

# Node.js e Docker no Ambiente de Desenvolvimento

Página Inicial / Node.js / Node.js e Docker no Ambiente de Desenvolvimento

 Node.js, Linux Containers  1/29/2021  Gabriel Willemann

## Node.js + Docker: Development Environment

Você sabe que instalar e configurar um ambiente de desenvolvimento parece fácil, mas nem sempre é. Quando você percebe seu software possui várias **dependências** e seu ambiente de desenvolvimento se tornou bem complexo.

O **Docker** é uma ótima opção para resolver este tipo de problema, uma vez que ele vai empacotar tudo que você precisa em um **ambiente virtualizado**.


Quando eu digo "ambiente virtualizado" não estou me referindo a uma máquina de virtual clássica. O Docker funciona um pouco diferente. Se você quer saber mais sobre Docker e seus conceitos leia o seguinte artigo: [Docker e seus conceitos](#).

Desta forma, o objetivo deste artigo é ensinar como um preparar um **ambiente de desenvolvimento** para um **servidor HTTP** escrito com **Node.js**.

## O que você precisa instalar

Para seguir este tutorial, você só precisa ter no seu computador os seguintes programas:

- **Docker**
- **Terminal**
- Seu **editor de código** favorito

Se você tem dúvidas sobre como instalar o Docker, você encontrará todas as [instruções de instalação na documentação oficial](#) .

## Node.js + Docker: Como criar o projeto

Primeiramente você deve criar uma pasta no seu computador. No meu caso vou criar a pasta: **/home/project-node-docker**

Lembrando que eu estou utilizando o **Linux Ubuntu 20.04**.

Acesse essa pasta pelo terminal e execute o seguinte comando:

```
docker run --rm --volume "/home/project-node-docker:/srv/app" --workdir "/srv/app" sh
```

Com este comando você executará um **container** baseado na imagem [node:12](#) e se conectará neste container via terminal.

Sobre os demais parâmetros deste comando, segue uma breve explicação:

- **--volume "/home/project-node-docker:/srv/app"**: Link entre a pasta **/home/project-node-docker** do computador hospedeiro com a pasta **/srv/app** do container.
- **--workdir "/srv/app"**: Diretório inicial quando o container é iniciado.
- **--publish 3000:3000**: Link entre a porta 3000 do computador hospedeiro com a porta 3000 do container.
- **--rm**: Exclui antigos containers.
- **-it**: Link entre o terminal do computador hospedeiro com o output do container.

Agora que você está conectado ao container, você pode inicializar o projeto:

```
npm init -y
```

Vamos instalar dois pacotes, o **express.js** que será o nosso servidor HTTP e o **nodemon** que fará o monitoramento de alterações de arquivos (algo muito útil durante o desenvolvimento).

```
npm install express
npm install nodemon
```

Aqui há algo muito importante para mencionar. As instalações que você faz quando está conectado ao container serão perdidas quando o container for reiniciado.

Porém as instalações que fizemos aqui, foram feitas dentro da pasta **"/srv/app"** que é compartilhada com o computador hospedeiro. Ou seja, tudo que é criado dentro desta nós não perderemos.

Mas fique atento a outros tipos de instalações, pois talvez seja necessário criar uma imagem com o [Docker File](#).

## Node.js + Docker: Desenvolvimento

Antes de iniciar o desenvolvimento, você precisa se atentar que vários arquivos foram criados por dentro do container e talvez você tenha que alterar o Owner das pastas e arquivos.

Se você estiver usando **Linux**, basta executar este comando no computador hospedeiro:

```
sudo chown -R <USERNAME> /home/project-node-docker/
```

sh

Continue no computador hospedeiro e abra seu editor de código favorito. Crie o arquivo `/home/project-node-docker/index.js`:

```
const express = require('express');
const app = express();
const port = 3000;

app.get('/', (req, res) => {
  res.send('Hello World!');
});

app.listen(port, () => {
  console.log(`Example app listening at http://localhost:${port}`);
});
```

js

Agora se conecte novamente ao container:

```
docker run --rm --volume "/home/project-node-docker:/srv/app" --workdir "/srv/app"
```

sh

E dentro do container, execute o seguinte comando:

```
npx nodemon index.js
```

sh

Pronto! Projeto sendo executado. Para testar acesse o endereço **http://localhost:3000** no navegador do seu computador.

## Node.js + Docker Compose

Com o passar do tempo você pode se cansar de executar o comando **Docker Run** toda vez que precisar iniciar o **container**.

