26-Docker安装nginx以及部署

> 整理：参码踪田不平
>
> 时间：2020-03-18
>
> 联系：3350996729（QQ）
>
> 官网：http://www.shenmazong.com

1、nginx简介

1）、来源

Nginx (engine x) 是一个高性能的HTTP和反向代理web服务器，同时也提供了IMAP/POP3/SMTP服务。Nginx是由伊戈尔·赛索耶夫为俄罗斯访问量第二的Rambler.ru站点（俄文：Рамблер）开发的，第一个公开版本0.1.0发布于2004年10月4日。
其将源代码以类BSD许可证的形式发布，因它的稳定性、丰富的功能集、示例配置文件和低系统资源的消耗而闻名。2011年6月1日，nginx 1.0.4发布。
Nginx是一款轻量级的Web 服务器/反向代理服务器及电子邮件（IMAP/POP3）代理服务器，在BSD-like 协议下发行。其特点是占有内存少，并发能力强，事实上nginx的并发能力在同类型的网页服务器中表现较好，中国大陆使用nginx网站用户有：百度、京东、新浪、网易、腾讯、淘宝等。

2）、源代码

Nginx代码完全用C语言从头写成，已经移植到许多体系结构和操作系统，包括：Linux、FreeBSD、Solaris、Mac OS X、AIX以及Microsoft Windows。Nginx有自己的函数库，并且除了zlib、PCRE和OpenSSL之外，标准模块只使用系统C库函数。而且，如果不需要或者考虑到潜在的授权冲突，可以不使用这些第三方库。

3）、服务器

Tomcat

IIS(Window)

Apache

Nginx

Nginx作为负载均衡服务：Nginx 既可以在内部直接支持 Rails 和 PHP 程序对外进行服务，也可以支持作为 HTTP代理服务对外进行服务。Nginx采用C进行编写，不论是系统资源开销还是CPU使用效率都比 Perlbal 要好很多。
处理静态文件，索引文件以及自动索引;打开文件描述符缓冲。
无缓存的反向代理加速，简单的负载均衡和容错。
FastCGI，简单的负载均衡和容错。
模块化的结构。包括 gzipping, byte ranges, chunked responses,以及 SSI-filter 等 filter。如果由 FastCG或其它代理服务器处理单页中存在的多个 SSI，则这项处理可以并行运行，而不需要相互等待。
支持 SSL 和 TLSSNI。

4）、优点

Nginx 可以在大多数 Unix/Linux OS 上编译运行，并有 Windows 移植版。 Nginx 的1.4.0稳定版已经于2013年4月24日发布，一般情况下，对于新建站点，建议使用最新稳定版作为生产版本，已有站点的升级急迫性不高。
Nginx 是一个很强大的高性能Web和反向代理服务，它具有很多非常优越的特性：
在连接高并发的情况下，Nginx是Apache服务不错的替代品：Nginx在美国是做虚拟主机生意的老板们经常选择的软件平台之一。能够支持高达 50,000 个并发连接数的响应，感谢Nginx为我们选择了 epoll and kqueue作为开发模型。

5）、代理

作为邮件代理服务：Nginx 同时也是一个非常优秀的邮件代理服务（最早开发这个产品的目的之一也是作为邮件代理服务器），Last.fm 描述了成功并且美妙的使用经验。
Nginx 是一个安装非常的简单、配置文件非常简洁（还能够支持perl语法）、Bug非常少的服务。Nginx 启动特别容易，并且几乎可以做到7*24不间断运行，即使运行数个月也不需要重新启动。你还能够不间断服务的情况下进行软件版本的升级。

2、nginx的安装

1）、下载镜像

```
docker pull nginx:1.17.8
```

```
[root@iZlwo3fsqciqqbZ ~]# docker pull nginx:1.17.8
1.17.8: Pulling from library/nginx
68ced04f60ab: Pull complete
c4039fd85dcc: Pull complete
c16ce02d3d61: Pull complete
Digest: sha256:380eb808e2a3b0dd954f92c1cae2f845e6558a15037efefcabc5b4e03d666d03
Status: Downloaded newer image for nginx:1.17.8
docker.io/library/nginx:1.17.8
[root@iZlwo3fsqciqqbZ ~]#
```

