



Prerequisite and Tools

Prerequisite	.NET Core 3.1, C#, MongoDB
Tools	Visual Studio 2019, Docker Desktop, Postman

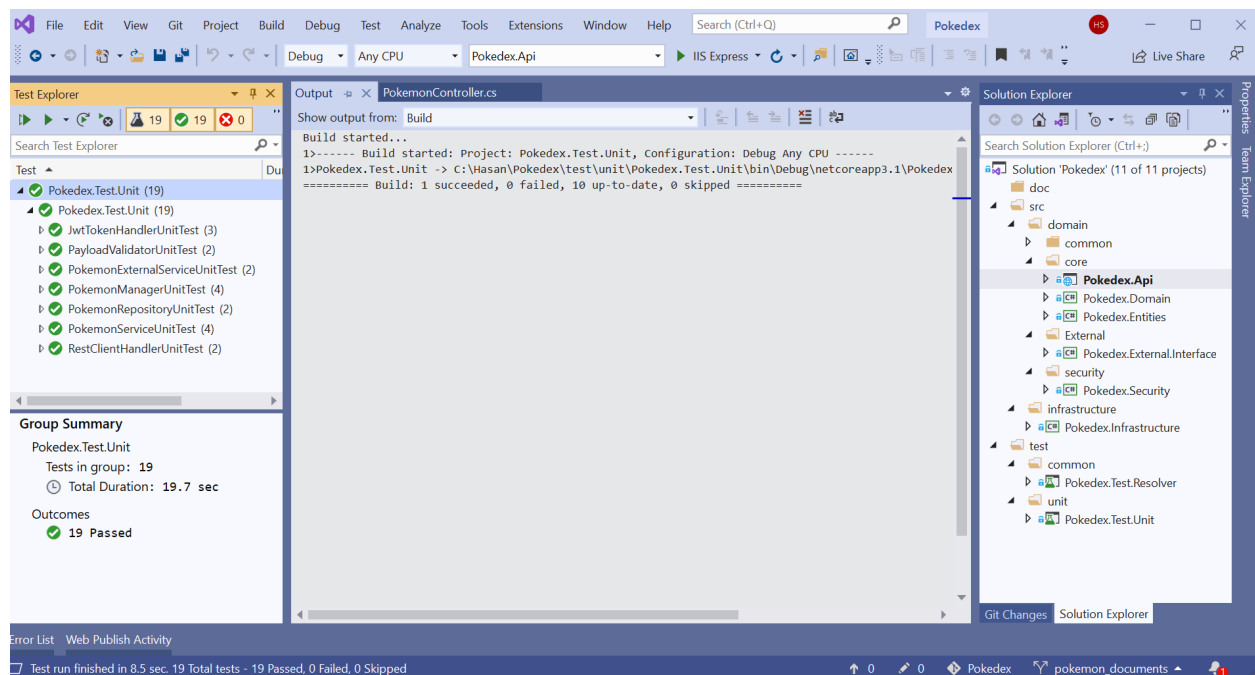
Github Repository

<https://github.com/HasanShahjahan/Pokedex>

Project Structure and Configuration

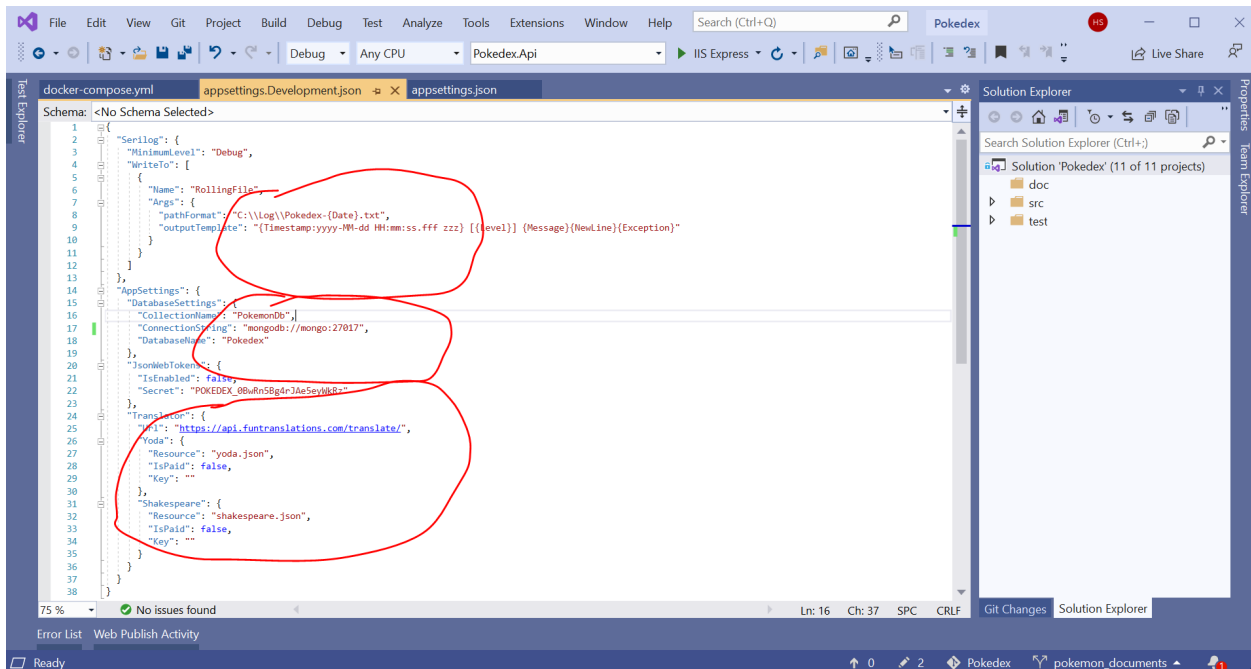
Project Structure

Project has basically three parts doc, src and test. All the documents are available under the doc folder, src includes main project and test section includes unit tests.



Project Configuration

Web Configuration



Serilog

Please specify the **log path** where application log will be saved. It's a **rolling file** and we can set the **log level** too.

Database Settings

Contains MongoDB database credentials including **connection string**, **collection name** and **database name**.

