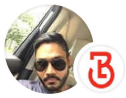
✨ Member-only story

How To Serve Angular Application With NGINX and Docker

A step by step guide with an example



Bhargav Bachina · Follow
Published in Bachina Labs
6 min read · Dec 20, 2019

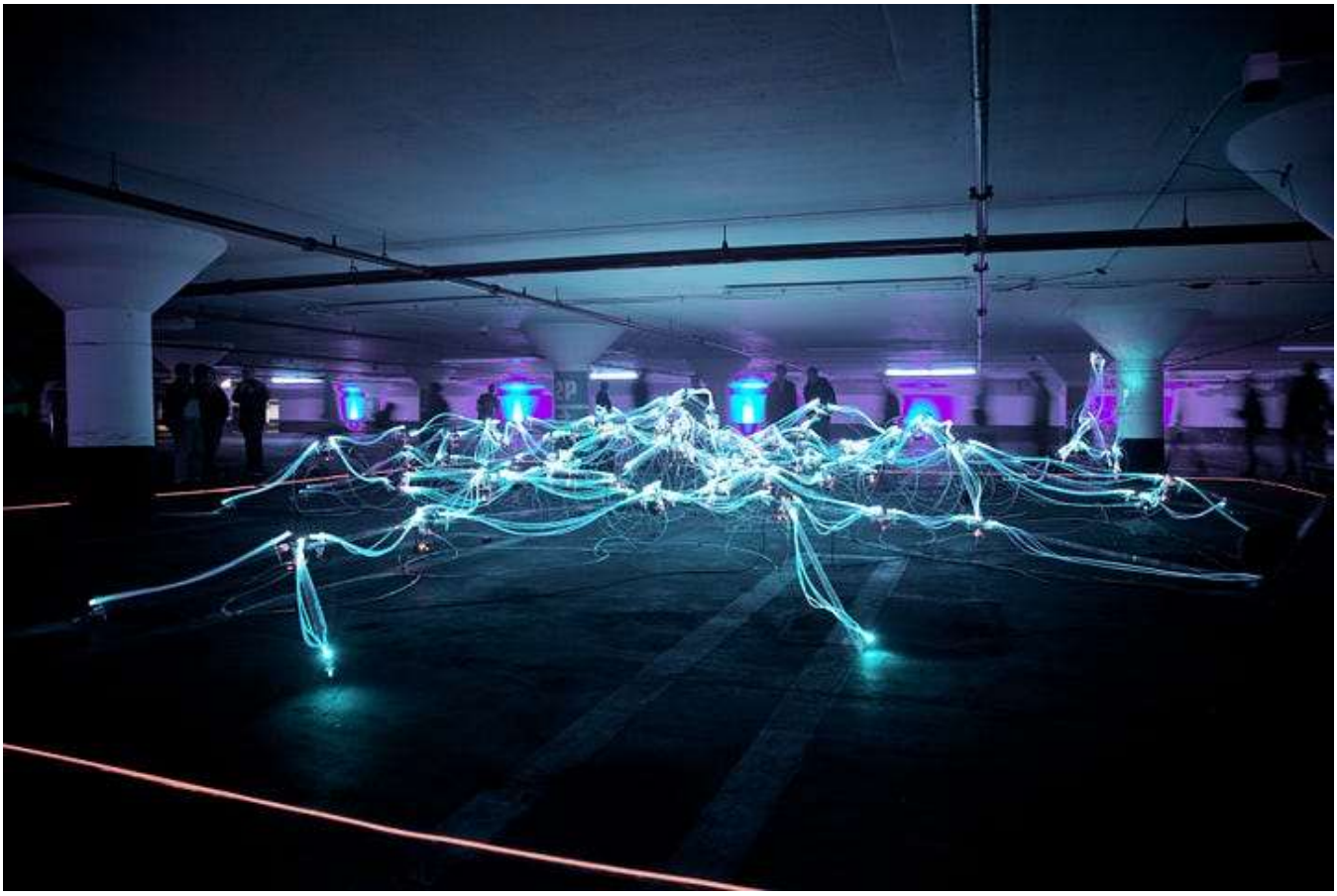


Photo by Marius Masalar on Unsplash

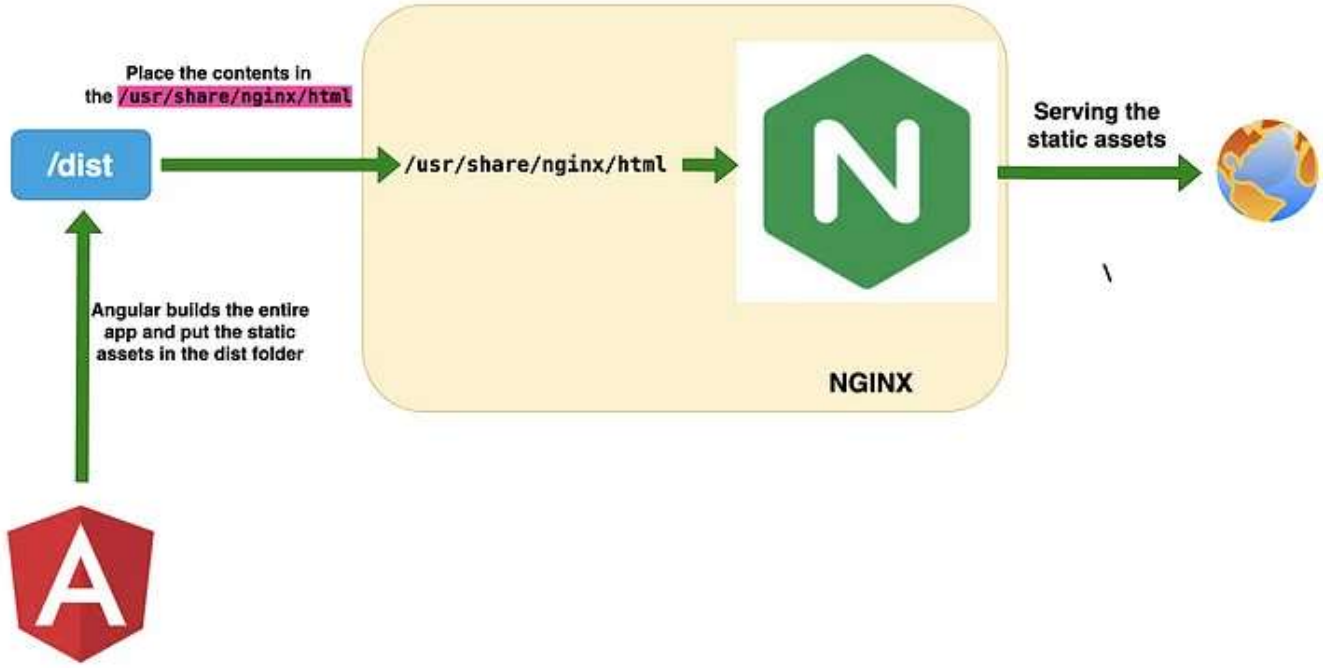
There are so many ways we can build Angular apps and ship for production. One way is to build Angular is to build the angular and serve that static co e build with NGINX and docker we don't have to deal with server runtime or server related code. All we need to build the Angular app for prod and serve the generated static content with the NGINX server.

In this post, we will see the details and implementation of the second approach. We will go through step by step with an example.

- *Introduction*
- *Prerequisites*
- *Example Project*
- *Just Enough NGINX For This Project*
- *Implementation*
- *Summary*
- *Conclusion*

Introduction

In this project, we are going to use Angular as a JS framework, NGINX as a web server, docker as a container runtime.



NGINX serving static files

The note you're looking for was deleted.

If we look at the above ( )lace the static assets in the `/dist` folder. We place these assets in the NGINX default location `/usr/share/nginx/html` where it serves the web content from.

