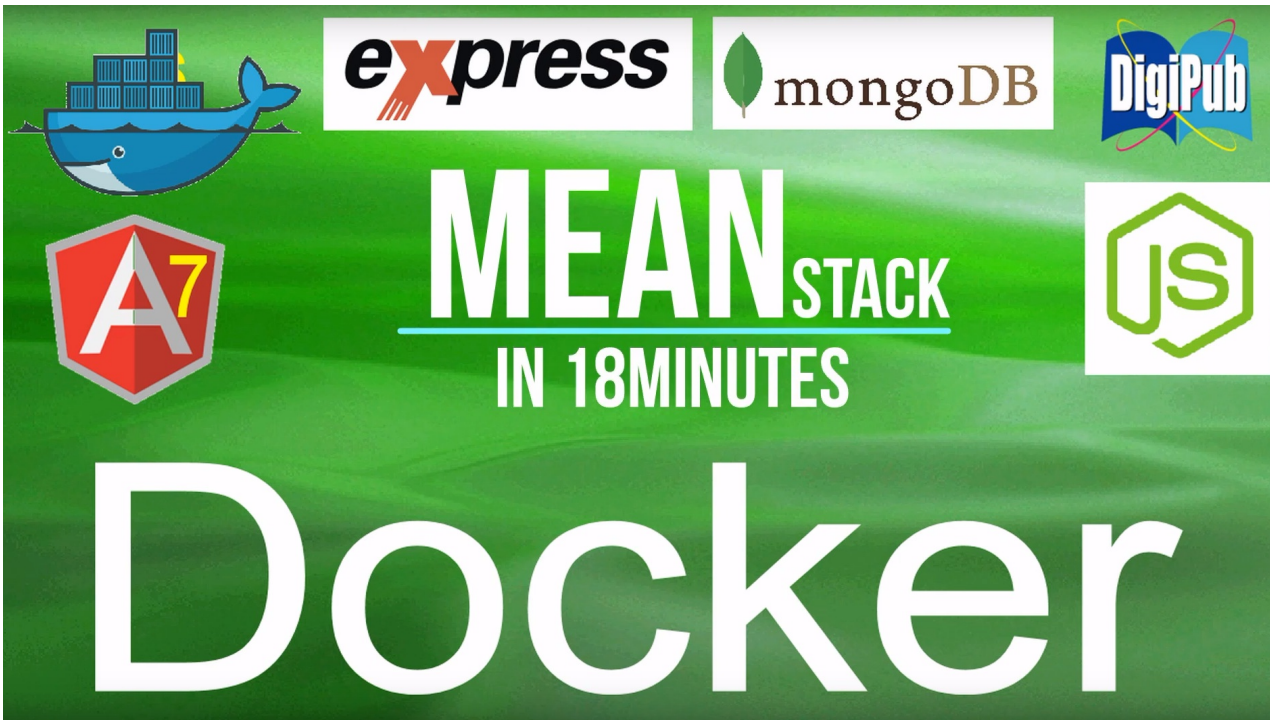
Narration&Reference MEAN Stack on Docker

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.

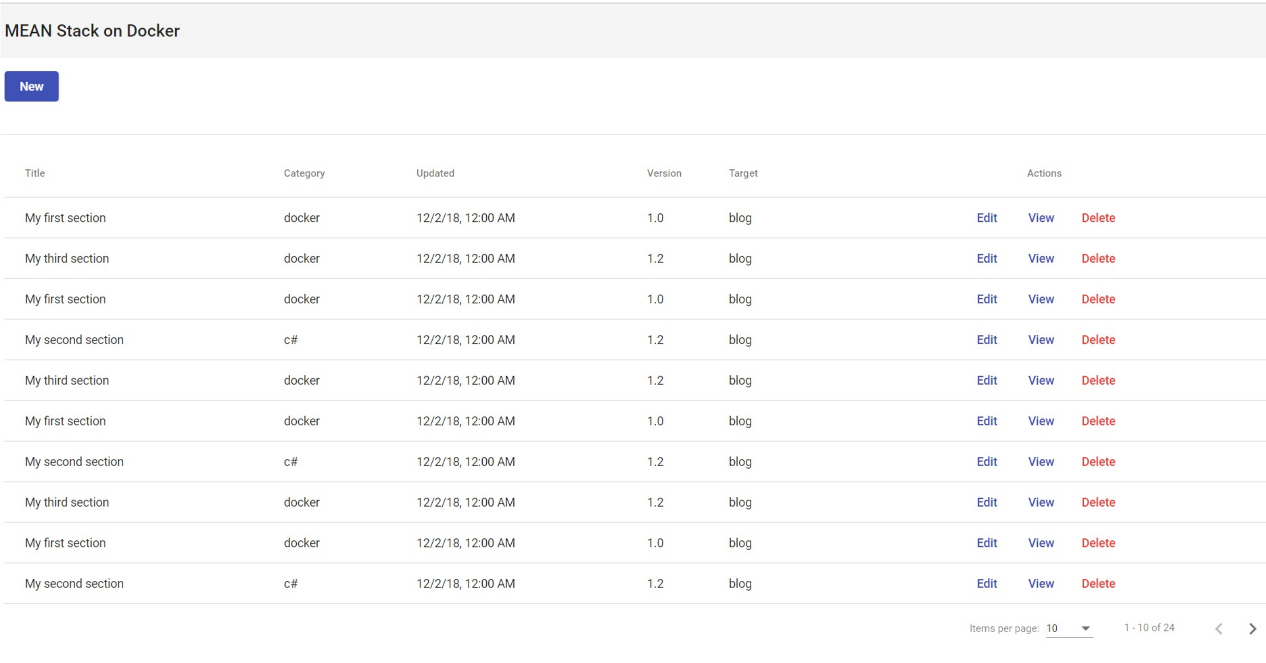
Whether in the Windows environment, Linux environment or Mac environment, it can run without problem. Moreover, it includes the server environment.

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.

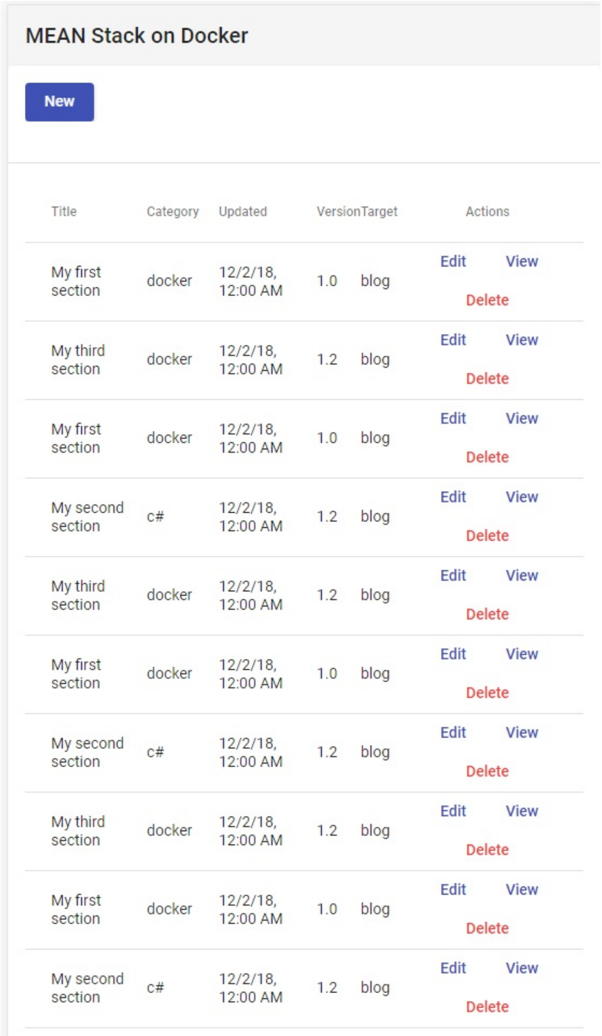
This time, this function is realized by using Angular Material based on Angular 7.

Desktop display



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

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

---

Install Docker-Compose

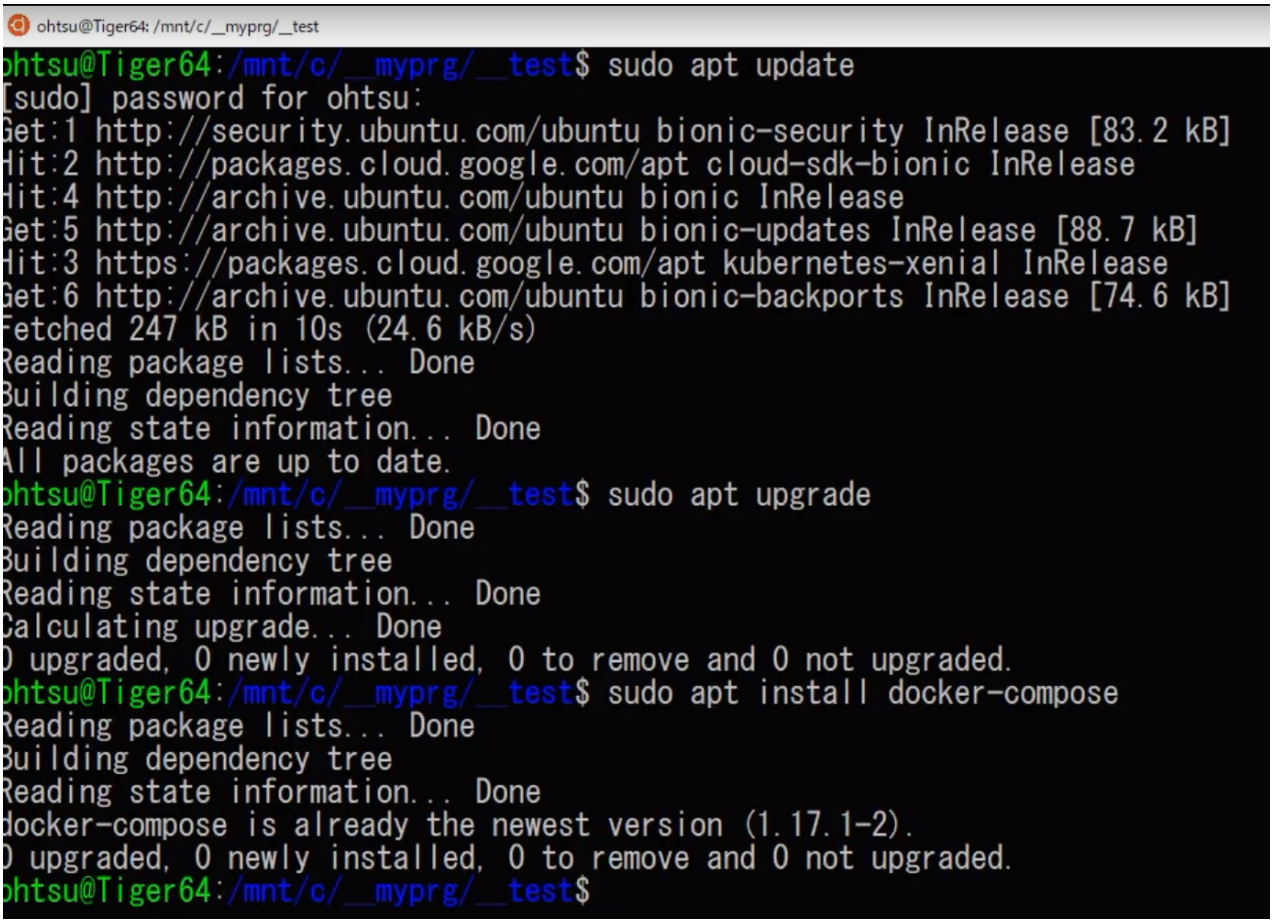
First, open Ubuntu on the WSL.

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`.