2）、查看镜像

```
docker images
```

```
[root@iZlwo3fsqciqqbZ ~]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
nginx               1.17.8              a1523e859360        41 hours ago        127MB
[root@iZlwo3fsqciqqbZ ~]#
```

3）、测试镜像

```
sudo docker run --name nginx -p 9091:80 -d nginx:1.17.8
docker ps
http://39.105.82.148:9091
```

```
[root@iZlwo3fsqciqqbZ nginx]# sudo docker run --name nginx -p 9091:80 -d nginx:1.17.8
47b68e53c58edafdb55bb7c3ed4ed3b67fc8b7a0c5e757450740cff2421f4a54
[root@iZlwo3fsqciqqbZ nginx]# docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                    NAMES
47b68e53c58e        nginx:1.17.8        "nginx -g 'daemon of…"   11 seconds ago      Up 10 seconds       0.0.0.0:9091->80/tcp     nginx
[root@iZlwo3fsqciqqbZ nginx]#
```

4）、容器部署

```
# 创建www目录
mkdir -p /server/nginx/html
# 创建日志目录
mkdir -p /server/nginx/logs
# 创建配置目录
mkdir -p /server/nginx/conf
```

拷贝容器内 Nginx 默认配置文件到本地当前目录下的 conf 目录，容器 ID 可以查看 docker ps 命令输入中的第一列：

```
## 拷贝配置文件
docker cp f77f78d2228d:/etc/nginx/nginx.conf ~/nginx/conf
## 映射容器目录
```

```
docker run -d -p 9091:80 --name nginx -v
/server/nginx/html:/usr/share/nginx/html -v
/server/nginx/conf/nginx.conf:/etc/nginx/nginx.conf -v
/server/nginx/logs:/var/log/nginx --privileged=true nginx:1.17.8
##
docker run -d -p 9091:80 --name nginx -v
/server/nginx/html:/usr/share/nginx/html nginx:1.17.8

##
sudo docker run -d -p 9091:443 --name nginx \
-v /home/centos/server/nginx/html:/usr/share/nginx/html \
-v /home/centos/server/nginx/conf/nginx.conf:/etc/nginx/nginx.conf \
-v /home/centos/server/nginx/logs:/var/log/nginx \
nginx:1.17.8

## 查看运行容器
docker ps
## 如果因为权限问题，需要加上--privileged=true这个参数
docker run -d -p 9091:80 --name nginx -v
/server/nginx/html:/usr/share/nginx/html --privileged=true nginx:1.17.8

[root@localhost nginx]# docker run -d -p 9091:80 --name nginx -v
/server/nginx/html:/usr/share/nginx/html -v
/server/nginx/conf/nginx.conf:/etc/nginx/nginx.conf -v
/server/nginx/logs:/var/log/nginx nginx:1.17.8
949f5d7cdebff5de5edc4d204f73d88da9e663eae75059295ea80c39138df0b0
docker: Error response from daemon: OCI runtime create failed:
container_linux.go:346: starting container process caused
"process_linux.go:449: container init caused \"rootfs_linux.go:58: mounting
\\\"/server/nginx/conf/nginx.conf\\\" to rootfs
\\\"/var/lib/docker/overlay2/7c94763f88d871bb4d80d41fecbac8d6595b17cfc6e636c81
744ddcb21a42270/merged\\\" at
\\\"/var/lib/docker/overlay2/7c94763f88d871bb4d80d41fecbac8d6595b17cfc6e636c81
744ddcb21a42270/merged/etc/nginx/nginx.conf\\\" caused \\\"not a
directory\\\"\"": unknown: Are you trying to mount a directory onto a file (or
vice-versa)? Check if the specified host path exists and is the expected type.
```

5）、检查配置

为了验证配置文件是否正确，可以通过如下命令：

```
# 进入容器
docker exec -it nginx /bin/bash
# 检查配置文件
nginx -t
```

```
[root@iZlwo3fsqciqqbZ ~]# docker exec -it nginx /bin/bash
root@804b395f0864:/# nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
root@804b395f0864:/#
```

