Nginx 入門

～大規模・高パフォーマンス HTTP サーバ～

Ver. 1.0

リナックスアカデミー矢越昭仁

2015/11/15

高いパフォーマンスと大規模処理能力により急速にシェアを拡大しつつある Web サーバー Nginx(エンジンエックス)は LPIC 試験の対象範囲にも入り、Web サーバー技術でも重要な位置をしめるようになりました。この講座では Nginx について、Apache と比較しながら解説します。

目次

はじめに

1995 年に一般公開され、瞬く間に広がった WWW サービス。今日では PC だけでなく、スマートフォンやタブレット PC などを使って、オフィスだけでなく屋外や地下鉄の中からでもアクセスされる生活になくてはならないサービスとなりました。この WWW サービスを支える HTTP サーバーで、この数年急速にシェアを拡大している Nginx(えんじんえっくす)。この講座では、LPIC 試験範囲にも取り込まれた Nginx を Linux ベーシックで取り上げた Apache HTTPD の内容と比較しながら実習を交え解説します。

表記について

この資料では以下の表記としています。

・フォント

コンピュータの操作および設定ファイルはクーリエフォント(タイプライター風)を用います。

```
search t123006.la.net
nameserver 10.20.123.6
```

・プロンプト

コマンド入力例がある場合は、先頭はプロンプト($または#)で始めます。

$ は一般ユーザでの操作、#はルートユーザでの操作を表します。なおユーザ切り替え(su)は省略しています。

・強調(ボールド)

コマンド入力では、キーボードから入力する場合を、設定ファイルの場合は修正箇所など特に強調したい場合に**ボールド**を使います。

```
$ date
Mon Mar 5 12:32:41 JST 2012
```

動作環境

今回の実習では、CentOS7 を使用しています。CentOS5 とは大きく異なりますが、特に今後主流となり得るサービス起動方式 systemd を採用しています。Nginx は Apache に似た設定方法を用いていますが、サービス起動については、この systemd の影響で大きく異なります。

オンラインバックアップ

矢越が実施した IT 特別講座の資料(補足資料、例題等含む)は、以下の URL にて掲載しています。この URL はリナックスアカデミー会員限定となっていますので、それ以外への再配布・再掲載は遠慮ください。

