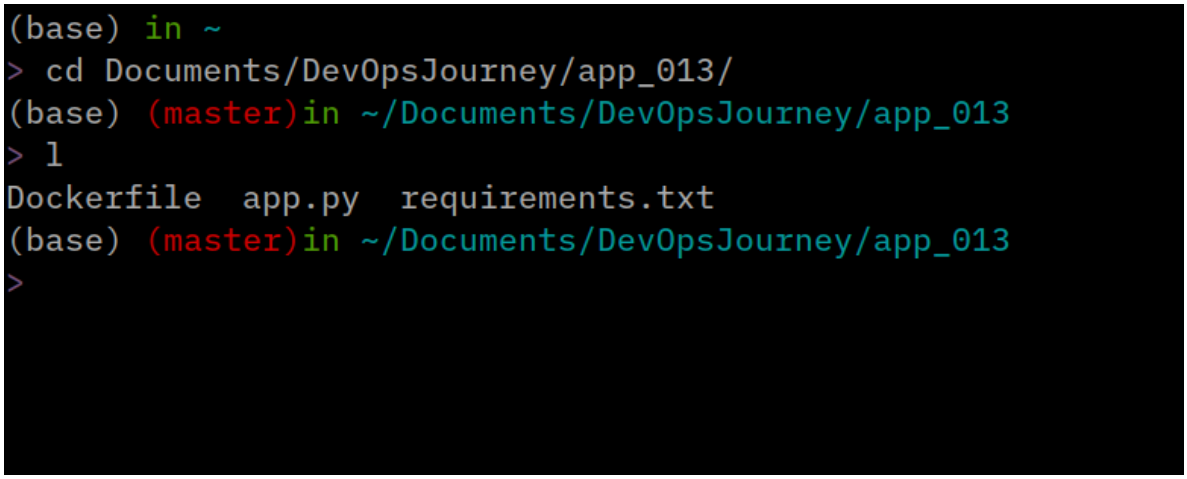
#021 docker-compose

Introduction

this is part 21 from the journey it's a long journey(360 day) so go please check previous parts , and if you need to walk in the journey with me please make sure to follow because I may post more than once in 1 Day but surely I will post daily at least one 😍.

And I will cover lot of tools as we move on.

Download app_021



if you follow part 9

```
cd location/DevOpsJourney/
git pull
cd app_021/
```

replace location with where you put the DevOpsJourney

if your new go to part 9 and do same steps will download old lecture files and new one.

Django

Django is a python web-framework .
In this part I will use an real world dockerized Django that ready for production.
Our project will connect to postgres also as database.

Without docker-compose

to build our images and run our containers before we run them manually and it's hard to remember what we need to pass as envirement variables such as port , also if we give our Dockerfile to someelse probably he doesn't know what to pass.
So here came the role of our docker-compose to make life more and more easy.

---

docker-compose

docker-compose is using an YAML (Yet Another Markup Language) to do the boring work instead of us.
So here how it works , it simply take as default input the yaml or yml file in the project the default is a file named docker-compose.yml , in our project we don't have it we have our own files called local.yml and production.yml , one for local development and the other for production.

