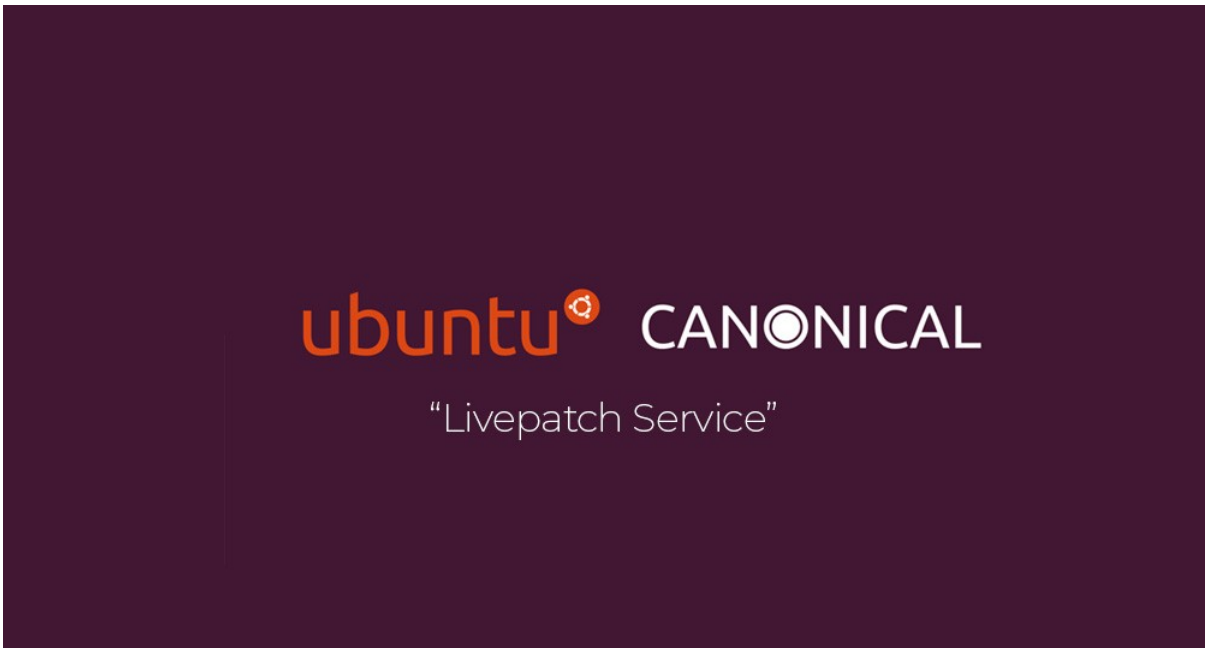
Ubuntu Linux Reference

Ubuntu Linux reference information

Canonical Livepatch



To avoid downtime during kernel upgrades, you can use a feature of the Linux kernel called live patching. This feature makes it possible to implement kernel updates without rebooting.

What Is Livepatch and How Does It Work?

Canonical Livepatch uses the Kernel Live Patching technology built into the standard Linux kernel. Canonical’s Livepatch website notes that massive corporations like AT&T, Cisco, and Walmart use it.

It’s free for personal use on up to three computers, these can be “desktops, servers, virtual machines, or cloud instances.” Organizations can use it on more systems with a paid Ubuntu Advantage subscription.

Kernel Patches Are Necessary But Inconvenient

Linux kernel patches are a fact of life. Keeping your system secure and patched up to date is vital in the inter-connected world we live in. But having to reboot your computer to apply kernel patches can be a pain. Especially if the computer is providing some sort of service to users and you have to co-ordinate or negotiate with them to take the service off-line. And there’s a multiplier. If you maintain several Ubuntu machines, at some point you have to bite the bullet and do each one in turn.

The Canonical Livepatch Service removes all of the aggravation of keeping your Ubuntu systems up to date with critical kernel patches. It's easy to set up and it takes one more chore off your shoulders.

Anything that reduces maintenance efforts, boosts security, and reduces downtime has to be an attractive proposition, right? Yes, but there are some caveats.

- You must be using a Long Term Support (LTS) release of Ubuntu such as 20.04 or 22.04.
- It must be a 64-bit version.
- You must be running Linux Kernel 4.4 or higher
- You need to have an Ubuntu One account. If you don't have an Ubuntu One account, you can sign up for a free account.
- You can use the Canonical Livepatch Service at no cost, but you're limited to three computers per Ubuntu One account. If you have to maintain more than three computers, you'll need additional Ubuntu One accounts.
- If you have physical, virtual, or cloud-hosted servers to look after, you'll need to become an Ubuntu Advantage customer.

Installation

There are two major maintainers for kernel live patches: Canonical, who provides their own Livepatch Service for Ubuntu, and KernelCare who support Ubuntu in addition to most other major Linux distributions. Both require registration to use, and only Canonical's service is free for individual use.

You can register for a Livepatch key at https://auth.livepatch.canonical.com/.

