# Configure Linux `iptables` Firewall for MongoDB



On contemporary Linux systems, the `iptables` program provides methods for managing the Linux Kernel's `netfilter` or network packet filtering capabilities. These firewall rules make it possible for administrators to control what hosts can connect to the system, and limit risk exposure by limiting the hosts that can connect to a system.

This document outlines basic firewall configurations for `iptables` firewalls on Linux. Use these approaches as a starting point for your larger networking organization. For a detailed overview of security practices and risk management for MongoDB, see Security.

## Overview

Rules in `iptables` configurations fall into chains, which describe the process for filtering and processing specific streams of traffic. Chains have an order, and packets must pass through earlier rules in a chain to reach later rules. This document addresses only the following two chains:

**`INPUT`**
: Controls all incoming traffic.

**`OUTPUT`**
: Controls all outgoing traffic.

Given the default ports of all MongoDB processes, you must configure networking rules that permit *only* required communication between your application and the appropriate `mongod` and `mongos` instances.

Be aware that, by default, the default policy of `iptables` is to allow all connections and traffic unless explicitly disabled. The configuration changes outlined in this document will create rules that explicitly allow traffic from specific addresses and on specific ports, using a default policy that drops all traffic that is not

explicitly allowed. When you have properly configured your `iptables` rules to allow only the traffic that you want to permit, you can Change Default Policy to `DROP`.

Share Feedback

## Patterns

This section contains a number of patterns and examples for configuring `iptables` for use with MongoDB deployments. If you have configured different ports using the `port` configuration setting, you will need to modify the rules accordingly.

### Traffic to and from `mongod` Instances

This pattern is applicable to all `mongod` instances running as standalone instances or as part of a replica set.

The goal of this pattern is to explicitly allow traffic to the `mongod` instance from the application server. In the following examples, replace `<ip-address>` with the IP address of the application server:

```
iptables -A INPUT -s <ip-address> -p tcp --destina
iptables -A OUTPUT -d <ip-address> -p tcp --source
```

The first rule allows all incoming traffic from `<ip-address>` on port `27017`, which allows the application server to connect to the `mongod` instance. The second rule, allows outgoing traffic from the `mongod` to reach the application server.

> **NOTE**
>
> **Optional**
>
> If you have only one application server, you can replace `<ip-address>` with either the IP address itself, such as: `198.51.100.55`. You can also express this using CIDR notation as `198.51.100.55/32`. If you want to permit a larger block of possible IP addresses you can allow traffic from a `/24` using one of the following specifications for the `<ip-address>`, as follows:
>
> ```
> 10.10.10.10/24
> 10.10.10.10/255.255.255.0
> ```

### Traffic to and from `mongos` Instances

`mongos` instances provide query routing for sharded clusters. Clients connect to `mongos` instances, which behave from the client's perspective as `mongod` instances. In turn, the `mongos` connects to all `mongod` instances that are components of the sharded cluster.

Use the same `iptables` command to allow traffic to and from these instances as you would from the `mongod` instances that are members of the replica set. Take the configuration outlined in the Traffic to and from `mongod` Instances section as an example.

Share Feedback

### Traffic to and from a MongoDB Config Server

Config servers host the config database that stores metadata for sharded clusters. Config servers listen for connections on port `27019`. As a result, add the following `iptables` rules to the config server to allow incoming and outgoing connection on port `27019`, for connection to the other config servers.

```
iptables -A INPUT -s <ip-address> -p tcp --destina
iptables -A OUTPUT -d <ip-address> -p tcp --source
```

Replace `<ip-address>` with the address or address space of *all* the `mongod` that provide config servers.

Additionally, config servers need to allow incoming connections from all of the `mongos` instances in the cluster *and* all `mongod` instances in the cluster. Add rules that resemble the following:

```
iptables -A INPUT -s <ip-address> -p tcp --destina
```

Replace `<ip-address>` with the address of the `mongos` instances and the shard `mongod` instances.

### Traffic to and from a MongoDB Shard Server

Shard servers default to port number `27018`. You must configure the following `iptables` rules to allow traffic to and from each shard:

```
iptables -A INPUT -s <ip-address> -p tcp --destina
iptables -A OUTPUT -d <ip-address> -p tcp --source
```

Replace the `<ip-address>` specification with the IP address of all `mongod`. This allows you to permit incoming and outgoing traffic between all shards including constituent replica set members, to:

- all `mongod` instances in the shard's replica sets.
- all `mongod` instances in other shards. [1]

Furthermore, shards need to be able make outgoing connections to:

- all `mongod` instances in the config servers.

Create a rule that resembles the following, and replace the `<ip-address>` with the address of the config servers and the `mongos` instances:

Share Feedback

```
iptables -A OUTPUT -d <ip-address> -p tcp --source
```

[1] All shards in a cluster need to be able to communicate with all other shards to facilitate chunk and balancing operations.

#### Provide Access For Monitoring Systems

The `mongostat` diagnostic tool, when running with the `--discover` needs to be able to reach all components of a cluster, including the config servers, the shard servers, and the `mongos` instances.

*Changed in version 3.6*: MongoDB 3.6 removes the deprecated HTTP interface and REST API to MongoDB.

### Change Default Policy to `DROP`

The default policy for `iptables` chains is to allow all traffic. After completing all `iptables` configuration changes, you *must* change the default policy to `DROP` so that all traffic that isn't explicitly allowed as above will not be able to reach components of the MongoDB deployment. Issue the following commands to change this policy:

```
iptables -P INPUT DROP

iptables -P OUTPUT DROP
```

### Manage and Maintain `iptables` Configuration

This section contains a number of basic operations for managing and using `iptables`. There are various front end tools that automate some aspects of `iptables` configuration, but at the core all `iptables` front ends provide the same basic functionality:

#### Make all `iptables` Rules Persistent

By default all `iptables` rules are only stored in memory. When your system restarts, your firewall rules will revert to their defaults. When you have tested a rule set and have guaranteed that it effectively controls traffic you can use the following operations to you should make the rule set persistent.

On Red Hat Enterprise Linux, Fedora Linux, and related distributions you can issue the following command:

```
service iptables save
```

On Debian, Ubuntu, and related distributions, you can use the following command to dump the `iptables` rules to the `/etc/iptables.conf` file:

```
iptables-save > /etc/iptables.conf
```

Share Feedback

Run the following operation to restore the network rules:

```
iptables-restore < /etc/iptables.conf
```

Place this command in your `rc.local` file, or in the `/etc/network/if-up.d/iptables` file with other similar operations.

### List all `iptables` Rules

To list all of currently applied `iptables` rules, use the following operation at the system shell.

```
iptables -L
```

### Flush all `iptables` Rules

If you make a configuration mistake when entering `iptables` rules or simply need to revert to the default rule set, you can use the following operation at the system shell to flush all rules:

```
iptables -F
```

If you've already made your `iptables` rules persistent, you will need to repeat the appropriate procedure in the Make all `iptables` Rules Persistent section.

About

Careers
Investor Relations
Legal Notices
Privacy Notices
Security Information
Trust Center

Support

Contact Us
Customer Portal
Atlas Status
Paid Support

Social

Github

Stack Overflow

Share Feedback

LinkedIn

Youtube

Twitter

Twitch

Facebook

© 2023 MongoDB, Inc.