

A nota que você procura foi excluída.

Esta história exclusiva para membros é por nossa conta. [Atualize](#) para acessar todo o Medium.

◆ História exclusiva para membros

Como servir aplicativos angulares com NGINX e Docker

Um guia passo a passo com um exemplo



Bhargav Bachina · [Seguir](#)

3

Publicado em Laboratórios Bachina

6 minutos de leitura · 20 de dezembro de 2019

Ouvir

Compartilhar

••• Mais

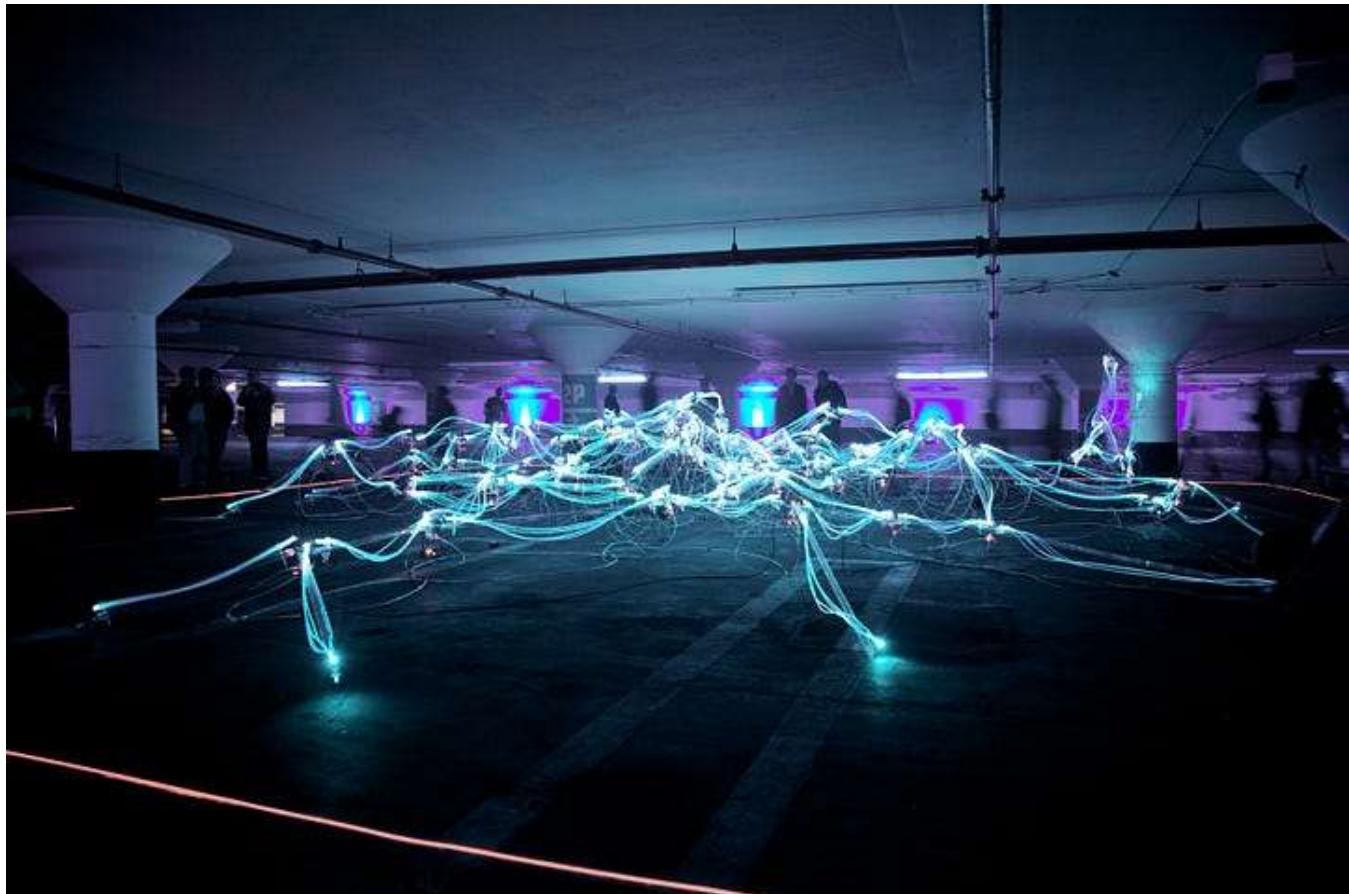


Foto de [Marius Masalar](#) no [Unsplash](#)

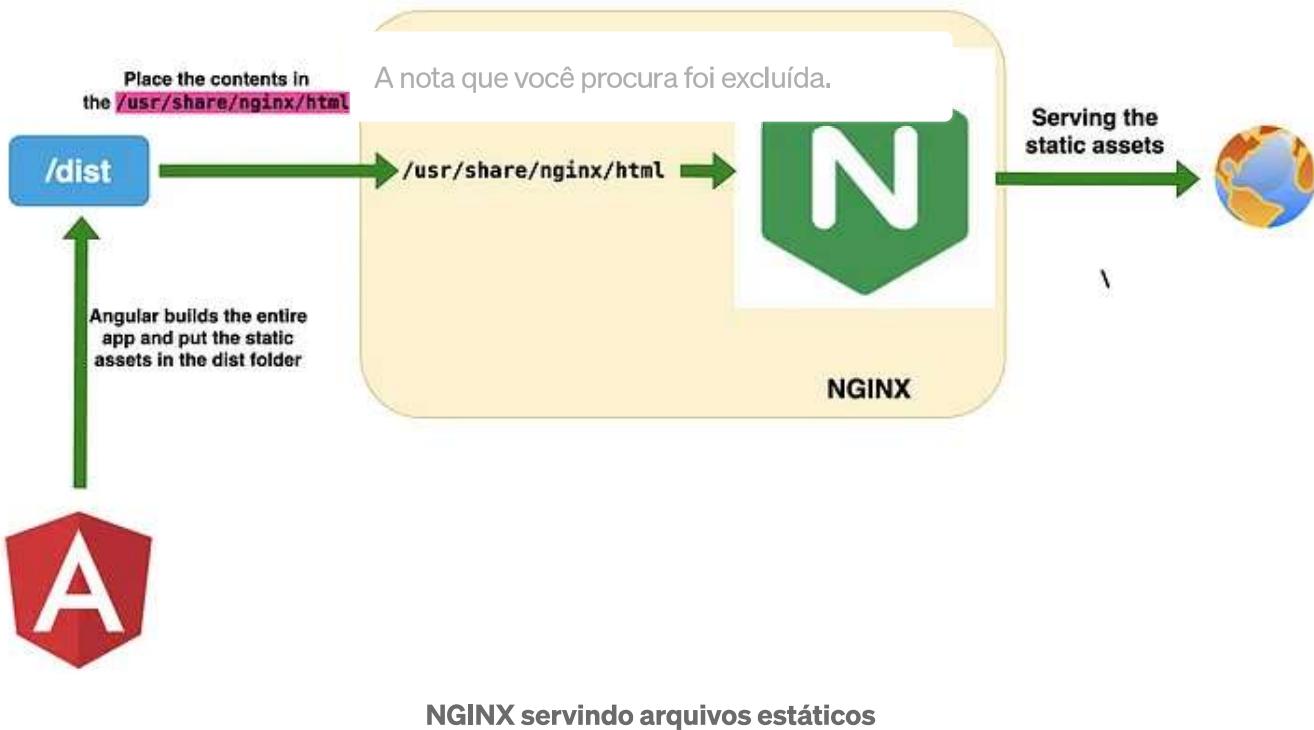
Há muitas maneiras de ~~criar aplicativos Angular e enviá-los para produção. Uma maneira é construir An~~ A nota que você procura foi excluída. ~~Outra maneira é construir o angular e servir esse conteúdo estático com o servidor web NGINX. Quando construímos com NGINX e docker, não precisamos lidar com o tempo de execução do servidor ou com o código relacionado ao servidor. Tudo o que precisamos para construir o aplicativo Angular para produção e servir o conteúdo estático gerado com o servidor NGINX.~~

Neste post, veremos os detalhes e a implementação da segunda abordagem. Iremos passo a passo com um exemplo.

