Deploy a Prototype Webapp of a Bank Login Page Using Docker

Introduction

In this project, we will create a prototype web application for a bank, consisting of a login page and a dashboard page. We will deploy this web application to a Docker container running IIS (Internet Information Services). This project aims to demonstrate the process of containerizing a web application for deployment using Docker.

Prerequisites

Before starting the project, ensure that the following tools and software are installed:

Visual Studio: An Integrated Development Environment (IDE) for coding the web application.

Docker 2.2.x: For containerization and application composition.

Windows 10: As the operating system.

Create an ASP.NET Core Web Application

Start by creating an ASP.NET Core web application with two pages:

Login Page: This page allows customers to enter their username and password.

Dashboard Page: The dashboard page displays account-related activities.

We can use Visual Studio to create this application. Define the required models, controllers, and views for the login and dashboard pages.

Set Up Docker

Next, set up Docker on the computer. We can download and install Docker Desktop for Windows from the official Docker website. Ensure that Docker is running and that we can access the Docker command-line interface (CLI).

Create a Dockerfile

Create a Dockerfile to define the containerization of your web application.

```
#See https://aka.ms/customizecontainer to learn how to customize your debug
container and how Visual Studio uses this Dockerfile to build your images for
faster debugging.

FROM mcr.microsoft.com/dotnet/aspnet:6.0 AS base
WORKDIR /app
EXPOSE 80

FROM mcr.microsoft.com/dotnet/sdk:6.0 AS build
WORKDIR /src
COPY ["PractieProjDocker/PractieProjDocker.csproj", "PractieProjDocker/"]
RUN dotnet restore "PractieProjDocker/PractieProjDocker.csproj"
COPY . .
WORKDIR "/src/PractieProjDocker"
RUN dotnet build "PractieProjDocker.csproj" -c Release -o /app/build

FROM build AS publish
RUN dotnet publish "PractieProjDocker.csproj" -c Release -o /app/publish
/p:UseAppHost=false

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

Build the Docker Image

Using the Docker CLI, navigate to the directory containing our Dockerfile and build the Docker image. Run the following command:

docker build -t bank-proj-img -f PracticeProjDocker/Dockerfile .

This command builds a Docker image tagged as bank-proj-img.

Run the Docker Container

Once the Docker image is built, we can run a Docker container from it. Bind the container's port 80 to a port on he host machine.

This creates a container named bankprojcontainer based on the bank-proj-img image, mapping port 5200 on the host to port 80 in the container.

Access the Web Application

We can now access the bank web application by opening a web browser and navigating to http://localhost:5200. We should see our login page.

Push to Dockerhub

After creating the image and container we can push it to dockerhub using the following commands:

```
docker login

docker tag bank-proj-img prashastvats/practiceprojdocker:latest

docker push prashastvats/practiceprojdocker:latest
```

Conclusion

In this project, we created a prototype web application for a bank and deployed it to a Docker container running IIS. The steps included setting up Docker, creating a Dockerfile, building a Docker image, running a container, and accessing the web application. This project demonstrates a basic workflow for containerizing and deploying web applications using Docker.

Dockerhub Link :

https://hub.docker.com/repository/docker/prashastvats/practiceprojdocker/general

Github Link :

https://github.com/PrashastVats1/Practice-Projects/tree/Phase3Docker