第一周 web环境搭建和安全加固

CentOS版本：

[centos@bogon ~]$ cat /etc/redhat-release

CentOS Linux release 7.6.1810 (Core)

修改CentOS默认yum源为国内yum镜像源：

备份下原来的yum源

cd /etc/yum.repos.d/

mv CentOS-Base.repo CentOS-Base.repo_bak

网易yum源：

wget -O /etc/yum.repos.d/CentOS-Base.repo http://mirrors.163.com/.help/CentOS7-Base-163.repo

yum clean all

yum makecache

阿里云yum源：

wget -O /etc/yum.repos.d/CentOS-Base.repo http://mirrors.aliyun.com/repo/Centos-7.repo

yum clean all

yum makecache

一、LEMP环境搭建

1. 安装 nginx

1.1 安装稳定版本：

#wget https://nginx.org/download/nginx-1.16.0.tar.gz

1.2 直接./configure安装报错：

./configure: error: the HTTP rewrite module requires the PCRE library.

安装pcre-devel与openssl-devel解决问题：

#yum -y install pcre-devel openssl openssl-devel

编译安装：

#make

#make install

1.3 启动重启nginx

#/usr/local/nginx/sbin/nginx 启动

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

1.4 开机自启动

即在rc.local增加启动代码就可以了。

#vim /etc/rc.local

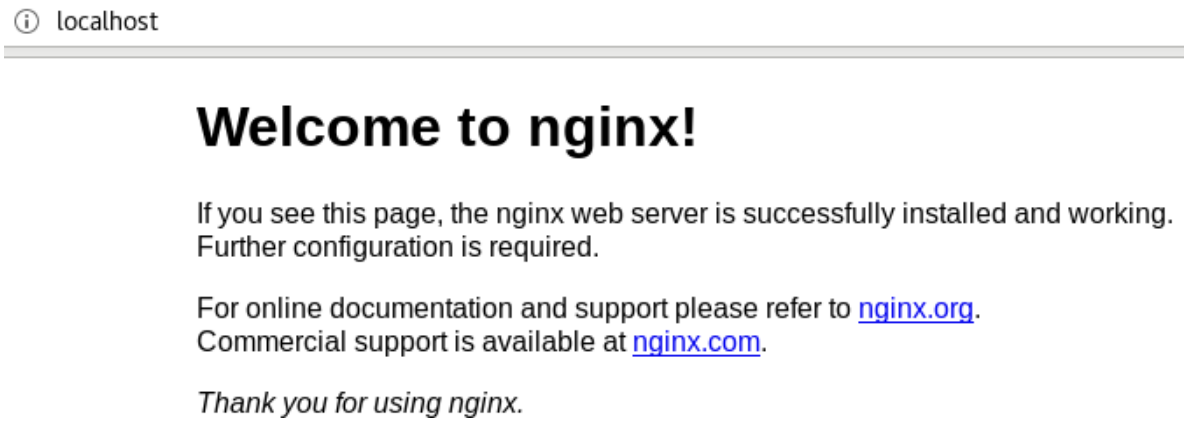
增加一行 /usr/local/nginx/sbin/nginx

设置执行权限：

#chmod 755 rc.local

1.5 测试nginx是否安装成功

浏览器访问：http://localhost，出现下面的信息代表安装成功：



屏幕剪辑的捕获时间: 2019/7/21 16:07

2. 安装MySQL

2.1 添加MySQL的yum源

wget https://dev.mysql.com/get/mysql80-community-release-el7-3.noarch.rpm

yum localinstall mysql80-community-release-el7-3.noarch.rpm

查看可供安装的版本，默认安装8.0版本

[root@bogon src]# yum repolist all | grep mysql

mysql-cluster-7.5-community/x86_64 MySQL Cluster 7.5 Community disabled

mysql-cluster-7.5-community-source MySQL Cluster 7.5 Community - disabled

mysql-cluster-7.6-community/x86_64 MySQL Cluster 7.6 Community disabled

mysql-cluster-7.6-community-source MySQL Cluster 7.6 Community - disabled

mysql-cluster-8.0-community/x86_64 MySQL Cluster 8.0 Community disabled

mysql-cluster-8.0-community-source MySQL Cluster 8.0 Community - disabled

mysql-connectors-community/x86_64 MySQL Connectors Community enabled: 108

mysql-connectors-community-source MySQL Connectors Community - disabled

mysql-tools-community/x86_64 MySQL Tools Community enabled: 90

mysql-tools-community-source MySQL Tools Community - Sourc disabled

mysql-tools-preview/x86_64 MySQL Tools Preview disabled

mysql-tools-preview-source MySQL Tools Preview - Source disabled

mysql55-community/x86_64 MySQL 5.5 Community Server disabled

mysql55-community-source MySQL 5.5 Community Server - disabled

mysql56-community/x86_64 MySQL 5.6 Community Server disabled

mysql56-community-source MySQL 5.6 Community Server - disabled

mysql57-community/x86_64 MySQL 5.7 Community Server disabled

mysql57-community-source MySQL 5.7 Community Server - disabled

mysql80-community/x86_64 MySQL 8.0 Community Server enabled: 113

mysql80-community-source MySQL 8.0 Community Server - disable

安装

yum install mysql-community-server

Total download size: 441 M，需要安装一段时间

安装完后启动mysql服务：

service mysqld start

2.2 配置mysql

配置前需要查看MySQL的临时root密码

grep 'temporary password' /var/log/mysqld.log

mysql_secure_installation

输入当前 root 用户密码，即MySQL的临时root密码

Enter current password for root :

要设置 root 密码吗？输入 y 表示愿意

Set root password? [Y/n] y

要移除掉匿名用户吗？输入 y 表示愿意

Remove anonymous users? [Y/n] y

不想让 root 远程登陆吗？输入 y 表示愿意

Disallow root login remotely? [Y/n] y

要去掉 test 数据库吗？输入 y 表示愿意

Remove test database and access to it? [Y/n] y

想要重新加载权限吗？输入 y 表示愿意

Reload privilege tables now? [Y/n] y

2.3 测试MySQL

```
[root@bogon src]# mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 15
Server version: 8.0.16 MySQL Community Server - GPL

Copyright (c) 2000, 2019, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

屏幕剪辑的捕获时间: 2019/7/22 8:24

创建用户：

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

创建数据库，出于兼容性考虑，在新建数据库的时候指定字符集：

```
CREATE DATABASE databasename default charset utf8 collate utf8_general_ci;
```

授权：

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

设置与更改用户密码：

```
SET PASSWORD FOR 'username'@'host' = PASSWORD('newpassword');
```

如果是当前登录用户：

```
SET PASSWORD = PASSWORD("newpassword");
```

3. 安装php

3.1 安装稳定版本

wget https://www.php.net/distributions/php-7.3.7.tar.gz

直接./configure会报错，提示安装依赖。

yum -y install libjpeg libjpeg-devel libpng libpng-devel freetype freetype-devel libxml2 libxml2-devel pcre-devel

yum -y install curl-devel

yum -y install libxslt-devel

libmcrypt、mcrypt、mhash这三个需要下载安装包：

wget -c https://sourceforge.net/projects/mcrypt/files/MCrypt/2.6.8/mcrypt-2.6.8.tar.gz

wget -c https://sourceforge.net/projects/mcrypt/files/Libmcrypt/2.5.8/libmcrypt-2.5.8.tar.gz

wget -c https://sourceforge.net/projects/mhash/files/mhash/0.9.9.9/mhash-0.9.9.9.tar.gz

先安装Libmcrypt

#tar -zxvf libmcrypt-2.5.8.tar.gz

#cd libmcrypt-2.5.8

#./configure

#make

#make install 说明：libmcript默认安装在/usr/local

安装mhash

#tar -zxvf mhash-0.9.9.9.tar.gz

#cd mhash-0.9.9.9

#./configure

#make

#make install

安装mcrypt

#tar -zxvf mcrypt-2.6.8.tar.gz

#cd mcrypt-2.6.8

#LD_LIBRARY_PATH=/usr/local/lib ./configure

#make

#make install

至此，PHP需要的扩展环境基本上就搭好了，下面开始正式编译安装：

./configure --prefix=/usr/local/php --with-config-file-path=/usr/local/php/etc --with-pdo-mysql --with-mysqli --with-gd --with-zlib --with-mcrypt --enable-fpm --enable-soap --enable-sockets --enable-mbstring=all

解释配置行含义：

--prefix=/usr/local/php ： 表示将php安装到/usr/local/php目录

--with-config-file-path=/usr/local/php/etc ： 表示将php配置文件存放在/usr/local/php/etc

-with-pdo-mysql ： 需要安装mysql原生模块

--with-mysqli ： 需要安装mysql 增强模块

--enable-fpm ： 需要安装fpm模块，这个很重要，需要它才能打通nginx

运行结果最后一条WARNING：

configure: WARNING: unrecognized options: --with-mcrypt

如果没有任何问题，即不报error之类（warning不算），那么我们就可以开始编译了，很简单就一条指令：

make && make install

然后屏幕开始翻滚，这个过程非常漫长，大概会在半个小时之后完成。

3.2 编辑配置文件

3.2.1 php.ini

在php的安装目录下找不到php.ini，从安装包中拷贝配置文件到目标路径下：

cp /home/centos/src/php-7.3.7/php.ini-production /usr/local/php/lib/php.ini

vim php.ini

找到：cgi.fix_pathinfo

改为：cgi.fix_pathinfo=0

3.2.2 php-fpm.conf

cd /usr/local/php/etc/php-fpm.conf.default /usr/local/php/etc/php-fpm.conf

vim php-fpm.conf

找到：pid = run/php-fpm.pid，取消注释

3.2.3 www.conf

cp /usr/local/php/etc/php-fpm.d/www.conf.default /usr/local/php/etc/php-fpm.d/www.conf

vim www.conf

修改：

user = www

group = www

listen = 127.0.0.1:9000

3.2.4 nginx.conf

修改nginx.conf ，相当于就把nginx和php-fpm连通了

cd /usr/local/nginx/conf

vim nginx.conf

对原文件以下几行取消注释：