- *Introdução*
- *Pré-requisitos*
- *Exemplo de projeto*
- *NGINX suficiente para este projeto*
- *Implementação*
- *Resumo*
- *Conclusão*

Introdução

Neste projeto, usaremos Angular como framework JS, NGINX como servidor web, docker como tempo de execução de contêiner.



Se observarmos o diagrama acima, o Angular cria o aplicativo e coloca os ativos estáticos na `/dist` pasta. Colocamos esses ativos no local padrão do NGINX, `/usr/share/nginx/html` de onde ele veicula o conteúdo da web.

Pré-requisitos

Existem alguns pré-requisitos para realizar esta tarefa. Temos que executar o NGINX na janela de encaixe e colocar os ativos estáticos no NGINX e executar toda a configuração dentro da janela de encaixe. Precisamos instalar o nodejs para a configuração da API. Instale todas as ferramentas abaixo para acompanhar este tutorial ou você deseja executá-lo em sua máquina.

- [NodeJS](#)
- [CLI angular](#)
- [Docker para área de trabalho](#)
- [NGINX \(é necessário instalar dentro do docker\)](#)

Exemplo de projeto

Este é um projeto simples que demonstra o atendimento de aplicativos Angular estáticos com NGINX e A nota que você procura foi excluída. com uma tabela que obtém alguns dados do servidor.

Welcome to envapp!

Environment Configuration

#	Name	Version	Environment	Base HREF
0	envapp	1.0.0	development	envapp
1	envapp	1.0.0	test	envapp
2	envapp	1.0.0	uat	envapp
3	envapp	1.0.0	production	envapp

Exemplo de aplicativo Angular

Aqui está o projeto de exemplo onde você pode clonar e executar em sua máquina

```
// clonar o projeto
git clone https://github.com/bbachi/angular-nginx-docker.git

//instala e inicia as dependências
npm install
npm start

// constrói a imagem do docker
docker build -t appui .

// executa o aplicativo
docker run -d --name appui -p 80:80 appui
```

Este é um aplicativo Angular simples que mostra a tabela e obtém os dados app.config.json da pasta de ativos do aplicativo. Aqui estão os arquivos app.component.html e app.component.ts.

```
1  <!--The content below is only a placeholder and can be replaced.-->
2  <div style="text-align:center">
3    <h1>
4      Welcome to {{ title }}!
5    </h1>
6  </div>
```

```

7  <div class="tblclas">
8    <h2>Environment Configuration</h2> A nota que você procura foi excluída.
9    <table *ngIf="appData">
10      <thead class="thead-dark">
11        <tr>
12          <th scope="col">#</th>
13          <th scope="col">Name</th>
14          <th scope="col">Version</th>
15          <th scope="col">Environment</th>
16          <th scope="col">Base HREF</th>
17        </tr>
18      </thead>
19      <tbody>
20        <tr *ngFor="let data of appData; let i = index">
21          <th scope="row">{{i}}</th>
22          <td>{{data.name}}</td>
23          <td>{{data.version}}</td>
24          <td>{{data.environment}}</td>
25          <td>{{data.basehref}}</td>
26        </tr>
27      </tbody>
28    </table>
29  </div>