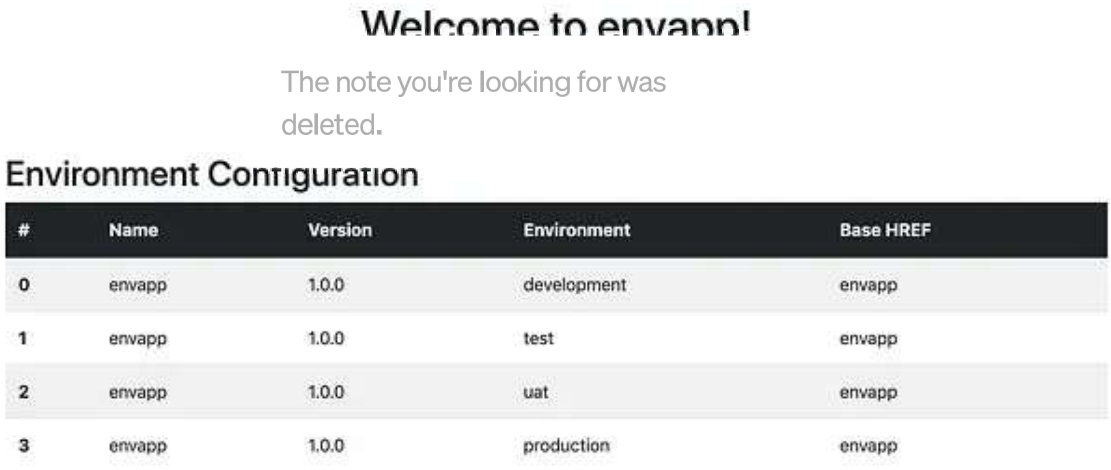
Prerequisites

There are some prerequisites to accomplish this task. We have to run NGINX in the docker and place the static assets in the NGINX and run the whole setup inside the docker. We need to install nodejs for the API setup. Please install all the below tools to follow along with this tutorial or you want to run this on your machine.

- NodeJS
- Angular CLI
- Docker for Desktop
- NGINX (need to install inside docker)

Example Project

This is a simple project which demonstrates serving static Angular application with NGINX and Docker. We have a simple app with a table which gets some data from the server.



Sample Angular App

Here is the example project where you can clone and run on your machine

```
// clone the project
git clone https://github.com/bbachi/angular-nginx-docker.git

// install and start the dependencies
npm install
npm start

// build the docker image
docker build -t appui .

// run the app
docker run -d --name appui -p 80:80 appui
```

This is a simple Angular app that shows the table and gets the data `app.config.json` from the assets folder of the application. Here are the app.component.html and app.component.ts files.

```
<!--The content below is only a placeholder and can be replaced.-->
<div style="text-align:center">
  <h1>
    Welcome to {{ title }}!
  </h1>
</div>
<div class="tblclas">
  <h2>Environment Configuration </h2>
  <table *ngIf="appData" class="table table-striped">
    <thead class="thead-dark">
      <tr>
```

```
          <th scope="col">#</th>
          <th scope="co
          <th scope="co
          <th scope="co
          <th scope="col">Base HREF</th>
        </tr>
      </thead>
      <tbody>
        <tr *ngFor="let data of appData; let i = index">
          <th scope="row">{{i}}</th>
          <td>{{data.name}}</td>
          <td>{{data.version}}</td>
          <td>{{data.environment}}</td>
          <td>{{data.basehref}}</td>
        </tr>
      </tbody>
    </table>
</div>
```

The note you're looking for was deleted.

app.component.html hosted with ❤ by GitHub view raw

```
import { Component } from '@angular/core';
import { AppService } from './app.service'

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'envapp';
  appData: any;

  constructor(private appService: AppService) {
      this.appService.getConfig().subscribe(data => {
          this.appData = data;
      })
  }
}
```

app.component.ts hosted with ❤ by GitHub view raw

App Component

Here is the service file which reads the **app.config.json** file from the assets folder.

```
[
    {
        "name": "envapp",
        "version": "1.0.0",
```

```
        "environment": "development",
        "basehref": "
    },
    {
        "name": "envapp",
        "version": "1.0.0",
        "environment": "test",
        "basehref": "envapp"
    },
    {
        "name": "envapp",
        "version": "1.0.0",
        "environment": "uat",
        "basehref": "envapp"
    },
    {
        "name": "envapp",
        "version": "1.0.0",
        "environment": "production",
        "basehref": "envapp"
    }
]
```

The note you're looking for was deleted.

app.config.json hosted with ❤ by GitHub view raw

```
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';

@Injectable({
  providedIn: 'root'
})
export class AppService {

  constructor(private http: HttpClient) { }

  configUrl = 'envapp/assets/app.config.json';

  getConfig() {
    return this.http.get(this.configUrl);
  }
}
```

app.service.ts hosted with ❤ by GitHub view raw

Service reading config json file from assets folder

Just Enough NGINX For This Project

We are not going through everything about NGINX here and we just go through just enough for this project. The note you're looking for was deleted. tuff, you can skip over to the next section.

