**CentOS系统环境搭建**

# 1 VMware安装centos7

安装虚拟机时版本要选centos 64位

修改网络和主机



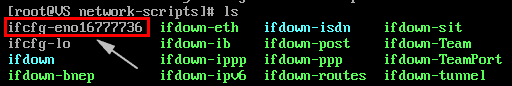


# 2 配置网络

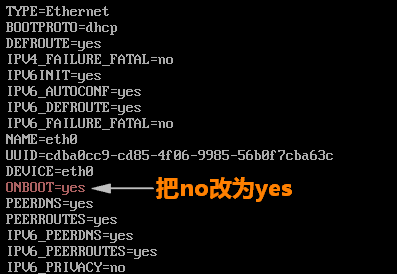
CentOS7安装完成后，发现无法上网，需要做简单的配置

**(1) 将onboot的"no"改为“yes”**

**cd /etc/sysconfig/network-scripts/**



**vim ifcfg-eno16777736**



如果安装过程中有修改主机名，把主机名添加到文件/etc/hosts

获得主机名：hostname

**vim /etc/hosts**

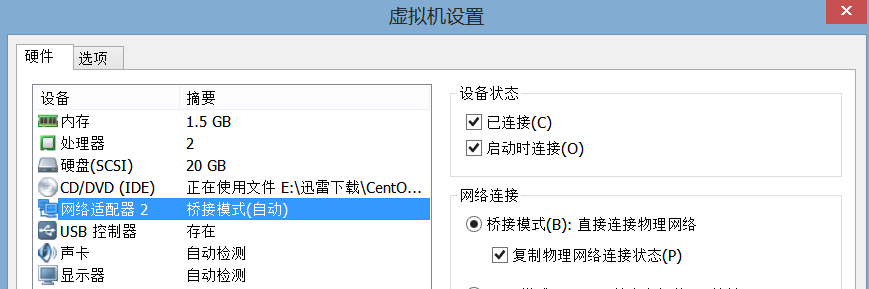
添加内容

127.0.0.1 主机名

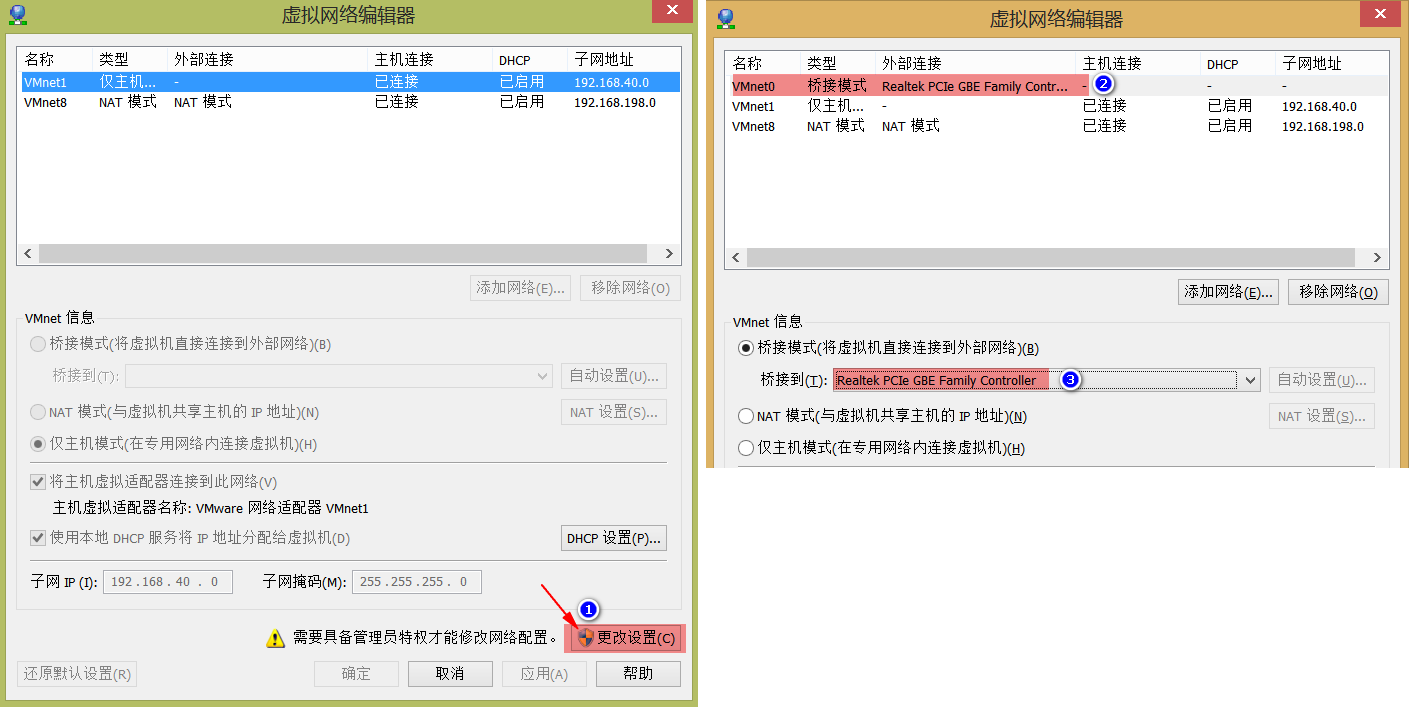
**(2) 重启网络service network restart或重启系统**

