

Iptables Essentials: Common Firewall Rules and Commands

Posted August 10, 2015 © 984.7k FIREWALL CENTOS UBUNTU



By Mitchell Anicas
[Become an author](#)

Introduction

Iptables is the software firewall that is included with most Linux distributions by default. This cheat sheet-style guide provides a quick reference to iptables commands that will create firewall rules are useful in common, everyday scenarios. This includes iptables examples of allowing and blocking various services by port, network interface, and source IP address.

How To Use This Guide

- If you are just getting started with configuring your iptables firewall, check out our [introduction to iptables](#)
- Most of the rules that are described here assume that your iptables is set to **DROP** incoming traffic, through the default input policy, and you want to selectively allow traffic in

- Use whichever subsequent sections are applicable to what you are trying to achieve. Most sections are not predicated on any other, so you can use the examples below independently
- Use the Contents menu on the right side of this page (at wide page widths) or your browser's find function to locate the sections you need
- Copy and paste the command-line examples given, substituting the values in red with your own values

Keep in mind that the order of your rules matter. All of these `iptables` commands use the `-A` option to append the new rule to the end of a chain. If you want to put it somewhere else in the chain, you can use the `-I` option which allows you to specify the position of the new rule (or simply place it at the beginning of the chain by not specifying a rule number).

Note: When working with firewalls, take care not to lock yourself out of your own server by blocking SSH traffic (port 22, by default). If you lose access due to your firewall settings, you may need to connect to it via the console to fix your access. Once you are connected via the console, you can change your firewall rules to allow SSH access (or allow all traffic). If your saved firewall rules allow SSH access, another method is to reboot your server.

Remember that you can check your current iptables ruleset with `sudo iptables -S` and `sudo iptables -L`.

Let's take a look at the iptables commands!

Saving Rules

Iptables rules are ephemeral, which means they need to be manually saved for them to persist after a reboot.

Ubuntu

On Ubuntu, the easiest way to save iptables rules, so they will survive a reboot, is to use the `iptables-persistent` package. Install it with apt-get like this:

```
$ sudo apt-get install iptables-persistent
```

During the installation, you will be asked if you want to save your current firewall rules.

If you update your firewall rules and want to save the changes, run this command:

```
$ sudo netfilter-persistent save
```

On versions of Ubuntu prior to 16.04, run this command instead:

```
$ sudo invoke-rc.d iptables-persistent save
```

CentOS 6 and Older

On CentOS 6 and older—CentOS 7 uses FirewallD by default—you can use the `iptables` init script to save your iptables rules:

```
$ sudo service iptables save
```

This will save your current iptables rules to the `/etc/sysconfig/iptables` file.

Listing and Deleting Rules

If you want to learn how to list and delete iptables rules, check out this tutorial: [How To List and Delete Iptables Firewall Rules](#).

Generally Useful Rules

This section includes a variety of iptables commands that will create rules that are generally useful on most servers.

Allow Loopback Connections

The **loopback** interface, also referred to as `lo`, is what a computer uses to forward network connections to itself. For example, if you run `ping localhost` or `ping 127.0.0.1`, your server will ping itself using the loopback. The loopback interface is also used if you configure your application server to connect to a database server with a "localhost" address. As such, you will want to be sure that your firewall is allowing these connections.

To accept all traffic on your loopback interface, run these commands:

```
$ sudo iptables -A INPUT -i lo -j ACCEPT
$ sudo iptables -A OUTPUT -o lo -j ACCEPT
```

Allow Established and Related Incoming Connections

As network traffic generally needs to be two-way—incoming and outgoing—to work properly, it is typical to create a firewall rule that allows **established** and **related** incoming traffic, so that the server will allow return traffic to outgoing connections initiated by the server itself. This command will allow that:

```
$ sudo iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
```

Allow Established Outgoing Connections

You may want to allow outgoing traffic of all **established** connections, which are typically the response to legitimate incoming connections. This command will allow that:

```
$ sudo iptables -A OUTPUT -m conntrack --ctstate ESTABLISHED -j ACCEPT
```

