Products ›

Solutions ›

Developers ›

Partners ›

Pricing

DigitalOcean

Log in ⌄

Sign up

Blog

Docs

Get Support

Contact Sales

Tutorials Questions Learning Paths For Businesses Proc

CONTENTS

// TUTORIAL //

How To Host a Website Using Cloudflare and Nginx on Ubuntu 22.04

Published on July 26, 2022

DNS Nginx Security Ubuntu 22.04

anondon and Rachel Lee

Not using Ubuntu 22.04?
Choose a different version or distribution.

Ubuntu 22.04 ⌄

Th selected the Electronic Frontier Foundation to receive a dc part of the Write for DOnations program.

Introduction

Cloudflare is a service that sits between the visitor and the website owner's server, acting as a reverse proxy for websites. Cloudflare provides a Content Delivery Network (CDN), as well as DDoS mitigation and distributed domain name server services.

Nginx is a popular web server responsible for hosting some of the largest and highest-traffic sites on the internet. It's common for organizations to serve websites with Nginx and use Cloudflare as a CDN and DNS provider.

In this tutorial, you will secure your website served by Nginx with an Origin CA certificate from Cloudflare and then configure Nginx to use authenticated pull requests. The advantages of using this setup are that you benefit from Cloudflare's CDN and fast DNS resolution while ensuring that all connections pass through Cloudflare. This prevents any malicious requests from reaching your server.

Prerequisites

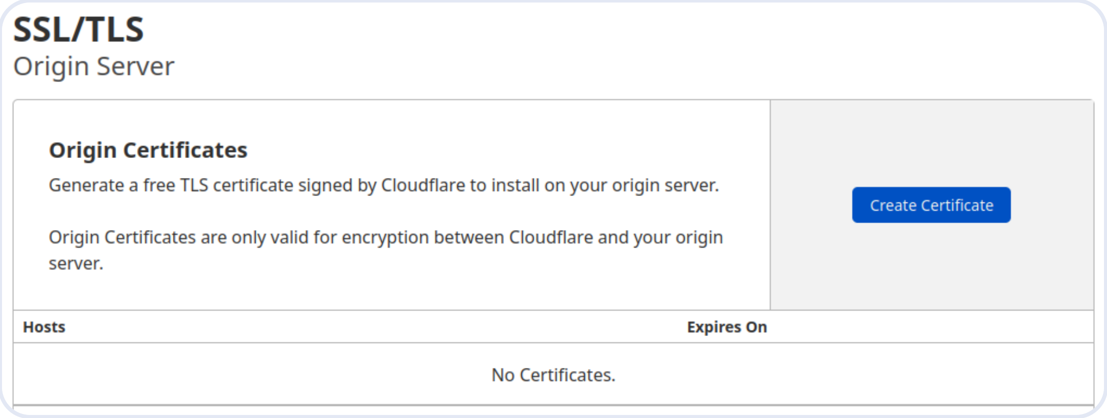
To complete this tutorial, you'll need the following:

- One Ubuntu 22.04 server set up by following the Ubuntu 22.04 initial server setup guide, including a `sudo` non-root user and a firewall.
- Nginx installed on your server. You can follow our guide on how to install Nginx on Ubuntu 22.04.
- A Cloudflare account.
- A registered domain added to your Cloudflare account that points to your Nginx server. Our guide on how to mitigate DDoS attacks against your website with Cloudflare can help you set this up. Our introduction to DNS terminology, components, and concepts can also provide assistance.
- An Nginx Server Block configured for your domain, which you can do by following Step 5 of How To Install Nginx on Ubuntu 22.04.

S - Generating an Origin CA TLS C cate

The Cloudflare Origin CA lets you generate a free TLS certificate signed by Cloudflare to install on your Nginx server. By using the Cloudflare generated TLS certificate you can secure the connection between Cloudflare's servers and your Nginx server.

To generate a certificate with Origin CA, log in to your Cloudflare account in a web browser. Select the domain that you want to secure and navigate to the **SSL/TLS** section of your Cloudflare dashboard. From there, navigate to the **Origin Server** tab and click on the **Create Certificate** button:



SSL/TLS
Origin Server

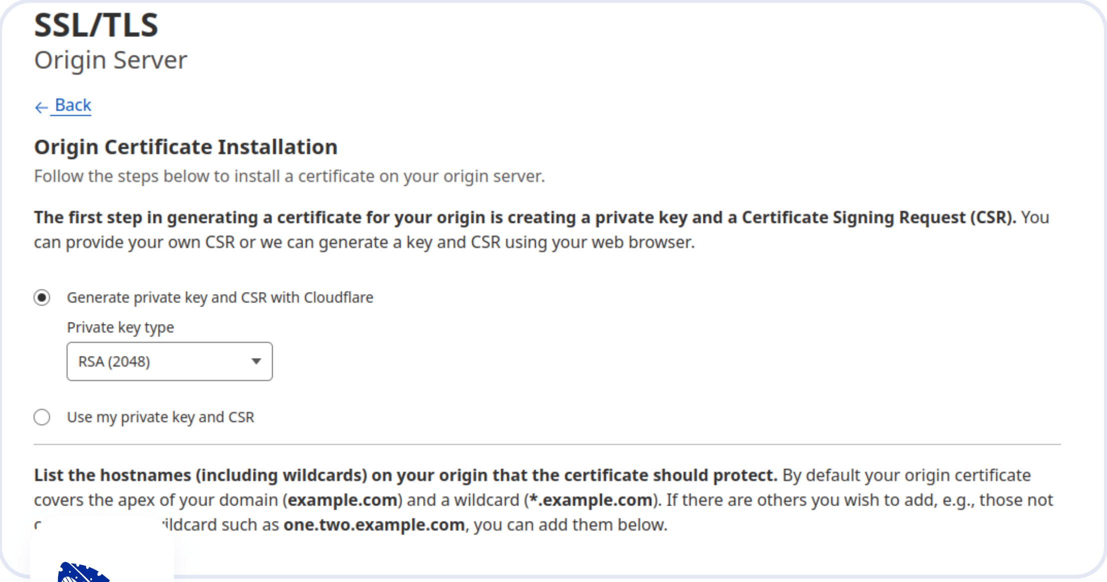
Origin Certificates

Generate a free TLS certificate signed by Cloudflare to install on your origin server.

Origin Certificates are only valid for encryption between Cloudflare and your origin server.

Hosts	Expires On
No Certificates.	

Leave the default option of **Generate private key and CSR with Cloudflare** selected.



SSL/TLS
Origin Server

Origin Certificate Installation

Follow the steps below to install a certificate on your origin server.

The first step in generating a certificate for your origin is creating a private key and a Certificate Signing Request (CSR). You can provide your own CSR or we can generate a key and CSR using your web browser.

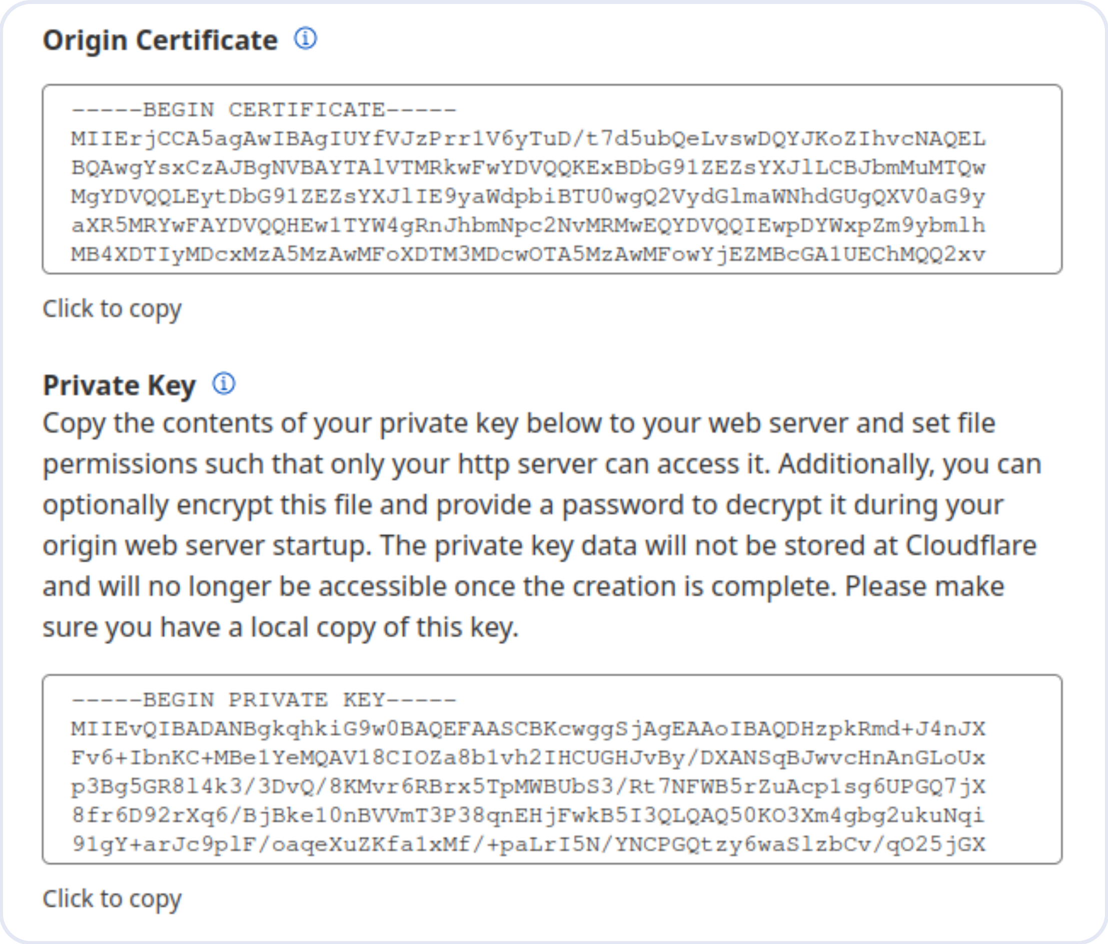
● Generate private key and CSR with Cloudflare

Private key type
RSA (2048)

○ Use my private key and CSR

List the hostnames (including wildcards) on your origin that the certificate should protect. By default your origin certificate covers the apex of your domain (**example.com**) and a wildcard (***.example.com**). If there are others you wish to add, e.g., those not c[illegible]ildcard such as **one.two.example.com**, you can add them below.

