Visual Studio – Solution and Projects

This document describes how to organize projects in Visual Studio.

In Visual Studio, a solution is a collection of projects. You can think of a solution as "the entire product". The projects are "the pieces of the product". For example, assume that the software product you are working on is Canvas. Canvas might consist of many sub-parts. For example, Canvas web UI, Canvas framework, Canvas database layer, Canvas server, etc. A good way to organize this is Visual Studio is shown in Figure 1:

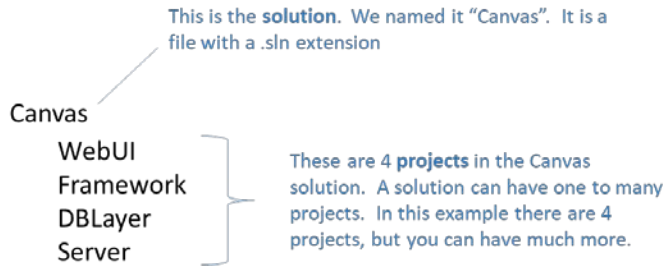


Figure 1: In Visual Studio, a **solution** is a collection of **projects**.

When you want to start a Visual Studio project, always create a blank solution first. Then add one or more projects to this solution.

Why is this recommended? If you don't create a solution and give it a meaningful name, Visual Studio creates one for you and gives it a name identical to the project you created which will not necessarily be a meaningful name. For example, consider the example of Figure 1 where you did not first create a solution named "Canvas". Instead you started by creating the WebUI project first, and then later added the other 3 projects. In this case, Visual Studio organizes your solution like in Figure 2 (which is not what you want).

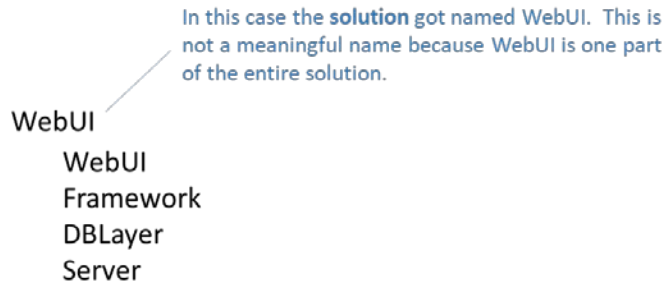
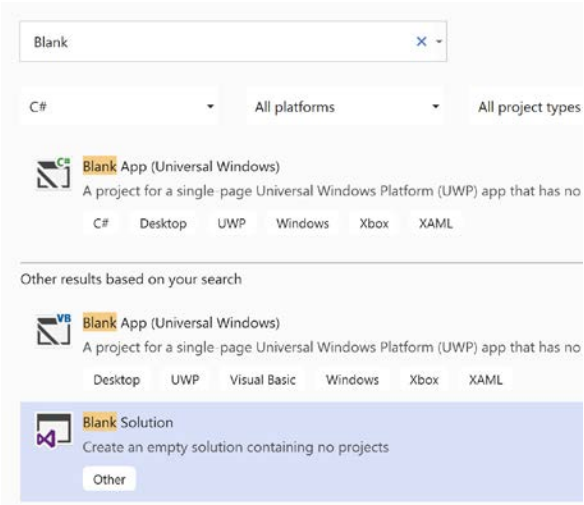


Figure 2: If you don't create a solution, Visual Studio creates one for you and gives it a name identical to the first project (not necessarily what you had in mind).

How to Create a Solution in Visual Studio?

1. Open Visual Studio and click **File → New Project**.

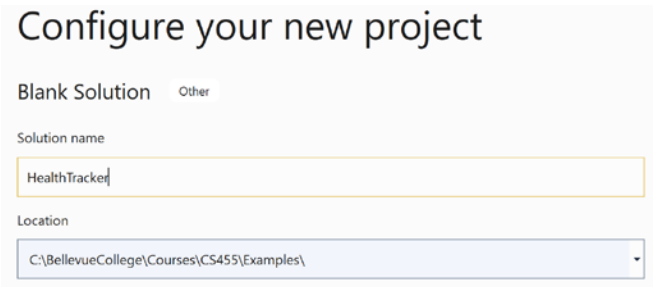
2. In the **Create a new project** screen, type Blank in the search text box. This should bring the Blank Solution template into view



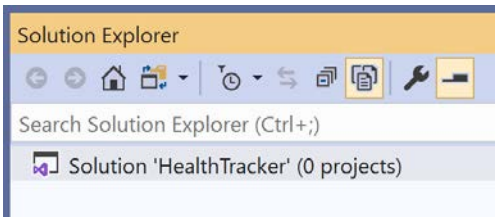
3. Select **Blank Solution** and click **Next**.

4. Give your solution a meaningful name. The name should not be a subpart of the solution (that's reserved to the projects you will create under the solution). For example, a solution might consist of a desktop application + a mobile application + a Windows service + several libraries. All subparts together make the solution. So do not name your solution HealthTrackerApp because this name is more suitable for the mobile app which is a sub-part of the solution. A more suitable name is HeatlhTracker (assume our product is a system for managing health data).

