Visual Studio

Lecture 03

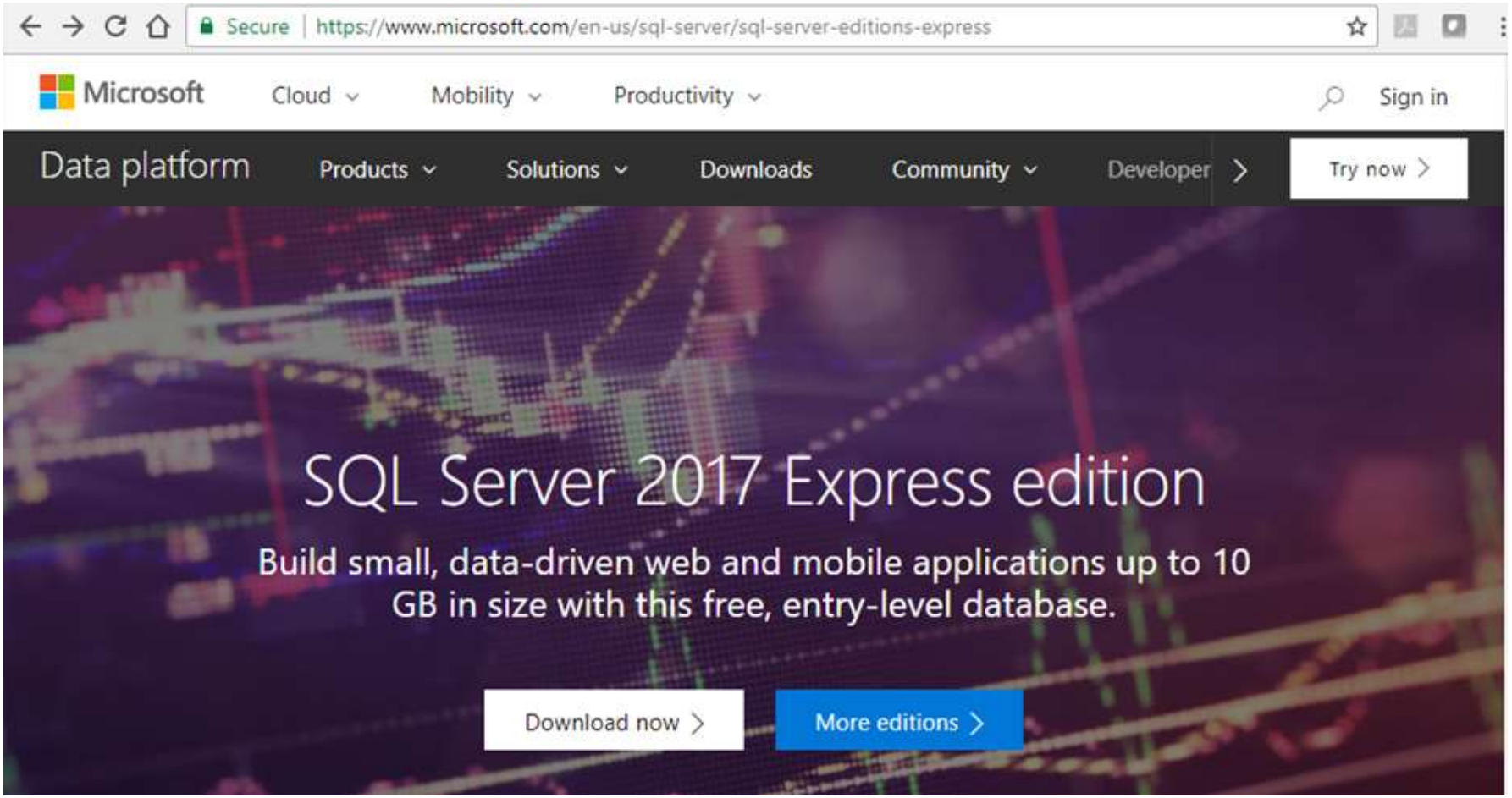
Deep Azure @ McKesson

Zoran B. Djordjević

SQL Server Express

- Large majority of our applications stores data in a relational database.
- In Azure, one could use practically any commercial RDBMS: Oracle, DB, MySQL , PostgreSQL, MS SQL Server, and others.
- MS SQL Servers play somewhat privileged role in Azure since several of native Azure's storage services are based on SQL Server, or at least on its client API's.
- We will not go into the details of SQL Server technology, SQL and associated API's.
- We will just build some familiarity with the technology, in the case you have none.
- On our local desktops we will use a free version of SQL Server called SQL Server Express.
- SQL Server Express restricts you to 10GB of saved data. Otherwise all APIs are the same as in production quality SQL Server.

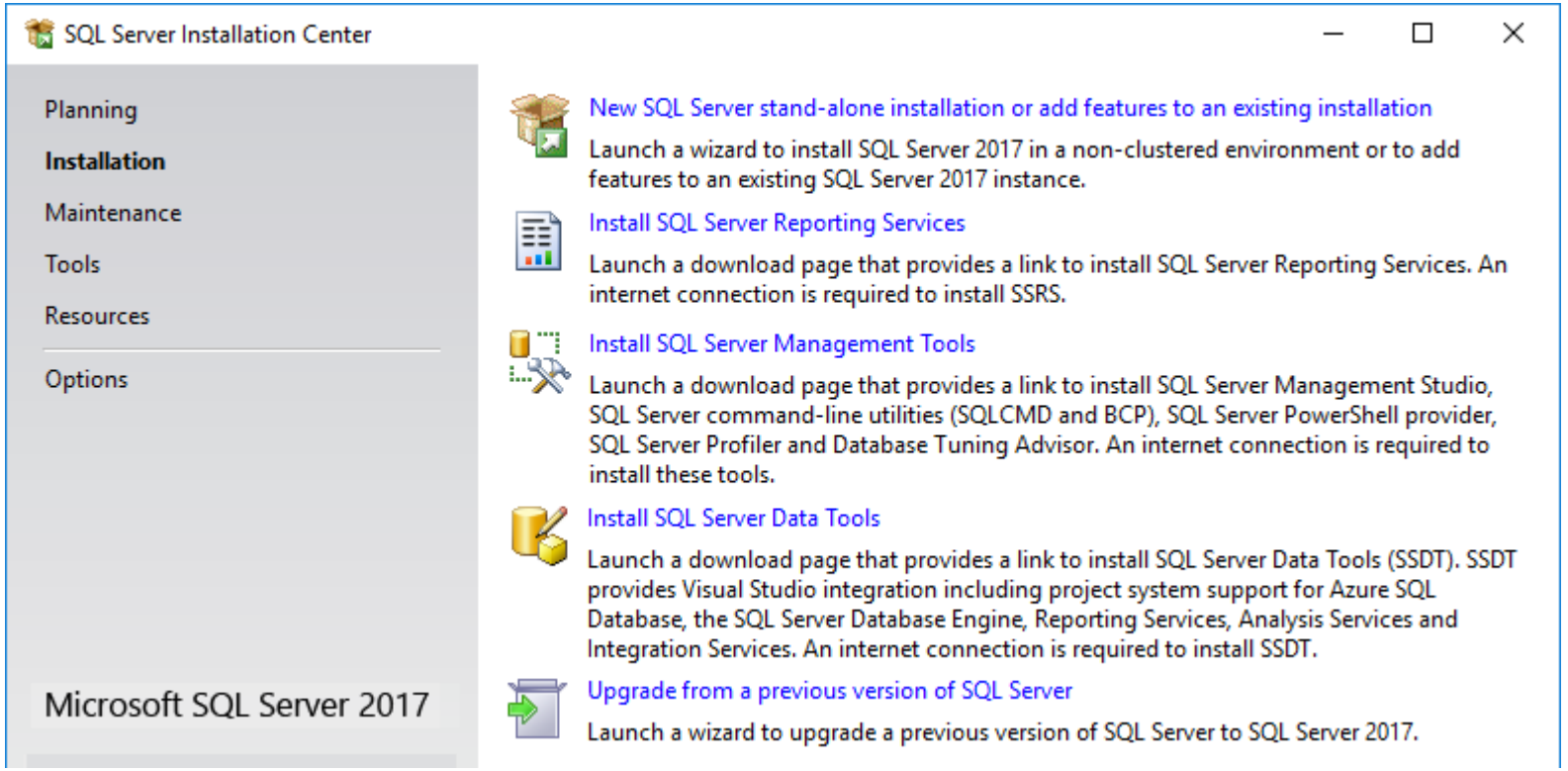
SQL Server Express



- Download Installation executable `SQLServer2017-SSEI-Expr.exe` and run it as an administrator.

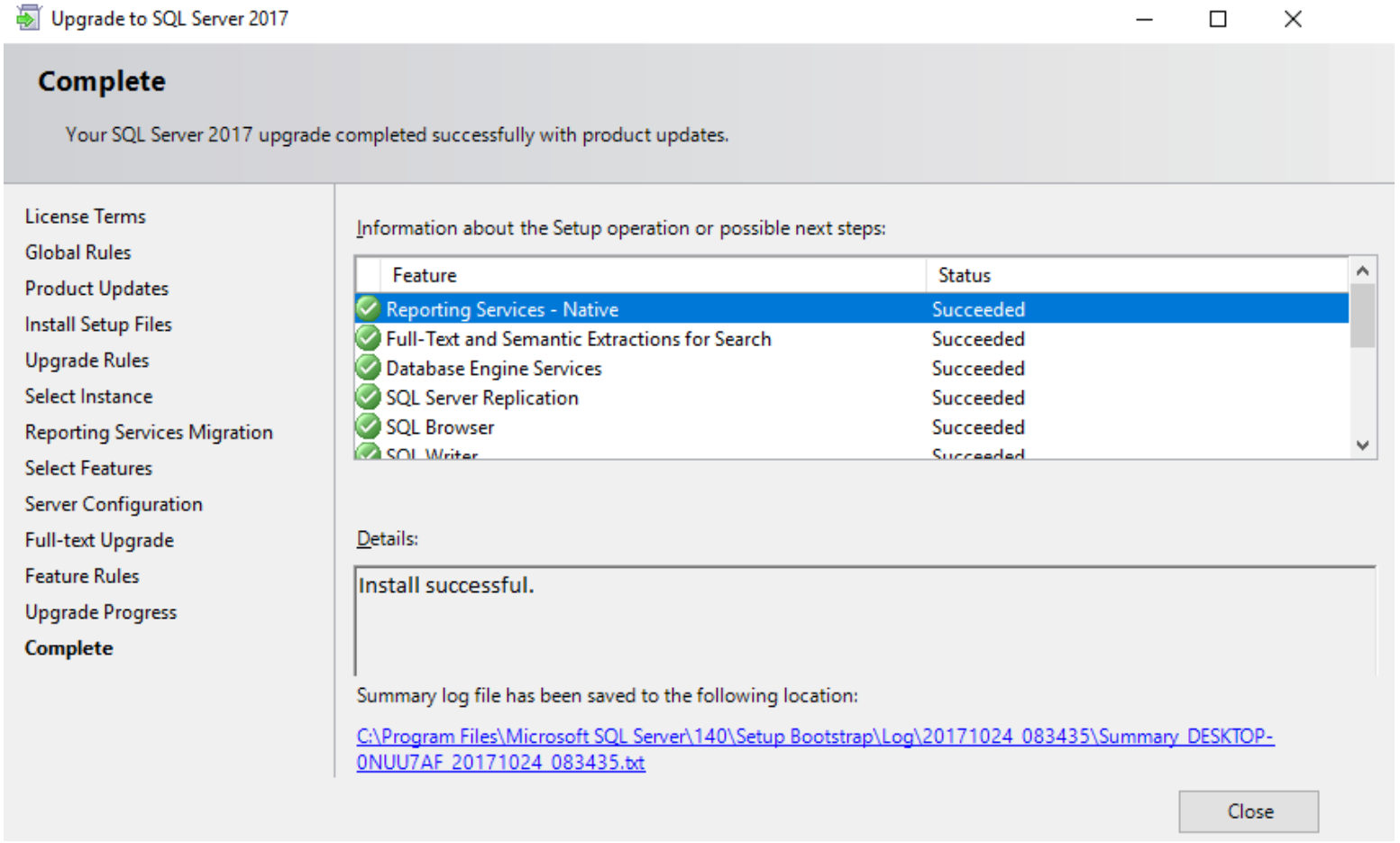
Installation Steps

- If you do not have SQL Server Express installed, select New SQL Server stand-alone.
- You will also need SQL Server Management (Tools) Studio. Once SQL Server itself is installed, you run new installation in order to select and then install SSMS
- On the following page(s) accept the license, click Next. Keep on hitting Next.



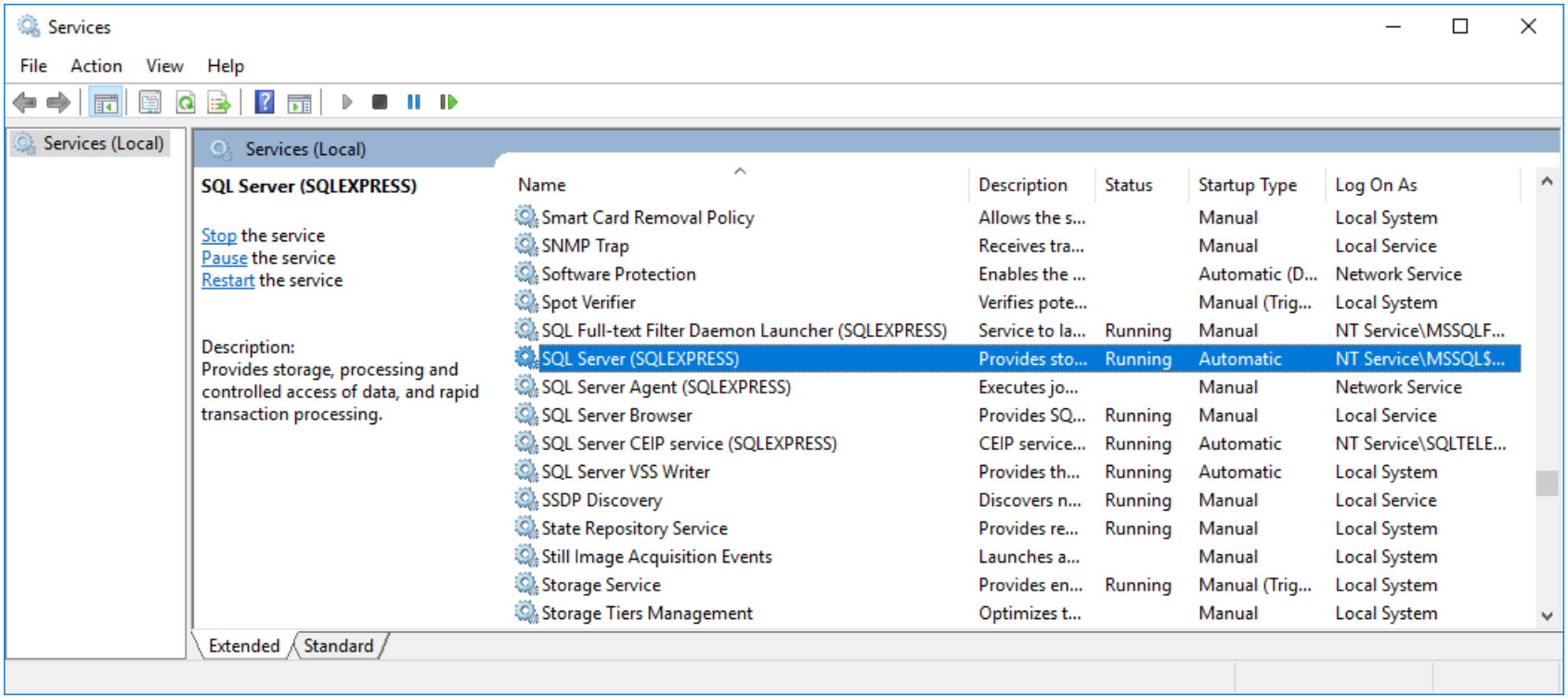
Finished SQL Server Installation

- Once the installation of the SQL server is done you will get a widget that looks like this. Hit Close. You will have to reboot your machine, afterwards.



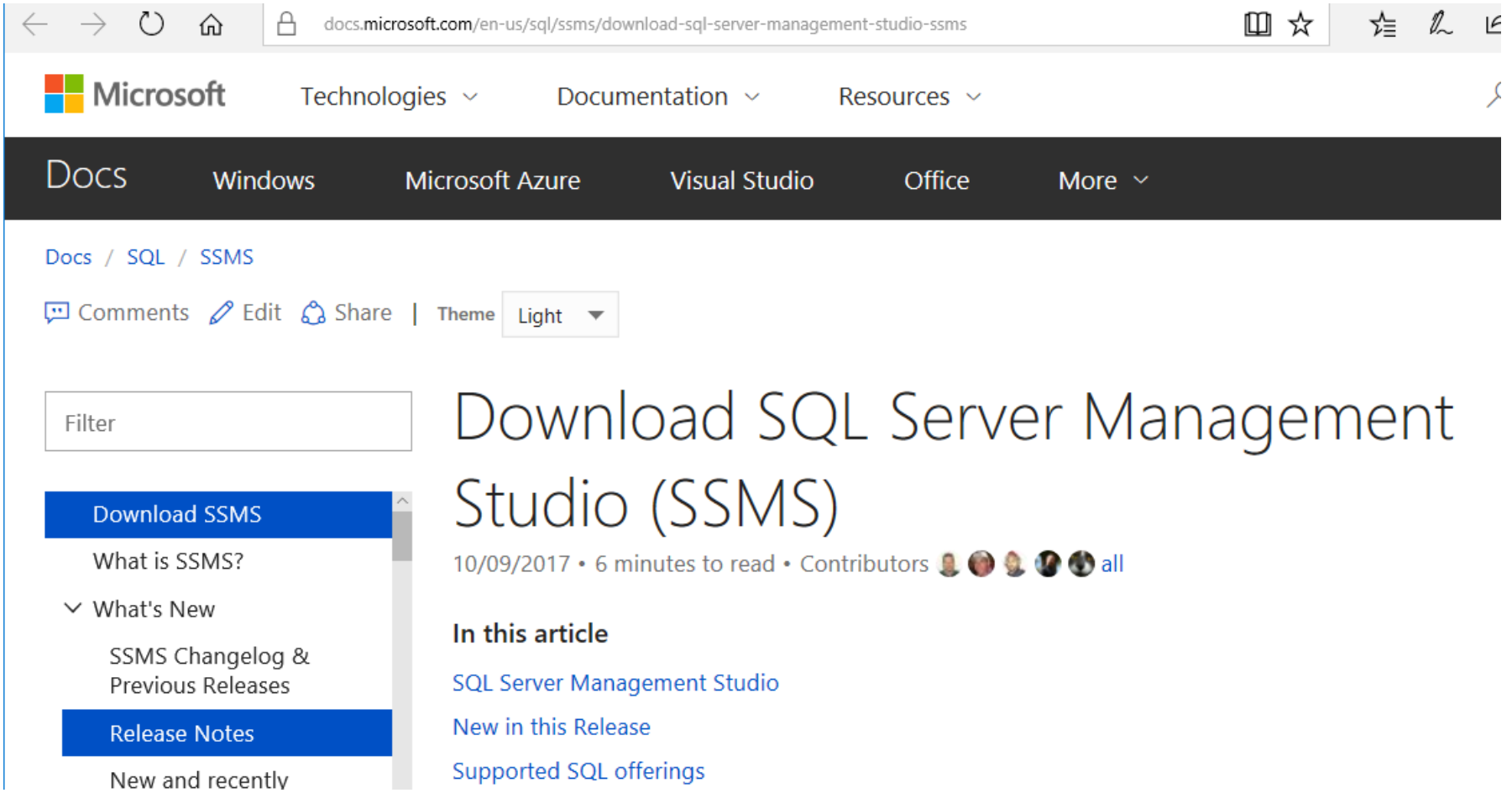
Start, Stop SQL Server

- To start or stop your SQL Server Express, in the bottom left command window on you Windows desktop, type `Services`, and then select `Services Desktop App.`
- In the window that opens scroll down to `SQL Server` collection of services. `SQL Server (SQLEXPRESS)` is important to you. Others you might use.
- If you are an infrequent user of these services you should change all of them from Automatic to Manual. Double click on a service and change Automatic to Manual.
- To stop a service, select it, and then hit the black square on the top bar.

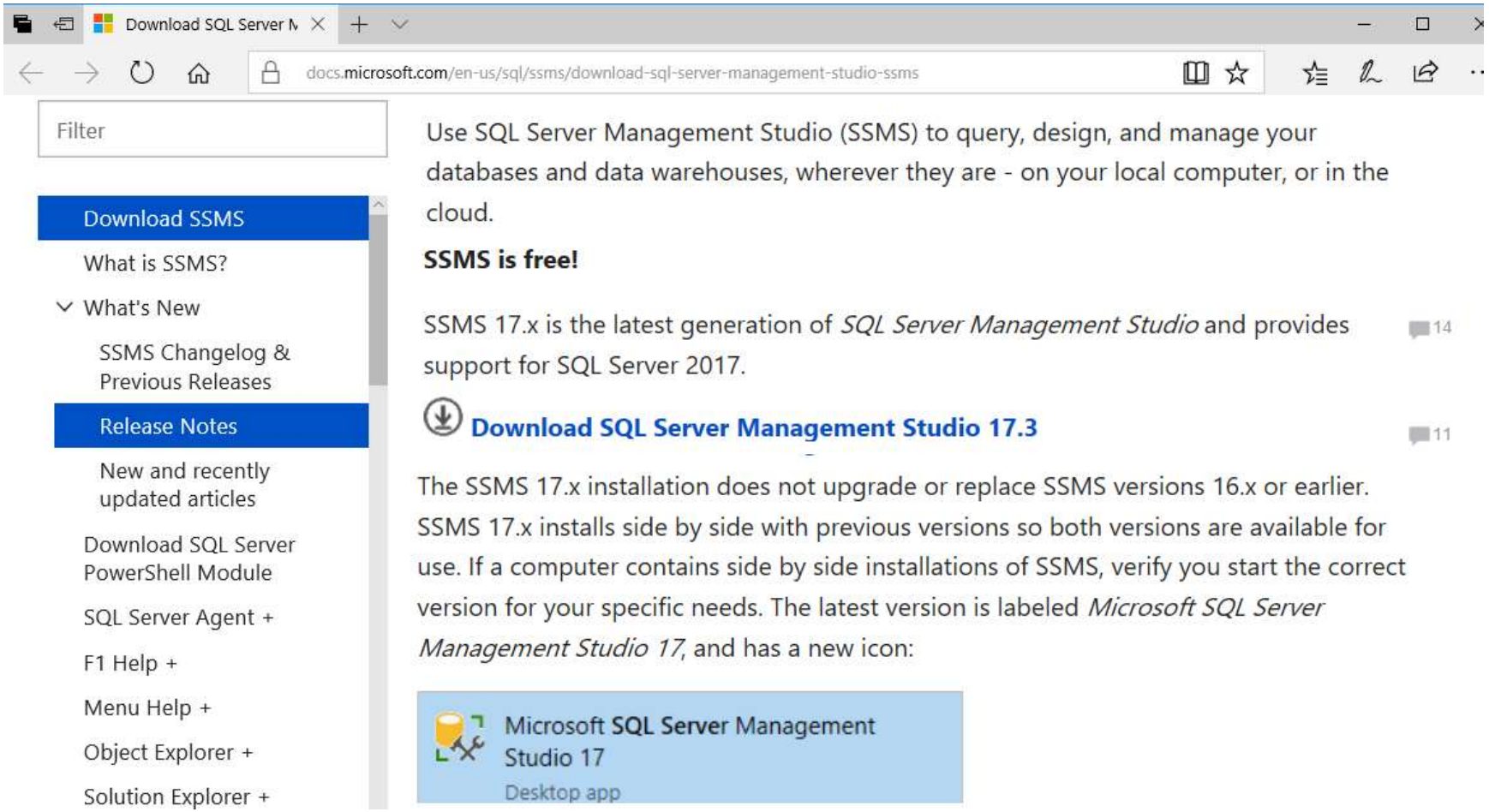


Download SSMS

- Your programs could access SQL Server as is. For human inspection and random queries you need another tool, SQL Server Management Studio (SSMS)

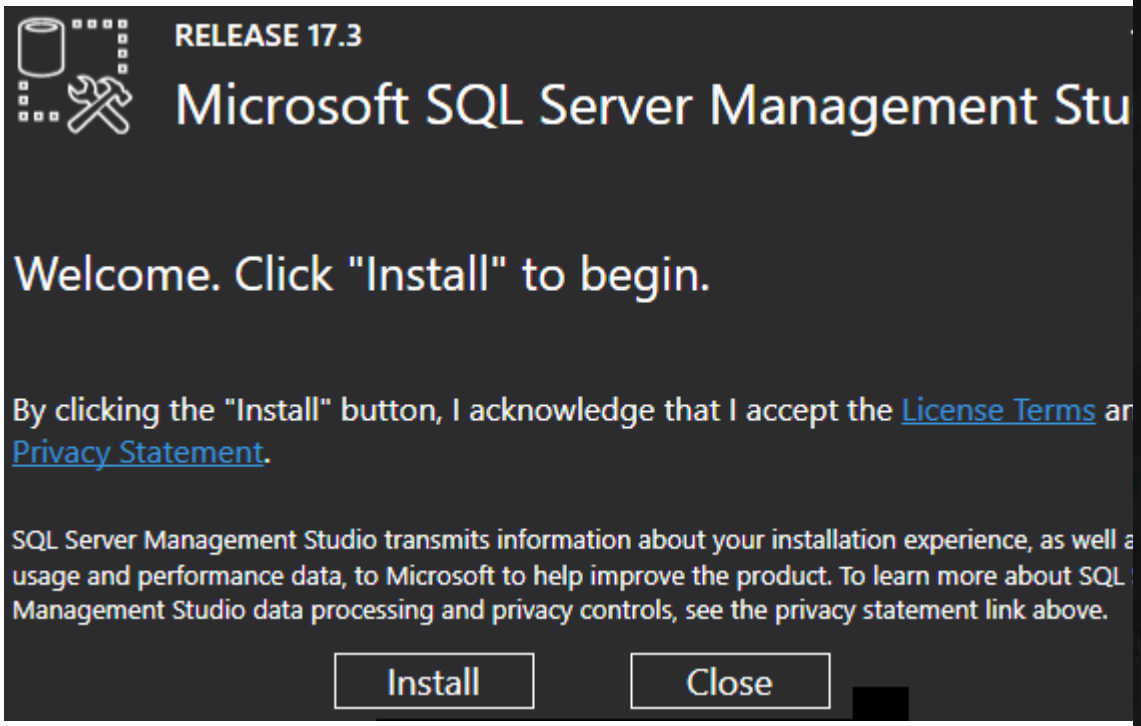


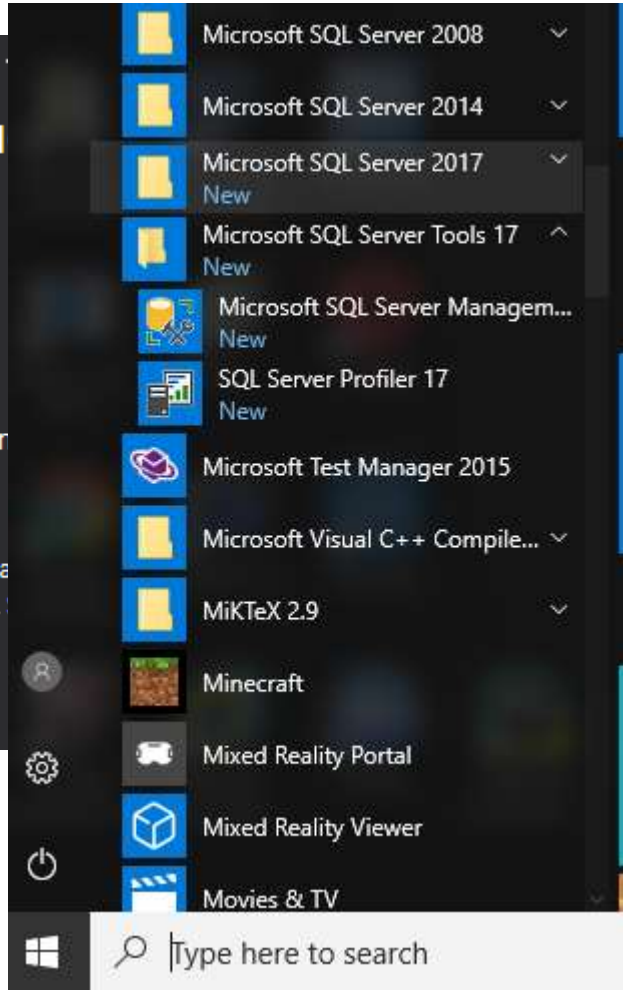
Find Download Link



- Download is bulky, some 800 MB. Once it is done, run `SSMS-Setup-ENU.exe`

Install, That is All

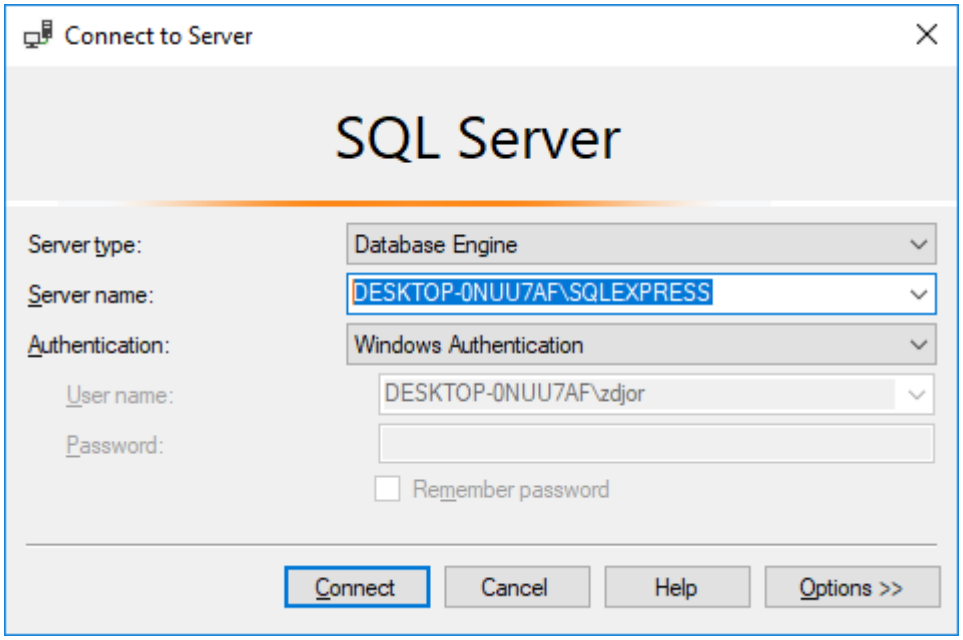


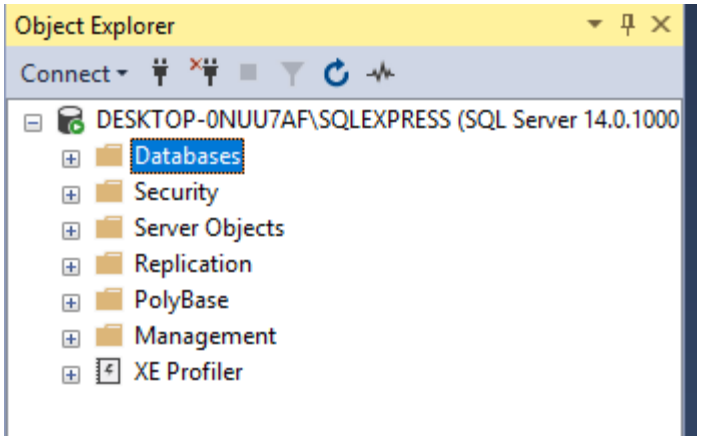


- Among installed applications you will see Microsoft SQL Server Tools 17 group and it that group `Microsoft SQL Server Management Studio`
- Click on the line and start `SSMS`

Connect to local SQL EXPRESS, create a Database

- Please note, that your local SQL Server EXPRESS most probably has the name that starts with your machine name, like: `DESKTOP_0NUU7AF\SQLEXPRESS`
- In the Connect to Server widget enter local SQL Server name. Click `Connect`.
- In the `Object Explorer` to the left, right click on `Databases` and select `New Database`. `New Database` wizard appears.

