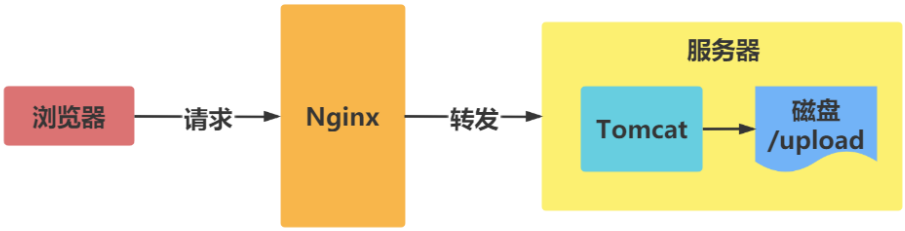
为什么要使用分布式文件系统

单机时代

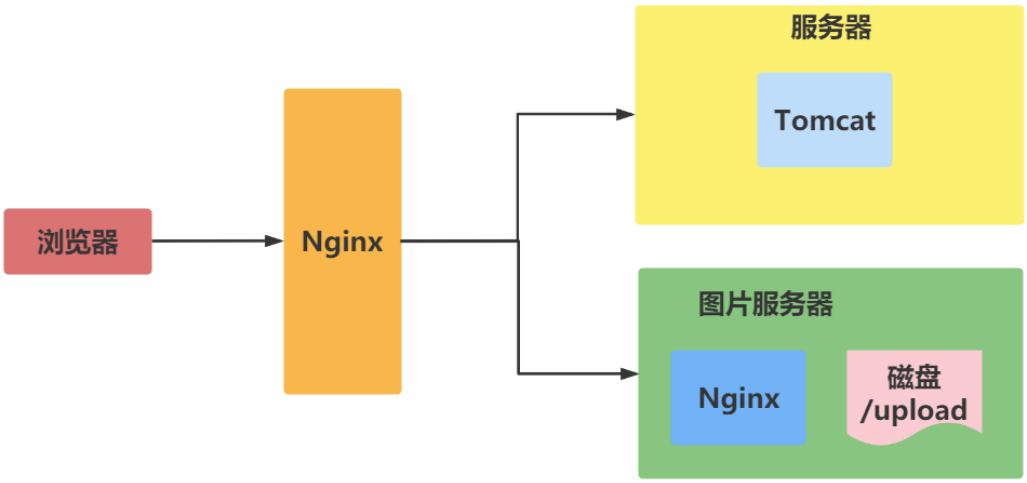
初创时期由于时间紧迫，在各种资源有限的情况下，通常就直接在项目目录下建立静态文件夹，用于用户存放项目中的文件资源。如果按不同类型再细分，可以在项目目录下再建立不同的子目录来区分。例如：`resources\static\file`、`resources\static\img`等。



> **注意：**
>
> **优点**：便利，使用方便。
>
> **缺点**：文件越多存放越混乱。

独立文件服务器

随着公司业务不断发展，将代码和文件放在同一服务器的弊端就会越来越明显。为了解决上面的问题引入独立图片服务器，

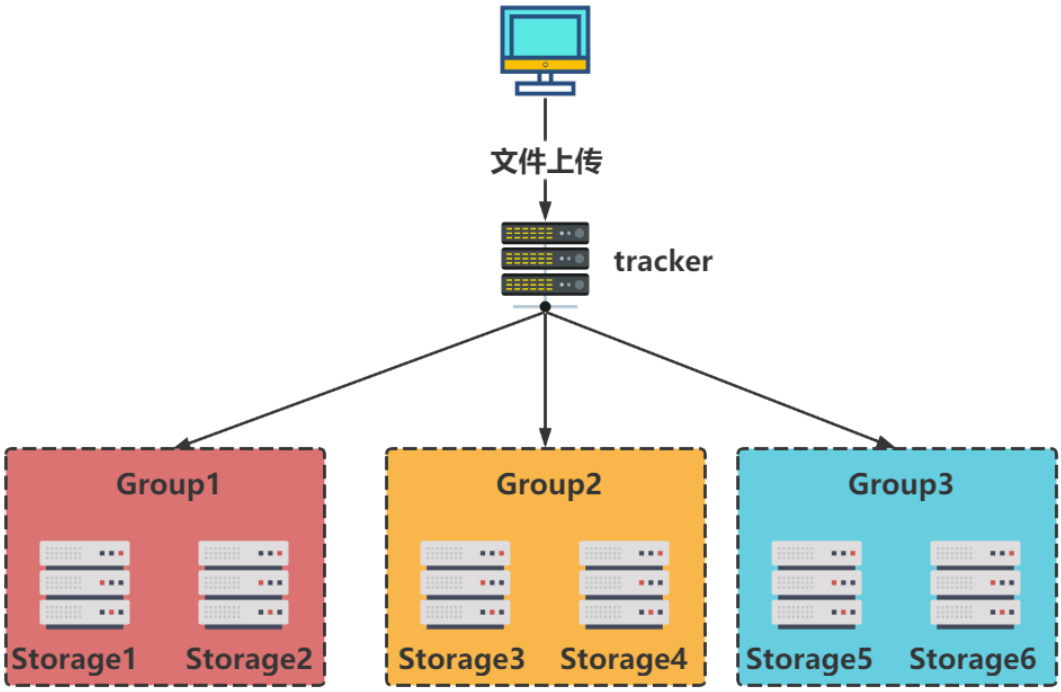


流程：

项目上传文件时，首先通过ftp或者ssh将文件上传到图片服务器的某个目录下，再通过Ngnix或者Apache来访问此目录下的文件，返回一个独立域名的图片URL地址，前端使用文件时就通过这个URL地址读取。

分布式文件系统

业务继续发展，单台服务器存储和响应也很快到达了瓶颈，新的业务需要文件访问具有高响应性、高可用性来支持系统。



优点：

- **扩展能力**: 毫无疑问，扩展能力是一个分布式文件系统最重要的特点；
- **高可用性**: 在分布式文件系统中，高可用性包含两层，一是整个文件系统的可用性，二是数据的完整和一致性；
- **弹性存储**: 可以根据业务需要灵活地增加或缩减数据存储以及增删存储池中的资源，而不需要中断系统运行。

