

# **Aplicação de contêineres Docker no desenvolvimento NodeJS**

---



# Docker

---

“Os contêineres permitem que um desenvolvedor empacote um aplicativo com todas as partes necessárias, como bibliotecas e outras dependências, e implante-o como um pacote. Ao fazer isso, graças ao contêiner, o desenvolvedor pode ter certeza de que o aplicativo será executado em qualquer outra máquina Linux, independentemente das configurações personalizadas que a máquina possa ter e que possam diferir da máquina usada para escrever e testar o código.”

["What is Docker?" 2020]



# Comparativo

---

## VANTAGENS

- Fácil configuração para novos ambientes
- Código rodando sobre recursos iguais ou próximos ao ambiente de produção
- Possibilidade de simular máquinas com diferentes capacidades de processamento
- “Portabilidade” da execução do código
- Docker roda em praticamente qualquer ambiente
- Ampla comunidade e documentações
- Diversas ferramentas para gerenciamento de contêineres

## DESVANTAGENS

- Algumas aplicações podem demandar de configurações complexas
- No Windows o Docker pode apresentar instabilidades ao trabalhar com recursos mais avançados (como volumes) — WSL costuma resolver

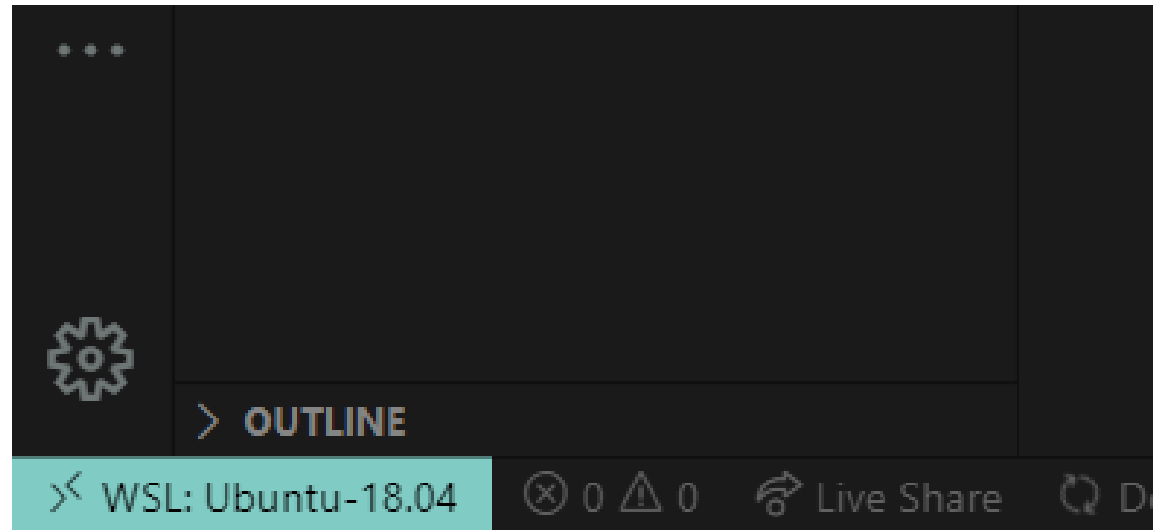




# Hands-on

---



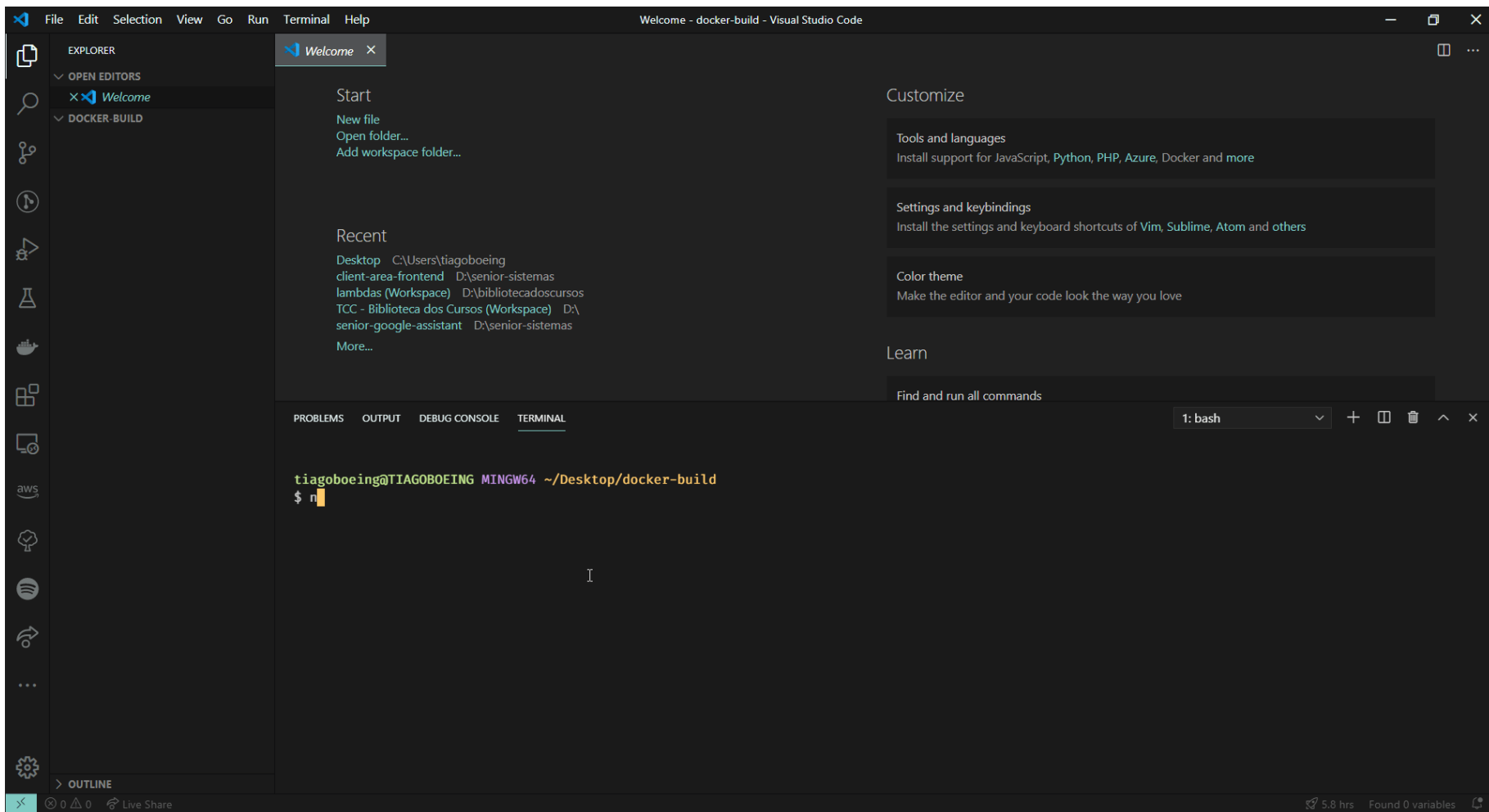


# WSL 2 com VSCode



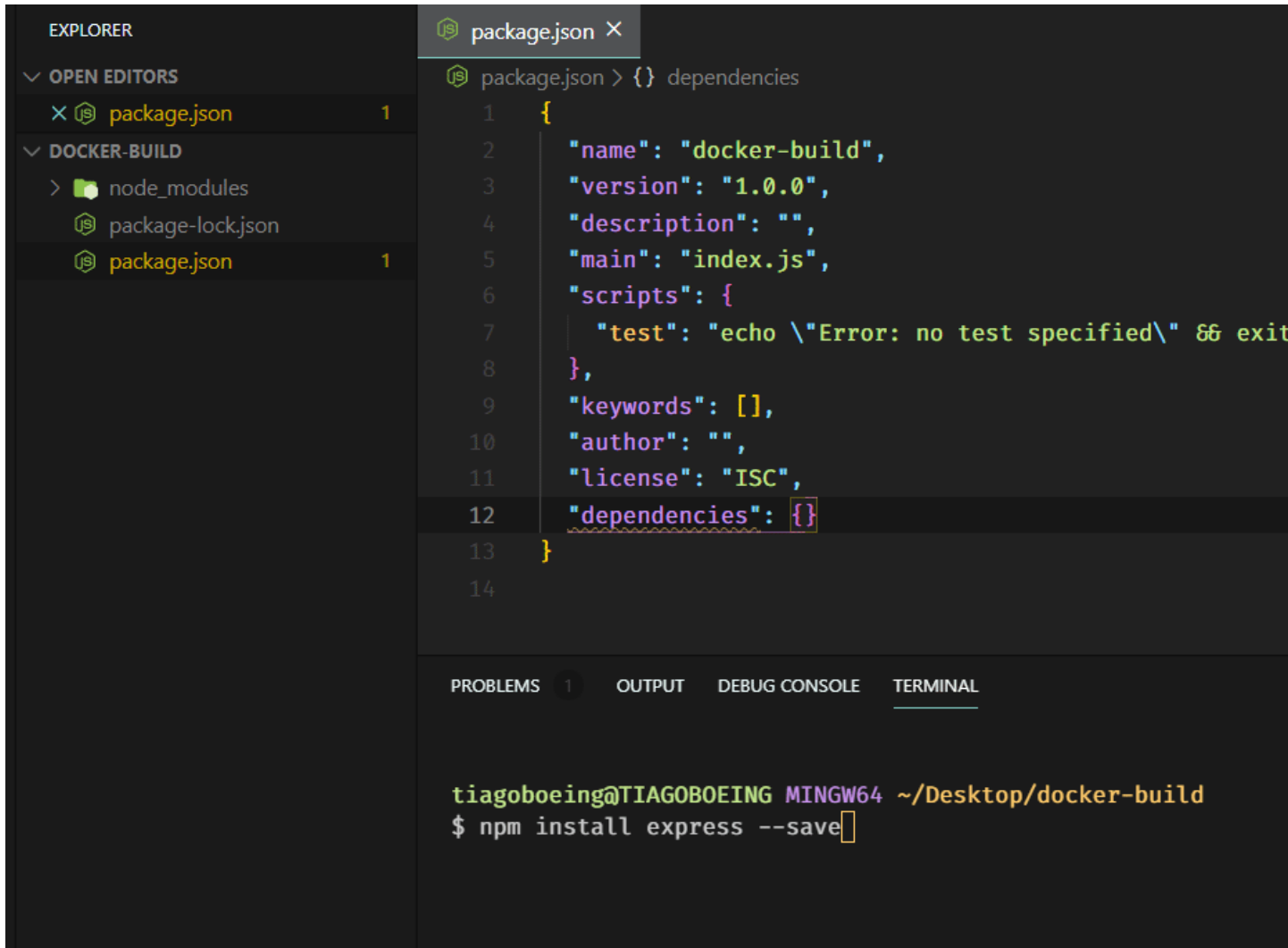
---

<https://docs.microsoft.com/pt-br/windows/wsl/install-win10#update-to-wsl-2>



# npm init -y





The screenshot shows the Visual Studio Code interface. On the left, the Explorer sidebar shows