

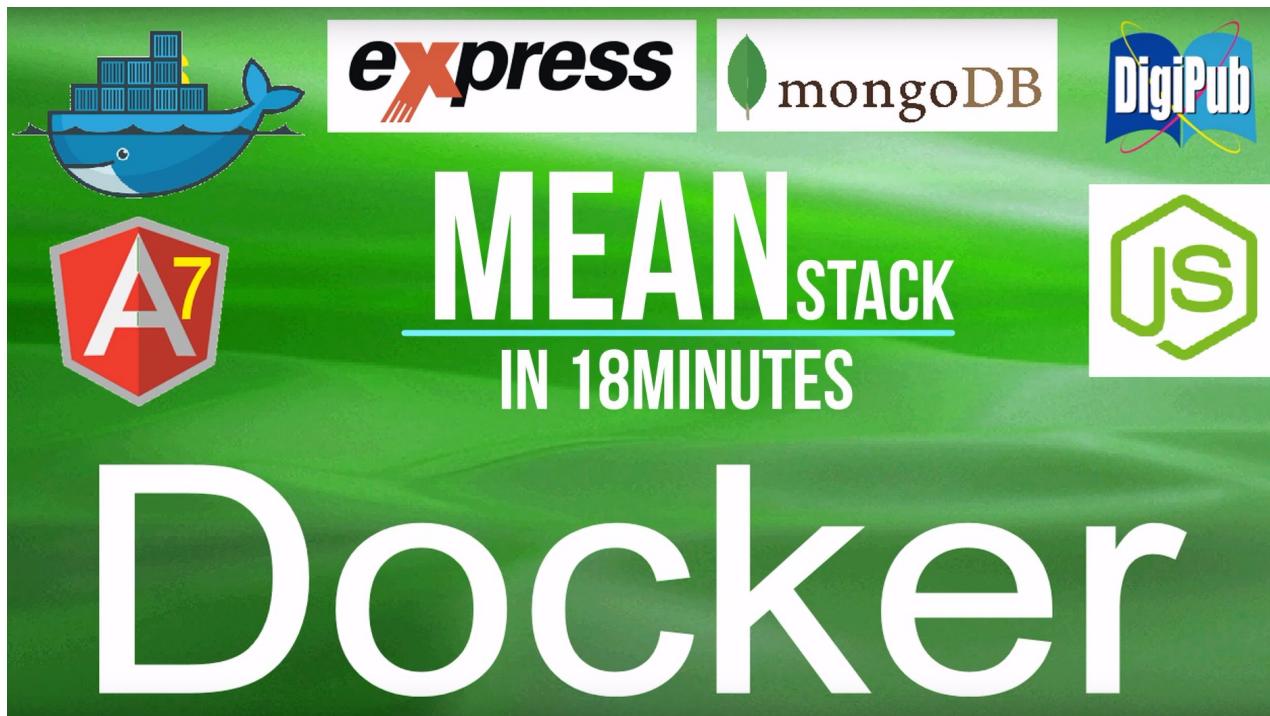
Narration&Reference MEAN Stack on Docker

Base File Name: NarrationReference_Docker_MEAN_Stack_ja_13_en

2019.1.1

MEAN Stack on Docker

by *Shuichi Ohtsu*



Video Explanation(Japanese)

<https://youtu.be/RJsJ6jcJDGM>

Video Explanation(English)

<https://youtu.be/mjKe-FF4gMc>

Full Source Code

https://github.com/Ohtsu/mean_stack_on_docker

Operating environment

This time, we will introduce an example of easily creating environment of MEAN Stack by using Docker on WSL.

MEAN Stack is an acronym for JavaScript based free software which is popular all over the world and is named after it.

M represents MongoDB of the database, *E* represents middleware Express which can easily construct Web server, *A* represents Angular which is a client framework that Google provides free, *N* is JavaScript based Node.js that provides a common basis for clients and servers.

The most important point is that you can build an environment from server to client with JavaScript.

In other words, whatever the device it is, if the browser is decorated it will be able to run.

It clears all the differences in the operating environments that had plagued developers in the past.

Whether in the Windows environment, Linux environment or Mac environment, it can run without problem. Moreover, it includes the server environment.

Furthermore, we do not care about the recipient's device environment. That is, desktop machines or smartphones will work without problems.

With this display, you can avoid having to be conscious of the device environment, especially by using Angular Material.

This time, this function is realized by using Angular Material based on Angular 7.

Desktop display

MEAN Stack on Docker					
New					
Title	Category	Updated	Version	Target	Actions
My first section	docker	12/2/18, 12:00 AM	1.0	blog	Edit View Delete
My third section	docker	12/2/18, 12:00 AM	1.2	blog	Edit View Delete
My first section	docker	12/2/18, 12:00 AM	1.0	blog	Edit View Delete
My second section	c#	12/2/18, 12:00 AM	1.2	blog	Edit View Delete
My third section	docker	12/2/18, 12:00 AM	1.2	blog	Edit View Delete
My first section	docker	12/2/18, 12:00 AM	1.0	blog	Edit View Delete
My second section	c#	12/2/18, 12:00 AM	1.2	blog	Edit View Delete
My third section	docker	12/2/18, 12:00 AM	1.2	blog	Edit View Delete
My first section	docker	12/2/18, 12:00 AM	1.0	blog	Edit View Delete
My second section	c#	12/2/18, 12:00 AM	1.2	blog	Edit View Delete
Items per page:					10 < >
1 - 10 of 24					

Smartphone display

MEAN Stack on Docker						
New						
Title	Category	Updated	VersionTarget	Actions		
My first section	docker	12/2/18, 12:00 AM	1.0	blog	Edit	View
					Delete	
My third section	docker	12/2/18, 12:00 AM	1.2	blog	Edit	View
					Delete	
My first section	docker	12/2/18, 12:00 AM	1.0	blog	Edit	View
					Delete	
My second section	c#	12/2/18, 12:00 AM	1.2	blog	Edit	View
					Delete	
My third section	docker	12/2/18, 12:00 AM	1.2	blog	Edit	View
					Delete	
My first section	docker	12/2/18, 12:00 AM	1.0	blog	Edit	View
					Delete	
My second section	c#	12/2/18, 12:00 AM	1.2	blog	Edit	View
					Delete	
My third section	docker	12/2/18, 12:00 AM	1.2	blog	Edit	View
					Delete	
My first section	docker	12/2/18, 12:00 AM	1.0	blog	Edit	View
					Delete	
My second section	c#	12/2/18, 12:00 AM	1.2	blog	Edit	View
					Delete	

Install Docker-Compose

First, open Ubuntu on the WSL.

Then type `sudo apt update` to get update information.

Next, type `sudo apt upgrade` and upgrade based on this update information.

Next, install *docker-compose*.

Type `sudo apt install docker-compose`.

```
ohtsu@Tiger64:/mnt/c/_myprg/_test$ sudo apt update
[sudo] password for ohtsu:
Get:1 http://security.ubuntu.com/ubuntu bionic-security InRelease [83.2 kB]
Hit:2 http://packages.cloud.google.com/apt cloud-sdk-bionic InRelease
Hit:4 http://archive.ubuntu.com/ubuntu bionic InRelease
Get:5 http://archive.ubuntu.com/ubuntu bionic-updates InRelease [88.7 kB]
Hit:3 https://packages.cloud.google.com/apt kubernetes-xenial InRelease
Get:6 http://archive.ubuntu.com/ubuntu bionic-backports InRelease [74.6 kB]
Fetched 247 kB in 10s (24.6 kB/s)
Reading package lists... Done
Building dependency tree
Reading state information... Done
All packages are up to date.
ohtsu@Tiger64:/mnt/c/_myprg/_test$ sudo apt upgrade
Reading package lists... Done
Building dependency tree
Reading state information... Done
Calculating upgrade... Done
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
ohtsu@Tiger64:/mnt/c/_myprg/_test$ sudo apt install docker-compose
Reading package lists... Done
Building dependency tree
Reading state information... Done
docker-compose is already the newest version (1.17.1-2).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
ohtsu@Tiger64:/mnt/c/_myprg/_test$
```

Installation of client, Rest API and server software

Next, download the source of client, middleware and server software to be used this time.

