# nginx集群（二）

目录

[nginx集群（二） 1](#_Toc531190435)

[一、任务调度指令 1](#_Toc531190436)

[二、Nginx rewrite 2](#_Toc531190437)

[1、基本案例 3](#_Toc531190438)

[2、break与last的区别； 4](#_Toc531190439)

[3、redirect与permanent区别 5](#_Toc531190440)

[三、gzip压缩提升网站速度； 7](#_Toc531190441)

[四、LNMP环境安装MySql 9](#_Toc531190442)

[1、 查看旧的mysql情况 9](#_Toc531190443)

[2、下载mysql组件 9](#_Toc531190444)

[3、上传mysql组件 9](#_Toc531190445)

[4、安装mysql的4个组件 10](#_Toc531190446)

[5、启动mysql 10](#_Toc531190447)

[6、查看mysql登录密码 11](#_Toc531190448)

[7、 登录mysql操作 11](#_Toc531190449)

[8、 修改登录密码 11](#_Toc531190450)

[9、 控制mysql服务 12](#_Toc531190451)

[10、数据库存储的目录 12](#_Toc531190452)

[11、数据库配置文件 12](#_Toc531190453)

[12、给mysql设置字符集 12](#_Toc531190454)

[五、LNMP环境安装PHP 13](#_Toc531190455)

[1、安装 14](#_Toc531190456)

[2、配置 17](#_Toc531190457)

[3、控制服务 19](#_Toc531190458)

[4、修改配置 20](#_Toc531190459)

[5、编译 php与nginx整合 20](#_Toc531190460)

[（1）编辑文件： 20](#_Toc531190461)

[（2）编辑nginx配置文件(找到server项目设置如下内容)： 21](#_Toc531190462)

[4、nginx上面部署TP框架 22](#_Toc531190463)

[5、nginx上面部署TP框架实现伪静态 23](#_Toc531190464)

[6、安装dedecms软件 24](#_Toc531190465)

[（1）新建一个虚拟主机； 24](#_Toc531190466)

[（2）上传dedecms的软件，进行解压 25](#_Toc531190467)

[（3）把解压后的软件拷贝到虚拟主机目录里面 25](#_Toc531190468)

[（4）在浏览器输入域名开始安装测试。 25](#_Toc531190469)

[六、nginx的负载均衡的配置 27](#_Toc531190470)

[1、配置的原理图 27](#_Toc531190471)

[2、具体的配置 27](#_Toc531190472)

[3、连接池里面的其他指令 29](#_Toc531190473)

[七、window下安装NGINX 29](#_Toc531190474)

[1、NGINX安装 29](#_Toc531190475)

[2、nginx的虚拟主机的配置 30](#_Toc531190476)

[3、nginx支持php 31](#_Toc531190477)

# 一、任务调度指令

linux系统运行过程中有时候需要定制调度任务，例如在每天晚上12点对数据库备份一次，每两个小时检查下系统有没有需要发送的邮件并发送等等，这时候就可以使用任务调度指令

定义：规定系统在指定的时间完成指定的任务。

设置：

> crontab –e //编辑任务调度指令

> crontab –l //查看任务调度指令

(root用户设置)

**#进入crontab编辑界面，会打开vim编辑你的工作。**

**语法：**

**\* \* \* \* \* 执行的任务**

**比如每分钟执行一个命令：echo ‘nihao’ >> /demo/nihao.txt**

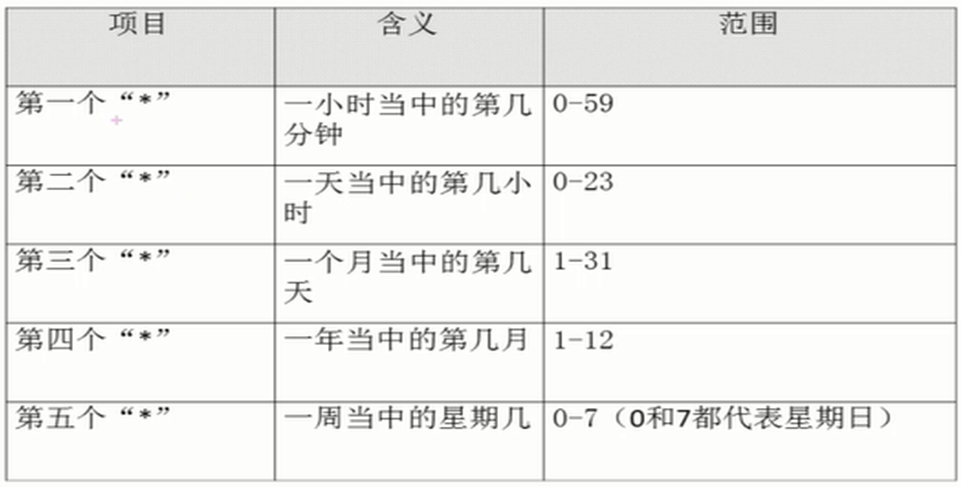
要注意：5个星表示要执行的时间，



比如每天晚上凌晨开始执行一个php文件。

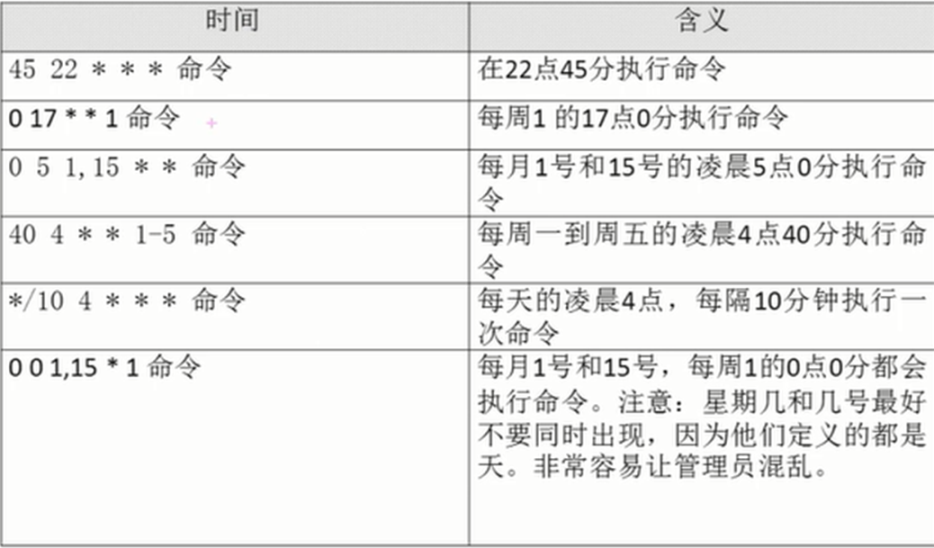
0 0 \* \* \* /usr/local/php/php.exe /var/www/shop/index.php

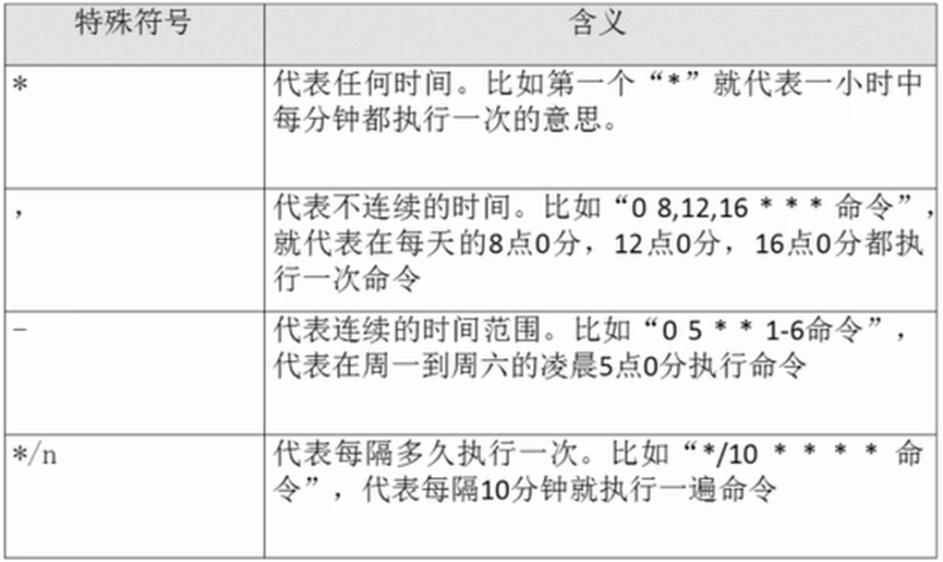
星号参数的解释说明：



比如5月份15号5点50要操作一个命令： 50 5 15 5 \* 命令

比如每周1的5点10分执行一个命令： 10 5 \* \* 1 命令





#!/bin/bash 命令的集合

# 二、Nginx rewrite

<http://www.abc.com/index.html> <http://www.abc.com/index.php>

Nginx rewrite主要功能是实现URL地址重写，需要PCER的支持，前面已经安装。

语法：

rewrite 匹配url 目标url [flag],应用位置 server location if段中。

rewrite 是实现URL重写的关键指令，根据匹配url部分的内容，重定向到目录url上，结尾是flag标记；

比如：

rewrite ^/(.\*) <http://www.abc.com/$1> permanent;

|  |  |
| --- | --- |
| Flag标记符号 | 说明 |
| last | 本条规则匹配完成后，继续向下匹配新的locationURL规则 |
| break | 本条规则匹配完成即终止，不再匹配后面的任何规则 |
| redirect | 返回302临时重定向，浏览器地址栏会显示跳转后的URL地址 |
| permanent | 返回301永久重定向，浏览器地址栏会显示跳转后的url地址 |

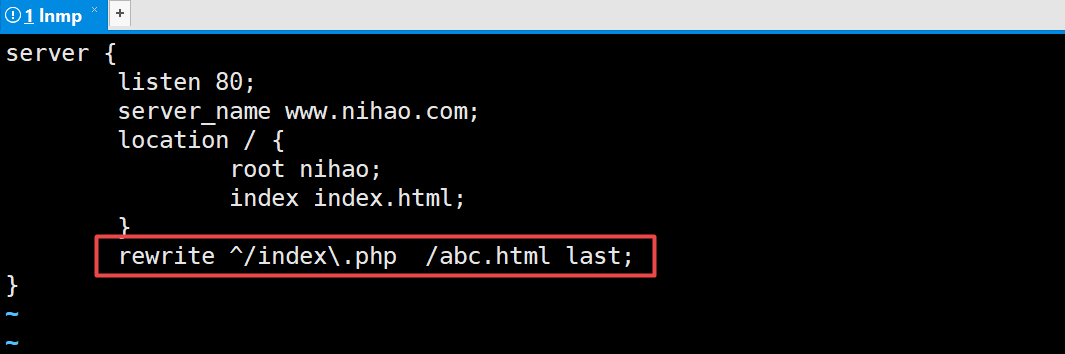
在以上的flag标记中，last和break用来实现url重写，浏览器地址栏的URL地址不变，单在服务器端访问的程序及路径发生了变化。redirect和permanent用来实现URL跳转，浏览器地址栏会显示跳转后的URL地址。

## 1、基本案例

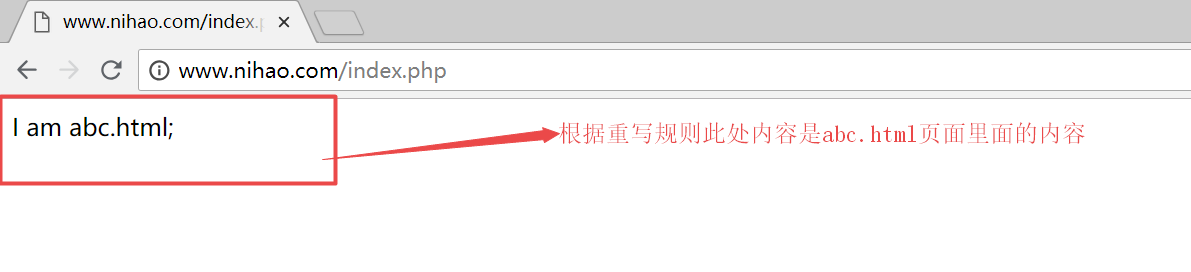
案例1：简单重写，访问index.php 重写到abc.html

rewrite ^/index\.php /abc.html last;

使用 [www.nihao.com域名来测试，打开extra目录下面的nihao.conf](http://www.nihao.com域名来测试，打开extra目录下面的nihao.conf)配置文件。

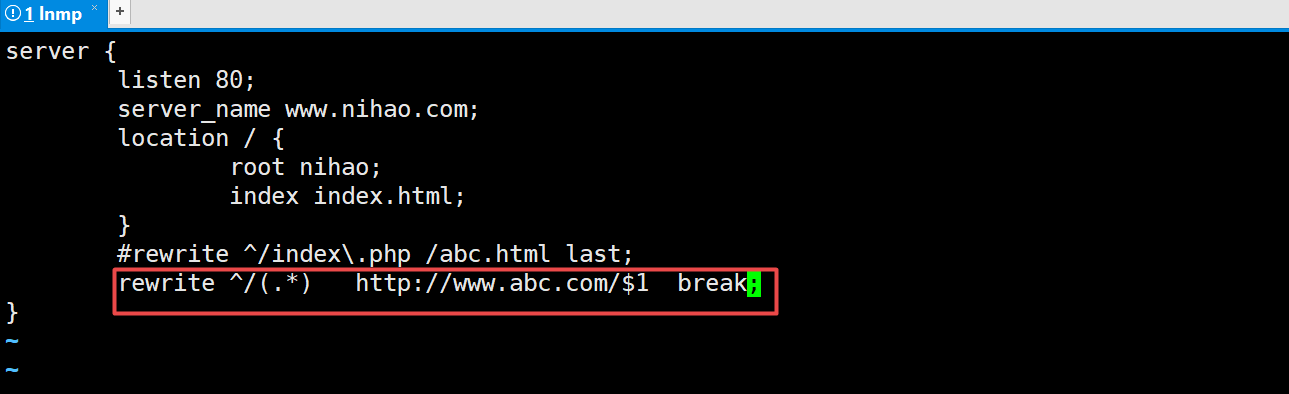


访问效果如下：



案例2：实现访问 <http://www.nihao.com>时跳转到<http://www.abc.com>

需要对[www.nihao.com域名对应配置文件，进行配置，打开extra/nihao.conf](http://www.nihao.com域名对应配置文件，进行配置，打开extra/nihao.conf)文件配置内容如下：