```
# 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;
}
```

屏幕剪辑的捕获时间: 2019/7/24 15:55

另外还要在系统中添加一个组和一个账号，执行以下命令：

groupadd www

useradd -g www www

至此配置文件修改结束。

3.3 启动php-fpm和nginx

3.3.1 首先启动php-fpm

#/usr/local/php/sbin/php-fpm

3.3.2 然后启动nginx

#/usr/local/nginx/sbin/nginx

如果没有报错，就可以进行下一步了，这个地方报错的可能性也比较大，多半都是因为上面4个配置文件没有设置好导致，几乎每个设置文件的目录下都有一个默认设置文件备份，改得太乱大不了重新恢复到原始状态再重新折腾即可，例如：

cp nginx.conf.default nginx.conf

3.4 测试php文件是否能运行

去nginx的主目录下创建一个php测试文件

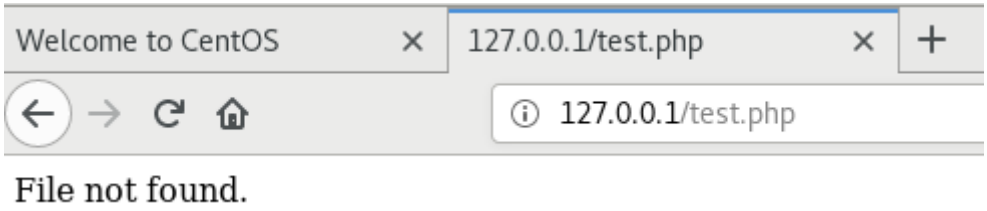
cd /usr/local/nginx/html

vim test.php

内容只有一行：

```
<?php phpinfo(); ?>
```

然后去浏览器里输入http://127.0.0.1/test.php，文件找不到？？？



屏幕剪辑的捕获时间: 2019/7/24 15:51

```
[root@bogon html]# cat test.php
<?php
phpinfo();
?>
```

屏幕剪辑的捕获时间: 2019/7/24 15:52

明明已经创建了文件。

判断应该是nginx.conf配置出了问题：

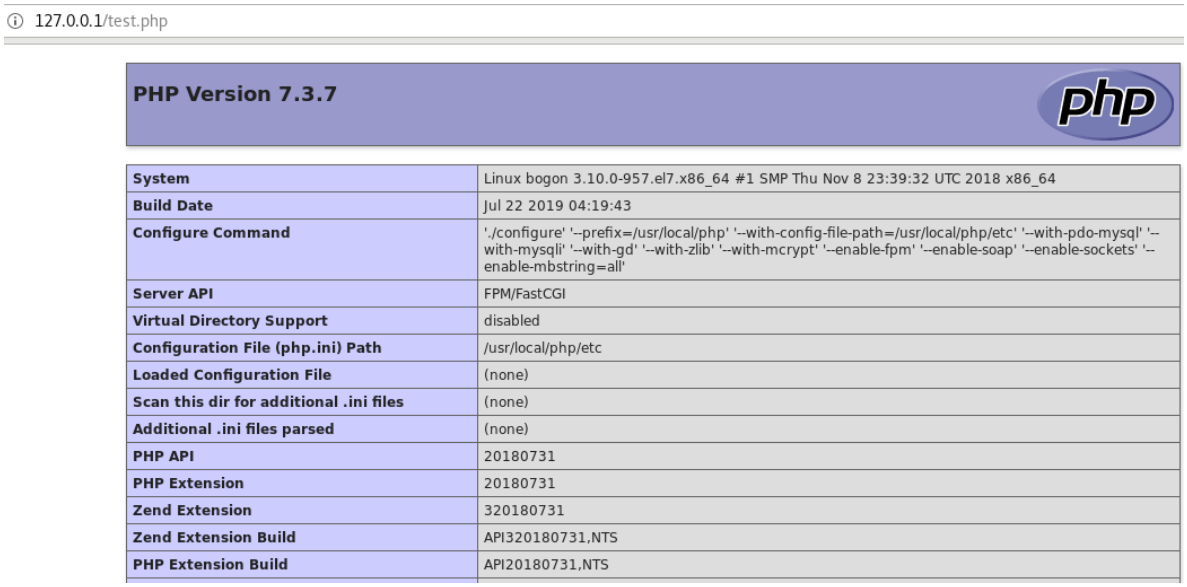
上面对原文取消注释的地方，将原/scripts修改成$document_root，修改后为：

fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;

保存退出。重新加载nginx：

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

终于看到这个画面：



屏幕剪辑的捕获时间: 2019/7/24 16:03

3.5 测试php连接mysql

实例PDO

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
```

```php
try {
    $conn = new PDO("mysql:host=$servername;", $username, $password);
    echo "connect success!!!";
}
catch(PDOException $e)
{
    echo $e->getMessage();
}
?>
```

执行脚本时报错：SQLSTATE[HY000] [2002] No such file or directory

出现这个问题的原因是PDO无法找到mysql.sock或者mysqld.sock。

解决方法1：找到相应的.sock文件，并设置php.ini文件中的pdo_mysql.default_socket的值为.sock文件的路径。

找到两个跟MySQL相关的.sock文件，分别将对应路径添加到php.ini配置文件。

```
[root@bogon html]# find / -name *.sock
find: ‘/proc/119155’: No such file or directory
/run/chrony/chronyd.sock
/run/gssproxy.sock
/run/rpcbind.sock
/run/mysqld/mysqlx.sock
find: ‘/run/user/1000/gvfs’: Permission denied
/run/cups/cups.sock
/var/lib/gssproxy/default.sock
/var/lib/mysql/mysql.sock
```

屏幕剪辑的捕获时间: 2019/7/25 11:49

修改完php.ini配置文件，需要重启nginx和php-fpm服务。

重启nginx方法：nginx -s reload

重启php-fpm方法：

kill -INT `cat /usr/local/php/var/run/php-fpm.pid`

或者：pkill php-fpm

然后再运行php-fpm启动。

但是，执行脚本还是报之前的错误。

解决方法2：将PDO连接中的dsn的host由“localhost”改为“127.0.0.1”

继续报错：SQLSTATE[HY000] [2054] The server requested authentication method unknown to the client

发生这种错误，是由于MySQL 8默认使用了新的密码验证插件：caching_sha2_password，而之前的PHP版本中所带的mysqlnd无法支持这种验证。解决这个问题，有两种办法。