Open the GitHub site in the browser.

That URL is https://github.com/Ohtsu/mean_stack_on_docker.

Ohtsu / mean_stack_on_docker

No description, website, or topics provided.

Manage topics

2 commits 1 branch 0 releases 1 contributor

Branch: master New pull request

Create new file Upload files Find file Clone or download

Ohtsu First Upload

backend First Upload an hour ago

frontend First Upload an hour ago

README.md Initial commit an hour ago

docker-compose.yml First Upload an hour ago

README.md

mean_stack_on_docker

When the page is displayed, click *Clone or download* on the right side and copy its download address.

Clone with HTTPS ⓘ

Use Git or checkout with SVN using the web URL.

https://github.com/Ohtsu/mean_stack_on_docker

Use SSH

Open in Desktop Open in Visual Studio

Download ZIP

First, create a directory for downloading.

Here I created the __dummy directory in /mnt/c/__myprg.

In this directory,

Type `git clone https://github.com/Ohtsu/mean_stack_on_docker.git`

and download the source programs.

```
ohtsu@Tiger64:/mnt/c/_myprg/_dummy$ git clone https://github.com/Ohtsu/mean_stack_on_docker.git
Cloning into 'mean_stack_on_docker'...
remote: Enumerating objects: 82, done.
remote: Counting objects: 100% (82/82), done.
remote: Compressing objects: 100% (70/70), done.
remote: Total 82 (delta 7), reused 79 (delta 7), pack-reused 0
Unpacking objects: 100% (82/82), done.
ohtsu@Tiger64:/mnt/c/_myprg/_dummy$
```

Next, in this directory, enter `code .` to start Visual Studio Code.

```
$ cd mean_stack_on_docker
/mount/c/_myprg/_dummy$ code .
```

Check the docker-compose file

First, check the contents of the docker-compose file.

This file describes cooperation relationships such as client and server software.

For example, Angular-related software, which is a client software, depends on MongoDB.

```
version: "3.3"
services:
  frontend:
    build:
      context: ./frontend
      dockerfile: Dockerfile
    ports:
      - "8080:80"
    networks:
      - webnet
    depends_on:
      - mongo
      - backend
    links:
      - backend

  mongo:
    image: mongo
    restart: always
    volumes:
      - mongodb-data:/data/db
    environment:
      MONGO_INITDB_ROOT_USERNAME: username
```

```

    MONGO_INITDB_ROOT_PASSWORD: password
  ports:
    - 27017:27017
  networks:
    - webnet

mongo-express:
  container_name: mongo-express
  image: mongo-express
  restart: always
  ports:
    - 8081:8081
  networks:
    - webnet
  environment:
    ME_CONFIG_MONGODB_ADMINUSERNAME: username
    ME_CONFIG_MONGODB_ADMINPASSWORD: password
    ME_CONFIG_BASICAUTH_USERNAME: username
    ME_CONFIG_BASICAUTH_PASSWORD: password
  depends_on:
    - mongo
  links:
    - mongo

backend:
  build:
    context: ./backend
    dockerfile: Dockerfile
  restart: always
  ports:
    - "5000:5000"
  networks:
    - webnet
  depends_on:
    - mongo
  links:
    - mongo

networks:
  webnet:

volumes:
  mongodb-data:

```

Also, middleware, Express related software also depends on MongoDB.

```
version: "3.3"
services:
  frontend:
    build:
      context: ./frontend
      dockerfile: Dockerfile
    ports:
      - "8080:80"
    networks:
      - webnet
    depends_on:
      - mongo
      - backend
    links:
      - backend
  mongo:
```

These are specified with `depends_on` and `links` commands .

```
depends_on:
  - mongo
  - backend
links:
  - backend
```

In addition, for MongoDB, we also specify the administrator user name for initialization and its password.

Please note that these user names and passwords will be used in the MongoDB administration tool that we will introduce next.

```
mongo:
  image: mongo
  restart: always
  volumes:
    - mongodb-data:/data/db
  environment:
    MONGO_INITDB_ROOT_USERNAME: username
    MONGO_INITDB_ROOT_PASSWORD: password
  ports:
    - 27017:27017
  networks:
    - webnet
```

Check the Dockerfile files

First of all, regarding Express related Dockerfile, we copy the files in the current directory into the container and run `npm install`.

```
# stage 1
FROM node:alpine as node
WORKDIR /app
COPY . .
RUN npm install
# RUN npm run build
EXPOSE 5000
ENTRYPOINT [ "node", "server.js" ]
```

Also, port number 5000 is opened.

```
backend:
  build:
    context: ./backend
    dockerfile: Dockerfile
  restart: always
  ports:
    - "5000:5000"
  networks:
    - webnet
  depends_on:
    - mongo
  links:
    - mongo

networks:
  webnet:
```

Next, for Angular files related to clients, we are going to open the Dockerfile first.

The specification of this file is in two stages.

```
# stage 1
FROM node:10.12 as node
WORKDIR /app
COPY . .
RUN npm install
RUN npm run build --prod

# # stage 2
FROM nginx:alpine
COPY --from=node /app/dist/frontend /usr/share/nginx/html
```

That is, first build the same environment in a container and compile it in production mode.

It copies the generated JavaScript compressed file to the home of nginx on lightweight Alpine Linux.

The screenshot shows a code editor interface with two tabs open. The left tab is titled 'backend' and contains the following file structure:

- backend
- models
- Dockerfile
- package-lock.json
- package.json
- server.js

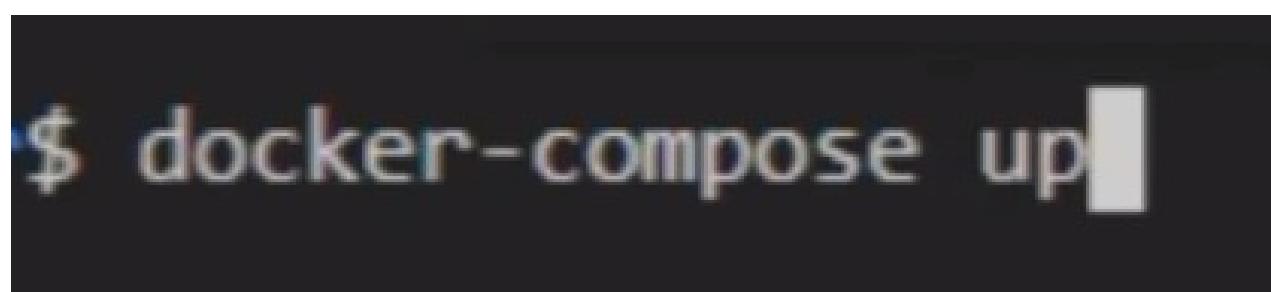
The right tab is titled 'Dockerfile' and contains the following Dockerfile content:

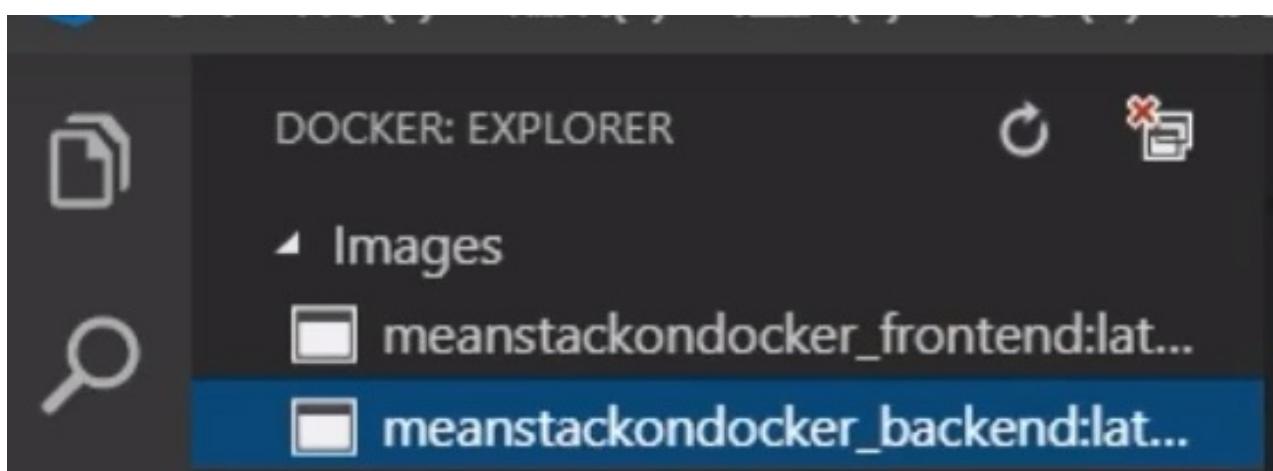
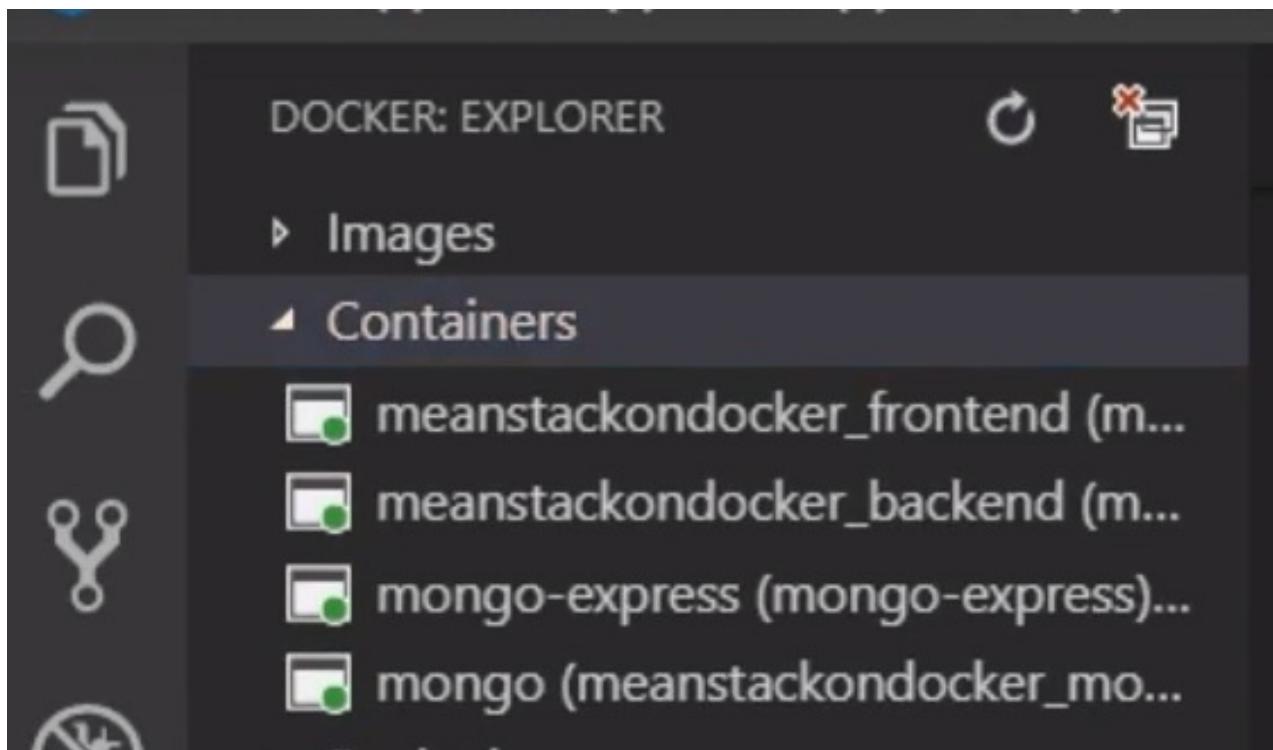
```
1 # stage 1
2 FROM node:alpine as node
3 WORKDIR /app
4 COPY . .
5 RUN npm install
6 # RUN npm run build
7 EXPOSE 5000
8 ENTRYPOINT [ "node", "server.js" ]
9
10 |
```

Please note that since this client software is in the form of Markdown's editing software, it also installs Markdown-to-HTML conversion tool `markdown-it`.

```
"@angular/common": "~7.0.0",
"@angular/compiler": "~7.0.0",
"@angular/core": "~7.0.0",
"@angular/forms": "~7.0.0",
"@angular/http": "~7.0.0",
"@angular/material": "^7.1.0",
"@angular/platform-browser": "~7.0.0",
"@angular/platform-browser-dynamic": "~7.0.0",
"@angular/router": "~7.0.0",
"core-js": "^2.5.4",
"gulp": "^3.9.1",
"hammerjs": "^2.0.8",
"markdown-it": "^8.4.2",
"rxjs": "~6.3.3",
"zone.js": "~0.8.26"
```

First of all, enter `docker-compose up` to create the related docker image.





Next, type `docker-compose down` and delete all related containers once.



First of all, you need to register data in MongoDB for display on the client.

This is a procedure for generating a container image first.

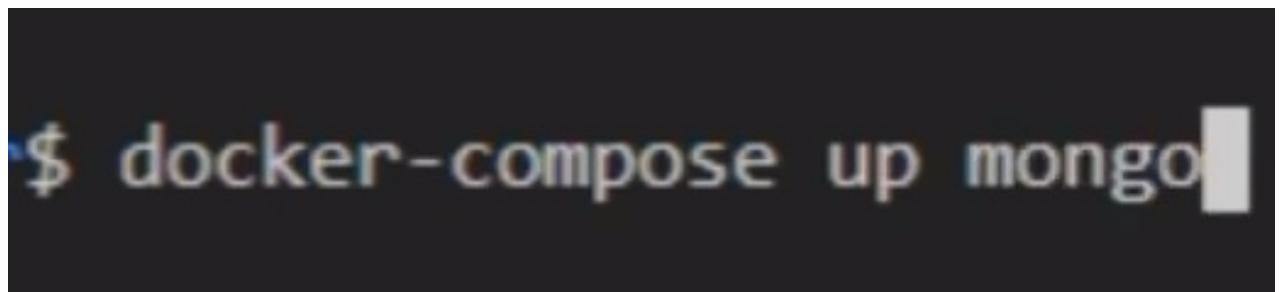
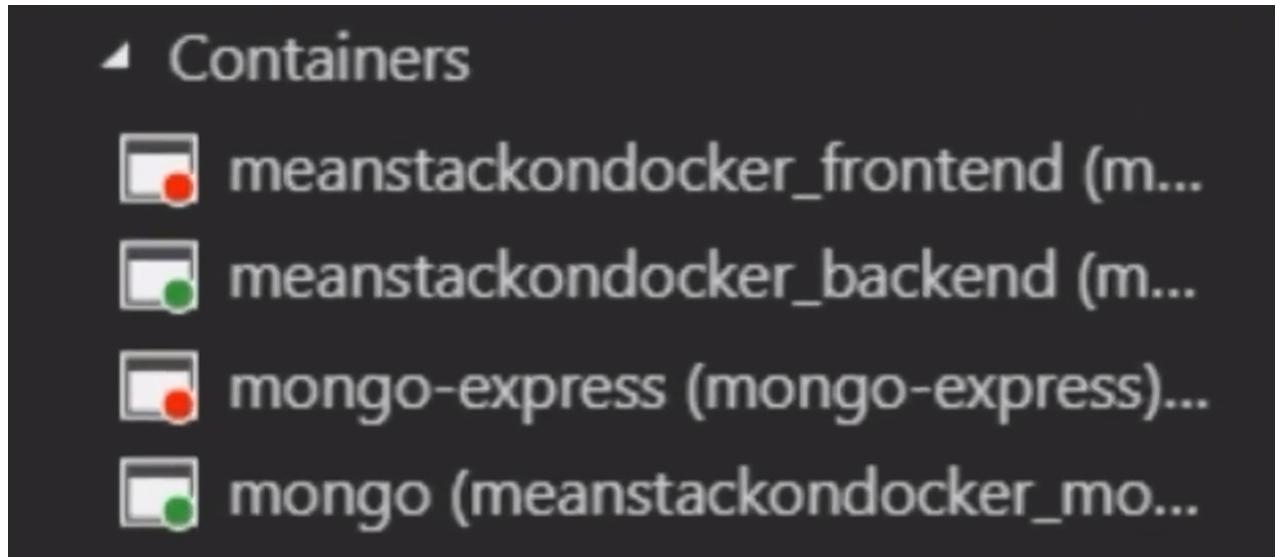
That is, if it takes time to start a database such as MongoDB even if you specify `startup_order` of each software in `docker-compose` by `depends_on` etc, it will be a linkage error while creating the

