第 11 讲 Apache 服务器

王晓庆

wangxiaoqing@outlook.com

June 13, 2016

Outline

安装和管理 Apache 服务器

- 安装 Apache 服务器

```
yum install httpd
```

- 管理 Apache 服务器

```
chkconfig --level 345 httpd on
service httpd configtest # 检查配置文件
httpd -t                 # 检查配置文件
service httpd start
# 客户端测试
# 默认网站主目录：/var/www/html
```

主配置文件/etc/httpd/conf/httpd.conf(1)

- 配置文件组成部分
  1. 全局环境
  2. 主服务器配置
  3. 虚拟主机
- 凡是虚拟主机不能处理的请求都由主服务器处理。
- 可对 Apache 服务器进行自上而下的分层管理：
  1. 服务器 (全局)
  2. 网站 (虚拟主机)
  3. 目录 (虚拟目录)
  4. 文件
     - 下级层次的设置继承上级层次，但也可以覆盖上级设置

主配置文件/etc/httpd/conf/httpd.conf(2)

- **基本格式**

  指令名称 参数 # 指令名称不区分大小写，但参数区分大小写

- **容器**
  - **用于封装一组指令，限制指令的条件或作用域**

```
<容器名 参数>
  一组指令
</容器名>
```

<IfModule> 若指定模块存在则执行其内指令，否则忽略
<Directory> 目录容器
<Files> 文件容器
<Location> URL地址容器
<VirtualHost> 虚拟主机容器

Apache 服务器全局配置 (1)

- 设置服务器根目录

```
ServerRoot "/etc/httpd" # 配置、日志及相关文件的根目录
```

- 设置运行 Apache 所使用的 PidFile 路径

```
PidFile run/httpd.pid # 记录父 httpd 进程的 pid
```

- 设置连接参数

```
TimeOut # 连接请求超时时间
KeepAlive # 是否启用持久连接
MaxKeepAliveRequests # 单个持久连接所允许的最大请求数
KeepAliveTimeout # 持久连接中等待下一次请求的最长时间
```

Apache 服务器全局配置 (2)

- 配置 MPM(多处理模块)(1)
  - Apache 是模块化的，通过 MPM 实现模块化功能。MPM 必须静态编译，每次只能有一个 MPM 是活动的。目前主要使用两种模块：
    1. 传统的 prework：每个请求使用一个进程
    2. 线程化的 worker：使用多进程，每个进程又有多个线程
  - 获知当前使用的 MPM 模型

```
httpd -l
```

Apache 服务器全局配置 (3)

- 配置 MPM(多处理模块)(2)
  - prework 默认配置

```
<IfModule prefork.c>
StartServers          8 # 启动时的服务器进程数
MinSpareServers       5 # 最小的空闲子进程数
MaxSpareServers      20 # 最大的空闲子进程数
ServerLimit         256 # 所允许的 MaxClients 最大值
MaxClients          256 # 所允许的最大服务器进程数
MaxRequestsPerChild 4000 # 每个进程可以处理的最大请求数
</IfModule>
```

Apache 服务器全局配置 (4)

- 配置 MPM(多处理模块)(3)
- worker 默认配置

```
<IfModule worker.c>
StartServers              2 # 启动时的服务器进程数
MaxClients              150 # 所允许的最大并发连接数
MinSpareThreads          25 # 最小的空闲线程数
MaxSpareThreads          75 # 最大的空闲线程数
ThreadsPerChild          25 # 每个进程的线程数
MaxRequestsPerChild       0 # 每个进程处理的最大请求数
</IfModule>
```

Apache 服务器全局配置 (5)