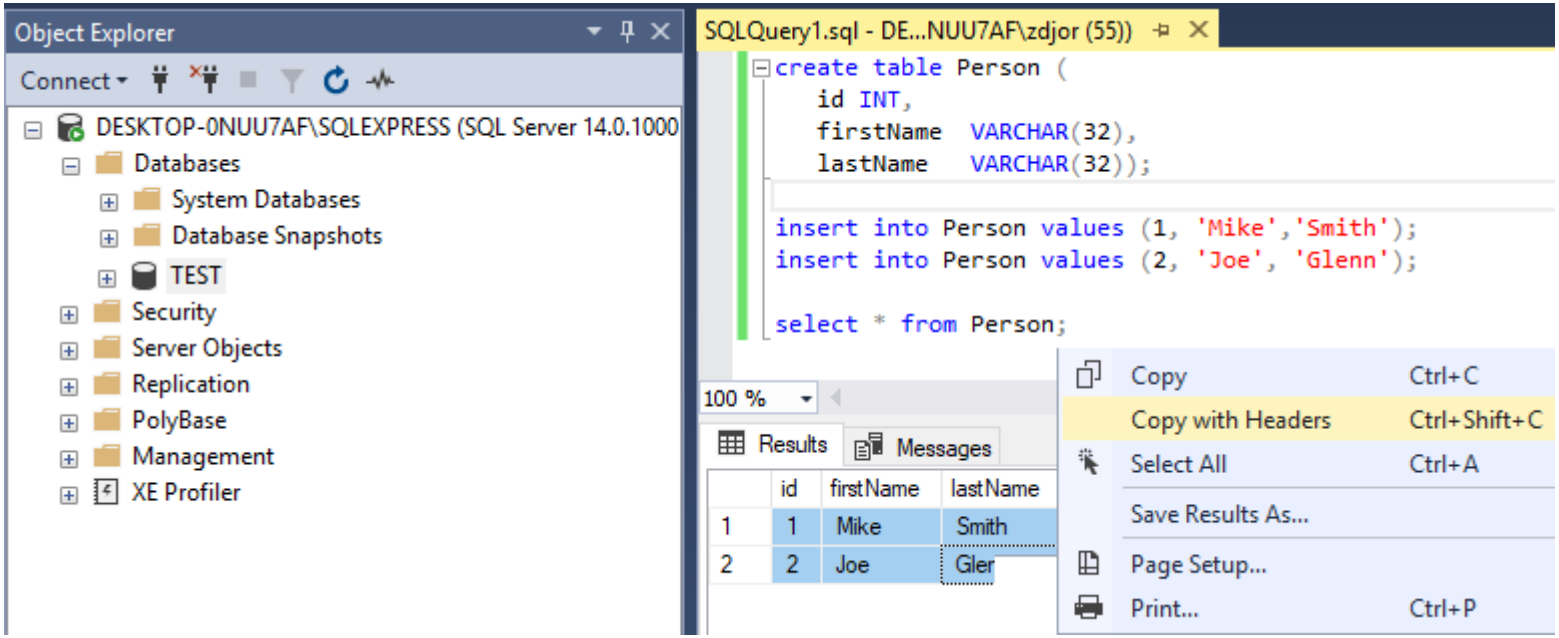




- Enter the name of the database, e.g. Test, and hit OK

Create a Table, Insert and Query Data

- Select your new database, on the list of databases in the Object Explorer and then select New Query on the top menu. In the editor that opens, a command to create a table, insert a row or two of data and then query that table. Every command can be highlighted and executed separately by hitting `Execute` on the top command bar .



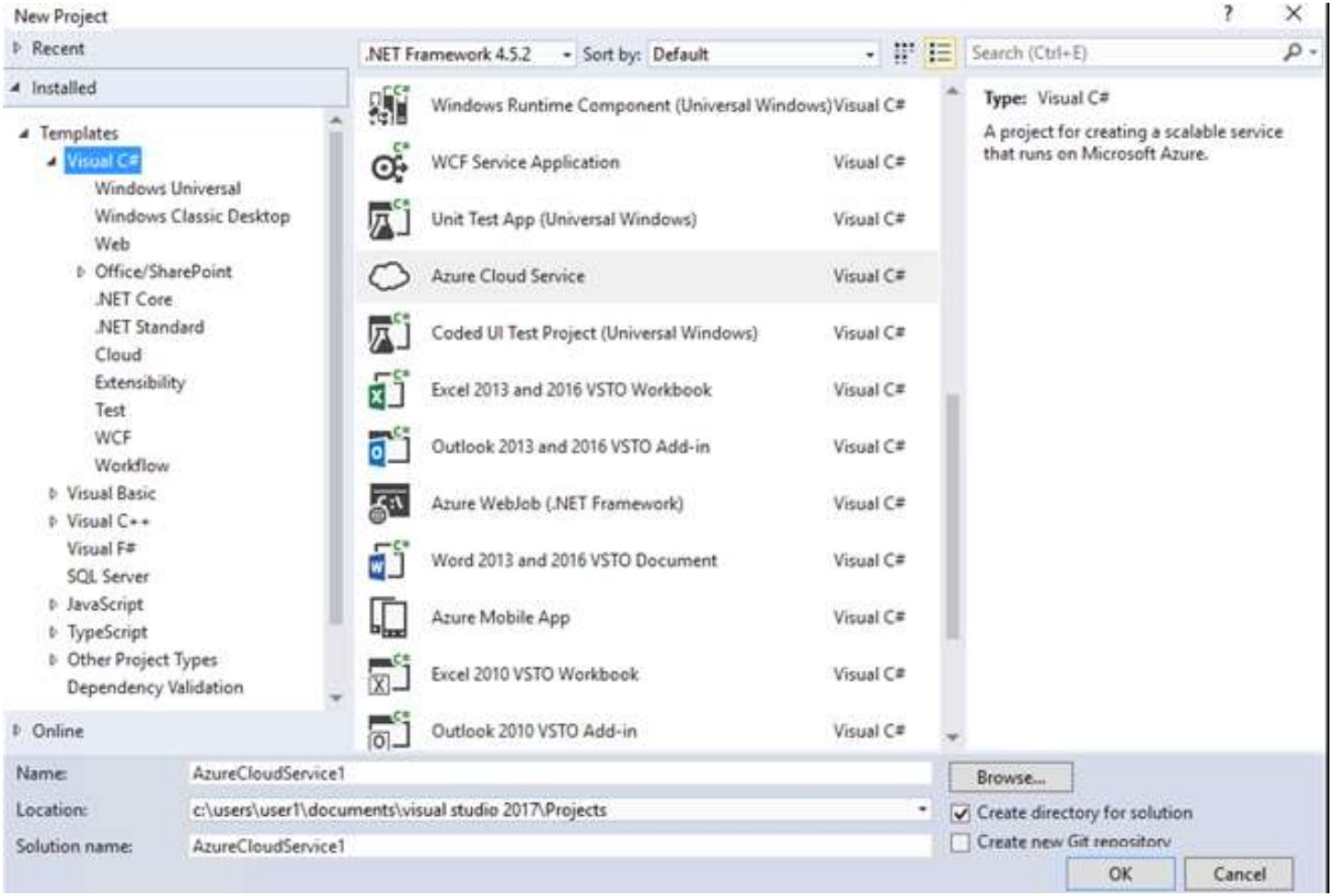
- You can save your queries by selecting `File -> Save SQLQuery1.sql as`
- You can save results of the query by selecting data, right clicking on selected region and then choosing Copy or Copy with headers.

Visual Studio 2017, Marketing

- The Visual Studio (2017) is an interactive development environment (IDE) that you can use to view and edit a wide variety of code, and then debug, build, and publish apps for Android, iOS, Windows, the web, and the Cloud.
- Visual Studio is available for Mac and Windows.
- With VS 2017 you can create:
  - Apps for an Android and IPhone,
  - Video games using C++,
  - Websites,
  - Desktop apps,
  - Apps for Office,
  - Business Applications for Windows servers and SQL Servers
  - Business Applications for the Cloud and multitude of Cloud services.
- In VS 2017 you can develop your own code or you could modified code produced by other developers.
- 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.

Templates for Code Generation

- VS 2017 comes with templates for code generation and creation in many technologies. Those templates are accessible when you try to create new project in any of supported technologies:

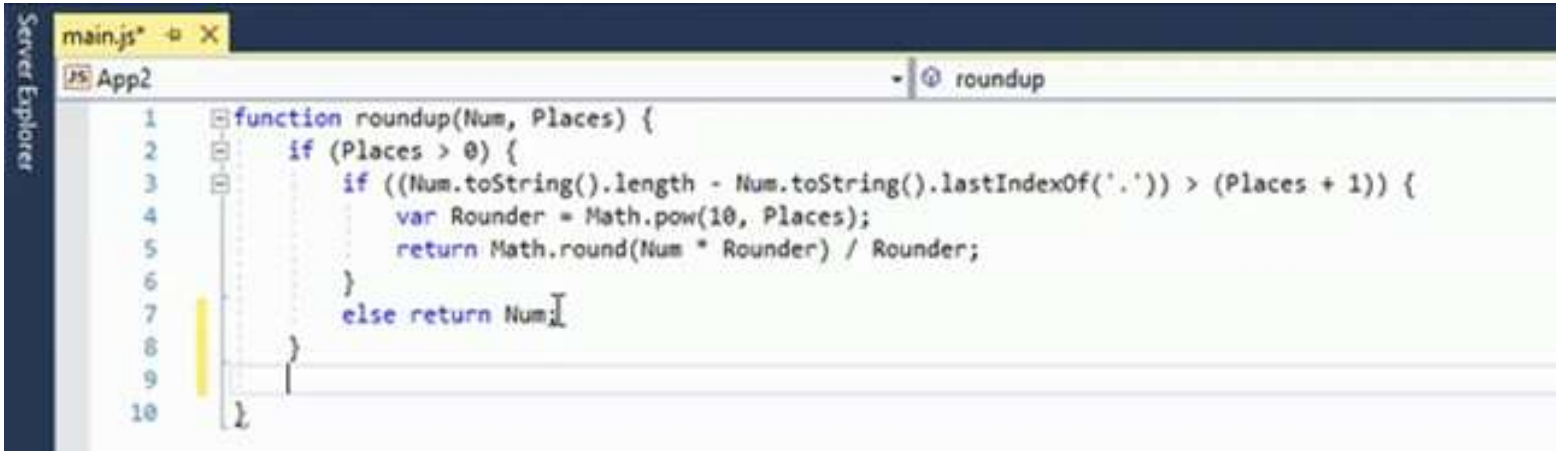


Create Cloud Apps for Azure

- 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.
- You 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.
- 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.

Create Apps for the Web using Sophisticated Editors

- The Web drives our modern world.
- 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.
- Fundamental Microsoft development frameworks: ASP.NET Core and .NET Core run on Windows, Mac, and Linux operating systems.

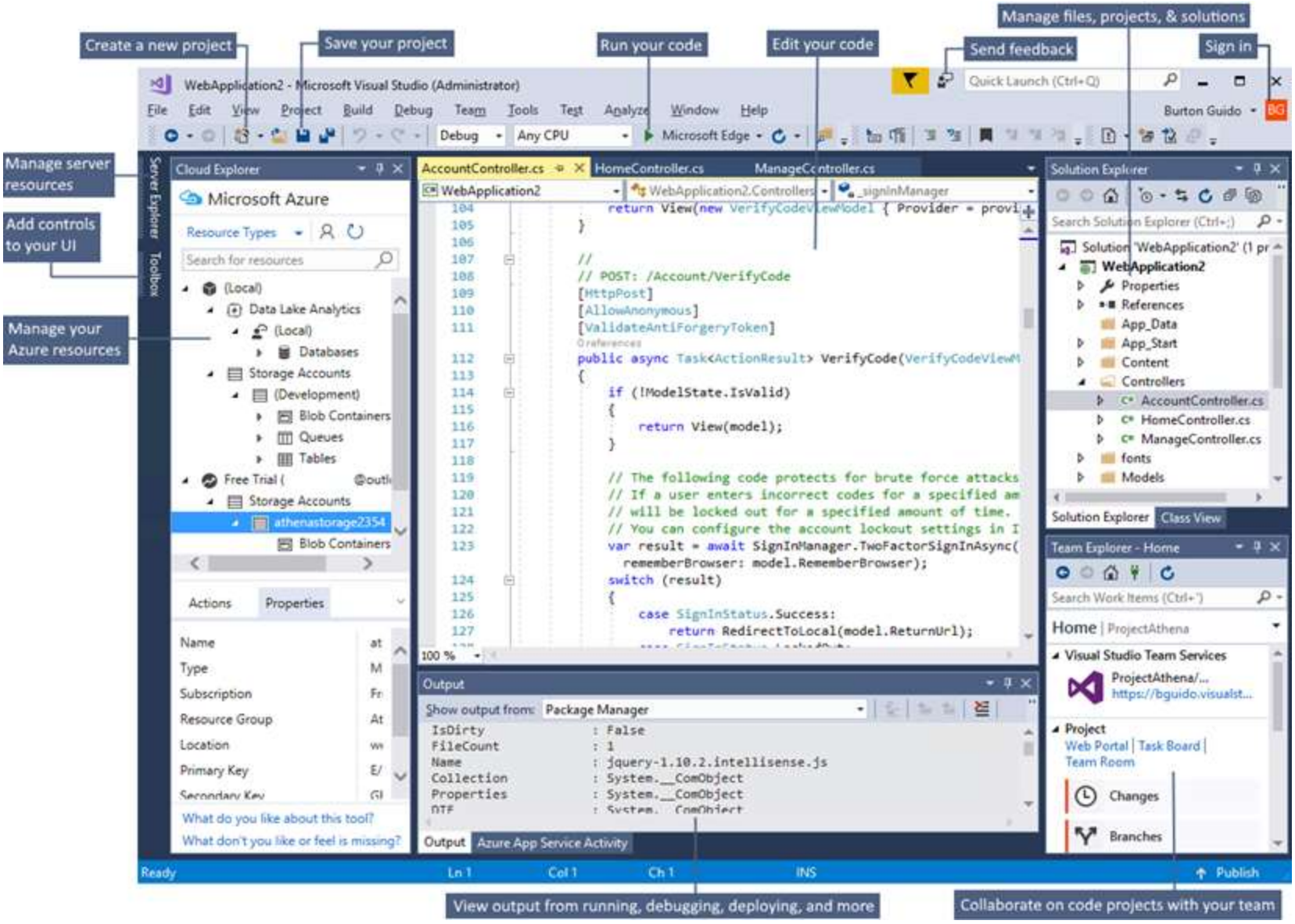


- We will use free version of Visual Studio called Community 2017.
- This version is lightweight compared with previous versions of VS. The new modular installer enables you to choose and install *workloads*, which are groups of features needed for the programming language or platform you prefer.

Quick Tour of IDE

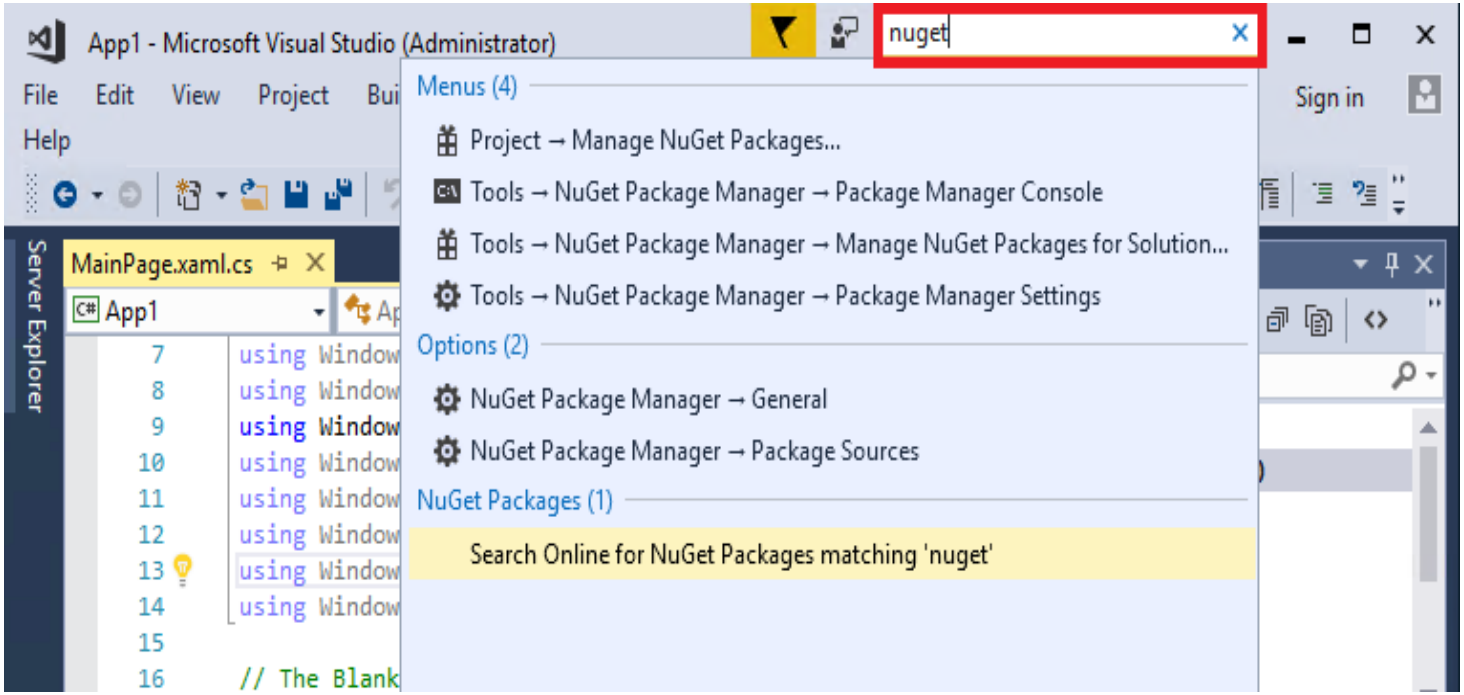
- The image on the next slide shows Visual Studio with an open project along with several key tool windows you will most likely use:
- Solution Explorer lets you view, navigate, and manage your code files. Solution Explorer can help organize your code by grouping the files into solutions and projects.
- The Editor window, where you'll likely spend a majority of your time, shows your code and enables you to edit source code and design a UI.
- The Output window is where Visual Studio sends its notifications, such as debugging and error messages, compiler warnings, publishing status messages, and more. Each message source has its own tab.
- 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. If a particular operation requires the Azure portal, Cloud Explorer provides links that take you to the place in the Azure portal you need to go.

Modules and Panels in VS



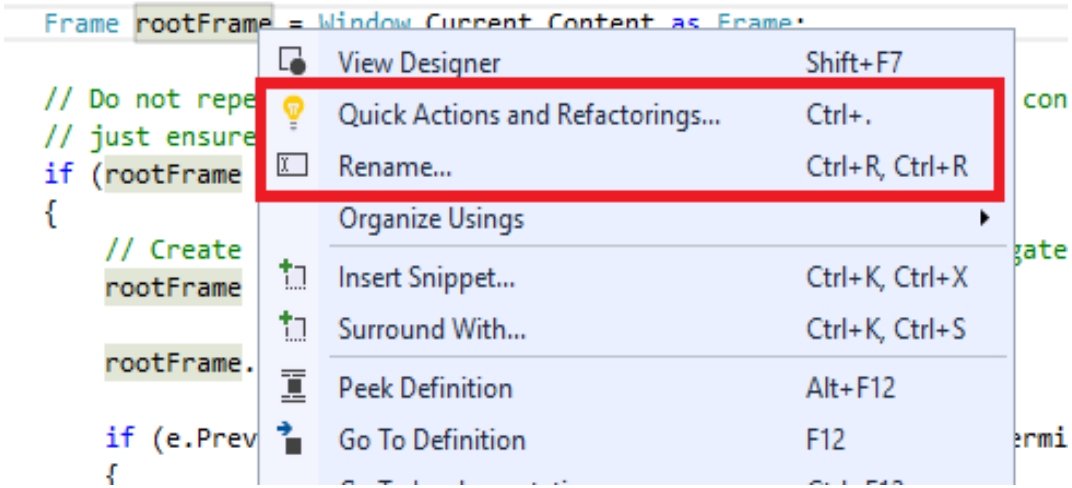
Quick Launch

- 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 lists results 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.



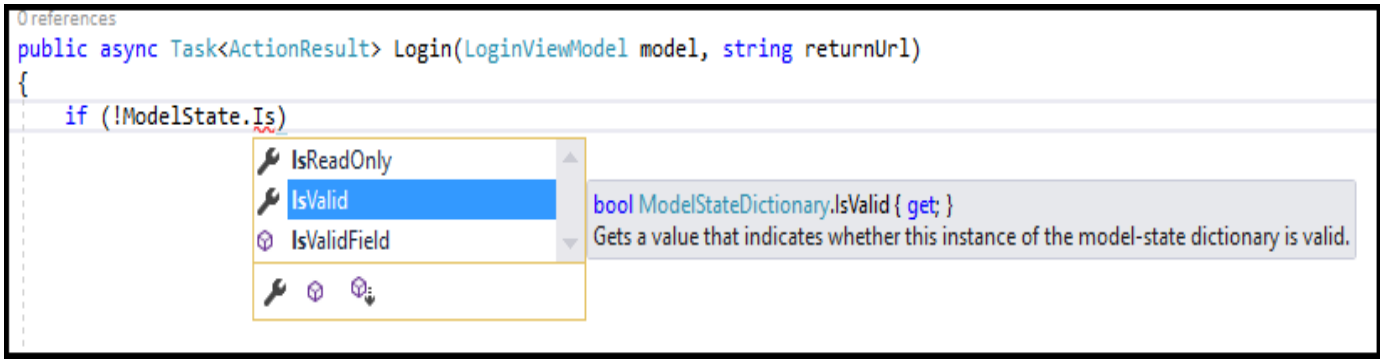
Refactoring

- 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.



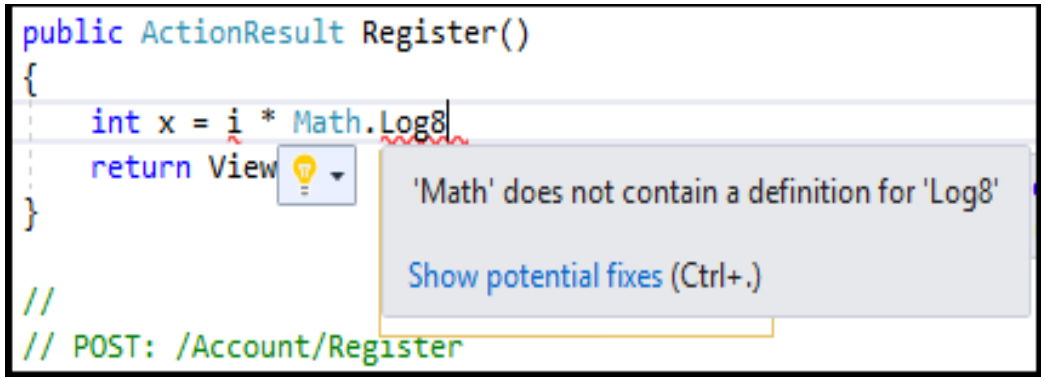
IntelliSense

- 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.
- IntelliSense exists for Visual C#, Visual C++, JavaScript, and Visual Basic.
- The following illustration shows some IntelliSense at work:



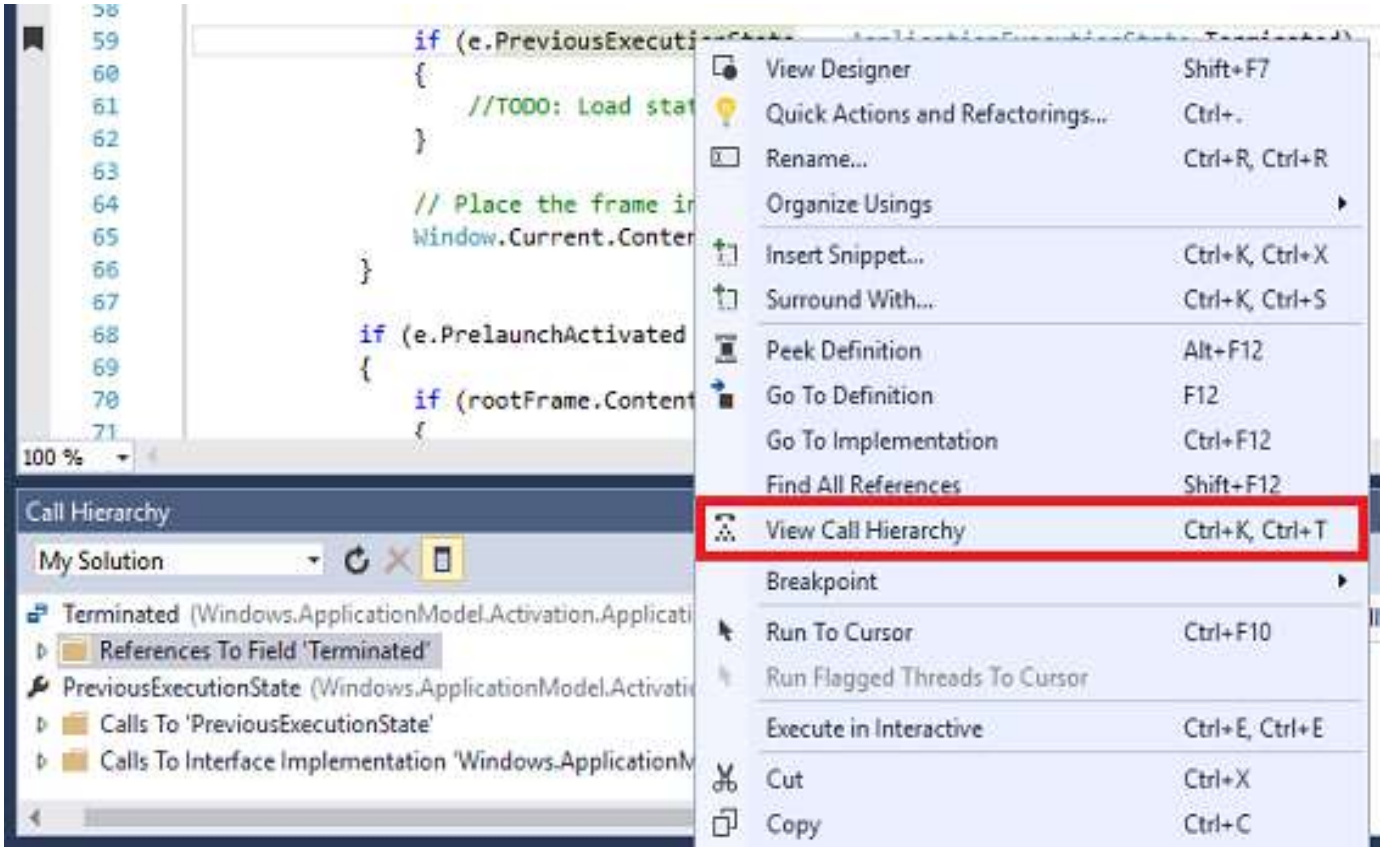
Squiggles

- 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.



