

# Visual Studio IDE

Visual Studio 2015

[Other Versions](#)



- [Visual Studio 2013](#)

For the latest documentation on Visual Studio 2017 RC, see [Visual Studio 2017 RC Documentation](#).

Microsoft Visual Studio 2015 is a suite of tools for creating software, from the planning phase through UI design, coding, testing, debugging, analyzing code quality and performance, deploying to customers, and gathering telemetry on usage. These tools are designed to work together as seamlessly as possible, and are all exposed through the Visual Studio Integrated Development Environment (IDE).

You can use Visual Studio to create many kinds of applications, from simple store apps and games for mobile clients, to large, complex systems that power enterprises and data centers. You can create

1. apps and games that run not only on Windows, but also Android and iOS.
2. websites and web services based on ASP.NET, JQuery, AngularJS, and other popular frameworks
3. applications for platforms and devices as diverse as Azure, Office, Sharepoint, Hololens, Kinect, and Internet of Things, to name just a few examples
4. games and graphics-intensive applications for a variety of Windows devices, including Xbox, using DirectX.

Visual Studio by default provides support for C#, C and C++, JavaScript, F#, and Visual Basic. Visual Studio works and integrates well with third-party applications like Unity through the [Visual Studio Tools for Unity](#) extension and Apache Cordova through [Visual Studio Tools for Apache Cordova](#). You can extend Visual Studio yourself by creating custom tools that perform specialized tasks.

If you've never used Visual Studio before, learn the basics with our [Get Started Developing with Visual Studio](#) tutorials and walkthroughs.

If you want to find out about new features in Visual Studio 2015, see [What's New in Visual Studio 2015](#).

## Visual Studio Setup

---

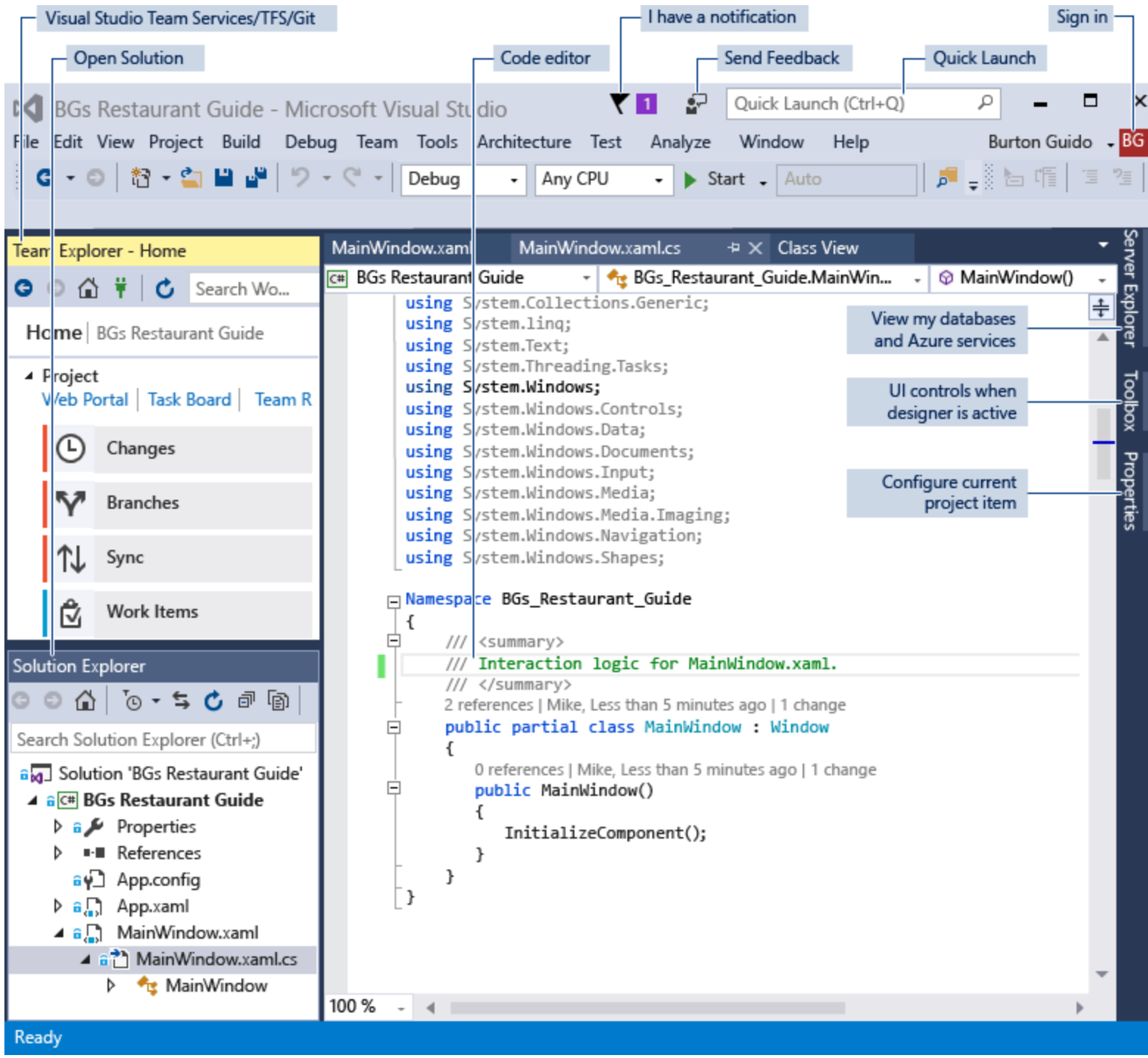
You can find out which edition of Visual Studio is right for you at [Visual Studio Editions](#).

You can install Visual Studio 2015 by downloading it from [Visual Studio Downloads](#). If you need to know more about the installation process, see [Installing Visual Studio 2015](#).

## IDE Basics

---

The following image shows the Visual Studio IDE with an open project, and the Solution Explorer window for navigating in the project files, and the Team Explorer window for navigating source control and work item tracking. The features in the title bar that are called out are explained below in more detail.



## Signing in

When you start Visual Studio for the first time, you can sign in using your Microsoft account, or your work or school account. Being signed in allows you to synchronize your settings, such as window layouts, across multiple devices and connect automatically to the services you might need, such as Azure subscriptions and Visual Studio Team Services. If you have a subscription-based license, you'll need to sign in to Visual Studio on a regular basis in order to keep your license token fresh. If you have a product key license, you don't have to sign in, but doing so makes it more convenient to connect to Visual Studio Team Services and your accounts with Azure, Office 365, Salesforce.com. For more information, see [Signing in to Visual Studio](#).

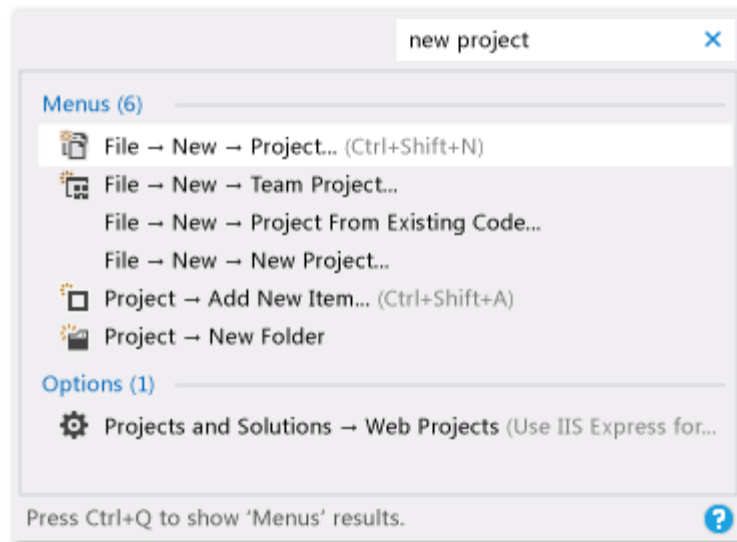
If you have multiple Visual Studio Team Services accounts, Azure accounts or MSDN subscriptions, you can link them and access resources and services in all your accounts with a single sign-in. For more information, see [Work with multiple user accounts](#).

## Staying up to date

The notification icon in the upper right of the title bar tells you when updates are available for Visual Studio or any related components that you have installed. You can choose whether to dismiss or act on these notifications. For more information, see [Visual Studio Notifications](#).

## Finding things and getting help

The [Quick Launch](#) window shown below is a fast way to find Visual Studio commands, tools, features, and so on when you don't know the keyboard shortcut or menu location. Just type what you are looking for and Quick Launch will give you a link to it.



MSDN is the Microsoft web site for technical documentation; you are reading this page on MSDN right now! In Visual Studio, you can press **F1** to go to the MSDN help page for the active window. You can also press **F1** in the code editor to go to the MSDN help page for the API or keyword at the current caret position. For example, in a C# file, place the caret somewhere in or just at the end of a `System.String` declaration, and press **F1** to go to the MSDN help page for [String](#).

## Giving feedback

It's easy to give us feedback on Visual Studio whenever you like. Click the feedback icon in the title bar next to **QuickLaunch** and then click on **Report a Problem** or **Provide a Suggestion**. Pre-release editions of Visual Studio also have a **Rate this Product** option. We look at all these comments and use them to improve the product. For more information, see [Talk to Us](#).

## Personalizing the IDE

You can customize the window layout to fit your development style. You can dock, float or hide any window at any time, and you also can run the editor in full-screen mode. You can create and save multiple custom window layouts that show only the windows you need for specific contexts. For example, you can create a full-screen layout so that all you see is the code editor. And you can create different layouts for debugging and for team operations. For more information, see [Customizing window layouts](#).

You can customize Visual Studio in many other ways, and roam your settings if you work on multiple machines. For more information, see [Personalizing the IDE](#).

There are keyboard shortcuts for almost everything, and you can customize them as well. To create new shortcuts, type "Keyboard" in Quick Launch to open the Keyboard dialog box. From there, you can press F1 to go to the MSDN help page if you need more information about the options. For more information, see [Default Keyboard Shortcuts in Visual Studio](#).

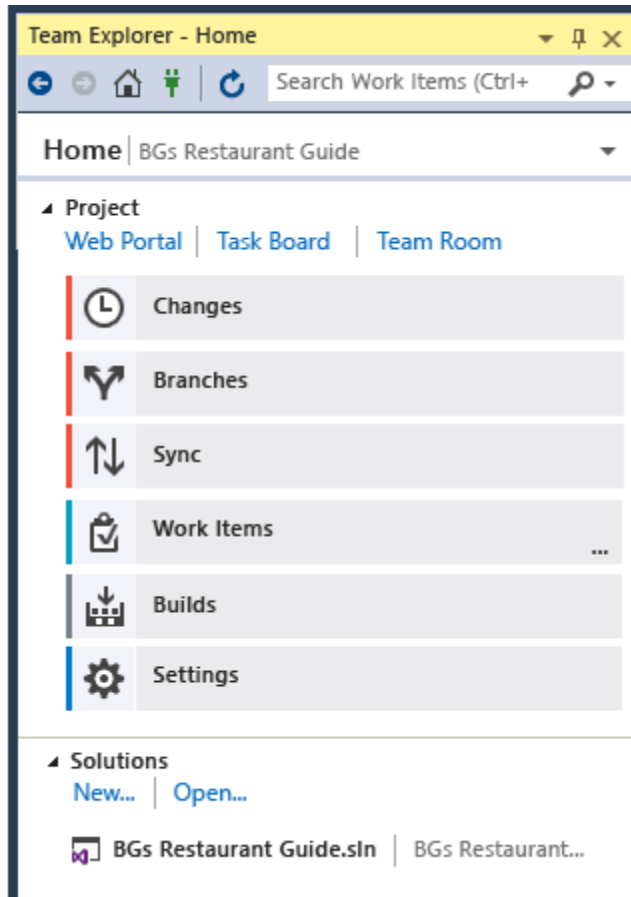
## Connecting to Visual Studio Team Services and Team Foundation Server

---

Visual Studio Team Services (VSTS) is a cloud-based service for hosting software projects and enabling collaboration in teams. VSTS supports both Git and Team Foundation Source Control systems, as well as Scrum, CMMI and Agile development methodologies. Team Foundation Version Control (TFVC) uses a single, centralized server repository to track and version files. Local changes are always checked in to the central server where other developers can get the latest changes. Team Foundation Server (TFS) 2015 is the application lifecycle management hub for Visual Studio. It enables everyone involved with the development process to participate using a single solution. TFS is useful for managing heterogeneous teams and projects, too.

If you have a Visual Studio Team Services account or a Team Foundation Server on your network, you connect to it through the Team Explorer window. From this window you can check code into or out of source control, manage work items, start builds, and access team rooms and workspaces. You can open Team Explorer from **Quick Launch** or on the main menu from **View | Team Explorer** or from **Team | Manage Connections**. For more information about Visual Studio Team Services, see [www.visualstudio.com](http://www.visualstudio.com). For more information about Team Foundation Server, see [Team Foundation Server](#).

The following image shows the Team Explorer pane for a solution that is hosted in VSTS:



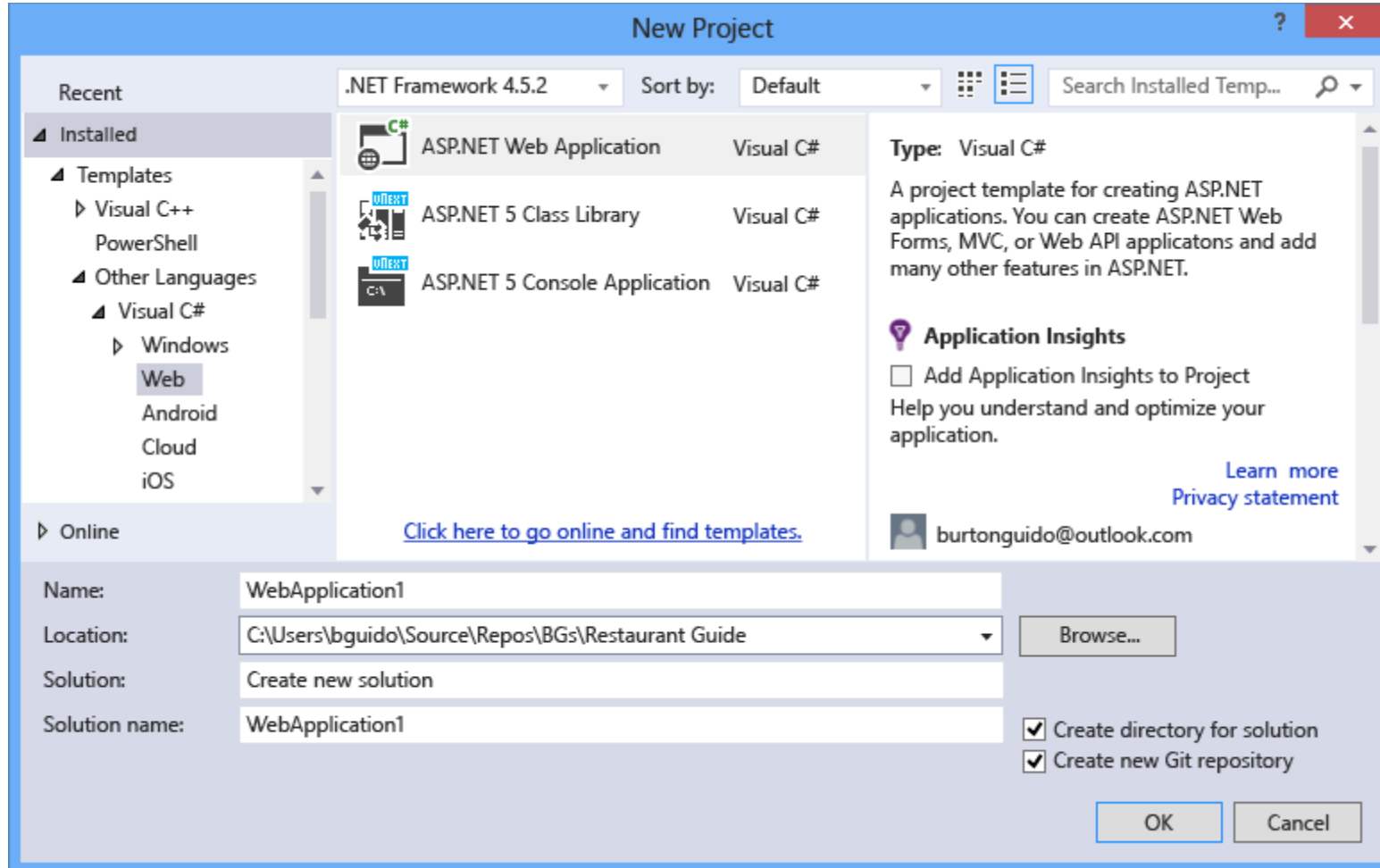
## Creating solutions and projects

---

Although you can use Visual Studio to browse individual code files, more commonly you will be working in a *project*. A Visual Studio project is a collection of files and resources that are compiled to a single binary executable file for applications (for example, an .exe, DLL, or appx). For non-ASP.NET websites, no executable is produced and the project contains only the HTML, JavaScript files, and images. Because sometimes you might need to create multiple binaries or websites that are closely related, Visual Studio has the concept of the Solution, which can contain multiple projects or websites. When you create a project, you are actually creating a project-in-a-solution, and you can add more projects to that solution later if you need to. For example, if you have a DLL project, you can add an .exe project to the solution that loads and consumes the DLL.

