Gitlab and Certbot – Lets Encrypt

SSL and HTTPS

In this post we will talk about HTTPS and how to add it to your GitLab Pages site with Let's Encrypt.

Why TLS/SSL?

When discussing HTTPS, it's common to hear people saying that a static website doesn't need HTTPS, since it doesn't receive any POST requests, or isn't handling credit card transactions or any other secure request. But that's not the whole story.

TLS (formerly SSL) is a security protocol that can be added to HTTP to increase the security of your website by:

1. properly authenticating yourself: the client can trust that you are really **you**. The TLS handshake that is made at the beginning of the connection ensures the client that no one is trying to impersonate you;
2. data integrity: this ensures that no one has tampered with the data in a request/response cycle;
3. encryption: this is the main selling point of TLS, but the other two are just as important. This protects the privacy of the communication between client and server.

The TLS layer can be added to other protocols too, such as FTP (making it FTPS) or WebSockets (making `ws://wss://`).

HTTPS Everywhere

Nowadays, there is a strong push for using TLS on every website. The ultimate goal is to make the web safer, by adding those three components cited above to every website.

The first big player was the **HTTPS Everywhere** browser extension. Google has also been using HTTPS compliance to better rank websites since **2014**.

TLS certificates

In order to add TLS to HTTP, one would need to get a certificate, and until 2015, one would need to either pay for it or figure out how to do it with one of the available **Certificate Authorities**.

Enter **Let's Encrypt**, a free, automated, and open Certificate Authority. Since **December 2015** anyone can get a free certificate from this new Certificate Authority from the comfort of their terminal.

Implementation

So, let's suppose we're going to create a static blog with **Jekyll 3**. If you are not creating a blog or are not using Jekyll just follow along, it should be straightforward enough to translate the steps for different purposes. You can also find many example projects using different static site generators (like Middleman or Hugo) in **GitLab's example projects**.

A simple example blog can be created with:

```
$ jekyll new cool-blog
New jekyll site installed in ~/cool-blog.
$ cd cool-blog/
```

Now you have to create a GitLab project. Here we are going to create a "user page", which means that it is a project created within a user account (not a group account), and that the name of the project looks like `YOURUSERNAME.gitlab.io`. Refer to the **"Getting started" section of the GitLab Pages manual** for more information on that.

From now on, remember to replace `YOURDOMAIN.org` with your custom domain and `YOURUSERNAME` with, well, your username. ;)

Create a project named `YOURUSERNAME.gitlab.io` so that GitLab will identify the project correctly. After that, upload your code to GitLab:

```
$ git remote add origin git@gitlab.com:YOURUSERNAME/YOURUSERNAME.gitlab.io.git
$ git push -u origin master
```

OK, so far we have a project uploaded to GitLab, but we haven't configured GitLab Pages yet. To configure it, just create a `.gitlab-ci.yml` file in the root directory of your repository with the following contents:

```
pages:
  stage: deploy
  image: ruby:2.3
  script:
    - gem install jekyll
    - jekyll build -d public/
  artifacts:
    paths:
      - public
  only:
    - master
```

This file instructs GitLab Runner to `deploy` by installing Jekyll and building your website under the `public/`folder (`jekyll build -d public/`).

While you Wait for the build process to complete, you can track the progress in the Builds page of your project. Once it starts, it probably won't take longer than a few minutes. Once the build is finished, your website will be available at `https://YOURUSERNAME.gitlab.io`. Note that GitLab already provides TLS certificates to all subdomains of `gitlab.io` (but it has some limitations, so please **refer to the documentation for more**). So if you don't want to add a custom domain, you're done.

Configuring the TLS certificate of your custom domain.

Once you buy a domain name and point that domain to your GitLab Pages website, you need to configure 2 things:

1. add the domain to GitLab Pages configuration (see documentation);
2. add your custom certificate to your website.

Once you add your domain, your website will be available under both `http://YOURDOMAIN.org` and `https://YOURUSERNAME.gitlab.io`.

But if you try to access your custom domain with `HTTPS` (`https://YOURDOMAIN.org` in this case), your browser will show that horrible page, saying that things are going wrong and someone is trying to steal your information. *Why is that?*

Since GitLab offers TLS certificates to all `gitlab.io` pages and your custom domain is just a `CNAME` over that same domain, GitLab serves the `gitlab.io` certificate, and your browser receives mixed messages: on one side, the browser is trying to access `YOURDOMAIN.org`, but on the other side it is getting a TLS certificate for `*.gitlab.io`, signaling that something is wrong.

