

การขึ้นโปรเจกต์ React + Vite + TS + SWC + Docker (Dev and Prod)

Step 1: คำสั่งขึ้นโปรเจกต์ React ด้วย Vite

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
PS D:\ReactNodeDocker> npm create vite@latest
```

Step 2: ตั้งชื่อโปรเจกต์ และเลือกรูปแบบเป็น typescript + swc

```
PS D:\ReactNodeDocker> npm create vite@latest
✓ Project name: ... reacte-layout-test
✓ Select a framework: » React
✓ Select a variant: » TypeScript + SWC

Scaffolding project in D:\ReactNodeDocker\reacte-layout-test...

Done. Now run:

  cd reacte-layout-test
  npm install
  npm run dev
```

Step 3: เปิดเข้า VSCode

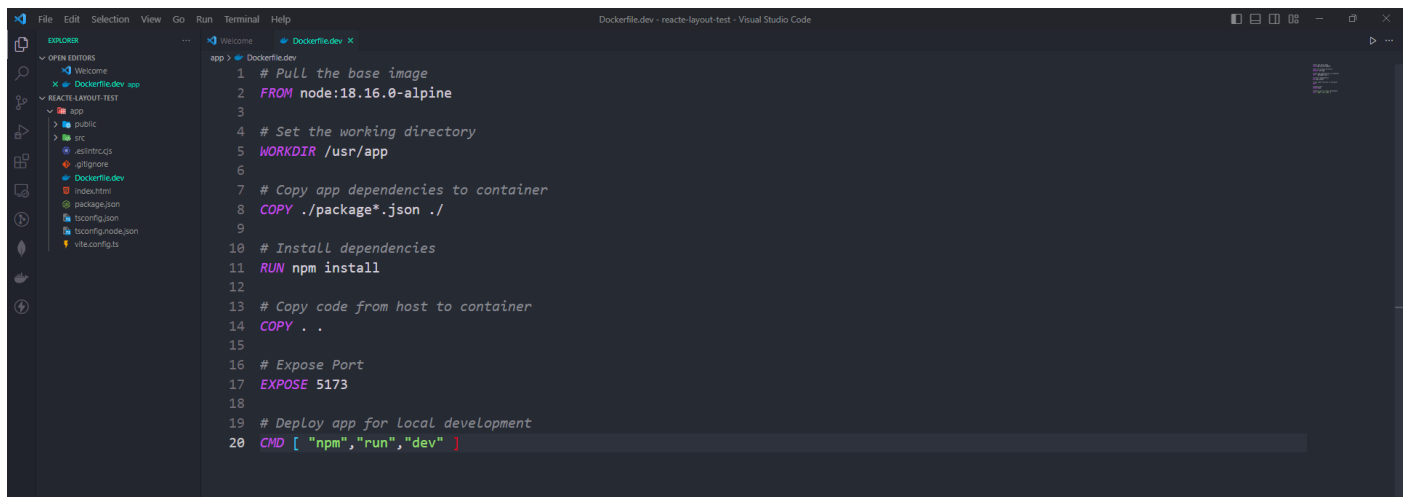
```
PS D:\ReactNodeDocker> code .\reacte-layout-test\ -r
```

กด tab เพื่อเลือก Folder

Step 4: ทำการย้ายไฟล์ทั้งหมดไว้ใน react-layout/app

```
▼ REACTE-LAYOUT-TEST
  ✓ app
    > public
    > src
    .eslintrc.cjs
    .gitignore
    index.html
    package.json
    tsconfig.json
    tsconfig.node.json
    vite.config.ts
```

Step 5: สร้างไฟล์ Dockerfile.dev สำหรับ Devmode



