Kapitel 10

Konfiguration av Apache

På Red Hat heter den viktigaste konfigurationsfilen för Apache `/etc/httpd/conf/httpd.conf`. På Debian och liknande system är det `/etc/apache2/apache2.conf`, men de har lite annorlunda system för att välja moduler och sätta upp virtual hosts. Jag börjar med att beskriva konfiguration för Red Hat och liknande system.

10.1 Apachekonfiguration på Red Hat

I princip kan man göra all konfiguration i `httpd.conf`, men vissa saker finns i egna filer i katalogen `/etc/httpd/conf.d/`. Allt som slutar på .conf och ligger där inkluderas från `httpd.conf`. Speciellt bör man lägga märke till att `ssl.conf` ligger där. Jag har nån gång raderat den och lagt till https-saker i `httpd.conf`. Det funkar bra – tills det kommer en uppdatering av Apache. Bättre än att radera `ssl.conf` är att kommentera bort rader i den, eller skapa en tom fil med samma namn. Då råkar du inte ut för att Apache vägrar starta på grund av dubbla ”Listen 443”. Eller så kan du göra som Red Hat vill och lägga allt om https i `ssl.conf` istället för `httpd.conf`.

Det går även att låta `httpd.conf` var helt orörd och ändra allt i `../conf.d/*` Men det kräver att du vet vad direktiven heter, och exakt vad som ska ändras. För en nybörjare är det nog lättare att se och ändra exempel än att skriva själv från en tom fil.

En annan fil som finns i `/etc/httpd/conf.d/` är `php.conf`, men den är enbart för att ladda rätt modul och lägga till hanterare för ändelsen .php och index.php. Den riktiga konfigurationsfilen för PHP heter `/etc/php.ini`. Vad som kan behöva ändras där är ämne för en annan bok. Filen `/etc/httpd/conf.d/php.conf` behöver man sällan ändra i.

Fördelen med att ha en katalog som `/etc/httpd/conf.d/` är att andra paket kan lägga till sin konfiguration där utan att man behöver redigera `httpd.conf`, varken automatiskt eller manuellt. Exempel på andra filer som kan finnas där är `manual.conf`, `nagios.conf`, `python.conf`, `squid.conf`, `webalizer.conf`, `mrtg.conf`, `perl.conf` och `welcome.conf`. Men det beror förstås på vilka paket du har installerat.

10.2 Apachekonfiguration på Debian

Debian är lite annorlunda. Ubuntu gör förstås likadant som Debian. Utöver apache2.conf finns `/etc/apache2/ports.conf` där du anger vilka portar Apache ska lyssna på, ofta 80 och 443. I samma katalog finns även katalogen `conf.d` med filerna `charset`, `localized-error-pages` och `security`. Dessa behöver man sällan ändra i, men du bör känna till att de inkluderas. Och på samma vis som i Red Hat kan andra paket lägga till flera filer i `conf.d`. I `/etc/apache2/` finns även katalogerna `mods-available` och `mods-enabled` för moduler. `sites-available` och `sites-enabled` för virtuella värdar (virtual hosts). Det är enbart katalogerna som heter -enabled som Apache använder, så du kan redigera filerna i -available och sen skapa symlänk i -enabled. Detta system gör det lätt att lägga till och ta bort såväl moduler som virtuella värdar. För att skapa länkarna kan du ägen använda kommandona `a2enmod` och `a2ensite`. För att radera länkarna använder du kommandona `a2dismod` och `a2dissite`.

10.3 httpd.conf

Jag kallar detta delkapitel för `httpd.conf`, som den heter på Red Hat, men motsvarande gäller för `apache2.conf` och de andra filerna på Debian. Här beskriver jag översiktligt hur direktiv och containrar fungerar. Alltså hur du får Apache att göra som du vill.

10.3.1 Direktiv och containrar

Direktiv är ord som betyder något för Apache. Till exempel Listen, MaxSpareServers eller LogLevel. De finns alltid i något sammanhang (omgivning, kontext), antingen för hela servern, en virtuell värd, en katalog eller för vissa filer. Orden som anger sammanhanget kallar jag för containrar, de beskriver vilken sektion som direktiven gäller i.

Containrar anges med <ord plats/sätt>direktiv</ord>. De som finns visar jag i tabell 10.1.

Tabell 10.1: Containrar

Namn	Förklaring
<Directory>	Katalog i filsystemet
<DirectoryMatch>	Katalog, med regexp, i filsystemet
<Files>	Fil(er) i filsystemet
<FilesMatch>	Filer, med regexp, i filsystemet
<Location>	URL på servern
<LocationMatch>	URL-regexp på servern
<Limit>	Åtkomstmetod
<LimitExcept>	Negerad åtkomstmetod
<VirtualHost>	Virtuell värd

Nu behövs nog ett exempel. Antag att du har katalogen `/var/www/html/internt`, och vill att den enbart ska vara åtkomlig från nätet 192.168.32.0/24. Detta kan du göra genom följande rader:

```
<Directory /var/www/html/internt>
Order deny, allow
Deny from all
Allow from 192.168.32.
</Directory>
```

Men eftersom dokumentroten är `/var/www/html` så skulle du kunnat göra samma sak med <Location /internt>. Eller med <Files /var/www/html/internt/*.html> om du enbart vill att HTML-filer, men inte t.ex. bilder, ska vara spärrade för andra. De containrar som heter något med Match använder reguljära uttryck, de andra (Directory, Files och Location) använder jokertecken liknande skalets.

<Limit> och <LimitExcept> anger inte platser utan HTTP-metoder, som GET, HEAD och POST.