http://ycos.sakura.ne.jp/LA

また講座・資料への質問、要望は下記までメールをお願いします。

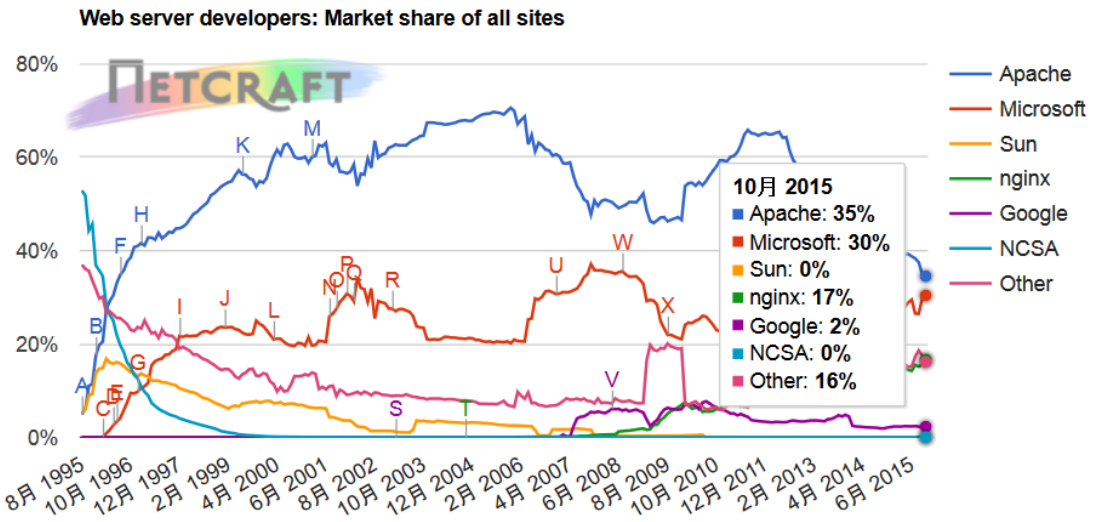
mailto:ycos001@yahoo.co.jp

Nginx とは

概要

Nginx（エンジン X）はロシアの大手検索サイトである Ramber が自社使用のため構築したサーバーで、完成当初は静的コンテンツしか扱えませんでした。それでも高速処理が可能であったこと、またリバースプロキシとしても利用できる事から、公開当初より人気があり、現在では主要 Web サーバーでは Apache(httpd / Apache ソフトウエア財団)、IIS（Internet Information Server / Microsoft 社）に次ぐ 3 位の位置にあります。

Netcraft 社 Web サーバーシェア動向



http://news.netcraft.com/archives/category/web-server-survey/

Nginx が生まれたのは 2002 年ですが、2004 年の公開後非常に反響があったので、2011 年夏には NGINX Inc. として独立しサポートを続けています。
また HTTP、SMTP、POP、IMAP の Proxy サーバーとして動作する事もでき、今後のシェア拡大が予想されます。そのため現在では LPIC 試験対象となっています。

Apache の限界

Apache は多くのプラットフォームで動作し、良好なパフォーマンスと OSS の恩恵を受けた多機能が評価され、1994年4月の登場から瞬く間にシェアトップとなりました。特に 1996 年に公開された Ver.1.1 でモジュールがサポートされると、多くの機能を取込みトップとなってから現在に至るまで、2014 年の 7 月を除きトップの座にあります。
しかし、クラウド・コンピューティングが定着し大規模なサービスが一般的になると、そのプロセス管理方法が問題視されるようになりました。Apache は MPM（Multi Processing Module）により、多重化処理を行います。MPM は prefork と worker の 2 種類用意されており、以下の特徴があります。

方式	Prefork	Worker
概要	1プロセスが1リクエストを処理	1プロセスで複数リクエスト処理 (プロセス内のスレッドが対応)
メリット	安定性・セキュリティ強度が高い 言語プロセッサを取り込みやすい	低メモリ、高パフォーマンス スレッド間での情報共有が簡単
デメリット	メモリ消費が多い パフォーマンスが低い	リクエスト隔離性が低い CGI 実行に制約がある

Apache 2.4.1 からは、worker の Keep-alive 機能を変更した event が加わっている。

MPM ではリクエスト数が増えると、プロセスも増えるため OS の性能限界に近づくと機能しないという事態が発生します。一般論として同時アクセス1万クライアントが上限といわれ、C10K(Client 10,000)問

題と呼ばれています。
Nginx ではイベント駆動方式を採用しています。イベント駆動方式ではプロセス数を限定させつつ、高速に処理することでリクエストを捌きます。リクエストを回転すしのコンベアのように次々と処理してゆきます。プロセスが増加しないため、メモリ使用量は低く抑える事ができ、CPU 性能によって処理能力が決定するため、スケールアウトしやすい特性を持ちます。

性能比較サンプル
ab –n 5000 –c 1000 http://localhost 結果比較

指標	Apache MPM			Nginx
	Prefork	Worker	Event	
Transfer rate [KB/s]	60.9	67.1	71.3	27,025.7
Requests per sec	209.4	232.0	243.5	7082.5