```
1 # Pull the base image
2 FROM node:18.16.0-alpine
3
4 # Set the working directory
5 WORKDIR /usr/app
6
7 # Copy app dependencies to container
8 COPY ./package*.json ./
9
10 # Install dependencies
11 RUN npm install
12
13 # Copy code from host to container
14 COPY . .
15
16 # Expose Port
17 EXPOSE 5173
18
19 # Deploy app for local development
20 CMD [ "npm", "run", "dev" ]
```

Pull the base image

FROM node:18.16.0-alpine

Set the working directory

WORKDIR /usr/app

Copy app dependencies to container

COPY ./package*.json ./

Install dependencies

RUN npm install

Copy code from host to container

COPY . .

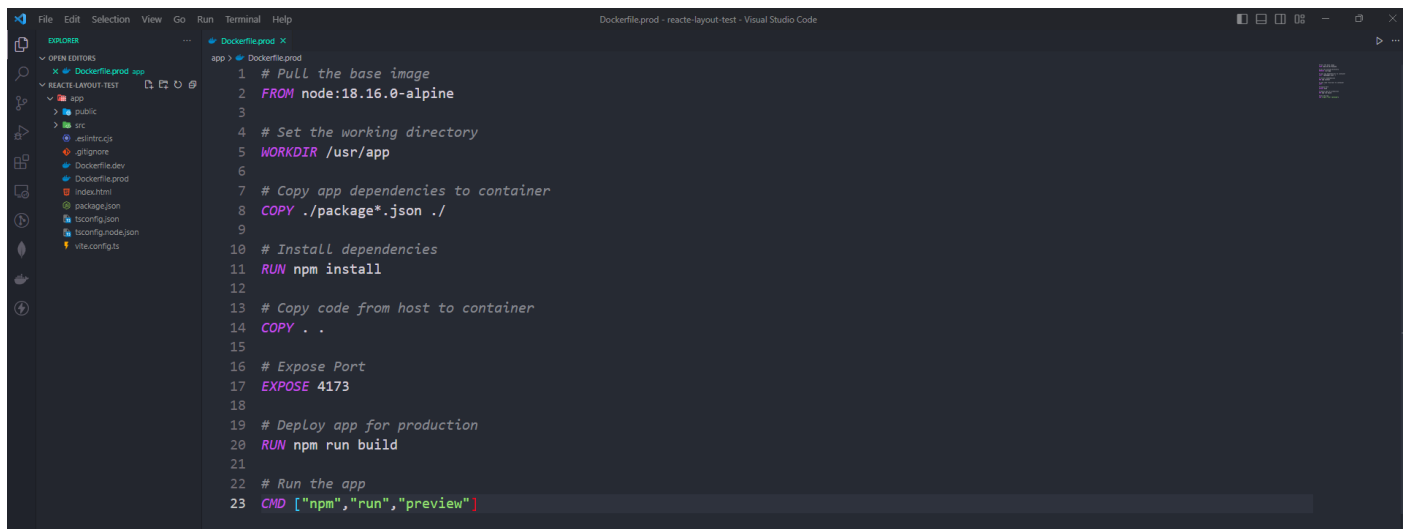
Expose Port

EXPOSE 5173

Deploy app for local development

CMD ["npm", "run", "dev"]

Step 6: สร้างไฟล์ Dockerfile.prod สำหรับ Production



