FastDFS学习文档

# 相关网址

入门上手：<https://www.cnblogs.com/chiangchou/p/fastdfs.html>

官网：<https://github.com/happyfish100>

# 用FastDFS一步步搭建文件管理系统

<https://www.cnblogs.com/chiangchou/p/fastdfs.html>

## 介绍

FastDFS开源地址：[https://github.com/happyfish100](https://github.com/happyfish100" \t "_blank)

参考：[分布式文件系统FastDFS设计原理](http://blog.chinaunix.net/uid-20196318-id-4058561.html" \t "_blank)

参考：[FastDFS分布式文件系统](http://www.cnblogs.com/Leo_wl/p/6731647.html)

个人封装的FastDFS Java API：[https://github.com/bojiangzhou/lyyzoo-fastdfs-java](https://github.com/bojiangzhou/lyyzoo-fastdfs-java" \t "_blank)

### 简介

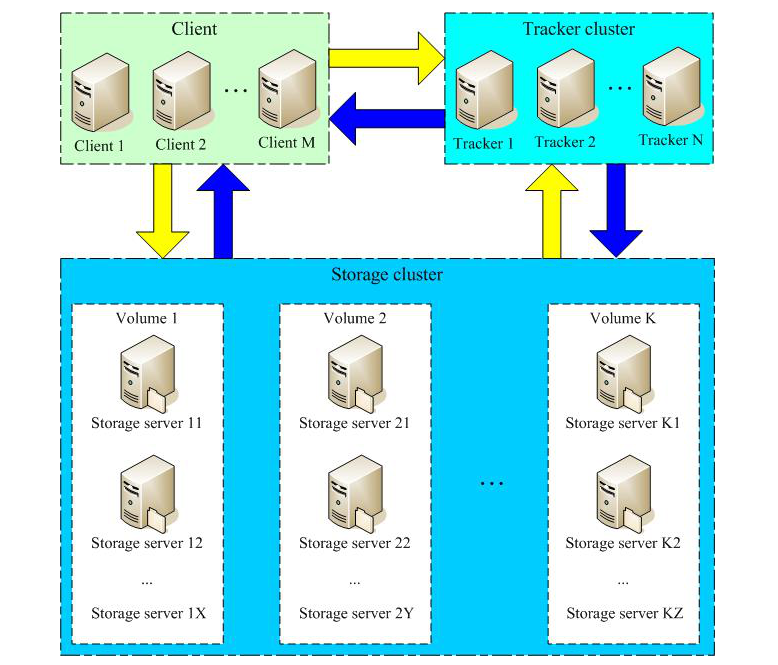
FastDFS 是一个开源的高性能分布式文件系统（DFS）。 它的主要功能包括：文件存储，文件同步和文件访问，以及高容量和负载平衡。主要解决了海量数据存储问题，特别适合以中小文件（建议范围：4KB < file\_size <500MB）为载体的在线服务。

FastDFS 系统有三个角色：跟踪服务器(Tracker Server)、存储服务器(Storage Server)和客户端(Client)。

**Tracker Server**：跟踪服务器，主要做调度工作，起到均衡的作用；负责管理所有的 storage server和 group，每个 storage 在启动后会连接 Tracker，告知自己所属 group 等信息，并保持周期性心跳。

**Storage Server**：存储服务器，主要提供存储和备份服务；以 group 为单位，每个 group 内可以有多台 storage server，数据互为备份。

**Client**：客户端，上传下载数据的服务器，也就是我们自己的项目所部署在的服务器。



### ****FastDFS的存储策略****

为了支持大容量，存储节点（服务器）采用了分卷（或分组）的组织方式。存储系统由一个或多个卷组成，卷与卷之间的文件是相互独立的，所有卷的文件容量累加就是整个存储系统中的文件容量。一个卷可以由一台或多台存储服务器组成，一个卷下的存储服务器中的文件都是相同的，卷中的多台存储服务器起到了冗余备份和负载均衡的作用。

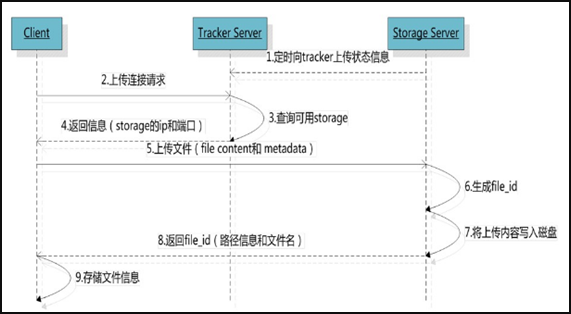
在卷中增加服务器时，同步已有的文件由系统自动完成，同步完成后，系统自动将新增服务器切换到线上提供服务。当存储空间不足或即将耗尽时，可以动态添加卷。只需要增加一台或多台服务器，并将它们配置为一个新的卷，这样就扩大了存储系统的容量。

### ****FastDFS的上传过程****

FastDFS向使用者提供基本文件访问接口，比如upload、download、append、delete等，以客户端库的方式提供给用户使用。

Storage Server会定期的向Tracker Server发送自己的存储信息。当Tracker Server Cluster中的Tracker Server不止一个时，各个Tracker之间的关系是对等的，所以客户端上传时可以选择任意一个Tracker。

当Tracker收到客户端上传文件的请求时，会为该文件分配一个可以存储文件的group，当选定了group后就要决定给客户端分配group中的哪一个storage server。当分配好storage server后，客户端向storage发送写文件请求，storage将会为文件分配一个数据存储目录。然后为文件分配一个fileid，最后根据以上的信息生成文件名存储文件。



### ****FastDFS的文件同步****

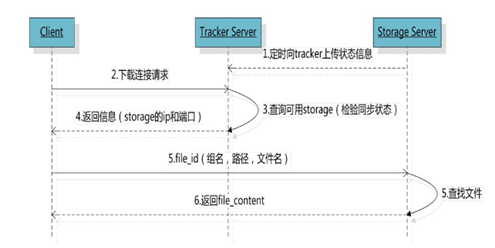
写文件时，客户端将文件写至group内一个storage server即认为写文件成功，storage server写完文件后，会由后台线程将文件同步至同group内其他的storage server。

每个storage写文件后，同时会写一份binlog，binlog里不包含文件数据，只包含文件名等元信息，这份binlog用于后台同步，storage会记录向group内其他storage同步的进度，以便重启后能接上次的进度继续同步；进度以时间戳的方式进行记录，所以最好能保证集群内所有server的时钟保持同步。

storage的同步进度会作为元数据的一部分汇报到tracker上，tracke在选择读storage的时候会以同步进度作为参考。

### ****5、FastDFS的文件下载****

客户端uploadfile成功后，会拿到一个storage生成的文件名，接下来客户端根据这个文件名即可访问到该文件。



跟upload file一样，在downloadfile时客户端可以选择任意tracker server。tracker发送download请求给某个tracker，必须带上文件名信息，tracke从文件名中解析出文件的group、大小、创建时间等信息，然后为该请求选择一个storage用来服务读请求。

