

SSL certificates

[[ssl](#), [certificates](#), [letsencrypt](#)]

[letsencrypt certification authority](#) is free, automated and open.

certbot

```
snap install --classic certbot
```

```
# or for focal pre
add-apt-repository ppa:certbot/certbot
apt-get update
apt-get install -y certbot python-certbot-nginx
```

```
certbot certonly --webroot -w /var/www/html -d mail.veronamobile.it
```

wildcard

```
certbot certonly \
  --manual \
  --preferred-challenges=dns \
  --email stefano.scipioni@csgalileo.org \
  --server https://acme-v02.api.letsencrypt.org/directory \
  --agree-tos -d *.iotaiuto.it
```

nginx

```
server {
    listen 80;
    server_name nextcloud.csgalileo.org;
    server_tokens off;

    location /.well-known/acme-challenge {
        root /var/www;
        allow all;
    }

    location / {
        return 301 https://$server_name$request_uri;
    }
}

server {
    listen 443;
```

```
server_name nnextcloud.csgalileo.org;
ssl_certificate
/etc/letsencrypt/live/nextcloud.csgalileo.org/fullchain.pem;
ssl_certificate_key
/etc/letsencrypt/live/nextcloud.csgalileo.org/privkey.pem;
}
```

renew

```
certbot renew [--dry-run]
```

automatic renew

```
systemctl status certbot.service
```

/etc/letsencrypt/cli.ini

```
max-log-backups = 0
deploy-hook = systemctl reload nginx
```

acme.sh integration for letsencrypt

On host that has apache/nginx install acme.sh

```
wget -O - https://get.acme.sh | sh
. ~/.bashrc
# now /root/.acme.sh/acme.sh.env is available with bash alias
```

certificate generation

with nginx enable this server on port 80 for initial challenge

[site.conf](#)

```
server {
    listen 80;
    server_name "mail.csgalileo.org";

    # create this folder empty
    location /.well-known/acme-challenge {
        root /var/www;
    }
    allow all;
}

location / {
    return 301 https://$server_name$request_uri;
}
```

}

```
# /var/www is documentroot of mail.csgalileo.org
acme.sh --issue -w /var/www -d mail.csgalileo.org --keylength ec-256
```

