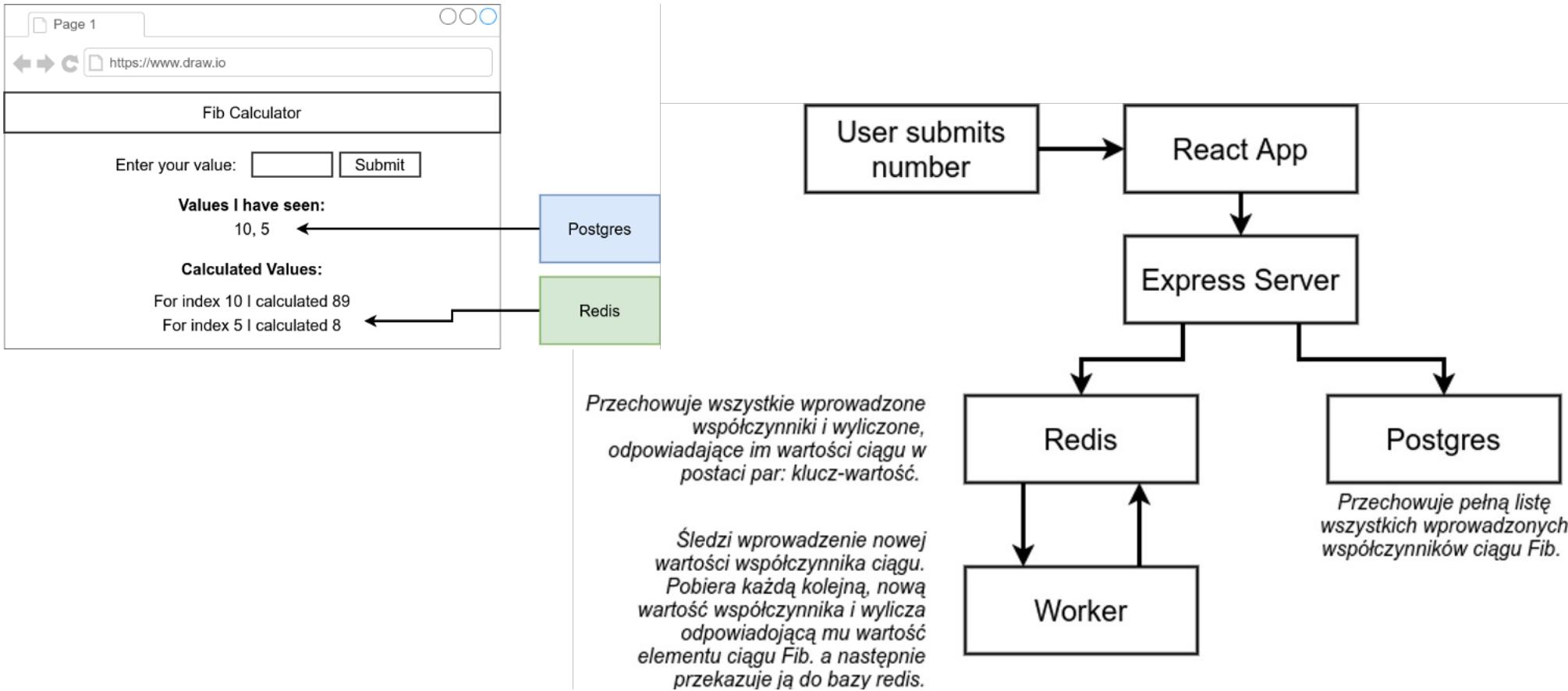


# PFSwChO – Laboratorium11

## Recap z poprzedniego laboratorium

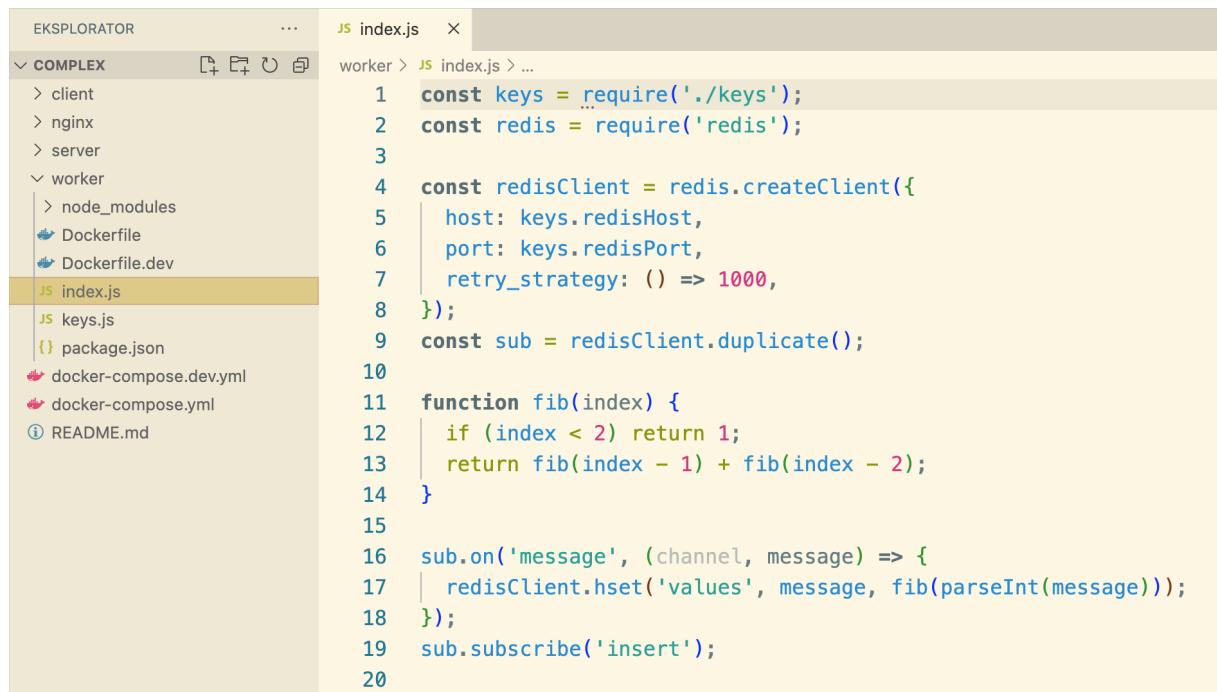


# PFSwChO – Laboratorium11

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 utworzyć repozytorium na GitHub oraz zsynchronizować je z katalogiem *complex*.



The screenshot shows a code editor with a file tree on the left and a code editor on the right.

**File Tree:**

- COMPLEX
- client
- nginx
- server
- worker
  - node\_modules
  - Dockerfile
  - Dockerfile.dev
  - index.js** (highlighted)
  - keys.js
  - package.json
  - docker-compose.dev.yml
  - docker-compose.yml
  - README.md

**Code Editor (index.js):**

```
1 const keys = require('./keys');
2 const redis = require('redis');
3
4 const redisClient = redis.createClient({
5   host: keys.redisHost,
6   port: keys.redisPort,
7   retry_strategy: () => 1000,
8 });
9 const sub = redisClient.duplicate();
10
11 function fib(index) {
12   if (index < 2) return 1;
13   return fib(index - 1) + fib(index - 2);
14 }
15
16 sub.on('message', (channel, message) => {
17   redisClient.hset('values', message, fib(parseInt(message)));
18 });
19 sub.subscribe('insert');
```

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

# PFSwChO – Laboratorium11

## Przykładowa usługa wielo-kontenerowa – UI routing (1)

File Edit Selection View Go Run Terminal Help

EXPLORER JS Fib.js

```
client > src > JS Fib.js > ...
1 import React, { Component } from 'react';
2 import axios from 'axios';
3
4 class Fib extends Component {
5   state = {
6     seenIndexes: [],
7     values: {},
8     index: '',
9   };
10
11   componentDidMount() {
12     this.fetchValues();
13     this.fetchIndexes();
14   }
15
16   async fetchValues() {
17     const values = await axios.get('/api/values/current');
18     this.setState({ values: values.data });
19   }
20
21   async fetchIndexes() {
22     const seenIndexes = await axios.get('/api/values/all');
23     this.setState({
24       seenIndexes: seenIndexes.data,
25     });
26   }
27 }
```

File Edit Selection View Go Run Terminal Help

EXPLORER JS OtherPage.js

```
client > src > JS OtherPage.js > ...
1 import React from 'react';
2 import { Link } from 'react-router-dom';
3
4 export default () => {
5   return (
6     <div>
7       I'm some other page!
8       <Link to="/">Go back home</Link>
9     </div>
10   );
11 }
12 }
```

The image displays two side-by-side screenshots of a web browser window. Both screenshots show a 'Fib Calculator' application running on a local server at <https://www.draw.io>.  
Left Screenshot (Calculator Page): The title bar says 'Page 1'. The main content area has a form with 'Enter your index:' and an input field containing '7'. Below the form, it says 'Indices I have seen:' followed by '10, 5, 7'. At the bottom, it says 'Calculated Values:' followed by three lines: 'For index 7 I calculated 21', 'For index 10 I calculated 89', and 'For index 5 I calculated 8'.  
Right Screenshot (Other Page): The title bar says 'Page 1'. The main content area displays the text 'I'm some other page!'.

# PFSwChO – Laboratorium11

## Przykładowa usługa wielo-kontenerowa – UI routing (2)

The screenshot shows the VS Code interface with the Explorer view open. The project structure is as follows:

- client**:
  - nginx
  - node\_modules
  - public
  - src
    - .gitignore
    - Dockerfile
    - Dockerfile.dev
    - package.json**
  - README.md
  - server
  - worker
  - docker-compose.yml
  - Dockerrun.aws.json

The **package.json** file is selected in the Explorer. A red box highlights the following code block:

```
    "name": "client",
    "version": "0.1.0",
    "private": true,
    "dependencies": {
      "@testing-library/jest-dom": "^4.2.4",
      "@testing-library/react": "^9.5.0",
      "@testing-library/user-event": "^7.2.1",
      "react": "^16.13.1",
      "react-dom": "^16.13.1",
      "react-scripts": "3.4.3",
      "react-router-dom": "4.3.1",
      "axios": "0.18.0"
    },
    "scripts": {
      "start": "react-scripts start",
      "build": "react-scripts build",
      "test": "react-scripts test",
      "eject": "react-scripts eject"
    },
```

The screenshot shows the VS Code interface with the Explorer view open. The project structure is as follows:

- client**:
  - nginx
  - node\_modules
  - public
  - src
    - # App.css
    - App.js**
    - App.test.js
    - Fib.js
    - # index.css
    - index.js
    - logo.svg
    - OtherPage.js
    - serviceWorker.js
    - setupTests.js
    - .gitignore
    - Dockerfile
    - Dockerfile.dev
    - package.json
    - README.md
    - server
    - worker
    - docker-compose.yml
    - Dockerrun.aws.json

