apache

1. 安装 Apache：

```
yum -y install httpd httpd-devel
```

在 Ubuntu 里面叫做 Apache2，输入 localhost 能打开就算成功了。

2. Apache 命令：

(1) 使用 apachectl 命令启动脚本，-f 参数用于指定主配置文件。

```
/usr/local/apache2/bin/apachectl -f /usr/local/apache2/conf/httpd.conf
```

如果编译时已经指定主配置文件，则可以省略。

(2) 关闭　　（-k stop、关闭）

```
apachectl -k stop
```

注意：apache 指安装目录的 bin 目录下的 apachectl 控制脚本，下同。

-k 选项应该是 kill 的意思，以下几个命令的意思是发送相应的信号并执行动作。

(3) 立即重启　　（-k restart、立即重启）

```
apachectl -k restart
```

(4) 完成请求后停止脚本　　（-k graceful-stop、平滑关闭）

```
apachectl -k graceful-stop
```

(5) 完成请求后退出重启　　（-k graceful、平滑重启）

```
apachectl -k graceful
```

3. Apache 的 MPM 实际上就是用于处理请求的，不同的 mpm 方式处理请求的方式不同。

(1) prefork：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，在这种场景下，它会将请求放进队列中，一直等到有可用进程，请求才会被处理。

(2) Worker：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，因为多个子线程是共享父进程的内存地址的。如果使用 keep-alive 的长连接方式，某个线程会一直被占据，也许中间几乎没有请求，需要一直等待到超时才会被释放。如果过多的线程，被这样占据，也会导致在高并发场景下的无服务线程可用。该问题在 prefork 模式下，同样会发生。

(3) event：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（某些线程因为被 keep-alive，空挂在哪里等待，中间几乎没有请求过来，甚至等到超时）。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。

- 编译时可以编译为三种都支持，通过修改配置来更换：

```
--enable-mpms-shared=all
```

修改 httpd.conf 中的多路处理模式：　　（修改 httpd.conf 的多路处理模式）

```
#LoadModule mpm_prefork_module modules/mod_mpm_prefork.so
LoadModule mpm_worker_module modules/mod_mpm_worker.so
#LoadModule mpm_event_module modules/mod_mpm_event.so
```

4. 在使用 configure 脚本时用 --with-mpm=NAME 选项指定 MPM，NAME 就是你想使用的 MPM 的名称。一旦服务器编译完成，就可以用 ./httpd -l 命令来查看使用了哪个 MPM。这个命令将列出所有已经被编译到服务器中的模块，包括 MPM。
5. AddHandler 指令用于在文件扩展名与特定的处理器之间建立映射：

```
AddHandler php5-script .php
```

指定扩展名为 .php 的文件应被 php5-srcipt 处理器来处理。

6. AddType 指令用于在给定的文件扩展名与特定的内容类型之间建立映射。

```
AddType application/x-x509-ca-cert .crt
```

如果客户端需要 application/x-x509-ca-cert 资源，则返回.crt 类型的资源。

7. Options 指令的主要作用是控制特定目录将启用哪些服务器特性。指令的完整语法为：（Options、服务器特性）

```
Options [+|-]option [[+|-]option] ...
```

Options 指令后可以附加指定多种服务器特性，特性选项之间以空格分隔。（Options、指定服务器特性）

```
<Directory />
#指定根目录"/"启用 Indexes、FollowSymLinks 两种特性。
   Options Indexes FollowSymLinks
   AllowOverride all
   Order allow,deny
   Allow from all
```

```
</Directory>
```

Options 指令后可以附加的特性选项的具体作用及含义：

- All：表示除 MultiViews 之外的所有特性。这也是 Options 指令的默认设置。（all、允许所有特性）
- None：表示不启用任何的服务器特性。（None、不启用任何特性）
- FollowSymLinks：服务器允许在此目录中使用符号连接。（FollowSymLinks、允许符号连接）
- Indexes：如果输入的网址对应服务器上的一个文件目录，而此目录中又没有 DirectoryIndex 指令，那么服务器会返回由 mod_autoindex 模块生成的一个格式化后的目录列表，并列出该目录下的所有文件。（Indexes、没有 DirectoryIndex 则自动生成目录列表）
- MultiViews：允许使用 mod_negotiation 模块提供内容协商的"多重视图"。简而言之，如果客户端请求的路径可能对应多种类型的文件，那么服务器将根据客户端请求的具体情况自动选择一个最匹配客户端要求的文件。（MultiViews、多重视图、自动选择最匹配的文件）
- SymLinksIfOwnerMatch：服务器仅在符号连接与目标文件或目录的所有者具有相同的用户 ID 时才使用它。简而言之，只有当符号连接和符号连接指向的目标文件或目录的所有者是同一用户时，才会使用符号连接。如果该配置选项位于<Location>配置段中，将会被忽略。（符号连接与目标文件、相同的用户 ID 才使用）
- ExecCGI：允许使用 mod_cgi 模块执行 CGI 脚本。（ExecCGI、允许执行 CGI 脚本）
- Includes：允许使用 mod_include 模块提供的服务器端包含功能。（Includes、允许服务器端包含功能）
- IncludesNOEXEC：允许服务器端包含，但禁用"#exec cmd"和"#exec cgi"。但仍可以从 ScriptAlias 目录使用"#include virtual"虚拟 CGI 脚本。（IncludesNOEXEC、允许服务器端包含）

8. SSI，通常称为"服务器端包含"，用于在静态 HTML 文件中，根据需求动态地插入不同的内容。例如在一个静态网页中插入当前系统时间等。

- SSI 默认的扩展名是 .stm、.shtm 和 .shtml。
- SSI 技术通过在 html 文件中加入 SSI 指令，让服务器端在输出 html 之前解释 SSI 指令，并把解释完的结果和 html 代码一同输出给客户端。

