

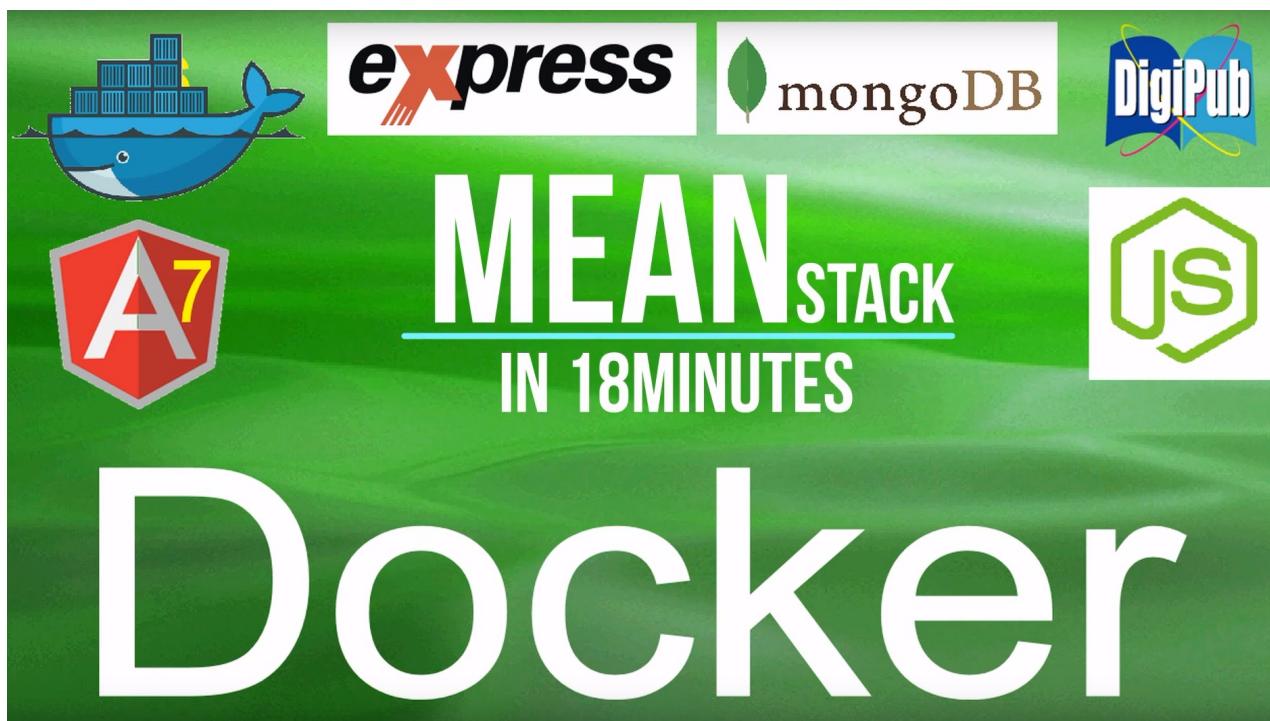
# *Narration&Reference* DockerでMEAN Stackの稼働環境を簡単実現

Base File Name: NarrationReference\_Docker\_MEAN\_Stack\_ja

2018.12.31

## *DockerでMEAN Stackの稼働環境を簡単実現*

by *Shuichi Ohtsu*



ビデオ解説(日本語)

<https://youtu.be/RJsJ6jcJDGM>

ビデオ解説(英語)

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

Full Source Code

[https://github.com/Ohtsu/mean\\_stack\\_on\\_docker](https://github.com/Ohtsu/mean_stack_on_docker)

---

### 動作環境

今回は、WSL上のDockerを利用することにより、MEAN Stackの環境を簡単に作成する例をご紹介いたします。

MEAN Stackとは、世界中に普及している、JavaScriptベースのフリーソフトの頭文字をとった名付けたものです。

Mは、データベースのMongoDB、Eは、Webサーバを簡単に構築できるミドルウェアのExpress、AはGoogle社がフリーで提供しているクライアント用フレームワークであるAngularを表しており、NはJavaScriptをベースとして、クライアント及びサーバの共通基盤を提供するNode.jsを表しています。

最も重要な点は、JavaScriptでサーバからクライアントまでの環境を構築できる点です。

すなわち、どのようなデバイスであろうと、ブラウザが内装されているのであれば、稼働できてしまいます。

従来開発者を悩ましていた、稼働環境の違いをすべてクリアしてしまうのです。

Windows環境であろうと、Linux環境であろうと、あるいはMac環境であろうと、問題なく稼働できてしまいます。しかも、サーバ環境も含めてです。

さらに受け手であるデバイス環境も問いません。すなわち、デスクトップ・マシンであろうが、スマートフォンであろうが、問題なく稼働します。

この表示については、特にAngular Materialを利用することにより、デバイス環境を意識しないで済むことになります。

今回は、Angular7ベースのAngular Materialを利用することにより、この機能を実現しています。

## デスクトップ表示

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

Items per page: [10](#) [1 - 10 of 24](#) [<](#) [>](#)

