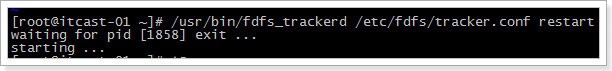
FastDFS+Nginx实现文件服务器

确认tracker是否启动

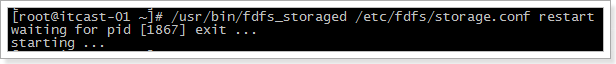
[root@itcast-01 ~]# /usr/bin/fdfs\_trackerd /etc/fdfs/tracker.conf restart

如下图关闭线程再执行则表示成功



确认storage是否启动

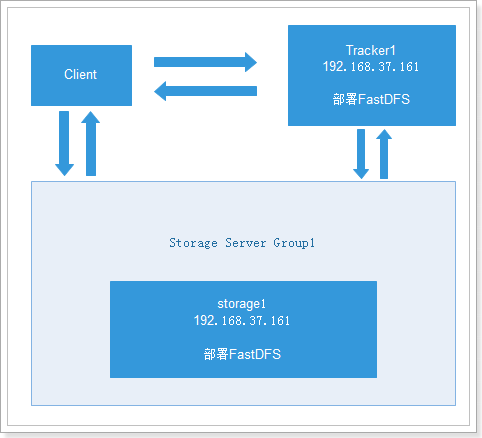
[root@itcast-01 ~]# /usr/bin/fdfs\_storaged /etc/fdfs/storage.conf restart



如果不成功，同样的命令再执行一次

# 架构

## 架构图



# FastDFS--tracker安装

在192.168.37.161上安装tracker。

## 下载

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

## FastDFS安装环境

FastDFS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。

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，

如果没有，需要安装：

yum -y install libevent

## 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。

上传

taotao\01课前资料\02技术资料\FastDFS\bin\libfastcommonV1.0.7.tar.gz上传到Linux中

将libfastcommonV1.0.7.tar.gz移动至/usr/local/下

[root@itcast-01 ~]# mv libfastcommonV1.0.7.tar.gz /usr/local/

进入/usr/local/，解压libfastcommon

[root@itcast-01 local]# cd /usr/local/

[root@itcast-01 local]# tar -zxvf libfastcommonV1.0.7.tar.gz

进入解压文件夹，编译，安装

[root@itcast-01 local]# cd libfastcommon-1.0.7/

编译：

[root@itcast-01 libfastcommon-1.0.7]# ./make.sh

安装：

[root@itcast-01 libfastcommon-1.0.7]# ./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。**