NGINX processes are divided into one master process and several worker processes. The master process takes care of evaluating configuration and maintaining worker processes and the worker processes take care of actual requests. We can define the number of worker processes in the configuration file which can be placed in the directory `/usr/local/etc/nginx`, `/etc/nginx` or `/usr/local/nginx/conf`.

The configuration file consists of directives that form the modules or contexts. There are two kinds of directives: simple directives and block directives. A simple directive has names and parameters separated by a space and ends with a semicolon like this `listen 80;` . A block directive is the same but has additional information and surrounded by braces like this `{ listen 80; root /usr/share/nginx/html; }`.

Let's understand the NGINX configuration file that we used in this project. Below is the ***nginx.conf*** file which is located under folder ***.nginx*** at root location of the project.

Everything is a context in the configuration file. We have a hierarchical context that starts with the main context. For example, **worker_processes** and **events** are defined in the main context and another context starts with **http.** We have another context inside **http** called **server** which listens on port 80 and serving static assets from the root location `/usr/share/nginx/html`.

We can have multiple declarations of the **server** inside the **http** context and we can have multiple declarations of **location** inside the **server** context.

```
worker_processes 4;

events { worker_connections 1024; }

http {
    server {
        listen 80;
        root  /usr/share/nginx/html;
        include /etc/nginx/mime.types;

        location /appui {
```

```
12              try_files $uri /envapp/index.html;
13          }
14      }
15  }
```

nginx.conf hosted with ❤ by GitHub view raw

nginx.conf

The note you're looking for was deleted.

Implementation

We use Docker as a container runtime for this project. We are using multi-stage builds to build the Docker image. Here is the Dockerfile for the project.

```
# stage1 as builder
FROM node:10-alpine as builder

# copy the package.json to install dependencies
COPY package.json package-lock.json ./

# Install the dependencies and make the folder
RUN npm install && mkdir /app-ui && mv ./node_modules ./app-ui

WORKDIR /app-ui

COPY . .

# Build the project and copy the files
RUN npm run ng build -- --deploy-url=/envapp/ --prod


FROM nginx:alpine

#!/bin/sh

COPY ./.nginx/nginx.conf /etc/nginx/nginx.conf

## Remove default nginx index page
RUN rm -rf /usr/share/nginx/html/*

# Copy from the stahg 1
COPY --from=builder /app-ui/dist /usr/share/nginx/html

EXPOSE 4200 80
```

```
31
32  ENTRYPOINT ["nginx", '
```

Dockerfile hosted with ❤ by Git — The note you're looking for was deleted. — view raw

Stage 1

We are using `node:10-alpine` as a base image for the stage1 and copying `package.json` to install all the dependencies. We then copy the remaining project later, in that way we can skip the installing dependencies every time there is a change in the files. Docker uses cache to build the image from existing layers if there is no change.

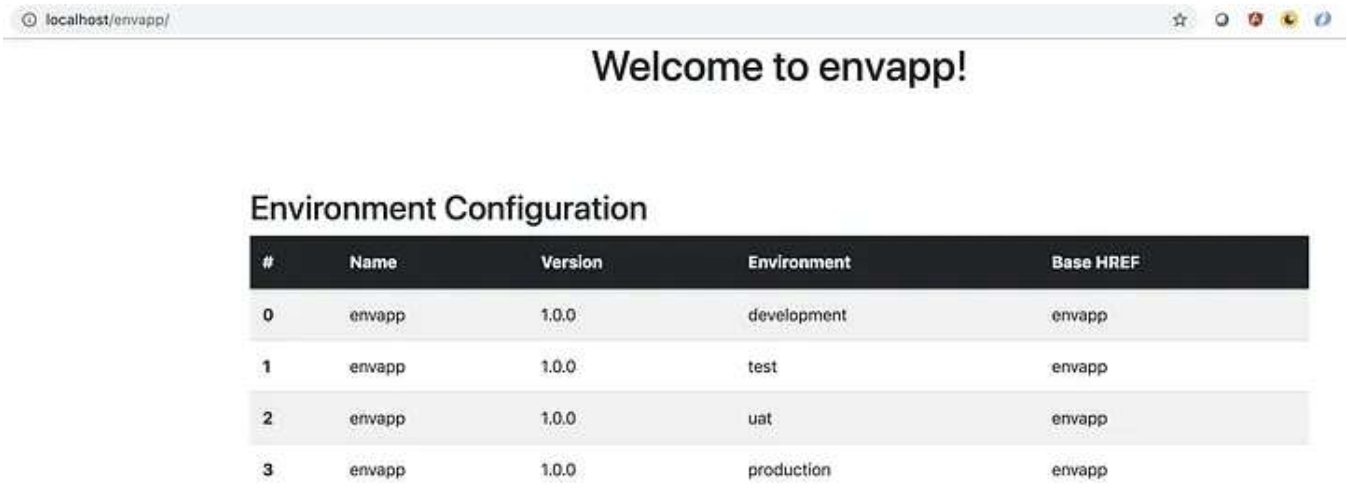
We build the project with the `deploy-url` with **/envapp/** and all the built static files are placed in the **/dist** folder.