After enrolling, you can install the `canonical-livepatch` snap package. Snap is another Ubuntu package manager that runs alongside `apt`.

```
sudo snap install canonical-livepatch
```

You can enable `canonical-livepatch` with a one-line command using the key you obtained when registering:

```
sudo canonical-livepatch enable your-key
```

The output should contain the message `Successfully enabled device.` The service should run in the background from now on without any further intervention, and you can check its status using `canonical-livepatch status`:

```
sudo canonical-livepatch status
```

After installing, you should see something like this:

```
Output
last check: 50 seconds ago
kernel: 5.15.0-25-generic
server check-in: succeeded
patch state: ✓ all applicable livepatch modules inserted
patch version: 84.1
tier: updates (Free usage; This machine beta tests new patches.)
machine id: 2565a9e7fc9f4405a167e4caf9b9dcf3
```

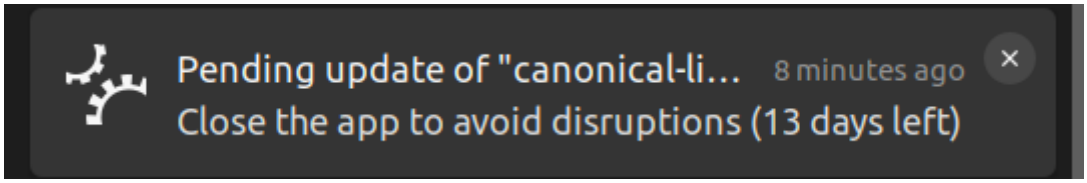
You have now configured automatic kernel updates for your server, meaning it should no longer be necessary to reboot in order to maintain a secure and up-to-date environment.

Updates

From time to time, you may need to update your livepatch software. Since this is a Snap package you will need to perform the following to check if you need to update it.

```
sudo snap info canonical-livepatch
```

You may also see a notification similar to this appear on your desktop's notification panel



To perform the update, follow these steps:

1. Stop your existing Livepatch instance

```
sudo snap stop canonical-livepatch
```

2. Refresh your Livepatch instance

```
sudo snap refresh canonical-livepatch
```

If successful, you should see output in your terminal similar to this:

`canonical-livepatch 10.5.4 from Canonical✓ refreshed`

Commands

Most frequently used CLI (Terminal) commands on Linux

Commands used for System Information

SHUTDOWN

The command “shutdown” is used to shut down the system.

> Note: The shutdown command needs superuser privileges. Hence, you should either be root or run the command with sudo.

Using the command with no flags will schedule a shutdown 1 minute from execution.

```
sudo shutdown
```

Use the following to IMMEDIATELY shutdown your system.

```
sudo shutdown now
```

You can schedule a shutdown in future by providing the time argument either in +t format or in hh:mm format. For example, if you want to shutdown the system after 15 minutes, you can use this command:

```
sudo shutdown +15
```

If you want to shutdown the system at 6 PM in the afternoon, you can use it in the following manner:

```
sudo shutdown 18:00
```

Cancel a shutdown

```
sudo shutdown -c
```

Reboot a system

```
sudo shutdown -r
```

```
sudo reboot
```

```
$ sudo shutdown
Shutdown scheduled for Thu 2023-03-02 20:12:13 EST, use 'shutdown -c' to cancel.
```

Commands used for File Handling

LS
The command “ls” displays the list of all directories, folder, and files present in the current directory.

LS - LTR
The above-mentioned command displays the name of directories, folders, files with their respective owner name, group’s name, and rights your user has over these.

```
ls
ls -ltr
```

```
/$ ls
bin    dev   lib
libx32      mnt   root
snap      sys  usr
boot   etc   lib32
lost+found  opt   run
srv       tim  var
cdrom  home  lib64
media       proc  sbin
swapfile  tmp
```

```
/$ ls -ltr
total 2097256
drwxr-xr-x   2 root
root       4096 Feb  9
2021 mnt
drwxr-xr-x  15 root
root       4096 Feb  9
2021 var
drwx------   2 root
root      16384 Feb 11
2022 lost+found
lrwxrwxrwx   1 root
root          8 Feb 11
2022 sbin -> usr/sbin
lrwxrwxrwx   1 root
root         10 Feb 11
2022 libx32 -> usr/libx32
lrwxrwxrwx   1 root
root          9 Feb 11
2022 lib64 -> usr/lib64
lrwxrwxrwx   1 root
root          9 Feb 11
2022 lib32 -> usr/lib32
lrwxrwxrwx   1 root
root          7 Feb 11
2022 lib -> usr/lib
lrwxrwxrwx   1 root
root          7 Feb 11
2022 bin -> usr/bin
drwxrwxr-x   2 root
root       4096 Feb 11
2022 cdrom
```

MKDIR
The command “mkdir” allows users to create directories/folders in the system. The user running this command must have suitable rights over the parent directory to create a directory or they will receive an error.
Syntax: *mkdir New_Directory’s_Name*
	`mkdir NewDirectory`	`~$ mkdir poopoo`
		`~$`
		`~$ ls`
		`Android`
		`Pictures`
		`AppImages poopoo`

