Download the networking code from the following Github URl:

<https://github.com/harikrishna83/Docker-firstdemo-handson/tree/master/networks-starting-setup>

Our requirement is from our container we need send a network request to outside of the world some API and need get the output.

https://swapi.dev/api/films

https://swapi.dev/api/people

Those two are API’s which are outside, which will provide some movies and actors information, now we run a node JS application and will establish a http connect to those outside world API’s and get the response. Also this container need to communicate with some service on our local machine.

Some-api.com/

Requests from container to WWW

<Your App>

GET /some-api.com/

SQL Database

Requests from container to other container

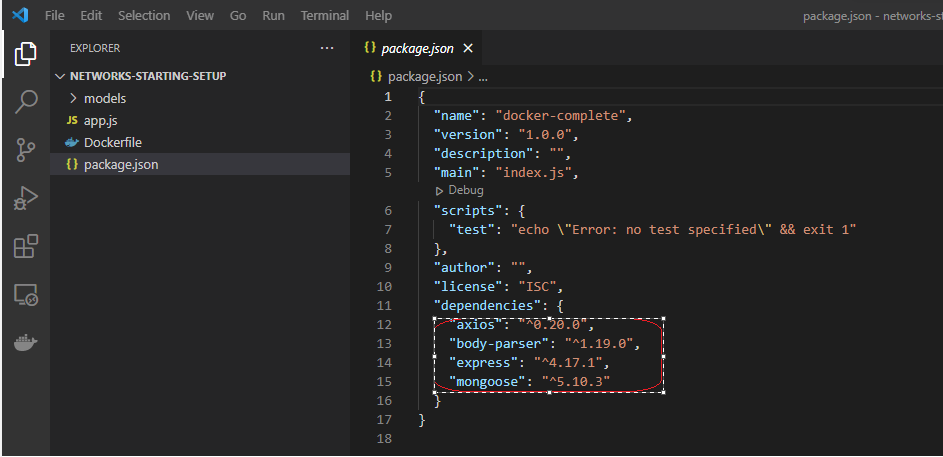
Requests from container to host machine

Your Host Machine

In brief there are three scenarios

1. From our container to outside API on www web.
2. From our container to access some service on our localhost machine.
3. From our container to access SQL database which is running on another container.

Let’s try to understand the application, it had couple of dependencies which need to be installed as shown below.



Now It has some custom code which return some and post some data via WebAPI. This is an web application which doesn’t return anything like a webpage instead it just returns data and also it updates data when request was send through this WebAPI.

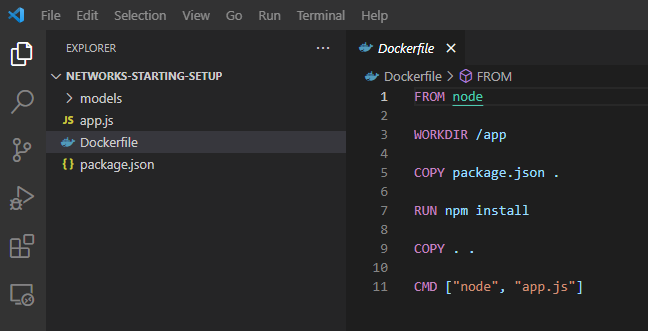
This webapi had four so called end points

1. It accepts get requests to <Web API URl>/favorites.
2. It accepts post requests to <Web API URL>/favorites, which will post data to mango DB by supplying some variables in the form of Json( name, type, url).
3. It send get requests to some outside world starwar Dummy API (‘https://swapi.dev.api/films’)/movies.
4. It send get requests to some outside world starwar Dummy API (‘https://swapi.dev.api/films’)/people.

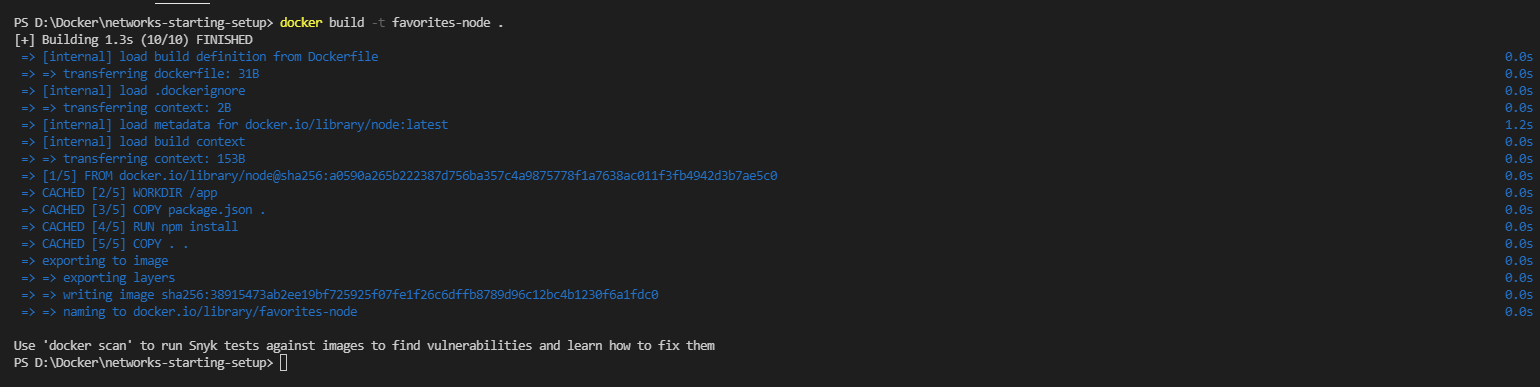
For 3 & 4 points we are sending a requests to get the data from outside world start was API.

Note: We will use postman software to send the requests to and from via webapi.

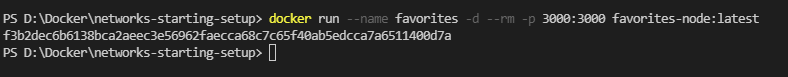
Now let’s dockerize the container by writing the instructions on Dockerfile.

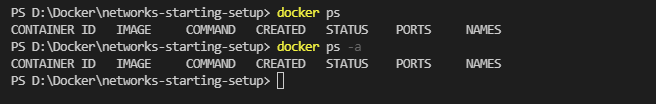


Now let’s build the image by using the instruction now.

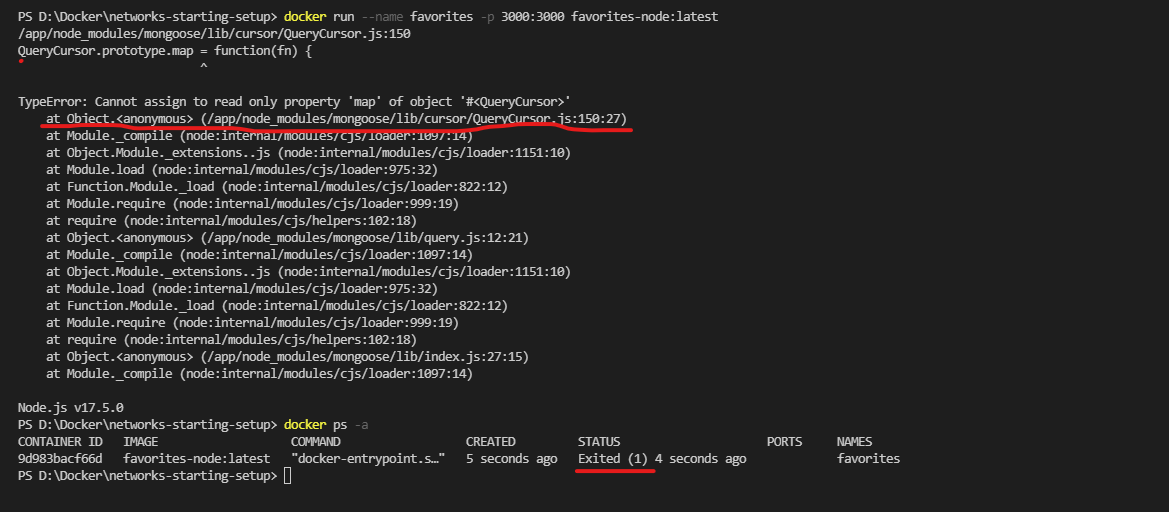


Let’s run the container with docker run





I see there is no container running, let’s what went wrong. To see the error, I will start the container in attached mode.



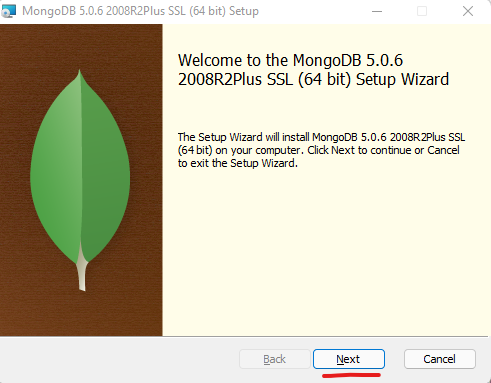
On my app.js I am trying to connect mongo db on my host file system and trying to connect to that and failing as I didn’t installed.

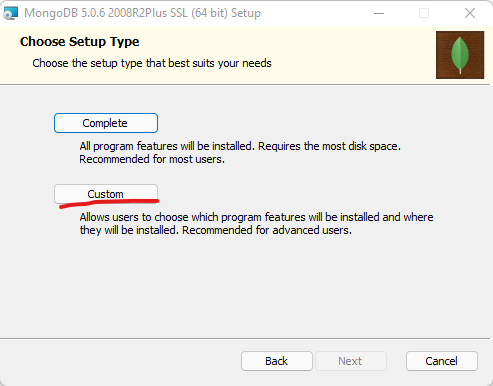
Now let’s install MongoDB on my host file system

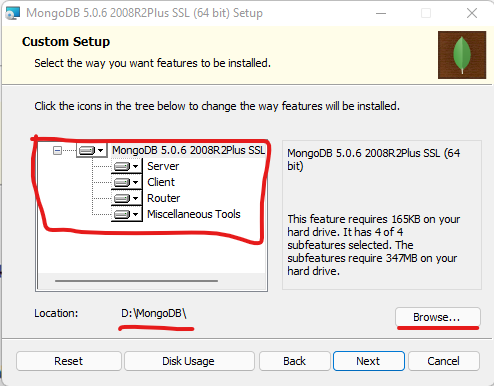
<https://docs.mongodb.com/manual/tutorial/install-mongodb-on-windows/>

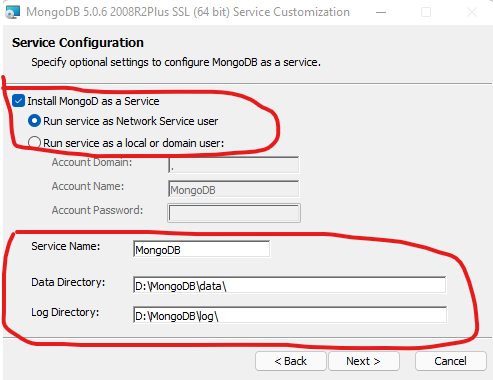
Download the MongoDB Community .msi installer from the following link:

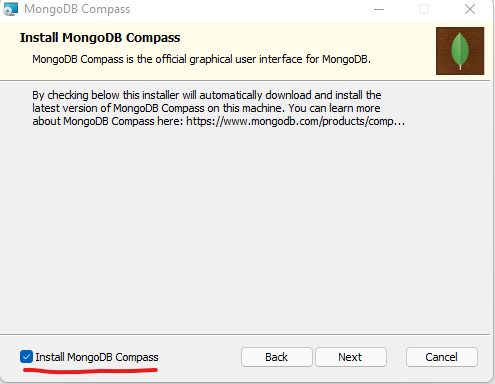
<https://fastdl.mongodb.org/windows/mongodb-windows-x86_64-5.0.6-signed.msi>