The **App.js** file is selected in the Explorer. A red box highlights the following code block:

```
import React from 'react';
import logo from './logo.svg';
import './App.css';
import { BrowserRouter as Router, Route, Link } from 'react-router-dom';
import OtherPage from './OtherPage';
import Fib from './Fib';

function App() {
  return (
    <Router>
      <div className="App">
        <header className="App-header">
          <img src={logo} className="App-logo" alt="logo" />
          <a
            className="App-link"
            href="https://reactjs.org"
            target="_blank"
            rel="noopener noreferrer"
          >
            Learn React 3
          </a>
          <Link to="/">Home</Link>
          <Link to="/otherpage">Other Page</Link>
        </header>
        <div>
          <Route exact path="/" component={Fib} />
          <Route path="/otherpage" component={OtherPage} />
        </div>
      </div>
    </Router>
  );
}

export default App;
```

A green box highlights the following code block:

```
<Link to="/">Home</Link>
<Link to="/otherpage">Other Page</Link>
```

A blue box highlights the following code block:

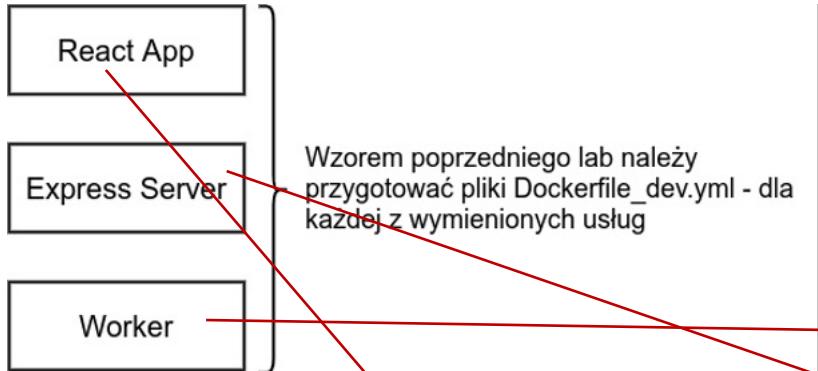
```
<Route exact path="/" component={Fib} />
<Route path="/otherpage" component={OtherPage} />
```

**React-app (server)**

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

# PFSwChO – Laboratorium11

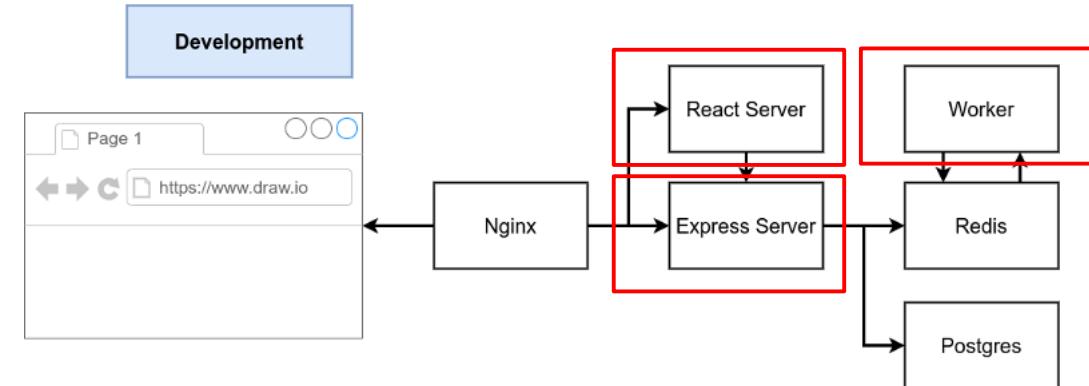
Przykładowa usługa wielo-kontenerowa  
– faza rozwojowa (1) – Dockerfile.dev



**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**



**Dockerfile.dev**

```
server > Dockerfile.dev > ...
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**

**Dockerfile.dev**

```
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**

**package.json**

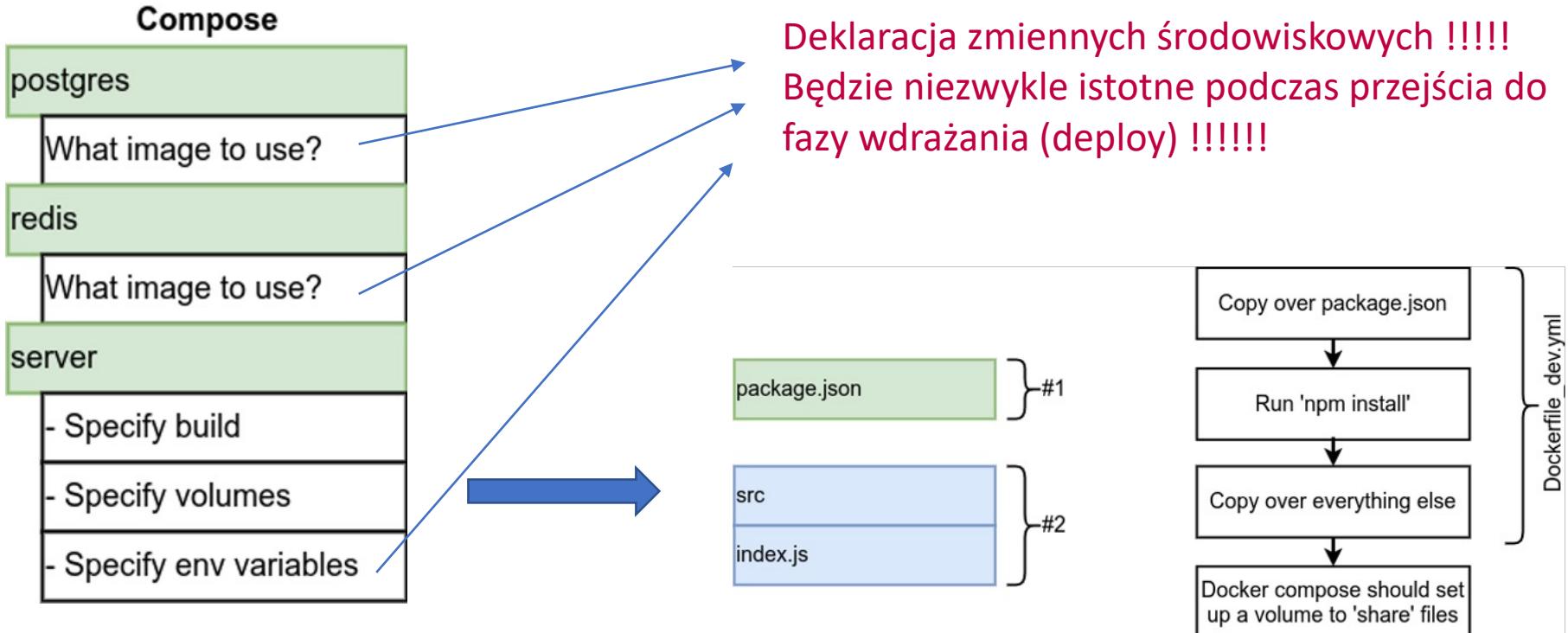
```
worker > package.json > ...
1 {
2   "dependencies": {
3     "nodemon": "1.18.3",
4     "redis": "2.8.0"
5   },
6   "scripts": {
7     "start": "node index.js",
8     "dev": "nodemon"
9   }
10 }
```

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

# PFSwChO – Laboratorium11

Przykładowa usługa wielo-kontenerowa

– faza rozwojowa (2) – docker compose (1)



# PFSwChO – Laboratorium11

## Przykładowa usługa wielo-kontenerowa

– faza rozwojowa (3) – docker compose (2) – podstawowe deklaracje dla: postgres i redis

```
docker-compose.yml
1 version: '3'
2 services:
3   postgres:
4     image: 'postgres:latest'
5     environment:
6       - POSTGRES_PASSWORD=postgres_password
```

### 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/_/postgres)

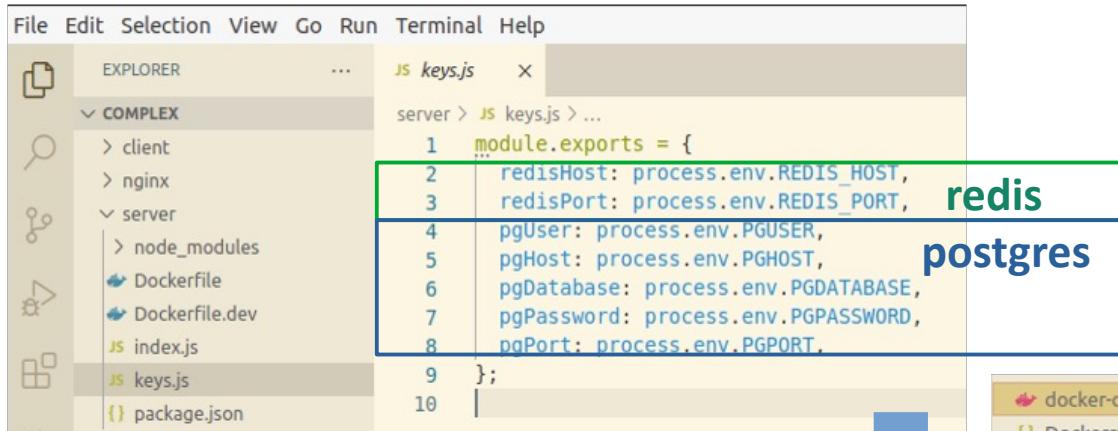
```
docker-compose.yml
1 version: '3'
2 services:
3   postgres:
4     image: 'postgres:latest'
5     environment:
6       - POSTGRES_PASSWORD=postgres_password
7   redis:
8     image: 'redis:latest'
```

[https://hub.docker.com/\\_/redis](https://hub.docker.com/_/redis)

Zawsze należy zapoznać się z AKTUALNA 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

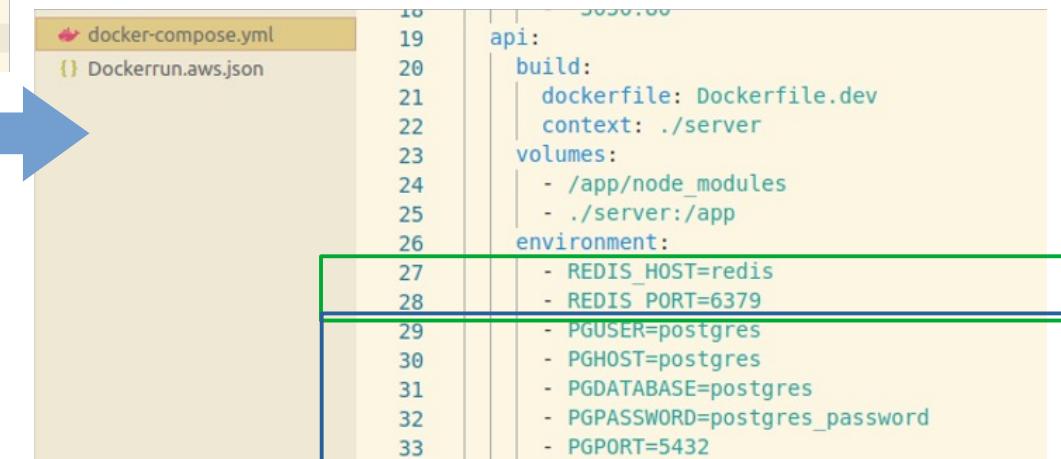


```
File Edit Selection View Go Run Terminal Help
EXPLORER      ...
COMPLEX
> client
> nginx
server
> node_modules
Dockerfile
Dockerfile.dev
index.js
keys.js
package.json

JS keys.js  x
server > JS keys.js > ...
1 module.exports = {
2   redisHost: process.env.REDIS_HOST,
3   redisPort: process.env.REDIS_PORT,    redis
4   pgUser: process.env.PGUSER,           postgres
5   pgHost: process.env.PGHOST,
6   pgDatabase: process.env.PGDATABASE,
7   pgPassword: process.env.PGPASSWORD,
8   pgPort: process.env.PGPORT,
9 };
10 |
```

