# Kernel patching with kexec: updating a CentOS 7 kernel without a full reboot

Mattias Geniar, Thursday, February 23, 2017

tl;dr: you can use `kexec` to stage a kernel upgrade in-memory without the need for a *full* reboot. Your system will reload the new kernel on the fly and activate it. There will be a service restart of every running service as the new kernel is loaded, but you skip the entire bootloader & hardware initialization.

By using kexec you can upgrade your running Linux machine's kernel without a *full* reboot. Keep in mind, there's still a new kernel load, but it's significantly faster than doing the whole bootloader stage and hardware initialization phase performed by the system firmware (BIOS or UEFI).

Yes, calling this *kernel upgrades without reboots* is a vast exaggeration. You skip *parts of* the reboot, though, usually the slowest parts.

## Installing kexec

On a CentOS 7 machine the `kexec` tools should be installed by default, but just in case they aren't;

```
$ yum install kexec-tools
```

After that, the `kexec` binary should be available to you.

## Install your new kernel

In this example I'll upgrade a rather old CentOS 7 kernel to the latest.

```
$ uname -r
3.10.0-229.14.1.el7
```

So I'm now running the `3.10.0-229.14.1.el7` kernel.

To upgrade your kernel, first install the latest kernel packages.

```
$ yum update kernel
...
==================================================================================
 Package                  Arch      Version                         Repository  Size
==================================================================================
Installing:
 kernel                   x86_64    3.10.0-514.6.1.el7              updates     37 M
```

This will install the

This will install the

```
3.10.0-514.6.1.el7
```

kernel on my machine.

So a quick summary (on new lines, so you see the kernel version difference):

```
From: 3.10.0-229.14.1.el7
To: 3.10.0-514.6.1.el7

$ rpm -qa | grep kernel | sort
kernel-3.10.0-229.14.1.el7.x86_64
kernel-3.10.0-514.6.1.el7.x86_64
```

Once you installed the new kernel, it's time for the `kexec` in-memory upgrading magic.

## In-memory kernel upgrade with kexec

As a safety command, unload any previously attempted kernels first. This is harmless and will make sure you start "cleanly" with your upgrade process.

```
$ kexec -u
```

Now, state the new kernel to be loaded. Note these are the version numbers of the latest installed kernel with `yum`, as shown above.

```
$ kexec -l /boot/vmlinuz-3.10.0-514.6.1.el7.x86_64 \
 --initrd=/boot/initramfs-3.10.0-514.6.1.el7.x86_64.img \
 --reuse-cmdline
```

**Careful: next command will reload a new kernel and *will* impact running services!**

Once prepared, start kexec.

```
$ systemctl kexec
```

Your system will freeze for a couple of seconds, load the new kernel and be good to go.

## Some benchmarks

A very quick and *unscientific* benchmark of doing a `yum update kernel` with and without `kexec`.

```
Normal way, kernel upgrade + reboot: 28s
Kexec way, kernel upgrade + reload: 19s
```

```
Normal way, kernel upgrade + reboot: 28s
Kexec way, kernel upgrade + reload: 19s
```

So you have a couple of seconds of the new kernel load, for big physical machines with lots of RAM, this will be even more as the entire POST check can be skipped with this method.

Here's a side-by-side run of the same kernel update. On the left: the `kexec` flow you've read above. On the right, a classic `yum update kernel && reboot`.

Notice how the left VM never goes into the BIOS or POST checks.

If you're going to be automating these updates, have a look at some existing scripts to help you going: kexec-reboot , ArchWiki on kexec

.