

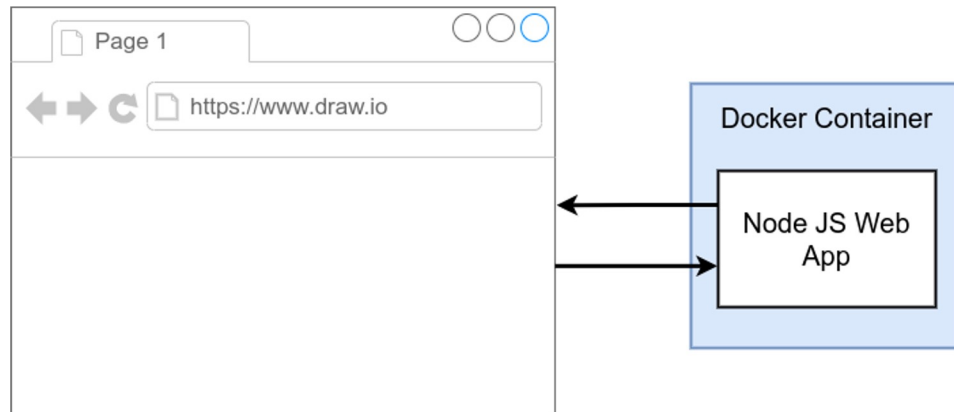
PFSwChO - Laboratorium2

W procesie tworzenie aplikacji typu full-stack kluczową rolę odgrywa optymalizacja mikroserwisów (obrazów Docker) i poprawne posługiwanie się mechanizmami wersjonowania (obrazy w wersji deweloperskiej i produkcyjnej)

<https://docs.docker.com/build/building/cache/>

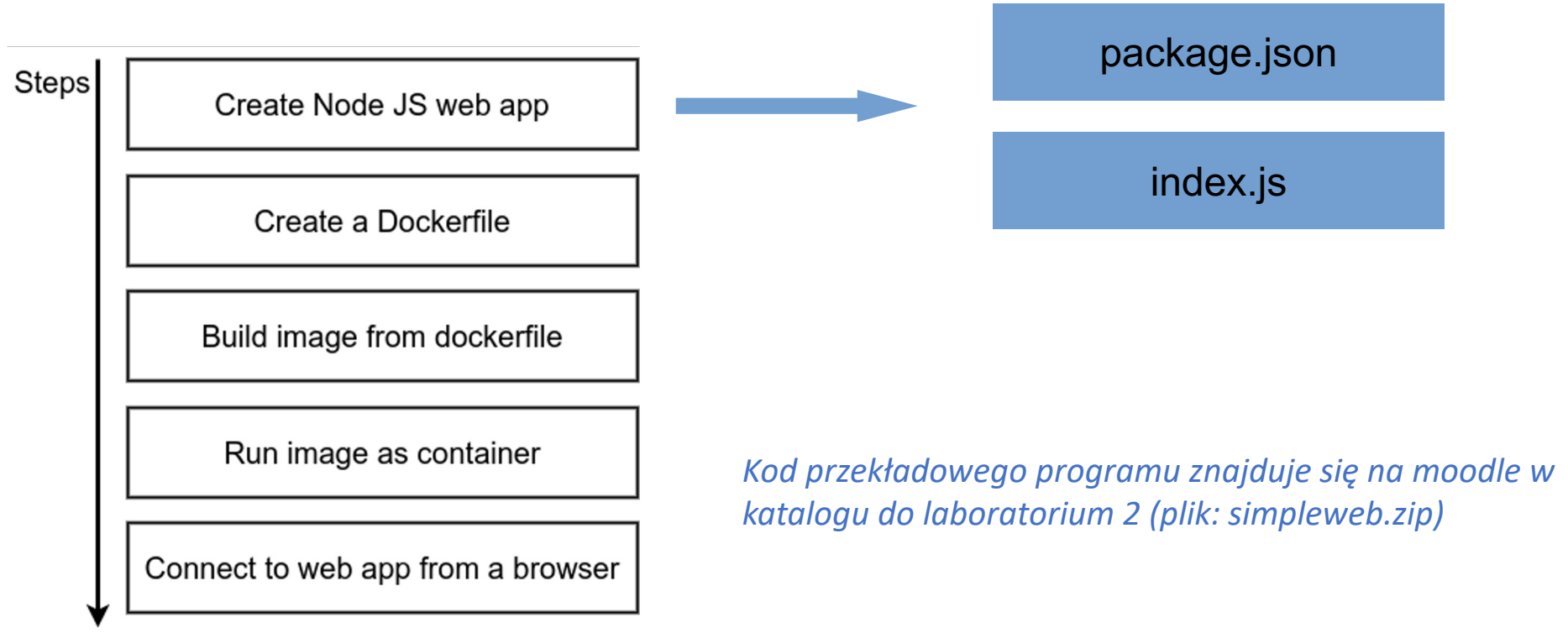
Należy zapoznać się z informacjami wyżej i wykorzystywać je w trakcie WSZYSTKICH realizowanych zadań. Zgodność z dobrymi praktykami przy budowaniu obrazów będzie wpływała na ocenę poszczególnych zadań.

