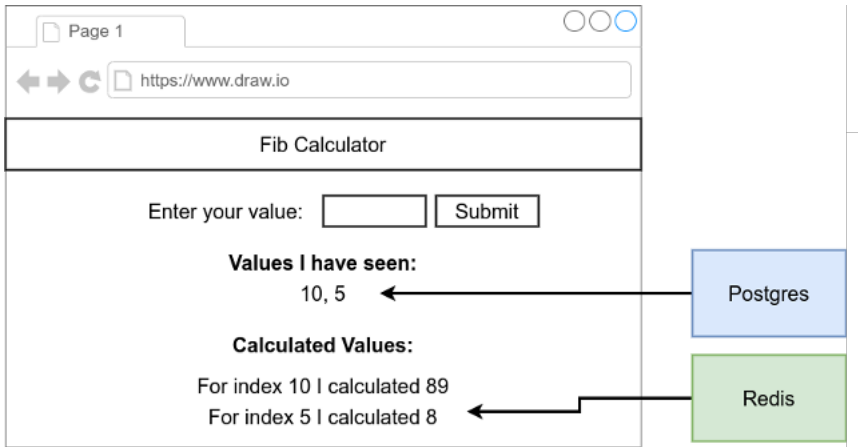
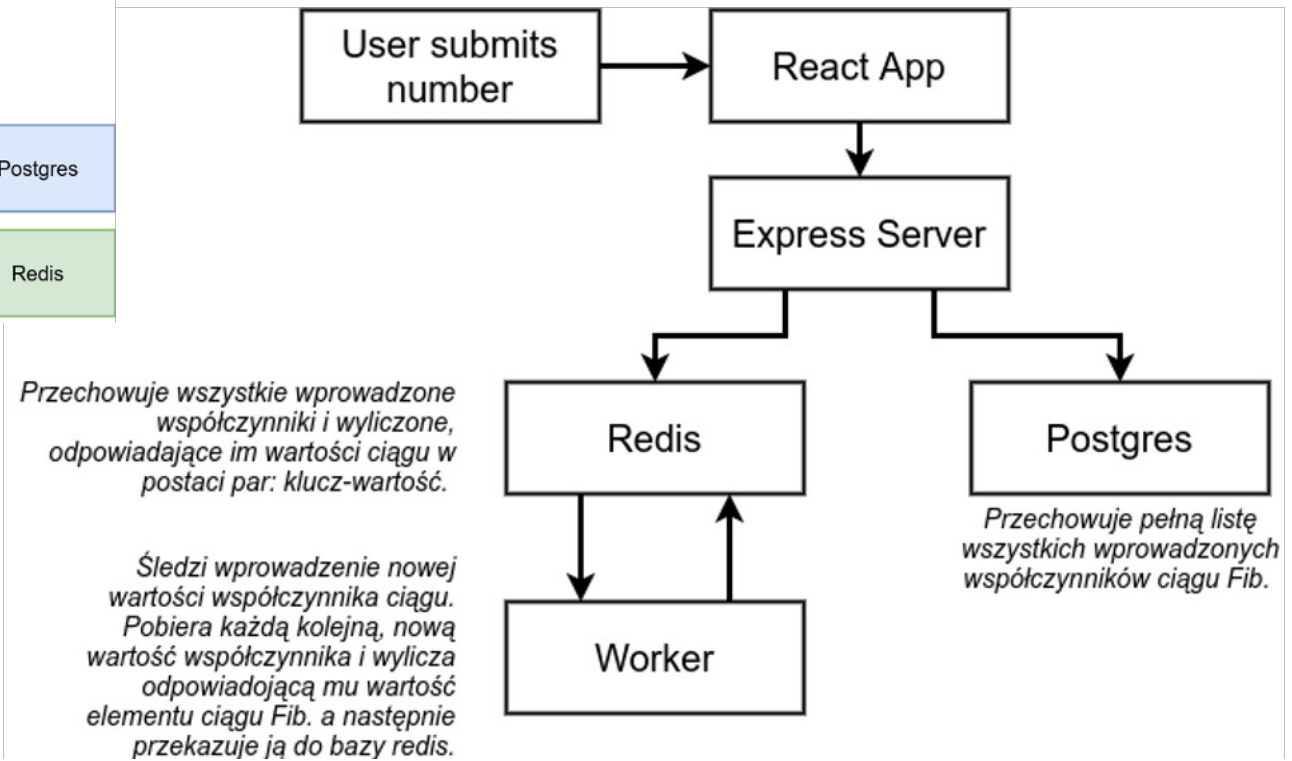
PFSwChO – Laboratorium11

Recap z poprzedniego laboratorium

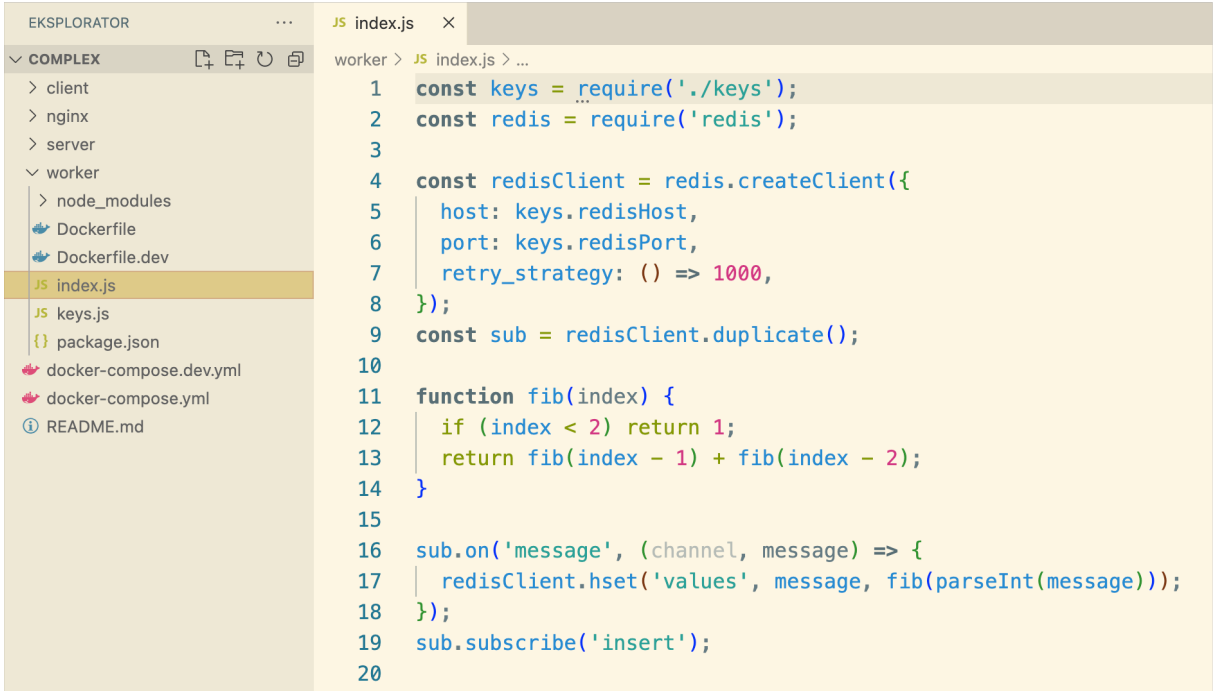




PFSwChO – Laboratorium11

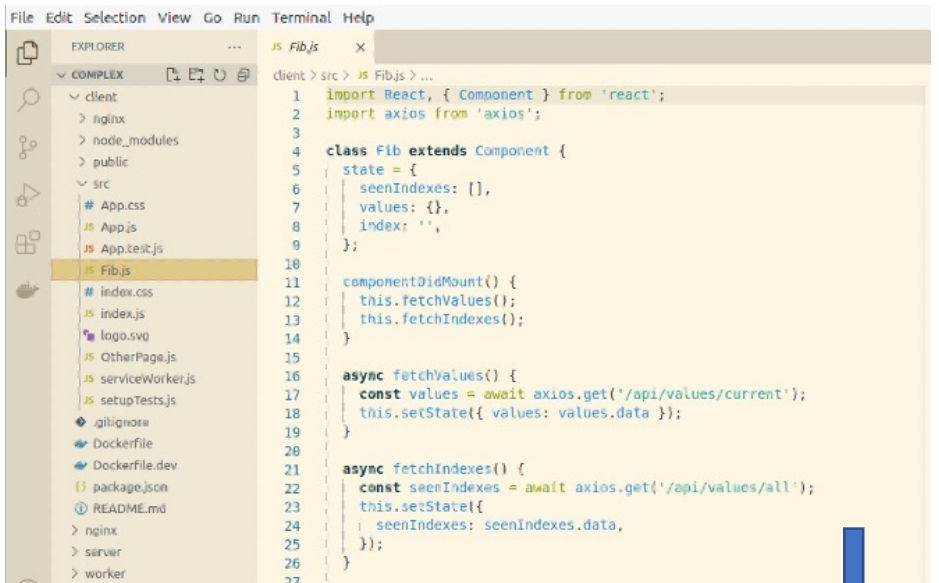
Należy wykorzystać pliki źródłowe, dostępne na moodle – plik: **Lab11.zip**.
Następnie należy rozpakować ten plik w katalogu domowym – do podkatalogu o nazwie ***complex***.
Analogicznie jak na poprzednich laboratoriach należy utworzy repozytorium na GitHub oraz zsynchronizować je z katalogiem *complex*.

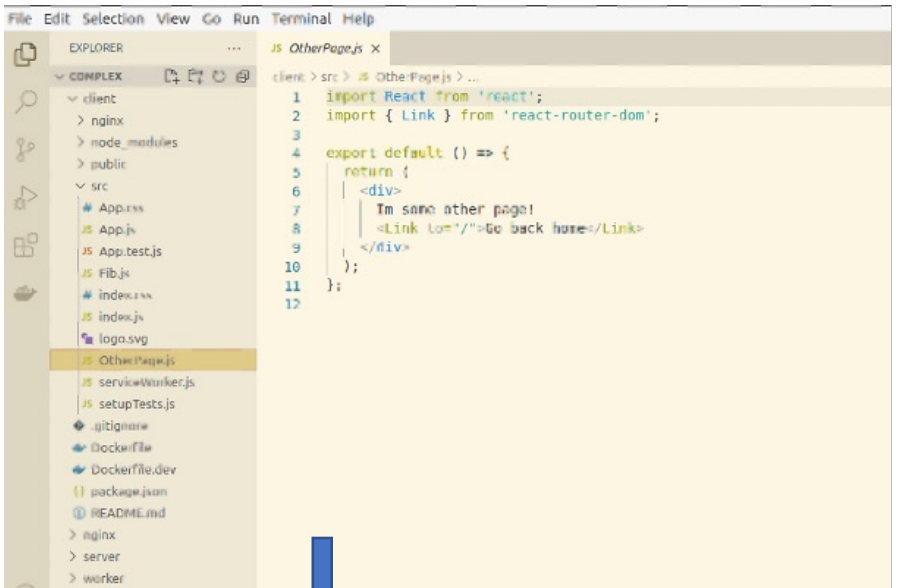
Przykładowa usługa wielo-kontenerowa – funkcja *fib* w mikrousłudze “worker”

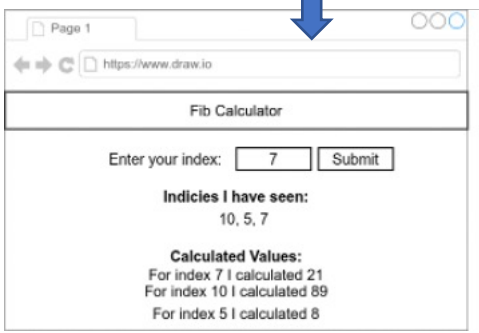


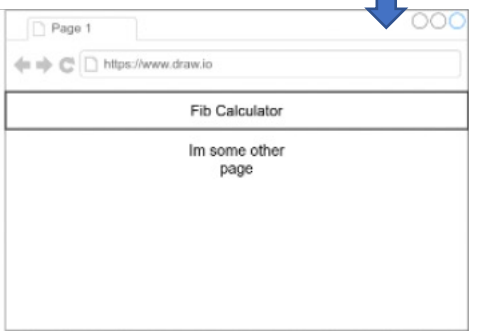
PFSwChO – Laboratorium11

Przykładowa usługa wielo-kontenerowa – UI routing (1)



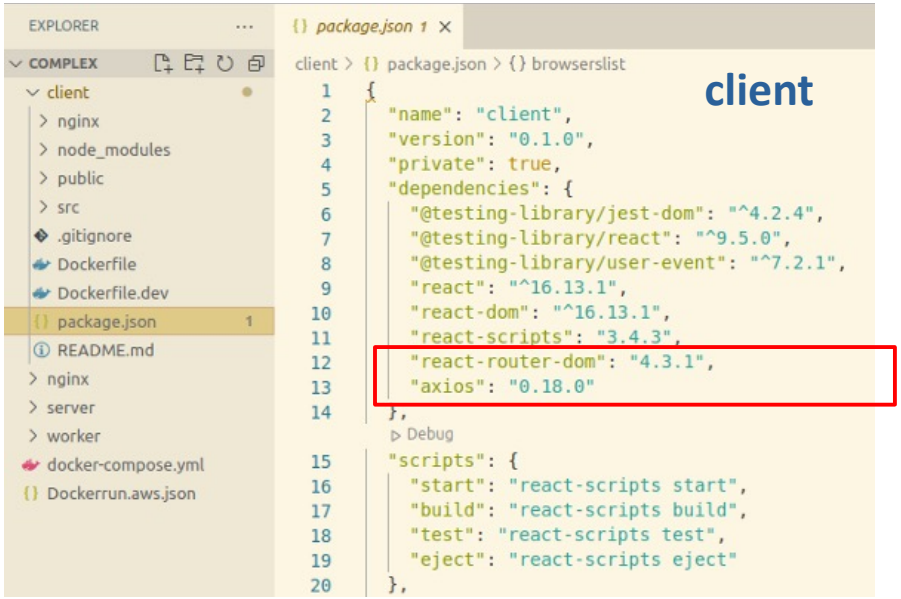




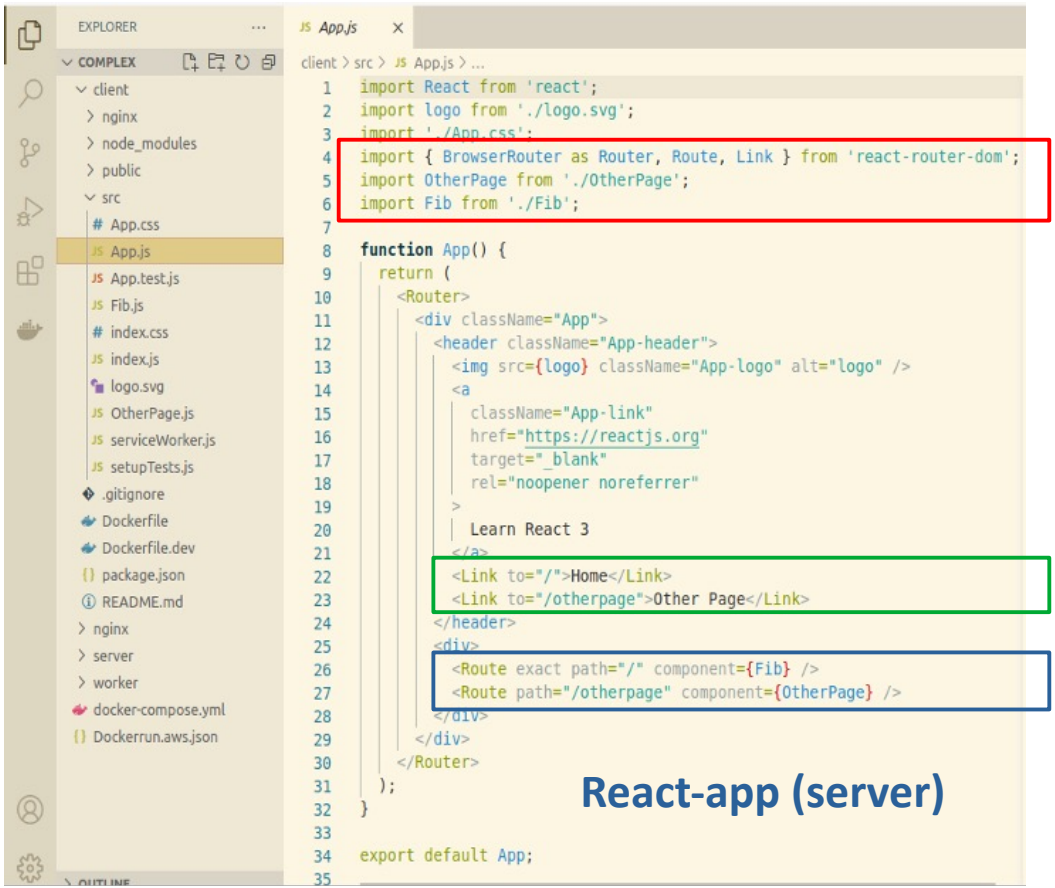


PFSwChO – Laboratorium11

Przykładowa usługa wielo-kontenerowa – UI routing (2)

