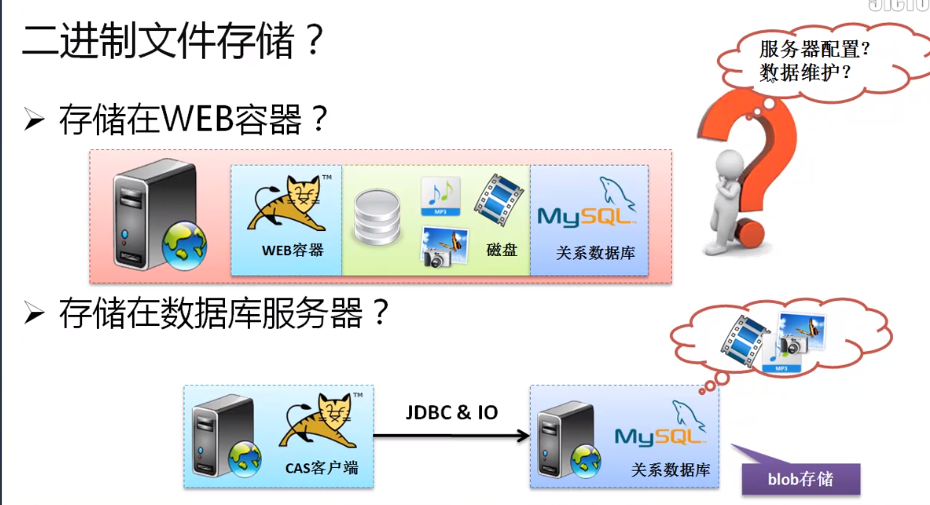
# FastDFS

## 认识FastDFS

系统中需要存放大量的图片或者是视频资源，fastDFS就是最好的选择。

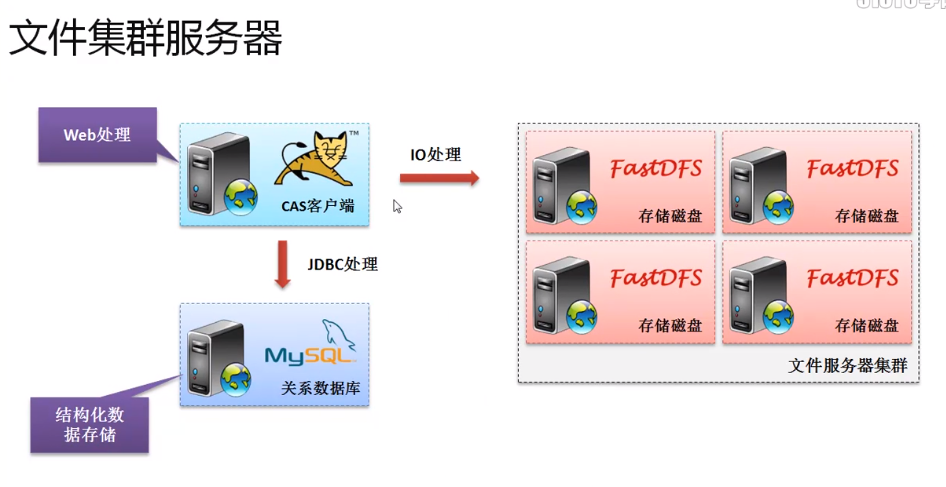
### FastDFS简介



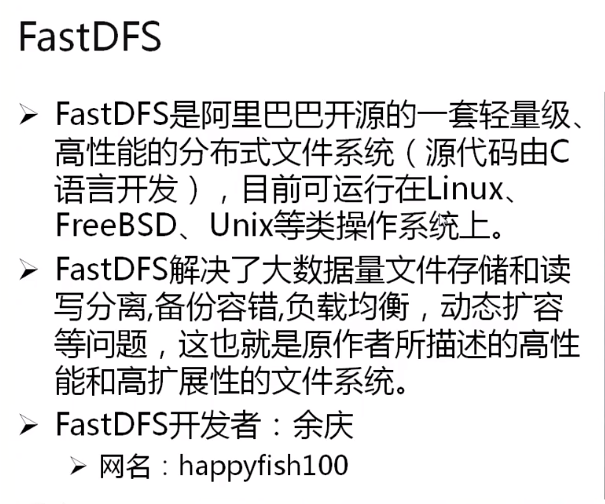


文件集群应该满足于如下的要求：

1. 该文件集群可以无限制的扩大；
2. 该文件集群里面需要对数据进行备份；
3. 灾难恢复；

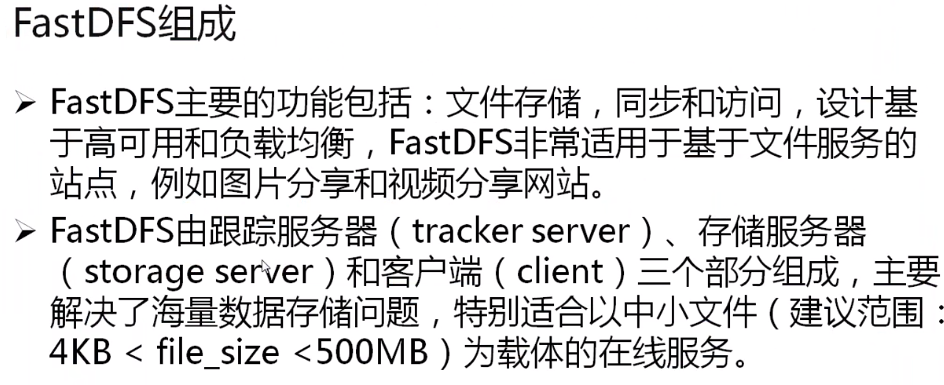


最明显的特征是文件服务器的集群数据量很大，但是真正参与的部分很少。这类的集群可以是用大磁盘，低CPU的服务器。



FastDFS是现在大部分企业公司所采用的文件系统的结构，可以方便的进行容量的扩充以及灾难恢复。

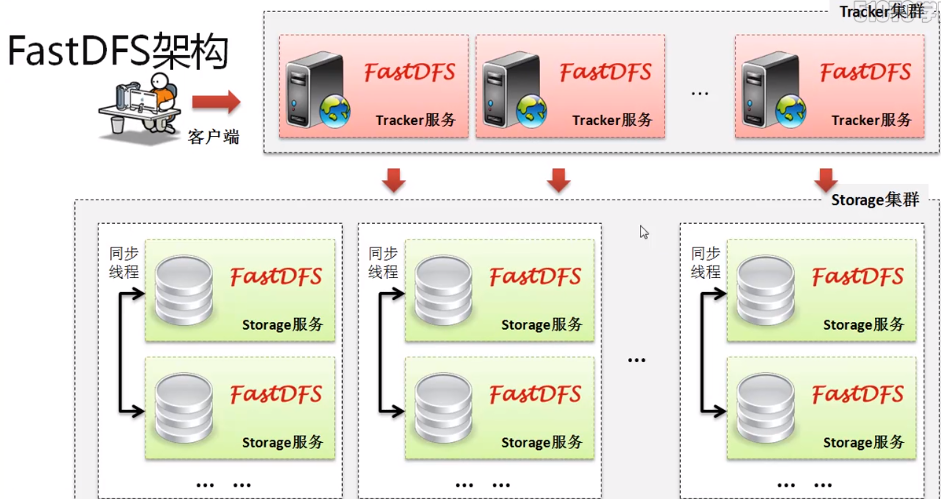
### FastDFS设计原理



FastDFS核心组成部分：

1. Tracker:负责保存所有的storage的元信息以及信息查找。
2. Storage:真正的数据存储

用户不会参与到Storage存储上，只是找到了Tracker.



所有的存储数据节点都是用组的概念来描述，一组的存储里面可能会包含有多个存储的主机。为进行组的区分，需要依靠Tracker完成。

在每一个存储组的概念里面，都具备良好的数据同步效果，也就是说一组中多台主机的数据都是相同的，如果一组中某台主机出现问题后，那么该主机变更后，原始存储的数据将自动复制（同步过来），并且自动提供相应的服务。同步同时解决了数据备份问题。

