# CS 470 Module Two Assignment One Guide

## Introduction

In this lesson, you will build on your previous work with Docker. You will create two containers. One will host an Angular frontend application. The other will host a Node JS backend REST API. You will use example applications covered in articles on the [Angular Templates](https://angular-templates.io/) website. You do not need to read the articles or build the code by hand. But the articles 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

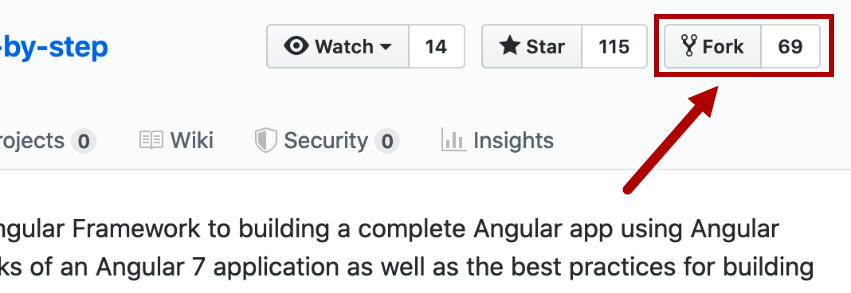
1. Create container for Angular frontend:
   1. Fork the Angular project.
   2. Clone your Angular repo: git clone {your\_repo} lafs-web.
   3. Install nvm.
   4. Create a Dockerfile.
   5. Create a container image using docker build -t node-lafs-web.
   6. Run the Docker container using docker run -p 4200:4200 -d node-lafs-web.
2. Create container for MEAN REST API backend:
   1. Fork the MEAN project.
   2. Clone your MEAN repo: git clone {your\_repo} lafs-api.
   3. Create a Dockerfile.
   4. Create a container image using docker build -t node-lafs-api.
   5. Run the Docker container using docker run -p 3000:3000 -d node-lafs-api.

## Detailed Steps

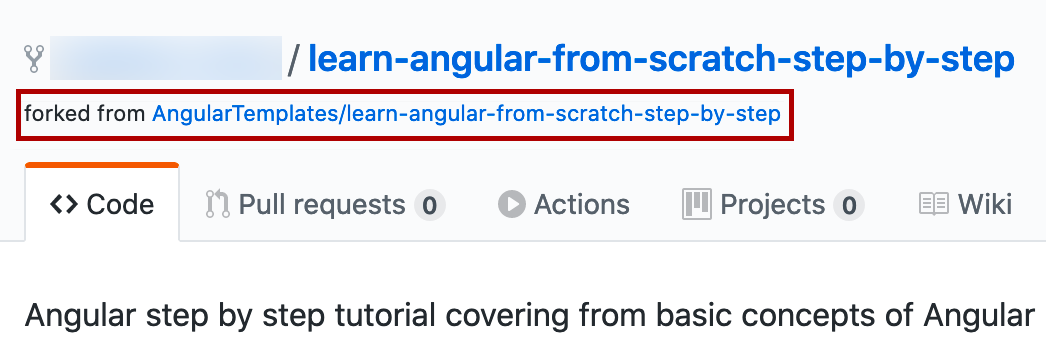
### Frontend Application

The frontend application you will use to explore containerization is available on GitHub in the Angular Templates repository [learn-angular-from-scratch-step-by-step](https://github.com/AngularTemplates/learn-angular-from-scratch-step-by-step). To start, you will make a copy of that repository using a process known as forking. To learn about forking, visit the [Contributing to a Project](https://docs.github.com/en/get-started/exploring-projects-on-github/contributing-to-a-project) webpage.

Make certain that you are logged into GitHub and navigate to the above repository. Click the **Fork** button in the upper-right corner of the page.



You will be returned 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.



Clone your new repository in a folder on your computer.

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

Windows PowerShell showing the command line interface and cloning your new repository into a folder on your computer:

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

An arrow indicates that "lafs-web" is the "Short folder name". 

**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. Having a version manager will let you change your nodejs version per project. You must uninstall any existing versions of nodejs first. Once you have installed the nvm, 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

Create a new Docker container by creating a Dockerfile in the top directory of the website. **Note:** Make certain that your Dockerfile is named “Dockerfile” with no file extension. Learn more about Dockerfiles on the [Dockerfile Reference](https://docs.docker.com/engine/reference/builder/) webpage.

The commands should resemble the following text:

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

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

The commands should resemble the following text:

.dockerignore

.env

.git

.gitignore

node\_modules

After creating the Dockerfile and Docker Ignore file, use the Docker build command to create a container image using the command below. **Note:** Make certain you are in the current working directory. For example, /Github/lafs-web/in your local shell. To learn more, visit the [Docker Build](https://docs.docker.com/engine/reference/commandline/build/) webpage.