## スマートフォン表示

MEAN Stack on Docker						
<a href="#" style="color: white; background-color: #007bff; padding: 2px 10px;">New</a>						
Title	Category	Updated	VersionTarget	Actions		
My first section	docker	12/2/18, 12:00 AM	1.0	blog	<a href="#">Edit</a>	<a href="#">View</a>
					<a href="#">Delete</a>	
My third section	docker	12/2/18, 12:00 AM	1.2	blog	<a href="#">Edit</a>	<a href="#">View</a>
					<a href="#">Delete</a>	
My first section	docker	12/2/18, 12:00 AM	1.0	blog	<a href="#">Edit</a>	<a href="#">View</a>
					<a href="#">Delete</a>	
My second section	c#	12/2/18, 12:00 AM	1.2	blog	<a href="#">Edit</a>	<a href="#">View</a>
					<a href="#">Delete</a>	
My third section	docker	12/2/18, 12:00 AM	1.2	blog	<a href="#">Edit</a>	<a href="#">View</a>
					<a href="#">Delete</a>	
My first section	docker	12/2/18, 12:00 AM	1.0	blog	<a href="#">Edit</a>	<a href="#">View</a>
					<a href="#">Delete</a>	
My second section	c#	12/2/18, 12:00 AM	1.2	blog	<a href="#">Edit</a>	<a href="#">View</a>
					<a href="#">Delete</a>	
My third section	docker	12/2/18, 12:00 AM	1.2	blog	<a href="#">Edit</a>	<a href="#">View</a>
					<a href="#">Delete</a>	
My first section	docker	12/2/18, 12:00 AM	1.0	blog	<a href="#">Edit</a>	<a href="#">View</a>
					<a href="#">Delete</a>	
My second section	c#	12/2/18, 12:00 AM	1.2	blog	<a href="#">Edit</a>	<a href="#">View</a>
					<a href="#">Delete</a>	

## docker-composeのインストール

まず、WSL上のUbuntuを開きます。

そして、`sudo apt update`と入力して、更新情報を取得します。

次に、`sudo apt upgrade`と入力して、この更新情報に基づいて、アップグレードを行います。

次に、目的の `docker-compose`をインストールします。

`sudo apt install docker-compose`と入力します。

```
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$
```

## クライアント、Rest API、サーバソフトのインストール

次に今回利用する、クライアント、ミドルウェア並びにサーバソフトのソースをダウンロードします。

ブラウザで、GitHubのサイトを開きます。

そのURLは、[https://github.com/Ohtsu/mean\\_stack\\_on\\_docker](https://github.com/Ohtsu/mean_stack_on_docker)です。

The screenshot shows a GitHub repository page for 'Ohtsu / mean\_stack\_on\_docker'. The 'Code' tab is selected. At the top right, there are buttons for 'Unwatch', 'Star', 'Fork', and 'Edit'. Below the navigation bar, it says 'No description, website, or topics provided.' and 'Manage topics'. A summary bar shows '2 commits', '1 branch', '0 releases', and '1 contributor'. Below this, a table lists files: 'backend', 'frontend', 'README.md', and 'docker-compose.yml', all uploaded at 'an hour ago'. A 'README.md' file is shown below the table. The repository name 'mean\_stack\_on\_docker' is displayed at the bottom.

ページが表示されましたら、右側の *Clone or download*をクリックし、そのダウンロード・アドレスをコピーします。

A modal window titled 'Clone with HTTPS' is open. It contains the text 'Use Git or checkout with SVN using the web URL.' and a text input field containing the URL 'https://github.com/Ohtsu/mean\_stack\_on\_d'. To the right of the input field is a blue button with a white clipboard icon and a hand cursor, indicating it can be clicked to copy the URL. Below the input field are two buttons: 'Open in Desktop' and 'Open in Visual Studio'. At the bottom is a large blue button labeled 'Download ZIP'.

まず、ダウンロード用のディレクトリを作成します。

ここでは、/mnt/c/\_\_myprg 内に \_\_dummy ディレクトリを作成しました。

そして、このディレクトリ内で、

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

と入力して、ソースをダウンロードします。

```
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$
```

次に、このディレクトリ内で、`code .`と入力して、Visual Studio Codeを起動します。

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

## docker-composeファイルの確認

まず、docker-composeファイルの内容を確認します。

このファイルでは、クライアント、サーバソフトなどの連携関係を記述しています。

たとえば、クライアント・ソフトであるAngular関連のソフトは、MongoDBに依存しています。

```
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:

```

**また、ミドルウェアである、Express関連のソフトもMongoDBに依存しています。**

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

これらは、`depends_on`及び`links`のコマンドで指定しています。

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

また、MongoDB関連では、初期設定用の管理者ユーザ名及びそのパスワードも指定しています。

これらのユーザ名やパスワードは、次にご紹介する、MongoDB管理ツールで使用しますので、銘記しておください。

```
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
```

## Dockerfileファイルの確認

まず、Express関連のDockerfileについてですが、カレントディレクトリにあるファイル群をコンテナ内にコピーし、`npm install`を実行しています。