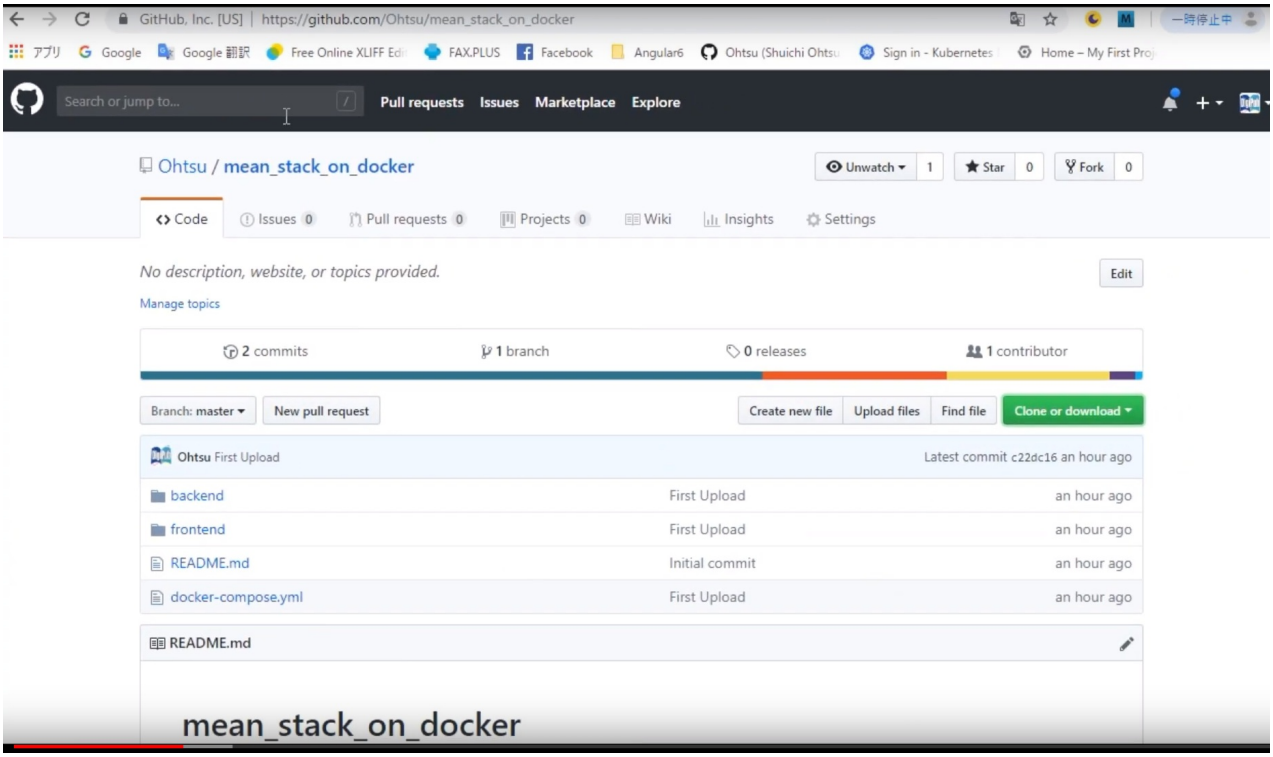


Installation of client, Rest API and server software

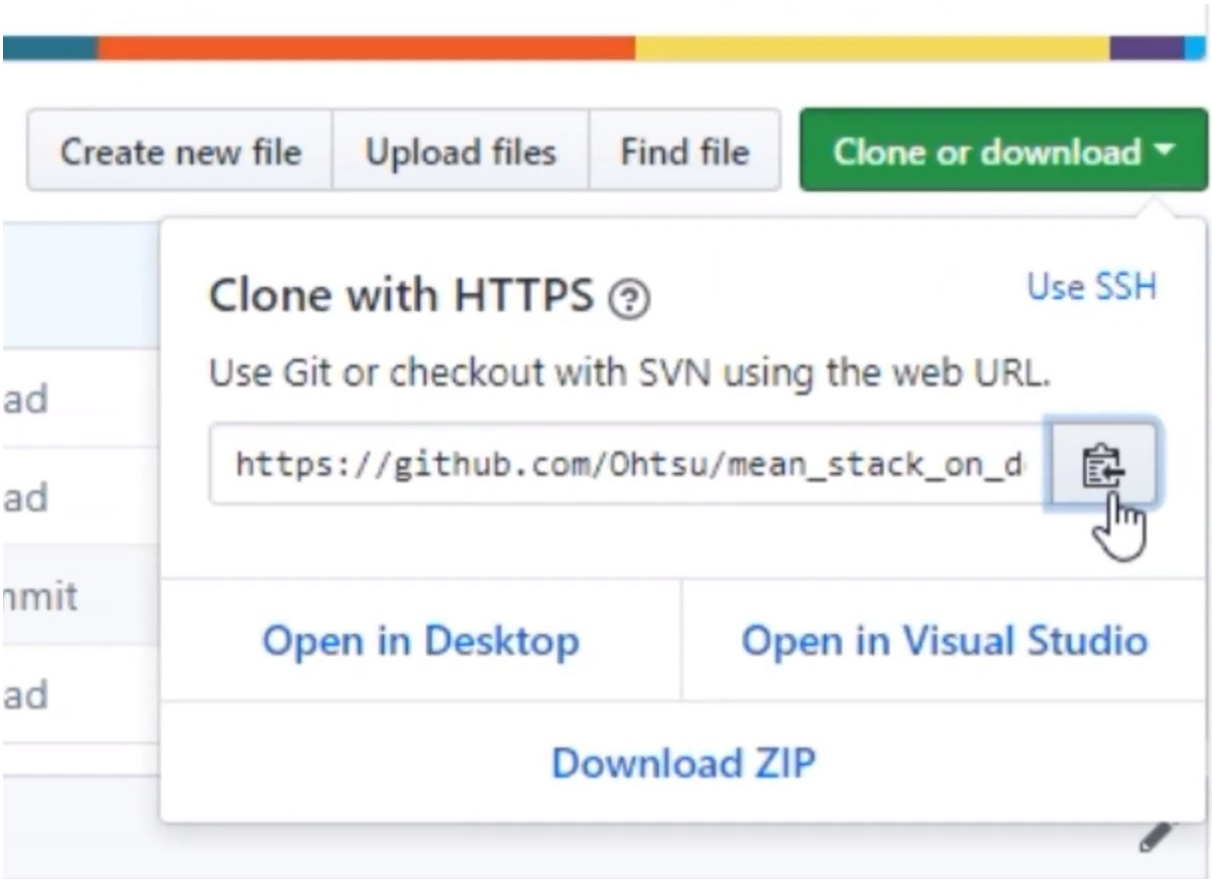
Next, download the source of client, middleware and server software to be used this time.

Open the GitHub site in the browser.

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



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



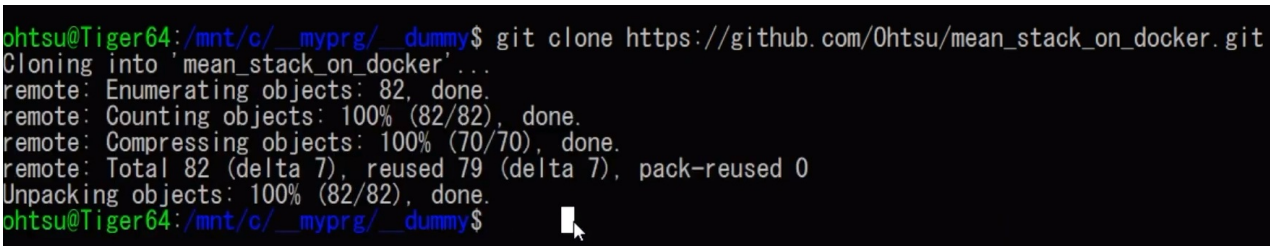
First, create a directory for downloading.

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

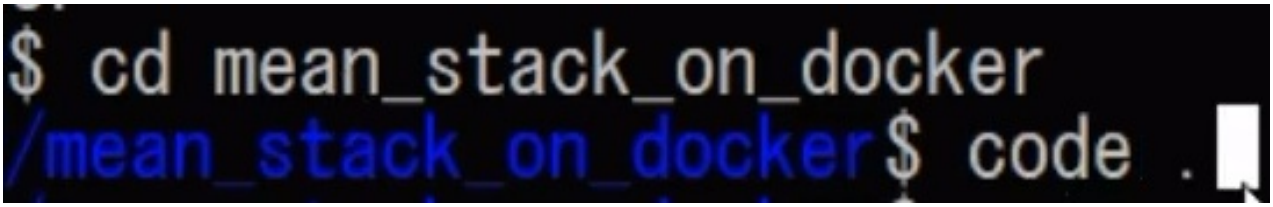
In this directory,

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

and download the source programs.



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



Check the docker-compose file

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

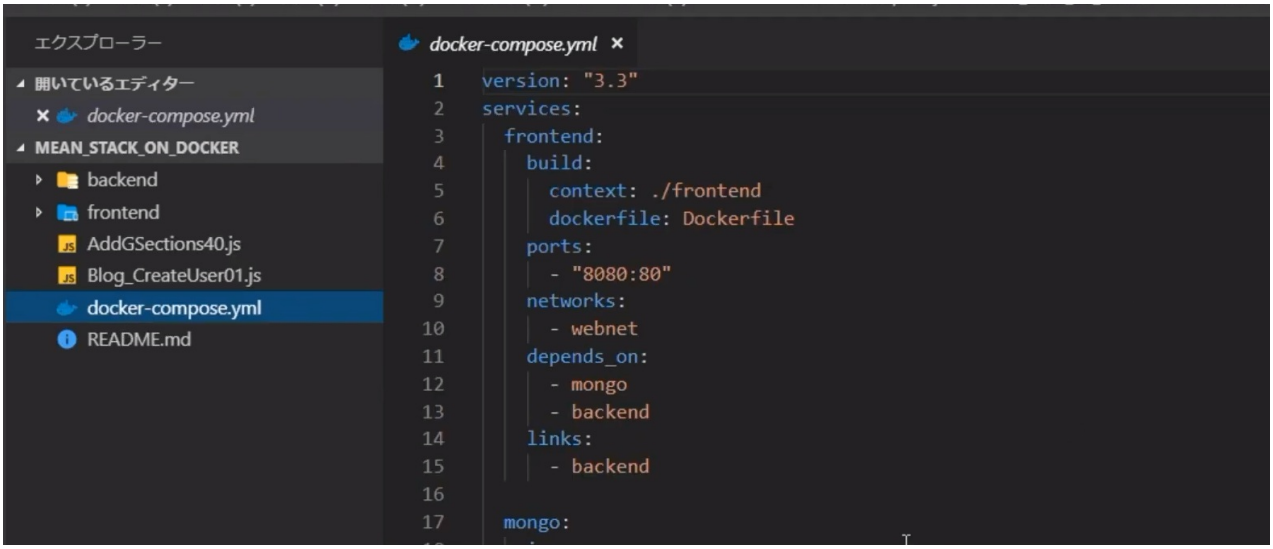
This file describes cooperation relationships such as client and server software.

