

腾讯课堂 零声学院 Darren QQ326873713

课程地址: <https://ke.qq.com/course/420945?tuin=137bb271>

该文档, 不包括fastdfs部署, fastdfs参考《1-3.1 FastDFS 单机版环境搭建》文档。

redis、nginx、mysql组件如果已经安装不需要重复安装, 只需要根据文档做必要的配置。

# 1 FastCGI

## 1.1 CGI

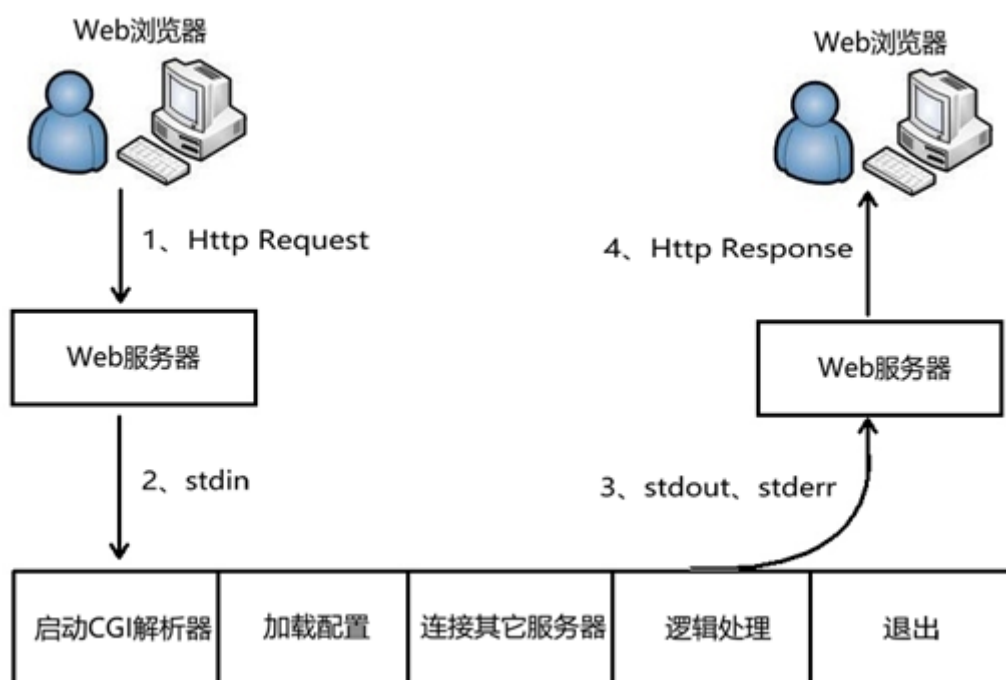
### 1.1.1 什么是CGI

通用网关接口(Common Gateway Interface、CGI)描述了客户端和服务程序之间传输数据的一种标准, 可以让一个客户端, 从网页浏览器向执行在网络服务器上的程序请求数据。

CGI独立于任何语言的, CGI 程序可以用任何脚本语言或者是完全独立编程语言实现, 只要这个语言可以在这个系统上运行。Unix shell script、Python、Ruby、PHP、perl、Tcl、C/C++和 Visual Basic 都可以用来编写 CGI 程序。

最初, CGI 是在 1993 年由美国国家超级电脑应用中心(NCSA)为 NCSA HTTPd Web 服务器开发的。这个 Web 服务器使用了 UNIX shell 环境变量来保存从 Web 服务器传递出去的参数, 然后生成一个运行 CGI 的独立的进程。

### 1.1.2 CGI处理流程



1.web服务器收到客户端(浏览器)的请求Http Request, 启动CGI程序, 并通过环境变量、标准输入传递数据

2.CGI进程启动解析器、加载配置(如业务相关配置)、连接其它服务器(如数据库服务器)、逻辑处理等

3.CGI进程将处理结果通过标准输出、标准错误, 传递给web服务器

4.web服务器收到CGI返回的结果, 构建Http Response返回给客户端, 并杀死CGI进程

web服务器与CGI通过环境变量、标准输入、标准输出、标准错误互相传递数据。在遇到用户连接请求:

- 先要创建CGI子进程，然后CGI子进程处理请求，处理完事退出这个子进程：fork-and-execute
- CGI方式是客户端有多少个请求，就开辟多少个子进程，每个子进程都需要启动自己的解释器、加载配置，连接其他服务器等初始化工作，这是CGI进程性能低下的主要原因。当用户请求非常多的时候，会占用大量的内存、cpu等资源，造成性能低下。

CGI使外部程序与Web服务器之间交互成为可能。CGI程序运行在独立的进程中，并对每个Web请求建立一个进程，这种方法非常容易实现，但效率很差，难以扩展。面对大量请求，进程的大量建立和消亡使操作系统性能大大下降。此外，由于地址空间无法共享，也限制了资源重用。

### 1.1.3 环境变量

GET请求，它将数据打包放置在环境变量QUERY\_STRING中，CGI从环境变量QUERY\_STRING中获取数据。

常见的环境变量如下表所示：

环境变数	含义
AUTH_TYPE	存取认证类型
CONTENT_LENGTH	由标准输入传递给CGI程序的数据长度，以bytes或字元数来计算
CONTENT_TYPE	请求的MIME类型
GATEWAY_INTERFACE	服务器的CGI版本编号
HTTP_ACCEPT	浏览器能直接接收的Content-types, 可以有HTTP Accept
header定义	
HTTP_USER_AGENT	递交表单的浏览器的名称、版本和其他平台性的附加信息
HTTP_REFERER	递交表单的文本的URL，不是所有的浏览器都发出这个信息，不要依赖它
PATH_INFO	传递给CGI程序的路径信息
QUERY_STRING	传递给CGI程序的请求参数，也就是用"?"隔开，添加在URL
后面的字串	
REMOTE_ADDR	client端的host名称
REMOTE_HOST	client端的IP位址
REMOTE_USER	client端送出来的使用者名称
REMOTE_METHOD	client端发出请求的方法(如get、post)
SCRIPT_NAME	CGI程序所在的虚拟路径，如/cgi-bin/echo
SERVER_NAME	server的host名称或IP地址
SERVER_PORT	收到request的server端口
SERVER_PROTOCOL	所使用的通讯协定和版本编号
SERVER_SOFTWARE	server程序的名称和版本

## 1.1.4 标准输入

环境变量的大小是有一定的限制的，当需要传送的数据量大时，储存环境变量的空间可能会不足，造成数据接收不完全，甚至无法执行CGI程序。

因此后来又发展出另外一种方法：POST，也就是利用I/O重新导向的技巧，让CGI程序可以由stdin和stdout直接跟浏览器沟通。

当我们指定用这种方法传递请求的数据时，web服务器收到数据后会先放在一块输入缓冲区中，并且将数据的大小记录在CONTENT\_LENGTH这个环境变量，然后调用CGI程序并将CGI程序的stdin指向这块缓冲区，于是我们就可以很顺利的通过stdin和环境变数CONTENT\_LENGTH得到所有的信息，再没有信息大小的限制了。

## 2.2 FastCGI

### 2.2.1 什么是FastCGI

快速通用网关接口(Fast Common Gateway Interface / FastCGI)是通用网关接口(CGI)的改进，描述了客户端和服务程序之间传输数据的一种标准。

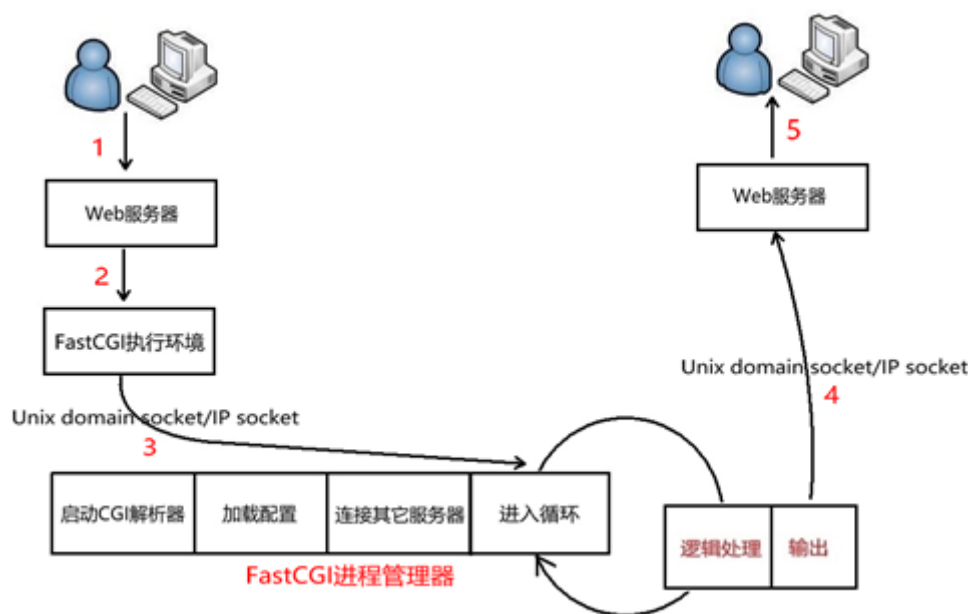
FastCGI致力于减少Web服务器与CGI程式之间互动的开销，从而使服务器可以同时处理更多的Web请求。与为每个请求创建一个新的进程不同，FastCGI使用持续的进程来处理一连串的请求。这些进程由FastCGI进程管理器管理，而不是web服务器。