- 设置 Apache 服务器侦听的 IP 地址和端口

```
Listen [ip] 80
```

- 设置动态加载模块
  - Apache 是高度模块化的，可将模块直接编译到 Apache 中，也可以动态加载。动态加载模块位于/usr/lib/httpd/modules 目录，可通过 LoadModule 语句加载。

```
LoadModule auth_basic_module modules/mod_auth_basic.so
LoadModule auth_digest_module modules/mod_auth_digest.so
LoadModule authn_file_module modules/mod_authn_file.so
LoadModule authn_alias_module modules/mod_authn_alias.so
LoadModule authn_anon_module modules/mod_authn_anon.so
LoadModule authn_dbm_module modules/mod_authn_dbm.so
......
```

Apache 服务器全局配置 (6)

- 设置包含文件

```
Include conf.d/*.conf
```

- 设置运行 Apache 服务器的用户或群组

```
User apache
Group apache
```

Apache 主服务器基本配置

```
ServerAdmin root@localhost # 管理员 email
ServerName www.example.com:80 # 服务器名和端口
UseCanonicalName Off # 使用客户端提供的服务器名和端口号
DocumentRoot "/var/www/html" # 网站主目录 (最后不能加/)
DirectoryIndex index.html index.html.var # 网站默认文档
ErrorLog logs/error_log # 错误日志
LogLevel warn # 错误日志等级高于等于 warn
CustomLog logs/access_log combined # 访问日志及其格式
LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \
\"%{User-Agent}i\"" combined          # 定义日志格式别名
LogFormat "%h %l %u %t \"%r\" %>s %b" common # 同上
LogFormat "%{Referer}i -> %U" referer        # 同上
LogFormat "%{User-agent}i" agent             # 同上
AddDefaultCharset UTF-8                # 设置默认字符集
```

配置目录访问控制 (1)

- 目录访问控制的两种实现方式
  1. 在 httpd.conf 文件中使用 <Directory> 容器进行设置
  2. 在目录内建立访问控制文件.htaccess，将访问控制参数写入其中，下层目录自动继承上层目录的访问控制设置
- 目录默认配置

```
<Directory />
    Options FollowSymLinks
    AllowOverride None
</Directory>
```

配置目录访问控制 (2)

- 网站根目录配置

```
<Directory "/var/www/html">
    Options Indexes FollowSymLinks
    AllowOverride None # 不允许被.htaccess 的设置覆盖
    Order allow,deny
    Allow from all
</Directory>
```

- Order 指令
  - Order Allow,Deny

    先匹配所有 Allow 指令，再匹配所有 Deny 指令，Deny 指令可以覆盖 Allow 指令，若两者均无匹配则拒绝访问。
  - Order Deny,Allow

    先匹配所有 Deny 指令，再匹配所有 Allow 指令，Allow 指令可以覆盖 Deny 指令，若两者均无匹配则允许访问。
  - Order Mutual-failure

    效果与 Order Allow,Deny 相同，但不建议使用。

配置目录访问控制 (3)

示例 1

```
Order Deny,Allow
Deny from all
Allow from abc.com
# 拒绝所有客户端访问，但允许 abc.com 域的主机访问
```

示例 2

```
Order Allow,Deny
Allow from xyz.net
Deny from jx.xyz.net
# 允许来自 xyz.net 域的主机访问 (jx.xyz.net 子域除外)
# 拒绝所有其他主机
```

- 若示例 2 改为 Order Deny,Allow，则哪些主机可以访问？

配置和管理虚拟目录

- 虚拟目录和物理目录
  - 物理目录：网站主目录内的实际子目录
  - 虚拟目录：网站主目录之外的其他目录，但在 URL 中显示为网站子目录，方便灵活
  - 物理目录与虚拟目录重名时，优先访问虚拟目录。
  - ISP 常使用虚拟目录提供免费个人主页，企业往往使用虚拟目录作为各部门的子网站。
- 创建虚拟目录

```
Alias /icons/ "/var/www/icons/"
Alias /error/ "/var/www/error/"
```

- 根据需要还可以对相应的实际物理目录设置访问权限。

为用户配置个人 Web 空间 (1)

