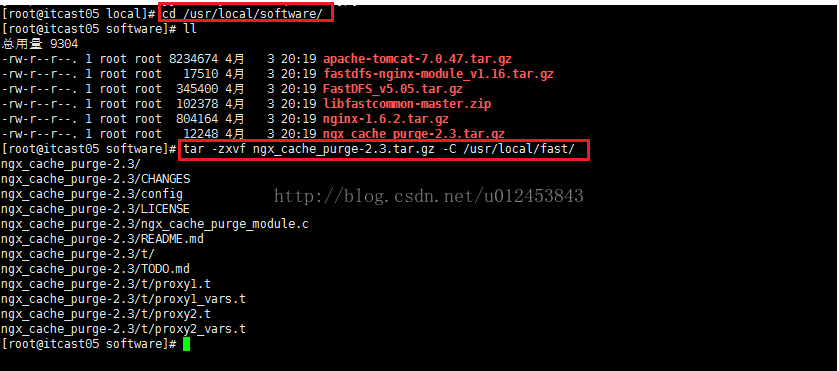
    由于博客图片量大，篇幅太长，因此需要分上、中、下三篇博客来写，上篇和中篇我们已经一起学习完了，这篇博客我们一起学习下剩余部分。

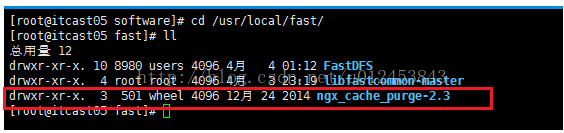
一、配置反向代理

       我们需要在两个跟踪器上安装nginx（也就是192.168.156.5和192.168.156.6）以提供反向代理服务，目的是使用统一的一个IP地址对外提供服务。为了避免一些不必要的错误，我们先把其它四台虚拟机的窗口关掉。

       1.解压ngx\_cache\_purge-2.3.tar.gz，解压命令：tar -zxvf ngx\_cache\_purge-2.3.tar.gz -C /usr/local/fast/，如下图所示（另一台设备就不粘贴图片了）。



