
Revisão do Tópico 208 – Web Services

208.1 – Implementando um Servidor Web (Apache)

Arquivo de Configuração

Principal: /etc/httpd/conf/httpd.conf
Configurações extras: /etc/httpd/conf.d/

O pacote do apache em distribuições baseadas em Debian utilizam como principal arquivo de configuração o /etc/apache2/apache2.conf, mas a LPI e a documentação do Apache HTTPD mencionam o httpd.conf como sendo o arquivo de configuração principal.

As principais configurações do httpd.conf são:
- **ServerRoot** – Diretório base do apache, em geral o /etc/httpd/
- **Listen** – IP/Porta em que o Apache escutará as conexões
- **ServerAdmin** – Define um e-mail de administrador do servidor
- **ServerName** – Nome/Endereço e Porta que o Apache utiliza para se identificar
- **DocumentRoot** – Diretório em que se encontram os arquivos que serão servidos pelo servidor web

* Outras configurações serão estudadas a seguir

Processos e Comandos

Processo httpd

O processo padrão do Apache HTTPD é o **/usr/sbin/httpd**. Esse padrão é utilizado em distribuições baseadas em RedHat.
No Debian o processo é o /usr/sbin/apache2.

Tanto o “httpd” quanto o “apache2” disponibilizam uma série de opções, entre elas:
- -v : Exibir a versão do Apache
- -V : Exibir as configurações definidas na compilação do Apache
- -l : Exibir os módulos compilados com o Apache

Comando apache2ctl (apachectl)

Principal comando para administração do Apache HTTPD. Entre suas opções temos:
- status/stop/start/restart
- graceful – Reinicia o apache aguardando que as conexões ativas terminem
- graceful-stop – Pára o apache aguardando que as conexões ativas terminem
- configtest – Verifica se há erros de sintaxe nas configurações

---

MPM (Multi-Processing Modules) – Gerenciamento de Processos e Performance

O MPM determina a forma como o Apache gerencia seus processos e as requisições que recebe.

As 3 principais opções são:

- **Prefork**: Nesse modo, o Apache possui um processo pai para gerenciamento e vários sub-processos para tratamento das requisições. Cas sub-processo trata uma requisição por vez.
- **Worker**: Assim como no prefork, nesse modo há um processo pai e vários sub-processos. A diferença é que cada sub-processo possui diversas threads e cada thread trata uma requisição por vez.
- **Event**: Muito similar ao worker, mas consegue gerenciar melhor as conexões dentro das threads.

Os principais parâmetros de configuração que podem ser introduzidos no httpd.conf para configurar o funcionamento do MPM são:

- **StartServers** – Número de instâncias do apache (processos-filho) iniciados ao se iniciar o apache.
- **MaxClients** – Limite de conexões simultâneas que o Apache tratará.
- **MaxRequestsPerChild** – Número de requests que uma instância tratará antes de morrer.
- **MinSpareServers** (prefork) – Quantidade mínima de processos extras que serão mantidos pelo Apache
- **MaxSpareServers** (prefork) – Quantidade máxima de processos extras que serão mantidos pelo Apache
- **MinSpareThreads** (worker/event) – Quantidade mínima de threads extras que serão mantidos pelo Apache
- **MaxSpareThreads** (worker/event) – Quantidade máxima de threads extras que serão mantidos pelo Apache
- **ThreadsPerChild** (worker/event) – Quantidade de threads em cada processo-filho.

Módulos

O Apache é uma aplicação modular, ou seja, recursos podem ser incorporados através de módulos externos.

Esses módulos ficam normalmente localizados no diretório **/etc/httpd/modules** e possuem nomes que seguem o padrão **mod_*.so**.

Os módulos são carregados no Apache através do parâmetro LoadModule, como no exemplo:

- **LoadModule** php5_module /etc/httpd/modules/libphp5.so

Dentre os módulos mais utilizados temos o mod_php, para suporte à linguagem PHP, e o mod_perl, para suporte à linguagem Perl. Nesses casos, as principais configurações são:

