Глобальні інформаційні мережі.

*Лекція № **12**.*
ТЕЗИ

Веб-сервер Apache (продовження)

Архітектура apache

httpd – демон apache (якщо не зкомпільовано інакше, на деяких Linux дистрибутивах буває і демон є apache).

Для перевірки правильності конфігурації, для старту та зупинки apache використовується утиліта apachectl. По суті вона просто викликає httpd з різними ключами.
Наприклад
apachectl configtest всього-навсього викликає httpd з ключем -t.

Центральний конфігураційний файл – httpd.conf як правило знаходиться в /etc/httpd/ чи /etc/httpd/conf/ чи /etc/apache/ в Linux системах, і в /usr/local/etc/apache/ в BSD. На нашому сервері напр. /usr/local/etc/apache22/.

Як правило конфіг файл включає в себе часом навіть десятки інших конфіг файлів з допомогою директиви ***Include.*** ПОЯСНИТИ.

Відносна частота внесення змін в конфіг-файл: як правило набагато частіше, ніж наприклад в конфігурацію postfix (postfix вичитує таблиці транспорту, віртуальних хостів чи аліасів без перезавантаження, а apache – ні, кожен новий вірт хост, кожна зміна настройок вимагає перечитування конфігурації сервером (root процес перечитує і вбиває та запускає заново дочірні процеси).

Apache – від слова “patch” – латка.
Apache отже складається з багатьох модулів-латок: ***DSO (Dynamich Shared Modules)*** модулів. Абсолютна більшість функціональності забезпечується модулями.
Директиви ***LoadModule та <IfModule>***

Документація є на студ. сервері, також важливо зауважити, що до кожної директиви вказано, до якого модулю вона відноситься а також ***область її дії (контекст, де директива має сенс і вживається):*** може мати дію тільки всередині інших парних директив, в окремих файлах (.htaccess) або глобальну .
(Ми не розглядатимемо всіх директив, тільки основні з акцентом на функціональності).

Apache - багатозадачний (і багатопотоковий поч. від версії 2) сервер. В режимі демона запускається 1 управляючий процес і решта – запускаються і “вбиваються” ним же.
Всі дочірні процеси працюють від непривілейованого користувача (www, web etc.)
Головний – від root.
Кожен дочірній процес має обслуговувати запити поки не досягнеться якесь число (конф. директива ***MaxRequestsPerChild***) а тоді “прибивається” материнським процесом. Така

архітектура зумовлена зокрема тим, що дочірні процеси можуть накопичувати помилки різного роду як наприклад виділення і незвільнення пам'яті і т.д. (як правило внаслідок виконання сторонніх модулів, таких наприклад, як php5_mod).

Apache можна ще запускати через inetd, але це доцільно робити дуже рідко. ПОЯСНИТИ.

Скільки може бути дочірніх процесів?
Директиви:
ServerLimit 25 - буде запущено не більше ніж 25 дочірніх процесів
StartServers 5 - стільки буде запушено дочірніх процесів зпочатку
MaxClients 250 - буде дозволено обслуговування стількох клієнтів одночасно
MinSpareThreads 25 - min к-сть вільних (запасних) потоків
MaxSpareThreads 75 - max к-сть вільних (запасних) потоків
ThreadsPerChild 25 - стільки запускати потоків в одному дочірньому процесі

Директиви, що використовуються сервером для "самоідентифікації" :
ServerName example.ldi.org.ua
ServerAlias www.example.ldi.org.ua www2.example.ldi.org.ua

Прив'язка до IP адрес і портів

Коли сервер стартує, він запускається по замовчуванню на порті 80 на всіх інтерфейсах. Для зміни цього сліжить директива ***Listen***
Listen 80
Listen 443
або
Listen 192.170.2.1:80
Listen 192.170.2.5:8000

Listen не конфігурує віртуальні хости. Ця директива тільки каже серверу на яких портах і адресах "слухати".
Віртуальні хости конфігуруються всередині парної директиви
<VirtualHost>...</VirtualHost> (про це ще піде мова пізніше). З допомогою цих парних директив сервер може по-різному поводитись в залежності від адреси, порта та заголовку запиту (Host) запиту.
Однак перед тим, як конфігурувати конкретні вірт. хости сервер повинен знати, на яких адресах і портах слухати. В конфігурації самого вірт хоста це НЕ вказується. ***Іншими словами директива Listen має виключно глобальних характер.***
Існує ще директива ***NameVirtualHost*** , яка вказує на якій IP будуть конфігуруватися віртуальні хости **по імені** (dns).

Отже:
Apache може обслуговувати як кілька IP адрес на одній машині так і масу різних доменних імен (сайтів) на одній адресі.
ПОЯСНИТИ:
протоколи HTTP 1.0 та 1.1 (віртуальний хостінг та NameVirtualHost):
історично згідно протоколу HTTP 1.0 сервер обслуговував один сайт (віртуальний хост) на одній IP адресі. Сайт міг мати псевдоніми (aliases), але не могло бути кілька різних сайтів на одній IP.
З введенням HTTP 1.1 стало можливим на одній IP мати як завгодно багато сайтів, очевидно, що під різними dns іменами (Заголовок "Host:"). Аліасінг також можливий.

Тобто маємо дуже гнучку схему адресації, обслуговування сайтів.
Не можливо на одній IP мати лише кілька https-сайтів з різними (валідними) сертифікатами.

Всі вище згадані директиви мали голобальну область дії

Парні директиви, що обмежують область дії інших директив:
<Directory назва>...</Directory>
<DirectoryMatch регулярний_вираз>...</DirectoryMatch>
<Files назва>...</Files>
<FilesMatch регулярний_вираз>...</FilesMatch>
<Location назва>...</Location>
<LocationMatch регулярний_вираз>...</LocationMatch>
<VirtualHost addr[:port] [addr[:port]] ...>...</VirtualHost>

Ще однією областю дії для директив apache є так звані ***.htaccess*** файли.
Увага! Вони НЕ належать до конфігурації сервера і НЕ включаються в конф файл.
Використання .htaccess файлів регулюється директивами:
AccessFileName .htaccess
AllowOverride All|None|directive-type [directive-type] ...
(directive-type: AuthConfig FileInfo Indexes Limit Options)
Ці файли дозволяється поміщати будь-де в дереві директорій веб-сервера. Сервер їх зобов'язаний розбирати та застосовувати якщо AllowOverride не стоїть None.
Очевидно, що в .htaccess дозволяється зовсім мало директив.
Плюси:

- даємо користувачам самим керувати своїм контентом,
- не треба перезавантажувати сервер при змінах в .htaccess
- при зміні назв дирктрій нема потреби міняти конфіг-файл

Мінуси:

- помилки в такім файлі приводять до невідображення контенту ВСЬОГО піддерева
- відносне сповільнення роботи сервера (зараз як правило це не суттєво) оскільки розбір і примінення файлу відбувається за кожним запитом
- безпека – хто може знати ЩО захоче помістити туди користувач.

Однак .htaccess файли зараз є дуже поширеною практикою, багато програмних продуктів для вебу їх використовує (часто для організації єдиної точки входу в програму).