缺点：系统复杂度稍高，需要更多服务器

实时学习反馈

1.下列属于独立文件服务器缺点是___。

A 容灾

B 单点故障问题

C 垂直扩展性稍差

D 以上都是

2. 下列属于分布式文件系统缺点的是____。

A 扩展能力强

B 高可用性

C 弹性存储

D 系统复杂度稍高

答案

1=>D 2=>D

FastDFS概述_简介

FastDFS是一个开源的轻量级分布式文件系统。它解决了大数据量存储和负载均衡等问题。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，如相册网站、视频网站等等。

FastDFS特性:

- 文件不分块存储，上传的文件和OS文件系统中的文件一一对应
- 支持相同内容的文件只保存一份，节约磁盘空间
- 下载文件支持HTTP协议，可以使用内置Web Server，也可以和其他Web Server配合使用
- 支持在线扩容
- 支持主从文件

分布式文件服务提供商

1、阿里的OSS

2、七牛云存储

3、百度云储存

实时学习反馈

1.FastDFS是一个开源的轻量级____。

A Web平台

B 文件系统

C 分布式文件系统

D 代码托管平台

2. FastDFS分布式文件系统特别适合存储____文件。

A 小文件

B 大文件

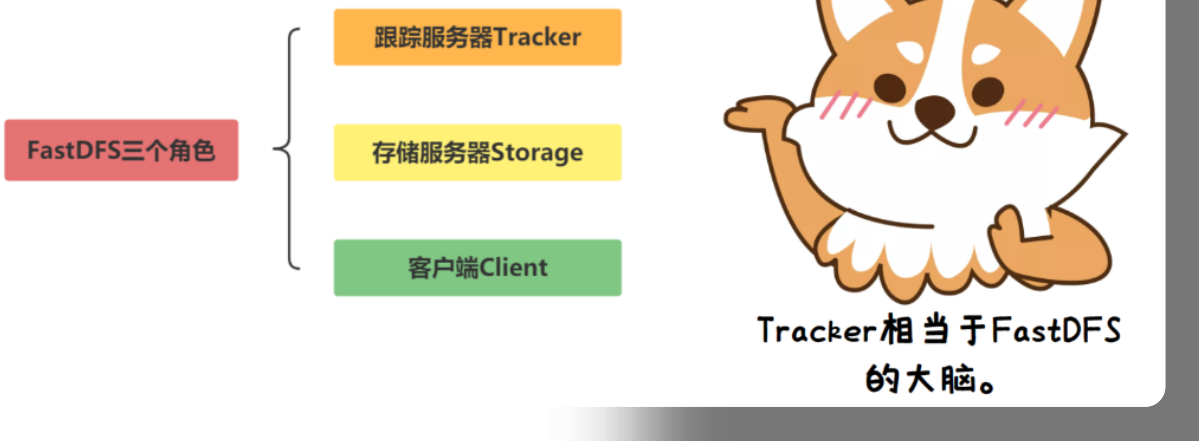
C 中小型

D 以上都是错误

答案

1=>C 2=>C

FastDFS概述_核心概念



FastDFS服务端有三个角色：跟踪服务器（tracker）、存储服务器（storage）和客户端（client）。

tracker

跟踪服务器，主要做调度工作，起负载均衡的作用。在内存中记录集群中所有存储组和存储服务器的状态信息，是客户端和数据服务器交互的枢纽。

storage

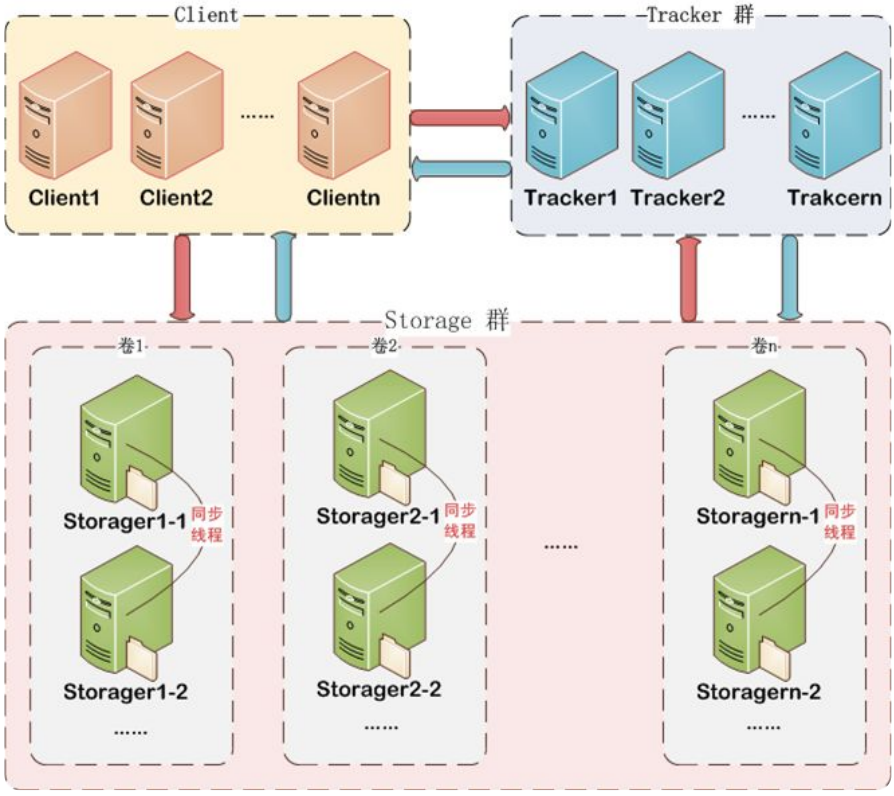
存储服务器（又称：存储节点或数据服务器），文件和文件属性（meta data）都保存到存储服务器上。Storage server直接利用OS的文件系统调用管理文件。

client

客户端，作为业务请求的发起方，通过专有接口，使用TCP/IP协议与跟踪器服务器或存储节点进行数据交互。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。

group

组， 也可称为卷。 同组内服务器上的文件是完全相同的， 同一组内的storage server之间是对等的， 文件上传、 删除等操作可以在任意一台storage server上进行 。



流程：

Tracker相当于FastDFS的大脑，不论是上传还是下载都是通过tracker来分配资源；客户端一般可以使用Ngnix等静态服务器来调用或者做一部分的缓存；存储服务器内部分为卷（或者叫做组），卷于卷之间是平行的关系，可以根据资源的使用情况随时增加，卷内服务器文件相互同步备份，以达到容灾的目的。

