CS 470 Module Two Assignment One Guide

Introduction

In this lesson you will build on your previous work with Docker by creating two containers, one to host an Angular frontend application and one to host a Node JS backend REST API. We will be using example applications covered in articles on the Angular Templates website. You do not need to read the articles or build the code by hand; however, the articles do provide useful information to help your study of full stack development. For this assignment, you will make a copy of the published code and deploy it into containers.

Summary Steps

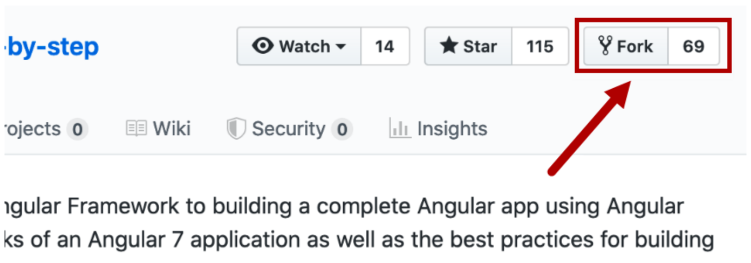
- Create container for Angular frontend:
    - Fork the Angular project
    - Clone your Angular repo: git clone {your_repo} lafs-web
    - Install nvm
    - Create dockerfile
    - docker build -t node-lafs-web
    - docker run -p 4200:4200 -d node-lafs-web
- Create container for MEAN REST API backend
    - Fork the MEAN project
    - Clone your MEAN repo: git clone {your_repo} lafs-api
    - Create dockerfile
    - docker build -t node-lafs-api
    - docker run -p 3000:3000 -d node-lafs-api

Detailed Steps

Frontend Application

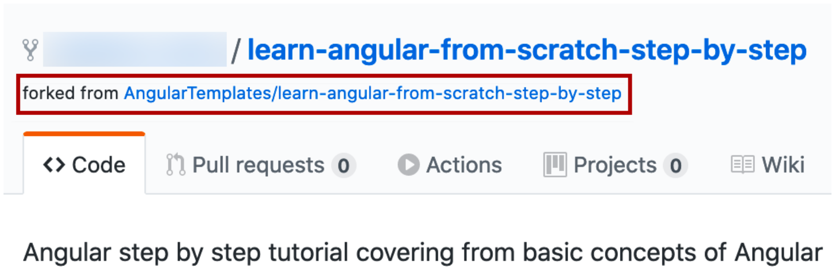
The frontend application we will use to explore containerization is available on GitHub in the AngularTemplates repository "learn-angular-from-scratch-step-by-step". To start, you will make a copy of that repository using a process known as "forking".

Ensure that you are logged into GitHub and navigate to the above repository. Click the **Fork** button on the upper right of the page.



ıgular Framework to building a complete Angular app using Angular
ks of an Angular 7 application as well as the best practices for building

You will be taken back to your GitHub account with a new repository, named the same as the original, and a message below the title indicating that it was forked.

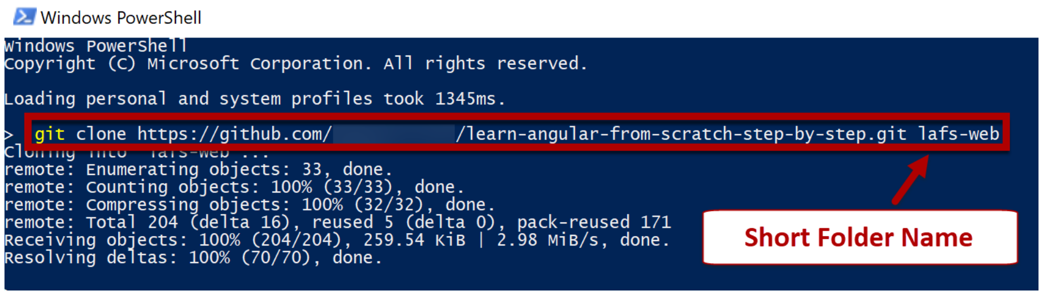

[redacted] / learn-angular-from-scratch-step-by-step

forked from AngularTemplates/learn-angular-from-scratch-step-by-step

Angular step by step tutorial covering from basic concepts of Angular

Clone your new repository into a folder on your computer.

```
> git clone {your-github-repo-url} lafs-web
```



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Loading personal and system profiles took 1345ms.

