



# An Introduction to Useful Bash Aliases and Functions

Posted March 21, 2014 ©381.8k

LINUX BASICS

By [Justin Ellingwood](#)[Become an author](#)

## Introduction

The more you operate on the command line, the more you will find that the majority of the commands you use are a very small subset of the available commands. Most tasks are habitual and you may run these the same way every day.

While the makers of many of the most common command utilities have attempted to eliminate extraneous typing by using shortened names (think of how many keystrokes you save daily by typing “ls” instead of “list” and “cd” instead of “change-directory”), these are not ubiquitous. Additionally, many people always run commands with the same few options enabled every time.

Luckily, bash allows us to create our own shortcuts and time-savers through the use of aliases and shell functions. In this guide, we’ll discuss how to make use of these and give you some useful examples to get you started in the right direction.

---

## How To Declare a Bash Alias

Declaring aliases in bash is very straight forward. It’s so easy that you should try it now.

Sign up for our newsletter. Get the latest tutorials on SysAdmin and open source topics.



Sign Up

[ROLL TO TOP](#)

```
alias alias_name="command_to_run"
```

Note that there is no spacing between the neighbor elements and the equal sign. This is not optional. Spaces here will break the command.

Let's create a common bash alias now. One idiomatic command phrase that many people use frequently is `ls -lha` or `ls -lhA` (the second omits the current and parent directory listing). We can create a shortcut that can be called as `ll` by typing:

```
alias ll="ls -lhA"
```

Now, we can type `ll` and we'll get the current directory's listing, in long format, including hidden directories:

```
ll
```

```
-rw-r--r-- 1 root root 3.0K Mar 20 18:03 .bash_history
-rw-r--r-- 1 root root 3.1K Apr 19 2012 .bashrc
drwx----- 2 root root 4.0K Oct 24 14:45 .cache
drwx----- 2 root root 4.0K Mar 20 18:00 .gnupg
-rw-r--r-- 1 root root 0 Oct 24 17:03 .mysql_history
-rw-r--r-- 1 root root 140 Apr 19 2012 .profile
drwx----- 2 root root 4.0K Oct 24 14:21 .ssh
-rw----- 1 root root 3.5K Mar 20 17:24 .viminfo
```

If you want to get rid of an alias, just use the `unalias` command:

```
unalias ll
```

The alias is now removed.

You can list all of your configured aliases by passing the `alias` command without any arguments:

Sign up for our newsletter. Get the latest tutorials on SysAdmin and open source topics. 

Sign Up

[ROLL TO TOP](#)

To temporarily bypass an alias (say we aliased `ls` to `ls -a`), we can type:

```
\ls
```

This will call the normal command found in our path, without using the aliased version.

Assuming you did not unset it, the `ll` alias will be available throughout the current shell session, but when you open a new terminal window, this will not be available.

To make this persistent, we need to add this into one of the various files that is read when a shell session begins. Popular choices are `~/.bashrc` and `~/.bash_profile`. We just need to open the file and add the alias there:

```
nano ~/.bashrc
```

At the bottom or wherever you'd like, add the alias you added on the command line. Feel free to add a comment declaring an entire section devoted to bash aliases:

```
#####  
# Aliases  
#####  
  
alias ll="ls -lhA"
```

This alias or a variation might actually already be in your file. Many distributions ship with a set of standard bash configuration files with a few useful aliases.

Save and close the file. Any aliases you added will be available next time you start a new shell session. To read any changes you made in your file into your *current* session, just tell bash to re-read the file now:

```
source ~/.bashrc
```

Sign up for our newsletter. Get the latest tutorials on SysAdmin and open source topics.



Sign Up

ROLL TO TOP

## Alias Examples

Now that you know how to create your own aliases, let's talk about some popular ones that may be useful to you. These can be found throughout the web, and some may also be included in your distribution's default bash configuration as well.

### Navigation and Listing

Many of the most simple Linux commands are more helpful when you apply some formatting and options.

We discussed one `ls` example above, but there are many others you may find.

Make `ls` display in columns and with a file type indicator (end directories with `/`, etc) by default:

```
alias ls="ls -CF"
```

We can also anticipate some typos to make it call the correct command:

```
alias sl="ls"
```

Let's also make an alias to pipe our output to `less` for viewing large directory listings with the long format:

```
alias lsl="ls -lhFA | less"
```

How about we stray from `ls` and try some helpful commands for `cd`.