   Choose the location where you want to save the solution. Then click the **Create** button.



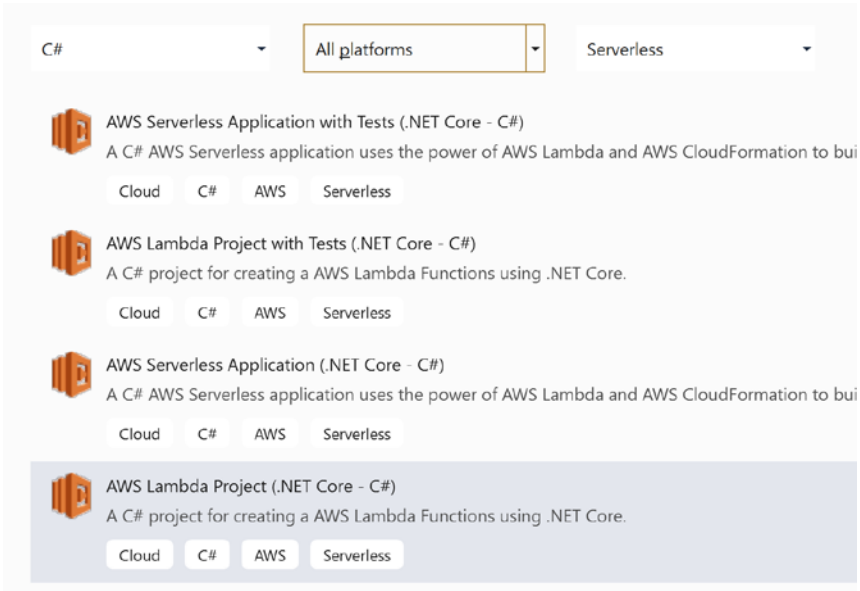
5. Visual Studio creates a blank solution for you. Notice that because we chose a Blank Solution template, the HealthTracker solution is blank and there is nothing in it (0 projects). In the next step we can start adding projects to the solution.



6. In the Solution Explorer pane, right-click on HealthTracker and choose menu **Add → New Project…**. This brings the **Create a new project** back into view.

   (Assume that one sub-part of our HealthTracker solution is an AWS Lambda function).

7. Enter C#, All platforms, Serverless and this should bring a few AWS templates into view.

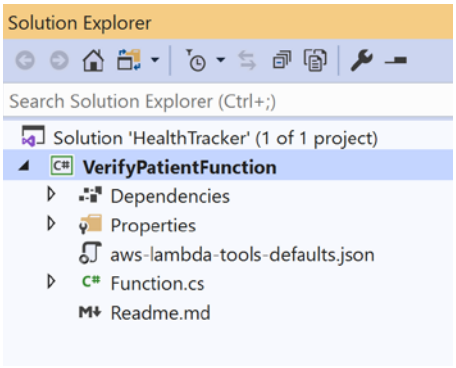


   Select the **AWS Lambda Project (.NET Core – C#)** template and click **Next**.

8. Give your AWS Lambda a name. Let's call it VerifyPatientFunction.
   Keep the Location as is.

   Click the **Create** button.
   Choose some AWS template for this function (e.g., **Simple S3 Function**).
   Click the **Finish** button.

When done, you should notice that our HealthTracker solution now has one child project: the VerifyPatientFunction.



You can add additional projects to the HealthTracker solution (just like we added the VerifyLambdaFunction in step 6.

9. Let's continue with this example and add another project. The new project we will add is a console project. That is, a program that is usually used to read/write to/from the command window. In the process, we will demonstrate how to add references to some libraries (e.g., some AWS SDK DLLs that allows you to interact with AWS resources).

   In the Solution Explorer pane, right-click on HealthTracker and choose menu **Add → New Project…**. This brings the **Create a new project** back into view.

10. Enter C#, All platforms, Console
    Select the **Console App (.NET Core)** project template.

    Now the platform you choose and the type of project template become important. If you were to choose **Console App** (**.NET Framework)**, your application will only work on Windows. If, however, you choose **.NET Core**, your application will be platform-independent and should work on Windows, Linux, and macOS (Figure 3).

    For the remainder of this course, always use **.NET Core**.