解压完之后我们在/usr/local/fast/目录下可以看到多了一个ngx\_cache\_purge-2.3文件夹。如下图所示。



    2.下载需要的依赖库，在两台设备上依次执行下面四条命令。

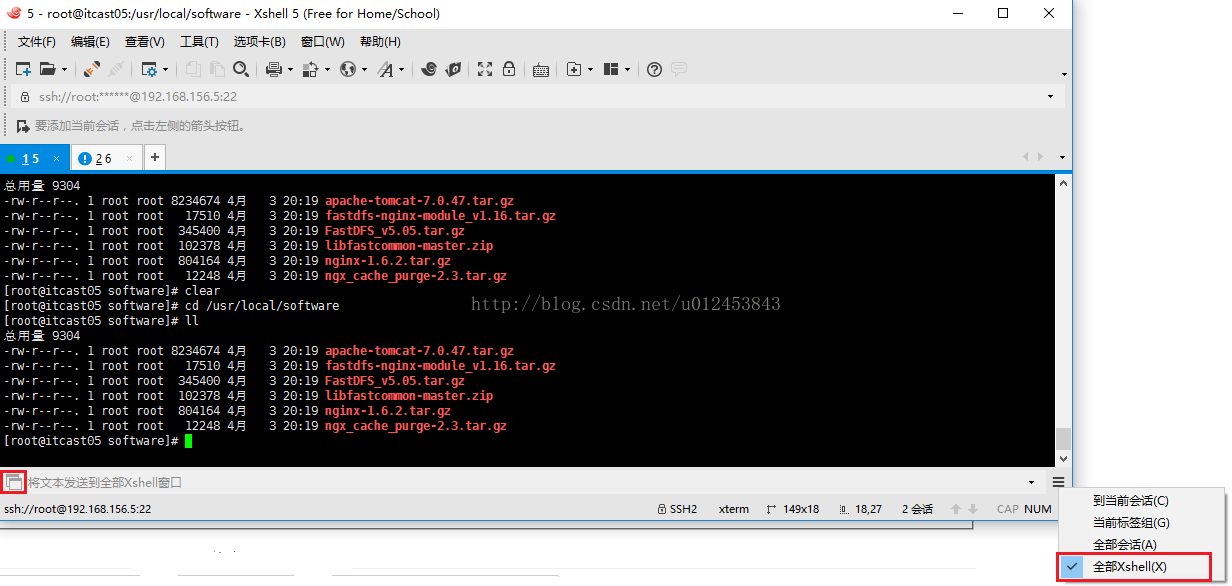
yum install pcre

yum install pcre-devel

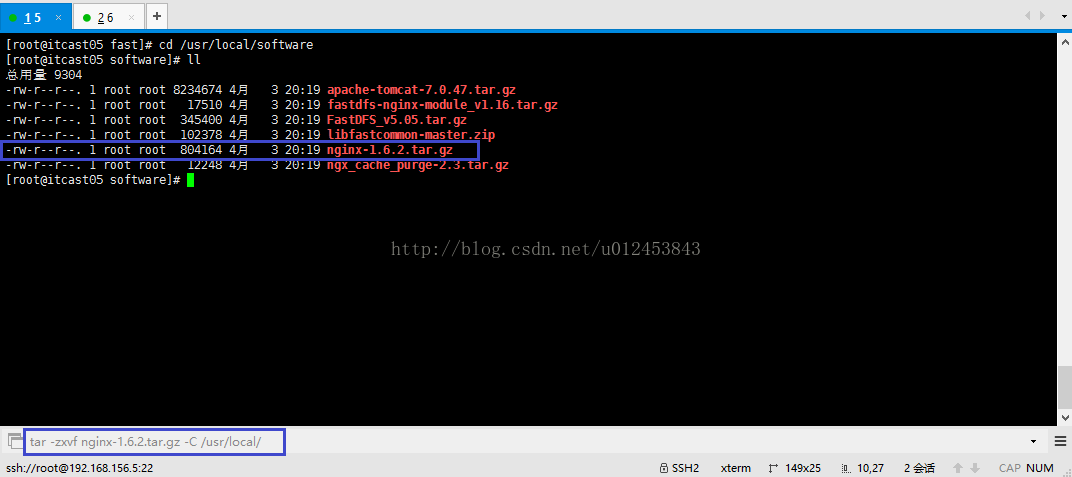
yum install zlib

yum install zlib-devel

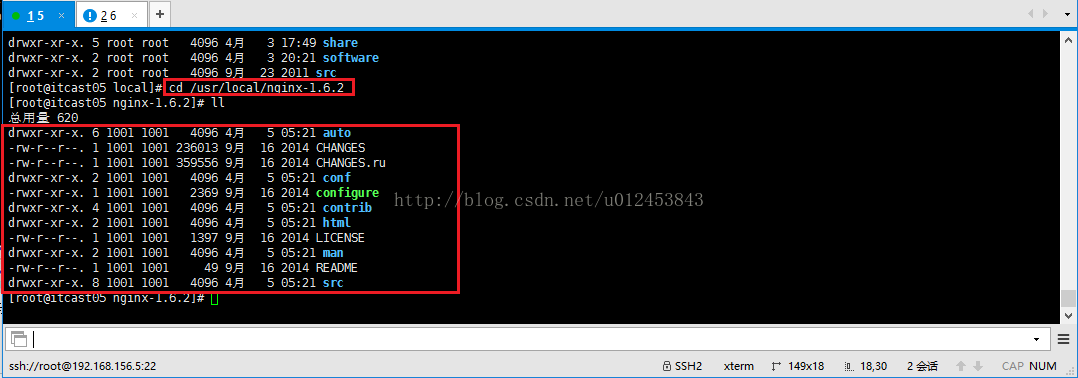
     3.为两台设备都安装nginx，我们在XShell的下方的输入框中输入命令：cd /usr/local/software/并敲回车，两个窗口都会进入到/usr/local/software目录下，然后在下面的输入框再输入"ll"来查看/usr/local/software目录下的文件，如下图所示（只有输入框左边的图标是多窗口的情况下才能一次作用所有窗口，如果当前是单窗口图标，就如下图那样选择全部XShell）。



