4.5 Выбор хостинг-провайдера: Fly.io

Выбор хостинг-провайдера: Fly.io

Приложения Flet используют WebSockets для частичного обновления пользовательского интерфейса в реальном времени и отправки событий обратно в вашу программу. При выборе хостинг-провайдера для своего Flet-приложения следует обратить внимание на поддержку им WebSockets. Иногда WebSockets запрещены или входят в состав более дорогих предложений, иногда имеется прокси, который периодически разрывает WebSocket-соединение по таймауту (Flet реализует логику повторного соединения, но в любом случае это может быть неприятным поведением для пользователей вашего приложения).

Еще одним важным фактором при выборе хостинг-провайдера для приложения Flet является задержка. Каждое действие пользователя над пользовательским интерфейсом посылает сообщение приложению Flet, а приложение отправляет обновленный пользовательский интерфейс обратно. Убедитесь, что у хостинг- провайдера есть несколько центров обработки данных, чтобы вы могли разместить свое приложение ближе к большинству пользователей.

**Fly.io**

[Fly.io](https://fly.io/) обладает надежной поддержкой WebSocket и может развернуть ваше приложение в дата-центре, который находится ближе к вашим пользователям. У них очень привлекательные цены с щедрым бесплатным уровнем, который позволяет вам размещать до 3 приложений бесплатно.

Чтобы начать работу с *Fly*, установите flyctl , а затем пройдите аутентификацию:

flyctl auth login

Чтобы развернуть приложение с помощью flyctl , вы должны добавить следующие 3 файла в папку с вашим приложением Python.

Создайте со списком зависимостей приложения. Как минимум, он должен содержать модуль :

flet

requirements.txt

requirements.txt

flet

Создайте fly.toml , описывающий приложение *Fly*:

fly.toml

app = "<имя-вашего-приложения>"

kill\_signal = "SIGINT"

kill\_timeout = 5

processes = []

[env]

FLET\_SERVER\_PORT = "8080"

[experimental]

allowed\_public\_ports = []

auto\_rollback = true

[[services]]

http\_checks = []

internal\_port = 8080

processes = ["app"]

protocol = "tcp"

script\_checks = []

[services.concurrency]

hard\_limit = 25

soft\_limit = 20

type = "connections"

[[services.ports]]

force\_https = true

handlers = ["http"]

port = 80

[[services.ports]]

handlers = ["tls", "http"]

port = 443

[[services.tcp\_checks]]

grace\_period = "1s"

interval = "15s"

restart\_limit = 0

timeout = "2s"

Замените ` <имя-вашего-приложения> ` на желаемое имя приложения, которое будет также использоваться в URL приложения, таком как

. Обратите внимание, что мы устанавливаем значение

https://<имя-вашего-

FLET\_SERVER\_PORT

приложения>.fly.dev

переменной окружения в 8080 , что является внутренним TCP-

портом, на котором будет работать веб-приложение Flet.

Создайте *Dockerfile*, содержащий команды для сборки контейнера вашего приложения:

FROM python:3-alpine

WORKDIR /app

COPY requirements.txt ./

RUN pip install --no-cache-dir -r requirements.txt

COPY . .

EXPOSE 8080

CMD ["python", "./main.py"]

- это файл с вашей программой на Python.

main.py

**ПРИМЕЧАНИЕ**

*Fly.io развертывает каждое приложение в виде контейнера Docker, но отличительной особенностью Fly является то, что она предоставляет бесплатный удаленный конструктор Docker, поэтому вам не нужно устанавливать Docker на своей машине.*

Далее переключите командную строку на папку с вашим приложением и выполните следующую команду, чтобы создать и инициализировать новое приложение Fly:

flyctl apps create --name <имя-вашего-приложения>

Разверните приложение, выполнив:

flyctl deploy

Вот и все! Откройте ваше приложение в браузере, выполнив:

flyctl apps open