4.2 Публикация статического веб-сайта с Pyodide

Публикация статического веб-сайта с

Pyodide

Flet приложение можно опубликовать в виде автономного статического веб-сайта (SPA) и запустить полностью в браузере с помощью Pyodide.

Pyodide - это порт CPython в WebAssembly (WASM), который является новой технологией со своими ограничениями. Pyodide включает в себя большой список встроенных пакетов. Однако, для использования пакета Python из PyPI он должен быть чистым Python-пакетом или предоставлять колесо с бинарниками, построенными для Emscripten.

# Статический флет в сравнении с серверным

Преимущества статического Flet:

* Нулевая задержка между событиями, генерируемыми пользователем (клики, изменения текстового поля, перетаскивание) и обновлениями страницы. Нет сервера Fletd, нет WebSockets - Python-программа общается с веб-клиентом Flutter напрямую через JavaScript.
* Дешевый хостинг - статическое приложение Flet не требует никакого кода для работы на сервере и, таким образом, может быть размещено где угодно: GitHub Pages, Cloudflare Pages, Replit, Vercel, общий хостинг или ваш собственный VPS.
* Более высокая масштабируемость - статическое приложение Flet работает полностью в браузере и, если оно не использует никакого API с серверной стороны, оно может быть масштабировано для любого числа пользователей даже только при помощи CDN.

Недостатки статического Flet:

* Более медленное время загрузки - требуется дополнительное время для загрузки Python движка (Pyodide), встроенных и пакетов flet-pyodide , а

также пользовательской программы. Кроме того, инициализация самого движка Pyodide занимает около 2-4 секунд, что команда Flet планирует улучшить в будущем.

* Ограниченная совместимость с Python - не каждая программа, которая работает с нативным Python, может быть запущена с Pyodide.
* Ниже производительность - на данный момент Pyodide медленнее нативного Python в 3-5 раз, поэтому приложения Flet с большим объемом обработки лучше развернуть на стороне сервера.
* Пользователи могут получить доступ к коду программы, так как он загружается в браузер в форме архива tar.gz .

# Асинхронный или не асинхронный?

Как asyncio , так и "обычные" синхронные приложения Flet могут быть опубликованы в виде статического веб-сайта. В терминах параллелизма, на веб-

сайте будет только один поток с одним пользователем - вами. Если ваше приложение содержит вычислительно сложную логику, она может повлиять на отзывчивость пользовательского интерфейса, независимо от того, является ли приложение асинхронным или нет.

Однако, если ваше приложение содержит I/O логику (например, обертку

fetch

Pyodide), которая в браузере по определению является асинхронной, то ваше приложение должно быть асинхронным.

для

# Публикуем приложение в виде статического веб- сайта

Выполните следующую команду для публикации Flet приложения на автономном веб-сайте:

flet publish <your-flet-app.py>

Статический веб-сайт опубликовывается в директории ./dist .

Опциональные коммандные аргументы:

* - позволяет micropip устанавливать предварительные версии Python пакетов.

--pre

-a ASSETS\_DIR, --assets ASSETS\_DIR

* - путь к директории ресурсов.
* - название приложения.

--app-title APP\_TITLE

* --app-description APP\_DESCRIPTION - описание приложения.
* - базовый URL для приложения.

--base-url BASE\_URL

* --web-renderer {canvaskit,html} - используемый веб-рендерер.
* - стратегия маршрутизации URL.

--route-url-strategy {path,hash}

# Тестирование веб-сайта

Вы можете протестировать опубликованное приложение Flet, используя встроенный в Python модуль http.server :

python -m http.server --directory dist

Откройте [http://localhost:8000](http://localhost:8000/) в вашем браузере, чтобы проверить опубликованное приложение.