6）、启动服务

```
# 重新载入配置文件
nginx -s reload
# 重启 Nginx
nginx -s reopen
# 停止 Nginx
nginx -s stop
```

3、web服务器

1）默认配置

```
user  nginx;
worker_processes  1;

error_log  /var/log/nginx/error.log warn;
pid        /var/run/nginx.pid;


events {
    worker_connections  1024;
}


http {
    include       /etc/nginx/mime.types;
    default_type  application/octet-stream;

    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  /var/log/nginx/access.log  main;

    sendfile        on;
    #tcp_nopush     on;

    keepalive_timeout  65;

    #gzip  on;
```

```
    include /etc/nginx/conf.d/*.conf;
}
```

2）、server_name

（1）准确匹配

```
server {
  listen 80;
  server_name www.shenmazong.com;
  #...
}
```

（2）通配符开始

```
server {
  listen 80;
  server_name *.shenmazong.com;
  #...
}
```

（3）通配符结束

```
server {
  listen 80;
  server_name www.shenmazong.*;
  #...
}
```

（4）正则表达式

```
server {
  listen 80;
  server_name ~^(?.+)\.shenmazong\.org$;
  #...
}
```

3）、普通配置

（1）创建html 文件

（2）修改hosts文件

（3）修改nginx.conf文件

（4）重启docker

```
# For more information on configuration, see:
#   * Official English Documentation: http://nginx.org/en/docs/
#   * Official Russian Documentation: http://nginx.org/ru/docs/

user nginx;
worker_processes auto;
error_log /var/log/nginx/error.log;
pid /run/nginx.pid;

# Load dynamic modules. See /usr/share/nginx/README.dynamic.
# include /usr/share/nginx/modules/*.conf;

events {
    worker_connections 1024;
}

http {
    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  /var/log/nginx/access.log  main;

    sendfile            on;
    tcp_nopush          on;
    tcp_nodelay         on;
    keepalive_timeout   65;
    types_hash_max_size 2048;

    include             /etc/nginx/mime.types;
    default_type        application/octet-stream;

    # Load modular configuration files from the /etc/nginx/conf.d directory.
    # See http://nginx.org/en/docs/ngx_core_module.html#include
    # for more information.
    # include /etc/nginx/conf.d/*.conf;

  # www.shenmazong.com
    server {
        listen       80;
        listen       [::]:80;
        server_name  www.shenmazong.com;
        root         /usr/share/nginx/html/www.shenmazong.com;

        # Load configuration files for the default server block.
        # include /etc/nginx/default.d/*.conf;

        location / {
          root    /usr/share/nginx/html/www.shenmazong.com;
```

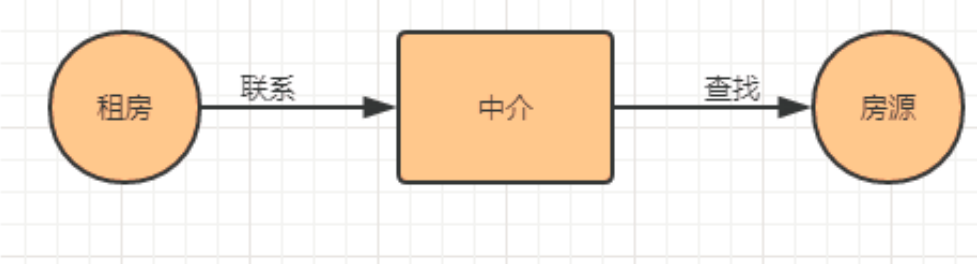
```
          index   index.html index.htm;
        }

        error_page 404 /404.html;
            location = /40x.html {
        }

        error_page 500 502 503 504 /50x.html;
            location = /50x.html {
        }
    }

  # blog.shenmazong.com
    server {
        listen       80;
        listen       [::]:80;
        server_name  blog.shenmazong.com;
        root         /usr/share/nginx/html/blog.shenmazong.com;

        # Load configuration files for the default server block.
        # include /etc/nginx/default.d/*.conf;

        location / {
          root    /usr/share/nginx/html/blog.shenmazong.com;
          index   index.html index.htm;
        }

        error_page 404 /404.html;
            location = /40x.html {
        }

        error_page 500 502 503 504 /50x.html;
            location = /50x.html {
        }
    }

# Settings for a TLS enabled server.
#
#    server {
#        listen       443 ssl http2 default_server;
#        listen       [::]:443 ssl http2 default_server;
#        server_name  _;
#        root         /usr/share/nginx/html;
#
#        ssl_certificate "/etc/pki/nginx/server.crt";
#        ssl_certificate_key "/etc/pki/nginx/private/server.key";
#        ssl_session_cache shared:SSL:1m;
#        ssl_session_timeout  10m;
#        ssl_ciphers HIGH:!aNULL:!MD5;
```

```
#        ssl_prefer_server_ciphers on;
#
#        # Load configuration files for the default server block.
#        include /etc/nginx/default.d/*.conf;
#
#        location / {
#        }
#
#        error_page 404 /404.html;
#            location = /40x.html {
#        }
#
#        error_page 500 502 503 504 /50x.html;
#            location = /50x.html {
#        }
#    }
}
```