```
1 # Pull the base image
2 FROM node:18.16.0-alpine
3
4 # Set the working directory
5 WORKDIR /usr/app
6
7 # Copy app dependencies to container
8 COPY ./package*.json ./
9
10 # Install dependencies
11 RUN npm install
12
13 # Copy code from host to container
14 COPY . .
15
16 # Expose Port
17 EXPOSE 4173
18
19 # Deploy app for production
20 RUN npm run build
21
22 # Run the app
23 CMD ["npm","run","preview"]
```

Pull the base image

FROM node:18.16.0-alpine

Set the working directory

WORKDIR /usr/app

Copy app dependencies to container

COPY ./package*.json ./

Install dependencies

RUN npm install

Copy code from host to container

COPY . .

Expose Port

EXPOSE 4173

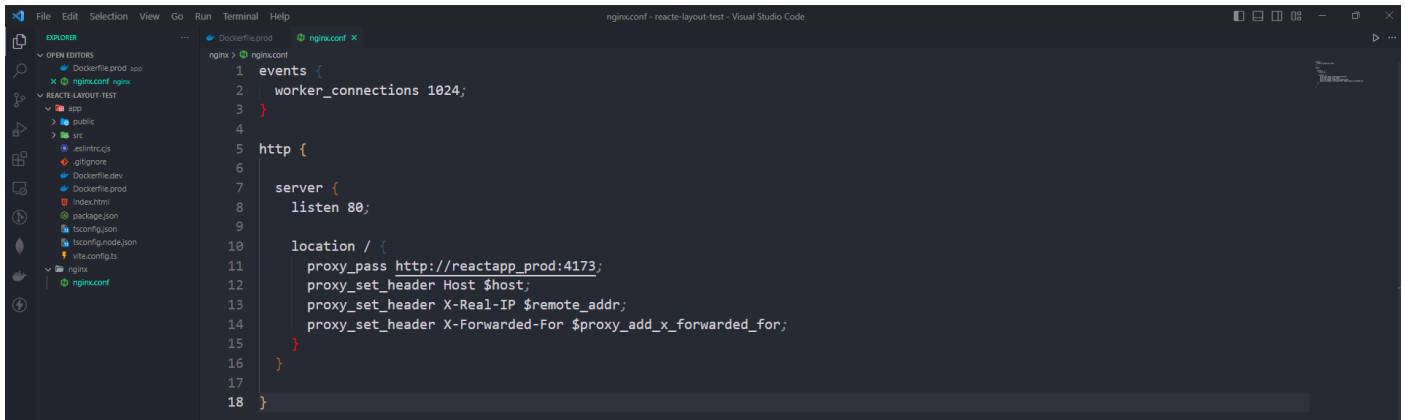
Deploy app for production

RUN npm run build

Run the app

CMD ["npm","run","preview"]

Step 7: สร้าง config Nginx web server

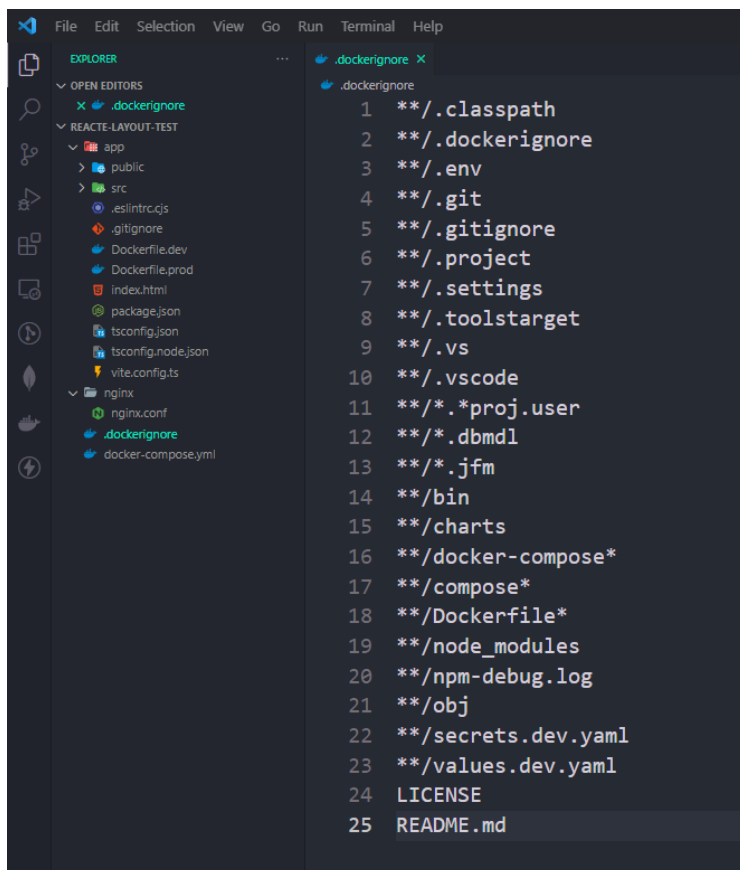
A screenshot of the Visual Studio Code editor interface. The Explorer sidebar on the left shows a project structure with folders like 'react-app' and 'react-app-prod'. The main editor area displays the 'nginx.conf' file. The configuration is as follows:

```
1 events {  
2     worker_connections 1024;  
3 }  
4  
5 http {  
6  
7     server {  
8         listen 80;  
9  
10        location / {  
11            proxy_pass http://reactapp_prod:4173;  
12            proxy_set_header Host $host;  
13            proxy_set_header X-Real-IP $remote_addr;  
14            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;  
15        }  
16    }  
17 }  
18 }
```

```
events {  
    worker_connections 1024;  
}  
  
http {  
  
    server {  
        listen 80;  
  
        location / {  
            proxy_pass http://reactapp_prod:4173;  
            proxy_set_header Host $host;  
            proxy_set_header X-Real-IP $remote_addr;  
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;  
        }  
    }  
}
```

Step 8: สร้าง docker-compose.yml ไฟล์สำหรับสร้าง container react + nginx

สร้างไฟล์ .dockerignore สำหรับกำหนดว่าไม่ให้ docker container นำไฟล์เหล่านี้ไปรวม

A screenshot of the Visual Studio Code editor. The Explorer sidebar on the left shows a project structure with folders like 'app', 'public', 'src', and 'nginx'. The main editor area displays the content of the '.dockerignore' file, which is a list of 25 patterns to ignore. The patterns include various system files, development files, and project-specific files.

```
1  **/.classpath
2  **/.dockerignore
3  **/.env
4  **/.git
5  **/.gitignore
6  **/.project
7  **/.settings
8  **/.toolstarget
9  **/.vs
10 **/.vscode
11 **/*.proj.user
12 **/*.dbmdl
13 **/*.jfm
14 **/bin
15 **/charts
16 **/docker-compose*
17 **/compose*
18 **/Dockerfile*
19 **/node_modules
20 **/npm-debug.log
21 **/obj
22 **/secrets.dev.yaml
23 **/values.dev.yaml
24 LICENSE
25 README.md
```

**/.classpath

**/.dockerignore

**/.env

**/.git

**/.gitignore

**/.project

**/.settings

**/.toolstarget

**/.vs

**/.vscode

**/*.proj.user

**/*.dbmdl

**/*.jfm

**/bin

**/charts

**/docker-compose*

**/compose*

**/Dockerfile*

**/node_modules

**/npm-debug.log

**/obj

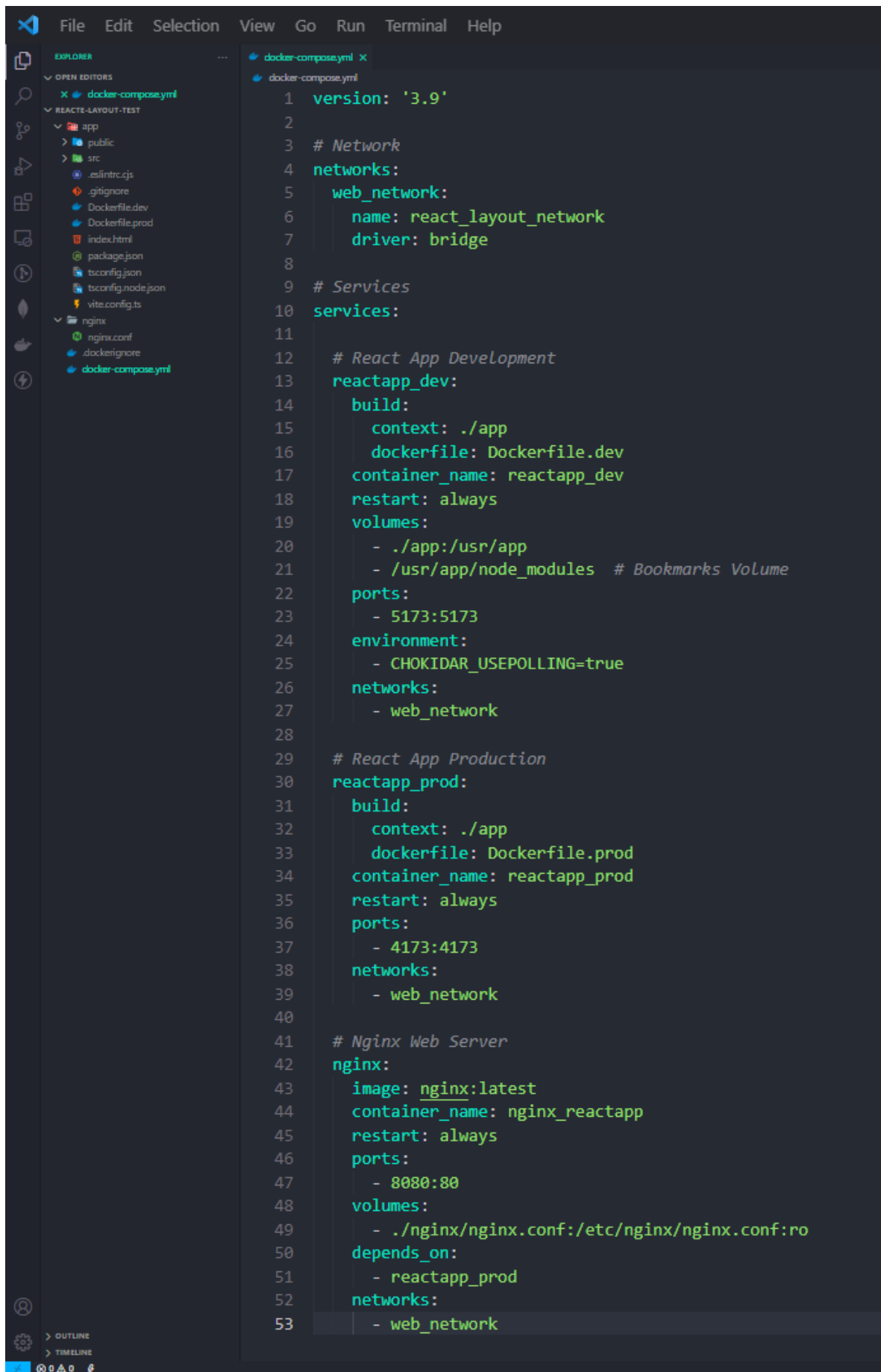
**/secrets.dev.yaml

**/values.dev.yaml

LICENSE

README.md

กำหนด script ใน docker-compose.yml