PHP:

```
# Faz o carregamento do módulo
LoadModule php5_module /etc/httpd/modules/libphp5.so
```

```
# Define o Handler (processo interno) que tratará arquivos que terminem com .php
<FilesMatch \.php$>
    SetHandler application/x-httpd-php
</FilesMatch>

# Inclui arquivos .php como tipo “text/html”. Também chamado de “Content Type” ou “Mime Type”
AddType text/html .php

# Inclui os arquivos de nome index.php na lista de possíveis Index considerados pelo Apache
DirectoryIndex index.php
Perl:

# Faz o carregamento do módulo
LoadModule perl_module modules/mod_perl.so

Alias /perl /var/www/perl
<Directory /var/www/perl>
    SetHandler perl-script                    # Define o Handler
    PerlResponseHandler ModPerl::Registry
    PerlOptions +ParseHeaders
    Options +ExecCGI                          # Permite a execução através de CGI
</Directory>
```

Configurações e Arquivos de Log

Os diretórios padrão de logs do Apache são o /var/log/httpd ou o /etc/httpd/logs, que normalmente é um link para o primeiro.

As seguintes configurações são adicionadas ao httpd.conf para definir como será o registro de logs:

- **ErrorLog** – Define o arquivo em que serão inseridos os registros de erro
  - ErrorLog "logs/error_log"
- **LogLevel** – Define o nível das mensagens do ErrorLog
  - LogLevel debug
  - Os valores possíveis são: debug, info, notice, warn, error, crit, alert, emerg
- **LogFormat** – Define um formato de log de acessos. Define o que será registrado
  - LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\"" combined
- **CustomLog** – Define o arquivo em que será registrado o formato definido no LogFormat
  - CustomLog "logs/access_log" combined

Controle de Acesso

Os recursos de controle de acesso também é realizado através de módulos. Os principais deles são:

- mod_auth_basic – Possibilita autenticação através de usuários e senhas
- mod_authz_core – Implementa uma base para autenticação e controle de acesso. Normalmente usado em conjunto com outros módulos
- mod_authn_file – Autenticação através de arquivos texto com senhas
- mod_access_compat – Controle de acesso através de IP e Hostname
- mod_authz_host – Controle de acesso através de IP e Hostname

Controle de Acesso por Usuário e Senha:

Geração do Arquivo de Senhas:

- # htpasswd -s -c .htpasswd usuario
  - -s : Uso do SHA1
  - -c: Cria um novo arquivo. Deve ser omitido para inserir novos usuários no mesmo arquivo
- Sem opções o htpasswd irá alterar a senha do usuário. Exemplo:
  - # htpasswd .htpasswd usuario

Configurações do httpd.conf. Exemplo:

```
<Directory /var/www/html >
   Options Indexes FollowSymLinks
   AllowOverride all                    # Permite o uso do arquivo .htpasswd
   Require all granted
</Directory>

<Directory /var/www/html/topsecret>
   AuthType Basic
   AuthName "Area Secreta de Acesso"
   AuthUserFile /var/www/html/topsecret/.htpasswd
   AuthGroupFile /var/www/html/topsecret/.htgroup
#    Require group suporte                # Exige que o usuário faça parte do grupo suporte
                                         # definido no arquivo .htgroup
   Require valid-user                    # Exige que o usuário exista no arquivo .htpasswd
#    Require user ricardo aluno1          # Libera acesso apenas para os usuários especificados
</Directory>
```

Controle de Acesso por IP/Rede/Host

Configurações do httpd.conf. Exemplo:

Modo “atual”, pelos módulos mod_authz_host e mod_authz_core.

```
<Directory /var/www/html/admin>
    Require ip 192.168.1.210
    Require ip 192.168.1.1
    Require ip 172.16
    Require ip 10.0.0.0/16
    Require host dominioexemplo.com.br
</Directory>
```

Modo “antigo”, pelo módulo mod_access_compat

```
<Directory /var/www/html/admin>
    Order Deny,Allow
    Deny from all
    Allow from 192.168.1.210
    Allow from 192.168.1.1
</Directory>
```

Redirecionamentos

- **Redirect**
  - O Apache diz ao cliente requisitar outra URL.
  - Redirecionamento do lado do cliente (client-side)
  - Exemplo: Redirect /google http://www.google.com.br
  - Exemplo: Redirect /antigo.html http://www.dominioexemplo.com.br/teste.php
- RedirectMatch – O mesmo que o Redirect mas possibilita o uso de expressões regulares
- **Alias**
  - Redirecionamento é feito internamente no servidor (server-side)
  - Pode levar a um diretório fora do DocumentRoot
  - Exemplo: Alias /docs /var/www/docs
- AliasMatch – O mesmo que o Alias mas possibilita o uso de expressões regulares

---

UserDir

Possibilita que cada usuário do servidor possua uma área que será disponibilizada pelo Apache.

