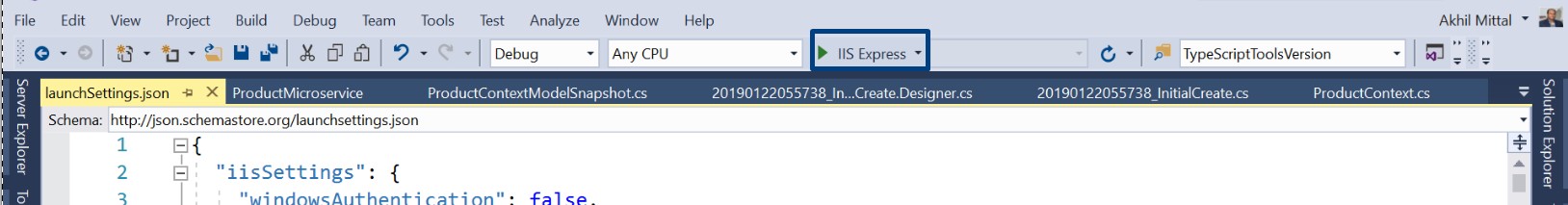
# Lab 11

# Running the Product Microservice

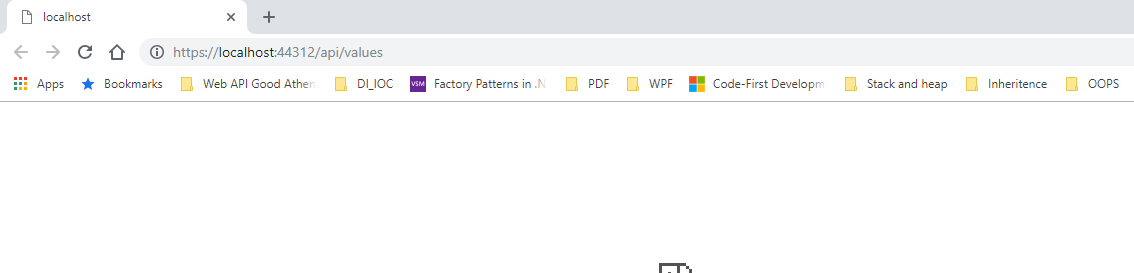
The service could be run via IIS Express i.e. Visual Studio default or via Docker container as well.

## Via IIS Express

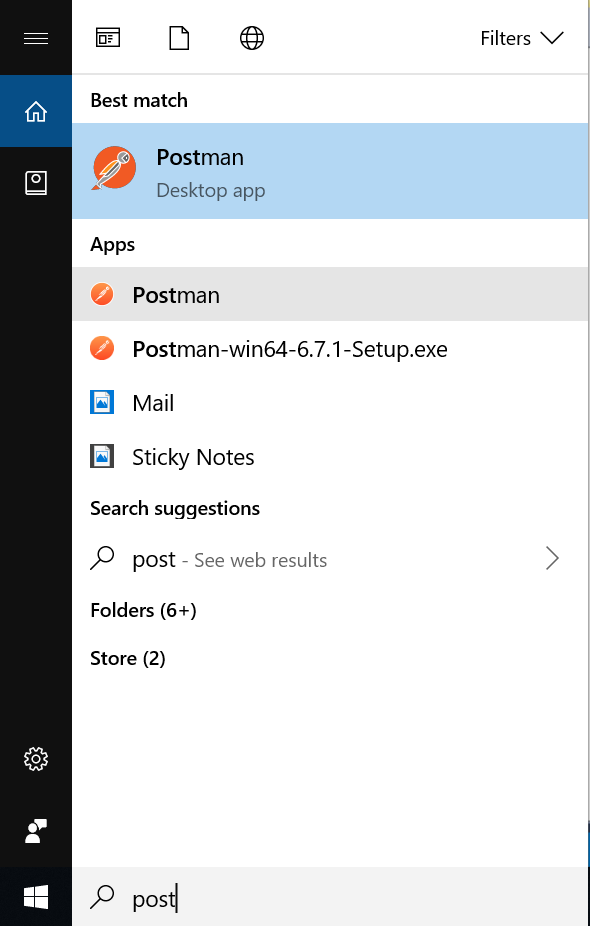
Choose IIS Express in the Visual Studio as shown below and press F5 or click that IIS Express button itself.



The application will be up once the browser page is launched. Since it has nothing to show, it will be blank, but the service could be tested via any API testing client. Here Postman is used to testing the service endpoints. Keep it opened and application running.

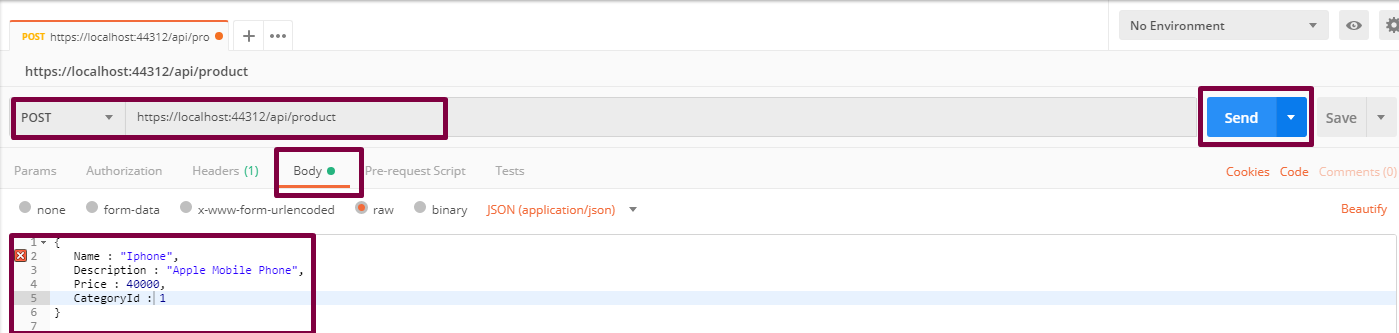


Install Postman if it is not on the machine and launch it.

