Random stuff linux systemd mount

This is another short video in the random linux stuff playlist and this one is on mounting filesystems using systemd.

As you undoubtedly know, to mount file systems at boottime we typically use the /etc/fstab file, or at least most of us have been doing so for a very long time. And there's nothing wrong with that except that sometimes it might lead to some problems if for example a file system or disk is no longer present in the system, but there is still an entry in the fstab file or if there is a error in the fstab syntax….your system will not boot properly anymore.

To avoid this potential problem we can now use systemd to mount filesystems at boottime.

This can be realized by creating a new mount service in which we specify some of the arguments we would also use in the fstab file, but now the system will not have any problems booting if something changes in the configuration…in the case of using systemd the file system will simply not be mounted and cause no further harm

Basically we only need one small UNIT file which reflects the name of the mountpoint. This is an important feature. If the name of the unit file does not reflect the mountpoint, you will get an appropriate error and the mount will fail.

So what we'll do is create a simple ext2 filesystem on a tiny disk, create a mountpoint in /mnt and we'll create the unit file in /lib/systemd/system, enable and start the service.

And I apologize that I am logged in as root.

So we run lsblk -f to list our block devices

- lsblk -f

And we see we've got some available disks…and we decide to use sdb so we create a filesystem on that disk using mkfs which will by default create an ext2 file system.

- mkfs /dev/sdb

When we run lsblk once more we see that it is of the filesystem type ext2

- lsblk -f

Now to avoid problems with regards to unpredicted device name changes we could use the UUID to mount the file system, but let's use a Label instead…so we create a LABEL in the file system.

So we run e2label like this….and of course the label can be anything.

- e2label /dev/sdb SMALL

Next we create a mountpoint, check the mount and unmount it again.

So we create /mnt/small and mount the filesystem

- mkdir /mnt/small
- mount /dev/sdb /mnt/small
- df -h

And it's mounted, so let's unmount it again.

- umount /mnt/small

Now let's create the unit file….
The quickest way to do that maybe is by using an existing mount service.
So we list them and pick the tmp mount service…

- cd /lib/systemd/system
- ls *mount
- cp tmp.mount mnt-small.mount

Again…the name of the unitfile should reflect the mount point….

No let's edit the content of the file….

- vi mnt-small.mount

Now we do want a description in:wq! which we enter the mount target.
Then we get rid of all the other stuff and want to mount it with as part of the multi-user target

Now here comes the interesting stuff…

In the MOUNT part
We need to define what we want to mount which would be the device with the files system that contains our label…then we need give it the mountpoint we created and the filesystemtype which would be ext2 and finally the options…let's give it the default options… filesystemtype and the options…so instead of putting this information in the fstab file we put it in this.

- What=/dev/disk/by-label/SMALL
- Where=/mnt/small
- Type=ext2
- Options=defaults

And the last line will also have the multi-user target…

-

So we save the file and check our mounts….

- df -h

Our file system is not mounted yet….

So we enable the service and check it again…

- systemctl enable mnt-small.mount --now
- df -h

And it’s mounted. And when we stop the service we see that it is automatically unmounted.

- systemctl stop mnt-small
- df -h

And to test whether it also works when we reboot, we reboot

- reboot

And we login and check and it’s what we had expected….