Prerequisite and Tools

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

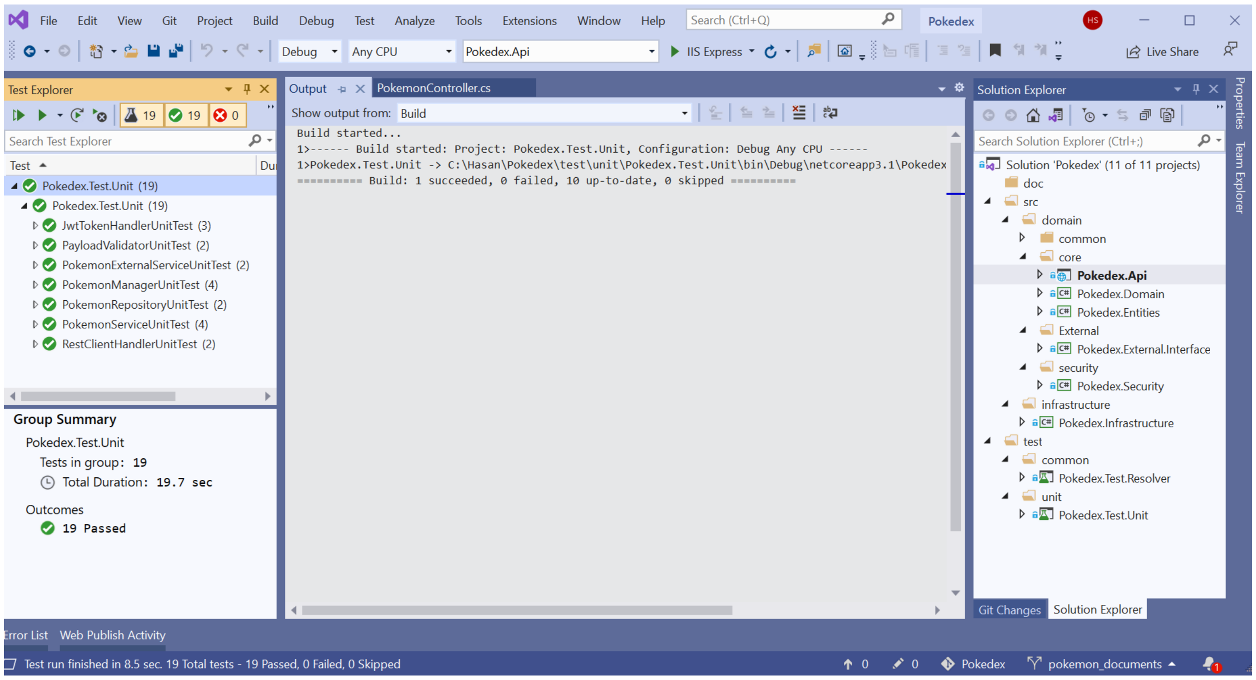
Github Repository

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.

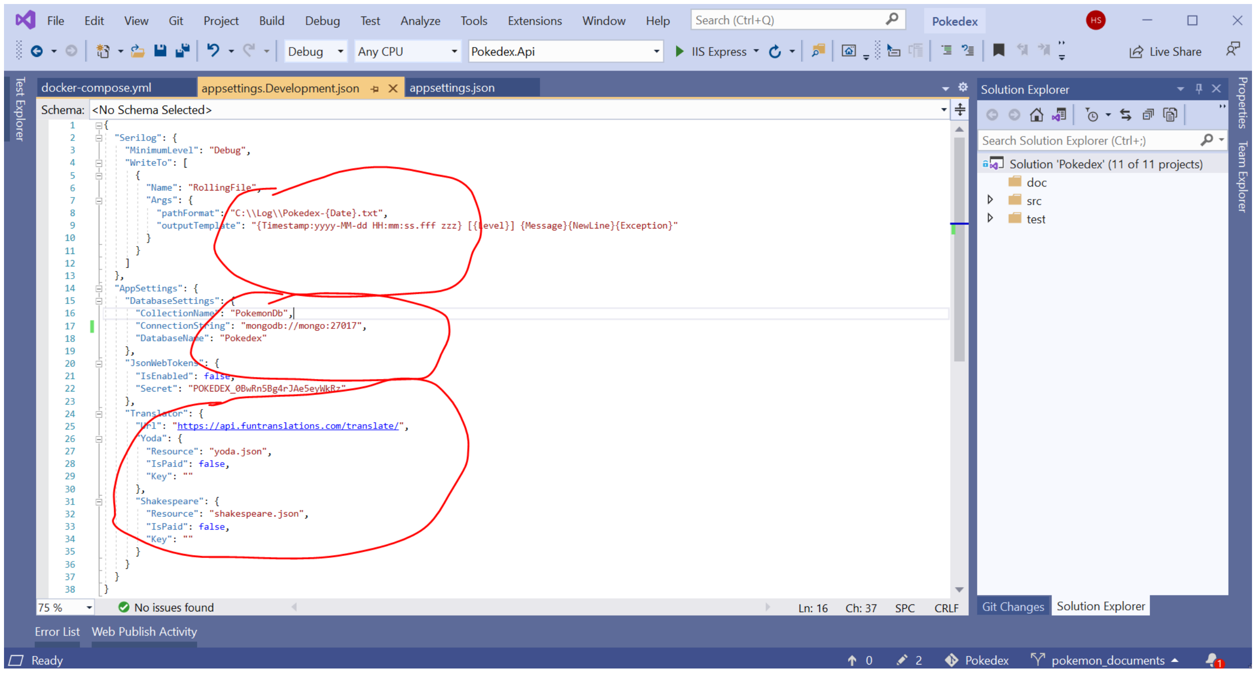


```
Build started...
1>------ Build started: Project: Pokedex.Test.Unit, Configuration: Debug Any CPU ------
1>Pokedex.Test.Unit -> C:\Hasan\Pokedex\test\unit\Pokedex.Test.Unit\bin\Debug\netcoreapp3.1\Pokedex
========== Build: 1 succeeded, 0 failed, 10 up-to-date, 0 skipped ==========
```