思考：也就是说我们访问[www.nihao.com/abc.php](http://www.nihao.com/abc.php) 则变成访问www.abc.com/abc.php

rewrite ^/(.\*) <http://www.abc.com/$1> break;

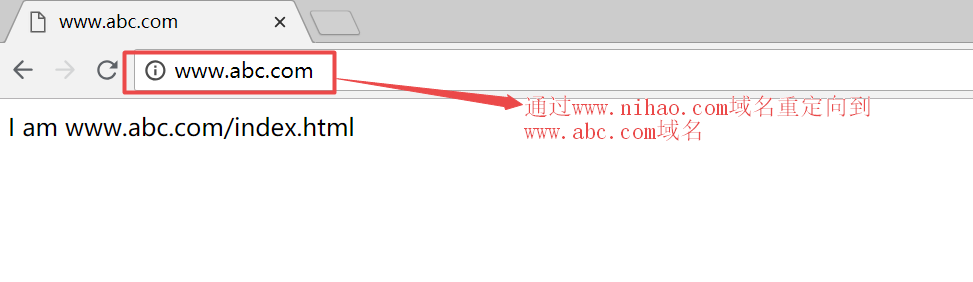
点号，在正则里面表示除了换行符以外的任何字符，

星号，表示0到多个。

使用.\*来表示任何字符。

$1符号，表示前面(.\*)里面的内容。

访问效果如下：



## 2、break与last的区别；

location ~ ^/break {

rewrite ^/break /test break;

}

location ~ ^/last {

rewrite ^/last /test last;

}

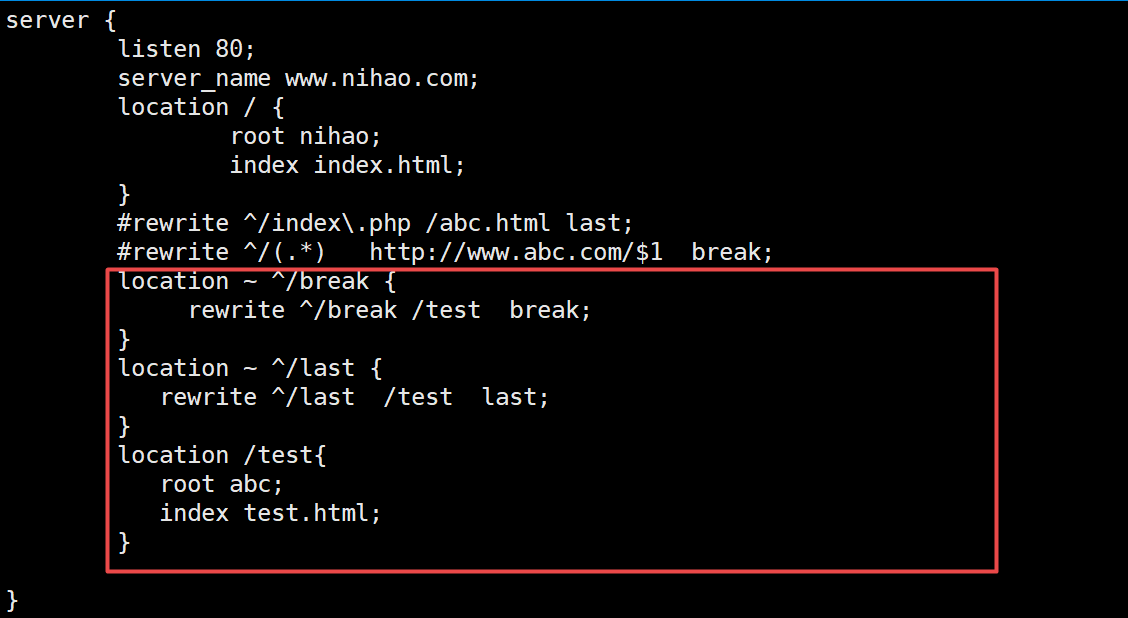
location /test{

root abc;

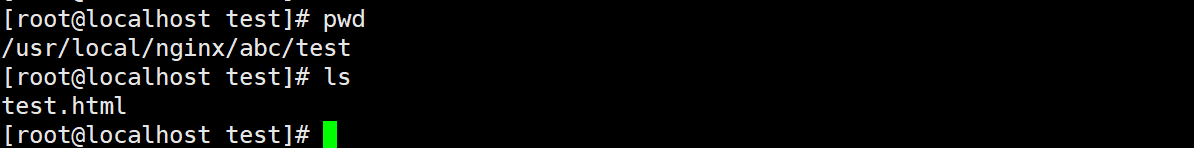
index 123.html;

