# ufw Examples

The following is a list of start-to-finish ufw commands to set up a firewall for a particular service. I assume you are starting from the default state, that is, ufw is disabled. Also, I assume you will probably want to manage your server remotely with SSH even if it’s a Web, mail, or DNS server, so I add rules to enable SSH in each example.

**NOTE About Remote Firewall Management**

If this is the first time you have enabled ufw, and you plan to set this up remotely over the network, you should be careful about your steps. It’s very easy to make a mistake and lock yourself out. Specifically, when ufw is enabled, it flushes all connection data, so if you manage the server over SSH, your connection will be closed, and if you deny by default and haven’t set up an SSH rule yet, when ufw is enabled, you will be locked out. One simple safeguard you can put in place is a cron job that disables ufw every 15 minutes or so. That way, if you make a mistake and lock yourself out, you just have to wait at most 15 minutes for the firewall to be reset. To do this, add the following line to your /etc/crontab file:

\*/15 \* \* \* \* root ufw disable

Of course, the downside to this is that every 15 minutes while you are tweaking your firewall, ufw will be disabled and you will have to remember to enable it. Still, it’s better than being locked out of the system completely. Just remember to delete the crontab rule once you are finished tweaking. On Ubuntu 8.10 and later, ufw actually warns you if you enable ufw while using SSH and prompts you before it enables.

## SSH

I’m assuming you will probably want to have SSH enabled on just about any server you manage. Note the order in which I run the commands here, as I enable ufw at the very end. That way I don’t risk locking myself out because the SSH rule is defined before ufw is enabled:

$ sudo ufw allow ssh

$ sudo ufw default deny

$ sudo ufw enable

$ sudo ufw status

Status: active

To Action From

\_\_ \_\_\_\_\_\_ \_\_\_\_

22 ALLOW Anywhere

## DNS

$ sudo ufw allow ssh

$ sudo ufw allow domain

$ sudo ufw default deny

$ sudo ufw enable

$ sudo ufw status

Status: active

To Action From

\_\_ \_\_\_\_\_\_ \_\_\_\_

22 ALLOW Anywhere

53 ALLOW Anywhere

## Web

Here I open up ports for both HTTP (80) and HTTPS (443), but if you don’t use HTTPS, you can remove that particular rule from the list:

$ sudo ufw allow ssh

$ sudo ufw allow www

$ sudo ufw allow https

$ sudo ufw default deny

$ sudo ufw enable

$ sudo ufw status

Status: active

To Action From

\_\_ \_\_\_\_\_\_ \_\_\_\_

22 ALLOW Anywhere

80 ALLOW Anywhere

443 ALLOW Anywhere

## SMTP

$ sudo ufw allow ssh

$ sudo ufw allow smtp

$ sudo ufw default deny

$ sudo ufw enable

$ sudo ufw status

Status: active

To Action From

\_\_ \_\_\_\_\_\_ \_\_\_\_

22 ALLOW Anywhere

25/tcp ALLOW Anywhere

## POP/IMAP

To simplify things, I list rules to enable POP2, POP3, and POP3 with SSL, IMAP2, IMAP3, and IMAP with SSL, since many administrators end up supporting all of them on the same server.

$ sudo ufw allow ssh

$ sudo ufw allow pop2

$ sudo ufw allow pop3

$ sudo ufw allow pop3s

$ sudo ufw allow imap2

$ sudo ufw allow imap3

$ sudo ufw allow imaps

$ sudo ufw default deny

$ sudo ufw enable

$ sudo ufw status

Status: active

To Action From

\_\_ \_\_\_\_\_\_ \_\_\_\_

22 ALLOW Anywhere

109 ALLOW Anywhere

110 ALLOW Anywhere

995 ALLOW Anywhere

143 ALLOW Anywhere

220 ALLOW Anywhere

993 ALLOW Anywhere

## MySQL

This example uses the default MySQL ports. Of course, if you have moved MySQL to listen on a different port, you will have to manually specify the port to open.

$ sudo ufw allow ssh

$ sudo ufw allow mysql

$ sudo ufw default deny

$ sudo ufw enable

$ sudo ufw status

Status: active

To Action From

\_\_ \_\_\_\_\_\_ \_\_\_\_

22 ALLOW Anywhere

3306 ALLOW Anywhere

## PostgreSQL

This example uses the default PostgreSQL ports. Of course, if you have moved PostgreSQL to listen on a different port, you will have to manually specify the port to open.

$ sudo ufw allow ssh

$ sudo ufw allow postgresql

$ sudo ufw default deny

$ sudo ufw enable

$ sudo ufw status

Status: active

To Action From

\_\_ \_\_\_\_\_\_ \_\_\_\_

22 ALLOW Anywhere

5432 ALLOW Anywhere

## Samba

Samba is a little trickier to open because it listens on a set of ports and none of them are labeled in /etc/services with “Samba.”

$ sudo ufw allow ssh

$ sudo ufw allow netbios-ns

$ sudo ufw allow netbios-dgm

$ sudo ufw allow netbios-ssn

$ sudo ufw default deny

$ sudo ufw enable

$ sudo ufw status

Status: active

To Action From

\_\_ \_\_\_\_\_\_ \_\_\_\_

22 ALLOW Anywhere

137 ALLOW Anywhere

138 ALLOW Anywhere

139 ALLOW Anywhere

## NFS

NFS is a little trickier to firewall off than most other services because the connections don’t necessarily use a defined set of ports. As a result, it can be difficult to open up a range of ports for NFS that will work long term.

The simplest solution, if you want to enable a firewall on an NFS server, is to deny by default and then allow access to all ports from specific NFS clients. If you don’t want to add a firewall rule for each individual host because there are many, you might consider putting all NFS clients on their own subnet and then allowing that subnet. I show two examples. The first allows all access from the 10.1.1.7, 10.1.1.8, and 10.1.1.9 hosts. The second example opens up access for the entire 10.1.2.0/24 subnet:

$ sudo ufw allow ssh

$ sudo ufw allow from 10.1.1.7

$ sudo ufw allow from 10.1.1.8

$ sudo ufw allow from 10.1.1.9

$ sudo ufw default deny

$ sudo ufw enable

$ sudo ufw status

Status: active

To Action From

\_\_ \_\_\_\_\_\_ \_\_\_\_

22 ALLOW Anywhere

Anywhere ALLOW 10.1.1.7

Anywhere ALLOW 10.1.1.8

Anywhere ALLOW 10.1.1.9

Here are the steps to allow all of 10.1.2.0/24 access to NFS:

$ sudo ufw allow ssh

$ sudo ufw allow from 10.1.2.0/24

$ sudo ufw default deny

$ sudo ufw enable

$ sudo ufw status

Status: active

To Action From

\_\_ \_\_\_\_\_\_ \_\_\_\_

22 ALLOW Anywhere

Anywhere ALLOW 10.1.2.0/24