Test run finished in 8.5 sec. 19 Total tests - 19 Passed, 0 Failed, 0 Skipped

Project Configuration

Web Configuration



```json
{
  "Serilog": {
    "MinimumLevel": "Debug",
    "WriteTo": [
      {
        "Name": "RollingFile",
        "Args": {
          "pathFormat": "C:\\Log\\Pokedex-{Date}.txt",
          "outputTemplate": "{Timestamp:yyyy-MM-dd HH:mm:ss.fff zzz} [{Level}] {Message}{NewLine}{Exception}"
        }
      }
    ]
  },
  "AppSettings": {
    "DatabaseSettings": {
      "CollectionName": "PokemonDb",
      "ConnectionString": "mongodb://mongo:27017",
      "DatabaseName": "Pokedex"
    },
    "JsonWebTokens": {
      "IsEnabled": false,
      "Secret": "POKEDEX_0BwRn5Bg4rJAe5eyWkRz"
    },
    "Translator": {
      "Url": "https://api.funtranslations.com/translate/",
      "Yoda": {
        "Resource": "yoda.json",
        "IsPaid": false,
        "Key": ""
      },
      "Shakespeare": {
        "Resource": "shakespeare.json",
        "IsPaid": false,
        "Key": ""
      }
    }
  }
}
```

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

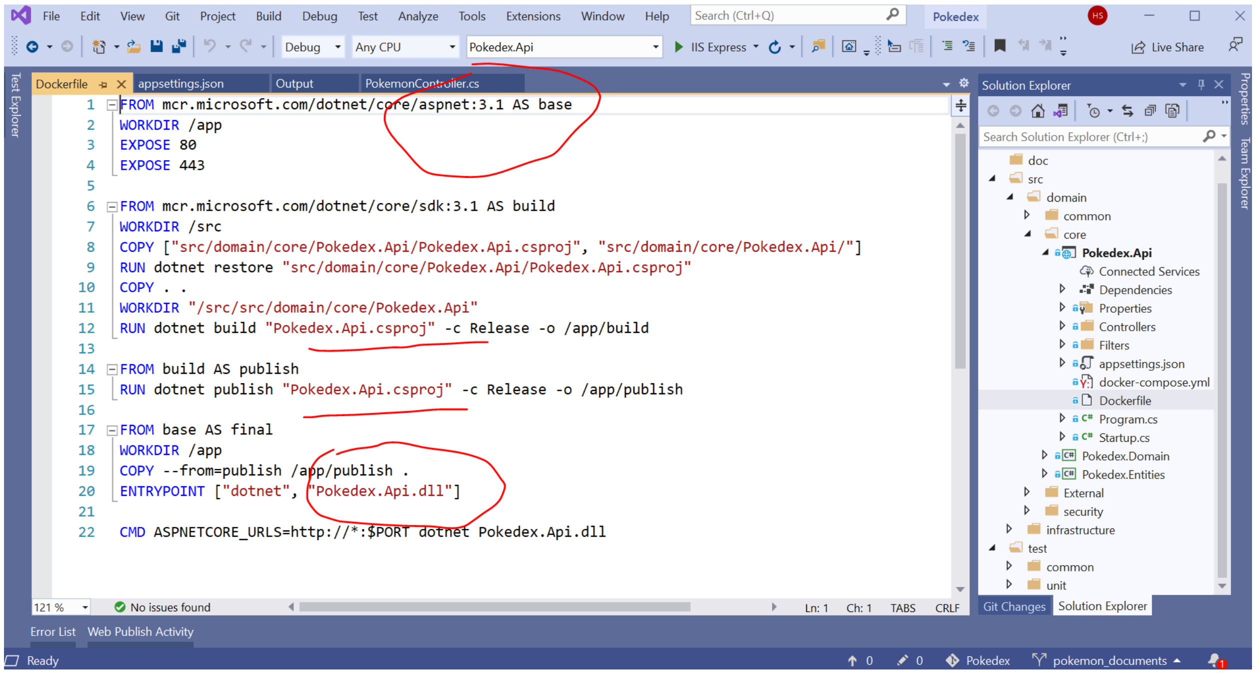
Contains MongoDB database credentials including ***connection string, collection name and database name***.