实时学习反馈

1.FastDFS的大脑是___。

A Tracker

B storage

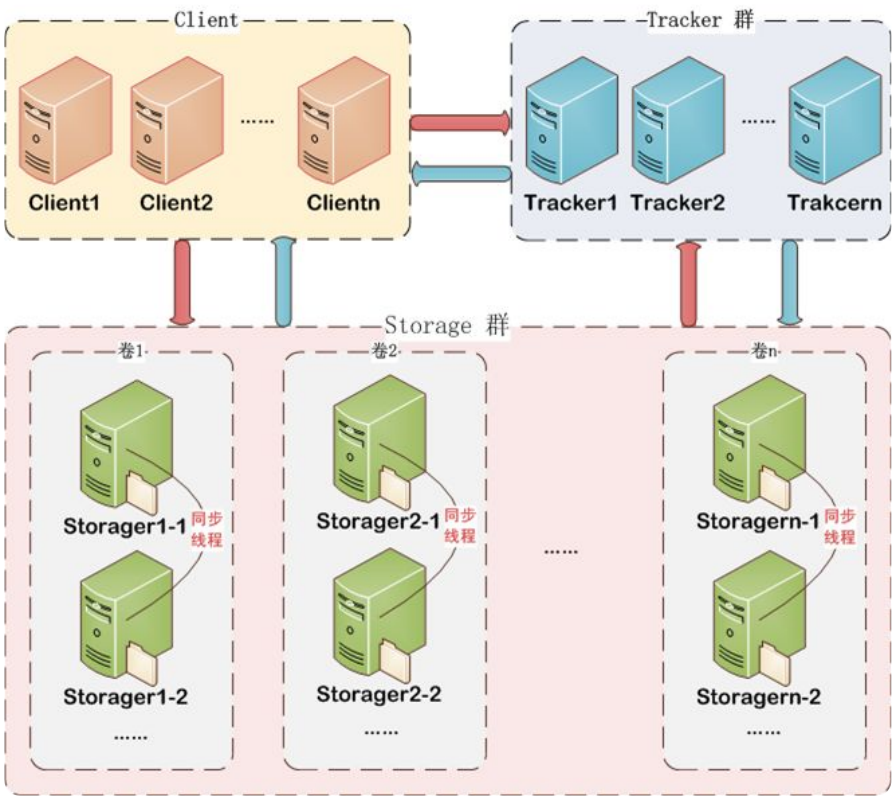
C group

D client

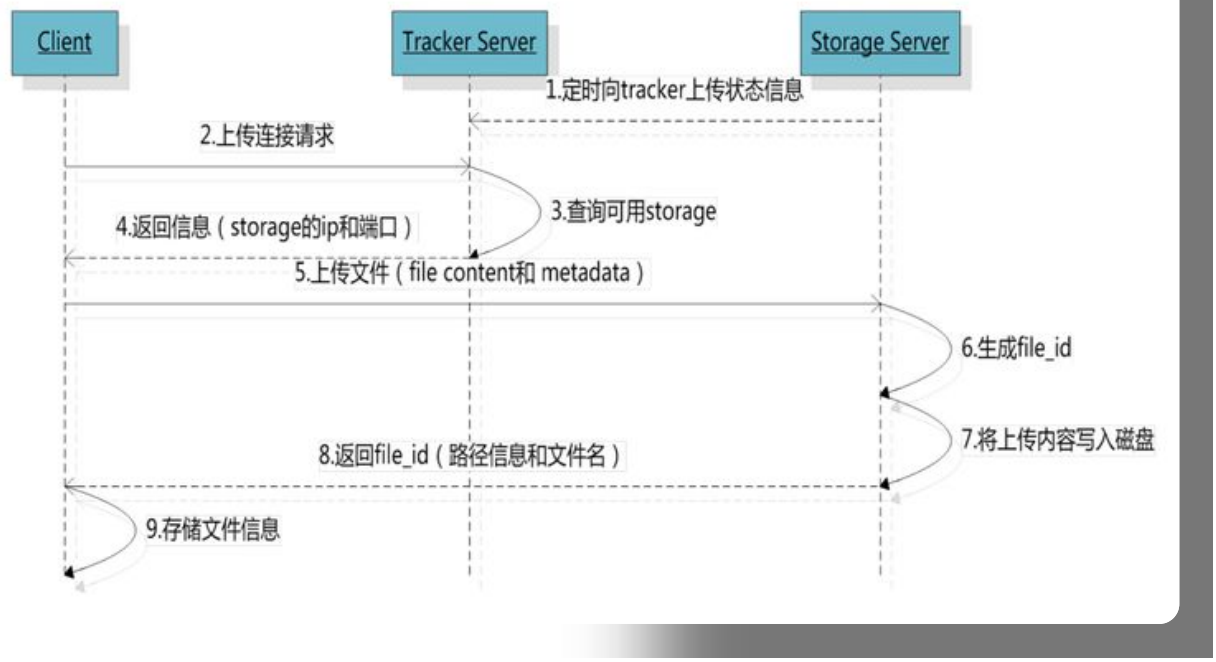
答案

1=>B

FastDFS概述_上传机制



首先客户端请求Tracker服务获取到存储服务器的ip地址和端口，然后客户端根据返回的IP地址和端口号请求上传文件，存储服务器接收到请求后生产文件，并且将文件内容写入磁盘并返回给客户端file_id、路径信息、文件名等信息，客户端保存相关信息上传完毕。



内部机制如下

1、选择Tracker server

当集群中不止一个Tracker server时，由于Tracker之间是完全对等的关系，客户端在upload文件时可以任意选择一个trakcer。

2、选择Storage server

当选定Group后，Tracker会在Group内选择一个Storage Server给客户端

3、选择Storage path

当分配好Storage Server后，客户端将向Storage发送写文件请求，Storage将会为文件分配一个数据存储目录。

> **注意：**
>
> 剩余存储空间最多的优先。

4、生成Fileid

选定存储目录之后，Storage会为文件生一个Fileid，由Storage Server Ip、文件创建时间、文件大小、文件crc32和一个随机数拼接而成，然后将这个二进制串进行base64编码，转换为可打印的字符串。

