


How to Configure the Linux Firewall for Docker Swarm on CentOS 7



Posted January 11, 2017  49.7k

FIREWALL

DOCKER

SECURITY

UBUNTU 16.04

By: finid

Introduction

Docker Swarm is a feature of Docker that makes it easy to run Docker hosts and containers at scale. A Docker Swarm, or Docker cluster, is made up of one or more Dockerized hosts that function as *manager* nodes, and any number of *worker* nodes. Setting up such a system requires careful manipulation of the Linux firewall.

The network ports required for a Docker Swarm to function properly are:

- TCP port 2376 for secure Docker client communication. This port is required for Docker Machine to work. Docker Machine is used to orchestrate Docker hosts.
- TCP port 2377. This port is used for communication between the nodes of a Docker Swarm or cluster. It only needs to be opened on manager nodes.
- TCP and UDP port 7946 for communication among nodes (container network discovery).
- UDP port 4789 for overlay network traffic (container ingress networking).

Note: Aside from those ports, port 22 (for SSH traffic) and any other ports needed for specific services to run on the cluster have to be open.

In this article, you'll configure the Linux firewall on CentOS 7 using FirewallD and IPTables. FirewallD is the default firewall application on CentOS 7, but IPTables is also available. While this tutorial covers both methods, each one delivers the same outcome, so you can choose the one you are most familiar with.

Prerequisites

Before proceeding with this article, you should:

- Set up the hosts that make up your cluster, including at least one swarm manager and one swarm worker. You can follow the tutorial [How To Provision and Manage Remote Docker Hosts with Docker Machine on CentOS 7](#) to set these up.

Note: You'll notice that the commands (and all the commands in this article) are not prefixed with `sudo`. That's because it's assumed that you're logged into the server using the `docker-machine ssh` command after provisioning it using Docker Machine.

Method 1 — Open Docker Swarm Ports Using FirewallD

Firewalld is the default firewall application on CentOS 7, but on a new CentOS 7 server, it is disabled out of the box. So let's enable it and add the network ports necessary for Docker Swarm to function.

Before starting, verify its status:

```
$ systemctl status firewalld
```

It should not be running, so start it:

```
$ systemctl start firewalld
```

Then enable it so that it starts on boot:

```
$ systemctl enable firewalld
```

On the node that will be a Swarm manager, use the following commands to open the necessary ports:

```
$ firewall-cmd --add-port=2376/tcp --permanent
$ firewall-cmd --add-port=2377/tcp --permanent
$ firewall-cmd --add-port=7946/tcp --permanent
$ firewall-cmd --add-port=7946/udp --permanent
$ firewall-cmd --add-port=4789/udp --permanent
```

Note: If you make a mistake and need to remove an entry, type:

```
firewall-cmd --remove-port=port-number/tcp --permanent .
```

Afterwards, reload the firewall:

```
$ firewall-cmd --reload
```

Then restart Docker.

```
$ systemctl restart docker
```

Then on each node that will function as a Swarm worker, execute the following commands:

```
$ firewall-cmd --add-port=2376/tcp --permanent
$ firewall-cmd --add-port=7946/tcp --permanent
$ firewall-cmd --add-port=7946/udp --permanent
$ firewall-cmd --add-port=4789/udp --permanent
```

Afterwards, reload the firewall:

```
$ firewall-cmd --reload
```

Then restart Docker.

```
$ systemctl restart docker
```

You've successfully used FirewallD to open the necessary ports for Docker Swarm.

Note: If you'll be testing applications on the cluster that require outside network access, be sure to open the necessary ports. For example, if you'll be testing a Web application that requires access on port 80, add a rule that grants access to that port using the following command on all the nodes (managers and workers) in the cluster:

```
$ firewall-cmd --add-port=80/tcp --permanent
```

Remember to reload the firewall when you make this change.

Method 2 — Open Docker Swarm Ports Using IPTables

To use IPTables on any Linux distribution, you'll have to first uninstall any other firewall utilities. To switch to IPTables from FirewallD, first stop FirewallD:

```
$ systemctl stop firewalld
```

Then disable it

```
$ systemctl disable firewalld
```

Then install the `iptables-services` package, which manages the automatic loading of IPTables rules:

```
$ yum install iptables-services
```

Next, start IPTables:

```
$ systemctl start iptables
```

Then enable it so that it automatically starts on boot:

```
$ systemctl enable iptables
```

Before you start adding Docker Swarm-specific rules to the INPUT chain, let's take a look at the default rules in that chain:

```
$ iptables -L INPUT --line-numbers
```

The output should look exactly like this:

Output

Chain INPUT (policy ACCEPT)

num	target	prot	opt	source	destination	
1	ACCEPT	all	--	anywhere	anywhere	state RELATED,ESTABLISHED
2	ACCEPT	icmp	--	anywhere	anywhere	
3	ACCEPT	all	--	anywhere	anywhere	
4	ACCEPT	tcp	--	anywhere	anywhere	state NEW tcp dpt:ssh
5	REJECT	all	--	anywhere	anywhere	reject-with icmp-host-prohibited

Taken together, the default rules provide stateful protection for the server, denying all input traffic except those that are already established. SSH traffic is allowed in. Pay attention to rule number 5, highlighted above, because it's a catchall reject rule. For your Docker Swarm to function properly, the rules you add need to be added *above* this rule. That means the new rules need to be inserted, instead of appended to the INPUT chain.

