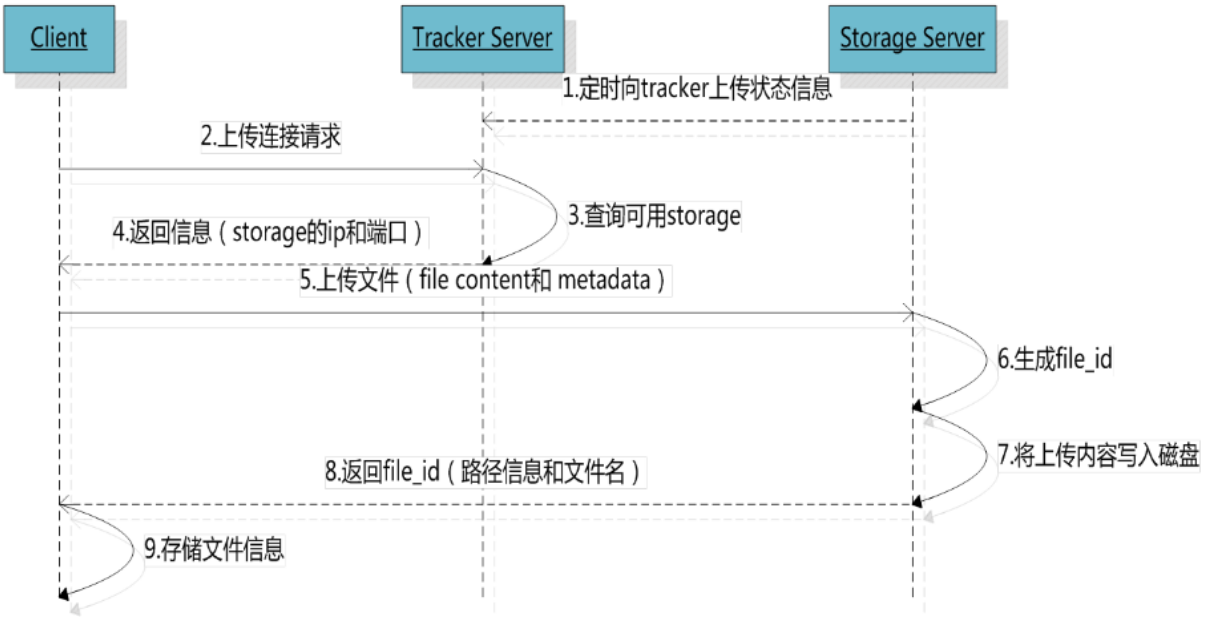
1. fastDFS

1.1 fastDFS介绍

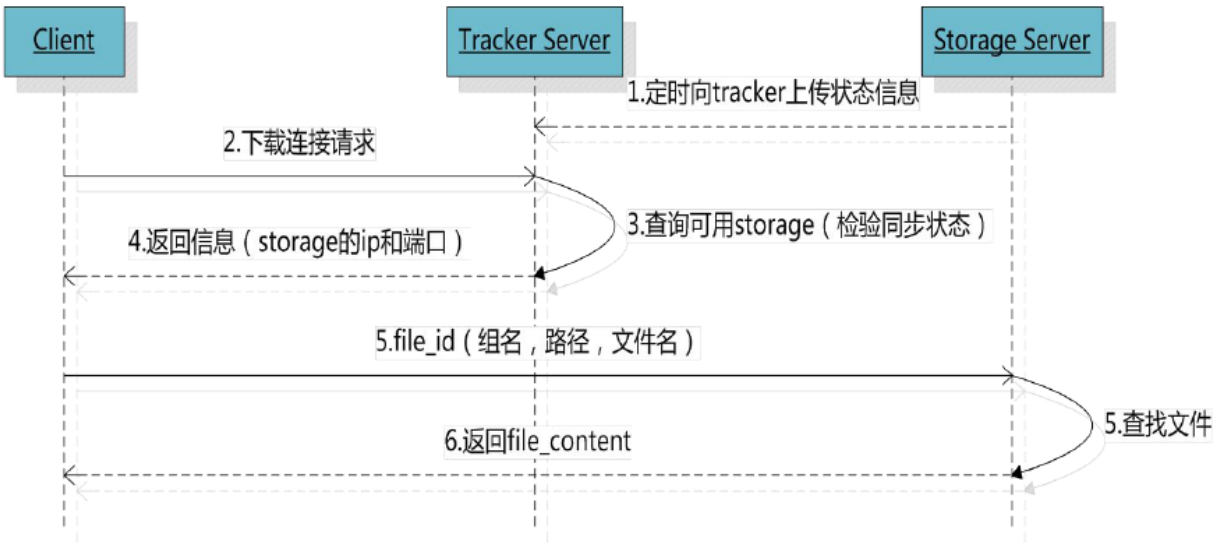
1. fastDFS概述

   > - 是用**c语言**编写的一款开源的分布式文件系统。
   >   - 余庆
   > - 为互联网量身定制，充分考虑了冗余备份、负载均衡、线性扩容等机制，注重高可用、高性能等指标
   >   - 冗余备份: 纵向扩容
   >   - 线性扩容: 横向扩容, 增加容量
   > - 可以很容易搭建一套高性能的文件服务器集群提供文件 **上传、下载** 等服务。
   >   - 应用场景:
   >   - 文件上传 -> 存储
   >   - 文件下载

