# Tips and tools

- Edit /etc/passwd correctly
- Create multiple parent directories with mkdir
- Make a Linux VM template unique

# Edit /etc/passwd correctly

In my recent guides, you might have noticed that I sometimes edit `/etc/passwd` directly with `vi` or `nano`. **This is not advised behavior.** The main reason being the possibility of file corruption and other users making changes to their account while you are editing. The latter is not very likely to happen as I am the only user on the system. In any case, here's how to do it more properly.

## Use vipw

vipw is a command line utility designed to make edits to the `/etc/passwd` file and prevents corruption by setting appropriate locks. You will need elevated privileges to edit `/etc/passwd`.

```
$ sudo vipw
```

It might ask you to pick an editor you want to use, usually `nano` or `vi`. Choose which one you prefer and the file will open, ready to be edited. I usually do this to edit shell for a service user, but do whatever you like.

# Create multiple parent directories with mkdir

You have a directory `/home/mydir` and you want to create two new directories like this: `/home/mydir/backup/something`

When you run `mkdir /home/mydir/backup/something` you will get an error:

```
mkdir: cannot create directory ‘/home/mydir/backup/something’: No such file or directory
```

Normally, you would have to make both directories separately like `mkdir /home/mydir/backup /home/mydir/backup/something`

Add the `-p` flag *(make parent directories along the way if needed)* to fix it:

```
mkdir -p /home/mydir/backup/something
```

> mkdir created both `backup` and `something` directory in the same path

# Make a Linux VM template unique

If you have ever worked with Virtual Machines (VMs), you are most likely familiar with the term "VM cloning" or VM templates in general. It's a process of taking an existing VM with the operating system installed and cloning it to create a new identical VM. This comes in handy when you are tasked with creating multiple VMs for the same purpose or you want to achieve consistency across your environment (same OS settings, FS layout etc.). All of this is possible without having to install X number of VMs manually - you can just do it once and then clone the VM in a few clicks.

The problem with this time saving feature is that each VM is ***really*** identical. This may not seem like a problem at a first glance - we wanted to make the VMs identical, right? Well, that's true, but there's certain things that should not be kept the same. One of them is e.g SSH host keys - without any further modification, you would have the same SSH host keys on all of your systems, which is certainly not a good thing for security.

There's a couple of ways we can solve this - most commonly through Guest OS Customization features of different hypervisors or with automation (cloud-init, ansible...). Most of the automation stuff requires quite a bit of preparation and takes time. Guest OS customization is a great option, if it's available, but doesn't solve everything either. Today, I'm going to show you a couple of things that you should definitely do on each cloned system to make it "unique". The list may not include everything ever, but has worked for me for quite some time.

> This example is based on Debian 12. The configuration may be different on other systems

## Change hostname

Use the ***hostnamectl*** to change the hostname. The command may not return the new hostname until reboot.

```
sudo hostnamectl set-hostname new_hostname.domain


hostnamectl
 Static hostname: new_hostname.domain
       Icon name: computer-vm
         Chassis: vm □□
      Machine ID: XXXXXXXXXXXXXXXXXXXXXXXXXX
         Boot ID: XXXXXXXXXXXXXXXXXXXXXXXXXX
```

```
  Virtualization: kvm
Operating System: Debian GNU/Linux 12 (bookworm)
          Kernel: Linux 6.1.0-22-amd64
    Architecture: x86-64
 Hardware Vendor: QEMU
  Hardware Model: Standard PC _i440FX + PIIX, 1996_
Firmware Version: rel-1.16.2-0-prebuilt.qemu.org
```

## Fix /etc/hosts

Sometimes people forget that the hostname also lives in the hosts file and won't change on its own.

```
vim /etc/hosts
127.0.0.1       localhost
192.168.0.20    new_hostname.domain       new_hostname

# The following lines are desirable for IPv6 capable hosts
::1     localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

## Edit IP

You don't want to cause an IP collision on your network, so don't forget to give each system a different IP and apply correct settings in the OS.

```
sudo vim /etc/network/interfaces
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
allow-hotplug ens18
iface ens18 inet static
        address 192.168.0.20/24
```

```
gateway 192.168.0.1
# dns-* options are implemented by the resolvconf package, if installed
dns-nameservers 8.8.8.8
dns-search domain
```

## Change ID

This is the "Machine ID" you saw when running the ***hostnamectl*** command. You only want to change the Machine ID, but not the Boot ID - that is usually assigned from the virtualization side and is already unique (happens during cloning)

```
sudo rm /etc/machine-id
sudo rm /var/lib/dbus/machine-id
sudo dbus-uuidgen --ensure=/etc/machine-id
sudo dbus-uuidgen --ensure
```

## Change SSH keys

I found that the easiest way to do this is actually just reinstalling the whole SSH deamon using purge (deletes all config). Or you can just not include SSH in the template and install it on each system later.

```
sudo apt purge openssh-server openssh-client
sudo apt install openssh-server openssh-client
```

> Perform a reboot after all the changes above