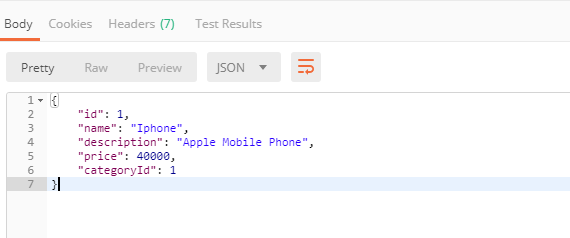


POST

To test the POST method i.e. creating a new resource, select the method as POST in postman and provide the endpoint i.e. https://localhost:44312/api/product and in the Body section, add a json similar to having properties of Product model as shown below and click on Send.

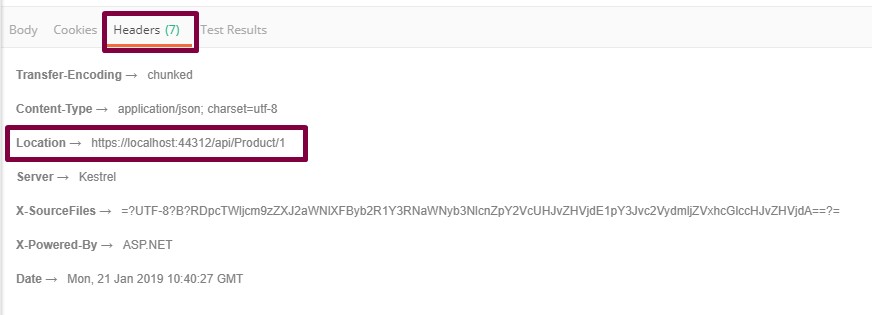


The response is returned with the Id of the product as well.

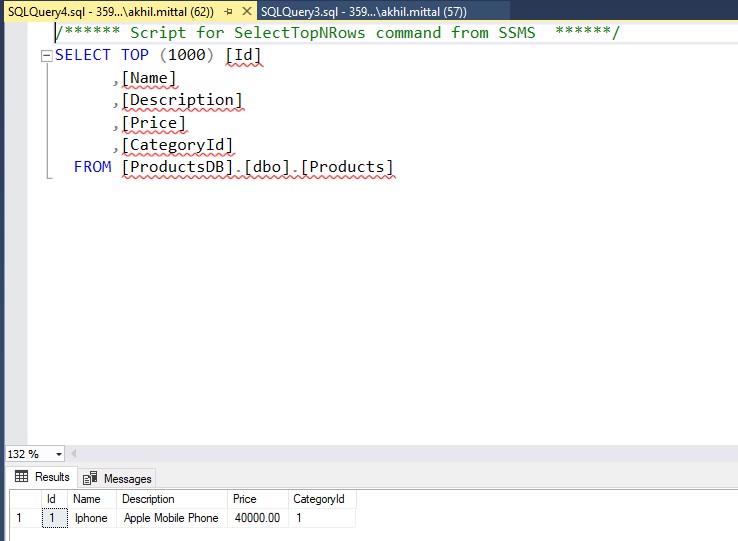


The “Post” method of the controller is responsible to create a resource in the database and send the response.

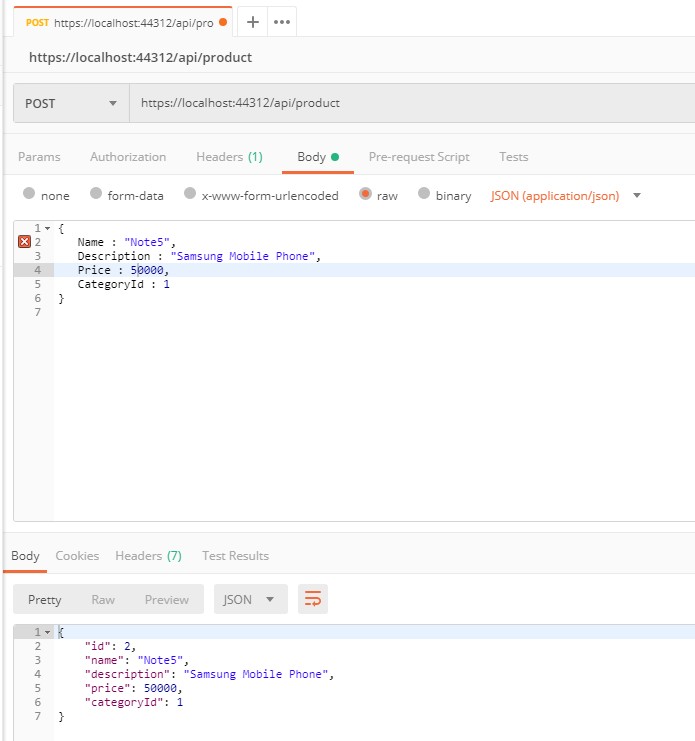
The line return CreatedAtAction(nameof(Get), new { id = product.Id }, product); returns the location of the created resource that could be checked in Location attribute in the response under Headers tab.



Perform a select query on the product table and an added row is shown for the newly created product.

