FastDFS图片服务器快速搭建（单机版）

# 前言

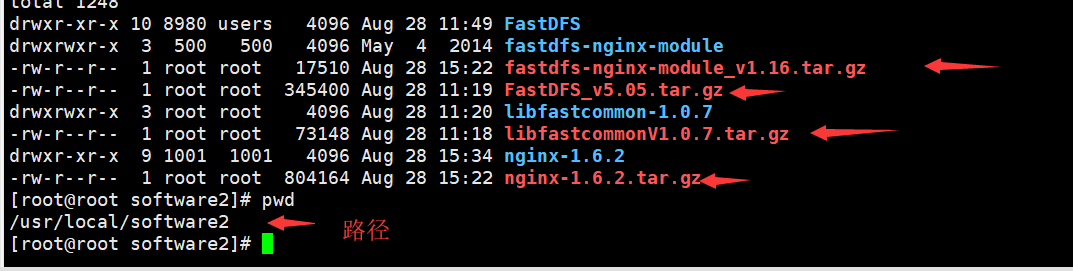
都是在同一个服务器中搭建（无第二个），在搭建过程中，不要被ip地址迷惑了

# 准备工作

## 需要上传的文件

tracker和storage使用相同的安装包，下载地址：http://sourceforge.net/projects/FastDFS/ 或https://github.com/happyfish100/FastDFS（推荐）

本教程下载：FastDFS\_v5.05.tar.gz



一共4个文件需要上传到服务器当中。

## 需要安装的环境(FastDFS)

安装FastDFS需要的环境

### 安装gcc

FastDFS是C语言开发，建议在linux上运行，本教程使用Centos7.2作为安装环境。

安装FastDFS需要先将官网下载的源码进行编译，编译依赖gcc环境，如果没有gcc环境，需要安装gcc：

yum install gcc-c++

### 安装libevent

FastDFS依赖libevent库，需要安装：

yum -y install libevent

# 安装上传的文件

## 安装libfastcommon

libfastcommon是FastDFS官方提供的，libfastcommon包含了FastDFS运行所需要的一些基础库。

将libfastcommonV1.0.7.tar.gz拷贝至/usr/local/下（我的路径图中有/usr/local/software2）

software2 是使用 mkdir software2 创建的

cd /usr/local

tar -zxvf libfastcommonV1.0.7.tar.gz

cd libfastcommon-1.0.7

./make.sh

./make.sh install

**注意：libfastcommon安装好后会自动将****库文件拷贝至****/usr/lib64下，由于FastDFS程序引用usr/lib目录所以需要将/usr/lib64下的库文件拷贝至/usr/lib下。**

**要拷贝的文件如下：**



将图中的文件拷贝至**/usr/lib/**

cp libfastcommon.so /usr/lib

## tracker编译安装

在192.168.101.3上安装tracker。

将FastDFS\_v5.05.tar.gz拷贝至/usr/local/下（我的路径图中有/usr/local/software2）

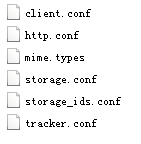
tar -zxvf FastDFS\_v5.05.tar.gz

cd FastDFS

./make.sh

./make.sh install

安装成功将安装目录下的conf下的文件拷贝到/etc/fdfs/下。



cp -r /usr/local/software2/FastDFS/conf/\* /etc/fdfs/

### 配置

安装成功后进入/etc/fdfs目录



修改tracker.conf

vi tracker.conf

base\_path=/home/yuqing/FastDFS

改为：

base\_path=/home/FastDFS

* 创建FastDFS的目录：

cd /home

mkdir FastDFS

### 启动

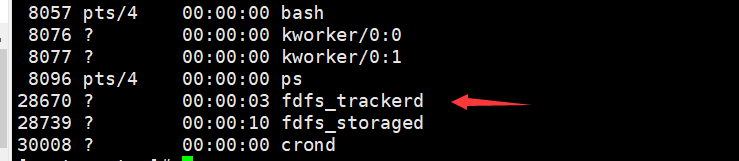
/usr/bin/fdfs\_trackerd /etc/fdfs/tracker.conf restart