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

また、ポート番号5000を公開しています。

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

networks:
  webnet:
```

次に、クライアント関連のAngularファイルについてですが、まずDockerfileを開いてみます。

```
# 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
```

このファイルの指定は2段階になっています。

すなわち、まず同一の環境をコンテナ内に構築し、プロダクションモードでコンパイルします。

その生成されたJavaScript圧縮ファイルを、軽量のAlpineLinux上のnginxのホームにコピーします。

The screenshot shows a code editor interface with a sidebar on the left containing a file tree. The tree includes a 'backend' folder, which contains 'models', 'Dockerfile', 'package-lock.json', 'package.json', and 'server.js'. The 'Dockerfile' file is currently selected and open in the main editor area. The Dockerfile content is as follows:

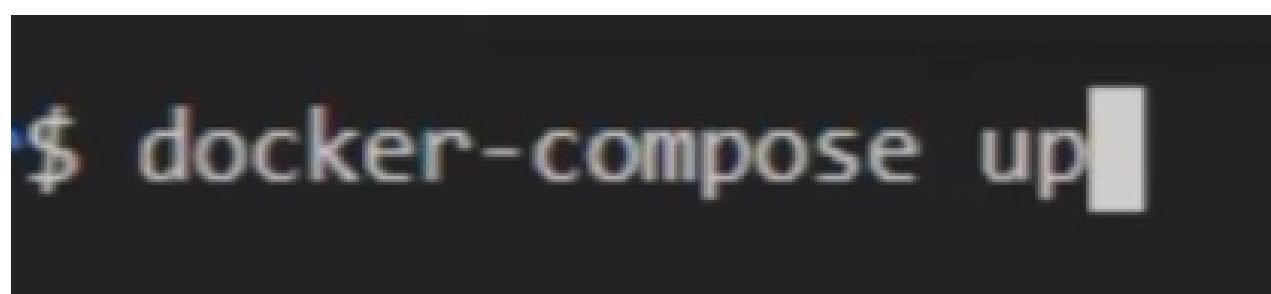
```
1 # stage 1
2 FROM node:alpine as node
3 WORKDIR /app
4 COPY . .
5 RUN npm install
6 # RUN npm run build
7 EXPOSE 5000
8 ENTRYPOINT [ "node", "server.js" ]
9
10 |
```

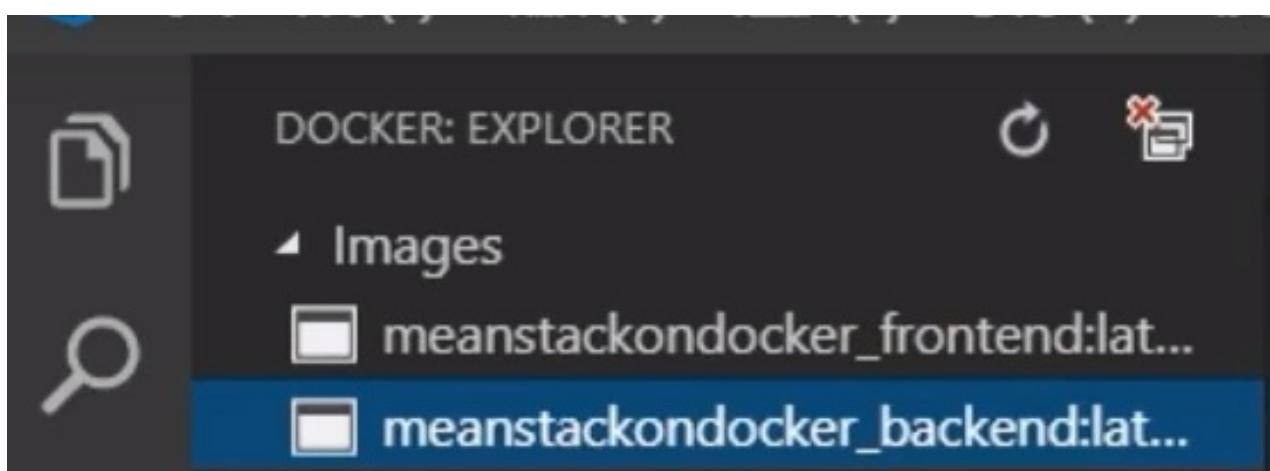
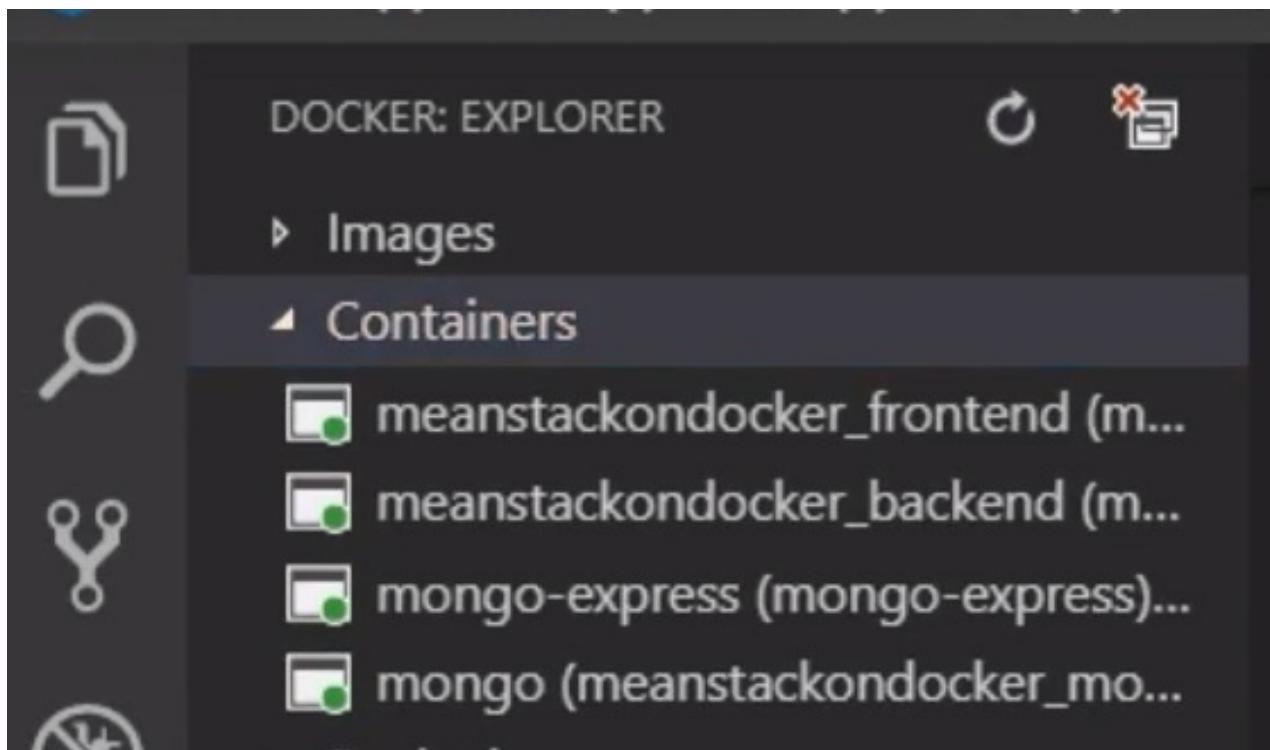
なお、このクライアントソフトは、Markdownの編集ソフトという形になっていますので、MarkdownからHTMLへの変換ツールである`markdown-it`もインストールしていることに注意してください。

The screenshot shows a code editor interface with a file named 'package.json' open. The content of the file is as follows:

```
"@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"
```

まず、`docker-compose up`と入力し、関連のDockerイメージを作成します。





次に、`docker-compose down`と入力し、一旦すべての関連コンテナを削除します。

```
$ docker-compose down
```

まず、クライアントでの表示用に、MongoDBにデータを登録する必要があります。

これは、コンテナ・イメージを先に生成するための手手続きです。

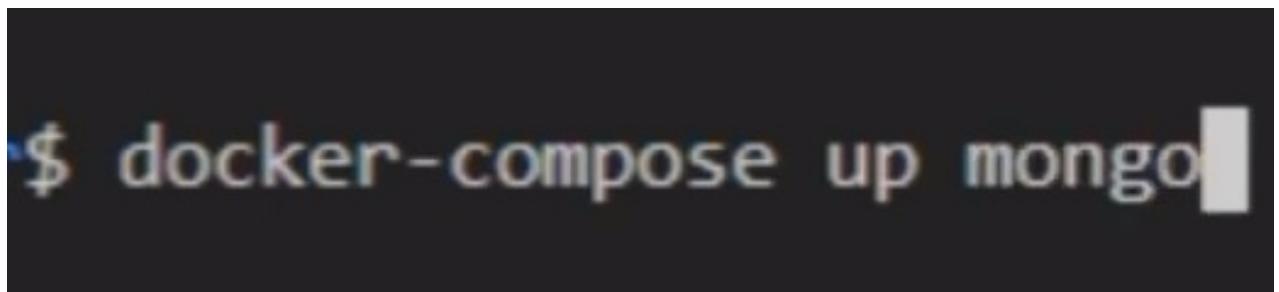
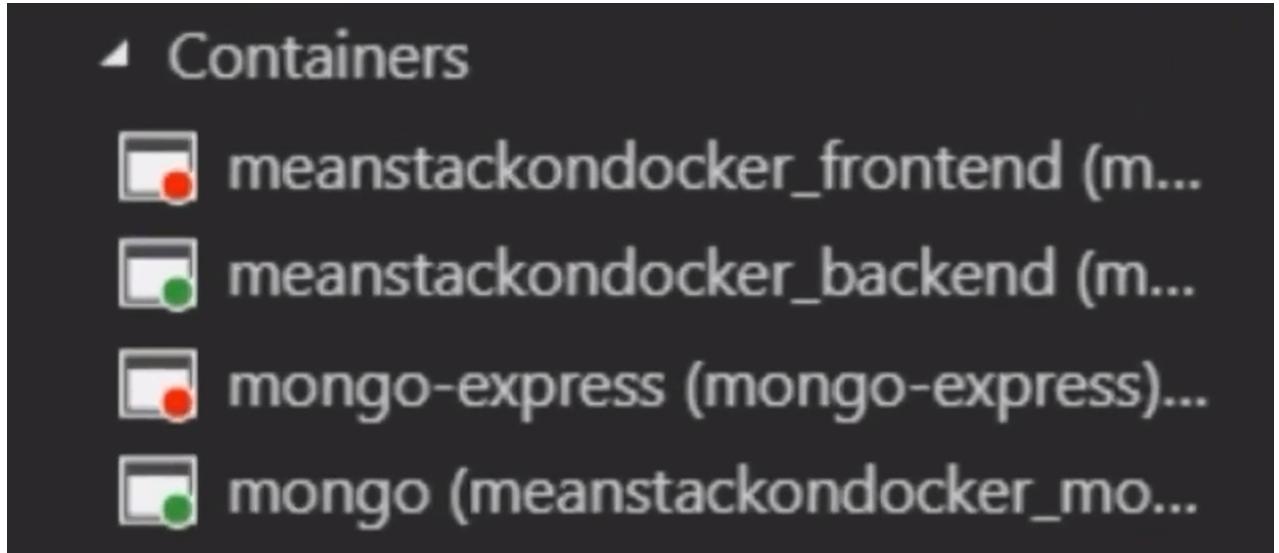
というのは、`docker-compose`において各ソフトウェアの起動順序を`depends_on`などによって、指定してもMongoDBなどのデータベースの起動に時間がかかる場合は、イメージを作成中に連携エラーとな

ってしまうからです。

## MongoDBデータの入力

次に、MongoDBにサンプルデータを入力することにします。

まず、一旦、`docker-compose down`と入力して、すべてのコンテナを削除し、再度`docker-compose up mongo`と入力し、MongoDBだけを起動します。



ここでは、MongoDBデータの入力用に、*Studio 3T*というツールを利用しています。

試用期間中はフリーですのでMongoDBユーザにとっては大変便利なツールです。

The screenshot shows the official website for Studio 3T. The header features the Studio 3T logo and navigation links for