}

注意 当访问 域名/test url时，实际上是返回的 abc/test/123.html页面的内容；



在/usr/local/nginx/abc目录下面新建一个test目录，在里面新建一个test.html文件；



本条规则匹配完成后，继续向下匹配新的locationURL规则

## 3、redirect与permanent区别

301永久重定向，浏览器会记住，

比如a.com网站 301到 b.com网站，

浏览器中输入a.com时，就不请求a.com了，就直接请求b.com网站了；

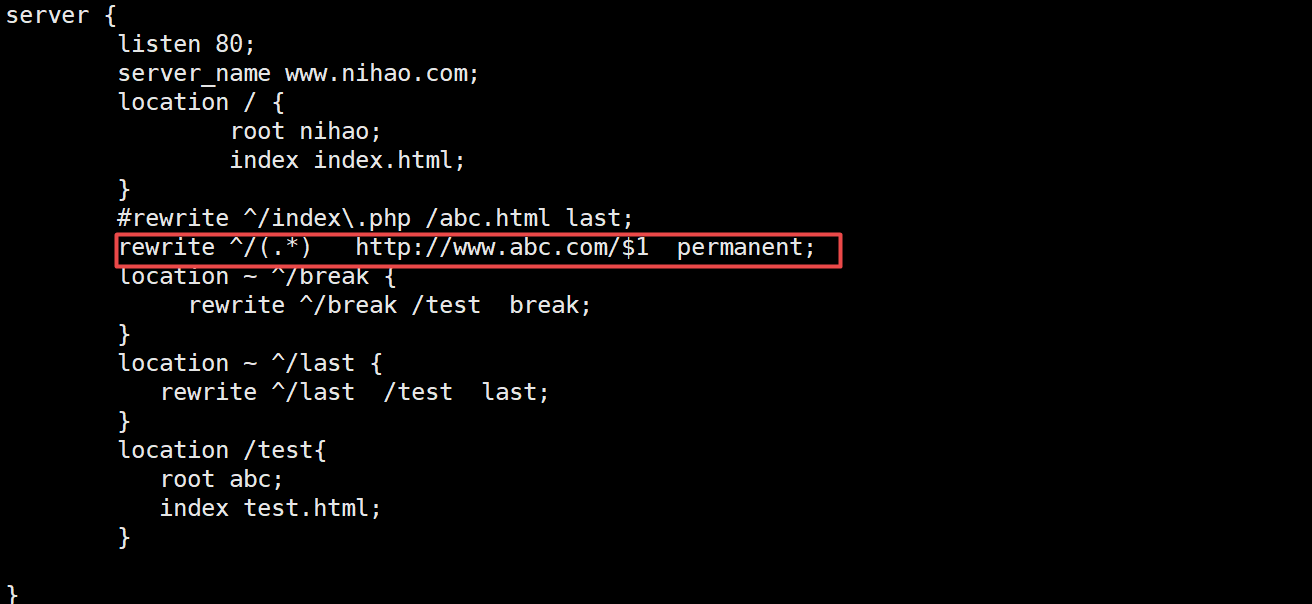
302临时重定向，浏览器不记住，

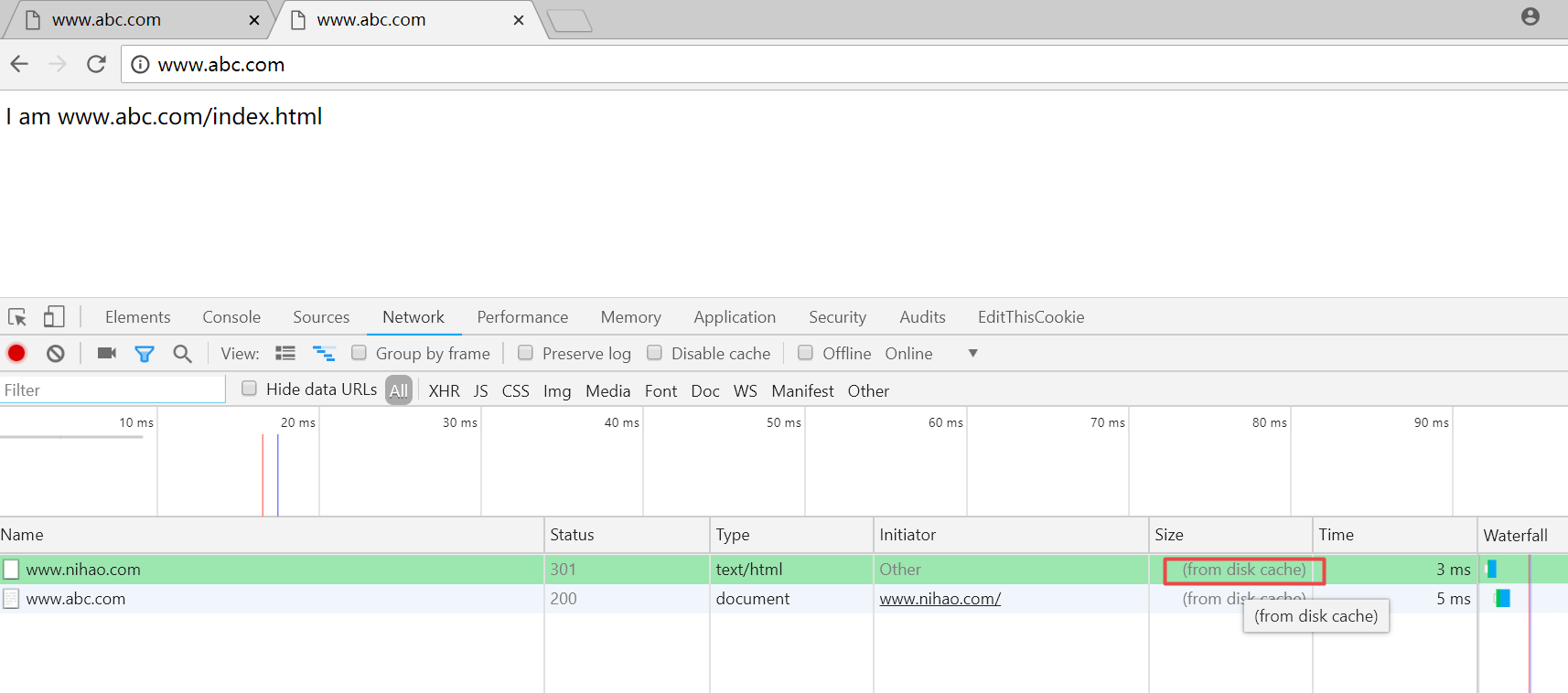
比如a.com网站 302到 b.com网站，

浏览器中输入a.com时，还是请求a.com网站，根据a.com网站响应的location内容，再去请求b.com网站；

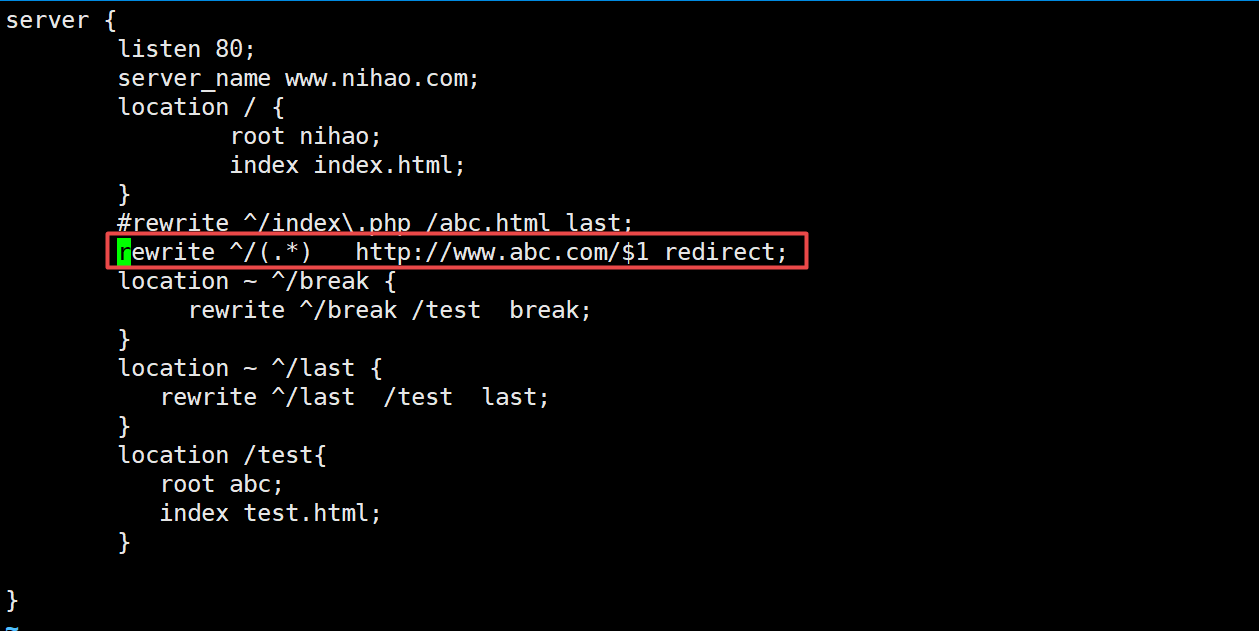
**301重定向；**

**rewrite ^/(.\*)** [**http://www.nihao.com/$1**](http://www.nihao.com/$1) **permanent;**



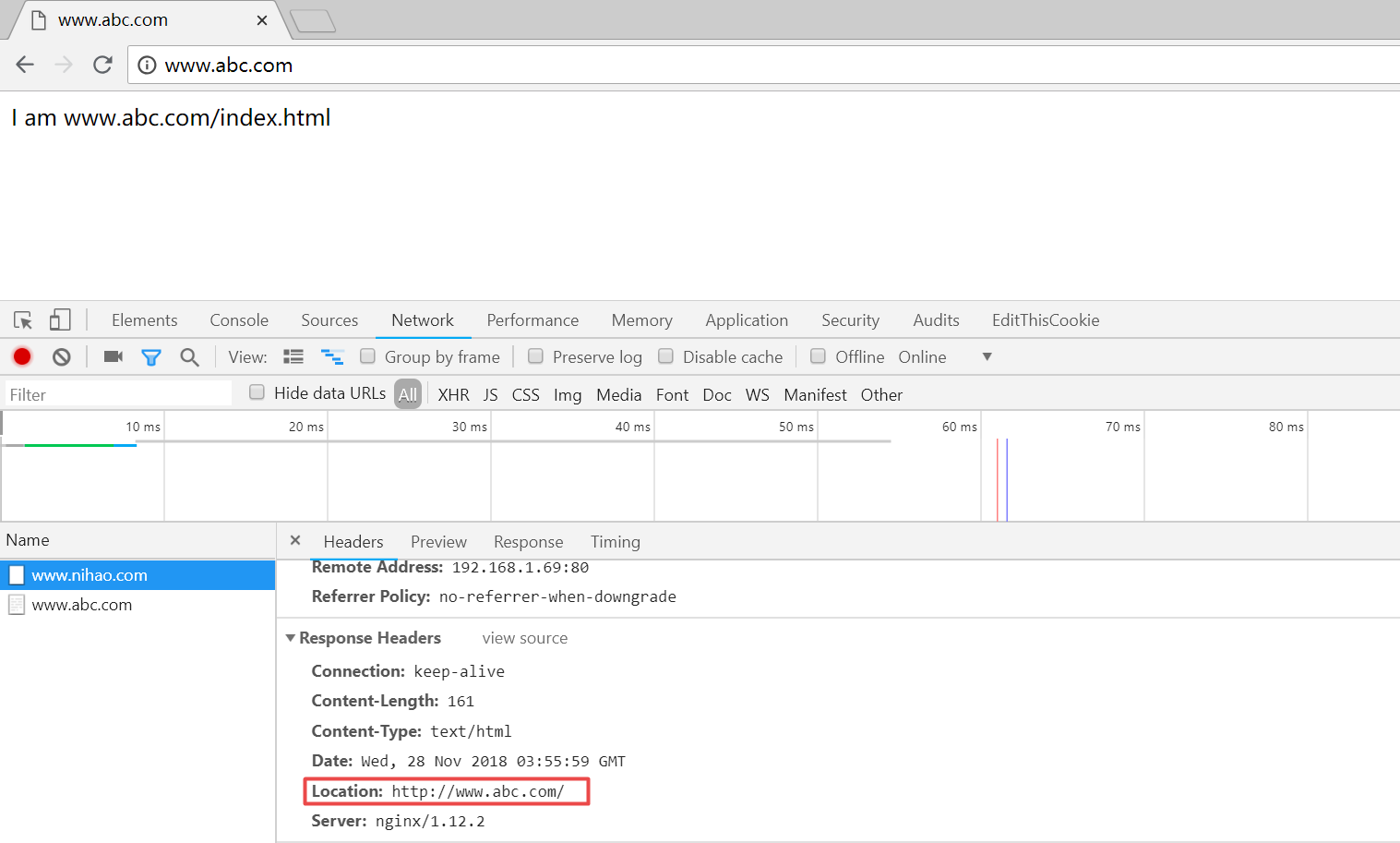


**302重定向；**

****

302重定向只是暂时的重定向，搜索引擎会抓取新的内容而保留旧的地址，**因为服务器返回302，所以，搜索搜索引擎认为新的网址是暂时的。**

**而301重定向是永久的重定向，搜索引擎在抓取新的内容的同时也将旧的网址替换为了重定向之后的网址。**



# 三、gzip压缩提升网站速度；

**gzip on;**

**#开启gzip压缩功能**

**gzip\_min\_length 1k;#(开始压缩的最小长度，再小不要压缩)**

**#设置允许压缩的页面最小字节数，页面字节数从header头的content-length中获取。默认值是0,不管页面多大都进行压缩。建议设置成大于1k。如果小于1k可能会越压越大。**

**gzip\_buffers 4 16k;#(压缩在内存中缓冲几块，每块多大？)**

**#压缩缓冲区大小。表示申请4个单位为16k的内容作为压缩结果流缓存，默认值是申请与原始数据大小相同的内存空间来存储gzip压缩结果。**

**gzip\_disable;#正则匹配，什么样的uri不进行gzip压缩。**

**gzip\_http\_version 1.0;**

**#压缩版本（默认1.1，前端为squid2.5时使用1.0）用于设置识别http协议版本，默认是1.1,目前大部分浏览器已经支持gzip解压，使用默认即可。**

**gzip\_comp\_level 6;**

**#压缩比率。用来指定gzip压缩比，1压缩比量小，处理速度快；9压缩比量大，传输速度快，但处理最慢，也必将消耗cpu资源。**

**gzip\_types text/plain application/x-javascript text/css application/xml;**

**#用来指定压缩的类型，“text/html”类型总是会被压缩。**

**gzip\_vary on;#是否传输gzip压缩标志。**

**#vary header支持。该选项可以让前端的缓存服务器缓存经过gzip压缩的页面，例如用squid缓存经过nginx压缩的数据。**

**要注意：需要和不需要压缩的对象**

**（1）大于1k的纯文本文件html,js,css,xml,html.**

**（2）图片，视频等不要压缩，因为不但不会减小，在压缩时消耗cpu和内存资源。**

**gzip on;**

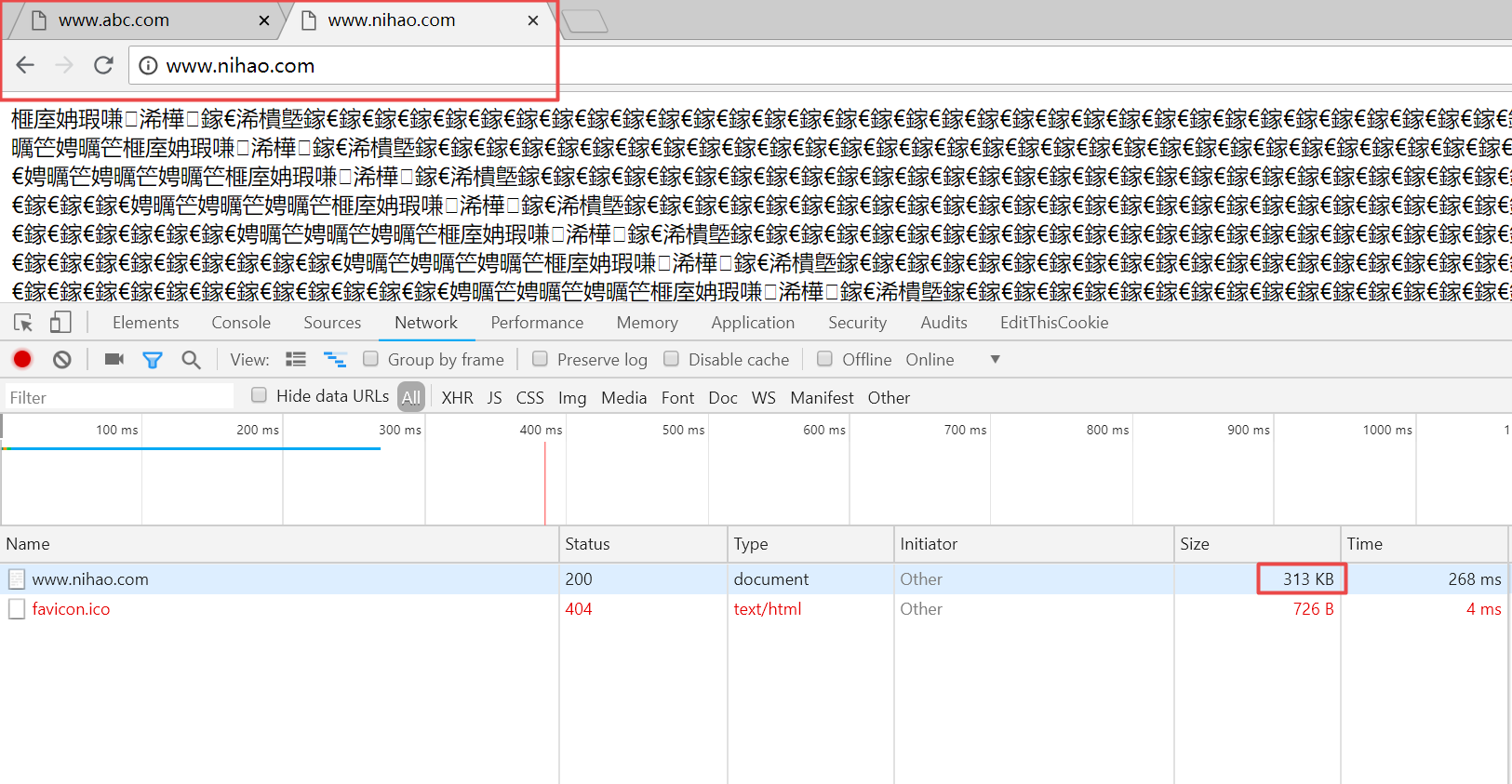
**gzip\_min\_length 1k;**

**gzip\_buffers 4 16k;**

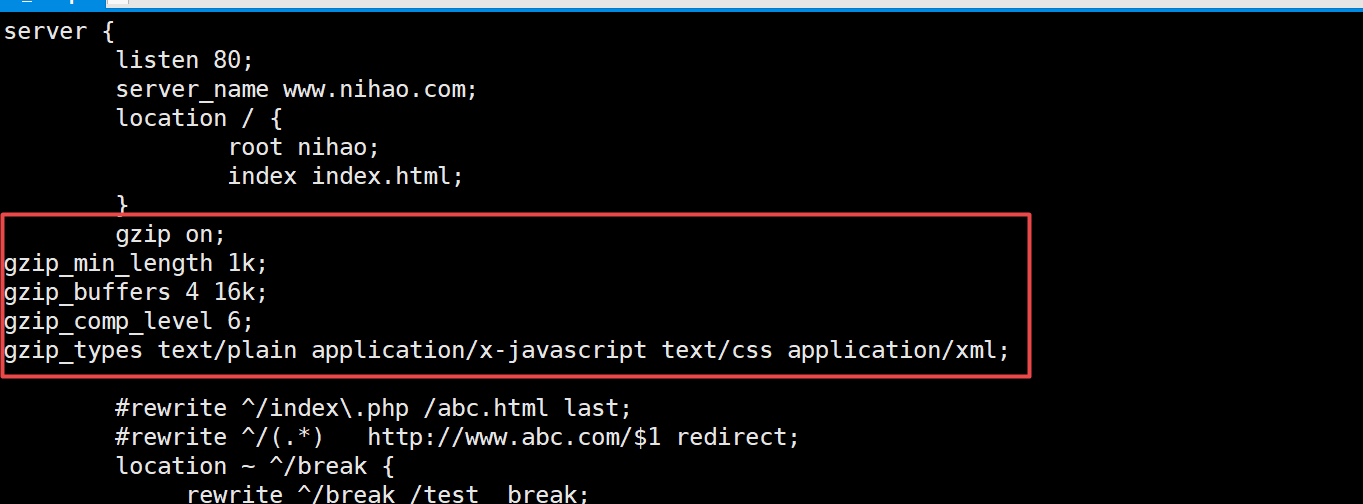