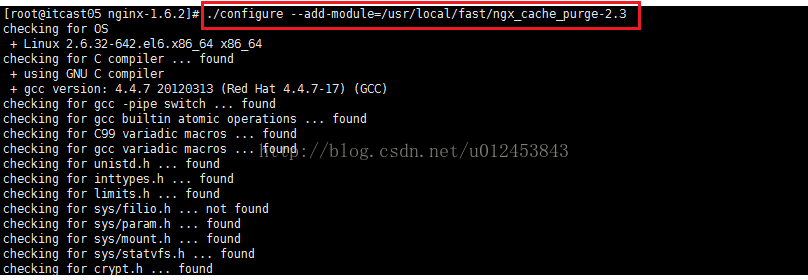
接着，我们在下面的输入框中输入：tar -zxvf nginx-1.6.2.tar.gz -C /usr/local/并按回车，会在两个窗口同时执行解压操作。如下图所示。



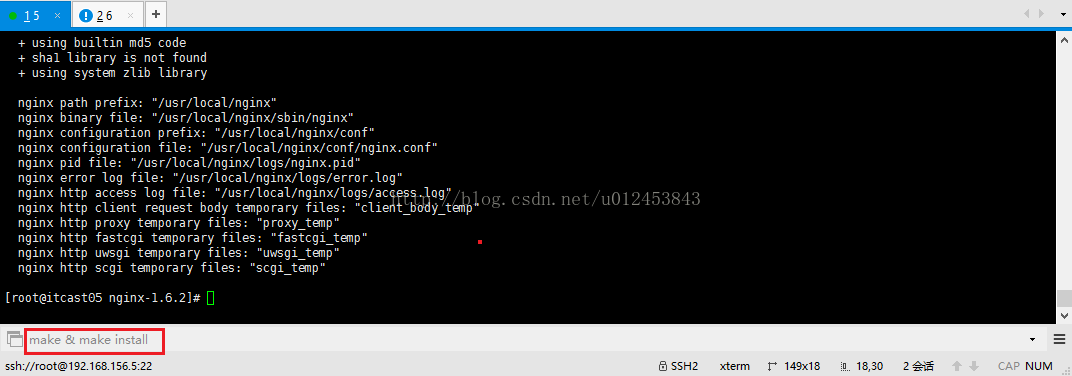
接下来我们在下面的输入框中输入：cd /usr/local并按回车，两台设备都进入到/usr/local/nginx-1.6.2目录下。如下图所示。



接着，在下面的输入框中加入模块命令：./configure --add-module=/usr/local/fast/ngx\_cache\_purge-2.3，回车就会在两台设备上都执行添加cache模块并会检查环境。



接着在下面的输入框中输入命令：make && make install，回车就会在两台设备上都执行编译安装。如下图所示。



 下面我们需要修改下两台设备/usr/local/nginx/conf/目录下的nginx.conf文件，大家可以直接把下面代码替换这个目录下的该文件，也可以直接到：http://download.csdn.net/detail/u012453843/9803673这个地址下载nginx.conf文件来替换。不过由于我们搭建环境的虚拟机IP可能不一样，因此，我们需要根据实际情况修改下IP等信息。（注意192.168.156.5和192.168.156.6这两台设备的/usr/local/nginx/conf/目录下的nginx.conf都要修改）

#user nobody;

worker\_processes 1;

#error\_log logs/error.log;

#error\_log logs/error.log notice;

#error\_log logs/error.log info;

#pid logs/nginx.pid;

events {

worker\_connections 1024;

use epoll;

}

http {

include 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 logs/access.log main;

sendfile on;

tcp\_nopush on;

#tcp\_nopush on;

#keepalive\_timeout 0;

keepalive\_timeout 65;

#gzip on;

#设置缓存

server\_names\_hash\_bucket\_size 128;

client\_header\_buffer\_size 32k;

large\_client\_header\_buffers 4 32k;

client\_max\_body\_size 300m;

proxy\_redirect off;

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;

proxy\_connect\_timeout 90;

proxy\_send\_timeout 90;

proxy\_read\_timeout 90;

proxy\_buffer\_size 16k;

proxy\_buffers 4 64k;

proxy\_busy\_buffers\_size 128k;

proxy\_temp\_file\_write\_size 128k;

#设置缓存存储路径，存储方式，分别内存大小，磁盘最大空间，缓存期限

proxy\_cache\_path /fastdfs/cache/nginx/proxy\_cache levels=1:2

keys\_zone=http-cache:200m max\_size=1g inactive=30d;

proxy\_temp\_path /fastdfs/cache/nginx/proxy\_cache/tmp;

