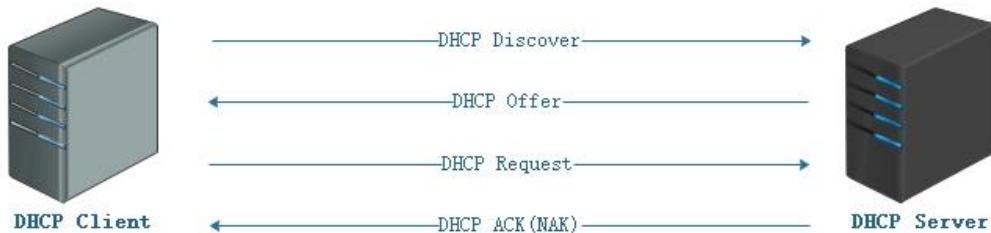


网络服务-DHCP

1. DHCP 简介

DHCP (Dynamic Host Configuration Protocol, 动态主机配置协议) 是一个工作在应用层的局域网网络协议，数据传输时使用 UDP 不可靠传输协议工作，通常被应用在大型的局域网络环境中，主要作用是集中的管理、分配网络资源，使网络环境中的主机能动态的获得 IP 地址、Gateway 地址、DNS 服务器地址等信息，并能够提升地址的使用率。

2. DHCP 工作原理（租约四部曲+续租）



2.1 DHCP 客户端进行 IP 请求

当一个 DHCP 客户机启动时，会自动将自己的 IP 地址配置成 0.0.0.0，由于使用 0.0.0.0 不能进行正常通信，所以客户机就必须通过 DHCP 服务器来获取一个合法的地址。由于客户机不知道 DHCP 服务器的 IP 地址，所以它**使用 0.0.0.0 的地址作为源地址，使用 255.255.255.255 作为目标地址，使用 UDP 67 端口作为目的端口来广播请求 IP 地址信息**。广播信息 DHCP Discover 中包含了 DHCP 客户机的 MAC 地址和计算机名，以便使 DHCP 服务器能确定是哪个客户机发送的请求。

2.2 DHCP 服务器响应请求

当DHCP服务器接收到客户机请求IP地址的信息时，它就在自己的IP地址池中查找是否有合法的IP地址提供给客户机。如果有，DHCP服务器就将此IP地址做上标记，加入到DHCP OFFER的消息中，然后DHCP服务器就广播一则包括下列信息的DHCP OFFER消息：

DHCP客户机的MAC地址；DHCP服务器提供的合法IP地址；子网掩码；默认网关（路由）；租约的期限；DHCP服务器的IP地址-MAC。

因为DHCP客户机还没有IP地址，所以**DHCP服务器使用自己的IP地址作为源地址，使用255.255.255.255作为目标地址，使用UDP 68端口作为源端口来广播DHCP OFFER信息**

2.3 DHCP 客户机选择 IP

DHCP客户机从接收到的第一个DHCP OFFER消息中选择IP地址，发出IP地址的DHCP服务器将该地址保留，这样该地址就不能提供给另一个DHCP客户机。当客户机从第一个DHCP服务器接收DHCP OFFER并选择IP地址后，DHCP租约的第三过程发生。客户机将DHCP REQUEST消息广播到所有的DHCP服务器，表明它接受提供的内容。DHCP REQUEST消息包括为该客户机提供IP配置的服务器的服务标识符（IP地址）。DHCP服务器查看服务器标识符字段，以确定它自己是否被选择为指定的客户机提供IP地址，如果那些DHCP OFFER被拒绝，则DHCP服务器会取消提供并保留其IP地址以用于下一个IP租约请求。

在客户机选择IP的过程中，虽然客户机选择了IP地址，但是还没有配置IP地址，而在一个网络中可能有几个DHCP服务器，所以**客户机仍然使用0.0.0.0的地址作为源地址，使用255.255.255.255作为目标地址，使用UDP 67端口作为目的端口来广播DHCP REQUEST信息**

2.4 DHCP 服务器确认租约

服务器确认租约：DHCP ACK

DHCP服务器接收到DHCP REQUEST消息后，以DHCPACK消息的形式向客户机广播成功的确认，该消息包含有IP地址的有效租约和其他可能配置的信息。虽然服务器确认了客户机的租约请求，但是客户机还没有收到服务器的DHCPACK消息，所以**服务器仍然使用自己的IP地址作为源地址，使用255.255.255.255作为目标地址，使用UDP 68端口作为源端口来广播DHCP ACK信息**。当客户机收到DHCP ACK消息时，它就配置了IP地址，完成了TCP/IP的初始化。

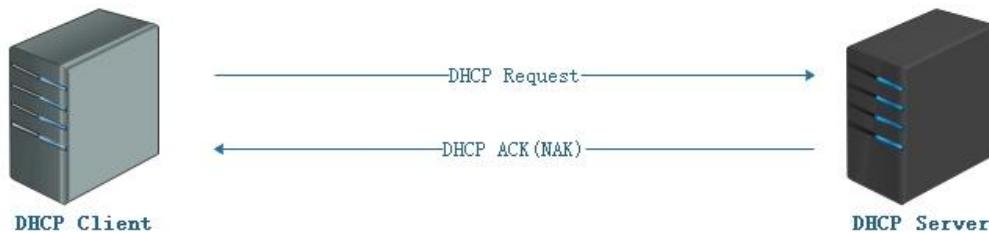
服务器拒绝租约：DHCP NACK (DHCP NAK)

如果DHCP REQUEST不成功，例如客户机试图租约先前的IP地址，但该IP地址不再可用，或者因为客户机移到其他子网，该IP无效时，DHCP服务器将广播否定确认消息DHCP NACK。当客户机接收到不成功的确认时，它将重新开始DHCP租约过程。

注1：如果DHCP客户机无法找到DHCP服务器，它将从TCP/IP的B类网段169.254.0.0/16中挑选一个IP地址作为自己的IP地址，继续每隔5分钟尝试与DHCP服务器进行通讯，一旦与DHCP服务器取得联系，则客户机放弃自动配置的IP地址，而使用DHCP服务器分配的IP地址。

注2：DHCP客户机收到DHCP服务器回应的ACK报文后，通过地址冲突检测（arp）发现服务器分配的地址冲突或者由于其他原因导致不能使用，则发送DECLINE报文，通知服务器所分配的IP地址不可用。

2.5 DHCP 客户机续租



DHCP客户机会在租期过去50%的时候，直接向为其提供IP地址的DHCP服务器发送DHCP REQUEST消息包。如果客户机接收到该服务器回应的DHCP ACK消息包，客户机就根据包中所提供的新的租期以及其它已经更新的TCP/IP参数，更新自己的配置，IP租用更新完成。如果没有收到该服务器的回复，则客户机继续使用现有的IP地址，因为当前租期还有50%。

如果在租期过去50%的时候没有更新，则DHCP客户机将在租期过去87.5%的时候再次向为其提供IP地址的DHCP服务器联系。如果还不成功，到租约的100%时候，DHCP客户机必须放弃这个IP地址，重新申请。**如果此时无DHCP服务器可用，DHCP客户机会使用169.254.0.0/16中随机的一个地址，并且每隔5分钟再进行尝试。**

3. DHCP 服务搭建

3.1 准备实验环境

两台机器，网络连接模式设为自定义VMnet*模式

防护的关闭：

1. `iptables -L` #防火墙
2. `getenforce` #SELinux
3. **关闭 VMware 虚拟网络编辑器的DHCP功能，切记**

3.2 DHCP 相关信息

软件名：

<code>dhcpc</code>	#DHCP服务软件包
<code>dhcpc-common</code>	#DHCP命令软件包（默认已安装）

服务名：

<code>dhcpd</code>	#DHCP服务名
<code>dhcrelay</code>	#DHCP中继服务名

端口号：

<code>udp 67</code>	#作为客户端的目标端口，接收客户端的请求DHCP请求
---------------------	----------------------------

```
udp 68          #作为服务器的源端口，用来向客户端回复数据包
```

配置文件：

```
dhcpd /etc/dhcp/dhcpd.conf      #此配置文件默认是空的，需要找模板文件重新生成
```

```
dhcpd.conf.sample /usr/share/doc/dhcp-4.*.*/dhcpd.conf.sample
```

#DHCP的模板配置文件

```
dhcrelay /etc/sysconfig/dhcrelay    #该文件时中继配置文件，中继实验中用到
```

3.3 DHCP 配置文件详解

```
subnet 192.168.88.0 netmask 255.255.255.0 {  
    range 192.168.88.3 192.168.88.254;  
    option domain-name "atguigu.com";  
    option domain-name-servers 8.8.8.8;  
    option routers 192.168.88.2;      #声明要分配的网段和子网掩码  
    option broadcast-address 192.168.88.255;  #声明可用 IP 地址池  
    default-lease-time 600;           #设置 DNS 域  
    max-lease-time 7200;            #设置 DNS 服务器地址  
}  
                                #默认网关的地址  
                                #广播地址（可不写）  
                                #默认租约（s）  
                                #最大租约（s）
```

4. DHCP 实验部署

4.1 DHCP 基本功能实验

4.1.1 生成配置文件

```
cp -a /usr/share/doc/dhcp-4.*.*/dhcpd.conf.sample /etc/dhcp/dhcpd.conf
```

4.1.2 修改配置文件

将配置文件的前几个 subnet 声明注释掉。修改最后一个 subnet 声明

注：注意配置文件中每行结尾的分号和结束大括号，谢谢！

```
subnet 192.168.88.0 netmask 255.255.255.0 {  
    range 192.168.88.3 192.168.88.254;  
    option domain-name "atguigu.com";  
    option domain-name-servers 8.8.8.8;  
    option routers 192.168.88.2;      #  
    option broadcast-address 192.168.88.255;  
    default-lease-time 600;  
    max-lease-time 7200;  
}
```

4.1.3 重启服务

```
service dhcpcd start
```

4.1.4 重启客户机的网卡

```
ifdown eth0; ifup eth0
```

4.2 保留地址（固定地址分配）

4.2.1 获取客户端的mac地址

```
arp -a #查看客户机的mac地址
```

4.2.2 修改/etc/dhcp/dhcpd.conf文件

```
host fantasia {  
    hardware ethernet mac地址;      #客户机的mac地址  
    fixed-address IP地址;          #固定分配给客户机的ip地址（可以使用地址池以外的IP）  
}
```

4.2.3 重启DHCP服务

```
service dhcpcd restart
```

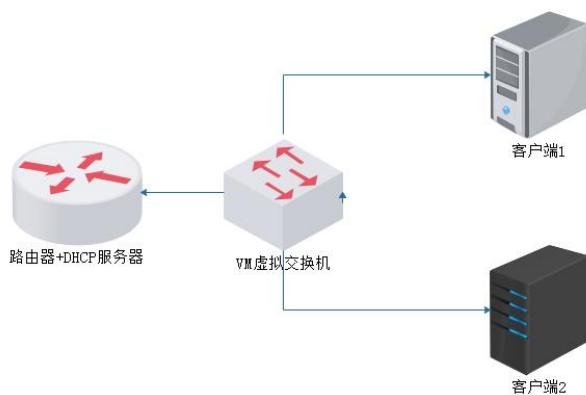
4.2.4 重启客户机网卡验证IP获取是否成功

```
ifdown eth0; ifup eth0
```

4.3 超级作用域（同一局域网）

4.3.1 超级作用域介绍

DHCP服务器可为单个物理网络上的客户端提供多个作用域租约地址



4.3.2 实验环境准备

三台虚拟机同一网络模式，一个DHCP服务器，两个客户机

4.3.3 实验步骤

1. 设置DHCP服务器的单臂路由所需子网卡：

```
cp -a ifcfg-eth0 ifcfg-eth0:0 #编辑此文件，修改网卡名和IP地址即可
```

2. 开启路由转发：

```
vim /etc/sysctl.conf
```

```
net.ipv4.ip_forward = 1 #此选项修改为1即可  
sysctl -p #刷新内核参数配置文件
```

3. 修改/etc/dhcp/dhcpd.conf文件

#之前的网段声明和主机声明全都注释掉！

```

shared-network public {
    subnet 192.168.88.0 netmask 255.255.255.0 {
        option routers 192.168.88.10;
        range 192.168.88.100 192.168.88.100; }
    subnet 192.168.99.0 netmask 255.255.255.0 {
        option routers 192.168.99.10;
        range 192.168.99.100 192.168.99.110; }
}

```

#剩余内容注释掉或删除掉，切记别落下括号

4. 重启DHCP服务
service dhcpcd restart
5. 分别重启两台机器的网卡，查看获取的地址
ifdown eth0; ifup eth0



4.4 DHCP 中继

4.4.1 DHCP中继介绍

DHCP Relay (DHCP中继) 是一个小程序，可以实现在不同子网和物理网段之间处理和转发dhcp信息的功能。



4.4.2 实验环境准备

DHCP服务器:

eth0 (192.168.10.10) VMnet10

DHCP中继:

eth0 (192.168.10.20) VMnet10

eth1 (100.100.100.20) VMnet11

外网客户机:

eth0 (IP地址自动获取) VMnet11

注: 关闭所有防护: iptables、SELinux

4.4.3 配置DHCP服务器

1. 软件安装:

yum -y install dhcpcd

2. 修改/etc/dhcp/dhcpcd.conf文件:

#声明两个subnet，其他无关可以不做操作或删除

```

subnet 192.168.10.0 netmask 255.255.255.0 {
    range 192.168.10.100 192.168.10.110;
    option routers 192.168.10.20;
}

subnet 100.100.100.0 netmask 255.255.255.0 {
    range 100.100.100.100 100.100.100.110;
    option routers 100.100.100.20;
}

```

#实验中并未用到该地址池分配IP

- ```

 }
3. 重启dhcpd服务:
 service dhcpd start
4. 指定网关:
 只能中继器的内网IP为网关地址

```

#### 4.4.4 配置DHCP中继服务器

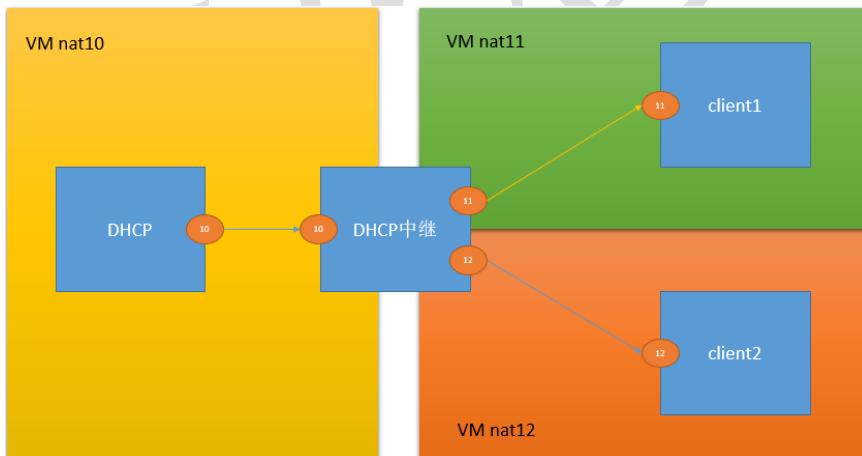
- 网卡配置  
一块网卡ip=192.168.10.20  
一块网卡ip=100.100.100.20
- 软件安装  
yum -y install dhcp
- 修改中继配置文件  
vim /etc/sysconfig/dhcrelay文件  
INTERFACES="eth0 eth1"  
DHCPSERVERS="192.168.10.10"
- 开启路由转发  
vim /etc/sysctl.conf文件。  
net.ipv4.ip\_forward = 1  
sysctl -p

- 重启中继服务  
service dhcrelay start

#### 4.4.5 测试外网主机

重启网卡 ifdown ifup ifconfig

#### 4.4.6 拓展实验



注: 此图和实验规划有所区别, 实验规划只是用一台测试机, 此图使用了两台

# 网络服务—DNS 域名系统服务

## 1. DNS 介绍

### 1.1 什么是域名？

域名（Domain Name），简称域名、网域，是由一串用点分隔的名字组成的 Internet 上某一台计算机或计算机组的名称，用于在数据传输时标识计算机的电子方位。具有独一无二，不可重复的特性。

### 1.2 什么是 DNS？

域名系统（Domain Name System，缩写：DNS）是互联网的一项服务。域名解析是把域名指向网站空间 IP，让人们通过注册的域名可以方便地访问到网站的一种服务。IP 地址是网络上标识站点的数字地址，为了方便记忆，采用域名来代替 IP 地址标识站点地址。域名解析就是域名到 IP 地址的转换过程。域名的解析工作由 DNS 服务器完成。可以理解为 DNS 就是翻译官。

正向解析：域名 --> IP 地址  
反向解析：IP 地址 --> 域名

### 1.3 域名的组成和分类

常见格式：[www.atguigu.com](http://www.atguigu.com)

完整格式：[www.atguigu.com.](http://www.atguigu.com.)

. : 根域，可省略不写

com: 顶级域，由 ICANN 组织指定和管理

分类：

国家地区域名：cn（中国）、hk（香港）、sg（新加坡）等

通用顶级域名：com（商业机构）、org（非营利组织）、edu（教育机构）等

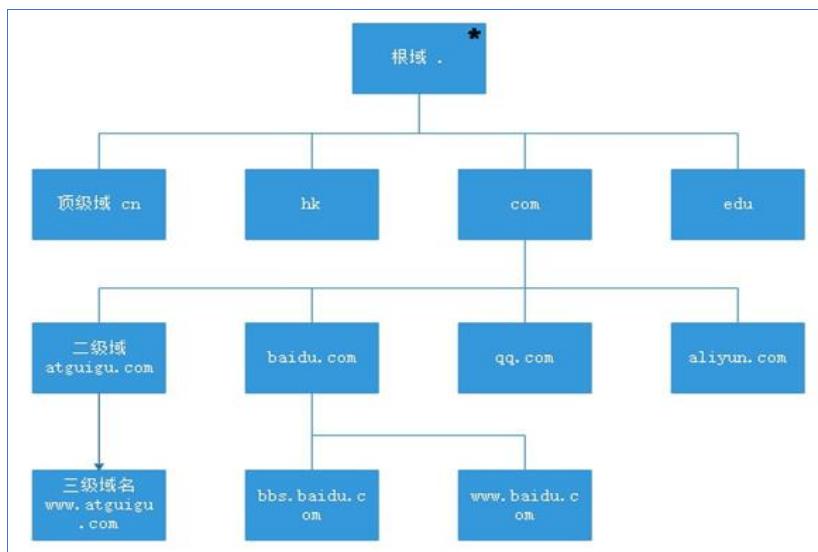
新通用顶级域名：red（红色、热情）、top（顶级、高端）等

atguigu: 二级域（注册域），可由个人或组织申请注册

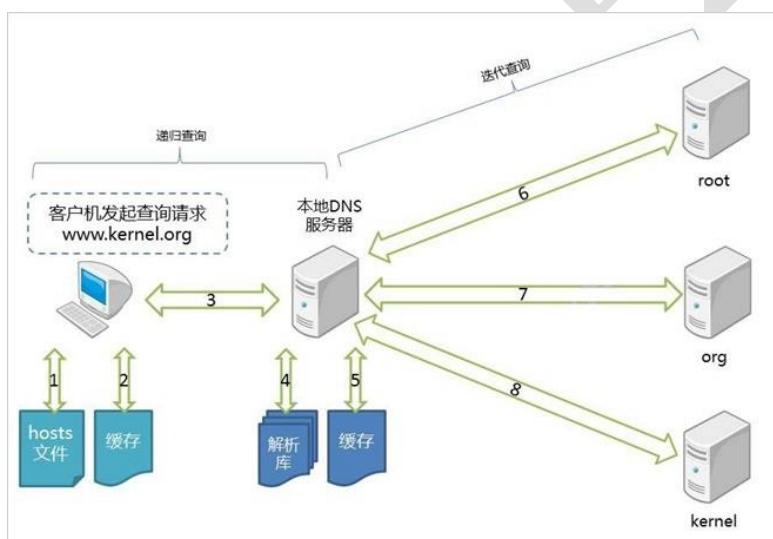
www: 三级域（子域），服务器网站名代表

主机名：s1.www.atguigu.com. 中的 s1 就是主机名，一般用来表示具体某一台主机

拓展：com.cn 属于“二级域名”，是 cn 顶级域的子域



## 2. 域名解析过程



1. 客户机首先查看**查找本地 hosts 文件**，如果有则返回，否则进行下一步。
2. 客户机**查看本地缓存**，是否存在本条目的缓存，如果有则直接返回，否则进行下一步。
3. 将请求转发给指向的 DNS 服务器。
4. 查看域名是否本地解析，是则本地解析返回，否则进行下一步。
5. 本地 DNS 服务器首先在缓存中查找，有则返回，无则进行下一步。
6. 向**全球 13 个根域服务器发起 DNS 请求**，根域返回 org 域的地址列表。
7. 使用某一个 org 域的 IP 地址，发起 DNS 请求，org 域返回 kernel 域服务器地址列表。
8. 使用某一个 kernel 域 IP 地址，发起 DNS 请求，kernel 域返回 www.kernel.org 主机的 IP 地址，本地 DNS 服务收到后，返回给客户机，并在本地 DNS 服务器保存一份。

### 3. DNS 软件信息

软件名称:

bind

服务名称:

named

软件端口:

UDP 53 数据通信（域名解析）

TCP 53 数据同步（主从同步）

配置文件:

**主配置文件: /etc/named.conf** (服务器运行参数)

```
options {
 listen-on port 53 { 127.0.0.1; };
 listen-on-v6 port 53 { ::1; }; 设置服务器监听网卡 (可以写具体某一个IP, 也可以写成any)
 directory "/var/named";
 dump-file "/var/named/data/cache_dump.db"; 数据文件位置
 statistics-file "/var/named/data/named_stats.txt";
 memstatistics-file "/var/named/data/named_mem_stats.txt";
 allow-query { localhost; }; 设置可以访问服务器的客户端IP (可用any)
 recursion yes;
```

**区域配置文件: /etc/named.rfc1912.zones** (服务器解析的区域配置, 正反向区域定义信息)

```
zone "localhost.localdomain" IN {
 type master; 正向区域配置文件标签, 修改为要解析的域
 file "named.localhost"; 正向数据配置文件名称 (默认保存在/var/named下)
 allow-update { none; }; 允许数据更新的列表 (填写IP地址)
};

zone "1.0.0.127.in-addr.arpa" IN {
 type master; 反向区域配置文件标签, 仅修改IP位置, 并且将IP反写
 file "named.loopback"; 例如: 0.168.192.in-addr.arpa
 allow-update { none; };
};
```

**数据配置文件: /var/named/xx.xx** (主机名和 IP 地址的对应解析关系, 及主从同步信息)

```
$TTL 1D 域名有效解析生存周期 (一般指缓存时间)
@ IN SOA @ rname.invalid. (
 0 ; serial 配置文件修改版本(如:20190826)
 1D ; refresh 更新频率 (从向主的查询周期)
 1H ; retry 更新失败的重试时间周期
 1W ; expire 无法更新时的失效周期
 3H) ; minimum 缓存服务器无法更新时的失效时间
NS @ 设置DNS服务器的域名
A 127.0.0.1 IPv4的域名IP解析记录
AAAA ::1 IPv6的域名IP解析记录
```

记录类型:

|        |                                                  |
|--------|--------------------------------------------------|
| A:     | 地址记录, 用来指定域名的 IPv4 地址的记录                         |
| CNAME: | 将域名指向另一个域名, 再由另一个域名提供 ip 地址, 就需要添加 CNAME 记录      |
| TXT:   | 可填写任何东西, 长度限制 255。绝大多数的 TXT 记录是用来做 SPF 的 (反垃圾邮件) |
| NS:    | 域名服务器记录, 如果需要把子域名交给其他 DNS 服务商解析, 就需要添加 NS 记录。    |
| AAAA:  | 地址记录, 用来指定域名的 IPv6 地址的记录                         |

|     |                                       |
|-----|---------------------------------------|
| MX: | 邮件交换记录，如果需要设置邮箱，让邮箱能收到邮件，就需要添加 MX 记录。 |
|-----|---------------------------------------|

## 4. DNS 实验搭建

### 4.1 DNS 服务搭建

#### 先关闭服务器和客户机上的防火墙和 SELinux

##### 1. 软件安装

```
yum -y install bind
```

##### 2. 配置主配置文件 (/etc/named.conf)

##### 3. 配置区域文件 (/etc/named.rfc1912.zones)

注：先对区域文件进行备份，删除多余的模板，只留下一个正向和一个反向（反向修改时，网络位的反写格式，如 192.168.100.2-->100.168.192.）

##### 4. 配置数据文件/var/named/

A. 先复制生成正向解析文件和反向解析文件

B. 编辑正向解析文件（注意域名结尾的“.”）

C. 编辑反向解析文件（注意域名结尾的“.”）

##### 5. 重启 DNS 服务

```
service named restart
```

##### 6. 客户端测试

在网卡配置文件中添加 DNS 服务器的地址，然后用 nslookup 测试。

### 4.2 主从 DNS 服务器

#### 实验目的：

减轻主服务器的压力

#### 先关闭服务器和客户机上的防火墙和 SELinux

#### 实验准备：

一台主服务器、一台从服务器、一台测试机

#### 搭建过程：

##### 1. 搭建主服务器步骤（同上，不截图了）：

- 安装 bind 软件
- 主配置文件的修改
- 区域配置文件的修改
- 配置数据文件  
    正向数据文件  
    反向数据文件（可选做）
- 启动 named 服务

注意：主 DNS 的区域配置文件中 allow-updates 参数添加从服务器 IP 地址。

##### 2. 搭建从服务器步骤：

- 安装 bind 软件

- b. 修改主配置文件/etc/named.conf
- c. 配置区域文件 (/etc/named.rfc1912.zones)

注意：从配置文件的类型需要修改为 slave，并且需要填写主服务器的地址，如下

```
type slave;
masters { 192.168.0.10; }; #大括号两侧留有空格
文件保存位置修改为 file "slaves/atguigu.localhost";
```

- d. 重启服务
- e. 在测试机上填写从服务器的 IP，并使用 nslookup 测试

## 4.3 DNS 缓存服务器

先关闭服务器和客户机上的防火墙和 SELinux

实验作用：

加快解析速度，提高工作效率

实验软件：

dnsmasq

配置文件：

```
/etc/dnsmasq.conf
domain=域名 #需要解析的域名
server=ip #主 DNS 服务器 IP
cache-size=15000 #声明缓存条数
```

重启服务：

service dnsmasq restart

测试效果：

在测试机上填写 DNS 缓存服务器的 ip 地址

## 4.4 智能 DNS（分离解析）

**实验原理：**DNS 分离解析即将相同域名解析为不同的 IP 地址。现实网络中一些网站为了让用户有更好的体验效果解析速度更快，就把来自不同运营商的用户解析到相对应的服务器这样就大大提升了访问速度

实验环境：

- 一台内网测试机（单网卡）
- 一台网关+DNS（双网卡）
- 一台外网测试机（单网卡）
- 一台 web 服务器（双网卡）

先关闭服务器和客户机上的防火墙和 SELinux

实验步骤：

1. 安装 bind 软件
2. 内核配置文件开启路由转发，修改/etc/sysctl.conf
3. 修改主配置文件/etc/named.conf

```
view lan {
 match-clients { 192.168.10.0/24; };
 zone "." IN {
 type hint;
 file "named.ca";
 };
 include "/etc/lan.zones";
};

view wan {
 match-clients { any; };
 zone "." IN {
 type hint;
 file "named.ca";
 };
 include "/etc/wan.zones";
};
#include "/etc/named.rfc1912.zones";
include "/etc/named.root.key";
```

注意：不同的解析放在了各自的区域配置文件（便于区分和维护更新）

#### 4. 生成自己定义的区域文件（反向解析省略掉了）

```
cp -a named.rfc1912.zones lan
cp -a named.rfc1912.zones wan
```

#### 5. 配置数据文件

配置内网的正向解析文件  
配置外网的正向解析文件

#### 6. 重启服务

```
service named restart
```

#### 7. 效果测试

内网客户端网卡配置  
将 dns 和网关都指为网关服务器的内网口地址  
外网客户端网卡配置  
将 dns 和网关都指为网关服务器的外网口地址

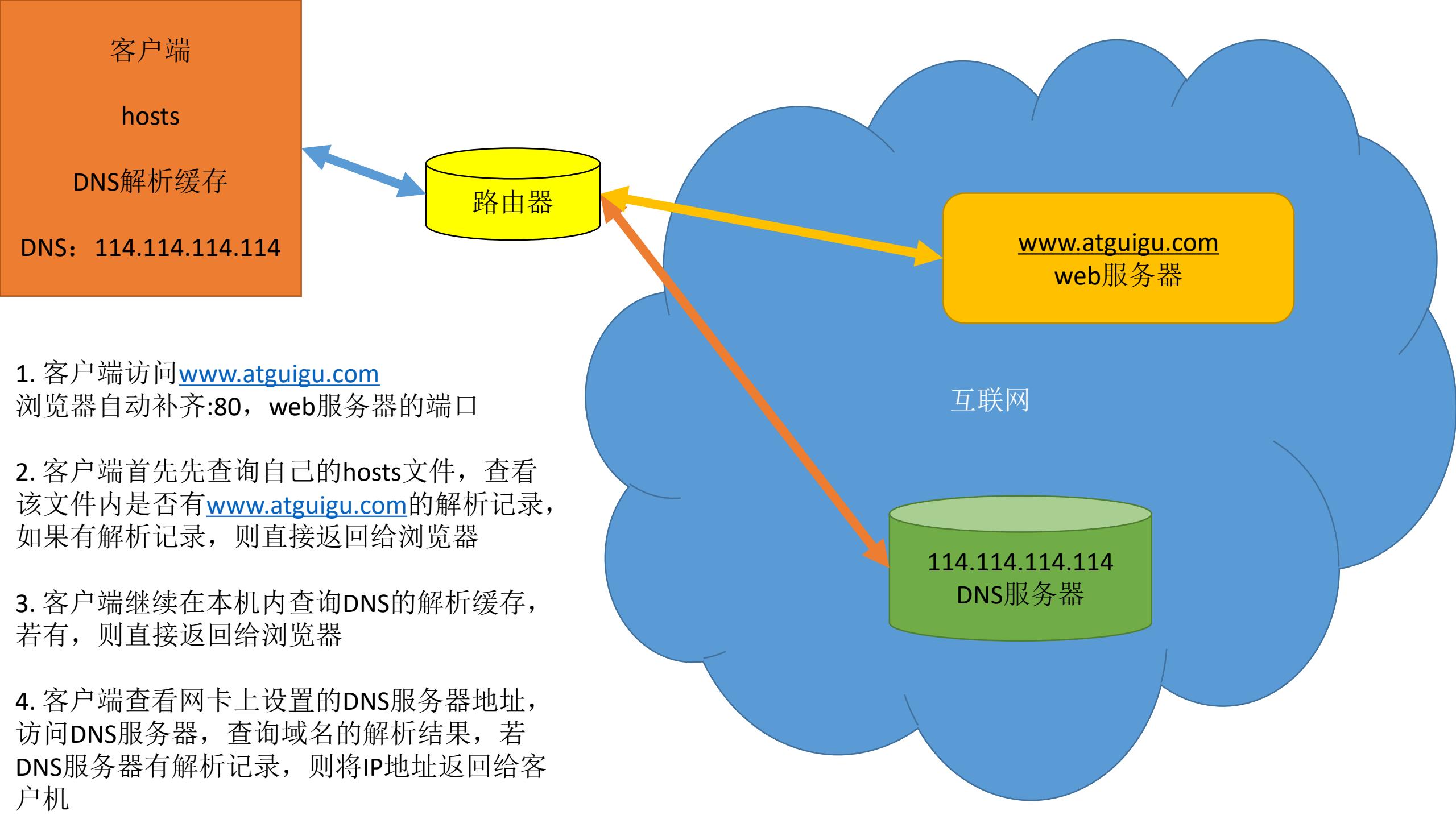
主根域 .

辅根域 .

辅根域 .

....

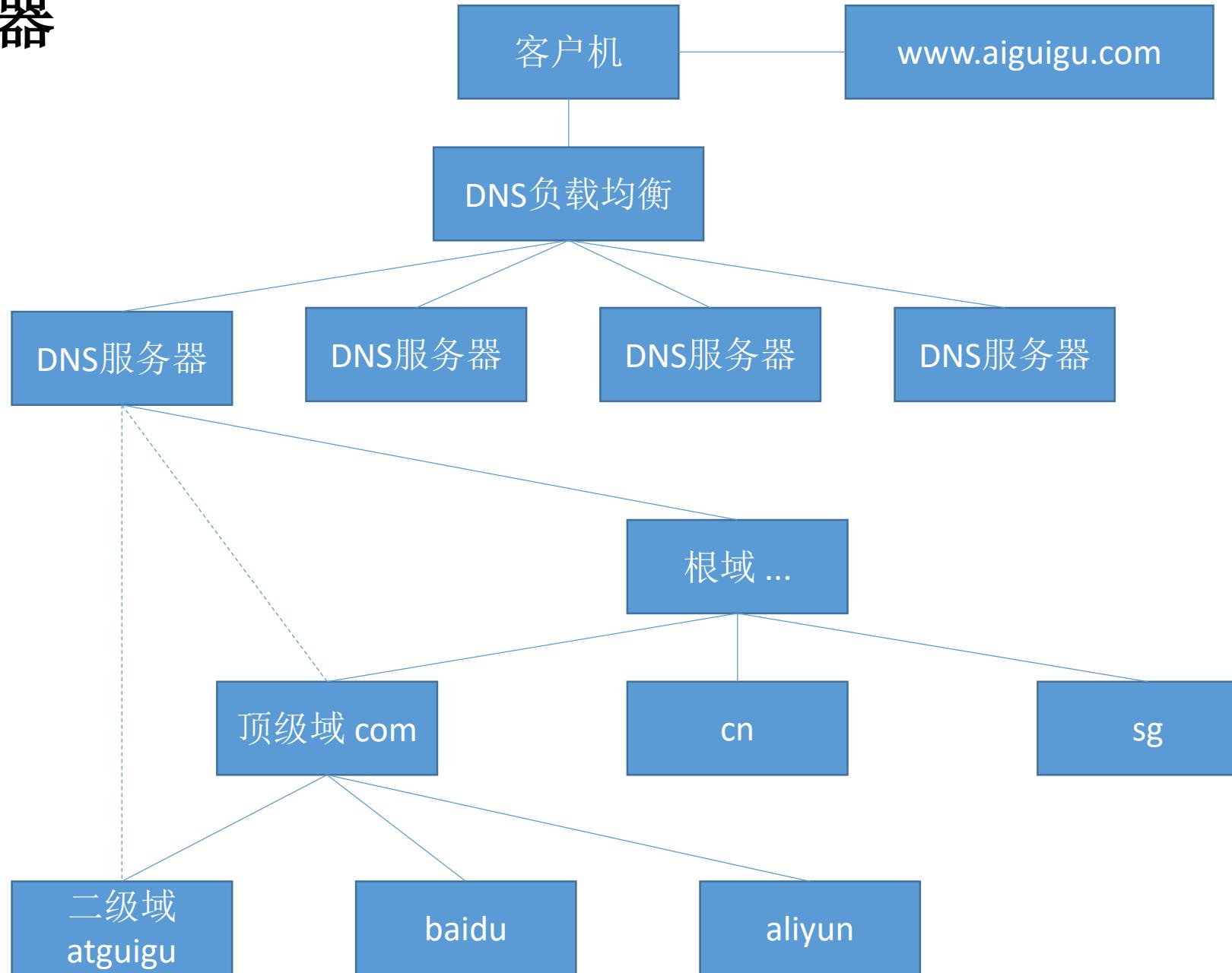
镜像根域 .



# 拓展：分布式 DNS 服务器

**分布式：**同一个任务，由不同步骤共同完成的过程就叫分布式（生产车间中的某一条流水线，流水线上有很多步骤，不同步骤之间就叫分布式）

**负载均衡：**将用户的请求，分配到多个功能相同的服务器上（一个生产车间中的多条相同功能的流水线）



DNS  
IP: 192.168.88.20  
/etc/named.conf  
/etc/named.rfc1912.zones  
/var/named/\*localhost  
/var/named/\*loopback

Client  
IP: 192.168.88.10  
在网卡上填写DNS地址

httpd  
IP: 192.168.88.30  
/var/www/html/index.html

主DNS  
/etc/named.conf  
/etc/named.rfc1912.zones  
/var/named/\*.localhost  
/var/named/\*.loopback

从DNS  
/etc/named.conf  
/etc/named.rfc1912.zones  
/var/named/slave/\*.localhost  
/var/named/slave/\*.loopback

Client  
在网卡上填写从DNS地址

httpd  
/var/www/html/index.html

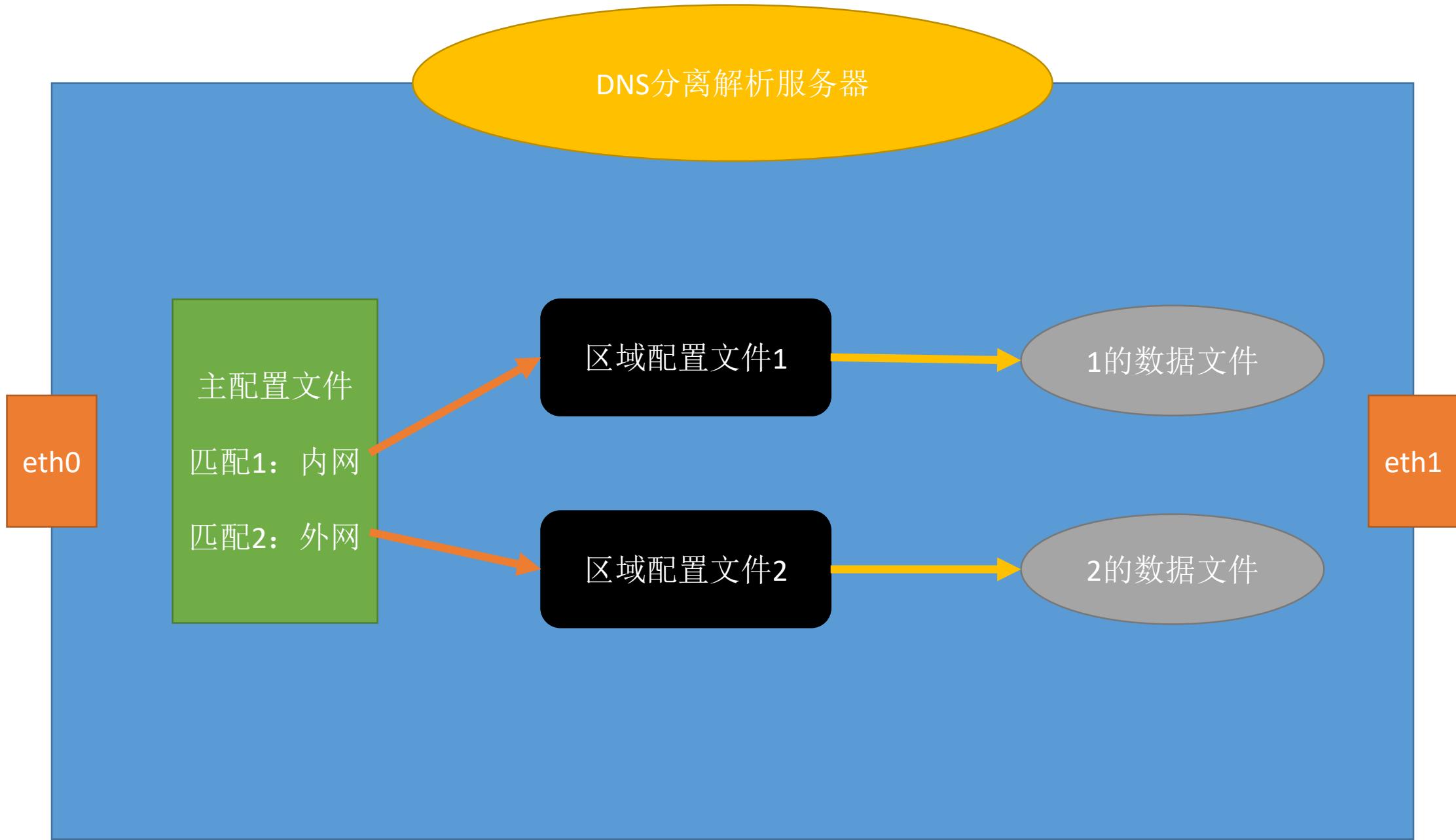
0

主配置文件  
匹配访问用户  
决定  
是否允许访问

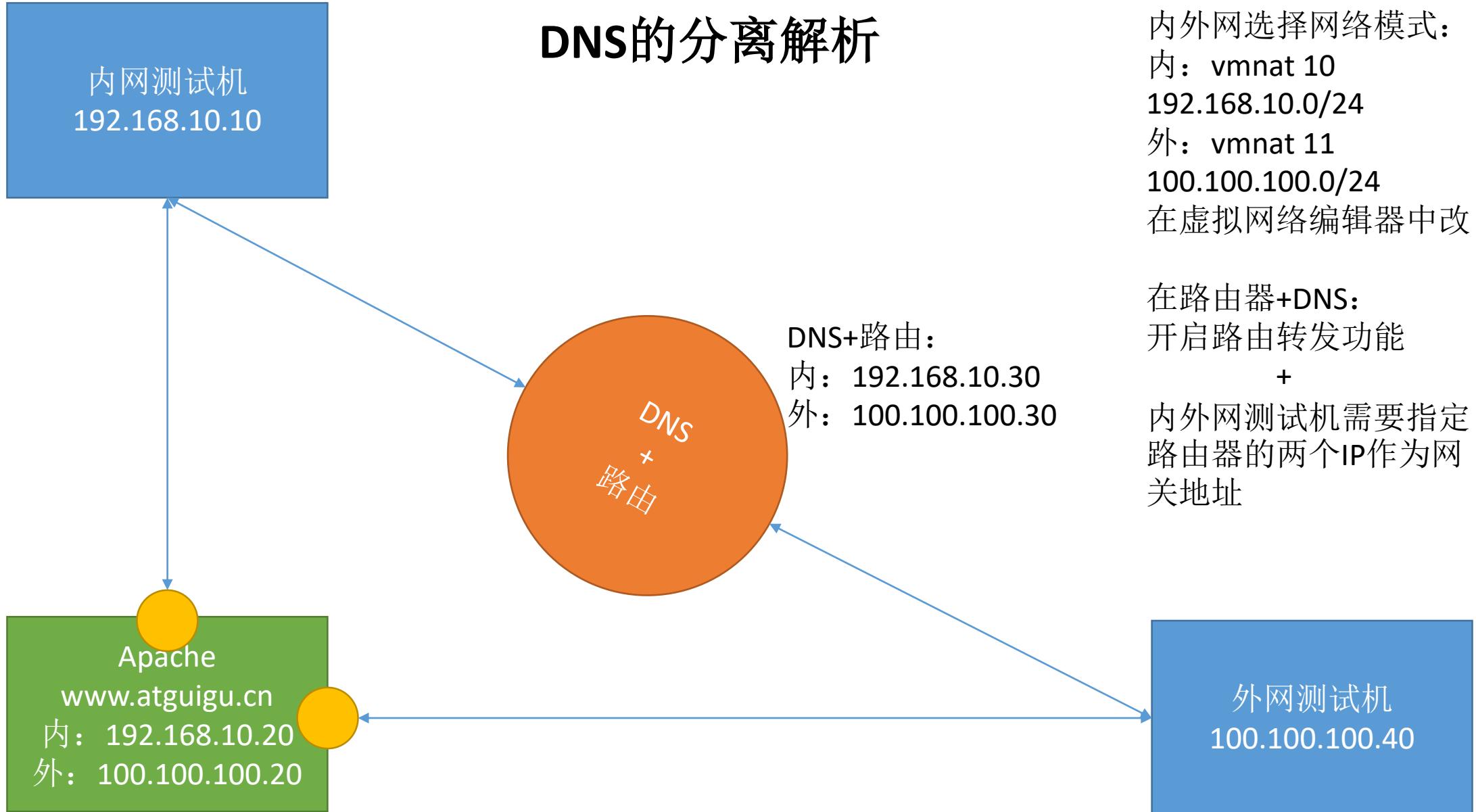
区域配置文件  
正: atguigu.cn  
反: 192.168.100.N

数据配置文件  
正向解析记录  
www.atguigu.cn 192.168.100.10  
book.atguigu.cn 192.168.100.11  
image.atguigu.cn 192.168.100.12

数据配置文件  
反向解析记录  
192.168.100.10 www.atguigu.cn



# DNS的分离解析



# 网络服务-VSFTP

## 1. VSFTP 概述

FTP 是 File Transfer Protocol (文件传输协议) 的英文简称，用于 Internet 上的文件的双向传输。使用 FTP 来传输时，是具有一定程度的危险性，因为数据在因特网上面是完全没有受到保护的明文传输方式！

VSFTP 是一个基于 GPL 发布的类 Unix 系统上使用的 FTP 服务器软件，它的全称是 Very Secure FTP，从名称定义上基本可以看出，这是为了解决 ftp 传输安全性问题的。

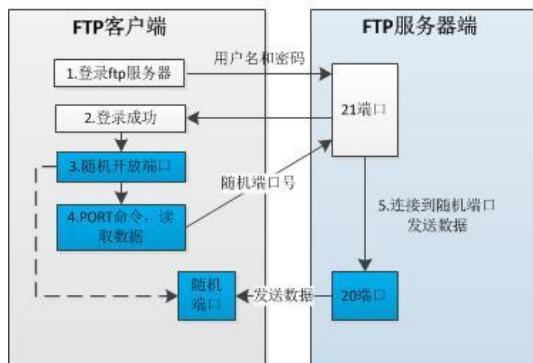
### 1.1 安全特性

1. vsftpd 程序的运行者一般是普通用户，降低了相对应进程的权限，提高了安全性
2. 任何需要执行较高权限的指令都需要上层程序许可
3. ftp 所需要使用的绝大多数命令都被整合到了 vsftpd 中，基本不需要系统额外提供命令
4. 拥有 chroot 功能，可以改变用户的根目录，限制用户只能在自己的家目录

## 2. VSFTP 连接类型



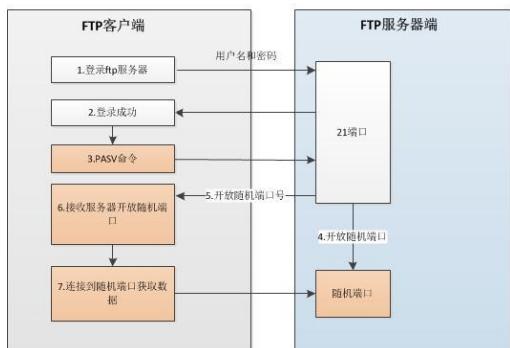
## 3. VSFTP 工作模式



### Port模式

FTP 客户端首先和服务器的 TCP 21 端口建立连接，用来发送命令，客户端需要接收数据的时候在这个通道上发送 PORT 命令。PORT 命令包含了客户端用什么端口接收数据。在传送数据的时候，服务器端通过自己的 TCP 20 端口连接至客户端的指定端口发送数据。FTP server 必须和客户端建立一个新的连接用来传送数据。

被动模式



### Passive 模式

FTP 客户端首先和服务器的 TCP 21 端口建立连接，用来建立控制通道发送命令，但建立连接后客户端发送 Pasv 命令。服务器收到 Pasv 命令后，打开一个临时端口（端口大于 1023 小于 65535）并且通知客户端在这个端口上上传送数据的请求，客户端连接 FTP 服务器的临时端口，然后 FTP 服务器将通过这个端口传输数据。

**注意：**由于VSFTP的被动模式是随机端口进行数据传输，所以在设置防火墙时需要刻意放行。

## 4. VSFTP 传输模式

**Binary模式：**不对数据进行任何处理，适合进行可执行文件、压缩文件、图片等

**ASCII模式：**进行文本传输时，自动适应目标操作系统的结束符，如回车符等

Linux的红帽发行版中VSFTP默认采用的是Binary模式，这样能保证绝大多数文件传输后能正常使用

**切换方式：**在ftp>提示符下输入ascii即转换到ACSI方式，输入bin，即转换到Binary方式。

## 5. VSFTP 软件信息

**服务端软件名：** vsftpd

**客户端软件名：** ftp

**服务名：** vsftpd

**端口号：** 20、21、指定范围内随机端口

**配置文件：** /etc/vsftpd/vsftpd.conf

## 6. 登录验证方式

### 匿名用户验证：

用户账号名称：ftp或anonymous

用户账号密码：无密码

工作目录：/var/ftp

默认权限：默认可下载不可上传，上传权限由两部分组成（主配置文件和文件系统）

### 本地用户验证：

用户账号名称：本地用户（/etc/passwd）

用户账号密码：用户密码（/etc/shadow）

工作目录：登录用户的宿主目录

权限：最大权限（drwx-----）

### 虚拟 (virtual) 用户验证:

1. 创建虚拟用户用来代替本地用户，减少本地用户曝光率
2. 使用本地用户作为虚拟用户的映射用户，为虚拟用户提供工作目录和权限控制
3. 能够设置严格的权限（为每一个用户生成单独的配置文件）

## 7. VSFTP 实验部署

**注: 先关闭服务器和客户机上的防火墙和 SELinux**

### 7.1. 匿名用户验证实验:

#### 匿名权限控制:

```
anonymous_enable=YES #启用匿名访问
anon_umask=022 #匿名用户所上传文件的权限掩码
anon_root=/var/ftp #匿名用户的 FTP 根目录
anon_upload_enable=YES #允许上传文件
anon_mkdir_write_enable=YES #允许创建目录
anon_other_write_enable=YES #开放其他写入权(删除、覆盖、重命名)
anon_max_rate=0 #限制最大传输速率 (0 为不限速, 单位: bytes/秒)
```

#### 实验需求与流程:

注意: 在客户端登录后, 默认情况下是可以下载的, 但不能上传

1. 实现可以上传
  - a. anon\_upload\_enable=YES
  - b. 在/var/ftp/下创建上传目录
  - c. 修改上传目录的权限或所有者, 让匿名用户有写入权限
2. 实现创建目录和文件其他操作

```
anon_mkdir_write_enable=YES #允许创建目录
anon_other_write_enable=YES #删除文件、文件改名、文件覆盖
```
3. 用户进入某个文件夹时, 弹出相应的说明
  - a. 在对应目录下创建 .message 文件, 并写入相应内容
  - b. 确认dirmessage\_enable=YES是否启用
  - c. 尝试切换目录查看效果 (同一次登录仅提示一次)
4. 实现上传的文件可下载  
默认情况下开放上传权限后, 上传的文件是无法被下载的, 因为文件的其他人位置没有r权限  
设置anon\_umask=022, 可以让上传的文件其他人位置拥有r权限, 然后才能被其他人下载

## 7.2. 本地用户验证实验：

本地用户权限控制：

```
local_enable=YES #是否启用本地系统用户
local_umask=022 #本地用户所上传文件的权限掩码
local_root=/var/ftp #设置本地用户的 FTP 根目录
chroot_local_user=YES #是否将用户禁锢在主目录
local_max_rate=0 #限制最大传输速率
ftpd_banner=Welcome to blah FTP service #用户登录时显示的欢迎信息
userlist_enable=YES & userlist_deny=YES
#禁止/etc/vsftpd/user_list 文件中出现的用户名登录 FTP
userlist_enable=YES & userlist_deny=NO
#仅允许/etc/vsftpd/user_list 文件中出现的用户名登录 FTP
配置文件： ftpusers
#禁止/etc/vsftpd/ftpusers 文件中出现的用户名登录 FTP,权限比 user_list 更高，即时生效
```

实验需求与流程：

1. 服务端需要创建用户并设置密码(所创建的用户，不需要登录操作系统，仅用来登录VSFTP)  
`useradd -s /sbin/nologin username`
2. 将所有用户禁锢在自己的家目录下  
注：默认没有禁锢用户时，客户端登录后可以随意切换目录，查看文件所在位置和文件名  
`chroot_local_user=YES`  
`#开启用户家目录限制，限制所有用户不能随便切换目录`
3. 将部分用户禁锢在自己的家目录下  
`chroot_list_enable=YES`  
`#开启白名单功能，允许白名单中的用户随意切换目录`  
`chroot_list_file=/etc/vsftpd/chroot_list`  
`#白名单文件所在位置（需自己创建）`
4. 配置文件： /etc/vsftpd/ftpusers  
所有写入此文件内的用户名都不允许登录ftp，立刻生效。
5. 修改被动模式数据传输使用端口  
`pasv_enable=YES`  
`pasv_min_port=30000`  
`pasv_max_port=35000`

## 7.3. 虚拟用户验证实验：

1. 建立 FTP 的虚拟用户的用户数据库文件（在/etc/vsftpd）

```
vim vsftpd.user
```

注：该文件名可以随便定义，文件内容格式：奇数行用户名，偶数行密码

```
db_load -T -t hash -f vsftpd.user vsftpd.db
```

```
#将用户密码的存放文本转化为数据库类型，并使用 hash 加密
chmod 600 vsftpd.db
#修改文件权限为 600，保证其安全性
```

## 2. 创建 FTP 虚拟用户的映射用户，并制定其用户家目录

```
useradd -d /var/ftproot -s /sbin/nologin virtual
#创建 virtual 用户作为 ftp 的虚拟用户的映射用户
```

## 3. 建立支持虚拟用户的 PAM 认证文件，添加虚拟用户支持

```
cp -a /etc/pam.d/vsftpd /etc/pam.d/vsftpd.pam
#使用模板生成自己的认证配置文件，方便一会调用
编辑新生成的文件 vsftpd.pam (清空原来内容，添加下列两行)
auth required pam_userdb.so db=/etc/vsftpd/vsftpd
account required pam_userdb.so db=/etc/vsftpd/vsftpd
在 vsftpd.conf 文件中添加支持配置
修改：
```

```
pam_service_name=vsftpd.pam
```

添加：

```
guest_enable=YES
guest_username=virtual
user_config_dir=/etc/vsftpd/dir
```

## 4. 为虚拟用户建立独立的配置文件，启动服务并测试

注：做虚拟用户配置文件设置时，将主配置文件中自定义的匿名用户相关设置注释掉。

用户可以上传：

```
anon_upload_enable=YES #允许上传文件
```

用户可以创建目录或文件：

```
anon_mkdir_write_enable=YES #允许创建目录
```

用户可以修改文件名：

```
anon_upload_enable=YES #允许上传文件（为了覆盖开启的）
```

```
anon_other_write_enable=YES #允许重名和删除文件、覆盖
```

注：给映射用户的家目录 设置 o+rw 让虚拟用户有读权限。

## 7.4. openssl+vsftpd 加密验证方式：

拓展：使用 tcpdump 工具进行指定端口抓包，抓取 ftp 登录过程中的数据包

```
tcpdump -i eth0 -nn -X -vv tcp port 21 and ip host 来源ip
-i #interface：指定tcpdump需要监听的接口
-n #对地址以数字方式显示，否则显示为主机名
-nn #除了-n的作用外，还把端口显示为数值，否则显示端口服务名
-X #输出包的头部数据，会以16进制和ASCII两种方式同时输出
```

```
-vv #产生更详细的输出
```

1. 查看是否安装了 openssl

```
rpm -q openssl
```

2. 查看 vsftpd 是否支持 openssl

```
ldd /usr/sbin/vsftpd | grep libssl
```

3. 生成加密信息的秘钥和证书文件

位置: /etc/ssl/certs/

- a. openssl genrsa -out vsftpd.key 1024

#建立服务器私钥, 生成 RSA 密钥

- b. openssl req -new -key vsftpd.key -out vsftpd.csr

#需要依次输入国家, 地区, 城市, 组织, 组织单位, Email 等信息。最重要的是有一个 common name, 可以写你的名字或者域名。如果为了 https 申请, 这个必须和域名吻合, 否则会引发浏览器警报。生成的 csr 文件交给 CA 签名后形成服务端自己的证书

- c. openssl x509 -req -days 365 -sha256 -in vsftpd.csr -signkey vsftpd.key -out vsftpd.crt

#使用 CA 服务器签发证书, 设置证书的有效期等信息

**注意 1:** 生成完秘钥和证书文件后, 将本目录{/etc/ssl/certs/}的权限修改为 500.

**注意 2:** 在实验环境中可以用命令生成测试, 在生产环境中必须要在 https 证书厂商注册 (否则浏览器不识别)

4. 修改主配置文件/etc/vsftpd/vsftpd.conf

```
ssl_enable=YES
```

#启用 ssl 认证

```
ssl_tlsv1=YES
```

```
ssl_sslv2=YES
```

```
ssl_sslv3=YES
```

#开启 tlsv1、sslv2、sslv3 都支持

```
allow_anon_ssl=YES
```

#允许匿名用户 {虚拟用户}

```
force_anon_logins_ssl=YES
```

```
force_anon_data_ssl=YES
```

#匿名登录和传输时强制使用 ssl

```
force_local_logins_ssl=YES
```

```
force_local_data_ssl=YES
```

#本地登录和传输时强制使用 ssl

```
rsa_cert_file=/etc/ssl/certs/vsftpd.crt
```

#rsa格式的证书

```
rsa_private_key_file=/etc/ssl/certs/vsftpd.key
```

#rsa格式的密钥

**注:** 密钥文件要在配置文件中单独声明 (写入配置文件时, 注释要单独一行, 否则会报错)

5. 重启服务

```
service vsftpd restart
```

6. 测试(使用第三方客户端连接)



FileZilla-FTP（第三方客户端工具）

连接测试时选择：

服务器类型：显式 TLS/SSL

登录类型：一般或匿名



## vsftpd 配置文件详解

### (一) 默认配置:

1. 允许匿名用户和本地用户登陆。

anonymous\_enable=YES

local\_enable=YES

2. 匿名用户使用的登陆名为 ftp 或 anonymous, 口令为空; 匿名用户不能离开匿名用户家目录/var/ftp, 且只能下载不能上传。

3. 本地用户的登录名为本地用户名, 口令为此本地用户的口令; 本地用户可以在自己家目录中进行读写操作; 本地用户可以离开自家目录切换至有权限访问的其他目录, 并在权限允许的情况下进行上传/下载。

write\_enable=YES

4. 写在文件/etc/vsftpd.ftpusers 中的本地用户禁止登陆。

### (二) 配置文件格式:

vsftpd.conf 的内容非常单纯, 每一行即为一项设定。若是空白行或是开头为#的一行, 将会被忽略。内容的格式只有一种, 如下所示

option=value

要注意的是, 等号两边不能加空白。

### (三) 匿名用户 (anonymous) 设置:

**anonymous\_enable=YES/NO (YES)**

控制是否允许匿名用户登入, YES 为允许匿名登入, NO 为不允许。默认值为 YES。

**write\_enable=YES/NO (YES)**

是否允许登陆用户有写权限。属于全局设置, 默认值为 YES。

**no\_anon\_password=YES/NO (NO)**

若是启动这项功能, 则使用匿名登入时, 不会询问密码。默认值为 NO。

**ftp\_username=ftp**

定义匿名登入的使用者名称。默认值为 ftp。

**anon\_root=/var/ftp**

使用匿名登入时, 所登入的目录。默认值为 /var/ftp。注意 ftp 目录不能是 777 的权限属性, 即匿名用户的家目录不能有 777 的权限。

**anon\_upload\_enable=YES/NO (NO)**

如果设为 YES, 则允许匿名登入者有上传文件(非目录)的权限, 只有在 write\_enable=YES 时, 此项才有效。当然, 匿名用户必须要有对上层目录的写入权。默认值为 NO。

**anon\_world\_readable\_only=YES/NO (YES)**

如果设为 YES, 则允许匿名登入者下载可阅读的档案(可以下载到本机阅读, 不能直接在 FTP 服务器

中打开阅读）。默认值为 YES。

**anon\_mkdir\_write\_enable=YES/NO (NO)**

如果设为 YES，则允许匿名登入者有新增目录的权限，只有在 write\_enable=YES 时，此项才有效。当然，匿名用户必须要有对上层目录的写入权。默认值为 NO。

**anon\_other\_write\_enable=YES/NO (NO)**

如果设为 YES，则允许匿名登入者更多于上传或者建立目录之外的权限，譬如删除或者重命名。（如果 anon\_upload\_enable=NO，则匿名用户不能上传文件，但可以删除或者重命名已经存在的文件；如果 anon\_mkdir\_write\_enable=NO，则匿名用户不能上传或者新建文件夹，但可以删除或者重命名已经存在的文件夹。）默认值为 NO。

**chown\_uploads=YES/NO (NO)**

设置是否改变匿名用户上传文件（非目录）的属主。默认值为 NO。

**chown\_username=username**

设置匿名用户上传文件（非目录）的属主名。建议不要设置为 root。

**anon\_umask=077**

设置匿名登入者新增或上传档案时的 umask 值。默认值为 077，则新建档案的对应权限为 700。

**deny\_email\_enable=YES/NO (NO)**

若是启动这项功能，则必须提供一个档案/etc/vsftpd/banner\_emails，内容为 email address。若是使用匿名登入，则会要求输入 email address，若输入的 email address 在此档案内，则不允许进入。默认值为 NO。

**banned\_email\_file=/etc/vsftpd/banner\_emails**

此文件用来输入 email address，只有在 deny\_email\_enable=YES 时，才会使用到此档案。若是使用匿名登入，则会要求输入 email address，若输入的 email address 在此档案内，则不允许进入。

## (四) 本地用户设置：

**local\_enable=YES/NO (YES)**

控制是否允许本地用户登入，YES 为允许本地用户登入，NO 为不允许。默认值为 YES。

**local\_root=/home/username**

当本地用户登入时，将被更换到定义的目录下。默认值为各用户的家目录。

**write\_enable=YES/NO (YES)**

是否允许登陆用户有写权限。属于全局设置，默认值为 YES。

**local\_umask=022**

本地用户新增档案时的 umask 值。默认值为 077。

**file\_open\_mode=0755**

本地用户上传档案后的档案权限，与 chmod 所使用的数值相同。默认值为 0666。

## (五) 欢迎语设置：

**dirmessage\_enable=YES/NO (YES)**

如果启动这个选项，那么使用者第一次进入一个目录时，会检查该目录下是否有.message 这个档案，如果有，则会出现此档案的内容，通常这个档案会放置欢迎话语，或是对该目录的说明。默认值为开启。