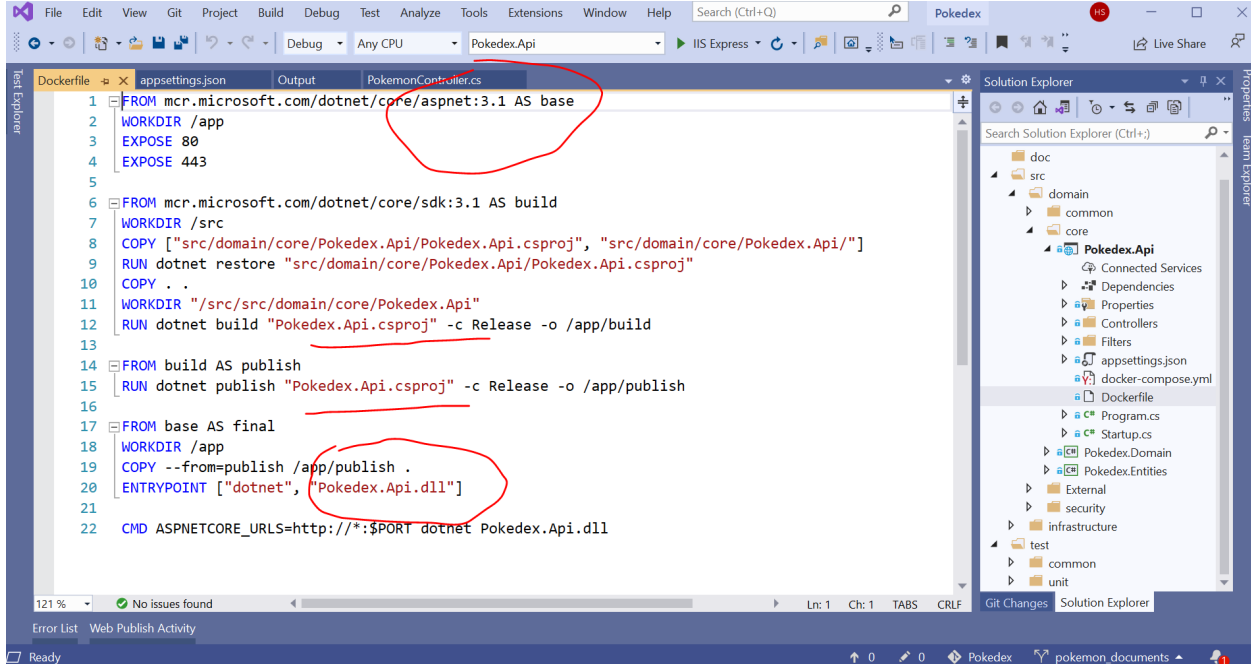
JWT Token

Requests will be authorized by JsonWebToken **secret**. Pokedex api is also configurable with token **enable or disable**. If we don't want to authorize, please specify false it's isEnabled parameter.



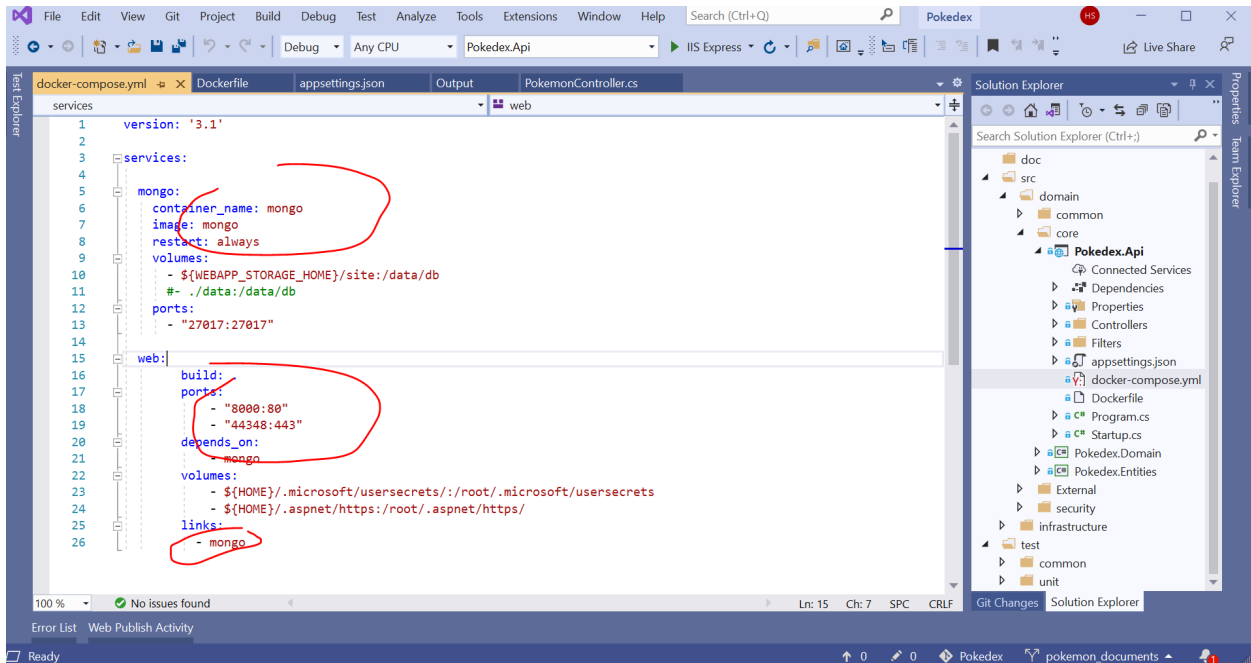
Docker Container Configuration

Docker file - For docker images



```
1 FROM mcr.microsoft.com/dotnet/core/aspnet:3.1 AS base
2 WORKDIR /app
3 EXPOSE 80
4 EXPOSE 443
5
6 FROM mcr.microsoft.com/dotnet/core/sdk:3.1 AS build
7 WORKDIR /src
8 COPY ["src/domain/core/Pokedex.Api/Pokedex.Api.csproj", "src/domain/core/Pokedex.Api/"]
9 RUN dotnet restore "src/domain/core/Pokedex.Api/Pokedex.Api.csproj"
10 COPY .
11 WORKDIR "/src/src/domain/core/Pokedex.Api"
12 RUN dotnet build "Pokedex.Api.csproj" -c Release -o /app/build
13
14 FROM build AS publish
15 RUN dotnet publish "Pokedex.Api.csproj" -c Release -o /app/publish
16
17 FROM base AS final
18 WORKDIR /app
19 COPY --from=publish /app/publish .
20 ENTRYPOINT ["dotnet", "Pokedex.Api.dll"]
21
22 CMD ASPNETCORE_URLS=http://*:$PORT dotnet Pokedex.Api.dll
```

