.NET 9 App Dev Hands-On Lab Prerequisites

Before starting the rest of the workshop, you must have the .NET (Core) SDK, .NET Runtime, access to a local SQL Server Database, an appropriate .NET development IDE, and an SQL Server IDE.

Part 1: Permissions

If you do not have permission to install the software and services listed in this document, please have someone in your organization install the requirements,

Part 2: .NET SDK/Runtime

.NET/IDE Version Choices

This workshop uses .NET 9. If using Visual Studio for the labs, reference the minimum version based on the runtime and SDK. This will be necessary when you install (or confirm your installation of) Visual Studio for Windows.

Available .NET 9 versions:

- SDK => 9.0.100+
- Runtime => 9.0.0+

Step 1: Install the .NET Runtime and SDK

- Download and install the latest .NET 9 SDK from http://dot.net. The .NET 9 Runtime, ASP.NET Core 9 Runtime, and .NET Desktop Runtime are included in the SDK but can also be installed separately. Note: The .NET Desktop Runtime is not used for this hands-on lab.

Step 2: Confirm the .NET Runtime and SDK installation

- Check the version of the .NET Runtime by entering (must have 9.0.x or above):

```
dotnet --list-runtimes
```

- Check the version of the .NET SDK by entering (must have 9.0.xxx or above):

```
dotnet --list-sdks
```

- You can check the support status and if updates are available to your installed versions by using the following:

```
dotnet sdk check
```

Step 3: Enable Tab Completion for the .NET CLI (Command Line Interface)

The following are excerpts from https://learn.microsoft.com/en-us/dotnet/core/tools/enable-tab-autocomplete

Option 1: PowerShell

- Visual Studio uses PowerShell v5.x in the Package Manager Console. Most Windows 11 machines have version 7.x as the default. To have autocomplete in both versions, you must update both user profile files.
- Open your user profile in PowerShell:

```
notepad $PROFILE.CurrentUserAllHosts
or
notepad $PROFILE
```

- Add the following to your profile file:

```
# PowerShell parameter completion shim for the dotnet CLI
Register-ArgumentCompleter -Native -CommandName dotnet -ScriptBlock {
    param($wordToComplete, $commandAst, $cursorPosition)
        dotnet complete --position $cursorPosition "$commandAst" | ForEach-Object {
            [System.Management.Automation.CompletionResult]::new($_, $_, 'ParameterValue', $_)
        }
}
```

Option 2: bash

- Add the following to your `.bashrc` file:

```
# bash parameter completion for the dotnet CLI

function _dotnet_bash_complete()
{
  local cur="${COMP_WORDS[COMP_CWORD]}" IFS=$'\n' # On Windows you may need to use use IFS=$'\r\n'
  local candidates

  read -d '' -ra candidates < <(dotnet complete --position "${COMP_POINT}" "${COMP_LINE}"
2>/dev/null)

  read -d '' -ra COMPREPLY < <(compgen -W "${candidates[*]:-}" -- "$cur")
}

complete -f -F _dotnet_bash_complete dotnet
```

Part 3: .NET Development Tool

Supported .NET development IDEs include (one is required):

- [Windows] any edition of Visual Studio (the Community Edition is free to use). See below for minimum version information.
- [Any OS] Visual Studio Code 1.96+ (free) or JetBrains Rider (free for non-commercial use)
- **NOTE:** Visual Studio for the Mac is no longer supported

Visual Studio Required Minimum Version (based on .NET SDK version)

- Visual Studio =>

.NET Runtime Version	.NET SDK Version	VS Win Version
9.0.5+	9.0.3XX	**2022 17.14**+
9.0.2+	9.0.2XX	**2022 17.13**+
9.0.0+	9.0.1XX	**2022 17.12**+

See https://dotnet.microsoft.com/en-us/download/dotnet/9.0 for the complete version correlations.

NOTE: The "2022" label is used for branding purposes. The actual version of Visual Studio can be found under the **Help => About Microsoft Visual Studio** menu item (see image below):

Step 1: Install a Development IDE (one is required)

Option 1: [Windows] Download and install any edition of Visual Studio 2022 (Community is Free)

- Download Visual Studio 2022 (any edition) from the Visual Studio home page: https://www.visualstudio.com
- The Community Edition is free and has everything you need to complete this Hands-On Lab
- The Visual Studio installation is divided into workloads based on what type of work you intend to do. For this lab, select the "**ASP.NET and web development**" workload and the "**Data storage and processing**" workloads.

Option 2: [Any OS] Download and install Visual Studio Code (Free)

- Download Visual Studio Code from https://visualstudio.microsoft.com/.