## 安装FastDFS环境

### 前言

操作环境：CentOS7 X64，以下操作都是单机环境。

### 下载安装 libfastcommon

libfastcommon是从 FastDFS 和 FastDHT 中提取出来的公共 C 函数库，基础环境，安装即可 。

① 下载libfastcommon

# wget https://github.com/happyfish100/libfastcommon/archive/V1.0.7.tar.gz

② 解压

# tar -zxvf V1.0.7.tar.gz

# cd libfastcommon-1.0.7

可能需要

yum -y install zlib zlib-devel pcre pcre-devel gcc gcc-c++ openssl openssl-devel libevent libevent-devel perl unzip net-tools wget

③ 编译、安装

# ./make.sh

# ./make.sh install

④ libfastcommon.so 安装到了/usr/lib64/libfastcommon.so，但是FastDFS主程序设置的lib目录是/usr/local/lib，所以需要创建软链接。

ln -s /usr/lib64/libfastcommon.so /usr/local/lib/libfastcommon.so

ln -s /usr/lib64/libfastcommon.so /usr/lib/libfastcommon.so

ln -s /usr/lib64/libfdfsclient.so /usr/local/lib/libfdfsclient.so

ln -s /usr/lib64/libfdfsclient.so /usr/lib/libfdfsclient.so

### 下载安装FastDFS

① 下载FastDFS

wget https://github.com/happyfish100/fastdfs/archive/V5.05.tar.gz

② 解压

tar -zxvf V5.05.tar.gz

cd fastdfs-5.05

③ 编译、安装

./make.sh

./make.sh install

④ 默认安装方式安装后的相应文件与目录  
　　A、服务脚本：

/etc/init.d/fdfs\_storaged

/etc/init.d/fdfs\_tracker

　　B、配置文件（这三个是作者给的样例配置文件） :

/etc/fdfs/client.conf.sample

/etc/fdfs/storage.conf.sample

/etc/fdfs/tracker.conf.sample

　　C、命令工具在 /usr/bin/ 目录下：

[复制代码](javascript:void(0);)

fdfs\_appender\_test

fdfs\_appender\_test1

fdfs\_append\_file

fdfs\_crc32

fdfs\_delete\_file

fdfs\_download\_file

fdfs\_file\_info

fdfs\_monitor

fdfs\_storaged

fdfs\_test

fdfs\_test1

fdfs\_trackerd

fdfs\_upload\_appender

fdfs\_upload\_file

stop.sh

restart.sh

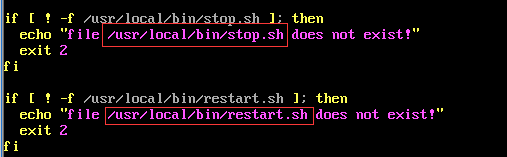
[复制代码](javascript:void(0);)

⑤ FastDFS 服务脚本设置的 bin 目录是 /usr/local/bin， 但实际命令安装在 /usr/bin/ 下。

　　两种方式：

　　》 一是修改FastDFS 服务脚本中相应的命令路径，也就是把 /etc/init.d/fdfs\_storaged 和 /etc/init.d/fdfs\_tracker 两个脚本中的 /usr/local/bin 修改成 /usr/bin。

 　　　　 vim fdfs\_trackerd  
　　　　使用查找替换命令进统一修改:%s+/usr/local/bin+/usr/bin  
　　　　 vim fdfs\_storaged  
　　　　使用查找替换命令进统一修改:%s+/usr/local/bin+/usr/bin



　　》 二是建立 /usr/bin 到 /usr/local/bin 的软链接，我是用这种方式。

ln -s /usr/bin/fdfs\_trackerd /usr/local/bin

ln -s /usr/bin/fdfs\_storaged /usr/local/bin

ln -s /usr/bin/stop.sh /usr/local/bin

ln -s /usr/bin/restart.sh /usr/local/bin

### 配置FastDFS跟踪器(Tracker)

配置文件详细说明参考：[FastDFS 配置文件详解](http://bbs.chinaunix.net/forum.php?mod=viewthread&tid=1941456&extra=page%3D1%26filter%3Ddigest%26digest%3D1)

① 进入 /etc/fdfs，复制 FastDFS 跟踪器样例配置文件 tracker.conf.sample，并重命名为 tracker.conf。

cd /etc/fdfs

cp tracker.conf.sample tracker.conf

vi tracker.conf

② 编辑tracker.conf ，标红的需要修改下，其它的默认即可。

# 配置文件是否不生效，false 为生效

disabled=false

# 提供服务的端口

port=22122

# Tracker 数据和日志目录地址(根目录必须存在,子目录会自动创建)

base\_path=/home/fastdfs/tracker/

# HTTP 服务端口

http.server\_port=80

③ 创建tracker基础数据目录，即base\_path对应的目录

# mkdir -p /home/fastdfs/tracker/

④ 防火墙中打开跟踪端口（默认的22122）

添加如下端口行：

firewall-cmd --zone=public --add-port=22122/tcp --permanent

重启防火墙：

firewall-cmd --reload

⑤ 启动Tracker

初次成功启动，会在 /home/fdfsdfs/tracker/ (配置的base\_path)下创建 data、logs 两个目录。

可以用这种方式启动

# /etc/init.d/fdfs\_trackerd start

也可以用这种方式启动，前提是上面创建了软链接，后面都用这种方式

# service fdfs\_trackerd start

查看 FastDFS Tracker 是否已成功启动 ，22122端口正在被监听，则算是Tracker服务安装成功。

# netstat -unltp|grep fdfs

https://images2017.cnblogs.com/blog/856154/201710/856154-20171011121344184-1089101646.png