拷贝文件

[root@itcast-01 libfastcommon-1.0.7]# cp /usr/lib64/libfastcommon.so /usr/lib/

## tracker编译安装

将FastDFS\_v5.05.tar.gz拷贝至/usr/local/下

上传FastDFS\_v5.05.tar.gz

taotao\01课前资料\02技术资料\FastDFS\bin\FastDFS\_v5.05.tar.gz上传到Linux中

移动到/usr/local

[root@itcast-01 ~]# mv FastDFS\_v5.05.tar.gz /usr/local/

解压

[root@itcast-01 ~]# cd /usr/local/

[root@itcast-01 local]# tar -zxvf FastDFS\_v5.05.tar.gz

进入解压文件夹，编译，安装

[root@itcast-01 local]# cd FastDFS

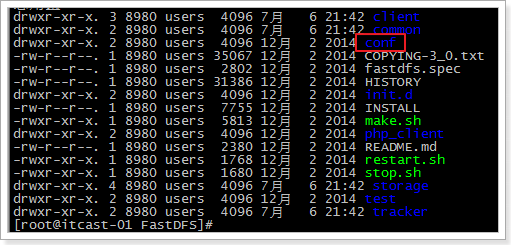
编译

[root@itcast-01 FastDFS]# ./make.sh

安装

[root@itcast-01 FastDFS]# ./make.sh install

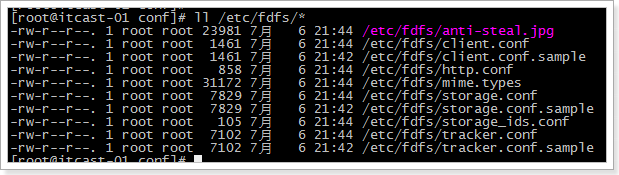
安装成功后，将安装目录下的conf下的文件拷贝到/etc/fdfs/下。



[root@itcast-01 FastDFS]# cd conf/

[root@itcast-01 conf]# cp \* /etc/fdfs/

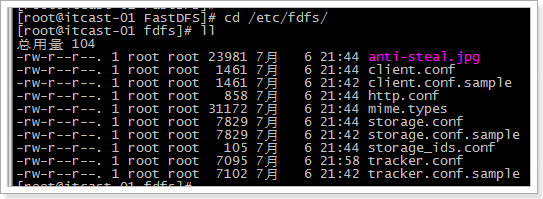
效果：



## 配置

安装成功后进入/etc/fdfs目录：

[root@itcast-01 FastDFS]# cd /etc/fdfs/



修改tracker.conf

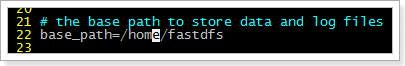
[root@itcast-01 fdfs]# vim tracker.conf

把

base\_path=/home/yuqing/fastdfs

改为：

base\_path=/home/fastdfs



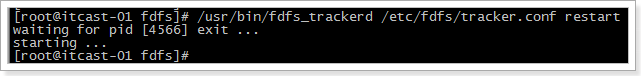
创建fastdfs文件夹

[root@itcast-01 fdfs]# mkdir /home/fastdfs

## 启动

[root@itcast-01 fdfs]# /usr/bin/fdfs\_trackerd /etc/fdfs/tracker.conf restart

多次restart，启动日志显示先停止4555进程（实际环境可能不是4555）再启动，如下图：

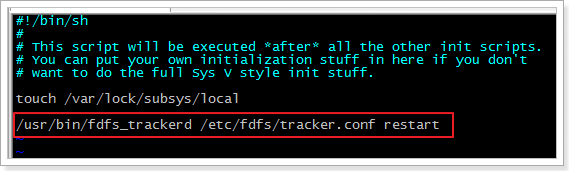


如果没有显示上图要确认原有进程是否正常停止。

设置开机自动启动。

[root@itcast-01 fdfs]# 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



# FastDFS--storage安装

在192.168.37.161上安装storage。

## 安装libevent

同tracker安装。（如果是一台虚拟机安装Tracker和Storage，这里可以省略）

## 安装libfastcommon

同tracker安装。（如果是一台虚拟机安装Tracker和Storage，这里可以省略）

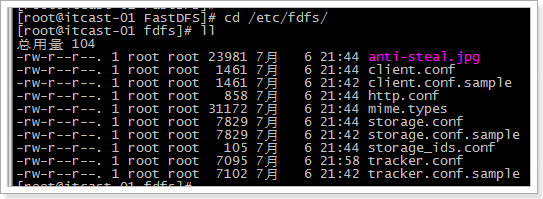
## storage编译安装

同tracker编译安装。（如果是一台虚拟机安装Tracker和Storage，这里可以省略）

## 配置

安装成功后进入/etc/fdfs目录：

[root@itcast-01 FastDFS]# cd /etc/fdfs/



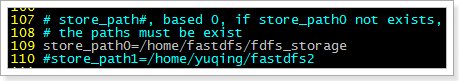
修改storage.conf

[root@itcast-01 fdfs]# vim storage.conf

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



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

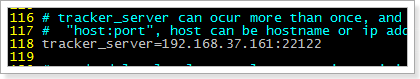


#如果有多个挂载磁盘则定义多个store\_path，如下

#store\_path1=.....