**gzip\_comp\_level 6;**

**gzip\_types text/plain application/x-javascript text/css application/xml;**

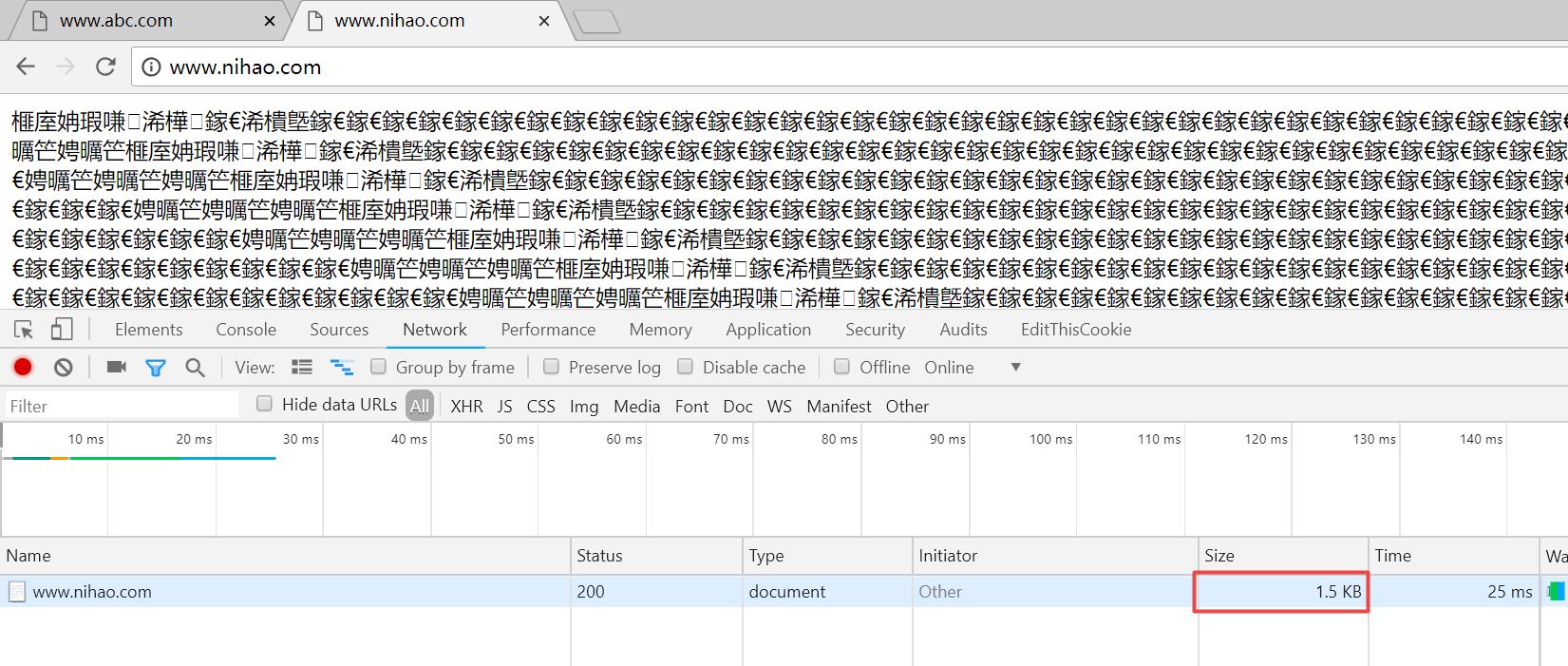
没有压缩时，文件大小还是原来的大小



进行压缩配置如下；



压缩效果如下；



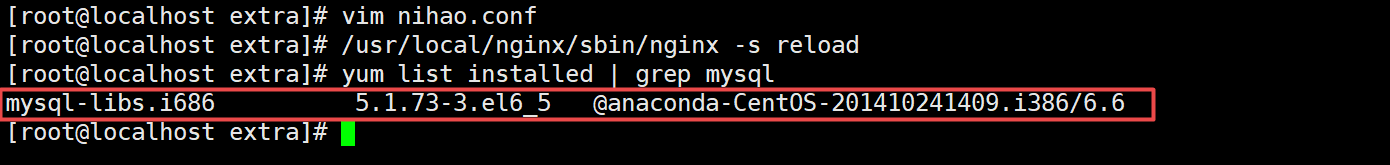
# 四、LNMP环境安装MySql

## 1、 查看旧的mysql情况

[root@localhost ~] yum list installed | grep mysql

删除旧的mysql残余

[root@localhost ~] yum remove -y mysql-libs



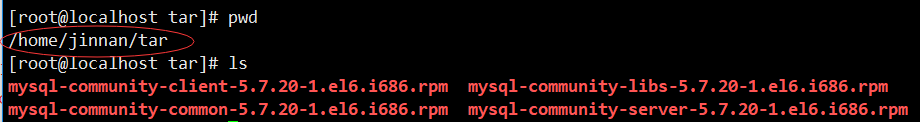
## 2、下载mysql组件

官网：<https://www.mysql.com>

下载地址：<https://dev.mysql.com/downloads/mysql/>

## 3、上传mysql组件

上传mysql的4个组件到/home/jinnan/tar目录下



## 4、安装mysql的4个组件

[root@localhost ~]

rpm -ivh mysql-community-common-5.7.20-1.el6.i686.rpm

[root@localhost ~]

rpm -ivh mysql-community-libs-5.7.20-1.el6.i686.rpm

[root@localhost ~]

rpm -ivh mysql-community-client-5.7.20-1.el6.i686.rpm

[root@localhost ~]

rpm -ivh mysql-community-server-5.7.20-1.el6.i686.rpm

或

rpm -ivh mysql-community-common-5.7.20-1.el6.i686.rpm

rpm -ivh mysql-community-libs-5.7.20-1.el6.i686.rpm

rpm -ivh mysql-community-client-5.7.20-1.el6.i686.rpm

rpm -ivh mysql-community-server-5.7.20-1.el6.i686.rpm

## 5、启动mysql

第一次启动MySQL的同时也会自动初始化mysql数据库，

初始化：给MySQL数据库创建初始化数据文件，文件生成在/var/lib/mysql目录（现在该目录时空的）

[root@localhost ~] service mysqld start

第一次失败效果：



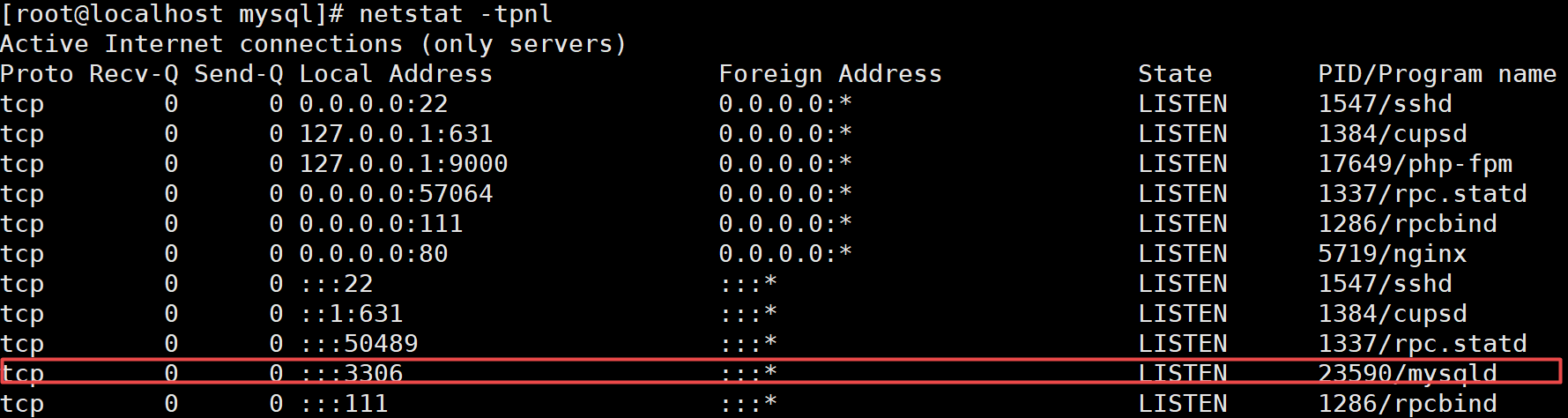
再次执行start，发现成功



查看/var/lib/mysql目录有文件生成。

查看mysql启动进程

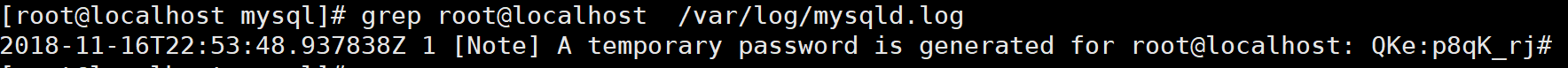
[root@localhost ~] netstat -lntp



## 6、查看mysql登录密码

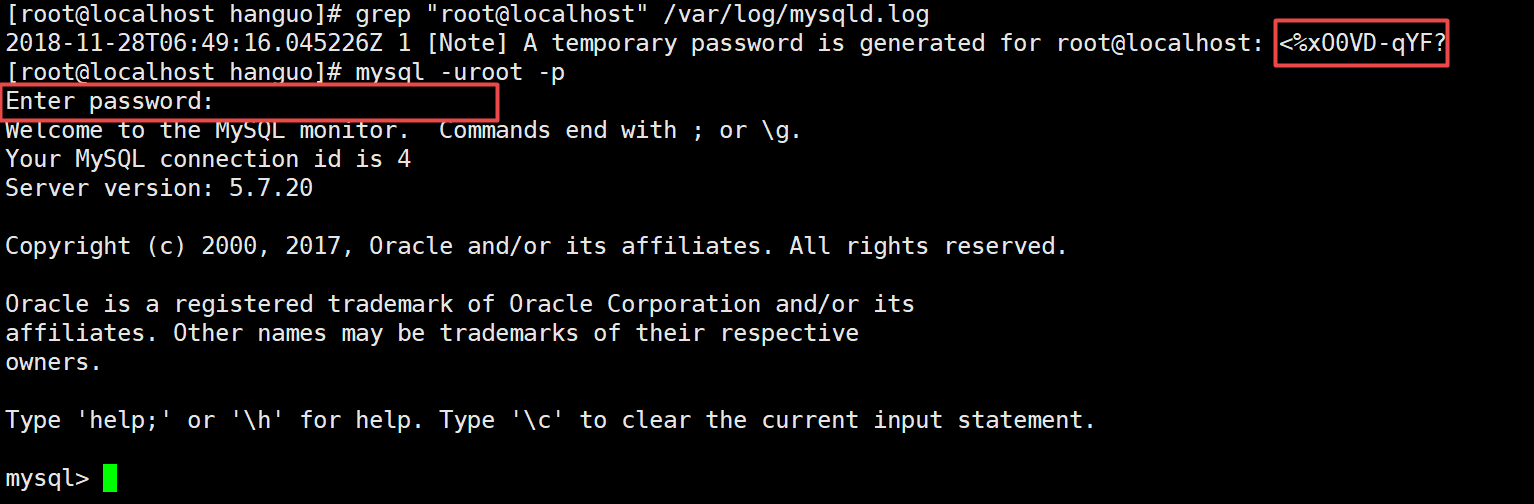
[root@localhost ~] grep "root@localhost" /var/log/mysqld.log

上述指令意思是，在/var/log/mysqld.log安装日志文件中查看root账号的初始密码。



## 7、 登录mysql操作

[root@localhost ~] mysql -uroot -p密码

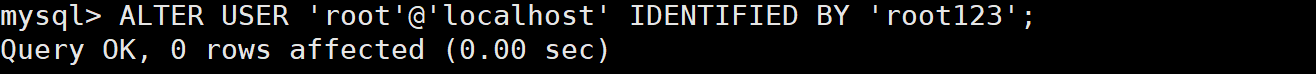


## 8、 修改登录密码

mysql> ALTER USER 'root'@'localhost' IDENTIFIED BY 'abc.123456.ABC';

alter user ‘root’@’localhost’ IDENTIFIED by ‘root123’

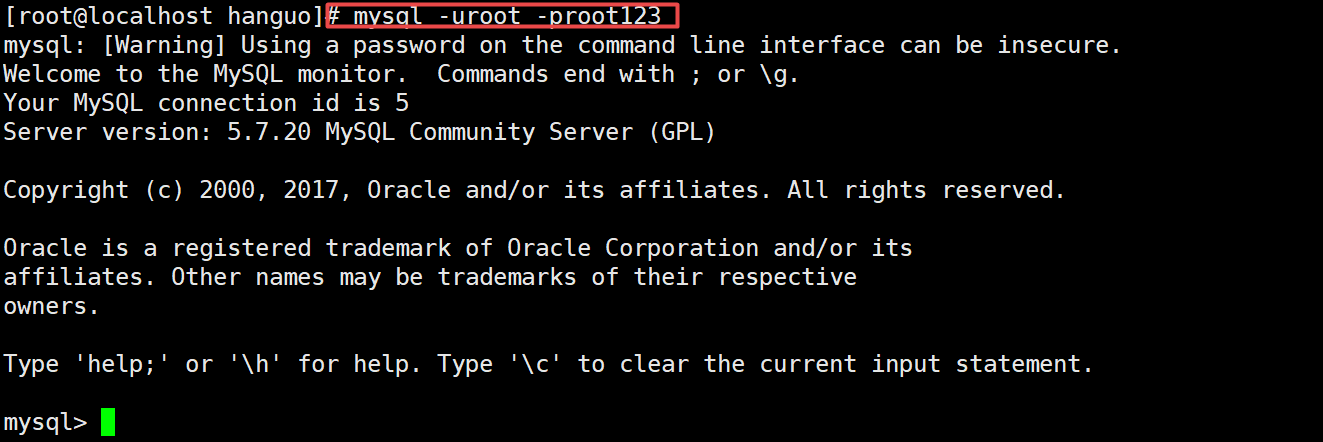
执行完上述指令，密码即设置成功，无需flush刷新



注意，密码要求有一定复杂度，否则会失败，如下图(密码为123456，太过简单):