**message\_file=.message**



设置目录消息文件，可将要显示的信息写入该文件。默认值为 message。

**banner\_file=/etc/vsftpd/banner**

当使用者登入时，会显示此设定所在的档案内容，通常为欢迎话语或是说明。默认值为无。如果欢迎信息较多，则使用该配置项。

**ftpd\_banner=Welcome to BOB's FTP server**

这里用来定义欢迎话语的字符串，banner\_file 是档案的形式，而 ftpd\_banner 则是字符串的形式。预设为无。

## (六) 控制用户是否允许切换到上级目录：

在默认配置下，本地用户登入 FTP 后可以使用 cd 命令切换到其他目录，这样会对系统带来安全隐患。可以通过以下三条配置文件来控制用户切换目录。

**chroot\_list\_enable=YES/NO (NO)**

设置是否启用 chroot\_list\_file 配置项指定的用户列表文件。默认值为 NO。

**chroot\_list\_file=/etc/vsftpd.chroot\_list**

用于指定用户列表文件，该文件用于控制哪些用户可以切换到用户家目录的上级目录。

**chroot\_local\_user=YES/NO (NO)**

用于指定用户列表文件中的用户是否允许切换到上级目录。默认值为 NO。

**通过搭配能实现以下几种效果：**

- ①当 chroot\_list\_enable=YES, chroot\_local\_user=YES 时，在/etc/vsftpd.chroot\_list 文件中列出的用户，可以切换到其他目录；未在文件中列出的用户，不能切换到其他目录。
- ②当 chroot\_list\_enable=YES, chroot\_local\_user=NO 时，在/etc/vsftpd.chroot\_list 文件中列出的用户，不能切换到其他目录；未在文件中列出的用户，可以切换到其他目录。
- ③当 chroot\_list\_enable=NO, chroot\_local\_user=YES 时，所有的用户均不能切换到其他目录。
- ④当 chroot\_list\_enable=NO, chroot\_local\_user=NO 时，所有的用户均可以切换到其他目录。

## (七) 数据传输模式设置：

FTP 在传输数据时，可以使用二进制方式，也可以使用 ASCII 模式来上传或下载数据。

**ascii\_upload\_enable=YES/NO (NO)**

设置是否启用 ASCII 模式上传数据。默认值为 NO。

**ascii\_download\_enable=YES/NO (NO)**

设置是否启用 ASCII 模式下载数据。默认值为 NO。

## (八) 访问控制设置：

两种控制方式：一种控制主机访问，另一种控制用户访问。

**①控制主机访问：**

**tcp\_wrappers=YES/NO (YES)**

设置 vsftpd 是否与 tcp wrapper 相结合来进行主机的访问控制。默认值为 YES。如果启用，则 vsftpd 服务器会检查/etc/hosts.allow 和/etc/hosts.deny 中的设置，来决定请求连接的主机，是否允许访问该 FTP 服务器。这两个文件可以起到简易的防火墙功能。

比如：若要仅允许 192.168.0.1—192.168.0.254 的用户可以连接 FTP 服务器，则在/etc/hosts.allow



文件中添加以下内容：

```
vsftpd:192.168.0. :allow
```

```
all:all :deny
```

## ②控制用户访问：

对于用户的访问控制可以通过/etc 目录下的 vsftpd.user\_list 和 ftpusers 文件来实现。

```
userlist_file=/etc/vsftpd.user_list
```

控制用户访问 FTP 的文件，里面写着用户名。一个用户名一行。

```
userlist_enable=YES/NO (NO)
```

是否启用 vsftpd.user\_list 文件。

```
userlist_deny=YES/NO (YES)
```

决定 vsftpd.user\_list 文件中的用户是否能够访问 FTP 服务器。若设置为 YES，则 vsftpd.user\_list 文件中的用户不允许访问 FTP，若设置为 NO，则只有 vsftpd.user\_list 文件中的用户才能访问 FTP。  
/etc/vsftpd/ftpusers 文件专门用于定义不允许访问 FTP 服务器的用户列表（注意：如果 userlist\_enable=YES, userlist\_deny=NO, 此时如果在 vsftpd.user\_list 和 ftpusers 中都有某个用户时，那么这个用户是不能够访问 FTP 的，即 ftpusers 的优先级要高）。默认情况下 vsftpd.user\_list 和 ftpusers，这两个文件已经预设置了一些不允许访问 FTP 服务器的系统内部账户。如果系统没有这两个文件，那么新建这两个文件，将用户添加进去即可。

## (九) 访问速率设置：

```
anon_max_rate=0
```

设置匿名登入者使用的最大传输速度，单位为 B/s，0 表示不限制速度。默认值为 0。

```
local_max_rate=0
```

本地用户使用的最大传输速度，单位为 B/s，0 表示不限制速度。预设值为 0。

## (十) 超时时间设置：

```
accept_timeout=60
```

设置建立 FTP 连接的超时时间，单位为秒。默认值为 60。

```
connect_timeout=60
```

PORT 方式下建立数据连接的超时时间，单位为秒。默认值为 60。

```
data_connection_timeout=120
```

设置建立 FTP 数据连接的超时时间，单位为秒。默认值为 120。

```
idle_session_timeout=300
```

设置多长时间不对 FTP 服务器进行任何操作，则断开该 FTP 连接，单位为秒。默认值为 300。

## (十一) 日志文件设置：

```
xferlog_enable= YES/NO (YES)
```

是否启用上传/下载日志记录。如果启用，则上传与下载的信息将被完整纪录在 xferlog\_file 所定义的档案中。预设为开启。

```
xferlog_file=/var/log/vsftpd.log
```

设置日志文件名和路径，默认值为 /var/log/vsftpd.log。



**xferlog\_std\_format=YES/NO (NO)**

如果启用，则日志文件将会写成 xferlog 的标准格式，如同 wu-ftpd 一般。默认值为关闭。

**log\_ftp\_protocol=YES|NO (NO)**

如果启用此选项，所有的 FTP 请求和响应都会被记录到日志中，默认日志文件在 /var/log/vsftpd.log。

启用此选项时，xferlog\_std\_format 不能被激活。这个选项有助于调试。默认值为 NO。

## (十二) 定义用户配置文件：

在 vsftpd 中，可以通过定义用户配置文件来实现不同的用户使用不同的配置。

**user\_config\_dir=/etc/vsftpd/userconf**

设置用户配置文件所在的目录。当设置了该配置项后，用户登陆服务器后，系统就会到 /etc/vsftpd/userconf 目录下，读取与当前用户名相同的文件，并根据文件中的配置命令，对当前用户进行更进一步的配置。

例如：定义 user\_config\_dir=/etc/vsftpd/userconf，且主机上有使用者 test1, test2，那么我们就在 user\_config\_dir 的目录新增文件名为 test1 和 test2 两个文件。若是 test1 登入，则会读取 user\_config\_dir 下的 test1 这个档案内的设定。默认值为无。利用用户配置文件，可以实现对不同用户进行访问速度的控制，在各用户配置文件中定义 local\_max\_rate=XX，即可。

## (十三) FTP 的工作方式与端口设置：

FTP 有两种工作方式：PORT FTP（主动模式）和 PASV FTP（被动模式）

**listen\_port=21**

设置 FTP 服务器建立连接所监听的端口，默认值为 21。

**connect\_from\_port\_20=YES/NO**

指定 FTP 使用 20 端口进行数据传输，默认值为 YES。

**ftp\_data\_port=20**

设置在 PORT 方式下，FTP 数据连接使用的端口，默认值为 20。

**pasv\_enable=YES/NO (YES)**

若设置为 YES，则使用 PASV 工作模式；若设置为 NO，则使用 PORT 模式。默认值为 YES，即使用 PASV 工作模式。

**pasv\_max\_port=0**

在 PASV 工作模式下，数据连接可以使用的端口范围的最大端口，0 表示任意端口。默认值为 0。

**pasv\_min\_port=0**

在 PASV 工作模式下，数据连接可以使用的端口范围的最小端口，0 表示任意端口。默认值为 0。

## (十四) 与连接相关的设置：

**listen=YES/NO (YES)**

设置 vsftpd 服务器是否以 standalone 模式运行。以 standalone 模式运行是一种较好的方式，此时 listen 必须设置为 YES，此为默认值。建议不要更改，有很多与服务器运行相关的配置命令，需要在此模式下才有效。若设置为 NO，则 vsftpd 不是以独立的服务运行，要受到 xinetd 服务的管控，功能上会受到限制。

**max\_clients=0**

设置 vsftpd 允许的最大连接数，默认值为 0，表示不受限制。若设置为 100 时，则同时允许有 100 个连接，超出的将被拒绝。只有在 standalone 模式运行才有效。

**max\_per\_ip=0**

设置每个 IP 允许与 FTP 服务器同时建立连接的数目。默认值为 0，表示不受限制。只有在 standalone 模式运行才有效。

**listen\_address=IP 地址**

设置 FTP 服务器在指定的 IP 地址上侦听用户的 FTP 请求。若不设置，则对服务器绑定的所有 IP 地址进行侦听。只有在 standalone 模式运行才有效。

**setproctitle\_enable=YES/NO (NO)**

设置每个与 FTP 服务器的连接，是否以不同的进程表现出来。默认值为 NO，此时使用 ps aux | grep ftp 只会有一个 vsftpd 的进程。若设置为 YES，则每个连接都会有一个 vsftpd 的进程。

## (十五) 虚拟用户设置：

虚拟用户使用 PAM 认证方式。

**pam\_service\_name=vsftpd**

设置 PAM 使用的名称，默认值为 /etc/pam.d/vsftpd。

**guest\_enable= YES/NO (NO)**

启用虚拟用户。默认值为 NO。

**guest\_username=ftp**

这里用来映射虚拟用户。默认值为 ftp。

**virtual\_use\_local\_privs=YES/NO (NO)**

当该参数激活 (YES) 时，虚拟用户使用与本地用户相同的权限。当此参数关闭 (NO) 时，虚拟用户使用与匿名用户相同的权限。默认情况下此参数是关闭的 (NO)。

## (十六) 其他设置：

**text\_userdb\_names= YES/NO (NO)**

设置在执行 ls -la 之类的命令时，是显示 UID、GID 还是显示出具体的用户名和组名。默认值为 NO，即以 UID 和 GID 方式显示。若希望显示用户名和组名，则设置为 YES。

**ls\_recurse\_enable=YES/NO (NO)**

若是启用此功能，则允许登入者使用 ls -R (可以查看当前目录下子目录中的文件) 这个指令。默认值为 NO。

**hide\_ids=YES/NO (NO)**

如果启用此功能，所有档案的拥有者与群组都为 ftp，也就是使用者登入使用 ls -al 之类的指令，所看到的档案拥有者跟群组均为 ftp。默认值为关闭。

**download\_enable=YES/NO (YES)**

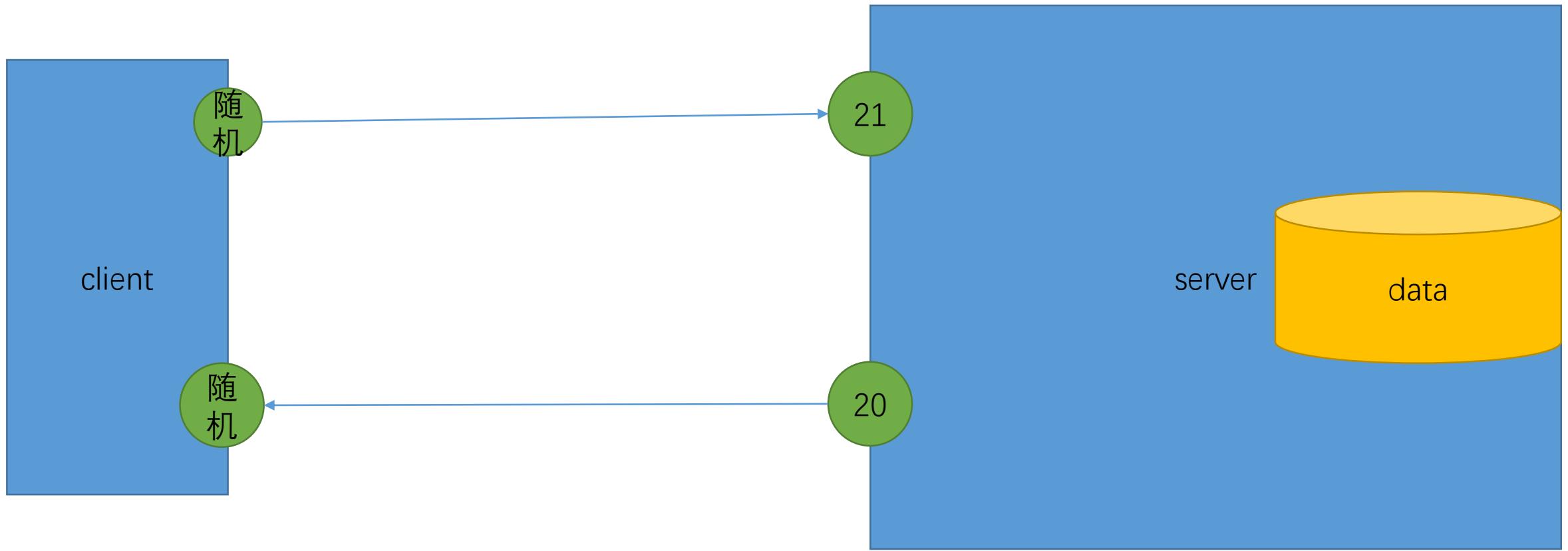
如果设置为 NO，所有的文件都不能下载到本地，文件夹不受影响。默认值为 YES。

## (十七) 响应代码解释说明：

110 新文件指示器上的重启标记

120 服务器准备就绪的时间（分钟数）

125 打开数据连接, 开始传输  
150 打开连接  
200 成功  
202 命令没有执行  
211 系统状态回复  
212 目录状态回复  
213 文件状态回复  
214 帮助信息回复  
215 系统类型回复  
220 服务就绪  
221 退出网络  
225 打开数据连接  
226 结束数据连接  
227 进入被动模式 (IP 地址、ID 端口)  
230 登录因特网  
250 文件行为完成  
257 路径名建立  
331 要求密码  
332 要求帐号  
350 文件行为暂停  
421 服务关闭  
425 无法打开数据连接  
426 结束连接  
450 文件不可用  
451 遇到本地错误  
452 磁盘空间不足  
500 无效命令  
501 错误参数  
502 命令没有执行  
503 错误指令序列  
504 无效命令参数  
530 未登录网络  
532 存储文件需要帐号  
550 文件不可用  
551 不知道的页类型  
552 超过存储分配  
553 文件名不允许



客户端

匿名登录

匿名账户:  
ftp、anonymous  
/var/ftp

服务器

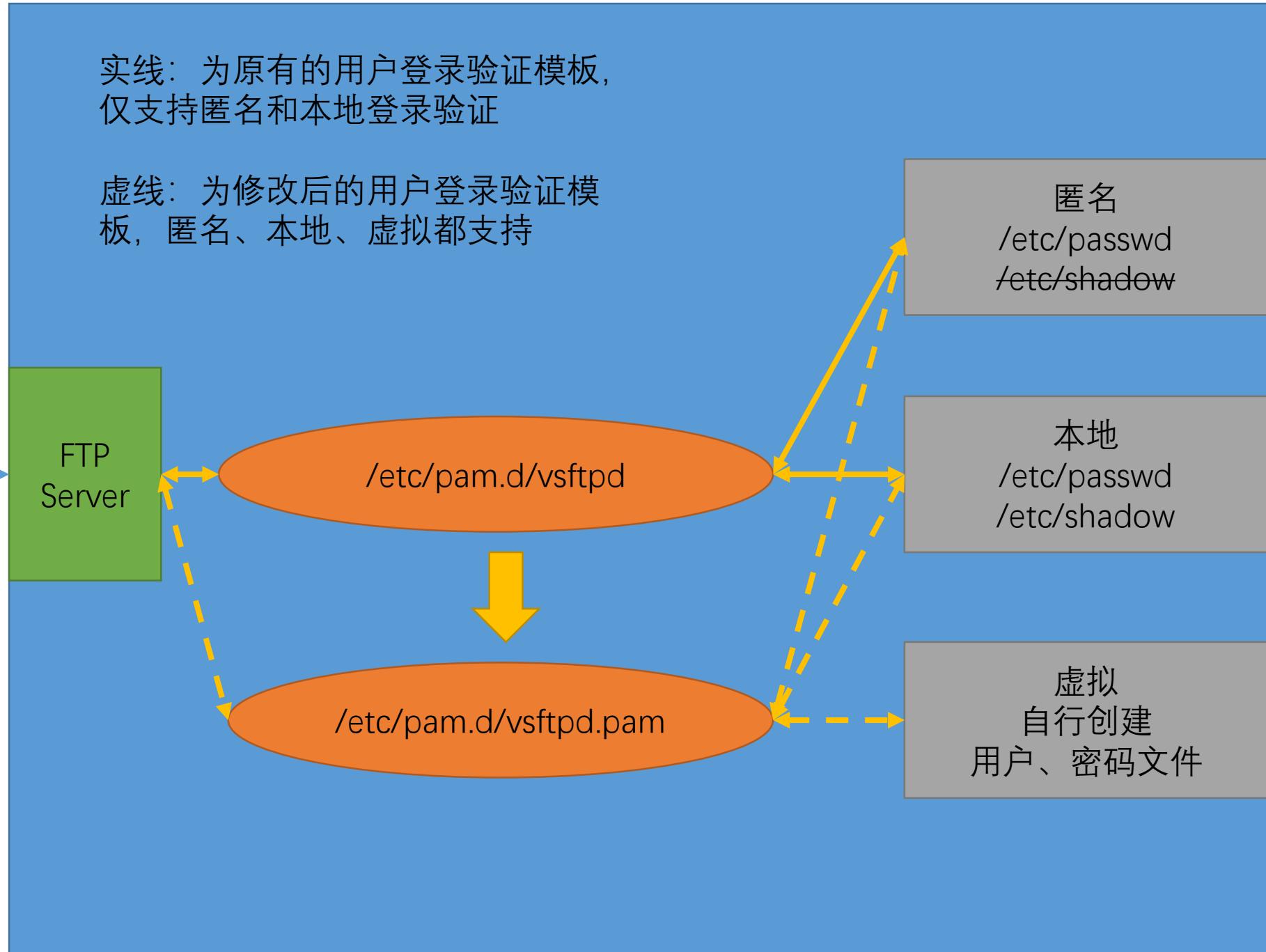
本地登录

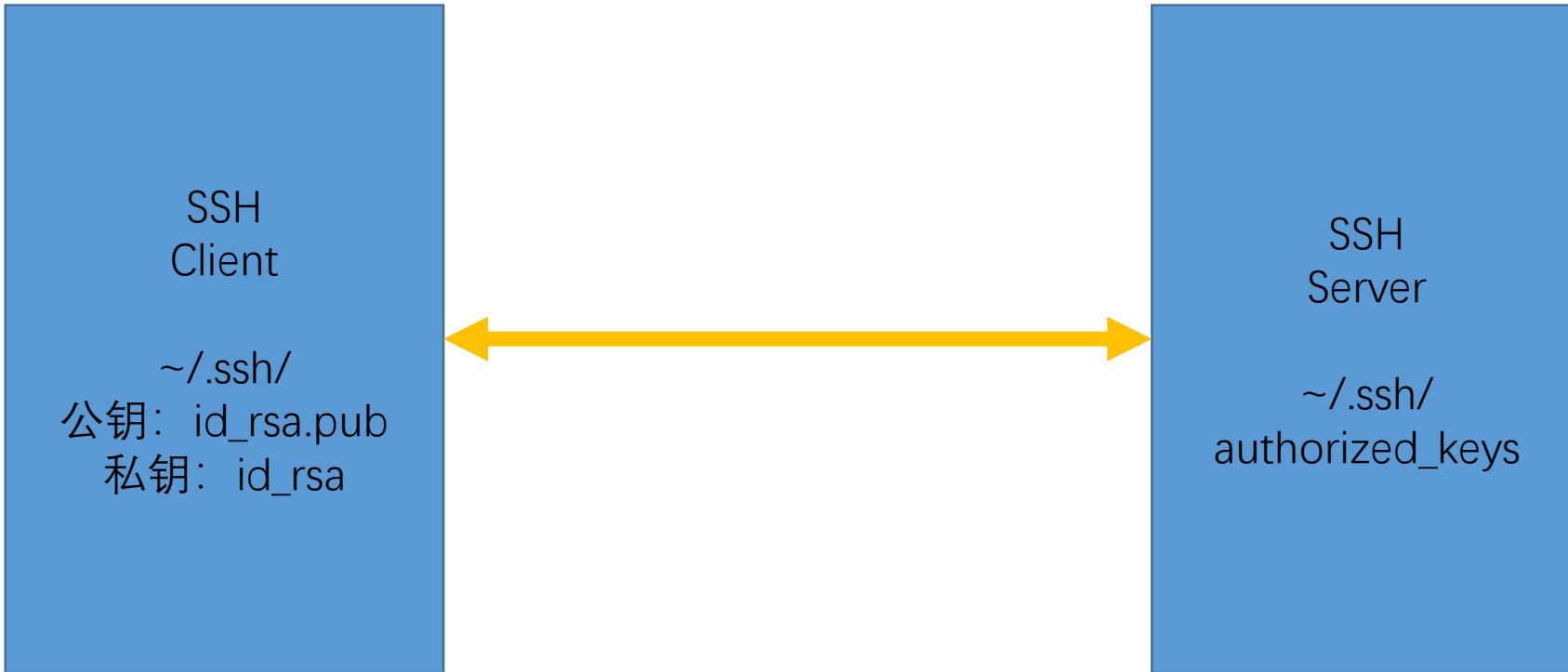
本地登录:  
/etc/passwd 普通用户  
/etc/shadow 用户的家目录

虚拟账户

虚拟账户:  
人为创建，生成数据库文件，找一个系统用  
户作为虚拟用户的映射用户，借助系统用  
户的家目录作为默认登录点  
默认登录目录：/home/\*\*\*

每一个虚拟账户的权限都可以单独定制



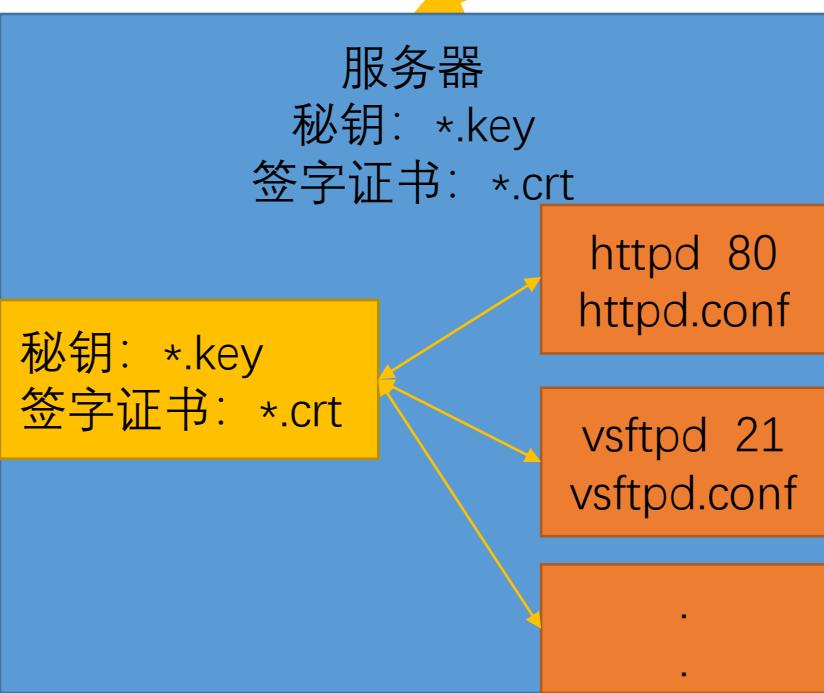


**注: sshd服务的配置文件中, 注释掉的是默认生效的**

1. 由客户端生成密钥对文件, 并将公钥文件上传到服务器的指定目录下, 修改指定名称: `ssh-keygen -t rsa -b 1024 & ssh-copy-id user@ip`
2. 需要在服务器端开启密钥对登录认证模式 (默认是开启的)

秘钥：解密加密的数据包  
证书：提供加密所需的功能{加密类型、加密长度}

因为数据包是从客户端发向服务器端的，所以需要将证书给客户端使用，客户端使用证书将要发送的数据包加密，然后再进行传输  
服务器端接收到客户端发送的加密数据包后，拿着秘钥文件进行解析，得到明文



CA 证书服务器  
秘钥: \*.key  
证书: \*.csr  
签字后的证书: \*.crt

客户端所使用的证书是，CA 服务器签名后的证书。

签名后的证书：  
1. 有一定有效期  
2. 有加密类型  
3. 有加密长度  
.....

客户端  
证书: \*.crt

# 网络服务—RSYNC

## 1. rsync 概述

rsync 是类 unix 系统下的数据镜像备份工具。一款支持快速完全备份和增量备份的工具，支持本地复制，远程同步等，类似于 scp 命令；rsync 命令在同步文件之前要先登录目标主机进行用户身份认证，认证过后才能进行数据同步，身份认证方式取决于所使用的协议类型，rsync 一般使用两种协议进行数据同步：ssh 协议和 rsync 协议。

## 2. rsync 特性

- 能更新整个目录树和文件系统
- 有选择性的保留符号链接、硬链接、文件属性、权限、设备以及时间等
- 对于安装来说，无任何特殊权限要求
- 对于多个文件来说，文件传输效率高
- 能用 ssh 或自定义端口作为传输入口端口

## 3. rsync 工作原理

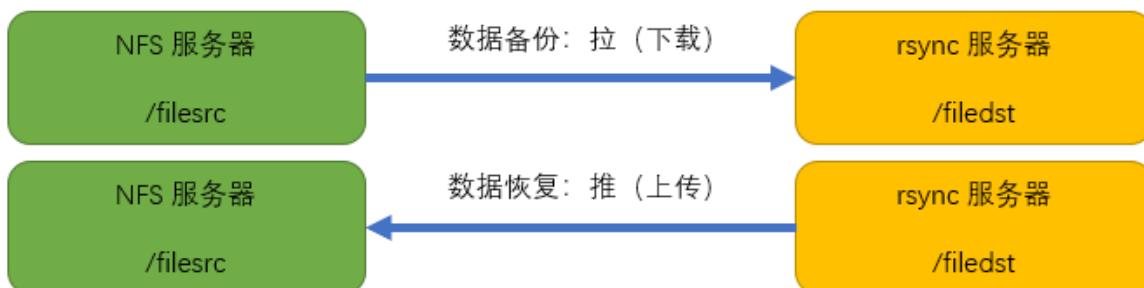
既然涉及到数据同步，必要的两个概念是：源地址（文件），目标地址（文件），以及以哪一方为基准，例如，想让目标主机上的文件和本地文件保持同步，则是以本地文件为同步基准，将本地文件作为源文件推送到目标主机上。

rsync 在进行数据同步之前需要先进行用户身份验证，验证方式取决于使用的连接方式：

**ssh 登录验证模式：** 使用 ssh 协议作为基础进行用户身份认证，然后进行数据同步。

**rsync 登录验证模式：** 使用 rsync 协议进行用户身份认证（非系统用户），然后进行数据同步。

**数据同步方式：** 推送（上传）、拉取（下载）



## 4. rsync 实验演示

我们一般使用 rsync 来进行单向数据同步，因此我们需要确定一个基准，比如：两台服务器，一台 NFS 作为网站数据服务器（基准服务器），另外一台专门做 rsync 数据备份服务器，我们以此为基础开始我们的实验。

### 4.1 ssh 协议数据同步：将 NFS 服务器数据同步备份到 rsync 服务器

实验环境：一台 NFS 服务器，一台 rsync 服务器

在两台服务器上分别创建目录（/filesrvc、/filedst）

#### 下行同步（下载）

格式：rsync -avz 服务器地址:/服务器目录/\* /本地目录

示例：rsync -avz root@192.168.88.10:/filesrvc/\* /filedst

-a：归档模式，递归并保留对象属性

-v：显示同步过程

-z：在传输文件时进行压缩

#### 上行同步（上传）

格式：rsync -avz /本地目录/\* 服务器地址:/服务器目录

示例：rsync -avz /filedst/\* root@192.168.88.10:/filesrvc

**注意：**使用 root 用户进行实验可以，但生产环境中尽量使用单独创建的普通用户，减少权限溢出  
（创建用来做数据同步的用户，并给予用户对目录的相应权限，一般使用 ACL 设置权限）

```
useradd zhangsan
passwd zhangsan
setfacl -m u:zhangsan:rwx /filesrvc
```

**拓展：**若要实现免密码数据同步，只需要做好 ssh 密钥对登录即可

### 4.2 rsync 协议数据同步：将 NFS 服务器数据同步备份到 rsync 服务器

实验环境：一台服务器，一台客户端

1. 在两台服务器上分别创建目录（/filesrvc、/filedst）

2. 搭建 rsync 服务（仅需要在 NFS 服务器上搭建即可）

a. 创建主配置文件（/etc/rsyncd.conf）

```
address = 192.168.88.10 #rsync 服务绑定 IP
port 873 #默认服务端口 873
log file = /var/log/rsyncd.log #日志文件位置
pid file = /var/run/rsyncd.pid #进程号文件位置
[web] #共享名：用来连接是写在 url 上的，切记
 comment = web directory backup #共享描述话语
 path = /filesrvc #实际共享目录
 read only = no #是否仅允许读取
 dont compress = *.gz *.bz2 #哪些文件类型不进行压缩
```

```
auth users = user1 #登录用户名（非系统用户，需要自行创建）
secrets file = /etc/rsyncd_users.db #认证所需账户密码文件（需自行创建-同上）
```

b. 创建认证所需账户密码文件

```
vim /etc/rsyncd_users.db
user1:123456
chmod 600 /etc/rsyncd_users.db 必须修改权限，否则登录报错
```

c. 启动服务

```
rsync --daemon
netstat -antp | grep :873
```

d. 设置映射用户对共享目录有权限 (r)

```
setfacl -m u:nobody:rxw /filesrc
```

注意：关闭服务可使用 kill 命令，但偶尔会造成服务被结束，但进程号配置文件不被删除的问题，若遇到此类问题可自己手动删除，再启动则正常（建议自己写一个 rsync 的服务管理脚本）

### 下行同步（下载）

格式：rsync -avz rsync://用户名@服务器地址/共享模块名 /本地目录

示例：rsync -avz rsync://user1@192.168.88.10/web /filedst

拓展：--delete：删除本地比服务器多出来的文件（源地址没有，目标地址有的删掉）

```
rsync -avz --delete rsync://user1@192.168.88.10/web /filedst
```

### 上行同步（上传）

格式：rsync -avz /本地目录/\* rsync://用户名@服务器地址/共享模块名

示例：rsync -avz /filedst/\* rsync://user1@192.168.88.10/web

拓展：rsync 协议的免密码可以借助一个环境变量实现

```
export RSYNC_PASSWORD=虚拟用户密码（客户端生成）
```

## 5. 配置 rsync+inotify 实时同步

定期同步的缺点：

执行备份的时间固定，延期明显，实时性差

当同步源长期不变化时，密集的定期任务是不必要的（浪费资源）

实时同步的优点：

一旦同步源出现变化，立即启动备份，实时性好

只要同步源无变化，则不执行备份，节省资源

### inotify 简介

inotify 是一个 Linux 内核特性，它监控文件系统，并且及时向专门的应用程序发出相关的事件警告，比如删除、读、写和卸载操作等。要使用 inotify，必须具备一台带有 2.6.13 版本的内核操作系统。

inotify 两个监控命令：

inotifywait: 用于持续监控，实时输出结果（常用）

inotifywatch: 用于短期监控，任务完成后再出结果

### inotify 部署

```
yum -y install gcc*
tar -xf inotify-tools-3.14.tar.gz
cd inotify-tools-3.14
./configure && make && make install
```

### inotifywait 命令格式

格式: inotifywait -mrq -e 监控动作 1, 监控动作 2 /监控目录 &

示例: inotifywait -mrq -e create,delete /filesrc &

-m: 始终保持事件监听状态

-r: 递归查询目录

-q: 只打印监控事件的信息

监控动作: modify (内容), create, attrib (权限), move, delete

### 利用 rsync+inotifywait 结合脚本实现单向实时同步

```
vim src.sh
#!/bin/bash
a="inotifywait -mrq -e create,delete /filesrc"
b="rsync -avz /filesrc/* root@192.168.88.20:/filedst"
$a | while read directory event file #while 判断是否接收到监控记录
do
 $b
done
```

注: 用户登录时要求免密码验证

### 实验结果验证

在服务器端创建, 删除文件, 查看备份端是否正常

拓展: 调整 inotify 监控的文件数量

调整 inotify 内核参数 (/etc/sysctl.conf)

|                    |             |
|--------------------|-------------|
| max_queue_events   | 监控队列大小      |
| max_user_instances | 最多监控实例数     |
| max_user_watches   | 每个实例最多监控文件数 |

## 6. 配置 unison+inotify 实现双向实时同步

rsync 在单向同步上支持的非常好, 且效率很高, 但是在双向同步支持较差; unison 则是双向同步的优秀工具, 但其缺点是同步效率较低。

### 1. 环境要求

1) 准备好同步所需的两个目录



2) 如若用 root 来实现登录的话，生成密钥对，以便于免密码验证

3) 准备好 inotify 和 unison 的软件包

## 2. 安装步骤

1) 先安装 inotify

```
tar -xf inotify-tools-3.14.tar.gz
cd inotify-tools-3.14
./configure && make && make install
```

2) 再安装 ocaml

```
tar -xf ocaml-3.10.1.tar.gz
cd ocaml-3.10.1
./configure #忽略所有报错
make world opt
make install
```

3) 安装 unison

```
tar -xf unison-2.13.16.tar.gz
cd unison-2.13.16
make UISTYLE=text THREADS=true STATIC=true #已经存在Makefile文件,不需要./configure
cp unison /usr/local/bin/ #把生成的脚本拷贝出来
```

**注意：同样的操作在服务器端也做一遍。**

## 3. 配置脚本

**注：双向自动同步，监控目录和数据同步时，源目录不能使用\*通配符传输，否则会变成死循环。**

**filesrc 端：**

```
#!/bin/bash
a="inotifywait -mrq -e create,delete /filesrc"
b="/usr/local/bin/unison -batch /filesrc/ ssh://192.168.88.20//filedst/" #batch: 批处理
$a | while read directory event file
do
 $b
done
```

**filedst 端：**

```
#!/bin/bash
a="inotifywait -mrq -e create,delete /filedst"
b="/usr/local/bin/unison -batch /filedst/ ssh://192.168.88.10//filesr/" #batch: 批处理
$a | while read directory event file
do
 $b
done
```

## 4. 测试

讲两个脚本放入后台执行： bash src.sh &

分别在两个主机上创建文件查看是否可以实现双向实时同步（可能会有延迟）

# ELK 日志分析

## 1. 为什么用到 ELK

一般我们需要进行日志分析场景：直接在日志文件中 grep、awk 就可以获得自己想要的信息。但在规模较大的场景中，此方法效率低下，面临问题包括日志量太大如何归档、文本搜索太慢怎么办、如何多维度查询。需要集中化的日志管理，所有服务器上的日志收集汇总。常见解决思路是建立集中式日志收集系统，将所有节点上的日志统一收集，管理，访问。

一般大型系统是一个分布式部署的架构，不同的服务模块部署在不同的服务器上，问题出现时，大部分情况需要根据问题暴露的关键信息，定位到具体的服务器和服务模块，构建一套集中式日志系统，可以提高定位问题的效率。

一个完整的集中式日志系统，需要包含以下几个主要特点：

收集—能够采集多种来源的日志数据  
传输—能够稳定的把日志数据传输到中央系统  
存储—如何存储日志数据  
分析—可以支持 UI 分析

警告—能够提供错误报告，监控机制 ELK 提供了一整套解决方案，并且都是开源软件，之间互相配合使用，完美衔接，高效的满足了很多场合的应用。目前主流的一种日志系统。

## 2. ELK 简介

ELK 是三个开源软件的缩写，分别表示：Elasticsearch, Logstash, Kibana，它们都是开源软件。新增了一个 FileBeat，它是一个轻量级的日志收集处理工具(Agent)，Filebeat 占用资源少，适合于在各个服务器上搜集日志后传输给 Logstash，官方也推荐此工具。

Elasticsearch 是个开源分布式搜索引擎，提供搜集、分析、存储数据三大功能。它的特点有：分布式，零配置，自动发现，索引自动分片，索引副本机制，restful 风格接口，多数据源，自动搜索负载等。

Logstash 主要是用来日志的搜集、分析、过滤日志的工具，支持大量的数据获取方式。一般工作方式为 c/s 架构，client 端安装在需要收集日志的主机上，server 端负责将收到的各节点日志进行过滤、修改等操作在一并发往 Elasticsearch 上去。

Kibana 也是一个开源和免费的工具，Kibana 可以为 Logstash 和 Elasticsearch 提供的日志分析友好的 Web 界面，可以帮助汇总、分析和搜索重要数据日志。

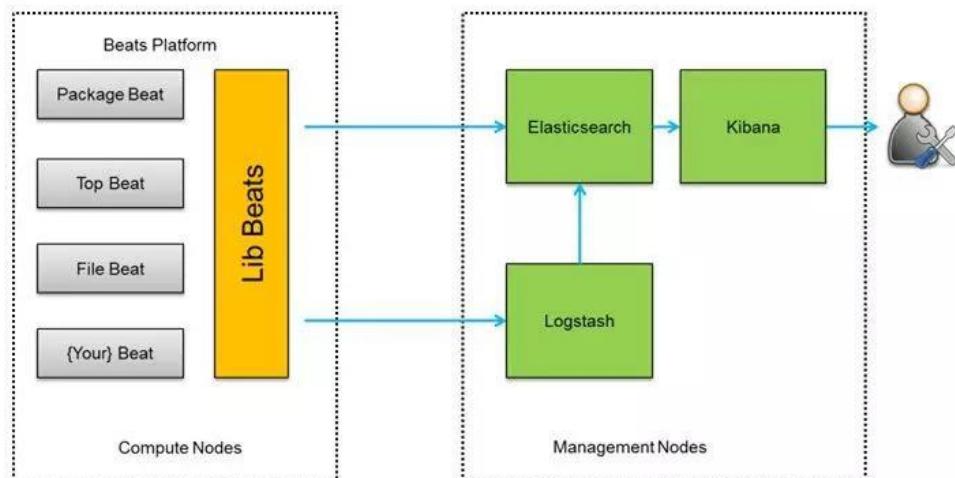
Filebeat 隶属于 Beats。目前 Beats 包含四种工具：

Packetbeat (搜集网络流量数据)

Topbeat (搜集系统、进程和文件系统级别的 CPU 和内存使用情况等数据)

Filebeat (搜集文件数据)

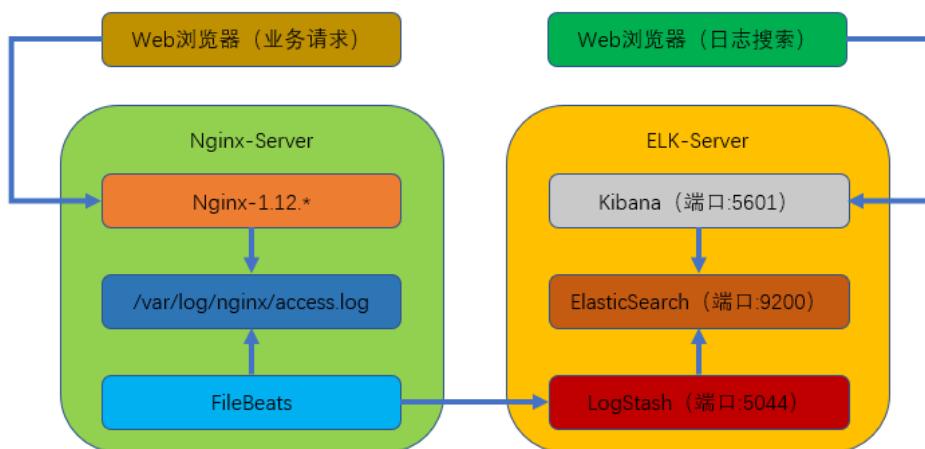
Winlogbeat (搜集 Windows 事件日志数据)



### 3. 实验部署

本次部署的是 filebeats(客户端), logstash+elasticsearch+kibana(服务端)组成的架构。

业务请求到达 nginx-server 机器上的 Nginx; Nginx 响应请求，并在 access.log 文件中增加访问记录; FileBeat 搜集新增的日志，通过 LogStash 的 5044 端口上传日志; LogStash 将日志信息通过本机的 9200 端口传入到 ElasticSerach; 搜索日志的用户通过浏览器访问 Kibana，服务器端口是 5601; Kibana 通过 9200 端口访问 ElasticSerach;



#### 实验环境:

本次部署的是单点 ELK 用了两台机器(CentOS-7.5)

ELK 服务端: 192.168.88.100

Nginx 客户端: 192.168.88.110

## 1. 准备工作:

配置好网络 yum 源

```
wget http://mirrors.aliyun.com/repo/Centos-7.repo
```

```
wget http://mirrors.aliyun.com/repo/epel-7.repo
```

关闭防火墙: systemctl stop(disable) firewalld

关闭 SELinux: SELINUX=disabled

## 2. 下载并安装软件包:

```
mkdir /elk;cd /elk
```

```
wget https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-6.2.3.tar.gz
```

```
wget https://artifacts.elastic.co/downloads/logstash/logstash-6.2.3.tar.gz
```

```
wget https://artifacts.elastic.co/downloads/kibana/kibana-6.2.3-linux-x86_64.tar.gz
```

全部解压缩，并复制到/usr/local/目录下

## 3. 安装 JDK(java)环境工具:

```
yum -y install java-1.8*
```

## 4. 配置 elasticsearch:

1) 新建 elasticsearch 用户并启动(用 elasticsearch 普通用户启动)

```
useradd elasticsearch
```

```
chown -R elasticsearch.elasticsearch /usr/local/elasticsearch-6.2.3/
```

```
su - elasticsearch
```

```
cd /usr/local/elasticsearch-6.2.3/
```

```
./bin/elasticsearch -d
```

2) 查看进程是否启动成功（等待一下）

```
netstat -antp
```

3) 若出现错误可以查看日志

```
cat /usr/local/elasticsearch-6.2.3/logs/elasticsearch.log
```

4) 测试是否可以正常访问

```
curl localhost:9200
```

## 5. 配置 logstash

Logstash 收集 nginx 日志之使用 grok 过滤插件解析日志，grok 作为一个 logstash 的过滤插件，支持根据模式解析文本日志行，拆成字段。

1) logstash 中 grok 的正则匹配

```
vim vendor/bundle/jruby/2.3.0/gems/logstash-patterns-core-4.1.2/patterns/grok-patterns
```

```
WZ ([^]*)
```

```
NGINXACCESS %{IP:remote_ip} \- \- [%{HTTPDATE:timestamp}\] "%{WORD:method} %{WZ:request}
HTTP/%{NUMBER:httpversion}" %{NUMBER:status} %{NUMBER:bytes} %{QS:referer} %{QS:agent}
%{QS:xforward}
```

2) 创建 logstash 配置文件

```
vim /usr/local/logstash-6.2.3/default.conf
```

```
input {
```

```
 beats {
```



```
port => "5044"
}

}

#数据过滤
filter {
 grok {
 match => { "message" => "%{NGINXACCESS}" }
 }
 geoip {
 # nginx 客户端 ip
 source => "192.168.88.110"
 }
}

#输出配置为本机的 9200 端口，这是 ElasticSearch 服务的监听端口
output {
 elasticsearch {
 hosts => ["127.0.0.1:9200"]
 }
}
```

3) 进入到/usr/local/logstash-6.2.3 目录下，并执行下列命令

后台启动 logstash: nohup bin/logstash -f default.conf &

查看启动日志: tailf nohup.out

查看端口是否启动: netstat -napt|grep 5044

## 6. 配置 kibana

1) 打开 Kibana 配置文件/usr/local/kibana-6.2.3-linux-x86\_64/config/kibana.yml，找到下面这行并修改

```
vim /usr/local/kibana-6.2.3-linux-x86_64/config/kibana.yml
```

#server.host: "localhost"

修改为

```
server.host: "192.168.88.100"
```

这样其他电脑就能用浏览器访问 Kibana 的服务了；

2) 进入 Kibana 的目录: cd /usr/local/kibana-6.2.3-linux-x86\_64

执行启动命令: nohup bin/kibana &

查看启动日志: tail -f nohup.out

查看端口是否启动: netstat -napt|grep 5601

3) 测试:

在浏览器访问 192.168.88.100:5601

至此。ELK 部署完成

## 7. Nginx 客户端配置

1) yum 安装二进制 nginx 软件包

```
yum -y install nginx
```

2) 下载 filebeat 并解压到/usr/local/

```
wget https://artifacts.elastic.co/downloads/beats/filebeat/filebeat-6.2.3-linux-x86_64.tar.gz
```

```
tar -xf ./filebeat-6.2.3-linux-x86_64.tar.gz -C /usr/local/
```

3) 打开文件/usr/local/filebeat-6.2.3-linux-x86\_64/filebeat.yml，找到如下位置：修改三处

```
enable: false
```

#修改为 true

```
paths: /var/log/*.log
```

#修改为/var/log/nginx/\*.log

```
#output.elasticsearch:
```

#将此行注释掉

```
#hosts: ["localhost:9200"]
```

#将此行注释掉

```
output.logstash:
```

#取消此行注释

```
hosts: ["192.168.88.100:5044"] #取消此行注释并修改 IP 地址为 ELK 服务器地址
```

4) 切换到/usr/local/filebeat-6.2.3-linux-x86\_64 目录下

```
cd /usr/local/filebeat-6.2.3-linux-x86_64
```

后台启动 filebeat: nohup ./filebeat -e -c filebeat.yml &

查看日志: tailf nohup.out

5) 通过浏览器多访问几次 nginx 服务，这样能多制造一些访问日志，访问地址: <https://192.168.137.131>

6) 访问 Kibana: <https://192.168.88.100:5601>，点击左上角的 Discover，就可以看到访问日志已经被 ELK 搜集了，然后按照下列步骤完成设置

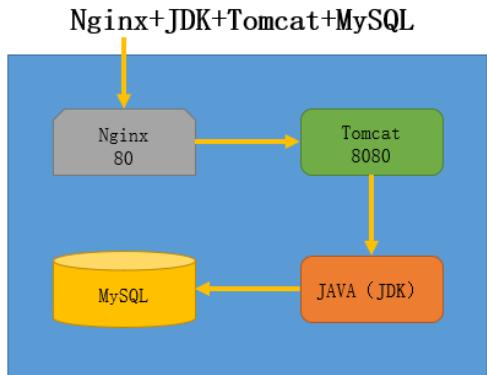
- 输入 logstash-\*，点击“Next step”
- 选择 Time Filter，再点击“Create index pattern”
- 然后可自行创建日志内容查询规则

# Java web 环境搭建

## 1. 初识 Tomcat

Tomcat 服务器是一个免费的开放源代码的 Web 应用服务器，属于轻级应用服务器，在中小型系统和并发访问用户不是很多的场合下被普遍使用，是开发和调试 JSP 程序的选择。Tomcat 是 Apache 服务器的扩展，但运行时它是独立运行的，所以当你运行 tomcat 时，它实际上作为一个与 Apache 独立的进程单独运行的。

## 2. Java web 环境：Nginx+JDK+Tomcat+MySQL



1. 所有服务部署在同一个主机上，也可分开部署
2. Nginx 默认开启的是 80 端口，用来接收用户的 web 请求
3. Tomcat 默认开启的是 8080 端口，用来接收 Nginx 转发过来的 web 请求

## 3. 环境部署流程

### 1. 安装 JDK (java 解析器)

#### 1、先安装 gcc

```
yum -y install gcc
```

#### 2、将软件包上传、解压、并移至指定位置

```
mv 解压缩目录 /usr/local/jdk1.7
```

#### 3、设置 JDK 的环境变量

```
vim /etc/profile #添加以下内容
```

```
export JAVA_HOME=/usr/local/jdk1.7
```

```
export JAVA_BIN=/usr/local/jdk1.7/bin
```

```
export PATH=$PATH:$JAVA_HOME/bin
```

```
export CLASSPATH=.:${JAVA_HOME}/lib/dt.jar:${JAVA_HOME}/lib/tools.jar
```

```
source /etc/profile
```

#### 4、查看 java 是否安装成功

```
java -version
```

## 2. 安装 tomcat

1、将软件包上传、解压、并复制到指定目录下

```
cp -a 解压目录 /usr/local/tomcat
```

2、置 Tomcat 的环境变

```
vim /etc/profile
export TOMCAT_HOME=/usr/local/tomcat
export PATH=$PATH:$TOMCAT_HOME/bin
source /etc/profile
```

3、将 tomcat 的启动脚本赋予执行权

```
chmod +x /usr/local/tomcat/bin/*
```

4、开启 tomcat

```
/usr/local/tomcat/bin/catalina.sh start
netstat -antp #查看端口，确认是否启动
```

5、在客户端访 192.168.10.20:8080 进行测试

## 3. 安装 MySQL 数据库

1、安装依赖包 ncurses-devel

```
yum -y install ncurses-devel gcc
```

2、将 mysql 文件进行传输到 192.168.10.10 上进行安装

```
useradd -r -s /sbin/nologin mysql
./configure --prefix=/usr/local/mysql --with-charset=utf8
--with-collation=utf8_general_ci --with-extra-charsets=gbk,gb2312
make
make install
```

3、生成 置文件

```
cp -a support-files/my-medium.cnf /etc/my.cnf
ln -s /usr/local/mysql/bin/* /usr/local/bin/
ln -s /usr/local/mysql/sbin/* /usr/local/sbin/
```

4、初始化数据库，生成授权表

```
cd /usr/local/mysql
./bin/mysql_install_db --user=mysql
```

5、生成启动管理脚本，启动 mysql 并设置开机自启

```
cd ~/mysql-5.1.55/support-files
cp -a mysql.server /etc/init.d/mysqld
chmod +x /etc/init.d/mysqld
chkconfig --add mysqld
chkconfig mysqld on
service mysqld start|stop|restart
```

6、为数据库的管理用户 root 设置登录密码

```
mysqladmin -uroot password 123456
```

7、登录数据库，查看是否安装正确



#### 4. 安装 nginx

##### 1、解压 nginx

```
tar -xf nginx-1.2.6.tar.gz
```

##### 2、安装 nginx 依赖包

```
yum -y install pcre-devel zlib-devel gcc
```

##### 3、添加用户

```
useradd -r -s /sbin/nologin nginx
```

##### 4、编译并安装

```
./configure --user=nginx --group=nginx
```

```
make
```

```
make install
```

##### 5、修改 nginx 配置文件

```
vim /usr/local/nginx/conf/nginx.conf
```

```
user nginx;
```

```
upstream tomcat { #添加负载调度（为了后期扩展更多 Tomcat 服务器方便）
```

```
 server 192.168.10.20:8080;
```

```
}
```

```
location / { #添加反向代理
```

```
 proxy_pass http://tomcat;
```

```
 proxy_set_header Host $host;
```

```
}
```

##### 6、启服务

```
pkill -HUP nginx
```

##### 7、在客户端进行测试，输入 nginx 地址，打开的为 tomcat 署的网站

## 4. 署 Java 的 WAR 包

##### 1、在 /usr/local/tomcat/conf/server.xml 配置文件中的 server 区域中添加标红内容

```
<Host name="localhost" appBase="webapps" unpackWARs="true" autoDeploy="true">
<Context path="" docBase="test.war" debug="0" privileged="true"/>
```

##### 2、将 war 包拷贝到 webapps 目录中

```
cp -a test.war /usr/local/tomcat/webapps/
```

##### 4、删 网站的 ROOT 目录

```
rm -rf ROOT
```

##### 4、启 tomcat 服务

```
/usr/local/tomcat/bin/catalina.sh stop
```

```
/usr/local/tomcat/bin/catalina.sh start
```

##### 5、测试

使用客户端浏览访 Nginx 服务的端口

注意：原本的 tomcat 使用 8080 端口进行访，也可以将端口修改为 80 使用 IP 直接访，修改 /usr/local/tomcat/conf/server.xml 配置中的 8080 改为 80（切记不要和其他 web 服务器冲突）

## 5. 创建多个 tomcat 实例（拓展）

### 1、先将 tomcat 进行停止

```
/usr/local/tomcat/bin/catalina.sh stop
```

### 2、拷贝原来的 Tomcat 到另外一个目录，如 tomcat-2，清 logs 目录

```
cp -a tomcat/ tomcat-2
rm -rf tomcat-2/logs/*
```

### 3、修改 Tomcat-2 中的/conf/server.xml 文件，把 shutdown 和 Connector 端口修改成另外的数值， 关 端口修改为 8006，连接端口修改为 8090

```
vim /usr/local/tomcat-2/conf/server.xml
<Server port=" 8006" shutdown=" SHUTDOWN" >
<Connector port=" 8090" protocol=" HTTP/1.1"
connectionTimeout=" 20000"
redirectPort=" 8443" />
```

### 4、修改 startup.sh 和 shutdown.sh 文件

在第一行均加入： export CATALINA\_HOME=/usr/local/tomcat-2

### 5、启动 tomcat

```
/usr/local/tomcat/bin/catalina.sh start
/usr/local/tomcat-2/bin/catalina.sh start
netstat -antp #查看 8080 和 8090 端口是否正常启动
```

### 6、查看结果

浏览 http://192.168.10.20:8080

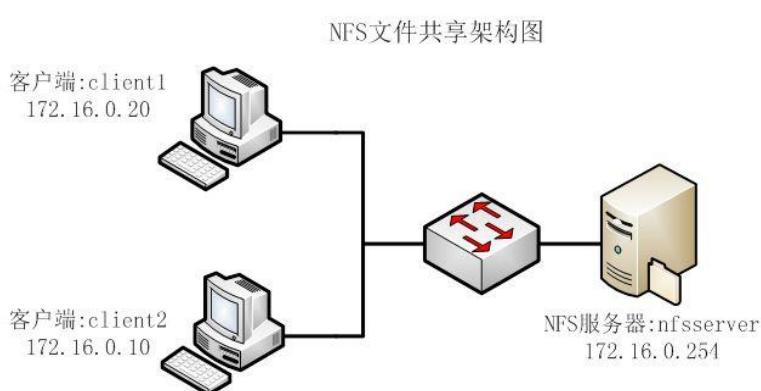
浏览 http://192.168.10.20:8090

可以直接将 8090 写入 置好的 Nginx 负载均衡中

## 网络服务—NFS

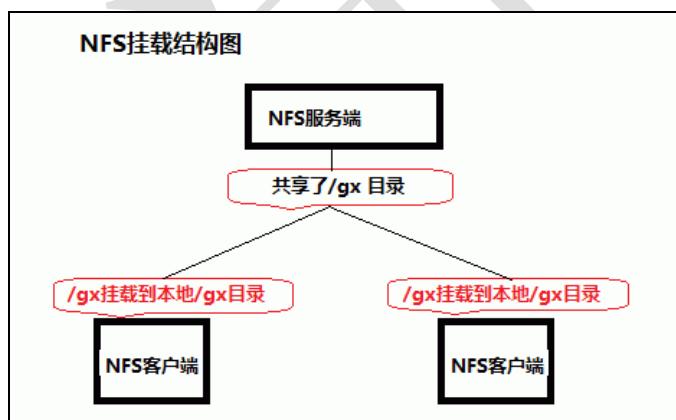
### 1. 什么是 NFS ?

