

# 第 15 章

## Nginx 的邮件模块

Nginx 除了可用作 HTTP 服务器外，还可用作邮件代理，支持 IMAP、POP3、SMTP 协议。

IMAP (Internet Message Access Protocol) 是一种用于邮箱访问的协议，使用 IMAP 协议可以在客户端管理服务器上的邮箱。它与 POP 不同，邮件是保留在服务器上而不是下载到本地，在这一点上 IMAP 是与 Webmail 相似的。但 IMAP 有比 Webmail 更好的地方，它比 Webmail 更高效和安全，可以离线阅读等。如果想用 Outlook Express、Foxmail 试试，只要配置好一个账号，将邮件接收服务器设置为 IMAP 服务器就可。此外，IMAP 还提供了一个更好的方法供您从多个设备中访问邮件。不管您是在工作中检查邮件，或者通过手机检查邮件，还是在家里检查邮件，IMAP 都能确保您从任何设备随时访问新邮件。总之，IMAP 可从整体上为用户带来更为可靠的体验。虽然 POP 更易丢失邮件或多次下载相同的邮件，但 IMAP 通过邮件客户端与邮件服务器之间的双向同步功能很好地避免了这些问题。例如，Gmail、QQ 邮箱都支持 IMAP 协议访问。

POP3 (Post Office Protocol 3) 即邮局协议的第 3 个版本，它规定怎样将个人计算机连接到 Internet 的邮件服务器和下载电子邮件的电子协议。它是 Internet 电子邮件的第一个离线协议标准，POP3 允许用户从服务器上把邮件存储到本地主机（即自己的计算机）上，同时删除保存在邮件服务器上的邮件，而 POP3 服务器则是遵循 POP3 协议的接收邮件服务器，用来接收电子邮件的。

SMTP (Simple Mail Transfer Protocol) 即简单邮件传输协议，它是一组用于由源地址到目的

地址传送邮件的规则, 由它来控制信件的中转方式。SMTP 协议属于 TCP / IP 协议族, 它帮助每台计算机在发送或中转信件时找到下一个目的地。通过 SMTP 协议所指定的服务器, 可以把 E-mail 寄到收信人的服务器上, 整个过程只要几分钟。SMTP 服务器则是遵循 SMTP 协议的发送邮件服务器, 用来发送或中转你发出的电子邮件。

Nginx 使用外部的类 HTTP 服务器去获知应该连接到哪个 IMAP/POP 后端服务器。

Nginx 在 HTTP Header 头中传递认证信息如下:

```
GET /auth HTTP/1.0
Host: auth.server.hostname
Auth-Method: plain
Auth-User: user
Auth-Pass: password
Auth-Protocol: imap
Auth-Login-Attempt: 1
Client-IP: 192.168.1.1
```

正常的应答为:

```
HTTP/1.0 200 OK      # 此行实际上被忽略, 可能根本不存在
Auth-Status: OK
Auth-Server: 192.168.1.10
Auth-Port: 110
Auth-User: newname  # 你可以忽略登录到后端的用户名
```

在 POP3 中, 为了避免发送明文口令的问题, 有一种新的认证方法, 命令为 APOP, 使用 APOP, 口令在传输之前被加密。当 POP3 认证使用 APOP 时, 你必须返回 Auth-Pass:

```
HTTP/1.0 200 OK      # 此行实际上被忽略, 可能根本不存在
Auth-Status: OK
Auth-Server: 192.168.1.10
Auth-Port: 110
Auth-User: newname  # 你可以忽略登录到后端的用户名
Auth-Pass: password # 这必须用户的明文密码
```

认证失败的应答结果为:

```
HTTP/1.0 200 OK      # 此行实际上被忽略, 可能根本不存在
Auth-Status: Invalid login or password
Auth-Wait: 3          # Nginx 在重新读取客户端登录账号、密码之前会等待 3 秒钟
# client's login/passwd again
```