- 1. 修改/etc/httpd/conf/httpd.conf 文件

```
<IfModule mod_userdir.c>
    UserDir disable root # 禁止 root 用户使用个人主页
    UserDir public_html  # 设置用户 Web 站点目录
</IfModule>
# 取消注释以下容器及其指令
<Directory /home/*/public_html>
    AllowOverride FileInfo AuthConfig Limit
    ......
</Directory>
```

为用户配置个人 Web 空间 (2)

- 2. 创建用户个人主页主目录并修改权限

```
su - mike
mkdir public_html
chmod 711 ~mike
chmod 755 ~mike/public_html
```

- 3. 在 public 目录中创建 index.html 文件
- 4. 重启 httpd 服务，或重新加载 httpd 配置文件

```
service httpd restart|reload
```

- 5. 测试访问

```
http://192.168.0.200/~mike
```

配置 PHP 应用程序

- 1. 安装 PHP 解释器

```
yum install php-common php-cli php
```

- 2. 配置 Apache 以支持 PHP

```
vim /etc/httpd/conf/httpd.conf
...
Include conf.d/*.conf

cat /etc/httpd/conf.d/php.conf
ls /etc/php.ini #php 本身的配置文件
```

- 3. 测试

```
cd /var/www/html
cat '<?phpinfo()?>' >test.php # 创建 php 测试页面
访问 http://192.168.56.200/test.php
```

配置和管理 MySQL 数据库服务器 (1)

- 1. 安装 MySQL 及相关程序

```
yum install perl-DBI perl-DBD-MySQL mysql mysql-server
service mysqld start
# 管理员账户为 root, 默认密码为空
mysqladmin -u root password 123456 # 首次设置 root 用户
mysqladmin -u root -p password 111111 # 修改 root 用户密
mysql -uroot -p111111 # 登录 mysql, -p 后面不能有空格！
mysql> show databases; # 后面要加分号
mysql> exit
```

配置和管理 MySQL 数据库服务器 (2)

- 2. 使用 phpMyAdmin 管理 MySQL

```
yum install php-mysql php-mbstring
# 下载 phpMyAdmin 至/var/www/html 目录
cd /var/www/html
tar -xjvf phpMyAdmin-2.11.11.3-all-languages.tar.bz2
mv phpMyAdmin-2.11.11.3-all-languages phpMyAdmin
cd phpMyAdmin
cp -p config.sample.inc.php config.inc.php
vim config.inc.php
$cfg['blowfish_secret'] = 'www.abc.com';
...
$cfg['Servers'][$i]['user'] = 'root';
$cfg['Servers'][$i]['password'] = '111111';
访问 http://192.168.56.200/phpMyAdmin
```

基于 IP 的虚拟主机 (1)

```
#1. 为服务器配置多块网卡或配置虚拟网卡
ifconfig eth0:0 192.168.56.201
ifconfig eth0:1 192.168.56.202
#2. 为虚拟主机注册域名，此处为方便可直接修改 hosts 文件
vim /etc/hosts
......
192.168.56.201   www.abc.com
192.168.56.202   www.xyz.net
#3. 为两个网站分别创建网站根目录
mkdir -p /var/www/abc.com
mkdir -p /var/www/xyz.net
#4. 在两个网站的根目录中分别创建 index.html 文件
#5. 编辑 httpd.conf 文件，确认配置有以下 Listen 指令
Listen 80
```

基于 IP 的虚拟主机 (2)

```
#6. 编辑 httpd.conf 文件，定义虚拟主机
<VirtualHost 192.168.56.201>
    ServerName www.abc.com
    DocumentRoot /var/www/abc.com
</VirtualHost>
<VirtualHost 192.168.56.202>
    ServerName www.xyz.net
    DocumentRoot /var/www/xyz.net
</VirtualHost>
#7. 重启 httpd 服务
service httpd restart
#8. 测试访问
```

1. 如果配置了 DNS，则即可以通过 ip 也可以通过域名访问。
2. 对于<VritualHost> 容器中未定义地址的请求(如 localhost)，都将指向主服务器。

基于名称的虚拟主机 (1)