关闭Tracker命令：

# service fdfs\_trackerd stop

⑥ 设置Tracker开机启动

chkconfig fdfs\_trackerd on

⑦ tracker server 目录及文件结构

Tracker服务启动成功后，会在base\_path下创建data、logs两个目录。目录结构如下：

${base\_path}

|\_\_data

| |\_\_storage\_groups.dat：存储分组信息

| |\_\_storage\_servers.dat：存储服务器列表

|\_\_logs

| |\_\_trackerd.log： tracker server 日志文件

### 配置 FastDFS 存储 (Storage)到 这里

① 进入 /etc/fdfs 目录，复制 FastDFS 存储器样例配置文件 storage.conf.sample，并重命名为 storage.conf

cd /etc/fdfs

cp storage.conf.sample storage.conf  
vi storage.conf

② 编辑storage.conf

标红的需要修改，其它的默认即可。

# 配置文件是否不生效，false 为生效

disabled=false

# 指定此 storage server 所在 组(卷)

group\_name=group1

# storage server 服务端口

port=23000

# 心跳间隔时间，单位为秒 (这里是指主动向 tracker server 发送心跳)

heart\_beat\_interval=30

# Storage 数据和日志目录地址(根目录必须存在，子目录会自动生成)

base\_path=/home/fastdfs/storage

# 存放文件时 storage server 支持多个路径。这里配置存放文件的基路径数目，通常只配一个目录。

store\_path\_count=1

# 逐一配置 store\_path\_count 个路径，索引号基于 0。

# 如果不配置 store\_path0，那它就和 base\_path 对应的路径一样。

store\_path0=/home/fastdfs/file

# FastDFS 存储文件时，采用了两级目录。这里配置存放文件的目录个数。

# 如果本参数只为 N（如： 256），那么 storage server 在初次运行时，会在 store\_path 下自动创建 N \* N 个存放文件的子目录。

subdir\_count\_per\_path=256

# tracker\_server 的列表 ，会主动连接 tracker\_server

# 有多个 tracker server 时，每个 tracker server 写一行

tracker\_server=192.168.202.129:22122  
  
# 允许系统同步的时间段 (默认是全天) 。一般用于避免高峰同步产生一些问题而设定。  
sync\_start\_time=00:00  
sync\_end\_time=23:59

# 访问端口  
http.server\_port=80

③ 创建Storage基础数据目录，对应base\_path目录

mkdir -p /home/fastdfs/storage

# 这是配置的store\_path0路径

mkdir -p /home/fastdfs/file

④ 防火墙中打开存储器端口（默认的 23000）

添加如下端口行：

firewall-cmd --zone=public --add-port=23000/tcp --permanent

重启防火墙：

firewall-cmd --reload

https://images2017.cnblogs.com/blog/856154/201710/856154-20171011133233777-1903096242.png

⑤ 启动 Storage

启动Storage前确保Tracker是启动的。初次启动成功，会在 /ljzsg/fastdfs/storage 目录下创建 data、 logs 两个目录。

可以用这种方式启动

# /etc/init.d/fdfs\_storaged start

也可以用这种方式，后面都用这种

# service fdfs\_storaged start

查看 Storage 是否成功启动，23000 端口正在被监听，就算 Storage 启动成功。

# netstat -unltp|grep fdfs

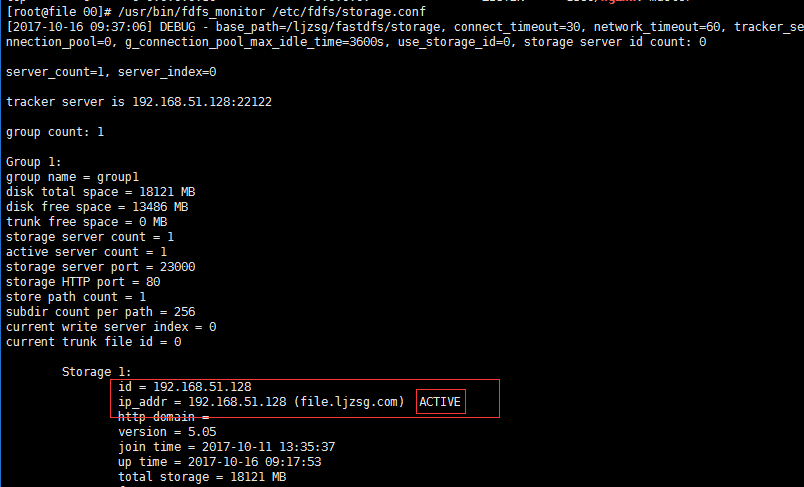
https://images2017.cnblogs.com/blog/856154/201710/856154-20171011134723387-604894314.png

关闭Storage命令：

# service fdfs\_storaged stop

查看Storage和Tracker是否在通信：

/usr/bin/fdfs\_monitor /etc/fdfs/storage.conf



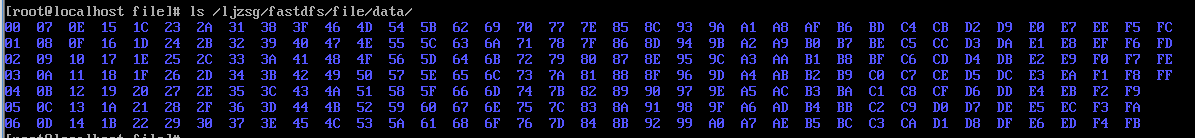
⑥ 设置 Storage 开机启动

# chkconfig fdfs\_storaged on

⑦ Storage 目录

同 Tracker，Storage 启动成功后，在base\_path 下创建了data、logs目录，记录着 Storage Server 的信息。

在 store\_path0 目录下，创建了N\*N个子目录：



### 文件上传测试

① 修改 Tracker 服务器中的客户端配置文件

cd /etc/fdfs

cp client.conf.sample client.conf

vi client.conf

修改如下配置即可，其它默认。

# Client 的数据和日志目录

base\_path=/home/fastdfs/client

# Tracker端口

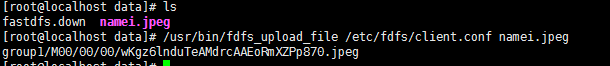
tracker\_server=192.168.202.129:22122

② 上传测试

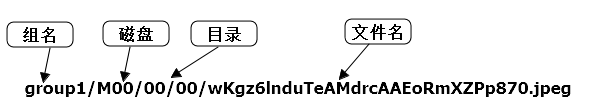
 在linux内部执行如下命令上传 namei.jpeg 图片

/usr/bin/fdfs\_upload\_file /etc/fdfs/client.conf namei.jpeg

上传成功后返回文件ID号：group1/M00/00/00/wKgz6lnduTeAMdrcAAEoRmXZPp870.jpeg



返回的文件ID由group、存储目录、两级子目录、fileid、文件后缀名（由客户端指定，主要用于区分文件类型）拼接而成。



## 安装Nginx

上面将文件上传成功了，但我们无法下载。因此安装Nginx作为服务器以支持Http方式访问文件。同时，后面安装FastDFS的Nginx模块也需要Nginx环境。

Nginx只需要安装到StorageServer所在的服务器即可，用于访问文件。我这里由于是单机，TrackerServer和StorageServer在一台服务器上。