Stage 2

Stage 2 starts with the base image `nginx:alpine` and copy the nginx.conf file, remove the index file from the root location, and finally, copy all the files from stage 1 to the root location where it can serve the content from.

Build the Image and Run the Project

Let's build the project with this command `docker build -t app-ui .` and you can run the project with this command `docker run -d --name -p 80:80 app-ui` . You can run the app on `http://localhost:80/appui`



Project running on localhost

Important Things to notice

The container port and nginx listen port should be the same which is 80 otherwise you would get **ERR_EMPTY_RESPONSE** when you run the project.

```
// container port
docker run -d --na

// nginx conf
```

The note you're looking for was deleted.

We should include this directive in the nginx.conf file otherwise all the styles are rendered as plain text in the browser.

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

Exec Into the Running Container

While the container is in running state we can exec into it and see the contents of the file system.

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

We can actually see all the contents under `/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
```

File system inside docker

Summary

- NGINX can be used as a web server or reverse proxy to serve the static content.
- All the NGINX configuration can be placed in this file `nginx.conf` .

- We need to build the angular app and place all the static files in the root location of the NGINX to ser

- Docker is used as the container runtime.
- We use multi-stage builds to reduce the final image size and remove unnecessary files from the production environment.
- Docker image can be built with `docker build -t app-ui .`
- Run the container with this command `docker run -d --name appui -p 80:80 app-ui.`
- It's very important to match ports while running the container and the listen port in nginx.conf file. Otherwise, you would get an **ERR_EMPTY_RESPONSE** error.
- You can exec into the container to explore the file system with this command `docker exec -it appui /bin/sh.`

Conclusion

NGINX is a high-performance web server that serves content and application delivery, improves security, facilitates availability and scalability for the web apps. If you want to avoid or there is no need for building UI apps with Java or node js server runtime, you can build the UI app and serve the static files with the NGINX at scale. In the next post, we can explore how to interact with the API.

JavaScript Angular Web Development Programming Software Development

The note you're looking for was deleted.

Follow

Written by Bhargav Bachina

22K Followers · Editor for Bachina Labs

Entrepreneur | 600+ Tech Articles | Subscribe to upcoming Videos
https://www.youtube.com/channel/UCWLSuUulkLIQvbMHRUfKM-g | https://www.linkedin.com/in/bachina