```
1  version: '3.9'
2
3  # Network
4  networks:
5    web_network:
6      name: react_layout_network
7      driver: bridge
8
9  # Services
10 services:
11
12  # React App Development
13  reactapp_dev:
14    build:
15      context: ./app
16      dockerfile: Dockerfile.dev
17    container_name: reactapp_dev
18    restart: always
19    volumes:
20      - ./app:/usr/app
21      - /usr/app/node_modules # Bookmarks Volume
22    ports:
23      - 5173:5173
24    environment:
25      - CHOKIDAR_USEPOLLING=true
26    networks:
27      - web_network
28
29  # React App Production
30  reactapp_prod:
31    build:
32      context: ./app
33      dockerfile: Dockerfile.prod
34    container_name: reactapp_prod
35    restart: always
36    ports:
37      - 4173:4173
38    networks:
39      - web_network
40
41  # Nginx Web Server
42  nginx:
43    image: nginx:latest
44    container_name: nginx_reactapp
45    restart: always
46    ports:
47      - 8080:80
48    volumes:
49      - ./nginx/nginx.conf:/etc/nginx/nginx.conf:ro
50    depends_on:
51      - reactapp_prod
52    networks:
53      - web_network
```

version: '3.9'

Network

networks:

web_network:

name: react_layout_network

driver: bridge

Services

services:

React App Development

reactapp_dev:

build:

context: ./app

dockerfile: Dockerfile.dev

container_name: reactapp_dev

restart: always

volumes:

- ./app:/usr/app

- /usr/app/node_modules # Bookmarks Volume

ports:

- 5173:5173

environment:

- CHOKIDAR_USEPOLLING=true

networks:

- web_network

React App Production

reactapp_prod:

build:

context: ./app

dockerfile: Dockerfile.prod

container_name: reactapp_prod

restart: always

ports:

- 4173:4173

networks:

- web_network

Nginx Web Server

nginx:

image: nginx:latest

container_name: nginx_reactapp

restart: always

ports:

- 8080:80

volumes:

- ./nginx/nginx.conf:/etc/nginx/nginx.conf:ro

depends_on:

- reactapp_prod

networks:

- web_network

Step 9: ตรวจสอบความถูกต้องของไฟล์ docker-compose.yml

docker compose config

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

PS D:\ReactNodeDocker\reacte-layout-test> docker compose config
name: reacte-layout-test
services:
  nginx:
    container_name: nginx_reactapp
    depends_on:
      reactapp_prod:
        condition: service_started
    image: nginx:latest
    networks:
      web_network: null
    ports:
      - mode: ingress
        target: 80
        published: "8080"
        protocol: tcp
        create_host_path: true
      - type: volume
        target: /usr/app/node_modules
        volume: {}
  reactapp_prod:
    build:
      context: D:\ReactNodeDocker\reacte-layout-test\app
      dockerfile: Dockerfile.prod
    container_name: reactapp_prod
    networks:
      web_network: null
    ports:
      - mode: ingress
        target: 4173
        published: "4173"
        protocol: tcp
    restart: always
networks:
  web_network:
    name: react_layout_network
    driver: bridge
PS D:\ReactNodeDocker\reacte-layout-test> 
```