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._

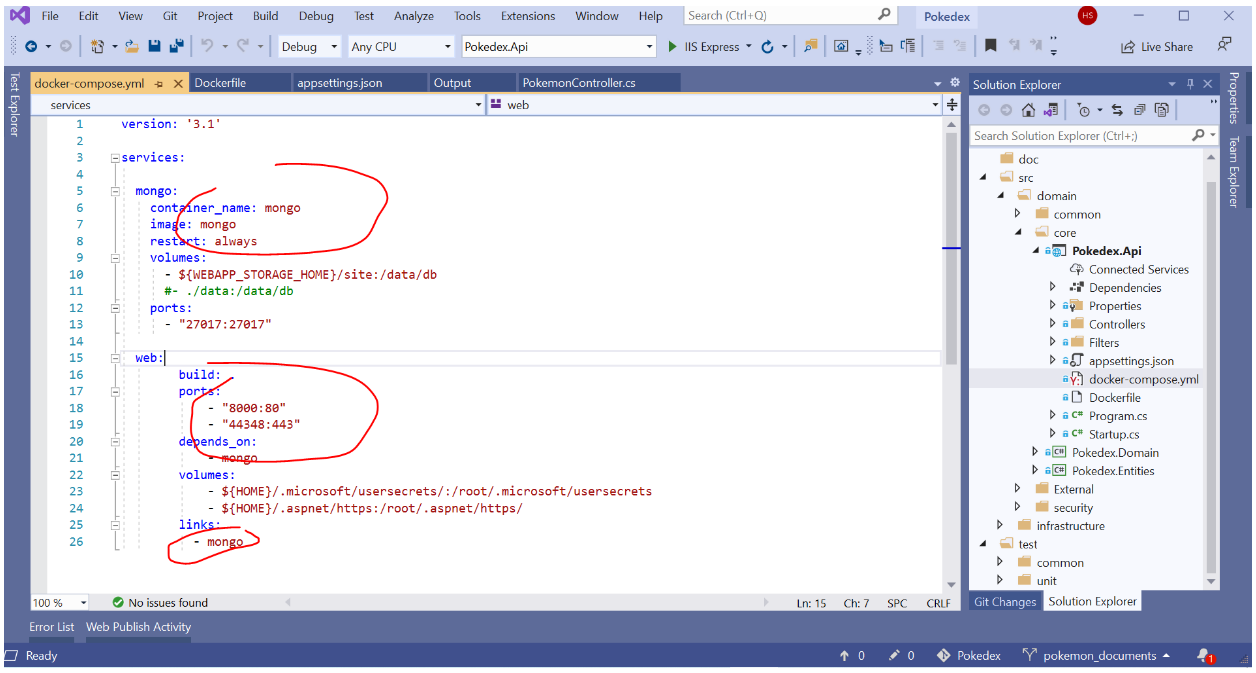
Docker Container Configuration

Docker file - For docker images



```
FROM mcr.microsoft.com/dotnet/core/aspnet:3.1 AS base
WORKDIR /app
EXPOSE 80
EXPOSE 443

FROM mcr.microsoft.com/dotnet/core/sdk:3.1 AS build
WORKDIR /src
COPY ["src/domain/core/Pokedex.Api/Pokedex.Api.csproj", "src/domain/core/Pokedex.Api/"]
RUN dotnet restore "src/domain/core/Pokedex.Api/Pokedex.Api.csproj"
COPY . .
WORKDIR "/src/src/domain/core/Pokedex.Api"
RUN dotnet build "Pokedex.Api.csproj" -c Release -o /app/build

FROM build AS publish
RUN dotnet publish "Pokedex.Api.csproj" -c Release -o /app/publish

FROM base AS final
WORKDIR /app
COPY --from=publish /app/publish .
ENTRYPOINT ["dotnet", "Pokedex.Api.dll"]

CMD ASPNETCORE_URLS=http://*:$PORT dotnet Pokedex.Api.dll
```

Docker Compose - For mongodb and browser localhost port configuration.



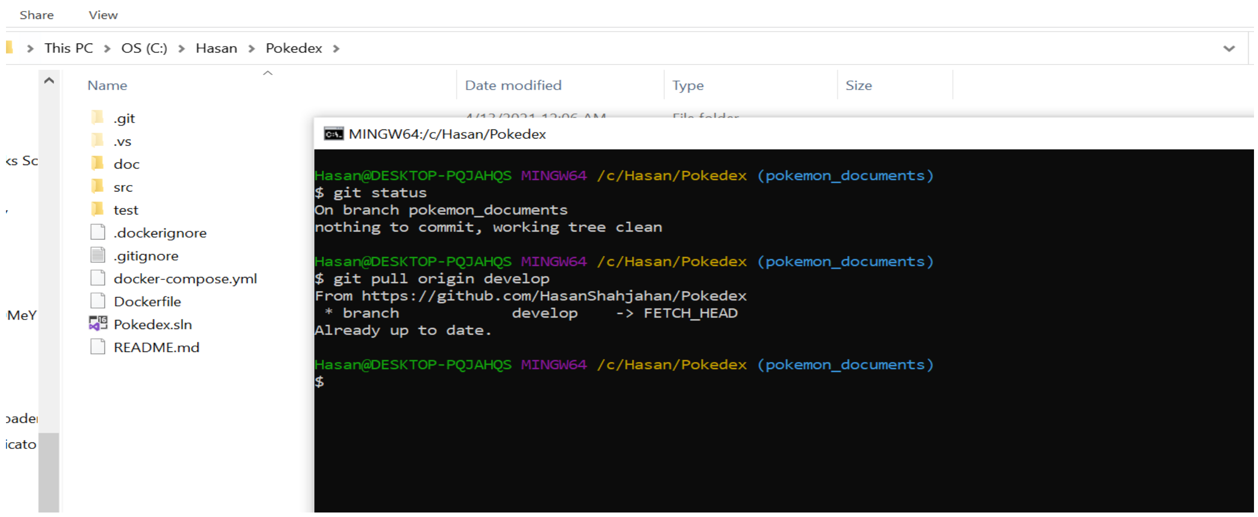
```
 version: '3.1'

services:

  mongo:
    container_name: mongo
    image: mongo
    restart: always
    volumes:
      - ${WEBAPP_STORAGE_HOME}/site:/data/db
      #- ./data:/data/db
    ports:
      - "27017:27017"

  web:
        build: .
        ports:
            - "8000:80"
            - "44348:443"
        depends_on:
            - mongo
        volumes:
            - ${HOME}/.microsoft/usersecrets/:/root/.microsoft/usersecrets
            - ${HOME}/.aspnet/https:/root/.aspnet/https/
        links:
          - mongo
```