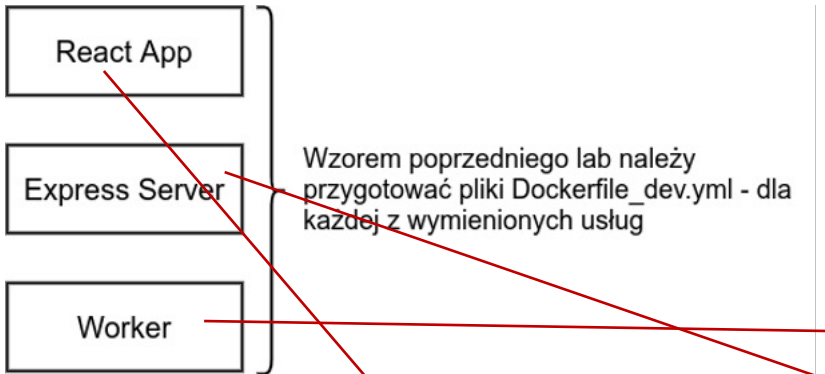


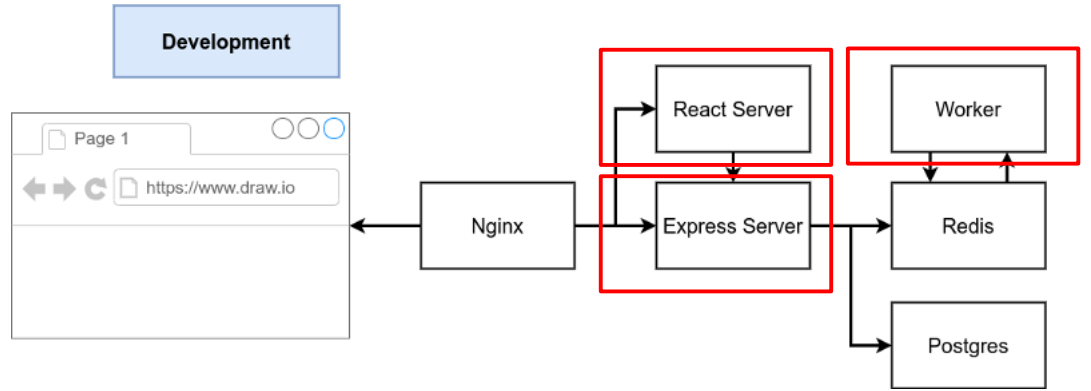
UWAGA: Wykorzystanie “routingu” we wdrażanej aplikacji zostanie finalnie skonfigurowane w pliku docker-compose



PFSwChO – Laboratorium11

Przykładowa usługa wielo-kontenerowa – faza rozwojowa (1) – Dockerfile.dev





client > Dockerfile.dev > ...

```
1  FROM node:alpine
2  WORKDIR "/app"
3  COPY ./package.json ./
4  RUN npm install
5  COPY . .
6  CMD ["npm", "run", "start"]
```

client

server > Dockerfile.dev > FROM

```
1  FROM node:14.14.0-alpine
2  WORKDIR "/app"
3  COPY ./package.json ./
4  RUN npm install
5  COPY . .
6  CMD ["npm", "run", "dev"]
```

server

worker > Dockerfile.dev > ...

```
1  FROM node:alpine
2  WORKDIR "/app"
3  COPY ./package.json ./
4  RUN npm install
5  COPY . .
6  CMD ["npm", "run", "dev"]
```

worker

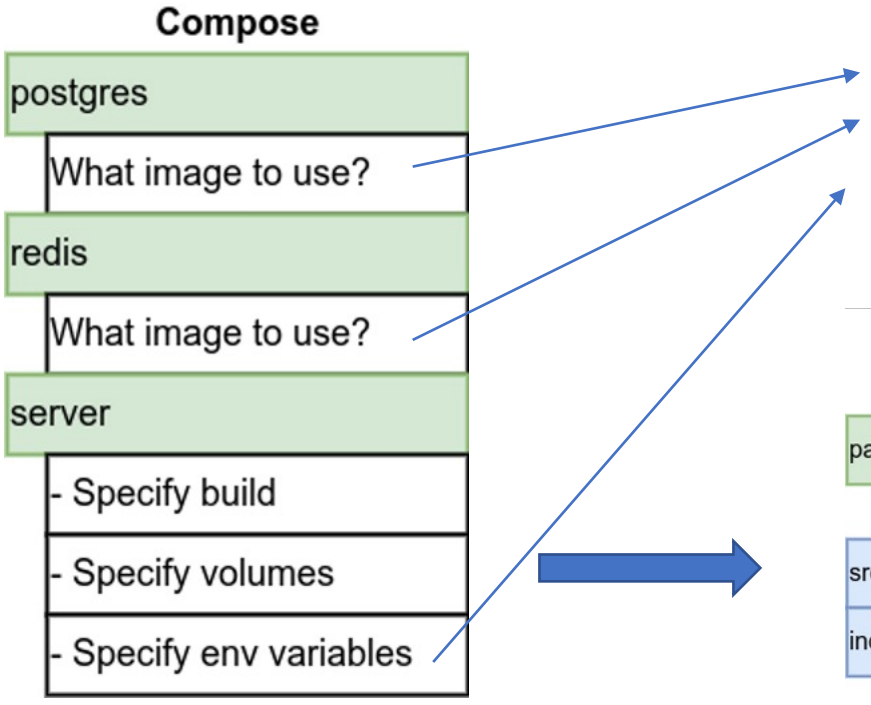
worker > {} package.json > ...

```
1  {
2    "dependencies": {
3      "nodemon": "1.18.3",
4      "redis": "2.8.0"
5    },
       ▷ Debug
6    "scripts": {
7      "start": "node index.js",
8      "dev": "nodemon"
9    }
10 }
11
```

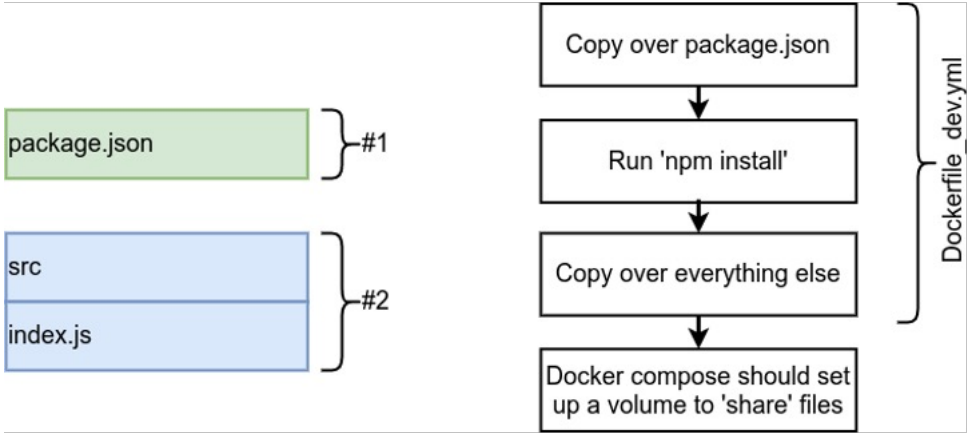
UWAGA: przed dalszymi etapami ZAWSZE należy zbudować testowo obrazy i uruchomić na ich podstawiekolejne kontenery. Wszelkie błędy nie wynikające z braku możliwości podłączenia do współpracujących komponentów oznaczają błąd w przygotowaniu opisu obrazu.

PFSwChO – Laboratorium11

Przykładowa usługa wielo-kontenerowa
– faza rozwojowa (2) – docker compose (1)



Deklaracja zmiennych środowiskowych !!!!!
Będzie niezwykle istotne podczas przejścia do fazy wdrażania (deploy) !!!!!!



PFSwChO – Laboratorium11

Przykładowa usługa wielo-kontenerowa
– faza rozwojowa (3) – docker compose (2) – podstawowe deklaracje dla: postgres i redis

docker-compose.yml

```yaml
version: '3'
services:
  postgres:
    image: 'postgres:latest'
    environment:
      - POSTGRES_PASSWORD=postgres_password
```

docker-compose.yml

```yaml
version: '3'
services:
  postgres:
    image: 'postgres:latest'
    environment:
      - POSTGRES_PASSWORD=postgres_password
  redis:
    image: 'redis:latest'
```

Environment Variables

The PostgreSQL image uses several environment variables which are easy to miss. The only variable required is `POSTGRES_PASSWORD`, the rest are optional.

Warning: the Docker specific variables will only have an effect if you start the container with a data directory that is empty; any pre-existing database will be left untouched on container startup.

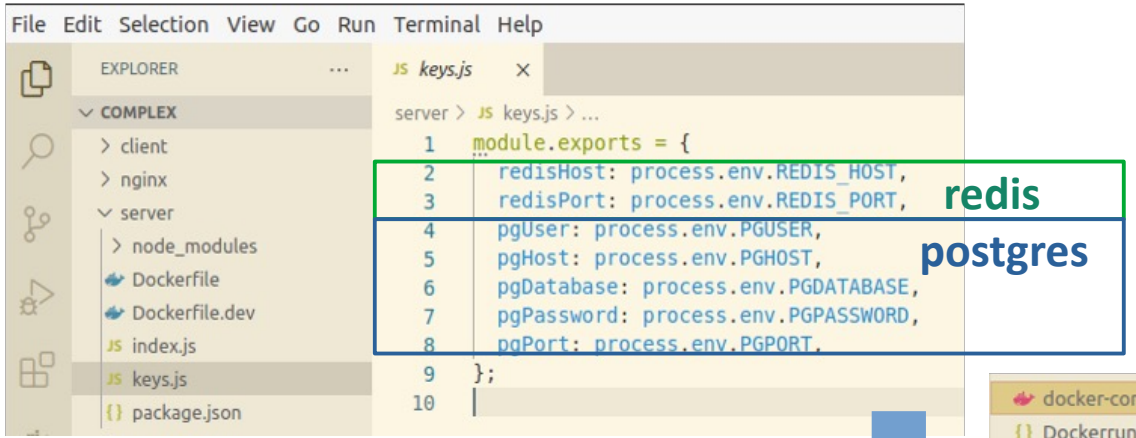
https://hub.docker.com/_/postgres

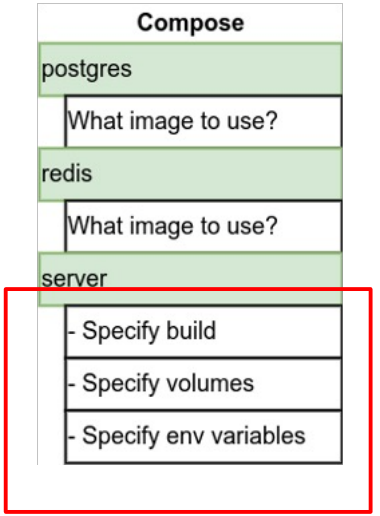
https://hub.docker.com/_/redis

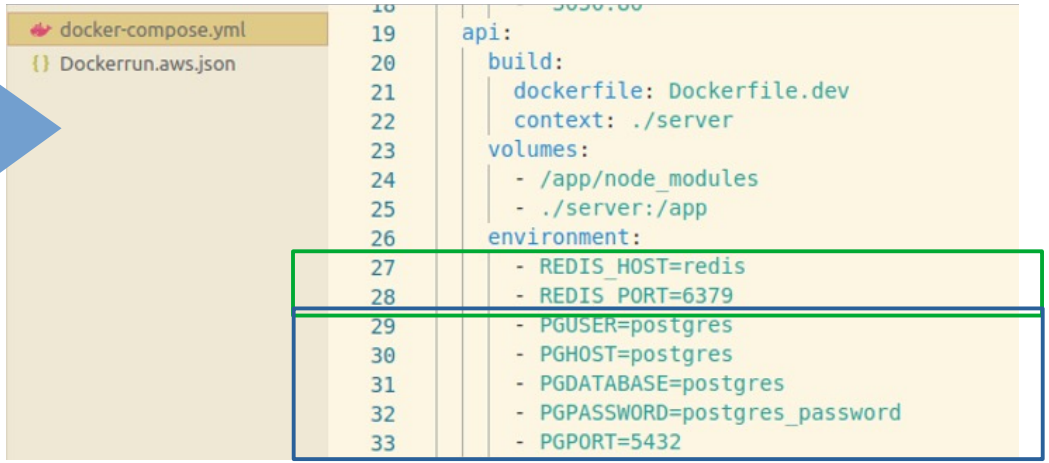
Zawsze należy zapoznać się z AKTUALNĄ dokumentacją wybranych (rozważanych do wyboru) obrazów - w omawianym przykładzie: z dokumentacją na DockerHub.