Install the "C#" and "C# Dev Kit" extensions.

Option 3: [Any OS] Download and install Rider from JetBrains (Free for Non-Commercial Use)

- Download Visual Studio Code from https://www.jetbrains.com/rider/

Part 4: Install xUnit V3 Templates

Until the new templates for xUnit ship with Visual Studio/.NET installs, the templates must be installed manually.

Step 1: Install the xUnit V3 Templates

- Run the following from a command line:

```
dotnet new install xunit.v3.templates
```

Step 2: Confirm the xUnit V3 Templates

Option 1: Show the list of installed templates using the .NET CLI

- Run the following from a command line:

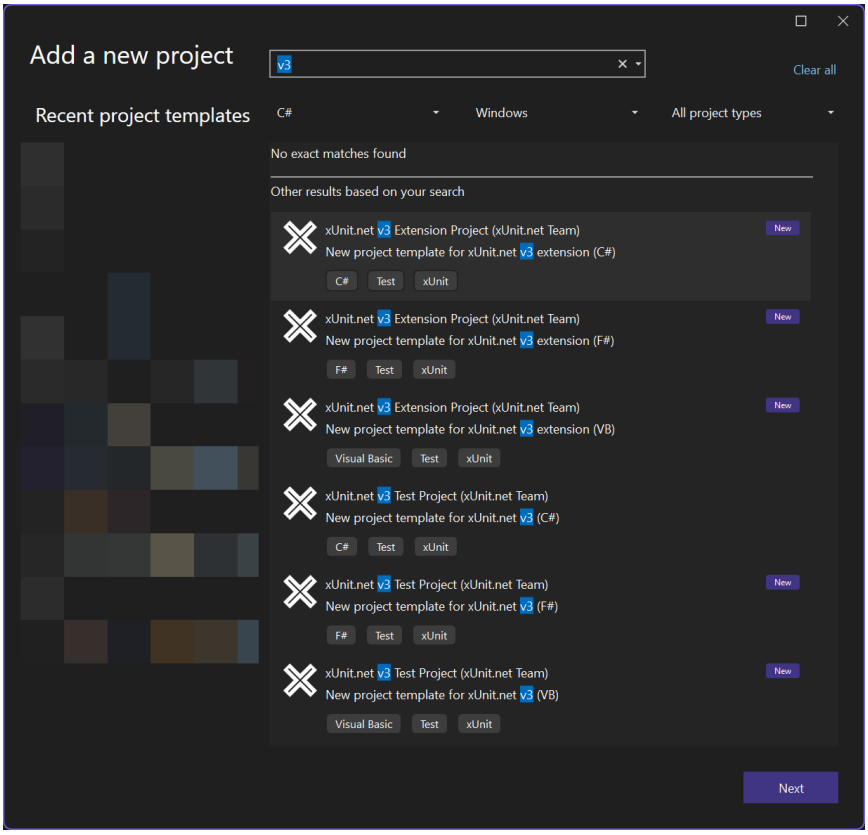
```
dotnet new list
```

- You should see xUnit.net v3 templates at the end of the list:

```
Template Name                    Short Name        Language   Tags
xUnit.net v3 Extension Project   xunit3-extension  [C#],F#,VB  Test/xUnit
xUnit.net v3 Test Project        xunit3            [C#],F#,VB  Test/xUnit
```

Option 2: [Windows] Show the list of installed templates using the Visual Studio

- Open Visual Studio and select File -> New -> Project. In the new project dialog, enter "v3" in the search box, and you should see the following list:



Part 5: SQL Server Development Environment

To complete this workshop, you must install SQL Server Management Studio (SSMS) or Azure Data Studio (ADS).

Supported SQL Server IDEs:

- [Windows] SQL Server Management Studio 18.11+/19.2+/20.0+
  - **NOTE:** SQL Server Management Studio version has no relation to SQL Server version.
- [Any OS] Azure Data Studio 1.50+

Step 1: Install SQL Server IDE (SSMS, ADS, or Visual Studio Code)

Option 1: [Windows] SQL Server Management Studio

- Download/Install SQL Server Management Studio (SSMS) from https://aka.ms/ssmsfullsetup

Option 2: [Windows] Visual Studio

- See above for download instructions.

Option 3: [Any OS] Visual Studio Code

- Download Visual Studio Code from https://visualstudio.microsoft.com/.
- Install the "SQL Server (mssql)" extension.

Option 4: [Any OS] Azure Data Studio (deprecated)

- Download/Install the free Azure Data Studio from https://learn.microsoft.com/en-us/azure-data-studio/download-azure-data-studio.