A *project template* is a collection of pre-populated code files and configuration settings that get you set up quickly to create a specific kind of application. Visual Studio comes with many project templates to choose from, and if none of the default templates work for you, you can create your own. After you create a project with a template, you can start writing your own code in it, either

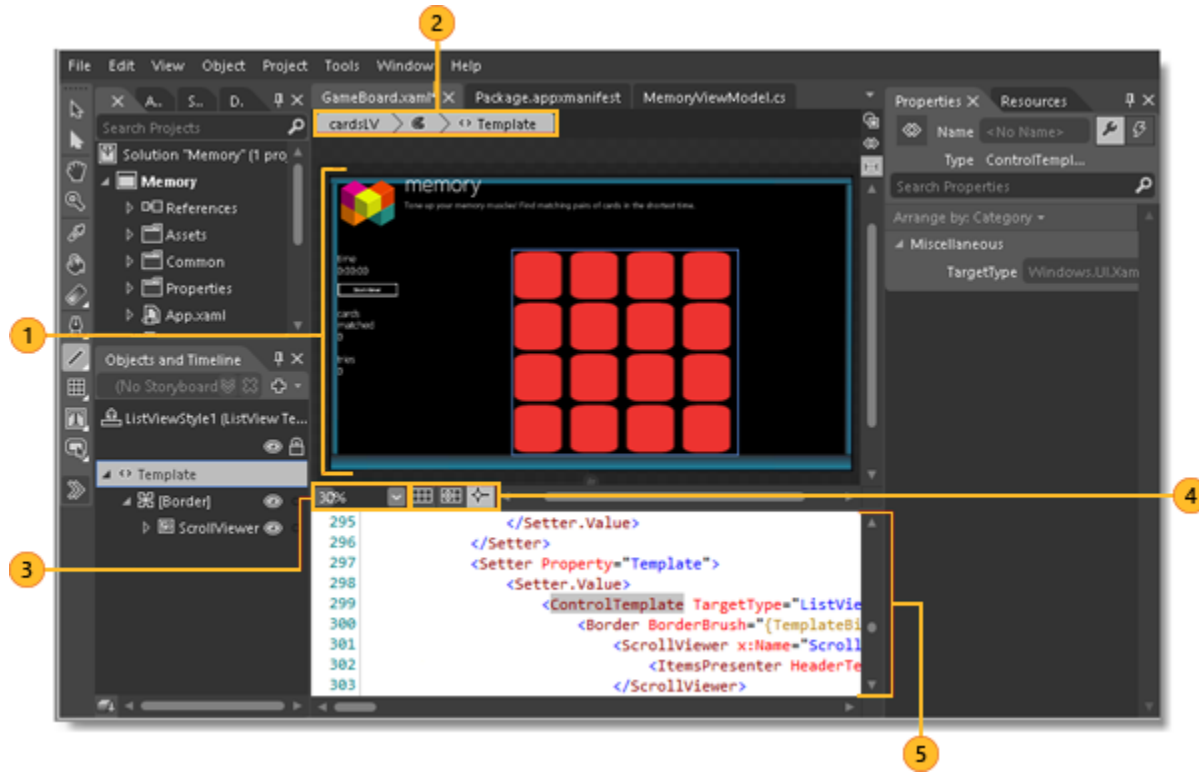
in the files provided or in new files you add. For more information, see [Solutions and Projects](#). The following illustration shows the New Project dialog with the project templates that are available for ASP.NET applications.



## Designing the user interface

A designer is an intuitive tool that enables you to create a user interface without writing code. You can drag UI controls such as list boxes, calendars, and buttons from the [Toolbox](#) window onto a design surface that represents the window or dialog box. You can resize and rearrange the elements without writing any code. Designers are included for any project type that has a user interface.

If your project has a XAML-based user interface, the default designer is Blend for Visual Studio, a sophisticated graphics tool that works seamlessly with Visual Studio.



- 1 **Design view** Displays the visual design of your document. In this view, you can draw or modify objects on the design surface.
- 2 **Breadcrumb** Quickly move between template-editing mode, style-editing mode, and object-editing scope for a selected object.
- 3 **Zoom** Use to zoom the design surface or objects on the design surface.
- 4 **Design surface controls** Use these controls (**Show snap grid**, **Snap to gridlines** and **Turn on or off snapping to snaplines**) to set snapping options. Snapping is useful for lining objects up to each other or to evenly-spaced lines on the design surface.
- 5 **Code editor** Edit your XAML, C#, C++ or Visual Basic code manually in the Code editor.

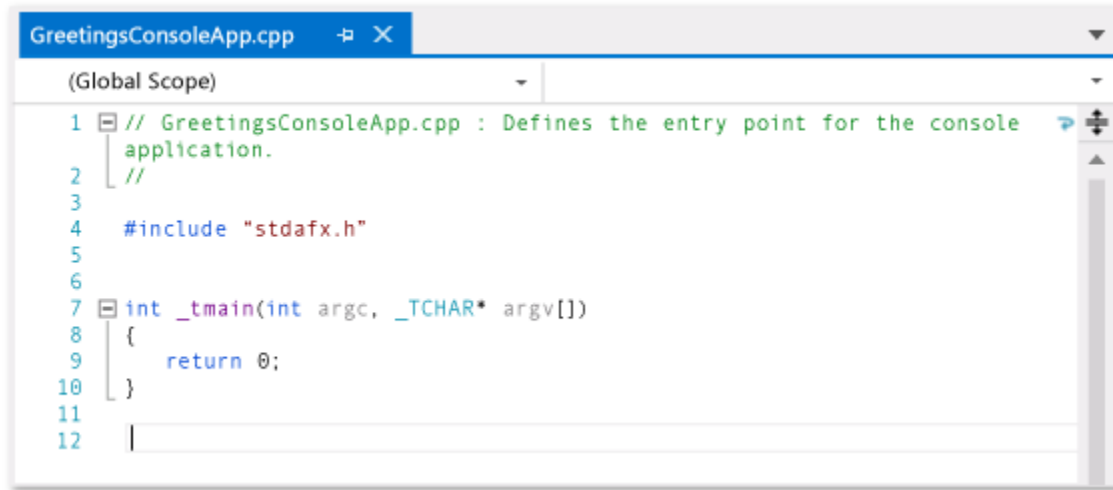
For more information, see [Designing XAML in Visual Studio and Blend for Visual Studio](#).

## Writing, navigating and understanding code



If you are a developer, the editor window is where you will probably spend most of your time. Visual Studio includes editors for C#, C++, Visual Basic, JavaScript, XML, HTML, CSS, and F#, and third parties offer plug-in editors (and compilers) for many other languages.

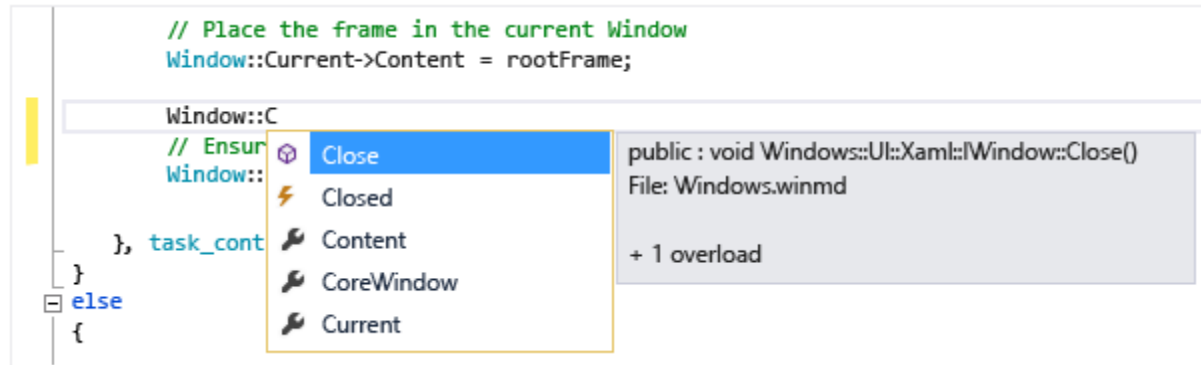
You can edit individual files in the text editor by clicking **File | Open | File**. To edit files in an open project, click on the file name in Solution Explorer. The code is colorized, and you can personalize the color scheme by typing “Colors” in Quick Launch. You can have lots of text editor tabbed windows open at once. You can split each window independently. You can also run the text editor in full-screen mode.



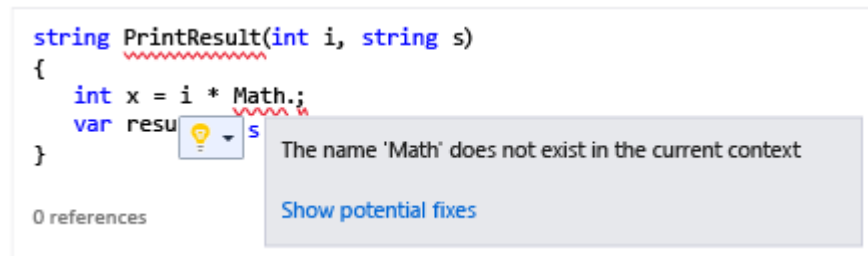
```
GreetingsConsoleApp.cpp
(Global Scope)
1 // GreetingsConsoleApp.cpp : Defines the entry point for the console
  application.
2 //
3
4 #include "stdafx.h"
5
6
7 int _tmain(int argc, _TCHAR* argv[])
8 {
9     return 0;
10 }
11
12
```

The text editor is highly interactive (if you want it to be) with many productivity features that help you write better code faster. The features vary by language, and you don't have to use any of them (Type "Editor" in Quick Launch) to turn features on or off: Some of the common productivity features are:

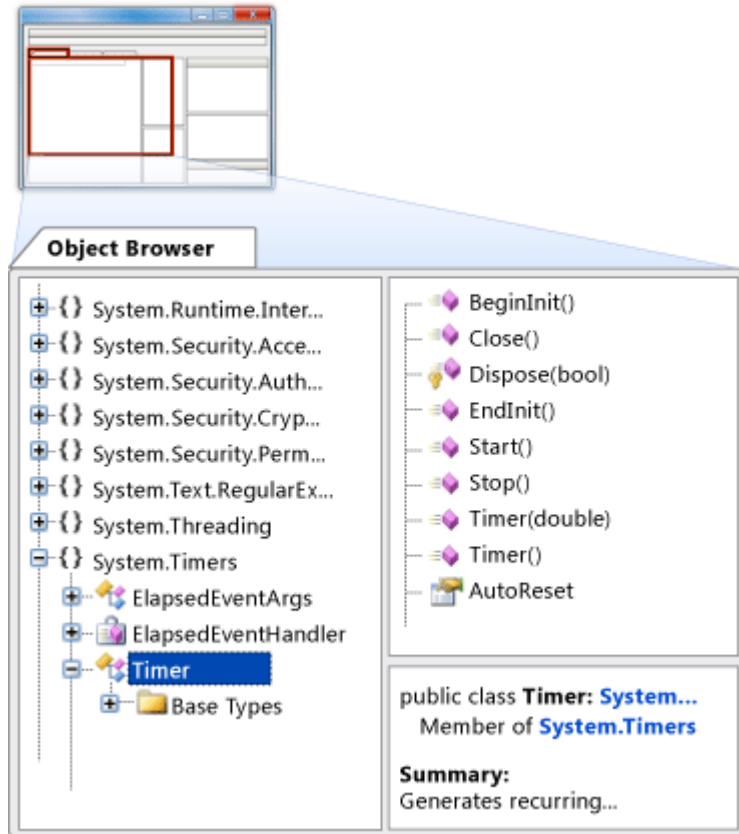
1. [Refactoring](#) includes operations such as intelligent renaming of variables, moving selected lines of code into a separate function, moving code to other locations, reordering function parameters, and more.
2. *IntelliSense* is an umbrella term for a set of popular features that display type information about your code directly in the editor and, in some cases, write small bits of code for you. It's like having basic documentation inline in the editor, which saves you from having to look up type information in a separate help window. IntelliSense features vary by language. For more information, see [Visual C# IntelliSense](#), [Visual C++ Intellisense](#), [JavaScript IntelliSense](#), [Visual Basic-Specific IntelliSense](#). The following illustration shows some IntelliSense features at work:



3. **Squiggles** alert you to errors or potential problems in your code in real time as you type, which enables you to fix them immediately without waiting for the error to be discovered during compilation or run time. If you hover over the squiggle, you see additional information about the error. A light bulb may also appear in the left margin with suggestions for how to fix the error. For more information, see [Perform quick actions with light bulbs](#).



4. [Bookmarks](#) enable you to navigate quickly to specific lines in files that you are actively working on.
5. The [Call Hierarchy](#) window can be invoked in the text editor context menu to show the methods that call, and are called by, the method under the caret.
6. **Code Lens** enables you to find references and changes to your code, linked bugs, work items, code reviews, and unit tests, all without leaving the editor. For more information, see [Find code changes and other history](#).
7. The **Peek to Definition** window shows a method or type definition inline, without navigating away from your current context. This window now works for XAML, too.
8. The **Go To Definition** context menu option takes you directly to the place where the function or object is defined. Other navigation commands are also available by right-clicking in the editor.
9. A related tool, the [Object Browser](#), enables you to inspect .NET or Windows Runtime assemblies on your system to see what types they contain and what methods and properties those types contain.



Most of the items on the Edit menu and View menu relate to the code editor in some way. For more information about the editor, see [Writing Code](#) and [Editing Your Code](#).

## Compiling and building your code

---

To build a project means to compile the source code and perform whatever steps are necessary to produce the executable. Different languages have different build operations, and regular websites don't build at all. Regardless of the project type, the Build menu is the standard location for these commands. To compile and run your code with a single keystroke, press F5. Every compiler is completely configurable through the IDE. The Build toolbar enables you to specify whether to build a debug version of your program, with symbols and extra error checking enabled to support breakpoints and single stepping in the debugger, or a release build, which is what you will ultimately give to customers. You can configure more build settings and many other settings on the property page for a project. Right-click on the project node in Solution Explorer and choose Properties. You can also run builds from the command line.

The output from the build, including an error or success messages, appear in the Output Window. The Error List window (shown below) gives detailed information on build errors.

```
static void Main (string[] args)
{
    Console.WriteLine("Hello world");
}
}
```

100 %

Error List

Entire Solution 1 Error 0 Warnings 0 Messages Build Only Search Error L...