Deployment

Local Deployable Docker container

Step 1

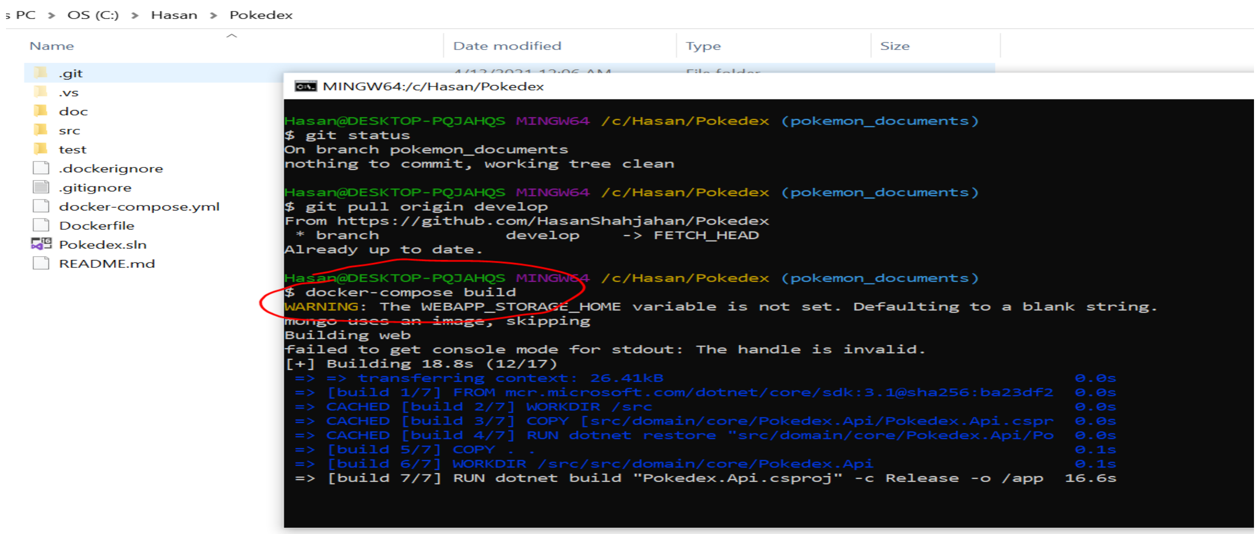
Download or clone project from https://github.com/HasanShahjahan/Pokedex



```
Hasan@DESKTOP-PQJAHQS MINGW64 /c/Hasan/Pokedex (pokemon_documents)
$ git status
On branch pokemon_documents
nothing to commit, working tree clean

Hasan@DESKTOP-PQJAHQS MINGW64 /c/Hasan/Pokedex (pokemon_documents)
$ git pull origin develop
From https://github.com/HasanShahjahan/Pokedex
 * branch            develop    -> FETCH_HEAD
Already up to date.

Hasan@DESKTOP-PQJAHQS MINGW64 /c/Hasan/Pokedex (pokemon_documents)
$
```