nginx服务支持FastCGI模式，能够快速高效地处理动态请求。而nginx对应的FastCGI模块为：

- ngx\_http\_fastcgi\_module。

ngx\_http\_fastcgi\_module 模块允许将请求传递给 FastCGI 服务器。

### 2.2.2 FastCGI处理流程



1. Web 服务器启动时载入初始化FastCGI执行环境。例如IIS、ISAPI、apache mod\_fastcgi、nginx ngx\_http\_fastcgi\_module、lighttpd mod\_fastcgi。
2. FastCGI进程管理器自身初始化，启动多个CGI解释器进程并等待来自Web服务器的连接。启动FastCGI进程时，可以配置以ip和UNIX 域socket两种方式启动。
3. 当客户端请求到达Web 服务器时，Web 服务器将请求采用socket方式转发FastCGI主进程，FastCGI主进程选择并连接到一个CGI解释器。Web 服务器将CGI环境变量和标准输入发送到FastCGI子进程。

4. FastCGI子进程完成处理后将标准输出和错误信息从同一socket连接返回Web 服务器。当FastCGI子进程关闭连接时，请求便处理完成。
5. FastCGI子进程接着等待并处理来自Web 服务器的下一个连接。

由于FastCGI程序并不需要不断的产生新进程，可以大大降低服务器的压力并且产生较高的应用效率。它的速度效率最少要比CGI 技术提高 5 倍以上。它还支持分布式的部署，即FastCGI 程序可以在web 服务器以外的主机上执行。

CGI 是所谓的短生存期应用程序，FastCGI 是所谓的长生存期应用程序。FastCGI像是一个常驻(long-live)型的CGI，它可以一直执行着，不会每次都要花费时间去fork一次(这是CGI最为人诟病的fork-and-execute 模式)。

## 2.2.3 进程管理器管理：spawn-fcgi

### 2.2.3.1 什么是spawn-fcgi

Nginx不能像Apache那样直接执行外部可执行程序，但Nginx可以作为代理服务器，将请求转发给后端服务器，这也是Nginx的主要作用之一。其中Nginx就支持FastCGI代理，接收客户端的请求，然后将请求转发给后端FastCGI进程。

由于FastCGI进程由FastCGI进程管理器管理，而不是Nginx。这样就需要一个FastCGI进程管理器，管理我们编写FastCGI程序。

spawn-fcgi是一个通用的FastCGI进程管理器，简单小巧，原先是属于lighttpd的一部分，后来由于使用比较广泛，所以就迁移出来作为独立项目。

spawn-fcgi使用pre-fork 模型，功能主要是打开监听端口，绑定地址，然后fork-and-exec创建我们编写的FastCGI应用程序进程，退出完成工作。FastCGI应用程序初始化，然后进入死循环侦听socket的连接请求。

### 2.2.3.2 编译安装spawn-fcgi

spawn-fcgi源码包下载地址：<http://redmine.lighttpd.net/projects/spawn-fcgi/wiki>

编译和安装spawn-fcgi相关命令：

```
wget http://download.lighttpd.net/spawn-fcgi/releases-1.6.x/spawn-fcgi-1.6.4.tar.gz
tar -zxf spawn-fcgi-1.6.4.tar.gz
cd spawn-fcgi-1.6.4/
./configure
make
make install
```