# Загрузка пакетов

Вы можете загрузить пользовательские пакеты из PyPI во время старта приложения,

перечислив их в

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| requirements.txt | | . Файл | requirements.txt | |
|  | <your-flet-app.py> | | | . |

в той же директории, что и

должен быть создан

Каждая строка файла необязательный указатель версии.

requirements.txt

содержит имя пакета, за которым следует

***СПРАВКА***

*Для установки пользовательских пакетов Pyodide использует micropip - легкую версию pip, которая работает в браузере.*

*Вы можете использовать API Micropip напрямую в вашем приложении Flet:*

import sys

if sys.platform == "emscripten": # проверка работы в среде Pyodide

import micropip

await micropip.install("regex")

### Предварительные версии пакетов Python

Вы можете разрешить загрузку предварительных версий пакетов PyPI, добавив

опцию в команду flet publish :

--pre

flet publish <your-flet-app.py> --pre

# Ресурсы (Assets)

Если вашему приложению требуются ресурсы (изображения, шрифты и т.д.), вы можете скопировать их в директорию веб-сайта, используя опцию

--assets

с командой flet publish :

<directory>

flet publish <your-flet-app.py> --assets assets

***ВНИМАНИЕ*** *Если в директории вашего приложения есть директория ресурсов*

*( assets ) и вы не указываете опцию --assets , то содержимое упаковано вместе с Python-приложением, а не скопировано в*

|  |  |  |
| --- | --- | --- |
|  | *assets* | |
| *dist* | | *.* |

# Стратегия URL

*будет*

Приложения Flet поддерживают два способа конфигурирования маршрутизации на основе URL:

* **Path** (по умолчанию) - пути считываются и записываются без хеша. Например, fletapp.dev/path/to/view .
* **Hash** - пути считываются и записываются в фрагмент хеша. Например, fletapp.dev/#/path/to/view .

Если провайдер хостинга поддерживает рендеринг одностраничного приложения (Single-page application или SPA), вы можете оставить стратегию URL "path" по умолчанию, так как это даст вам красивые URL.

Однако, если провайдер хостинга (например, GitHub Pages) не поддерживает режим SPA, то вам нужно опубликовать ваше приложение со стратегией URL "hash".

Чтобы указать стратегию "hash" URL при публикации статического приложения, используйте опцию --route-url-strategy :

flet publish <your-flet-app.py> --route-url-strategy hash

# Веб-рендерер

Вы можете изменить стандартный веб-рендерер "canvaskit" на "html" с помощью опции --web-renderer :

flet publish <your-flet-app.py> --web-renderer html

# Цветные эмодзи

Чтобы уменьшить размер приложения, стандартный рендерер "CanvasKit" не использует цветные эмодзи, поскольку файл шрифта с цветными эмодзи весит около 8 МБ.

Однако вы можете включить использование цветных эмодзи с помощью флага

--

use-color-emoji :

flet publish <your-flet-app.py> --use-color-emoji

В качестве альтернативы, переключитесь на рендерер *html*, который использует шрифты браузера.

# Хостинг веб-сайта в подкаталоге

Несколько приложений Flet могут быть размещены на одном домене - каждое приложение в своем собственном подкаталоге.

Чтобы опубликованное приложение Flet работало в подкаталоге, вы должны опубликовать его с опцией --base-url :

flet publish <your-flet-app.py> --base-url <sub-directory>

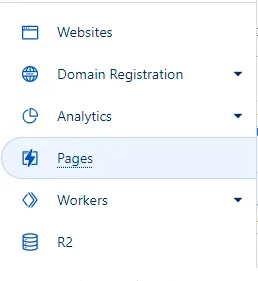
Например, если URL приложения [https://mywebsite.com/myapp,](https://mywebsite.com/myapp) то его нужно опубликовать с --base-url myapp .

