**nginx使用ssl模块配置HTTPS支持**

**CentOS7下：**

**生成证书**

可以通过以下步骤生成一个简单的证书：

首先，进入你想创建证书和私钥的目录，例如：

$ cd /usr/local/nginx/conf

创建服务器私钥，命令会让你输入一个口令：

$ openssl genrsa -des3 -out server.key 1024

创建签名请求的证书（CSR）：

$ openssl req -new -key server.key -out server.csr

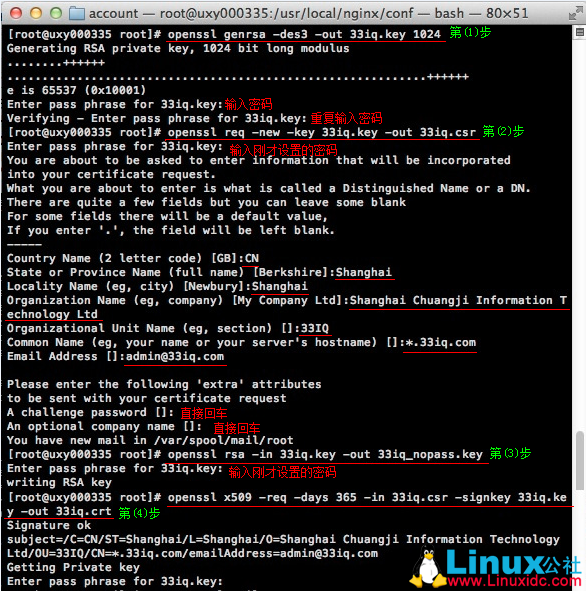
在加载SSL支持的Nginx并使用上述私钥时除去必须的口令：

$ cp server.key server.key.org

$ openssl rsa -in server.key.org -out server.key

最后标记证书使用上述私钥和CSR：

$ openssl x509 -req -days 365 -in server.csr -signkey server.key -out server.crt



**配置nginx**

修改Nginx配置文件nginx.conf，让其包含新标记的证书和私钥：

server {

server\_name YOUR\_DOMAINNAME\_HERE;

listen 443;

ssl on;

ssl\_certificate /usr/local/nginx/conf/server.crt;

ssl\_certificate\_key /usr/local/nginx/conf/server.key;

}

重启nginx。

这样就可以通过以下方式访问：

<https://YOUR_DOMAINNAME_HERE>

已经配置好的nginx.conf，提供参考：



**Windows下：**

将证书放到nginx.conf的相同目录下进行配置

**nginx基本配置与参数说明**

user nobody;

#启动进程,通常设置成和cpu的数量相等

worker\_processes 1;

#全局错误日志及PID文件

#error\_log logs/error.log;

#error\_log logs/error.log notice;

#error\_log logs/error.log info;

#pid logs/nginx.pid;

#工作模式及连接数上限

events {

#epoll是多路复用IO(I/O Multiplexing)中的一种方式,

#仅用于linux2.6以上内核,可以大大提高nginx的性能

use epoll;

#单个后台worker process进程的最大并发链接数

worker\_connections 1024;

# 并发总数是 worker\_processes 和 worker\_connections 的乘积

# 即 max\_clients = worker\_processes \* worker\_connections

# 在设置了反向代理的情况下，max\_clients = worker\_processes \* worker\_connections / 4 为什么

# 为什么上面反向代理要除以4，应该说是一个经验值

# 根据以上条件，正常情况下的Nginx Server可以应付的最大连接数为：4 \* 8000 = 32000

# worker\_connections 值的设置跟物理内存大小有关

# 因为并发受IO约束，max\_clients的值须小于系统可以打开的最大文件数

# 而系统可以打开的最大文件数和内存大小成正比，一般1GB内存的机器上可以打开的文件数大约是10万左右

# 我们来看看360M内存的VPS可以打开的文件句柄数是多少：

# $ cat /proc/sys/fs/file-max

# 输出 34336

# 32000 < 34336，即并发连接总数小于系统可以打开的文件句柄总数，这样就在操作系统可以承受的范围之内

# 所以，worker\_connections 的值需根据 worker\_processes 进程数目和系统可以打开的最大文件总数进行适当地进行设置

# 使得并发总数小于操作系统可以打开的最大文件数目

# 其实质也就是根据主机的物理CPU和内存进行配置

# 当然，理论上的并发总数可能会和实际有所偏差，因为主机还有其他的工作进程需要消耗系统资源。

# ulimit -SHn 65535

}

http {

#设定mime类型,类型由mime.type文件定义

include mime.types;

default\_type application/octet-stream;

#设定日志格式

log\_format main '$remote\_addr - $remote\_user [$time\_local] "$request" '

'$status $body\_bytes\_sent "$http\_referer" '

'"$http\_user\_agent" "$http\_x\_forwarded\_for"';

access\_log logs/access.log main;

#sendfile 指令指定 nginx 是否调用 sendfile 函数（zero copy 方式）来输出文件，

#对于普通应用，必须设为 on,

#如果用来进行下载等应用磁盘IO重负载应用，可设置为 off，

#以平衡磁盘与网络I/O处理速度，降低系统的uptime.

sendfile on;

#tcp\_nopush on;

#连接超时时间

#keepalive\_timeout 0;

keepalive\_timeout 65;

tcp\_nodelay on;

#开启gzip压缩

gzip on;

gzip\_disable "MSIE [1-6].";

#设定请求缓冲

client\_header\_buffer\_size 128k;

large\_client\_header\_buffers 4 128k;

#设定虚拟主机配置

server {

#侦听80端口

listen 80;

#定义使用 www.nginx.cn访问

server\_name www.nginx.cn;

#定义服务器的默认网站根目录位置

root html;

#设定本虚拟主机的访问日志

access\_log logs/nginx.access.log main;

#默认请求

location / {

#定义首页索引文件的名称

index index.php index.html index.htm;

}

# 定义错误提示页面

error\_page 500 502 503 504 /50x.html;

location = /50x.html {

}

#静态文件，nginx自己处理

location ~ ^/(images|javascript|js|css|flash|media|static)/ {

#过期30天，静态文件不怎么更新，过期可以设大一点，

#如果频繁更新，则可以设置得小一点。

expires 30d;

}

#PHP 脚本请求全部转发到 FastCGI处理. 使用FastCGI默认配置.

location ~ .php$ {

fastcgi\_pass 127.0.0.1:9000;

fastcgi\_index index.php;

fastcgi\_param SCRIPT\_FILENAME $document\_root$fastcgi\_script\_name;

include fastcgi\_params;

}

#禁止访问 .htxxx 文件

location ~ /.ht {

deny all;

}

}

}