Prosta idea → przykłady na co zwracać szczególną uwagę przy budowaniu obrazów



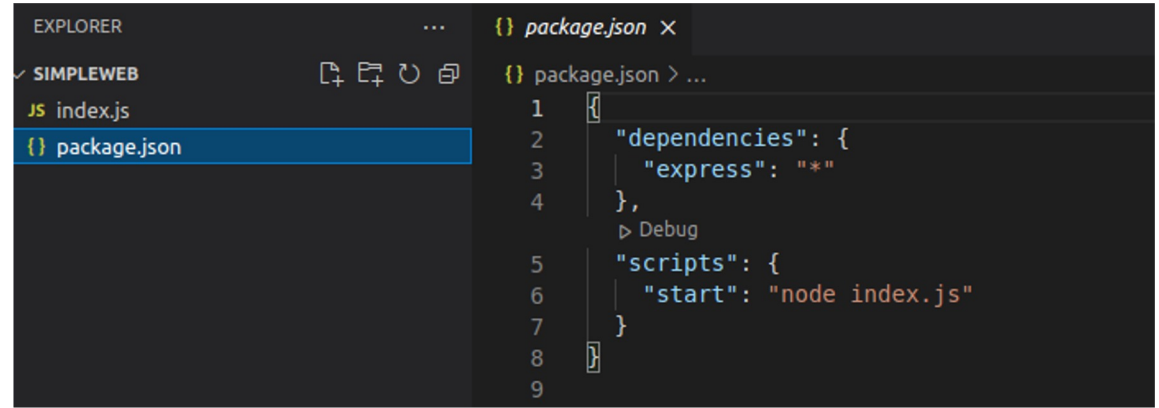
PFSwChO - Laboratorium2

Do tej pory powinniśmy wiedzieć, że postępowanie z wykorzystaniem kontenerów Docker polega na:



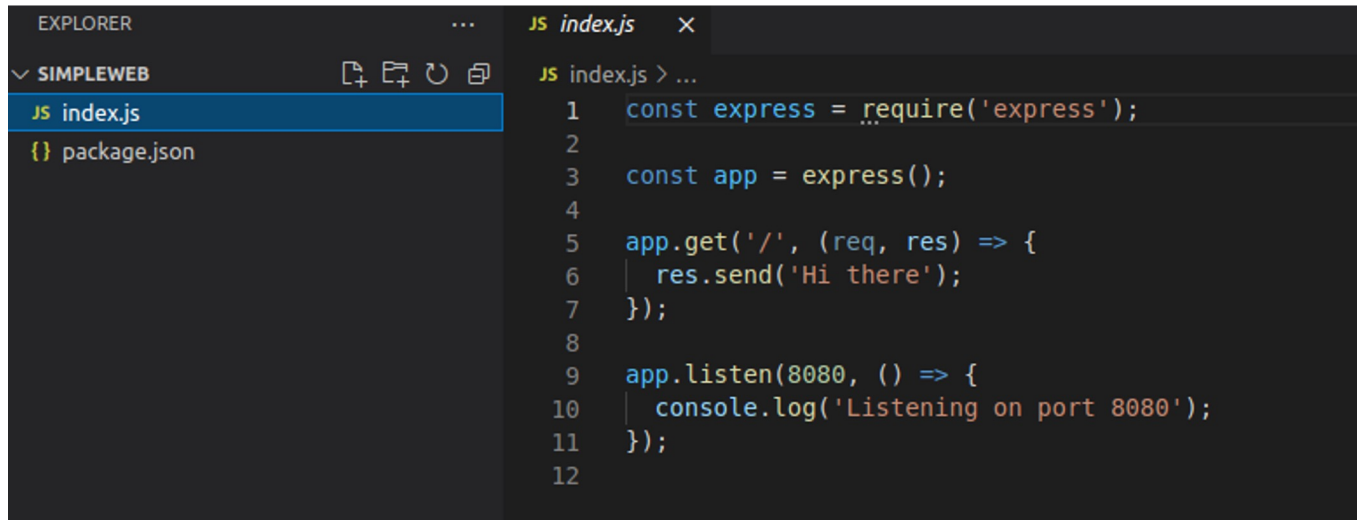
PFSwChO - Laboratorium2

Node JS web app
“simpleweb”



The screenshot shows the VS Code interface with the Explorer sidebar on the left displaying the file structure of a project named 'SIMPLEWEB'. The files listed are 'index.js' and 'package.json', with 'package.json' selected. The main editor area on the right displays the content of 'package.json'.

