.Deb

銀河系唯一のDebian専門誌

2025 年 11 月 15 日

WSGI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 MiniDebConf Japan 2025

2025 年 9 月 6 日 (土) に 北海道旭川市の旭川北洋ビル 北洋ホール 8F 小会議室で、日本では 9 年ぶりとなる MiniDebConf Japan 2025[*1] が開催されました。

イベントでは 9 つの発表[*2]と BoF を行い、参加者は合計 90 名でした (会場参加者は 30 名、リモート参加者は 60 名)。

イベント終了後は懇親会を実施し、Debian 13 trixie のリリースのお祝いも行いました。

1.2 OSC2025 Online/Fall (2025 年 10 月度 東京エリア Debian 勉強会)

2025 年 10 月 18 日 (土) に Debian JP Project ／東京エリア Debian 勉強会は、オープンソースカンファレンス 2025 Online/Fall[*3]に参加しました。イベント会場はオンライン会場（Zoom & YouTube Live）で、イベント当日の全体申し込みは 199 名でした。

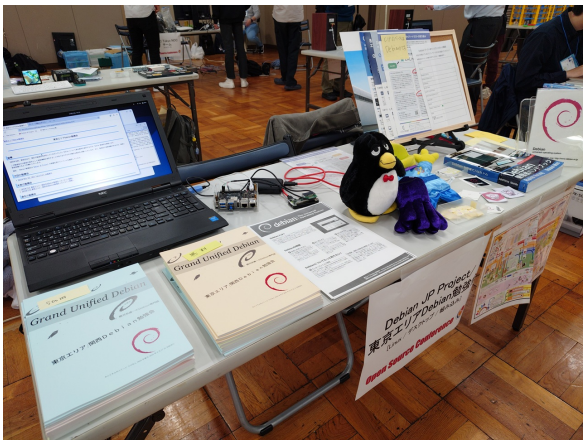
セミナーでは dictoss さんがタイトル「Debian Updates」の発表を行い[*4]、14 名が参加しました。セミナー内容は Debaian Project や Debian JP Project、Debian 勉強会、MiniDebConf Japan 2025、2025 年 8 月にリリースした Debian 13 trixie の情報を話しました。

[*1] https://tokyodebian-team.pages.debian.net/minidebconf-japan-2025/
[*2] 発表動画 https://www.youtube.com/live/t63YUYVnD8M
[*3] https://event.ospn.jp/osc2025-online-fall/
[*4] https://tokyodebian-team.pages.debian.net/pdf2025/debianmeetingresume202510-osc2025online-presentation.pdf

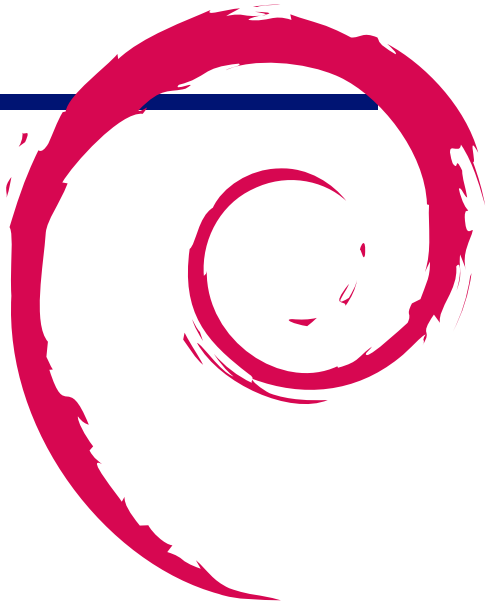
1.3 OSC2025 Tokyo/Fall

2025 年 10 月 25 日 (土) に Debian JP Project ／東京エリア Debian 勉強会は、オープンソースカンファレンス 2025 Tokyo/Fall[*5]に参加しました。イベント会場は東京都立産業貿易センター台東館 7F で、イベント当日の全体の参加者は 350 名でした。

