

# Gerando certificados com o LetsEncrypt

**SO utilizado:** Ubuntu 16.04 LTS - 18.04 LTS - 20.04 LTS - Demais Linux suportados pelo acme.sh

**Fonte para HAproxy Ubuntu:** <https://www.digitalocean.com/community/tutorials/how-to-secure-haproxy-with-let-s-encrypt-on-ubuntu-14-04>

**Fonte para HAproxy CentOS:** <https://www.digitalocean.com/community/tutorials/how-to-secure-haproxy-with-let-s-encrypt-on-centos-7>

## 1. Introdução

Para a maioria das necessidades de certificação SSL, o IFSP recomenda a geração de certificados através da utilização do CertBot\Let's Encrypt.

Através do cliente do programa Let's Encrypt, o Certbot, podemos gerar certificados de forma dinâmica para os portais que utilizam SSL de forma segura e com renovação automatizada.

## 2. Instalação do software de aquisição e renovação de certificados

### 2.1. CertBot\Let's Encrypt

#### 2.1.2. UBUNTU

##### 2.1.2.1. NGINX:

```
#!/bin/bash
apt install letsencrypt && apt install python-certbot-nginx
mkdir /var/www/html/letsencrypt
chown www-data:www-data /var/www/html/letsencrypt
```

##### 2.1.2.2. APACHE2:

```
#!/bin/bash
apt install letsencrypt && apt install python-certbot-apache
mkdir /var/www/html/letsencrypt
chown www-data:www-data /var/www/html/letsencrypt
```

### 2.1.2.3. HAPROXY:

```
#!/bin/bash
apt update && apt install certbot
mkdir /var/www/html/letsencrypt
chown www-data:www-data /var/www/html/letsencrypt
```

## 2.1.3. CentOS

### 2.1.3.1. APACHE2

```
#!/bin/bash
yum install epel-release mod_ssl && yum install certbot python-certbot-apache mod_ssl
mkdir /var/www/html/letsencrypt
chown apache:apache /var/www/html/letsencrypt
```

### 2.1.3.2. NGINX

```
#!/bin/bash
yum install epel-release && yum install certbot-nginx
mkdir /var/www/html/letsencrypt
chown www-data:www-data /var/www/html/letsencrypt
```

### 2.1.3.3. HAPROXY

```
#!/bin/bash
yum install epel-release && yum install certbot
```

## 2.2. ACME.SH - Alternativa universal para Linux

Se você utiliza outro sistema operacional Linux ou somente não quer utilizar o certbot você pode utilizar o script **acme.sh**, ele tem uma lista extensa de compatibilidade de sistemas e no final irá gerar os mesmos resultados, apesar de ser menos integrado ao sistema.

Mais detalhes no [LINK](#)

```
mkdir /var/www/html/letsencrypt  
chown www-data:www-data /var/www/html/letsencrypt  
curl https://get.acme.sh | sh -s email=exemplo@ifsp.edu.br
```

O comando irá baixar e instalar o software automaticamente.

Se estiver usando SSH faça uma nova conexão para ativar o comando abaixo

Feito isso você pode ativar o update automático do script:

```
acme.sh --upgrade --auto-upgrade
```

## 3. Configuração do VHost

### 3.1. CertBot\Let's Encrypt

Devemos alterar o arquivo vhost do site alvo para que hospede o arquivo de verificação do domínio, este arquivo será utilizado para verificação da renovação no futuro pelo Let's Encrypt.