一种办法是升级PHP支持MySQL 8的新验证插件。PHP 7.2.8和PHP 7.1.20已经可以支持caching_sha2_password，直接连接MySQL 8。但是，为什么PHP 7.3.7不支持呢？

mysqlnd

mysqlnd	enabled
Version	mysqlnd 5.0.12-dev - 20150407 - $Id: 7cc7cc96e675f6d72e5cf0f267f48e167c2abb23 $
Compression	supported
core SSL	supported
extended SSL	not supported
Command buffer size	4096
Read buffer size	32768
Read timeout	86400
Collecting statistics	Yes
Collecting memory statistics	No
Tracing	n/a
Loaded plugins	mysqlnd,debug_trace,auth_plugin_mysql_native_password,auth_plugin_mysql_clear_password
API Extensions	pdo_mysql,mysqli

屏幕剪辑的捕获时间: 2019/7/25 9:45

经查验，最新的 PHP 7.2.12 版本 和 PHP 7.3.0 版本都不支持 caching_sha2_password。

方法二，MySQL 8中创建（或修改）使用caching_sha2_password插件的账户，使之使用mysql_native_password

1. 在CREATE USER时，使用IDENTIFIED WITH xxx_plugin BY 'password'，比如：

CREATE USER 'native'@'localhost' IDENTIFIED WITH mysql_native_password BY 'password!2#4';

2. 使用ALTER USER修改已有账户的验证插件：

ALTER USER 'native'@'localhost' IDENTIFIED WITH mysql_native_password

或

ALTER USER 'native'@'localhost' IDENTIFIED WITH mysql_native_password BY 'new_password';

采用前一种方式，账户的密码将被清除；BY子句将为账户设置新的密码。

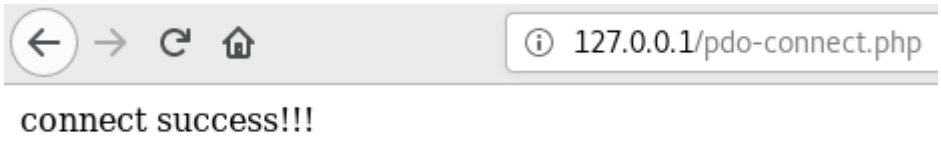
FLUSH PRIVILEGES

3. /etc/my.cnf配置文件中，有一行：

default-authentication-plugin=mysql_native_password

删除注释符号"#"，并重新启动mysqld使之生效，此后创建的账户均默认使用mysql_native_password。

php脚本可以成功连接mysql



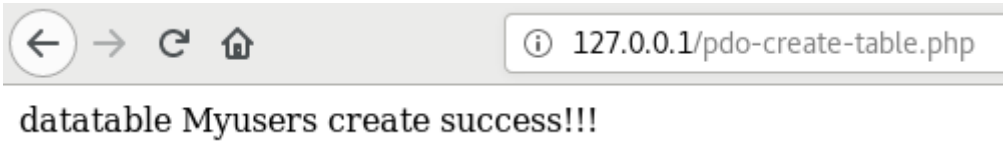
屏幕剪辑的捕获时间: 2019/7/25 10:16

3.6 PHP MySQL创建数据表

实例PDO

```
<?php
$servername = "127.0.0.1";
$username = "username";
$password = "password";
$dbname = "myDBPDO";

try {

    $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username,
$password);
    // 设置 PDO 错误模式，用于抛出异常
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    // 使用 sql 创建数据表
    $sql = "CREATE TABLE MyGuests (
    id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
    firstname VARCHAR(30) NOT NULL,
    lastname VARCHAR(30) NOT NULL,
    email VARCHAR(50),
    reg_date TIMESTAMP
    )";

    // 使用 exec() ，没有结果返回
    $conn->exec($sql);
     echo "datatable Myusers create success!!!";
}
catch(PDOException $e)
{
    echo $sql . "<br>" . $e->getMessage();
}

$conn = null;
?>
```

脚本执行成功返回结果：



屏幕剪辑的捕获时间: 2019/7/25 10:55

查看MySQL数据库，添加数据表成功：

```
mysql> show tables;
+-----------------+
| Tables_in_test1 |
+-----------------+
| MyUsers         |
+-----------------+
1 row in set (0.00 sec)

mysql> describe MyUsers;
+-----------+-----------------+------+-----+---------+----------------+
| Field     | Type            | Null | Key | Default | Extra          |
+-----------+-----------------+------+-----+---------+----------------+
| id        | int(6) unsigned | NO   | PRI | NULL    | auto_increment |
| firstname | varchar(30)     | NO   |     | NULL    |                |
| lastname  | varchar(30)     | NO   |     | NULL    |                |
| email     | varchar(50)     | YES  |     | NULL    |                |
| reg_date  | timestamp       | YES  |     | NULL    |                |
+-----------+-----------------+------+-----+---------+----------------+
5 rows in set (0.01 sec)
```

屏幕剪辑的捕获时间: 2019/7/25 11:00

二、安全加固

1. PHP安全配置

1.1 确保运行php的用户为一般用户，如www

```
www      112682  0.0  0.1 158416  2460 ?        S    Jul24   0:00 php-fpm: pool
www
www      112683  0.0  0.1 158416  2696 ?        S    Jul24   0:00 php-fpm: pool
www
```

屏幕剪辑的捕获时间: 2019/7/25 14:51

1.2 php.ini参数设置

disable_functions = passthru,exec,system,chroot,chgrp,chown,shell_exec,proc_open,proc_get_status,ini_alter,ini_restore,dl,openlog,syslog,readlink,symlink,popepassthru,stream_socket_server,fsocket,phpinfo #禁用的函数，默认为空

expose_php = off #避免暴露PHP信息，默认为On

display_errors = off

enable_dl = off

allow_url_include = off

session.cookie_httponly = 1 #默认为空，禁止 JavaScript 访问会话 cookie。 此设置项可以保护 cookie 不被 JavaScript 窃取。

upload_tmp_dir = /tmp #默认为空，临时路径用于存储上传的文件，需要 php 进程有读写权限。如果不指定将使用系统默认设置。

open_basedir = ./:/tmp:/home/wwwroot/ #默认为空，将用户访问文件的活动范围限制在指定的区域，通常是其家目录的路径，也可用符号"."来代表当前目录。从网上获取的资料来看，open_basedir会对php操作io的性能产生很大的影响。研究资料表明，配置了php_basedir的脚本io执行速度会比没有配置的慢10倍甚至更多。

2. MySQL安全设置

2.1 MySQL版本选择

在正式生产环境中，禁止使用4.1系列的MySQL数据库。至少需要使用5.1.39或以上版本。

2.2 网络和端口的配置

在数据库只需供本机使用的情况下，使用--skip-networking参数禁止监听网络 。

2.3 确保运行MySQL的用户为一般用户，如mysql，注意存放数据目录权限为mysql

vim /etc/my.conf

user = mysql

```
[root@bogon html]# ps aux | grep mysql
root      19836  0.0  0.0 112708   976 pts/0    S+   04:48   0:00 grep --color=a
uto mysql
mysql    101788  1.4 19.2 1833120 357944 ?      Ssl  Jul24   5:24 /usr/sbin/mysq
ld
```

屏幕剪辑的捕获时间: 2019/7/25 16:55

2.4 开启mysql二进制日志，在误删除数据的情况下，可以通过二进制日志恢复到某个时间点

vim /etc/my.conf

log_bin = mysql-bin

expire_logs_days =7

2.5 认证和授权

（1）禁止root账号从网络访问数据库，root账号只允许来自本地主机的登陆。

mysql>grant all privileges on . to root @localhost identified by 'password' with grant option;

mysql>flush priveleges;

（2）删除匿名账号和空口令账号

mysql>USE mysql;

mysql>delete from user where User=;

mysql>delete from user where Password=;

mysql>delete from db where User=;

2.6 设置可信 IP 访问控制

通过数据库所在操作系统的防火墙限制，实现只有信任的 IP 才能通过监听器访问数据库。

```
GRANT ALL PRIVILEGES ON db.* TO 用户名@'IP子网/掩码';
```

2.7 连接数设置

根据机器性能和业务需求，设置最大、最小连接数。在 MySQL 配置文件（my.conf 或 my.ini）的 [mysqld] 配置段中添加max_connections = 1000，保存配置文件，重启 MySQL 服务后即可生效。

2.8 php处理MySQL数据库使用预处理语句

预处理语句用于执行多个相同的 SQL 语句，并且执行效率更高。

预处理语句的工作原理如下：

1. 预处理：创建 SQL 语句模板并发送到数据库。预留的值使用参数 "?" 标记 。例如：

```
INSERT INTO MyGuests (firstname, lastname, email) VALUES(?, ?, ?)
```