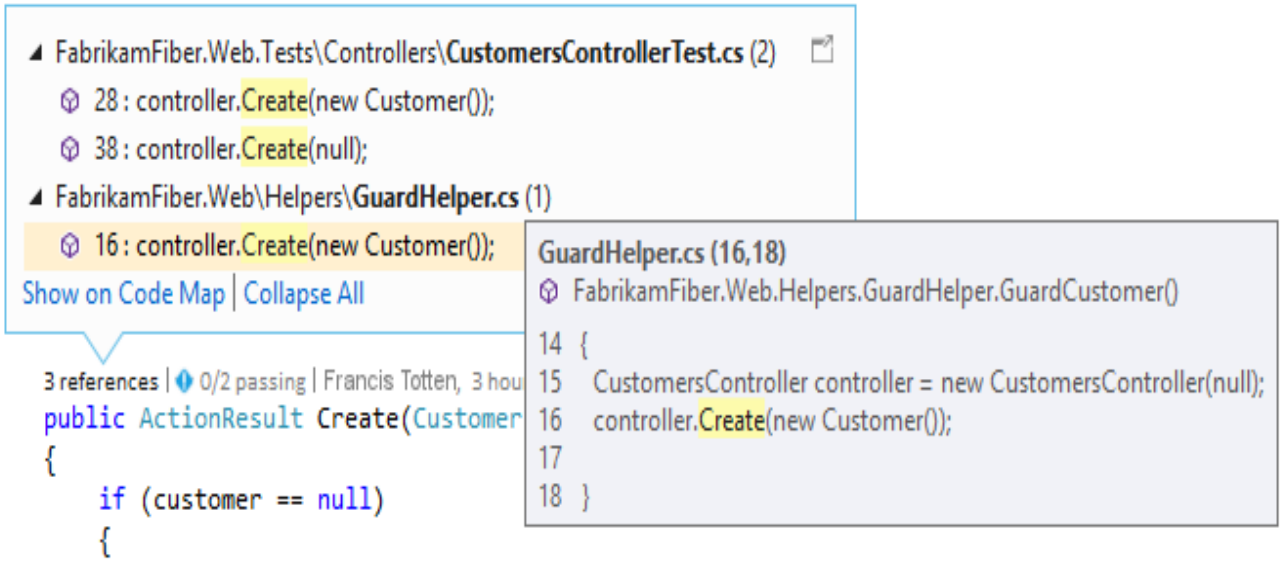
Call Hierarchy

- 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).



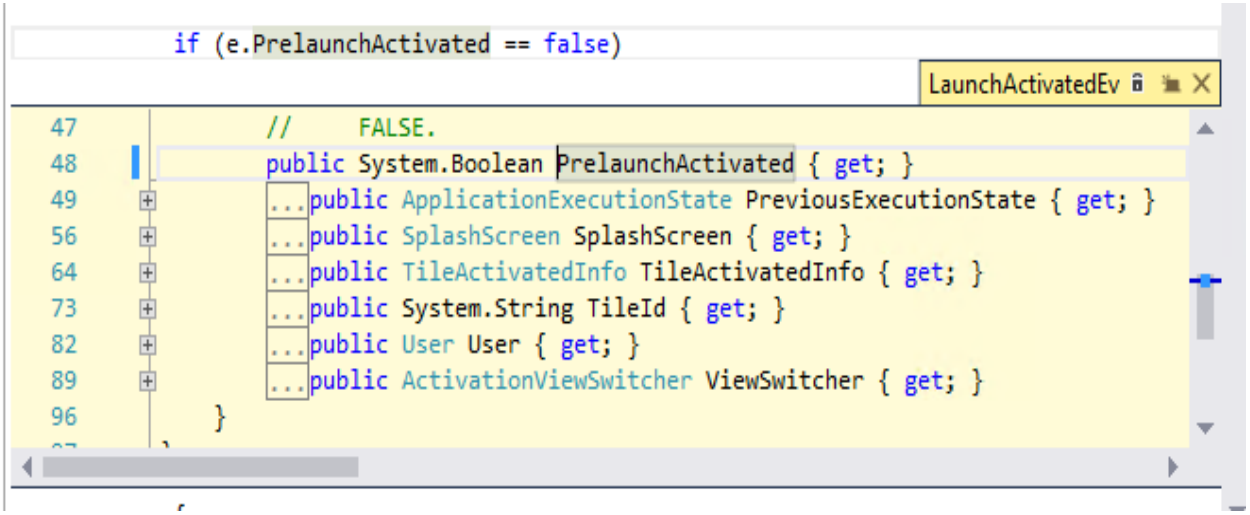
CodeLens

- 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.



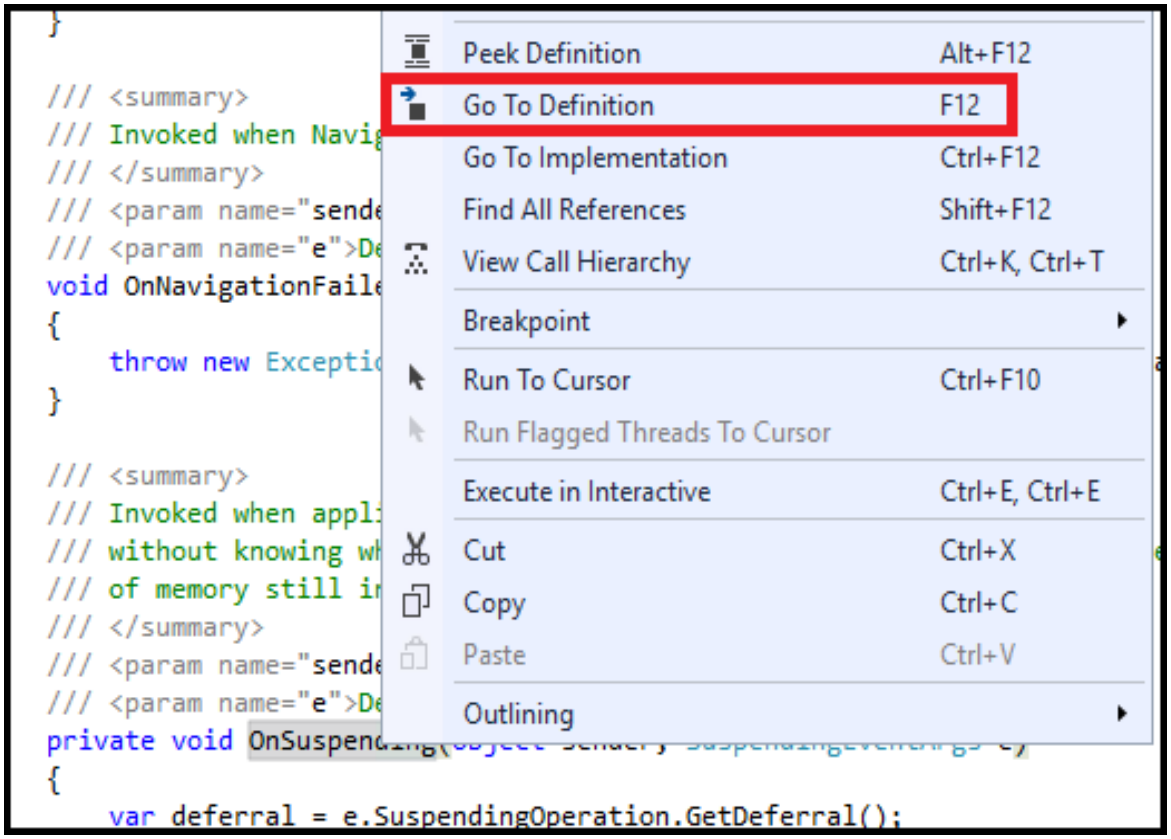
Peek Definition

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



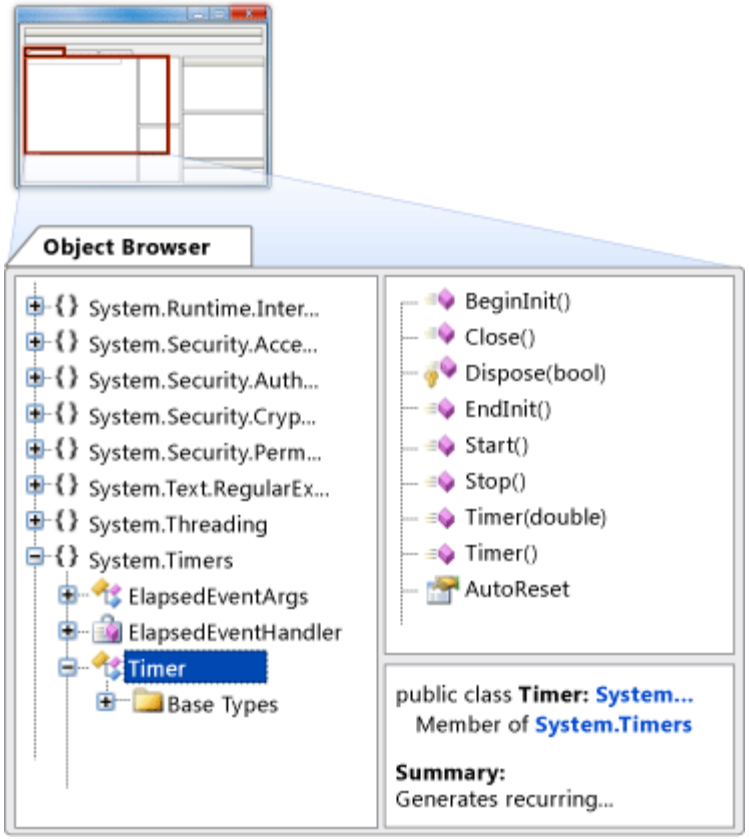
Go To Definition

- 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



Object Browser

- 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, etc.) those types contain.

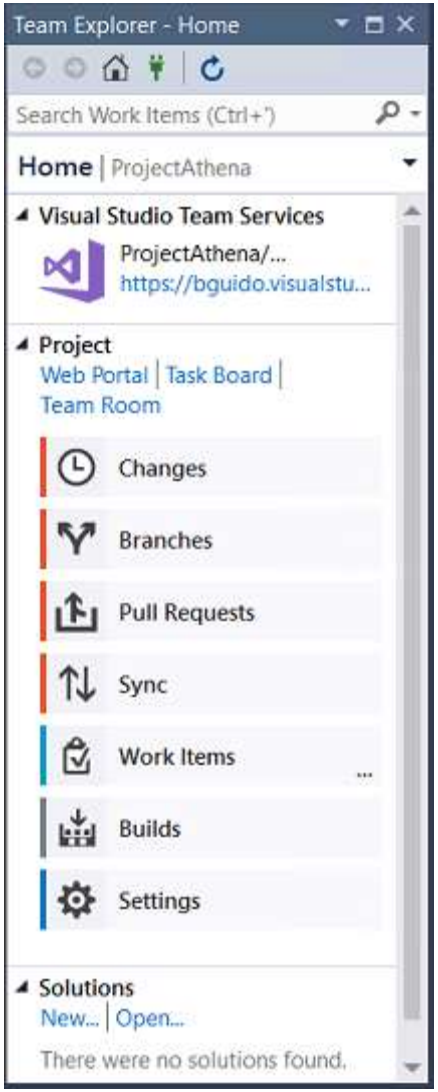


Source Code Management and Collaboration

- You can manage your source code in Git repositpry hosted by any provider, including GitHub using Team Explorer
- You can also use Visual Studio Team Services (VSTS) to manage code including bugs and work items for your whole project.
- Visual Studio also has other built-in source control features.
- 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**.

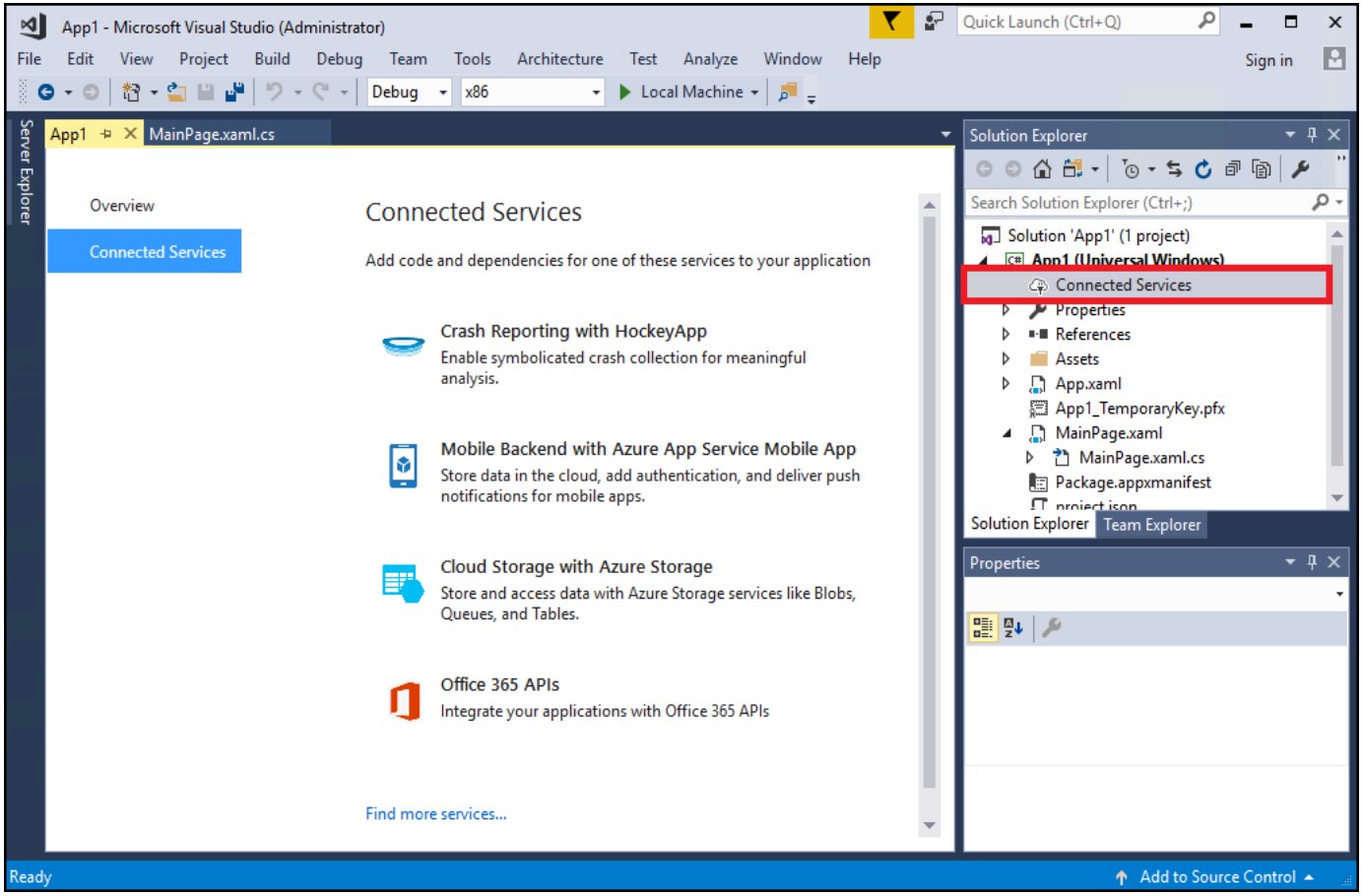
Team Explorer Windows

- This image shows a Team Explorer window for a solution hosted in VSTS



Connect to services, databases and Cloud resources

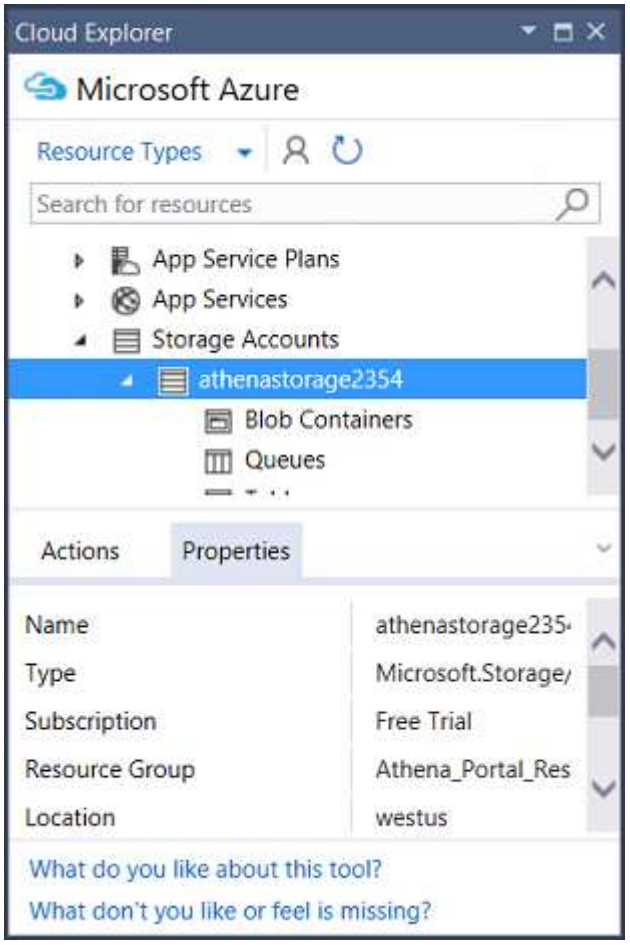
- Visual Studio provides you the means to leverage the Cloud.
- Connected Services feature of Visual Studio enables you to connect your app to services. Your apps can, for example, use the database service to store their data on Azure storage.



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

View Azure Resources

- 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

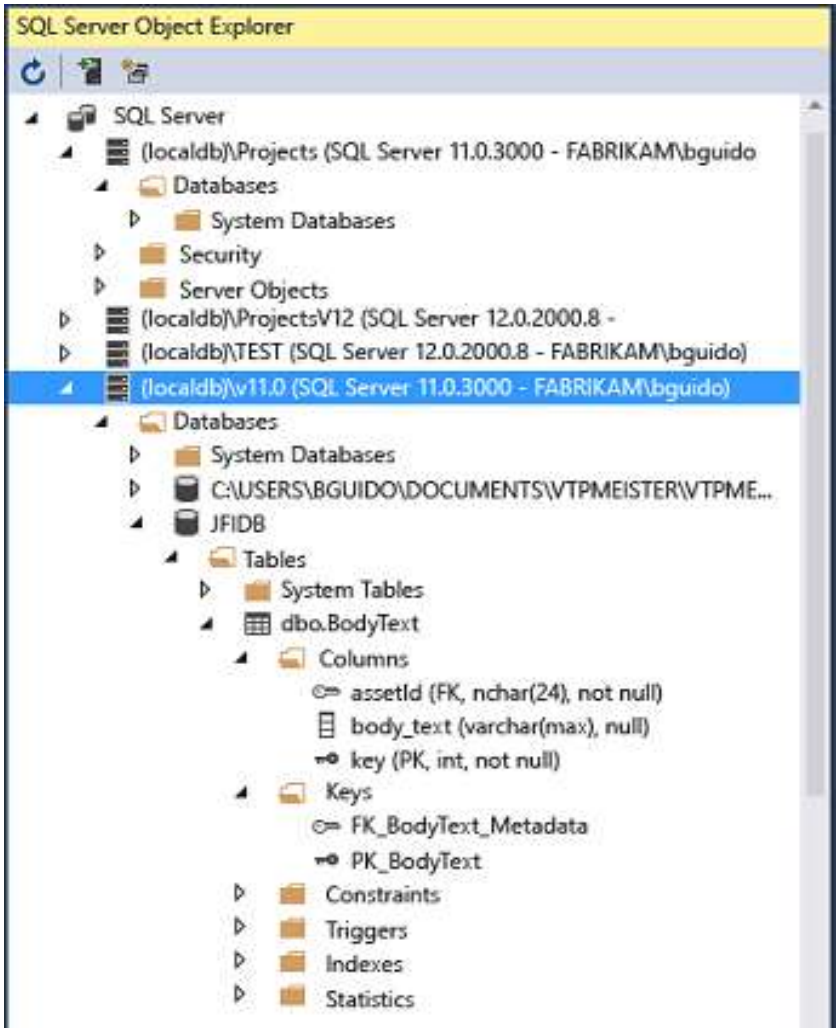


Server Explorer

- Server Explorer helps you browse and manage SQL Server instances and assets locally, remotely, and on Azure, Salesforce.com, Office 365, and websites. To open Server Explorer, on the main menu, choose **View**, **Server Explorer**.
- SQL Server Data Tools (SSDT) is a powerful development environment for SQL Server, Azure SQL Database and Azure SQL Data Warehouse.
- SSDT and SSMS 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-premise

Server Object Explorer

- 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



ASP.Net Core

- ASP.NET Core is a cross-platform, high-performance, open-source framework for building modern, cloud-based, Internet-connected applications. With ASP.NET Core, you can:
  - Build web apps and services, IoT apps, and mobile backends.
  - Use your favorite development tools on Windows, MacOS, and Linux.
  - Deploy to the cloud or on-premises
  - Run on .NET Core or .NET Framework.
- ASP.NET Core is a redesign of ASP.NET, with architectural changes that result in a leaner and modular framework.2
- ASP.NET Core provides the following benefits:
  - A unified approach to building web UI and web APIs.
  - Integration of modern client-side frameworks and development workflows.
  - A cloud-ready, environment-based configuration system.
  - Built-in dependency injection.
  - A lightweight, high-performance, and modular HTTP request pipeline.
  - Ability to host on IIS or self-host in your own process.
  - Can run on .NET Core, which supports true side-by-side app versioning.
  - Tooling that simplifies modern web development.
  - Ability to build and run on Windows, MacOS, and Linux.
  - Open-source and community-focused.

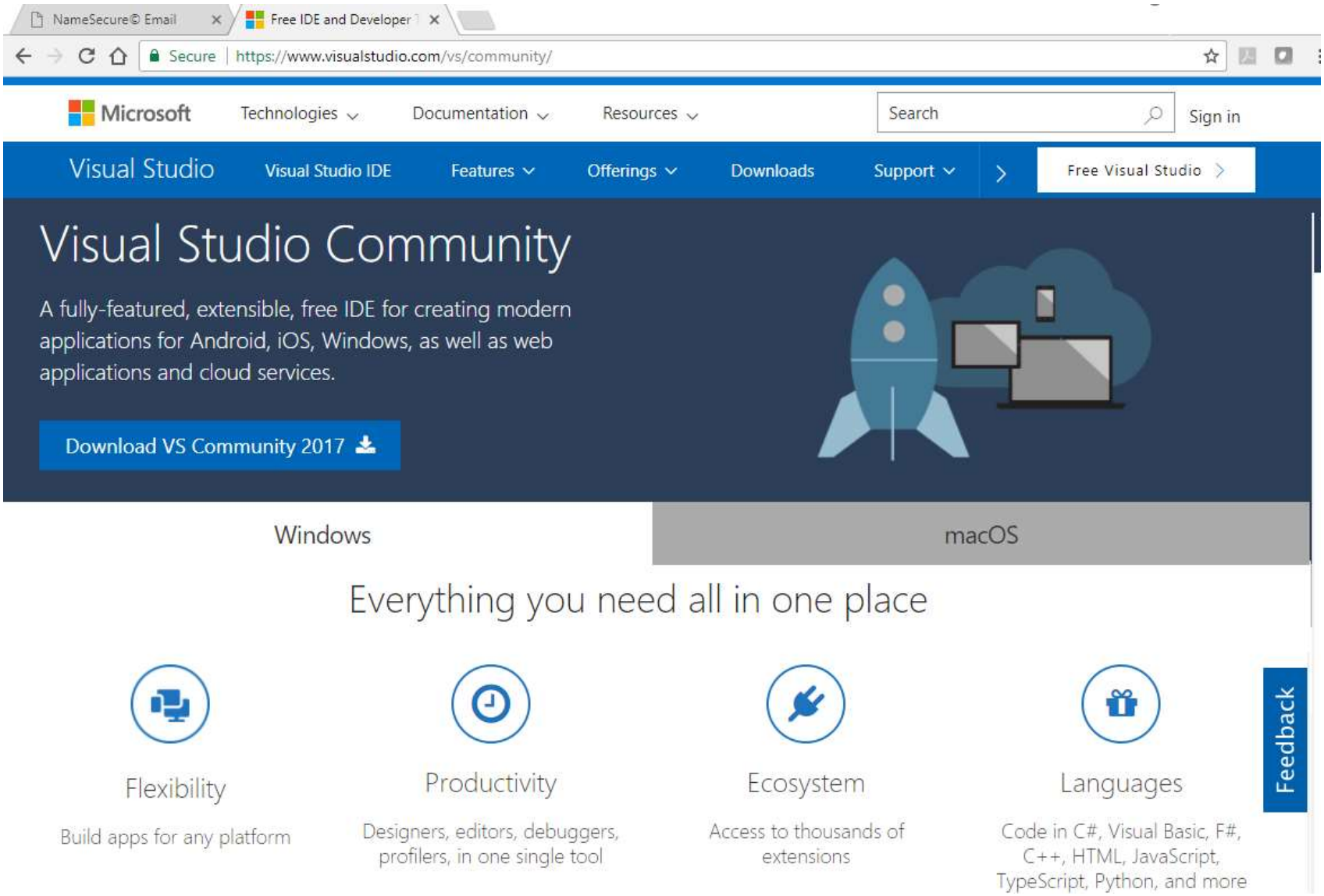
Web APIs and Web UI using ASP.NET Core MVC

- ASP.NET Core MVC provides features that help you build Web APIs and Web apps:
  - The Model-View-Controller (MVC) pattern helps make your web APIs and web apps testable.
  - Razor Pages (new in 2.0) is a page-based programming model that makes building web UI easier and more productive.
  - Razor syntax provides a productive language for Razor Pages and MVC Views.
  - Tag Helpers enable server-side code to participate in creating and rendering HTML elements in Razor files.
  - Built-in support for multiple data formats and content negotiation lets your web APIs reach a broad range of clients, including browsers and mobile devices.
  - Model Binding automatically maps data from HTTP requests to action method parameters.
  - Model Validation automatically performs client and server-side validation.