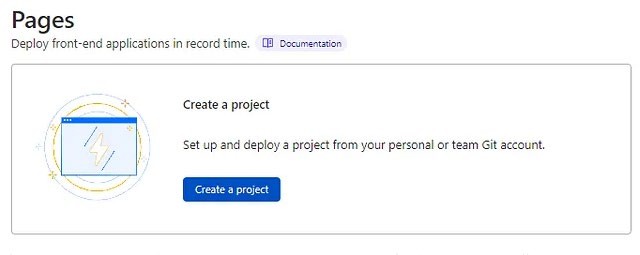
# Развертывание веб-сайта

Разверните статический веб-сайт на любом бесплатном хостинге, таком как GitHub Pages, Cloudflare Pages или Vercel!

### Cloudflare Pages

Прежде чем приступить к развертыванию, вам понадобится аккаунт. Вы можете получить его [здесь,](https://dash.cloudflare.com/sign-up/pages) или просто войдите в систему, если у вас уже есть аккаунт. После регистрации вам нужно будет подтвердить свой адрес электронной почты, кликнув по ссылке, которую вы получите на свою электронную почту. Если вдруг ссылка к вам на мейл не пришла оперативно, то проверьте также папку "спам".

В боковом меню вашего аккаунта выберите "Pages", как показано ниже:

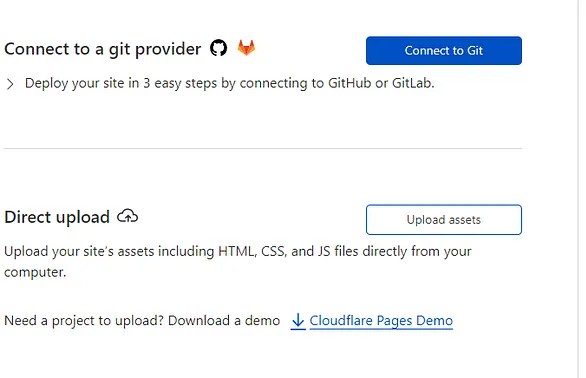
И оттуда выберите кнопку "Создать проект":

Cloudflare предлагает три способа создания проекта. Здесь будут рассмотрены только первые два:

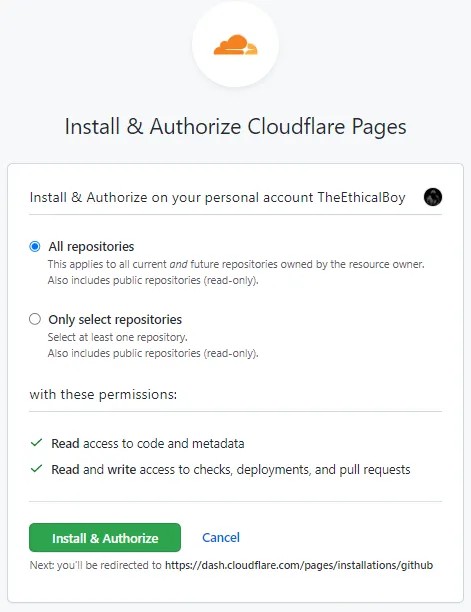
1. Подключение к провайдеру git
2. Прямая загрузка

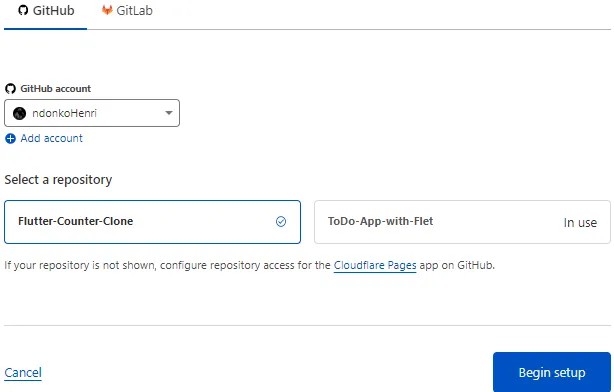
### Подключение к провайдеру Git

Для этого вам понадобится аккаунт GitHub или GitLab. В этом аккаунте должен быть репозиторий, который вы планируете использовать. Пример можно найти в этом репо.

