阿里云服务器centos5.4 lnmp环境搭建

原创

不想离开水的鱼 于 2014-03-28 10:22:19 发布 980 收藏
分类专栏： 服务器 文章标签： 阿里云服务器 配置环境

本文链接：https://blog.csdn.net/ivyandrich/article/details/22379641


服务器 专栏收录该内容
10 篇文章 0 订阅
订阅专栏

阿里云服务器一台，配置：1u（Xeon 2.26GHz），1G内存，30g硬盘，2m带宽，centos5.4 32位 （ps：1350每年的价格比较诱惑）

默 认如果是linux服务器，进入管理界面用“df -h”命令查看硬盘，会发现只有一个20g的盘，其实是数据盘（在我的环境中是30g的盘未被挂载）没 有挂载，可以参考阿里云的官方手册进行挂载：http://help.aliyun.com/knowledge_detail/5974154.html?spm=5176.7618386.5.3.Jy0YGX

想搭建一个生产环境：nginx + mysql + php 的

废话不多说，步骤如下（从网上查的步骤，有些有问题的地方自己解决）：

1.yum安装编译工具：

yum -y install gcc gcc-c++ libjpeg-devel libpng-devellibtiff-devel fontconfig-devel freetype-devel libXpm-develgettext-devel openssl-devel libtool-ltdl-devel

但是实际上云服务器上居然连make命令都木有，所以还得安装一下，好在有yum：

```
yum -y install gcc automake autoconf libtool make
```

2.安装pcre(用于nginx的rewrite模块)

tar xjvf pcre-8.20.tar.bz2

```
cd pcre-8.20
```

./configure
make

make install

3.安装nginx

tar xzf nginx-1.0.0.tar.gz

cd nginx-1.0.0

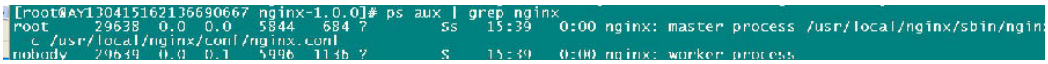
./configure --prefix=/usr/local/nginx/ --with-http_stub_status_module --with-http_ssl_module --with-sha1=/usr/lib --with-pcre=/root/soft/pcre-8.20/

make

make install

安装完成后可以用“/usr/local/nginx/sbin/nginx -c /usr/local/nginx/conf/nginx.conf”启动服务

可以用‘ps aux | grep nginx’查看服务是否存在。



测试完成后，用kill -quit （master process进程号）

4.安装mysql

tar xzf mysql-5.1.26-rc.tar.gz

cd mysql-5.1.26-rc

autoreconf --force --install

./configure --prefix=/usr/local/mysql --without-debug --with-unix-socket-path=/usr/local/mysql/mysql.sock --with-client-ldflags=-all-static --with-mysqld-ldflags=-all-static --enable-assembler --with-extra-charsets=gbk,gb2312,utf8 --with-pthread --enable-thread-safe-client --localstatedir=/usr/local/mysql/var

config报错：

checking for termcap functions library... configure: error: No curses/termcap library found

查了一下，发现缺少ncurses包，没办法，装一个吧，好在有yum：

yum -y install ncurses-devel

还好，很顺利，继续重新执行上边的configure

……

成功，继续

make

make install

……

添加mysql用户及用户组：

```
groupadd mysql
```

```
useradd -r -g mysql mysql
```

权限处理

cd /usr/local/mysql;

chown -R root:mysql .;

初始化系统表（创建系统数据库的表）

./bin/mysql_install_db --user=mysql;

这时会发现/usr/local/mysql下多了一个var目录（因为编译时未指定数据目录，默认会在安装目录下创建var目录用于数据目录）

chown -R mysql:mysql ./var;

chmod -R 777 ./var;

设置环境变量：（话说其实我不太明白这部分是做啥）

```
vi /root/.bash_profile;
```

把PATH=$PATH:$HOME/bin修改为：

PATH=$PATH:$HOME/bin:/usr/local/mysql/bin:/usr/local/mysql/lib

完后保存退出

source /root/.bash_profile

配置启动项：

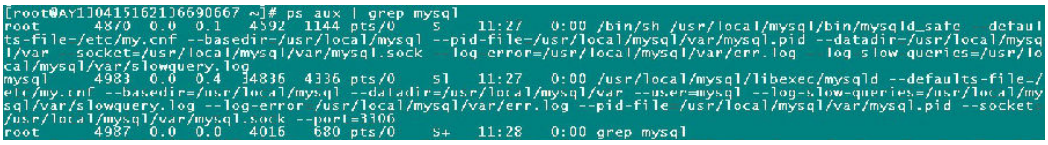
/usr/local/mysql/bin/mysqld_safe --defaults-file=/etc/my.cnf --basedir=/usr/local/mysql --pid-file=/usr/local/mysql/var/mysql.pid --datadir=/usr/local/mysql/var --socket=/usr/local/mysql/var/mysql.sock --log-error=/usr/local/mysql/var/err.log --log-slow-queries=/usr/local/mysql/var/slowquery.log &

