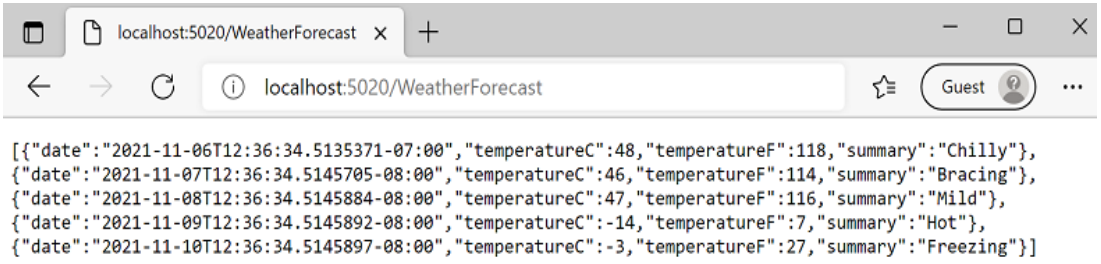
@SampradaN

Simple Microservice without Visual Studio support

Pre-Requisites: Docker Desktop, .Net 6 SDK

---

1. Creating a Web Api using dotnet cli
    a. `dotnet new webapi -o MyMicroservice --no-https -f net6.0`
    b. `cd MyMicroservice`
    c. A web Api project will be created with a default controller called WeatherForecastController
2. Run the web api
    a. `dotnet run`
    b. Now navigate to the url through the browser. http://localhost:portNumber/WeatherForecast
    c. Output resembles the following screenshot



    d.
3. Open Docker Desktop to ensure the host is running. Test it by typing the following in the command prompt
    a. `docker --version`
4. You should now be in the project directory as shown in step 1.
5. To convert your webapi to a microservice, we need to containerize it. To containerize it, Docker is used. To start with creating docker container, a Dockerfile is required. This can be created in multiple ways. As we are doing everything through command-line, let's create the docker file through command-line
    a. `fsutil file createnew Dockerfile 0`
    b. `start Dockerfile`
6. Add the following to the docker file in the editor. Here you can choose the editor as notepad

```
FROM mcr.microsoft.com/dotnet/sdk:6.0 AS build
WORKDIR /src
COPY MyMicroservice.csproj .
RUN dotnet restore
COPY . .
RUN dotnet publish -c release -o /app

FROM mcr.microsoft.com/dotnet/aspnet:6.0
WORKDIR /app
```

```
COPY --from=build /app .
ENTRYPOINT ["dotnet", "MyMicroservice.dll"]
```

7. Adding a .dockerignore file. This step is optional. It is highlighted here, as when you create a microservice from Visual Studio the .dockerignore file will be available.
    a. `fsutil file createnew .dockerignore 0`
    b. `start .dockerignore`
    c. We shall be ignoring the bin, obj folders completely. To do so, please add the following content to the file

```
Dockerfile
[b|B]in
[O|o]bj
```

8. Now, we build the docker container with a tag name, to easily access the image created.
    a. `docker build -t mymicroservice .`
9. To verify that the image was created successfully, use the following command
    a. `docker images`
10. You can run the docker image in the background or interactive mode.
    a. `docker run -it --rm -p 3000:80 --name mymicroservicecontainer mymicroservice`
        i. -i : Interactive mode,
        ii. -t: A pseudo terminal that connects the user's terminal to container through shell / bash,
        iii. --rm : Automatically removes the container when it exits,
        iv. 3000:80: Bind the container port to the hosts port. While running the container,
        v. localhost:3000 will be used to execute your container application,
        vi. --name : container name
11. Open another command prompt, to view that your container is running
    a. `docker ps`
12. This completes the creation of a microservice using Docker container.