**(3) 也可以设置为静态IP网络**

设置VMware的网络为桥接状态







**vim /etc/sysconfig/network-scripts/ifcfg-eno16777736**

**设置静态ip**

|  |
| --- |
| TYPE=Ethernet  BOOTPROTO=static  DEFROUTE=yes  PEERDNS=yes  PEERROUTES=yes  IPV4\_FAILURE\_FATAL=no  IPV6INIT=yes  IPV6\_AUTOCONF=yes  IPV6\_DEFROUTE=yes  IPV6\_PEERDNS=yes  IPV6\_PEERROUTES=yes  IPV6\_FAILURE\_FATAL=no  NAME=eno16777736  UUID=34bbe4fa-f0b9-4ced-828a-f7f7e1094e4a  DEVICE=eno16777736  ONBOOT=yes  IPADDR=192.168.8.202  NETMASK=255.255.255.0  GATEWAY=192.168.8.1  DNS1=202.96.128.86  DNS2=114.114.114.114 |

其中DNS1、DNS2为域名解析地址，设置之后自动复制到/etc/resolv.conf

**可以设置网桥**

|  |
| --- |
| TYPE=Bridge  DEVICE=br0  BOOTPROTO=static  ONBOOT=yes  IPADDR=192.168.6.110  NETMASK=255.255.255.0  GATEWAY=192.168.6.1  DNS1=202.96.128.86  DNS2=114.114.114.114 |

重启网络**service network restar**t或系统

查看网络地址命令：**ip addr**

centos7取消了ifconfig命令

# 3 安装vim

**yum install vim**

配置.vimrc

**vim ~/.vimrc**

添加内容：

|  |
| --- |
| set nu  syntax on  set tabstop=4  set shiftwidth=4  set expandtab  set guioptions-=T  set nobackup  set incsearch  set encoding=utf-8  set fileencodings=utf-8  #set mouse=a#会屏蔽鼠标右键  set ruler  set ff=unix  set t\_Co=256 |

# 4 安装系统相关库文件

**yum install -y gcc gcc-c++ autoconf automake zlib\* libxml\* ncurses-devel libmcrypt\* libtool\* cmake**

# 5 修改终端颜色为xtem-256

查看终端支持的颜色：**tput colors**

查看终端类型： **echo $TERM**

修改.bashrc文件

**vim ~/.bashrc**文件添加

|  |
| --- |
| if [ "$TERM" == "xterm" ]; then  export TERM=xterm-256color  fi  或者  if [ -e /usr/share/terminfo/x/xterm-256color ]; then  #debian在/lib/terminfo/x/xterm-256color  export TERM='xterm-256color'  else  export TERM='xterm-color'  fi |

使.bashrc 生效：**source ~/.bashrc**

再次查看终端支持的颜色：**tput colors**

再次查看终端类型： **echo $TERM**

# 6 安装samba

**(1) 安装samba**

**yum install -y samba samba-client samba-swat**

**(2) 编辑配置文件**

**cp /etc/samba/smb.conf /etc/samba/smb.conf\_backup**

**vim /etc/samba/smb.conf**

在[global]添加下面内容：

|  |
| --- |
| load printers = no  printing = bsd  printcap name = /dev/null  disable spoolss = yes |

注销在字段#**[homes]**，为了自定义共享

|  |
| --- |
| #[homes]  #browseable = yes  #writable = yes |

在文件末尾添加创建共享文件夹字段

|  |
| --- |
| [myShare]  path = /home/vison  public = no  writable = yes  browseable = yes  create mask = 0774  directory mask = 0774  guest ok = no |