NFS 是 Network File System 的缩写，即网络文件系统。一种使用于分散式文件系统的协定，由 Sun 公司开发，于 1984 年向外公布。功能是通过网络让不同的机器、不同的操作系统能够彼此分享个别的数据，让应用程序在客户端通过网络访问位于服务器磁盘中的数据，是在类 Unix 系统实现磁盘文件共享的一种方法。



它的主要功能是通过网络让不同的机器系统之间可以彼此共享文件和目录。NFS 服务器可以允许 NFS 客户端将远端 NFS 服务器端的共享目录挂载到本地的 NFS 客户端中。在本地的 NFS 客户端的机器看来，NFS 服务器端共享的目录就好像自己的磁盘分区和目录一样。一般客户端挂载到本地目录的名字可以随便，但为方便管理，我们要和服务器端一样比较好。

NFS 一般用来存储共享视频、图片等静态数据。



### 2. NFS 挂载原理

NFS 是通过网络来进行服务端和客户端之间的数据传输。两者之间要传输数据就要有相对应的网络端口来进行传

输。NFS 服务器到底使用什么网络端口来传输数据的，NFS 服务器端其实是机选择端口来进行数据传输。 NFS 客户端又是如何知 NFS 服务器端到底使用的是哪个端口呢？其实 NFS 服务器时通过远程过程调用(remote procedure call 简称 RPC) 协议/服务来实现的。也就是说 RPC 服务会统一管理 NFS 的端口，客户端和服务端通过 RPC 来先沟通 NFS 使用了哪些端口，之后再利用这些端口（小于 1024）来进行数据的传输。

也就是 RPC 管理服务端的 NFS 端口分，客户端要传数据，客户端的 RPC 会先跟服务端的 RPC 去要服务器的端口，要到端口后再建立连接，然后传输数据。

### RPC 和 NFS 之 又是如何之 相互通讯的？

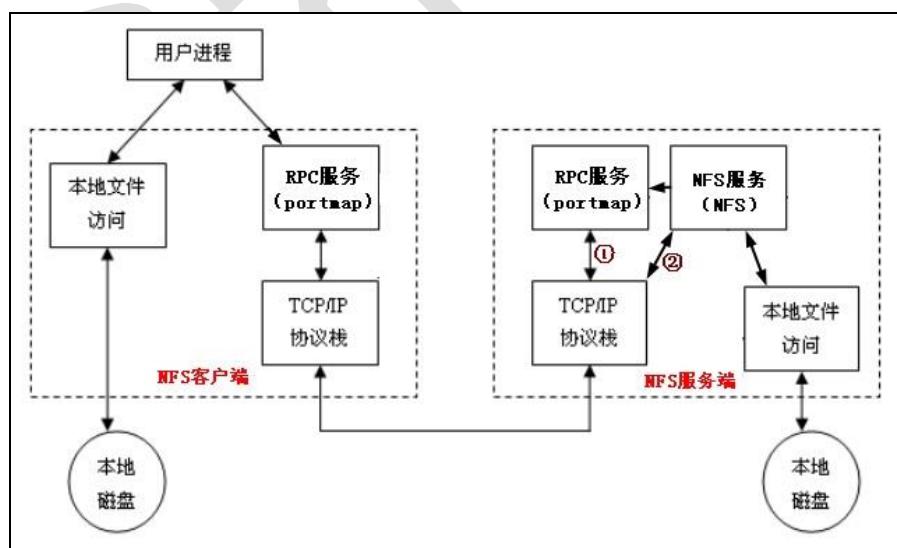
先当 NFS 启动后，就会机的使用一些端口，然后 NFS 就会向 RPC 去注册这些端口。RPC 就会记录下这些端口。并且 RPC 会开启 111 端口，等待客户端 RPC 的请求，如果客户端有请求，服务端的 RPC 就会将记录的 NFS 端口信息告知客户端。

### RPC 和 NFS 的启动 序是怎样的？

在启动 NFS SERVER 之前，先要启动 RPC 服务（即 portmap 服务，下同）否则 NFS SERVER 就无法向 RPC 服务区注册，另外，如果 RPC 服务新启动，原来已经注册好的 NFS 端口数据就会全丢失。因此此时 RPC 服务管理的 NFS 程序也要新启动以新向 RPC 注册。特别注意：一般修改 NFS 配置文档后，是不要启 NFS 的，直接在命令执行 /etc/init.d/nfs reload

### 总结：客户端 NFS 和服务端 NFS 通讯过程

- 1) 先服务器端启动 RPC 服务，并开启 111 端口
- 2) 启动 NFS 服务，并向 RPC 注册端口信息
- 3) 客户端启动 RPC (portmap 服务)，向服务端的 RPC(portmap) 服务请求服务端的 NFS 端口**
- 4) 服务端的 RPC(portmap) 服务反 NFS 端口信息给客户端。
- 5) 客户端通过获取的 NFS 端口来建立和服务端的 NFS 连接并进行数据的传输。



### 3. NFS 相关协议及软件安装管理

协议：

RPC (Remote Procedure Call Protocol) ——远程过程调用协议

软件：

nfs-utils-\*：包括 NFS 命令与监控程序

rpcbind-\*：支持安全 NFS RPC 服务的连接

注：通常情况下，是作为系统的 认包安装的

Cent OS6.\*之前 rpcbind 叫 portmap

### 4. NFS 系统守护进程

nfs：它是基本的 NFS 守护进程，主要功能是管理客户端是否能够登录服务器

rpcbind：主要功能是进行端口映射工作。当客户端尝试连接并使用 RPC 服务器提供的服务（如 NFS 服务）时，rpcbind 会将所管理的与服务对应的端口提供给客户端，从而使客户可以通过该端口向服务器请求服务。

### 5. NFS 服务器的 配置

NFS 服务器的 配置相对比较简单，只 要在相应的 配置文件中进行设置，然后启动 NFS 服务器即可。

NFS 服务的 配置文件为 /etc/exports，这个文件是 NFS 的主要 配置文件，不过系统并没有 认值，所以这个文件不一定会存在，可能要使用 vim 手动建立，然后在文件 写入 配置内容。

**/etc/exports 文件内容格式：**

共享目录 客户端 1 (访 权 , 用户映射, 其他) 客户端 2 (访 权 , 用户映射, 其他)

a. **共享目录：** 共享目录是指 NFS 服务器共享给客户机使用的目录

b. **客户端：** 客户端是指网络中可以访问这个 NFS 共享目录的计算机

**客户端常用的指定方式：**

指定 ip 地址的主机: 192.168.0.200

指定子网中的所有主机: 192.168.88.0

指定域名的主机: www.atguigu.com

指定域中的所有主机: \*.atguigu.com

所有主机: \*

c. 设置输出目录的访 权 、用户映射等。

**访 权 选 :**

设置输出目录只读: ro

设置输出目录读写: rw

**用户映射选 :**

root\_squash: 将 root 用户的访 映射为匿名 (nfsnobody) 用户 uid 和 gid; ( 认生效)

no\_root\_squash: 保留管理员权 ，以服务器管理员的权 管理;

all\_squash: 将远程访 的用户及所属组 映射为指定 uid、gid 的匿名用户;

anonuid=xxx: 将远程访的所有用户 映射为指定 uid 的匿名用户;

anongid=xxx: 将远程访的所有用户组 映射为指定 gid 匿名组账户;

其它选：

sync: 将数据同步写入内存缓冲区与磁盘中，效率低，但可以保证数据的一致性（同步）；

async: 将数据先保存在内存缓冲区中，必要时才写入磁盘（异步）；

## 6. NFS 服务器的启动与停止

### 1、启动 NFS 服务器

为了使 NFS 服务器能正常工作，要启动 rpcbind 和 nfs 两个服务，并且 rpcbind 一定要先于 nfs 启动。

```
service rpcbind start
service nfs start
```

### 2、查询 NFS 服务器状态

```
service rpcbind status
service nfs status
```

### 3、停止 NFS 服务器

要停止 NFS 运行时，要先停止 nfs 服务再停止 rpcbind 服务，对于系统中有其他服务(如 NIS) 要使用时，不要停止 rpcbind 服务

```
service nfs stop
service rpcbind stop
```

### 4、设置 NFS 服务器的自动启动状态

设置 rpcbind 和 nfs 服务在系统运行级别 2345 自动启动。

```
chkconfig --level 2345 rpcbind on
chkconfig --level 2345 nfs on
```

### 5、查看 RPC 服务器开启了哪些端口

```
rpcinfo -p localhost
```

## 7. 实 相关实例

### 1. 将NFS服务器的/home/zhangsan共享给192.168.115.0网段，rw权

```
vi /etc/exports
/home/zhangsan 192.168.115.0 (rw)
```

### 2. 启rpcbind 和nfs 服务

```
service rpcbind restart
service nfs restart
exportfs
```

### 3. 服务器端查看nfs共享状态

```
showmount -e 本机ip
查看自己共享的服务
4. 客户端查看nfs共享状态
showmount -e NFS服务器IP
5. 客户端挂载nfs服务器共享目录
命令格式: mount NFS服务器IP:共享目录 本地挂载点目录
mount 192.168.115.10:/home/zhangsan/ /media/zhangsan/
mount | grep nfs
mount -o vers=3 共享 本地 #指定挂载使用nfs V3版本（免同步延迟）
```

验证客户端和nfs服务器端文件是否一致:

```
[root@CentOS-2 ~]# cd /media/zhangsan/
-bash: cd: /media/zhangsan/: 权限不够
[root@CentOS-2 ~]#
[root@CentOS-2 ~]#
```

修改服务器端相应权限，不然客户端无法正常访问和使用

## 6. nfs共享权限和访问控制

a. 客户端root用户

使用客户端的root身份在nfs服务器上创建文件，文件的所有者和所属组是nobody。

b. 客户端普通用户

使用客户端的普通用户身份在nfs服务器上创建文件，所有者和所属组是nobody或普通用户。

如果明确设定了普通用户的映射用户身份，那么此时客户端用户的身份转换为指定映射用户；

如果NFS server上有同名用户，那么此时客户端登录账户的身份转换为NFS server上的同名用户；

## 7. 卸载和自动挂载

卸载:

1. 卸载客户端的挂载目录

umount 挂载点

2. 停止服务器端的共享

exportfs -au

自动挂载: /etc/fstab

格式: <server>:</remote/export> </local/directory> nfs < options> 0 0

```
#192.168.115.10:/home/zhangsan /media/zhangsan nfs defaults 0 0
```

```
#mount -a
```

## 8. 相关命令

### exportfs命令

如果我们在启动了NFS之后又修改了/etc/exports，是不是还要重新启动nfs呢？这个时候我们就可以用exportfs命令来使改动立刻生效，该命令格式如下：

格式: exportfs [-aruv]

-a 全部挂载或卸载 /etc/exports中的内容

-r 新读取/etc/exports 中的信息，并同步更新/etc/exports、/var/lib/nfs/xtab

-u 卸载单一目录（和-a一起使用为卸载所有/etc/exports文件中的目录）

-v 在export的时候，将详细的信息输出到屏幕上。

具体例子：

```
exportfs -au 卸载所有共享目录
exportfs -ra 新共享所有目录并输出详细信息
```

### rpcinfo命令

利用rpcinfo -p 可以查看出RPC开启的端口所提供的程序有哪些

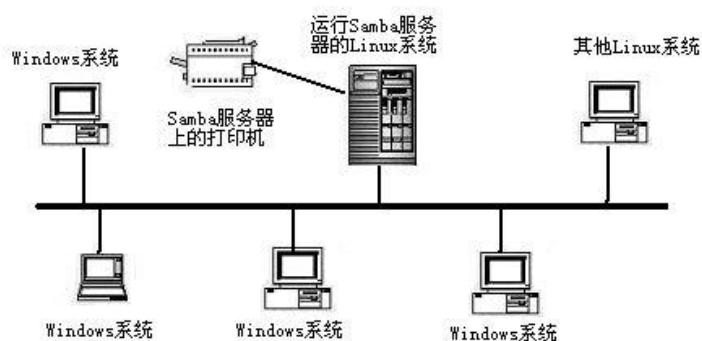
其中nfs 开启的是2049， portmapper(rpcbind) 开启的是111， 其余则是rpc开启的



## 网络服务-SAMBA

### 1. Samba 概述

SMB (Server Messages Block, 信息服务块) 是一种在局域网上共享文件和打印机的一种通信协议，它为局域网内的不同操作系统的计算机之间提供文件及打印机等资源的共享服务。SMB 协议是客户机/服务器型协议，客户机通过该协议可以访问服务器上的共享文件系统、打印机及其他资源。如图：



### 为什么要讲 SAMBA?

#### ftp 的优缺点:

优点：文件传输、应用层协议、可跨平台

缺点：只能实现文件传输，无法实现文件系统挂载；无法直接修改服务器端文件

#### Samba 的特性:

使用 smb/cifs 协议、可跨平台、可实现文件系统挂载、可实现服务器端修改文件

### smb 协议和 cifs 之间的关系

随着 Internet 的流行，Microsoft 希望将这个协议扩展到 Internet 上去，成为 Internet 上计算机之间相互共享数据的一种标准。因此它将原有的几乎没有多少技术文档的 SMB 协议进行整理，重新命名为 CIFS(Common Internet File System)，它使程序可以访问远程 Internet 计算机上的文件并要求此计算机提供服务。客户程序请求远在服务器上的服务器程序为它提供服务。服务器获得请求并返回响应。CIFS 是公共的或开放的 SMB 协议版本，并由 Microsoft 使用。SMB 协议在局域网上用于服务器文件访问和打印的协议。

## 2. Samba 服务详解

### Samba 软件相关信息

1. 协议: SMB/CIFS
2. 服务:

smb	实现资源共享、权限验证	TCP 139 445
3. 配置文件 (/etc/samba/)		
smb.conf	主配置文件	
smbusers	别名配置文件	

### 登录验证模式（安全级别）

- ◆ share 匿名验证
- ◆ user 本地用户验证（Samba服务器默认的安全级别，用户在访问共享资源之前必须提供用户名和密码进行验证）  
**拓展:** tdbSAM: 该方式是使用一个数据库文件来验证。数据库文件叫passdb. tdb。可以通过pdbedit -a 向数据库中添加新用户，不过要建立的Samba用户必须先是系统用户。也可以理解为我们使用pdbedit -a 将系统用户转化为了samba用户。pdbedit命令的参数很多，列出几个主要的。  
pdbedit -a username: 新建Samba账户（将系统用户转化为samba用户，并设置密码）  
pdbedit -x username: 删除Samba账户  
pdbedit -L: 列出Samba用户列表，读取passdb. tdb数据库文件。
- ◆ 别名用户访问（虚拟用户）

### 常见配置参数解释

[global] 用于定义Samba服务器的总体特性，其配置项对所有共享资源生效

workgroup = WORKGROUP

#设定 Samba Server 所要加入的工作组或者域。

server string = Samba Server Version %v

#设定 Samba Server 的注释，可以是任何字符串，也可以不填。宏%v表示显示Samba的版本号。

interfaces = lo eth0 192.168.12.2/24

#设置Samba Server监听哪些网卡，可以写网卡名，也可以写该网卡的IP地址。

hosts allow = 127. 192.168.1. 192.168.10.1

#表示允许连接到Samba Server的客户端，多个参数以空格隔开。可以用一个IP表示，也可以用一个网段表示。

hosts deny 与hosts allow 刚好相反（二选一）。

例如：

hosts allow=172.17.2. EXCEPT172.17.2.50

表示容许来自172.17.2.\*.\*的主机连接，但排除172.17.2.50

hosts allow=172.17.2.0/255.255.0.0

表示容许来自172.17.2.0/255.255.0.0子网中的所有主机连接

```

log file = /var/log/samba/log.%m
#设置Samba Server日志文件的存储位置以及日志文件名称。在文件名后加个宏%m（主机名），表示对每台访问
Samba Server的机器都单独记录一个日志文件。

max log size = 50
#设置Samba Server日志文件的最大容量，单位为kB，0代表不限制

security = user
#设置用户访问Samba Server的验证方式。

passdb backend = tdbSAM
load printers = yes/no
#设置是否在启动Samba时就共享打印机

[homes]用于设置用户宿主目录的共享属性（特殊共享）
[homes]
comment = Home Directories
browseable = no
writable = yes
;valid users = %S
#共享名（特殊共享，泛指每个用户对应的家目录）
#共享描述
#共享是否可被查看
#共享是否可写
#允许访问该共享的用户

例如：valid users = bob, @bob（多个用户或者组中间用逗号隔开，如果要加入一个组就用“@组名”表示。）

[printers]用于设置打印机共享资源的属性（特殊共享，共享打印设备，现在基本不用）
[printers]
comment = All Printers
path = /var/spool/samba
browseable = no
guest ok = no
writable = no
printable = yes
#共享名
#共享描述
#共享路径
#共享是否可被查看
#是否可以匿名访问，类似于public
#是否可写
#是否可以打印

[自定义]自定义共享区域
[自定义]
comment = the share is xbz
path = /share/zdy
public = yes
browseable = yes
writable = yes
#共享名
#共享描述
#共享路径
#是否可以匿名访问，类似于guest ok
#共享是否可被查看
#是否可写(同时设置目录的W)

```

#### 配置文件检查工具

testparm : 若显示"Loaded services file OK."信息表示配置文件的语法是正确的  
-v: 显示samba所支持的所有选项

## 访问控制

写入权限的控制方式（类似于vsftp的限制方式）：

- ◆ 配置文件开启，文件系统严格控制（尽量采用这种）

writable = yes

- ◆ setfacl 或 chmod 777
- ◆ 文件系统开启，配置文件严格控制
  - chmod 777 /dir
  - read only = yes
  - write list = 用户, @组

## 服务启动管理

- ◆ 启动、停止、重新启动和重新加载Samba服务
  - service smb start|stop|restart|reload
- ◆ 开机自动启动samba服务
  - chkconfig --level 2345 smb on|off

## 客户端登录方式

Linux 端:

```
smbclient -U 用户名 -L //服务器 IP
smbclient -U 用户名 //服务器 ip/共享名
```

Window 端

```
\\"服务器 ip\共享名
net use * /del
```

#查看服务器共享  
#登录服务器共享

#清空登录缓存

## 3. samba 部署与实验

**注：先关闭服务器和客户机上的防火墙和 SELinux**

部署流程：

1. 服务器端安装 samba
  - yum -y install samba
2. 确认客户端和相关命令软件包是否安装（默认是安装的）
  - rpm -q samba-client
  - rpm -q samba-common
3. 创建共享区域
  - 备份主配置文件
  - 创建独立的共享区间（仿照模板编写）
4. 启动 smb 服务并查看默认共享区域
  - a、service smb start
  - b、smbclient -U 用户名 -L smbserverIP

## 本地验证（登录、上传、下载）

a、修改配置文件（添加自定义共享）

[自定义]

```
comment = the share is xbz
path = /share/zdy
public = yes
browseable = yes
writable = yes
```

b、创建共享目录并给定相应权限

```
mkdir /share/zdy
chmod 777 /share/zdy #最好使用 ACL 权限
```

c、测试配置文件并重启服务

```
testparm
```

```
service smb restart
```

d、首先，创建 Linux 用户

```
useradd -s /sbin/nologin zhangsan
passwd zhangsan #不需要创建系统密码
```

e、转换为 samba 用户

```
pdbedit -a zhangsan
```

f、客户端查看共享文件夹并登录测试

```
smbclient -U zhangsan -L IP 地址
smbclient -U zhangsan //IP 地址/共享名
```

注：由于未设置上传文件的默认权限，指定用户上传的文件只有自己能修改和覆盖。

## 访问控制 - 通过配置限制

valid users 仅允许部分用户访问共享区域

注：前提条件是将指定目录权限给到最大，通过修改配置文件来实现实验结果

### 部分用户登录 samba 服务器

修改/etc/samba/smb.conf 中自定义的共享区域

添加：设置合法用户列表

```
valid users = 用户, @组 (多个逗号分隔)
```

### 部分用户对共享区域有写权限

修改/etc/samba/smb.conf 中自定义的共享区域

添加：开启只读，设置可写列表

```
read only = yes
```

```
write list = lisi
```

### 设置上传文件的默认权限

create mask 文件的默认权限



directory mask 目录的默认权限

修改配置文件自定义的共享区域

添加:

```
create mask = 666
```

```
directory mask = 777
```

## 用户别名（虚拟用户）

1) 添加别名 (/etc/samba/smbusers)

添加:

```
zhangsan = zs
```

2) 启用别名（修改主配置文件）

```
vim /etc/samba/smb.conf
```

添加:

```
username map = /etc/samba/smbusers
```

3) 测试

```
smbclient -U 别名 //服务器 ip/共享名
```

## 映射网络驱动器（挂载）

Linux 下:

临时挂载:

```
mount -t cifs -o username=xxx,password=xxx //服务器 ip/服务器共享 /本地挂载目录
```

永久挂载: /etc/fstab

```
//服务器 ip/服务器共享 /本地挂载目录 cifs defaults,username=xxx,password=xxx 0 0
```

Window 下:

我的电脑、计算机、此电脑、这台电脑等右键映射网络驱动器【注意是反斜杠 \】

## 图形化 web 管理界面

1. 安装（导入安装包）

使用 `yum -y install *` 安装所有上传的 rpm 包

2. 修改/etc/xinetd.d/swat

添加:

```
only_from = 登录来源 IP
```

```
disable = no
```

3. 重启 xinetd 服务

```
service xinetd restart
```

4. 测试

浏览器: IP: 901 #登录时注意端口号

## SAMBA 配置文件详解

### 全局参数：

=====Global Settings =====

[global]

config file = /usr/local/samba/lib/smb.conf.%m

说明： config file 可以让你使用另一个配置文件来覆盖缺省的配置文件。如果文件不存在，则该项无效。这个参数很有用，可以使得 samba 配置更灵活，可以让一台 samba 服务器模拟多台不同配置的服务器。比如，你想让 PC1（主机名）这台电脑在访问 Samba Server 时使用它自己的配置文件，那么先在/etc/samba/host/下为 PC1 配置一个名为 smb.conf.pc1 的文件，然后在 smb.conf 中加入：config file = /etc/samba/host/smb.conf.%m。这样当 PC1 请求连接 Samba Server 时，smb.conf.%m 就被替换成 smb.conf.pc1。这样，对于 PC1 来说，它所使用的 Samba 服务就是由 smb.conf.pc1 定义的，而其他机器访问 Samba Server 则还是应用 smb.conf。

workgroup = WORKGROUP

说明：设定 Samba Server 所要加入的工作组或者域。

server string = Samba Server Version %v

说明：设定 Samba Server 的注释，可以是任何字符串，也可以不填。宏%v 表示显示 Samba 的版本号。

netbios name = smbserver

说明：设置 Samba Server 的 NetBIOS 名称。如果不填，则默认会使用该服务器的 DNS 名称的第一部分。netbios name 和 workgroup 名字不要设置成一样了。

interfaces = lo eth0 192.168.12.2/24 192.168.13.2/24

说明：设置 Samba Server 监听哪些网卡，可以写网卡名，也可以写该网卡的 IP 地址。

hosts allow = 127. 192.168.1. 192.168.10.1

说明：表示允许连接到 Samba Server 的客户端，多个参数以空格隔开。可以用一个 IP 表示，也可以用一个网段表示。hosts deny 与 hosts allow 刚好相反。

例如：hosts allow=172.17.2.EXCEPT172.17.2.50

表示容许来自 172.17.2.\*.\* 的主机连接，但排除 172.17.2.50

hosts allow=172.17.2.0/255.255.0.0

表示容许来自 172.17.2.0/255.255.0.0 子网中的所有主机连接

hosts allow=M1, M2

表示容许来自 M1 和 M2 两台计算机连接

hosts allow=@xq

表示容许来自 XQ 网域的所有计算机连接

```
max connections = 0
```

说明：max connections 用来指定连接 Samba Server 的最大连接数目。如果超出连接数目，则新的连接请求将被拒绝。0 表示不限制。

```
deadtime = 0
```

说明：deadtime 用来设置断掉一个没有打开任何文件的连接的时间。单位是分钟，0 代表 Samba Server 不自动切断任何连接。

```
time server = yes/no
```

说明：time server 用来设置让 nmbd 成为 windows 客户端的时间服务器。

```
log file = /var/log/samba/log.%m
```

说明：设置 Samba Server 日志文件的存储位置以及日志文件名称。在文件名后加个宏%*m*（主机名），表示对每台访问 Samba Server 的机器都单独记录一个日志文件。如果 pc1、pc2 访问过 Samba Server，就会在 /var/log/samba 目录下留下 log.pc1 和 log.pc2 两个日志文件。

```
max log size = 50
```

说明：设置 Samba Server 日志文件的最大容量，单位为 kB，0 代表不限制。

```
security = user
```

说明：设置用户访问 Samba Server 的验证方式，一共有四种验证方式。

1. share：用户访问 Samba Server 不需要提供用户名和口令，安全性能较低。
2. user：Samba Server 共享目录只能被授权的用户访问，由 Samba Server 负责检查账号和密码的正确性。账号和密码要在本 Samba Server 中建立。
3. server：依靠其他 Windows NT/2000 或 Samba Server 来验证用户的账号和密码，是一种代理验证。此种安全模式下，系统管理员可以把所有的 Windows 用户和口令集中到一个 NT 系统上，使用 Windows NT 进行 Samba 认证，远程服务器可以自动认证全部用户和口令，如果认证失败，Samba 将使用用户级安全模式作为替代的方式。
4. domain：域安全级别，使用主域控制器(PDC)来完成认证。

```
passdb backend = tdbsam
```

说明：passdb backend 就是用户后台的意思。目前有三种后台：smbpasswd、tdbsam 和 ldapsam。sam 应该是 security account manager（安全账户管理）的简写。

1. smbpasswd：该方式是使用 smb 自己的工具 smbpasswd 来给系统用户（真实用户或者虚拟用户）设置一个 Samba 密码，客户端就用这个密码来访问 Samba 的资源。smbpasswd 文件默认在 /etc/samba 目录下，不过有时候要手工建立该文件。
2. tdbsam：该方式则是使用一个数据库文件来建立用户数据库。数据库文件叫 passdb.tdb， 默认在 /etc/samba 目录下。passdb.tdb 用户数据库可以使用 smbpasswd -a 来建立 Samba 用户，不过要建立的 Samba 用户必须先是系统用户。我们也可以使用 pdbedit 命令来建立 Samba 账户。pdbedit 命令的参数很多，我们列出几个主要的。

`pdbedit -a username:` 新建 Samba 账户。

`pdbedit -x username:` 删除 Samba 账户。



pdedit - L: 列出 Samba 用户列表, 读取 passdb. tdb 数据库文件。

pdedit - Lv: 列出 Samba 用户列表的详细信息。

pdedit - c “[D]” - u username: 暂停该 Samba 用户的账号。

pdedit - c “[ ]” - u username: 恢复该 Samba 用户的账号。

3. ldapsam: 该方式则是基于 LDAP 的账户管理方式来验证用户。首先要建立 LDAP 服务, 然后设置“passdb backend = ldapsam:ldap://LDAP Server”

encrypt passwords = yes/no

说明: 是否将认证密码加密。因为现在 windows 操作系统都是使用加密密码, 所以一般要开启此项。不过配置文件默认已开启。

smb passwd file = /etc/samba/smbpasswd

说明: 用来定义 samba 用户的密码文件。smbpasswd 文件如果没有那就要手工新建。

username map = /etc/samba/smbusers

说明: 用来定义用户名映射, 比如可以将 root 换成 administrator、admin 等。不过要事先在 smbusers 文件中定义好。比如: root = administrator admin, 这样就可以用 administrator 或 admin 这两个用户来代替 root 登陆 Samba Server, 更贴近 windows 用户的习惯。

guest account = nobody

说明: 用来设置 guest 用户名。

socket options = TCP\_NODELAY SO\_RCVBUF=8192 SO\_SNDBUF=8192

说明: 用来设置服务器和客户端之间会话的 Socket 选项, 可以优化传输速度。

domain master = yes/no

说明: 设置 Samba 服务器是否要成为网域主浏览器, 网域主浏览器可以管理跨子网域的浏览服务。

local master = yes/no

说明: local master 用来指定 Samba Server 是否试图成为本地网域主浏览器。如果设为 no, 则永远不会成为本地网域主浏览器。但是即使设置为 yes, 也不等于该 Samba Server 就能成为主浏览器, 还需要参加选举。

preferred master = yes/no

说明: 设置 Samba Server 一开机就强迫进行主浏览器选举, 可以提高 Samba Server 成为本地网域主浏览器的机会。如果该参数指定为 yes 时, 最好把 domain master 也指定为 yes。使用该参数时要注意: 如果在本 Samba Server 所在的子网有其他的机器(不论是 windows NT 还是其他 Samba Server)也指定为首要主浏览器时, 那么这些机器将会因为争夺主浏览器而在网络上大发广播, 影响网络性能。如果同一个区域内有多台 Samba Server, 将上面三个参数设定在一台即可。

os level = 200

说明: 设置 samba 服务器的 os level。该参数决定 Samba Server 是否有机会成为本地网域的主浏览器。os level 从 0 到 255, winNT 的 os level 是 32, win95/98 的 os level 是 1。Windows 2000 的

os level 是 64。如果设置为 0，则意味着 Samba Server 将失去浏览选择。如果想让 Samba Server 成为 PDC，那么将它的 os level 值设大些。

```
domain logons = yes/no
```

说明：设置 Samba Server 是否要做为本地域控制器。主域控制器和备份域控制器都需要开启此项。

```
logon . = %u.bat
```

说明：当使用者用 windows 客户端登陆，那么 Samba 将提供一个登陆档。如果设置成%u.bat，那么就要为每个用户提供一个登陆档。如果人比较多，那就比较麻烦。可以设置成一个具体的文件名，比如 start.bat，那么用户登陆后都会去执行 start.bat，而不用为每个用户设定一个登陆档了。这个文件要放置在 [netlogon] 的 path 设置的目录路径下。

```
wins support = yes/no
```

说明：设置 samba 服务器是否提供 wins 服务。

```
wins server = wins 服务器 IP 地址
```

说明：设置 Samba Server 是否使用别的 wins 服务器提供 wins 服务。

```
wins proxy = yes/no
```

说明：设置 Samba Server 是否开启 wins 代理服务。

```
dns proxy = yes/no
```

说明：设置 Samba Server 是否开启 dns 代理服务。

```
load printers = yes/no
```

说明：设置是否在启动 Samba 时就共享打印机。

```
printcap name = cups
```

说明：设置共享打印机的配置文件。

```
printing = cups
```

说明：设置 Samba 共享打印机的类型。现在支持的打印系统有：bsd, sysv, plp, lprng, aix, hpx, qnx

## 共享参数：

```
===== Share Definitions =====
```

[共享名]

```
comment = 任意字符串
```

说明：comment 是对该共享的描述，可以是任意字符串。

path = 共享目录路径

说明：path 用来指定共享目录的路径。可以用%u、%m 这样的宏来代替路径里的 unix 用户和客户机的 Netbios 名，用宏表示主要用于[homes]共享域。例如：如果我们不打算用 home 段做为客户的共享，而是在/home/share/下为每个 Linux 用户以他的用户名建个目录，作为他的共享目录，这样 path 就可以写成：path = /home/share/%u; 。用户在连接到这共享时具体的路径会被他的用户名代替，要注意这个用户名路径一定要存在，否则，客户机在访问时会找不到网络路径。同样，如果我们不是以用户名来划分目录，而是以客户机来划分目录，为网络上每台可以访问 samba 的机器都各自建个以它的 netbios 名的路径，作为不同机器的共享资源，就可以这样写：path = /home/share/%m。

browsable = yes/no

说明：browsable 用来指定该共享是否可以浏览。

writable = yes/no

说明：writable 用来指定该共享路径是否可写。

available = yes/no

说明：available 用来指定该共享资源是否可用。

admin users = 该共享的管理者

说明：admin users 用来指定该共享的管理员（对该共享具有完全控制权限）。在 samba 3.0 中，如果用户验证方式设置成“security=share”时，此项无效。

例如：admin users =bobyuan, jane (多个用户中间用逗号隔开)。

valid users = 允许访问该共享的用户

说明：valid users 用来指定允许访问该共享资源的用户。

例如：valid users = bobyuan, @bob, @tech (多个用户或者组中间用逗号隔开，如果要加入一个组就用“@+组名”表示。)

invalid users = 禁止访问该共享的用户

说明：invalid users 用来指定不允许访问该共享资源的用户。

例如：invalid users = root, @bob (多个用户或者组中间用逗号隔开。)

write list = 允许写入该共享的用户

说明：write list 用来指定可以在该共享下写入文件的用户。

例如：write list = bobyuan, @bob

public = yes/no

说明：public 用来指定该共享是否允许 guest 账户访问。

guest ok = yes/no

说明：意义同“public”。

几个特殊共享：

```
[homes]
comment = Home Directories
browseable = no
writable = yes
valid users = %S
; valid users = MYDOMAIN\%S
```

```
[printers]
comment = All Printers
path = /var/spool/samba
browseable = no
guest ok = no
writable = no
printable = yes
```

```
[netlogon]
comment = Network Logon Service
path = /var/lib/samba/netlogon
guest ok = yes
writable = no
share modes = no
```

```
[Profiles]
path = /var/lib/samba/profiles
browseable = no
guest ok = yes
```

# web 平台搭建-LAMP (CentOS-6)

## 一. 准备工作

### 环境要求:

操作系统: CentOS 6.X 64 位

关闭 SELinux 和 iptables 防火墙

#### 1. 安装编译工具 gcc、gcc-c++等

注意解决依赖关系, 推荐使用 yum 安装, 若不能联网可使用安装光盘做为 yum 源

##### a. 编辑 yum 配置文件, 启用本地光盘源 (双光盘)

```
mount /dev/sr0 /mnt
mount /dev/sr1 /media
vim /etc/yum.repos.d/CentOS-Media.repo
[c6-media]
name=CentOS-$releasever - Media
baseurl=file:///mnt
file:///media
gpgcheck=0
enabled=1
```

##### b. 调整 yum 源配置文件引导优先级

```
mv /etc/yum.repos.d/CentOS-Base.repo /backup
```

##### c. 安装 gcc、gcc-c++、make 等编译工具

```
yum -y install gcc gcc-c++ make
```

#### 2. 关闭系统 RPM 安装包的 Apache、MySQL 等服务

为了防止 rpm 安装的软件和接下来安装的源码软件包冲突

```
service httpd stop
service mysqld stop
.....
```

确定 rpm 包安装的 httpd 和 mysqld 不能开机自启动

```
chkconfig httpd off
chkconfig mysqld off
.....
```

#### 3. 关闭 SELinux 和 iptables

防止软件安装和调试过程被 iptables 和 SELinux 所限制, 无法实现效果

##### a. 关闭 SELinux (需重启)

```
vim /etc/selinux/config
SELINUX=disabled
```



```
reboot
b. 关闭 iptables
iptables -F
chkconfig iptables off
```

#### 4. 拷贝源码包，解包解压缩

建议将 LAMP 环境安装源码包统一存放在一个目录下，如/lamp，可以使用解压脚本解压缩

```
vim tar.sh
cd /lamp
/bin/ls *.tar.gz > ls.list
for TAR in `cat ls.list`
do
 /bin/tar -xf $TAR
done
/bin/rm ls.list
```

#### 5. 查看安装软件的磁盘空间是否充足

保证软件能正常安装，空间不足时会导致软件安装失败

```
df -h
```

#### 6. 源码软件包安装报错确认与解决方案

```
echo $? #安装软件过程中由于频繁刷屏，建议在每个步骤结束后执行此命令
.configure #此步骤报错多是依赖关系没解决或是编译工具未安装（注意关键词提示）
make #此步骤多是编译时选项参数书写错误、不存在、漏写等问题
 #一般需要检查上一个步骤：./configure --help
```

**注意：**若遇到报错，最简答的办法是，找到问题解决后重新解压软件，重新安装，步骤最简洁

## 二. 编译安装

**注意：**每个源码包配置编译安装完成后，确认安装目录下是否生成安装文件（并确定目录是否正确）

建议将安装路径指定为[**--prefix=/usr/local/软件名**]格式

### 1. 安装 libxml2

Libxml2 是一个 xml c 语言版的解析器，本来是为 Gnome 项目开发的工具，是一个基于 MIT License 的免费开源软件。它除了支持 c 语言版以外，还支持 c++、PHP、Pascal、Ruby、Tcl 等语言的绑定，能在 Windows、Linux、Solaris、MacOsX 等平台上运行。功能还是相当强大的，相信满足一般用户需求没有任何问题。

```
yum install -y libxml2-devel python-devel
cd /lamp/libxml2-2.9.1
./configure --prefix=/usr/local/libxml2/
make
```



```
make install
```

## 2. 安装 libmcrypt

libmcrypt 是加密算法扩展库。支持 DES, 3DES, RIJNDAEL, Twofish, IDEA, GOST, CAST-256, ARCFOUR, SERPENT, SAFER+等算法。

```
cd /lamp/libmcrypt-2.5.8
./configure --prefix=/usr/local/libmcrypt/
make
make install
```

安装 libltdl，也在 libmcrypt 源码目录中，非新软件

```
cd /lamp/libmcrypt-2.5.8/libltdl
./configure --enable-ltdl-install
make
make install
```

## 3. 安装 mhash

mhash 是基于离散数学原理的不可逆向的 php 加密方式扩展库，其在默认情况下不开启。mhash 的可以用于创建校验数值，消息摘要，消息认证码，以及无需原文的关键信息保存（如密码）等。

```
cd /lamp/mhash-0.9.9.9
./configure
make
make install
```

## 4. 安装 mcrypt

mcrypt 是 php 里面重要的加密支持扩展库。mcrypt 库支持 20 多种加密算法和 8 种加密模式

```
cd /lamp/mcrypt-2.6.8
export LD_LIBRARY_PATH=/usr/local/libmcrypt/lib:/usr/local/lib
变量：LD_LIBRARY_PATH 用于指定 libmcrypt 和 mhash 的库的位置
./configure --with-libmcrypt-prefix=/usr/local/libmcrypt
make
make install
```

## 5. 安装 zlib

zlib 是提供数据压缩用的函式库，由 Jean-loup Gailly 与 Mark Adler 所开发，初版 0.9 版在 1995 年 5 月 1 日发表。zlib 使用 DEFLATE 算法，最初是为 libpng 函式库所写的，后来普遍为许多软件所使用。此函式库为自由软件，使用 zlib 授权

```
cd /lamp/zlib-1.2.3
./configure
```

然后修改配置文件，否则无法正常安装此软件

```
vi Makefile
CFLAGS=-O3 -DUSE_MMAP -fPIC
#找到 CFLAGS=-O3 -DUSE_MMAP，在后面加入 -fPIC 变成（注意：小 f 大 PIC，空格）
make
make install
```

## 6. 安装 libpng

libpng 软件包包含 libpng 库。这些库被其他程式用于解码 png 图片

```
cd /lamp/libpng-1.2.31
./configure --prefix=/usr/local/libpng
make
make install
```

## 7. 安装 jpeg6

jpeg6 提供用于解码. jpg 和. jpeg 图片的库文件

```
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
```

注意：此软件默认不会自动创建所需目录，所以目录必须手工建立

```
yum -y install libtool*
cd /lamp/jpeg-6b
cp -a /usr/share/libtool/config/config.sub ./
cp -a /usr/share/libtool/config/config.guess ./
```

复制 libtool 中的文件，覆盖 jpeg-6b 中的文件（64 位中的问题）

```
./configure --prefix=/usr/local/jpeg6/ --enable-shared --enable-static
make
make install
```

--enable-shared 与--enable-static 参数分别为建立共享库和静态库使用的 libtool

## 8. 安装 freetype

FreeType 库是一个完全免费(开源)的、高质量的且可移植的字体引擎，它提供统一的接口来访问多种字体格式文件，支持单色位图、反走样位图的渲染。

```
cd /lamp/freetype-2.3.5
./configure --prefix=/usr/local/freetype/
make
make install
```



## 9. 安装 Apache

a. 源码包 2.4.\* 版本中默认没有集成 apr 的依赖包，所以需要提前解决依赖问题

```
cp -a /lamp/apr-1.4.6 /lamp/httpd-2.4.7/src/lib/apr
cp -a /lamp/apr-util-1.4.1 /lamp/httpd-2.4.7/src/lib/apr-util
```

解压 apr 和 apr-util，复制整个目录并取消目录上的版本号到指定位置，./configure 时会检测

b. Apache 默认需要依赖 pcre 软件，但由于 Apache 软件版本较高，则系统预安装的 pcre 无法使用，所以需要人为手动安装适合版本

```
cd /lamp/pcre-8.34
./configure
make
make install
```

c. Apache 的加密传输模块 mod\_ssl，需要安装此软件产生

```
yum install openssl-devel
```

d. httpd 软件安装

```
cd /lamp/httpd-2.4.7
./configure --prefix=/usr/local/apache2 --sysconfdir=/usr/local/apache2/etc
--with-included-apr --enable-so --enable-deflate=shared --enable-expire=shared
--enable-rewrite=shared --enable-ssl
make
make install
```

若前面配置 zlib 时没有指定安装目录，Apache 配置时不要添加--with-z=/usr/local/zlib/参数，--enable-ssl 选项是为了后期实现 https 提前设置的参数

e. 启动 Apache 测试

```
/usr/local/apache2/bin/apachectl start
ps aux | grep httpd
使用进程查看命令确认 Apache 是否启动，是否产生进程
netstat -tlun | grep :80
使用网络进程查看命令确认 Apache 是否启动，是否开启了 80 监听端口
```

**报错提示：**若启动时提示/usr/local/apache2/modules/mod\_deflate.so 无权限，可关闭 SELinux 解决，类似此类 so 文件不能载入或没有权限的问题，都是 SELinux 问题，MySQL 和 Apache 都可能有类似问题。

**警告提示：**发现启动服务提示：AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using localhost.localdomain. Set the 'ServerName' directive globally to suppress this message

**解决办法：**打开主配置文件 httpd.conf

搜索 ServerName (约在 200 行左右)



改为 ServerName localhost:80 (并且去掉前面的#注释)

**验证：**通过浏览器输入地址访问：http://服务器 ip，若显示 “It works” 即表明 Apache 正常工作

## 10. 安装 ncurses

Ncurses 提供字符终端处理库，包括面板和菜单。它提供了一套控制光标，建立窗口，改变前景背景颜色以及处理鼠标操作的函数。使用户在字符终端下编写应用程序时绕过了那些恼人的底层机制。简而言之，他是一个可以使应用程序直接控制终端屏幕显示的函数库。

```
yum -y install ncurses-devel
cd /lamp/ncurses-5.9
./configure --with-shared --without-debug --without-ada --enable-overwrite
make
make install
若不安装 ncurses 编译 MySQL 时会报错
```

## 11. 安装 cmake 和 bison

mysql 在 5.5 以后，不再使用 ./configure 工具，进行编译安装。而使用 cmake 工具替代了 ./configure 工具。bison 是一个自由软件，用于自动生成语法分析器程序，可用于所有常见的操作系统

```
yum -y install cmake bison
```

## 12. 安装 MySQL

```
useradd -r -s /sbin/nologin mysql
```

为 MySQL 软件创建运行用户，创建为系统用户，并限制此用户登录操作系统

```
cd /lamp/mysql-5.5.48
cmake -DCMAKE_INSTALL_PREFIX=/usr/local/mysql -DMYSQL_UNIX_ADDR=/tmp/mysql.sock
-DEXTRA_CHARSETS=all -DDEFAULT_CHARSET=utf8 -DDEFAULT_COLLATION=utf8_general_ci
-DWITH_MYISAM_STORAGE_ENGINE=1 -DWITH_INNOBASE_STORAGE_ENGINE=1
-DWITH_MEMORY_STORAGE_ENGINE=1 -DWITH_READLINE=1 -DENABLED_LOCAL_INFILE=1
-DMYSQL_USER=mysql -DMYSQL_TCP_PORT=3306
make
make install
```

选项详解：

-DCMAKE_INSTALL_PREFIX=/usr/local/mysql	安装位置
-DMYSQL_UNIX_ADDR=/tmp/mysql.sock	指定 socket (套接字) 文件位置
-DEXTRA_CHARSETS=all	扩展字符支持
-DDEFAULT_CHARSET=utf8	默认字符集
-DDEFAULT_COLLATION=utf8_general_ci	默认字符校对
-DWITH_MYISAM_STORAGE_ENGINE=1	安装 myisam 存储引擎
-DWITH_INNOBASE_STORAGE_ENGINE=1	安装 innodb 存储引擎
-DWITH_MEMORY_STORAGE_ENGINE=1	安装 memory 存储引擎



-DWITH_READLINE=1	支持 readline 库
-DENABLED_LOCAL_INFILE=1	启用加载本地数据
-DMYSQL_USER=mysql	指定 mysql 运行用户
-DMYSQL_TCP_PORT=3306	指定 mysql 端口

MySQL 安装后需要调整相应配置文件和参数才能正常运行

a. 修改 MySQL 目录的用户归属

```
cd /usr/local/mysql/
chown -R root .
chown -R mysql data
```

b. 生成配置文件，并初始化授权表

```
cp -a /lamp/mysql-5.5.48/support-files/my-medium.cnf /etc/my.cnf
复制 MySQL 配置文件到指定位置，覆盖掉系统自带文件
cd /usr/local/mysql
./scripts/mysql_install_db --user=mysql
创建数据库授权表，初始化数据库，相当于安装完操作系统后的引导设置（添加第一个用户）
```

**报错提示：**FATAL ERROR: Could not find ./bin/my\_print\_defaults

**原因：**mysql\_install\_db 初始化所调用文件时使用的是相对路径，路径不在/usr/local/mysql 时，是无法调用 my\_print\_defaults 文件并初始化成功的。

c. 启动 MySQL 服务

用原本源代码的方式去使用和启动 mysql

```
/usr/local/mysql/bin/mysqld_safe --user=mysql &
```

d. 设定 MySQL 密码

```
/usr/local/mysql/bin/mysqladmin -uroot password 123456
```

e. 登录 MySQL

```
/usr/local/mysql/bin/mysql -u root -p
mysql>show databases;
mysql>use test;
mysql>show tables;
mysql>exit
```

## 13. 安装 PHP

```
cd /lamp/php-7.0.7
./configure --prefix=/usr/local/php/ --with-config-file-path=/usr/local/php/etc/
--with-apxs2=/usr/local/apache2/bin/apxs --with-libxml-dir=/usr/local/libxml2/
--with-jpeg-dir=/usr/local/jpeg6/ --with-png-dir=/usr/local/libpng/
--with-freetype-dir=/usr/local/freetype/ --with-mcrypt=/usr/local/libmcrypt/
```

```
--with-mysqli=/usr/local/mysql/bin/mysql_config --enable-soap --enable-mbstring=all
--enable-sockets --with-pdo-mysql=/usr/local/mysql --with-gd --without-pear
make
make install
```

### 选项详解：

--with-config-file-path=/usr/local/php/etc/	指定配置文件目录
--with-apxs2=/usr/local/apache2/bin/apxs	指定 apache 动态模块位置
--with-libxml-dir=/usr/local/libxml2/	指定 libxml 位置
--with-jpeg-dir=/usr/local/jpeg6/	指定 jpeg 位置
--with-png-dir=/usr/local/libpng/	指定 libpng 位置
--with-freetype-dir=/usr/local/freetype/	指定 freetype 位
--with-mcrypt=/usr/local/libmcrypt/	指定 libmcrypt 位置
--with-mysqli=/usr/local/mysql/bin/mysql_config	指定 mysqli 位置
--with-gd	启用 gd 库
--enable-soap	支持 soap 服务
--enable-mbstring=all	支持多字节，字符串
--enable-sockets	支持套接字
--with-pdo-mysql=/usr/local/mysql	启用 mysql 的 pdo 模块支持
--without-pear	不安装 pear(安装 pear 需要连接互联网)

### PHP 安装后需要调整相应配置文件和参数才能正常运行

#### a. 生成 php 配置文件

```
mkdir /usr/local/php/etc
cp /lamp/php-7.0.7/php.ini-production /usr/local/php/etc/php.ini
```

#### b. 修改 Apache 配置文件，使其识别\*.php 文件，并能通过 php 模块调用 php 进行页面解析

```
vim /usr/local/apache2/etc/httpd.conf
AddType application/x-httpd-php .php .phtml
AddType application/x-httpd-php-source .phps
```

重启 Apache 服务

```
/usr/local/apache2/bin/apachectl stop
/usr/local/apache2/bin/apachectl start
```

#### c. 测试 php 页面是否能正常解析（即 apache 和 php 连通性）

```
vim /usr/local/apache2/htdocs/test.php
<?php
phpinfo();
?>
```

通过浏览器输入地址访问：http://Apache 服务器地址/test.php

## 14. 为 PHP 安装 openssl 模块



OpenSSL 是一个强大的安全套接字层密码库，囊括主要的密码算法、常用的密钥和证书封装管理功能及 SSL 协议，并提供丰富的应用程序供测试或其它目的使用。

```
cd /lamp/php-7.0.7/ext/openssl
mv config0.m4 config.m4
/usr/local/php/bin/phpize
./configure --with-openssl --with-php-config=/usr/local/php/bin/php-config
make
make install
```

## 15. 为 PHP 安装 memcache 模块

Memcache 是一个高性能的分布式的内存对象缓存系统，通过在内存里维护一个统一的巨大的 hash 表，它能够用来存储各种格式的数据，包括图像、视频、文件以及数据库检索的结果等。简单的说就是将数据调用到内存中，然后从内存中读取，从而大大提高读取速度。

```
unzip pecl-memcache-php7.zip
cd pecl-memcache-php7
/usr/local/php/bin/phpize
./configure --with-php-config=/usr/local/php/bin/php-config
make
make install
```

## 16. 修改 php 配置文件，使其识别并调用 openssl 和 memcache 两个模块

```
vi /usr/local/php/etc/php.ini
extension_dir="/usr/local/php/lib/php/extensions/no-debug-zts-20151012/"
取消分号注释，并添加以上路径（此路径来自于模块安装命令的结果）
extension="openssl.so";
extension="memcache.so";
添加以上两个库文件的调用
```

重启 apache，刷新 phpinfo 页面，并查看是否有两个新增的模块

## 17. 安装 memcached 服务

```
yum -y install libevent-devel
cd /lamp/memcached-1.4.17
./configure --prefix=/usr/local/memcache
make
make install
```

```
useradd -r -s /sbin/nologin memcache
添加 memcache 用户，此用户不用登录，不设置密码
/usr/local/memcache/bin/memcached -umemcache &
启动 memcache 服务，并设置为后台运行
```



```
netstat -an | grep :11211
检查 memcache 是否正常启动，并监听了 11211 端口
```

## 18. 安装 phpMyAdmin

phpMyAdmin 是一个以 PHP 为基础，以 Web-Base 方式架构在网站主机上的 MySQL 的数据库管理工具，让管理者可用 Web 接口管理 MySQL 数据库。

```
cp -a /lamp/phpMyAdmin-4.1.4-all-languages /usr/local/apache2/htdocs/phpmyadmin
cd /usr/local/apache2/htdocs/phpmyadmin
cp -a config.sample.inc.php config.inc.php
vim config.inc.php
$cfg['Servers'][$i]['auth_type'] = 'cookie';
$cfg['Servers'][$i]['auth_type'] = 'http';
设置 auth_type 为 http，即设置为 HTTP 身份认证模式（新增即可）
```

通过浏览器输入地址访问：http://Apache 服务器地址/phpmyadmin/index.php  
用户名为 root，密码为 MySQL 设置时指定的 root 密码 123456

## 19. 设置 Apache、MySQL、Memcache 开机自启

借助系统自带脚本/etc/rc.local，此脚本开机后会自动加载，我们可以将源码安装的服务启动命令写入该脚本，间接实现开机自启动

```
vi /etc/rc.local
/usr/local/apache2/bin/apachectl start
/usr/local/mysql/bin/mysqld_safe --user=mysql &
/usr/local/memcache/bin/memcached -umemcache &
```

## 20. 项目迁移：

- 1、把 php 项目拷贝到网站默认目录下：/usr/local/apache2/htdocs/\*\*
- 2、使用 phpMyAdmin 创建网站所需数据库

注意事项：注意目录权限和归属，防止权限过大或者权限过小

**切记：做完 LAMP 环境后保存一个快照，后面讲 Apache 要使用！**

# web 平台搭建-LAMP (CentOS-7)

## 一. 准备工作

### 环境要求:

操作系统: CentOS 7.X 64 位

网络配置: nmtui 字符终端图形管理工具或者直接编辑配置文件

关闭 SELinux 和 firewalld 防火墙

防火墙:

临时关闭: systemctl stop firewalld

永久关闭: systemctl disable firewalld

### 1. 安装编译工具 gcc、gcc-c++ 等

注意解决依赖关系, 推荐使用 yum 安装, 若不能联网可使用安装光盘做为 yum 源

#### a. 编辑 yum 配置文件, 启用本地光盘源 (只有一张盘)

```
mount /dev/sr0 /mnt
vim /etc/yum.repos.d/CentOS-Media.repo
[c7-media]
name=CentOS-$releasever - Media
baseurl=file:///mnt
gpgcheck=0
enabled=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-7
```

#### b. 调整 yum 源配置文件引导优先级

```
mv /etc/yum.repos.d/CentOS-Base.repo /backup
```

#### c. 安装 gcc、gcc-c++、make 等编译工具

```
yum -y install gcc gcc-c++ make
```

### 2. 关闭系统 RPM 安装包的 Apache、MySQL 等服务

为了防止 rpm 安装的软件和接下来安装的源码软件包冲突

```
systemctl stop httpd
systemctl stop mysqld
.....
```

确定 rpm 包安装的 httpd 和 mysqld 不能开机自启动

```
systemctl disable httpd
systemctl disable mysqld
.....
```

### 3. 关闭 SELinux 和 firewalld

防止软件安装和调试过程被 firewalld 和 SELinux 所限制, 无法实现效果

**a. 关闭 SELinux (需重启)**

```
vim /etc/selinux/config
SELINUX=disabled
reboot
```

**b. 关闭 firewalld**

```
iptables -F
systemctl disable firewalld
reboot
```

**4. 拷贝源码包，解包解压缩**

建议将 LAMP 环境安装源码包统一存放在一个目录下，如/lamp，可以使用解压脚本解压缩

**注意：使用 Xshell 上传时传到/tmp 下，/root 目录无法上传**

```
vim tar.sh
cd /lamp
/bin/ls *.tar.gz > ls.list
for TAR in `cat ls.list`
do
 /bin/tar -xf $TAR
done
/bin/rm ls.list
```

**5. 查看安装软件的磁盘空间是否充足**

保证软件能正常安装，空间不足时会导致软件安装失败

```
df -h
```

**6. 源码软件包安装报错确认与解决方案**

```
echo $? #安装软件过程中由于频繁刷屏，建议在每个步骤结束后执行此命令
.configure #此步骤报错多是依赖关系没解决或是编译工具未安装（注意关键词提示）
make #此步骤多是编译时选项参数书写错误、不存在、漏写等问题
 #一般需要检查上一个步骤：./configure --help
```

**注意：**若遇到报错，最简答的办法是，找到问题解决后重新解压软件，重新安装，步骤最简洁

## 二. 编译安装

**注意：每个源码包配置编译安装完成后，确认安装目录下是否生成安装文件（并确定目录是否正确）**

建议将安装路径指定为[--prefix=/usr/local/软件名]格式

### 1. 安装 libxml2

```
yum install -y libxml2-devel python-devel
cd /lamp/libxml2-2.9.1
./configure --prefix=/usr/local/libxml2/
```

```
make
make install
```

## 2. 安装 libmcrypt

```
cd /lamp/libmcrypt-2.5.8
./configure --prefix=/usr/local/libmcrypt/
make
make install
```

安装 libltdl，也在 libmcrypt 源码目录中，非新软件

```
cd /lamp/libmcrypt-2.5.8/libltdl
./configure --enable-ltdl-install
make
make install
```

## 3. 安装 mhash

```
cd /lamp/mhash-0.9.9.9
./configure
make
make install
```

## 4. 安装 mcrypt

```
cd /lamp/mcrypt-2.6.8
export LD_LIBRARY_PATH=/usr/local/libmcrypt/lib:/usr/local/lib
变量: LD_LIBRARY_PATH 用于指定 libmcrypt 和 mhash 的库的位置
./configure --with-libmcrypt-prefix=/usr/local/libmcrypt
make
make install
```

## 5. 安装 zlib

```
cd /lamp/zlib-1.2.3
./configure
然后修改配置文件, 否则无法正常安装此软件
vi Makefile
CFLAGS=-O3 -DUSE_MMAP -fPIC
#找到 CFLAGS=-O3 -DUSE_MMAP, 在后面加入 -fPIC 变成 (注意: 小 f 大 PIC, 空格)
make
make install
```



## 6. 安装 libpng

```
cd /lamp/libpng-1.2.31
./configure --prefix=/usr/local/libpng
make
make install
```

## 7. 安装 jpeg6

```
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
```

注意：此软件默认不会自动创建所需目录，所以目录必须手工建立

```
yum -y install libtool*
cd /lamp/jpeg-6b
cp -a /usr/share/libtool/config/config.sub ./
cp -a /usr/share/libtool/config/config.guess ./
```

复制 libtool 中的文件，覆盖 jpeg-6b 中的文件（64 位中的问题）

```
./configure --prefix=/usr/local/jpeg6/ --enable-shared --enable-static
make
make install
```

--enable-shared 与--enable-static 参数分别为建立共享库和静态库使用的 libtool

## 8. 安装 freetype

```
cd /lamp/freetype-2.3.5
./configure --prefix=/usr/local/freetype/
make
make install
```

## 9. 安装 Apache

a. 源码包 2.4.\* 版本中默认没有集成 apr 的依赖包，所以需要提前解决依赖问题

```
cp -a /lamp/apr-1.4.6 /lamp/httpd-2.4.7/src/lib/apr
cp -a /lamp/apr-util-1.4.1 /lamp/httpd-2.4.7/src/lib/apr-util
```

解压 apr 和 apr-util，复制整个目录并取消目录上的版本号到指定位置，./configure 时会检测

b. Apache 默认需要依赖 pcre 软件，但由于 Apache 软件版本较高，则系统预安装的 pcre 无法使用，所以需要人为手动安装适合版本

```
cd /lamp/pcre-8.34
```

```
./configure
make
make install
```

c. Apache 的加密传输模块 mod\_ssl，需要安装此软件产生

```
yum -y install openssl-devel
```

#### d. httpd 软件安装

```
cd /lamp/httpd-2.4.7
./configure --prefix=/usr/local/apache2 --sysconfdir=/usr/local/apache2/etc
--with-included-apr --enable-so --enable-deflate=shared --enable-expire=shared
--enable-rewrite=shared --enable-ssl
make
make install
```

若前面配置 zlib 时没有指定安装目录，Apache 配置时不要添加--with-z=/usr/local/zlib/参数，  
--enable-ssl 选项是为了后期实现 https 提前设置的参数

#### e. 启动 Apache 测试

```
/usr/local/apache2/bin/apachectl start
```

```
ps aux | grep httpd
```

使用进程查看命令确认 Apache 是否启动，是否产生进程

```
netstat -tlun | grep :80
```

使用网络进程查看命令确认 Apache 是否启动，是否开启了 80 监听端口

**注意事项：**在 CentOS 7 操作系统上 Apache 默认监听了 Ipv6 地址的 80 端口，没有监听 Ipv4 的地址，所以需要修改下配置文件使其监听。

Listen 0.0.0.0:80

**报错提示：**若启动时提示/usr/local/apache2/modules/mod\_deflate.so 无权限，可关闭 SELinux 解决，类似此类 so 文件不能载入或没有权限的问题，都是 SELinux 问题，MySQL 和 Apache 都可能有类似问题。

**警告提示：**发现启动服务提示：AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using localhost.localdomain. Set the 'ServerName' directive globally to suppress this message

**解决办法：**打开主配置文件 httpd.conf

搜索 ServerName (约在 200 行左右)

改为 ServerName localhost:80 (并且去掉前面的#注释)

**验证：**通过浏览器输入地址访问：http://服务器 ip，若显示“It works”即表明 Apache 正常工作

## 10. 安装 ncurses

```
yum -y install ncurses-devel
cd /lamp/ncurses-5.9
```

```
./configure --with-shared --without-debug --without-ada --enable-overwrite
make
make install
若不安装 ncurses 编译 MySQL 时会报错
```

## 11. 安装 cmake 和 bison

```
yum -y install cmake bison
```

## 12. 安装 MySQL

```
useradd -r -s /sbin/nologin mysql
为 MySQL 软件创建运行用户，创建为系统用户，并限制此用户登录操作系统
cd /lamp/mysql-5.5.48
cmake -DCMAKE_INSTALL_PREFIX=/usr/local/mysql -DMYSQL_UNIX_ADDR=/tmp/mysql.sock
-DEXTRA_CHARSETS=all -DDEFAULT_CHARSET=utf8 -DDEFAULT_COLLATION=utf8_general_ci
-DWITH_MYISAM_STORAGE_ENGINE=1 -DWITH_INNODB_STORAGE_ENGINE=1
-DWITH_MEMORY_STORAGE_ENGINE=1 -DWITH_READLINE=1 -DENABLED_LOCAL_INFILE=1
-DMYSQL_USER=mysql -DMYSQL_TCP_PORT=3306
make
make install
```

选项详解：

-DCMAKE_INSTALL_PREFIX=/usr/local/mysql	安装位置
-DMYSQL_UNIX_ADDR=/tmp/mysql.sock	指定 socket (套接字) 文件位置
-DEXTRA_CHARSETS=all	扩展字符支持
-DDEFAULT_CHARSET=utf8	默认字符集
-DDEFAULT_COLLATION=utf8_general_ci	默认字符校对
-DWITH_MYISAM_STORAGE_ENGINE=1	安装 myisam 存储引擎
-DWITH_INNODB_STORAGE_ENGINE=1	安装 innodb 存储引擎
-DWITH_MEMORY_STORAGE_ENGINE=1	安装 memory 存储引擎
-DWITH_READLINE=1	支持 readline 库
-DENABLED_LOCAL_INFILE=1	启用加载本地数据
-DMYSQL_USER=mysql	指定 mysql 运行用户
-DMYSQL_TCP_PORT=3306	指定 mysql 端口

MySQL 安装后需要调整相应配置文件和参数才能正常运行

### a. 修改 MySQL 目录的用户归属

```
cd /usr/local/mysql/
chown -R root .
chown -R mysql data
```

### b. 生成配置文件，并初始化授权表

```
cp -a /lamp/mysql-5.5.48/support-files/my-medium.cnf /etc/my.cnf
```

复制 MySQL 配置文件到指定位置，覆盖掉系统自带文件

```
cd /usr/local/mysql
```

```
./scripts/mysql_install_db --user=mysql
```

创建数据库授权表，初始化数据库，相当于安装完操作系统后的引导设置（添加第一个用户）

**报错提示：**FATAL ERROR: Could not find ./bin/my\_print\_defaults

**原因：**mysql\_install\_db 初始化所调用文件时使用的是相对路径，路径不在/usr/local/mysql 时，是无法调用 my\_print\_defaults 文件并初始化成功的。

#### c. 启动 MySQL 服务

用原本源代码的方式去使用和启动 mysql

```
/usr/local/mysql/bin/mysqld_safe --user=mysql &
```

#### d. 设定 MySQL 密码

```
/usr/local/mysql/bin/mysqladmin -uroot password 123456
```

#### e. 登录 MySQL

```
/usr/local/mysql/bin/mysql -u root -p
```

```
mysql>show databases;
```

```
mysql>use test;
```

```
mysql>show tables;
```

```
mysql>exit
```

## 13. 安装 PHP

```
cd /lamp/php-7.0.7
```

```
./configure --prefix=/usr/local/php/ --with-config-file-path=/usr/local/php/etc/ --with-apxs2=/usr/local/apache2/bin/apxs --with-libxml-dir=/usr/local/libxml2/ --with-jpeg-dir=/usr/local/jpeg6/ --with-png-dir=/usr/local/libpng/ --with-freetype-dir=/usr/local/freetype/ --with-mcrypt=/usr/local/libmcrypt/ --with-mysqli=/usr/local/mysql/bin/mysql_config --enable-soap --enable-mbstring=all --enable-sockets --with-pdo-mysql=/usr/local/mysql --with-gd --without-pear
```

```
make
```

```
make install
```

#### 选项详解：

--with-config-file-path=/usr/local/php/etc/ 指定配置文件目录

--with-apxs2=/usr/local/apache2/bin/apxs 指定 apache 动态模块位置

--with-libxml-dir=/usr/local/libxml2/ 指定 libxml 位置

--with-jpeg-dir=/usr/local/jpeg6/ 指定 jpeg 位置

--with-png-dir=/usr/local/libpng/ 指定 libpng 位置

--with-freetype-dir=/usr/local/freetype/ 指定 freetype 位



--with-mcrypt=/usr/local/libmcrypt/	指定 libmcrypt 位置
--with-mysqli=/usr/local/mysql/bin/mysql_config	指定 mysqli 位置
--with-gd	启用 gd 库
--enable-soap	支持 soap 服务
--enable-mbstring=all	支持多字节, 字符串
--enable-sockets	支持套接字
--with-pdo-mysql=/usr/local/mysql	启用 mysql 的 pdo 模块支持
--without-pecl	不安装 pecl(安装 pecl 需要连接互联网)

PHP 安装后需要调整相应配置文件和参数才能正常运行

a. 生成 php 配置文件

```
mkdir /usr/local/php/etc
cp /lamp/php-7.0.7/php.ini-production /usr/local/php/etc/php.ini
```

b. 修改 Apache 配置文件，使其识别\*.php 文件，并能通过 php 模块调用 php 进行页面解析

```
vim /usr/local/apache2/etc/httpd.conf
AddType application/x-httpd-php .php .phtml
AddType application/x-httpd-php-source .phps
```

重启 Apache 服务

```
/usr/local/apache2/bin/apachectl stop
/usr/local/apache2/bin/apachectl start
```

c. 测试 php 页面是否能正常解析（即 apache 和 php 连通性）

```
vim /usr/local/apache2/htdocs/test.php
<?php
 phpinfo();
?>
```

通过浏览器输入地址访问：http://Apache 服务器地址/test.php

## 14. 为 PHP 安装 openssl 模块

```
cd /lamp/php-7.0.7/ext/openssl
mv config0.m4 config.m4
/usr/local/php/bin/phpize
./configure --with-openssl --with-php-config=/usr/local/php/bin/php-config
make
make install
```

## 15. 为 PHP 安装 memcache 模块

```
unzip pecl-memcache-php7.zip
cd /lamp/pecl-memcache-php7
/usr/local/php/bin/phpize
```



```
./configure --with-php-config=/usr/local/php/bin/php-config
make
make install
```

## 16. 修改 php 配置文件，使其识别并调用 openssl 和 memcache 两个模块

```
vi /usr/local/php/etc/php.ini
extension_dir="/usr/local/php/lib/php/extensions/no-debug-zts-20151012/"
取消分号注释，并添加以上路径（此路径来自于模块安装命令的结果）
extension="openssl.so";
extension="memcache.so";
添加以上两个库文件的调用
```

重启 apache，刷新 phpinfo 页面，并查看是否有两个新增的模块

## 17. 安装 memcached 服务

```
wget
ftp://ftp.pbone.net/mirror/ftp.centos.org/7.6.1810/os/x86_64/Packages/libevent-devel-2.
0.21-4.el7.x86_64.rpm
yum -y install libevent-devel
cd /lamp/memcached-1.4.17
./configure --prefix=/usr/local/memcache
make
make install
```

```
useradd -r -s /sbin/nologin memcache
添加 memcache 用户，此用户不用登录，不设置密码
/usr/local/memcache/bin/memcached -umemcache &
启动 memcache 服务，并设置为后台运行
netstat -an | grep :11211
检查 memcache 是否正常启动，并监听了 11211 端口
```

## 18. 安装 phpMyAdmin

```
cp -a /lamp/phpMyAdmin-4.1.4-all-languages /usr/local/apache2/htdocs/phpmyadmin
cd /usr/local/apache2/htdocs/phpmyadmin
cp -a config.sample.inc.php config.inc.php
vim config.inc.php
$cfg['Servers'][$i]['auth_type'] = 'cookie';
$cfg['Servers'][$i]['auth_type'] = 'http';
设置 auth_type 为 http，即设置为 HTTP 身份认证模式（新增即可）
```

通过浏览器输入地址访问：http://Apache 服务器地址/phpmyadmin/index.php

用户名为 root , 密码为 MySQL 设置时指定的 root 密码 123456

## 19. 设置 Apache、MySQL、Memcache 开机自启

借助系统自带脚本/etc/rc.local，此脚本开机后会自动加载，我们可以将源码安装的服务启动命令写入该脚本，间接实现开机自启动

```
vi /etc/rc.local
/usr/local/apache2/bin/apachectl start
/usr/local/mysql/bin/mysqld_safe --user=mysql &
/usr/local/memcache/bin/memcached -umemcache &
```

## 20. 项目迁移：

- 1、把 php 项目拷贝到网站默认目录下：/usr/local/apache2/htdocs/\*\*
- 2、使用 phpMyAdmin 创建网站所需数据库