Configuração no httpd.conf:
UserDir public_html

Nesse caso, os arquivos devem estar localizados no diretório:
/home/user/public_html/

O acesso será feito pela URL:
http://example.com/~user

VirtualHost

Possibilita a hospedagem de diversos sites no mesmo servidor.

O site/host virtual pode ser identificado pelo IP e pelo Nome:

- **Baseado no IP:** Cada IP da máquina é associado a um conteúdo web, a um DocumentRoot.
- **Baseado no Domínio:** Nesse caso, cada URL é associada a um conteúdo. Para isso o servidor Web identifica o domínio através do request feito pelo cliente (navegador). Modo mais comum de uso.

Exemplo de Configuração Baseada em IP:

Listen 192.168.1.100:80
Listen 192.168.1.200:80

<VirtualHost 192.198.1.100>
    DocumentRoot /var/www/html/teste1
</VirtualHost>

<VirtualHost 192.198.1.200>
    DocumentRoot /var/www/html/teste2
</VirtualHost>

Exemplo de Configuração Baseada em Nome:

Listen 192.168.1.100:80

<VirtualHost 192.198.1.100>
    ServerName www.teste1.com.br
    DocumentRoot /var/www/html/teste1
</VirtualHost>

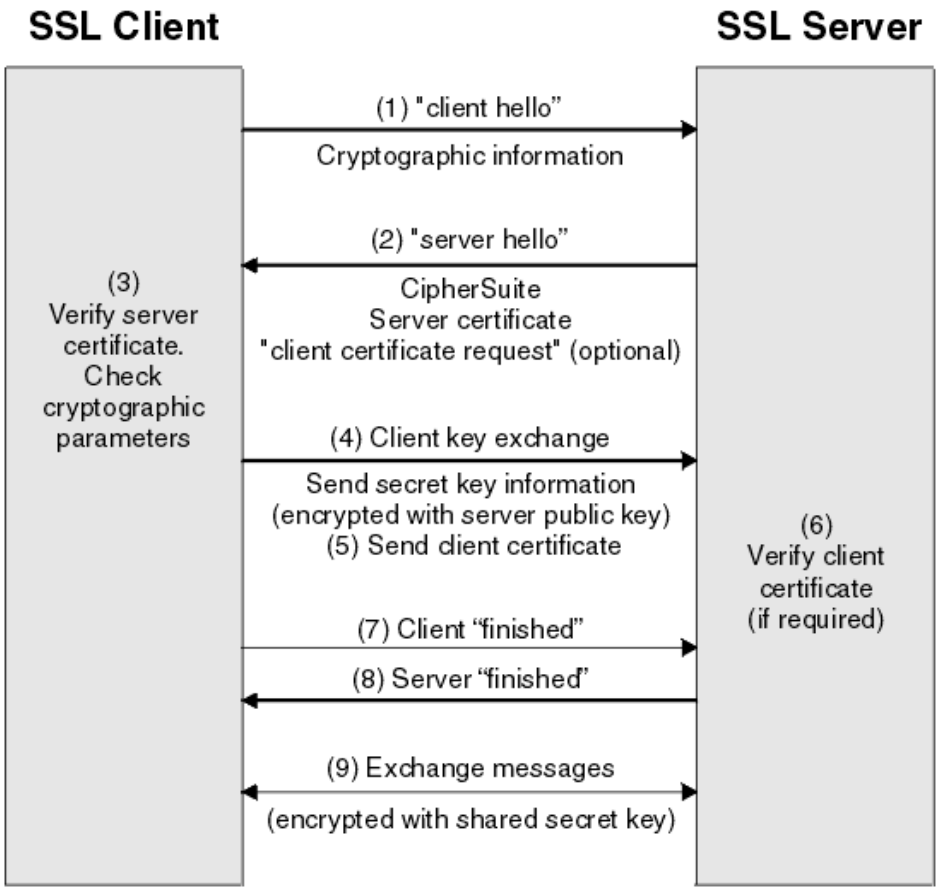
<VirtualHost 192.198.1.100>
    ServerName www.teste2.com.br
    DocumentRoot /var/www/html/teste2
</VirtualHost>

208.2 – Configurações do Apache para HTTPS

Implementação de protocolos de criptografia para encriptação e autenticidade de acesso em sites HTTP. Protocolos:

- SSL: Secure Socket Layer
- TLS: Transport Layer Security

SSL/TLS Handshake:



Fonte: https://www.ibm.com/support/knowledgecenter/en/SSFKSJ_7.1.0/com.ibm.mq.doc/sy10660_.htm

---

Gerando o Certificado

A geração do certificado digital que será instalado em um domínio pode seguir os seguintes passos:

1. Geração da Chave Privada

É comum utilizar o diretório /etc/ssl/certs para armazenar as chaves e certificados gerados.

