


ХОЧУ ПОМОЧЬ
ПРОЕКТУ



skillbox.ru

Станьте «DevOps-инженером» всего за 7 месяцев!
Обучим DevOps с гарантией трудоустройства. Трудоустроим – или вернем деньги.
[Узнать больше](#)

kata.academy

Курсы frontend-разработки. Оплата после трудоустройства

4,1 Рейтинг организации

Каждый наш выпускник получает предложение о работе и ЗП на 30% выше ожидаемой.

Для кого

Модель ISA

[Узнать больше](#)

Модуль pyftplib в Python, FTP-сервер

Содержание:

- Установка pyftplib в виртуальное окружение;
- Использование командной строки для запуска FTP-сервера;
- Базовый FTP-сервер;
- Ведение логов FTP-сервера;
- Хранение паролей в виде хэш-дайджестов;
- Ограничение скорости загрузки и выгрузки данных;
- FTPS-сервер (FTP через TLS/SSL);
- FTP-сервер с авторизацией пользователей Linux;
- FTP-сервер с авторизацией пользователей Windows;
- Изменение модели параллелизма FTP-сервера.

Установка pyftplib в виртуальное окружение.

Так как модуль pyftplib не входит в [стандартную библиотеку Python](#), его необходимо установить отдельно. Сделать это можно с помощью [менеджера пакетов pip](#).

```
# создаем виртуальное окружение, если нет
$ python3 -m venv .venv --prompt VirtualEnv
# активируем виртуальное окружение
$ source .venv/bin/activate
# обновляем `pip`
(VirtualEnv):~$ python3 -m pip install -U pip
# ставим модуль `pyftplib`
(VirtualEnv):~$ python3 -m pip install pyftplib -U
```

Использование командной строки для запуска FTP-сервера.

Модуль pyftplib можно запускать как простой автономный сервер с помощью опции python3 -m, что особенно полезно, когда необходимо быстро поделиться каталогом.

Анонимный FTPd, использующий текущий каталог:

\$


Вверх

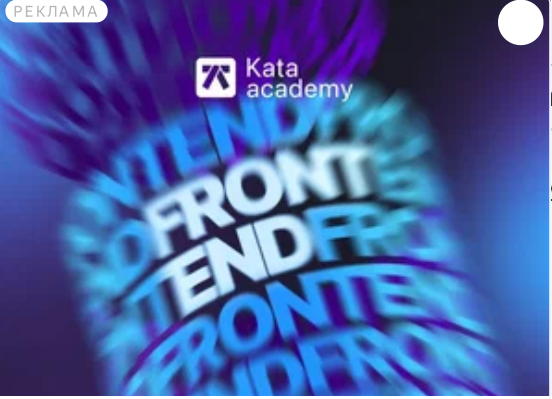
-m pyftplib


Анонимный FTPd с разрешением на запись:

```
$ python3 -m pyftplib -w
```


РЕКЛАМА

**Kata academy**



kata.academy

Курсы frontend-разработки. Оплата после трудоустройства

4,1  Рейтинг организации ⓘ

Каждый наш выпускник получает предложение о работе и ЗП на 30% выше ожидаемой.

Для кого

Модель ISA

Узнать больше

домашнего каталога:

```
host -p 8021 -d /home/someone
```

МИ:

ESS: указать интерфейс для запуска (по умолчанию все интерфейсы);
е номер порта для запуска (по умолчанию 2121);
оступ на запись для вошедшего в систему пользователя (по умолчанию только для чтения);
R: каталог для общего доступа (текущий каталог по умолчанию);
DRESS: адрес NAT для пассивных подключений;
диапазон TCP-портов, используемых для пассивных подключений (например, -r 8000-9000);
журнала DEBUG;
более подробное ведение журнала;
NAME: имя пользователя для входа в систему (анонимный вход будет отключен, и потребуется
WORD: пароль для входа в систему (имя пользователя должно быть полезным).