FEATURES, SOLUTIONS, RESOURCES, CONTACT US, PRICING, and DOWNLOAD. A search bar is also present. The main content area has a green header with the text "The professional MongoDB GUI". Below it, a sub-section 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.

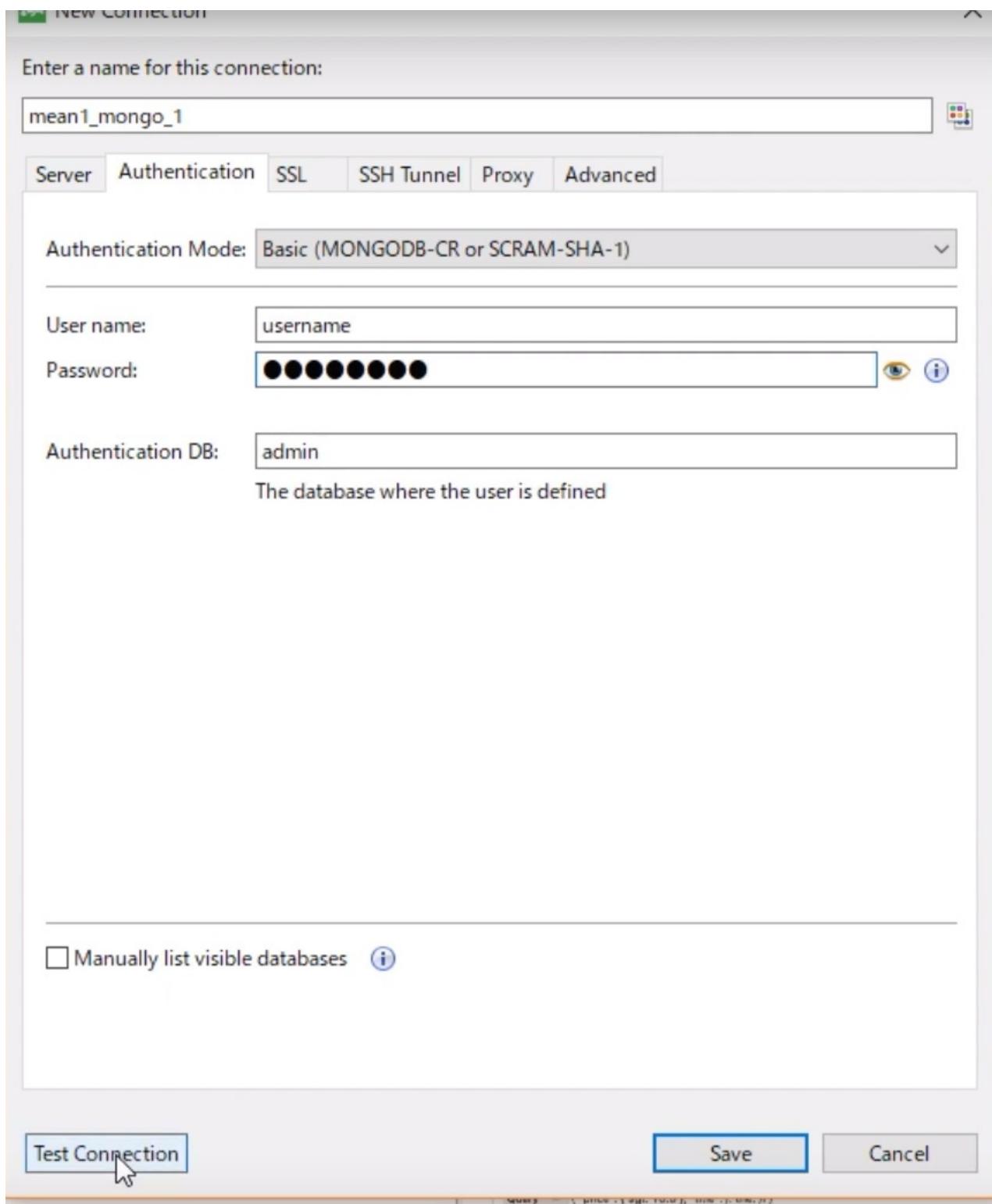
まず、MongoDBに接続する必要があります。

Studio 3Tを起動後、コネクション・アイコンをクリックします。

The screenshot shows the Studio 3T application window. The top menu bar includes "Edit", "Database", "Collection", "Index", and several tool icons: Connect, 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 content area is titled "Recent Connections" and lists eight connection entries: 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), and ver37\_mongo\_1 (username@localhost:27017). A "Connect..." button is located below the recent connections list. At the bottom, there is a "What's New" section with "Older" and "Newer" links.

