# .Deb

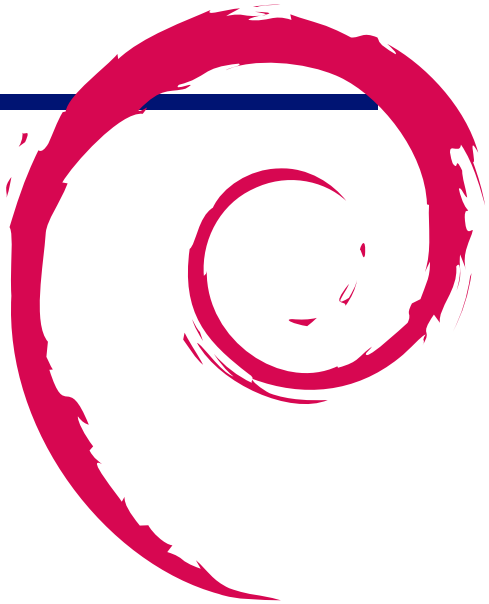
銀河系唯一のDebian専門誌

2018 年 11 月 17 日

## nginx 設定特集

# デビアン勉強会

## 目次



# 1 最近の Debian 関連のミーティング報告

杉本　典充

## 1.1 第 166 回東京エリア Debian 勉強会

2018 年 9 月 15 日 (土) に第 166 回東京エリア Debian 勉強会を開催しました。会場は東銀座にある朝日ネットさんをお借りして行いました。参加者は 5 名でした。

セミナー発表は、「Rethinking of debian/watch rule」という表題で林さんが発表しました。DebConf18 で発表した内容を日本語で再演しました。

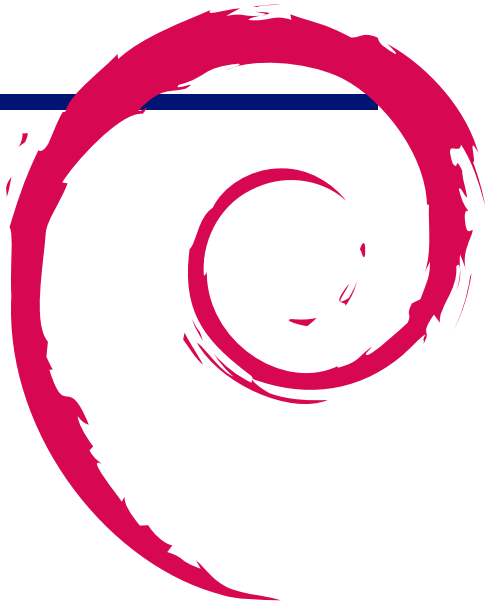
ハックタイムでは、持ち寄った課題をハックしました。

## 1.2 OSC 2018 Tokyo/Fall（第 167 回東京エリア Debian 勉強会）

2018 年 10 月 27 日 (土) に第 167 回東京エリア Debian 勉強会を OSC 2018 Tokyo/Fall のセミナーとして開催しました。OSC の会場は明星大学様でした。

セミナーは「Debian Updates」という表題で杉本が発表しました。参加者は 18 名でした。

ブース展示を行い、49 名のイベント参加者と交流しました。



# 2 参加者の紹介

杉本 典充

今回の事前課題は以下です。

1. Hack Time は何をしますか
2. 利用している web サーバを教えてください

## 2.1 gyx

1. learn about changes in Alioth / update maintained packages.
2. apache2, nginx, lighttpd, その他