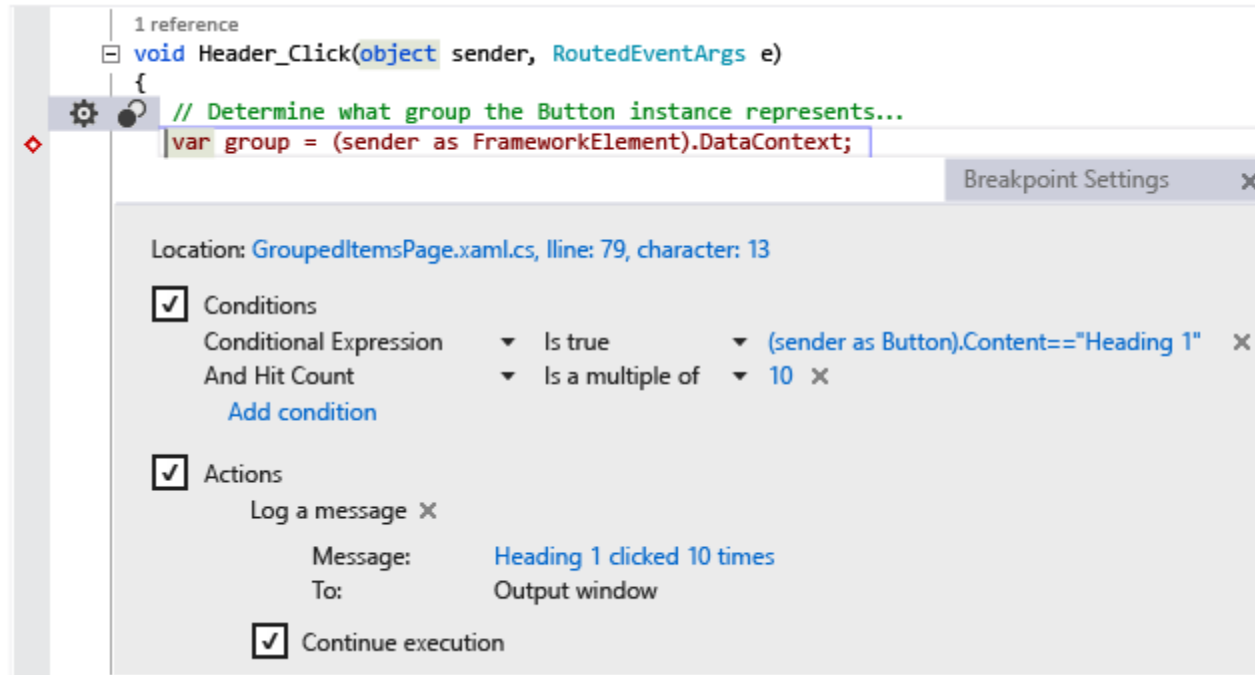
Code	Description	Project	File	Line	Tool	Supp...
CS1026	) expected	ConsoleApplication2	Program.cs	13	Compiler	

Error List Output

## Debugging your code

---

Visual Studio's state-of-the-art debugger enables you to debug code running in your local project, on a remote device, or on an emulator such as the ones for Android or Windows Phone. You can step through code one statement at a time and inspect variables as you go, you can step through multi-threaded applications, and you can set breakpoints that are only hit when a specified condition is true. All of this can be configured in the code editor itself, so that you don't have to leave the context of your code.



The debugger itself has multiple windows that enable you to view and manipulate local variables, the call stack, and other aspects of the runtime environment. You can find these windows on the **Debug** menu.

The [Immediate Window](#) enables you to type in an expression and see its result immediately.

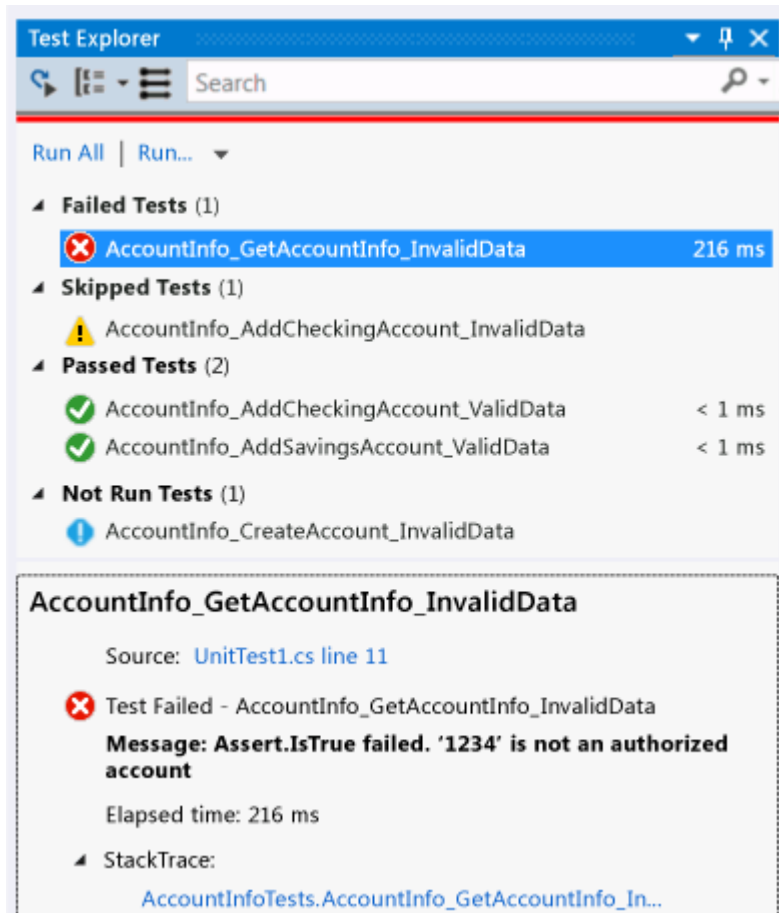
The [IntelliTrace](#) window records each method call and other events in a running .NET program and can help you to quickly locate where a problem originates.

For more information, see [Debugging in Visual Studio](#).

## Testing your code

---

Visual Studio includes a unit test framework for managed code (.NET) and one for native C++. To create unit tests, simply add a Test Project to your solution, write your tests, and then run them from the Test Explorer window. For more information, see [Unit Test Your Code](#).



## Analyzing code quality and performance

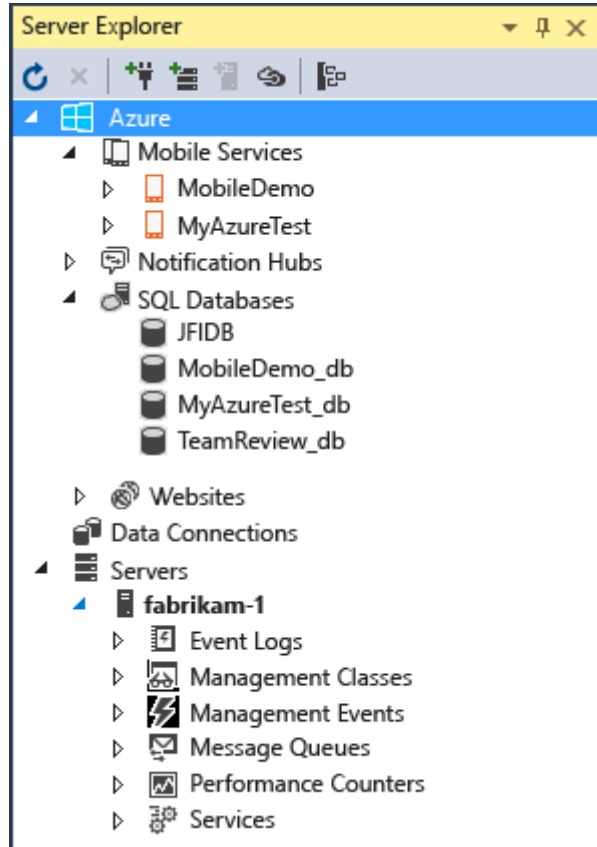
---

Visual Studio includes powerful tools for static and runtime analysis. The static analysis tools help you identify potential errors in design, globalization, interoperability, performance, security, and other categories. Performance testing, or profiling, involves measuring how your program runs. You access these tools from the **Analyze** menu. For more information, see [Improving Quality with Visual Studio Diagnostic Tools](#).

## Connecting to cloud services and databases

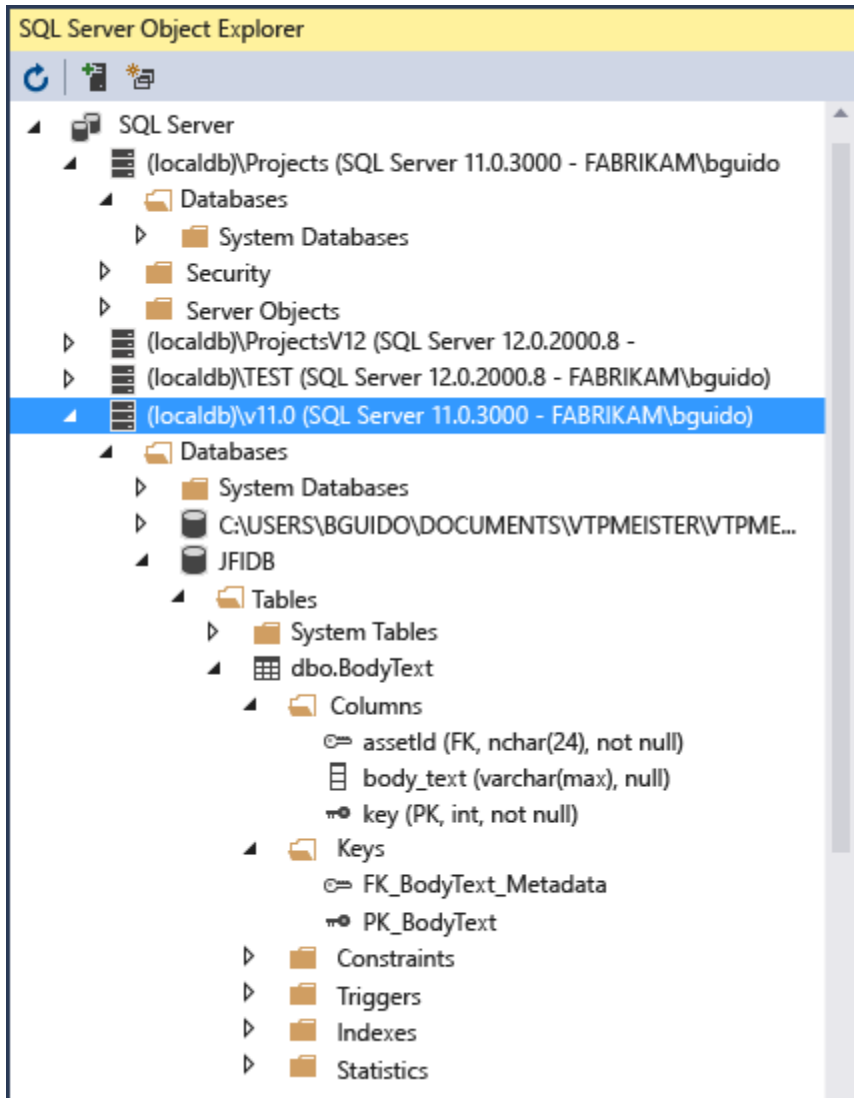
---

The [Server Explorer](#) window in Visual Studio shows the resources in all the accounts managed under your personalization account (the one you logged in with), including SQL Server instances, Azure, Salesforce.com, Office 365, and websites.



Visual Studio includes [Microsoft SQL Server Data Tools](#) (SSDT), which enable you to build, debug, maintain, and refactor databases. You can work with a database project, or directly with a connected database instance on- or off-premises.

The SQL Server Object Explorer in Visual Studio offers a view of your database objects similar to SQL Server Management Studio. SQL Server Object Explorer allows you to do light-duty database administration and design work, including editing table data, comparing schemas, and executing queries by using contextual menus right from the SQL Server Object Explorer. SSDT also includes special project types and tools for developing SQL Server 2012 Analysis Services, Reporting Services, and Integration Services Business Intelligence (BI) solutions (formerly known as Business Intelligence Development Studio).



## Deploying your finished application

When your application is ready to deploy to customers, Visual Studio provides the tools to do that, whether you're deploying to the Windows Store, to a SharePoint site, or with InstallShield or Windows Installer technologies. It's all accessible via the IDE. For more information, see [Deploying Applications, Services, and Components](#).



## Architecture and modeling tools (Enterprise only)

---

You can use Visual Studio architecture and modeling tools to design and model your app. These tools help you to visualize the code's structure, behavior, and relationships. You can create models at different levels of detail throughout the application lifecycle as part of your development process. You can track requirements, tasks, test cases, bugs, and other work associated with your models by linking model elements to Team Foundation Server work items and your development plan. For more information, see [Design and model your app](#).

## Extending Visual Studio through the Visual Studio SDK

---

Visual Studio is an extensible platform. A Visual Studio extension is a custom tool that integrates with the IDE. You can add third-party extensions or create your own. For more information, see [Developing Visual Studio Extensions](#).

The [Visual Studio User Experience Guidelines](#) are an essential reference for anyone writing extensions for Visual Studio. These platform-specific guidelines include information on dialog design, fonts, colors, icons, common controls, and other interaction patterns that will make your new feature integrate seamlessly with Visual Studio.

## In this Guide

---

[User Accounts and Updates](#)

[How to: Move Around in the IDE](#)

[Finding and Using Visual Studio Extensions](#)

[Writing Code](#)

[Profiling Tools](#)

[Designing User Interfaces](#)

[Compiling and Building](#)

[Visual Studio IDE 64-Bit Support](#)

[Visual Studio Samples](#)

[Globalizing and Localizing Applications](#)

[Personalizing the IDE](#)

[Get Started Developing with Visual Studio](#)

[Solutions and Projects](#)

[Debugging in Visual Studio](#)

[Improve Code Quality](#)

[Analyzing and Modeling Architecture](#)

[Deploying Applications, Services, and Components](#)

[Security](#)

[Microsoft Help Viewer](#)

[Reference](#)

## See Also

---

[Installing Visual Studio 2015](#)

[Editing Your Code](#)

[What's New in Visual Studio 2015](#)

[Porting, Migrating, and Upgrading Visual Studio Projects](#)

[Talk to Us](#)

# VISUAL STUDIO 2017

# DATE 2017-03-08

[Skip to main content](#)



Docs

- Technologies
  - Windows
    - [Apps](#)
    - [Internet of Things](#)
    - [Mixed Reality](#)
    - [Microsoft Edge](#)

- [Hardware](#)
- [IT Center](#)
- Microsoft Azure
  - [What is Azure](#)
  - [Products](#)
  - [Solutions](#)
  - [Pricing](#)
  - [Create a free account](#)
- Visual Studio
  - [Visual Studio](#)
  - [Visual Studio IDE](#)
  - [Visual Studio Team Services](#)
  - [Visual Studio Code](#)
  - [Xamarin](#)
  - [Visual Studio Dev Essentials](#)
  - [Subscriptions](#)
- Office
  - [Office Dev Center](#)
  - [Office 365 for IT pros](#)
- [Microsoft Graph](#)
- Services
  - [Store](#)
  - [Cortana](#)

- [Bing](#)
- [Application Insights](#)
- Documentation
  - [Microsoft Developer Network](#)
  - [TechNet](#)
  - Platforms
    - [Microsoft Azure](#)
    - [Microsoft Graph](#)
    - [Visual Studio](#)
    - [Visual Studio Team Services](#)
    - [Windows](#)
    - [Office](#)
    - [All Developer Centers](#)
  - IT TechCenters
    - [Windows IT Center](#)
    - [Office 365 for IT Pros](#)
    - [All IT TechCenters](#)
  - Downloads
    - [Microsoft Download Center](#)
    - [Microsoft Azure](#)
    - [Visual Studio](#)
    - [SDKs](#)
    - [Windows](#)