Deployment of Finished Applications

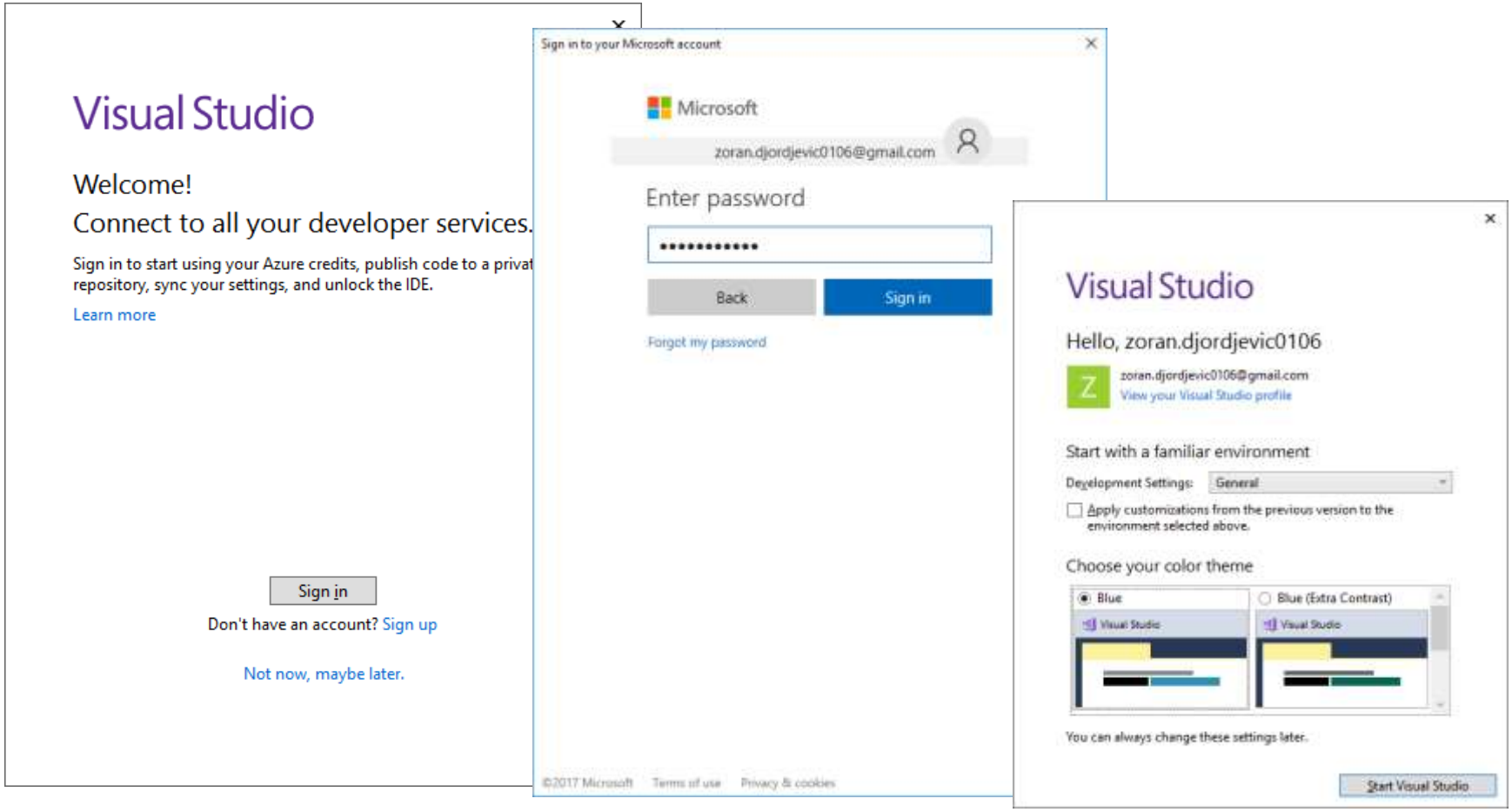
- When your application is developed and ready to deploy to users or customers, Visual Studio provides the tools to do that.
- Using Visual Studio you can deploy your apps to the Windows Store, to a SharePoint site, to the Cloud or with InstallShield or Windows Installer technologies.

Visual Studio Community Edition



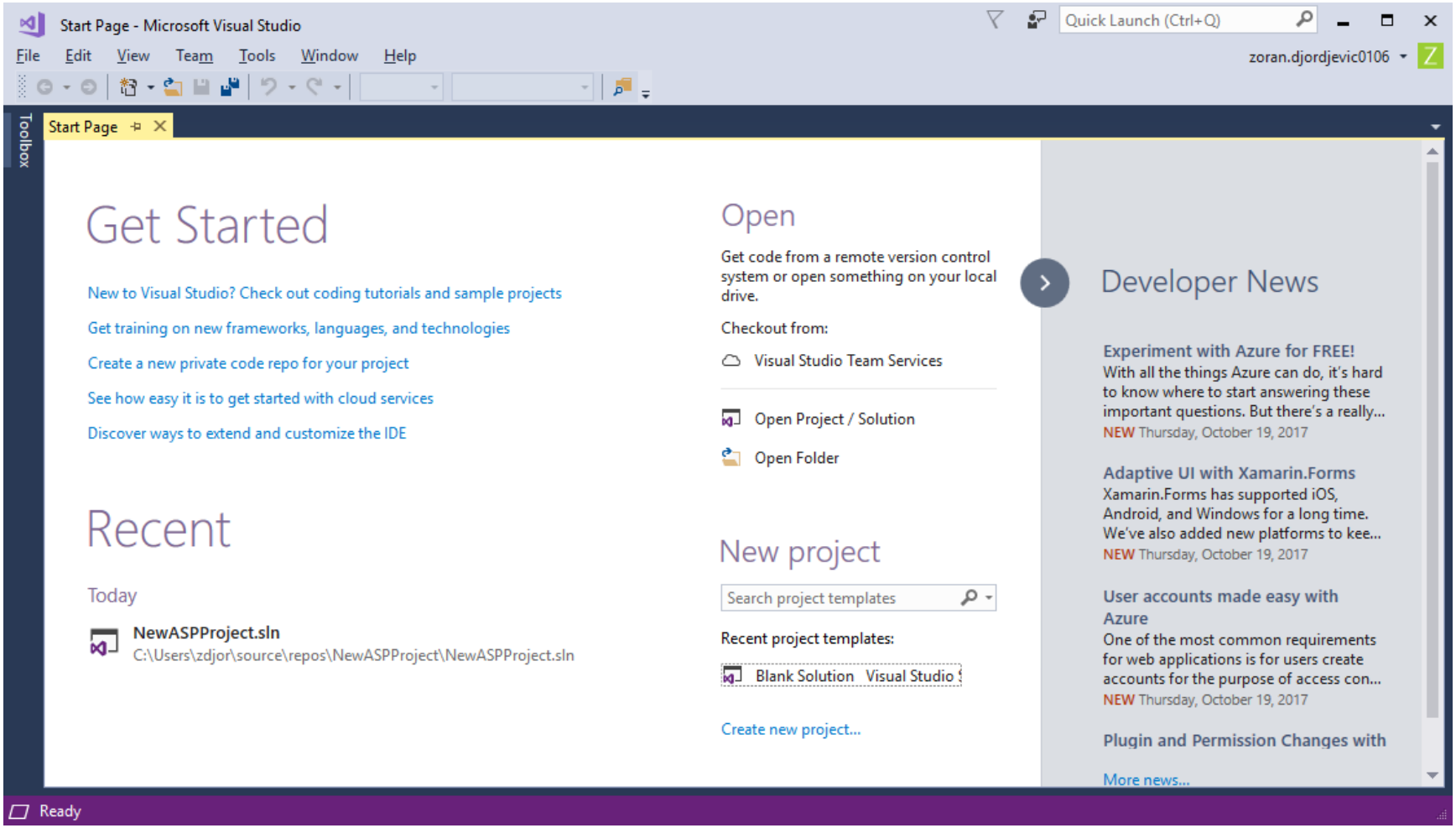
Use Azure Credits

- When you launch Visual Studio Community 2017 it will ask you whether you want to sign in into your Azure account. Do it. Hit `Sign in`, eventually `Start Visual Studio`



The First Page

- Your first page, when you open Visual Studio should look like the one below.
- You can get back to that page, if you ever have a reason to by doing `File > Start Page`

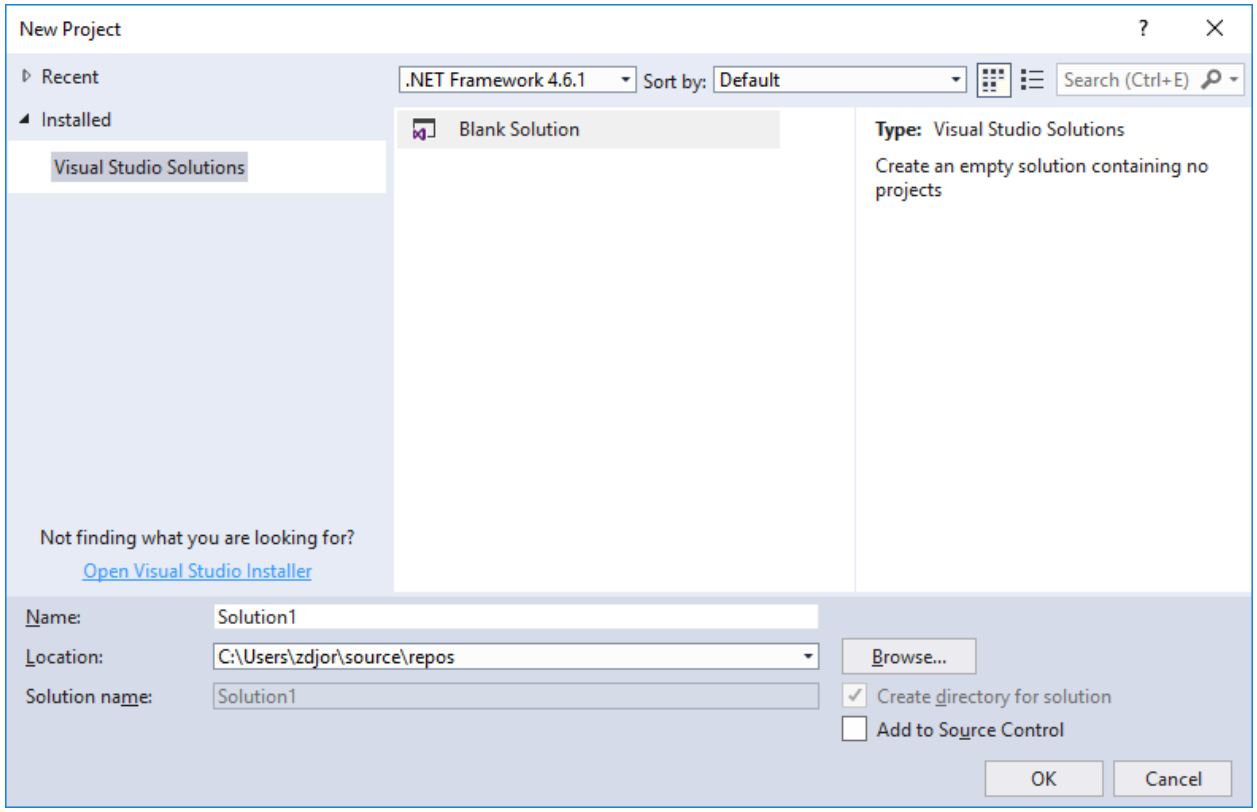


Creating a Web App

- As an example of the power of Visual Studio and its close integration with Azure, let us go through the process of creating a Web Application and then deploying it to Azure.
- Whatever you do in VS is a Project. If you have some work done, you will open an existing project. If you are starting new work, you will create new project, etc.

Try to Create a New Project

- Select `File > New > Project`. If you are really new to Visual Studio, you will get a page that is fairly empty and looks like this:

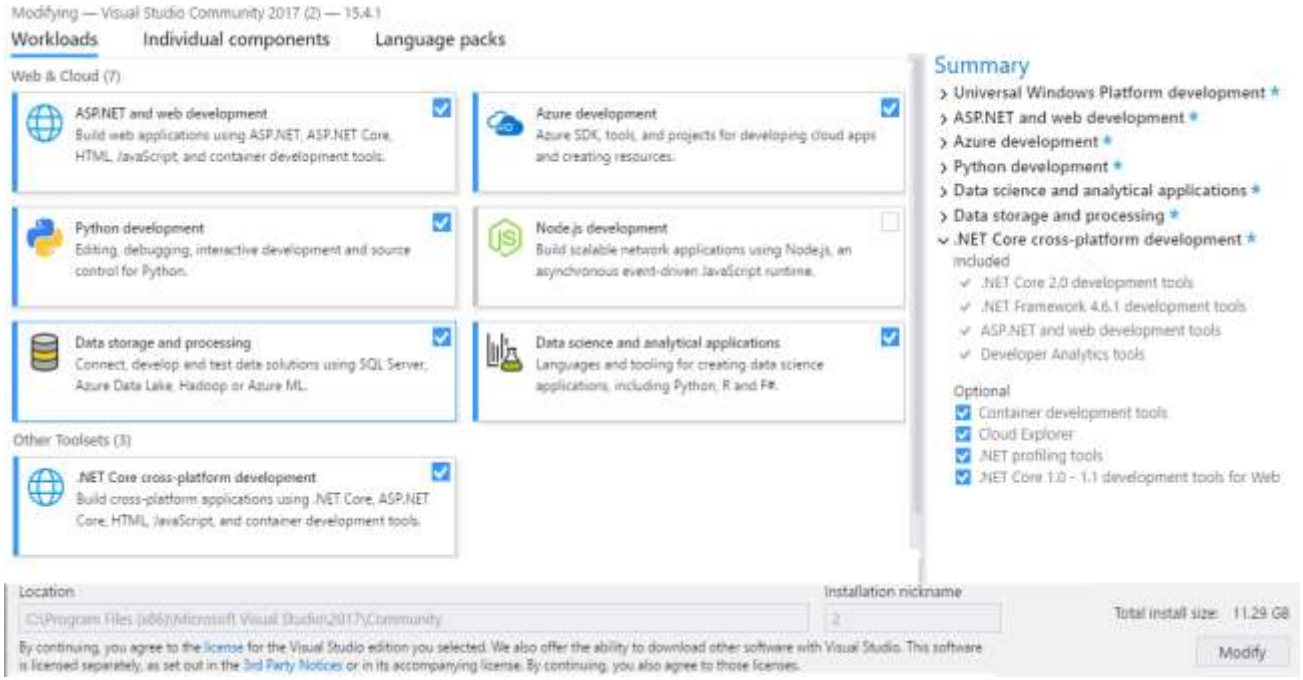


- You can not do much here. The issue is that the initial installation of your VS Community 2017 did not install all components you might want to use or need.

Visual Studio Installer

- Select “Open Visual Studio Installer”. You will get options to select from: Workload, Individual Components, Language Packs. Let us stick with Workloads and select:

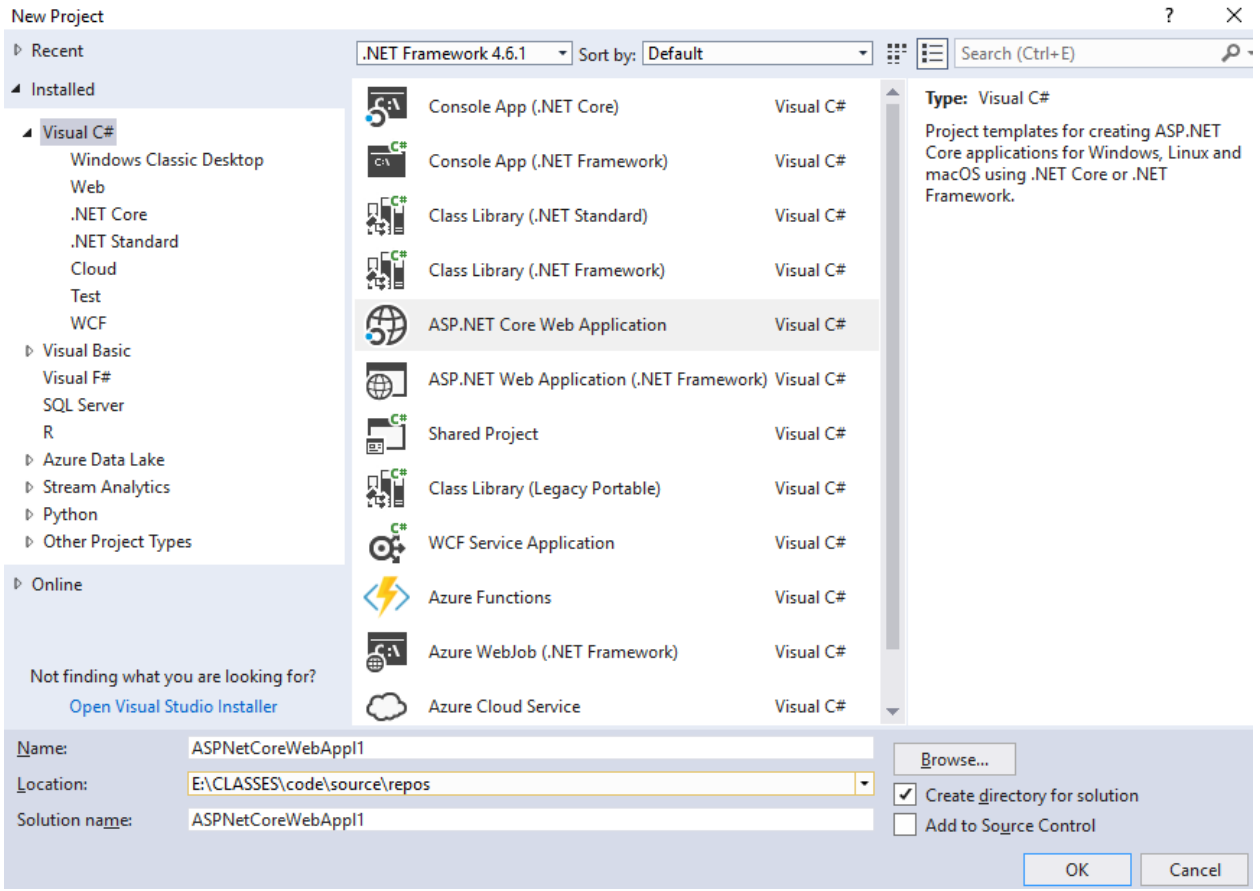
  - Universal Windows Platform ASP.NET and web
  - Azure development
  - Python development
  - Data science and analytical applications
  - Data storage and processing
  - .NET Core cross-platform



- At the bottom right you will see a Modify button. Press it. Then go shopping.

New Project for real

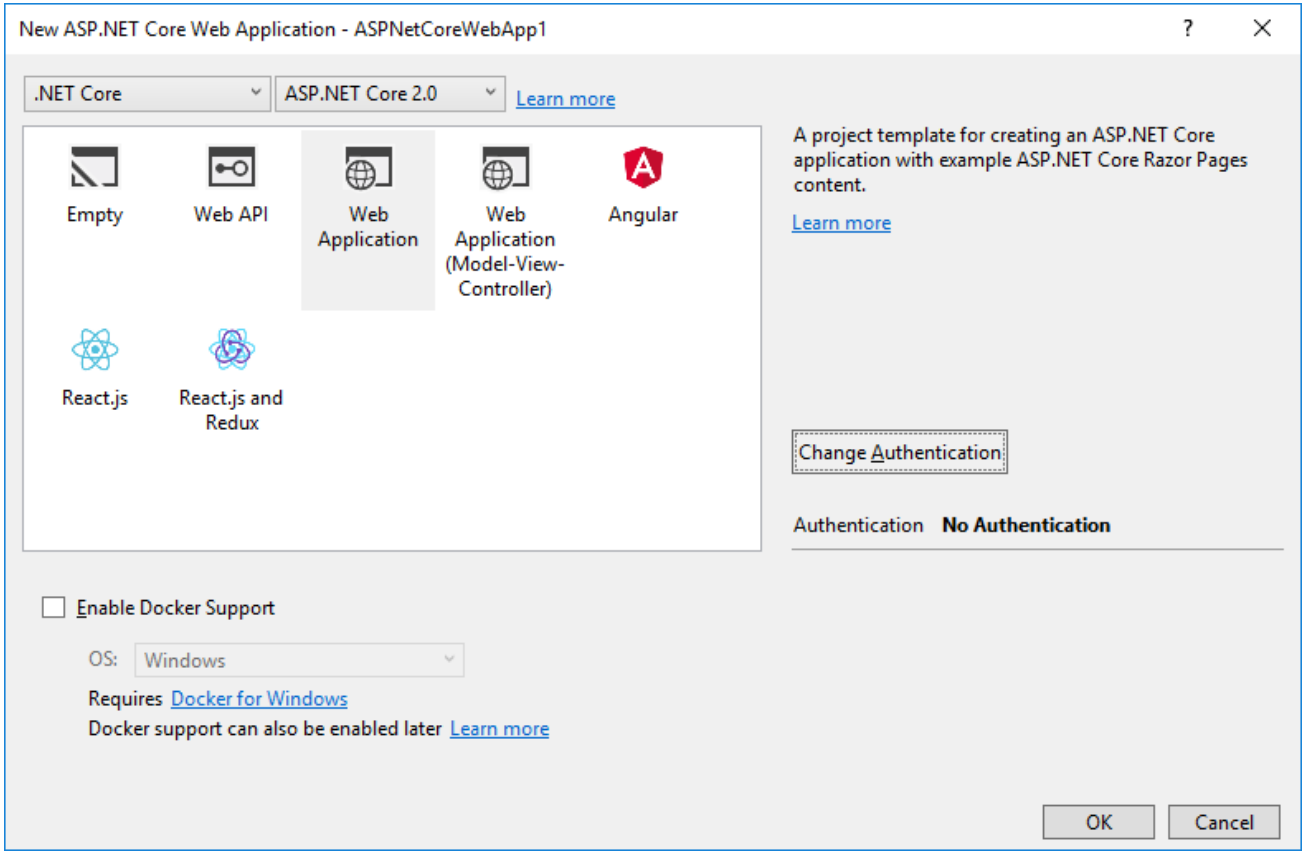
- Now that installation of all the components is done, you can go back to **New -> Project.**
- On the Left panel select **Visual C#**.
- In the center pane that pops up select **ASP.NET Core Web Application**.
- Provide a name for your application, e.g. ASPNetCoreWebApp1. You might want files associated with this project in a non-default location



- Select **OK**.
- One the following screen select Web Application

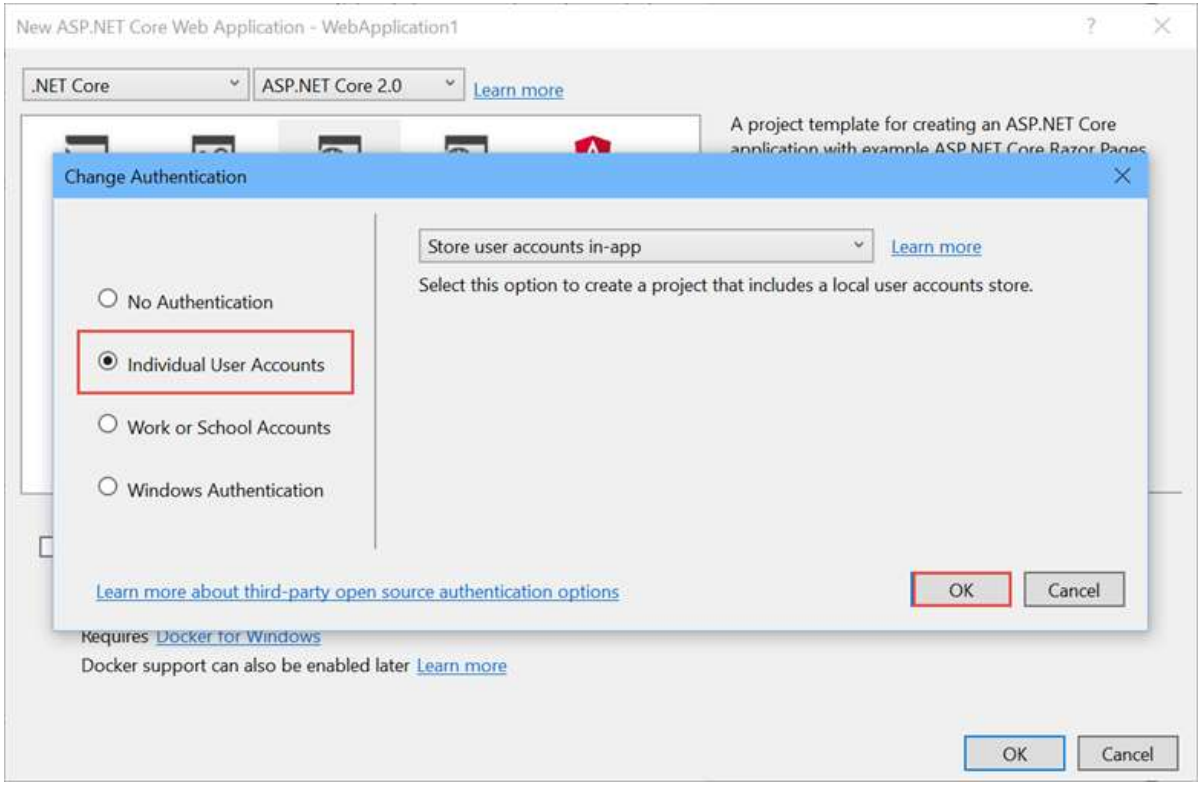
New ASP.NET Core Application

- Select **Web Application**.
- Select **Change Authentication**.



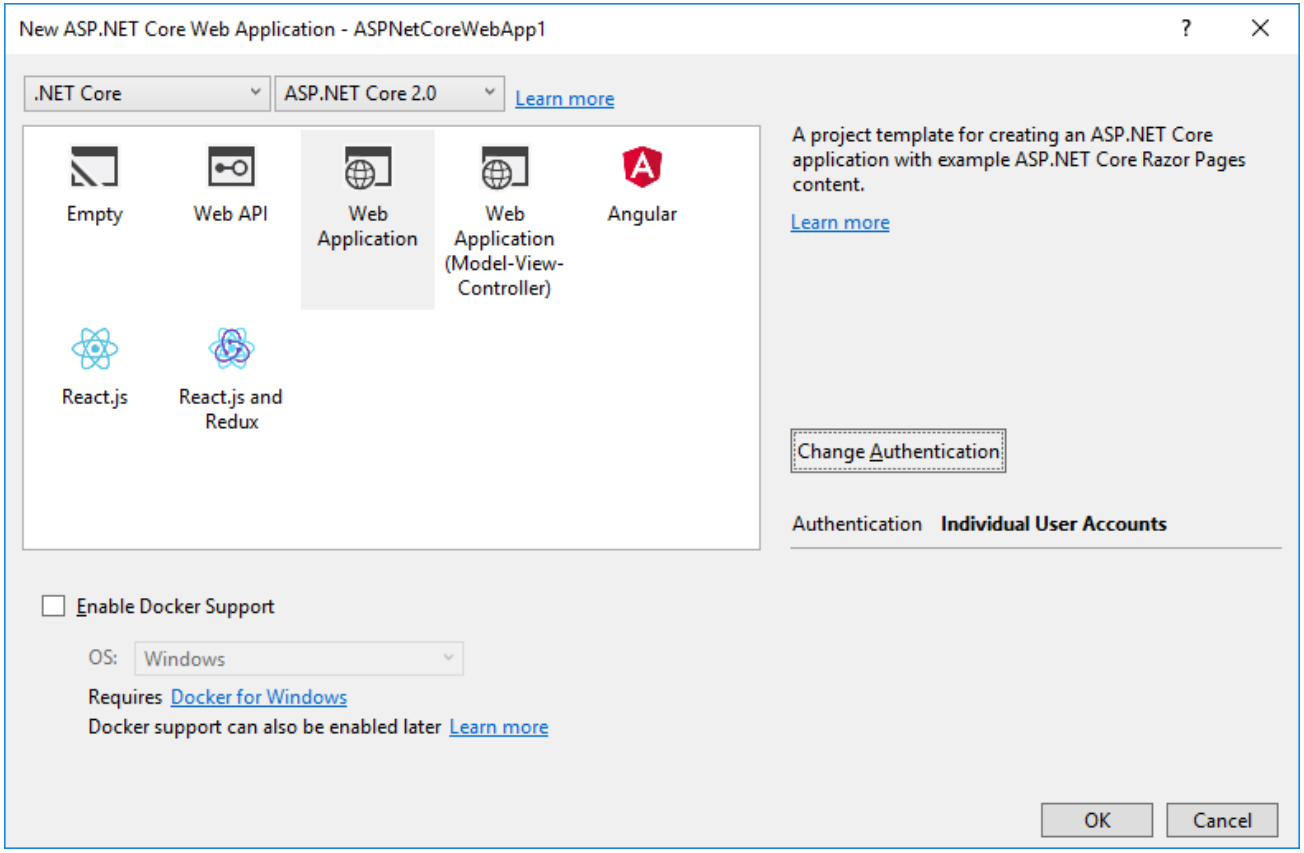
Change Authentication

- Select Individual User Accounts. Click OK.



