Linux Network Servers

Apache – Parte 1

HTTP

HTTP (acrônimo para Hypertext Transfer Protocol, que significa Protocolo de Transferência de Hipertexto) é um protocolo de comunicação (na camada de aplicação segundo o Modelo OSI) utilizado para transferir dados por intranets e pela World Wide Web.

O servidor Apache é o mais bem sucedido servidor web livre. Foi criado em 1995 por Rob McCool, então funcionário do NCSA (National Center for Supercomputing Applications).

Suas funcionalidades são mantidas através de uma estrutura de módulos, permitindo inclusive que o usuário escreva seus próprios módulos.

Outros servidores HTTP:

O IIS da Microsoft, lighttpd, tomcat, nginx, etc.

Ranking de servidores web utilizados no mundo:

http://news.netcraft.com/archives/web_server_survey.html

Segundo a pesquisa: 50% dos servidores WEB no mundo são Apache.

Esse número ainda pode ser maior, pois vários administradores de redes configuram seus servidores para não mostrar informações verdadeiras sobre a versão do servidor web a fim de não facilitar ataques de crackers.

Linux Network Servers

Apache MPM Worker e Pre Fork

De acordo com a documentação oficial do projeto Apache, é possível configurar o Apache para trabalhar com aplicações mais novas, aumentando a performance ou manter o modo optar por manter a compatibilidade e estabilidade com aplicações que trabalham nativamente com processos.

Estes modos são o MPM Worker e Pre Fork.

O Módulo MPM Pre Fork trabalha utilizando a estrutura clássica de processos Unix, mantendo a compatibilidade com o Apache versão 1.X.

Neste modo, um processo Apache será responsável por executar uma série de processos filhos, para atender as conexões que chegarem.

Já no modo MPM Worker, o Apache trabalhará com uma implementação mista de processos e threads, possibilitando atender uma quantidade maior de processos sem a necessidade de um aumento de hardware, comparado com a configuração MPM Pre Fork.

Neste modo, o Apache mantém um processo controlando uma série de threads, o que otimiza o desempenho para aplicações que suportam threads.

Instalando o Apache:

```
# aptitude install apache2-common apache2-mpm-prefork apache2-utils
```

No Red Hat:

```
# yum install httpd
# service httpd start
```

Observação importante:

Linux Network Servers

No Debian, o apache se chama apache2, enquanto que no Red Hat / CentOS ou derivados, ele se chama httpd.

O Apache já vai estar pronto para ser acessado!

* Abra um navegador e acesse http://127.0.0.1, você entrará no Apache de sua máquina.

* Verificando se o servidor está no ar:

```
netstat -nptl | grep 80
```

A porta padrão do protocolo HTTP é a 80!

* Parando e iniciando:

```
/etc/init.d/apache2 stop
/etc/init.d/apache2 start
```

A função do apache por padrão é fornecer páginas html ou outros arquivos através de download.

Se você quer colocar páginas com script PHP, você deve acionar o módulo para isso.

Vamos agora dar uma olhada nos arquivos de configuração.:

```
cd /etc/apache2
ls

apache2.conf <- arquivo principal de configuração
httpd.conf <- compatibilidade com Apache 1
```

Linux Network Servers

ports.conf <- portas

sites-available e sites-enabled <- guardar virtual hosts

mods-available e mods-enabled <- armazenam configurações de módulos

Na plataforma Red Hat, todos as configurações são feitas dentro de /etc/httpd/conf/httpd.conf.

É hora de vermos como é o arquivo de configuração principal, é onde falamos para o Apache onde estão todas nossas configurações

* Por segurança, faça uma cópia:

```
cp apache2.conf apache2.conf.orig
```

* Abra ele agora:

```
vim apache2.conf
```

Vamos conhecer alguns parâmetros:

Timeout 300

Serve para que seu Apache não fique eternamente esperando a resposta de um cliente. Quando alguém acessa o seu site, vocês passam a trocar informações, conforme o seu cliente vai acessando páginas, seu apache vai disponibilizando o site esse tempo é o tempo máximo que o seu server vai esperar a resposta do cliente. Esse cliente nada mais é do que o browser de alguem acessando seu site.

Caso esse alguém fique boiando e não peça mais nada para o server, passou o

Linux Network Servers

tempo definido e ele descarta a conexão.

O MPM é o Multi Processing Modules, ou seja, os módulos de multiprocessamento, que nada mais é do que como irá funcionar o processamento do apache em relação aos acessos no site. Na versão 1 do apache o mesmo só usava um modo de processamento, que era baseado em prefork. Ou seja, o apache assim que iniciava, fazia um fork onde gerava os processos filhos para que esses ficassem responsáveis pelas solicitações ao site. E fork nada mais é do que o processo pai ao iniciar já criar processos filhos.