2. fastDFS中的三个角色
   - 追踪器 (tracker) -> 守护进程
     - 管理者
     - 第一个启动
   - 存储节点 (storage) -> 守护进程
     - 可以理解为网络环境中可以存储文件的主机
     - 存储文件
     - 第二个启动
   - 客户端 - client
     - 程序猿写的
     - 发起上传请求, 完成上传操作
     - 发起下载请求, 将服务器文件下载到本地
     - 最后启动, 是一个普通的应用程序
3. fastDFS三个角色的关系



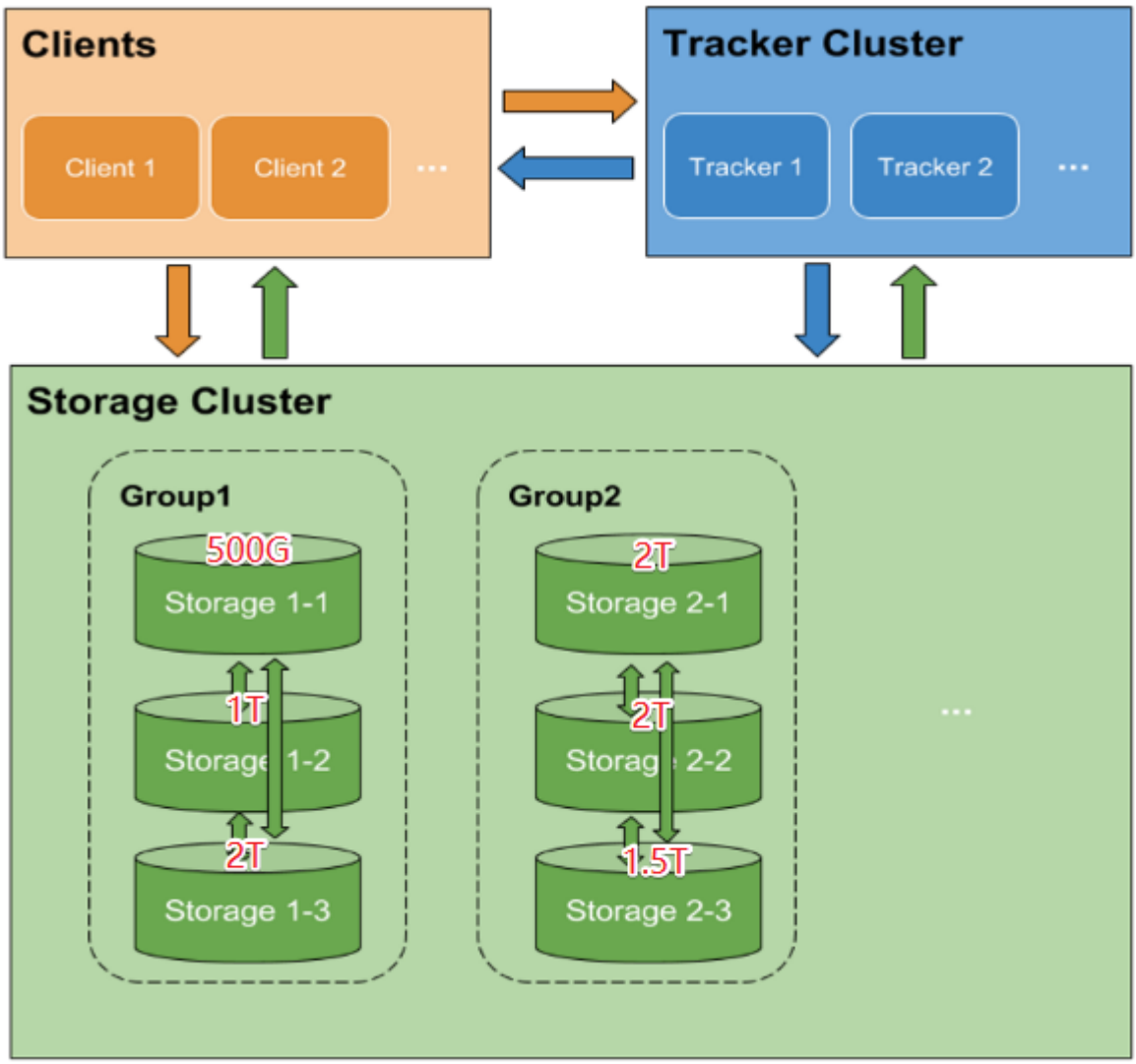
- 先启动追踪器
- 启动存储节点
  - 主动连接追踪器, 汇报当前存储节点的状态信息
  - 后边定时汇报状态
- 客户端程序启动, 连接追踪器, 发给上传请求
  - 客户端询问追踪, 看那个存储节点有足够的容量
  - 追踪查询存储节点信息
  - 将查到的节点信息发送给客户端
- 客户端通过得到的存储节点地址, 连接存储节点
- 将文件上传到存储节点上, 存储节点得到一个file_id, 并将其发送给客户端
- 客户端需要存储这个fileID, 下载的时候要用



- 先启动追踪器
- 启动存储节点

        - 主动连接追踪器, 汇报当前存储节点的状态信息
        - 后边定时汇报状态
    - 客户端程序启动, 连接追踪器, 发送下载请求
        - 客户端询问追踪, 看那个存储节点上有要下载的文件
        - 追踪查询存储节点信息
        - 将查到的节点地址发送给客户端
    - 客户端通过得到的存储节点地址, 连接存储节点
    - 将存储节点发送给客户端的文件, 保存到本地

4. fastDFS集群



    - tracker集群
        - 为了避免单点故障
        - 工作方式: 轮询
        - 集群方式: 修改配置文件
    - 存储节点的集群
        - 存储节点的管理:
            - 是以组的方式来管理的
        - 横向扩容 -> 添加新的分组, fastDFS容量增加了
            - 不同组的主机之间不通信