```

app.component.html hosted with ❤ by GitHub

[view raw](#)

```

1  import { Component } from '@angular/core';
2  import { AppService } from './app.service'
3
4  @Component({
5    selector: 'app-root',
6    templateUrl: './app.component.html',
7    styleUrls: ['./app.component.css']
8  })
9  export class AppComponent {
10    title = 'envapp';
11    appData: any;
12
13    constructor(private appService: AppService) {
14      this.appService.getConfig().subscribe(data => {
15        this.appData = data;
16      })
17    }
18  }

```

app.component.ts hosted with ❤ by GitHub

[view raw](#)

Componente do aplicativo

Aqui está o arquivo de serviço que lê o arquivo `app.config.json` da pasta de ativos.

```

1  [
2      {
3          "name": "envapp",
4              "version": "1.0.0",
5                  "environment": "development",
6                      "basehref": "envapp"
7      },
8      {
9          "name": "envapp",
10             "version": "1.0.0",
11                 "environment": "test",
12                     "basehref": "envapp"
13     },
14     {
15         "name": "envapp",
16             "version": "1.0.0",
17                 "environment": "uat",
18                     "basehref": "envapp"
19     },
20     {
21         "name": "envapp",
22             "version": "1.0.0",
23                 "environment": "production",
24                     "basehref": "envapp"
25     }
26 ]

```

app.config.json hosted with ❤ by GitHub

[view raw](#)

```

1  import { Injectable } from '@angular/core';
2  import { HttpClient } from '@angular/common/http';
3
4  @Injectable({
5      providedIn: 'root'
6  })
7  export class AppService {
8
9      constructor(private http: HttpClient) { }
10
11      configUrl = 'envapp/assets/app.config.json';
12
13      getConfig() {
14          return this.http.get(this.configUrl);
15      }
16  }

```

app.service.ts hosted with ❤ by GitHub

[view raw](#)

Serviço lendo arquivo json de configuração da pasta de ativos

A nota que você procura foi excluída.

NGINX suficiente para este projeto

Não estamos analisando tudo sobre o NGINX aqui e apenas analisamos o suficiente para este projeto. Se você já estiver familiarizado com esse assunto, pode pular para a próxima seção.

Os processos NGINX são divididos em um processo mestre e vários processos de trabalho. O processo mestre se encarrega de avaliar a configuração e manter os processos de trabalho e os processos de trabalho cuidam das solicitações reais. Podemos definir o número de processos de trabalho no arquivo de configuração que pode ser colocado no diretório `/usr/local/etc/nginx`,

`/etc/nginx` ou `/usr/local/nginx/conf`.

O arquivo de configuração consiste em diretivas que formam os módulos ou contextos. Existem dois tipos de diretivas: diretivas simples e diretivas de bloco. Uma diretiva simples possui nomes e parâmetros separados por um espaço e termina com um ponto e vírgula como este `listen 80;`. Uma diretiva de bloco é a mesma, mas contém informações adicionais e está rodeada por chaves como esta `{ listen 80; root /usr/share/nginx/html; }`.

Vamos entender o arquivo de configuração NGINX que usamos neste projeto.

Abaixo está o arquivo `nginx.conf` localizado na pasta `.nginx` no local raiz do projeto.

Tudo é um contexto no arquivo de configuração. Temos um contexto hierárquico que começa com o contexto principal. Por exemplo, **processos_trabalhadores** e **eventos** são definidos no contexto principal e outro contexto começa com **http**. Temos outro contexto dentro de **http** chamado **servidor** que escuta na porta 80 e atende ativos estáticos a partir do local raiz `/usr/share/nginx/html`.

