# Updates & Upgrades

# Upgrading Debian 10 (Buster) to Debian 11 (Bullseye)

I am a big fan of Debian in the server environment, mainly due to its great record in stability, security and other important aspects like having a huge number of tutorials and guides available online. Since the release of Debian 11 (codenamed Bullseye), I've been thinking about upgrading to the latest version. I will start on one of my VPS servers to test how is everything working. There are number of tutorials online explaining the exact same thing. Feel free to follow which you find the best, this is mainly for my documentation.

> Before you begin the upgrade process, make sure you know/have these two things:

> Standard Debian installation with the **default kernel** – in this case it is probably fine to update **with SSH access only**.

> However, if you are using a **different kernel** or **backports,** I recommend you also have **access to the boot process** and **recovery** in case your system doesn't boot after the upgrade.

## Summary

1. **Backup your system** – You never know what can go wrong, be prepared.
2. **Edit apt's *sources.list*** – In order to fetch and install packages meant for Debian 11, we need to change some lines in the */etc/apt/sources.list* file.
3. **Update software repos** – Make apt aware of the changes you've made in sources.list and upgrade existing packages.
4. **Upgrade the system itself** – After upgrading packages, you can upgrade the system as well.

## Backup your system

Unless you actually have a testing environment, where loss of files won't cause even a minimal headache, please backup your data. In most cases, it is OK to at least backup all configuration files for any services running on the server. You can always rebuild the server using them in case something bad happens. Always have a precise upgrade plan when upgrading production servers, including plans B and C and D, depenging on the criticality of the service you are running. I am upgrading a server with two services only I use, so I can afford a very simple "backup" – *cat* all config files into terminal and copy them to notepad on my workstation, that's it. You can also setup something fancier like rsnapshot or restic.

## Prepare for the changes

### Check system version

There are numerous ways (from basic to more fancy) to check what version and distribution you are running, try running couple of these:

```
lsb_release -a
```

If you don't have that installed like me, try on of the other options. Simply cat */etc/debian_version* or */etc/os-release.* The latter will give you more detailed information.

```
$ cat /etc/debian_version
10.10

$ cat /etc/os-release
PRETTY_NAME="Debian GNU/Linux 10 (buster)"
NAME="Debian GNU/Linux"
VERSION_ID="10"
VERSION="10 (buster)"
VERSION_CODENAME=buster
ID=debian
HOME_URL="https://www.debian.org/"
SUPPORT_URL="https://www.debian.org/support"
BUG_REPORT_URL="https://bugs.debian.org/"
```

You should also probably know about the *uname* command, combined with the -a (--all) prints some system information.

```
$ uname -a
Linux hostname 5.4.0-74-generic #83~18.04.1-Ubuntu SMP Tue May 11 16:01:00 UTC 2021 x86_64 GNU/Linux
```

If you run this command on a vanilla Debian 10 distribution, you will most likely see kernel version 4.19, which is the version Debian 10 shipped with. For some reason, my VPS provider is using a newer kernel and somehow managed to throw Ubuntu into the mix, even though the system clearly runs on Debian, based on the multiple command outputs above.

## Update and upgrade existing software

It is recommended before installing any new packages or performing a large update such as this one to update and upgrade the existing system.

First of all, update apt repositories. Unless you are using Nginx from their official repository instead from the Debian one, you won't see the lines containing *nginx*.

```
$ sudo apt update
Get:1 https://nginx.org/packages/mainline/debian buster InRelease [3,607 B]
Hit:2 http://deb.debian.org/debian buster InRelease
Hit:3 http://security.debian.org/debian-security buster/updates InRelease
Hit:4 http://deb.debian.org/debian buster-backports InRelease
Get:5 https://nginx.org/packages/mainline/debian buster/nginx amd64 Packages [49.9 kB]
Fetched 53.5 kB in 1s (47.0 kB/s)
Reading package lists... Done
Building dependency tree
Reading state information... Done
```