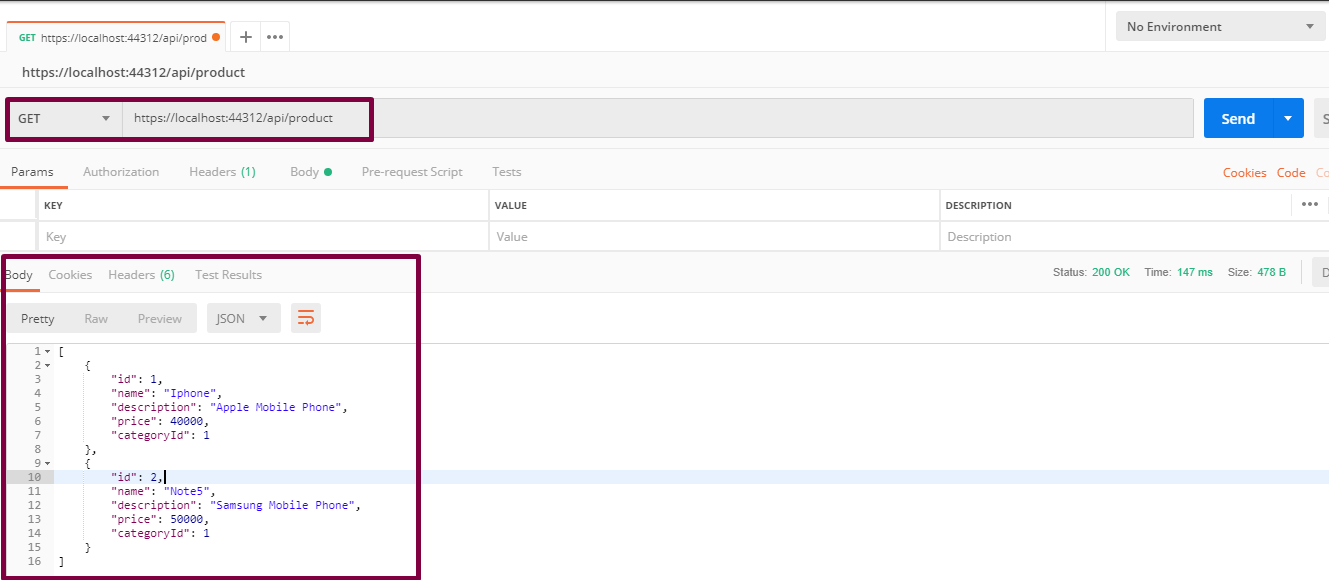


Create one more product in a similar way.



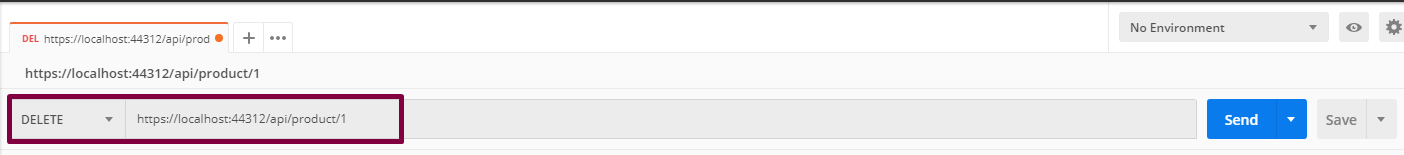
GET

Perform a GET request now with the same address and two records are shown as a JSON result response.

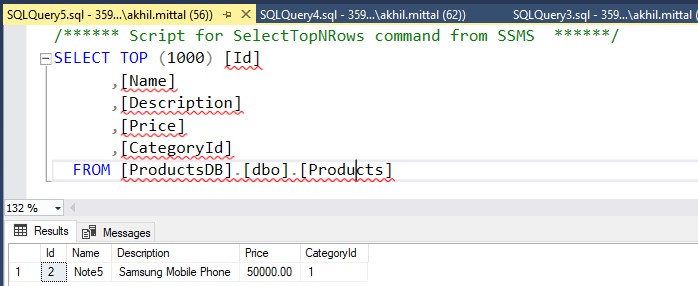


DELETE

Perform the delete request by selecting DELETE as the verb and appending id as 1 (if the product with id 1 needs to be deleted) and press Send.

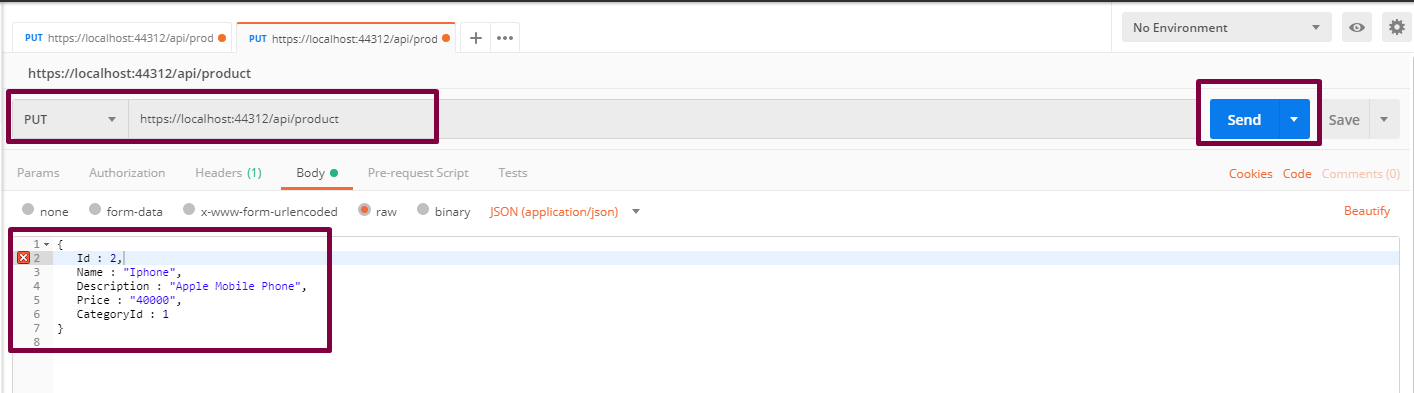


In the database, one record with Id 1 gets deleted.

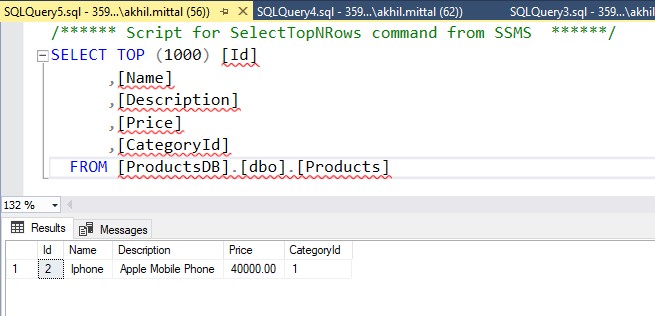


PUT

PUT verb is responsible for updating the resource. Select PUT verb, provide the API address and in the Body section, provide details of which product needs to be updated in JSON format. For e.g. update the product with Id 2 and update its name, description, and price from Samsung to iPhone specific. Press Send.