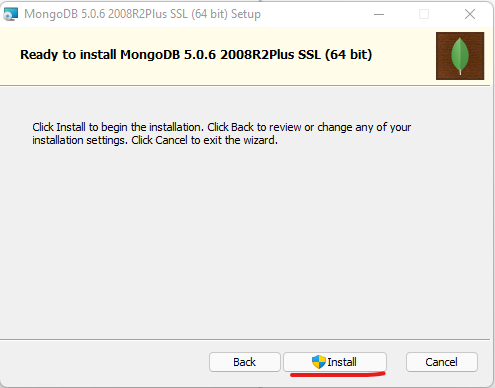


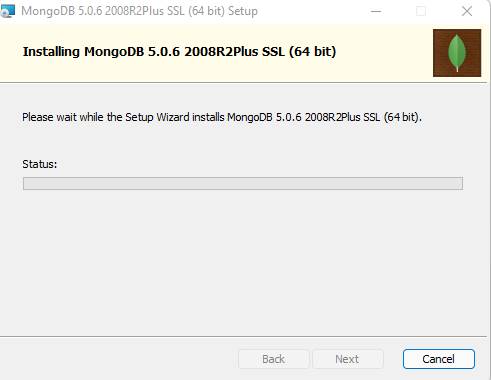


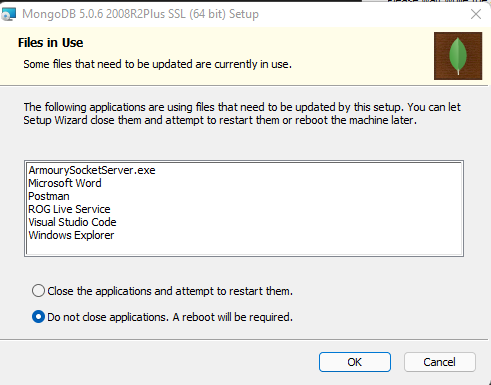


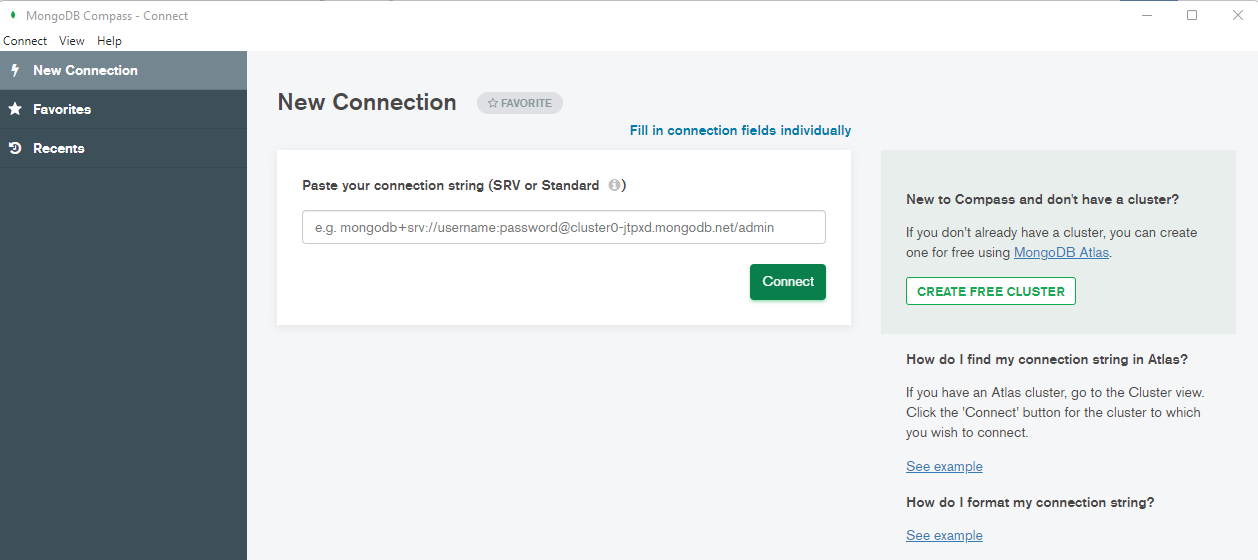


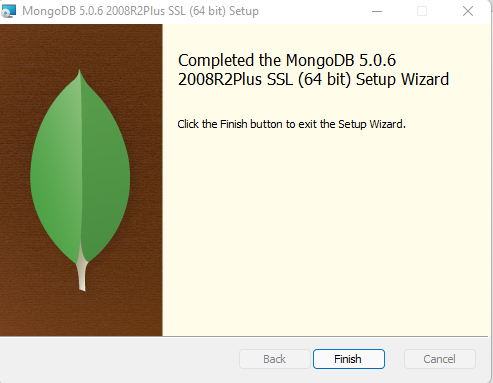




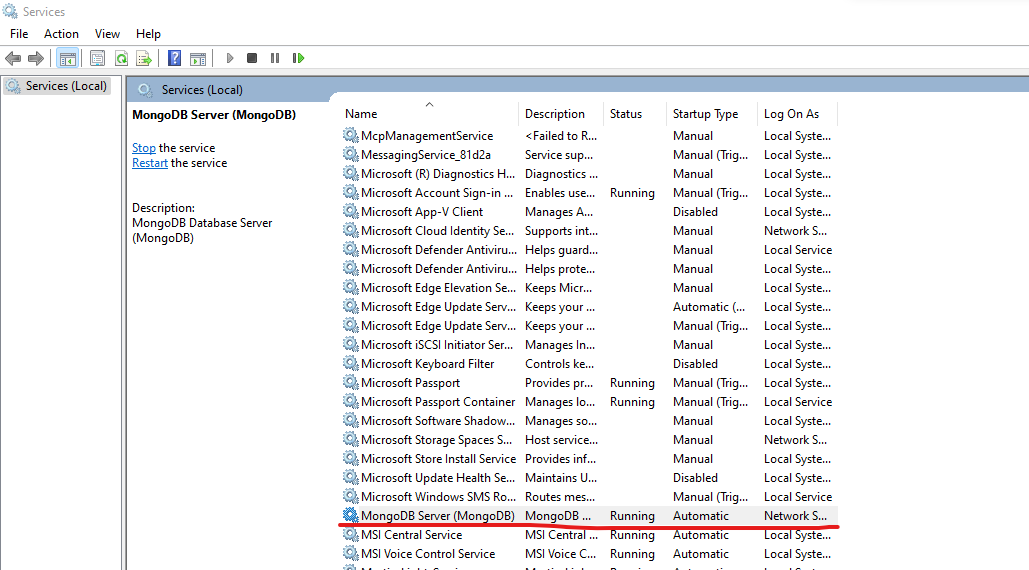






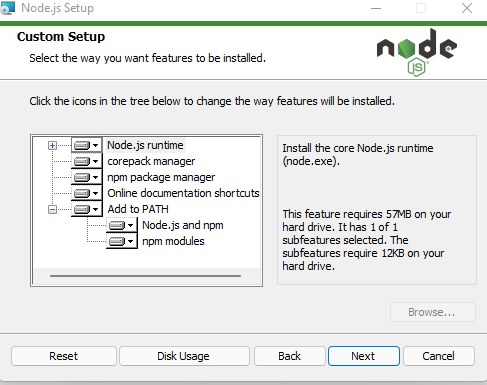


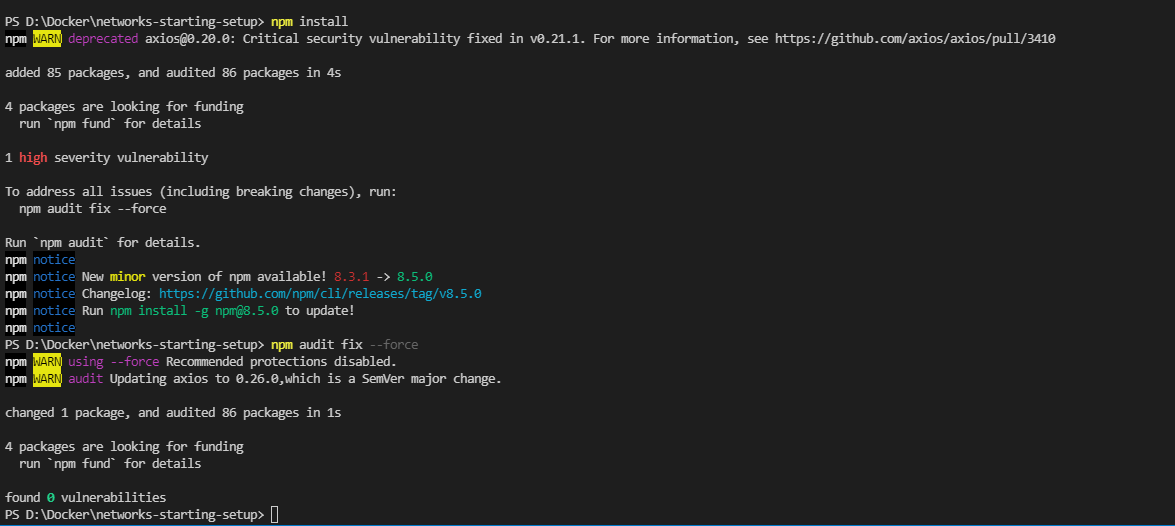
After my local machine restart, I saw the mangodb service is running

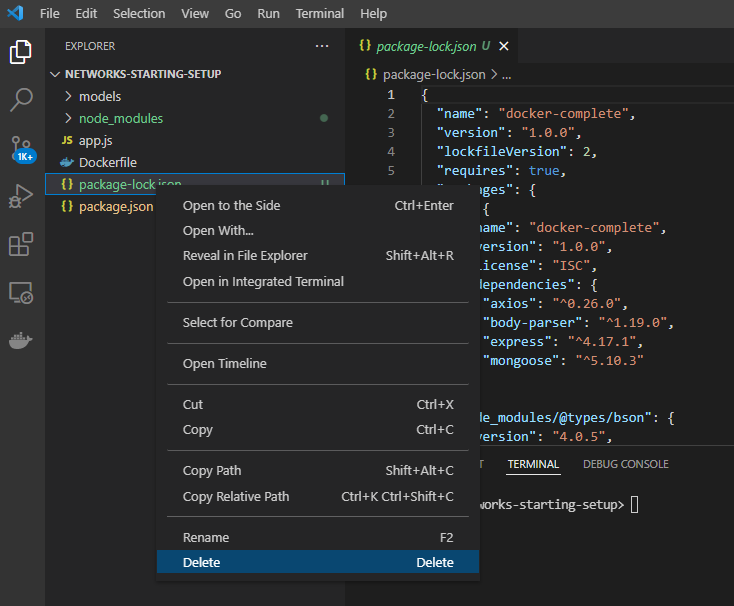


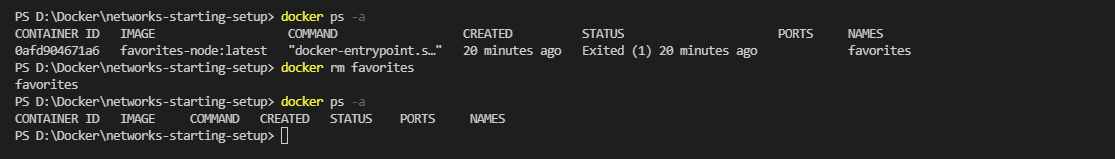
Now let’s start the container and see whether I am able to run the container or not after installing the MongoDB.