イベントスペースに Debian ブースを出展して RISC-V64 のボード PC 等の展示や利用 OS のアンケートを行い、64 名がブースに訪れました。



[*5] https://event.ospn.jp/osc2025-fall/



2 事前課題

杉本 典充

今回の事前課題は以下です。

1. python で Web アプリケーションを作ってみたことはありますか。
2. Debian に関して調べていることがあれば教えてください。

2.1 dictoss

1. 作ったことがある
2. Debian 13 trixie のアップグレード作業情報

2.2 ebekei

1. 作ったことはない
2. (回答なし)

2.3 yy_y_ja_jp

1. 作ったことがある
2. (回答なし)

2.4 23corona

1. 作ったことがある
2. (回答なし)

2.5 su_do

1. 作ったことがある
2. (回答なし)

2.6 PenguinCabinet

1. 作ったことがある
2. (回答なし)

2.7 kuribayashi9

1. 作ったことがある
2. (回答なし)

2.8 kenhys

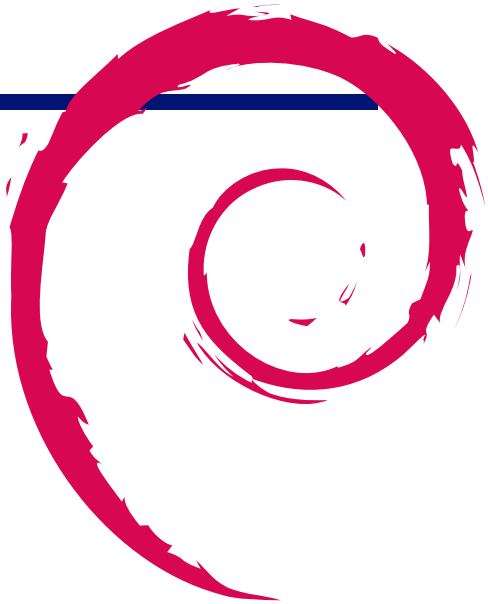
1. 作ったことはない
2. (回答なし)

2.9 koedoyoshida

1. 作ったことがある
2. (回答なし)

2.10 岡　大輔 (daisukeokaoss)

1. 作ったことはない
2. x86 で動く Linux アプリケーションを RISC-V または OpenPOWER で動かす



3 Python のいろいろな WSGI サーバを試してみる

杉本 典充

3.1 はじめに

本発表では、私がよく開発・運用している Python の Web アプリケーションを実行するための通信規格である WSGI(ウィスキー) と Debian 13 でパッケージを提供している WSGI サーバについて紹介します。

3.2 WSGI と WSGI サーバ

WSGI (Web Server Gateway Interface) とは Python を用いた Web アプリケーションにおいて Web サーバと Web アプリケーションを接続する通信規格です。WSGI は 2003 年に PEP 333[*6] として v1.0 が定義されたのが初出であり、現在は改定版である PEP 3333[*7] として v1.0.1 が定義されています。

3.3 Debian 13 で提供している WSGI サーバのパッケージ

3.3.1 mod_wsgi

mod_wsgi[*8] は web サーバである apache の拡張モジュールとして実装された WSGI サーバです。歴史は古く mod_wsgi-1.0 は 2007 年にリリースされており、この頃は python2.5 の時代です (2006-09-19 リリース)。

特徴として実行モードが embedded mode (1 プロセス、1 スレッド) と daemon mode (1 プロセス、マルチスレッド) の 2 種類あることです。Windows 環境では embedded mode のみ利用でき、Linux 環境では embedded mode と daemon mode の両方を使えますが daemon mode の利用が推奨になっています。

mod_wsgi を使って同時に実行できる WSGI アプリケーションは設定次第で複数実行することができます。

debian では昔は「libapache2-mod-wsgi-py」パッケージ (python2 向け) が提供され、現在の debian 13 では「libapache2-mod-wsgi-py3」パッケージ (python3 向け) が提供されています[*9]。

