

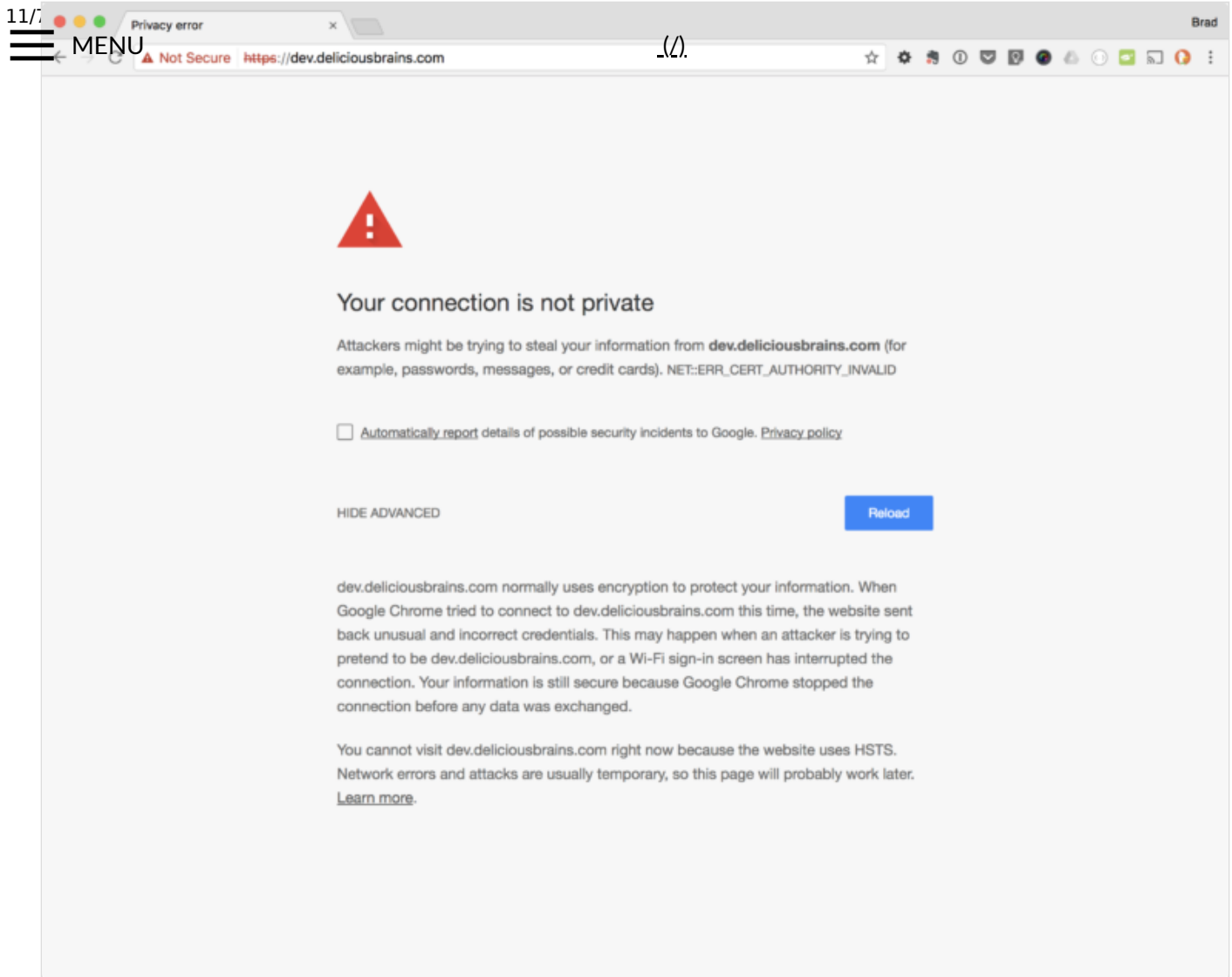


# How to Create Your Own SSL Certificate Authority for Local HTTPS Development

#(<https://deliciousbrains.com/ssl-certificate-authority-for-local-https-development/>). Published Jul 25, 2017



In my last article (<https://deliciousbrains.com/https-locally-without-browser-privacy-errors/>) I described how to generate your own self-signed SSL certificates and add them to macOS Keychain so that your browser doesn't give you a privacy error.



