Linux: Advanced SoHo- Server Configuration

Welcome to my **Advanced Server Setup- Documentation**.

In these chapters, i will explain how to setup and configure a full featured Active Domain- Network with Kerberos Single-Sign-On and Domain Integration of Linux Clients on a rootless containerized Docker- Installation including Nextcloud as personal Cloud to store all your Data and PIM locally and safe. That way you get a fully managed, Cloud enabled Homeoffice Network at low costs and much space for your personal data on your own pc.

Current State

This Document is currently under developement and chapters are not final right now. This will change in the Future.

Usecase

This is not a slim Setup - so if you only have old hardware or you are trying to figure out on yoru small office-pc, this may not work as well as you need it.

You should have at least

- Large Harddrives: If you have maybe 1.5 TB of Data all togehter, you will need:
  - 3 TB of space on your working directory / raid5 = 3 Harddrives, each 1 TB at least
  - 6 TB of space on your backup / raid5 = 3 Harddrives, each 2 TB at least
  - about maybe 100GB for the system / raid1 = 2 Harddrives
  - about maybe 100GB for the databases / raid1 = 2 Harddrives
  - maybe two extra drives for external backups, each 6 TB (you can also store that in the internet, but you will need a large space there too)
- A Server, that has relyable, quite fast internet in Download and Upload rates - while Upload may be more Importen
- The Server should be reachable all the time

How to Start

First, read this Page, get the Hardware and install the system. You should understand the Hardwaresetup and the installation of Linux and Raid- Systems first (as decribend beneath).

Then, go on whith DynDNS- Setup to make your PC reachable from the net.

Next, setup docker as decribed in the Chapter. When you have portainer running, you can go like this:

1. Nextcloud-AIO
2. FreeIPA
3. Authentik

Then glue them together with SSO, SPNEGO and Nextcloud-SSO. Then you should have understood everything, you can now play around on your own.

Subpages

Basic System

As Hardware, you should have at least:

- a single standard Desktop- PC with 4 or more Cores
- equipped with at least 16 GB of RAM and
- for failure of Discs a swappable mounting Rack to contain at least 5 Discs (should not have Raid as Hardware, as Software Raid in Linux is much more efficient!)
- Additional at least one external Disk, you may use to copy your Backups to and store them on a different physikal location

Mountpoints

By default openSuSE will set some conservative mountoptions, that are save, but not best choice for homeoffice use and maybe could also improve company servers. Here are some proposals to think about.

Basically i would recommend to use UEFI only in Bios and GPT- Partitiontable on at least two Harddrives. The Linux- Root- System AND the EFI- Partitions should be mirrored (raid1) for failsafe and mak it possible to have the system booting from ONE disk (which is not possible with raid5).

The Data (like Home and program data) can have raid5 with 3 or more disks.

Always use LVM, as this has many benefits. On OpenSuSE btrfs is the best Filesystem if you disable Quotas on datapartitions.

Example-Setup

My small Homeoffice-Server described here, will have 5 Disks:

- 2x SSD with 2 TB each
- 3x HDD with 4 TB each

My Setup will look like this:

The SSDs will bothe have the same Layout:

- 1x 1GB Raid1 FAT32 EFIBOOT
- 1x 100%FREE LVM2 PV in Volumegroup vgssd
  - 100GB Raid1 lvroot btrfs,compress=zstd:3 root
  - 50GB Raid1 lvmariadb xfs for docker service mariadb
  - Space left blank for other high performance- services or growth

The HDDs will have:

- 1x 100%Free LVM2 PV Volumegroup vgdata
  - 1x 100GB Raid5 xfs, home and docker-service
  - 1x 4,4TB Raid5 lvbackup btrfs,compress=zstd:7 for internal daily Backup

Raided EFI-BOOT

Nowadays, UEFI is always the best choice to boot. UEFI- Boot is quite straight forward: You first take some device, make it gpt- partitioned, create a partition (i would at least take 500 MB today, better 1GB in size), format that partition with FAT32 and mark the partition as efi-boot via the partition flag. Thats usually all for a small office system. After some OS installed to that partition in a UEFI- way, the bios can load those files and start the OS.