This one will change to your parent directory, even when you forget the space:

```
alias cd..="cd .."
```

Sign up for our newsletter. Get the latest tutorials on SysAdmin and open source topics.



Sign Up

ROLL TO TOP

We can find files in our current directory easily by setting this alias:

```
alias fhere="find . -name "
```

## System Aliases

How about some of our monitoring and system stats commands? I call these with the same options every time, so I might as well make some aliases.

This one will list our disk usage in human-readable units including filesystem type, and print a total at the bottom:

```
alias df="df -Tha --total"
```

We might as well add an alias for our preferred `du` output as well:

```
alias du="du -ach | sort -h"
```

Let's keep going in the same direction by making our free output more human friendly:

```
alias free="free -mt"
```

We can do a lot with our listing process table. Let's start out by setting a default output:

```
alias ps="ps auxf"
```

How about we make our process table searchable. We can create an alias that searches our process for an argument we'll pass:

```

alias exec "set aux | grep -v grep | grep -i 'VC7_0'"

```

**Sign up for our newsletter.** Get the latest tutorials on SysAdmin and open source topics.

Enter your email address

## Sign Up

[ROLL TO TOP](#)

Now, when we call it with the process name we're looking for as an argument, we'll get a nice, compact output:

```
psg bash
```

USER	PID	%CPU	%MEM	VSZ	RSS	TTY	STAT	START	TIME	COMMAND
1001	5227	0.0	0.0	26320	3376	pts/0	Ss	16:29	0:00	bash

## Miscellaneous Aliases

One common option to the `mkdir` command that we use often is the `-p` flag to make any necessary parent directories. We can make this the default:

```
alias mkdir="mkdir -p"
```

We might want to add a `-v` flag on top of that so we are told of every directory creation, which can help us recognize quickly if we had a typo which caused an accidental directory branch:

```
alias mkdir="mkdir -pv"
```

When downloading files from the internet with `wget`, in almost all circumstances, you'll want to pass the `-c` flag in order to continue the download in case of problems. We can set that with this:

```
alias wget="wget -c"
```

We can search our history easily like with a `grep` of the `history` command's output. This is sometimes more useful than using `CTRL-R` to reverse search because it gives you the command number to do more complex recalls afterwards:

Sign up for our newsletter. Get the latest tutorials on SysAdmin and open source topics.



Sign Up

ROLL TO TOP

I have a few system tools that I prefer to upgrade from the standard version to more complex tools. These will only work if you've downloaded the required utilities, but they can be very helpful. Keep in mind that these may affect your other aliases.

This one replaces the conventional `top` command with an enhanced version that is much easier on the eyes and can be sorted, searched, and scrolled without complications:

```
alias top="htop"
```

In a similar way, the `ncdu` command can be downloaded which presents file and directory sizes in an interactive ncurses display that you can browse and use to perform simple file actions:

```
alias du="ncdu"
```

There's an upgraded utility for `df` as well that's called `pydf`. It provides colored output and text-based usage bars. We can default to using this utility if we have it:

```
alias df="pydf"
```

Have you ever needed your public IP address from the command line when you're behind a router using NAT? Something like this could be useful:

```
alias myip="curl http://ipecho.net/plain; echo"
```

For my own purposes, I like to optimize the images I upload for articles to be 690px or less, so I use the ImageMagick package (`sudo apt-get install imagemagick` if not already available) which contains a command called `mogrify` that does just this. I have this command in my `~/.bashrc` file:

```
alias mogrify="mogrify -resize 690x *.*"
```

Sign up for our newsletter. Get the latest tutorials on SysAdmin and open source topics.



Sign Up

ROLL TO TOP

This will resize all of the PNG images in the current directory, only if they are wider than 690px.

If I then have to upload them to a server, I can use `sftp` to connect and automatically change to a specific directory:

```
alias upload="sftp username@server.com:/path/to/upload/directory"
```

---

## Getting Started with Bash Functions

Although aliases are quick and easy to implement, they are quite limited in their scope. You'll find as you're trying to chain commands together that you can't access arguments given at runtime very well, among other things. Aliases can also be quite slow at times because they are read after all functions.

There is an alternative to aliases that is more robust and can help you bridge the gap between bash aliases and full shell scripts. These are called shell functions. They work in almost the same way as aliases but are more programmatic and accept input in a standard way.