5、生成文件名

当文件存储到某个子目录后，即认为该文件存储成功，接下来会为该文件生成一个文件名，文件名由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。

实时学习反馈

1.FastDFS上传文件成功后返回文件名由上面组成的___。

A 存储目录、两级子目录、fileid、文件后缀名

B group、两级子目录、fileid、文件后缀名

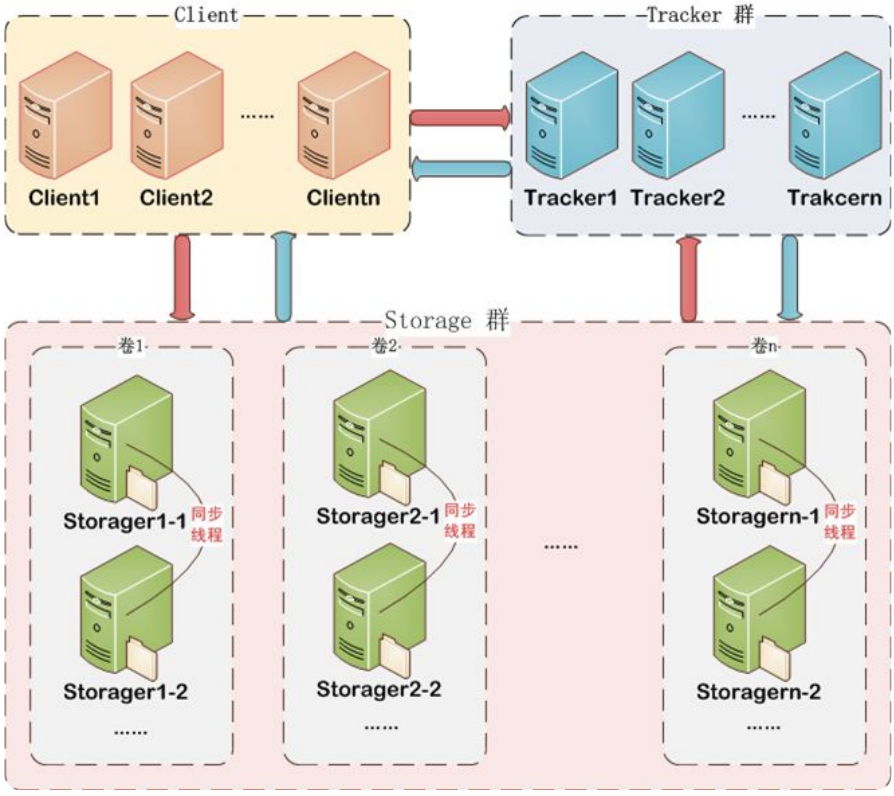
C group、存储目录、两级子目录、fileid、文件后缀名

D group、存储目录、fileid、文件后缀名

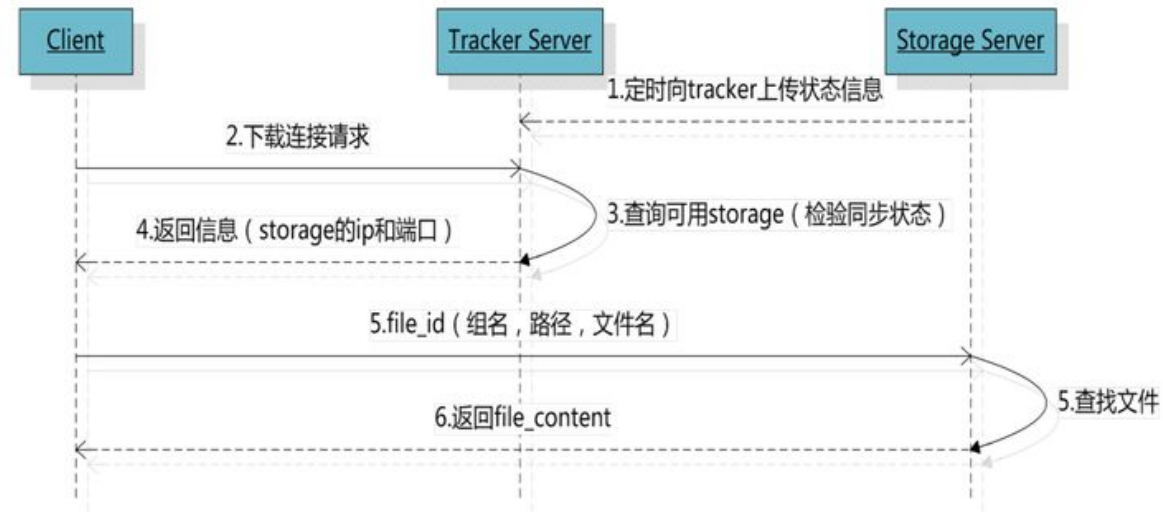
答案

1=>C

FastDFS概述_下载机制



客户端带上文件名信息请求Tracker服务获取到存储服务器的ip地址和端口，然后客户端根据返回的IP地址和端口号请求下载文件，存储服务器接收到请求后返回文件给客户端。



内部机制如下

1. client询问tracker下载文件的storage，参数为文件标识（组名和文件名）
2. tracker返回一台可用的storage
3. client直接和storage通讯完成文件下载

实时学习反馈

1.FastDFS分布式文件系统在下载文件的时候客户端带文件名请求___。

A Tracker

B Stroage

C Group

D 以上都是错误

答案

1=>A

FastDFS环境搭建_Linux

下载安装gcc

安装方式为yum安装（需网络）：

```
yum install gcc-c++ perl-devel pcre-devel openssl-devel zlib-devel wget
```

下载安装FastDFS

```
wget https://github.com/happyfish100/fastdfs/archive/V6.06.tar.gz
```

下载安装FastDFS依赖

```
wgethttps://github.com/happyfish100/libfastcommon/archive/V1.0.43.tar.gz
```

解压缩依赖tar包

```
tar -zxvf V1.0.43.tar.gz -C /usr/local
tar -zxvf V5.11.tar.gz -C /usr/local
```

编译并安装libfastcommon

```
 cd /usr/local/libfastcommon-1.0.43/
 ./make.sh && ./make.sh install
```

