github actions: introduction

Introducing GitHub Actions

Next Video

Notes

I have a very small Python project running. It has an `app.py` file that contains a FastAPI service.

```
import time
import asyncio

from pydantic import BaseModel, validator
from fastapi import FastAPI

app = FastAPI()


class Item(BaseModel):
    name: str
    price: float

    @validator("price")
    def price_must_be_positive(cls, value):
        if value <= 0:
            raise ValueError(f"we expect price >= 0, we received {value}")
        return value


@app.get("/")
def root():
    return {"message": "hello world again"}


@app.get("/users/{user_id}")
def read_user(user_id: str):
    return {"user_id": user_id}


@app.post("/items/")
def create_item(item: Item):
    return item


@app.get("/sleep_slow")
def sleep_slow():
    _ = time.sleep(1)
    return {"status": "done"}
```

```
@app.get("/sleep_fast")
async def sleep_fast():
    _ = await asyncio.sleep(1)
    return {"status": "done"}
```

It it also has a `**test_app.py**` file that contains some unit tests for it.

```
from starlette.testclient import TestClient

from app import app

client = TestClient(app)


def test_root_endpoint():
    resp = client.get("/")
    assert resp.status_code == 200
    assert resp.json() == {"message": "hello world again"}


def test_users_endpoint():
    resp = client.get("/users/1")
    assert resp.status_code == 200
    assert resp.json() == {"user_id": "1"}


def test_correct_item():
    json_blob = {"name": "shampoo", "price": 1.5}
    resp = client.post("/items/", json=json_blob)
    assert resp.status_code == 200


def test_wrong_item():
    json_blob = {"name": "shampoo", "price": -1.5}
    resp = client.post("/items/", json=json_blob)
    assert resp.status_code != 200
```

The project also comes with some files that contain all the requirements. There's a `**requirements.txt**` file with the following contents.

```
fastapi==0.78.0
uvicorn==0.18.2
requests==2.28.1
```

And there's another one for the developer tools.

```
flake8==4.0.1
pytest==7.1.2
```

You can install everything locally by running:

```
python -m pip install --upgrade
python -m pip install -r requirements.txt
python -m pip install -r dev-requirements.txt
```

Adding GitHub Actions

You can confirm locally that the tests pass by running:

```
python -m pytest
```

This is nice, but we'd like these unit tests not just to work locally on the current code base. We'd also like to run them on every pull request such that we don't accidentally introduce code that breaks the unit tests.

To set that up, we are going to use GitHub Actions. You can configure this by creating a file in your repository over at `**.github/workflows/unit-tests.yml**`. This is where we can define steps that GitHub needs to run on our behalf.

Here's an example that we use in the video:

```
name: Python Unit Tests

on:
  pull_request:
    branches:
    - main

jobs:
  build:
    runs-on: ubuntu-latest

    steps:
    - uses: actions/checkout@v2
    - name: Set up Python
      uses: actions/setup-python@v4
      with:
        python-version: '3.7'
    - name: Install dependencies
      run: |
        python -m pip install --upgrade pip
        python -m pip install -r requirements.txt
        python -m pip install -r dev-requirements.txt
    - name: Test with pytest
      run: |
        pytest --verbose
```

Story

Content

Labs

TILs

Tracks

Datasets

Old Content

Testimonials

Terms

Statistics

Status

Contact

Twitter

Newsletter

Made by people who want to remedy the skill anxiety. Tools and thoughts that might make your professional life more enjoyable.