9. 使用 SSI 可以动态地生成内容，而无须使用 CGI 接口。

➢ PHP 等动态语言就是通过 CGI 接口实现的。

10. Apache 过滤器分为输入过滤器和输出过滤器，用于对输入和输出的数据进行处理。使用过滤器可以实现统计流量、压缩等功能。（过滤器、流量统计、流量压缩）

11. Apache 内置了一些过滤器模块：

(1) mod_include：输出过滤器，用于启用 SSI。

(2) mod_ssl：实现了 SSL 加密。

(3) mod_deflate：输出过滤器，用于对输出进行压缩。

(4) mod_charset_lite：用于在发送数据到客户端之前转换数据的字符集。

(5) mod_ext_filter：用于自定义过滤器。

12.通过在配置文件中添加下列配置启用 SSI：

(1) 在 httpd.conf 文件中添加下列两行：

```
AddType text/html .shtml
AddOutputFilter INCLUDES .shtml
```

(2) 找到这样一行 Options Indexes FollowSymLinks，在后面添加 INCLUDES，变成这样：

```
Options Indexes FollowSymLinks INCLUDES
```

13.过滤器常见指令：

- SetInputFilter：设置输入过滤器。
- SetOutputFilter：设置输出过滤器。
- AddInputFilter：添加输入过滤器。
- AddOutputFilter：添加输出过滤器。

  例如：

```
<Directory /www/data/>
   SetOutputFilter INCLUDES
</Directory>
```

14.压缩由 DEFLATE 过滤器实现 。 以下指令将为容器中的文档启用压缩：

```
SetOutputFilter DEFLATE
```

- 使用 AddOutputFilterByType 指令添加压缩类型：

```
<Directory "/your-server-root/manual">
        AddOutputFilterByType DEFLATE text/html
</Directory>
```

- 使用 BrowserMatch 指令为特定浏览器设置是否进行压缩。如果某些浏览器无法处

理某种类型的压缩，可以使用这个指令。

```
BrowserMatch ^Mozilla/4 gzip-only-text/html
BrowserMatch ^Mozilla/4\.0[678] no-gzip
BrowserMatch \bMSIE !no-gzip !gzip-only-text/html
```

其中中间字段为正则表达式，而最后一个字段的 no-gzip 表示不压缩，而 gzip-only-text/html 表示只压缩 html 文档。

15. DEFLATE 过滤器提供了解压请求压缩包的功能，通过下列指令开启解压功能：

```
<Location /dav-area>
    SetInputFilter DEFLATE
</Location>
```

16. 如果客户端连接未使用 SSL 加密，通过下列指令设置拒绝响应：

```
Require ssl
```

如果既需要用户的客户端拥有证书，也要用户拥有正确的账号密码进行身份认证，则通过以下指令设置：

```
Require ssl-verify-client
Require valid-user
```

17. Listen 指令告诉服务器接只受来自特定端口(或地址+端口的组合)的请求。如果 Listen 指令仅指定了端口，则服务器会监听所有的 IP 地址；如果指定了地址+端口的组合，则服务器只监听来自此特定地址上特定端口的请求。使用多个 Listen 指令，可以指定在多个地址和端口上进行监听。（listen、监听端口、地址+端口、监听特定地址）

```
Listen 80
Listen 8000
Listen 192.170.2.1:80
Listen 192.170.2.5:8000
```

IPv6 地址必须用方括号括起来： （IPv6、方括号括起来）

```
Listen [2001:db8::a00:20ff:fea7:ccea]:80
```

18. 要使 Apache 仅仅只处理 IPv4 连接，无论你的平台是什么或者 APR 是否支持，只须对所有 Listen 指令都指定 IPv4 地址即可，如下所示：

```
Listen 0.0.0.0:80
Listen 192.170.2.1:80
```

19. Apache 主配置文件通常叫 httpd.conf ，其位置是编译时确定的，但可以用命令行

参数 -f 来改变。

20. AcceptFilter 指令用于配置协议的套接字优化。内核在收到数据或缓冲完整个 HTTP 请求之前不向服务器进程发送套接字。linux 默认的优化是：

```
AcceptFilter http data
AcceptFilter https data
```

21. 用 Include 指令和通配符附加许多其他配置文件，对主配置文件的更改需要重启才能生效。相关指令：

(1) define：用于定义一个变量。

```
<IfDefine TEST>
    Define servername test.example.com
</IfDefine>
```

如果 TEST 为真，则定义变量 servername，变量值为 test.example.com。

(2) <IfDefine> 指令：IfDefine 后面的条件为真时才生效。

```
httpd -DReverseProxy ...

# httpd.conf
<IfDefine ReverseProxy>
  LoadModule rewrite_module modules/mod_rewrite.so
  LoadModule proxy_module modules/libproxy.so
</IfDefine>
```

只有定义了 ReverseProxy 才执行里面的语句。

(3) Include 指令：在服务器配置文件中包含其它配置文件

```
Include /usr/local/apache2/conf/ssl.conf
Include /usr/local/apache2/conf/vhosts/*.conf
```

(4) TypesConfig 指令：指定 mime.types 文件的位置

22. Apache 配置文件的每一行包含一个指令，在行尾使用反斜杠"\"可以表示续行，但是反斜杠与下一行之间不能有任何其他字符(包括空白字符)。

23. 可以用 apachectl configtest 或者命令行选项 -t 检查配置文件中的错误，而无须启动 Apache 服务器。

24. Apache 是模块化的服务器，核心中只包含实现最基本功能的模块。

(1) <IfModule> 指令：如果启用了指定的模块，则执行标签中间的语句。

```
<IfModule mod_ssl.c>
  Include conf/ssl.conf
</IfModule>
```

(2) LoadModule 指令：加载目标文件或库，并将其添加到活动模块列表。

```
LoadModule status_module modules/mod_status.so
```

