

“十三五”普通高等教育规划教材

# Linux 基础及应用教程 (基于 CentOS 7)

第②版 梁如军 王宇昕 车亚军 等编著



提供电子教案

<http://www.cmpedu.com>



机械工业出版社  
CHINA MACHINE PRESS

## 第14章 Apache基础

主讲人：梁如军

2015-05-05

# 本章内容要点

- Web服务和HTTP协议
- Apache的特性、结构
- Apache的安装、启动和管理
- Apache的配置文件语法
- Apache的认证和授权
- Apache的虚拟主机
- Apache的日志管理
- Apache的SSL支持

# 本章学习目标

- 掌握Web组件的组成
- 熟悉Apache的特性、结构和运行机制
- 掌握Apache的安装、启动与停止
- 熟悉Apache的配置文件语法
- 掌握别名和目录容器的配置
- 掌握主机访问控制、认证和授权的配置
- 掌握基于IP和域名的虚拟主机配置
- 掌握基于SSL协议的Apache配置

# WWW和APACHE

# Web服务器简介

- WWW 是一种交互式图形界面的 Internet 服务
  - WWW（World Wide Web），也称 Web
  - 具有强大的信息连接功能
  - Internet上最热门的服务之一
  - 成为人们在网上查找、浏览信息的主要手段
- Web服务具有如下特点
  - Web是图形化的和易于导航的
  - Web是与平台无关的
  - Web是分布式的
  - Web是动态的
  - Web是交互的

- 统一资源标识符 URI
- Web 客户和 Web 服务器
- 超文本传输协议 HTTP
- Web缓存和Web代理
- Cookie 和Session机制
- Web内容的构建组件

# Web组件——URI

协议名称://机器地址:端口号/路径名/文件名

- 协议名称 — 所使用的访问协议。如：http、ftp 等
- 机器地址 — 数据所在的机器，IP地址/域名
- 端口号 — 请求数据的数据源端口（可省略）
- 路径名 — 数据所在的相对路径
- 文件名 — 请求数据的文件名

`http://www.centos.org`

`http://192.168.0.191`

`http://192.168.0.191:8080`

`http://woodpecker.org.cn/diveintopython3/whats-new.html`

`ftp://192.168.0.191`

`ftp://192.168.0.191:8021`

## ■ Web 服务器的职责

- ❑ 默认监听TCP/IP的80端口
- ❑ 接受Web客户请求
- ❑ 检查请求的合法性，包括安全性屏蔽
- ❑ 针对请求获取并制作和处理数据
- ❑ 把处理后的信息发送给提出请求的客户机

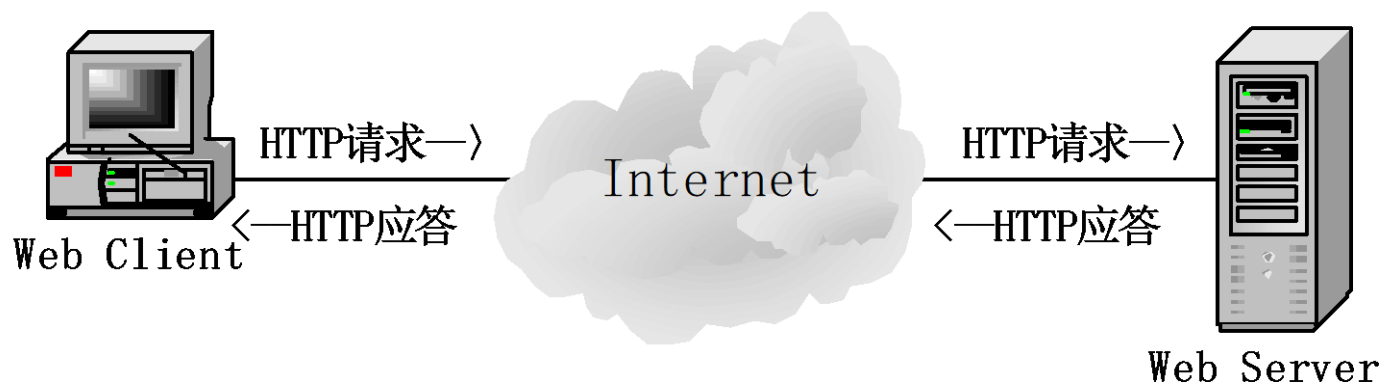
## ■ Web 浏览器的职责

- ❑ 生成一个 Web 请求（通常在单击某个链接点时启动）
- ❑ 通过网络将 Web 请求发送给某个 Web 服务器
- ❑ 解释服务器传来的 Web 文档，并把结果显示在屏幕上



# Web客户与服务器通信过程

- 每取一个网页建立一次连接，读完后马上断开；当需要另一个网页时重新连接，周而复始。



# Web组件——HTTP协议（1）



- 超文本传输协议（Hyper Text Transfer Protocol）是在 Web 上传输资源最常用的方式
- HTTP 规定了客户机和服务器等 Web 组件 相互交换信息的格式和含义
- HTTP 协议的特点
  - URI 资源识别
  - 请求-响应方式
  - 无状态性
  - 携带元数据

## ■ HTTP协议的版本

### □ HTTP/1.0

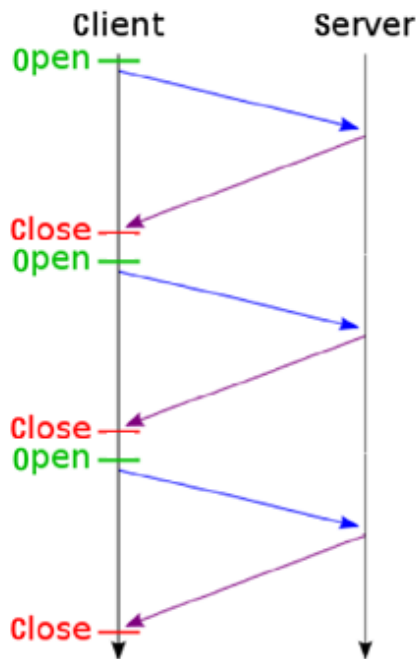
### □ HTTP/1.1 ——当前广泛使用的协议标准。

- RFC 7230, HTTP/1.1: Message Syntax and Routing
- RFC 7231, HTTP/1.1: Semantics and Content
- RFC 7232, HTTP/1.1: Conditional Requests
- RFC 7233, HTTP/1.1: Range Requests
- RFC 7234, HTTP/1.1: Caching
- RFC 7235, HTTP/1.1: Authentication