But: Unfortunatelly, the designers of UEFI forgot, that if your not using hardware- raid (which i don't recommend, as your losing the ability to switch harddisks between your hardware), there is no standard way to raid the partition as FAT32 is not suitable for that while it would overwrite the parts in the partition, that are needed by MD Raid1 to store its metadata.

Fortunatelly the designers of OSS software- raid were smarter: They found a way to work around that: They made a special Version of MD Metadata called V1.0 which will store its Metadata at the end of the partition - so it will not interfere with FAT32. For FAT32 it can work as usual and for MD-Tools it will be able to detect the devices as Raid1.

But still - LVM will not work in this case. MD Partitions and Raid1 need to be outside of the LVM-Partition.

So I would suggest to use two disks both partioned with GPT and same sized efi-partitions (as said, at least 500 Megabytes in Size to store Bios or UCODE updates for Firmware Updater) and before creating the FAT32 filesystem do software raid on it. E.g.:

```
~ # mdadm --create --verbose /dev/md/efiboot --level=1 --raid-devices=2 --metadata=1.0 --name=efiboot --homehost=system
/dev/sda1 /dev/sdb1
```

The important part is metadata=1.0 - this format has especially designed to fit the needs of raid1 of fat/efi- systems.

You than install your Linux Bootmanager / EFIBOOT to that md- Device. If its not found in the beginning of the installation, scan for raid- devices or just create it while installing with the line above.

Recover faulty Disc

If some Raid- Disc becomes faulty, it will show up like this (its for raid5, but raid1 will look alkie):

```
obel1x:~ # mdadm -D /dev/md126
/dev/md126:
          Version : 1.0
    Creation Time : Fri Apr 10 11:44:19 2020
       Raid Level : raid5
       Array Size : 1460286976 (1392.64 GiB 1495.33 GB)
    Used Dev Size : 730143488 (696.32 GiB 747.67 GB)
     Raid Devices : 3
    Total Devices : 2
      Persistence : Superblock is persistent

    Intent Bitmap : Internal

      Update Time : Sat Oct 26 14:26:37 2024
            State : clean, degraded
   Active Devices : 2
  Working Devices : 2
   Failed Devices : 0
    Spare Devices : 0

           Layout : left-symmetric
       Chunk Size : 128K

Consistency Policy : bitmap

             Name : any:slowstorage
             UUID : 6542dc7c:a8f93b36:15f90ca1:54d03417
           Events : 285411

   Number   Major   Minor   RaidDevice State
      0       8        5        0      active sync   /dev/sda5
      1       8       21        1      active sync   /dev/sdb5
      -       0        0        2      removed
```

Maybe instead of removed you can see some entry like faulty instead of removed - this is, when the array had just failed.

To add a new device, you need an empty partiotion with at least the expected size (here 696 GB would be enough):

```
obel1x:~ # fdisk -l /dev/sdc
Disk /dev/sdc: 698.64 GiB, 750156374016 bytes, 1465149168 sectors
Disk model: WDC WD7500AAVS-0
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: gpt
Disk identifier: 699DC7F4-D344-4447-8C5B-1F98E017A12B

Device     Start        End    Sectors   Size Type
/dev/sdc1   2048 1465149134 1465147087 698.6G Linux RAID
```

That Partition should have the Type Linx Raid. If you don't have that, create it with partition- tool of kde or what you want.