#group1的服务设置

upstream fdfs\_group1 {

server 192.168.156.7:8888 weight=1 max\_fails=2 fail\_timeout=30s;

server 192.168.156.8:8888 weight=1 max\_fails=2 fail\_timeout=30s;

}

#group2的服务设置

upstream fdfs\_group2 {

server 192.168.156.9:8888 weight=1 max\_fails=2 fail\_timeout=30s;

server 192.168.156.10:8888 weight=1 max\_fails=2 fail\_timeout=30s;

}

server {

listen 8000;

server\_name localhost;

#charset koi8-r;

#access\_log logs/host.access.log main;

#group1的负载均衡配置

location /group1/M00 {

proxy\_next\_upstream http\_502 http\_504 error timeout invalid\_header;

proxy\_cache http-cache;

proxy\_cache\_valid 200 304 12h;

proxy\_cache\_key $uri$is\_args$args;

#对应group1的服务设置

proxy\_pass http://fdfs\_group1;

expires 30d;

}

location /group2/M00 {

proxy\_next\_upstream http\_502 http\_504 error timeout invalid\_header;

proxy\_cache http-cache;

proxy\_cache\_valid 200 304 12h;

proxy\_cache\_key $uri$is\_args$args;

#对应group2的服务设置

proxy\_pass http://fdfs\_group2;

expires 30d;

}

location ~/purge(/.\*) {

allow 127.0.0.1;

allow 192.168.156.0/24;

deny all;

proxy\_cache\_purge http-cache $1$is\_args$args;

}

location / {

root html;

index index.html index.htm;

}

#error\_page 404 /404.html;

# redirect server error pages to the static page /50x.html

#

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

location = /50x.html {

root html;

}

# proxy the PHP scripts to Apache listening on 127.0.0.1:80

#

#location ~ \.php$ {

# proxy\_pass http://127.0.0.1;

#}

# pass the PHP scripts to FastCGI server listening on 127.0.0.1:9000

#

#location ~ \.php$ {

# root html;

# fastcgi\_pass 127.0.0.1:9000;

# fastcgi\_index index.php;

# fastcgi\_param SCRIPT\_FILENAME /scripts$fastcgi\_script\_name;

# include fastcgi\_params;

#}

# deny access to .htaccess files, if Apache's document root

# concurs with nginx's one

#

#location ~ /\.ht {

# deny all;

#}

}

# another virtual host using mix of IP-, name-, and port-based configuration

#

#server {

# listen 8000;

# listen somename:8080;

# server\_name somename alias another.alias;

# location / {

# root html;

# index index.html index.htm;

# }

#}

# HTTPS server

#

#server {

# listen 443 ssl;

# server\_name localhost;

# ssl\_certificate cert.pem;

# ssl\_certificate\_key cert.key;

# ssl\_session\_cache shared:SSL:1m;

# ssl\_session\_timeout 5m;

# ssl\_ciphers HIGH:!aNULL:!MD5;

# ssl\_prefer\_server\_ciphers on;

# location / {

# root html;

# index index.html index.htm;

# }

#}

}

      修改完nginx.conf文件之后，我们下面需要创建/fastdfs/cache/nginx/proxy\_cache和/fastdfs/cache/nginx/proxy\_cache/tmp目录，这是因为我们在nginx.conf文件中配置缓存路径时指定了该目录，但是这两个目录目前还不存在，因此我们需要在192.168.156.5和192.168.156.6这两台设备上都创建下这两个目录，由于涉及到多级，因此需要递归创建目录，使用命令：mkdir -p /fastdfs/cache/nginx/proxy\_cache和mkdir -p /fastdfs/cache/nginx/proxy\_cache/tmp，如下图所示。



   由于我们配置了两个tracker的访问端口是8000，而我们的防火墙是不允许访问该端口的，因此我们需要修改下防火墙，使其允许访问8000端口，这个操作我在上篇和中篇都介绍过了，这里就不啰嗦了。

       下面我们便来启动这两台设备上的nginx。启动所使用的命令是/usr/local/nginx/sbin/nginx。启动完之后，可以使用ps -ef | grep nginx命令来查看nginx是否正常启动，如果看到root       4027      1  0 08:18 ?        00:00:00 nginx: master process /usr/local/nginx/sbin/nginx这条信息，说明正常启动了。