使用修改的新密码登录即可；



## 9、 控制mysql服务

[root@localhost ~] service mysqld start/stop/restart

注意：如果启动mysql失败，则执行如下命令，删除干净。

rm -fr /var/lib/mysql/\*

rm /var/lock/subsys/mysqld

killall mysqld

## 10、数据库存储的目录

datadir=/var/lib/mysql

## 11、数据库配置文件

/etc/my.cnf

## 12、给mysql设置字符集

[root@localhost ~] vi /etc/my.cnf

在配置文件最后添加以下内容：

init-connect='SET NAMES utf8mb4'

default-storage-engine=INNODB

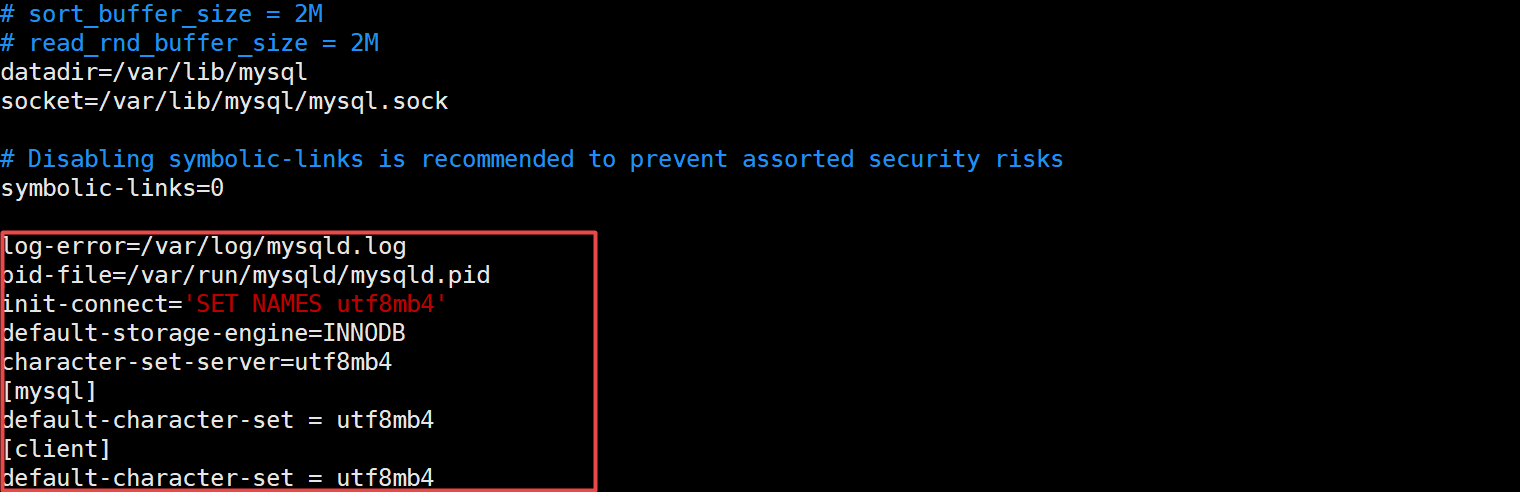
character-set-server=utf8mb4

[mysql]

default-character-set = utf8mb4

[client]

default-character-set = utf8mb4



之后重启mysql

[root@localhost ~] service mysqld restart

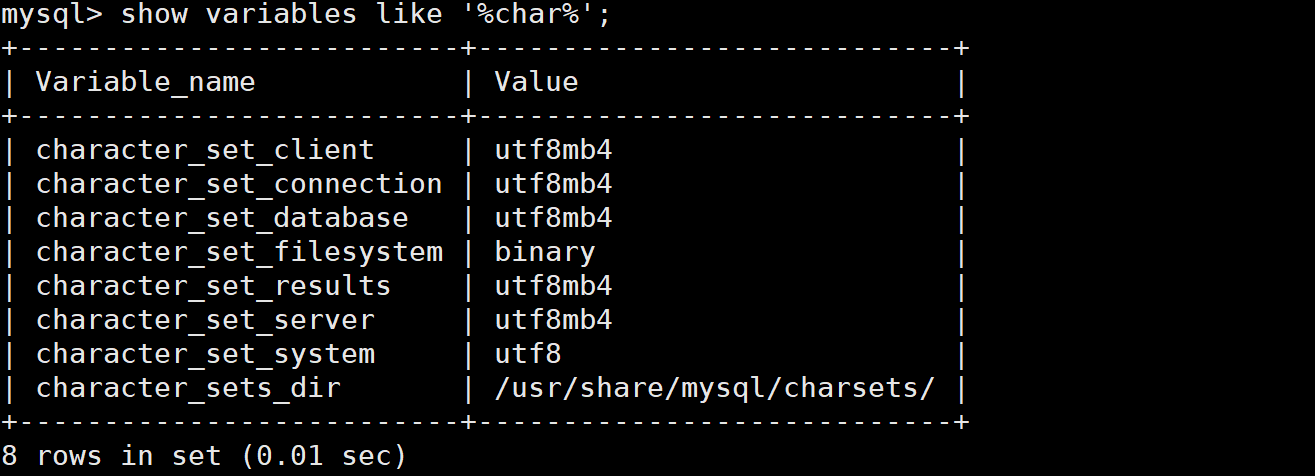
之后登录mysql操作终端

[root@localhost ~] mysql -uroot –proot123

查看mysql字符集

mysql> show variables like '%char%';

出现下图效果即说明mysql字符串设置ok



# 五、LNMP环境安装PHP

apache+PHP：php是apache内部的功能模块

apache服务

php模块

nginx+PHP：php作为独立服务运行，与nginx地位平等

Nginx服务

端口：80

php服务

端口默认：9000

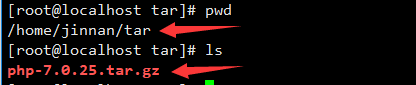
找到php服务

解释运行php代码

监听 127.0.0.1的9000端口，当ngxin遇到php文件时，会把这个php的请求转给9000端口的php来处理，nginx本身不能处理php.

## 1、安装

[root@localhost ~] rz //上传php-7.0.25.tar.gz到/home/jinnan/tar



[root@localhost ~] cd /home/jinnan/tar

[root@localhost ~] tar zxf php-7.0.25.tar.gz

[root@localhost ~] cd php-7.0.25

现在我们给php做configure配置，就是去除apache参数，增加enable-fpm和--with-config-file-path参数，就两个变化

[root@localhost ~] ./configure \

--prefix=/usr/local/php7.0 \

--with-config-file-path=/usr/local/php7.0/etc \

--with-curl --with-freetype-dir --with-gd \

--with-gettext --with-iconv-dir --with-kerberos \

--with-libxml-dir \

--with-mysqli --with-openssl --with-pcre-regex \

--with-pdo-mysql --with-pear --with-png-dir \

--with-xmlrpc --with-xsl --with-zlib \

--enable-bcmath --enable-libxml \

--enable-inline-optimization \

--enable-gd-native-ttf \

--enable-mbregex --enable-mbstring --enable-opcache \

--enable-pcntl --enable-shmop --enable-soap \

--enable-sockets --enable-sysvsem \

--enable-xml --enable-zip \

--enable-fpm

参数说明：

--with-config-file-path 设定php.ini的存储目录

--with-curl 打开curl的支持

--with-freetype-dir 字体库支持

--with-gd 画图技术支持

--with-gettext 支持[开发多语言系统](http://www.cnblogs.com/ccdc/archive/2012/05/03/2480074.html)

--with-iconv-dir iconv函数库能够完成各种字符集间的转换

--with-kerberos kerberos支持

--with-libxml-dir libxml2库的支持

--with-mysqli Mysqli数据库的支持

--with-openssl openssl的支持，加密传输时用到的

--with-pcre-regex 正则表达式支持

--with-pdo-mysql pdo-mysql支持

--with-pear pear是PHP的扩展和应用程序库，包含了很多有用的类

--with-png-dir png图片支持

--with-xmlrpc xml相关的扩展库支持

--with-xsl 打开XSLT 文件支持，扩展了libXML2库 ，需要libxslt软件

--with-zlib zlib压缩库支持

--enable-bcmath 图片大小调整技术支持

--enable-libxml xml支持

--enable-inline-optimization 优化线程，给php整体做性能优化处理

--enable-gd-native-ttf 画图字体库支持，支持TrueType字符串函数库

--enable-mbregex 正则表达式支持

--enable-mbstring mb宽字节函数库支持

--enable-opcache 缓存支持

--enable-pcntl pcntl扩展可以支持php的多线程操作

--enable-shmop shmop是一个易于使用的功能集，允许PHP读，写，创建和删除UNIX共享内存段

--enable-sysvsem 作用同上

--enable-soap SOAP 的全称为简单对象访问协议 (Simple Object Access Protocol)。它是一种基于 XML 的，可扩展的通信协议。SOAP 提供了一种标准，使得运行在不同平台上并使用不同的编程语言编写的应用程序可以互相进行通信

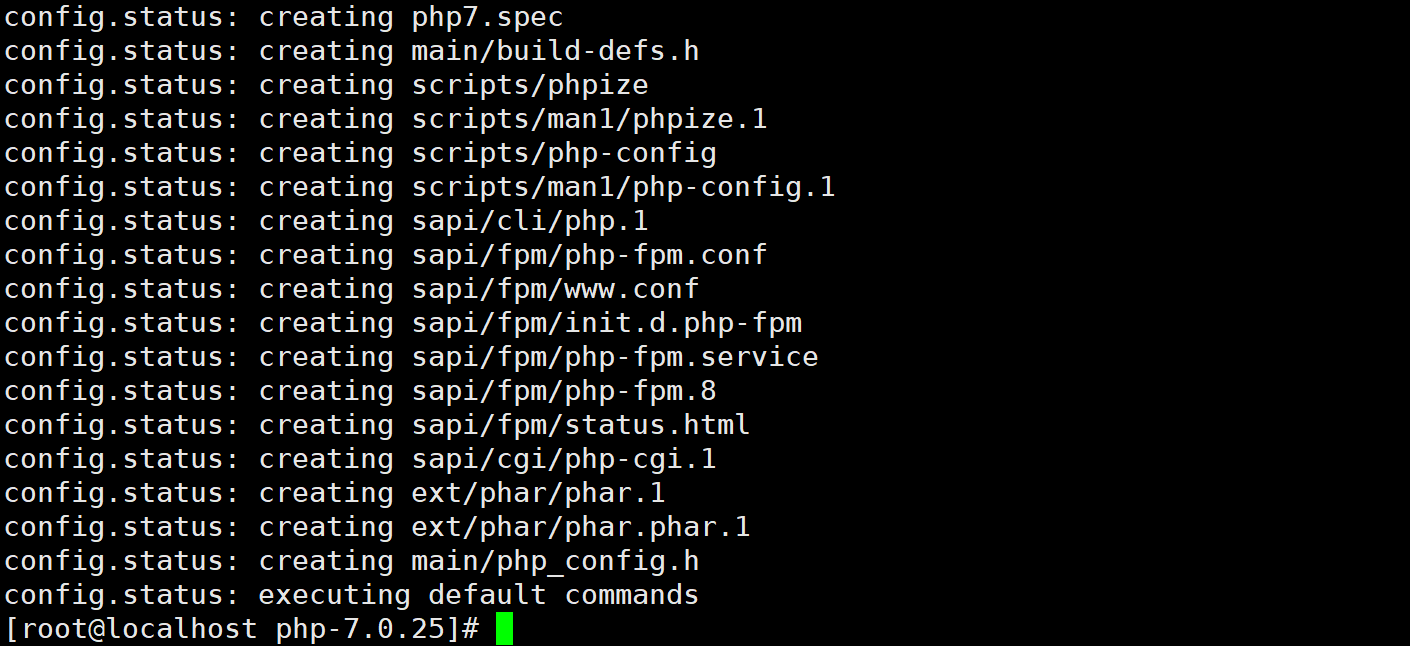
--enable-sockets sockets 支持

--enable-xml xml支持

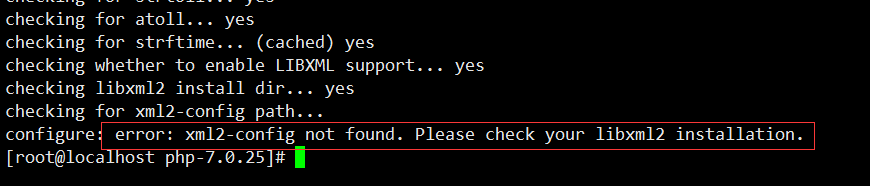
--enable-zip php支持对zip压缩包处理

--enable-fpm 将php作为独立服务运行

configure执行完毕效果：



我们第一次给php做configure配置的时候报错了：



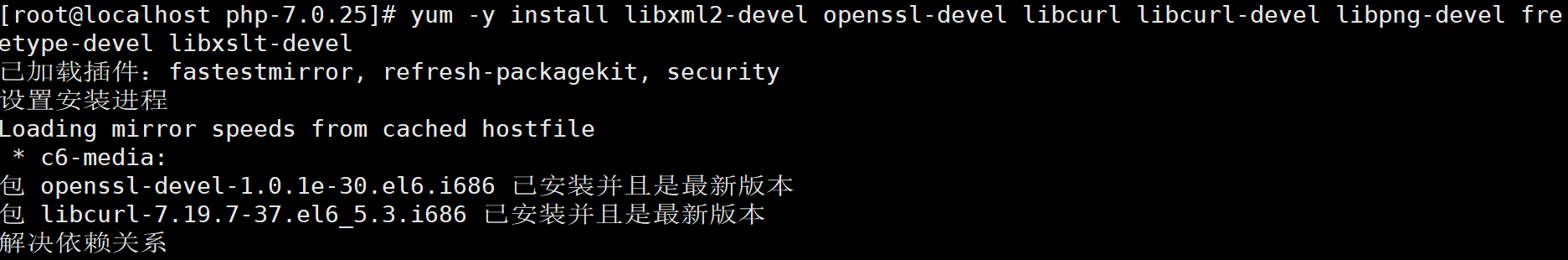
原因：根据错误提示，告知libxml2依赖包没有安装

这一步会遇到一些报错，提示各种依赖缺失，按照提示一个一个用yum安装就行，完毕再重复执行configure指令

通过一条指令统一处理下述依赖(yum可以一次性安装多个软件)：

[root@localhost ~] yum -y install libxml2-devel openssl-devel libcurl libcurl-devel libpng-devel freetype-devel libxslt-devel

值重复执行php的configure指令



/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*依赖区\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

error: xml2-config not found. Please check your libxml2 installation

[root@localhost ~] yum -y install libxml2-devel(xml库依赖支持)

error: Cannot find OpenSSL's <evp.h>

[root@localhost ~] yum -y install openssl-devel  
(openssl： 支持安全传输协议https)

checking for cURL in default path... not found  
[root@localhost ~] yum -y install libcurl libcurl-devel

(curl是服务器彼此间调用的接口，常用于爬虫技术)

configure: error: png.h not found.