> git clone https://github.com/[redacted]/learn-angular-from-scratch-step-by-step.git lafs-web
Cloning into 'lafs-web'...
remote: Enumerating objects: 33, done.
remote: Counting objects: 100% (33/33), done.
remote: Compressing objects: 100% (32/32), done.
remote: Total 204 (delta 16), reused 5 (delta 0), pack-reused 171
Receiving objects: 100% (204/204), 259.54 KiB | 2.98 MiB/s, done.
Resolving deltas: 100% (70/70), done.
```

Note: Think of the cloned repository on your computer as a "working copy".

Install nvm, the nodejs version manager. It is not unusual to have multiple projects that each run different versions of nodejs and having a version manager will let you change your nodejs version per project. You will need to uninstall any existing versions of nodejs first. Once you have nvm installed you can install the right version of node with:

```
> nvm install 10.23.0
and then switch to it with
> nvm use 10.23.0
and double check with
> node -v
```

At this point you can optionally run the application on your computer. To do so, run the following commands:

1. cd lafs-web
2. npm install -g @angular/cli@v6-lts
3. npm install
4. ng serve
5. Open a browser and navigate to http://localhost:4200
6. **Ctrl+C** to stop the web server

NOTE: “-g” is needed here so that “ng serve” will work

Create a new Docker container by creating a Dockerfile in the top directory of the website. Note: Make sure that your Dockerfile is named "Dockerfile" with no file extension.

Dockerfile

```
# using Node v10
FROM node:10

# Create app directory
WORKDIR /usr/src/lafs

# Install app dependencies
# A wildcard is used to ensure both package.json AND package-lock.json are
copied
# where available (npm@5+)
COPY package*.json ./

RUN npm install -g @angular/cli@v6-lts
RUN npm install
# If you are building your code for production
# RUN npm ci --only=production

# Bundle app source
COPY . .

# Expose port 3000 outside container
EXPOSE 4200
# Command used to start application
CMD ng serve --host 0.0.0.0
```

Note: Refer to the NodeJS article Dockerizing a Node.js web app for more information.

Create a Docker Ignore file to indicate the files and folders that will not be copied into the container.

.dockerignore

```
.env
.git
.gitignore
node_modules
npm-debug.log
```

After creating the dockerfile and Docker Ignore file, use the Docker build command to create a container image using the following command:

```
> docker build -t node-lafs-web .
```

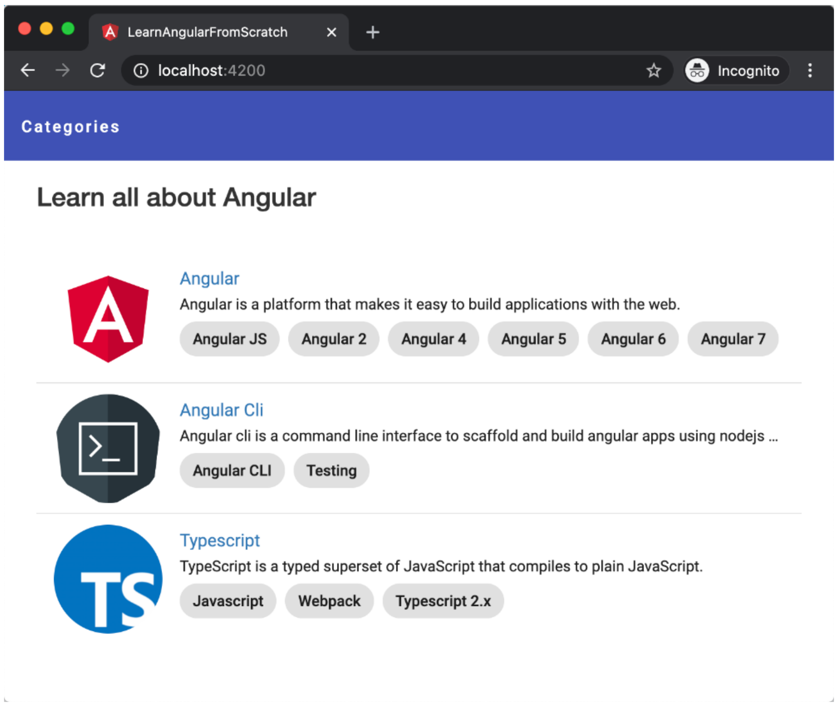
***-t** assigns a "tag" to the image used when storing in the local Docker repository.*

Run the Docker container using the following command:

```
> docker run -p 4200:4200 -d node-frontend-web
```

***-d** means "detach" the console from the container so it runs in the background and lets you use additional commands.*
***-p 4200:4200** means to map port 4200 on your computer to port 4200 inside the container.*

At this point you can open a web browser on your computer and navigate to http://localhost:4200.



Learn all about Angular

Angular
Angular is a platform that makes it easy to build applications with the web.
Angular JS | Angular 2 | Angular 4 | Angular 5 | Angular 6 | Angular 7

Angular Cli
Angular cli is a command line interface to scaffold and build angular apps using nodejs ...
Angular CLI | Testing

Typescript
TypeScript is a typed superset of JavaScript that compiles to plain JavaScript.
Javascript | Webpack | Typescript 2.x

Backend REST API
The backend application we will use to explore containerization is available on GitHub in the AngularTemplates repository "[learn-how-to-build-a-mean-stack-application]()". To start, you will make a copy of that repository using a process known as "[forking]()".

Ensure that you are logged into GitHub and navigate to the repository URL: [https://github.com/AngularTemplates/learn-how-to-build-a-mean-stack-application.](https://github.com/AngularTemplates/learn-how-to-build-a-mean-stack-application) **Fork** this repository as you did for the frontend application.

Clone your new repository into a folder on your computer:

```
> git clone {your-github-repo-url} lafs-api
```

At this point you can optionally run the application on your computer. To do so, you will need to install Node JS and run the following commands:

1. cd lafs-api
2. npm install
3. npm run start
4. Open a browser and navigate to http://localhost:3000/

5. **Ctrl+C** to stop the web server

Create a Dockerfile in the top directory of the website, similar to what you did for the frontend application. Refer to the NodeJS article Dockerizing a Node.js Web App as needed. This file will be a little different than your other Dockerfile.

Dockerfile

```
# using Node v10
FROM node:10

# Create app directory
WORKDIR /usr/src/app

# Install app dependencies
# A wildcard is used to ensure both package.json AND package-lock.json are
copied
# where available (npm@5+)
COPY package*.json ./

RUN npm install
# If you are building your code for production
# RUN npm ci --only=production

# Bundle app source
COPY . .

# Expose port 3000 outside container
EXPOSE 3000

# Command used to start application
CMD npm run start
```

After creating the dockerfile and Docker Ignore file, use the Docker build command to create a container image using the following command:

```
> docker build -t node-backend-api .
```

***-t** assigns a "tag" to the image used when storing in the local Docker repository.*

Run the Docker container using the following command:

```
> docker run -p 3000:3000 -d node-lafs-api
```

***-d** means "detach" the console from the container so it runs in the background and lets you use additional commands.*

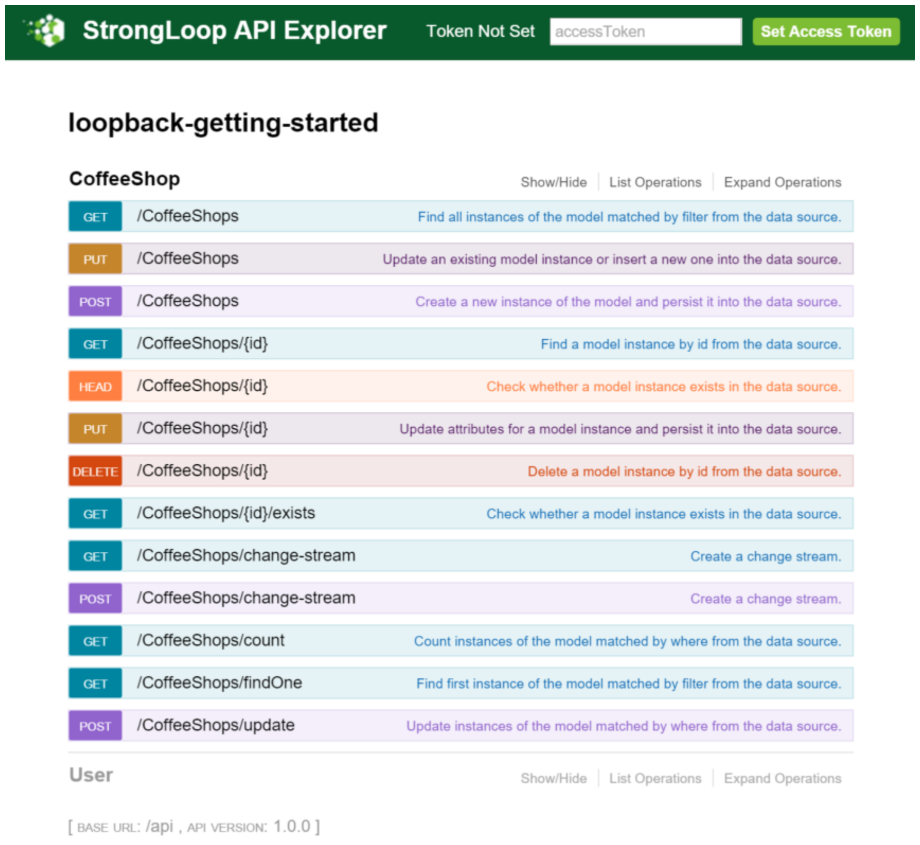
***-p 3000:3000** means to map port 3000 on your computer to port 3000 inside the container.*

At this point you can open a web browser on your computer and navigate to http://localhost:3000/.

You will receive an error because the backend REST API is looking for a MongoDB instance to connect to. In the next lesson, you will use more advanced Docker techniques to get multiple containers working together and able to interact over the network.

Loopback
A helpful aspect of the example application is the use of LoopBack to build the REST API. LoopBack is a framework that saves time by "discovering" the models in your project and automatically generating all the REST endpoints without you having to code every one of them by hand. Another useful feature is the API Explorer, consisting of generated documentation and the ability to test the endpoints right from the documentation pages without having to create client code to invoke the endpoints.

Here is an example of the API Explorer from the documentation:



loopback-getting-started

CoffeeShop

GET	/CoffeeShops	Find all instances of the model matched by filter from the data source.
PUT	/CoffeeShops	Update an existing model instance or insert a new one into the data source.
POST	/CoffeeShops	Create a new instance of the model and persist it into the data source.
GET	/CoffeeShops/{id}	Find a model instance by id from the data source.
HEAD	/CoffeeShops/{id}	Check whether a model instance exists in the data source.
PUT	/CoffeeShops/{id}	Update attributes for a model instance and persist it into the data source.
DELETE	/CoffeeShops/{id}	Delete a model instance by id from the data source.
GET	/CoffeeShops/{id}/exists	Check whether a model instance exists in the data source.
GET	/CoffeeShops/change-stream	Create a change stream.
POST	/CoffeeShops/change-stream	Create a change stream.
GET	/CoffeeShops/count	Count instances of the model matched by where from the data source.
GET	/CoffeeShops/findOne	Find first instance of the model matched by filter from the data source.
POST	/CoffeeShops/update	Update instances of the model matched by where from the data source.

User

[BASE URL: /api , API VERSION: 1.0.0]