- Code samples
  - [Office](#)
  - [Microsoft Graph](#)
  - [MSDN](#)
  - [Azure](#)
  - [Windows](#)
- [PowerShell scripts](#)
- Resources
- Blogs
  - [Microsoft Azure](#)
  - [Visual Studio](#)
  - [Visual Studio Team Services](#)
  - [Developer tools](#)
  - [Office Dev Blog](#)
  - [Server & management](#)
  - [Windows](#)
- Community & Forums
  - [Developers](#)
  - [TechNet](#)
  - [Microsoft Tech Community](#)
- Subscriptions
  - [MSDN subscriptions](#)
  - [IT Pro Cloud Essentials](#)

- Training
  - [Microsoft Virtual Academy](#)
  - [IT Pro Career Center](#)
- [TechNet Evaluation Center](#)
- [Channel 9](#)

Search

- 
- No results

VS IDE

[Docs](#) <https://docs.microsoft.com> [Windows](#) [Microsoft Azure](#) [Visual Studio](#) [Office](#)

More

- [.NET](#)
- [ASP.NET](#)
- [Dynamics 365](#)
- [Enterprise Mobility + Security](#)
- [NuGet](#)
- [SQL Server](#)
- [Xamarin](#)
  
- [Docs](#)
- [Visual Studio](#)
- [Documentation](#)





- [IDE](#)
- Comments
- Edit
- Share
- |
- Theme

### **In this article**

1. [What can you do with the Visual Studio IDE?](#)
2. [Install the Visual Studio IDE](#)
3. [Sign in](#)
4. [Create a program](#)
5. [Debug, test, and improve your code](#)
6. [Deploy your finished application](#)
7. [Quick tour of the IDE](#)
8. [Collaborate with others and control your source code](#)
9. [Connect to services, databases, and cloud-based resources](#)
10. [Extend Visual Studio](#)
11. [Learn more and find out what's new](#)
12. [See also](#)

# Visual Studio IDE feature tour

3/7/2017 14 min to read Contributors

- 
- 
- 
- 



- 
- [all](#)

## In this article

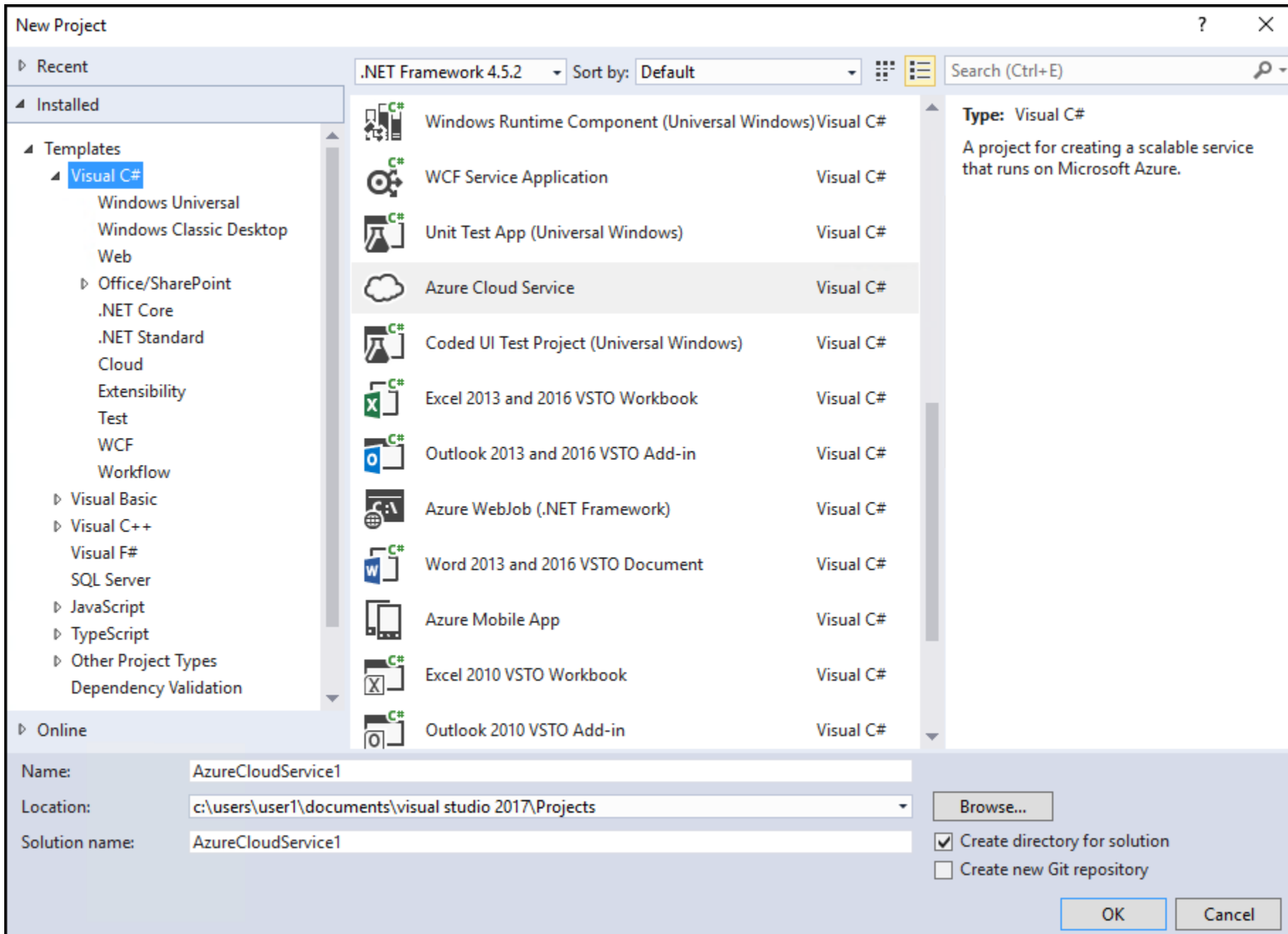


1. [What can you do with the Visual Studio IDE?](#)
  2. [Install the Visual Studio IDE](#)
  3. [Sign in](#)
  4. [Create a program](#)
  5. [Debug, test, and improve your code](#)
  6. [Deploy your finished application](#)
  7. [Quick tour of the IDE](#)
  8. [Collaborate with others and control your source code](#)
  9. [Connect to services, databases, and cloud-based resources](#)
  10. [Extend Visual Studio](#)
  11. [Learn more and find out what's new](#)
  12. [See also](#)
- +

This topic introduces you to the features of the Visual Studio IDE. The Visual Studio IDE is an interactive development environment (IDE); a creative launching pad that you can use to view and edit nearly any kind of code, and then debug, build, and publish apps for Android, iOS, Windows, the web, and the cloud. There are versions available for Mac and Windows. We'll walk through some things you can do with Visual Studio and how to install and use it, walk through creating a simple project, get pointers on debugging and deploying code, and take a tour of the various tool windows.+

## **What can you do with the Visual Studio IDE?**

Do you want to create an app for an Android phone? You can do that. How about create a cutting edge game using C++? You can do that too and much, much more. Visual Studio provides templates that help you make websites, games, desktop apps, mobile apps, apps for Office, and more.+



Or, you can simply open nearly any code you get from almost anywhere and get working. See a project on GitHub that you like? Just clone the repository, open it in Visual Studio, and start coding!+

### **Create mobile apps**

You can create native mobile apps for different platforms by using Visual C# and Xamarin, or Visual C++, or hybrid apps using JavaScript with Apache Cordova. You can write mobile games for Unity, Unreal, DirectX, Cocos, and more. Visual Studio includes an Android emulator to help you run and debug Android apps.+

You can leverage the power of the cloud for your mobile apps by creating Azure app services. Azure app services enable your apps to store data on the cloud, securely authenticate users, and automatically scale its resources up or down to accommodate the needs of your app and your business. To learn more, see [Mobile App Development](#).+

### **Create cloud apps for Azure**

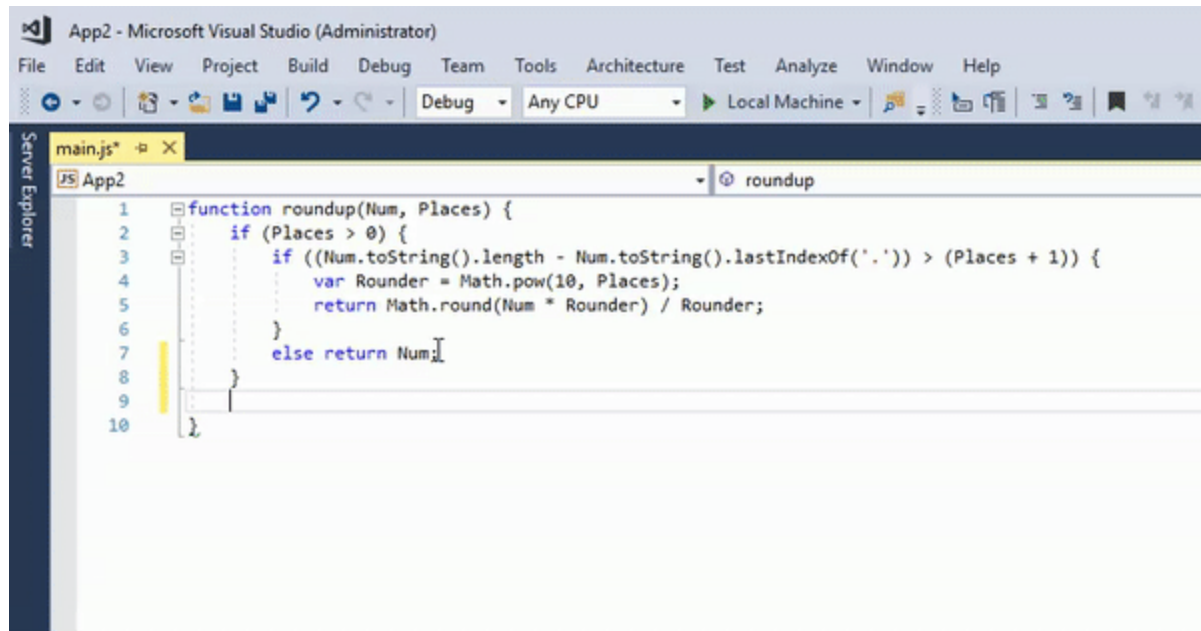
Visual Studio offers a suite of tools that enable you to easily create cloud-enabled applications powered by Microsoft Azure. You can configure, build, debug, package, and deploy applications and services on Microsoft Azure directly from the IDE. Leverage Azure services for your apps using Connected Services. To get Azure Tools for .NET, select the **Azure development** workload when you install Visual Studio. For more information, see [Visual Studio Tools for Azure](#).+

### **Create apps for the web**

The web drives our modern world, and Visual Studio can help you write apps for it. You can create web apps using ASP.NET, Node.js, Python, JavaScript and TypeScript. Visual Studio understands web frameworks like Angular, jQuery, Express, and more. ASP.NET Core and .NET Core run on Windows, Mac, and Linux operating systems. For more information, see [Modern Web Tooling](#).+

### **Write code in a world class editing environment**

Visual Studio helps you write code quickly and easily through features such as syntax colorization, statement completion, IntelliSense (pop-up descriptions of the selected code element), code outlining, setting breakpoints for debugging, and much more.+



The screenshot shows the Microsoft Visual Studio IDE. The title bar reads 'App2 - Microsoft Visual Studio (Administrator)'. The menu bar includes File, Edit, View, Project, Build, Debug, Team, Tools, Architecture, Test, Analyze, Window, and Help. The toolbar shows various icons for file operations and debugging. The 'Server Explorer' pane on the left shows a project named 'App2'. The main editor window displays a JavaScript file named 'main.js' with the following code:

```
1 function roundup(Num, Places) {
2     if (Places > 0) {
3         if ((Num.toString().length - Num.toString().lastIndexOf('.') > (Places + 1)) {
4             var Rounder = Math.pow(10, Places);
5             return Math.round(Num * Rounder) / Rounder;
6         }
7         else return Num;
8     }
9 }
10
```

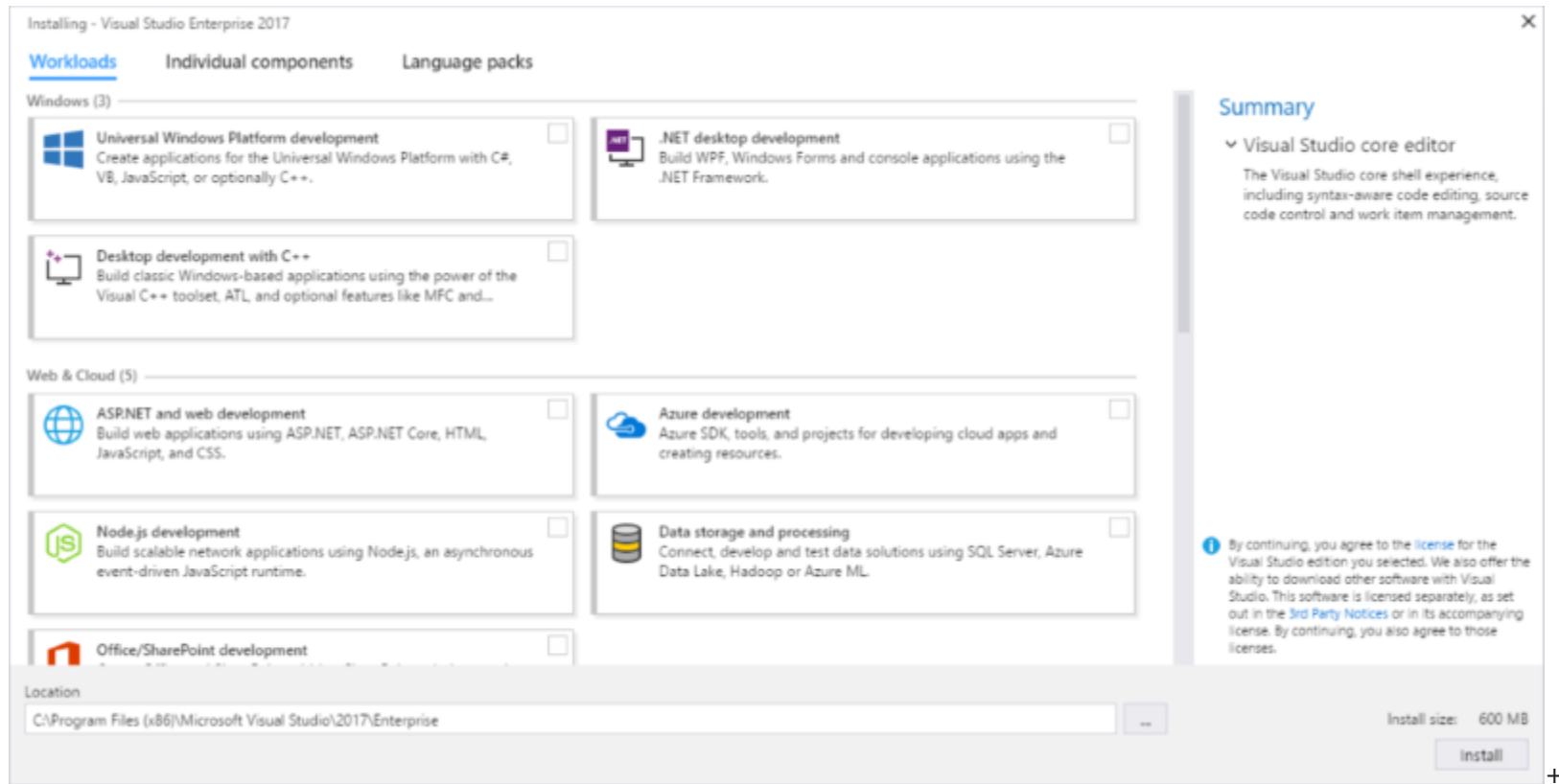
To learn more, see [Writing Code in the Code and Text Editor](#).

Visual Studio can do help you do many more things. For a more complete list, see [Visual Studio IDE](#).

## Install the Visual Studio IDE

To get started, download Visual Studio and install it on your system. You can download it at [Visual Studio 2017](#).