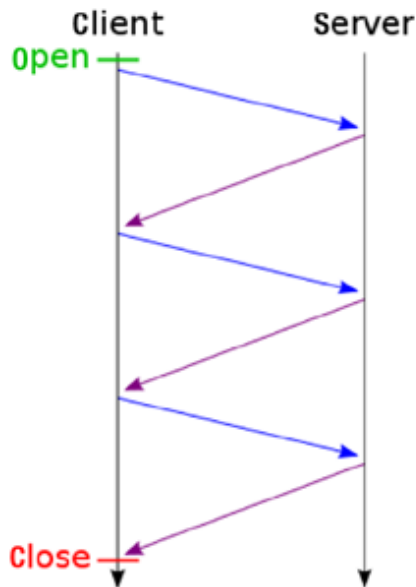
### □ HTTP/2

# Web组件——HTTP协议 (3)

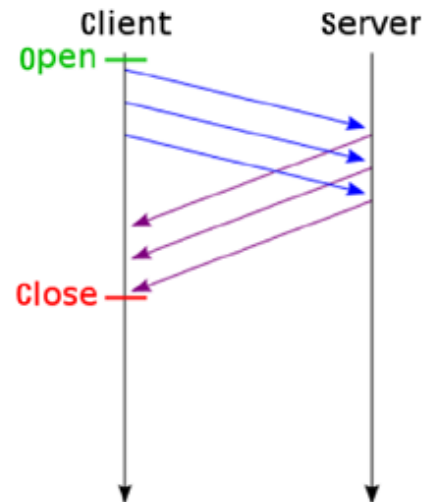
## ■ HTTP 的连接方式



(a) 传统方式



(b) 持久连接方式



(c) 管线化方式

## ■ HTTP的协议头

□ （HTTP header）是HTTP会话请求和响应的一部分，用于客户端和服务端进行HTTP协议协商。

■ **请求头**（Request Header Fields）

（<https://tools.ietf.org/html/rfc7231#section-5>）

■ **响应头**（Response Header Fields）

（<https://tools.ietf.org/html/rfc7231#section-7>）

□ 使用curl命令获取HTTP的协议头

■ `curl -s -I -v www.centos.com | egrep '^>|^<'`

## ■ HTTP的请求方法

- ❑ HEAD
- ❑ **GET**
- ❑ **POST**
- ❑ **PUT**
- ❑ DELETE
- ❑ CONNECT
- ❑ OPTIONS
- ❑ TRACE

## ■ HTTP的响应代码

### □ 信息 1xx

- 表明服务端接收了客户端请求，客户端继续发送请求

### □ 成功 2xx

- 客户端发送的请求被服务端成功接收并成功进行了处理

### □ 重定向 3xx

- 服务端给客户端返回用于重定向的信息

### □ 客户端错误 4xx

- 客户端的请求有非法内容

### □ 服务器错误 5xx

- 服务端未能正常处理客户端的请求而出现意外错误

## Web缓存和Web代理

### ■ Web缓存

- HTTP协议定义了客户端缓存机制。
- 架设Web缓存服务器和内容分发网络（Content Delivery Network, CDN）可以加快客户端访问。

### ■ Web代理

- 同时扮演着客户和服务器的双重身份
  - 对于 Web 客户来说是服务器
  - 对于 Web 服务器来说是客户
- 还可以过滤不希望的 Web 请求，实现高速缓存等



## Cookie 和Session机制

- HTTP 是一个无状态协议，因此当Web服务器将Web客户请求的响应发送出去后，服务器便不必保存任何信息了。
- Web服务器可以指示Web客户以存储 Cookie 的方式在一系列请求和响应之间维持状态，而服务器端则采用Session机制保持状态。

## Web内容的构建组件

- 使用HTML/XHTML、CSS、Javascript构建静态Web页面
- 使用CGI、PHP、Python、Ruby、Java Servlet、Node.js等技术构建动态Web应用
- 使用各种数据发布格式及语言（XML、YAML、JSON、RSS/Atom）交换数据

- 超文本标记语言（**HTML**）是为纯文本格式的超文本文档提供了一种标准的表述方式
- **HTML** 是由标准通用标记语言（**SGML**）演化而来的
- **HTML** 可以使用标记格式化文本、引用图片或嵌入其他文档的超链接
- 有关 **HTML** 和 **XHTML** 的更多信息，请参见：  
<http://www.w3.org/MarkUp/>

# Linux下常用的Web服务器



- **Apache**

- <http://httpd.apache.org>

- **Nginx**

- <http://nginx.org/>

- **Lighttpd**

- <http://www.lighttpd.net/>

- Apache 是一个知名的开源Web服务器
  - 由NCSA httpd 1.3 经过较为完整的代码重写
  - 名称 Apache意为 A Patchy Server，即它是基于现存的代码和一系列的Patch文件
  - Apache软件基金会（ASF，Apache Software Foundation）<http://www.apache.org> 维护
  - 2012.02 —— Apache 2.4版发行
- 在功能、效率、扩展及速度方面居于领先的地位
  - 根据[Netcraft](#)提供的最新调查资料，Apache Web服务器是使用比例最高的Web服务器