如果遇到以下错误：

```
./autogen.sh: x: autoreconf: not found
```

因为没有安装automake工具，ubuntu用下面的命令安装即可：

```
apt-get install autoconf automake libtool
```

spawn-fcgi的帮助信息可以通过man spawn-fcgi或spawn-fcgi -h获得，下面是部分常用spawn-fcgi参数信息：

参数	含义
f	指定调用FastCGI的进程的执行程序位置
-a	绑定到地址addr
-p	绑定到端口port
-s	绑定到unix domain socket
-C	指定产生的FastCGI的进程数，默认为5(仅用于PHP)
-P	指定产生的进程的PID文件路径
-F	指定产生的FastCGI的进程数(C的CGI用这个)
-u和-g FastCGI	使用什么身份(-u用户、-g用户组)运行，CentOS下可以使用apache用户，其他的根据情况配置，如nobody、www-data等

## 2.2.4 软件开发套件：fcgi

### 2.2.4.1 编译安装fcgi

使用C/C++编写FastCGI应用程序，可以使用FastCGI软件开发套件或者其它开发框架，如fcgi。

fcgi下载地址：wget <https://fossies.org/linux/www/old/fcgi-2.4.0.tar.gz>

编译和安装fcgi相关命令：

```
wget https://fossies.org/linux/www/old/fcgi-2.4.0.tar.gz --no-check-certificate
tar -zxf fcgi-2.4.0.tar.gz
cd fcgi-2.4.1-SNAP-0910052249/
./configure

编译前在libfcgi/fcgio.cpp 文件上添加#include <stdio.h>
make
make install
```

如果编译出现下面问题：

fcgio.cpp: In destructor 'virtual fcgi\_streambuf::~~fcgi\_streambuf()':

fcgio.cpp:50:14: **error:** 'EOF' was not declared in this scope

```
    overflow(EOF);
```

^

fcgio.cpp: In member function 'virtual int fcgi\_streambuf::overflow(int)':

fcgio.cpp:70:72:\*\* error:\*\* 'EOF' was not declared in this scope

```
    if (FCGX_PutStr(pbase(), plen, this->fcgx) != plen) return EOF;
```

^

fcgio.cpp:75:14: **error:** 'EOF' was not declared in this scope

```
    if (c != EOF)
```

^

fcgio.cpp: In member function 'virtual int fcgi\_streambuf::sync()':

fcgio.cpp:86:18: **error:** 'EOF' was not declared in this scope

```
    if (overflow(EOF)) return EOF;
```

^

fcgio.cpp:87:41: **error:** 'EOF' was not declared in this scope

```
    if (FCGX_FFlush(this->fcgx)) return EOF;
```

^

fcgi.cpp: In member function 'virtual int fcgi\_streambuf::underflow()':

fcgi.cpp:113:35: **error:** 'EOF' was not declared in this scope

if (glen <= 0) return EOF;

**解决方法:** 在fcgi.h/fcgi.cpp 文件上添加#include <stdio.h>

## 2.2.4.2 测试程序

示例代码: vim fcgi.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include "fcgi_stdio.h"

int main(int argc, char *argv[])
{
    int count = 0;

    //阻塞等待并监听某个端口, 等待Nginx将数据发过来

    while (FCGI_Accept() >= 0)
    {
        //如果想得到数据, 需要从stdin去读, 实际上从Nginx上去读
        //如果想上传数据, 需要往stdout写, 实际上是给Nginx写数据

        printf("Content-type: text/html\r\n");
        printf("\r\n");
        printf("<title>Fast CGI Hello!</title>");
        printf("<h1>Fast CGI Hello!</h1>");
        //SERVER_NAME: 得到server的host名称
        printf("Request number %d running on host <i>%s</i>\n",
            ++count, getenv("SERVER_NAME"));
    }

    return 0;
}
```

编译代码:

```
gcc fcgi.c -o test -lfcgi
```

test就是其中一个针对client一个http请求的业务服务应用程序。需要在后台跑起来, 并且让spawn负责管理。

**记得先: ldconfig, 否则spawn-fcgi启动异常**

