Generating multiple certificates with Letsencrypt from a single instance

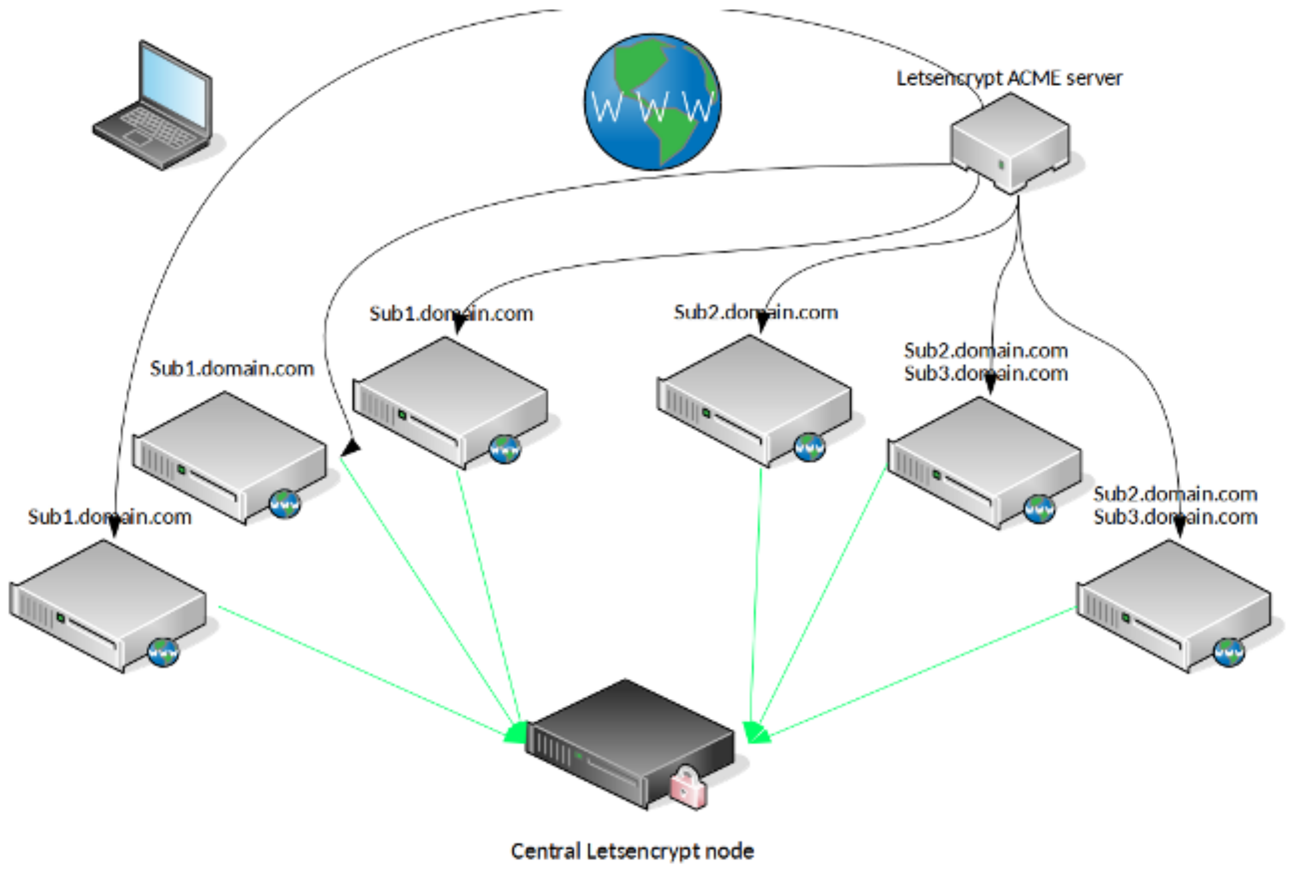
Recently I was discussing with some people about TLS everywhere, and we then started to discuss about the Letsencrypt initiative. I had to admit that I just tested it some time ago (just for "fun") but I suddenly looked at it from a different angle : while the most used case is when you install/run the letsencrypt client on your node to directly configure it, I have to admit that it's something I didn't want to have to deal with. I still think that proper web server configuration has to happen through cfgmgmt, and not through another process. (and same for the key/cert distribution, something for a different blog post maybe).