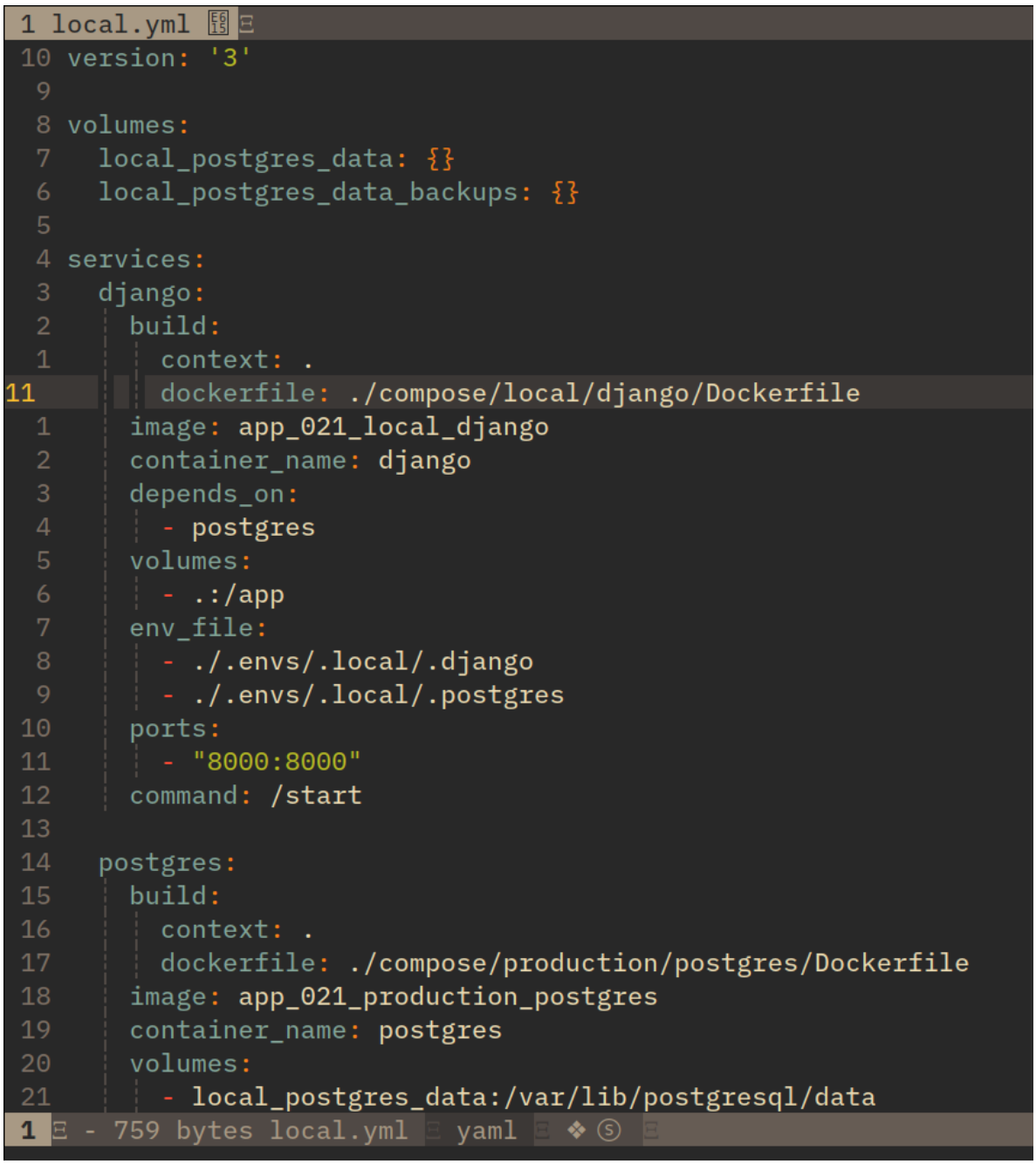
Explain project

usually when we don't use docker , Django application run with

```
python manage.py runserver 0.0.0.0:8000
```

8000 is the port here , so we need to open that port in our docker.