```
root@izbp1d83xkvoja33dm7ki2Z:~/0voice/cloud-drive/test# ldconfig
root@izbp1d83xkvoja33dm7ki2Z:~/0voice/cloud-drive/test# spawn-fcgi -a 127.0.0.1 -
p 8001 -f ./test
spawn-fcgi: child spawned successfully: PID: 24314
```

查看8001是否有被监听

```
root@izbp1h2l856zgoegc8rvnhZ:~/tuchuang# lsof -i:8001
COMMAND  PID USER  FD   TYPE    DEVICE  SIZE/OFF  NODE NAME
test      3370 root   0u    IPv4  5624558      0t0  TCP localhost:8001 (LISTEN)
```

### 2.2.4.3 有关Nginx的fcgi的配置

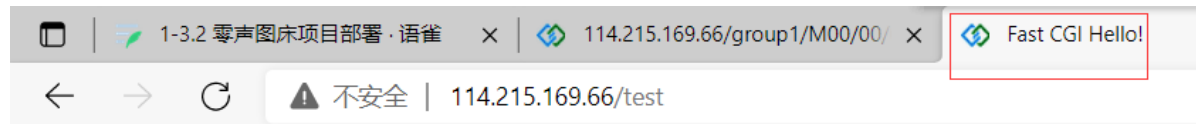
#监听用户的test请求，通过fastcgi\_pass交给本地8001端口处理  
#此时spwan-cgi已经将8001端口交给之前我们写好的test进程处理

```
location /test {
    fastcgi_pass 127.0.0.1:8001;
    fastcgi_index test;
    include fastcgi.conf;
}
```

记得nginx先重新加载配置文件

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

当Nginx收到<http://ip/test>请求时，比如<http://114.215.169.66/test>，会匹配到location test块，将请求传到后端的FastCGI应用程序处理：



**Fast CGI Hello!**

Request number 1 running on host *localhost*

## 3. MySQL

### 3.1 Ubuntu Linux安装MySQL

(1) 安装Mysql Server

```
apt-get install mysql-server
```

(2) 安装Mysql Client

```
apt-get install mysql-client
```

(3) 安装libmysqlclient

```
apt-get install libmysqlclient-dev
```

如果安装mysql-server时没有提示设置密码，得手动设置密码

步骤 1): 输入命令mysql -u root -p指定 root 用户登录 MySQL，输入后按回车键输入密码。如果没有配置环境变量，请在 MySQL 的 bin 目录下登录操作。

步骤 2): 修改密码 (比如root密码修改为123456)

```
use mysql;
update user set authentication_string=PASSWORD("123456") where user='root';
update user set plugin="mysql_native_password";
flush privileges;
quit;
```

然后 /etc/init.d/mysql restart 重启mysql, 再mysql -u root -p 登录测试密码是否正确。

## 3.2 Mysql启动/停止/重启

(1) Mysql启动

/etc/init.d/mysql start

(2) Mysql停止

/etc/init.d/mysql stop

(3) Mysql重启

/etc/init.d/mysql restart

## 3.3 创建用户

### 3.3.1 创建用户

```
CREATE USER username@host IDENTIFIED BY password;
```

说明:

- username: 你将创建的用户名
- host: 指定该用户在哪个主机上可以登陆, 如果是本地用户可用localhost, 如果想让该用户可以从任意远程主机登陆, 可以使用通配符%
- password: 该用户的登陆密码, 密码可以为空, 如果为空则该用户可以不需要密码登陆服务器

范例: CREATE USER 'darren'@'%' IDENTIFIED BY '123456';

### 3.3.2 授权

```
GRANT privileges ON databasename.tablename TO 'username'@'host' WITH GRANT OPTION;
```

说明:

- privileges: 用户的操作权限, 如SELECT, INSERT, UPDATE等, 如果要授予所有的权限则使用ALL
- databasename: 数据库名
- tablename: 表名, 如果要授予该用户对所有数据库和表的相应操作权限则可用表示, 如.\*

#设置darren用户拥有0voice\_tuchuang数据库所有权限

范例:

```
GRANT ALL PRIVILEGES ON 0voice_tuchuang.* TO 'darren'@'%';
FLUSH PRIVILEGES;
```

## 3.4 设置远程连接

### 3.4.1 修改配置文件

vi /etc/mysql/mysql.conf.d/mysqld.cnf

#注释blind-address

#bind-address=127.0.0.1(默认是没有注释的)



### 3.4.2 修改远程连接

#查看mysql所有用户

