Linux 服务篇之——httpd的配置(一) - linux菜鸟的IT之路丶 - 51CTO技术博客

Httpd（即Apache）作为当今最流行的的web server之一，在互联网中起着至关重要的作用，那么这么一个强大的工具是如何配置工作的呢，本节我就带大家了解实现一下httpd的基本配置。

在这里一些基本的httpd知识点我可能介绍的不是那么的详细，希望大家不是很懂的地方可以通过Google等方式进行查找了解。那么下面我们就开始吧。

实验运行环境：

Centos6.5x86_64

IP：172.16.249.57

Httpd的安装

Httpd的安装方式通常有两种，一直是直接利用rpm包进行直接安装，另一种是源码编译安装，这里我们先用rpm包直接安装就行了，至于源码编译安装的方式我会在后面进行介绍并实现。OK，那就开始安装了。

```
1
```

```
[root@localhost ~]# yum install httpd
```

安装好之后我们看下生成了哪些文件

```
1
```

```
[root@localhost ~]# rpm -ql httpd
```

```
[root@localhost ~]# rpm -ql httpd
/etc/httpd
/etc/httpd/conf
/etc/httpd/conf.d
/etc/httpd/conf.d/README
/etc/httpd/conf.d/welcome.conf
/etc/httpd/conf/httpd.conf
/etc/httpd/conf/magic
/etc/httpd/logs
/etc/httpd/modules
/etc/httpd/run
/etc/logrotate.d/httpd
/etc/rc.d/init.d/htcacheclean
/etc/rc.d/init.d/httpd
/etc/sysconfig/htcacheclean
/etc/sysconfig/httpd
/usr/lib64/httpd
/usr/lib64/httpd/modules
/usr/lib64/httpd/modules/mod_actions.so
/usr/lib64/httpd/modules/mod_alias.so
/usr/lib64/httpd/modules/mod_asis.so
/usr/lib64/httpd/modules/mod_auth_basi
/usr/lib64/httpd/modules/mod_auth_dige
/usr/lib64/httpd/modules/mod_authn_alias.so
```


wKioL1PnWJLwFmmUAAHUWDQzTa8687.jpg

下面是几个比较重要的文件

配置文件：

/etc/httpd/conf/httpd.conf

/etc/httpd/conf.d/*.conf

服务脚本：

/etc/rc.d/init.d/httpd

脚本配置文件：**/etc/sysconfig/httpd**

模块目录：

/etc/httpd/modules:　　链接文件

/usr/lib64/httpd/modules

主程序：

/usr/sbin/httpd: prefork

/usr/sbin/httpd.event: event

/usr/sbin/httpd.worker: worker

日志文件目录：

/var/log/httpd

access_log: 访问日志

error_log: 错误日志

好了下面我们就对主配置文件进行介绍了。

注意：由于该配置文件比较大，找配置的时候不好找，因此多多使用vim的查找功能。

```
1  [root@localhost ~]#vim /etc/httpd/conf/httpd.conf
```

持久连接（又称为长连接）

这里需要解释一下持久连接，持久连接即是在规定的一定时间或者规定的请求数内，客户端向服务器端请求资源时，不必每次都重新经过TCP三次握手建立连接，而是直接请求，这样就大大的节约了时间。但它并非适合所有请求者，对于一些只请求一次或者次数很少的用户，这就变成了占用资源，因此长连接直接影响到了服务器的性能。

wKiom1PnV-nhhTEbAADQN4dQFI8053.jpg

Timeout 60

在客户端和服务器端TCP三次握手的时候，当客户端发起请求，服务器端响应请求之后，服务器端等待客户端确认的时间，如果客户端在时间内未确认，则服务器将关闭该次TCP握手。