Visual Studio is now more lightweight than ever! The new modular installer enables you to choose and install *workloads*, which are groups of features needed for the programming language or platform you prefer. This strategy helps keep the footprint of the Visual Studio installation smaller than ever before, which means it installs and updates faster too.



In addition to improved installation performance, many improvements have been made in Visual Studio 2017 to improve overall IDE start-up and solution load time. For example, selecting the new Lightweight Solution Load feature, located on the main menu under **Tools, Options, Projects and Solutions**, enables larger solutions to load faster.

To learn more about setting up Visual Studio on your system, see [Install Visual Studio 2017](#).

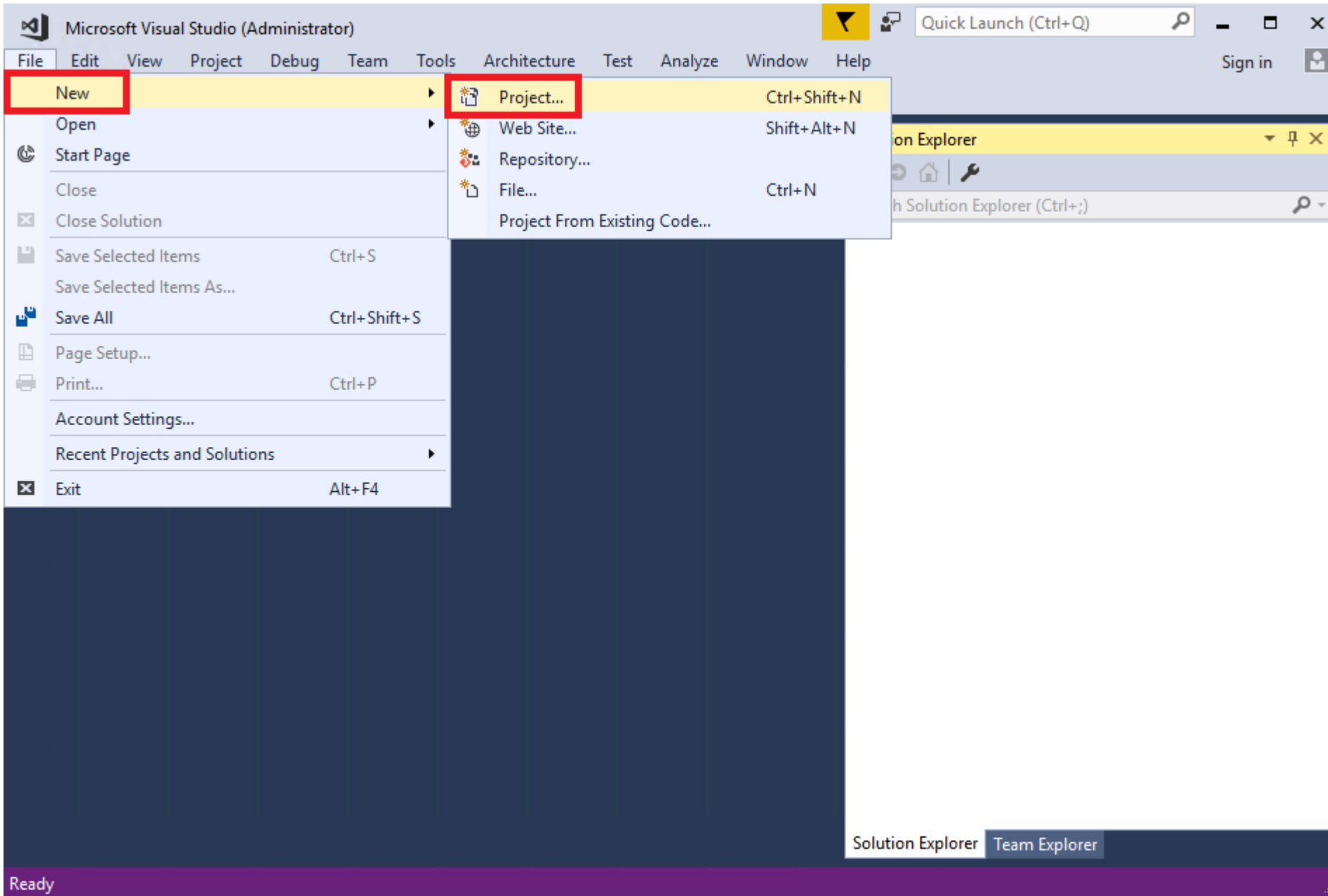
## Sign in

When you start Visual Studio for the first time, you can optionally sign in using your Microsoft account, or your work or school account. Being signed in lets you synchronize Visual Studio settings, such as window layouts, across multiple devices. It also connects you automatically to the services you might need, such as Azure subscriptions and Visual Studio Team Services.

## Create a program

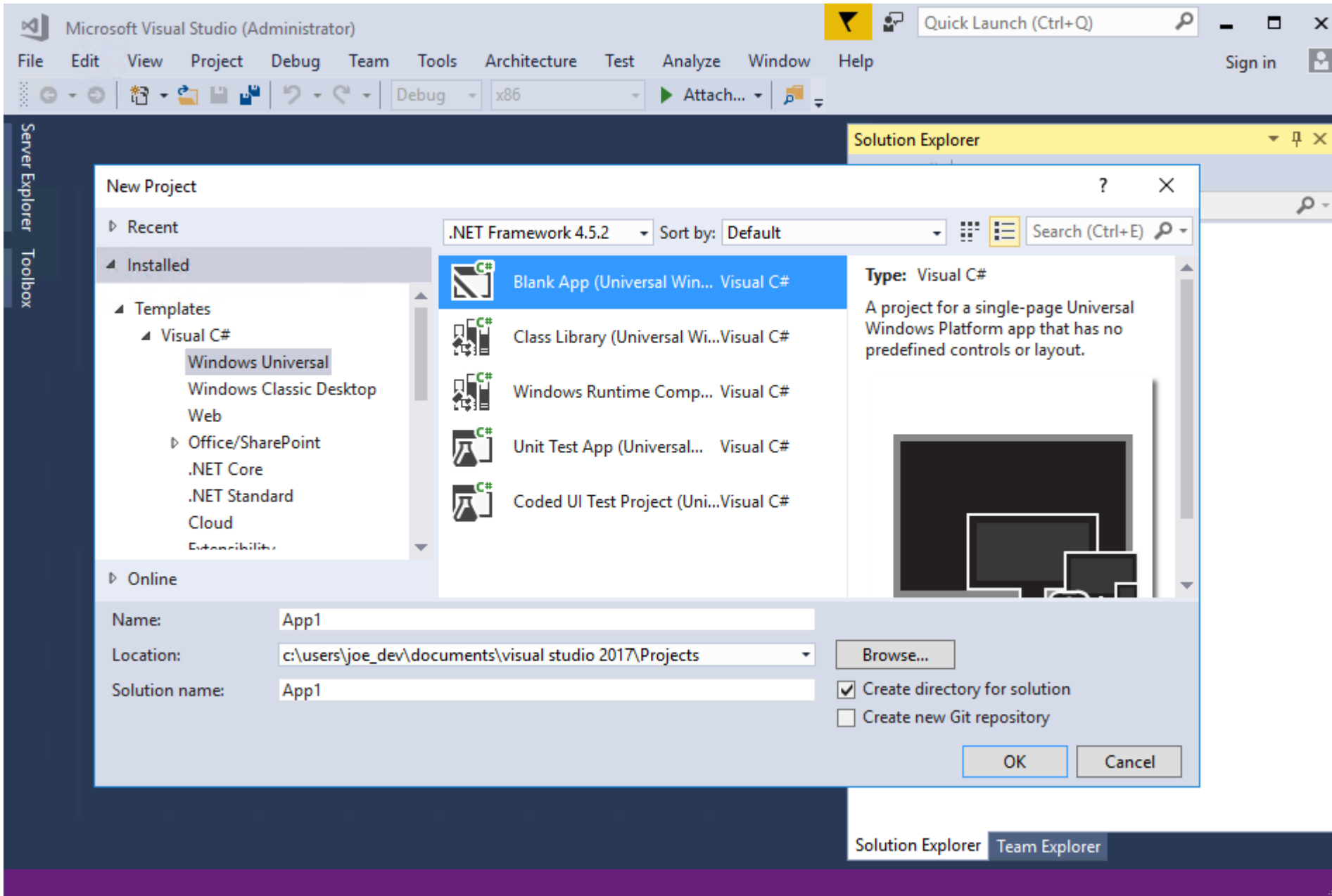
One good way to learn about something is to use it! Let's dive in and create a new, simple program. +

1. Open Visual Studio. On the menu, choose **File, New, Project**. (Use the default project values.)





2. The **New Project** dialog box shows several project templates. Choose the **Windows Universal** category under **Visual C#**, choose the **Blank App (Universal Windows)** template, and then choose the **OK** button.



This creates a new blank Universal Windows app project using Visual C# and XAML as the programming languages. Wait for a bit while Visual Studio sets up the project for you. If you are prompted for any information, just accept the default values for now.

3. Shortly, you should see something like the following screenshot. Your project files are listed on the right side in a window called Solution Explorer.

App1 - Microsoft Visual Studio (Administrator) Quick Launch (Ctrl+Q)

File Edit View Project Build Debug Team Tools Architecture Test Analyze Window Help Sign in

Debug x86 Local Machine

Server Explorer Toolbox

MainPage.xaml.cs MainPage.xaml

C# App1 App1.MainPage MainPage()

```
7 using Windows.Foundation.Collections;
8 using Windows.UI.Xaml;
9 using Windows.UI.Xaml.Controls;
10 using Windows.UI.Xaml.Controls.Primitives;
11 using Windows.UI.Xaml.Data;
12 using Windows.UI.Xaml.Input;
13 using Windows.UI.Xaml.Media;
14 using Windows.UI.Xaml.Navigation;
15
16 // The Blank Page item template is documented at https://go.microsoft.com/fwlink/?LinkId=391641
17
18 namespace App1
19 {
20     /// <summary>
21     /// An empty page that can be used on its own or navigated to within a
22     /// Frame.
23     /// </summary>
24     public sealed partial class MainPage : Page
25     {
26         0 references
27         public MainPage()
28         {
29             this.InitializeComponent();
30         }
31     }
}
```

Solution Explorer

Search Solution Explorer (Ctrl+;)

- Solution 'App1' (1 project)
- App1 (Universal Windows)
- Connected Services
- Properties
- References
- Assets
- App.xaml
- App1\_TemporaryKey.pfx
- MainPage.xaml
- MainPage.xaml.cs
- Package.appxmanifest