不 得不多说两句的是，因为在编辑阶段很2的把mysql.sock放到了mysql的根目录下，完后发现mysql.sock所在文件夹要很高的写权限，为 安全起见，不能放mysql的根目录，所以启动时候放到了数据目录（/usr/local/mysql/var/）,因为反正这个目录也是最高写权限，但 是这样做带来的后果是，用/usr/local/mysql/bin/mysqld进入mysql命令行，或者用mysqlamdin等一些管理工具的时 候，甚至写程序时候都需要手工指定sork文件位置，如果你是按照我这个方法做到这儿的话，请不要怪我，要不你就再回去重新编译一把（偷笑中），不过其实 也没麻烦多少。

完后检查下是否启动成功

ps | aux | grep mysql

……噔噔 噔噔 ，成功啦



修改root用户密码：

/usr/local/mysql/bin/mysqladmin --socket=/usr/local/mysql/var/mysql.sock -uroot -p shutdown

这就是我说的得手工修改sock文件位置

```
mysql>use mysql;
```

mysql>desc user;

```
mysql>update user set Password = password(&#39xxxxxx') where User=&#39root'
```

mysql>exit

5.安装php

编译安装gd2(图片处理模块，可视需要添加)

wget http://down1.chinaunix.net/distfiles/gd-2.0.33.tar.gz

tar xzf gd-2.0.33.tar.gz;

cd gd-2.0.33;

./configure --prefix=/usr/local/gd2;
make;

make install;

编译安装 libiconv（编码转换模块）
wget http://down1.chinaunix.net/distfiles/libiconv-1.13.1.tar.gz;

tar xzf libiconv-1.13.1.tar.gz;

cd libiconv-1.13.1;

./configure --prefix=/usr/local/iconv;

make

make install

编译安装libmcrypt（提供加密算法，如果要php应用mcrypt库函数，需要安装此模块）

```
wget http://www.51osos.com/uploads/soft/libmcrypt-2.5.8.tar.gz;
tar xzf libmcrypt-2.5.8.tar.gz;
cd libmcrypt-2.5.8;
./configure;
make
make install
```

编译安装 mhash(是一个哈希演函数库，如果要php应用mcrypt库函数，需要安装此模块)

```
tar xzf mhash-0.9.9.9.tar.gz;
cd mhash-0.9.9.9;
./configure;
make
make install
```

编译安装spawn-fcgi（包作为FastCGI支持模块）

```
wget http://vbets.googlecode.com/files/spawn-fcgi-1.6.3.tar.gz
tar xzf spawn-fcgi-1.6.3.tar.gz;
cd spawn-fcgi-1.6.3;
./configure;
make;
make install;
```

编译安装libevent（一个网络事件库）

```
tar xzf libevent-2.0.16-stable.tar.gz;
cd libevent-2.0.16-stable;
./configure;
make;
make install;
```

编译安装freetype（字体引擎）

```
tar xzf freetype-2.1.10.tar.gz;
```

```
cd freetype-2.1.10;

./configure --prefix=/usr/local/freetype;

make;

makeinstall
```

编译安装jpeg（对gd库支持的一个包）

```
tar xzf jpegsrc.v6b.tar.gz;

cd jpeg-6b;

mkdir /usr/local/jpeg6;
mkdir /usr/local/jpeg6/bin;
mkdir /usr/local/jpeg6/lib;
mkdir /usr/local/jpeg6/include;
mkdir -p /usr/local/jpeg6/man/man1;

./configure --prefix=/usr/local/jpeg6/ --enable-shared --enable-static;

make;

make install;
```

编译安装libxml（一个用来解析XML文档的函数库）：

```
tar xzf libxml2-2.6.32.tar.gz;

cd libxml2-2.6.32;

./configure --prefix=/usr/local/libxml2;

make;

make install;
```

手工安装成功但是在编辑php的时候出错，还是yum吧，不过yum安装后，php编译时此模块的路径得用系统默认路径/usr/lib

```
yum install libxml
yum install libxml-devel
yum install libxml2
yum install libxml2-devel
```

编译安装curl（模拟http请求）：

```
tar xzf curl-7.15.0.tar.gz;

cd curl-7.15.0;

./configure --prefix=/usr/local/curl;
```

make;

make install;

终于完成了七七八八的准备工作，可能有些木有用到啊。不管了，先装php吧。

编译安装php：

tar xzf php-5.3.6.tar.gz;

cd php-5.3.6;

./configure --prefix=/usr/local/php --with-config-file-path=/usr/local/php/etc --with-mysql=/usr/local/mysql --with-iconv-dir=/usr/local/iconv --with-freetype-dir=/usr/local/freetype --with-jpeg-dir=/usr/local/jpeg6/ --with-zlib-dir=/usr/lib --with-libxml-dir=/usr/lib --with-curl-dir=/usr/local/curl --enable-xml --disable-debug --disable-rpath --enable-discard-path --enable-safe-mode --enable-bcmath --enable-shmop --enable-sysvsem --enable-inline-optimization --with-curlwrappers --enable-mbregex --enable-fastcgi --enable-fpm --enable-force-cgi-redirect --enable-mbstring --with-mcrypt --with-gd --enable-gd-native-ttf --with-openssl;

make

最后出现很多“undefined reference to `libiconv_open”这样的问题

查了一下，发现是libiconv一直出问题

……

表面看，是libiconv安装问题，重装libiconv之后问题依旧，网上看有人舍弃libiconv，使用 –without-iconv，我觉的不可取，这样是回避问题。