- 定义
  - 将多个域名绑定到同一个 IP 地址，服务器通过 HTTP 请求中的主机名来确定客户请求的是哪个网站。
- 特点
  - 节省 IP 地址，只能通过域名访问，不能通过 IP 地址访问
- 虚拟主机匹配顺序
  - 检查 HTTP 请求是否使用了与 NameVirtualHost 指令匹配的 ip 地址，如果是，则逐一查找使用该 IP 地址的 <VirtualHost> 段，并尝试找出一个 ServerName 或 ServerAlias 指令与该请求匹配的虚拟主机。如果无匹配，则使用符合该地址的第一个虚拟主机，即排在最前面的虚拟主机成为默认虚拟主机。
  - 当请求的 IP 地址与 NameVirtualHost 指令中的地址匹配时，主服务器中的 DocumentRoot 将永远不会被用到。

基于名称的虚拟主机 (2)

- 示例 1：在单一 IP 地址上运行多个基于名称的网站 (1)

```
#1. 配置域名解析，为方便可以编辑 hosts 文件
vim /etc/hosts
192.168.56.200   www.abc.com
192.168.56.200   www.xyz.net
#2. 为每个网站创建根目录
mkdir -p /var/www/abc.com
mkdir -p /var/www/xyz.net
#3. 在每个网站根目录下准备 index.html 文件
#4. 编辑 httpd.conf 文件
......
Listen 80
......
```

基于名称的虚拟主机 (3)

- 示例 1：在单一 IP 地址上运行多个基于名称的网站 (2)

```
#5. 配置虚拟主机
NameVirtualHost *:80 # 侦听所有 IP 地址的虚拟主机请求
<VirtualHost *:80>    # 地址要与 NameVirtualHost 指令一致
    ServerName www.abc.com
    DocumentRoot /var/www/abc.com
</VirtualHost>
<VirtualHost *:80>
    ServerName www.xyz.net
    DocumentRoot /var/www/xyz.net
</VirtualHost>
#6. 重启 httpd 服务并测试
service httpd restart
```

- 说明：本例中 * 与任何地址均匹配，因此 www.abc.com 将成为默认或主服务器。

基于名称的虚拟主机 (4)

- 示例 2：在多个 IP 地址上运行基于名称的 Web 网站 (2)

```
Listen 80
ServerName www.mainsite.com
DocumentRoot "/var/www/html"
NameVirtualHost 192.168.56.201
<VirtualHost 192.168.56.201>
    ServerName www.abc.com
    DocumentRoot /var/www/abc.com
</VirtualHost>
<VirtualHost 192.168.56.201>
    ServerName www.xyz.net
    DocumentRoot /var/www/xyz.net
</VirtualHost>
```

- 说明：对于 192.168.56.201 之外的访问均由主服务器响应，而 www.abc.com 则是 192.168.56.201 上的默认服务器。

基于名称的虚拟主机 (5)

- 示例 3：在不同地址 (内外网) 上运行相同的网站

```
NameVirtualHost 192.168.56.200
NameVirtualHost 200.1.1.200
<VirtualHost 192.168.56.200 200.1.1.200>
    ServerName www.abc.com
    ServerAlias webserver
    DocumentRoot /var/www/abc.com
</VirtualHost>
```

- 内网用户可以使用 ServerAlias 定义的别名 webserver，而不用 www.abc.com 来访问网站。

基于 TCP 端口号架设多个 Web 网站

- 示例

```
Listen 80
Listen 8080
<VirtualHost 192.168.56.200:80>
    ServerName www.abc.com
    DocumentRoot /var/www/abc.com
</VirtualHost>
<VirtualHost 192.168.56.200:8080>
    ServerName www.xyz.net
    DocumentRoot /var/www/xyz.net
</VirtualHost>
```

用户认证 (1)