Solution Explorer Team Explorer

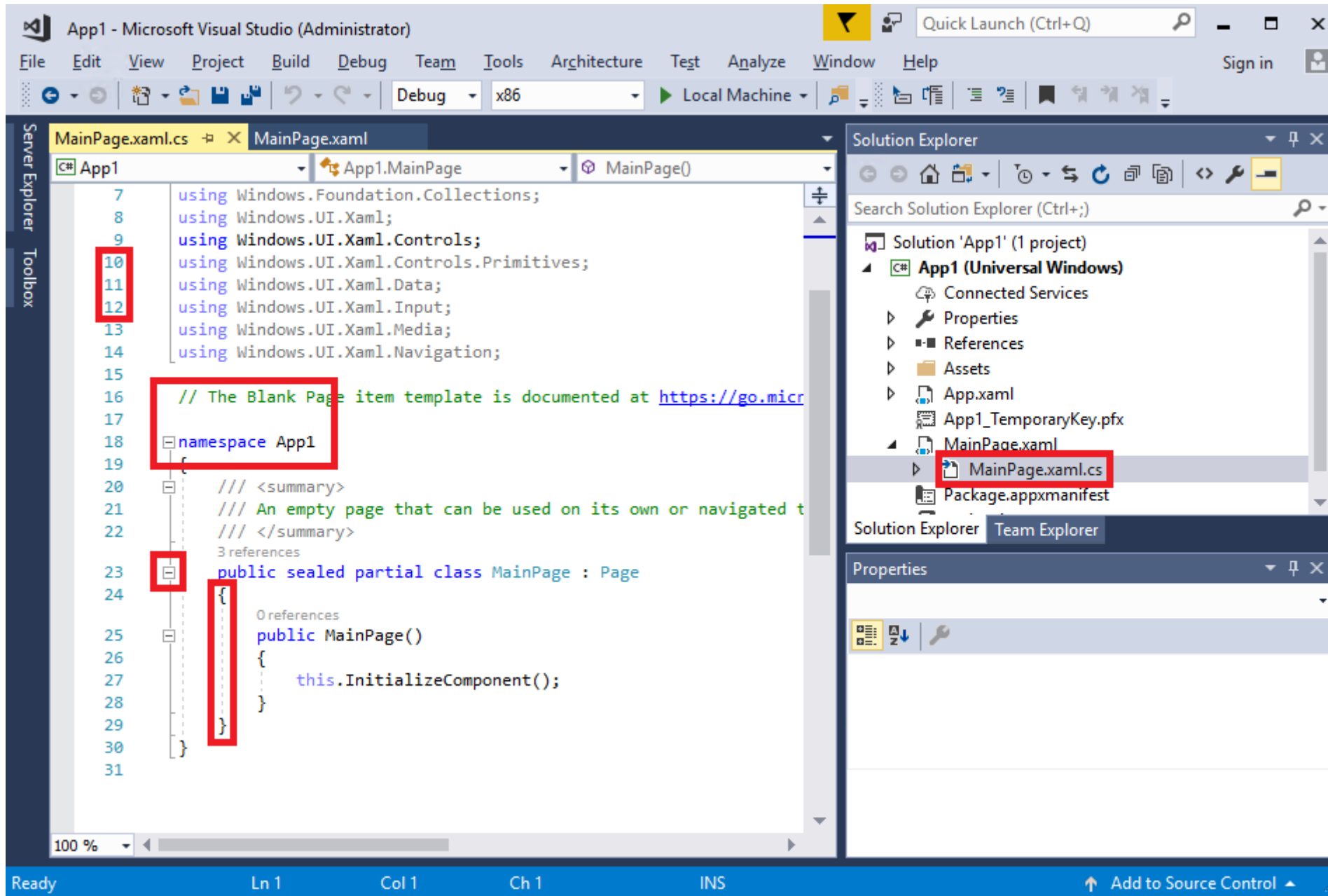
Properties

100 %

Ready Ln 1 Col 1 Ch 1 INS Add to Source Control

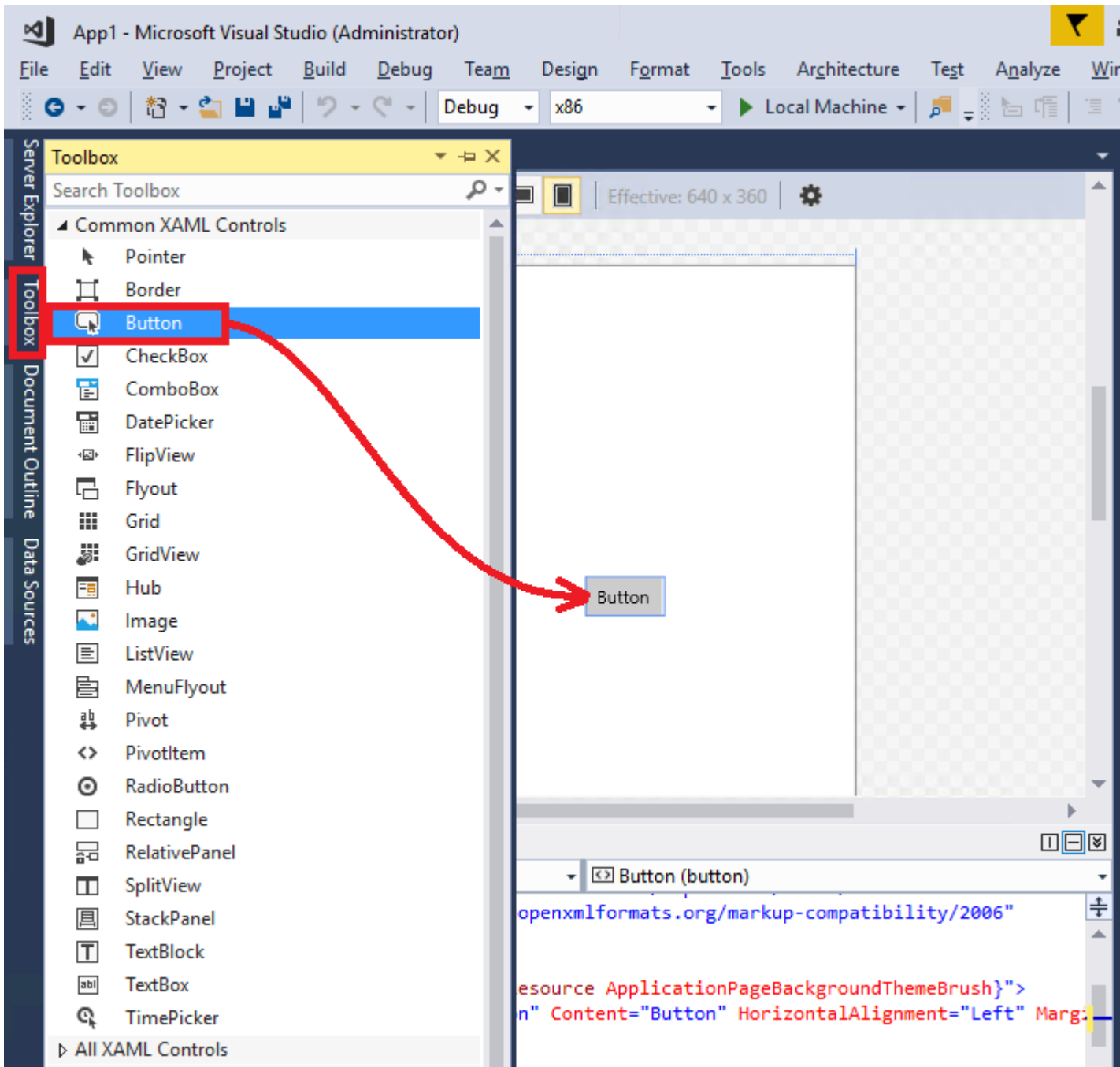
4. In Solution Explorer, choose the little black triangle next to the MainPage.xaml file to expand it, and you should see a MainPage.xaml.cs file underneath. Choose this file (which contains C# code) to open it.

The C# code in MainPage.xaml.cs appears in the code editor on the left side of the screen. Notice that the code syntax is automatically colorized to indicate different types of code, such as statements or comments. In addition, small, vertical dashed lines in the code indicate which braces match one another, and line numbers help you locate code later. You can choose the small, boxed minus signs to collapse or expand code. This code outlining feature lets you hide code you don't need, helping to minimize onscreen clutter.



There are other menus and tool windows available, but let's move on for now.

5. Add a button to the XAML form to give users a way to interact with your app. To do this, open the MainPage.xaml file. This shows a split view: a designer above, for visually placing controls, and a code view below, which shows the XAML code behind the designer. When you run the program later, what you see in the designer becomes a window that users will see, a "form," and the underlying XAML determines what appears on the form.
6. On the left side of the screen, choose the **Toolbox** tab to open the Toolbox. The Toolbox contains a number of visual controls that you can add to forms. For now, we'll just add a button control.
7. Expand the **Common XAML Controls** section and then drag the Button control out to about the middle of the form. (The exact location doesn't matter.)





When you're done, you should see something similar to the following.

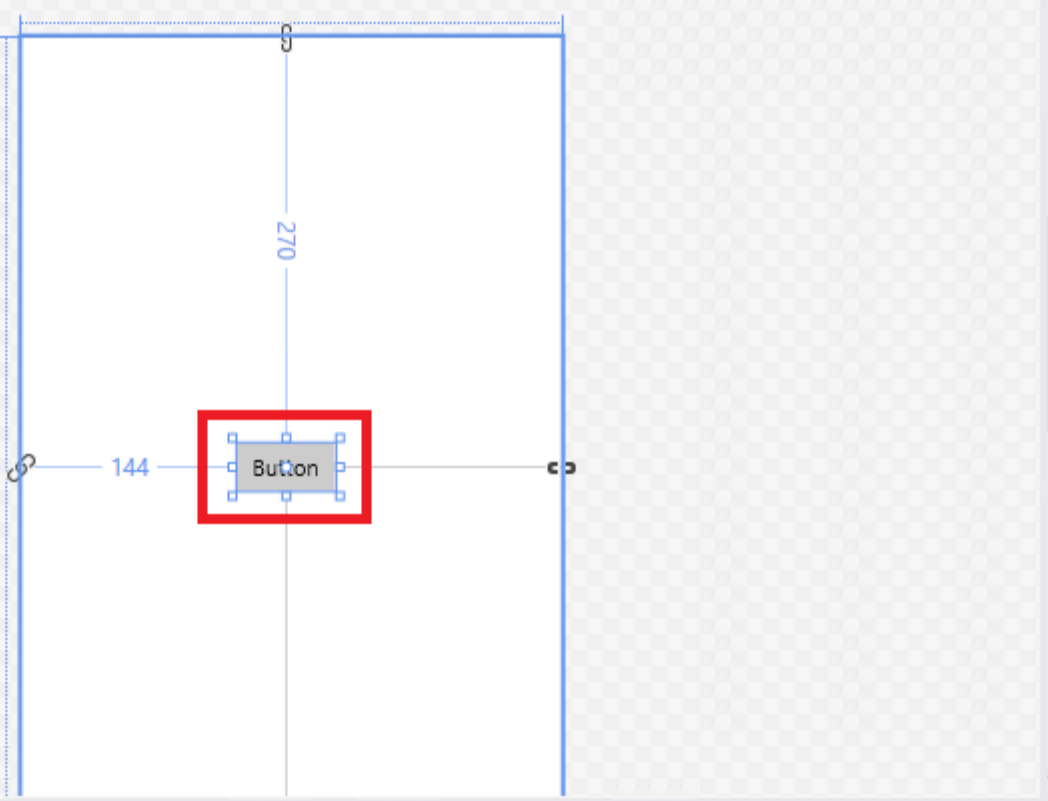
App1 - Microsoft Visual Studio (Administrator)

File Edit View Project Build Debug Team Design Format Tools Architecture Test Analyze Window Help

Debug x86 Local Machine

MainPage.xaml.cs MainPage.xaml\*

5" Phone (1920 x 1080) 300% scale Effective: 640 x 360



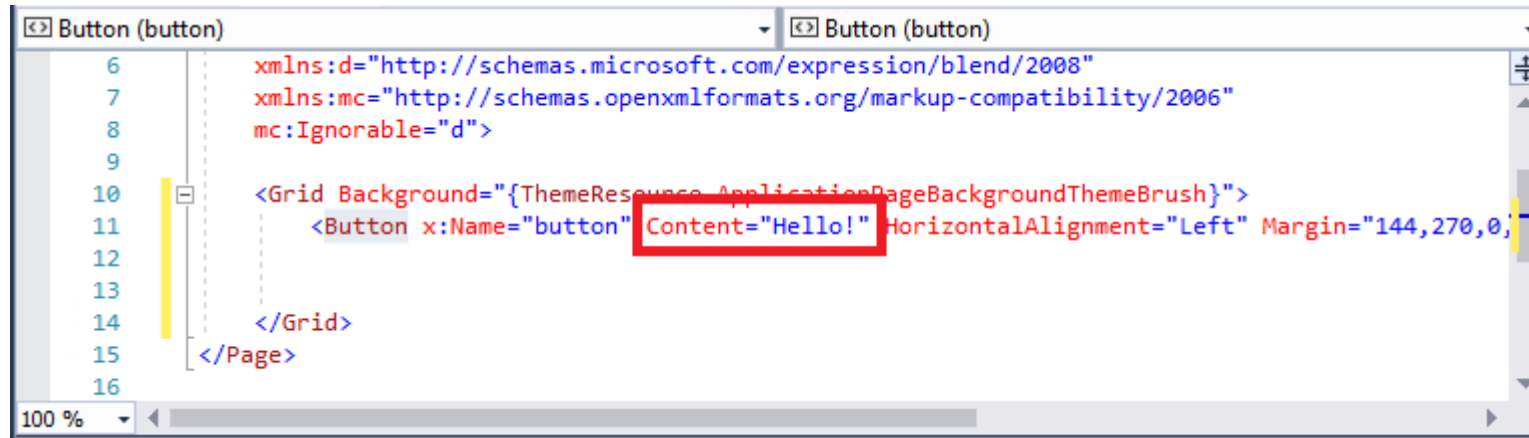
25% Design XAML

Button (button) Button (button)

```
6 xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
7 xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
8 mc:Ignorable="d">
9
10 <Grid Background="{ThemeResource ApplicationPageBackgroundThemeBrush}"
11 <Button x:Name="button" Content="Button" HorizontalAlignment="Left" Margin="144,270,0,0"
12
13
14 </Grid>
15 </Page>
```

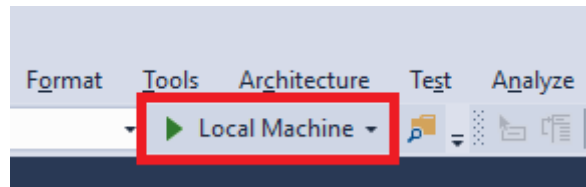
The button is on the designer, and its underlying code (highlighted) is automatically added to the designer's XAML code.

- Let's change some of the XAML code. Rename the text in the button code from `Button` to `Hello!`.



```
6  xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
7  xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
8  mc:Ignorable="d">
9
10 <Grid Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">
11     <Button x:Name="button" Content="Hello!" HorizontalAlignment="Left" Margin="144,270,0,
12
13
14 </Grid>
15 </Page>
16
```

- Now, start the app. You can do this by choosing the **Start** (▶) button on the toolbar, or by choosing the F5 key, or on the menu, choosing **Debug, Start Debugging**.



The app begins its build process and status messages appear in the Output window. Soon, you should see the form appear with your button in it. You now have a running app!

```
MainPage.xaml.cs | MainPage.xaml | Button (button) | Button (button)
8      mc:Ignorable="d">
9
10     <Grid Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">
11         <Button x:Name="button" Content="Hello!" HorizontalAlignment="Left" Margin="144,
12
13     </Grid>
14 </Page>
15
16
```



**Diagnostics Tools**

Diagnostics session: 39 seconds

30s

Events

Process Memory (MB) shot Private Bytes

CPU (% of all processors)

Summary Events Memory Usage CPU Usage

Events

- Show Events (0 of 0)
- Exceptions (0 of 0)
- IntelliTrace Events (0 of 0)
- UI Analysis Events (0 of 0)

Memory Usage

Autos

Name	Value
------	-------

Of course, it doesn't do much right now, but you can add more functionality to it later if you want.

10. When you're done running the program, choose the Stop (  ) button on the toolbar to stop it.

+

Let's recap what you did so far: you created a new C# Windows Universal project in Visual Studio, viewed its code, added a control to the designer, changed some XAML code, and then ran the project. Although the process was simplified for this example, this shows you some common parts of the Visual Studio IDE that you will use when you develop your own apps. If you want further details about this example, see [Create a "Hello, world" app \(XAML\)](#).+

## **Debug, test, and improve your code**

Nothing runs perfectly all the time. When you write code, you need to run it and test it for bugs and performance. Visual Studio's cutting edge debugging system enables you to debug code running in your local project, on a remote device, or on an emulator such as the ones for Android or Windows Phone devices. You can step through code one statement at a time and inspect variables as you go, you can step through multi-threaded applications, and you can set breakpoints that are only hit when a specified condition is true. You can monitor the values of variables as the code runs, and more. All of this can be managed in the code editor itself, so that you don't have to leave your code.+

ConsoleApp1 - Microsoft Visual Studio (Administrator)

File Edit View Project Build Debug Team Tools Architecture Test Analyze Window Help

Debug Any CPU Start

SBTester.cs

ConsoleApp1 ServiceBusTester.Program SendToEventHubs()

```
1 reference
66 public static async Task SendToQueue()
67 {
68     // Inform user
69     Console.WriteLine("Sending message to queue directly");
70
71     // Send to ServiceBus with object in constructor
72     await
73         queueClient.SendAsync(
74             new BrokeredMessage(Encoding.UTF8.GetBytes(JsonConvert.SerializeObject(new Pa
75
76     // Send to ServiceBus with Stream in constructor
77     await
78         queueClient.SendAsync(
79             new BrokeredMessage(
80                 new MemoryStream(
81                     Encoding.UTF8.GetBytes(JsonConvert.SerializeObject(new Pa
82
83     // Inform user
84     Console.WriteLine("Finished sending to queue directly");
85 }
86
87 /// <summary>
88 /// Send message to EventHubs.
89 /// </summary>
90 /// <returns>
91 /// The <see cref="Task"/>.
92 /// </returns>
93 1 reference
94 public static async Task SendToEventHubs()
95 {
96     // Inform user
97     Console.WriteLine("Sending via Event Hubs");
```

(awaitable) class System.Threading.Tasks.Task  
Represents an asynchronous operation. To browse the .NET Framework source code for this type, see the Reference Source.

References

- App.config
- packages.config
- Program.cs
- SBTester.cs

Properties

100 %

The key combination (Ctrl+Alt+W, 1) is bou... Ln 89 Col 24 Ch 24 INS Add to Source Control

For testing, Visual Studio offers unit testing, IntelliTest, load and performance testing, and more. To get more details about the Visual Studio debugging process, see [Debugger Feature Tour](#). To learn more about testing, see [Testing Tools](#). To learn more about improving the performance of your apps, see [Profiling Tools](#).+

## Deploy your finished application

When your application is ready to deploy to users or customers, Visual Studio provides the tools to do that, whether you're deploying to the Windows Store, to a SharePoint site, or with InstallShield or Windows Installer technologies. It's all accessible through the IDE. For more information, see [Deploying Applications, Services, and Components](#).+

## Quick tour of the IDE

To give you a high-level visual overview of Visual Studio, the following image shows Visual Studio with an open project along with several key tool windows you will most likely use.+

- [Solution Explorer](#) lets you view, navigate, and manage your code files.
- The [Editor](#) window shows your code and enables you to edit source code and designer data.
- The [Output](#) window shows output messages from compiling, running, debugging, and more.
- [Team Explorer](#) lets you track work items and share code with others using version control technologies such as [Git](#) and [Team Foundation Version Control \(TFVC\)](#).
- [Cloud Explorer](#) lets you view and manage your Azure resources, such as virtual machines, tables, SQL databases, and more.