Podemos ter múltiplas declarações do **servidor** dentro do contexto **http** e podemos ter múltiplas declarações de **localização** dentro do contexto do **servidor**.

```
1 worker_processes 4;
2
3 events { worker_connections 1024; }
4
5 http {
```

```

1      http {
2
3          server {
4              listen 80;      A nota que você procura foi excluída.
5              root /usr/share/nginx/html;
6              include /etc/nginx/mime.types;
7
8
9          location /appui {
10            try_files $uri /envapp/index.html;
11        }
12    }
13  }
14 }
15 }
```

nginx.conf hosted with ❤ by GitHub

[view raw](#)

nginx.conf

Implementação

Usamos Docker como tempo de execução de contêiner para este projeto. Estamos usando compilações de vários estágios para construir a imagem Docker. Aqui está o Dockerfile do projeto.

```

1  # stage1 as builder
2  FROM node:10-alpine as builder
3
4  # copy the package.json to install dependencies
5  COPY package.json package-lock.json ./
6
7  # Install the dependencies and make the folder
8  RUN npm install && mkdir /app-ui && mv ./node_modules ./app-ui
9
10 WORKDIR /app-ui
11
12 COPY . .
13
14 # Build the project and copy the files
15 RUN npm run ng build -- --deploy-url=/envapp/ --prod
16
17
18 FROM nginx:alpine
19
20 #!/bin/sh
21
22 COPY ./nginx/nginx.conf /etc/nginx/nginx.conf
23
```

```

24 ## Remove default nginx index page
25 RUN rm -rf /usr/share/, A nota que você procura foi excluída.
26
27 # Copy from the stahg 1
28 COPY --from=builder /app-ui/dist /usr/share/nginx/html
29
30 EXPOSE 4200 80
31
32 ENTRYPOINT ["nginx", "-g", "daemon off;"]

```

Dockerfile hosted with ❤ by GitHub

[view raw](#)

Dockerfile

Estágio 1

Estamos usando `node:10-alpine` como imagem base para o stage1 e copiando `package.json` para instalar todas as dependências. Em seguida, copiamos o projeto restante posteriormente, dessa forma podemos pular a instalação das dependências sempre que houver uma alteração nos arquivos. Docker usa cache para construir a imagem a partir de camadas existentes se não houver alteração.

Construímos o projeto com `deploy-url /envapp/` e todos os arquivos estáticos construídos são colocados na pasta `/dist`.

Estágio 2

O estágio 2 começa com a imagem base `nginx:alpine` e copia o arquivo `nginx.conf`, remove o arquivo de índice do local raiz e, finalmente, copia todos os arquivos do estágio 1 para o local raiz de onde pode servir o conteúdo.

Construa a imagem e execute o projeto

Vamos construir o projeto com este comando `docker build -t app-ui .` e você pode executar o projeto com este comando `docker run -d --name app-ui -p 80:80 app-ui`. Você pode executar o aplicativo em `http://localhost:80/appui`



Environment Configuration

#	Name	Version	Environment	Base HREF
0	envapp	1.0.0	development	envapp
1	envapp	1.0.0	test	envapp
2	envapp	1.0.0	uat	envapp
3	envapp	1.0.0	production	envapp

Projeto em execução no localhost

Coisas importantes a serem lembradas:

A nota que você procura foi excluída.

A porta do contêiner e a porta de escuta do nginx devem ser as mesmas, que é 80, caso contrário você obteria **ERR_EMPTY_RESPONSE** ao executar o projeto.

```
// porta do contêiner
docker run -d --name appui -p 80:80 appui

// configuração nginx

http{

    servidor {
        ouvir 80;
    }
}
```

Devemos incluir esta diretiva no arquivo nginx.conf, caso contrário todos os estilos serão renderizados como texto simples no navegador.

```
inclusa /etc/nginx/mime.types;
```

Exec no contêiner em execução

Enquanto o contêiner está em estado de execução, podemos executá-lo e ver o conteúdo do sistema de arquivos.

```
docker exec -it appui /bin/sh
```

Na verdade, podemos ver todo o conteúdo em /usr/share/nginx/html

```
/ # cd /usr/share/nginx/html
/usr/share/nginx/html # ls
envapp
/usr/share/nginx/html # cd envapp/
/usr/share/nginx/html/envapp # ls
3rdpartylicenses.txt      favicon.ico          polyfills.8bbb231b43165d65d357.js
assets                   index.html           runtime.e951f52a329703759934.js
es2015-polyfills.b0657154bc33c6ff11ae.js main.514b0641425e63253f38.js
styles.52dd26ce13c05e1a50ca.css
```

Sistema de arquivos dentro do docker

Resumo

A nota que você procura foi excluída.