注意事项：注意目录权限和归属，防止权限过大或者权限过小

切记：做完 LAMP 环境后保存一个快照，后面讲 Apache 要使用！



# web网站平台-LAMP

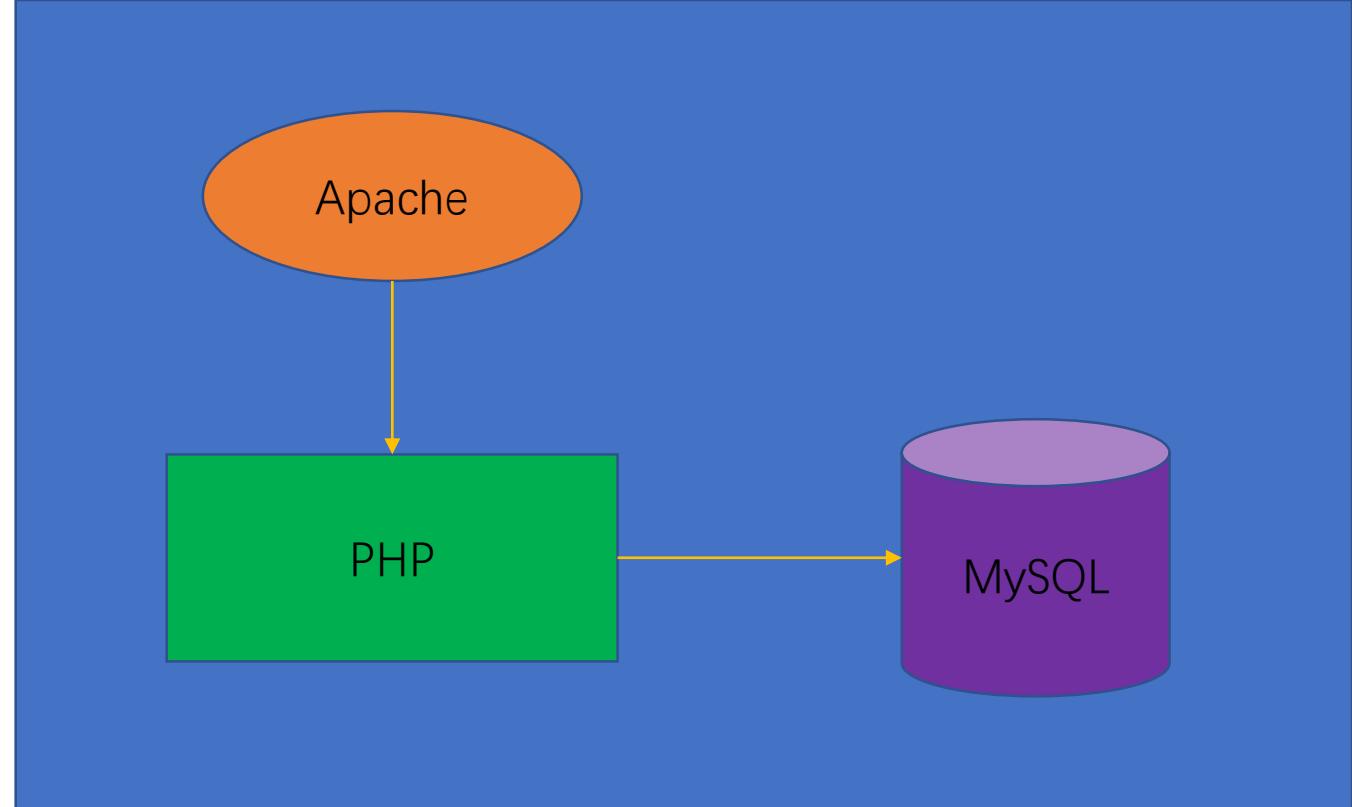
Linux + Apache + MySQL + PHP

Apache: 实现网页共享传输

MySQL: 实现数据存储

PHP: 实现页面解析的解析器

浏览器: 仅能解析简单的HTML语言, 无法直接解析PHP语言



部署方式:

yum安装:

优点: 安装部署便捷, 快速

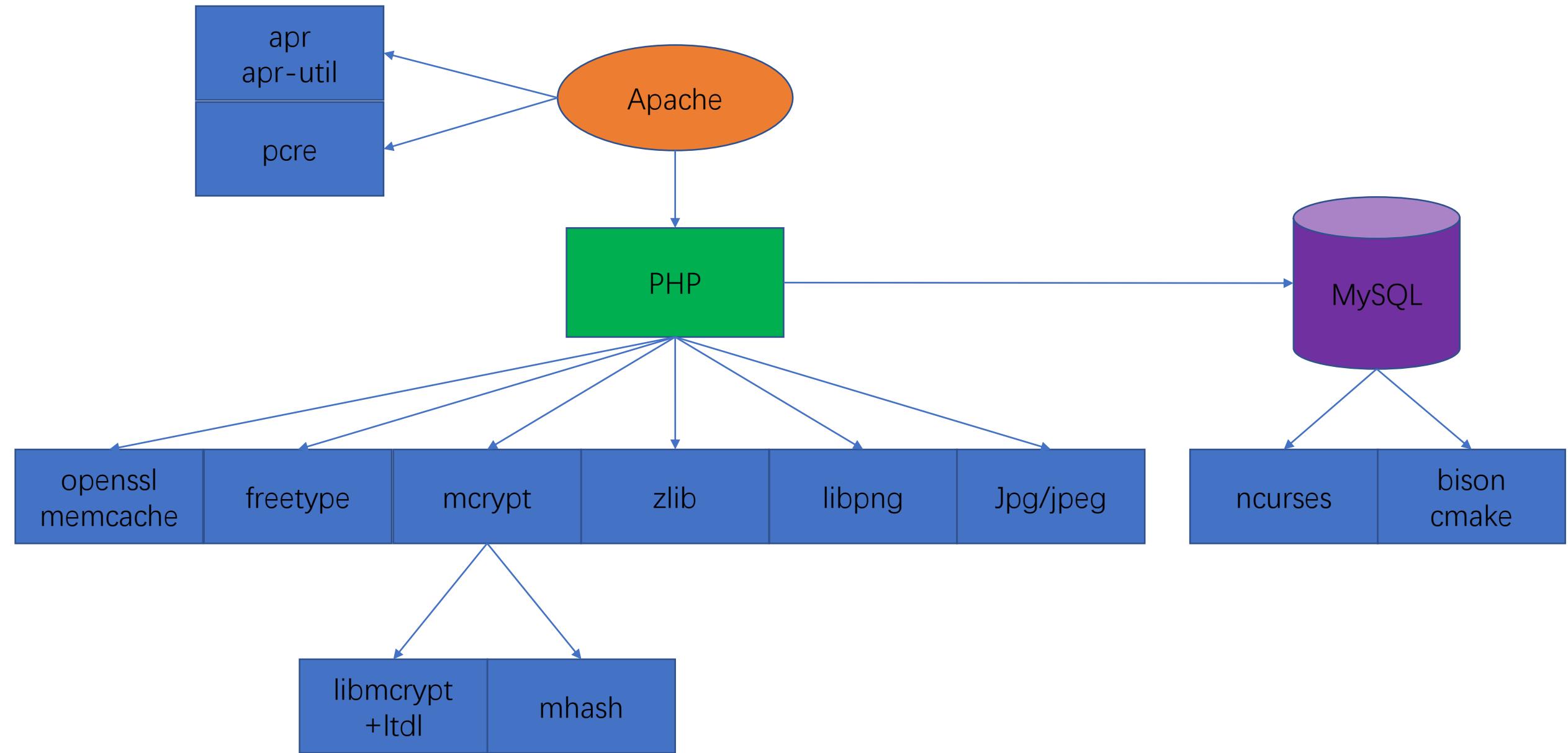
缺点: 软件版本固定, 且版本较低

源码安装:

优点: 版本可自选, 可自定义性强

缺点: 难度较大, 维护相对困难

# LAMP软件关系图谱



## 8. Web 服务器-Apache

### 一. 讲在 Apache 之前

**HTML语言：**超文本标记语言，使用html语言编写的文本叫超文本，“超文本”就是指页面内可以包含图片、链接，甚至音乐、程序等非文字元素。

**HTTP协议：**超文本传输协议

HTTP使用统一资源标识符（URL）来建立连接和传输数据。是一个基于TCP/IP通信协议来传递数据的协议，属于应用层协议。

**URL：**统一资源定位符

统一资源定位符是对可以从互联网上得到的资源的位置和访问方法的一种简洁的表示，是互联网上标准资源的地址。

格式：

http://www.atguigu.com:80/image/a.jpg

知识拓展：

**URI：**统一资源标志符，URI与URL都是定位资源位置的，就是表示这个资源的位置信息，就像经纬度一样可以表示你在世界的哪个角落。URI是一种宽泛的含义更广的定义，而URL则是URI的一个子集，就是说URL是URI的一部分。

### 二. Apache 详解

#### 1. 概述

Apache是世界使用排名第一的Web服务器软件。它可以运行在几乎所有广泛使用的计算机平台上，由于其跨平台和安全性被广泛使用，是最流行的Web服务器端软件之一。它快速、可靠并且可通过简单的API扩充，将Perl/Python/php等解释器编译到服务器中。

Apache有多种产品，可以支持SSL技术，支持多个虚拟主机。Apache是以进程为基础的结构，进程要比线程消耗更多的系统开支，不太适合于多处理器环境，因此，在一个Apache Web站点扩容时，通常是增加服务器或扩充群集节点而不是增加处理器。到目前为止Apache仍然是世界上用的最多的Web服务器，市场占有率达60%左右。

#### 2. 工作模式

Apache一共有3种稳定的MPM模式(MPM: 多进程处理模块)，它们分别是 prefork、worker、event  
**prefork 工作模式**



Apache在启动之初，就预先fork一些子进程，然后等待请求进来。之所以这样做，是为了减少频繁创建和销毁进程的开销。每个子进程只有一个线程，在一个时间点内，只能处理一个请求。

**优点：**成熟稳定，兼容所有新老模块。同时，不需要担心线程安全的问题。

**缺点：**一个进程相对占用更多的系统资源，消耗更多的内存。而且，它并不擅长处理高并发请求。

#### worker 工作模式

使用了多进程和多线程的混合模式。它也预先fork了几个子进程(数量比较少)，然后每个子进程创建一些线程，同时包括一个监听线程。每个请求过来，会被分配到1个线程来服务。线程比起进程会更轻量，因为线程通常会共享父进程的内存空间，因此，内存的占用会减少一些。在高并发的场景下，因为比起prefork有更多的可用线程，表现会更优秀一些。

**优点：**占据更少的内存，高并发下表现更优秀。

**缺点：**必须考虑线程安全的问题。

#### event 工作模式

它和worker模式很像，最大的区别在于，它解决了keep-alive场景下，长期被占用的线程的资源浪费问题。event MPM中，会有一个专门的线程来管理这些keep-alive类型的线程，当有真实请求过来的时候，将请求传递给服务线程，执行完毕后，又允许它释放。这样增强了高并发场景下的请求处理能力。

HTTP采用keepalive方式减少TCP连接数量，但是由于需要与服务器线程或进程进行绑定，导致一个繁忙的服务器会消耗完所有的线程。Event MPM是解决这个问题的一种新模型，它把服务进程从连接中分离出来。在服务器处理速度很快，同时具有非常高的点击率时，可用的线程数量就是关键的资源限制，此时Event MPM方式是最有效的，但不能在HTTPS访问下工作。

#### 查看方式：

```
httpd -V | grep -i "server mpm"
```

#### 指定方式：

在编译时，在选项中指定，--with-mpm=xxx

## 3. 相关文件保存位置

#### 配置文件位置：

源码包安装： PREFIX/etc/httpd.conf (主配置文件)

PREFIX/etc/extr/\*.conf (子配置文件)

rpm包安装： /etc/httpd/conf/httpd.conf

#### 网页文件位置：

源码包安装： PREFIX/htdocs/

rpm包安装： /var/www/html/

#### 日志文件位置：

源码包安装： PREFIX/logs/

rpm包安装： /var/log/httpd/



## 4. 配置文件详解

注意：apache配置文件严格区分大小写

### 针对主机环境的基本配置参数

```
ServerRoot /usr/local/apache2 #apache主目录
Listen :80 #监听端口
LoadModule php7 #加载的相关模块
User
Group #用户和组
ServerAdmin
ServerName #服务器名（没有域名解析时，使用临时解析。默认不开启）
ErrorLog "logs/error_log" #服务器错误日志
CustomLog "logs/access_log" common #访问记录日志
DirectoryIndex index.html index.php #默认网页文件名,优先级顺序
Include etc/extr/httpd-vhosts.conf #子配置文件中内容也会加载生效
```

### 主页目录及权限

```
DocumentRoot "/usr/local/apache2/htdocs"
#网页文件存放目录（默认）

<Directory "/usr/local/apache2/htdocs">
#定义指定目录的权限

 Options Indexes FollowSymLinks
 None #没有任何额外权限
 All #所有权限（除去MultiViews以外）
 Indexes #浏览权限（当此目录下没有默认网页文件时，显示目录内容）

 FollowSymLinks #准许软连接到其他目录
 MultiViews #准许文件名泛匹配（需要手动开启模块才有效negotiation）

 AllowOverride None
#定义是否允许目录下.htaccess文件中的权限生效

 None #.htaccess中权限不生效
 All #文件中所有权限都生效
 AuthConfig #文件中，只有网页认证的权限生效
 Require all granted (denied) #访问控制列表

</Directory>

<IfModule dir_module> #此标签用来指定访问到指定目录时自动加载哪个页面文件
 DirectoryIndex index.php index.html #可以写多个，但是有优先级之分
</IfModule>
```



## 5. Apache 实验

**实验环境：**建议使用之前搭建好的 lamp 环境进行试验测试

### 1) Apache 的目录别名

当 apache 接受请求时，在默认情况下会将 DocumentRoot 目录中的文件送到客户端，如果想将某一不在 DocumentRoot 目录中的文件共享到网站上，并希望将它们留在本来位置而不需要进行移动的话，处理这种情况可以通过建立别名的方式将 URL 指向特定的目录

#### 1、编辑主配置文件

```
vim /usr/local/apache2/conf/httpd.conf
Include etc/extra/httpd-autoindex.conf #去掉注释，开启调用子配置文件
```

#### 2、编辑子配置文件

```
vim /usr/local/apache2/conf/extra/httpd-autoindex.conf
alias /icons/ "/usr/local/apache2/icons/"
结构：别名“真实目录” #真实目录的结尾要有/，否则报错
<Directory "/usr/local/apache2/icons">
 Options Indexes FollowSymLinks
 AllowOverride None
 Require all granted
</Directory>
```

可以根据模板编写一个自己需要的目录别名

### 2) Apache 的用户认证

有时候，我们需要给一些特殊的访问设置一个用户认证机制，增加安全。比如我们的个人网站，一般都是有一个管理后台的，虽然管理后台本身就有密码，但我们为了更加安全，可以再设置一层用户身份认证。

#### 1、编辑配置文件

```
vim /usr/local/apache2/etc/httpd.conf
在需要进行登录认证的目录标签中加入如下配置：
<Directory "/usr/local/apache2/htdocs/admin"> #声明被保护目录
 Options Indexes FollowSymLinks
 AllowOverride All #开启权限认证文件. htaccess
 Require all granted
</Directory>
```

#### 2、在指定目录下创建权限文件



切换到/usr/local/apache2/htdocs/admin，创建 .htaccess 文件，并添加下面的内容

```
vi .htaccess
AuthName "Welcome to atguigu"
#提示信息
AuthType basic
#加密类型
AuthUserFile /usr/local/apache2/htdocs/admin/apache.passwd
#密码文件，文件名自定义。（使用绝对路径）
require valid-user
#允许密码文件中所有用户访问
```

### 3、建立密码文件，加入允许访问的用户。（此用户和系统用户无关）

```
htpasswd -c /usr/local/apache2/htdocs/admin/apache.passwd test1
-c 建立密码文件，只有添加第一个用户时，才能-c
htpasswd -m /usr/local/apache2/htdocs/admin/apache.passwd test2
-m 再添加更多用户时，使用-m 参数
注意： htpasswd 该命令是 httpd 的命令，需要绝对路径
```

### 4、重启 apache 服务

```
/usr/local/apache2/bin/apachectl -t
/usr/local/apache2/bin/apachectl stop
/usr/local/apache2/bin/apachectl start
```

先检查配置是否正确，然后通过浏览器输入要访问的资源时就会提示输入密码了。

## 3) 虚拟主机（重点）

虚拟主机，也叫“网站空间”，就是把一台运行在互联网上的物理服务器划分成多个“虚拟”服务器。虚拟主机技术极大的促进了网络技术的应用和普及。同时虚拟主机的租用服务也成了网络时代的一种新型经济形式。

虚拟主机的分类：

基于 IP 的虚拟主机：一台服务器，多个 ip，搭建多个网站

基于端口的虚拟主机：一台服务器，一个 ip，搭建多个网站，每个网站使用不同端口访问

基于域名的虚拟主机：一台服务器，一个 ip，搭建多个网站，每个网站使用不同域名访问

实验准备：

### 1. 域名解析：准备两个域名

www.sohu.com

www.sina.com

#使用本地 hosts 文件进行解析

### 2. 网站主页目录规划

在/htdocs/目录下分别创建 sohu 和 sina 两个目录，并在新建目录内创建 index.html 文件（分别写



不一样的内容)

#### 试验步骤:

##### 1. 修改主配置文件开启文件关联

```
vi /usr/local/apache2/etc/httpd.conf
Include etc//extra/httpd-vhosts.conf #此行取消注释
```

##### 2. 编辑子配置文件，编写虚拟主机标签

```
vi /usr/local/apache2/etc/extra/httpd-vhosts.conf
添加下方内容，有几个虚拟主机就写几组（添加之前先把原先存在的示例删除掉）
<Directory "/usr/local/apache2/htdocs/sina">
 Options Indexes FollowSymLinks
 AllowOverride None
 Require all granted
</Directory>
#目录权限标签根据需要自行添加
<VirtualHost 192.168.88.10:80> #虚拟主机标签
 ServerAdmin webmaster@sina.com #管理员邮箱
 DocumentRoot "/usr/local/apache2/htdocs/sina" #网站主目录
 ServerName www.sina.com #完整域名
 ErrorLog "logs/sina-error_log" #错误日志
 CustomLog "logs/sina-access_log" common #访问日志
</VirtualHost>
```

##### 3. 重启服务，验证结果

Windows 下：浏览器下输入两个不同的域名验证网页内容（提前修改 windows 的 hosts 文件）

Linux 下：通过 elinks/curl 命令验证：elinks/curl URL 地址（提前修改 windows 的 hosts 文件）

## 4) 域名跳转

一个站点难免会有多个域名，而多个域名总得有一个主次，比如我的网站可以用两个域名访问：

www.sina.com 和 www.sohu.cn 但大家发现不管我用哪个域名访问，最终都会跳转到 www.sina.com 上来。这个行为就叫做域名跳转，状态码：301 是永久跳转，302 是临时跳转，网站上一定要设置为 301，这样对搜索引擎是比较友好的。

#### 实验条件:

1. 虚拟主机能正常访问
2. 打开主配置文件开启重写模块

```
LoadModule rewrite_module modules/mod_rewrite.so #取消注释
```

#### 实验步骤:

##### 1. 修改虚拟主机配置文件

```
vi */extra/httpd-vhosts.conf
<Directory "/usr/local/apache2/htdocs/sohu">
 Options Indexes FollowSymLinks
```



```
AllowOverride All
Require all granted
</Directory>
```

## 2. 创建规则匹配文件

```
vi */.htaccess
在指定的网站目录下创建文件，并添加以下内容
RewriteEngine on
开启rewrite功能
RewriteCond %{HTTP_HOST} ^www.sohu.com
把以www.sina.com开头的内容赋值给HTTP_HOST变量
RewriteRule ^(.*)$ http://www.sina.com/$1 [R=permanent,L]
^(.*)$ 指代客户端要访问的资源
$1 把.*所指代的内容赋值到$1变量中
R=permanent 永久重定向 = 301
L 指定该规则为最后一条生效的规则，以后的不再生效
```

## 3. 重启服务器并测试

```
/usr/local/apache2/bin/apachectl -t
/usr/local/apache2/bin/apachectl stop
/usr/local/apache2/bin/apachectl start
```

通过上述测试，发现无论是 sina 或 sohu 最终都是访问到 www.sina.com 域名上来则试验成功

## 5) Apache+openssl 实现 https（重点）

HTTPS（全称：Hypertext Transfer Protocol Secure，超文本传输安全协议），是以安全为目标的 HTTP 通道，简单讲是 HTTP 的安全版。即 HTTP 下加入 SSL 层，用于安全的 HTTP 数据传输。这个系统被内置于浏览器中，提供了身份验证与加密通讯方法。现在它被广泛用于万维网上安全敏感的通讯，例如交易支付方面。

### 1. 准备工作：

检查 Apache 是否支持 SSL，检查相应模块是否安装，若安装则将模块启用

模块存放目录：/usr/local/apache2/modules

检查模块是否启用：apachectl -M

### 2. CA 证书申请：

a. openssl genrsa -out ca.key 1024

#建立服务器私钥，生成 RSA 密钥

b. openssl req -new -key ca.key -out atguigu.csr

#需要依次输入国家，地区，城市，组织，组织单位，Email 等信息。最重要的是有一个 common name，可以写你的名字或者域名。如果为了 https 申请，这个必须和域名吻合，否则会引发浏览器警报。生成的 csr 文件交给 CA 签名后形成服务端自己的证书

c. openssl x509 -req -days 365 -sha256 -in atguigu.csr -signkey ca.key -out atguigu.crt

#使用 CA 服务器签发证书，设置证书的有效期等信息

注意 1：生成完秘钥和证书文件后，将文件存放在 Apache 的安装目录下的 cert 目录下

注意 2：在生产环境中必须要在 https 证书厂商注册（否则浏览器不识别）



### 3. 配置文件修改:

- a. 调用 ssl 模块，并启用 ssl 独立配置文件

```
vim /usr/local/apache2/etc/httpd.conf
LoadModule ssl_module modules/mod_ssl.so #取消注释
Include etc/extra/httpd-ssl.conf #取消注释
```

- b. 修改 conf/extra/httpd-ssl.conf 配置文件，调用证书等文件

```
#添加 SSL 协议支持协议，去掉不安全的协议
SSLPotocol all -SSLv2 -SSLv3
#修改加密套件如下
SSLCipherSuite HIGH:!RC4:!MD5:!aNULL:!eNULL:!NULL:!DH:!EDH:!EXP:+MEDIUM
SSLHonorCipherOrder on
#证书公钥配置（签字的）
SSLCertificateFile cert/atguigu.crt
#证书私钥配置
SSLCertificateKeyFile cert/ca.key
```

- c. 修改主配置文件，添加虚拟主机

```
<VirtualHost _default_:443>
 # DocumentRoot 目录位置要和 httpd.conf 里面的一致
 DocumentRoot "/usr/local/apache2/htdocs"
 ServerName localhost:443
 SSLCertificateFile cert/atguigu.crt
 SSLCertificateKeyFile cert/ca.key
 SSLCertificateChainFile cert/atguigu.crt
</VirtualHost>
```

### 4. 结果验证:

```
apachectl -t #检查配置文件语法
apachectl restart #重启 apache，并测试是否可以使用 https 访问
```

**报错提示：** AH00526: Syntax error on line 78 of /usr/local/apache2/etc/extra/httpd-ssl.conf:

SSLSessionCache: 'shmcb' session cache not supported (known names: ). Maybe you need to load the appropriate socache module (mod\_socache\_shmcb?).

**解决方案：** 要么不调用此模块，要么让调用的模块加载上

### 5. 强制跳转 https:

有些时候为了安全，网站不允许使用 http 访问，仅允许使用 https 访问，目的是为了更加安全  
在 http 部分的目录权限标签中添加一下内容

```
<Directory "/usr/local/apache2/htdocs">

 RewriteEngine on #开启转发规则
 RewriteCond %{SERVER_PORT} !^443$ #检查访问端口只要目标不是443的
 RewriteRule ^(.*)? https:// %{SERVER_NAME} /$1 [R=301, L] #全都使用https重新访问
</Directory>
```



在做后面实验时为了更加方便理解，我们可以先把 https 关闭掉

需要关闭：跳转&虚拟主机&ssl 配置文件调用

## 6) Apache 日志切割

我们每访问一次网站，那么就会记录若干条日志。如果日志不去管理，时间长了日志文件会越来越大，如何避免产生大的日志文件？其实 apache 有相关的配置，使日志按照我们的需求进行归档，比如每天一个新日志，或者每小时一个新的日志。

1. 首先简单设置日志的路径名称

```
vim /usr/local/apache2/etc/httpd.conf
```

编辑添加内容如下：

```
ErrorLog "logs/error.log"
CustomLog "logs/access.log" combined
```

指定了日志存放在/usr/local/apache2/logs 目录下分别为 error.log 和 access.log，combined 为日志显示的格式，日志格式可以参考配置文件 httpd.conf 中格式的指定，如下：

```
LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\"\" combined
LogFormat "%h %l %u %t \"%r\" %>s %b" common
```

2. 设置 apache 日志分割

同样编辑配置文件 httpd.conf

```
ErrorLog "|/usr/local/apache2/bin/rotatelogs -1 /usr/local/apache2/logs/error_%Y%m%d.log
86400"
CustomLog "|/usr/local/apache2/bin/rotatelogs -1
/usr/local/apache2/logs/access_%Y%m%d.log 86400" combined
```

**注意 1：**以上仅为两条命令（一条错误日志，一条访问日志），路径太长写不开  
**注意 2：**若开启了 https，则需要修改 http-ssl.conf 配置文件中的日志记录条目

ErrorLog 是错误日志，CustomLog 是访问日志。|就是管道符，意思是把产生的日志交给 rotatelogs 这个工具，而这个工具就是 apache 自带的切割日志的工具。-1 的作用是校准时区为 UTC，也就是北京时间。86400，单位是秒，正好是一天，那么日志会每天切割一次。而最后面的 combined 为日志的格式，在 httpd.conf 中有定义。

## 7) 不记录指定文件类型的日志

如果一个网站访问量特别大，那么访问日志就会很多，但有一些访问日志我们其实是可以忽略掉的，比如网站的一些图片，还有 js、css 等静态对象。而这些文件的访问往往是巨量的，而且即使记录这些日志也没有什么用，那么如何忽略不记录这些日志呢？

## 1、配置日志不记录图片的访问

```
vim /usr/local/apache2/conf/httpd.conf
```

相关配置为：

```
SetEnvIf Request_URI ".*\.gif$" image-request
SetEnvIf Request_URI ".*\.jpg$" image-request
SetEnvIf Request_URI ".*\.png$" image-request
SetEnvIf Request_URI ".*\.bmp$" image-request
SetEnvIf Request_URI ".*\.swf$" image-request
SetEnvIf Request_URI ".*\.js$" image-request
SetEnvIf Request_URI ".*\.css$" image-request
CustomLog "|/usr/local..._%Y%m%d.log 86400" combined env=!image-request
```

说明：在原来的访问日志配置基础上，增加了一些 image-request 的定义，比如把 gif、jpg、bmp、swf、js、css 等结尾的全标记为 image-request，然后在配置日志后加一个标记 env=!image-request，表示取反。

## 8) Apache 配置静态缓存

所说的静态文件指的是图片、js、css 等文件，用户访问一个站点，其实大多数元素都是图片、js、css 等，这些静态文件其实是会被客户端的浏览器缓存到本地电脑上的，目的就是为了下次再请求时不再去服务器上下载，这样就加快了速度，提高了用户体验。但这些静态文件总不能一直缓存，它总有一些时效性，那么就得设置这个过期时间。

### 1、配置静态缓存

```
vim /usr/local/apache2/conf/httpd.conf
<IfModule mod_expires.c> #此模块默认未启用，请手动启用
 ExpiresActive on
 ExpiresByType image/gif "access plus 1 days"
 ExpiresByType image/jpeg "access plus 24 hours"
 ExpiresByType image/png "access plus 24 hours"
 ExpiresByType text/css "now plus 2 hours"
 ExpiresByType application/x-javascript "now plus 2 hours"
 ExpiresByType application/javascript "now plus 2 hours"
 ExpiresByType application/x-shockwave-flash "now plus 2 hours"
 ExpiresDefault "now plus 0 min"
</IfModule>
```

或者使用 mod\_headers 模块实现：该模块默认启用

```
<IfModule mod_headers.c>
 # htm,html,txt 类的文件缓存一个小时
 <filesmatch "\.(htm|html|txt)$">
```



```
header set cache-control "max-age=3600"
</filesmatch>

css, js, swf 类的文件缓存一个星期
<filesmatch "\.(css|js|swf)$">
 header set cache-control "max-age=604800"
</filesmatch>

jpg, gif, jpeg, png, ico, flv, pdf 等文件缓存一年
<filesmatch "\.(ico|gif|jpg|jpeg|png|flv|pdf)$">
 header set cache-control "max-age=29030400"
</filesmatch>
</IfModule>
```

说明：这里的时间单位可以 days、hours 甚至是 min，两种不同的方法，上面使用的是 mod\_expires，而下面用的是 mod\_headers，要想使用这些模块，必须要事先已经支持。如何查看是否支持，使用命令：

```
/usr/local/apache2/bin/apachectl -M
```

## 2、重启服务器并验证

```
/usr/local/apache2/bin/apachectl -t
/usr/local/apache2/bin/apachectl stop
/usr/local/apache2/bin/apachectl start
```

验证：

```
curl -x127.0.0.1:80 'http://www.sohu.com/image/a.jpg' -I
HTTP/1.1 200 OK
Date: Wed, 26 Oct 2016 03:51:26 GMT
Server: Apache/2.2.31 (Unix) PHP/5.5.38
Last-Modified: Tue, 31 May 2016 03:08:36 GMT
ETag: "46891b-16b-5341ab0597500"
Accept-Ranges: bytes
Content-Length: 363
Cache-Control: max-age=86400
Expires: Thu, 27 Oct 2016 03:51:26 GMT
Content-Type: image/jpg
```

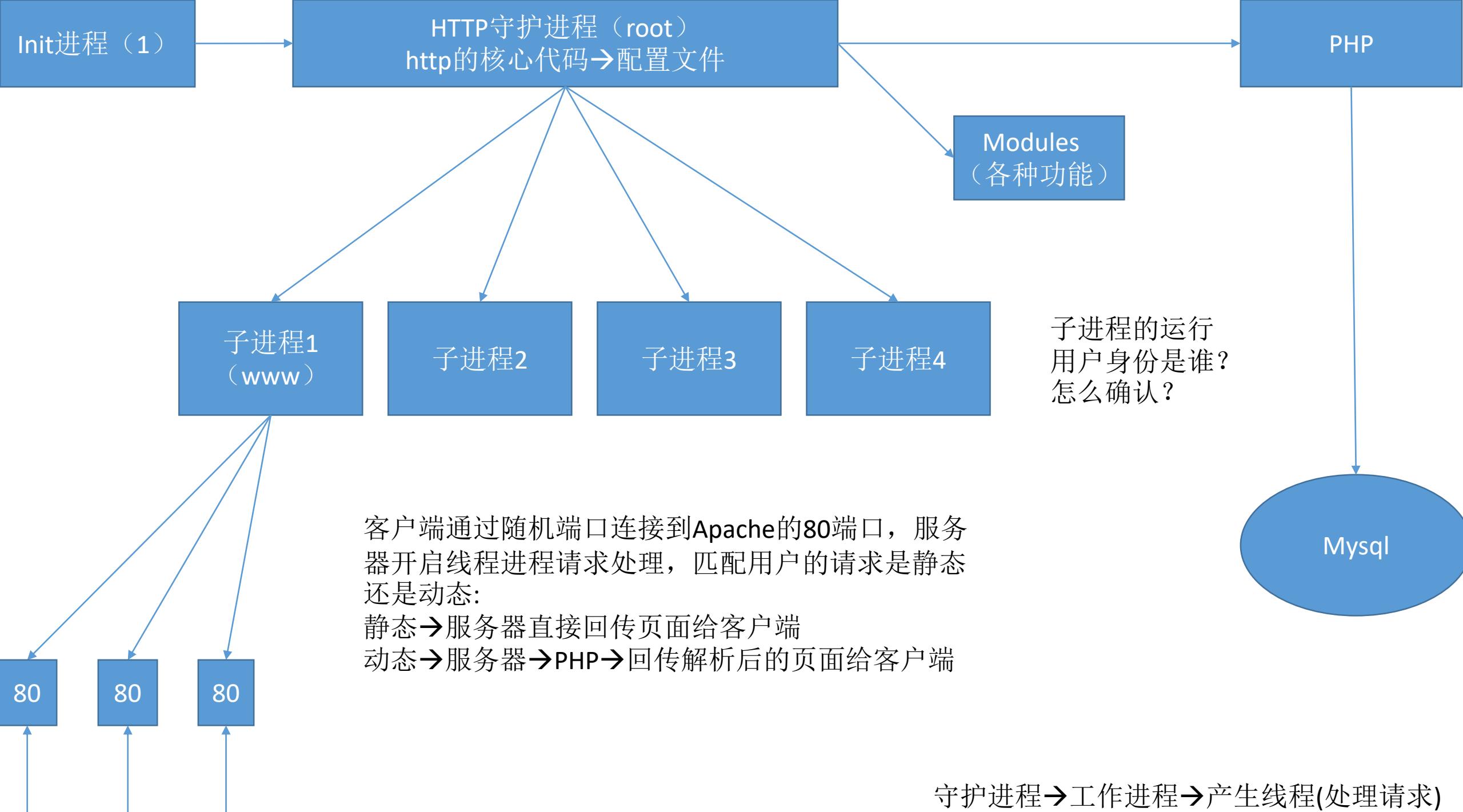
## 9) 禁止解析 PHP

某个目录下禁止解析 PHP，这个很有作用，我们做网站安全的时候，这个用的很多，比如某些目录可以上传文件，为了避免上传的文件有木马，所以我们禁止这个目录下面的访问解析 PHP。

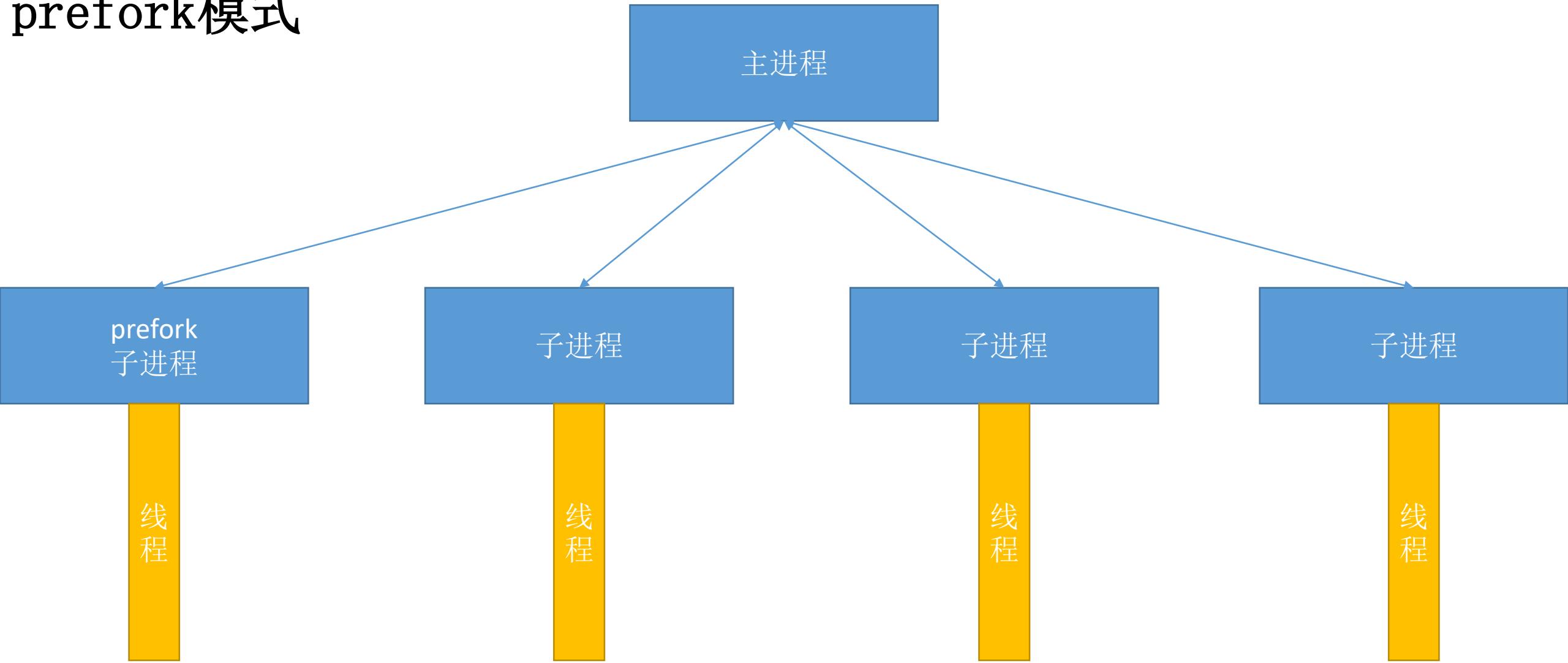
### 配置禁止解析 php

```
<Directory /usr/local/apache2/htdocs/data>
 php_admin_flag engine off
 <filesmatch "(.*).php">
 Order deny,allow
 Deny from all
 </filesmatch>
</Directory>
```

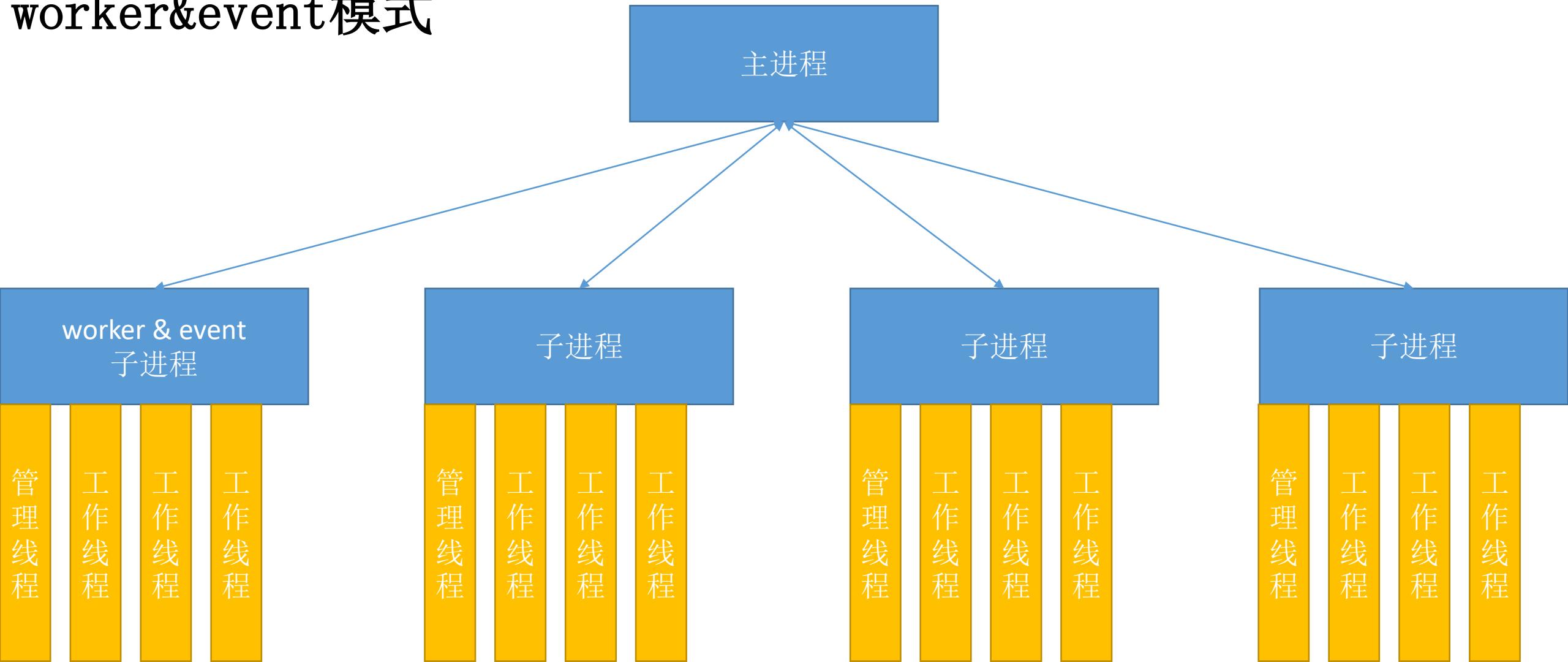




# prefork模式



# worker&event模式



## keep-alive

在http早期，每个http请求都要求打开一个tcp socket连接，并且使用一次之后就断开这个tcp连接。

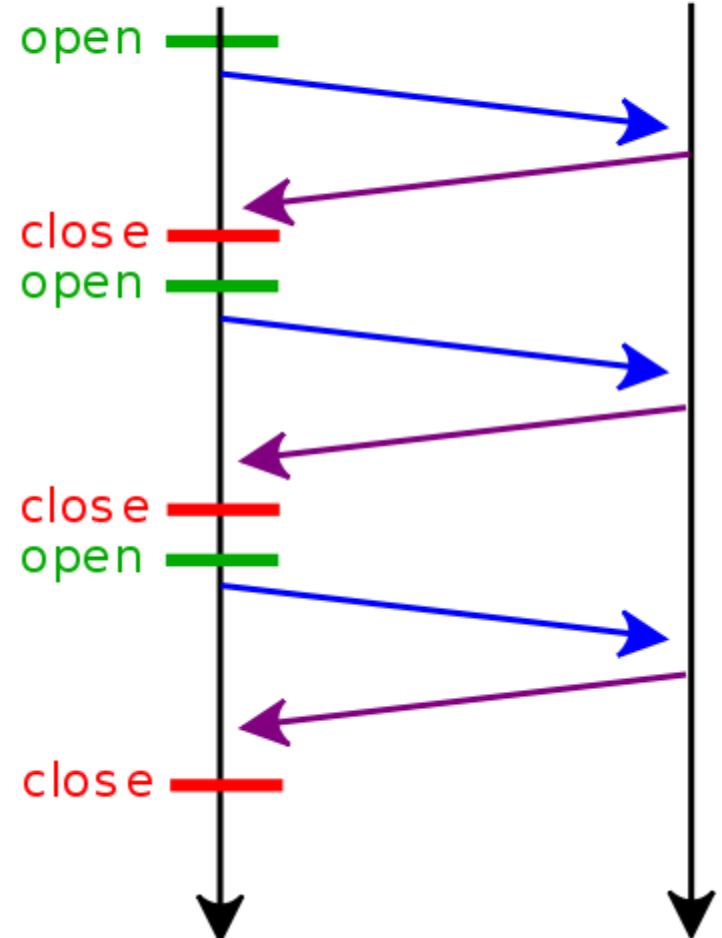
当httpd守护进程发送完一个响应后，理应马上主动关闭相应的tcp连接，设置keepalive\_timeout后，httpd守护进程会想说：“再等等吧，看看浏览器还有没有请求过来”，这一等，便是keepalive\_timeout时间。如果守护进程在这个等待的时间里，一直没有收到浏览器发过来http请求，则关闭这个http连接。

使用keep-alive可以改善这种状态，即在一次TCP连接中可以持续发送多份数据而不会断开连接。通过使用keep-alive机制，可以减少tcp连接建立次数，也意味着可以减少TIME\_WAIT状态连接，以此提高性能和提高httpd服务器的吞吐率（更少的tcp连接意味着更少的系统内核调用）。

但是，长时间的tcp连接容易导致系统资源无效占用。配置不当的keep-alive，有时比重复利用连接带来的损失还更大。所以，正确地设置keep-alive timeout时间非常重要。

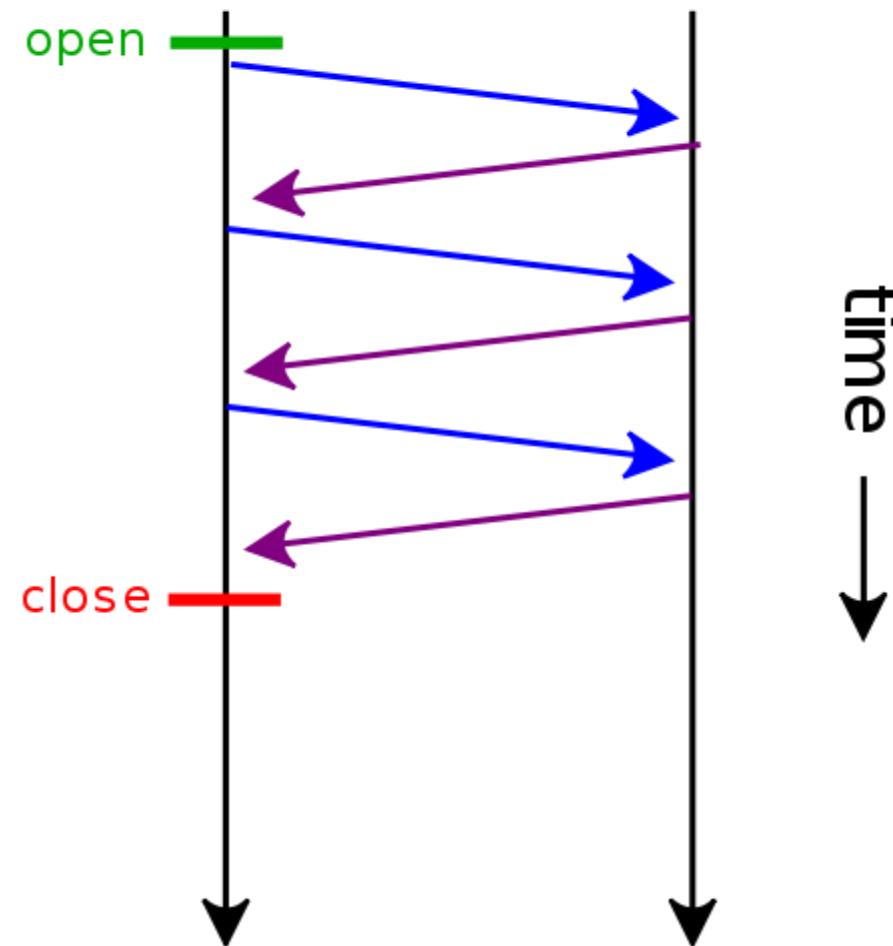
## multiple connections

client                    server



## persistent connection

client                    server

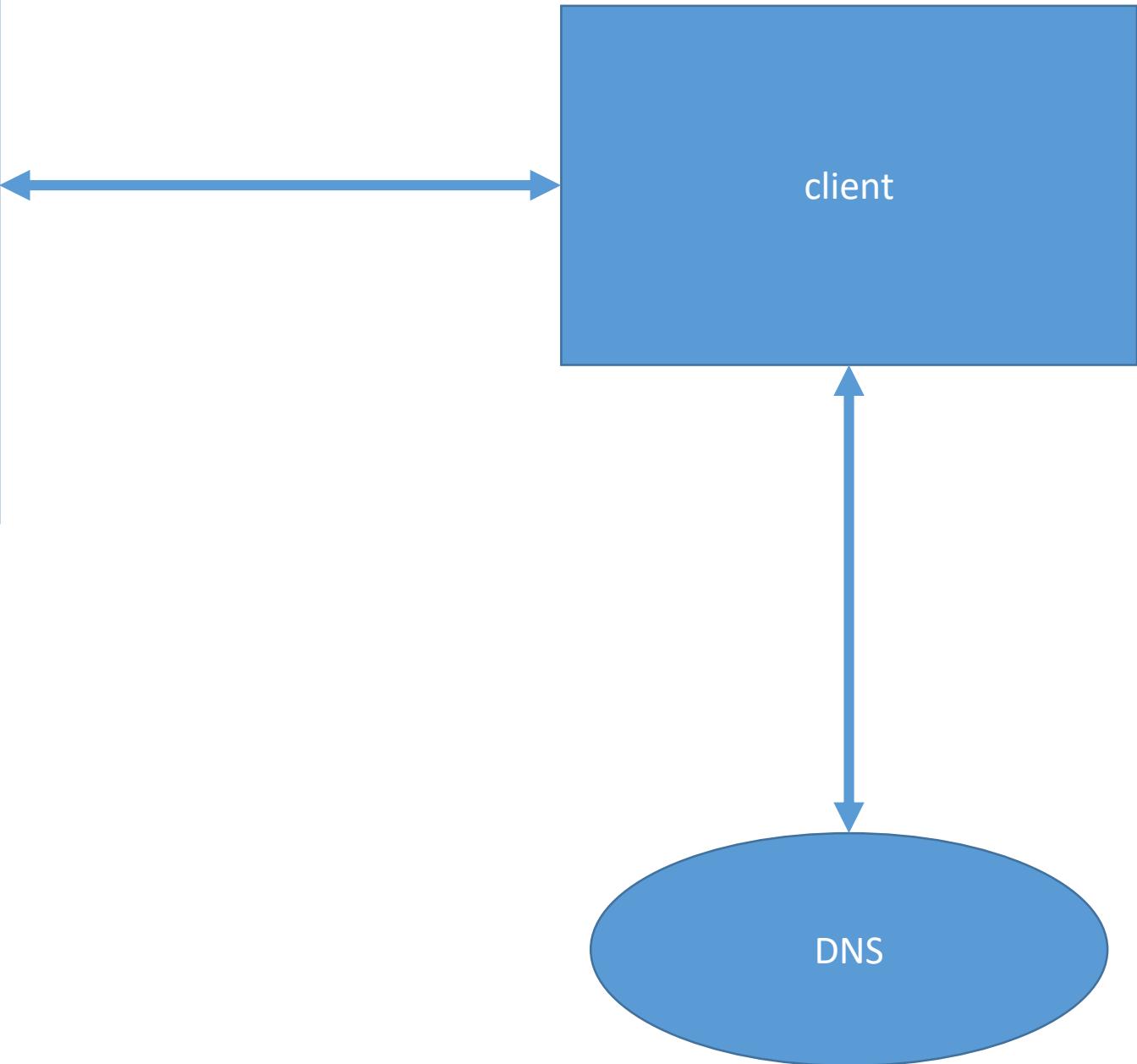


虚拟机：

    虚拟硬件，真正操作系统

虚拟主机：

    虚拟空间



**注意1：**服务器端，是通过客户端向服务器端发送的数据包中包含的域名，进行识别和区分，以此确定客户端所访问的时位于哪一个域名下的网页资源

**注意2：**当搭建了虚拟主机后，主配置文件中的单独的网站配置，无法正常生效

## 9. web 平台搭建-LNMP

### 一、准备工作

#### 1. 环境要求:

操作系统: CentOS 6.X 64 位

关闭 SELinux 和 iptables 防火墙

此次试验环境使用网络 yum 源, 保证系统能正常连接互联网

#### 2. 网络 yum 源:

先将系统自带的 yum 配置文件移除或者删除, 然后下载以下两个配置文件

官方基础: <http://mirrors.163.com/.help/CentOS6-Base-163.repo>

<http://mirrors.aliyun.com/repo/Centos-6.repo>

epel 拓展: <http://mirrors.aliyun.com/repo/epel-6.repo>

下载完成后, 需要使用命令清除掉原有的 yum 缓存, 使用新的配置文件建立缓存

```
yum clean all #清除掉原有缓存列表
yum makecache #建立新的缓存列表
yum update #将所有能更新的软件更新（非必选）
```

#### 3. 安装编译工具和依赖软件包:

```
yum -y install gcc* pcre-devel openssl-devel zlib-devel ncurses-devel cmake bison
```

```
libxml2-devel libpng-devel
```

#### 4. Nginx、MySQL、PHP 三大软件的源码包下载地址:

Nginx: <http://nginx.org/en/download.html>

MySQL: <https://dev.mysql.com/downloads/mysql/>

PHP: <http://www.php.net/>

#### 版本选用:

Nginx: 1.12.\* #选用软件的稳定版即可

Mysql: 5.5.\* #5.5 以上版本需要 1G 以上的内存, 否则无法安装

PHP: 5.6.\* #LAMP 中我们使用的是 php7, 此次使用 php5

注意: 每次安装 LNMP 时, 软件包的小版本都不一样, 官方会对其大版本下的小版本进行覆盖式更新, 本文内部分链接会失效, 切记按照下载版本进行安装。

### 二、源码软件包安装

#### 1. Nginx

Nginx 是一款轻量级的 Web 服务器/反向代理服务器及电子邮件(IMAP/POP3)代理服务器, 在 BSD-like 协议下发行。其特点是占有内存少, 并发能力强。

##### 1.1 下载 Nginx 源码包

```
wget http://nginx.org/download/nginx-1.12.2.tar.gz
```

### 1.2 创建用于运行 Nginx 的用户

```
useradd -r -s /sbin/nologin nginx
```

### 1.3 解压缩 Nginx 并安装

```
./configure --prefix=/usr/local/nginx --user=nginx --group=nginx --with-http_stub_status_module
--with-http_ssl_module
make
make install
```

### 1.4 上传编写好的 nginx 启动管理脚本（见文本尾部）

## 2. MySQL

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

选择: MySQL Community Server 5.5 »

选择: Select Version: 按照自己要求选择

Select Operating System: Source Code

Select OS Version: Generic Linux

格式: mysql-N.N.NN.tar.gz

```
wget https://cdn.mysql.com//Downloads/MySQL-5.5/mysql-5.5.62.tar.gz
```

### 2.1 创建用于运行 Mysql 的用户:

```
useradd -r -s /sbin/nologin mysql
```

### 2.2 解压缩 Mysql 并安装:

```
cmake -DCMAKE_INSTALL_PREFIX=/usr/local/mysql -DMYSQL_UNIX_ADDR=/tmp/mysql.sock
-DEXTRA_CHARSETS=all -DDEFAULT_CHARSET=utf8 -DDEFAULT_COLLATION=utf8_general_ci
-DWITH_MYISAM_STORAGE_ENGINE=1 -DWITH_INNOBASE_STORAGE_ENGINE=1
-DWITH_MEMORY_STORAGE_ENGINE=1 -DWITH_READLINE=1
-DENABLED_LOCAL_INFILE=1 -DMYSQL_USER=mysql -DMYSQL_TCP_PORT=3306
make
make install
ln -s /usr/local/mysql/bin/* /usr/local/bin
```

### 2.3 修改安装后的目录权限

```
cd /usr/local/mysql
chown -R root .
chown -R mysql data
```

### 2.4 生成 Mysql 配置文件

```
cp -a /lnmp/mysql-5.5.62/support-files/my-medium.cnf /etc/my.cnf
```



## 2.5 初始化，生成授权表

```
cd /usr/local/mysql #一定要先切换到此目录下，然后再执行下一步。
./scripts/mysql_install_db --user=mysql
初始化成功标志：两个 ok
```

## 2.6 生成 Mysql 的启动和自启动管理脚本

```
cd /lnmp/mysql-5.5.62/support-files
切换到 mysql 的源码解压缩目录下的 support-files
cp -a mysql.server /etc/init.d/mysqld
chmod +x /etc/init.d/mysqld

chkconfig --add mysqld
chkconfig mysqld on
service mysqld start|stop|restart
```

## 2.7 给 mysql 的 root 用户设置密码

```
mysqladmin -uroot password 123456
```

## 3. PHP

下载：<http://www.php.net/>

```
wget http://tw2.php.net/distributions/php-5.6.38.tar.gz
```

### 3.1 解压缩 PHP 并安装：

```
./configure --prefix=/usr/local/php/ --with-config-file-path=/usr/local/php/etc/
--with-mysqli=/usr/local/mysql/bin/mysql_config --enable-soap --enable-mbstring=all --enable-sockets
--with-pdo-mysql=/usr/local/mysql --with-gd --without-pecl --enable-fpm
make
make install
```

报错提示：若遇到 libpng.so not found .报错（老版本的 PHP 会出现此问题）

解决方案：

```
ln -s /usr/lib64/libpng.so /usr/lib
```

### 3.2 生成 php 配置文件

```
cp -a /lnmp/php-5.6.38/php.ini-production /usr/local/php/etc/php.ini
```

复制源码包内的配置文件到安装目录下，并改名即可

### 3.3 创建软连接，使用 php 相关命令是更方便

```
ln -s /usr/local/php/bin/* /usr/local/bin/
ln -s /usr/local/php/sbin/* /usr/local/sbin/
```



## 4. 配置 Nginx 连接 PHP（重难点）

### 4.1 nginx 连接 php 需要启动 php-fpm 服务

```
cd /usr/local/php/etc/
cp -a php-fpm.conf.default php-fpm.conf
生成 php-fpm 的配置文件，并修改指定参数
vim php-fpm.conf
修改指定条目的参数：
pid = run/php-fpm.pid
user = nginx
group = nginx
pm.start_servers = 2
pm.min_spare_servers = 1
pm.max_spare_servers = 3
启动时开启的进程数、最少空闲进程数、最多空闲进程数（默认值，未修改）
修改 Nginx 启动管理脚本：将 php-fpm 的注释取消掉即可
```

### 4.2 修改 Nginx 的配置文件，使其识别.php 后缀的文件

```
vim /usr/local/nginx/conf/nginx.conf
取消下列行的注释，并修改 include 选项的后缀为 fastcgi.conf，并注意每一行结尾的分号和大括号
#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; #修改为 fastcgi.conf
}
测试：
```

重启 Nginx 服务，创建 php 测试文件，访问并查看是否解析

### 4.3 修改 Nginx 配置文件，使其默认自动加载 php 文件

```
location / {
root html; #Nginx 的默认网页路径:PREFIX/html
index index.php index.html; #设置默认加载的页面，以及优先级
}
```

**附件：**建议使用时先复制到文本文件中查看是否有字符集问题

```
######Nginx 启动管理脚本#####
#!/bin/bash
#Author: liu
#chkconfig: 2345 99 33
```



```
#description: nginx server control tools

ngxc="/usr/local/nginx/sbin/nginx"
pidf="/usr/local/nginx/logs/nginx.pid"
ngxc_fpm="/usr/local/php/sbin/php-fpm"
pidf_fpm="/usr/local/php/var/run/php-fpm.pid"
case "$1" in
 start)
 $ngxc -t &> /dev/null
 if [$? -eq 0];then
 $ngxc
 $ngxc_fpm
 echo "nginx service start success!"
 else
 $ngxc -t
 fi
 ;;
 stop)
 kill -s QUIT $(cat $pidf)
 kill -s QUIT $(cat $pidf_fpm)
 echo "nginx service stop success!"
 ;;
 restart)
 $0 stop
 $0 start
 ;;
 reload)
 $ngxc -t &> /dev/null
 if [$? -eq 0];then
 kill -s HUP $(cat $pidf)
 kill -s HUP $(cat $pidf_fpm)
 echo "reload nginx config success!"
 else
 $ngxc -t
 fi
 ;;
 *)
 echo "please input stop|start|restart|reload."
 exit 1
esac
```

# web 平台搭建-LNMP (CentOS-7)

## 一、准备工作

### 1. 环境要求:

操作系统: CentOS 7.X 64 位

关闭 SELinux 和 iptables 防火墙

此次试验环境使用网络 yum 源, 保证系统能正常连接互联网

### 2. 网络 yum 源:

先将系统自带的 yum 配置文件移除或者删除, 然后下载以下两个配置文件

官方基础: <http://mirrors.163.com/.help/CentOS7-Base-163.repo>

<http://mirrors.aliyun.com/repo/Centos-7.repo>

epel 拓展: <http://mirrors.aliyun.com/repo/epel-7.repo>

下载完成后, 需要使用命令清除掉原有的 yum 缓存, 使用新的配置文件建立缓存

```
yum clean all #清除掉原有缓存列表
yum makecache #建立新的缓存列表
yum update #将所有能更新的软件更新（非必选）
```

### 3. 安装编译工具和依赖软件包:

```
yum -y install gcc gcc-c++ pcre-devel openssl-devel zlib-devel ncurses-devel cmake bison
libxml2-devel libpng-devel
```

### 4. Nginx、MySQL、PHP 三大软件的源码包下载地址:

Nginx: <http://nginx.org/en/download.html>

MySQL: <https://dev.mysql.com/downloads/mysql/>

PHP: <http://www.php.net/>

#### 版本选用:

Nginx: 1.12.\* #选用软件的稳定版即可

Mysql: 5.5.\* #5.5 以上版本需要 1G 以上的内存, 否则无法安装

PHP: 7.1.\* #我们使用的是 php7

注意: 每次安装 LNMP 时, 软件包的小版本都不一样, 官方会对其大版本下的小版本进行覆盖式更新, 本文内部分链接会失效, 切记按照下载版本进行安装。

## 二、源码软件包安装

### 1. Nginx

Nginx 是一款轻量级的 Web 服务器/反向代理服务器及电子邮件(IMAP/POP3)代理服务器, 在 BSD-like 协议下发行。其特点是占有内存少, 并发能力强。

#### 1.1 下载 Nginx 源码包

```
wget http://nginx.org/download/nginx-1.12.2.tar.gz
```

### 1.2 创建用于运行 Nginx 的用户

```
useradd -r -s /sbin/nologin nginx
```

### 1.3 解压缩 Nginx 并安装

```
./configure --prefix=/usr/local/nginx --user=nginx --group=nginx --with-http_stub_status_module
--with-http_ssl_module
make
make install
```

### 1.4 上传编写好的 nginx 启动管理脚本（见文本尾部）

## 2. MySQL

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

选择: MySQL Community Server 5.5 »

选择: Select Version: 按照自己要求选择

Select Operating System: Source Code

Select OS Version: Generic Linux

格式: mysql-N.N.NN.tar.gz

```
wget https://cdn.mysql.com//Downloads/MySQL-5.5/mysql-5.5.62.tar.gz
```

### 2.1 创建用于运行 Mysql 的用户:

```
useradd -r -s /sbin/nologin mysql
```

### 2.2 解压缩 Mysql 并安装:

```
cmake -DCMAKE_INSTALL_PREFIX=/usr/local/mysql -DMYSQL_UNIX_ADDR=/tmp/mysql.sock
-DEXTRA_CHARSETS=all -DDEFAULT_CHARSET=utf8 -DDEFAULT_COLLATION=utf8_general_ci
-DWITH_MYISAM_STORAGE_ENGINE=1 -DWITH_INNOBASE_STORAGE_ENGINE=1
-DWITH_MEMORY_STORAGE_ENGINE=1 -DWITH_READLINE=1
-DENABLED_LOCAL_INFILE=1 -DMYSQL_USER=mysql -DMYSQL_TCP_PORT=3306
make
make install
ln -s /usr/local/mysql/bin/* /usr/local/bin
```

### 2.3 修改安装后的目录权限

```
cd /usr/local/mysql
chown -R root .
chown -R mysql data
```

### 2.4 生成 Mysql 配置文件

```
cp -a /lnmp/mysql-5.5.62/support-files/my-medium.cnf /etc/my.cnf
```



## 2.5 初始化，生成授权表

```
cd /usr/local/mysql #一定要先切换到此目录下，然后再执行下一步。
./scripts/mysql_install_db --user=mysql
初始化成功标志：两个 ok
```

## 2.6 生成 Mysql 的启动和自启动管理脚本

```
cd /lnmp/mysql-5.5.62/support-files
切换到 mysql 的源码解压缩目录下的 support-files
cp -a mysql.server /etc/init.d/mysqld
chmod +x /etc/init.d/mysqld

chkconfig --add mysqld
chkconfig mysqld on
service mysqld start|stop|restart
```

## 2.7 给 mysql 的 root 用户设置密码

```
mysqladmin -uroot password 123456
```

## 3. PHP

下载：<http://www.php.net/>

```
wget https://www.php.net/distributions/php-7.1.29.tar.gz
```

### 3.1 解压缩 PHP 并安装：

```
./configure --prefix=/usr/local/php/ --with-config-file-path=/usr/local/php/etc/
--with-mysqli=/usr/local/mysql/bin/mysql_config --enable-soap --enable-mbstring=all --enable-sockets
--with-pdo-mysql=/usr/local/mysql --with-gd --without-pecl --enable-fpm
make
make install
```

报错提示：若遇到 libpng.so not found .报错（老版本的 PHP 会出现此问题）

解决方案：

```
ln -s /usr/lib64/libpng.so /usr/lib
```

### 3.2 生成 php 配置文件

```
cp -a /lnmp/php-7.1.29/php.ini-production /usr/local/php/etc/php.ini
```

复制源码包内的配置文件到安装目录下，并改名即可

### 3.3 创建软连接，使用 php 相关命令是更方便

```
ln -s /usr/local/php/bin/* /usr/local/bin/
ln -s /usr/local/php/sbin/* /usr/local/sbin/
```



## 4. 配置 Nginx 连接 PHP（重难点）

### 4.1 nginx 连接 php 需要启动 php-fpm 服务

```
cd /usr/local/php/etc/
cp -a php-fpm.conf.default php-fpm.conf
生成 php-fpm 的配置文件，并修改指定参数
```

```
vim php-fpm.conf
```