+

Create a new project

Save your project

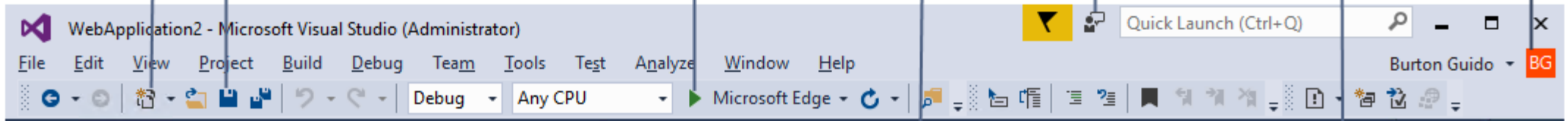
Run your code

Edit your code

Manage files, projects, & solutions

Send feedback

Sign in



Manage server resources

Add controls to your UI

Manage your Azure resources

Server Explorer

Microsoft Azure

Resource Types

Search for resources

- (Local)
  - Data Lake Analytics
  - (Local)
    - Databases
  - Storage Accounts
    - (Development)
      - Blob Containers
      - Queues
      - Tables
- Free Trial ( @outl
  - Storage Accounts
    - athenastorage2354
      - Blob Containers

Actions Properties

Name	at
Type	M
Subscription	Fr
Resource Group	At
Location	wi
Primary Key	E/

```

AccountController.cs HomeController.cs ManageController.cs
[C#] WebApplication2
  WebApplication2.Controllers
    _signInManager
    return View(new VerifyCodeViewModel { Provider = provi
  104
  105
  106
  107 //
  108 // POST: /Account/VerifyCode
  109 [HttpPost]
  110 [AllowAnonymous]
  111 [ValidateAntiForgeryToken]
  112 0 references
  113 public async Task<ActionResult> VerifyCode(VerifyCodeViewM
  114 {
  115     if (!ModelState.IsValid)
  116     {
  117         return View(model);
  118     }
  119
  120 // The following code protects for brute force attacks
  121 // If a user enters incorrect codes for a specified am
  122 // will be locked out for a specified amount of time.
  123 // You can configure the account lockout settings in I
  124 var result = await SignInManager.TwoFactorSignInAsync(
  125     rememberBrowser: model.RememberBrowser);
  126 switch (result)
  127 {
  128     case SignInStatus.Success:
  129         return RedirectToLocal(model.ReturnUrl);
  130         // SignInStatus.LockedOut:
  131
  132
  133
  134
  135
  136
  137
  138
  139
  140
  141
  142
  143
  144
  145
  146
  147
  148
  149
  150
  151
  152
  153
  154
  155
  156
  157
  158
  159
  160
  161
  162
  163
  164
  165
  166
  167
  168
  169
  170
  171
  172
  173
  174
  175
  176
  177
  178
  179
  180
  181
  182
  183
  184
  185
  186
  187
  188
  189
  190
  191
  192
  193
  194
  195
  196
  197
  198
  199
  200
  
```

Solution Explorer

Search Solution Explorer (Ctrl+)

Solution 'WebApplication2' (1 pr

- WebApplication2
  - Properties
  - References
  - App\_Data
  - App\_Start
  - Content
  - Controllers
    - AccountController.cs
    - HomeController.cs
    - ManageController.cs
  - fonts
  - Models

Solution Explorer Class View

Output

Show output from: Package Manager

IsDirty	: False
FileCount	: 1
Name	: jquery-1.10.2.intellisense.js
Collection	: System.__ComObject

Team Explorer - Home

Search Work Items (Ctrl+')

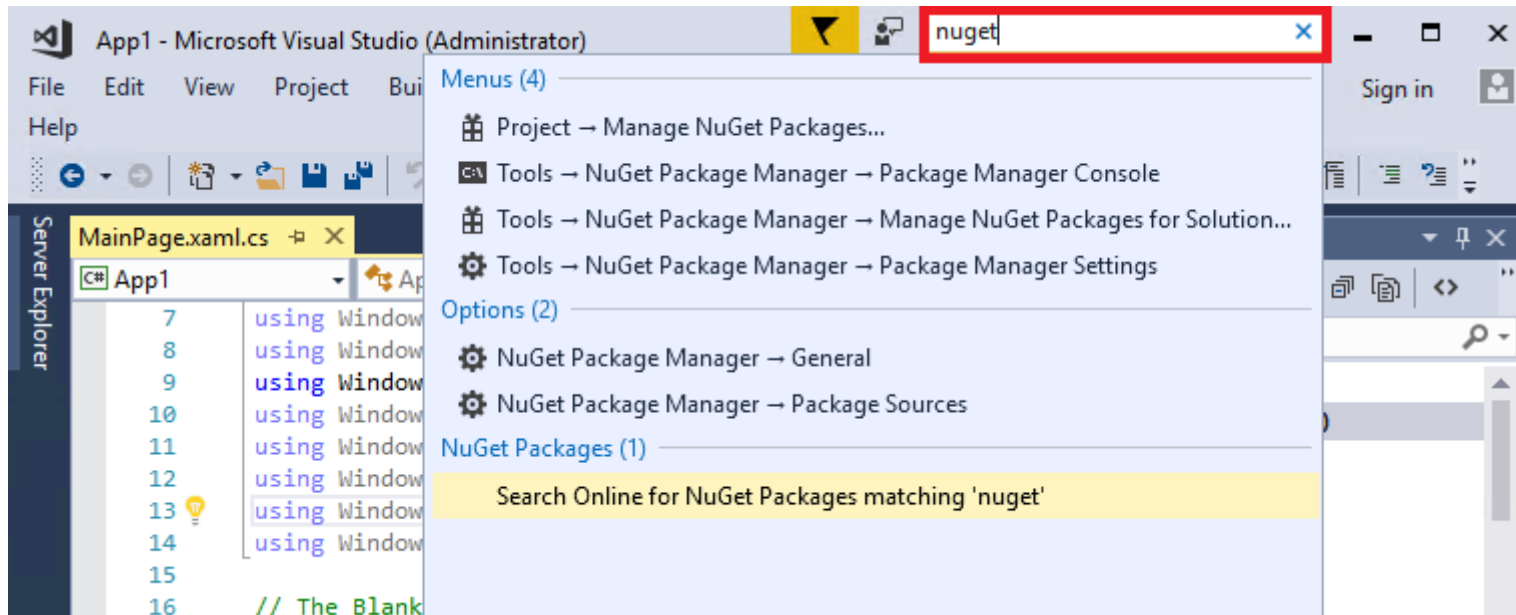
Home | ProjectAthena

- Visual Studio Team Services
  - ProjectAthena/...
    - <https://bguido.visualst...>
- Project
  - Web Portal | Task Board | Team Room

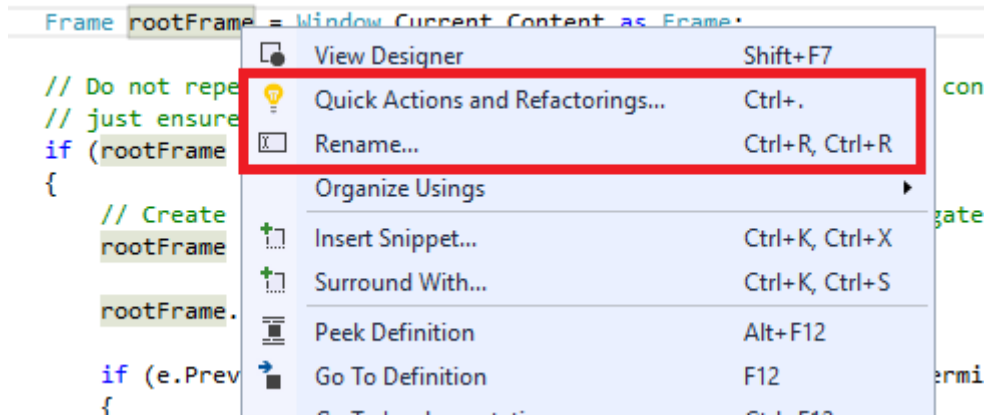


Following are some other common productivity features in Visual Studio. +

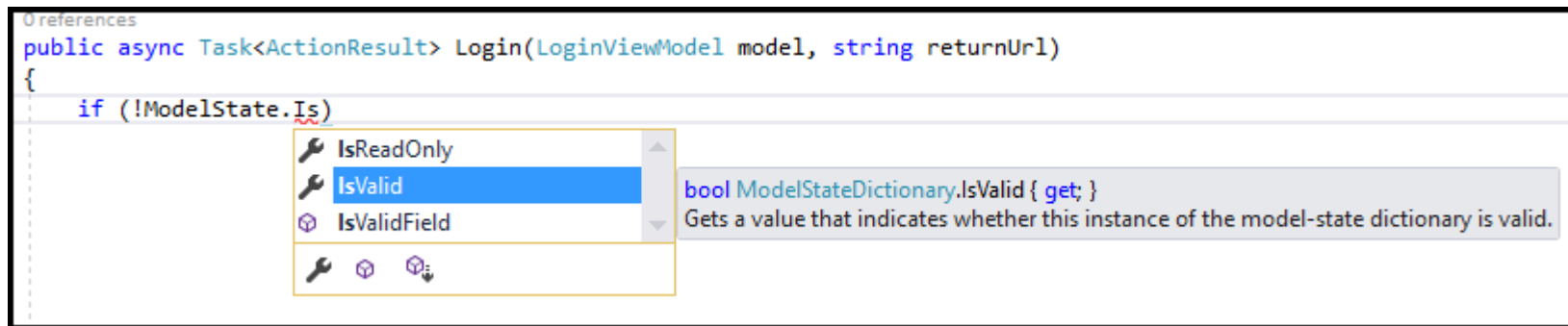
- The [Quick Launch](#) search box is a great way to rapidly find what you need in Visual Studio. Just start entering in the name of whatever you are looking for and Visual Studio give you options that take you exactly where you want to go. Quick Launch also shows links that start the Visual Studio Installer for any Workload or Individual component.



- [Refactoring](#) includes operations such as intelligent renaming of variables, moving selected lines of code into a separate function, moving code to other locations, reordering function parameters, and more.



- **IntelliSense** is an umbrella term for a set of popular features that display type information about your code directly in the editor and, in some cases, write small bits of code for you. It's like having basic documentation inline in the editor, which saves you from having to look up type information in a separate help window. IntelliSense features vary by language. For more information, see [Visual C# IntelliSense](#), [Visual C++ Intellisense](#), [JavaScript IntelliSense](#), [Visual Basic-Specific IntelliSense](#). The following illustration shows some IntelliSense features at work:



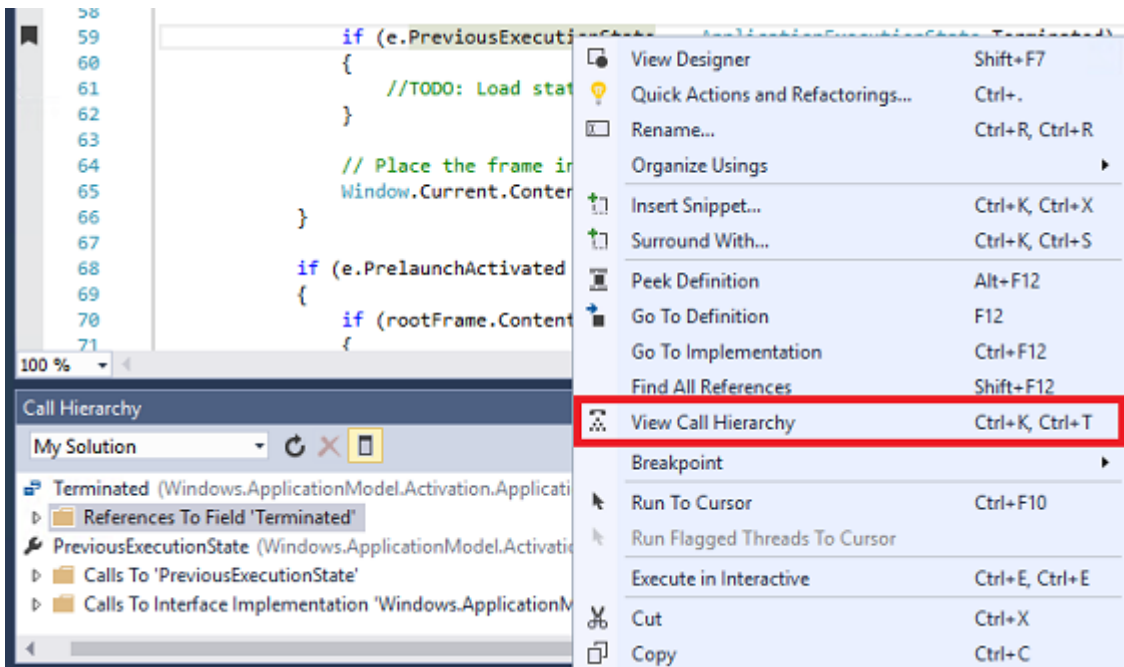
- **Squiggles** are wavy red underlines that alert you to errors or potential problems in your code in real time as you type. This enables you to fix them immediately without waiting for the error to be discovered during compilation or run time. If you hover over the squiggle, you see additional information about the error. A light bulb may also appear in the left margin with suggestions for how to fix the error. For more information, see [Perform quick actions with light bulbs](#).

```
public ActionResult Register()
{
    int x = i * Math.Log8
    return View
}
//
// POST: /Account/Register
```

'Math' does not contain a definition for 'Log8'

Show potential fixes (Ctrl+.)

- The [Call Hierarchy](#) window can be opened on the text editor context menu to show the methods that call, and are called by, the method under the caret (insertion point).



- [CodeLens](#) enables you to find references and changes to your code, linked bugs, work items, code reviews, and unit tests, all without leaving the editor.

```

FabrikamFiber.Web.Tests\Controllers\CustomersControllerTest.cs (2)
  28 : controller.Create(new Customer());
  38 : controller.Create(null);
FabrikamFiber.Web\Helpers\GuardHelper.cs (1)
  16 : controller.Create(new Customer());
Show on Code Map | Collapse All
3 references | 0/2 passing | Francis Totten, 3 hou
public ActionResult Create(Customer
{
    if (customer == null)
    {
GuardHelper.cs (16,18)
FabrikamFiber.Web.Helpers.GuardHelper.GuardCustomer()
14 {
15     CustomersController controller = new CustomersController(null);
16     controller.Create(new Customer());
17
18 }

```

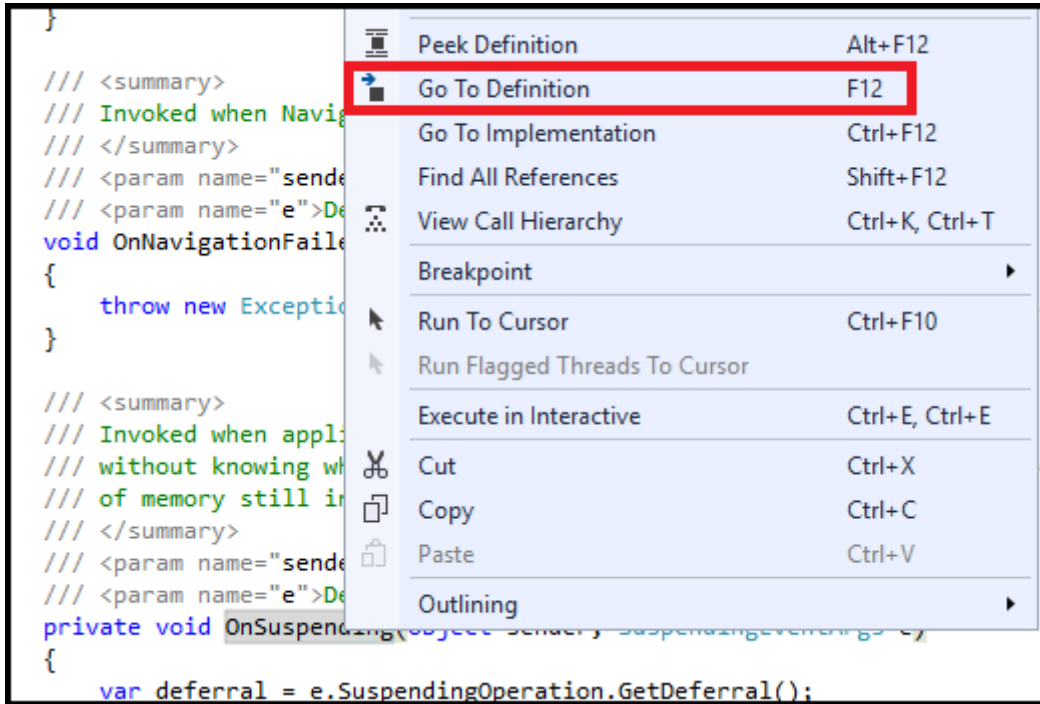
- The [Peek to Definition](#) window shows a method or type definition inline, without navigating away from your current context.

```

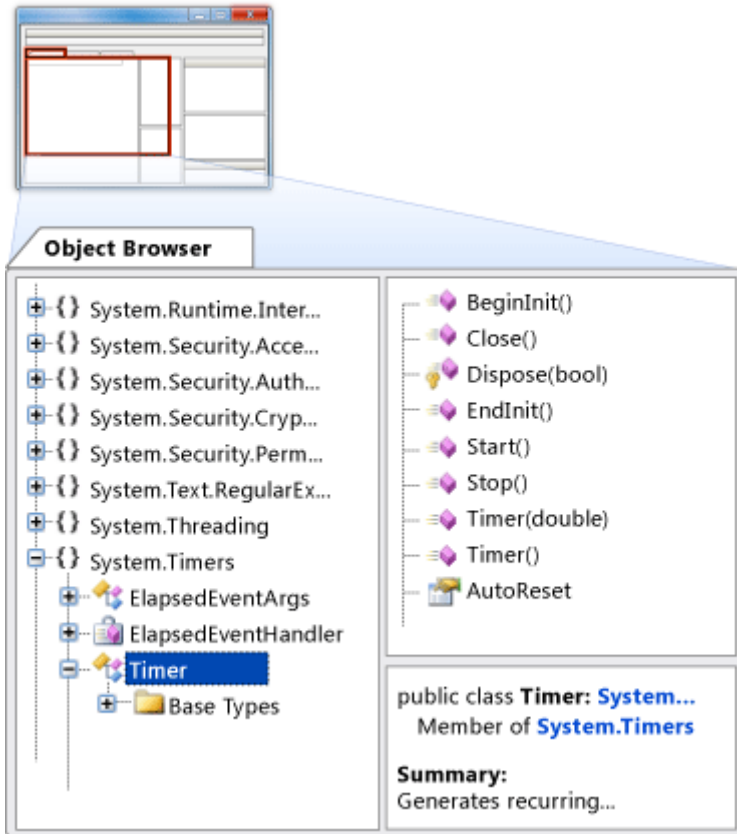
if (e.PrelaunchActivated == false)
LaunchActivatedEv
47 // FALSE.
48 public System.Boolean PrelaunchActivated { get; }
49 ...public ApplicationExecutionState PreviousExecutionState { get; }
56 ...public SplashScreen SplashScreen { get; }
64 ...public TileActivatedInfo TileActivatedInfo { get; }
73 ...public System.String TileId { get; }
82 ...public User User { get; }
89 ...public ActivationViewSwitcher ViewSwitcher { get; }
96 }

```

- The **Go To Definition** context menu option takes you directly to the place where the function or object is defined. Other navigation commands are also available by right-clicking in the editor.