在本章的指令介绍中, 指令的“使用环境”是该指令在 Nginx 配置文件中使用的位置, 例如使用环境为“mail, server”, 表示该指令可以在以下位置使用:

mail { ..... }大括号内; server { ..... }大括号内。

## 15.1 Nginx 邮件核心模块

### 15.1.1 auth 指令

从 Nginx 0.5.15 版本开始，该指令重命名为 pop3\_auth。

### 15.1.2 imap\_capabilities 指令

语法: `imap_capabilities "capability1" [ "capability2" .. "capabilityN" ]`

默认值: "IMAP4" "IMAP4rev1" "UIDPLUS"

使用环境: main, server

在客户端发送 IMAP 命令 CAPABILITY 的时候，通过这个指令，你可以设置 IMAP 协议扩展列表。如果启用 starttls 指令，STARTTLS 将被自动添加。

目前，标准的 IMAP 扩展发布在网站 [www.iana.org](http://www.iana.org)。imap\_capabilities 设置示例如下：

```
mail {  
    imap_capabilities NAMESPACE SORT QUOTA;  
}
```

### 15.1.3 imap\_client\_buffer 指令

语法: `imap_client_buffer size`

默认值: 4k/8k

使用环境: main, server

使用本指令可以设置 IMAP 命令的读取缓冲区。默认值等于页大小（该值取决于操作系统，为 4k 或 8k）。

### 15.1.4 listen 指令

语法: `listen address:port [ bind ]`

默认值: no

使用环境: server

该指令用于指定接收请求的服务器地址和端口。它也可以单一指定服务器地址或端口,此外,地址客户使用服务器名称,示例如下:

```
listen 127.0.0.1:8000;  
listen 127.0.0.1;  
listen 8000;  
listen *:8000;  
listen localhost:8000;
```

IPv6 地址 (Nginx 0.7.58 以上版本支持) 请在中括号中设置, 示例如下:

```
listen [::]:8000;  
listen [fe80::1];
```

在 listen 指令中, 可以设置系统调用 bind (2)。

### 15.1.5 pop3\_auth 指令

语法: pop3\_auth [plain] [apop] [cram-md5]

默认值: plain

使用环境: main, server

该指令可以设置 POP3 客户端认证的许可方法:

plain——USER/PASS, AUTH PLAIN, AUTH LOGIN

apop——APOP

cram-md5——AUTH CRAM-MD5

### 15.1.6 pop3\_capabilities 指令

语法: pop3\_capabilities “capability1” [ “capability2” .. “capabilityN” ]

默认值: “TOP” “USER” “UIDL”

使用环境: main, server

在客户端发送 POP3 命令 CAPA 的时候, 通过这个指令, 你可以设置 POP3 协议扩展列表。如果你启用 starttls 指令, STLS 将被自动添加。SASL 将在 auth 指令中添加。

### 15.1.7 protocol 指令

语法: protocol [ pop3 | imap | smtp ];

默认值: IMAP

使用环境: server

该指令用于设置虚拟主机 `server{...}` 块使用的协议。

## 15.1.8 server 指令

语法: `server {...}`

默认值: no

使用环境: mail

该指令用于配置虚拟主机。这里没有明确区分基于 IP 地址的虚拟主机和基于域名的虚拟主机（客户端请求中 Header 头 Host 行的值）。该指令用于配置虚拟主机。在 `server {.....}` 中，使用 `listen` 指令来为一个虚拟主机设置监听的 IP 和端口，使用 `server_name` 指令来设置不同的虚拟主机名称。

## 15.1.9 server\_name 指令

语法: `server_name name fqdn_server_host`

默认值: The name of the host, obtained through `gethostname()`

使用环境: mail, server

该指令用于设置虚拟主机的主机名，示例如下：

```
server {  
    server_name  example.com www.example.com;  
}
```

`server_name` 参数中的第一个名称将成为服务器的基础名称。默认名称为被使用的服务器名称（hostname），你可以使用通配符 “\*” 代替名称的首部分：

```
server {  
    server_name  example.com *.example.com;  
}
```