修改指定条目的参数：

```
pid = run/php-fpm.pid
cd /usr/local/php/etc/php-fpm.d/
cp -a www.conf.default www.conf
vim www.conf
```

修改用户和组的指定用户

```
user = nginx
group = nginx
```

修改 Nginx 启动管理脚本：将 php-fpm 的注释取消掉即可

### 4.2 修改 Nginx 的配置文件，使其识别.php 后缀的文件

```
vim /usr/local/nginx/conf/nginx.conf
取消下列行的注释，并修改 include 选项的后缀为 fastcgi.conf，并注意每一行结尾的分号和大括号
#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; #修改为 fastcgi.conf
#}
```

测试：

重启 Nginx 服务，创建 php 测试文件，访问并查看是否解析

### 4.3 修改 Nginx 配置文件，使其默认自动加载 php 文件

```
location / {
root html; #Nginx 的默认网页路径:PREFIX/html
index index.php index.html; #设置默认加载的页面，以及优先级
}
```

附件：建议使用时先复制到文本文件中查看下是否有字符集问题

```
######Nginx 启动管理脚本#####
#!/bin/bash
#Author: liu
#chkconfig: 2345 99 33
```



```
#description: nginx server control tools

ngxc="/usr/local/nginx/sbin/nginx"
pidf="/usr/local/nginx/logs/nginx.pid"
ngxc_fpm="/usr/local/php/sbin/php-fpm"
pidf_fpm="/usr/local/php/var/run/php-fpm.pid"
case "$1" in
 start)
 $ngxc -t &> /dev/null
 if [$? -eq 0];then
 $ngxc
 $ngxc_fpm
 echo "nginx service start success!"
 else
 $ngxc -t
 fi
 ;;
 stop)
 kill -s QUIT $(cat $pidf)
 kill -s QUIT $(cat $pidf_fpm)
 echo "nginx service stop success!"
 ;;
 restart)
 $0 stop
 $0 start
 ;;
 reload)
 $ngxc -t &> /dev/null
 if [$? -eq 0];then
 kill -s HUP $(cat $pidf)
 kill -s HUP $(cat $pidf_fpm)
 echo "reload nginx config success!"
 else
 $ngxc -t
 fi
 ;;
 *)
 echo "please input stop|start|restart|reload."
 exit 1
esac
```

## web 服务器-Nginx

### 一. 讲在 Nginx 之前

#### 同步与异步:

同步与异步的重点在消息通知的方式上，也就是调用结果的通知方式不同。

**同步：**当一个同步调用发出去后，调用者要一直等待调用的结果通知后，才能进行后续的执行。

**异步：**当一个异步调用发出去后，调用者不必一直等待调用结果的返回，异步调用，要想获得结果，一般有两种方式：

- 1、主动轮询异步调用的结果；
- 2、被调用方通过 callback（回调通知）来通知调用方调用结果。

#### 实例解释:

**同步取快递：**小明收到快递将送达的短信，在楼下一直等到快递送达。

**异步取快递：**小明收到快递将送达的短信，快递到楼下后，小明再下楼去取。

异步取快递，小明知道快递到达楼下有两种方式：

- 1、不停的电话问快递小哥到了没有，即主动轮询；
- 2、快递小哥到楼下后，打电话通知小明，然后小明下楼取快递，即回调通知。

#### 阻塞与非阻塞:

阻塞与非阻塞的重点在于进/线程等待消息时候的行为，也就是在等待消息的时候，当前进/线程是挂起状态，还是非挂起状态。

**阻塞：**调用在发出去后，在消息返回之前，当前进/线程会被挂起，直到有消息返回，当前进/线程才会被激活

**非阻塞：**调用在发出去后，不会阻塞当前进/线程，而会立即返回。

#### 实例解释:

**阻塞取快递：**小明收到快递即将送达的信息后，什么事都不做，一直专门等快递。

**非阻塞取快递：**小明收到快递即将送达的信息后，等快递的时候，还一边敲代码、一边刷微信。

同步与异步，重点在于消息通知的方式；阻塞与非阻塞，重点在于等消息时候的行为。

所以，就有了下面 4 种组合方式

- **同步阻塞：**小明收到信息后，啥都不干，等快递；
- **同步非阻塞：**小明收到信息后，边刷微博，边等着取快递；
- **异步阻塞：**小明收到信息后，啥都不干，一直等着快递员通知他取快递；
- **异步非阻塞：**小明收到信息后，边刷着微博，边等快递员通知他取快递。

大部分程序的 I/O 模型都是同步阻塞的，单个进程每次只在一个文件描述符上执行 I/O 操作，每次 I/O 系统调用都会阻塞，直到完成数据传输。传统的服务器采用的就是同步阻塞的多进程模型。一个 server 采用一个进程负责一个 request 的方式，一个进程负责一个 request，直到会话结束。进程数就是并发数，而操作系统支持的进程数是有限的，且进程数越多，调度的开销也越大，因此无法面对高并发。

Nginx 采用了异步非阻塞的方式工作。我们先来先了解一下 I/O 多路复用中的 epoll 模型。

### epoll 模型：

当连接有 I/O 事件产生的时候，epoll 就会去告诉进程哪个连接有 I/O 事件产生，然后进程就去处理这个事件。

例如：小明家楼下有一个收发室，每次有快递到了，门卫就先代收并做了标记；然后通知小明去取送给小明的快递。

## 为什么 Nginx 比其他 web 服务器并发高（Nginx 工作原理）：

Nginx 配置 use epoll 后，以异步非阻塞方式工作，能够轻松处理百万级的并发连接。

**处理过程：**每进来一个 request，会有一个 worker 进程去处理。但不是全程的处理，处理到可能发生阻塞的地方。比如向后端服务器转发 request，并等待请求返回。那么，这个处理的 worker 不会这么傻等着，他会在发送完请求后，注册一个事件：“如果后端服务器返回了，告诉我一声，我再接着干”。于是他就休息去了。此时，如果有新的 request 进来，他就可以很快再按这种方式处理。而一旦后端服务器返回了，就会触发这个事件，worker 才会来接手，这个 request 才会接着往下走。通过这种快速处理，快速释放请求的方式，达到同样的配置可以处理更大并发量的目的。

## 二. Nginx 详解

### 1. 概述

Nginx (engine x) 是一个高性能的 HTTP 和反向代理 web 服务器，同时也提供了 IMAP/POP3/SMTP 服务。Nginx 是由伊戈尔·赛索耶夫为俄罗斯访问量第二的 Rambler.ru 站点开发的，第一个公开版本 0.1.0 发布于 2004 年 10 月 4 日。

Nginx 是一款轻量级的 Web 服务器/反向代理服务器及电子邮件（IMAP/POP3）代理服务器，在 BSD-like 协议下发行。其特点是占有内存少，并发能力强。

## 2. 工作模式

nginx 有两种工作模式：master-worker 模式和单进程模式。在 master-worker 模式下，有一个 master 进程和至少一个的 worker 进程，单进程模式顾名思义只有一个进程。这两种模式有各自的特点和适用场景。

### master-worker：

该模式下，nginx 启动成功后，会有一个 master 进程和至少一个的 worker 进程。master 进程负责处理



系统信号，加载配置，管理 worker 进程（启动，杀死，监控，发送消息/信号等）。worker 进程负责处理具体的业务逻辑，也就是说，对外部来说，真正提供服务的是 worker 进程。生产环境下一般使用这种模式，因为这种模式有以下优点：

1. 稳定性高，只要还有 worker 进程存活，就能够提供服务，并且一个 worker 进程挂掉 master 进程会立即启动一个新的 worker 进程，保证 worker 进程数量不变，降低服务中断的概率。
2. 配合 linux 的 cpu 亲和性配置，可以充分利用多核 cpu 的优势，提升性能
3. 处理信号/配置重新加载/升级时可以做到尽可能少或者不中断服务（热重启）

### 单进程模式：

单进程模式下，nginx 启动后只有一个进程，nginx 的所有工作都由这个进程负责。由于只有一个进程，因此可以很方便地利用 gdb 等工具进行调试。该模式不支持 nginx 的平滑升级功能，任何的信号处理都可能造成服务中断，并且由于是单进程，进程挂掉后，在没有外部监控的情况下，无法重启服务。因此，该模式一般只在开发阶段和调试时使用，生产环境下不会使用。（了解）

## 3. 配置文件结构

```
user www www;
#程序运行用户和组
worker_processes auto;
#启动进程，指定 nginx 启动的工作进程数量，建议按照 cpu 数目来指定，一般等于 cpu 核心数目
error_log /home/wwwlogs/nginx_error.log crit;
#全局错误日志
pid /usr/local/nginx/logs/nginx.pid;
#主进程 PID 保存文件
worker_rlimit_nofile 51200;
#文件描述符数量
events
{
 use epoll;
 #使用 epoll 模型，对于 2.6 以上的内核，建议使用 epoll 模型以提高性能
 worker_connections 51200;
 #工作进程的最大连接数量
}
http{
 #网站优化参数
 server { #具体的某一网站的配置信息
 listen 80; #监听端口
 root html; #网页根目录 (/usr/local/nginx/html)
 server_name www.atguigu.com; #服务器域名
 index index.html; #默认加载页面
 access_log logs/access.log; #访问日志保存位置
 ;
 }
}
```

```
location (*.*)\.php$ {
 用正则匹配具体的访问对象;
}
location {
 跳转等规则;
}
}
server {
 虚拟主机;
}
}
```

## 4. Nginx 相关实验

### <注意事项>

1. 注意配置文件中的结尾有;作为结束~!（切记！）
  2. 每次实验修改完配置文件后需要重启 nginx 才会生效
- ```
# pkill -HUP nginx
```

实验 1：Nginx 的状态统计

- a、安装 nginx 时将 --with-http_stub_status_module 模块开启
- b、修改 nginx 配置文件（写入要访问的 server 标签中）

```
location /nginx_status{  
    stub_status on;  
    access_log off;  
}
```

- c、客户端访问网址:http://IP/nginx_status

"Active connections"表示当前的活动连接数；
"server accepts handled requests"表示已经处理的连接信息
三个数字依次表示已处理的连接数、成功的 TCP 握手次数、已处理的请求数

实验 2：目录保护

- a、原理和 apache 的目录保护原理一样（利用上一个实验接着完成）
- b、在状态统计的 location 中添加：

```
auth_basic "Welcome to nginx_status!";  
auth_basic_user_file /usr/local/nginx/html/htpasswd.nginx;
```

- c、使用 http 的命令 htpasswd 进行用户密码文件的创建（**生成在上面指定的位置**）

```
# htpasswd -c /usr/local/nginx/html/htpasswd.nginx user
```

- d、重启 nginx 并再次访问统计页面

实验 3：基于 IP 的身份验证（访问控制）

- a、接着上一个实验完成操作

b、在状态统计的 location 中添加:

```
allow 192.168.88.1;  
deny 192.168.88.0/24;  
仅允许 192.168.88.1 访问服务器
```

实验 4: nginx 的虚拟主机 (基于域名)

a、提前准备好两个网站的域名，并且规划好两个网站网页存放目录

b、在 Nginx 主配置文件中并列编写两个 server 标签，并分别写好各自信息

```
server {  
    listen 80;  
    server_name blog.atguigu.com;  
    index index.html index.htm index.php;  
    root html/blog;  
    access_log logs/blog-access.log main;  
}  
  
server {  
    listen 80;  
    server_name bbs.atguigu.com;  
    index index.html index.htm index.php;  
    root html/bbs;  
    access_log logs/bbs-access.log main;  
}
```

c、分别访问两个不同的域名验证结果

实验 5: nginx 的反向代理

代理和反向代理?

代理: 找别人代替你去完成一件你完不成的事(代购), 代理的对象是客户端

反向代理: 替厂家卖东西的人就叫反向代理(烟酒代理), 代理的对象是服务器端

a、在另外一台机器上安装 apache, 启动并填写测试页面

b、在 nginx 服务器的配置文件中添加 (写在某一个网站的 server 标签内)

```
location / {  
    proxy_pass http://192.168.88.100:80;          #此处填写 apache 服务器的 IP 地址  
}
```

c、重启 nginx, 并使用客户端访问测试

实验 6: 负载调度 (负载均衡)

负载均衡 (Load Balance) 其意思就是将任务分摊到多个操作单元上进行执行, 例如 Web 服务器、FTP 服务器、企业关键应用服务器和其它关键任务服务器等, 从而共同完成工作任务。

a、使用默认的 rr 轮训算法, 修改 nginx 配置文件

```
upstream bbs {                      #此标签在 server 标签前添加  
    server 192.168.88.100:80;  
    server 192.168.88.200:80;
```

```
}

server {
    .......;

    #修改自带的 location / 的标签，将原内容删除，添加下列两项
    location / {
        proxy_pass http://bbs;    #添加反向代理，代理地址填写 upstream 声明的名字
        proxy_set_header Host $host;    #重写请求头部，保证网站所有页面都可访问成功
    }
}
```

c、开启并设置两台 88.100 & 88.200 的主机

安装 apache 并设置不同的 index.html 页面内容（设置不同页面是为了看实验效果）

d、重启 nginx，并使用客户端访问测试

拓展补充：rr 算法实现加权轮询（后期集群再讲更多算法类型和功能）

```
upstream bbs {
    server 192.168.88.100:80 weight=1;
    server 192.168.88.200:80 weight=2;
}
```

实验 7：nginx 实现 https {证书+rewrite}

- a、安装 nginx 时，需要将--with-http_ssl_module 模块开启
- b、在对应要进行加密的 server 标签中添加以下内容开启 SSL

```
server {
    ....;
    ssl on;
    ssl_certificate /usr/local/nginx/conf/ssl/atguigu.crt;
    ssl_certificate_key /usr/local/nginx/conf/ssl/atguigu.key;
    ssl_session_timeout 5m;
    ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
    ssl_prefer_server_ciphers on;
    ssl_ciphers
        "EECDH+CHACHA20:EECDH+CHACHA20-draft:EECDH+AES128:RSA+AES128:EECDH+AES2
56:RSA+AES256:EECDH+3DES:RSA+3DES:!MD5";
}
```

c、生成证书和秘钥文件

注意：在实验环境中可以用命令生成测试，在生产环境中必须要在 https 证书厂商注册

```
# openssl genrsa -out atguigu.key 1024
```

建立服务器私钥，生成 RSA 密钥

```
# openssl req -new -key atguigu.key -out atguigu.csr
```

需要依次输入国家，地区，组织，email。最重要的是有一个 common name，可以写你的名字或者域名。如果为了 https 申请，这个必须和域名吻合，否则会引发浏览器警报。生成的 csr 文件交给 CA 签名后形成服务端自己的证书

```
# openssl x509 -req -days 365 -sha256 -in atguigu.csr -signkey atguigu.key -out atguigu.crt
```

生成签字证书

```
# cp atguigu.crt /usr/local/nginx/conf/ssl/atguigu.crt  
# cp atguigu.key /usr/local/nginx/conf/ssl/atguigu.key
```

将私钥和证书复制到指定位置

d、设置 http 自动跳转 https 功能

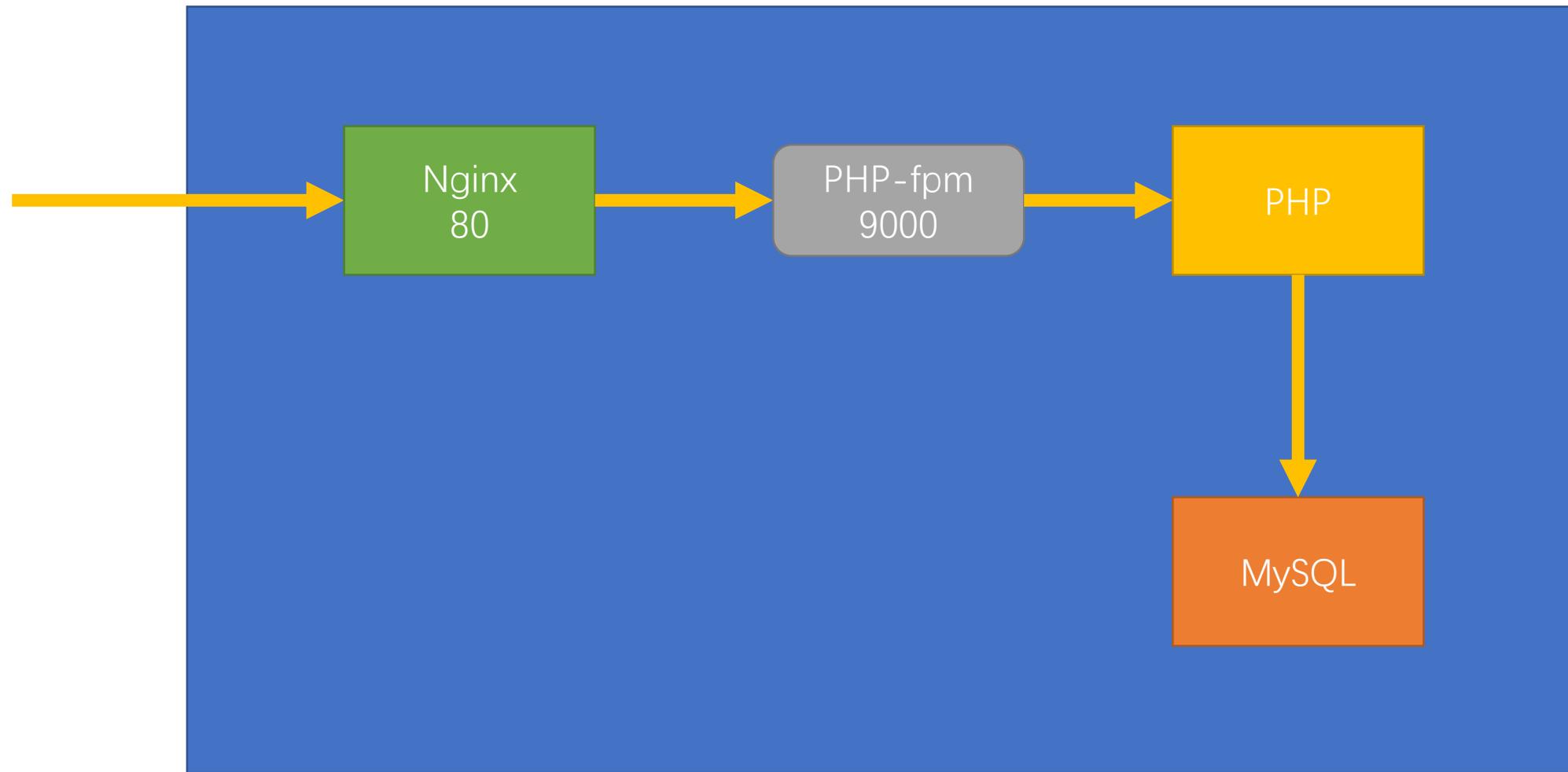
原有的 server 标签修改监听端口

```
server {  
    .....;  
    listen 443;  
}
```

新增以下 server 标签（利用虚拟主机+rewrite 的功能）

```
server{  
    listen 80;  
    server_name bbs.atguigu.com;  
    rewrite ^(.*)$ https://bbs.atguigu.com permanent;  
    root html;  
    index index.html index.htm;  
}
```

e、重启 nginx，并测试



Nginx 配置文件详解

```
user    www www;
#程序运行用户和组

worker_processes auto;
#启动进程，指定 nginx 启动的工作进程数量，建议按照 cpu 数目来指定，一般等于 cpu 核心数目

error_log  /home/wwwlogs/nginx_error.log  crit;
#全局错误日志

pid      /usr/local/nginx/logs/nginx.pid;
#主进程 PID 保存文件

worker_rlimit_nofile 51200;
#文件描述符数量

events
{
    use epoll;
    #使用 epoll 模型，对于 2.6 以上的内核，建议使用 epoll 模型以提高性能

    worker_connections 51200;
    #工作进程的最大连接数量，根据硬件调整，和前面工作进程配合起来用，尽量大，但是别把 cpu 跑到 100% 就行每个进程允许的最大连接数，理论上每台 nginx 服务器的最大连接数为 worker_processes*worker_connections，具体还要看服务器的硬件、带宽等。
}

http
#整体环境配置--网站配置
{
    include      mime.types;
    default_type application/octet-stream;
    #设定 mime 类型,文件传送类型由 mime.type 文件定义
    server_names_hash_bucket_size 128;
    #保存服务器名字的 hash 表大小
    client_header_buffer_size 32k;
    #客户端请求头部缓冲区大小
    large_client_header_buffers 4 32k;
    #最大客户端头缓冲大小
```

```
client_max_body_size 50m;  
#客户端最大上传文件大小 (M)
```

sendfile on;

#sendfile 指令指定 nginx 是否调用 sendfile 函数来输出文件，对于普通应用，必须设为 on。如果用来进行下载等应用磁盘 IO 重负载应用，可设置为 off，以平衡磁盘与网络 I/O 处理速度，降低系统的 uptime。

```
tcp_nopush on;
```

#这个是默认的，结果就是数据包不会马上传送出去，等到数据包最大时，一次性的传输出去，这样有助于解决网络堵塞。（只在 sendfile on 时有效）

```
keepalive_timeout 60;  
#连接超时时间
```

```
tcp_nodelay on;
```

#禁用 nagle 算法，也即不缓存数据。有效解决网络阻塞

```
fastcgi_connect_timeout 300;
```

```
fastcgi_send_timeout 300;
```

```
fastcgi_read_timeout 300;
```

```
fastcgi_buffer_size 64k;
```

```
fastcgi_buffers 4 64k;
```

```
fastcgi_busy_buffers_size 128k;
```

```
fastcgi_temp_file_write_size 256k;
```

#fastcgi 设置

```
gzip on;
```

```
gzip_min_length 1k;
```

```
gzip_buffers 4 16k;
```

```
gzip_http_version 1.1;
```

```
gzip_comp_level 2;
```

```
gzip_types text/plain application/javascript application/x-javascript text/javascript  
text/css application/xml application/xml+rss;
```

```
gzip_vary on;
```

```
gzip_proxied expired no-cache no-store private auth;
```

```
gzip_disable "MSIE [1-6]\.";
```

```
#limit_conn_zone $binary_remote_addr zone=perip:10m;
```

##If enable limit_conn_zone, add "limit_conn perip 10;" to server section.

```
server_tokens off;
```

#隐藏 nginx 版本号（curl -I 192.168.4.154 可以查看，更加安全）

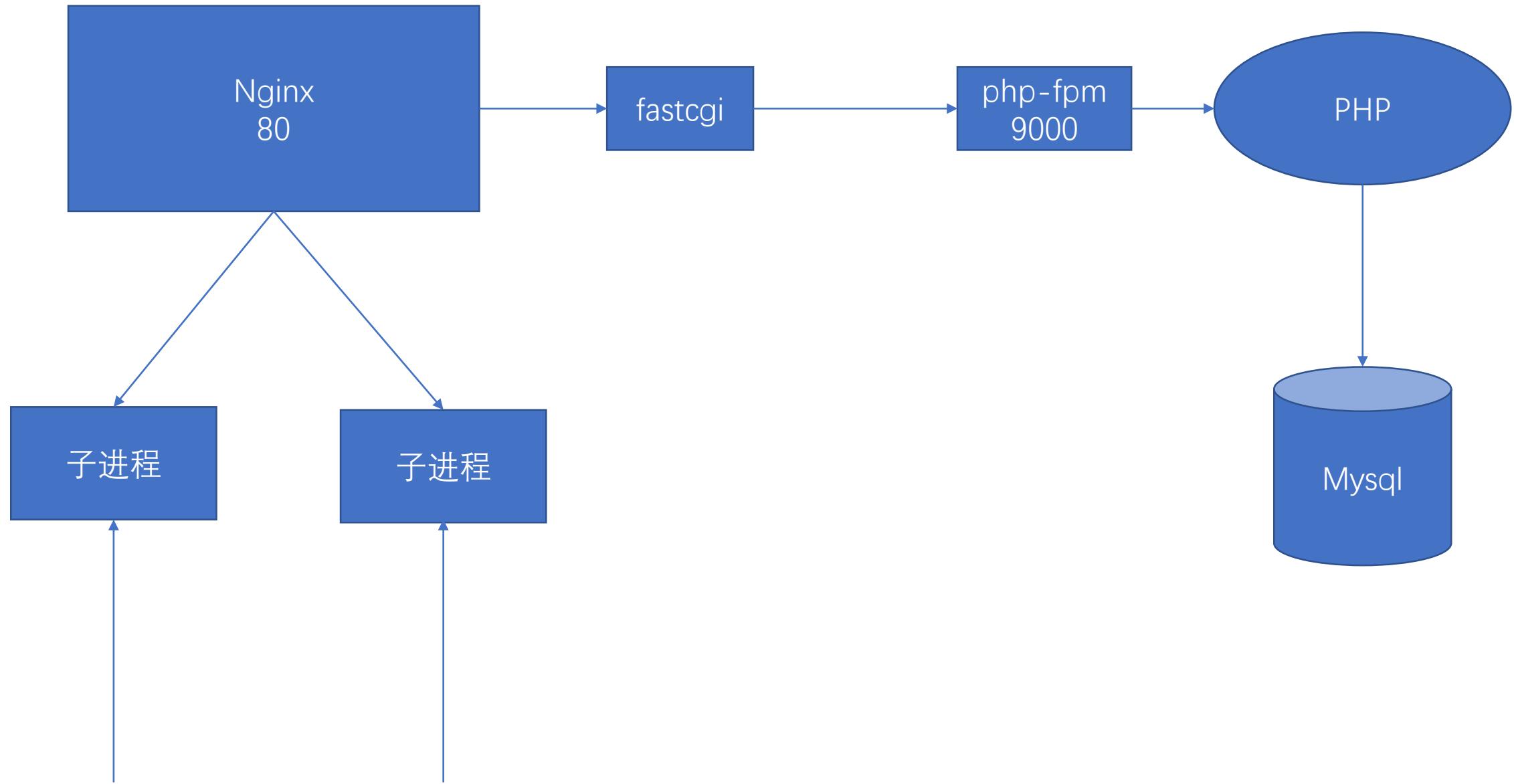
```
#log format
```

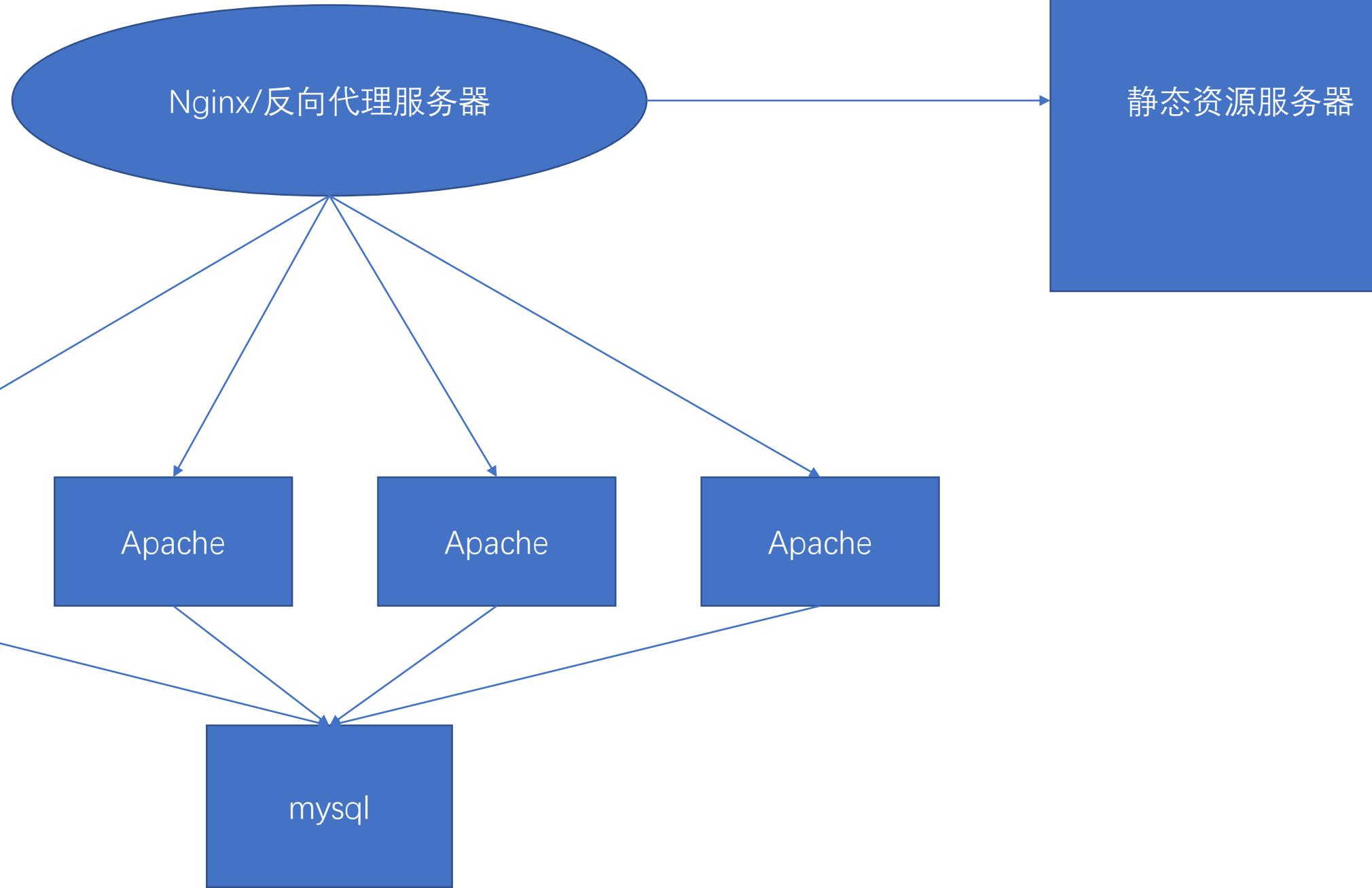
```
log_format access '$remote_addr - $remote_user [$time_local] "$request"  
'$status $body_bytes_sent "$http_referer"  
'"$http_user_agent" $http_x_forwarded_for';  
#定义日志格式  
  
server  
{  
    listen 80 default_server;  
    #listen [::]:80 default_server ipv6only=on;  
    #监听 80 端口，WEB 服务的监听设置，可以采用"IP 地址:端口"形式  
    server_name www.lnmp.org lnmp.org;  
    #服务器名，可以写多个域名，用空格分隔  
    index index.html index.htm index.php;  
    #默认网页文件  
    root /home/wwwroot/default;  
    #网页主目录  
  
    #error_page 404 /404.html;  
    include enable-php.conf;  
  
    location /nginx_status  
    {  
        stub_status on;  
        access_log off;  
    }  
    #开启 status 状态监测  
    location ~ \.(gif|jpg|jpeg|png|bmp|swf)$  
    {  
        expires 30d;  
    }  
    #静态文件处理，保存期 30 天  
    location ~ \.(js|css)?$  
    {  
        expires 12h;  
    }  
    #js 和 css 文件处理，保存期 12 小时  
    location ~ \.  
    {  
        deny all;  
    }  
  
    access_log /home/wwwlogs/access.log access;
```

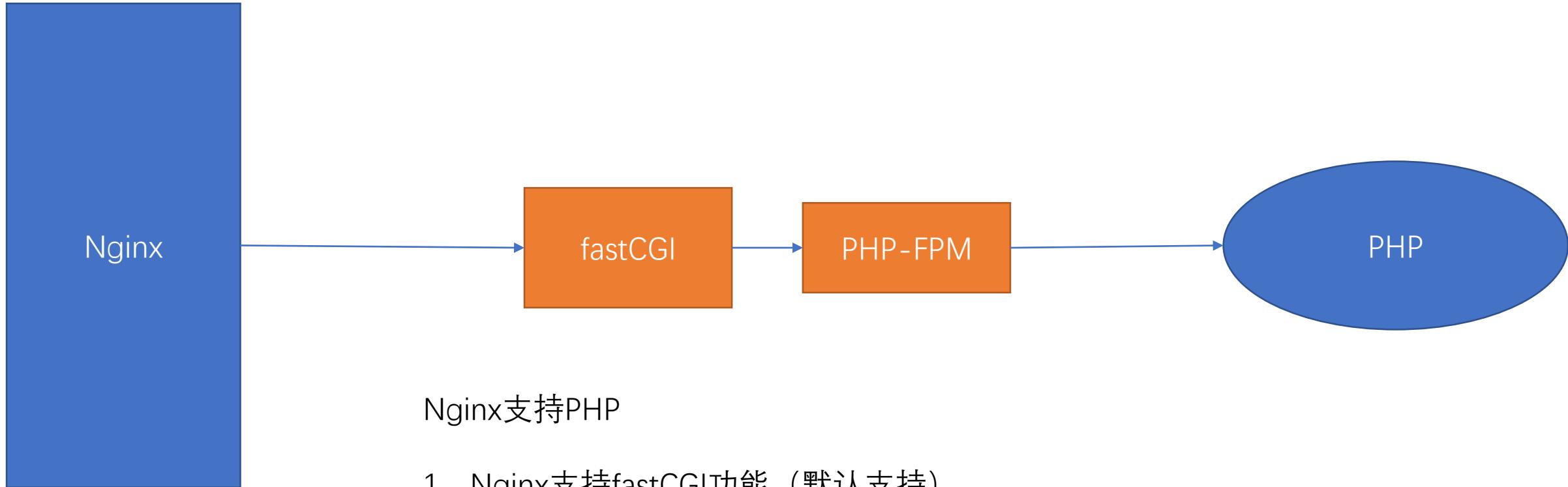


```
#正确访问日志
}
include vhost/*.conf;
#vhost/下子配置文件生效
}
```



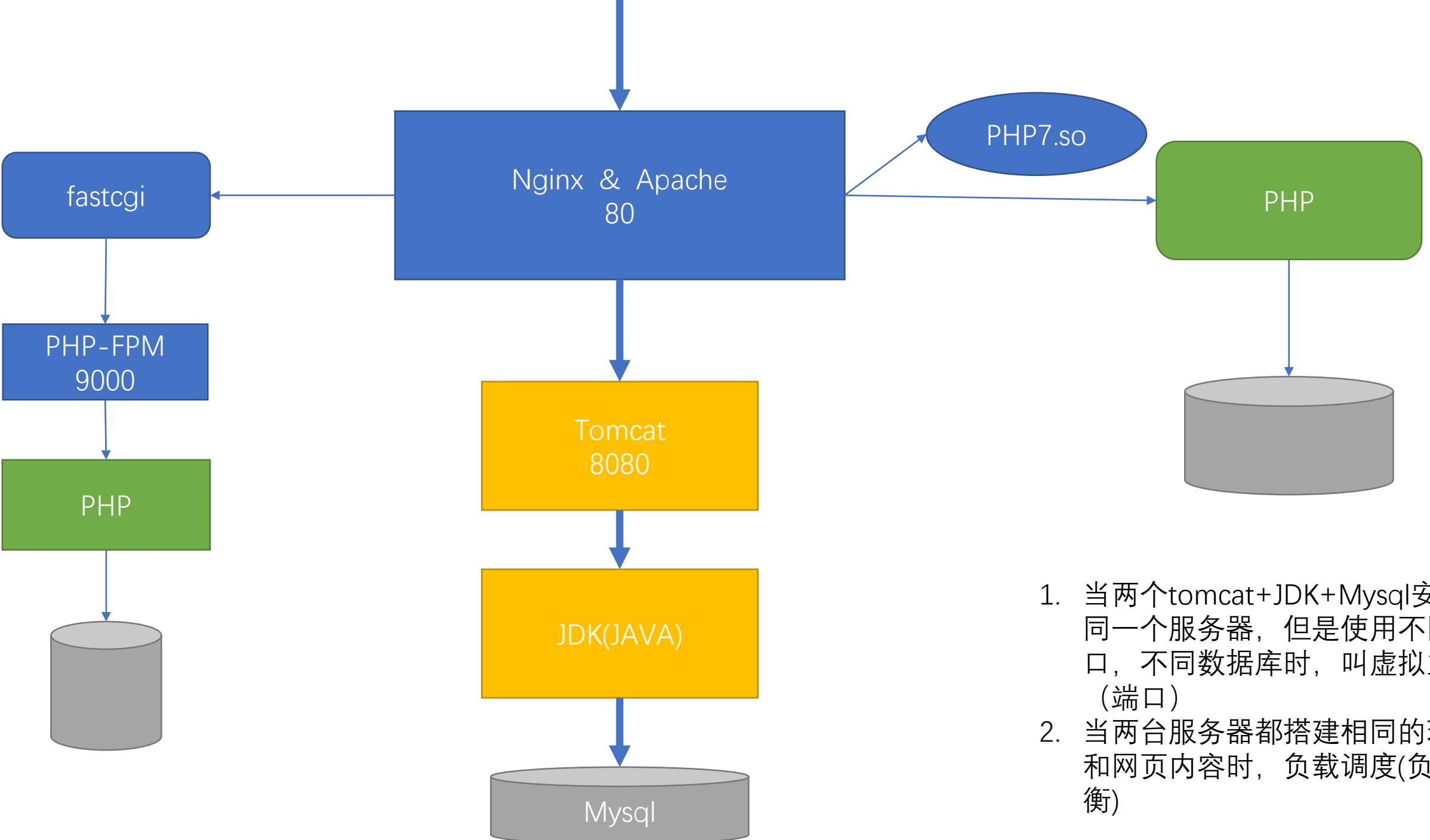




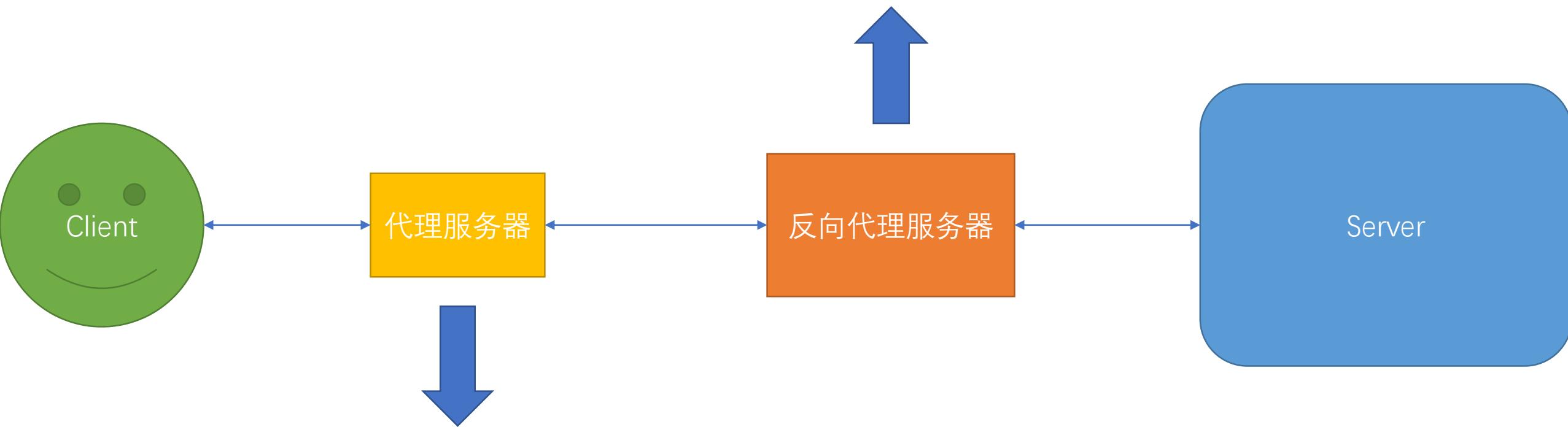


Nginx支持PHP

1. Nginx支持fastCGI功能（默认支持）
2. PHP编译时开启FPM服务（编译时指定）
3. 在Nginx配置文件中添加匹配规则（匹配后缀是.php）
(修改nginx.conf)

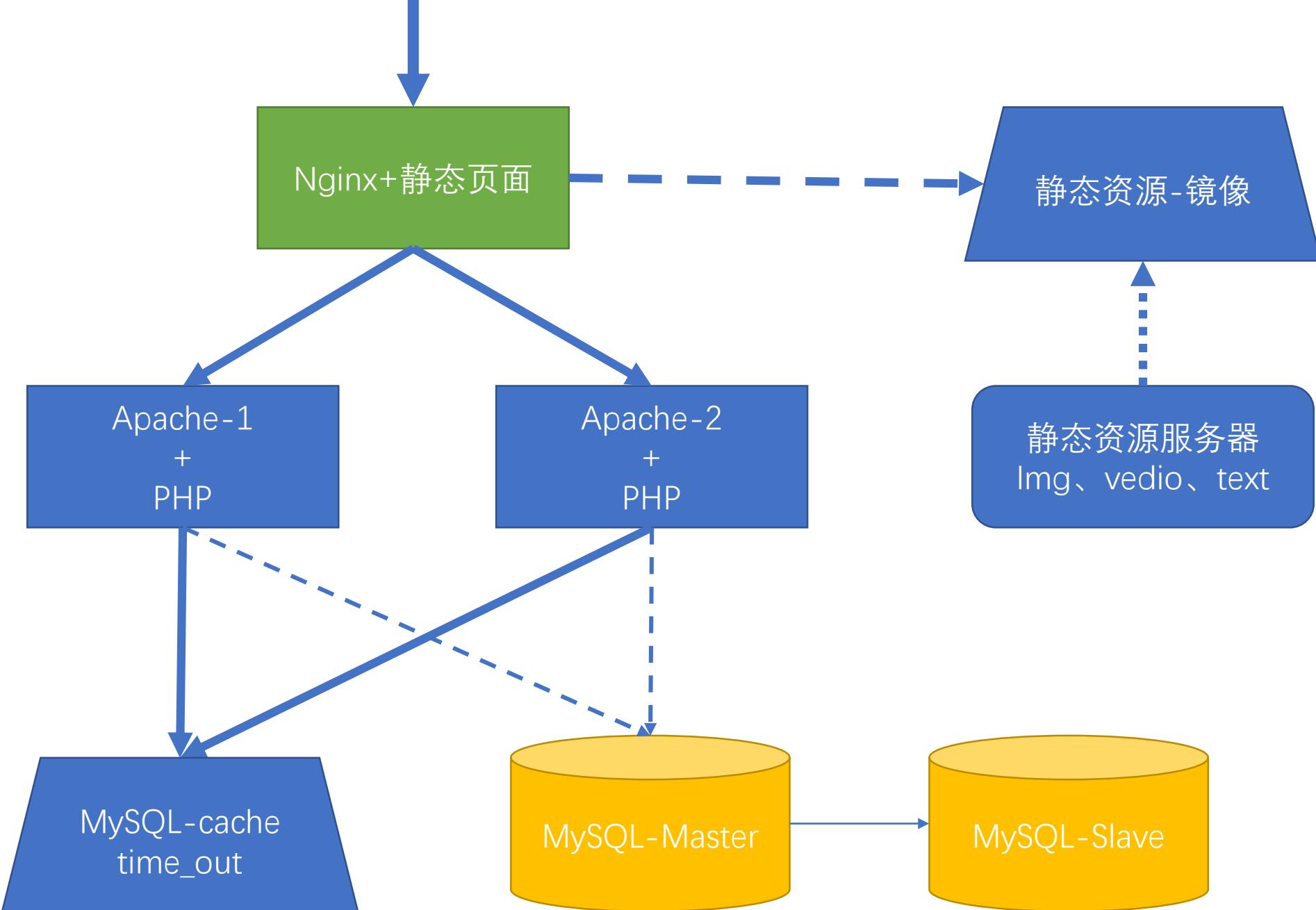


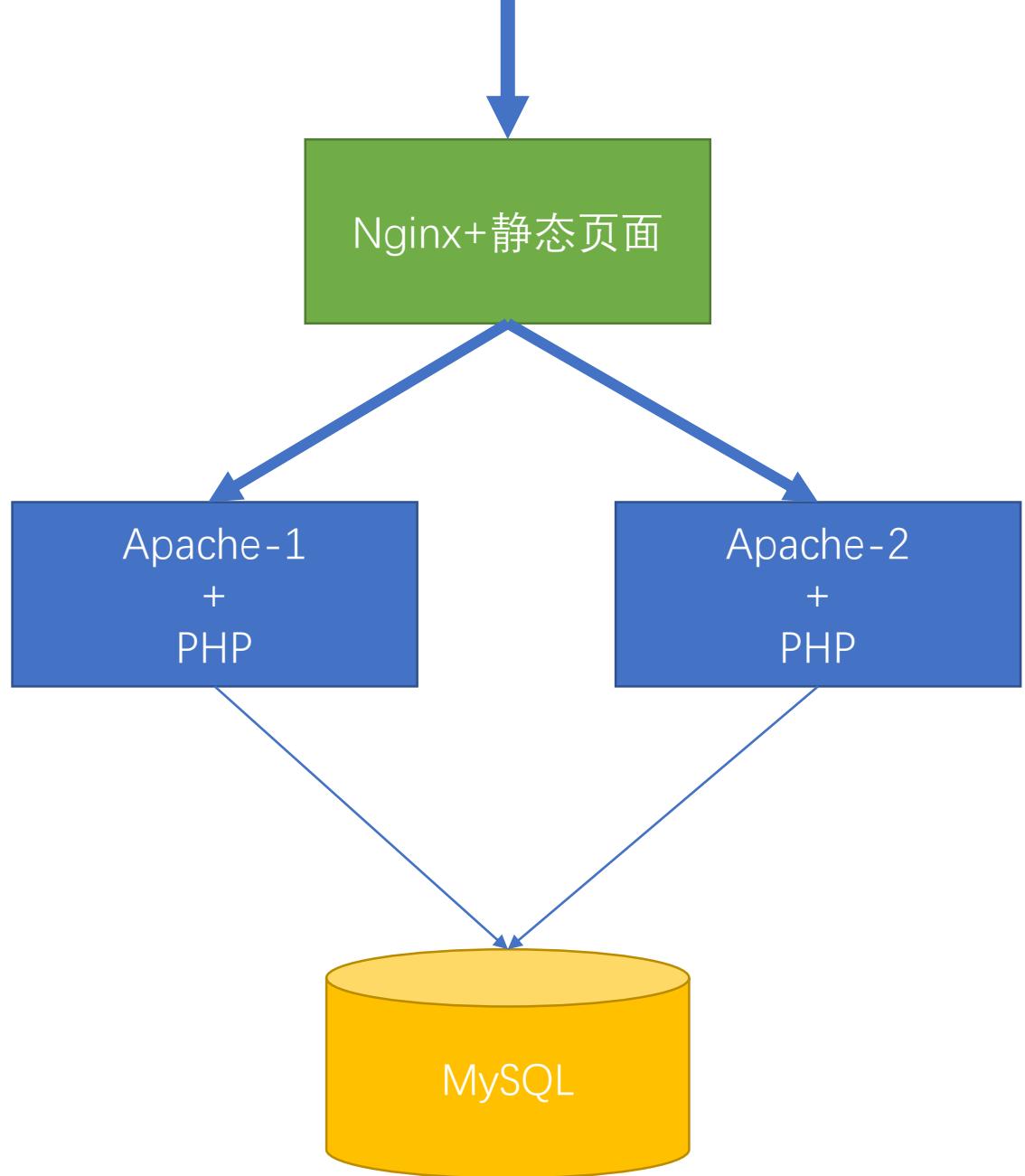
1. 当两个tomcat+JDK+Mysql安装在同一个服务器，但是使用不同端口，不同数据库时，叫虚拟主机（端口）
2. 当两台服务器都搭建相同的环境和网页内容时，负载调度(负载均衡)



相当于厂商代理，代替
服务器端接收用户的请
求，完成请求处理工作
Nginx、squid

相当于是代购，代
替客户端完成客户
端无法完成的任务
squid





企业邮件部署

一. 邮件概述

电子邮件服务器是处理邮件交换的软硬件设施的总称，包括电子邮件程序、电子邮箱等。为用户提供基于 E-mail 服务的电子邮件系统，人们通过访问服务器实现邮件的交换。

常见的邮件服务器

| 类型 | 名称 | 特点 |
|------|----------|---|
| 服务器端 | Sendmail | 资格最古老,运行稳定,但安全性欠佳 |
| | Postfix | 采用模块化设计,在投递效率、稳定性、性能及安全性方面表现优秀,与 sendmail 保持足够的兼容性。 |
| | Qmail | 采用模块化设计,速度快、执行效率高,配置稍微复杂点 |
| 客户端 | Outlook | 都是用来收邮件的客户端! |
| | foxmail | |
| | 浏览器 | |

二. 邮件应用协议

SMTP 简单邮件传输协议（发邮件）TCP 25 端口，加密时使用 TCP 465 端口

POP3 第三版邮局协议（收邮件）TCP 110 端口，加密时使用 TCP 995 端口

IMAP4 第四版互联网邮件访问协议（收邮件）TCP 143 端口，加密时使用 TCP 993 端口

三. 软件相关

软件名：Postfix

主目录：/etc/postfix

主配置文件：main.cf

myhostname:邮件服务器主机名

mydomain:邮件域

myorigin:设置允许发信的用户的邮件域

mydestination:设置允许收信的用户的邮件域

四. 企业级邮件服务

前提条件：要有一个 DNS 服务器，将准备好的软件包导入虚拟机

发送方配置

1. DNS 搭建

```
# yum -y install bind  
# vi /etc/named.conf  
    修改监听地址和访问控制为 any  
# vi /etc/named.rfc1912.zones  
    使用 extmail.org 作为解析域，只保留正向解析即可  
# vi /var/named/extmail.localhost  
    修改数据文件，如下  
        NS      dns.extmail.org  
        MX 3   mail.extmail.org  
dns     A      192.168.88.10  
mail    A      192.168.88.10
```

在另一台上使用 nslookup 测试 dns 是否能解析

2. 安装 gcc 以及其他依赖软件

```
# yum -y install gcc* mysql-server mysql httpd mailx  
启动 MySQL（Apache）并设置开机自启  
# chkconfig mysqld(httpd) on  
# service mysqld(httpd) start
```

3. 将 web 页面放到 Apache 的网页目录下

```
# mkdir -p /var/www/extsuite  
创建一个单独的目录  
# tar -xf extmail-1.2.tar.gz -C /var/www/extsuite/  
# tar -xf extman-1.1.tar.gz -C /var/www/extsuite/  
将 extmail 和 extman 解压到创建的目录中  
# cd /var/www/extsuite  
# mv extmail-1.2/ extmail  
# mv extman-1.1/ extman  
# chown -R root.root *  
将两个解压后的目录去掉版本号，并修改文件归属
```

4. 将成品数据库文件导入到 MySQL 中（没设置密码，空密码登录）

```
# cd /var/www/extsuite/  
# mysql < ./extman/docs/extmail.sql  
将./extman/docs 中模板和数据导入到数据库中  
# vi ./extman/docs/init.sql  
INSERT INTO `manager` VALUES ('root@extmail.org','123456','admin','root')  
将此文件中该位置的密码修改为 123456，切记先修改，再导入  
# mysql < ./extman/docs/init.sql
```

5. 将邮件模板拷贝到邮件服务器的主目录下

```
# cd /var/www/extsuite/extman/docs/  
# cp -a mysql_virtual_alias_maps.cf mysql_virtual_domains_maps.cf mysql_virtual_mailbox_maps.cf  
/etc/postfix/
```

6. 创建映射用户&修改配置文件

```
# useradd -u 600 -s /sbin/nologin vmail  
# vim /etc/postfix/main.cf  
inet_interfaces = all          #将此选项取消注释  
#inet_interfaces = localhost   #将此选项注释掉  
在尾部添加下列内容  
virtual_mailbox_base = /home/vmail  
virtual_uid_maps = static:600  
virtual_gid_maps = static:600  
virtual_alias_maps = mysql:/etc/postfix/mysql_virtual_alias_maps.cf  
virtual_mailbox_domains = mysql:/etc/postfix/mysql_virtual_domains_maps.cf  
virtual_mailbox_maps = mysql:/etc/postfix/mysql_virtual_mailbox_maps.cf
```

7. 重启服务&发送邮件测试&查看结果

```
# service postfix restart  
# echo "hello" | mail -s test support@extmail.org  
# ls /home/vmail/extmail.org/postmaster/Maildir/new/
```

接收方配置

1. 安装 dovecot 相关软件

```
# yum -y install dovecot dovecot-devel dovecot-mysql  
# chkconfig dovecot on  
# service dovecot start
```

2. 配置 dovecot 能够去数据库里读数据

1) 修改/etc/dovecot/conf.d/10-mail.conf

在配置文件中增加下列两行

```
mail_location = maildir:/home/vmail/%d/%n/Maildir #定义 dovecot 查询邮件的位置（顶头写）  
first_valid_uid = 600
```

2) 修改/etc/dovecot/conf.d/10-auth.conf

```
!include auth-sql.conf.ext #取消调用数据库的记录注释
```

3) 修改数据库连接配置文件（需要拷贝模板生成）

```
# cp -a /usr/share/doc/dovecot-2.0.9/example-config/dovecot-sql.conf.ext /etc/dovecot/  
# vim dovecot-sql.conf.ext      #将下列内容加入配置文件即可
```

driver = mysql

驱动类型

```
connect = host=localhost dbname=extmail user=extmail password=extmail
```

连接数据库的信息

```
default_pass_scheme = MD5
password_query = \
    SELECT username, domain, password \
    FROM mailbox WHERE username = '%u' AND domain = '%d'
```

验证登录密码的查询命令

```
user_query = SELECT maildir, 600 AS uid, 600 AS gid FROM mailbox WHERE username = '%u'
```

查询虚拟用户对应的邮箱目录

3. 重启 dovecot 验证是否能连接

安装 telnet 客户端进行登录验证

```
# yum -y install telnet
# telnet mail.extmail.org 110
user postmaster@extmail.org      #登录 postmaster 用户
pass extmail                      #密码是 extmail
retr 1                            #查看第一封邮件
```

MAIL+WEB 页面

1. 修改/etc/httpd/conf/httpd.conf 配置文件，能加载邮件 web 页面

```
NameVirtualHost *:80      #取消注释，开启虚拟主机功能
```

添加一下内容

```
<VirtualHost *:80>
    DocumentRoot /var/www/extsuite/extmail/html
    ServerName mail.extmail.org
    scriptalias /extmail/cgi /var/www/extsuite/extmail/cgi
    alias /extmail /var/www/extsuite/extmail/html
    scriptalias /extman/cgi /var/www/extsuite/extman/cgi
    alias /extman /var/www/extsuite/extman/html
    suexecusergroup vmail vmail
</VirtualHost>
```

2. extmail 目录中更改 cgi 的属组属主，让 vmail 有权限执行

```
# chown -R vmail.vmail cgi/
# cp -a webmail.cf.default webmail.cf
# vim webmail.cf
SYS_MAILDIR_BASE = /home/vmail          #邮件存放目录
SYS_CRYPT_TYPE = plain                  #加密类型
SYS_MYSQL_USER = extmail               #MySQL 用户名
SYS_MYSQL_PASS = extmail               #MySQL 密码
```

3. extman 中更改 cgi 的属组属主，让 vmail 有权限执行

```
# chown -R vmail.vmail cgi/
# cp -a webman.cf.default webman.cf
```

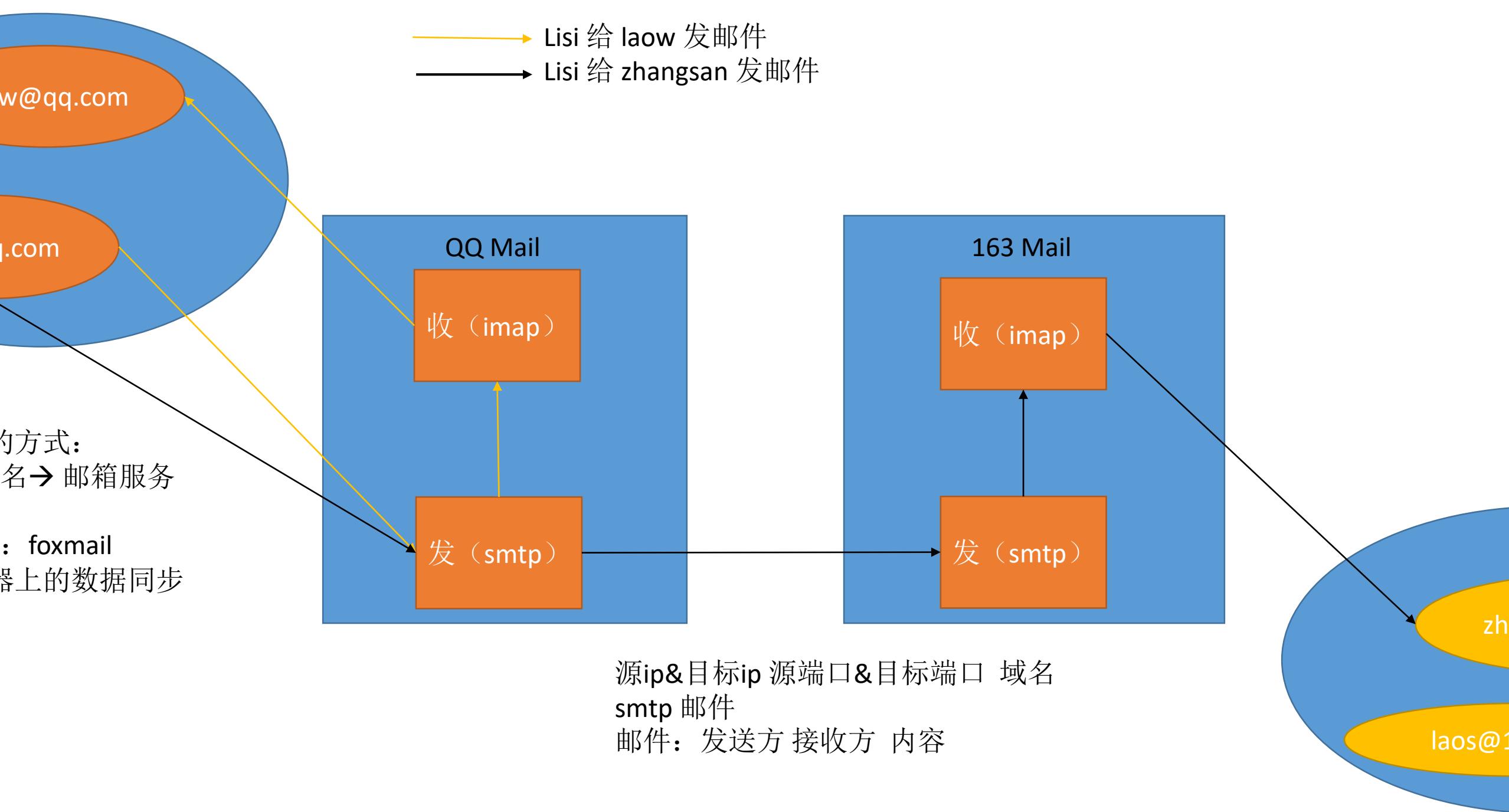
```
# vim webman.cf
SYS_MAILDIR_BASE = /home/vmail
SYS_SESS_DIR = /tmp
SYS_CAPTCHA_ON = 0          #生产环境中开启，实验环境无法显示验证码
SYS_CRYPT_TYPE = plain
```

4. 安装 Unix-Syslog 软件

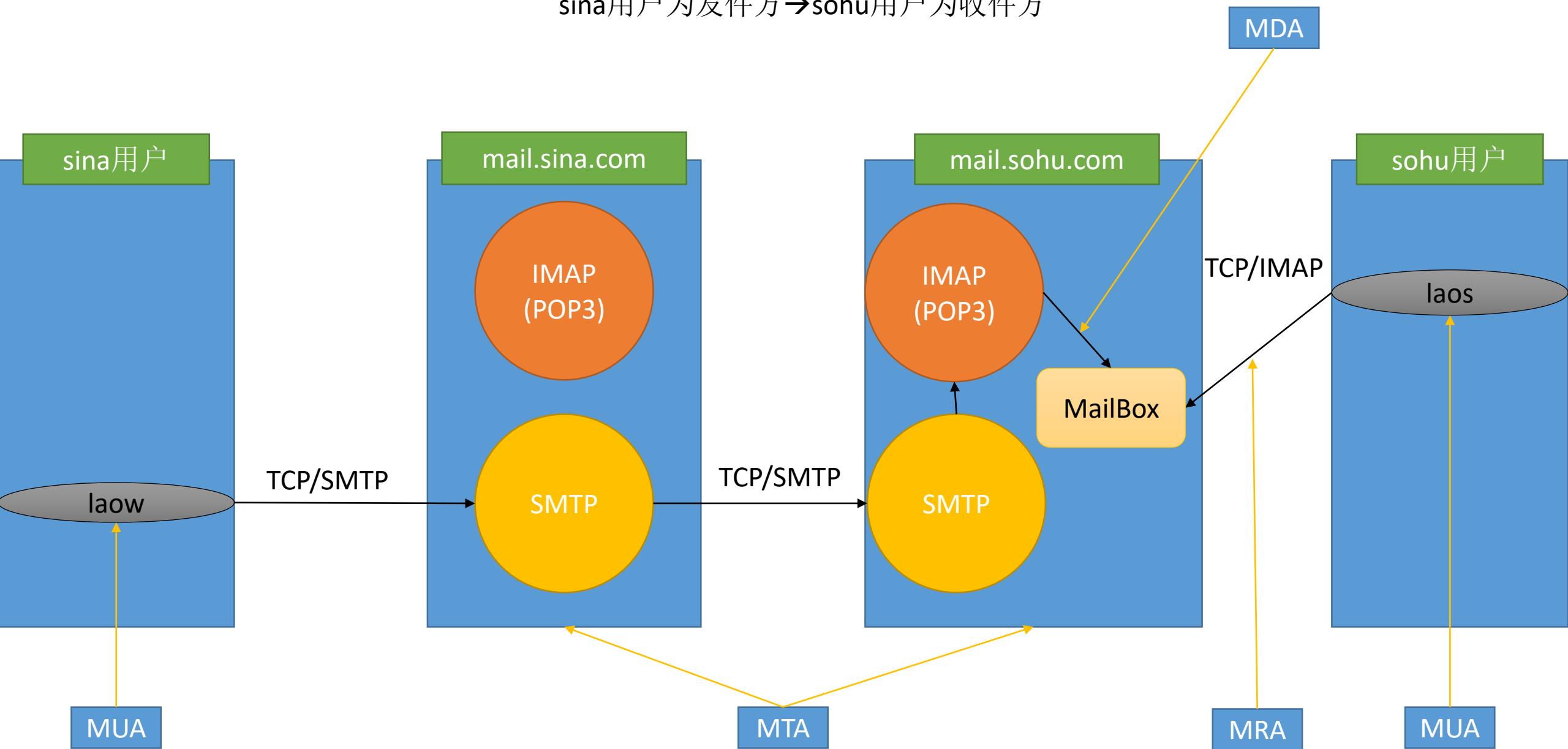
解压缩 Unix-Syslog-1.1.tar.gz 软件

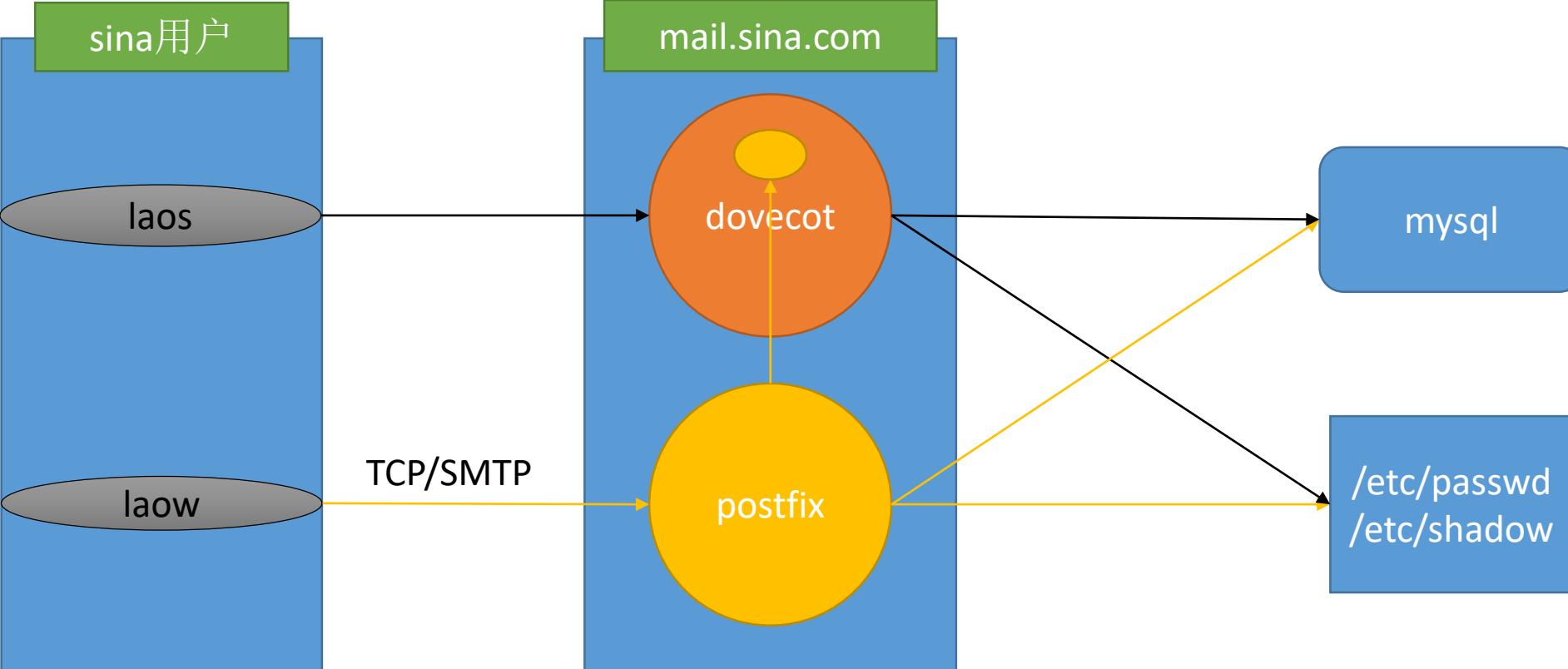
```
# cd Unix-Syslog-1.1
# perl Makefile.PL
# make
# make install
```