In order to fix it, you need to obtain a certificate for `YOURDOMAIN.org` and add it to GitLab Pages. For that we are going to use **Let's Encrypt**.

Let's Encrypt is a new certificate authority that offers both *free* and *automated* certificates. That's perfect for us: we don't have to pay for having HTTPS and you can do everything within the comfort of your terminal.

We begin with downloading the `letsencrypt-auto` utility. Open a new terminal window and type:

```
$ git clone https://github.com/letsencrypt/letsencrypt

$ cd letsencrypt
```

`letsencrypt-auto` offers a lot of functionality. For example, if you have a web server running Apache, you could add `letsencrypt-auto --apache` inside your webserver and have everything done for you. `letsencrypt` targets primarily Unix-like webservers, so the `letsencrypt-auto` tool won't work for Windows users. Check **this tutorial**to see how to get Let's Encrypt certificates while running Windows.

Since we are running on GitLab's servers instead, we have to do a bit of manual work:

```
$ ./letsencrypt-auto certonly -a manual -d YOURDOMAIN.org
#
# If you want to support another domain, www.YOURDOMAIN.org, for example, you
# can add it to the domain list after -d like:
# ./letsencrypt-auto certonly -a manual -d YOURDOMAIN.org www.YOURDOMAIN.org
#
```

After you accept that your IP will be publicly logged, a message like the following will appear:

```
Make sure your web server displays the following content at
http://YOURDOMAIN.org/.well-known/acme-challenge/5TBu788fW0tQ5EOwZMdu1Gv3e9C33gxjV58hVtWTbDM
before continuing:


5TBu788fW0tQ5EOwZMdu1Gv3e9C33gxjV58hVtWTbDM.ewlbSYgvIxVOqiP1lD2zeDKWBGEZMRfO_4kJyLRP_4U


#
# output omitted
#


Press ENTER to continue
```

Now it is waiting for the server to be correctly configured so it can go on. Leave this terminal window open for now.

So, the goal is to the make our already-published static website return said token when said URL is requested. That's easy: create a custom page! Just create a file in your blog folder that looks like this:

```
---
layout: null
permalink: /.well-known/acme-challenge/5TBu788fW0tQ5EOwZMdu1Gv3e9C33gxjV58hVtWTbDM
.html
---


5TBu788fW0tQ5EOwZMdu1Gv3e9C33gxjV58hVtWTbDM.ewlbSYgvIxVOqiP1lD2zeDKWBGEZMRfO_4kJyL
RP_4U
```

This tells Jekyll to create a static page, which you can see at `cool-blog/_site/.well-known/acme-challenge/5TBu788fW0tQ5EOwZMdu1Gv3e9C33gxjV58hVtWTbDM.html`, with no extra HTML, just the token in plain text. As we are using the `permalink` attribute in the front matter, you can name this file anyway you want and put it anywhere, too. Note that the behaviour of the `permalink` attribute has **changed** from Jekyll 2 to Jekyll 3, so make sure you have Jekyll 3.x installed. If you're not using version 3 of Jekyll or if you're using a different tool, just create the same file in the exact path, like `cool-blog/.well-known/acme-challenge/5TBu788fW0tQ5EOwZMdu1Gv3e9C33gxjV58hVtWTbDM.html` or an equivalent path in your static site generator of choice. Here we'll call it `letsencrypt-setup.html` and place it in the root folder of the blog. In order to check that everything is working as expected, start a local server with `jekyll serve` in a separate terminal window and try to access the URL:

```
$ curl http://localhost:4000/.well-known/acme-challenge/5TBu788fW0tQ5EOwZMdu1Gv3e9
C33gxjV58hVtWTbDM

# response:

5TBu788fW0tQ5EOwZMdu1Gv3e9C33gxjV58hVtWTbDM.ewlbSYgvIxVOqiP1lD2zeDKWBGEZMRfO_4kJyL
RP_4U
```

Note that I just replaced the `http://YOURDOMAIN.org` (from the `letsencrypt-auto` instructions) with `http://localhost:4000`. Everything is working fine, so we just need to upload the new file to GitLab:

```
$ git add letsencrypt-setup.html
$ git commit -m "add letsencypt-setup.html file"
$ git push
```

Once the build finishes, test again if everything is working well:

```
# Note that we're using the actual domain, not localhost anymore
$ curl http://YOURDOMAIN.org/.well-known/acme-challenge/5TBu788fW0tQ5EOwZMdu1Gv3e9
C33gxjV58hVtWTbDM
```

If you get a `404 page not found`, check if you missed any step, or get in touch in the comments below.