I am going to take look inside the local.yml for now



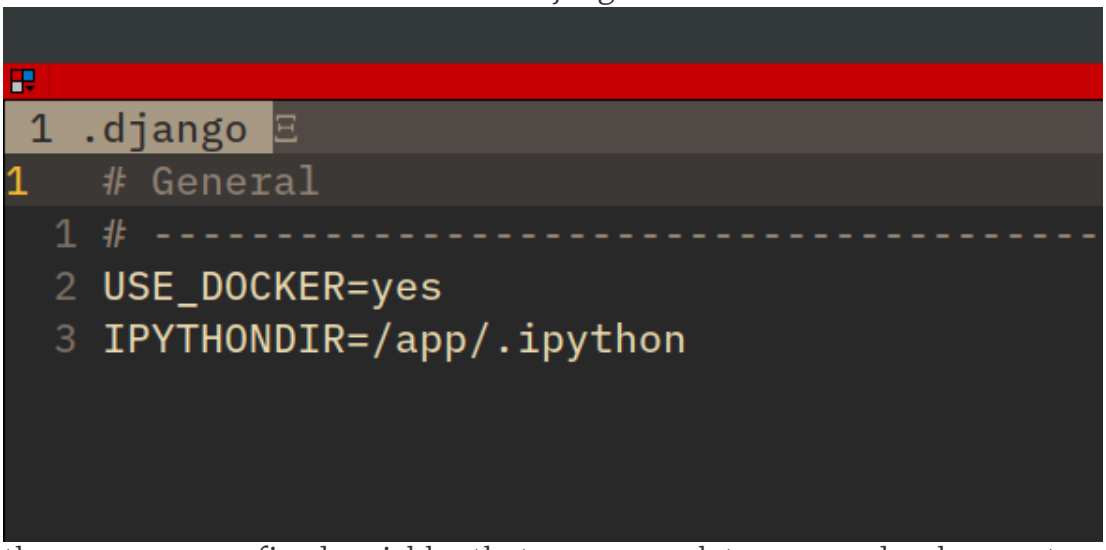
we can see a lot of stuff here but if you follow me from start of journey you will be comfortable with what you see.

let's talk about each line here:

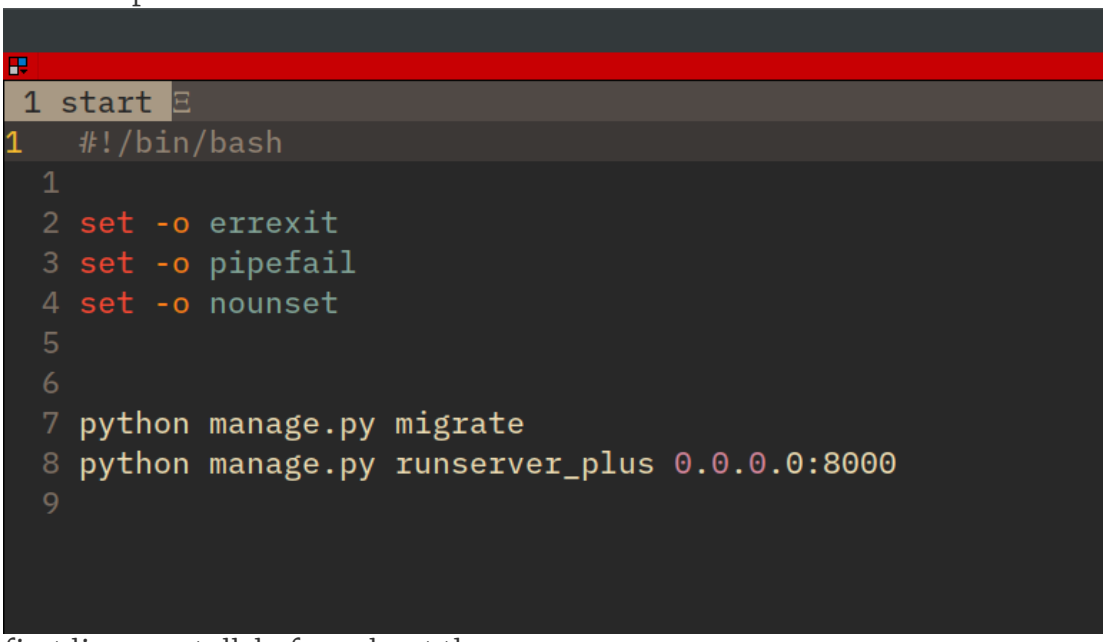
1. version: '3' mean we are using docker-compose api version 3 (I will talk more about in upcoming part)
2. volumes here we setup 2 postgres volumes one for data and one for backup (we talk about volumes also before)
3. services here the important part

- django is the name of our service , build we need to specify the dockerfile path in my case it's located at ./compose/local/django/Dockerfile
- image is the name of our image
- container_name name of our container
- depend_on postgres it mean she need the postgres service
- volumes of our Django