Internal to External

Assuming `eth0` is your external network, and `eth1` is your internal network, this will allow your internal to access the external:

```
$ sudo iptables -A FORWARD -i eth1 -o eth0 -j ACCEPT
```

Drop Invalid Packets

Some network traffic packets get marked as **invalid**. Sometimes it can be useful to log this type of packet but often it is fine to drop them. Do so with this command:

```
$ sudo iptables -A INPUT -m conntrack --ctstate INVALID -j DROP
```

Block an IP Address

To block network connections that originate from a specific IP address, `15.15.15.51` for example, run this command:

```
$ sudo iptables -A INPUT -s 15.15.15.51 -j DROP
```

In this example, `-s 15.15.15.51` specifies a **source** IP address of "15.15.15.51". The source IP address can be specified in any firewall rule, including an **allow** rule.

If you want to **reject** the connection instead, which will respond to the connection request with a "connection refused" error, replace "DROP" with "REJECT" like this:

```
$ sudo iptables -A INPUT -s 15.15.15.51 -j REJECT
```

Block Connections to a Network Interface

To block connections from a specific IP address, e.g. `15.15.15.51`, to a specific network interface, e.g. `eth0`, use this command:

```
$ iptables -A INPUT -i eth0 -s 15.15.15.51 -j DROP
```

This is the same as the previous example, with the addition of `-i eth0`. The network interface can be specified in any firewall rule, and is a great way to limit the rule to a particular network.

Service: SSH

If you're using a cloud server, you will probably want to allow incoming SSH connections (port 22) so you can connect to and manage your server. This section covers how to configure your firewall with various SSH-related rules.

Allow All Incoming SSH

To allow all incoming SSH connections run these commands:

```
$ sudo iptables -A INPUT -p tcp --dport 22 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT
$ sudo iptables -A OUTPUT -p tcp --sport 22 -m conntrack --ctstate ESTABLISHED -j ACCEPT
```

The second command, which allows the outgoing traffic of **established** SSH connections, is only necessary if the **OUTPUT** policy is not set to **ACCEPT**.

Allow Incoming SSH from Specific IP address or subnet

To allow incoming SSH connections from a specific IP address or subnet, specify the source. For example, if you want to allow the entire **15.15.15.0/24** subnet, run these commands:

```
$ sudo iptables -A INPUT -p tcp -s 15.15.15.0/24 --dport 22 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT
$ sudo iptables -A OUTPUT -p tcp --sport 22 -m conntrack --ctstate ESTABLISHED -j ACCEPT
```

The second command, which allows the outgoing traffic of **established** SSH connections, is only necessary if the **OUTPUT** policy is not set to **ACCEPT**.

Allow Outgoing SSH

If your firewall **OUTPUT** policy is not set to **ACCEPT**, and you want to allow outgoing SSH connections—your server initiating an SSH connection to another server—you can run these commands:

```
$ sudo iptables -A OUTPUT -p tcp --dport 22 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT
$ sudo iptables -A INPUT -p tcp --sport 22 -m conntrack --ctstate ESTABLISHED -j ACCEPT
```

Allow Incoming Rsync from Specific IP Address or Subnet

Rsync, which runs on port 873, can be used to transfer files from one computer to another.

To allow incoming rsync connections from a specific IP address or subnet, specify the source IP address and the destination port. For example, if you want to allow the entire **15.15.15.0/24** subnet to be able to rsync to your server, run these commands:

```
$ sudo iptables -A INPUT -p tcp -s 15.15.15.0/24 --dport 873 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT
$ sudo iptables -A OUTPUT -p tcp --sport 873 -m conntrack --ctstate ESTABLISHED -j ACCEPT
```

The second command, which allows the outgoing traffic of **established** rsync connections, is only necessary if the **OUTPUT** policy is not set to **ACCEPT**.