image It is from.

Enter MongoDB sample data

Next, we are going to input sample data in MongoDB.

First of all, enter `docker-compose down`, delete all the containers, enter `docker-compose up mongo` again, and start only MongoDB.



Here, we use *Studio 3T*<https://studio3t.com> tool for inputting MongoDB data.

It's free during the trial period so it is a very useful tool for MongoDB users.

The screenshot shows the official website for Studio 3T. At the top, there's a navigation bar with links for FEATURES, SOLUTIONS, RESOURCES, CONTACT US, PRICING, and DOWNLOAD. A search icon is also present. The main heading is "The professional MongoDB GUI". Below it, a sub-headline reads "Query faster with the most powerful MongoDB GUI and IDE available for Windows, macOS, and Linux". A video player window is overlaid on the page, showing a screenshot of the Studio 3T interface with a play button. A yellow "DOWNLOAD FREE TRIAL" button is located at the bottom left of the main content area.

First of all, you need to connect to MongoDB.

After starting *Studio 3T*, click the connection icon.

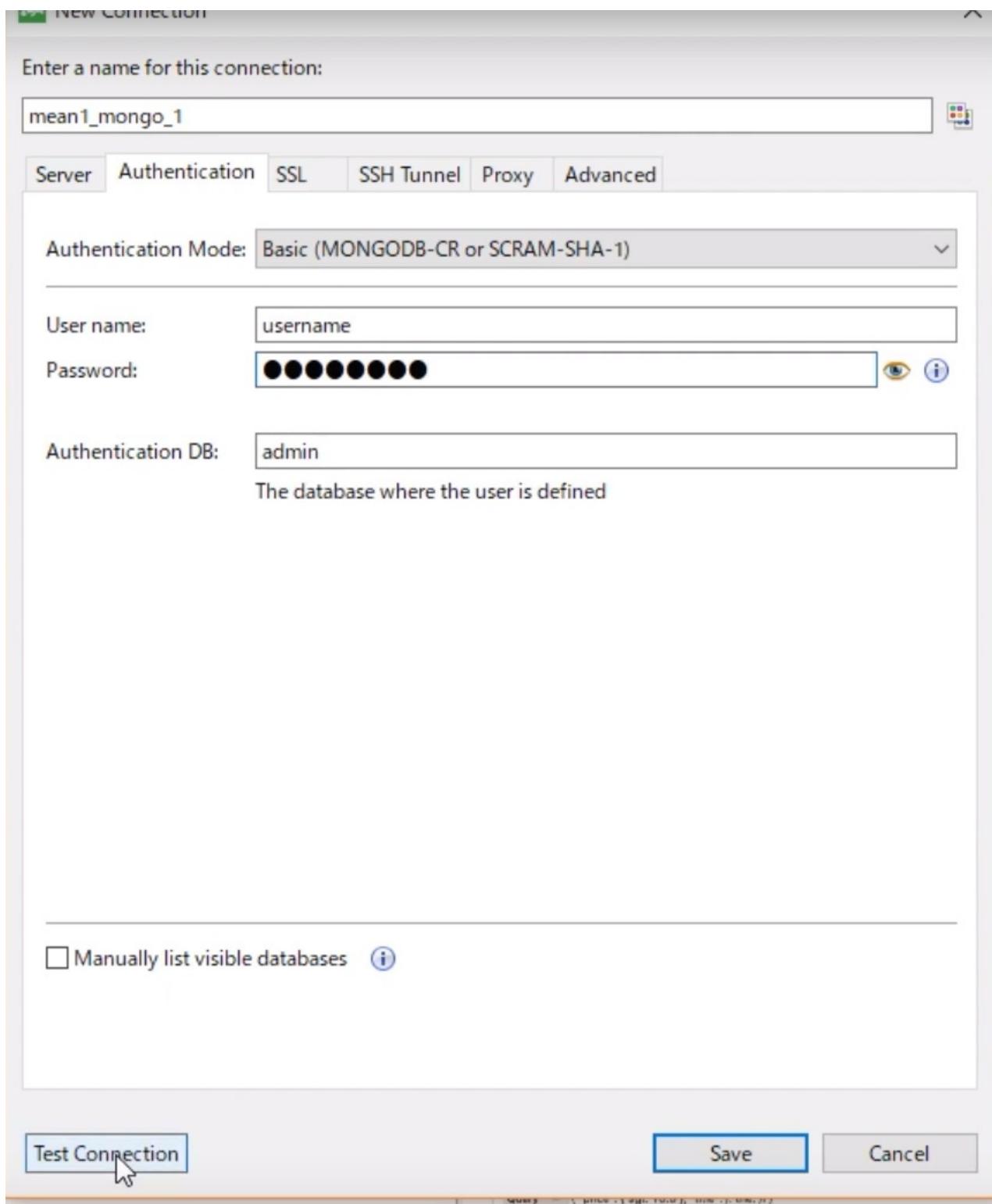
The screenshot shows the Studio 3T application window. The top menu bar includes "Edit", "Database", "Collection", "Index", and several tool icons: Collection, IntelliShell, SQL, Aggregate, Map-Reduce, Export, Import, Users, Roles, Schema, Compare, and Feedback. A green banner at the top states, "come to your trial of full free access to Studio 3T. The trial is available for 6 more days. [How to get a license?](#)". Below the banner is a search bar with the placeholder "Search Open Connections (Ctrl+F) ...". The main panel is titled "Recent Connections" and lists the following connections:

- ▶ mean_mongo_1 (username@localhost:27017)
- ▶ ver51_mongo_1 (username@localhost:27017)
- ▶ ver50_mongo_1 (username@localhost:27017)
- ▶ ver42_mongo_1 (username@localhost:27017)
- ▶ ver41_mongo_1 (username@localhost:27017)
- ▶ ver40_mongo_1 (username@localhost:27017)
- ▶ ver38_mongo_1 (username@localhost:27017)
- ▶ ver37_mongo_1 (username@localhost:27017)