#store\_path2=......  
tracker\_server=192.168.37.161:22122 #配置tracker服务器:IP

#如果有多个则配置多个tracker



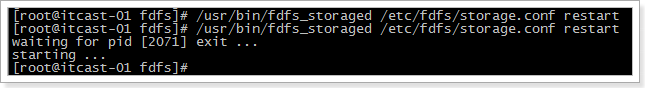
创建fdfs\_storage文件夹

[root@itcast-01 fdfs]# mkdir /home/fastdfs/fdfs\_storage

## 启动

[root@itcast-01 fdfs]# /usr/bin/fdfs\_storaged /etc/fdfs/storage.conf restart

多次restart ，启动的日志显示先停止2071进程（实际环境可能不是2071）再启动，如下图：

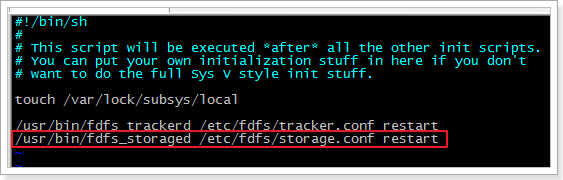


如果没有显示上图要确认原有进程是否正常停止。

设置开机自动启动。

[root@itcast-01 fdfs]# 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

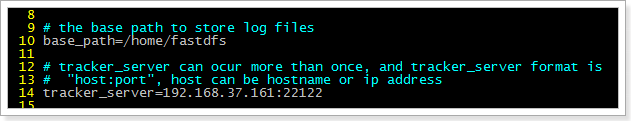
FastDFS安装成功可通过/usr/bin/fdfs\_test测试上传、下载等操作。

修改/etc/fdfs/client.conf

[root@itcast-01 ~]# vim /etc/fdfs/client.conf

base\_path=/home/fastdfs

tracker\_server=192.168.37.161:22122



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/root下的日志上传到FastDFS中：

[root@itcast-01 ~]# /usr/bin/fdfs\_test /etc/fdfs/client.conf upload install.log

返回信息

example file url: http://192.168.37.161/group1/M00/00/00/wKgloVd9Gg6ADCGQAACrB89lLCo483\_big.log

就是文件下载路径

对应storage服务器上的

/home/fastdfs/fdfs\_storage/data/00/00/wKgloVd9Gg6ADCGQAACrB89lLCo483\_big.log

由于现在还没有和nginx整合无法使用http下载。

# FastDFS 和nginx整合

## 在tracker上安装nginx

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。

## 在Storage上安装nginx

### FastDFS-nginx-module

上传

taotao\01课前资料\02技术资料\FastDFS\bin\fastdfs-nginx-module\_v1.16.tar.gz上传到Linux

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

[root@itcast-01 ~]# mv fastdfs-nginx-module\_v1.16.tar.gz /usr/local/

解压

[root@itcast-01 ~]# cd /usr/local/

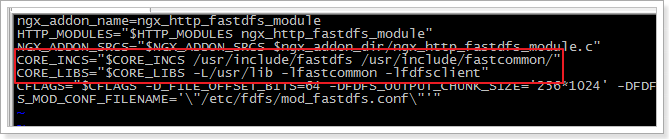
[root@itcast-01 local]# tar -zxvf fastdfs-nginx-module\_v1.16.tar.gz

编辑配置文件

[root@itcast-01 local]# cd fastdfs-nginx-module/src/

修改config文件将/usr/local/路径改为/usr/

[root@itcast-01 src]# vim config



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

[root@itcast-01 src]# cp mod\_fastdfs.conf /etc/fdfs/

并修改mod\_FastDFS.conf的内容：

[root@itcast-01 src]# vim /etc/fdfs/mod\_fastdfs.conf

base\_path=/home/fastdfs

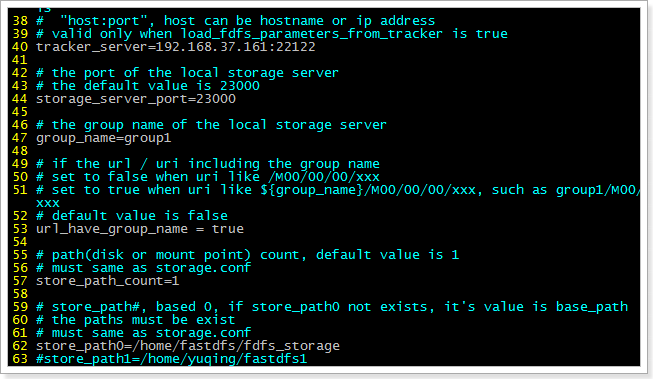


tracker\_server=192.168.37.161:22122

#tracker\_server=192.168.37.162:22122（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