Нажмите на синюю кнопку "Подключиться к Git":

Затем выберите вкладку со сервисом, содержащим ваш репозиторий. Затем подключите свой аккаунт. Выберите один из предложенных вариантов, затем нажмите на "Установить и авторизовать".

Выберите репозиторий, который будет использоваться, и нажмите кнопку "Начать настройку".

Прежде чем двигаться дальше, добавьте файл в свой репозиторий. В

runtime.txt

нем должна содержаться версия Python, которую будет использовать приложение. Введите 3.7, что является последней версией python, которую Cloudflare использует на момент написания этого урока.

[Вот](https://github.com/ndonkoHenri/Flutter-Counter-Clone/blob/master/runtime.txt) пример из вышеупомянутого репозитория.

Сделав это, мы теперь можем перейти к следующему шагу, который будет заключаться в настройке некоторых параметров сборки и развертывания для вашего сайта.

Установите имя вашего проекта и ветку репозитория (branch), которую нужно использовать. То есть по сути - это просто ветка вашего репозитория, в которой

любое нажатие на автоматически вызовет развертывание в вашей

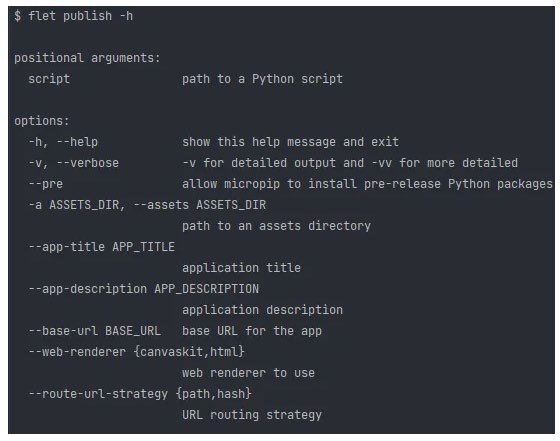
commit

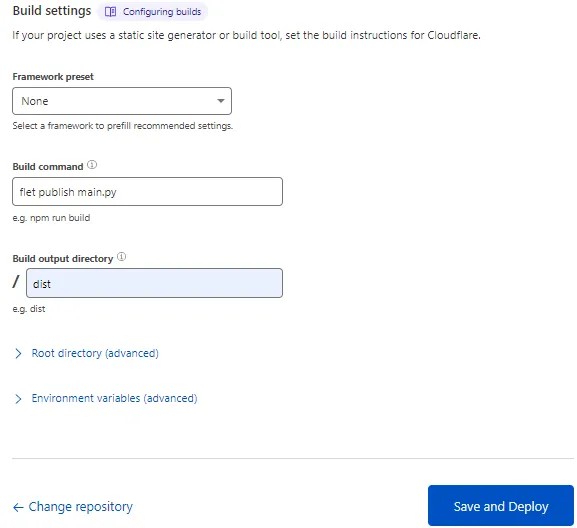
производственной среде. Нажатия на другие ветки вместо этого вызовут развертывание в вашей предварительной среде (preview environment).

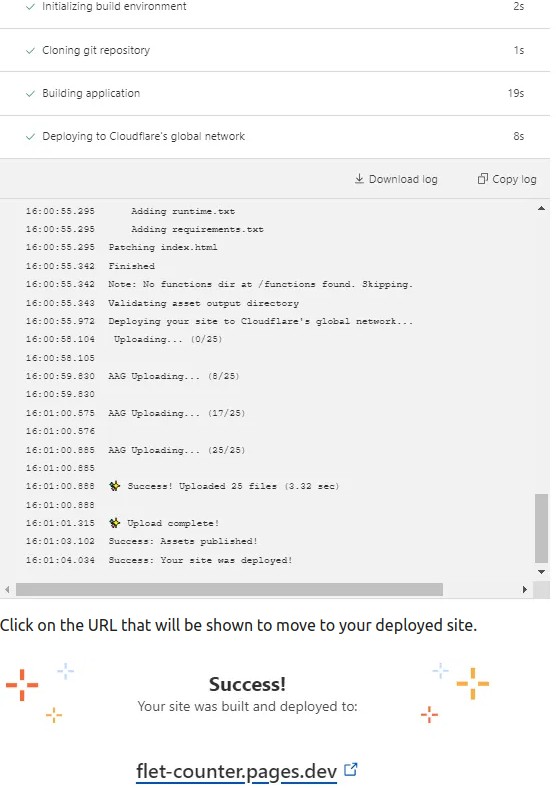
Далее, перейдите к разделу "Настройки сборки", где мы будем настраивать инструкции по сборке.