If so you're (pushing|pulling) automatically your web servers configuration from $cfgmgmt, but that you want to use/deploy TLS certificates signed by letsencrypt, what can you do ? Well, the good news is that you don't have to be forced to let the letsencrypt client touch your configuration at all : you can use the "certonly" option to just generate the private key locally, send the csr and get the signed cert back (and the whole chain too) One thing to know about letsencrypt is that the validation/verification process isn't the one that you can see in most of the companies providing CA/signing capabilities : as there is no ID/Paper verification (or something else) , the only validation for the domain/sub-domain that you want to generate a certificate for happens over http request (basically creating a file with a challenge , process a request from their "ACME" server[s] to retrieve that file back, and validate content)

So what are our options then ? The letsencrypt documentation mentions several plugins like manual (involves you to then create the file with the challenge answer to the webserver, then launching the validation process) , or standalone (doesn't work if you already have a httpd/nginx process as there will be a port conflict) , or even webroot (working fine as it will then just write the file itself under /.well-kwown/ under the DocumentRoot)

The webroot seems easy, but as said, we don't want to even install letsencrypt on the web server[s]. Even worse, suppose (and that's the case I had in mind) that you have multiple web nodes configured in a kind of CDN way : you don't want to distribute that file on all the nodes for validation/verification (when using the "manual" plugin) and you'd have to do it on *all* the nodes (as you don't know in advance which one will be verified by the ACME server)

So what about something centralized (where you'd run the letsencrypt client locally) for all your certs (including some with SANs) in a transpartent way ? I so thought about something like this :



The idea would be to :

- use a central node : let's call it central.domain.com (vm, docker

container, make-your-choice-here) to launch the letsencrypt client
- have the ACME server hitting transparently one of the web servers without any changed/uploaded file
- the server getting the GET request for that file using the letsencrypt central node as a backend node
- ACME server being happy and so signed certificates being available automatically on the centralize letsencrypt node.

The good news is that it's possible and even really easy to implement, through ProxyPass (for httpd/Apache web server) or proxy_pass (for nginx based setup)

For example, for the httpd vhost config for sub1.domain.com (three nodes in our example) we can just add this in the .conf file :

```
<Location "/.well-known/">
    ProxyPass "http://central.domain.com/.well-known/"
</Location>
```

So now, once in place everywhere, you can generate the cert for that domain on the central letsencrypt node (assuming that httpd is running on that node, and reachable from the "frontend" nodes, and that /var/www/html is indeed the DocumentRoot (default) for httpd on that node):

```
letsencrypt certonly --webroot --webroot-path /var/www/html --manual-p
```

Same if you run nginx instead (let's assume this for sub2.domain.com and sub3.domain.com) , you just have to add a snippet in your vhost .conf file (and before the / definition too):

```
location /.well-known/ {
        proxy_pass      http://central.domain.com/.well-known/ ;
    }
```

And then on the central node, do the same thing, but you can add multiple -d for multiple SubjectAltName in the same cert :

```
letsencrypt certonly --webroot --webroot-path /var/www/html --manual-p
```

Transparent, smart, easy to do and even something you can deploy when you need to renew, and then remove to be back with initial config files too (if you don't want to have those ProxyPass directives active all the time)

The only thing you have also to know is that once you have proper TLS in place, it's usually better to redirect transparently all requests to your http server to the https version. Most of the people will do that (next example for httpd/apache) like this :

```
   RewriteEngine On
   RewriteCond %{HTTPS} !=on
   RewriteRule ^/?(.*) https://%{SERVER_NAME}/$1 [R,L]
```

It's good, but when you'll renew the certificate, you'll probably just want to be sure that the GET request for /.well-known/* will continue to work over http (from the ACME server) so we can tune a little bit those rules (RewriteCond are cumulatives so it will not be redirect if url starts with .well-known:

```
   RewriteEngine On
   RewriteCond $1 !^.well-known
   RewriteCond %{HTTPS} !=on
   RewriteRule ^/?(.*) https://%{SERVER_NAME}/$1 [R,L]
```

Different syntax, but same principle for nginx : (also snippet, not full configuration file for that server/vhost):

```
location /.well-known/ {
        proxy_pass      http://central.domain.com/.well-known/ ;
    }
location / {
        rewrite        ^ https://$server_name$request_uri? permanent;
   }
```

Hope that you'll have found that useful, especially if you don't want to deploy letsencrypt everywhere but still use it to generate locally your keys/certs. Once done, you can then distribute/push/pull (depending on your cfgmgmt) those files and don't forget to also implement proper monitoring for cert validity and automation around that too (consider that your homework)