编译并安装FastDFS

```
cd /usr/local/fastdfs-6.06
./make.sh && ./make.sh install
```

进入etc目录下复制配置文件

```
cd /etc/fdfs/
cp client.conf.sample client.conf
cp storage.conf.sample storage.conf
cp tracker.conf.sample tracker.conf
```

创建tracker服务

创建tracker目录

```
mkdir -p  /data/fastdfs/tracker
```

修改配置文件

```
vim /etc/fdfs/tracker.conf
disabled=false                    #启用配置文件
port=22122                        #设置 tracker 的端口号
base_path=/data/fastdfs/tracker #设置 tracker 的数据文件和日志目录（需预先创建）
http.server_port=8888             #设置 http 端口号
http.server_port=8888             #指的是在tracker服务器上启动http服务进程，如:apache或者nginx 启动时所监听的端口
```

启动tracker服务

```
/etc/init.d/fdfs_trackerd start
```

检查tracker服务

```
netstat -lntup |grep fdfs
tcp        0      0 0.0.0.0:22122           0.0.0.0:*               LISTEN      10757/fdfs_trackerd
```

创建storage服务

创建storage目录

```
mkdir -p /data/fastdfs/base
mkdir -p /data/fastdfs/storage
```

修改配置文件

```
vim /etc/fdfs/storage.conf
disabled=false                                  #启用配置文件
group_name=group1                               #组名，根据实际情况修改
port=23000                                      #设置 storage 的端口号
base_path=/data/fastdfs/base                    #设置 storage 的日志目录（需预先创建）
store_path_count=1                              #存储路径个数，需要和 store_path 个数匹配
store_path0=/data/fastdfs/storage               #存储路径
tracker_server=172.31.16.121:22122              #tracker 服务器的 IP 地址和端口号
http.server_port=8888                           #设置storage上启动的http服务的端口号，如安装的nginx的端口号
```

启动storage服务

```
/etc/init.d/fdfs_storaged start
```

查看storage服务

```
netstat -lntup |grep fdfs
tcp        0      0 0.0.0.0:23000           0.0.0.0:*               LISTEN      10892/fdfs_storaged
tcp        0      0 0.0.0.0:22122           0.0.0.0:*               LISTEN      10757/fdfs_trackerd
```

修改Client配置文件

```
vim /etc/fdfs/client.conf
connect_timeout=30
network_timeout=60
base_path=/data/fastdfs/client                  # 日志路径
tracker_server=192.168.66.100:22122             # 追踪服务器的IP，有多个服务器可以另一行
```

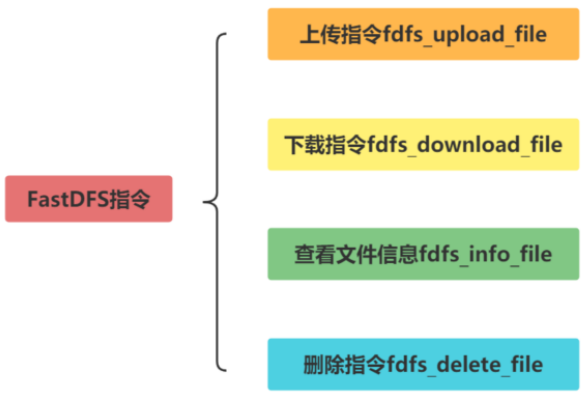
创建日志目录

```
mkdir -p /data/fastdfs/client
```

查看启动的服务

```
ps -ef | grep fdfs
```

FastDFS指令



上传指令

指令参数

```
fdfs_upload_file <config_file>
<local_filename> [storage_ip:port]
[store_path_index]
```

> **参数含义：**
>
> 1. <config_file>： 配置文件路径
> 2. <local_filename>： 本地文件路径
> 3. [storage_ip:port]： （可选参数）
> 4. [store_path_index]： （可选参数）

指令使用

```
[root@tracker fdfs]# fdfs_upload_file
/etc/fdfs/client.conf 上传的文件路径
```

```
[root@fastdfs fdfs]# fdfs_upload_file /etc/fdfs/client.conf ~/java/jdk-7u79-linux-i586.rpm
group1/M00/00/00/wKhyqVwnUUOAcgYHCCZp2Ahsb3g964.rpm
```

> **注意：**
>
> 上传文件后会返回文件在FastDFS中的唯一文件标识，即卷名+文件名

下载指令

指令参数

```
fdfs_download_file <config_file> <file_id>
[local_filename] [<download_offset>
<download_bytes>]
```

> **参数含义：**
>
> 1. <config_file>： 配置文件路径
> 2. <file_id>： 文件在FastDFS中的唯一文件标识，即卷名+文件名
> 3. [local_filename]： 文件下载地址

4. <download_offset>： （可选参数）文件下载开始时间
5. <download_bytes>： （可选参数）文件下载的字节数

指令使用

```
[root@tracker fdfs]# fdfs_download_file /etc/fdfs/client.conf group1/M00/00/00/wKhygVwnUUOAcgYHCCZp2Ahsb3g964.rpm /root/java/xxx.rpm
```

查看文件信息指令

指令参数

```
fdfs_file_info <config_file> <file_id>
```

参数含义：

1. <config_file>： 配置文件路径
2. <file_id>： 文件在FastDFS中的唯一文件标识，即卷名+文件名

指令使用

```
[root@tracker fdfs]# fdfs_file_info /etc/fdfs/client.conf group1/M00/00/00/wKhygVwnUUOAcgYHCCZp2Ahsb3g964.rpm
```

删除指令

指令参数

```
fdfs_delete_file <config_file> <file_id>
```

参数含义：

1. <config_file>： 配置文件路径
2. <file_id>： 文件在FastDFS中的唯一文件标识，即卷名+文件名

指令使用

```
[root@tracker fdfs]# fdfs_delete_file
/etc/fdfs/client.conf
group1/M00/00/00/wKhygVwnUUOAcgYHCCZp2Ahsb3g9
64.rpm
```

> **注意:**
>
> 删除指令使用后，文件在该卷中的所有备份都会被删除，因为卷内的存储节点会相互同步，故慎用。

实时学习反馈

1.FastDFS分布式文件系统中如何上传文件___。

A fdfs_delete_file

B fdfs_info_file

C fdfs_upload_file

D fdfs_download_file

2. FastDFS分布式文件系统中如何下载文件____。

A fdfs_delete_file

B fdfs_info_file

C fdfs_upload_file

D fdfs_download_file

答案

1=>C　2=>D

SpringBoot操作FastDFS

由GitHub大牛tobato在原作者YuQing与yuqih发布的JAVA客户端基础上进行了大量重构工作，并于GitHub上发布了FastDFS-Client1.26.5。