        - 各组的容量相加就是整体容量
    - 纵向扩容 -> 在现有的组中添加新的主机, 和同组主机之间互为备份关系
        - 同组主机中存储的内容相同
        - 同组主机之间是通信的
        - 当前组的容量按照最小的主机来算

1.2 fastDFS安装

1. 安装
    - libfastcommon-1.36.zip
        - fastDFS的基础库包
        - unzip libfastcommon-1.36.zip
        - cd libfastcommon-master
        - ./make.sh
        - sudo ./make.sh install
    - fastdfs-5.10.tar.gz
        - tar zxvf fastdfs-5.10.tar.gz
        - cd fastdfs-5.10
        - ./make.sh
        - sudo ./make.sh install
2. 配置

> fastDFS的配置文件默认存储目录: /etc/fdfs
>
> client.conf.sample storage.conf.sample storage_ids.conf.sample tracker.conf.sample

- tracker

```
bind_addr=
 - 追踪器对应的主机的IP地址
 - 如果不写，会自动绑定本机IP地址
 - 如果阿里云，这个地方空着就行了，否则有可能会无法启动
port=22122
 - 追踪器绑定的端口
 - 只要是一个空闲的没有被占用的端口就可以
base_path=/home/yuqing/fastdfs
 - 追踪器存储log日志或者一些进程文件相关的目录
 - 对应的路径必须要存在
```

- storage

```
group_name=group1
 - 当前存储节点所属的组
 - 横向扩容还是纵向扩容，是通过该属性控制的
bind_addr=
 - 存储节点的IP地址
 - 如果不写，会自动绑定本机IP地址
 - 如果阿里云，这个地方空着就行了，否则有可能会无法启动
port=23000
 - 客户端连接存储节点是时候使用的
```

```
  - 只要是一个空闲的没有被占用的端口就可以
base_path=/home/yuqing/fastdfs
  - 存储节点存储log日志的目录
  - 这个目录必须存在
store_path_count=2
  - 存储节点上，存储文件的路径个数
  - 一块硬盘对应一个存储路径就可以
store_path0=/home/yuqing/fastdfs
store_path1=/home/yuqing/fastdfs1
  - 存储文件的具体目录
tracker_server=192.168.247.131:22122
  - 连接的追踪器的地址
tracker_server=192.168.247.132:22122
  - 追踪去集群的声明方式
```

- client

```
base_path=/home/yuqing/fastdfs
 - 客户端写log日志的目录
tracker_server=192.168.0.197:22122
 -客户端要连接的追踪器的地址
```

1.3 fastDFS使用

1. 命令
   - tracker

```
# 启动
$ fdfs_trackerd /etc/fdfs/tracker.conf
# 停止
$ fdfs_trackerd /etc/fdfs/tracker.conf stop
# 重启
$ fdfs_trackerd /etc/fdfs/tracker.conf restart
```

   - storage

```
# 启动
$ fdfs_storaged /etc/fdfs/storage.conf
# 停止
$ fdfs_storaged /etc/fdfs/storage.conf stop
# 重启
$ fdfs_storaged /etc/fdfs/storage.conf restart
```

   - client

```
# 上传
$ fdfs_upload_file /etc/fdfs/client.conf 要上传的文件
# 下载
$ fdfs_download_file /etc/fdfs/client.conf FileID
$ fdfs_upload_file /etc/fdfs/client.conf a.yaml
group1/M00/00/00/wKj3g1v0zTuAW1zaAAAUrymD-Z449.yaml
 - group1 -> 文件上传到了哪个组
 - M00 -> store_path0
 - M01 -> store_path1
```

2. nginx

2.1 nginx介绍

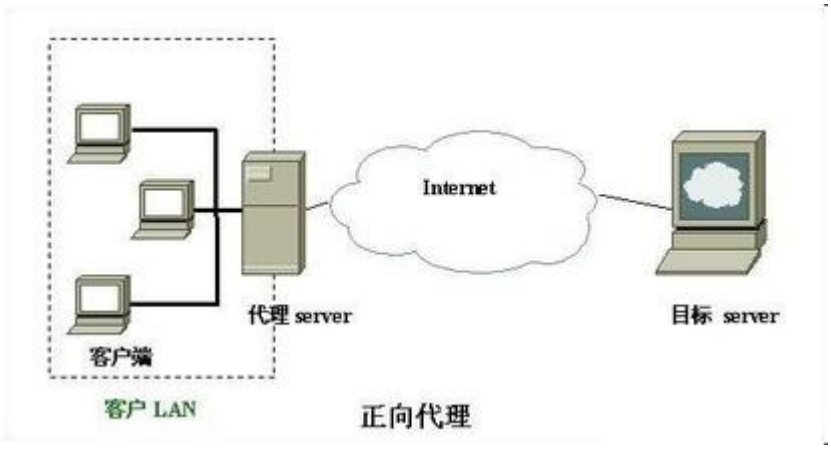
1. 能干什么
    - engine x
    - 俄罗斯人, 开源的框架
    - web服务器
        - 能够解析http协议
    - 反向代理服务器
    - 邮件服务器
        - pop3/smtp/imap
2. 优势

    - 更快
        - 高峰期(数以万计的并发时)nginx可以比其它web服务器更快的响应请求
    - 高扩展
        - **低耦合**设计的模块组成,丰富的第三方模块支持
    - 高可靠, 经过大批网站检验]
        - www.xunlei.com
        - www.sina.com.cn
        - www.126.com
        - www.taobao.com
    - 低内存消耗
        - 一般情况下,10000个非活跃的HTTP Keep-Alive连接在nginx中仅消耗 2.5M内存
    - 单机支持10万以上的并发连接
        - 取决于内存,10万远未封顶
    - 热部署
        - master和worker的分离设计,可实现7x24小时不间断服务的前提下升级nginx可执行文件
    - 最自由的BSD许可协议
        - BSD许可协议允许用户免费使用nginx, 修改nginx源码,然后再发布