We won't go into extensive detail here, because these can be used in so many complex situations and bash is an entire scripting language, but we'll go over some basic examples.

For starters, there are two basic ways to declare a bash syntax. The first uses the `function` command and looks something like this:

```
function function_name {  
    command1  
    command2  
}
```

The other syntax uses a set of parentheses which is more "C-like":

Sign up for our newsletter. Get the latest tutorials on SysAdmin and open source topics. 

Sign Up

[ROLL TO TOP](#)



```
command2  
}
```

We can compress this second form into one line and separate the commands with semicolons. A semicolon *must* come after the last command too:

```
function_name () { command1; command2; }
```

Let's start off by demonstrating an extremely useful bash function. This one will create a directory and then immediately move into that directory. This is usually exactly the sequence we take when making new directories:

```
mcd () {  
    mkdir -p $1  
    cd $1  
}
```

Now, when we use use this function instead of the regular `mkdir` command to auto change into the directory after creation:

```
mcd test  
pwd
```

```
/home/demouser/test
```

One cool function that you'll see around is the `extract` function. This combines a lot of utilities to allow you to decompress just about any compressed file format. There are a number of variations, but this one comes from [here](#):

```
function extract {  
    if [ -z "$1" ]; then  
        # display usage if no parameters given  
        echo "Usage: extract <path/file name>.<zip|rar|bz2|gz|tar|tbz2|tgz|Z|7z|xz|ex|tar.bz2|tar.gz|tar.xz>"  
    fi  
}
```

Sign up for our newsletter. Get the latest tutorials on SysAdmin and open source topics.

✕ h/file\_name\_3

ROLL TO TOP

Sign Up

```

for n in $@
do
    if [ -f "$n" ] ; then
        case "${n%,}" in
            *.tar.bz2|*.tar.gz|*.tar.xz|*.tbz2|*.tgz|*.txz|*.tar)
                tar xvf "$n" ;;
            *.lzma)      unlzma ./"$n" ;;
            *.bz2)       bunzip2 ./"$n" ;;
            *.rar)       unrar x -ad ./"$n" ;;
            *.gz)        gunzip ./"$n" ;;
            *.zip)       unzip ./"$n" ;;
            *.z)         uncompress ./"$n" ;;
            *.7z|*.arj|*.cab|*.chm|*.deb|*.dmg|*.iso|*.lzh|*.msi|*.rpm|*.udf|*.wim|*.xar)
                7z x ./"$n" ;;
            *.xz)        unxz ./"$n" ;;
            *.exe)       cabextract ./"$n" ;;
            *)
                echo "extract: '$n' - unknown archive method"
                return 1
                ;;
        esac
    else
        echo "'$n' - file does not exist"
        return 1
    fi
done
fi
}

```

This function takes the first argument and calls the appropriate utility program based on the file extension used.

## Conclusion

Hopefully this guide has given you some inspiration for creating your own aliases and bash functions. Extensive use of these can help make your time in the shell more enjoyable and less complex.

Sign up for our newsletter. Get the latest tutorials on SysAdmin and open source topics.




Sign Up

[ROLL TO TOP](#)

Remember to be wary of redefining existing commands with behavior that is potentially destructive. Even doing the opposite and aliasing a command to a safer variant (always asking for confirmation before deleting recursively, for instance) can come back to bite you the first time you're on a system without it once you've come to rely on it.

To find candidates that might be good to create aliases for, it might be a good idea to search your history for your most commonly used commands. A one-liner from [here](#) allows us to see our most used commands:

```
history | awk '{CMD[$2]++;count++;}END { for (a in CMD)print CMD[a] " " CMD[a]/count*100
```

```
1 247 24.7% cd
2 112 11.2% vim
3 90 9% exit
4 72 7.2% ls
5 70 7% xset
6 56 5.6% apt-get
7 40 4% vlc
8 40 4% rm
9 38 3.8% screen
10 27 2.7% htop
```

We can easily use this list as a starting point for commands that we frequently utilize. In the comments section, feel free to share your favorite bash aliases and functions:

By Justin Ellingwood

By [Justin Ellingwood](#)

Sign up for our newsletter. Get the latest tutorials on SysAdmin and open source topics.