Step 2

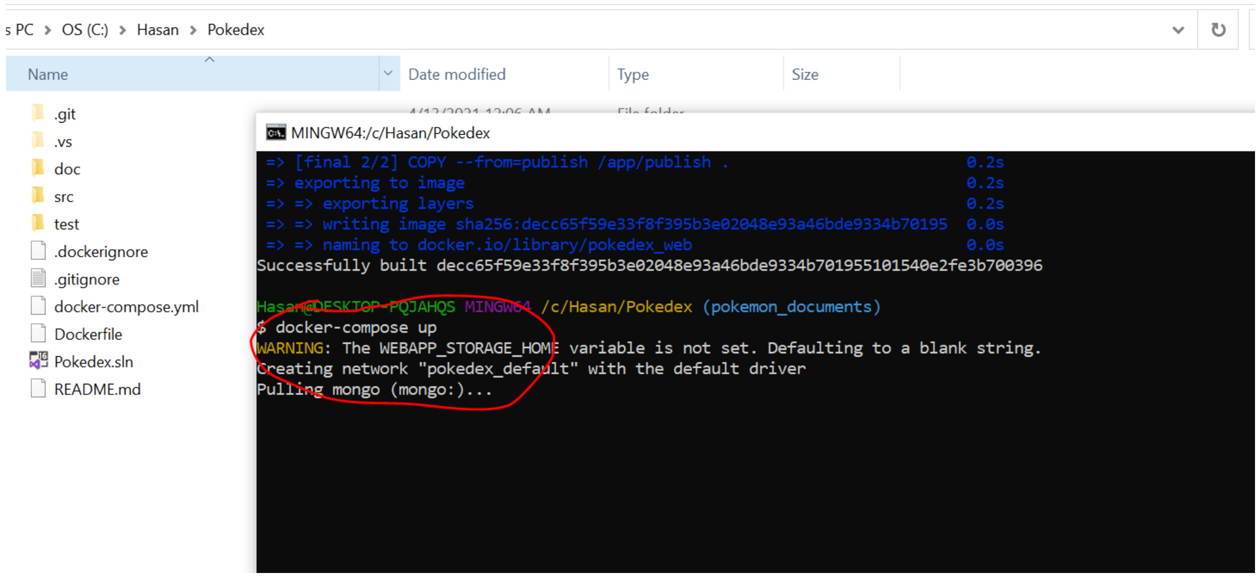
Write below command and enter -
docker-compose build



```
Hasan@DESKTOP-PQJAHQS MINGW64 /c/Hasan/Pokedex (pokemon_documents)
$ git status
On branch pokemon_documents
nothing to commit, working tree clean

Hasan@DESKTOP-PQJAHQS MINGW64 /c/Hasan/Pokedex (pokemon_documents)
$ git pull origin develop
From https://github.com/HasanShahjahan/Pokedex
 * branch            develop    -> FETCH_HEAD
Already up to date.

Hasan@DESKTOP-PQJAHQS MINGW64 /c/Hasan/Pokedex (pokemon_documents)
$ docker-compose build
WARNING: The WEBAPP_STORAGE_HOME variable is not set. Defaulting to a blank string.
mongo uses an image, skipping
Building web
failed to get console mode for stdout: The handle is invalid.
[+] Building 18.8s (12/17)
 => => transferring context: 26.41kB                                        0.0s
 => [build 1/7] FROM mcr.microsoft.com/dotnet/core/sdk:3.1@sha256:ba23df2  0.0s
 => CACHED [build 2/7] WORKDIR /src                                         0.0s
 => CACHED [build 3/7] COPY [src/domain/core/Pokedex.Api/Pokedex.Api.cspr  0.0s
 => CACHED [build 4/7] RUN dotnet restore "src/domain/core/Pokedex.Api/Po  0.0s
 => [build 5/7] COPY . .                                                    0.1s
 => [build 6/7] WORKDIR /src/src/domain/core/Pokedex.Api                    0.1s
 => [build 7/7] RUN dotnet build "Pokedex.Api.csproj" -c Release -o /app  16.6s
```