2.2 正向/反向代理

1. 正向代理

> 正向代理是位于客户端和原始服务器之间的服务器，为了能够从原始服务器获取请求的内容，客户端需要将请求发送给代理服务器，然后再由代理服务器将请求转发给原始服务器，原始服务器接受到代理服务器的请求并处理，然后将处理好的数据转发给代理服务器，之后再由代理服务器转发发给客户端，完成整个请求过程。
>
> **正向代理的典型用途就是为在防火墙内的局域网客户端提供访问Internet的途径**，比如:
>
> - 学校的局域网
> - 单位局域网访问外部资源
>
> 正向代理作用: 帮助用户访问用户访问不了的网络, 正向代理服务器是为用户服务的.



2. 反向代理

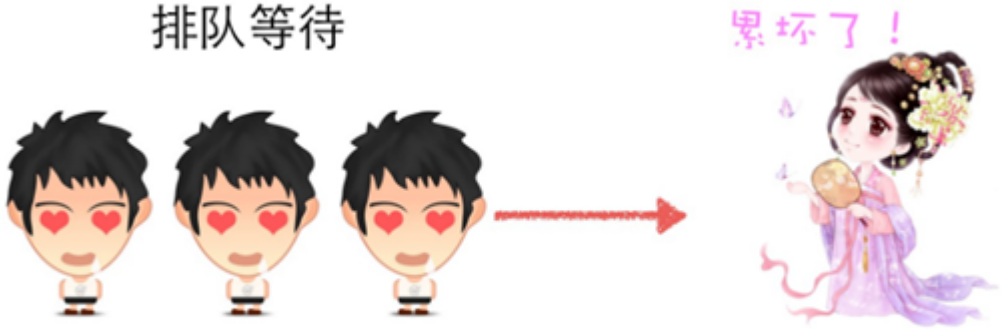
> 反向代理方式是指代理原始服务器来接受来自Internet的链接请求，然后将请求转发给内部网络上的原始服务器，并将从原始服务器上得到的结果转发给Internet上请求数据的客户端。那么顾名思义，反向代理就是位于Internet和原始服务器之间的服务器，对于客户端来说就表现为一台服务器，客户端所发送的请求都是直接发送给反向代理服务器，然后由反向代理服务器统一调配。