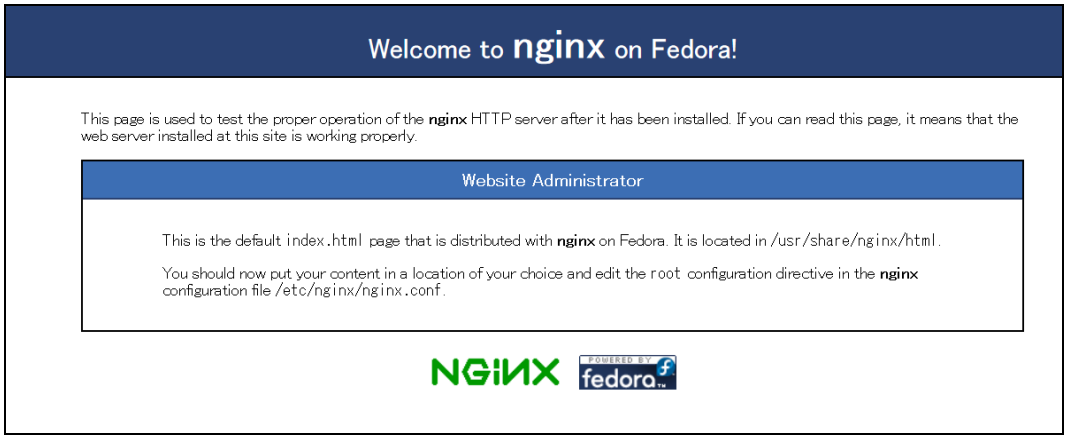
Apache の Event 機能は Nginx の仕組みに似た構造となっていますが、現時点ではまだチューニングが難しく Nginx ほどの性能を発揮していません。今後の改良が望まれます。

Nginx の導入

オフィシャルサイト http://nginx.org/en/ からの入手もしくは、EPEL リポジトリからの導入可能です。
今回は、EPEL を採用します（https://fedoraproject.org/wiki/EPEL/ja 、epel-release)。

```
$ env LANG=C yum info nginx
Loaded plugins: fastestmirror, langpacks
Loading mirror speeds from cached hostfile
 * base: www.ftp.ne.jp
 * centosplus: www.ftp.ne.jp
 * epel: ftp.kddilabs.jp
 * extras: www.ftp.ne.jp
 * updates: www.ftp.ne.jp
Installed Packages
Name        : nginx
Arch        : x86_64
Epoch       : 1
Version     : 1.6.3
Release     : 6.el7
Size        : 1.4 M
Repo        : installed
From repo   : epel
Summary     : A high performance web server and reverse proxy server
URL         : http://nginx.org/
License     : BSD
Description : Nginx is a web server and a reverse proxy server for HTTP, SMTP,
            : POP3 and IMAP protocols, with a strong focus on high concurrency,
            : performance and low memory usage.
```

インストール後は service nginx start によりサービスを起動できます。自身にアクセスすると以下のテストページが表示されます。



性能測定を行うab(apache bench)や、htpasswd などのツールはNginxでも利用できるので、合わせて httpd-tools もインストールしておくとよいでしょう。
性能比較を行うのであれば、Apacheもインストールし、それぞれabコマンドを実行し測定します。

設定ファイル

インストール直後のNginxの設定ファイルは /etc/nginx 下に格納され、以下のファイル群があります。

ファイル名	概要
fastcgi.conf	CGI 用サブシステム FastCGI の設定ファイル
mime.types	拡張子とコンテンツタイプ対応表(原則編集しません)
nginx.conf	主たる設定ファイル
scgi_params	CGI 用サブシステム Simple CGI の設定ファイル
uwsgi_params	Python CGI 用サブシステム uWSGI の設定ファイル
koi*, win-utf	ロシア語の言語(文字)関連ファイルで、今回は使用しません。

これらを編集し、サーバー構築の演習を行います。

静的コンテンツ

まず HTTP サーバーとして基本動作である、静的コンテンツを提供する設定について解説します。
設定ファイルは nginx.conf となり、基本的には以下の文法となります。詳細についてはオリジナルサイト http://nginx.org/en/docs を参照してください。現在、英語とロシア語の 2 言語が提供されています。

```
# コメント
設定名　設定値　…;
ブロック名　{
　設定名　設定値;
};
```