Step 3

Write below command and enter -
docker-compose up

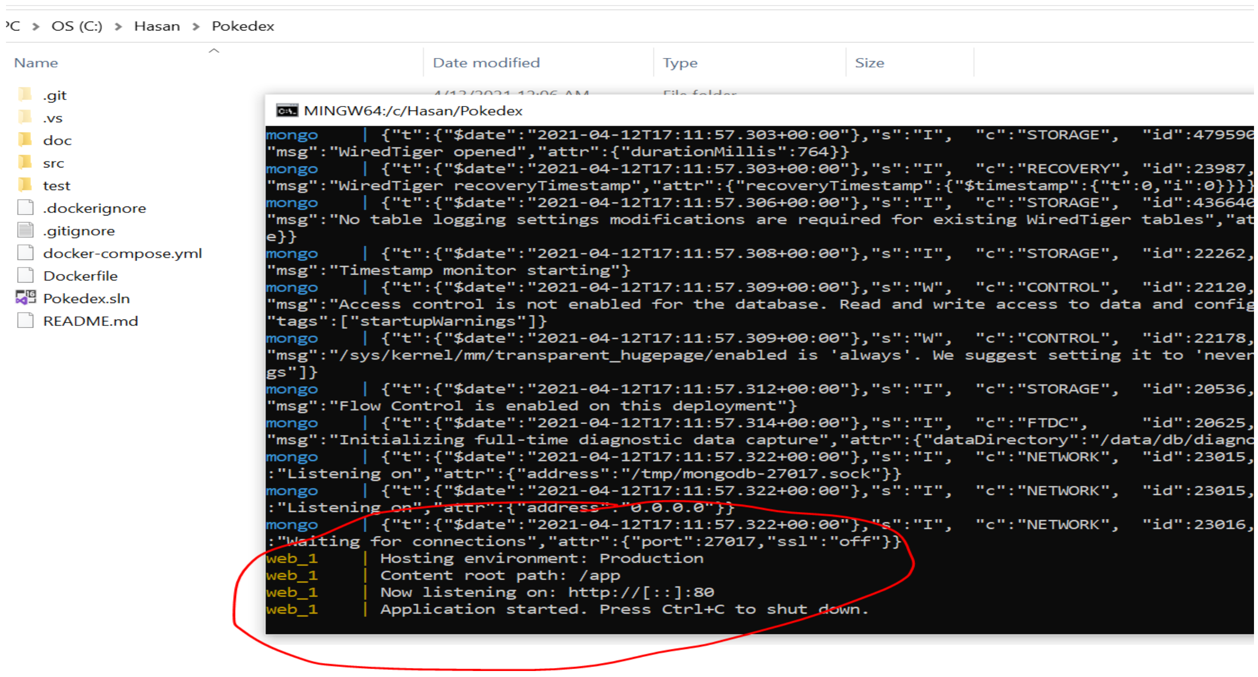


s PC > OS (C:) > Hasan > Pokedex

MINGW64:/c/Hasan/Pokedex

```
=> [final 2/2] COPY --from=publish /app/publish .                           0.2s
=> exporting to image                                                       0.2s
=> => exporting layers                                                      0.2s
=> => writing image sha256:decc65f59e33f8f395b3e02048e93a46bde9334b70195  0.0s
=> => naming to docker.io/library/pokedex_web                               0.0s
Successfully built decc65f59e33f8f395b3e02048e93a46bde9334b701955101540e2fe3b700396

Hasan@DESKTOP-PQJAHQS MINGW64 /c/Hasan/Pokedex (pokemon_documents)
$ docker-compose up
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.

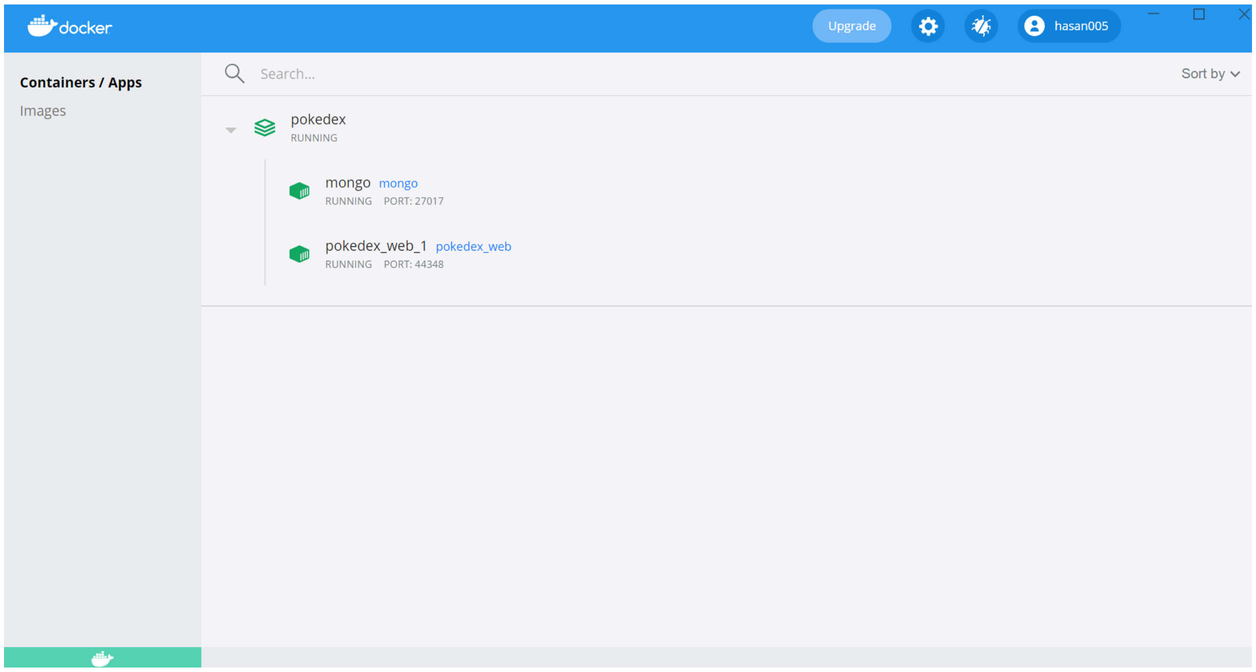


PC > OS (C:) > Hasan > Pokedex

MINGW64:/c/Hasan/Pokedex