Sign Up

ROLL TO TOP

[Report an issue](#)

## Related

### TUTORIAL

#### How to Add and Delete Users on Ubuntu 16.04

Learning how to manage users effectively is an essential skill for any Linux system administrator. In this guide, we will discuss

### TUTORIAL

#### How To Set Up SSH Keys on CentOS 8

SSH, or secure shell, is an encrypted protocol used to administer and communicate with servers. When working with a

### TUTORIAL

#### Initial Server Setup with CentOS 8

When you first create a new CentOS 8 server, there are a few configuration steps that you should take early on as

### TUTORIAL

#### How to Add and Delete Users on Ubuntu 18.04

Learning how to manage users effectively is an essential skill for any Linux system administrator. In this guide, we will discuss

## Still looking for an answer?

Sign up for our newsletter. Get the latest tutorials on SysAdmin and open source topics.



Sign Up

ROLL TO TOP

[Ask a question](#)[Search for more help](#)

## 15 Comments

Leave a comment...

[Sign In to Comment](#)

 [aysorth](#) March 26, 2014

0 I also have pseudo-aliases, for example a function that does service \$1 start, which is otherwise impossible. Might save someone a lot of time.

[Reply](#) [Report](#)

 [heliocampos](#) June 13, 2015

0 Hi,

- Want to give my “two cents help” on aliases:

1) Using your preferred editor, type the following command to a file called ~/.bash\_aliases

```
alias updateAliases="/etc/alternatives/editor ~/.bash_aliases; source ~/.bash_aliases"
```

Sign up for our newsletter. Get the latest tutorials on SysAdmin and open source topics.



[Sign Up](#)

[ROLL TO TOP](#)

```
source ~/.bash_aliases
```

- Now you have a command that helps you create all the aliases you need. To create a new alias just do the following:

1) Type the following command in your prompt:

```
updateAliases
```

2) It will open the `~/.bash_aliases` for you to write/update your aliases. Now you do what you need to do regarding creating new aliases or updating existing ones;

3) Save the file and exit the editor;

- Now you have your aliases updated without the need to type other command or restart the prompt.

PS: If doing all the above, when you enter a prompt, it does not find your aliases, do the following:

```
echo -e '\nsource ~/.bash_aliases' >> ~/.bashrc
```

- It will ensure you have your aliases on every terminal you open.

Regards,

[Reply](#) [Report](#)

 [ZacharySmith4989](#) October 18, 2015

 1 I couldn't get:

```
alias updateAliases="/etc/alternatives/editor ~/.bash_aliases; source  
~/.bash_aliases"
```

to work. I think the 2nd command is running before I've added the alias, because when I try to call a newly created alias, it does not work at first. But, after running `updateAliases` again,

Sign up for our newsletter. Get the latest tutorials on SysAdmin and open source topics. 

Sign Up

[ROLL TO TOP](#)

Anyway, I have an alias that opens up where I keep my aliases, and then an alias that refreshes stuff.

Also, I have this command:

```
als () {  
  echo "alias" $1="'$2'" >> ~/.bash_profile  
  source ~/.bash_profile  
}
```

It allows me to quickly add an alias. Eg, I could in terminal:


```
als hola echo\ hola
```

which immediately adds this alias to my permanent and current aliases:

```
hola="echo hola"
```

I hope this helps someone! (:

[Reply](#) [Report](#)

 [sundragmehere](#) July 4, 2017  
0 Nice!

[Reply](#) [Report](#)

 [kalfusisagod](#) May 8, 2016

0 Good article, just remember that if you create a **.bash\_profile** file to hold your aliases, it will override the **.bashrc** ones (at least in Ubuntu 14.04). In my case, I had a custom prompt for **PS1** set in **.bashrc** and once I've created a *.bashprofile with my custom aliases, my prompt change back to the long default one. Once I've removed the .bashprofile* it started working. So I just put all my aliases in **.bashrc**

Some of my custom aliases are:

```
alias update='sudo apt-get update && sudo apt-get upgrade && sudo apt-get autoremove'  
alias reboot='sudo shutdown -r now'
```

Sign up for our newsletter. Get the latest tutorials on SysAdmin and open source topics.

 so, can I just

Enter your email address

Sign Up

ROLL TO TOP

multiple servers to update my preferences.

Thanks for the article!

[Reply](#) [Report](#)

^ [rampatra](#) July 1, 2016

0 How can I add a file containing multiple shell functions to the path so that I can just run the functions from any location?

[Reply](#) [Report](#)

^ [jellingwood](#) July 1, 2016

1 @ramswaroop You can do that by sourcing the file into your `~/.bashrc` or `~/.bash_profile`. Bash functions don't really use the `PATH` variable.