Upgrade existing packages with *apt upgrade*.

```
$ sudo apt upgrade
Reading package lists... Done
Building dependency tree
Reading state information... Done
Calculating upgrade... Done
The following packages will be upgraded:
  nginx
1 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
Need to get 880 kB of archives.
After this operation, 0 B of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 https://nginx.org/packages/mainline/debian buster/nginx amd64 nginx amd64 1.21.3-1~buster [880 kB]
Fetched 880 kB in 11s (83.5 kB/s)
debconf: delaying package configuration, since apt-utils is not installed
(Reading database ... 42829 files and directories currently installed.)
Preparing to unpack .../nginx_1.21.3-1~buster_amd64.deb ...
Unpacking nginx (1.21.3-1~buster) over (1.21.2-1~buster) ...
Setting up nginx (1.21.3-1~buster) ...
```

```
Processing triggers for systemd (241-7~deb10u8) ...
```

> As you can see I had a pending upgrade of Nginx from 1.21.2 to 1.21.3. In production, always check before upgrading individual packages in case there is a major change that might break your system.

```
$ sudo apt dist-upgrade
Reading package lists... Done
Building dependency tree
Reading state information... Done
Calculating upgrade... Done
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
```

If you want to know the difference between *apt upgrade* and *apt dist-upgrade*, read this paragraph from apt's man page:

"*dist-upgrade in addition to performing the function of upgrade, also intelligently handles changing dependencies with new versions of packages; apt-get has a "smart" conflict resolution system, and it will attempt to upgrade the most important packages at the expense of less important ones if necessary. So, dist-upgrade command may remove some packages. The /etc/apt/sources.list file contains a list of locations from which to retrieve desired package files. See also apt_preferences(5) for a mechanism for overriding the general settings for individual packages.*"

Clean any leftovers using the following commands:

```
$ sudo apt autoremove
Reading package lists... Done
Building dependency tree
Reading state information... Done
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.

$ sudo apt autoclean
Reading package lists... Done
Building dependency tree
Reading state information... Done
```

Again, to learn why we are running these commands, read this from the man page:

- ***clean**: clean clears out the local repository of retrieved package files. It removes everything but the lock file from /var/cache/apt/archives/ and /var/cache/apt/archives/partial/. When APT is used as a dselect(1) method, clean is run automatically. Those who do not use dselect will likely want to run apt-get clean from time* *to time to free up disk space.*

- ***autoclean**: Like clean, autoclean clears out the local repository of retrieved package files. The difference is that it only removes package files that can no longer be downloaded, and are largely useless. This allows a cache to be maintained over a long period without it growing out of control. The configuration option APT::Clean-Installed will prevent installed packages from being erased if it is set to off.*
- ***autoremove**: is used to remove packages that were automatically installed to satisfy dependencies for some package and that are no longer needed.*

## Edit & Update software repos

Before making any changes to the */etc/apt/sources.list* file, back it up in a different directory. Do the same for anything in the */etc/apt/sources.list.d* folder. This will copy the file to your home directory under the name *sources.list.bak*

```
$ cp /etc/apt/sources.list ~/sources.list.bak
```

Use your favorite editor or *sed* to replace all "**buster**" references with "**bullseye**", without quotes of course.

The problem is, the sources.list can be set up differently based on your needs. For example, this is *the sources.list* on my VPS now.

```
# Generated by distrobuilder
deb http://deb.debian.org/debian buster main
deb http://security.debian.org/debian-security buster/updates main
deb http://deb.debian.org/debian buster-backports main

# Official Nginx repo
deb https://nginx.org/packages/mainline/debian/ buster nginx
```

Meanwhile the full sources.list (with official Debian repos only) can look like this:

```
deb http://deb.debian.org/debian buster main contrib non-free
deb-src http://deb.debian.org/debian buster main contrib non-free

deb http://deb.debian.org/debian buster-updates main contrib non-free
deb-src http://deb.debian.org/debian buster-updates main contrib non-free

deb http://deb.debian.org/debian buster-backports main contrib non-free
deb-src http://deb.debian.org/debian buster-backports main contrib non-free
```

```
deb http://security.debian.org/debian-security/ buster/updates main contrib non-free

deb-src http://security.debian.org/debian-security/ buster/updates main contrib non-free
```

Let me explain what each of these mean. Feel free to skip this part or read it from the official source and continue the migration process.