PFSwChO – Laboratorium11

Przykładowa usługa wielo-kontenerowa
– faza rozwojowa (4) – docker compose (3) - server



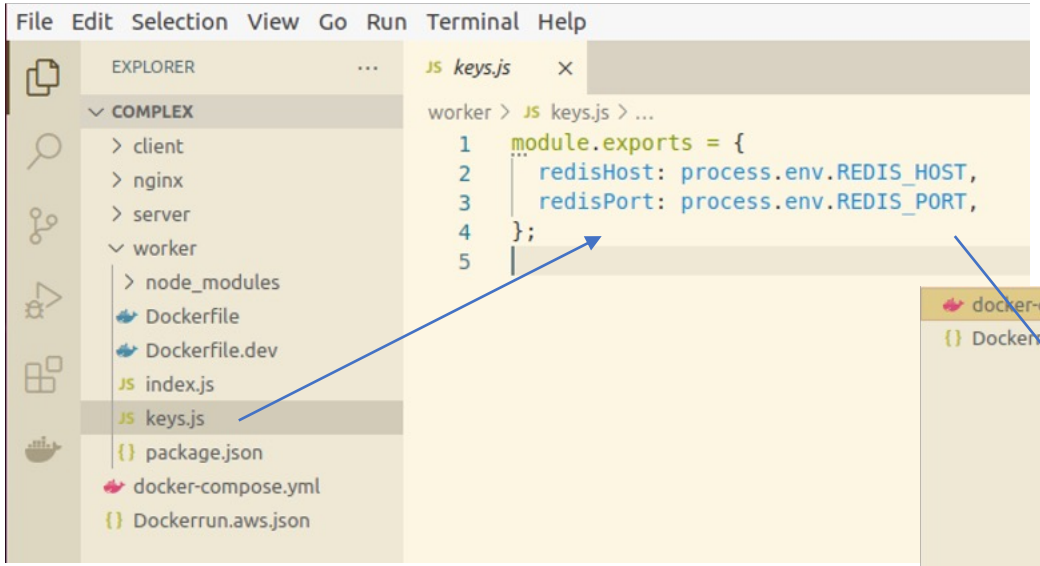


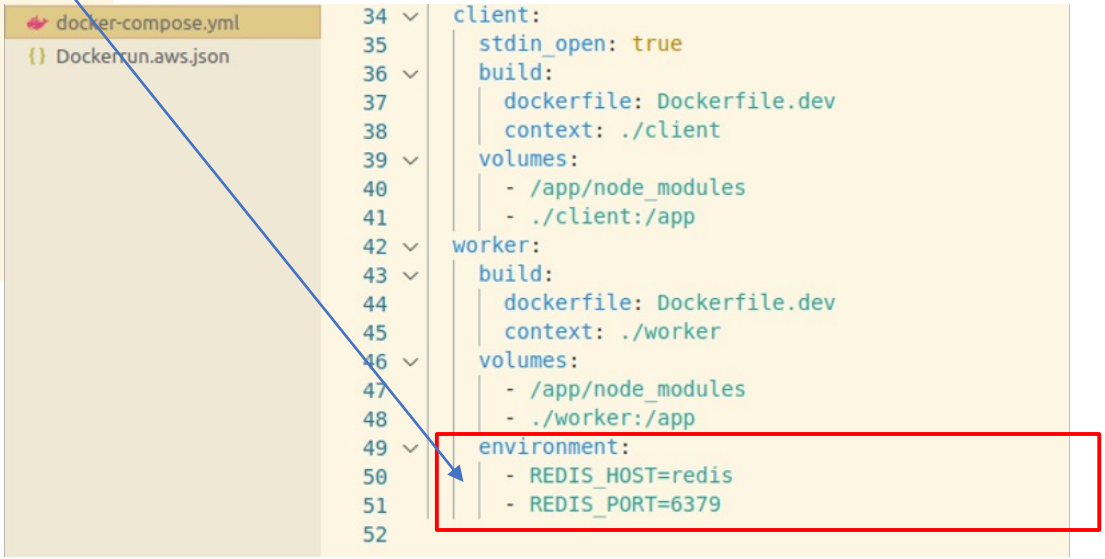


UWAGA: kolejny raz – dobrą praktyką jest uruchomić usługę (docker compose up) i sprawdzić czy poszczególna usługa nie generuje błędów oraz poprawnie “nasłuchuje” na połączenia na wybranym porcie.

PFSwChO – Laboratorium11

Przykładowa usługa wielo-kontenerowa
– faza rozwojowa (5) – docker compose (4) – worker

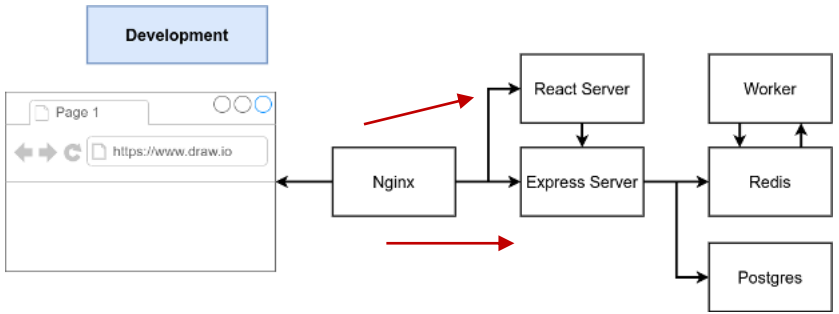




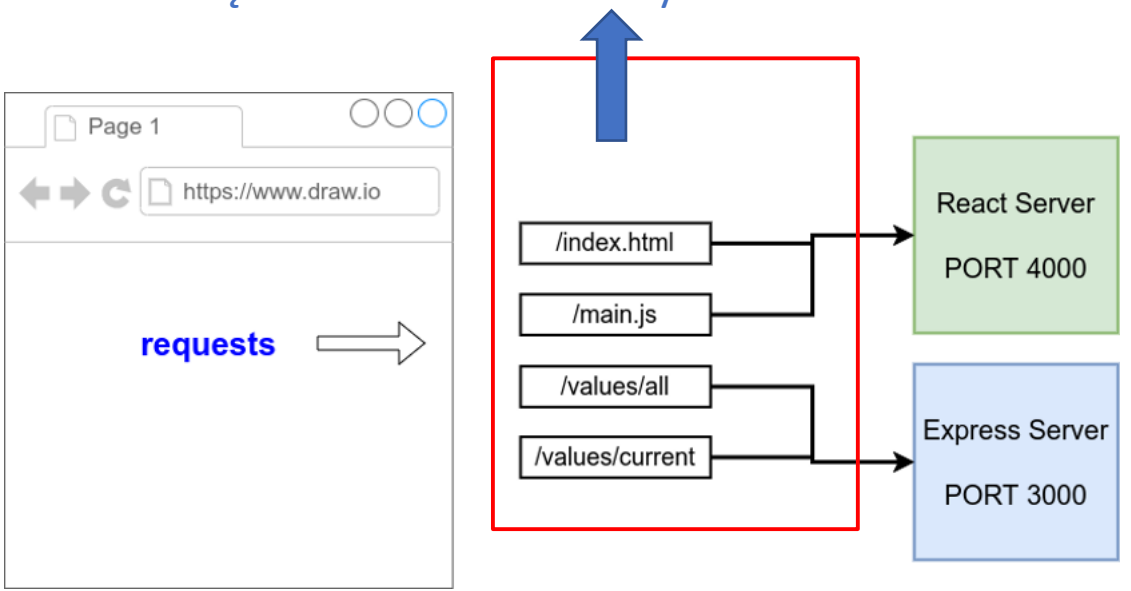
Postępowanie analogiczne jak w przypadku server-a (poprzedni slajd)

PFSwChO – Laboratorium11

Przykładowa usługa wielo-kontenerowa
– faza rozwojowa (6) – docker compose (5) – powrót do tematu: UI routing (path routing)



Potrzebne jest rozwiązanie “patch routingu”, które może zaproponować np. serwer nginx. Jednak implementacja rozwiązania zależy od przyjętych rozwiązań w kodzie źródłowym.



UWAGA: rozwiązanie to nie ma to nic wspólnego z rolą jaką pełnił serwer ngnix w przykładach odnoszących się do usługi jedno-kontenerowej

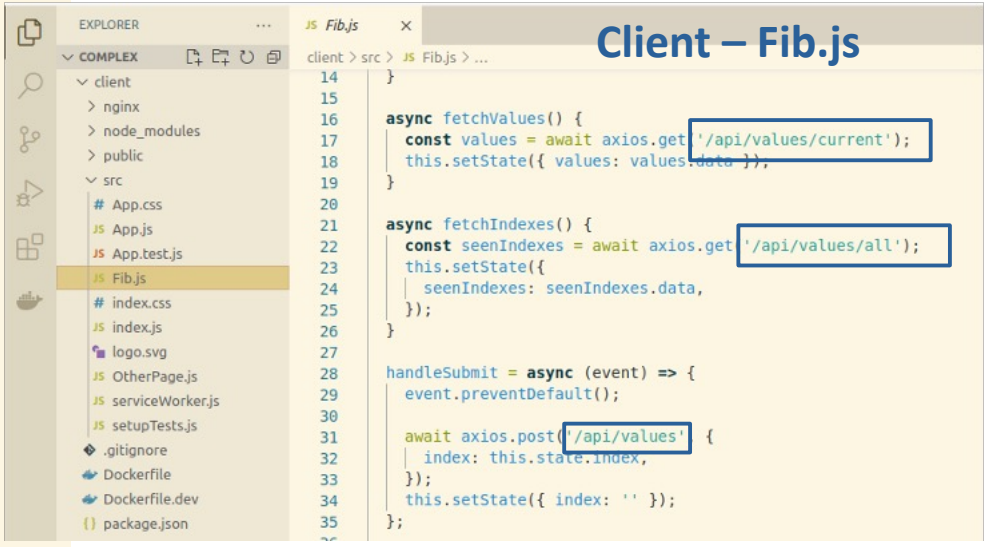
PFSwChO – Laboratorium11

Przykładowa usługa wielo-kontenerowa – faza rozwojowa (7) – docker compose (6)
– propozycja rozwiązania tematu: UI routing (path routing) (1)

React-app (server)

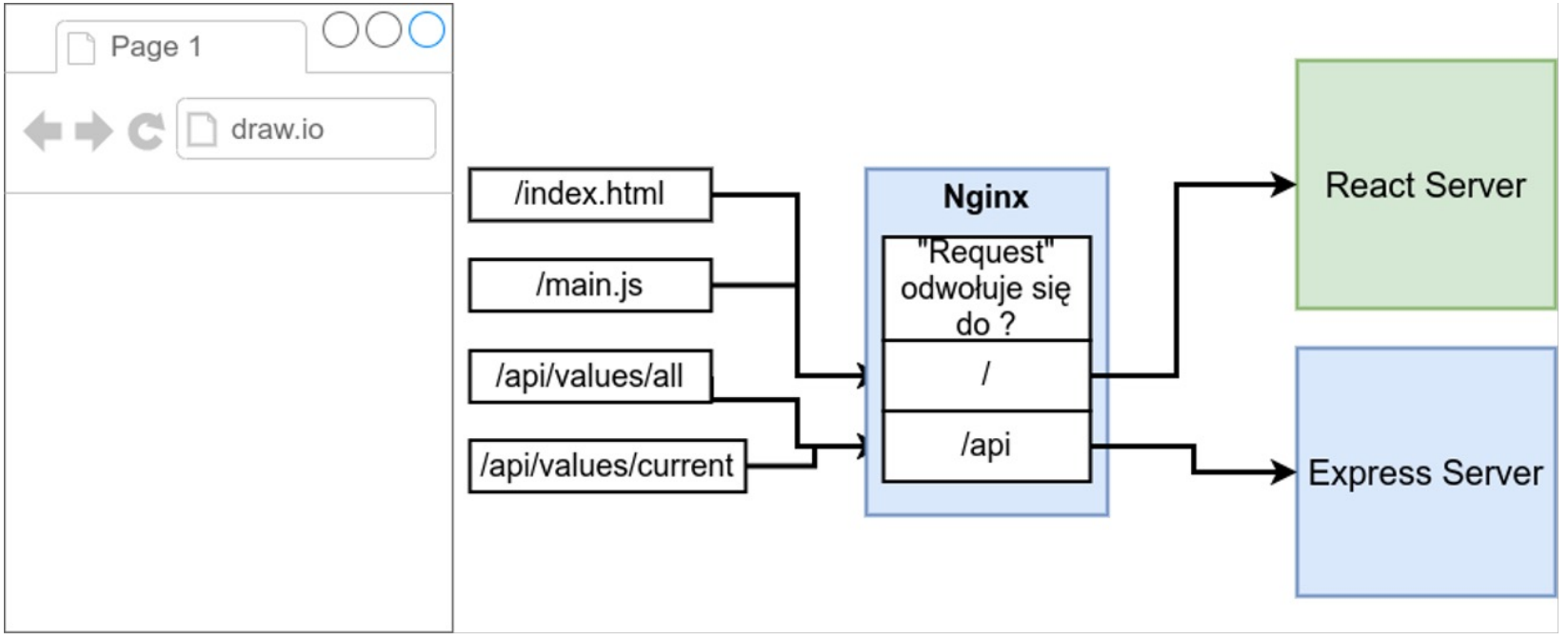
Slajd 4 + poniższe definicje

Client – Fib.js



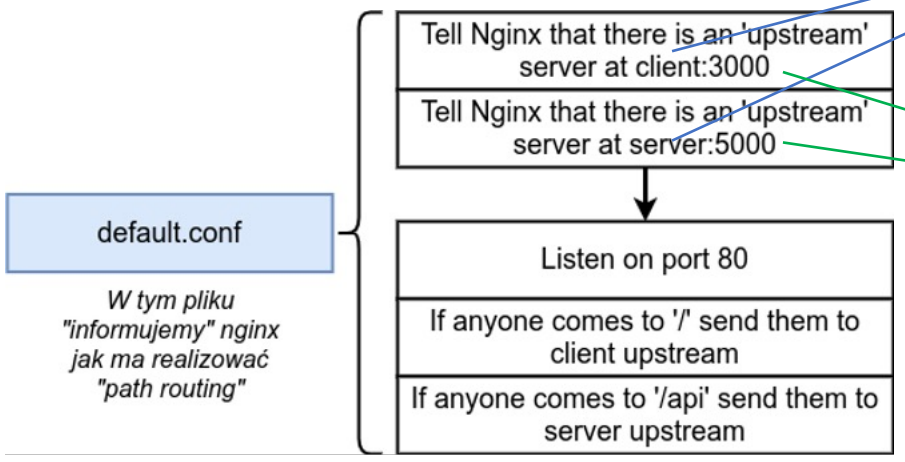
PFSwChO – Laboratorium11

Przykładowa usługa wielo-kontenerowa – faza rozwojowa (8) – docker compose (7) – propozycja rozwiązania tematu: UI routing (path routing) (2)



PFSwChO – Laboratorium11

Przykładowa usługa wielo-kontenerowa – faza rozwojowa (9) – docker compose (8) – propozycja rozwiązania tematu: UI routing (path routing) (3)



UWAGA: nazwy hostów docelowych (server oraz client) muszą być zgodne z nazwami usług (services) zdeklarowanych w pliku docker-compose.yml