Soon after it was published, Ross McKay (<https://twitter.com/webawareros>) made a very interesting comment (<https://deliciousbrains.com/https-locally-without-browser-privacy-errors/#comment-3285792308>) on that article:

“ If you have a few servers you need to do this with, you can just create yourself a CA (Certifying Authority) certificate and load that instead. Then your self-signed certs, signed by your CA cert, will all be accepted without you needing to load each one.



So basically he's saying that I can be a certificate authority (CA) like Let's Encrypt, Amazon, Verisign, Comodo, etc but just for my local network. How did I not know about that? So cool. But how does it work exactly?

## How It Works

After some research I think I get it now. To request a certificate from a CA like Verisign, you send them a Certificate Signing Request (CSR), and they give you a certificate in return that they signed using their root certificate and private key. All browsers have a copy (or access a copy from the operating system) of Verisign's root certificate, so the browser can verify that your certificate was signed by a trusted CA.

That's why when you generate a self-signed certificate the browser doesn't trust it. It's self-signed. It hasn't been signed by a CA. But as Ross pointed out (<https://deliciousbrains.com/https-locally-without-browser-privacy-errors/#comment-3285792308>), we can generate our own root certificate and private key, add the root certificate to all the devices we own just once, and then all certificates that we generate and sign will be inherently trusted.

## Becoming a (tiny) Certificate Authority

It's kind of ridiculous how easy it is to generate the files needed to become a certificate authority. It only takes two commands. First, we generate our private key:

```
openssl genrsa -des3 -out myCA.key 2048
```

You will be prompted for a pass phrase, which I recommend not skipping and keeping safe. The pass phrase will prevent anyone who gets your private key from generating a root certificate of their own. Output should look like this:

```
Generating RSA private key, 2048 bit long modulus
.....+++
.....+++
e is 65537 (0x10001)
Enter pass phrase for myCA.key:
Verifying - Enter pass phrase for myCA.key:
```

Then we generate a root certificate:

```
openssl req -x509 -new -nodes -key myCA.key -sha256 -days 1825 -out myCA.pem
```

You will be prompted for the pass phrase of your private key (that you just choose) and a bunch of questions. The answers to those questions aren't that important. They show up when looking at the certificate, which you will almost never do. I suggest making the Common Name something

that you'll recognize as your root certificate in a list of other certificates. That's really the only thing that matters.

Enter pass phrase for myCA.key:

You are about to be asked to enter information that will be incorporated into your certificate request.

What you are about to enter is what is called a Distinguished Name or a DN.

There are quite a few fields but you can leave some blank

For some fields there will be a default value,

If you enter '.', the field will be left blank.

-----

Country Name (2 letter code) [AU]:CA

State or Province Name (full name) [Some-State]:Nova Scotia

Locality Name (eg, city) []:Truro

Organization Name (eg, company) [Internet Widgits Pty Ltd]:Delicious Brains Inc

Organizational Unit Name (eg, section) []:

Common Name (e.g. server FQDN or YOUR name) []:Delicious Brains

Email Address []:noreply@deliciousbrains.com (http://deliciousbrains.com)

You should now have two files: myCA.key (your private key) and myCA.pem (your root certificate).



Congratulations, you're now a CA. Sort of.

To become a real CA, you need to get your root certificate on all the devices in the world. Let's start with the ones you own.