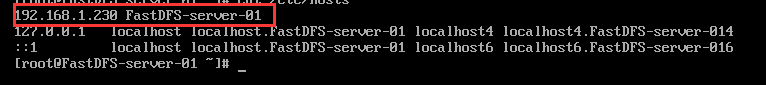
## 搭建FastDFS伪分布式环境

### 配置FastDFS

要配置FastDFS，需要先安装支持包libfastcomm的支持库文件。

官方位置： https://github.com/happyfish100/libfastcommon

1、【fastDFS-server-01】修改Host文件



1. 【fastDFS-server-01】为了能在github上的libfastcommon开发包，所以需要先生成本机的SSH配置。

(1)重新设置一个新的SSHKEY：

(2)设置本机的免登陆配置：



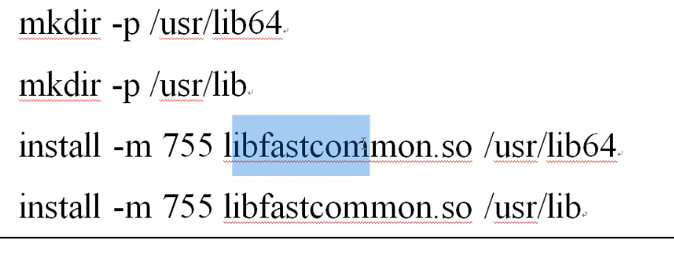
1. 取得SSH-Key的内容：

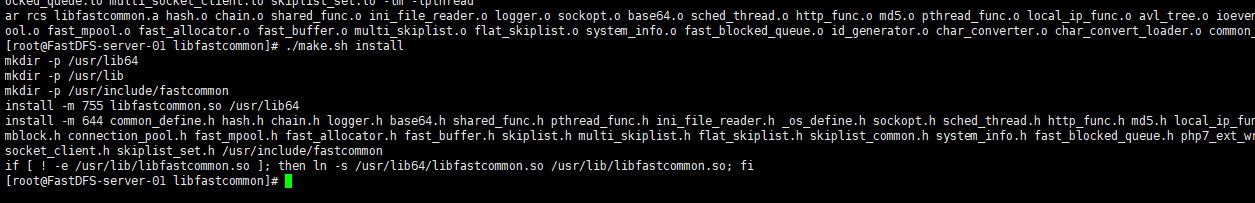


1. 登录GITHUB上，将该SSH\_KEY的信息配置（https方式可以跳过）
2. 【fastDFS-server-01】使用git克隆出此代码：
3. 所有的源代码都应该保存在“/usr/local/src”目录下(克隆)：

git clone https://github.com/happyfish100/libfastcommon.git

1. 【fastDFS-server-01】进行libfastcommon组件进行编译：
2. 进行项目的编译：./make.sh
3. 进行项目安装：./make.sh install