そして新規コネクションを選択し、サーバ名を適当に入力した後、Authenticationsタブを選択します。

そしてさらに、Authentication ModeにおいてBasicを選択し、docker-composeファイルで管理者用に設定している、ユーザ名(username)とパスワード(password)をここにも同様に設定します。



そして、左下のTest Connectionボタンをクリックして、OKであれば、接続完了です。

Connecting X

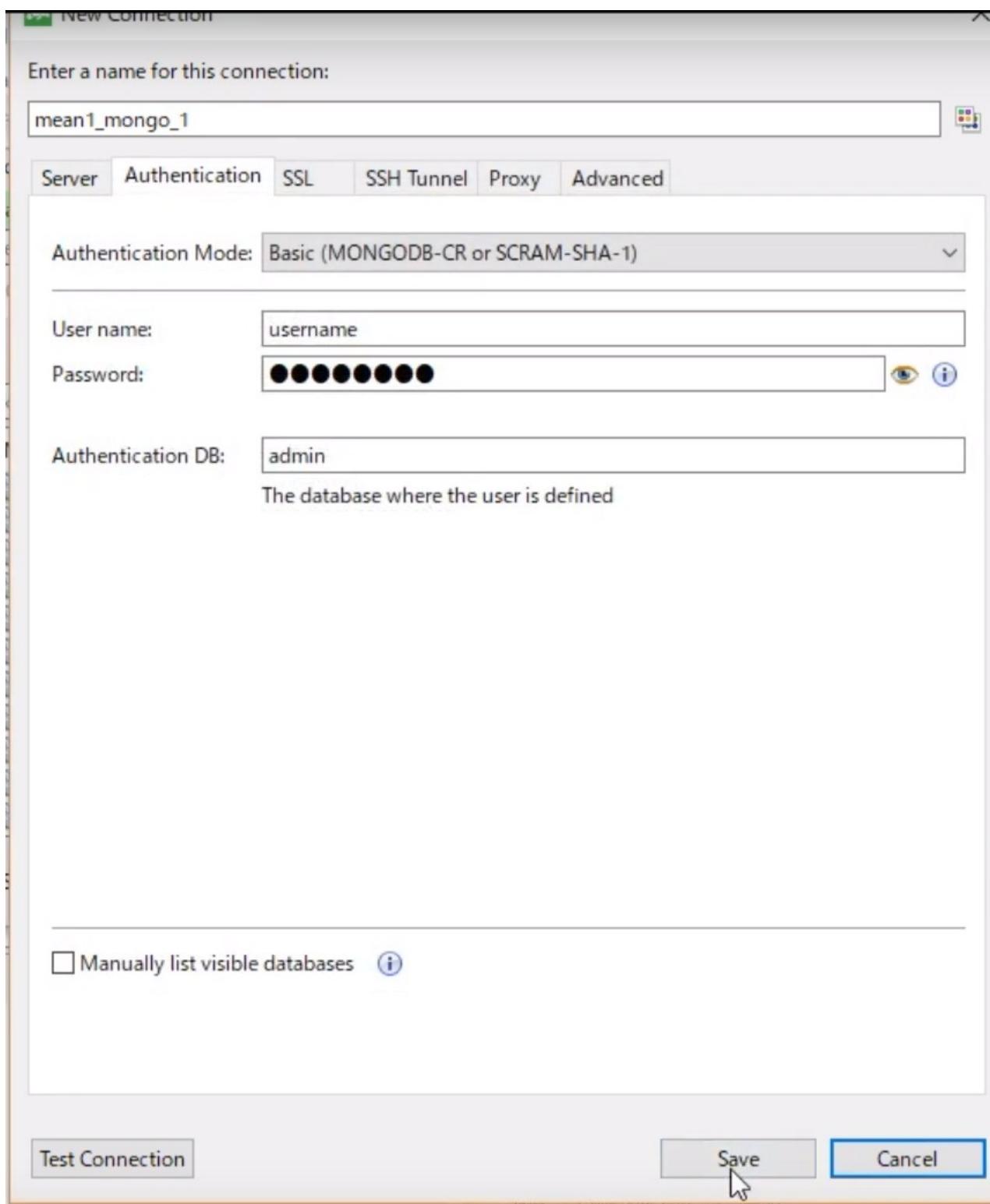
Testing connection

mean1\_mongo\_1

Operation	Status
Connecting to localhost:27017	OK
Authenticating as username	OK
Reading server status from connection	OK
Detecting accessible databases	OK
Detecting MongoDB server version	OK
Detecting MongoDB server feature compatibility version	OK
Connected	OK

Hide details OK Cancel

Saveボタンをクリックし、この設定を保存します。



さらに、Connectボタンをクリックし、接続します。

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

次に、IntelliShellボタンをクリックし、MongoDBシェルウィンドウを開き、`use blog`と入力し、blogデータベースを作成します。