[root@localhost ~] yum -y install libpng-devel

(png图片格式支持)

configure: error: freetype-config not found.

[root@localhost ~] yum -y install freetype-devel

(画图字体库支持)

configure: error: xslt-config not found. Please reinstall the libxslt >= 1.1.0 distribution

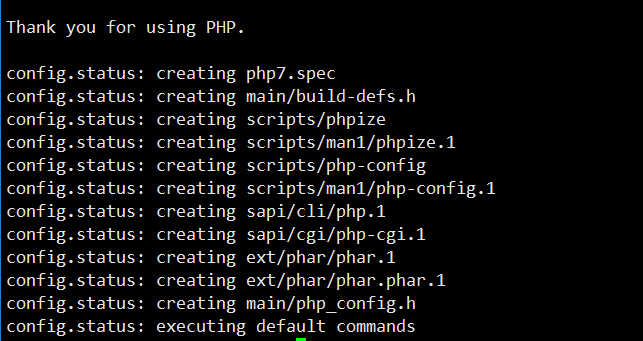
[root@localhost ~] yum -y install libxslt-devel

(打开XSLT 文件支持，扩展了libXML2库)

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*依赖区\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

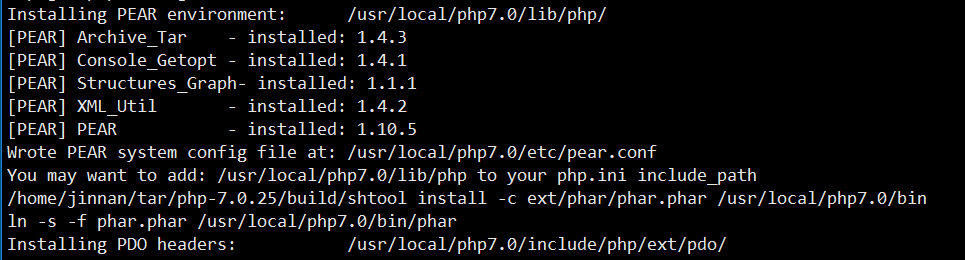
[root@localhost ~] make && make install

php执行完毕configure的效果：



[root@localhost ~] make && make install

执行完毕效果:

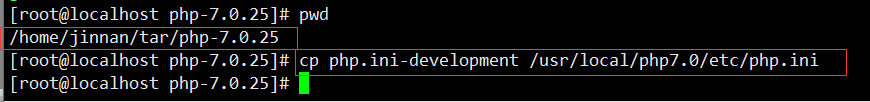


## 2、配置

（1）复制php.ini-development配置文件

安装完成后将php.ini-development 复制到/usr/local/php7.0/etc/php.ini

[root@localhost ~] cp php.ini-development /usr/local/php7.0/etc/php.ini

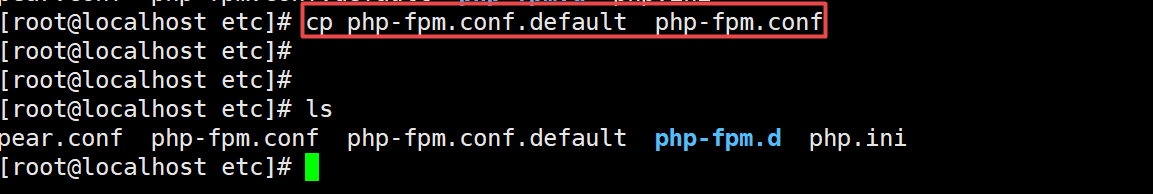


（2）给php服务，服务名称是fpm创建服务配置文件

[root@localhost ~] cd /usr/local/php7.0/etc

//给php服务，服务名称是fpm创建服务配置文件

[root@localhost ~] cp php-fpm.conf.default php-fpm.conf



[root@localhost ~] vim /usr/local/php7.0/etc/php-fpm.conf

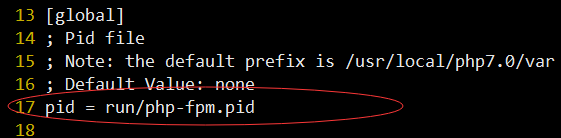
找到如下内容：

;pid = run/php-fpm.pid

去除;分号修改为：

pid = run/php-fpm.pid

上述文件是用于保存php服务进程号码的

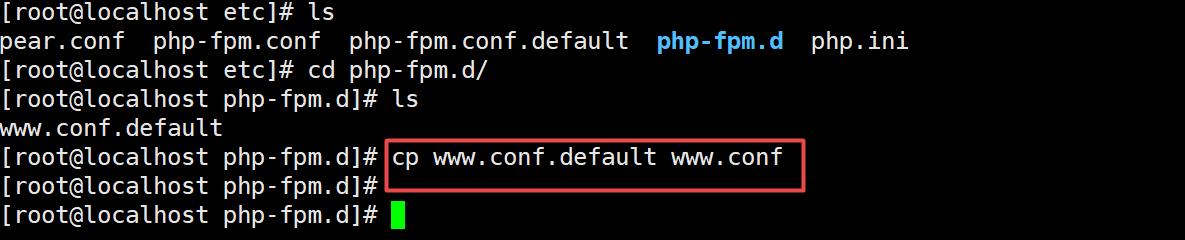


:wq保存退出

（3）配置辅助配置文件

[root@localhost ~] cd /usr/local/php7.0/etc/php-fpm.d

[root@localhost ~] cp www.conf.default [www.conf](http://www.conf)



/usr/local/php7.0/etc/php-fpm.conf: 是php服务的主配置文件

/usr/local/php7.0/etc/php-fpm.d/www.conf: 是php服务辅助配置文件，需要被上述的php-fpm.conf引入

## 3、控制服务

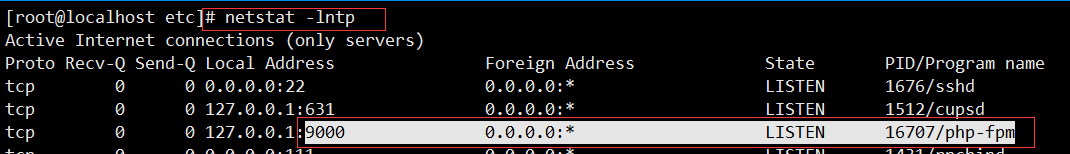
启动php服务

[root@localhost ~] /usr/local/php7.0/sbin/php-fpm



查看php服务进程

[root@localhost ~] netstat -lntp

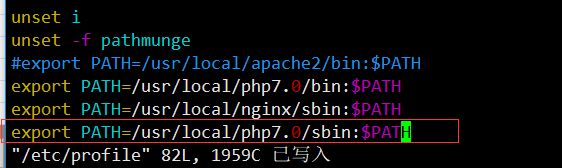


设置环境变量，使得在任何地方都可以直接访问php-fpm

[root@localhost ~] vim /etc/profile

再文档最后设置如下内容：

export PATH=/usr/local/php7.0/sbin:$PATH



:wq保存退出

使得环境变量立即生效

[root@localhost ~] source /etc/profile

也可以配置nginx的环境变量

export PATH=/usr/local/nginx/sbin:$PATH

关闭php服务：

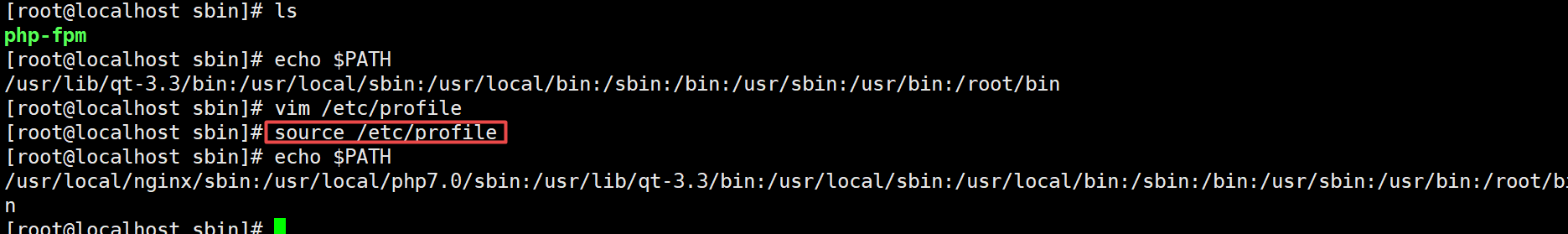
> killall php-fpm

杀掉php服务进程：



启动php服务进程：

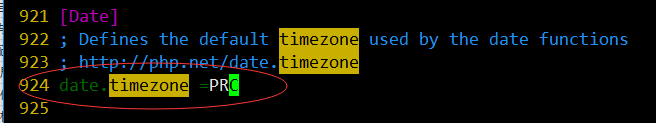
> php-fpm



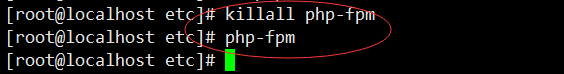
## 4、修改配置

修改php时区：

> vi /usr/local/php7.0/etc/php.ini



重启php服务：

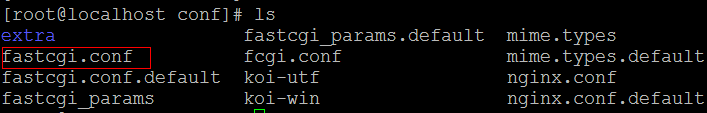


## 5、编译 php与nginx整合

### （1）编辑文件：

*shell>#* vim /usr/local/nginx/conf/**fastcgi.conf**

**直接使用conf配置目录下面的fastcgi.conf文件也可以，不需要自己在建立fcgi.conf 文件了。**

****

并写入如下内容

fastcgi\_param GATEWAY\_INTERFACE CGI/1.1;

fastcgi\_param SERVER\_SOFTWARE nginx;

fastcgi\_param QUERY\_STRING $query\_string;

fastcgi\_param REQUEST\_METHOD $request\_method;

fastcgi\_param CONTENT\_TYPE $content\_type;

fastcgi\_param CONTENT\_LENGTH $content\_length;

fastcgi\_param SCRIPT\_FILENAME $document\_root$fastcgi\_script\_name;

fastcgi\_param SCRIPT\_NAME $fastcgi\_script\_name;

fastcgi\_param REQUEST\_URI $request\_uri;

fastcgi\_param DOCUMENT\_URI $document\_uri;

fastcgi\_param DOCUMENT\_ROOT $document\_root;

fastcgi\_param SERVER\_PROTOCOL $server\_protocol;

fastcgi\_param REMOTE\_ADDR $remote\_addr;

fastcgi\_param REMOTE\_PORT $remote\_port;

fastcgi\_param SERVER\_ADDR $server\_addr;

fastcgi\_param SERVER\_PORT $server\_port;

fastcgi\_param SERVER\_NAME $server\_name;

# PHP only, required if PHP was built with --enable-force-cgi-redirect

fastcgi\_param REDIRECT\_STATUS 200;

### （2）编辑nginx配置文件(找到server项目设置如下内容)：

*shell># vim /usr/local/nginx/conf/nginx.conf*

location ~ .\*\.(php|php5)?$

{

fastcgi\_pass 127.0.0.1:9000;

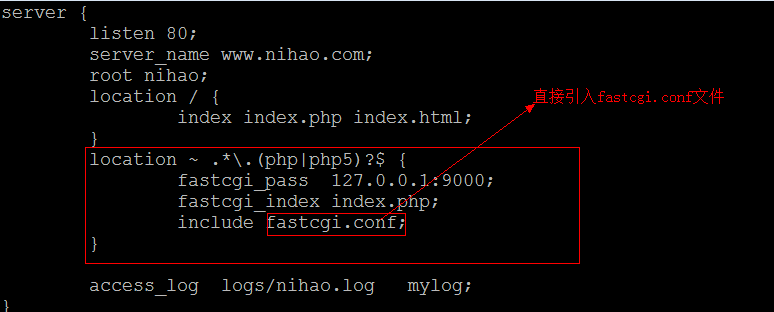
#fastcgi（管理PHP），

fastcgi\_index index.php;