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

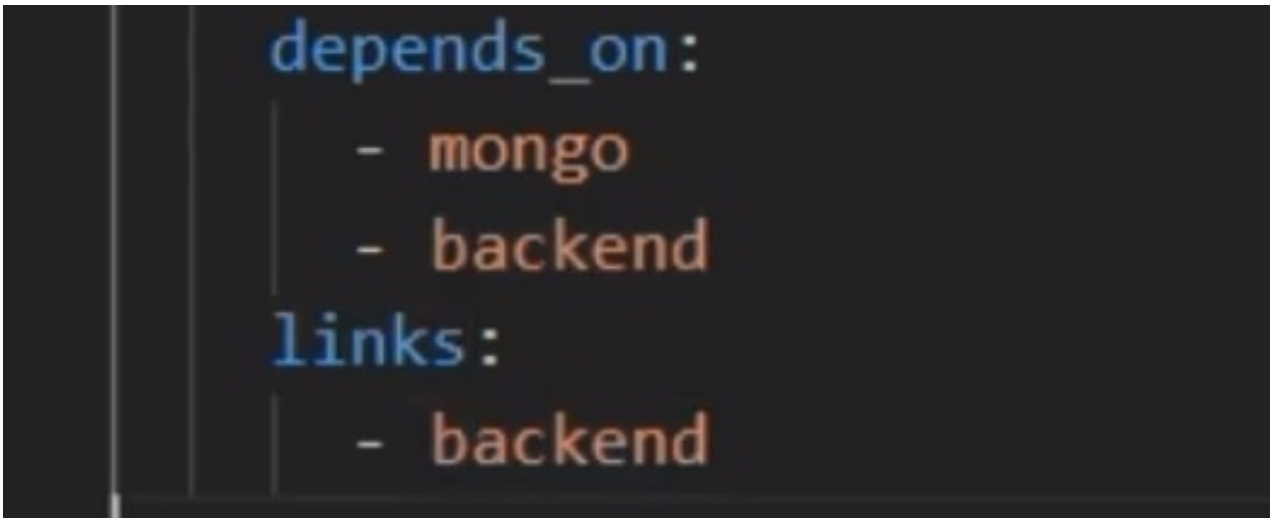
```
version: "3.3"
services:
  frontend:
    build:
      context: ./frontend
      dockerfile: Dockerfile
    ports:
      - "8080:80"
    networks:
      - webnet
    depends_on:
      - mongo
      - backend
    links:
      - backend

  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_UERNAME: 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.

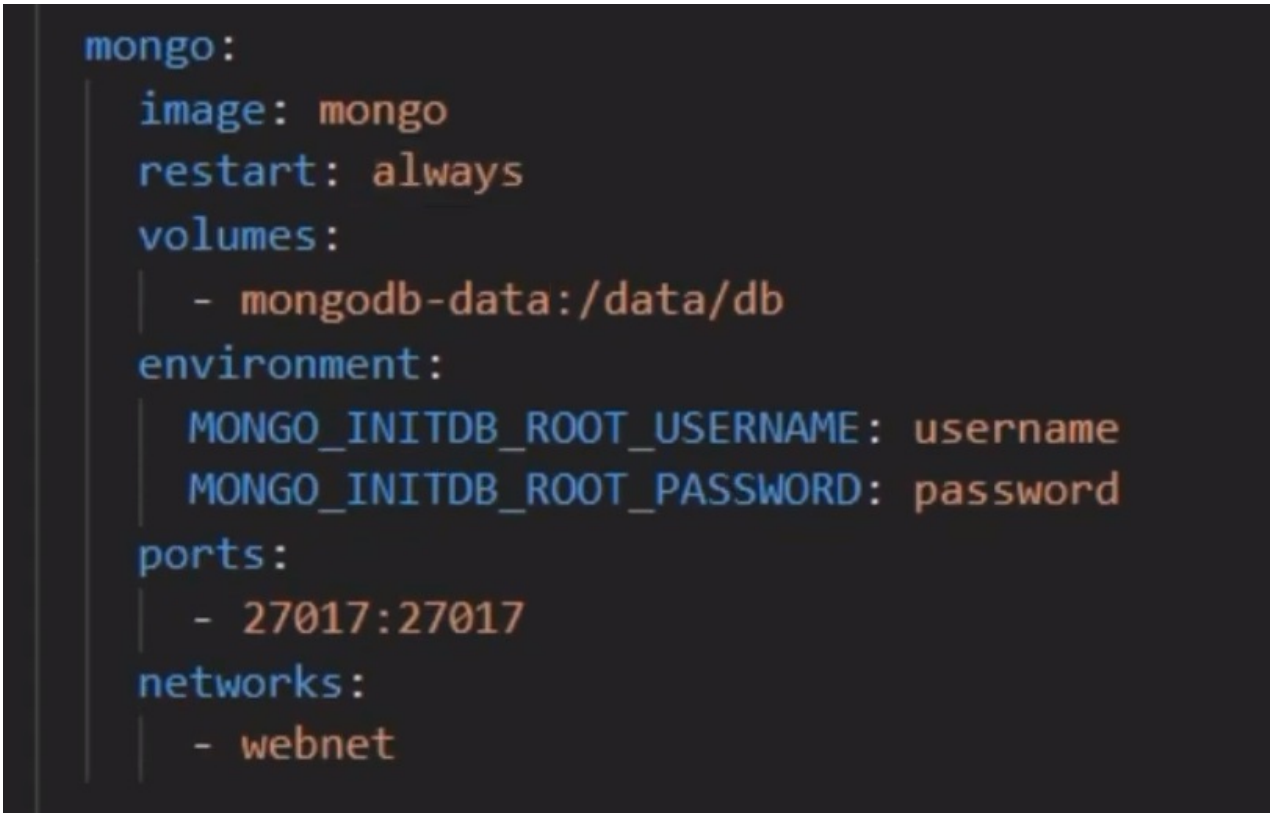


These are specified with `depends_on` and `links` commands ·



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.

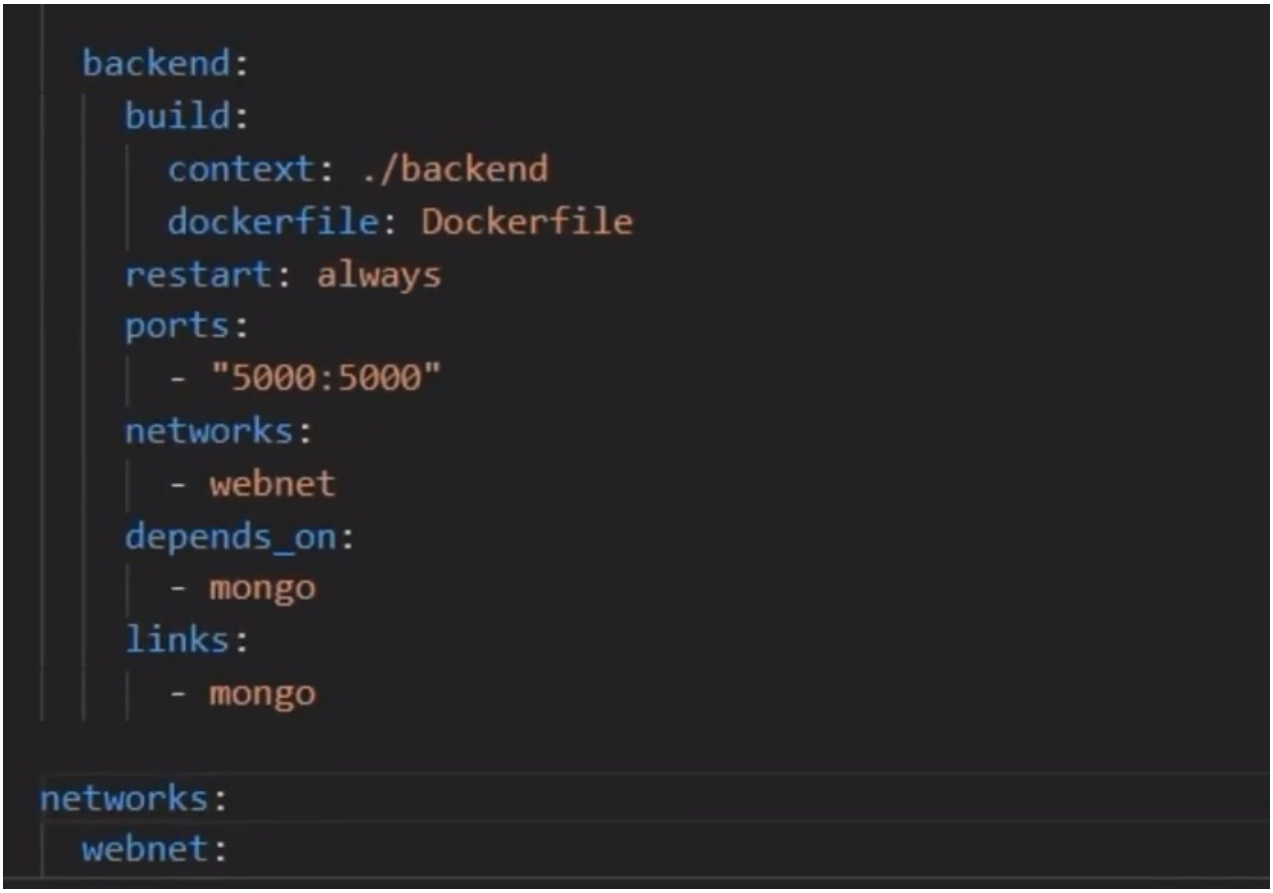


Check the Dockerfile files

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.



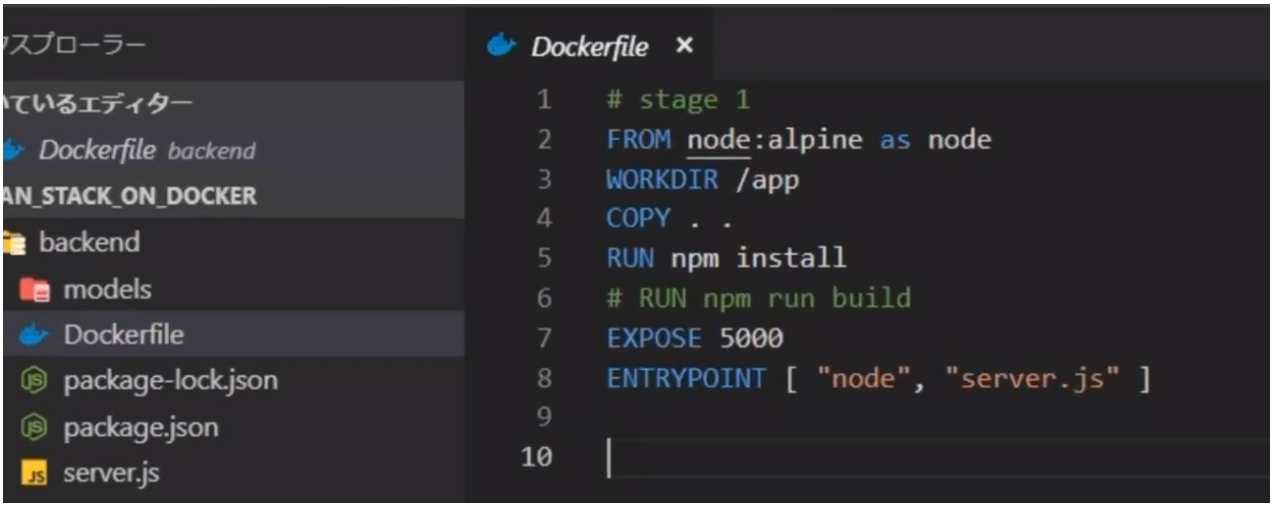
Next, for Angular files related to clients, we are going to open the Dockerfile first.

The specification of this file is in two stages.

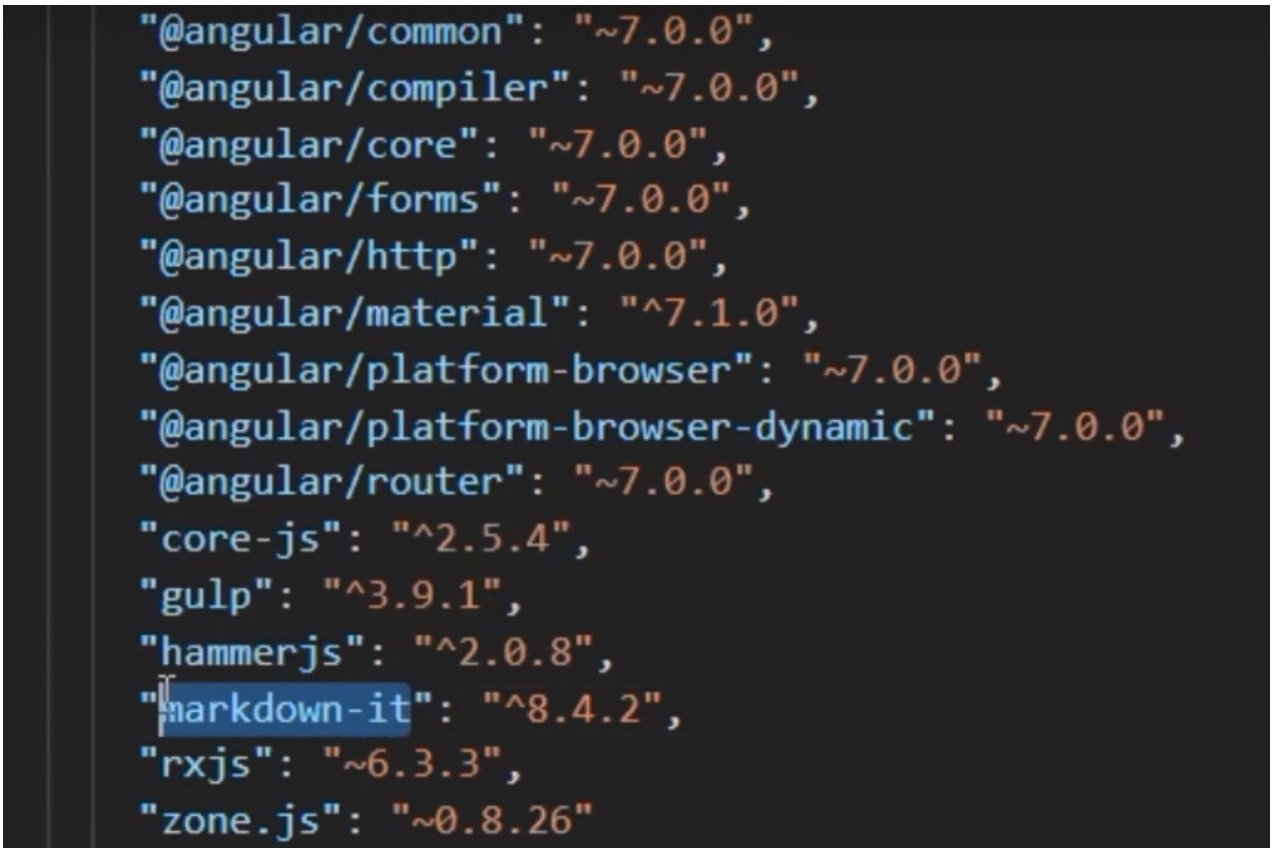
```
# stage 1
FROM node:10.12 as node
WORKDIR /app
COPY . .
RUN npm install
RUN npm run build --prod

# # 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.

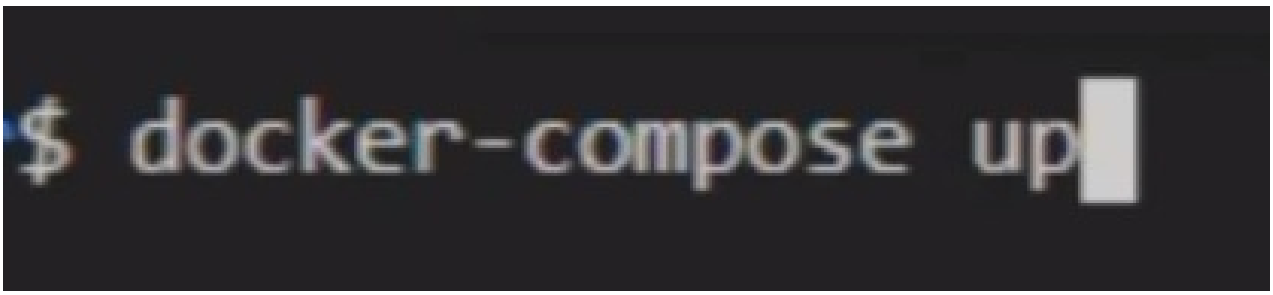
It copies the generated JavaScript compressed file to the home of nginx on lightweight Alpine Linux.