## Installing Your Root Certificate

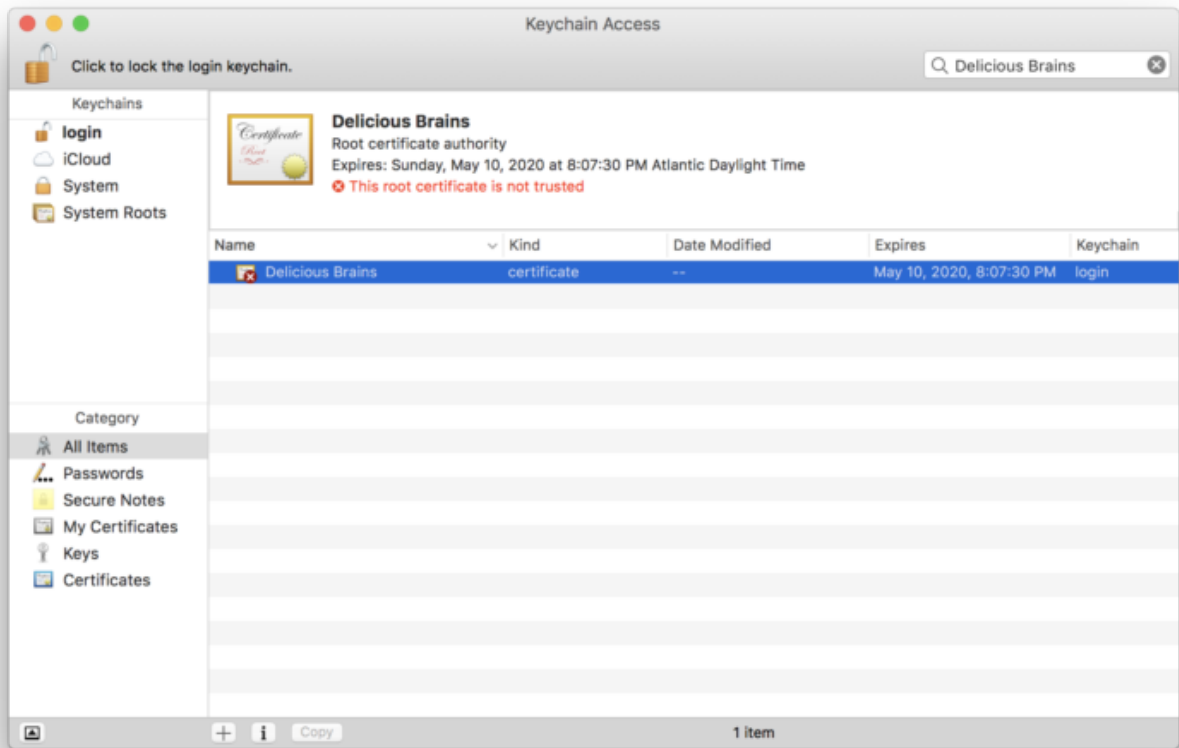
We need to add the root certificate to any laptops, desktops, tablets, and phones that will be accessing your HTTPS sites. This can be a bit of a pain, but the good news is that we only have to do it once. Once our root certificate is on each device, it will be good until it expires.

### Adding the Root Certificate to macOS Keychain

1. Open the macOS Keychain app
2. Go to *File > Import Items...*
3. Select your private key file (i.e. myCA.pem)



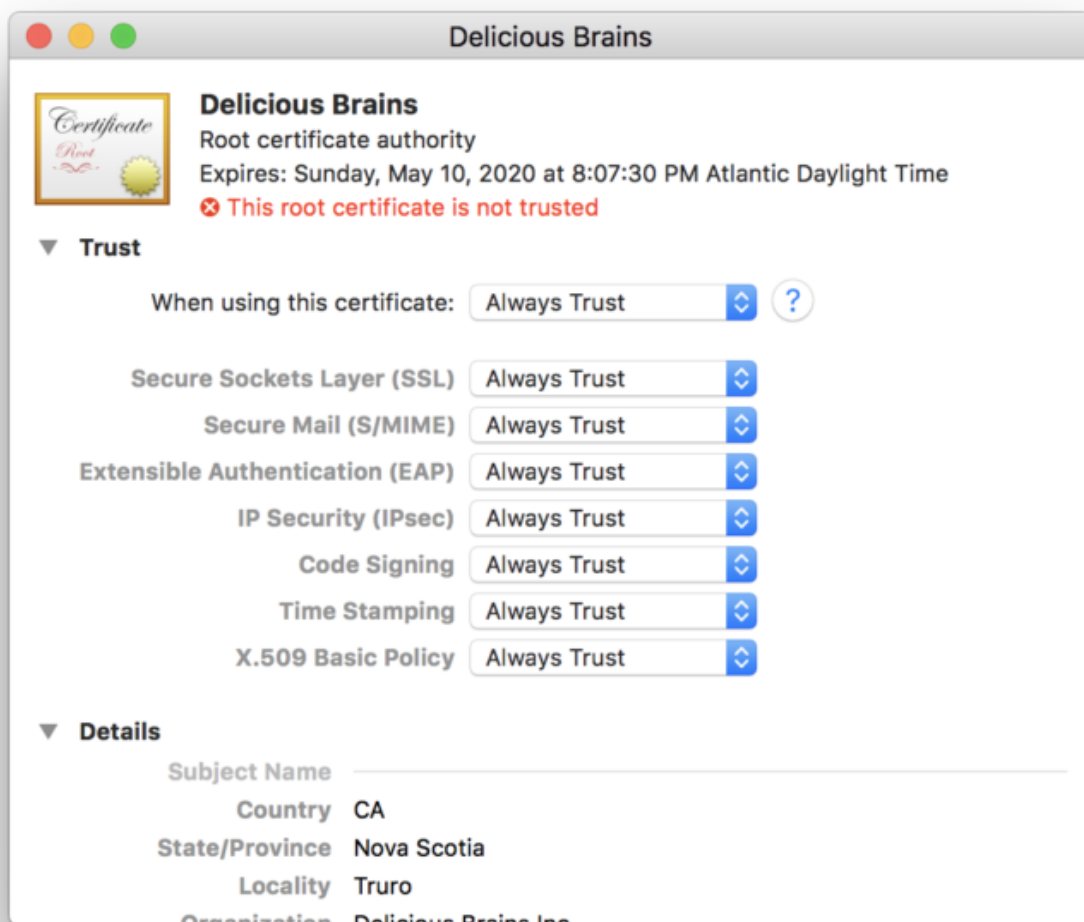
4. Search for whatever you answered as the **Common Name** name above  
..(.)