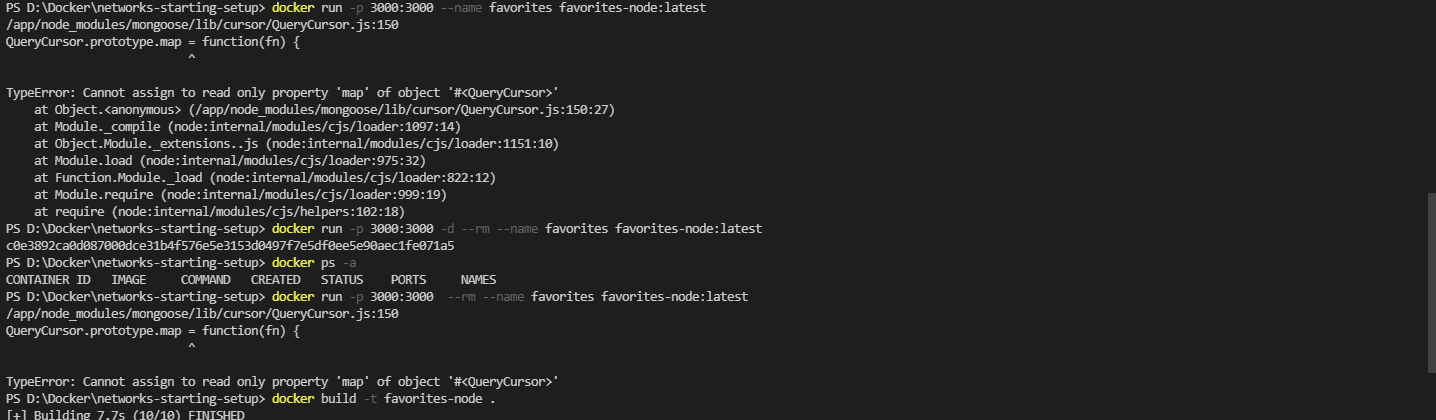
<https://nodejs.org/en/download/>



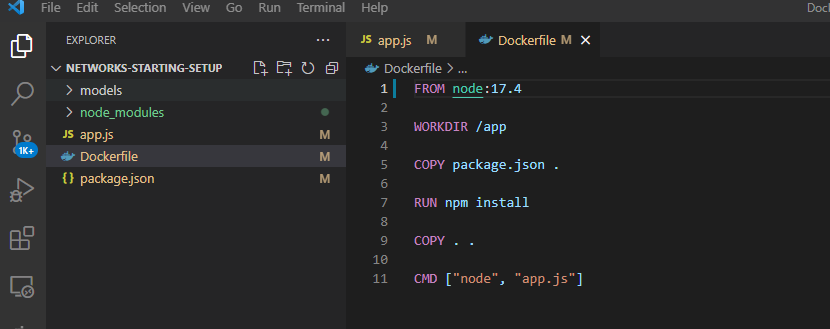


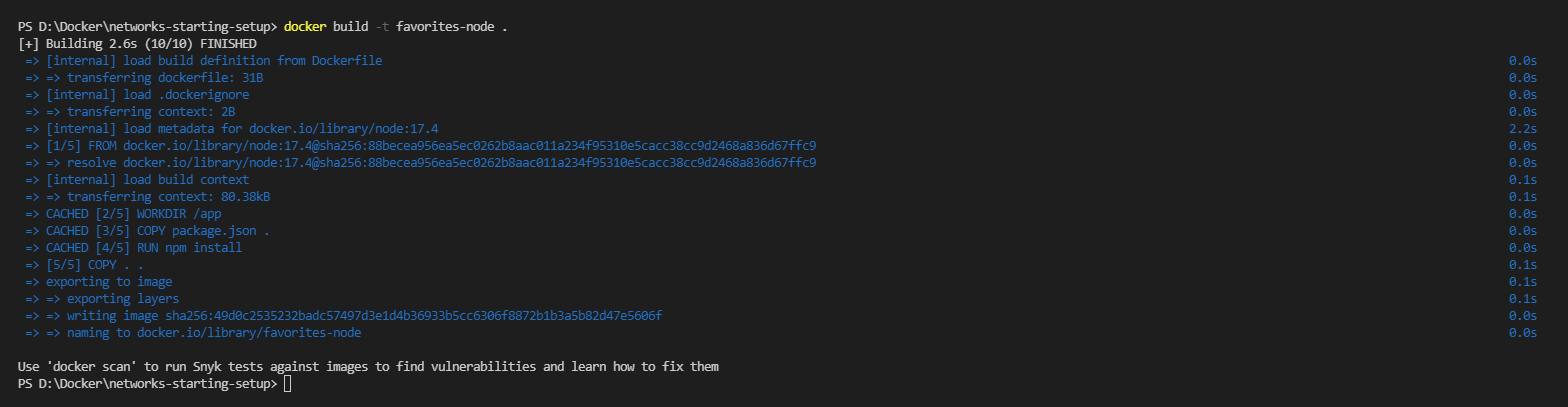


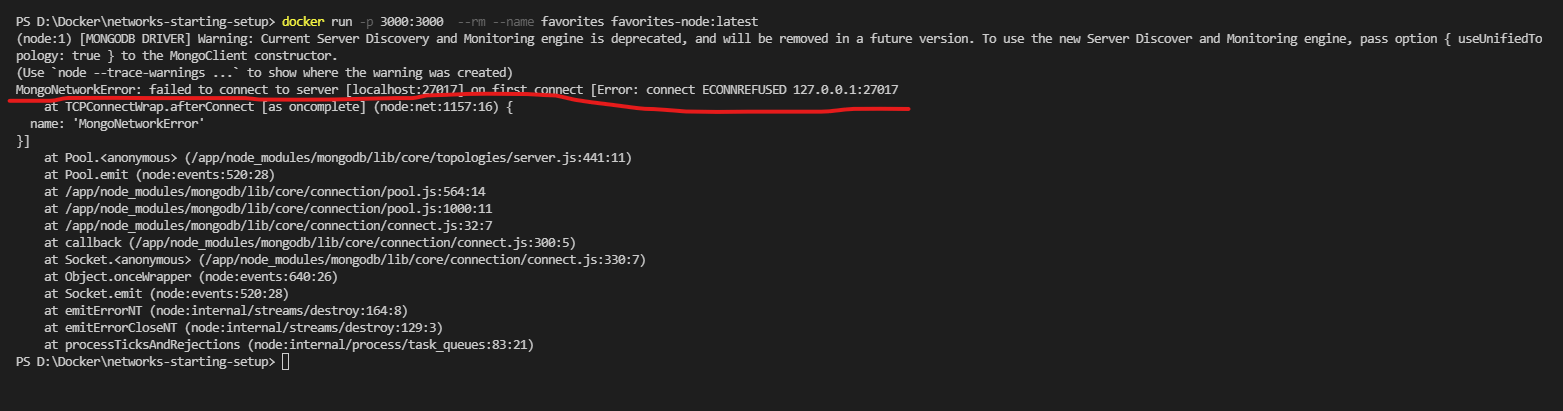




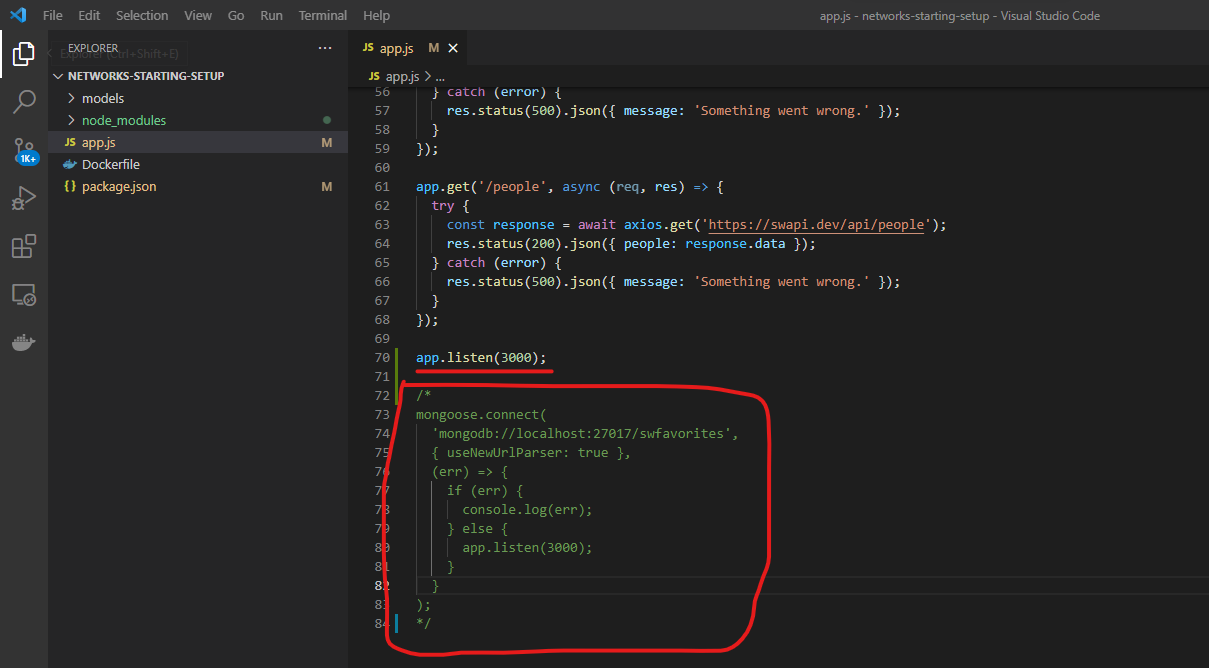
You will get this error if you don’t have latest version on NodesJS on container. So I am trying to install Node:17.4 version on the container.

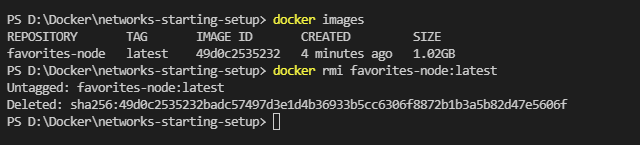


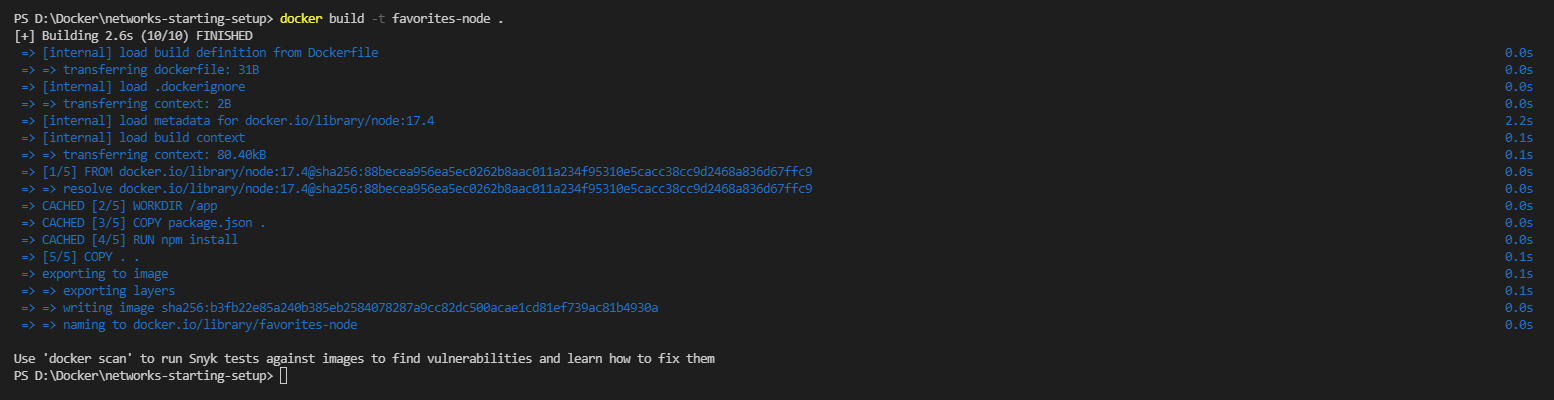




As I don’t written any MangoDB connection instructions on Dockerfile, it is unable to establish a local DB connection. So let’s comment the MangoDB connection code and start the container, and will see if the /movies and /people end points are working or not. Because /favorites get & post end points are depend on MangoDB connection and we will work that later.

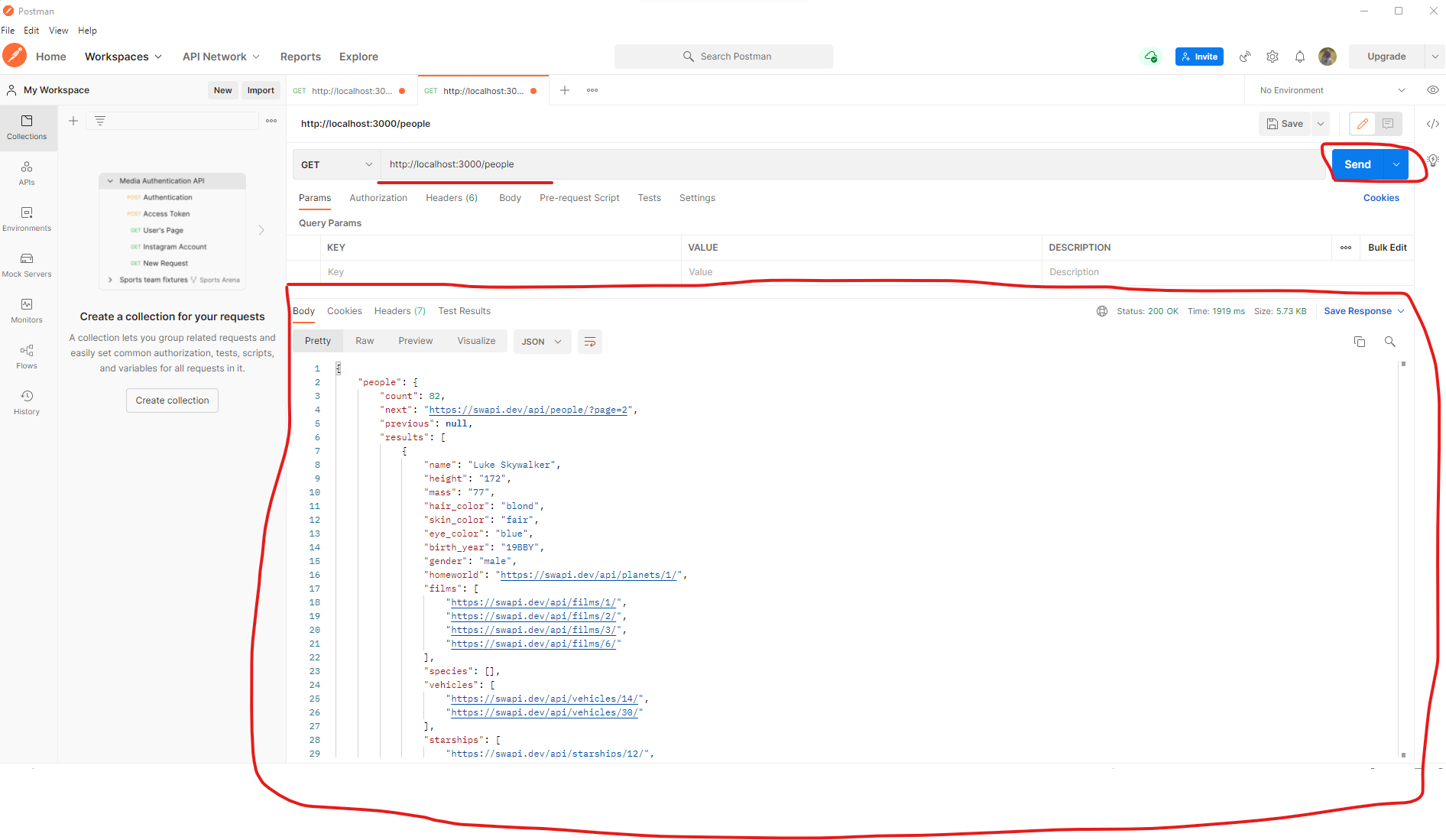
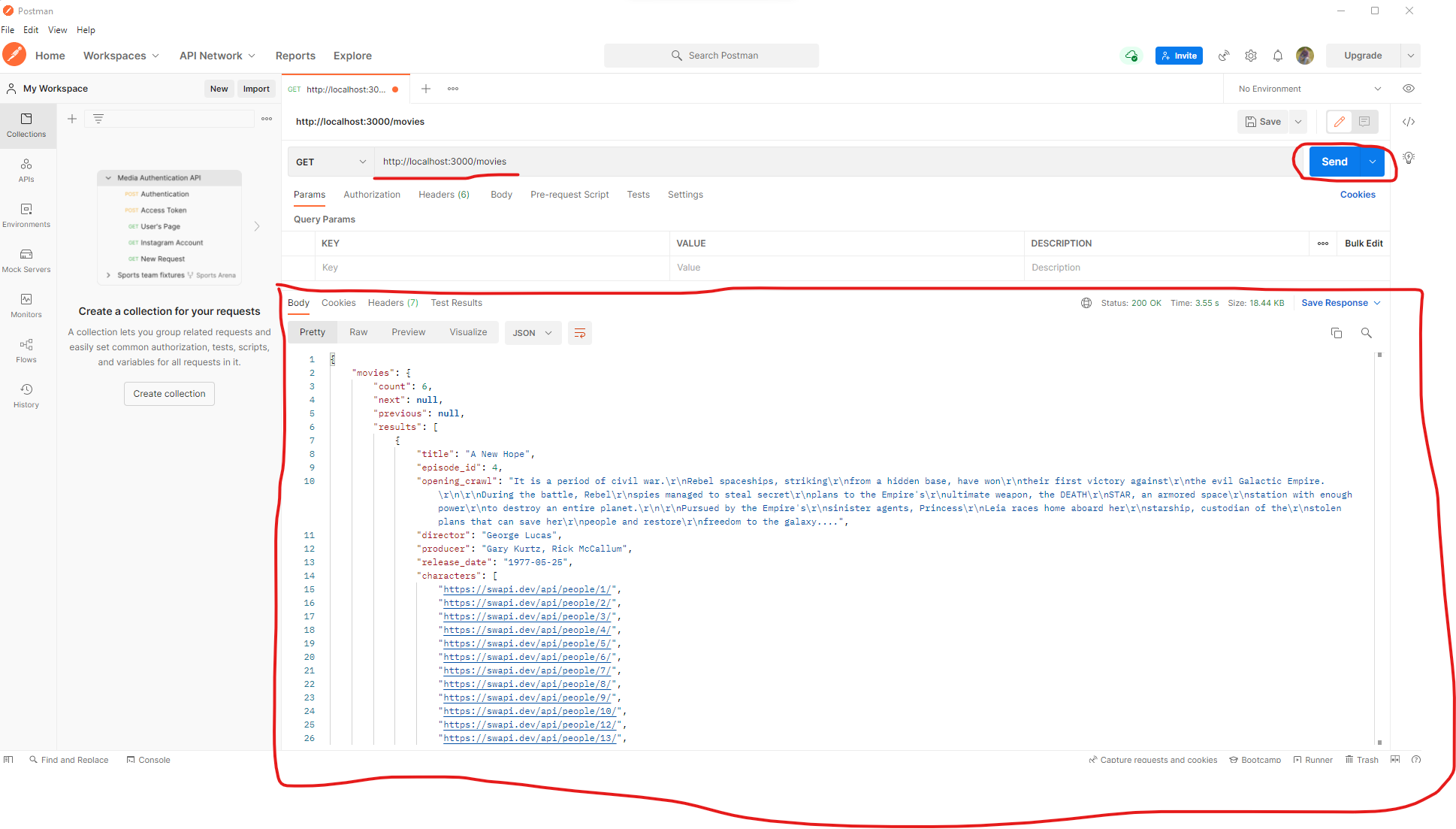








Open post man App and verify the end point /movies and /people.

SO 

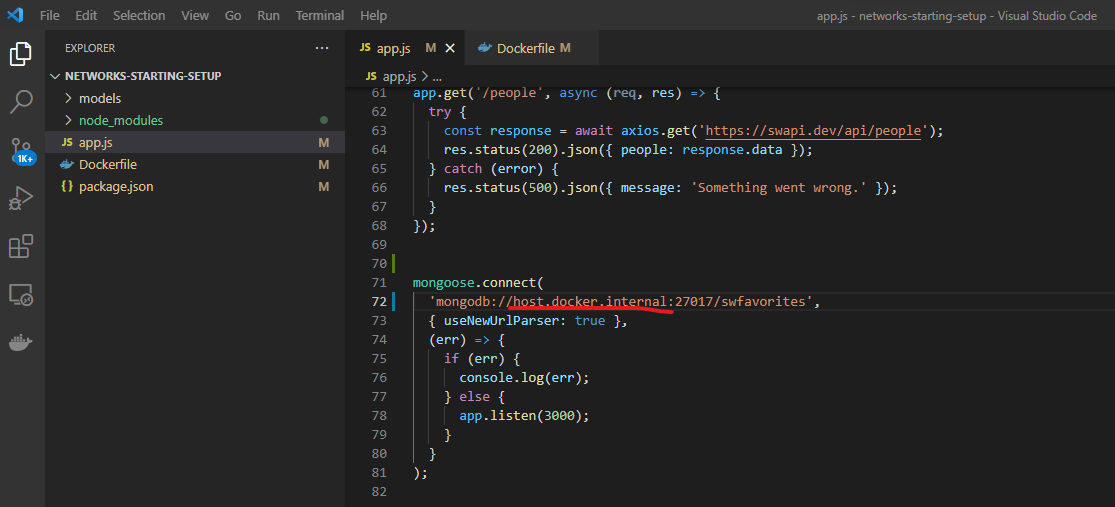
So as you observe, with you modifying additional network configurations, sending requests to outside world Just works, the API GET request sent from the container to startwars Dummy API just works fine.. Bingo!!!!

Next section we will how the other type’s connections will work.

