**Configure Docker:**

Generate a token from:

<https://hub.docker.com/settings/security>

paste it in your GitHub projects Secrets actions with secret name: DOKERHUB\_TOKEN:

<https://github.com/yugants/recipe-app-api/settings/secrets/actions>

create another secret with: DOCKERHUB\_USER

and save docker username in it.

Create a requirements.txt and type versions of Django and djangorestframework

.dockerignore – ignored by docker

.gitignore – ignored by git

**DockerFile:**

The Dockerfile is used to build our image, which contains a mini Linux Operating System with all the dependencies needed to run our project.

In Dockerfile we performed:

* Copy operation from local to docker
* Created a virtual env. on docker
* Created a user for docker named “django-user”
* Set the python env. path

Create an app folder in main DIR and run:

docker build .

**Docker-compose:**

We are initializing docker compose for our project, after configuring run:

docker-compose build

**Linting:**

That’s right! Linting is used to ensure code is formatted correctly. It highlights issues like invalid tab spacing and line lengths.

We will use “flake8” for linting

Now make changes in docker-compose, Dockerfile and create .flake8

To check if flake8 is installed correctly or not type:

docker-compose run --rm app sh -c "flake8"

**Create Project:**

docker-compose run --rm app sh -c "django-admin startproject app ."

**Run Server in Docker:**

docker-compose up

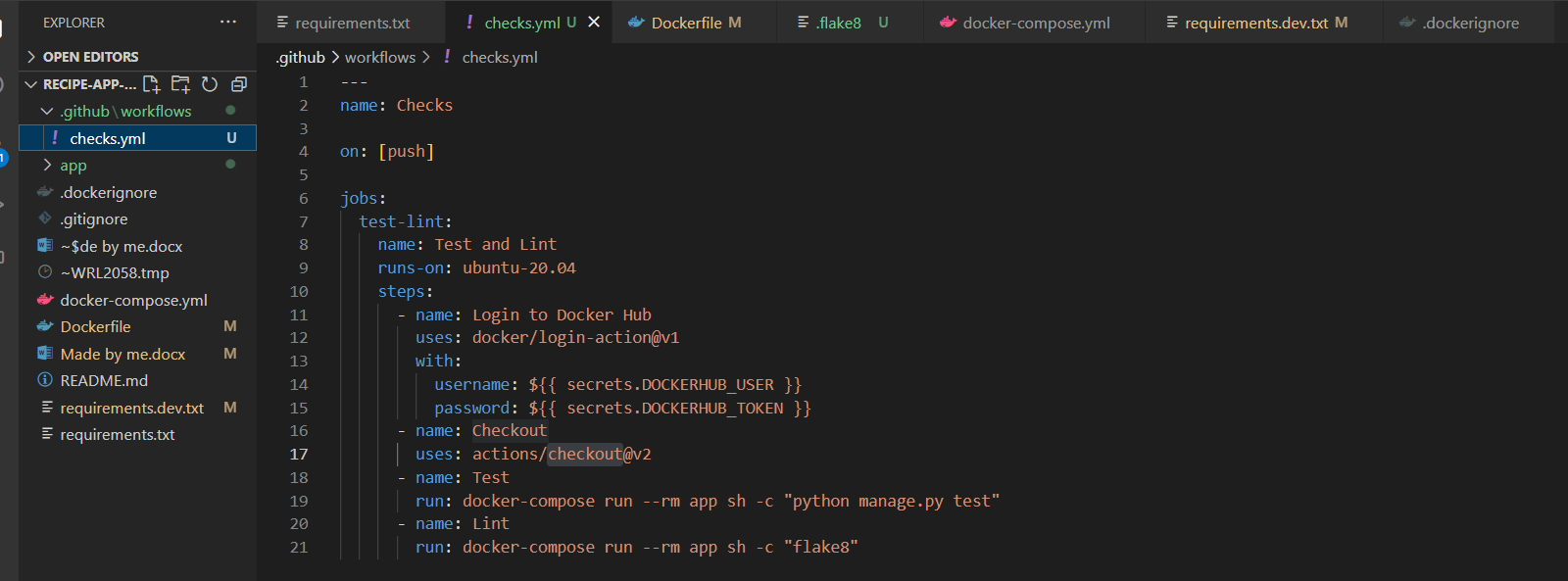
**GitHub Actions:**

* Automation tool
* Similar to Jenkins, GitLab CI/CD
* Run jobs when code changes
* Automate tasks

Our trigger for CI/CD is push operation on github.

**For Configuring CI/CD:**

Make a folder in main file: .github/workflows -> in it create a file -> checks.yml:



Then run the in shell to check:

docker-compose run --rm app sh -c "python manage.py test"