Now that everything is working as expected, go back to the terminal window that's waiting for you and hit `ENTER`. This instructs the Let's Encrypt's servers to go to the URL we just created. If they get the response they were waiting for, we've proven that we actually own the domain and now they'll send you the TLS certificates. After a while it responds:

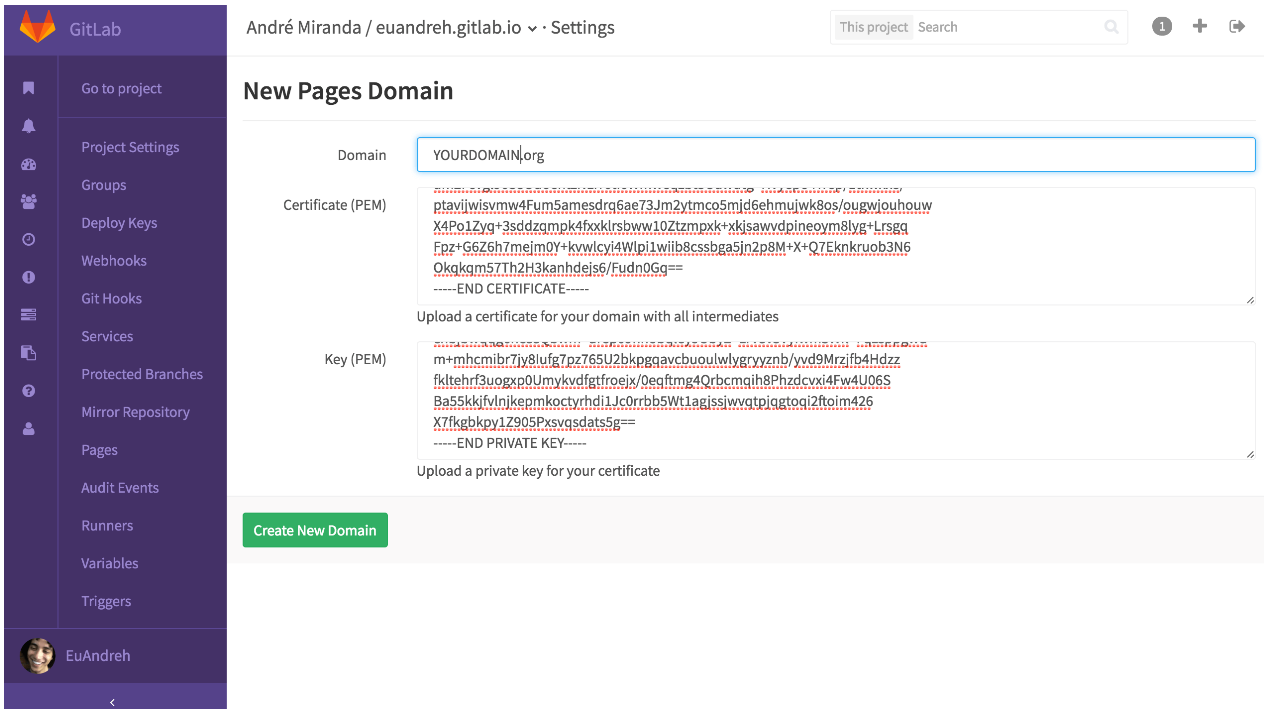
```
IMPORTANT NOTES:
 - Congratulations! Your certificate and chain have been saved at
   /etc/letsencrypt/live/YOURDOMAIN.org/fullchain.pem. Your cert will
   expire on 2016-07-04. To obtain a new version of the certificate in
   the future, simply run Let's Encrypt again.
 - If you like Let's Encrypt, please consider supporting our work by:

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

Success! We have correctly acquired a free TLS certificate for our domain!

Note, however, that like any other TLS certificate, it has an expiration date, and in the case of certificates issued by Let's Encrypt, the certificate will remain valid for 90 days. When you finish setting up, just put in your calendar to remember to renew the certificate in time, otherwise it will become invalid, and the browser will reject it.

Now we just need to upload the certificate and the key to GitLab. Go to **Settings** -> **Pages** inside your project, remove the old `CNAME` and add a new one with the same domain, but now you'll also upload the TLS certificate. Paste the contents of `/etc/letsencrypt/live/YOURDOMAIN.org/fullchain.pem` (you'll need `sudo` to read the file) to the "Certificate (PEM)" field and `/etc/letsencrypt/live/YOURDOMAIN.org/privkey.pem` (also needs `sudo`) to the "Key (PEM)" field.



New Pages Domain

Domain: YOURDOMAIN.org

Certificate (PEM):