a project named 'DOCKER-BUILD' with files 'node\_modules', 'package-lock.json', and 'package.json'. The main editor area displays the 'package.json' file with the following content:

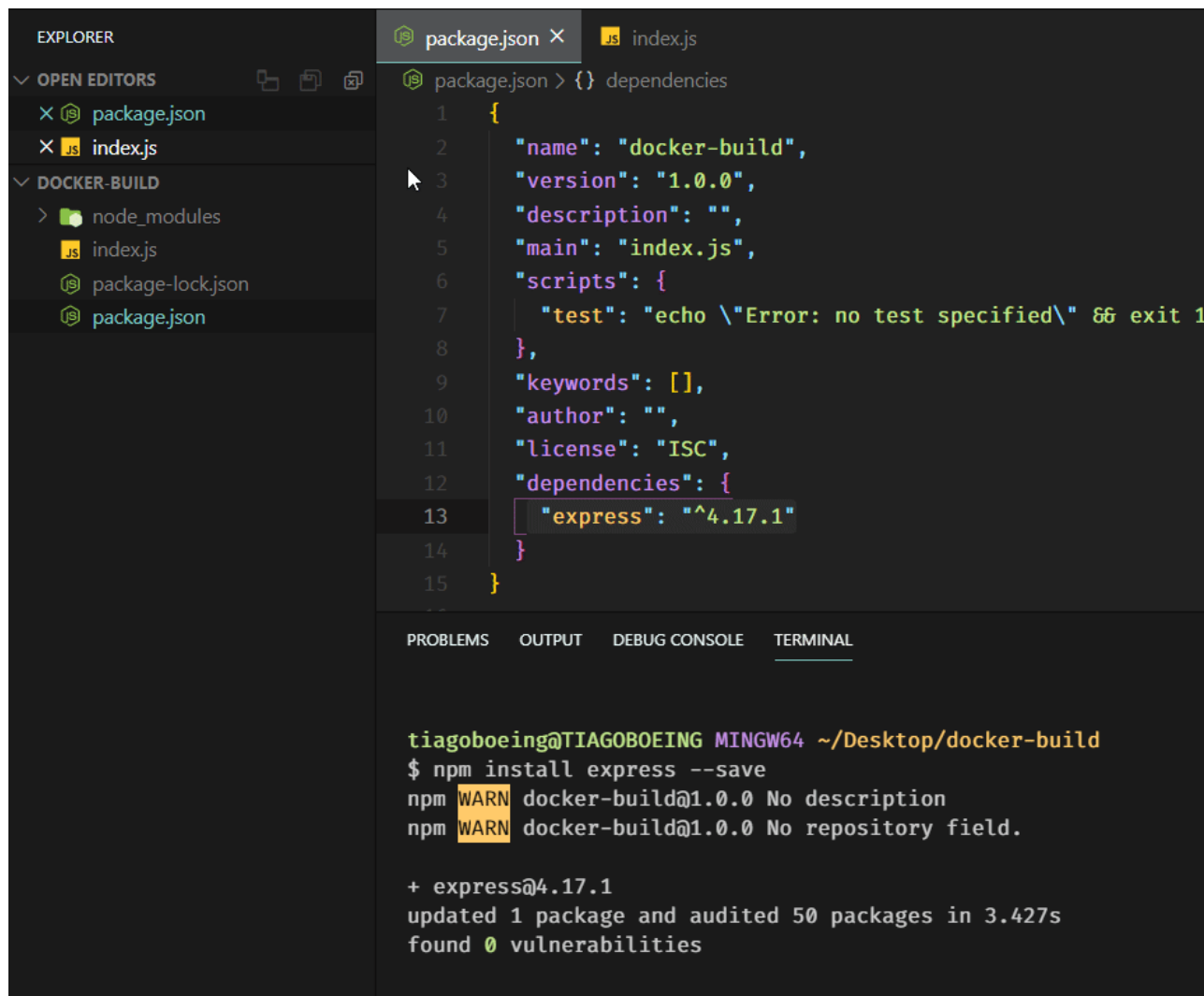
```
1  {
2    "name": "docker-build",
3    "version": "1.0.0",
4    "description": "",
5    "main": "index.js",
6    "scripts": {
7      "test": "echo \"Error: no test specified\" && exit 1",
8    },
9    "keywords": [],
10   "author": "",
11   "license": "ISC",
12   "dependencies": {}
13 }
```

