**Step1: Start AWS Resources**

* Start EC2 Instance
* Start RDS Instance
* Start S3 Instance

**Step2: Configure Spring and Angular Projects**

* Added “\*” inside *@CrossOrigins*
* Replaced “*localhost*” inside angular’s service methods with window.location.hostname to automatically call the EC2 server instance’s IP address while calling RestFul services. This takes care of the changing dynamic IP of the EC2 instance. This way we don’t have to manually change our URLs inside angular each time we restart our EC2 server
* Test database connection using HeidiSQL.
* Update *application.properties* and change the database settings as per your RDS configuration.

**Step3: Upload SpringRestful jar file and Angular’s zip file (Without node modules) to S3**

* Upload files to S3 and make sure to grant access to everyone so that the files can be downloaded into our EC2 instances.

**Step4: Connect to the EC2 instance using Putty**

**Step5: Configure docker**

1. *sudo su* to get admin/root access
2. *yum docker install* to install docker into your ec2 instance.
3. *docker –v* to check version
4. *#sudo yum install –y nodejs*
5. *service docker start* to start docker

**Angular Image Creation:**

[**https://medium.com/joolsoftware/how-to-set-up-an-angular-cli-project-with-docker-compose-a3ec78f179ab**](https://medium.com/joolsoftware/how-to-set-up-an-angular-cli-project-with-docker-compose-a3ec78f179ab)

[**https://mherman.org/blog/dockerizing-an-angular-app/**](https://mherman.org/blog/dockerizing-an-angular-app/)

1. *mkdir angular* to create a separate folder for your angular docker image creation
2. *cd angular* to change the current directory to the newly created angular folder.
3. Now, download the angular zip file from your S3 instance using the command *wget <URL>*
4. Extract the zip file using *unzip <zipname>.zip*
5. *cd <zipname>* to change the current directory to the folder created after unzipping your angular zip file.
6. Now, create a dockerfile using the command *vi Dockerfile* and paste the following code inside it:

#Install nodejs in the container/image

***FROM node:12***

#make a directory

***RUN mkdir /usr/src/app***

#make the created directly as your working directory for the RUN #commands

***WORKDIR /usr/src/app***

#install angular cli in workdir

***RUN npm install –g @angular/cli***

#Install all angular dependencies (node\_modules)

***RUN npm install***

#Copy your files from the current directory into workdir

# COPY <src> <dest>

#COPY . <dest> --->current to dest directory

***COPY . .***

#Command to be executed on calling docker run

# By default, Docker exposes container ports to the IP address 0.0.0.0

***CMD ng serve –host 0.0.0.0***

1. *docker build –t <imagename> .*this will now create a docker image of your angular application by the name you specify.
2. *docker run –p angularport:dockerport <imagename>* will now run your docker image into the container. The port command is to link your docker port with your ec2 port and expose it so that it will be accessible.

Alternate Dockerfile:

*FROM node:14.16.0*

*WORKDIR /app*

*COPY package.json /app/package.json*

*RUN npm install*

*RUN npm install rxjs*

*RUN npm install -g @angular/cli@7.3.9*

*COPY . /app*

*CMD ng serve --host 0.0.0.0*

Troubleshooting:

1. RxJs error - node\_modules/rxjs/internal/types.d.ts(81,44): error TS1005: ';' expected error after installation of Angular
   1. Go to package.json and modify "rxjs": "^6.0.0" to "rxjs": "6.0.0"
   2. Run npm install in your project.
2. Issue after running npm audit - errors: Data path “. builders['app-shell']” should have required property 'class'
   1. In your package.json change the devkit builder, change "@angular-devkit/build-angular": "^0.800.1", to @angular-devkit/build-angular": "0.13.4"

Similarly, for Spring, create a separate folder and change current directory into it and follow the following steps:

To create jar file --> Run as --> Maven build --> goals: package

1. wget <url> to get your .jar file of your spring project
2. Now create a docker file using *vi Dockerfile* and add the following code:

#Get runtime java environment

***FROM java:8***

#Does not expose port to outside, but just used as metadata to inform that #this app uses 8080

***EXPOSE 8080***

#add the jar file into your docker container

***ADD <jarname>.jar <jarname>.jar***

#Command to execute when calling docker run

***CMD java –jar <jarname>.jar***

1. *docker build –t <imagename> .* to create docker image of your spring application. (You need to add a dot, which means to use the Dockerfile in the local directory.)
2. *docker run –p springport:dockerport <imagename>* to start your docker image containing the spring application.

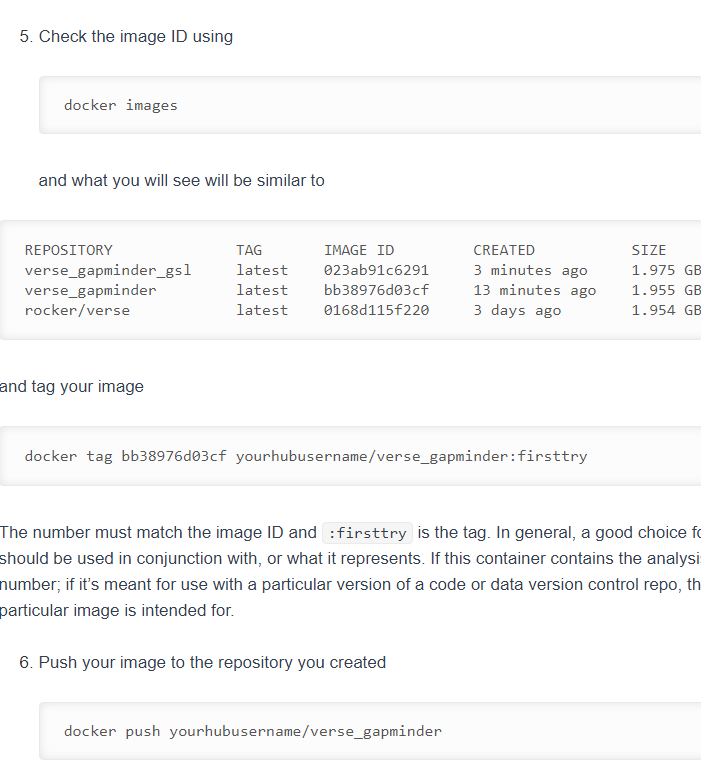
And that’s it! Both docker containers are running. You can check that using the command ***docker container ls*** or ***docker ps***

**Push image:**

[**https://ropenscilabs.github.io/r-docker-tutorial/04-Dockerhub.html**](https://ropenscilabs.github.io/r-docker-tutorial/04-Dockerhub.html)

*docker login --username=yourusername*

Enter password when prompted



To pull

*docker pull saurabhpotdar94/angularrms:v3*

To pull and run image (Docker automatically downloads from hub)

*docker run -p 4200:4200 saurabhpotdar94/angularrms:v3*