### 安装Nginx

1、安装nginx所需环境

① gcc 安装

# yum install gcc-c++

② PCRE pcre-devel 安装

# yum install -y pcre pcre-devel

③ zlib 安装

# yum install -y zlib zlib-devel

④ OpenSSL 安装

# yum install -y openssl openssl-devel

2、安装Nginx

① 下载nginx

# wget -c https://nginx.org/download/nginx-1.12.1.tar.gz

② 解压

# tar -zxvf nginx-1.12.1.tar.gz

# cd nginx-1.12.1

③ 使用默认配置

# ./configure

④ 编译、安装

# make

# make install

⑤ 启动nginx

# cd /usr/local/nginx/sbin/

# ./nginx

其它命令

# ./nginx -s stop

# ./nginx -s quit

# ./nginx -s reload

⑥ 设置开机启动

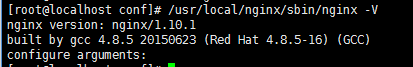
# vim /etc/rc.local

添加一行：

/usr/local/nginx/sbin/nginx  
  
# 设置执行权限  
# chmod 755 rc.local

⑦ 查看nginx的版本及模块

/usr/local/nginx/sbin/nginx -V



⑧ 防火墙中打开Nginx端口（默认的 80）

添加后就能在本机使用80端口访问了。

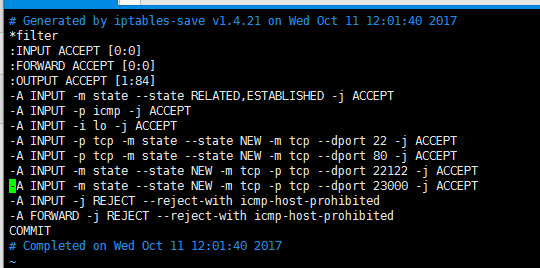
# vim /etc/sysconfig/iptables

添加如下端口行：

-A INPUT -m state --state NEW -m tcp -p tcp --dport 80 -j ACCEPT

重启防火墙：

# service iptables restart



### 访问文件

简单的测试访问文件

① 修改nginx.conf

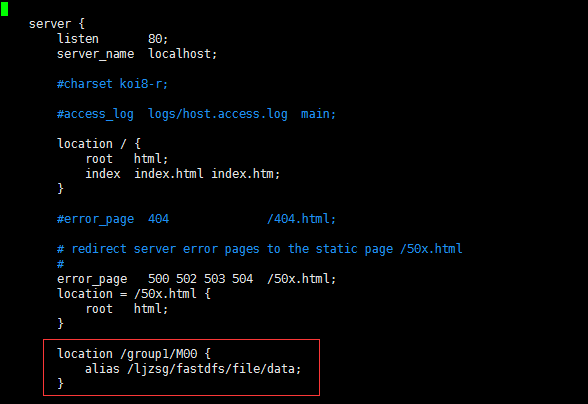
# vim /usr/local/nginx/conf/nginx.conf

添加如下行，将 /group1/M00 映射到 /ljzsg/fastdfs/file/data

location /group1/M00 {

alias /ljzsg/fastdfs/file/data;

}  
  
# 重启nginx  
# /usr/local/nginx/sbin/nginx -s reload



location /group1/M00/ {

alias /home/fastdfs/file/data/;

allow all;

}

location /pic/ {

alias /home/;

allow all;

}

② 在浏览器访问之前上传的图片、成功。

[http://192.168.202.129](http://192.168.202.129/)/group1/M00/00/00/wKgz6lnduTeAMdrcAAEoRmXZPp870.jpeg

## FastDFS 配置 Nginx 模块

### 安装配置Nginx模块

① fastdfs-nginx-module 模块说明

　　FastDFS 通过 Tracker 服务器，将文件放在 Storage 服务器存储， 但是同组存储服务器之间需要进行文件复制， 有同步延迟的问题。

　　假设 Tracker 服务器将文件上传到了 192.168.51.128，上传成功后文件 ID已经返回给客户端。

　　此时 FastDFS 存储集群机制会将这个文件同步到同组存储 192.168.51.129，在文件还没有复制完成的情况下，客户端如果用这个文件 ID 在 192.168.51.129 上取文件,就会出现文件无法访问的错误。

　　而 fastdfs-nginx-module 可以重定向文件链接到源服务器取文件，避免客户端由于复制延迟导致的文件无法访问错误。

② 下载 fastdfs-nginx-module、解压

[复制代码](javascript:void(0);)

# 这里为啥这么长一串呢，因为最新版的master与当前nginx有些版本问题。

# wget https://github.com/happyfish100/fastdfs-nginx-module/archive/5e5f3566bbfa57418b5506aaefbe107a42c9fcb1.zip

# 解压

# unzip 5e5f3566bbfa57418b5506aaefbe107a42c9fcb1.zip

# 重命名

# mv fastdfs-nginx-module-5e5f3566bbfa57418b5506aaefbe107a42c9fcb1 fastdfs-nginx-module-master

[复制代码](javascript:void(0);)

③ 配置Nginx

在nginx中添加模块

[复制代码](javascript:void(0);)

# 先停掉nginx服务  
# /usr/local/nginx/sbin/nginx -s stop  
  
进入解压包目录

# cd /softpackages/nginx-1.12.1/

# 添加模块

# ./configure --add-module=../fastdfs-nginx-module-master/src

重新编译、安装

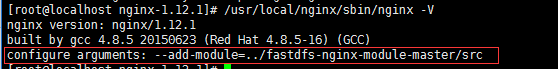
# make && make install

[复制代码](javascript:void(0);)

 ④ 查看Nginx的模块

# /usr/local/nginx/sbin/nginx -V

有下面这个就说明添加模块成功



⑤ 复制 fastdfs-nginx-module 源码中的配置文件到/etc/fdfs 目录， 并修改

# cd /softpackages/fastdfs-nginx-module-master/src  
  
# cp mod\_fastdfs.conf /etc/fdfs/

修改如下配置，其它默认

# 连接超时时间  
connect\_timeout=10

# Tracker Server

tracker\_server=192.168.202.129:22122

# StorageServer 默认端口

storage\_server\_port=23000

# 如果文件ID的uri中包含/group\*\*，则要设置为true

url\_have\_group\_name = true

# Storage 配置的store\_path0路径，必须和storage.conf中的一致

store\_path0=/home/fastdfs/file

⑥ 复制 FastDFS 的部分配置文件到/etc/fdfs 目录

# cd /softpackages/fastdfs-5.05/conf/

# cp anti-steal.jpg http.conf mime.types /etc/fdfs/

 ⑦ 配置nginx，修改nginx.conf

# vim /usr/local/nginx/conf/nginx.conf

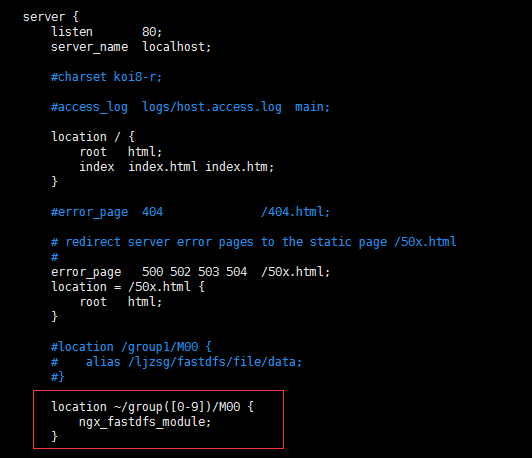
修改配置，其它的默认

在80端口下添加fastdfs-nginx模块

location ~/group([0-9])/M00 {

ngx\_fastdfs\_module;

