David Ziegelbauer

Installation and Maintenance

TicketPro

Table of Contents

[Required Infrastructure 2](#_Toc157238785)

[Database 2](#_Toc157238786)

[Compute Resources 2](#_Toc157238787)

[Licensing 2](#_Toc157238788)

[Traditional Deployment 3](#_Toc157238789)

[Building From Source 3](#_Toc157238790)

[Deploying the Application 4](#_Toc157238791)

[Containerized Deployment 5](#_Toc157238792)

# Required Infrastructure

## Database

TicketPro requires a PostgreSQL database to store user and ticket information. The implementation details of that database can be determined by the user (on premises, containerized, cloud, etc.). Once an implementation of PostgreSQL has been chosen, you’ll need to make note of the connection details. For security reason, no connection information is hard coded or stored within TicketPro, they are passed as a connection string via the “CONNECTION\_STRING” environment variable at runtime. The connection string should be sent in standard PostgreSQL format as referenced in the official PostgreSQL documentation at <https://www.npgsql.org/doc/connection-string-parameters.html>.

## Compute Resources

The TicketPro application is designed to run in either a traditional (dedicated physical or virtual server) or containerized environment. It is compatible with any operating system that supports the .NET 8 runtime. The selected compute environment must have network connectivity to the selected database environment. They may even operate on the same server if you choose.

## Licensing

The TicketPro MVP was created using an academic license for the Syncfusion Blazor library. To run the application in production, a commercial license must be procured. A valid license key must be passed to the application at runtime through the “SYNCFUSION\_KEY” environment variable. Licenses can be procured from Syncfusion directly on their web site at <https://www.syncfusion.com/sales/unlimitedlicense>

# Traditional Deployment

## Building From Source

If you have received TicketPro as a source bundle, you’ll need to build it prior to deploying the application. To begin, ensure you have the .NET 8 SDK installed on the machine where you will build the application. It can be obtained from Microsoft at no charge from <https://dotnet.microsoft.com/en-us/download/dotnet/8.0>.

Next, navigate to the root of the application source code tree and run the “dotnet restore” command to download the required NUGET packages.

A black background with white text

Description automatically generated

Figure 1: Restore NUGET dependencies

With the dependencies in place, the next step is to compile the source code using the “dotnet build” command. Make sure to pass the “-c Release” argument to build in optimized release mode.

A screen shot of a computer

Description automatically generated

Figure 2: Compile the application

Finally, build an installation bundle using the “dotnet publish” command. Once again, ensure you include the “-c Release” argument for an optimized package.

A screen shot of a computer

Description automatically generated

Figure 3: Build the application bundle

You’ll find your bundle in the source tree at “./TicketPro/bin/Release/net8.0/publish”. You’ll need to take everything in this bundle to your desired compute platform.

## Deploying the Application

Once you have the application files stored in your desired location, you’ll need to build a launch script. The purpose of the script is to set the appropriate environment variables prior to launching the application. There are many ways to go about this depending on where and how your organization stores sensitive information. Here is a basic (not secure!) example of how it might look in a bash shell script:

export ASPNETCORE\_ENVIRONMENT=Production  
export ASPNETCORE\_URLS="http://\*:8080;https://\*:8081"  
export SYNCFUSION\_KEY="<YourKeyHere>"  
export CONNECTION\_STRING="User ID=<username>;Password=<password>;Host=<db\_server>;Port=5432;Database=ticketpro;"  
dotnet TicketPro.dll

And here is a similar (still not secure, do not store your secrets this way in production!) PowerShell version:

$env:ASPNETCORE\_ENVIRONMENT = "Production"

$env:ASPNETCORE\_URLS = "http://\*:8080;https://\*:8081"

$env:SYNCFUSION\_KEY = "<YourKeyHere>"

$env:CONNECTION\_STRING = "User ID=<u>;Password=<p>;Host=<s>;Port=5432;Database=ticketpro;"

& dotnet TicketPro.dll

You can use these scripts to launch your application. You must ensure the following four environment variables are set correctly in the script:

* ASPNETCORE\_ENVIRONMENT – This should be set to “Production” in most cases but can be set to “Development” for more verbose debug output. The “Development” setting is not secure and should not be used in production.
* ASPNETCORE\_URLS – This string sets what protocols, addresses, and ports the application will listen on. This is a semicolon separated list of listeners formatted as “<protocol>://<address>:<port>”.
* SYNCFUSION\_KEY – This is the commercial license key purchased for the Syncfusion Blazor library.
* CONNECTION\_STRING – This is the PostgreSQL connection string for the database. Specification details can be found at <https://www.npgsql.org/doc/connection-string-parameters.html>.

# Containerized Deployment

The containerized deployment option is based on the Docker ecosystem. If you prefer a different containerization solution, TicketPro can be configured to run in other environments; however, instructions for doing so are the responsibility of the user.

A dockerfile for containerized deployment is provided in the source code bundle at “./TicketPro/Dockerfile”. An example shell script to build and push the container image might look like this:

export REPOSITORY\_URI="your\_container\_repository\_uri"  
export IMAGE\_TAG="your\_image\_tag"  
docker build -t $REPOSITORY\_URI:latest --file ./TicketPro/Dockerfile .  
docker tag $REPOSITORY\_URI:latest $REPOSITORY\_URI:$IMAGE\_TAG  
docker push $REPOSITORY\_URI:latest  
docker push $REPOSITORY\_URI:$IMAGE\_TAG

You could then deploy the image from your container repository like any other container. When deploying the container, you must ensure the following four environment variables are set:

* ASPNETCORE\_ENVIRONMENT – This should be set to “Production” in most cases but can be set to “Development” for more verbose debug output. The “Development” setting is not secure and should not be used in production.
* ASPNETCORE\_URLS – This string sets what protocols, addresses, and ports the application will listen on. This is a semicolon separated list of listeners formatted as “<protocol>://<address>:<port>”.
* SYNCFUSION\_KEY – This is the commercial license key purchased for the Syncfusion Blazor library.
* CONNECTION\_STRING – This is the PostgreSQL connection string for the database. Specification details can be found at <https://www.npgsql.org/doc/connection-string-parameters.html>.