25. Order 指令用于指定命令的执行顺序，命令之间使用逗号隔开：

```
Order Deny,Allow
Allow from all
Deny from domain.org
```

上面例子虽然 deny 命令在 allow 命令之后，但由于 order 命令指定最后一个命令是 allow，apache 在处理到第二句 allow 的时候就已经匹配成功，根本就不会执行第三句。解决办法是将语句改为：

```
Order Allow,Deny
Allow from all
Deny from domain.org
```

26. allow 指令和 deny 指令用于允许或禁止相关 ip 对目录进行访问：

```
Order Allow,Deny
Allow from all
Deny from domain.org
```

27. 主配置文件中的指令对整个服务器都有效。相关指令：

(1) <Directory> 指令：封装一组指令，使之仅对文件空间中的某个目录及其子目录生效（Directory 、对某个目录生效）

```
<Directory /usr/local/httpd/htdocs>
   Options Indexes FollowSymLinks
</Directory>
```

(2) <DirectoryMatch> 指令：封装一些指令并作用于文件系统中匹配正则表达式的所有目录及其子目录（DirectoryMatch、对某个目录生效，正则表达式）

```
<DirectoryMatch "^/www/(.+/)*[0-9]{3}">
```

(3) <Files> 指令：包含作用于匹配指定文件名的指令（Files 、对某个文件生效）

```
<Files ~ "\.(gif|jpe?g|png)$">
```

在 Apache1.3 及其后继版本中，更推荐使用<FilesMatch>指令。

(4) <FilesMatch> 指令：包含作用于与正则表达式匹配的文件名的指令（FilesMatch、正则表达式）

```
<FilesMatch "\.(gif|jpe?g|png)$">
```

(5) <Location> 指令：将封装的指令作用于匹配的 URL（location、对某个 URL 生效）

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

使用<Location>来将指令应用于独立于文件系统之外的内容。文件系统之内的内容请使用<Directory>和<Files>指令。不过一个例外是<Location /> ，它可以方便的作用于所用 URL。

(6) <LocationMatch> 指令：将封装的指令作用于正则表达式匹配的 URL。<LocationMatch>和<Location>指令相同，提供了基于 URL 的访问控制。但它使用正则表达式作为参数，而不是简单字符串。（LocationMatch、匹配正则表达式的 URL）

```
<LocationMatch "/(extra|special)/data">
```

(7) <VirtualHost> 指令：包含仅作用于指定主机名或 IP 地址的指令（虚拟主机）

```
<VirtualHost 10.1.2.3>
  ServerAdmin webmaster@host.foo.com
  DocumentRoot /www/docs/host.foo.com
  ServerName host.foo.com
  ErrorLog logs/host.foo.com-error_log
  TransferLog logs/host.foo.com-access_log
</VirtualHost>
```

28. .htaccess 主要的作用有：URL 重写、自定义错误页面、MIME 类型配置以及访问权限控制等。主要体现在伪静态的应用、图片防盗链、自定义 404 错误页面、阻止/允许特定 IP/IP 段、目录浏览与主页、禁止访问指定文件类型、文件密码保护等。

29. .htaccess 的用途范围主要针对当前目录。

30. AccessFileName 指令用于设置.htaccess 配置文件的名字：

```
AccessFileName .acl
```

31. AllowOverride 指令：确定允许存在于.htaccess 文件中的指令类型。语法如下：

```
AllowOverride All|None|directive-type [directive-type] ...
```

32.缓存相关的模块：

(1) mod_cache：基于 URI 键的内容动态缓冲(内存或磁盘)

(2) mod_mem_cache：基于内存的缓冲管理器

(3) mod_disk_cache：基于磁盘的缓冲管理器

(4) mod_file_cache：提供文件描述符缓存支持，从而提高 Apache 性能

33.缓存相关指令：

(1) CacheEnable 指令：使用特定的存储类型缓存某些特定的 URL。

```
CacheEnable mem /manual
CacheEnable fd /images
CacheEnable disk /
```

(2) CacheDisable 指令：禁止缓存某些特定的 URL。

```
CacheDisable /local_files
```

(3) CacheFile 指令：将文件句柄存入内存的高速缓冲区，稍微降低系统性能，但是占用内存较少。

```
CacheFile /usr/local/apache/htdocs/index.html
```

(4) UseCanonicalName 指令：配置服务器如何确定它自己的域名。

- 使用 UseCanonicalName On 会使用 ServerName 这个域名生成完整规范的 URL。
- 设置为 UseCanonicalName Off 时，如果客户端提供了主机名和端口，Apache 将会使用这些信息来构建 URL。

(5) ForceLanguagePriority 指令：指定无法匹配单个文档的情况下所采取的动作。

34.存在于 Apache 中最基本的缓冲方式是由 mod_file_cache 实现的文件句柄缓存。胜于缓存文件内容本身，这个缓冲区维护一张打开的文件描述符表，用于保存在配置文件中使用 CacheFile 指令指定的文件的文件句柄。

```
CacheFile /usr/local/apache2/htdocs/index.html
```

如果缓存内容被删除，缓存句柄依然在生效，直到 Apache 被停止、文件描述符被关闭。

35.mod_mem_cache 也提供了一个文件句柄缓冲方案，可以通过 CacheEnable 指令来启用。

```
CacheEnable fd /
```

与 mod_cache 的方案相比，这种方案更加智能：缓存内容失效以后相应的句柄将不再被维护。

36. 内容协商功能可以根据浏览器提供的参数选择一个资源最合适的媒体类型、语言、字符集和编码的表现方式。

37. 有两种方式可以实现内容协商：（内容协商、类型表）

(1) 使用类型表(也就是一个 *.var 文件)明确指定各变种的文件名。

(2) 使用"MultiViews"搜索，即服务器执行一个隐含的文件名模式匹配，并在其结果中选择。

38. 类型表是一个与 type-map 处理器关联的文档。要使用这个功能，必须在配置中建立处理器，最好的方法是在配置文件中这样设置：

```
AddHandler type-map .var
```