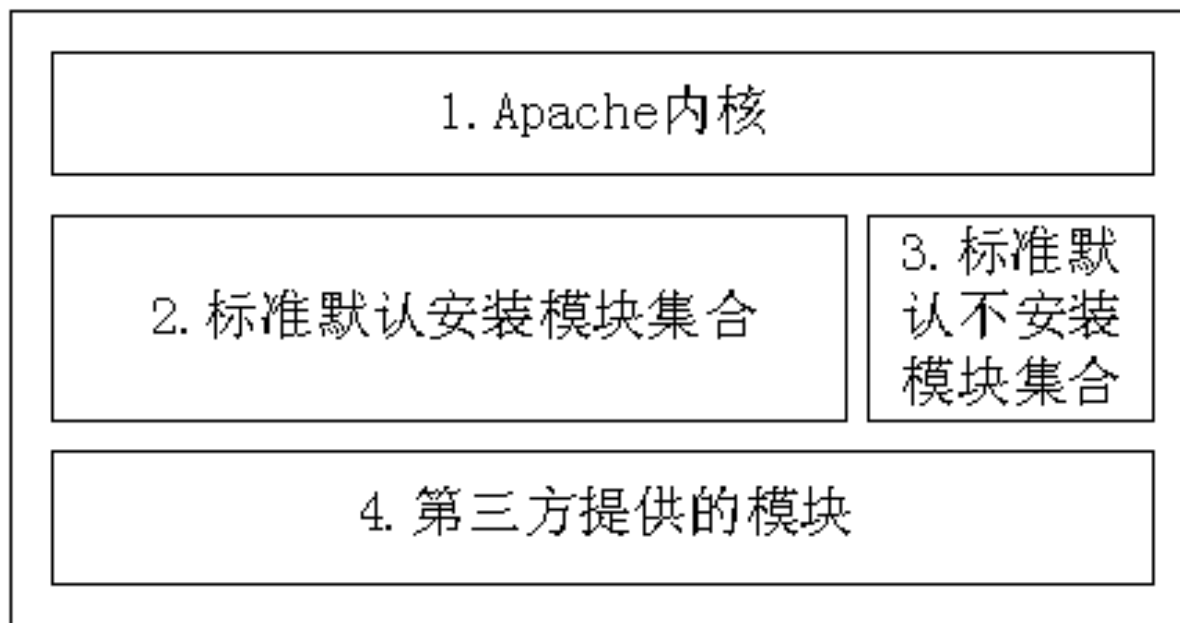
# Apache 的特性

- 开放源代码、跨平台应用。
- 模块化设计、运行稳定、良好的安全性。
- 实现了动态共享对象(DSO)，允许在运行时动态装载功能模块。
- 支持最新的HTTP 1.1协议。
- 支持虚拟主机、支持HTTP认证、集成了代理服务、支持安全Socket层（SSL）。
- 使用简单而强有力的基于文本的配置文件、具有可定制的服务日志。
- 支持通用网关接口CGI、FastCGI、服务器端包含命令（SSI）。
- 支持PHP/Perl/Python/Ruby/Java Servlets等脚本编程语言。
- 支持第三方软件开发商提供的大量功能模块。

参考 [http://httpd.apache.org/docs/2.4/new\\_features\\_2\\_4.html](http://httpd.apache.org/docs/2.4/new_features_2_4.html)  
查看Apache 2.4版的新特性

# Apache 的结构

- Apache由内核、标准模块和第三方提供的模块三个层次组成
- 模块信息：<http://modules.apache.org>



# Apache 的运行模式

- Apache 2.X使用新的多处理模块（Multi-Processing Module, **MPM**）
  - 在服务器处理多个请求时控制Apache的运行方式
- Apache中的3种运行模式
  - 多进程模型
    - 预派生（**Prefork**）**MPM**
  - 多进程多线程混合模型
    - 工作者（**Worker**）**MPM**
    - 事件（**Event**）**MPM**



# CENTOS下的APACHE

# Apache服务概览

- 软件包: `httpd`, `httpd-devel`, `httpd-manual`
- 服务类型: 由Systemd启动的守护进程
- 配置单元: `/usr/lib/systemd/system/httpd.service`
- 守护进程: `/usr/sbin/httpd`
- 端口: `80(http)`, `443(https)`
- 配置: `/etc/httpd/`
- Web文档: `/var/www/`
- 相关软件包: `mod_ssl`

# Apache的安装和启动

## ■ 安装

```
# yum install httpd httpd-manual
```

## ■ 管理httpd服务

```
# systemctl {start|stop|status|restart|reload} httpd
```

```
# systemctl {enable|disable} httpd
```

## ■ 检查配置文件的正确性

```
# apachectl -t
```

```
# httpd -t
```

# Apache的相关文件

## ■ 管理工具

- ❑ **/usr/sbin/apachectl**: Apache HTTP 服务器控制接口
- ❑ **/usr/bin/ab**: Apache HTTP 服务器性能测试工具
- ❑ **/usr/bin/logresolve**: 将 Apache 日志文件中的 IP 地址解析为主机名
- ❑ **/usr/sbin/rotatelogs**: 滚动 Apache 日志而无须终止服务器

# 查看Apache 的相关信息

- 显示Apache的编译参数  
**# apachectl -V 或 httpd -V**
- 查看已经被编译的模块  
**# apachectl -l 或 httpd -l**
- 列出所有模块，包括mod\_so加载的DSO  
**# apachectl -M 或 httpd -M**

# CentOS下Apache的默认配置



- 服务器的根目录: `/etc/httpd`
- 运行Apache的用户: `apache`
- 运行Apache的组: `apache`
- 监听端口: `80`
- 模块存放路径: `/usr/lib/httpd/modules`
- prefork MPM 运行方式的参数:
  - `StartServers 8`
  - `MinSpareServers 5`
  - `MaxSpareServers 20`
  - `MaxClients 150`