設定ファイル

設定内容はコンテントと呼び、各行はセミコロン(;)で終了します。機能によって大きくブロックと呼ばれる単に分割されます。ブロックはグローバルや event、http などがあり、グローバル以外は、中かっこ{}で囲まれます。http ブロックはさらに、server、location ブロックが入れ子構造をとります。階層構造の上位で設定した内容は、省略値として下位の構造に引き継がれます。

```
# シンプルな Nginx 設定例
user              nginx;
worker_processes auto;
error_log         /var/log/nginx/error.log;

events {
      worker_connections          1024;
}
http {
      include             /etc/nginx/mime.types;
      default_type        application/octet-stream;
      include             /etc/nginx/conf.d/*.conf
      server {
                  server_name       _;
                  listen            80;
                  root              /usr/share/nginx/html;
                  location / {
                  }
      }
}
```

グローバルブロック

Nginx 全体に関する定義を行います。Nginx は1つのワーカープロセスで多くのリクエストを処理しますが、そのプロセスの数やプロセスのユーザ等を指定しています。

event ブロック

ワーカープロセスの動作を定義します、この例では一度に処理するクエスト数を定義しています。

http ブロック

概要でも説明したように、Nginx は HTTP サーバー以外にも、メール Proxy など複数の機能を提供しています。そのため、HTTP サーバーに関する設定は、http ブロック内に記述します。
インストール直後では、include と default_type が定義されています。include は外部ファイルをと取り込み、この例ではファイル名の拡張子に HTTP ヘッダに用いる Conte-Type を紐づけています。
また、default_type は特に拡張子を指定しない場合に採用される Content-Type です。application/octet-stream の場合、多くのブラウザは「ファイルに名前をつけて保存」を行います。
またモジュールごと、/etc/nginx/conf.d/に設定ファイルを配置しています。

Server ブロック

Server ブロックでは、サーバーの IP アドレスやホスト名などを定義し Apache の仮想サーバーに相当します。root ディレクトリは Apache の DocumentRoot に相当します。
Nginx では、バーチャルサーバとして動作する事を前提としているため、リクエスト URL のホスト名と、server_name を比較し合致したブロックの設定を採用します。その為、どの server_name にも合致しなかった場合に備え、全受容名(catch-all)として、下線(_)を用います。server_name には FQDN だけでなく、「.la.net」のようなドメイン名などを空白で区切って複数指定する事ができます。

location ブロック

location ブロックは、ファイルシステム上のコンテンツ配置やアクセス制限などを定義します。Nginx の特徴的な機能としては、単にコンテンツ格納先ディレクトリを指し示すだけでなく、URI から、ファイル単位の配置を管理できます。たとえば、ファイル名が.jpg, .png を含む物だけを別のディレクトリに格納する事ができます。チルダ(～)が、続く文字列が正規表現である事を表しています。

```
location / {
}
location ~ ¥(jpg|png)$ {
        root        /usr/share/nginx/images;
}
```

nginx.conf を修正したら、reload により設定を読み込みます。

```
# service   nginx   reload
```

この例では、図形イメージ(.jpg, png で終わるファイル名)がアクセスされた場合、/usr/share/nginx/images ディレクトリをアクセスするよう変更されたため、先のテストページではロゴの部分が欠落します。ロゴに相当するファイルを上記ディレクトリへ移動させ、ファイルの種類によってアクセスするディレクトリが変わっている事を確認しましょう。

ユーザディレクトリ

location の正規表現を応用すると、ユーザディレクトリを定義できます。

```
location   ~   ^/~(.+?)(/.*)?$   {
        alias       /home/$1/public_html$2;
}
```

URI を示す正規表現でカッコ()を用いると、続く本文で$1、$2 として一致した文字列を参照できます。この例では、/~で始まる、次の/を含まない文字列を$1、更に/を含み最後までの文字列を$2 としています。よって、/~student/dir/abc の場合、$1 は student、$2 は/dir/abc が代入されます。
なお Apache 同様、Nginx のユーザ権限で当該ディレクトリがアクセスできる必要があります。