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

username f21ef08f47b4 (mongod-4.0.4) blog

Visual Query Builder

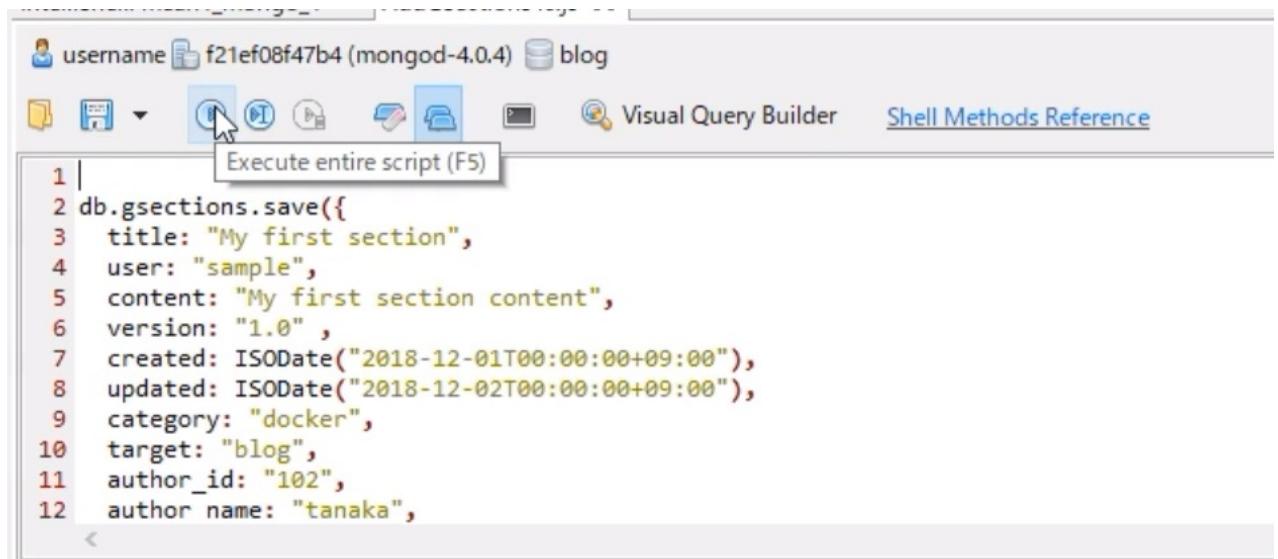
```
1 use blog
2
```

Text

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

再度、IntelliShellボタンをクリックし、データを登録します。これはGitHubに登録しておりますので、そのファイルを貼り付けて登録します。

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

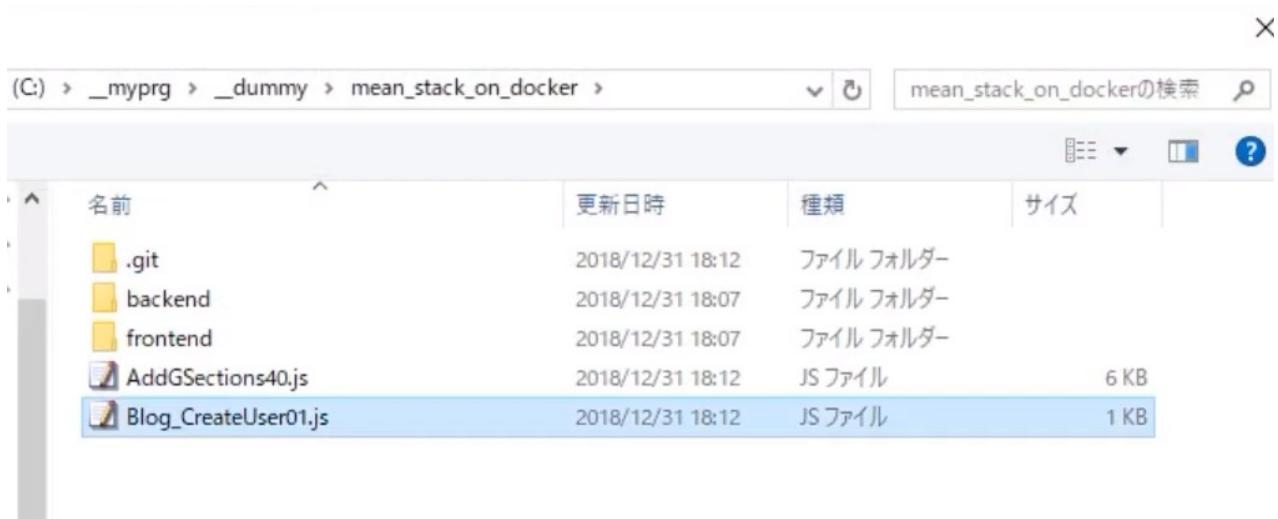


The screenshot shows the MongoDB shell interface. The top bar displays the connection information: "username f21ef08f47b4 (mongod-4.0.4) blog". Below the bar are several icons and buttons, including one highlighted in blue with a tooltip "Execute entire script (F5)". The main area contains a code editor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

最後に、参照ユーザ(sample)のアクセス権限を登録します。

これもGitHubに登録してあるものをそのまま利用します。