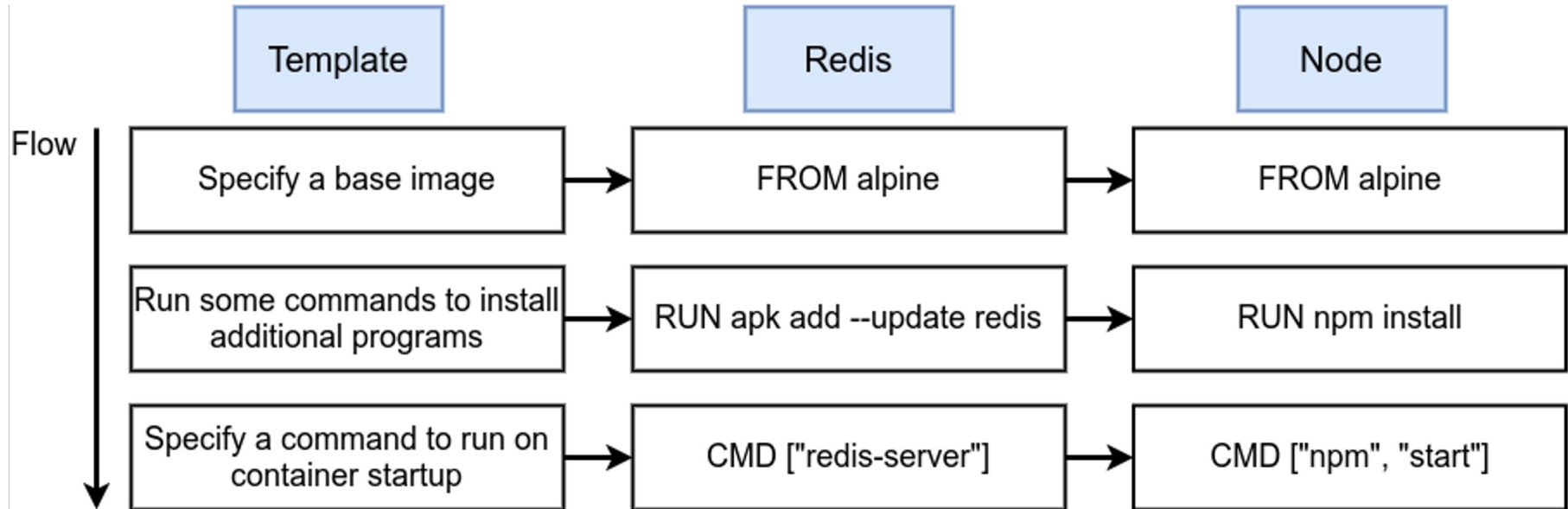
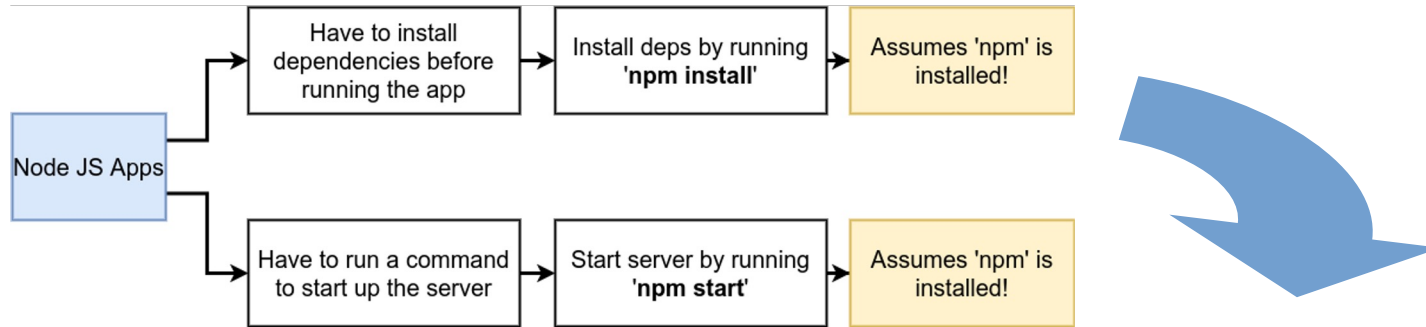
```
package.json > ...
1 {
2   "dependencies": {
3     "express": "*"
4   },
5   "scripts": {
6     "start": "node index.js"
7   }
8 }
9
```



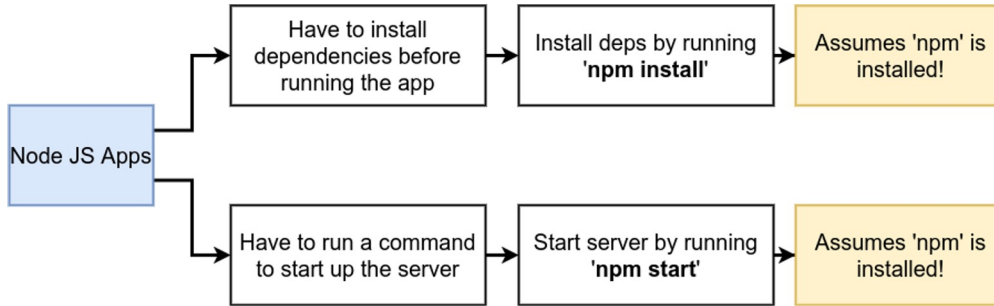
The screenshot shows the VS Code interface with the Explorer sidebar on the left displaying the file structure of a project named 'SIMPLEWEB'. The files listed are 'index.js' and 'package.json', with 'index.js' selected. The main editor area on the right displays the content of 'index.js'.

```
JS index.js > ...
1 const express = require('express');
2
3 const app = express();
4
5 app.get('/', (req, res) => {
6   res.send('Hi there');
7 });
8
9 app.listen(8080, () => {
10   console.log('Listening on port 8080');
11 });
12
```