4、反向代理

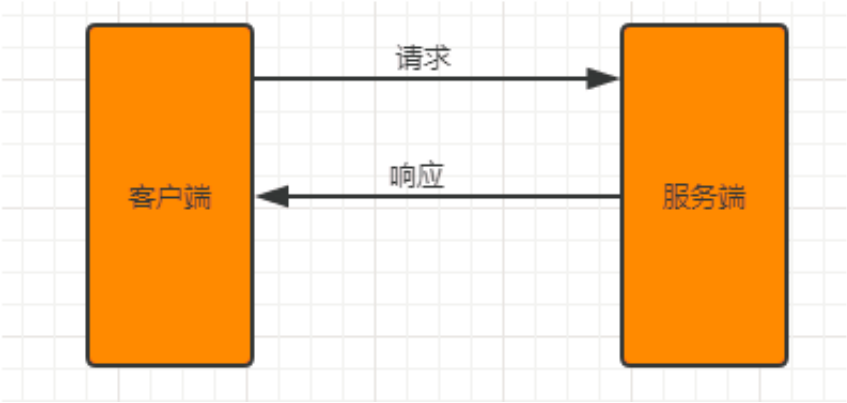
1）、什么是代理

代理服务器，客户机在发送请求时，不会直接发送给目的主机，而是先发送给代理服务器，代理服务接受客户机请求之后，再向主机发出，并接收目的主机返回的数据，存放在代理服务器的硬盘中，再发送给客户机。



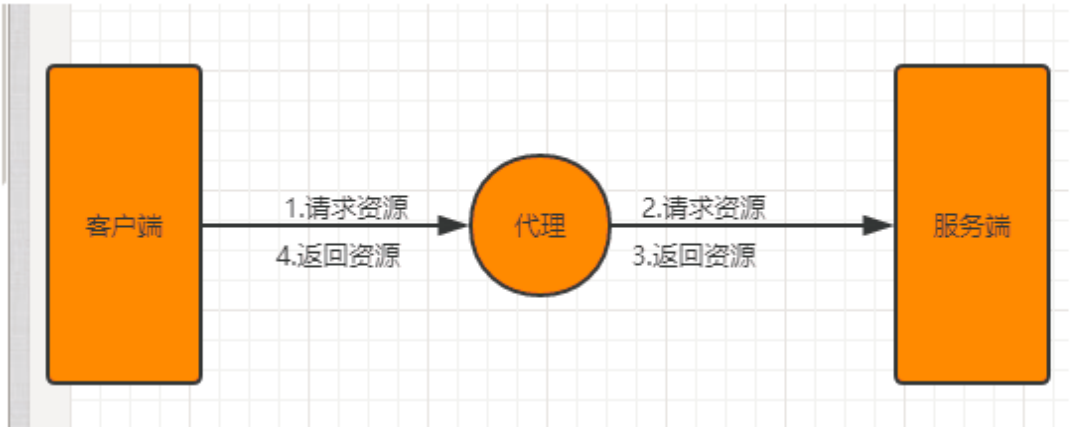
2）、没有代理

在没有代理的情况下，客户端和nginx服务端，都是客户端，都是客户端之间请求服务端，服务端则直接响应客户端。



3）、企业场景

在互联网请求里面，客户端往往无法直接向服务端发起请求，那么就需要实现客户端和服务通信。



4）、代理模式

nginx 作为代理服务，按照应用场景进行总结，代理分为正向代理，反向代理

（1）正向代理

正向代理:**客户端 <—> 代理 —>服务端**:

正向代理简单地打个租房的比方:

A（租客）B（中介）C（房东）

A(客户端)想租C(服务端)的房子,但是A(客户端)并不认识C(服务端)租不到。 B(代理)认识C(服务端)能租这个房子所以你找了B(代理)帮忙租到了这个房子。

在这个过程中:

C(服务端)不认识A(客户端)只认识B(代理) C(服务端)并不知道A(客户端)租了房子，只知道房子租给了B(代理) 。而A（租客）是知道自己租的是C（房东）的房子

正向代理，架设在客户机与目标主机之间，只用于代理内部网络对Internet的连接请求，客户机必须指定代理服务器,并将本来要直接发送到Web服务器上的http请求发送到代理服务器中。

（2）反向代理

反向代理:**客户端 —>代理 <—> 服务端**

反向代理也用一个租房的例子:

A(客户端)想租一个房子,B(代理)就把这个房子租给了他。 这时候实际上C(服务端)才是房东。 B(代理)是中介把这个房子租给了A(客户端)。这个过程中A(客户端)并不知道这个房子到底谁才是房东 他都有可能认为这个房子就是B(代理)的。

反向代理服务器架设在服务器端，通过缓冲经常被请求的页面来缓解服务器的工作量，将客户机请求转发给内部网络上的目标服务器；并将从服务器上得到的结果返回给Internet上请求连接的客户端，此时代理服务器与目标主机一起对外表现为一个服务器。

5、反向代理的作用

现在许多大型web网站都用到反向代理。除了可以防止外网对内网服务器的恶性攻击、缓存以减少服务器的压力和访问安全控制之外，还可以进行负载均衡，将用户请求分配给多个服务器。

1）提高访问速度 由于目标主机返回的数据会存放在代理服务器的硬盘中，因此下一次客户再访问相同的站点数据时，会直接从代理服务器的硬盘中读取，起到了缓存的作用，尤其对于热门站点能明显提高请求速度。

2）防火墙作用 由于所有的客户机请求都必须通过代理服务器访问远程站点，因此可在代理服务器上设限，过滤某些不安全信息。

3）通过代理服务器访问不能访问的目标站点 互联网上有许多开发的代理服务器，客户机在访问受限时，可通过不受限的代理服务器访问目标站点，通俗说，我们使用的翻墙浏览器就是利用了代理服务器，虽然不能出国，但也可直接访问外网。

6、支持的代理协议

http 属于七层的应用层 代理 超文本传输协议

https 代理 http/https协议

TCP 属于四层传输层 代理tcp/dupxiey

websocket 用于开发 代理http1.1长链接 通讯协议

GRPC 代理go语言远程调用

POP/IMAP 代理邮件收发协议 RTMP 代理 流媒体，直播

7、配置反向代理

```
user  nginx;
worker_processes  1;

error_log  /var/log/nginx/error.log warn;
pid        /var/run/nginx.pid;


events {
    worker_connections  1024;
}


http {
    include       /etc/nginx/mime.types;
    default_type  application/octet-stream;

    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  /var/log/nginx/access.log  main;

    sendfile        on;
    #tcp_nopush     on;

    keepalive_timeout  65;

    #gzip  on;

    include /etc/nginx/conf.d/*.conf;
    # www.91souyun.com
    server {
listen       80;
        listen       [::]:80;
        server_name  www.91souyun.com;
        root         /usr/share/nginx/html/www.91souyun.com;

        location / {
            root    /usr/share/nginx/html/www.91souyun.com;
            index   index.html index.htm;
        }
    }

    # blog.91souyun.com
    server {
        listen       80;
        listen       [::]:80;
        server_name  blog.91souyun.com;
        root         /usr/share/nginx/html/blog.91souyun.com;

        location / {
            root    /usr/share/nginx/html/blog.91souyun.com;
            index   index.html index.htm;
        }
    }

    #
    server {
        listen 80;
        server_name bw.91souyun.com;
        #index index.html index.htm index.php;

        location / {
                proxy_pass http://www.homosafe.com;
                root html;
                index index.html index.htm index.php;
```

```
                expires -1;
                proxy_set_header HOST $http_host;
                proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;

                proxy_http_version 1.1;
                proxy_connect_timeout 60s;
                proxy_read_timeout 60s;
                proxy_send_timeout 60s;
                proxy_buffering on;
                proxy_buffer_size 8k;
                proxy_buffers 8 8k;
        }
    }
}
```