2. 数据库解析，编译，对SQL语句模板执行查询优化，并存储结果不输出。
3. 执行：最后，将应用绑定的值传递给参数（"?" 标记），数据库执行语句。应用可以多次执行语句，如果参数的值不一样。

相比于直接执行SQL语句，预处理语句主要优点：

- 预处理语句大大减少了分析时间，只做了一次查询（虽然语句多次执行）。
- 绑定参数减少了服务器带宽，你只需要发送查询的参数，而不是整个语句。
- 预处理语句针对SQL注入是非常有用的，因为参数值发送后使用不同的协议，保证了数据的合法性。

MySQLi 和 PDO 两者都支持预处理语句。 预处理语句可以防止 SQL 注入，对于 web 项目的安全性是非常重要的。

3. web服务器安全

3.1 确保运行Nginx或者Apache的用户为一般用户，如www，注意存放数据目录权限为www

```
[root@bogon html]# ps aux | grep nginx
www       32056  0.0  0.0  23144  1804 ?        S    04:57   0:00 nginx: worker
process
root      40121  0.0  0.0 112708   976 pts/0    S+   05:02   0:00 grep --color=a
uto nginx
root      86708  0.0  0.0  20612  1356 ?        Ss   Jul24   0:00 nginx: master
process nginx
```

屏幕剪辑的捕获时间: 2019/7/25 17:03

3.2 防止sql注入

```
if ( $query_string ~* ".*[\;'\<\>].*" ){
    return 404;
}
```

3.3 关闭存放数据上传等目录的PHP解析

```
location ~* ^/(attachments|data)/.*\.(php|php5)${
    deny all;
}
```

3.4 针对Apache：关闭图片目录/上传等目录的PHP解析

```
<Files ~ ".php">
order allow,deny
Deny from all
</Files>
```

三、 LAMP环境搭建

之前的LEMP环境已经搭建好了，所以这里只安装Apache，为了避免冲突，先关闭nginx。

1. 安装Apache

yum -y install httpd

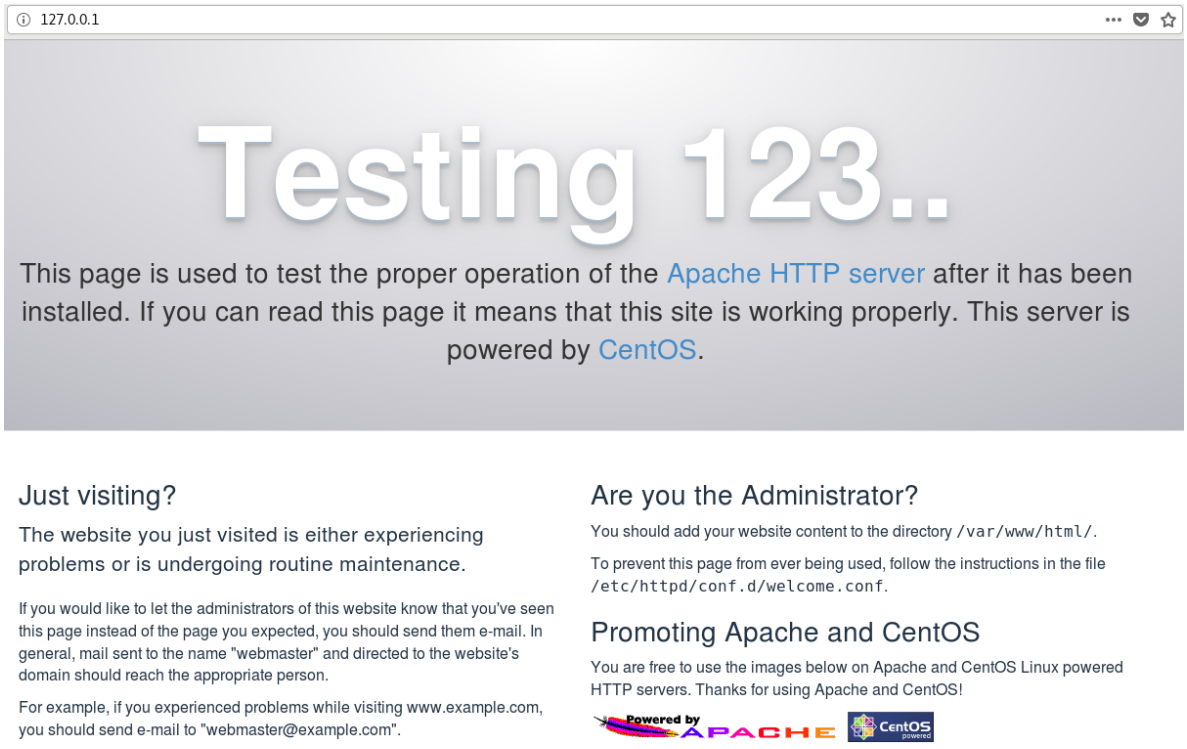
安装完成后，打开 httpd 的配置文件，

vim /etc/httpd/conf/httpd.conf

把 ServerName 前的 # 去掉，并修改为：ServerName localhost 并保存，启动httpd：

service httpd start

访问：http://127.0.0.1，出现如下界面，代表Apache安装成功：



屏幕剪辑的捕获时间: 2019/7/26 10:48

2. Apache处理php代码

在 Apache 的默认网站目录添加 phpinfo.php 文件：

vim /var/www/html/phpinfo.php

浏览器访问：http://127.0.0.1/phpinfo.php，浏览器返回空页面，没有出现想要的信息。初步判断是之前配置php.ini文件禁用了phpinfo等函数生效了。

修改完php.ini配置文件后，重启httpd服务，依然解析不了php文件。

尝试关闭php-fpm，还是无效。

经查阅资料，这种情况是由于Apache没有配置好运行PHP需要的环境导致的。

由于本次环境搭建先搭建了LEMP环境，PHP已经按照指定的"--enable-fpm"编译选项来启动php-fpm，而常规搭建LAMP环境，编译PHP需要指定编译选项"--with-apxs2=/usr/local/apache2/bin/apxs"，来使用Apache的内置模块mod_php来运行PHP。

正常yum源安装Apache没有apxs（apxs是一个为Apache HTTP服务器编译和安装扩展模块的工具，用于编译一个或多个源程序或目标代码文件为动态共享对象，使之可以用由mod_so提供的LoadModule指令在运行时加载到Apache服务器中，简单点说apxs就是实现apache扩展功能使php和web服务结合使用），只有在以源码包方式编译安装才有。

解决办法：

用yum list httpd*查找资源，安装httpd-devel套件包：

yum install httpd-devel.x86_64

安装之后就有apxs了：

```
[root@bogon html]# which apxs
/usr/bin/apxs
```

屏幕剪辑的捕获时间: 2019/7/26 15:11

编译选项应为："--with-apxs2=/usr/bin/apxs"，重新编译PHP。

编辑apache配置文件httpd.conf，以apache支持php

vim /etc/httpd/httpd.conf

添加如下二行

AddType application/x-httpd-php .php

AddType application/x-httpd-php-source .phps

定位至DirectoryIndex index.html

修改为：

DirectoryIndex index.php index.html

而后重新启动httpd，或让其重新载入配置文件即可测试php是否已经可以正常使用。此时发现使用rpm格式的httpd也能和mysql以及编译安装的php构建LAMP平台了。虽然编译安装比较麻烦，但还是建议使用编译安装实现LAMP平台的构建。

原来的 mod_php 采用 SetHandler 的方式处理 php 文件并不需要特别的设置，因为在安装 PHP 的时候会自动在 Apache 的配置文件目录写入一个 php.conf 的配置文件，里面有告诉 Apache 处理 php 需要的操作：

```
<FilesMatch \.php$>
    SetHandler application/x-httpd-php
</FilesMatch>
```

这是很方便的。因为只要 Apache 遇到 .php 类型的文件就会知道要让 mod_php 来解释运行。这种方式是在收到每个请求之后才处理的，所以可以用于全局。

3. 从Apache mod_php到php-fpm

mod_php和php-fpm是运行php的两种基本方法

mod_php是Apache的内置php解释模块，使用prefork方式，不需要额外的进程来做通讯和应用解释，即php请求在Apache进程中运行，php使用Apache权限运行。缺点是把应用和HTTP服务器绑定在了一起，另外每个Apache进程都需要加载mod_php而不论这个请求是处理静态内容还是动态内容，这样导致浪费内存，效率下降，此外php.ini文件的变更需要重新启动apache服务器才能生效，这使得无法进行平滑配置变更。

php-fpm作为独立的Fast-CGI服务器运行，常和nginx搭配使用的程序，php-fpm实际上就是对FASTCGI协议的一个加强实现，已经被纳入PHP内核，可以通过--enable-fpm编译选项来启用，php-fpm支持配置的平滑变更（通过fork新的worker进程），性能好，内存使用效率高，这也是为什么nginx+php-fpm的配置组合会替代apache+mod_cgi以及apache+mod_php的重要原因。