## 2.2 @clothoid

1. Debian 上の emacs の設定を見直します
2. その他

## 2.3 koedoyoshida

1. （未定）
2. apache2, nginx

## 2.4 yy_y_ja_jp

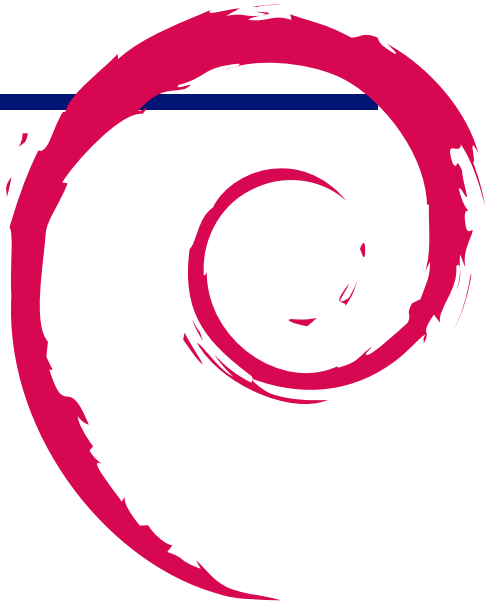
1. パッケージ更新
2. apache2, nginx

## 2.5 dictoss

1. Buster RC バグの探索、uwsgi パッケージの複数アプリ配備の確認
2. apache2, nginx

## 2.6 NOKUBI Takatsugu

1. nlp deb, 次回発表ネタの準備
2. apache2, nginx



# 3 debian における nginx の設定例

杉本　典充

## 3.1 はじめに

近年、nginx を web サーバとして採用する例が増えています。debian においては nginx を deb パッケージとして提供しています。

本発表では、debian における nginx のインストール、使い方、設定例を紹介します。

## 3.2 debian における web サーバのパッケージ

UNIX/Linux 環境で利用できる web サーバは、2018 年 11 月現在では apache、nginx の順に人気が高いです[*1]。

debian では apache、nginx の両方を apt でインストールできるようパッケージを提供しています。Debian 9 (stretch) では以下のパッケージ名とバージョンを提供しています。

| web サーバ名 | パッケージ名 | Debian 9 の提供バージョン |
|---|---|---|
| apache | apache2 | 2.4.25 |
| nginx | nginx | 1.10.3 |

## 3.3 nginx

### 3.3.1 nginx とは

nginx（エンジンエックス）とは、2004 年にリリースした web サーバです。nginx は大量のクライアントを同時に捌くことに主眼を置き、linux-2.5.44 で導入した epoll などのいわゆるイベント駆動のシステムコールを利用したサーバアプリケーションです[*2]。

nginx は HTTP/HTTPS のリクエストを処理するだけでなく、HTTP/HTTPS のリバースプロキシサーバ、mail プロキシサーバ、TCP/UDP 通信のプロキシサーバとして動作するように実装しています[*3]。

web サイトは `https://nginx.org/` であり、`https://nginx.org/en/docs/` にドキュメントも豊富にあります。

nginx のソースコードは 2-clause BSD-like license で提供しており、debian においては main セクションのパッケージとして配布しています。

[*1] `https://w3techs.com/technologies/comparison/ws-apache,ws-microsoftiis,ws-nginx`
[*2] いわゆる C10K 問題の解決を図るために実装したと言われています。
[*3] `https://nginx.org/en/`

### 3.3.2 debian における nginx のインストール方法とディレクトリ構成

nginx は apt コマンドでインストールすることができます。

```
# apt install nginx
```

Debian における”nginx”パッケージは拡張モジュールを含んでいるかで以下の 3 種類に分けています。通常の web サーバ、リバースプロキシとしての利用であれば、”nginx”パッケージの指定で十分と思います。

- nginx (または nginx-full)
  - 通常の利用ではこれを利用する
  - upstream が提供する基本的な拡張モジュールを含んでいる
- nginx-light
  - HTTP/HTTPS のリクエストを受け付けるサーバとしてのみ動作すればよい場合に利用する
  - インストールする拡張モジュールを最低限に抑えている
  - SSL/TLS、FastCGI、uWSGI の拡張モジュールも含んでいる
- nginx-extras
  - nginx-full の拡張モジュールに加えて、多くのサードパーティ製の拡張モジュールを含んでいる