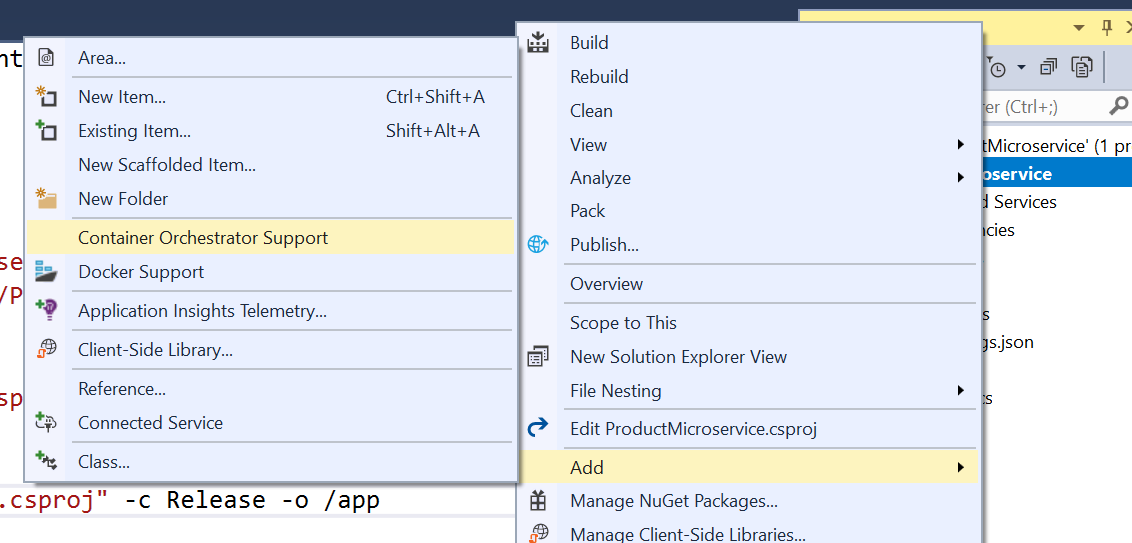
Check the database to see the updated product.



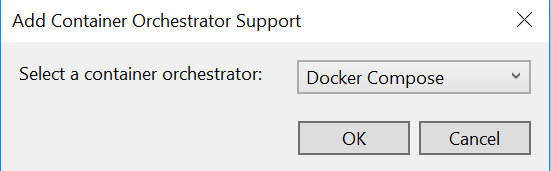
## Via Docker Containers

Running the service could be done via docker commands to be run in docker command prompt and using visual studio as well. Since we added the docker support, it is easy to run the service in docker container using visual studio.

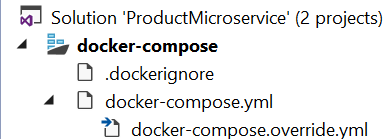
1. Add container orchestrator support in the solution as shown below.



1. This will ask for the orchestrator. Select Docker Compose and press OK.

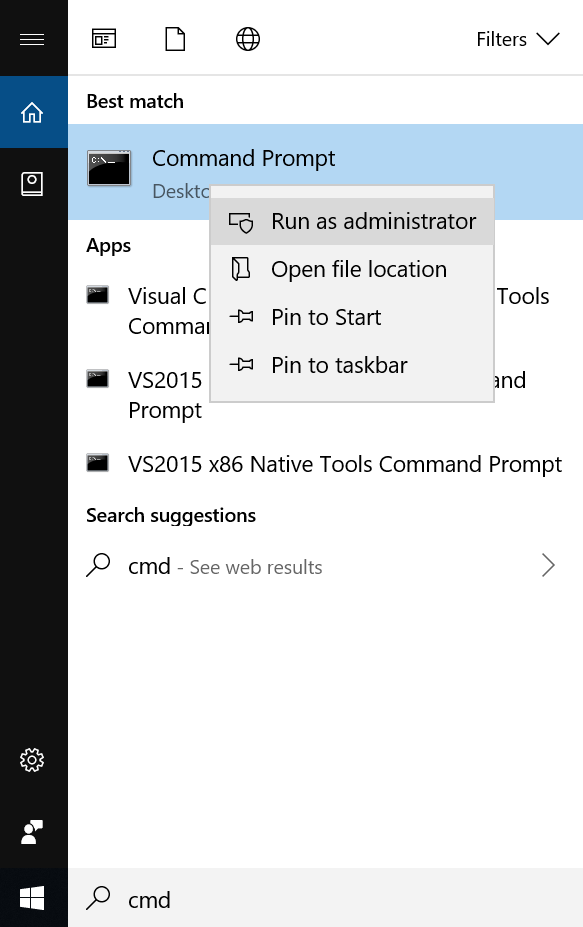


Once added to the solution, the solution will look like shown below having docker-compose with dockerignore and docker-compose.yml and its override file.

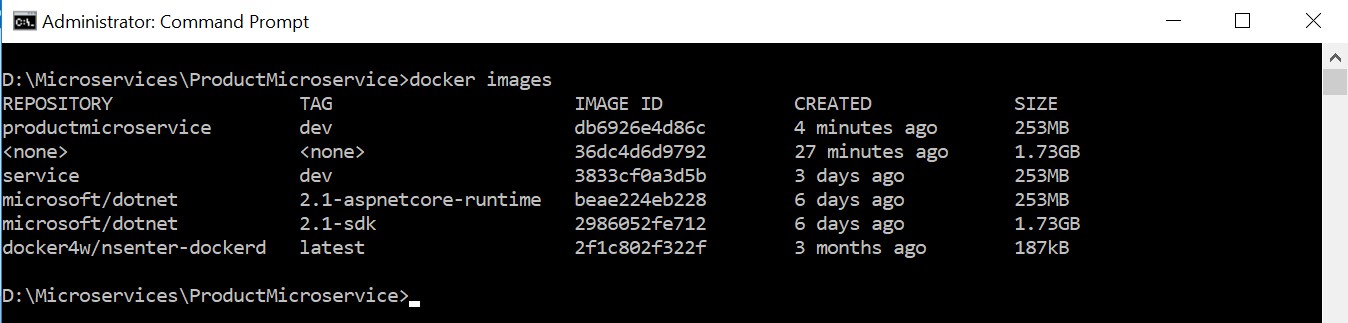


As soon as the solution is saved, it builds the project under the container and creates a docker image. All the commands execution can be seen in the output window when the solution is saved.