39. 动态共享对象(DSO)机制也就是动态加载。

40. 设置一个 Apache 环境变量最基本的方法，就是使用没有什么限制的 SetEnv 指令。也可以使用 PassEnv 指令将启动 Apache 的操作系统 shell 的环境变量传进来。

41. 错误日志的格式相对灵活，并可以附加文字描述。某些信息会出现在绝大多数记录中，一个典型的例子是：

```
[Wed Oct 11 14:32:52 2000] [error] [client 127.0.0.1] client denied by server configuration: /export/home/live/ap/htdocs/test
```

其中，第一项是错误发生的日期和时间；第二项是错误的严重性，LogLevel 指令使只有高于指定严重性级别的错误才会被记录；第三项是导致错误的 IP 地址；此后是信息本身，在此例中，服务器拒绝了这个客户的访问。服务器在记录被访问文件时，用的是文件系统路径，而不是 Web 路径。 （时间、错误等级、IP 地址、信息本身）

42. 通用日志格式：（通用日志、IP 地址、身份、客户标识、服务器完成时间、动作、返回的状态码、字节数）

```
LogFormat "%h %l %u %t \"%r\" %>s %b" common
CustomLog logs/access_log common
```

例如：

```
127.0.0.1 - frank [10/Oct/2000:13:55:36 -0700] "GET /apache_pb.gif HTTP/1.0" 200 2326
```

- 127.0.0.1 (%h)：这是发送请求到服务器的客户的 IP 地址。
- - (%l)：这是由客户端 identd 进程判断的 RFC1413 身份(identity)，输出中的符号"-"表示此处的信息无效。
- frank (%u)：这是 HTTP 认证系统得到的访问该网页的客户标识(userid)，环境变量 REMOTE_USER 会被设为该值并提供给 CGI 脚本。
- [10/Oct/2000:13:55:36 -0700] (%t)：这是服务器完成请求处理时的时间
- "GET /apache_pb.gif HTTP/1.0" (\"%r\")：该客户的动作是 GET ，请求的资源是/apache_pb.gif ，使用的协议是 HTTP/1.0 。

- 200 (%>s)：这是服务器返回给客户端的状态码。
- 2326 (%b)：最后这项是返回给客户端的不包括响应头的字节数。

43.组合日志格式：（组合日志、referer、UserAgent）

```
LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-agent}i\"" combined
CustomLog log/access_log combined
```

这种格式与通用日志格式类似，但是多了两个 %{header}i 项，其中的 header 可以是任何请求头。这种格式的记录形如：

```
127.0.0.1 - frank [10/Oct/2000:13:55:36 -0700] "GET /apache_pb.gif HTTP/1.0" 200 2326 "http://www.example.com/start.html" "Mozilla/4.08 [en] (Win98; I ;Nav)"
```

其中，多出来的项是：

- "http://www.example.com/start.html"　　(\"%{Referer}i\")

"Referer"请求头。此项指明了该请求是被从哪个网页提交过来的，这个网页应该包含有/apache_pb.gif 或者其连接。

- "Mozilla/4.08 [en] (Win98; I ;Nav)"　(\"%{User-agent}i\")

"User-Agent"请求头。此项是客户端提供的浏览器识别信息。

44.以下是一个典型的日志滚动和为节省存储空间而压缩旧日志的例子：

```
$ mv access_log access_log.old
$ mv error_log error_log.old
$ apachectl graceful
$ sleep 600
$ gzip access_log.old error_log.old
```

45.MaxClients 指令：允许同时伺服的最大接入请求数量。

46. HostnameLookups 指令：启用对客户端 IP 的 DNS 查找。它会带来延迟，因为对每一个请求都需要作一次 DNS 查询。

为了提高性能，可以把这些指令包含在<Location /server-status>段中使之局部化。在这种情况下，只有对这个区域的请求才会发生 DNS 查询。下例禁止除了.html 和.cgi 以外的所有 DNS 查询：

```
HostnameLookups off
<Files ~ "\.(html|cgi)$">
  HostnameLookups on
</Files>
```

47.DocumentRoot 指令：组成网络上可见的主文档树的根目录。

48.Alias 指令：映射 URL 到文件系统的特定区域。

```
Alias /image /ftp/pub/image
<Directory /ftp/pub/image>
   Order allow,deny
   Allow from all
</Directory>
```

49. AliasMatch 指令：使用正则表达式映射 URL 到文件系统。

```
AliasMatch ^/icons(.*) /usr/local/apache/icons$1
```

50. ScriptAlias 指令：映射一个 URL 到文件系统并视之为 CGI 脚本。

```
ScriptAlias /cgi-bin/ /web/cgi-bin/
```

51. ScriptAliasMatch 指令：使用正则表达式映射一个 URL 到文件系统并视之为 CGI 脚本。

52. Apache 还允许将远程文档纳入本地服务器的网络空间中，因为 Web 服务器扮演一个代理服务器的角色，所以这种机制被称为反向代理。下例演示了当客户请求位于/foo/目录下的文档时，服务器从 internal.example.com 的/bar/目录下取回文档并返回给客户，似乎文档原本就在本地服务器上：

```
ProxyPass /foo/ http://internal.example.com/bar/
ProxyPassReverse /foo/ http://internal.example.com/bar/
ProxyPassReverseCookieDomain internal.example.com public.example.com
ProxyPassReverseCookiePath /foo/ /bar/
```

ProxyPass 指令使服务器正确地取回文档，同时，ProxyPassReverse 指令改变了起始于 internal.example.com 的请求，使之指向本地服务器上的目录。同样，ProxyPassReverseCookieDomain 和 ProxyPassReverseCookieDomain 指令将会改变后端服务器设置的 cookie 。