- A related tool, the [Object Browser](#), enables you to inspect .NET or Windows Runtime assemblies on your system to see what types they contain and what members (properties, methods, events) those types contain.



+

## Collaborate with others and control your source code

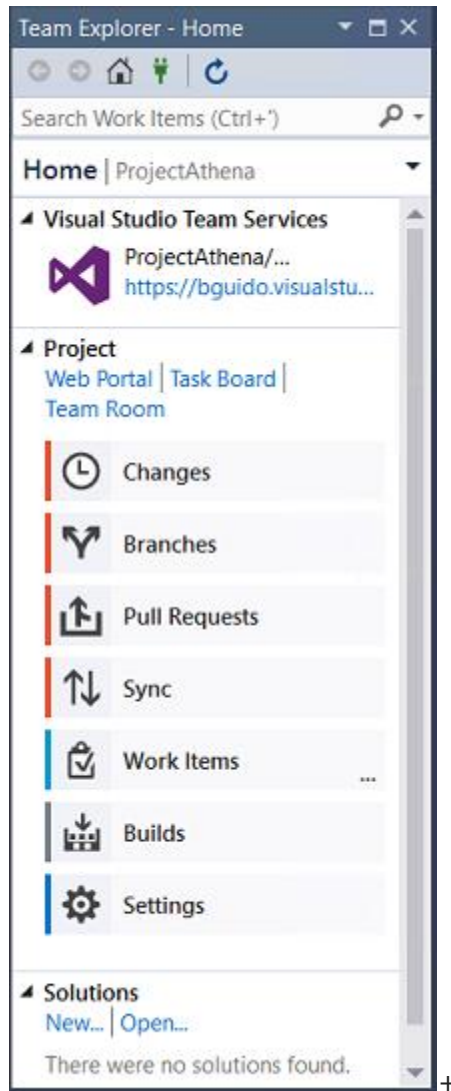
You can manage your source code in Git repos hosted by any provider, including GitHub. Or use [Visual Studio Team Services \(VSTS\)](#) to manage code alongside bugs and work items for your whole project. +

Visual Studio Team Services is a cloud-based service for hosting software projects and enabling collaboration in teams. VSTS supports both Git and Team Foundation Source Control systems, as well as Scrum, CMMI and Agile development methodologies. Team Foundation Version Control (TFVC) uses a single, centralized server repository to track and version files. Local changes are always checked in to the central server where other developers can get the latest changes. +

Team Foundation Server (TFS) is the application lifecycle management hub for Visual Studio. It enables everyone involved with the development process to participate using a single solution. TFS is useful for managing heterogeneous teams and projects, too. +

If you have a Visual Studio Team Services account or a Team Foundation Server on your network, you connect to it through the Team Explorer window in Visual Studio. From this window you can check code into or out of source control, manage work items, start builds, and access team rooms and workspaces. You can open Team Explorer from the **Quick Launch** box, or on the main menu from **View, Team Explorer** or from **Team, Manage Connections**. +

The following image shows the Team Explorer window for a solution that is hosted in VSTS: +

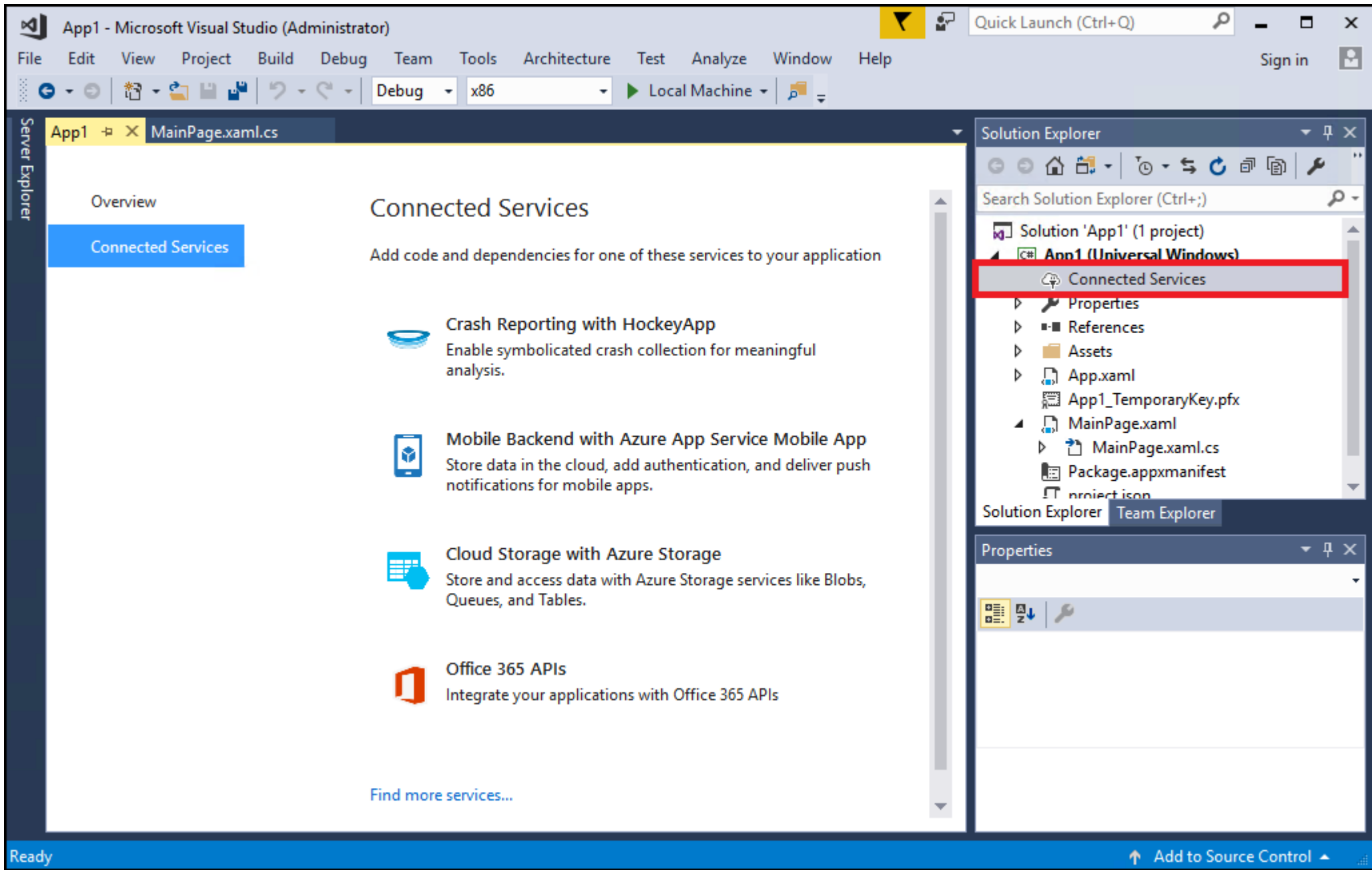


For more information about Visual Studio Team Services, see [Visual Studio Team Services](#). For more information about Team Foundation Server, see [Team Foundation Server](#).

## Connect to services, databases, and cloud-based resources

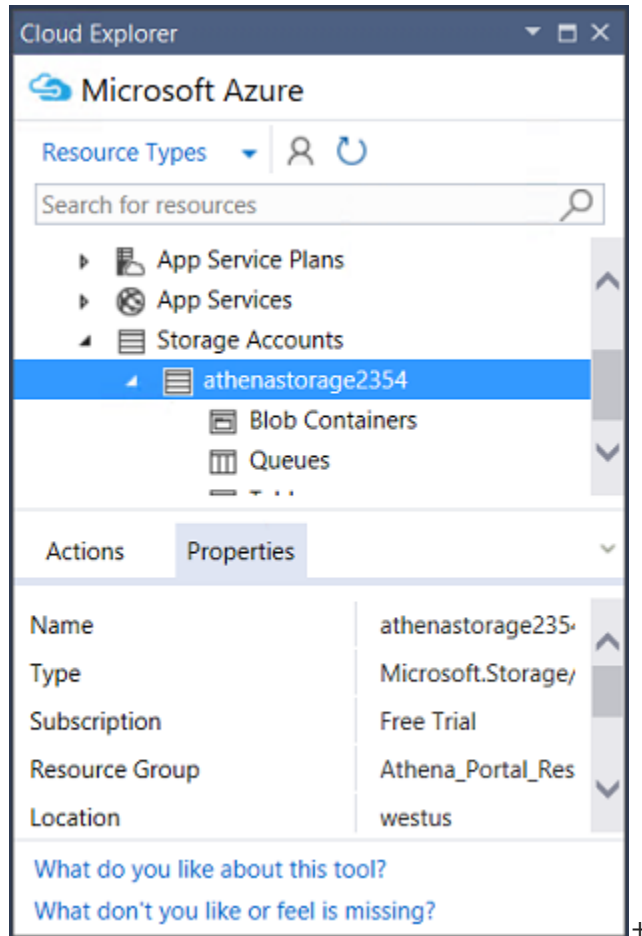


The cloud is critical for today's online world, and Visual Studio provides you the means to leverage it. For example, the Connected Services feature enables you to connect your app to services. Your apps can use it to store their data on Azure storage, among other things. +



Choosing a service on the **Connected Services** page starts a Connected Services Wizard that configures your project and downloads the necessary NuGet packages to help get you started coding against the service.+

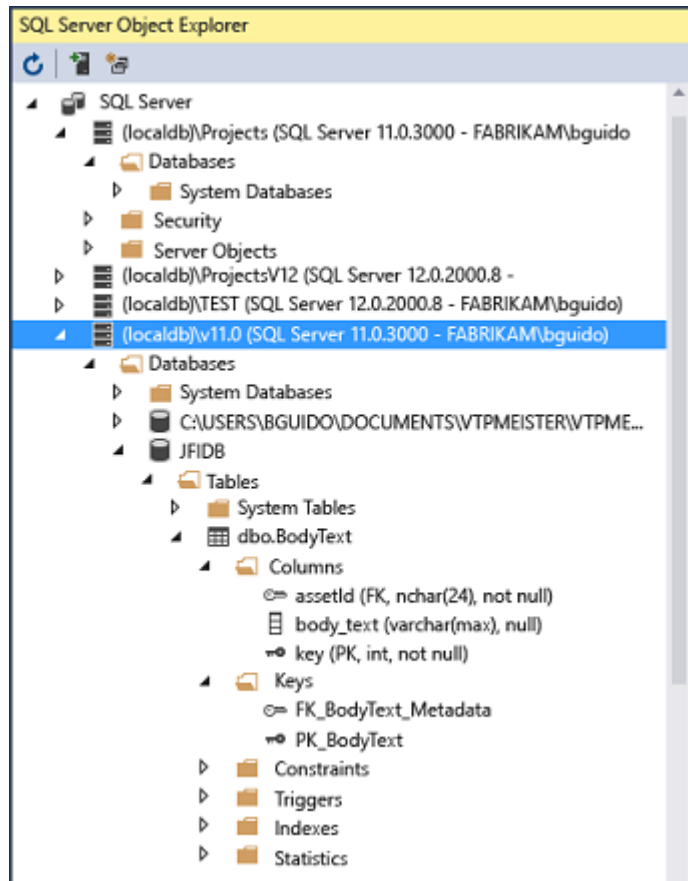
You can view and manage your Azure-based cloud resources within Visual Studio using [Cloud Explorer](#). Cloud Explorer shows the Azure resources in all the accounts managed under the Azure subscription you are logged into. You can get Cloud Explorer by selecting the Azure development workload in the Visual Studio installer.+



**Server Explorer** helps you browse and manage SQL Server instances and assets on Azure, Salesforce.com, Office 365, and websites. To open Server Explorer, on the main menu, choose **View, Server Explorer**. See [Add new connections](#) for more information on using Server Explorer. +

[SQL Server Data Tools \(SSDT\)](#) is a powerful development environment for SQL Server, Azure SQL Database and Azure SQL Data Warehouse. It enables you to build, debug, maintain, and refactor databases. You can work with a database project, or directly with a connected database instance on- or off-premises. +

**SQL Server Object Explorer** in Visual Studio provides a view of your database objects similar to SQL Server Management Studio. SQL Server Object Explorer enables you to do light-duty database administration and design work, including editing table data, comparing schemas, executing queries by using contextual menus right from SQL Server Object Explorer, and more. See [Manage Objects by Using Object Explorer](#) for more information. +



## Extend Visual Studio

If Visual Studio doesn't have the exact functionality you need, you can add it! You can personalize the IDE based on your workflow and style, add support for external tools not yet integrated with Visual Studio, and modify existing functionality to increase your productivity. Visual Studio provides tools, controls, and templates from Microsoft, our partners, and the community. To learn more about extending Visual Studio, see [Extend Visual Studio IDE](#).

## Learn more and find out what's new

If you've never used Visual Studio before, learn the basics, starting with [Get Started with Visual Studio](#), or check out the free Visual Studio courses available on [Microsoft Virtual Academy](#). If you want to find out about new features in Visual Studio 2017, see [What's New in Visual Studio 2017](#).+

Congratulations on completing the tour of the Visual Studio IDE! We hope you learned something useful about some of its main features.+

## See also

- [Visual Studio IDE](#)
- [Visual Studio Downloads](#)
- [The Visual Studio Blog](#)
- [Visual Studio Forums](#)
- [Microsoft Virtual Academy](#)
- [Channel 9](#)

+

0 comments

▼

- [Get Livefyre](#)
- [FAQ](#)

Sign in

*26 people listening*



+ Follow

Share

Post comment as...

Newest | Oldest

### Is this page helpful?

YesNo

- [Blog](#)
- [Privacy & Cookies](#)
- [Terms of Use](#)
- [Feedback](#)
- [Impressum](#)
- [Terms of Use](#)
- [FeedbackTrademarks](#)