启动的日志显示先停止5619进程（实际环境不是5619）再启动，如下图：



注意：如果没有显示上图要注意是否正常停止原有进程。（先杀掉有的进程，在重新启动）

不一定是5619这个进程（比如下图中我的进程是28670），可以使用ps -A 查看所有的进程



* 设置开机自动启动。

|  |
| --- |
| * [root@tracker FastDFS]# vim /etc/rc.d/rc.local |

将运行命令行添加进文件：/usr/bin/fdfs\_trackerd /etc/fdfs/tracker.conf restart

## Storage编译安装

在192.168.101.3上安装storage。

### 安装libevent

同tracker安装（在同一台服务器中就不需要在安装了因为和安装tracker步骤一样的）

### 安装libfastcommon

同tracker安装（在同一台服务器中就不需要在安装了因为和安装tracker步骤一样的

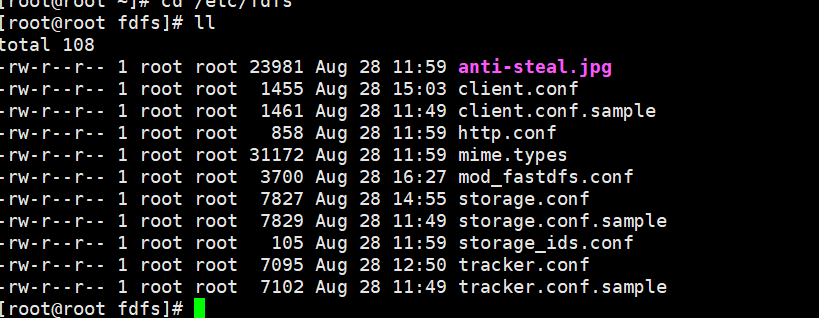
### storage编译安装

同tracker编译安装。（在同一台服务器中就不需要在编译了因为和安装tracker步骤一样的）

**上面三步在配置单机版中如果先配置了tracker就不需要安装了，直接跳过，直接到配置**

### 配置

安装成功后进入/etc/fdfs目录：



修改storage.conf

vi storage.conf

group\_name=group1

base\_path=/home/yuqing/FastDFS改为：base\_path=/home/FastDFS

store\_path0=/home/yuqing/FastDFS改为：store\_path0=/home/FastDFS/fdfs\_storage

#如果有多个挂载磁盘则定义多个store\_path，如下

#store\_path1=.....

#store\_path2=......  
tracker\_server=192.168.101.3:22122 #配置tracker服务器:IP

#如果有多个则配置多个tracker

tracker\_server=192.168.101.4:22122

* 创建fds\_storage

[root@localhost FastDFS]# cd /home/FastDFS/

[root@localhost FastDFS]# mkdir fdfs\_storage

### 启动

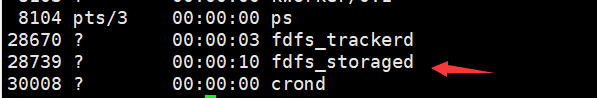
/usr/bin/fdfs\_storaged /etc/fdfs/storage.conf restart

启动的日志显示先停止8931进程（实际环境不是8931）再启动，如下图：



**注意：如果没有显示上图要注意是否正常停止原有进程。**

**不一定是8931这个进程（比如下图中我的进程是28739），可以使用ps -A 查看所有的进程**



* 设置开机自动启动。

|  |
| --- |
| [root@storage1 FastDFS]# vim /etc/rc.d/rc.local |

将运行命令行添加进文件：/usr/bin/fdfs\_storaged /etc/fdfs/storage.conf restart

# 上传图片测试

## 通过fdfs\_test程序

FastDFS安装成功可通过/usr/bin/fdfs\_test测试上传、下载等操作。

修改/etc/fdfs/client.conf

base\_path=/home/FastDFS

tracker\_server=192.168.101.3:22122

使用格式：

/usr/bin/fdfs\_test 客户端配置文件地址 upload 上传文件

比如将/home下的图片上传到FastDFS中：

/usr/bin/fdfs\_test /etc/fdfs/client.conf upload /home/tomcat.png

http://192.168.101.3/group1/M00/00/00/wKhlBVVY2M-AM\_9DAAAT7-0xdqM485\_big.png就是文件的下载路径。

对应storage服务器上的

/home/fastdfs/ \_storage/data/00/00/wKhlBVVY2M-AM\_9DAAAT7-0xdqM485\_big.png文件。

由于现在还没有和nginx整合无法使用http下载。

测试结果如下：

|  |
| --- |
| [root@localhost fdfs]# /usr/bin/fdfs\_test /etc/fdfs/client.conf upload /etc/fdfs/anti-steal.jpg  This is FastDFS client test program v5.05  Copyright (C) 2008, Happy Fish / YuQing  FastDFS may be copied only under the terms of the GNU General  Public License V3, which may be found in the FastDFS source kit.  Please visit the FastDFS Home Page http://www.csource.org/  for more detail.  [2017-04-05 00:29:04] DEBUG - base\_path=/home/FastDFS, connect\_timeout=30, network\_timeout=60, tracker\_server\_count=1, anti\_steal\_token=0, anti\_steal\_secret\_key length=0, use\_connection\_pool=0, g\_connection\_pool\_max\_idle\_time=3600s, use\_storage\_id=0, storage server id count: 0  tracker\_query\_storage\_store\_list\_without\_group:  server 1. group\_name=, ip\_addr=192.168.25.141, port=23000  group\_name=group1, ip\_addr=192.168.25.141, port=23000  storage\_upload\_by\_filename  group\_name=group1, remote\_filename=M00/00/00/wKgZjVjknMCAMd1sAABdrZgsqUU115.jpg  source ip address: 192.168.25.141  file timestamp=2017-04-05 00:29:04  file size=23981  file crc32=2553063749  example file url: http://192.168.25.141/group1/M00/00/00/wKgZjVjknMCAMd1sAABdrZgsqUU115.jpg  storage\_upload\_slave\_by\_filename  group\_name=group1, remote\_filename=M00/00/00/wKgZjVjknMCAMd1sAABdrZgsqUU115\_big.jpg  source ip address: 192.168.25.141  file timestamp=2017-04-05 00:29:04  file size=23981  file crc32=2553063749  example file url: http://192.168.25.141/group1/M00/00/00/wKgZjVjknMCAMd1sAABdrZgsqUU115\_big.jpg |