PFSwChO - Laboratorium2



PFSwChO - Laboratorium2

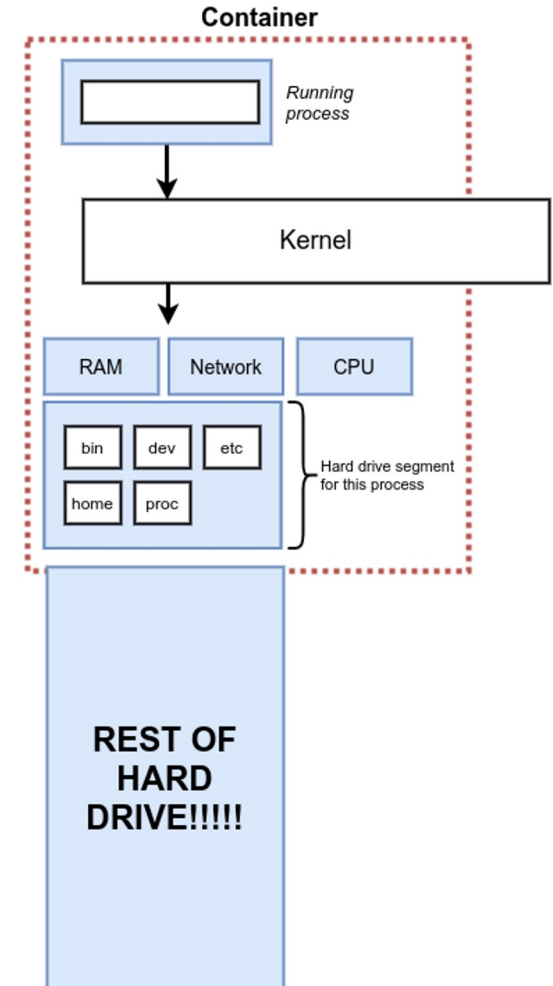
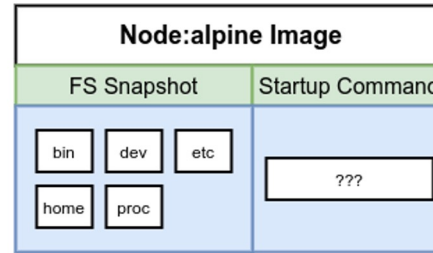
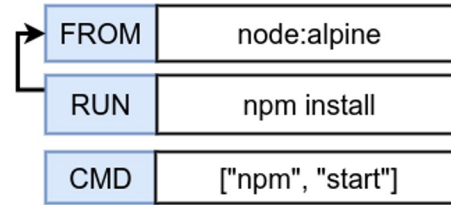


The screenshot shows a code editor with two panels. The left panel, titled 'EXPLORER', shows a project named 'SIMPLEWEB' with files 'index.js' and 'package.json'. The 'Dockerfile1' file is selected. The right panel, titled 'Dockerfile1 > ...', shows the content of the Dockerfile:

```
1 # Specify a base image
2 FROM alpine
3
4 # Install some dependencies
5 RUN npm install
6
7 # Default command
8 CMD ["npm", "start"]
```

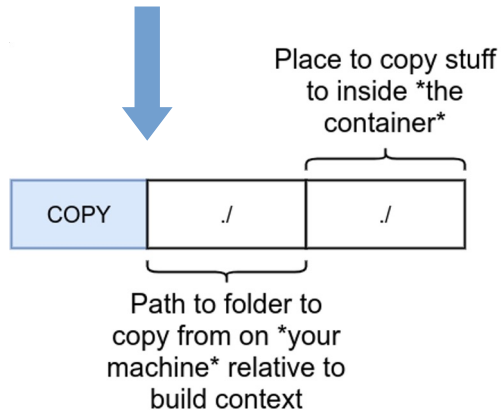
PFSwChO - Laboratorium2

Powstaje wątpliwość: czy pliki aplikacji “simpleweb” są widoczne dla npm w trakcie procesu budowania obrazu?