## 3.4 nginx の設定例

### 3.4.1 nginx パッケージのインストール直後の設定

apt コマンドで nginx をインストールした直後は、以下のディレクトリ構造で設定ファイルが配備されています。

nginx の設定ファイルは、”/etc/nginx/nginx.conf” が基点となっており、nginx.conf が様々なファイルを include することで設定を構成しています。

```
$ tree /etc/nginx
/etc/nginx
├── conf.d
├── fastcgi.conf
├── fastcgi_params
├── koi-utf
├── koi-win
├── mime.types
├── modules-available
├── modules-enabled
│   ├── 50-mod-http-auth-pam.conf -> /usr/share/nginx/modules-available/mod-http-auth-pam.conf
│   ├── 50-mod-http-dav-ext.conf -> /usr/share/nginx/modules-available/mod-http-dav-ext.conf
│   ├── 50-mod-http-echo.conf -> /usr/share/nginx/modules-available/mod-http-echo.conf
│   ├── 50-mod-http-geoip.conf -> /usr/share/nginx/modules-available/mod-http-geoip.conf
│   ├── 50-mod-http-image-filter.conf -> /usr/share/nginx/modules-available/mod-http-image-filter.conf
│   ├── 50-mod-http-subs-filter.conf -> /usr/share/nginx/modules-available/mod-http-subs-filter.conf
│   ├── 50-mod-http-upstream-fair.conf -> /usr/share/nginx/modules-available/mod-http-upstream-fair.conf
│   ├── 50-mod-http-xslt-filter.conf -> /usr/share/nginx/modules-available/mod-http-xslt-filter.conf
│   ├── 50-mod-mail.conf -> /usr/share/nginx/modules-available/mod-mail.conf
│   └── 50-mod-stream.conf -> /usr/share/nginx/modules-available/mod-stream.conf
├── nginx.conf
├── proxy_params
├── scgi_params
├── sites-available
│   └── default
├── sites-enabled
│   └── default -> /etc/nginx/sites-available/default
├── snippets
│   ├── fastcgi-php.conf
│   └── snakeoil.conf
├── uwsgi_params
└── win-utf

6 directories, 24 files
```

コメントの部分を除いた nginx.conf は以下となります。標準で gzip を使ったファイル圧縮転送の機能が on になっています。

```
$ grep -v -e '^\s*#' -e '^\s*$' /etc/nginx/nginx.conf
user www-data;
worker_processes auto;
pid /run/nginx.pid;
include /etc/nginx/modules-enabled/*.conf;
events {
  worker_connections 768;
}
http {
  sendfile on;
  tcp_nopush on;
  tcp_nodelay on;
  keepalive_timeout 65;
  types_hash_max_size 2048;
  include /etc/nginx/mime.types;
  default_type application/octet-stream;
  ssl_protocols TLSv1 TLSv1.1 TLSv1.2; # Dropping SSLv3, ref: POODLE
  ssl_prefer_server_ciphers on;
  access_log /var/log/nginx/access.log;
  error_log /var/log/nginx/error.log;
  gzip on;
  gzip_disable ‘‘msie6’’;
  include /etc/nginx/conf.d/*.conf;
  include /etc/nginx/sites-enabled/*;
}
```

debian では VirtualHost で運用しやすいようになっており、”default”という HTTP で待ち受けする設定ファイルが標準で入っています。この場合、公開する静的ファイルを”/var/www/html”配下に配置するとクライアントへ HTTP でファイルを配布できます。

```
$ grep -v -e '^\s*#' -e '^\s*$' /etc/nginx/sites-available/default
server {
  listen 80 default_server;
  listen [::]:80 default_server;
  root /var/www/html;
  index index.html index.htm index.nginx-debian.html;
  server_name _;
  location / {
    try_files $uri $uri/ =404;
  }
}
```

### 3.4.2 SSL/TLS 設定

nginx は、SSL 証明書を用いた SSL/TLS の https 通信の待ち受けを行うことができます。

SSL/TLS 設定として、”/etc/nginx/sites-available/ssl”ファイルを生成し、このファイルを”/etc/nginx/sites-enabled/ssl”のシンボリックリンクを作成することで VirtualHost を追加できます。

mozilla が SSL/TLS として web サーバを動かすときに設定例を公開しているため、参考にするとよいです。

`https://mozilla.github.io/server-side-tls/ssl-config-generator/`

SSL/TLS で通信する設定を行う時のポイントは以下です。

- SSL 証明書で中間証明書を必要とする場合は、サーバ証明書と中間証明書を結合して 1 つの”crt”ファイルにする
- ”ssl_protocols TLSv1.2;”とだけ書くと、TLSv1.2 のみで通信を受け付けるサーバにできる
- 過去に判明している openssl 関連の脆弱性対応に考慮した設定とすること
- `https://www.ssllabs.com/ssltest/` などのスキャンを行い、サーバに脆弱性があるか確認する
- ssl_dhparam は、DH 1024bit がデフォルトのため Logjam 対策として 2048bit で処理するよう指定するもの[*4]