```
mongo      | {"t":{"$date":"2021-04-12T17:11:57.303+00:00"},"s":"I",  "c":"STORAGE",  "id":479590
"msg":"WiredTiger opened","attr":{"durationMillis":764}}
mongo      | {"t":{"$date":"2021-04-12T17:11:57.303+00:00"},"s":"I",  "c":"RECOVERY", "id":23987,
"msg":"WiredTiger recoveryTimestamp","attr":{"recoveryTimestamp":{"$timestamp":{"t":0,"i":0}}}}
mongo      | {"t":{"$date":"2021-04-12T17:11:57.306+00:00"},"s":"I",  "c":"STORAGE",  "id":436640
"msg":"No table logging settings modifications are required for existing WiredTiger tables","at
e}}
mongo      | {"t":{"$date":"2021-04-12T17:11:57.308+00:00"},"s":"I",  "c":"STORAGE",  "id":22262,
"msg":"Timestamp monitor starting"}
mongo      | {"t":{"$date":"2021-04-12T17:11:57.309+00:00"},"s":"W",  "c":"CONTROL",  "id":22120,
"msg":"Access control is not enabled for the database. Read and write access to data and config
"tags":["startupWarnings"]}
mongo      | {"t":{"$date":"2021-04-12T17:11:57.309+00:00"},"s":"W",  "c":"CONTROL",  "id":22178,
"msg":"/sys/kernel/mm/transparent_hugepage/enabled is 'always'. We suggest setting it to 'never
gs"]}
mongo      | {"t":{"$date":"2021-04-12T17:11:57.312+00:00"},"s":"I",  "c":"STORAGE",  "id":20536,
"msg":"Flow Control is enabled on this deployment"}
mongo      | {"t":{"$date":"2021-04-12T17:11:57.314+00:00"},"s":"I",  "c":"FTDC",     "id":20625,
"msg":"Initializing full-time diagnostic data capture","attr":{"dataDirectory":"/data/db/diagno
mongo      | {"t":{"$date":"2021-04-12T17:11:57.322+00:00"},"s":"I",  "c":"NETWORK",  "id":23015,
:"Listening on","attr":{"address":"/tmp/mongodb-27017.sock"}}
mongo      | {"t":{"$date":"2021-04-12T17:11:57.322+00:00"},"s":"I",  "c":"NETWORK",  "id":23015,