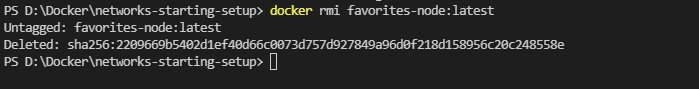
Sending Http requests to outside world is working, but the connection to your local host and accessing service is not working, because we had installed Mangodb on my local host file system and it not able to communicate to my localhost from the container, due to we had commented the MangoDB code previously.

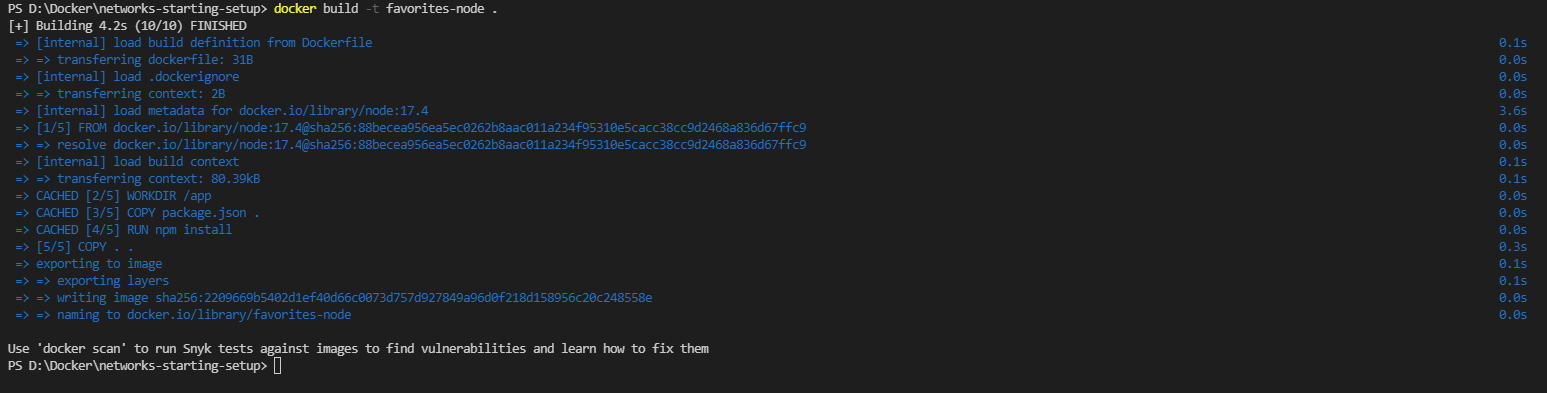
Now in order to enable communication from the container the local host, we need to change the domain to a specific docker domain “host.docker.internal”. This special domain is recognized by docker and it will converts the local host IP/domian as docker container IP/domain.

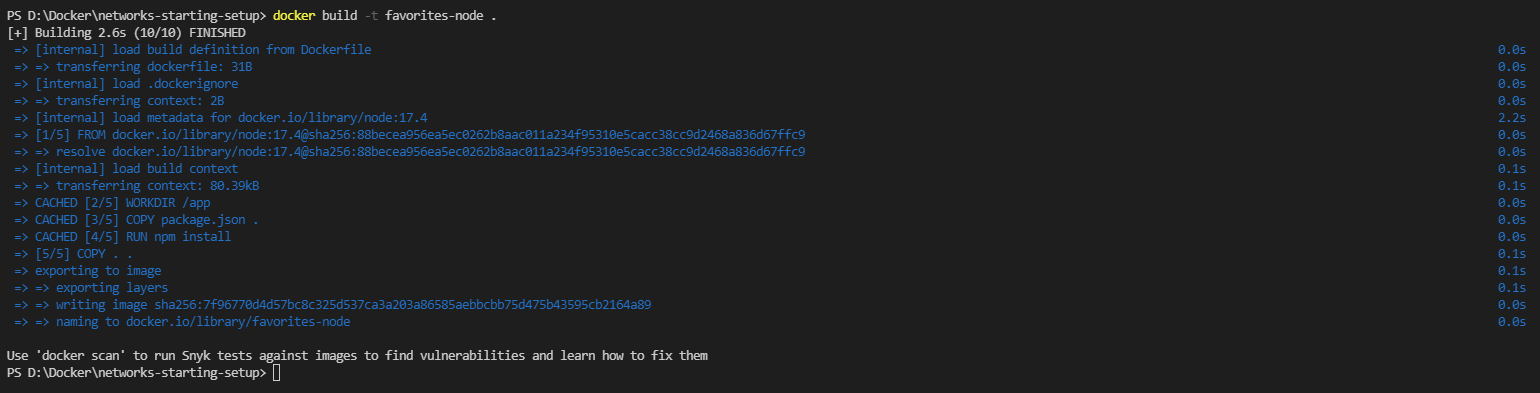
This will work for other types of requests also here we are use mongodb:// but it will work of http:// requests you need to replace <http://localhost> to <http://host.docker.internal> in order to communicate with local host file system.

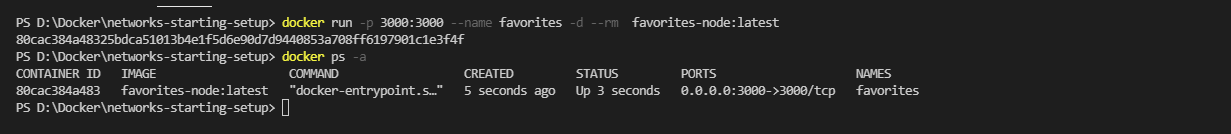


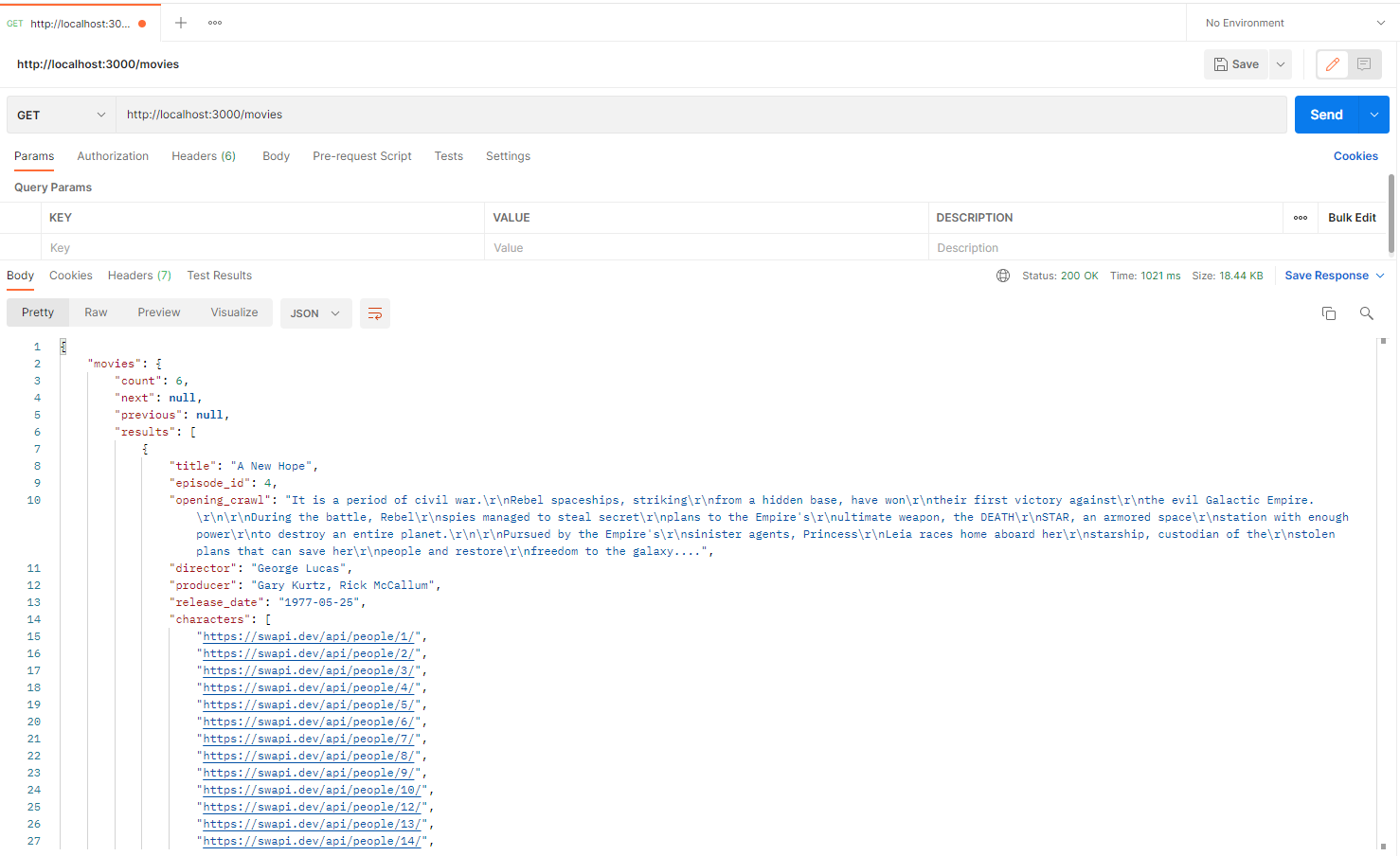
As we had made the changes on app.js, let’s re-build the image again.