KeepAlive {On|Off}

持久连接是否启用

MaxKeepAliveRequests 100

服务器单个持久连接最大的请求数，超过即断开

KeepAliveTimeout 15

单个持久连接最大连接时长，超过即断开

MPM参数

这里定义的是httpd的工作模式（注意:只能使用其中一种）

httpd在linux下默认使用prefork, 当然这是我们可以自己定义的。

prefork： httpd使用进程来提供服务，每个进程在同一时间提供一次服务。

worker： httpd在启动的时候，会由root进程派生出几个子进程，每个子进程中会有固定数量的线程，到时候提供服务的，就是这些线程，也就是说一个进程能够同时提供多次服务。

```
<IfModule prefork.c>
StartServers       8
MinSpareServers    5
MaxSpareServers   20
ServerLimit      256
MaxClients       256
MaxRequestsPerChild  4000
</IfModule>

# worker MPM
# StartServers: initial number of server processes to start
# MaxClients: maximum number of simultaneous client connections
# MinSpareThreads  minimum number of worker threads which are kept spare
# MaxSpareThreads  maximum number of worker threads which are kept spare
# ThreadsPerChild: constant number of worker threads in each server process
# MaxRequestsPerChild: maximum number of requests a server process serves
<IfModule worker.c>
StartServers         4
MaxClients         300
MinSpareThreads     25
MaxSpareThreads     75
ThreadsPerChild     25
MaxRequestsPerChild  0
</IfModule>
```


wKioL1PnWRCRAYRkAAHixoa9DUY726.jpg

注意，这里面的参数尤其关键，务必要搞清楚每个参数的意义。

StartServers 8 是httpd刚启动时，root进程创建的子进程数

MinSpareServers 5 最少的空闲子进程数

MaxSpareServers 20 最大的空闲子进程数

ServerLimit 256 服务器能接受的最大的并发请求数

MaxClients 256 客户端最大并发请求的数量 显然它必须小于serverlimit

MaxRequestsPerChild 4000 每个子进程最多可以接受的请求数，超过即KILL

StartServers 4 是httpd刚启动时，root进程创建的子进程数

MaxClients 300 客户端最大并发请求的数量

MinSpareThreads 25 最少的空闲线程数

MaxSpareThreads 75 最大的空闲进程数

ThreadsPerChild 25 每个子进程最多可以生成的线程数

MaxRequestsPerChild 0 每个子进程可接受的请求数，0表示任意个

指定监听的地址和端口

格式：Listen [IP:]PORT IP地址可以省略

端口是可以定义多个的，他并不是唯一的

一般httpd默认监听在80端口

```
#Listen 12.34.56.78:80
Listen 80
```

wKioL1PnWS3CBD17AAAZiumeCRU516.jpg

DSO机制装载的模块

DSO即Dynamic Shared Object动态分享对象。

这里定义了系统装载的一些模块，如截图所示，其格式为

LoadModule Module_Name（模块名称） /path/to/Module_File

（模块路径，这里是相对路径，相对于前面定义的

```
ServerRoot "/etc/httpd"
```

wKiom1PnWXuSZSx_AAAVEgpOTDk649.jpg

目录下，当然也可以使用绝对路径）

如果你想添加模块的话，直接按照格式写在配置文件中即可；

当然，你不想启用的话可以直接通过#进行注释即可，保存后即可生效。

```
# Dynamic Shared Object (DSO) Support
#
# To be able to use the functionality of a module which was built as a DSO you
# have to place corresponding `LoadModule' lines at this location so the
# directives contained in it are actually available _before_ they are used.
# Statically compiled modules (those listed by `httpd -l') do not need
# to be loaded here.
#
# Example:
# LoadModule foo_module modules/mod_foo.so
#
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
LoadModule authn_default_module modules/mod_authn_default.so
LoadModule authz_host_module modules/mod_authz_host.so
LoadModule authz_user_module modules/mod_authz_user.so
LoadModule authz_owner_module modules/mod_authz_owner.so
LoadModule authz_groupfile_module modules/mod_authz_groupfile.so
LoadModule authz_dbm_module modules/mod_authz_dbm.so
LoadModule authz_default_module modules/mod_authz_default.so
LoadModule ldap_module modules/mod_ldap.so
LoadModule authnz_ldap_module modules/mod_authnz_ldap.so
LoadModule include_module modules/mod_include.so
LoadModule log_config_module modules/mod_log_config.so
```