PFSwChO - Laboratorium2

Instruction telling Docker Server what to do	Argument to the instruction
FROM	node:alpine
COPY	./ ./
RUN	npm install
CMD	["npm", "start"]



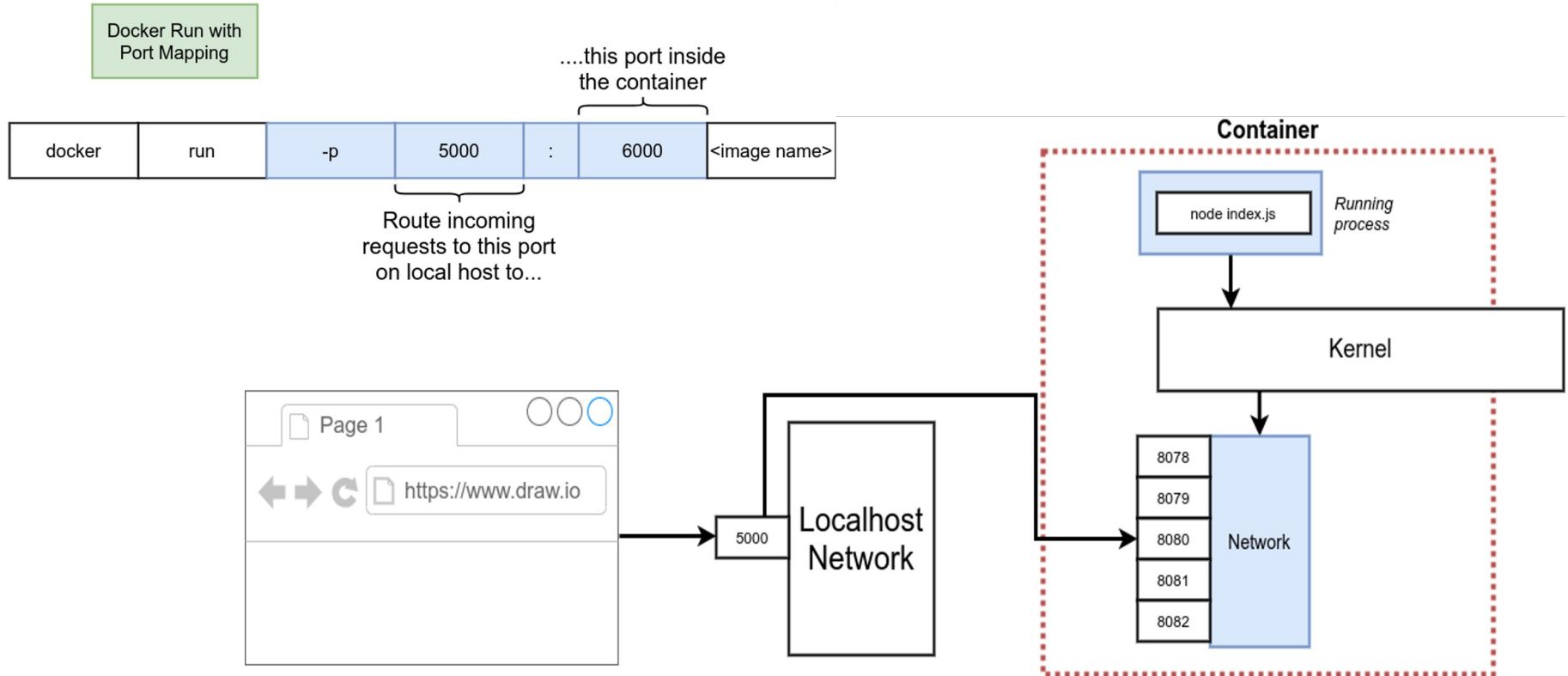
Najprostsze rozwiązanie – zapewnić, że pliki aplikacji będą widziane przez npm