53. 浏览器地址栏或者 HTML 连接中的 URL 被拼写错了，Apache 提供了 mod_speling 模块来帮助解决这个问题，它会接管"File Not Found"错误并查找相似文件，如果找到了唯一的一个，则会重定向到这个文件，如果不止一个，则会列一张表反馈给用户。

54. Apache 提供了多种方法对 DocumentRoot 以外的文件进行访问，在 Unix 系统中，可以在文件系统的 DocumentRoot 目录下放置符号链接以访问其外部文件，考虑到安全问题，这种方法仅在相应目录的 Options 指令中设置了 FollowSymLinks 或 SymLinksIfOwnerMatch 时才有效。

55. Apache 文件说明

(1) 主配置文件　（主配置文件、httpd.conf）

```
/etc/httpd/conf/httpd.conf
```

(2) 扩展配置文件　　（扩展配置文件、/etc/httpd/conf.d/*.conf）

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

当 Apache 启动时，会加载/etc/httpd/conf.d/目录中的所有以.conf 结尾的文件，做为配置文件来使用，所以管理员可以将配置推荐写在.conf 中，如果将配置项写入主配置文件，系统升级时，配置项还要重新修改一遍，如果写在扩展配置项，则不存在此问题，同时也可以从繁杂的主配置文件中脱离出来。

(3) 扩展模块目录

```
/usr/lib/httpd/modules/
```

apache 是模块化的,访问 php 的时候 Apache 就调用 php 模块来执行,访问 SVN 的时候 Apache 就调用 svn 模块来执行。

(4) 默认数据目录

```
/var/www/html
```

(5) 日志目录

```
/var/log/httpd/
```

56.主配置文件说明:

(1) ServerTokens OS：系统信息，在访问出错时出现；把 OS 改为 Prod，就不显示系统信息　　（ServerTokens OS、系统信息、Prod、不显示）

```
ServerTokens OS
```

(2) ServerSignature On　　（ServerSignature、隐藏系统、email、显示邮箱）

把 On 改为 Off 就连普通的系统都给隐藏起来；改为 Email 就会显示管理员的邮箱(邮箱需要另外配置 ServerAdmin。

(3) ServerAdmin root@localhost　（ServerAdmin、管理员邮箱）

管理员邮箱

(4) ServerName localhost　　（ServerName、主机名）

服务器的主机名，一般是用虚拟机来设置，通常这个值是自动指定的，推荐显式的指定它以防止启动时出错。

(5) ServerRoot "/etc/httpd"　　（ServerRoot、设置服务器所在的目录）

配置项的根目录，类似 html 里面的 base；默认到这个路径里面找；

(6) PidFile run/httpd.pid　　（PidFile、进程 PID）

进程 PID，位置在 /etc/httpd/run/httpd.pid，主进程决定着子进程 。

(7) Timeout 60 （Timeout、设置连接空等时间）

若 60 秒后没有收到或送出任何数据就切断该连接

(8) KeepAlive Off （KeepAlive、是否开启持久化链接）

是否开启持久化链接，访问网站时要对网站的很多资源，如 css、js、image 等等创建不同的链接;事实上我们可以建立一个持久化链接来应对多个请求;

(9) MaxKeepAliveRequests 100 （MaxKeepAliveRequests、持久化链接能应对的请求数）

一个持久化链接最多能应对多少个请求。

(10) KeepAliveTimeout 15 （KeepAliveTimeout、15 秒不链接就断开）

15 秒不链接就断开。

(11) Listen 80 （Listen、监听指定端口）

监听端口，默认是 80，一般不同改变；如果要改变，注意以下几点:

- 如果修改为 192.168.1.22:8080，表示只能通过 192.168.1.22:8080 访问
- 如果这里要更改为其他端口比如 88 的话，下面的 ServerName localhost:88 也得更改(如果是注释掉的，要取消注释)
- 如果要监听多个端口，就多写几个 Listen

(12) Include conf.d/*.conf （Include、扩展配置文件）

扩展配置文件 /etc/httpd/conf.d/。我们一般在配置文件尾部再加上一句 Include conf/vhosts/*.conf，把其他虚拟主机的配置分离开。

(13) User apache （User、进程所有者）

Apache 进程所有者

(14) Group apache （Group、进程所属组）

Apache 进程所属组

(15) DirectoryIndex index.html index.html.var（DirectoryIndex、默认主文件）

默认主文件

(16) DocumentRoot "/var/www/html" （DocumentRoot、网站根目录）

网站数据根目录。

(17) ErrorDocument 404 /404.html（ErrorDocument、指定 404 文件）

创建 404 文件。404 可以通过 PHP 程序来处理(在框架中)，可以通过 rewrite 来处理，但是最理想的模式是让 Apache 来处理 。

➢ 注意:

- Apache 的配置规则是“后出现，先应用”，后面的出现的配置会覆盖前面的。

- 以上配置都应该在扩展配置里面覆盖更改或增加;

57.配置虚拟主机时建议在主配置文件中增加一句

```
Include conf/vhosts/*.conf
```

然后就在 vhosts 目录下添加虚拟主机配置文件。在配置前打开 NameVirtualHost *:80 注释，注意此处要与 Apache 主配置监听端口一致。（include、添加虚拟主机配置文件）

(1) 按域名配置

```
<VirtualHost *:80>
  ServerAdmin webmaster@dummy-host.example.com  #站点邮箱
  DocumentRoot /www/docs/dummy-host.example.com #访问目录
  ServerName dummy-host.example.com           #域名
  ErrorLog logs/dummy-host.example.com-error_log                  #错误日志目录
  CustomLog logs/dummy-host.example.com-access_log common    #访问日志目录

  <Directory  /var/www/html/web1>
    此处可以覆盖主配置文件中的 Directory 部分，配置规则完全按照 Directory 来
  </Directory>
</VirtualHost>
```