**(3) 添加smb用户，并设置linux用户密码**

添加linux用户

**sudo adduser -a user** *# user为用户名*

　　把新添加的Linux用户添加到smb用户，并设置smb密码

**sudo smbpasswd -a user** *# 已存在的linux用户*

**(4) 禁止selinux**

vi /etc/selinux/config

把SELINUX字段设置为disabled

**setenforce 0**

**(5) 启动samba服务**

**systemctl enable smb.service** // 设置开机启动

**systemctl enable nmb.service**  // 设置开机启动

**systemctl restart smb.service**

**systemctl restart nmb.service**

**systemctl status smb** // 查看smb的状态

**(6) 添加主机名到/etc/hosts，解决连接慢的方法**

获得主机名：hostname

**vim /etc/hosts**

把下面内容添加到

127.0.0.1 主机名

**(7) window访问共享文件**

在windown系统下，随便打开一个文件夹，然后输入路径\\192.168.8.101 (centos IP)

**注意：samba最好在单网卡桥接模式下**

# 7 搭建web服务器

安装命令行 Web 浏览器

**yum install links**

## 安装mysql

|  |  |
| --- | --- |
| 安装前创建用户和数据文件存储文件夹 | mkdir -p /data/mysql  groupadd mysql  useradd -g mysql mysql  chown mysql.mysql -R /data/mysql |
| 下载mysql源码 | wget http://mirrors.sohu.com/mysql/MySQL-5.5/mysql-5.5.35.tar.gz |
| 解压 | tar -zxvf mysql-5.5.35.tar.gz |
| 进入目录 | cd mysql-5.5.35 |
| 检查编译环境 | cmake -DCMAKE\_INSTALL\_PREFIX=/usr/local/mysql \  -DMYSQL\_UNIX\_ADDR=/tmp/mysqld.sock \  -DDEFAULT\_CHARSET=utf8 \  -DDEFAULT\_COLLATION=utf8\_general\_ci \  -DWITH\_EXTRA\_CHARSETS:STRING=utf8,gbk \  -DWITH\_MYISAM\_STORAGE\_ENGINE=1 \  -DWITH\_INNOBASE\_STORAGE\_ENGINE=1 \  -DWITH\_MEMORY\_STORAGE\_ENGINE=1 \  -DWITH\_READLINE=1 \  -DENABLED\_LOCAL\_INFILE=1 \  -DMYSQL\_DATADIR=/data/mysql \  -DMYSQL\_USER=mysql \  -DMYSQL\_TCP\_PORT=3306 |
| 编译安装 | make && make install |
| 复制配置文件 | cp support-files/my-medium.cnf /etc/my.cnf |
| 赋给scripts/mysql\_install\_db文件执行权限 | chmod 755 scripts/mysql\_install\_db  scripts/mysql\_install\_db --user=mysql --basedir=/usr/local/mysql/ --datadir= /data/mysql/  注：basedir：mysql安装路径 datadir：数据库文件储存路径 |
| 设置mysqld的开机启动 | cp support-files/mysql.server /etc/init.d/mysql  chmod 755 /etc/init.d/mysql  chkconfig mysql on |
| 为MySQL配置环境变量 | export PATH=/usr/local/mysql/bin:$PATH  alias mysql\_start="mysqld\_safe &"  alias mysql\_stop="mysqladmin –u root -p shutdown"  注：将mysql的bin目录加到PATH中，有利于以后管理和维护，在/etc/profile中加入myslq/bin，同时增加两个别名方便操作。 |