Now you can simply add the device to the raid and it will begin to work:

```
obel1x:~ # mdadm /dev/md126 --add /dev/sdc1
mdadm: re-added /dev/sdc1

obel1x:~ # mdadm -D /dev/md126
/dev/md126:
          Version : 1.0
    Creation Time : Fri Apr 10 11:44:19 2020
       Raid Level : raid5
       Array Size : 1460286976 (1392.64 GiB 1495.33 GB)
    Used Dev Size : 730143488 (696.32 GiB 747.67 GB)
     Raid Devices : 3
    Total Devices : 3
      Persistence : Superblock is persistent

    Intent Bitmap : Internal

      Update Time : Sat Oct 26 14:34:57 2024
            State : clean, degraded, recovering
   Active Devices : 2
  Working Devices : 3
   Failed Devices : 0
```

```
    Spare Devices : 1

           Layout : left-symmetric
       Chunk Size : 128K

Consistency Policy : bitmap

   Rebuild Status : 1% complete

             Name : any:slowstorage
             UUID : 6542dc7c:a8f93b36:15f90ca1:54d03417
           Events : 285497

   Number   Major   Minor   RaidDevice State
      0       8        5        0      active sync   /dev/sda5
      1       8       21        1      active sync   /dev/sdb5
      3       8       33        2      spare rebuilding   /dev/sdc1
```

LVM

LVM is a powerful partition-management-layer and should always be used, when there is some none low-end hardware present. If you can use the **KDE Partitioning-Tool** (which means having Plasma=KDE Desktop compatible support), the support is very inuitive and opens a lot of flexibility whne handling partitions, like adding more disk space or moving partitions, but also on console this offers good functionality. OpenSuSE offer to create LVM- Styled system setup in installation optionally (not by default). If you can: use it.

Mirror- Raided LVM- Volumes (RAID1)

Noadays, MD raid1 or raid5 for system without LVM is outdated. Those things are integrated in LVM - so use it!

For our Setup we want to have the Linux Base System on Raid1, because Raid1 offers the flexibility to only have one phisical device that will work for its own without configuring. If you want to have the system on one disk, this is really nice.

So first, create a partition on both disks which is maximum in Size. Than, create a Volume group containing those partitions and finally, create a raid1 on it (for example):

```
vgcreate vgsystem /dev/sdX1 /dev/sdY1
lvcreate -m1 --type raid1 -L 100GB -n lvroot vgsystem
```

where 100%FREE means 100% of Free Space used...

To check if raid1 works, use:

```
lvs -a -o name,copy_percent,devices vg_xxx
```

If this has not worked, use:

```
lvconvert --type raid1 -m1 vg_xxx/lvol1
```

Or - you can do raid5 with:

```
lvcreate -n lvdata --type raid5 -l 100%FREE -i 2 vgdata
```

where i equals the number of devices with Data (not including parity- storage)

Useful Commands

The KDE- Partitionmanager is still not perfect. LVM is mor powerful in these things:

Moving logical Volumes to physical Devices

Usually Partitions or Devices are only assigned to Volume-Groups (VG) and Logical Volumes (LV) are using them dynamically as needed. This makes it sometimes hard to understand, where the Data really is located right now. Especially when you are having different physical Devices, you may want one LV to use one Device.