льзуется базовая конфигурация, и это, вероятно, лучшая отправная точка для понимания
ер использует базовый [DummyAuthorizer](#) для добавления группы "виртуальных" пользователей и
соединения и диапазон пассивных портов.

```
from pyftplib.authorizers import DummyAuthorizer
from pyftplib.handlers import FTPHandler
from pyftplib.servers import FTPServer

def main():
    # экземпляр фиктивного средства авторизации
    # для управления "виртуальными" пользователями
    authorizer = DummyAuthorizer()

    # добавляем нового пользователя, имеющего полные права доступа `r/w`
    # и анонимного пользователя, для которого FS доступна только для чтения
    authorizer.add_user('user', '12345', '/home/user/some-dir', perm='elradfmwMT')
    authorizer.add_anonymous(os.getcwd())

    # экземпляр класса обработчика FTP
    handler = FTPHandler
    handler.authorizer = authorizer

    # настраиваемый баннер (строка, возвращаемая при подключении клиента)
    handler.banner = "pyftplib основанный на ftpd."

    # masquerade-адрес и диапазон портов, которые будут использоваться
    # для пассивных подключений. Строки ниже нужно раскомментировать
    # если вы находитесь за NAT (masquerade_address укажите свой).
    # handler.masquerade_address = '151.25.42.11'
    # handler.passive_ports = range(60000, 65535)

    # экземпляр класса FTP-сервера, который слушает `0.0.0.0:2121`
    address = ('', 2121)
    server = FTPServer(address, handler)

    # лимиты на соединения
    server.max_cons = 256
    server.max_cons_per_ip = 5

    server.serve_forever()
```

Вверх


https://docs-python.ru/packages/modul-pyftplib-python/

2/6


```
if __name__ == '__main__':
    main()
```

Ведение логов FTP-сервера.

РЕКЛАМА

**Kata academy**

Курсы frontend-разработки. Оплата после трудоустройства

4,1  Рейтинг организации ⓘ

Каждый наш выпускник получает предложение о работе и ЗП на 30% выше ожидаемой.

Для кого

Модель ISA

Узнать больше

Для ведения логов используется модуль logging для ведения журнала логов. Если не настроить логирование, то журнал будет пустым. Включение ведения журнала необходимо до вызова `server.serve_forever()`.

```
handler = FTPHandler
server = FTPServer
auth = DummyAuthorizer

auth.add_user('user', 'password', '45', '.', perm='elradfmwMT')

logging.basicConfig(filename='pyftpd.log',
                    filemode='a',
                    format='%(asctime)s,%(msecs)d %(levelname)s %(message)s',
                    datefmt='%m-%d-%Y %H:%M:%S',
                    level=logging.INFO)
```

Модуль logging будет выводить все команды и ответы, которыми обмениваются клиент и FTP-сервера. Также будут выводиться внутренние ошибки, которые могут возникнуть при вызовах, связанных с сокетами, таких как `socket.error`.

Для включения режима DEBUG из кода, используйте:

```
logging.basicConfig(level=logging.DEBUG)
```

Для включения режима DEBUG из командной строки, используйте:

```
$ python -m pyftplib -D
```

Изменение префикса строки журнала.

```
handler = FTPHandler
handler.log_prefix = 'XXX [%s]@[%s]' % (username, remote_ip)
server = FTPServer(('localhost', 2121), handler)
server.serve_forever()
```

Хранение паролей в виде хэш-дайджестов.

Использование FTP-сервера по умолчанию с [DummyAuthorizer](#) означает, что пароли будут храниться в открытом виде, а хранение паролей в открытом виде, конечно, нежелательно. Самый простой способ избежать подобного сценария - сначала создать новых пользователей, а затем сохранить их имена пользователей и пароли в виде хэш-дайджестов в файле или там, где это удобно. В приведенном ниже примере показано, как хранить пароли в виде односторонних хешей с использованием алгоритма SHA1 ([hashlib.sha1](#)).

```
import os
import sys
from hashlib import sha1

from pyftplib.handlers import FTPHandler
from pyftplib.servers import FTPServer
from pyftplib.authorizers import DummyAuthorizer, AuthenticationFailed

class DummySHA1Authorizer(DummyAuthorizer):
    def validate_authentication(self, username, password, handler):
        hash = sha1(password.encode('utf-8')).hexdigest()
        try:
            if self.user_table[username]['pwd'] != hash:
                raise KeyError
        except KeyError:
```

Вверх

[illegible]

Курсы frontend-разработки. Оплата после трудоустройства

4,1 ★ Рейтинг организации ⓘ

Каждый наш выпускник получает предложение о работе и ЗП на 30% выше ожидаемой.