2）、设置发往后端服务器的请求头信息

```
# 用户请求的时候HOST的值是www.oldboy.com，那么代理服务会像后端传递请求的还是
www.oldboy.com
proxy_set_header Host $http_host;
# 将$remote_addr的值放进变量X-Real-IP中，$remote_addr的值为客户端的ip
proxy_set_header X-Real-IP $remote_addr;
# 客户端通过代理服务访问后端服务，后端服务通过该变量会记录真实客户端地址
proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
```

3）、代理到后端的设置

```
# nginx代理与后端服务器连接超时时间(代理连接超时)
Syntax: proxy_connect_timeout time;
Default: proxy_connect_timeout 60s;
Context: http, server, location

# nginx代理等待后端服务器的响应时间
Syntax:    proxy_read_timeout time;
Default:    proxy_read_timeout 60s;
Context:    http, server, location

# 后端服务器数据回传给nginx代理超时时间
Syntax: proxy_send_timeout time;
Default: proxy_send_timeout 60s;
Context: http, server, location
```

8、负载均衡

1）、负载均衡的目的

提升吞吐率，提升请求性能， 提高高容灾。

2）、负载均衡的实现

Nginx 实现负载均衡用到了 proxy_pass 代理模块核心配置, 将客户端请求代理转发至一组 upstream 虚拟服务池。

3）、与反向代理的区别

负载均衡和反向代理的区别是，反向代理由代理服务器指定特定的服务器去请求资源，而负载均衡中的代理服务器将请求转发给虚拟服务池，具体由那个服务器处理根据相应的算法来定。

4）、负载均衡的配置

（1）配置web服务

```
## server.conf
server  {
  listen 8081;
  root  /soft/code1;
  index index.html;
}
server  {
  listen 8082;
  root  /soft/code2;
  index index.html;
}
server  {
  listen 8083;
  root  /soft/code3;
  index index.html;
}
```