For an overview how the Data is split, you can use:

```
# lvs -P -a -o +devices,segtype
  LV                VG       Attr       LSize    Pool Origin Data%  Meta%  Move Log Cpy%Sync Convert Devices
Type
  lvbackup          vgdata   rwi-aor---    4.40t                                    100.00
```

```
lvbackup_rimage_0(0),lvbackup_rimage_1(0),lvbackup_rimage_2(0) raid5
  [lvbackup_rimage_0] vgdata   iwi-aor---    2.20t                                                    /dev/sde1(377061)
linear
  [lvbackup_rimage_1] vgdata   iwi-aor---    2.20t                                                    /dev/sda1(377061)
linear
  [lvbackup_rimage_2] vgdata   iwi-aor---    2.20t                                                    /dev/sdd1(377061)
linear
  [lvbackup_rmeta_0]  vgdata   ewi-aor---    4.00m                                                    /dev/sde1(377060)
linear
  [lvbackup_rmeta_1]  vgdata   ewi-aor---    4.00m                                                    /dev/sda1(377060)
linear
  [lvbackup_rmeta_2]  vgdata   ewi-aor---    4.00m                                                    /dev/sdd1(377060)
linear
  lvdata              vgdata   rwi-aor--- 1007.30g                                      100.00
lvdata_rimage_0(0),lvdata_rimage_1(0),lvdata_rimage_2(0)       raid5
  [lvdata_rimage_0]   vgdata   iwi-aor---  503.65g                                                    /dev/sde1(1)
linear
  [lvdata_rimage_1]   vgdata   iwi-aor---  503.65g                                                    /dev/sda1(1)
linear
  [lvdata_rimage_2]   vgdata   iwi-aor---  503.65g                                                    /dev/sdd1(1)
linear
  [lvdata_rmeta_0]    vgdata   ewi-aor---    4.00m                                                    /dev/sde1(0)
linear
  [lvdata_rmeta_1]    vgdata   ewi-aor---    4.00m                                                    /dev/sda1(0)
linear
  [lvdata_rmeta_2]    vgdata   ewi-aor---    4.00m                                                    /dev/sdd1(0)
linear
  lvdocker            vgdata   rwi-aor---    1.89t                                      100.00
lvdocker_rimage_0(0),lvdocker_rimage_1(0),lvdocker_rimage_2(0) raid5
  [lvdocker_rimage_0] vgdata   iwi-aor---  969.23g                                                    /dev/sde1(128936)
linear
  [lvdocker_rimage_1] vgdata   iwi-aor---  969.23g                                                    /dev/sda1(128936)
linear
  [lvdocker_rimage_2] vgdata   iwi-aor---  969.23g                                                    /dev/sdd1(128936)
linear
  [lvdocker_rmeta_0]  vgdata   ewi-aor---    4.00m                                                    /dev/sde1(128935)
linear
  [lvdocker_rmeta_1]  vgdata   ewi-aor---    4.00m                                                    /dev/sda1(128935)
```

```
linear
  [lvdocker_rmeta_2]  vgdata   ewi-aor---    4.00m                                                    /dev/sdd1(128935)
linear
  lvhome              vgsystem rwi-aor---   94.93g                                 100.00
lvhome_rimage_0(0),lvhome_rimage_1(0)                          raid1
  [lvhome_rimage_0]   vgsystem iwi-aor---   94.93g                                                    /dev/sdc2(166910)
linear
  [lvhome_rimage_1]   vgsystem iwi-aor---   94.93g                                                    /dev/sdb2(166910)
linear
  [lvhome_rmeta_0]    vgsystem ewi-aor---    4.00m                                                    /dev/sdc2(166909)
linear
  [lvhome_rmeta_1]    vgsystem ewi-aor---    4.00m                                                    /dev/sdb2(166909)
linear
  lvroot              vgsystem rwi-aor---   97.52g                                 100.00
lvroot_rimage_0(0),lvroot_rimage_1(0)                          raid1
  [lvroot_rimage_0]   vgsystem iwi-aor---   97.52g                                                    /dev/sdc2(1)
linear
  [lvroot_rimage_1]   vgsystem iwi-aor---   97.52g                                                    /dev/sdb2(1)
linear
  [lvroot_rmeta_0]    vgsystem ewi-aor---    4.00m                                                    /dev/sdc2(0)
linear
  [lvroot_rmeta_1]    vgsystem ewi-aor---    4.00m                                                    /dev/sdb2(0)
linear
```

You can also move them to single Devices if needed. Here i wanted my home to also be on the faster Device sda. As sda4 had enough free space, i could do:

```
# pvmove -n system/home /dev/sdb2 /dev/sda4
```

Resizing logical Volumes with mounted Filesystem

can be done by e.g.

```
lvresize --size 20G /dev/vgfast/lvfast --resizefs
```

LVM Error Recovery

In case on Harddrive is failing, the Array gets degraded. If you boot your system without that disk, it will not start due to inaktive volume groups.