- ***deb*** or ***deb-src*** indicate the type of archive. ***deb*** consists of binary (=already compiled) packages, while ***deb-src*** contains the source code and other necessary files for building applications from source. Unless you plan to build official packages from source on the system, you can completely leave out the ***deb-src*** lines.
- ***http://deb.debian.org/debian*** and ***http://security.debian.org/debian-security/*** are URLs that point to mirrors which contain the actual packages. There are hundreds of mirrors and it is generaly recommended to use the closest one to you with the smallest latency (you can find the list here). The two mentioned above are actually just pointers to CDN network which should redirect you to the fastest official mirror (as explained here).
- ***buster*** – this part refers to the codename of the distribution (e.g. buster, bullseye, etc.) The first refers to the *base* Debian repository, while the others to the *updates*, *backports* and *security* respectively. The backports repository is used in cases you want a stable system (use the stable branch), and have newer versions of software available through the official repository. However, you must be careful and don't mess with fundamental libraries and other core packages, which could result in a broken system due to mismatched versions of important packages.
- ***main, contrib*** and ***non-free*** – these are called *components* and specify which kinds of packages you would like to have access to. Some users won't need *non-free* packages, because they support the idea of *libre* software, while others might need them to install additional firmware to make their devices work properly.

Finally change the *sources.list* file accordingly. I won't be building from source and I don't need any packages outside of main. I will be adding new line with debian updates, which wasn't there for some reason before.

```
# Generated by distrobuilder
deb http://deb.debian.org/debian bullseye main


deb http://security.debian.org/debian-security bullseye-security main


deb http://deb.debian.org/debian bullseye-updates main


deb http://deb.debian.org/debian bullseye-backports main


# Official Nginx repo
deb https://nginx.org/packages/mainline/debian/ bullseye nginx
```

**Be careful, syntax of the security repository was changed from the previous release.** Instead of ***buster/updates***, it is now ***bullseye-security.***

Now run apt to make it aware of configuration changes

```
$ sudo apt update
Get:1 http://security.debian.org/debian-security bullseye-security InRelease [44.1 kB]
Get:2 http://deb.debian.org/debian bullseye InRelease [113 kB]
Get:3 http://deb.debian.org/debian bullseye-updates InRelease [36.8 kB]
Get:4 http://deb.debian.org/debian bullseye-backports InRelease [39.3 kB]
Get:5 http://security.debian.org/debian-security bullseye-security/main amd64 Packages [29.6 kB]
Get:6 http://security.debian.org/debian-security bullseye-security/main Translation-en [16.0 kB]
Get:7 http://deb.debian.org/debian bullseye/main amd64 Packages [8,178 kB]
Get:8 http://deb.debian.org/debian bullseye/main Translation-en [6,241 kB]
Get:9 http://deb.debian.org/debian bullseye-backports/main amd64 Packages [56.8 kB]
Get:10 http://deb.debian.org/debian bullseye-backports/main Translation-en [42.9 kB]
Get:11 https://nginx.org/packages/mainline/debian bullseye InRelease [2,860 B]
Get:12 https://nginx.org/packages/mainline/debian bullseye/nginx amd64 Packages [7,716 B]
Fetched 14.8 MB in 10s (1,415 kB/s)
Reading package lists... Done
Building dependency tree
Reading state information... Done
384 packages can be upgraded. Run 'apt list --upgradable' to see them.
```

## Upgrade the packages and distro

### Upgrade existing packages only

```
384 packages can be upgraded. Run 'apt list --upgradable' to see them.
```

As you can see, a lot of packages will be upgraded. In order to avoid major breaks, we need to run apt upgrade **--without-new-pkgs.** This will only upgrade existing packages, but won't remove or add any new. The reason we are doing this is to prevent the system from breaking down due to missing packages after the upgrade.

```
$ sudo apt upgrade --without-new-pkgs
```

You may see the following line somewhere in the output:

```
The following packages have been kept back:
```

This is the result of --without-new-pkgs. The reason for these *kept back* packages is the following:

"*If the dependencies have changed on one of the packages you have installed so that a new package must be installed to perform the upgrade then that will be listed as* "*kept-back*"."

> **Carefully watch the upgrade process!** In case a service needs to be **restarted** or the system **doesn't know what to do** with new configuration files (e.g. you have custom config file and the upgrade brings a new one, should it overwrite, keep original or let you compare and merge manually?) **You might be asked these kinds of questions.**

It is not strictly necessary to reboot now, but I just want to make sure nothing broke during this process and the system can safely boot.