No exemplo utilizaremos o vhost teste.ifsp.edu.br

#### 3.1.1. VHost NGINX:

```
vi /etc/nginx/sites-available/teste.ifsp.edu.br
```

Insira as linhas abaixo:

```
location ~ /\.well-known/acme-challenge/ {  
    allow all;  
    root /var/www/html/letsencrypt;  
    try_files $uri =404;  
    break;  
}
```

O arquivo template do vhost já está atualizado com estas linhas

#### 3.1.2. VHost Apache2 (Ubuntu) ou (CentOS):

```
vi /etc/apache2/conf-available/letsencrypt.conf
```

```
vi /etc/httpd/conf.d/letsencrypt.conf
```

Insira as linhas abaixo:

```
Alias /.well-known/acme-challenge/ "/var/www/html/letsencrypt/.well-known/acme-challenge/"
<Directory "/var/www/html/letsencrypt/">
    AllowOverride None
    Options MultiViews Indexes SymLinkIfOwnerMatch IncludesNoExec
    Require method GET POST OPTIONS
</Directory>
```

Em seguida execute (Ubuntu):

```
a2enconf letsencrypt && systemctl reload apache2
```

No CentOS:

```
systemctl reload httpd
```

### 3.1.3. VHost HAproxy:

Bom, primeiro não existe isto de VHost HAproxy, ele não é um webserver, o que faremos posteriormente é utilizar um mini servidor embutido no próprio certbot para hospedar o arquivo de verificação, chamado de "plugin standalone".

O plugin standalone fornece um meio simples de obter o certificado SSL, ele funciona a partir de um pequeno web-server embutido que permite que o CA da Let's Encrypt se conecte e valide a identidade do servidor antes de emitir o certificado.

O plugin standalone funciona na porta 80, portanto o serviço do HAproxy deve estar desabilitado antes de emitir um certificado, se ele também utilizar a porta 80, a renovação do certificado não exige isto, como mostrado mais adiante.

## 3.2. ACME.SH

### 3.2.1. VHost NGINX:

Você pode seguir os mesmos procedimentos usados no certbot.

### 3.2.2. VHost Apache2:

Você pode seguir os mesmos procedimentos usados no certbot.

### 3.2.3. VHost HAproxy:

O princípio de funcionamento é o mesmo, porém será adaptado ao funcionamento do script acme, que também possui um modo standalone com um pequeno webserver integrado.

## 4. Geração do certificado

Antes de gerar o certificado se certifique de que já criou as entradas relativas ao domínio no DNS.

### 4.1. CertBot\Let's Encrypt

#### 4.1.1. NGINX:

```
certbot --nginx -d teste.ifsp.edu.br
```

#### 4.1.2. APACHE2:

```
certbot --apache -d teste.ifsp.edu.br
```

Este comando irá gerar o certificado e irá gravar o arquivo challenge na pasta "**/var/www/html/letsencrypt/.well-known/acme-challenge**"

Na primeira execução, o servidor irá fazer algumas perguntas, as respostas estão listadas a seguir

```
root@DIR-SLNX-PRX-nginx-002:~# certbot --nginx -d teste.ifsp.edu.br
```

Saving debug log to /var/log/letsencrypt/letsencrypt.log

Plugins selected: Authenticator nginx, Installer nginx

Enter email address (used for urgent renewal and security notices) (Enter 'c' to cancel): ctidocampus@ifsp.edu.br

-----  
Please read the Terms of Service at

<https://letsencrypt.org/documents/LE-SA-v1.2-November-15-2017.pdf>. You must agree in order to register with the ACME server at

<https://acme-v02.api.letsencrypt.org/directory>

-----  
(A)gree/(C)ancel: A

-----  
Would you be willing to share your email address with the Electronic Frontier Foundation, a founding partner of the Let's Encrypt project and the non-profit organization that develops Certbot? We'd like to send you email about our work encrypting the web, EFF news, campaigns, and ways to support digital freedom.

-----  
(Y)es/(N)o: N

Obtaining a new certificate

Performing the following challenges:

http-01 challenge for teste.ifsp.edu.br

Waiting for verification...

Cleaning up challenges

Deploying Certificate to VirtualHost /etc/nginx/sites-enabled/teste.ifsp.edu.br

Please choose whether or not to redirect HTTP traffic to HTTPS, removing HTTP access.