O primeiro passo é gerar um a chave privada RSA:
openssl genrsa -de3 -out dominio.key 1024

2. Geração do CSR (Certificate Signing Request)

Nessa passo geramos o arquivo CSR.

openssl req -new -key dominio.key -out dominio.csr

Esse arquivo será enviado ao CA (Certification Authority), que responderá com o certificado (crt) que será instalado no servidor web.

3. Assinando o Próprio Certificado

Em ambientes de testes, em vez de utilizar um CA para a geração de um certificado, pode ser feito um procedimento interno, que simule um CA, assinando e gerando o próprio certificado.

Uma das formas de se fazer isso é pelo script /etc/pki/tls/misc/CA.pl, pelos seguintes passos:

```
# cd /etc/pki/CA
# /etc/pki/tls/misc/CA.pl -newca
# cp /etc/ssl/certs/dominio.csr newreq.pem
# /etc/pki/tls/misc/CA.pl -signreq
# cp newcert.pem /etc/ssl/certs/dominio.crt
```

Instalando e Configurando o Certificado

As configurações SSL devem estar sempre dentro de um VirtualHost.

Os principais parâmetros são:

- Listen 443 – Habilitar o servidor a escutar a porta 443
- SSLEngine on - Habilita o recurso de SSL
- SSLCertificateFile – Arquivo do certificado digital para o domínio
- SSLCertificateKeyFile – Arquivo da chave privada utilizada para o certificado

Exemplo de Configuração:

```
<VirtualHost 192.168.1.220:443>
    ServerName www.dominioexemplo.com.br
    DocumentRoot /var/www/html
    Redirect /antigo.html /teste.php
    SSLEngine on
    SSLCertificateFile /etc/ssl/certs/dominioexemplo.com.br.crt
    SSLCertificateKeyFile /etc/ssl/certs/dominioexemplo.com.br.key
</VirtualHost>
```

Certificado de Cliente

É possível exigir que o cliente possua um certificado específico para que possa acessar determinado conteúdo dentro do site.

Nesse caso o certificado deve ser gerado através dos seguintes passos:

1. Geração da Chave Privada

openssl genrsa -de3 -out usuario.key 1024

2. Geração do CSR (Certificate Signing Request)

openssl req -new -key usuario.key -out usuario.csr

3. Assinando o Próprio Certificado

cd /etc/pki/CA
cp /etc/ssl/certs/usuario.csr newreq.pem
/etc/pki/tls/misc/CA.pl -signreq
cp newcert.pem /etc/ssl/certs/usuario.crt

4. Gerar o Arquivo PKCS12, contendo a chave privada e o certificado:

openssl pkcs12 -export -inkey usuario.key -in usuario.crt -out usuario.p12

5. Importar o arquivo usuario.p12 no navegador

6. Configurar os seguintes parâmetros no httpd.conf:

- SSLCACertificateFile – Indica o certificado do CA
- SSLCACertificatePath (opcional) – Diretório que contenha os certificados do CA local
- SSLVerifyCliente require – Exige um certificado do cliente

Exemplo de Configuração:

```
<VirtualHost 192.168.1.220:443>
    ServerName www.dominioexemplo.com.br
    DocumentRoot /var/www/html
    SSLEngine on
    SSLCertificateFile /etc/ssl/certs/dominioexemplo.com.br.crt
    SSLCertificateKeyFile /etc/ssl/certs/dominioexemplo.com.br.key
    SSLCACertificateFile /etc/ssl/certs/cacert.pem
    <Directory /var/www/html/cert>
        SSLVerifyClient require
    </Directory>
</VirtualHost>
```

Configurações de Segurança e SSL

O Apache possui uma série de configurações relacionadas tanto à segurança geral do servidor, como ao uso dos protocolos SSL/TLS.

