CONTENTS

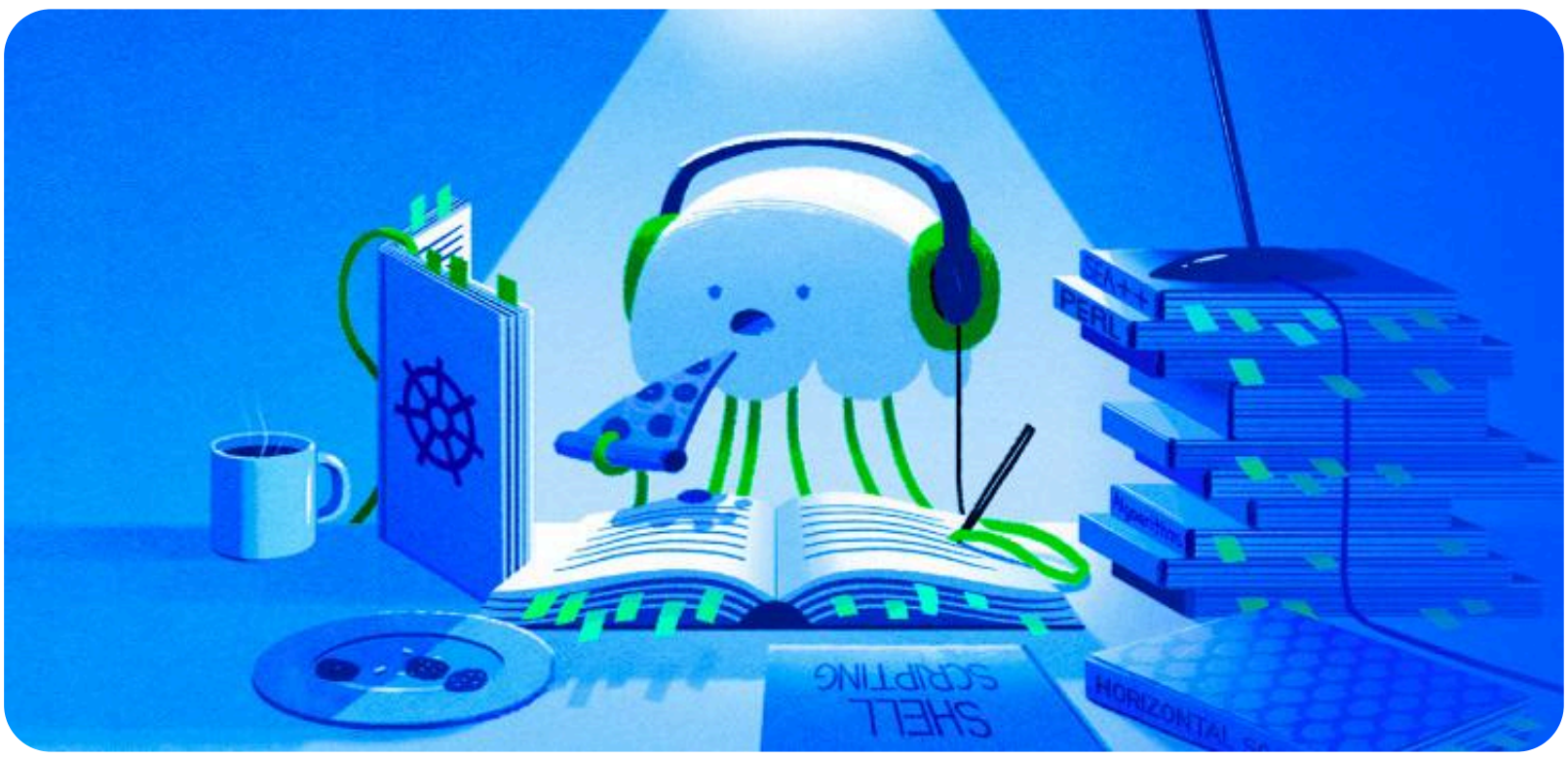
// TUTORIAL //

How To Secure Nginx with Let's Encrypt on Debian 11

Published on September 2, 2022

Nginx Security Ubuntu Ubuntu 22.04

Alex Garnett
Senior DevOps Technical Writer



Not using Debian 11?
Choose a different version or distribution.

Debian 11

Introduction

Let's Encrypt is a Certificate Authority (CA) that provides an easy way to obtain and install free TLS/SSL certificates, thereby enabling encrypted HTTPS on web servers. It simplifies the process by providing a software client, Certbot, that attempts to automate most (if not all) of the required steps. Currently, the entire process of obtaining and installing a certificate is fully automated on both Apache and Nginx.