```
ptavijwisvmw4Fum5amesdrq6ae73Jm2ytmco5mjd6ehmujwk8os/ougwjouhouw
X4Po1Zyq+3sddzqmpk4fxxklrsbww10Ztzmpxk+xkjsawvdpineoym8lyg+Lrsgq
Fpz+G6Z6h7mejm0Y+kvwlcyi4Wlpi1wiib8cssbga5jn2p8M+X+Q7Eknkruob3N6
Okqkqm57Th2H3kanhdejs6/Fudn0Gq==
-----END CERTIFICATE-----
```

Upload a certificate for your domain with all intermediates

Key (PEM):

```
m+mhcmibr7jy8Iufg7pz765U2bkpgqavcbuoulwlygryyznb/yvd9Mrzjfb4Hdzz
fkltehrf3uogxp0Umykvdfgtfroejx/0eqftmg4Qrbcmqih8Phzdcvxi4Fw4U06S
Ba55kkjfvlnjkepmkoctyrhdi1Jc0rrbb5Wt1agjssjwvqtpjqgtoqi2ftoim426
X7fkgbkpy1Z905Pxsvqsdats5g==
-----END PRIVATE KEY-----
```

Upload a private key for your certificate

And you're done! You now have a fully working HTTPS website:

```
$ curl -vX HEAD https://YOURDOMAIN.org/
#
# starting connection
#
* TLS 1.2 connection using TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
* Server certificate: YOURDOMAIN.org
* Server certificate: Lets Encrypt Authority X3
* Server certificate: DST Root CA X3
```

Redirecting

Everything is working fine, but now we have an extra concern: we have two working versions of our website, both HTTP **and** HTTPS. We need a way to redirect all of our traffic to the HTTPS version, and tell search engines to do the same.

Search Engines

Instructing the search engines is really easy: just tell them that the HTTPS version is the "canonical" version, and they send all the users to it. And how do you do that? By adding a `link` tag to the header of the HTML:

```
<link rel="canonical" href="https://YOURDOMAIN.org/specific/page" />
```

Adding this to every header on a blog tells the search engine that the correct version is the HTTPS one, and they'll comply.

Internal links

Remember to use HTTPS for your CSS or JavaScript file URLs, because when the browser accesses a secure website that relies on an insecure resource, it may block that resource.

It is considered a good practice to use the protocol-agnostic path:

```
<link rel="stylesheet" href="//YOURDOMAIN.org/styles.css" />

<script src="//YOURDOMAIN.org/script.js"></script>
```

JavaScript-based redirect

There is, however, a case where the user specifically types in the URL **without** using HTTPS, and they'll access the HTTP version of your website.

The correct way of handling that would be to respond with a 301 "Moved permanently" HTTP code, and the browser would remember it for the next request. However, that's not a possibility we have here, since we're running on GitLab's servers.

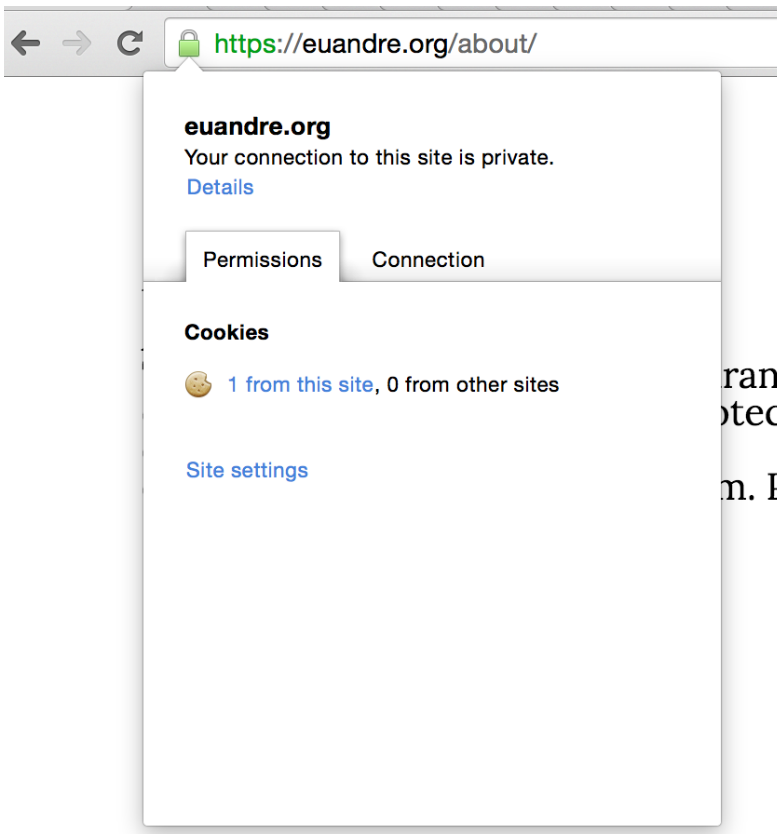
A small hack you can do is to redirect your users with a bit of JavaScript code:

```
var host = "YOURDOMAIN.org";
if ((host == window.location.host) && (window.location.protocol != 'https:')) {
  window.location = window.location.toString().replace(/^http:/, "https:");
}
```

This redirects the user to the HTTPS version, but there are a few problems with it:

1. a user could have JavaScript disabled, and would not be affected by that;
2. an attacker could simply remove that code and behave as a **Man in the Middle**;
3. the browser won't remember the redirect instruction, so every time the user types that same URL, the website will have to redirect him/her again.

Wrap up


https://euandre.org/about/

euandre.org
Your connection to this site is private.
Details

Cookies

1 from this site, 0 from other sites

Site settings

That's how easy it is to have a free HTTPS-enabled website. With these tools, I see no reason not to do it.

If you want to improve GitLab's support for Let's Encrypt, you can discuss and contribute in issues #474, #467 and #472 from GitLab EE. They are open to merge requests!

There's an excellent talk by Pierre Far and Ilya Grigorik on HTTPS where you can learn more about it.

If you want to check the status of your HTTPS enabled website, SSL Labs offers a free online service that "performs a deep analysis of the configuration of any SSL web server on the public Internet".

This article is based on Paul Wakeford's post.

I hope it helps you :)

- **fan** • 2 years ago

  We plan to introduce built-in support for Let's Encrypt: https://gitlab.com/gitlab-o.... The plan is to enable automatic management of certificates.

  - 9
  - •
  - Reply
  - •
  - Share ›

Justin_Aiken ayufan • 9 months ago

Until you do... Here's a jekyll plugin to automate most of the process:

https://github.com/JustinAi...

- •
- Reply
- •
- Share ›

- **djm** • a year ago

  If someone get 404 error because .html extension for file - I had the same and here is my Stackoverflow question with answer http://stackoverflow.com/qu...

  - 4
  - •
  - Reply
  - •
  - Share ›

Marthym djm • 6 months ago

Or you can just add `/` at the end of permalink url :

```
---
layout: null
permalink: /.well-known/acme-challenge/5TBu788fW0tQ5EOwZMdu1Gv3e9C33gxjV58hVtWTbDM/
---