Mas a questão de processos funciona muito bem no Linux :

<IfModule mpm_prefork_module>

StartServers 5

MinSpareServers 5

MaxSpareServers 10

MaxClients 150

MaxRequestsPerChild 0

</IfModule>

StartServers <- quantidade inicial de processos spare = reserva spare = sem fazer nada MinSpareServers <- pelo menos 5 servidores carregados MaxSpareServers <- maximo de servidores reserva aguardando MaxClients <- quantas conexões são aceitas, quando o apache chega nesse limite, ele começa a ficar lento colocando as outras requisições em uma fila

User www-data

Linux Network Servers

```
Group www-data
```

Essas linhas indicam quem é o usuário e grupo dono dos processos do apache. Na versão antiga do apache, o usuario e grupo dono dos processos era o httpd. Nessa versão nova é o www-data.

```
ErrorLog /var/log/apache2/error.log
cd /etc/apache2
cat ports.conf
Listen 80
```

Nesse arquivo só temos que definir a porta que o apache vai abrir para aceitar requisições.

```
cd /etc/apache2
cd sites-available
vim default
DocumentRoot /var/www/
```

Essa linha é muito importante. É aqui que fala onde estará o seu site e é aqui que o apache vai procurar um index (sua página).

Vamos colocar um arquivo lá:

```
cp /etc/resolv.conf /var/www
```

E agora acesse http://127.0.0.1/resolv.conf.

Linux Network Servers

Vamos agora renomear o arquivo index :

```
cd /var/www; mv index.html index2.html
```

Acessem http://localhost/

Vamos voltar para o arquivo **default**, temos essa linha:

```
RedirectMatch ^/$ /apache2-default/
```

Explicando a expressão regular:

```
^ = começo de linha
$ = fim de linha
http://127.0.0.1/ <- o " / " vai casar
```

Comente a linha:

```
#RedirectMatch ^/$ /apache2-default/
```

Abra o arquivo /etc/apache2/conf.d/security e modifique as configurações para que o apache mostre a menor quantidade possível de informações ao usuário:

```
# vim /etc/apache2/conf.d/security
ServerSignature Off
ServeTokens Prod
TraceEnable Off
```

Reinicie o apache:

```
/etc/init.d/apache2 restart
```

Linux Network Servers

Acesse de novo:

```
http://127.0.0.1/
```

Listou o diretório.

O que é um virtual host? Qual a finalidade disso?

Ele permite hospedar vários sites, com domínios diferentes, usando um único servidor e um único endereço IP.

Qual a desvantagem disso?

Não se pode ter um volume muito grande de sites, pois os recursos como hardware e banda são limitados.

São configurados nestes dois diretórios:

/etc/apache2/sites-available: Neste diretório ficam todos os arquivos de configuração dos domínios virtuais.

/etc/apache2/sites-enabled: Neste diretório ficam todos os domínios virtuais ativos, que na verdade são links simbólicos para os arquivos de configuração localizados no diretório citado anteriormente.

Sites available são as configurações disponíveis e sites enables, as ativas.

Criando um Virtual Host

```
vim /etc/apache2/sites-available/teste-ht.com.br
```

Linux Network Servers

* Dentro coloque esse conteúdo:

```
NameVirtualHost *
<VirtualHost *>
DocumentRoot /var/www/teste-ht
ServerName www.teste-ht.com.br
ErrorLog /var/log/apache2/teste-ht-error.log
CustomLog /var/log/apache2/teste-ht-access.log common
</VirtualHost>
```

* Criar o DocumentRoot:

```
mkdir /srv/teste-ht
cd /srv/teste-ht
echo "<html><body>Oi mundo</body></html>" > index.html
```

* Ativar o site:

```
a2ensite teste-ht.com.br
```

a2 = apache2

en = enable

site = site

* Verifique a sintaxe:

```
apache2ctl -S
```

Syntax OK

* Reconfigure o Apache:

```
/etc/init.d/apache2 reload
```

Linux Network Servers

* Agora acesse:

```
http://www.teste-ht.com.br
```

* Só para garantir:

```
vim /etc/hosts
127.0.0.1 www.teste-ht.com.br
```

O NameVirtualHost diz ao apache que ele deve mapear todos os IPs do servidor para os hosts virtuais abaixo.

Instalando o PHP

PHP (um acrónimo recursivo para "PHP: Hypertext Preprocessor") é uma linguagem de programação de computadores interpretada, livre e muito utilizada para gerar conteúdo dinâmico na Web, como por exemplo a Wikipédia. O PHP é uma poderosa linguagem orientada a objetos também.

* Instalando:

```
aptitude install libapache2-mod-php5 php5
```

No Red Hat:

```
# yum install php
```

Veremos o básico do básico.

O PHP funciona como um módulo do apache. Para vermos os módulos que temos disponíveis usamos o comando:

```
aptitude install libapache2-mod-php5 php5
apache2ctl -M
```

Linux Network Servers

Verifiquem se o php5 está na lista. O debian já instala pra gente sem muito esforço. Praticamente todas as distros fazem.

O que se tem que tomar cuidado é o seguinte: com o módulo PHP seu apache pode executar praticamente qualquer código. Ele vira um servidor de aplicação.