ログファイル

ログファイルは/var/log/nginx 下に格納されます。ファイルは access.log と error.log が日付ごとに切り分けられ、古いものは圧縮されます。

```
# ls /var/log/nginx
access.log                  access.log-20150929         error.log-20150921.gz
access.log-20150921.gz      error.log                   error.log-20150929
```

access.log

```
::1 - - [03/Oct/2015:23:13:41 +0900] "GET / HTTP/1.1" 200 3700 "-" "Mozilla/5.0 (X11; Linux x86_64; rv:38.0) Gecko/20100101 Firefox/38.0" "-"
::1 - - [03/Oct/2015:23:13:41 +0900] "GET /nginx-logo.png HTTP/1.1" 200 368 "http://localhost/" "Mozilla/5.0 (X11; Linux x86_64; rv:38.0) Gecko/20100101 Firefox/38.0" "-"
```

error.log

```
2015/10/04 22:57:39 [error] 2128#0: *3 open() "/usr/share/nginx/images/poweredby.png" failed (2: No such file or directory), client: 192.168.1.9, server: _, request: "GET /poweredby.png HTTP/1.1", host: "192.168.1.14", referrer: "http://192.168.1.14/"
2015/10/04 22:58:07 [error] 2128#0: *2 user "user3" was not found in "/usr/share/nginx/staff/.htpasswd", client: 192.168.1.9, server: _, request: "GET /staff/ HTTP/1.1", host: "192.168.1.14"
```

ホストベースアクセス制限

ホストベースのアクセス制限では、allow と deny を用います。引数は IP アドレスと CIDR 表記が使え、複数設定する場合は、複数行記述します。また Apache と異なり、allow, deny の評価順序は登場順になります。

```
location / {
        allow       127.0.0.1;
        allow       ::1/128;
        deny        all;
}
```

Nginx には、ディレクトリごとにユーザによるアクセス制御の上書き機能（AllowOverride）はありません。

ベーシック認証

Nginx は Apache 同様の認証システムをサポートしています。ベーシック認証で必要となるパスワードファイルは Apache の htpasswd で作成できます。Apache をインストールすれば同梱されていますが、httpd-tools に必要なツールがまとめられています。

今回は /usr/share/nginx/staff ディレクトリを作成し、そこにベーシック認証で保護されたコンテンツを作成します。ユーザファイルは Apache と同様に作成することが出来ます。

```
# mkdir /usr/share/nginx/staff
# htpasswd  –c  /usr/share/nginx/.htpasswd  staff
```

ユーザファイルはユーザ名、暗号化されたパスワード、コメントをコロン(:)で区切って表します。
htpasswd で作成する以外にも、暗号化パスワードを openssl passwd コマンドで作成し、手作業で作成する事もできます。

```
# コメント
ユーザ名:暗号化パスワード:コメント
```

Openssl password を使い他の暗号化アルゴリズムを利用することも出来ます。アルゴリズムは以下の3種が選択できます。

- -crypt　伝統的な UNIX パスワード、規定値)
- -1　　MD5 ベース
- -apr1　Apache で採用されている MD5

使用例

```
$ openssl  passwd  -1  himitu
$1$TEz3HbxG$LMjmsF3n13y1xmqVYdLEL0
```

ユーザファイルを作成したら、nginx.conf に次の設定を追加し、reload します。

```
location /staff {
      root       /usr/share/nginx/;
      auth_basic          “Staff only area”;
      auth_basic_user_file          /usr/share/nginx/staff/.htpasswd;
}
```

ユーザファイルは相対パスで指定すると、root の設定とは関係なく、/etc/nginx/を起点とします。
また、Apache と異なり、ユーザファイルの中から認証に使うユーザを選択する事はできず、ファイルに登録したすべてのユーザで認証が可能です(Apache の require valid-user と同等)

最後に適当な index.html ファイルを作成し、動作確認を行います。

動的コンテンツ（CGI）

Nginx はそのままでは CGI に対応していませんが、FastCGI や SimpleCGI といった別モジュールにより実現しています。今回は Nginx 同様にプロセスを常駐させることで、メモリ使用量が少なくてすむ FastCGI を採用します。これも EPEL で提供されており、パッケージ名は「spawn-fcgi」です。また CGI として動作させるプログラムは、XAMPP 等で幅広く活用されている PHP を用います。