Porty wynikają z deklaracji w kodzie źródłowym opracowywanej usługi, np. server

PFSwChO – Laboratorium11

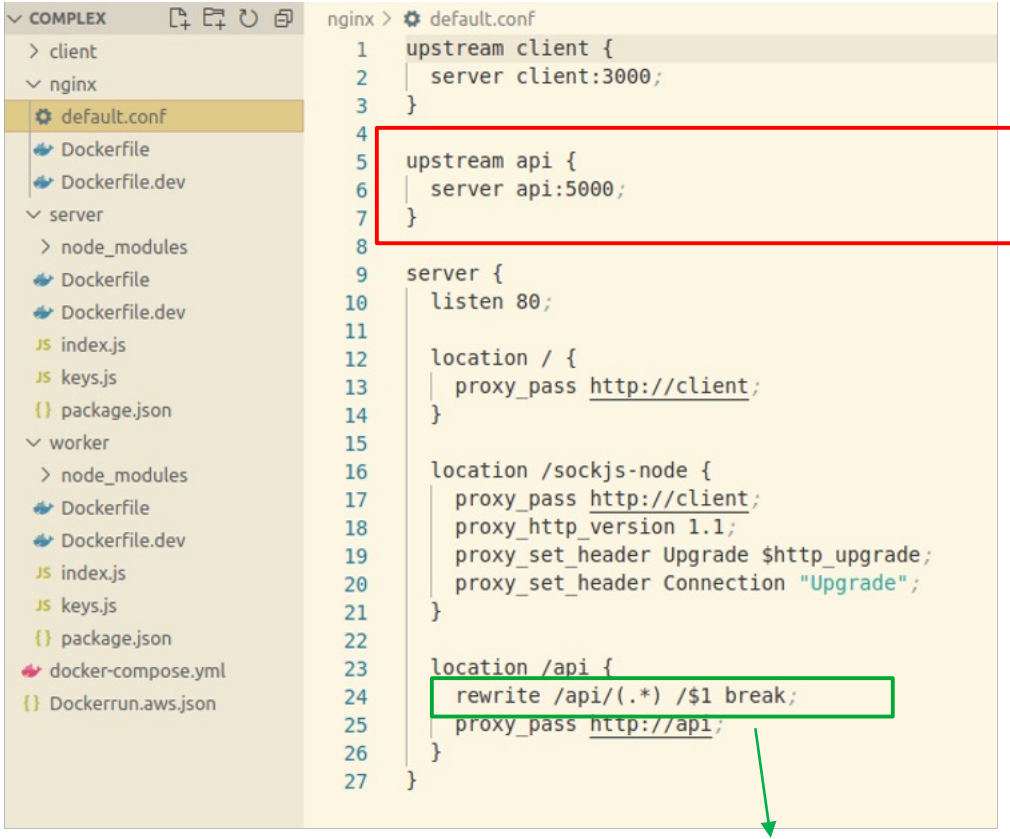
Przykładowa usługa wielo-kontenerowa – faza rozwojowa (10) – docker compose (9) – konfiguracja nginx (1)

default.conf

```
1  upstream client {
2    server client:3000;
3  }
4
5  upstream server {
6    server server:5000;
7  }
```

!!! Pojawi się ERROR !!!

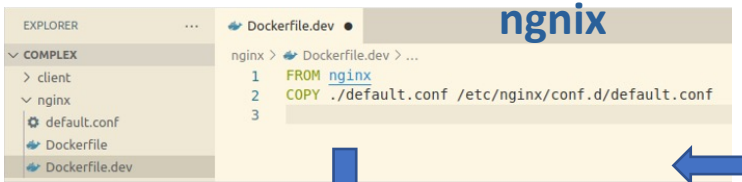
Należy zmienić nazwę serwisu w pliku docker-compose.dev.yml (w przykładowych plikach zmieniono na *api*



Opcjonalna reguła "wycinająca" *api* ze ścieżki requestu przekazywanego do upstream serwera – większa ogólność serwera upstream

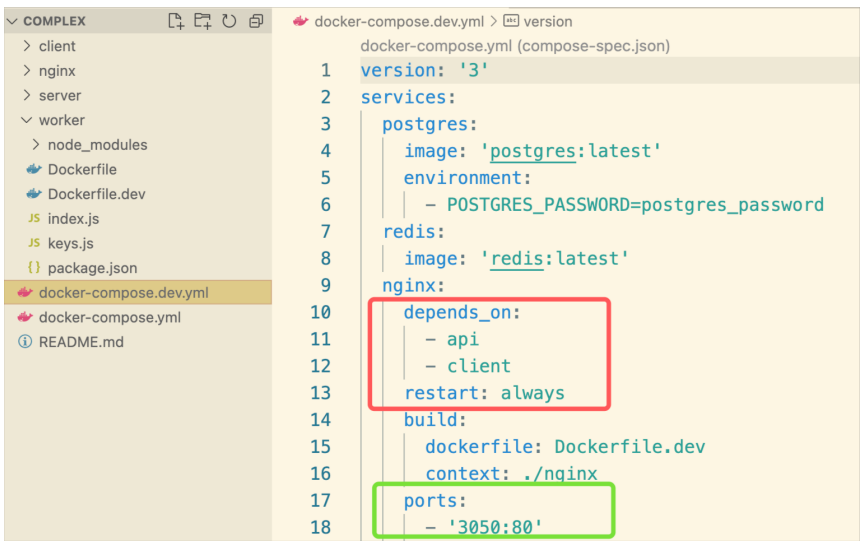
PFSwChO – Laboratorium11

Przykładowa usługa wielo-kontenerowa – faza rozwojowa (11) – docker compose (10) – konfiguracja nginx (2)



ngnix

https://hub.docker.com/_/nginx



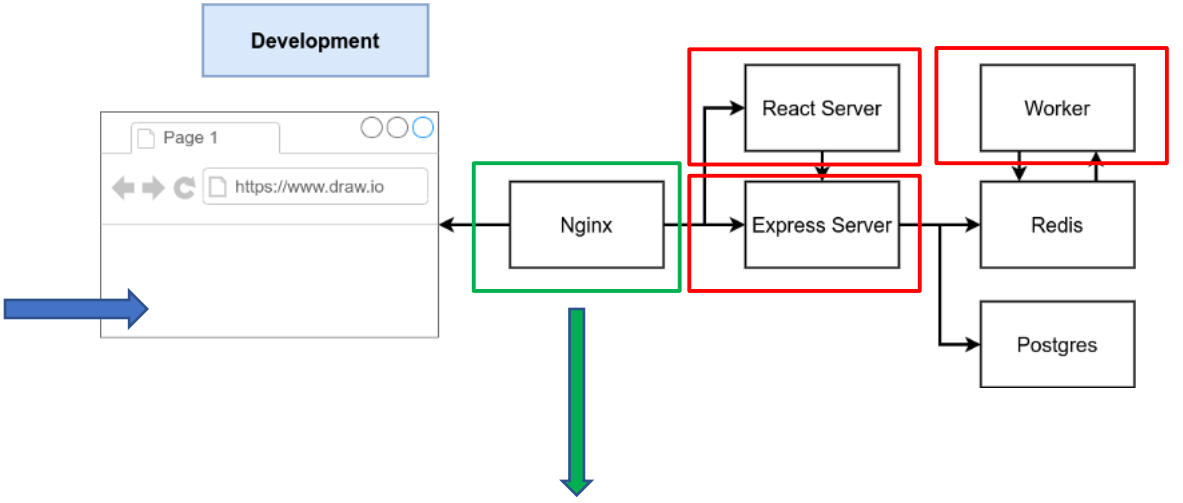
Należy uruchomić aplikację i zweryfikować poprawność jej działania

```
docker compose -f docker-compose.dev.yml up --build
```

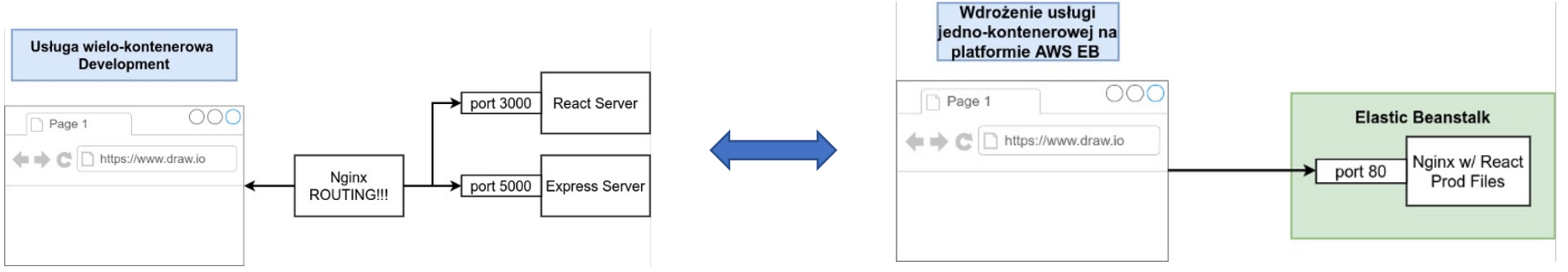
PFSwChO – Laboratorium11

Usługa wielo-kontenerowa – tworzenie “produkcyjnych” Dockerfiles (1)

Pliki Dockerfile dla zaznaczonych czerwonym obramowaniem skladowych usługi nie różnią istnie od wersji Dockerfile.dev (prosze prześledzić dostarczone kody).

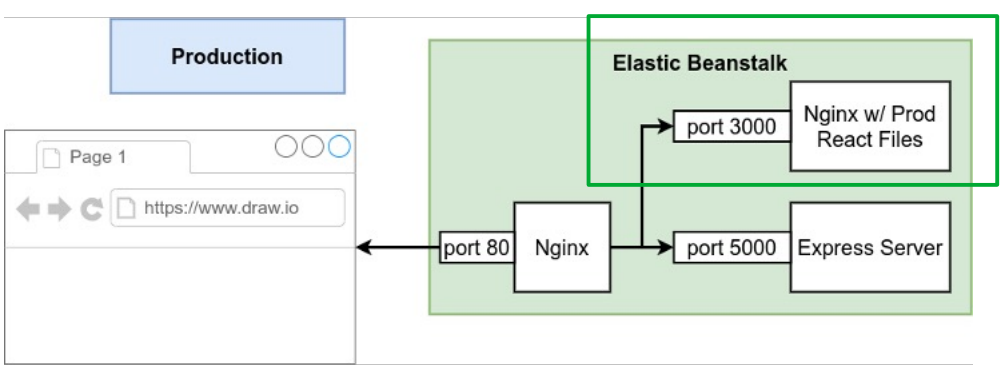


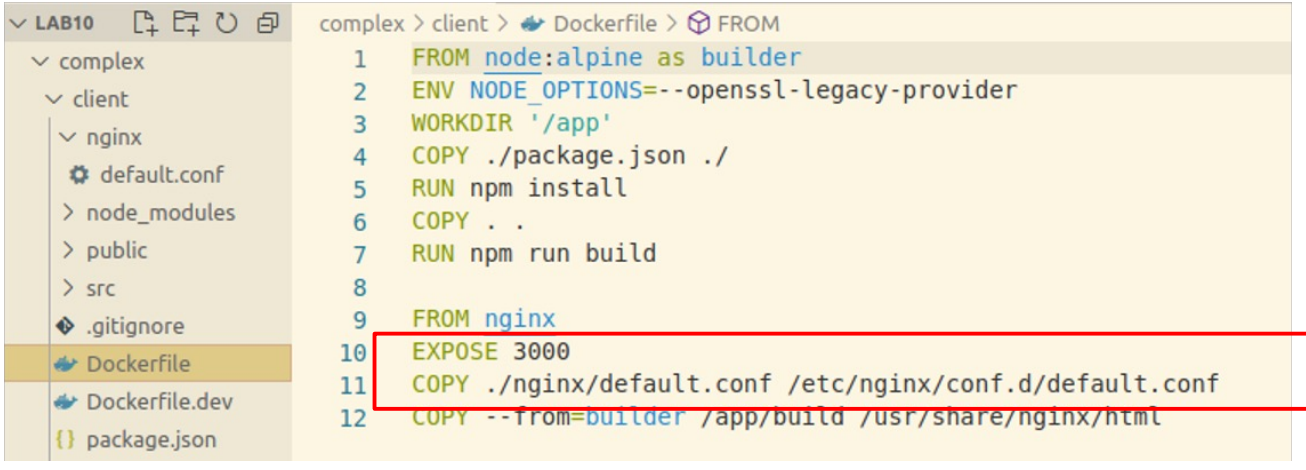
Sytuacja jest bardziej skomplikowana dla serwera nginx



PFSwChO – Laboratorium11

Usługa wielo-kontenerowa – tworzenie “produkcyjnych” Dockerfiles (2)



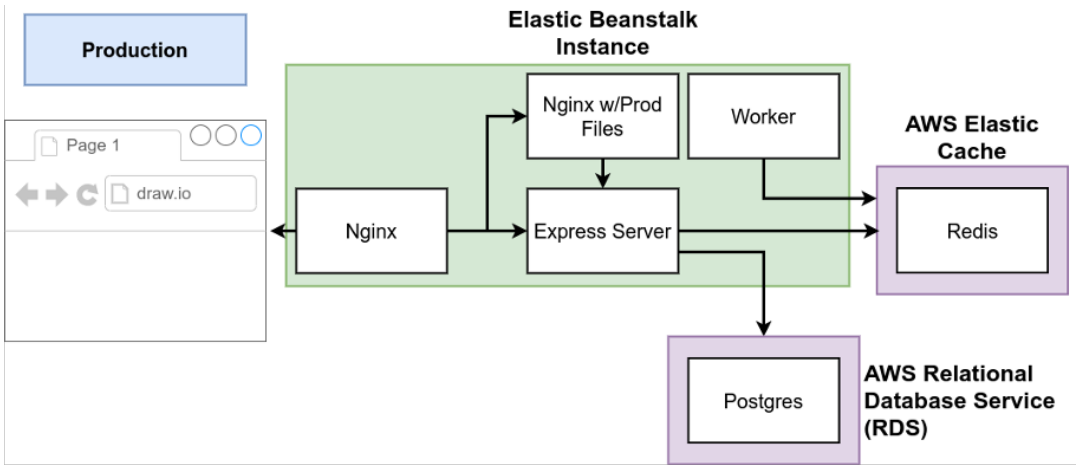


W usłudze EBS tworzymy nową aplikację (w dalszych przykładach: **Fib-multi**), analogicznie jak na poprzednim laboratorium

PFSwChO – Laboratorium11

Usługa wielo-kontenerowa – wdrożenie na AWS (1)
- bazy danych (1)

W pliku “produkcyjnym” docker-compose.yml brak jest definicji usługi redis oraz postgres !!!! W praktyce można zdefiniować “własny” kontener ale ZDECYDOWANIE częściej wykorzystywane są usługi gwarantujące bezpieczne i skalowalne wdrożenie predefiniowanych baz danych



