Requirements:

To create a docker image with the following:

* Base: WindowsServer2022
* Should have .net framework installed with all required runtime, sdk packages.
* Should have IIS enabled and service running whenever Image is deployed in container.
* Should have SQL server installed, up and running.

Doc Brief:

* This document shows the steps to build the above requirement.
* Recommended approach as per multiple research and Microsoft guidelines as most of the above products (SQL,.Net,WindowsServer) are developed by MS teams, so they exposes docker images which are updated and maintained by Microsoft.
* Test Step to see if SQL Server is up and running.
* Test step to see if IIS service is working.

Pre-requisite:

* If running locally, Setup Docker Desktop
* OS should be Windows Enterprise or Windows Professional. Windows Home will not allow to run windows container, it uses WSL which is linux container. But this requirement need Windows Container.

Ref:

A screenshot of a computer

Description automatically generated

* Run Docker desktop and switch to Windows container.

Recommended Approach:

* Ideally IIS and SQL are two separate service, this should be isolated in two docker images and both containers can talk to each other.
* Using Docker compose, we can make Windows Dotnet IIS depends on SQL server. This way the IIS container will only come up once SQL container is up and running.
* Details related to Docker image for Windows Server with DotNet and IIS: It has below things:
  + Base Image: WindowsServer2022
  + Install .NET Fx 4.8.1
  + Apply latest patch
  + Install NuGet CLI
  + Install VS components
  + Install VS Build Tools
  + Install Targeting Packs
  + Install IIS Feature
  + Expose Port 80 and make IIS service up and running
* Note: All the docker files are made from reference with Microsoft docker images to be in sync with Microsoft Dotnet Docker Images Updates and SQL updates,

Ref:

* + [microsoft/dotnet-framework-docker: The repo for the official docker images for .NET Framework on Windows Server Core. (github.com)](https://github.com/microsoft/dotnet-framework-docker)
  + [mssql-docker/windows/mssql-server-windows-developer at master · microsoft/mssql-docker (github.com)](https://github.com/microsoft/mssql-docker/tree/master/windows/mssql-server-windows-developer)
  + [mssql-docker/windows/mssql-server-windows-developer at windows · sqlchen/mssql-docker · GitHub](https://github.com/sqlchen/mssql-docker/tree/windows/windows/mssql-server-windows-developer)
  + [mssql-docker/windows at master · microsoft/mssql-docker (github.com)](https://github.com/microsoft/mssql-docker/tree/master/windows)
  + <https://www.twilio.com/blog/containerize-your-aspdotnet-core-application-and-sql-server-with-docker>
* Running in a single docker file will result in an issue, as IIS service and SQL server needto be up during everytime when we spin up the container. This will result in error when the container tries to do both the things one after the other.

Steps to Build the image:

* Extract the Project folder:

A screenshot of a computer

Description automatically generated

* Note the path to the project.
* Open Windows PowerShell
* Navigating to the Root of the Project

A blue screen with white text

Description automatically generated

* Run command: **docker-compose up --build**

A blue screen with white text

Description automatically generated

* You will see below output: This builds both the images WindowsServer-DotNet-IIS and WindowsServer-SQL. Also it starts with the images in a container.
* Open a new Powershell window and Check if both the containers are up and healthy using command: **docker ps -a**

A blue screen with white text

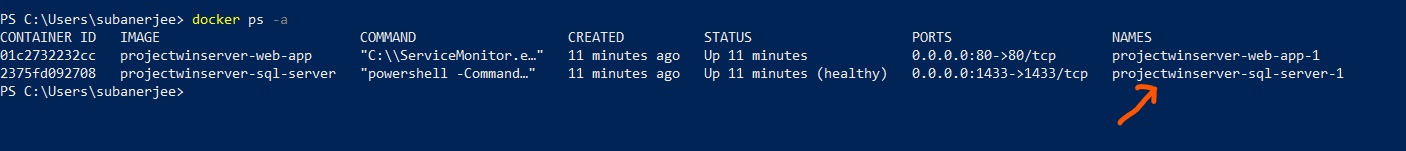
Description automatically generated

* We are all set!!! Now lets test.

Test SQL server Type1:

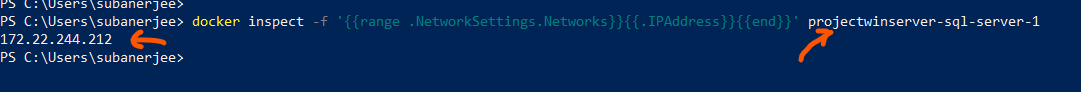
* You can verify if SQL Server is running on TCP port 1433 in a Windows container by using various methods, such as checking the container's network configuration or by running some commands. Here's a step-by-step guide to help you with this verification:
* Note the container name for SQL server after using command: **docker ps -a**

Name found: projectwinserver-sql-server-1



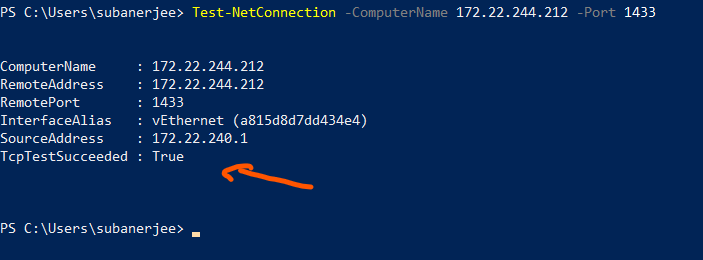
* Run command:

docker inspect -f '{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}' **<container\_name\_or\_id>**



* Use the IP extracted in above step and run below command:

Test-NetConnection -ComputerName **<container\_ip>** -Port 1433



Test SQL server Type2:

* You can also verify the SQL Server configuration from within the container. You can attach to the running container and use the SQL Server tools to check the configuration:
* Command:

docker exec -it **<container\_name\_or\_id>** powershell

A screenshot of a computer program

Description automatically generated

* Run the commands shown in screenshot  
  A screenshot of a computer program

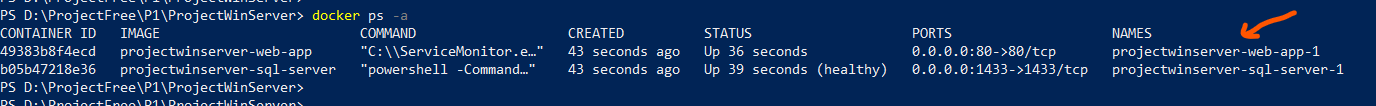
  Description automatically generated
* Type **exit** to come out of interactive PowerShell mode.

A computer screen with text and arrow

Description automatically generated

Test IIS Service on WindowsServer with Dotnet and IIS:

* Run **docker ps -a**



* Note the container name.
* Run : docker exec -it projectwinserver-web-app-1 powershell
* Then check if service is running.

A blue screen with white text and orange arrow

Description automatically generated