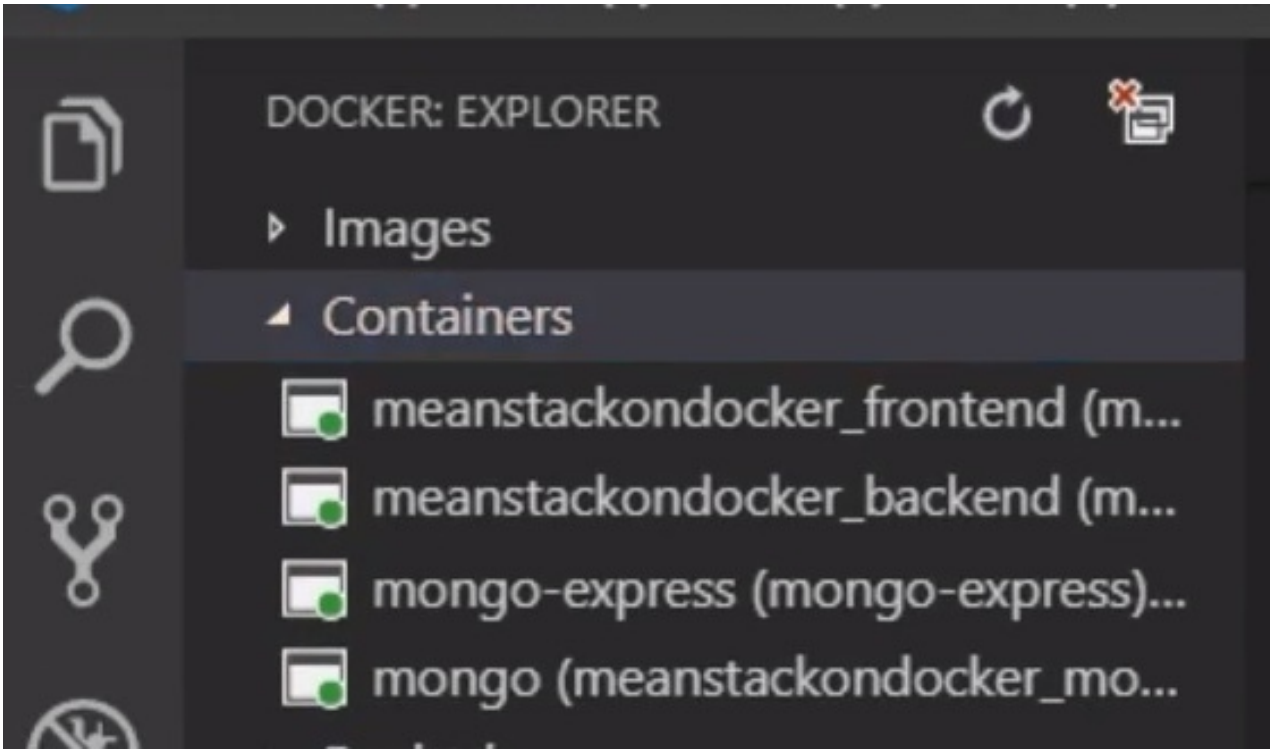


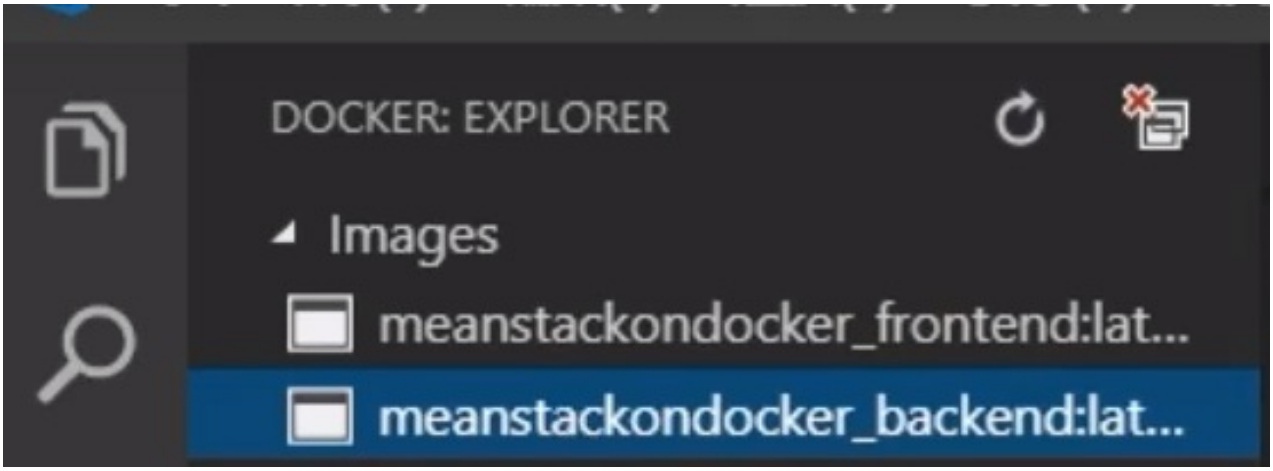
Please note that since this client software is in the form of Markdown's editing software, it also installs Markdown-to-HTML conversion tool `markdown-it`.



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.

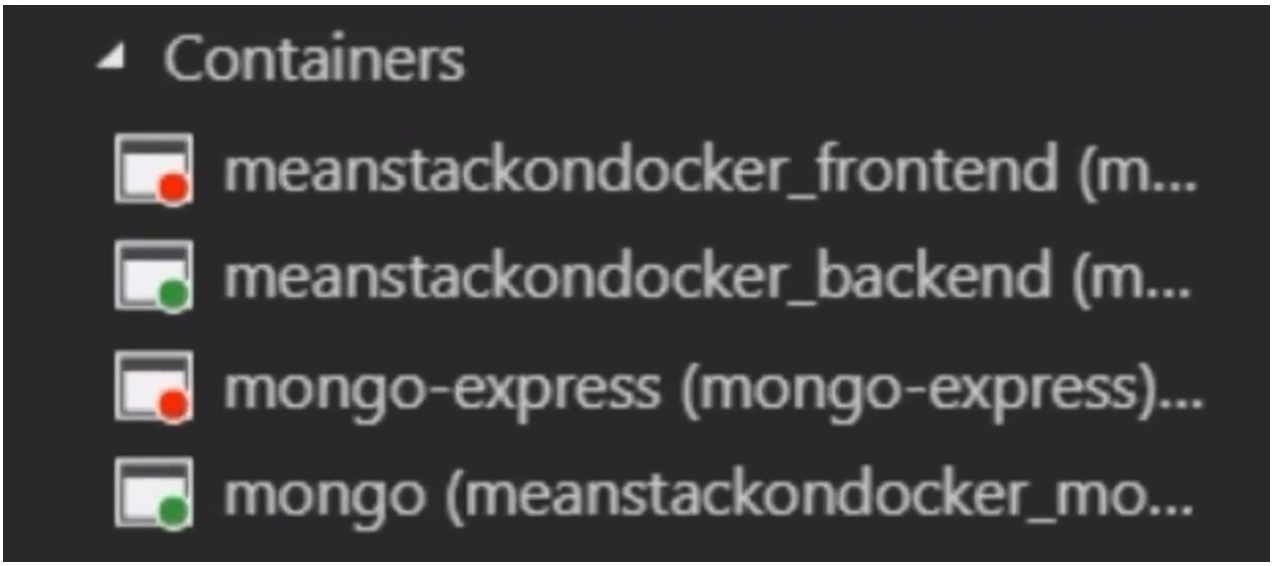
That is, if it takes time to start a database such as MongoDB even if you specify `startup order` of each software in `docker-compose` by `depends_on` etc, it will be a linkage error while creating the

image It is from.

Enter MongoDB sample data

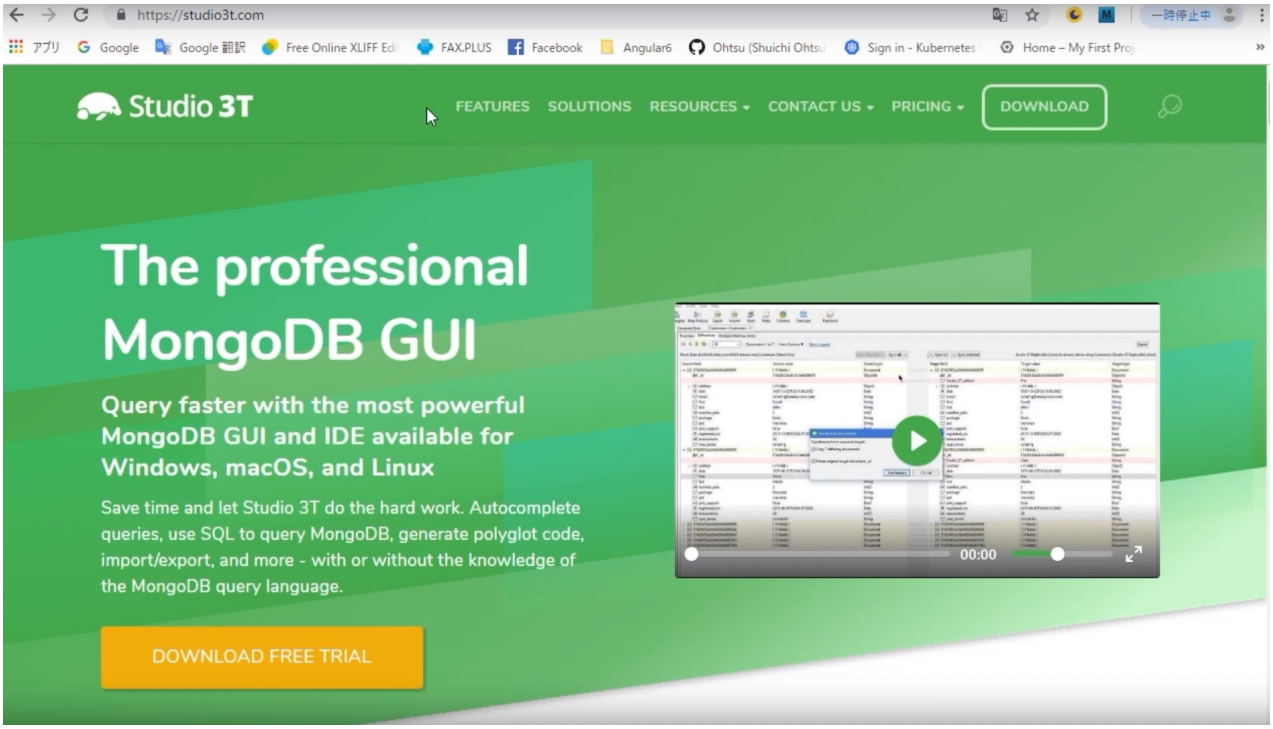
Next, we are going to input sample data in MongoDB.

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



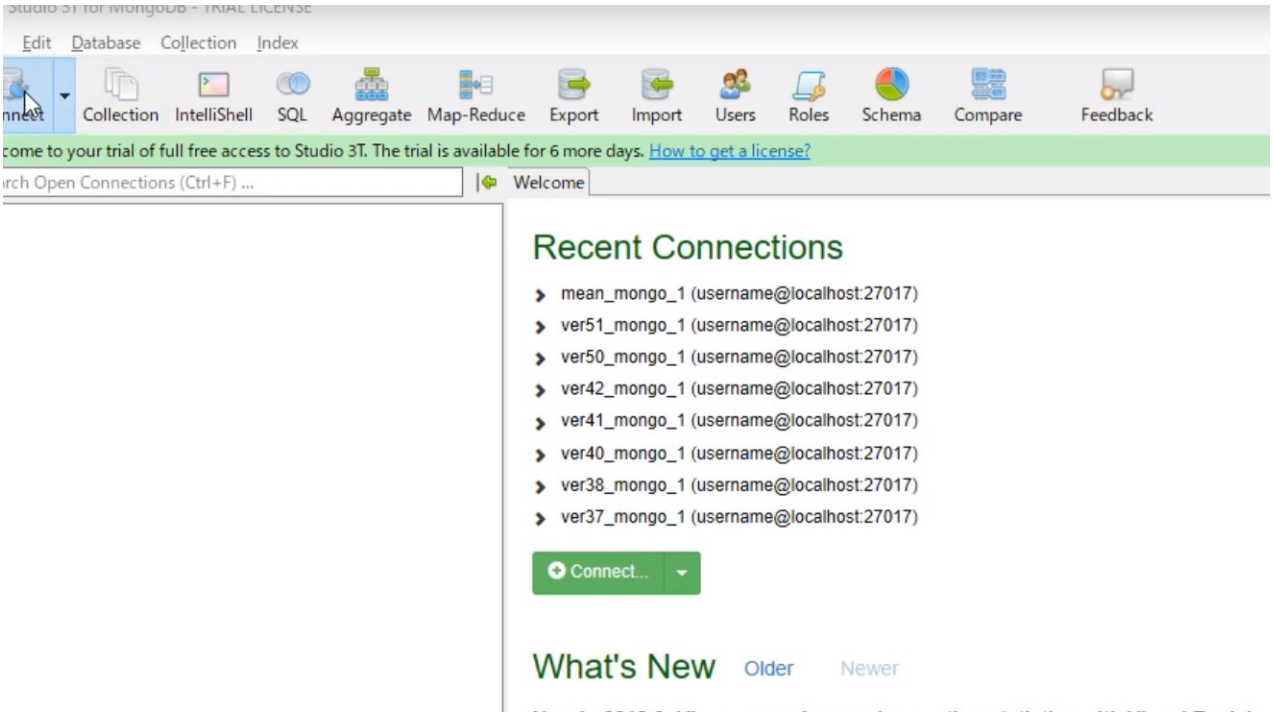
Here, we use *Studio 3T*https://studio3t.com tool for inputting MongoDB data.

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



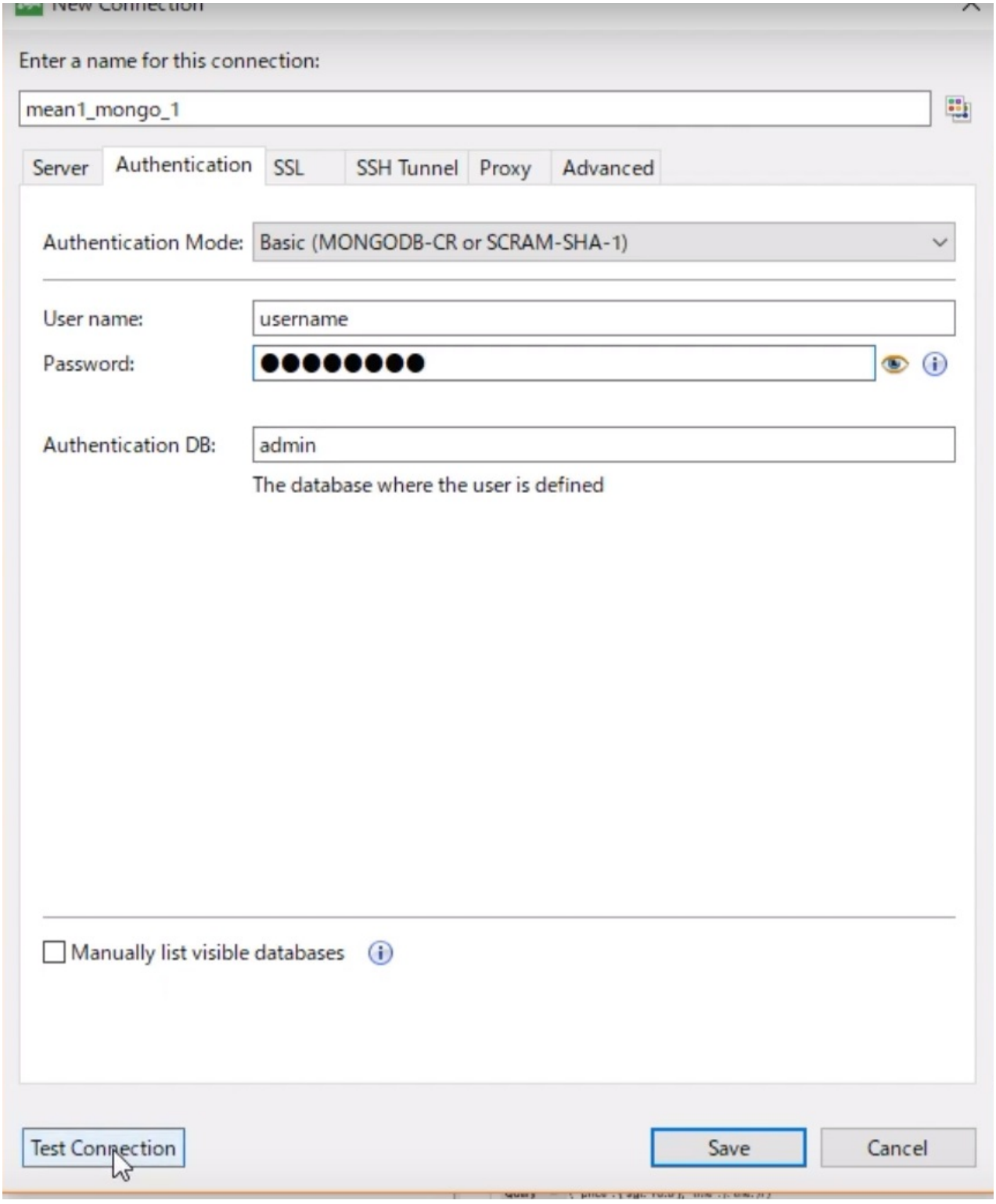
First of all, you need to connect to MongoDB.