Para resolver isso você pode utilizar o **Docker Compose**. No computador hospedeiro, crie o arquivo `/home/project-node-docker/docker-compose.yml` com o seguinte conteúdo:

```
version: '3'

services:
  node:
    image: node:12
```

```
ports:
  - '3000:3000'
volumes:
  - ./:/srv/app
working_dir: /srv/app
command: 'npx nodemon index.js '
```

Ainda no computador hospedeiro, execute o seguinte comando via terminal:

```
docker-compose up -d
```

sh

Pronto! Container sendo executado em background.

Caso você precise se conectar ao container via terminal, você primeiramente deve descobrir o ID do container com o seguinte comando:

```
docker ps
```

sh

Feito isso, agora execute mais um comando:

```
docker exec -it <CONTAINER_ID> bash
```

sh

Pronto! Conectado ao container.

## Por fim

Neste tutorial eu utilizei as seguintes versões:

- [Docker](#): 19.03
- [Docker Compose](#): 1.25
- [Node](#): 12.20
- [Npm](#): 6.14
- [Express](#): 4.17
- [Nodemon](#): 2.0

Dúvidas ou sugestões é só entrar em contato. Abraço.

## Autor

---

Gabriel Willemann



Software Developer

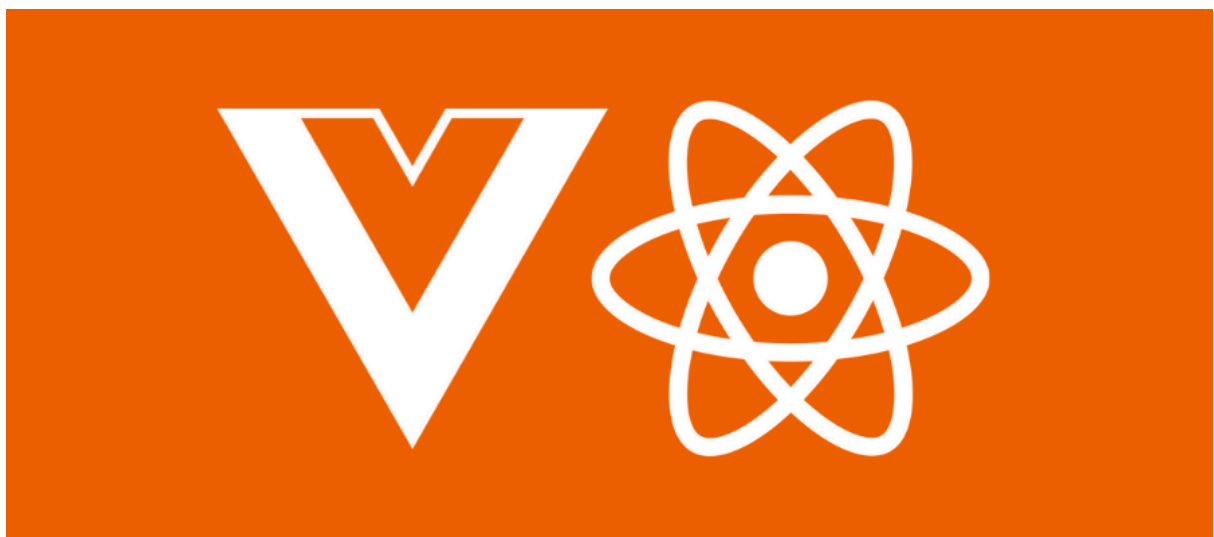
## Acesse



## Postagens Populares



Vue.js + PWA + Workbox: Aplicativo Off-line com Background-Sync



## Para Programadores Vue.js: Comparação com React

---



Aprenda a Criar um Blog com Vuepress

---



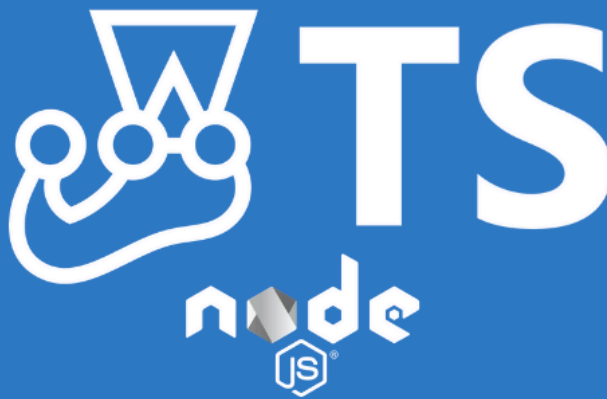
Vue.js + Prerender: Melhorando o SEO do seu SPA

---



Vue.js + Quasar + Mobile App: Um caso real

---



Testes Unitários com Node.js, Jest e TypeScript



Ruby on Rails + Puma + Nginx: Proxy Reverso com Unix Socket

"Any fool can write code that a computer can understand. Good programmers write code that humans can understand." Martin Fowler