- O NGINX pode ser usado como um servidor web ou proxy reverso para servir o conteúdo estático.
- Toda a configuração do NGINX pode ser colocada neste arquivo `nginx.conf`.
- Precisamos construir o aplicativo angular e colocar todos os arquivos estáticos no local raiz do NGINX para servir a web.
- Docker é usado como tempo de execução do contêiner.
- Usamos compilações em vários estágios para reduzir o tamanho final da imagem e remover arquivos desnecessários do ambiente de produção.
- A imagem Docker pode ser construída com `docker build -t app-ui .`
- Execute o contêiner com este comando `docker run -d --name appui -p 80:80 app-ui`.
- É muito importante combinar as portas durante a execução do contêiner e a porta de escuta no arquivo `nginx.conf`. Caso contrário, você receberá um erro **ERR_EMPTY_RESPONSE**.

Abrir no aplicativo ↗



Conclusão

NGINX é um servidor web de alto desempenho que fornece entrega de conteúdo e aplicativos, melhora a segurança e facilita a disponibilidade e escalabilidade dos aplicativos web. Se você quiser evitar ou não houver necessidade de construir aplicativos de UI com Java ou tempo de execução do servidor node js, você pode construir o aplicativo de UI e servir os arquivos estáticos com o NGINX em escala. No próximo post, podemos explorar como interagir com a API.

JavaScript

Angular

Desenvolvimento web

Programação

Desenvolvimento de software

A nota que você procura foi excluída.



Seguir



Escrito por Bhargav Bachina

22 mil seguidores · Editor para Laboratórios Bachina

Empreendedor | Mais de 600 artigos técnicos | Inscreva-se nos próximos vídeos

<https://www.youtube.com/channel/UCWLSuUulkLIQvbMHRUfKM-g> | <https://www.linkedin.com/in/bachina>

Mais de Bhargav Bachina e Bachina Labs



Bhargav Bachina em Laboratórios Bachina

Kubernetes – Aprenda o padrão de contêiner Sidecar

Compreendendo o padrão de contêiner Sidecar com um projeto de exemplo

◆ 6 minutos de leitura · 7 de setembro de 2020

388

2

A nota que você procura foi excluída.

...