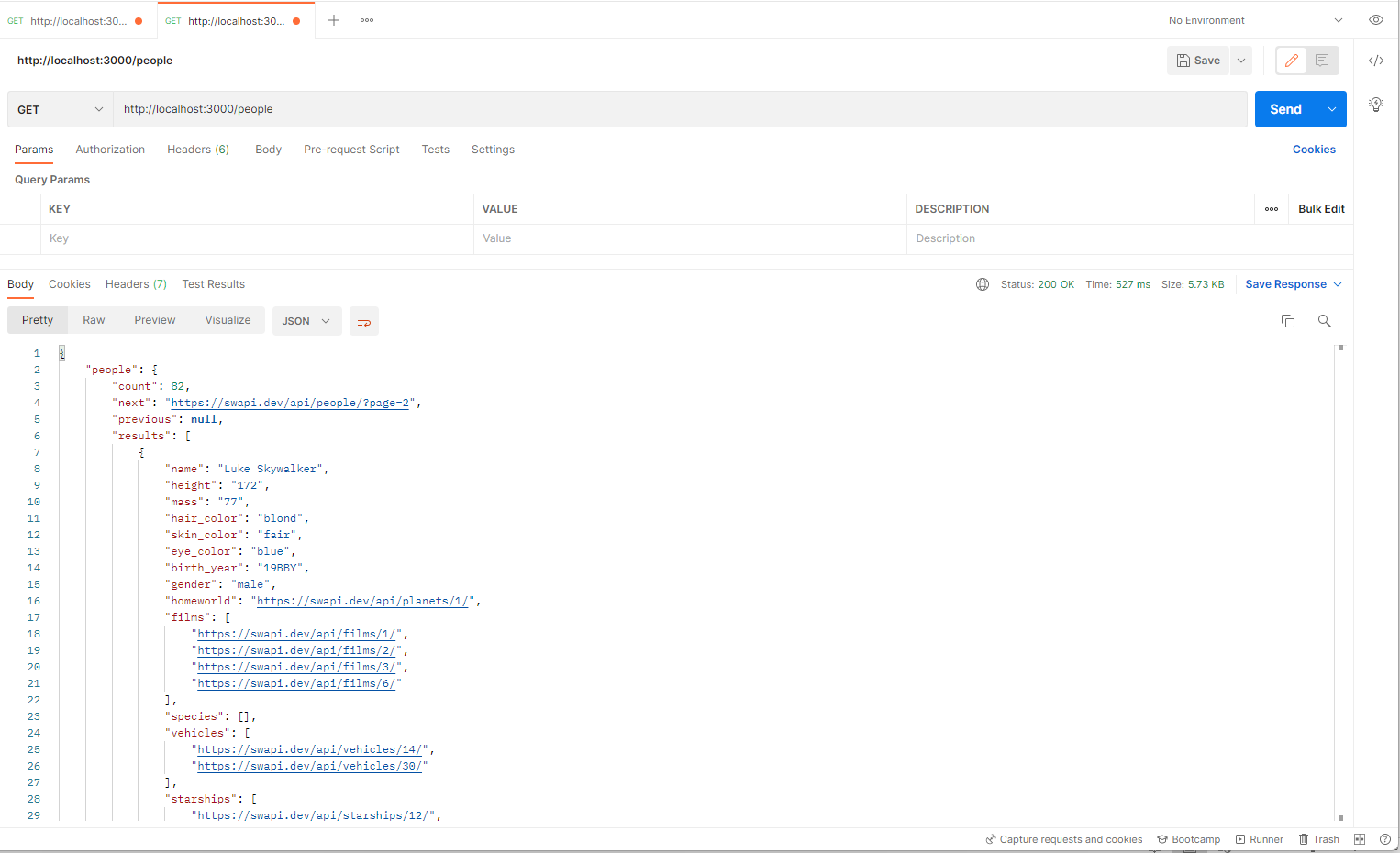




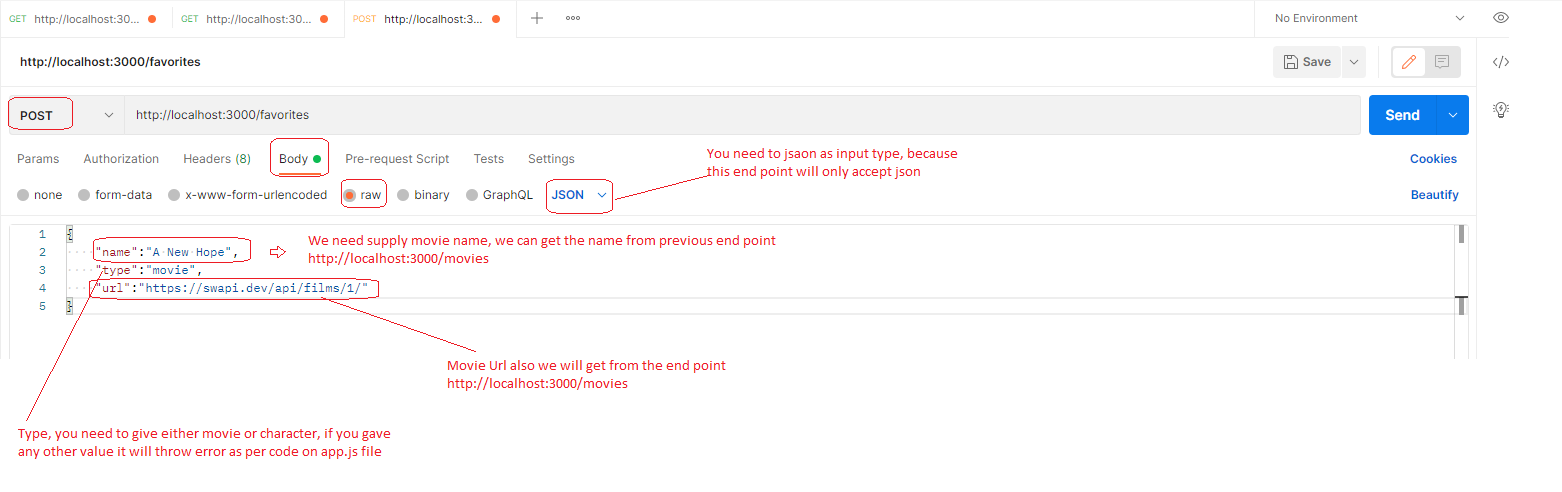


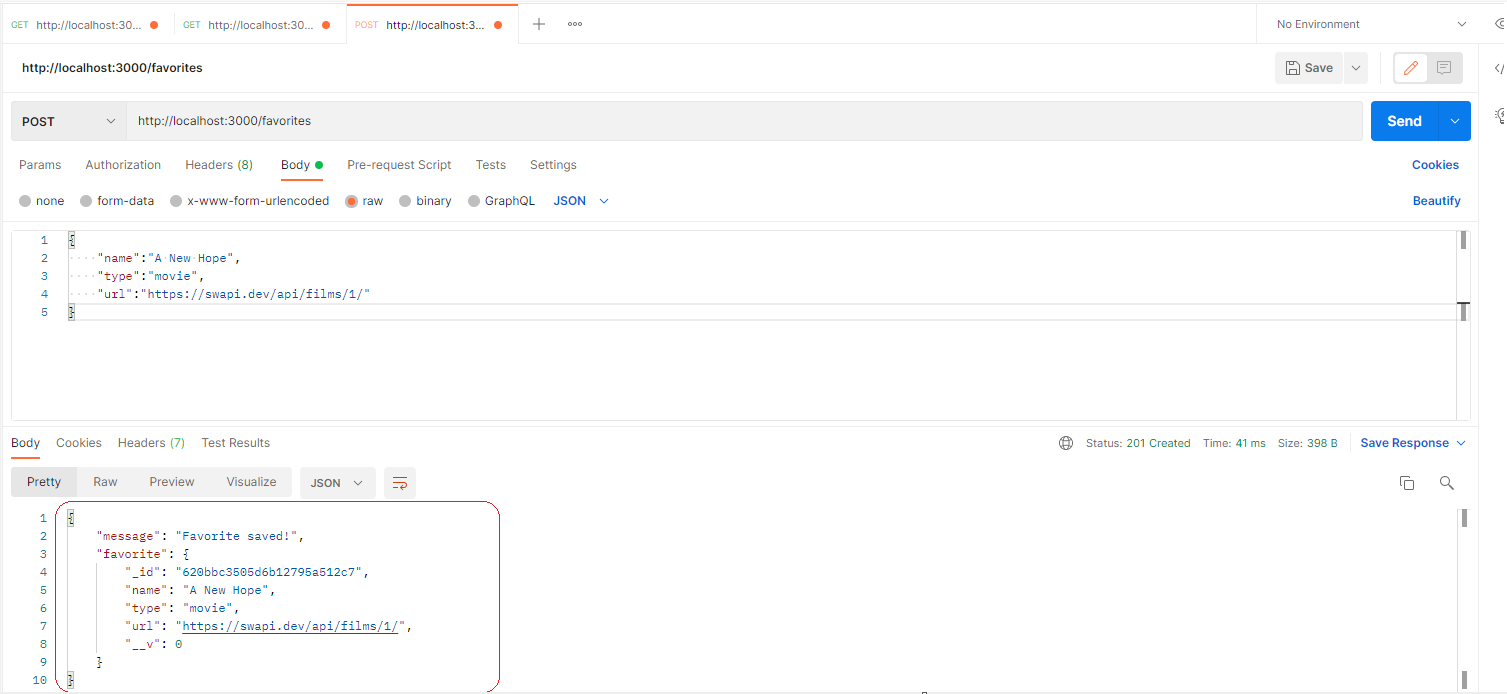




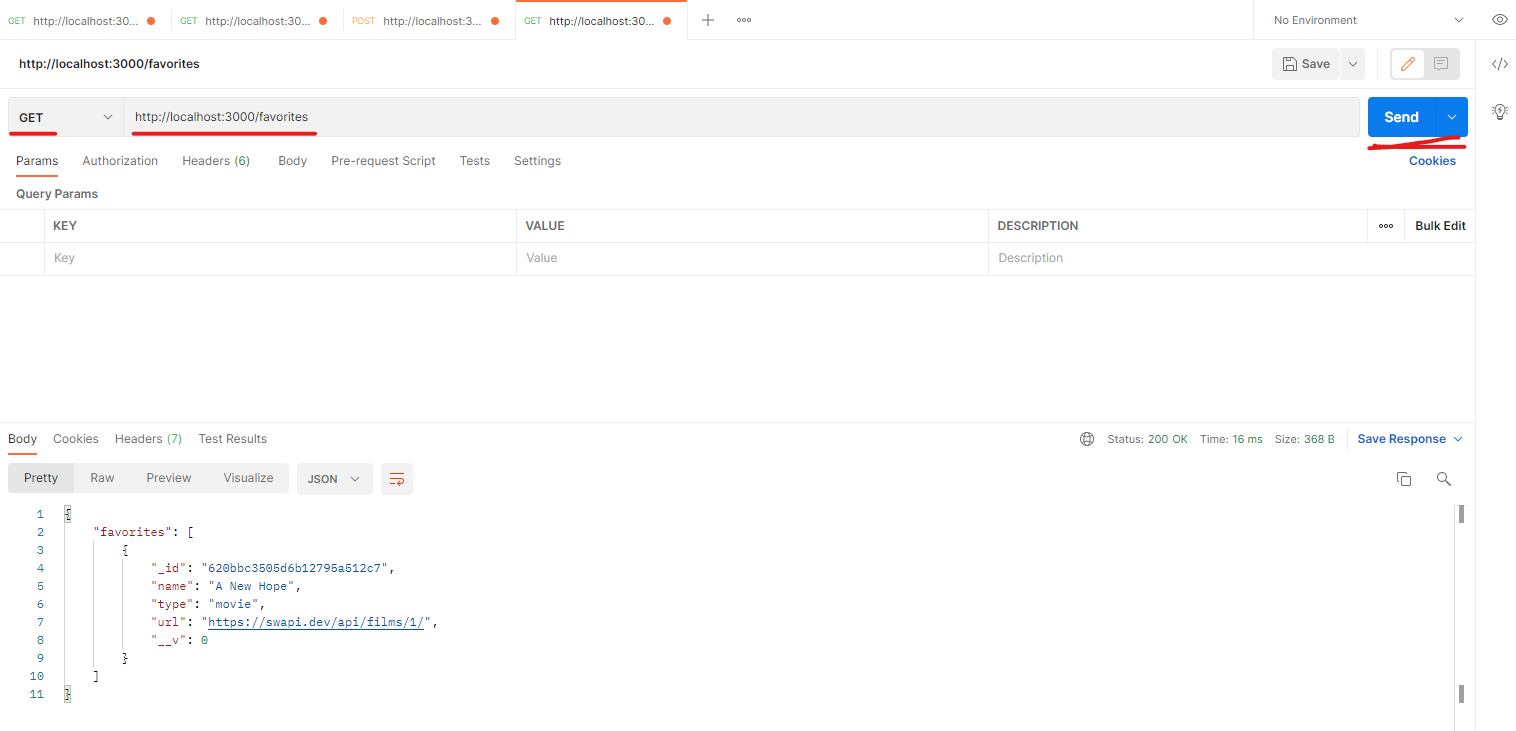


While using post request, we need to follow the below instruction to update the data into mongoDB on our localhost.





Now in order to use the get on favorites, use like below

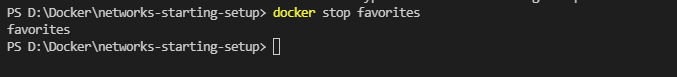


Now you can see that communication from your container to localhost machine is possible via special domain “host.docker.internal”.

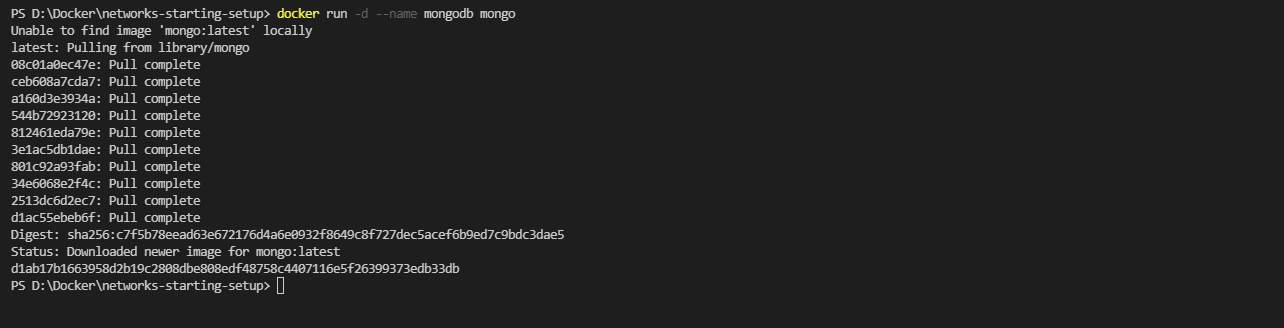
Now let’s the other type of communication which is container <-> container.

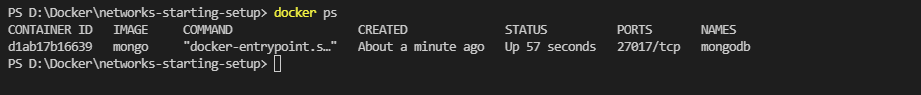
For this I am planning to build two container one container contains Node app and other container contains mangodb and both need to talk to each other.

Up to now we had installed mongodb on my localhost and connected from the container to localhost, now I need to configure another container with mongodb.



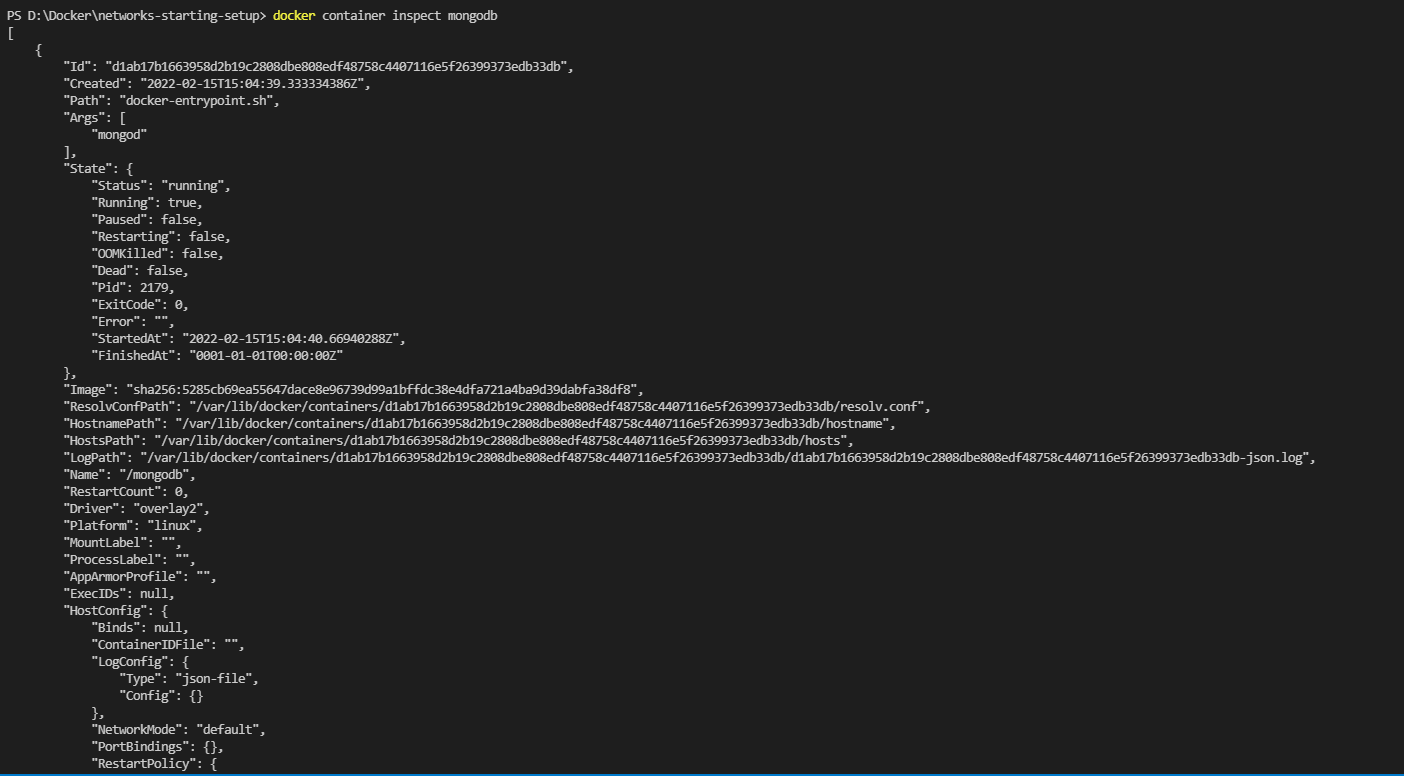
Creating mongodb container now, there is no need to write a separate Dockerfile file for this we have an official mongodb image on docker hub, we can use that for our purpose.