To recover, do this:

1. Get Volume Groups up, if degraded

vgchange -a y

2. Add a new PV to the VG that is large enough to hold the Data

vgextend vgname /dev/sdX

3. Repair the logical volume by searching for usable PVs automagically

lvconvert –repair vgname/lvname

This should rebuild your logical Volume

4. After rebuild, remove the faild drive from the vg:

vgreduce –removemissing vgdata

Thats it, your System should become usable after that.

Moving Data before Drive fails

If you have the possibility to add a new PV before the array gets degraded, you can use the replace- method after adding the new pv to the VG:

lvconvert –replace /dev/sdX1 vgname/lvname /dev/sdY1

More Info for LVMs

https://docs.redhat.com/en/documentation/red_hat_enterprise_linux/8/html/configuring_and_managing_logical_volumes/configuring-raid-logical-volumes_configuring-and-managing-logical-volumes

Filesystem

Brtfs is the way to go everywhere where you need big data and flexibility. There are some disadvanteges while it is still in developement and sometimes it is a bit oversized for homeoffice, but no other filesystem is that good in general usage. Only use other Filesystems, if there are reasons for - e.g. when exchanging files with another windows on that pc.

And there is one Reason: Docker - at the current time of writing this (20.04.2024) you should NOT USE BTRFS with Docker. More is explained later.

Mountoptions

BTRFs has a lot of Mountoptions and some here are really odd ones for every linux. I would suggest at least those:

For **Desktopusage**: compress=zstd:3,noatime,nodiratime,autodefrag

While autodefrag should not be necessary on ssd- harddiscs.

For **Databases** or files that need speed and **are well backed up otherwise** : nodatacow,nodatasum,noatime,nodiratime

Sources:

- https://fedoraproject.org/wiki/Changes/BtrfsTransparentCompression
- https://btrfs.readthedocs.io/en/latest/Administration.html

Powermode settings

Your Harddrives may have set a power level, that allows spindown. I personally would not let your harddrives spindown, because every start brings your harddrives mechanics nearer to death. In Fact there is not very much worse than spinning up and down every few minutes for a harddrive with physical discs.

To change that, create the following file:

```
pcserver2023:/usr/lib/udev/rules.d # cat 85-hdparm.rules
ACTION=="add|change", SUBSYSTEM=="block", KERNEL=="[sh]d[a-z]", RUN+="/sbin/hdparm -S 0 -B 128 /dev/%k"
```

That way, your harddrives will stay up all time.

Quotas

Brief: Quotas should be disabled

What is it for?

BTRFS comes with included support of disk quotas. It is enabled by default if a btrfs-volume is created. Disk quotas are useful to mangage diskspace, as they store informations of directory sizes in the filesystem. There may be programs, that calculate disk usage of directories only that way. One example is snapper: It will automatically delete old snapshots, if the space is running low on a device using snapshots. This is done, by checking the qoutas. Running snapper without qoutas will make this not working anymore. Instead other functions will be used - e.g. the maximum number of old snapshots to keep.

To find out, if quoutas are enabled, do:

```
btrfs qgroup show /
```

What is the problem?

Quotas are complicated to manage by btrfs. As there are many situations, where the qoutas may get incorrect, they will often be invalidated and will need to be recalculated from scratch. Furthermore checking if they are correct is often needed - e.g. at startup or after some time. This process consumes a lot of cpu and disc utilisation and makes the hardware slow, sometimes rendering a computer useless for some time.

Solution

Because of this, the kernel.org- team recommends to turn disk quotas off if not needed.