}



注意：

　　listen 80 端口值是要与 /etc/fdfs/storage.conf 中的 http.server\_port=80 (前面改成80了)相对应。如果改成其它端口，则需要统一，同时在防火墙中打开该端口。

　　location 的配置，如果有多个group则配置location ~/group([0-9])/M00 ，没有则不用配group。

⑧ 在/home/fastdfs/file 文件存储目录下创建软连接，将其链接到实际存放数据的目录，这一步可以省略。

# ln -s /home/fastdfs/file/data/ /home/fastdfs/file/data/M00

⑨ 启动nginx

# /usr/local/nginx/sbin/nginx

打印处如下就算配置成功

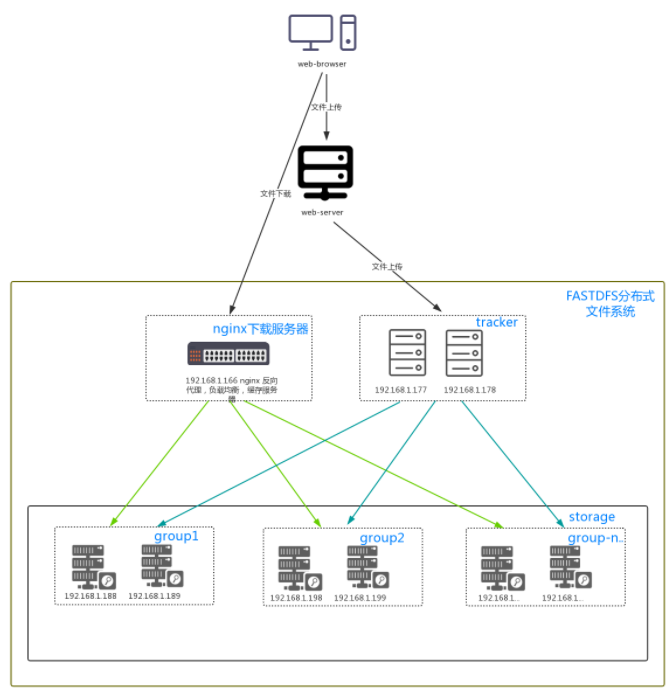
https://images2017.cnblogs.com/blog/856154/201710/856154-20171011230509512-654301113.png

⑩ 在地址栏访问。

能下载文件就算安装成功。注意和第三点中直接使用nginx路由访问不同的是，这里配置 fastdfs-nginx-module 模块，可以重定向文件链接到源服务器取文件。

http://192.168.202.129 /group1/M00/00/00/wKgz6lnduTeAMdrcAAEoRmXZPp870.jpeg

最终部署结构图(盗的图)：可以按照下面的结构搭建环境。



## Java客户端

前面文件系统平台搭建好了，现在就要写客户端代码在系统中实现上传下载，这里只是简单的测试代码。

### 首先需要搭建 FastDFS 客户端Java开发环境

① 项目中使用maven进行依赖管理，可以在pom.xml中引入如下依赖即可：

<dependency>

<groupId>net.oschina.zcx7878</groupId>

<artifactId>fastdfs-client-java</artifactId>

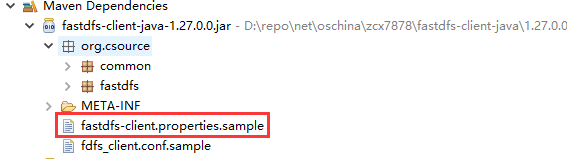
<version>1.27.0.0</version>

</dependency>

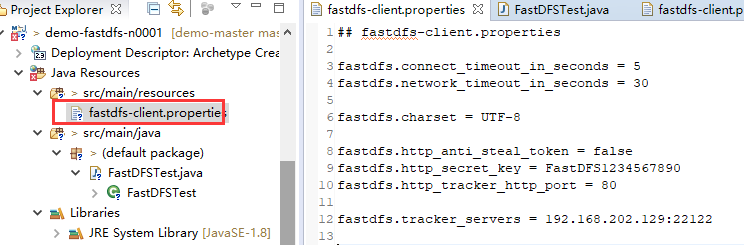
其它的方式，参考官方文档：https://github.com/happyfish100/fastdfs-client-java

② 引入配置文件

可直接复制包下的 fastdfs-client.properties.sample 或者 fdfs\_client.conf.sample，到你的项目中，去掉.sample。



我这里直接复制 fastdfs-client.properties.sample 中的配置到项目配置文件 fastdfs-client.properties 中，修改tracker\_servers。只需要加载这个配置文件即可



### 客户端测试

**import** java.io.BufferedOutputStream;

**import** java.io.File;

**import** java.io.FileOutputStream;

**import** org.csource.fastdfs.ClientGlobal;

**import** org.csource.fastdfs.ServerInfo;

**import** org.csource.fastdfs.StorageClient;

**import** org.csource.fastdfs.StorageServer;

**import** org.csource.fastdfs.TrackerClient;

**import** org.csource.fastdfs.TrackerServer;

**import** org.junit.Test;

**public** **class** FastDFSTest {

@SuppressWarnings("resource")

@Test

**public** **void** testUpload() **throws** Exception {

//可参看https://github.com/happyfish100/fastdfs-client-java/blob/master/src/test/java/org/csource/fastdfs/TestClient.java

// 1、把FastDFS提供的jar包添加到工程中

// 2、初始化全局配置。加载一个配置文件。

ClientGlobal.*initByProperties*("fastdfs-client.properties");

// 3、创建一个TrackerClient对象。

TrackerClient trackerClient = **new** TrackerClient();

// 4、创建一个TrackerServer对象。

TrackerServer trackerServer = trackerClient.getConnection();

// 5、声明一个StorageServer对象，null。

StorageServer storageServer = **null**;

// 6、获得StorageClient对象。

StorageClient storageClient = **new** StorageClient(trackerServer, storageServer);

// 7、直接调用StorageClient对象方法上传文件即可。

String[] results = storageClient.upload\_file("D:/Study/7M.jpg", "jpg", **null**);

String group\_name;

String remote\_filename;

ServerInfo[] servers;

**byte**[] file\_buff;

**if** (results == **null**) {

System.***err***.println("upload file fail, error code: " + storageClient.getErrorCode());

**return**;

} **else** {

group\_name = results[0];//group1 组名，卷名

remote\_filename = results[1];//M00/00/00/wKjKgVyu9sqAZC5\_AHEGV7OaW-0895.jpg 远程文件路径

servers = trackerClient.getFetchStorages(trackerServer, group\_name, remote\_filename);

**if** (servers == **null**) {

System.***err***.println("get storage servers fail, error code: " + trackerClient.getErrorCode());

} **else** {

file\_buff = storageClient.download\_file(group\_name, remote\_filename);//下载

//将文件保存到本地文件

**if** (file\_buff != **null**) {

System.***out***.println("file length:" + file\_buff.length);

BufferedOutputStream bos = **null**;

FileOutputStream fos = **null**;