More from Bhargav Bachina and Bachina Labs

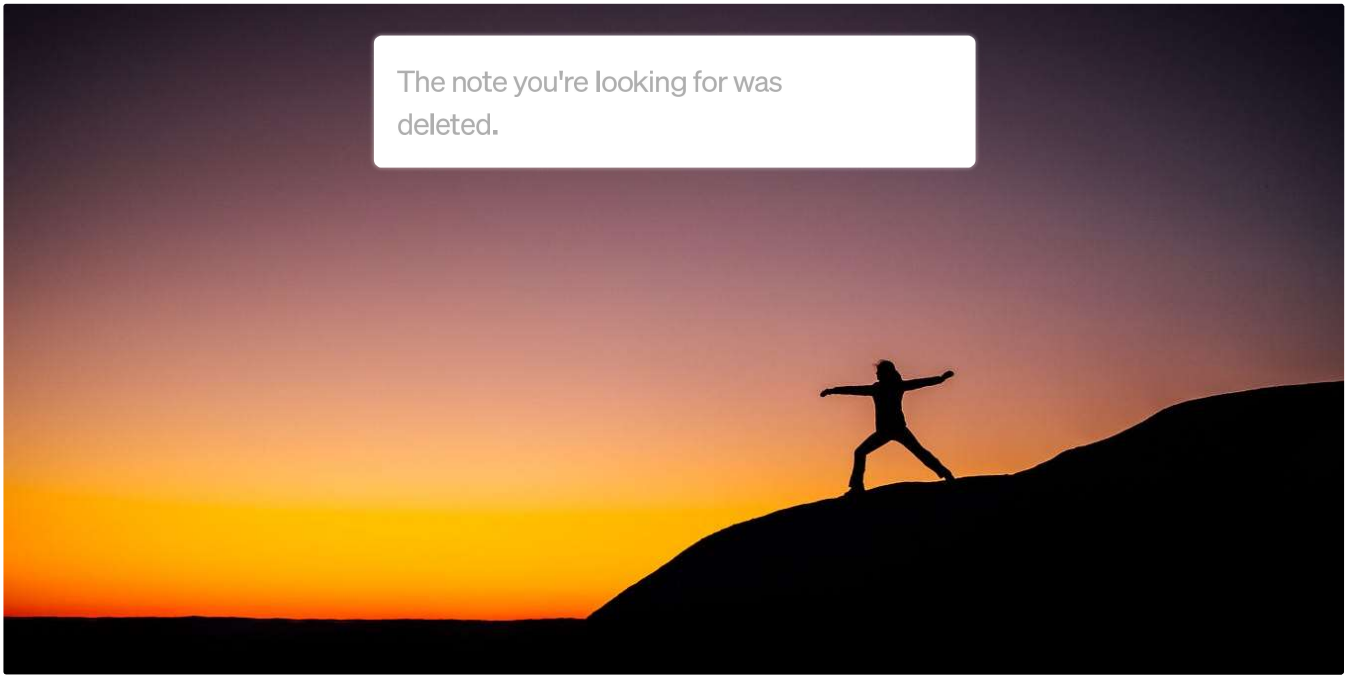
Bhargav Bachina in Bachina Labs

Kubernetes—Learn Sidecar Container Pattern

Understanding Sidecar Container Pattern With an Example Project

✦ · 6 min read · Sep 7, 2020

388 2



Bhargav Bachina in Bachina Labs

Practice Enough With These 150 Questions for the CKAD Exam

Exercises get you ready for the Certified Kubernetes Application Developer exam

✦ · 21 min read · Nov 11, 2019

2.2K 55

Bhargav Bachina in Bachina Labs

250 Practice Questions For Terraform Associate Certification

Read and Practice these q

The note you're looking for was deleted.

✦ · 47 min read · Jul 13, 202(

1.6K 39

Bhargav Bachina in Bits and Pieces

How to Implement Idle Timeout in Angular

Learn how to use idle-timeout in your Angular app to improve performance and security

6 min read · Sep 30, 2019

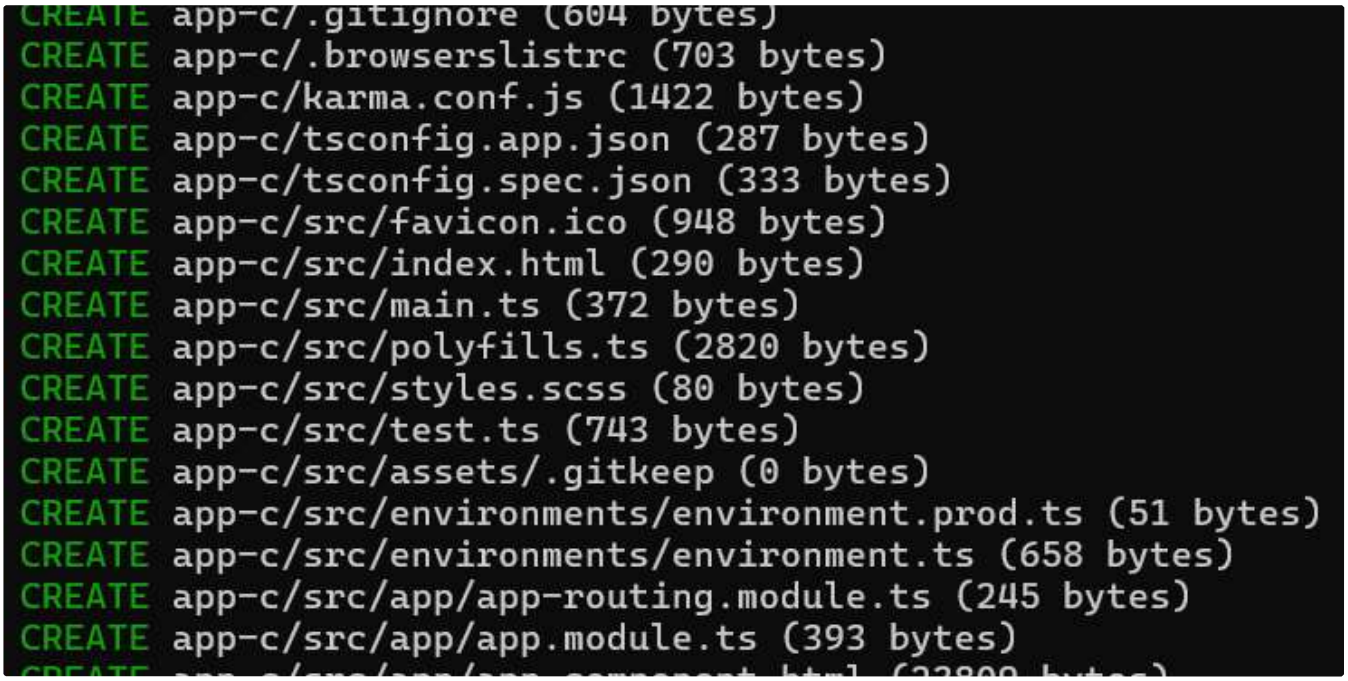
946 13

See all from Bhargav Bachina

See all from Bachina Labs

The note you're looking for was deleted.

Recommended from Medium



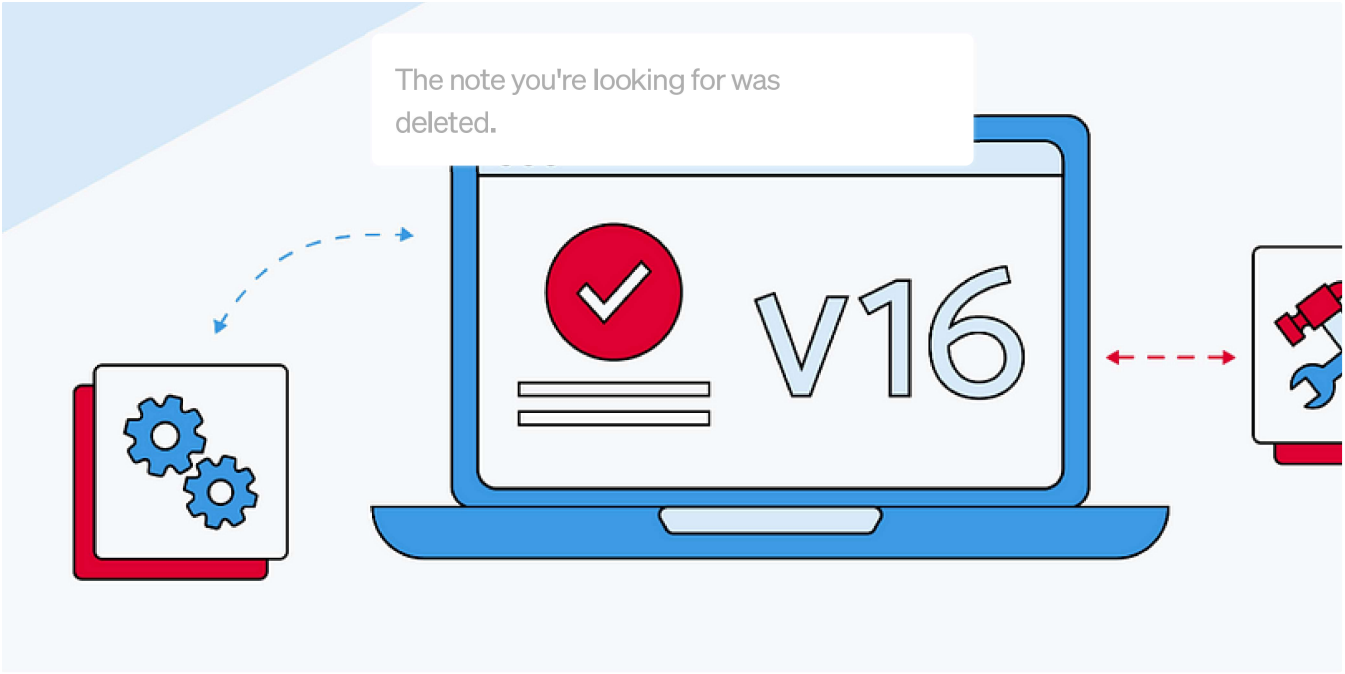
 Thilina Ranathunga

Deploying Multiple Containerized Angular Applications in different subdirectories of a single...

Docker Swarm provides a robust platform for running distributed services at scale, and NGINX is a versatile tool for managing network...

5 min read · Jul 2

 51  



Minko Gechev in Angular Blog