上述示例中的两个名称可以合并成以下的一个：

```
server {  
    server_name  .example.com;  
}
```

如果在客户端请求中没有 Header 头 “Host”，或者 Header 头不匹配任何 `server_name`，服务

器的基础名称将被用于 HTTP 重定向。你可以使用 “\*” 强制 Nginx 在 HTTP 重定向中使用 Host 头（注意：“\*” 不能作为虚拟主机的第一个名称，但是，你可以使用一个伪名称 “\_” 来代替第一个名称）：

```
server {  
    server_name example.com *;  
}  
server {  
    server_name _ *;  
}
```

### 15.1.10 smtp\_auth 指令

语法：smtp\_auth [login] [plain] [cram-md5]；

默认值：login plain

使用环境：main, server

该指令可以设置 SMTP 客户端认证的许可方法：

login——AUTH LOGIN

plain——AUTH PLAIN

cram-md5——AUTH CRAM-MD5

### 15.1.11 smtp\_capabilities 指令

语法：smtp\_capabilities “capability1” [ “capability2” .. “capabilityN” ]

默认值：no

使用环境：main, server

在客户端发送 SMTP 命令 EHLO 的时候，通过这个指令，你可以设置 SMTP 协议扩展列表。此列表通过 smtp\_auth 指令启用的方法自动扩展。目前，标准的 SMTP 扩展发布在网站 [www.iana.org](http://www.iana.org)。

### 15.1.12 so\_keepalive 指令

语法：so\_keepalive on|off;

默认值：off

使用环境: main, server

通过该指令, 你可以为连接到 IMAP/POP3 后端服务器的套接字设置 SO\_KEEPALIVE 选项。在 FreeBSD 中, keepalive 选项将被用于所有的连接, 它可以通过内核参数 (见 sysctl net.inet.tcp.always\_keepalive) 关闭。

### 15.1.13 timeout 指令

语法: timeout milliseconds;

默认值: 60000

使用环境: main, server

通过该指令, 你可以设置代理服务器连接到后端的超时时间。

## 15.2 Nginx 邮件认证模块

Nginx 邮件认证模块示例如下:

```
{  
    auth_http      localhost:9000/cgi-bin/nginxauth.cgi;  
    auth_http_timeout 5;  
}
```

### 15.2.1 auth\_http 指令

语法: auth\_http URL

默认值: no

使用环境: mail, server

通过该指令, 你可以为认证设置 URL 到扩展的类 HTTP 服务器。

### 15.2.2 auth\_http\_header 指令

语法: auth\_http\_header header value

默认值: no

使用环境: mail, server

通过该指令，你可以在身份认证的过程中添加一个 HTTP 头和它的值。这使得它可以使用一个共享的秘密字符串，确保请求是由 Nginx 应答。

示例如下：

```
auth_http_header X-NGX-Auth-Key "secret_string";
```

### 15.2.3 auth\_http\_timeout 指令

语法：auth\_http\_timeout milliseconds;

默认值：60000

使用环境：mail, server

通过该指令，你可以设置认证过程的超时时间。

## 15.3 Nginx 邮件代理模块

Nginx 可以代理 IMAP、POP3、SMTP 协议。

### 15.3.1 proxy 指令

语法：proxy on | off

默认值：off

使用环境：mail, server

通过该指令，你可以开启或关闭邮件代理。

### 15.3.2 proxy\_buffer 指令

语法：proxy\_buffer size

默认值：4k/8k

使用环境：mail, server

通过该指令，你可以设置代理连接的缓冲区大小。默认值等于页大小（该值取决于操作系统，为 4k 或 8k）。



### 15.3.3 proxy\_pass\_error\_message 指令

语法: proxy\_pass\_error\_message on | off

默认值: off

使用环境: mail, server

通过该指令, 你可以传递从后端被代理服务器获取的错误信息给客户端。通常情况下, 如果成功通过 Nginx 的认证, 后端被代理服务器将不会传递错误信息给客户端。

但是在一些正确密码的应答过程中, POP3 服务器错误属于正常的行为。例如 CommuniGatePro 通知用户关于邮箱存储空间满(或者其他事件)将在认证时周期性地发出错误信息。在这种情况下, 有必要设置 proxy\_pass\_error\_message 指令的参数为 on。