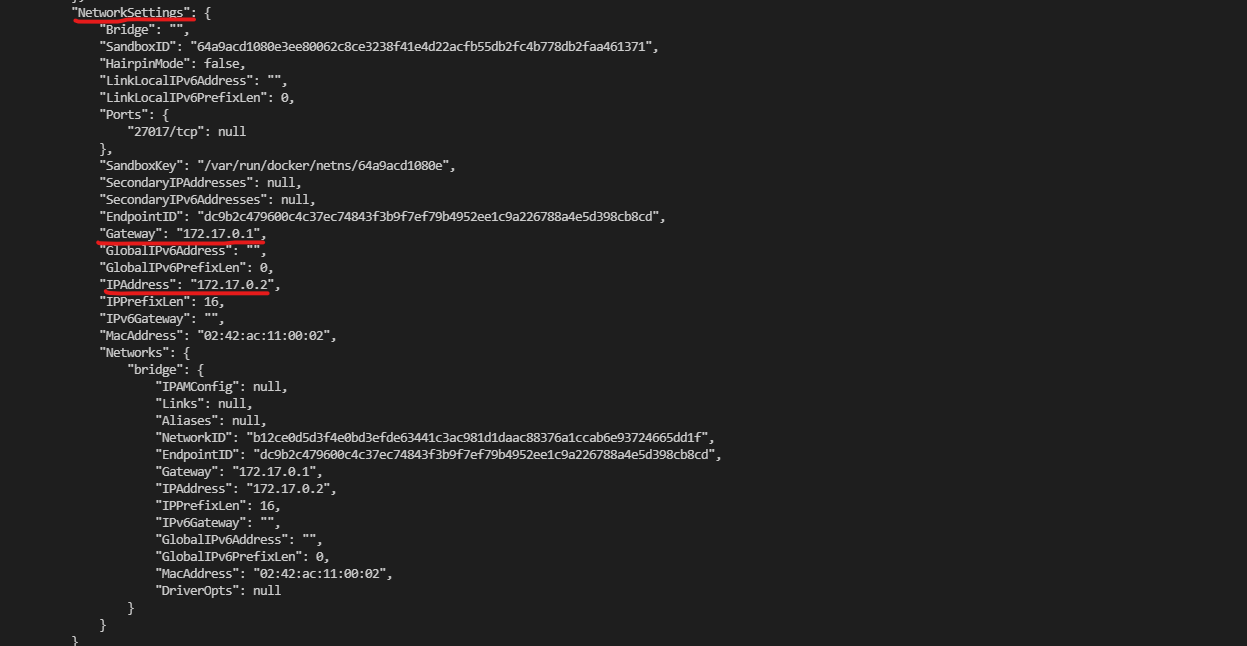


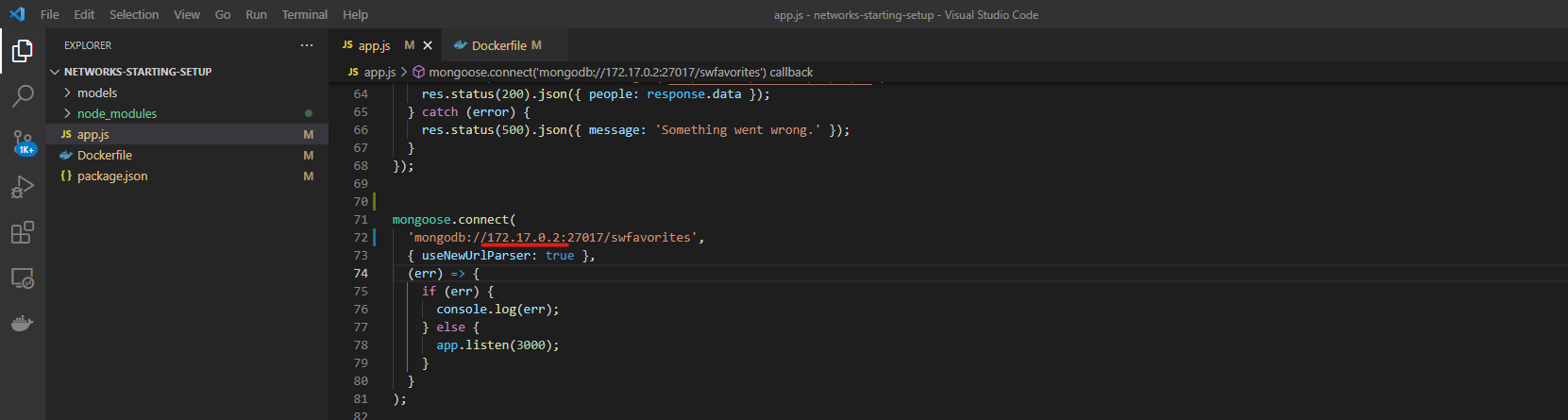
Now the application node will need to connect to this mongodb container, previously we had placed “host.docker.internal” to communicate but this doesn’t work, we need to replace the IP of the mongodb container on the app.js to connect.

In order to find the IP of the mongodb container, issue the following command.

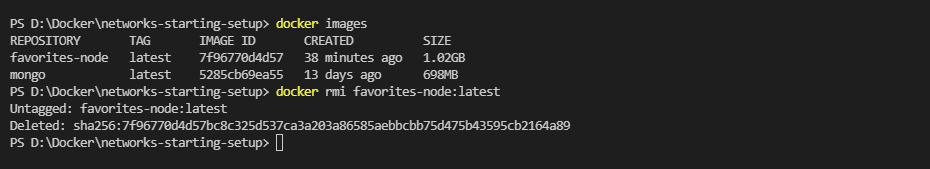


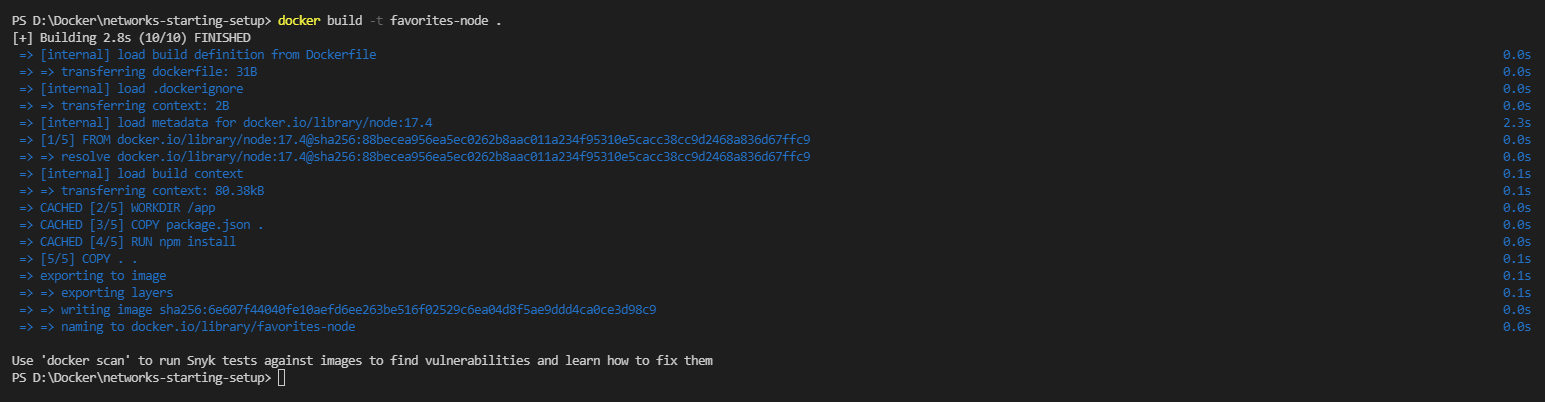
Here go to network settings and find out the IP address as shown below.

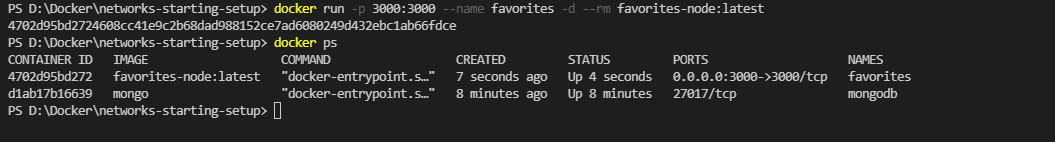


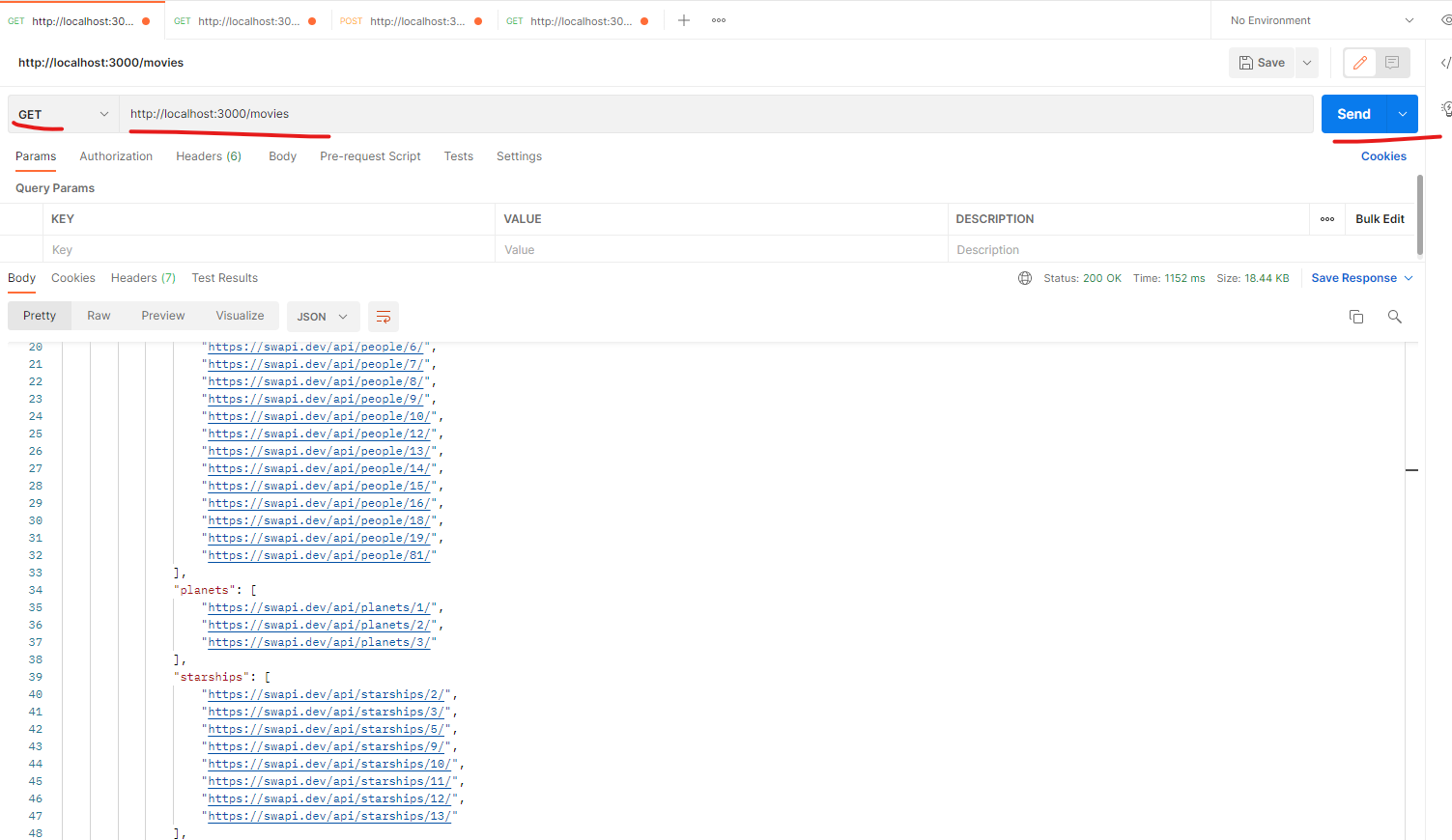


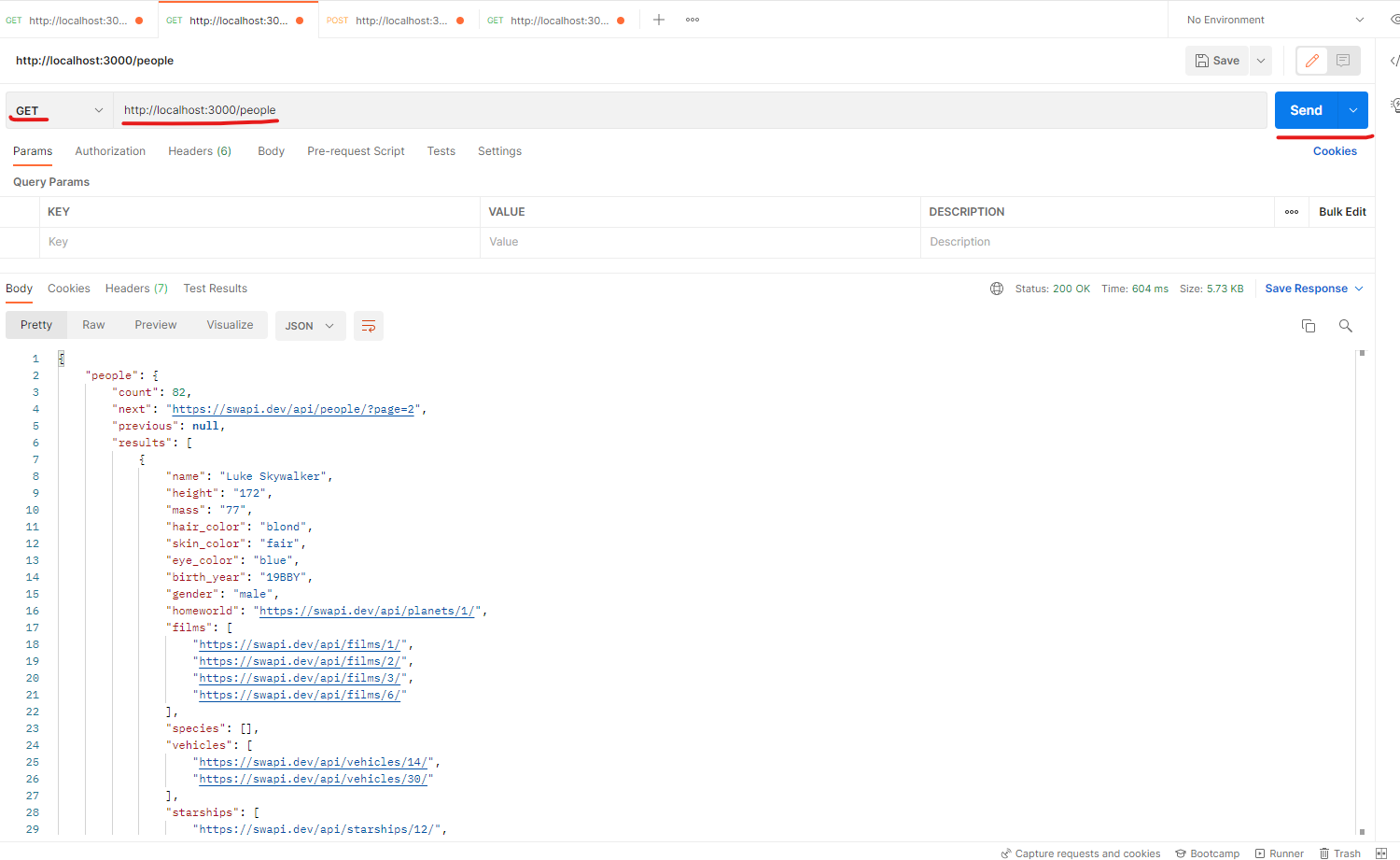
As we had modified the app.js file, let’s re-build the image and start the node app also.