.travis.yml | docker-compose.yml | docker-compose.dev.yml

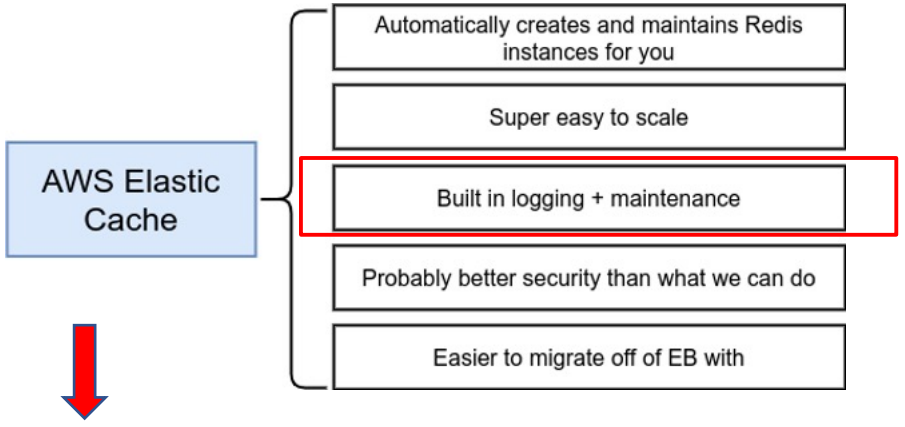
complex > docker-compose.yml

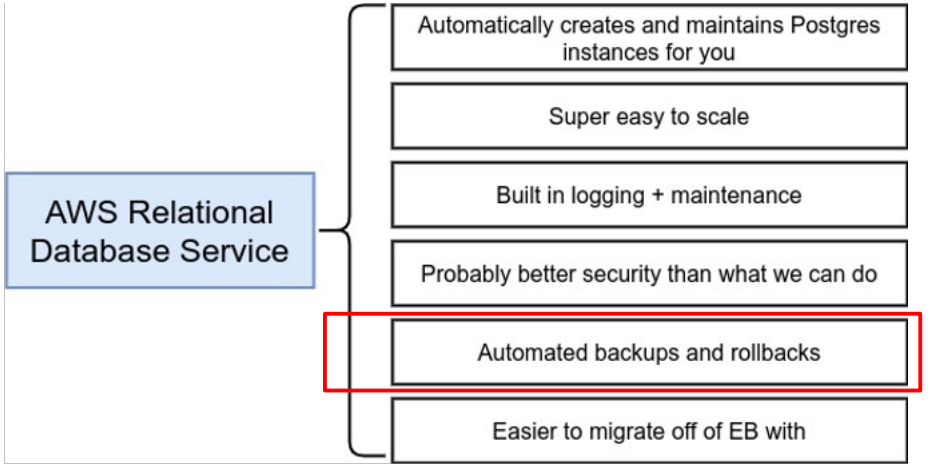
```
1  version: "3"
2  services:
3    client:
4      image: "spg51/lab:fib-client"
5      mem_limit: 128m
6      hostname: client
7    server:
8      image: "spg51/lab:fib-server"
9      mem_limit: 128m
10     hostname: api
11     environment:
12       - REDIS_HOST=$REDIS_HOST
13       - REDIS_PORT=$REDIS_PORT
14       - PGUSER=$PGUSER
15       - PGHOST=$PGHOST
16       - PGDATABASE=$PGDATABASE
17       - PGPASSWORD=$PGPASSWORD
18       - PGPORT=$PGPORT
19   worker:
20     image: "spg51/lab:fib-worker"
21     mem_limit: 128m
22     hostname: worker
23     environment:
24       - REDIS_HOST=$REDIS_HOST
25       - REDIS_PORT=$REDIS_PORT
26   nginx:
27     image: "spg51/lab:fib-nginx"
28     mem_limit: 128m
29     hostname: nginx
30     ports:
31       - "80:80"
32
```

PFSwChO – Laboratorium11

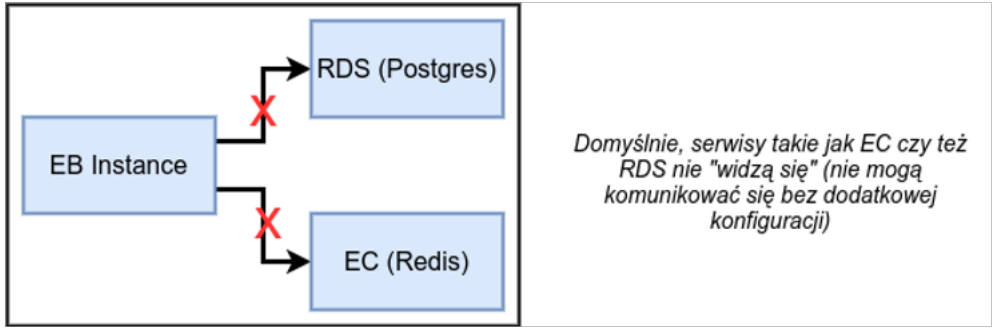
Usługa wielo-kontenerowa – wdrożenie na AWS (2)

- bazy danych (2)





UWAGA !

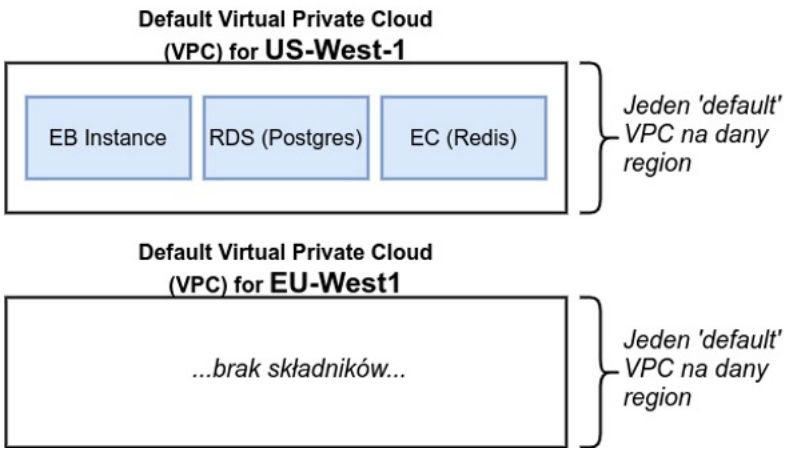


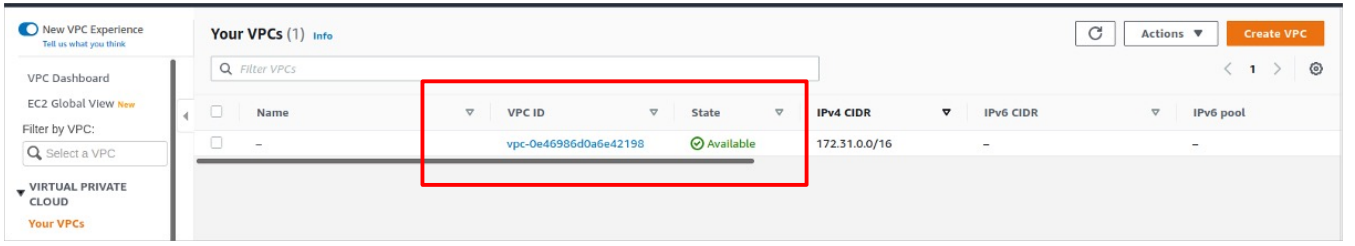
Należy skonfigurować kolejną usługę AWS – **VPC (ang. Virtual Private Cloud)**

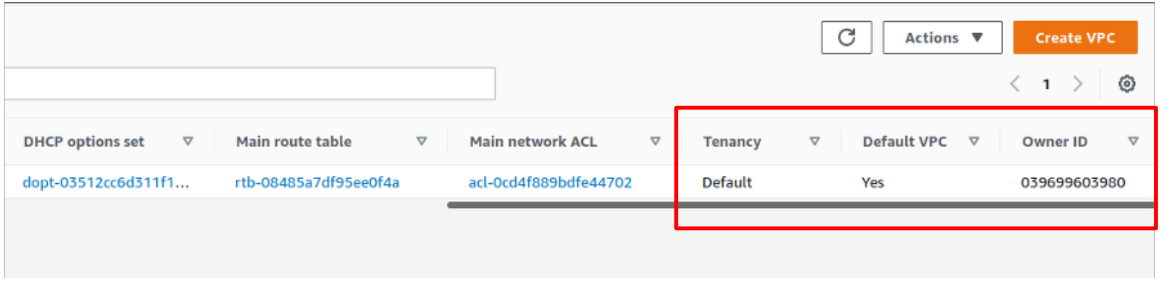
PFSwChO – Laboratorium11

Usługa wielo-kontenerowa – wdrożenie na AWS (3)
- pojęcie VPC (1)

Usługa VPC idea



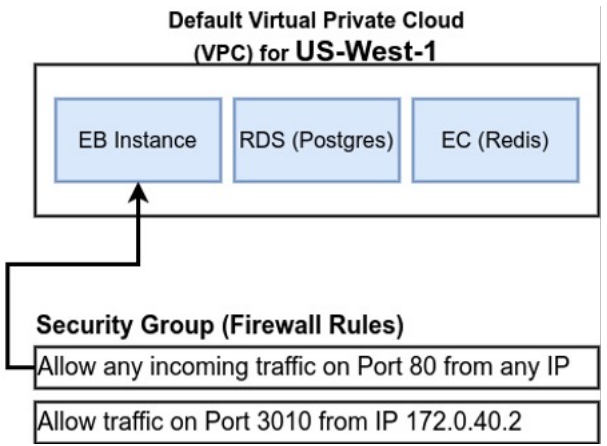


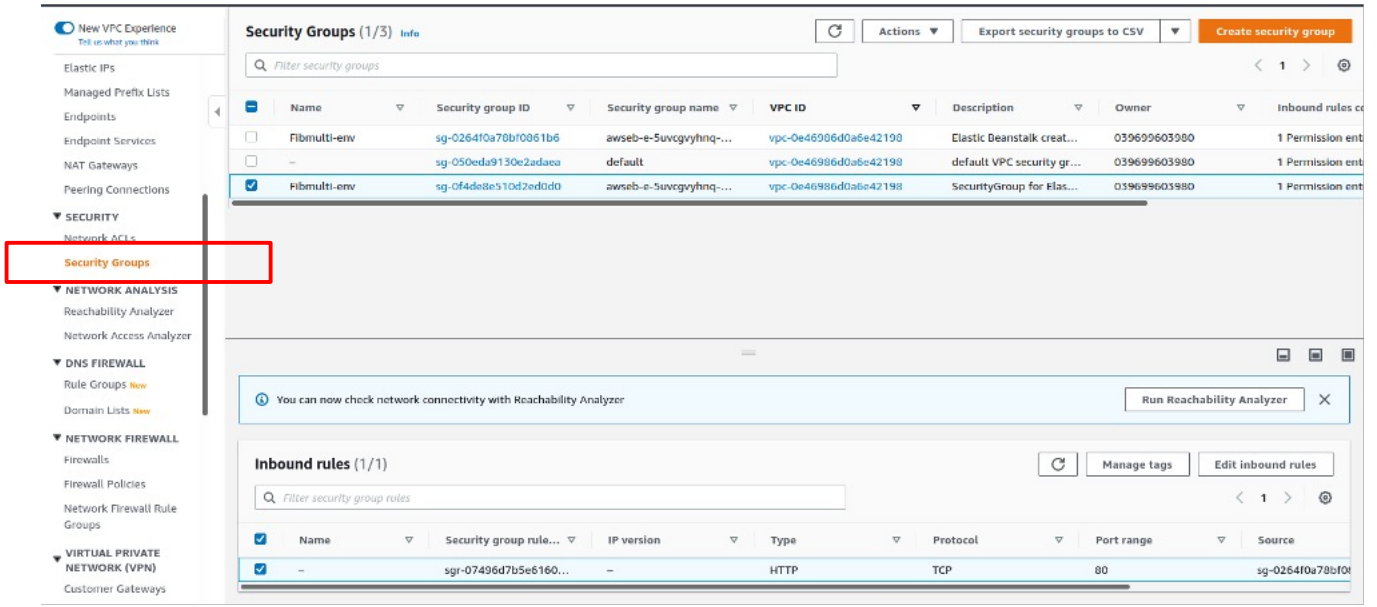


PFSwChO – Laboratorium11

Usługa wielo-kontenerowa – wdrożenie na AWS (4)
- pojęcie VPC (2) – Security Groups (1)

SG - ustawiane automatycznie podczas tworzenia środowiska



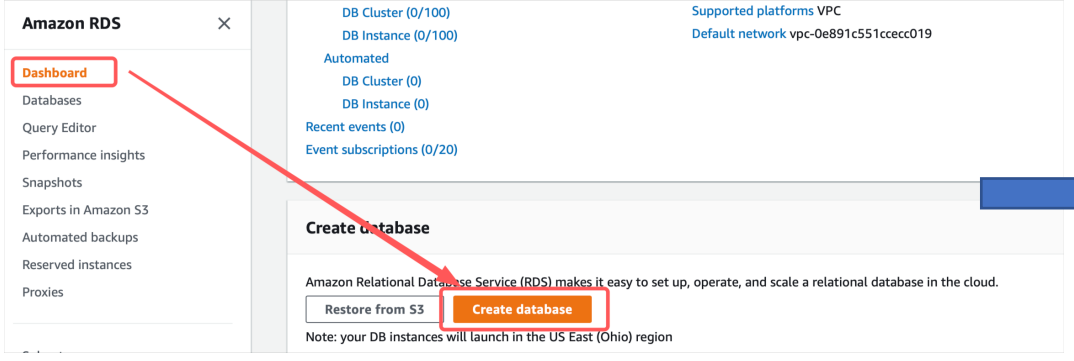


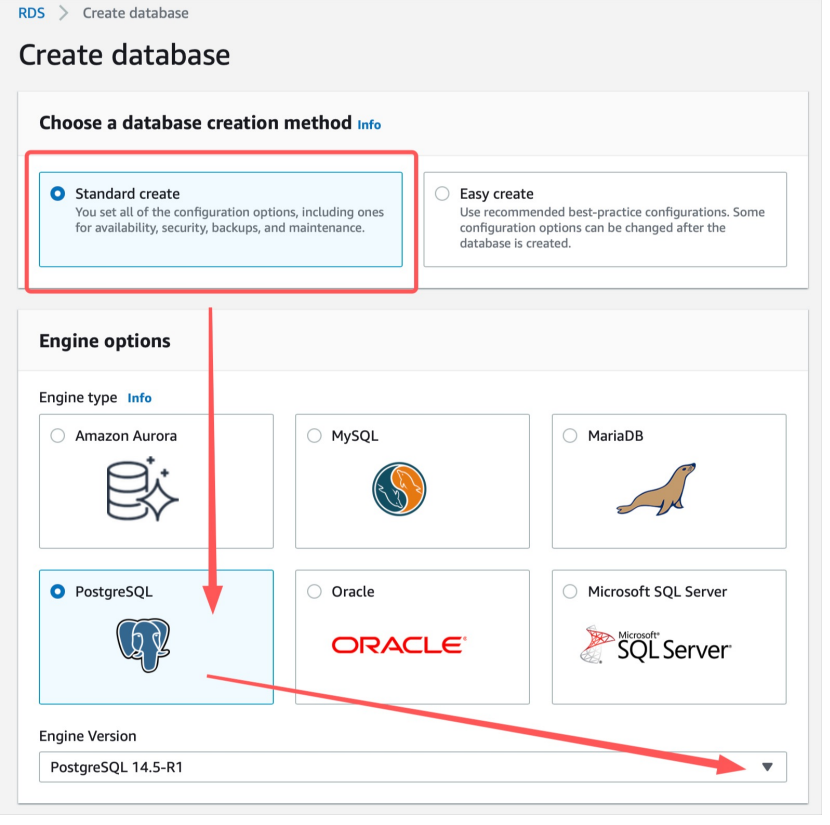
Należy zdefiniować nową, własną “security group”, która pozwoli na komunikację składowych usługi w obrębie danego VPC – ale to wymaga wcześniejszego zdefiniowania instacji baz danych (redis i postgres)