## 通过java Client API

部分测试代码。

|  |
| --- |
| **package com.antbbc.demo**;  **import org.csource.fastdfs.**\*;  **public class Test** {  **public static void** main(**String**[] args) **throws Exception** {  //1、加载配置文件 物理url  **ClientGlobal**.init("E:\\ideatools\\fastDFSdemo\\src\\main\\resources\\fdfs\_client.conf");  //2、构建一个管理者客户端  **TrackerClient** client = **new** TrackerClient();  //3、通过客户端来连接得到一个服务端对象  **TrackerServer** trackerServer = client.getConnection();  //4、声明构建一个存储的服务端  **StorageServer** storageServer = **null**;  //5、创建一个storage的client来获取存储服务器的客户端对象  **StorageClient** storageClient = **new** StorageClient(trackerServer, storageServer);  //6、上传文件  */\*第三个参数的文件的扩展信息，比如是当我打开图片的属性，  里面有详情信息（声明像素、宽度、长度）等都是可以指定的  指定为null也行  \*/* **String**[] strings= storageClient.upload\_file("E:\\imge\\2.jpg", "jpg", **null**);  //7、显示上传结果file\_id  **for** (**String** str:strings){  **System**.***out***.println(str);   }    } } |

记得把fdfs\_client.conf文件中的ip地址改了，改为你的服务器ip地址



fdfs\_client.conf 文件的内容如下：

tracker\_server=192.168.101.3:22122

**通过这段代码测试成功后，会返回一个file\_id，ip+file\_id就能进入访问图片了，但是这里没有配置nginx所有暂时还访问不了**

距离成功就还差一点，耐心点。骚年

# FastDFS 和nginx整合

## 在tracker上安装nginx

nginx的安装细节参考nginx文档（**nginx安装步骤下面有，不要着急，先做FastDFS-nginx-module**）。

在每个tracker上安装nginx，的主要目的是做负载均衡及实现高可用。如果只有一台tracker服务器可以不配置nginx。

## 在Storage上安装nginx

## FastDFS-nginx-module

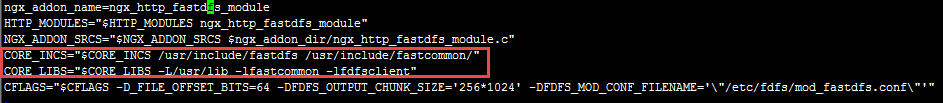
将FastDFS-nginx-module\_v1.16.tar.gz传至/usr/local/下