## Perform full upgrade – remove old, install new

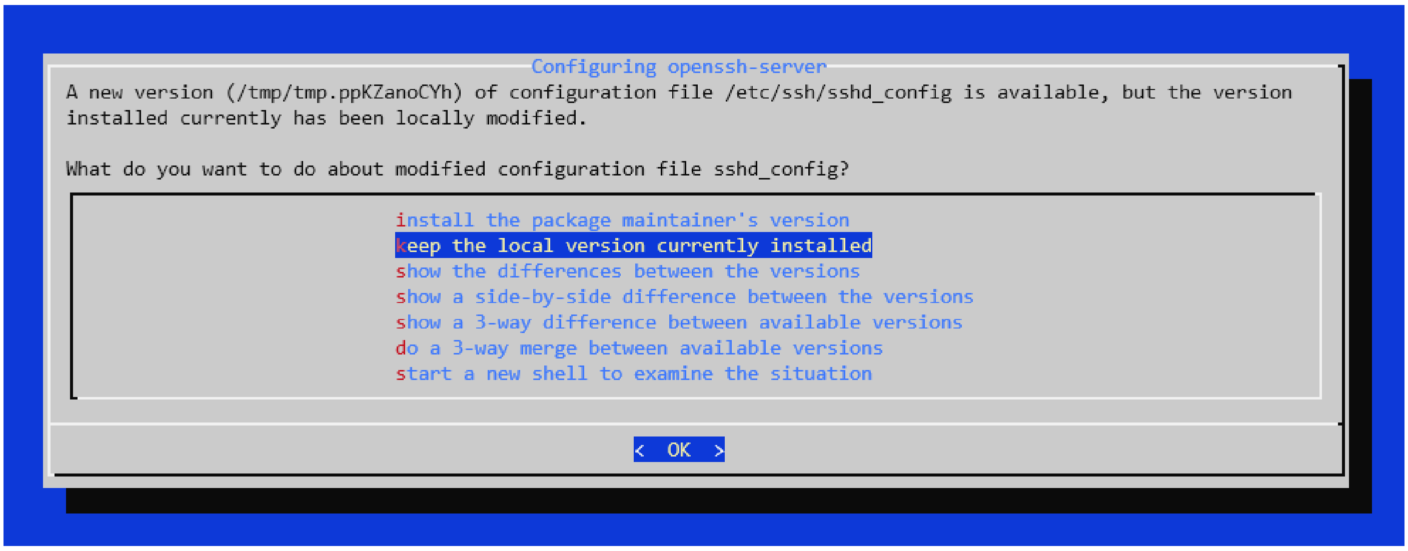
Fortunately, in my case the system booted successfully. Now it's time to perform a full upgrade. This command can actually cause issue, since it does install new packages and removes old ones. Again, watch the screen carefully.

```
$ sudo apt full-upgrade
...
...
...
149 upgraded, 90 newly installed, 19 to remove and 0 not upgraded.
Need to get 276 MB of archives.
After this operation, 625 MB of additional disk space will be used.
Do you want to continue? [Y/n]
```

This is an example question you might get. I have previously edited the sysctl.conf file to fix some performance issues with the service I was running. I pressed D to get the list of changes and after concluding that I want to keep my file, I pressed N.

```
Configuration file '/etc/sysctl.conf'
 ==> Modified (by you or by a script) since installation.
 ==> Package distributor has shipped an updated version.
   What would you like to do about it ?  Your options are:
    Y or I  : install the package maintainer's version
    N or O  : keep your currently-installed version
      D     : show the differences between the versions
      Z     : start a shell to examine the situation
 The default action is to keep your current version.
*** sysctl.conf (Y/I/N/O/D/Z) [default=N] ?
```

or sshd_config changes



```
Configuring openssh-server
A new version (/tmp/tmp.ppKZanoCYh) of configuration file /etc/ssh/sshd_config is available, but the version
installed currently has been locally modified.

What do you want to do about modified configuration file sshd_config?

                      install the package maintainer's version
                      keep the local version currently installed
                      show the differences between the versions
                      show a side-by-side difference between the versions
                      show a 3-way difference between available versions
                      do a 3-way merge between available versions
                      start a new shell to examine the situation

                                   <  OK  >
```

Now reboot and pray :)

```
$ sudo reboot
```

Well, since I was using a 5.4 kernel from Debian 10 backports, instead of the default 4.19, I had to explicitly confirm something during the boot process. Unfortunately, my VPS provider doesn't provide me with access to the pre-boot environment and GRUB, therefore I wasn't able to boot. Took them about a day to get the VPS up and running and even though I am now running Debian 11 which comes with 5.10, I still have the 5.4. Unless you have also installed backported kernel, you should be already running Debian 11 with 5.10 without any troubles.

```
$ cat /etc/debian_version
11.0
```