# CentOS下Apache的默认配置 (续)



## ■ 默认的Web文档

- `/var/www/html/`: 根文档目录
- `/var/www/cgi-bin/`: CGI程序目录
- `/var/www/error/`: 默认的错误文档目录
- `/var/www/icons/`: 与icons相关的图片目录

## ■ 默认的日志文件

- `/var/log/httpd/access_log`: 访问日志
- `/var/log/httpd/error_log`: 错误日志

# CentOS 下 Apache 的默认配置

## (续2)



### ■ Apache 的配置文件

- **/etc/httpd/conf/httpd.conf**: **Apache** 的主配置文件
- **/etc/httpd/conf.d/\*.conf**: 被主配置文件包含的配置文件
- **/etc/httpd/conf/magic**: mod\_mime\_magic 模块使用的 Magic 数据, 无需配置
- **/etc/logrotate.d/httpd**: Apache 的日志滚动配置文件
- **/etc/sysconfig/httpd**: httpd 守护进程的启动配置文件



# APACHE配置基础

# Apache配置文件种类

- 主配置文件
  - **/etc/httpd/conf/httpd.conf**
- 被主配置文件包含的配置文件
  - **/etc/httpd/conf.d/\*.conf**
  - 可以用**Include/IncludeOptional**指令定义被包含的配置文件
- 基于目录的配置文件
  - 使用分布在网站目录树中的特殊文件来进行分散配置
  - 这些特殊的文件默认为 **.htaccess**
  - 也可以用 **AccessFileName** 指令来改变它的名字

# 配置文件的基本语法

- 每一行包含一个指令，在行尾使用反斜杠“\”可以表示续行
- 配置文件中的指令不区分大小写，但是指令的参数 (argument) 通常区分大小写
- 以 “#” 开头的行被视为注解并在读取时被忽略。注解不能出现在指令的后边
- 空白行和指令前的空白字符将在读取时被忽略，因此可以采用缩进以保持配置层次的清晰

# Apache 的两种编译方式

## ■ 静态编译

- ❑ 将核心模块和所需要的模块一次性编译
- ❑ 优点：运行速度快
- ❑ 缺点：要增加或删除模块必须重新编译整个 Apache

## ■ 动态编译

- ❑ 只编译核心模块和 **DSO**（动态共享对象）模块 — **mod\_so**
- ❑ 优点：各模块可以独立编译，并可随时用 **LoadModule** 指令加载，用于特定模块的指令可以用 **<IfModule>** 指令包含起来，使之有条件地生效。
- ❑ 缺点：运行速度稍慢

# Apache 的模块

- 查看Apache的编译参数

**# httpd -l**

- RHEL/CentOS 中的Apache是动态编译的
    - mod\_so.c 模块使 Apache 支持 Dynamic Shared Objects (DSO)
    - 可在不重新编译 Apache 的情况下使用 APache eXtenSion (apxs) 编译 Apache 的其他模块
  - 所有动态编译的模块在使用时需要使用 **LoadModule** 指令加载
- LoadModule rewrite\_module modules/mod\_rewrite.so**

# 获得Apache配置的帮助

- 查看配置文件的 MAN 手册  
**# man httpd.conf**
- 查看本机安装的HTML手册  
**# w3m http://localhost/manual/**
- 输出Apache 的指令列表  
**# httpd -L 或 apachectl -L**

# Apache 的基本配置指令

- Apache的服务器标识指令
- Apache的文件定位指令
- Apache MPM 的相关指令
- Apache 常用的全局配置指令

# Apache的服务器标识指令

- **ServerName:** 服务器用于辨识自己的主机名和端口号
- **ServerAdmin:** 服务器返回给客户端的错误信息中包含的管理员邮件地址
- **ServerSignature:** 配置服务器生成页面的页脚的信息
- **ServerTokens:** 控制了服务器回应给客户端的”Server:“应答头是否包含关于服务器操作系统类型和编译进的模块描述信息
- **UseCanonicalName:** 决定 Apache 如何构造 URL 中 SERVER\_NAME 和 SERVER\_PORT 的指令



# Apache的文件定位指令

- **ServerRoot**: 指定服务器安装的基础目录
- **DocumentRoot**: 组成网络上可见的主文档树的根目录
- **ErrorLog**: 存放错误日志的位置
- **CustomLog**: 访问日志文件的位置
- **LockFile**: Apache 使用的锁文件的位置
- **PidFile**: 设置服务器用于记录父进程(监控进程) PID 的文件

# Apache Profork MPM 的相关指令

- **StartServers**: 启动时服务器启动的进程数
- **MinSpareServers**: 保有的备用进程的最小数目
- **MaxSpareServers**: 保有的备用进程的最大数目
- **MaxClients**: 服务器允许启动的最大进程数
- **MaxRequestsPerChild**: 一个服务进程允许的最大请求数

# Apache 常用的全局配置指令



- **Listen**: 指定监听的IP地址、端口号，默认为80
- **User**: 指定运行Apache服务的用户名，默认为apache
- **Group**: 指定运行Apache服务的组名，默认为apache
- **LogLevel**: 指定错误日志的记录级别
- **Timeout**: 指定网络连接超时，默认为120（单位为秒）
- **KeepAlive**: 指定是否保持连接，默认为Off
- **KeepAliveTimeout**: 保持连接状态时的超时时间，默认为15（单位为秒）
- **MaxKeepAliveRequests**: 保持连接状态时，每次连接最多请求文件数，默认为100
- **DirectoryIndex**: 指定默认的索引页文件，默认为index.html  
index.html.var
- **IndexOptions**: 指定服务器所生成的列表页面的输出选项