Step 10: แก้ไข config ของไฟล์ vite.config.ts

```
app > vite.config.ts > default
1  import { defineConfig } from 'vite'
2  import react from '@vitejs/plugin-react-swc'
3
4  // https://vitejs.dev/config/
5  export default defineConfig({
6    plugins: [react()],
7    server: {
8      watch: {
9        usePolling: true,
10     },
11     host: true,
12     strictPort: true,
13     port: 5173,
14   }
15 })
```

```
import { defineConfig } from 'vite'
```

```
import react from '@vitejs/plugin-react-swc'
```

```
// https://vitejs.dev/config/
```

```
export default defineConfig({
```

```
  plugins: [react()],
```

```
  server: {
```

```
    watch: {
```

```
      usePolling: true,
```

```
    },
```

```
    host: true,
```

```
    strictPort: true,
```

```
    port: 5173,
```

```
  }
```

```
})
```

Step 11: ทำการสั่ง Run Container ด้วย docker compose

```
docker compose up -d
```

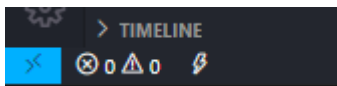
ลองทำการ Down Service ทั้งหมดดู

```
docker compose down
```

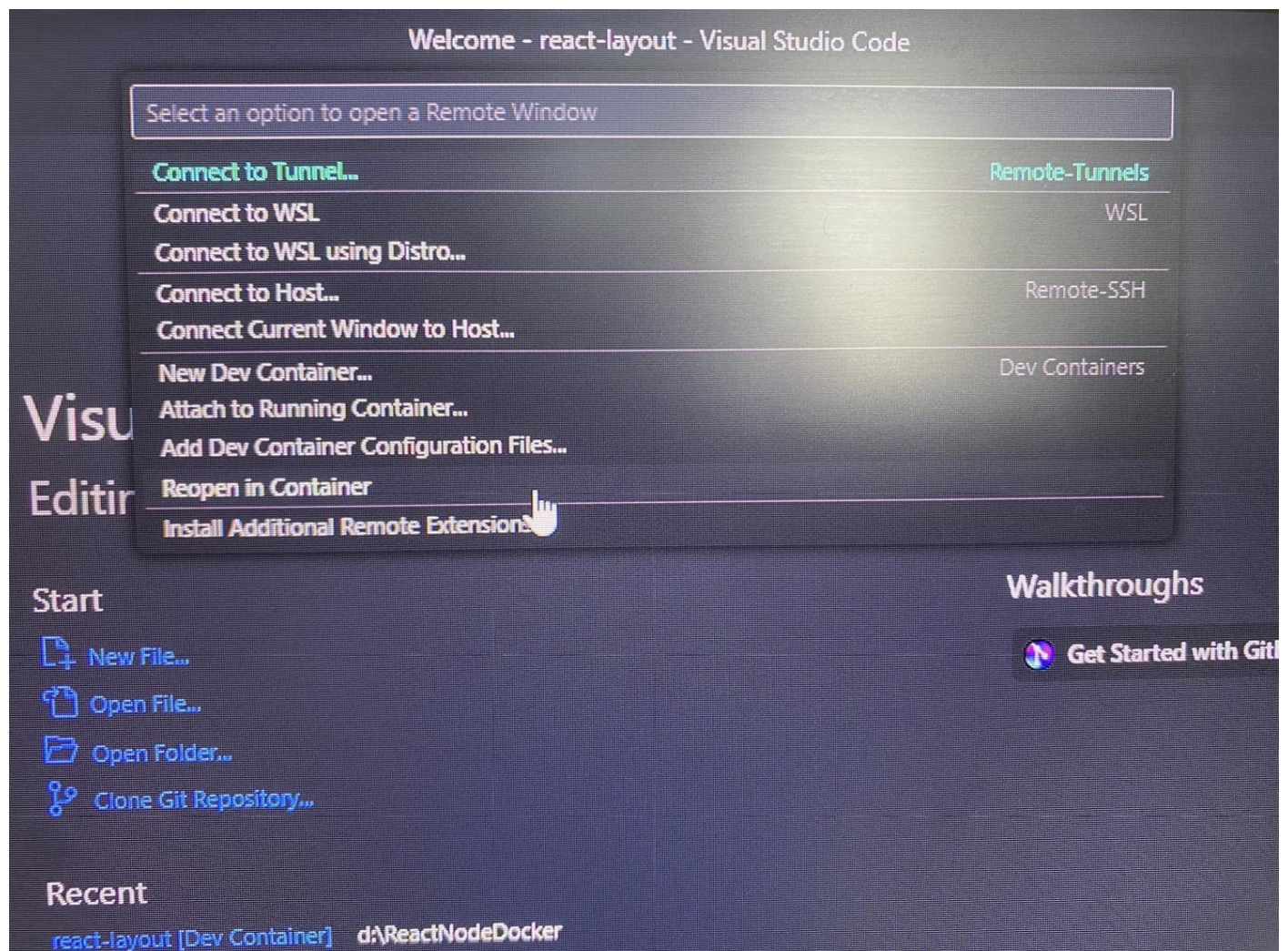
Step 12: ทดลอง Run Service "reactapp_dev" เพียง service เดียว

```
PS D:\ReactNodeDocker\react-layout> docker compose up reactapp_dev -d
[+] Running 1/0
 ✓ Container reactapp_dev Running
PS D:\ReactNodeDocker\react-layout> 
```

