

// Tutorial //

# Iptables Essentials: Common Firewall Rules and Commands

Published on August 10, 2015 · Updated on July 9, 2021

Ubuntu Firewall



By [Mitchell Anicas](#)

Developer and author at DigitalOcean.



## Introduction

Iptables is a software firewall for Linux distributions. This cheat sheet-style guide provides a quick reference to iptables commands that will create firewall rules that 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

- 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 inbound traffic
- 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 highlighted values with your own

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 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 a web-based console to fix your access. If you're using DigitalOcean, you can read [our Recovery Console product documentation](#) for more information. 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.

On Ubuntu, one way to save `iptables` rules is to use the `iptables-persistent` package. Install it with `apt` like this:

```
sudo apt install iptables-persistent
```

Copy

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
```

Copy

Other Linux distributions may have alternate ways of making your `iptables` changes permanent. Please refer to the relevant documentation for more information.

# 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.

### Allowing 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
```

Copy

### Allowing 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 for outgoing connections initiated by the server itself. This command will allow that:

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

Copy

### Allowing 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
```

Copy

### Allowing Internal Network to access 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
```

Copy

## Dropping 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
```

Copy

## Blocking an IP Address

To block network connections that originate from a specific IP address, `203.0.113.51` for example, run this command:

```
sudo iptables -A INPUT -s 203.0.113.51 -j DROP
```

Copy

In this example, `-s 203.0.113.51` specifies a **source** IP address of “203.0.113.51”. 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 203.0.113.51 -j REJECT
```

Copy

## Blocking Connections to a Network Interface

To block connections from a specific IP address, e.g. `203.0.113.51`, to a specific network interface, e.g. `eth0`, use this command:

```
iptables -A INPUT -i eth0 -s 203.0.113.51 -j DROP
```

Copy

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 server without a local console, 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.

## Allowing 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`.

## Allowing 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 `203.0.113.0/24` subnet, run these commands:

```
sudo iptables -A INPUT -p tcp -s 203.0.113.0/24 --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`.

## Allowing 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
```

## Allowing 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 `203.0.113.0/24` subnet to be able to rsync to your server, run these commands:

```
sudo iptables -A INPUT -p tcp -s 203.0.113.0/24 --dport 873 -m conntrack --ctstate NEW,ESTAB
sudo iptables -A OUTPUT -p tcp --sport 873 -m conntrack --ctstate ESTABLISHED -j ACCEPT
```

Copy

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.

### Allowing 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
```

Copy

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

### Allowing 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
```

Copy

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

### Allowing 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
```

Copy

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.

### Allowing 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 `203.0.113.0/24` subnet, run these commands:

```
sudo iptables -A INPUT -p tcp -s 203.0.113.0/24 --dport 3306 -m conntrack --ctstate Copy [A
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`.

### Allowing 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,ESTABL Copy -j
sudo iptables -A OUTPUT -o eth1 -p tcp --sport 3306 -m conntrack --ctstate ESTABLISHED -j AC
```



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 `203.0.113.0/24` subnet, run these commands:



```
sudo iptables -A INPUT -p tcp -s 203.0.113.0/24 --dport 5432 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT
sudo iptables -A OUTPUT -p tcp --sport 5432 -m conntrack --ctstate ESTABLISHED -j ACCEPT
```

Copy

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

## Allowing 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
```

Copy

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.

### Blocking 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
```

Copy

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, substitute that port number for the `25` above.

### Allowing All Incoming SMTP

To allow your server to respond to SMTP connections on 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
```

Copy



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

## Allowing 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 / Copy
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`.

## Allowing 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 / Copy
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`.

## Allowing 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 / Copy
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`.

## Allowing 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 / Copy
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](#).