# Apache 的配置容器

## ■ **<Directory></Directory>**

- 用于对指定的目录（可使用Shell通配符）实施额外的配置

## ■ **<Files></Files>**

- 用于对指定的文件（可使用Shell通配符）实施额外的配置

## ■ **<Location></Location>**

- 用于对指定的 URL（可使用Shell通配符）实施额外的配置

## ■ **<Limit></Limit>**

- 用于对指定的HTTP方法实施额外的配置

## ■ **<LimitExcept></LimitExcept>**

- 用于对指定的HTTP方法之外的方法实施额外的配置

# Apache 的配置容器（续）

- 功能与Directory、Files、Location容器相同，在描述目录、文件URL时可以使用正则表达式
  - **<DirectoryMatch></DirectoryMatch>**
  - **<FilesMatch></FilesMatch>**
  - **<LocationMatch></LocationMatch>**
- 虚拟主机容器
  - **<VirtualHost></VirtualHost>**
  - 用于对虚拟主机实施额外的配置（一台计算机支持多个站点的能力）

# Apache主配置文件的组成和配置指令的作用域

## ■ Apache主配置文件的组成

- 全局环境配置
- 主服务器配置
- 虚拟主机配置

## ■ 配置指令的作用域

- 配置指令作用范围可以是全局或只能在容器
    - **server config、virtual host、directory、.htaccess**
  - 查看指令的作用域
    - Apache 手册中指令的作用域（Context）项
- # httpd -L 或 apachectl -L**

# APACHE的基本配置

# 主机访问控制简介

- Apache可以根据访问者的 IP 地址或域名来决定是否为之提供资源，也称强验证
- 访问控制的功能由 **mod\_authz\_core**和 **mod\_authz\_host** 模块提供
- 使用Require指令实现访问控制



# 访问控制的指令的作用范围



- 可用在<Location>、<Directory>、<Files> 和 <Limit>容器中
- 既可以用在主配置文件或其包含的配置文件中，也可以用在.htaccess配置文件中
- 既可以放在“主配置”部分用于控制主服务器；也可以放在<VirtualHost>容器中用于控制虚拟主机

- 允许所有主机访问
  - **Require all granted**
- 拒绝所有主机访问
  - **Require all denied**
- 仅允许本地主机访问
  - **Require local**
- 允许或[禁止]指定的主机或域访问
  - **Require [not] host <主机名或域名列表>**
- 允许或[禁止]指定IP地址的访问
  - **Require [not] ip <IP地址或网段列表>**

# 访问控制举例

Require ip 10.1.2.3

Require ip 10 172.20 192.168.2

Require ip 10.1.0.0/255.255.0.0

Require ip 10.1.0.0/16 192.168.1.0/24

Require host server1.example.org

Require host example.org abc.net

Require host .net .example.edu

# 访问控制举例 续

```
<RequireAll>
```

```
    Require all granted
```

```
    Require not ip 10.252.46.165
```

```
</RequireAll>
```

```
<RequireAll>
```

```
    Require ip 10.252.46.0/24
```

```
    Require not ip 10.252.46.165
```

```
</RequireAll>
```

- 别名可以将文档根目录（`/var/www/html`）以外的内容加入站点，也称虚拟目录
- Alias 指令
  - ❑ **Alias /URL-path "/path/to/other/directory/"**
  - ❑ 将以 /URL-path 开头的 URL 映射到 /path/to/other/directory 中的文件
- Alias 举例
  - ❑ **Alias /manual "/var/www/manual"**
  - ❑ **Alias /ks /kickstart**

# 容器选项配置（Options）

- **Options** 指令用于控制当前容器中可以使用的服务器特性
- 可以出现在
  - 主配置文件或.htaccess配置文件中
  - <Directory>、<Location>容器中
- **Options** 指令格式

**Options [+|-]Option1 [+|-]Option2 .....**

  - 选项之前添加加号（+）表示添加此特性
  - 选项之前添加减号（-）表示去掉此特性

# Options指令的常用选项

- **All**: 除MultiViews之外的所有特性。默认设置
- **None**: 将不启用任何额外特性
- **ExecCGI**: 允许使用mod\_cgi执行CGI脚本
- **FollowSymLinks**: 服务器允许在此目录中使用符号连接
- **Indexes**: 若一个映射到目录的URL被请求，而此目录中又没有DirectoryIndex指定的文件（例如index.html），则服务器会返回由mod\_autoindex模块生成的一个格式化后的目录列表
- **MultiViews**: 允许使用mod\_negotiation提供内容协商的“多重视图”

- 用于配置 `mod_autoindex` 模块生成目录列表的显示特性
- IndexOptions 指令的常用选项
  - FancyIndexing
    - 对每种类型的文件前加上一个小图标以示区别
  - VersionSort
    - 对同一个软件的多个版本进行排序
  - NameWidth=\*
    - 文件名子段自动适应当前目录下最长文件名
  - FoldersFirst
    - 让目录列在前面（类似于资源管理器）



# 主机访问控制和别名的配置举例

## ■ 使用别名配置对yum仓库和Kickstart的访问

```
Alias /mirrors /var/ftp/mirrors
<Directory /var/ftp/mirrors>
    Options Indexes FollowSymlinks
    IndexOptions +DescriptionWidth=* +FoldersFirst
    Require local
    Require ip 192.168.0.0/24 192.168.85.0/24 192.168.17.0/24
</Directory>
Alias /ks /kickstart
<Directory /kickstart>
    Options Indexes FollowSymlinks
    IndexOptions +DescriptionWidth=* +FoldersFirst

    Require local
    Require ip 192.168.0.0/24 192.168.85.0/24 192.168.17.0/24
</Directory>
```