[*4] 古いクライアントプログラムは DH 1024bit までしか対応していないことがあり、悩ましい。

```
$ grep -v -e '^\s*#' -e '^\s*$' /etc/nginx/sites-available/ssl
server {
  listen 443 ssl http2;
  listen [::]:443 ssl http2;
  ssl on;
  ssl_certificate      /etc/ssl/private/server.crt;
  ssl_certificate_key  /etc/ssl/private/server.key;
  ssl_session_cache    shared:SSL:1m;
  ssl_session_timeout  5m;
  ssl_protocols TLSv1.2; # Dropping SSLv3, ref: POODLE
  ssl_prefer_server_ciphers on; # ref: FREAK

  # Diffie-Hellman parameter for DHE ciphersuites, recommended 2048 bits
  #ssl_dhparam /path/to/dhparam.pem;

  # see https://mozilla.github.io/server-side-tls/ssl-config-generator/
  ssl_ciphers  HIGH:!aNULL:!MD5;

  access_log /var/log/nginx/ssl_access.log;
  error_log /var/log/nginx/ssl_error.log;
  gzip off;  # unuse gzip, ref: BREACH

  root /var/www/html;
  index index.html index.htm index.nginx-debian.html;
  server_name www.example.com;
  location / {
    try_files $uri $uri/ =404;
  }
}
```

### 3.4.3 リバース proxy 設定

リバース proxy とは、クライアントと直接接続するのは nginx が担い、クライアントへ応答する実際のデータ生成はバックエンドのサーバが行うように通信を中継する機能です。

特に SSL/TLS 通信の場合、リバース proxy の設定を行うと SSL 証明書を設定するサーバは nginx のみでよく、バックエンドの web サーバは http のみで通信する設定にすることができます。

リバース proxy の設定を行うには、まず upstream の設定を行います。

upstream には、中継するバックエンドの web サーバのホストとポート番号を指定します。複数のバックエンドを設定することが可能で、ロードバランサーのように振る舞うことが可能です[*5]。

ロードバランスするときのアルゴリズムの指定は、least_conn と ip_hash を指定でき、何も定義しない場合は server の行に書いた weight 値を用いた重み付きラウンドロビンで振り分けます[*6]。

```
$ sudo vi /etc/nginx/conf.d/upstream_proxy.conf

upstream backend_app1 {
  # least_conn;
  # ip_hash;

  server 192.168.1.100:80 weight=1;
  server 192.168.1.101:80 weight=1;
}
```

proxy するときの upstream への通信はデフォルトで HTTP/1.0 になっています。この設定は「proxy_http_version 1.1;」と書くと、upstream のサーバとの通信を HTTP/1.1 にできます[*7]。

[*5] クライアントの cookie を参照してどのバックエンドに中継するか制御する sticky の拡張モジュールは、deb パッケージには入っていません。サードパーティ製のモジュールで提供されるため nginx のコンパイルが必要です。

[*6] `http://nginx.org/en/docs/http/ngx_http_upstream_module.html` に `server` 句に利用できるオプションが書いてあります

[*7] `http://nginx.org/en/docs/http/ngx_http_proxy_module.html#proxy_http_version`

```
$ sudo vi /etc/nginx/sites-available/default

server {
  # (snip)
  location ~ ^/proxy/(.*)$ {
    proxy_pass http://backend_proxy1;
    #proxy_pass http://backend_proxy1/yourpath/;

    # if require.
    proxy_http_version 1.1;

    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-Proto $scheme;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header Host $http_host;
    #proxy_redirect http://backend_proxy1/  http://www.example.com/;
  }
  # (snip)
}
```

nginx の設定を確認する”nginx -t”を実行し、設定ファイルに間違いないか確認します。設定に間違いがなければ、nginx を restart します。

```
$ sudo nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful

$ sudo systemctl restart nginx
```

nginx のサーバの”/proxy/”へアクセスし、proxy 先の web 画面が表示されることを確認できれば、設定ができています。

### 3.4.4 FastCGI 設定

FastCGI とは、常駐した CGI プロセスが外部プロセス（ここでは nginx）と CGI 処理を行うためのリクエストとレスポンスのやりとりを定義したインタフェースの 1 つです。

nginx には CGI を動かす apache でいうところの mod_cgi 相当の機能はありません[*8]。そのため、nginx では外部のプロセスへ通信を中継することで CGI 処理に対応しており、FastCGI のインタフェースを持つ常駐プロセスと通信が可能です。