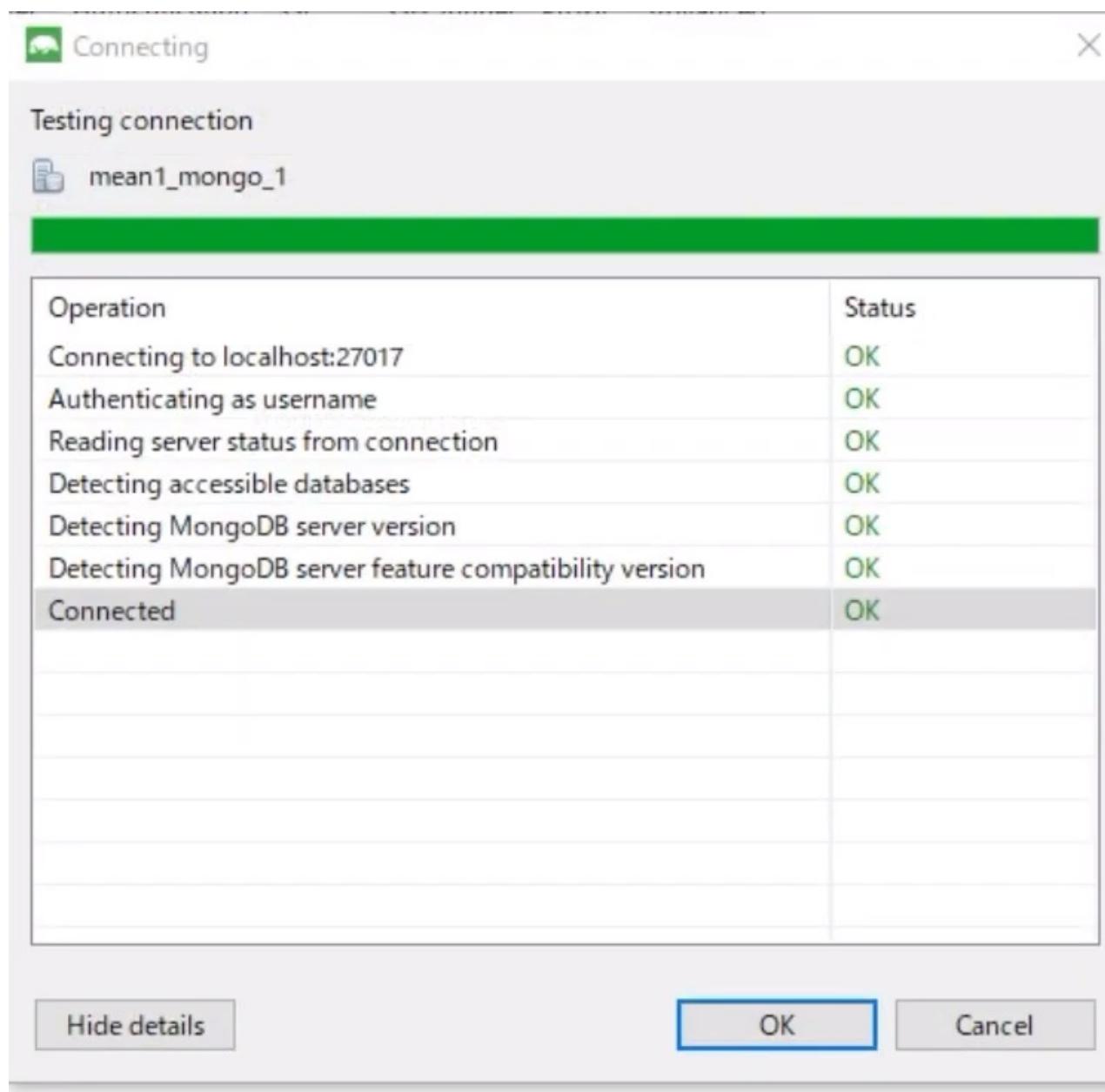
A green "Connect..." button is located at the bottom of this list. Below this section is a "What's New" section with "Older" and "Newer" links.

Then select the new connection, enter the server name appropriately, then select the `Authentications` tab.

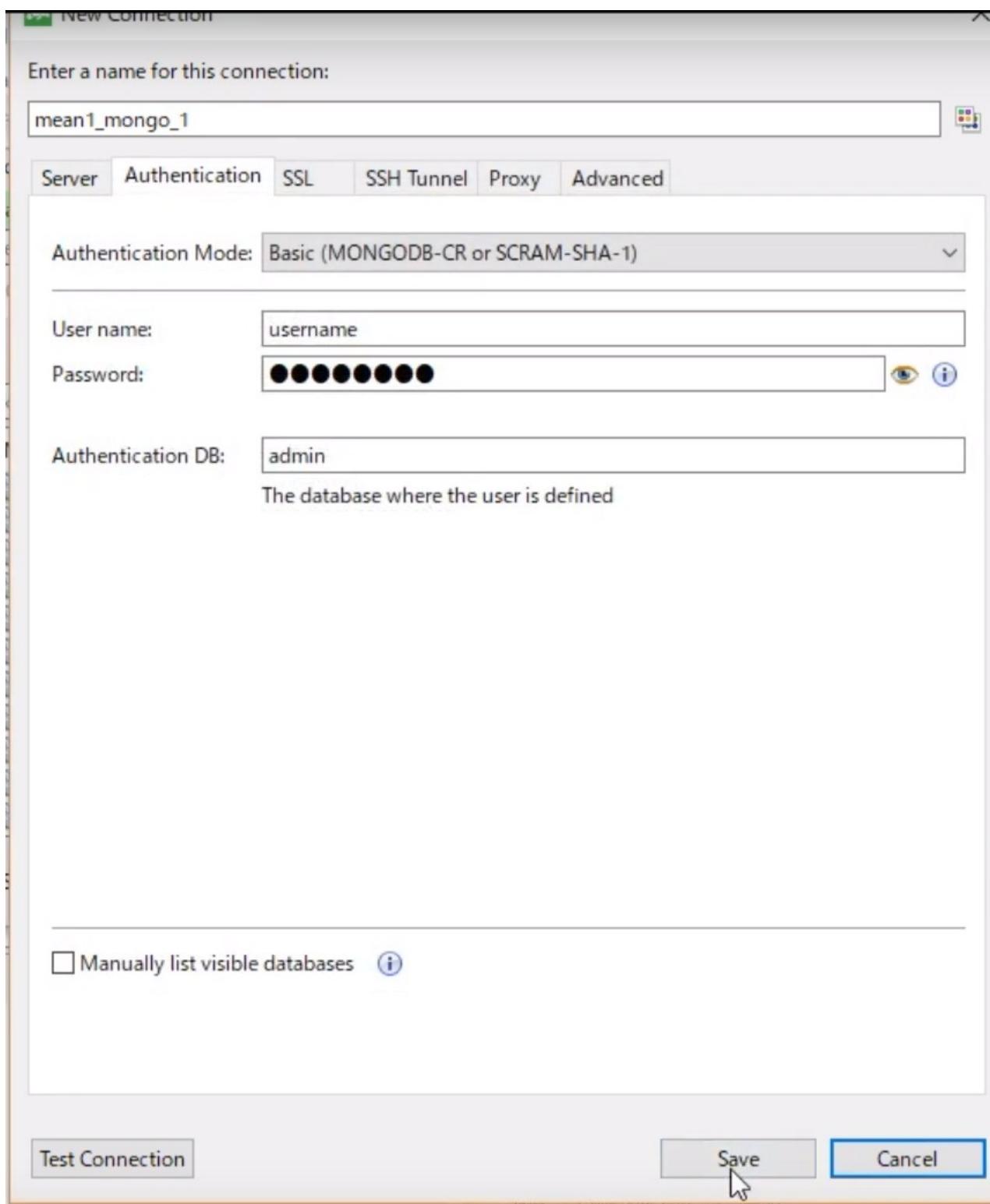
In addition, select `Basic` in `Authentication Mode` and set the user name (username) and password (password) set for the administrator in the `docker-compose` file in the same way.