1. 【fastDFS-server-01】为了编译方便，做一些软链接到指定的目录上：

ln -sv /usr/include/fastcommon /usr/local/include/fastcommon

ln -sv /usr/include/fastdfs /usr/local/include/fastdfs

ln -sv /usr/lib/libfastcommon.so /usr/local/lib/libfastcommon.so

ln -sv /usr/lib/libfastclient.so /usr/local/lib/libfastclient.so

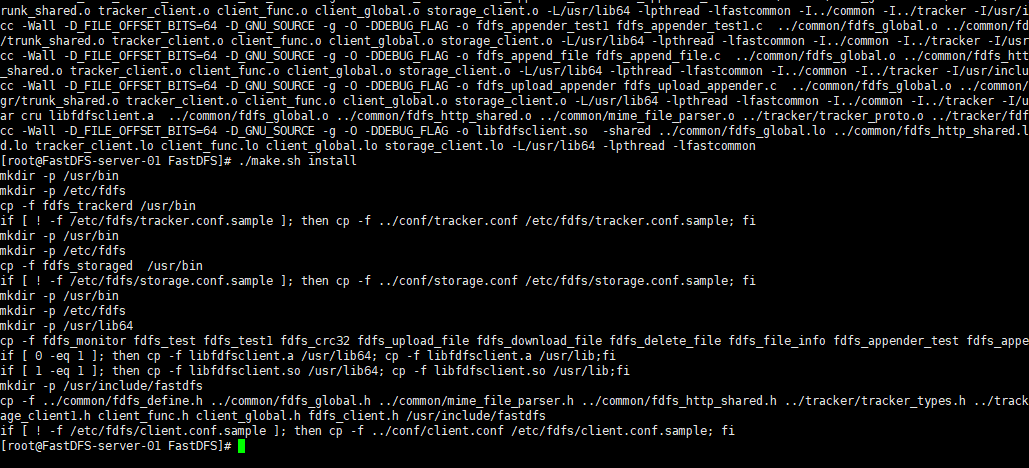
ln -sv /usr/lib64/libfdfsclient.so /usr/local/lib/libfdfsclient.so

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

1. 【fastDFS-server-01】将FastDFS源代码上床到linux服务器，随后进行解压缩。

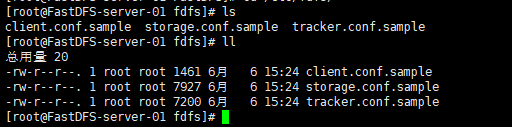
tar zxvf FastDFS\_v5.08.tar.gz -C /usr/local/src/

1. 【fastDFS-server-01】进行FastDFS源代码编译：
2. 进行源代码的编译处理：./make.sh
3. 进行安装处理：./make.sh install



要是没有编译报错，基本没问题 。

1. 【fastDFS-server-01】查看“etc/fdfs”目录下是否有以下三个文件：



### 配置FastDFS

FastDFS的配置实际上就只有两个组成部分：tracker、storage

1. 【FastDFS-server-01】编译tracker的配置文件：

(1)通过给出的模板文件复制出一个新的文件：

cp /etc/fdfs/tracker.conf.sample /etc/fdfs/tracker.conf

1. 编辑tracker.conf配置文件：



1. 建立一个tracker的信息保存目录：



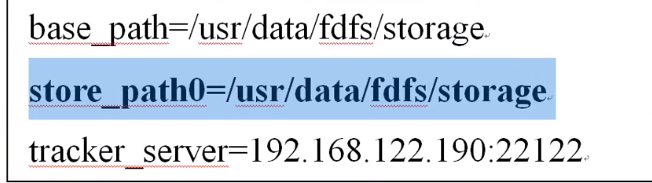
1. 【FastDFS-server-01】配置storage配置文件
2. 通过模板文件拷贝出storage.conf文件：

cp /etc/fdfs/storage.conf.sample /etc/fdfs/storage.conf

1. 建立一个保存storage的相关目录（以后要与tracke所在目录分开）：



1. 编译storage.conf配置文件：



1. 【FastDFS-server-01】启动相应的服务：
2. 启动tracker服务：

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

1. 启动storage服务：

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

### FastDFS测试

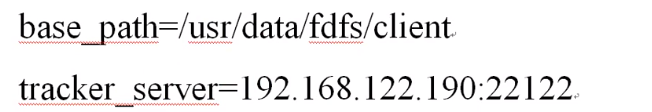
1. 要想操作客户端，那么需要一个客户端专属的配置文件
2. 通过模板拷贝一个配置文件:



1. 建立一个客户端的保存目录：



1. 编辑配置文件：



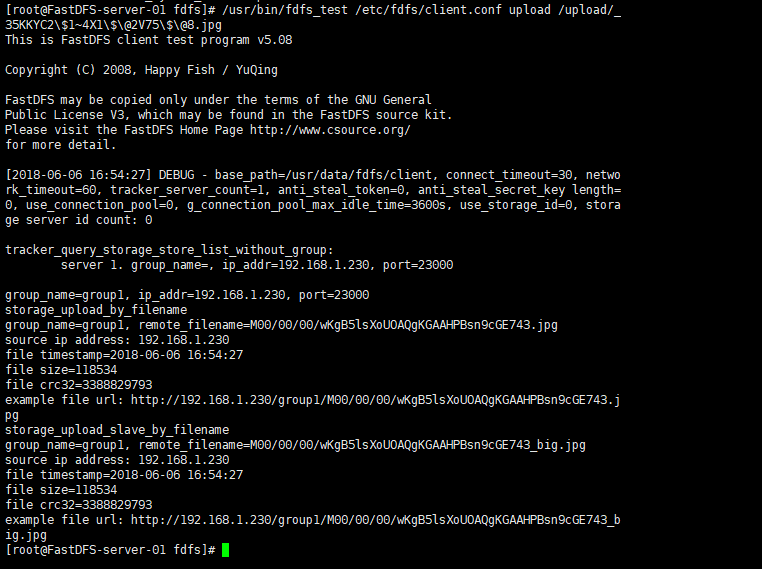
1. 通过客户端查看服务器的当前状态：

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

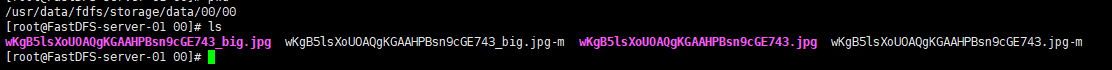
1. 通过客户端工具上传一个文件：



此时得到如下信息：



文件保存目录：/usr/data/fdfs/storage/data/00/00



1. 下载测试

/usr/bin/fdfs\_download\_file /etc/fdfs/client.conf group1/M00/00/00/wKgB5lsXoUOAQgKGAAHPBsn9cGE743.jpg

1. 取得文件信息:

/usr/bin/fdfs\_file\_info /etc/fdfs/client.conf group1/M00/00/00/wKgB5lsXoUOAQgKGAAHPBsn9cGE743.jpg

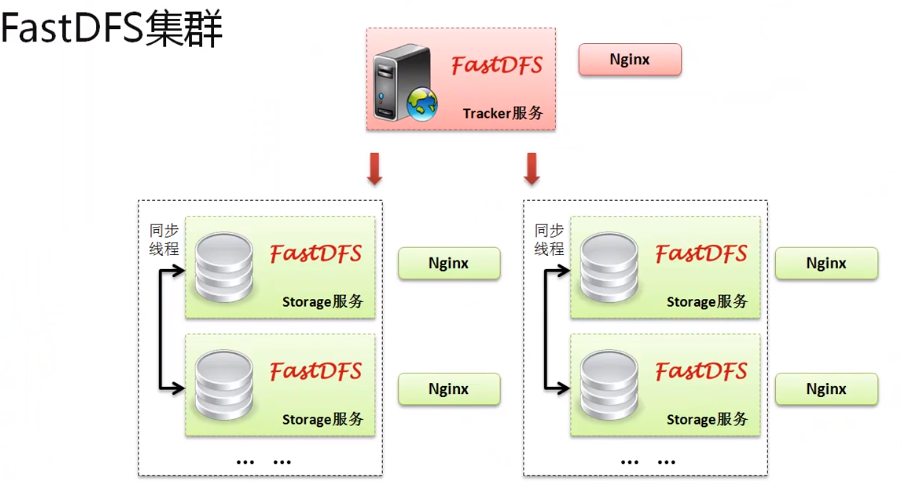
1. 删除文件

/usr/bin/fdfs\_delete\_file /etc/fdfs/client.conf group1/M00/00/00/wKgB5lsXoUOAQgKGAAHPBsn9cGE743.jpg

## 搭建FastDFS分布式集群

由于FastDFS本身不再默认集成WEB服务器，所以需要结合Nginx搭建好WEB服务器。

本次将搭建两组服务器：group1、group2。



### 环境搭建与主机说明

|  |  |  |  |
| --- | --- | --- | --- |
| No. | 主机名称 | IP地址 | 描述 |
| 1 | fastdfs-tracker-server | 192.168.1.225 | Ttracker、ngnix(tracker) |
| 2 | fastdfs-storage-group1-01 | 192.168.1.229 | storage、ngnix(fastdfs-module) |
| 3 | fastdfs-storage-group1-02 | 192.168.1.228 | storage、ngnix(fastdfs-module) |
| 4 | fastdfs-storage-group2-01 | 192.168.1.227 | storage、ngnix(fastdfs-module) |
| 5 | fastdfs-storage-group2-02 | 192.168.1.226 | storage、ngnix(fastdfs-module) |

### 配置storage主机

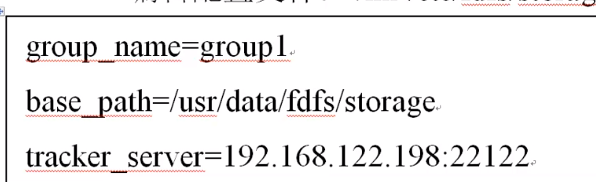
1. 【fastdfs-storage-group1-01】进行storage配置
2. 通过模板文件拷贝出配置文件：

cp /etc/fdfs/storage.conf.sample /etc/fdfs/storage.conf

1. 建立工作目录：

mkdir -p /usr/data/fdfs/storage

1. 编辑配置文件(storage.conf)：



1. 【fastdfs-storage-group1-01】将本次使用的Nginx开发包上传到服务器：

nginx-1.11.3.tar、echo-nginx-module-0.59.tar、fastdfs-nginx-module\_v1.16.tar、

ngx\_cache\_purge-2.3.tar、nginx-upstream-fair-a18b409.tar（负载均衡）、zlib-1.2.8.tar

、pcre-8.36.tar

1. 将所有源代码开发包进行解压缩到“/usr/local/src”。

tar xvf /upload/zlib-1.2.8.tar.gz -C /usr/local/src/

上面这个压缩包不加z参数

1. 【fastdfs-storage-group1-01】编译nginx:

(1)需要建立好nginx编译之后保存的数据目录：

mkdir -p /usr/local/nginx/{logs,conf,fastcgi\_temp,sbin,client\_body\_temp,proxy\_temp,uwsgi\_temp,scgi\_temp}

(2)进行aginx编译(nginx组件目录)：

./configure --prefix=/usr/local/nginx/ \

--sbin-path=/usr/local/nginx/sbin/ \

--with-http\_ssl\_module \

--conf-path=/usr/local/nginx/conf/nginx.conf \

--pid-path=/usr/local/nginx/logs/nginx.pid \

--error-log-path=/usr/local/nginx/logs/error.log \

--http-log-path=/usr/local/nginx/logs/access.log \

--http-fastcgi-temp-path=/usr/local/nginx/fastcgi\_temp \

--http-client-body-temp-path=/usr/local/nginx/client\_body\_temp \

--http-proxy-temp-path=/usr/local/nginx/proxy\_temp \

--http-uwsgi-temp-path=/usr/local/nginx/uwsgi\_temp \

--http-scgi-temp-path=/usr/local/nginx/scgi\_temp \

--add-module=/usr/local/src/echo-nginx-module-0.59 \

--add-module=/usr/local/src/gnosek-nginx-upstream-fair-a18b409 \

--add-module=/usr/local/src/ngx\_cache\_purge-2.3 \

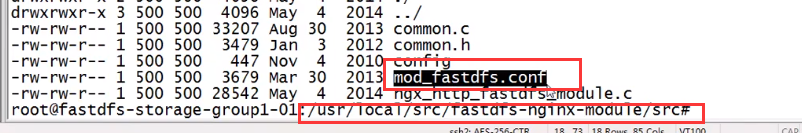
--add-module=/usr/local/src/fastdfs-nginx-module/src \

--with-zlib=/usr/local/src/zlib-1.2.8 \

--with-pcre=/usr/local/src/pcre-8.36

(3)编译并安装(nginx组件目录)：make && make install

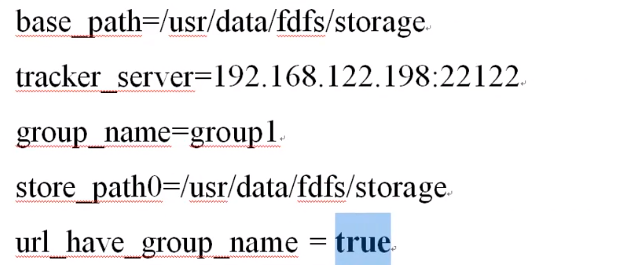
1. 【fastdfs-storage-group1-01】将fastdfs-nginx-modle模块中的配置文件拷贝到“/etc/fdfs”目录下：





cp /usr/local/src/fastdfs-nginx-module//src/mod\_fastdfs.conf /etc/fdfs/

1. 【fastdfs-storage-group1-01】修改“mod\_fastdfs.conf”配置文件：



“url\_have\_group\_name”表示现在的访问路径上是否需要追加有组名称

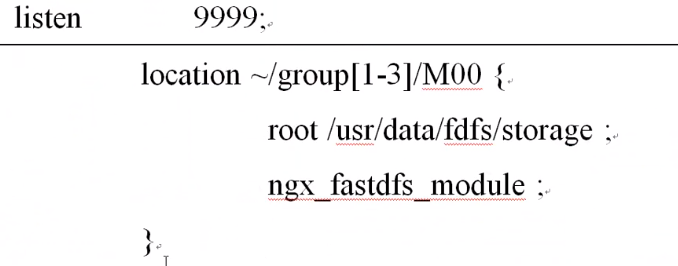
1. 【fastdfs-storage-group1-01】拷贝配置文件：
2. 拷贝http.conf配置文件：



1. 拷贝mime.types配置文件:



1. 【fastdfs-storage-group1-01】修改nginx配置项:



1. 【fastdfs-storage-group1-01】做一个软连接：



ln -sv /usr/data/fdfs/storage/data /usr/data/fdfs/storage/M00

1. 【fastdfs-storage-group1-01】检测当前nginx配置：



### fdfs-0307\_配置storage

1、【astdfs-storage-group1-01】关机克隆，依次复制为：

|  |  |
| --- | --- |
| fastdfs-storage-group1-02 | 192.168.1.228 |
| fastdfs-storage-group2-01 | 192.168.1.227 |
| fastdfs-storage-group2-02 | 192.168.1.226 |

然后修改主机名称，ip,及主机名称映射：

/etc/hostname、/etc/sysconfig/network-scripts/ifcfg-ens33、/etc/hosts

2、【astdfs-storage-group2-\*】修改组的名称：





3、【astdfs-storage-group2-\*】修改组名称：





### 配置tracker主机

每一台storage上的nginx只是作为负载均衡使用的，真正起作用的还在于tracker主机上。

1. 【fastdfs-tracker-server】将nginx有关的开发包上传到服务器解压到“/usr/local/src”。

nginx-1.11.3.tar、echo-nginx-module-0.59.tar、

ngx\_cache\_purge-2.3.tar、nginx-upstream-fair-a18b409.tar（负载均衡）、zlib-1.2.8.tar

、pcre-8.36.tar

1. 【fastdfs-tracker-server】进行nginx编译处理：
2. 建立相应目录：

mkdir -p /usr/local/nginx/{logs,conf,fastcgi\_temp,sbin,client\_body\_temp,proxy\_temp,uwsgi\_temp,scgi\_temp}