cd /usr/local

tar -zxvf FastDFS-nginx-module\_v1.16.tar.gz

cd FastDFS-nginx-module/src

修改config文件将/usr/local/路径改为/usr/



图中有3处需要修改不要着急

将FastDFS-nginx-module/src下的mod\_FastDFS.conf拷贝至/etc/fdfs/下

cp mod\_FastDFS.conf /etc/fdfs/

并在/etc/fdfs/ 目录中修改mod\_FastDFS.conf的内容：

vi /etc/fdfs/mod\_FastDFS.conf

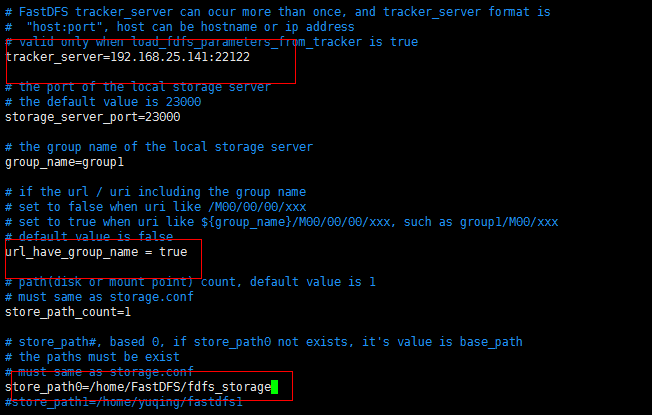
base\_path=/home/FastDFS

tracker\_server=192.168.101.3:22122

#tracker\_server=192.168.101.4:22122（多个tracker配置多行）

url\_have\_group\_name=true #url中包含group名称

store\_path0=/home/FastDFS/fdfs\_storage #指定文件存储路径



**图中的ip地址还是你自己的服务器地址，不要被迷惑了**

将libfdfsclient.so拷贝至/usr/lib下

cp /usr/lib64/libfdfsclient.so /usr/lib/



创建nginx/client目录

mkdir -p /var/temp/nginx/client

# Nginx安装

* 上传ningx的安装包：nginx-1.8.0.tar.gz 到/usr/local/下
* **我使用的是1.6.2版本的比较稳定**
* 解压 tar -xvf nginx-1.8.0.tar.gz
* cd 到nginx-1.8.0
* **添加FastDFS-nginx-module模块**
* **下面一步不要冲动看完在执行**
* ./configure \
* --prefix=/usr/local/nginx \
* --pid-path=/var/run/nginx/nginx.pid \
* --lock-path=/var/lock/nginx.lock \
* --error-log-path=/var/log/nginx/error.log \
* --http-log-path=/var/log/nginx/access.log \
* --with-http\_gzip\_static\_module \
* --http-client-body-temp-path=/var/temp/nginx/client \
* --http-proxy-temp-path=/var/temp/nginx/proxy \
* --http-fastcgi-temp-path=/var/temp/nginx/fastcgi \
* --http-uwsgi-temp-path=/var/temp/nginx/uwsgi \
* --http-scgi-temp-path=/var/temp/nginx/scgi \
* --add-module=/usr/local/fastdfs-nginx-module/src

--add-module=/usr/local/fastdfs-nginx-module/src 这个为你解压的fastdfs-nginx-module的目录

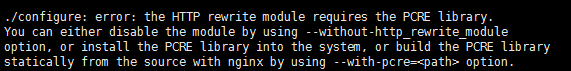
我的是：/usr/local/software2/fastdfs-nginx-module/src

所有根据根据自己的来添加

--add-module=/usr/local/software2/fastdfs-nginx-module/src

**注意：上边将临时文件目录指定为/var/temp/nginx，需要在/var下创建temp及nginx目录**

报错如下：



## 安装nginx的环境

需要安装nginx的安装环境：

**yum install gcc-c++（如果已经安装则不需要再安装）**

**yum install -y pcre pcre-devel**

**yum install -y zlib zlib-devel**

**yum install -y openssl openssl-devel**

**有兴趣可以阅读下，下面的资料有解释安装的到低是什么东西，如果安装的时候报错：**

错误： Multilib version problems found. This often means that the root

cause is something else and multilib version checking is just

pointing out that there is a problem. Eg.:

**阅读完，你就知道解决方案了。。。。**

nginx是C语言开发，建议在linux上运行，本教程使用Centos6.5作为安装环境。

* gcc

安装nginx需要先将官网下载的源码进行编译，编译依赖gcc环境，如果没有gcc环境，需要安装gcc：yum install gcc-c++

* PCRE

PCRE(Perl Compatible Regular Expressions)是一个Perl库，包括 perl 兼容的正则表达式库。nginx的http模块使用pcre来解析正则表达式，所以需要在linux上安装pcre库。

**yum install -y pcre pcre-devel**

注：pcre-devel是使用pcre开发的一个二次开发库。nginx也需要此库。

* zlib

zlib库提供了很多种压缩和解压缩的方式，nginx使用zlib对http包的内容进行gzip，所以需要在linux上安装zlib库。

**yum install -y zlib zlib-devel**

* openssl

OpenSSL 是一个强大的安全套接字层密码库，囊括主要的密码算法、常用的密钥和证书封装管理功能及SSL协议，并提供丰富的应用程序供测试或其它目的使用。

nginx不仅支持http协议，还支持https（即在ssl协议上传输http），所以需要在linux安装openssl库。

**yum install -y openssl openssl-devel**

上面如果报错解决方案：

它是一个保护多库版本：zlib-1.2.7-15.el7.x86\_64 != zlib-1.2.7-13.el7.i686

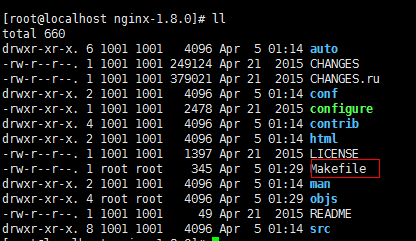
使用 yum install --setopt=protected\_multilib=false zlib