パッケージの導入

FastCGI モジュールである spawn-fcgi、PHP およびコマンド・ラインで PHP を起動する php-cli を導入します。また php の従属パッケージも合わせて導入します。

```
# yum install spawn-fcgi php php-cli
```

FastCGI 用 PHP 起動スクリプトの作成

FastCGI は予め外部プログラムを起動しておき、HTTPD からの要求によってそのレスポンスを返します。処理後も待機状態とし、プロセスは終了しません。
この処理用ワーカープロセスを生成する、スクリプトを用意します。

/usr/local/bin/php-fastcgi

```
#!/bin/sh
# FastCGI PHP worker
FASTCGI_USER=nginx
FASTCGI_PORT=9000
/usr/bin/spawn-fcgi -a 127.0.0.1 -p $FASTCGI_PORT -C 2 -u $FASTCGI_USER　¥
   -f /usr/bin/php-cgi
```

長いので、見やすくするため行末にバックスラッシュ（¥）を入れ改行していますが、1 行に記述してもかまいません。
spawn-fcgi が FastCGI としてワーカープロセスを軌道します、主なオプションは次の通りです。

- -a IP アドレス　　接続するサーバーの IP アドレス（例では 127.0.0.1）
- -p ポート　　接続するサーバーのポート番号（例では 9000）
- -C 個数　　起動するワーカープロセスの個数です。（例では 2+制御プロセス x1）
- -u ユーザ　　起動するワーカープロセスのユーザ名
- -f ファイルパス　　FastCGI が起動するプログラム（php-cgi は php-cli パッケージが提供）

systemd 制御ファイルの作成

続いて、systemd の制御ファイルを作成します。これは旧来の/etc/init.d/下にあった制御シェルスクリプトに代わるもので、先の FastCGI を起動するよう設計しています。

/etc/systemd/system/php-fastcgi.service

```
[Unit]
Description= php-fastcgi systemd service script

[Service]
Type=forking
ExecStart=/usr/local/bin/php-fastcgi start

[Install]
WantedBy=multi-user.target
```

主な設定

- Description　　このサービスの解説
- Type　　サービスプロセスの起動方法、fork により子プロセスを生成します。
- ExecStart　　サービス起動プログラム、先に作ったスクリプトです。
  今回引数の処理は省略しましたが、stop, status などを実装するとよいでしょう。
- WantedBy　　マルチユーザモードで動作する事を表しています。

systemd のリロード

制御ファイルを追加したので、systemd に制御ファイルの再読み込みと、php-fastcgi サービス起動を行います。

```
# systemctl daemon-reload
# systemctl start php-fastcgi
# systemctl status php-fastcgi
php-fastcgi.service - php-fastcgi systemd service script
   Loaded: loaded (/etc/systemd/system/php-fastcgi.service; disabled)
   Active: active (running) since Sun 2015-10-04 23:17:14 JST; 13min ago
 Main PID: 2419 (php-cgi)
   CGroup: /system.slice/php-fastcgi.service
           ├2419 /usr/bin/php-cgi
           ├2420 /usr/bin/php-cgi
           └2421 /usr/bin/php-cgi


Oct 04 23:17:14 cent71.localdomain systemd[1]: Starting php-fastcgi systemd ....
Oct 04 23:17:14 cent71.localdomain php-fastcgi[2417]: spawn-fcgi: child spawn...
Oct 04 23:17:14 cent71.localdomain systemd[1]: Started php-fastcgi systemd s....
Oct 04 23:30:56 cent71.localdomain systemd[1]: Started php-fastcgi systemd s....
Hint: Some lines were ellipsized, use -l to show in full.
```