主要特性

1. 对关键部分代码加入了单元测试，便于理解与服务端的接口交易，提高接口质量
2. 将以前对byte硬解析风格重构为使用对象+注解的形式，尽量增强了代码的可读性
3. 支持对服务端的连接池管理
4. 支持上传图片时候检查图片格式，并且自动生成缩略图
5. 在SpringBoot当中自动导入依赖

实战开发

导入FastDFS依赖jar

```
<dependency>
    <groupId>com.github.tobato</groupId>
    <artifactId>fastdfs-client</artifactId>
    <version>1.26.5</version>
</dependency>
```

配置SpringBoot的入口类

```
1  package com.demo;
2
3  import
   com.github.tobato.fastdfs.FdfsClientConfig;
4  import
   org.springframework.boot.SpringApplication;
5  import
   org.springframework.boot.autoconfigure.Sprin
   gBootApplication;
6  import
   org.springframework.context.annotation.Enabl
   eMBeanExport;
7  import
   org.springframework.context.annotation.Impor
   t;
8  import
   org.springframework.jmx.support.Registration
   Policy;
9  //==========================================
   ==========================
10 //获取带有连接池的FastDFS Java客户端
11 @Import(FdfsClientConfig.class)
12 // 解决jmx重复注册bean的问题
13 @EnableMBeanExport(registration =
   RegistrationPolicy.IGNORE_EXISTING)
14 //==========================================
   ==========================
15 @SpringBootApplication
16 public class
   FastdfsSpringbootDriverApplication {
17     public static void main(String[] args) {
```

```
18
    SpringApplication.run(FastdfsSpringbootDriv
erApplication.class, args);
19      }
20  }
```

配置springboot的application.yml配置文件

```
1  ================================================
   ========================
2  # 分布式文件系统FDFS配置
3  ================================================
   ========================
4  fdfs:
5    so-timeout: 1501
6    connect-timeout: 601
7    thumb-image:                #缩略图生成参数
8      width: 150
9      height: 150
10   tracker-list:               #TrackerList参数,
   支持多个
11     - 192.168.66.100:22122
12     - 192.168.66.101:22122
13
```

上传文件操作

```
1  /**
2       * 测试springboot环境下的javaAPI对分布式文件
   系统的上传文件的操作
3       * @throws FileNotFoundException
4       */
5      @Test
```

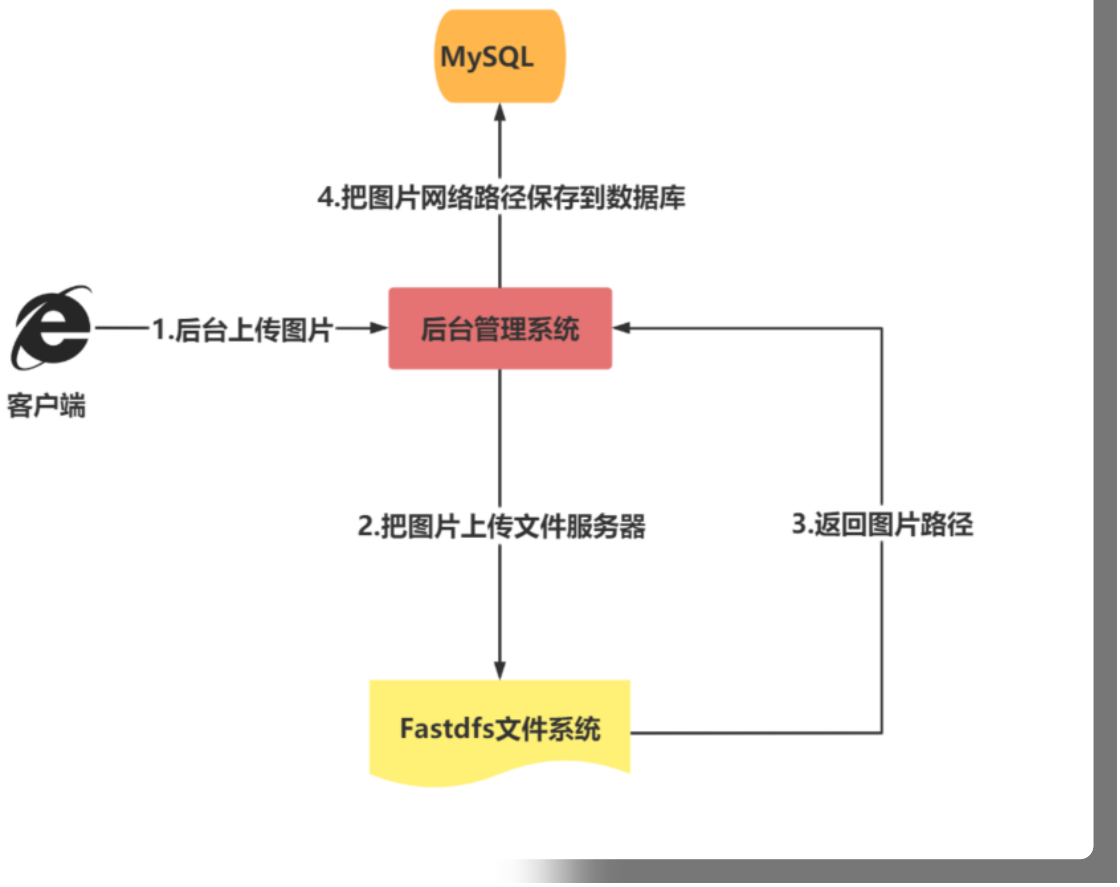
```
    public void testUpload() throws FileNotFoundException {
        //获取本地文件
        File file = new File("G:\\图片\\mei.jpg");
        //创建传输文件的输入流
        FileInputStream fileInputStream = new FileInputStream(file);
        //文件上传：参数一：传输文件内容的输入流；参数二：文件的size；参数三：文件扩展名；参数四：描述文件的元数据；返回值：上传文件在存储节点的唯一标识（卷名+文件名）
        StorePath storePath = fastFileStorageClient.uploadFile(fileInputStream, file.length(), "jpg", null);
        //将卷名与文件名一起打印

System.out.println(storePath.getFullPath());
        //将卷名与文件名分别打印

System.out.println(storePath.getGroup()+" | "+storePath.getPath());
    }
```