# 配置每个用户的Web站点

- 使拥有用户账号的每个用户都能够架设自己单独的Web站点
- 使用 `mod_userdir` 模块，可以用如下的URL
  - `http://IPorFQDN/~username`
  - 访问系统用户username的Web站点
- 使用**UserDir**指令指定用户站点的文档根目录
- 配置步骤
  - 修改配置文件（启用`mod_userdir` 模块并配置每个用户Web站点目录的访问控制）
  - 设置`$HOME`对其他目录的可执行权限

# 基于目录的配置文件

- 可以使用`.htaccess`文件改变主配置文件中的配置，但是它只能设置对目录的访问控制，这个目录就是`.htaccess`文件存放的目录
- 使用`.htaccess`文件的场合
  - 当内容提供者需要针对目录改变服务器的配置而对服务器系统没有`root`权限时
  - 管理员不愿意频繁修改配置并重启服务，因为修改`.htaccess`文件后**立即生效**，无需重新启动服务

# 使用.htaccess文件的方法

- 首先在主配置文件中启用并控制对.htaccess文件的使用
  - **AllowOverride all|none**
- 然后在需要覆盖主配置文件的目录下生成.htaccess文件
- 使用.htaccess文件举例

# 认证和授权

- **Apache**的认证和授权（基于用户的访问控制）
  - 认证和授权是 **Apache** 允许指定用户使用用户名和口令访问特定资源的一种方式
  - **认证**（Authentication）是指任何识别用户身份的过程
  - **授权**（Authorization）是允许特定用户访问特定区域或信息的过程
  - 认证和授权也称弱验证
  - 认证和授权的配置指令既可以出现在主（或其包含的）配置文件的<Directory>或<Location>容器中，也可以出现在./htaccess文件中

# 两种认证

	基本认证	摘要认证
Apache 模块	mod_auth_basic	mod_auth_digest
证书管理程序	htpasswd	htdigest
浏览器支持	所有浏览器均支持	绝大多数浏览器均支持
特点	可用于任何认证领域	只用于指定的认证领域
	在网络中传输 Base64编码的明文 口令，不安全	在网络中只传输质询码 和摘要信息，不传输口 令，更安全

# 认证和授权的证书存储和相关模块

	认证	授权
纯文本文件	<b>mod_authn_file</b>	<b>mod_authz_user</b> <b>mod_authz_groupfile</b>
DBM 数据库	<b>mod_authn_dbm</b>	<b>mod_authz_dbm</b>
关系数据库	<b>mod_authn_dbd</b>	
LDAP	<b>mod_authnz_ldap</b>	

- 通常使用纯文本文件存储认证口令证书
- 为了加快检索可以使用DBM数据库
- 为了与其他应用集成可以使用关系数据库或LDAP存储



# 认证相关指令

- 定义受保护领域的名称
  - **AuthName** <认证领域名称>
- 定义使用的认证方式
  - **AuthType Basic**或**Digest**
- 指定认证组文件的位置
  - **AuthGroupFile** <文件名>
- 指定认证口令文件的位置
  - **AuthUserFile** <文件名>
- 指定摘要需要认证的 URI（仅用于摘要认证）
  - **AuthDigestDomain URI [URI] ...**

- 当使用认证指令配置了认证之后，还需要使用 **Require** 指令为指定的用户或组进行授权
- **Require** 指令的三种使用格式
  - 授权给指定的一个或多个用户  
**Require user** 用户名 [用户名] .....
  - 授权给指定的一个或多个组  
**Require group** 组名 [组名] .....
  - 授权给认证口令文件中的所有用户  
**Require valid-user**

# 管理基本认证的口令文件

## ——htpasswd 命令

- 添加一个认证用户的同时创建认证口令文件  
**# htpasswd -cm <认证口令文件名> <用户名>**
- 向现存的口令文件中添加用户或修改已存在的用户的口令  
**# htpasswd -m <认证口令文件名> <用户名>**
- 从认证口令文件中删除用户及其口令  
**# htpasswd -D <认证口令文件名> <用户名>**

-m 参数可以生成MD5算法的加密口令（CentOS7中为默认参数）  
-b 参数用于在命令行上直接指定用户名及其口令，而非交互模式

# 管理摘要认证的口令文件

## ——htdigest 命令

- 添加一个认证用户的同时创建认证口令文件  
**# htdigest -c** <认证口令文件名> <认证领域> <用户名>
- 向现存的口令文件中添加用户或修改已存在的用户的口令  
**# htdigest** <认证口令文件名> <认证领域> <用户名>

没有提供从认证口令文件中删除指定用户及其口令的功能，需要直接编辑认证口令文件

# 管理认证组文件

- Apache 支持认证组文件
- Apache 没有提供创建认证组文件的命令
- 认证组文件只是一个文本文件，可以使用任何文本编辑器创建并修改
- 认证组文件中每一行的格式

组名:用户名 用户名 .....

在认证组文件中指定的用户名，必须先使用 **htpasswd** 或 **htdigest** 命令添加到认证口令文件中

# 认证证书的权限

- 认证证书包扩认证口令文件和/或认证组文件
- 基于安全因素的考虑
  - 认证证书不应该存放在 **DocumentRoot** 指令指定的目录或其子目录下
  - 建议存放在 **/etc/httpd/passwd** 子目录或与虚拟主机根文档目录同级别的**conf**或**passwd**子目录下
- 确保执行**Apache**守护进程的用户（**CentOS**默认为 **apache**）能读取认证证书
  - 确保 **apache** 用户能进入存放认证证书的目录
  - 确保 **apache** 用户能读取认证证书文件

# 认证和授权配置举例

- 在主配置文件中配置认证和授权
- 在.htaccess文件中配置认证和授权

# 虚拟主机



- 在一台**Web**服务器上，通过多个独立的**IP**地址、域名或端口号提供不同的**Web**站点
  - 基于**IP**地址的虚拟主机
    - 每个网站拥有不同的 **IP** 地址
    - 通过访问服务器上不同的**IP**地址访问不同的网站
  - 基于域名的虚拟主机
    - 所有的虚拟主机可以共享同一个**IP**地址
    - 使用不同的域名来访问不同的网站
  - 基于端口的虚拟主机
    - 所有的虚拟主机可以共享同一个**IP**地址
    - 各虚拟主机之间通过不同的端口号进行区分