The screenshot shows a VS Code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project named 'SIMPLEWEB' with files 'Dockerfile1', 'Dockerfile2', 'Dockerfile3', and 'Dockerfile4'. The code editor shows the content of 'Dockerfile4'.

```
1 # Specify a base image
2 FROM node:alpine
3
4 WORKDIR /usr/app
5
6 # Install some dependencies
7 COPY ./ ./
8 RUN npm install
9
10 # Default command
11 CMD ["npm", "start"]
```

PFSwChO - Laboratorium2

I jeszcze mapowanie portów i jeszcze jedna uwaga



PFSwChO - Laboratorium2


Zmieńmy coś w projekcie → np. Niech “Hi there” zastąpi “Bye, that is all”

```
slawek@acer4k:~/Documents/PFSLab3/Simpleweb$ docker build -t local/simpleweb -f Dockerfile4 .
Sending build context to Docker daemon 8.192kB
Step 1/5 : FROM node:alpine
----> 0f877e6f48f8
Step 2/5 : WORKDIR /usr/app
----> Using cache
----> 96a77453c76d
Step 3/5 : COPY ./ ./
----> f71160a64f52
Step 4/5 : RUN npm install
----> Running in 80cb0c9ba064

added 50 packages, and audited 51 packages in 7s

found 0 vulnerabilities
npm notice
npm notice New minor version of npm available! 8.0.0 -> 8.1.0
npm notice Changelog: <https://github.com/npm/cli/releases/tag/v8.1.0>
npm notice Run `npm install -g npm@8.1.0` to update!
npm notice
Removing intermediate container 80cb0c9ba064
----> 82cd886ac676
Step 5/5 : CMD ["npm", "start"]
----> Running in fddc13a7c575
Removing intermediate container fddc13a7c575
----> 5853db04479c
Successfully built 5853db04479c
Successfully tagged local/simpleweb:latest
```

JS index.js X

JS index.js >  app.get("/") callback

```
1  const express = require('express');
2
3  const app = express();
4
5  app.get('/', (req, res) => {
6    res.send('Bye, that is all');
7  });
8
9  app.listen(8080, () => {
10   console.log('Listening on port 8080');
11 });
12
```

PFSwChO - Laboratorium2

Jeśli powrócimy do poprzedniej wersji pliku index.js to ...jest poprawa dzięki temu, że Dockerfile poprawiono tak jak poniżej:



```
slawek@acer4k:~/Documents/PFSLab3/Simpleweb$ docker build -t local/simpleweb -f Dockerfile5 .
Sending build context to Docker daemon 8.192kB
Step 1/6 : FROM node:alpine
---> 0f877e6f48f8
Step 2/6 : WORKDIR /usr/app
---> Using cache
---> 96a77453c76d
Step 3/6 : COPY ./package.json ./
---> Using cache
---> 351d10d84c14
Step 4/6 : RUN npm install
---> Using cache
---> a2916116ccb5
Step 5/6 : COPY ./ ./
---> 469d93ab51b2
Step 6/6 : CMD ["npm", "start"]
---> Running in 655bc9840fda
Removing intermediate container 655bc9840fda
---> 6d705b5b13ea
Successfully built 6d705b5b13ea
Successfully tagged local/simpleweb:latest
```

EXPLORER

▼ SIMPLEWEB

≡ Dockerfile1

≡ Dockerfile2

≡ Dockerfile3

≡ Dockerfile4

🔗 Dockerfile5

JS index.js

{ } package.json

Dockerfile5 x

🔗 Dockerfile5 > ...

```
1 # Specify a base image
2 FROM node:alpine
3
4 WORKDIR /usr/app
5
6 # Install some dependencies
7 COPY ./package.json ./
8 RUN npm install
9 COPY ./ ./
10
11 # Default command
12 CMD ["npm", "start"]
```

Idea BuildKit

Od wersji 18.09 środowiska Docker dostępna jest nowa wersja silnika (backendu) budowania obrazów. W obecnej wersji docker Desktop jest on domyślnym składnikiem środowiska.