Compose

postgres	What image to use?
redis	What image to use?
server	- Specify build - Specify volumes - Specify env variables



```
version: '3.8'
services:
  api:
    build:
      dockerfile: Dockerfile.dev
      context: ./server
    volumes:
      - /app/node_modules
      - ./server:/app
    environment:
      - REDIS_HOST=redis
      - REDIS_PORT=6379
      - PGUSER=postgres
      - PGHOST=postgres
      - PGDATABASE=postgres
      - PGPASSWORD=postgres_password
      - PGPORT=5432
```

**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

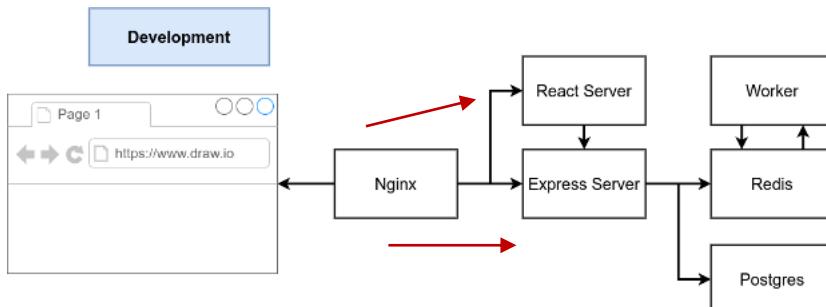
```
File Edit Selection View Go Run Terminal Help
EXPLORER ...
COMPLEX
    > client
    > nginx
    > server
    > worker
        > node_modules
        Dockerfile
        Dockerfile.dev
        index.js
        keys.js
        package.json
        docker-compose.yml
        Dockerrun.aws.json
JS keys.js  x
worker > JS keys.js > ...
1 module.exports = {
2   ...
3     redisHost: process.env.REDIS_HOST,
4     redisPort: process.env.REDIS_PORT,
5   };
dockercfg.yml
34  client:
35      stdin_open: true
36      build:
37          dockerfile: Dockerfile.dev
38          context: ./client
39          volumes:
40              - /app/node_modules
41              - ./client:/app
42  worker:
43      build:
44          dockerfile: Dockerfile.dev
45          context: ./worker
46          volumes:
47              - /app/node_modules
48              - ./worker:/app
49      environment:
50          - REDIS_HOST=redis
51          - REDIS_PORT=6379
52
```

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

# PFSwChO – Laboratorium11

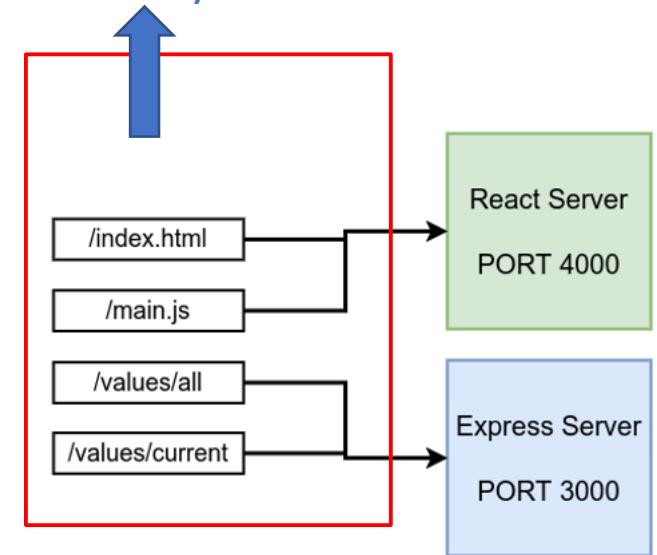
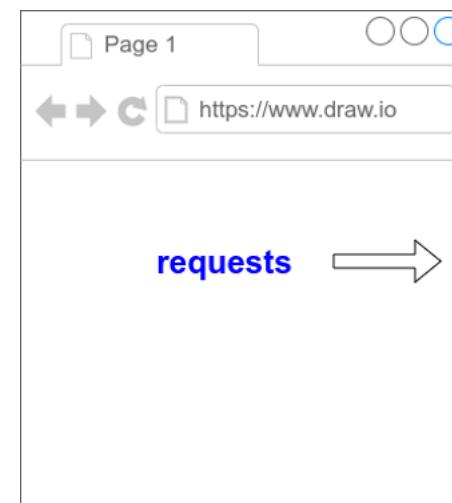
Przykładowa usługa wielo-kontenerowa

– faza rozwojowa (6) – docker compose (5) – powrót do tematu: UI routing (path routing)



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

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



# PFSwChO – Laboratorium 11

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

**React-app (server)**

```
EXPLORER ... JS index.js x
COMPLEX ⌂ ⌂ ⌂ ⌂
> client
> nginx
server
> node_modules
  Dockerfile
  Dockerfile.dev
JS index.js
JS keys.js
{} package.json
worker
> node_modules
  Dockerfile
  Dockerfile.dev
JS index.js
JS keys.js
{} package.json
docker-compose.yml
{} Dockerrun.aws.json

server > JS index.js > ...
30
31
32
33
34
35
36
37 // Express route handlers
38
39 app.get("/", (req, res) => {
40   res.send("Hi");
41 });
42
43 app.get("/values/all", async (req, res) => {
44   const values = await pgClient.query("SELECT * from values");
45
46   res.send(values.rows);
47 });
48
49 app.get("/values/current", async (req, res) => {
50   redisClient.hgetall("values", (err, values) => {
51     res.send(values);
52   });
53 });
54
55 app.post("/values", async (req, res) => {
56   const index = req.body.index;
57
58   if (parseInt(index) > 40) {
59     return res.status(422).send("Index too high");
60   }
61
62   redisClient.hset("values", index, "Nothing yet!");
63   redisPublisher.publish("insert", index);
64   pgClient.query("INSERT INTO values(number) VALUES($1)", [index]);
65
66   res.send({ working: true });
67 });
```



Slajd 4 + poniższe definicje

**Client – Fib.js**

```
EXPLORER ... JS Fib.js x
COMPLEX ⌂ ⌂ ⌂ ⌂
client
> nginx
> node_modules
> public
src
# App.css
JS App.js
JS App.test.js
JS Fib.js
# index.css
JS index.js
logo.svg
JS OtherPage.js
JS serviceWorker.js
JS setupTests.js
.gitignore
Dockerfile
Dockerfile.dev
{} package.json

client > src > JS Fib.js > ...
14 }
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35

async fetchValues() {
  const values = await axios.get('/api/values/current');
  this.setState({ values: values.data });
}

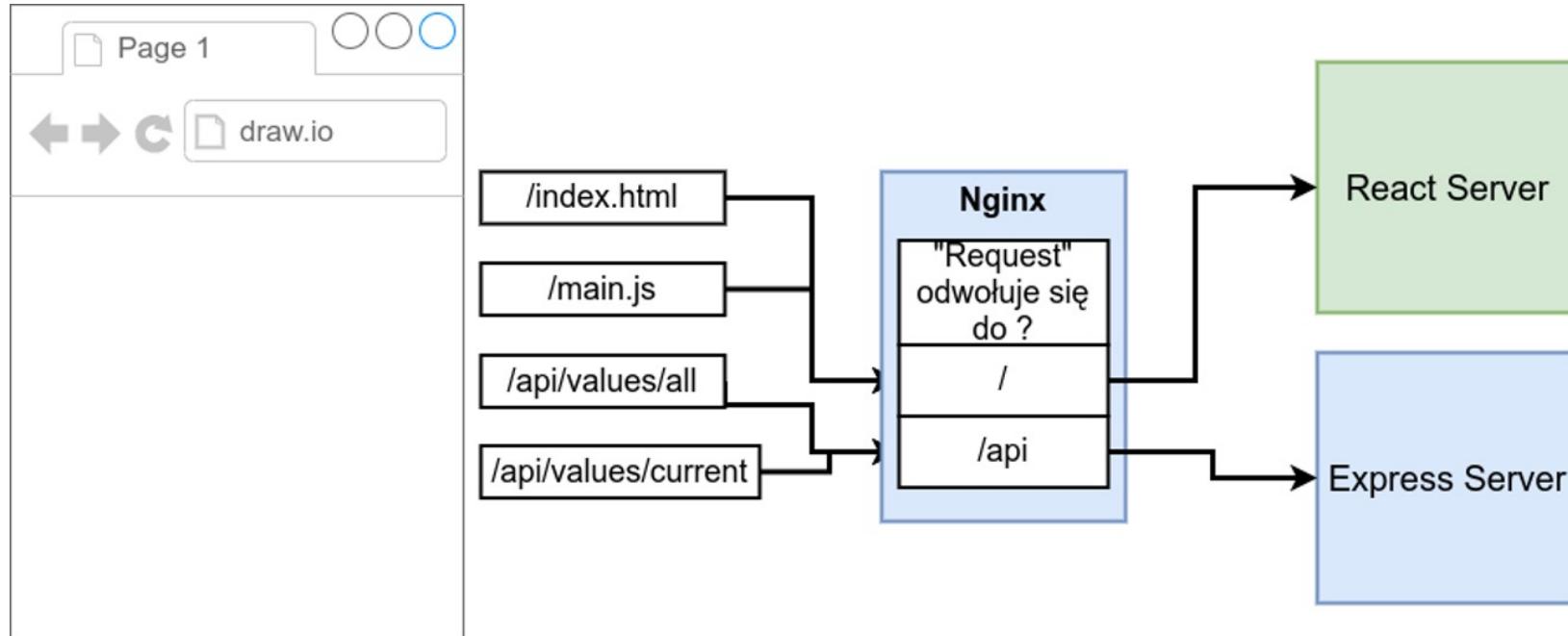
async fetchIndexes() {
  const seenIndexes = await axios.get('/api/values/all');
  this.setState({
    seenIndexes: seenIndexes.data,
  });
}

handleSubmit = async (event) => {
  event.preventDefault();

  await axios.post('/api/values', {
    index: this.state.index,
  });
  this.setState({ index: '' });
};
```