-----  
1: No redirect - Make no further changes to the webserver configuration.

2: Redirect - Make all requests redirect to secure HTTPS access. Choose this for new sites, or if you're confident your site works on HTTPS. You can undo this change by editing your web server's configuration.

-----  
Select the appropriate number [1-2] then [enter] (press 'c' to cancel): 1

-----  
Congratulations! You have successfully enabled <https://teste.ifsp.edu.br>

You should test your configuration at:

<https://www.ssllabs.com/ssltest/analyze.html?d=teste.ifsp.edu.br>

-----  
**IMPORTANT NOTES:**

- Congratulations! Your certificate and chain have been saved at:

</etc/letsencrypt/live/teste.ifsp.edu.br/fullchain.pem>

Your key file has been saved at:

</etc/letsencrypt/live/teste.ifsp.edu.br/privkey.pem>

Your cert will expire on 2020-01-28. To obtain a new or tweaked

version of this certificate in the future, simply run certbot again with the "certonly" option. To non-interactively renew \*all\* of your certificates, run "certbot renew"

- Your account credentials have been saved in your Certbot configuration directory at /etc/letsencrypt. You should make a secure backup of this folder now. This configuration directory will also contain certificates and private keys obtained by Certbot so making regular backups of this folder is ideal.
- If you like Certbot, please consider supporting our work by:

Donating to ISRG / Let's Encrypt: <https://letsencrypt.org/donate>

Donating to EFF: <https://eff.org/donate-le>

Feito! O Certbot automaticamente vai acessar o arquivo vhost do domínio e fará as alterações necessárias para incluir os novos certificados nele.

Não se esqueça de verificar os arquivos vhost em busca de algo que possa ter permanecido como lixo.

### 4.1.3. HAPROXY

Para que o módulo standalone funcione no HAProxy durante a geração do certificado, o HAProxy deve ser desligado momentaneamente:

```
systemctl stop haproxy
```

Em seguida já podemos gerar o certificado

```
certbot certonly --standalone --preferred-challenges http --http-01-port 80 -d teste.ifsp.edu.br
```

E religar o serviço:

```
systemctl start haproxy
```

Em seguida configure o certificado gerado no HAProxy e aproveite para configurar o redirecionamento dos pedidos de renovação para o pequeno webserver do certbot (módulo standalone), para isto, registramos abaixo uma configuração de exemplo completa (e editada de forma genérica) utilizada em um servidor HAProxy do IFSP, mostrada abaixo.

Esta alteração irá redirecionar o módulo standalone para a porta 54321, possibilitando assim a renovação com o serviço do HAProxy ligado.

global

```
log /dev/log    local0
log /dev/log    local1 notice
chroot /var/lib/haproxy
stats socket /run/haproxy/admin.sock mode 660 level admin expose-fd listeners
stats timeout 30s
user haproxy
group haproxy
daemon
```

# Default SSL material locations

```
ca-base /etc/ssl/certs
crt-base /etc/ssl/private
```

```
ssl-default-bind-ciphers ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-
ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-
RSA-CHACHA20-POLY1305:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:DHE-RSA-CHACHA20-
POLY1305:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-SHA256:ECDHE-ECDSA-AES128-SHA:ECDHE-RSA-
AES128-SHA:ECDHE-ECDSA-AES256-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-ECDSA-AES256-SHA:ECDHE-
RSA-AES256-SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES256-SHA256:AES128-GCM-SHA256:AES256-GCM-
SHA384:AES128-SHA256:AES256-SHA256:AES128-SHA:AES256-SHA:DES-CBC3-SHA
```

```
ssl-default-bind-options no-sslv3 no-tls-tickets
```

```
ssl-default-server-ciphers ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-
ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-
RSA-CHACHA20-POLY1305:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:DHE-RSA-CHACHA20-
POLY1305:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-SHA256:ECDHE-ECDSA-AES128-SHA:ECDHE-RSA-
AES128-SHA:ECDHE-ECDSA-AES256-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-ECDSA-AES256-SHA:ECDHE-
RSA-AES256-SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES256-SHA256:AES128-GCM-SHA256:AES256-GCM-
SHA384:AES128-SHA256:AES256-SHA256:AES128-SHA:AES256-SHA:DES-CBC3-SHA
```

```
ssl-default-server-options no-sslv3 no-tls-tickets
```

```
# curl https://ssl-config.mozilla.org/ffdh2048.txt > /etc/ssl/dhparam/dhparam
```

```
ssl-dh-param-file /etc/ssl/dhparam/dhparam
```