5. 在浏览器上访问，windows 测试需要手动指向 dns 服务器



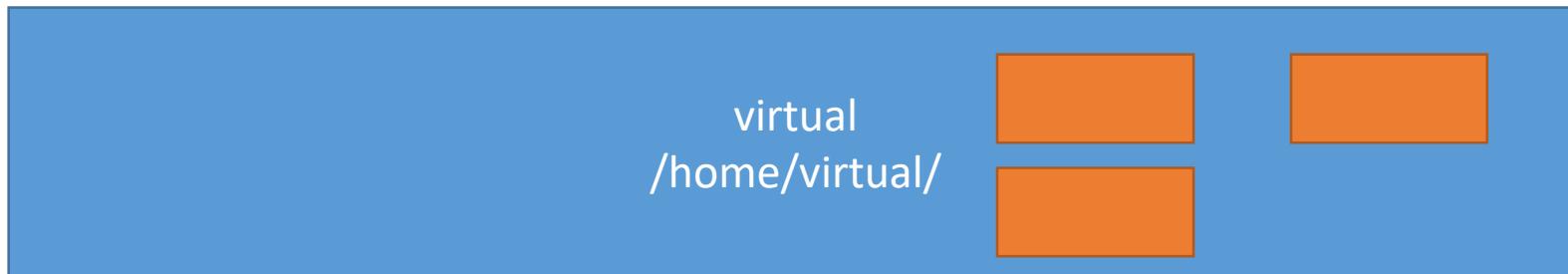
sina用户为发件方→sohu用户为收件方





1. 系统账户之间的收发（利用/etc/passwd）发: telnet 收: 直接查看对应家目录
2. 系统账户之间的收发（利用/etc/passwd）发: telnet 收: telnet (dovecot服务)
3. 数据库账户之间的收发（利用数据库验证）发: echo 收: dovecot (mysql)
4. 数据库账户之间的收发（利用数据库验证）发: 网页 收: 网页: dovecot (mysql)

virtual
/home/virtual/



The diagram illustrates a network structure. At the top is a large blue rectangular box containing the text "virtual" and "/home/virtual/". Inside this box are three smaller orange rectangular blocks arranged in a 2x2 grid (two on top, one on bottom). Below this main box are three smaller blue rectangular boxes, each containing a name: "zhangsan" on the left, "lisi" in the middle, and "laow" on the right. Blue lines connect each of these three lower boxes to the single point at the top center of the main blue box.

zhangsan

lisi

laow

数据库-理论基础

1. 什么是数据库？

数据：描述事物的符号记录，可以是数字、文字、图形、图像、声音、语言等，数据有多种形式，它们都可以经过数字化后存入计算机。

数据库：存储数据的仓库，是长期存放在计算机内、有组织、可共享的大量数据的集合。数据库中的数据按照一定数据模型组织、描述和存储，具有较小的冗余度，较高的独立性和易扩展性，并为各种用户共享，总结为以下几点：

- 数据结构化
- 数据的共享性高，冗余度低，易扩充
- 数据独立性高
- 数据由 DBMS 统一管理和控制（安全性、完整性、并发控制、故障恢复）

解释：DBMS 数据库管理系统（能够操作和管理数据库的大型软件）

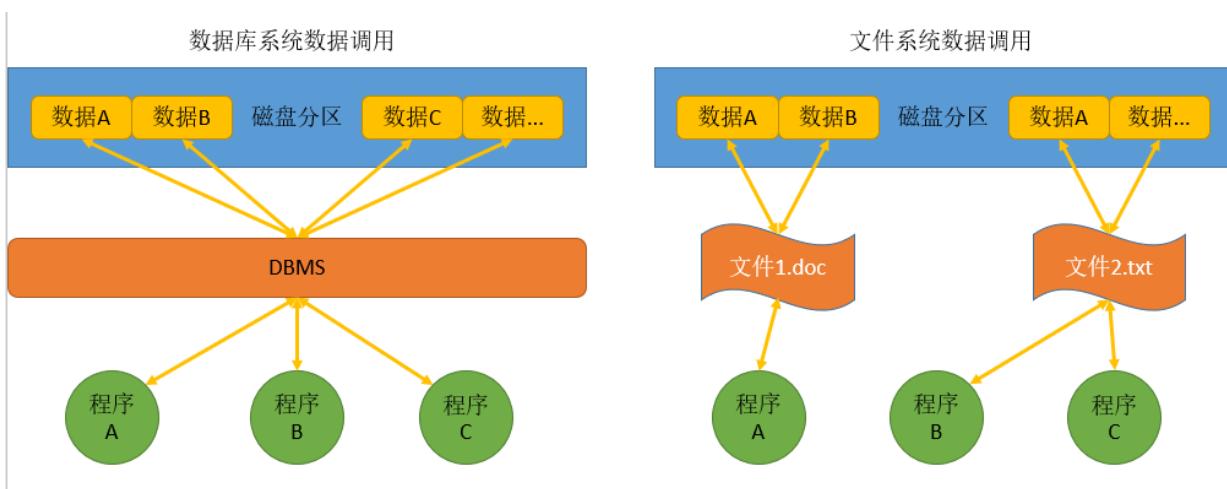
2. 数据库与文件系统的区别？

文件系统：文件系统是操作系统用于明确存储设备（常见的是磁盘）或分区上的文件的方法和数据结构；即在存储设备上组织文件的方法。操作系统中负责管理和存储文件信息的软件机构称为文件管理系统，简称文件系统。

数据库系统：数据库管理系统 (Database Management System) 是一种操纵和管理数据库的大型软件，用于建立、使用和维护数据库，简称 DBMS。它对数据库进行统一的管理和控制，以保证数据库的安全性和完整性。

对比区别：

1. 管理对象不同：**文件系统的管理对象是文件**，并非直接对数据进行管理，不同的数据结构需要使用不同的文件类型进行保存（举例：txt 文件和 doc 文件不能通过修改文件名完成转换）；而数据库直接对数据进行存储和管理
2. 存储方式不同：文件系统使用不同的文件将数据分类（.doc/.mp4/.jpg）保存在外部存储上；数据库系统使用标准统一的数据类型进行数据保存（字母、数字、符号、时间）
3. 调用数据的方式不同：文件系统使用不同的软件打开不同类型的文件；数据库系统由 DBMS 统一调用和管理。如下图：



优缺点总结:

- 由于 DBMS 的存在, 用户不再需要了解数据存储和其他实现的细节, 直接通过 DBMS 就能获取数据, 为数据的使用带来极大便利。
- 具有以数据为单位的共享性, 具有数据的并发访问能力。DBMS 保证了在并发访问时数据的一致性。
- 低延时访问, 典型例子就是线下支付系统的应用, 支付规模巨大的时候, 数据库系统的性能表现远远优于文件系统。
- 能够较为频繁的对数据进行修改, 在需要频繁修改数据的场景下, 数据库系统可以依赖 DBMS 来对数据进行操作且对性能的消耗相比文件系统比较小。
- 对事务的支持。DBMS 支持事务, 即一系列对数据的操作集合要么都完成, 要么都不完成。在 DBMS 上对数据的各种操作都是原子级的。

3. 常见数据库有哪些?

● 关系型数据库

关系数据库是建立在**关系模型**基础上的数据库, 借助于集合代数等数学概念和方法来处理数据库中的数据。现实世界中的各种实体以及实体之间的各种联系均用关系模型来表示。简单说, 关系型数据库是由多张能互相联接的二维行列表格组成的数据库。

关系模型就是指**二维表格模型**, 因而一个关系型数据库就是由二维表及其之间的联系组成的一个数据组织。当前主流的关系型数据库有 Oracle、DB2、Microsoft SQL Server、Microsoft Access、MySQL、浪潮 K-DB 等。

实体关系模型简称**E-R 模型**, 是一套数据库的设计工具, 他运用真实世界中事物与关系的观念, 来解释数据库中的抽象的数据架构。实体关系模型利用图形的方式(实体-关系图)来表示数据库的概念设计, 有助于设计过程中的构思及沟通讨论。

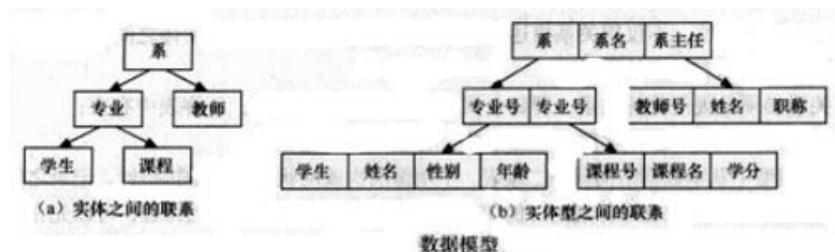
● 非关系型数据库

非关系型数据库: 又被称为**NoSQL (Not Only SQL)**, 意为不仅仅是 SQL, 是一种轻量、开源、不兼容 SQL 功能的数据库, 对 NoSQL 最普遍的定义是“非关联型的”, 强调**Key-Value 存储**和**文档数据库的优点**, 而不是单纯地反对 RDBMS (关系型数据库管理系统)。

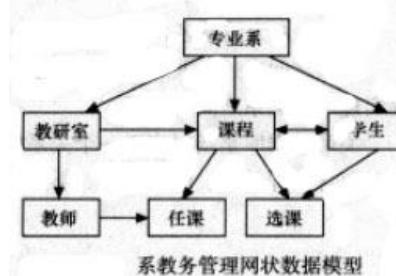
4. 关系型数据库（MySQL）的特征及组成结构介绍

关系型数据库的发展历程

- ### ● 层次模型



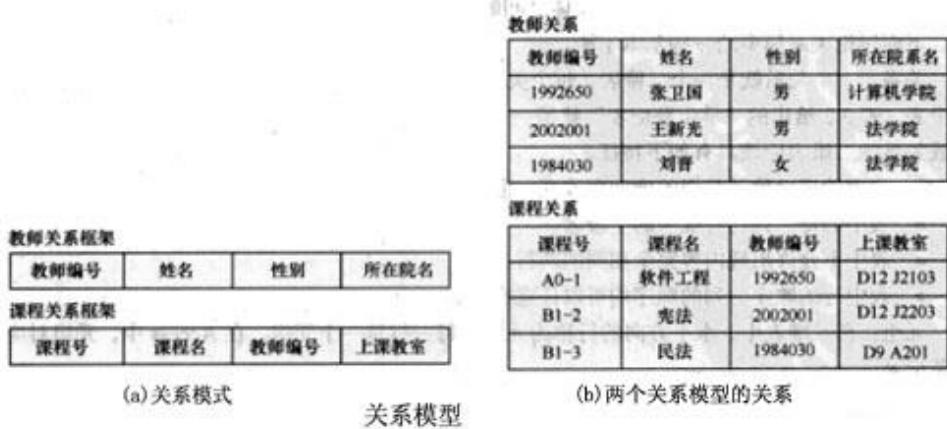
- ### ● 网状模型



关系模型 (Relation)

关系模型以二维表结构来表示实体与实体之间的联系，关系模型的数据结构是一个“二维表框架”组成的集合。每个二维表又可称为关系。在关系模型中，操作的对象和结果都是二维表。

关系模型是目前最流行的数据库模型。支持关系模型的数据库管理系统称为关系数据库管理系统，Access 就是一种关系数据库管理系统。图所示为一个简单的关系模型，其中图(a)所示为关系模式，图(b)所示为这两个关系模型的关系，关系名称分别为教师关系和课程关系，每个关系均含 3 个元组，其主码均为“教师编号”。



在关系模型中基本数据结构就是二维表，不用像层次或网状那样的链接指针。记录之间的联系是通过不同关系中同名属性来体现的。例如，要查找“刘晋”老师所上的课程，可以先在教师关系中根据姓名找到教师编号“1984030”，然后在课程关系中找到“1984030”任课教师编号对应的课程名即可。通过上述查询过程，同名属性教师编号起到了连接两个关系的纽带作用。由此可见，关系模型中的各个关系模式不应当是孤立的，也不是随意拼凑的一堆二维表，它必须满足相应的要求。



关系式数据库的组成结构和名词解释

数据以表格的形式出现，每行为单独的一条记录，每列为一个单独的字段，许多的记录和字段组成一张表单（table），若干的表单组成库（database）

- **记录（一条数据）**

在数据库当中，表当中的行称之为记录

- **字段（id name ...）**

在数据库当中，表当中的列称之为字段

- **MySQL 数据类型**

数据类型用于指定特定字段所包含数据的规则，它决定了数据保存在字段里的方式，包括分配给字段的宽度，以及值是否可以是字母、数字、日期和时间等。任何数据或数据的组合都有对应的数据类型，用于存储字母、数字、日期和时间、图像、二进制数据等。数据类型是数据本身的特征，其特性被设置到表里的字段。

MySQL 常见基础数据类型：

- * 字符串类型（CHAR（0-255 固定长度），VARCHAR（0-255 可变长度））
- * 数值类型（INT（整数型）、FLOAT（浮点型））
- * 日期和时间类型（DATE（年月日）、TIME（时分秒））

- **MySQL 约束类型**

约束是一种限制，它通过对表的行或列的数据做出限制，来确保表的数据的完整性、唯一性。

*** 主键约束 primary key:** 主键约束相当于唯一约束+非空约束的组合，主键约束列不允许重复，也不允许出现空值。每个表最多只允许一个主键，建立主键约束可以在列级别创建，也可以在表级别创建。当创建主键的约束时，系统默认会在所在的列和列组合上建立对应的唯一索引。

*** 外键约束 foreign key:** 外键约束是保证一个或两个表之间的参照完整性，外键是构建于一个表的两个字段或是两个表的两个字段之间的参照关系。

*** 唯一约束 unique:** 唯一约束是指定 table 的列或列组合不能重复，保证数据的唯一性。唯一约束不允许出现重复的值，但是可以为多个 null。同一个表可以有多个唯一约束，多个列组合的约束。在创建唯一约束时，如果不给唯一约束名称，就默认和列名相同。唯一约束不仅可以在一个表内创建，而且可以同时多表创建组合唯一约束。

*** 非空约束 not null 与默认值 default:** 非空约束用于确保当前列的值不为空值，非空约束只能出现在表对象的列上。Null 类型特征：所有的类型的值都可以是 null，包括 int、float 等数据类型

- **MySQL 索引**

索引是一个单独的、物理的数据库结构，它是某个表中一字段或若干字段值的集合。表的存储由两部分组成，一部分用来存放数据，另一部分存放索引页面。通常，索引页面相对于数据页面来说小得多。数据检索花费的大部分开销是磁盘读写，没有索引就需要从磁盘上读表的每一个数据页，如果有索引，则只需查找索引页面就可以了。所以建立合理的索引，就能加速数据的检索过程。

- **MySQL 锁**

数据库是一个多用户使用的共享资源。当多个用户并发地存取数据时，在数据库中就会产生多个事务同时存取同一数据的情况。若对并发操作不加控制就可能会读取和存储不正确的数据，破坏数据

库的一致性。

加锁是实现数据库并发控制的一个非常重要的技术。当事务在对某个数据对象进行操作前，先向系统发出请求，对其加锁。加锁后事务就对该数据对象有了一定的控制，在该事务释放锁之前，其他的事务不能对此数据对象进行更新操作。

- MySQL 的存储引擎

存储引擎就是存储数据，建立索引，更新查询数据等等技术的实现方式。存储引擎是基于表的，而不是基于库的。所以存储引擎也可被称为表类型。Oracle, SqlServer 等数据库只有一种存储引擎。MySQL 提供了插件式的存储引擎架构。所以 MySQL 存在多种存储引擎，可以根据需要使用相应引擎，或者编写存储引擎。

MYISAM：默认引擎、插入和查询速度较快，支持全文索引，不支持事务、行级锁和外键约束等功能

INNODB：支持事务、行级锁和外键约束等功能

MEMORY：工作在内存中，通过散列字段保存数据，速度快、不能永久保存数据

.....

- 事务（Transaction）是并发控制的基本单位。

可以把一系列要执行的操作称为事务，而事务管理就是管理这些操作要么完全执行，要么完全不执行

经典案例：银行转账工作，从一个账号扣款并使另一个账号增款，这两个操作要么都执行，要么都不执行。所以，应该把它们看成一个事务。事务是数据库维护数据一致性的单位，在每个事务结束时，都能保持数据一致性。

注意：mysql 中并不是所有的数据引擎都支持事务管理的，只有 innodb 支持事务管理。

MySQL 数据库-操作基础

MySQL 官网: <https://www.mysql.com/>

1. MySQL 常见版本

- MySQL Community Server 社区版，开源免费，但不提供官方技术支持。
- MySQL Enterprise Edition 企业版，需付费，可以试用 30 天。
- MySQL Cluster 集群版，开源免费。可将几个 MySQL Server 封装成一个 Server。
- MySQL Cluster CGE 高级集群版，需付费

2. MySQL 安装部署

MySQL: MySQL 客户端程序

MySQL-Server: MySQL 服务器端程序

源代编译安装:

编译工具: configure、cmake、make
数据库常用的配置选项

| | |
|-----------------------------------|-----------------------------------|
| -DCMAKE_INSTALL_PREFIX=/PREFIX | ----指定安装路径（默认的就是/usr/local/mysql） |
| -DMYSQL_DATADIR=/data/mysql | ----mysql 的数据文件路径 |
| -DSYSCONFDIR=/etc | ----配置文件路径 |
| -DWITH_INNODB_STORAGE_ENGINE=1 | ----使用 INNODB 存储引擎 |
| -DWITH_READLINE=1 | ----支持批量导入 mysql 数据 |
| -DWITH_SSL=system | ----mysql 支持 ssl |
| -DWITH_ZLIB=system | ----支持压缩存储 |
| -DMYSQL_TCP_PORT=3306 | ----默认端口 3306 |
| -DENABLED_LOCAL_INFILE=1 | ----启用加载本地数据 |
| -DMYSQL_USER=mysql | ----指定 mysql 运行用户 |
| -DMYSQL_UNIX_ADDR=/tmp/mysql.sock | ----默认套接字文件路径 |
| -DEXTRA_CHARSETS=all | ----是否支持额外的字符集 |
| -DDEFAULT_CHARSET=utf8 | ----默认编码机制 |
| -DWITH_DEBUG=0 | ----DEBUG 功能设置 |

常见资料:

服务: mysqld

端口: 3306

主配置文件: /etc/my.cnf

初始化脚本: mysql_install_db

启动命令: mysqld_safe

数据目录 : /var/lib/mysql

套接字文件: /var/lib/mysql/mysql.sock

#当意外关闭数据库时，再开启时假如开启不了，找到这个，删除再启动
进程文件：/var/run/mysqld/mysqld.pid

MySQL 登录及退出命令：

设置密码：mysqladmin -uroot password ‘123456’
登录：mysql -u 用户名 -p 密码 -P 端口 -S 套接字文件
-p 用户密码
-h 登陆位置（主机名或 ip 地址）
-P 端口号（3306 改了就不是了）
-S 套接字文件（/var/lib/mysql/mysql.sock）
退出命令：exit 或 ctrl+d

3. MySQL 管理命令

1. 创建登录用户

```
mysql>create user zhangsan@ '%' identified by '123456';  
%:指任意的远程终端
```

2. 测试用户登录

```
# yum -y install mysql  
# mysql -uzhangsan -p123456 -h 192.168.88.10
```

3. 用户为自己更改密码

```
mysql>set password=password( '123456' );
```

4. root 用户为其他用户找回密码

```
mysql>set password for atguigu@ '%' =password( '123123' );
```

5. root 找回自己的密码并修改

关闭数据库，修改主配置文件（/etc/my.cnf）添加：skip-grant-tables

```
# vim /etc/my.cnf  
skip-grant-tables
```

启动数据库，空密码登录并修改密码

```
update mysql.user set password=password( '新密码' ) where user='root' ;
```

删除 skip-grant-tables，重启数据库验证新密码

6. 创建查询数据库

```
mysql>create database web;  
mysql>show databases;
```

7. 创建数据表

```
Mysql>use web;
```



#选择要使用的数据库

```
Mysql>create table a1 (id int ,name char(30));
```

#创建 a1 表，并添加 id 和 name 字段以及类型

```
Mysql>describe a1;
```

#查看表结构（字段）

复杂一点的

```
Mysql>create table a2 (
```

```
    ->id int unsigned not null auto_increment,      #字段要求为正数、且自增长、主键  
    ->name char(30) not null default ' ',          #字符型长度 30 字节，默认值为空格  
    ->age int not null default 0,                  #字段默认值为 0  
    ->primary key (id));                          #设置 id 为主键
```

```
Mysql> describe a2;
```

8. 插入数据

```
Mysql>insert into a2 (id,name,age) values (1, 'zhangsan',21); #指明插入字段和数据
```

```
Mysql>select * from a2;
```

```
Mysql>insert into a2 values (2, 'lisi',20);           #按顺序插入指定字段
```

```
Mysql>insert into a2 values (3, 'wangwu');            #未声明年龄
```

```
Mysql>insert into a2 values (4, 'zhao',19), (5, 'sun',25); #插入多条数据
```

9. 将表 a2 的数据复制到表 a1

```
Mysql>select * from a1;
```

```
Mysql>insert into a1 (id,name) select id,name from a2;
```

#查询 a2 值，并写入到 a1

```
Mysql>select * from a1;
```

10. 删除数据库

```
Mysql>drop database abc;
```

```
Mysql>show databases;
```

11. 删除数据表

```
Mysql>drop table a1;
```

```
Mysql>show table;
```

12. 删除表里的数据记录

```
Mysql>delete from a2 where id=5;                      #删除 id=5 的记录
```

```
Mysql>delete from a2 where between 23 and 25;        #删除年龄在 23-25 之间的
```

注：库和表的删除用 drop, 记录删除用 delete

13. 修改表中的数据

```
Mysql>update a2 set age=21 where id=3;
```

14. 修改数据表的名称

```
Mysql>alter table a2 rename a1;
```

15. 修改数据表的字段类型

```
Mysql>describe a1;  
Mysql>alter table a1 modify name char(50);  
Mysql>describe a1;
```

16. 修改数据表的字段类型详情

```
Mysql>describe a1;  
Mysql>alter table a1 change name username char(50) not null default '' ;  
Mysql>describe a1;
```

17. 添加字段

```
Mysql>describe a1;  
Mysql>alter table a1 add time datetime;  
Mysql>describe a1;  
#添加位置默认在末尾  
Mysql>alter table a1 add birthday year first;          #添加字段到第一列  
Mysql>alter table a1 add sex nchar(1) after id;        #添加到指定字段后
```

18. 删除字段

```
Mysql>alter table a1 drop birthday;
```

19. Mysql 用户授权

授予用户全部权限

```
Mysql>select user from mysql.user;  
Mysql>grant all on aa.a1 to atguigu@ '%' ;      #给已存在用户授权  
Mysql>grant all on aa.a1 to abc@ '%' identified by '123456' ;    #创建用户并授权
```

取消 abc 用户的删除库、表、表中数据的权限

```
Mysql>revoke drop,delete on aa.a1 from abc@ '%' ;      #取消删除权限（登录 abc 测试）  
Mysql>show grants for abc@ '%' ;                      #查看指定用户的授权  
Mysql>show grants for atguigu@ '%' ;
```

4. 备份和还原

mysqldump 备份：

备份：

```
mysqldump -u 用户名 -p 数据库名 > /备份路径/备份文件名 (备份整个数据库)
```



```
mysqldump -u 用户名 -p 数据库名 表名 > /备份路径/备份文件名 (备份数据表)
```

备份多个库: --databases 库 1, 库 2

备份所有库: --all-databases

备份多个表: 库名 表 1 表 2

还原: mysql 数据库 < 备份文件

注意: 还原时, 若导入的是某表, 请指定导入到哪一个库中

mysqlhotcopy 备份:

```
备份: mysqlhotcopy --flushlog -u=' 用户' -p=' 密码' --regexp=正则 备份目录
```

```
还原: cp -a 备份目录 数据目录 (/var/lib/mysql)
```

mysqldump 和 mysqlhotcopy 示例:

Mysql 备份和还原

把数据库 aa 备份到/root 目录下

```
# mysqldump -uroot -p aa > ~/aa.sql
```

模拟数据库 aa 丢失 (删除数据库 aa)

```
# Mysql>drop database aa;
```

通过 aa.sql 文件还原 (指定导入到哪个库中)

```
# mysql -uroot -p test < aa.sql
```

备份多个数据库 (--databases)

```
# mysqldump -uroot -p --databases aa test > abc.sql
```

还原 (先模拟丢失)

```
# mysql -uroot -p < abc.sql
```

备份有规则的数据库

```
Mysql>create database a1; #连续创建三个 a 开头的数据库
```

```
# mysqlhotcopy --flushlog -u='root' -p='456' --regexp='^a'
```

还原 (先模拟丢失)

```
Mysql>drop database a1; #顺序删除 a 开头的数据库
```

```
# cp -a /mnt/* /var/lib/mysql/ #复制产生的文件到数据库目录下
```

```
#登录数据库查看即可
```

mysql-binlog 日志备份:

二进制日志 (log-bin 日志) : 所有对数据库状态更改的操作 (create、drop、update 等)

修改 my.cnf 配置文件开启 binlog 日志记录功能

```
# vim /etc/my.cnf
```

```
log-bin=mysql-bin #启动二进制日志
```

按时间还原:

```
--start-datetime
```

```
--stop-datetime
```

格式: mysqlbinlog --start-datetime 'YY-MM-DD HH:MM:SS' --stop-datetime 'YY-MM-DD'

HH:MM:SS' 二进制日志 | mysql -uroot -p

按文件大小还原:

--start-position

--stop-position

mysql-binlog 日志备份示例:

开启二进制日志

```
[mysqld]
datadir=/var/lib/mysql
socket=/var/lib/mysql/mysql.sock
user=mysql
# Disabling symbolic-links is recommended to prevent assorted security risks
symbolic-links=0
general-log=ON
log=ON
log-slow-queries=/var/log/mysql-slow.log
log-bin=mysql-bin 在主配置文件中添加
[mysqld_safe]
```

查看二进制日志文件

```
[root@localhost ~]# cd /var/lib/mysql/
[root@localhost mysql]# ls
aa  cc  ib_logfile0  mysql  mysql-bin.index  ON
bb  ibdata1  ib_logfile1  mysql-bin.000001  mysql.sock  test
[root@localhost mysql]# 在数据目录下会产生二进制日志文件
[root@localhost mysql]#
[root@localhost mysql]#
```

按时间还原:

如果数据库中的 bb 库被删, 需要还原

```
mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| aa           |
| cc           |
| mysql        |
| test         |
+-----+
5 rows in set (0.00 sec)
```

查看二进制日志内容

```
/*!*/;          mysqlbinlog mysql-bin.000001 查看二进制日志的内容如下一部
# at 185      分:
#170225  0:35:40 server id 1  end_log_pos 264    Query    thread_id=2    e
exec_time=0    error_code=0
SET TIMESTAMP=1487954140/*!*/;
create database bb    找到丢失的地方, bb库被删, 那么找到创建它的时间和删除时的
/*!*/;          时间
# at 264
#170225  0:35:53 server id 1  end_log_pos 341    Query    thread_id=2    e
exec_time=0    error_code=0
SET TIMESTAMP=1487954153/*!*/;
drop database bb
/*!*/;
```

还原并查看

```
mysqlbinlog --start-datetime='2018-09-11 14:24:00' --stop-datetime='2018-09-11 14:28:00'  
mysql-bin.000006 | mysql -uroot -p123123  
注：所选时间段一定要完整包含所有动作（可以在原来基础上稍微增加点时间）
```

按文件大小还原：还原到 bb 库被删除的数据状态

1. 查看 bb 库被删除前后的文件大小

```
# at 264  
#170225 0:35:53 server id 1 end_log_pos 341    Query    thread_id=2      e  
exec_time=0      error_code=0  
SET TIMESTAMP=1487954153/*!*/;  
drop database bb  
/*!*/;  
# at 341  
#170225 0:36:16 server id 1 end_log_pos 420    Query    thread_id=2      e  
exec_time=0      error_code=0
```

还原并查看

MySQL 数据类型

在 MySQL 中，有三种主要的类型：文本、数字和日期/时间类型。

Text 类型：

| 数据类型 | 描述 |
|---------------------|--|
| CHAR(size) | 保存固定长度的字符串（可包含字母、数字以及特殊字符）。在括号中指定字符串的长度。最多 255 个字符。 |
| VARCHAR(size) | 保存可变长度的字符串（可包含字母、数字以及特殊字符）。在括号中指定字符串的最大长度。最多 255 个字符。

注释：如果值的长度大于 255，则被转换为 TEXT 类型。 |
| TINYTEXT | 存放最大长度为 255 个字符的字符串。 |
| TEXT | 存放最大长度为 65,535 个字符的字符串。 |
| BLOB | 用于 BLOBs (Binary Large OBjects)。存放最多 65,535 字节的数据。 |
| MEDIUMTEXT | 存放最大长度为 16,777,215 个字符的字符串。 |
| MEDIUMBLOB | 用于 BLOBs (Binary Large OBjects)。存放最多 16,777,215 字节的数据。 |
| LONGTEXT | 存放最大长度为 4,294,967,295 个字符的字符串。 |
| LONGBLOB | 用于 BLOBs (Binary Large OBjects)。存放最多 4,294,967,295 字节的数据。 |
| ENUM(x, y, z, etc.) | 允许你输入可能值的列表。可以在 ENUM 列表中列出最大 65535 个值。如果列表中不存在插入的值，则插入空值。

注释：这些值是按照你输入的顺序存储的。

可以按照此格式输入可能的值： ENUM('X', 'Y', 'Z') |
| SET | 与 ENUM 类似，SET 最多只能包含 64 个列表项，不过 SET 可存储一个以上的值。 |

Number 类型：

| 数据类型 | 描述 |
|------------------|---|
| TINYINT(size) | -128 到 127 常规。0 到 255 无符号*。在括号中规定最大位数。 |
| SMALLINT(size) | -32768 到 32767 常规。0 到 65535 无符号*。在括号中规定最大位数。 |
| MEDIUMINT(size) | -8388608 到 8388607 普通。0 to 16777215 无符号*。在括号中规定最大位数。 |
| INT(size) | -2147483648 到 2147483647 常规。0 到 4294967295 无符号*。在括号中规定最大位数。 |
| BIGINT(size) | -9223372036854775808 到 9223372036854775807 常规。0 到 18446744073709551615 无符号*。在括号中规定最大位数。 |
| FLOAT(size, d) | 带有浮动小数点的小数字。在括号中规定最大位数。在 d 参数中规定小数点右侧的最大位数。 |
| DOUBLE(size, d) | 带有浮动小数点的大数字。在括号中规定最大位数。在 d 参数中规定小数点右侧的最大位数。 |
| DECIMAL(size, d) | 作为字符串存储的 DOUBLE 类型，允许固定的小数点。 |

* 这些整数类型拥有额外的选项 UNSIGNED。通常，整数可以是负数或正数。如果添加 UNSIGNED 属性，那么范围将从 0 开始，而不是某个负数。

Date 类型:

| 数据类型 | 描述 |
|-------------|--|
| DATE() | 日期。格式：YYYY-MM-DD

注释：支持的范围是从 '1000-01-01' 到 '9999-12-31' |
| DATETIME() | *日期和时间的组合。格式：YYYY-MM-DD HH:MM:SS

注释：支持的范围是从 '1000-01-01 00:00:00' 到 '9999-12-31 23:59:59' |
| TIMESTAMP() | *时间戳。TIMESTAMP 值使用 Unix 纪元('1970-01-01 00:00:00' UTC) 至今的描述来存储。格式：YYYY-MM-DD HH:MM:SS |

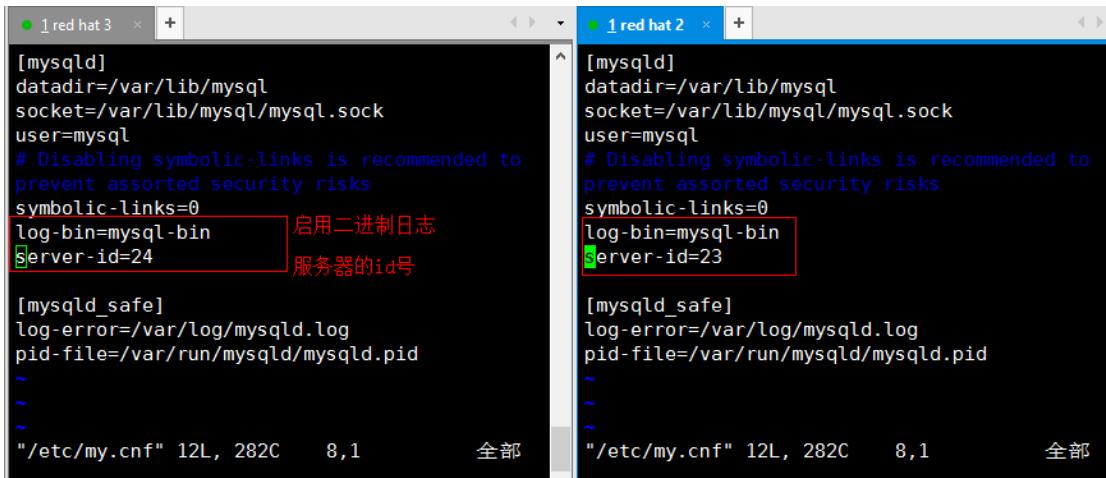
| | |
|--------|---|
| | 注释：支持的范围是从 ’1970-01-01 00:00:01’ UTC 到
’2038-01-09 03:14:07’ UTC |
| TIME() | 时间。格式：HH:MM:SS 注释：支持的范围是从 ’-838:59:59’
到 ’838:59:59’ |
| YEAR() | 2 位或 4 位格式的年。

注释：4 位格式所允许的值：1901 到 2155。2 位格式所允许
的值：70 到 69，表示从 1970 到 2069 |

MySQL 集群

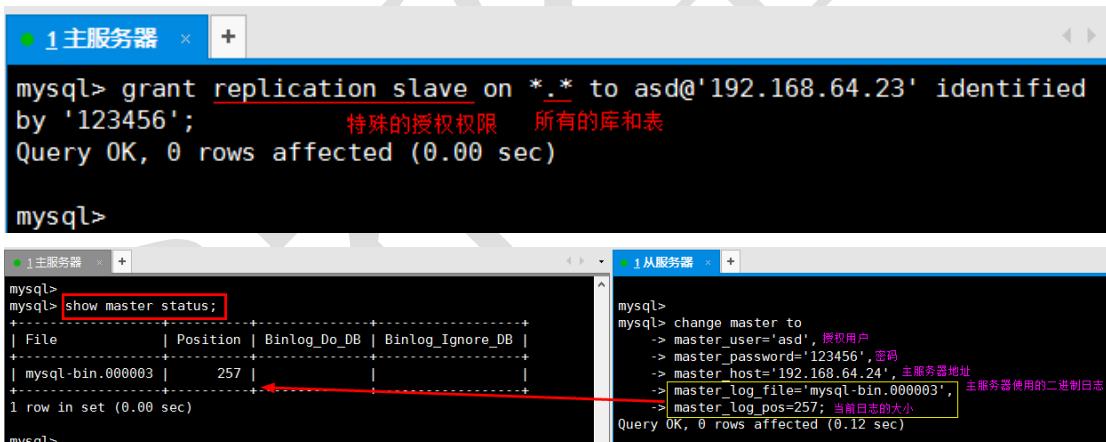
1. MySQL 主从备份

前提条件：安装了 mysql，开启了二进制日志



The image shows two terminal windows side-by-side. Both windows display the MySQL configuration file (`/etc/my.cnf`). The left window is for the master server (IP 192.168.64.24) and the right window is for the slave server (IP 192.168.64.23). In both files, the `log-bin=mysql-bin` and `server-id=24` (master) and `server-id=23` (slave) lines are highlighted with red boxes. A callout box labeled "启用二进制日志" (Enable Binary Log) points to the `log-bin` line, and another callout box labeled "服务器的id号" (Server's ID) points to the `server-id` line.

在主服务器上授权，从服务器保存授权的信息



The image shows three terminal windows. The top window is titled "主服务器" (Master Server) and shows the MySQL command: `grant replication slave on *.* to asd@'192.168.64.23' identified by '123456';`. The middle window is titled "从服务器" (Slave Server) and shows the MySQL command: `show master status;`. The bottom window is also titled "从服务器" and shows the MySQL command: `change master to` followed by parameters: `master_user='asd'`, `master_password='123456'`, `master_host='192.168.64.24'`, `master_log_file='mysql-bin.000003'`, and `master_log_pos=257`. A red arrow points from the "从服务器" window to the "从服务器" window, indicating the flow of information.

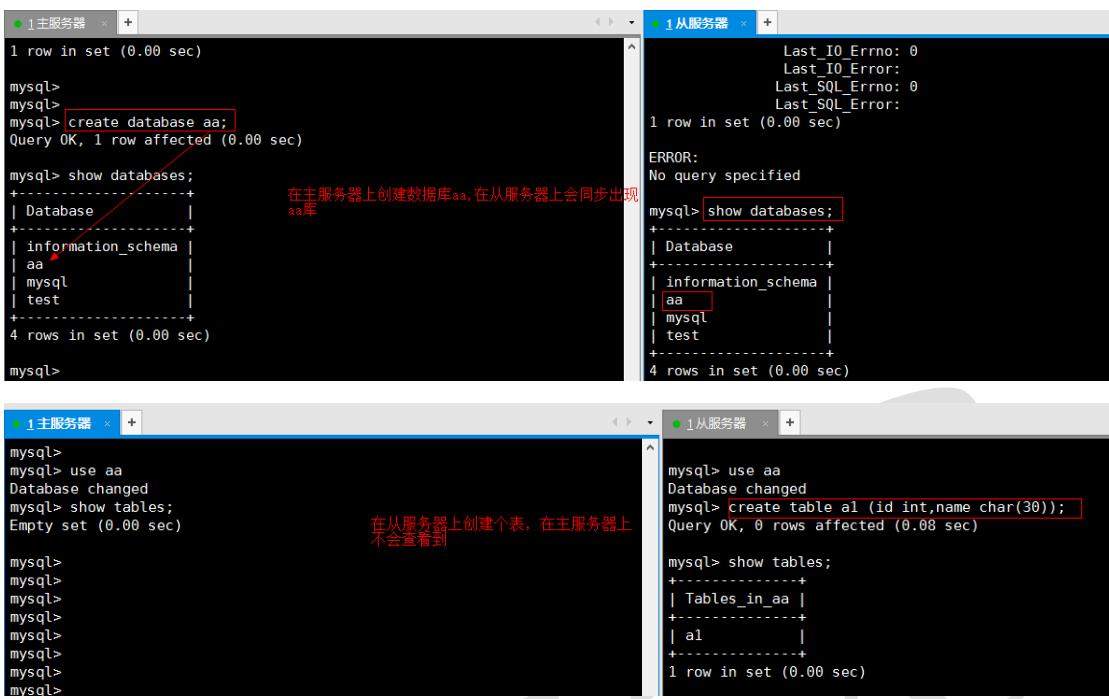
之后在从服务器会产生授权信息文件

```
● 1 从服务器 +  
mysql> exit  
Bye  
[root@localhost mysql]# ls  
aa          mysql-bin.000001      mysqld-relay-bin.index  
ibdata1     mysql-bin.000002      mysql.sock  
ib_logfile0 mysql-bin.000003      relay-log.info  
ib_logfile1 mysql-bin.index       test  
master.info mysqld-relay-bin.000001  
mysql        mysqld-relay-bin.000002  
[root@localhost mysql]# cat master.info  
15  
mysql-bin.000003  
336  
192.168.64.24  
asd  
123456          授权信息内容  
3306  
60  
0  
  
0  
[root@localhost mysql]#
```

开启从服务器 start slave，并查看

```
● 1 从服务器 +  
mysql> start slave; 开启从服务器  
Query OK, 0 rows affected (0.01 sec)  
  
mysql> show slave status\G; 查看从服务器的内容  
*****  
                1. row *****  
    Slave_IO_State: Waiting for master to send event  
              Master_Host: 192.168.64.24  
            Master_User: asd  
            Master_Port: 3306  
      Connect_Retry: 60  
    Master_Log_File: mysql-bin.000003  
  Read_Master_Log_Pos: 257  
      Relay_Log_File: mysqld-relay-bin.000002  
      Relay_Log_Pos: 251  
Relay_Master_Log_File: mysql-bin.000003  
    Slave_IO_Running: Yes          线程已启用  
   Slave_SQL_Running: Yes  
     Replicate_Do_DB:  
   Replicate_Ignore_DB:  
  Replicate_Do_Tables:  
Replicate_Ignore_Tables:  
Replicate_Wild_Do_Table:  
Replicate_Wild_Ignore_Table:  
Last_Errno:  
Last_Error:  
Last_SQL_Error:
```

测试



```

1 主服务器 > + 1 从服务器 > +
1 row in set (0.00 sec)
mysql>
mysql>
mysql> create database aa;
Query OK, 1 row affected (0.00 sec)

mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| aa          |
| mysql       |
| test        |
+-----+
4 rows in set (0.00 sec)

mysql>

1 从服务器 > + 1 从服务器 > +
Last_IO_Error: 0
Last_IO_Error:
Last_SQL_Error: 0
Last_SQL_Error:
1 row in set (0.00 sec)

ERROR:
No query specified

mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| aa          |
| mysql       |
| test        |
+-----+
4 rows in set (0.00 sec)

1 主服务器 > + 1 从服务器 > +
mysql>
mysql> use aa
Database changed
mysql> show tables;
Empty set (0.00 sec)

mysql>
mysql>
mysql>
mysql>
mysql>
mysql>
mysql>
mysql>

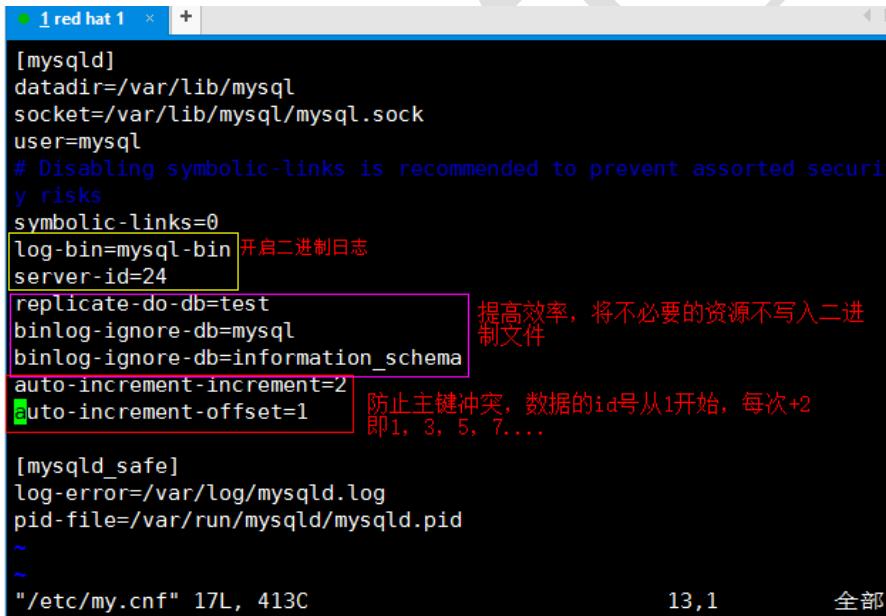
1 从服务器 > + 1 从服务器 > +
mysql> use aa
Database changed
mysql> create table a1 (id int, name char(30));
Query OK, 0 rows affected (0.08 sec)

mysql> show tables;
+-----+
| Tables_in_aa |
+-----+
| a1           |
+-----+
1 row in set (0.00 sec)

```

2. MySQL 主主备份

- 以 1 为主，2 为从配置一遍主从
在主配置文件中配置一下（开启二进制日志和其他内容）



```

[mysqld]
datadir=/var/lib/mysql
socket=/var/lib/mysql/mysql.sock
user=mysql
# Disabling symbolic-links is recommended to prevent assorted security risks
symbolic-links=0
log-bin=mysql-bin 开启二进制日志
server-id=24
replicate-do-db=test
binlog-ignore-db=mysql
binlog-ignore-db=information_schema 提高效率, 将不必要的资源不写入二进制文件
auto-increment-increment=2 防止主键冲突, 数据的id号从1开始, 每次+2即1, 3, 5, 7...
auto-increment-offset=1

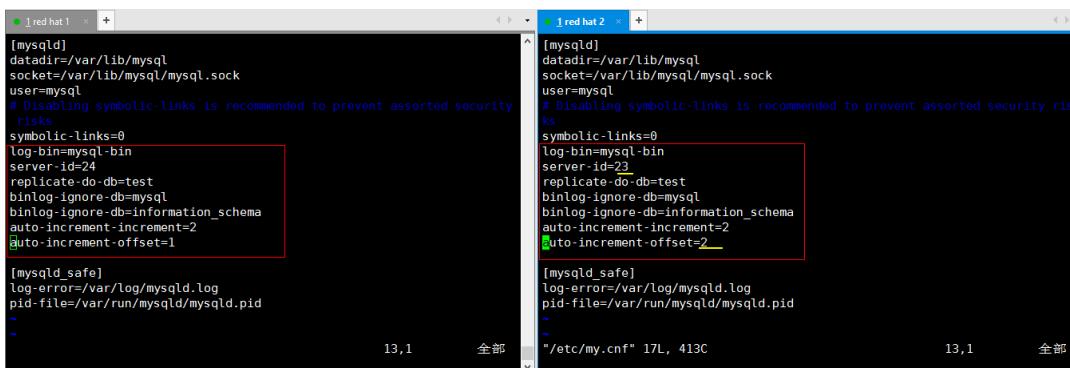
[mysqld_safe]
log-error=/var/log/mysqld.log
pid-file=/var/run/mysqld/mysqld.pid
~

"/etc/my.cnf" 17L, 413C

```

13, 1 全部

- 在 2 上做相同的配置



```
[mysql]
datadir=/var/lib/mysql
socket=/var/lib/mysql/mysql.sock
user=mysql
# Disabling symbolic-links is recommended to prevent assorted security risks
symbolic-links=0
log-bin=mysql-bin
server-id=24
replicate-do-db=test
binlog-ignore-db=mysql
binlog-ignore-db=information_schema
auto-increment-increment=2
auto-increment-offset=1

[mysqld_safe]
log-error=/var/log/mysqld.log
pid-file=/var/run/mysqld/mysqld.pid
~
~

[mysql]
datadir=/var/lib/mysql
socket=/var/lib/mysql/mysql.sock
user=mysql
# Disabling symbolic-links is recommended to prevent assorted security risks
symbolic-links=0
log-bin=mysql-bin
server-id=23
replicate-do-db=test
binlog-ignore-db=mysql
binlog-ignore-db=information_schema
auto-increment-increment=2
auto-increment-offset=2

[mysqld_safe]
log-error=/var/log/mysqld.log
pid-file=/var/run/mysqld/mysqld.pid
~
~

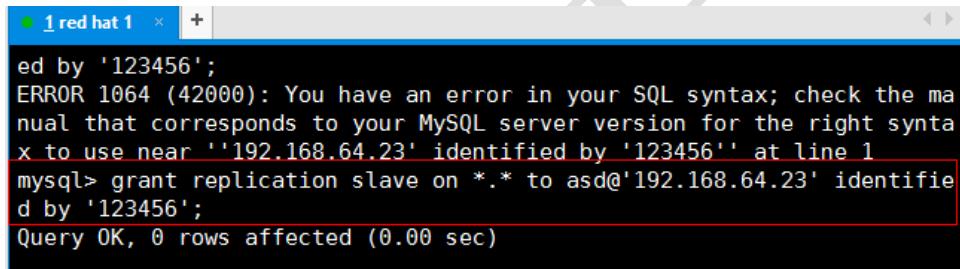
"/etc/my.cnf" 17L, 413C
```

3. 启动服务器

```
[root@localhost mysql]# vim /etc/my.cnf
[root@localhost ~]# service mysqld start
初始化 MySQL 数据库: Installing MySQL system tables...
OK
Filling help tables...
OK
```

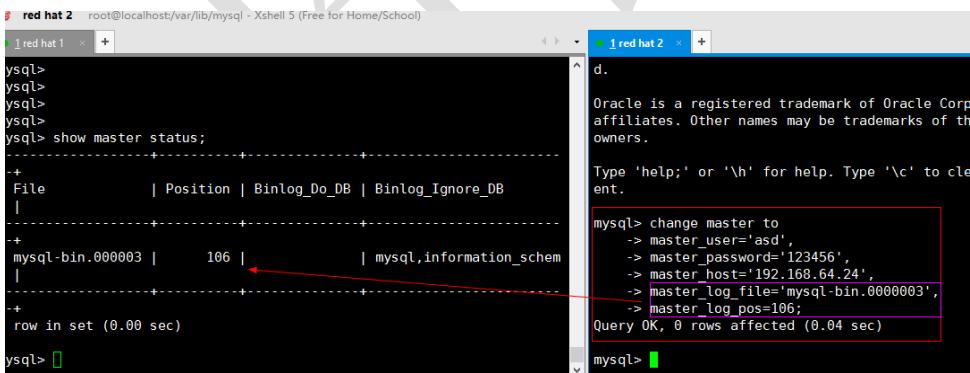
1为主 2为从:

在主服务器(1)上授权



```
ed by '123456';
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near ''192.168.64.23' identified by '123456'' at line 1
mysql> grant replication slave on *.* to asd@'192.168.64.23' identified by '123456';
Query OK, 0 rows affected (0.00 sec)
```

在从服务器(2)上保存授权信息



```
mysql>
mysql>
mysql>
mysql>
mysql> show master status;
+-----+-----+-----+
| File | Position | Binlog_Do_DB | Binlog_Ignore_DB |
+-----+-----+-----+
| mysql-bin.000003 | 106 | | mysql,information_schema |
+-----+-----+-----+
1 row in set (0.00 sec)

mysql> change master to
-> master_user='asd',
-> master_password='123456',
-> master_host='192.168.64.24',
-> master_log_file='mysql-bin.000003',
-> master_log_pos=106;
Query OK, 0 rows affected (0.04 sec)
```

2为主 1为从:

在主服务器(2)上授权

```

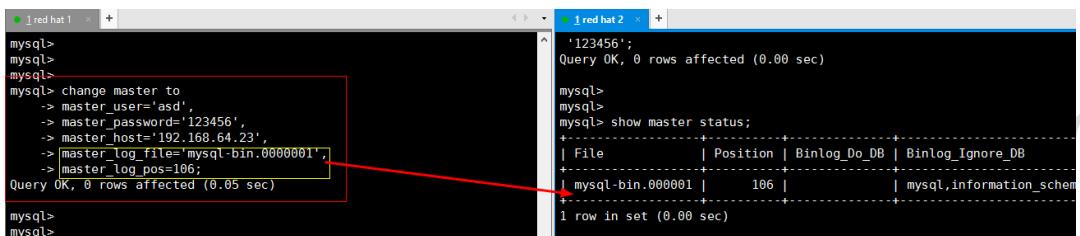
1 red hat 2 + 
    -> master_log_pos=106;
Query OK, 0 rows affected (0.04 sec)

mysql> grant replication slave on *.* to asd@'192.168.64.24' identified by
'123456';
Query OK, 0 rows affected (0.00 sec)

mysql>
mysql>

```

在从服务器(1)上保存授权信息



```

1 red hat 1 + 
mysql>
mysql>
mysql> change master to
-> master_user='asd',
-> master_password='123456',
-> master_host='192.168.64.23',
-> master_log_file='mysql-bin.000001',
-> master_log_pos=106;
Query OK, 0 rows affected (0.05 sec)

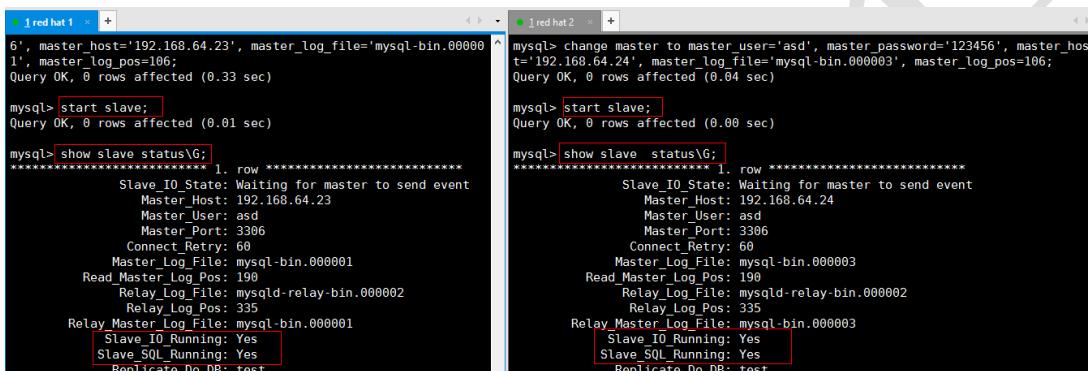
mysql>
mysql>

1 red hat 2 + 
'123456';
Query OK, 0 rows affected (0.00 sec)

mysql>
mysql> show master status;
+-----+-----+-----+
| File | Position | Binlog_Do_DB | Binlog_Ignore_DB |
+-----+-----+-----+
| mysql-bin.000001 | 106 | mysql | mysql.information_schema |
+-----+-----+-----+
1 row in set (0.00 sec)

```

1 和 2 都执行 start slave (互为主从)



```

1 red hat 1 + 
6', master_host='192.168.64.23', master_log_file='mysql-bin.000000
1', master_log_pos=106;
Query OK, 0 rows affected (0.33 sec)

mysql> start slave;
Query OK, 0 rows affected (0.01 sec)

mysql> show slave status\G;
***** 1. row *****
Slave_IO_State: Waiting for master to send event
Master_Host: 192.168.64.23
Master_User: asd
Master_Port: 3306
Connect Retry: 60
Master_Log_File: mysql-bin.000001
Read_Master_Log_Pos: 190
Relay_Log_File: mysqld-relay-bin.000002
Relay_Log_Pos: 335
Relay_Master_Log_File: mysql-bin.000001
Slave_IO_Running: Yes
Slave_SQL_Running: Yes
Replicate_Do_DB: test

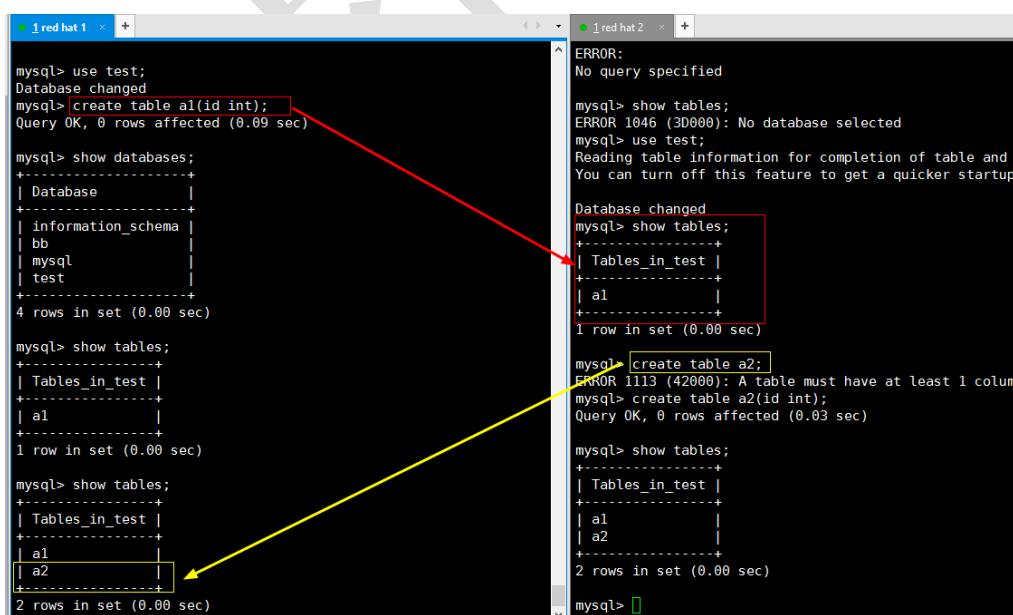
1 red hat 2 + 
mysql> change master to master_user='asd', master_password='123456', master_host='192.168.64.24', master_log_file='mysql-bin.000003', master_log_pos=106;
Query OK, 0 rows affected (0.04 sec)

mysql> start slave;
Query OK, 0 rows affected (0.00 sec)

mysql> show slave status\G;
***** 1. row *****
Slave_IO_State: Waiting for master to send event
Master_Host: 192.168.64.24
Master_User: asd
Master_Port: 3306
Connect Retry: 60
Master_Log_File: mysql-bin.000003
Read_Master_Log_Pos: 190
Relay_Log_File: mysqld-relay-bin.000002
Relay_Log_Pos: 335
Relay_Master_Log_File: mysql-bin.000003
Slave_IO_Running: Yes
Slave_SQL_Running: Yes
Replicate_Do_DB: test

```

测试



```

1 red hat 1 + 
mysql> use test;
Database changed
mysql> create table a1(id int);
Query OK, 0 rows affected (0.09 sec)

mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| bb |
| mysql |
| test |
+-----+
4 rows in set (0.00 sec)

mysql> show tables;
+-----+
| Tables_in_test |
+-----+
| a1 |
+-----+
1 row in set (0.00 sec)

mysql> show tables;
+-----+
| Tables_in_test |
+-----+
| a1 |
+-----+
1 row in set (0.00 sec)

mysql> show tables;
+-----+
| Tables_in_test |
+-----+
| a1 |
| a2 |
+-----+
2 rows in set (0.00 sec)

1 red hat 2 + 
ERROR:
No query specified

mysql> show tables;
ERROR 1046 (3D000): No database selected
mysql> use test;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup

Database changed
mysql> show tables;
+-----+
| Tables_in_test |
+-----+
| a1 |
+-----+
1 row in set (0.00 sec)

mysql> create table a2;
EROR 1113 (42000): A table must have at least 1 column
mysql> create table a2(id int);
Query OK, 0 rows affected (0.03 sec)

mysql> show tables;
+-----+
| Tables_in_test |
+-----+
| a1 |
| a2 |
+-----+
2 rows in set (0.00 sec)

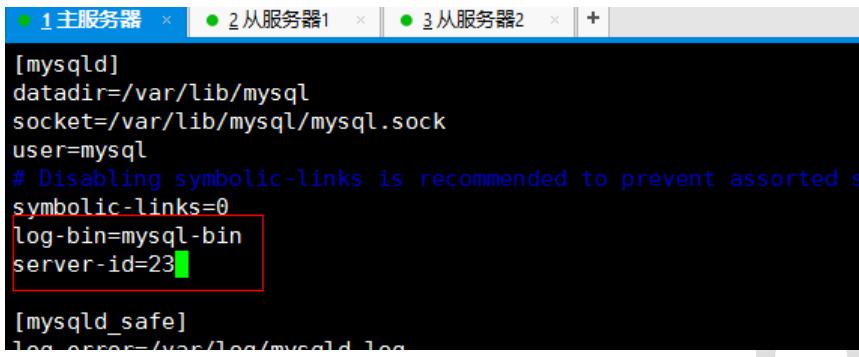
mysql>