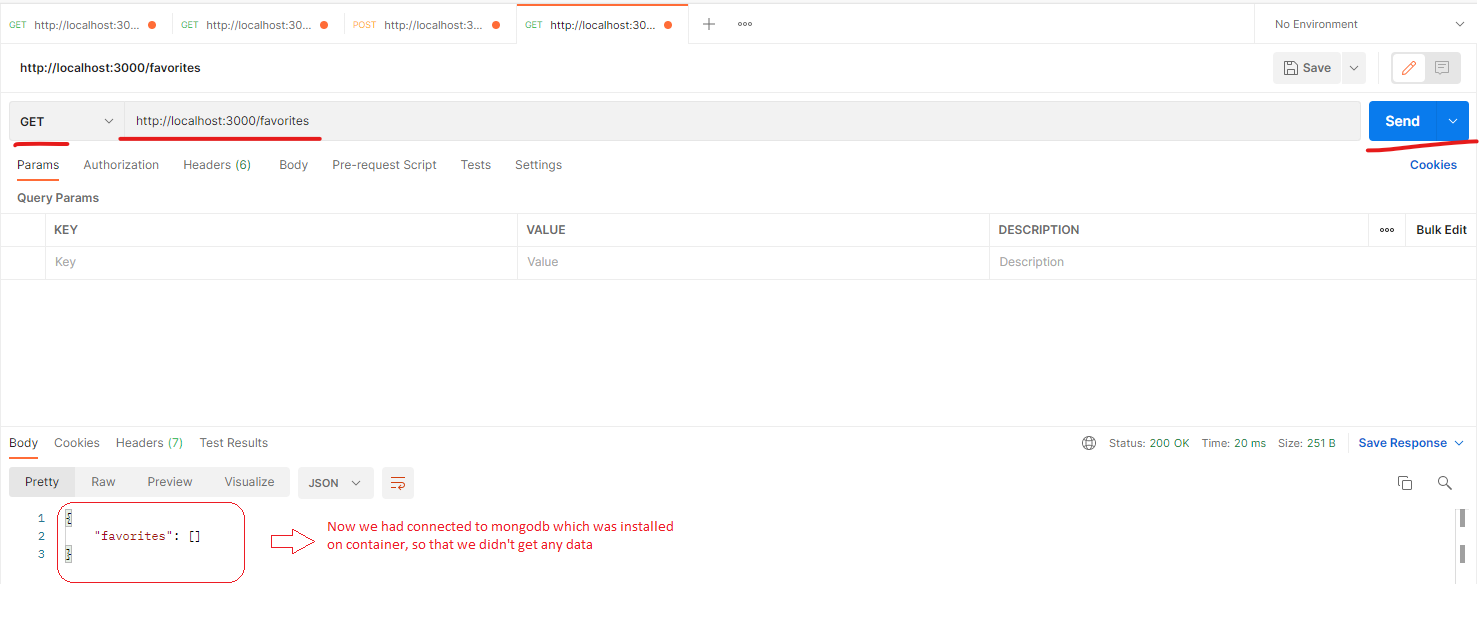




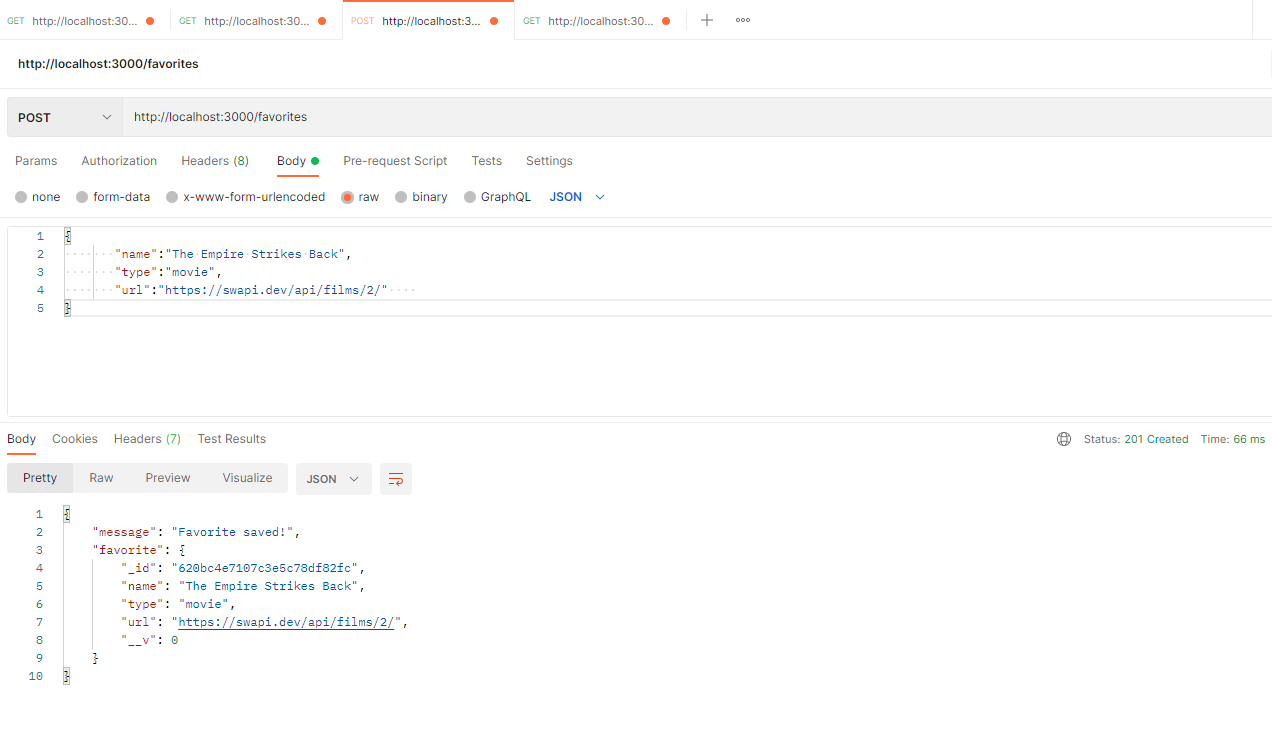


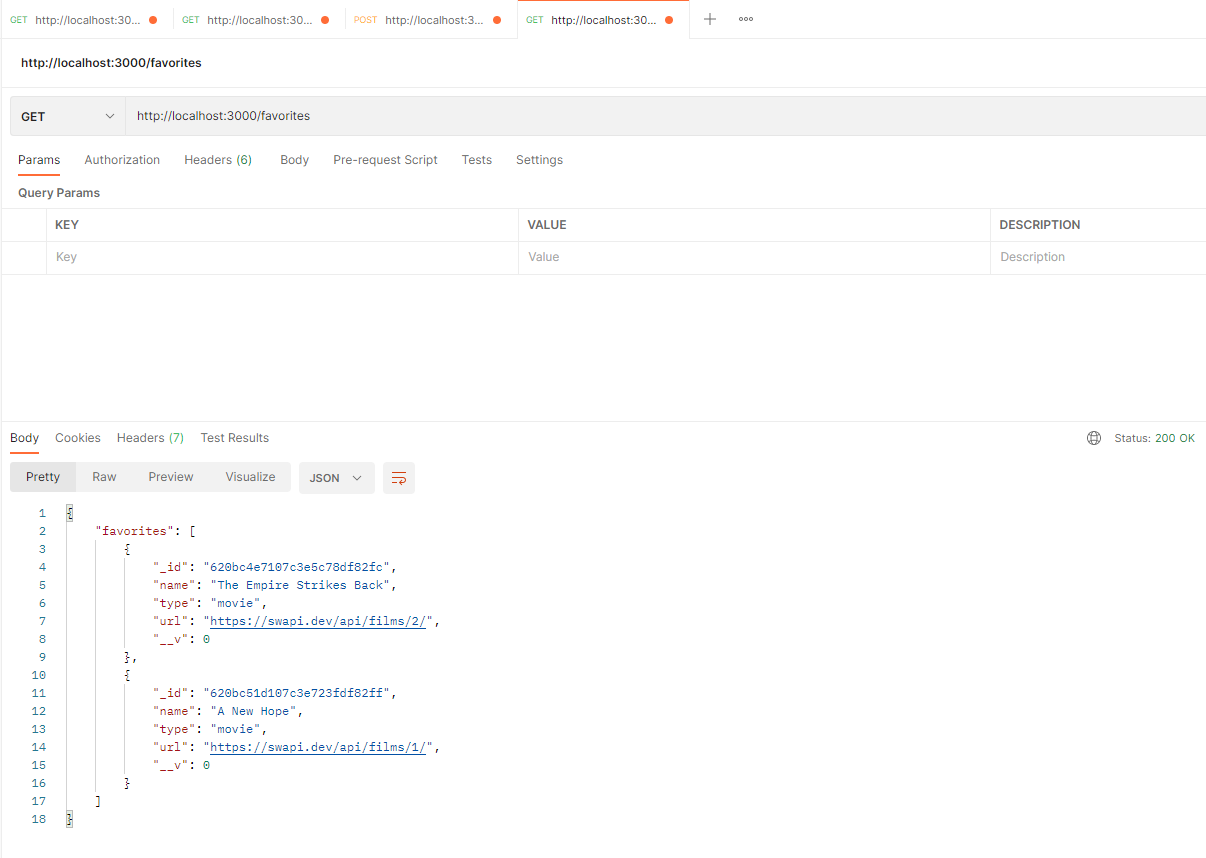






Now let’s insert the data by using post method on /favorites api and see whether the data had been inserted to the new mongodb or not.





Here we have a problem, that we need to issue the command “docker container inspect mongo” and find out the IP and need to hardcode the IP on Node container, every time we need to change once the IP changes.

There is an elegant way we can update this network section, we will do it on next section.

We can enable communication between two or more containers by simply placing all of them into one network.

docker run --network my\_network …

Network

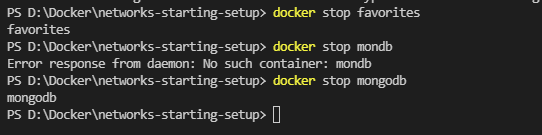
Container 3

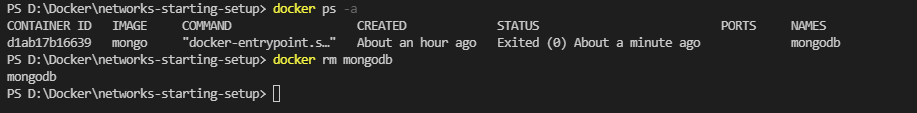
Container 2

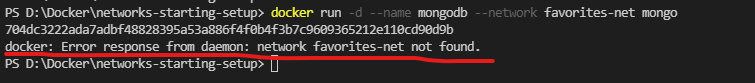
Container 1

Within a Docker network, all containers can communicate with each other and IPs are automatically resolved.

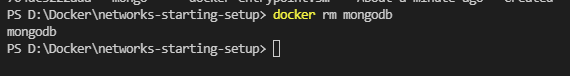
Just by placing all the containers into one network it will automatically communicate each other, so we need to create one network and start the containers by placing them into that network.

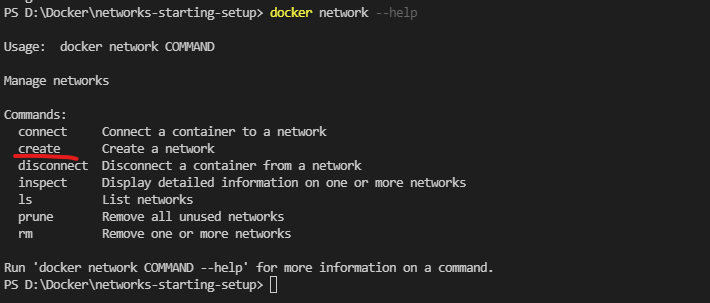


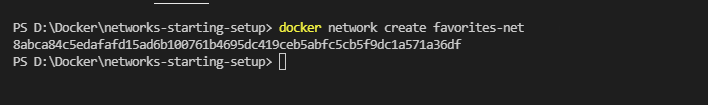


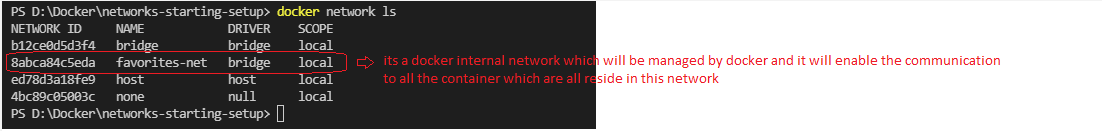


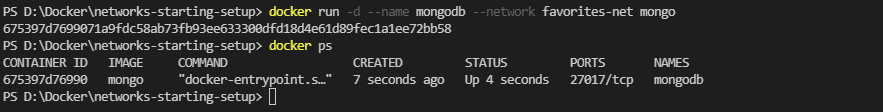
Unlike the volumes, docker won’t create the networks automatically if it doesn’t exists, we need to create those manually using the command “docker network create”



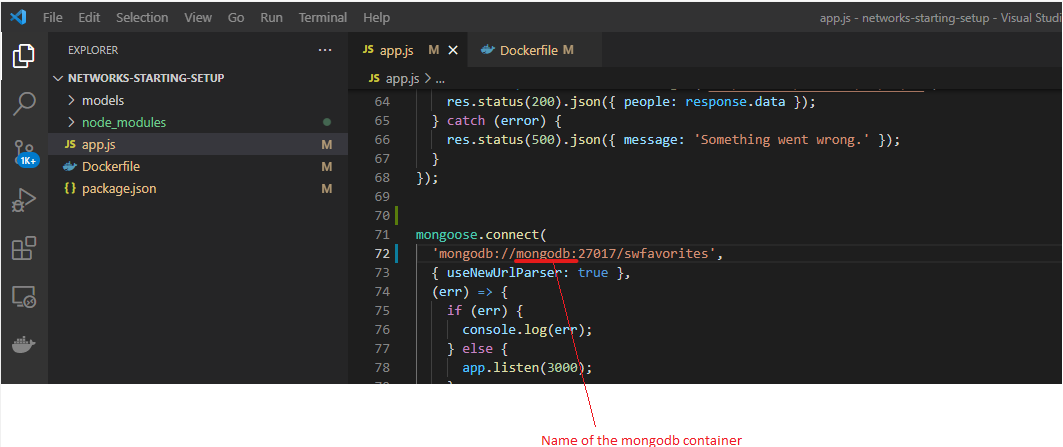




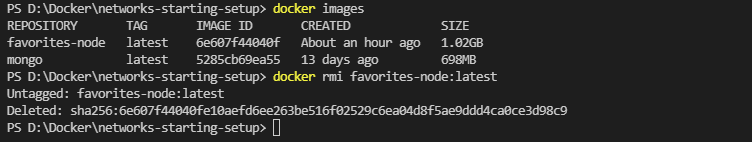


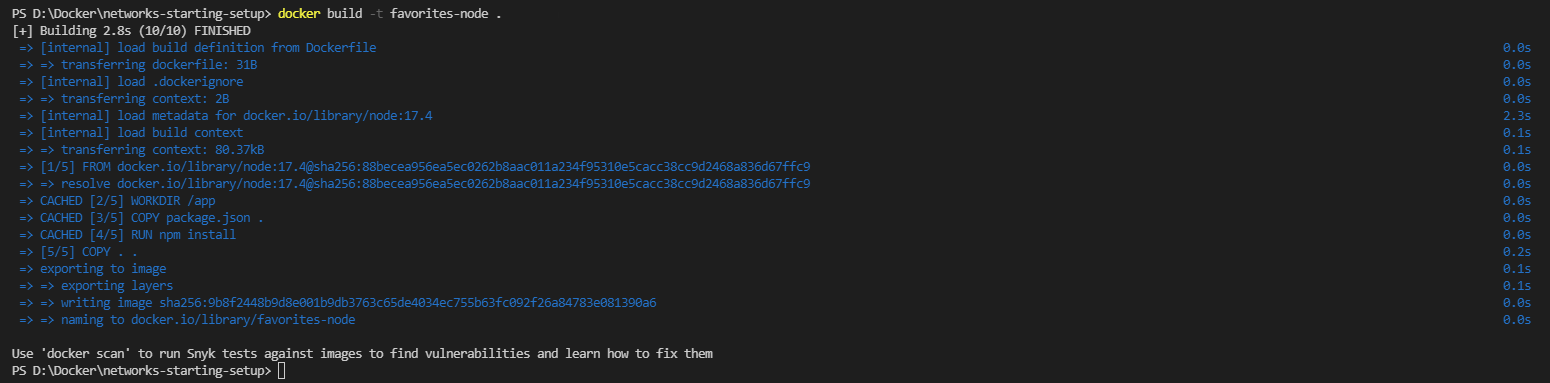


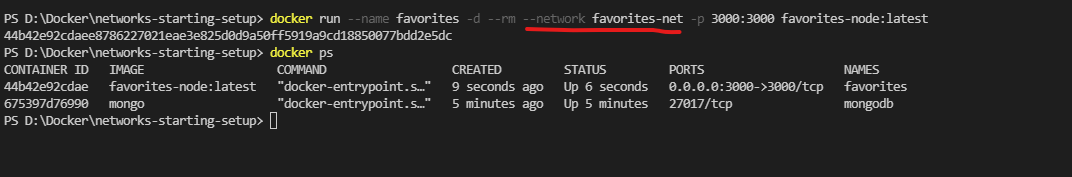
Now we need to configure the NodeJS code to connect this mongodb. So cool thing is if two containers are in same network, you need to replace the name of the mongodb container name inside the url on app.js file.