- environment files let's take a look at .django



they are some prefixed variables that we can use later on our development
- ports are the port we need to open in our case is 8000
- command: /start
it's a script we need to run let's take a look at it



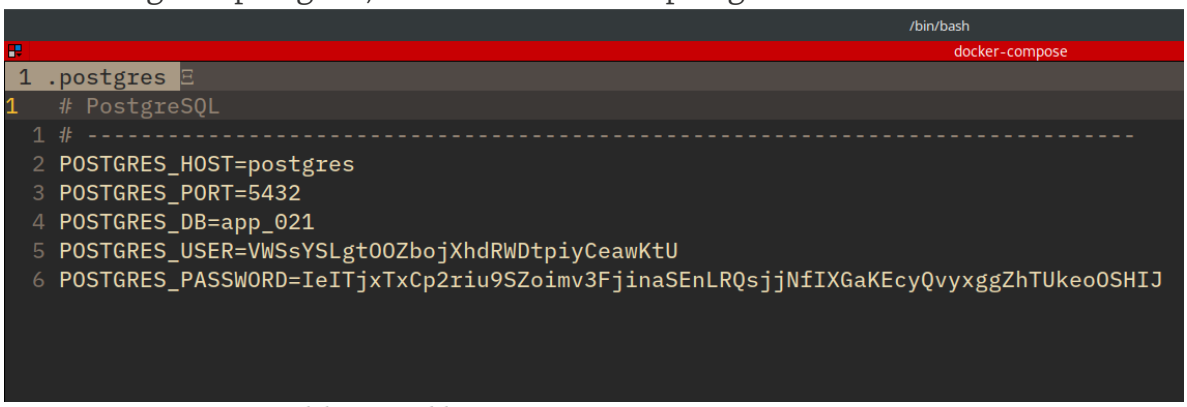
first lines we talk before about them
- python manage.py migrate it's Django thing to migrate our changes to database
- python manage.py runserver_plus 0.0.0.0:8000 it's Django way of running our server

```
      - ./.envs/.local/.django
      - ./.envs/.local/.postgres
    ports:
      - "8000:8000"
    command: /start

  postgres:
    build:
      context: .
      dockerfile: ./compose/production/postgres/Dockerfile
    image: app_021_production_postgres
    container_name: postgres
    volumes:
      - local_postgres_data:/var/lib/postgresql/data
      - local_postgres_data_backups:/backups
    env_file:
      - ./.envs/.local/.postgres
```

1 - 759 bytes local.yml yaml

same things for postgres , this is a look inside .postgres file



It conatin some variables used by postgres.

Running our compose file

Since we use an modified file (local.yml) , in able to run it we should use -f which stand for file we need right after docker-compose

```
docker-compose -f local.yml up -d
```

up here is to create and start containers
-d here is for running in background

```
Removing intermediate container 74460277ac28
 ---> 5440b4e9d1df
Step 12/14 : RUN chmod +x /start
 ---> Running in a617d448e605
Removing intermediate container a617d448e605
 ---> 9edb9bbe5d44
Step 13/14 : WORKDIR /app
 ---> Running in 67518f66dcb3
Removing intermediate container 67518f66dcb3
 ---> 62a18b0d89e2
Step 14/14 : ENTRYPOINT ["/entrypoint"]
 ---> Running in 744e23201d68
Removing intermediate container 744e23201d68
 ---> 0f38574a7851
Successfully built 0f38574a7851
Successfully tagged app_021_local_django:latest
WARNING: Image for service django was built because it did not already exist. To rebuild this image you must use `docker-compose build` or `docke
-compose up --build`.
Creating postgres ... done
Creating django   ... done
(base) (master)in ~/Documents/DevOpsJourney/app_021
>
```

after waiting for some time it will download lot of thing because it's real world app.