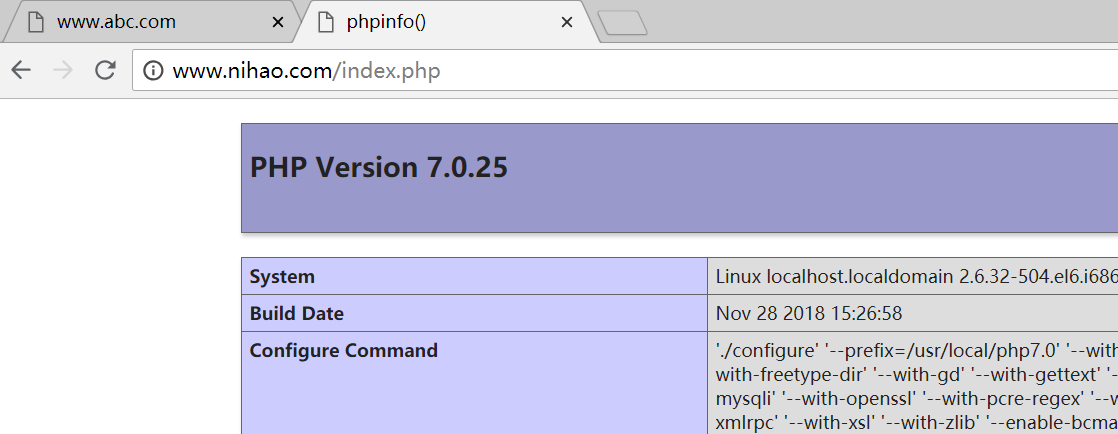
include fastcgi.conf;

}

我们就在 [www.nihao.com域名下面进行测试，因此打开extra/nihao.conf](http://www.nihao.com域名下面进行测试，因此打开extra/nihao.conf)文件，添加以上代码。



在nihao目录下面新建一个php文件，进行测试。



## 4、nginx上面部署TP框架

具体的配置如下，可以把如下内容，添加到虚拟主机里面，即可。

location ~ .+\.php($|/) {

set $script $uri;

set $path\_info "/";

if ($uri ~ "^(.+\.php)(/.+)") {

set $script $1;

set $path\_info $2;

}

fastcgi\_pass 127.0.0.1:9000;

fastcgi\_index index.php?IF\_REWRITE=1;

include fastcgi.conf;

fastcgi\_param PATH\_INFO $path\_info;

fastcgi\_param SCRIPT\_FILENAME $document\_root/$script;

fastcgi\_param SCRIPT\_NAME $script;

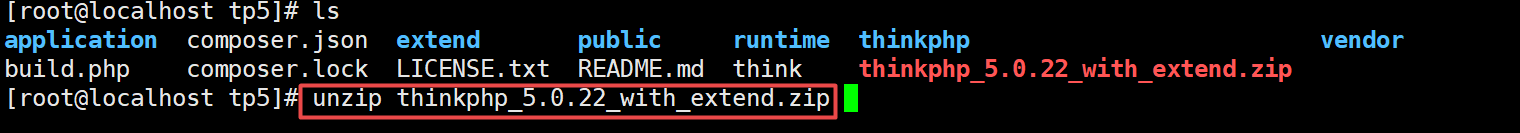
}

我们就在[www.tp5.com](http://www.tp5.com)域名下面测试。

第一步：

把tp的源码包拷贝到项目中来(/usr/local/nginx/tp5目录下面)，完成解压

**针对扩展名为zip的文件进行解压，使用unzip来完成，**



第二步：

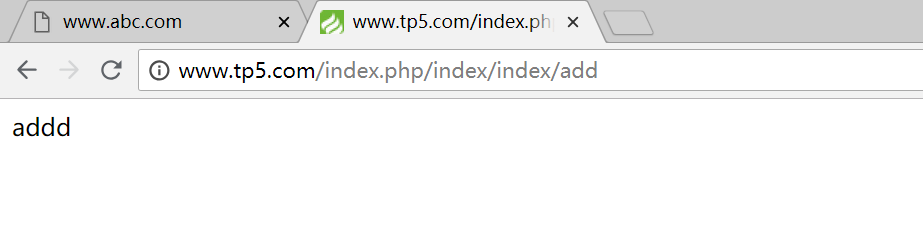
在访问时如果出现如下提示，则说明App目录没有写权限，因此要添加写权限。

访问效果如下：



第三步：在tp的默认控制器里面，新建一个方法，进行测试。

通过浏览器访问效果如下：



第四步，修改虚拟主机，添加如下配置，支持TP框架。

虚拟主机的配置如下：



## 5、nginx上面部署TP框架实现伪静态

只需要在虚拟主机的配置文件里面，添加如下内容即可：

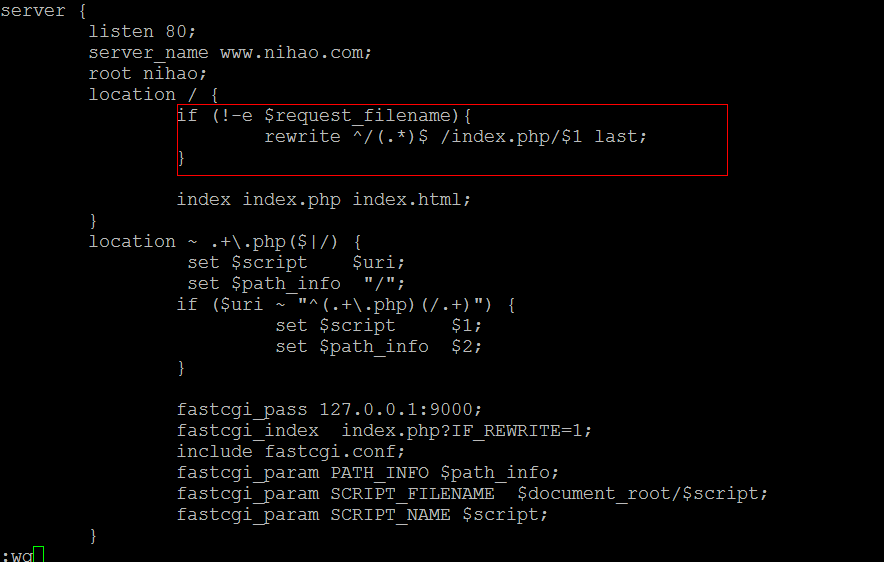
if (!-e $request\_filename){

rewrite ^/(.\*)$ /index.php/$1 last;

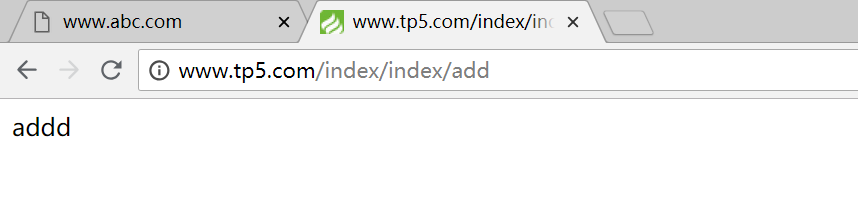
}

第一步：以[www.nihao.com域名来完成配置，打开extra/nihao.conf](http://www.nihao.com域名来完成配置，打开extra/nihao.conf)配置文件

添加如下内容：



访问效果如下：



## 6、安装dedecms软件

cp -Rp uploads/ ../dedecms/

### （1）新建一个虚拟主机；

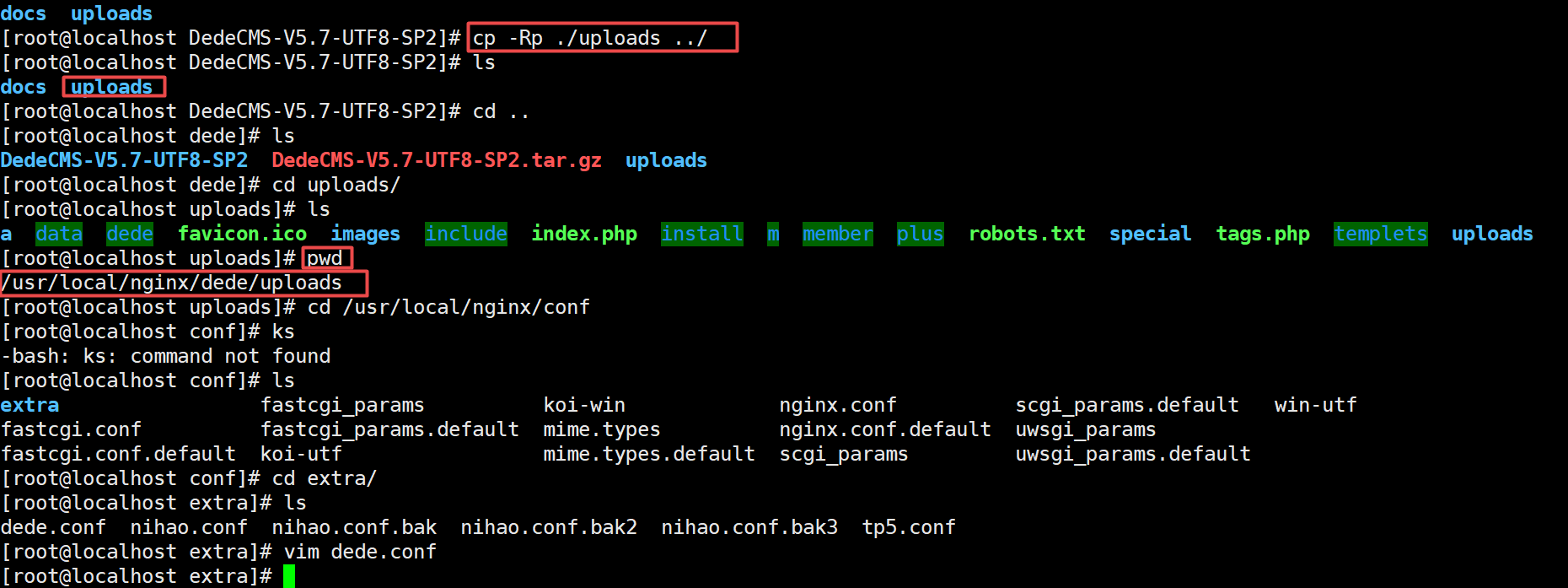
第一步：在extra目录下面新建一个dede.conf的文件

第二步：在nginx.conf配置文件中，引入dede.conf文件



### （2）上传dedecms的软件，进行解压

### （3）把解压后的软件拷贝到虚拟主机目录里面

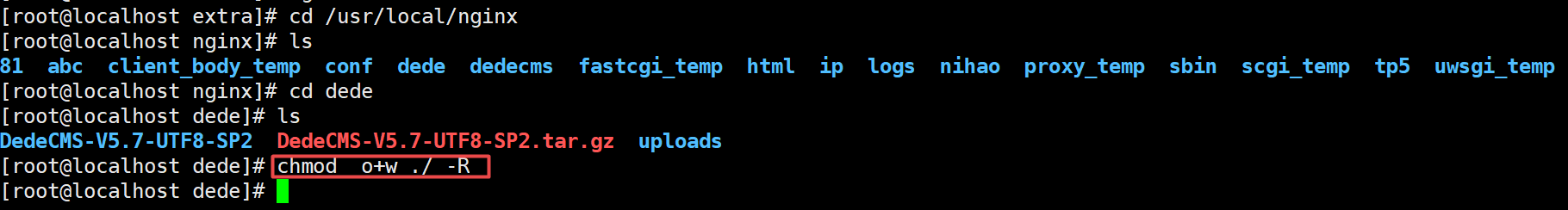


### （4）在浏览器输入域名开始安装测试。

出现了如下问题：某些目录不可写，授权该目录可写。



进行授权的代码如下：



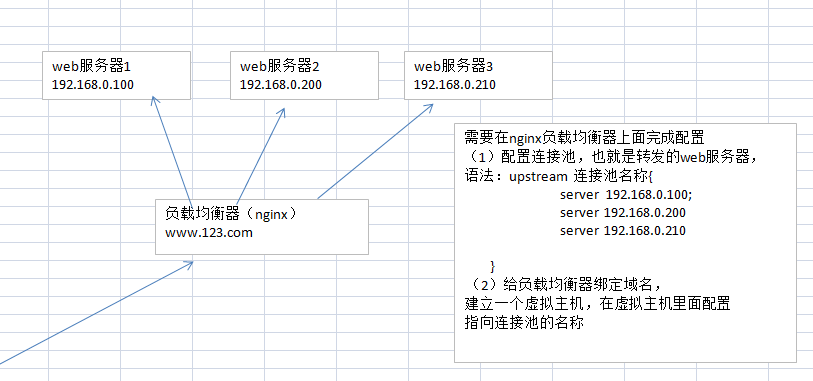
[root@localhost dedecms]# chmod o+w ./ -R

继续安装如下效果；



# 六、nginx的负载均衡的配置

## 1、配置的原理图

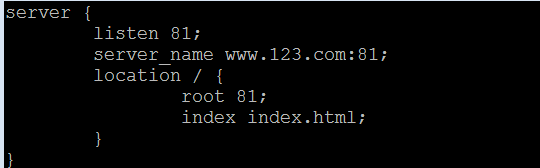


## 2、具体的配置

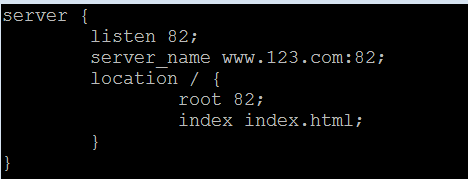
建立两个基于端口的虚拟主机来模拟两台web服务器。

（1）新建一个[www.123.com:81](http://www.123.com:81)和[www.123.com:82](http://www.123.com:82)的虚拟主机