Try adding something like this to one of those files:

```
. . .  
if [ -f /path/to/bash/functions/file ]; then  
    source /path/to/bash/functions/file  
fi
```

Hopefully that helps.

[Reply](#) [Report](#)

^ [rampatra](#) July 1, 2016

0 Hey, thanks for the answer.

[Reply](#) [Report](#)

^ [vikky](#) July 25, 2016

0 It is really nice

[Reply](#) [Report](#)

^ [vikky](#) July 25, 2016

0 please send a customization of commands of linux in #program

[Reply](#) [Report](#)

^ [doceantemp](#) November 14, 2016

Sign up for our newsletter. Get the latest tutorials on SysAdmin and open source topics. X

Enter your email address

Sign Up


[ROLL TO TOP](#)



in all of your example shell 'c-style' functions, you've included a space between, `**function_name**` and `()`.

my functions would not work unless this erroneous space was eliminated, not sure if this is because of strict bash syntax or something else but if anyone else comes across this issue and needs help resolving, viola.

[Reply](#) [Report](#)

 **pifou42** February 15, 2017  
0

I've been using simple aliases for years, but this tutorial was still instructive, and the "extract" function decided me to make a little effort, and create one "fuction" I've often found myself missing.

Since I'm still quite newbie at this, I finally opted for a script, but it works the same, could probably be translated to a function very easily, and more importantly, demonstrates that pretty useful things can be done **without needing much skill or specific knowledge**. *(I don't know bash scripting syntax at all, but with StackExchange, about anyone can code something)*

I've created a simple script to show the different apache logs from a computer, with a selection menu letting the user choose one for *a tail -f*, typically useful for debugging.

The script also accepts a single argument, for quicker use.

Here is it, in the form of copy/pastable shell code, intended to be both **readable** and **maintainable**.

Just don't forget to also put the alias in your `.profile` or `.bashrc` if you intend to use it.

```
mkdir -p $HOME/functions/ 2>/dev/null
FCT_APACHE_LOGS="$HOME/functions/show_apache_logs.sh"
echo '#!/bin/bash'
echo '# Variables used'
echo 'unset log logFile logsList i choice'
echo ''
echo '# Retrieval of existing Apache logs list'
echo 'for logFile in /var/log/apache2/*.log; do # Whitespace-safe but not recursive.'
echo '  logsList[++i]="$logFile"'
echo 'done'
echo ''
echo '# If a parameter is set, menu is skipped'
echo 'if [ $1 ]; then'
```

Sign up for our newsletter. Get the latest tutorials on SysAdmin and open source topics.




Sign Up


[ROLL TO TOP](#)

```

echo '  tailf ${logsList[$1]}'
echo 'else'
echo '# Else, the list of apache logs is displayed, and one can be chosen with a numbe
echo '  i=0'
echo '  echo'
echo '  echo "Logs availables :"'
echo '  echo'
echo '  for log in "${logsList[@]}" ; do'
echo '    ((++i))'
echo '    echo "  $i) $log" ;'
echo '  done'
echo '  echo'
echo '  while true ; do'
echo '    read -p "Tailf which log file ? " choice'
echo '    if [ ! $choice ] ; then #Defaults to standard error log'
echo '      tailf /var/log/apache2/error.log'
echo '    else'
echo '      if [ "${logsList[$choice]+'isset'}" ] ; then'
echo '        echo "  Content of ${logsList[choice]} : "'
echo '        echo'
echo '        tailf ${logsList[choice]}'
echo '        break'
echo '      else'
echo '        echo "Invalid choice"'
echo '      fi'
echo '    fi'
echo '  done'
echo 'fi;'
chmod +x $FCT_APACHE_LOGS
alias _al="$FCT_APACHE_LOGS"

```

[Reply](#) [Report](#)

 [jimsander](#) May 25, 2018

0 just to add mine also... "heredoc" is very handy for all those echo's

```

mkdir -p $HOME/functions/ 2>/dev/null
FCT_APACHE_LOGS="$HOME/functions/show_apache_logs.sh"

```

Sign up for our newsletter. Get the latest tutorials on SysAdmin and open source topics. 