PFSwChO – Laboratorium11

Usługa wielo-kontenerowa – wdrożenie na AWS (5)
- usługa RDS (postgres) (1)

UWAGA: proszę nie zmieniać VPC do końca wdrażania usługi wielo-kontenerowej

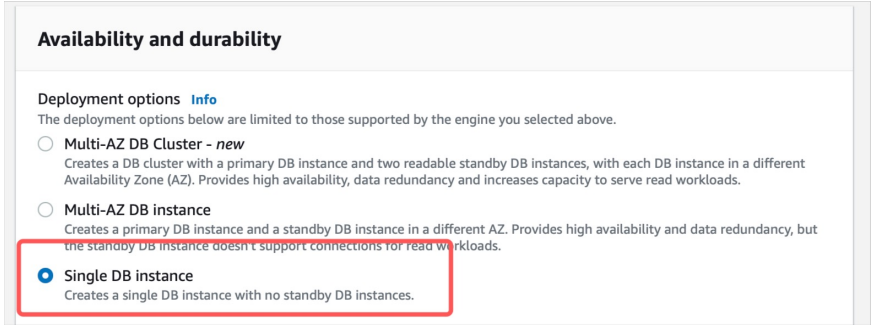


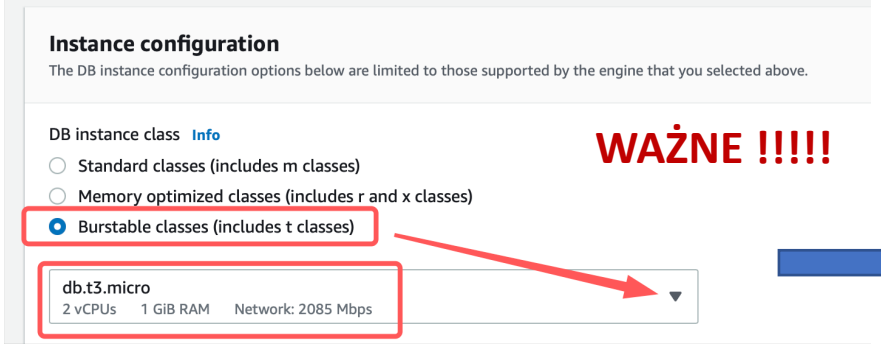


PFSwChO – Laboratorium11

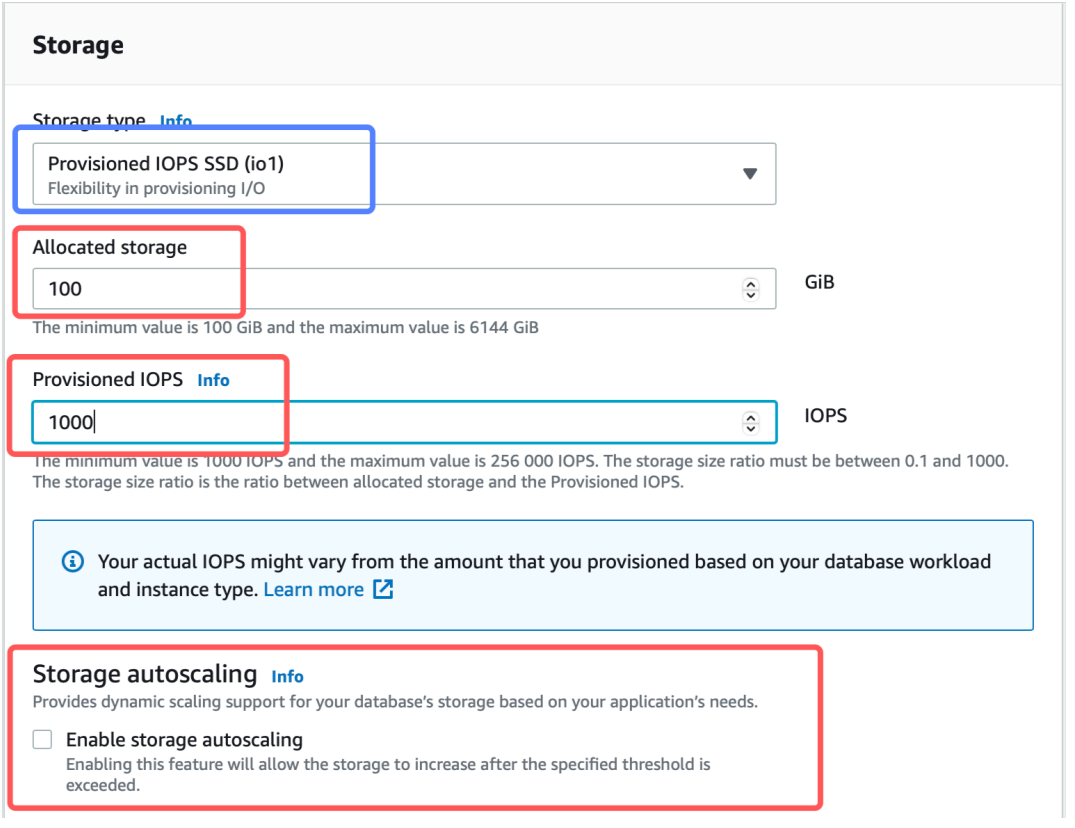
Usługa wielo-kontenerowa – wdrożenie na AWS (6) - usługa RDS (postgres) (2)

UWAGA: poniższe ustawienia mają silny wpływ na całkowite koszty korzystania z usługi RDS



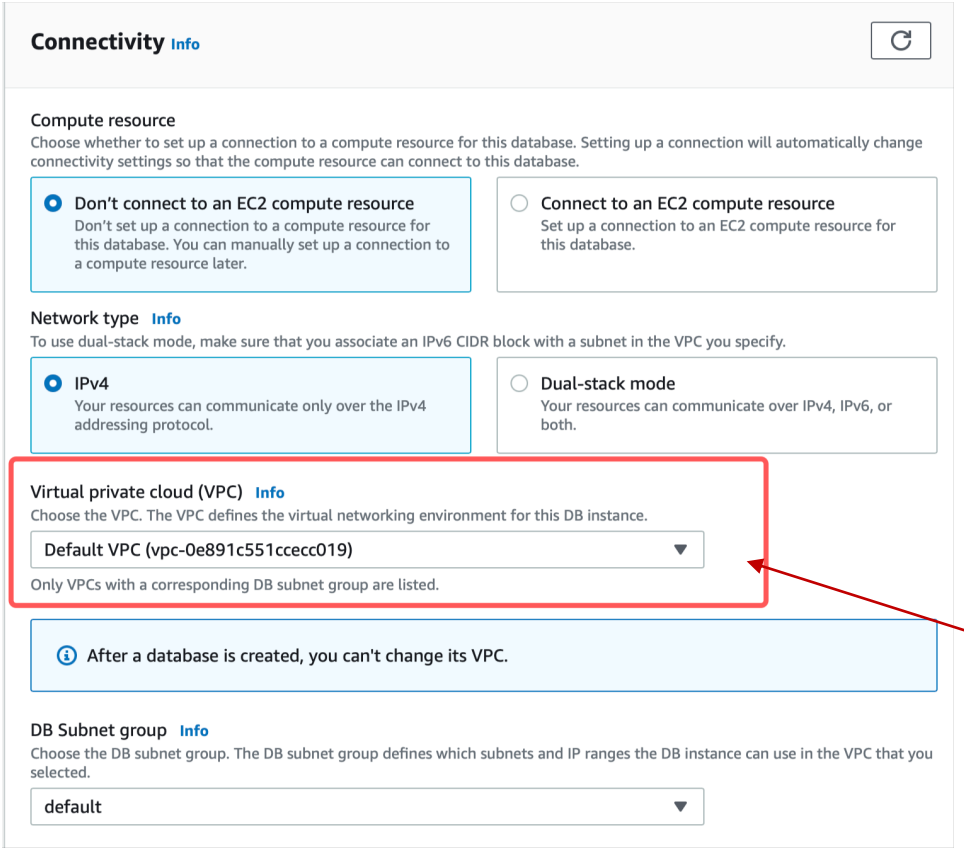


WAŻNE !!!!!



PFSwChO – Laboratorium11

Usługa wielo-kontenerowa – wdrożenie na AWS (7)
- usługa RDS (postgres) (3)



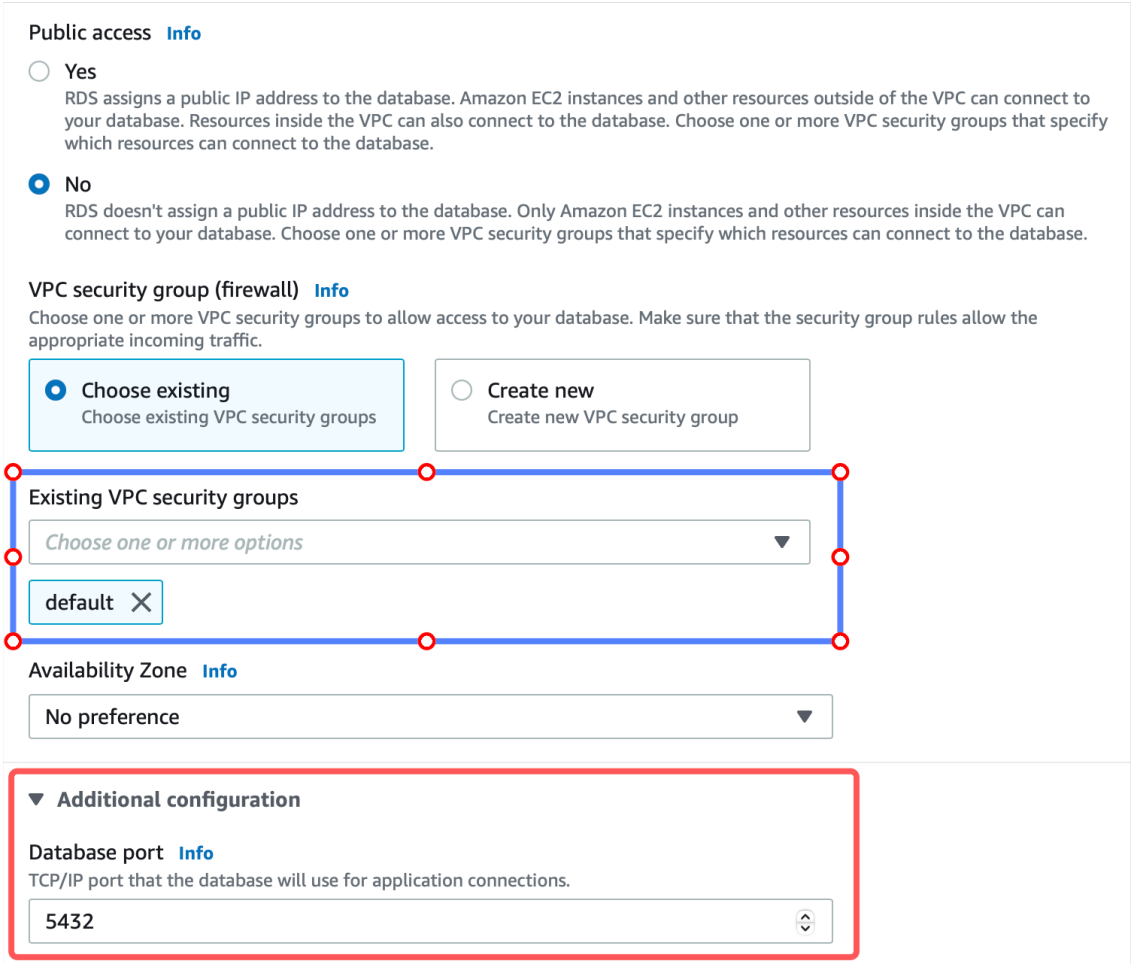
Prosze zawsze sprawdzać czy ustawiony jest poprawny numer VPC. W dalszym procesie konfiguracji tego ustawienia nie można zmienić !!!!!!!!.

PFSwChO – Laboratorium11

Usługa wielo-kontenerowa – wdrożenie na AWS (8) - usługa RDS (postgres) (4)

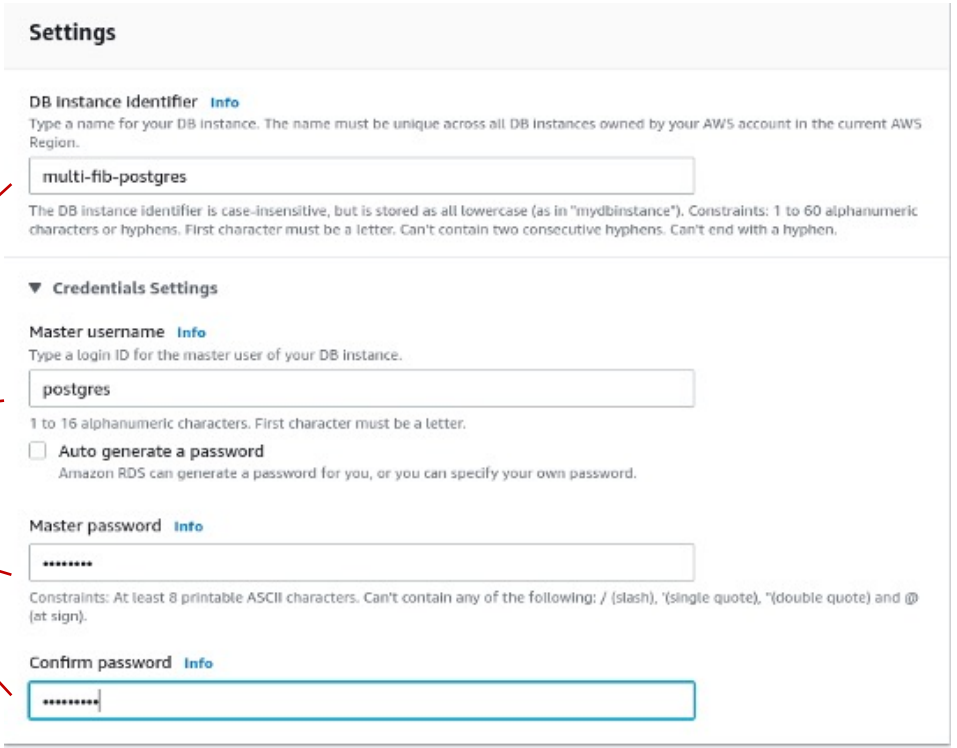
Nowa grupa bezpieczeństwa (Security Group) dla “naszej” usługi zostanie utworzona później

Należy zapamiętać port – będzie używany w plikach konfiguracyjnych usługi wielo-kontenerowej