(2) 按 IP 端口配置

```
Listen 127.0.0.1:81
<VirtualHost 127.0.0.1:81>
  DocumentRoot "/www/docs/dummy-host.example.com"
  ServerName 127.0.0.1:81
</VirtualHost>
```

注意:

- 虚拟主机的目录名最好用网站域名作为目录名，这是一个好习惯（目录名用网站域名）
- 在 LNMPA 架构中,虚拟主机的配置是在 Apache 中，而不是在 Nginx 里面

58.require 用于进行访问控制:

```
<Files ".ht*">
  Require all denied                                  #对相关文件拒绝所有请求访问资源
</Files>
```

常见访问控制指令:

(1) Require all granted：允许所有请求访问资源 （允许请求）

(2) Require all denied：拒绝所有请求访问资源 （拒绝请求）

(3) Require env env-var [env-var] ...：当指定环境变量设置时允许访问（指定环境变

量时允许）

(4) Require method http-method [http-method] ...：允许指定的 http 请求方法访问资源（指定请求方法时允许）

(5) Require expr expression：当 expression 返回 true 时允许访问资源 （返回 true 时允许）

(6) Require user userid [userid] ...：允许指定的用户 id 访问资源 （允许指定用户访问）

(7) Require group group-name [group-name] ...：允许指定的组内的用户访问资源（允许组用户访问）

(8) Require valid-user：所有有效的用户可访问资源 （允许所有用户访问）

(9) Require ip 10 172.20 192.168.2：允许指定 IP 的客户端可访问资源（允许指定 IP 用户访问）

(10) Require not group select：select 组内的用户不可访问资源（不允许 select 组用户访问）

59. .htaccess 分布式配置文件，它没有文件名，只有一个文件后缀就是.htaccess。所以一般在 windows 下还没法新建这个文件，因为 windows 不允许文件名是空的。

60. 内部重定向，是指域名并不发生变化，只是将某个 URL 映射到了文件系统中的另一个路径。

61. 外部重定向，是指 URL 已经法发生变化，在浏览器中显示看到的域名也跟着变化了。

62. URL 重定向是.htaccess 的重头戏，它可以将长地址转为短地址、将动态地址转为静态地址、重定向丢失的页面、防止盗链、实现自动语言转换等。实现所有这些神奇功能的模块叫做 mod_rewrite，请确保你的服务器安装并启用了该模块：

```
sudo a2enmod rewrite
```

63. 重写规则的作用范围，有如下三种：

(1) 可以使用在 Apache 主配置文件 httpd.conf 中

(2) 可以使用在 httpd.conf 里定义的虚拟主机配置中

(3) 可以使用在基本目录的跨越配置文件.htaccess 中

64. 重写规则具体有 RewriteBase、 RewriteCond、 RewriteEngine、RewriteRule 等九个指令。通常最常用的是 RewriteBase、 RewriteCond、 RewriteEngine、RewriteRule 四个指令：

(1) RewriteEngine：是否使用 Rewrite 模式的开关，使用就设置成 on，否则设置成 off。作用域在：server config、virtual host、directory、.htaccess（RewriteEngine、是否使用 Rewrite 模式）

(2) RewriteBase：显式地设置了目录级重写的基准 URL，也就是前缀 URL。假设一个

网站目录使用了别名操作: Alias /xyz /abc/def 那么当客户端访问/xyz/xxx.html 文件时是相当于访问 /abc/def/xxx.html 的:

```
RewriteEngine On
RewriteBase /xyz
RewriteRule ^oldstuff\.html$ newstuff.html
```

(3) RewriteCond 就像我们程序中的 if 语句一样，表示如果符合某个或某几个条件则执行 RewriteCond 下面紧邻的 RewriteRule 语句，这就是 RewriteCond 最原始、基础的功能，例如:

```
RewriteEngine on
RewriteCond %{HTTP_USER_AGENT} ^Mozilla/5.0.*
# RewriteRule 当条件满足时，把 url 怎么改写 紧跟在 RewriteCond 后面
RewriteRule index.php index.m.php
RewriteCond %{HTTP_USER_AGENT} ^Lynx.*
RewriteRule index.php index.L.php
# 上面的条件都不满足时，执行下列这个改写规则
RewriteRule index.php index.b.php
```

注意：RewriteCond 下面那条语句只能是 RedwriteRule，不能是 RewriteCond。

51. Apache mod_rewrite 规则重写的标志一览

(1) R[=code](force redirect)： 强制外部重定向 （R、强制重定向）

(2) F(force URL to be forbidden)：禁用 URL,返回 403HTTP 状态码。 （F、禁止 URL）

(3) G(force URL to be gone)： 强制 URL 为 GONE，返回 410HTTP 状态码。（G、强制 URL 为 GONE）

(4) P(force proxy)： 强制使用代理转发。 （P、强制使用代理转发）

(5) L(last rule)： 表明当前规则是最后一条规则，停止分析以后规则的重写。（L、最后一条规则）

(6) N(next round)： 重新从第一条规则开始运行重写过程。（N、重新开始运行）

(7) T=MIME-type(force MIME type)：强制 MIME 类型（T、强制 MIME 类型）

(8) NC(no case)： 不区分大小写 （NC、不区分大小写）

(9) QSA(query string append)： 追加请求字符串 （QSA、追加请求字符串）

(10) NE(no URI escaping of output)： 不再输出转义特殊字符（NE、不再输出转义字符）

实例:

```
RewriteCond %{REQUEST_URI} dianping/page/2
RewriteRule ^(.*)$ . [R=301,L]
```

将 xxx.com/dianping/page/2 重定向到 xxx.com。

注意：

- 因为.htaccess 默认是当前目录，所以 dianping 前面的 xxx.com 可以省略，"dianping”前面不能加"/"。第二句的第一个点号表示当前目录的地址。
- (.*)前面也不能加"/"。
- REQUEST_URI 是 php 中的$_SERVER[]变量。