就能解决 ，当然这是在安装zlib 报的错就这个填写，如果是**openssl安装报错就应该**

yum install --setopt=protected\_multilib=false **openssl（举一反三，依次类推）**

再运行：./configure\ 开头的代码

运行成功：

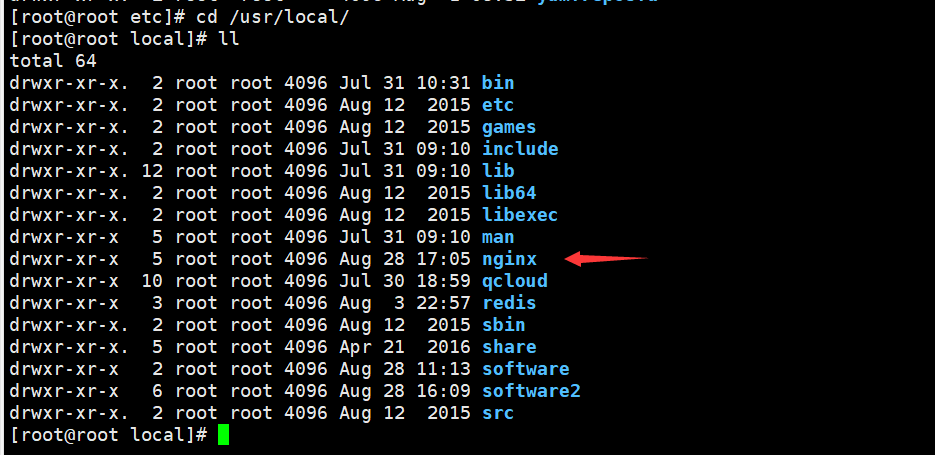


make（#编译）

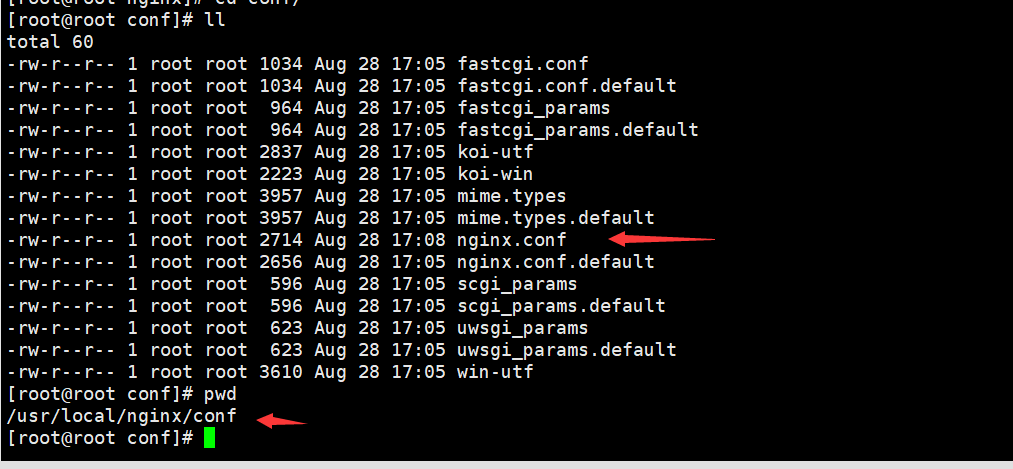
make install（#安装）

安装成功后会在

/usr/local/目录生成nginx文件夹



## 配置nginx配置文件



在 /usr/local/nginx/conf 目录配置

nginx.conf中

添加server:

server {

listen 80;

server\_name 192.168.101.3;

location /group1/M00/{

ngx\_fastdfs\_module;

}

}

不懂的话，看下图



说明：

server\_name指定本机ip

location /group1/M00/：group1为nginx 服务FastDFS的分组名称，M00是FastDFS自动生成编号，对应store\_path0=/home/FastDFS/fdfs\_storage，如果FastDFS定义store\_path1，这里就是M01

<https://blog.csdn.net/u012453843/article/details/69951920>（这个网站在配置nginx文件，可以学习下）

## 启动nginx

cd /usr/local/nginx/sbin/

./nginx （如果配置了指向配置文件nginx.conf可以使用此启动方式）否则不会加载nginx.conf文件

查询nginx进程：



15098是nginx主进程的进程id，15099是nginx工作进程的进程id

**注意：执行./nginx启动nginx，这里可以-c指定加载的nginx配置文件，如下：**

**./nginx -c /usr/local/nginx/conf/nginx.conf（使用这个启动）**

**如果不指定-c，nginx在启动时默认加载conf/nginx.conf文件，此文件的地址也可以在编译安装nginx时指定./configure的参数（--conf-path= 指向配置文件（nginx.conf）**

## 停止

方式1，快速停止：

cd /usr/local/nginx/sbin

./nginx -s stop

此方式相当于先查出nginx进程id再使用kill命令强制杀掉进程。

方式2，完整停止(建议使用)：

cd /usr/local/nginx/sbin

./nginx -s quit

此方式停止步骤是待nginx进程处理任务完毕进行停止。

## 重启ngingx。

方式1，先停止再启动（建议使用）：

对nginx进行重启相当于先停止nginx再启动nginx，即先执行停止命令再执行启动命令。

如下：

./nginx -s quit

./nginx

方式2，重新加载配置文件：

当nginx的配置文件nginx.conf修改后，要想让配置生效需要重启nginx，使用-s reload不用先停止nginx再启动nginx即可将配置信息在nginx中生效，如下：