5. Double click on your root certificate in the list

6. Expand the *Trust* section

7. Change the When using this certificate: select box to "Always Trust"



8. Close the certificate window
9. It will ask you to enter your password (or scan your finger), do that
10. 🎉 Celebrate!

## Creating CA-Signed Certificates for Your Dev Sites

Now that we're a CA on all our devices, we can sign certificates for any new dev sites that need HTTPS. First, we create a private key:

```
openssl genrsa -out dev.mergebot.com.key 2048
```

Then we create a CSR:

```
openssl req -new -key dev.mergebot.com.key -out dev.mergebot.com.csr
```

You'll get all the same questions as you did above and, again, your answers don't matter. In fact, they matter even less because you won't be looking at this certificate in a list next to others.



More about to be asked to enter information (that will be incorporated into your certificate request).

What you are about to enter is what is called a Distinguished Name or a DN.

There are quite a few fields but you can leave some blank

For some fields there will be a default value,

If you enter '.', the field will be left blank.

-----

Country Name (2 letter code) [AU]:CA

State or Province Name (full name) [Some-State]:Nova Scotia

Locality Name (eg, city) []:Truro

Organization Name (eg, company) [Internet Widgits Pty Ltd]:Delicious Brains Inc

Organizational Unit Name (eg, section) []:

Common Name (e.g. server FQDN or YOUR name) []:Mergebot

Email Address []:noreply@mergebot.com (<http://mergebot.com>)

Please enter the following 'extra' attributes

to be sent with your certificate request

A challenge password []:

An optional company name []:

Next we'll create the certificate using our CSR, the CA private key, the CA certificate, and a config file, but first we need to create that config file.

The config file is needed to define the Subject Alternative Name (SAN) extension we discussed in my last article (<https://deliciousbrains.com/https-locally-without-browser-privacy-errors/#creating-self-signed-certificate>). I created a new file named

`dev.mergebot.com (http://dev.mergebot.com).ext` and added the following contents:

```
authorityKeyIdentifier=keyid,issuer
basicConstraints=CA:FALSE
keyUsage = digitalSignature, nonRepudiation, keyEncipherment, dataEncipherment
subjectAltName = @alt_names

[alt_names]
DNS.1 = dev.mergebot.com (http://dev.mergebot.com)
DNS.2 = dev.mergebot.com (http://dev.mergebot.com).192.168.1.19.xip.io
```

We're using a different config file for the SAN than in my last article (<https://deliciousbrains.com/https-locally-without-browser-privacy-errors/#creating-self-signed-certificate>) because we'll be running the `openssl x509` command instead of the `openssl req` command. From what I understand, the `x509` command is needed to do the signing with the root certificate and private key. Again, I found this example config file (<https://stackoverflow.com/a/43665244>) on Stack Overflow and it seems to work.