In this tutorial, you will use Certbot to obtain a free SSL certificate for Nginx on Debian 11 and set up your certificate to renew automatically.

This tutorial will use a separate Nginx server configuration file instead of the default file. We recommend creating new Nginx server block files for each domain because it helps to avoid common mistakes and maintains the default files as a fallback configuration.

Prerequisites

To follow this tutorial, you will need:

- One Debian 11 server set up by following this initial server setup for Debian 11 tutorial, including a sudo-enabled non-**root** user and a firewall.
- A registered domain name. This tutorial will use `example.com` throughout. You can purchase a domain name from Namecheap, get one for free with Freenom, or use the domain registrar of your choice.
- Both of the following DNS records set up for your server. If you are using DigitalOcean, please see our DNS documentation for details on how to add them.
    - An A record with `example.com` pointing to your server's public IP address.
    - An A record with `www.example.com` pointing to your server's public IP address.
- Nginx installed by following How To Install Nginx on Debian 11. Be sure that you have a server block for your domain. This tutorial will use `/etc/nginx/sites-available/example.com` as an example.

Step 1 – Installing Certbot

The first step to using Let's Encrypt to obtain an SSL certificate is to install the Certbot software on your server.

Install Certbot and its Nginx plugin with `apt`:

```
$ sudo apt install certbot python3-certbot-nginx
```

Certbot is now ready to use, but in order for it to automatically configure SSL for Nginx, we need to verify some of Nginx's configuration.

Step 2 – Confirming Nginx's Configuration

Certbot needs to be able to find the correct `server` block in your Nginx configuration for it to be able to automatically configure SSL. Specifically, it does this by looking for a `server_name` directive that matches the domain you request a certificate for.

If you followed the server block set up step in the Nginx installation tutorial, you should have a server block for your domain at `/etc/nginx/sites-available/example.com` with the `server_name` directive already set appropriately.

To check, open the configuration file for your domain using `nano` or your favorite text editor:

```
$ sudo nano /etc/nginx/sites-available/example.com
```

Find the existing `server_name` line. It should look like this:

/etc/nginx/sites-available/example.com Copy

```
...
server_name example.com www.example.com;
...
```

If it does, exit your editor and move on to the next step.

If it doesn't, update it to match. Then save the file, quit your editor, and verify the syntax of your configuration edits:

Copy

```
$ sudo nginx -t
```

If you get an error, reopen the server block file and check for any typos or missing characters. Once your configuration file's syntax is correct, reload Nginx to load the new configuration:

Copy

```
$ sudo systemctl reload nginx
```

Certbot can now find the correct `server` block and update it automatically.

Next, let's update the firewall to allow HTTPS traffic.

Step 3 – Allowing HTTPS Through the Firewall

If you have the `ufw` firewall enabled, as recommended by the prerequisite guides, you'll need to adjust the settings to allow for HTTPS traffic. Luckily, Nginx registers a few profiles with `ufw` upon installation.

You can see the current setting by typing:

Copy

```
$ sudo ufw status
```

It will probably look like this, meaning that only HTTP traffic is allowed to the web server:

```
Output
Status: active

To                         Action      From
--                         ------      ----
OpenSSH                    ALLOW       Anywhere
Nginx HTTP                 ALLOW       Anywhere
OpenSSH (v6)               ALLOW       Anywhere (v6)
Nginx HTTP (v6)            ALLOW       Anywhere (v6)
```

To additionally let in HTTPS traffic, allow the Nginx Full profile and delete the redundant Nginx HTTP profile allowance:

Copy

```
$ sudo ufw allow 'Nginx Full'
$ sudo ufw delete allow 'Nginx HTTP'
```

Your status should now look like this:

```
$ sudo ufw status
```

```
Output
Status: active

To                         Action      From
--                         ------      ----
OpenSSH                    ALLOW       Anywhere
Nginx Full                 ALLOW       Anywhere
OpenSSH (v6)               ALLOW       Anywhere (v6)
Nginx Full (v6)            ALLOW       Anywhere (v6)
```

Next, let's run Certbot and fetch our certificates.

Step 4 – Obtaining an SSL Certificate

Certbot provides a variety of ways to obtain SSL certificates through plugins. The Nginx plugin will take care of reconfiguring Nginx and reloading the config whenever necessary. To use this plugin, type the following:

```
$ sudo certbot --nginx -d example.com -d www.example.com
```

This runs `certbot` with the `--nginx` plugin, using `-d` to specify the domain names we'd like the certificate to be valid for.

If this is your first time running `certbot`, you will be prompted to enter an email address and agree to the terms of service. After doing so, `certbot` will communicate with the Let's Encrypt server, then run a challenge to verify that you control the domain you're requesting a certificate for.