启动mysql服务：**/etc/init.d/mysql start**

查看是否启动成功：**ps -ef | grep mysqld**

登陆mysql：**mysql -uroot -p**

## 安装web服务软件

### 7.2.1 安装Apache

|  |  |
| --- | --- |
| 下载apache依赖apr源码 | wget <http://archive.apache.org/dist/apr/apr-1.5.2.tar.gz> |
| 解压 | tar -zxvf apr-1.5.2.tar.gz |
| 进入目录 | cd apr-1.5.2 |
| 检查编译环境 | ./configure --prefix=/usr/local/apr |
| 编译安装apr | make && make install |
| 下载apache依赖apr-util源码 | wget <http://archive.apache.org/dist/apr/apr-util-1.5.4.tar.gz> |
| 解压 | tar -zxvf apr-util-1.5.4.tar.gz |
| 进入目录 | cd apr-util-1.5.4 |
| 检查配置环境 | ./configure --prefix=/usr/local/apr-util --with-apr=/usr/local/apr |
| 编译安装apr-util | make && make install |
| apache依赖pcre | yum install pcre |
| 下载apache源码 | wget <http://mirrors.cnnic.cn/apache/httpd/httpd-2.4.23.tar.gz> |
| 解压 | tar -zxvf httpd-2.4.23.tar.gz |
| 进入目录 | cd httpd-2.4.23 |
| 检查配置环境 | ./configure --prefix=/usr/local/apache \  --sysconfdir=/usr/local/apache/httpd \  --with-apr=/usr/local/apr \  --with-apr-util=/usr/local/apr-util \  --enable-so \  --enable-ssl \  --enable-cgi \  --enable-rewrite \  --with-zlib \  --with-pcre \  --enable-modules=all \  --enable-mpms-shared=all \  --with-mpm=event  注：httpd配置参数说明   |  |  | | --- | --- | | --prefix= | 安装位置； | | --sysconfdir= | 配置文件位置； | | --with-apr= | 指定依赖apr程序安装位置； | | --with-apr-util= | 指定依赖apr-util程序安装位置； | | --enable-so | 支持DSO动态装载模块； | | --enable-ssl | 支持SSL/TLS，可实现https协议访问，需要安装openssl-devel； | | --enable-cgi | 支持CGI脚本； | | --enable-rewrite | 支持URL重写； | | --with-zlib | 使用指定的zlib压缩库，不指定路径会自动寻找； | | --with-pcre | 使用指定的pcre库，增强的正则表达式分析工具；不指定路径会自动寻找 需已安装pcre-devel； | |
| 编译安装 | make && make install |

**(1) 启动apache**

**/usr/local/apache/bin/apachectl start**

如果出现下面警告

AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using 127.0.0.1. Set the 'ServerName' directive globally to suppress this message

在httpd.conf的全局域添加：ServerName 127.0.0.1:80

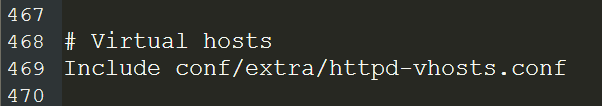
测试连接：**links 127.0.0.1**

如果出现It works!说明apache安装完成。

**(2) 配置apache，使其支持虚拟主机**

**vim /usr/local/apache/httpd/httpd.conf**

找到Virtual hosts字段，去掉Include conf/extra/httpd-vhosts.conf前面的#号，就能支持虚拟主机的配置了，如下图所示：



**(3) 配置虚拟主机信息**

vim /usr/local/apache/httpd/extra/httpd-vhosts.conf

配置信息如下：

|  |
| --- |
| <VirtualHost \*:80>  ServerAdmin webmaster@dummy-host.example.com  DocumentRoot "/home/vison/www"  ServerName 127.0.0.1  ServerAlias www.vison.com  ErrorLog "logs/vison\_error\_log"  CustomLog "logs/vison\_access\_log" common  </VirtualHost> |

其中字段ServerName是域名，字段DocumentRoot是访问的根目录。当浏览器输入域名时，Apache返回DocumentRoot目录下的index.html文件给浏览器，注意：DocumentRoot目录要真实存在，并且目录有index.html文件，如果不存在会报错403。

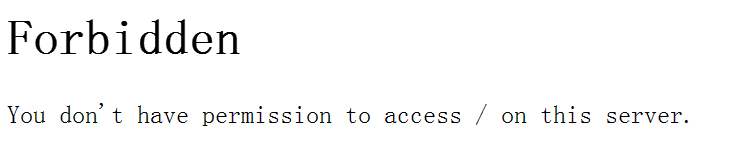
如果想设置多个虚拟主机，就重复复制上面代码，修改字段DocumentRoot和ServerName即可，例如下图设置了两个虚拟主机。



设置完毕重启Apache。

**(4) 验证设置虚拟主机是否成功**

在浏览器上输入域名，如果出现403 Forbidden页面，如下图所示：



打开Apache配置文件，找到字段<Directory />，把deny from all修改为allow from all，

**vim /usr/local/apache/httpd/httpd.conf**

修改结果如下：

|  |
| --- |
| <Directory />  AllowOverride none  # Require all denied  allow from all  </Directory> |