wKiom1PnWYiBNDDYAAMU1E1sEeE825.jpg

你也可以通过命令查看装载的模块信息

```
1  # httpd -D DUMP_MODULES
```

```
[root@localhost ~]#  httpd -D DUMP_MODULES
httpd: Could not reliably determine the server's fully qualified domain name, using localhost.localdomain for ServerName
Loaded Modules:
 core_module (static)
 mpm_prefork_module (static)
 http_module (static)
 so_module (static)
 auth_basic_module (shared)
 auth_digest_module (shared)
 authn_file_module (shared)
 authn_alias_module (shared)
 authn_anon_module (shared)
 authn_dbm_module (shared)
 authn_default_module (shared)
 authz_host_module (shared)
 authz_user_module (shared)
 authz_owner_module (shared)
 authz_groupfile_module (shared)
 authz_dbm_module (shared)
 authz_default_module (shared)
 ldap_module (shared)
 authnz_ldap_module (shared)
 include_module (shared)
 log_config_module (shared)
 logio_module (shared)
 env_module (shared)
 ext_filter_module (shared)
 mime_magic_module (shared)
 expires_module (shared)
 deflate_module (shared)
 headers_module (shared)
 usertrack_module (shared)
 setenvif_module (shared)
```

wKioL1PnWrLgumiYAAI-KnH73T8343.jpg

指定站点根目录

```
DocumentRoot "/var/www/html"
```

wKiom1PnWcbD9L2NAAAbX1syWFQ273.jpg

如图，DocumentRoot定义了httpd服务器的站点根目录，你在互联网上访问该服务器时访问的都是该根目录下的内容；当然，路径也是可以自己定义的，默认一般都为/var/www/html

站点路径访问控制

这里的访问控制可以有2种方式：

基于本地文件系统路径：

格式

基于URL访问路径做访问控制

格式

如下图所示：

这是一个基于本地文件系统的访问控制：这里对里面的一些选项进行介绍

(1) **Options**

Indexes: 当访问的路径下无默认的主页面，将所有资源以列表形式呈现给用户；

这项比较危险，一般不建议启用；当然如果作为文件服务器让别人下载文件的话可以启用。

FollowSymlinks: 跳跃符号链接，直接相当于访问符号链接指向的文件。

(2) **AllowOverride**

支持在每个页面目录下创建.htaccess用于实现对此目录中资源访问时的访问控制功能。

（3）**Order**

Deny为拒绝，**allow**为允许。

这里可以对IP地址或网络进行控制。

注意：网络地址格式较为灵活：

172.16

172.16.0.0

172.16.0.0/16

172.16.0.0/255.255.0.0

可以通过deny和allow的先后顺序不同来定义白名单和黑名单

例如：

```
order deny allow
deny 192.168.0.1
```

这样就定义了一个黑名单，除了**192.168.0.1**都可以访问

再如：

```
order allow deny
allow 172.16.0.0/16
```

这样就定义了个白名单，除了**172.16.0.0**网段的都不能访问

```
<Directory "/var/www/html">

#
# Possible values for the Options directive are "None
# or any combination of:
#   Indexes Includes FollowSymLinks SymLinksifOwnerMa
#
# Note that "MultiViews" must be named *explicitly* -
# doesn't give it to you.
#
# The Options directive is both complicated and impor
# http://httpd.apache.org/docs/2.2/mod/core.html#opti
# for more information.
#
    Options Indexes FollowSymLinks

#
# AllowOverride controls what directives may be place
# It can be "All", "None", or any combination of the
#   Options FileInfo AuthConfig Limit
#
    AllowOverride None

#
# Controls who can get stuff from this server.
#
    Order allow,deny
    Allow from all

</Directory>

#
```