New ASP.NET Core Web Application

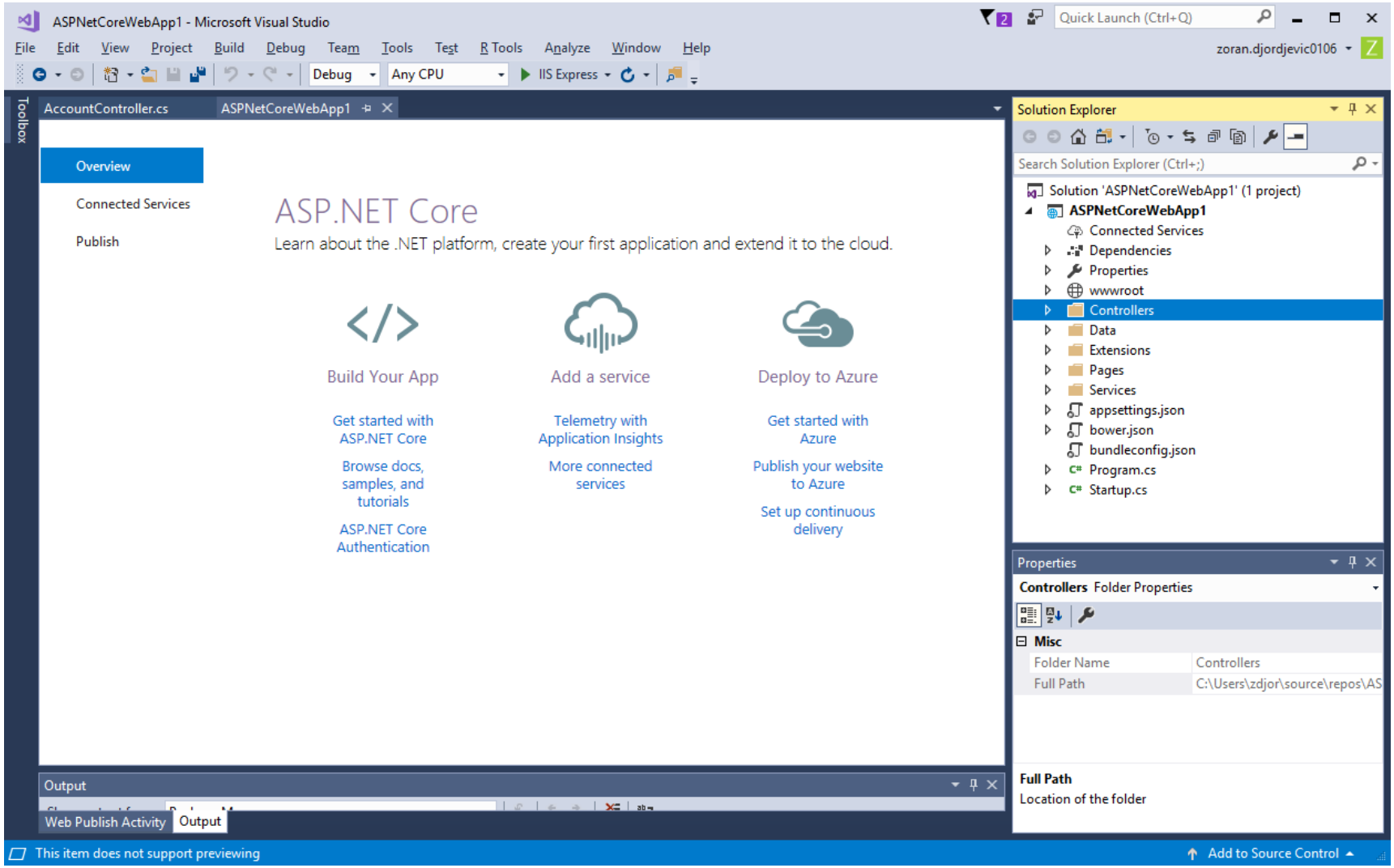
- You will return to the **New ASP.NET Core Web Application**, then
- Select **OK** again



- Visual Studio will generate `ASPNetCoreWebApp1` project and display its contents in Solution Explorer as seen on the next screen.

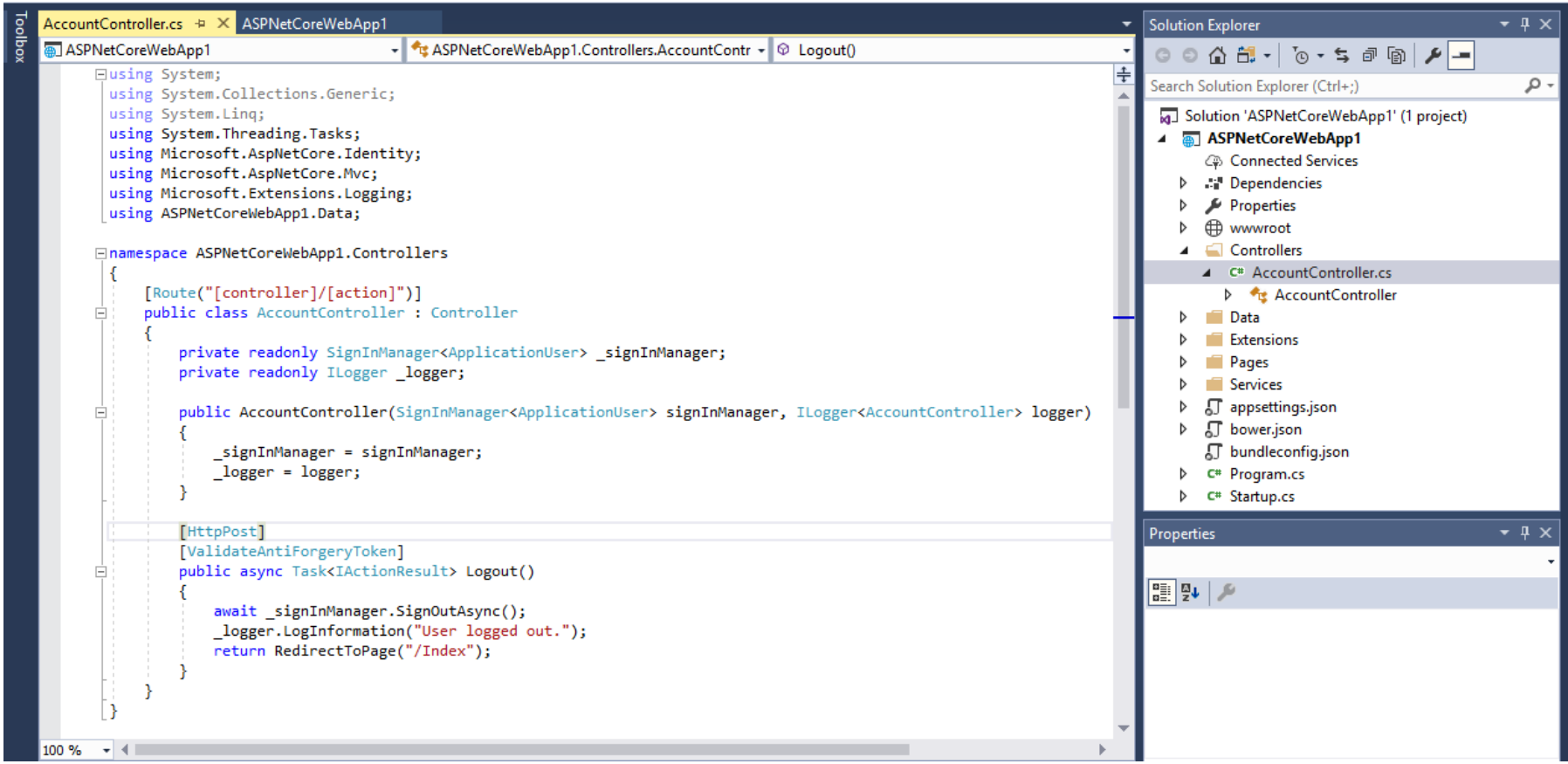
Your Application

- This is your first view of the application:



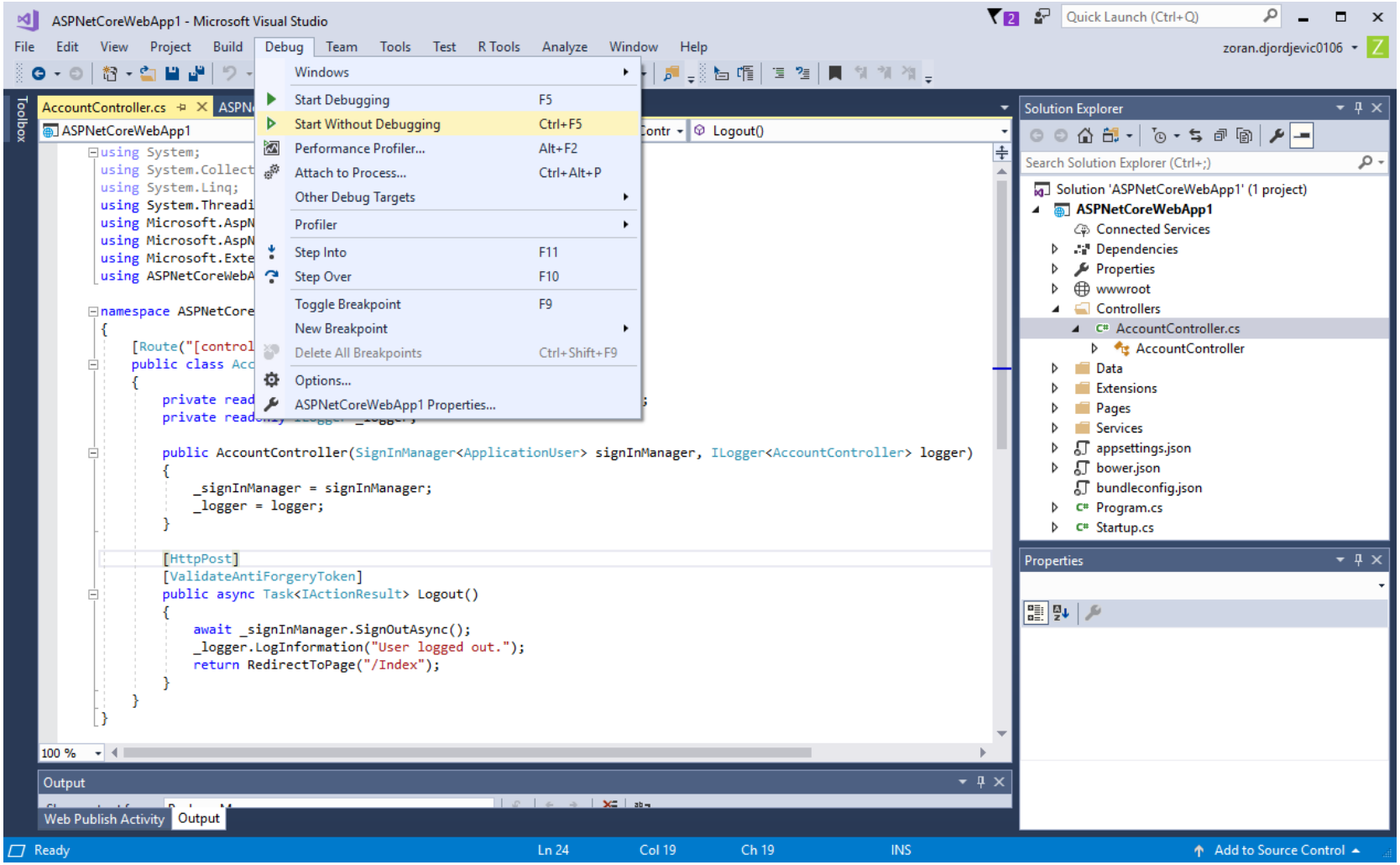
Explore your App

- If you click any of the packages, for example `Controllers` and then navigate into one of *.cs files and double click the file name, you will see generated C# code.
- C# is the main language of MS frameworks.



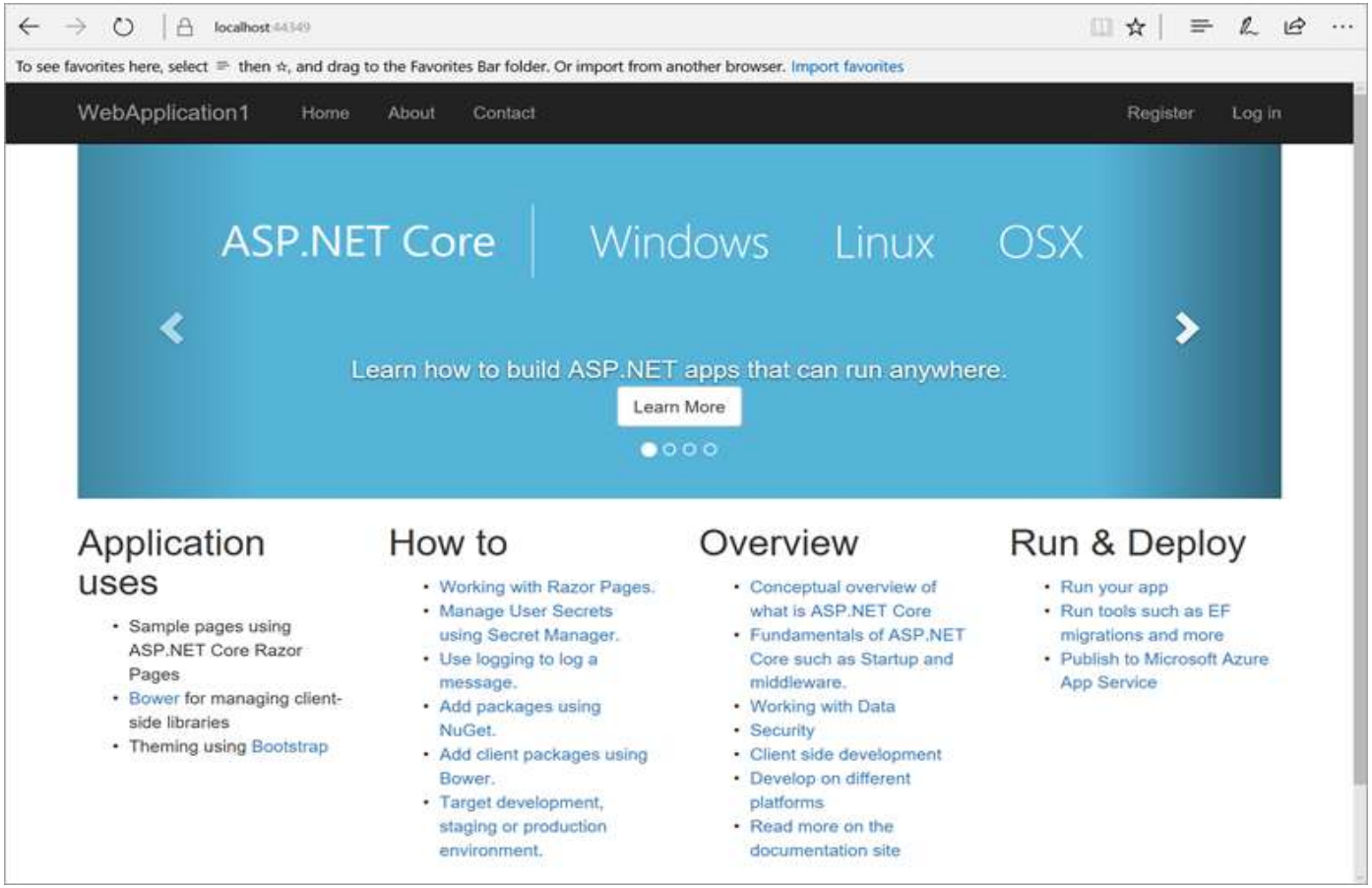
Try to Run the App Locally

- In the top menu choose **Debug,** and then **Start Without Debugging** to run the app locally.



Your app is published to IIS

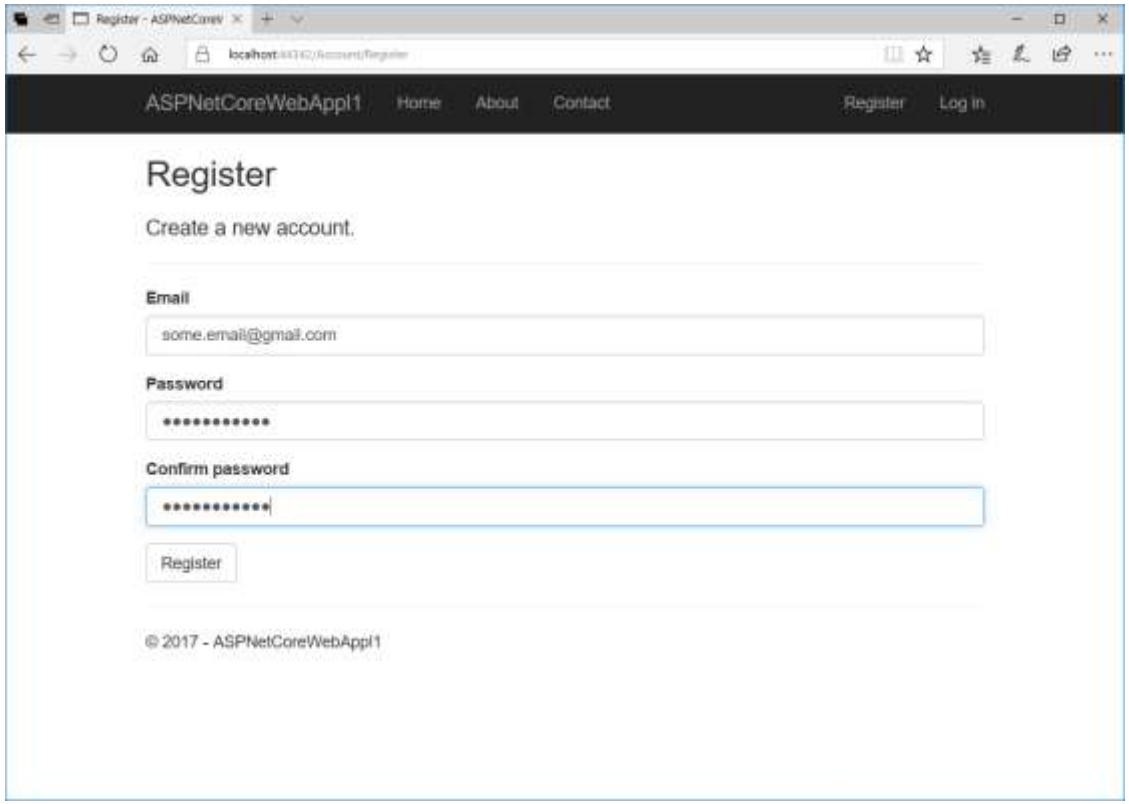
- Your default browser pops up on a seemingly random port.



- Click the **About** and **Contact** links to verify the web application works.
- This application is supposed to allow you to register users, so let us click Register link.

Register Users

- We select **Register** and register a new user. We will use a fictitious email address

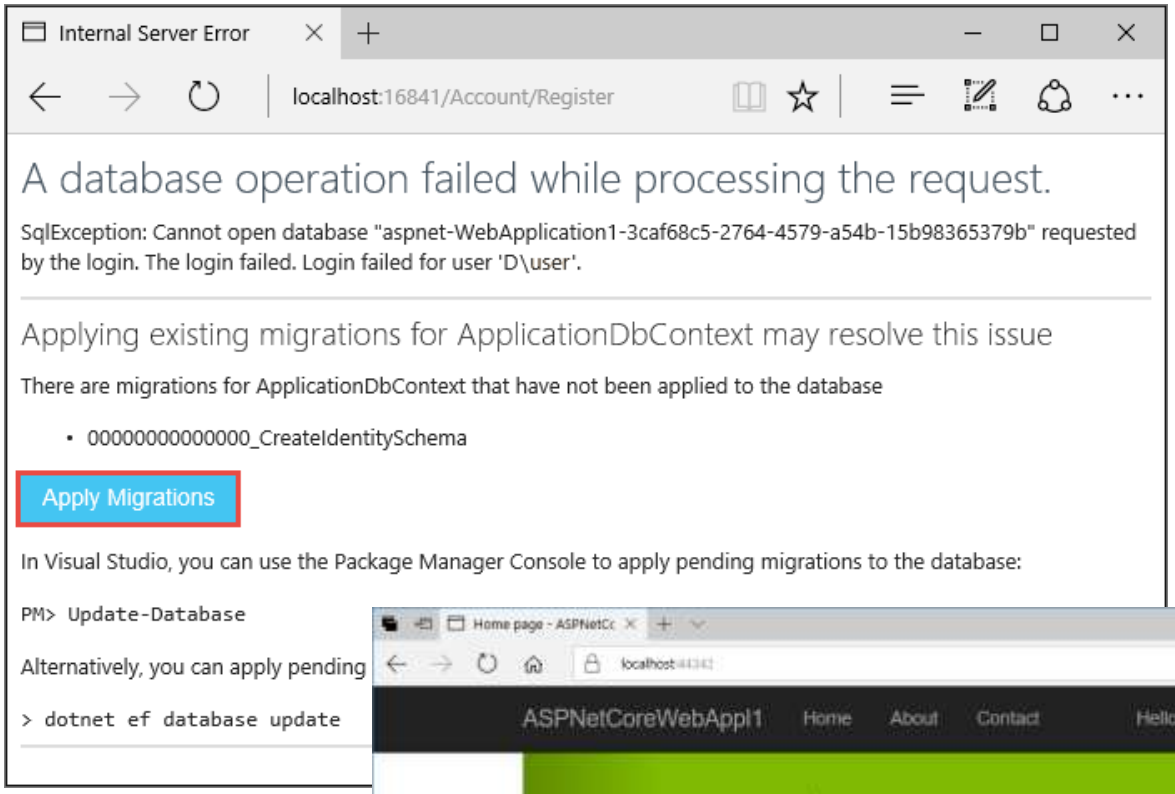


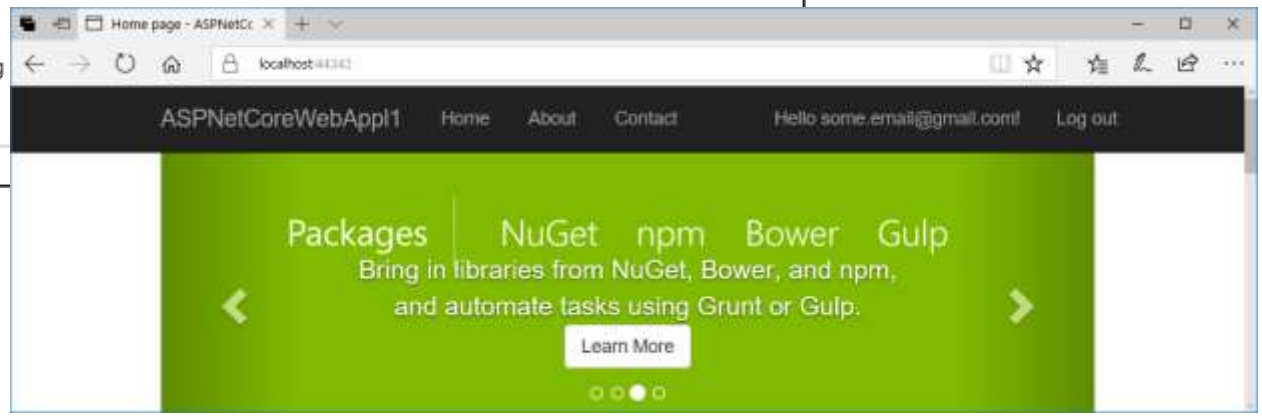
When you submit, hit Register, the page displays the following error:
"Internal Server Error: A database operation failed while processing the request. SQL exception: Cannot open the database. Applying existing migrations for Application DB context may resolve this issue."

The reason for this error is quite simple we provided no database where the email and password could be stored.

Apply Migrations

- This app, we know little about was build in such a way to deal with errors gracefully. So, hit Apply Migrations button, and after a while refresh the page.

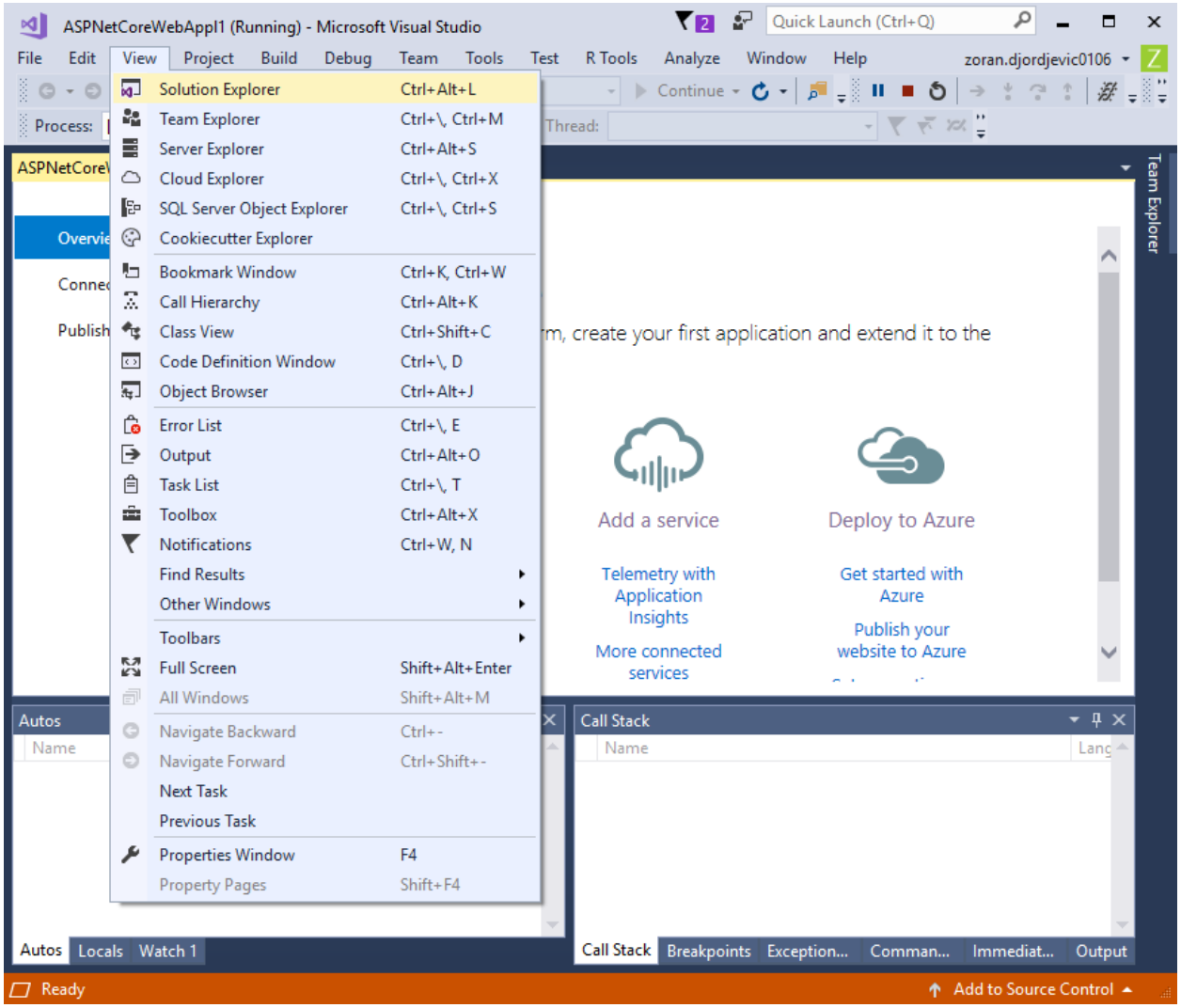




- The app reappears and displays the email used to register the new user and a **Log out** link.
- We need a database. Let us deploy this app to Azure and provision a database there.

