Впервые опубликовано на developerWorks 25.04.2006

Подготовка к экзамену LPI: Web-сервисы

Средний уровень администрирования (LPIC-2) тема 208

Об Apache и Squid

Web-сервер Apache

Apache -- это самый распространенный web-сервер во всем Интернете, и среди серверов на основе Linux его преимущество выглядит подовляющим. Кроме него существуют и другие специализированные web-сервера (некоторые из них показывают более высокую производительность для определенных задач), но именно Apache устанавлиевается по умолчанию.

В большинстве дистрибутивов Linux Apache предустановлен и часто уже работает, будучи запущенным во время установки, даже если вы и не настраивали его. Если Apache не установлен, его можно установить, используя обычную систему установки вашего дистрибутива, или же вы можете скачать последнюю версию с Apache HTTP Server Project. Большое количество дополнительных возможностей может быть обеспечено модулями, многие из которых поставляются вместе с Apache, а остальные доступны от третьих лиц.

Хотя с 2001 года последней версией Apache является ветка 2.x, Apache 1.3.x все еще широко используется, и ветка 1.3.x продолжает поддерживатсья в виде bug fixes и обновлений безопасности. Между 1.3 и 2.x существуют незначительные различия в конфигурации; некоторые модули доступны для 1.3, и не доступны для 2.x. Последние версии на момент написания этого пособия -- 1.3.34 (стабильная), 2.0.55 (стабильная), и 2.1.9 (бета).

Как правило, новый сервер использует последнюю стабильную версию семейства 2.x. Если у вас нет нужды использовать именно более старый модуль, 2.x дает хорошую стабильность, больше возможностей, и совокупную лучшую производительность (в некоторых задачах, таких как поддержка PHP в 1.3 все еще работает лучше). Заглядывая в будущее, учтите, что новые возможности будут, безусловно, лучше поддерживаться версией 2.x, а не 1.3.x.

Squid прокси-сервер

Squid -- это прокси-кеширующий сервер для web-клиентов, который поддерживает протоколы HTTP, FTP, TLS, SSL, и HTTPS. Скорость передачи может быть увеличена, а ширина канала сети -- снижена благодаря работе прокси-сервера в локальной сети, или, по меньшей мере, где-нибудь ближе к вашей сети, чем запрашиваемые ресурсы. Когда один и тот же ресурс запрашивается машинами,

обслуживаемыми одним и тем же сервером Squid несколько раз, этот ресурс доставляется из локальной копии на сервере, и не запросу не требуется проходить через множество сетевых маршрутизаторов и потенциально замедлять или нагружать сервера назначения.

Можно настроить Squid в качестве явного прокси, которого следует настраивать для каждого web-клиента (браузера), или же можно сделать так, чтобы он перехватывал все web-запросы, выходящие за пределы LAN и кэшировал весь такой трафик. Можно также указать Squid при помощи его многочисленных опций, как долго и при каких условиях следует хранить web-страницы в кэше.

Другие источники

Как и при изучении других приложений Linux, всегда полезно обращаться к man-страницам любых рассматриваемых здесь утилит. Версии и опции могут отличаться для различных версий утилиты или ядра, или же у различных дистирбутивов Linux. За дополнительной информацией обращайтесь на Linux Documentation Project, он содержит большое количество различных полезных документов, в особенности HOWTO. Также опубликовано множество книг по сетям Linux; я считаю, что книга O'Reilly's TCP/IP Network Administration, Крейга Ханта, вполне может помочь. (Ссылки Источники располагаются ниже.)

По работе с Apache тоже написано много хороших книг. Некоторые касаются общих вопросов администрирования, тогда как другие охватывают конкретные модули или специальные настройки Apache. Посетите ваш любимый книжный магазин и поищите там книги с подходящими названиями.

Внедрение web-сервера

Туча демонов

Запуск Apache похож на запуск любого другого демона. Обычно хочется поместить запуск в скрипт инициализации системы, однако, в принципе, можно запускать Apache в любое время. В большей части систем сервер Apache называется httpd, хотя он может вместо этого называться и apache2. Вероятнее всего, сервер установлен в /usr/sbin/, но возможны и другие расположения, в зависимости от вашего дистрибутива и от того, как вы установили сервер.