[root@itcast05 conf]# /usr/local/nginx/sbin/nginx

[root@itcast05 conf]# ps -ef | grep nginx

root 4027 1 0 08:18 ? 00:00:00 nginx: master process /usr/local/nginx/sbin/nginx

nobody 4028 4027 0 08:18 ? 00:00:00 nginx: worker process

nobody 4029 4027 0 08:18 ? 00:00:00 nginx: cache manager process

nobody 4030 4027 0 08:18 ? 00:00:00 nginx: cache loader process

root 4032 1522 0 08:18 pts/0 00:00:00 grep nginx

两台设备都启动完nginx之后，我们再在192.168.156.5上上传两次次图片，第一次返回的路径是在group1下，第二次返回的路径是在group2下。

[root@itcast05 conf]# /usr/bin/fdfs\_upload\_file /etc/fdfs/client.conf /usr/local/3.jpg

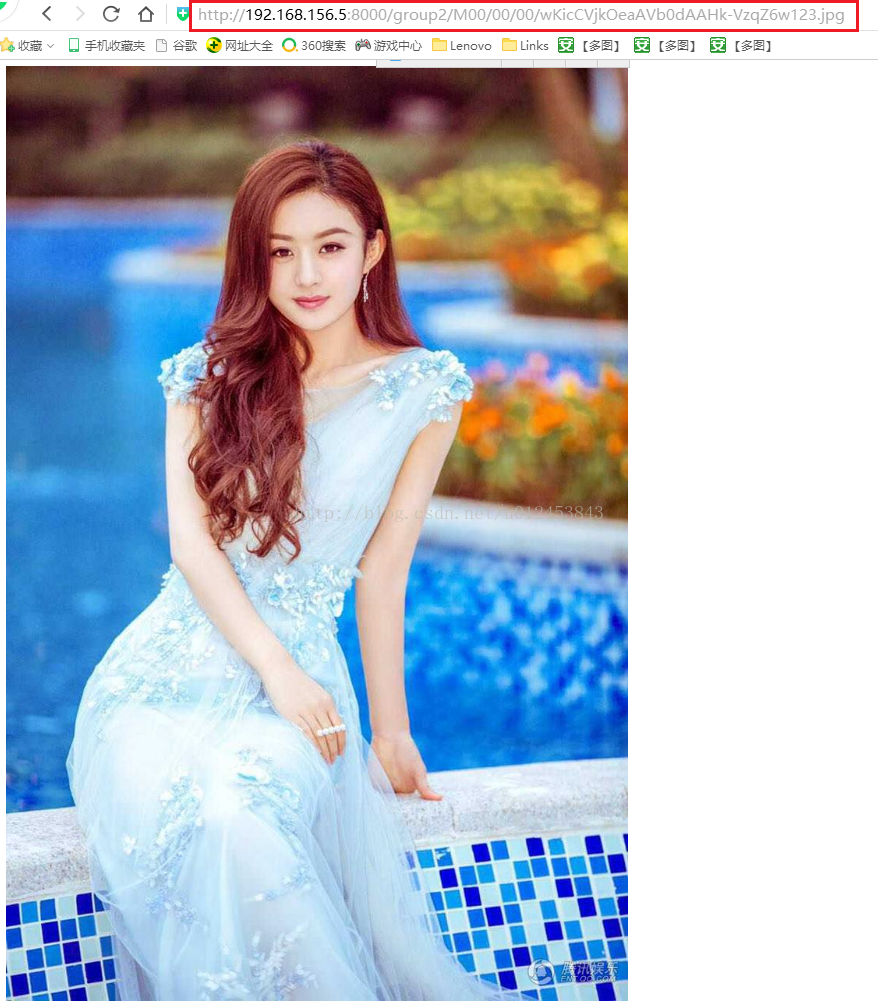
group1/M00/00/00/wKicCFjkOVGAMlQvAAHk-VzqZ6w757.jpg