3.3.2 uwsgi

uwsgi [*10]はデーモンとして動作することを想定して実装された WSGI サーバです。2010 年頃に開発が始まり、2011 年 12 月 29 日に バージョン 1.0 がリリースされ、最新版は 2025-10-12 にリリースした uwsgi-2.0.31 です。なお、現在の開発はメンテナンスモード扱いになっており、軽微な修正のみを受け付けています。

[*6] https://peps.python.org/pep-0333/
[*7] https://peps.python.org/pep-3333/
[*8] https://modwsgi.readthedocs.io/en/master/
[*9] 「libapache2-mod-wsgi-py」と「libapache2-mod-wsgi-py3」を同時にパッケージ提供していた Debian のバージョンもありますが、同時利用は不可になっています。
[*10] https://uwsgi-docs.readthedocs.io/en/latest/

特徴としてはプラグイン方式を採用しており、python 以外にも ruby や php、java なども混在して実行することができます (python のバージョンごとにプラグインがあれば、複数の python のバージョンを混在して実行することも可能です)。

uwsgi を使って同時に実行できる WSGI アプリケーションは設定次第で複数実行することができます。

debian では 2011 年に uwsgi パッケージとして uwsgi-0.9.8 がアップロードされたのが最初であり、debian 13 では uwsgi-2.0.28 がパッケージとして提供されています。

3.3.3 gunicorn

gunicorn [*11] は単独プロセスで動作する WSGI サーバです。gunicorn は "Green Unicorn" の略になっており、Unicorn という ruby 向けのアプリケーションサーバの python 移植版です。web サイトにある changelog では 2010 年に 0.5.0 がリリースされ[*12]、最新版は 2024-08-10 にリリースした gunicorn-23.0.0 です[*13]。

実装しているプログラミング言語は python のみ書かれているのが特徴です。

debian では 2010 年に gunicorn パッケージとして gunicorn-0.7.1 がアップロードされたのが最初であり、debian 13 では gunicorn-23.0.0 がパッケージとして提供されています。

3.4 Debian 13 で WSGI サーバを実行してみる

3.4.1 実行する WSGI サーバと venv 環境

実行する Web アプリケーションは WSGI サーバに対応している django という Python 向け Web フレームワークのチュートリアルのプログラムを利用します。ソースコードは以下にあります。

- `https://github.com/dictoss/django-tutorial`
- Django-5.2 LTS を使ったサンプルアプリケーション

Debian 13 では Python のライブラリをシステムワイドにインストールすることは非推奨になっているため、事前に venv 環境を作っておきます。

```
# apt-get install python3 python3.13-venv
# mkdir -p /var/www/wsgi_apps
# cd /var/www/wsgi_apps
# python3 -m venv venv_django-tutorial
# ls venv_django-tutorial
bin  include  lib  lib64  pyvenv.cfg
```

venv 環境に入り、django をインストールします。

```
# cd venv_django-tutorial
# source bin/activate
(venv_django-tutorial) # pip install django==5.2.8
(venv_django-tutorial) # deactivate
#
```

3.4.2 mod_wsgi

mod_wsgi は apache2 の拡張モジュールのため、apache2 と一緒に拡張モジュールのパッケージをインストールします。

```
# apt-get install apache2 libapache2-mod-wsgi-py3
```

apache2 の wsgi モジュールを有効にします。

[*11] `https://gunicorn.org/`
[*12] `https://docs.gunicorn.org/en/latest/2010-news.html`
[*13] `https://docs.gunicorn.org/en/latest/2024-news.html`

```
# a2enmod wsgi
Enabling module wsgi.
To activate the new configuration, you need to run:
  systemctl restart apache2
```