Чаще всего Apache запускается без опций, хотя об опциях `-d serverroot` и `-f config` следует помнить. Первая позволяет указывать расположение локального каталога, откуда поставляется содержимое; вторая позволяет указывать конфигурационный файл, отличный от используемого по умолчанию. Файл конфигурации может отменять опцию `-f` с помощью директивы `ServerRoot`. По умолчанию, конфигурационными файлами являются либо apache2.conf, либо httpd.conf, в зависимости от установок при компиляции. Эти файлы, скорее всего,

располагаются в /etc/apache2/, /etc/apache/, /etc/httpd/conf/, /etc/httpd/apache/conf, или в каких-нибудь других местах, в зависимости от версии, дистрибутива Linux, и от того, как вы установили и скомпилировали Apache. Вызов `man apache2` или `man httpd` должен выдать вам системно-зависимые подробности.

Демон Apache отличается от других серверов тем, что он обычно создает несколько выполняющихся копий самого себя. Первичная копия просто порождает остальных, в то время как эти вторичные копии и обслуживают входящие запросы. Целью наличия множества запущенных копий является создание набора обработчиков на случай нескольких одновременных запросов к серверу; при необходимости могут запускаться дополнительные копии демона в соответствии с параметрами конфигурации. Первичная копия обычно запускается от root, но остальные копии по соображениям безопасности запускаются как более ограниченный пользователь. Например:

Listing 1. The многоликость выполняющихся копий Apache

```
# ps axu | grep apache2
root      6620     Ss   Nov12   0:00 /usr/sbin/apache2 -k start -DSSL
www-data  6621     S    Nov12   0:00 /usr/sbin/apache2 -k start -DSSL
www-data  6622     Sl   Nov12   0:00 /usr/sbin/apache2 -k start -DSSL
www-data  6624     Sl   Nov12   0:00 /usr/sbin/apache2 -k start -DSSL
dqm        313     S+   03:44   0:00 man apache2
root       637     S+   03:59   0:00 grep apache2
```

На большом числе систем ограниченным пользователем является nobody. В Listing 1 это -- пользователь `www-data`.

Включение конфигурационных файлов

Как уже упоминалось, поведение Apache определяется директивами в его конфигурационном файле. Для систем Apache2, главный конфигурационный файл, скорее всего, находится в /etc/apache2/apache2.conf, но часто этот файл содержит многочисленные `Include` statements для добавления информации о конфигурации из других файлов, возможно, даже с шаблонами. В общем случае, конфигурация Apache, вероятно, содержит сотни директив и опций (большая часть которых не описывается в этом пособии).

В частности, несколько файлов, вероятно должны быть включены. Можно взглянуть в настройки "пользователей" файла httpd.conf, для использования прежних Apache 1.3 файлов конфигурации, использующих то же имя. Виртуальные хосты обычно задаются в отдельных конфигурационных файлах, соответствующих шаблону, например, вот так:

Listing 2. Задание виртуальных хостов

```
# Include the virtual host configurations:
Include /etc/apache2/sites-enabled/[^.#]*
```

При использовании Apache 2.x, модули обычно тоже определяются в отдельных конфигурационных файлах (более часто в том же файле в 1.3.x). Например, в моей системе включения таковы:

Listing 3. From /etc/apache2/apache2.conf

```
# Include module configuration:
Include /etc/apache2/mods-enabled/*.load
Include /etc/apache2/mods-enabled/*.conf
```

Фактически использование модуля в запущенном сервере Apache требует два шага в файле конфигурации, оба загружающих и и активирующих его:

Listing 4. Загрузка дополнительного модуля Apache

```
      # cat /etc/apache2/mods-enabled/userdir.load
      LoadModule userdir_module /usr/lib/apache2/modules/mod_userdir.so
      # cat /etc/apache2/mods-enabled/userdir.conf
      <IfModule mod_userdir.c>
          UserDir public_html
          UserDir disabled root

          <Directory /home/*/public_html>
              AllowOverride FileInfo AuthConfig Limit
              Options MultiViews Indexes SymLinksIfOwnerMatch
IncludesNoExec
          </Directory>
      </IfModule>
```

Шаблоны в строках `Include` вставят все файлы .load и .conf в каталоге /etc/apache2/mods-enabled/

Обратите внимание на общую мысль: Основные директивы -- это команды в одну строку с некоторыми параметрами; более сложные директивы используют XML-льный тег открыть/закрыть для вложенных команд. Является ли директива однострочной или же стиль открыть/закрыть, следует знать -- по своему усмотрению стили выбирать нельзя.

Файлы журнала

Важный класс директив конфигурации касается журналирования действий Apache. Можно задавать различные типы информации и степени детализации для операций Apache. Ведение журнала ошибок всегда приветствуется; это можно задать одной директивой:

Listing 5. Задание журнала ошибок

```
# Global error log.
```

```
ErrorLog /var/log/apache2/error.log
```

Можно добавить другие журналы для записи обращений к серверу, ссылающего сайта, и другой информации, удовлетворяющей вашим индивидуальным целям. Операция журналирования настраивается двумя директивами. Сначала директива `LogFormat` использует набор специальных переменных для задания, того, что помещать в файл журнала; затем, директива `CustomLog` говорит Apache actually записывать события в указанном формате. Можно задать бесчисленное число форматов невзирая на то, используются ли они на самом деле. Это позволяет включать и выключать журналирование подробностей, в зависимости от меняющихся потребностей.

Переменные в `LogFormat` похожи на переменные оболочки, но имеют в начале %. Некоторые переменные состоят из одной буквы, в то время как другие имеют длинные имена, окруженные скобками, как показано в Listing 6.

Listing 6. Переменные LogFormat

```
LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\""
combined
CustomLog /var/log/apache2/referer_log combined
```

Обратитесь к книге с полной документацией Apache за списком всех переменных. Широко используются следующие: `%h` для IP-адреса клиента, выполняющего запрос, `%t` для дати и времени запроса, `%>s` для HTTP статуса кода, и написанный с ошибкой `%{Referer}` для адреса ссылающего сайта, на котором есть ссылка на обрабатываемую страницу.

Имена, используемые в директивах `LogFormat` и `CustomLog` являются произвольными. В Listing 6 использовалось имя `combined`, но вместо нее могла стоять, например, `myfoobarlog`. Однако несколько имен являются общепринятыми и даются в образцах файлах конфигурации, такие как `combined`, `common`, `referer`, и `agent`. Эти специальные форматы зачастую поддерживаются утилитами, анализирующими log-файлы.

Поддержка работы web-сервера

Виртуальные хосты, multi-homing и отдельные настройки различных каталогов

Индивидуальные каталоги, обслуживаемые сервером Apache, могут иметь свои собственные настройки конфигурации. Однако в главном конфигурационном файле может указать ограничения на то, какие опции могут быть настроены локально. Если настойка каждого каталога в отдельности разрешается, то используется директива `AccessFileName`, и обычно прописывается локальное конфигурационное имя файла .htaccess. Ограничения на возможности локальной настройки каталога

определяются в директиве <Directory>. Например:

Listing 7. Пример директивы directory

```
#Let's have some Icons, shall we?
Alias /icons/ "/usr/share/apache2/icons/"
<Directory "/usr/share/apache2/icons">
      Options Indexes MultiViews
      AllowOverride None
      Order allow,deny
      Allow from all
</Directory>
```

Часто одновременно с индивидуально настраиваемыми каталогами Apache может обслуживать виртуальные хосты. Многочисленные доменные имена могут быть обслужены одним и тем же процессом Apache, каждый имея доступ к определенному каталогу. Виртуальные хосты можно определить директивой <VirtualHost>; разместите файлы конфигурации во внутреннем каталоге, например, /etc/apache2/sites-enabled/, или в главном конфигурационном файле. Например, можно их задать вот так:

Listing 8. Конфигурация виртуальных хостов

```
      <VirtualHost "foo.example.com">
          ServerAdmin webmaster@foo.example.com
          DocumentRoot /var/www/foo
          ServerName foo.example.com
          <Directory /var/www/foo>
                Options Indexes FollowSymLinks MultiViews
                AllowOverride None
                Order allow,deny
                allow from all
          </Directory>
          ScriptAlias /cgi-bin/ /usr/lib/cgi-bin/
          <Directory "/usr/lib/cgi-bin">
                AllowOverride None
                Options ExecCGI -MultiViews +SymLinksIfOwnerMatch
                Order allow,deny
                Allow from all
          </Directory>
          CustomLog /var/log/apache2/foo_access.log combined
      </VirtualHost>
      <VirtualHost "bar.example.org">
          DocumentRoot /var/www/bar
          ServerName bar.example.org
      </VirtualHost>
      <VirtualHost *>
          DocumentRoot /var/www
      </VirtualHost>
```

Последняя опция * принимает все HTTP запросы, которые направлены не на один из явно определенных имен (как те, адресуемые IP-адресом или адресуемые неуказанным символьным доменом, что также приводят к машине-серверу). Чтобы

виртуальные хосты работали, DNS должна определять каждый псевдоним с записью CNAME.

Возможность multi-homing по названию похожа на виртуальный хостинг, однако идея здесь совсем другая. Используя multi-homing, можно указывать IP-адреса, с которыми машина соединяется для того, чтобы разрешить web-запросы. Например, можно дать HTTP-доступ только локальной сети LAN, но не для остального мира. Если указывается адрес, от которого ждут сигнала, также можно указать неумолчательный порт. Значение для `BindAddress`, выставленное по умолчанию -- это *, что означает принимать запросы на все IP-адреса, с котрыми связан этот сервер. Пример смешанного использования выглядит примерно вот так:

Listing 9. Конфигурация multi-homing

```
BindAddress 192.168.2.2
Listen 192.168.2.2:8000
Listen 64.41.64.172:8080
```

В этом случае, все клиентские запросы из локальной LAN (использующие адрес 192.168.2.2) будут приняты на порт 80 и специальный порт 8000. Этот дистрибутив Apache также будет отслеживать клиентские HTTP запросы из WAN адреса, но только на потр 8080.

Ограничение доступа к Сети

Командами `Order`, `Allow from`, и `Deny from` в директиве `<Directory>` можно управлять доступом к отдельным каталогам сервера. Запрещенные или разрешенные адреса можно задавать полными или частичными именами хостов или IP-адресами. `Order` позволяет задавать приоритет между списком разрешения и списком запрета.

Часто требуется более тонкий контроль, нежели тот, что задается простым разрешением определенным хостам обращаться к web-серверу. Для подключения требований входа пользователя, используется семейство команд `Auth*`, опять-таки, внутри директивы `<Directory>`. Например, для установке базовой аутентификации можно использовать директиву, как показано в Listing 10.

Listing 10. Конфигурация базовой аутентификации

```
      <Directory "/var/www/baz">
          AuthName "Baz"
          AuthType Basic
          AuthUserFile /etc/apache2/http.passwords
          AuthGroupFile /etc/apache2/http.groups
          Require john jill sally bob
      </Directory>
```

Можно также указать базовую аутентификацию внутри .htaccess-файла. Аутентификация дайджестом более безопасна, нежели базовая, но не так широко реализована в браузерах. Однако слабость базовой схемы (когда пароль передается открытым текстом) в любом случае лучше решается с помощью уровня SSL.

Поддержка SSL-шифрования web-трафика обеспечивается модулем `mod_ssl`. Если используется SSL, данные, передаваемые между сервером и клиентом, шифруются с динамически изменяемым паролем, что является стойким по отношению к перехвату. Все основные браузеры поддерживают SSL. За большей информацией по настройке Apache 2.x с `mod_ssl`, обратитесть к описанию на web-сайте Apache.