```
select user, host from mysql.user;
+-----+-----+
| user          | host          |
+-----+-----+
| root          | localhost    |
```

#update修改连接用户的host字段值为'%', 此处以root用户为例。  
update user set host='%' where user='root';

### 3.4.3 其他

防火墙原因(云服务器需要放开端口)

## 4. Redis

如果已经安装过redis不需要再重新安装。

### 4.1 Ubuntu Redis安装

```
#下载
git clone https://gitee.com/mirrors/redis.git
cd redis
git checkout 6.2.3
make
make install
```

编译 hiredis

```
cd ./deps/hiredis
make
make install
```

查看版本命令:

```
redis-server -v
```

显示: Redis server v=6.2.3 sha=00000000:0 malloc=jemalloc-5.1.0 bits=64  
build=77053994c60ea3c2

## 5 零声图床项目部署

解压 0voice\_tuchuang.tar.gz

```
tar -zxvf 0voice_tuchuang.tar.gz
```

服务器端和客户端程序都在0voice\_tuchuang目录。

```
.
├── 0voice_tuchuang_clear.sql
├── 0voice_tuchuang.sql 需要导入的数据库
├── bin_cgi 编译后的执行文件
├── common 通用源文件
├── conf 配置文件
├── fastdfs.sh 脚本
├── fcgi.sh 脚本
├── include 头文件
├── logs 程序日志文件
├── Makefile 编译的makefile
├── nginx.sh nginx启动的脚本
├── redis
├── redis.sh redis启动脚本
├── src_cgi
├── ** start.sh ** 服务器程序启动脚本
├── tc-front web客户端程序（编译后的文件）
└── test
```

## 5.1 客户端部署

### 5.1.1 安装nodejs

#### 1. 下载nodejs




官方: <https://nodejs.org/en/download/>

如图:

#### Downloads

Latest LTS Version: **16.14.0** (includes npm 8.3.1)

Download the Node.js source code or a pre-built installer for your platform, and start developing today.

LTS Recommended For Most Users	Current Latest Features	
 Windows Installer node-v16.14.0-x64.msi	 macOS Installer node-v16.14.0.pkg	 Source Code node-v16.14.0.tar.gz
Windows Installer (.msi)	32-bit	64-bit
Windows Binary (.zip)	32-bit	64-bit
macOS Installer (.pkg)	64-bit / ARM64	
macOS Binary (.tar.gz)	64-bit	ARM64
Linux Binaries (x64)	64-bit	
Linux Binaries (ARM)	ARMv7	ARMv8
Source Code	node-v16.14.0.tar.gz	

复制下载链接后, 从命令行下载:

```
wget https://nodejs.org/dist/v16.14.0/node-v16.14.0-linux-x64.tar.xz
```

## 2. 解压node安装包

```
xz -d node-v16.14.0-linux-x64.tar.xz
tar xf node-v16.14.0-linux-x64.tar
```

3. 创建符号链接，供直接从命令行访问无需输入路径（/root/0voice/node-v16.14.0-linux-x64路径是自己的路径，不要照抄）

```
ln -s /root/0voice/node-v16.14.0-linux-x64/bin/node /usr/local/bin/node
ln -s /root/0voice/node-v16.14.0-linux-x64/bin/npm /usr/local/bin/npm
```

打印版本：

```
~/0voice/node-v16.14.0-linux-x64/bin# node -v
v16.14.0
```

```
~/0voice/node-v16.14.0-linux-x64/bin# npm -v
8.3.1
```

## 5.1.2 获取web客户端的路径

```
root@izbp1h2l856zgoegc8rvnhZ:~/tuchuang/0voice_tuchuang# cd tc-front/
root@izbp1h2l856zgoegc8rvnhZ:~/tuchuang/0voice_tuchuang/tc-front# pwd
/root/tuchuang/0voice_tuchuang/tc-front
```

/root/tuchuang/0voice\_tuchuang/tc-front 该路径配置到nginx.conf，在5.3.2节时统一配置。

## 5.2 Redis配置

0voice\_tuchuang提供了redis的启动脚本，在conf/redi.conf目录，对应的启动脚本为redis.sh。

## 5.3 创建mysql

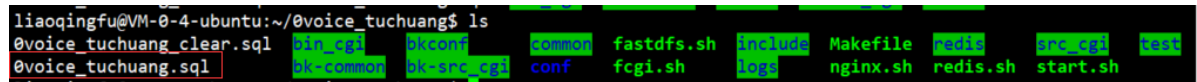
导入0voice\_tuchuang.sql文件

```
milo@VM-0-15-ubuntu:/home/milo/project$ mysql -uroot -p #登录mysql
```

.....