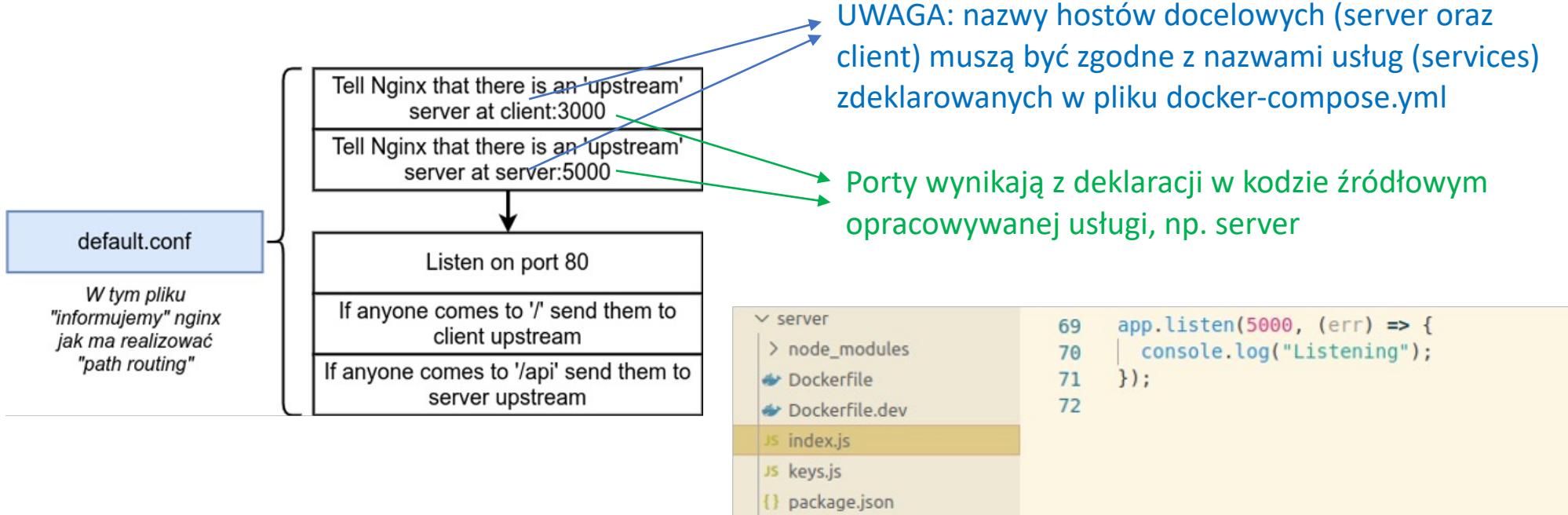
## PFSwChO – Laboratorium11

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)



# PFSwChO – Laboratorium 11

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

```
default.conf x
1 upstream client {
2   server client:3000;
3 }
4
5 upstream server {
6   server server:5000;|
```



!!! Pojawi się ERROR !!!



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

```
COMPLEX . . .
> client
< nginx
  default.conf
  Dockerfile
  Dockerfile.dev
< server
  > node_modules
  Dockerfile
  Dockerfile.dev
  index.js
  keys.js
  package.json
< worker
  > node_modules
  Dockerfile
  Dockerfile.dev
  index.js
  keys.js
  package.json
  docker-compose.yml
  Dockerrun.aws.json
```

nginx > default.conf

```
1 upstream client {
2   server client:3000;
3 }
4
5 upstream api { | server api:5000;
6
7
8
9 server {
10   listen 80;
11
12   location / {
13     proxy_pass http://client;
14   }
15
16   location /sockjs-node {
17     proxy_pass http://client;
18     proxy_http_version 1.1;
19     proxy_set_header Upgrade $http_upgrade;
20     proxy_set_header Connection "Upgrade";
21   }
22
23   location /api {
24     rewrite /api/(.*) /$1 break;
25     proxy_pass http://api;
26   }
27 }
```

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)

The screenshot shows the Docker IDE interface. On the left, the Explorer sidebar lists projects: 'COMPLEX' (client, nginx, server), 'worker' (node\_modules, Dockerfile, Dockerfile.dev, index.js, keys.js, package.json), and 'docker-compose.dev'. The 'Dockerfile.dev' tab is active, displaying:

```
FROM nginx
COPY ./default.conf /etc/nginx/conf.d/default.conf
```

A large blue arrow points from the Dockerfile.dev tab down to the docker-compose.dev.yml tab. The 'docker-compose.dev.yml' tab is also active, displaying:

```
version: '3'
services:
  postgres:
    image: 'postgres:latest'
    environment:
      - POSTGRES_PASSWORD=postgres_password
  redis:
    image: 'redis:latest'
  nginx:
    depends_on:
      - api
      - client
    restart: always
    build:
      dockerfile: Dockerfile.dev
      context: ./nginx
    ports:
      - '3050:80'
```

A red box highlights the 'depends\_on' section of the nginx service, and a green box highlights the 'ports' section.

If you wish to adapt the default configuration, use something like the following to copy it from a running nginx container:

```
$ docker run --name tmp-nginx-container -d nginx
$ docker cp tmp-nginx-container:/etc/nginx/nginx.conf /host/path/nginx.conf
$ docker rm -f tmp-nginx-container
```

This can also be accomplished more cleanly using a simple Dockerfile (in /host/path/):

```
FROM nginx
COPY nginx.conf /etc/nginx/nginx.conf
```

[https://hub.docker.com/\\_/nginx](https://hub.docker.com/_/nginx)

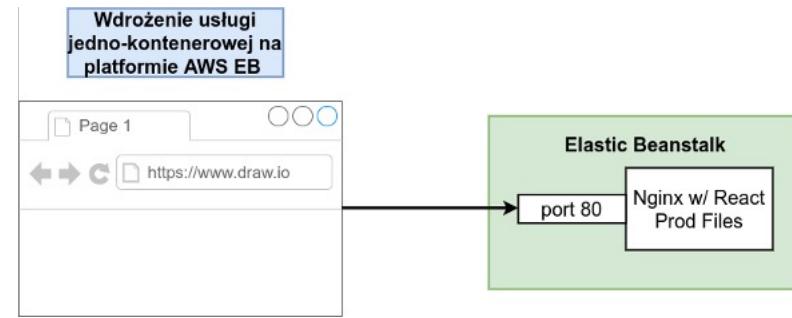
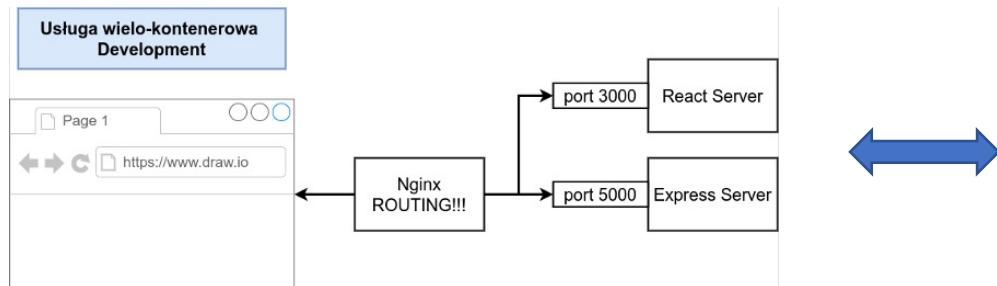
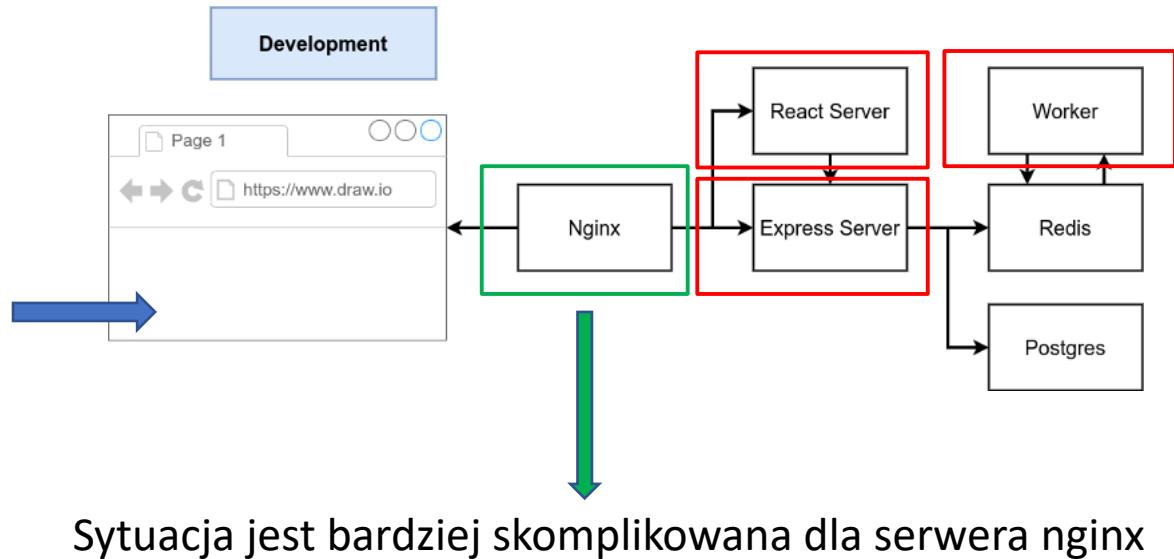
Należy uruchomić aplikację i zweryfikować poprawność jej działania