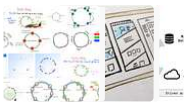
Angular v16 is here!

Six months ago, we reached a significant milestone in Angular's simplicity and developer experience by graduating the standalone APIs from...

12 min read · May 3

4.95K 35

Lists



General Coding Knowledge

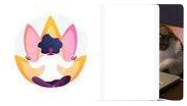
20 stories · 308 saves

It's never too late or early to start something

15 stories · 108 saves

Coding & Development

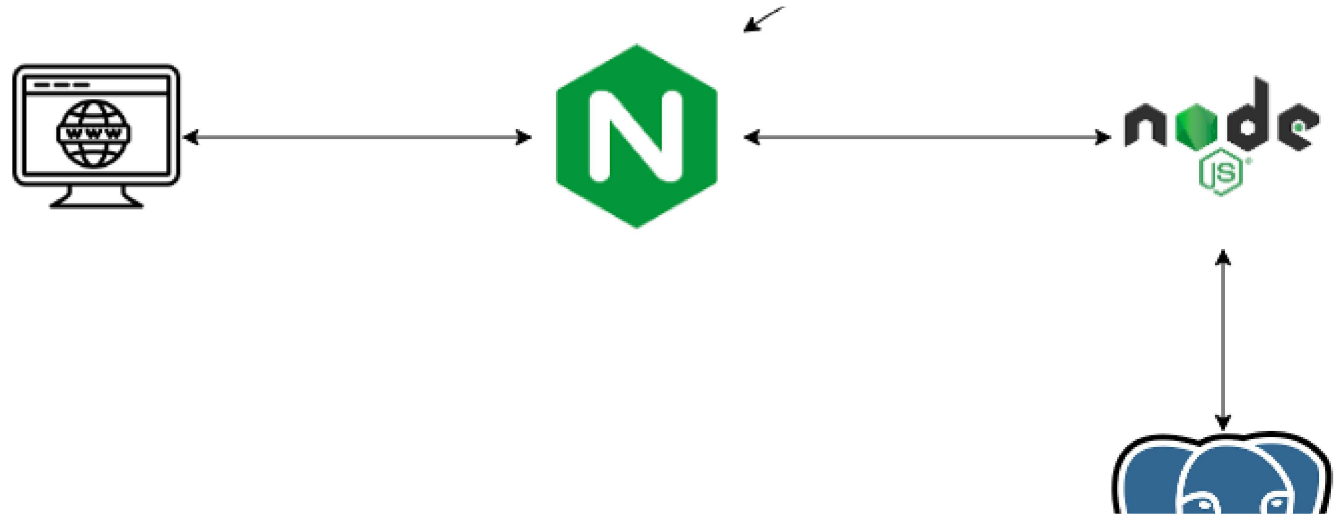
11 stories · 158 saves



Stories to Help You Grow as a Software Developer

19 stories · 348 saves

The note you're looking for was deleted.



Antonio Maccarini

Dockerize a React application with Node.js, Postgres and Nginx

This tutorial explains how to Dockerize a React application with Node.js, Postgres and Nginx. It provides step-by-step instructions on...

11 min read · Jun 8

58

Macdonald Chika

How To Install TLS/SSL on Docker Nginx Container With Let's Encrypt

Discover how to secure yo The note you're looking for was deleted. et's Encrypt and Certbot.

5 min read · Jun 9

31

Patric in Bits and Pieces

10 Common Mistakes in Angular Development

Develop High-Performance, Robust, and Secure Applications

29 min read · Apr 18

496 12

The note you're looking for was deleted.

Aphinya Dechalert in Better Programming

Should You Use Angular in 2023?

Where does the framework stand now?

· 8 min read · Apr 4

See more recommendations