defaults

```
log    global
mode   http
option httplog
option dontlognull
```



```
timeout connect 5s
timeout client 720s
timeout server 720s
errorfile 400 /etc/haproxy/errors/400.http
errorfile 403 /etc/haproxy/errors/403.http
errorfile 408 /etc/haproxy/errors/408.http
errorfile 500 /etc/haproxy/errors/500.http
errorfile 502 /etc/haproxy/errors/502.http
errorfile 503 /etc/haproxy/errors/503.http
errorfile 504 /etc/haproxy/errors/504.http
```

```
frontend frontend-dsi-docker-cluster
```

```
bind :80
bind :443 ssl crt /etc/haproxy/certs/teste.ifsp.edu.br.pem alpn h2,http/1.1
http-request redirect scheme https unless { ssl_fc }
# ACL para permitir o redirecionamento da renovacao para o lets encrypt
acl letsencrypt-acl path_beg /.well-known/acme-challenge/
```

```
http-response set-header Strict-Transport-Security max-age=63072000
```

```
# Redireciona para o lets encrypt quando a ACL respectiva for verdadeira
use_backend letsencrypt-backend if letsencrypt-acl
default_backend teste-haproxy-ifsp-cluster
```

```
backend teste-haproxy-ifsp-cluster
```

```
balance roundrobin
option forwardfor
server servidor1.ifsp.edu.br 10.10.10.1:80 check
server servidor2.ifsp.edu.br 10.10.10.2:80 check
server servidor3.ifsp.edu.br 10.10.10.3:80 check
http-request set-header X-Forwarded-Port %[dst_port]
http-request add-header X-Forwarded-Proto https if { ssl_fc }
```

```
# Backend criado para possibilitar a utilizacao do lets encrypt que por padrao tambem usa a porta 80
```

```
backend letsencrypt-backend
```

```
server letsencrypt 127.0.0.1:54321
```

```
listen stats
```

```
bind 10.10.10.4:1936
```

```
stats enable
stats hide-version
stats refresh 30s
stats show-node
stats auth username:password
stats uri /stats
```

Os detalhes desta configuração referentes ao "**letsencrypt-backend**" são importantes para a renovação automática dos certificados, tratadas em tópico posterior.

## 4.2. ACME.SH

Para gerar os certificados usando o script **acme.sh**, use os seguintes comandos:

```
acme.sh --issue -d teste.ifsp.edu.br -w /var/www/teste
```

Agora para instalar os certificados nos WebServers:

Não utilize os certificados armazenados em "~/.acme.sh/" eles são para uso interno somente.

### 4.2.1. NGINX

#### 4.2.1.1. Automaticamente via interação com o serviço

```
acme.sh --issue --nginx -d exemplo.ifsp.edu.br -d www.exemplo.ifsp.edu.br
```

#### 4.2.1.2. Manualmente

Primeiro criamos a pasta que irá abrigar nosso certificado:

```
mkdir -p /etc/nginx/acme.sh/teste.ifsp.edu.br
```