排队等待

客官　客官　客官　　青楼mm

一个青楼mm同意时刻只能接待一个用户。

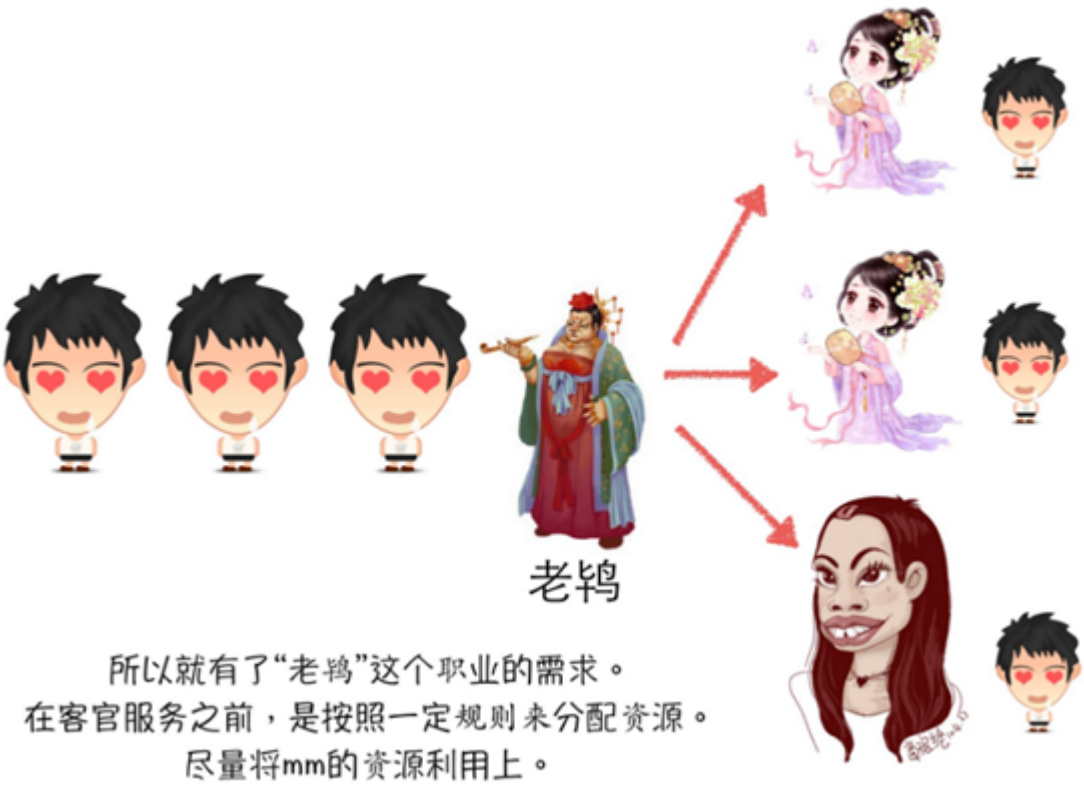
排队等待

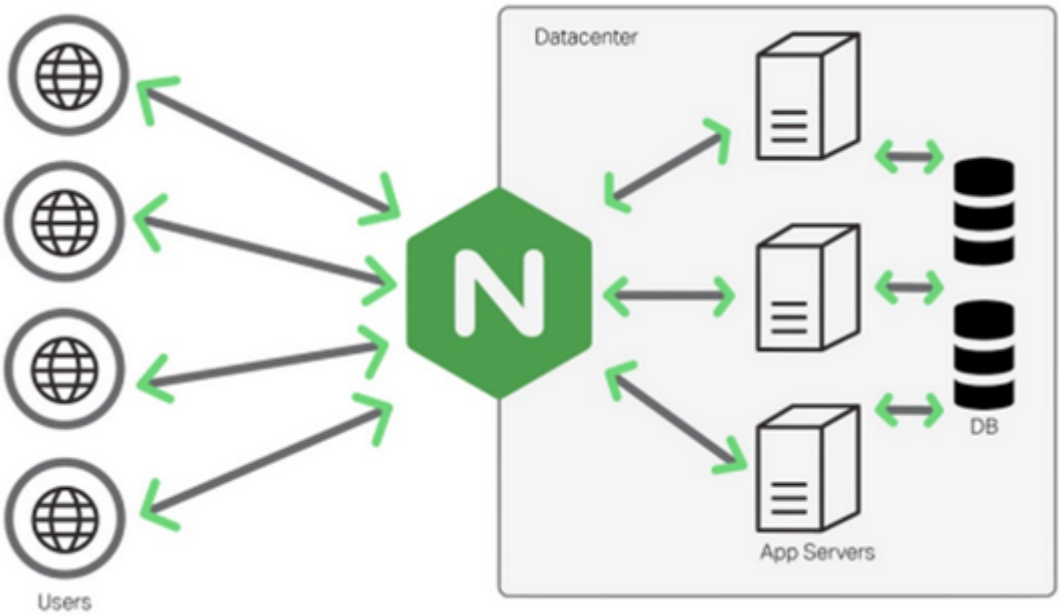


客官　客官　客官　　青楼mm

即使增加了服务人员，
但有时候依然无法平均分配用户。

青楼新来的mm





- 客户端发起请求, 发送给反向代理服务器
- 反向代理服务器将收到的请求转发给后台的web服务器, 反向代理server不处理请求
- web服务器处理收到的请求, 得到结果
- web服务器将节点发送给反向代理服务器
- 反向代理服务器将结果发送给客户端
- 反向代理服务器是为web服务器服务 的

2.3 nginx的安装

- 安装依赖的库

```
# openssl-1.0.1t.tar.gz
$ tar zxvf openssl-1.0.1t.tar.gz
$ cd openssl-1.0.1t
$ ./config  # 生成makefile文件
$ make
$ sudo make install
# pcre-8.40.tar.bz2 -> 解析正则表达式
$ tar jxvf pcre-8.40.tar.bz2
$ cd  pcre-8.40
$ ./configure
$ make
$ sudo make install
# zlib-1.2.11.tar.gz -> 压缩
$ tar zxvf zlib-1.2.11.tar.gz
$ cd zlib-1.2.11
$ ./configure
$ make
$ sudo make install
```

- nginx的安装

```
# nginx-1.10.1.tar.gz
$ tar zxvf nginx-1.10.1.tar.gz
$ cd nginx-1.10.1
########### 简易安装，不指定依赖的库 #############
$ ./configure
################################################
########### 工作环境中的安装######## #############
$ ./configure --with-openssl=openssl源码目录 --with-pcre=prcre的源码目录 --with-zlib=zlib的源码目录
################################################
$ make
$ sudo make install
```

```
itcast@ubuntu:nginx-1.10.1$ ls ../
fastdfs-nginx-module             nginx-1.10.1         openssl-1.0.1t         pcre-8.40          zlib-1.2.11
fastdfs-nginx-module_v1.16.tar.gz  nginx-1.10.1.tar.gz  openssl-1.0.1t.tar.gz  pcre-8.40.tar.bz2  zlib-1.2.11.tar.gz
itcast@ubuntu:nginx-1.10.1$ ./configure --with-openssl=../openssl-1.0.1t --with-pcre=../pcre-8.40 --with-zlib=../zlib-1.2.11
```

make过程中有一个错误

```
src/core/ngx_murmurhash.c: In function ‘ngx_murmur_hash2’:
src/core/ngx_murmurhash.c:37:11: error: this statement may fall through [-Werror=implicit-fallthrough=]
         h ^= data[2] << 16;
         ~~^~~~~~~~~~~~~~~~
src/core/ngx_murmurhash.c:38:5: note: here
     case 2:
     ^~~~
src/core/ngx_murmurhash.c:39:11: error: this statement may fall through [-Werror=implicit-fallthrough=]
         h ^= data[1] << 8;
         ~~^~~~~~~~~~~~~~~
src/core/ngx_murmurhash.c:40:5: note: here
     case 1:
     ^~~~
```

2.4 相关操作命令

```
# nginx安装完毕，默认的安装目录
/usr/local/nginx
$ cd /usr/local/nginx
$ ls
conf  html  logs  sbin
conf -> nginx需要的一些配置文件
html -> 存储静态资源的目录：xx.html，图片
        也可以创建自己的资源目录，在 /usr/local/nginx 下创建即可
logs -> 存储log日志的目录，错误日志文件名：error.log
sbin -> 存储了启动nginx的可执行程序
```