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

# PFSwChO – Laboratorium11

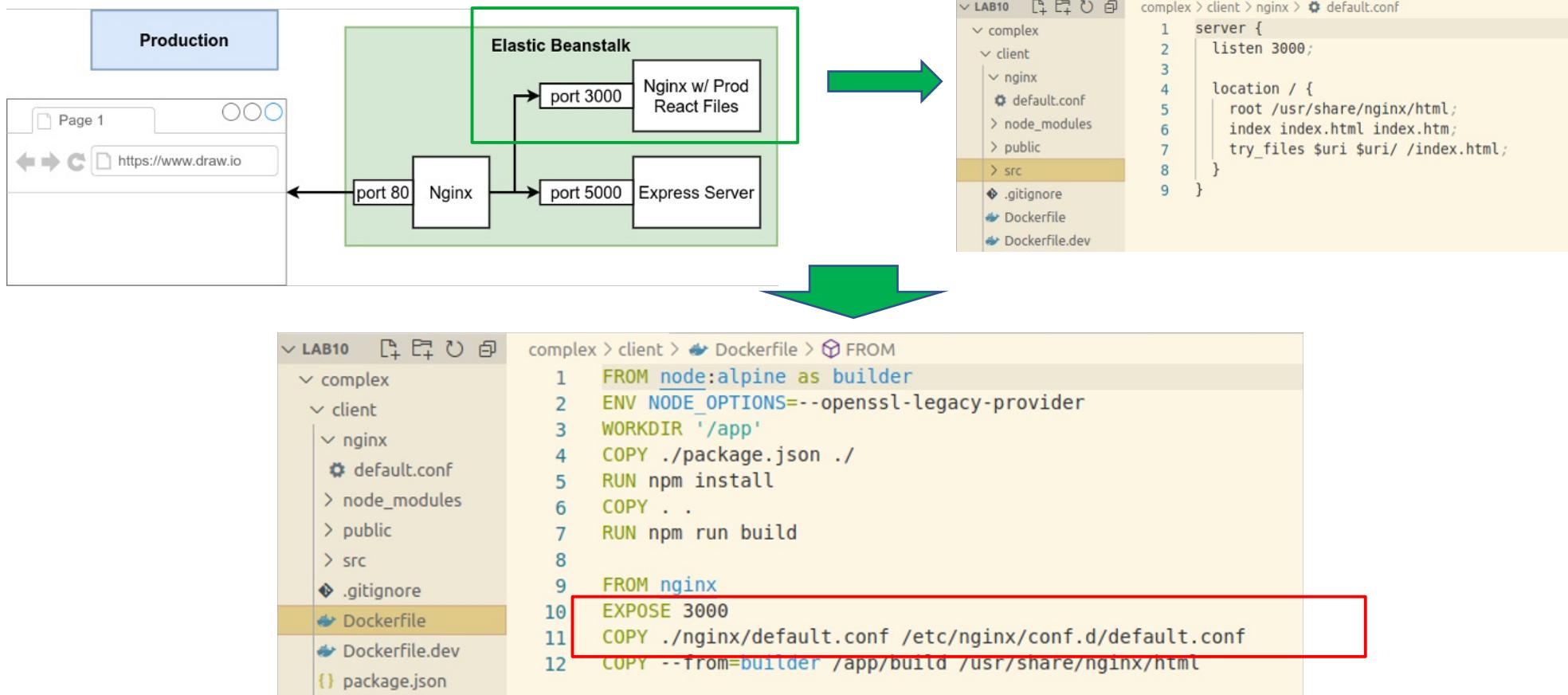
Usługa wielo-kontenerowa – tworzenie “produkcyjnych” Dockerfiles (1)

Pliki Dockerfile dla zaznaczonych czerwonym obramowaniem składowych usługi nie różnią się od wersji Dockerfile.dev (proszę prześledzić dostarczone kody).



# PFSwChO – Laboratorium11

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



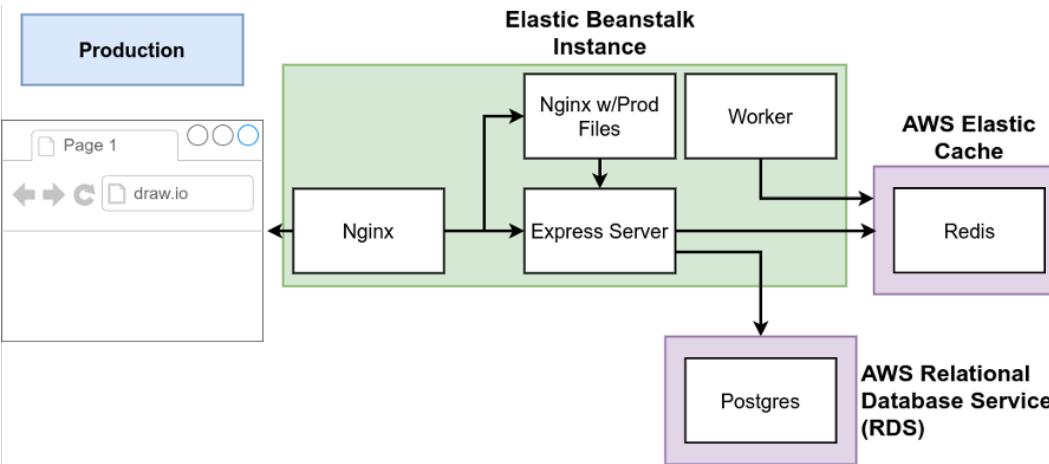
W usłudze EBS tworzymy nową aplikację (w dalszych przykładach: **Fib-multi**), analogicznie jak na poprzednim laboratorium

## PFSwChO – Laboratorium 11

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

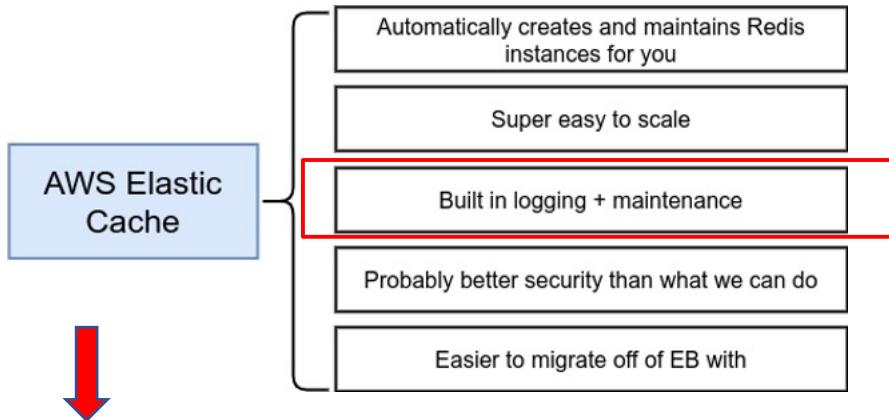


```
! .travis.yml docker-compose.yml docker-compose.dev.yml
complex > docker-compose.yml
1 version: "3"
2 services:
3   client:
4     image: "spg51/lab:fib-client"
5     mem_limit: 128m
6     hostname: client
7   server:
8     image: "spg51/lab:fib-server"
9     mem_limit: 128m
10    hostname: api
11    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"
```

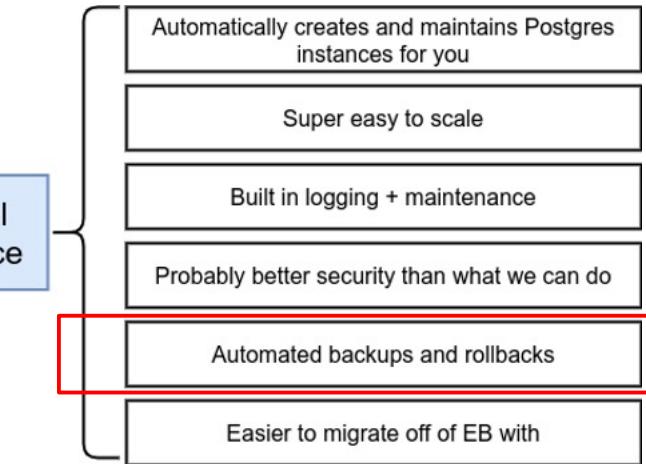
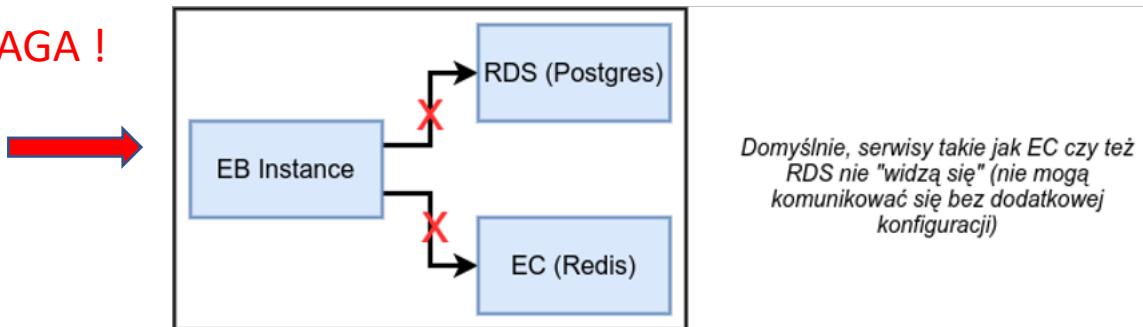
# PFSwChO – Laboratorium11

## 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



Default Virtual Private Cloud  
(VPC) for US-West-1



Jeden 'default'  
VPC na dany  
region

Default Virtual Private Cloud  
(VPC) for EU-West1

...brak składników...

Jeden 'default'  
VPC na dany  
region

Name	VPC ID	State	IPv4 CIDR	IPv6 CIDR	IPv6 pool
-	vpc-0e46986d0a6e42198	Available	172.31.0.0/16	-	-

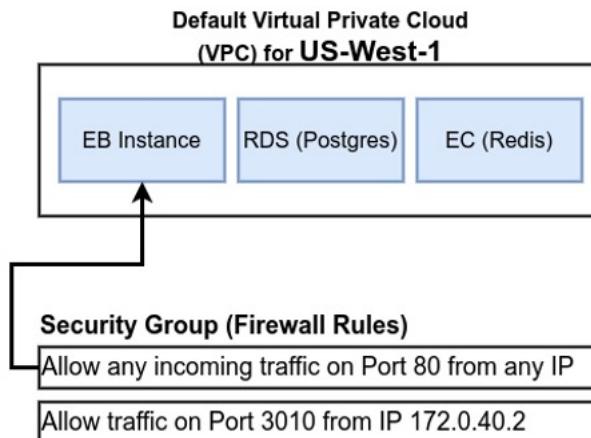
DHCP options set	Main route table	Main network ACL	Tenancy	Default VPC	Owner ID
dopt-03512cc6d311f1...	rtb-08485a7df95ee0f4a	acl-0cd4f889bdfe44702	Default	Yes	039699603980

# PFSwChO – Laboratorium11

## Usługa wielo-kontenerowa – wdrożenie na AWS (4)

### - pojęcie VPC (2) – Security Groups (1)

SG - ustawiane  
automatycznie podczas  
tworzenia środowiska



Security Groups (1/3) [Info](#)

Name	Security group ID	Security group name	VPC ID	Description	Owner	Inbound rules
Fibmulti-env	sg-0264f0a70b10061b6	awseb-e-5uvvgvyhinq...	vpc-0e46906d0a6e42190	Elastic Beanstalk creat...	039699603980	1 Permission ent
-	sg-050eda9130e2adacea	default	vpc-0e46986d0a6e42198	default VPC security gr...	039699603980	1 Permission ent
<input checked="" type="checkbox"/> Fibmulti-env	sg-0f4de8e510d2ed0d0	awseb-e-5uvvgvyhinq...	vpc-0e46986d0a6e42198	SecurityGroup for Elas...	039699603980	1 Permission ent

You can now check network connectivity with Reachability Analyzer [Run Reachability Analyzer](#)

Inbound rules (1/1)

Name	Security group rule...	IP version	Type	Protocol	Port range	Source	
<input checked="" type="checkbox"/>	-	sgr-07496d7b5e6160...	-	HTTP	TCP	80	sg-0264f0a70b10061b6

