Community

TUTORIAL

How To Secure Nginx with Let's Encrypt on Ubuntu 20.04

Nginx   Ubuntu   Security   Ubuntu 20.04

By Brian Boucheron

Posted May 20, 2020   ◉ 30.1k

English ⌄

Not using Ubuntu 20.04?
Choose a different version or distribution.

Ubuntu 20.04 ⌄

Introduction

and install

ROLL TO TOP

most (if not

Sign up for our newsletter. Get the latest tutorials on SysAdmin and open source topics.   ✕

| Enter your email address | **Sign Up** |

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 Ubuntu 20.04 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 Ubuntu 20.04 server set up by following this initial server setup for Ubuntu 20.04 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 Ubuntu 20.04. 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

Install Certbot and it's Nginx plugin with `apt`:

```
$ sudo apt instal  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

```
...
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:

ROLL TO TOP

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:

```
$ 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:

```
$ 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:

ROLL TO TOP

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.

```
Please choose whether or not to redirect HTTP traffic to HTTPS, removing HTTP access.
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
1: No redirect - Make no further changes to the webserver configuration.
2: Redirect - Make all requests redirect to secure HTTPS access. Choose this for
new sites, or if you're confident your site works on HTTPS. You can undo this
change by editing your web server's configuration.
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Select the appropriate number [1-2] then [enter] (press 'c' to cancel):
```

Select your choice then hit ENTER. 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:

```
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 2020-08-18. 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.

ROLL TO TOP

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`:

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

Output

```
● certbot.timer - Run certbot twice daily
     Loaded: loaded (/lib/systemd/system/certbot.timer; enabled; vendor preset: enable
     Active: active (waiting) since Mon 2020-05-04 20:04:36 UTC; 2 weeks 1 days ago
    Trigger: Thu 2020-05-21 05:22:32 UTC; 9h left
   Triggers: ● certbot.service
```

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

```
$ 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

Brian Boucheron

Senior Technical Writer at DigitalOcean

Related

TUTORIAL

How To Troubleshoot Common Apache Errors

There are three main commands, and a set of common log locations that you can use to get started troubleshooting Apache …

TUTORIAL

How To Configure Remote Access for MongoDB on Ubuntu 20.04

This tutorial walks you through configuring a MongoDB installation to …

TUTORIAL

How To Secure MongoDB on Ubuntu 20.04

TUTORIAL

How to Install MongoDB on Ubuntu 20.04

SCROLL TO TOP

default, meaning that any user with access to the …

in modern web applications. It is classified

Still looking for an answer?

Ask a question

Search for more help

Comments

0 Comments

Leave a comment...

Sign In to Comment

ROLL TO TOP

BECOME A CONTRIBUTOR

You get paid; we donate to tech nonprofits.

GET OUR BIWEEKLY NEWSLETTER

Sign up for Infrastructure as a Newsletter.

Sign up for our newsletter. Get the latest tutorials on SysAdmin and open source topics.

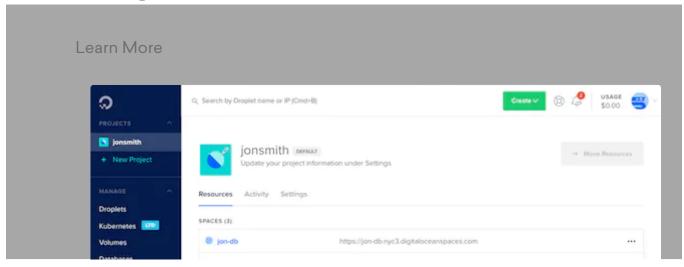Enter your email address | Sign Up

SCROLL TO TOP

Featured on Community   Kubernetes Course   Learn Python 3   Machine Learning in Python
Getting started with Go   Intro to Kubernetes

DigitalOcean Products   Droplets   Managed Databases   Managed Kubernetes   Spaces Object Storage
Marketplace

Welcome to the developer cloud

DigitalOcean makes it simple to launch in the cloud and scale up as you grow – whether you're running one virtual machine or ten thousand.

Learn More

ROLL TO TOP

© 2020 DigitalOcean, LLC. All rights reserved.

About
Leadership

Blog
Careers
Partners
Referral Program
Press
Legal
Security & Trust Center

Products

Pricing
Products Overview

Droplets
Kubernetes
Managed Databases
Spaces
Marketplace
Load Balancers
Block Storage
API Documentation
Documentation
Release Notes

Community

Tutorials
Q&A

Tools and Integrations
Tags
Product Ideas
Write for DigitalOcean
Presentation Grants
Hatch Startup Program
Shop Swag
Research Program
Open Source
Code of Conduct

Contact

Get Support
Trouble Signing In?
Sales
Report Abuse
System Status

Sign up for our newsletter. Get the latest tutorials on SysAdmin and open source topics. ✕

Enter your email address

Sign Up

ROLL TO TOP