- 认证指令：可以出现在 <Directory> 容器或.htaccess 文件中。

```
AuthType Basic|Digest
#Digest 更安全，但并非所有浏览器都支持，因此通常用 Basic
AuthName "Please login: "              # 设置登录提示内容
AuthUserFile /etc/httpd/testpwd   # 设置用户密码文件
AuthGroupFile /etc/httpd/testgrp # 设置组密码文件
```

- 授权命令：为指定用户/组授权访问资源

```
Require user usr1 usr2 ...   # 为指定用户授权
Require group grp1 grp2 ... # 为指定组授权
Require valid-user # 为认证密码文件中所有用户授权
```

用户认证 (2)

- **管理密码文件**

```
用户名：加密的密码 # 密码文件格式
htpasswd [-c][-m][-D] 密码文件名 用户名
#-c：创建密码文件，若文件已存在则清空并改写
#-m：使用 md5 加密密码
#-D：如果用户存在于密码文件中，则删除该用户
htpasswd -c file user # 添加认证用户并创建密码文件
htpasswd file user # 在现有密码文件中添加/修改用户密码
```

注意

密码文件应存储在不能被网络用户访问的位置，以免被下载！

用户认证 (3)

- 示例：使用基本认证方法实现 Web 用户认证

```
#1. 为用户 mike 创建一个密码文件
htpasswd -c /etc/httpd/passwords mike
#2. 配置 Web 服务器，要求经过认证才能访问某网站/目录
<Directory "/var/www/html/dev">
    AuthType Basic
    AuthName "Restricted Files:"
    AuthUserFile /etc/httpd/passwds
    Require user mike # 授权 mike 可以访问该目录
</Directory>
#3. 重启 httpd 服务
service httpd restart
#4. 访问测试
http://192.168.56.200/dev
```

访问控制

- 限制目录访问：<Directory> 容器
- 限制文件访问：<Files> 容器

```
<Files ~ "^\.ht"> # 文件名以.ht 开头的文件
    Order allow,deny
    Deny from all # 拒绝所有客户端访问
</Files>
```

- 限制 URL 地址访问：<Location> 容器

```
<Location /inner> # 网站中以/inner 开头的 URL
    Order deny,allow
    Deny from all
    Allow from 192.168.0.1
</Location>
```

- 通过本地文件权限控制访问

为 Apache 服务器配置 SSL(1)

- 基于 SSL 的 Web 网站可以实现以下安全目标：
  1. Web 浏览器确认 Web 服务器的身份，防止假冒网站。
  2. 在 Web 服务器和 Web 浏览器之间建立安全的数据通道，确保安全传输敏感数据，防止数据被第三方非法获取。
  3. 如有必要，可以让服务器确认用户身份，防止假冒用户。
- 假设 SSL 安全网站，关键要具备以下条件：
  1. 需要从可信的或权威的证书颁发机构 (CA) 获取证书，也可以创建自签名的证书 (X509 结构)。另外还要保证证书不能过期。
  2. 必须在 Web 服务器上安装服务器证书并启用 SSL 功能。
  3. 如果要求对客户端进行身份验证，则客户端需要申请和安装用户证书；否则，客户端必须与服务器信任同一证书认证机构，需要安装 CA 证书。

为 Apache 服务器配置 SSL(2)

- 1. 安装必要的软件包

```
rpm -qa | grep openssl
rpm -qa | grep mod_ssl
```

- 2. 为 Apache 服务器准备 SSL 证书

```
cd /etc/pki/tls/private
openssl genrsa -out abcsrv.key 1024 # 为服务器创建私钥
# 利用上一步产生的私钥创建一个证书签名请求文件
openssl req -new -key abcsrv.key -out abcsrv.csr
mkdir /etc/pki/tls/csr; mv abcsrv.csr /etc/pki/tls/csr
cd /etc/pki/tls/certs
# 基于服务器私钥为服务器创建一个自签名证书
openssl x509 -req -days 365 -in \
/etc/pki/tls/csr/abcsrv.csr -signkey \
/etc/pki/tls/private/abcsrv.key -out abcsrv.crt
```