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

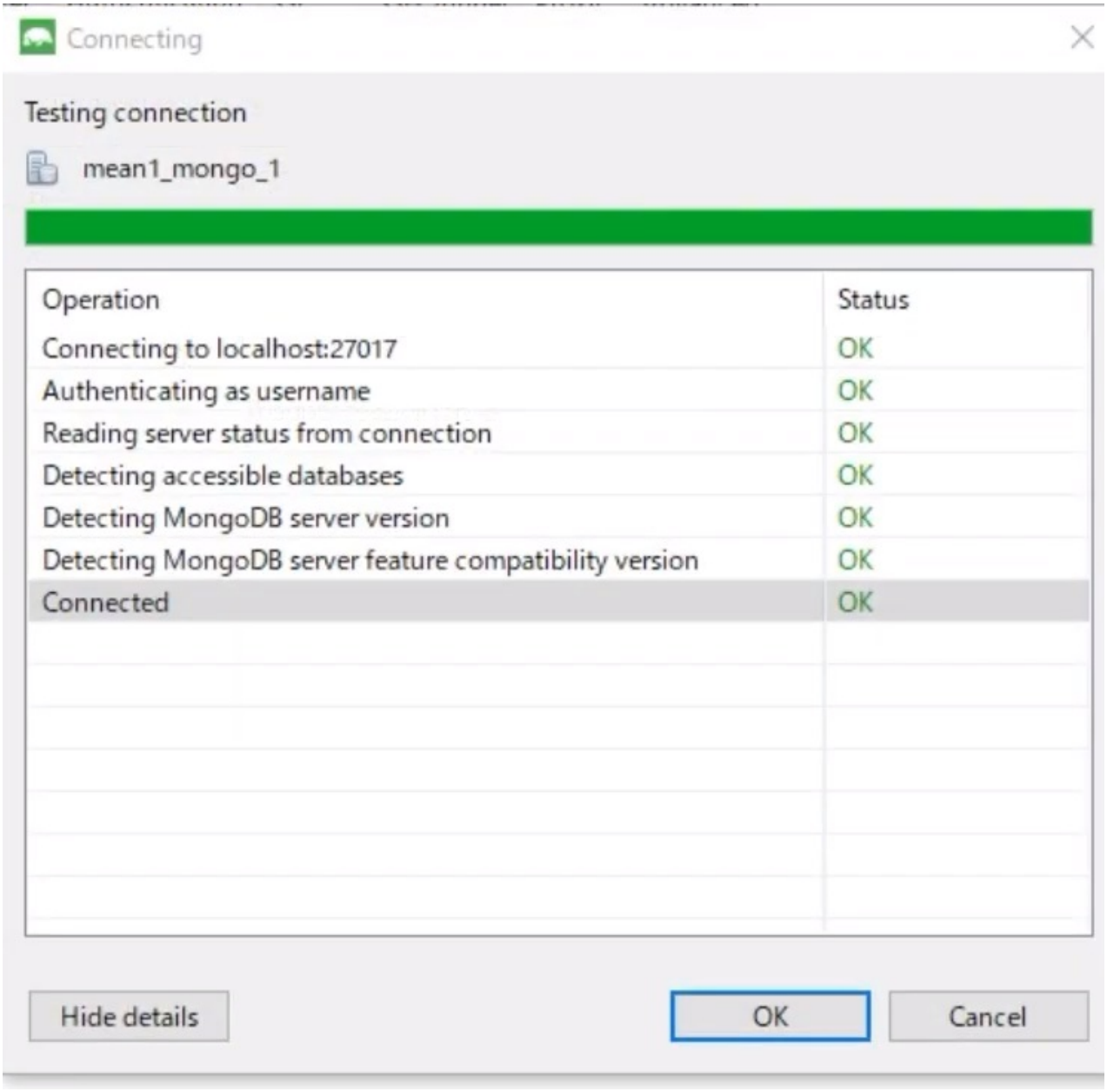


Then select the new connection, enter the server name appropriately, then select the `Authentications` tab.

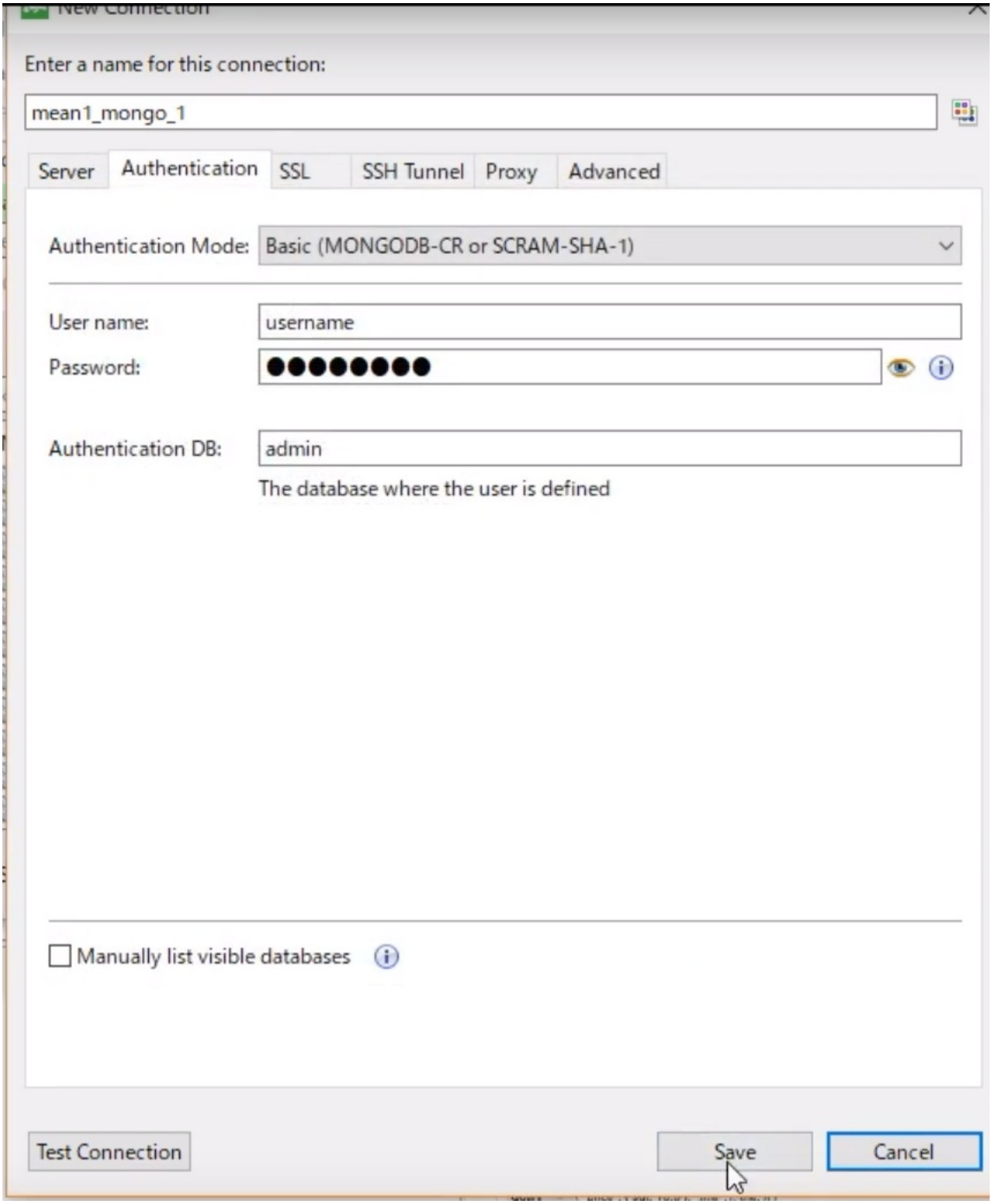
In addition, select `Basic` in `Authentication Mode` and set the user name (username) and password (password) set for the administrator in the `docker-compose` file in the same way.



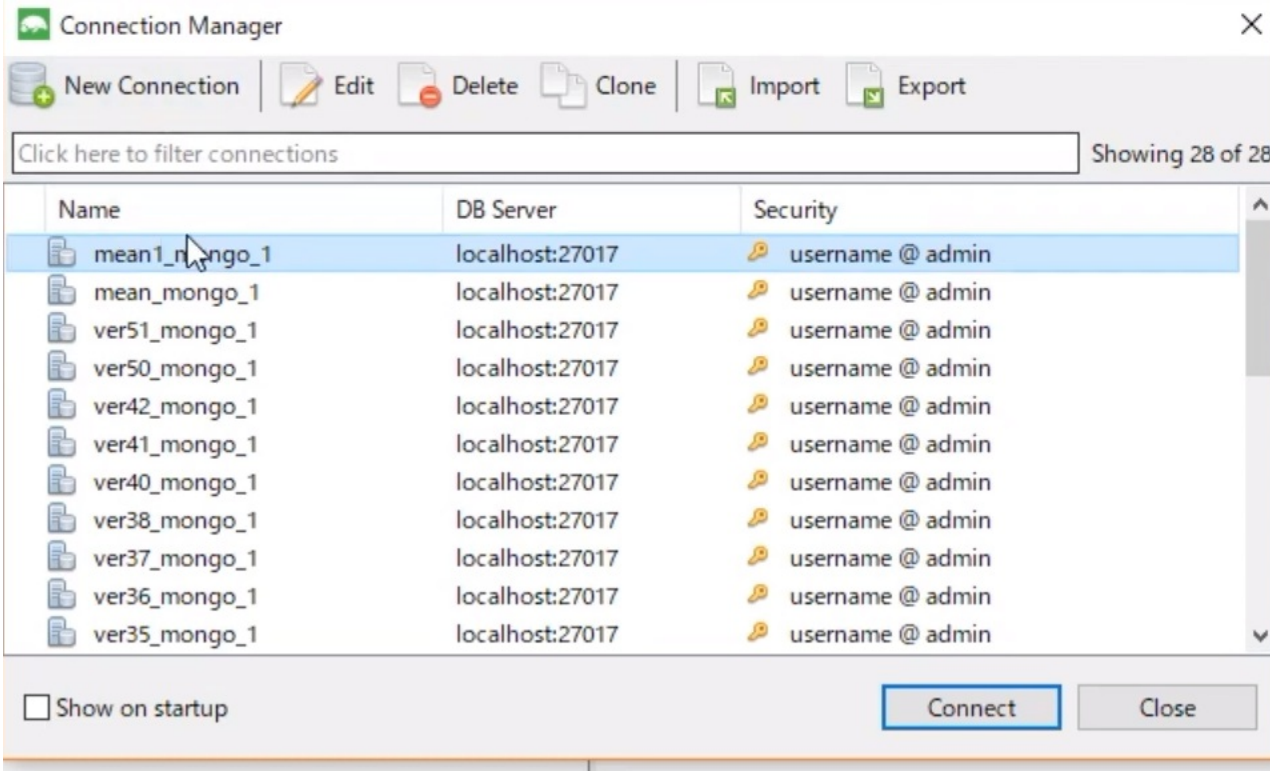
Then click the `Test Connection` button on the bottom left, and if it is `OK`, the connection is complete.



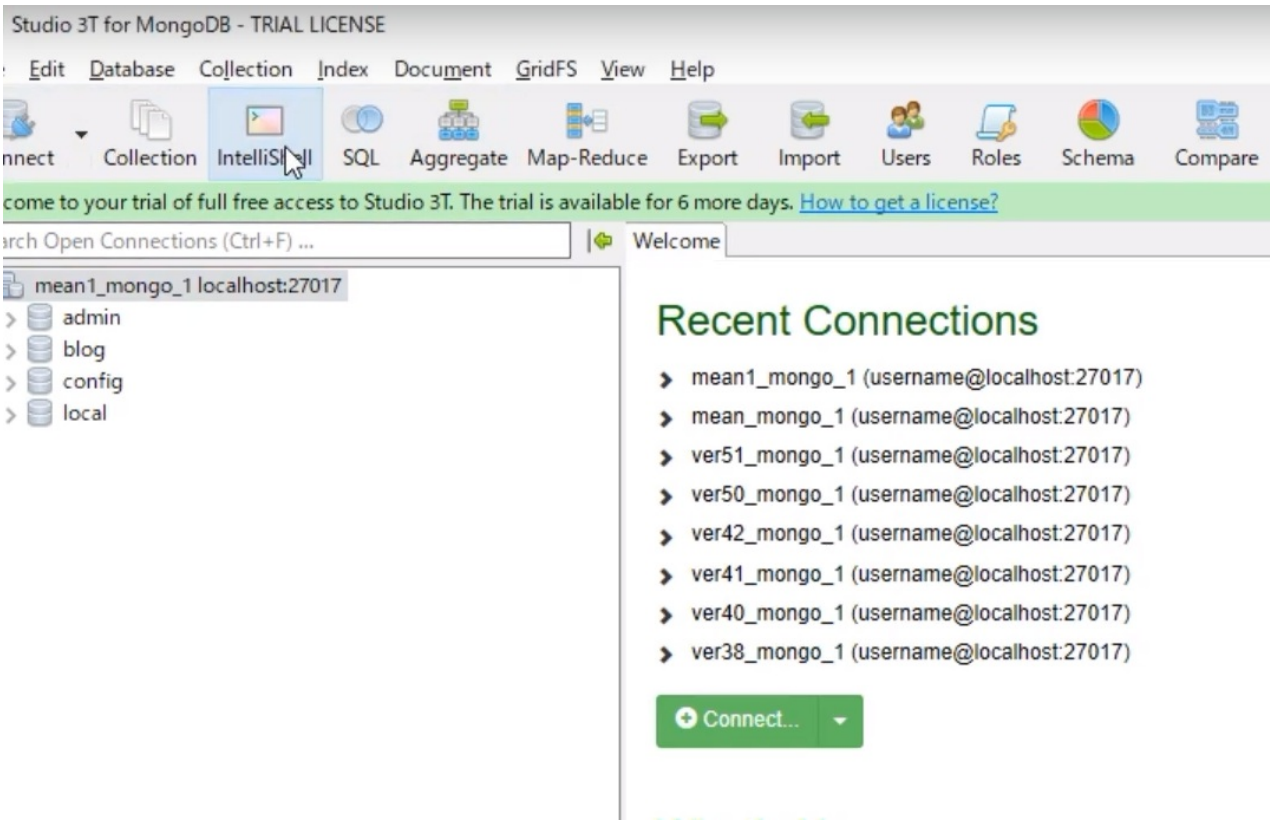
Click the `Save` button and save this setting.