1. 进入nginx源代码目录下：

./configure --prefix=/usr/local/nginx/ \

--sbin-path=/usr/local/nginx/sbin/ \

--with-http\_ssl\_module \

--conf-path=/usr/local/nginx/conf/nginx.conf \

--pid-path=/usr/local/nginx/logs/nginx.pid \

--error-log-path=/usr/local/nginx/logs/error.log \

--http-log-path=/usr/local/nginx/logs/access.log \

--http-fastcgi-temp-path=/usr/local/nginx/fastcgi\_temp \

--http-client-body-temp-path=/usr/local/nginx/client\_body\_temp \

--http-proxy-temp-path=/usr/local/nginx/proxy\_temp \

--http-uwsgi-temp-path=/usr/local/nginx/uwsgi\_temp \

--http-scgi-temp-path=/usr/local/nginx/scgi\_temp \

--add-module=/usr/local/src/echo-nginx-module-0.59 \

--add-module=/usr/local/src/gnosek-nginx-upstream-fair-a18b409 \

--add-module=/usr/local/src/ngx\_cache\_purge-2.3 \

--with-zlib=/usr/local/src/zlib-1.2.8 \

--with-pcre=/usr/local/src/pcre-8.36

1. 编译与安装：make && make install
2. 【fastdfs-tracker-serve】配置tracker：
3. 建立tracker工作目录：

mkdir -p /usr/data/fdfs/tracker

1. 通过模板拷贝配置文件：

cp /etc/fdfs/tracker.conf.sample /etc/fdfs/tracker.conf

1. 编辑配置文件：



1. 【fastdfs-tracker-serve】配置nginx:
2. tracker所在主机整体访问的主机，所以为了访问迅速需要配置缓存：



1. 修改nginx配置（nginx.conf）：

#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 65536;

use epoll;

}

http {

include mime.types;

default\_type application/octet-stream;

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;

proxy\_cache\_path /usr/data/nginx/cache levels=1:2 keys\_zone=http-cache:500m max\_size=10g inactive=30d;

proxy\_temp\_path /usr/data/nginx/tmp;

upstream fdfs\_group1 {

server 192.168.1.229:9999 weight=1 max\_fails=2 fail\_timeout=30s;

server 192.168.1.228:9999 weight=1 max\_fails=2 fail\_timeout=30s;

}

upstream fdfs\_group2 {

server 192.168.1.227:9999 weight=1 max\_fails=2 fail\_timeout=30s;

server 192.168.1.226:9999 weight=1 max\_fails=2 fail\_timeout=30s;

}

#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;

server {

listen 80;

server\_name localhost;

#charset koi8-r;

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

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;

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;

proxy\_pass http://fdfs\_group2;

expires 30d;

}

location ~ /purge(/.\*) {

allow 127.0.0.1;

allow 192.168.1.0/24;

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;

# }

#}

}

### 集群启动与测试

1. 【fastdfs-tracker-server】启动相应的tracker进程：

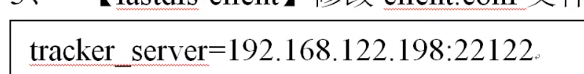
/usr/bin/fdfs\_trackerd /etc/fdfs/tracker.conf

1. 【fastdfs-storage-\*】启动所有的storage进程：

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

1. 【fastdfs-client】修改client.conf文件，主要设置tracker:





1. 【fastdfs-client】检测当前的服务状态：



1. 【fastdfs-storage-\*】启动nginx服务：

/usr/local/nginx/sbin/nginx

1. 【fastdfs-tracker-server】启动nginx服务：

/usr/local/nginx/sbin/nginx

1. 【fastdfs-client】上传图片：

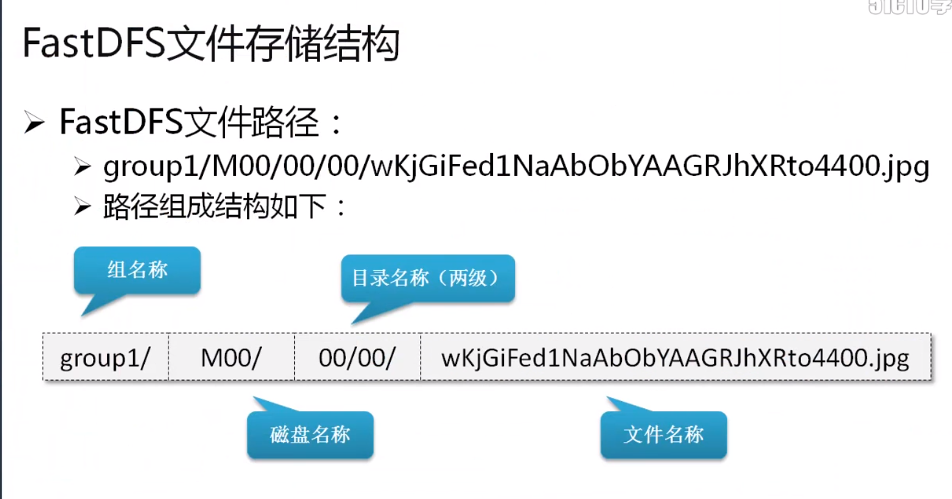


给出的访问地址：

http://192.168.1.225/group2/M00/00/00/wKgB4lsYsgmAe2tQAAHPBsn9cGE252\_big.jpg

清除缓存：

http://192.168.1.225/purge/group2/M00/00/00/wKgB4lsYsgmAe2tQAAHPBsn9cGE252\_big.jpg