```
mysql>
```

```
mysql> source /root/tuchuang/0voice_tuchuang/0voice_tuchuang.sql; #导入数据库，具体看自己存放的路径
```



查看数据库创建情况：

```
mysql> use 0voice_tuchuang;
```

```
Database changed
```

```
mysql> show tables;
```

```
+-----+
| Tables_in_0voice_tuchuang |
+-----+
| file_info                  |
| share_file_list            |
| share_picture_list         |
```

```
| user_file_count |
| user_file_list |
| user_info      |
+-----+
6 rows in set (0.01 sec)
```

## 5.3 服务端部署

### 5.3.1 配置conf/cfg.json

具体的ip地址根据自己机器进行配置。

**修改配置 conf/cfg.json（具体的ip、port、账号等是自己的，不能照抄）**

```
{
  "redis":
  {
    "ip": "127.0.0.1",
    "port": "6379"
  },
  "mysql":
  {
    "ip": "127.0.0.1",
    "port": "3306",
    "database": "0voice_tuchuang",
    "user": "root",
    "password": "123456"
  },
  "dfs_path":
  {
    "client": "/etc/fdfs/client.conf"
  },
  "web_server":
  {
    "ip": "114.215.169.66",
    "port": "80"
  },
  "storage_web_server":
  {
    "ip": "114.215.169.66",
    "port": "80"
  }
}
```

#### 设置权限

```
chmod a+x start.sh
chmod a+x fastdfs.sh
chmod a+x fcgi.sh
chmod a+x nginx.sh
chmod a+x redis.sh
```

可以用一条命令执行 `chmod a+x *.sh`

## 5.3.2 配置nginx

配置fastcgi程序，这里是修改nginx.conf，可以直接使用课程提供的nginx.conf(在0voice\_tuchuang目录)覆盖自己本地的nginx.conf。

```
user root;
#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 1024;
}

http {
    include mime.types;
    default_type application/octet-stream;

    #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;
        index index.html index.htm default.htm default.html;
        #root /home/liaoqingfu/tc-front;
        root /root/tuchuang/0voice_tuchuang/tc-front;

        autoindex off;

        #access_log logs/host.access.log main;
        # 指定允许跨域的方法，*代表所有
        add_header Access-Control-Allow-Methods *;

        # 预检命令的缓存，如果不缓存每次会发送两次请求
        add_header Access-Control-Max-Age 3600;
        # 带cookie请求需要加上这个字段，并设置为true
        add_header Access-Control-Allow-Credentials true;
```

```
# 表示允许这个域跨域调用（客户端发送请求的域名和端口）
# $http_origin动态获取请求客户端请求的域 不用*的原因是带cookie的请求不支持*号
add_header Access-Control-Allow-Origin $http_origin;

# 表示请求头的字段 动态获取
add_header Access-Control-Allow-Headers
$http_access_control_request_headers;

#charset koi8-r;

#access_log logs/host.access.log main;

location / {
    root /root/tuchuang/0voice_tuchuang/tc-front;
    index index.html index.htm;
    try_files $uri $uri/ /index.html;
}
location ~/(group|[0-9])/M([0-9])([0-9]) {
    ngx_fastdfs_module;
}
location /test {
    fastcgi_pass 127.0.0.1:8001;
    fastcgi_index test;
    include fastcgi.conf;
}
location /api/login{
    fastcgi_pass 127.0.0.1:10000;
    include fastcgi.conf;
}

location /api/reg{
    fastcgi_pass 127.0.0.1:10001;
    include fastcgi.conf;
}

location /api/upload{
    fastcgi_pass 127.0.0.1:10002;
    include fastcgi.conf;
}

location /api/md5{
    fastcgi_pass 127.0.0.1:10003;
    include fastcgi.conf;
}

location /api/myfiles{
    fastcgi_pass 127.0.0.1:10004;
    include fastcgi.conf;
}

location /api/dealfile{
    fastcgi_pass 127.0.0.1:10005;
    include fastcgi.conf;
}

location /api/sharefiles{
    fastcgi_pass 127.0.0.1:10006;
    include fastcgi.conf;
}
```