（2）、配置负载均衡

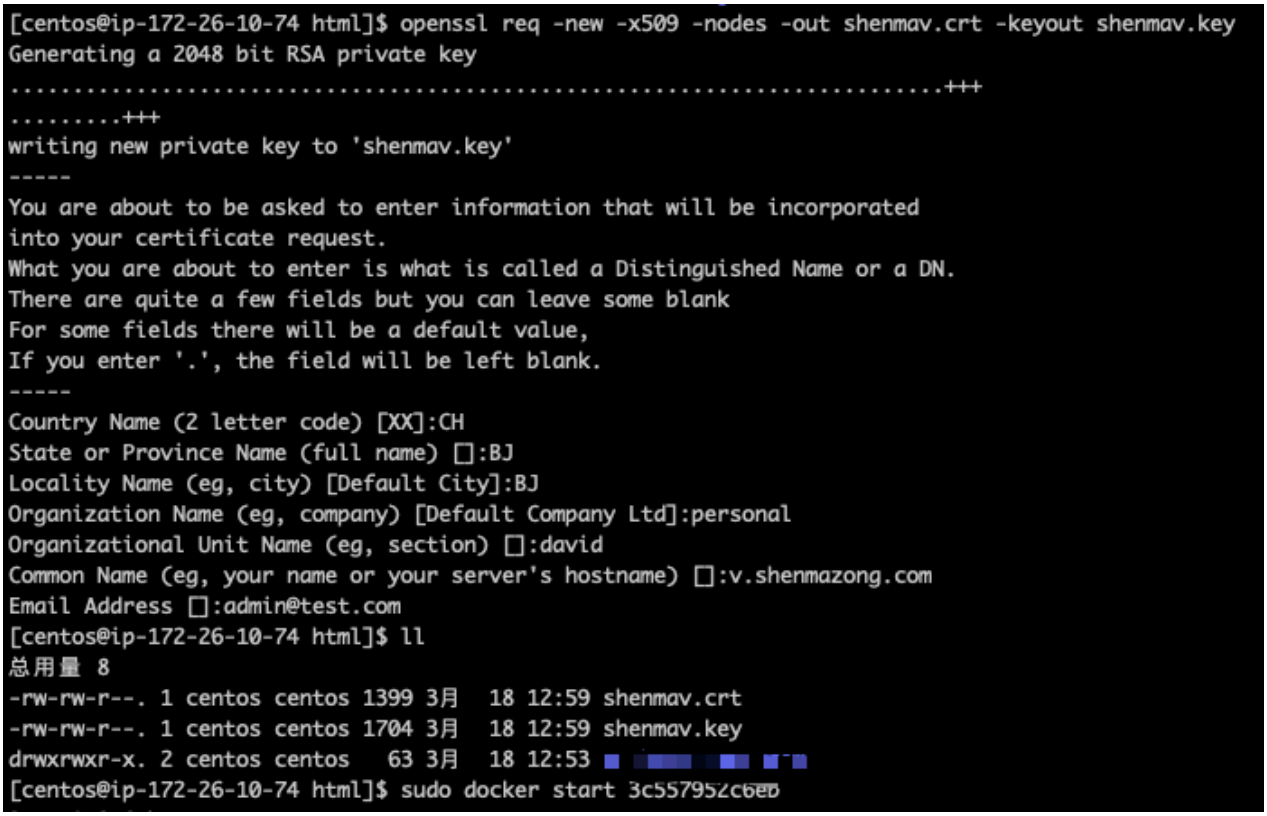
```
## proxy.conf
upstream  node  {
  server  你的IP:8081;
  server  你的IP:8082;
  server  你的IP:8083;
}
server  {
  server_name localhost;
  listen  80;
    location  / {
    proxy_pass  http://node;
    proxy_set_header    Host    $http_host;
    proxy_set_header    X-Real-IP   $remote_addr;
    proxy_set_header    X-Forwarded-For $proxy_add_x_forwarded_for;
  }
```

```
}
```

9、ngin配置https

1）自定义证书

```
# 生成证书
openssl req -new -x509 -nodes -out shenmav.crt -keyout shenmav.key
```



2）、nginx配置

```
## nginx 配置
user  nginx;
worker_processes  1;

error_log  /var/log/nginx/error.log warn;
pid        /var/run/nginx.pid;


events {
    worker_connections  1024;
}


http {
    include       /etc/nginx/mime.types;
```

```
    default_type  application/octet-stream;

    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  /var/log/nginx/access.log  main;

    sendfile        on;
    #tcp_nopush     on;

    keepalive_timeout  65;

    #gzip  on;

    include /etc/nginx/conf.d/*.conf;

    # v.shenmazong.com
    server {
        listen       443 default ssl;
        server_name  v.shenmazong.com;

        ssl_certificate /usr/share/nginx/html/shenmav.crt;
        ssl_certificate_key /usr/share/nginx/html/shenmav.key;

        ssl_session_cache    shared:SSL:1m;
        ssl_session_timeout  5m;

        server_tokens off;
        keepalive_timeout 70;

        ssl_ciphers  HIGH:!aNULL:!MD5;
        ssl_prefer_server_ciphers  on;

        location / {
            proxy_pass https://www.google.com;
            root html;
            index index.html index.htm;

            expires -1;
            #proxy_set_header HOST $http_host;
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;

            proxy_http_version 1.1;
            proxy_connect_timeout 60s;
            proxy_read_timeout 60s;
            proxy_send_timeout 60s;
            proxy_buffering on;
            proxy_buffer_size 8k;
```

```
            proxy_buffers 8 8k;
        }
    }
}
```