```

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

```

_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...	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...

## サーバ及びクライアントソフトの起動

まず、起動中のMongoDBを停止します。

docker-compose downと入力します。

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
```

次に、すべてのソフトウェアを起動します。

docker-compose upと入力します。

```
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
```

## クライアントソフトの確認

データベース、Express並びにAngularがうまく連携していれば、<http://localhost:8080>にブラウザ上でアクセスすれば、スマホにも対応したMarkdown Editorが起動できます。

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

このソフトは、Angular Materialの機能を利用しておおり、デスクトップマシンやスマートフォンなどの画面解像度に応じて、スタイルが設定されているようになっています。

また、ページング機能、各列ごとのソート機能なども有効になっています。  
表示行数も変更可能です。

Edit View Delete

Edit View Delete

3

5 

10

25

50

100

250

Items per page

View Delete

View Delete

View Delete

1 - 10 of 24



各列のヘッダの矢印をクリックすると、その列をキーとしてソートが行われます。

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

右下のアイコンをクリックすると、次ページが表示されます。



各列のEditボタンをクリックすると、修正用のページが表示されます。

## 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

各列のViewボタンをクリックすると、Markdownが認識されたページが表示されます。

## MEAN Stack on Docker

### View Markdown



## Markdown Sample

### Syntax

This is syntax

Back

デスクトップ表示

## MEAN Stack on Docker

New

Title	Category	Updated	Version	Target	Actions		
My first section	docker	12/2/18, 12:00 AM	1.0	blog	<a href="#">Edit</a>	<a href="#">View</a>	<a href="#">Delete</a>
My third section	docker	12/2/18, 12:00 AM	1.2	blog	<a href="#">Edit</a>	<a href="#">View</a>	<a href="#">Delete</a>
My first section	docker	12/2/18, 12:00 AM	1.0	blog	<a href="#">Edit</a>	<a href="#">View</a>	<a href="#">Delete</a>
My second section	c#	12/2/18, 12:00 AM	1.2	blog	<a href="#">Edit</a>	<a href="#">View</a>	<a href="#">Delete</a>
My third section	docker	12/2/18, 12:00 AM	1.2	blog	<a href="#">Edit</a>	<a href="#">View</a>	<a href="#">Delete</a>
My first section	docker	12/2/18, 12:00 AM	1.0	blog	<a href="#">Edit</a>	<a href="#">View</a>	<a href="#">Delete</a>
My second section	c#	12/2/18, 12:00 AM	1.2	blog	<a href="#">Edit</a>	<a href="#">View</a>	<a href="#">Delete</a>
My third section	docker	12/2/18, 12:00 AM	1.2	blog	<a href="#">Edit</a>	<a href="#">View</a>	<a href="#">Delete</a>
My first section	docker	12/2/18, 12:00 AM	1.0	blog	<a href="#">Edit</a>	<a href="#">View</a>	<a href="#">Delete</a>
My second section	c#	12/2/18, 12:00 AM	1.2	blog	<a href="#">Edit</a>	<a href="#">View</a>	<a href="#">Delete</a>

Items per page: 10 1 - 10 of 24

< >

## スマートフォン表示

### MEAN Stack on Docker

New

Title	Category	Updated	Version	Target	Actions		
My first section	docker	12/2/18, 12:00 AM	1.0	blog	<a href="#">Edit</a>	<a href="#">View</a>	<a href="#">Delete</a>
My third section	docker	12/2/18, 12:00 AM	1.2	blog	<a href="#">Edit</a>	<a href="#">View</a>	<a href="#">Delete</a>
My first section	docker	12/2/18, 12:00 AM	1.0	blog	<a href="#">Edit</a>	<a href="#">View</a>	<a href="#">Delete</a>
My second section	c#	12/2/18, 12:00 AM	1.2	blog	<a href="#">Edit</a>	<a href="#">View</a>	<a href="#">Delete</a>
My third section	docker	12/2/18, 12:00 AM	1.2	blog	<a href="#">Edit</a>	<a href="#">View</a>	<a href="#">Delete</a>
My first section	docker	12/2/18, 12:00 AM	1.0	blog	<a href="#">Edit</a>	<a href="#">View</a>	<a href="#">Delete</a>
My second section	c#	12/2/18, 12:00 AM	1.2	blog	<a href="#">Edit</a>	<a href="#">View</a>	<a href="#">Delete</a>
My third section	docker	12/2/18, 12:00 AM	1.2	blog	<a href="#">Edit</a>	<a href="#">View</a>	<a href="#">Delete</a>
My first section	docker	12/2/18, 12:00 AM	1.0	blog	<a href="#">Edit</a>	<a href="#">View</a>	<a href="#">Delete</a>
My second section	c#	12/2/18, 12:00 AM	1.2	blog	<a href="#">Edit</a>	<a href="#">View</a>	<a href="#">Delete</a>

# 最後に

このシステムの作成については、簡単にできると考えていましたが、意外に時間がかかってしまいました。

その原因は、Docker-Composeファイルにおける`depends_on`や`links`の使い方がよくわからていなかったからによる。Web上の情報では、`links`はいずれ廃止されるとのことであり、また他の実装例でも`links`は記述されていなかったので(たぶんRedisをベースとしていたことが影響していると思います)、当初から使用していませんでした。

ところが何度も繰り返してもうまく稼働しません。そういううちに、下記参考文献の"Cannot connect to MongoDB via node.js in Docker"のページに出会い、試しに`links`を使用してみました。

すると簡単にMongoDBに接続することができました。これには、各コンテナ間の連携に、接続のタイミングが影響していることが伺えます。

幸い、下記参考文献上の『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』の231ページにある「ここで注意していただきたいのは、`depends_on`がコンテナの開始の順序を制御するだけで、コンテナ上のアプリケーションが利用可能になるまで待つという制御を行わない点です。つまり、依存関係にあるデータベースサーバの準備が整うまで待つわけではないため、アプリケーション側で対策する必要があります。」との記述をかすかに覚えていました。

そこで、時間のかかるDockerイメージの作成を行い、そして一旦Docker-Composeを中断し、再起動するという方法をとることにしました。

これにより、ローカル上の既存のDockerイメージを利用できるので、各コンテナの接続がうまくいくことになりました。

皆さんもぜひこの点に注意してください。

## 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](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](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=PhlhNU5MLqo>
- "Web Application Example of MEAN Stack",  
<https://github.com/hil280/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/leafia78/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/>