<VirtualHost> beskriver jag mer längre fram.
I manualen på
http://httpd.apache.org/docs/2.4/mod/quickreference.html
anges med bokstäver var varje direktiv är tillåtet. s=hela servern, utanför containrar. v=i VirtualHosts, d=inuti Directory, Files, Location och deras respektive *Match. h betyder att de är tillåtna i .htaccess-filer.

Filer som heter `.htaccess` (namnet går att ändra) kan ligga i kataloger och gäller där och i underkataloger. Det är ett sätt att låta användare styra vilka direktiv som ska gälla dem. Administratören bestämmer vilken sorts direktiv som får finnas med AllowOverride. Vissa saker kan medföra säkerhetsproblem, och att servern behöver leta efter `.htaccess` överallt ger minskad prestanda. Så när du inte behöver denna funktion rekommenderar jag AllowOverride None.

10.3.2 Alias, Redirect och Rewrite

Jag har redan nämnt några direktiv (t.ex. Listen, Order, Deny, Allow och AllowOverride). För mer komplett lista se webben. Sajten *httpd.apache.org* återkommer man ofta till, det är svårt att hålla allt i huvudet, och vissa saker förändras (2.4.3 och 2.2.23 är de senaste versionerna av Apache httpd när jag skriver detta).

Några direktiv bör man dock känna till. Detaljerna kan man få slå upp vid behov, men det är lättare att hitta när man vet vad man ska leta efter.

I modulen mod_alias finns bl.a. direktiven Alias och Redirect. Alias är för att ge en resurs flera namn. Det sköts helt på serversidan, man kan jämföra med en länk i filsystemet, men det är för sökvägen i en URL. Syntaxen är:
Alias /nyttnamn /riktigtnamn

Den kan ofta vara bra att använda.

Direktivet Redirect är däremot för klientsidan. Servern skickar status 3xx och ber klienten att begära en annan resurs. 302 betyder tillfälligt flyttad (standardvärdet) och 301 betyder permanent flyttad. Skillnaden är bland annat vad som ska cachas och bokmärkas. Med Redirect kan du tillfälligt stänga av vissa sidor, och hänvisa besökaren till en annan katalog, fil eller server. Allt som går att göra med Redirect går även göra med mod_rewrite (tvärtom är inte fallet), men Rewrite har krångligare syntax, så går det att lösa med Redirect är det att föredra.

Jag får nämna lite mera om mod_rewrite. Det finns flera direktiv, du börjar med RewriteEngine On, sen kan du ha kombinationer av RewriteCond och RewriteRule. Det finns bra HowTo på:

http://httpd.apache.org/docs/2.2/mod/mod_rewrite.html
och
http://httpd.apache.org/docs/2.2/rewrite/

Du bör känna till, eller lära dig, reguljära uttryck för att förstå de mer avancerade detaljerna.

10.3.3 UserDir

Om du vill att användare ska kunna lägga till egna filer, åtkomliga på /~user använder du direktivet ”UserDir public_html” så blir katalogen `/home/ao/public_html/` åtkomlig på *http://server.namn.se/~ao/*. Detta var kanske mer vanligt förr, men används i vissa fall fortfarande. UserDir är avstängt eller bortkommenterat som standard i de flesta distributioner, men är lätt att slå på. Dock behöver du även

köra `setsebool httpd_enable_homedirs on` om du använder SELinux.

10.4 VirtualHost

Det är vanligt att man vill att samma server ska visa olika sidor beroende på vilket namn den anropas som. Det är ett bra sätt att effektivisera användningen. Det numera vanligaste sättet är namnbaserade <VirtualHost>s, men i vissa fall (t.ex. för https) bör man ha separata IP-nummer.

Det kan se ut så här:

```
#ao051031
NameVirtualHost *:80
<VirtualHost *:80>
    ServerName pics.exempel.com
    ServerAlias www.pics.exempel.com
    ServerAdmin root@exempel.com
    DocumentRoot /var/www/pics.exempel.com
    ErrorLog logs/pics.exempel.com-error_log
    CustomLog logs/pics.exempel.com-access_log
       combined
</VirtualHost>
```

Om servern har flera IP-nummer och du enbart vill använda ett av dem skriver du det istället för *, alltså <VirtualHost 192.168.44.20:80>.

Du får enbart ha en rad med ServerName, men du kan ha många ServerAlias.

DocumentRoot för en server kan ligga såväl innanför som utanför DocumentRoot för en annan. Men du bör tänka på vad som ska gå att komma åt via namn respektive med sökväg.

Den första VirtualHost som definieras blir standardalternativet, till exempel om du skriver in IP-nummer i URL:en eller något namn som inte finns i `httpd.conf`. Så lägg den vanligaste först, ofta den som heter www.företagsnamn.se

Du måste förstås definiera alla namn i DNS, det går bra med såväl A-records som CNAME.

Det är bra att ange separata ErrorLog och CustomLog för varje server. Gör du inte det så ärvs de filnamnen från den allmänna serverkonfigurationen. Många andra saker ärvs, till exempel hade jag nog inte behövt ange ServerAdmin. Ändringar som du gör inom VirtualHost-containern är däremot lokala just för den, till exempel Alias, Allow, Deny och så vidare.

Om du har en server med https (port 443) så måste den ha unikt IP-nummer. Vid vanlig HTTP används raden i huvudet som börjar med Host för att avgöra vilken vhost som ska väljas, men vid https är även det krypterat och kan inte avkodas innan valet av VirtualHost. Det finns sätt att kringgå detta, men det stöds inte av alla webbläsare än. Så enklast är att tilldela flera IP-nummer med eth0:0, eth0:1 osv och ange <VirtualHost 192.168.44.21:443> för varje. Detta går dock att kombinera med namnbaserade vhostar för port 80.

På Red Hat brukar jag lägga VirtualHost sist i `httpd.conf`. På Debian är det bättre att använda `sites-available` och symlänk i `../sites-enabled`.