今回は FastCGI サーバとして PHP-FPM を使ってみます。PHP のバージョンは 7.0 系を利用するとし、以下コマンドでインストールします。

```
$ sudo apt install php7.0 php7.0-fpm
```

php7.0-fpm は ”/etc/php/7.0”配下に設定ファイルが配置されます。php-fpm の常駐プロセスの設定は、”/etc/php/7.0/fpm/pool.d/www.conf”です。

ここでは、www.conf を inet socket で待ち受けするように設定を変更します。

```
$ sudo vi /etc/php/7.0/fpm/pool.d/www.conf
;listen = /run/php/php7.0-fpm.sock
+listen = 9000
```

php-fpm を再起動すると、9000 ポートで listen しています。これで、/var/www/html 配下の”.php”のスクリプトはすべて php-fpm として動作するようになります。

```
$ sudo systemctl restart php7.0-fpm
$ ss -npta | grep 9000
LISTEN     0      128         :::9000                    :::*
```

phpinfo() のみを行う PHP スクリプトを配置しておきます。

```
$ sudo mkdir /var/www/html/myphpapp
$ sudo vi /var/www/html/myphpapp/phpinfo.php
<?php
  phpinfo();
```

[*8] apache で CGI プログラムを利用する場合、mod_cgi を利用して Perl プログラムを動作させることが過去には多かったと思います。

nginx の fastcgi サーバの upstream を設定します。

```
$ sudo vi /etc/nginx/conf.d/upstream_fcgi.conf
upstream backend_fcgi1 {
  # least_conn;
  # ip_hash;

  server 127.0.0.1:9000;
}
```

次に、nginx の VirtualHost 設定の default ファイル、必要であれば ssl ファイルに”.php”の拡張子を持つファイルへアクセスした場合はm fastcgi サーバへ proxy するように設定します。

```
$ sudo vi /etc/nginx/sites-available/default
  # (snip)
  location ~ \.php$ {
    include snippets/fastcgi-php.conf;
    fastcgi_pass backend_fcgi1;
  }
  # (snip)

$ sudo vi /etc/nginx/sites-available/ssl
  # (snip)
    location ~ \.php$ {
      include snippets/fastcgi-php.conf;
      fastcgi_pass backend_fcgi1;
  }
  # (snip)
```

nginx の設定を確認する”nginx -t”を実行し、設定ファイルに間違いないか確認します。設定に間違いがなければ、nginx を restart します。

```
$ sudo nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful

$ sudo systemctl restart nginx
```

nginx のサーバの”/myphpapp/phpinfo.php”へアクセスし、phpinfo の画面が表示されることを確認できれば、設定ができています。

### 3.4.5 WSGI 設定

WSGI（Web Server Gateway Interface）とは、常駐した CGI プロセスが外部プロセス（ここでは nginx）と CGI 処理を行うためのリクエストとレスポンスのやりとりを定義したインタフェースの 1 つです。

FastCGI と同様に、WSGI サーバへ通信を中継することで CGI 処理を行うことができます。

今回は WSGI サーバとして uwsgi を利用します。python のバージョンは Debian 9 (stretch) にパッケージが存在する python3.5、アプリケーションは django のチュートリアルを実装したものを使ってみます[*9]。

python3、pip3、uwsgi をインストールします。git はアプリケーションのソースコードを clone するためにインストールします。

```
$ sudo apt install python3 python3-pip uwsgi uwsgi-plugin-python3
$ sudo apt install git
```

django のライブラリのインストールと django を用いたアプリケーションを配置します。

```
$ sudo pip3 install -U django==2.0.9

$ cd
$ git clone https://github.com/dictoss/django-tutorial.git
$ sudo mkdir /var/www/wsgi_apps_uwsgi
$ sudo cp -r django-tutorial/2.0/mysite /var/www/wsgi_apps_uwsgi/
$ sudo chown -fR www-data:www-data /var/www/wsgi_apps_uwsgi/mysite
$ ls /var/www/wsgi_apps_uwsgi/mysite
db.sqlite3  manage.py  mysite  polls
```

uwsgi は、”/etc/uwsgi” 配下に設定ファイルを配置します。nginx の VirtualHost 設定のように”apps-available”、”apps-enabled” ディレクトリがあります。