- nginx操作命令

```
# 进入到 /usr/local/nginx/sbin
# 启动
$ sudo ./nginx
# 关闭
$ sudo ./nginx -s stop   # 进程马上终止
$ sudo ./nginx -s quit   # 完成当前操作之后再终止
# 重新加载
$ sudo ./nginx -s reload  # 重新加载nginx的配置文件
# 可以将当前目录下的nginx添加软连接到 /usr/local/bin 下 , 这是$PATH中的一个路径
$ sudo ln -s /usr/local/nginx/sbin/nginx /usr/local/bin/nginx
```

- 测试nginx是否能使用
  - 先看nginx所对应的主机的IP地址
  - nginx启动
  - 找一台和nginx在同一网络的主机,通过浏览器访问 IP地址

2.4 静态网页的部署

- 准编写好的静态网页
- 将这些网页访问nginx的资源目录中
  - 默认的 -> html
  - 自己创建
- 场景举例

> 在Nginx服务器上进行网页部署, 实现如下访问:
>
> 在/usr/local/nginx/创建新的目录, yundisk用来存储静态网页

  - 访问地址: **http://192.168.80.254/login.html**

```
# http://192.168.80.254 -> 服务器的资源根目录
# 去资源根目录中找 login.html
# 如果想让nginx找到静态资源, 必须在配置文件中指定
# /usr/local/nginx/conf/nginx.conf
# nginx配置文件中每个location对应一个用户请求
location 请求指令
{
    root 资源根目录;
    index xx.html;  # -> 用户访问的是目录才有用
}
# nginx处理指令的提取:
url: http://192.168.80.254/login.html
 - 去掉协议: http
 - 去掉IP/域名
 - 去掉端口
 - 去掉尾部的文件名

```

```
location /
{
    root yundisk; # 资源根目录的指定
    index xx.html; # 在这不生效, 可以不写
}
```

```
$ sudo nginx -s reload
```

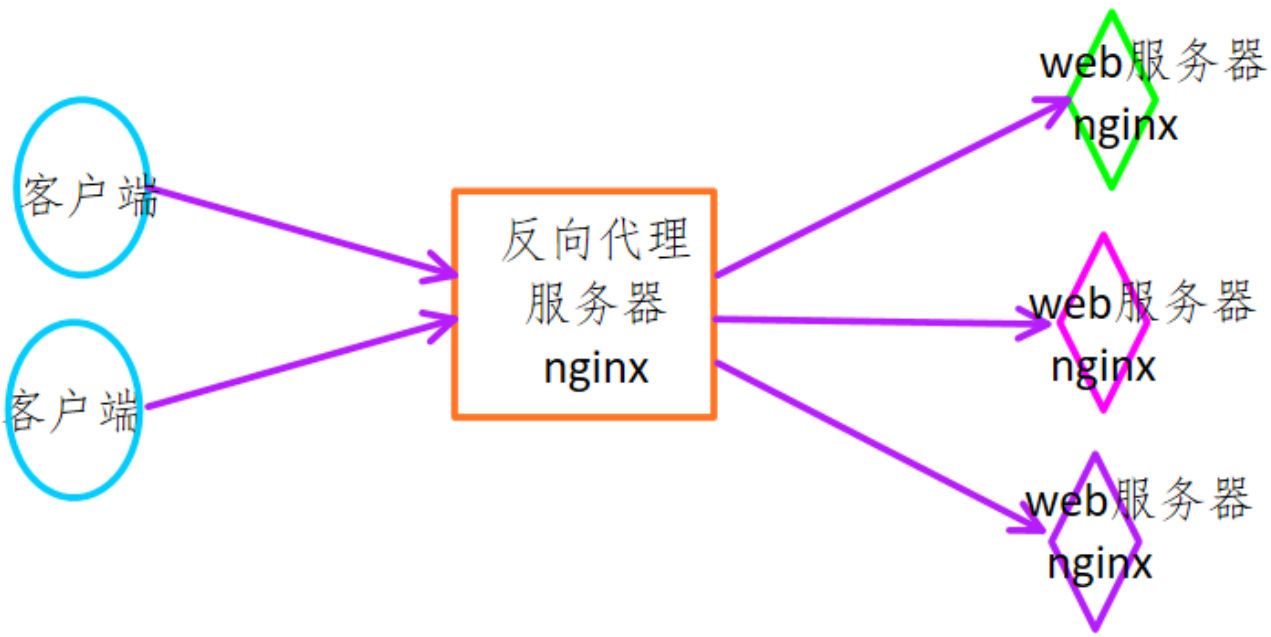
  - 访问地址: **http://192.168.80.254/hello/reg.html**
    - hello是什么?
      - / -> 资源根目录
      - hello是资源根目录的子目录
    - reg.html放到哪儿?
      - 放在hello子目录中
    - 如何添加location

```
http://192.168.80.254/hello/reg.html
# nginx处理指令的提取:
http://192.168.80.254/hello/reg.html
 - 去掉协议: http
 - 去掉IP/域名
 - 去掉端口
 - 去掉尾部的文件名
处理指令: /hello/
location /hello/
{
    root yundisk;
}
```

- 访问地址: http://192.168.80.254/upload/ 浏览器显示upload.html
  - 直接访问一个目录, 得到一默认网页
  - upload -> 资源目录中的子目录
  - upload.html 放到哪儿?
    - 放到upload子目录中
  - 处理指令

```
指令: /upload/
location /upload/
{
    root 资源根目录;
    index upload.html up1.html ;
}
```