文件下载的操作

```
/**
     * 测试springboot环境下的javaAPI对分布式文件系统的下载文件的操作
     * @throws IOException
     */
    @Test
```

```
    public void testDownload() throws IOException {
        //下载文件：参数一：文件处于存储节点的卷名；参数二：文件在存储节点的文件名；参数三：下载的回调函数；返回值：文件内容的字节数组
        byte[] bytes = fastFileStorageClient.downloadFile("group1", "M00/00/00/wKhCZWICJcqAftV0AAChCwXlPdE133.jpg", new DownloadByteArray());
        //创建文件输出流，指定输出位置及文件名
        FileOutputStream fileOutputStream = new FileOutputStream("G:\\mei.jpg");
        //使用文件输出流将文件内容字节数组写出
        fileOutputStream.write(bytes);
        //刷新输出流
        fileOutputStream.flush();
        //关闭输出流
        fileOutputStream.close();
    }
```

文件上传_SpringBoot基于FastDFS实现



引入Thymeleaf视图解析器

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-thymeleaf</artifactId>
</dependency>
 <dependency>

<groupId>org.springframework.boot</groupId>
   <artifactId>spring-boot-starter-web</artifactId>
</dependency>
```

下载wangEditor富文本编辑器

为何选择 wangEditor

- 简洁、轻量级、文档齐全
- 万星项目 Github Star 1w+
- npm 周下载量 1w+

- CDN 月下载量百万+（来自 jsdelivr）
- QQ 群及时答疑
- 开源团队维护，非个人单兵作战

编写index页面

```
<!DOCTYPE html>
<html lang="en"
xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>wangEditor demo</title>
</head>
<body>

<div id="div1">
    <p>欢迎使用 <b>wangEditor</b> 富文本编辑器
</p>
</div>
<script type="text/javascript"
th:src="@{wangEditor.min.js}"
src="../static/wangEditor.min.js"></script>
```

```
13
14 <script type="text/javascript">
15     const E = window.wangEditor
16     const editor = new E('#div1')
17     //设置文件上传的参数名称
18     editor.config.uploadFileName = 'files'
19     // 配置 server 接口地址
20     editor.config.uploadImgServer =
'/upload/uploadMoreImage.do'
21     // 2M
22     editor.config.uploadImgMaxSize = 2 *
1024 * 1024
23     editor.config.uploadImgAccept = ['jpg',
'jpeg', 'png', 'gif', 'bmp', 'webp']
24     // 一次最多上传 5 个图片
25     editor.config.uploadImgMaxLength = 5
26     editor.create()
27 </script>
28
29 </body>
30 </html>
31
```

编写Controller接口

```
1 @RestController
2 @RequestMapping("/upload")
3 public class UoloadToFastDFSController {
4
5     //fastdfs存储节点的客户端对象
6     @Autowired
7     private FastFileStorageClient
fastFileStorageClient;
```

```
8
9       @GetMapping("/upload")
10      public void
uploadMoreImage(MultipartFile[] files){
11          //判断是否上传图片
12          if(files != null && files.length !=
0 ){
13              //遍历上传图片
14              for (MultipartFile multipartFile
: files) {
15                  //获取上传文件名
16                  String filename =
multipartFile.getOriginalFilename();
17                  //获取最后一个“.”的下标，并获取从
这个下标的下一个下标开始后的字符作为文件后缀
18                  String fileSuffix =
filename.substring(filename.lastIndexOf(".")
+ 1);
19                  StorePath storePath = null;
20                  try {
21                      //上传文件
22                      storePath =
fastFileStorageClient.uploadFile(multipartFi
le.getInputStream(),
multipartFile.getSize(), fileSuffix, null);
23                  } catch (IOException e) {
24                      e.printStackTrace();
25                  }
26                  //打印返回的文件在存储节点的唯一标
识
27
System.out.println(storePath.getFullPath());
```