找了n久，终于找到bug所在：在执行完 ./configure … 之后，修改下 Makefile，找到其中的

EXTRA_LIBS = -lcrypt -lz -lcrypt -lrt -lmysqlclient -lmcrypt -lldap -llber -lfreetype -lpng -lz -ljpeg -lcurl -lz -lrt -lm -ldl -lnsl -lrt -lxml2 -lz -lm -lssl -lcrypto -ldl -lz -lcurl -ldl -lgssapi_krb5 -lkrb5 -lk5crypto -lcom_err -lidn -lssl -lcrypto -lz -lxml2 -lz -lm -lssl -lcrypto -ldl -lz -lxml2 -lz -lm -lxml2 -lz -lm -lcrypt -lxml2 -lz -lm -lxml2 -lz -lm -lxml2 -lz -lm -lxml2 -lz -lm -lcrypt

在最后加上 **-liconv** 修改成

EXTRA_LIBS = -lcrypt -lz -lcrypt -lrt -lmysqlclient -lmcrypt -lldap -llber -lfreetype -lpng -lz -ljpeg -lcurl -lz -lrt -lm -ldl -lnsl -lrt -lxml2 -lz -lm -lssl -lcrypto -ldl -lz -lcurl -ldl -lgssapi_krb5 -lkrb5 -lk5crypto -lcom_err -lidn -lssl -lcrypto -lz -lxml2 -lz -lm -lssl -lcrypto -ldl -lz -lxml2 -lz -lm -lxml2 -lz -lm -lcrypt -lxml2 -lz -lm -lxml2 -lz -lm -lxml2 -lz -lm -lxml2 -lz -lm -lcrypt -liconv

然后再

make

……终于成功了，

make install（到这儿我都不想记录了，太累了）

总算是成功安装了，之后是一些配置工作（未完，待续）

cp php.ini-development /usr/local/php/etc/php.ini（根据你php安装包目录下具体情况配置）

cp -R ./sapi/fpm/php-fpm.conf /usr/local/php/etc/php-fpm.conf

sed -i &#39s/;pm.min_spare_servers/pm.min_spare_servers/g' /usr/local/php/etc/php-fpm.conf

sed -i &#39s/;pm.max_spare_servers = 35/pm.max_spare_servers = 35/g' /usr/local/php/etc/php-fpm.conf

sed -i &#39s/;pm.max_spare_servers = 35/pm.max_spare_servers = 35/g' /usr/local/php/etc/php-fpm.conf

/usr/local/php/bin/php-fpm

php启动了

安装imagemagic：（切图神器，因为习惯用命令行，所以不安装php扩展了）

```
tar jxvf ImageMagick-6.5.9-10.tar.bz2
cd ImageMagick-6.5.9-10

./configure
make
make install
```

安装php扩展eaccelerator （php加速用）：

tar xzf eaccelerator-eaccelerator-42067ac.tar.gz;

cd eaccelerator-eaccelerator-42067ac;

/usr/local/php/bin/phpize;

注意phpize之后生成的信息需要拷贝下来，在之后配置php.ini时需要用到

```
./configure --enable-eaccelerator=shared --with-phpconfig=/usr/local/php/bin/php-config
make;

make install;
```

建立eaccelerator缓存目录

mkdir /tmp/eaccelerator;

chmod -R 777 /tmp/eaccelerator;

打开/usr/local/php/etc/php.ini //红色数字部分需要根据“/usr/local/php/bin/phpize;”生成的数字自己修改；

找到
extension_dir
改为
extension_dir ="/usr/local/php/lib/php/extensions/no-debug-non-zts-20090626/"

最后添加如下代码

```
[eaccelerator]
extension="/usr/local/php/lib/php/extensions/no-debug-non-zts-20090626/eaccelerator.so"
eaccelerator.shm_size="32"
eaccelerator.cache_dir="/tmp/eaccelerator"
eaccelerator.enable="1"
eaccelerator.optimizer="1"
eaccelerator.check_mtime="1"
eaccelerator.debug="0"
eaccelerator.filter=""
eaccelerator.shm_max="0"
eaccelerator.shm_ttl="0"
eaccelerator.shm_prune_period="0"
eaccelerator.shm_only="0"
eaccelerator.compress="1"
eaccelerator.compress_level="9"
```

保存php.ini;

重启nginx;

检查eaccelerator是否成功安装：/usr/local/php/bin/php -v

看看mysql启动没有，如果没有的话，启动一下，完后做一些测试工作。

应该差不多就这样了。

欢迎各位提出修改意见，我会不断完善此文档