Em seguida instalamos o certificado:

```
acme.sh --install-cert -d teste.ifsp.edu.br \
--cert-file /etc/nginx/acme.sh/teste.ifsp.edu.br/cert.pem \
--key-file /etc/nginx/acme.sh/teste.ifsp.edu.br/key.pem \
--fullchain-file /etc/nginx/acme.sh/teste.ifsp.edu.br/fullchain.pem \
--reloadcmd "service apache2 force-reload"
```

Feito isso já pode configurar os certificados contidos na pasta criada, adicione as seguintes linhas na configuração do seu VHost:

```
ssl_certificate /etc/nginx/acme.sh/teste.ifsp.edu.br/fullchain.pem;  
ssl_certificate_key /etc/nginx/acme.sh/teste.ifsp.edu.br/key.pem;  
ssl_trusted_certificate /etc/nginx/acme.sh/teste.ifsp.edu.br/cert.pem;
```

## 4.2.2. APACHE2

### 4.2.2.1. Automaticamente via interação com o serviço

```
acme.sh --issue --apache -d exemplo.ifsp.edu.br -d www.exemplo.ifsp.edu.br
```

### 4.2.2.2. Manualmente

Primeiro criamos a pasta que irá abrigar nosso certificado:

```
mkdir -p /etc/apache2/acme.sh/teste.ifsp.edu.br
```

Em seguida instalamos o certificado:

```
acme.sh --install-cert -d teste.ifsp.edu.br \  
--cert-file /etc/apache2/acme.sh/teste.ifsp.edu.br/cert.pem \  
--key-file /etc/apache2/acme.sh/teste.ifsp.edu.br/key.pem \  
--fullchain-file /etc/apache2/acme.sh/teste.ifsp.edu.br/fullchain.pem \  
--reloadcmd "service apache2 force-reload"
```

Feito isso já pode configurar os certificados contidos na pasta criada, adicione as seguintes linhas na configuração do seu VHost:

```
SSLCertificateFile "/etc/apache2/acme.sh/teste.ifsp.edu.br/cert.pem"  
SSLCertificateKeyFile "/etc/apache2/acme.sh/teste.ifsp.edu.br/key.pem"  
SSLCertificateChainFile "/etc/apache2/acme.sh/teste.ifsp.edu.br/fullchain.pem"
```

## 4.2.3. HAPROXY

Os procedimentos de configuração do HAProxy são os mesmos adotados para o certbot, conforme descrito acima, somente o comando para geração do certificado será alterado:

```
acme.sh --issue -d teste.ifsp.edu.br --standalone -d /pasta/de/destino --httpport 54321
```

## 4.2.4. OUTROS

Imagine um cenário hipotético onde vc tem um servidor com um software diferenciado instalado, que hospeda um container Docker Nginx que esta ocupando a porta 80 e que precisa de um certificado SSL instalado na pasta `"/etc/nginx/conf.d/"` ; você não pode instalar um novo servidor web local, não pode alterar o Dockerfile, nem tampouco executar um novo container Docker do Certbot ou ACME para gerar os certificados.

Bom, neste caso você pode contornar o problema com o comando abaixo na hora de gerar o certificado:

Crie o seguinte script:

```
vi /usr/local/bin/renew.sh
```

Cole o seguinte conteúdo no arquivo, editando as partes necessárias:

```
#!/bin/sh

SITE=teste.ifsp.edu.br

# move to the correct let's encrypt directory
cd /etc/letsencrypt/live/$SITE

# Ajustes para Docker
docker cp /etc/letsencrypt/live/teste.ifsp.edu.br.crt nome-do-container:/etc/nginx/conf.d/teste.ifsp.edu.br.crt
docker cp /etc/letsencrypt/live/teste.ifsp.edu.br.key nome-do-container:/etc/nginx/conf.d/teste.ifsp.edu.br.key
docker start nome-do-container
```

Agora atribua permissão de execução ao script:

```
chmod u+x /usr/local/bin/renew.sh
```

```
certbot certonly --standalone --pre-hook "docker stop container-na-porta-80" --post-hook  
"/usr/local/bin/renew.sh" --preferred-challenges http --http-01-port 80 -d teste.ifsp.edu.br
```

Este comando vai:

**Parar o container atualmente em execução antes de tentar gerar o certificado: <--pre-hook "docker stop container-na-porta-80">**

**Executar o script acima após a geração do certificado, o script vai copiar os certificados para o container e em seguida reinicia-lo: <--post-hook "/usr/local/bin/renew.sh">**

Estes comandos também serão executados durante o processo de renovação automáticos certificados, tema que será coberto mais adiante.