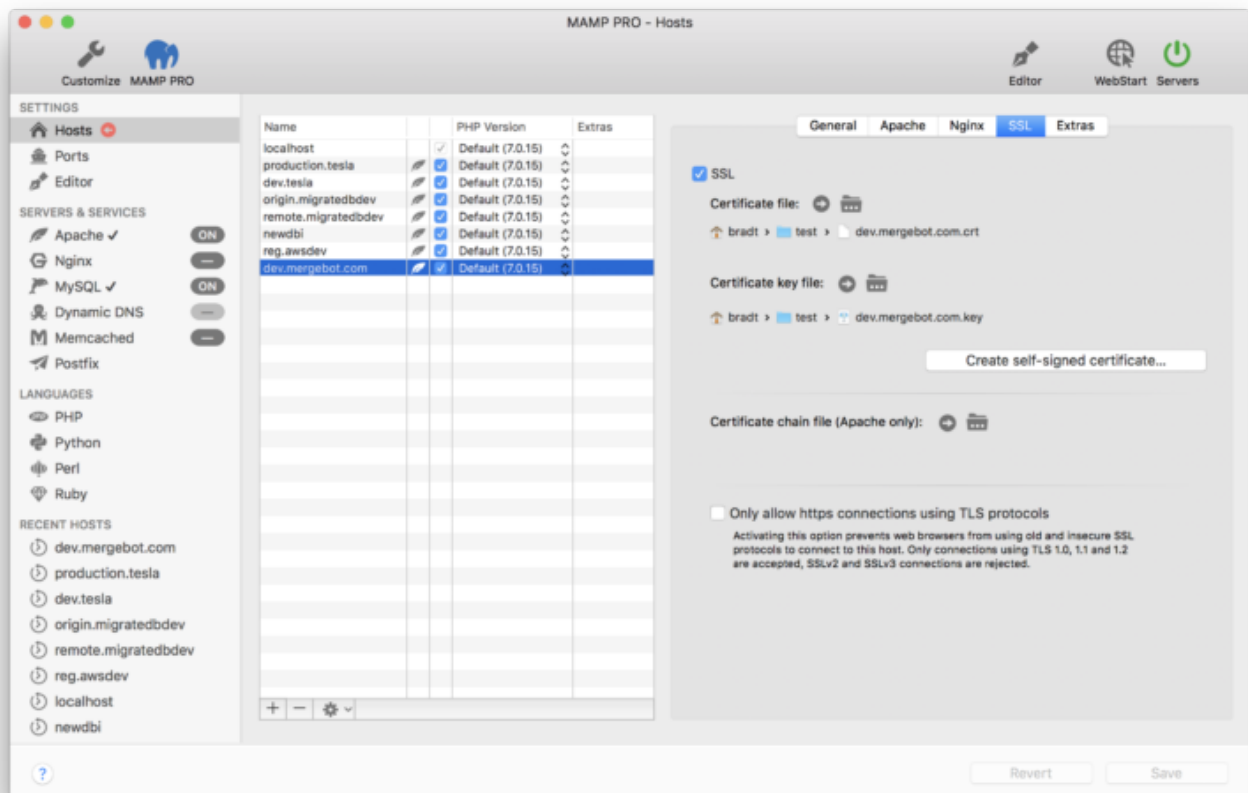
Now we run the command to create the certificate:  
MENU (L)

```
openssl x509 -req -in dev.mergebot.com.csr -CA myCA.pem -CAkey myCA.key -CAcreateserial \
-out dev.mergebot.com.crt -days 1825 -sha256 -extfile dev.mergebot.com.ext
```

You should be prompted for your CA private key pass phrase.

I now have three files: dev.mergebot.com (http://dev.mergebot.com).key (the private key), dev.mergebot.com (http://dev.mergebot.com).csr (the certificate signing request), and dev.mergebot.com (http://dev.mergebot.com).crt (the signed certificate).

I can now configure my web server with the private key and the certificate. If you're running a Linux server, you can use the instructions in our Hosting WordPress Yourself series (<https://deliciousbrains.com/hosting-wordpress-yourself-ssl-spdy/#installing-the-certificate>). If you're using MAMP, you can select the certificate and key files using the UI:




For any other dev sites, we can just repeat this last part of creating a certificate. No need to install any new certificates on any of my devices. Much better.

## Conclusion

After going through the process of setting up my own CA and signing certificates with it, there's no question this is a better way to go than creating individual self-signed certificates as described in my last article (<https://deliciousbrains.com/https-locally-without-browser-privacy-errors/>). The



 fact that you only have to setup the CA and add it to your devices once means you save a bunch of time for each new certificate you create in the future. It's a no-brainer to go this route.

Have you tried setting up a CA of your own? Maybe you've been using your own CA for years? Are there any drawbacks that I missed? Let me know in the comments below.

---

This entry was tagged [SSL](https://deliciousbrains.com/tag/ssl/), [HTTPS](https://deliciousbrains.com/tag/https/), [Development Tips](https://deliciousbrains.com/tag/development-tips/), [Development Environment](https://deliciousbrains.com/tag/development-environment/), [MAMP](https://deliciousbrains.com/tag/mamp/), [Certificate Authority](https://deliciousbrains.com/tag/certificate-authority/), [OpenSSL](https://deliciousbrains.com/tag/openssl/).