最后重启Apache

### 7.2.2 安装Nginx

**方式一：离线安装nginx**

|  |  |
| --- | --- |
| 安装依赖包 | yum install pcre pcre-devel |
| 下载一致性hash模块 | wget https://github.com/replay/ngx\_http\_consistent\_hash/archive/master.zip |
| 解压 | unzip ngx\_http\_consistent\_hash-master.zip |
| 下载nginx源码 | wget http://nginx.org/download/nginx-1.10.1.tar.gz |
| 解压 | tar -zxvf nginx-1.10.1.tar.gz |
| 进入目录 | cd nginx-1.10.1 |
| 检查编译环境 | ./configure --prefix=/usr/local/nginx --with-http\_stub\_status\_module --add-module=/home/vison/ngx\_http\_consistent\_hash-master |
| 编译安装 | make && make install |

启动Nginx：**/usr/local/nginx/sbin/nginx**

如果有其他服务占用80端口，会导致启动失败，先查看哪个服务占有80端口，然后关闭进程即可

**netstat -antp**

**kill -9 进程id**

验证是否安装成功

links 127.0.0.1

**方式二：联网安装nginx**

|  |  |
| --- | --- |
| 下载对应当前系统版本的nginx包 | wget <http://nginx.org/packages/centos/7/noarch/RPMS/nginx-release-centos-7-0.el7.ngx.noarch.rpm> |
| 建立nginx的yum仓库 | rpm -ivh nginx-release-centos-7-0.el7.ngx.noarch.rpm |
| 安装nginx | yum install nginx |
| 启动nginx服务 | systemctl start nginx |
| 验证是否安装成功 | links 127.0.0.1 |

## 安装php

注意：apache一般是把php当做自己的一个模块来启动的，而nginx则是把http请求变量(如get、user\_agent等)转发给 php进程，即php独立进程，与nginx进行通信，称为 fastcgi运行方式。

因此，为apache所编译的php，是不能用于nginx的。

### 7.3.1 安装适合nginx的php

|  |  |
| --- | --- |
| 安装依赖包 | yum -y install libxml2 libjpeg libjpeg-devel libpng \  libpng-devel freetype freetype-devel libxml2 libxml2-devel \  mysql pcre-devel openssl openssl-devel curl-devel libxslt-devel |
| 下载源码 | wget http://cn2.php.net/distributions/php-7.0.6.tar.gz |
| 解压 | tar -zxvf php-7.0.6.tar.gz |
| 进入目录 | cd php-7.0.6 |
| 检查编译环境 | ./configure --prefix=/usr/local/php \  --enable-fpm \  --enable-mysqlnd \  --with-mysqli \  --with-pdo-mysql \  --with-pdo-sqlite \  --with-gd \  --enable-gd-native-ttf \  --enable-gd-jis-conv  安装详细模块：  ./configure --prefix=/usr/local/php7 \  --enable-fpm \  --enable-mysqlnd \  --with-mysqli \  --with-gd \  --enable-gd-native-ttf \  --enable-gd-jis-conv \  --with-curl \  --with-freetype-dir \  --with-gettext \  --with-iconv-dir \  --with-kerberos \  --with-libdir=lib64 \  --with-libxml-dir \  --with-openssl \  --with-pcre-regex \  --with-pdo-mysql \  --with-pdo-sqlite \  --with-pear \  --with-png-dir \  --with-xmlrpc \  --with-xsl \  --with-zlib \  --enable-bcmath \  --enable-libxml \  --enable-inline-optimization \  --enable-mbregex \  --enable-mbstring \  --enable-opcache \  --enable-pcntl \  --enable-shmop \  --enable-soap \  --enable-sockets \  --enable-sysvsem \  --enable-xml \  --enable-zip |
| 编译安装 | make && make install  注：安装完后在/usr/local/php/sbin目录下有个php-fpm运行文件，这就是php进程管理器，要执行php文件，必须启动php-fpm。 |
| 复制配置文件 | cp php.ini-development /usr/local/php/lib/php.ini  cd /usr/local/php/etc  cp php-fpm.conf.default php-fpm.conf  cd php-fpm.d/  cp www.conf.default [www.conf](http://www.conf)  注：其中www.conf文件为 fmp pool的定义，可以定义多个，通过cat www.conf，看到fpm的配置信息，其中listen为php-fpm监听的地址 |