## 5. Deleção de certificados

### 5.1. CertBot\Let's Encrypt

Para deletar o certificado no certbot, execute o comando abaixo:

```
certbot delete --cert-name teste.ifsp.edu.br
```

### 5.2. ACME.SH

```
acme.sh --remove -d teste.ifsp.edu.br
```

## 6. Renovação automática do certificado

### 6.1. CertBot\Let's Encrypt

O certificado gerado pelo Let's Encrypt tem validade de 3 meses, para renova-lo automaticamente, execute os comandos abaixo:

#### 6.1.1. Teste para ver se a renovação dos certificados ocorrerá sem erros:

```
certbot renew --dry-run
```

Se nenhuma mensagem de erro aparecer pode prosseguir para o próximo passo.

#### 6.1.2. Script para a criação da regra no crond:

##### 6.1.2.1. NGINX:

```
#!/bin/bash
touch /etc/cron.daily/letsencrypt
chmod +x /etc/cron.daily/letsencrypt
cat > /etc/cron.daily/letsencrypt <<DELIM
#!/bin/bash
/usr/bin/letsencrypt renew --renew-hook "/etc/init.d/nginx reload"
```

```
DELIM
```

### 6.1.2.2. APACHE2:

```
#!/bin/bash
touch /etc/cron.daily/letsencrypt
chmod +x /etc/cron.daily/letsencrypt
cat > /etc/cron.daily/letsencrypt <<DELIM
#!/bin/bash
/usr/bin/letsencrypt renew --renew-hook "/etc/init.d/apache2 reload"
DELIM
```

### 6.1.2.3. HAPROXY:

Primeiro, devemos criar um script que fará a mesclagem dos certificados emitidos pelo LE, o HAproxy só aceita certificados em Bundles:

```
vi /usr/local/bin/renew.sh
```

```
#!/bin/sh

SITE=teste.ifsp.edu.br

# move to the correct let's encrypt directory
cd /etc/letsencrypt/live/$SITE

# cat files to make combined .pem for haproxy
cat fullchain.pem privkey.pem > /etc/haproxy/certs/$SITE.pem

# reload haproxy
service haproxy reload
```

```
chmod u+x /usr/local/bin/renew.sh
```

Agora devemos alterar a porta do mini servidor web do certbot para utilizar outra porta que não seja a 80, evitando o conflito com o HAproxy:

```
vi /etc/letsencrypt/renewal/teste.ifsp.edu.br.conf
```

```
# Altere este linha
http01_port = 54321
```

Agora vamos agendar a renovação com o cron:

```
crontab -e
```

```
30 2 * * * /usr/bin/certbot renew --post-hook "/usr/local/bin/renew.sh" >> /var/log/le-renewal.log
```

## 6.2. ACME.SH

Os certificados emitidos pelo acme.sh são renovados automaticamente a cada 60 dias e não necessitam de intervenção ou configuração extra.

## 7.0 Dicas

### 7.1 Verificar conteúdo dos certificados emitidos

Para verificar o conteúdo do certificado e suas datas de expiração e criação, basta utilizar o comando:

```
openssl x509 -in /etc/letsencrypt/live/teste.ifsp.edu.br/cert.pem -text -noout
```

---

Revision #48

Created 6 July 2020 18:08:52 by Rafael Manochio

Updated 21 August 2024 14:52:14 by Rafael Manochio