Then click the `Test Connection` button on the bottom left, and if it is OK, the connection is complete.



Click the **Save** button and save this setting.



In addition, click the **Connect** button and connect.

Connection Manager

New Connection Edit Delete Clone Import Export

Click here to filter connections Showing 28 of 28

Name	DB Server	Security
mean1_mongo_1	localhost:27017	username @ admin
mean_mongo_1	localhost:27017	username @ admin
ver51_mongo_1	localhost:27017	username @ admin
ver50_mongo_1	localhost:27017	username @ admin
ver42_mongo_1	localhost:27017	username @ admin
ver41_mongo_1	localhost:27017	username @ admin
ver40_mongo_1	localhost:27017	username @ admin
ver38_mongo_1	localhost:27017	username @ admin
ver37_mongo_1	localhost:27017	username @ admin
ver36_mongo_1	localhost:27017	username @ admin
ver35_mongo_1	localhost:27017	username @ admin

Show on startup Connect Close

Next, click the **IntelliShell** button, open the MongoDB shell window and enter `use blog` to create a blog database.

Studio 3T for MongoDB - TRIAL LICENSE

Edit Database Collection Index Document GridFS View Help

Connect Collection IntelliShell SQL Aggregate Map-Reduce Export Import Users Roles Schema Compare

Welcome

Recent Connections

- mean1_mongo_1 (username@localhost:27017)
- mean_mongo_1 (username@localhost:27017)
- ver51_mongo_1 (username@localhost:27017)
- ver50_mongo_1 (username@localhost:27017)
- ver42_mongo_1 (username@localhost:27017)
- ver41_mongo_1 (username@localhost:27017)
- ver40_mongo_1 (username@localhost:27017)
- ver38_mongo_1 (username@localhost:27017)

Connect...

IntelliShell: mean1_mongo_1 X

username f21ef08f47b4 (mongod-4.0.4) blog

File New Open Save Run Stop Visual Query Builder

```
1 use blog
2 |
```

Text X

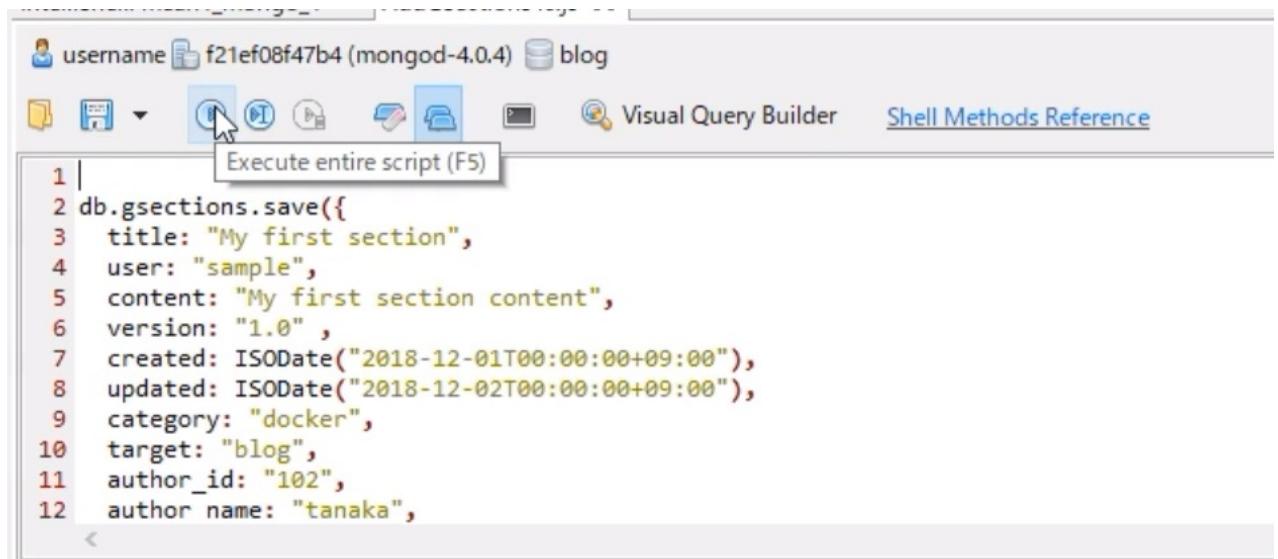
```
1 switched to db blog
2
```

Click the `IntelliShell` button again to register the data. Since this is registered in GitHub, paste the file and register it.

(C) > _myprg > _dummy > mean_stack_on_docker >

mean_stack_on_dockerの検索

名前	更新日時	種類	サイズ
.git	2018/12/31 18:12	ファイル フォルダー	
backend	2018/12/31 18:07	ファイル フォルダー	
frontend	2018/12/31 18:07	ファイル フォルダー	
AddGSection40.js	2018/12/31 18:12	JS ファイル	6 KB
Blog_CreateUser01.js	2018/12/31 18:12	JS ファイル	1 KB

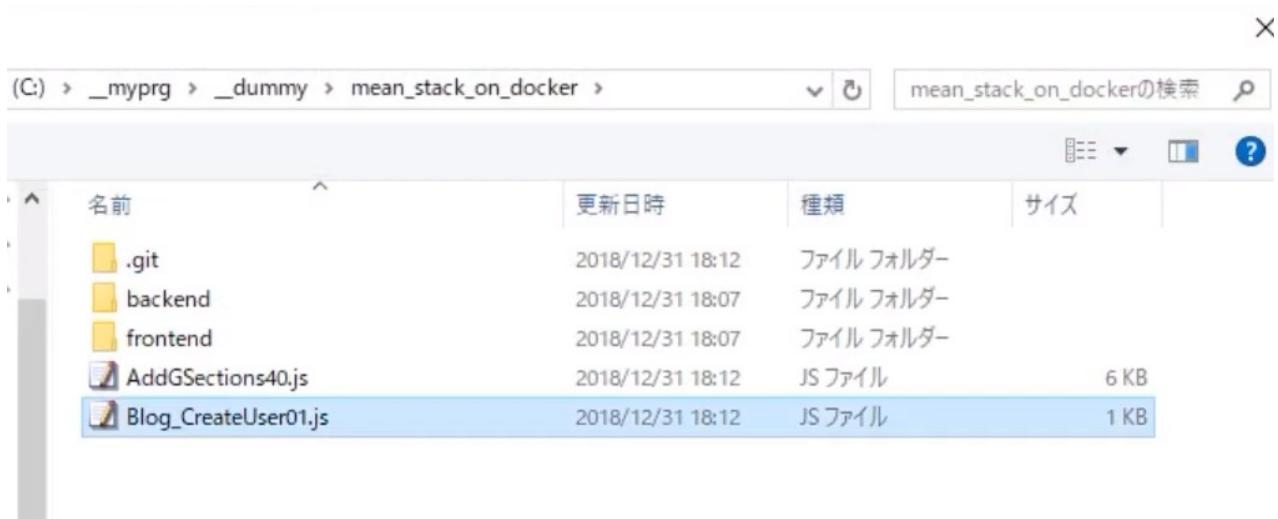


The screenshot shows the MongoDB shell interface. At the top, it displays the connection information: "username f21ef08f47b4 (mongod-4.0.4) blog". Below the header are several icons and buttons, including one highlighted in blue with a tooltip "Execute entire script (F5)". The main area contains a code editor with the following JavaScript code:

```
1 | db.gsections.save({  
2 |   title: "My first section",  
3 |   user: "sample",  
4 |   content: "My first section content",  
5 |   version: "1.0" ,  
6 |   created: ISODate("2018-12-01T00:00:00+09:00"),  
7 |   updated: ISODate("2018-12-02T00:00:00+09:00"),  
8 |   category: "docker",  
9 |   target: "blog",  
10|   author_id: "102",  
11|   author_name: "tanaka",  
12| } )
```

Finally, register the access authority of the reference user (sample).