As principais são:

- **SSLProtocol** – Define as versões suportadas para os protocolos SSL e TLS
- **SSLCipherSuite** – Define os ciphers (algoritmos) suportados no Handshake
- **SSLHonorCipherOrder** – Define se a ordem dos ciphers definido no SSLCipherSuite deve ser seguida
- **ServerTokens** – Define o nível de detalhes quanto à versão do Apache e seus módulos que pode ser disponibilizada. “Prod” ou “ProductOnly” é o menor (e recomendado) nível.

- **ServerSignature** – Define se será exibido um rodapé com informações do servidor em mensagens de erro.
- **TraceEnable** – Define se será possível utilizar o recurso do TRACE nas requisições

---

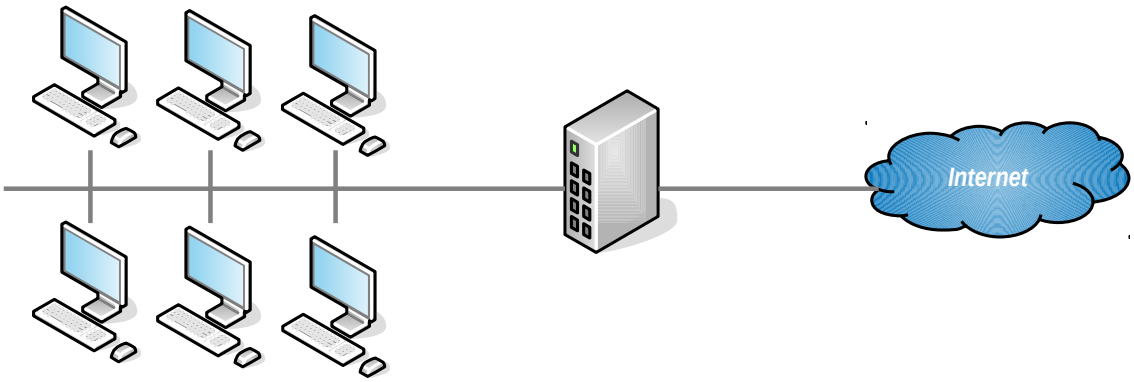
SNI (Server Name Indication)

É uma extensão do SSL/TLS que permite o uso de VirtualHost em conexões HTTPS.

O recurso deve estar habilitado tanto no cliente (navegador) que irá enviar os dados da URL no Handshake SSL/TLS, quanto no servidor web.

208.3 – Implementando um Servidor Proxy (Squid)

Um proxy é um intermediário nas conexões entre os clientes e a Internet.



Squid

O Squid é o principal servidor proxy para conexões HTTP e HTTPS, podendo também atuar com outros protocolos.

Arquivo de Configuração: **/etc/squid/squid.conf**

Arquivos de Logs: /var/log/squid/

Principais Comandos:
- # squid -v : Versão
- # squid -k reconfigure : Relê as configurações
- # squid -z : Recria os diretórios de cache

Configuração de Porta no squid.conf
- **http_port** 3128

Configurações de Cache

O seguinte parâmetro no squid.conf implementa o recurso de cache:

cache_dir ufs /var/spool/squid 100 16 256

Nessa configuração temos que:
- ufs : Estrutura de armazenamento dos arquivos do cache
- /var/spool/squid : Diretório em que o cache será armazenado
- 100 : Quantidade máxima, em MB, de armazenamento
- 16 : Quantidade de diretórios no primeiro nível. Nomes em formato hexa
- 256 : Quantidade de subdiretórios em cada diretório. Nomes em formato hexa

---

Controle de Acesso

Para definição das regras de controle de acesso são utilizadas acls, definidas no seguinte formato:

acl <nome-da-acl> <tipo> <conteúdo>

Os tipos possíveis de acl são:

- src : IP/Rede de Origem
- dst : IP/Rede de Destino
- port : Porta ou lista de portas
- srcdomain : Domínio de origem
- dstdomain : Domínio de destino
- time : Dia da Semana e Horário
  - Exemplo: time MTWHF 12:00-14:00
  - S = Domingo
  - M = Segunda
  - T = Terça
  - W = Quarta
  - H = Quinta
  - F = Sexta
  - A = Sábado
- proto : Protocolo utilizado
- browser : Navegador utilizado
- url_regex : String ou expressão presente na URL
- urlpath_regex : String ou expressão presente na URL, excluindo-se o endereço do domínio