Cl[illegible]e and you will see a dialog with the **Origin Certificate** and **Private key**. You need to transfer both the origin certificate and private key from Cloudflare to your server. For security reasons, the **Private Key** information will not be displayed again, so copy the key to your server before clicking **Ok**.



Origin Certificate ⓘ

```
-----BEGIN CERTIFICATE-----
MIIErjCCA5agAwIBAgIUYfVJzPrr1V6yTuD/t7d5ubQeLvswDQYJKoZIhvcNAQEL
BQAwgYsxCzAJBgNVBAYTAlVTMRkwFwYDVQQKExBDbG91ZEZsYXJlLCBJbmMuMTQw
MgYDVQQLEytDbG91ZEZsYXJlIE9yaWdpbiBTU0wgQ2VydGlmaWNhdGUgQXV0aG9y
aXR5MRYwFAYDVQQHEw1TYW4gRnJhbmNpc2NvMRMwEQYDVQQIEwpDYWxpZm9ybmlh
MB4XDTIyMDcxMzA5MzAwMFoXDTM3MDcwOTA5MzAwMFowYjEZMBcGA1UEChMQQ2xv
```

Click to copy

Private Key ⓘ

Copy the contents of your private key below to your web server and set file permissions such that only your http server can access it. Additionally, you can optionally encrypt this file and provide a password to decrypt it during your origin web server startup. The private key data will not be stored at Cloudflare and will no longer be accessible once the creation is complete. Please make sure you have a local copy of this key.

```
-----BEGIN PRIVATE KEY-----
MIIEvQIBADANBgkqhkiG9w0BAQEFAASCBKcwggSjAgEAAoIBAQDHzpkRmd+J4nJX
Fv6+IbnKC+MBe1YeMQAV18CIOZa8b1vh2IHCUGHJvBy/DXANSqBJwvcHnAnGLoUx
p3Bg5GR8l4k3/3DvQ/8KMvr6RBrx5TpMWBUbS3/Rt7NFWB5rZuAcp1sg6UPGQ7jX
8fr6D92rXq6/BjBke10nBVVmT3P38qnEHjFwkB5I3QLQAQ50KO3Xm4gbg2ukuNqi
91gY+arJc9plF/oaqeXuZKfa1xMf/+paLrI5N/YNCPGQtzy6waSlzbCv/qO25jGX
```

Click to copy

You'll use the `/etc/ssl` directory on the server to hold the origin certificate and the private key files. The folder already exists on the server.

First, copy the contents of the **Origin Certificate** displayed in the dialog box in your browser.

Then, on your server, open `/etc/ssl/cert.pem` in your preferred text editor:

Copy

```
nano /etc/ssl/cert.pem
```

Paste the certificate contents into the file. Then save and exit the editor. If you are using `nano`, press `Ctrl+X`, then when prompted, `Y` and then Enter.

Then return to your browser and copy the contents of the **Private key**. Open the file `/etc/ssl/key.pem` for editing:

Copy

```
$ sudo nano /etc/ssl/key.pem
```

Paste the private key into the file, save the file, and exit the editor.

> **Note:** Sometimes, when you copy the certificate and key from the Cloudflare dashboard and paste it into the relevant files on the server, blank lines are inserted. Nginx will treat such certificates and keys as invalid, so ensure that there are no blank lines in your files.

> **Warning:** Cloudflare's Origin CA Certificate is only trusted by Cloudflare and therefore should only be used by origin servers that are actively connected to Cloudflare. If at any point you pause or disable Cloudflare, your Origin CA certificate will throw an untrusted certificate error.

Now that you copied the key and certificate files to your server, you need to update the Nginx configuration to use them.

Step 2 – Installing the Origin CA Certificate in Nginx

In the previous section, you generated an origin certificate and private ke[illegible] Cloudflare's dashboard and saved the files to your server. Now yo[illegible]e the Nginx configuration for your site to use the origin ce[illegible]nd private key to secure the connection between Cloudflare's servers and your server.