wKioL1PnW3PwJtY3AAGKhf-pqko402.jpg

定义默认的主页面

DirectoryIndex可以定义服务器的默认主页面

```
#
DirectoryIndex index.html index.html.var
```

wKioL1PnW5KjWi6aAAAhUhMCJns267.jpg

这里需要解释的是：当通过互联网访问你的服务器时，访问的是某个路径，而非路径下的文件时，如果该路径下

有对应的index.html或者index.html.var文件，则显示为该文件的内容，否则，则会根据站点访问控制里的options选项显示相应的内容。

配置日志功能

这里定义了错误日志以及访问日志，日志的等级，日志的格式等。

ErrorLog "/path/to/error_log" 错误日志路径

LogLevel {debug|info|notice|warn|error|crit|alert|emerg} 日志等级

LogFormat 日志格式

CustomLog "/path/to/access_log" LogFormat_Name 访问日志路径以及日志格式名称

下面是日志格式中一些选项的意义。

%h: 客户端地址

%l: 远程登录名，通常为-

%u: 认证时输入用户名，没有认证时为-

%t: 服务器收到 用户请求时的时间

%r: 请求报名的起始行

%>s: 响应状态码

%b: 响应报文的长度，单位是字节

%{HEADER_NAME}i: 记录指定首部对应的值

```
ErrorLog logs/error_log

#
# LogLevel: Control the number of messages logged to the error_log.
# Possible values include: debug, info, notice, warn, error, crit,
# alert, emerg.
#
LogLevel warn

#
# The following directives define some format nicknames for use with
# a CustomLog directive (see below).
#
LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\"" combined
LogFormat "%h %l %u %t \"%r\" %>s %b" common
LogFormat "%{Referer}i -> %U" referer
LogFormat "%{User-agent}i" agent
```

wKiom1PnXF3zn_uXAAELZjoWZp0008.jpg

```
#
CustomLog logs/access_log combined
```

wKioL1PnXYaR4-X8AAAhavPmhUg856.jpg

路径别名

路径别名可以实现URL路径的映射，从而所访问的页面资源不再依赖于站点的根目录。

格式：

Alias /URL/ "/path/to/somewhere/"

```
#
Alias /icons/ "/var/www/icons/"

<Directory "/var/www/icons">
    Options Indexes MultiViews FollowSymLinks
    AllowOverride None
    Order allow,deny
    Allow from all
</Directory>

#
```

wKiom1PnXKLTUgmvAABb2pKtvas777.jpg

设定默认字符集

```
AddDefaultCharset UTF-8
```

wKioL1PnXcaxZn2cAAAYPlNKers497.jpg

这里默认的是UTF-8，即8-bit Unicode Transformation Format

CGI脚本

CGI脚本路径别名

脚本的默认存放位置：**/var/www/cgi-bin/**

在浏览器中的访问格式：**http://server/cgi-bin/**

```
ScriptAlias /cgi-bin/ "/var/www/cgi-bin/"

#
# "/var/www/cgi-bin" should be changed to what
# CGI directory exists, if you have that confi
#
<Directory "/var/www/cgi-bin">
    AllowOverride None
    Options None
    Order allow,deny
    Allow from all
</Directory>
```

wKioL1PnXfqyg76PAAClqxxKF2k884.jpg

虚拟主机

虚拟主机可以分成3类

1、基于端口

2、基于IP

3、基于主机名

注意，使用虚拟的前提：取消主服务器，即注释主服务器的站点根路径指定：DocumentRoot

定义虚拟主机

NameVirtualHost IP:PORT

ServerName

DocumentRoot

ServerAlias

ErrorLog

CustomLog

配置文件语法检查：

httpd -t

service httpd configtest

配置示例：

1 2 3 4 5 6 7 8 9 10 11 12	第一个和第二个是基于主机名的，第三个是基于端口的 ServerName www.a.org DocumentRoot "/web/a" 需要声明的是这些目录都要先创建起来，并且每个根目录下 </VirtualHost> 都有创建index.html文件，里面分别写上a，b，c ServerName www.b.net DocumentRoot "/web/b" </VirtualHost> ServerName www.c.gov DocumentRoot "/web/c" </VirtualHost>

```
<VirtualHost 172.16.249.57:80>

    ServerName www.a.org
    DocumentRoot "/web/a"

</VirtualHost>

<VirtualHost 172.16.249.57:80>

    ServerName www.b.net
    DocumentRoot "/web/b"

</VirtualHost>

<VirtualHost 172.16.249.57:8080>

    ServerName www.c.gov

    DocumentRoot "/web/c"

</VirtualHost>
```

wKioL1PnXvTCn8wXAADwmSYv680459.jpg

测试：elinks

-dump: 获取到页面数据后直接退出进程；

测试结果如下

```
[root@localhost a]# elinks -dump 172.16.249.57:8080
                                    c
[root@localhost a]# elinks -dump www.a.org
                                    a
[root@localhost a]# elinks -dump www.b.net
                                    b
[root@localhost a]# elinks -dump 172.16.249.57:8080
                                    c
```

wKiom1PnXerC5ffSAACJF40yZK0347.jpg

这节就先讲到这里，后边的配置下节继续讲，谢谢！