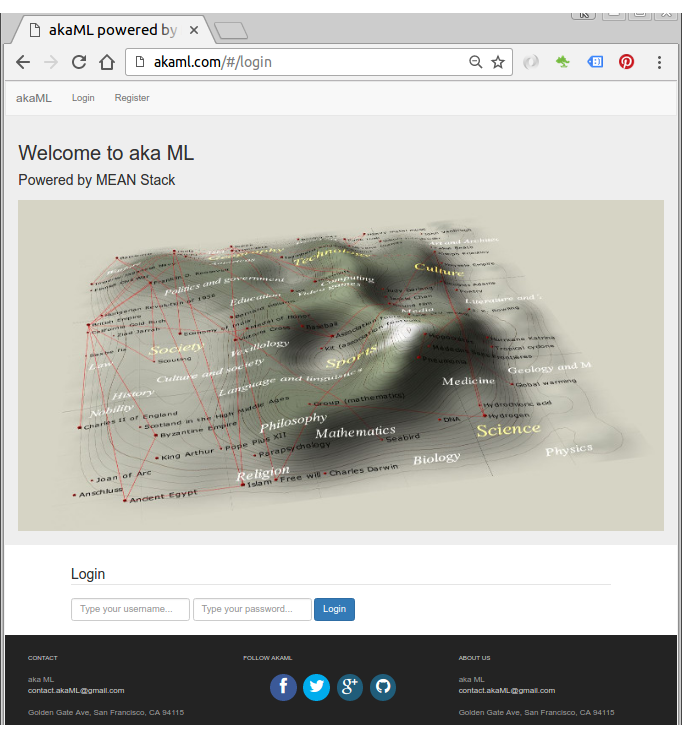
[MEAN Stack app on Docker containers : micro services](http://www.bogotobogo.com/MEAN-Stack/MEAN-Stack-NodeJS-Angular-Docker.php)

Introduction

In this tutorial, we'll deploy MEAN application to two Docker containers, and our local machine will be hosting the two containers:

1. mongodb container
2. node/express/angular app

Here is the home page of the app:



Source code : [Github](https://github.com/Einsteinish/akaML" \t "_blank)

The app uses passport for user authentication, and for more features, please consult the README.md of the repo.

Local nginx for testing app

Though we don't need Nginx server in the Docker work flow of this tutorial, before we deploy our app to Docker containers, we may want to test it with Nginx. The configuration file looks like this:

server {

listen 80;

server\_name akaml.com;

location / {

proxy\_pass http://localhost:3000;

proxy\_http\_version 1.1;

proxy\_set\_header Upgrade $http\_upgrade;

proxy\_set\_header Connection 'upgrade';

proxy\_set\_header Host $host;

proxy\_cache\_bypass $http\_upgrade;

}

}

With **/etc/hosts**:

127.0.0.1 akaml.com

To run our MEAN app on host machine but not on container, the following line in **config/database.js** should be modified like this instead of "container's ip":

var dbURI = 'mongodb://localhost:27017/myApp';

Also, mongodb should be running before MEAN app's run:

$ sudo service mongodb restart

In the project folder, install packages:

$ npm install

$ sudo npm install -g bower

$ bower install

Then start Nginx proxy server:

$ sudo service nginx start

Now, we're ready to run MEAN app using the Nginx as a reverse proxy configuration:

$ node server.js

Or

$ nodemon server.js

Or

$ pm2 start server.js

Apache config is also available from the [repo](https://github.com/Einsteinish/akaML.git).

MongoDB docker

To get our application running, the MongoDB container needs be started first.

We'll use [OFFICIAL REPOSITORY : mongo](https://hub.docker.com/_/mongo/).

$ docker run [-p 27017:27017] --name mymongodb -d mongo

1. **-p 27017:27017** exposes the MongoDB port so the mean container can connect to it.
2. **-d** runs it as a background process (detached mode).
3. **--name mymongodb** gives this container a name so it can be referenced.
4. **mongo** is the image name that should be run.

Our **mongo** will be listening on 27017 port by default and that's it. We don't have to do anything once we launched the container.