Apache 官方网站也提供了配置 Apache httpd 2.4.x 使用 mod_proxy_fcgi 和 PHP-FPM 运行 php 程序的基本方法和设置运行方式的简单介绍。使用 PHP-FPM 就意味着不用 Apache 内置的 mod_php，也就是要在 Apache 之外处理 php 程序的解释运行问题。看起来是多引入了一个额外的程序 PHP-FPM，既占 CPU 又占内存。但是这样一来，因为 Apache 可以专心处理除 php 之外的静态网页及元素，反而 httpd 进程本身占用的 CPU 和内存可以显著降低，从而从整体上降低资源消耗。

编辑httpd.conf，添加以下内容：

```
<IfModule mpm_event_module>
    ProxyPassMatch ^/(.*\.php(/.*)?)$ fcgi://127.0.0.1:9000/path/to/webroot/$1
</IfModule>
```

找到DirectoryIndex，改成如下：

DirectoryIndex /index.php

启动php-fpm，重启httpd，访问http://127.0.0.1/phpinfo.php，还是无法执行。

httpd.conf中没有配置启动mod_proxy_fcgi.so模块，添加以下启动模块：

#php-fpm模块

LoadModule proxy_module /usr/lib64/httpd/modules/mod_proxy.so

LoadModule proxy_fcgi_module /usr/lib64/httpd/modules/mod_proxy_fcgi.so

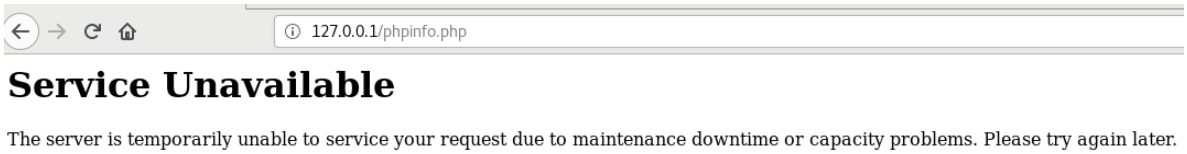
#url rewrite模块

LoadModule rewrite_module /usr/lib64/httpd/modules/mod_rewrite.so

#首页默认文件

```
<IfModule dir_module>
    DirectoryIndex index.html index.php
</IfModule>
```

一顿配置后：



屏幕剪辑的捕获时间: 2019/7/26 17:38

尝试直接关掉SElinux，依然无效。

还是不死心，网上查阅了通过单机的方式配置Apache和php-fpm相结合，发现LoadModule不是在httpd.conf中配置的，在00-proxy.conf 已经配置了启动模块：

[root@bogon Linux]# cat /etc/httpd/conf.modules.d/00-proxy.conf

This file configures all the proxy modules:

LoadModule proxy_module modules/mod_proxy.so

LoadModule lbmethod_bybusyness_module modules/mod_lbmethod_bybusyness.so

LoadModule lbmethod_byrequests_module modules/mod_lbmethod_byrequests.so

LoadModule lbmethod_bytraffic_module modules/mod_lbmethod_bytraffic.so

LoadModule lbmethod_heartbeat_module modules/mod_lbmethod_heartbeat.so

LoadModule proxy_ajp_module modules/mod_proxy_ajp.so

LoadModule proxy_balancer_module modules/mod_proxy_balancer.so

LoadModule proxy_connect_module modules/mod_proxy_connect.so

LoadModule proxy_express_module modules/mod_proxy_express.so

LoadModule proxy_fcgi_module modules/mod_proxy_fcgi.so

LoadModule proxy_fdpass_module modules/mod_proxy_fdpass.so

LoadModule proxy_ftp_module modules/mod_proxy_ftp.so

LoadModule proxy_http_module modules/mod_proxy_http.so

LoadModule proxy_scgi_module modules/mod_proxy_scgi.so

LoadModule proxy_wstunnel_module modules/mod_proxy_wstunnel.so

重新修改httpd.conf配置文件：

1. 关闭反向代理，把以.php结尾的文件请求发送到php-fpm进程，php-fpm至少需要知道运行的目录和URI，所以这里直接在fcgi://xxx.x.x.x:9000后指明了这两个参数，其它的参数的传递已经被mod_proxy_fcgi.so进行了封装，不需要手动指定。

```
# DocumentRoot: The directory out of which you will serve your
# documents. By default, all requests are taken from this directory, but
# symbolic links and aliases may be used to point to other locations.
#
DocumentRoot "/var/www/html"
ProxyRequests Off
ProxyPassMatch ^/(.*\.php(/.*)?)$ fcgi://127.0.0.1:9000/var/www/html/$1
```

屏幕剪辑的捕获时间: 2019/7/29 16:08

2. 添加主页识别

```
<IfModule dir_module>
    DirectoryIndex index.html index.php
</IfModule>
```

屏幕剪辑的捕获时间: 2019/7/29 16:11

3. 添加页面类型识别

```
#AddEncoding x-compress .Z
#AddEncoding x-gzip .gz .tgz
#
# If the AddEncoding directives above are commented-out, then you
# probably should define those extensions to indicate media types:
#
AddType application/x-compress .Z
AddType application/x-gzip .gz .tgz
AddType application/x-httpd-php .php
AddType application/x-httpd-php-source .phps
#
```

屏幕剪辑的捕获时间: 2019/7/29 16:11

终于看到这个熟悉的界面，可以看到Server API为FPM，Server_Software为Apache/2.4.6 (CentOS)：

127.0.0.1/phpinfo.php

PHP Version 7.3.7

System	Linux bogon 3.10.0-957.21.3.el7.x86_64 #1 SMP Tue Jun 18 16:35:19 UTC 2019 x86_64
Build Date	Jul 22 2019 04:19:43
Configure Command	'./configure' '--prefix=/usr/local/php' '--with-config-file-path=/usr/local/php/etc' '--with-pdo-mysql' '--with-mysqli' '--with-gd' '--with-zlib' '--with-mcrypt' '--enable-fpm' '--enable-soap' '--enable-sockets' '--enable-mbstring=all'
Server API	FPM/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/usr/local/php/etc
Loaded Configuration File	(none)
Scan this dir for additional .ini files	(none)
Additional .ini files parsed	(none)
PHP API	20180731
PHP Extension	20180731
Zend Extension	320180731
Zend Extension Build	API320180731,NTS
PHP Extension Build	API20180731,NTS
Debug Build	no
Thread Safety	disabled

屏幕剪辑的捕获时间: 2019/7/29 15:55

PHP Variables

Variable	Value
$_SERVER['USER']	www
$_SERVER['HOME']	/home/www
$_SERVER['SCRIPT_NAME']	/phpinfo.php
$_SERVER['REQUEST_URI']	/phpinfo.php
$_SERVER['QUERY_STRING']	*no value*
$_SERVER['REQUEST_METHOD']	GET
$_SERVER['SERVER_PROTOCOL']	HTTP/1.1
$_SERVER['GATEWAY_INTERFACE']	CGI/1.1
$_SERVER['REMOTE_PORT']	60456
$_SERVER['SCRIPT_FILENAME']	/var/www/html/phpinfo.php
$_SERVER['SERVER_ADMIN']	root@localhost
$_SERVER['CONTEXT_DOCUMENT_ROOT']	/var/www/html
$_SERVER['CONTEXT_PREFIX']	*no value*
$_SERVER['REQUEST_SCHEME']	http
$_SERVER['DOCUMENT_ROOT']	/var/www/html
$_SERVER['REMOTE_ADDR']	127.0.0.1
$_SERVER['SERVER_PORT']	80
$_SERVER['SERVER_ADDR']	127.0.0.1
$_SERVER['SERVER_NAME']	127.0.0.1
$_SERVER['SERVER_SOFTWARE']	Apache/2.4.6 (CentOS)