1. 设置自动切换（tracker端的tracker.conf）：



表示会根据磁盘空间自动分配存储（注释store\_group=group2）,重新启动tracker服务，先杀死进程：



客户端测试遇到问题：

1. [2018-06-07 10:31:10] ERROR - file: connection\_pool.c, line: 130, connect to 192.168.1.225:22122 fail, errno: 113, error info: No route to host

解决方法：关闭防火墙

2、storage服务无法启动，查看日志发现是store\_path0路径没改。

## 使用java操作FastDFS

使用java操作fastdfs,需要支持包：fastdfs\_client\_v1.20，同时为了方便需要一个专门的客户端的配置文件，一般会保存在classpath路径下，为方便读取，直接使用spring完成资源的读取。

### 上传文件

1. 将开发包加入到项目中：

fastdfs\_client\_v1.20、spring-core-4.3.3.RELEASE

1. 在src下建立一个fdfs\_client.conf的配置文件：

tracker\_server=192.168.1.225:22122

1. 实现上传：

// 1、设置一个上传文件的路径；

File imgFile = new File("D:" + File.separator + "pic-c.jpg") ;

// 取得文件扩展名称

String fileExtName = imgFile.getName().substring(imgFile.getName().lastIndexOf(".") + 1) ;

// 2、读取上传的配置文件，此配置文件在CLASSPATH路径下

ClassPathResource res = new ClassPathResource("fdfs\_client.conf") ;

// 3、初始化FastDFS上传的环境

ClientGlobal.init(res.getClassLoader().getResource("fdfs\_client.conf").getPath());

// 4、建立Tracker的客户端连接

TrackerClient tracker = new TrackerClient() ;

TrackerServer trackerServer = tracker.getConnection() ; // 取得服务器连接

// 5、真正负责数据保存的是Storage

StorageServer storageServer = null ;

// 6、进行StoragerClient的创建

StorageClient client = new StorageClient(trackerServer,storageServer) ;

// 7、创建文件的元数据对象

NameValuePair[] metaList = new NameValuePair[3] ;

metaList[0] = new NameValuePair("fileName",imgFile.getName()) ;

metaList[1] = new NameValuePair("fileExtName",fileExtName) ;

metaList[2] = new NameValuePair("fileLength",String.valueOf(imgFile.length())) ;

// 8、实现文件上传，返回文件的路径名称

String [] fileId = client.upload\_file(imgFile.getPath(), fileExtName, metaList) ;

System.out.println(Arrays.toString(fileId));

trackerServer.close();

// 1、设置一个上传文件的路径；

File imgFile = new File("D:" + File.separator + "pic-c.jpg") ;

// 取得文件扩展名称

String fileExtName = imgFile.getName().substring(imgFile.getName().lastIndexOf(".") + 1) ;

// 2、读取上传的配置文件，此配置文件在CLASSPATH路径下

ClassPathResource res = new ClassPathResource("fdfs\_client.conf") ;

// 3、初始化FastDFS上传的环境

ClientGlobal.init(res.getClassLoader().getResource("fdfs\_client.conf").getPath());

// 4、建立Tracker的客户端连接

TrackerClient tracker = new TrackerClient() ;

TrackerServer trackerServer = tracker.getConnection() ; // 取得服务器连接

// 5、真正负责数据保存的是Storage

StorageServer storageServer = null ;

// 6、进行StoragerClient的创建

StorageClient1 client = new StorageClient1(trackerServer,storageServer) ;

// 7、创建文件的元数据对象

NameValuePair[] metaList = new NameValuePair[3] ;

metaList[0] = new NameValuePair("fileName",imgFile.getName()) ;

metaList[1] = new NameValuePair("fileExtName",fileExtName) ;

metaList[2] = new NameValuePair("fileLength",String.valueOf(imgFile.length())) ;

// 8、实现文件上传，返回文件的路径名称

String fileId = client.upload\_file1(imgFile.getPath(), fileExtName, metaList) ;

System.out.println(fileId);

trackerServer.close();

### 取得文件信息

// 1、读取上传的配置文件，此配置文件在CLASSPATH路径下

ClassPathResource res = new ClassPathResource("fdfs\_client.conf") ;

// 2、初始化FastDFS上传的环境

ClientGlobal.init(res.getClassLoader().getResource("fdfs\_client.conf").getPath());

// 3、建立Tracker的客户端连接

TrackerClient tracker = new TrackerClient() ;

TrackerServer trackerServer = tracker.getConnection() ; // 取得服务器连接

// 4、真正负责数据保存的是Storage

StorageServer storageServer = null ;

// 5、进行StoragerClient的创建

StorageClient1 client = new StorageClient1(trackerServer,storageServer) ;

// 6、取得文件的相关信息

String fileName = "group2/M00/00/00/wKh6yVg7zQSAFqYWAAL6dz7aOB0479.jpg" ;

FileInfo fi = client.get\_file\_info1(fileName) ; // 取得该文件内容的信息

System.out.println("得到文件大小：" + fi.getFileSize());

System.out.println("创建日期：" + fi.getCreateTimestamp());

System.out.println("真实保存主机：" + fi.getSourceIpAddr());

System.out.println("CRC32：" + fi.getCrc32());

trackerServer.close();