/etc/apache2/ 配下 に設定ファイル wsgi_django-tutorial.conf を作成します。

```
# vi /etc/apache2/wsgi_django-tutorial.conf

    # for wsgi settings.
    WSGIScriptAlias  /django-tutorial-apache2 \
        /var/www/wsgi_apps/django-tutorial-apache2/5.2/djangotutorial/mysite/wsgi.py
    WSGIDaemonProcess wsgi-djangotutorial \
        user=www-data group=www-data display-name=%{GROUP} processes=4 threads=16 maximum-requests=10240 \
        python-home=/var/www/wsgi_apps/venv_django-tutorial \
        python-path=/var/www/wsgi_apps/django-tutorial-apache2/5.2/djangotutorial
    WSGIProcessGroup  wsgi-djangotutorial

    <Directory /var/www/wsgi_apps/django-tutorial-apache2/5.2/djangotutorial/mysite>
        <Files wsgi.py>
            Require all granted
        </Files>
    </Directory>

    # for wsgi application static file settings.
    Alias /static/django-tutorial/ \
        "/var/www/wsgi_apps/django-tutorial-apache2/5.2/djangotutorial/polls/static/polls/"
```

000-default.conf ファイルに作成した wsgi_django-tutorial.conf を Include する設定を追加します。

```
# vi /etc/apache2/sites-available/000-default.conf

        Include wsgi_django-tutorial.conf
</VirtualHost>
```

apache2 を再起動し、WSGI サーバが一緒に実行されます。

```
# systemctl restart apache2
# ps aux | grep wsgi | grep -v grep
www-data    7472  0.0  2.7 333880 53440 ?        Sl   00:43   0:00 (wsgi:wsgi-django -k start
www-data    7473  0.0  2.7 333880 53440 ?        Sl   00:43   0:00 (wsgi:wsgi-django -k start
www-data    7474  0.0  2.7 399440 53704 ?        Sl   00:43   0:00 (wsgi:wsgi-django -k start
www-data    7475  0.0  2.7 333880 53440 ?        Sl   00:43   0:00 (wsgi:wsgi-django -k start
```

http://hostname/django-tutorial-apache2/ へアクセスすると、Web アプリケーションの画面を表示できます。

3.4.3 uwsgi

uwsgi の設定

uwsgi を使うには uwsgi パッケージと実行するアプリケーションの実行環境である python3 向けのプラグインをインストールします。また、uwsgi を WSGI サーバとして実行すると HTTP を直接受け付けないため (設定で HTTP サーバとして実行するモードもあります)、HTTP のリクエストを受けて WSGI サーバへ中継する機能をもつ web サーバである nginx をインストールします。

```
# apt-get install uwsgi uwsgi-plugin-python3
# apt-get install nginx
```

/etc/uwsgi の下には apps-available と apps-enabled のディレクトリがあります。apps-available は設定ファイルの実体、apps-enabled は有効無効を切り替えるために apps-available の設定ファイルのシンボリックリンクを置くよう README に書いてあります。

まずは、uwsgi で WSGI サーバのアプリケーションを実行する設定ファイルを作成します。

```
# cd /etc/uwsgi/apps-available/
# vi django-tutorial.ini

[uwsgi]
uid = www-data
gid = www-data
plugin-dir = /usr/lib/uwsgi/plugins
plugin = python3
base  = /var/www/wsgi_apps/django-tutorial-uwsgi/5.2/djangotutorial
chdir = /var/www/wsgi_apps/django-tutorial-uwsgi/5.2/djangotutorial
home  = /var/www/wsgi_apps/venv_django-tutorial
module = mysite.wsgi
callable = application
env =
socket = 127.0.0.1:3031
thunder-lock = true
processes = 4
threads = 16
master = True
vacuum = True
harakiri = 60
max-requests = 10240
max-requests-delta = 64
post-buffering = 8192
```