First, make sure that UFW will allow HTTPS traffic. Enable `Nginx Full`, which will open both port `80` (HTTP) and port `443` (HTTPS):

Copy

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

Now reload UFW:

Copy

```
$ sudo ufw reload
```

Finally, check that your new rules are allowed and that UFW is active:

Copy

```
$ sudo ufw status
```

You will see an output like this:

```
Output
Status:  active

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

No[illegible]e ready to adjust your Nginx server block. Nginx creates a de[illegible]er block during installation. Remove it if it still exists, as you've already configured a custom server block for your domain:

Copy

```
$ sudo rm /etc/nginx/sites-enabled/default
```

Next, open the Nginx configuration file for your domain:

Copy

```
$ sudo nano /etc/nginx/sites-available/ your_domain
```

The file should look like this:

/etc/nginx/sites-available/your_domain

Copy

```
server {
        listen 80;
        listen [::]:80;

        root /var/www/ your_domain /html;
        index index.html index.htm index.nginx-debian.html;

        server_name  your_domain   www.your_domain ;

        location / {
                try_files $uri $uri/ =404;
        }
}
```

You'll modify the Nginx configuration file to do the following:

- Listen on port `80` and redirect all requests to use `https`.
- Listen on port `443` and use the origin certificate and private key added in the previous section.

Modify the file so it looks like the following:

/etc/nginx/sites-available/your_domain Copy

```
server {
    listen 80;
    listen [::]:80;
    server_name  your_domain   www.your_domain ;
     return 302 https://$server_name$request_uri;
}

 server {

     # SSL configuration

     listen 443 ssl http2;
     listen [::]:443 ssl http2;
     ssl_certificate         /etc/ssl/cert.pem;
     ssl_certificate_key     /etc/ssl/key.pem;

    server_name  your_domain   www.your_domain ;

    root /var/www/ your_domain /html;
    index index.html index.htm index.nginx-debian.html;


    location / {
            try_files $uri $uri/ =404;
    }
 }
```

Save the file and exit the editor.

Next, test to ensure that there are no syntax errors in any of your Nginx configuration files:

Copy

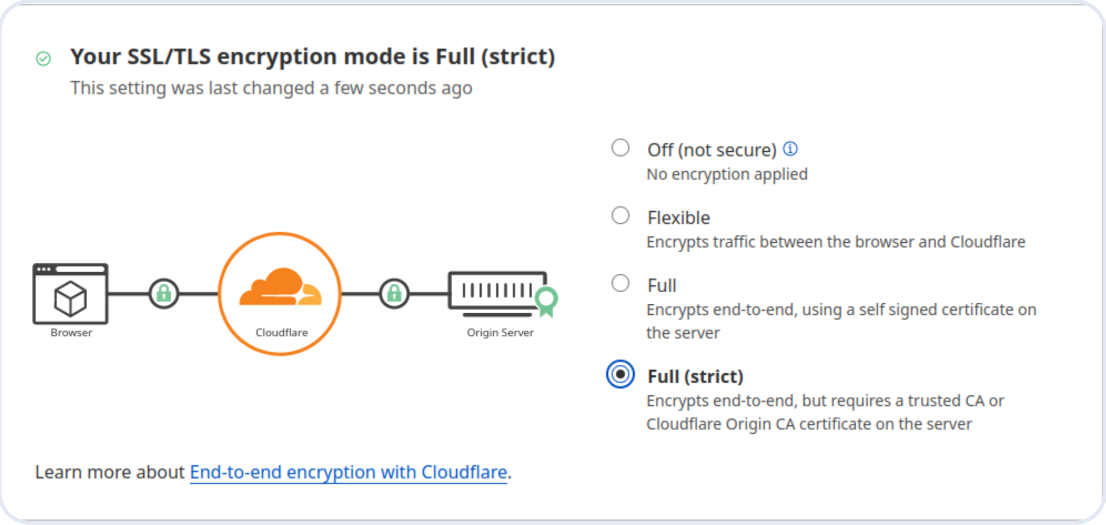
```
$ sudo nginx -t
```

If no problems, restart Nginx to enable your changes:

Copy