Bhargav Bachina em Laboratórios Bachina

Pratique o suficiente com estas 150 perguntas para o exame CKAD

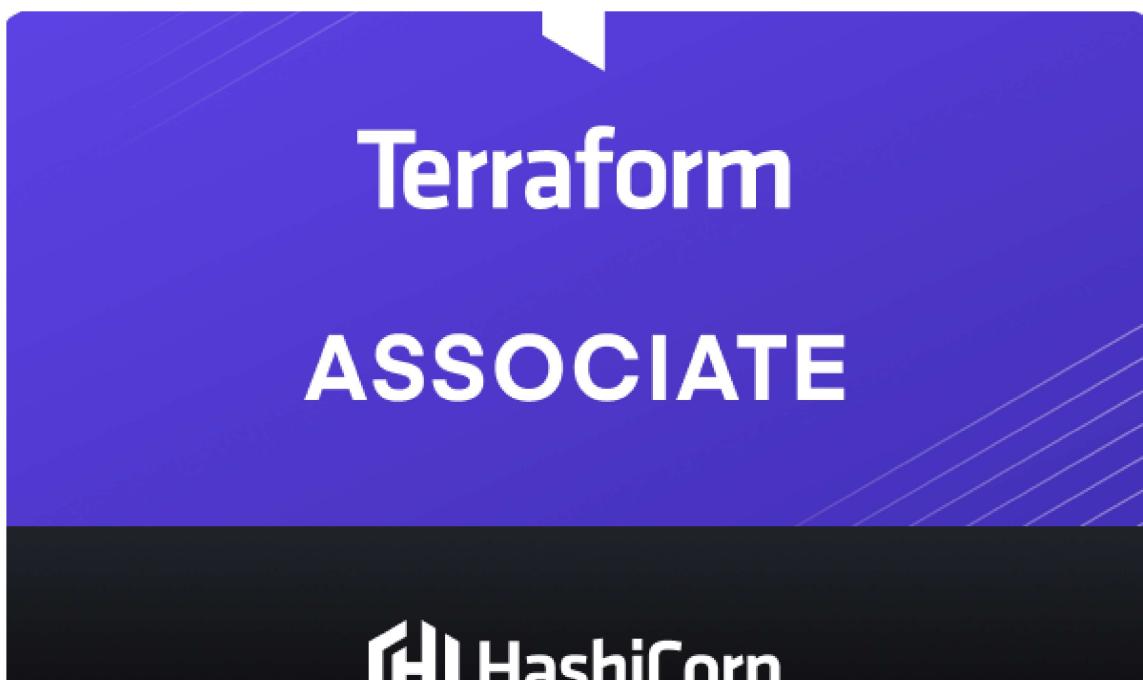
Os exercícios preparam você para o exame Certified Kubernetes Application Developer

⭐ · 21 minutos de leitura · 11 de novembro de 2019

2,2 mil

55

...





Bhargav Bachina em Laboratórios Bachina

A nota que você procura foi excluída.

250 questões práticas

associate

Leia e pratique essas questões antes do exame

◆ 47 minutos de leitura • 13 de julho de 2020

1,6K

39

...



Bhargav Bachina em Pedaços e peças

Como implementar o tempo limite de inatividade em Angular

Aprenda como usar o tempo limite de inatividade em seu aplicativo Angular para melhorar o desempenho e a segurança

6 minutos de leitura • 30 de setembro de 2019

946

13

...

[Ver tudo de Bhargav Bachina](#)[Veja tudo do Bachina Labs](#)

A nota que você procura foi excluída.

Recomendado do Médio

```
CREATE app-c/.gitignore (604 bytes)
CREATE app-c/.browserslistrc (703 bytes)
CREATE app-c/karma.conf.js (1422 bytes)
CREATE app-c/tsconfig.app.json (287 bytes)
CREATE app-c/tsconfig.spec.json (333 bytes)
CREATE app-c/src/favicon.ico (948 bytes)
CREATE app-c/src/index.html (290 bytes)
CREATE app-c/src/main.ts (372 bytes)
CREATE app-c/src/polyfills.ts (2820 bytes)
CREATE app-c/src/styles.scss (80 bytes)
CREATE app-c/src/test.ts (743 bytes)
CREATE app-c/src/assets/.gitkeep (0 bytes)
CREATE app-c/src/environments/environment.prod.ts (51 bytes)
CREATE app-c/src/environments/environment.ts (658 bytes)
CREATE app-c/src/app/app-routing.module.ts (245 bytes)
CREATE app-c/src/app/app.module.ts (393 bytes)
CREATE app-c/src/app/app.component.html (22800 bytes)
```

 Thilina Ranathunga

Implantando vários aplicativos angulares em contêineres em diferentes subdiretórios de um único...

Docker Swarm fornece uma plataforma robusta para executar serviços distribuídos em escala, e NGINX é uma ferramenta versátil para gerenciar redes...

5 minutos de leitura · 2 de julho



51



...



Minko Gechev em Blogue Angular

Angular v16 está aqui!

Há seis meses, alcançamos um marco significativo na simplicidade e experiência do desenvolvedor do Angular ao graduar as APIs independentes de...