This can be done by:

```
btrfs quota disable /
```

Snapper

After that, /etc/snapper/configs/root should be checked:

```
# limit for number cleanup
NUMBER_MIN_AGE="1800"
NUMBER_LIMIT="10"
NUMBER_LIMIT_IMPORTANT="10"
```

Swappiness

If you have a lot of Ram, you may adjust the swappiness to better fit your needs - or turn off swap completely.

e.g. /etc/sysctl.conf:

```
vm.swappiness = 60
```

Filesystem and User rights in Linux

While linux itself is a very secure system (when set up the right way), the rights given to files by default are not secure at all. Setting good rights is not an intuive process and is mostly not well done. So it needs some attention.

Why care about rights

If you are the only user on your pc, or your linux pc is a machine for the net, then you maybe fine. But if your pc is shared between some users e.g. in your family and used by some other persons, then you may wish, that personal informations of the one user is not accessible to the other one.

Even if your pc is not used by others, there maybe other users on your pc by services running in different accounts, so maybe you want Data not to be visible to any user.

Test the rights

Lets start with a simple new file, let's say „~\securedata.txt“ with text „secure data“ in it. Let say we are user named testuser:

New File ~\securedata.txt :

```
testuser@xubuntu-stick:~$ echo secure data> ~/securedata.txt
```

Then become someone else, maybe user testuser2 and read that file:

```
testuser@xubuntu-stick:~$ su -l testuser2
Passwort:
testuser2@xubuntu-stick:~$
```

Now lets see what the other user has written in its secure file:

```
testuser2@xubuntu-stick:~$ cat ../testuser/securedata.txt
cat: ../testuser/securedata.txt: Keine Berechtigung
```

So everything right? No - not quite. Lets say horst decides to store is secure data in another location and do it again:

```
testuser@xubuntu-stick:~$ mv securedata.txt /tmp
testuser@xubuntu-stick:~$ su -l testuser2
Passwort:
testuser2@xubuntu-stick:~$ cat /tmp/securedata.txt
secure data
```

So now the data the one user created is not secure any more. Well ok this sounds fine, because the data was moved to an insecure directory.

But: Can you make sure that everybody knows which directories are secure and which aren't? Can you be really sure, that no personal information saved by any user-context-program is written only to secure directories and not visible to the other? I guess not.

So it may be a better approach to not make the files created by someone readable by other user by default.

UMask- Approach

There ist a tool for this called umask. This tool defines the permission for new created files.

By default the umask is 0002 or 0022. Those values are substracted from 0777, which would mean full access for everyone. You can check out the docs in the net how they work. I won't explain here, cause there is a big problem with umask: The value can only be changed on process level or user or systemwide. This means you cannot set them per directory - which would be intentional to the user.

So you should maybe think of setting a better umask than 022 - which would make all users of you group have read access to you files to lets say 077. Or - even better don't use the group „users“, but make a group with the same name as the user per User itself. Than you can have umask 007.

On my system the umask can be defined in the file `/etc/login.defs` .

But to go on directory- permissions: forget about umask.

FACLs

F… what??? Yes: facl is the tool to do so. with that tool you can very much expand the rights per directory an on every file in detail. It ist also possible to have multiple group- access definitions, which are not possible otherwise.

So lets do some facl- work

FACL: get infos about settings

```
testuser@xubuntu-stick:~$ mkdir temp
testuser@xubuntu-stick:~$ getfacl temp
# file: temp
# owner: testuser
# group: testuser
user::rwx
group::rwx
other::r-x
```

As you can see, that directory has been created quite insecure. There is only the above permission preventing everyone to read the informations in it. Creating a new file in it, would make it the same way insecure, as it would have been before.

FACL: set default permissions

But now lets set the mode to better fit our needs:

```
testuser@xubuntu-stick:~$ setfacl -m d:o::--- temp
testuser@xubuntu-stick:~$ getfacl temp
# file: temp
# owner: testuser
# group: testuser
user::rwx
group::rwx
other::r-x
default:user::rwx
default:group::rwx
default:other::---
```

Note, that we only changed the DEFAULT permissions to be more secure (d:).