```
$ sudo systemctl restart nginx
```

Now go to the Cloudflare dashboard's **SSL/TLS** section, navigate to the **Overview** tab, and change **SSL/TLS encryption mode** to **Full (strict)**. This informs Cloudflare to always encrypt the connection between Cloudflare and your origin Nginx server.



Your SSL/TLS encryption mode is Full (strict)
This setting was last changed a few seconds ago

○ Off (not secure)
No encryption applied

○ Flexible
Encrypts traffic between the browser and Cloudflare

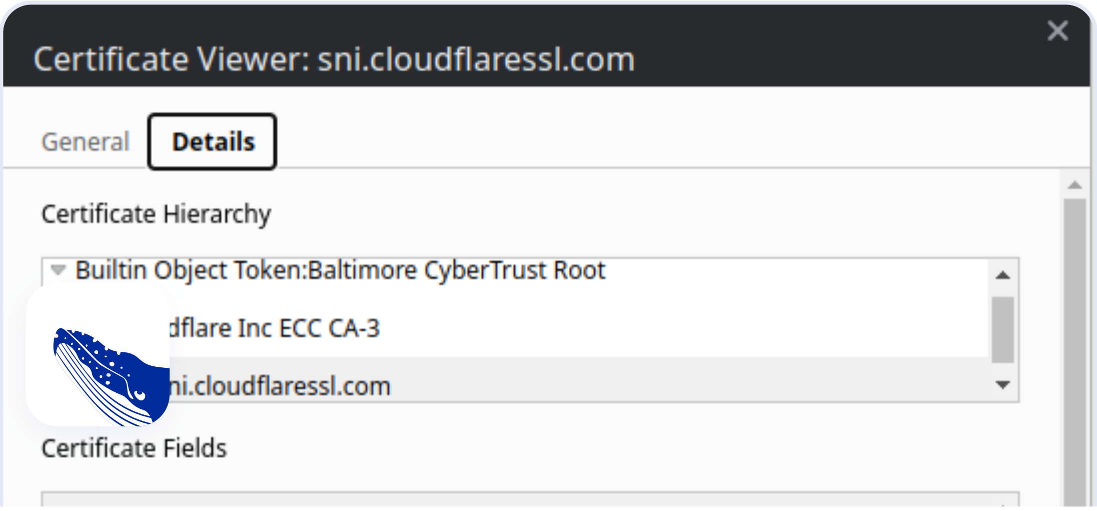
○ Full
Encrypts end-to-end, using a self signed certificate on the server

● **Full (strict)**
Encrypts end-to-end, but requires a trusted CA or Cloudflare Origin CA certificate on the server

Learn more about End-to-end encryption with Cloudflare.

Now visit your website at `https:// your_domain` to verify that it's set up properly. You'll see your home page displayed, and the browser will report that the site is secure.

To view the details of your certificate, access your browser's **Developer Tools**, select the **Security** tab, and then **View Certificate.**

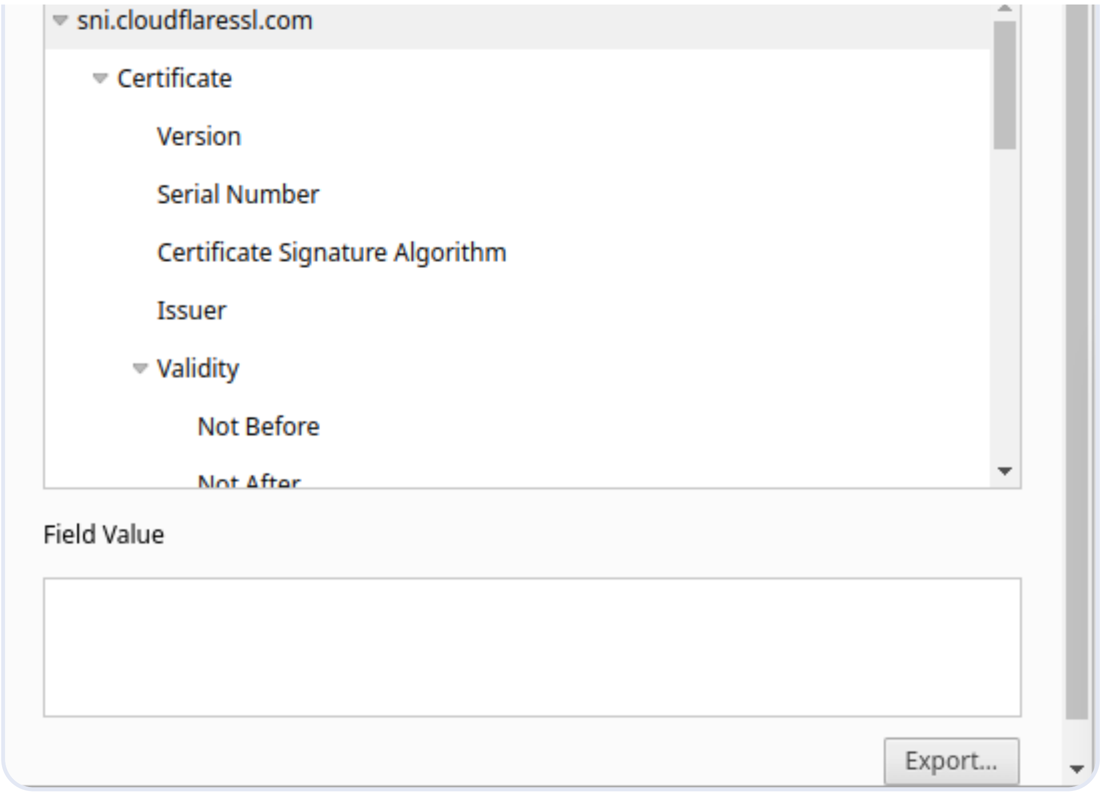


Certificate Viewer: sni.cloudflaressl.com

Certificate Hierarchy

Builtin Object Token:Baltimore CyberTrust Root
dflare Inc ECC CA-3
ni.cloudflaressl.com

Certificate Fields



sni.cloudflaressl.com
- Certificate
  - Version
  - Serial Number
  - Certificate Signature Algorithm
  - Issuer
  - Validity
    - Not Before
    - Not After

Field Value

> **Note:** You may notice that your certificate does not list Cloudflare as the issuer. This is because Cloudflare may use other certificate authorities, such as Let’s Encrypt. For a complete list, check out Cloudflare’s product documentation for certificate authorities.

In the next section, you will set up Authenticated Origin Pulls to verify that your origin server is indeed talking to Cloudflare and not some other server. By doing so, Nginx will be configured to only accept requests that use a valid client certificate from Cloudflare; all requests that have not passed through Cloudflare will be dropped.

Step 3 – Setting Up Authenticated Origin Pulls

The Origin CA certificate will help Cloudflare verify that it is talking to the correct origin server. This step will use TLS Client Authentication to verify that your origin Nginx server is talking to Cloudflare.

In a client-authenticated TLS handshake, both sides provide a certificate to be verified. The origin server is configured to only accept requests that use a valid client certificate from Cloudflare. Requests which have not passed through Cloudflare will be dropped as they will not have Cloudflare's certificate. This means that attackers cannot circumvent Cloudflare's security measures and directly connect to your Nginx server.

Cloudflare presents certificates signed by a CA with the following certificate:

```
-----BEGIN CERTIFICATE-----
MIIGCjCCA/KgAwIBAgIIV5G6lVbCLmEwDQYJKoZIhvcNAQENBQAwgZAxCzAJBgN
BAYTAlVTMRkwFwYDVQQKExBDbG91ZEZsYXJlLCBJbmMuMRQwEgYDVQQLEwtPcml
aW4gUHVsbDEWMBQGA1UEBxMNU2FuIEZyYW5jaXNjbzETMBEGA1UECBMKQ2FsaWZ
cm5pYTEjMCEGA1UEAxMab3JpZ2luLXB1bGwuY2xvdWRmbGFyZS5uZXQwHhcNMTk
MDEwMTg0NTAwWhcNMjkxMTAxMTcwMDAwWjCBkDELMAkGA1UEBhMCVVMxGTAXBgN
BAoTEENsb3VkRmxhcmUsIEluYy4xFDASBgNVBAsTC09yaWdpbiBQdWxsMRYwFAY
VQQHEw1TYW4gRnJhbmNpc2NvMRMwEQYDVQQIEwpDYWxpZm9ybmlhMSMwIQYDVQQ
ExpvcmlnaW4tcHVsbC5jbG91ZGZsYXJlLm5ldDCCAiIwDQYJKoZIhvcNAQEBBQA
ggIPADCCAgoCggIBAN2y2zojYfl0bKfhp0AJBFeV+jQqbCw3sHmvEPwLmqDLqyn
42tZXR5y914ZB9ZrwbL/K5O46exd/LujJnV2b3dzcx5rtiQzso0xzljqbnbQT20
ihx/WrF4OkZKydZzsdaJsWAPuplDH5P7J82q3re88jQdgE5hqjqFZ3clCG7lxoB
hLaazm3NJJlUfzdk97ouRvnFGAuXd5cQVx8jYOOeU60sWqmMe4QHdOvpqB91bJo
QSKVFjUgHeTpN8tNpKJfb9LIn3pun3bC9NKNHtRKMNX3Kl/sAPq7q/AlndvA2Kw
Dkum2mHQUGdzVHqcOgea9BGjLK2h7SuX93zTWL02u799dr6Xkrad/WShHchfjjR
aL35niJUDr02YJtPgxWObsrfOU63B8juLUphW/4BOjjJyAG5l9j1//aUGEi/sEe
lqVv0P78QrxoxR+MMXiJwQab5FB8TG/ac6mRHgF9CmkX90uaRh+OC07XjTdfSKG
PpM9hB2ZhLol/nf8qmoLdoD5HvODZuKu2+muKeVHXgw2/A6wM7OwrinxZiyBk5H
CvaADH7PZpU6z/zv5NU5HSvXiKtCzFuDu4/Zfi34RfHXeCUfHAb4KfNRXJwMsxU
+4ZpSAX2G6RnGU5meuXpU5/V+DQJp/e69XyyY6RXDoMywaEFlIlXBqjRRA2pAgM
AAGjZjBkMA4GA1UdDwEB/wQEAwIBBjASBgNVHRMBAf8ECDAGAQH/AgECMB0GA1U
DgQWBBRDWUsraYuA4REzalfNVzjann3F6zAfBgNVHSMEGDAWgBRDWUsraYuA4RE
alfNVzjann3F6zANBgkqhkiG9w0BAQ0FAAOCAgEAkQ+T9nqcSlAuW/90DeYmQOW
QhqOor5psBEGvxbNGV2hdLJY8h6QUq48BCevcMChg/L1CkznBNI40i3/6heDn3I
zVEwXKf34pPFCACWVMZxbQjkNRTiH8iRur9EsaNQ5oXCPJkhwg2+IFyoPAAYURo
VcI9SCDUa45clmYHJ/XYwV1icGVI8/9b2JUqklnOTa5tugwIUi5sTfipNcJXHhg
6BKYDl0/UP0lLKbsUETXeTGDiDpxZYIgbcFrRDDkHC6BSvdWVEiH5b9mH2BON60
0O0j8EEKTwi9jnafVtZQXP/D8yoVowdFDjXcKkOPF/1gIh9qrFR6GdoPVgB3SkL
5ulBqZaCHm563jsvWb/kXJnlFxW+1bsO9BDD6DweBcGdNurgmH625wBXksSdD7y
fakk8DagjbjKShYlPEFOAqEcliwjF45eabL0t27MJV61O/jHzHL3dknXeE4BDa2
bA+JbyJeUMtU7KMsxvx82RmhqBEJJDBCJ3scVptvhDMRrtqDBW5JShxoAOcpFQG
iYWicn46nPDjgTU0bX1ZPpTpryXbvciVL5RkVBuyX2ntcOLDPlZWgxZCBp96x07
        4RzZPNAxCXERVxajn/FLcOhglVAKo5H0ac+AitlQ0ip55D2/mf8o72t
        EXdiIXWUq/o=
         CERTIFICATE-----
```

You can also download the certificate directly from Cloudflare's documentation.

Copy this certificate.

Then create the file `/etc/ssl/cloudflare.crt` file to hold Cloudflare's certificate:

Copy