Below the editor, the TERMINAL panel shows the command prompt for 'tiagoboeing@TIAGOBOEING' in a 'MINGW64' shell at the path '~/Desktop/docker-build'. The command entered is 'npm install express --save'.

# Instalar Express

`npm install express --save`





The screenshot shows a Visual Studio Code editor with a dark theme. On the left, the Explorer sidebar shows a project named 'DOCKER-BUILD' containing files like 'node\_modules', 'index.js', 'package-lock.json', and 'package.json'. The main editor area has two tabs: 'package.json' and 'index.js'. The 'package.json' tab is active, displaying a JSON configuration for a project named 'docker-build' with version '1.0.0'. The 'dependencies' section is highlighted, showing 'express' as a dependency with version '^4.17.1'. Below the editor, the Terminal panel is open, showing the output of an npm command. The terminal text indicates that 'express@4.17.1' was installed successfully, with some warnings about missing description and repository fields.

```
package.json > {} dependencies
1  {
2    "name": "docker-build",
3    "version": "1.0.0",
4    "description": "",
5    "main": "index.js",
6    "scripts": {
7      "test": "echo \"Error: no test specified\" && exit 1",
8    },
9    "keywords": [],
10   "author": "",
11   "license": "ISC",
12   "dependencies": {
13     "express": "^4.17.1"
14   }
15 }
```

```
tiagoboeing@TIAGOBOEING MINGW64 ~/Desktop/docker-build
$ npm install express --save
npm WARN docker-build@1.0.0 No description
npm WARN docker-build@1.0.0 No repository field.

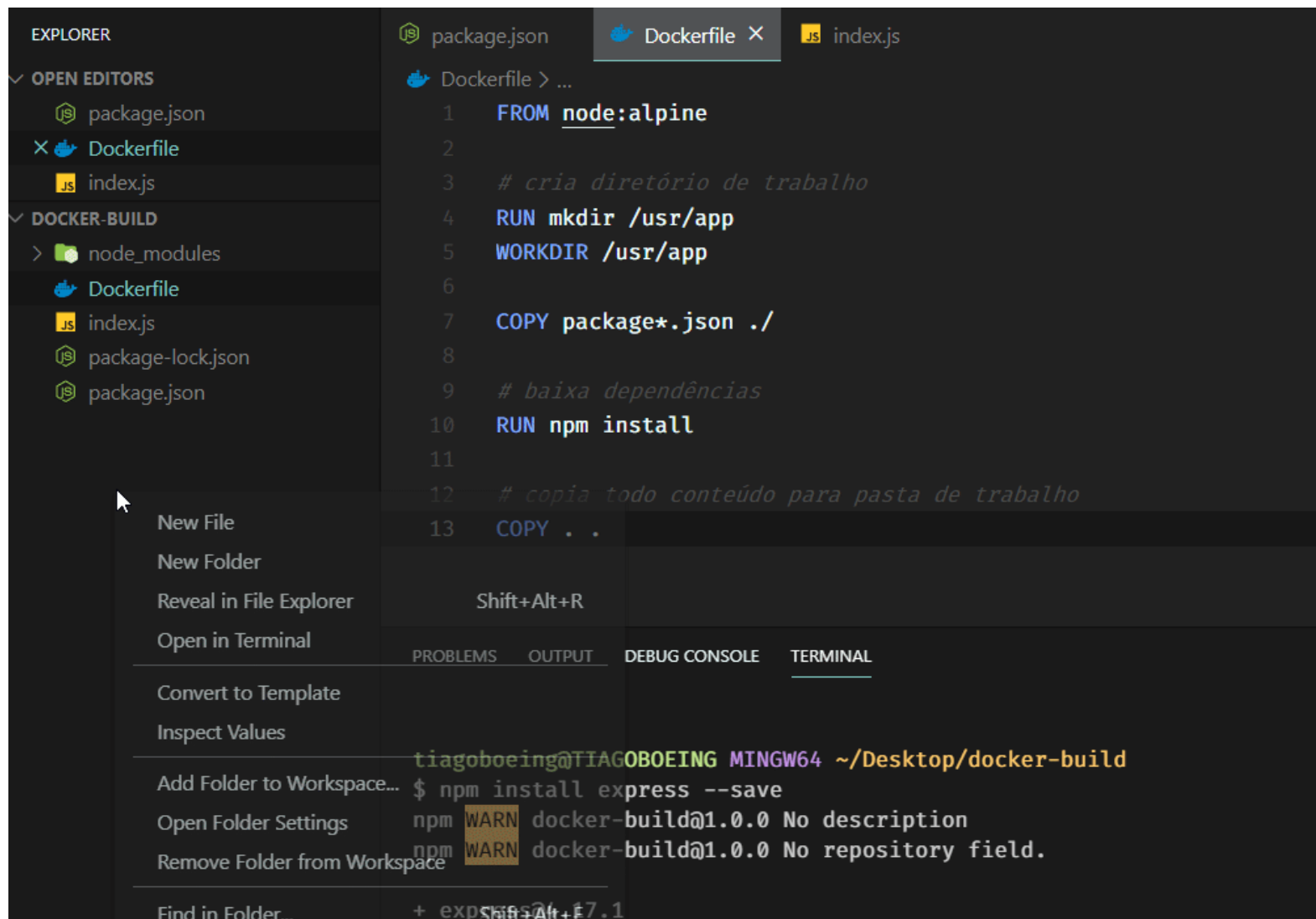
+ express@4.17.1
updated 1 package and audited 50 packages in 3.427s
found 0 vulnerabilities
```