Для кого

Модель ISA

Узнать больше

```
def main():
    authorizer = DummyAuthorizer()
    authorizer.add_user('user', '12345', os.getcwd(), perm='elradfmwMT')
    authorizer.add_anonymous(os.getcwd())

    dtp_handler = ThrottledDTPHandler
    dtp_handler.read_limit = 30720 # 30 Kb/sec (30 * 1024)
    dtp_handler.write_limit = 30720 # 30 Kb/sec (30 * 1024)

    ftp_handler = FTPHandler
    ftp_handler.authorizer = authorizer
    # обработчик ftp использует альтернативный класс обработчика dtp
    ftp_handler.dtp_handler = dtp_handler

    server = FTPServer(('', 2121), ftp_handler)
    server.serve_forever()

if __name__ == '__main__':
    main()
```

Модуль `pyftplib` включает полную поддержку FTPS, реализующую протоколы TLS и SSL, а также команды AUTH, PBSZ и PROT, как определено в RFC-4217. Это поведение реализовано с помощью модуля `PyOpenSSL`, необходимого для запуска приведенного ниже кода. Для класса `TLS_FTPHandler` требуется указать как минимум файл сертификата и, возможно, ключевой файл.

```

"""
Асинхронный FTPS-сервер, поддерживающий SSL и TLS.
Требуется модуль PyOpenSSL (http://pypi.python.org/pypi/pyOpenSSL).
"""

from pyftplib.servers import FTPServer
from pyftplib.authorizers import DummyAuthorizer
from pyftplib.handlers import TLS_FTPHandler


def main():
    authorizer = DummyAuthorizer()
    authorizer.add_user('user', '12345', '.', perm='elradfmwMT')
    authorizer.add_anonymous('.')
    handler = TLS_FTPHandler
    handler.certfile = 'cert.pem'

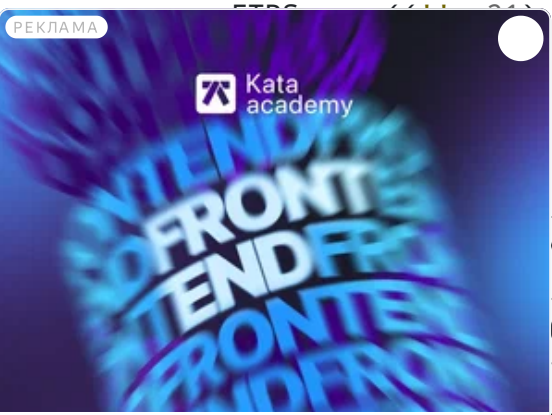
```




```
handler.authorizer = authorizer
# requires SSL for both control and data channel
#handler.tls_control_required = True
#handler.tls_data_required = True
server = FTPServer(('', 21), handler)
```



РЕКЛАМА

 Kata academy



 kata.academy

Курсы frontend-разработки. Оплата после трудоустройства

4,1  Рейтинг организации 

Каждый наш выпускник получает предложение о работе и ЗП на 30% выше ожидаемой.

Для кого

Модель ISA

Узнать больше

Сертификат с помощью OpenSSL можно командой:

```
openssl req -x509 -keyout key.pem -out cert.pem -sha256 -days 365
```

Вместо парольной фразой, то можно добавить опцию `-nodes` (сокращение от "no DES"). В противном случае придется ввести пароль не менее 4 символов.

Срок действия (по умолчанию 365 дней) можно заменить любым числом. Затем будет предложено ввести такие данные, как организация и т.д. - это можно игнорировать просто нажав *"Enter"*, тем самым принять значения по умолчанию.

При создании сертификата, можно добавить опцию `-subj '/CN=localhost'` (замените localhost на имя хоста).