PFSwChO – Laboratorium11

Usługa wielo-kontenerowa – wdrożenie na AWS (9)
- usługa RDS (postgres) (5)

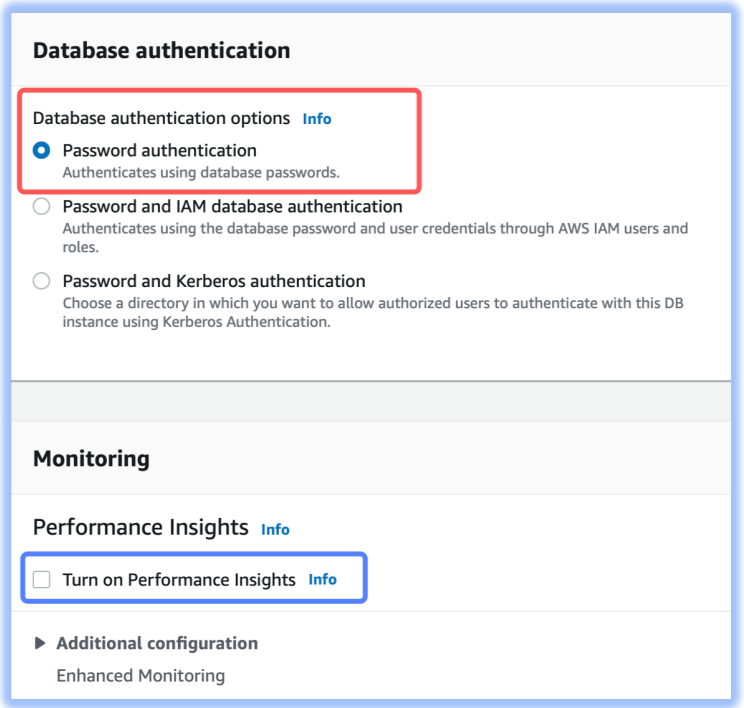


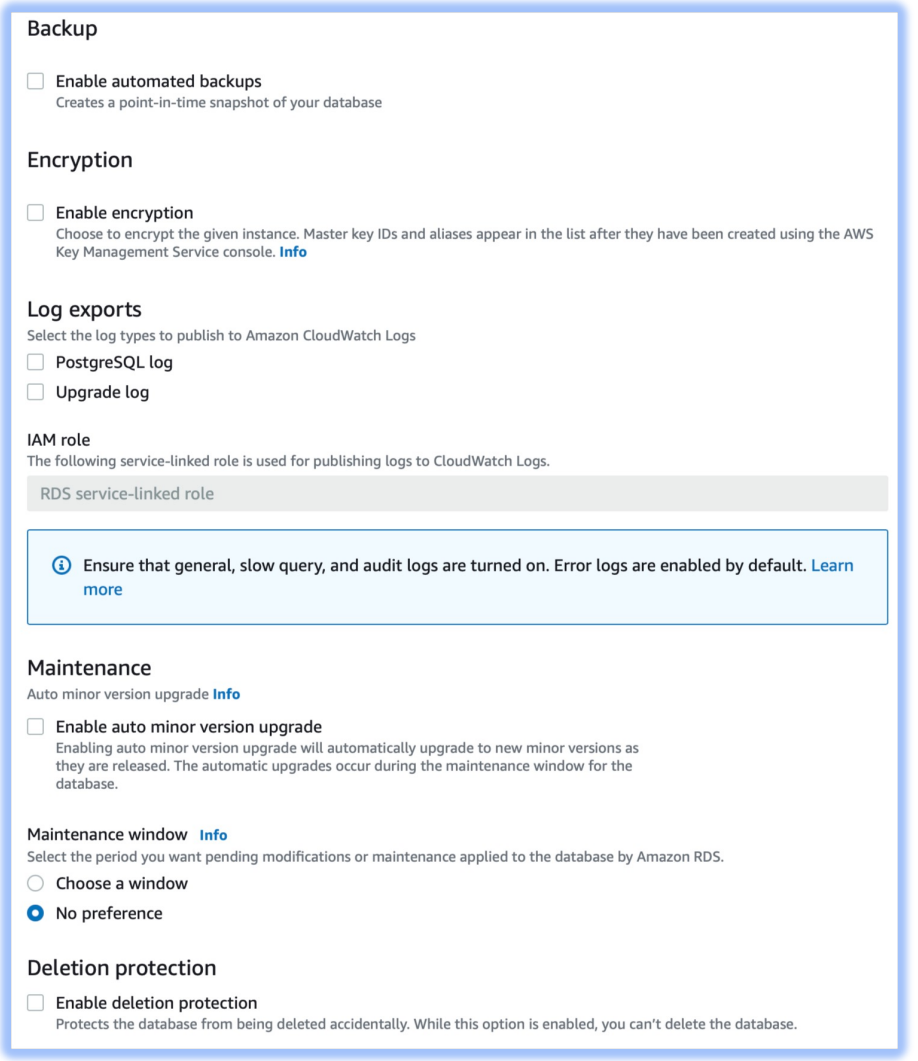
Wprowadzane dane należy zapamiętać ponieważ ich wartości będą wykorzystywane w dalszych krokach wdrożenia – w “produkcyjnej” wersji docker-compose.yml.

PGUSER=postgres
PGPASSWORD=postgres

PFSwChO – Laboratorium11

Usługa wielo-kontenerowa – wdrożenie na AWS (10)
- usługa RDS (postgres) (6)

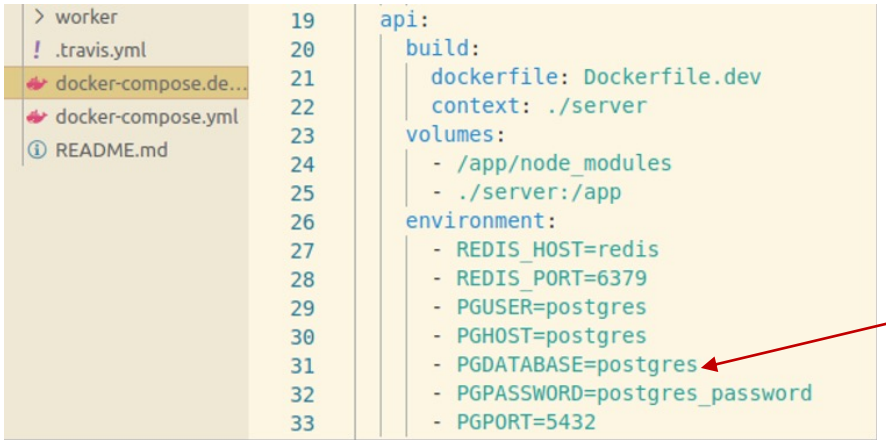


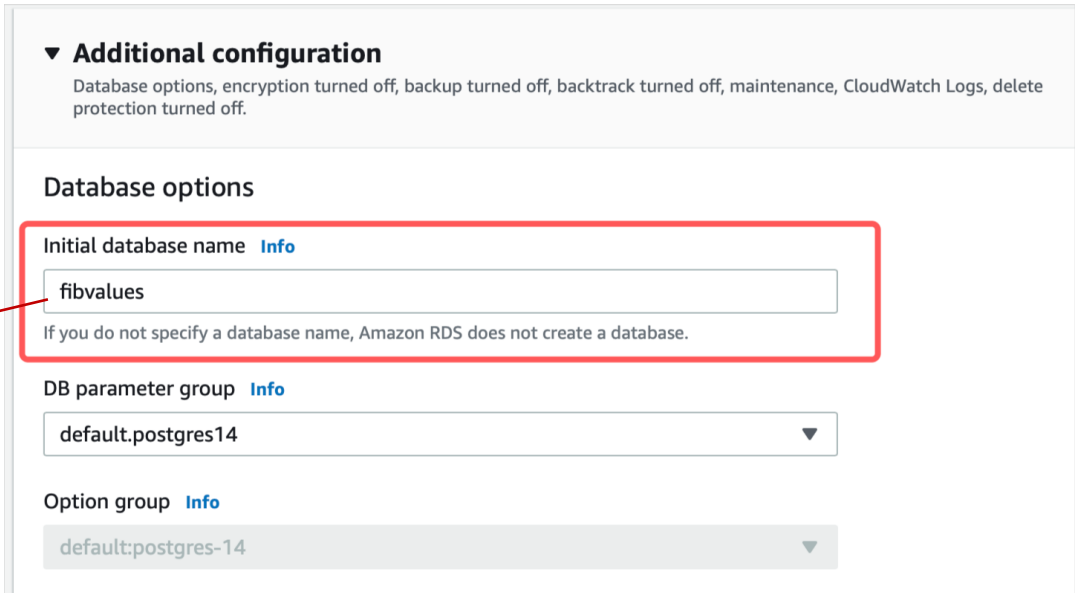


Proszę „odznaczyć wszystkie nieistotne opcje dla realizacji przykładu

PFSwChO – Laboratorium11

Usługa wielo-kontenerowa – wdrożenie na AWS (11)
- usługa RDS (postgres) (7)

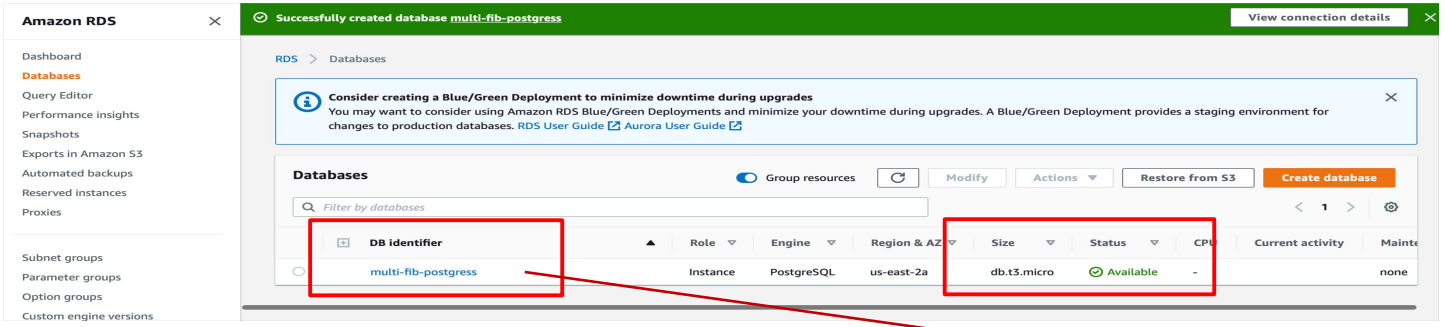


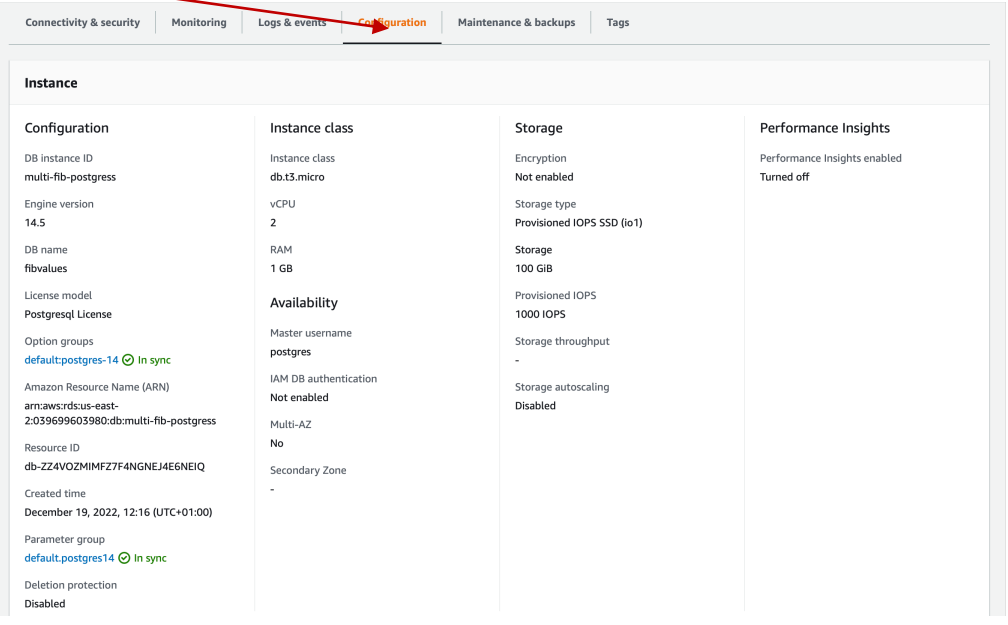


PGHOST=multi-fib-postgres
PGUSER=postgres
PGPASSWORD=postgres
PGDATABASE=fibvalues
PGPORT=5432

PFSwChO – Laboratorium11

Usługa wielo-kontenerowa – wdrożenie na AWS (12) - usługa RDS (postgres) (8)

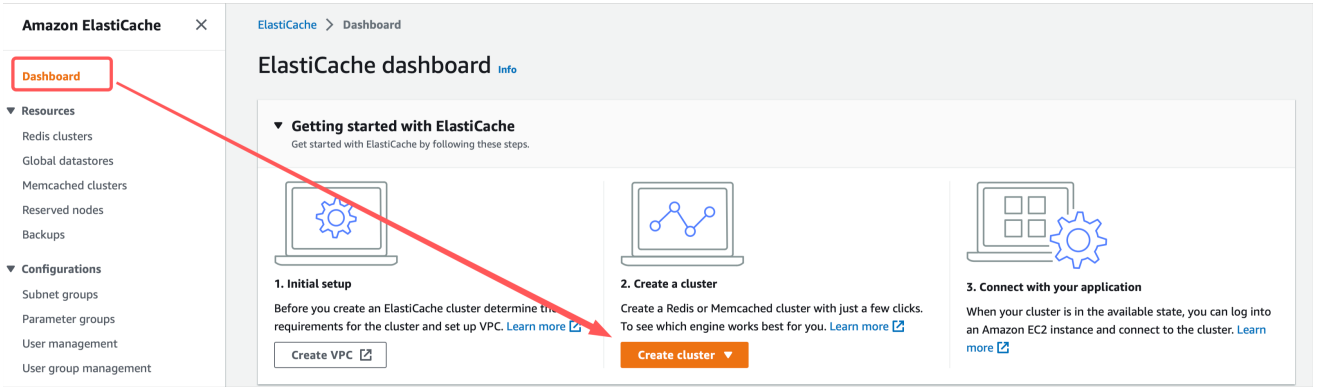


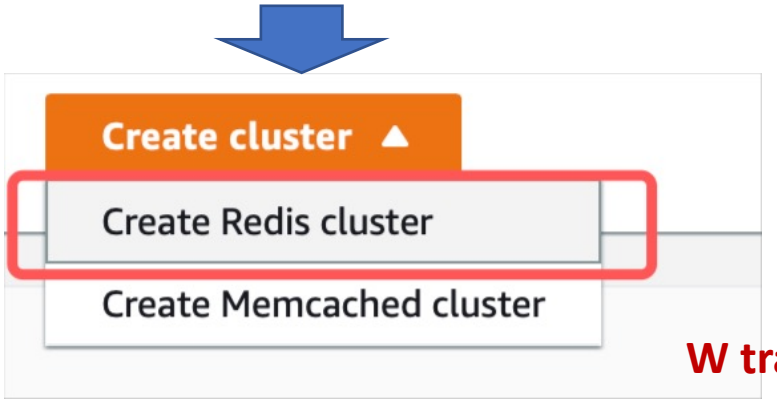


Należy koniecznie sprawdzić konfigurację utworzonej bazy i w razie potrzeby wprowadzić korekty lub utworzyć bazę ponownie