测试配置是否正确：**/usr/local/php/sbin/php-fpm -t**

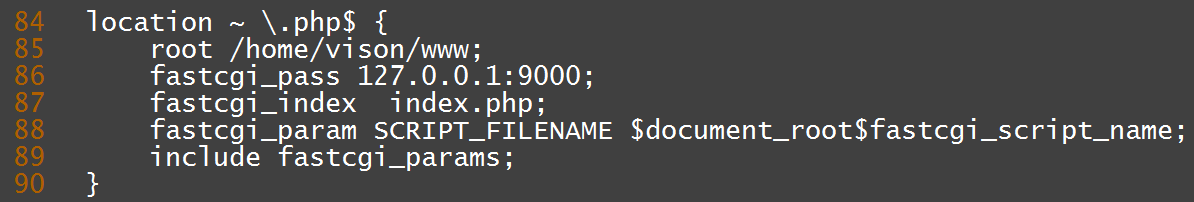
在nginx整合php

**vim /etc/nginx/nginx.conf**

在http{}里添加下面内容

|  |
| --- |
| server{  listen 80;  server\_name 192.168.8.104;  #access\_log logs/log.log;  location ~ \.php$ {  root /home/vison/www;  fastcgi\_pass 127.0.0.1:9000;  fastcgi\_index index.php;  fastcgi\_param SCRIPT\_FILENAME /home/vison/www$fastcgi\_script\_name;  include fastcgi\_params;  }  } |

配置结果如下图：



测试nginx配置信息是否正确：**/usr/sbin/nginx -t**

重新加载配置文件：**/usr/sbin/nginx -s reload**

运行fmp程序：**/usr/local/php/sbin/php-fpm**

测试是否整合成功：**links 127.0.0.1/info.php**

### 7.3.2 安装适合apache的php

|  |  |
| --- | --- |
| 下载php依赖包libxml2 | wget http://xmlsoft.org/sources/old/libxml2-2.6.32.tar.gz |
| 解压 | tar -zxvf libxml2-2.6.32.tar.gz |
| 进入目录 | cd libxml2-2.6.32 |
| 检查配置环境 | ./configure --prefix=/usr/local/libxml2 |
| 编译安装libxml2 | make && make install |
| 下载php源码 | wget http://cn2.php.net/get/php-5.5.38.tar.gz/from/this/mirror |
| 解压 | tar -zxvf php-5.5.38.tar.gz |
| 进入目录 | cd php-5.5.38 |
| 检查配置环境 | ./configure --prefix=/usr/local/php \  --with-mysql=mysqlnd \  --enable-mysqlnd \  --with-mysqli \  --with-pdo-mysql \  --with-gd \  --enable-gd-native-ttf \  --enable-gd-jis-conv \  --with-libxml-dir=/usr/local/libxml2 \  --with-apxs2=/usr/local/apache/bin/apxs |
| 编译安装php | make && make install |
| 复制配置文件 | cp php.ini-development /usr/local/php/lib/php.ini |

测试配置是否正确：**/usr/local/php/bin/php -t**

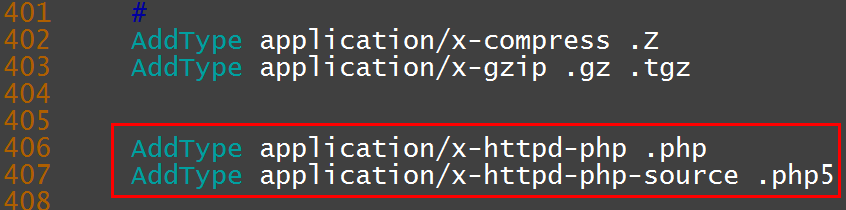
在apache整合php

**vim /usr/local/apache/httpd/httpd.conf**

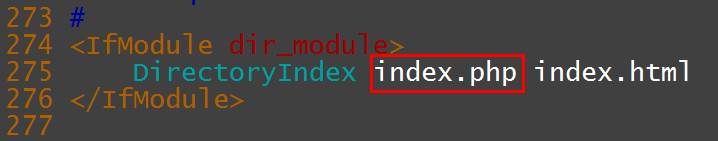
找到字段<IfModule mime\_module>，添加下面内容

|  |
| --- |
| AddType application/x-httpd-php .php  AddType application/x-httpd-php-source .php5 |