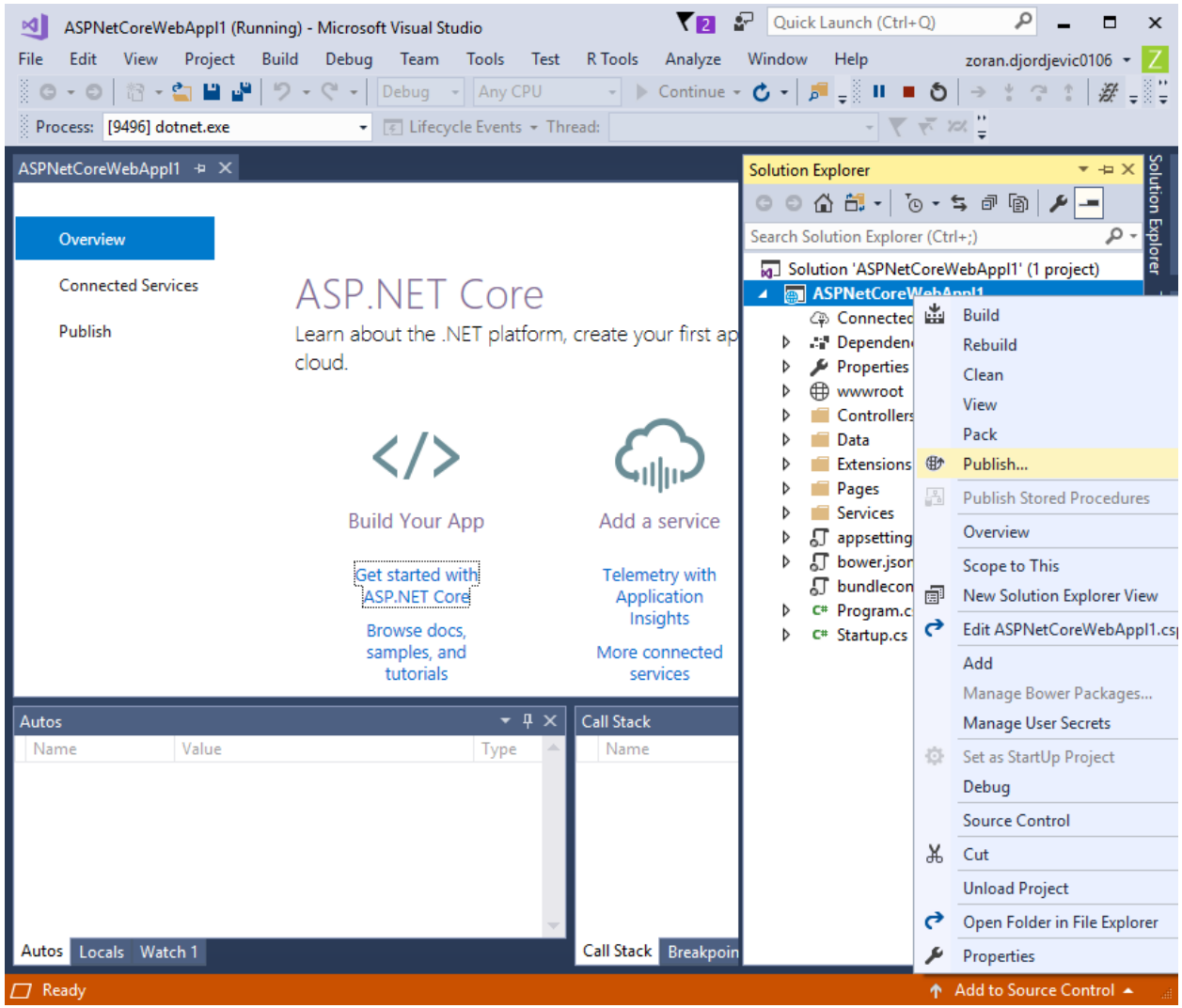
Find Solution Explorer

- If you do not see Solution Explorer, go to View -> Solution Explorer



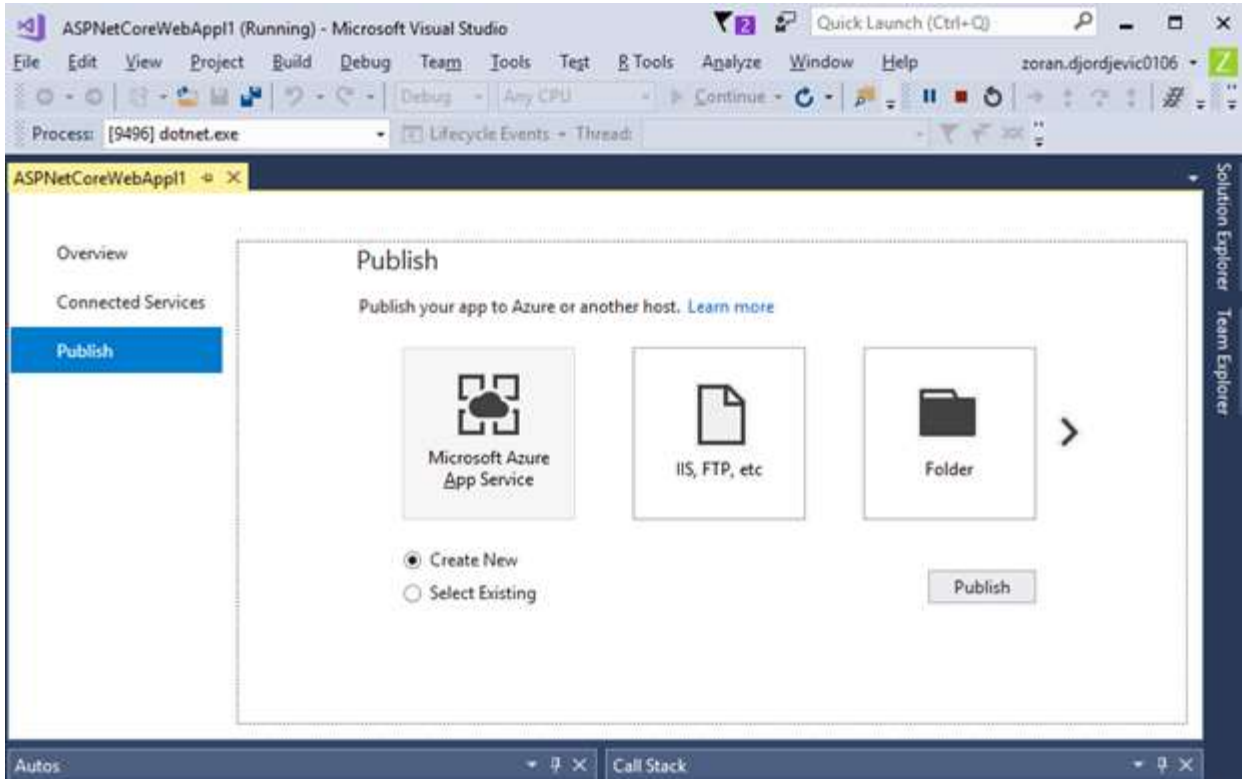
Deploy the App in Azure

- Close the web page, return to Visual Studio. (Had you started Debugging by any chance, **Stop Debugging** from the **Debug** menu)
- Right-click on the project `ASPNETCore…` in Solution Explorer and click **Publish...**.



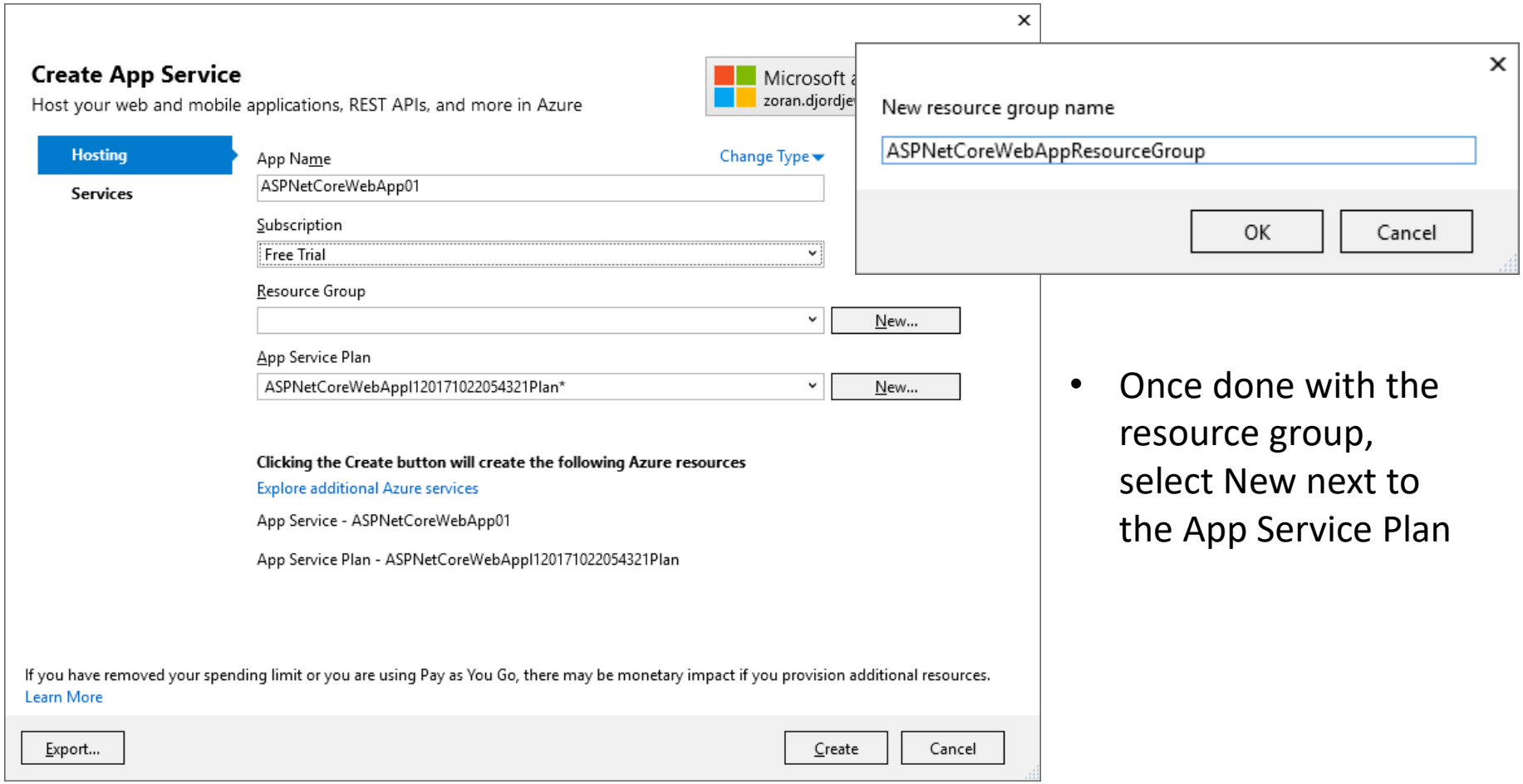
Publish Dialog

- In the Publish dialogue select `Microsoft Azure App Service` and click `Publish`



Create App Service wizard

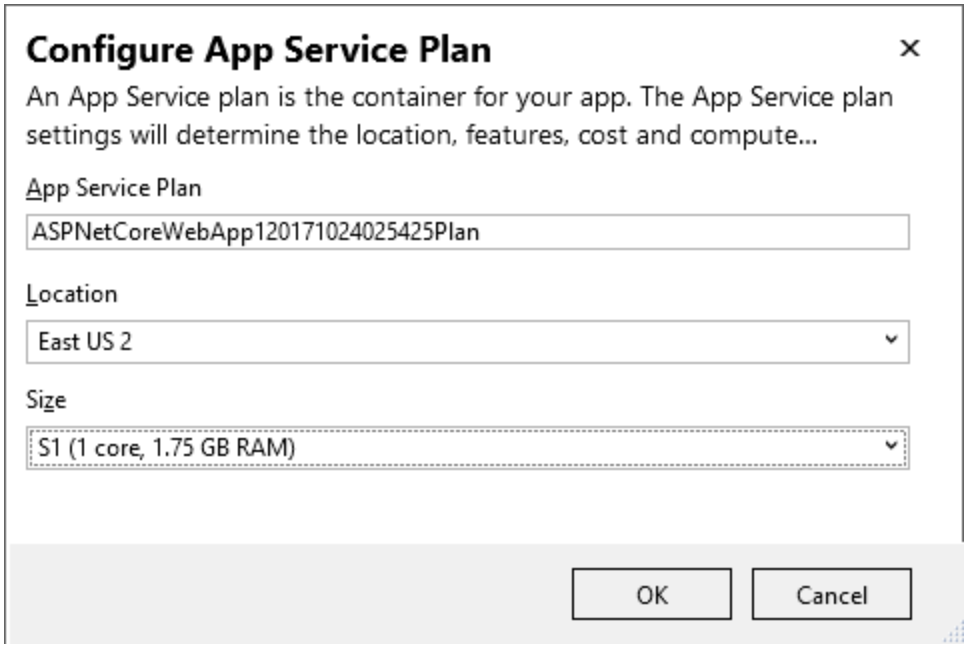
- Provide a unique App Name, e.g. ASPNetCoreWebApp01
- Select a subscription, e.g. Free Trial
- Select New next to the Resource Group field and enter a new name. Hit OK.



- Once done with the resource group, select New next to the App Service Plan

New App Service Plan

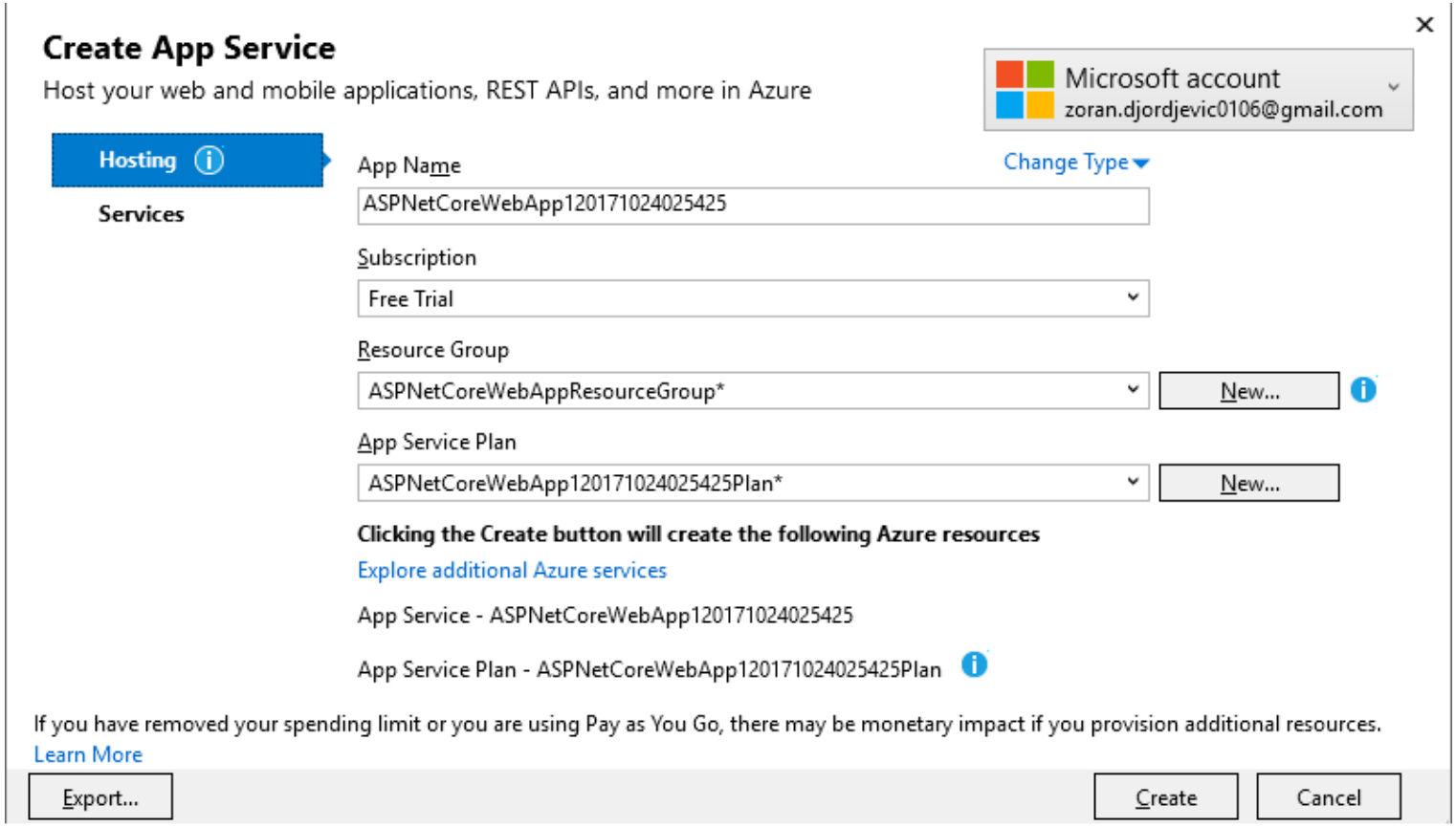
- Select **New...** for the App Service Plan and select a location near you. Location is important.
- Latency of your app is smaller if it is hosted in a data (Cloud) center close to you or your customers.
- You can keep the name that is generated by the system.
- You can choose the size of the machine hosting your App if important.
- Click OK.
- You will return to Create App Service wizard.



- App Service Plan could contain many services and apps and could allow you to move those together if you need to, perhaps to another location.

Create App Service wizard

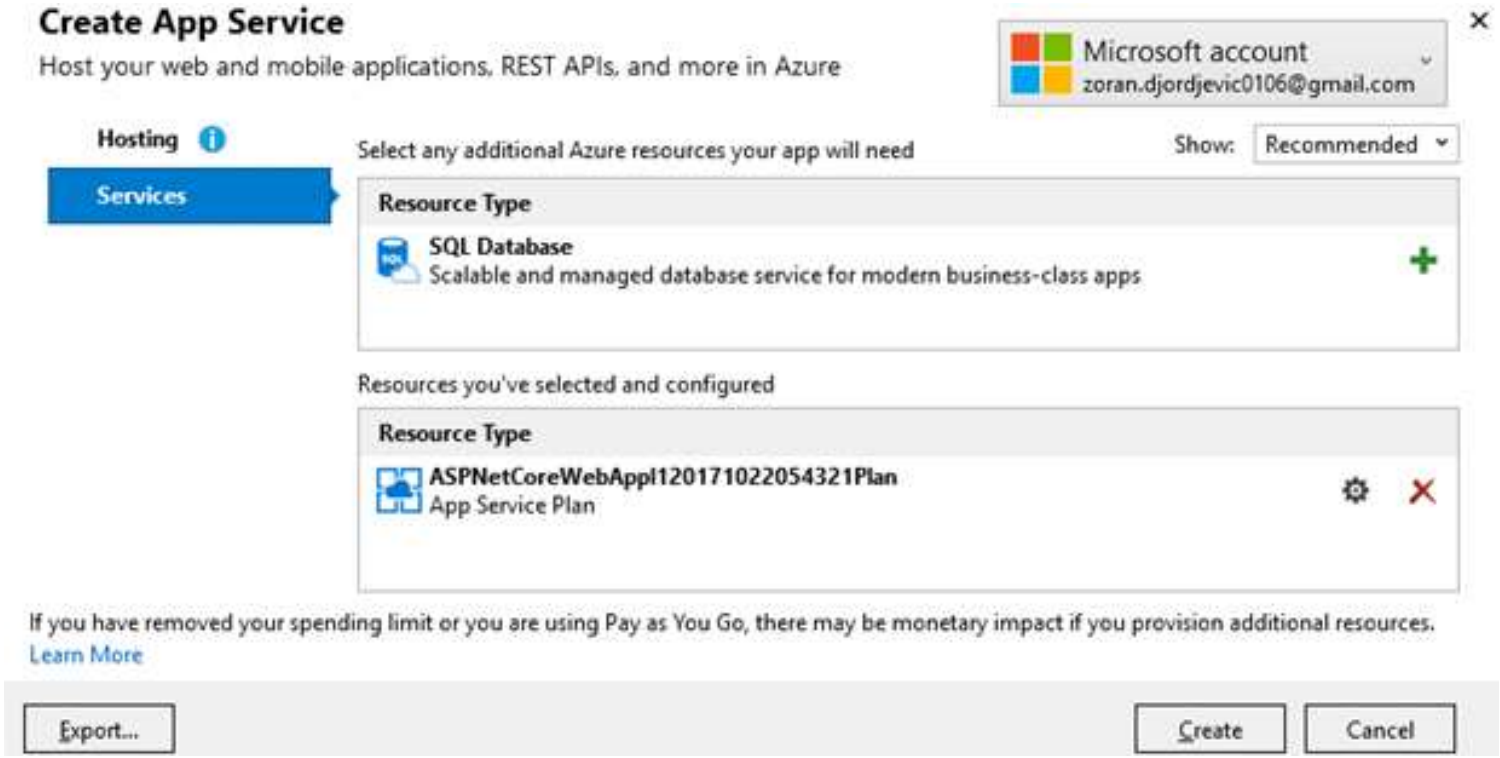
- This is the state of Create App Service wizard after adding resource group and App Service Plan.



- We can not hit `Create` yet. We need database Services.
- Select `Services` tab (below Hosting) to create a new database

After Selecting Services tab

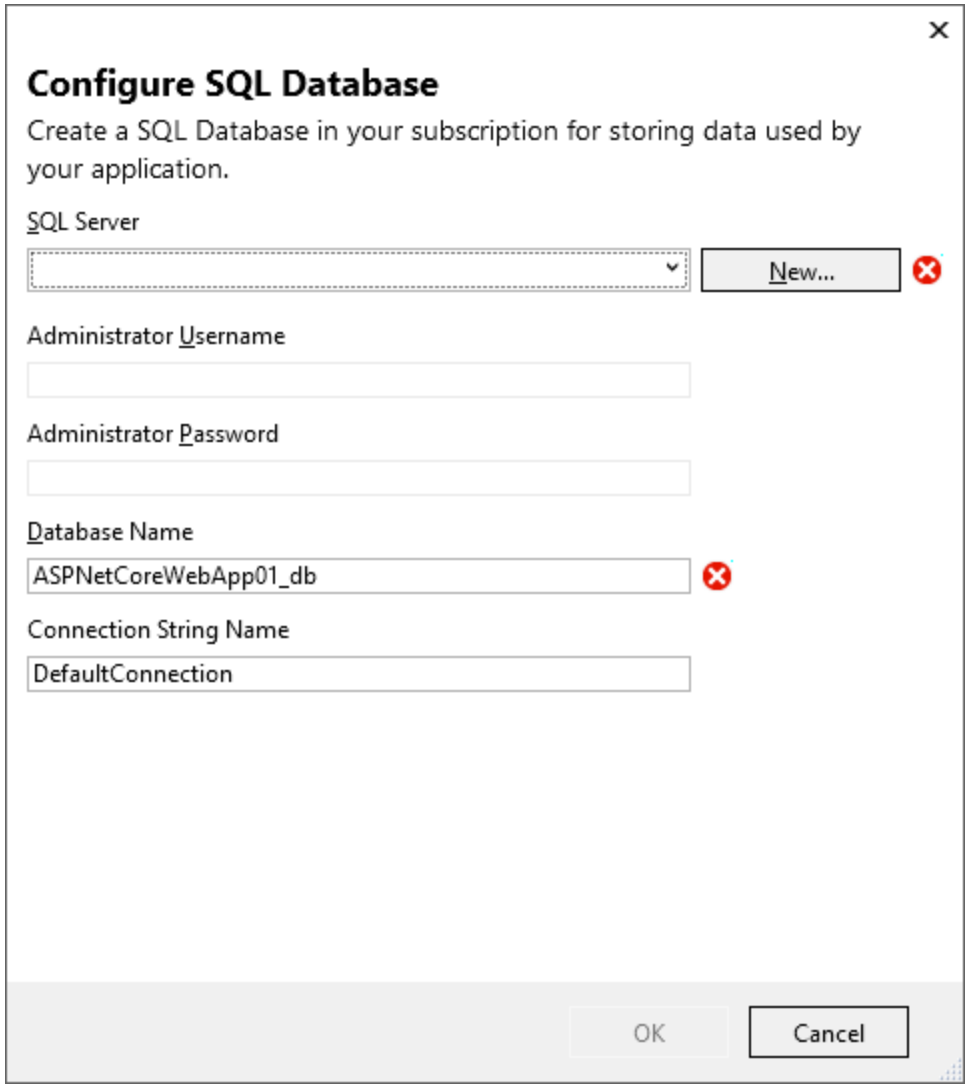
- Additional resources and offered to be added to the system. Click on Recommended and select All. Go back to Recommended.
- Then Select the green + icon to create a new SQL Database



- `Configure SQL Database` wizard appears

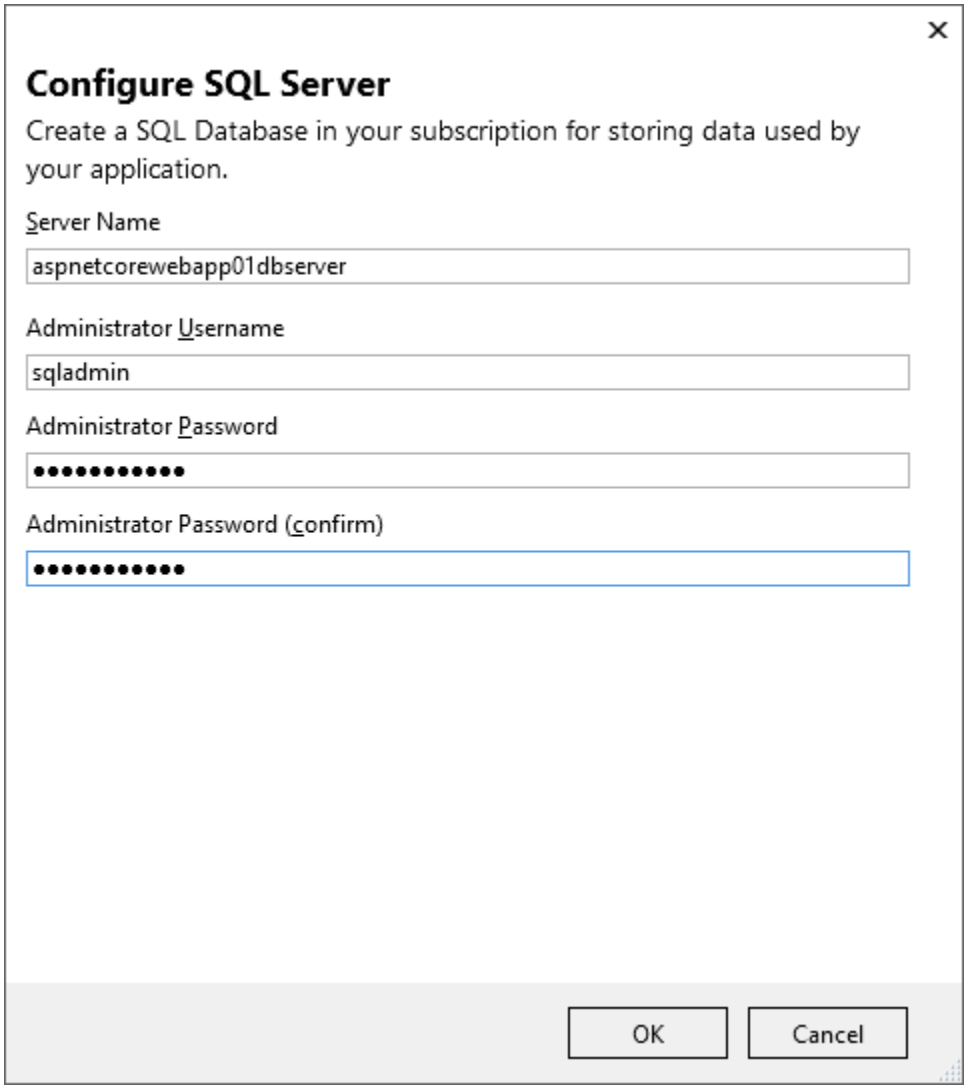
Configure SQL Database

- Next to SQL Server field, hit New.
- Configure SQL Server wizard appears.



Configure SQL Server

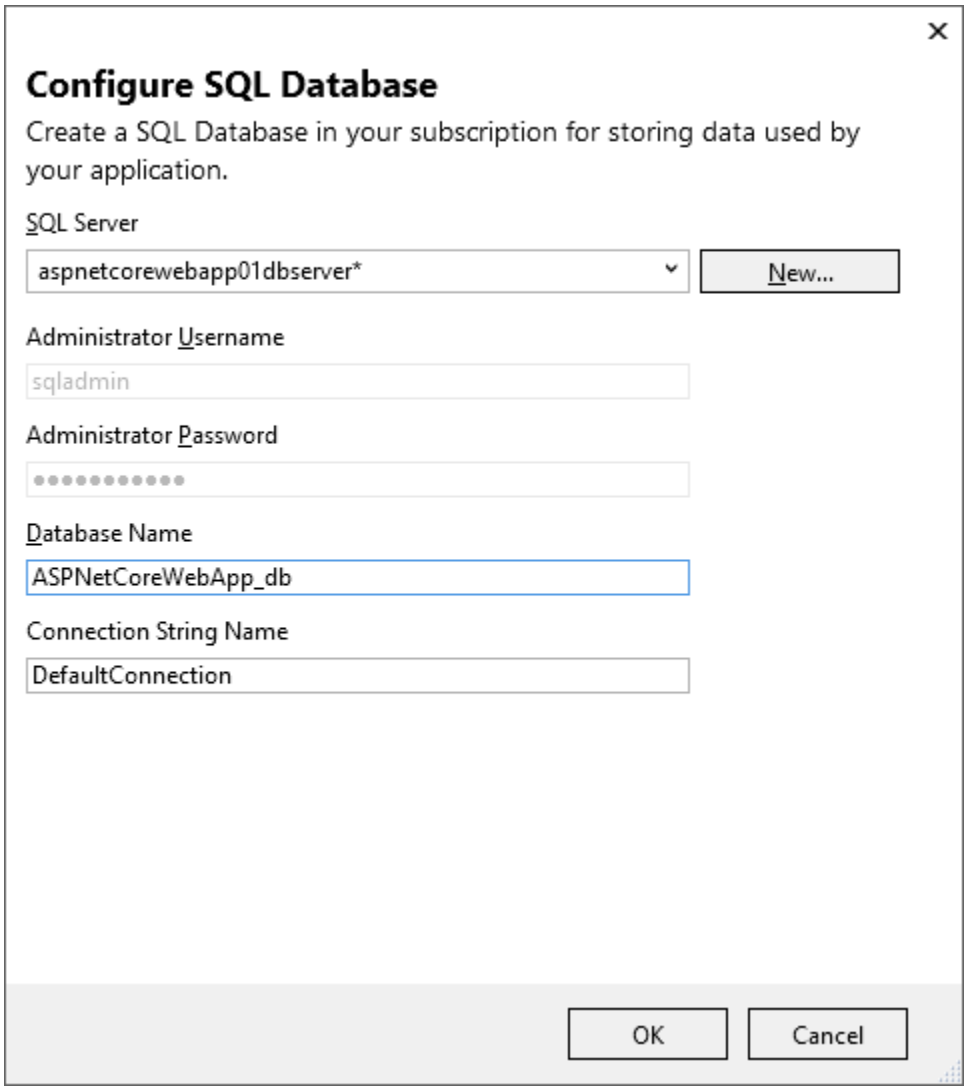
- Enter an administrator user name and password.
- Azure is serious about security. You cannot enter sqladmin123 as a password. Then select **OK**.
- Don't forget the user name and password you created in this step.
- Keep the default **Server Name**.
- Hit OK
- You will be back on Configure SQL Database wizard.