### 删除文件

// 1、读取上传的配置文件，此配置文件在CLASSPATH路径下

ClassPathResource res = new ClassPathResource("fdfs\_client.conf") ;

// 2、初始化FastDFS上传的环境

ClientGlobal.init(res.getClassLoader().getResource("fdfs\_client.conf").getPath());

// 3、建立Tracker的客户端连接

TrackerClient tracker = new TrackerClient() ;

TrackerServer trackerServer = tracker.getConnection() ; // 取得服务器连接

// 4、真正负责数据保存的是Storage

StorageServer storageServer = null ;

// 5、进行StoragerClient的创建

StorageClient1 client = new StorageClient1(trackerServer,storageServer) ;

// 6、取得文件的相关信息

String fileName = "group2/M00/00/00/wKh6yVg7zQSAFqYWAAL6dz7aOB0479.jpg" ;

System.out.println(client.delete\_file1(fileName));

trackerServer.close();

删除成功返回0，删除失败返回2.

## FastDFS安全配置

### 连接白名单

链接有黑名单和白名单两个概念，所谓的白名单指的是允许访问的用户。

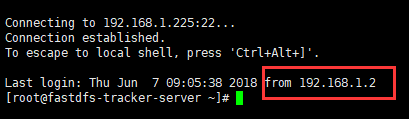
修改白名单主要配置项在tracker.conf文件中。

allow\_host=192.168.1。[225-229]

表示允许上床的主机为：192.168.1.225~192.168.1.229之间的IP地址。

allow\_host=192.168.1。[225-229，230]，现在表示230的IP也可以访问。

如果是虚拟机的话，可能需要将网关配置上去。



### 盗链防范

所谓盗链指的是生成一个随机的token，这个token在某一段时间内有效，超过了指定时间将出现错误页面。防盗链的功能可以通过修改storage主机配置实现。

1. 【fastdfs-storage-\*】上传盗链之后的显示图片：
2. 所有的fastdfs的配置文件的保存路径“/etc/fdfs”,将图片保存在此路径中：

scp /upload/noauth.gif 192.168.1.229:/etc/fdfs/

scp /upload/noauth.gif 192.168.1.228:/etc/fdfs/

scp /upload/noauth.gif 192.168.1.227:/etc/fdfs/

scp /upload/noauth.gif 192.168.1.226:/etc/fdfs/

1. 将图片拷贝到所有的storage主机：
2. 【fastdfs-storage-\*】修改防盗链的配置文件（/etc/fdfs/http.conf）：
3. 是否检测token，是否开启防盗链：

http.anti\_steal.check\_token=true

1. 设置每一个token的有效时间(秒)：

http.anti\_steal.token\_ttl=60

1. 设置生成token的密码：

http.anti\_steal.secret\_key=yangmi1234567890

1. token失效后显示的图片

http.anti\_steal.token\_check\_fail=/etc/fdfs/noauth.gif

1. 【fastdfs-storage-\*】将此时的http.conf配置拷贝到其他storage主机：

#scp /etc/fdfs/http.conf 192.168.1.229:/etc/fdfs/

scp /etc/fdfs/http.conf 192.168.1.228:/etc/fdfs/

scp /etc/fdfs/http.conf 192.168.1.227:/etc/fdfs/

scp /etc/fdfs/http.conf 192.168.1.226:/etc/fdfs/

4、【fastdfs-storage-\*】重新启动所有storage服务及nginx服务：

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

/usr/local/nginx/sbin/nginx -s reload

5、 【fastdfs-storage-\*】伤透token失效后要求显示t固定图片，这张图片需要授权。

chmod 777 -R /etc/fdfs/noauth.gif

6、【fastdfs-storage-\*】通过java程序生成客户端的token(本机和服务器时间同步)。

(1)修改“fasf\_client.conf”：配置文件：

tracker\_server=192.168.1.225:22122  
http.anti\_steal\_token=true  
http.anti\_steal.secret\_key=yangmi1234567890

(2)编写程序生成token:

**public static void** main(String[] args) **throws** Exception {  
 *// 1、读取上传的配置文件，此配置文件在CLASSPATH路径下* ClassPathResource res = **new** ClassPathResource(**"fdfs\_client.conf"**) ;  
 *// 2、初始化FastDFS上传的环境* ClientGlobal.*init*(res.getClassLoader().getResource(**"fdfs\_client.conf"**).getPath());  
 *// 3、建立Tracker的客户端连接* TrackerClient tracker = **new** TrackerClient() ;  
 TrackerServer trackerServer = tracker.getConnection() ; *// 取得服务器连接  
 // 定义文件id的时候千万不要增加组名称，否则无法访问* String fileId = **"M00/00/00/wKgB5FsY652AL6vyAAHPBsn9cGE577.jpg"** ;  
 **int** ts = (**int**) (System.*currentTimeMillis*() / 1000) ; *// 取得一个当前的时间标记* String token = ProtoCommon.*getToken*(fileId, ts, ClientGlobal.*g\_secret\_key*) ;  
 StringBuffer fileUrl = **new** StringBuffer() ;  
 fileUrl.append(**"http://"**) ;   
 fileUrl.append(trackerServer.getInetSocketAddress().getHostString()) ;  
 fileUrl.append(**"/group2/"**).append(fileId) ;  
 fileUrl.append(**"?token="**).append(token).append(**"&ts="**).append(ts) ;  
 System.***out***.println(fileUrl);  
 trackerServer.close();   
}