如下图：



找到字段<IfModule dir\_module>，在下面添加索引index.php，如下图：



最后重启apache

测试是否整合成功： **links 127.0.0.1/info.php**

### 7.3.3 测试网页和mysql连接

单独开启nginx或apache测试，如果同时开启，需要把其中一个80端口改为其他端口号。

(1) 测试phpinfo()函数

在网站根目录下新建index.php文件

**vim /home/vison/www/index.php**

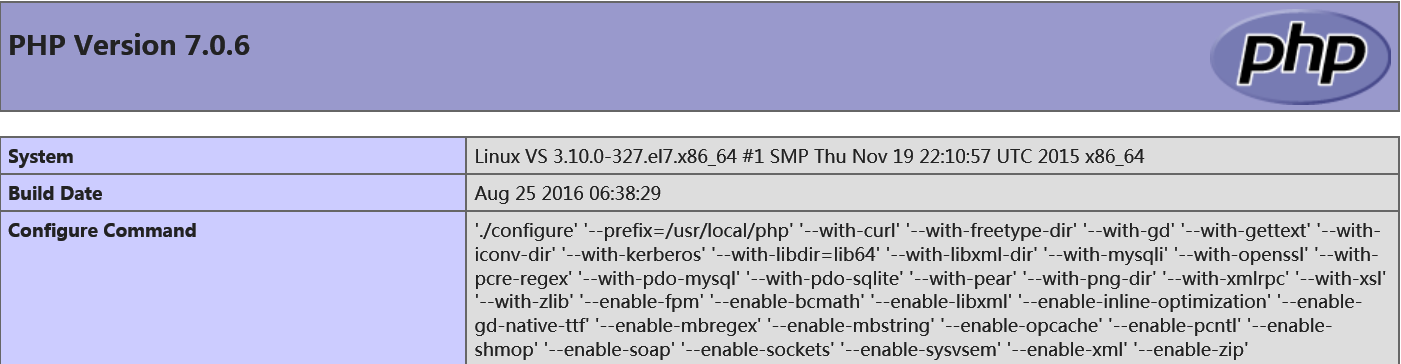
输入内容：<?php echo phpinfo(); ?>

在防火墙开通 80 端口，

**firewall-cmd --permanent --add-port=80/tcp**

**firewall-cmd --reload**

在浏览器中输入<http://192.168.8.104>，页面结果如下：



(2)测试mysql连接

在网站根目录下新建mysql.php文件

**vim /home/vison/www/mysql.php**

输入内容：

用升级版的mysqli函数

|  |
| --- |
| <?php  $conn = new mysqli('localhost', 'root', '','test');  var\_dump($conn); |

或用旧的mysql\_connect函数

|  |
| --- |
| <?php  $conn = mysql\_connect('localhost', 'root', '');  var\_dump($conn); |

如果出现下面错误

**Warning**: mysqli::\_\_construct(): (HY000/2002): No such file or directory in **/home/vison/www/conn.php** on line **2**

错误原因：使用主机名为localhost时是走socket连接的，使用主机名为ip的是走TCP连接的，

两种解决方法：

①用IP代替localhost

$conn = new mysqli('127.0.0.1', 'root', '123456','test')

②设置php.ini文件字段mysqli.default\_socket=路径/mysql.sock

## 7.4 安装Memcached缓存

Memcached 是一个高性能的分布式内存对象缓存系统，用于动态Web应用以减轻数据库负载。它通过在内存中缓存数据和对象来减少读取数据库的次数，从而提高动态、数据库驱动网站的速度。Memcached基于一个存储键/值对的hashmap。其守护进程（daemon ）是用C写的，但是客户端可以用任何语言来编写，并通过memcached协议与守护进程通信

|  |  |
| --- | --- |
| 下载源码依赖包libevent | wget <https://github.com/libevent/libevent/releases/download/release-2.0.22-stable/libevent-2.0.22-stable.tar.gz> |
| 解压 | tar -zxvf libevent-2.0.22-stable.tar.gz |
| 进入目录 | cd libevent-2.0.22-stable |
| 检查编译环境 | ./configure --prefix=/usr/local/libevent |
| 编译和安装libevent | make && make install |
| 下载Memcached源码 | wget <http://www.memcached.org/files/memcached-1.4.31.tar.gz> |
| 解压 | tar -zxvf memcached-1.4.31.tar.gz |
| 进入目录 | cd memcached-1.4.31 |
| 检查编译环境 | ./configure --prefix=/usr/local/memcached --with-libevent=/usr/local/libevent |
| 编译和安装Memcached | make && make install |

memcached常用启动参数

注意：处于安全考虑，linux下不能用root用户启动memcached

几个重要参数的使用：

|  |  |
| --- | --- |
| -p <num> | 监听端口，默认端口11211 |
| -u <username> | 用户，在linux下时使用 |
| -m <num> | 分配给memcached的最大缓存，默认64M |
| -c <num> | 最大连接数，默认是1024 |
| -f <factor> | 增长因子 |
| -vv | 打印信息 |
| -d | 选项是启动一个守护进程 |