File file = **new** File("D:/Study/download.jpg");

fos = **new** FileOutputStream(file);

bos = **new** BufferedOutputStream(fos);

bos.write(file\_buff);

}

}

}

}

}

## 权限控制

前面使用nginx支持http方式访问文件，但所有人都能直接访问这个文件服务器了，所以做一下权限控制。

FastDFS的权限控制是在服务端开启token验证，客户端根据文件名、当前unix时间戳、秘钥获取token，在地址中带上token参数即可通过http方式访问文件。

① 服务端开启token验证

修改http.conf

vi /etc/fdfs/http.conf

设置为true表示开启token验证

http.anti\_steal.check\_token=true  
  
设置token失效的时间单位为秒(s)  
http.anti\_steal.token\_ttl=1800

密钥，跟客户端配置文件的fastdfs.http\_secret\_key保持一致

http.anti\_steal.secret\_key=FASTDFS1234567890

如果token检查失败，返回的页面

http.anti\_steal.token\_check\_fail=/home/fastdfs/page/403.html

[复制代码](javascript:void(0);)

记得重启服务。

nginx -s reload

② 配置客户端

客户端只需要设置如下两个参数即可，两边的密钥保持一致。

# token 防盗链功能

fastdfs.http\_anti\_steal\_token=true

# 密钥

fastdfs.http\_secret\_key=FASTDFS1234567890

③ 客户端生成token

访问文件需要带上生成的token以及unix时间戳，所以返回的token是token和时间戳的拼接。

之后，将token拼接在地址后即可访问：http://192.168.202.129/group1/M00/00/00/wKjKgVyvJ9WAXiJ7AHEGV7OaW-0795.jpg?token=078d370098b03e9020b82c829c205e1f&ts=1508141521

[复制代码](javascript:void(0);)

1 /\*\*

2 \* 获取访问服务器的token，拼接到地址后面

3 \*

4 \* @param filepath 文件路径 group1/M00/00/00/wKgzgFnkTPyAIAUGAAEoRmXZPp876.jpeg

5 \* @param httpSecretKey 密钥

6 \* @return 返回token，如： token=078d370098b03e9020b82c829c205e1f&ts=1508141521

7 \*/

8 public static String getToken(String filepath, String httpSecretKey){

9 // unix seconds

10 int ts = (int) Instant.now().getEpochSecond();

11 // token

12 String token = "null";

13 try {

14 token = ProtoCommon.getToken(getFilename(filepath), ts, httpSecretKey);

15 } catch (UnsupportedEncodingException e) {

16 e.printStackTrace();

17 } catch (NoSuchAlgorithmException e) {

18 e.printStackTrace();

19 } catch (MyException e) {

20 e.printStackTrace();

21 }

22

23 StringBuilder sb = new StringBuilder();

24 sb.append("token=").append(token);

25 sb.append("&ts=").append(ts);

26

27 return sb.toString();

28 }

[复制代码](javascript:void(0);)

④ 注意事项

如果生成的token验证无法通过，请进行如下两项检查：  
　　A. 确认调用token生成函数(ProtoCommon.getToken)，传递的文件ID中没有包含group name。传递的文件ID格式形如：M00/00/00/wKgzgFnkTPyAIAUGAAEoRmXZPp876.jpeg

　　B. 确认服务器时间基本是一致的，注意服务器时间不能相差太多，不要相差到分钟级别。

⑤ 对比下发现，如果系统文件隐私性较高，可以直接通过fastdfs-client提供的API去访问即可，不用再配置Nginx走http访问。配置Nginx的主要目的是为了快速访问服务器的文件(如图片)，如果还要加权限验证，则需要客户端生成token，其实已经没有多大意义。

关键是，这里我没找到FastDFS如何对部分资源加token验证，部分开放。有知道的还请留言。

# 手把手教你搭建FastDFS集群

<https://blog.csdn.net/u012453843/article/details/68957209>

<https://blog.csdn.net/u012453843/article/details/69055570>

<https://blog.csdn.net/u012453843/article/details/69172423>

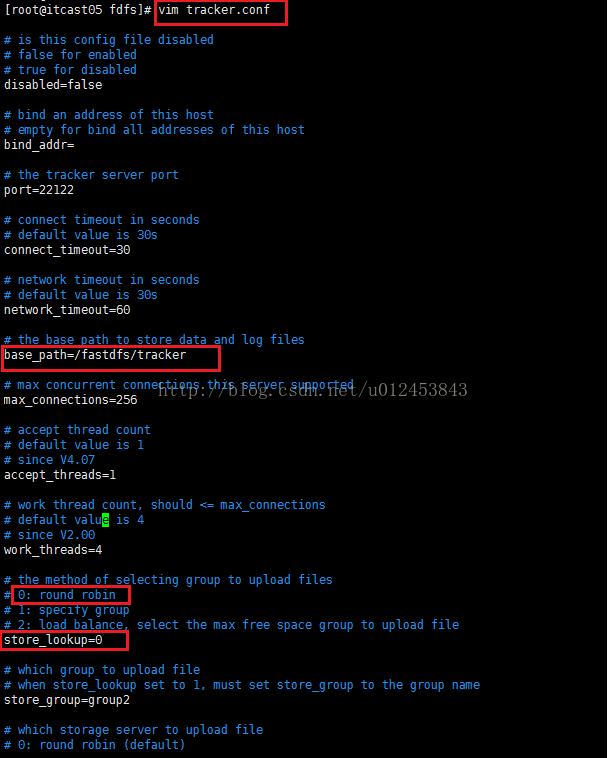
集群思路：

2台（tracker）4台（storage）2台（nginx），其中2台为一个group，共2个group

Tracker、storage都安装好FastDFS

Tracker配置：

编辑tracker.conf文件，需要修改的地方有两处，第一处：base\_path，将默认的路径修改为/fastdfs/tracker。第二处：store\_lookup，该值默认是2（即负载均衡策略），现在把它修改为0（即轮询策略，修改成这样方便一会儿我们进行测试，当然，最终还是要改回到2的。如果值为1的话表明要始终向某个group进行上传下载操作，这时下图中的"store\_group=group2"才会起作用，如果值是0或2，则"store\_group=group2"不起作用）。



