#024 Docker with microservices

Introduction

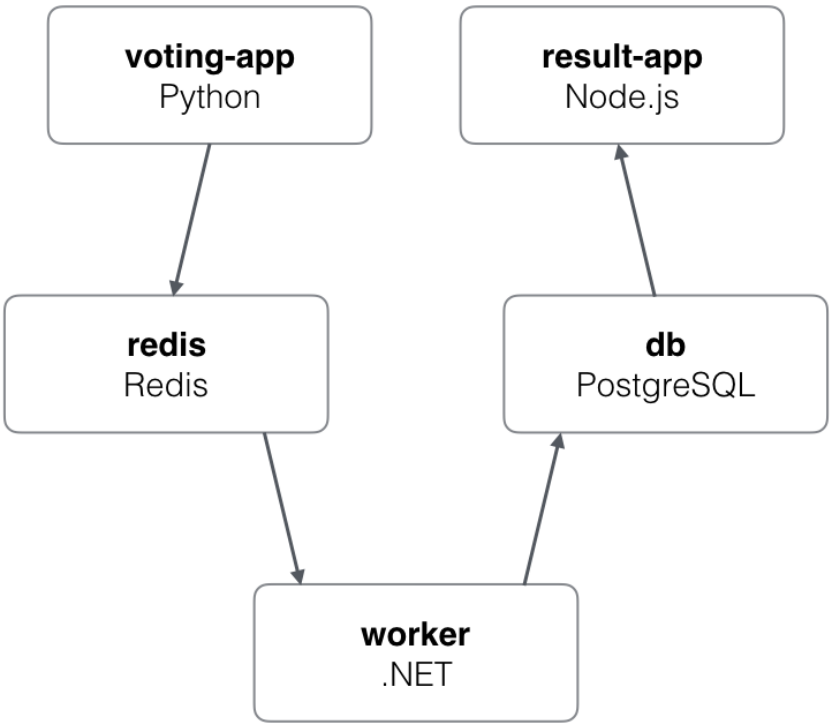
this is part 24 from the journey it's a long journey(360 day) so go please check previous parts , and if you need to walk in the journey with me please make sure to follow because I may post more than once in 1 Day but surely I will post daily at least one 😍.

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

What is microservices

basically microservices is dividing our big application to small apps , each app can have his own front-end and backend but they can communicate with each other.

Architecture



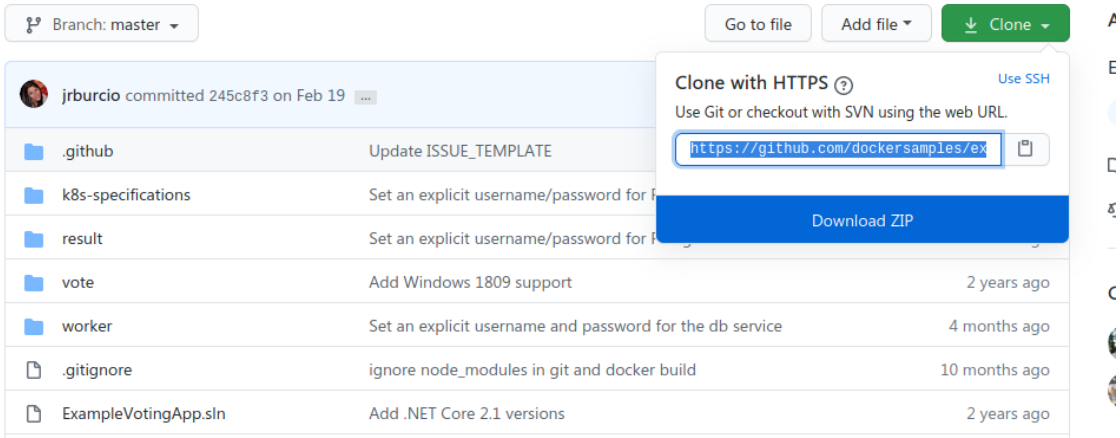
This image explain how this project work .

.A front-end web app in Python or ASP.NET Core which lets you vote between two options
.A Redis or NATS queue which collects new votes
.A .NET Core, Java or .NET Core 2.1 worker which consumes votes and stores them in...
.A Postgres or TiDB database backed by a Docker volume
.A Node.js or ASP.NET Core SignalR webapp which shows the results of the voting in real time