屏幕剪辑的捕获时间: 2019/7/29 16:20

四、 LNMP + Apache 架构配置

在查阅资料的过程中，看到了一个比较有意思的配置：Nginx 做前端代理 + Apache 做后端服务。

以下为摘录：

Apache 和 Nginx 说是当今最流行的两个 Web 服务器一点也不为过，Apache 用户基数大，稳定，兼容性高（比如jsp/php/cgi/python等等），但与 Ngnix 相比，Apache过于臃肿以及对静态文件响应过于缓慢让很多使用者感到头疼，而 Nginx 对于高并发性能出众，Proxy 功能强效率高，占用系统资源少。

但是 Nginx 也有劣势，它在处理 php 脚本时需要通过 php-fpm(FastCGI) 解析，而 php-fpm 不够稳定，经常出现 502 错误，生成相对复杂的页面没有优势，反而会使 php-cgi 进程变为僵尸进程。而 Apache 在高并发时对队列的处理比 FastCGI 更好，并且在处理动态 php 页面时，mod_php 模块也比 php-cgi 模块更稳定更高效。

事实上很多大型的网站都是采用 Nginx 前端 + Apache 后端的服务器架构，这样可以很好地结合了 Nginx 高并发和静态页面高效率以及 Apache 稳定的动态页面处理特点，再也不用担心 Nginx 以 FastCGI 模式运行 PHP 时的502问题和 Apache 处理静态页面过慢、负载过高的问题。

解决思路就是让Nginx做代理，将运行PHP脚本请求交由Apache来处理。

打开编辑 Nginx 的默认配置文件：

vim /etc/nginx/conf.d/default.conf

注释掉之前 FastCGI 监听的配置，并添加如下代码：

```
# proxy the PHP scripts to Apache listening on 127.0.0.1:8080
location ~ \.php$ {
    proxy_pass    http://127.0.0.1:8080
}
```

接下来打开编辑 Apache 的配置文件：

vim /etc/httpd/conf/httpd.conf

找到 Listen 字段，并改为：Listen 127.0.0.1:8080，让 Apache 来监听这个端口，修改 Apache 的网站根目录为：”/usr/share/nginx/html”，与上述 Nginx 对应的网站目录保持一致：

```
Listen 127.0.0.1:8080
...
ServerName localhost
...
DocumentRoot "/usr/share/nginx/html"
...
<Directory "/usr/share/nginx/html">
    ...
</Directory>
...
```

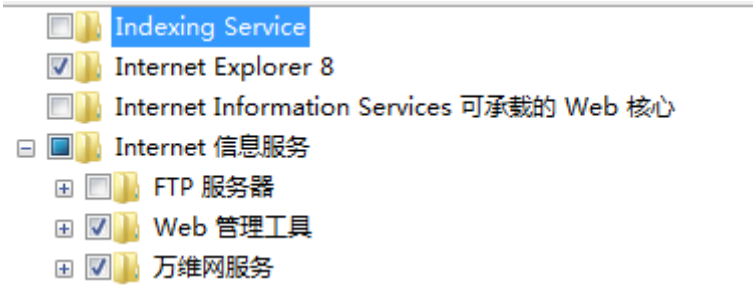
保存退出编辑，重启 Nginx 服务，并启动 Apache 服务。

五、Windows server的IIS环境搭建

1. 安装IIS

用win7替代windows server来进行IIS环境搭建。

控制面板-程序和功能-打开或关闭Windows功能，勾选Internet信息服务选项下的内容：



屏幕剪辑的捕获时间: 2019/7/29 17:28

本地访问http://127.0.0.1，出现如下界面证明IIS7安装成功：

屏幕剪辑的捕获时间: 2019/7/29 17:30

2. 安装PHP

2.1 选择安装7.3版本，VC15 x64 Thread Safe

需要依赖MSVC15 (Visual C++ 2017)

PHP 7.3 (7.3.7)

Download source code [26.78MB]

Download tests package (phpt) [14.16MB]

VC15 x64 Non Thread Safe (2019-Jul-03 17:49:08)

- Zip [24.34MB]
  sha256: 1b1a74962752ff6278ef7bcb811af951d25583e9da5d81fa93e90d54616a6db3
- Debug Pack [22.93MB]
  sha256: e1d798dc467c818f0308eaada9a530fdb740297fcd5c6faf176a87bf2f031b25
- Development package (SDK to develop PHP extensions) [1.24MB]
  sha256: eff175a4d80db7b13986facfe032800da9234f439d13d49c0ada462479d50227

VC15 x64 Thread Safe (2019-Jul-03 17:49:17)

- Zip [24.47MB]
  sha256: 2f8f0b3734d1bd574ae05f5d79f90aa40a5e7f330bb7052973478c68f7622b18

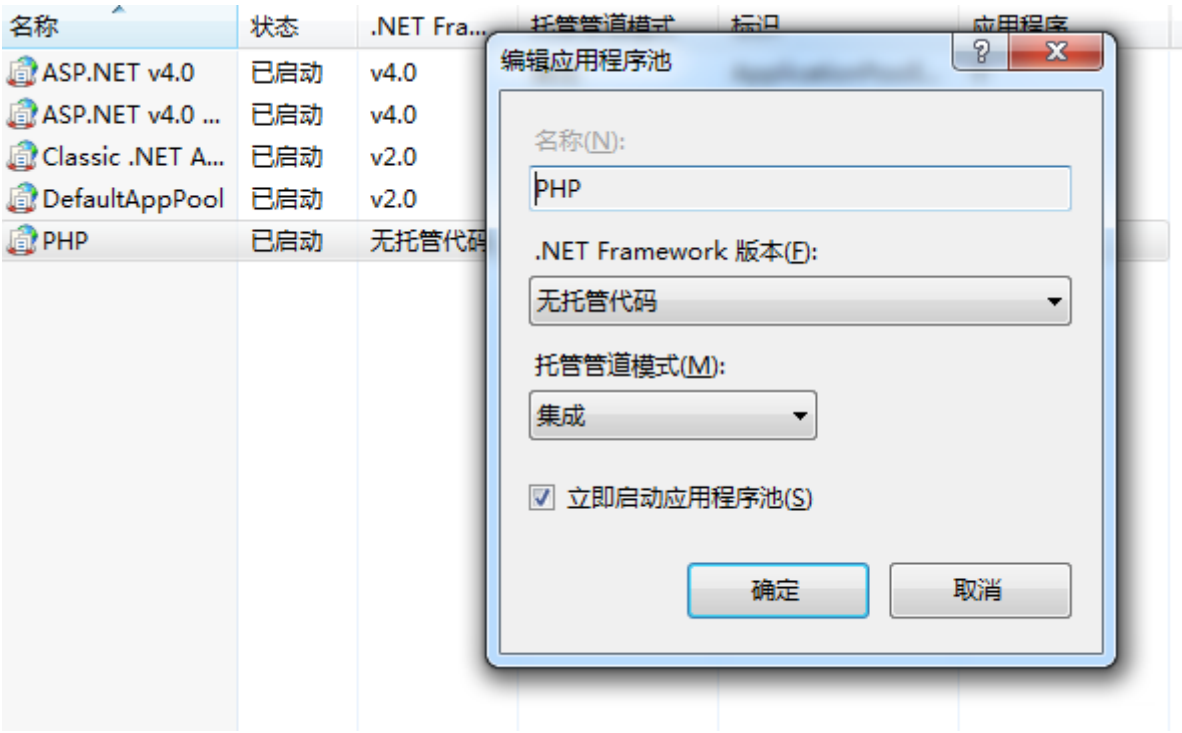
屏幕剪辑的捕获时间: 2019/7/29 17:41

解压到指定目录，并添加环境变量：

```
C:\Users\john>php -v
PHP 7.3.7 (cli) (built: Jul  3 2019 14:34:10) ( ZTS MSVC15 (Visual C++ 2017) x64
 )
Copyright (c) 1997-2018 The PHP Group
Zend Engine v3.3.7, Copyright (c) 1998-2018 Zend Technologies
```