次に apps-enabled ディレクトリに django-tutorial.ini のシンボリックリンクを作成します。

```
# cd ../apps-enabled
# ln -s ../apps-available/django-tutorial.ini .
# ls -l
合計 4
-rw-r--r-- 1 root root 424 11 月 25  2023 README
lrwxrwxrwx 1 root root  37 11 月 14 21:43 django-tutorial.ini -> ../apps-available/django-tutorial.ini
```

uwsgi を再起動します。

```
# systemctl restart uwsgi
# ps aux | grep uwsgi | grep -v grep
www-data    5527  0.0  2.3  61852 46876 ?        S    11 月 14   0:03
  /usr/bin/uwsgi --ini /usr/share/uwsgi/conf/default.ini --ini /etc/uwsgi/apps-enabled/django-tutorial.ini
   --daemonize /var/lo/uwsgi/app/django-tutorial.log
www-data    5534  0.0  2.5 1169428 49444 ?       Sl   11 月 14   0:03
  /usr/bin/uwsgi --ini /usr/share/uwsgi/conf/default.ini --ini /etc/uwsgi/apps-enabled/django-tutorial.ini
  --daemonize /var/lo/uwsgi/app/django-tutorial.log
www-data    5537  0.0  2.5 1169496 49580 ?       Sl   11 月 14   0:03
  /usr/bin/uwsgi --ini /usr/share/uwsgi/conf/default.ini --ini /etc/uwsgi/apps-enabled/django-tutorial.ini
   --daemonize /var/lo/uwsgi/app/django-tutorial.log
www-data    5540  0.0  2.5 1169692 49564 ?       Sl   11 月 14   0:03
  /usr/bin/uwsgi --ini /usr/share/uwsgi/conf/default.ini --ini /etc/uwsgi/apps-enabled/django-tutorial.ini
   --daemonize /var/lo/uwsgi/app/django-tutorial.log
www-data    5556  0.0  2.5 1169692 49400 ?       Sl   11 月 14   0:03
  /usr/bin/uwsgi --ini /usr/share/uwsgi/conf/default.ini --ini /etc/uwsgi/apps-enabled/django-tutorial.ini
   --daemonize /var/lo/uwsgi/app/django-tutorial.log
```

uwsgi 向けの nginx 設定

次は nginx で パス「/django-tutorial-uwsgi/」へアクセスしたときに uwsgi へ中継するよう default ファイルに設定を追加します。

```
# vi /etc/nginx/sites-available/default

    location ~ ^/django-tutorial-uwsgi/(.*)$ {
        include uwsgi_params;
        uwsgi_param SCRIPT_NAME /django-tutorial-uwsgi;
        uwsgi_param PATH_INFO /$1;
        uwsgi_pass 127.0.0.1:3031;
    }
```

nginx を再起動します。

```
# systemctl restart nginx
```

http://hostname/django-tutorial-uwsgi/ へアクセスすると、Web アプリケーションの画面を表示できます。

3.4.4 gunicorn

gunicorn の設定

gunicorn を使う場合はホスト環境で実行するときは Debian パッケージの gunicorn を使い、python の venv 環境で実行するときは venv 環境で pip install した gunicorn を使うことになっています。

今回は venv 環境のため、pip で gunicorn をインストールします。

```
# cd /var/www/wsgi_apps/venv_django-tutorial
# source bin/activate
(venv_django-tutorial) # pip install gunicorn==23.0.0
(venv_django-tutorial) # find bin -name gunicorn
bin/gunicorn
(venv_django-tutorial) # deactivate
#
```

gunicorn は単体の実行ファイルになっているため、systemd で起動する方法はデフォルトで提供されていません[*14]。今回は /usr/local にディレクトリを作成して設定ファイルを書き、フォアグラウンドで gunicorn を実行することにします。

```
# mkdir /usr/local/django-tutorial-gunicorn
# cd /usr/local/django-tutorial-gunicorn
# mkdir logs
# chown -fR www-data:www-data logs
```

gunicorn.conf.py の bind 設定値が WSGI サーバ兼 HTTP サーバの待ち受けポートになります。

```
# vi gunicorn.conf.py

#
# Gunicorn config file
#
wsgi_app = 'mysite.wsgi'
chdir = '/var/www/wsgi_apps/django-tutorial-gunicorn/5.2/djangotutorial'

daemon = False
raw_env = [
#    'SCRIPT_NAME=/django-tutorial-gunicorn',
]
bind = '127.0.0.1:8081'

user = 'www-data'
group = 'www-data'

# for sync
worker_class = 'sync'
workers = 4

# for gthread
#worker_class = 'gthread'
#workers = 4
#threads = 16

max_requests = 10240

accesslog = '/usr/local/django-tutorial-gunicorn/logs/access.log'
access_log_format = '%(h)s %(l)s %(u)s %(t)s "%(r)s" %(s)s %(b)s "%(f)s" "%(a)s"'

# gunicorn log
#errorlog = '/usr/local/django-tutorial-gunicorn/logs/error.log'
errorlog = '-'
loglevel = 'info'
```