Aquele arquivo .html que criamos é estático, vamos criar um dinâmico.

```
vim /srv/teste-ht/teste.php
```

Esse arquivo vai ficar assim:

```
<html>
<body>
<?
$variavel = "oi mundo";
echo $variavel;
?>
</body>
</html>
```

Salve ele e acesse no navegador:

http://www.teste-ht.com.br/teste.php

```
# vim /var/www/index.php
<?
phpinfo ()
?>
```

Linux Network Servers

A linguaguem PHP é grande e tem muitas funcionalidades. Uma delas é pode rodar comandos no sistema, o que é extremamente perigoso.

Como desabilitar funcionar perigosas

Editem esse arquivo:

```
vim /etc/php5/apache2/php.ini
```

Aproximadamente na linha 222.

A arquivo php.ini configura todo o comportamento do PHP dentro do apache.

Por exemplo, no PHP existe a função "exec".

exec faz exatamente o que diz: executa qualquer comando na máquina.

Isso é ruim.

Não pode ser permitido para uma aplicação web em hipótese alguma,

pois muitas aplicações são bugadas como blogs, forums, etc, e permitem a injeção de comandos por formulários.

Então, temos que desabilitar o exec.

```
disable_functions = exec
```

Deixe assim. Salve, reinicie o apache. Agora, edite o teste.php. O código PHP fica sempre entre <? ?>.

No teste.php coloque:

```
exec("ls /etc");
```

Linux Network Servers

Sempre dentro dos <? ?>.

Rode o teste.php no browser.

http://www.teste-ht.com.br/teste.php

Warning: exec() has been disabled for security reasons in /srv/teste-ht/teste.php on line 4

Outras opções pode ser desabilitadas por medida de segurança no arquivo php.ini:

```
show_source
system
shell_exec
passthru
popen
symlink
```

expose_php = Off -> Essa opção está relacionada mais com privacidade do que com a segurança propriamente dita. Com expose_php habilitada, será possível determinar se o PHP está instalado no servidor. Isso pode ser feito com uma consulta ao servidor web.

Habilitando essa diretriz, a versão do PHP também ficará exposta. Em um caso extremo de haver algum exploit com alguma versão em específico do PHP, essa informação pode ser valiosa para pessoas mal intencionadas.

Exemplo:

$ telnet localhost 80 Trying 127.0.0.1... Connected te localhost. Escape character is ‘A]’. HEAD / HTTP/1.0

Linux Network Servers

HTTP/1.1 200 OK Date: Sun, 12 Aug 2007 01:33:33 GMT Server: Apache/2.2.3 (Distro) PHP/5 .2.1 X-Powered-By: PHP/5.2.1 Connection: close Content-Type: text/html; charset=IS0-8859-1

Connection closed by foreign host.

register_globals = Off -> Essa opção não representa um problema de segurança no PHP, porém pode proporcionar potenciais riscos à segurança do sistema por induzir o desenvolvedor ao erro. Não existe uma boa razão para habilitar essa diretriz e, de modo geral, register_globals deve permanecer sempre com o valor Off.

A partir da versão 4.2 do PHP, essa diretriz foi desabilitada no arquivo PHP.INI padrão e até então essa pequena alteração tem gerado muitas dores de cabeça para desenvolvedores PHP que mantêm scripts antigos, pois é muito comum encontrar códigos que foram escritos sem nenhuma preocupação com relação ao registro de variáveis.

Essas opções abaixo permitem abrir ou processar uma pagina ou arquivo externo dentro do script php. Vários servidores de hospedagem desativam essa opção.

```
allow_url_fopen = Off
allow_url_include = Off
```

Habilitar uso de um Banco de Dados

No Debian:

```
# aptitude install php5-mysql
```

No Red Hat:

```
# yum install php-mysql
```

Linux Network Servers

Se for postresql:

```
# aptitude install php5-pgsql
```

Obs: Claro que já devo ter instalados os SGBDS.

Instalando o Mysql

```
# aptitude install mysql-server
```

Red Hat / CentOS / Fedora:

```
# yum install mysql mysql-server
```

Para iniciar o banco de dados:

```
# mysql_install_db
```

Esse comando serve para criar a base de dados.

Próximo passo:

```
# /etc/init.d/mysql start
```

Fedora/CentOS:

```
# service mysqld start
```

Vamos definir a senha de root do banco de dados:

```
# mysqladmin -u root password <senha>
```

Linux Network Servers

Para trocar a senha depois o comando é:

```
# mysqladmin -u root -p password <senha>
```

Para acessar o banco como administrador:

```
# mysql -u root -p <enter>
```

e digite a senha.

```
mysql>
```

Criando um BD

```
mysql > CREATE DATABASE teste;
```

Se deu certo vai aparecer a seguinte mensagem:
Query OK, 1 row affected (0.00 sec)

Para ver se o banco foi criado:

```
mysql > SHOW DATABASES;

+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| teste              |
+--------------------+
3 rows in set (0.00 sec)
```

Linux Network Servers

```
mysql > quit
```