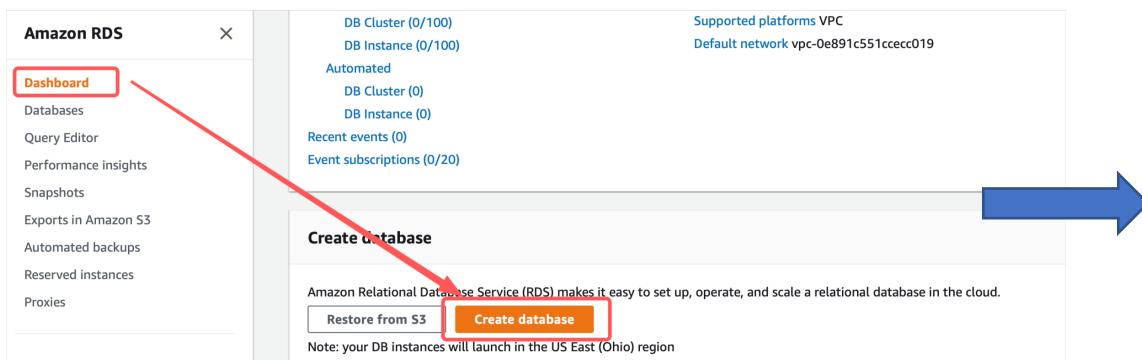
Należy zdefiniować nową, własną “security group”, która pozwoli na komunikację składowych usługi w obrębie danego VPC – ale to wymaga wcześniejszego zdefiniowania instacji baz danych (redis i postgres)

# PFSwChO – Laboratorium11

## Usługa wielo-kontenerowa – wdrożenie na AWS (5)

- usługa RDS (postgres) (1)

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



Amazon RDS Dashboard

DB Cluster (0/100)  
DB Instance (0/100)

Automated  
DB Cluster (0)  
DB Instance (0)

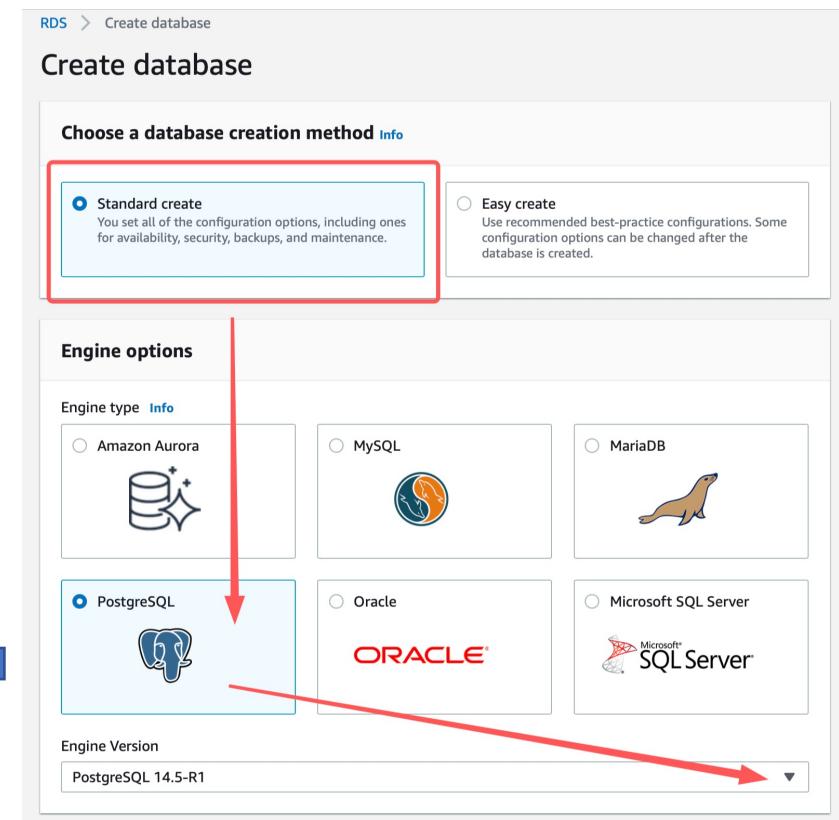
Recent events (0)  
Event subscriptions (0/20)

Create database

Amazon Relational Database Service (RDS) makes it easy to set up, operate, and scale a relational database in the cloud.

Restore from S3   Create database

Note: your DB instances will launch in the US East (Ohio) region



RDS > Create database

### Create database

Choose a database creation method [Info](#)

Standard create  
You set all of the configuration options, including ones for availability, security, backups, and maintenance.

Easy create  
Use recommended best-practice configurations. Some configuration options can be changed after the database is created.

Engine options

Engine type [Info](#)

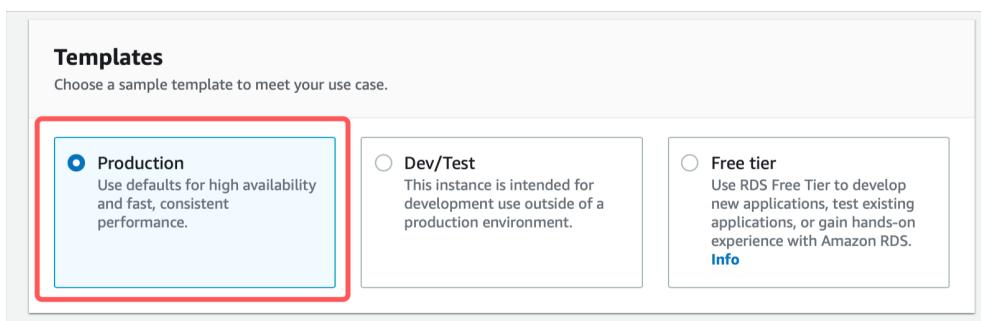
Amazon Aurora

MySQL

MariaDB

PostgreSQL

Engine Version  
PostgreSQL 14.5-R1



Templates

Choose a sample template to meet your use case.

Production  
Use defaults for high availability and fast, consistent performance.

Dev/Test  
This instance is intended for development use outside of a production environment.

Free tier  
Use RDS Free Tier to develop new applications, test existing applications, or gain hands-on experience with Amazon RDS.  
[Info](#)

# PFSwChO – Laboratorium11

## Usługa wielo-kontenerowa – wdrożenie na AWS (6)

### - usługa RDS (postgres) (2)

**Availability and durability**

Deployment options [Info](#)  
The deployment options below are limited to those supported by the engine you selected above.

- Multi-AZ DB Cluster - new  
Creates a DB cluster with a primary DB instance and two readable standby DB instances, with each DB instance in a different Availability Zone (AZ). Provides high availability, data redundancy and increases capacity to serve read workloads.
- Multi-AZ DB instance  
Creates a primary DB instance and a standby DB instance in a different AZ. Provides high availability and data redundancy, but the standby DB instance doesn't support connections for read workloads.
- Single DB instance  
Creates a single DB instance with no standby DB instances.



**Instance configuration**

The DB instance configuration options below are limited to those supported by the engine that you selected above.

**WAŻNE !!!!!**

DB instance class [Info](#)

- Standard classes (includes m classes)
- Memory optimized classes (includes r and x classes)
- Burstable classes (includes t classes)

db.t3.micro  
2 vCPUs 1 GiB RAM Network: 2085 Mbps



**Storage**

Storage type [Info](#)  
Provisioned IOPS SSD (io1)  
Flexibility in provisioning I/O

Allocated storage  
100 GiB

The minimum value is 100 GiB and the maximum value is 6144 GiB

Provisioned IOPS [Info](#)  
1000 IOPS

The minimum value is 1000 IOPS and the maximum value is 256 000 IOPS. The storage size ratio must be between 0.1 and 1000. The storage size ratio is the ratio between allocated storage and the Provisioned IOPS.

Your actual IOPS might vary from the amount that you provisioned based on your database workload and instance type. [Learn more](#)

**Storage autoscaling** [Info](#)  
Provides dynamic scaling support for your database's storage based on your application's needs.

Enable storage autoscaling  
Enabling this feature will allow the storage to increase after the specified threshold is exceeded.

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

# PFSwChO – Laboratorium11

## Usługa wielo-kontenerowa – wdrożenie na AWS (7)

### - usługa RDS (postgres) (3)

**Connectivity** [Info](#) [C](#)

**Compute resource**  
Choose whether to set up a connection to a compute resource for this database. Setting up a connection will automatically change connectivity settings so that the compute resource can connect to this database.

**Don't connect to an EC2 compute resource**  
Don't set up a connection to a compute resource for this database. You can manually set up a connection to a compute resource later.

**Connect to an EC2 compute resource**  
Set up a connection to an EC2 compute resource for this database.

**Network type** [Info](#)  
To use dual-stack mode, make sure that you associate an IPv6 CIDR block with a subnet in the VPC you specify.

**IPv4**  
Your resources can communicate only over the IPv4 addressing protocol.

**Dual-stack mode**  
Your resources can communicate over IPv4, IPv6, or both.

**Virtual private cloud (VPC)** [Info](#)  
Choose the VPC. The VPC defines the virtual networking environment for this DB instance.

Default VPC (vpc-0e891c551ccecc019) [▼](#)

Only VPCs with a corresponding DB subnet group are listed.

**After a database is created, you can't change its VPC.**

**DB Subnet group** [Info](#)  
Choose the DB subnet group. The DB subnet group defines which subnets and IP ranges the DB instance can use in the VPC that you selected.

default [▼](#)

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

Public access [Info](#)

Yes  
RDS assigns a public IP address to the database. Amazon EC2 instances and other resources outside of the VPC can connect to your database. Resources inside the VPC can also connect to the database. Choose one or more VPC security groups that specify which resources can connect to the database.

No  
RDS doesn't assign a public IP address to the database. Only Amazon EC2 instances and other resources inside the VPC can connect to your database. Choose one or more VPC security groups that specify which resources can connect to the database.

VPC security group (firewall) [Info](#)

Choose one or more VPC security groups to allow access to your database. Make sure that the security group rules allow the appropriate incoming traffic.