If we run *sudo apt upgrade* again, we can see a number of unused packages that can be remove.

```
The following packages were automatically installed and are no longer required:
  cpp-8 dh-python gir1.2-glib-2.0 golang-1.11 golang-1.11-doc golang-1.11-go golang-1.11-src libasan5 libdns-export1104
  libfl2 libgirepository-1.0-1 libicu63 libidn11 libip4tc0 libip6tc0 libiptc0 libisc-export1100 libisl19 libjson-c3
  liblua5.2-0 libmatheval1 libmpdec2 libmpx2 libnftables0 libperl5.28 libpgm-5.2-0 libprocps7 libpython2-stdlib
  libpython2.7-minimal libpython2.7-stdlib libpython3.7 libpython3.7-dev libpython3.7-minimal libpython3.7-stdlib
  libreadline7 perl-modules-5.28 python2 python2-minimal python2.7 python2.7-minimal python3-asn1crypto python3-dbus
  python3-entrypoints python3-future python3-gi python3-jeepney python3-keyring python3-keyrings.alt python3-mock
  python3-pbr python3-pycryptodome python3-secretstorage python3-xdg python3.7-minimal
```

```
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
```

In order to do so, run

```
$ sudo apt autoremove
```

or use the *purge* option to remove configuration files of these packages as well

```
$ sudo apt --purge autoremove
```

> That's it. Now make sure that everything works as expected, especially the services and apps you are running. Again, in my case, both of the applications that I am running are unable to start, guess that will take some more troubleshooting :) That's why you never try this on production without prior testing.

# Upgrading Debian kernel (5.4 to 5.10)

I have recently upgraded one of my Debian servers from 10 to 11. The main issue I ran into was the fact that the kernel was backported. This meant that when running Debian 10, the kernel was newer than original (5.4 vs 4.19), but now that I have upgraded to Debian 11, which ships with 5.10 by default, I am now running older version than default. This is nothing to worry about, since the 5.4 is the 20th LTS (Long-Term-Support) release, meaning it will be supported up until December 2025. The sole reason why am I even doing this is to have the default and newest Debian kernel and avoid having a FrankenDebian.

## Check current version

First, check the current version of your kernel.

```
$ uname -r


5.4.0-74-generic
```

## Update and upgrade

It is always recommended to update apt repositories and upgrade any pending packages.

```
$ sudo apt update
$ sudo apt upgrade
```

## Search apt

Issue a search query to apt to see which kernel versions are available in the default repository.

```
$ apt-cache search linux-image | grep amd64


linux-headers-5.10.0-8-amd64 - Header files for Linux 5.10.0-8-amd64
linux-headers-5.10.0-8-cloud-amd64 - Header files for Linux 5.10.0-8-cloud-amd64
linux-headers-5.10.0-8-rt-amd64 - Header files for Linux 5.10.0-8-rt-amd64
linux-image-5.10.0-8-amd64-dbg - Debug symbols for linux-image-5.10.0-8-amd64
linux-image-5.10.0-8-amd64-unsigned - Linux 5.10 for 64-bit PCs
```

```
linux-image-5.10.0-8-cloud-amd64-dbg - Debug symbols for linux-image-5.10.0-8-cloud-amd64
linux-image-5.10.0-8-cloud-amd64-unsigned - Linux 5.10 for x86-64 cloud
linux-image-5.10.0-8-rt-amd64-dbg - Debug symbols for linux-image-5.10.0-8-rt-amd64
linux-image-5.10.0-8-rt-amd64-unsigned - Linux 5.10 for 64-bit PCs, PREEMPT_RT
linux-image-amd64-dbg - Debugging symbols for Linux amd64 configuration (meta-package)
linux-image-amd64-signed-template - Template for signed linux-image packages for amd64
linux-image-cloud-amd64-dbg - Debugging symbols for Linux cloud-amd64 configuration (meta-package)
linux-image-rt-amd64-dbg - Debugging symbols for Linux rt-amd64 configuration (meta-package)
linux-image-5.10.0-8-amd64 - Linux 5.10 for 64-bit PCs (signed)
linux-image-5.10.0-8-cloud-amd64 - Linux 5.10 for x86-64 cloud (signed)
linux-image-5.10.0-8-rt-amd64 - Linux 5.10 for 64-bit PCs, PREEMPT_RT (signed)
linux-image-amd64 - Linux for 64-bit PCs (meta-package)
linux-image-cloud-amd64 - Linux for x86-64 cloud (meta-package)
linux-image-rt-amd64 - Linux for 64-bit PCs (meta-package)
```