./nginx -s reload

## 测试nginx

nginx安装成功，启动nginx，即可访问虚拟机上的nginx：



到这说明nginx上安装成功。

## 开机自启动nginx

### 编写shell脚本

这里使用的是编写shell脚本的方式来处理

vi /etc/init.d/nginx (输入下面的代码)

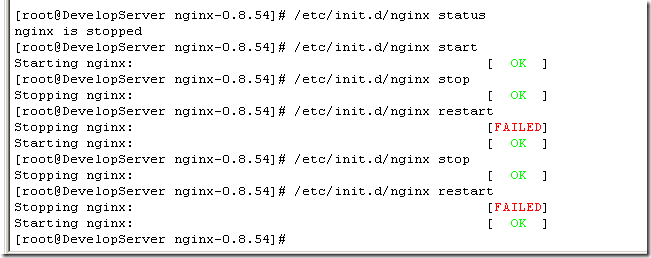
|  |
| --- |
| #!/bin/bash  # nginx Startup script for the Nginx HTTP Server  # it is v.0.0.2 version.  # chkconfig: - 85 15  # description: Nginx is a high-performance web and proxy server.  # It has a lot of features, but it's not for everyone.  # processname: nginx  # pidfile: /var/run/nginx.pid  # config: /usr/local/nginx/conf/nginx.conf  nginxd=/usr/local/nginx/sbin/nginx  nginx\_config=/usr/local/nginx/conf/nginx.conf  nginx\_pid=/var/run/nginx.pid  RETVAL=0  prog="nginx"  # Source function library.  . /etc/rc.d/init.d/functions  # Source networking configuration.  . /etc/sysconfig/network  # Check that networking is up.  [ ${NETWORKING} = "no" ] && exit 0  [ -x $nginxd ] || exit 0  # Start nginx daemons functions.  start() {  if [ -e $nginx\_pid ];then  echo "nginx already running...."  exit 1  fi  echo -n $"Starting $prog: "  daemon $nginxd -c ${nginx\_config}  RETVAL=$?  echo  [ $RETVAL = 0 ] && touch /var/lock/subsys/nginx  return $RETVAL  }  # Stop nginx daemons functions.  stop() {  echo -n $"Stopping $prog: "  killproc $nginxd  RETVAL=$?  echo  [ $RETVAL = 0 ] && rm -f /var/lock/subsys/nginx /var/run/nginx.pid  }  # reload nginx service functions.  reload() {  echo -n $"Reloading $prog: "  #kill -HUP `cat ${nginx\_pid}`  killproc $nginxd -HUP  RETVAL=$?  echo  }  # See how we were called.  case "$1" in  start)  start  ;;  stop)  stop  ;;  reload)  reload  ;;  restart)  stop  start  ;;  status)  status $prog  RETVAL=$?  ;;  \*)  echo $"Usage: $prog {start|stop|restart|reload|status|help}"  exit 1  esac  exit $RETVAL |

:wq 保存并退出

### 设置文件的访问权限

chmod a+x /etc/init.d/nginx (a+x ==> all user can execute 所有用户可执行)

这样在控制台就很容易的操作nginx了：查看Nginx当前状态、启动Nginx、停止Nginx、重启Nginx…



如果修改了nginx的配置文件nginx.conf，也可以使用上面的命令重新加载新的配置文件并运行，可以将此命令加入到rc.local文件中，这样开机的时候nginx就默认启动了

### 加入到rc.local文件中

vi /etc/rc.local

加入一行 /etc/init.d/nginx start 保存并退出，下次重启会生效。

## 测试图片上传

通过java客户端上传文件，使用浏览器访问，比如访问上传图片测试的文件：

访问storage：

http://192.168.101.3/group1/M00/00/00/wKhlBVVY2M-AM\_9DAAAT7-0xdqM485\_big.png

# 附录

## tracker.conf

1 基本配置

disable

#func：配置是否生效

#valu：true、false

disable=false

bind\_addr

#func：绑定IP

#valu：IP地址

bind\_addr=192.168.6.102

port

#func：服务端口

#valu：端口整数值

port=22122

connect\_timeout

#func：连接超时

#valu：秒单位正整数值

connect\_timeout=30

network\_timeout

#func：网络超时

#valu：秒单位正整数值

network\_timeout=60

base\_path

#func：Tracker数据/日志目录地址

#valu：路径

base\_path=/home/michael/fdfs/base4tracker

max\_connections

#func：最大连接数

#valu：正整数值

max\_connections=256

work\_threads

#func：线程数，通常设置CPU数

#valu：正整数值

work\_threads=4

store\_lookup

#func：上传文件的选组方式。

#valu：0、1或2。

# 0：表示轮询

# 1：表示指定组

# 2：表示存储负载均衡（选择剩余空间最大的组）

store\_lookup=2

store\_group

#func：指定上传的组，如果在应用层指定了具体的组，那么这个参数将不会起效。另外如果store\_lookup如果是0或2，则此参数无效。

#valu：group1等

store\_group=group1

store\_server

#func：上传服务器的选择方式。(一个文件被上传后，这个storage server就相当于这个文件的storage server源，会对同组的storage server推送这个文件达到同步效果)