# Criar Dockerfile

Conteúdo do arquivo se encontra no artigo







The screenshot shows the Visual Studio Code interface. The Explorer sidebar on the left shows a project structure with files like package.json, Dockerfile, and index.js. The Dockerfile is open in the editor, showing a Dockerfile with the following content:

```
1 FROM node:alpine
2
3 # cria diretório de trabalho
4 RUN mkdir /usr/app
5 WORKDIR /usr/app
6
7 COPY package*.json ./
8
9 # baixa dependências
10 RUN npm install
11
12 # copia todo conteúdo para pasta de trabalho
13 COPY . .
```

A context menu is open over the Dockerfile, showing options like 'New File', 'New Folder', 'Reveal in File Explorer', 'Open in Terminal', 'Convert to Template', 'Inspect Values', 'Add Folder to Workspace...', 'Open Folder Settings', 'Remove Folder from Workspace', and 'Find in Folder...'. The 'Open in Terminal' option is highlighted.

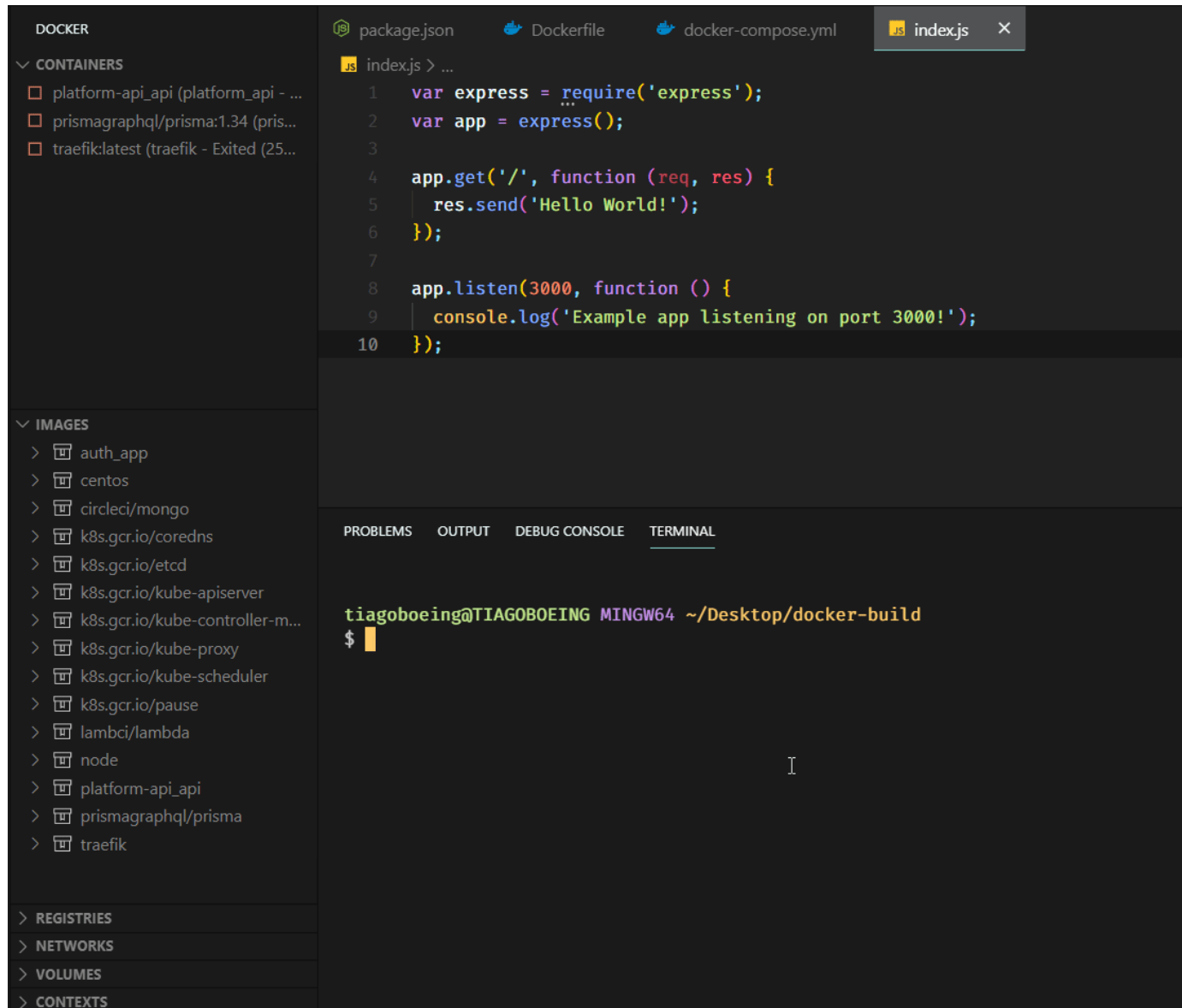
The terminal at the bottom shows the command `$ npm install express --save` and the output:

```
tiagoboering@TIAGOBOEING MINGW64 ~/Desktop/docker-build
$ npm install express --save
npm WARN docker-build@1.0.0 No description
npm WARN docker-build@1.0.0 No repository field.
+ express@4.17.1
```