```
$ sudo nano /etc/ssl/cloudflare.crt
```

Add the certificate to the file. Then save the file and exit the editor.

Now update your Nginx configuration to use TLS Authenticated Origin Pulls. Open the configuration file for your domain:

Copy

```
$ sudo nano /etc/nginx/sites-available/ your_domain
```

Add the `ssl_client_certificate` and `ssl_verify_client` directives as shown in the following example:

/etc/nginx/sites-available/your_domain

Copy

```
. . .

server {

    # SSL configuration

    listen 443 ssl http2;
    listen [::]:443 ssl http2;
    ssl_certificate         /etc/ssl/cert.pem;
    ssl_certificate_key     /etc/ssl/key.pem;
    ssl_client_certificate /etc/ssl/cloudflare.crt;
    ssl_verify_client on;
    . . .
```

Save the file and exit the editor.

Next, test Nginx to make sure that there are no syntax errors in your Nginx configuration:

Copy

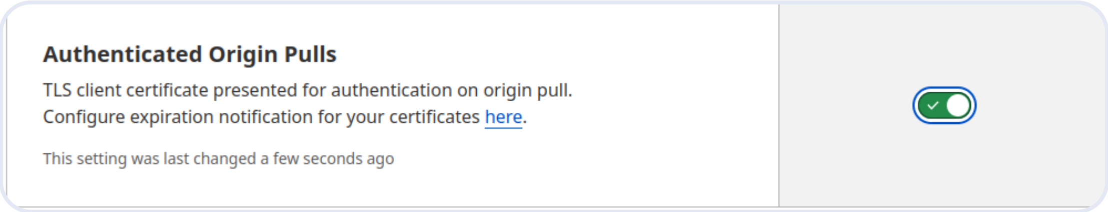
```
$ sudo nginx -t
```

If no problems were found, restart Nginx to enable your changes:

Copy

```
$ sudo systemctl restart nginx
```

Finally, to enable Authenticated Pulls, open the **SSL/TLS** section in the Cloudflare dashboard, navigate to the **Origin Server** tab and toggle the **Authenticated Origin Pulls** option .



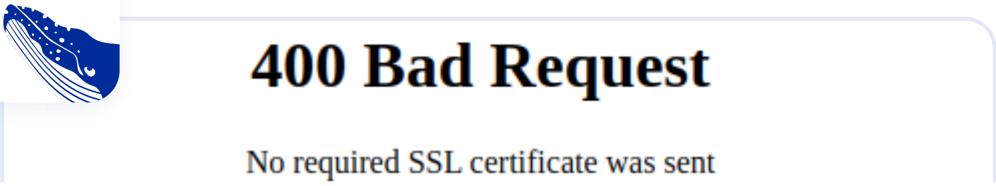
Authenticated Origin Pulls

TLS client certificate presented for authentication on origin pull.
Configure expiration notification for your certificates here.

This setting was last changed a few seconds ago

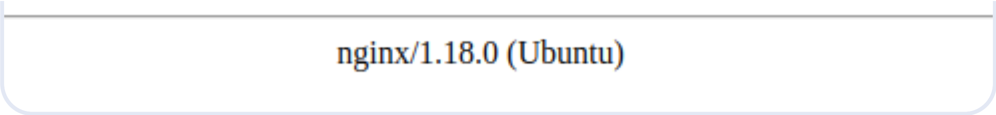
Now visit your website at `https:// your_domain` to verify that it was set up properly. As before, you'll see your home page displayed.

To verify that your server will only accept requests signed by Cloudflare's CA, toggle the **Authenticated Origin Pulls** option to disable it and then reload your website. You should get the following error message :



400 Bad Request

No required SSL certificate was sent



nginx/1.18.0 (Ubuntu)

Your origin server raises an error if Cloudflare’s CA does not sign a request.

> **Note:** Most browsers will cache requests, so to see the above change you can use Incognito/Private browsing mode in your browser. To prevent Cloudflare from caching requests while you set up your website, navigate to **Overview** in the Cloudflare dashboard and toggle **Development Mode**.

Now that you know it works properly return to the **SSL/TLS** section in the Cloudflare dashboard, navigate to the **Origin Server** tab and toggle the **Authenticated Origin Pulls** option again to enable it.

Conclusion

In this tutorial, you secured your Nginx-powered website by encrypting traffic between Cloudflare and the Nginx server using an Origin CA certificate from Cloudflare. You then set up Authenticated Origin Pulls on the Nginx server to ensure that it only accepts Cloudflare servers’ requests, preventing anyone else from directly connecting to the Nginx server.

About the authors

anondon Author

Rachel Lee Editor
Technical Editor

Still looking for an answer? Ask a question

Search for more help

Was this helpful? Yes No

Comments

3 Comments

Leave a comment...

Th x 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

62c4a6767f714ffa9e34d1192642bf • November 5, 2023

Not working!, the 400 bad request has no fix. wasted hours of my time.

Reply

rrttrrtt2 • September 14, 2023

Hi The website i want to host is using nextjs. To do this i've setup a reverse proxy however i don't know the correct way todo it. I have followed your guide and others online but i only receive the 'Bad gateway' message when trying to access the page. I guess what i'm trying to say is that a guide on how to create a revers proxy with nginx and cloudflare would be nice or even just some insight as to where i've gone wrong.