Storage配置

编辑storage.conf文件，对于group1的192.168.156.7、192.168.156.8这两台设备需要修改的配置如下。

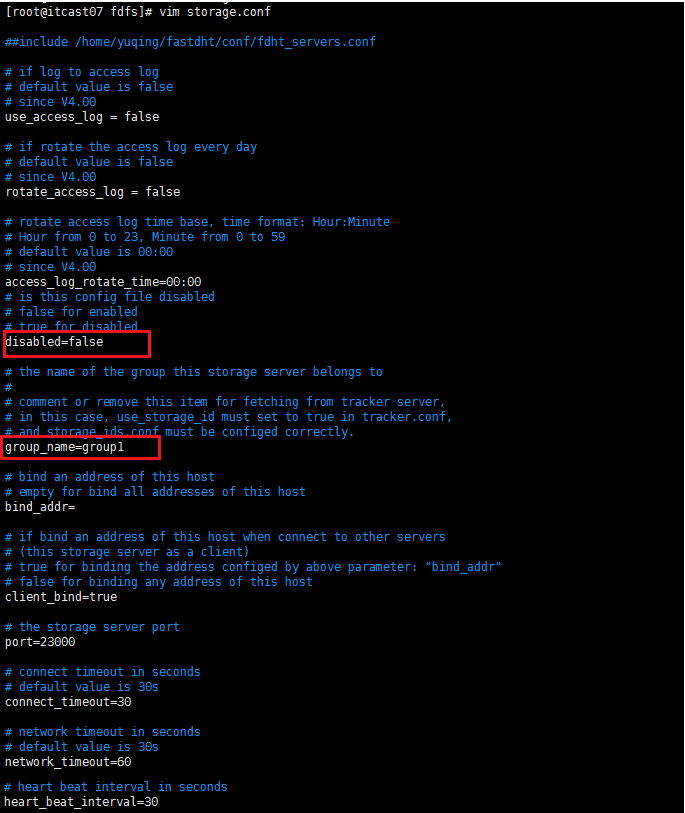
       base\_path=/fastdfs/storage

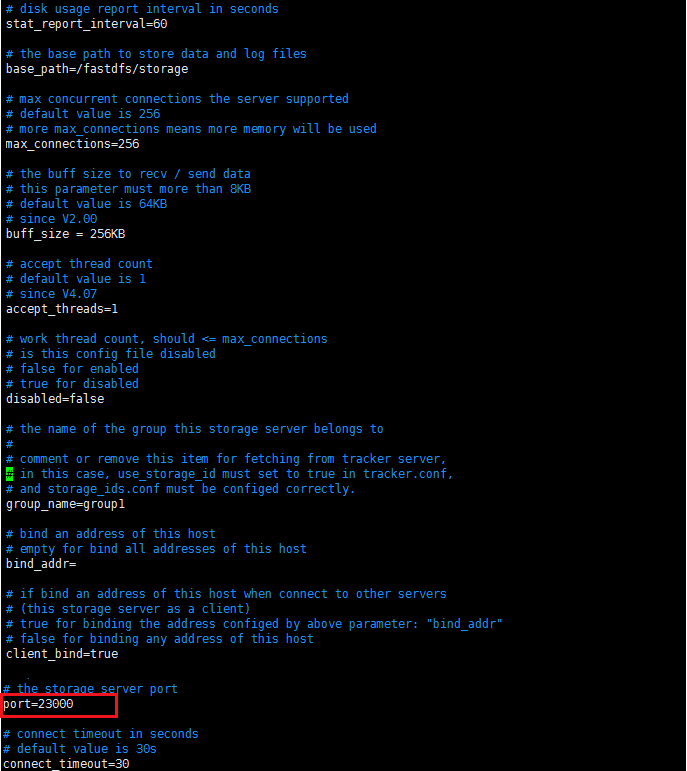
       store\_path0=/fastdfs/storage

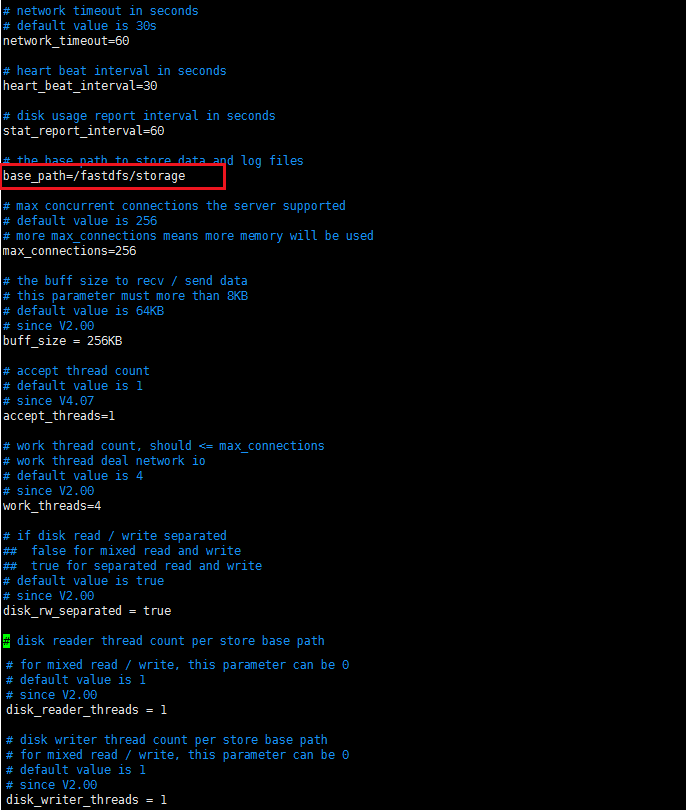
       tracker\_server=192.168.156.5:22122

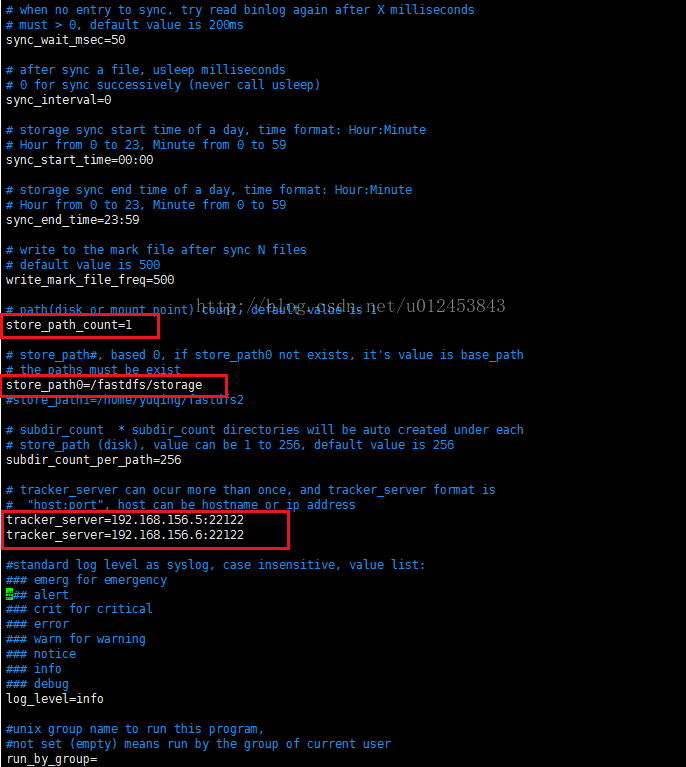
       tracker\_server=192.168.156.6:22122