(1) 启动memcached

以守护进程启动：**memcached -m 128 -u nobody -d**

以打印信息方式启动：**memcached -m 128 -u nobody -vv**

查看memcached进程：**ps -A | grep memcached**

关闭memcached进程：**kill -9 memcached的进程id**

(2) telnet连接memcached

memcached的客户端和服务器端的通信使用的是文本协议，所以可以用telnet连接memcached来做交互。

在终端连接：**telnet localhost 11211**，然后按下ctrl+]组合键打开回显功能，再按下确认键即可。

也可以使用secureCRT连接。

## 7.6 安装php-memcached扩展

注意：php的缓存扩展有两种：php-memcache和php-memcached

名字只是多一个字母d

区别：

(1) memcache是完全在PHP框架内开发的，基于pecl扩展库libmemcache。而memecached扩展是基于libmemcached库开发的，所以支持cas操作，cas操作就是多个线程并发修改同一个key的值的时候能够进行处理。

(2) memcached可以使用函数setOption()统一设置选项，memcache通过php.ini设置。

(3) memcache支持面向对象和面向过程两套接口操作服务端。而memcached扩展只支持面向对象方式操作。这也是库(libmemcached与libmemcache)的不同影响。

(4) memcached支持Binary Protocol，而memcache不支持。理论上二进制解码速度快于文本，但实际优势微乎其微。且目前大部分用户还是使用文本协议，比较简单稳定。实践中使用二进制的还是比较少。

### 7.6.1 安装php-memcached

|  |  |
| --- | --- |
| 安装依赖包 | yum install libmemcached libmemcached-devel |
| 下载php-memcached源码 | wget <https://pecl.php.net/get/memcached-2.2.0.tgz>  适合php7下载：wget <https://github.com/php-memcached-dev/php-memcached/tree/php7> |
| 解压 | tar -zxvf memcached-2.2.0.tgz |
| 进入目录 | cd memcached-2.2.0 |
| 生成配置文件 | /usr/local/php/bin/phpize --with=/usr/local/php/bin/php-config |
| 检测编译环境 | ./configure --with-php-config=/usr/local/php/bin/php-config |
| 编译安装php-memcached | make && make install |

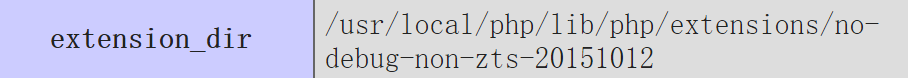
### 7.6.2 安装php-memcache

|  |  |
| --- | --- |
| 下载php-memcache源码 | wget <http://pecl.php.net/get/memcache-2.2.4.tgz> |
| 解压 | tar -zxvf memcached-2.2.4.tgz |
| 进入目录 | cd memcached-2.2.4 |
| 生成配置文件 | /usr/local/php/bin/phpize --with=/usr/local/php/bin/php-config |
| 检测编译环境 | ./configure --with-php-config=/usr/local/php/bin/php-config |
| 编译安装php-memcache | make && make install |

### 7.6.3 配置php.ini

通过phpinfo()函数输出，在网页上看到的配置文件路径和扩展包的路径





打开配置文件：vim /usr/local/php/lib/php.ini

(如果配置文件不存在则从php安装包中复制复制一份配置文件)

查找.so，在附近添加下面内容：

|  |
| --- |
| extension=/usr/local/php/lib/php/extensions/no-debug-zts-20121212/memcached.so  extension=/usr/local/php/lib/php/extensions/no-debug-zts-20121212/memcache.so |

检验安装扩展是否成功

重启apache或nginx

通过phpinfo()函数输出，查看网页上有没有memcached和memcache模块。

## 7.7 开启和关闭服务

### 7.7.1 服务器nginx+php

开启服务：

**systemctl start nginx**

**/usr/local/php/sbin/php-fpm**

关闭服务：

**systemctl stop nginx**

**pkill -9 php-fpm**

### 7.7.2 服务器apche +php

开启服务：

**/usr/local/apache/bin/apachectl start**

关闭服务：

**pkill -9 httpd**

### 7.7.3 服务器nginx+apche +php

开启服务：

**systemctl start nginx**

**/usr/local/apache/bin/apachectl start**

关闭服务：

**systemctl stop nginx**

**pkill -9 httpd**

# 安装软件是configure一直循环问题

原因：虚拟机系统日期时间和实际日期时间有相差，在时间差内当前软件最新版本还么发布，有可能一直循环检测编译环境，

解决放法：校准系统日期时间

**date -s '2016-06-06 06:06:06'**

**clock -w**