52. 常见的$_SERVER[]变量：

(1) HTTP_HOST：当前请求头中 Host 项的内容 （HTTP_HOST、host 项的内容）

(2) REQUEST_URI：保存要访问的页面 （REQUEST_URI、要访问的页面）

(3) HTTP_REFERER：当前页面的来源地址，也就是当前客户端从哪里跳转到当前页面的。 （'TTP_REFERER、来源地址）

(4) HTTP_USER_AGENT：当前请求头中 User-Agent 项的内容
（HTTP_USER_AGENT、User-Agent 项的内容）

实例 1：

```
RewriteCond %{HTTP_REFERER} im286.com [NC]
  RewriteRule .*\.(jpg|jpeg|gif|png|rar|zip|txt|ace|torrent|gz|swf)$
http://www.xxx.com/fuck.png [R,NC,L]
```

防止 im286.com 盗链。

实例 2：

```
RewriteEngine on
  RewriteCond %{HTTP_USER_AGENT} MSIE [NC,OR]
  RewriteCond %{HTTP_USER_AGENT} Opera [NC]
  RewriteRule ^.* – [F,L]
```

屏蔽 IE 和 Opera 浏览器。这里的减号表示不允许访问。（减号、不允许访问）

53. 将旧的 URL 做 301 重定向到新的 URL，使用 mod_rewrite 并不是优先考虑的，有更简单的方式，比如 mod_alias 模块提供的 Redirect 指令；在什么情况下使用 mod_rewrite 呢？如果你不能够获取配置 Apache 配置文件的权限，只能够编辑.htaccess 文件，这时就只能用 mod_rewrite 提供的指令。

54. Redirect 指令用于对 URL 进行重定向：

```
Redirect [status] [URL-path] URL
```

旧的路径 URL-path 不允许使用相对路径。新的 URL 可以是以方案和主机名开头的绝对 URL，也可以是以斜线开始的 URL 路径。

```
Redirect permanent "/dianping" "/category/dianping"
```

将 xxx.com/dianping 重定向到 xxx.com/category/dianping。

- 注意：由于在 wordpress 中 xxx.com/dianping 和 xxx.com/category/dianping 实际上都是分类目录，所以重定向时无法 mod_rewrite 来进行重定向，使用 mod_rewrite 进行重定向会导致循环重定向而出错，只能使用 Redirect。

status 有以下几个选项：

- permanent：永久重定向状态（301）
- temp：临时重定向状态（302）
- seeother：指示资源已被替换的“查看其他”状态（303）
- gone：指示资源已被永久删除的“Gone”状态（410）。 当使用此状态时，应该省略 URL 参数

55. RedirectMatch 指令相当于 Redirect，使用正则表达式，而不是简单的前缀匹配。

```
RedirectMatch [ status ] regex URL
```

例如：

```
RedirectMatch "(.*)\.gif$" "http://other.example.com$1.jpg"
```

要将所有 GIF 文件重定向到其他服务器上的相似名称的 JPEG 文件。

56. RedirectPermanent 指令指永久重定向状态，完全相当于 Redirect permanent。

```
RedirectPermanent URL-path URL
```

57. RedirectTemp 指令指临时重定向，完全相当于 Redirect temp。
（RedirectTemp、 完全相当于 Redirect temp）

```
RedirectTemp URL-path URL
```

58. NameVirtualHost 指令：为一个基于域名的虚拟主机指定一个 IP 地址
（NameVirtualHost、指定 IP 地址）

```
NameVirtualHost 111.22.33.44
```

如果服务器上所有的 IP 地址都会用到，你可以用"*"作为 NameVirtualHost 的参数。如果你打算使用多端口(如运行 SSL)你必须在参数中指定一个端口号，比如"*:80"。

59. <VirtualHost>的参数与 NameVirtualHost 的参数必须是一样的。在每个<VirtualHost>段中，至少要有一个 ServerName 指令来指定伺服哪个主机和一个 DocumentRoot 指令来说明这个主机的内容位于文件系统的什么地方。

```
NameVirtualHost *:80

<VirtualHost *:80>
  ServerName www.domain.tld
  ServerAlias domain.tld *.domain.tld
DocumentRoot /www/domain
</VirtualHost>

<VirtualHost *:80>
  ServerName www.otherdomain.tld
  DocumentRoot /www/otherdomain
```

```
</VirtualHost>
```

你可以用一个固定的 IP 地址来代替 NameVirtualHost 和<VirtualHost>指令中的"*"号。

60.把 ServerAlias 指令放入<VirtualHost>中可以解决多域名访问的问题。

```
ServerAlias domain.tld *.domain.tld
```

Nginx

1. 默认，nginx 将其主进程的 pid 写入到 /usr/local/nginx/nginx.pid 文件中。通过传递参数给 ./configure 或使用 pid 指令，来改变该文件的位置。（nginx.pid、进程 pid）
2. 关闭 nginx

(1) nginx 从容停止命令，等所有请求结束后关闭服务：（从容停止、kill -QUIT）

```
kill -QUIT  nginx 主进程号
```

(2) nginx 快速停止命令，立刻关闭 nginx 进程：（立刻关闭、kill -TERM）

```
kill -TERM nginx 主进程号
```

(3) 如果以上命令不管用，可以强制停止 （强制停止、kill -9）

```
kill -9 nginx 主进程号
```

3. 如果嫌麻烦可以不用查看进程号，直接使用命令进行操作。其中/usr/local/nginx/nginx.pid 为 nginx.conf 中 pid 命令设置的参数，用来存放 nginx 主进程号的文件：

```
kill -QUIT `cat /usr/local/nginx/nginx.pid`
```