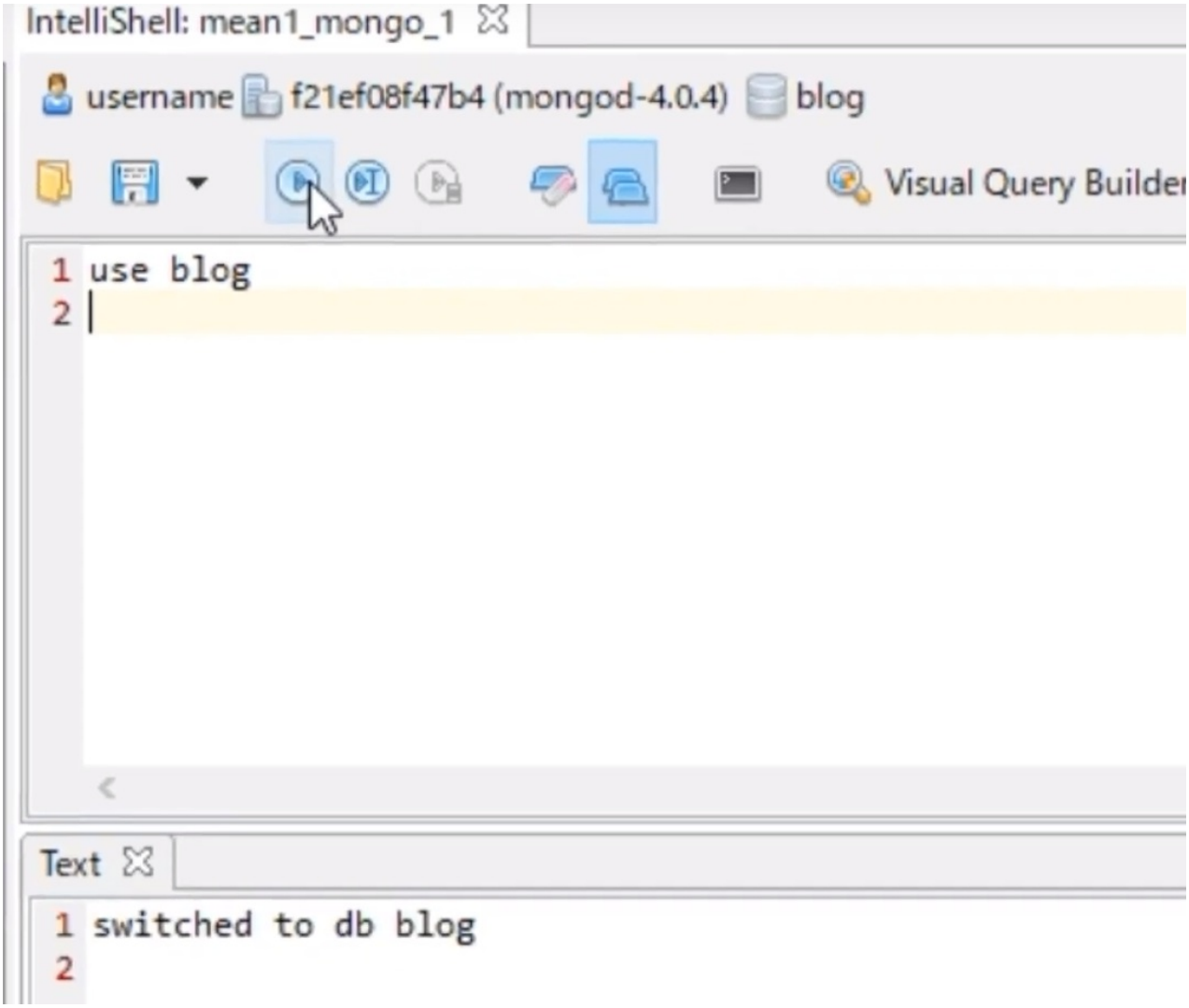


In addition, click the `Connect` button and connect.

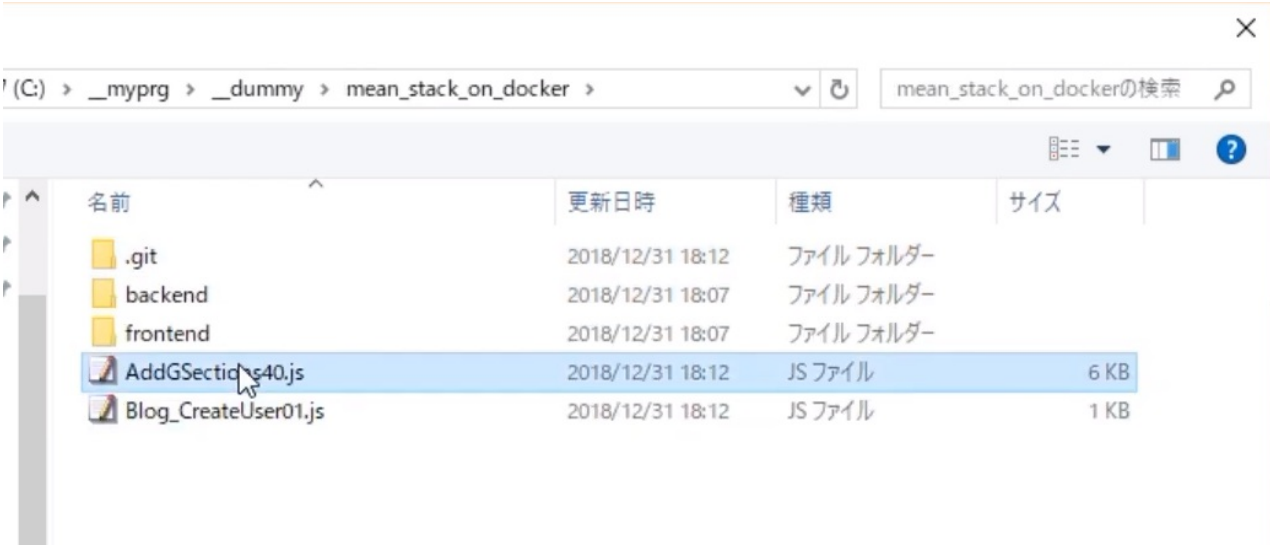


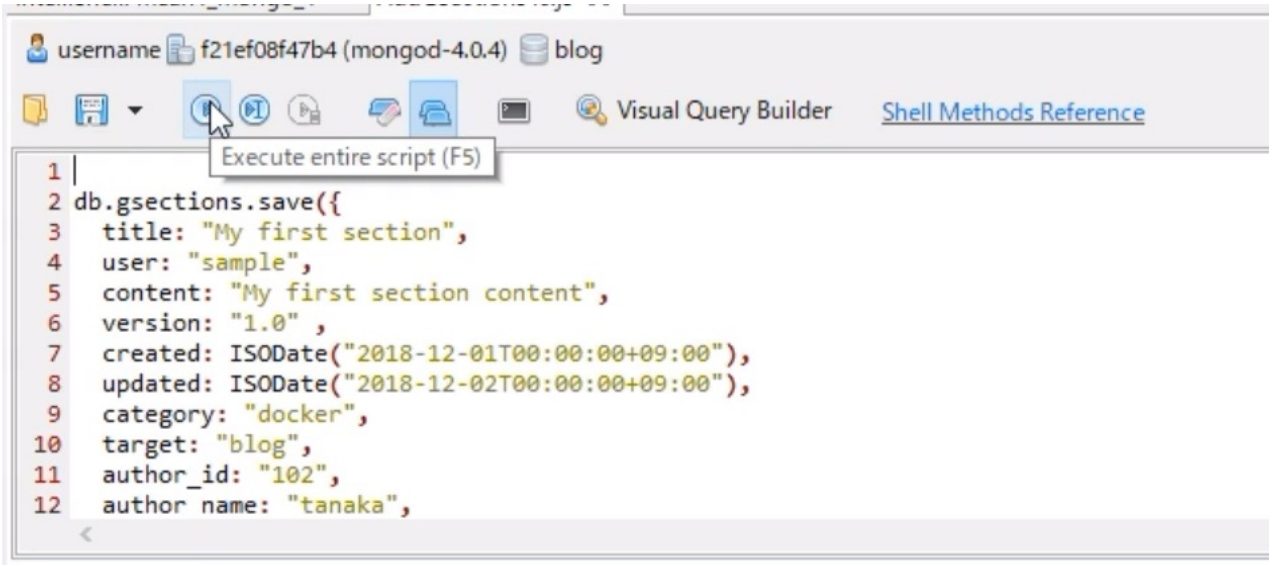
Next, click the `IntelliShell` button, open the MongoDB shell window and enter `use blog` to create a blog database.





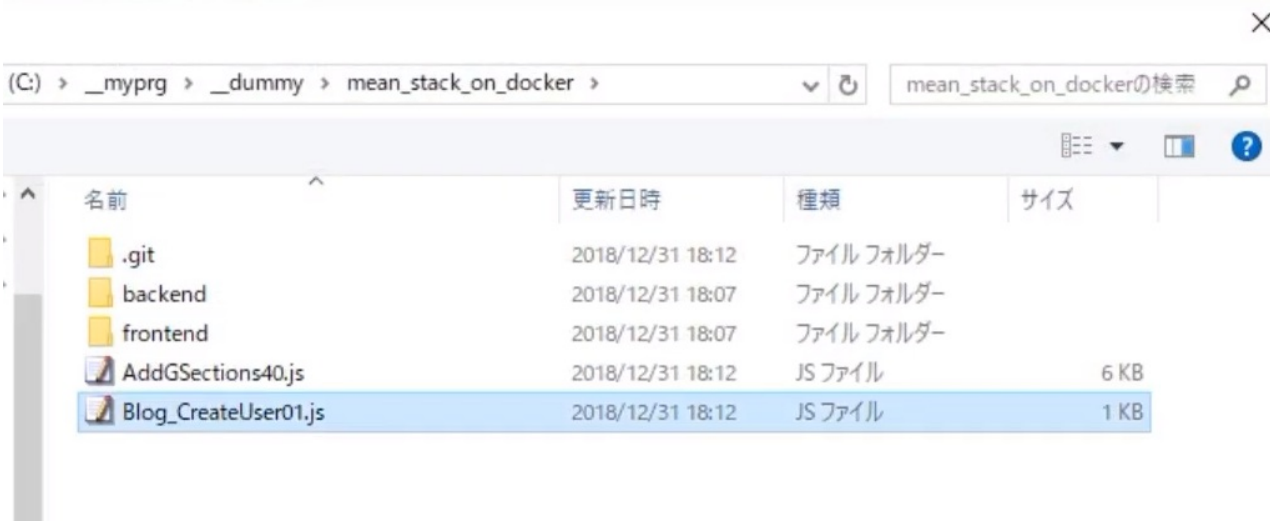
Click the `IntelliShell` button again to register the data. Since this is registered in GitHub, paste the file and register it.