### 15.3.4 proxy\_timeout 指令

语法: proxy\_timeout time

默认值: 24h

使用环境: mail, server

通过该指令, 你可以设置代理连接的超时时间。

### 15.3.5 xclient 指令

语法: xclient on | off

默认值: on

使用环境: mail, server

通过该指令, 你可以允许或禁止连接 SMTP 后端服务器时使用 XCLIENT 命令。这将允许后端强制限制客户端连接基于 IP/HELO/LOGIN。

如果 xclient 指令的参数为 on, Nginx 将首先传递以下信息给后端服务器:

```
EHLO server_name
```

然后:

```
XCLIENT PROTO=ESMTP HELO=client_helo ADDR=client_ip LOGIN=authenticated_user  
NAME=[UNAVAILABLE]
```

## 15.4 Nginx 邮件 SSL 模块

本模块为 POP3/IMAP/SMTP 提供 SSL/TLS 支持。配置几乎与 HTTP SSL 模块相同，但是不支持检查客户端证书。

### 15.4.1 ssl 模块

语法: `ssl on | off`

默认值: `ssl off`

使用环境: `mail, server`

为所在虚拟主机启用或禁用 SSL/TLS。

### 15.4.2 ssl\_certificate 指令

语法: `ssl_certificate file`

默认值: `cert.pem`

使用环境: `mail, server`

为所在虚拟主机指定 PEM 格式的证书文件。相同的文件客户包含其他的证书，并且密钥也是 PEM 格式。

### 15.4.3 ssl\_certificate\_key 指令

语法: `ssl_certificate_key file`

默认值: `cert.pem`

使用环境: `mail, server`

为所在虚拟主机指定 PEM 格式的密钥文件。

### 15.4.4 ssl\_ciphers 指令

语法: `ssl_ciphers file ciphers`

默认值: ALL:!ADH:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP

使用环境: mail, server

指出允许的密码格式描述, 密码被指定为 OpenSSL 支持的格式。



### 15.4.5 ssl\_prefer\_server\_ciphers 指令

语法: ssl\_prefer\_server\_ciphers on | off

默认值: off

使用环境: mail, server

要求 SSLv3 和 TLSv1 协议的服务器密码优先于客户端的密码。

### 15.4.6 ssl\_protocols 指令

语法: ssl\_protocols [SSLv2] [SSLv3] [TLSv1]

默认值: SSLv2 SSLv3 TLSv1

使用环境: mail, server

该指令用于指定使用的 SSL 协议。

### 15.4.7 ssl\_session\_cache 指令

语法: ssl\_session\_cache [builtin[:size] [shared:name:size]

默认值: builtin:20480

使用环境: mail, server

该指令用于设置存储 SSL 会话的缓存类型和大小。缓存类型如下:

**builtin**——OpenSSL 内建缓存, 只能在一个工作进程中使用。缓存大小为会话数。

**shared**——缓存在所有工作进程中共享。缓存的大小以字节数指定, 1MB 的缓存客户容纳大约 4 000 个会话。每个共享缓存必须有独自的名称, 同名的缓存客户在不同的虚拟主机中使用。

可以同时使用两种缓存类型, 例如:

```
ssl_session_cache builtin:1000 shared:SSL:10m;
```

然而, 只使用共享缓存而不使用内建缓存要更有效。

## 15.4.8 ssl\_session\_timeout 指令

语法: `ssl_session_timeout time`

默认值: 5m

使用环境: mail, server

该指令用于设置客户端可以重复使用存储在缓存中的会话参数时间。

## 15.4.9 starttls 指令

语法: `starttls on | off | only`

默认值: off

使用环境: mail, server

on——允许使用 POP3 的 STLS 命令和 IMAP/SMTP 的 STARTTLS 命令。

off——不允许使用命令 STLS/STARTTLS。

only——开启 STLS/STARTTLS 支持，并且需要客户端使用 TLS 编码。

## 15.5 Nginx 邮件模块配置实例

在这一节中，我们通过使用 PHP 脚本来进行后端认证，以展示如何配置 Nginx 邮件模块：

- (1) 假设你的 POP3/IMAP 邮件代理服务器 Nginx 运行在 192.168.1.1。
- (2) 假设你有两台 POP3/IMAP 后端邮件服务器：192.168.22 和 192.168.33。
- (3) 假设你有一台认证和重定向服务器：192.168.1.44。
- (4) 认证脚本为：/mail/auth.php。

192.168.1.1 服务器上的 `nginx.conf` 配置文件如代码 15-1 所示：

代码 15-1

```
user nobody;  
worker_processes 1;  
error_log logs/error.log info;  
pid logs/nginx.pid;
```

```

events {
    worker_connections 1024;
    multi_accept on;
}

mail {
    auth_http 192.168.1.44:80/mail/auth.php;
    pop3_capabilities "TOP" "USER";
    imap_capabilities "IMAP4rev1" "UIDPLUS";

    server {
        listen 110;
        protocol pop3;
        proxy on;
    }

    server {
        listen 143;
        protocol imap;
        proxy on;
    }
}

```

192.168.1.44 服务器上的/mail/auth.php 脚本内容如代码 15-2 所示:

#### 代码 15-2

```

<?php
/*
Nginx sends headers as
Auth-User: somuser
Auth-Pass: somepass
On my php app server these are seen as
HTTP_AUTH_USER and HTTP_AUTH_PASS
*/
if (!isset($_SERVER["HTTP_AUTH_USER"]) || !isset($_SERVER["HTTP_AUTH_PASS"])){
    fail();
}
$username=$_SERVER["HTTP_AUTH_USER"] ;
$password=$_SERVER["HTTP_AUTH_PASS"] ;
$protocol=$_SERVER["HTTP_AUTH_PROTOCOL"] ;
// default backend port
$backend_port=110;
if ($protocol=="imap") {
    $backend_port=143;
}
if ($protocol=="smtp") {
    $backend_port=25;
}
// nginx likes ip address so if your
// application gives back hostname, convert it to ip address here
$backend_ip["mailhost01"] ="192.168.1.22";

```



```

$backend_ip["mailhost02"] = "192.168.1.33";
// Authenticate the user or fail
if (!authuser($username,$userpass){
    fail();
    exit;
}
// Get the server for this user if we have reached so far
$userserver=getmailserver($username);
// Get the ip address of the server
// We are assuming that you backend returns hostname
// We try to get the ip else return what we got back
$server_ip=(isset($backend_ip[$userserver])?$backend_ip[$userserver]:$userserver;
// Pass!
pass($server_ip, $backend_port);

//END

function authuser($user,$pass){
    // put your logic here to authen the user to any backend
    // you want (database, ldap, etc)
    // for example, we will just return true;
    return true;
}

function getmailserver($user){
    // put the logic here to get the mailserver
    // backend for the user. You can get this from
    // some database or ldap etc
    // dummy logic, all users that start with a,c,f and g get mailhost01
    // the others get mailhost02
    if in_array(substr($user,0,1), array("a", "c", "f", "g")){
        return "mailhost01";
    } else {
        return "mailhost02";
    }
}

function fail(){
    header("Auth-Status: Invalid login or password");
    exit;
}

function pass($server,$port){
    header("Auth-Status: OK");
    header("Auth-Server: $server");
    header("Auth-Port: $port");
    exit;
}
?>

```