在extra目录下面新建一个81.conf文件，内容如下：



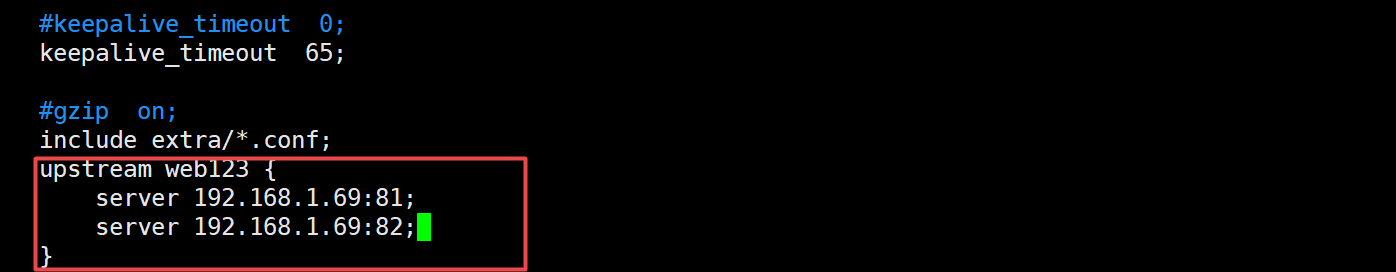
在extra目录下面新建一个82.conf文件，内容如下：



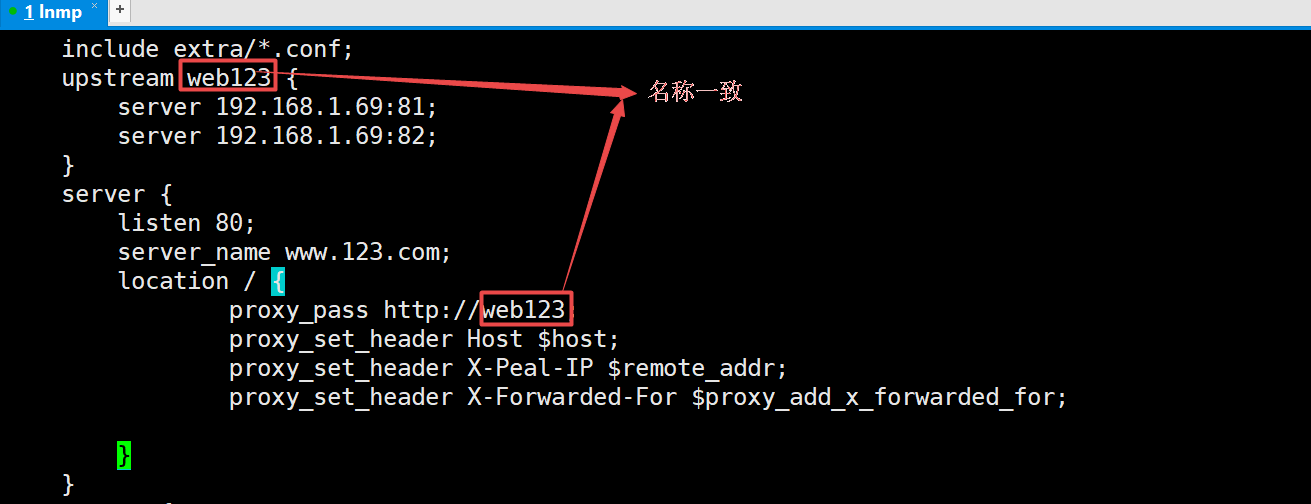
（2）在nginx目录下面分别建立81和82的目录，在该目录下面分别建立index.html文件，

为了查看到负载均衡的效果，可以让81里面的index.html和82里面的index.html文件的内容不同。

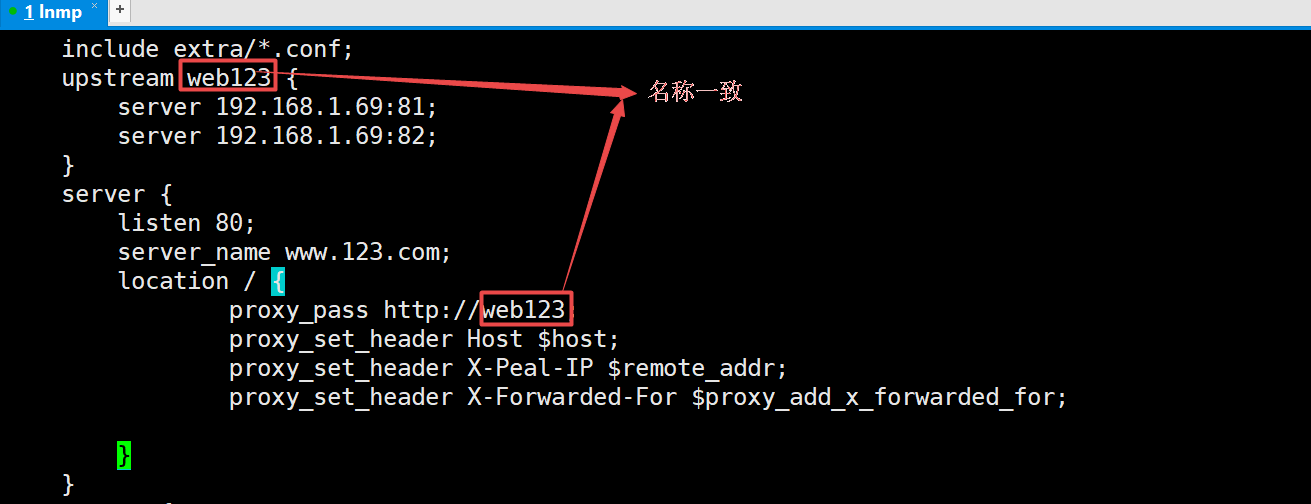
（3）在nginx.conf文件里面，建立一个连接池



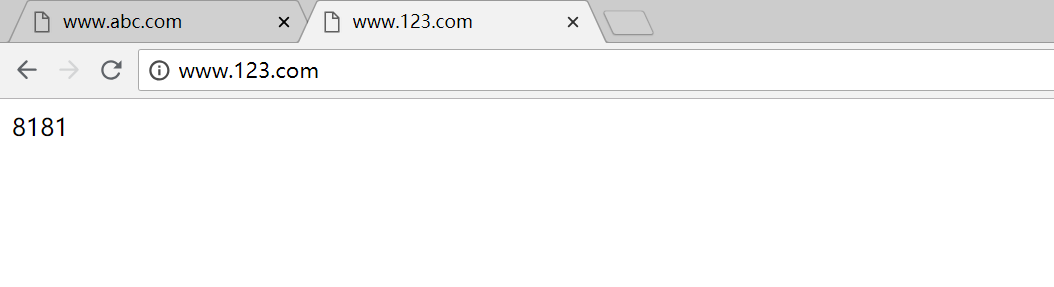
（4）在nginx.conf文件里面，建立一个[www.123.com](http://www.123.com)的虚拟主机



（5）把之前配置的81.conf和82.conf文件引入，



效果如下：



## 3、连接池里面的其他指令

server指令：

语法：server name [参数]

使用环境：upstream

该指令用于指定后端服务器的名称和参数。服务器的名称可以是一个域名，-个ip地址，端口号。

在后端服务器名称之后，可以跟以下参数：

**weight=number**  设置服务器的权重，权重数值越高，被分配到的客户端请求数越多。

如果没有设置权重，则为默认权重为1.

upstream 连接池名称{

server 192.168.0.100 weight=5;

server 192.168.0.200 weight=1;

server 192.168.0.210 weight=3;

}

max\_fails=number 在参数fail\_timeout指定的时间内对后端服务器请求失败的次数，如果检测到后端服务器无法连接及发生服务器错误（404错误除外），则标记为失败。如果没有设置，则为默认值1。设为数值0将关闭这项检查。

fail\_timeout=time(30s)在经历参数max\_fails设置的失败次数后，暂停的世界。

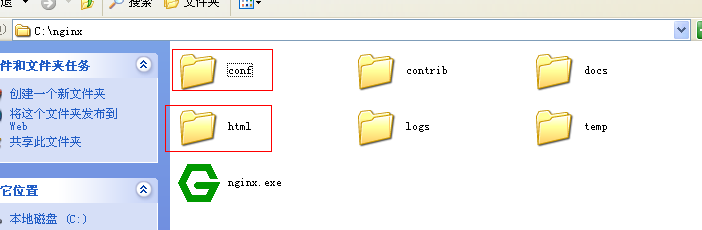
down 标记服务器为永久离线状态，用于ip\_hash指令。

backup 仅仅在非backup服务器全部宕机或繁忙的时候，才启用。

# 七、window下安装NGINX

## 1、NGINX安装

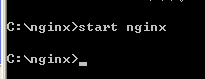
**（1）解压会，拷贝到指定的目录**

****

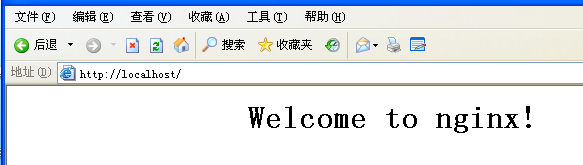
网站的根目录

配置文件所在的目录

**（2）以cmd的方式进入到nginx的目录，执行start nginx 直接启动即可。**

****

**（3）我们通过浏览器访问：**

****

## 2、nginx的虚拟主机的配置

**（1）打开nginx的配置文件，里面看到**

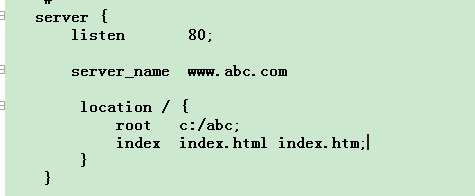
**server{**

**//具体的内容**

**}**

**每一个server段是一个虚拟主机**

监听的端口

****

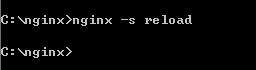
访问的默认文件。

虚拟主机的目录

虚拟主机的域名

**（2）配置完成后要重新加载nginx**

**执行命令：nginx –s reload**

****

**(3)修改hosts 文件，访问：**

****

**cd nginx**

**start nginx//启动nginx服务。**

**nginx -s stop // 停止nginx  
nginx -s reload // 重新加载配置文件  
nginx -s quit // 退出nginx**

## 3、nginx支持php

NGINX运行PHP需要运行NGINX和FASTCGI+php

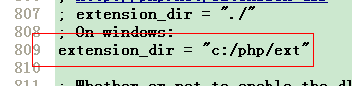
Nginx监听80：

Fastcgi+php : 9000

然后当NGINX遇到PHP文件时，会把这个PHP的请求转给9000端口 的PHP来处理，NGINX本身不能处理PHP。

**（1）把php程序拷贝和nginx同一个目录下面，**

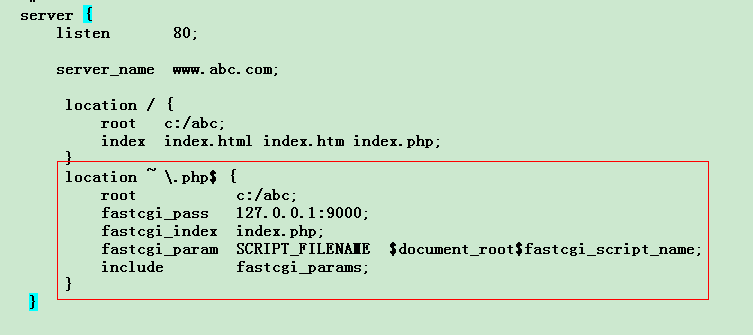
**（2）配置加载扩展的目录**

****

**（3）打开一个cgi的一个接口，nginx遇到后缀是php文件的文件，交给9000端口来处理。**

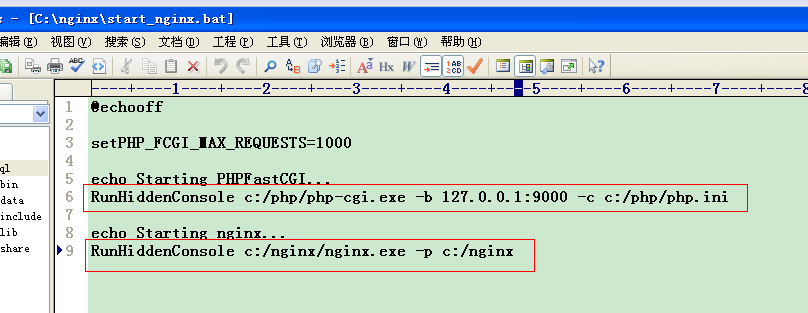
****

**（4）打开nginx的配置文件，进行配置**

****

该段内容是支持php程序的代码

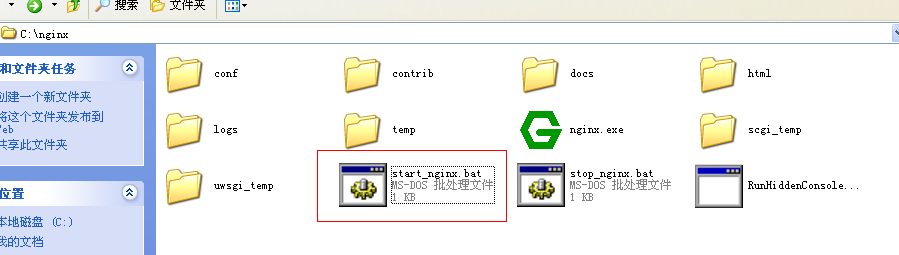
**（5）写一个脚本便于管理：**

****

一定要按照当前安装的目录进行设置，千万不还要照抄

**RunHiddenConsole一个让你的程序隐藏运行的小工具**

**（6）**

****

双击即可启动

**最终效果：**

****