4. nginx 重启命令可以分成几种类型：

(1) 简单型，先关闭进程，修改你的配置后，重启进程。（简单重启、从容停止后修改）

```
kill -QUIT cat /usr/local/nginx/nginx.pid
sudo /usr/local/nginx/nginx
```

(2) 平滑重启：如果更改了配置就要重启 Nginx，要先关闭 Nginx 再打开？不是的，可以向 Nginx 发送信号，平滑重启。 （平滑重启、kill -HUP）

```
kill -HUP 主进程号或进程号文件路径
```

或者使用：

```
/usr/nginx/sbin/nginx -s reload
```

(3) 平滑升级

步骤 1：如果升级 Nginx 程序，先用新程序替换旧程序文件，编译安装的话新程序直接编译到 Nginx 安装目录中。

步骤 2：执行命令

```
kill -USR2 旧版程序的主进程号或进程文件名
```

步骤 3：逐步停止旧版 Nginx，输入命令

```
kill -WINCH 旧版主进程号
```

5. 保存服务器名字的 hash 表是由指令 server_names_hash_max_size 和 server_names_hash_bucket_size 所控制的。参数 hash bucket size 总是等于 hash 表的大小。（hash 表、server_names_hash_max_size 、server_names_hash_bucket_size）
6. nginx 采用了非阻塞的事件模型，可以同时监听多个事件。nginx 目前实现了 kqueue、epoll、select、poll、eventport、aio、devpoll、rtsig 和基于 windows 的 select 共 9 个事件模型。具体使用哪一个可以在配置文件中设置，nginx 关于事件模型的设置是在 events{}块中使用 use 指使设置：（事件模型、events、use 指令）

```
events{
   use kqueue;
}
```

7. 事件模型：

(1) select – 标准方法。如果当前平台没有更有效的方法，它是编译时默认的方法。你可以在编译时使用配置参数

```
--with-select_module
```

和

```
--without-select_module
```

来启用或禁用这个模块。

(2) poll - 标准方法。 如果当前平台没有更有效的方法，它是编译时默认的方法。你可以在编译时使用配置参数

```
--with-poll_module
```

和

```
--without-poll_module
```

来启用或禁用这个模块。

8. 相关指令：

(1) worker_cpu_affinity：仅适用于 linux，使用该选项可以绑定 worker 进程和 CPU。（worker_cpu_affinity、绑定 worker 进程和 CPU）

```
worker_proceses    4;
worker_cpu_affinity 0001 0010 0100 1000;
```

分别给每个 worker 进程绑定一个 CPU。

(2) worker_priority：使用该选项可以给所有的 worker 进程分配优先值。（worker_priority 分配优先值）

```
worker_priority on;
```

(3) worker_processes：

```
worker_processes 5;
```

nginx 使用 5 个 worker 进程。

9. 相比 apache 的指令，nginx 的指令有下划线，且全是小写字母，而 apache 的指令没有下划线，每个单词首字母为大写：（nginx、小写下划线、apache、首字母大写、无下划线）

```
ServerName localhost    #主机域名
```

而 nginx 的指定主机域名的指令为：

```
server_name  www.xiaopet.com;
```

10.

个人总结

1. vsftpd 的/etc/vsftpd/ftpusers 文件控制了哪些用户可以登陆，哪些用户不能登陆：

```
# Users that are not allowed to login via ftp
root
bin
daemon
adm
lp
sync
shutdown
halt
mail
news
```

```
uucp
operator
games
nobody
#www
```

只要在需要登陆的地方注释掉即可。

2. apache 目录的用户和属主必须的 httpd.conf 中的 User 和 Group 指令指定的用户和组相同，否则会出现各种问题。
3. php 不需要执行，只要安装后在 PATH 变量中能找到 PHP 目录下的 bin 目录即可，不需要执行。
4. 对于站点根目录的设置，apache 使用 DocumentRoot，而 nginx 则直接使用 root。另外，apache 还有服务器根目录指令 ServerRoot。
5. apache 首页使用 dir_module 模块的 DirectoryIndex 来设置，而 nginx 则在 location 里使用 index 来设置，apache 设置首页：

```
<IfModule dir_module>
  DirectoryIndex index.php index.php3 index.html index.htm          #设置首页
</IfModule>
```

nginx 设置首页：

```
location / {
        root   html/xiaopet;              # 站点根目录
        index  index.html index.htm; # 首页，多个用空格分开
    }
```

6. apache 也支持 location：

```
<VirtualHost 107.150.12.152>          #拒绝 IP 访问
  ServerName 107.150.12.152           #设置的关键是这句，还有 Deny 语句
  <Location />
    Order Allow,Deny
    Deny from all
  </Location>
</VirtualHost>
```

7. nginx 的基本设置格式如下：

```
……
event{
  ……
}
http{
```

```
  ……
  server{
    listen xxx;
    server_name xxx;
    （然后就是各种 location 语句）
  }
}
```

server 里面最基本的配置是 listen、server_name、access_log、error_log 还有 location。

apache 虚拟主机配置（不包括 http.conf 主配置）的的基本设置如下：

```
<VirtualHost 107.150.12.152>          #拒绝 IP 访问
  ServerName 107.150.12.152           #设置的关键是这句，还有 Deny 语句
  <Location />
    Order Allow,Deny
    Deny from all
  </Location>
</VirtualHost>
```

而 VirrualHost 里面最基本的配置是 ServerName 还有 Location。

8. 在使用 apache 禁止 ip 访问时，需要首先使用 NameVirtualHost 设置 ip：

```
NameVirtualHost 107.150.12.152

<virtualhost 107.150.12.152:80>
        ServerName  107.150.12.152
        <location />
                Order Allow,Deny
                Deny from all
        </location>
</virtualhost>

<virtualhost 107.150.12.152:80>
        ServerName  aiairedian.com
        ServerAlias www.aiairedian.com
        DocumentRoot  /var/www/html
</virtualhost>
```

9.