Enter your email address

Sign Up

[ROLL TO TOP](#)

```
# Variables used
. . .
done
fi;
EOF
```

[Reply](#) [Report](#)

^ [nire0510](#) April 30, 2017

0 <https://www.npmjs.com/package/as-known-as>

A CLI tool which lets you access your aliases from all of your machines

[Reply](#) [Report](#)

^ [tarpanpathak](#) January 6, 2018

0 Great article. Thanks [@jellingwood](#).

[Reply](#) [Report](#)

This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.



Sign up for our newsletter. Get the latest tutorials on SysAdmin and open source topics. ✕

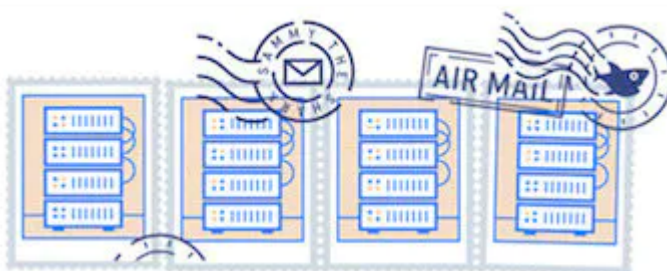
[Sign Up](#)[ROLL TO TOP](#)

You get paid; we donate to tech nonprofits.



**CONNECT WITH OTHER DEVELOPERS**

Find a DigitalOcean Meetup near you.



**GET OUR BIWEEKLY NEWSLETTER**

Sign up for Infrastructure as a Newsletter.

Featured on [Community](#) [Kubernetes Course](#) [Learn Python 3](#) [Machine Learning in Python](#)  
[Getting started with Go](#) [Intro to Kubernetes](#)

DigitalOcean Products [Droplets](#) [Managed Databases](#) [Managed Kubernetes](#) [Spaces](#) [Object Storage](#)  
[Marketplace](#)

Sign up for our newsletter. Get the latest tutorials on SysAdmin and open source topics.

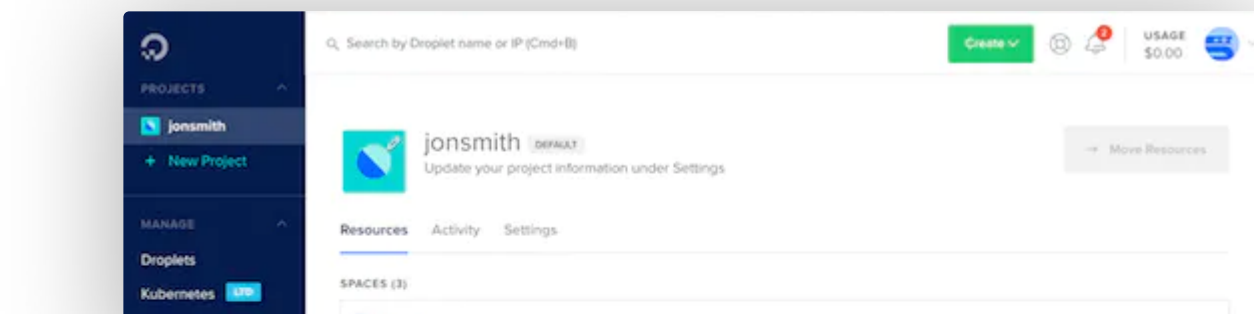


Sign Up

[ROLL TO TOP](#)

DigitalOcean makes it simple to launch in the cloud and scale up as you grow – whether you're running one virtual machine or ten thousand.

Learn More



© 2020 DigitalOcean, LLC. All rights reserved.

## Company

About  
Leadership  
Blog  
Careers  
Partners  
Referral Program  
Press  
Legal & Security

## Products

Products Overview  
Pricing  
Droplets  
Kubernetes  
Managed Databases  
Spaces  
Marketplace  
Load Balancers  
Block Storage  
Tools & Integrations  
API  
Documentation  
Release Notes

Community

Contact

Tutorials

Get Support

Sign up for our newsletter. Get the latest tutorials on SysAdmin and open source topics.



Sign Up

ROLL TO TOP

[Tags](#)[Report Abuse](#)[Product Ideas](#)[System Status](#)[Meetups](#)[Write for DOnations](#)[Droplets for Demos](#)[Hatch Startup Program](#)[Shop Swag](#)[Research Program](#)[Open Source](#)[Code of Conduct](#)

Sign up for our newsletter. Get the latest tutorials on SysAdmin and open source topics.



Sign Up

[ROLL TO TOP](#)