Service: Web Server

Web servers, such as Apache and Nginx, typically listen for requests on port 80 and 443 for HTTP and HTTPS connections, respectively. If your default policy for incoming traffic is set to drop or deny, you will want to create rules that will allow your server to respond to those requests.

Allow All Incoming HTTP

To allow all incoming HTTP (port 80) connections run these commands:

```
$ sudo iptables -A INPUT -p tcp --dport 80 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT
$ sudo iptables -A OUTPUT -p tcp --sport 80 -m conntrack --ctstate ESTABLISHED -j ACCEPT
```

The second command, which allows the outgoing traffic of **established** HTTP connections, is only necessary if the **OUTPUT** policy is not set to **ACCEPT**.

Allow All Incoming HTTPS

To allow all incoming HTTPS (port 443) connections run these commands:

```
$ sudo iptables -A INPUT -p tcp --dport 443 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT
$ sudo iptables -A OUTPUT -p tcp --sport 443 -m conntrack --ctstate ESTABLISHED -j ACCEPT
```

The second command, which allows the outgoing traffic of **established** HTTP connections, is only necessary if the **OUTPUT** policy is not set to **ACCEPT**.

Allow All Incoming HTTP and HTTPS

If you want to allow both HTTP and HTTPS traffic, you can use the **multiport** module to create a rule that allows both ports. To allow all incoming HTTP and HTTPS (port 443) connections run these commands:

```
$ sudo iptables -A INPUT -p tcp -m multiport --dports 80,443 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT
$ sudo iptables -A OUTPUT -p tcp -m multiport --sports 80,443 -m conntrack --ctstate ESTABLISHED -j ACCEPT
```

```
$ sudo iptables -A OUTPUT -p tcp -m multiport --ports 80,443 -m conntrack --ctstate ESTABLISHED -j ACCEPT
```

The second command, which allows the outgoing traffic of **established** HTTP and HTTPS connections, is only necessary if the **OUTPUT** policy is not set to **ACCEPT**.

Service: MySQL

MySQL listens for client connections on port 3306. If your MySQL database server is being used by a client on a remote server, you need to be sure to allow that traffic.

Allow MySQL from Specific IP Address or Subnet

To allow incoming MySQL connections from a specific IP address or subnet, specify the source. For example, if you want to allow the entire **15.15.15.0/24** subnet, run these commands:

```
$ sudo iptables -A INPUT -p tcp -s 15.15.15.0/24 --dport 3306 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT
$ sudo iptables -A OUTPUT -p tcp --sport 3306 -m conntrack --ctstate ESTABLISHED -j ACCEPT
```

The second command, which allows the outgoing traffic of **established** MySQL connections, is only necessary if the **OUTPUT** policy is not set to **ACCEPT**.

Allow MySQL to Specific Network Interface

To allow MySQL connections to a specific network interface—say you have a private network interface **eth1**, for example—use these commands:

```
$ sudo iptables -A INPUT -i eth1 -p tcp --dport 3306 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT
```

```
$ sudo iptables -A INPUT -i eth1 -p tcp --dport 3306 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT
$ sudo iptables -A OUTPUT -o eth1 -p tcp --sport 3306 -m conntrack --ctstate ESTABLISHED -j ACCEPT
```

The second command, which allows the outgoing traffic of **established** MySQL connections, is only necessary if the **OUTPUT** policy is not set to **ACCEPT**.

Service: PostgreSQL

PostgreSQL listens for client connections on port 5432. If your PostgreSQL database server is being used by a client on a remote server, you need to be sure to allow that traffic.

PostgreSQL from Specific IP Address or Subnet

To allow incoming PostgreSQL connections from a specific IP address or subnet, specify the source. For example, if you want to allow the entire **15.15.15.0/24** subnet, run these commands:

```
$ sudo iptables -A INPUT -p tcp -s 15.15.15.0/24 --dport 5432 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT
$ sudo iptables -A OUTPUT -p tcp --sport 5432 -m conntrack --ctstate ESTABLISHED -j ACCEPT
```

The second command, which allows the outgoing traffic of **established** PostgreSQL connections, is only necessary if the **OUTPUT** policy is not set to **ACCEPT**.