FACL: check new settings

Now lets again create a file there as we did before just in that - safe - directory. Also we can use getfacl on that file to check if it works:

```
testuser@xubuntu-stick:~$ echo secure data> ~/temp/securedata.txt
testuser@xubuntu-stick:~$ getfacl temp/securedata.txt
# file: temp/securedata.txt
# owner: testuser
# group: testuser
user::rw-
group::rw-
other::---
```

As you can see, that file is more secure than before. So lets check, what happens now if we move that file as before.

```
testuser@xubuntu-stick:~$ rm /tmp/securedata.txt
testuser@xubuntu-stick:~$ mv temp/securedata.txt /tmp
```

And check if it is accessible by another one:

```
testuser@xubuntu-stick:~$ su -l testuser2
Passwort:
testuser2@xubuntu-stick:~$ cat /tmp/securedata.txt
cat: /tmp/securedata.txt: Keine Berechtigung
```

You can also check the rights now:

```
testuser2@xubuntu-stick:~$ getfacl /tmp/securedata.txt
getfacl: Entferne führende '/' von absoluten Pfadnamen
# file: tmp/securedata.txt
# owner: testuser
# group: testuser
user::rw-
group::rw-
other::---
```

Now „Testuser2“ knows, that he has to ask „Testuser“ to tell him the secret he wrote in that file.

That way you can also have one or more default group(s) assigned and to give only those groups access to the file(s), which is very powerful. As for the user perspective i would rate that approach more secure, than the default one.

Its up to you to decide if this is more usable or not.

FACL: full ACL- Sytnax

The full Syntax is:

```
      [d[efault]:] [u[ser]:]uid [:perms]
             Permissions of a named user. Permissions of the file owner if uid is empty.

      [d[efault]:] g[roup]:gid [:perms]
             Permissions of a named group. Permissions of the owning group if gid is empty.
```

```
      [d[efault]:] m[ask][:] [:perms]
             Effective rights mask

      [d[efault]:] o[ther][:] [:perms]
             Permissions of others.
```

That means you can only set the defaults per user or per group and only files or directories at once.

FACL: use in batch and recursively

FACLs do also have good ways to be used for whole directories, chek out:

`setfacl` has a *recursive* option (`-R`) just like `chmod`:

-R, –recursive Apply operations to all files and directories recursively. This option cannot be mixed with `–restore'.

FACL: handle execute-bit with files and directories

...it also allows for the use of the capital-x X **permission**, which means: execute only if the file is a directory or already has execute permission for some user (X)

so doing the following should work:

Set all Files AND the directories recursively to be readwriteable by user colleague and only give X to all Directories and only those Files, that already have x set:

`setfacl -R -m u:colleague:rwX .`

For setting the default permissions to be like that:

`setfacl -R -m **d**:u:colleague:rwX .`

Last words

Some other intersting aspects:

- There are some interesting usages of the sticky bits for a. the user and group- bit and b. to files and directories in seperate

- Mind, that only the user of the file can change its ownership. Per default all files created by the user are owned by the user.
- That means: If you don't want a user be able to change the ownership of a file into insecure permissions, it may be a good way to set the default user of files in the directory to root and only allow group- access to it. That way directories can be read and written, but not the content will not be opened to everyone by some hacky user (or virus in the last mail of that user?)

All in all thinking about permissions is a basic one whenever there is personal data that needs to be secured somehow. One cannot rely on the defaults and hope its all fine.

And with FACLs there are powerful tools that should cover everything an administrator needs.

Firewall

To check, which services are open, use:

PLEASE, Before opening the Ports, check the Services described at the Sub-Pages first to secure them!

```
servername:~ # firewall-cmd --list-ports
3478/tcp 3478/udp
servername:~ # for s in $(firewall-cmd --list-services); do firewall-cmd --permanent --service "$s" --get-ports; done;
546/udp
53/tcp 53/udp
80/tcp
443/tcp
88/tcp 88/udp
389/tcp
636/tcp
873/tcp 873/udp
22/tcp
```