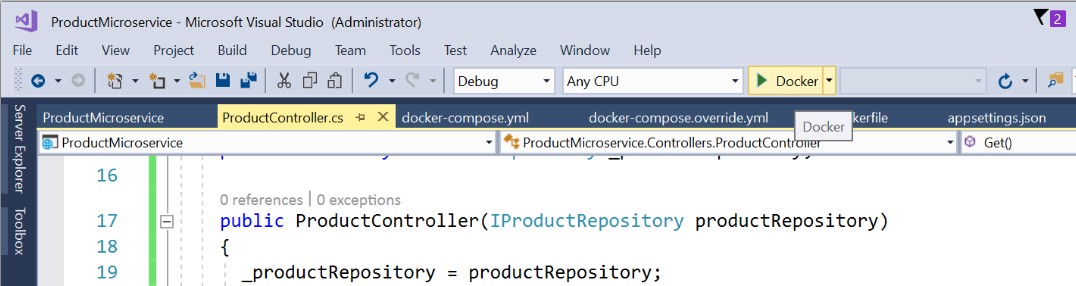
1. Open the command prompt in admin mode and navigate to the same folder where the project files are.



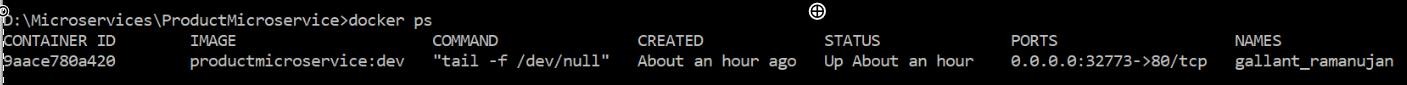
1. Run the command **docker images** to see all the created images. We see the productmicroserviceimage the latest one.



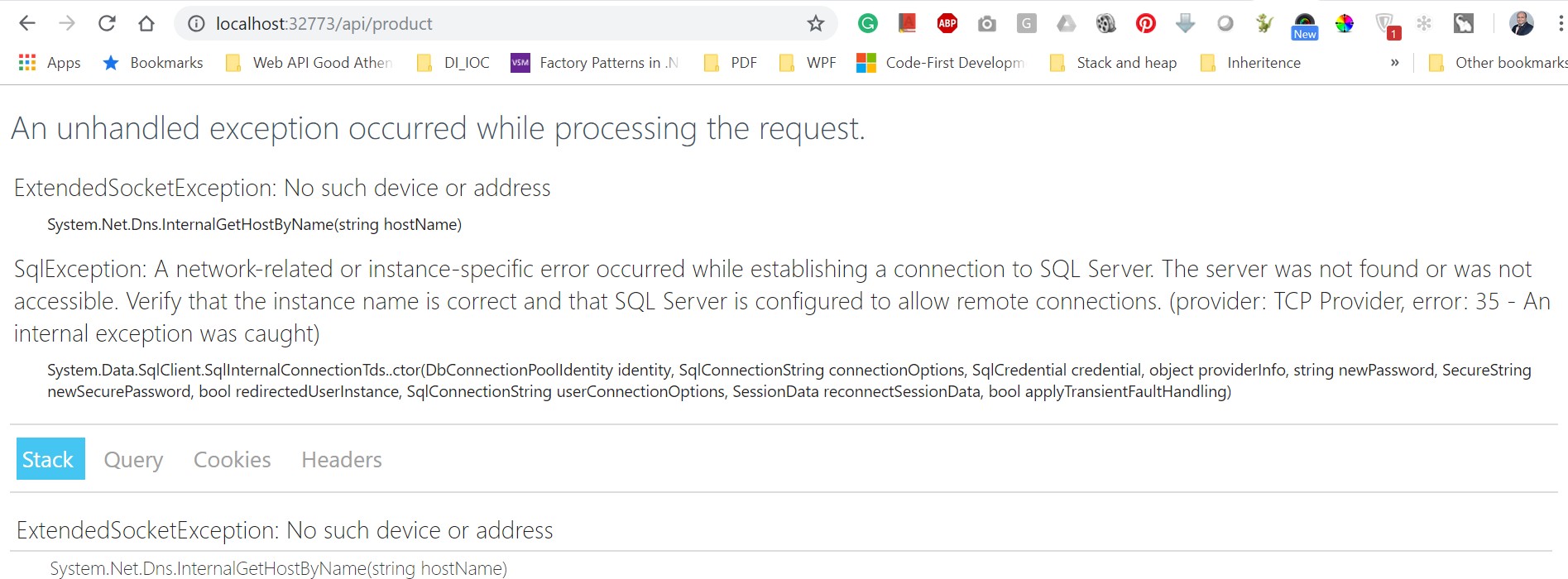
1. Now run the application with Docker as an option as shown below.



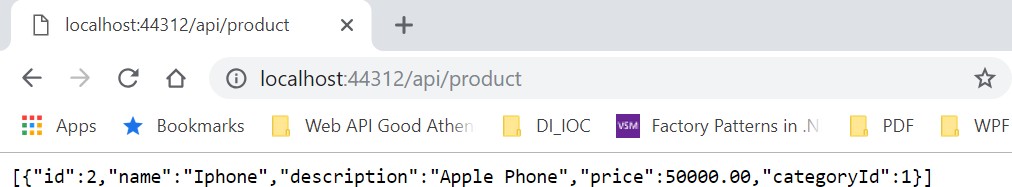
1. Now, run the command **docker ps** to see the running containers. It shows the container is running on 32773:80 port.