RMDIR
The command “rmdir” allows users to remove directories/folders from the system. The user running this command must have suitable rights over the parent directory to remove a directory AND the directory must not have any files or sub-directories within it or you will receive an error.
Syntax: *rmdir Directory’s_Name*
	`rmdir DirectoryName`	`~$ rmdir poopoo`
		`rmdir: failed to remove 'poopoo': Directory not empty`
		`# Could not delete directory`
		`# "poopoo" because it is not`
		`# empty`
		
		`~$ rm poopoo`
		`rm: cannot remove 'poopoo': Is a directory`
		`# Could not remove "poopoo"`
		
		`# because it is not a file`

RM
The command “rm” is used to remove files from a directory.

```
rm filename
```

```
# Listing shows poopoo.txt
# file exists under
# direcotry "poopoo"
~/poopoo$ ls
poopoo.txt

~/poopoo$ rm poopoo.txt

# listing now shows
# poopoo.txt has been
# removed (deleted)
# from directory "poopoo"
~/poopoo$ ls
~/poopoo$
```

RM -RF
Permanently deletes the specified directory and ALL files and sub-directories beneath the specified directory.

> Be VERY careful using this command as you can inadvertently delete your whole drive!

```
rm -rf /path/to/dir/name
```

```
# Directory "poopoo" exists

# in the listing below
~$ ls
 Android
Pictures
 AppImages          poopoo
 Audio              Public


~$ rm -rf poopoo
~$
# Successfully removed
# "poopoo" directory
# and all its contents
# as can be seen in the
# listing below

~$ ls
 Android
Parkitect
 AppImages
Pictures
 Audio              Public
```

MV
The command “mv” is used for two purposes
• To move files or directories from one path to another path in the system.
• To rename a file or folder.
	`mv Source_File_name Destination_File_Name`
	
	`mv File_name New_name_for_file`	

CP The command “cp” is used to copy data from a source file to the destination file. Its function is almost like the command “mv”. The only difference is by using the command “cp” the source file is not removed from the directory after its data is moved to the destination file.	`cp source_file_name destination_file_name`	

TOUCH
Creates an empty file at the specified path with the specified name.
Useful for creating a blank file you intend to edit with a CLI editor, such as VIM or NANO.
	`touch /path/name/filename.ext`	`~$ ls doc.txt`
		`ls: cannot access 'doc.txt': No such file or directory`
		`~$ touch /home/tim/doc.txt`
		`~$ ls doc.txt`
		`doc.txt`

CAT The command “cat” is a reverse of the command “tac”. It is used to display each line of the file starting from the first row and finishing on its last row. This command is more frequently used than “tac”.	`cat file_name`	
ECHO The command “echo” used to display any expression that is passed as an argument.	`echo expression_to_be_displayed`	`~/poopoo$ echo something-poopoo` `something-poopoo`
GREP The command “grep” is used to search for a text in the specified file/folder.	`grep “expression_to_be_Searched” file_name_to_search_in`	

ZIP
The command “zip” is used to compress one or more files and store them in a new file with .zip extension.
	`zip new_zip_file_name.zip`	`~/poopoo$ zip files.zip file1.txt file2.txt file3.txt`
		`  adding: file1.txt (stored 0%)`
		`  adding: file2.txt (stored 0%)`
		`  adding: file3.txt (stored 0%)`
		`~/poopoo$ ls`
		`file1.txt  file2.txt  file3.txt  files.zip`

UNZIP
The command “unzip” is used to decompress a .zip file and extract all the files within to current directory.
	`unzip zip_file_name.zip`	`~/poopoo$ unzip files.zip`
		`Archive:  files.zip`
		`replace file1.txt? [y]es, [n]o, [A]ll, [N]one, [r]ename: A`
		`  extracting: file1.txt`
		`  extracting: file2.txt`
		`  extracting: file3.txt`

SUDO
Sudo stands for SuperUser DO and is used to access restricted files and operations. By default, Linux restricts access to certain parts of the system preventing sensitive files from being compromised.
The `sudo` command temporarily elevates privileges allowing users to complete sensitive tasks without logging in as the root user.

```
sudo some-command
```

sudo -i elevates the user to root for the remainder of the session rather than a command by command basis.

```
sudo -i
```

```
# No directory called
"peepee"
# exists
$ ls
bin    dev   lib
libx32      mnt   root
snap      sys  usr
boot   etc   lib32
lost+found  opt   run
srv       tim  var
cdrom  home  lib64
media       proc  sbin
swapfile  tmp

# Attempt to make directory

# "peepee" as a noraml user
# fails because I'm
# trying to make the
# directory in a path I
# don't have rights to
$ mkdir peepee
mkdir: cannot create
directory ‘peepee’:
Permission denied

# using SUDO to temporarily

# elevate my privileges, I
# can now create the
# directory "peepee"
# in the path as can be
seen
# in the listing below:
$ sudo mkdir peepee
[sudo] password for tim:

$ ls
bin    dev   lib
libx32      mnt     proc
sbin  swapfile  tmp
boot   etc   lib32
lost+found  opt     root
```