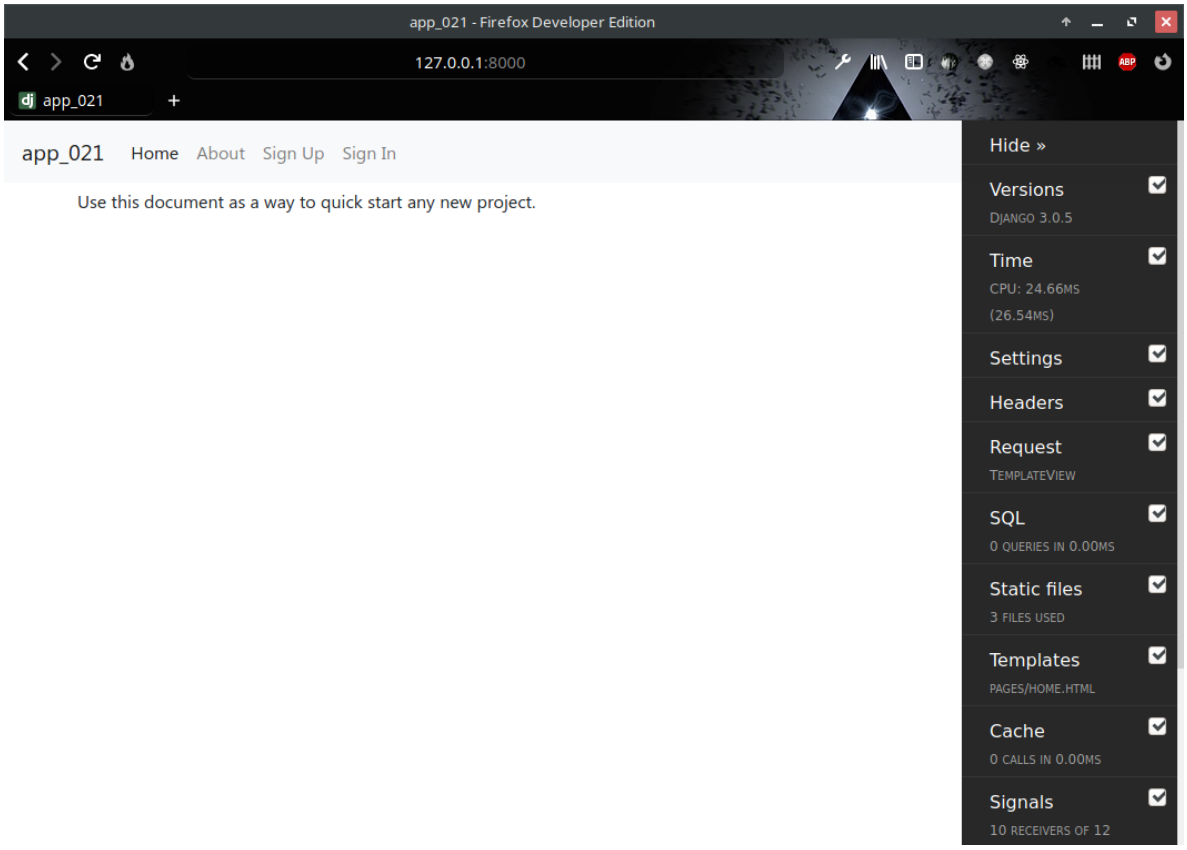
```
(base) (master)in ~/Documents/DevOpsJourney/app_021
> docker ps
CONTAINER ID        IMAGE                      COMMAND                  CREATED             STATUS              PORTS                    NAMES
d44f8b556385        app_021_local_django       "/entrypoint /start"     27 seconds ago      Up 24 seconds       0.0.0.0:8000->8000/tcp   django
a1b07ee62af8        app_021_production_postgres   "docker-entrypoint.s…"   28 seconds ago      Up 26 seconds       5432/tcp                 postgr
es
```

```
docker ps -a
```

we can see we have our containers running in the background.

in our browser go to

```
127.0.0.1:8000
```



we have a website running just with one command , amazing!

Challenge

go and take a look at Dockerfile and try to understand it ,
also take a look at production.yml file