2.5 负载均衡的配置



- 准备工作:
  - 客户端 -> window

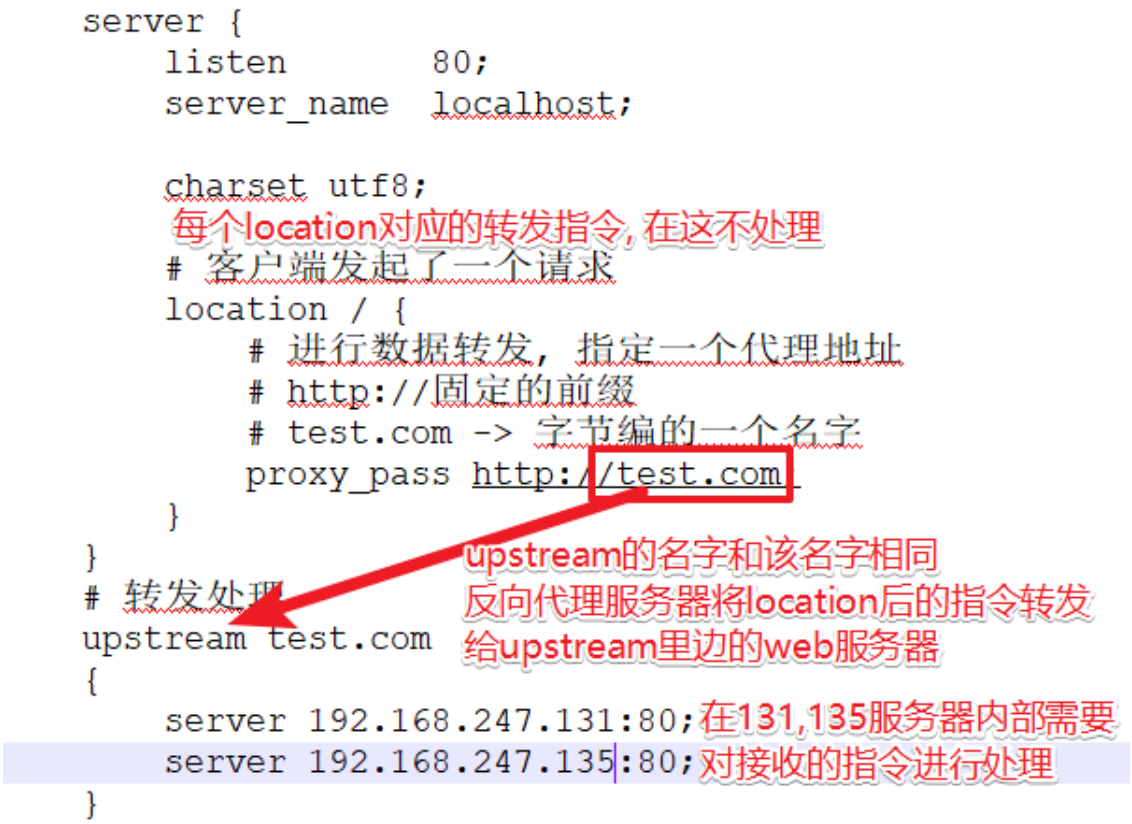
        - 电脑上的浏览器
    - 反向代理服务器 -> window
        - 将nginx的window版解压的没有中文的目录下
        - 需要修改对应的配置文件 -> conf/nginx.conf

```
http -> server

server {
        listen       80;
        server_name  localhost;

        charset utf8;

        # 客户端发起了一个请求
        location / {
            # 进行数据转发，指定一个代理地址
            # http://固定的前缀
            # test.com -> 字节编的一个名字
            proxy_pass http://test.com;
        }
        location /hello/ {
            # 进行数据转发，指定一个代理地址
            # http://固定的前缀
            # test.com -> 字节编的一个名字
            proxy_pass http://test.com;
        }
    }
    # 转发处理
    upstream test.com
    {
        server 192.168.247.131:80 weight=1;
        server 192.168.247.135:80 weight=5;
    }
}
```



- Web服务器
  - itcast : 192.168.247.131
  - robin: 192.168.247.135

3. nginx和fastDFS整合

3.1 安装

1. 在存储节点上安装Nginx, 将软件安装包拷贝到fastDFS存储节点对应的主机上

```
# nginx安装了fastDFs插件, 向让nginx和fastDFS存储节点进行数据通信
# 在一台主机上 同时 安装nginx 和 fastDFS{存储节点的角色}
```