results in `/root/.acme.sh/mail.csgalileo.orgecc/** <file> [Fri Oct 14 08:05:13 CEST 2016] Creating account key [Fri Oct 14 08:05:15 CEST 2016] Registering account [Fri Oct 14 08:05:18 CEST 2016] Registered [Fri Oct 14 08:05:20 CEST 2016] Update success. [Fri Oct 14 08:05:20 CEST 2016] Creating domain key [Fri Oct 14 08:05:20 CEST 2016] Single domain='mail.csgalileo.org' [Fri Oct 14 08:05:20 CEST 2016] Verify each domain [Fri Oct 14 08:05:20 CEST 2016] Getting webroot for domain='mail.csgalileo.org' [Fri Oct 14 08:05:20 CEST 2016] _w='/var/www' [Fri Oct 14 08:05:20 CEST 2016] Getting new-authz for domain='mail.csgalileo.org' [Fri Oct 14 08:05:23 CEST 2016] Verifying:mail.csgalileo.org [Fri Oct 14 08:05:31 CEST 2016] Success [Fri Oct 14 08:05:31 CEST 2016] Verify finished, start to sign. [Fri Oct 14 08:05:34 CEST 2016] Cert success. ---BEGIN CERTIFICATE--- ... ---END CERTIFICATE--- [Fri Oct 14 08:05:34 CEST 2016] Your cert is in /root/.acme.sh/mail.csgalileo.orgecc/mail.csgalileo.org.cer [Fri Oct 14 08:05:34 CEST 2016] Your cert key is in /root/.acme.sh/mail.csgalileo.orgecc/mail.csgalileo.org.key [Fri Oct 14 08:05:34 CEST 2016] The intermediate CA cert is in /root/.acme.sh/mail.csgalileo.orgecc/ca.cer [Fri Oct 14 08:05:34 CEST 2016] And the full chain certs is there: /root/.acme.sh/mail.csgalileo.org_ecc/fullchain.cer </file> ===== certificate integration for apache ===== <code bash> HOST=mail.csgalileo.org acme.sh -installcert -d $HOST \ -certpath /etc/ssl/certs/${HOST}.cer \ -keypath /etc/ssl/private/${HOST}.key \ -capath /etc/ssl/certs/ca.cer \ -fullchainpath /etc/apache2/fullchain.cer \ -ecc \ -reloadcmd "service apache2 reload" </code> <file yml apache.conf> <VirtualHost *:80> ServerName projects.csgalileo.org DocumentRoot /var/www/html Alias /.well-known/acme-challenge/ /var/www/html/.well-known/acme-challenge/ <Directory "/var/www/html/.well-known/acme-challenge/"> Options None AllowOverride None ForceType text/plain RedirectMatch 404 "(?!/.well-known/acme-challenge/[w-]{43}$)" </Directory> RewriteEngine On RewriteCond %{REQUEST_URI} !^/.well-known. RewriteRule ^/?(.) https://%{SERVERNAME}:443/$1 [R,L] # Redirect permanent / https://projects.csgalileo.org/ </VirtualHost> <VirtualHost *:443> # ... SSLengine on SSLCertificateFile /etc/ssl/certs/mail.csgalileo.org.cer SSLCertificateKeyFile /etc/ssl/private/mail.csgalileo.org.key SSLCertificateChainFile /etc/apache2/fullchain.cer SSLCACertificateFile /etc/ssl/certs/ca.cer </VirtualHost> </file> ===== certificate integration for nginx ===== <file yml site.conf> server { listen 443 ssl; server_name "scipio.csgalileo.org"; # ... sslcertificate /etc/ssl/certs/scipio.csgalileo.org.cer; sslcertificate_key /etc/ssl/private/scipio.csgalileo.org.key; } </file> <code bash> HOST=mail.csgalileo.org acme.sh -installcert -d $HOST \ -keypath /etc/ssl/private/${HOST}.key \ -capath /etc/ssl/certs/ca.cer \ -fullchainpath /etc/ssl/certs/${HOST}.cer \ -ecc \ -reloadcmd "service nginx reload" </code> ===== renew automatic every 60 days ===== in cron there is already <code bash> 04 0 * * * "/root/.acme.sh"/acme.sh -cron -home "/root/.acme.sh" > /dev/null </code> ===== renew manual ===== <code bash> acme.sh -renew -d mail.csgalileo.org -force -ecc </code> ===== multi server ===== i can confirm this works. <code> location ~ /.well-known/acme-challenge/ { proxy_pass http://ctrl.mydomain.com:80; } </code> using nginx i added this location to ALL server blocks. You then run lets encrypt on the machine ctrl.mydomain.com (this machine typically is the controller machine, and is not serving web stuff - its pure purpose from a web POV is to handle incoming cert requests - if you don't know what a controller machine`

is then read up on ansible) To make it work I had to use the webroot plugin for Let's Encrypt. I could not get standalone mode to work. my A records look like .. `www01.mydomain.com`1 points to 1.2.3.4 `www02.mydomain.com` points to 2.3.4.5 `ctrl.mydomain.com` points to 3.4.5.6 `mydomain.com` points to 1,2,3,4 and 2,3,4,5 (multiple A records) `www.mydomain.com` is an alias (cname) for `mydomain.com` </code> NGINX runs on `www01` and `www02` on port 80 to load balance requests (e.g. `www01` load balances between `www01` and `www02`, `www02` ALSO load balances between `www01` and `www02`) the above lets encrypt location block is added to NGINX running on both `www01` and `www02` for all NGINX server blocks now run lets encrypt in webroot mode (you will need to standup a web server on your controller machine) and request a single certificate for `www01.mydomain.com` `www02.mydomain.com` `mydomain.com` `www.mydomain.com` when you run this command on your controller machine (`ctrl.mydomain.com`) it will fireoff a request to each of the 4 domains in return. Every single request will be proxied back to `ctrl.mydomain.com` via NGINX bosh! 2 tips 1 - to use webroot mode you will need to have a basic web server running on `ctrl.domain.com` which can serve content from a specified directory 2 - do not use standalone mode, i could not get it to work 3 - this solution sits very nicely if you are using ansible, since the certs will live on the controller machine and can be copied across to all slave machines with a single command

From:
<https://wiki.csgalileo.org/> - Galileo Labs

Permanent link:
<https://wiki.csgalileo.org/tips/ssl?rev=1623909943>

Last update: 2021/06/17 08:05