Configure SQL Database

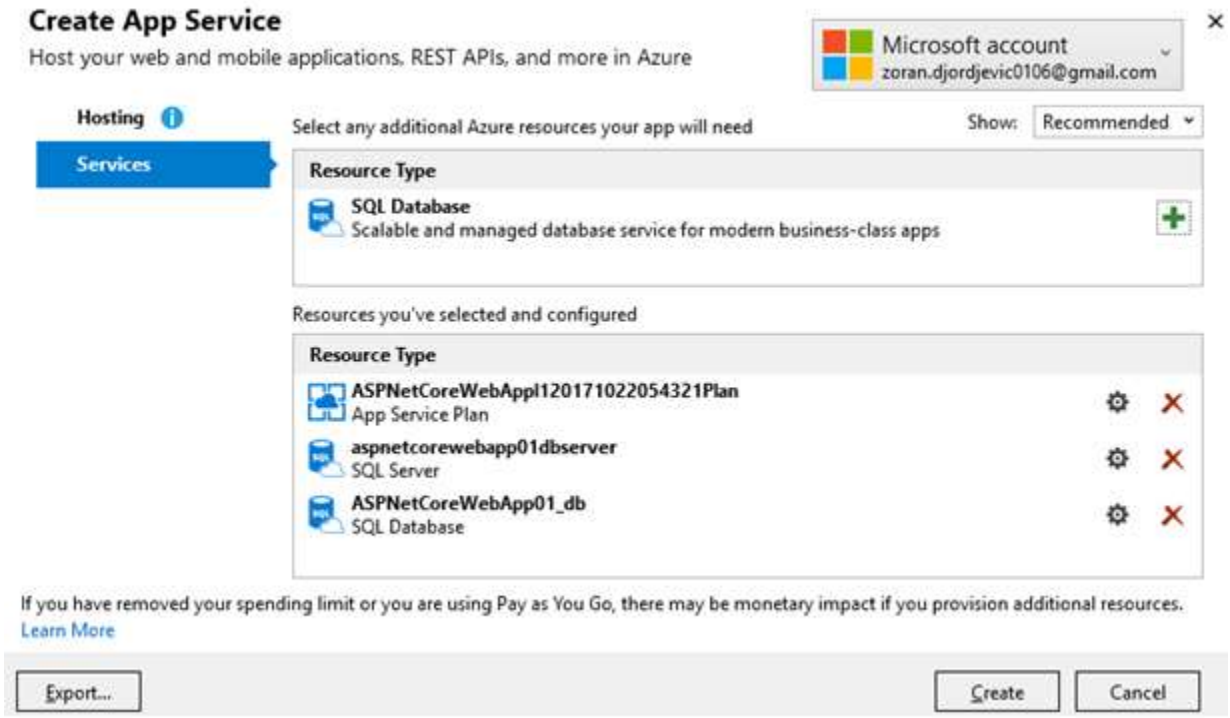
- On the Configure SQL Database you can keep provided DB name or choose your own.
- Keep the Connection string as `DefaultConnection`.
- Hit OK

- Visual Studio returns to the **Create App Service** dialog



Create App Service

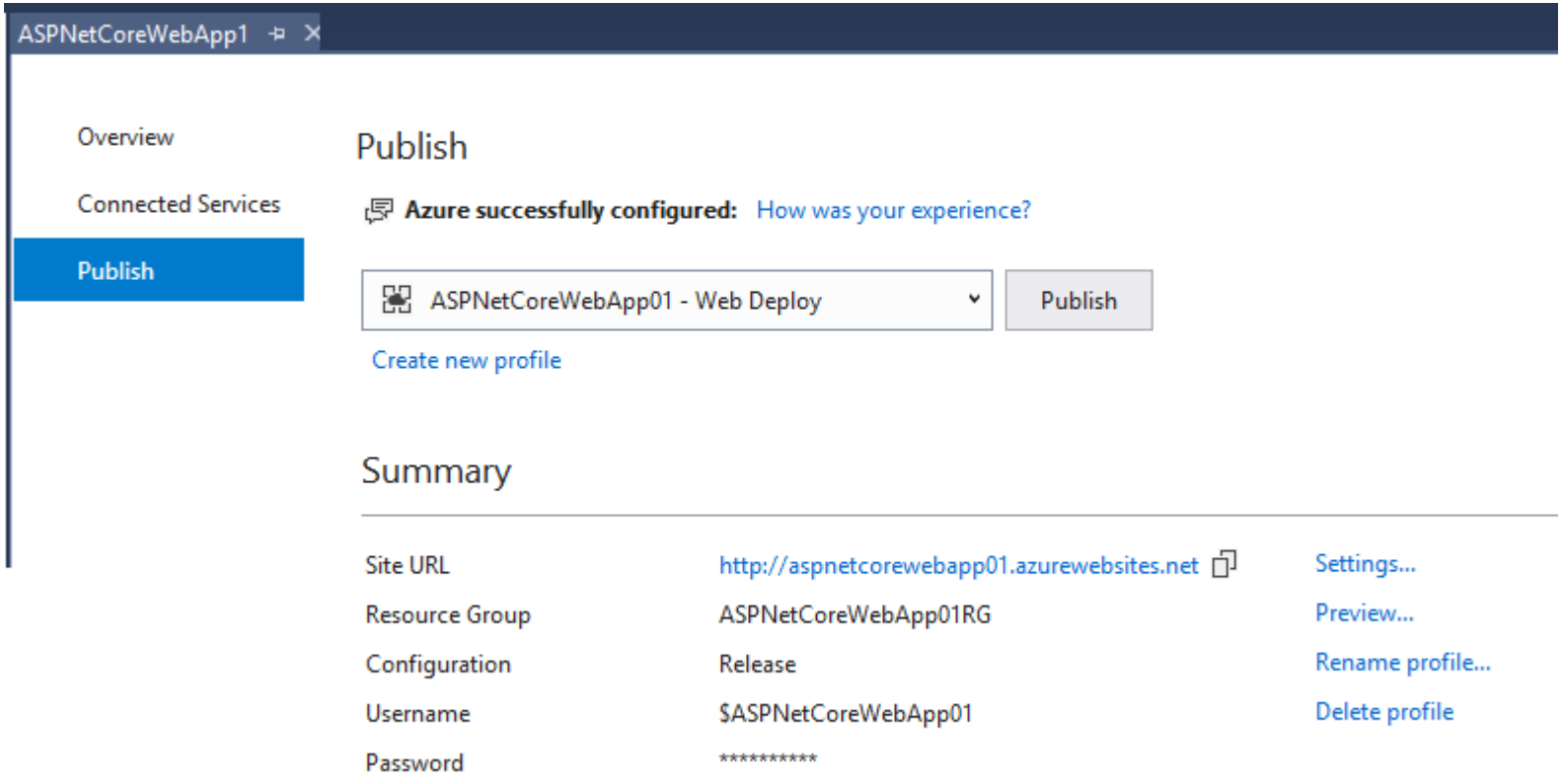
- Back on the Create App Service wizard. Now you can hit `Create`.



- If you changed your mind, you can remove a resource here. Just hit x next to it
- Deployment goes through 6 steps. Just wait. Once the deployment is finished, the Publish wizard reappears and a new web page pops up, this time on ab Azure server. You can examine the URL and you will see this is not your local IIS.

Publish Wizard

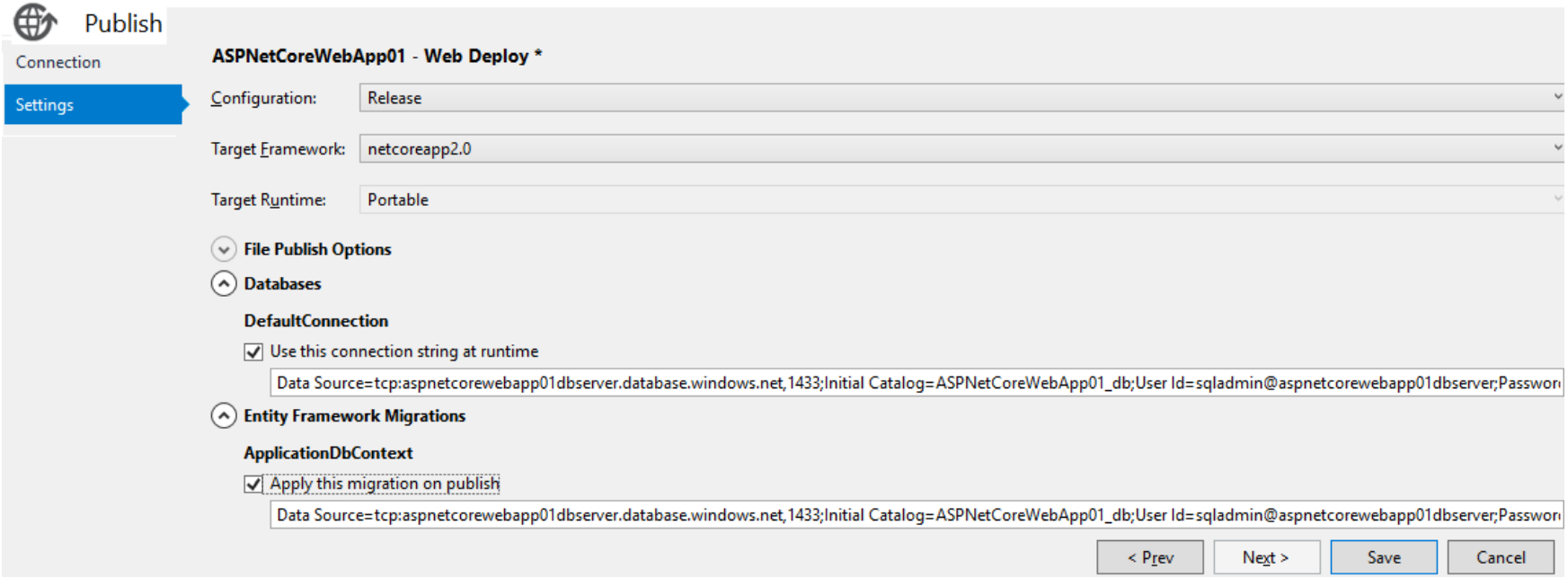
- You are back on the Publish wizard.



- VS will actually Publish your app to Azure. Wait until process finished and then select `Settings` in the continuation of Site URL.
- On the following page, also select `Settings` on the Left.

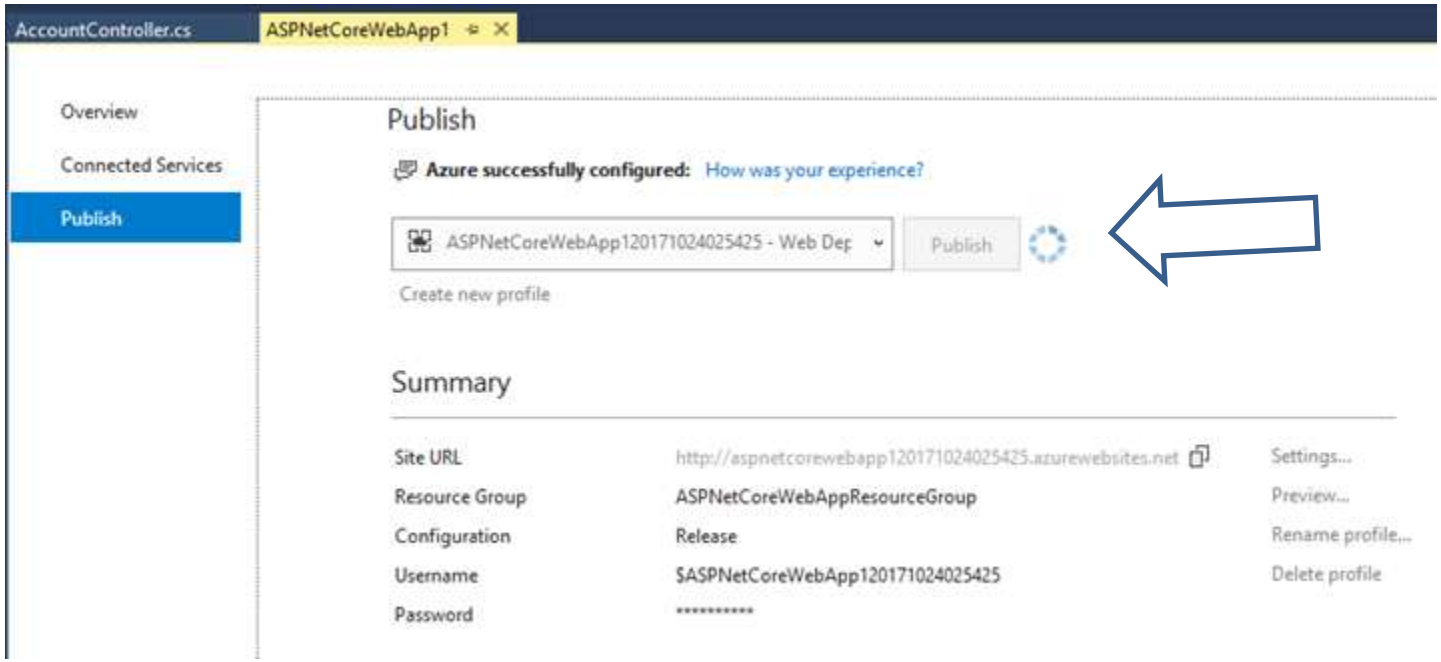
Publish Web Deploy

- Select Settings (bellow Connection). Then expand Databases
- Checkmark `"DefaultConnection" Use this connection string at runtime.`
- Expand Entity Framework Migration and check `"Apply this migration on publish".`
- Hit `Save`. Visual Studio returns to Publish wizard.



Publish

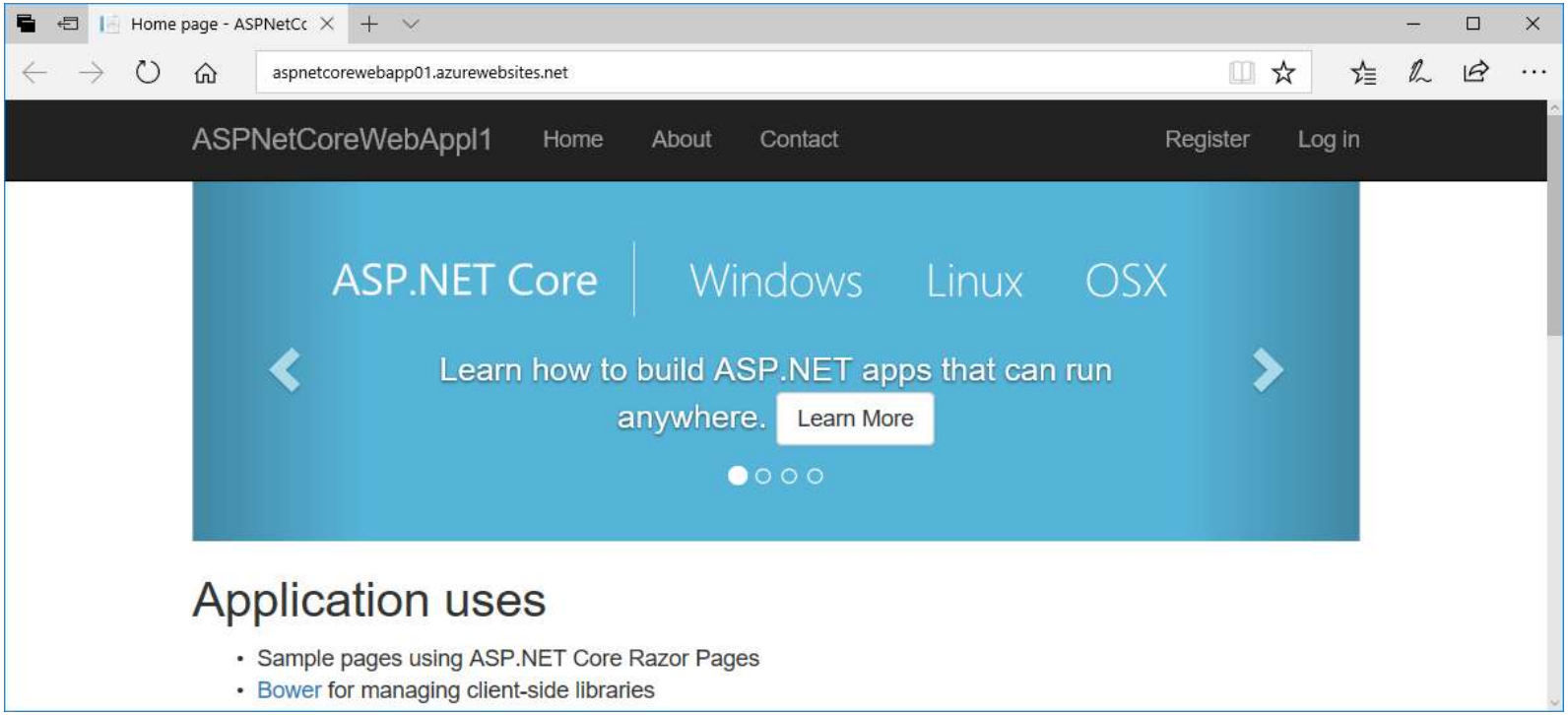
- On Publish wizard, hit `Publish`.



- Visual Studio will publish your app to Azure (again) and launch the cloud app in your browser.
- This time I got an Error telling me that my dll on the Azure server is locked.
- I opened by Azure Portal. Located App Service `AspNetCoreWebApp01` and restarted it. Presumably that removed the lock.
- I went back to my project and Published it again. This time I just verified Settings.

Browser with Azure Address

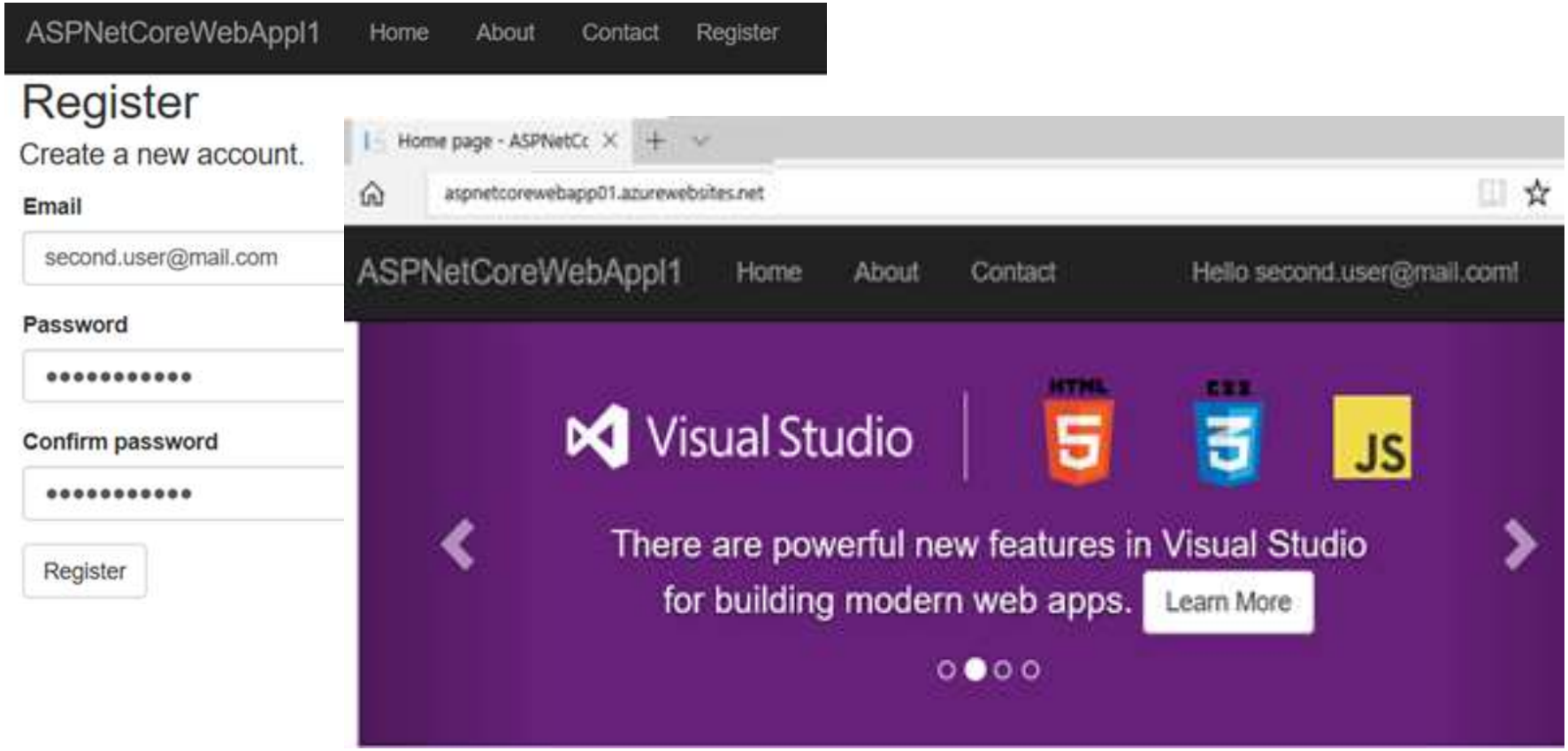
- Notice that browser now has an Azure Address:



- We could test the **About** and **Contact** links and **Register** new user

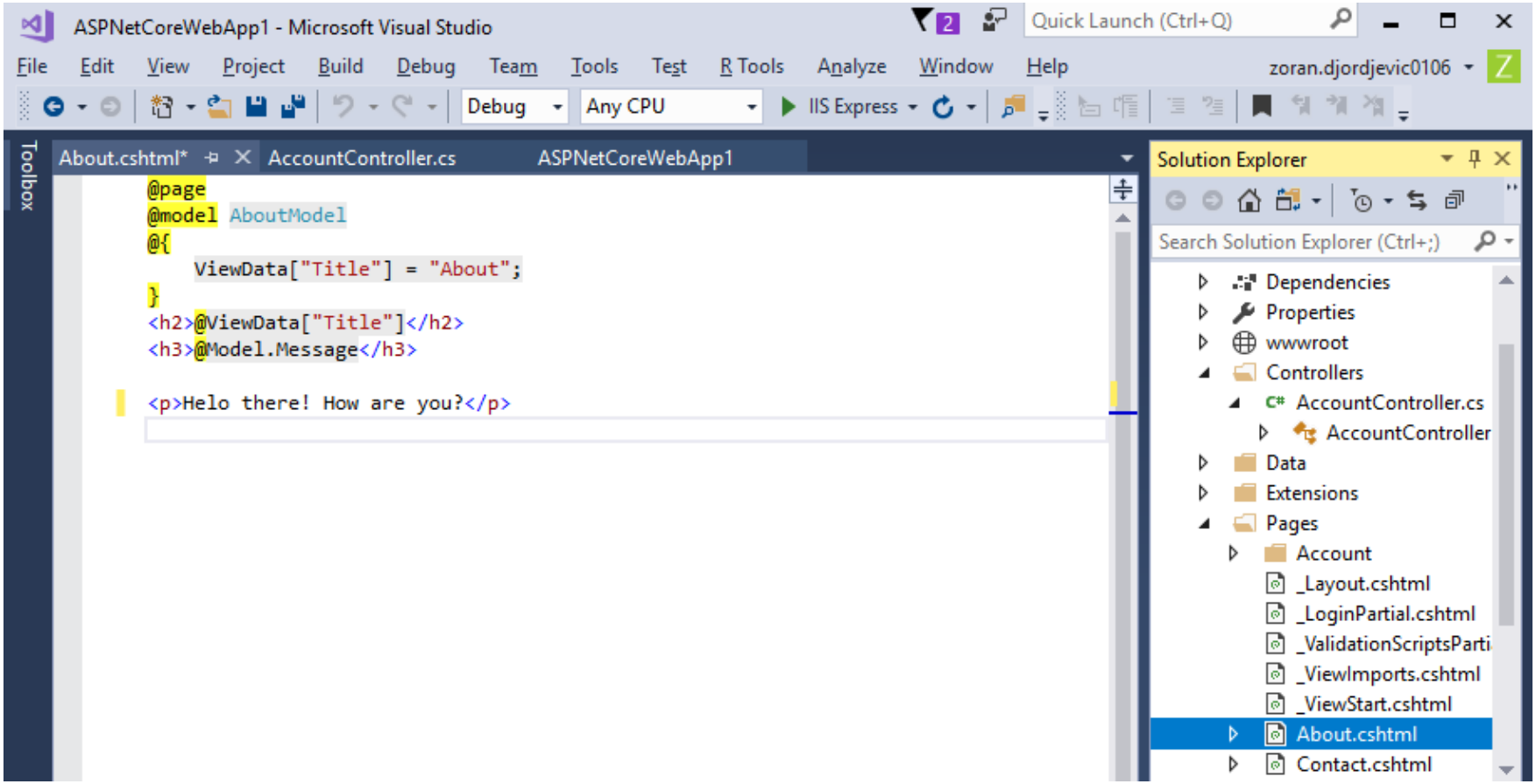
User is Registered, without an Error

- New user email: second.email@gmail.com appears in the browser.