サービスの恒久化

Nginxを含めシステム起動時に、自動的にサービスを起動するには、chkconfigに変えsystemctlコマンドを使います。

```
# systemctl enable nginx
ln -s '/usr/lib/systemd/system/nginx.service' '/etc/systemd/system/multi-user.target.wants/nginx.service'
# systemctl enable php-fastcgi
ln -s '/etc/systemd/system/php-fastcgi.service' '/etc/systemd/system/multi-user.target.wants/php-fastcgi.service'
```

nginx.conf の変更

php ファイルに対しては、上記の php-fastcgi サービスを呼び出すよう、Nginx の設定に追加を行います。静的コンテンツで行った、ファイル名の正規表現に基づき設定内容を定義します。

/etc/nginx/nginx.conf へ追加

```
location ~ ¥.*php$ {
        root       /usr/share/nginx/html;
        fastcgi_pass       127.0.0.1:9000;
        fastcgi_index      index.php;
        fastcgi_param      SCRIPT_FILENAME $document_root$fastcgi_script_name;
        include            fastcgi_params;
}
```

/etc/nginx/fastcgi_param が php-cgi と Nginx を連携させるための設定ファイルになっており、Nginx に同梱されています。

最後にサンプルとして、test.php を作成します。PHP の設定内容を表示する非常に簡単なプログラムです。

/usr/share/nginx/html/test.php

```
<?php
phpinfo();
?>
```

ロードバランサー

Nginx の Apache にない機能としてプロキシ―サーバー機能が有名です。Proxy は代理サーバーとも呼ばれ、リクエストに対し自身が処理を行うのではなく、他のサーバーへ連携する機能を提供します。一般的な Web プロキシ―は、クライアントからのリクエストを一手に引き受け、代わりに WWW サーバーに問合せを行い、結果をキャッシュとして蓄積します。場合によっては不適切なリクエストを却下する機能もあり、セキュリティ向上のために用いられる事が多いです。
逆にクライアントからのリクエストを複数のサーバーへ振り分けるものをリバースプロキシ―または、ロードバランサーと呼んでいます。今回はロードバランサーの例を紹介します。

upstream ブロック

upstream はリクエストを転送する先を定義します。server に続けて FQDN を記述し、特に指定しない場合はアクセスがあるごとに、上から順に切替えます。これをラウンドロビンと呼びます。他にも重みを指定や、接続時間の長さによって順序を変えるといった機能があります。
この upstream の名前を server ブロックで proxy_pass として指定します。

```
upstream myservers {
        server    www.lpi.org;
        server    www.linuxacademy.ne.jp;
        server    www.lpi.or.jp;
}
server {
        listen       8080;
        location / {
                    proxy_pass              http://myservers;
        }
}
```

付録：ディレクティブ比較表

Linux ベーシックのテキストで登場した主な Apache ディレクティブと Nginx の対比表

#	Apache HTTPD	Nginx
1	ServerRoot	なし(ハードコーディング)
2	Timeout	*_timeout (バリエーション多数)
3	KeepAlive	なし(原則 KeepAlive)
4	StartServers	worker_processers
5	MaxClients	凡そ worker_processers *worker_connections/4
6	Listen	listen
7	User	user ユーザ名 [グループ名]
8	Group	user に統合
9	ServerAdmin	なし
10	ServerName (VirtualHost)	server_name
11	DocumentRoot	root
12	UserDir	location ~ ^/~云々 {}
13	DirectoryIndex	index
14	<Directory …>	location … {}
15	- ExecCGI	FastCGI などの設定
16	- FollowSymLinks	disable_symlinks off;
17	- Indexes	autoindex on;
18	- Order	評価順序は登場順に固定
19	- Allow …	allow (引数は1つ、複数行可)
20	- Deny	deny (同上)
21	- AccessFileName	なし
22	- AllowOverride	なし
23	AuthType Basic	auth_basic “レルム”;
24	- AuthName	auth_basic へ統合
25	- AuthBasicProvider	なし
26	- AuthUserFile	auth_basic_user_file
27	- Require	なし。ユーザファイル全件対象