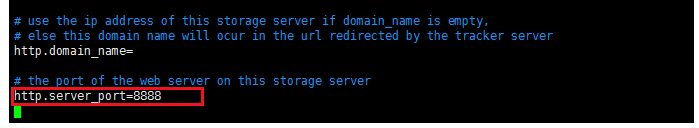
       修改完后，如下图所示。



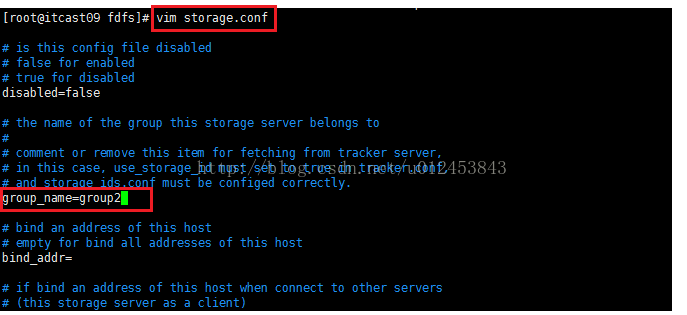








       配置完了192.168.156.7，接下来我们把storage.conf文件复制到192.168.156.8、192.168.156.9、192.168.156.10这三台设备上。其中192.168.156.8这台设备与192.168.156.7同属于group1，因此把配置文件放到它的/etc/fdfs/目录后不用做任何修改。但是192.168.156.9和192.168.156.10这两台设备需要修改一下，修改也非常简单，只需要把group\_name由group1改为group2就可以了，如下图所示。



     由于四个配置文件的base\_path=/fastdfs/storage和store\_path0=/fastdfs/storage都配置成了/fastdfs/storage，但是目前我们这四台虚拟机还未创建过该目录，因此我们需要为这四台虚拟机都创建一下该目录,命令：mkdir -p /fastdfs/storage，如下图所示。

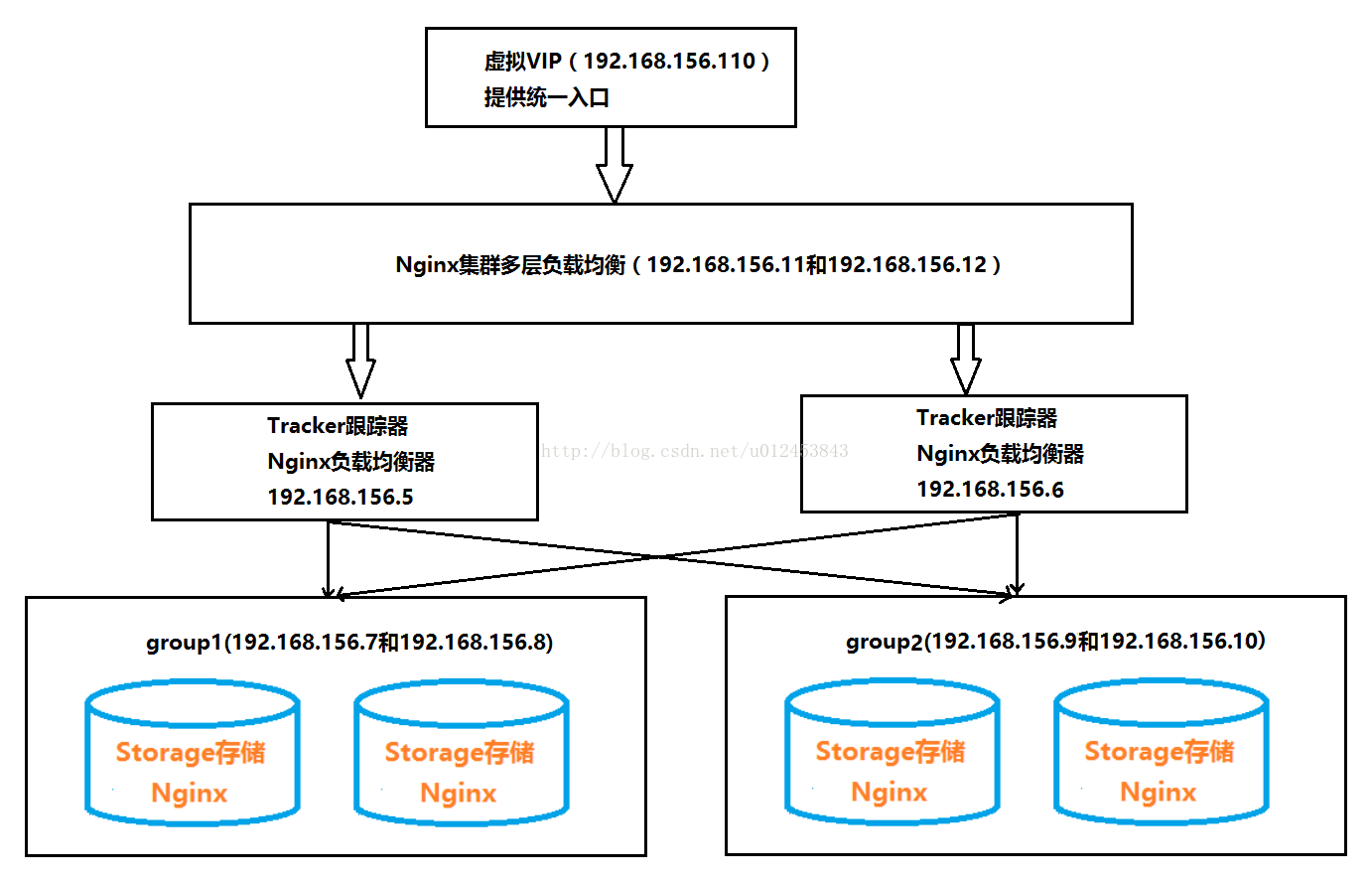


      启动storage之前，我们需要先对192.168.156.7、192.168.156.8、192.168.156.9、192.168.156.10这四台虚拟机的防火墙做下配置，添加端口2300，

      配置完防火墙之后，我们到storage所在的四台虚拟机的/fastdfs/storage目录下，使用命令：/etc/init.d/fdfs\_storaged start进行启动，然后使用命令：tailf logs/storaged.log来查看启动信息，如下图所示（192.168.156.7这台虚拟机的操作图）。可以看到，192.168.156.7这台设备成功与两个tracker设备连接了，其中选举了192.168.156.6作为tracker集群的leader。192.168.156.7和192.168.156.8这两台虚拟机同属一个分组(group1)，因此可以从下图的信息中看到成功与192.168.156.8连接的信息。

当我们所有的tracker和storage节点都启动成功之后，我们可以在任意的一个存储节点上查看存储集群的信息，命令：/usr/bin/fdfs\_monitor /etc/fdfs/storage.conf，可以看到如下图所示的集群信息。可以看到tracker Server有两个，当前提供服务的是192.168.156.5，group的数量是2，第一组的IP有192.168.156.7和192.168.156.8，第二组的IP有192.168.156.9和192.168.156.10，与我们规划的集群完全一致。

       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。