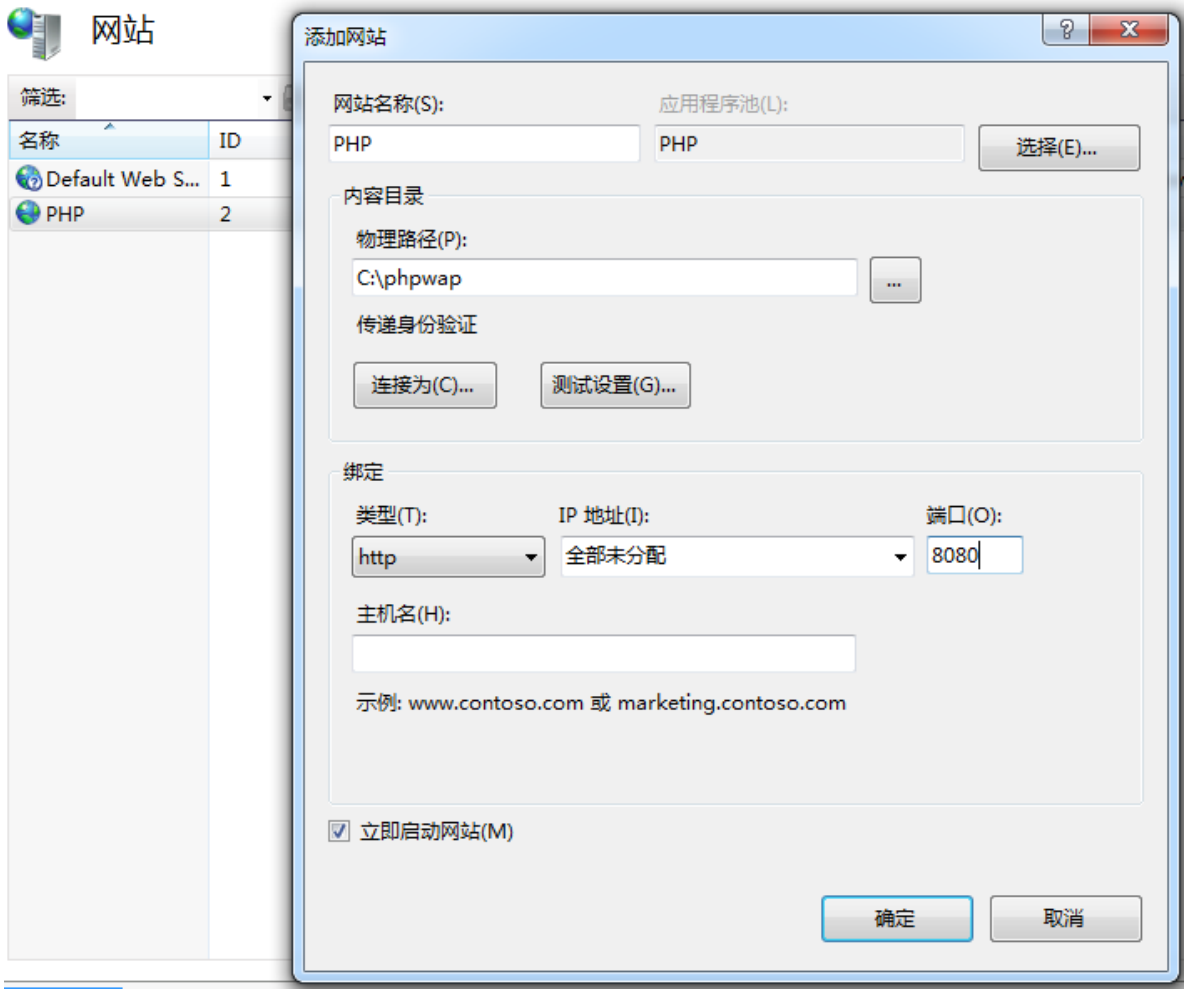
屏幕剪辑的捕获时间: 2019/7/29 20:02

2.2 编译应用程序池



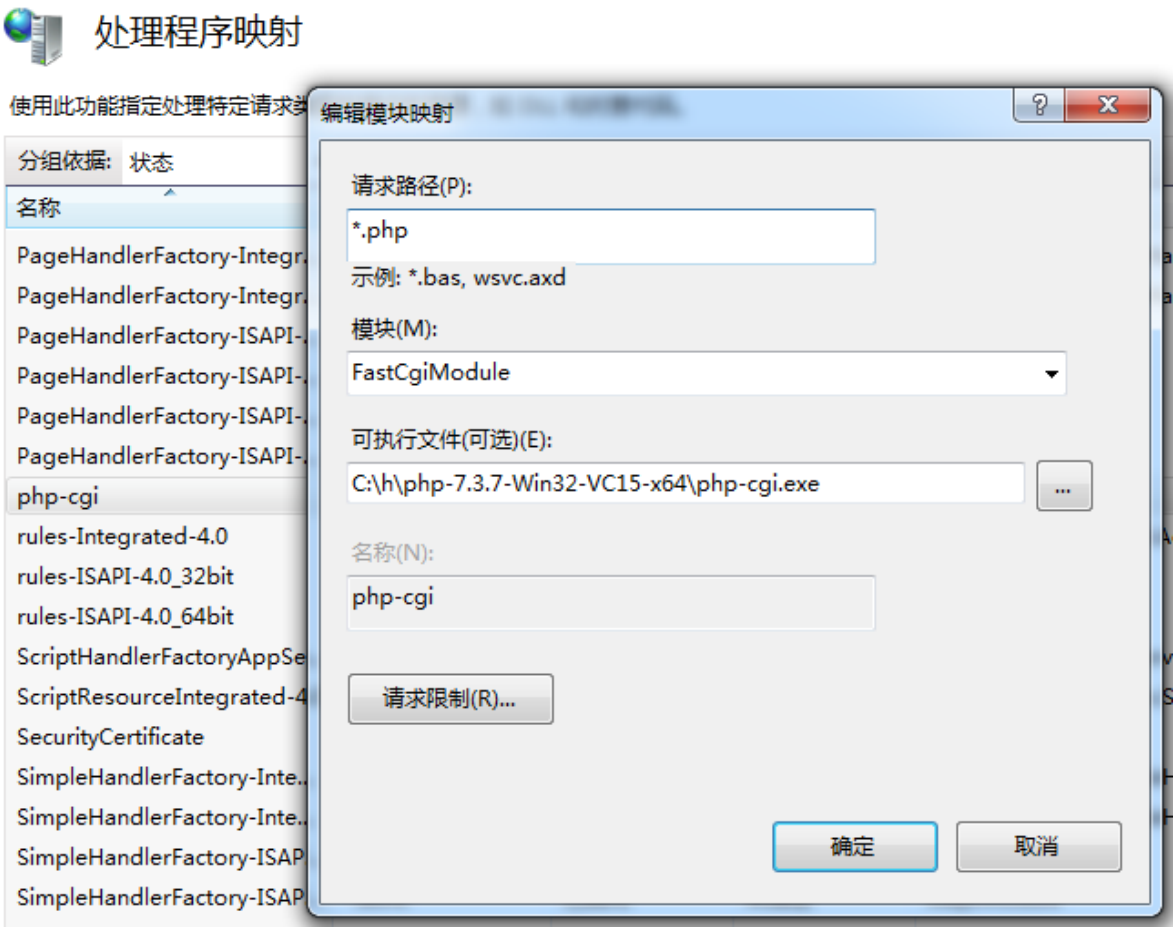
屏幕剪辑的捕获时间: 2019/7/29 21:05

2.3 添加网站



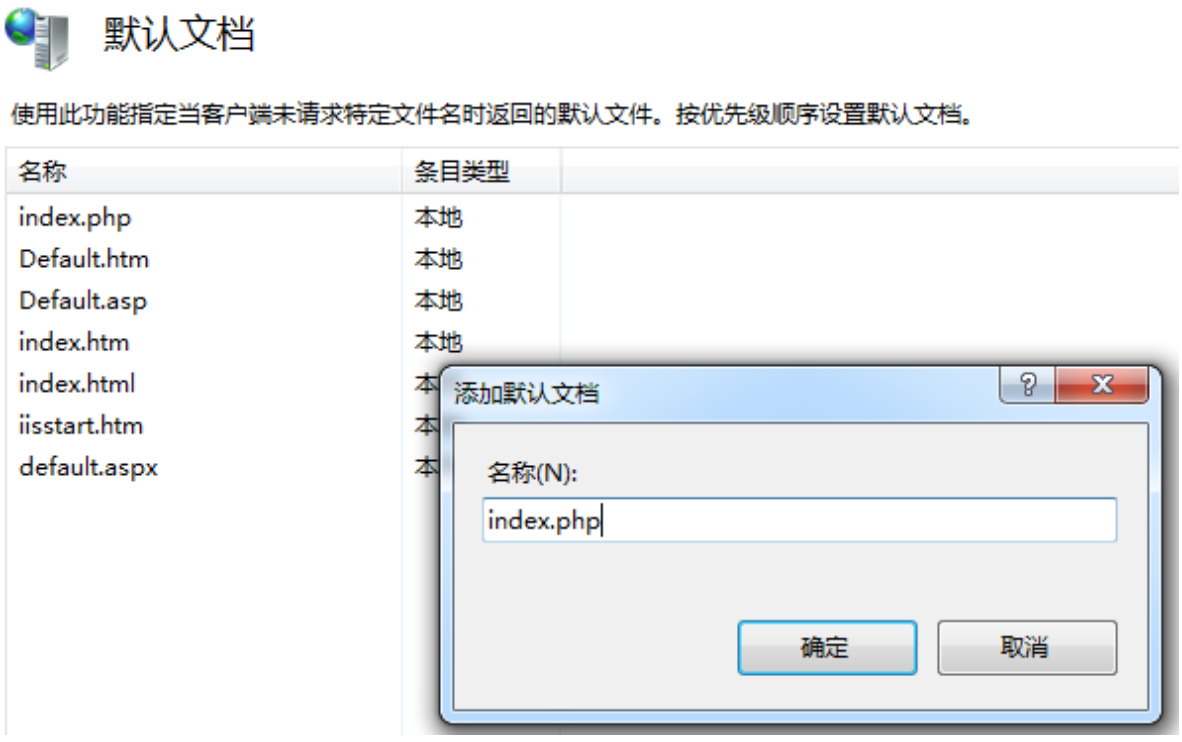
屏幕剪辑的捕获时间: 2019/7/29 21:10

2.4 添加模块映射



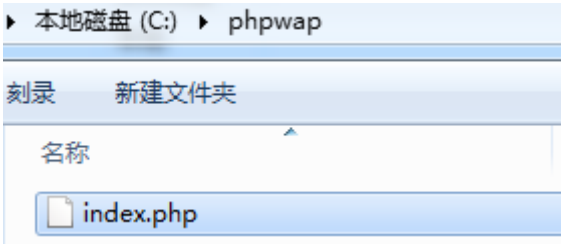
屏幕剪辑的捕获时间: 2019/7/29 21:06

2.5 添加默认文档



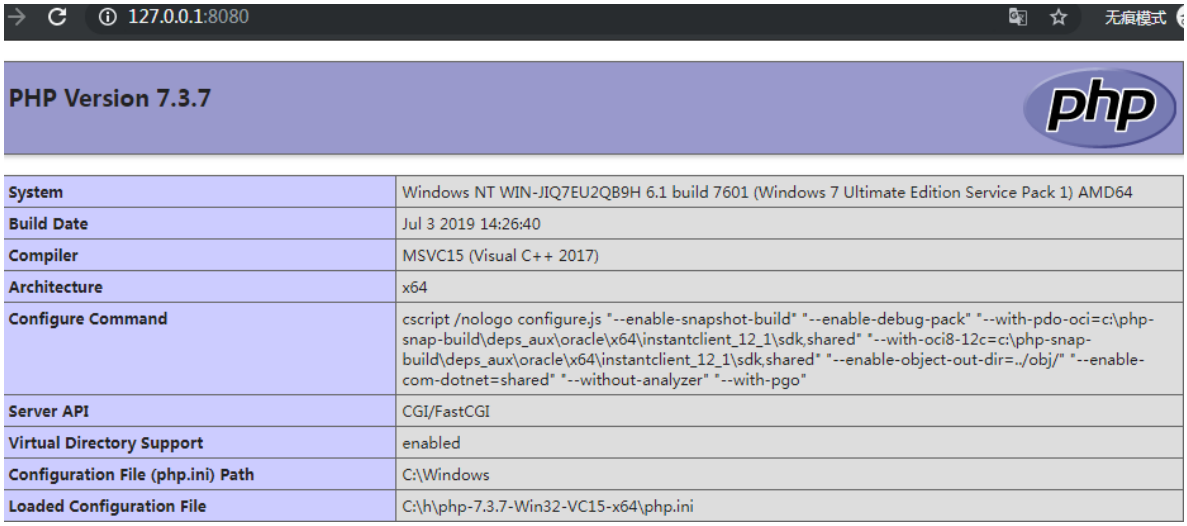
屏幕剪辑的捕获时间: 2019/7/29 21:12

在网站的物理路径下新建index.php



屏幕剪辑的捕获时间: 2019/7/29 21:13

访问网站http://127.0.0.1:8080，显示如下界面表示PHP环境设置完成：



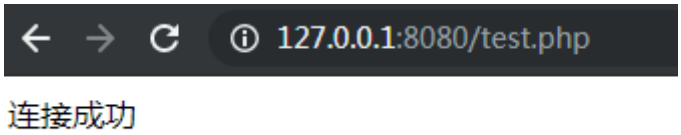
System	Windows NT WIN-JIQ7EU2QB9H 6.1 build 7601 (Windows 7 Ultimate Edition Service Pack 1) AMD64
Build Date	Jul 3 2019 14:26:40
Compiler	MSVC15 (Visual C++ 2017)
Architecture	x64
Configure Command	cscript /nologo configure.js "--enable-snapshot-build" "--enable-debug-pack" "--with-pdo-oci=c:\php-snap-build\deps_aux\oracle\x64\instantclient_12_1\sdk,shared" "--with-oci8-12c=c:\php-snap-build\deps_aux\oracle\x64\instantclient_12_1\sdk,shared" "--enable-object-out-dir=../obj/" "--enable-com-dotnet=shared" "--without-analyzer" "--with-pgo"
Server API	CGI/FastCGI
Virtual Directory Support	enabled
Configuration File (php.ini) Path	C:\Windows
Loaded Configuration File	C:\h\php-7.3.7-Win32-VC15-x64\php.ini

屏幕剪辑的捕获时间: 2019/7/29 21:02

3. 安装MySQL

需要安装依赖：NDP452-KB2901907-x86-x64-AllOS-ENU、vc_redist.x64

配置好root密码以及新添加用户和密码后，php脚本连接MySQL测试：



连接成功

屏幕剪辑的捕获时间: 2019/7/29 21:45

测试连接成功。

参考链接

1. https://blog.csdn.net/rockage/article/details/79441346
2. https://github.com/qiwsir/ITArticles/blob/master/Linux/CentOS-nginx%20%2B%20mysql%20%2B%20php-fpm.md