[root@itcast05 conf]# /usr/bin/fdfs\_upload\_file /etc/fdfs/client.conf /usr/local/3.jpg

group2/M00/00/00/wKicCVjkOeaAVb0dAAHk-VzqZ6w123.jpg

[root@itcast05 conf]#

        由于我们在192.168.156.5和192.168.156.6上配置了代理，代理端口是8000，所以我们可以访问这两个IP的8000端口来访问我们刚才上传的图片，如下图所示（我们访问http://192.168.156.5:8000/group1/M00/00/00/wKicCFjkOVGAMlQvAAHk-VzqZ6w757.jpg也能访问到该图片）。这说明我们配置的代理完全没问题。



   我们知道，nginx对外提供服务有可能碰到服务挂掉的时候，这时候高可用就显得异常重要了，因此现在我们搭建一个nginx和keepalived结合实现的nginx集群高可用的环境，大家可以参考http://blog.csdn.net/u012453843/article/details/69668663这篇博客进行学习。

       我们现在要把keepalived实现的nginx集群高可用应用到我们的FastDFS集群当中，现在用于搭建nginx集群高可用的设备是192.168.156.11和192.168.156.12，我们只需要修改下这两台设备的nginx.conf文件，配置文件如下

#user nobody;

worker\_processes 1;

#error\_log logs/error.log;

#error\_log logs/error.log notice;

#error\_log logs/error.log info;

#pid logs/nginx.pid;

events {

worker\_connections 1024;

}

http {

include 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 logs/access.log main;

sendfile on;

#tcp\_nopush on;

#keepalive\_timeout 0;

keepalive\_timeout 65;

#gzip on;

upstream fastdfs\_tracker {

server 192.168.156.5:8000 weight=1 max\_fails=2 fail\_timeout=30s;

server 192.168.156.6:8000 weight=1 max\_fails=2 fail\_timeout=30s;

}

server {

listen 80;

server\_name localhost;

#charset koi8-r;

#access\_log logs/host.access.log main;

location /fastdfs {

root html;

index index.html index.htm;

proxy\_pass http://fastdfs\_tracker/;

proxy\_set\_header Host $http\_host;

proxy\_set\_header Cookie $http\_cookie;

proxy\_set\_header X-Real-IP $remote\_addr;

proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

proxy\_set\_header X-Forwarded-Proto $scheme;

client\_max\_body\_size 300m;

}

#error\_page 404 /404.html;

# redirect server error pages to the static page /50x.html

#

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

location = /50x.html {

root html;

}

}

}

我们对配置文件做了两处修改，一处是添加了负载均衡upstream fastdfs\_tracker，如下所示。我们是把192.168.156.5和192.168.156.6两台设备作为tracker，现在我们加了一层nginx来代理这两个tracker。

upstream fastdfs\_tracker {

server 192.168.156.5:8000 weight=1 max\_fails=2 fail\_timeout=30s;

server 192.168.156.6:8000 weight=1 max\_fails=2 fail\_timeout=30s;

}

第二处修改是添加了一个location并且匹配规则是路径当中有fastdfs。如下所示。

location /fastdfs {

root html;

index index.html index.htm;

proxy\_pass http://fastdfs\_tracker/;

proxy\_set\_header Host $http\_host;

proxy\_set\_header Cookie $http\_cookie;

proxy\_set\_header X-Real-IP $remote\_addr;

proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

proxy\_set\_header X-Forwarded-Proto $scheme;

client\_max\_body\_size 300m;