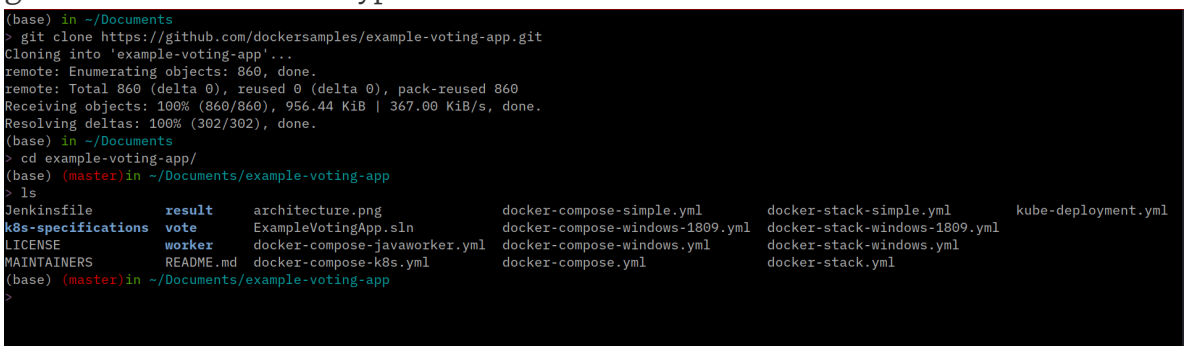
---

Example

we are going to use docker official example go to this link and copy link to clone it locally



go the the terminal and type



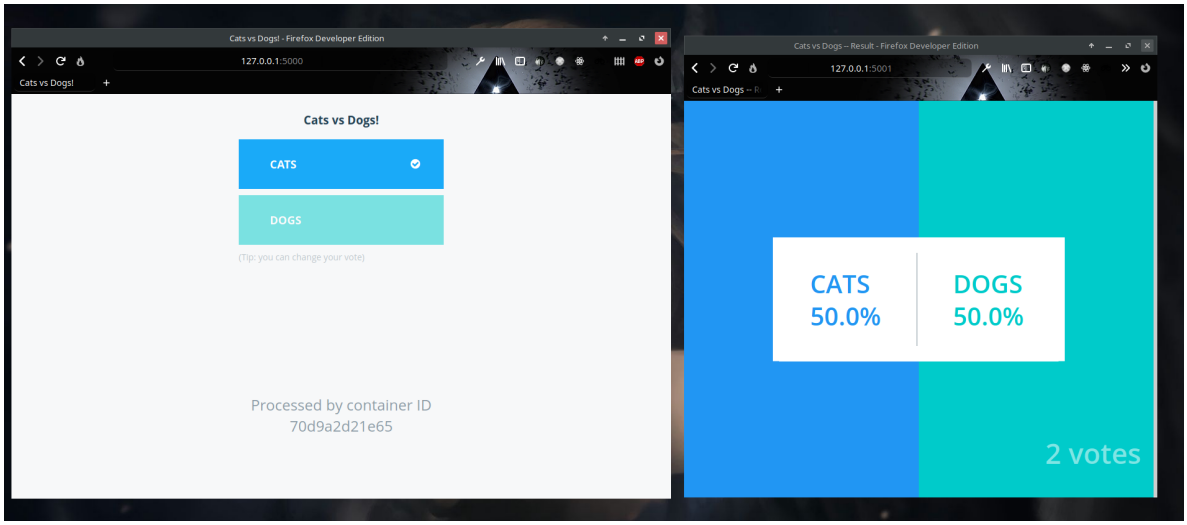

```
git clone https://github.com/dockersamples/example-voting-app.git
cd example-voting-app
```

now the project is ready to build , we can see many docker-compose files run what ever is compatible with your machine.
I am going to use docker-compose-javaworker.yml
javaworker will use the java version not dotnet

```
(base) (master)in ~/Documents/example-voting-app
> docker-compose -f docker-compose-javaworker.yml up
Building worker
Step 1/10 : FROM maven:3.5-jdk-8-alpine AS build
3.5-jdk-8-alpine: Pulling from library/maven
4fe2ade4980c: Pull complete
6fc58a8d4ae4: Pull complete
ef87ded15917: Downloading  13.46MB/70.61MB
7f88801c955d: Download complete
610acd6c8983: Downloading   8.58MB/8.989MB
e2814e90e330: Waiting
6b24ea8a8cda: Waiting
```

```
docker-compose -f docker-compose-javaworker.yml up
```

after finish building it's time to take a look at it
head in your browser to 127.0.0.1:5000 and in another window to 127.0.0.1:5001



have fun with it try to vote and see change live !

---

docker-compose

```yaml
version: "3"

services:
  vote:
    build: ./vote
    command: python app.py
    volumes:
     - ./vote:/app
    ports:
      - "5000:80"
    networks:
      - front-tier
      - back-tier

  result:
    build: ./result
    command: nodemon server.js
    volumes:
      - ./result:/app
    ports:
      - "5001:80"
      - "5858:5858"
    networks:
      - front-tier
      - back-tier

  worker:
    build:
      context: ./worker
      dockerfile: Dockerfile.j
    networks:
      - back-tier

  redis:
    image: redis:alpine
    container_name: redis
    ports: ["6379"]
    networks:
      - back-tier
```

if you see they are using different services each use it's own front-end and back-end , and they are connected together using docker network you can see that vote and result connected to both networks back and front , but redis and worker and postgres are only to back-end.

that's a real example of power of docker how it's connected together to make a complete microservice app.

Also There is a deployment stack , it will be later later in the end when we cover all the tools , I will try to make an app contain all what we learn together :)