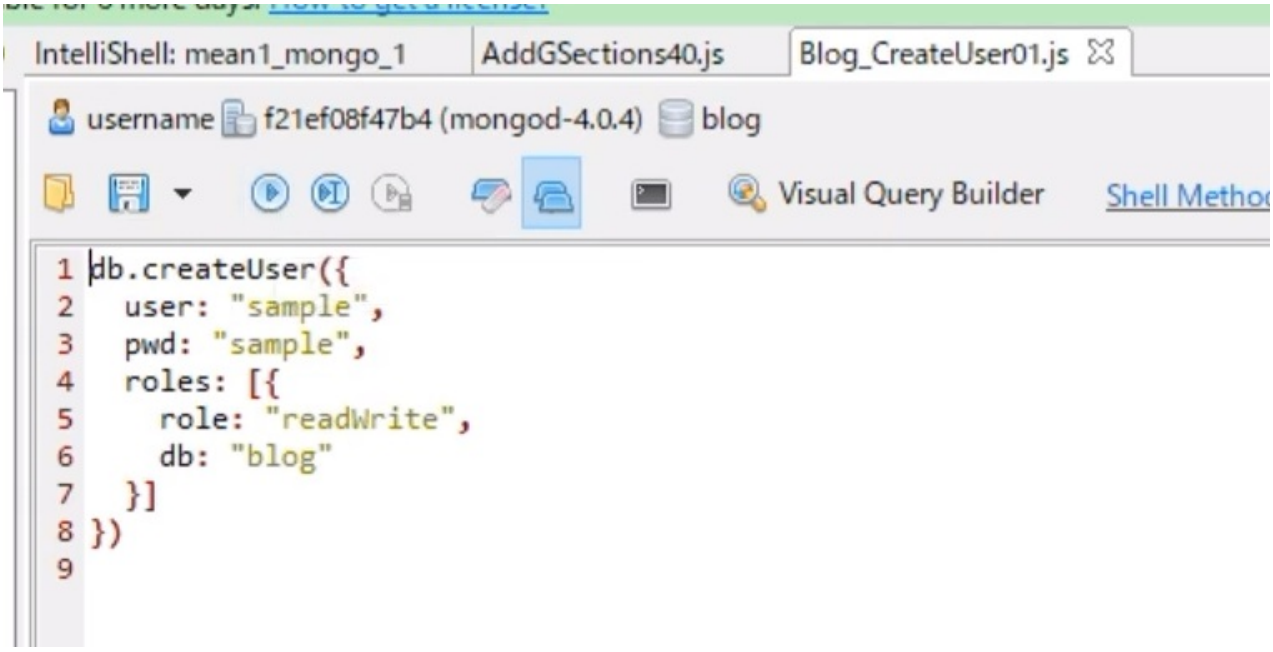


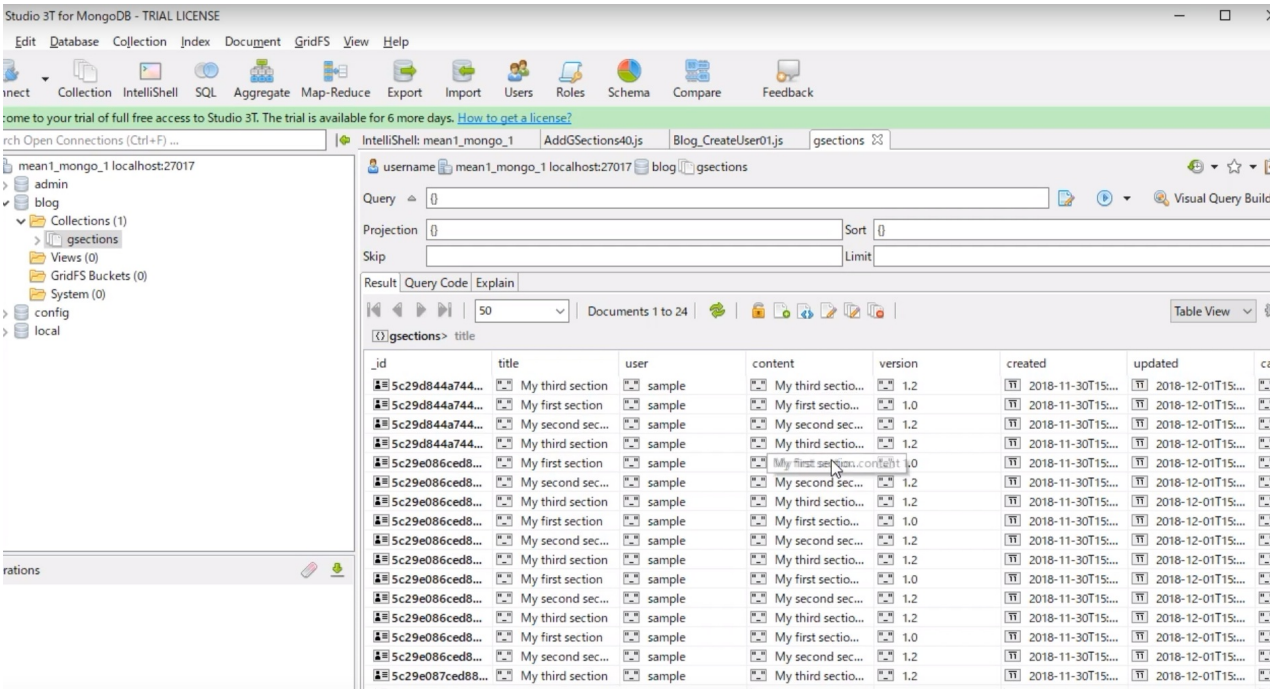


Finally, register the access authority of the reference user (sample).

This is also used as it is registered in GitHub.



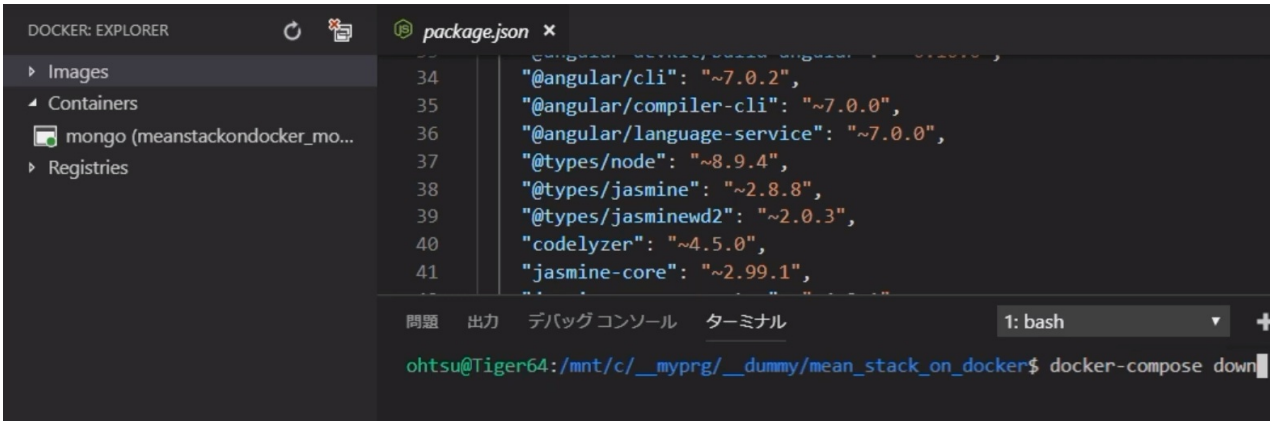




Start server and client software

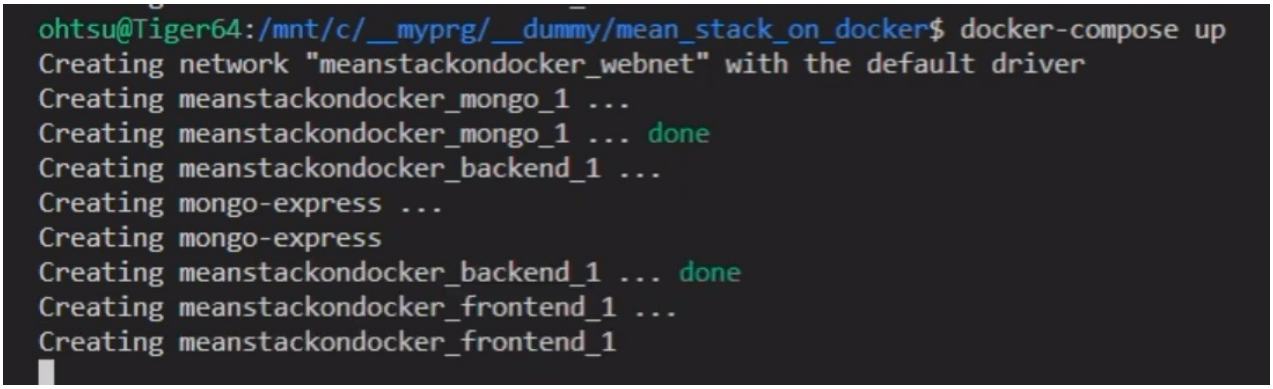
First, stop running MongoDB.

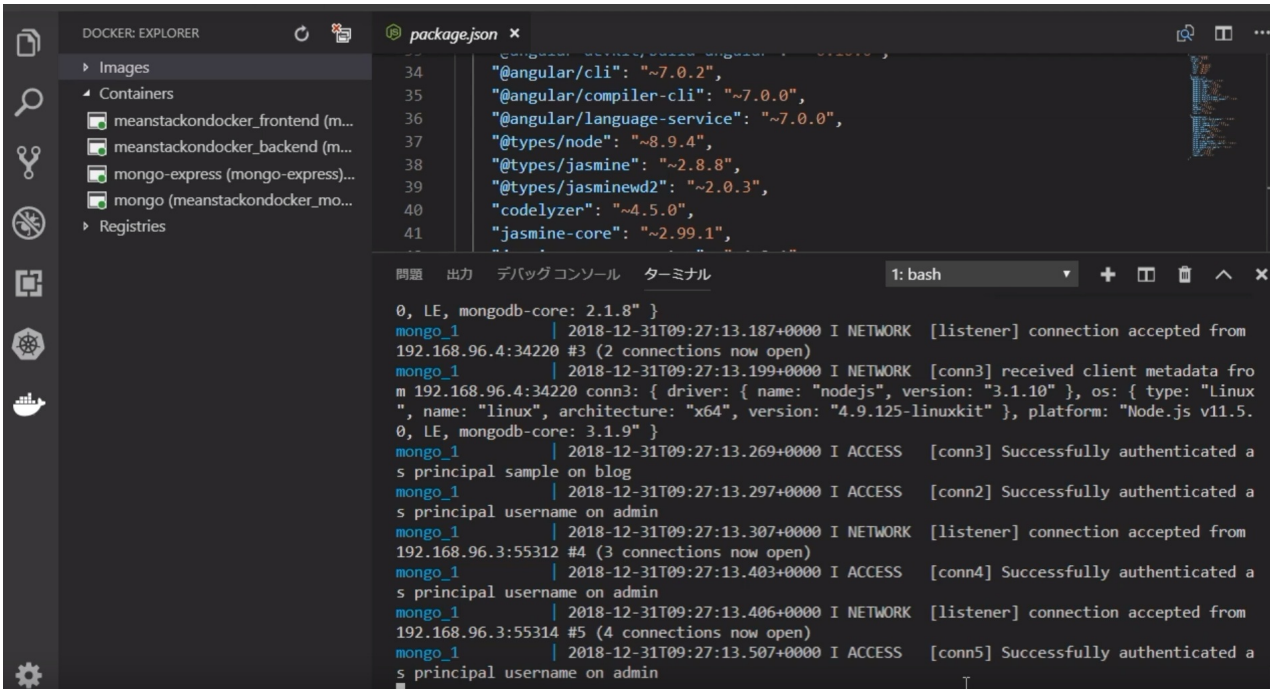
Type `docker-compose down`.



Next, start all the software.

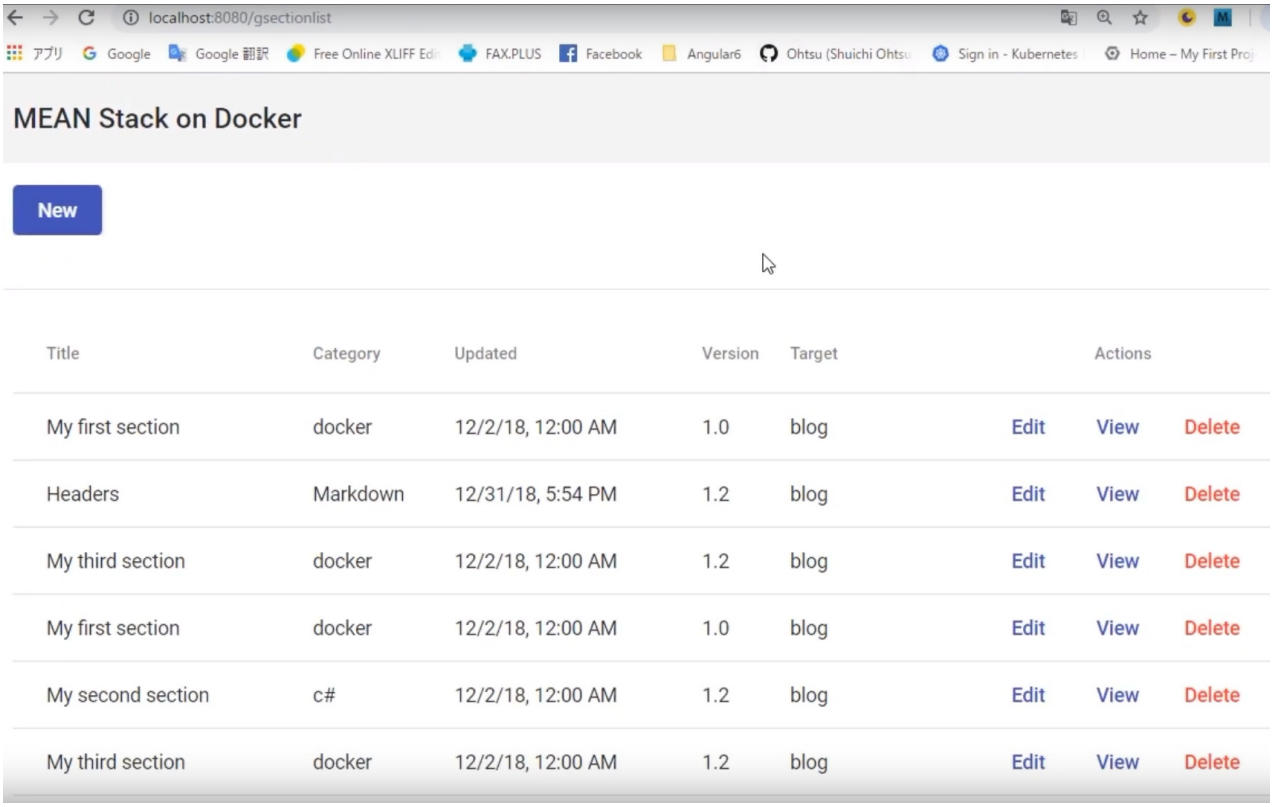
Type `docker-compose up`.