:"Listening on","attr":{"address":"0.0.0.0"}}
mongo      | {"t":{"$date":"2021-04-12T17:11:57.322+00:00"},"s":"I",  "c":"NETWORK",  "id":23016,
:"Waiting for connections","attr":{"port":27017,"ssl":"off"}}
web_1      | Hosting environment: Production
web_1      | Content root path: /app
web_1      | Now listening on: http://[::]:80
web_1      | Application started. Press Ctrl+C to shut down.
```

Our application will be running our local docker container.



pokedex
RUNNING

mongo mongo
RUNNING PORT: 27017

pokedex_web_1 pokedex_web
RUNNING PORT: 44348

Step 4

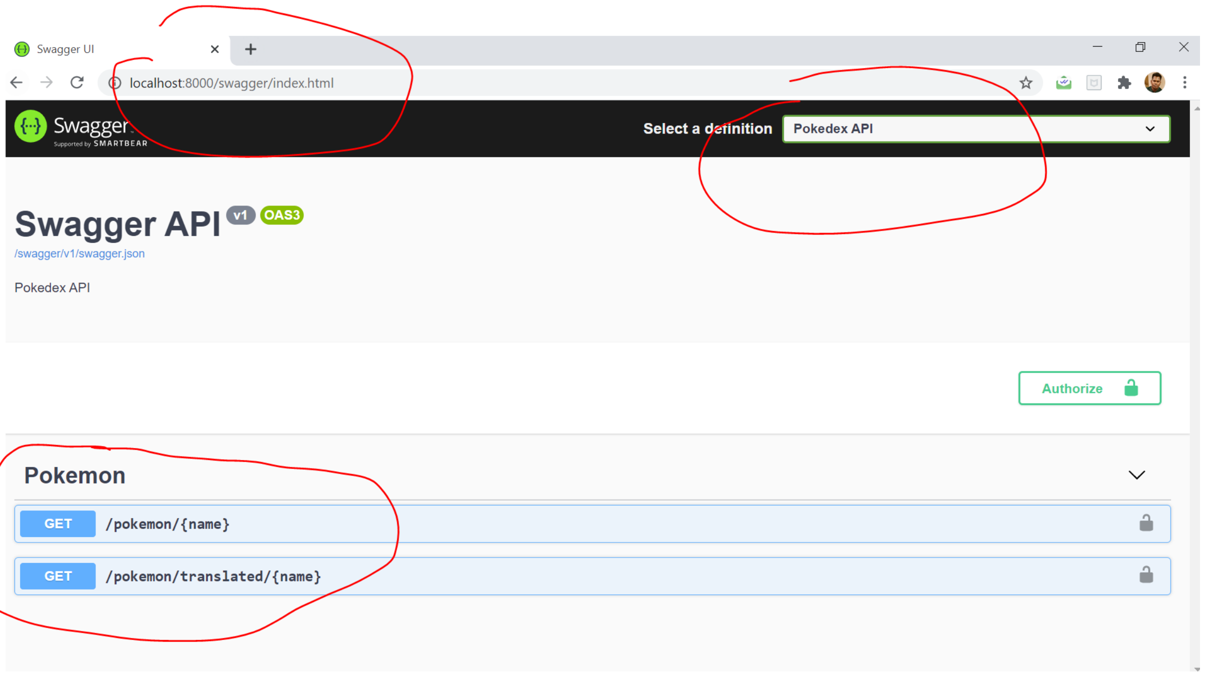
Congratulations !!!

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.eyJ1c2VyX2lkZW50aXR5IjoiSGFzYW4iLCJuYmYiOjE2MTc5MDU4NjYsImV4cCI6MTYyMDQ5Nzg2NiwiaWF0IjoxNjE3OTA1ODY2fQ.o2rTundIHpSacmWA8hR130GGHtSWH9ufWbURBnSJ6G8***

Let’s try -

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



Swagger API v1 OAS3
/swagger/v1/swagger.json

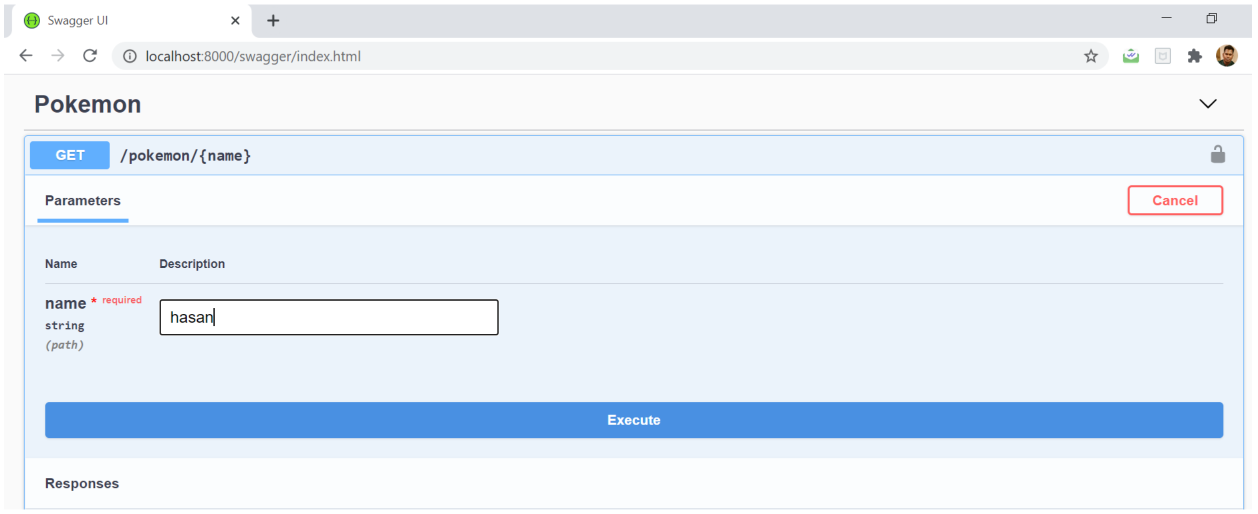
Pokedex API

Pokemon

GET /pokemon/{name}

GET /pokemon/translated/{name}

Request



Pokemon

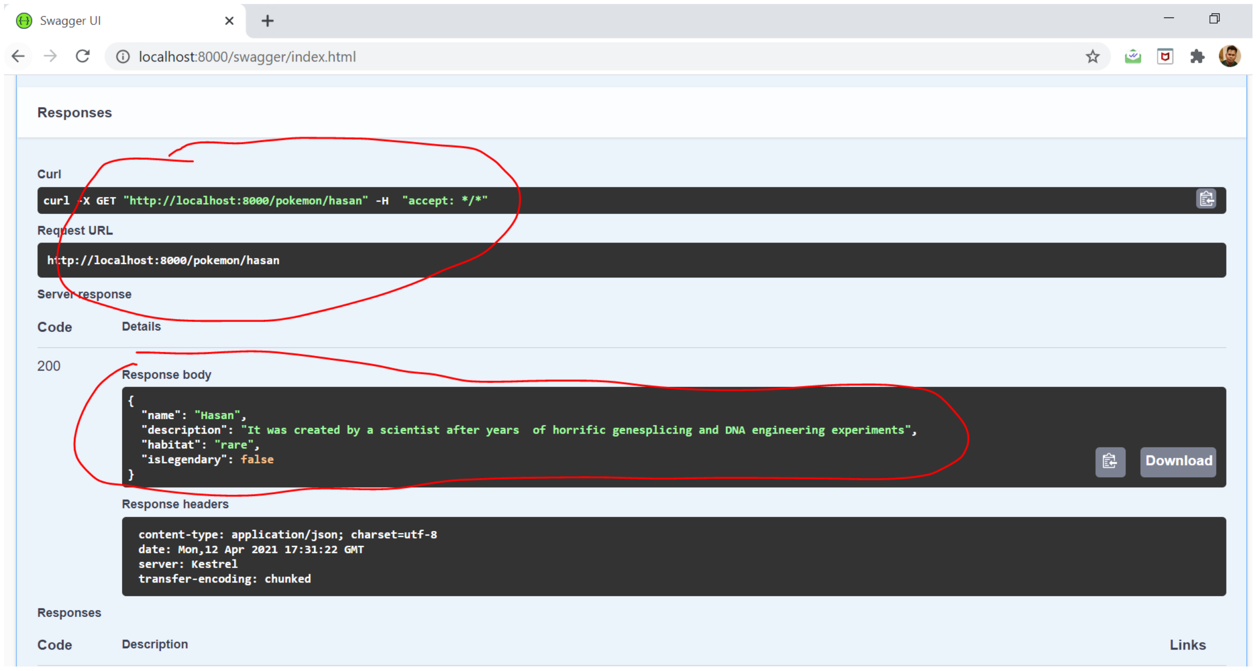
GET /pokemon/{name}

Parameters

Name	Description
name * required string (path)	hasan

Responses

Response



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

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

```
{
  "name": "Hasan",
  "description": "It was created by a scientist after years  of horrific genesplicing and DNA engineering experiments",
  "habitat": "rare",
  "isLegendary": false
}
```

```
content-type: application/json; charset=utf-8
date: Mon,12 Apr 2021 17:31:22 GMT
server: Kestrel
transfer-encoding: chunked
```

It is just a beginning, Cheers !!!