[*9] https://github.com/dictoss/django-tutorial

”apps-available” ディレクトリに”django-tutorial.ini”という WSGI アプリケーションの設定ファイルを配置し、”apps-enabled” ディレクトリにシンボリックリンクを生成します。

```
$ sudo vi /etc/uwsgi/apps-available/django-tutorial.ini
[uwsgi]
uid = www-data
gid = www-data
plugin-dir = /usr/lib/uwsgi/plugins
plugin = python3
base = /var/www/wsgi_apps_uwsgi/mysite
chdir = /var/www/wsgi_apps_uwsgi/mysite
module = mysite.wsgi
callable = application
env =
socket = 0.0.0.0:3031
thunder-lock = true
processes = 2
threads = 32
master = True
vacuum = True
harakiri = 60
max-requests = 512
max-requests-delta = 64
post-buffering = 8192
```

```
$ cd /etc/uwsgi/apps-enabled
$ sudo ln -fs ../apps-available/django-tutorial.ini .
$ tree /etc/uwsgi
/etc/uwsgi
├── apps-available
│   ├── README
│   └── django-tutorial.ini
└── apps-enabled
├── README
└── django-tutorial.ini -> ../apps-available/django-tutorial.ini

2 directories, 4 files
```

uwsgi を再起動すると ”/etc/uwsgi/apps-enabled” 配下に設定した WSGI アプリケーションが起動します。django-tutorial.ini の設定では 3031 ポートで listen しています。

```
$ sudo systemctl restart uwsgi
$ ss -npta | grep 3031
LISTEN     0      100          *:3031                     *:*
```

nginx の wsgi サーバの upstream を設定します。

```
$ sudo vi /etc/nginx/conf.d/upstream_uwsgi.conf
upstream backend_uwsgi1 {
  # least_conn;
  # ip_hash;

  server 127.0.0.1:3031;
}
```

VirtualHost 設定の default ファイル、必要であれば ssl ファイルに proxy するディレクトリ名を決めて wsgi へ proxy する設定をします。

```
$ sudo vi /etc/nginx/sites-available/default
  # (snip)
  location ~ ^/mysite/(.*)$ {
    include uwsgi_params;

    uwsgi_param SCRIPT_NAME /mysite;
    uwsgi_param PATH_INFO /$1;

    uwsgi_pass backend_uwsgi1;
  }
  # (snip)

$ sudo vi /etc/nginx/sites-available/ssl
  # (snip)
  location ~ ^/mysite/(.*)$ {
    include uwsgi_params;

    uwsgi_param SCRIPT_NAME /mysite;
    uwsgi_param PATH_INFO /$1;

    uwsgi_pass backend_uwsgi1;
  }
  # (snip)
```

nginx の設定を確認する”nginx -t”を実行し、設定ファイルに間違いないか確認します。設定に間違いがなければ、nginx を restart します。

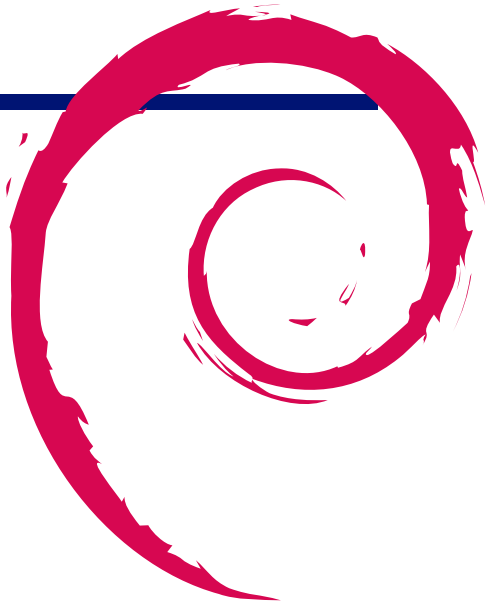
```
$ sudo nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful

$ sudo systemctl restart nginx
```

nginx のサーバの”/mysite/polls/”へアクセスし、”What’s up?”の画面が表示されることを確認できれば、設定ができています。

## 3.5 まとめ

nginx のよく使う設定をまとめてみました。web サーバに nginx を使おうを考えている場合は参考にしてみてください。



# 4 メモ

**Debian 勉強会資料**

2018 年 11 月 17 日　　初版第 1 刷発行

東京エリア Debian 勉強会 （編集・印刷・発行）