Pick the right image and headers for you. In my case it is the `linux-image-5.10.0-8-amd64` and `linux-headers-5.10.0-8-amd64`

## Install new kernel packages

```
$ sudo apt install linux-image-5.10.0-8-amd64 linux-headers-5.10.0-8-amd64
```

In my case the system wanted/needed to install some additional packages as well, don't worry about that too much right now, we can check what was installed later and possibly uninstall if we want to.

The last lines of the installation should look like this:

```
Setting up linux-headers-5.10.0-8-amd64 (5.10.46-4) ...
Setting up initramfs-tools-core (0.140) ...
Setting up initramfs-tools (0.140) ...
update-initramfs: deferring update (trigger activated)
Setting up linux-image-5.10.0-8-amd64 (5.10.46-4) ...
I: /vmlinuz.old is now a symlink to boot/vmlinuz-5.10.0-8-amd64
I: /initrd.img.old is now a symlink to boot/initrd.img-5.10.0-8-amd64
I: /vmlinuz is now a symlink to boot/vmlinuz-5.10.0-8-amd64
I: /initrd.img is now a symlink to boot/initrd.img-5.10.0-8-amd64
/etc/kernel/postinst.d/initramfs-tools:
update-initramfs: Generating /boot/initrd.img-5.10.0-8-amd64
Processing triggers for initramfs-tools (0.140) ...
update-initramfs: Generating /boot/initrd.img-5.10.0-8-amd64
```

## Reboot to apply changes

Such a big change as upgrading the kernel requires the system to be rebooted. Until then, the system will use the old kernel. Even in case the new kernel doesn't boot, you can still use the old one, because the new was just installed alongside with the old.

```
$ sudo reboot
```

It is recommended to have access to the booloader (recovery) in case something goes wrong. **Procceed with caution if you have SSH access only.**

You system should now reboot with the newest kernel. In my case, the VPS booted back to the old kernel, probably because the provider has hard set it to boot only that specific kernel. Consult your provider in case you have a VPS.

# Upgrade Debian 10 to 11 (speedrun)

> This is not supposed to be a quide, just a semi-public documentation of an update I performed on one of my servers.

## Update and upgrade existing

First update repos and upgrade any pending packages:

```
$ sudo apt update
$ sudo apt upgrade
```

I had one pending Nginx update, so I will install it (I have done the same on another server already, so I know the version is safe and doesn't break anything.) After the upgrade, Nginx has been restarted automatically. You can check the current version with `sudo nginx -v`

## Edit sources.list

Go to `/etc/apt` and edit `sources.list` (with elevated privileges).

### Remove deb-src

I removed lines with `deb-src`, because I simply don't need them.

### Change server location

Decided I wanted to change to a closer Debian mirror, so I replaced the old servers.

## Update from new repos

```
$ sudo apt update
```

Autoremove some packages

```
$ sudo apt autoremove
```

## Add bullseye to sources.list

Replace all occurances of `buster` in `/etc/apt/sources.list` with `bullseye`. Don't forget new syntax on `debian-security` line, which is now `bullseye-security`

### Add Nginx

On this server, I have Nginx repo in `/etc/apt/sources.list.d` in `nginx.list` file. Again, replace `buster` with `bullseye`.

## Update and upgrade fully

```
$ sudo apt update
```

```
$ sudo apt upgrade --without-new-pkgs
```

```
$ sudo reboot
```

```
$ sudo apt full-upgrade
```

```
$ sudo apt autoremove
```

This server had a default Debian 10 kernel, so the upgrade went smoothly. Unfortunately, it usually isn't the system itself that breaks during the upgrade, but the services running on it. This time, it was PHP. For whatever reason, the following happened:

- PHP-FPM7.3 service got masked by systemd.
- PHP commands in the command line stopped working.
- Even though Debian 11's PHP version is 7.4, I still only had PHP 7.3 installed, but not working.

> Fixed by installing PHP 7.4 from scratch and reinstalling the application depending on it.