---

## ABOUT THE AUTHOR



Brad Touesnard (<https://deliciousbrains.com/author/bradt/>)

As founder of Delicious Brains Inc, Brad has worn many hats. He now spends most of his time managing the product teams and growing the business. Before starting this company, Brad was a freelance web developer, specializing in front-end development.

@bradt (<https://twitter.com/bradt>)

bradt.ca (<https://bradt.ca/>)

# Authors of the ingenious WP Migrate DB Pro @dliciousbrains are at it again with [WP Offload Media]. Yes please.

Lara Schenck

@laras126

GET WP OFFLOAD MEDIA >

## Important Update

When you log in with Disqus, we process personal data to facilitate your authentication and posting of comments. We also store the comments you post and those comments are immediately viewable and searchable by anyone around the world.

- ☐ I agree to Disqus' Terms of Service
- ☐ I agree to Disqus' processing of email and IP address, and the use of cookies, to facilitate my authentication and posting of comments, explained further in the Privacy Policy

Proceed













SIGN UP FOR THE LATEST PRODUCT NEWS AND UPDATES

Enter your email address

SUBSCRIBE

## PLUGINS



### WP Migrate DB Pro

(<https://deliciousbrains.com/wp-migrate-db-pro/>).

VIEW PLUGIN ([HTTPS://DELICIOUSBRAINS.COM/WP-MIGRATE-DB-PRO/](https://deliciousbrains.com/wp-migrate-db-pro/))

Migration can be a beautiful thing. Copy your database from one WordPress install to another with one click in your dashboard.



### WP Offload Media

(<https://deliciousbrains.com/wp-offload-media/>).

VIEW PLUGIN ([HTTPS://DELICIOUSBRAINS.COM/WP-OFFLOAD-MEDIA/](https://deliciousbrains.com/wp-offload-media/))


Speed up your WordPress site by offloading your media to Amazon S3 or DigitalOcean Spaces

### Technical Support

To request support, visit the Help tab inside the plugin. You can also find the support email address by logging into [My Account](https://deliciousbrains.com/my-account/) (<https://deliciousbrains.com/my-account/>).

### General Inquiries

[nom@deliciousbrains.com](mailto:nom@deliciousbrains.com) (<mailto:nom@deliciousbrains.com>).

 [Home \(https://deliciousbrains.com/\)](https://deliciousbrains.com/)  
**MENU**  
[WP Migrate DB Pro \(https://deliciousbrains.com/wp-migrate-db-pro/\)](https://deliciousbrains.com/wp-migrate-db-pro/)  
[WP Offload Media \(https://deliciousbrains.com/wp-offload-media/\)](https://deliciousbrains.com/wp-offload-media/)  
[About \(https://deliciousbrains.com/about/\)](https://deliciousbrains.com/about/)  
[Careers \(https://deliciousbrains.com/about/#careers\)](https://deliciousbrains.com/about/#careers)

[Giving Back \(https://deliciousbrains.com/about/#giving-back\)](https://deliciousbrains.com/about/#giving-back)  
[Contact Us \(https://deliciousbrains.com/contact-us/\)](https://deliciousbrains.com/contact-us/)  
[Blog \(https://deliciousbrains.com/blog/\)](https://deliciousbrains.com/blog/)  
[Affiliates \(https://deliciousbrains.com/affiliates/\)](https://deliciousbrains.com/affiliates/)  
[My Account \(https://deliciousbrains.com/my-account/\)](https://deliciousbrains.com/my-account/)

© 2013–2018 DELICIOUS BRAINS, INC. ALL RIGHTS RESERVED. [PRIVACY POLICY \(HTTPS://DELICIOUSBRAINS.COM/PRIVACY-POLICY/\)](https://deliciousbrains.com/privacy-policy/), | [TERMS AND CONDITIONS \(HTTPS://DELICIOUSBRAINS.COM/TERMS-CONDITIONS/\)](https://deliciousbrains.com/terms-conditions/)

  [\(https://github.com/deliciousbrains\)](https://github.com/deliciousbrains)   [\(https://www.facebook.com/dliciousbrains\)](https://www.facebook.com/dliciousbrains)  
  [\(https://twitter.com/dliciousbrains\)](https://twitter.com/dliciousbrains)