$ docker ps

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

8f333617ed15 mongo "/entrypoint.sh mongo" 4 minutes ago Up 3 minutes 27017/tcp mymongodb

MEAN stack on docker

Even though we'll build our container starting from a simple **ubuntu:14.04** images and make it ready for MEAN app, we may use the ready-made image of MEANJS (<https://hub.docker.com/r/maccam912/meanjs/>).

If **maccam912/meanjs** is chosen, we can run the container with the following command:

$ docker run -i -t --name mymeanjs --link mymongodb:db\_1 -p 80:3000 maccam912/meanjs:latest bash

Then, we can skip this section.

$ docker run -i -t --name mymeanjs --link mymongodb:db\_1 -p 80:3000 ubuntu:14.04 bash

Here, the **--link mymongodb:db\_1** argument is a link between this **mymeanjs** container and **mymongodb** container. This is a docker's way of communicating between containers.

**db\_1** is an alias to reference this connected container. In other words, our MEAN application is set to use **db\_1**.

By the argument of **-p 80:3000**, we're mapping the 3000 container port to 80 host machine port. Our MEAN application is set to run on port 3000, and the mapping enables any request on http:80 from outside the container to access our app running deep inside our container.

To install MEAN, we need to work within the docker container and do the following:

# apt-get update

# apt-get install nodejs

# ln -s "$(which nodejs)" /usr/bin/node

# apt-get install npm

To check if our install:

# node -v

v0.10.25

# npm -v

1.3.10

Express & bower install:

# npm install -g express

# npm install -g bower

Get our MEAN app

Install git:

# apt-get install git

Clone our repo to get the source code:

# cd home

# git clone https://github.com/Einsteinish/akaML.git

# cd akaML

On our MEAN.JS folder, download all the package dependencies:

# npm install

Install the front-end dependencies running by running bower:

# bower install --allow-root

Run our MEAN app

Make sure our mongodb ip is correct in **config/database.js**:

var dbURI = 'mongodb://172.17.0.2:27017/myApp';

We can check it by issuing the following command on our host machine, and this will give us two ips:

172.17.0.3

172.17.0.2

On our MEAN container, we can check ip via "ifconfig":

inet addr:172.17.0.3

So, the ip for mongodb is '172.17.0.2', and our database configuration is correct!

Let's run our MEAN app:

root@bcd9985dc871:/home/akaML# node server.js

Express server listening on port : 3000

Mongoose connection open to mongodb://172.17.0.2:27017/myApp

GET / 304 71ms

GET /scripts/lib/angular-toastr/dist/angular-toastr.css 200 95ms - 6.64kb

GET /scripts/lib/bootstrap/dist/css/bootstrap.css 200 112ms - 142.59kb

GET /css/app.css 200 226ms - 1.32kb

GET /scripts/lib/requirejs/require.js 200 186ms - 84.24kb

GET /scripts/src/main.js 200 38ms - 1.46kb

GET /favicon.ico 404 5ms

GET /scripts/src/app.js 200 5ms - 3.89kb

GET /scripts/lib/jquery/dist/jquery.min.js 200 31ms - 84.33kb

GET /scripts/lib/angular/angular.min.js 200 37ms - 156.3kb

GET /scripts/src/controllers.js 200 5ms - 7.39kb

GET /scripts/lib/cryptojslib/rollups/pbkdf2.js 200 68ms - 5.4kb

GET /scripts/src/services.js 200 45ms - 4.23kb

GET /scripts/lib/bootstrap/dist/js/bootstrap.min.js 200 41ms - 36.18kb

GET /scripts/lib/angular-route/angular-route.min.js 200 26ms - 4.65kb

GET /scripts/lib/angular-animate/angular-animate.min.js 200 26ms - 25.11kb

GET /scripts/lib/angular-local-storage/dist/angular-local-storage.min.js 200 44ms - 6.25kb

GET /scripts/lib/angular-toastr/dist/angular-toastr.tpls.min.js 200 18ms - 7.02kb

{ REQUEST:

{ HEADERS:

{ host: 'akaml.com',

connection: 'keep-alive',

accept: 'application/json, text/plain, \*/\*',

'user-agent': 'Mozilla/5.0 (X11; Linux x86\_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/52.0.2743.82 Safari/537.36',

referer: 'http://akaml.com/',

'accept-encoding': 'gzip, deflate, sdch',

'accept-language': 'en-US,en;q=0.8',

'if-none-match': '"1892644392"' },

BODY: {} } }

GET /partials/login 304 35ms

{ REQUEST:

{ HEADERS:

{ host: 'akaml.com',

connection: 'keep-alive',

accept: 'application/json, text/plain, \*/\*',

'user-agent': 'Mozilla/5.0 (X11; Linux x86\_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/52.0.2743.82 Safari/537.36',

referer: 'http://akaml.com/',

'accept-encoding': 'gzip, deflate, sdch',

'accept-language': 'en-US,en;q=0.8',

'if-none-match': '"1227391045"' },

BODY: {} } }

GET /partials/nav.html 304 8ms

{ REQUEST:

{ HEADERS:

{ host: 'akaml.com',

connection: 'keep-alive',

accept: 'application/json, text/plain, \*/\*',

'user-agent': 'Mozilla/5.0 (X11; Linux x86\_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/52.0.2743.82 Safari/537.36',

referer: 'http://akaml.com/',

'accept-encoding': 'gzip, deflate, sdch',

'accept-language': 'en-US,en;q=0.8' },

BODY: {} } }

GET /partials/header.html 200 5ms - 170b

GET /img/home/akaML-home.jpg 200 28ms - 82.08kb

GET /scripts/src/main.js 304 3ms

GET /scripts/src/app.js 304 5ms

GET /scripts/lib/angular/angular.min.js 304 2ms

GET /scripts/src/controllers.js 304 4ms

GET /scripts/src/services.js 304 1ms

GET /scripts/lib/cryptojslib/rollups/pbkdf2.js 304 2ms

GET /scripts/lib/bootstrap/dist/js/bootstrap.min.js 304 2ms

GET /scripts/lib/angular-route/angular-route.min.js 304 4ms

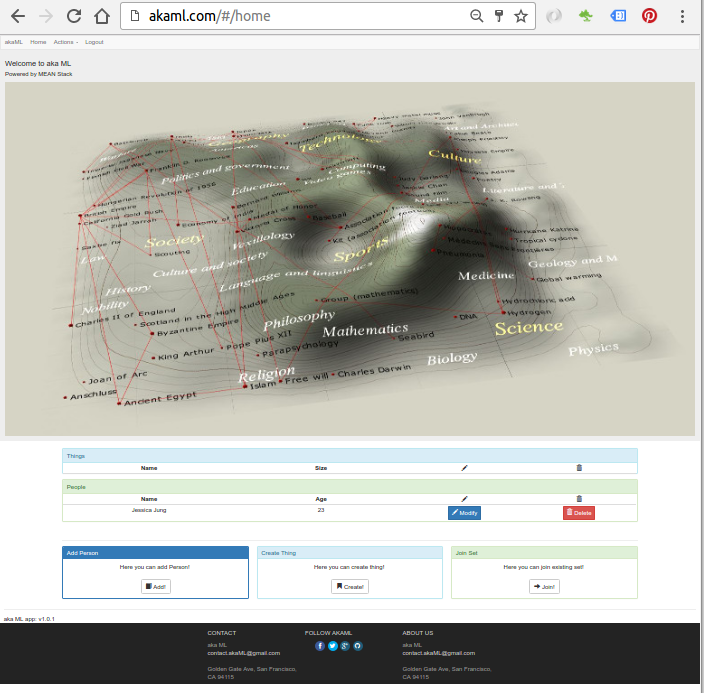
GET /scripts/lib/angular-animate/angular-animate.min.js 304 1ms

GET /scripts/lib/angular-local-storage/dist/angular-local-storage.min.js 304 3ms

GET /scripts/lib/angular-toastr/dist/angular-toastr.tpls.min.js 304 3ms

GET /scripts/lib/requirejs/require.js 304 2ms

At this point, we'll get same the page in the "Introduction" section of this tutorial. After Registration/Login, we will get the following:



Optional : Run node server with PM2

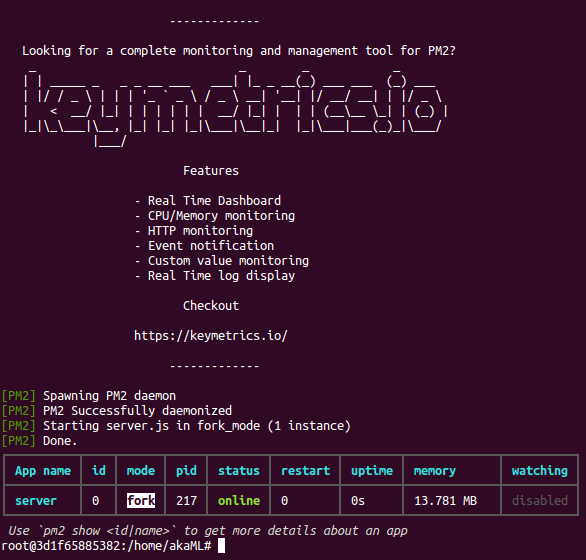
We may want to run node server as a daemon using [PM2](http://pm2.keymetrics.io/).

# npm install pm2 -g

Now that the PM2 is installed, let's start our Node application:

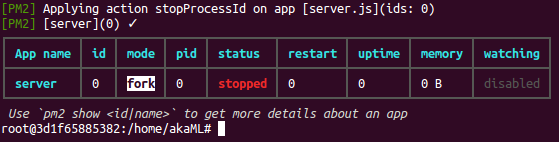
# pm2 start server.js

Then, we get the following screen output:



To stop the server:

# pm2 stop server.js



Optional : Run node server with nodemon

The PM2 has relatively bigger footprints, so as an alternative we can use **nodemon**.

# npm install -g nodemon

# nodemon server.js

Optional : Saving the docker image

We want to save our work (layers in docker terminology) so far.

Let's exit the NodeJS app container:

root@3d1f65885382:/home/akaML# exit

$

Then, check which containers are running:

$ docker ps

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

8f333617ed15 mongo "/entrypoint.sh mongo" 2 hours ago Up 2 hours 27017/tcp mymongodb

Since we exited from the NodeJS container, only the mongodb container is running.

To check our recent containers, we can use **docker ps -a** command:

$ docker ps -a

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

3d1f65885382 maccam912/meanjs:latest "bash" 2 hours ago Exited (0) 4 minutes ago mymeanjs

"docker commit" is the command we want to use to save the image:

$ docker commit -a khong 3d1f65885382 nodejs-micro-service:0.1

581298efda9f2874ed86f90f47b4834c6eb863650cdf4e3b764d939ece1cfd31

Here "-a" flag is for the author, "3d1f65885382" is the **container id**, and after that we specified the name (**repository**) of the image with version **tag**.

Let's check we really create a new image:

$ docker images

REPOSITORY TAG IMAGE ID CREATED VIRTUAL SIZE

nodejs-micro-service 0.1 581298efda9f 3 minutes ago 1.116 GB

Let's run our newly created image in background:

$ docker run -d -p 80:3000 nodejs-micro-service:0.1 pm2 start /home/akaML/server.js

8d3eb5ab0d89d65243b29ff06ccc781bca9c3698ebaa791ac1a4f8ae3a92a76d

Here, the "-d" flag is "detached" meaning background run, "80:3000" is port forwarding, "nodejs-micro-service:0.1" is the container name, and "node /home/akaML/server.js" is the command to run our nodejs app.

Now we should have two running containers - mongo and nodejs

$ docker ps

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

0af14aa1f952 nodejs-micro-service:0.1 "node /home/akaML/ser" 8 seconds ago Up 5 seconds 80/tcp, 443/tcp, 0.0.0.0:80->3000/tcp evil\_hugle

8f333617ed15 mongo "/entrypoint.sh mongo" 8 hours ago Up 8 hours 27017/tcp mymongodb

We can use "docker attach container-id" command to see what's going on inside our containers:

$ docker attach 0af14aa1f952

GET / 304 64ms

GET /css/app.css 304 16ms

...

Note: Unfortunately, the command "pm2 start /home/akaML/server.js" in detached mode does not seem to be working. So, here, I used "node server.js" as a command argument.