Allow PostgreSQL to Specific Network Interface

To allow PostgreSQL connections to a specific network interface—say you have a private network interface **eth1**, for example—use these commands:

```
$ sudo iptables -A INPUT -i eth1 -p tcp --dport 5432 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT
$ sudo iptables -A OUTPUT -o eth1 -p tcp --sport 5432 -m conntrack --ctstate ESTABLISHED -j ACCEPT
```

The second command, which allows the outgoing traffic of **established** PostgreSQL connections, is only necessary if the **OUTPUT** policy is not set to **ACCEPT**.

Service: Mail

Mail servers, such as Sendmail and Postfix, listen on a variety of ports depending on the protocols being used for mail delivery. If you are running a mail server, determine which protocols you are using and allow the appropriate types of traffic. We will also show you how to create a rule to block outgoing SMTP mail.

Block Outgoing SMTP Mail

If your server shouldn't be sending outgoing mail, you may want to block that kind of traffic. To block outgoing SMTP mail, which uses port 25, run this command:

```
$ sudo iptables -A OUTPUT -p tcp --dport 25 -j REJECT
```

This configures iptables to **reject** all outgoing traffic on port 25. If you need to reject a different service by its port number, instead of port 25, simply replace it.

Allow All Incoming SMTP

To allow your server to respond to SMTP connections, port 25, run these commands:

```
$ sudo iptables -A INPUT -p tcp --dport 25 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT
$ sudo iptables -A OUTPUT -p tcp --sport 25 -m conntrack --ctstate ESTABLISHED -j ACCEPT
```

The second command, which allows the outgoing traffic of **established** SMTP connections, is only necessary if the **OUTPUT** policy is not set to **ACCEPT**.

Note: It is common for SMTP servers to use port 587 for outbound mail.

Allow All Incoming IMAP

To allow your server to respond to IMAP connections, port 143, run these commands:

```
$ sudo iptables -A INPUT -p tcp --dport 143 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT
$ sudo iptables -A OUTPUT -p tcp --sport 143 -m conntrack --ctstate ESTABLISHED -j ACCEPT
```

The second command, which allows the outgoing traffic of **established** IMAP connections, is only necessary if the **OUTPUT** policy is not set to **ACCEPT**.

Allow All Incoming IMAPS

To allow your server to respond to IMAPS connections, port 993, run these commands:

```
$ sudo iptables -A INPUT -p tcp --dport 993 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT
$ sudo iptables -A OUTPUT -p tcp --sport 993 -m conntrack --ctstate ESTABLISHED -j ACCEPT
```

The second command, which allows the outgoing traffic of **established** IMAPS connections, is only necessary if the **OUTPUT** policy is not set to **ACCEPT**.

Allow All Incoming POP3

To allow your server to respond to POP3 connections, port 110, run these commands:

```
$ sudo iptables -A INPUT -p tcp --dport 110 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT
$ sudo iptables -A OUTPUT -p tcp --sport 110 -m conntrack --ctstate ESTABLISHED -j ACCEPT
```

The second command, which allows the outgoing traffic of **established** POP3 connections, is only necessary if the **OUTPUT** policy is not set to **ACCEPT**.

Allow All Incoming POP3S

To allow your server to respond to POP3S connections, port 995, run these commands:

```
$ sudo iptables -A INPUT -p tcp --dport 995 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT
$ sudo iptables -A OUTPUT -p tcp --sport 995 -m conntrack --ctstate ESTABLISHED -j ACCEPT
```

The second command, which allows the outgoing traffic of **established** POP3S connections, is only necessary if the **OUTPUT** policy is not set to **ACCEPT**.

Conclusion

That should cover many of the commands that are commonly used when configuring an iptables firewall. Of course, iptables is a very flexible tool so feel free to mix and match the commands with different options to match your specific needs if they aren't covered here.