```

3. MySQL 一主多从

主服务器配置

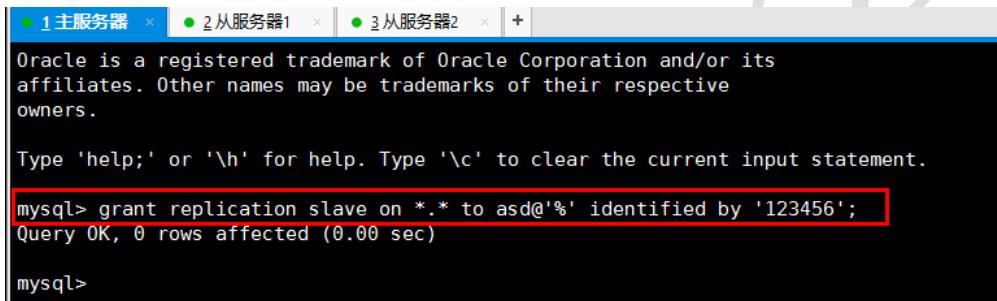
开启二进制日志，并启动 mysql



```
[mysqld]
datadir=/var/lib/mysql
socket=/var/lib/mysql/mysql.sock
user=mysql
# Disabling symbolic-links is recommended to prevent assorted security risks
symbolic-links=0
log-bin=mysql-bin
server-id=23

[mysqld_safe]
log-error=/var/log/mysqld.log
```

在主服务器上授权



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

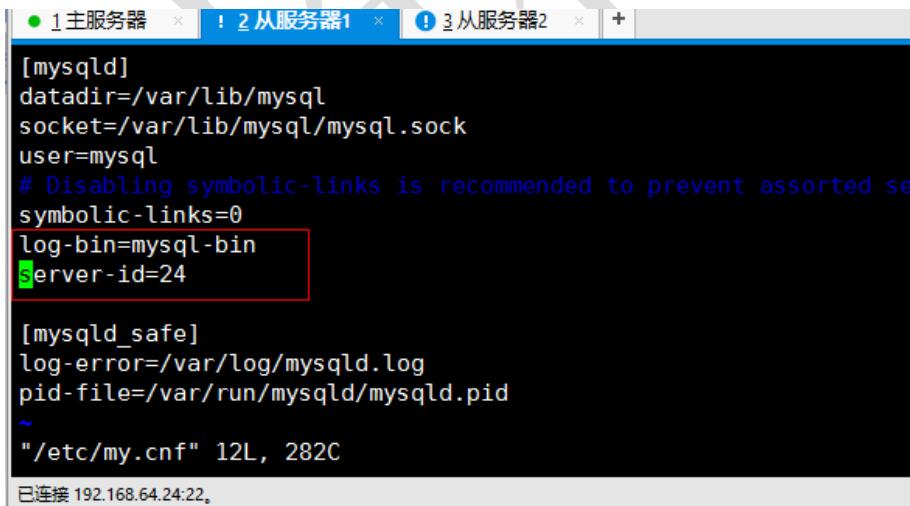
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> grant replication slave on *.* to asd@'%' identified by '123456';
Query OK, 0 rows affected (0.00 sec)

mysql>
```

从服务器配置

开启二进制日志，并启动 mysql

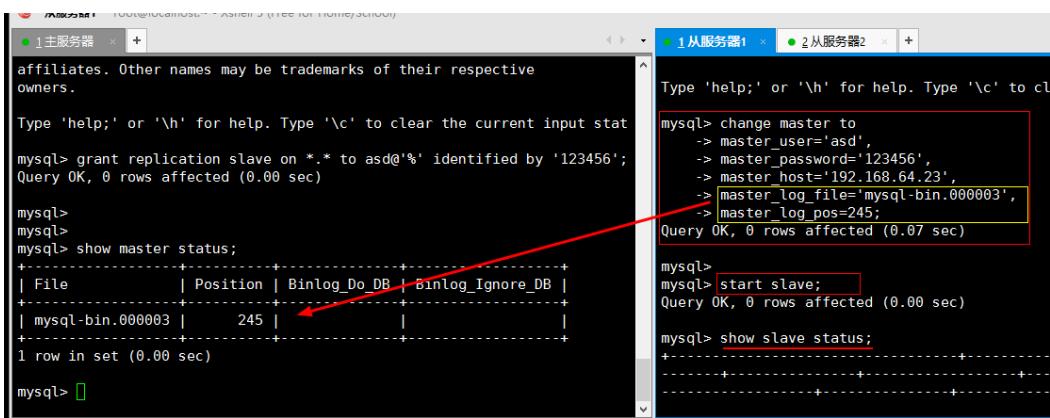


```
[mysqld]
datadir=/var/lib/mysql
socket=/var/lib/mysql/mysql.sock
user=mysql
# Disabling symbolic-links is recommended to prevent assorted security risks
symbolic-links=0
log-bin=mysql-bin
server-id=24

[mysqld_safe]
log-error=/var/log/mysqld.log
pid-file=/var/run/mysqld/mysqld.pid
^
"/etc/my.cnf" 12L, 2820

已连接 192.168.64.24:22,
```

保存授权信息



```

1 主服务器
affiliates. Other names may be trademarks of their respective owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input stat
mysql> grant replication slave on *.* to asd@'%' identified by '123456';
Query OK, 0 rows affected (0.00 sec)

mysql>
mysql> show master status;
+-----+-----+-----+-----+
| File | Position | Binlog_Do_DB | Binlog_Ignore_DB |
+-----+-----+-----+-----+
| mysql-bin.000003 | 245 |          |          |
+-----+-----+-----+-----+
1 row in set (0.00 sec)

mysql> 

```



```

1 从服务器1
Type 'help;' or '\h' for help. Type '\c' to cl
mysql> change master to
-> master_user='asd',
-> master_password='123456',
-> master_host='192.168.64.23',
-> master_log_file='mysql-bin.000003',
-> master_log_pos=245;
Query OK, 0 rows affected (0.07 sec)

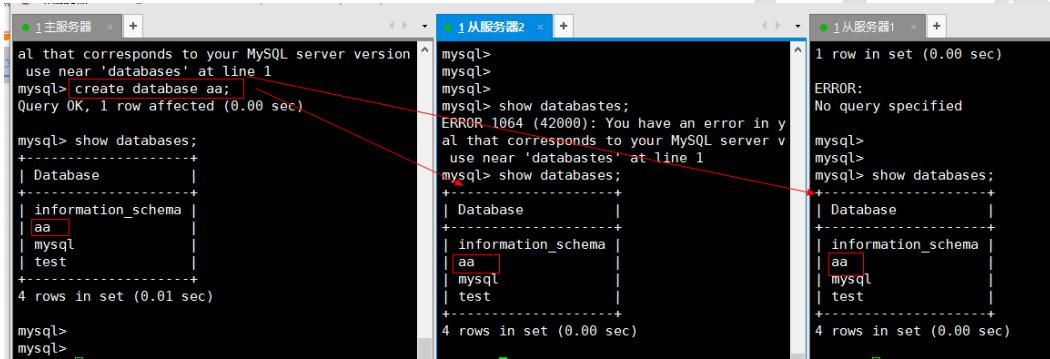
mysql>
mysql> start slave;
Query OK, 0 rows affected (0.00 sec)

mysql> show slave status;
+-----+-----+-----+-----+
|           |           |           |           |
+-----+-----+-----+-----+

```

#在另一台从服务器上做相同的配置（注意 id 不能相同）

测试



```

1 主服务器
al that corresponds to your MySQL server version
use near 'databases' at line 1
mysql> create database aa;
Query OK, 1 row affected (0.00 sec)

mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| aa          |
| mysql       |
| test        |
+-----+
4 rows in set (0.01 sec)

mysql>
mysql>

1 从服务器2
mysql>
mysql>
mysql> show databases;
ERROR: 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'show databases' at line 1
mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| aa          |
| mysql       |
| test        |
+-----+
4 rows in set (0.00 sec)

mysql>
mysql>

1 从服务器1
1 row in set (0.00 sec)

ERROR:
No query specified

mysql>
mysql>
mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| aa          |
| mysql       |
| test        |
+-----+
4 rows in set (0.00 sec)

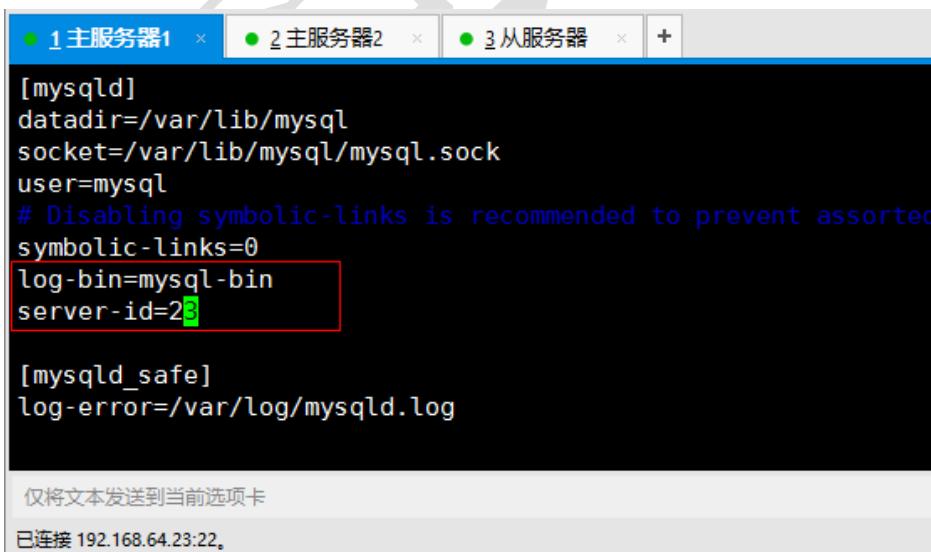
mysql>
mysql>

```

4. MySQL 多主一从

主服务器配置

开启二进制日志，启动服务



```

[mysqld]
datadir=/var/lib/mysql
socket=/var/lib/mysql/mysql.sock
user=mysql
# Disabling symbolic-links is recommended to prevent assorted
symbolic-links=0
log-bin=mysql-bin
server-id=2

[mysqld_safe]
log-error=/var/log/mysqld.log

仅将文本发送到当前选项卡
已连接 192.168.64.23:22,

```

授权

```
● 1 主服务器1 × ● 2 主服务器2 × ● 3 从服务器 × +  
mysql> grant replication slave on *.* to asd@'192.168.64.24' identified by '123456';  
Query OK, 0 rows affected (0.00 sec)  
  
mysql>
```

在主服务器 2 上做相同的操作

```
● 1 主服务器1 × ● 2 主服务器2 × ● 3 从服务器 × +  
  
[mysqld]  
datadir=/var/lib/mysql  
socket=/var/lib/mysql/mysql.sock  
user=mysql  
# Disabling symbolic-links is recommended to prevent assorted security risks.  
symbolic-links=0  
log-bin=mysql-bin  
server-id=25  
  
[mysqld_safe]  
log-error=/var/log/mysqld.log
```

授权

```
● 1 主服务器1 × ● 2 主服务器2 × ● 3 从服务器 × +  
  
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> grant replication slave on *.* to asd@'192.168.64.24' identified by '123456';  
Query OK, 0 rows affected (0.00 sec)
```

从服务器操作
对主配置文件操作

```
[mysqld]
datadir=/var/lib/mysql
socket=/var/lib/mysql/mysql.sock
user=mysql
# Disabling symbolic-links is recommended to prevent assorted security risks
symbolic-links=0

[mysqld_safe]
log-error=/var/log/mysqld.log
pid-file=/var/run/mysqld/mysqld.pid

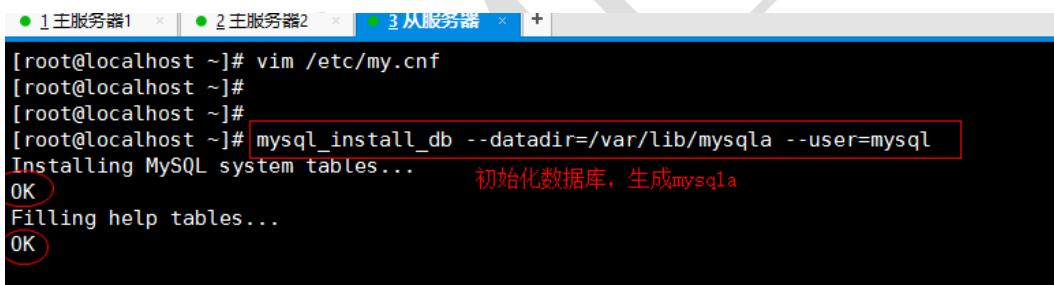
#M-M-S

[mysqld_multi]
mysqld=/usr/bin/mysqld_safe
mysqladmin=/usr/bin/mysqladmin
log=/tmp/multi.log

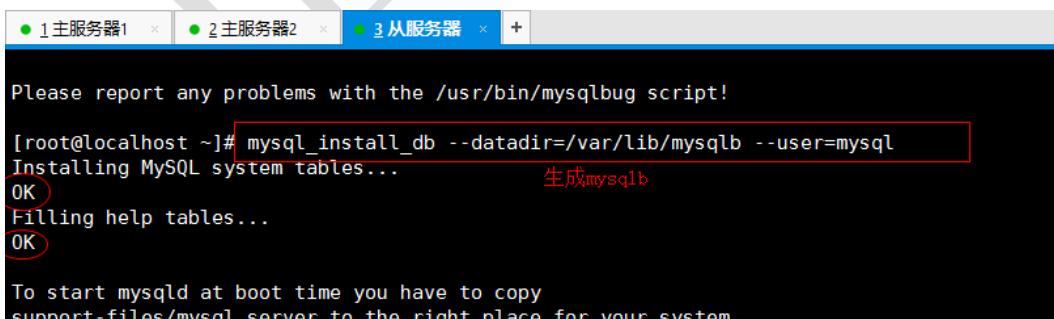
[mysqld10]
port=3306
datadir=/var/lib/mysqla/
pid-file=/var/lib/mysqla/mysqld.pid
socket=/var/lib/mysqla/mysql.sock
user=mysql
server-id=30

[mysqld20]
port=3307
datadir=/var/lib/mysqlb/
pid-file=/var/lib/mysqlb/mysqld.pid
socket=/var/lib/mysqlb/mysql.sock
user=mysql
server-id=30
```

初始化数据库，生成目录 mysqla, mysqlb



```
[root@localhost ~]# vim /etc/my.cnf
[root@localhost ~]#
[root@localhost ~]#
[root@localhost ~]# mysql_install_db --datadir=/var/lib/mysqla --user=mysql
Installing MySQL system tables...      初始化数据库，生成mysqla
OK
Filling help tables...
OK
```



```
Please report any problems with the /usr/bin/mysqlbug script!

[root@localhost ~]# mysql_install_db --datadir=/var/lib/mysqlb --user=mysql
Installing MySQL system tables...      生成mysqlb
OK
Filling help tables...
OK

To start mysqld at boot time you have to copy
support-files/mysql.server to the right place for your system.
```

```

● 1 主服务器1 × ● 2 主服务器2 × ● 3 从服务器 × + | 
[root@localhost ~]#
[root@localhost ~]#
[root@localhost ~]# cd /var/lib/
[root@localhost lib]# ls
alternatives certmonger fprintf logrotate.status mysql nfs polkit-1 random-seed
authconfig dbus games misc mysqla ntp postfix readahead
cas dhclient ipa-client mlocate mysqlb plymouth prelink rhsm
[root@localhost lib]# ls -lh mysqla/
总用量 8.0K
drwx----- 2 mysql root 4.0K 2月 25 07:26 mysql
drwx----- 2 mysql root 4.0K 2月 25 07:26 test
[root@localhost lib]# ls -lh mysqlb/
总用量 8.0K
drwx----- 2 mysql root 4.0K 2月 25 07:31 mysql
drwx----- 2 mysql root 4.0K 2月 25 07:31 test
[root@localhost lib]#

```

设置 mysqla, mysqlb 目录及以下文件的属主为 mysql (防止出现权限问题)

```

[root@localhost lib]# chown -R mysql /var/lib/mysqla/
[root@localhost lib]# chown -R mysql /var/lib/mysqlb/
[root@localhost lib]#
[root@localhost lib]#

```

启动从服务器线程

```

[root@localhost ~]# mysqld_multi --defaults-file=/etc/my.cnf start 23
[root@localhost ~]# netstat -anpt
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State      PID/Program name
tcp        0      0 0.0.0.0:111             0.0.0.0:*              LISTEN     1160/rpcbind
tcp        0      0 0.0.0.0:57680            0.0.0.0:*              LISTEN     1178/rpc.statd
tcp        0      0 0.0.0.0:22              0.0.0.0:*              LISTEN     1382/sshd
tcp        0      0 127.0.0.1:631            0.0.0.0:*              LISTEN     1259/cupsd
tcp        0      0 127.0.0.1:25              0.0.0.0:*              LISTEN     1458/master
tcp        0      0 0.0.0.0:3306            0.0.0.0:*              LISTEN     2107/mysql
tcp        0      0 192.168.64.24:22          192.168.64.1:58561    ESTABLISHED 1717/sshd
tcp        0      0 192.168.64.24:22          192.168.64.1:50013    ESTABLISHED 1646/sshd
tcp        0      0 ::53903                ::*                   LISTEN     1178/rpc.statd
tcp        0      0 ::1:111                 ::*                   LISTEN     1160/rpcbind
tcp        0      0 ::1:22                  ::*                   LISTEN     1382/sshd
tcp        0      0 ::1:631                 ::*                   LISTEN     1259/cupsd
tcp        0      0 ::1:25                  ::*                   LISTEN     1458/master

```

```

[root@localhost ~]# mysqld_multi --defaults-file=/etc/my.cnf start 25
[root@localhost ~]# netstat -anpt
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State      PID/Program name
tcp        0      0 0.0.0.0:3307            0.0.0.0:*              LISTEN     5245/mysql
tcp        0      0 0.0.0.0:111             0.0.0.0:*              LISTEN     1160/rpcbind
tcp        0      0 0.0.0.0:57680            0.0.0.0:*              LISTEN     1178/rpc.statd
tcp        0      0 0.0.0.0:22              0.0.0.0:*              LISTEN     1382/sshd
tcp        0      0 127.0.0.1:631            0.0.0.0:*              LISTEN     1259/cupsd
tcp        0      0 127.0.0.1:25              0.0.0.0:*              LISTEN     1458/master
tcp        0      0 0.0.0.0:3306            0.0.0.0:*              LISTEN     2107/mysql
tcp        0    264 192.168.64.24:22          192.168.64.1:58561    ESTABLISHED 1717/sshd
tcp        0      0 192.168.64.24:22          192.168.64.1:50013    ESTABLISHED 1646/sshd
tcp        0      0 ::53903                ::*                   LISTEN     1178/rpc.statd
tcp        0      0 ::1:111                 ::*                   LISTEN     1160/rpcbind
tcp        0      0 ::1:22                  ::*                   LISTEN     1382/sshd
tcp        0      0 ::1:631                 ::*                   LISTEN     1259/cupsd
tcp        0      0 ::1:25                  ::*                   LISTEN     1458/master

```

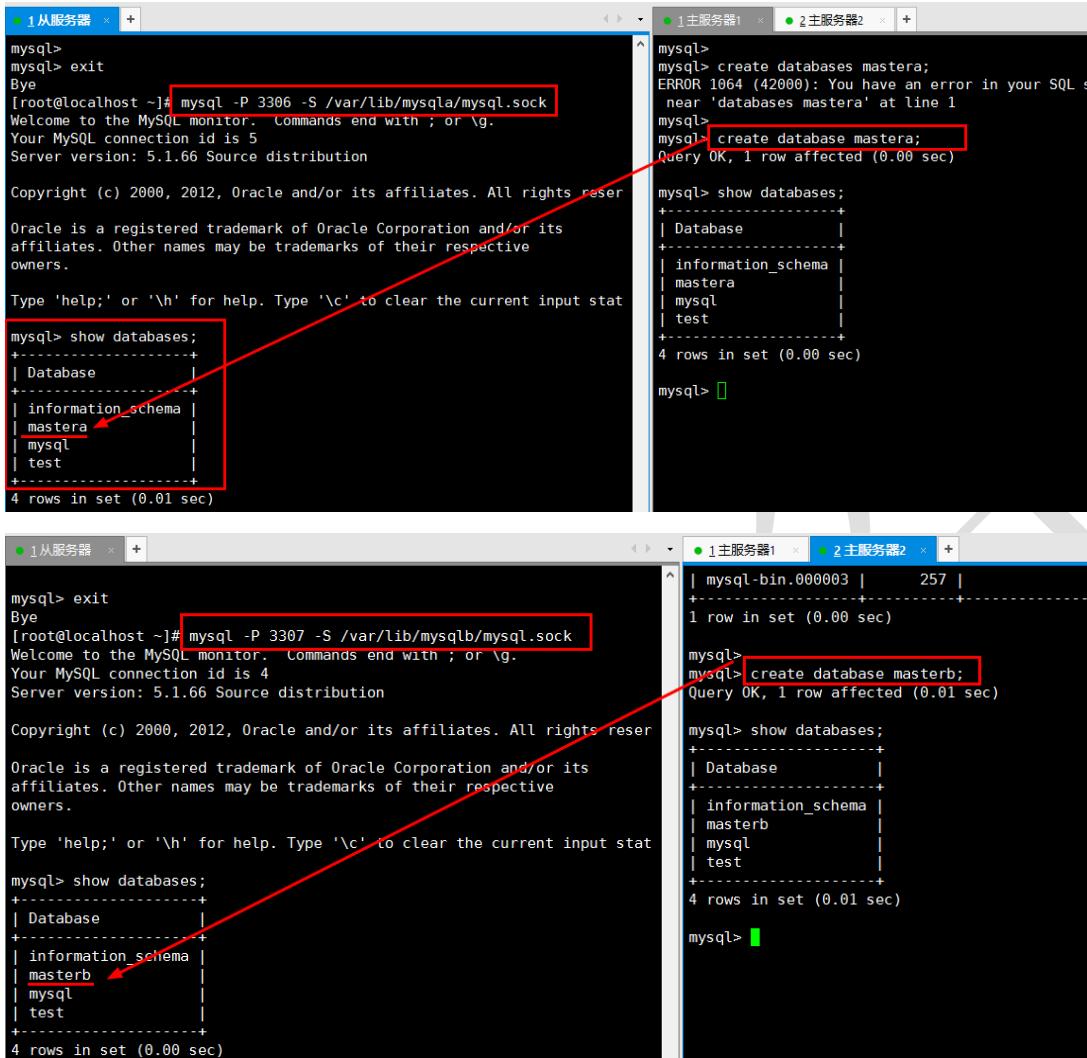
登录并保存授权信息



```
● 1 主服务器1 × ● 2 主服务器2 × ● 3 从服务器 × +  
[root@localhost ~]# mysql -P 3306 -S /var/lib/mysql/mysql.sock  
Welcome to the MySQL monitor. Commands end with ; or \g.  
Your MySQL connection id is 1 登录主服务器1  
Server version: 5.1.66 Source distribution  
  
Copyright (c) 2000, 2012, 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> change master to  
-> master_user='asd',  
-> master_password='123456',  
-> master_host='192.168.64.23',  
-> master_log_file='mysql-bin.000003',  
-> master_log_pos=257;  
Query OK, 0 rows affected (0.09 sec)  
  
mysql> start slave;  
Query OK, 0 rows affected (0.00 sec)  
  
mysql> show slave status\G;  
***** 1. row *****  
Slave_IO_State: Waiting for master to send event  
Master_Host: 192.168.64.23  
Master_User: asd  
Master_Port: 3306  
Connect_Retry: 60
```

```
● 1 主服务器1 × ● 2 主服务器2 × ● 3 从服务器 × +  
  
mysql>  
mysql>  
mysql> exit  
Bye  
[root@localhost ~]# mysql -P 3307 -S /var/lib/mysqlb/mysql.sock  
Welcome to the MySQL monitor. Commands end with ; or \g.  
Your MySQL connection id is 1 登录主服务器2, 同步2上的数据  
Server version: 5.1.66 Source distribution  
  
Copyright (c) 2000, 2012, 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> change master to  
-> master_user='asd',  
-> master_password='123456',  
-> master_host='192.168.64.25',  
-> master_log_file='mysql-bin.000003',  
-> master_log_pos=257;  
Query OK, 0 rows affected (0.10 sec)  
  
mysql> start slave;  
Query OK, 0 rows affected (0.00 sec)  
  
mysql> show slave status;
```

测试



The screenshot shows two MySQL sessions side-by-side:

- Server 1 (从服务器):** Shows the creation of a database named 'mastera' on the slave server. It also lists existing databases: information_schema, mastera, mysql, and test.
- Server 2 (主服务器1):** Shows the creation of a database named 'mastera' on the master server. It lists existing databases: information_schema, mastera, mysql, and test.
- Server 1 (从服务器):** Shows the creation of a database named 'masterb' on the slave server. It lists existing databases: information_schema, masterb, mysql, and test.
- Server 2 (主服务器1):** Shows the creation of a database named 'masterb' on the master server. It lists existing databases: information_schema, masterb, mysql, and test.

5. MySQL 中间件-Amoeba

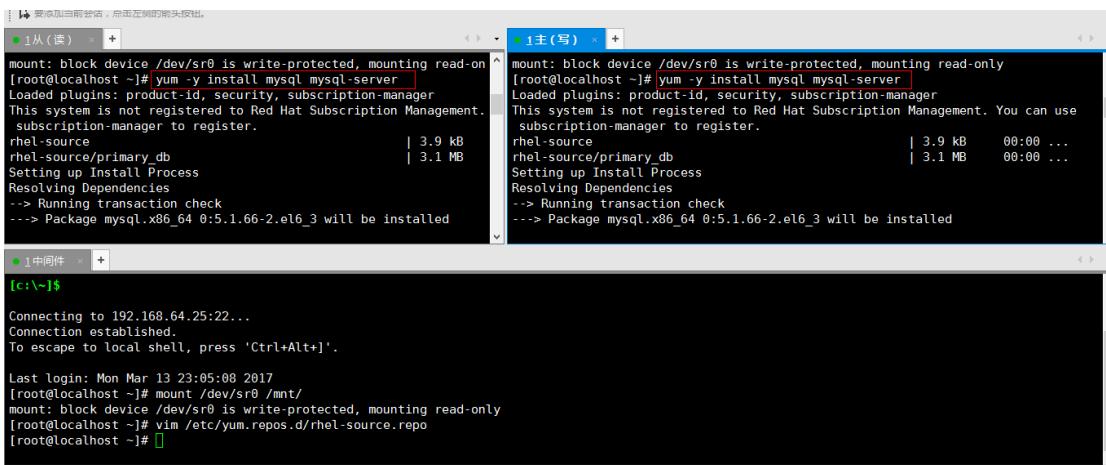
中间件：一种提供在不同技术、不同的软件之间共享资源的程序，更大化了利用了数据库的性能，可以无限扩展（注：真实环境中并非如此）

数据库的中间件：

- mysql proxy (官方版本) 性能低，需要 lua 脚本
- atlas 性能低，响应时间长
- amoeba 陈思儒研发的

一. 先搭建一个主从关系的服务器

在主、从服务器上安装 mysql mysql-server



```

mount: block device /dev/sr0 is write-protected, mounting read-only
[root@localhost ~]# yum -y install mysql mysql-server
Loaded plugins: product-id, security, subscription-manager
This system is not registered to Red Hat Subscription Management.
  subscription-manager to register.
rhel-source                                | 3.9 kB   00:00 ...
rhel-source/primary_db                      | 3.1 MB    00:00 ...
Setting up Install Process
Resolving Dependencies
--> Running transaction check
--> Package mysql.x86_64 0:5.1.66-2.el6_3 will be installed

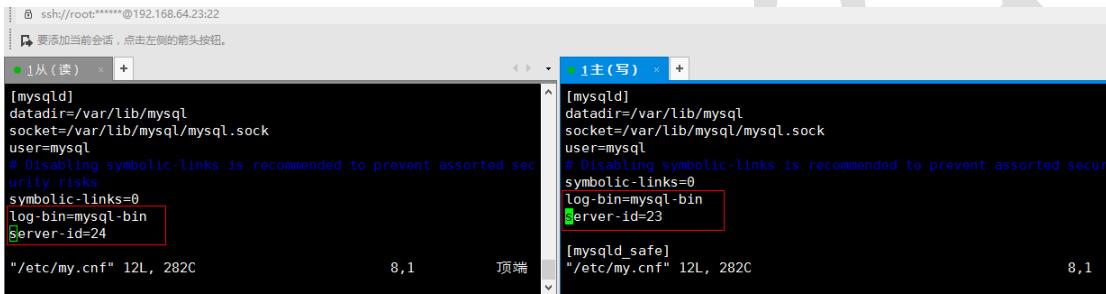
[1主(写)] mount: block device /dev/sr0 is write-protected, mounting read-only
[root@localhost ~]# yum -y install mysql mysql-server
Loaded plugins: product-id, security, subscription-manager
This system is not registered to Red Hat Subscription Management. You can use
subscription-manager to register.
rhel-source                                | 3.9 kB   00:00 ...
rhel-source/primary_db                      | 3.1 MB    00:00 ...
Setting up Install Process
Resolving Dependencies
--> Running transaction check
--> Package mysql.x86_64 0:5.1.66-2.el6_3 will be installed

[1从(读)] [c:\v-]$ Connecting to 192.168.64.25:22...
Connection established.
To escape to local shell, press 'Ctrl+Alt+]'.

Last login: Mon Mar 13 23:05:08 2017
[root@localhost ~]# mount /dev/sr0 /mnt/
mount: block device /dev/sr0 is write-protected, mounting read-only
[root@localhost ~]# vim /etc/yum.repos.d/rhel-source.repo
[root@localhost ~]#

```

1. 开启二进制日志



```

[mysqld]
datadir=/var/lib/mysql
socket=/var/lib/mysql/mysql.sock
user=mysql
# Disabling symbolic-links is recommended to prevent assorted security risks
symbolic-links=0
log-bin=mysql-bin
server-id=24

"/etc/my.cnf" 12L, 282C

```

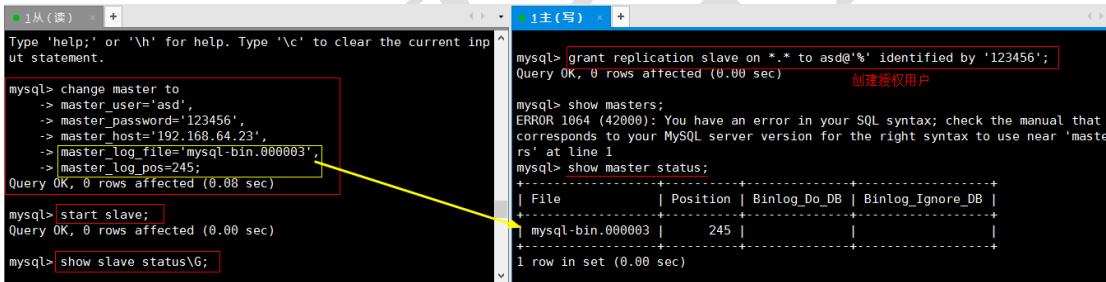
```

[mysqld]
datadir=/var/lib/mysql
socket=/var/lib/mysql/mysql.sock
user=mysql
# Disabling symbolic-links is recommended to prevent assorted security risks
symbolic-links=0
log-bin=mysql-bin
server-id=23

[mysqld_safe]
"/etc/my.cnf" 12L, 282C

```

2. 在主服务器上授权，从服务器上保存授权信息，并开启从服务线程。



```

mysql> change master to
      > master_user='asd',
      > master_password='123456',
      > master_host='192.168.64.23',
      > master_log_file='mysql-bin.000003',
      > master_log_pos=245;
Query OK, 0 rows affected (0.08 sec)

mysql> start slave;
Query OK, 0 rows affected (0.00 sec)

mysql> show slave status\G;

```

```

mysql> grant replication slave on *.* to asd@'%' identified by '123456';
Query OK, 0 rows affected (0.00 sec) 创建授权用户

mysql> show masters;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that
corresponds to your MySQL server version for the right syntax to use near 'maste
rs' at line 1
mysql> show master status;
+-----+-----+-----+
| File | Position | Binlog_Do_DB | Binlog_Ignore_DB |
+-----+-----+-----+
| mysql-bin.000003 | 245 |          |          |
+-----+-----+-----+
1 row in set (0.00 sec)

```

3. 关闭从服务器线程，为了做读写分离时，测试有明显的实验效果（实际生产环境中不能停掉。。。）

```
● 1从(读) × +  
mysql> stop slave;  
Query OK, 0 rows affected (0.00 sec)  
  
mysql> show slave status\G;  
***** 1. row *****  
Slave_IO_State:  
    Master_Host: 192.168.64.23  
    Master_User: asd  
    Master_Port: 3306  
    Connect_Retry: 60  
    Master_Log_File: mysql-bin.000003  
    Read_Master_Log_Pos: 324  
    Relay_Log_File: mysqld-relay-bin.000002  
    Relay_Log_Pos: 330  
    Relay_Master_Log_File: mysql-bin.000003  
    Slave_IO_Running: No  
    Slave_SQL_Running: No  
    Replicate_Do_DB:
```

二. 配置读写分离

1. 安装 gcc 环境 (amoeba 需要源码安装)

```
[root@localhost ~]# [root@localhost ~]# yum -y install gcc*  
Loaded plugins: product-id, security, subscription-manager  
This system is not registered to Red Hat Subscription Management. You can use subscription-manager to register.  
rhel-source  
rhel-source/primary_db  
Setting up Install Process  
Resolving Dependencies  
--> Running transaction check  
--> Package gcc.x86_64 0:4.4.7-3.el6 will be installed  
--> Processing Dependency: cpp = 4.4.7-3.el6 for package: gcc-4.4.7-3.el6.x86_64
```

2. 拷贝第三方软件, 创建单独的目录

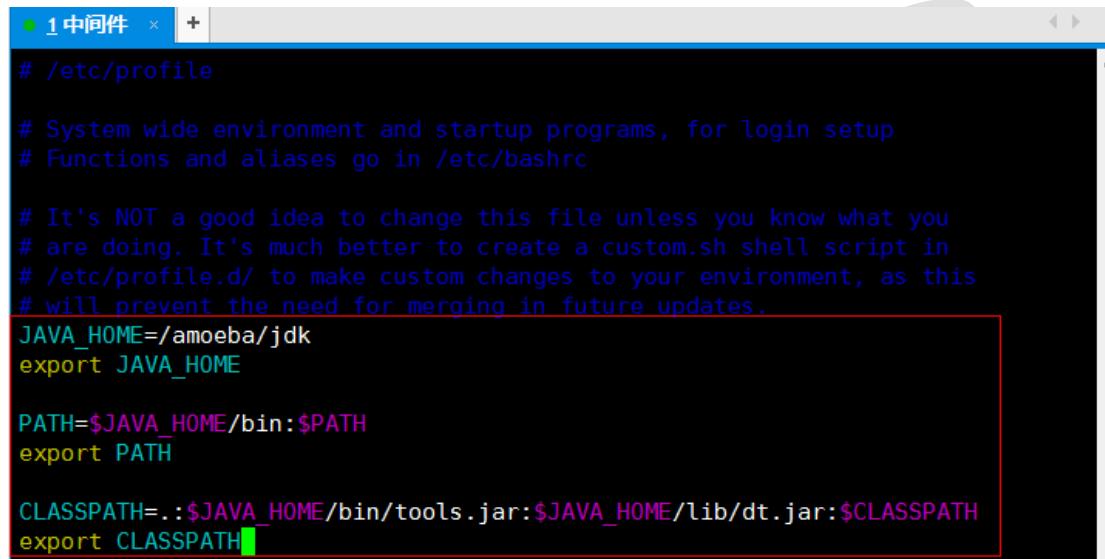
```
[root@localhost ~]# mount /dev/sr0 /mnt/  
mount: block device /dev/sr0 is write-protected, mounting read-only  
[root@localhost ~]# cd /mnt/  
[root@localhost mnt]# ls  
amoeba-mysql-1.3.1-BETA.zip jdk-7u40-linux-x64.gz  
[root@localhost mnt]# cp * /usr/src/  
[root@localhost mnt]# cd !$  
cd /usr/src/  
[root@localhost src]# ls  
amoeba-mysql-1.3.1-BETA.zip debug jdk-7u40-linux-x64.gz kernels
```

```
[root@localhost src]# cd  
[root@localhost ~]# mkdir /amoeba  
[root@localhost ~]# 创建单独的目录, 用来存储安装  
amoeba相关的东西  
[root@localhost ~]#  
[root@localhost ~]#
```

3. 先安装 jdk (amoeba 是由 java 语言编写的, 所以先安装 jdk), 配置 java 环境

```
[root@localhost src]# tar -xf jdk-7u40-linux-x64.gz -C /amoeba/  
[root@localhost src]# 解压到指定的目录下  
[root@localhost src]#  
[root@localhost src]# cd /amoeba/  
[root@localhost amoeba]# ls  
jdk1.7.0_40  
[root@localhost amoeba]# ln -s jdk1.7.0_40/ jdk  
[root@localhost amoeba]# ls 创个软链接去除版本号，方便操作  
jdk jdk1.7.0_40  
[root@localhost amoeba]#
```

4. 声明用 java 写出来的程序如何调用 (/etc/profile)

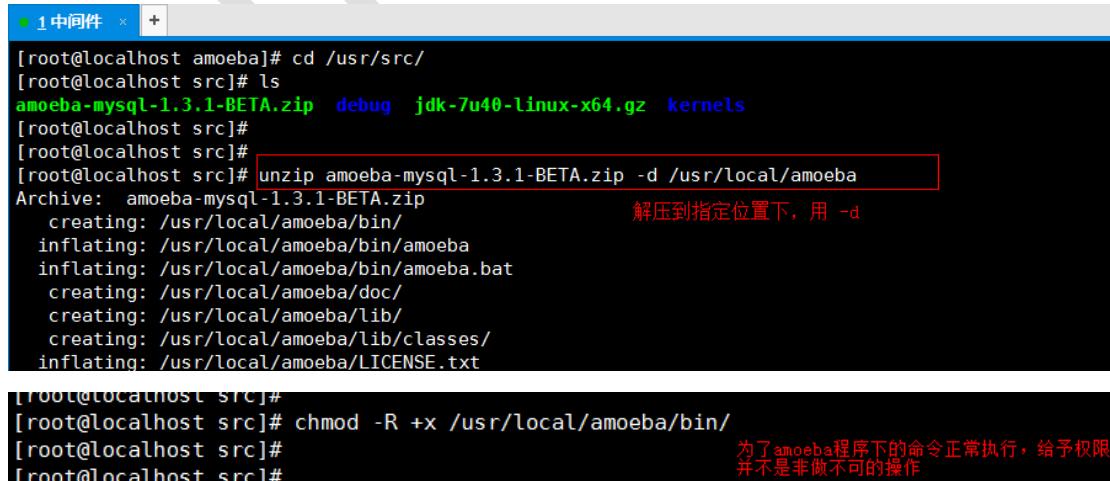


```
# /etc/profile  
  
# System wide environment and startup programs, for login setup  
# Functions and aliases go in /etc/bashrc  
  
# It's NOT a good idea to change this file unless you know what you  
# are doing. It's much better to create a custom.sh shell script in  
# /etc/profile.d/ to make custom changes to your environment, as this  
# will prevent the need for merging in future updates.  
JAVA_HOME=/amoeba/jdk  
export JAVA_HOME  
  
PATH=$JAVA_HOME/bin:$PATH  
export PATH  
  
CLASSPATH=.:$JAVA_HOME/bin/tools.jar:$JAVA_HOME/lib/dt.jar:$CLASSPATH  
export CLASSPATH
```

```
[root@localhost amoeba]#  
[root@localhost amoeba]# source /etc/profile 加载一下，使之生效  
[root@localhost amoeba]#  
[root@localhost amoeba]#
```

```
[root@localhost amoeba]#  
[root@localhost amoeba]# java -version 测试一下  
java version "1.7.0_40"  
Java(TM) SE Runtime Environment (build 1.7.0_40-b43)  
Java HotSpot(TM) 64-Bit Server VM (build 24.0-b56, mixed mode)  
[root@localhost amoeba]#
```

5. 安装 amoeba



```
[root@localhost amoeba]# cd /usr/src/  
[root@localhost src]# ls  
amoeba-mysql-1.3.1-BETA.zip debug jdk-7u40-linux-x64.gz kernels  
[root@localhost src]#  
[root@localhost src]# unzip amoeba-mysql-1.3.1-BETA.zip -d /usr/local/amoeba  
Archive: amoeba-mysql-1.3.1-BETA.zip  
  creating: /usr/local/amoeba/bin/ 解压到指定位置下，用 -d  
  inflating: /usr/local/amoeba/bin/amoeba  
  inflating: /usr/local/amoeba/bin/amoeba.bat  
  creating: /usr/local/amoeba/doc/  
  creating: /usr/local/amoeba/lib/  
  creating: /usr/local/amoeba/lib/classes/  
  inflating: /usr/local/amoeba/LICENSE.txt  
  
[root@localhost src]#  
[root@localhost src]# chmod -R +x /usr/local/amoeba/bin/ 为了amoeba程序下的命令正常执行，给予权限  
[root@localhost src]#  
[root@localhost src]#
```

配置 amoeba 这个软件

```
[root@localhost src]# cd /usr/local/amoeba/
[root@localhost amoeba]# ls
bin build.properties build.xml conf doc lib LICENSE.txt README.html src
[root@localhost amoeba]# cd conf/
[root@localhost conf]# ls
access_list.conf amoeba.dtd amoeba.xml function.dtd functionMap.xml log4j.dtd log4j.xml rule.dtd ruleFunctionMap.xml rule.xml
[root@localhost conf]# 主配置文件
[root@localhost conf]#
```

<server>.....</server>区域

```
7      <!-- proxy server 端口 -->
8      <property name="port">8066</property> 端口
9  server区域的配置
10     <!-- proxy server 的 IP -->
11
12     <property name="ipAddress">192.168.64.25</property> amoeba安装的 ip地址
13
14     <!-- proxy server net IO Read thread size -->
15     <property name="readThreadPoolSize">20</property>
16
17     <!-- proxy server client process thread size --> 使用默认值
18     <property name="clientSideThreadPoolSize">30</property>
19
20     <!-- mysql server data packet process thread size -->
21     <property name="serverSideThreadPoolSize">30</property>
22
23     <!-- socket Send and receive BufferSize(unit:K) -->
24     <property name="netBufferSize">128</property>
25
26     <!-- Enable/disable TCP_NODELAY (disable/enable Nagle's algorithm). -->
27     <property name="tcpNoDelay">true</property>
28
29     <!-- 登录amoeba的用户名，本来是root(和管理员用户没关系) -->
30     <property name="user">asd</property> 登录amoeba的用户名，本来是root(和管理员用户没关系)
31
32     <!-- 登录用户使用的密码 -->
33
34     <property name="password">123456</property> 登录用户使用的密码
35
36
37     <!-- query timeout( default: 60 second , TimeUnit:second) -->
38     <property name="queryTimeout">60</property>
```

```
<connectionManagerList>
连接时管理组件的值
<actionManager name="defaultManager" class="com.meidusa.amoeba.net.MultiConnectionManagerWrapper">
<property name="subManagerClassName">com.meidusa.amoeba.net.AuthingableConnectionManager</property>

<!--
    default value is available Processors
    <property name="processors">5</property>
-->
</connectionManager>
```

<dbServerList>.....</dbServerList>区域

```

1从(读) × 2中间件 × 3主(写) × +
63 <!-- PoolableObjectFactory配置 -->
64 <dbServer name="server1"> 服务器名字
65   <factoryConfig class="com.meidusa.amoeba.mysql.net.MysqlServerConnectionFactory">
66     <property name="manager">defaultManager</property>
67
68     <!-- 数据库连接配置 -->
69     <property name="port">3306</property>
70
71     <!-- 登录的IP地址 -->
72     <property name="ipAddress">192.168.64.23</property> 登录的是哪个数据库服务器（这里是主）
73     <property name="schema">test</property>
74
75     <!-- 登录用户名 -->
76     <property name="user">asd</property> 使用的用户名是什么（这里是在主服务器上授权的那个用户）
77
78     <property name="password">123456</property> 使用的密码
79
80
81   </factoryConfig>
82
83   <!-- ObjectPool配置 -->
84   <poolConfig class="com.meidusa.amoeba.net.poolable.PoolableObjectPool">
85     <property name="maxActive">200</property>
86     <property name="maxIdle">200</property>
87     <property name="minIdle">10</property>
88     <property name="minEvictableIdleTimeMillis">600000</property>
89     <property name="timeBetweenEvictionRunsMillis">600000</property>
90     <property name="testOnBorrow">true</property>
91     <property name="testWhileIdle">true</property>
92
93   </poolConfig>
94
95 </dbServer> 和优化相关的，可以不做操作
96
97
98
99
100 <dbServer name="master" virtual="true">
101   <poolConfig class="com.meidusa.amoeba.server.MultipleServerPool">
102     <!-- 池的名字，主的，提供写 -->
103     <!-- 1=ROUNDROBIN, 2=WEIGHTBASED, 3=HA -->
104     <property name="loadbalance">1</property>
105
106     <!-- 第二台服务器的pool，第二台的ip和库名 -->
107     <property name="poolNames">server1</property>
108
109   </poolConfig>
110 </dbServer>

```

由于只提供了一个服务器模板，需要自己复制另一个填写关于读的

```

<dbServer name="server2"> 服务器2
  <!-- PoolableObjectFactory配置 -->
  <factoryConfig class="com.meidusa.amoeba.mysql.net.MysqlServerConnectionFactory">
    <property name="manager">defaultManager</property>

    <!-- 数据库连接配置 -->
    <property name="port">3306</property> 端口不变
    <!-- 登录的IP地址 -->
    <property name="ipAddress">192.168.64.24</property> 第二台服务器的ip即从服务的
    <property name="schema">test</property> 代表同步的库
    <!-- 登录用户名 -->
    <property name="user">asd</property> 在从服务器上授权的那个用户
    <property name="password">123456</property>

  </factoryConfig>

  <!-- ObjectPool配置 -->
  <poolConfig class="com.meidusa.amoeba.net.poolable.PoolableObjectPool">
    <property name="maxActive">200</property>
    <property name="maxIdle">200</property>
    <property name="minIdle">10</property>
    <property name="minEvictableIdleTimeMillis">600000</property>
    <property name="timeBetweenEvictionRunsMillis">600000</property>
    <property name="testOnBorrow">true</property>
    <property name="testWhileIdle">true</property>
  </poolConfig>
</dbServer>

```

```

<dbServer name="slave" virtual="true">
    <poolConfig class="com.meidusa.amoeba.server.MultipleServerPool">
        <!-- , 0 1 2 3-->
        <property name="loadbalance">1</property>
        <!-- ioe, pool, 0 1 2 3-->
        <property name="poolNames">server1,server2</property>
    </poolConfig>
</dbServer>          读从server1 server2

```



```

181      <property name="LRUMapSize">1500</property>
182      <property name="defaultPool">master</property> 默认访问池
183
184
185      <property name="writePool">master</property> 写入池
186      <property name="readPool">slave</property> 读取池
187
188      <property name="needParse">true</property>
189  </queryRouter>
190 </amoeba:configuration>
-- 插入 --

```

启动 amoeba，修改一下启动脚本：/usr/local/amoeba/bin/amoeba

```

#!/bin/sh
# DEFAULT_OPTS="-server -Xms256m -Xmx256m -Xss256k"
# DEFAULT_OPTS="$DEFAULT_OPTS -XX:+HeapDumpOnOutOfMemoryError -XX:+AggressiveOpts -XX:+UseParallelGC -e=64m"
DEFAULT_OPTS="$DEFAULT_OPTS -Damoeba.home=\"$AMOEBA_HOME\""
DEFAULT_OPTS="$DEFAULT_OPTS -Dclassworlds.conf=\"$AMOEBA_HOME/bin/amoeba.classworlds\""

CMD="exec \"$JAVA_HOME/bin/java\" $DEFAULT_OPTS $OPTS -classpath \"$CLASSPATH\" $MAIN_CLASS $@"
eval $CMD

```

注意：将-Xss128k 修改为 -Xss256

#在主和从服务器上进行指定用户授权，授权目的为了让 amoeba 能连接到主从服务器进行查询。

nohup bash -x /usr/local/amoeba/bin/amoeba &
把这个放到后台 退出终端也可以继续运行

```

[root@localhost bin]# nohup bash -x /usr/local/amoeba/bin/amoeba &
[1] 2635
[root@localhost bin]# nohup: 忽略输入并把输出追加到"nohup.out"

```

ps aux | grep amoeba

查看一下运行的程序 查看到的话就说明程序已经运行了起来

```

[root@localhost bin]# ps aux | grep amoeba
root      2635 17.4  5.5 1072680 56092 pts/0    Sl   00:05   0:03 /amoeba/jdk/bin/java -s
moeba -Dclassworlds.conf=/usr/local/amoeba/bin/amoeba.classworlds -classpath /usr/local/a
cher
root      2663  0.0  0.0 103240   872 pts/0     S+   00:06   0:00 grep amoeba
[root@localhost bin]#

```

测试（安装一个 MySQL 软件包才可以连接）

```
[root@localhost conf]# mysql -uasd -p -h 192.168.64.25 -P 8066
Enter password:                                     指定登录的地址, 服务的端口
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1290243769
Server version: 5.1.45-mysql-amoeba-proxy-1.3.1-BETA Source distribution

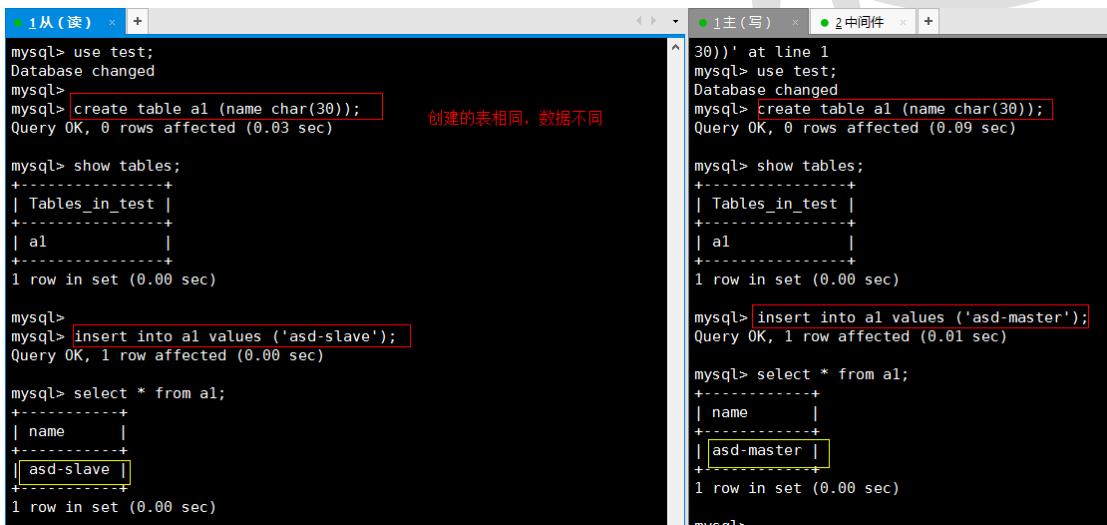
Copyright (c) 2000, 2012, Oracle and/or its affiliates. All rights reserved.

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>
```

在主、从服务器上创建表 a1，在主服务器的表中插入数据



The screenshot shows two MySQL command-line interfaces side-by-side. The left window (Master) has the title '1从(读)' and the right window (Slave) has the title '2中间件'. Both windows show the same sequence of commands:

```
mysql> use test;
Database changed
mysql> create table a1 (name char(30));
Query OK, 0 rows affected (0.03 sec)

mysql> show tables;
+-----+
| Tables_in_test |
+-----+
| a1           |
+-----+
1 row in set (0.00 sec)

mysql> insert into a1 values ('asd-slave');
Query OK, 1 row affected (0.00 sec)

mysql> select * from a1;
+-----+
| name   |
+-----+
| asd-slave |
+-----+
1 row in set (0.00 sec)
```

The right window (Slave) also shows the same commands and output, indicating successful replication.

之后在客户端登录测试

读取池的效果：



```
● 1 client  ×  + |  
mysql>  
mysql> use test;  
Database changed  
mysql> show tables;  
+-----+  
| Tables_in_test |  
+-----+  
| a1          |  
+-----+  
1 row in set (0.01 sec)  
  
mysql>  
mysql> select * from a1;  
+-----+  
| name      |  
+-----+  
| asd-master |  
+-----+  
1 row in set (0.02 sec)  
  
mysql> select * from a1;  
+-----+  
| name      |  
+-----+  
| asd-slave |  
+-----+  
1 row in set (0.03 sec)  
  
mysql> █
```

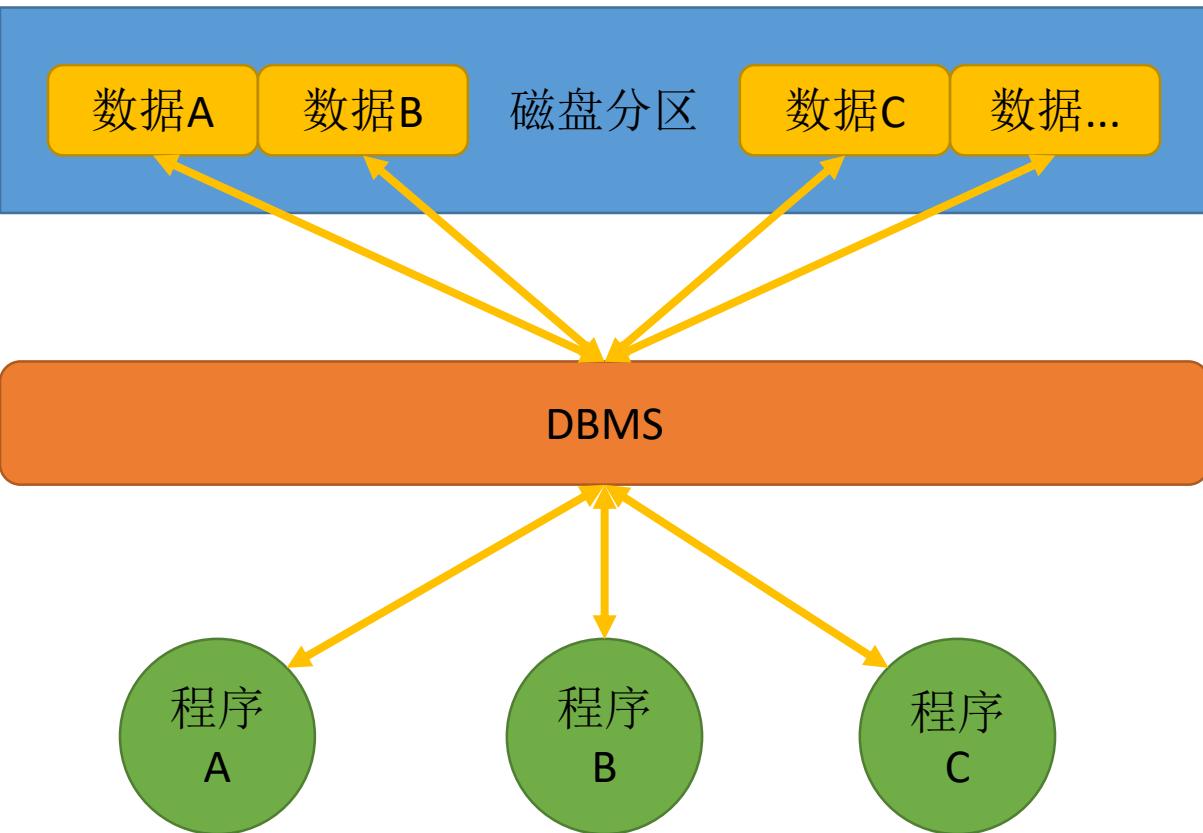
轮次出现，读是1 2 两台

写入池效果

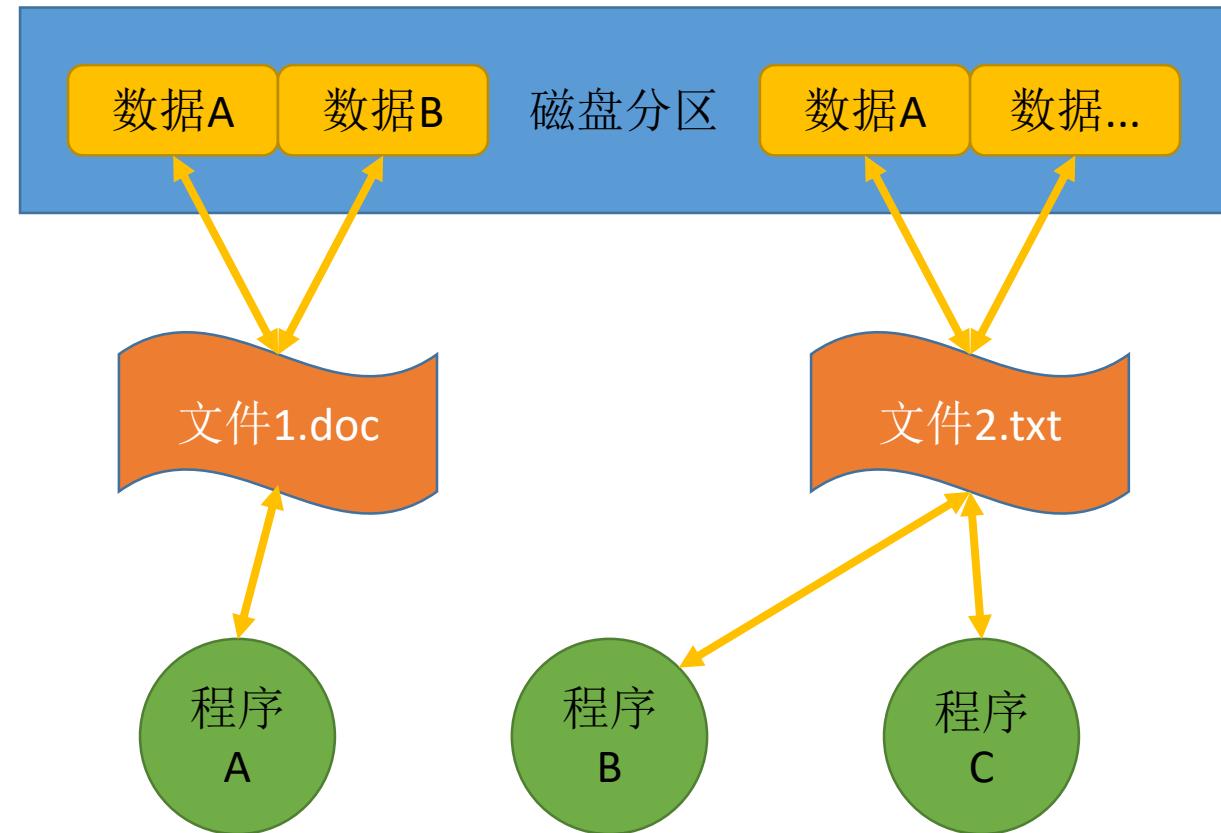
```
● 1 client × + |  
mysql>  
mysql> insert into a1 values ('asd-amoeba-01');  
Query OK, 1 row affected (0.04 sec)  
  
mysql> insert into a1 values ('asd-amoeba-02');  
Query OK, 1 row affected (0.03 sec)  
  
mysql> insert into a1 values ('asd-amoeba-03');  
Query OK, 1 row affected (0.02 sec) 模拟用户逛淘宝下单，是写入的操作。。  
  
mysql> select * from a1;  
+-----+  
| name |  
+-----+  
| asd-master |  
| asd-amoeba-01 |  
| asd-amoeba-02 |  
| asd-amoeba-03 |  
+-----+  
4 rows in set (0.01 sec)  
  