# 虚拟主机注意事项

- 可以在一台主机上混合配置不同方式的虚拟主机
- 在一台主机上配置基于**IP**的虚拟主机时
  - 既可以安装配置多个网络接口
  - 也可以为一个网络接口绑定多个 **IP** 地址
- 无论哪一种虚拟主机，都应该配置域名解析
  - 只有基于**IP**的虚拟主机可以使用**IP**地址和域名访问
  - 基于域名的虚拟主机只能使用域名访问

# 虚拟主机配置指令

- VirtualHost 容器内使用的指令
  - **ServerName**: 用于指定虚拟主机的名称和端口号
  - **ServerAdmin**: 用于指定虚拟主机的管理员E-mail地址
  - **DocumentRoot**: 用于指定虚拟主机的根文档目录
  - **ErrorLog**: 用于指定虚拟主机的错误日志存放路径
  - **CustomLog**: 用于指定虚拟主机的访问日志存放路径
  - 使用 <Directory>、<Location> 等容器设置访问控制等
- VirtualHost 容器之外使用的指令
  - **NameVirtualHost** : 用于为一个和多个**基于域名的虚拟主机**指定一个IP地址和端口
  - 在 **Apache 2.4** 版中, 可省略此指令

# 主服务器配置 与虚拟主机配置的关系

## ■ 覆盖性

- ❑ **VirtualHost** 容器中的指令会覆盖主服务器范围内同名的配置指令
- ❑ 主服务器（**Main Server**）范围内的配置指令（在所有 **<VirtualHost>** 容器之外的指令，包括主配置文件使用**Include**包含的配置文件中的指令）仅在它们没有被**VirtualHost** 容器的配置覆盖时才起作用

## ■ 继承性

- ❑ 每个虚拟主机都会从主服务器配置继承相关的配置
- ❑ 例如：虚拟主机继承主服务器的**DirectoryIndex**

# 使用单独的 虚拟主机配置文件

- 配置虚拟主机时可以在主配置文件中进行
  - 为了方便维护虚拟主机的配置，通常为某个虚拟主机或某组虚拟主机使用单独的配置文件
- 修改主配置文件 **`/etc/httpd/conf/httpd.conf`**  
**`Include vhosts.d/*.conf`**
  - 创建存放虚拟主机配置文件的目录  
**`# mkdir /etc/httpd/vhosts.d`**

# 配置基于IP的虚拟主机

- 基于IP的虚拟主机的配置步骤
  - 在一台主机上配置多个IP地址并配置域名解析
  - 创建文档目录和测试主页
  - 修改配置文件添加虚拟主机配置
  - 重新启动 **Apache**，分别使用IP和域名进行访问测试
- 基于IP的虚拟主机的配置举例

参考教材中的操作步骤

# 配置基于域名的虚拟主机

- 基于域名的虚拟主机的配置步骤
  - 配置虚拟主机的域名解析
  - 创建文档目录和测试主页
  - 修改配置文件添加虚拟主机配置
  - 重新启动 **Apache**，使用域名进行访问测试
- 基于域名的虚拟主机的配置举例

参考教材中的操作步骤

# 日志管理



# Apache 的日志

- 日志的种类
  - 错误日志
  - 访问日志
- Apache 默认的错误日志配置

**ErrorLog** logs/error\_log  
**LogLevel** warn

- Apache 默认的访问日志配置

**LogFormat** "%h %l %u %t \"%r\" %>s %b  
\"%{Referer}i\" \"%{User-Agent}i\" combined  
**CustomLog** logs/access\_log combined

# Apache的日志滚动

- Apache的日志滚动的必要性
  - 一个访问频繁的Web站点的日志会迅速增长
  - 定期清理以免造成磁盘空间的不必要的浪费
  - 查看日志时打开小文件的速度比大文件的速度要快
- Apache的日志滚动方法
  - RHEL/CentOS的默认配置
    - logrotate 和 crond 实现日志滚动
  - 其他工具
    - 使用 Apache 自带的 rotatelogs
    - 使用 cronolog (<http://cronolog.org/>)

- 若在虚拟主机的<VirtualHost>容器之内没有配置日志指令，则每个虚拟主机将继承使用主配置文件中<VirtualHost>容器之外的日志配置。
- 分离虚拟主机日志的方法
  - 在<VirtualHost>容器之内使用 **ErrorLog** 和 **CustomLog** 语句指定本虚拟主机单独使用的日志文件
  - 使用主配置文件将所有虚拟主机的日志记录到一个文件，然后可以使用分离脚本 **split-logfile** 将日志文件的内容按不同的虚拟主机拆分为多个文件

# APACHE+MOD\_SSL

- Apache HTTP 服务器模块 mod\_ssl
  - 提供了与 OpenSSL 的接口，它使用安全套接字层和传输层安全协议提供了强加密。
  - 此模块基于 Ralf S. Engelschall 的 mod\_ssl (<http://www.modssl.org/>) 项目
  - 安装
    - **# yum -y install mod\_ssl**
  - 配置文件
    - **/etc/httpd/conf.d/ssl.conf**