}

做好了修改之后，我们只需重新启动192.168.156.11和192.168.156.12这两台设备的nginx即可。

[root@nginx1 conf]# /usr/local/nginx/sbin/nginx -s reload

[root@nginx1 conf]#

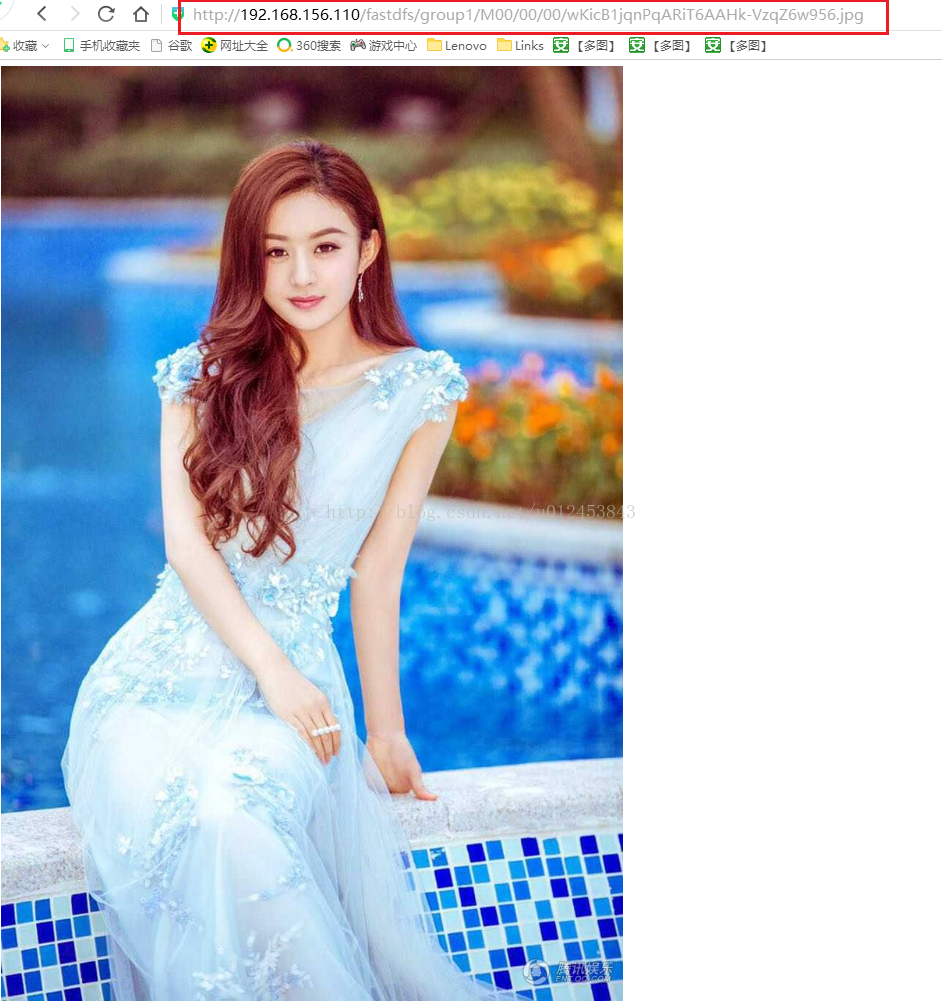
    这样我们便配置好了虚拟IP，现在我们从192.168.156.5再上传一张图片，如下所示。

[root@itcast05 conf]# /usr/bin/fdfs\_upload\_file /etc/fdfs/client.conf /usr/local/3.jpg