Exemplos de acls:

```
acl localnet src 10.0.0.0/8
acl SSL_ports port 443
acl horario-comercial time MTWHF 09:00-18:00
acl entretenimento url_regex futebol esporte novela
acl proibidos url_regex -i "/etc/squid/proibidos.txt"
acl dominios dstdomain .facebook.com .globoesporte.com .windows.com
```

Após definidas as acls, elas devem ser utilizadas pelo parâmetro http_access, no seguinte formato:

http_access allow/deny <acl>

Exemplos de regras:

```
http_access deny entretenimento !horario-comercial
http_access deny proibidos
http_access allow dominios
http_access allow localnet
http_access deny all
```

Autenticação

O Squid pode realizar a autenticação de usuário de várias formas, através de um simples arquivo com usuários e senhas, através de banco de dados, LDAP, e etc.

Na LPIC-2 pede-se apenas a configuração mais simples, através de arquivos texto gerados pelo htpasswd.

Definindo os Parâmetros da Autenticação

O realm define a mensagem a ser exibida
auth_param basic realm Por favor identifique-se para conseguir acesso

Define o modo de autenticação. Nesse caso também é indicado o arquivo que contém os usuários e senhas. Esse arquivo é criado pelo comando htpasswd, da mesma forma como foi criado para o controle de acesso no Apache.
auth_param basic program /usr/lib64/squid/basic_ncsa_auth /etc/squid/passwords

Criando as Regras para Autenticação

Criação da acl do tipo proxy_auth
acl autenticacao **proxy_auth REQUIRED**

Uso da acl
http_access allow autenticacao

208.4 – Implementando o NGINX como WebServer e Proxy Reverso

O NGINX é o servidor web que mais cresce em uso no mercado principalmente devido à sua excelente performance em ambiente com muito tráfego.

Além de um Servidor Web completo, ele é muito utilizado como Proxy Reverso e para Balanceamento de Carga.

Principal Arquivo de Configuração: **/etc/nginx/nginx.conf**

Principais Comandos:
- # nginx -t : Verifica a sintaxe no arquivo de configuração
- # nginx -s reload : Recarrega as configurações

Principais Configurações

O nginx.conf é estruturado através de blocos. As configurações referentes ao servidor HTTP são localizadas no **bloco httpd { }** e configurações específicas de cada servidor dentro do **bloco server { }**.

Os principais parâmetros de configuração são:
- listen : IP e Porta que o servidor escutará
- server_name : Nome de domínio ao qual o servidor pode responder
- root : Diretório que contém os arquivos que serão servidos
- index : Lista de arquivos considerados como Index

Exemplo de configuração:

```
http{
   ...
   server {
      listen       80;
      listen       [::]:80;
      server_name  www.dominioexemplo.com.br dominioexemplo.com.br;
      root         /var/www/html;
      index   index.html index.html index.php

      location / {
      }
}
```

Configuração do FastCGI para PHP

O FastCGI é um protocolo que faz uma interface entre o servidor web e programas ou scripts em PHP, C, Perl e etc.

Diferente do Apache que trabalha diretamente com o mod_php, o NGINX precisa de um processo a parte para servir os conteúdos desses programas. No caso do PHP o processo é o php-fpm, que roda na porta 9000 ou através de um socket.

Configurações utilizadas:

- **fastcgi_pass** : Indica o IP/Porta ou Socket para o qual os requests PHP serão encaminhados
- **fastcgi_param** : Determina parâmetros que serão enviados ao PHP

Exemplo de configuração:

```
location ~ \.php$ {
    fastcgi_pass 127.0.0.1:9000;
    fastcgi_param QUERY_STRING $query_string;
    fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
    include /etc/nginx/fastcgi_params;
}
```

Configuração de Proxy Reverso

O NGINX pode ser configurado para encaminho todo ou parte do tráfego para outro servidor, como um outro NGINX, Ou um Apache, Tomcat, Jboss e etc.

Para isso são utilizados os seguintes parâmetros:

- **proxy_pass** : Determina o IP/Porta para o qual o request será encaminhado
- **proxy_set_header** : Adiciona e faz alterações no cabeçalho dos requests antes de encaminhá-los ao destino

Exemplo de configuração:

```
location / {
    proxy_pass http://192.168.1.210:8080
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $remote_addr;
    proxy_set_header Host $host;
}
```