Choose existing  
Choose existing VPC security groups

Create new  
Create new VPC security group

Existing VPC security groups

Choose one or more options

default X

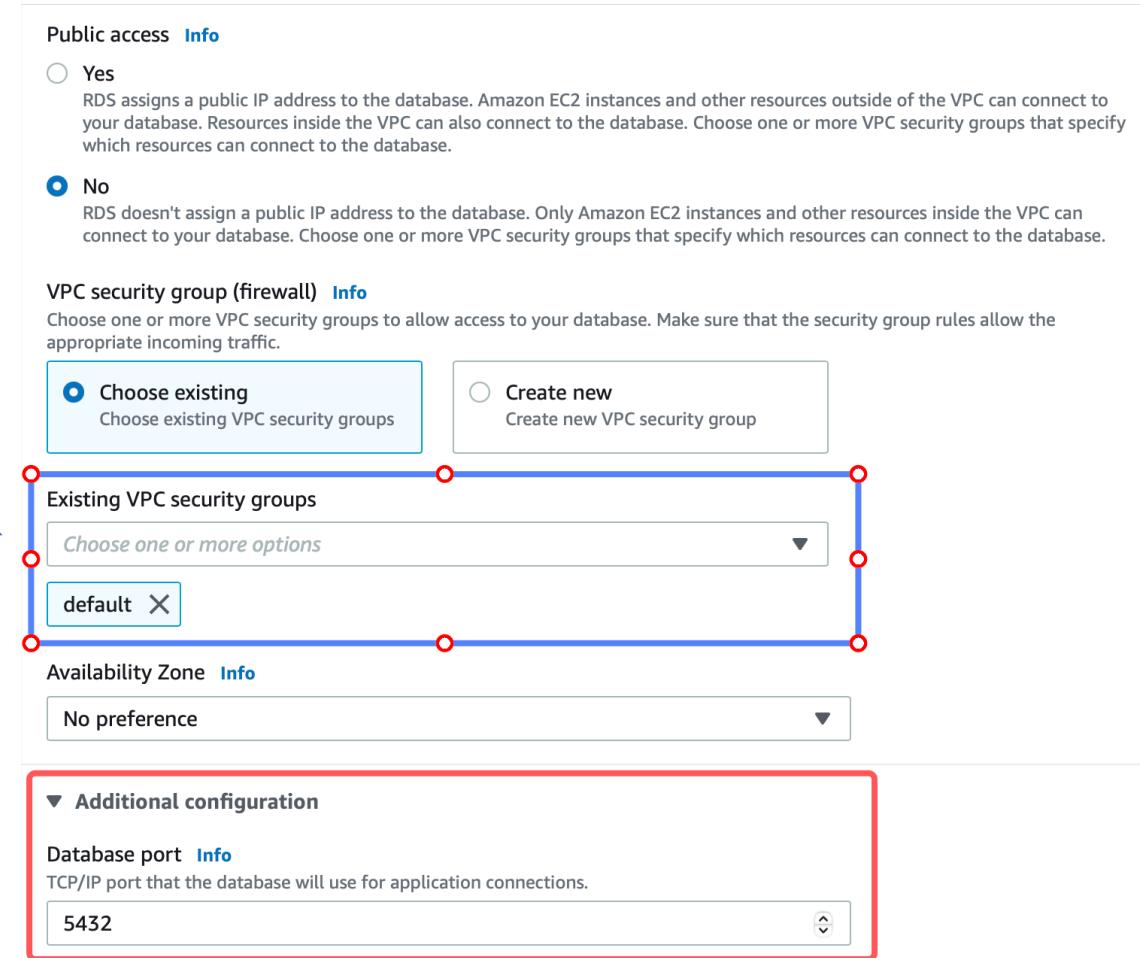
Availability Zone [Info](#)

No preference

▼ Additional configuration

Database port [Info](#)  
TCP/IP port that the database will use for application connections.

5432



# PFSwChO – Laboratorium11

## Usługa wielo-kontenerowa – wdrożenie na AWS (9)

### - usługa RDS (postgres) (5)

```
> worker          19
! .travis.yml      20
dockercap.de...    21
dockercap.yml      22
① README.md       23
  api:
    build:
      dockerfile: Dockerfile.dev
      context: ./server
    volumes:
      - /app/node_modules
      - ./server:/app
    environment:
      - REDIS_HOST=redis
      - REDIS_PORT=6379
      - PGUSER=postgres
      - PGHOST=postgres
      - PGDATABASE=postgres
      - PGPASSWORD=postgres_password
      - PGPORT=5432
```

Settings

**DB Instance Identifier** [Info](#)  
Type a name for your DB instance. The name must be unique across all DB Instances owned by your AWS account in the current AWS Region.  
**multi-fib-postgres**

The DB instance identifier is case-insensitive, but is stored as all lowercase (as in "mydbinstance"). Constraints: 1 to 60 alphanumeric characters or hyphens. First character must be a letter. Can't contain two consecutive hyphens. Can't end with a hyphen.

**Credentials Settings**

**Master username** [Info](#)  
Type a login ID for the master user of your DB instance.  
**postgres**

1 to 16 alphanumeric characters. First character must be a letter.  
 **Auto generate a password**  
Amazon RDS can generate a password for you, or you can specify your own password.

**Master password** [Info](#)  
\*\*\*\*\*  
Constraints: At least 8 printable ASCII characters. Can't contain any of the following: / (slash), '(single quote), "(double quote) and @ (at sign).

**Confirm password** [Info](#)  
\*\*\*\*\*

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)

**Database authentication**

**Database authentication options** [Info](#)

**Password authentication**  
Authenticates using database passwords.

**Password and IAM database authentication**  
Authenticates using the database password and user credentials through AWS IAM users and roles.

**Password and Kerberos authentication**  
Choose a directory in which you want to allow authorized users to authenticate with this DB instance using Kerberos Authentication.

**Monitoring**

**Performance Insights** [Info](#)

**Turn on Performance Insights** [Info](#)

▶ **Additional configuration**

Enhanced Monitoring

Proszę „odznaczyć wszystkie nieistotne opcje dla realizacji przykładu

### Backup

**Enable automated backups**

Creates a point-in-time snapshot of your database

### Encryption

**Enable encryption**

Choose to encrypt the given instance. Master key IDs and aliases appear in the list after they have been created using the AWS Key Management Service console. [Info](#)

### Log exports

Select the log types to publish to Amazon CloudWatch Logs

**PostgreSQL log**

**Upgrade log**

### IAM role

The following service-linked role is used for publishing logs to CloudWatch Logs.

RDS service-linked role

**Ensure that general, slow query, and audit logs are turned on. Error logs are enabled by default.** [Learn more](#)

### Maintenance

Auto minor version upgrade [Info](#)

**Enable auto minor version upgrade**

Enabling auto minor version upgrade will automatically upgrade to new minor versions as they are released. The automatic upgrades occur during the maintenance window for the database.

### Maintenance window

Select the period you want pending modifications or maintenance applied to the database by Amazon RDS.

**Choose a window**

**No preference**

### Deletion protection

**Enable deletion protection**

Protects the database from being deleted accidentally. While this option is enabled, you can't delete the database.

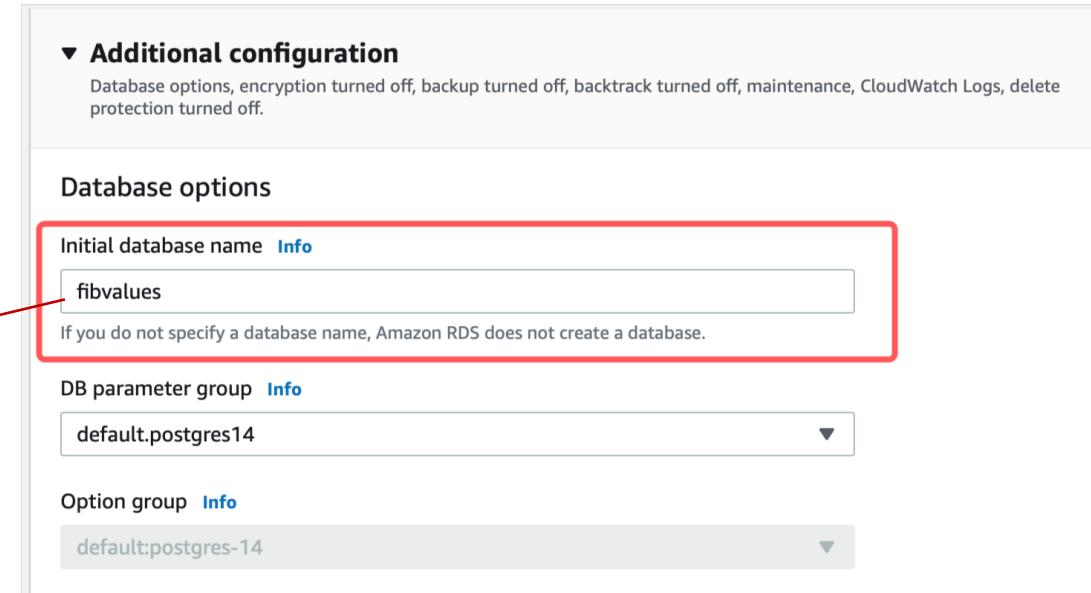
# PFSwChO – Laboratorium11

## Usługa wielo-kontenerowa – wdrożenie na AWS (11)

### - usługa RDS (postgres) (7)

```
> worker
! .travis.yml
dockercpompose.de...
dockercpomcompose.yml
 README.md

19  api:
20    build:
21      dockerfile: Dockerfile.dev
22      context: ./server
23      volumes:
24        - /app/node_modules
25        - ./server:/app
26      environment:
27        - REDIS_HOST=redis
28        - REDIS_PORT=6379
29        - PGUSER=postgres
30        - PGHOST=postgres
31        - PGDATABASE=postgres
32        - PGPASSWORD=postgres_password
33        - PGPORT=5432
```



**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)

The screenshot shows the Amazon RDS console with a success message: "Successfully created database multi-fib-postgres". The left sidebar includes options like Dashboard, Databases (selected), Query Editor, Performance insights, Snapshots, Exports in Amazon S3, Automated backups, Reserved instances, Proxies, Subnet groups, Parameter groups, Option groups, and Custom engine versions. The main area displays a "Databases" table with one row for "multi-fib-postgres". The table columns include DB identifier, Role, Engine, Region & AZ, Size, Status, CPU, Current activity, and Maintenance. The "Status" column shows "Available". Below this, a detailed view of the instance "multi-fib-postgres" is shown under the "Configuration" tab. The configuration table includes fields such as DB instance ID, Engine version, DB name, License model, Option groups, Amazon Resource Name (ARN), Resource ID, Created time, Parameter group, and Deletion protection. The instance class is db.t3.micro, and the storage type is Provisioned IOPS SSD (io1). The performance insights section indicates that Performance Insights are enabled but turned off.

Configuration	Instance class	Storage	Performance Insights
DB instance ID multi-fib-postgres	Instance class db.t3.micro	Encryption Not enabled	Performance Insights enabled Turned off
Engine version 14.5	vCPU 2	Storage type Provisioned IOPS SSD (io1)	
DB name fibvalues	RAM 1 GB	Storage 100 GiB	
License model Postgresql License	Availability	Provisioned IOPS 1000 IOPS	
Option groups <a href="#">default postgres-14</a> <small>In sync</small>	Master username postgres	Storage throughput -	
Amazon Resource Name (ARN) arnaws:rds:sus-east-2:039699603980:db:multi-fib-postgres	IAM DB authentication Not enabled	Storage autoscaling Disabled	
Resource ID db-ZZ4VOZMIMFZ7F4NGNEJ4E6NEIQ	Multi-AZ No		
Created time December 19, 2022, 12:16 (UTC+01:00)	Secondary Zone -		
Parameter group <a href="#">default postgres14</a> <small>In sync</small>			
Deletion protection Disabled			

# PFSwChO – Laboratorium11

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

The screenshot shows the Amazon ElastiCache Dashboard. On the left, there's a sidebar with options like 'Dashboard', 'Resources' (Redis clusters, Global datastores, Memcached clusters, Reserved nodes, Backups), and 'Configurations' (Subnet groups, Parameter groups, User management, User group management). The 'Dashboard' option is highlighted with a red box and has a red arrow pointing from it to the 'Create cluster' button. The main area is titled 'ElastiCache dashboard' and contains a section 'Getting started with ElastiCache' with three steps: 1. Initial setup, 2. Create a cluster, and 3. Connect with your application. Step 2 has a 'Create cluster' button. Below this, there's a large orange button labeled 'Create cluster ▲' which is also highlighted with a red box. Underneath it are two smaller buttons: 'Create Redis cluster' (also highlighted with a red box) and 'Create Memcached cluster'.