Docker Compose - For mongodb and browser localhost port configuration.



```
1 version: '3.1'
2
3 services:
4   mongo:
5     container_name: mongo
6     image: mongo
7     restart: always
8     volumes:
9       - $(WEBAPP_STORAGE_HOME)/site:/data/db
10      - ./data:/data/db
11     ports:
12       - "27017:27017"
13
14   web:
15     build:
16       context: .
17     ports:
18       - "8000:80"
19       - "44348:443"
20     depends_on:
21       - mongo
22     volumes:
23       - $(HOME)/.microsoft/usersecrets:/root/.microsoft/usersecrets
24       - $(HOME)/.aspnet/https:/root/.aspnet/https/
25     links:
26       - mongo
```

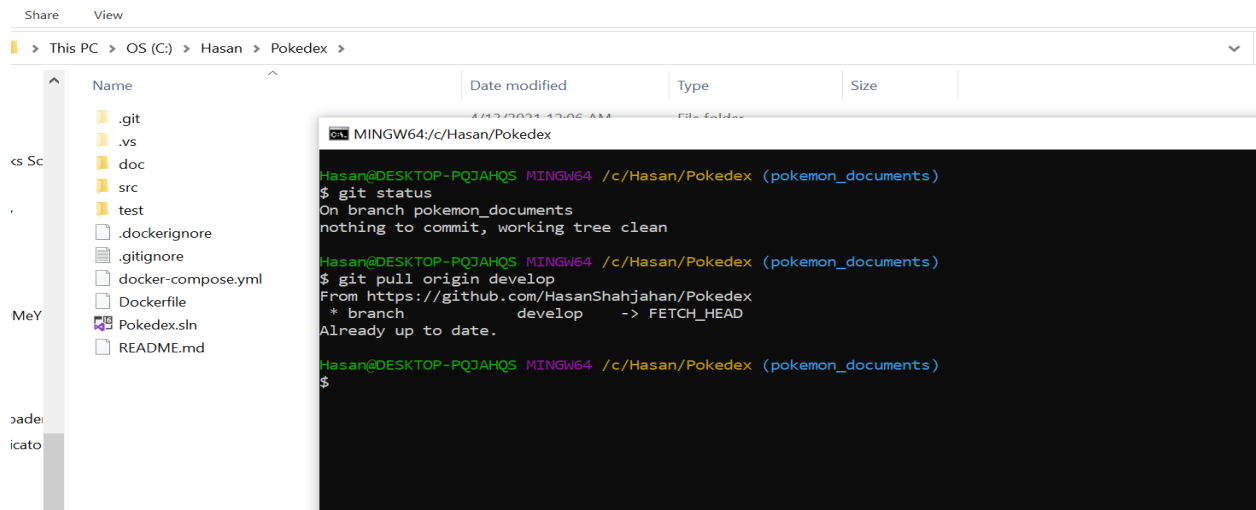


Deployment

Local Deployable Docker container

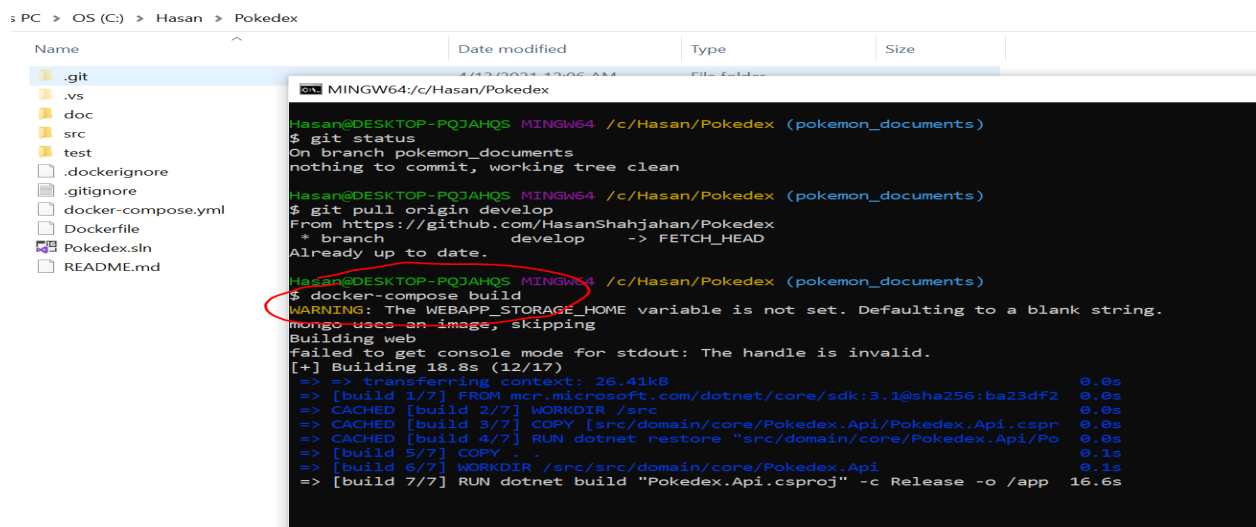
Step 1

Download or clone project from <https://github.com/HasanShahjahan/Pokedex>



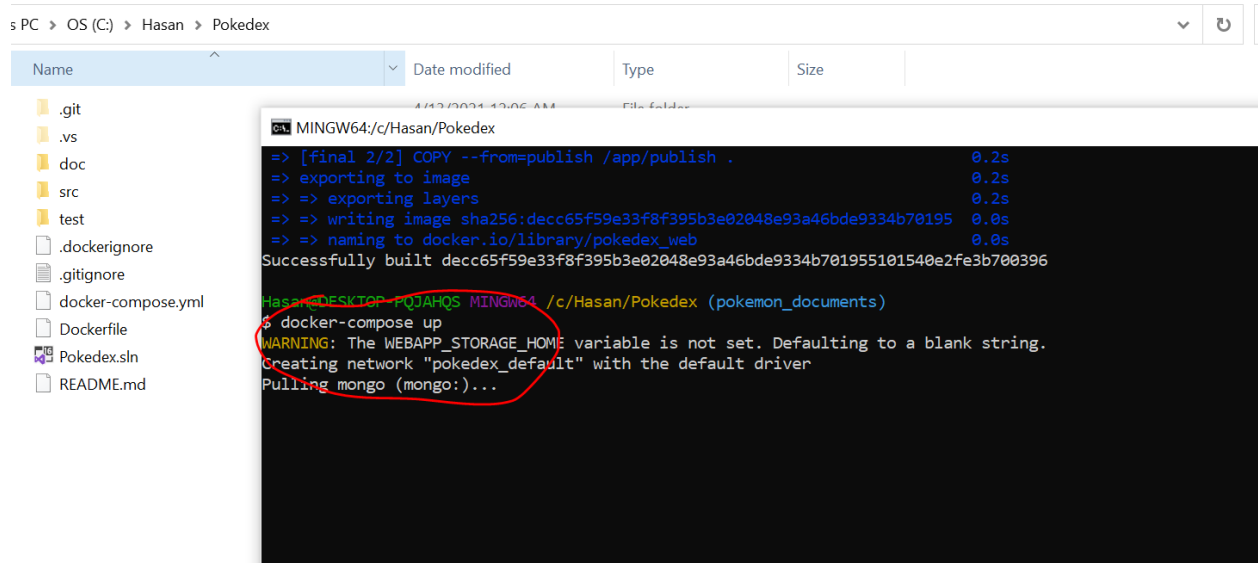
Step 2

Write below command and enter -
docker-compose build



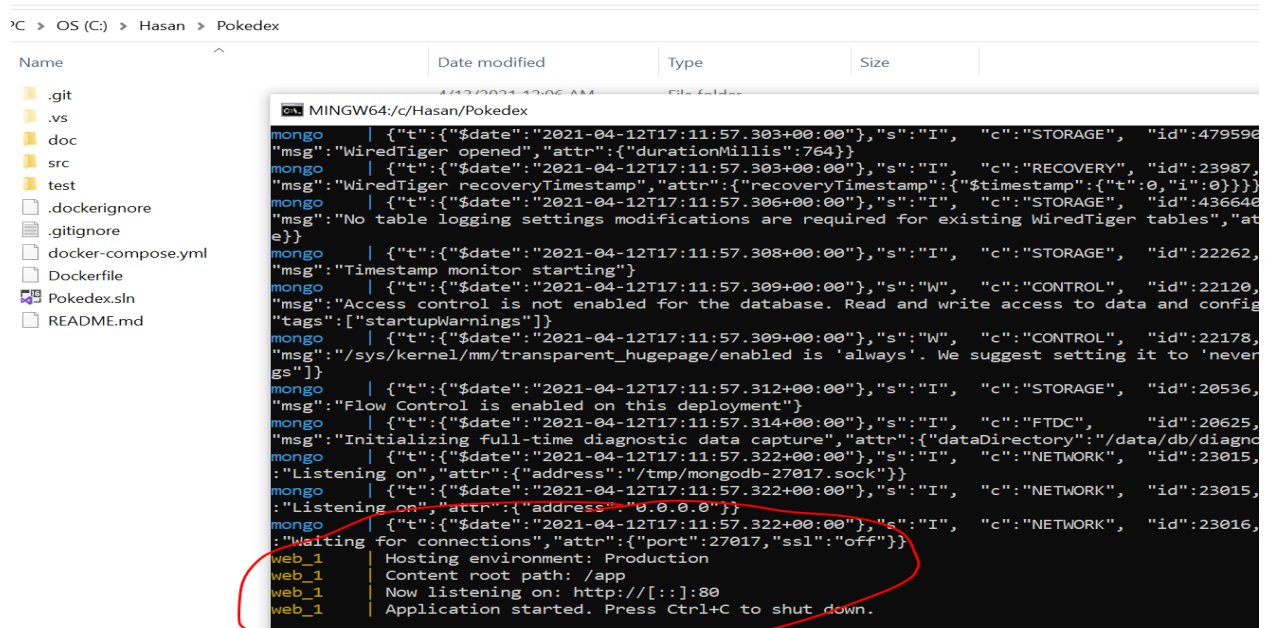
Step 3

Write below command and enter -
docker-compose up



The screenshot shows a Windows File Explorer window with the path 'C:\OS (C:) > Hasan > Pokedex'. The file list includes .git, .vs, doc, src, test, .dockerignore, .gitignore, docker-compose.yml, Dockerfile, Pokedex.sln, and README.md. Overlaid on the right is a terminal window titled 'MINGW64/c/Hasan/Pokedex'. The terminal output shows the successful build of the 'pokedex_web' image and the execution of 'docker-compose up'. A red circle highlights the command '\$ docker-compose up' and the subsequent warning message: 'WARNING: The WEBAPP_STORAGE_HOME variable is not set. Defaulting to a blank string. Creating network "pokedex_default" with the default driver Pulling mongo (mongo:)...'.