If you're looking for help determining how your firewall should be set up, check out this tutorial: [How To Choose an Effective Firewall Policy to Secure your Servers.](#)

Good luck!



By [Mitchell Anicas](#)

♡ Upvote (59)

📄 Subscribe

🔗 Share

Related Tutorials

How To Migrate Iptables Firewall Rules to a New Server







How To Set Up a Firewall with UFW on Debian 10

Recommended Steps For New FreeBSD 12.0 Servers

How To Secure Nginx with NAXSI on Ubuntu 16.04

How To Set Up a Firewall with UFW on Debian 9

23 Comments

B *I*      

Leave a comment...

Log In to Comment

^ [Tauseef](#) *August 12, 2015*

3 A very helping tutorial and very well explained. Thanks a lot :)

^ [GregGreen](#) *September 19, 2015*

0 Hello,

Thanks again for your tutorial, as usual they are very good and very clear.

Just a problem I have, I read it on my android phone (Samsung Galaxy Note 4) under chrome and the message box about "Infrastructure as a Newsletter" can't be removed...

^ [manicas](#) MOD September 21, 2015

0 Thanks for the feedback! I'll pass that along.

^ [Trustyama](#) October 29, 2015

0 I appreciate this site. You make a great job !
Thanks !!!

^ [jonathan](#) November 22, 2015

1 Thanks for this really good tutorial! Question: If I block ALL ports except ssh and 80, and even then restrict 80 to only allow incoming connections from Cloudflare's range, I know it'll work. But will it interfere with any of DigitalOcean's management tools like backups and monitoring? Or do they operate at a lower, more direct level?

By the way, I think it would be useful to add to the tutorial the ability to block a range. For example, as I understand it:

Block 15.15.15.x

```
sudo iptables -A INPUT -s 15.15.15.0/24 -j DROP
```

Block 15.15.x.x

```
sudo iptables -A INPUT -s 15.15.0.0/16 -j DROP
```

Block entire 15.x.x.x (use with caution!)

```
sudo iptables -A INPUT -s 15.0.0.0/8 -j DROP
```

Someone correct me if I'm wrong, but I've been using /16 to block ranges of baddies with success in the past!

^ [manicas](#) MOD November 23, 2015

- 3 Your firewall settings won't interfere with DigitalOcean backups or monitoring. DigitalOcean's backups and monitoring access Droplets at the hypervisor level, which does not rely on a Droplet's networking interfaces.

^ [adithyabenny](#) December 30, 2015

- 1 Hey, correct me if I'm wrong, but in the section 'Allow All Incoming HTTP and HTTPS', shouldn't the second firewall rule be:

```
sudo iptables -A OUTPUT -p tcp -m multiport --sports 80,443 -m conntrack --ctstate ESTABLISHED -j ACCEPT
```

as opposed to

```
sudo iptables -A OUTPUT -p tcp -m multiport --dports 80,443 -m conntrack --ctstate ESTABLISHED -j ACCEPT
```

?

^ [dienteperro](#) October 12, 2017

- 0 I think you're right. Should be --sports instead of --dports.

^ [lendog](#) February 18, 2016

¹ Hi, great article - well written and easy to follow as always. Small note - when i was testing out the rules for POP3,

```
sudo iptables -A INPUT -i -p tcp --dport 110 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT
```

i encountered the error

"Bad argument `tcp' "

The problem seems to lie with the "-i" after "INPUT". You might want to double check that command - what worked for me is

```
sudo iptables -A INPUT -p tcp --dport 110 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT
```

^ [richbhanover](#) June 2, 2016

- 0 This is a *great* tutorial series! Not only is it well written, but it seems to be the only one on the net that explains the basics, and gives the steps necessary to succeed.

But... I followed the instructions above for iptables & iptables-persistent on Ubuntu 15.10, I get this error:

```
$ sudo invoke-rc.d iptables-persistent save
/usr/sbin/invoke-rc.d: 631: /usr/sbin/invoke-rc.d: /etc/init.d/iptables-persistent: not found
invoke-rc.d: initscript iptables-persistent, action "save" failed.
```

Any ideas what to do about that? Thanks!