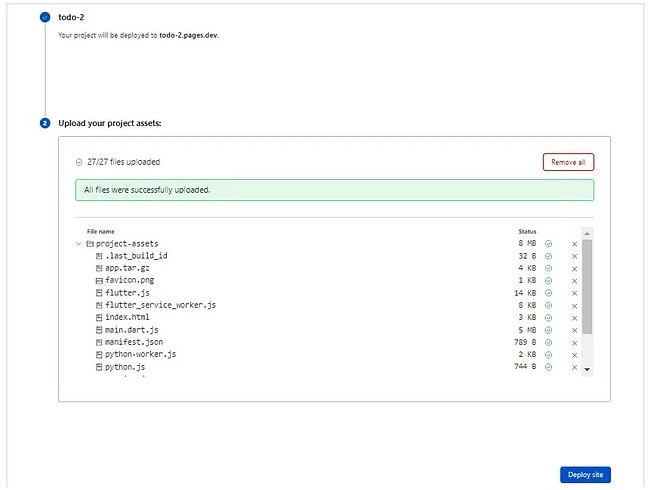
Пропустите "Фреймворк предустановки" (Framework preset) (используйте None), потому что Flet не находится в списке, и это не JavaScript-фреймворк :)

"Команда сборки" (Build command) зависит от структуры вашего приложения. Следуйте руководству в вышеуказанных разделах, чтобы придумать свою собственную команду сборки.

Когда команда сборки выполняется flet, создается папка с именем 'dist', которая будет содержать все веб-файлы, требуемые для Cloudflare pages. Установите ее как вашу директорию для вывода (output directory). Обратите внимание, что этот файл не будет добавлен в ваш репозиторий, потому что у Cloudflare есть только read-only (чтение) доступ к вашему коду.

Вы также можете указать дополнительные параметры: корневой каталог (каталог, в котором Cloudflare запускает команду сборки), и переменные окружения (переменные, которые будут использоваться во время сборки). Теперь нажмите на кнопку "Сохранить и развернуть" (Save and Deploy) и дайте Cloudflare выполнить остальную работу за вас.

Нажмите на URL, который будет показан, чтобы перейти к вашему развернутому сайту.

Если при открытии сайта вы видите ошибку Cloudflare, это означает, что они еще не закончили настройку. Так что просто подождите минуту, затем обновите страницу, и вы увидите работающее приложение. Вы можете протестировать вышеуказанный развернутый сайт здесь.

# Прямая загрузка

Нажмите на "Загрузить активы" (Upload assets). В первом шаге дайте вашему проекту имя. Помните, что имя, которое вы дадите, будет использоваться для определения ссылки, по которой будет развернут ваш проект.

Во втором шаге вам потребуется загрузить активы вашего проекта, либо в виде папки, либо в виде zip-файла (со всеми активами внутри). Если у вас уже все есть, то воспользуйтесь перетаскиванием (drag and drop) или выберите их с помощью средства выбора папок (folder picker).