5TBu788fW0tQ5EOwZMdu1Gv3e9C33gxjV58hVtWTbDM.ewlbSYgvIxVOqiP1lD2zeDKWBGEZMRfO_4kJyLRP_4U
```

- •
- Reply
- •
- Share ›

Jake Champion • 2 years ago

In the paul irish blog post on protocol agnostic links it actually states:

Update 2014.12.17:

Now that SSL is encouraged for everyone and doesn’t have performance concerns, this technique is now an anti-pattern. If the asset you need is available on SSL, then always use the https:// asset.

Allowing the snippet to request over HTTP opens the door for attacks like the recent Github Man-on-the-side attack. It’s always safe to request HTTPS assets even if your site is on HTTP, however the reverse is not true.

- 3
- •
- Reply

Share ›

Eric Mill Jake Champion • 2 years ago

Yeah, this is not recommended. Strongly recommend updating the post so as not to encourage more unnecessary potential HTTP connections.

4

Reply

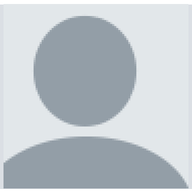
Share ›

André Miranda Eric Mill • 2 years ago

Thanks Jake and Eric for the tip. I'll update it.

Reply

Share ›



jj André Miranda • a year ago

Three months in and still not updated. Just a heads up.

- Reply
- Share ›

Jose Torres jj • a year ago

Thanks for the reminder, I've requested this at https://gitlab.com/gitlab-c... for us to follow up with the update.

- Reply
- Share ›

sickmartian • a year ago

I was able to secure the bare domain with this (sickmartian.com), thanks a lot! I'm having problems trying to secure www.sickmartian.com and trackendar.sickmartian.com.
I've got certificates for www and trackendar on my gitlab page but if I just use CNAME pointing to my gitlab.io repo page I get server the bare domain certificate... so I get a security error.
I'm able to use a DNS redirect record from http://www.sickmartian.com to https://sickmartian.com and that's great, but I'm unable to redirect https://www.sickmartian.com to https://sickmartian.com which really bother me.
So, in conclusion I'm unable to either use certificates on subdomains if I already have the bare domain with a certificate. Is this a constraint or am I just missing something?
I'm also unable to redirect from https to https (using namecheap, I haven't found anything about

this restriction.. I've found this restriction for other providers however).. Does anyone know if there is any restriction here I'm missing?

- 2
- •
- Reply
- •
- Share ›

dhruvkar • a year ago

This works well for me. I wanted to forward all urls to https://MYDOMAIN.ORG. I ran the letsencrypt tool for my root domain, and have a 301 redirect for the www to https://MYDOMAIN.ORG. But I still have one use case that's not working:

http://WWW.MYDOMAIN.ORG => https://MYDOMAIN.ORG - works!
http://MYDOMAIN.ORG => https://MYDOMAIN.ORG - works!
https://WWW.MYDOMAIN.ORG => Connection was interrupted - doesn't work :(

What can I do to get that last redirect working correctly?

- 2
- •
- Reply
- •
- Share ›

Abubakar Siddiq Ango dhruvkar • a year ago

It is recommended that you also run the letsencrypt tool for the www.MYDOMAIN.org also.

- 1
- •
- Reply

- Share ›

dhruvkar Abubakar Siddiq Ango • a year ago

Yes, I figured it out and added a certificate for www as well. However, I can't get the redirect to work from https://www.mydomain.org => https://mydomain.org. Suggestions.

- Reply
- Share ›

avrischneider dhruvkar • a year ago

I ran into this problem with namecheap. For URL redirect to work, there needs to be a web-server listening for an HTTP/HTTPS connection, and respond with the 302/301 HTTP header. The record that is returned by the DNS server is actually an IP address of namecheap's server. But it does not listen on port 443 for HTTPS connections - so it can't do any 302/301 redirects - and you get a connection refused error. I contacted support and they said they don't support HTTP redirect to SSL sites, and this needs to be solved at the hosting provider's end (gitlab pages in this case...).

- 1
- Reply
- Share ›

dhruvkar avrischneider • a year ago

That's the conclusion I came to as well. Need access to the server to do those redirects.

- 1
- •
- Reply
- •
- Share ›

animesh • a month ago