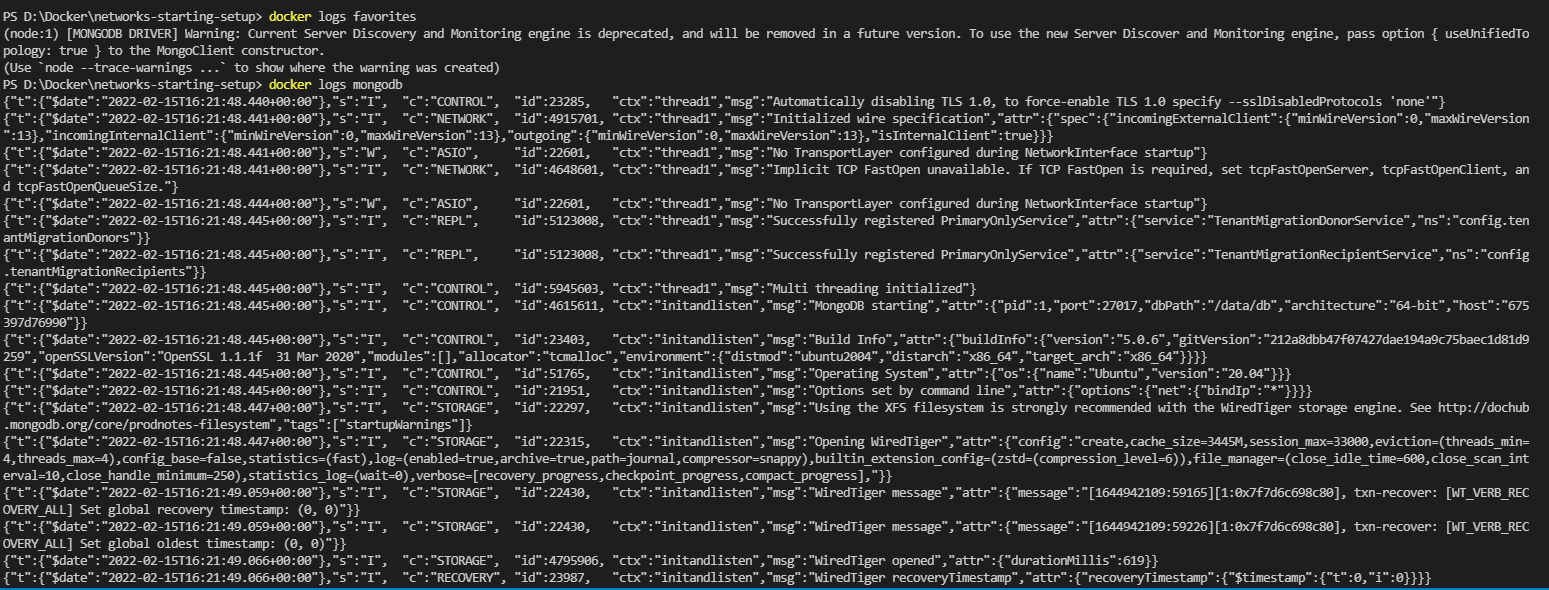


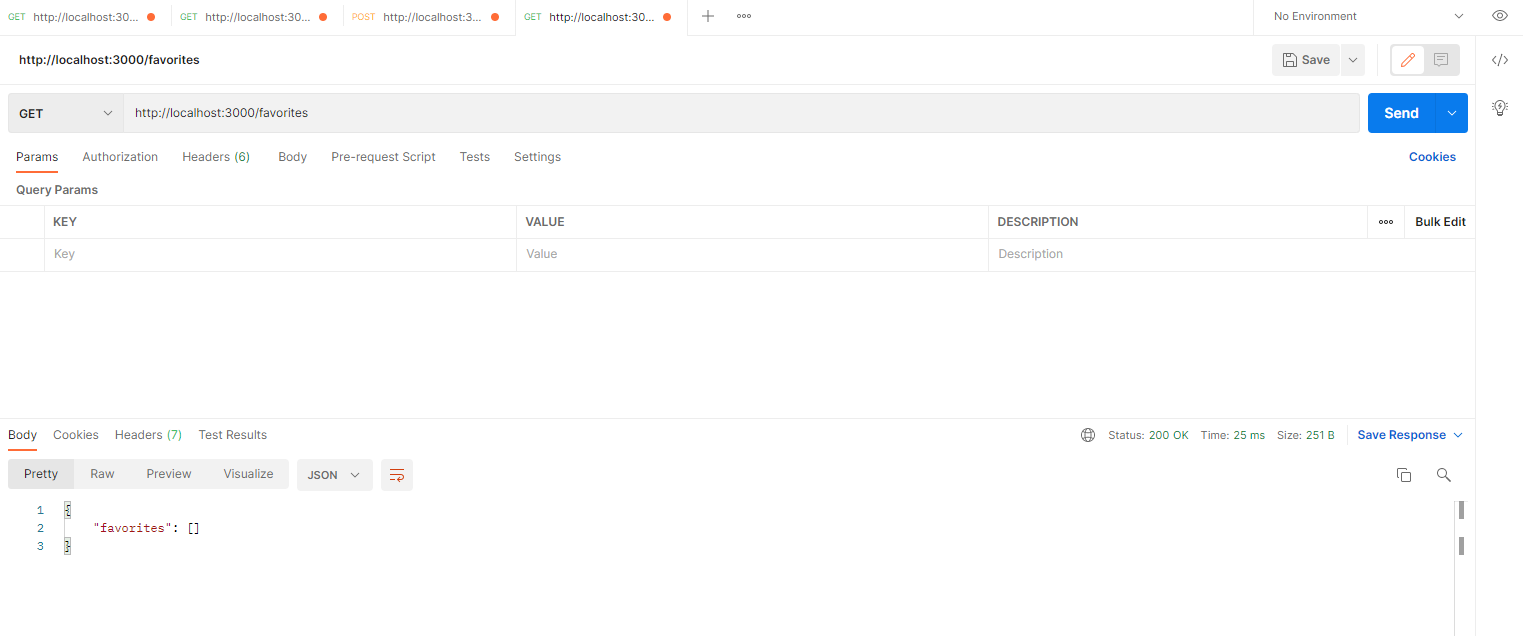
Now let’s rebuild the nodejs container and see if it starts or not as we had made the changes on code.

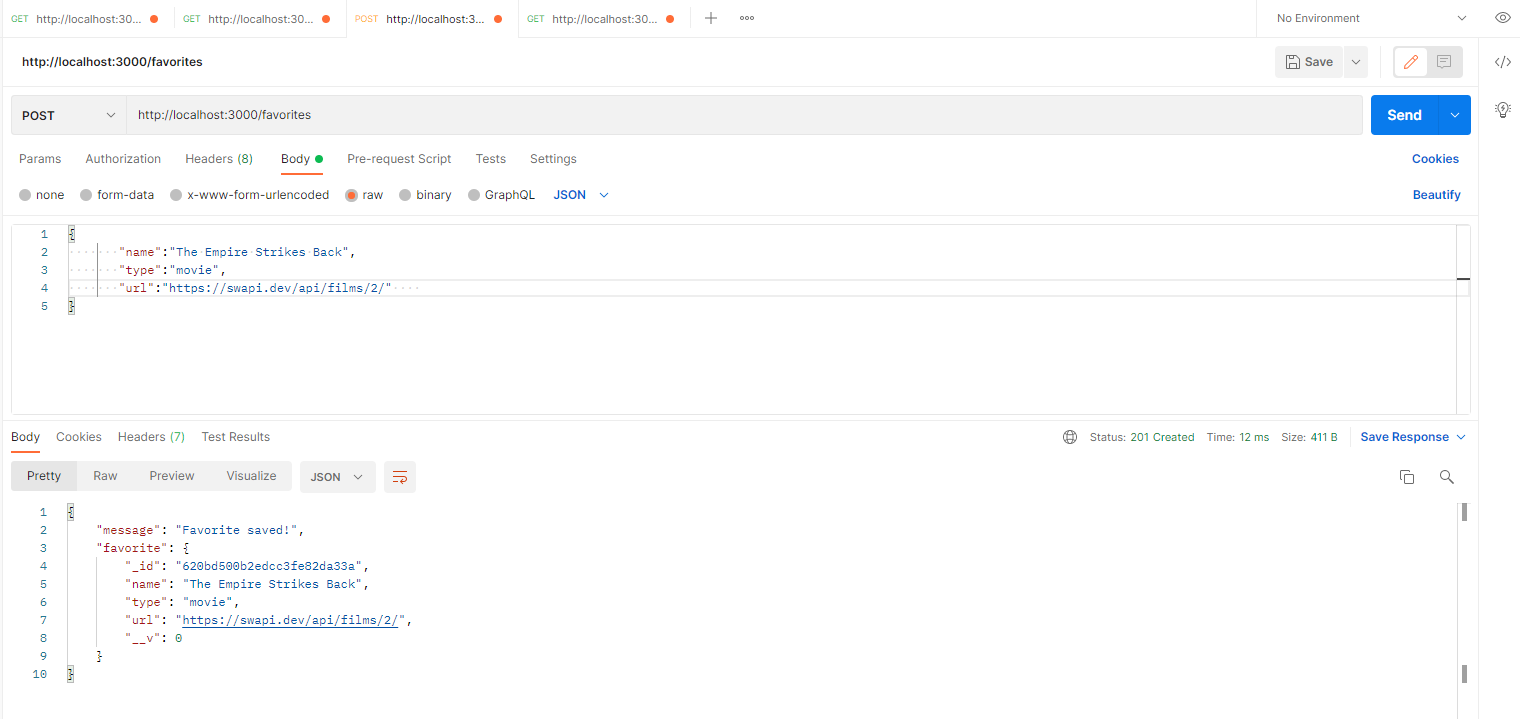


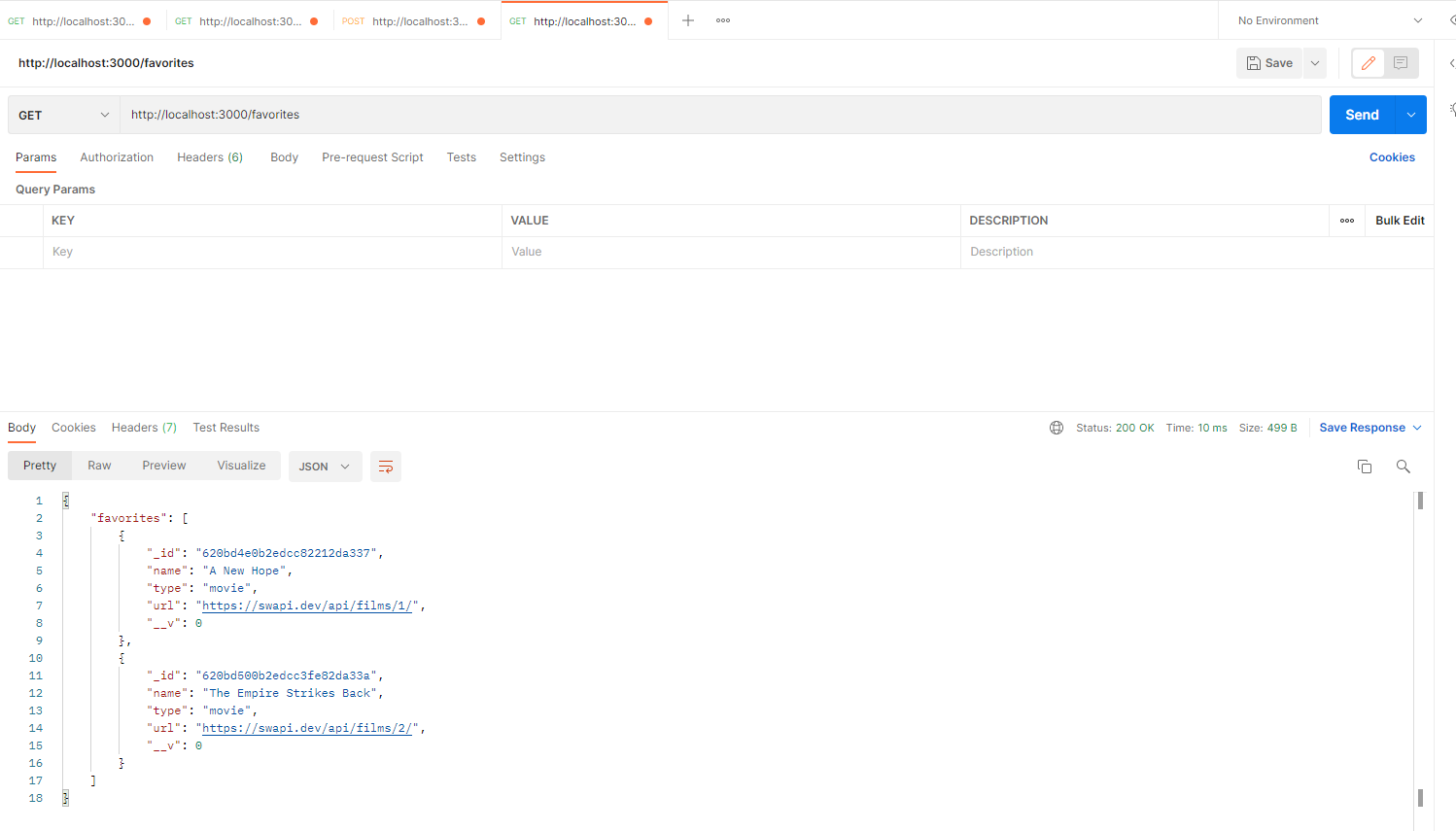












Two container won’t be able to communicate unless either you need to hardcode the IP of second container or other elegant way that create a common network.

Note: if you observe while starting mongodb we didn’t used –p option because this is required only if the communication is needed from container to our localhost or outside world.

Here our favorites app container need to only communicate to mongo so we won’t need to publish the port.