3. https://juejin.im/post/58db7d742f301e007e9a00a7

4. https://linuxeye.com/351.html
5. https://www.runoob.com/php/php-mysql-connect.html
6. https://blog.csdn.net/JoeChao1003/article/details/60633271
7. https://www.cnblogs.com/cndavidwang/p/9357684.html
8. https://tanghengzhi.com/tag/mysql/
9. https://ymx.iteye.com/blog/411467
10. https://www.4spaces.org/php-fpm-stop-start/
11. https://www.php.net/manual/zh/ini.core.php
12. https://www.lastupdate.net/4572.html
13. https://www.securitypaper.org/2.sdl%E8%A7%84%E8%8C%83%E6%96%87%E6%A1%A3/9-mysql%E5%AE%89%E5%85%A8%E9%85%8D%E7%BD%AE/
14. https://blog.csdn.net/dream_successor/article/details/78615825?locationNum=8&fps=1
15. https://kangzubin.com/blog/lnmp-apache/
16. https://kangzubin.com/blog/linux-apache-mysql-php-nginx/
17. https://www.cnblogs.com/zl0372/articles/php_4.html
18. http://www.onepx.com/from-apache-mod-php-to-php-fpm.html
19. https://weizhimiao.github.io/2016/10/21/apache%E4%B8%8A%E8%BF%90%E8%A1%8CPHP%E7%9A%84%E5%87%A0%E7%A7%8D%E6%96%B9%E5%BC%8F%E6%B1%87%E6%80%BB/
20. https://weizhimiao.github.io/2016/10/20/High-performance%20PHP%20on%20apache%20httpd%202.4.x%20using%20mod_proxy_fcgi%20and%20php-fpm/
21. https://cnzhx.net/blog/apache-httpd-mod_proxy_fcgi-php-fpm/
22. https://wenda.shukaiming.com/article/194
23. https://wiki.archlinux.org/index.php/Apache_HTTP_Server_(%E7%AE%80%E4%BD%93%E4%B8%AD%E6%96%87)
24. https://blog.csdn.net/reblue520/article/details/50405148
25. https://blog.csdn.net/yuanguozhengjust/article/details/25747729
26. https://www.jianshu.com/p/917c022bca1b
27. http://www.caizhichao.cn/465.html