# Create aio app Documentation

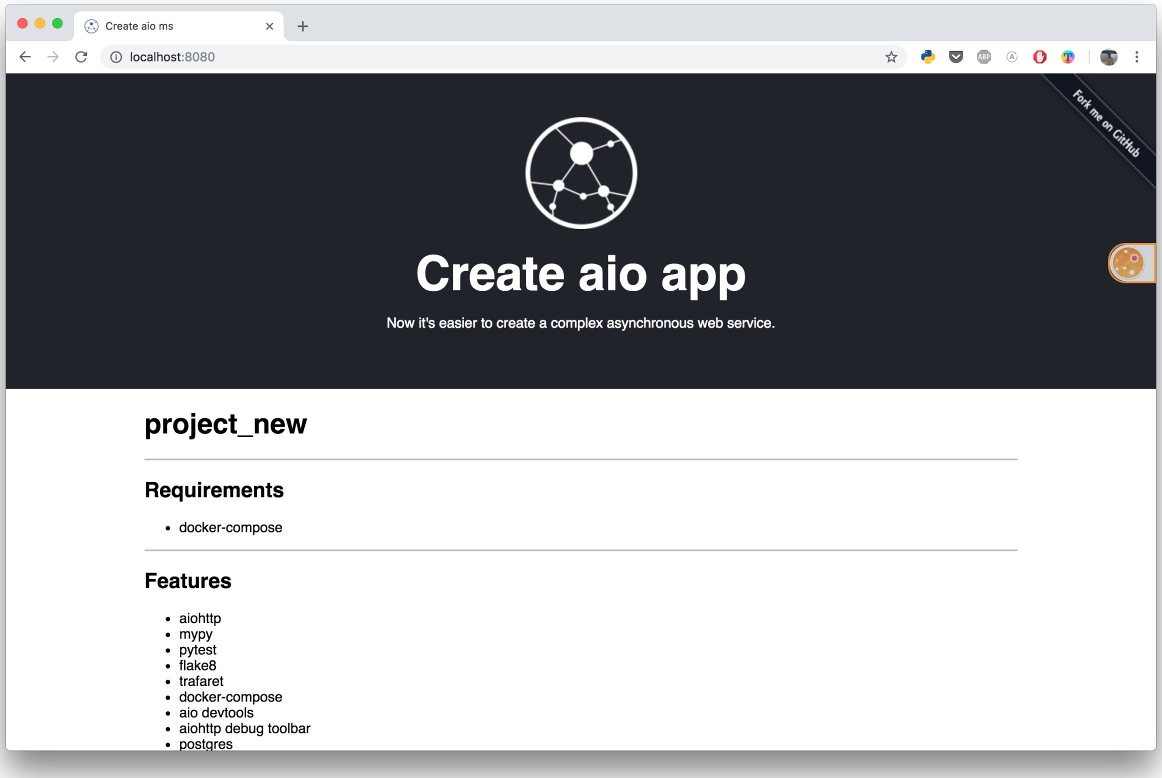
**Mykhailo Havelia**

**Dec 26, 2021**

# Aiohttp Quick start:

What is a *create aio app*? This project is designed to quickly and simply creating a web application based on *aiohttp* with use best practices.

A *create aio app* provide testing, documentation, deploying and a lot of helpful boilerplate code for quickly start with *aiohttp*.



Create aio app

Now it's easier to create a complex asynchronous web service.

project_new

Requirements

- docker-compose

Features

- aiohttp
- mypy
- pytest
- flake8
- trafaret
- docker-compose
- aio devtools
- aiohttp debug toolbar
- postgres

# CHAPTER 1

# Requirements

For start with *create aio app* you need to have:

- docker
- docker-compose

CHAPTER 2

# Features

## 2.1 Quick start for aiohttp

To use *create-aio-app* you have to meet the next requirements:

- python>=3.6
- docker-compose

### 2.1.1 Install

```
pip install create-aio-app
```

### 2.1.2 Usage

```
create-aio-app my_project
```

If you want to use interactive mode enter the next command:

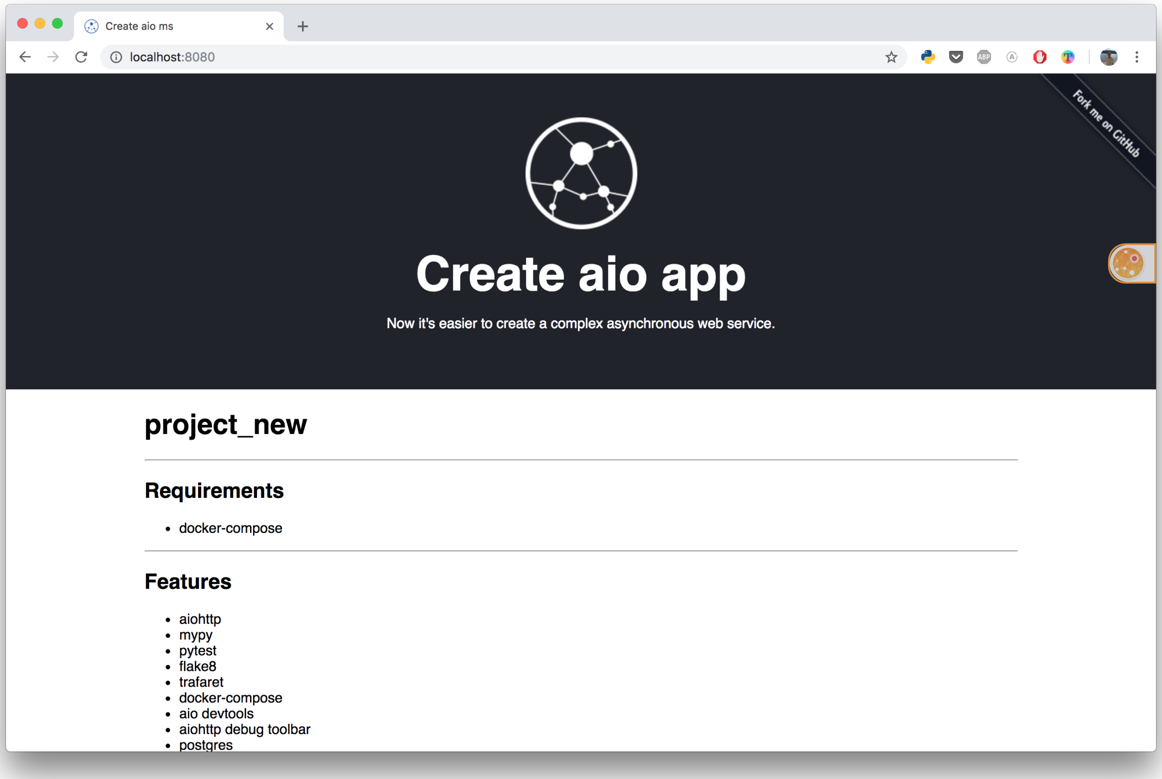
```
create-aio-app
```

This will create a new directory called *my_project*.

```
cd my_project

make run # start your project
```

Navigate in your browser to *http://localhost:8080/*



Create aio app

Now it's easier to create a complex asynchronous web service.

project_new

Requirements

- docker-compose

Features

- aiohttp
- mypy
- pytest
- flake8
- trafaret
- docker-compose
- aio devtools
- aiohttp debug toolbar
- postgres

## 2.2 Make commands

The set of commands available in the `Makefile`.

### 2.2.1 Common

| command | description |
|---|---|
| `make run (make)` | Start the development server |
| `make stop` | Stop docker containers |
| `make clean` | Clean up of docker containers |
| `make bash` | Interactive shell inside the running container (the command can be executed only if the server is running e.g. after `make run`) |
| `make upgrade` | Upgrade dependencies |

### 2.2.2 Testing

| command | description |
|---|---|
| `make test` | Run the `pytest` suite inside docker |
| `make mypy` | Run `mypy` for type checking |
| `make black` | Run `black` code formatter |
| `make lint` | Run flake8 (All the settings for *flake8* can be customized in *.flake8* file) |
| `make profile` | Run `py-spy` sampling profiler. It defaults to 60 seconds. Can be change by adding the `TIME` variable. eg `make profile TIME=30` |

### 2.2.3 Database

Next commands are available if you have not disabled `postgres` option when creating a project:

| command | description |
|---|---|
| `make migrations` | Generate a new migration |
| `make migrate` | Apply migrations |
| `make psql` | Connect to the postgres inside running container |

### 2.2.4 Other

| command | description |
|---|---|
| `make doc` | Generate a sphinx documentation |

## 2.3 Documentation

The best thing you can do for future self is to write both tests and documentation about your project. This section explains how to structure and organize documentation in your project.

### 2.3.1 Structure

We use the next structure:

- **pages/** - the directory for all **.rst* files about the boilerplate.
- **project/** - this directory is for **.rst* files associated with your project (this can include auto-generated docs, business logic, etc.)

All new files should be added to *index.rst*.

### 2.3.2 Linter

Doc8 is an opinionated style checker for *rst* styles of documentation. This linter used by default.

### 2.3.3 Read more:

## 2.4 Testing

Code without tests is broken by design. Please remember this and write tests;)

This project runs tests inside Docker. It allows you to run tests locally with maximum isolation from the environment. It also gives you a simple way of adding new resources required for your tests.

### 2.4.1 Pytest

Use this command to run the tests:

```
make test
```

This will run flake8 and after successful execution, the command will run a test suite with *pytest*

If you want to run a single test, you can pass an argument to *docker-compose* like this:

```
docker-compose run test project_name/main/tests/test_views.py::test_view
```

### 2.4.2 mypy

Mypy is an optional static type checker for Python. We suggest you try it out as it allows you to catch some errors, write safer code and make refactoring of the code easier in the future.

To run mypy use this command:

```
make mypy
```

Settings for mypy resides inside the mypy.ini file.

### 2.4.3 Read more:

# CHAPTER 3

# Indices and tables

- genindex
- search