> docker build -t node-lafs-web .

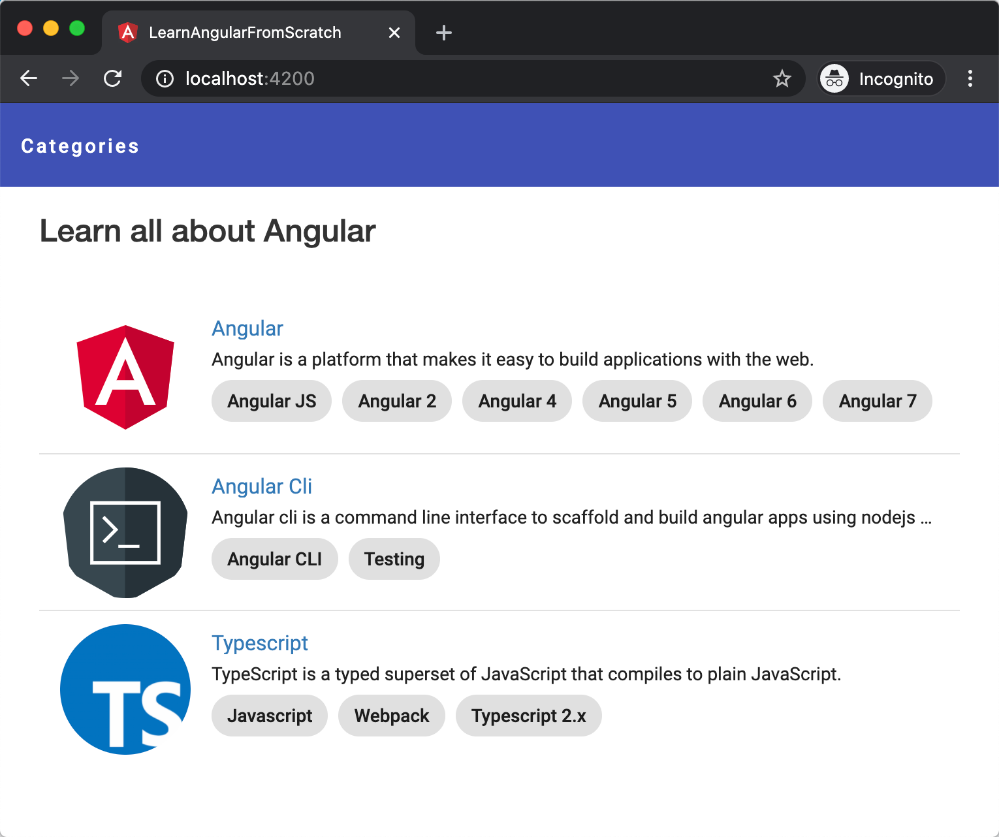
**-t** assigns a tag to the image used when storing in the local Docker repository.

Run the Docker container using the following command:

> docker run -p 4200:4200 -d node-lafs-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.

Now, you can open a web browser on your computer and navigate to http://localhost:4200.



### Backend REST API

The backend application you will use to explore containerization is available on GitHub in the Angular Templates repository [learn-how-to-build-a-mean-stack-application](https://github.com/AngularTemplates/learn-how-to-build-a-mean-stack-application). To start, you will make a copy of that repository using the forking process.

Make certain that you are logged into GitHub and navigate to the above repository. **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

Now, you may optionally run the application on your computer. To do so, you must 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. You can follow these steps to create this Dockerfile:

1. Start by copying the provided frontend Dockerfile to the lafs-api folder.
2. Edit the document to remove all references to Angular since this is backend only. Also, replace the last line with **CMD [“node”,”server/server.js”]**. It should look like below:

# 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

# 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 [“node”, “server/server.js”]

1. Copy the .dockerignore file you created in lafs-web folder to the lafs-api folder.

After creating the Dockerfile and Docker Ignore file, use the Docker build command to create a container image using the command below. To learn more, visit the [Docker Build](https://docs.docker.com/engine/reference/commandline/build/) webpage.

> docker build -t node-lafs-api .

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

**Note:** If you get errors related to nodejs, uninstall any existing versions, and then install the latest version. You might also need to install or update sass (npm i -D sass) if you get an error with the ng serve command.

Run the Docker container using the following command:

> run

* **-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

Using LoopBack to build the REST API is helpful for the example application. 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 each of them by hand. To learn more, visit the [LoopBack](https://loopback.io/lb3) website.

API Explorer consists of generated documentation and allows you to test the endpoints from the documentation pages without creating client code to invoke the endpoints. To learn more, visit the [Use API Explorer](https://loopback.io/doc/en/lb3/Use-API-Explorer.html) webpage.

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