为 Apache 服务器配置 SSL(3)

- 3. 为 Apache 服务器启用 SSL 功能

```
vim /etc/httpd/conf.d/ssl.conf
LoadModule ssl_module modules/mod_ssl.so
Listen 443
<VirtualHost _default_:443>
SSLEngine on  # 启用 SSL 功能
# 设置服务器证书文件
SSLCertificateFile /etc/pki/tls/certs/abcsrv.crt
# 设置服务器私钥文件
SSLCertificateKeyFile /etc/pki/tls/private/abcsrv.key
</VirtualHost>
```

- 4. 重启 httpd 服务

```
service httpd restart
```

为 Apache 服务器配置 SSL(4)

- 5. 客户端基于 SSL 连接到 Apache 服务器

  访问 `https://www.abc.com`

- 强制客户端使用 SSL 连接
  - 按照上述配置，HTTP 和 HTTPS 两种通信连接都支持。如果要强制客户端使用 HTTPS，只要屏蔽非 SSL 网站即可。例如，不允许侦听 80 端口，或不配置 80 端口的虚拟主机。

为 Apache 服务器配置 SSL(5)

- 为 Apache 虚拟主机启用 SSL(1)
  - 基于 IP 的虚拟主机：需为每个域名/IP 申请证书

```
Listen 443
<VirtualHost 192.168.56.201:443>
SSLEngine on
SSLCertificateFile /etc/pki/tls/certs/abcsrv.crt
SSLCertificateKeyFile /etc/pki/tls/private/abcsrv.key
......
</VirtualHost>
<VirtualHost 192.168.56.202:443>
SSLEngine on
SSLCertificateFile /etc/pki/tls/certs/xyzsrv.crt
SSLCertificateKeyFile /etc/pki/tls/private/xyzsrv.key
......
</VirtualHost>
```

为 Apache 服务器配置 SSL(6)

- 为 Apache 虚拟主机启用 SSL(2)
  - 基于 TCP 端口的虚拟主机：只需为一个域名/IP 申请证书

```
Listen 443
Listen 8443
<VirtualHost 192.168.56.201:443>
SSLEngine on
SSLCertificateFile /etc/pki/tls/certs/abcsrv.crt
SSLCertificateKeyFile /etc/pki/tls/private/abcsrv.key
......
</VirtualHost>
<VirtualHost 192.168.56.201:8443>
SSLEngine on
SSLCertificateFile /etc/pki/tls/certs/abcsrv.crt
SSLCertificateKeyFile /etc/pki/tls/private/abcsrv.key
......
</VirtualHost>
```

监控 Apache 服务器

- 编辑 httpd.conf，去掉以下指令前的注释：

```
<Location /server-status>
    SetHandler server-status
    Order deny,allow
    Deny from all
    Allow from .example.com
</Location>
```

- 访问 http://192.168.56.200/server-status

查看 Apache 服务器配置信息

- 编辑 httpd.conf，去掉以下指令前的注释：

```
<Location /server-info>
    SetHandler server-status
    Order deny,allow
    Deny from all
    Allow from .example.com
</Location>
```

- 访问 http://192.168.56.200/server-info

查看和分析 Apache 服务器日志 (1)

- 检查错误日志和访问日志

```
vim /etc/httpd/conf/httpd.conf
......
ErrorLog logs/error_log                 # 配置错误日志
......
CustomLog logs/access_log common    # 配置访问日志

tail -f /etc/httpd/logs/error_log # 实时监测错误日志
```

查看和分析 Apache 服务器日志 (2)

- 使用 Webalizer 分析访问日志

```
vim /etc/httpd/conf.d/webalizer.conf
Alias /usage /var/www/usage
<Location /usage>
    Order deny,allow
    Deny from all
    Allow from 127.0.0.1
    Allow from ::1
    Allow from 192.168.56.1 # 增加客户端许可
</Location>

service httpd restart
访问 http://192.168.56.200/usage
```