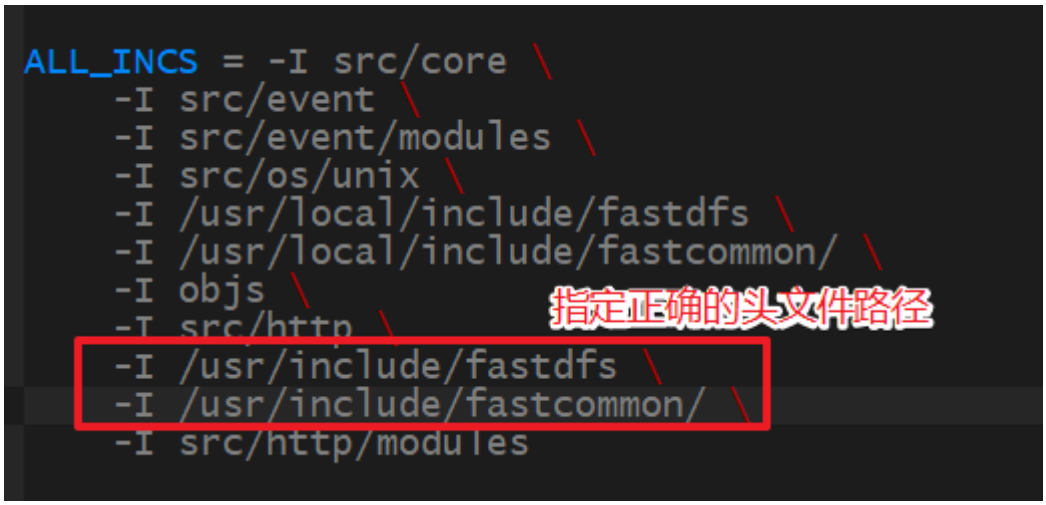
2. 在存储节点对应的主机上安装Nginx, 作为web服务器

```
# fastDFS插件源码包 -> fastdfs-nginx-module_v1.16.tar.gz
$ tar zxvf fastdfs-nginx-module_v1.16.tar.gz
$ cd fastdfs-nginx-module -> 里边有安装源码
# 进入nginx的源码安装目录
$ tree -L 1
.
├── auto
├── CHANGES
├── CHANGES.ru
├── conf
├── configure   -> 需要执行的文件
```

```
├── contrib
├── html
├── LICENSE
├── Makefile
├── man
├── objs
├── README
├── src
└── zlib-1.2.11
$ ./configure --add-module=fastdfs插件的源码根目录/src
$ make
$ sudo make install
```

3. make过程中有错误

```
# 1. fatal error: fdfs_define.h: 没有那个文件或目录
# 2. fatal error: common_define.h: 没有那个文件或目录
# 需要修改 objs/Makefile文件
正确的头文件路径需要通过find 进行搜索
$ find / -name fdfs_define.h
```



4. 安装成功, 启动Nginx, 发现没有 worker进程

```
$ sudo nginx
nginx: error while loading shared libraries: libpcre.so.1: cannot open shared object file: No such file or directory
# 解决方案
$  sudo find / -name libpcre.so
/usr/local/lib/libpcre.so
$ sudo vi /etc/ld.so.conf
    在这个文件中加一句话: /usr/local/lib/
$ sudo ldconfig
$ ps aux|grep nginx
root      85456  0.0  0.0  32960   444 ?        Ss   17:16   0:00 nginx: master process nginx
itcast    85478  0.0  0.0  21536  1060 pts/1    S+   17:18   0:00 grep --color=auto nginx
# 发现问题, 没有worker进程
# 去看 logs/error.log
```

3.2 解决问题

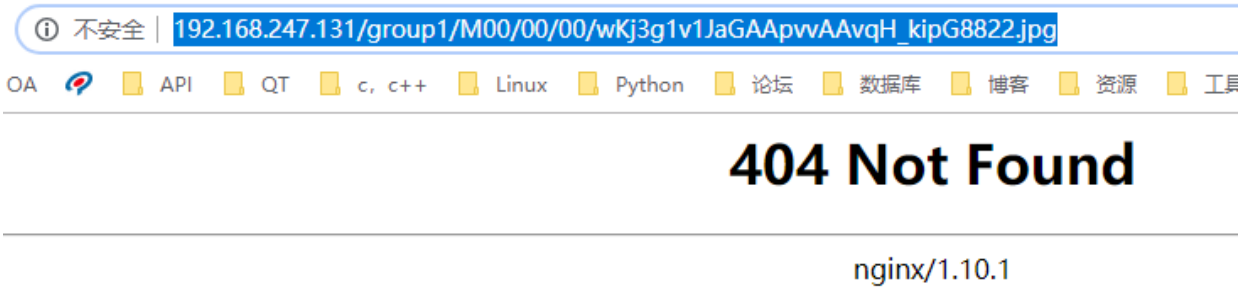
1. 拷贝文件 mod_fdfs.conf

```
# 进入fastDFS插件源码安装目录
itcast@ubuntu:src$ pwd
/home/itcast/package/nginx/fastdfs-nginx-module/src
itcast@ubuntu:src$ tree
.
├── common.c
├── common.h
├── config
├── mod_fastdfs.conf   -> 这就是不存在的文件
└── ngx_http_fastdfs_module.c
$ sudo cp ./mod_fastdfs.conf /etc/fdfs
```

2. 修改配置文件 mod_fdfs.conf - > 参考存储节点的配置文件进行修改

```
base_path=/home/itcast/myfastDFS/storage
 - 存储节点写log日志的目录
tracker_server=192.168.247.131:22122
 - 存储节点连接追踪器的地址
storage_server_port=23000
 - 存储节点绑定的端口
group_name=group1
 - 当前存储节点所属的组
url_have_group_name = true
 - 客户端访问fastdfs上存储的文件的地址的时候, url中是否包含组的名字
store_path_count=1
 - 存储节点上存储路径的个数
store_path0=/home/itcast/myfastDFS/storage
 - 具体的存储目录
```

3. 拷贝http.conf
    - 需要从fastdfs源码安装目录中找
        - conf/http.conf
        - `sudo cp http.conf /etc/fdfs`
4. 拷贝mime.types
    - 需要从nginx的源码安装目录中找
        - conf/mime.types
        - `sudo cp mime.types /etc/fdfs`
5. 浏览器访问




```
url: http://192.168.247.131/group1/M00/00/00/wKj3g1v1JaGAApvvAAvqH_kipG8822.jpg
在nginx端要处理一个指令
/group1/M00/00/00
location /group1/M00/00/00
{
}

M00 -> store_path0/data
location /group1/M00/
{
    # fastDFS存储文件的目录
    root /home/itcast/myfastDFS/storage/data;
    ngx_fastdfs_module; # 添加完成之后就可以和fastdfs存储节点通信
}
```