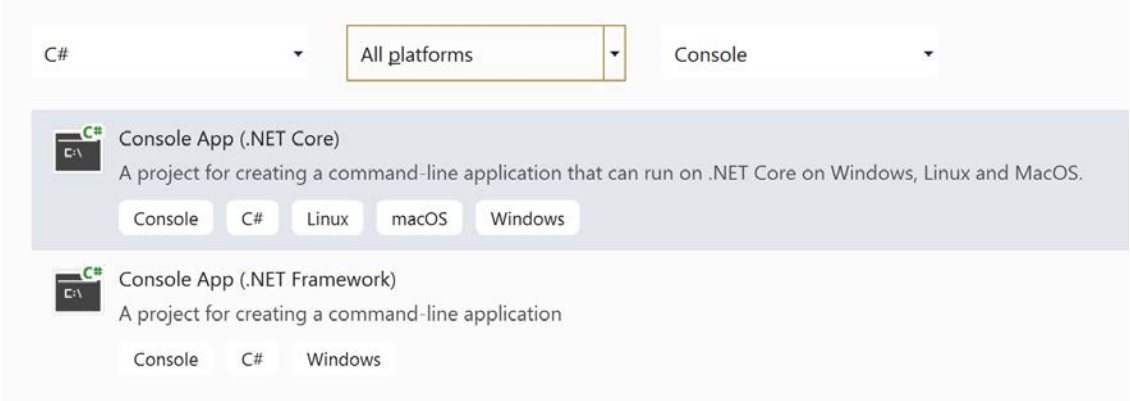


Figure 3: .NET Core vs. .NET Framework.

11. Select the **Console App (.NET Core)**, and click the **Next** button.

12. Give the project a name (e.g., AdministratorConsole) and leave the Location that Visual Studio determined (this is the location of the HealthTracker solution which is the correct place we want) (Figure 4).

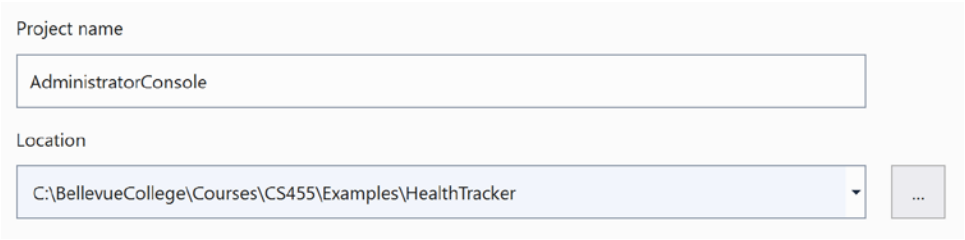


Figure 4: Creating a project under the solution.

13. Click the **Create** button. Visual Studio creates the AdministratorConsole project under the HealthTracker solution.

    You now have two different projects under the HealthTracker solution. You should now understand the concept of **solution** and **project** in Visual Studio. A solution has many projects. Different projects likely do different things (and that's why we created multiple ones) but they are all part of some product or solution to a problem.

14. Now assume that the AdministratorConsole needs to interact with some AWS resources. To do that it needs to use some AWS SDK. You usually need different libraries depending on what AWS service you need to interact with. Let's add references to two libraries: AWSSDK.Core and AWSSDK.S3.

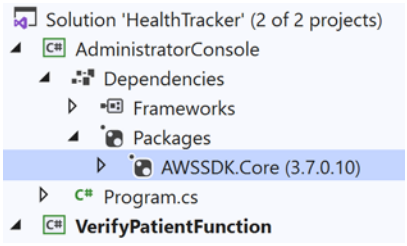
Add a reference to the AWSSDK.Core library:

In the AdministratorConsole project, right-click on **Dependencies** and choose menu **Manage NuGet Packages…**.

15. Make sure Browse is selected. Then type **AWSSDK.Core** in the search Textbox. This brings the AWSSDK.Core library into view.

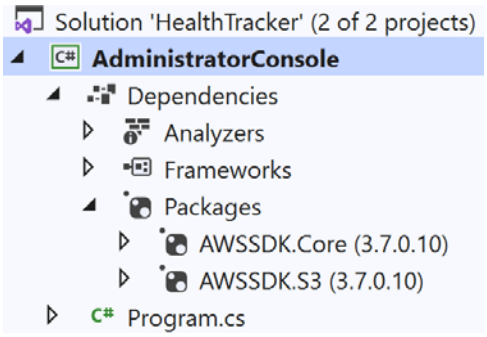
16. Select it. Then click the **Install** button.

17. Step 16 should install AWSSDK.Core and add it as a reference to the project. Expand Dependencies--Packages and you should see it listed in the Solution Explorer pane.

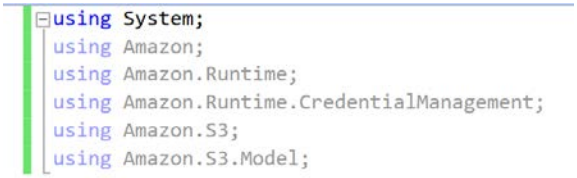


18. Adding a reference to the AWSSDK.S3 library:

Repeat steps 14 to 16, but now choose AWSSDK.S3.
You should now have references to both libraries:



19. Click file Program.cs to open it and enter these namespaces. The IDE code-completion should recognize the Amazon namespaces:



Summary

After completing this exercise, you should:

1. Know the difference between a Visual Studio **solution** and a visual studio **project**.
2. Know how to organize multiple projects into a solution.
3. Know how to create an AWS Lambda function template.
4. Know how to add libraries (e.g., AWS SDK libraries) to a given project.