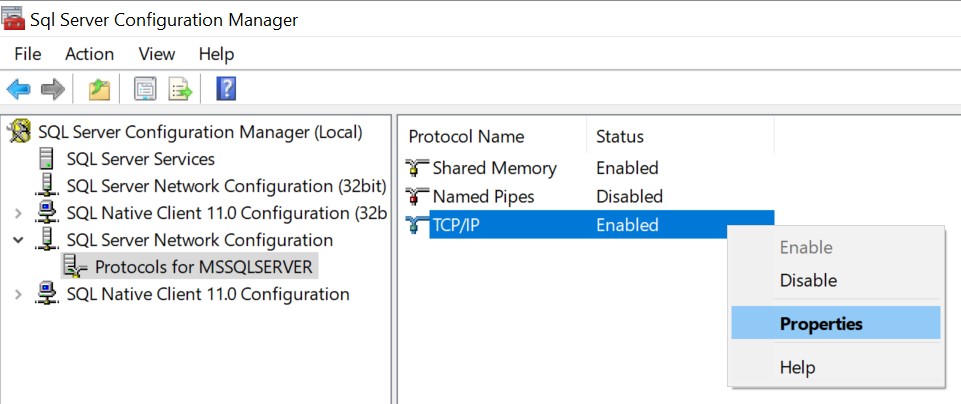
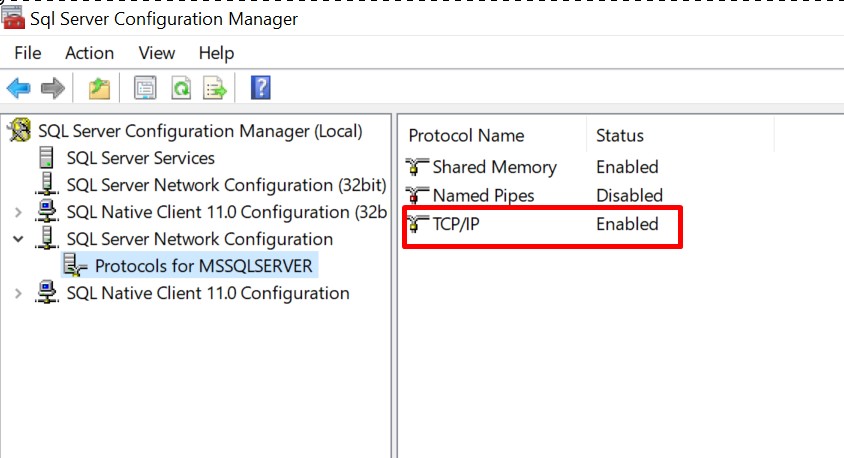
1. Since the container is in running state, it is good to test the service now running under the container. To test the service, replace ”values” with “product” in the address as shown below. Ideally, it should get the product details. But it gives exception as shown below.

