GB in size. Do it in two different configurations – 3x2 and 2x3

**Solution 1 (RAID 10 with 3x2):**

We assume that all required ZFS packages are installed, and the spare hard disks are **sdb**, **sdc**, **sdd**, **sde**, **sdf**, and **sdg**

The actual pool is created with:

**sudo zpool create zfs-raid10-3x2 mirror /dev/sd{b,c} mirror /dev/sd{d,e} mirror /dev/sd{f,g}**

The above will create the pool and mount it to **/zfs-raid10-3x2**

Should we want it mount to a different location, say **/storage**, we must change the above to this

**sudo zpool create -m /storage zfs-raid10-3x2 mirror /dev/sd{b,c} mirror /dev/sd{d,e} mirror /dev/sd{f,g}**

Then, we can check the result with

**zpool list**

Or with

**zpool status**

**Solution 2 (RAID 10 with 2x3):**

We assume that all required ZFS packages are installed, and the spare hard disks are **sdb**, **sdc**, **sdd**, **sde**, **sdf**, and **sdg**

The actual pool is created with:

**sudo zpool create zfs-raid10-2x3 mirror /dev/sd{b,c,d} mirror /dev/sd{e,f,g}**

The above will create the pool and mount it to **/zfs-raid10-2x3**

Should we want it mount to a different location, say **/storage**, we must change the above to this

**sudo zpool create -m /storage zfs-raid10-2x3 mirror /dev/sd{b,c,d} mirror /dev/sd{e,f,g}**

Then, we can check the result with

**zpool list**

Or with

**zpool status**

## Task 2

**Challenge:** Create a RAID6-based pool in ZFS out five devices each 5 GB in size

**Solution:**

We assume that all required ZFS packages are installed, and the spare hard disks are **sdb**, **sdc**, **sdd**, **sde**, and **sdf**

The actual pool is created with:

**sudo zpool create zfs-raid6 raidz2 /dev/sd{b,c,d,e,f}**

The above will create the pool and mount it to **/zfs-raid6**

Should we want it mount to a different location, say **/storage**, we must change the above to this

**sudo zpool create -m /storage zfs-raid6 raidz2 /dev/sd{b,c,d,e,f}**

Then, we can check the result with

**zpool list**

Or with

**zpool status**

## Task 3

**Challenge:** Research on how to use a key file to automount an encrypted volume on boot and demonstrate it for one drive (**sdb**)

**Solution:**

We assume that all required packages are installed, and the appropriate modules are loaded

Create a partition on the spare drive

**sudo parted -s /dev/sdb -- mklabel msdos mkpart primary 2048s -0m set 1**

Initiate the encryption procedure with

**sudo cryptsetup -y luksFormat /dev/sdb1**

Answer with **YES**

Then enter and re-enter encryption passphrase (for example **Parolka1**)

Now, let’s open it and create a file system

**sudo cryptsetup luksOpen /dev/sdb1 encr**

Let’s create a filesystem on it, for example XFS

**sudo mkfs.xfs /dev/mapper/encr**

Once we are done, we can close the partition

**sudo cryptsetup luksClose encr**

We are done with the initial setup. Before we continue, let’s dump the header information of our encrypted device

**sudo cryptsetup luksDump /dev/sdb1**

Notice the number of key slots used – one, with ID of 0

Now, let’s prepare a file with random data which will be used as a key

**sudo dd if=/dev/random of=/root/crypt.key bs=1024 count=2**

With the above, we created a **2048-byte** file named **crypt.key** which contains **random** data and stored it under the **root’s** home folder

We can further restrict the access to the file by changing its permissions

**sudo chmod 0400 /root/crypt.key**

Of course, we could have named the file differently (incl. to make it hidden), store it in another location, or create it shorter or longer

Now, that we have a key, we must add it to the encrypted device with

**sudo cryptsetup luksAddKey /dev/sdb1 /root/crypt.key**

We will be required to enter the existing pass phrase

Dump again the header of the encrypted device

**sudo cryptsetup luksDump /dev/sdb1**

Notice the number of key slots used – two (our key is on slot #1)

Try to open it using the key

**sudo cryptsetup luksOpen /dev/sdb1 encr --key-file /root/crypt.key**

We are ready to create a map in the **/etc/crypttab** file

Add the following record

**encr /dev/sdb1 /root/crypt.key luks**

Save and close the file

Now, open **/etc/fstab** file and add the following record

**/dev/mapper/encr /mnt/encr xfs defaults 0 0**

Save and close the file (of course, the mount point should exist)

Before rebooting the system, test that the changes made in the **/etc/fstab** file are acceptable and working

**sudo mount -a**

If everything went fine, it is safe to reboot the system. The drive should be auto opened and mounted