gunicorn をフォアグランドで実行します。

```
# /var/www/wsgi_apps/venv_django-tutorial/bin/gunicorn \
    --config /usr/local/django-tutorial-gunicorn/gunicorn.conf.py
[2025-11-15 11:28:39 +0900] [8337] [INFO] Starting gunicorn 23.0.0
[2025-11-15 11:28:39 +0900] [8337] [INFO] Listening at: http://127.0.0.1:8081 (8337)
[2025-11-15 11:28:39 +0900] [8337] [INFO] Using worker: sync
[2025-11-15 11:28:39 +0900] [8338] [INFO] Booting worker with pid: 8338
[2025-11-15 11:28:39 +0900] [8339] [INFO] Booting worker with pid: 8339
[2025-11-15 11:28:39 +0900] [8340] [INFO] Booting worker with pid: 8340
[2025-11-15 11:28:39 +0900] [8341] [INFO] Booting worker with pid: 8341
```

http://hostname:8081/ へアクセスすると、Web アプリケーションの画面を表示できます。

gunicorn 向けの nginx 設定

[*14] マニュアルに systemd で実行する例は書いてあります。https://docs.gunicorn.org/en/latest/deploy.html#systemd

gunicorn は WSGI サーバですが、HTTP サーバでもあります。直接ポート番号へ Web ブラウザでアクセスして Web ページ を見ることはできましたが、gunicorn の前に置いた web サーバを中継してアクセスすることが多いです。

今回は nginx でパス「/django-tutorial-gunicorn/」にアクセスしたときに gunicorn の HTTP サーバへ中継するよう default ファイルに設定を追加します。この設定は uwsgi のときと異なり、普通に HTTP をリバースプロキシするようにします。

```
# vi /etc/nginx/sites-available/default

    location ~ ^/django-tutorial-gunicorn/(.*)$ {
        include proxy_params;
        proxy_redirect off;
        proxy_buffering off;
        proxy_pass http://127.0.0.1:8081;
    }
```

nginx を再起動します。

```
# systemctl restart nginx
```

gunicorn で実行しているアプリケーションへのアクセスで、リバースプロキシによりパスの上にサブディレクトリが置かれる場合は環境変数「SCRIPT_NAME」を設定する必要があります。

```
# vi gunicorn.conf.py

raw_env = [
     'SCRIPT_NAME=/django-tutorial-gunicorn',
]
```

設定を修正して、gunicorn をフォアグランドで実行します。

```
# /var/www/wsgi_apps/venv_django-tutorial/bin/gunicorn \
    --config /usr/local/django-tutorial-gunicorn/gunicorn.conf.py
[2025-11-15 11:28:39 +0900] [8337] [INFO] Starting gunicorn 23.0.0
[2025-11-15 11:28:39 +0900] [8337] [INFO] Listening at: http://127.0.0.1:8081 (8337)
[2025-11-15 11:28:39 +0900] [8337] [INFO] Using worker: sync
[2025-11-15 11:28:39 +0900] [8338] [INFO] Booting worker with pid: 8338
[2025-11-15 11:28:39 +0900] [8339] [INFO] Booting worker with pid: 8339
[2025-11-15 11:28:39 +0900] [8340] [INFO] Booting worker with pid: 8340
[2025-11-15 11:28:39 +0900] [8341] [INFO] Booting worker with pid: 8341
```

http://hostname/django-tutorial-gunicorn/ へアクセスすると、Web アプリケーションの画面を表示できます。

3.5 WSGI サーバの比較

3.5.1 機能の比較

Debian 13 において、同じ WSGI の機能をもつパッケージが 3 つもあるとどれを選択してよいか迷ってしまいます。選択のポイントは個人的には以下の基準で使い分けるとよいと思いました。