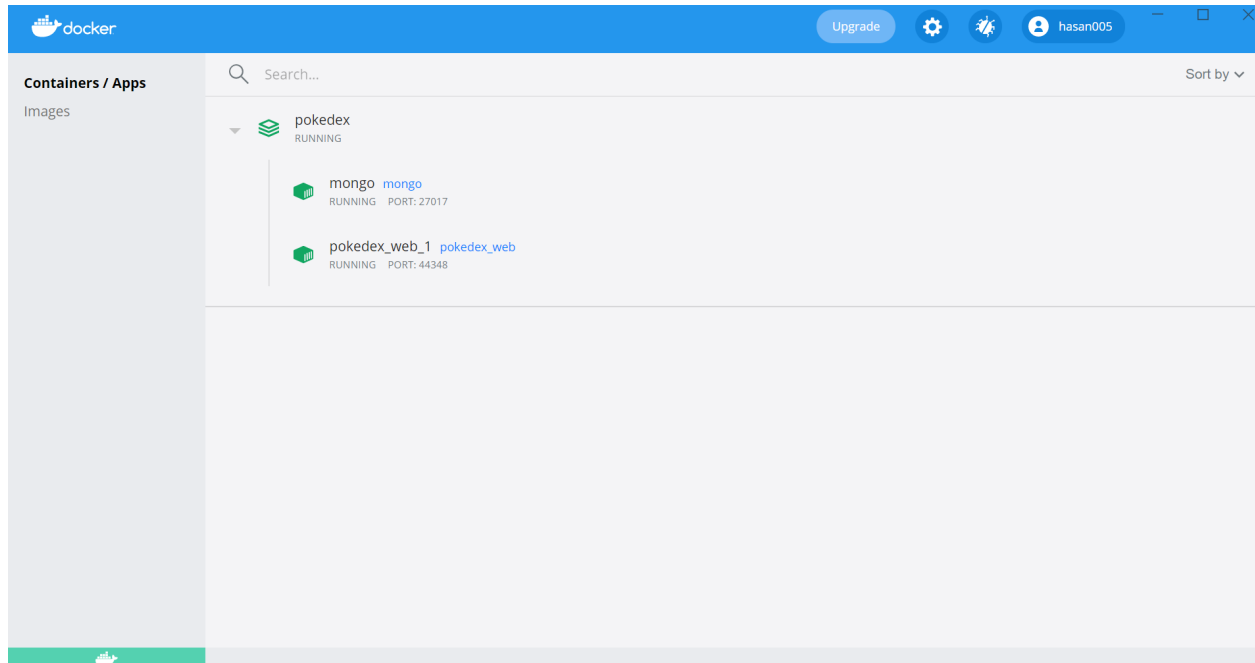
We are done.



The screenshot shows the same Windows File Explorer window as before. The terminal window now displays the logs for the 'web_1' container. A red circle highlights the final lines of the logs: 'web_1 | Hosting environment: Production', 'web_1 | Content root path: /app', 'web_1 | Now listening on: http://[::]:80', and 'web_1 | Application started. Press Ctrl+C to shut down.'.



Our application will be running our local docker container.



Step 4

Congratulations !!!

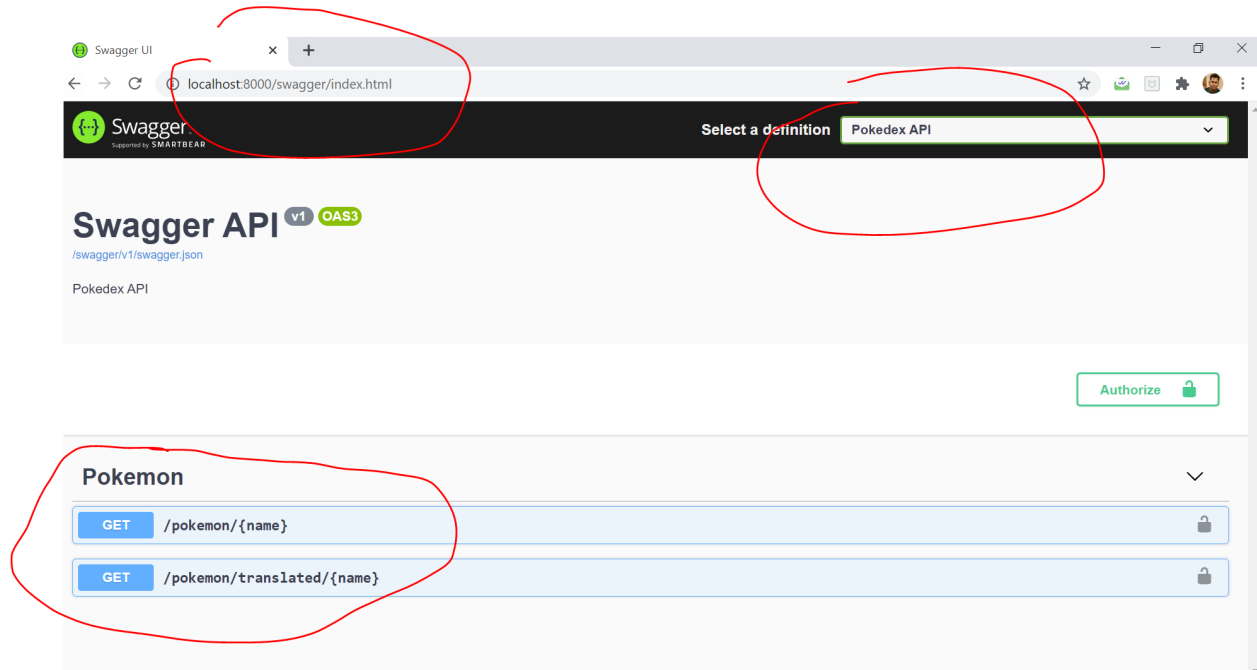
As we have seen our port configuration inside **docker-compose.yml**, *let's try through the browser*. Pokedex service also has implemented swagger, let's try through (We can also try by postman)