Now that you know what to do, you can add the rules you need by using the `iptables` utility. This first set of commands should be executed on the nodes that will serve as Swarm managers.

```
$ iptables -I INPUT 5 -p tcp --dport 2376 -j ACCEPT
$ iptables -I INPUT 6 -p tcp --dport 2377 -j ACCEPT
$ iptables -I INPUT 7 -p tcp --dport 7946 -j ACCEPT
$ iptables -I INPUT 8 -p udp --dport 7946 -j ACCEPT
$ iptables -I INPUT 9 -p udp --dport 4789 -j ACCEPT
```

Those rules are runtime rules and will be lost if the system is rebooted. To save the current runtime rules to a file so that they persist after a reboot, type:

```
$ /usr/libexec/iptables/iptables.init save
```

The rules are now saved to a file called `iptables` in the `/etc/sysconfig` directory. And if you view the rules using `iptables -L --line-numbers`, you'll see that all the rules have been inserted above the catch-all reject rule:

Output

```
Chain INPUT (policy ACCEPT)
num  target      prot opt source                destination              state
1    ACCEPT      all  --  anywhere              anywhere                  state RELATED,ESTABLISHED
2    ACCEPT      icmp --  anywhere              anywhere
3    ACCEPT      all  --  anywhere              anywhere
4    ACCEPT      tcp  --  anywhere              anywhere                  state NEW tcp dpt:ssh
5    ACCEPT      tcp  --  anywhere              anywhere                  tcp dpt:2376
6    ACCEPT      tcp  --  anywhere              anywhere                  tcp dpt:7946
7    ACCEPT      udp  --  anywhere              anywhere                  udp dpt:7946
8    ACCEPT      udp  --  anywhere              anywhere                  udp dpt:4789
9    ACCEPT      tcp  --  anywhere              anywhere                  tcp dpt:http
10   REJECT      all  --  anywhere              anywhere                  reject-with icmp-host-prohibited
```

Then restart Docker.

Output

```
$ systemctl restart docker
```

On the nodes that will function as Swarm workers, execute these commands:

```
$ iptables -I INPUT 5 -p tcp --dport 2376 -j ACCEPT
$ iptables -I INPUT 6 -p tcp --dport 7946 -j ACCEPT
$ iptables -I INPUT 7 -p udp --dport 7946 -j ACCEPT
$ iptables -I INPUT 8 -p udp --dport 4789 -j ACCEPT
```

Save the rules to disk:

```
$ /usr/libexec/iptables/iptables.init save
```

Then restart Docker:

```
$ systemctl restart docker
```

That's all it takes to open the necessary ports for Docker Swarm using IPTables. You can learn more about how these rules work in the tutorial [How the IPTables Firewall Works](#).

Note: If you'll be testing applications on the cluster that requires outside network access, be sure to open the necessary ports. For example, if you'll be testing a Web application that requires access on port 80, add a rule that grants access to that port using the following command on all the nodes (manager and workers) in the cluster:

```
$ iptables -I INPUT rule-number -p tcp --dport 80 -j ACCEPT
```

Be sure to insert the rule above the catchall reject rule.

Conclusion

Firewalld and IPTables are two of the most popular firewall management applications in the Linux world. You just read how to use these to open the network ports needed to set up Docker Swarm. The method you use is just a matter of personal preference, because they are all equally capable.

By: finid

 Upvote (5)  Subscribe  Share



Editor:
Brian Hogan



We just made it easier for you to deploy faster.

[TRY FREE](#)

Related Tutorials

[How to Manually Set Up a Prisma Server on Ubuntu 18.04](#)

[How To Use Traefik as a Reverse Proxy for Docker Containers on Debian 9](#)

[How To Set Up a Private Docker Registry on Ubuntu 18.04](#)

[How To Secure a Containerized Node.js Application with Nginx, Let's Encrypt, and Docker Compose](#)



[How To Set Up Multi-Node Deployments With Rancher 2.1, Kubernetes, and Docker Machine on Ubu...](#)

3 Comments

Leave a comment...



[Log In to Comment](#)

 [asuero](#) *March 14, 2017*

  Code snippet for opening this ports with ufw.

```
ufw allow 2376/tcp
ufw allow 2377/tcp
ufw allow 7946
ufw allow 4789/udp
```

 [riqra](#) *June 28, 2017*

  I can't connect to the remote docker daemon from my local machine. I opened the 2376/tcp port in the remote one, I am getting

```
cannot connect to the Docker daemon at Is the docker daemon running?
```

has someone dealt with it?

^ [AhmedKamel](#) November 29, 2017

0 There is a shortcut to open firewall ports faster without the need to even reload the daemon

```
firewall-cmd --add-port=2376/tcp --add-port=2377/tcp --add-port=7946/tcp --add-port=7946/udp  
firewall-cmd --add-port=2376/tcp --add-port=2377/tcp --add-port=7946/tcp --add-port=7946/udp
```



That will do it.



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



Copyright © 2019 DigitalOcean™ Inc.

[Community](#) [Tutorials](#) [Questions](#) [Projects](#) [Tags](#) [Newsletter](#) [RSS](#) 

[Distros & One-Click Apps](#) [Terms, Privacy, & Copyright](#) [Security](#) [Report a Bug](#) [Write for DOnations](#) [Shop](#)