#valu：0、1或2

# 0: 轮询方式（默认）

# 1: 根据ip 地址进行排序选择第一个服务器（IP地址最小者）

# 2: 根据优先级进行排序（上传优先级由storage server来设置，参数名为upload\_priority），优先级值越小优先级越高。

store\_server=0

store\_path

#func：上传路径的选择方式。storage server可以有多个存放文件的base path（可以理解为多个磁盘）。

#valu：

# 0: 轮流方式，多个目录依次存放文件

# 2: 存储负载均衡。选择剩余空间最大的目录存放文件（注意：剩余磁盘空间是动态的，因此存储到的目录或磁盘可能也是变化的）

store\_path=0

download\_server

#func：下载服务器的选择方式。

#valu：

# 0：轮询（默认）

# 1：IP最小者

# 2：优先级排序（值最小的，优先级最高。）

download\_server=0

reserved\_storage\_space

#func：保留空间值。如果某个组中的某个服务器的剩余自由空间小于设定值，则文件不会被上传到这个组。

#valu：

# G or g for gigabyte

# M or m for megabyte

# K or k for kilobyte

reserved\_storage\_space=1GB

log\_level

#func：日志级别

#valu：

# emerg for emergency

# alert

# crit for critical

# error

# warn for warning

# notice

# info for information

# debug for debugging

log\_level=info

run\_by\_group / run\_by\_user

#func：指定运行该程序的用户组

#valu：用户组名或空

run\_by\_group=

#func：

#valu：

run\_by\_user=

allow\_hosts

#func：可以连接到tracker server的ip范围。可设定多个值。

#valu

allow\_hosts=

check\_active\_interval

#func：检测 storage server 存活的时间隔，单位为秒。

# storage server定期向tracker server 发心跳，

# 如果tracker server在一个check\_active\_interval内还没有收到storage server的一次心跳，

# 那边将认为该storage server已经下线。所以本参数值必须大于storage server配置的心跳时间间隔。

# 通常配置为storage server心跳时间间隔的2倍或3倍。

check\_active\_interval=120

thread\_stack\_size

#func：设定线程栈的大小。 线程栈越大，一个线程占用的系统资源就越多。

# 如果要启动更多的线程（V1.x对应的参数为max\_connections，V2.0为work\_threads），可以适当降低本参数值。

#valu：如64KB，默认值为64，tracker server线程栈不应小于64KB

thread\_stack\_size=64KB

storage\_ip\_changed\_auto\_adjust

#func：这个参数控制当storage server IP地址改变时，集群是否自动调整。注：只有在storage server进程重启时才完成自动调整。

#valu：true或false

storage\_ip\_changed\_auto\_adjust=true

2 同步

storage\_sync\_file\_max\_delay

#func：同组storage服务器之间同步的最大延迟时间。存储服务器之间同步文件的最大延迟时间，根据实际情况进行调整

#valu：秒为单位，默认值为1天（24\*3600）

#sinc：v2.0

storage\_sync\_file\_max\_delay=86400

storage\_sync\_file\_max\_time

#func：存储服务器同步一个文件需要消耗的最大时间，缺省为300s，即5分钟。

#sinc：v2.0

storage\_sync\_file\_max\_time=300

sync\_log\_buff\_interval

#func：同步或刷新日志信息到硬盘的时间间隔。注意：tracker server 的日志不是时时写硬盘的，而是先写内存。

#valu：以秒为单位

sync\_log\_buff\_interval=10

3 trunk 和 slot

#func：是否使用trunk文件来存储几个小文件

#valu：true或false

#sinc：v3.0

use\_trunk\_file=false

#func：最小slot大小

#valu：<= 4KB，默认为256字节

#sinc：v3.0

slot\_min\_size=256

#func：最大slot大小

#valu：>= slot\_min\_size，当小于这个值的时候就存储到trunk file中。默认为16MB。

#sinc：v3.0

slot\_max\_size=16MB

#func：trunk file的size

#valu：>= 4MB，默认为64MB

#sinc：v3.0

trunk\_file\_size=64MB

4 HTTP 相关

是否启用 HTTP

#func：HTTP是否生效

#valu：true或false

http.disabled=false

HTTP 服务器端口号

#func：tracker server上的http port

#valu：

#note：只有http.disabled=false时才生效

http.server\_port=7271

检查Storage存活状态的间隔时间（心跳检测）

#func：检查storage http server存活的间隔时间

#valu：单位为秒