12 minutos de leitura · 3 de maio

4,95 mil

35



...

Listas



Conhecimento geral de codificação

20 histórias · 308 salvamentos



Nunca é tarde ou cedo para começar algo

15 histórias · 108 salvamentos



Codificação e Desenvolvimento

11 histórias · 158 salvamentos

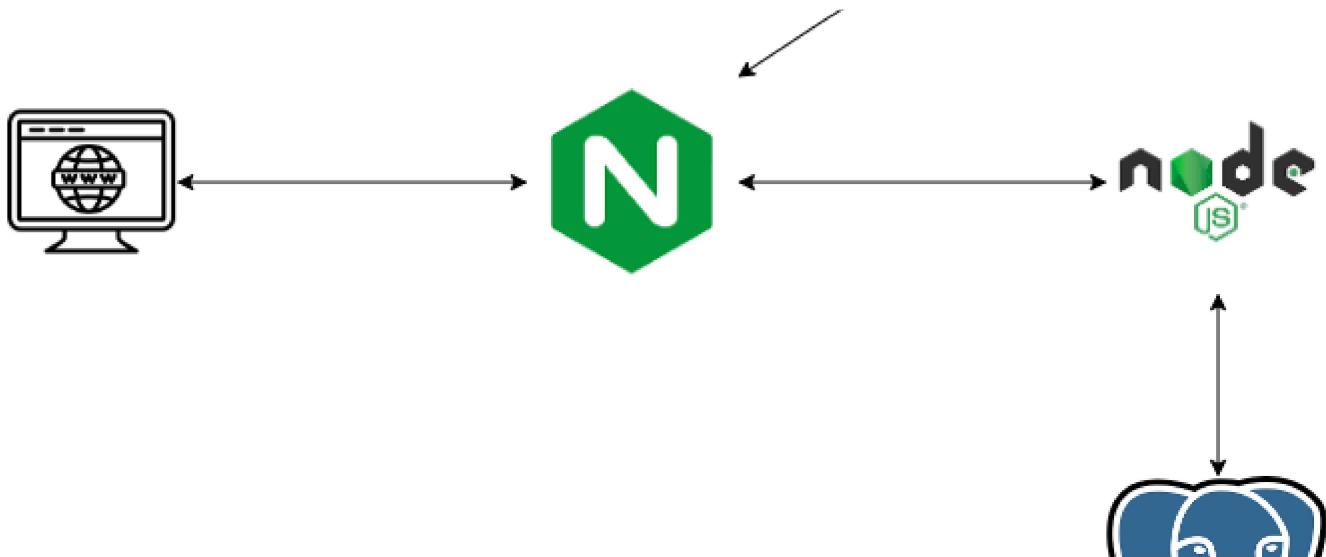


Histórias para ajudá-lo a crescer como desenvolvedor de software

19 histórias · 348 salvamentos



A nota que você procura foi excluída.



Antonio Maccarini

Dockerize um aplicativo React com Node.js, Postgres e Nginx

Este tutorial explica como encaixar um aplicativo React com Node.js, Postgres e Nginx. Ele fornece instruções passo a passo sobre...

11 minutos de leitura · 8 de junho

58

...



Macdonald Chika

Como instalar TLS/SSL no contêiner Docker Nginx com Let's Encrypt

Descubra como proteger seu aplicativo com SSL/TLS usando o certificado Let's Encrypt.

A nota que você procura foi excluída.

5 minutos de leitura · 9 de junho



31



...



Patric em Pedaços e peças

10 erros comuns no desenvolvimento angular

Desenvolva aplicativos de alto desempenho, robustos e seguros

29 minutos de leitura · 18 de abril



496



12



...

A nota que você procura foi excluída.



Aphinya Dechalert em Melhor programação

Você deve usar o Angular em 2023?

Onde está a estrutura agora?

◆ · 8 minutos de leitura · 4 de abril

1K

24



...

Veja mais recomendações