Podstawowe cechy BuildKit według dokumentacji

Docker Docs:

https://docs.docker.com/develop/develop-images/build_enhancements/

GitHub:

<https://github.com/moby/buildkit>

Przed kolejnym laboratorium należy KONIECZNIE zapoznać się z dokumentacją dostępną pod adresami jak wyżej.

PFSwChO - Laboratorium2

Podstawowe cechy BuildKit według dokumentacji

Detect and skip executing unused build stages

Parallelize building independent build stages

Incrementally transfer only the changed files in your build context between builds

Detect and skip transferring unused files in your build context

Use external Dockerfile implementations with many new features

Avoid side-effects with rest of the API (intermediate images and containers)

Prioritize your build cache for automatic pruning

Przełączanie pomiędzy klasycznym (starym) silnikiem a nowym (BuildKit) możliwe jest przez zdeklarowanie zmiennej środowiskowej **DOCKER_BUILDKIT**

DOCKER_BUILDKIT=0 stary silnik

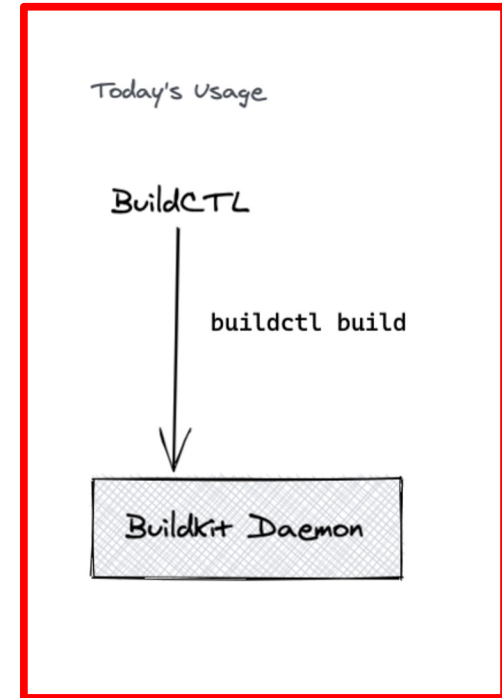
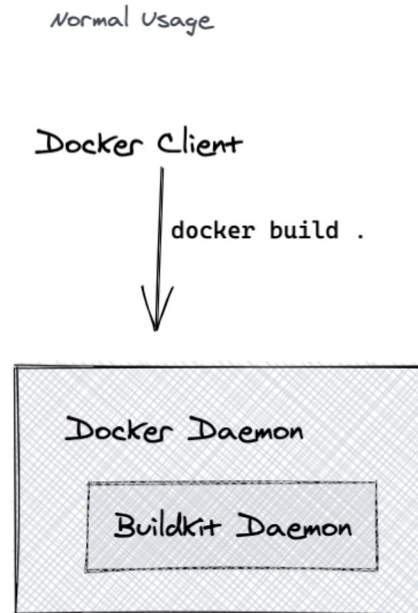
DOCKER_BUILDKIT=1 nowy silnik (default)

PFSwChO - Laboratorium2

BuildKit - frontends

*Przed kolejnym laboratorium należy
KONIECZNIE zapoznać się z
dokumentacją dostępną pod
adresem jak niżej.*

<https://hub.docker.com/r/docker/dockerfile/>



PFSwChO - Laboratorium2

Z laboratorium 2 nie jest wymagane wykonanie sprawozdania.

Przed kolejnymi zajęciami NALEŻY zapoznać się z dokumentacjami zaznaczonymi w instrukcji kolorem czerwonym

ZADANIE DO WYKONANIE PRZED KOLEJNYM LABORATORIUM

W systemie GitHub należy utworzyć swoje publiczne repozytorium zawierające przekładową aplikację Simpleweb (załączoną do materiałów do tego laboratorium). Następnie należy skonfigurować i zweryfikować możliwość dostępu do tego repozytorium za pomocą SSH.