I just renewed the certificate for the first time and found the process to be a bit clunky. I wish this could be automated somehow.

1. Renewed the certificate using

sudo certbot certonly -a manual -d mydomain.com
2. Uploaded the acme-challenge file to the repo with a git push as usual
3. Deleted and recreated the domain in settings>pages as there is no way to update the domain in place.
4. Updated the .pem files as instructed.

Are there any plans to automate this process in the future? That would be a fantastic addition.

- 1
- •
- Reply
- •
- Share ›

GitLab animesh • a month ago

We have an open feature proposal about it in https://gitlab.com/gitlab-o... Feel free to pop in

- •
- Reply
- •
- Share ›

codesimple • a year ago

This is great and I now have it working for one of my projects, but I found that just pasting your "privkey.pem" into the "Key (PEM)" field doesn't work. You first need to convert it to PKCS#1 format (see https://gitlab.com/gitlab-c.... Otherwise it appears to have worked but the certificate doesn't get used when you visit the domain.

- 1
- •
- Reply
- •
- Share ›

Paul John Diwa codesimple • 3 months ago

Maybe this should be added in the guide in some degree. This solved by problem too. https://stackoverflow.com/a...

- Reply
- Share ›

Abubakar Siddiq Ango codesimple • a year ago

Thank you very much for the information.

- Reply
- Share ›

Virtua Creative • 2 years ago

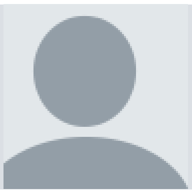
Congrats **André Miranda**! :) Great post!

- 1
- Reply
- Share ›

André Miranda Virtua Creative • 2 years ago

Thanks :)

- •
- Reply
- •
- Share ›



rosefour • 2 years ago

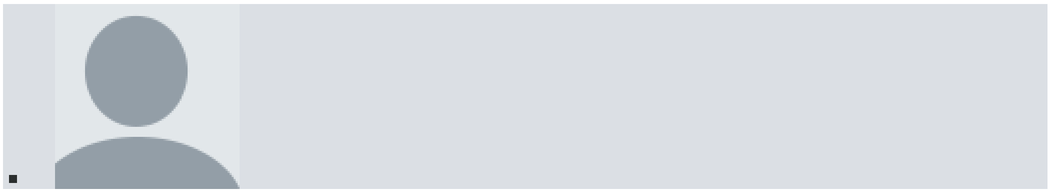
Thank you for posting this tutorial. Great post!
I wonder, does it work well with CloudFlare DNS?

- 1
- •
- Reply
- •
- Share ›

ayufan rosefour • 2 years ago

Yes. You can use CloudFlare DNS to also have a distributed CDN. It will reduce load times of your pages. Worth noting is that GitLab Pages do support HTTP/2 if HTTPS is used, so your loading times should be really good already.

- 1
- •
- Reply
- •
- Share ›



rosefour ayufan • 2 years ago

Thank you for the response.
I will try it. I have a question, the custom page that includes Let's Encrypt token should be written in markdown or no format?

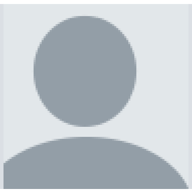
- •
- Reply
- •
- Share ›

André Miranda rosefour • 2 years ago

It should be in either plain text of HTML. If you write it in markdown, Jekyll will process it and put the token inside

tags.

- •
- Reply
- •
- Share ›



rosefour André Miranda • 2 years ago

I just tried it. But, when I save it as markdown, Jekyll will generate HTML file of Let's Encrypt token. It will fail when you access http://localhost:4000/.well-know/acme-challenge/token. It only success when you access http://localhost:4000/.well-know/acme-challenge/token.html (.html is added).

- 1
- •
- Reply
- •
- Share ›

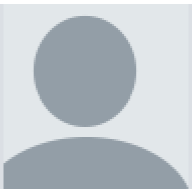
André Miranda rosefour • 2 years ago

With or without using markdown, it should generate an HTML file because of the permalink attribute on the front-matter.

I'm not sure why it is not working: for me it works both with and without ".html".

Is your "permalink" accurate?

- •
- Reply
- •
- Share ›



rosefour André Miranda • 2 years ago

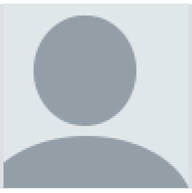
Yes, it is. When I access it without ".html" (http://mydomain.com/.well-k..., the response gives 404 page. But, when I access it with ".html", it success and the response is token with extra
tag.

- •
- Reply
- •
- Share ›

André Miranda rosefour • 2 years ago

Does Let's Encrypt also fails to fetch the token from GitLab Pages? Does it show any error message?

- •
- Reply
- •
- Share ›



rosefour André Miranda • 2 years ago

No, it doesn't. It success when you access the token url with ".html". Sorry, no extra tag. It shows plain token text.

- Reply
- Share ›

André Miranda rosefour • 2 years ago

Hmm, which extra tag?

- Reply
- Share ›

Virtua Creative rosefour • 2 years ago