This is also used as it is registered in GitHub.



IntelliShell: mean1_mongo_1 | AddGSections40.js | Blog_CreateUser01.js

username f21ef08f47b4 (mongod-4.0.4) blog

Visual Query Builder Shell Methods

```

1 db.createUser({
2   user: "sample",
3   pwd: "sample",
4   roles: [
5     { role: "readWrite", db: "blog" }
6   ]
7 })
8 })
9

```

Studio 3T for MongoDB - TRIAL LICENSE

Edit Database Collection Index Document GridFS View Help

Connect Collection IntelliShell SQL Aggregate Map-Reduce Export Import Users Roles Schema Compare Feedback

Welcome to your trial of full free access to Studio 3T. The trial is available for 6 more days. [How to get a license?](#)

mean1_mongo_1 localhost:27017 | IntelliShell: mean1_mongo_1 AddGSections40.js Blog_CreateUser01.js gsections

username mean1_mongo_1 localhost:27017 blog gsections

Query: {}

Projection: {} Sort: {}

Skip: Limit: 50

Result | Query Code | Explain | Table View

gsections > title

_id	title	user	content	version	created	updated
5c29d844a744...	My third section	sample	My third sectio...	1.2	2018-11-30T15...	2018-12-01T15...
5c29d844a744...	My first section	sample	My first sectio...	1.0	2018-11-30T15...	2018-12-01T15...
5c29d844a744...	My second sec...	sample	My second sect...	1.2	2018-11-30T15...	2018-12-01T15...
5c29d844a744...	My third section	sample	My third sectio...	1.2	2018-11-30T15...	2018-12-01T15...
5c29e086ced8...	My first section	sample	My first sectio... content: "My first section content"	1.0	2018-11-30T15...	2018-12-01T15...
5c29e086ced8...	My second sec...	sample	My second sect...	1.2	2018-11-30T15...	2018-12-01T15...
5c29e086ced8...	My third section	sample	My third sectio...	1.2	2018-11-30T15...	2018-12-01T15...
5c29e086ced8...	My first section	sample	My first sectio...	1.0	2018-11-30T15...	2018-12-01T15...
5c29e086ced8...	My second sec...	sample	My second sect...	1.2	2018-11-30T15...	2018-12-01T15...
5c29e086ced8...	My third section	sample	My third sectio...	1.2	2018-11-30T15...	2018-12-01T15...
5c29e086ced8...	My first section	sample	My first sectio...	1.0	2018-11-30T15...	2018-12-01T15...
5c29e086ced8...	My second sec...	sample	My second sect...	1.2	2018-11-30T15...	2018-12-01T15...
5c29e087ced8...	My third section	sample	My third sectio...	1.2	2018-11-30T15...	2018-12-01T15...

Start server and client software

First, stop running MongoDB.

Type docker-compose down.

DOCKER: EXPLORER

package.json

```

34   "@angular/cli": "~7.0.2",
35   "@angular/compiler-cli": "~7.0.0",
36   "@angular/language-service": "~7.0.0",
37   "@types/node": "~8.9.4",
38   "@types/jasmine": "~2.8.8",
39   "@types/jasminewd2": "~2.0.3",
40   "codelyzer": "~4.5.0",
41   "jasmine-core": "~2.99.1",
...

```

問題 出力 デバッグコンソール ターミナル

1: bash

```
ohtsu@Tiger64:/mnt/c/_myprg/_dummy/mean_stack_on_docker$ docker-compose down
```

Next, start all the software.

Type docker-compose up.

```

ohtsu@Tiger64:/mnt/c/_myprg/_dummy/mean_stack_on_docker$ docker-compose up
Creating network "meanstackondocker_webnet" with the default driver
Creating meanstackondocker_mongo_1 ...
Creating meanstackondocker_mongo_1 ... done
Creating meanstackondocker_backend_1 ...
Creating mongo-express ...
Creating mongo-express
Creating meanstackondocker_backend_1 ... done
Creating meanstackondocker_frontend_1 ...
Creating meanstackondocker_frontend_1

```

DOCKER: EXPLORER

package.json

```

34   "@angular/cli": "~7.0.2",
35   "@angular/compiler-cli": "~7.0.0",
36   "@angular/language-service": "~7.0.0",
37   "@types/node": "~8.9.4",
38   "@types/jasmine": "~2.8.8",
39   "@types/jasminewd2": "~2.0.3",
40   "codelyzer": "~4.5.0",
41   "jasmine-core": "~2.99.1",
...

```

問題 出力 デバッグコンソール ターミナル

1: bash

```

0, LE, mongodb-core: 2.1.8" }
mongo_1 | 2018-12-31T09:27:13.187+0000 I NETWORK [listener] connection accepted from
192.168.96.4:34220 #3 (2 connections now open)
mongo_1 | 2018-12-31T09:27:13.199+0000 I NETWORK [conn3] received client metadata fro
m 192.168.96.4:34220 conn3: { driver: { name: "nodejs", version: "3.1.10" }, os: { type: "Linux
", name: "linux", architecture: "x64", version: "4.9.125-linuxkit" }, platform: "Node.js v11.5.
0, LE, mongodb-core: 3.1.9" }
mongo_1 | 2018-12-31T09:27:13.269+0000 I ACCESS [conn3] Successfully authenticated a
s principal sample on blog
mongo_1 | 2018-12-31T09:27:13.297+0000 I ACCESS [conn2] Successfully authenticated a
s principal username on admin
mongo_1 | 2018-12-31T09:27:13.307+0000 I NETWORK [listener] connection accepted from
192.168.96.3:55312 #4 (3 connections now open)
mongo_1 | 2018-12-31T09:27:13.403+0000 I ACCESS [conn4] Successfully authenticated a
s principal username on admin
mongo_1 | 2018-12-31T09:27:13.406+0000 I NETWORK [listener] connection accepted from
192.168.96.3:55314 #5 (4 connections now open)
mongo_1 | 2018-12-31T09:27:13.507+0000 I ACCESS [conn5] Successfully authenticated a
s principal username on admin

```

Confirmation of client software

If MongoDB, Express and Angular are working well, accessing <http://localhost:8080> on the browser allows you to launch *Markdown Editor* which also supports smartphones.

localhost:8080/gsectionlist

Title	Category	Updated	Version	Target	Actions		
My first section	docker	12/2/18, 12:00 AM	1.0	blog	Edit	View	Delete
Headers	Markdown	12/31/18, 5:54 PM	1.2	blog	Edit	View	Delete
My third section	docker	12/2/18, 12:00 AM	1.2	blog	Edit	View	Delete
My first section	docker	12/2/18, 12:00 AM	1.0	blog	Edit	View	Delete
My second section	c#	12/2/18, 12:00 AM	1.2	blog	Edit	View	Delete
My third section	docker	12/2/18, 12:00 AM	1.2	blog	Edit	View	Delete

This software uses the function of Angular Material, and the style is set according to the screen resolution of the desktop machine, smartphone, etc.

In addition, paging function, sort function for each column, etc. are also enabled.

Edit View Delete

Edit View Delete

3

5 

10

25

50

100

250

Items per page

Edit View Delete

1 - 10 of 24



Clicking the arrow in the header of each column sorts by using that column as a key.