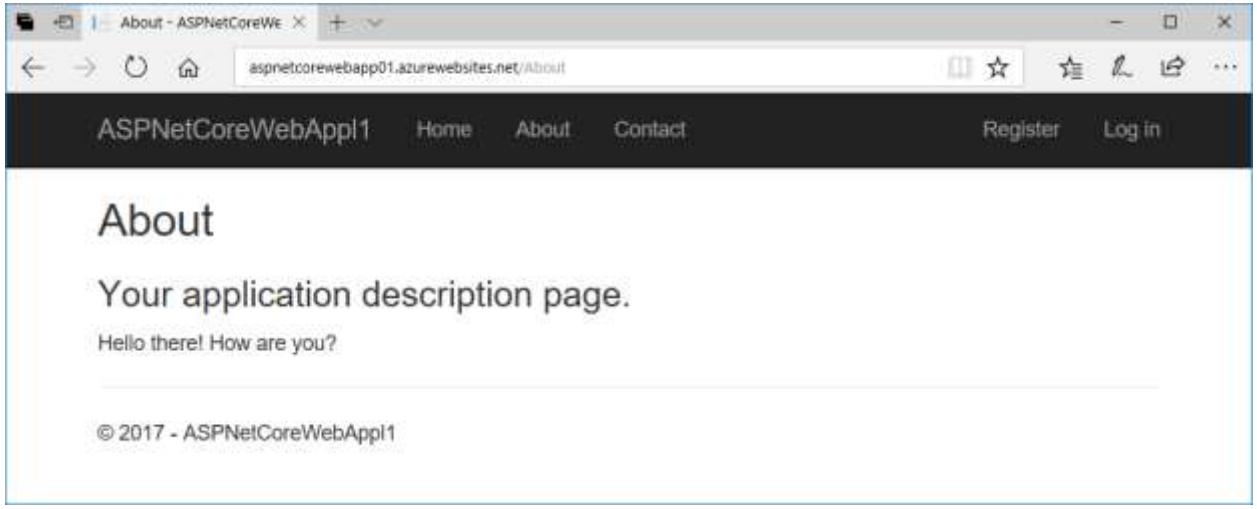
Update the App

- We still have application code. Open Solution explorer. Edit *`Pages/About.cshtml`* Razor page and change its contents.
- We could modify the paragraph to say "`Hello There! How are you?`":



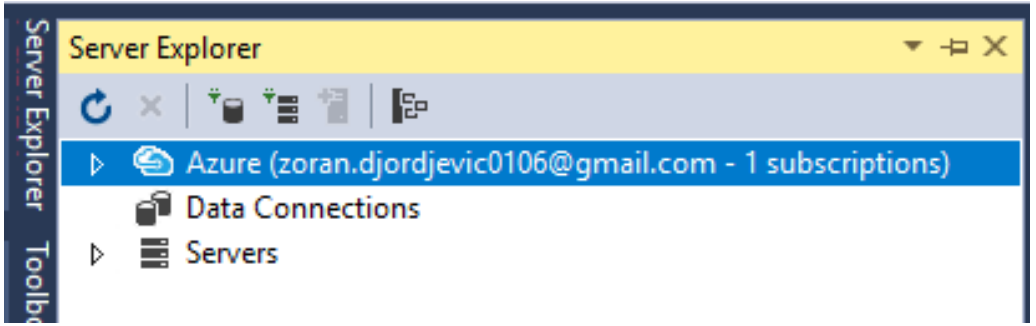
Save and Publish

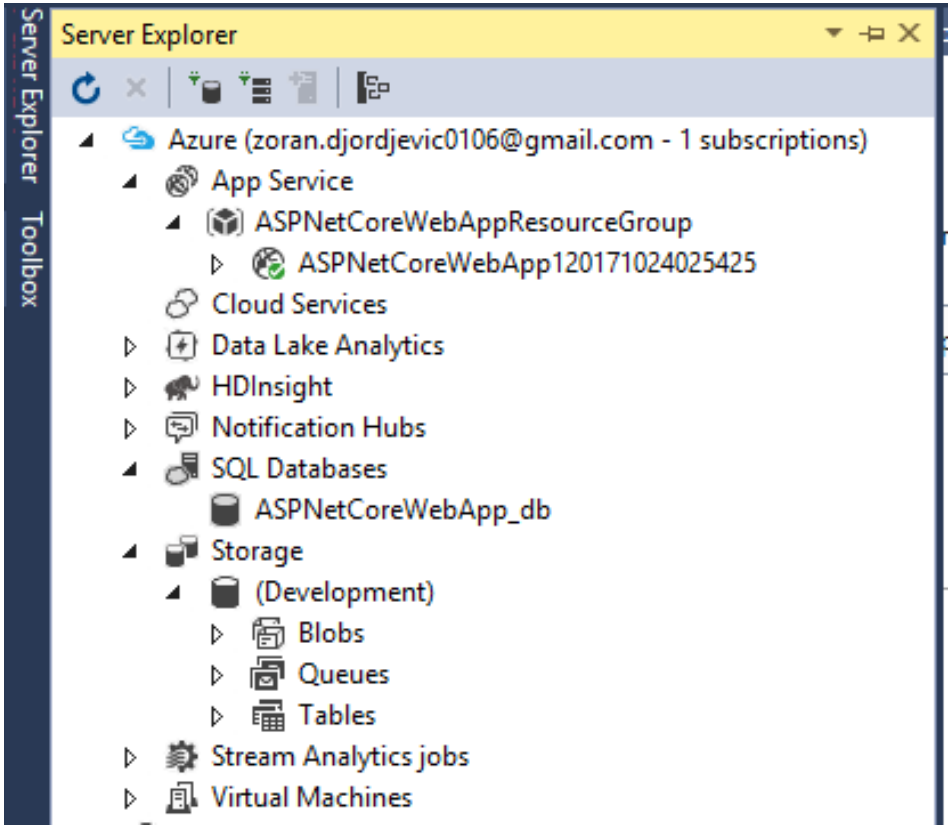
- Once done with editing the page, we can do `File -> Save About.cshtml` or just hit little floppy disk icon on the top menu. Then we go back to Solution Explorer, right click on the name of the project and select `Publish.` On the Publish wizard, we again hit `Publish.`
- New page appears in the browser. On new page, select `About` link
- Modified text should appear: `Hello there! How are you?`



Look into Server Explorer

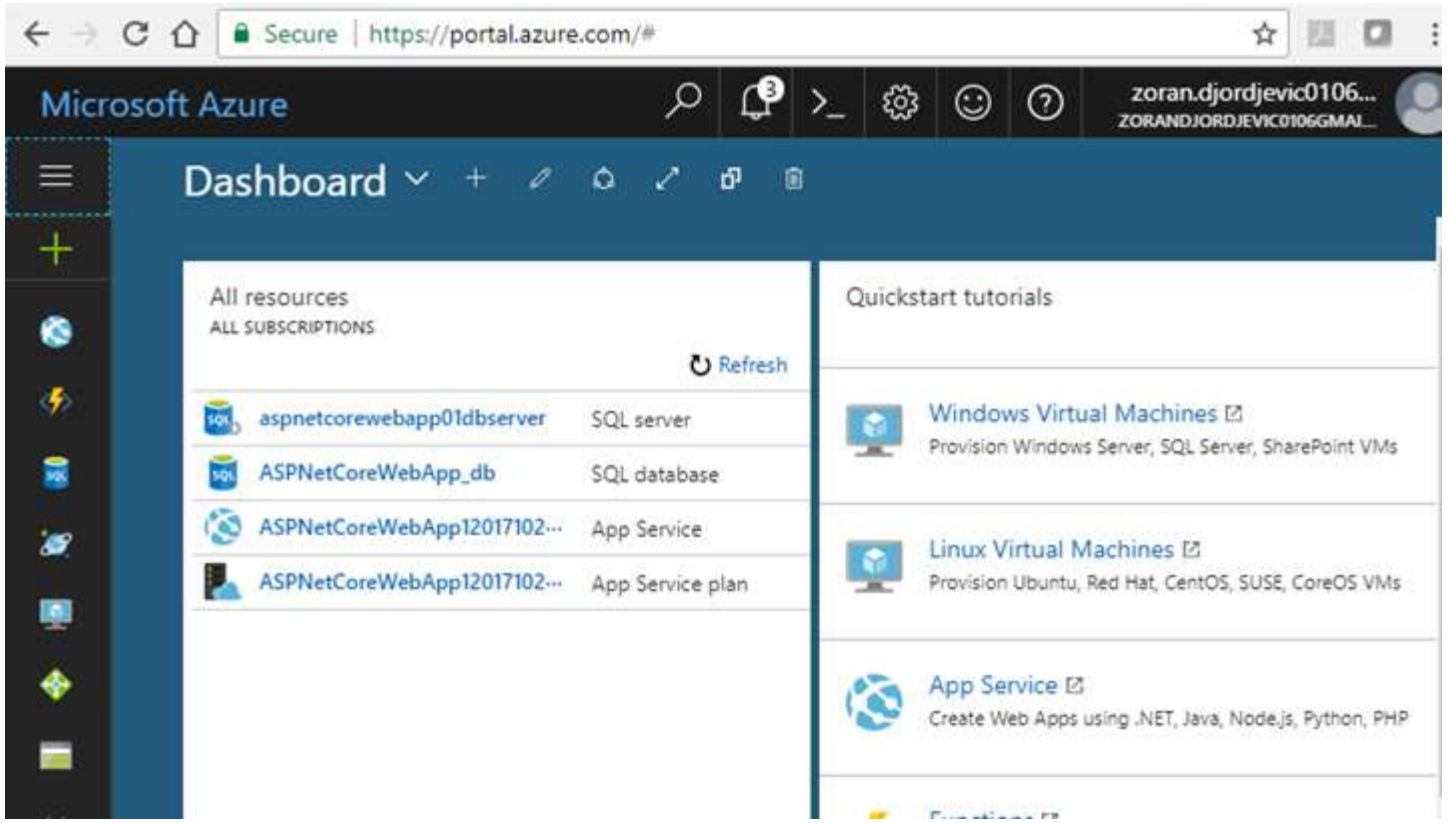
- If we go to View -> Server Explorer we will see: Azure and Date Connections.
- If we expand SQL Database, we will see or ASPNetCoreWebApp_db
- Other Azure resources we posses are also visible.





Opening Firewall in Azure Portal

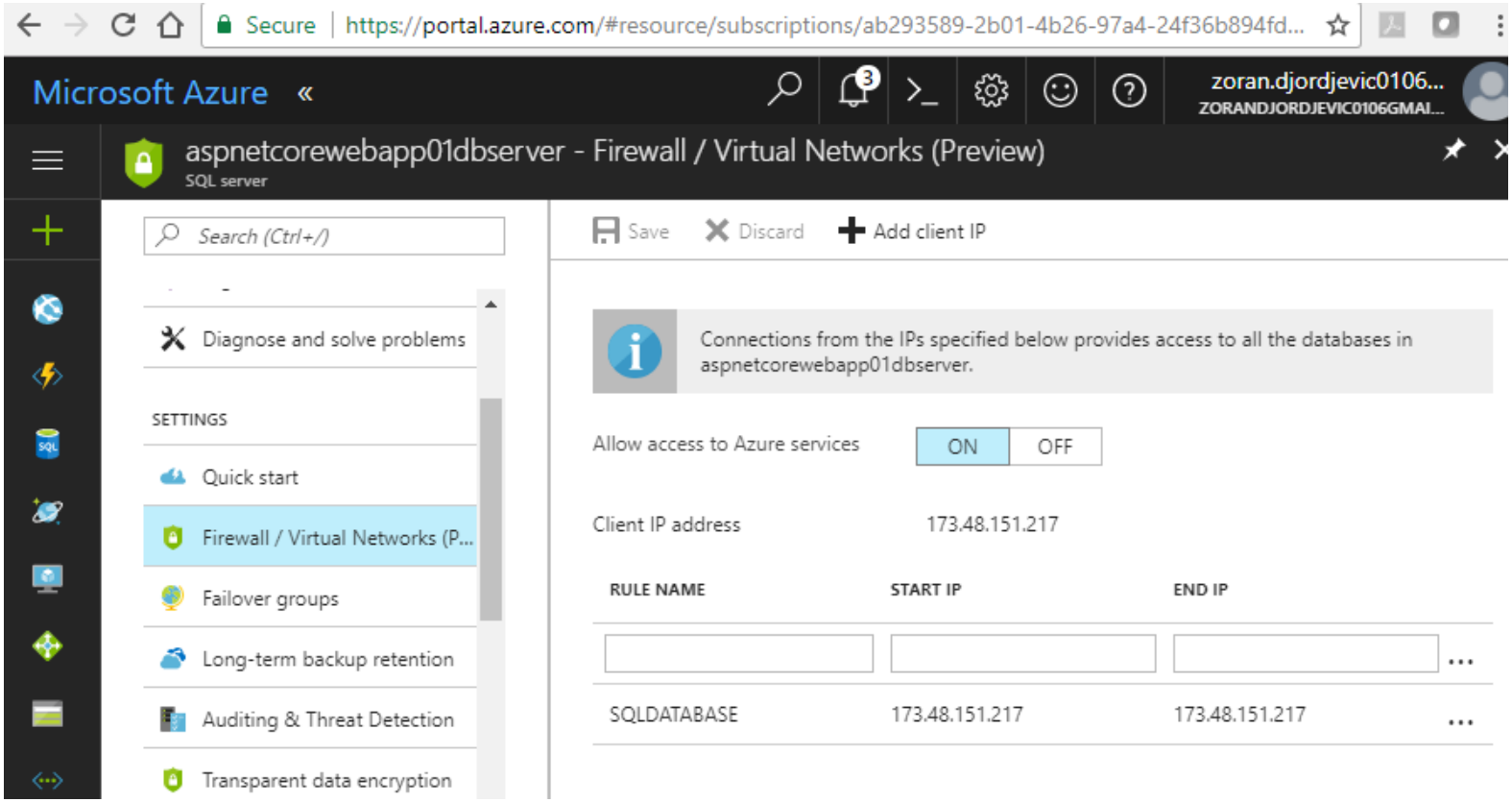
- Azure servers are protected by firewalls that only allow connections from approved ranges of IP addresses.
- In Azure Portal's Dashboard, double click on our SQL Server, `aspnetcorewebapp01dbserver`



- Detailed page with server properties appears .

Open Azure SQL Server Firewall

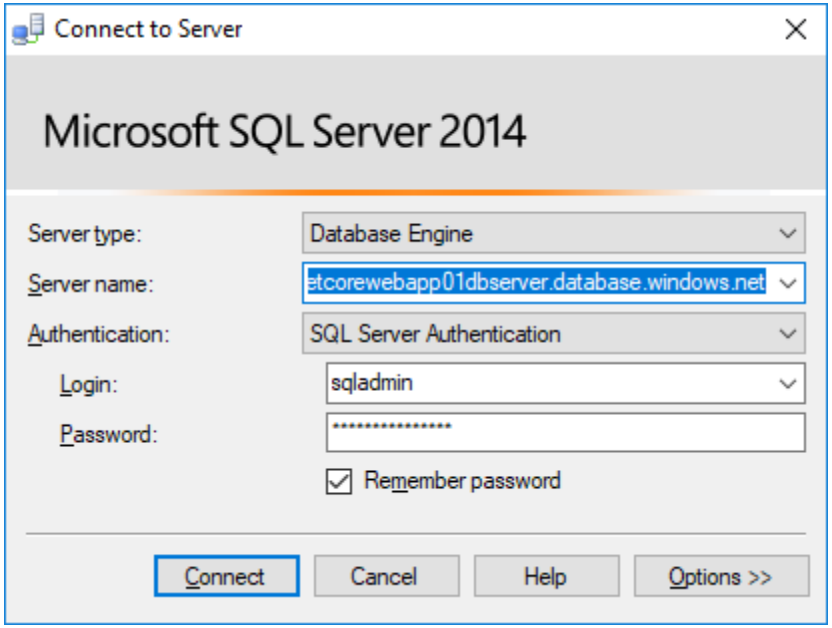
- On the left navigation pane select Firewall/Virtual network. On the right side, turn Allow access to Azure service ON. Copy your machine's IP address in both START IP and END IP field, or provide a meaningful range of IP addresses. Hit Save.



SQL Server Management Studio

- Let us go to the installed applications and select Microsoft SQL Server Management Tools 2017 -> Microsoft SQL Server Management Studio 2017.
- Connect to Server /Login Window appear.
- We need :
  - `Server name`
  - `Authentication (type)`
  - `Login and`
  - `Pasword`

Switch from `Windows Authentication` to `SQL Server Authentication`

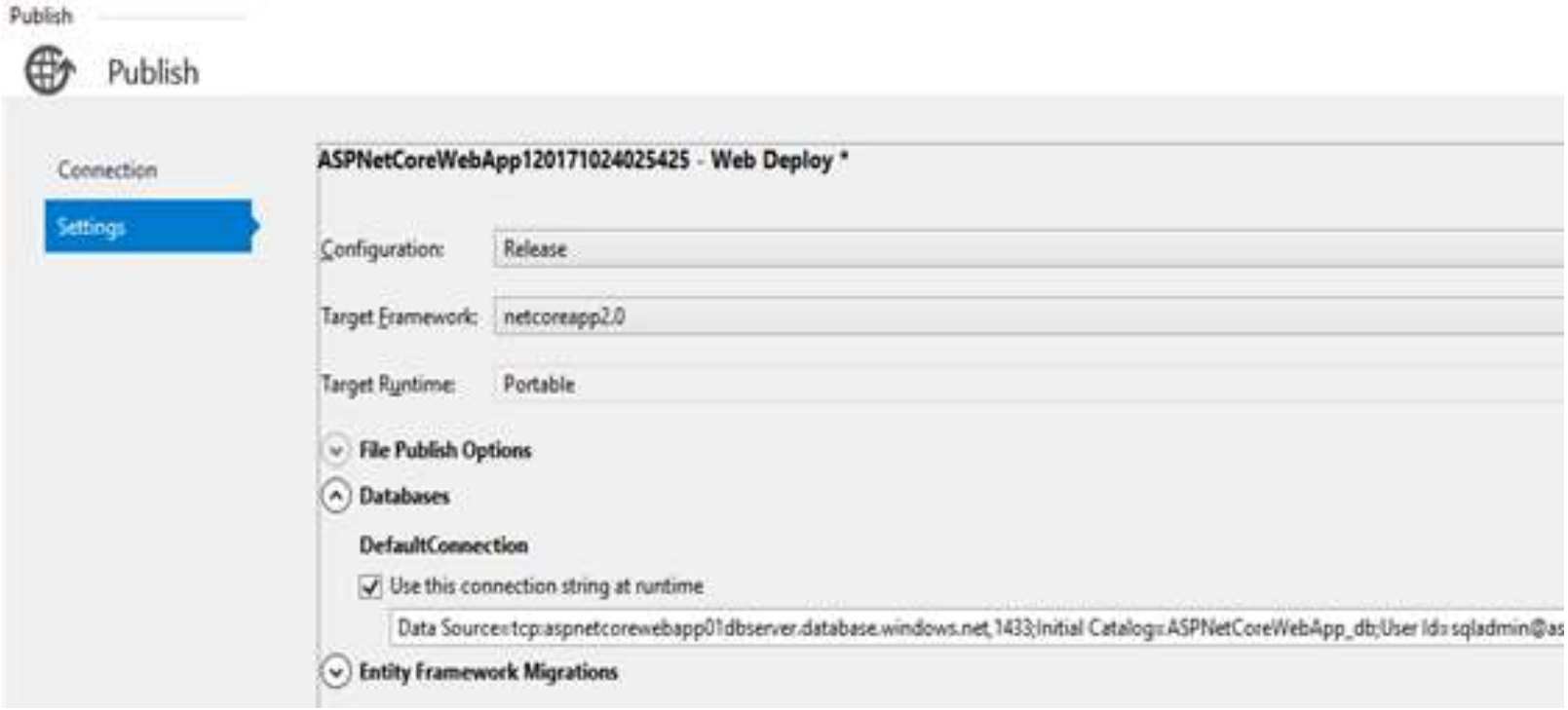


Server name we can find in the

Publish wizard -> Settings

Server Name of Azure SQL Server

- We right click on Project name, select Publish, than on Publish wizard we select Settings. On new Publish screen, we again select Settings and then expand Databases. The connection string and username and password are all there.



In my case:

Server name: `aspnetcorewebapp01dbserver.database.windows.net`

User: `sqladmin`, Password: `xxxxxxxxxxx`

We transfer those to SSMS Login and we are in.

Expand Databases

- Expand our database ASPNetCoreWebApp_db. Expand Tables.
- The table that might have our user is probably AspnetUsers. Open New Query editor and type:

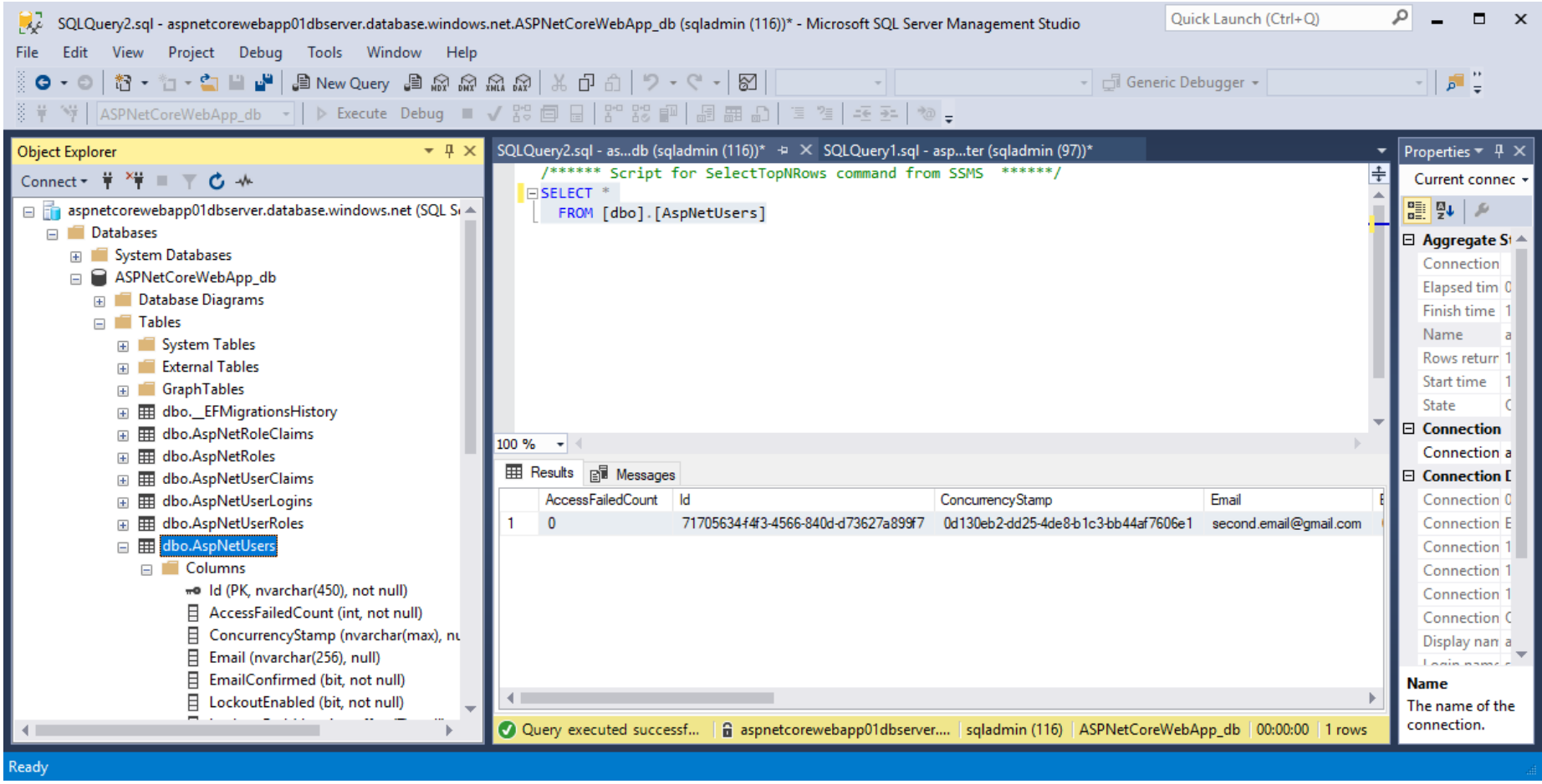
```
SELECT *
  FROM [dbo].[AspNetUsers]
```

In the results window we will get something like:

```
AccessFailedCount      Id      ConcurrencyStamp       Email
       EmailConfirmed LockoutEnabled LockoutEnd
       NormalizedEmail        NormalizedUserName
0      71705634-f4f3-4566-840d-d73627a899f7  0d130eb2-dd25-4de8-
b1c3-bb44af7606e1      second.email@gmail.com 0       1       NULL
       SECOND.EMAIL@GMAIL.COM SECOND.EMAIL@GMAIL.COM
```

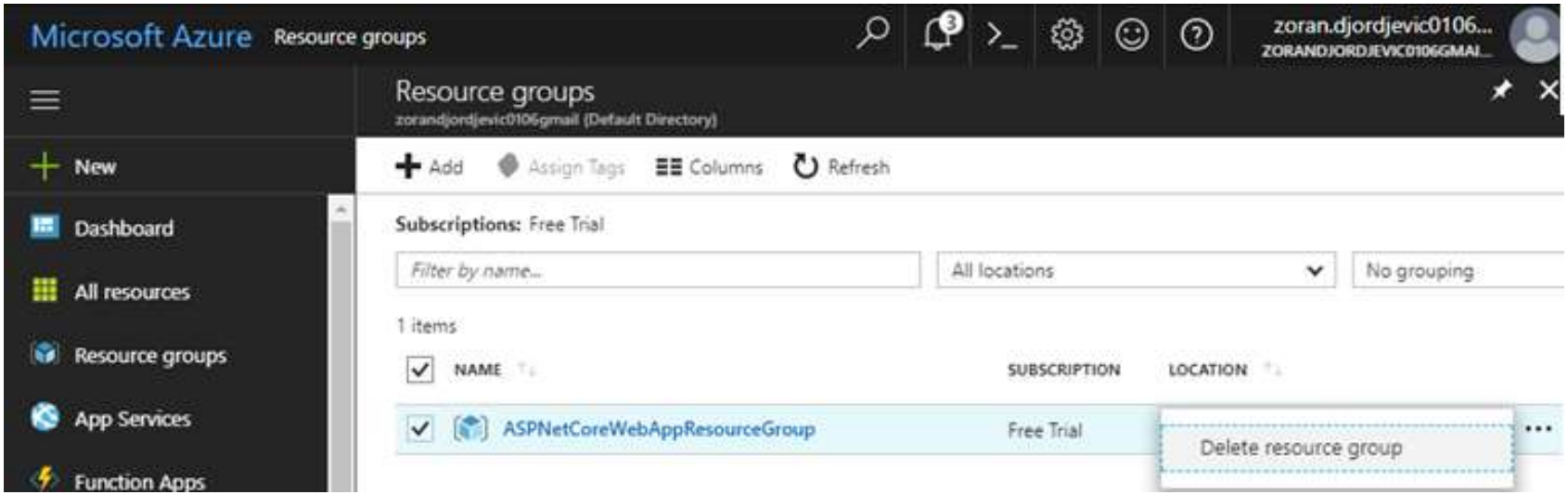
SQL Server Management Studio

- View of Azure database and Tables in SSMS



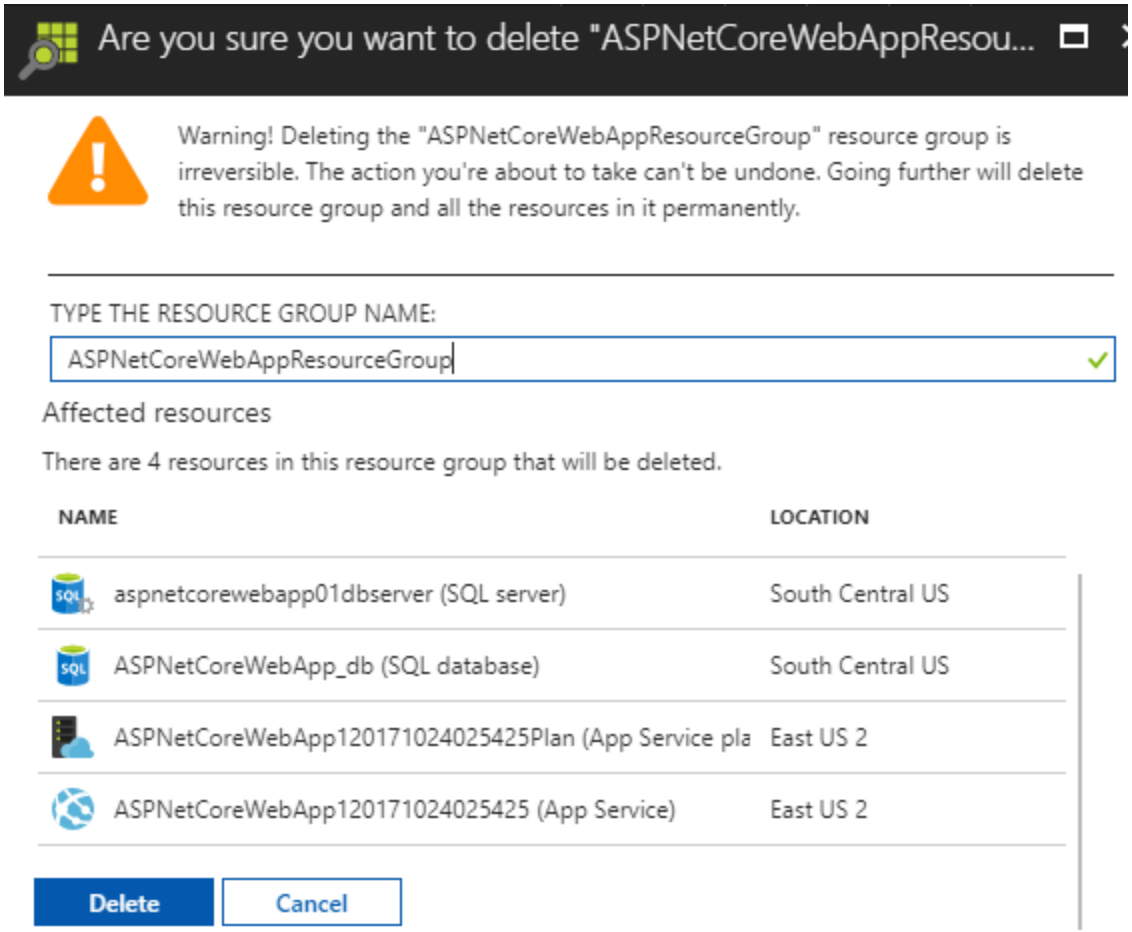
Cleanup

- Select your resource group. Mine is `ASPNetCoreWebAppResourceGroup`.
- Go to the ellipsis (. . .) at the end of the line and hit `Delete resource group`



Confirm Delete

- Azure does not trust you a bit.
- You have to retype that silly long name.
- Hit Delete again.



Check whether you are done

- It might appear that some resources like SQL Database are still there.
- Trying to delete them will not work. After a few minutes you should get a clean screen