mysql>  
mysql> select * from a1;  
+-----+  
| name |  
+-----+  
| asd-slave |  
+-----+  
1 row in set (0.05 sec)
```

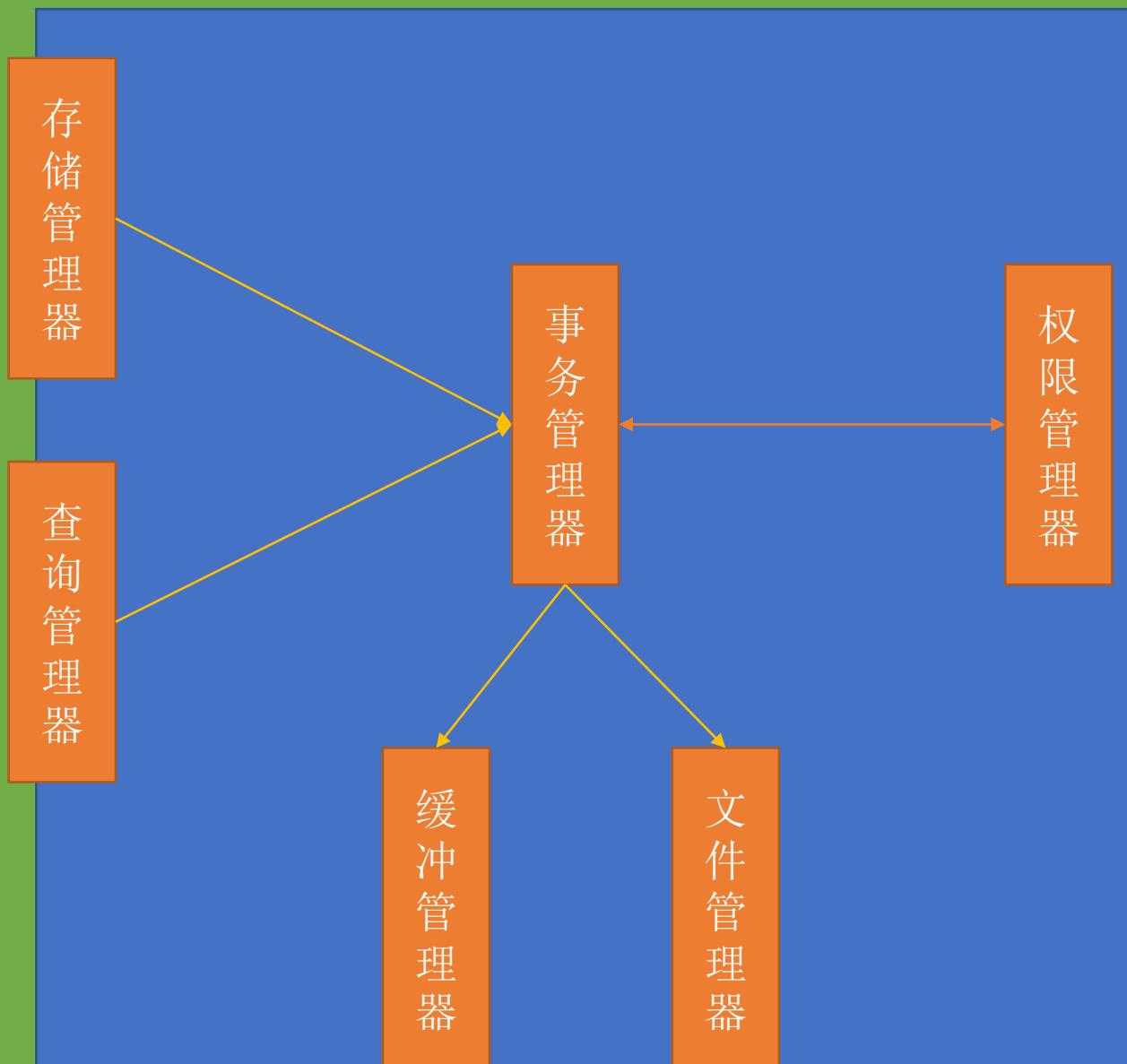
以上测试纯粹为了实验效果，在实际生产中，主从开启，主服务器上写入的数据也会同步到从服务器中

数据库系统数据调用

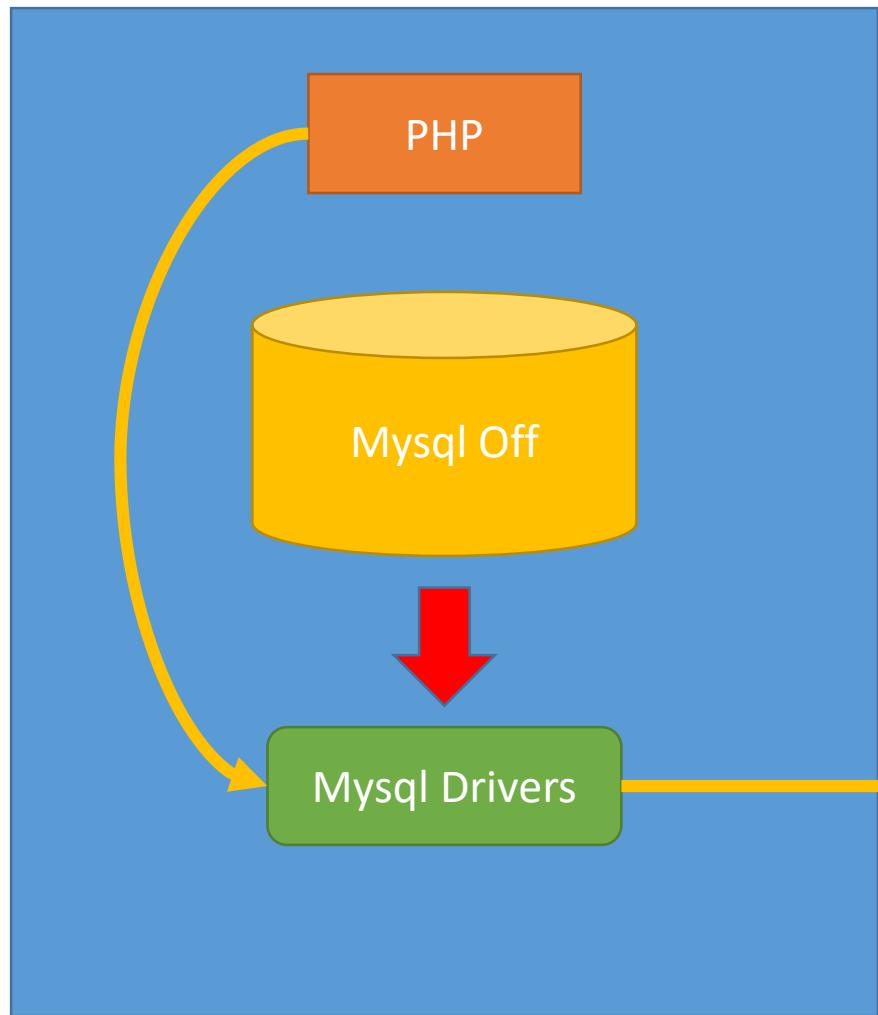


文件系统数据调用

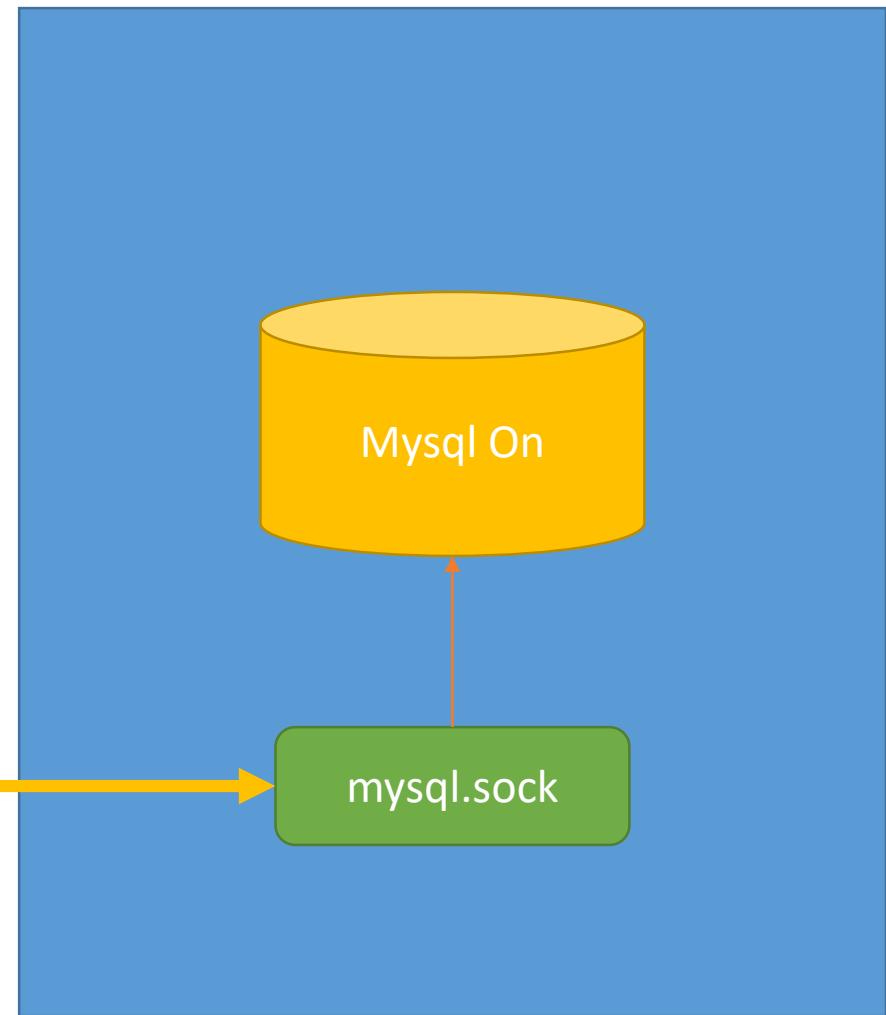


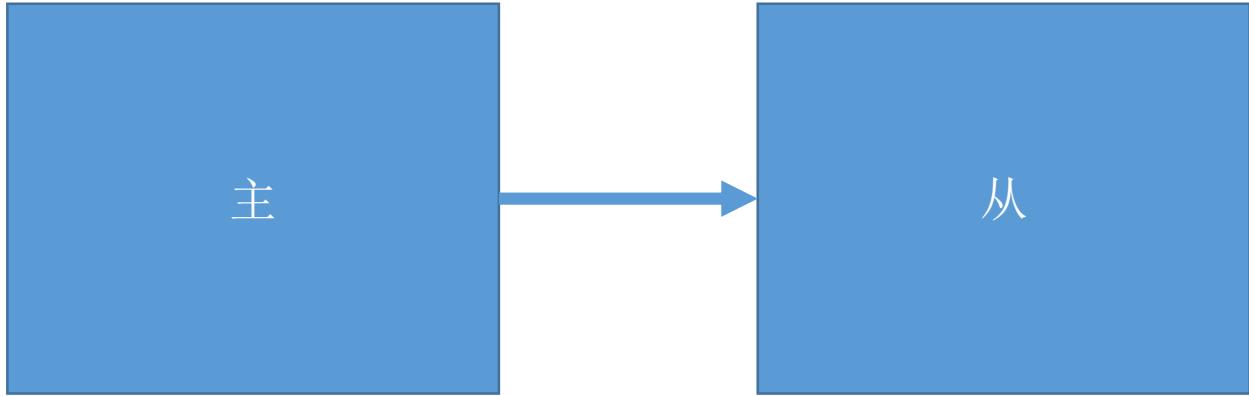


LAMP



MysqI





技术点：bin-log日志

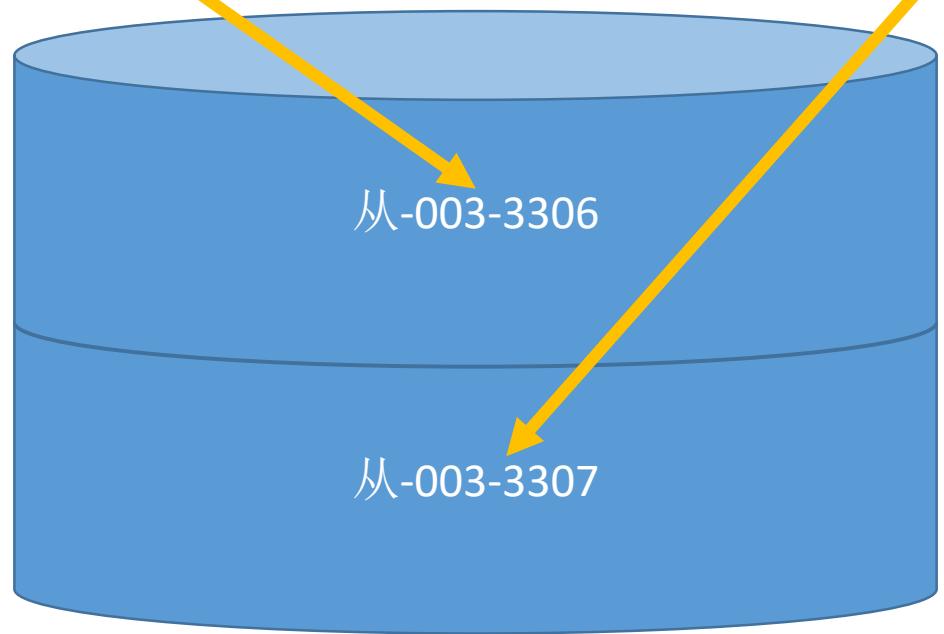
开启主服务器的bin-log日志记录功能，将主服务器的bin-log日志传到从服务器，从服务器根据日志内容将数据还原到本地。

主从服务器：

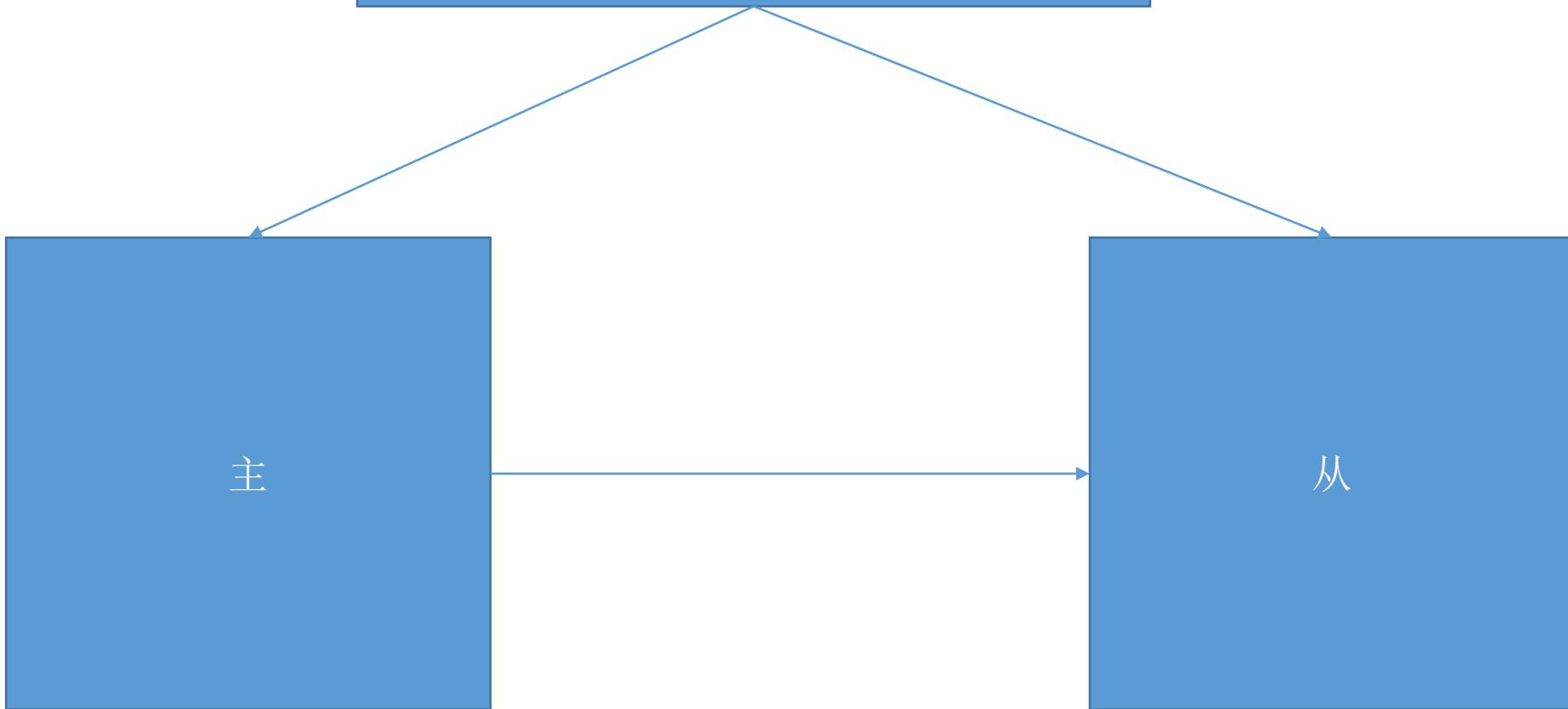
1. 从服务器主动把主服务器上的数据同步到本地（备份）
2. 从服务器分摊主服务器的查询压力（负载均衡）

主主服务器

1. 均摊写压力



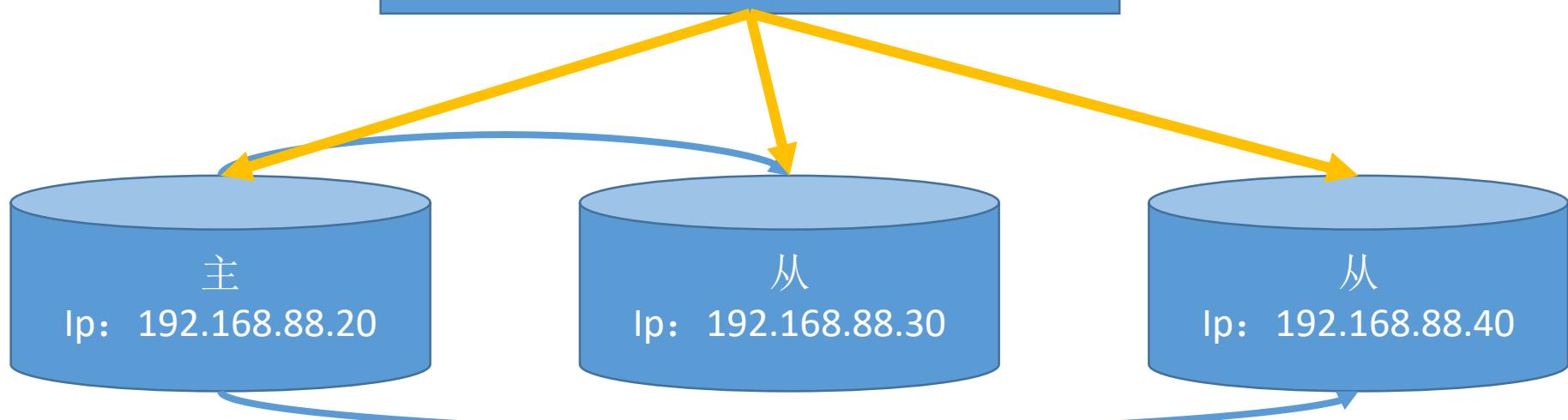
amoeba 123456
192.168.88.30:8066
中间键



Web server
连接数据库的函数

amoeba

```
<Server>
  IP:192.168.88.10
  Port:8066
  User:amoeba
  Password:123456
</Server>
<dbServerList>
  <dbServer name=serverN>
    </dbServer> X3
  <dbServer>
    <pool>
      serverN
    </pool>
  </dbServer>
</dbServerList>
```



redis

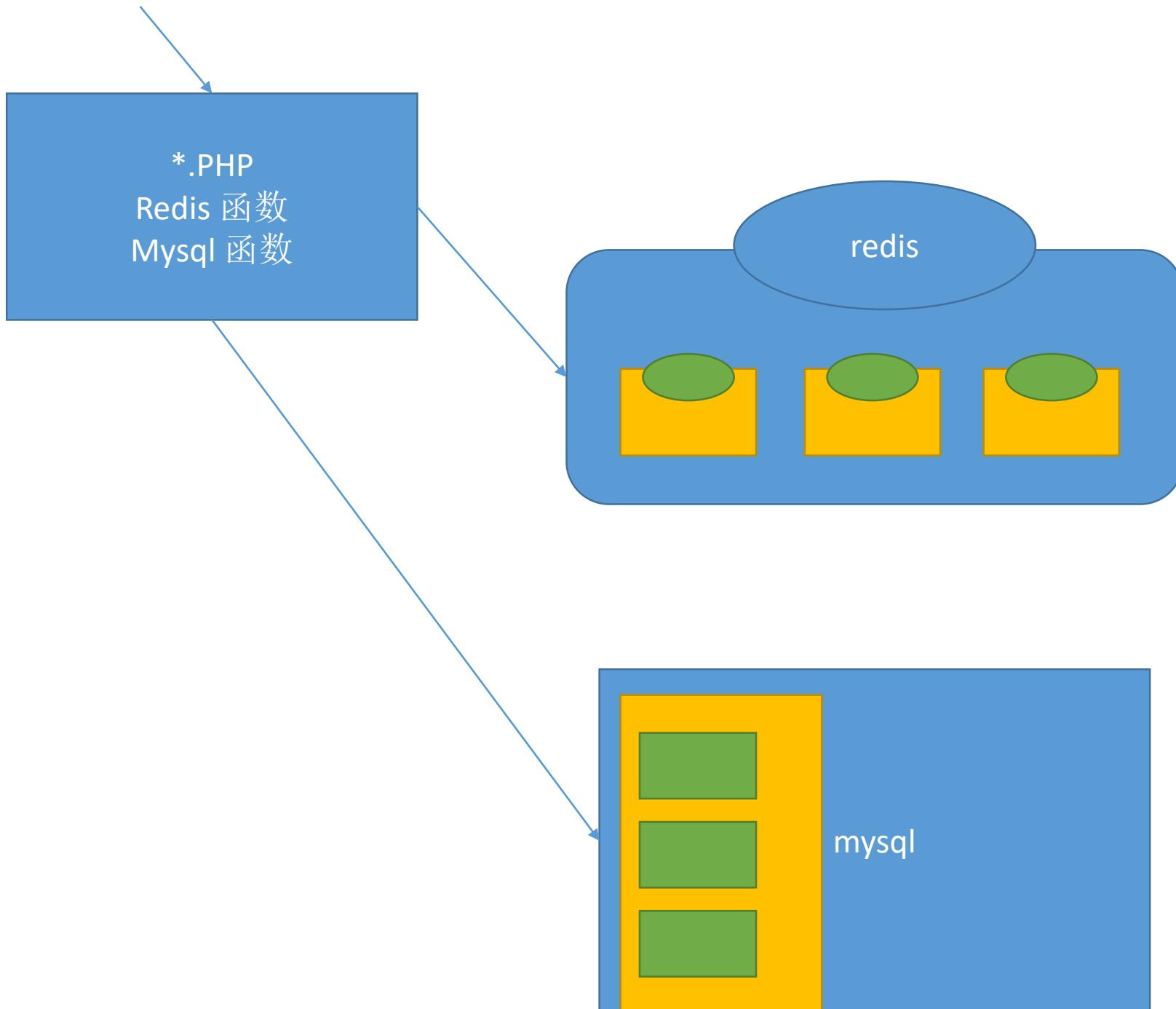
0

荣耀

智能手机

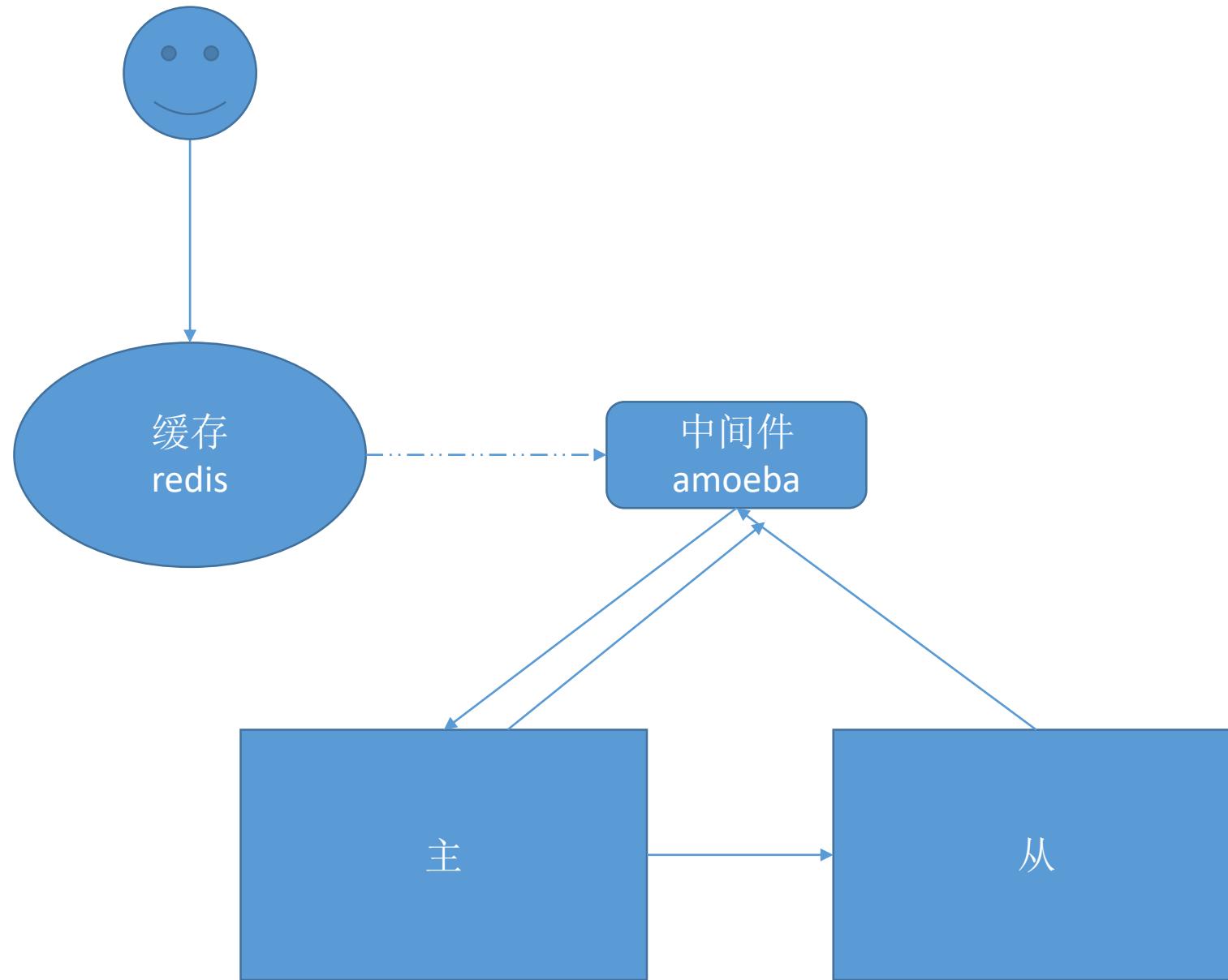
价格
3000-
3500

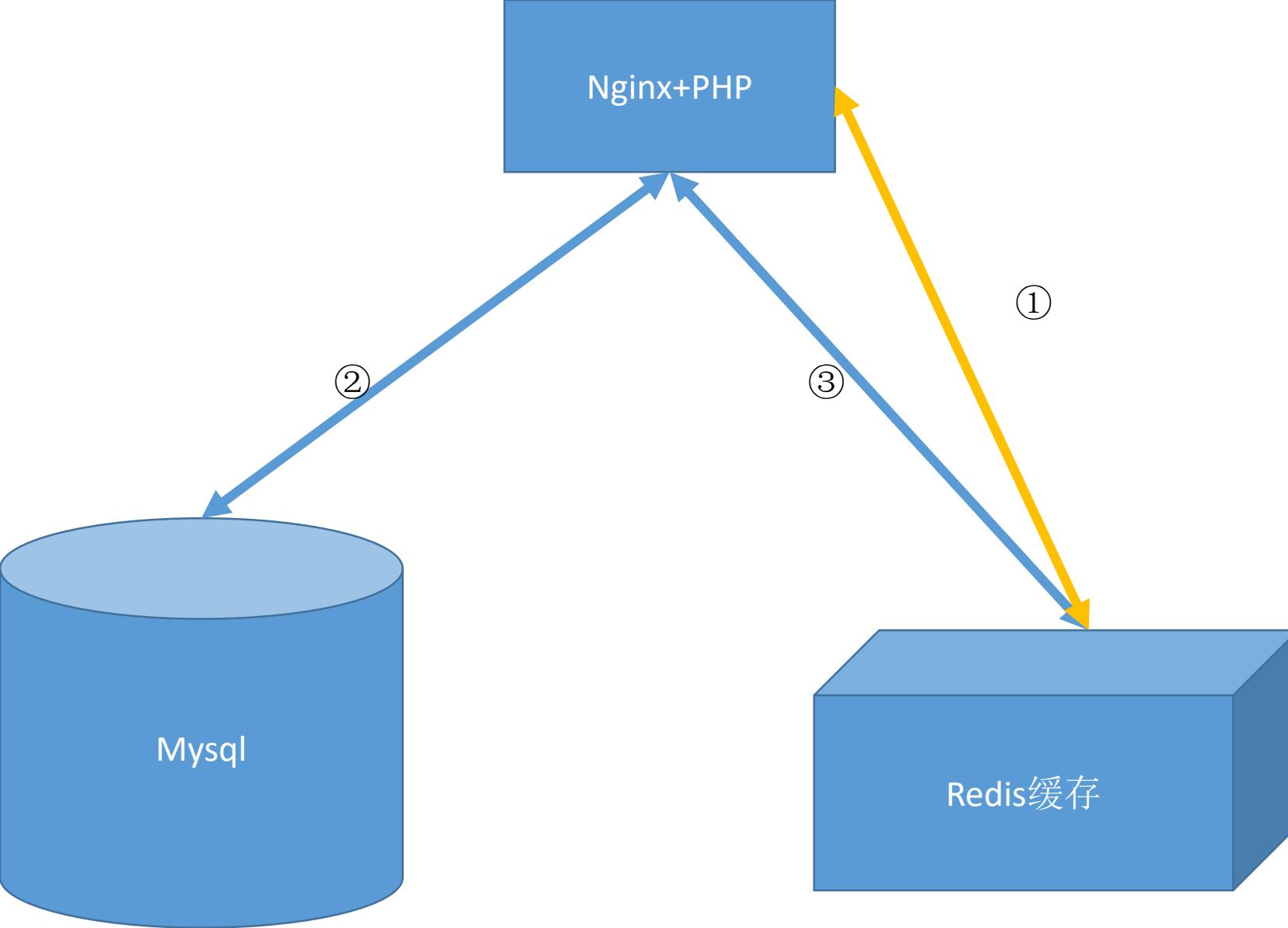
键值对：
键=值
变量=值



当用户通过网页访问数据库时分为两种情况：

1. 用户通过条件过滤产品列表
使用非关系型数据库进行查询，
利用非关系型数据库的交集进行
数据过滤
2. 当点击具体某一产品链接时，
则查询关系型数据库，即mysql





①: PHP通过redis函数请求redis数据库，查看是否有对应条件的检索结果集，若有，则直接套用php页面，并返回给用户，若没有，则php通过mysql函数连接mysql

②: php通过mysql函数连接mysql，一定会通过条件得到结果集，将检索到的结果集套用页面，并返回给用户

③: 并且，将检索结果，主动插入到redis当中，并设置数据的有效期

非关系型数据库-NoSQL

1. 什么是 NoSQL?

NoSQL(NoSQL = Not Only SQL)，意为反 SQL 运动，是一项全新的数据库革命性运动，2000 年前就有人提出，发展至 2009 年趋势越发高涨。它是指运用非关系型的数据存储，相对于铺天盖地的关系型数据库运用，这一概念无疑是一种全新的思维的注入。

随着互联网 web2.0 网站的兴起，传统的关系数据库在应付 web2.0 网站，特别是超大规模和高并发的 SNS 类型的 web2.0 纯动态网站已经显得力不从心，暴露了很多难以克服的问题，而非关系型的数据库则由于其本身的特点得到了非常迅速的发展。NoSQL 数据库的产生就是为了解决大规模数据集合多重数据种类带来的挑战，尤其是大数据应用难题。

分类	Examples 举例	典型应用场景	数据模型	优点
键值(key-value)	Tokyo Cabinet/Tyrant, Redis, Voldemort, Oracle BDB	内容缓存，主要用于处理大量数据的高访问负载，也用于一些日志系统等等。	Key 指向 Value 的键值对，通常用 hash table 来实现	查找速度快
列存储数据库	Cassandra, HBase, Riak	分布式的文件系统	以列簇式存储，将同一列数据存在一起	查找速度快，可扩展性强，更容易进行分布式扩展
文档型数据库	CouchDB, MongoDB	Web 应用（与 Key-Value 类似，Value 是结构化的，不同的是数据库能够了解 Value 的内容）	Key-Value 对应的键值对，Value 为结构化数据	数据结构要求不严格，表结构可变，不需要像关系型数据库一样需要预先定义表结构
图形(Graph)数据库	Neo4J, InfoGrid, Infinite Graph	社交网络，推荐系统等。专注于构建关系图谱	图结构	利用图结构相关算法。比如最短路径寻址，N 度关系查找等

2. NoSQL 的特性？

NoSQL 是 key-value 形式存储，和传统的关系型数据库不一样，不一定遵循传统数据库的一些基本要求，比如说遵循 SQL 标准、ACID 属性、表结构等等。

这类数据库主要有以下特点：

非关系型的、分布式、开源的、水平可扩展的
处理超大量数据

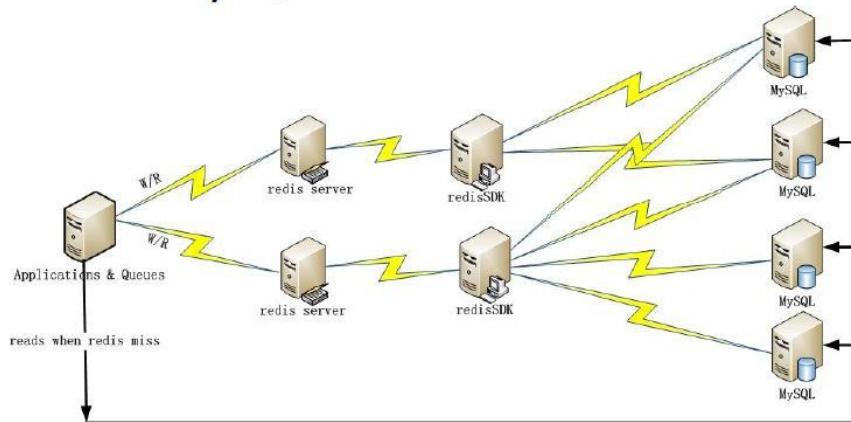
击碎了性能瓶颈
对数据高并发读写
对海量数据的高效率存储和访问
对数据的高扩展性和高可用性

3. 什么是 Redis?

Redis 是一个开源的，先进的 key-value 存储。它通常被称为数据结构服务器，因为键可以包含 string（字符串）、hash（哈希）、list（链表）、set（集合）和 zset（sorted-set--有序集合）。这些数据类型都支持 push/pop、add/remove 及取交集并集和差集及更丰富的操作。

Redis 和 Memcached 类似，它支持存储的 value 类型相对更多，与 memcached 一样，为了保证效率，数据都是缓存在内存中，区别是 Redis 会周期性的把更新的数据写入磁盘或者把修改操作写入追加的记录文件，并且在此基础上实现了 master-slave(主从)同步。

Redis → MySQL



4. Redis 安装部署

Redis 的官方网站是：<http://redis.io>

下载安装包：

```
# wget http://download.redis.io/releases/redis-2.8.6.tar.gz
```

编译安装：

```
# tar -zvxf redis-2.8.6.tar.gz
```

```
# cd redis-2.8.6
```

```
# make
```

```
# make PREFIX=/usr/local/redis install
```

指定安装位置，如果没有指定安装位置 PREFIX=/usr/local/redis，则 make install 会把 redis 安装到 /usr/local/bin/ 目录下

```
# mkdir /usr/local/redis/etc
```

```
# cp ./redis.conf /usr/local/redis/etc/
```

复制 Redis 的配置文件到/usr/local/redis/etc/下，便于管理。

修改配置文件：

```
# vi /usr/local/redis/etc/redis.conf  
daemonize no      修改为 yes      #后台启动
```

启动服务：

```
# /usr/local/redis/bin/redis-server 配置文件  
必须制定配置文件位置，否则会提示警告
```

客户端连接：

```
/usr/local/redis/bin/redis-cli  
-h IP:          连接指定的 redis 服务器  
-p 6379:        指定 redis 服务器的端口  
-a 密码:        使用密码登录  
-n 数据库号:    指定连接哪个数据库  
--raw:          redis 支持存储中文
```

停止 Redis：

```
# /usr/local/redis/bin/redis-cli shutdown  
或  
# pkill -9 redis
```

5. Redis 常用命令

1) string 类型及操作

string 是最简单的类型，一个 key 对应一个 value，string 类型是二进制安全的。redis 的 string 可以包含任何数据。

1. set: 设置 key 对应的值为 string 类型

例如：我们添加一个 name=atguigu 的键值对应

```
redis127.0.0.1:6379>set name atguigu
```

2. setnx: 设置 key 对应的值为 string 类型，如果 key 已经存在，返回 0，nx 是 not exist 的意思

3. get: 获取 key 对应的 string 值，如果 key 不存在返回 nil

4. mset & mget 同时设置和获取多个键值对

5. incrby: 对 key 的值做加加（指定值）操作，并返回新的值

6. del: 删除一个已创建的 key

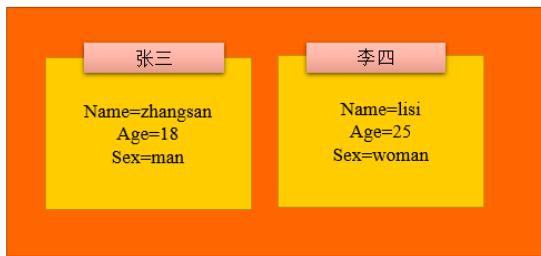
2) hash 类型及操作

Redis hash 是一个 string 类型的 field (字段) 和 value 的映射表，它的添加、删除操作都是 O(1) 平均；hash 特别适合用于存储对象，相较于将对象的每个字段存成单个 string 类型，将一个对象存储在 hash 类型中会占用更少的内存，并且可以更方便的存取整个对象。

1. hset: 设置 hash field 为指定值，如果 key 不存在，则先创建。

例如：为 num1 表创建一个叫 name 字段（key），键值是 liuchuan

```
redis127.0.0.1:6379>hset num1 name liuchuan
```



2. hget、hmset、hmget 意义同上近似
3. hdel: 删除制定表中的某一个键值对
4. hgetall: 列出表中的所有键值对

3) list 类型及操作

list 是一个链表结构，主要功能是 push、pop、获取一个范围内的所有值等等，操作中 key 理解为链表的名字。Redis 的 list 类型其实就是一个每个子元素都是 string 类型的双向链表。我们可以通过 push、pop 操作从链表的头部或尾部添加删除元素。

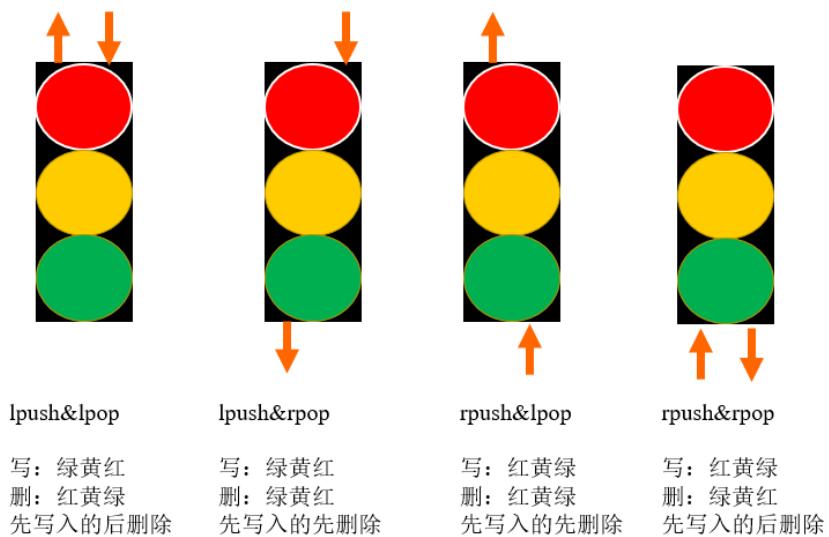


1. lpush: 在 key 对应 list 的头部添加字符串元素。

```
redis127.0.0.1:6379>lpush zhangsan zhangsan  
redis127.0.0.1:6379>lpush zhangsan 18
```
2. lrange: 从指定链表中获取指定范围的元素

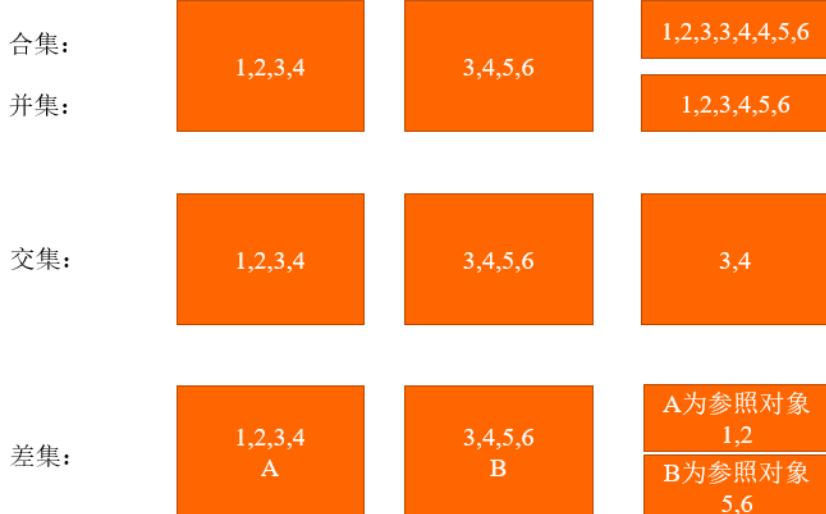
```
redis127.0.0.1:6379>lrange zhangsan 0 -1
```

0 -1: 此范围代表全部元素，意为从头到尾
3. lpush、rpush、lpop、rpop、lrange 详见图示



4) Set 类型及操作

set 是集合，他是 string 类型的无序集合。Set 是通过 hash table 实现的，对集、交集、差集。通过这些操作我们可以实现社交网站中的好友推荐和 blog 的 tag 功能。集合不允许有重复值。



1. **sadd:** 添加一个或多个元素到集合中
redis127.0.0.1:6379>sadd mset 1 2 3 4
2. **smembers:** 获取集合里面所有的元素
redis127.0.0.1:6379> smembers mset
3. **srem:** 从集合中删除指定的一个或多个元素
4. **spop:** 随机从集合中删除一个元素，并返回
5. **scard:** 获取集合里面的元素个数
6. **sdiff:** 返回集合 1 与集合 2 的差集。以集合 1 为主
redis127.0.0.1:6379>sdiff mset1 mset2
7. **sinter:** 获得两个集合的交集

8. sunion: 获得指定集合的并集

5) zset 类型及操作

zset 是 set 的一个升级版本，它在 set 的基础上增加了一个顺序属性，这一属性在添加修改元素的时候可以指定，每次指定后，zset 会自动重新按新的值调整顺序。可以理解为有两列的 mysql 表，一列存的 value，一列存的顺序。操作中 key 理解为 zset 的名字。

下标	分数:标签
0	2 zhangsan
1	1 lisi
2	1 laow

有序集合的排序方式：

1. 先按照预设设置好的分数进行先后排序
2. 相同分数的值，按照值的 ASCII 码表的顺序排序



1. zadd: 向一个指定的有序集合中添加元素，每一个元素会对应的有一个分数。你可以指定多个分数/成员组合。如果一个指定的成员已经在对应的有序集合中了，那么其分数就会被更新成最新的，并且该成员会重新调整到正确的位置，以确保集合有序。分数的值必须是一个表示数字的字符串。

redis127.0.0.1:6379>zadd zset 2 zhangsan 1 lisi 1 wangwu

2. zrange: 返回有序集合中，指定区间内的成员。其中成员按照 score (分数) 值从小到大排序。具有相同 score 值的成员按照字典顺序来排列。

redis127.0.0.1:6379>zrange zset 0 -1 withscores

注：withscores 返回集合中元素的同时，返回其分数 (score)

3. zrem: 删除有序集合中指定的值

redis127.0.0.1:6379>zrem zset zhangsan

4. zcard: 返回有序集合元素的个数

6) 其他相关命令

1. keys: 按照键名查找指定的键。支持通配符 (* ? 等)

redis127.0.0.1:6379>keys h*llo

2. exists: 确认一个键是否存在 (1 表示存在)

3. del: 删除一个键 (通用)

4. expire: 设置一个键 (已存在) 的过期时间，如果键已经过期，将会被自动删除

5. ttl: 以秒为单位，返回指定键的剩余有效时间

当 key 不存在时，返回 -2 。

当 key 存在但没有设置剩余生存时间时，返回 -1 。

否则，以秒为单位，返回 key 的剩余生存时间。

6. select: 选择一个数据库，默认连接的数据库是 0，可以支持共 16 个数据库。在配置文件中，通过 databases 16 关键字定义。

7. move: 将当前数据库的键移动到指定的数据库中

8. type: 返回键的类型

9. dbsize: 返回当前库中键的数量（所有类型）

10. save: 保存所有的数据。很少在生产环境直接使用 SAVE 命令，因为它会阻塞所有的客户端的请求，可以使用 BGSAVE 命令代替。如果在 BGSAVE 命令的保存数据的子进程发生错误的时，用 SAVE 命令保存最新的数据是最后的手段。

11. info: 获取服务器的详细信息

12. config get: 获取 redis 服务器配置文件中的参数。支持通配符

13. flushdb: 删除当前数据库中所有的数据

14. flushall: 删除所有数据库中的所有数据

6. Redis 高级应用

1) 密码防护

给 redis 服务器设置密码

1、修改配置文件

```
# vi /usr/local/redis/etc/redis.conf
```

```
requirepass 123456
```

2、重启 redis

```
# pkill redis
```

```
# .../bin/redis-server .../etc/redis.conf
```

3、客户端登录

```
# .../bin/redis-cli -a 123456
```

或

交互模式下使用【auth 密码】命令

2) 主从同步

Redis 主从复制过程：

- Slave 与 master 建立连接，发送 sync 同步命令
- Master 会启动一个后台进程，将数据库快照保存到文件中，同时 master 主进程会开始收集新的写命令并缓存。
- 后台完成保存后，就将此文件发送给 slave
- Slave 将此文件保存到硬盘上

1. 主服务器给自己设置好密码即可（iptables&SELinux 关闭）

2. 从服务器修改配置文件，用来连接主服务器

老版本：

从：

```
slaveof <masterip> <masterport>      #主服务器的 IP 和端口  
masterauth <masterpass>                #主服务器的密码(主服务器要设置好密码)
```

新版本 redis 5.* 以上：

主：

找到 bind 127.0.0.1 注释掉，或者修改为本机的 IP 地址(重启)

从：

```
replicaof <masterip> <masterport>      #主服务器的 IP 和端口  
masterauth <masterpass>                #主服务器的密码(主服务器要设置好密码)
```

3. 重启从服务器，然后测试（可通过 info 命令获取当前服务器身份类型）

3) 数据持久化

Redis 是一个支持持久化的内存数据库，也就是说需要经常将内存中的数据同步到硬盘来保证持久化。

snapshotting (快照) --默认方式

RDB 持久化方式能够在指定的时间间隔能对你的数据进行快照存储。是默认的持久化方式。这种方式是将内存中数据以快照的方式写入到二进制文件中，默认的文件名为 dump.rdb。这种持久化方式被称为快照 snapshotting (快照)。

```
# 过了 900 秒并且有 1 个 key 发生了改变 就会触发 save 动作  
# 过了 300 秒并且有 10 个 key 发生了改变 就会触发 save 动作  
# 过了 60 秒并且至少有 10000 个 key 发生了改变 也会触发 save 动作
```

结论：在 redis.conf 文件中 dir ./ 定义了数据库文件的存放位置，默认是当前目录。所以每次重启 redis 服务所在的位置不同，将会生成新的 dump.rdb 文件；建议服务器搭建完成时先修改快照文件保存位置。

append-only file (缩写 aof)

使用 AOF 会让你的 Redis 更加耐久：你可以使用不同的持久化策略：每次写的时候备份、每秒备份、无备份。使用默认的每秒备份策略，Redis 的性能依然很好(备份是由后台线程进行处理的，主线程会尽力处理客户端请求)，一旦出现故障，你最多丢失 1 秒的数据。

打开 redis.conf 配置文件开启 AOF 持久化

appendonly no

默认不使用 AOF 持久化 (450 行) 将 no 改成 yes。

appendfsync always

有写操作，就马上写入磁盘。效率最慢，但是最安全

appendfsync everysec

默认，每秒钟写入磁盘一次。

appendfsync no

不进行 AOF 备份，将数据交给操作系统处理。最快，最不安全

测试：重启 redis 服务，登录 client 添加一个键值，退出然后 ls 命令查看下是否生成 appendonly.aof。



可以用 cat 查看。



MySQL+NoSQL(Redis)

1. 安装 gcc*

```
已安装:
gcc.x86_64 0:4.4.7-16.el6      gcc-c++.x86_64 0:4.4.7-16.el6      gcc-gfortran.x86_64 0:4.4.7-16.el6      gcc-gnat.x86_64 0:4.4.7-16.el6
gcc-java.x86_64 0:4.4.7-16.el6  gcc-objc.x86_64 0:4.4.7-16.el6  gcc-objc++.x86_64 0:4.4.7-16.el6

作为依赖被安装:
cloop-ppl.x86_64 0:0.15.7-1.2.el6  cpp.x86_64 0:4.4.7-16.el6  ecj.x86_64 1:3.4.2-6.el6      java-1.5.0-gcj.x86_64 0:1.5.0.0-29.1.el6
java_cup.x86_64 1:0.10k-5.el6    libgcj.x86_64 0:4.4.7-16.el6  libgcj-devel.x86_64 0:4.4.7-16.el6  libgnat.x86_64 0:4.4.7-16.el6
libgnat-devel.x86_64 0:4.4.7-16.el6  libobjc.x86_64 0:4.4.7-16.el6  libstdc++-devel.x86_64 0:4.4.7-16.el6  mpfr.x86_64 0:2.4.1-6.el6
ppl.x86_64 0:0.10.2-11.el6      sinjdoc.x86_64 0:0.5-9.1.el6  zlib-devel.x86_64 0:1.2.3-29.el6

完毕!
[root@lym ~]# yum -y install gcc*
```

2. 安装所需要的包

```
[root@lym ~]# cd /media/
[root@lym media]# ls
autoconf-2.63-5.1.el6.noarch.rpm          libgfortran-4.4.7-17.el6.x86_64.rpm      php-cli-5.3.3-47.el6.x86_64.rpm
automake-1.11.1-4.el6.noarch.rpm          libgnat-4.4.7-17.el6.x86_64.rpm        php-common-5.3.3-47.el6.x86_64.rpm
cloop-ppl-0.15.7-1.2.el6.x86_64.rpm       libgnat-devel-4.4.7-17.el6.x86_64.rpm   php-dba-5.3.3-47.el6.x86_64.rpm
cpp-4.4.7-17.el6.x86_64.rpm               libgomp-4.4.7-17.el6.x86_64.rpm       php-devel-5.3.3-47.el6.x86_64.rpm
ecj-3.4.2-6.el6.x86_64.rpm                libobjc-4.4.7-17.el6.x86_64.rpm       php-enchant-5.3.3-47.el6.x86_64.rpm
gcc-4.4.7-17.el6.x86_64.rpm              libstdc++-4.4.7-17.el6.x86_64.rpm     php-fpm-5.3.3-47.el6.x86_64.rpm
gcc-c++-4.4.7-17.el6.x86_64.rpm          libXpm-3.5.10-2.el6.x86_64.rpm       php-gd-5.3.3-47.el6.x86_64.rpm
gcc-gfortran-4.4.7-17.el6.x86_64.rpm     mpfr-2.4.1-6.el6.x86_64.rpm        php-mbstring-5.3.3-47.el6.x86_64.rpm
gcc-gnat-4.4.7-17.el6.x86_64.rpm         mysql-5.1.73-7.el6.x86_64.rpm       php-mysql-5.3.3-47.el6.x86_64.rpm
gcc-java-4.4.7-17.el6.x86_64.rpm         mysql-libs-5.1.73-7.el6.x86_64.rpm   php-pdo-5.3.3-47.el6.x86_64.rpm
gcc-objc-4.4.7-17.el6.x86_64.rpm         mysql-server-5.1.73-7.el6.x86_64.rpm  phpredis-master.zip
gcc-objc++-4.4.7-17.el6.x86_64.rpm       nginx-1.10.1-1.el6.ngx.x86_64.rpm    php-xml-5.3.3-47.el6.x86_64.rpm
java-1.5.0-gcj-1.5.0.0-29.1.el6.x86_64.rpm perl-DBD-MySQL-4.013-3.el6.x86_64.rpm  php-xmlrpc-5.3.3-47.el6.x86_64.rpm
java_cup-0.10k-5.el6.x86_64.rpm          perl-DBI-1.609-4.el6.x86_64.rpm     ppl-0.10.2-11.el6.x86_64.rpm
libgcc-4.4.7-17.el6.x86_64.rpm          perl-PDBI-1.609-4.el6.x86_64.rpm   redis-2.8.19.tar.gz
libgcj-4.4.7-17.el6.x86_64.rpm          perl-5.3.3-47.el6.x86_64.rpm       sinjdoc-0.5-9.1.el6.x86_64.rpm
libgcj-devel-4.4.7-17.el6.x86_64.rpm    perl-Bcmath-5.3.3-47.el6.x86_64.rpm  zlib-devel-1.2.3-29.el6.x86_64.rpm

[root@lym media]# rpm -qf ppl.x86_64 0:0.10.2-11.el6           sinjdoc.x86_64 0:0.5-9.1.el6
zlib-devel.x86_64 0:1.2.3-29.el6

Dependency Installed:
  apr.x86_64 0:1.3.9-5.el6_2
  apr-util.x86_64 0:1.3.9-3.el6_0.1
  apr-util-ldap.x86_64 0:1.3.9-3.el6_0.1
  enchant.x86_64 1:1.5.0-5.el6
  httpd.x86_64 0:2.2.15-45.el6.centos
  httpd-tools.x86_64 0:2.2.15-45.el6.centos

Updated:
  libgcc.x86_64 0:4.4.7-17.el6           libgfortran.x86_64 0:4.4.7-17.el6
  libgomp.x86_64 0:4.4.7-17.el6          libstdc++.x86_64 0:4.4.7-17.el6

Complete!
[root@lym medial# yum -y install *_
```

3. 配置网站 nginx 并启动 nginx

a. vim /etc/nginx/nginx.conf

```
31      include /etc/nginx/conf.d/*.conf;  主配置文件中没有server
32 }
```

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

```

1 server {
2     listen      80;
3     server_name www.asd.com;           域名
4
5     #charset koi8-r;
6     #access_log  /var/log/nginx/log/host.access.log  main;
7
8     location / {
9         root   /www; 网站页面的位置
10        index index.php index.html index.htm;
11    }                   加一个支持php的
12
13    #error_page 404          /404.html;
14
15    # redirect server error pages to the static page /50x.html
16
17    "location ~ \.php$ {
18        root   /www;
19        fastcgi_pass 127.0.0.1:9000;
20        fastcgi_index index.php;
21        fastcgi_param SCRIPT_FILENAME /www/$fastcgi_script_name;
22        include       fastcgi_params;
23    }
24
25 -- 插入 --

```

启动 nginx

b. vim /etc/php-fpm.d/www.conf

```

37 ; will be used.
38 ; RPM: apache Choosed to be able to access some dir as httpd
39 user = nginx
40 ; RPM: Keep a group allowed to write in log dir.
41 group = nginx
42
43 ; Choose how the process manager will control the number of child processes.
44 ; Possible Values:
45 ; static - a fixed number (pm.max_children) of child processes;
"/etc/php-fpm.d/www.conf" 226L, 10016C

```

4. 启动 php 和数据库

```

[root@lym ~]#
[root@lym ~]# service php-fpm start
正在启动 php-fpm:                                         [确定]
[root@lym ~]# service mysql start
mysql: 未被识别的服务
[root@lym ~]# service mysqld start
初始化 MySQL 数据库: WARNING: The host 'lym' could not be looked up with resolveip.
This probably means that your libc libraries are not 100 % compatible
with this binary MySQL version. The MySQL daemon, mysqld, should work
normally with the exception that host name resolving will not work.
This means that you should use IP addresses instead of hostnames
when specifying MySQL privileges !
Installing MySQL system tables...
OK
Filling help tables...
OK

```

5. 授权，使登录数据库时使用‘123456’密码

```
[root@lym ~]# mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.1.73 Source distribution

Copyright (c) 2000, 2013, 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> grant all privileges on *.* to root@localhost identified by '123456';
Query OK, 0 rows affected (0.01 sec)          授权, 使nginx能读取数据库内容

mysql>
mysql> flush privileges;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the
   near 'privileges' at line 1
mysql> flush privileges;    刷新一下
Query OK, 0 rows affected (0.00 sec)
```

6. 测试网站和 php 的连通性



以上搭建了一个 Inmp 环境。

7. 安裝 redis

```
libgcc-devel-4.4.7-17.el6.x86_64.rpm          php-bcmath-5.3.3-47.el6.x86_64.rpm      zlib-devel-1.2.3-29.el6.x86_
[root@lym media]# tar -zxf redis-2.8.19.tar.gz -C /usr/src/
[root@lym media]#
[root@lym media]# cd /usr/src/
[root@lym src]#
[root@lym src]# cd redis-2.8.19/
[root@lym redis-2.8.19]# ls
00-RELEASENOTES  CONTRIBUTING  deps      Makefile    README      runtest      sentinel.conf  tests
BUGS             COPYING       INSTALL   MANIFESTO  redis.conf  runtest-sentinel  src        utils
[root@lym redis-2.8.19]# make
cd src && make all
make[1]: Entering directory `/usr/src/redis-2.8.19/src'
rm -rf redis-server redis-sentinel redis-cli redis-benchmark redis-check-dump redis-check-aof *.o *.gcda *.gcno *
(cd ..deps && make distclean)
make[2]: Entering directory `/usr/src/redis-2.8.19/deps'
(cd hiredis && make clean) > /dev/null || true
[1.14s] 100% of 1 file(s) checked
[1.14s] 100% of 1 file(s) checked
```

```
[root@lym redis-2.8.19]# make PREFIX=/usr/local/redis install
cd src && make install
make[1]: Entering directory `/usr/src/redis-2.8.19/src'
Hint: It's a good idea to run 'make test' ;)

  INSTALL install
  INSTALL install
  INSTALL install
  INSTALL install
  INSTALL install
make[1]: Leaving directory `/usr/src/redis-2.8.19/src'
[root@lym redis-2.8.19]#
[root@lym redis-2.8.19]#
[root@lym redis-2.8.19]# cp redis.conf /usr/local/redis/
[root@lym redis-2.8.19]#
[root@lym redis-2.8.19]# cd /usr/local/redis/
[root@lym redis]# ls
bin  redis.conf
```

8. 安装提供 php 和 redis 联系的软件

a. 解压

```
gcc-java-4.4.7-17.el6.x86_64.rpm      mysql-5.1.73-7.el6.x86_64.rpm      php_pdo-5.3.3-47.el6.x86_64.rpm
gcc-objc-4.4.7-17.el6.x86_64.rpm      mysql-libs-5.1.73-7.el6.x86_64.rpm    phppredis-master.zip
gcc-objc++-4.4.7-17.el6.x86_64.rpm     mysql-server-5.1.73-7.el6.x86_64.rpm
java-1.5.0-gcj-1.5.0.0-29.1.el6.x86_64.rpm  nginx-1.10.1-1.el6.ngx.x86_64.rpm   php-mem-5.3.3-47.el6.x86_64.rpm
java_cup-0.10k-5.el6.x86_64.rpm        perl-DBD-MySQL-4.013-3.el6.x86_64.rpm  php-xlrm-5.3.3-47.el6.x86_64.rpm
libgcc-4.4.7-17.el6.x86_64.rpm        perl-DBI-1.689-4.el6.x86_64.rpm       ppl-0.10.2-11.el6.x86_64.rpm
libgcj-4.4.7-17.el6.x86_64.rpm        php-5.3.3-47.el6.x86_64.rpm          redis-2.8.19.tar.gz
libgcc-devel-4.4.7-17.el6.x86_64.rpm   php-bcmath-5.3.3-47.el6.x86_64.rpm    sinjdoc-0.5-9.1.el6.x86_64.rpm
[root@lym media]# unzip phppredis-master.zip -d /usr/src/
Archive: phppredis-master.zip
90ecd1798da8177b3f02fd30633de5d127654
  creating: /usr/src/phppredis-master/
  inflating: /usr/src/phppredis-master/CREDITS
  inflating: /usr/src/phppredis-master/README.markdown
  inflating: /usr/src/phppredis-master/common.h
  inflating: /usr/src/phppredis-master/config.m4
```

b. 安装

```
[root@lym media]# cd /usr/src/
[root@lym src]# ls
debug_kernels phppredis-master redis-2.8.19
[root@lym src]# cd phppredis-master/
[root@lym phppredis-master]# ls
common.h CREDITS debian.control library.c mkdeb-apache2.sh README.markdown redis_session.c serialize.list
config.m4 debian igbinary library.h php_redis.h redis.c redis_session.h tests
[root@lym phppredis-master]# phpize
Configuring for:
PHP Api Version: 20090626
Zend Module Api No: 20090626
Zend Extension Api No: 220090626
[root@lym phppredis-master]# ./configure --with-php-config=/usr/bin/php-config
checking for grep that handles long lines and -e... /bin/grep
checking for egrep... /bin/grep -E
checking for a sed that does not truncate output... /bin/sed
checking for... config.status: executing libtool commands
[root@lym phppredis-master]# make && make install
/bin/sh /usr/src/phppredis-master/libtool --mode=compile cc -I. -I/usr/src/phppredis-ma
/srsrc/phppredis-master/main -I/usr/src/phppredis-master -I/usr/include/php -I/usr/include
/usr/include/php/ext -I/usr/include/php/ext/date/lib -DHAVE_CONFIG_H -g -O2 -c /us
libtool: compile: cc -I. -I/usr/src/phppredis-master -DPHP_ATOM_INC -I/usr/src/phppre
phppredis-master -I/usr/include/php -I/usr/include/php/main -I/usr/include/php/TSRM -I/u
```

c. 让 php 支持 redis

```

1650 soap.wsdl_cache_ttl=86400
1651
1652 [sysvshm]
1653 ; A default size of the shared memory segment
1654 ;sysvshm.init_mem = 10000
1655
1656
1657 ; Local Variables:
1658 ; tab-width: 4
1659 ; End:
1660 extension=redis.so 在php主配置文件中添加redis
"/etc/php.ini" 1660L, 69116C

```

```

[root@lym ~]# vim /etc/php.ini
[root@lym ~]# service php-fpm restart
停止 php-fpm. [确定]
正在启动 php-fpm: [确定]
[root@lym ~]#

```

redis

Redis Support	enabled
Redis Version	2.1.0

9. 进入 mysql 插数据

```

mysql> create database mytest;
Query OK, 1 row affected (0.00 sec)

mysql>
mysql> use mytest;
Database changed
mysql> create table test (
    -> id int not null auto_increment,
    -> name char(20) default null,
    -> primary key (id) engine=innodb auto_increment=10 default charset=utf8;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'engine=innodb auto_increment=10 default charset=utf8' at line 4
mysql> create table test (`id int not null auto_increment, name char(20) default null, primary key (id)` engine=innodb auto_increment=10 default charset=utf8; 存储引擎
Query OK, 0 rows affected (0.09 sec)

mysql> describe test;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'describe test' at line 1
mysql> describe test;
+-----+-----+-----+-----+-----+
| Field | Type  | Null | Key | Default | Extra       |
+-----+-----+-----+-----+-----+
| id   | int(11)| NO   | PRI | NULL    | auto_increment
| name | char(20)| YES  |     | NULL    |
+-----+-----+-----+-----+
2 rows in set (0.00 sec)

mysql>
mysql> insert into test values (1,'a1'),(2,'a2'),(3,'a3'),(4,'a4'),(5,'a5');
Query OK, 5 rows affected (0.02 sec)
Records: 5  Duplicates: 0  Warnings: 0

mysql> select * from test;
+---+---+
| id | name |
+---+---+
| 1  | a1  |
| 2  | a2  |
| 3  | a3  |
| 4  | a4  |
| 5  | a5  |
+---+---+
5 rows in set (0.00 sec)

mysql>

```

10. 开启 redis，并编写脚本

```

35 # By default Redis does not run
36 # Note that Redis will write a
37 daemonize yes
38
[root@lym www]# /usr/local/redis/bin/redis-server /usr/local/redis/redis.conf
[root@lym www]# netstat -anpt | grep redis
tcp        0      0 0.0.0.0:6379          0.0.0.0:*                  LISTEN      30951/redis-server
tcp        0      0 ::1:6379             ::*:*                    LISTEN      30951/redis-server
[root@lym www]#

```

```

1 <?php
2     $redis = new redis();
3     $redis->connect('127.0.0.1',6379) or die ("redis not fount");
4     $query = "select * from test limit 4";
5     for ($key=1;$key<5;$key++)
6     {
7         if (!$redis->get($key))
8         {
9             $connect = mysql_connect('127.0.0.1','root','123456');
10            mysql_select_db('mytest');
11            $result = mysql_query($query);
12            while ($row = mysql_fetch_assoc($result))
13            {
14                $redis->set($row['id'],$row['name']);
15            }
16            $myserver = 'mysql';
17            break;
18        }
19        else
20        {
21            $myserver = "redis";
22            $data[$key] = $redis->get($key);
23        }
24    }
25    echo $myserver;
26    echo "<br>";
27    for ($key=1;$key<5;$key++)
28    {
29        echo "number is <b><font color=#FF0000>$key</font></b>";
30        echo "<br>";
31        echo "name is <b><font color=#FF0000>$data[$key]</font></b>";
32        echo "<br>";
33    }
34 ?>
35

```

11. 验证 php 访问 redis 和 mysql