PFSwChO – Laboratorium11

Usługa wielo-kontenerowa – wdrożenie na AWS (13)
- usługa Elastic Cache (redis) (1)



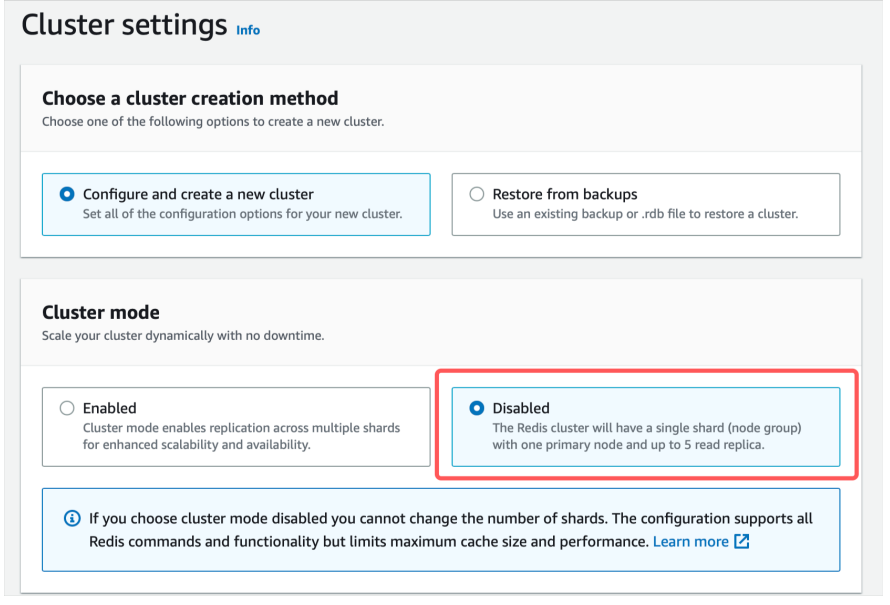


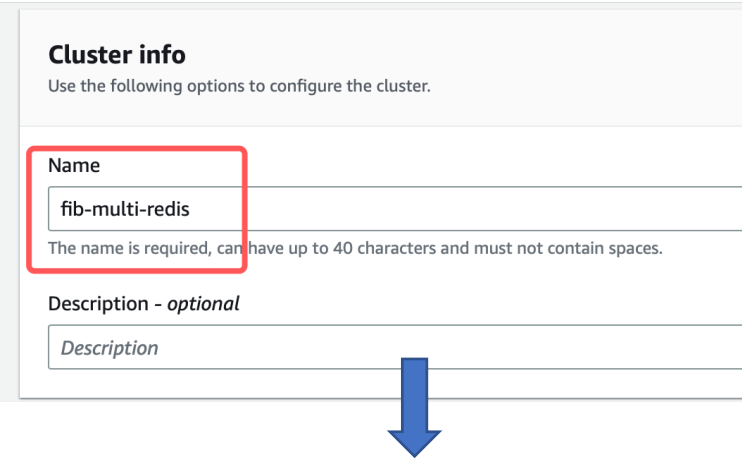
Opcja budowy klastra ma duży wpływ na koszty wykorzystania z usługi Elastic Cache – należy dokładnie przeanalizować cennik I wybrać najtańszą, wystarczającą konfiguracje

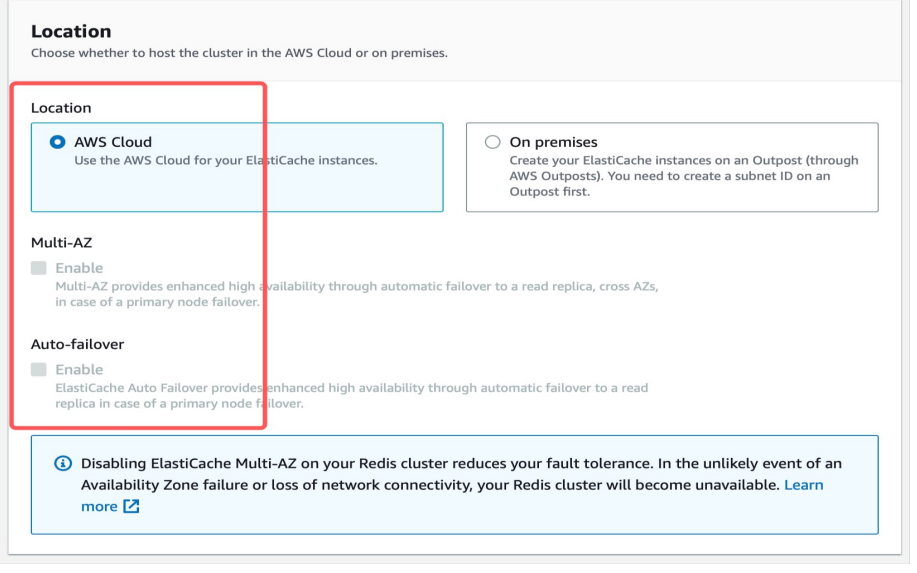
W trakcie lab wybieramy jeden węzeł !!!!

PFSwChO – Laboratorium11

Usługa wielo-kontenerowa – wdrożenie na AWS (14)
- usługa Elastic Cache (redis) (2)







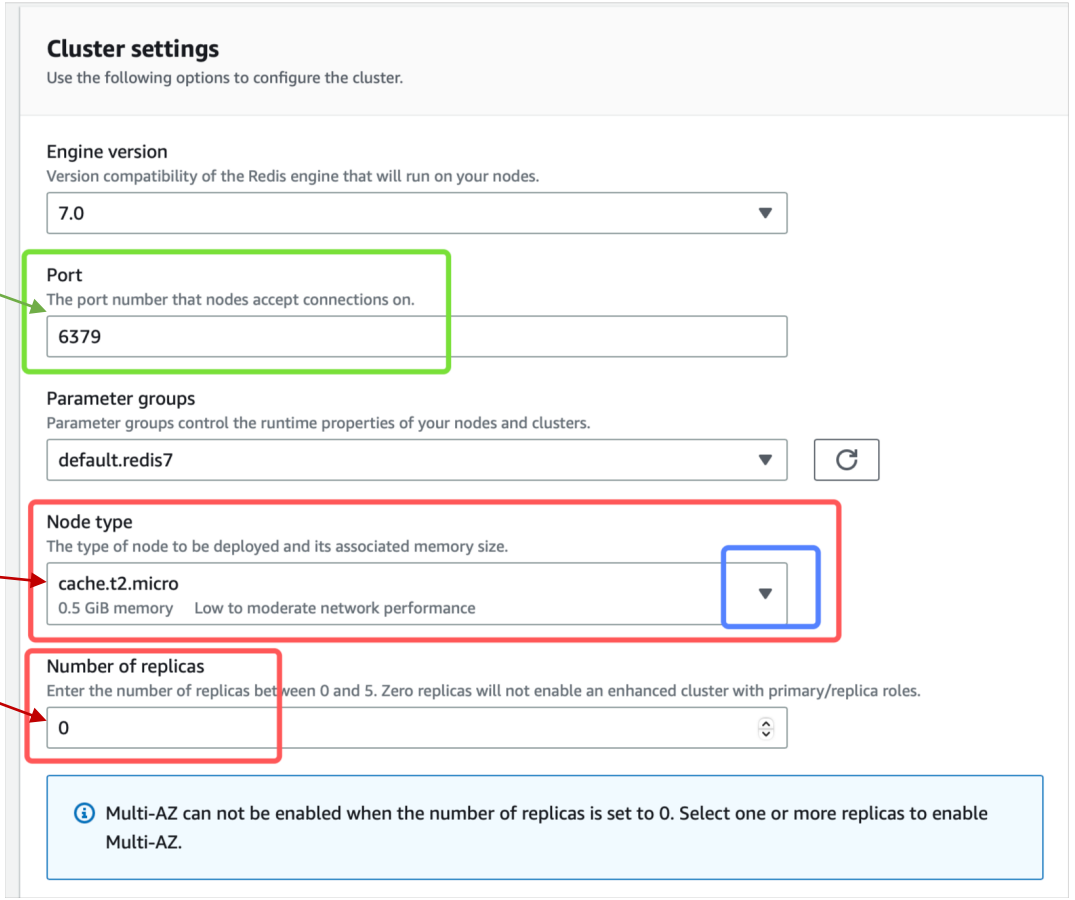
PFSwChO – Laboratorium11

Usługa wielo-kontenerowa – wdrożenie na AWS (15)
- usługa Elastic Cache (redis) (3)

Port należy zapamiętać, ponieważ będzie od wykorzystywany w pliku produkcyjnym docker-compose

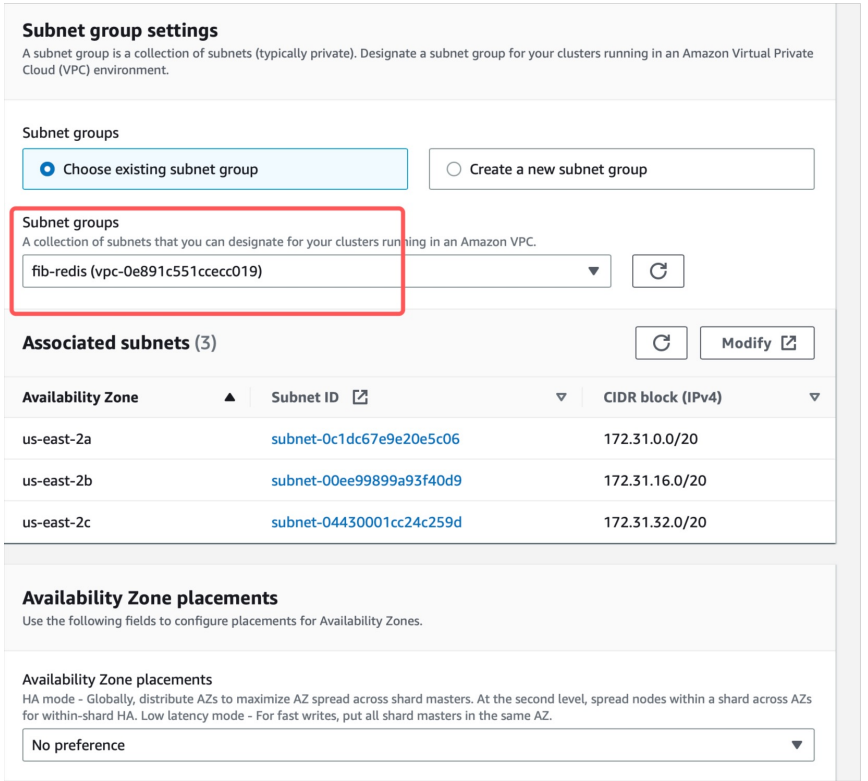
Proponowane domyślnie ustawienia Redis wiążą się z wysokimi kosztami

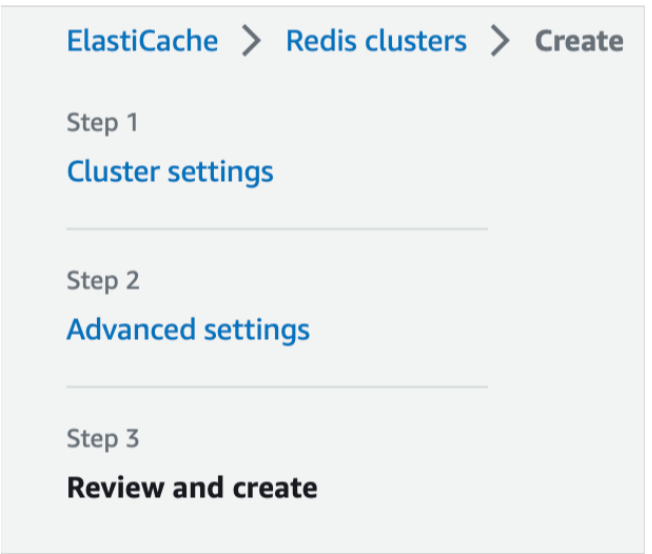
KONIECZNIE zmienić !!!!!



PFSwChO – Laboratorium11

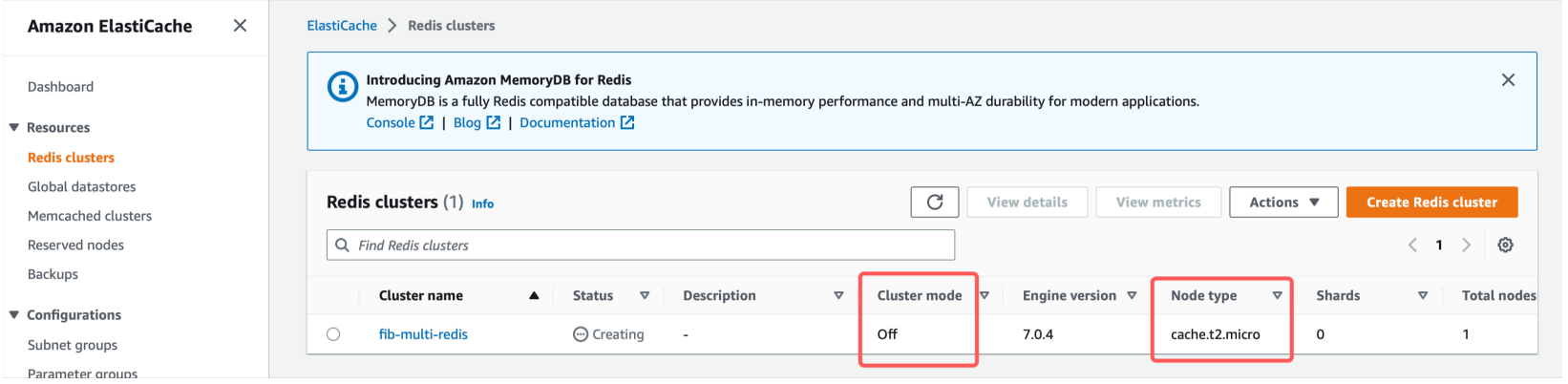
Usługa wielo-kontenerowa – wdrożenie na AWS (16)
- usługa Elastic Cache (redis) (4)





PFSwChO – Laboratorium11

Usługa wielo-kontenerowa – wdrożenie na AWS (16)
- usługa Elastic Cache (redis) (4)



Przed kolejnym laboratorium należy (w miarę możliwości) utworzyć elementy środowiska produkcyjnego na chmurze AWS (usługi: EBS, IAM, VPC, RDS, EC). Parametry znajdują się w początkowej części kolejnej instrukcji laboratoryjnej.

Na koniec laboratorium PROSZĘ USUNĄĆ skonfigurowane bazy (RDS, EC) oraz środowisko EBS