```

    }
    location /api/dealsharefile{
        fastcgi_pass 127.0.0.1:10007;
        include fastcgi.conf;
    }

    location /api/sharepic{
        fastcgi_pass 127.0.0.1:10008;
        include fastcgi.conf;
    }
    #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;
    }
}
}

```

## 停止并启动nginx

```
/usr/local/nginx/sbin/nginx -s stop
```

```
/usr/local/nginx/sbin/nginx
```



#### 异常处理

⚠ 不安全 | 114.215.169.66

## 403 Forbidden

nginx/1.16.1

1. 查看nginx日志: `tail -f /usr/local/nginx/logs/error.log`
2. 查看到访问的时候报错: `tail -f /usr/local/nginx/logs/error.log`
3. 原因: root权限的问题, 可以先在nginx.conf加入**user root;权限**。

```
1 114.215.169.66 x 2 114.215.169.66 x +
user root;
#user nobody;
worker_processes 1;
```

然后重新: `/usr/local/nginx/sbin/nginx -s reload`



## 5.2.3 部署服务器程序

在0voice\_tuchuang目录下进行

### 1 编译

```
make clean
ldconfig
make
```

### 2 运行

./start.sh

服务器运行成功状态

```
root@izbp1h2l856zgoegc8rvnhz:~/tuchuang/0voice_tuchuang# ./start.sh

===== fastdfs =====
waiting for pid [24088] exit ...
pid [24088] exit.
waiting for pid [24081] exit ...
pid [24081] exit.
storage start success ...
tracker start success ...

===== fastCGI =====
CGI 程序已经成功关闭, bye-bye ...
登录: spawn-fcgi: child spawned successfully: PID: 24618
注册: spawn-fcgi: child spawned successfully: PID: 24620
上传: spawn-fcgi: child spawned successfully: PID: 24622
MD5: spawn-fcgi: child spawned successfully: PID: 24624
MyFile: spawn-fcgi: child spawned successfully: PID: 24626
DealFile: spawn-fcgi: child spawned successfully: PID: 24628
ShareList: spawn-fcgi: child spawned successfully: PID: 24630
DealShare: spawn-fcgi: child spawned successfully: PID: 24632
CGI 程序已经成功启动 ^_<...

===== nginx =====
ngx_http_fastdfs_set pid=24635
nginx stop success ...
ngx_http_fastdfs_set pid=24638
nginx start success ...

===== redis =====
Redis stopping ...
### 通过 redis.pid文件 方式关闭进程 ###

Redis server pid = 24420
Redis server stop success!!!
Redis starting ...
Redis server start success!!!
***** Redis server PID: [ 24652 ] *****
***** Redis server PORT: [ 6379 ] *****
```

### 3 查看日志

```
~/tuchuang/0voice_tuchuang# ls logs/cgi/2022/02/  
dealfile-24.log dealsharefile-24.log md5-24.log myfiles-24.log sharefiles-  
24.log sharepic-24.log upload-24.log
```

通过tail命令查看日志。

比如

- 注册业务 **tail -f logs/cgi/2022/02/reg-24.log** (需要注册后才有该文件)

```
root@iZbp1h2l856zgoegc8rvnhZ:~/tuchuang/0voice_tuchuang# tail -f logs/cgi/2022/02/reg-24.log  
[2022-02-24 16:41:03]--[src_cgi/reg_cgi.c:235]--buf =  
{ "email": "", "firstPwd": "e10adc3949ba59abbe56e057f20f883e", "nickName": "zhangli", "phone": "", "us  
erName": "lili" }  
[2022-02-24 16:41:03]--[src_cgi/reg_cgi.c:127]--mysql_user = root, mysql_pwd = 123456, mysql_db  
= 0voice_tuchuang  
[2022-02-24 16:41:03]--[src_cgi/reg_cgi.c:140]--user_name = lili, nick_name = zhangli, pwd =  
e10adc3949ba59abbe56e057f20f883e, tel = , email =  
[2022-02-24 16:41:03]--[src_cgi/reg_cgi.c:141]--root 123456 0voice_tuchuang
```