- mod_wsgi は apache2 を使いたい人、デーモンを apache2 のみで完結させたい小規模運用向け
- uwsgi は 仮想マシンベースのスケールアウト構成向け (nginx から複数台の uwsgi サーバへ負荷分散できる)
- gunicorn はコンテナベースの実行環境向け (Dockerfile の RUN で gunicorn を直接起動する)

また、それぞれの WSGI サーバの機能を比較する表をつくってみました (表 1)

表 1　WSGI サーバの機能比較

WSGI サーバ	mod_wsgi	uwsgi	gunicorn
実装方法	C 言語 (apache2 拡張)	C 言語 +python	python
アプリのデーモン実行	標準で systemd 動作	標準で systemd 動作	systemd は要設定
実行できるアプリ数	複数同時実行	複数同時実行	1 つのみ (複数起動で可)
前段の Web サーバ	apache2 のみ	WSGI/HTTP 対応 Web サーバ	HTTP 対応 Web サーバ
HTTP 対応	対応 (apache2)	対応	対応
複数 python バージョン対応	なし	あり	複数起動で対応
1 プロセスのスレッド数	1 つ/複数	1 つ/複数	1 つ/複数

3.5.2　ベンチマークの比較

選択する基準で処理性能を優先したい方もいると思います。簡単なベンチマークを取ってみました (表 2)。
テスト環境は以下です。

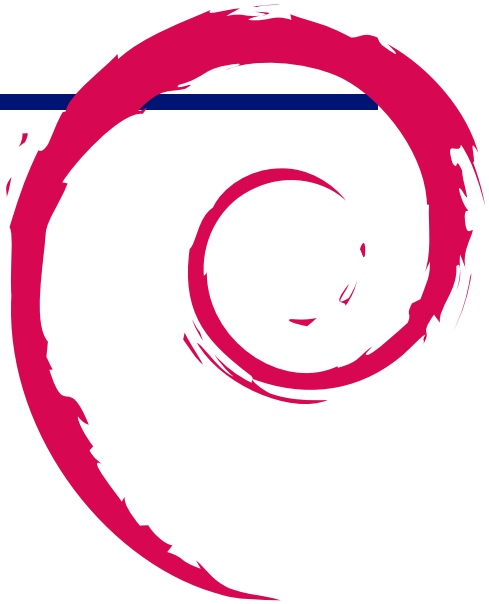
- サーバ
  - 実行するサーバは仮想サーバで 2 vCPU (Core i3-12100T 搭載 Windows 11 Pro の Hyper-V)
  - WSGI サーバのプロセス数、スレッド数は表 2 に記載
  - アクセスするページは /django-tutorial-apcach2/polls/、/django-tutorial-uwsgi/polls/、/django-tutorial-gunicorn/polls/
- クライアント
  - サーバと同一物理ホストの別仮想サーバから ab コマンドを実行
  - ab -c 8 -n 1000 url

表 2　WSGI サーバのベンチマーク

WSGI サーバ	mod_wsgi	uwsgi	gunicorn
web サーバ	apache2(event)	nginx	nginx
WSGI のプロセス数	4	4	4
WSGI のスレッド数	16	16	1
Requests per second	1041.41	1219.76	1193.47

3.6　おわりに

Debian 13 でパッケージを提供している WSGI サーバの mod_wsgi、uwsgi、gunicorn を紹介しました。特徴や Web サーバとの組み合わせるでどれを使うとよいか検討し、自分に合う WSGI サーバを使って Python の Web アプリケーションを動かしてみるとよいでしょう。



4 メモ

Debian 勉強会資料

2025 年 11 月 15 日　　初版第 1 刷発行

東京エリア Debian 勉強会 （編集・印刷・発行）