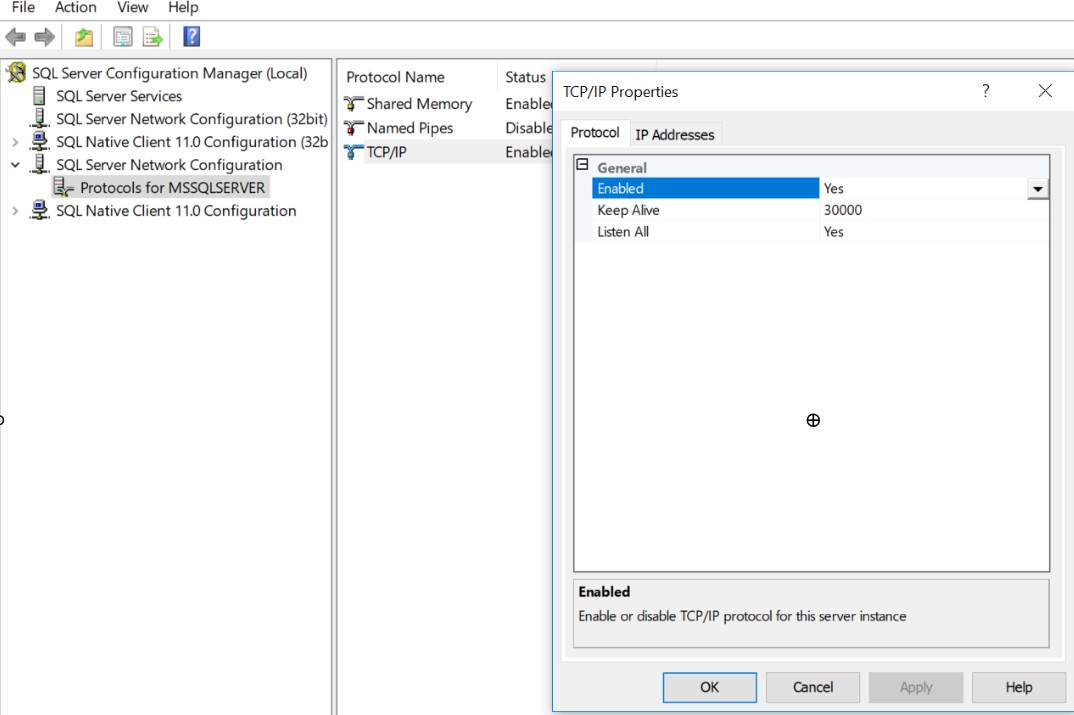


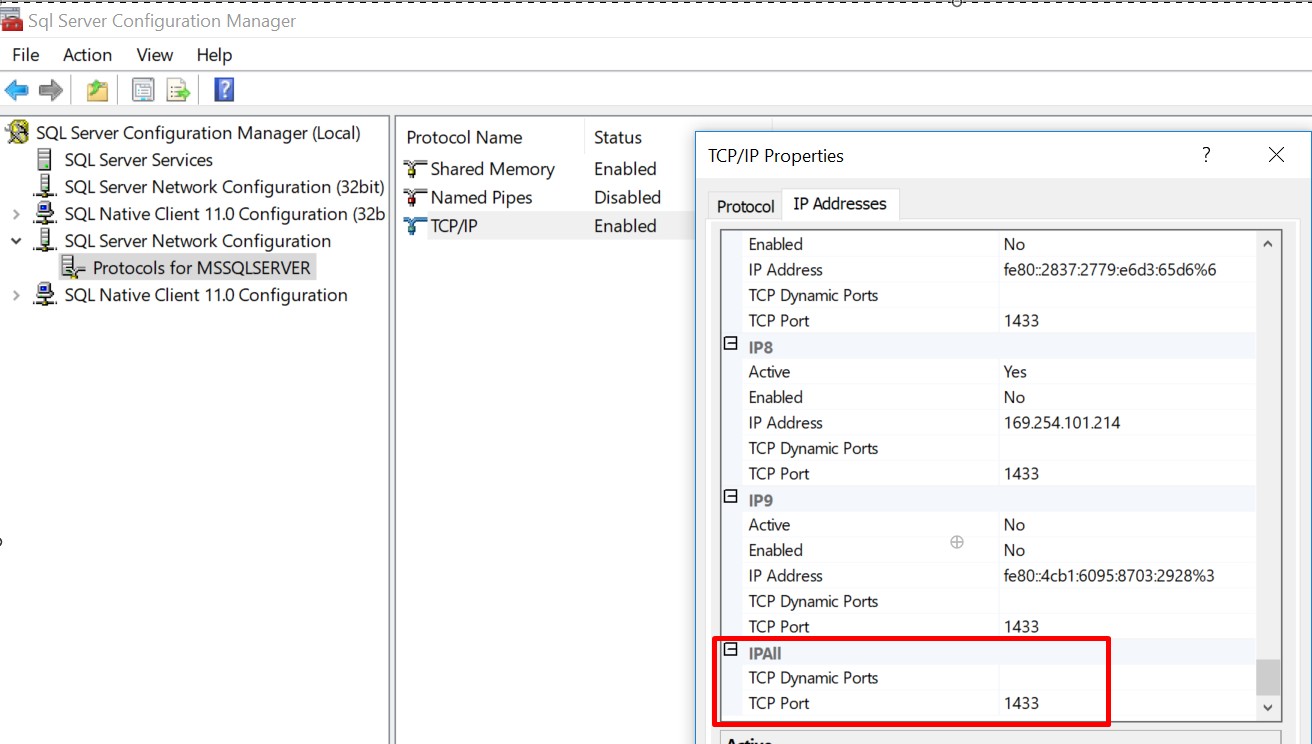
1. Running the same thing under IIS Express works fine i.e. on port 44312. Replace “values” with the product to get the product details,



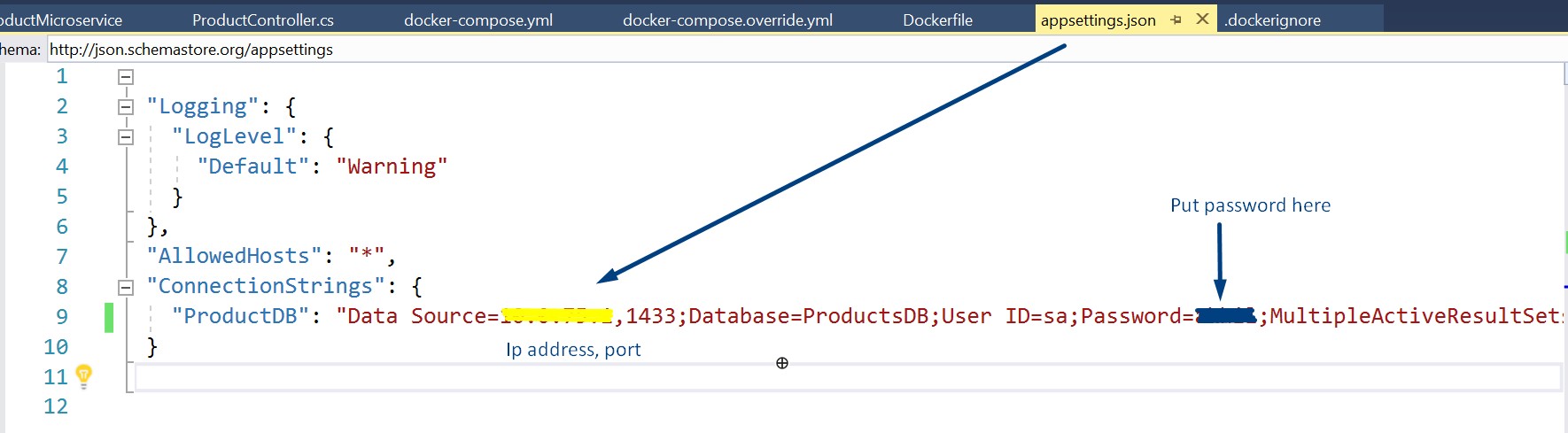
1. Since in IIS Express application runs fine and not in docker container and the error clearly shows that something is wrong with the SQL server that it does not understands our docker container or it is not running under docker container. In this scenario, the docker container is running as a separate machine inside the host computer. So, to connect to the SQL database in the host machine, remote connections to SQL needs to be enabled. We can fix this.
2. Open the SQL Server Configuration Manager. Now select Protocols for MSSQLSERVER and get the IPAll port number under TCP/IP section.



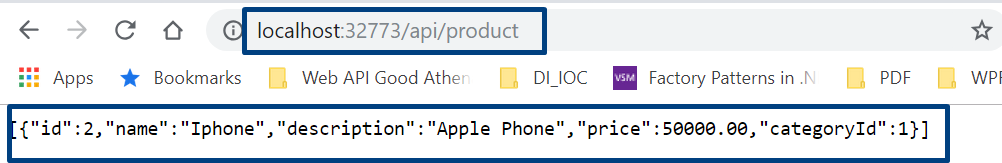




1. The connection string mentioned in the appsettings.json file points to the data source as local which the docker container do not understand. It needs proper IP addresses with port and SQL authentication. So, provide the relevant details i.e. Data Source as Ip address, port number and SQL authentication details as shown below.