NOTE: Azure Data Studio will be retired on **February 28, 2026**.

Step 2: Connect to your local SQL Server

If you already have SQL Server installed and can connect using SSMS, or ADS, you have completed the prerequisites for this workshop. Please note your connection string, as you will need to update the supplied connection strings in the labs.

When connecting to `LocalDb`, use Windows Authentication (on Windows), and the Server name to use is:

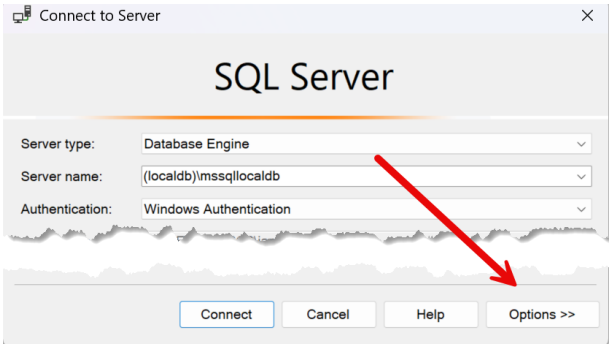
```
(localdb)\mssqllocaldb
```

If that server name doesn't work, you can also try the following, which is another location where VS2022 will install LocalDb:

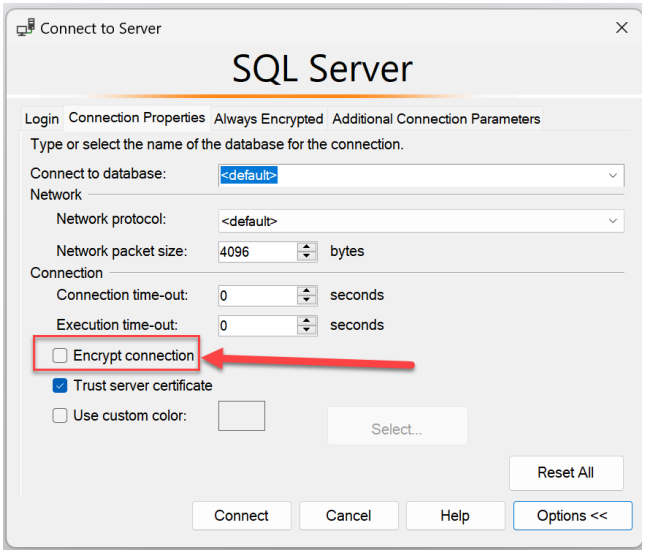
```
(localdb)\ProjectModels
```

If the connection to either of these does not work and you get an error about encryption, click the options button on the connection screen:

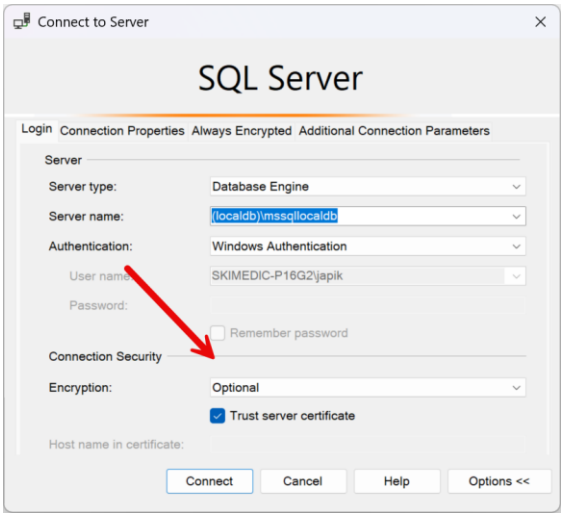
SSMS:



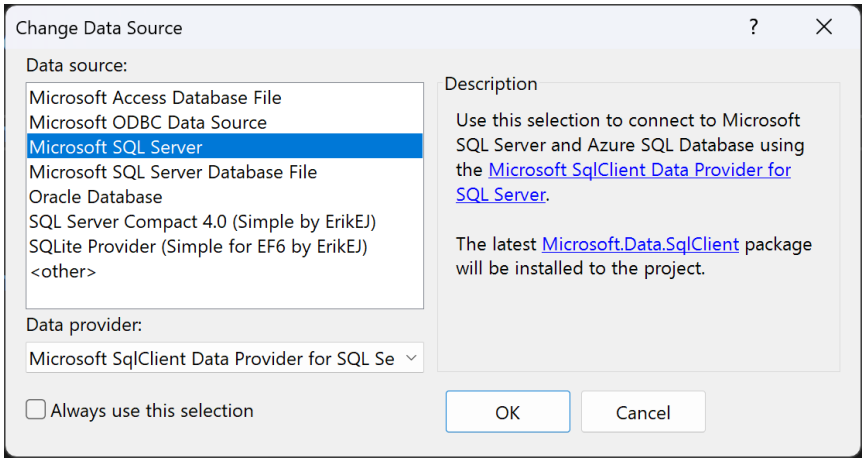
On the options page, uncheck the "Encrypt the connection" check box (if your options window looks different, proceed to the next image):