The configuration will be updated, and Nginx will reload to pick up the new settings. `certbot` will wrap up with a message telling you the process was successful and where your certificates are stored:

```
Output
IMPORTANT NOTES:
 - Congratulations! Your certificate and chain have been saved at:
   /etc/letsencrypt/live/ example.com /fullchain.pem
   Your key file has been saved at:
   /etc/letsencrypt/live/ example.com /privkey.pem
   Your cert will expire on 2022-08-08. To obtain a new or tweaked
   version of this certificate in the future, simply run certbot again
   with the "certonly" option. To non-interactively renew *all* of
   your certificates, run "certbot renew"
 - If you like Certbot, please consider supporting our work by:

   Donating to ISRG / Let's Encrypt:   https://letsencrypt.org/donate
   Donating to EFF:                    https://eff.org/donate-le
```

Your certificates are downloaded, installed, and loaded. Try reloading your website using `https://` and notice your browser's security indicator. It should indicate that the site is properly secured, usually with a lock icon. If you test your server using the SSL Labs Server Test, it will get an **A** grade.

Let's finish by testing the renewal process.

Step 5 – Verifying Certbot Auto-Renewal

Let's Encrypt's certificates are only valid for ninety days. This is to encourage users to automate their certificate renewal process. The `certbot` package we installed takes care of this for us by adding a

systemd timer that will run twice a day and automatically renew any certificate that's within thirty days of expiration.

You can query the status of the timer with `systemctl`:

Copy

```
$ sudo systemctl status certbot.timer
```

```
Output
● certbot.timer - Run certbot twice daily
     Loaded: loaded (/lib/systemd/system/certbot.timer; enabled; vendor preset: enabled)
     Active: active (waiting) since Mon 2022-08-08 19:05:35 UTC; 11s ago
    Trigger: Tue 2022-08-09 07:22:51 UTC; 12h left
   Triggers: ● certbot.service
```

To test the renewal process, you can do a dry run with `certbot`:

Copy

```
$ sudo certbot renew --dry-run
```

If you see no errors, you're all set. When necessary, Certbot will renew your certificates and reload Nginx to pick up the changes. If the automated renewal process ever fails, Let's Encrypt will send a message to the email you specified, warning you when your certificate is about to expire.

Conclusion

In this tutorial, you installed the Let's Encrypt client `certbot`, downloaded SSL certificates for your domain, configured Nginx to use these certificates, and set up automatic certificate renewal. If you have further questions about using Certbot, the official documentation is a good place to start.

About the authors

Alex Garnett Author
Senior DevOps Technical Writer

Comments

Leave a comment

Leave a comment...

This textbox defaults to using **Markdown** to format your answer.

You can type **!ref** in this text area to quickly search our full set of tutorials, documentation & marketplace offerings and insert the link!

Sign In or Sign Up to Comment

Popular Topics

AI/ML

Ubuntu

Linux Basics

JavaScript

Python

MySQL

Docker

Kubernetes

All tutorials →

Talk to an expert →

Congratulations on unlocking the whale ambience easter egg!

Thank you to the Glacier Bay National Park & Preserve and Merrick079 for the sounds behind this easter egg.

Click the whale button in the bottom left of your screen to toggle some ambient whale noises while you read.

Reset easter egg to be discovered again

Permanently dismiss and hide easter egg

Interested in whales, protecting them, and their connection to helping prevent climate change? We recommend checking out the Whale and Dolphin Conservation.

Become a contributor for community

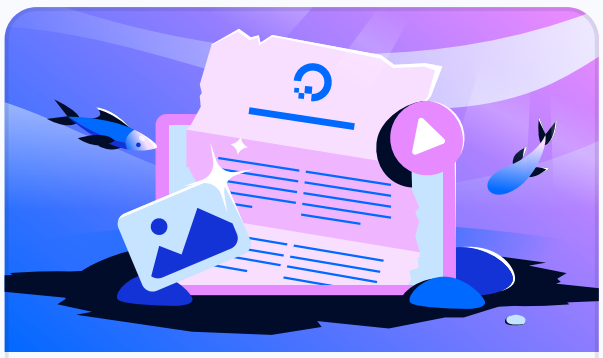
Get paid to write technical tutorials and select a tech-focused charity to receive a matching donation.

Sign Up →

DigitalOcean Documentation

Full documentation for every DigitalOcean product.

Learn more →



Resources for startups and SMBs

The Wave has everything you need to know about building a business, from raising funding to marketing your product.

Learn more →

Get our newsletter

Stay up to date by signing up for DigitalOcean's Infrastructure as a Newsletter.

Email address Submit

New accounts only. By submitting your email you agree to our Privacy Policy

Share your ideas