Opcja budowy klastra ma duży wpływ na koszty wykorzystania z usługą Elastic Cache – należy dokładnie przeanalizować cennik i wybrać najtańszą, wystarczającą konfigurację

**W trakcie lab wybieramy jeden węzeł !!!!**

# PFSwChO – Laboratorium11

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

**Cluster settings** Info

**Choose a cluster creation method**  
Choose one of the following options to create a new cluster.

**Configure and create a new cluster**  
Set all of the configuration options for your new cluster.

**Restore from backups**  
Use an existing backup or .rdb file to restore a cluster.

**Cluster mode**  
Scale your cluster dynamically with no downtime.

**Enabled**  
Cluster mode enables replication across multiple shards for enhanced scalability and availability.

**Disabled**  
The Redis cluster will have a single shard (node group) with one primary node and up to 5 read replica.

**Info** If you choose cluster mode disabled you cannot change the number of shards. The configuration supports all Redis commands and functionality but limits maximum cache size and performance. [Learn more](#)



**Cluster info**  
Use the following options to configure the cluster.

**Name**  
  
The name is required, can have up to 40 characters and must not contain spaces.

**Description - optional**

**Location**  
Choose whether to host the cluster in the AWS Cloud or on premises.

**Location**  
 **AWS Cloud**  
Use the AWS Cloud for your ElastiCache instances.

**On premises**  
Create your ElastiCache instances on an Outpost (through AWS Outposts). You need to create a subnet ID on an Outpost first.

**Multi-AZ**  
 **Enable**  
Multi-AZ provides enhanced high availability through automatic failover to a read replica, cross AZs, in case of a primary node failover.

**Auto-failover**  
 **Enable**  
ElastiCache Auto Failover provides enhanced high availability through automatic failover to a read replica in case of a primary node failover.

**Info** Disabling ElastiCache Multi-AZ on your Redis cluster reduces your fault tolerance. In the unlikely event of an Availability Zone failure or loss of network connectivity, your Redis cluster will become unavailable. [Learn more](#)



# PFSwChO – Laboratorium11

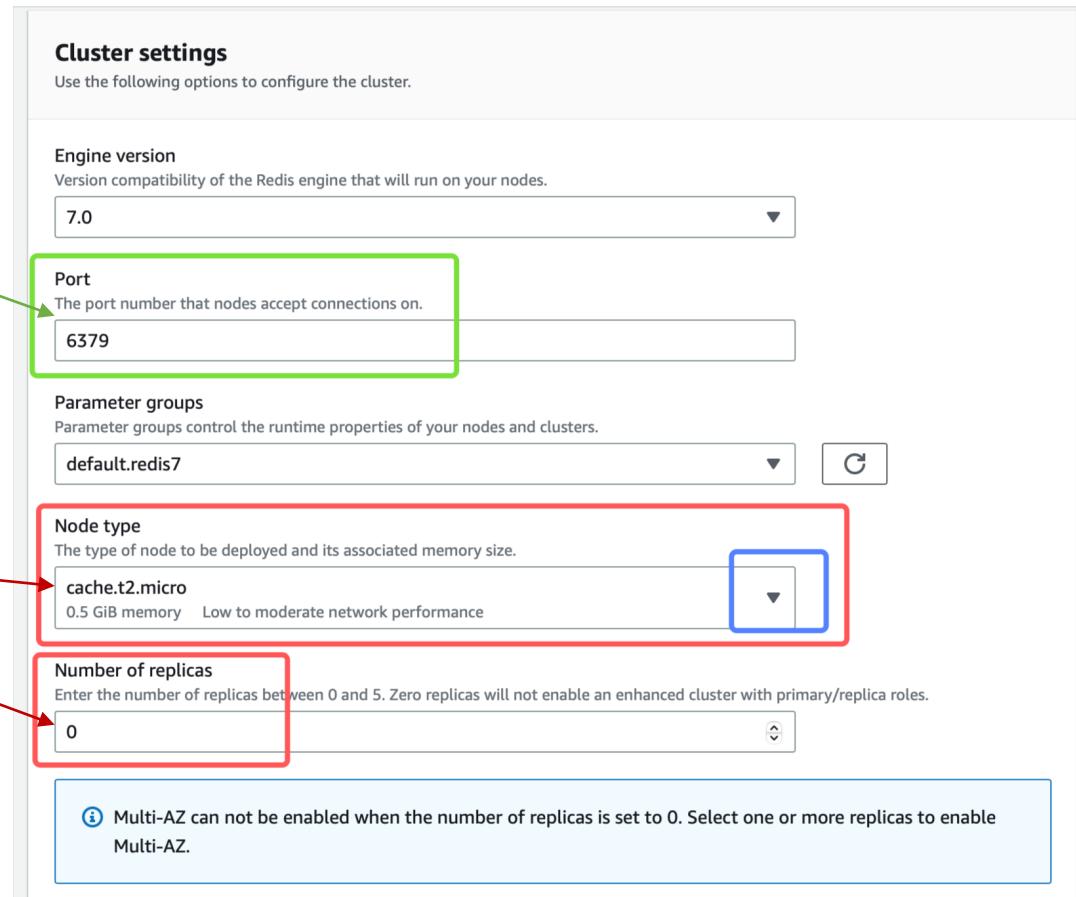
Usługa wielo-kontenerowa – wdrożenie na AWS (15)

- 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)

**Subnet group settings**  
A subnet group is a collection of subnets (typically private). Designate a subnet group for your clusters running in an Amazon Virtual Private Cloud (VPC) environment.

**Subnet groups**

Choose existing subnet group    Create a new subnet group

**Subnet groups**  
A collection of subnets that you can designate for your clusters running in an Amazon VPC.

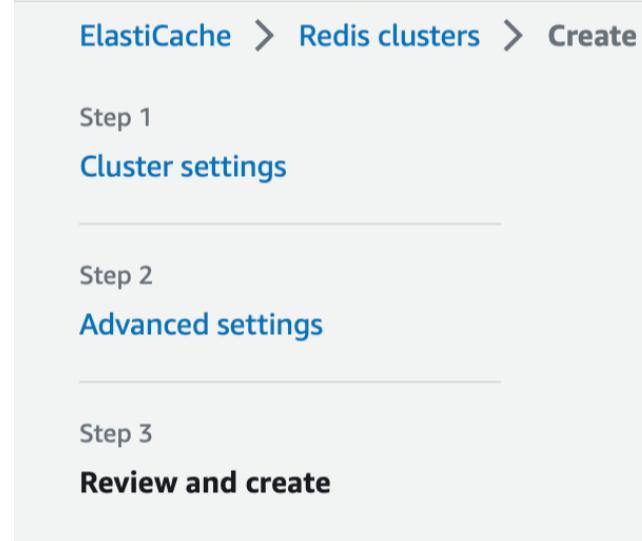
**Associated subnets (3)**

Availability Zone	Subnet ID	CIDR block (IPv4)
us-east-2a	subnet-0c1dc67e9e20e5c06	172.31.0.0/20
us-east-2b	subnet-00ee99899a93f40d9	172.31.16.0/20
us-east-2c	subnet-04430001cc24c259d	172.31.32.0/20

**Availability Zone placements**  
Use the following fields to configure placements for Availability Zones.

**Availability Zone placements**  
HA mode - Globally, distribute AZs to maximize AZ spread across shard masters. At the second level, spread nodes within a shard across AZs for within-shard HA. Low latency mode - For fast writes, put all shard masters in the same AZ.

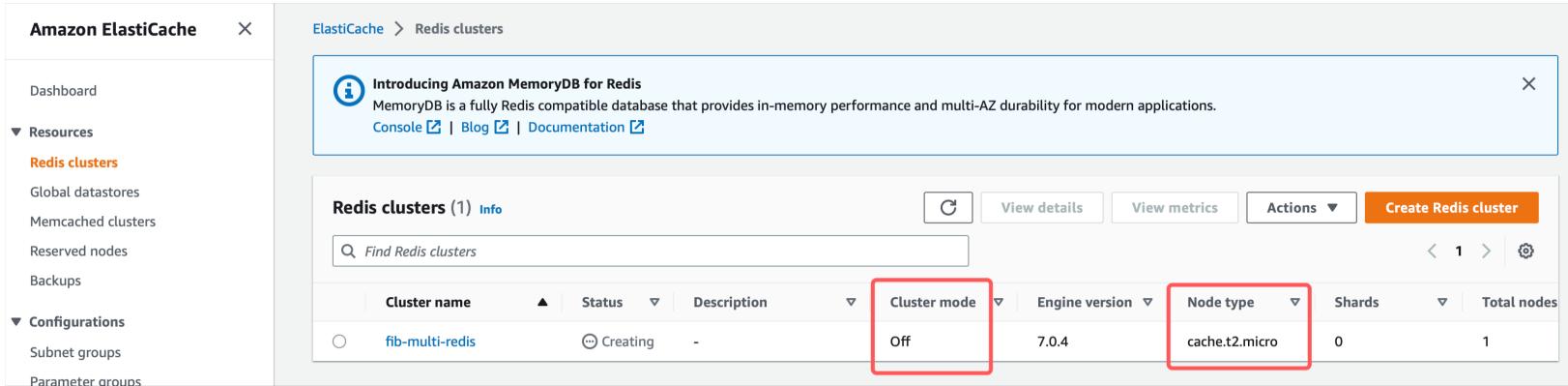
No preference



# PFSwChO – Laboratorium11

Usługa wielo-kontenerowa – wdrożenie na AWS (16)

- usługa Elastic Cache (redis) (4)



The screenshot shows the Amazon ElastiCache console. On the left, there's a sidebar with options like Dashboard, Resources (selected), Redis clusters (highlighted in orange), Global datastores, Memcached clusters, Reserved nodes, Backups, Configurations, Subnet groups, and Parameter groups. The main area is titled 'ElastiCache > Redis clusters'. A blue box at the top says 'Introducing Amazon MemoryDB for Redis' with links to Console, Blog, and Documentation. Below it, a table lists 'Redis clusters (1) info'. The table has columns: Cluster name, Status, Description, Cluster mode, Engine version, Node type, Shards, and Total nodes. The first row shows 'fib-multi-redis' in 'Creating' status, '-' in Description, 'Off' in Cluster mode (which is highlighted with a red box), '7.0.4' in Engine version, 'cache.t2.micro' in Node type (also highlighted with a red box), '0' in Shards, and '1' in Total nodes.

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**