```
server {

    server_name domain;

    location / {
         proxy_pass             http://127.0.0.1:8080;
         proxy_read_timeout     60;
         proxy_connect_timeout  60;
         proxy_redirect         off;

         # Allow the use of websockets
         proxy_http_version 1.1;
         proxy_set_header Upgrade $http_upgrade;
         proxy_set_header Connection 'upgrade';
         proxy_set_header Host $host;
         proxy_cache_bypass $http_upgrade;
    }


    listen [::]:443 ssl ipv6only=on;
    listen 443 ssl default_server;

    # SSL configuration
    ssl_certificate         /etc/ssl/cert.pem;
    ssl_certificate_key     /etc/ssl/key.pem;
    ssl_client_certificate /etc/ssl/cloudflare.crt;
    ssl_verify_client on;

 }
server {
    if ($host = www.DOMAINNAME.com) {
        return 301 https://$host$request_uri;
    }

    if ($host = DOMAINNAME.com) {
        return 301 https://$host$request_uri;
    }

    listen 80 ;
    listen [::]:80 ;
    server_name domain;
    return 301 https://$host$request_uri;
}
```

Reply

da9996d51cd1433bb1356e84ae926d • July 10, 2023

I shows that “400 Bad request No required SSL certificate was sent nginx/1.18.0 (Ubuntu)”

The website wroks fine before the ssl installation

ticated Origin Pulls : ✅ Full (strict)✅ SSL/TLS
mender✅

Copy

```
server {
    listen 80;
    listen [::]:80;
    server_name domain.xyz www.domain.xyz;
    return 302 https://$server_name$request_uri;
}
server {

    # SSL configuration

    listen 443 ssl http2;
    listen [::]:443 ssl http2;

    ssl_certificate         /etc/ssl/cert.pem;
    ssl_certificate_key     /etc/ssl/key.pem;
    ssl_client_certificate /etc/ssl/cloudflare.crt;
    ssl_verify_client on;


    server_name domain.xyz www.domain.xyz;

    location = /favicon.ico { access_log off; log_not_f


    location / {
        include proxy_params;
        proxy_pass http://unix:/run/gunicorn.sock;
    }


}
```

Please Help...

Show replies ⌄ Reply

Popular Topics

Ubuntu

Linux Basics

JavaScript

Python

MySQL

Docker

Kubernetes

All tutorials →

Talk to an expert →

ratulations on unlocking the whale ambience easter egg!
he whale button in the bottom left of your screen to
some ambient whale noises while you read.

Thank you to the Glacier Bay National Park & Preserve and

♡ Merrick079 for the sounds behind this easter egg.

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

Reset easter egg to be discovered again / Permanently dismiss and hide easter egg

Hollie's Hub for Good

Working on improving health and education, reducing inequality, and spurring economic growth? We'd like to help.

Learn more →

Featured Tutorials

Kubernetes Course Learn Python 3

Machine Learning in Python Getting started with Go

Intro to Kubernetes

DigitalOcean Products

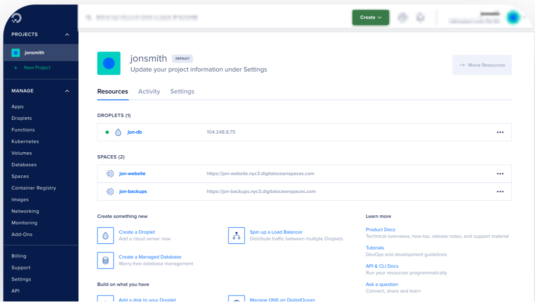
App Platform Virtual Machines Managed Databases

Managed Kubernetes Block Storage Object Storage

Marketplace VPC Load Balancers

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

Company

Products

Community

Solutions

Co

Sitemap. Cookie Preferences