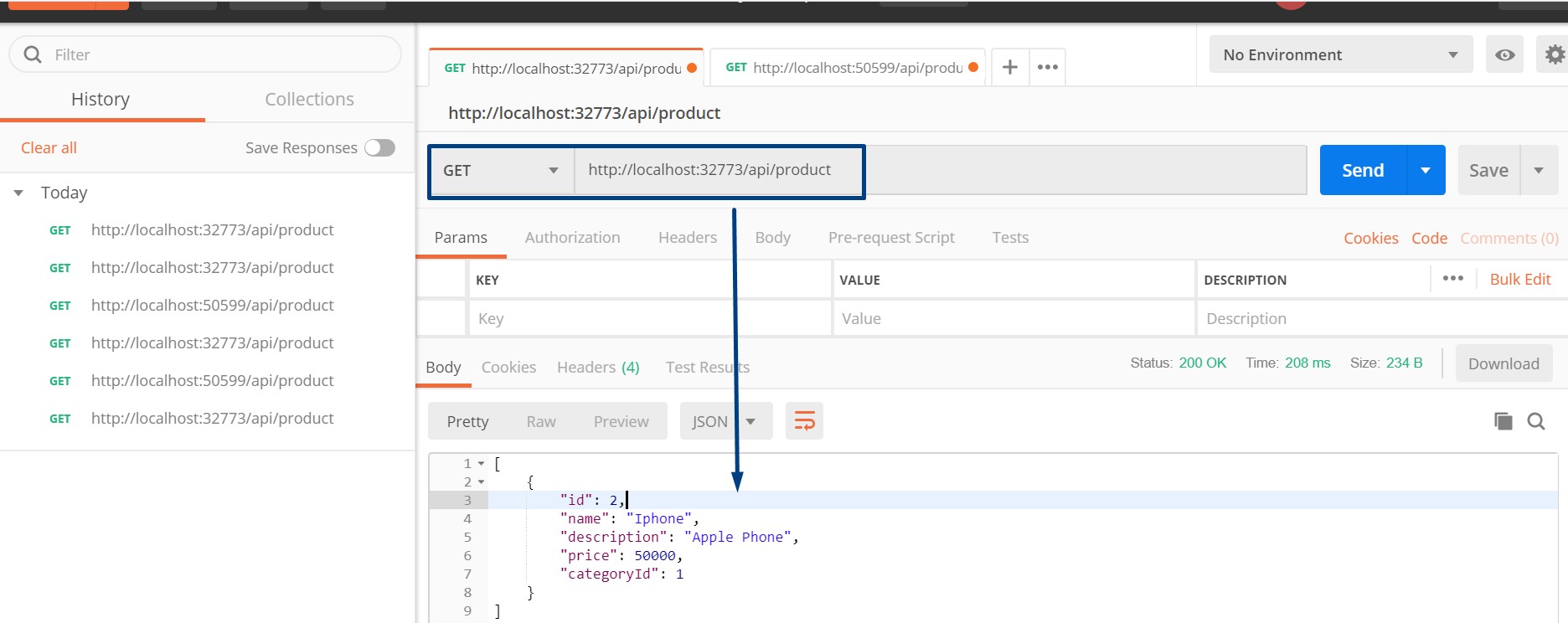


1. Now again run the application with Docker as an option like done earlier.

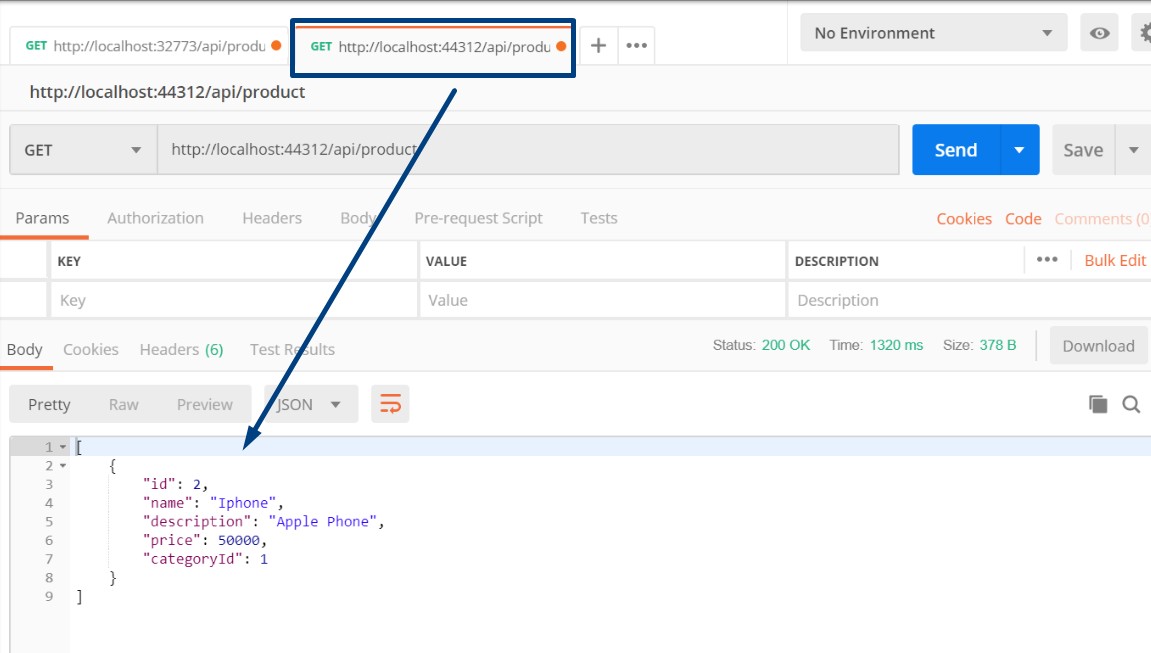


This time the response is received.

1. Test the same in the Postman.



1. Test again with IIS Express URL.



This proves that the microservice is running on two endpoints and on two operating systems independently locally deployed.