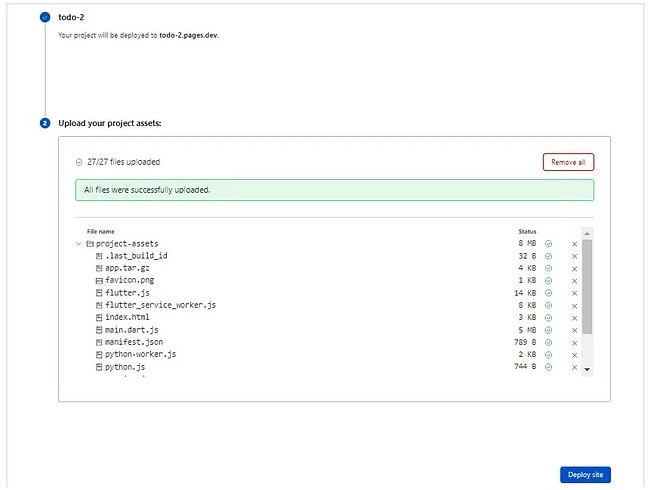
Если у вас еще нет этих активов, но уже есть приложение, которое вы построили,

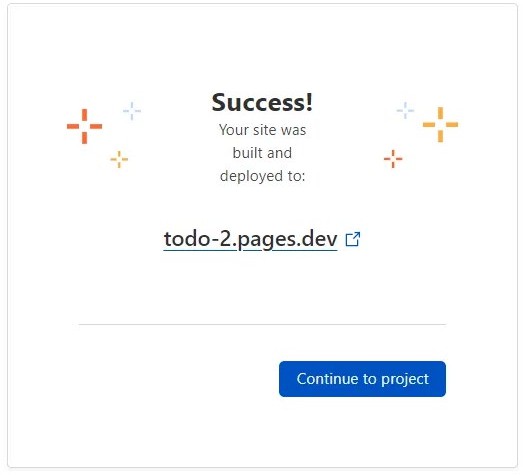
используйте команду в каталоге, содержащем файлы вашего

flet publish

приложения, в таком случае будет создана папка dist , которую вы затем загрузите на страницы Cloudflare, как уже упоминалось выше.

После загрузки нажмите на кнопку "Развернуть сайт" (Deploy site).

Затем вы увидите сообщение об успешном выполнении с ссылкой на ваш развернутый веб-сайт. Протестируйте пример развернутого сайта [здесь.](https://todo-2.pages.dev/)

Если при открытии сайта вы видите ошибку Cloudflare, это означает, что они еще не закончили настройку. Так что просто подождите минуту, затем обновите страницу, и вы увидите работающее приложение. Теперь вы можете нажать на кнопку "Продолжить к проекту" (Continue to project), чтобы отслеживать свои деплои или создавать новые, следуя тем же шагам, что и выше.

# Устранение неполадок

Когда приложение Flet работает в веб-браузере, все его

выражения отображаются на вкладке "Консоль" в Developer Tools

print()

браузера. print() может быть использован как простой инструмент для отладки.

Вы также можете использовать модуль логирования (logging) и выводить сообщения в консоль с различной степенью серьезности.

Чтобы включить подробное логирование Flet, просто добавьте в свою программу этот скрипт:

import logging

logging.basicConfig(level=logging.DEBUG)

Этот код позволит вам отслеживать и выводить подробные отчеты о работе вашего приложения Flet, которые будут помогать вам при отладке и выявлении проблем.

Теперь все сообщения логирования уровня DEBUG и выше будут выводиться в консоль, включая подробные сообщения о том, какие действия выполняет ваше приложение.

Важно отметить, что уровень логирования может быть изменен на любой другой в соответствии с вашими потребностями. В Python используется следующая иерархия уровней логирования, от наименее до наиболее серьезного:

* DEBUG
* INFO
* WARNING
* ERROR
* CRITICAL

Таким образом, если вы установите уровень логирования на ERROR , то будут

выводиться только сообщения уровня и CRITICAL , в то время как

ERROR

сообщения уровня DEBUG , и WARNING будут игнорироваться.

INFO

С помощью этой функции логирования вы сможете более эффективно отслеживать и устранять возможные проблемы в своем приложении Flet, что повысит его стабильность и надежность.