#note：只有http.disabled=false时才生效

http.check\_alive\_interval=30

心跳检测使用的协议方式

#func：检查storage http server存活的方式

#valu：

# tcp：连接到storage server的http端口，不进行request和response。

# http：storage check alive url must return http status 200.

#note：只有http.disabled=false时才生效

http.check\_alive\_type=tcp

检查 Storage 状态的 URI

#func：检查storage http server是否alive的uri/url

#note：只有http.disabled=false时才生效

http.check\_alive\_uri=/status.html

注意：

|  |
| --- |
| /usr/bin/fdfs\_storaged /etc/fdfs/storage.conf restart 重启storage  /usr/bin/fdfs\_trackerd /etc/fdfs/tracker.conf restart 重启tracker |

有三个地方配置了ip地址：

tracker.conf

storage.conf

client.conf

# [170217、nginx 安装时候报错：make: \*\*\* No rule to make target `build', needed by `default'. Stop.](https://www.cnblogs.com/zrbfree/p/6419043.html)

出现此种情况，是linux系统没有安装先决条件

**1、GCC——GNU编译器集合（GCC可以使用默认包管理器的仓库（repositories）来安装，包管理器的选择依赖于你使用的Linux发布版本，包管理器有不同的实现：yum是基于Red Hat的发布版本；apt用于Debian和Ubuntu；yast用于SuSE Linux等等。）**

**RedHat中安装GCC：**

yum install gcc

**Ubuntu中安装GCC：**

apt-get install gcc

**2、PCRE库（Nginx编译需要PCRE（Perl Compatible Regular Expression），因为Nginx的Rewrite模块和HTTP核心模块会使用到PCRE正则表达式语法。这里需要安装两个安装包pcre和pcre-devel。第一个安装包提供编译版本的库，而第二个提供开发阶段的头文件和编译项目的源代码，这正是我们需要的理由。）**

**RedHat中安装PCRE：**

yum install pcre pcre-devel

**Ubuntu中安装PCRE：**

apt-get install libpcre3 libpcre3-dev

**3、zlib库（zlib库提供了开发人员的压缩算法，在Nginx的各种模块中需要使用gzip压缩。如同安装PCRE一样，同样需要安装库和它的源代码：zlib和zlib-devel。）**

**RedHat中安装zlib：**

yum install zlib zlib-devel

**Ubuntu中安装zlib：**

apt-get install zlib1g zlib1g-dev

**4、OpenSSL库（在Nginx中，如果服务器提供安全网页时则会用到OpenSSL库，我们需要安装库文件和它的开发安装包（openssl和openssl-devel）。）**

**RedHat中安装OpenSSL：**

yum install openssl openssl-devel

**Ubuntu中安装OpenSSL：（注：Ubuntu14.04的仓库中没有发现openssl-dev）：**

apt-get install openssl openssl-dev

 注意：要不关闭防火墙，要不然放不了nginx页面

# fastdfs文件迁移：

一、先将要迁移的fastdfs环境配置好，并启动起来

  我是要将文件从**旧的fastdfs服务器**迁移到新的fastdfs文件服务器上来：

1、先将所有storage进程都停止：

/usr/local/fastdfs/bin/stop.sh /usr/local/fastdfs/bin/fdfs\_storaged（停止storage）

2、修改storage.conf文件

vi /etc/fdfs/storage.conf

将tracker\_server的IP设置成旧服务器IP192.168.1.53

tracker\_server=192.168.1.53:22122（原来的文件环境）

3、启动storage进程：

/usr/local/fastdfs/bin/fdfs\_storaged /etc/fdfs/storage.conf

此时storage服务器开始同步；

查看同步进程：

在旧tracker服务器执行：

fdfs\_monitor /etc/fdfs/storage.conf

可以看到同步进度；

4、同步完成后，在新的fastdfs文件服务器192.168.1.246上停止storage进程：

/usr/local/fastdfs/bin/stop.sh /usr/local/fastdfs/bin/fdfs\_storaged

5、修改storage.conf文件，将tracker修改成新的tracker IP

vi /etc/fdfs/storage.conf

tracker\_server=192.168.1.246:22122

6、修改.data\_init\_flag文件：

cd /usrdata/storage/data/

vi .data\_init\_flag

将sync\_src\_server配置项留空，其它配置项不变

sync\_src\_server=

7、启动storage进程

/usr/local/fastdfs/bin/fdfs\_storaged /etc/fdfs/storage.conf

查看storage进程状态：

fdfs\_monitor /etc/fdfs/storage.conf