Сертификаты, созданные таким образом, не будут проверяться какой-либо третьей стороной, если предварительно не импортировать их в браузер. Для обеспечения безопасности, то необходимо использовать сертификат, подписанный центром сертификации.

Для пользователей Linux.

Для запуска своего FTP-сервера с поддержкой реальных пользователей, существующих в системе, и для аутентификации в этой системе. В приведенном ниже примере для этого используются классы `UnixAuthorizer` и `UnixFilesystem`.

```
from pyftplib.handlers import FTPHandler
from pyftplib.servers import FTPServer
from pyftplib.authorizers import UnixAuthorizer
from pyftplib.filesystems import UnixFilesystem

def main():
    authorizer = UnixAuthorizer(rejected_users=["root"], require_valid_shell=True)
    handler = FTPHandler
    handler.authorizer = authorizer
    handler.abstracted_fs = UnixFilesystem
    server = FTPServer(('', 21), handler)
    server.serve_forever()

if __name__ == "__main__":
    main()
```

FTP-сервер с авторизацией пользователей Windows.

В следующем коде показано, как реализовать базовый авторизатор для рабочей станции Windows для аутентификации по существующим учетным записям пользователей Windows. Этот код требует установки модуля `pywin32`.

```
from pyftplib.handlers import FTPHandler
from pyftplib.servers import FTPServer
from pyftplib.authorizers import WindowsAuthorizer

def main():
    authorizer = WindowsAuthorizer()
    # для обработки анонимных сеансов можно использовать пользователя
    # Guest с пустым паролем. Дополнительно можно указать каталог профиля.
    # authorizer = WindowsAuthorizer(anonymous_user="Guest", anonymous_password="")
    handler = FTPHandler
    handler.authorizer = authorizer
    server = FTPServer(('', 2121), handler)
    server.serve_forever()

if __name__ == "__main__":
    main()
```

Изменение модели параллелизма FTP-сервера.

По своей природе pyftplib является асинхронным. Это означает, что он использует один процесс/поток для обработки запросов и передачи файлов. Вот почему он такой быстрый, легкий и масштабируемый. Однако у такой модели есть один большой недостаток: в коде не должно быть инструкций, блокирующих на длительный период FTP-сервер. Таким образом, пользователь должен избегать таких вызовов, как `time.sleep(3)`, `input()` и т. д. Более того, есть случаи, когда асинхронная модель не подходит - медленная файловая система (например, samba).

Если операция (например, `open(file, 'r').read(8192)` занимает 2 секунды), то FTP-сервер застрянет. Для решения этой проблемы есть классы, которые изменяют модель параллелизма по умолчанию, добавляя несколько потоков или процессов. Это означает, что когда клиент подключается, создается отдельный поток/процесс, и сервер продолжает свой цикл ввода-вывода. На практике это означает, что можно использовать блокирующие вызовы, не мешая другим клиентам.

Для этого нужно вместо класса `FTPServer` использовать `ThreadedFTPServer` или `ProcessedFTPServer`.

РЕКЛАМА

Kata academy



kata.academy

Курсы frontend-разработки. Оплата после трудоустройства

4,1

★

Рейтинг организации



Каждый наш выпускник получает предложение о работе и ЗП на 30% выше ожидаемой.

Для кого

>

Модель ISA

>

Узнать больше

```
server = ThreadedFTPServer(('', 2121), handler)
server.serve_forever()

if __name__ == "__main__":
    main()
```

Пример на основе процессоров:

```
from pyftplib.handlers import FTPHandler
from pyftplib.servers import MultiprocessFTPServer # <-
from pyftplib.authorizers import DummyAuthorizer

def main():
    authorizer = DummyAuthorizer()
    authorizer.add_user('user', '12345', '.')
    handler = FTPHandler
    handler.authorizer = authorizer
    server = MultiprocessFTPServer(('', 2121), handler)
    server.serve_forever()

if __name__ == "__main__":
    main()
```

Содержание раздела:

- [КРАТКИЙ ОБЗОР МАТЕРИАЛА.](#)
- [Класс DummyAuthorizer\(\) модуля pyftplib](#)
- [Класс FTPHandler\(\) модуля pyftplib](#)