将libfdfsclient.so拷贝至/usr/lib下

[root@itcast-01 src]# cp /usr/lib64/libfdfsclient.so /usr/lib/

创建nginx/client目录

[root@itcast-01 src]# mkdir -p /var/temp/nginx/client

### nginx安装

安装环境：

安装pcre库

yum -y install pcre-devel

安装zlib库

yum install -y zlib-devel

上传Nginx

上传\01课前资料\02技术资料\Nginx\nginx-1.7.7.tar.gz

解压，并进入解压文件夹

[root@itcast-01 ~]# tar -zxf nginx-1.7.7.tar.gz

[root@itcast-01 ~]# cd nginx-1.7.7

添加FastDFS-nginx-module模块

执行以下命令

./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 \

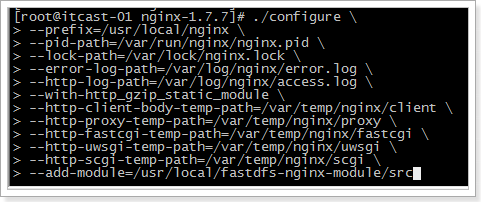
--http-proxy-temp-path=/var/temp/nginx/proxy \

--http-fastcgi-temp-path=/var/temp/nginx/fastcgi \

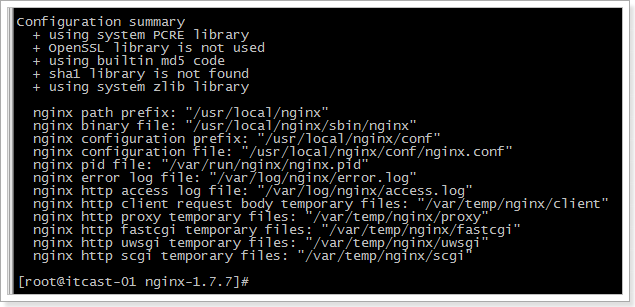
--http-uwsgi-temp-path=/var/temp/nginx/uwsgi \

--http-scgi-temp-path=/var/temp/nginx/scgi \

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



配置成功



编译并安装

[root@itcast-01 nginx-1.7.7]# make

[root@itcast-01 nginx-1.7.7]# make install

### nginx配置文件

修改Nginx配置文件

[root@itcast-01 nginx-1.7.7]# vim /usr/local/nginx/conf/nginx.conf

添加server:

server {

listen 80;

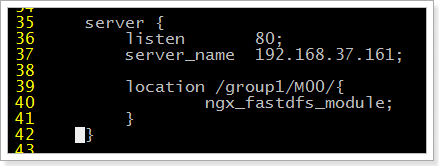
server\_name 192.168.37.161;

location /group1/M00/{

ngx\_fastdfs\_module;

}

}



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

## 启动

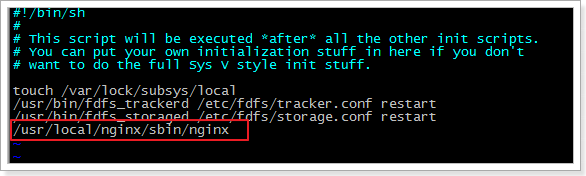
启动Nginx

[root@itcast-01 nginx-1.7.7]# /usr/local/nginx/sbin/nginx

设置开机自动启动。

[root@itcast-01 fdfs]# vim /etc/rc.d/rc.local

将运行命令行添加进文件：/usr/local/nginx/sbin/nginx



关闭防火墙

[root@itcast-01 nginx-1.7.7]# service iptables stop

设置防火墙不自动启动

[root@itcast-01 ~]# chkconfig iptables off

## 测试

通过测试方法上传文件，在宿主机使用浏览器访问，访问上传图片测试的文件：

访问storage：

http://192.168.37.161/group1/M00/00/00/wKgloVet2pmAKCe-AAFd-94XTw0759\_big.jpg



# 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