```
28                  }
29              }
30          }
31
32  }
33
```

FastDFS集成Nginx

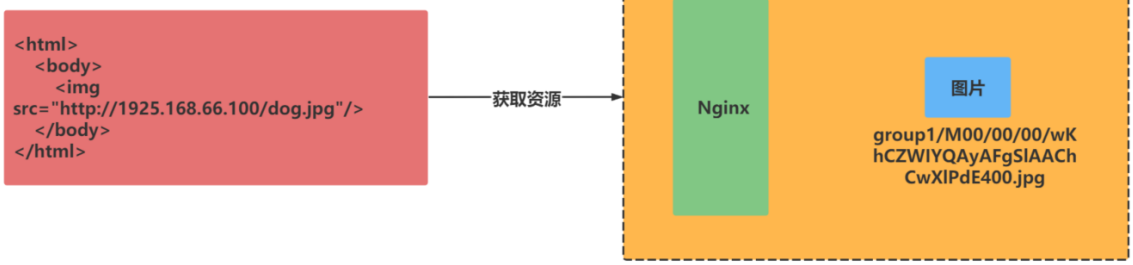
为分布式文件系统提供Http服务支持

Nginx服务器是一个高性能的web服务器与反向代理服务器。

FastDFS集成Nginx的2个原因

1. **为分布式文件系统提供Http服务支持**

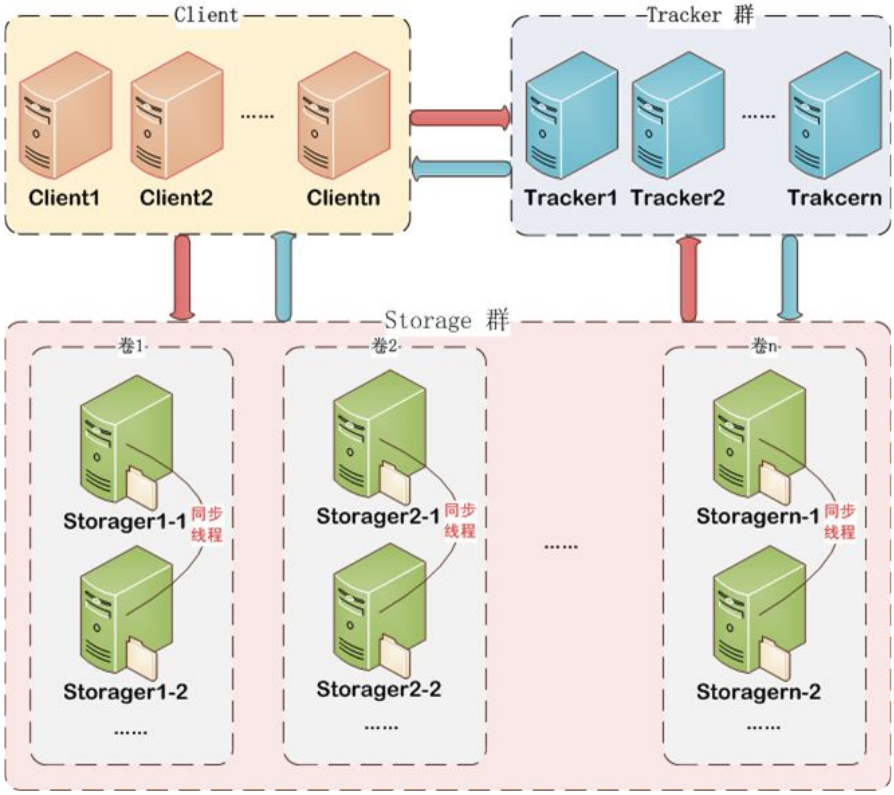
通过Nginx的web服务代理访问分布式文件系统的存储节点，从而实现通过http请求访问存储节点资源。



> **注意：**
>
> src 属性值图像文件的 URL。也就是引用该图像的文件的的绝对路径或相对路径。

1. **解决复制延迟问题**

由于FastDFS的同卷的存储节点之间需要同步，当文件尚未同步完成时，访问请求到达改节点，获取的数据将是未同步完的不完整数据，即为复制延迟问题。通过Nginx检测请求的存储节点的数据，若该存储节点的数据尚未同步完成，则将请求转发至数据的原存储节点，从而解决复制延迟问题。



实时学习反馈

1.FastDFS集成Nginx主要目的正确的是____。

A 为分布式文件系统提供Http服务支持

B 解决复制延迟问题

C 以上都是正确

答案

1=>C

FastDFS集成Nginx_环境搭建

下载Fastdfs的Nginx模块包

```
cd /usr/local
wget https://github.com/happyfish100/fastdfs-nginx-module/archive/V1.22.tar.gz
tar -zxvf V1.22.tar.gz
```

安装Nginx依赖文件

```
yum install -y gcc gcc-c++ zlib zlib-devel openssl openssl-devel pcre pcre-devel gd-devel epel-release
```

下载Nginx软件包

```
wget https://nginx.org/download/nginx-1.19.2.tar.gz
cd nginx-1.19.2/
```

配置Nginx服务器

```
#建立Makefile文件，检查Linux系统环境以及相关的关键属性。
./configure --add-module=/usr/local/fastdfs-nginx-module-1.22/src/
#编译项目，主要将gcc源代码编译成可执行的目标文件
make
#根据上一步骤编译完成的数据安装到预定的目录中。
make install
```

注意：

- –add-module：为nginx添加一个fastdfs-nginx-module模块，值为该模块在当前系统的路径
- –prefix：指定nginx安装位置

将Fastdfs软件包里面的http.conf和mime.types拷贝到/etc/fdfs目录下

```
cp /usr/local/src/fastdfs-6.06/conf/mime.types /etc/fdfs/
cp /usr/local/src/fastdfs-6.06/conf/http.conf /etc/fdfs/
```

配置Nginx的fastdfs模块，并编辑文件

```
#拷贝文件
[root@localhost opt]cp /usr/local/fastdfs-nginx-module-1.22/src/mod_fastdfs.conf /etc/fdfs/
[root@localhost fdfs] vim mod_fastdfs.conf
#保存日志目录
base_path=/data/fastdfs/storage
#tracker 服务器的 IP 地址以及端口号
```

```
tracker_server=192.168.66.100:22122
#文件url中是否有group 名
url_have_group_name = true
#存储路径
store_path0=/data/fastdfs/storage
group_count = 1                                    #设置组的个数
#然后在末尾添加分组信息，目前只有一个分组，就只写一个
[group1]
group_name=group1
storage_server_port=23000
store_path_count=1
store_path0=/data/fastdfs/storage
```

配置Nginx

```
server {
    listen       80;
    server_name  localhost;

    location ~ /group[1-3]/M00 {
        alias /data/fastdfs/storage/data;
        ngx_fastdfs_module;
    }
    # 根目录下返回403
    location = / {
         return 403;
     }
    # log file
    access_log  logs/img_access.log access;
}
```

启动Ningx服务

```
# 进入sbin目录
[root@tracker nginx]# cd sbin/
# 启动服务 -c：指定配置文件
[root@tracker sbin]# ./nginx -c
/usr/local/nginx/conf/nginx.conf
```

查看服务启动情况

```
[root@tracker sbin]# ps -ef | grep nginx
```

```
[root@tracker sbin]# ps -ef | grep nginx
root      10671      1  0 16:01 ?        00:00:00 nginx: master process ./nginx -c /usr/local/nginx/conf/nginx.conf
nobody    10672  10671  0 16:01 ?        00:00:00 nginx: worker process
root      10676   2002  0 16:05 pts/0    00:00:00 grep --color=auto nginx
[root@tracker sbin]#
```

启动追踪服务与存储节点服务

```
[root@tracker sbin]# fdfs_trackerd
/etc/fdfs/tracker.conf start
[root@tracker sbin]# fdfs_storaged
/etc/fdfs/storage.conf start
```

上传图片测试

将图片上传至linux系统后，使用指令上传至分布式文件系统

```
[root@tracker fdfs]# fdfs_upload_file
/etc/fdfs/client.conf /root/xxxxx.png
group1/M00/00/00/wKhyj1wrIUWAL5ASAAAfA8PiO7Y4
93.png
```

通过浏览器远程访问

```
http://192.168.66.100/group1/M00/00/00/wKhyj1
wrIfqAD3NFAAn1fNRE8_M976.png
```