I tested CloudFlare (free account) with a custom domain for a GitLab Pages project. If you use CloudFlare, as far as I know, you won't need any certificate at all! They provide us with a free SSL/TLS issued by Comodo CA, and it supports wildcard (*.example.com).

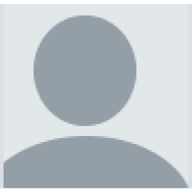
- Reply
- Share ›

André Miranda Virtua Creative • 2 years ago

That is half true: you don't need any other certificate only if you care solely about the communication between the client and CloudFlare. Because when doing this, the communication between CloudFlare and GitLab's server will still be unencrypted and vulnerable to Man in the Middle attacks in the same way.

But if your server already has a certificate and you put CloudFlare in the middle, everything is fine.

- 2
- •
- Reply
- •
- Share ›



Billy André Miranda • 2 years ago

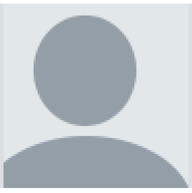
I have no luck integrate GitLab Pages, Let's Encrypt, and CloudFlare. I already set mydomain.com to username.gitlab.io (CNAME) and SSL to Full (Strict). But, whenever I access my mydomain.com it gives an error.

- •
- Reply
- •
- Share ›

ayufan Billy • 2 years ago

Did you add certificate for mydomain.com on GitLab? If not you should set SSL setting on CloudFlare to Off, cause GitLab can't secure page for your domain.

- 1
- •
- Reply
- •
- Share ›



Billy ayufan • 2 years ago

Yes, I did. I have added certificate for mydomain.com on GitLab.
Sometimes CloudFlare gives an error because domain doesn't match with certificate. Sometimes it gives success but it's using Universal SSL not Let's Encrypt.

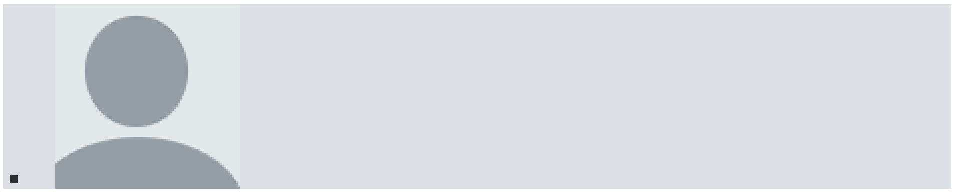
- •
- Reply
- •
- Share ›

ayufan Billy • 2 years ago

Please create a confidential issue in https://gitlab.com/gitlab-c... with information about your build and your domain to help us check what is a problem.

- •
- Reply
- •

- Share ›



Billy ayufan • 2 years ago

Sure, I will report it later. Thank you **ayufan**.

- Reply
- Share ›

Virtua Creative Billy • 2 years ago

Wouldn't that be the case you need to add a DNS A record instead of CNAME? See https://about.gitlab.com/20...

- Reply
- Share ›

Virtua Creative André Miranda • 2 years ago

Really? How does it works exactly? Does that means that I'll need both certificates? One from CloudFlare and another on GitLab Pages server?

- Reply
- Share ›

ayufan Virtua Creative • 2 years ago

There's second mode. That GitLab Pages will serve unencrypted content (only over HTTP), but the encryption will be on CloudFlare. The CloudFlare needs to have an SSL setting set to Off. It means that CloudFlare serves the page over HTTPS, but connect to GitLab Pages with HTTP. Thus it doesn't require HTTPS on GitLab Side.

- Reply
- Share ›

André Miranda Virtua Creative • 2 years ago

Two certificates will be involved: one in GitLab's server and one in CloudFlare's servers. But CloudFlare will handle all TLS stuff by themselves, you'll have to do just the same work on TLS, with or without CloudFlare.

- Reply

- Share ›

Virtua Creative André Miranda • 2 years ago

I think I got it. Thanks!

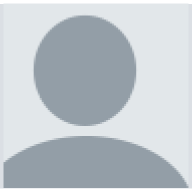
- 1
- •
- Reply
- •
- Share ›

André Miranda rosefour • 2 years ago

Thanks :)

You can follow the discussion on adding CloudFlare: https://gitlab.com/gitlab-c...

- •
- Reply
- •
- Share ›



rosefour André Miranda • 2 years ago

Thank you for sharing.

- 1
- •
- Reply
- •
- Share ›

- **Paul John Diwa** • 3 months ago

  I used zerossl, to provide a LetsEncrypt Certificate, but then my https site doesnt work. Can someone help me?

  see more

  - •
  - Reply
  - •
  - Share ›

- **Matija Čupić** Paul John Diwa • 3 months ago

  Could you provision a LE certificate manually and try that? Also make sure you're using the certificate in its entirety when adding it to the UI.

  - •
  - Reply
  - •
  - Share ›

Paul John Diwa Matija Čupić • 3 months ago

It's working now, I converted the private.key as @codesimple said. Thanks for the reply though. Maybe what I was missing was the modulus part which was added after doing conversion.

- Reply
- Share ›

animesh • 4 months ago

Thanks! Can we remove the .well-known/acme-challenge/... file ?

Edit: I deleted it as it is a one time thing.

- Reply
- Share ›