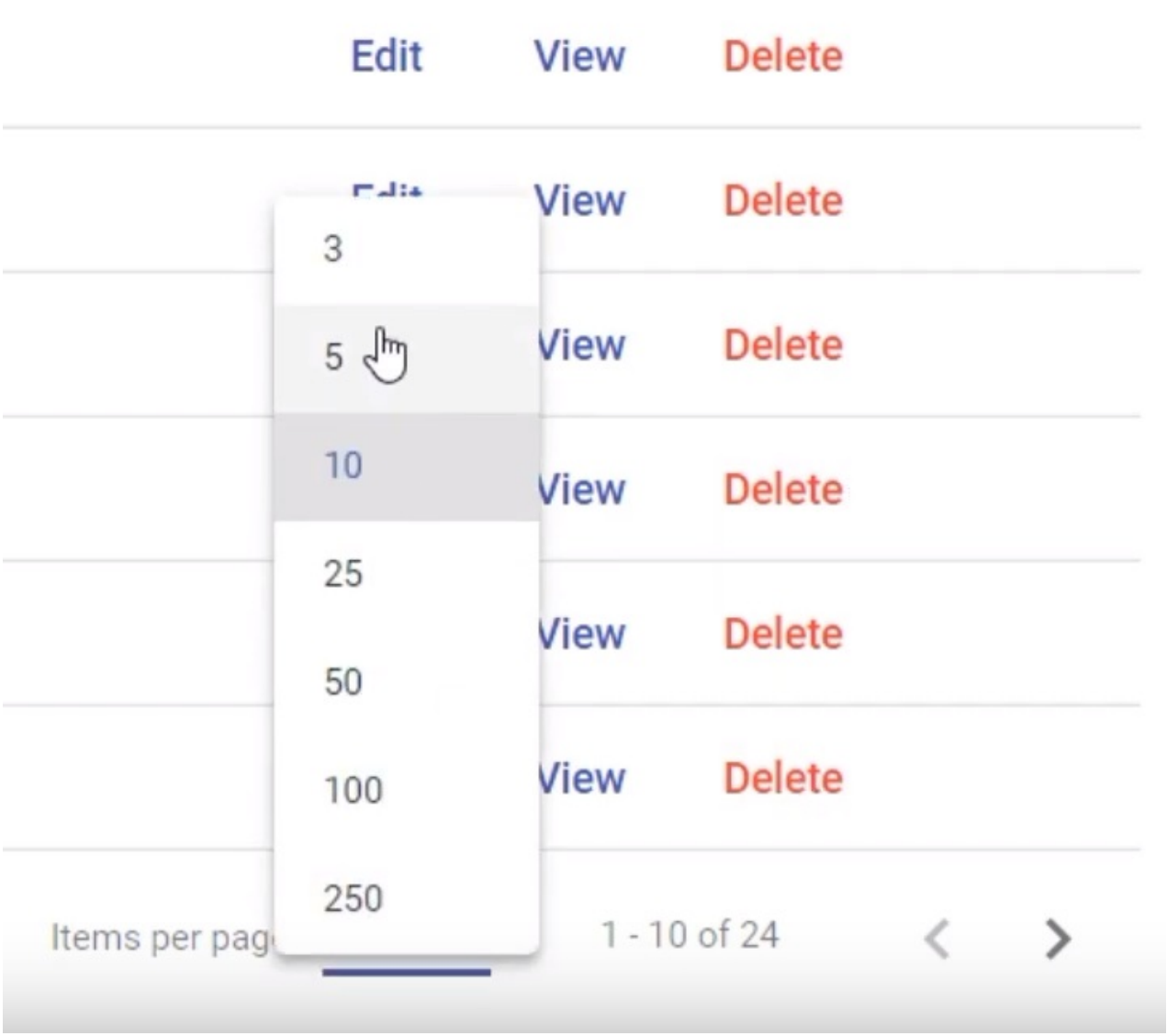
Confirmation of client software

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.



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.



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

New

Title

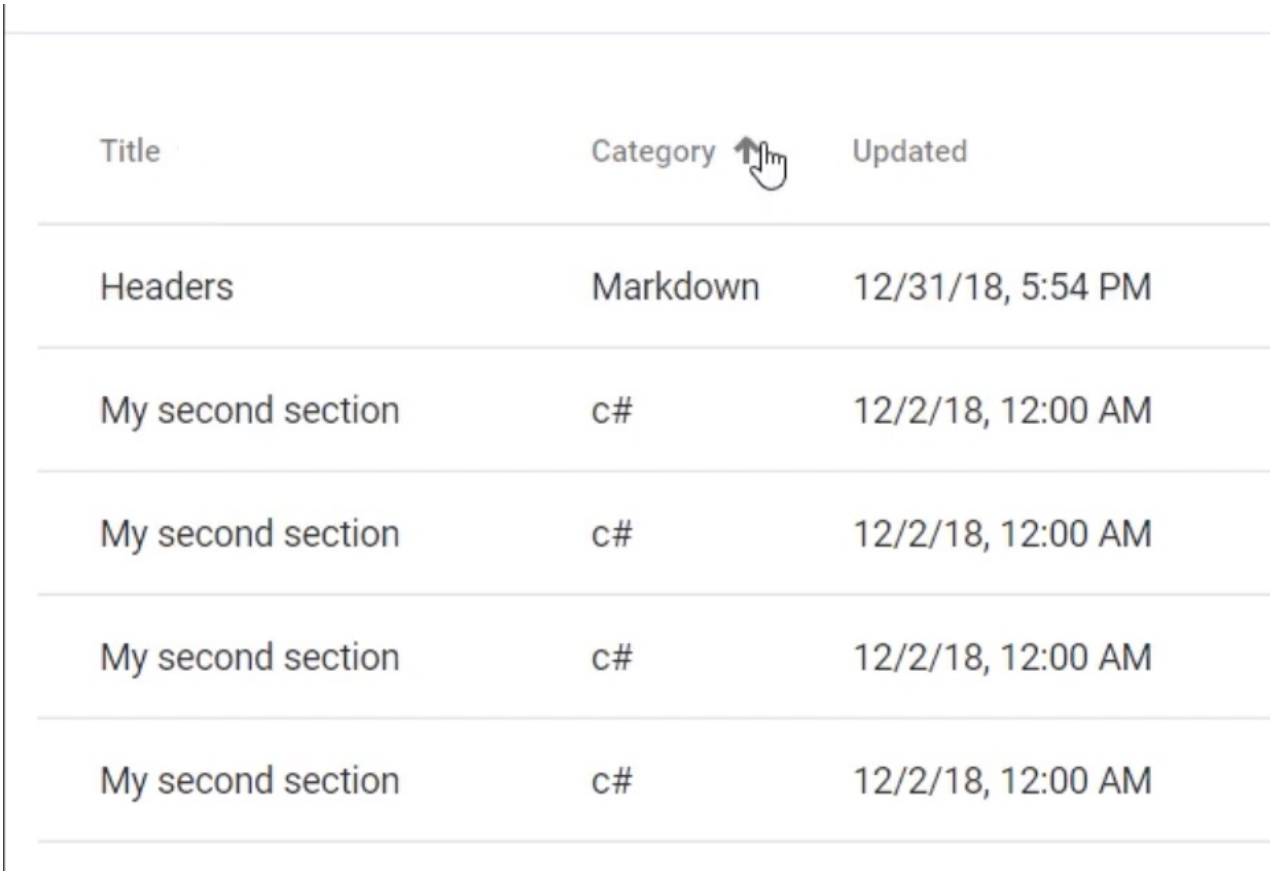
My first section

Headers

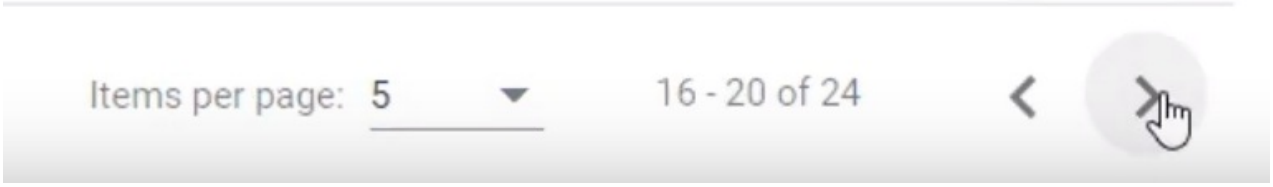
My third section

My first section

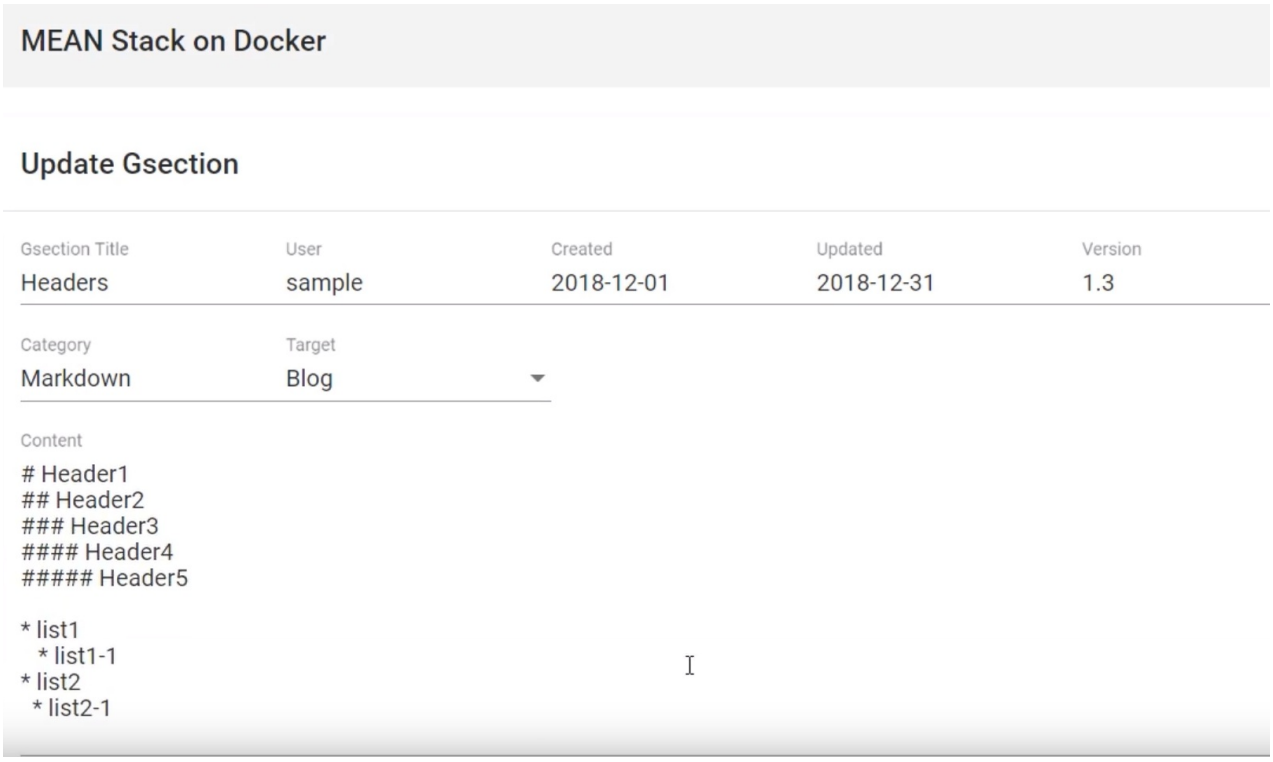
My second section



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

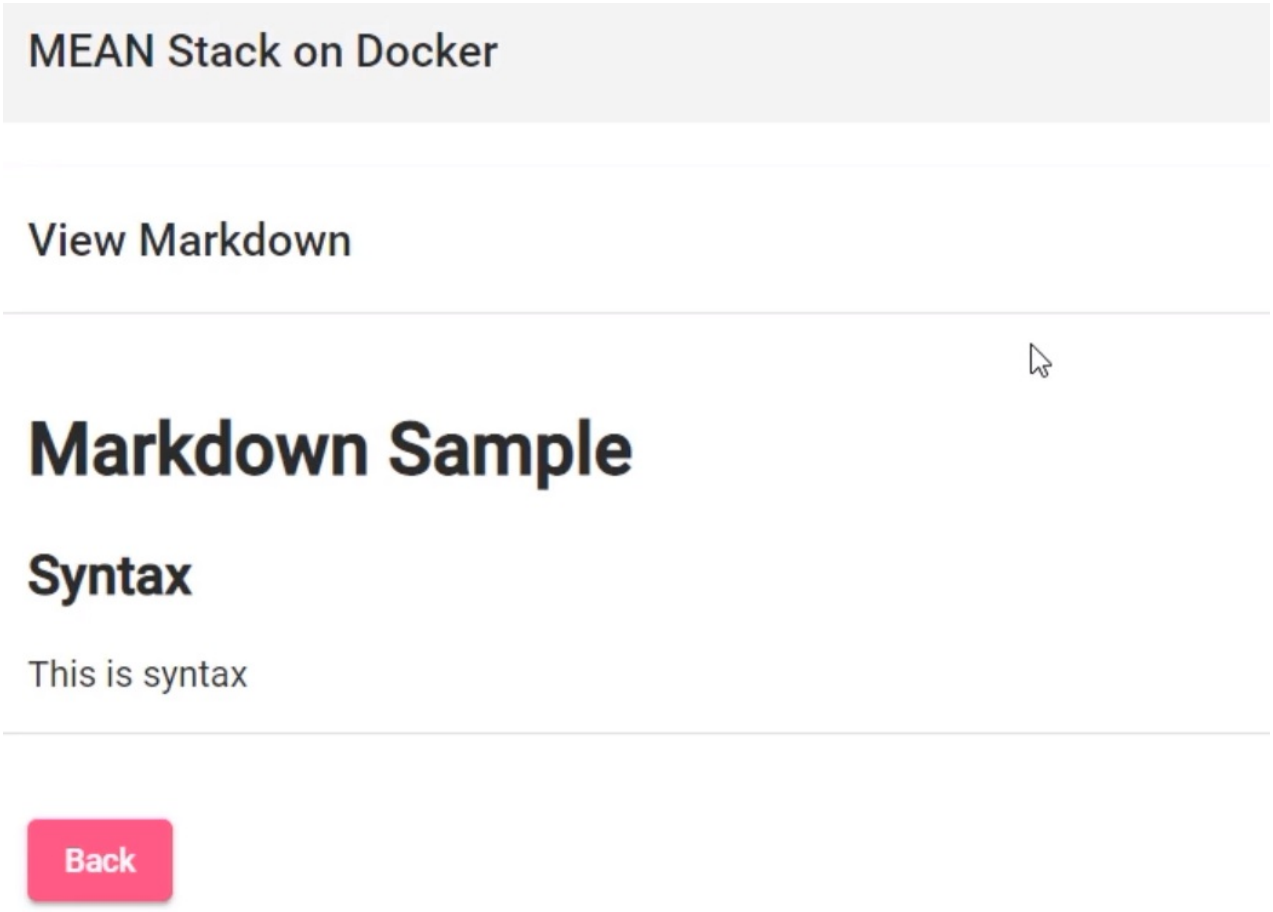


Click the `Edit` button in each column to display the page for correction.



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.



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
- "ngular 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/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-l/