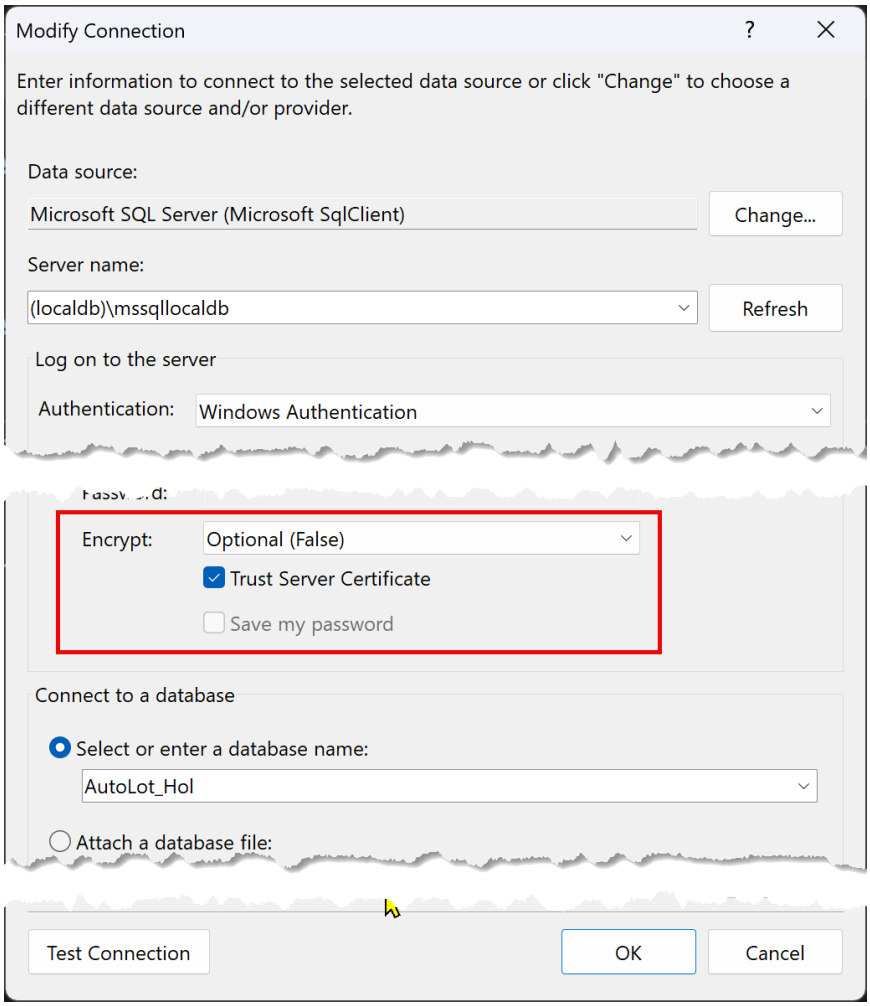


Based on your version of SSMS, it might look like this:



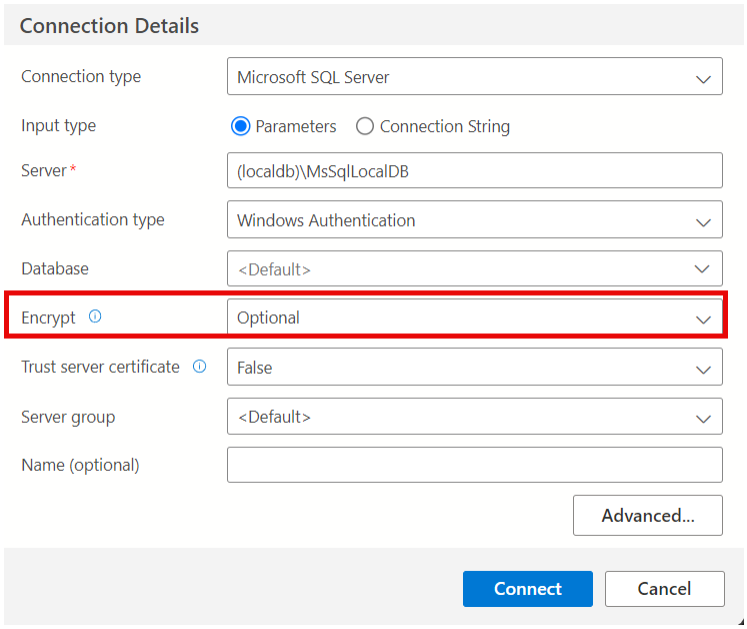
Visual Studio:





NOTE: Visual Studio requires a database name; otherwise, it defaults to Master. Since you haven't yet created the database for this workshop, you can choose Master and edit the connection later, after the database is created. Connections can be found in the Data Connections node in Server Explorer.

Visual Studio Code or Azure Data Studio:



Step 3: Confirm the Version of SQL Server (2019+)

Check that the version of SQL Server is 2019 or later by running the following in a query window:

```
SELECT @@Version
```

The command should return the following (or higher):

```
Microsoft SQL Server 2019 (rest omitted…)
```

If your version is not 2019 (or later), or you do not have SQL Server installed, follow one of the options below to install SQL Server.

Part 6: Installing SQL Server

SQL Server

Access to a local instance of SQL Server is required to complete this workshop. If you do not have a local instance of SQL Server, one of the following options must be installed:

SQL Server options (one is required):

- [Windows]
  - SQL Server LocalDb (installed with Visual Studio 2022), or
  - SQL Server 2019+ (or newer) Developer Edition (or above)
- [Any OS]
  - Docker Community with SQL Server 2019/2022 (instructions below)
- [ARM-Mac]
  - https://sqlblog.org/2023/03/03/sql-server-apple-silicon

Step 1: Installing SQL Server

You must have access to SQL Server 2019+ for this workshop. If you are on a Windows machine, using LocalDb, installed with Visual Studio 2022 is the easiest option. If you have another edition of SQL Server installed on your local machine, you have completed the prerequisites for this workshop.

Option 1: Download and install SQL Server 2022 Local DB (Windows)

NOTE: LocalDB (2019 or 2022) is already installed with Visual Studio

- Download and install SQL Server Local DB: https://learn.microsoft.com/en-us/sql/database-engine/configure-windows/sql-server-express-localdb?view=sql-server-ver16

Option 2: Download and install SQL Server 2022 Developer Edition (Windows)

- Download/Install SQL Server 2022 Developer Edition: https://go.microsoft.com/fwlink/p/?linkid=2215158

Option 3: Install Docker Desktop (Windows/Mac/Linux)

Docker is a containerization platform that runs on Windows, MacOS, and Linux.

NOTE: If you are using a Windows-based machine and have SQL Server installed, Docker is optional for this workshop. If you are not on a Windows/Linux machine or can't have SQL Server 2019/2022 installed on your laptop, Docker is required.

- Download and install Docker Desktop from https://www.docker.com/products/docker-desktop
    a) Select the edition for your operating system (Windows/Mac)
    b) During installation, when prompted about what type of containers to use, select Linux containers (and not Windows containers), even if you are on a Windows machine. This type of container will be loaded and unrelated to your computer's operating system.
    c) This is a free tool, but requires you to have a Docker user ID and password

A Docker image is like a class definition, and a Docker Container is like an instance of that class. To run SQL Server in Docker, you must first pull the image from Docker Hub and create a container using that image.

- Pull the SQL Server 2022 for Linux image. Enter the following command:

```
docker pull mcr.microsoft.com/mssql/server:2022-latest
```

- When creating an image, two required environment variables are "ACCEPT_EULA" and "SA_PASSWORD". An optional environment variable, "MSSQL_PID" sets the product version. The host port mapping to the image port needs to be set, and a friendly name added. Create the container using the following command:
    a) **NOTE:** On Windows, use double quotes ("). On Mac and Linux, use single quotes (').

```
docker run -e "ACCEPT_EULA=Y" -e "MSSQL_SA_PASSWORD=P@ssw0rd" -p 5433:1433 --name AutoLot2022 --
hostname AutoLotSql2022 -d mcr.microsoft.com/mssql/server:2022-latest
```

- To test the install, connect to the Docker instance of SQL Server using the following connection string:

```
server=.,5433;Database=AutoLot_Hol;User Id=sa;Password=P@ssw0rd;Encrypt=false;
```

Summary

These are all the tools you need to complete this Hands-on Lab. When the class starts, you will get the URL for the repo to clone/download.