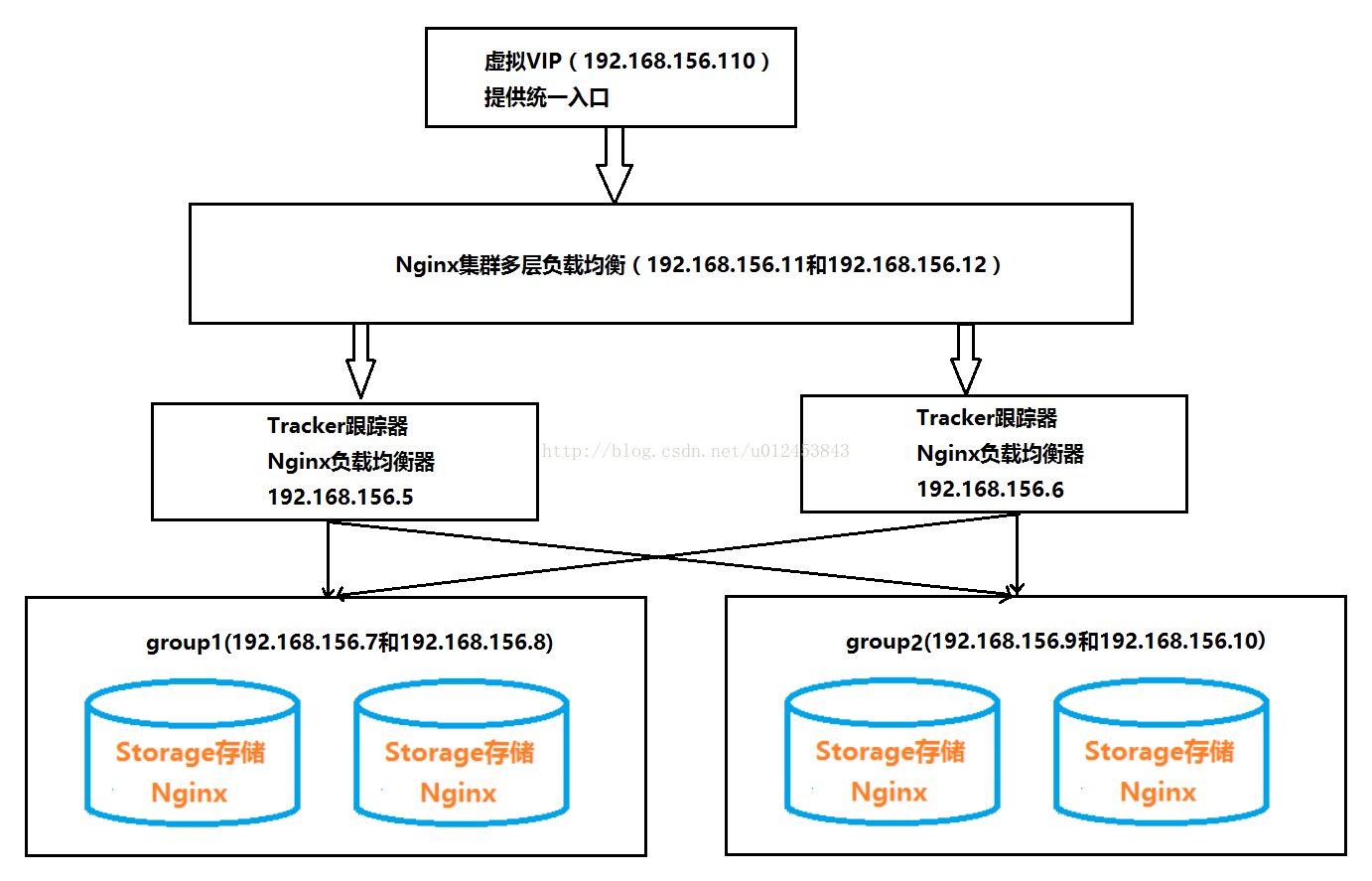
group1/M00/00/00/wKicB1jqnPqARiT6AAHk-VzqZ6w956.jpg

[root@itcast05 conf]#

   我们现在就用虚拟IP192.168.156.110来访问我们刚才上传的图片，只是注意在地址栏中要记得输入fastdfs（这是我们nginx.conf文件中location /fastdfs{}规则规定的）。如下图所示，发现，我们通过虚拟IP便可以访问我们上传的图片了。这样的好处是，对用户来说，只需要访问这个虚拟IP就可以了，不用关心FastDFS集群内部的转发机制。



 这样我们的FastDFS集群便搭建完了，搭建完后的集群图如下图所示。这个集群当中192.168.156.7、192.168.156.8、192.168.156.9、192.168.156.10这四台设备以8888端口对外提供服务，我们使用Tracker跟踪器管理这四台storage设备，两个Tracker的nginx对外提供的端口号是8000，也就是我们可以使用两台Tracker的任何一台设备的IP并且使用端口8000来访问存储在storage上的资源文件。其实我们完全可以在两台Tracker设备上搭建keepalived和nginx相结合的高可用环境并且对外提供虚拟IP192.168.156.110和端口80来访问资源文件。只不过这里为了展示多层nginx负载均衡所以才在192.168.156.11和192.168.156.12上专门搭建了keepalived和nginx相结合的高可用环境，由这两台设备对外提供虚拟IP服务，由于端口使用了默认的80，因此我们在使用虚拟IP192.168.156.110访问图片的时候才不用输入端口号的。



备注：启动集群步骤

        1.启动6台设备（192.168.156.5、192.168.156.6、192.168.156.7、192.168.156.8、192.168.156.9、192.168.156.10）的nginx（其中192.168.156.11和192.168.156.12配置了keepalived开机自启动，顺带会启动nginx，因此这两台设备不用启动nginx）

        2.启动tracker（192.168.156.5和192.168.156.6，启动命令：/etc/init.d/fdfs\_trackerd start）

        3.启动storage（192.168.156.7、192.168.156.8、192.168.156.9、192.168.156.10，启动命令：/etc/init.d/fdfs\_storaged start）

        这样FastDFS集群便都启动完了。