^ [roblindman](#) January 5, 2017

- 0 I have this same problem, I am on Ubuntu 16, and I cannot find anything that gives me an answer. I think I might have to read the man page or something.

^ [roblindman](#) January 5, 2017

- 0 <http://dev-notes.eu/2016/08/persistent-iptables-rules-in-ubuntu-16-04-xenial-xerus/> seems to help.

^ [jaingarima88](#) September 6, 2016

- 0 Is there a way to allow SFTP from specific IP but block ssh for all incoming traffic at port 22. Traffic for SSH is anyways moved to a different port but I want SFTP to run for specific IP's on port 22.

^ [kpendic](#) September 19, 2016

- 0 Hi, I also have ssh running on diff port then default, and I use native ubuntu nautilus to sftp on droplet,, but that port has to be the same as `Port xx` configured in `sshd_config` file,, that's the same protocol.. hth, k

^ [aubin](#) September 27, 2016

- 0 Hi
Just a quick addition to this...

You can use the invoke method to save the iptables rules or this:

```
service iptables save
```

Tested on centOS7 and Ubuntu14.04

^ [bumblingben](#) *April 20, 2017*

0 I'm loving the tutorial, everything has worked first time around using a Raspberry PI & headless Raspian.

I was wondering if you could help me with the IPTables rules though, I'm trying to add a second VPN connection (tun1), this connection will be accepting incoming connections so I can access my network when away from home. Could you provide some guidance on how I could add this functionality?

^ [tjgrst](#) *August 5, 2017*

0 If you're using a Ubuntu 16.04 server and you're having trouble with the iptables save command, see this answer:
<https://askubuntu.com/a/373526/586708>

^ [denadaisicari](#) *February 19, 2018*

0 Hi! Great post!! Can you explain bit more of this arguments: -m conntrack --ctstate NEW,ESTABLISHED at ALLOW SSH CONNECTIONS?
Thanks!

^ [happy218814](#) *March 23, 2018*

0 It does not block anything. There is no default drop policy.
You can set this:
`sudo iptables -P INPUT DROP` => gives a default drop policy (Chain INPUT (policy DROP)).
Add the other rules first! Otherwise you block yourself from the server.

or:

`sudo iptables -A INPUT -j DROP`
at the end of the ACCEPT chain.

^ [birdmarketing](#) *July 26, 2018*

0 Good article, might be an idea to add some common payment gateway IP addresses to the whitelist so callback APIs are whitelisted. E.g. SagePay have the following IP range: 195.170.169.0/255.

^ [ainurvaliev](#) *March 28, 2019*

0 it works! Thank you!

^ [zackinma](#) *April 1, 2019*

0 Awesome.

^ [superNUTNUT1](#) *May 28, 2019*

0 thank you for your tutorial thanks to you I could create this code

```
#!/bin/bash

### NAT ###
iptables -F && iptables -t nat -F
iptables -t nat -A PREROUTING -d 10.1.31.116/24 -p UDP --dport 53 -j DNAT --to-destination 10.101.10.130
iptables -t nat -A PREROUTING -d 10.1.31.116/24 -p TCP --dport 80 -j DNAT --to-destination 10.101.10.20
iptables -t nat -A POSTROUTING -s 10.101.10.128/25 -o enp0s9 -j SNAT --to-source 10.1.31.116

### INPUT ###
iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
iptables -A INPUT -i lo -j ACCEPT
iptables -A INPUT -j REJECT

### OUTPUT ###
iptables -A OUTPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
iptables -A OUTPUT -o lo -j ACCEPT
iptables -A OUTPUT -j REJECT

### FORWARD ###
iptables -A FORWARD -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT

#DMZ peut faire des requetes DNS
```

```
iptables -A FORWARD -s 10.101.10.0/25 -p udp --dport 53 -d 10.101.10.130 -j ACCEPT
```