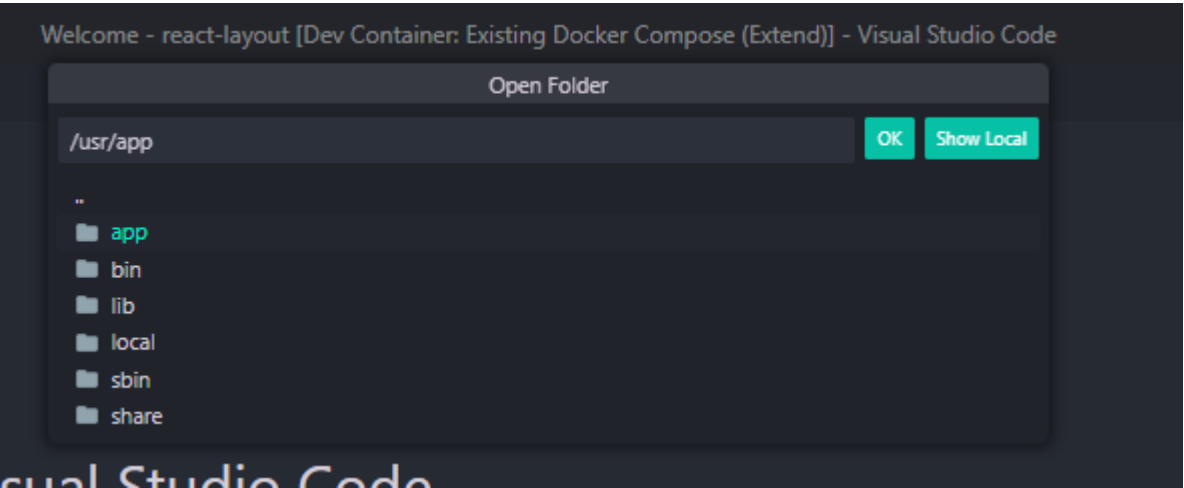
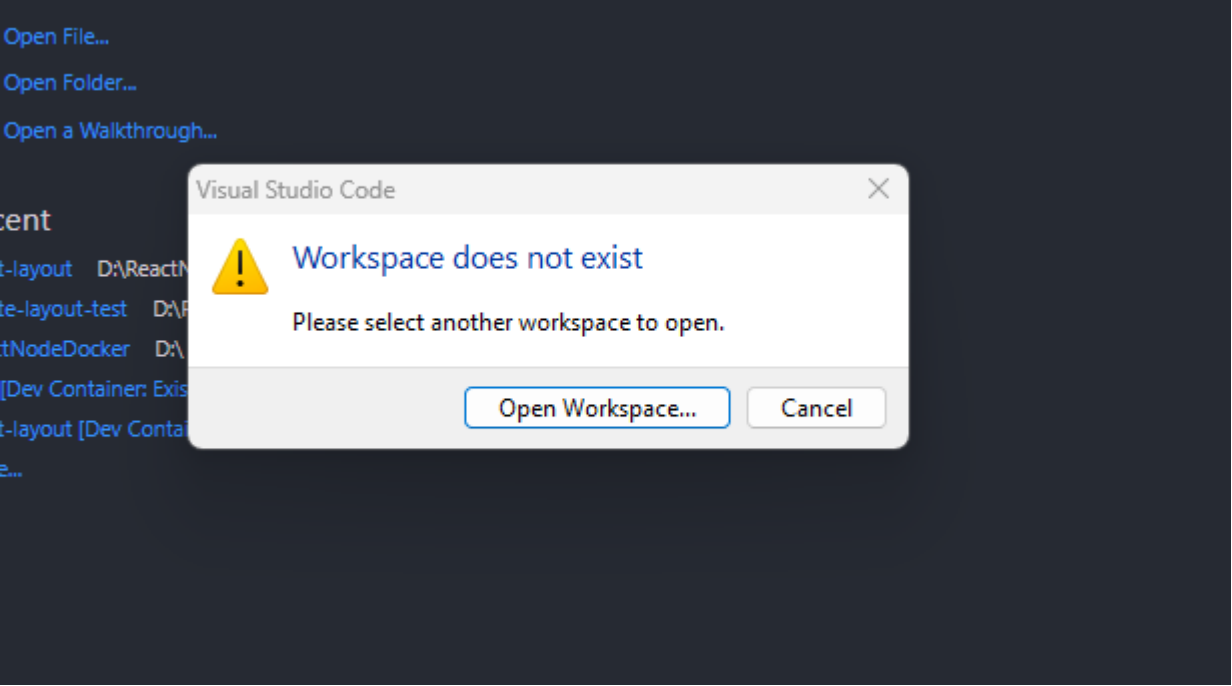
วิธีการเข้าไปที่หน้า Dev



Choose Reopen in Container



Choose open Workspace



successfully