# mod\_ssl的默认配置文件

## ■ /etc/httpd/conf.d/ssl.conf

LoadModule ssl\_module modules/mod\_ssl.so

Listen 443

<VirtualHost \_default\_:443>

SSLEngine on

SSLCertificateFile /etc/pki/tls/certs/localhost.crt

SSLCertificateKeyFile

/etc/pki/tls/private/localhost.key

</VirtualHost>

# SSL相关的Apache配置指令

## ■ SSLEngine

- 开启或关闭SSL/TLS协议引擎

## ■ SSLProtocol

- 允许使用哪些版本的SSL/TLS协议

## ■ SSLCipherSuite

- 告诉客户端允许使用哪些加密算法

## ■ SSLCertificateFile

- 指定服务器证书文件

## ■ SSLCertificateKeyFile

- 指定服务器私钥文件

# OpenSSL的加密算法套件

- OpenSSL 的加密算法套件由4个属性组成：
  - 密钥交换算法：
    - RSA或Diffie-Hellman算法的各种变种
  - 认证算法：
    - RSA, Diffie-Hellman, DSS 或 none
  - 加密算法：
    - AES, DES, Triple-DES, RC4, RC2, IDEA 或 none
  - 摘要算法：
    - MD5, SHA 或 SHA1



## ■ SSLCipherSuite

- 用于在SSL握手过程中进行加密算法协商时告诉客户端允许使用哪些加密算法。
- 指令的值是一个冒号分隔的OpenSSL加密算法套件字符串，默认值为  
`ALL:!ADH:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP`

## ■ 增删算法的语法

- [没有标记]: 向列表中增加一个算法套件
- +: 在列表中的相应的位置增加一个算法套件
- -: 从列表中临时删除相应的算法套件
- !: 从列表中永久删除相应的算法套件

# SSLCipherSuite（续）

## ■ CentOS的ssl.conf中的值为

SSLCipherSuite **ALL:!ADH:!EXPORT:!SSLv2:RC4+RSA:+HIGH:+MEDIUM:+LOW**

## ■ 表示

- 首先指定所有的加密算法套件 **【ALL】**
- 然后依次删除
  - 所有使用匿名Diffie-Hellman密钥交换（ADH）的算法 **【!ADH】**
  - 所有美国限制出口的算法 **【!EXPORT】**
  - 所有SSLv2协议的算法 **【!SSLv2】**
- 然后添加使用RC4+RSA的算法 **【RC4+RSA】**
- 最后顺序添加高中低强度的算法 **【+HIGH:+MEDIUM:+LOW】**

## ■ 查看算法套件支持的算法组合

**\$ openssl ciphers -v 'ALL:!ADH:!EXPORT:!SSLv2:RC4+RSA:+HIGH:+MEDIUM:+LOW'**

### □ 输出列解释

- 1. 套件名称
- 2. 支持的最小协议版本
- 3. 密钥交换算法（**Kx**）
- 4. 认证算法（**Au**）
- 5. 加密算法及长度（**Enc**）
- 6. 完整性校验算法（**Mac**）
- 7. 出口算法套件指示器

# 基于域名的SSL虚拟主机

- 使用支持X509 SAN扩展的多域名证书
  - 一个证书中支持多个域名的SSL虚拟主机
  - SAN（Subject Alternative Name）【RFC 4366】
  - 使用 subjectAltName指定多个域名
- 使用支持TLS的SNI扩展的Apache+mod\_ssl
  - SNI（Server Name Indication）【RFC 4366】，在SSL/TLS握手之初先向服务器端指出所要访问的主机域名
  - 可以在同一IP的443端口上创建多个 SSL/TLS 的基于域名的虚拟主机，且**每个虚拟主机可以分别使用各自的证书。**

# 配置SSL 虚拟主机

- 生成或获取密钥和证书
- 修改Apache配置文件

```
<VirtualHost *:443>
.....
SSLEngine on
SSLCipherSuite HIGH:MEDIUM:!aNULL:!SSLv2
SSLProtocol all -SSLv2

SSLCertificateFile           /etc/pki/tls/certs/olabs.net.crt
SSLCertificateKeyFile        /etc/pki/tls/private/olabs.net.key
.....
</VirtualHost>
```

# 本章思考题

- 什么是Apache？简述其特点。
- 如何配置CentOS默认的Apache以提高安全性？
- 如何设置基于主机的访问控制？
- 简述认证和授权指令的使用。
- Apache有哪几种日志？Apache的日志指令有哪些？
- 什么是虚拟主机？Apache支持几种类型的虚拟主机？
- 如何配置SSL/TLS的Apache基于域名的虚拟主机？
- 什么是SAN证书？TLS的SNI扩展的作用？
- 简述Apache常用的安全模块及其作用。

- 使用符号链接和别名管理站点。
- 配置访问控制、认证和授权。
- 配置基于IP和基于域名的虚拟主机。
- 查看Apache的日志文件。
- 配置虚拟主机的分离日志及其日志滚动。
- 配置SSL/TLS的Apache基于域名的虚拟主机。
- 安装配置mod-evasive和mod-qos模块。

## ■ 查看Apache手册

- 了解各个模块的作用。
- 学习常用指令的配置语法。
- 学习使用.htaccess文件分割配置任务。
- 学习 mod\_rewrite 模块的配置和使用。
- 学习使用mod\_rewrite模块配置虚拟主机的方法。
- 学习配置WebDAV实现站点内容上传。
- 学习每用户个人站点的配置



# 进一步学习（续）

- 学习配置httpd-itk的MPM模式运行。
- 学习配置SSL双向认证（包括客户端）的Apache虚拟主机。
- 学习使用[cronolog](#) 实现日志滚动。
- 学习使用[vlogger](#)实现虚拟主机的日志分离。
- 学习使用ab命令测试Apache服务器的性能。
- 学习安装和配置mod-evasive和mod-qos模块以提高Apache的安全性。

# Apache的安全模块

- mod\_evasive
  - 针对Dos和DDos攻击的检测和规避系统
- mod\_qos
  - 针对 Slowloris HTTP DoS 攻击的防护
- mod\_security
  - Web应用防火墙（WAF）
  - 使用 OWASP 提供的核心规则集保护Web应用程序
  - OWASP 可检测的攻击