#Local vers DMZ et internet

```
iptables -A FORWARD -s 10.101.10.128/25 -d 10.101.10.0/25 -j ACCEPT
```

```
iptables -A FORWARD -s 10.101.10.128/25 -o enp0s9 -j ACCEPT
```

connexion internet vers DMZ

```
iptables -A FORWARD -i enp0s9 -d 10.101.10.130 -p udp --dport 53 -j ACCEPT
```

```
iptables -A FORWARD -i enp0s9 -d 10.101.10.20 -p tcp --dport 80 -j ACCEPT
```

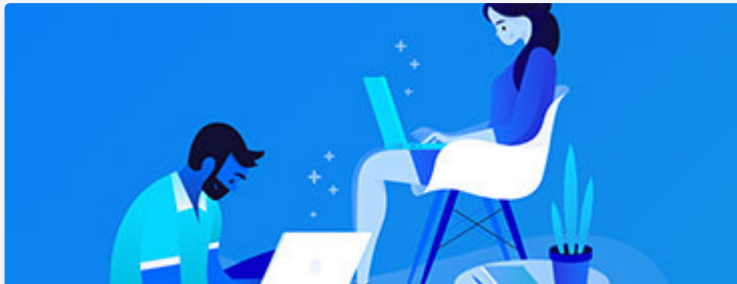
```
iptables -A FORWARD -j REJECT
```

```
iptables -A FORWARD -p icmp --icmp-type 0 -j ACCEPT #request
```

```
iptables -A FORWARD -p icmp --icmp-type 8 -j ACCEPT #reply
```

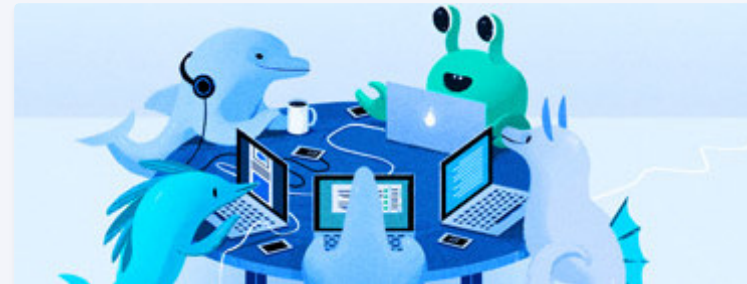


This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.



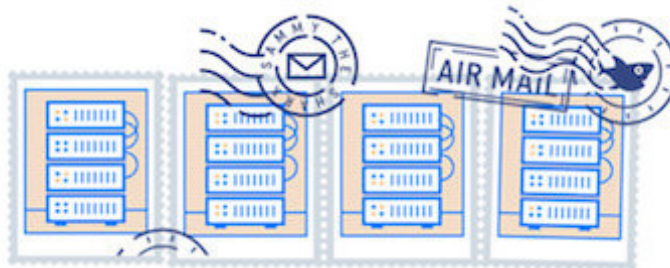
BECOME A CONTRIBUTOR

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](#)

[Intro to Kubernetes](#)

[Learn Python 3](#)

[Machine Learning in Python](#)

[Getting started with Go](#)

[Migrate Node.js to Kubernetes](#)

[DigitalOcean Products](#)

[Droplets](#)

[Managed Databases](#)

[Managed Kubernetes](#)

[Spaces Object Storage](#)

[Marketplace](#)

Welcome to the developer cloud

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](#)



© 2019 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

[Tutorials](#)
[Q&A](#)
[Projects and Integrations](#)
[Tags](#)
[Product Ideas](#)
[Meetups](#)
[Write for DOnations](#)
[Droplets for Demos](#)
[Hatch Startup Program](#)
[Shop Swag](#)
[Research Program](#)
[Currents Research](#)
[Open Source](#)

Contact

[Support](#)
[Sales](#)
[Report Abuse](#)
[System Status](#)