Platform	localhost
Publish Type	Docker Container
Local URL	http://localhost:8000/swagger/index.html
Api Authorization Token Optional	Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VyX2lkZW50aXR5IjoiaSGFzYW4iLCJuYmYiOiJlE2MTc5MDU4NjYsImV4cCI6MTYyMDQ5Nzg2NiwiYWV0IjoxNjE3OTA1ODY2fQ.o2rTundIHpSacmWA8hR130GGHtSWH9ufWbURBnSJ6G8

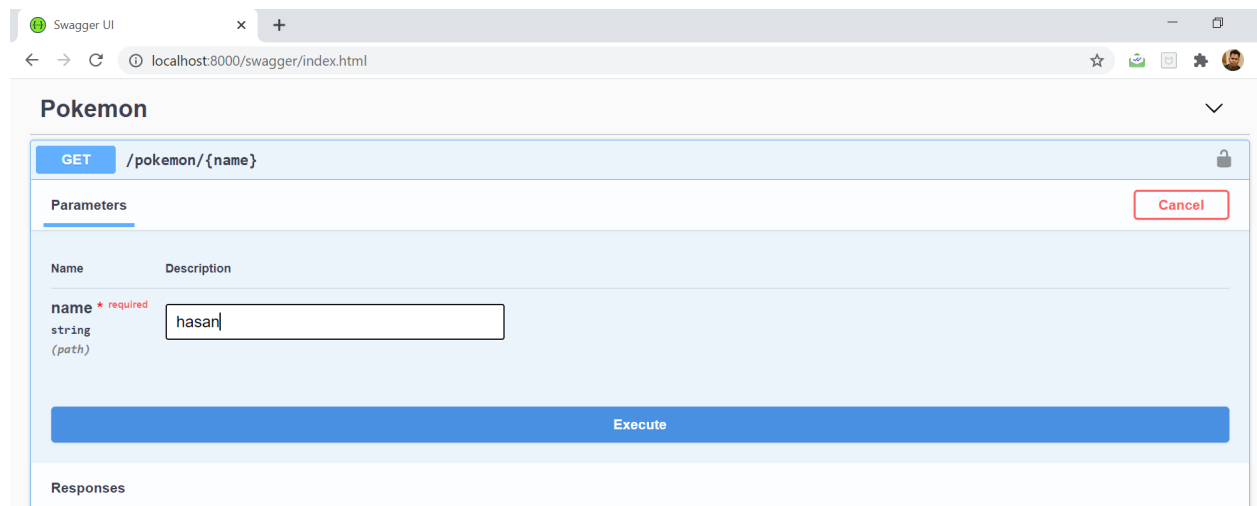


Let's try -

Open browser and paste <http://localhost:8000/swagger/index.html>



Request





Response

Swagger UI

localhost:8000/swagger/index.html

Responses

Curl

```
curl -X GET "http://localhost:8000/pokemon/hasan" -H "accept: */*"
```

Request URL

```
http://localhost:8000/pokemon/hasan
```

Server response

Code	Details
200	<p>Response body</p> <pre>{ "name": "Hasan", "description": "It was created by a scientist after years of horrific genesplicing and DNA engineering experiments", "habitat": "rare", "islegendary": false }</pre> <p>Response headers</p> <pre>content-type: application/json; charset=utf-8 date: Mon, 12 Apr 2021 17:31:22 GMT server: Kestrel transfer-encoding: chunked</pre>

Responses

Code	Description	Links
------	-------------	-------

It is just a beginning, Cheers !!!