New

Title 

My first section

Headers

My third section

My first section

My second section

Title	Category	Updated
Headers	Markdown	12/31/18, 5:54 PM
My second section	c#	12/2/18, 12:00 AM
My second section	c#	12/2/18, 12:00 AM
My second section	c#	12/2/18, 12:00 AM
My second section	c#	12/2/18, 12:00 AM

Click the icon in the lower right to display the next page.

Items per page: 

16 - 20 of 24  

Click the `Edit` button in each column to display the page for correction.

MEAN Stack on Docker

Update Gsection

Gsection Title	User	Created	Updated	Version
Headers	sample	2018-12-01	2018-12-31	1.3
Category	Target			

Content

```
# Header1  
## Header2  
### Header3  
#### Header4  
##### Header5
```

```
* list1  
  * list1-1  
* list2  
  * list2-1
```

I

The number of display lines can also be changed.

Click the View button in each column to display the page on which Markdown was recognized.

MEAN Stack on Docker

View Markdown



Markdown Sample

Syntax

This is syntax

Back

Desktop display

MEAN Stack on Docker					
New					
Title	Category	Updated	Version	Target	Actions
My first section	docker	12/2/18, 12:00 AM	1.0	blog	Edit View Delete
My third section	docker	12/2/18, 12:00 AM	1.2	blog	Edit View Delete
My first section	docker	12/2/18, 12:00 AM	1.0	blog	Edit View Delete
My second section	c#	12/2/18, 12:00 AM	1.2	blog	Edit View Delete
My third section	docker	12/2/18, 12:00 AM	1.2	blog	Edit View Delete
My first section	docker	12/2/18, 12:00 AM	1.0	blog	Edit View Delete
My second section	c#	12/2/18, 12:00 AM	1.2	blog	Edit View Delete
My third section	docker	12/2/18, 12:00 AM	1.2	blog	Edit View Delete
My first section	docker	12/2/18, 12:00 AM	1.0	blog	Edit View Delete
My second section	c#	12/2/18, 12:00 AM	1.2	blog	Edit View Delete

Items per page: [10](#) [1 - 10 of 24](#) [«](#) [»](#)

Smartphone display

MEAN Stack on Docker					
New					
Title	Category	Updated	Version	Target	Actions
My first section	docker	12/2/18, 12:00 AM	1.0	blog	Edit View Delete
My third section	docker	12/2/18, 12:00 AM	1.2	blog	Edit View Delete
My first section	docker	12/2/18, 12:00 AM	1.0	blog	Edit View Delete
My second section	c#	12/2/18, 12:00 AM	1.2	blog	Edit View Delete
My third section	docker	12/2/18, 12:00 AM	1.2	blog	Edit View Delete
My first section	docker	12/2/18, 12:00 AM	1.0	blog	Edit View Delete
My second section	c#	12/2/18, 12:00 AM	1.2	blog	Edit View Delete
My third section	docker	12/2/18, 12:00 AM	1.2	blog	Edit View Delete
My first section	docker	12/2/18, 12:00 AM	1.0	blog	Edit View Delete
My second section	c#	12/2/18, 12:00 AM	1.2	blog	Edit View Delete

Finally

Regarding creation of this system, I thought it could be done easily, but it took me a while to surprise.

It is because it was not well understood how to use `depends_on` and `links` in the Docker-Compose file. In the information on the Web, `links` is said to be deprecated, and since `links` was not described in other implementation examples (probably because it was based on Redis I think), I have not used it from the beginning.

However, no matter how many times you repeat it will not work well. While doing so, I encountered the page of "Can not connect to MongoDB via node.js in Docker" in the following reference, and I tried using `links` as a trial.

Then I could easily connect to MongoDB. For this, it can be said that the timing of connection affects the cooperation between containers.

Fortunately, I remembered the explanation of the page 231 of "Docker Textbook for Programmers 2nd Edition, Basic Information Infrastructure and Infrastructure Building Automated by Code". It describes "Here's what you want to be aware of is when `depends_on` starts a container, It does not control to wait until the application on the container becomes usable because it does not wait until the database server in the dependency is ready."

Therefore, I decided to create a time-consuming Docker image at first, and once I downed Docker-Compose and restarted it.

This makes it possible to use the existing Docker image on the local site, so the connection of each container will be successful.

Everyone please be aware of this point.

Reference

MEAN Stack

- "Angular 6 - MEAN Stack Crash Course - Part 1: Front-end Project Setup And Routing",
https://www.youtube.com/watch?v=x2_bcCZg8vQ
- "Angular 6 - MEAN Stack Crash Course - Part 2: Implementing The Back-end",
https://www.youtube.com/watch?v=a30flH_q5-A&t=774s
- "Angular 6 - MEAN Stack Crash Course - Part 3: Connecting Front-end To Back-end",
<https://www.youtube.com/watch?v=HTqghYMRrtA>
- "Angular 6 - MEAN Stack Crash Course - Part 4: Completing The User Interface",
<https://www.youtube.com/watch?v=PhIhNU5MLqo>
- "Web Application Example of MEAN Stack",
<https://github.com/hi1280/mean-example>

- "Integrating Angular with Node.js RESTful Services",
<https://github.com/DanWahlin/Angular-NodeJS-MongoDB-CustomersService>
- "Node.js+express+MongoDB+Mongooseで簡単なjsonサーバを構築するメモ",
<https://qiita.com/tdomen/items/4ecb15f25bf9c3652f59>
- "Mean stack の構築",
<https://qiita.com/baster/items/5b6a2e49030067b6e55c>
- "MEAN Stack Angular 6 CRUD Web Application",
<https://www.djamware.com/post/5b00bb9180aca726dee1fd6d/mean-stack-angular-6-crud-web-application>

Express

- "サルでも分かるExpressでのjsonAPIサーバーの作り方",
<https://qiita.com/leafa78/items/73cc7160d002a4989416>
- "Node.js, Express, sequelize, React で始めるモダン WEB アプリケーション入門
 (Express/sequelize編)",
<https://qiita.com/tatsurou313/items/2ba0387806b07f442b8c>
- "[Express] Expressの開発環境構築～デバッグ環境構築",
<https://qiita.com/ksh-fthr/items/c22dedbc0952bfdcc808>

MongoDB

- "MongoDB + mongo-expressをDocker Composeでお手軽構築",
<https://qiita.com/gldn/items/2a8486c4d7a42d7a0f1f>
- "Cannot connect to MongoDB via node.js in Docker",
<https://stackoverflow.com/questions/44938344/cannot-connect-to-mongodb-via-node-js-in-docker>

Docker

- "Docker Community Edition for Windows",
<https://store.docker.com/editions/community/docker-ce-desktop-windows>
- "Docker/Kubernetes 実践コンテナ開発入門",
<http://amazon.co.jp/o/ASIN/4297100339/>
- "プログラマのためのDocker教科書 第2版 インフラの基礎知識&コードによる環境構築の自動化",
<http://amazon.co.jp/o/ASIN/4798153222/>
- "Cannot link to a running container started by docker-compose",
<https://stackoverflow.com/questions/36489696/cannot-link-to-a-running-container-started-by-docker-compose>

Node.js

- "Node.js超入門[第2版]",
<http://amazon.co.jp/o/ASIN/4798055220/>

Angular

- "Containerizing Angular with Docker - Dan Wahlin",
<https://www.youtube.com/watch?v=cLT7eUWKZpg&t=1140s>
- "Deploy Angular 5 app in Docker Container in under 10 mins - For local development",
<https://www.youtube.com/watch?v=L2UkQ2CND68&t=178s>
- "Angular5, Angular6, Angular7 Custom Library: Step-by-step guide",
<https://www.udemy.com/angular5-custom-library-the-definitive-step-by-step-guide/>
- "Angular5, Angular6, Angular7用 カスタムライブラリの作成: 完全ステップ・バイ・ステップ・ガイド",
<https://www.udemy.com/angular5-1/>