# Criar docker-compose.yml

Conteúdo do arquivo se encontra no artigo

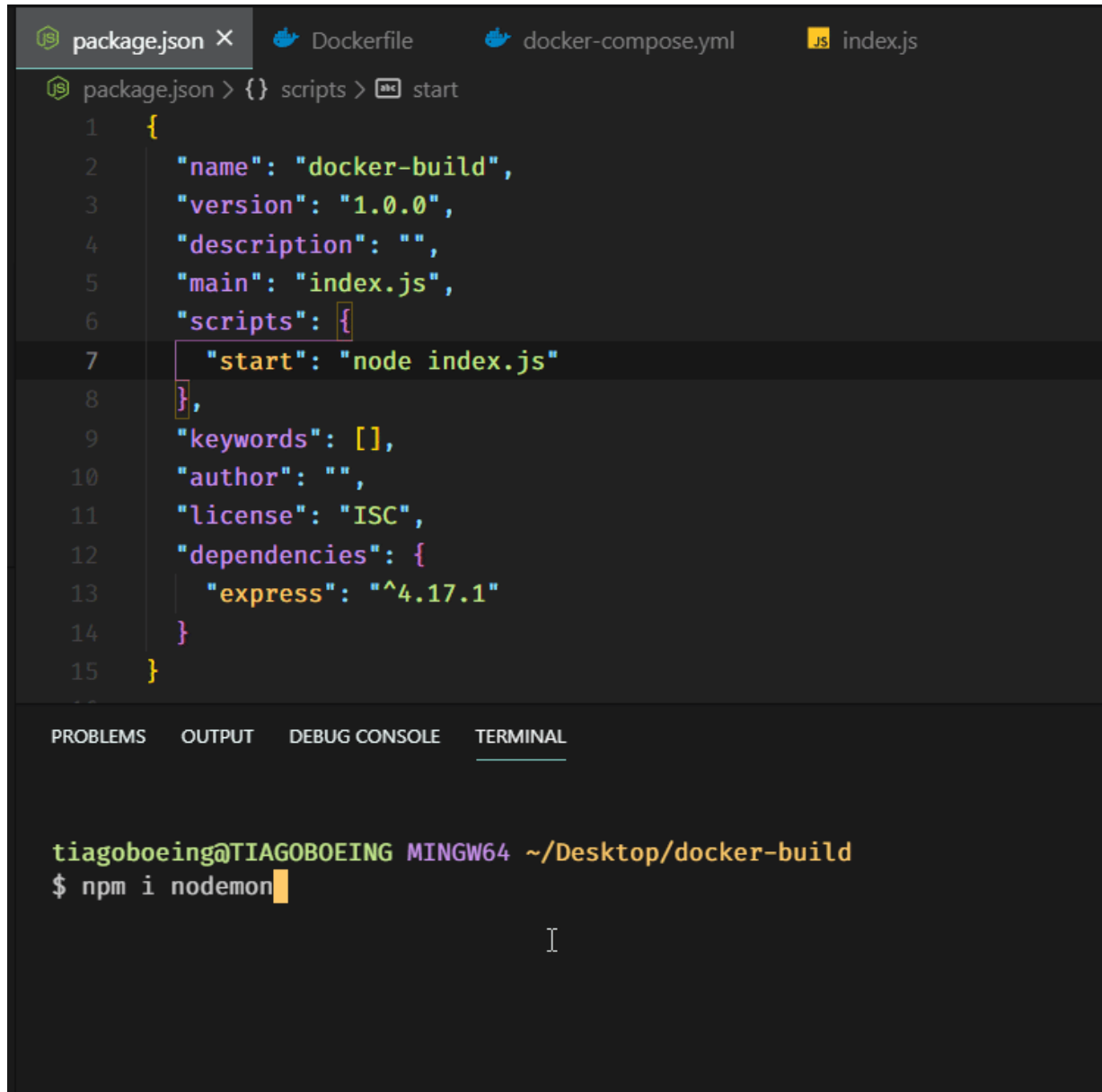




## Rodar contêiner pela primeira vez

`docker-compose up -d`





The image shows a VS Code editor with a dark theme. At the top, there are four tabs: 'package.json' (active), 'Dockerfile', 'docker-compose.yml', and 'index.js'. The 'package.json' tab displays the following JSON content:

```
1  {
2    "name": "docker-build",
3    "version": "1.0.0",
4    "description": "",
5    "main": "index.js",
6    "scripts": {
7      "start": "node index.js"
8    },
9    "keywords": [],
10   "author": "",
11   "license": "ISC",
12   "dependencies": {
13     "express": "^4.17.1"
14   }
15 }
```

Below the editor, the 'TERMINAL' tab is active, showing a command prompt with the following text:

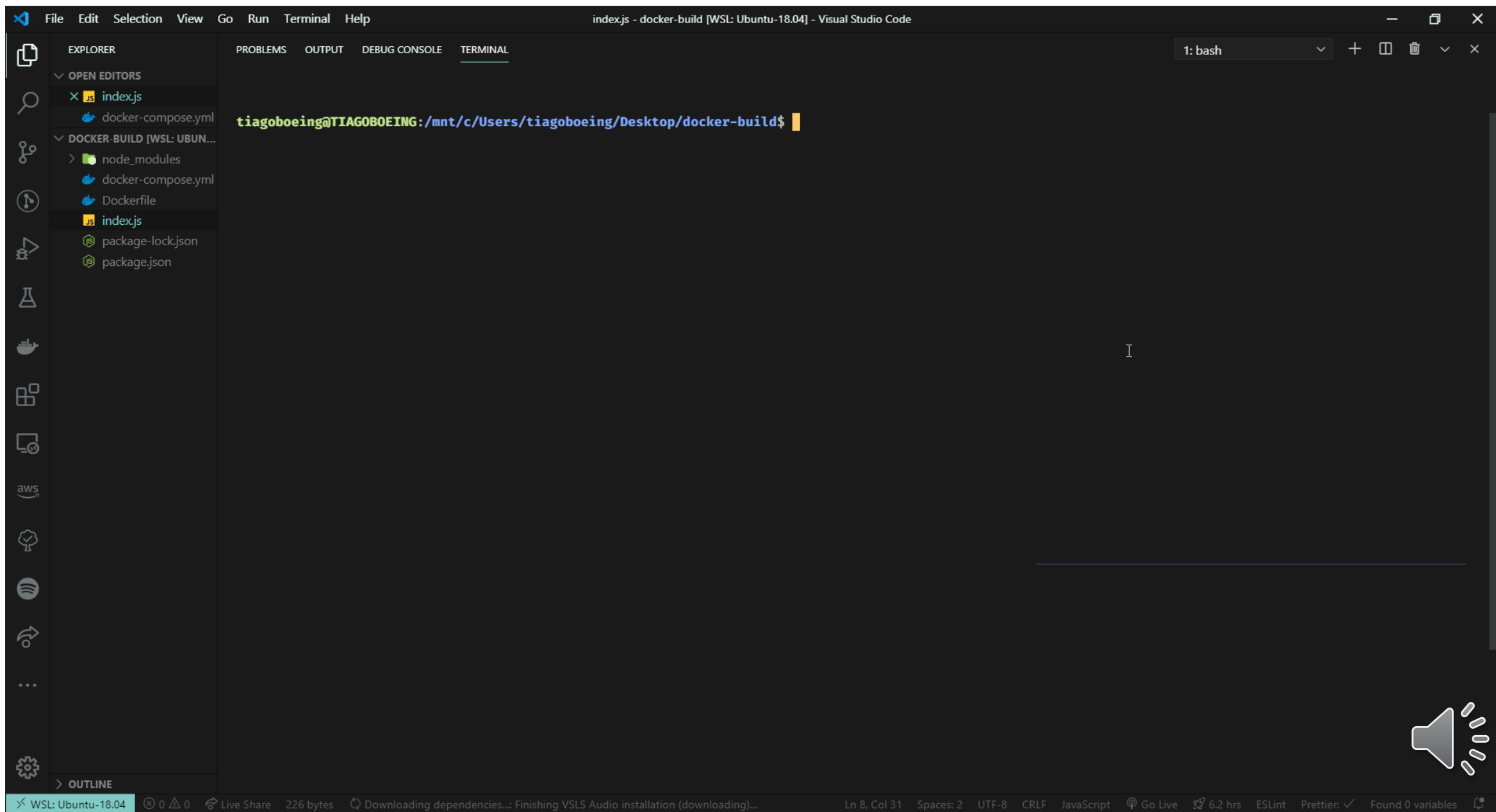
```
tiagoboeing@TIAGOBOEING MINGW64 ~/Desktop/docker-build
$ npm i nodemon
```

A cursor is visible on the line following the command.

## Nodemon para restart automático

`npm i nodemon`





**docker-compose up**

FileEditSelectionViewGoRunTerminalHelpindex.js - docker-build [WSL: Ubuntu-18.04] - Visual Studio Code

EXPLORER

OPEN EDITORS

index.js

Dockerfile

docker-compose.yml

DOCKER-BUILD [WSL: UBUN...]

node\_modules

docker-compose.yml

Dockerfile

index.js

package-lock.json

package.json

OUTLINE

index.js

app.listen() callback

```
1 var express = require('express');
2 var app = express();
3
4 app.get('/', function (req, res) {
5   res.send('Hello World!');
6 });
7
8 app.listen(3000, function () {
9   console.log('Example app listening on port 3000! :)');
10 });
11
```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

1: bash

tiagoboeing@TIAGOBOEING:/mnt/c/Users/tiagoboeing/Desktop/docker-build\$ docker-compose up

WSL: Ubuntu-18.04

0 0

Live Share

226 bytes

Ln 9, Col 57

Spaces: 2

UTF-8

CRLF

JavaScript

Go Live

6.2 hrs

ESLint

Prettier: ✓

Found 0 variables

# Dockerfile

```
FROM node:alpine
```

```
# cria diretório de trabalho
```

```
RUN mkdir /usr/app
```

```
WORKDIR /usr/app
```

```
COPY package*.json ./
```

```
# baixa dependências
```

```
RUN npm install
```

```
# copia todo conteúdo para pasta de trabalho
```

```
COPY . .
```



# docker-compose.yml

```
version: "3.4"
```

```
services:
```

```
  app:
```

```
    build: . # indica pasta raiz para buscar Dockerfile
```

```
    command: npm start # script no package.json
```

```
    ports:
```

```
      - "3000:3000" # bind entre portas
```

```
    volumes:
```

```
      - ./usr/app # workdir informado no Dockerfile
```



# Referência

---

What is Docker? (2020).

<https://opensource.com/resources/what-docker>, [accessed on May 9].